Master's degree thesis

LOG950 Logistics

Developing a heuristic algorithm for classification of problems with binary attributes

Anna Konovalenko

Number of pages including this page: 76

Molde, 24.05.2019

Molde University College
Specialized University in Logistics

Mandatory statement

Each student is responsible for complying with rules and regulations that relate to examinations and to academic work in general. The purpose of the mandatory statement is to make students aware of their responsibility and the consequences of cheating. Failure to complete the statement does not excuse students from their responsibility.

*Please complete the mandatory statement by placing a mark **in each box** for statements 1-6 below.*		
1.	**I/we hereby declare that my/our paper/assignment is my/our own work, and that I/we have not used other sources or received other help than mentioned in the paper/assignment.**	☒
2.	**I/we hereby declare that this paper** 1. Has not been used in any other exam at another department/university/university college 2. Is not referring to the work of others without acknowledgement 3. Is not referring to my/our previous work without acknowledgement 4. Has acknowledged all sources of literature in the text and in the list of references 5. Is not a copy, duplicate or transcript of other work	Mark each box: 1. ☒ 2. ☒ 3. ☒ 4. ☒ 5. ☒
3.	**I am/we are aware that any breach of the above will be considered as cheating, and may result in annulment of the examination and exclusion from all universities and university colleges in Norway for up to one year, according to the Act relating to Norwegian Universities and University Colleges, section 4-7 and 4-8 and Examination regulations** section 14 and 15.	☒
4.	**I am/we are aware that all papers/assignments may be checked for plagiarism by a software assisted plagiarism check**	☒
5.	**I am/we are aware that Molde University College will handle all cases of suspected cheating according to prevailing guidelines.**	☒
6.	**I/we are aware of the University College's rules and regulation for using sources**	☒

Personal protection

Personal Data Act

Research projects that processes personal data according to Personal Data Act, should be notified to Data Protection Services (NSD) for consideration.

Have the research project been considered by NSD? ☐**yes** ☒**no**

- If yes:

Reference number:

- If no:

I/we hereby declare that the thesis does not contain personal data according to Personal Data Act.: ☒

Act on Medical and Health Research

If the research project is effected by the regulations decided in Act on Medical and Health Research (the Health Research Act), it must be approved in advance by the Regional Committee for Medical and Health Research Ethic (REK) in your region.

Has the research project been considered by REK? ☐**yes** ☒**no**

- If yes:

Reference number:

Publication agreement

ECTS credits: 30

Supervisor: Lars Magnus Hvattum

Agreement on electronic publication of master thesis

Author(s) have copyright to the thesis, including the exclusive right to publish the document (The Copyright Act §2).

All theses fulfilling the requirements will be registered and published in Brage HiM, with the approval of the author(s).

Theses with a confidentiality agreement will not be published.

I/we hereby give Molde University College the right to, free of charge, make the thesis available for electronic publication: ☒yes ☐no

Is there an agreement of confidentiality? ☐yes ☒no

(A supplementary confidentiality agreement must be filled in)

- If yes:

Can the thesis be online published when the period of confidentiality is expired? ☐yes ☐no

Date: 24.05.2019

Preface

This thesis is a final work as partial fulfillment for the degree of Master of Science in Logistics Analytics in Molde University College. The thesis is researched and written from December 2018 to May 2019. The work on this thesis has been challenging, but, at the same time, interesting and exciting for me.

I would like to thank Lars Magnus Hvattum, who supported me at each step of my thesis, gave a lot of valuable comments and was very patient during whole semester.

At the same time, I would like to thank my parents for their care and belief in me.

Summary

This thesis focuses on analyzing and applying new techniques for the Satisfiability Data Mining Algorithm (Glover 2008), a method for classification of problems with binary attributes, and describes possible techniques that can be added to the SAT-DM algorithm to improve the results of classification. In particular, this thesis shows how a strategic oscillation approach, which was created to solve optimization problems, can be applied to the SAT-DM method, and shows how using Pareto layers of solutions can influence the accuracy of classification. In addition, this thesis describes how the technique of classification with Naïve Bayes classifiers can be used with the SAT-DM algorithm. The ideas presented are implemented and validated with real-world data sets. The results of this research encourage further research into the SAT-DM algorithm.

Contents

1. Introduction

1.1 Background of the study and research problem

Classification is one of the most broadly applied and important techniques for exploring data, with a wide array of different applications. The core goal of classification is to determine which set of categories/classes a new object belongs to, based on observations whose class is already known. Generally, it is the process of predicting the class of given object. The definition of classification came from Machine Learning, a field of study where the objective is to develop techniques that allow computers to learn from data and make decisions without human assistance.

The volume of data is practically exploding by the year. Therefore, there is a growing need for exploring data. A large proportion of the data is unstructured and classification is a vital concern nowadays. Different methods have already been proposed to classify elements. Linear classifiers, Naïve Bayes and logistic regression classifiers, tree-based classifiers, neutral networks, support vector machines, and many other different classification algorithms have been developed (Wu et al. 2008). However, it is difficult or even not possible to say which one is better than any other. Methods have different classification accuracy and CPU (process) time requirements. According to Dogan and Tanrikulu (2013) it depends on the nature of the available data, the type of problems, the implementation techniques, and a wide variety of other factors.

In addition to the existing methods, more sophisticated techniques have to be developed to work with classification. As mentioned above, the quality of classification and characteristics of the algorithm depend on data types. Dougherty, Kohavi and Sahami (1995) found that a separate discretization procedure could improve the performance of classifiers. This led to the idea of creating a classification algorithm for binary data according to laws of Boolean algebra. The first application of this idea was by Peter L. Hammer in his "logical analysis of data" (LAD) approach (Alexe et al. 2007). This approach represents an important body of work in applying logic to data mining, and provides a reference for solving classical satisfiability problems in Boolean algebra.

Subsequently, Glover (2008) introduced a new heuristic method for binary data called "Satisfiability Data Mining" (SAT-DM). Despite its similarity with the LAD approach, it adopts a different perspective, includes advanced guidelines, and is very attractive for investigation.

Preliminary work from the simplest implementation of the SAT-DM method showed compelling outcomes and motivates pursuing the SAT-DM framework further. (Gardeux, Hvattum and Glover 2014)

Therefore, the purpose of this study is to implement new features in the existing SAT-DM method, which may lead to attaining better results than the original classification algorithm. Specifically, this master thesis presents how techniques from solving optimization problems may be applied to SAT-DM classification and how it is possible to involve classification guidelines from the widely used Naïve Bayes classifier into the SAT-DM classifier.

1.2 Significance of the study

Classification has a wide range of applications in different spheres, e.g., sentiment analysis, speech and handwriting recognition, medical diagnosis, document classification, risk assessment, image classification and biometric identification. Classifiers can diagnose patients with a particular disease, classify chemical compounds by their expected behaviors, classify drugs by their efficiency in treatment of certain conditions, determine membership of biological organisms into groups, or classify investments by expected profitability (Schlkopf and Smola 2002).

Moreover, this field of study could be applied in logistics sphere, e.g., classification of transport types, materials in inventory stocks, logistics costs, or types of roads. Retailers, manufacturers, and distributors can apply it to managing the huge number of items they must track. The classification task is quite ubiquitous and necessary for a lot of industries.

1.3 Structure of the thesis

Chapter 2. Background
Provides insight into the most important terms used and the theoretical backdrop applied throughout the thesis.

Chapter 3. Research questions

This chapter provides an overview of the research questions.

Chapter 4. Experimental setup

Outlines the procedure for testing the research questions.

Chapter 5. New method
Outlines our contribution to this research field, providing pseudo code.

Chapter 6. Computational experiments

Presents the instances of data, the method of evaluation and the results of validation.

Chapter 7. Conclusion and future research

The concluding points of the research are presented. This chapter suggests some of the future work that can proceed from this thesis, which could be useful for further research in this field.

2. Background

This section outlines all theoretical knowledge that is necessary for understanding the research field. It consists of five subsections.

The first subsection covers the concept of classification in general terms, giving a small example of classification, and explaining basic terminology.

The second subsection presents a preprocessing technique for classification that transforms data from different types to binary. This technique will be applied to datasets intended for classification.

The third subsection presents which operations can be performed on Boolean transformed data. It outlines the basic theory of Boolean algebra and standard notation, which are needed in the next subsection.

The fourth subsection discusses the classification algorithm called Satisfiability Data Mining (SAT-DM, Glover 2008). This algorithm was the inspiration for this thesis and the new concepts developed here. This section also explains the general logic of the classifier that was presented in the original source (Glover 2008), as well as a detailed description of all necessary information for its implementation.

The fifth subsection presents a widely used and easy to build Naïve Bayes classifier, the principles of which will be used in this research.

2.1 Basic definition of a classification problem

Consider the problem of classification in a simple example. Suppose there is a database of clients of a travel agency with information on their ages and monthly incomes (Table 1). There are two types of travel tours: a more expensive, comfortable tour and a cheaper tour for students.
The travel agency wants to send advertising to clients based on which tour they can afford. Accordingly, two classes of customers are defined: Class 1 and Class 2.

Table 1. Travel agency customer data

Customer Code	Age	Income	Class
1	18	25	1
2	22	100	1
3	30	70	1
4	23	15	2
5	20	46	2
6	21	78	1
7	42	92	2
8	32	53	2

For clarity, the database is presented in a two-dimensional graph (axes: age and income), in the form of a set of objects belonging to Class 1 (orange) and Class 2 (blue). The classification task is to determine which class a new client belongs to, indicated in Figure 1 by a black point.

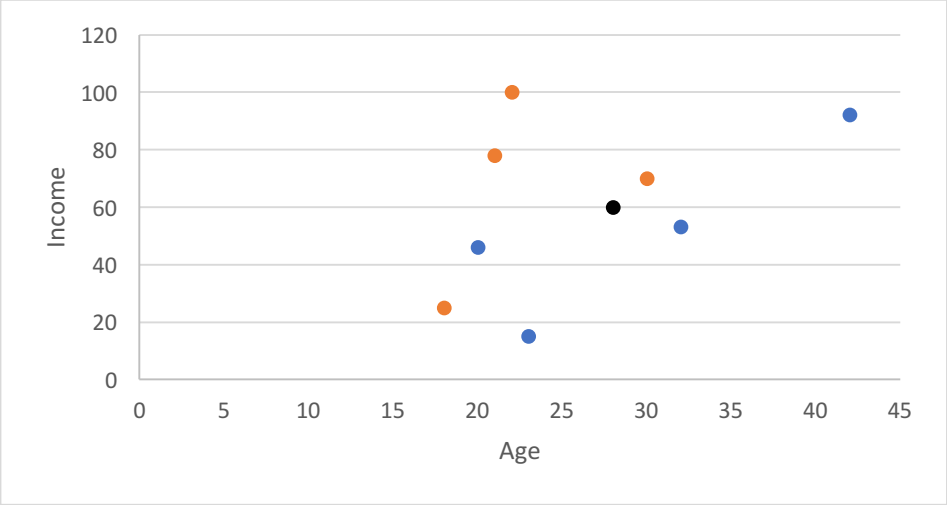


Figure 1. Objects from Table 1 in two dimensions

The collection of records/objects that are given in Table 1 are the training set. Each of these records—each separate client—is assigned the given class that characterize it.

A training set represents a finite set of precedents that are known: the observation and class. The training set in this case is shown in Table 1. A validation set is a set of records with new clients for which class is to be determined, which in this case is the black point shown in Figure 1. A

characteristic or property of a client (such as income or age) is called an attribute. Characteristics have types such as binary, categorical, or numerical.

Classification algorithms can be divided into two types: supervised and unsupervised.
Supervised classification (classification with a teacher) on the basis of a training set is required to predict the class of a new record based on the other attributes of this record. In unsupervised classification, there is no training data set and the classes of observations (outcomes) are unknown. A good example of solving an unsupervised classification problem is a clustering method, where classes have to be recognized/created in the process (Guido and Müller 2016).

The problem of this thesis is *supervised classification with binary attributes*.

2.2 Binarization procedure

The SAT-DM classification method examined by this thesis only allows for classifying objects with binary attributes, however there is a procedure for extending it with to different data types.

Binarization procedure transforms data from different types to binary. It is a preprocessing technique for classification that requires Boolean input data. The procedure is inspired by the iterative discriminant elimination (IDEAL) algorithm. It is a heuristic method that was presented at a conference by Moreira, Hertz and Mayoraz (1999). The method has significantly lower running times with the comparable performance overall. The IDEAL algorithm provides the notion discriminant—the divider that splits the whole attribute space into two subspaces, separating observations from different classes. The IDEAL algorithm can be divided into three steps.

In the first step, separate discriminants are generated for each dimension. Based to the dataset of clients of a travel agency (Table 1), Figure 2 contains all generated discriminants for both dimensions: age (x-axis) and income (y-axis).

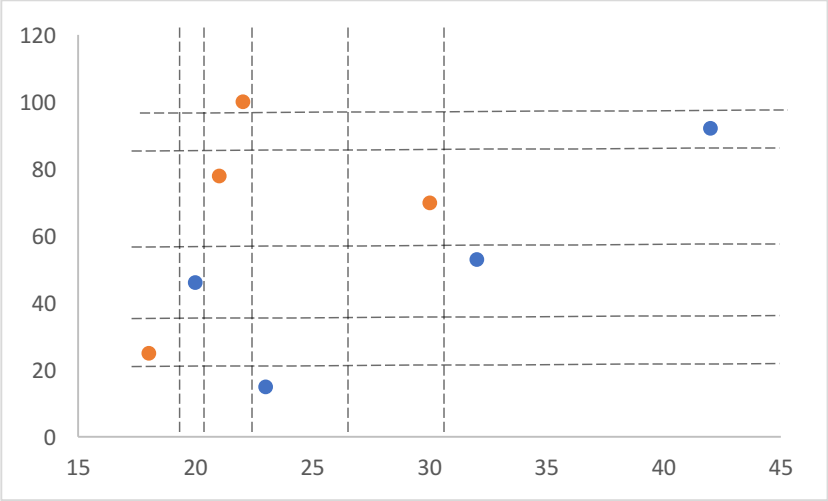


Figure 2. Binarization procedure, step 1

In general, the approach to creating discriminants differs by data types:

- ➔ Real data: generate a discriminant between each pair of values from different classes.
- ➔ Integer/binary data: generate a discriminant for each value, except all objects with the same or greater values of the same class.
- ➔ Categorical data: generate a discriminant for each category except one category (selected arbitrarily).
- ➔ Missing data: generally, as above, but one more discriminant added.

In the second step, the set is reduced using the discriminant elimination procedure. Discriminants are removed iteratively, as long as a consistency level (minimum number of discriminants, which separates any two objects from different classes) is maintained (Figure 3). The observations of each class must be in a separate section and not include any records from another section.

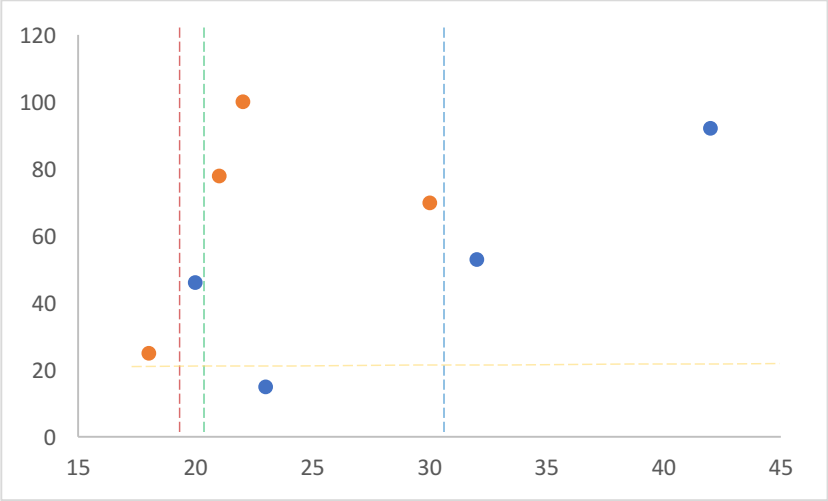


Figure 3. Binarization procedure: step 2

Finally, in the third step, the input data set is binarized using the obtained solution. At the end of the second step, four discriminants are left: red (R), green (G), blue (B) and yellow (Y) lines. For each discriminant on the *x* axis, if a record lies on the right side of the discriminant line, the value derived is 1, otherwise it is 0. For a discriminant on the *y* axis, records above the line are derived as 1, otherwise as 0.

