

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO CEARÁ  
PRÓ-REITORIA DE PESQUISA, PÓS-GRADUAÇÃO E INOVAÇÃO  
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Antônio Carlos Souza Ferreira Júnior

**An Incremental Model and an Iterative Model Based on MaxSAT for  
Learning Interpretable and Balanced Classification Rules**

Fortaleza – CE

July 3, 2024

Antônio Carlos Souza Ferreira Júnior

# **An Incremental Model and an Iterative Model Based on MaxSAT for Learning Interpretable and Balanced Classification Rules**

Dissertação submetida ao Programa de Pós-Graduação em Ciência da Computação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará, como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Linha de Pesquisa: Inteligência Artificial

Supervisor: Prof. Dr. Thiago Alves Rocha

Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE)

Pró-Reitoria de Pesquisa, Pós-Graduação e Inovação (PRPI)

Programa de Pós-Graduação em Ciência da Computação (PPGCC)

Fortaleza – CE

July 3, 2024



Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE)  
Pró-Reitoria de Pesquisa, Pós-Graduação e Inovação (PRPI)  
Programa de Pós-Graduação em Ciência da Computação (PPGCC)

Antônio Carlos Souza Ferreira Júnior

Esta dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação, sendo aprovada pela Coordenação do Programa de Pós-Graduação em Ciência da Computação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará e pela banca examinadora:

---

**Supervisor: Prof. Dr. Thiago Alves Rocha**  
Instituto Federal de Educação, Ciência e Tecnologia do  
Ceará (IFCE)

Fortaleza – CE  
July 3, 2024

Dados Internacionais de Catalogação na Publicação  
Instituto Federal do Ceará - IFCE  
Sistema de Bibliotecas - SIBI  
Ficha catalográfica elaborada pelo SIBI/IFCE, com os dados fornecidos pelo(a) autor(a)

---

Júnior, Antônio Carlos Souza Ferreira.

An Incremental Model and an Iterative Model Based on MaxSAT for Learning Interpretable and  
Balanced Classification Rules / Antônio Carlos Souza Ferreira Júnior. - 2024.  
38 f. : il.

Dissertação (Mestrado) - Instituto Federal do Ceará, Mestrado em Ciência da Computação, Campus  
Fortaleza, 2024.

Orientação: Prof. Dr. Thiago Alves Rocha.

1. Interpretable Artificial Intelligence. 2. Explainable Artificial Intelligence. 3. Rule Learning. 4.  
Maximum Satisfiability. I. Título.

CDD 004

---

# Abstract

The increasing advancements in the field of machine learning have resulted in the development of numerous applications that effectively address a wide range of problems with accurate predictions. However, in some cases, accuracy alone may not be sufficient. Many real-world problems also demand explanations and interpretability behind the predictions. Therefore, one of the most popular interpretable models for solving this type of problem is classification rules.

This work aims to propose an incremental model and an iterative model for learning interpretable and balanced rules based on MaxSAT, called IMLIB and I-IMLIB, respectively. This new model was based on two other approaches, one based on SAT and the other on MaxSAT. The former limits the size of each generated rule, making it possible to balance them. We suggest that such a set of rules seems more natural to be understood compared to a mixture of large and small rules. The second approach, called IMLI, presents two techniques to increase performance. One technique involves learning a set of rules by incrementally applying the model to partitions of the dataset. The other consists of learning a new rule at each iteration in the incremental process of the first technique.

The conducted experiments showed that IMLIB and its iterative version can generate smaller and more balanced sets of rules than IMLI and its iterative version. Moreover, the new approaches manage to stay close to or even surpass IMLI and its iterative version in terms of accuracy. Regarding training time, the iterative version of IMLIB achieved improvements compared to IMLIB, but the iterative version of IMLI still stood out in all tested datasets.

**Keywords:** Interpretable Artificial Intelligence. Explainable Artificial Intelligence. Rule Learning. Maximum Satisfiability.

# List of Tables

|                                                                                                                                                                                               |    |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| Table 1 – Datasets information. . . . .                                                                                                                                                       | 38 |
| Table 2 – Comparison between IMLI, I-IMLI, IMLIB and I-IMLIB in different datasets. The column Training time represents the training time in seconds. . . . .                                 | 39 |
| Table 3 – Comparison between IMLI and IMLIB in terms of balancing of the set of rules on the mushroom dataset. The configuration used was $k = 3$ , $\lambda = 10$ , and $lp = 16$ . . . . .  | 40 |
| Table 4 – Comparison between I-IMLI and I-IMLIB in terms of balancing the set of rules on the mushroom dataset. The configuration used was $k = 3$ , $\lambda = 10$ , and $lp = 16$ . . . . . | 42 |

# List of abbreviations and acronyms

|        |                                                                          |
|--------|--------------------------------------------------------------------------|
| ML     | Machine Learning                                                         |
| SAT    | Boolean Satisfiability Problem                                           |
| MaxSAT | Maximum Boolean Satisfiability Problem                                   |
| SQFSAT | Synthesis of quantifier-free first-order sentences over strings with SAT |
| DNF    | Disjunctive Normal Form                                                  |
| CNF    | Conjunctive Normal Form                                                  |

# List of symbols

|                    |                                                                                   |
|--------------------|-----------------------------------------------------------------------------------|
| $n$                | Number of samples in a dataset                                                    |
| $i$                | Integer variable ranging from 1 to $n$                                            |
| $m$                | Number of features in a dataset                                                   |
| $j$                | Integer variable ranging from 1 to $m$                                            |
| $\mathbf{X}$       | Binary matrix of size $n \times m$ representing a dataset                         |
| $\neg\mathbf{X}$   | Binary matrix representing $\mathbf{X}$ with opposite values                      |
| $\mathbf{X}_i$     | Binary vector of size $m$ representing a sample from $\mathbf{X}$                 |
| $\neg\mathbf{X}_i$ | Binary vector representing $\mathbf{X}_i$ with opposite values                    |
| $x_{i,j}$          | Binary value representing an element of $\mathbf{X}_i$                            |
| $\neg x_{i,j}$     | Binary value representing $x_{i,j}$ with the opposite value                       |
| $\mathbf{y}$       | Binary vector of size $n$ representing the classifications of $\mathbf{X}$        |
| $\neg\mathbf{y}$   | Binary vector representing $\mathbf{y}$ with opposite values                      |
| $y_i$              | Binary value representing an element of $\mathbf{y}$                              |
| $\neg y_i$         | Binary value representing $y_i$ with the opposite value                           |
| $x^j$              | Label representing a feature of $\mathbf{X}$                                      |
| $\neg x^j$         | Label representing the opposite of $x^j$                                          |
| $p$                | Number of partitions of $\mathbf{X}$                                              |
| $t$                | Integer variable ranging from 1 to $p$                                            |