The last column of Table 2 contains binarized values for the dataset in Table 1.

Table 2. Binarization procedure: step 3

Customer Code	Age	Income	RGBY
1	18	25	0001
2	22	100	1101
3	30	70	1101
4	23	15	1100
5	20	46	1001
6	21	78	1101
7	42	92	1111
8	32	53	1111

IDEAL is a destructive algorithm, which generates discriminants and then removes them in the course of execution. It is also possible to assign a weight to each discriminant, which allows one to manage which discriminant will be the next candidate (the one with the lowest weight) in the elimination process.

2.3 Basic theory of Boolean algebra

Using this binarization procedure, all attributes of a dataset can be transformed into binary values. This allows for building classification guidelines using laws of Boolean algebra, the branch of algebra in which the values of variables are binary.

This section presents further definitions from the basic theory of Boolean algebra that are directly used in the SAT-DM algorithm.

A *Boolean variable* is a variable that can accept a value from the set of Boolean values B = {0,1}. The Boolean value 0 can also be denoted by FALSE, and the Boolean value 1 as TRUE.

A *literal* is a Boolean variable or the negation of such a variable. Negation is a logical complement that is interpreted as being 1 when a variable is 0, and 0 when it is 1. The symbol ¬ represents the negation operator (Table 3).

Table 3. Truth table of the negation operator

x	$\neg x$
0	1
1	0

A *clause* is a disjunction of literals. It is a logical operator, which is interpreted as TRUE if at least one of the literals is TRUE, and FALSE otherwise. The symbol $\vee$ represents the disjunction operator (Table 4).

A *Boolean expression* is an expression that produces a Boolean value. Any Boolean expression can be presented in conjunctive normal form (CNF). A Boolean expression is in CNF if it is a conjunction of clauses. A *conjunction* is a logical operator that is interpreted as TRUE only if all of its clauses are TRUE, and FALSE otherwise. The symbol $\wedge$ represents the conjunction operator (Table 4).

Table 4. Truth table of disjunction and conjunction operators

x	y	$x \vee y$	$x \wedge y$
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

For example, the following formulas that use the Boolean variables p, q, r and t are in CNF:

- $p \vee \neg q$
- $(p \vee q) \wedge (r \vee \neg t)$
- $(p \vee \neg q) \wedge (r \vee t \vee \neg q) \wedge q$

Having presented the theoretical terms, it is helpful to look at the important problem of Boolean algebra, which is at the core of the SAT-DM classifier.

There are a lot of practical situations in which it is necessary to satisfy several potentially conflicting constraints. For example, in daily life when some human tries to make a decision consistent with different rules, or when a machine needs to confirm that a hardware/software system works correctly with its overall behavioral constraints. All such situations have variables whose values have to be determined, and constraints that these variables must satisfy (Malik and Zhang 2009). This leads to the definition of *Boolean Satisfiability Problem (SAT)*, one of the essential problems in computer science, artificial intelligence, bioinformatics, cryptography, etc. A wide range of practical applications are using SAT, from checking of pedigree consistency to software testing (Marques-Silva 2008). The success of existing SAT solvers motivates the development of more sufficient solvers and extend usage in applications.

In theoretical terms, a formula/Boolean expression $F(x_1, x_2, \dots x_n)$ that depends on Boolean variables is satisfiable if it is possible to find a solution x^*, such that $F(x^*) = true$. SAT checks whether the given formula is satisfiable.

Interestingly, all problems can be classified based on time or space complexity. The Boolean satisfiability problem is first proven to be a NP-complete problem (Cook 1971). NP-complete problems are those with a "Yes"/ "No" answer from the NP class, to which any other task from the class can be reduced in polynomial time. A NP class is the set of problems of related resource-based complexity.

2.4 Satisfiability data mining algorithm

Having reviewed all theoretical knowledge, it is time to introduce the classification algorithm that is the subject of this research.

Satisfiability Data Mining (SAT-DM) is a new method for binary data classification (Glover 2008). The method is based on generating a group of logical clauses in binary variables for each class. A new record with unknown membership is classified according to comparing the proportion of the clauses it satisfies for that class to the proportion it satisfies for other classes.

The problem addressed is as follows. Assume $G_k, k \in K$ is a set of classes, consists of records of class k in binary representation $x^i = (x_1^i, x_2^i \dots x_n^i), i \in G_k, n \in N$ (set of binary attributes). The mechanism is to build clauses for each class, $G_k, k \in K$, and, based on number of satisfied clauses, classify the vector.

Suppose that G_α is a set of objects of the same class for which representative inequalities are created, and G_β describes all objects of supplementary classes. The purpose of the SAT-DM method is to generate clauses that are satisfied by the elements of G_α and at the same time violated by elements of G_β.

2.4.1 Clauses represented as inequalities

The SAT-DM method (Glover 2008) begins from a compact representation of a clause (Section 2.4)—an inequality.

Let N_1 be a subset of N associated with unnegated variables ($x_j = 1$) and N_0 - subset of N associated with negated variables ($\overline{x}_j = 1$). A clause states:

$$\left(\vee_{j \in N_1} x_j\right) \vee \left(\vee_{j \in N_0} \overline{x}_j\right) \quad (2.1)$$

As mentioned above, the SAT-DM classification method determines the number of clauses it satisfies for a new record. However, it is inconvenient to work with clauses, so Glover suggests another representation of them.

Clause can be written as linear 0-1 inequality, where negated variables $\overline{x}_j$ are seen as $(1 - x_j)$. In addition, by means of analogy between the logical operator ∨ and the arithmetic operator +, a condition of satisfying the logical clause $x_1 \vee \neg x_2$ can be written as a linear inequality: $x_1 + (1 - x_2) \geq 1$ (Hooker 1992).

In terms of inequalities, the aim of the SAT-DM method is to identify instances (2.2) that are satisfied by all (or at least a valuable part) of the elements $x^i, i \in G_\alpha$.

$$\sum(x_j : j \in N_1) + \sum \left((1 - x_j) : j \in N_0\right) \geq 1 \quad (2.2)$$

To satisfy an inequality, a record needs to have at least one attribute either from N_1 with the value of 1 or from N_0 with the value of 0.

As noted earlier, the SAT-DM method attempts to find inequalities that are not only satisfied by records of G_α, but are also violated by records of G_β. Accordingly, records of G_β have to be satisfied by:

$$\sum(x_j : j \in N_1) + \sum \left((1 - x_j) : j \in N_0\right) \leq 0 \quad (2.3)$$

Summarized, the SAT-DM algorithm consists of the following steps:

1. Generate a set of inequalities for each class $G_k, k \in K$. The inequalities for G_k have the form of (2.2 - 2.3) with statement $G_\alpha = G_k$, G_β = complement of G_k.

2. The set of inequalities for G_k is composed by building solutions for an associated SAT problem. Each solution is an inequality that becomes as a separator for records from G_k class and other classes. Also, each inequality consists of combination of properties of the records in G_k.

3. A new record is classified according to the proportion it satisfies for each class $G_k, k \in K$.

2.4.2 Quasi Covering/Anti-Covering System

With the purpose of simultaneously satisfying a large number of inequalities by records from G_α (2.2) and violating by records from G_β (2.3) with using a small number of variables, Glover (2008) presents the Quasi Covering/Anti-Covering (QC/AC) system, which is shown below. In that system, inequalities are satisfied by elements of G_α (the covering problem) and are violated by the elements from G_β (the anti-covering problem). Variables $y_{1j}, y_{0j}, \ j \in N$ are binary variables, related to unnegated and negated attributes (respectively), and determined if an adjoining attribute is valuable in inequality.

$$Minimize \sum((y_{1j} + y_{0j}) : j \in N)$$

Subject to

$$\sum\left(x_j^i \, y_{1j} : j \in N_1\right) + \sum\left(\bar{x}_j^i \, y_{0j} : j \in N_0\right) \geq 1, \ i \in G_\alpha \tag{2.4}$$

$$\sum\left(x_j^i \, y_{1j} : j \in N_1\right) + \sum\left(\bar{x}_j^i \, y_{0j} : j \in N_0\right) \leq 0, i \in G_\beta \tag{2.5}$$

$$y_{1j} + y_{0j} \leq 1, \ j \in N \tag{2.6}$$

$$y_{1j}, y_{0j} \in \{0,1\}, \ j \in N \tag{2.7}$$

This model is supposed to find any solution that conforms to these constraints (2.4 - 2.7) as a strongly satisfying solution. Within the context of SAT-DM, this system demonstrates that this asymmetric form is beneficial in separating $G_k, k \in K$ from other classes.

However, the task of finding a group of inequalities that covers all of records of G_α and at the same time does not cover records of G_β (2.4-2.7) is most likely to be unfeasible for real data. This means it's necessary to specify a rule for creating inequalities—a method that allows the finding of inequalities in a simplified form that maximizes the coverage of G_α and minimizes the coverage of G_β.

2.4.3 Surrogate constraint

The Quasi Covering/Anti-Covering system raises the question: how to produce inequalities that include as many records as possible of G_α and exclude as many records as possible of G_β. Consequently, it can be interpreted as a covering problem.

To deal with covering problems, Glover (1968) proposes the use "surrogate constraint" as a heuristic method. In general, a surrogate constraint is an inequality that is designed for taking useful information that cannot be taken from parent constraint individually. It is generated by summarizing the right- and left-hand side inequalities it consists of. A surrogate constraint yields stronger relaxations for its components. This method is useful in a lot of optimization models.

This heuristic method seems very useful for creating inequalities, and is applied in the SAT-DM method. For implementation, two surrogate constraints have to be generated (for G_α and G_β) (2.4 - 2.5). For further explanation, assume that surrogate constraint coefficients are presented in the vectors s_a, s_b:

$$s_a = <a_1, a_2 \dots a_n>, n \in N \quad (2.8)$$

where $a_j = \sum(x_j^i : i \in G_\alpha), j = 1..N$

$$s_b = <b_1, b_2 \dots b_n>, n \in N \quad (2.9)$$

where $b_j = \sum(x_j^i : i \in G_\beta), j = 1..N$

2.4.4 Greedy construction

Having described these general concepts of the SAT-DM method and the surrogate constraint approach, it is time to present the process of generating an inequality.

An inequality is built with a “constructive heuristic”. In general terms, this is defined as:

- starting from scratch, values are iteratively assigned to the variables of a mathematical formulation of the problem
- the procedure is repeated until a complete solution is obtained

A “construction algorithm" is formed by “greedy” algorithms, a paradigm in which locally optimal choices are made at each stage. In the context of the SAT-DM method, this heuristic method looks like this: the specific parameter R stores the amount/proportion of records G_α sufficient to cover this class, and defines whether the process of building the inequality is complete. At each step, an attribute is added to the inequality, which has the best value for an inequality.

The simplest way of selecting the best candidate is to find the maximum difference between surrogate constraint coefficients (2.8) for unnegated literals (attributes which are equal to 1) :

$$best\ candidate = argmax(a_j - b_j), j = 1..N \quad (2.10)$$

The asymmetric way of selecting the best candidate for negated literals (attributes which are equal to 0), based on the number of records left uncovered, is:

$$best\ candidate = argmax((G'_\alpha - a_j) - (G'_\beta - b_j)), j = 1..N \quad (2.11),$$

where G'_α – number of uncovered records from G_α by generated inequality,

G'_β – number of uncovered records from G_β by generated inequality.

Referring to the example of a classification task in Section 2.1 and its binarization in Table 5, let us build a couple of inequalities to see the general idea.

Table 5. Example of training dataset for building inequalities

Customer Code	Binary data	Class
1	0001	1
2	1101	1
3	1101	1
4	1100	2
5	1001	2

6	1101	1
7	1111	2
8	1111	2

1. *Generate first inequality*

   s_a =<3, 3, 0, 4> according to (2.8)

   s_b =<4, 3, 2, 3> according to (2.9)

   Best candidate from unnegated literals = *argmax(-1,0,-2,1)* (from 2.10)

   Best candidate from negated literals = *argmax(1,0,2,-1)* (from 2.11)

It seems that the best option is to add a third negated literal to the inequality (2). It covers all records from G_α, so the process of generating inequalities can stop.

2. *Generate second inequality*

The next candidate is the fourth unnegated literal (1). As in the previous example, it covers all records from G_α, so the process of generating inequalities can stop.

Having presented these simple examples, a process of generating inequality in general terms can be addressed.

To know which attribute is added and which is the next candidate to add, the next sets are defined. N^0 stands for the value of the original N, G_α^0, G_β^0 denotes the original $|G_\alpha|, |G_\beta|$ and s_a^0, s_b^0 denotes the original form of s_a, s_b. The algorithm starts with $N' = N^0$, $G'_\alpha = G_\alpha^0$, $G'_\beta = G_\beta^0$, $s'_a = s_a^0, s'_b = s_b^0$. In each iteration, the best candidate is added to the inequality by removing it from the current N and updating G'_α, G'_β. Subsequently, the current N includes the indexes of the not-selected attributes, and G'_α, G'_β includes the numbers of records that are not covered by the current generated inequality. The surrogate sum coefficients s'_a and s'_b store (2.8-2.9) structures and can be determined by the number of uncovered vectors for the current iteration. The simplest rule for selecting the best candidate that was described in (2.10-2.11), but a more sophisticated guideline could be selected. Sets N_0 and N_1 are created to distinguish non-negated or negated attributes, which are added to the inequality and are empty in the beginning. The surrogate sum

rule uses the index $v^* \in \{0,1\}$ and the index $j^* \in N$. The construction proceeds until no more candidates are left or a sufficient number of records R is covered. $C_\alpha = <c_{\alpha 1}, c_{\alpha 2} \dots c_{\alpha n}>$, $n \in |G_\alpha|$ and $C_\beta = <c_{\beta 1}, c_{\beta 2} \dots c_{\beta n}>$, $n \in |G_\beta|$ store the number of records the current generated inequality covers.

Generate an inequality. Pseudo-code

0. Start with $G'_\alpha = G^0_\alpha$, $G'_\beta = G^0_\beta$, $N' = N^0$, $s'_a = s^0_a, s'_b = s^0_b$, $N_0 = \emptyset$, $N_1 = \emptyset$, $c_{\alpha j} = 0$, $c_{\beta j} = 0, j \in N$.
1. Identify v^* and j^* according to the rule:
   Set $v^* = 0$, $e_{0j} = max(a_j - b_j))$
   Set $v^* = 1$, $e_{1j} = max((G'_\alpha - a_j) - (G'_\beta - b_j))$
   $(v^*, j^*) = argmax(e_{0j}, e_{1j})$.
2. Set $N' := N' - \{j^*\}$, $N_{v^*} := N_{v^*} \cup \{j^*\}$, $G'_\alpha := G'_\alpha - a_{j^*}, G'_\beta := G'_\beta - b_{j^*}$ if $v^* = 1$
   $G'_\alpha := a_{j^*}, G'_\beta := b_{j^*}$ if $v^* = 0$
3. Update $a_j := \{ a_j - a_{j^*} | c_{\alpha j} = 0 \}$, $b_j := \{ b_j - b_{j^*} | c_{\beta j} = 0 \}$, $c_{\alpha j^*} := c_{\alpha j^*} + 1$; $c_{\beta j^*} := c_{\beta j^*} + 1$
4. If $N' = \emptyset$ or $G'_\alpha < (G^0_\alpha - R)$ terminate, else proceed to step 1.

Figure 4. Generating an inequality

2.4.5 Destruction

It is possible to use more intelligent guidance to build inequalities. The idea is to try to modify an existing solution by making small changes to the solution structure as more high quality inequalities could be found. In practice for the SAT-DM method, a destruction phase for building inequality processes could be added. It is applied by successively removing selected attributes from generated inequalities.

When in construction, the aim is to find the attribute that covers a maximum number of records from G_α and a minimum number of records from G_β; for the destructive phase, it is the opposite. The destructive process is performed until the defined threshold is achieved, i.e., the minimum amount/proportion of records G_α required by the inequality is covered.

Additionally, to reduce the number of inequalities or simplify the logic, the inequalities can be reduced to a prime implicant. This can be covered by a general inequality, in the sense that the generated subset of prime implicants is also satisfied by the same records.

Example: suppose A = (1100) and B = (0101) are the two records and two inequalities were generated, such as $c_1 = x_1 + x_2 \geq 1$ and $c_2 = x_1 + x_4 \geq 1$, where each of them satisfies both of the records. So, c_1 could be reduced to $c_1 = x_2$.

2.4.6 Creating multiple inequalities and summarizing the SAT-DM method

To avoid creating the same inequality several times and, thereby introducing a combination of attributes (literals) not previously checked, the SAT-DM method can be extended by introducing a simple memory structure to oversee the process. It general terms, the next inequality is started with a combination of two literals that have not been previously combined.

Summary of the steps of the SAT-DM method:

1. Find the attribute that maximizes the coverage of G_α and also minimizes the coverage of G_β using surrogate constraints rules.
2. Remove the records satisfied by the generated inequality and repeat the procedure, until no more records are left to be covered.
3. Generate the inequality corresponding to the selected attributes from step 1.
4. Cut/reduce the inequality to its prime implicant, if possible.
5. Add the current inequality into the set of inequalities.
5. Cut the inequality through a destruction phase, if possible, and add the inequality into the set of inequalities.
6. Repeat step 1 by excluding the first inequality (to avoid generating similar inequalities).
6. Finish the generation of inequalities when all records have been covered or when the number of inequalities reaches some reasonable threshold.
7. To classify, compare the proportion of satisfied or violated clauses and make a decision about the class for each record (Glover 2008).

The steps involving the destruction or reduction to prime implicants are optional.

2.5 Naïve Bayes classifier

The Naïve Bayes classifier an approach to classification. It is simple in implementation and fast in computation, which is useful for large data sets. Despite the simplicity, in many cases the classifier provides reasonable accuracy and can compete with more sophisticated algorithms. Therefore, it is attractive to investigate this technique and connect it with the SAT-DM method.

Referring back to the classification example dataset in Table 1, which consists of clients of a travel agency (Section 2.1), each record contains attributes of a client: income and age. The Naïve Bayes classifier can used to classify objects based on such attributes. The classifier is called "naïve" because it assumes that the existence of a particular attribute in a given class is unrelated to the existence of any other attribute. In addition, all attributes have the same influence on the outcome of the classification. For example, income and age values have equal importance and are considered as independent. A new client is placed to the class with Maximum Posterior Probability:

$$k_x = argmax_{k \in K}\, P(k|x) \quad (2.7),$$

where K is the set of classes and x is the object of classification.

To define the Maximum Posterior Probability, the classifier is based on Naïve Bayes theorem of probability. The theorem is stated mathematically in the formulas 2.8- 2.9 and allows one to find the probability of event *c* happening, given that event *d* has occurred. Event *d* is the hypothesis and *c* is the evidence.

$$P(c|d) = \frac{P(d|c) \times P(c)}{P(d)} \quad (2.8)$$

$$Posterior = \frac{likelihood * prior}{evidence} \quad (2.9)$$

- $P(c)$ is the *prior probability* of c. It can be directly estimated from the training set, where representativeness is the relative frequency of objects belonging to the class *c*.
- $P(c|d)$ is the conditional probability of *d* given *c*, the *posterior probability*.
- $P(d|c)$ is the conditional probability, the likelihood of c given d.
- $P(d)$ is the probability of *d*.

The theorem is used for classification in the following way: with the notation given in Section 2.3, where $k \in K, K$ is a set of classes, a classified record is in binary representation $x = (x_1, x_2 \dots x_n)$, $n \in N$ is a set of binary attributes (define as attributes in Naïve Bayes), the formula derived from 2.8 is as follows:

$$P(k|x_1x_2..x_n) = \frac{P(x_1x_2..x_n|k)\,P(k)}{P(x_1x_2..x_n)} \qquad (2.10)$$

Classifiers have a goal of defining Maximum Posterior Probability relating to (2.10), $k = argmax_{k\in K}P(k|x_1x_2..x_n)$. As all attributes are independent, it is possible to rewrite the numerator of 2.10 in the following way:

$$\begin{aligned} P(x_1x_2..x_n|k)\,P(k) &= P(x_1|k)P(x_2|kx_1)..(x_n|kx_1x_2..x_n)\,P(k) \\ &= P(x_1|k)P(x_2|k)..P(x_n|k)P(k) = \prod_{i\in 1..n} P(x_n|k)\;P(k) \end{aligned} \qquad (2.11)$$

The denominator does not change for all entries in the dataset. Therefore, the class of an object is determined (Cichosz 2015) by:

$$k = argmax_{k\in K}P(k|x_1x_2..x_n) = argmax_{k\in K} \prod_{i\in 1..k} P(x_i|k)\;P(k) \qquad (2.12)$$

Interestingly, if one of the probabilities is equal to 0 (2.12), the result will be distorted. The solution is to substitute probabilities $P(x_i|k)$, $i \in 1..k$ by log-probabilities using logarithmic properties:

$$\begin{aligned} k = argmax_{k\in K} \prod_{i\in 1..k} P(x_i|k)\,P(k) &= argmax_{k\in K} \ln\Big(\prod_{i\in 1..n} P(x_i|k) * P(k)\Big) \\ &= argmax_{k\in K} \ln\left(\prod_{i\in 1..n} P(x_i|k)\right) + \ln\big(P(k)\big) \\ &= argmax_{k\in K} \sum_{i\in 1..k} \ln P(x_i|k) + \ln\big(P(k)\big) \end{aligned} \qquad (2.13)$$

3. Research method and questions

The primary method for this research is empirical validation based on computational experiments. The purpose of this research is to find a means of improving the SAT-DM method.

The following research questions need to be answered:

1. Is it possible to build inequalities (clauses) in a more sophisticated way?
The original implementation of the method (Glover 2008) showed that the quality of generated inequalities, which are the basis of the classification method, need to be improved. This may be done by reducing the similarity of inequalities, increasing the number of generated inequalities, and applying some advanced techniques.

2. Is it effective to select inequalities for use in classification based on a Pareto principle?
Previously, classification was preformed using all generated inequalities. Classification might also be performed using a subset of inequalities, which have been selected based on an additional characteristic. The concept of a Pareto layer, which will be explained in a future section, is a good alternative for selecting a final set of inequalities.

3. Is it reasonable to combine the SAT-DM method with existing, broadly used classifiers?
A lot of different classifiers are available these days. After comparing the SAT-DM classification method with existing ones, there may be a case for combining approaches, classifying objects not just by comparing the proportion of clauses they satisfy/violate for each class, but by involving the guidance of a Naïve Bayes classifier.

4. Experimental setup

To validate the performance and reliability of results obtained from a classification model, special evaluation methods exist: holdout validation, k-fold cross-validation, leave-one-out cross validation, and repeated random sub-sampling validation. Any of these can be used to measure the accuracy of the results of classification, but for some data sets one of the evaluation methods will be preferable.

Typically, the k-fold validation technique is applied. It is a method of evaluating an analytical model and its behavior using independent data. In the model, the available data is divided into k parts. Then, the model is trained using $k - 1$ parts of the data, and the rest of the data is used for validation. This procedure is repeated k times, where each time processes a different part. As a result, each of the k parts of data is used for validation. Figure 5 presents an example of data splitting in 3-fold cross-validation.

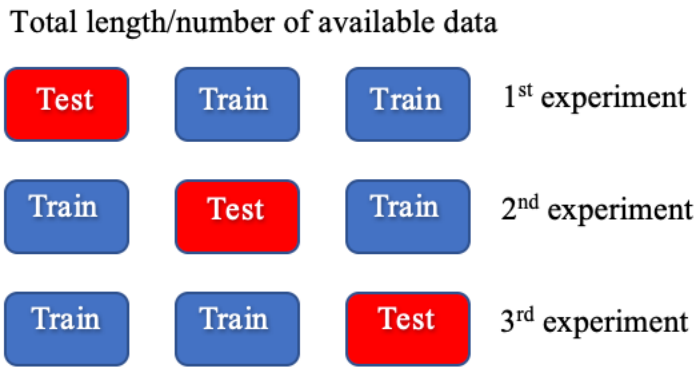


Figure 5. Data splitting in 3-fold cross validation

In the end, an assessment of the effectiveness of the selected model with the most uniform use of available data is generated. A problem with using k-fold validation is that the k repetitions are not independent of one another, and data that is used for training is also used for validation, making the estimator biased (López, Fernández and Herrera 2014). Performing multiple k-fold cross-validations may lower this bias and give a statistically better estimate. This could be done by mixing up observations in data sets and generating independent data sets. When repeating k-fold cross-validation, the average result of all of the k-fold cross-validations is taken.

For some datasets, a specific *k* has to be chosen, necessitates another evaluation method: leave-one-out cross-validation is a case of k-fold cross-validation in which the number of folds is equal to the number of instances in the dataset. If it is a dataset with *n* observations, then the training set contains *n-1* observations and the validation set contains just 1 observation. This process is repeated *n* times for each data point.

5. New method

This section provides techniques that answer the research questions. The first subsection describes the strategic oscillation approach and how it can be used in building inequalities. The second subsection presents a way of selecting high quality inequalities of a general set of inequalities. The third subsection outlines a method of classification involving the guidance of the Naïve Bayes classifier.

5.1 SAT-DM with strategic oscillation approach

Strategic oscillation (Glover and Kochenberger 1996) was introduced to extend the SAT-DM framework algorithm and is addressed here to answer to the first research question: *Is it possible to build clauses in a more sophisticated way?*

Originally, the strategic oscillation technique was created to solve optimization problems. This technique builds a solution using the oscillation between feasible and infeasible regions (Figure 6). As demonstrated by Glover and Kochenberger (1996), moving into the infeasible region is a good way of investigating the solution space and improving the quality of the solution. In addition, it forces the search for a solution into new areas.

This approach consists of two phases. In the first phase, it moves from the feasible to the infeasible space by incrementing the depth parameter to the threshold, improving the solution. In the second phase, it proceeds backward from the boundary into the feasible space. This could be described as the destructive phase.

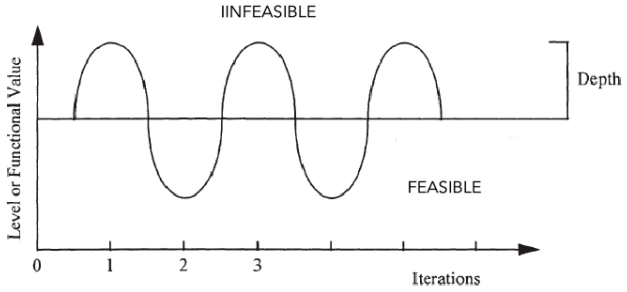


Figure 6. Strategic oscillation process (Source: Based on Glover and Kochenberger 1996)

Recall that in the original SAT-DM method greedy construction is used for building an inequality. After the inequality has covered sufficient number of records of G_α (define by R), it moves to the destruction process (Glover 2008). Afterwards, the generated inequality is saved and the process of building a new inequality starts from scratch. For the SAT-DM method, the strategic oscillation approach is to continue the search for new solutions (inequalities) after destruction by building/continuing construction for the previously destroyed inequality. It allows for the generation of more inequalities to explore the solution space. The process proceeds until the number of iterations of strategic oscillation reaches a defined threshold. This idea is presented in Figure 6, where the *c* and *d* parameters define thresholds (amount of covered records of G_α) for construction and destruction steps, respectively. All extremums (Figure 7) can be stored as a separate inequality in a set of inequalities.

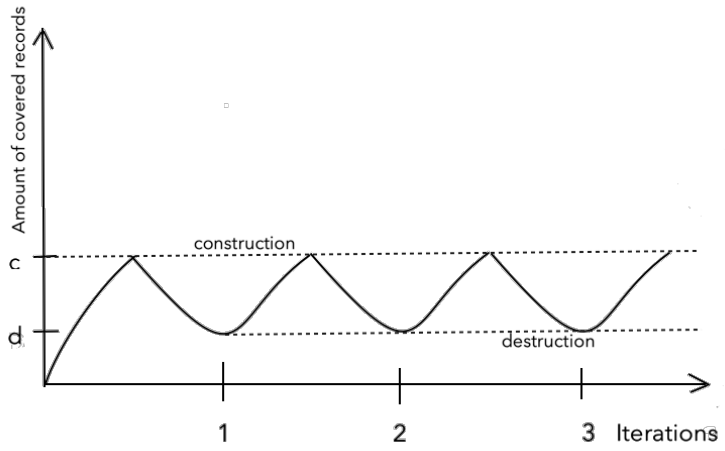


Figure 7. Strategic oscillation for the SAT-DM method

To avoid creating the same inequality several times and, therefore, introducing a combination of attributes (literals) not previously checked, the approach of Tabu Search is used (Glover and Laguna 1997). This technique will allow for the production of a different inequality. The word "tabu" comes from the Polynesian language Tongan, used by the aborigines of Tonga, to indicate things that cannot be touched because they are sacred.

Tabu Search is a memory-based method where one of the main components is to use adaptive memory. This strategy is allowed to perform an extensible search. It has a "tabu criterion" for

which the basic job is to not visit the same solution more than once. The memory structures can roughly be divided into three groups: short term (the list of solutions recently used), intermediate term, and long term (Glover and Løkketangen 2005).

For the SAT-DM algorithm, the approach of Tabu Search allows one to avoid the problem of generating the same inequalities several times. When an attribute is chosen to generate an inequality, it is given a "tabu tenure"—the number of iterations for which the value of that attribute is tabu.

Additionally, separate tabu tenures can be applied for the construction and destruction phases, to make oscillation more productive. This allows for the generation different inequalities in both phases for each iteration. Tabu tenures are applied to attributes and elements of inequality, and consists of a minimum number of strategic oscillations in which a literal is not allowed to be used.

In the implementation of the SAT-DM method in this thesis, tabu tenure for construction phase is linearly dependent on a strategic oscillation iteration and a number of attributes, what allows avoiding construction the same inequality several times, and therefore explicitly introduce a combination of attributes (literals) not previously tested.

For the destruction phase of SAT-DM in this paper, tabu tenure is defined to be quite short—one iteration of strategic oscillation—which creates a different destruction effect in each iteration.

The preprocessing step can be performed before building inequalities using a strategic oscillation process. This procedure identifies valuable attributes. In particular, if an attribute covered a sufficient number of records of G_α, it could be placed as a separate inequality with a size equal to 1. The corresponding attribute, which has been added to the preprocessed inequality, is ineligible for subsequent inequalities. However, a strategic oscillation procedure can also find inequalities with one attribute (which are received from the preprocessing step). Applying the preprocessing step helps to find inequalities with one attribute quickly, putting them into a separate set of inequalities and reducing the time of execution.

The following is pseudocode describing this strategic oscillation process for SAT-DM:

```
FUNCTION SAT-DM
PREPROCESSING
WHILE outer_iter < max_number_of_iterations
        INCREMENT outer_iter
        INITIALIZE inequality
        INITIALIZE tabu_status_c, tabu_status_d
                   sc_alpha, sc_beta
                   records_left_alpha, records_left_beta
                   alpha_covered, beta_covered
        WHILE iteration_so < max_number_of_so    //start "strategic oscillation"
                INCREMENT iteration_so
                        CONSTRUCTION STEP
                        DESTRUCTION STEP
        END WHILE            //end "strategic oscillation"
END WHILE                    //end the process of generating inequalities
END FUNCTION
```

Figure 8. Pseudocode of the SAT-DM algorithm with a strategic oscillation approach

This method aims to generate inequalities for G_α and supplementary group G_β. The process of generating inequalities proceeds until the threshold, *max_number_of_iterations*, is achieved in the loop (variable *outer_iter* is an iterator). *inequality* is a structure that stores selected literals for this inequality, as well as the number of records that are covered by the inequality for G_α and the supplementary group G_β, and a total number of records in the G_α and G_β groups. Tabu tenures are represented by the variables *tabu_status_c* and *tabu_status_d.* They store the number of the iteration during which it is already permitted to add or remove (respectively) an attribute after adding it to inequality. The variables *sc_alpha* and *sc_beta* store surrogate constraint coefficients for groups with the structure of the original SAT-DM method, updating with each iteration. *records_left_alpha, records_left_beta* control how many records are left uncovered, while *alpha_covered* and *beta_covered* store the number of inequalities each record is covered by.

Each inequality is generated using the strategic oscillation procedure. The stopping point for the procedure occurs when the variable *iteration_so* is greater than the number of iterations for the

strategic oscillation approach defined in *max_number_of_so*, or when construction/destruction is unsuccessful. Also, the process of generating inequalities will be finished when the construction step cannot add any new attributes for an inequality.

The construction step is described in this pseudocode:

```
CONSTRUCTION STEP
    WHILE records_left_alpha > max_records_left_construction
        INITIALIZE eval_best, attribute_best
            FOR ALL UNNEGATED ATTRIBUTES
                IF  outer_iter  == 1  AND attribute NOT IN tabu OR
                   outer_iter != 1  AND attribute NOT IN tabu_c
                      CALCULATE eval for attribute
                            IF eval > eval_best
                                  attribute_best := attribute
                            ELSE  BREAK
                            END IF
                ELSE BREAK
                END IF
             END FOR
            REPEAT FOR ALL NEGATED ATTRIBUTES
            ADD attribute_best to inequality
                      UPDATE tabu_status_s, tabu_status_d
                               records_left_alpha, records_left_beta
                               alpha_covered, beta_covered, sc_alpha, sc_beta
            ELSE BREAK
            END IF
        END WHILE
        ADD inequality to the set of generated inequalities
END CONSTRUCTION STEP
```

Figure 9. Pseudocode of the construction step

Construction of an inequality proceeds until a satisfying number of records for G_α is covered.
If this is the first iteration of the strategic oscillation procedure, a candidate for establishing an inequality must be checked for unavailability according tabu restriction. This implementation checks the value of *tabu*, which consists of the list of attributes which were previously used. The parameter *max_records_left_construction* contains the maximum number of records in G_α that can

be left uncovered by generated inequality. The variables *eval_best, attribute_best* help to determine the best candidate for addition to an inequality, and are empty in the beginning.
The evaluation process of a candidate to add to an inequality is based on 2.10-2.11 (Glover 2008).

After the construction step, the algorithm begins the destruction step (Figure 10). The main idea of this step is described in Section 2.5.5. The destruction step executes so long as a sufficient number of records for G_α are still covered.

This pseudocode describes the destruction step:

```
DESTRUCTION STEP
    INITIALIZE sc_alpha_destr, sc_beta_destr
        WHILE destruction possible
            INITIALIZE eval_best, attribute_best
            FOR ALL ATTRIBUTES IN INEQUALITY
                IF records_left_alpha + sc_alpha_destr <=
                    max_records_left_destruction
                        IF attribute NOT IN tabu_d
                            CALCULATE eval
                                IF eval > eval_best
                                    attribute_best := attribute
                                    eval_best := eval
                                ELSE  BREAK
                                END IF
                        ELSE  BREAK
                        END IF
                        UPDATE records_left_alpha, records_left_beta
                                sc_alpha_destr, sc_beta_destr
                                sc_alpha, sc_beta
                                alpha_covered, beta_covered,
                        ERASE attribute_best  from inequality
                ELSE BREAK //destruction is not possible
                END IF
        END WHILE
    ADD inequality to the set of generated inequalities
END DESTRUCTION STEP
```

Figure 10. Pseudocode of the destruction step

The variables *sc_alpha_destr, sc_beta_destr* contain surrogate constraint coefficients for records covered by one inequality. This allows for easy removal of a candidate and updating of the inequality. The implementation of all pseudocode provided in this section is presented in Appendix 1. The results of the implementation are shown in Section 6.2.

5.2 SAT-DM with Pareto principle

To generate a final set of inequalities, Glover (2008) suggests storing all resulting inequalities after construction and destruction phases and using all of them for classification. One of the research tasks (inspired by the second research question) in this master thesis is to select the final set of inequalities from all gradually generated inequalities by applying a sophisticated rule, which is explained below.

The SAT-DM method could be described as a bi-objective problem: maximizing the coverage of a particular class (G_α) and minimizing the violation of a supplementary group (G_β). In the process of building the solution (inequality), this approach is applied. However, some of generated inequalities are more optimal than others. In such cases, it is valuable to use the Pareto approach to find those optimal inequalities (Bandaru, Amos and Kalyanmoy 2014).

With this approach, the sequence of Pareto optimal solutions (inequalities) is generated. These are non-dominated solutions, from which it is possible to make a good decision in the process of choosing the end solutions.

Suppose $E = \{e_1, e_2, \dots, e_m\}$ is a set of generated inequalities, $f(e)/\ h(e)$ are functions, given the amount of G_α/G_β records respectively, covered by an inequality e.
The set of generated points is represented with the formula (with an example plot shown in Figure 11):

$$\{f(e_1), h(e_1);\ f(e_2), h(e_2); \dots\ ;\ f(e_n), h(e_m)\}$$

The Pareto optimal solutions are selected by applying the rule: $max\ f(E), min\ h(E)$.

A curve consisting of Pareto optimal solutions is called a "first Pareto layer". It is a set of inequalities are non-dominated solutions by values of $f(e)$ and $h(e)$ in appropriate max-min directions (the red points in Figure 11).

However, it could be useful to include several Pareto layers in situations where the first Pareto layer contains very few solutions (inequalities). After including the first one, the second Pareto layer of the solutions remaining after having excluded the first Pareto layer (the pink points in Figure 11) may be included, continuing to include more layers as needed. The number of Pareto layers used influences the number of inequalities in the final set: more layers leads to more inequalities.

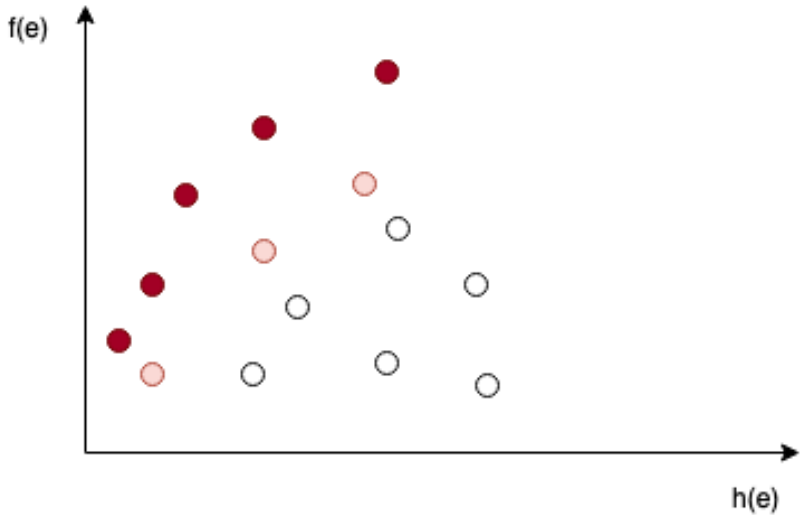


Figure 11. Pareto fronts in terms of inequalities

The implementation of this idea is presented in Figure 12.

Function Pareto Layer (PL) has a Boolean outcome and received inequalities *x* and *y* as parameters. The function specifies whether inequality *x* dominates inequality *y*, and returns an appropriate value in both cases.

FUNCTION PL(x,y)

IF *covered records of* G_α *by x == covered records of* G_α *by y* AND

covered records of G_β *by x == covered records of* G_β *by y*

RETURN false

END IF

IF *covered records of* G_α *by x > covered records of* G_α *by y* AND

covered records of G_β *by x < covered records of* G_β *by y*

OR

covered records of G_α *by x == covered records of* G_α *by y* AND

covered records of G_β *by x < covered records of* G_β *by y*

OR

covered records of G_α *by x > covered records of* G_α *by y* AND

covered records of G_β *by x == covered records of* G_β *by y*

RETURN true

ELSE

RETURN false

END IF

END FUNCTION

Figure 12. Function for determining the first Pareto layer points.

In the main method (function SAT, Section 5.1), function PL is applied for all generated inequalities to find dominant solutions (inequalities). Also, a set of preprocessed inequalities can be taken as a separate layer. The results of implementation are presented in the Section 6.3.

5.3 SAT-DM and Naïve Bayes classifier

One of the research tasks (inspired by the third research question) in this thesis is to combine the Naïve Bayes method with the SAT-DM classifier. Recall that in the origin paper (Glover 2008) a new record with unknown membership is classified by comparing the proportion of the inequalities it satisfies for that class to the proportion it satisfies for other classes. The task in this research is to use the inequalities in a better way when classifying (not comparing the proportion) using the Naïve Bayes classifier.

The approach is to extend attributes, specifically to add values of satisfying/non-satisfying (1/0 respectively), generated inequalities.

For attributes of a classified record $\mathbf{x}: = \{x_1, x_2 \dots x_n\} \cup \{s_{e_1}, s_{e_2}, \dots, s_{e_m}\}, n \in N$, N- a number of original attributes, $\{e_1, e_2, \dots, e_m\}$ is a set of generated inequalities.

s_{e_i} is a binary value indicating the satisfactory/unsatisfactory coverage status of an inequality e_i (Figure 13).

```
FOR inequalities of all classes
    IF inequality covers a record from a training set
        PUSH 1 into attributes of the record
                ELSE
        PUSH 0 into attributes of the record
    END IF
END FOR
```

Figure 13. Defining new attributes in the Naïve Bayes classifier

Thus, the classification is based on the Naïve Bayes classifier and the given attributes that are calculated based on the inequalities generated.

In addition, the original attributes could be removed and classification may be done just with $\{s_{e_1}, s_{e_2}, \dots, s_{e_m}\}$ attributes for the records. The results of classification with and without the original attributes are presented in the next section.

6. Computational experiments

6.1 Data sets

The data sets are taken from UC Irvine's open-source Machine Learning Repository, which was created to help the machine learning community (Machine learning community 2017). The validation process uses data sets of different nature, which allows for analyzing the results of classification more objectively.

1. Wisconsin breast cancer data set

This data set was produced by Dr. William H Wolberg of the University of Wisconsin–Madison Hospitals (Mangasarian and Wolberg 1990), and describes the diagnosis of breast tissues as either benign or malignant.

2. Chess (king-rook vs. king-pawn) data set

This data set was developed by Alen Shapiro and was supplied to Holte by Peter Clark of the Turing Institute in Glasgow (Shapiro 1983). It contains endgame results of chess games associated with a fixed set of starting positions of the game pieces.

3. Tic-tac-toe data set

This data set stores possible ending board arrangements for a set of tic-tac-toe games, with the player X always playing first, with to one of two possible outcomes: a win for X or a win for O (Aha 1991).

4. Molecular biology (splice-junction gene sequences) data set

This data set presents human splice-junction gene sequences (DNA) according to imperfect domain theory. Three possible outcomes exist: EI class (recognize exon/intron boundaries), IE class (recognize intron/exon boundaries), and neither (Noordewier, Towell and Shavlik 1991)

5. Lymphography data set

This data set was obtained from the University Medical Centre, Institute of Oncology, Ljubljana, Yugoslavia (Cestnik, Konenenko and Bratko 1987). Each record represents to one of four possible classes: normal result, metastases, malign lymph, or fibrosis.

6. Nursery data set

This dataset originally was produced to assist with nursery school enrollment. Five different outcomes were proposed: not recommended, recommended, very recommended, has priority, and has special priority (Olave, Rajkovic and Bohanec 1989).

7. LED display domain data set

Each record contains values of a light-emitting diode of a 7-segment display, and the classification task is to identify which digit is shown on the display. The dataset presents ten different outcomes, where each outcome has a 10% percent chance of occurring (Breiman et al. 1984).

Table 6. General description of the data sets

Nº	Name of Data Set	Number of instances	Number of classes	Number of attributes	Number of binary attributes
1	Wisconsin breast cancer	699	2	9	34
2	Chess	3196	2	36	36
3	Tic-tac-toe	958	2	9	18
4	Molecular biology (splicing)	3190	3	60	30
5	Lymphography	148	4	18	23
6	Nursery	12960	5	8	19
7	Led-display-domain	3200	10	7	24

6.2 Method of evaluation

For the lymphography and nursery data sets, the results are produced using leave-one-out cross-validation. For the other five data sets, the 3-fold cross-validation method (as described in Section 4) with 10 repetitions is applied.

6.3 Results

In the beginning of this subsection, the results for each dataset, shown using bar charts, will be provided and analyzed. Each bar represents the results of correct classifications, shown as a percentage, of applying each of the following techniques:

- **NB origin** bar: results of original Naïve Bayes Classifier
- **SAT** bar: the SAT-DM classifier with a strategic oscillation approach with using all generated inequalities for classification (Section 5.1)
- **SAT NB** bar: presents classification with guideline from the Naïve Bayes Classifier with additional attributes (satisfying inequalities, Section 5.2)
- **SAT NB W** bar: as previously described, but original attributes are not included
- **SAT PO, SAT PO NB** and **SAT PO NB W** bars: modifying the previous approaches by using points (inequalities) of the first Pareto layer for classification (Section 5.3)
- **SAT PO2, SAT PO2 NB** and **SAT PO2 NB W** bars: again modifying the previous approaches by using inequalities of two Pareto layers, the second Pareto layer of the solutions remaining after having excluded the first Pareto layer (Section 5.3)

Following these results, a general discussion of the research questions is provided.

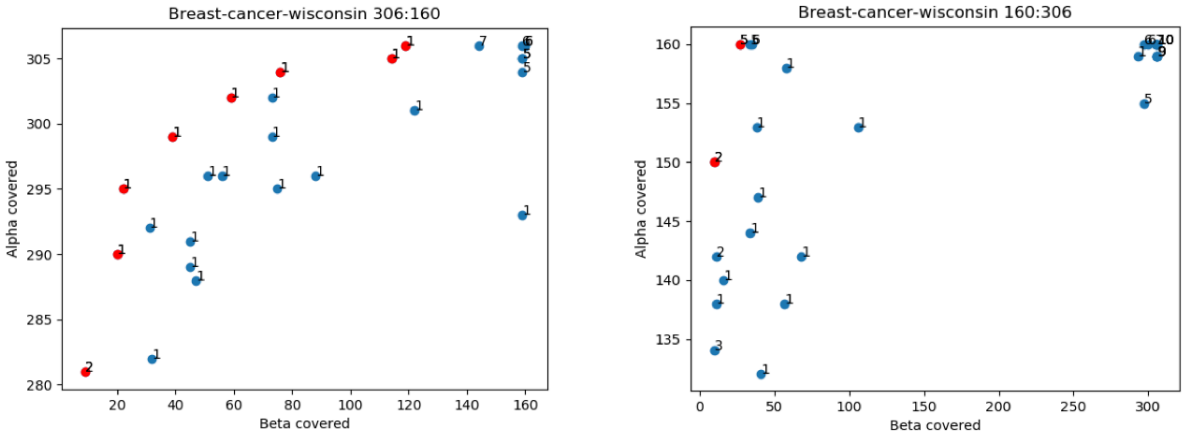
1. **WISCONSIN BREAST CANCER WISCONSIN DATA SET**

A three-fold cross-validation divides this data set into a training set with 466 records and a validation set with 233 records. As inequalities are built by training set records, it is valuable to know the number of records of each class. For example, for the first repetition (out of ten) the training set consists of:

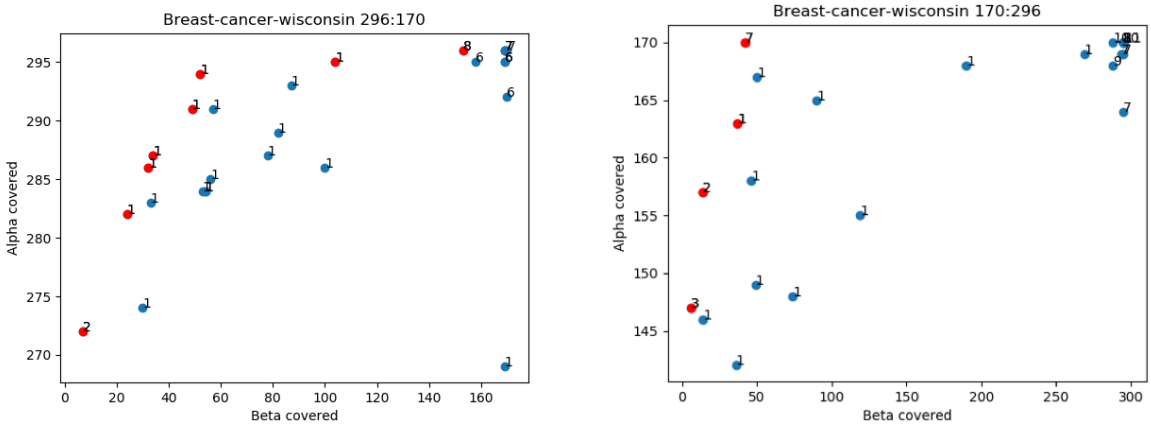
First experiment: the training set consists of 306 records of the benign class and 160 records of the malignant class.

Second experiment: the training set consists of 296 records of the benign class and 170 records of the malignant class.

Third experiment: the training set consists of 314 records of the benign class and 152 records of the malignant class.



Figures 14-15. The first experiment for the Wisconsin breast cancer data set



Figures 16-17. The second experiment for the Wisconsin breast cancer data set

The generated inequalities for two experiments by first repetition of a three-fold cross-validation are presented in Figures 14-17. Each inequality is a point, and its label is the number of attributes the inequality consists of. The y-axis shows the number of records from G_α covered by an inequality, and the x-axis shows the number of records from G_β it covers. The red points represent the first Pareto layer. In each experiment two figures are presented as it is two classes in this data set (see Appendix 2), and in Figures 14, 16 G_α is a set of records of the benign class where in Figures 15, 17 G_α is a set of records of the malignant class.

Figures 14-17 present inequalities just for one repetition of a three-fold cross-validation. However, it is possible to make a conclusion that the number of inequalities in general and in the first Pareto layer for this data set is too small. For example, in the first experiment of the first repetition 28

inequalities are generated and just 8 of them are selected (Figure 15), or 29 inequalities are generated and 2 of them are selected (Figure 16). Adding one more Pareto layer is not enough to receive good results. The problem of insufficiently selected inequalities based on a Pareto principle for use in classification was solved by treating preprocessed inequalities as a separate layer and including them in the final set of inequalities.

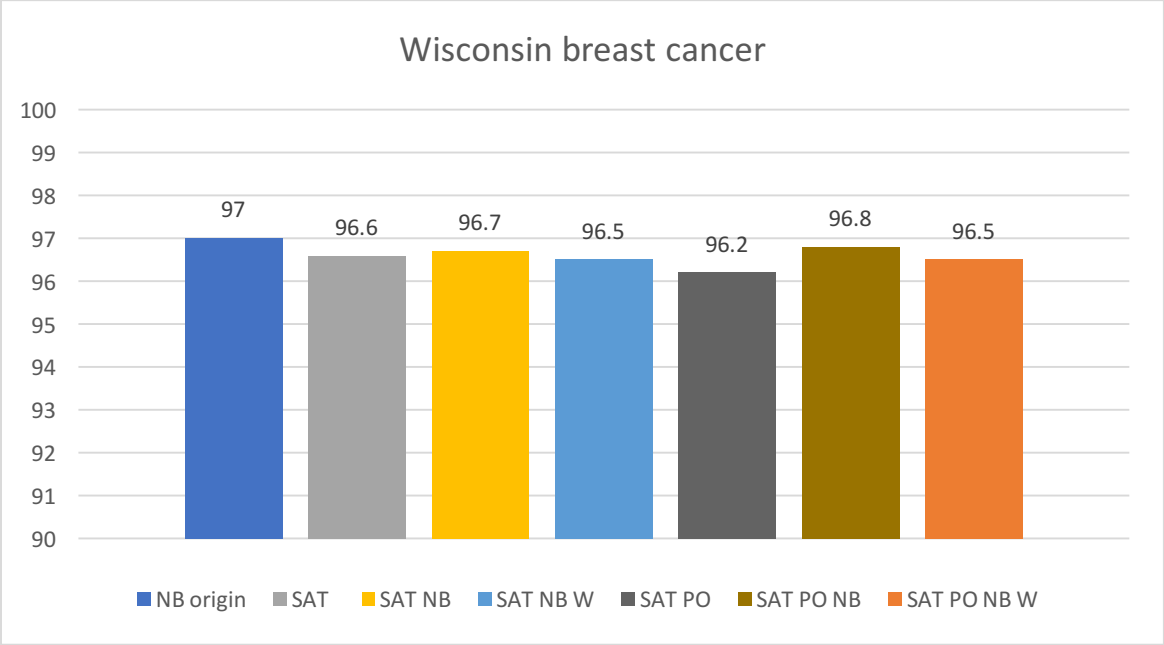


Figure 18. Results for the Wisconsin breast cancer data set

From Figure 18, the Naive Bayes classifier gives almost the same results as the new SAT-DM classifier in any implementation. However, the number of inequalities and the method of classification for the SAT-DM classifier are valuable. Selecting one layer of Pareto points (inequalities) for classification is able to produce almost the same result when the SAT-DM classifier takes all inequalities. Using inequalities in the original Naïve Bayes classifier improves the results of the SAT-DM classifier (the SAT NB and SAT PO NB bars) to nearly the same level as the original Naïve Bayes classifier.

2. CHESS DATA SET

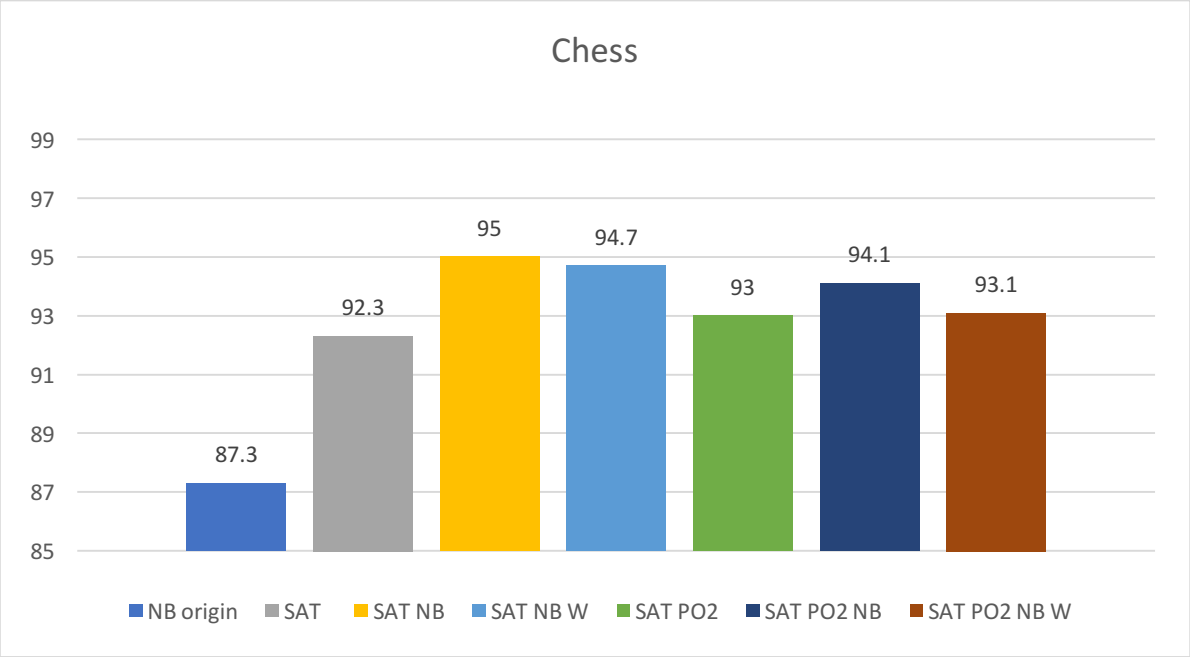


Figure 19. Results for the chess data set

The original Naïve Bayes classifier shows much worse results than the new SAT-DM classifier (Figure 19). However, the approach of making a classification based on the Naïve Bayes classifier using generated inequalities to build new attributes improves the results of the SAT-DM method (the SAT NB and SAT PO2 NB bars). This supports the theory that combining classifiers can produce a valuable outcome. Taking the first Pareto layer of points for classification was not enough, so the results for two Pareto layers is presented. The result shows that using a portion of the generated inequalities for classification can be enough to receive the same or even better results of classification (the SAT PO 2 bar).

3. TIC-TAC-TOE DATA SET

A three-fold cross-validation divides the data set into a training set with 639 records and a validation set with 319 records. For the first repetition (out of ten) the training set consists of:

- First experiment: 424 records of "win for X" and 215 records of "win for O"
- Second experiment: 410 records of "win for X" and 229 records of "win for O"
- Third experiment: 418 records of "win for X" and 210 records of "win for O"

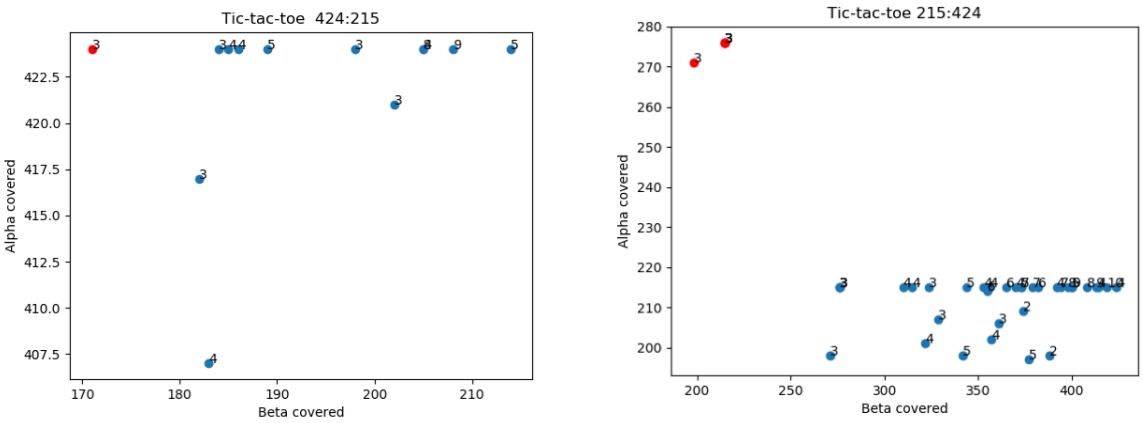


Figure 20-21. The first experiment for the tic-tac-toe data set

These figures show the generated inequalities for one experiment by first repetition of a three-fold cross-validation. Thus, it is possible to make a conclusion that the first Pareto layer consists of a small number of points (inequalities). Also, the number of attributes in the inequalities is large.

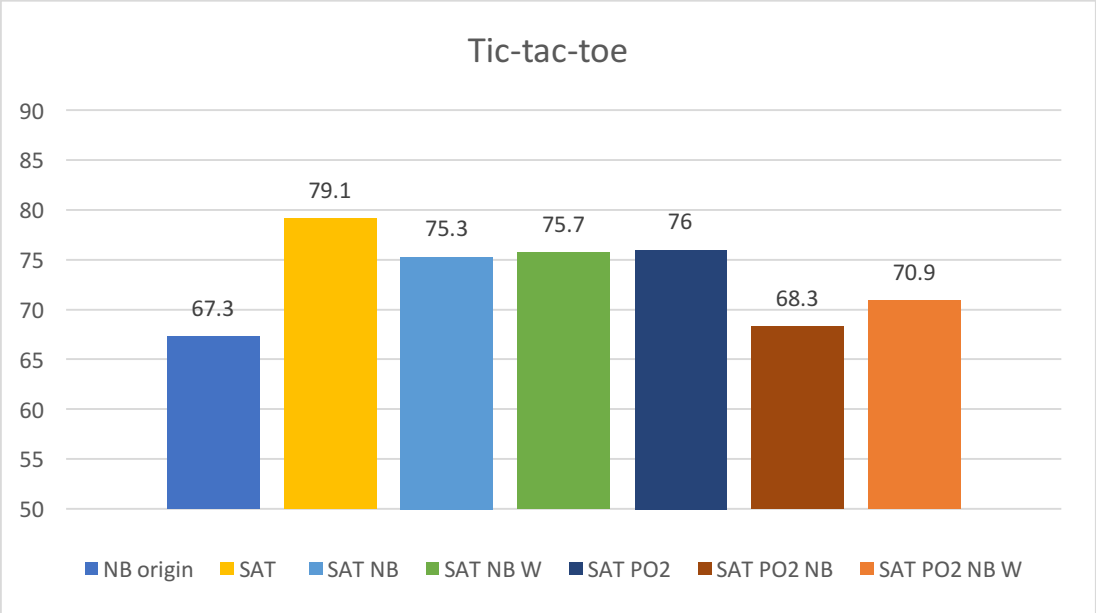


Figure 22. Results for the tic-tac-toe data set

As expected, taking two Pareto layers (Figure 22, last three columns) is insufficient for producing better results than classification with all inequalities. Also, the original Naive Bayes classifier gives poor results in classification. However, the approach of extending attributes for classification

(Section 5.3) improves the results of the Naive Bayes classifier at the cost of worsening the results of SAT-DM method.

4. SPLICE DATA SET

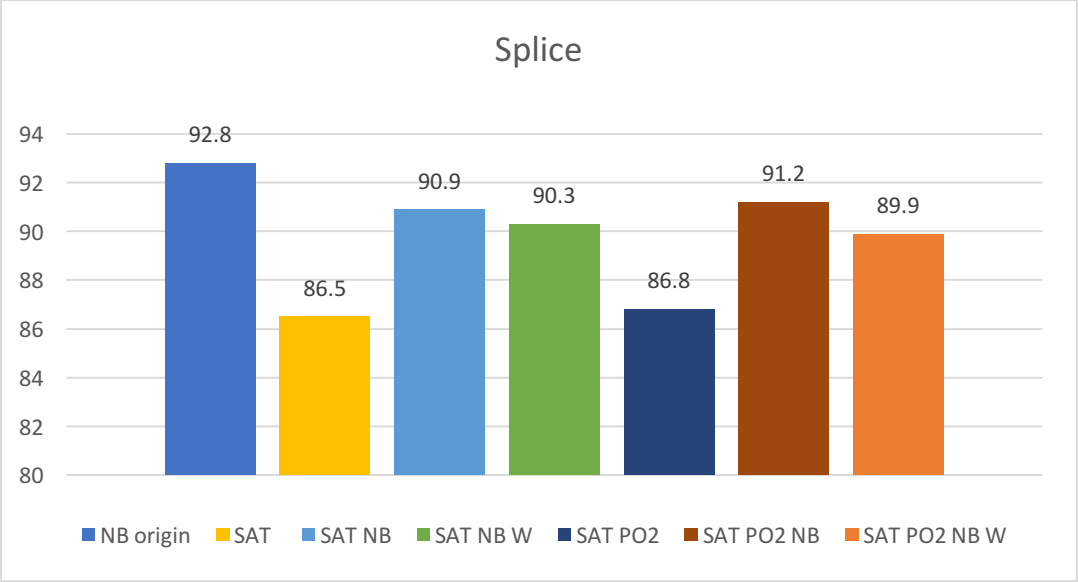


Figure 23. Results for the splice data set

The original Naive Bayes classifier results in a high quality classification compared to the SAT-DM classifier (Figure 23). This may be due to the increasing the number of classes (in this data set it is already three classes). However, involving the NB classifier in the SAT-DM classifier increases its performance nearly to the level of the original Naïve Bayes classifier.

The number of all generated inequalities is large as it depends on the number of binary attributes, which is valuable for this data set. Thus, using two Pareto layers of inequalities is better than taking all inequalities for classification and using a portion of the generated inequalities for classification can be enough to receive the same or even better results of classification (SAT PO2).

5. LYMPOGRAPHY DATA SET

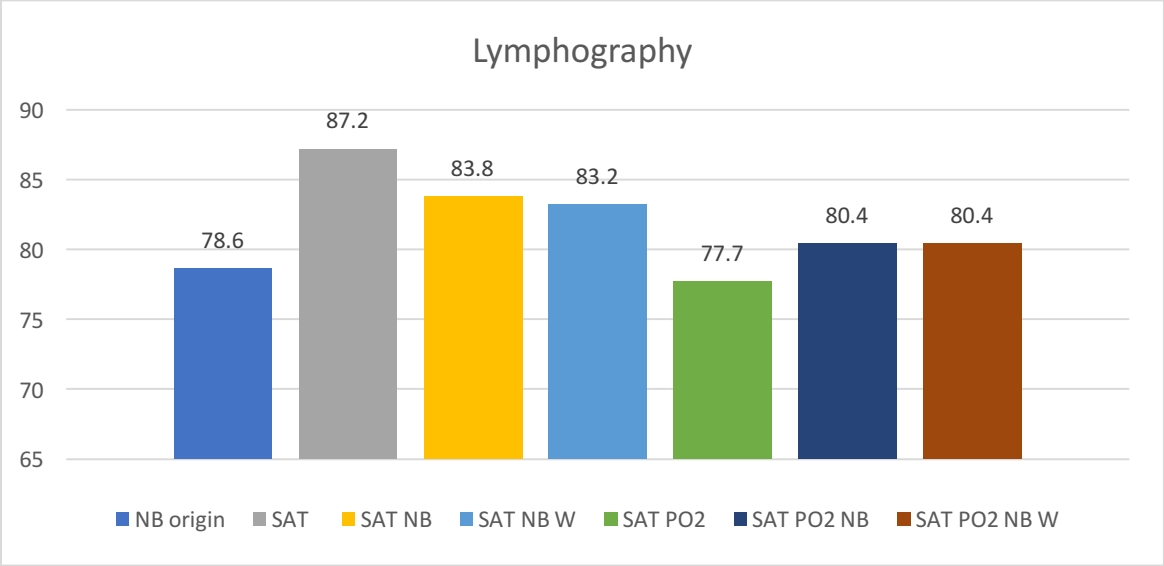


Figure 24. Results for the lymphography data set

The result of the original Naïve Bayes classifier is not satisfactory. This may be due to the weakness of the NB assumption that attributes are independent; the attributes (financial standing, social and health picture of the family) of this dataset are potentially related. Also, the original NB classifier needs more records to understand the probabilistic relationship of each attribute.

Despite the fact that this data set presents four different classes, two of them have a very small number of records. For more detailed information on each data set, please see Appendix 2. Thus, it is possible to assume that the classification for this dataset is binary (with two classes). This leads to the conclusion that the SAT-DM method performs very well for classification with two classes.

Compared to the origin NB classifier, the SAT-DM classifier shows better result in classification. However, the number of records (Table 6) influence on the number of generated inequalities, which is low. Therefore, the two Pareto layers of inequalities consist of few inequalities, which are not enough to produce good classification and requires taking all inequalities.

6. NURSERY DATA SET

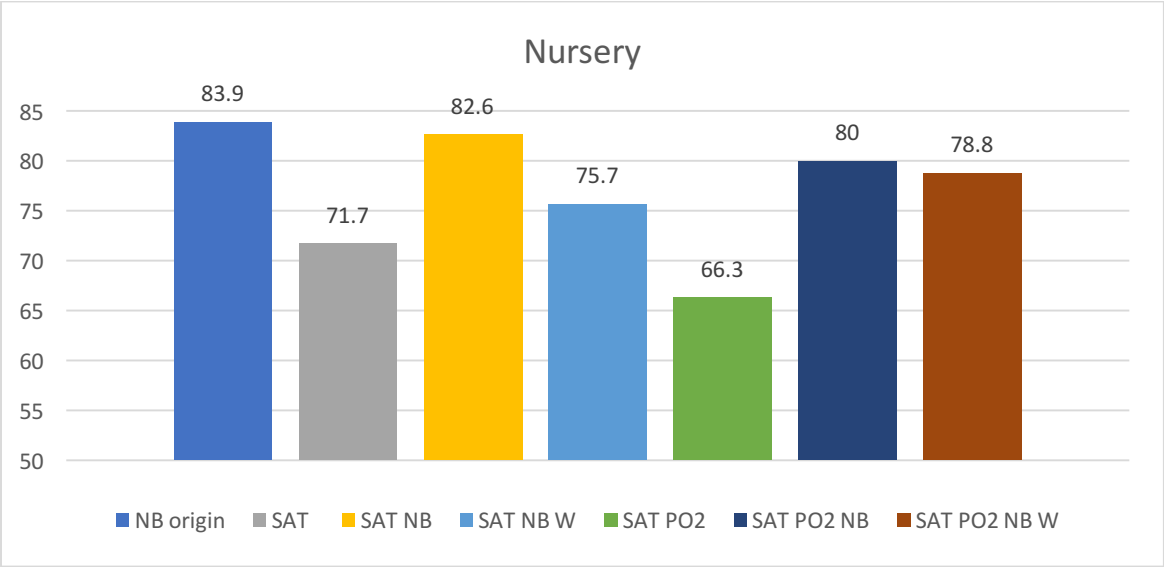


Figure 25. Results for the nursery data set

In this data set, the result of the original Naïve Bayes classifier is better than in the SAT-DM classifier. Referring to the results of previous data sets, this leads to the conclusion that the number of classes for the SAT-DM algorithm is important: better results are achieved with a smaller number of classes.

The result of the SAT-DM is not good as a small number of binary attributes for this data set is generated and therefore not enough inequalities exists to produce good classification. However, if the original Naïve Bayes classifier shows better results than the SAT-DM algorithm, involving NB techniques of classification will lead to improvement of the results.

As expected, two Pareto layers do not constitute enough inequalities, which requires taking all inequalities for classification.

7. LED DISPLAY DOMAIN DATA SET

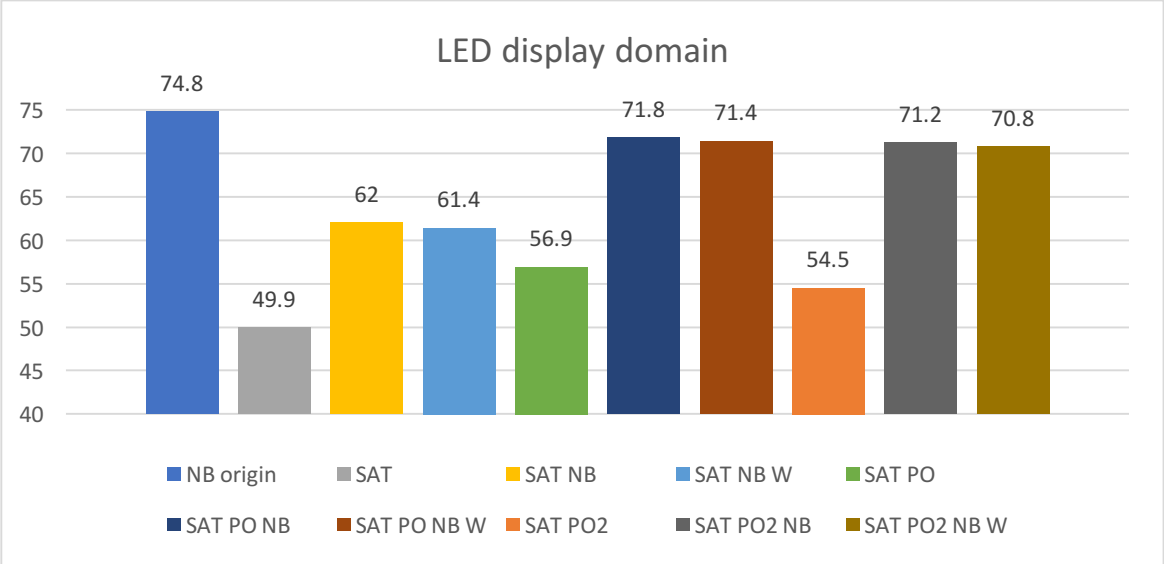


Figure 26. Results for the LED display domain data set

Classification of the LED display domain data set is an example of classification with a valuable number of classes (10 classes). As noted earlier, in such a case the SAT-DM classifier generates bad results (Figure 26). However, involving the NB classifier improves the performance of the method, as the original Naïve Bayes classifier shows better results (the same case with the nursery data set).

The first Pareto layer of inequalities is enough for classification if selecting more Pareto layers leads to worsening of the result (SAT PO2 column).

Involving the NB classifier technique to the SAT-DM classifier causes the result to improve nearly to the level of the original Naïve Bayes classifier, which can solve the problem of bad results for this data set.

A lot of inequalities are generated in this data set. This may due to the fact that the records are created artificially and each class consists of small number of records. However, tabu tenure, which is applied for the construction phase, also influence on this. For this data set, the value of this parameter, which is the same defined for all previous data sets, is too small and leads to the fact

that a lot of generated inequalities are similar. Increasing the tabu tenure for construction phase leads to the improving of the results and quality of generated solutions.
The results of this idea is presented in Figure 27.

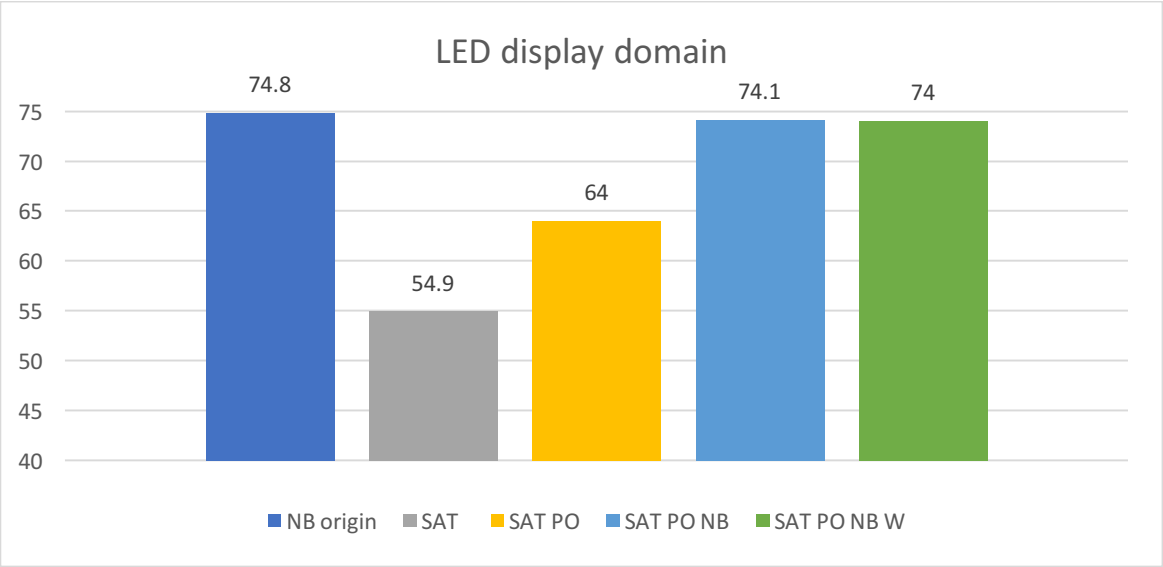


Figure 27. Results for the LED display domain data set with increased tabu tenure

As expected, classification with different inequalities is better. However, taking all generated inequalities for classification is not profitable compare to taking one Pareto layer, as enough inequalities are generated in this data set. Also, involving the NB classifier improves the performance of the method even with removing original attributes.

General conclusion

Having reviewed the results of validation for the data sets, the research questions may be answered:

1. Is it possible to build inequalities in a more sophisticated way?
The origin method of building inequalities in SAT-DM method may be extended by using a strategic oscillation approach. This makes it possible to improve the quality and number of

generated inequalities, as compared to the original method. However, results of new SAT-DM classifier are very sensitive to the parameters that define the strategic oscillation approach. It is vital to define the tabu tenures for the construction and destruction steps while taking into account the correlation between these parameters, the number of iterations of strategic oscillation, and the parameters that define the thresholds of the construction and destruction steps (Figure 7: *c* and *d* points).

Therefore, the total number of generated inequalities is dependent on the number of binary attributes a data set consists of. The result of the SAT-DM classifier also depends on the number of classes for a data set. It performs very well for classification with two classes, while increasing the number of classes worsens the results.

2. Is it effective to select inequalities for use in classification based on a Pareto principle?

From the whole collection of inequalities, it is possible to select the better ones (using separation on Pareto layers) and include them in the final set of inequalities for classification. However, selecting an insufficient number of inequalities (Pareto layers) may lead to poor results. The number of Pareto layers that need to be considered in classification is related to the total number of generated inequalities. With a large number of generated inequalities, it is enough to take a small number of Pareto layers.

For datasets with many classes, it is better to select fewer Pareto layers, as in general the SAT-DM algorithm is not suitable for these data sets, so just taking the best inequalities can provide some valuable result.

3. Is it reasonable to combine methods with existing, broadly used classifiers?

This thesis presents a way in which the classification for SAT-DM classifier can be modified. In the original method, a new record with unknown membership is classified by comparing the proportion of the inequalities it satisfies for that class to the proportion it satisfies for other classes. However, when the results of classification is poor, it is possible to involve classification using the Naïve Bayes classifier, as it described in this thesis.

In cases where the original Naïve Bayes classifier gives better results, it is very useful to apply the idea presented in Section 5.3 for improving the results of the SAT-DM classifier, which can solve the problem of bad results for data sets with many classes. Removing the original attributes from classification in SAT-DM with Naïve Bayes worsens its results, making them similar to the SAT-DM results.

When the original Naïve Bayes classifier gives worse results than the SAT-DM classifier, the approach of involving inequalities as new attributes is valuable in improving the original NB method. When that is done, the results become very close to those of the SAT-DM classifier. This leads to using the SAT-DM classifier as a preprocessor to generate more attributes to another classifier.

7. Concluding remarks and further research

7.1 Concluding remarks

This research aimed to analyze the Satisfiability data mining algorithm (Glover 2008) and identify effective strategies for improving it. This study shows a variety of techniques for improving the accuracy of classification: strategic oscillation approach, Pareto layers of solutions, and involving Naïve Bayes classifiers. The results of the proposed improvements heavily depend on a number of factors, including: the number of binary attributes, the nature of the data, and the result of the original Naïve Bayes classifier.

Using the strategic oscillation approach is profitable since all goals were achieved (reduced the similarity of clauses, increased the number of generated clauses). The technique of using Pareto layers (inequalities) is beneficial only if a sufficient total number of inequalities have been generated. The combination of classifiers leads to the improvement of the SAT-DM method, when the Naïve Bayes classifier generates better results. In conclusion, the techniques for modifying the existing SAT-DM algorithm in this paper are viable for improving classification results, as has been demonstrated through the application of the real-world data sets in Section 7.

7.2 Further research

This research shows the results of using more advanced implementations of the SAT-DM classifier, as compare to the original (Glover 2008). However, during the research it was found that the tabu statuses for the "strategic oscillation" procedure in implementation may be selected in a more sophisticated way.

Another suggestion for further research is to work with tree-based SAT-DM for generating decision trees. This framework can be used in a natural way within tree-based analysis approaches, giving another way of exploiting the inequalities. Refer to Glover (2006) for a more advanced treatment.

Reference list

Aha, David. 1991. "Incremental constructive induction: An instance-based approach." Edited by Morgan Kaufmann. *In Proceedings of the Eighth International Workshop on Machine Learning.* Evanston, ILL. 117-121.

Alexe, Gabriela, Sorin Alexe, Tiberius O. Bonates and Alexander Kogan. 2007. "Logical analysis of data – the vision of Peter L. Hammer." *Annals of Mathematics and Artificial Intelligence*, April: 265-312.

Bandaru, Sunith, Ng Amos and Deb Kalyanmoy. 2014. "On the Performance of Classification Algorithms for Learning Pareto-Dominance Relations." *IEEE Congress on Evolutionary Computation (CEC).* Beijing, China.

Breiman, Leo, Jerome Friedman, Charles J. Stone and R.A. Olshen. 1984. "Classification and Regression Trees." In *Wadsworth International Group*, 43-49. Belmont.

Cestnik, Bojan, Igor Konenenko and Ivan Bratko, I. 1987. "Assistant-86: A Knowledge-Elicitation Tool for Sophisticated Users." In *EWSL*, 31-45. Sigma Press.

Cichosz, Paweł. 2015. "Naïve Bayes classifier." In *Data Mining Algorithms, Part II: Classification*, 118-133.

Cook, Stephan. 1971. "The complexity of theorem-proving procedures." *STOC '71 Proceedings of the third annual ACM symposium on Theory of computing.* New York. 151-158.

Dogan, Neshihan and Zuhan Tanrikulu. 2013. "A comparative analysis of classification algorithms in data mining for accuracy, speed and robustness." *Information Technology and Management*, June: 105–124.

Dougherty, James, Ron Kohavi, and Mehran Sahami. 1995. "Supervised and Unsupervised Discretization of Continuous Features." *International Conference on Machine Learning.* Elsevier. 194–202.

Gardeux, Vincent, Lars Magnus Hvattum and Fred Glover. 2014. "SAT-DM: Satisfiability Data Mining for Classification Problems." *20th Conference of the International Federation of Operational Research Societies.* Barcelona: IFORS.

Glover, Fred. 2006. *Improved Classification and Discrimination by Successive Hyperplane and Multi-Hyperplane Separation.* University of Colorado, Boulder.

Glover, Fred. 2008. *Satisfiability Data Mining for Binary Data Classification Problems*. University of Colorado, Boulder.

Glover, Fred. 1968. "Surrogate Constraints." *Operations Research* 16 (4):741-749.

Glover, Fred and Arne Løkketangen. 2005. "Adaptive Memory Search Guidance for Satisfiability Problems." In *Metaheuristic Optimization via Memory and Evolution*, 213-227.

Glover, Fred and Manuel Laguna. 1997. "Tabu Search Principles." In *Tabu Search*, 125-151. Springer, Boston, MA.

Glover, Fred and Gary Kochenberger. 1996. "Critical Event Tabu Search for Multidimensional Knapsack Problems." In *Meta-Heuristics*, 407-427. Springer, Boston, MA.

Guido, Sarah and Anders C. Müller. 2016. *Introduction to Machine Learning with Python.* Edited by Dawn Schanafelt. Sebastopol: O'Reilly Media.

Hooker, John. 1992. "Generalized resolution for 0–1 linear inequalities." *Annals of Mathematics and Artificial Intelligence*, March: 271–286.

Machine learning community. 2017. *UC Irvine Machine Learning Repository.* University of California, School of Information and Computer Science. http://archive.ics.uci.edu/ml.

Malik, Sharad and Lintao Zhang. 2009. "Boolean Satisfiability: From Theoretical Hardness to Practical Success." *Communications of the ACM*, August: 76-82.

Mangasarian, Olwi and William Wolberg. 1990. "Cancer diagnosis via linear programming." *SIAM News*, September: 1-18.

Marques-Silva, Joao. 2008. "Practical applications of Boolean Satisfiability." *2008 9th International Workshop on Discrete Event Systems.* Goteborg, Sweden: IEEE. 74-80.

Moreira, Miguel, Alain Hertz and Eddy Mayoraz. 1999. "Data Binarization By Discriminant Elimination." *Proceedings of the ICML99 Workshop: From Machine Learning to Knowledge Discovery in Databases* 51-60.

Noordewier, Mochiel, Geoffrey Towell and Jude W. Shavlik. 1991. "Training Knowledge-Based Neural Networks to Recognize Genes in DNA Sequences." *Advances in Neural Information Processing Systems* (Morgan Kaufmann) 3.

Olave, Manuel, Vladislav Rajkovic and Marko Bohanec. 1989. "An application for admission in public school systems." In *Expert Systems in Public Administration*, 145-160.

Schlkopf, Bernhard and Alexander Smola. 2002. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond (Adaptive Computation and Machine Learning),* Cambridge, MA*:*MIT Press.

Shapiro, Alen. 1987. "Structured Induction in Expert Systems." *Addison Wesley Longman Publishing Co., Inc. Boston, MA, USA.*

López, Victoria, Alberto Fernández and Francisco Herrera. 2014. "On the importance of the validation technique for classification with imbalanced datasets: Addressing covariate shift when data is skewed." *Information Sciences* 257: 1-13.

Wu, Xindong, Vipin Kumar, Ross Quinlan and Joydeep Ghosh. 2008. "Top 10 algorithms in data mining." *Knowledge and Information Systems* (Springer-Verlag) 14 (1): 1-37.

Appendix 1

This part includes implementations of the main functions.

The following structures describe an inequality and a record respectively:

```
struct SATINEQ {
  double          eval;
  int             alpha_covered;
  int             alpha_total;
  int             beta_covered;
  int             beta_total;
  vector< int >   ineq;
};


struct DMBINARYRECORD {
  int             group;
  vector< bool >  record;
  string          name;
};
```

1. The first function (SAT_method_PO) presents the implementation of SAT-DM algorithm with a strategic oscillation approach and selection of two Pareto layers from a general set of inequalities.

```
bool SATDM::SAT_method_PO(vector< DMBINARYRECORD* > &G_alpha, vector< DMBINARYRECORD* >
&G_beta, vector< SATINEQ > &ineqs)
//------------------------------------------------------------------------------------
---------------------------
{
	cout << "CLASSIFICATION" << endl;
	if (G_alpha.size() == 0)
	{
		cout << " Generate inequalities for empty group." << endl;
		return true;
	}

	int i(0), j(0);
	ineqs.clear();

	vector< SATINEQ > preprocessed_ineqs;
	vector< SATINEQ > generated_ineqs;
	vector< SATINEQ > dominate_ineqs;
	vector< SATINEQ > t_ineqs;
	vector< SATINEQ > t2_ineqs;


	int n = (int)G_alpha[0]->record.size();
```

```
	int records_total_alpha = (int)G_alpha.size();
	int records_total_beta = (int)G_beta.size();
	double r = (double)records_total_beta / records_total_alpha;

	int max_records_left = (int)floor(records_total_alpha*(1 - min_cover_));

	int max_records_left_destruction = (int)floor(records_total_alpha*(1 -
min_cover_destruction_));



	vector< int > k_alpha, k_alpha_init;
	vector< int > k_beta, k_beta_init;

	//initialize the number of "true" value for surrogate constraint

	k_alpha_init.assign(n, 0);
	for (i = 0; i < records_total_alpha; i++)
	{
		for (j = 0; j < n; j++)
		{
			if (G_alpha[i]->record[j])
				k_alpha_init[j]++;
		}
	}

	k_beta_init.assign(n, 0);
	for (i = 0; i < records_total_beta; i++)
	{
		for (j = 0; j < n; j++)
		{
			if (G_beta[i]->record[j])
				k_beta_init[j]++;
		}
	}

	// 0,1,...,n-1 indices for unnegated, n,n+1,...,2n-1 indices for negated

	vector< int > tabu_status(2 * n, 0);

	vector< vector< int > > ts_general_attr(2 * n);


	// variables are available for consideration
	vector< bool > available(n, true);

	// the number of times attribute has been used in inequalities
	vector< int > times_used(2 * n, 0);

	double k_max = -DBL_MAX;
	int k_max_i(-1);
	int var(0), k_var(0);
	int neg;
	int max_appear = INT_MAX;


	// preprocess
```

```
	int preprocess_max = (max_records_left > max_records_left_destruction ?
max_records_left : max_records_left_destruction);

	for (i = 0; i < n; i++)
	{
		if (records_total_alpha - k_alpha_init[i] <= preprocess_max)
		{
			SATINEQ ineq;
			ineq.ineq.push_back(i + 1);
			available[i] = false;
			ineq.alpha_covered = k_alpha_init[i];
			ineq.alpha_total = records_total_alpha;
			ineq.beta_covered = k_beta_init[i];
			ineq.beta_total = records_total_beta;
			preprocessed_ineqs.push_back(ineq);
		}
		else if (k_alpha_init[i] <= preprocess_max)
		{
			SATINEQ ineq;
			ineq.ineq.push_back(-i - 1);
			available[i] = false;
			ineq.alpha_covered = records_total_alpha - k_alpha_init[i];
			ineq.alpha_total = records_total_alpha;
			ineq.beta_covered = records_total_beta - k_beta_init[i];
			ineq.beta_total = records_total_beta;
			preprocessed_ineqs.push_back(ineq);
		}
	}


	int outer_iter(0);
	bool done = false;

	while (!done)
	{

		SATINEQ ineq;
		vector< int > tabu_status_so(2 * n, 0);
		vector< int > tabu_status_destr(2 * n, 0);

		outer_iter++;
		max_appear = (int)ceil(const_max_appear);

		k_alpha = k_alpha_init;
		k_beta = k_beta_init;

		int records_left_alpha = records_total_alpha;
		int records_left_beta = records_total_beta;

		vector< int > alpha_covered(records_total_alpha, 0);
		vector< int > beta_covered(records_total_beta, 0);

		int amount_so = 0;
		bool so_done = false;

		//strategic oscillation

		while (amount_so < 100 && !so_done)
```

```
		{
			bool exist_construction = false;
			amount_so++;
			cout << amount_so << " strategic oscillation" << endl;
			//    CONSTRUCTION
			bool noimproving = false;
			bool ok = true;
			bool ok_destr = true;
			int so_iter = 0;
			double eval = -DBL_MAX;

			while (records_left_alpha > max_records_left && !noimproving &&
!so_done)
			{
				exist_construction = true;
				int in_iter = 0;
				k_max = -DBL_MAX;
				k_max_i = -1;


				// find attribute to add
				neg = n;
				for (i = 0; i < n; i++)
				{
					// non negated
					ok = true;
					if (available[i] && k_alpha[i] > 0 && times_used[i] <=
max_appear)
					{
						if ((tabu_status[i] > outer_iter && amount_so ==
1) || (tabu_status_so[i] > amount_so))
						{
							ok = false;
						}
						if (ok)
						{
							eval = k_alpha[i] - k_beta[i];

							if (eval > k_max)
							{
								k_max = eval;
								k_max_i = i;
							}
						}
					}
					// negated
					ok = true;
					if (available[i] && (records_left_alpha > k_alpha[i])
&& times_used[neg] <= max_appear)
					{
						if ((tabu_status[neg] > outer_iter  && amount_so
== 1) || (tabu_status_so[neg] > amount_so))
						{

							ok = false;
						}
						if (ok)
						{
```

```
                                                eval = (records_left_alpha - k_alpha[i])
- (records_left_beta - k_beta[i]);

                                                if (eval > k_max)
                                                {
                                                        k_max = eval;
                                                        k_max_i = neg;
                                                }
                                        }
                                }
                                neg++;
                        }

                        // add attribute, generate constraint
                        if (k_max_i >= 0)
                        {

                                tabu_status_so[k_max_i] = (int)(in_iter + amount_so +
0.5*n);

                                k_var = (k_max_i >= n ? k_max_i - n : k_max_i);

                                ineq.ineq.push_back((k_max_i == k_var ? k_var + 1 : -
k_var - 1));

                                available[k_var] = false;

                                tabu_status_destr[k_max_i] = (int)(amount_so + 1);


                                if (k_var == k_max_i) {
                                        records_left_alpha -= k_alpha[k_var];
                                        records_left_beta -= k_beta[k_var];
                                }
                                else {
                                        records_left_alpha = k_alpha[k_var];
                                        records_left_beta = k_beta[k_var];
                                }

                                if (in_iter == 0) {

                                        tabu_status[k_max_i] = (int)(t1_tenure_*n +
outer_iter);
                                }

                                for (i = 0; i < records_total_alpha; i++)
                                {
                                        if ((k_var == k_max_i && G_alpha[i]-
>record[k_var]) ||

                                                (k_var != k_max_i && !G_alpha[i]-
>record[k_var]))

                                        {
                                                if (alpha_covered[i] == 0)
                                                {
                                                        for (j = 0; j < n; j++)
                                                        {
                                                                if (G_alpha[i]->record[j])
```

```
                                                                        k_alpha[j]--;
                                                                }
                                                        }
                                                        alpha_covered[i]++;
                                                }
                                        }

                                        for (i = 0; i < records_total_beta; i++)
                                        {
                                                if ((k_var == k_max_i && G_beta[i]-
>record[k_var]) || (k_var != k_max_i && !G_beta[i]->record[k_var]))
                                                {

                                                        if (beta_covered[i] == 0)
                                                        {
                                                                for (j = 0; j < n; j++)
                                                                {
                                                                        if (G_beta[i]->record[j])
                                                                                k_beta[j]--;
                                                                }
                                                        }
                                                        beta_covered[i]++;
                                                }
                                        }
                                        //updated general tabu status
                                        int s = (int)ineq.ineq.size();
                                        int var, tvar = 0;
                                        for (i = 0; i < s; i++)
                                        {
                                                var = (ineq.ineq[i] < 0 ? n - ineq.ineq[i] - 1 :
ineq.ineq[i] - 1);

                                                times_used[var]++;
                                        }
                                }

                                else {
                                        noimproving = true;
                                }
                                if (noimproving && in_iter == 0)
                                {
                                        so_done = true;
                                }
                                if (noimproving && amount_so == 1) {
                                        done = true;
                                }
                                in_iter++;
                        }
                        if (!done && !so_done && exist_construction) {
                                ineq.alpha_covered = records_total_alpha - records_left_alpha;
                                ineq.alpha_total = records_total_alpha;
                                ineq.beta_covered = records_total_beta - records_left_beta;
                                ineq.beta_total = records_total_beta;
                                t_ineqs.push_back(ineq);
                        }

                        //destruction
```

```
                    //initialize sc prime implicant for records covered by 1,  to store
the number of uncovered records,
                    //if removing the attribute from the inequality
                    int iv = 0;
                    int s = (int)ineq.ineq.size();
                    vector< int > k_alpha_pi(s, 0);
                    vector< int > k_beta_pi(s, 0);

                    for (i = 0; i < records_total_alpha; i++)
                    {
                           if (alpha_covered[i] == 1)
                           {
                                  for (j = 0; j < s; j++)
                                  {
                                         iv = ineq.ineq[j];
                                         if (iv > 0 && G_alpha[i]->record[iv - 1])
                                                k_alpha_pi[j]++;
                                         if (iv < 0 && !G_alpha[i]->record[-iv - 1])
                                                k_alpha_pi[j]++;
                                  }
                           }
                    }

                    for (i = 0; i < records_total_beta; i++)
                    {
                           if (beta_covered[i] == 1)
                           {
                                  for (j = 0; j < s; j++)
                                  {
                                         iv = ineq.ineq[j];
                                         if (iv > 0 && G_beta[i]->record[iv - 1])
                                                k_beta_pi[j]++;
                                         if (iv < 0 && !G_beta[i]->record[-iv - 1])
                                                k_beta_pi[j]++;
                                  }
                           }
                    }


                    bool destructiondone = false;
                    bool changemade = false;
                    s = (int)ineq.ineq.size();

                    while (!destructiondone && !so_done)
                    {
                           k_max_i = -1;
                           k_max = -DBL_MAX;



                           for (j = 0; j < s; j++)
                           {
                                  if (records_left_alpha + k_alpha_pi[j] <=
max_records_left_destruction)
                                  {

                                         int attr = ineq.ineq[j];
```

```
                                        int dim_var = (attr < 0 ? n - attr - 1 : attr -
1);

                                        if (tabu_status_destr[dim_var] > amount_so  &&
amount_so != 1)
                                                a
                                        {
                                                ok_destr = false;
                                        }

                                        if (ok_destr) {

                                                eval = k_beta_pi[j] - k_alpha_pi[j];

                                                if (eval > k_max)
                                                {
                                                        k_max_i = j;
                                                        k_max = eval;
                                                }
                                        }
                                }
                        }
                        if (k_max_i < 0)
                        {
                                destructiondone = true;
                        }
                        else {
                                var = ineq.ineq[k_max_i];

                                if (var > 0)
                                        available[var - 1] = true;
                                else
                                        available[-1 - var] = true;

                                records_left_alpha += k_alpha_pi[k_max_i];
                                records_left_beta += k_beta_pi[k_max_i];
                                ineq.ineq.erase(ineq.ineq.begin() + k_max_i);
                                k_alpha_pi.erase(k_alpha_pi.begin() + k_max_i);
                                k_beta_pi.erase(k_beta_pi.begin() + k_max_i);

                                s--;
                                changemade = true;

                                //update coeffs (for each point, recalc number of times
it is covered, if reached 1 add to coefficient):


                                for (i = 0; i < records_total_alpha; i++)
                                {
                                        if ((var > 0 && G_alpha[i]->record[var - 1]) ||
                                                (var < 0 && !G_alpha[i]->record[-var -
1]))

                                        {
                                                alpha_covered[i]--;
                                                if (alpha_covered[i] == 1)
                                                {
                                                        for (j = 0; j < s; j++)
                                                        {
```

```
                                                int tvar = ineq.ineq[j];
                                                if ((tvar > 0 &&
G_alpha[i]->record[tvar - 1]) ||
                                                        (tvar < 0 &&
!G_alpha[i]->record[-tvar - 1]))
                                                {
                                                        k_alpha_pi[j]++;

                                                }
                                        }
                                }
                                else if (alpha_covered[i] == 0) {
                                        for (j = 0; j < s; j++)
                                        {
                                                int tvar = ineq.ineq[j];
                                                if ((tvar > 0 &&
G_alpha[i]->record[tvar - 1]) ||
                                                        (tvar < 0 &&
!G_alpha[i]->record[-tvar - 1]))
                                                {
                                                        k_alpha_pi[j]--;

                                                }
                                        }
                                        for (j = 0; j < n; j++)
                                        {
                                                if (G_alpha[i]->record[j])
                                                        k_alpha[j]++;
                                        }
                                }
                        }
                }
                for (i = 0; i < records_total_beta; i++)
                {
                        if ((var > 0 && G_beta[i]->record[var - 1]) ||
                                (var < 0 && !G_beta[i]->record[-var -
1]))
                        {
                                beta_covered[i]--;
                                if (beta_covered[i] == 1)
                                {
                                        for (j = 0; j < s; j++)
                                        {
                                                int tvar = ineq.ineq[j];


                                                if ((tvar > 0 && G_beta[i]-
>record[tvar - 1]) ||
                                                        (tvar < 0 &&
!G_beta[i]->record[-tvar - 1]))
                                                {
                                                        k_beta_pi[j]++;

                                                }
                                        }
                                }
                                else if (beta_covered[i] == 0)
                                {
```

```
                                                        for (j = 0; j < n; j++)
                                                        {
                                                               if (G_beta[i]->record[j])
                                                                      k_beta[j]++;
                                                        }

                                                        for (j = 0; j < s; j++)
                                                        {
                                                               int tvar = ineq.ineq[j];
                                                               if ((tvar > 0 && G_beta[i]-
>record[tvar - 1]) ||
                                                                      (tvar < 0 &&
!G_beta[i]->record[-tvar - 1]))
                                                               {
                                                                      k_beta_pi[j]--;

                                                               }
                                                        }
                                                 }
                                          }
                                   }

                                   }
                            }

                            if (changemade)
                            {
                                   s = (int)ineq.ineq.size();

                                   ineq.alpha_covered = records_total_alpha - records_left_alpha;
                                   ineq.alpha_total = records_total_alpha;
                                   ineq.beta_covered = records_total_beta - records_left_beta;
                                   ineq.beta_total = records_total_beta;
                                   t_ineqs.push_back(ineq);

                                   for (i = 0; i < s; i++)
                                   {
                                          int var = (ineq.ineq[i] < 0 ? n - ineq.ineq[i] - 1 :
ineq.ineq[i] - 1);
                                          times_used[var]++;

                                   }
                            }
                            else so_done = true;
                     }



                     for (i = 0; i < (int)ineq.ineq.size(); i++)
                     {
                            int var = ineq.ineq[i];
                            if (var > 0)
                                   available[var - 1] = true;
                            else
                                   available[-1 - var] = true;
                     }
```

```
		if (outer_iter >= max_iterations_)
		{
			done = true;
		}

		// termination: reached max_number inequalities, no attribute-combination
not used before, or no attribute improving coverage
		// generated a sufficient number of inequalities
	}


	//first Pareto layer

	int ts = t_ineqs.size();

	vector<bool> dominates(ts, true);

	for (i = 0; i < ts; i++) {
		for (j = 0; j < ts; j++) {
			if (dominate_SATINEQ(t_ineqs[i], t_ineqs[j]))
				dominates[j] = false;
		}
	}
	for (int i = 0; i < dominates.size(); i++) {

		if (dominates[i]) {
			dominate_ineqs.push_back(t_ineqs[i]);

		}
	}
	for (int i = 0; i < dominates.size(); i++) {

		if (!dominates[i]) {
			t2_ineqs.push_back(t_ineqs[i]);

		}
	}

	//second Pareto layer

	int ts2 = t2_ineqs.size();
	vector<bool> dominates2(ts2, true);

	for (i = 0; i < ts2; i++) {
		for (j = 0; j < ts2; j++) {
			if (dominate_SATINEQ(t2_ineqs[i], t2_ineqs[j]))
				dominates2[j] = false;
		}
	}

	for (int i = 0; i < dominates2.size(); i++) {


		if (dominates2[i]) {
			dominate_ineqs.push_back(t2_ineqs[i]);

		}
	}
```

```
	for (i = 0; i < (int)dominate_ineqs.size(); i++)
	{
		ineqs.push_back(dominate_ineqs[i]);
	}

	ineqs.insert(ineqs.end(), preprocessed_ineqs.begin(), preprocessed_ineqs.end());

	//evaluation
	for (int i = 0; i < ineqs.size(); i++) {
		ineqs[i].eval = (double)ineqs[i].alpha_covered / records_total_alpha
			- (double)ineqs[i].beta_covered / records_total_beta;
	}

	sort(ineqs.begin(), ineqs.end(), comp_SATINEQ_greater);

	if (ineqs.size() == 0)
		cout << "Cannot generate anything";
	return true;
}
```

2. The second function specifies whether inequality *a* dominates inequality *b*, and returns an appropriate value in both cases.

```
bool dominate_SATINEQ(SATINEQ &a, SATINEQ &b)
{
	if (a.alpha_covered == b.alpha_covered
		&& a.beta_covered == b.beta_covered) return false;
	if (a.alpha_covered > b.alpha_covered
		&& a.beta_covered < b.beta_covered) return true;
	if (a.alpha_covered == b.alpha_covered
		&& a.beta_covered < b.beta_covered) return true;
	if (a.alpha_covered > b.alpha_covered
		&& a.beta_covered == b.beta_covered) return true;
	return false;
}
```

3. The following function is the implementation of Naïve Bayes Classifier with adding new attributes, that are calculated based on the generated inequalities.

```
bool SATDM::naiveBayesEQ(DMBINARYRECORD * record, DMBinaryData &data, int
&classification) {
	int i, j;
	double best_cover_ratio = -DBL_MAX;
	classification = -1;
```

```
	bool q_satisf;
	double prob_small = 0.000000000001;

	int gen_inq_length = 0;
	for (int g = 0; g < data.getNoOfGroups(); g++) {
		gen_inq_length += inequalities_[g].size();
	}

	//prior probability contains amount of records for each group
	vector<int>prior_probability(data.getNoOfGroups(), 0);
	for (int i = 0; i < data.getRecordSet().size(); i++) {
		++prior_probability[data.getRecordSet()[i]->group];
	}

	//for each group if record (un)satisfy inequality-> push 1(0)
	vector<bool>record_sat_inq;
	for (int g = 0; g < data.getNoOfGroups(); g++) {

		for (i = 0; i < inequalities_[g].size(); i++) {

			q_satisf = false;
			for (j = 0; j < (int)inequalities_[g][i].ineq.size(); j++)
			{
				int var = abs(inequalities_[g][i].ineq[j]) - 1;

				if ((inequalities_[g][i].ineq[j] > 0 && record->record[var])
|| (inequalities_[g][i].ineq[j] < 0 && !record->record[var]))
				{
					q_satisf = true;
				}
			}
			if (q_satisf)
				record_sat_inq.push_back(1);
			else { record_sat_inq.push_back(0); }
		}
	}

	//for each training record, (un)satisfy inequality for group it classified -> push
1(0)
	vector<vector<bool>>data_sat_inq(data.getRecordSet().size(), vector<bool>(0));

	for (int r = 0; r < data.getRecordSet().size(); r++) {
		//int g = data.getRecordSet()[r]->group;
		vector<bool> dr(data.getRecordSet()[r]->record);

		for (int g = 0; g < data.getNoOfGroups(); g++) {
			for (i = 0; i < inequalities_[g].size(); i++)
			{
				q_satisf = false;
				for (j = 0; j < (int)inequalities_[g][i].ineq.size(); j++)
				{
					int var = abs(inequalities_[g][i].ineq[j]) - 1;

					if ((inequalities_[g][i].ineq[j] > 0 && dr[var]) ||
(inequalities_[g][i].ineq[j] < 0 && !dr[var]))
					{
						q_satisf = true;
						break;
```

```
                                }
                        }
                        if (q_satisf)
                                data_sat_inq[r].push_back(1);
                        else { data_sat_inq[r].push_back(0); }
                }
        }
    }

    //for each group the amount of records of training data satisfied by record value
    vector <vector<int>>likelihood(data.getNoOfGroups(), vector<int>(record-
>record.size(), 0));
    for (int k = 0; k < data.getRecordSet().size(); k++) {
        for (int j = 0; j < (int)record->record.size(); j++) {
            if (record->record[j] == data.getRecordSet()[k]->record[j])
                ++likelihood[data.getRecordSet()[k]->group][j];
        }
    }

    vector <vector<int>>likelihood_sat_inq(data.getNoOfGroups(),
vector<int>(gen_inq_length, 0));

    //the same as likelihood, just for clauses
    for (int k = 0; k < data.getRecordSet().size(); k++) {
        int g = data.getRecordSet()[k]->group;
        for (int j = 0; j < (int)record_sat_inq.size(); j++) {
            if (record_sat_inq[j] == data_sat_inq[k][j])
                ++likelihood_sat_inq[g][j];
        }
    }

    for (i = 0; i < data.getNoOfGroups(); i++)
    {
        double ratio;
        double likelh_pr = 1.0;
        double likelh_pr_sat_inq = 1.0;



        for (int j = 0; j < record->record.size(); j++) {
            if (likelihood[i][j] != 0 && prior_probability[i] != 0) {
                likelh_pr *= double((double)likelihood[i][j] /
(double)prior_probability[i]);
            }
            else likelh_pr *= prob_small;
        }

        for (int j = 0; j < record_sat_inq.size(); j++) {
            if (likelihood_sat_inq[i][j] != 0 && prior_probability[i] != 0) {

                likelh_pr_sat_inq *= double((double)likelihood_sat_inq[i][j] /
(double)prior_probability[i]);
            }
            else likelh_pr_sat_inq *= prob_small;
        }

        ratio = ((double)prior_probability[i] / (double)data.getRecordSet().size())
*(double)likelh_pr * likelh_pr_sat_inq;
```

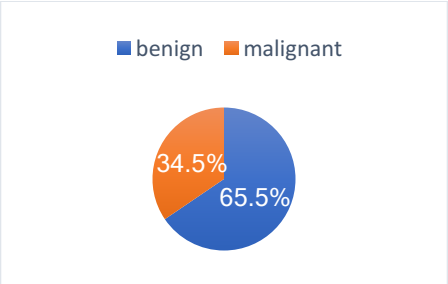
```
                if (ratio > best_cover_ratio)
                {
                        best_cover_ratio = ratio;
                        classification = i;
                }
        }

        return (classification >= 0);
}
```

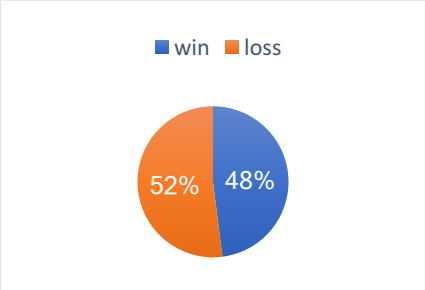
Appendix 2

This part includes distributions of records for data sets.

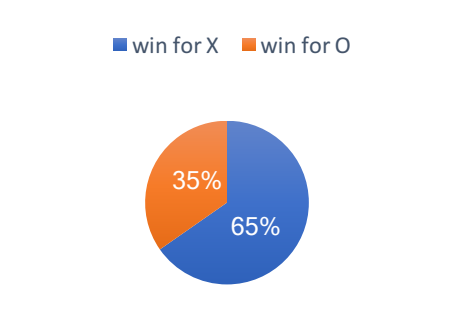
1. **Wisconsin breast cancer data set**



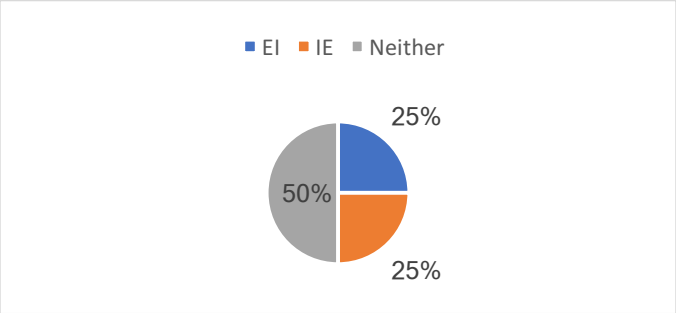
2. **Chess (king-rook vs. king-pawn) data set**



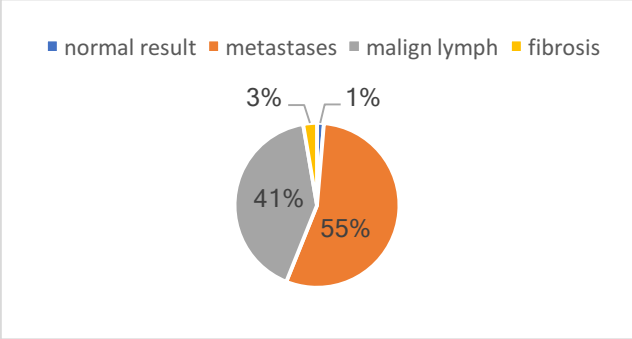
3. **Tic-tac-toe data set**



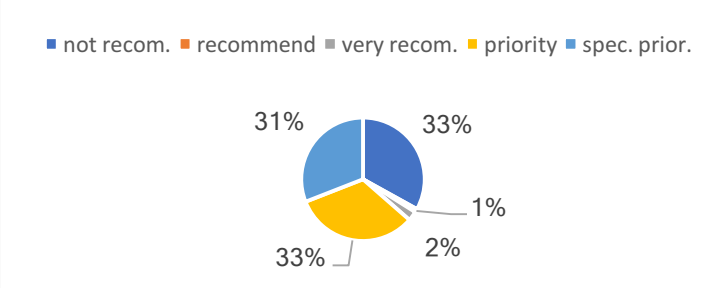
4. **Molecular biology (splice-junction gene sequences) data set**



5. Lymphography data set



6. Nursery data set



7. LED display domain data set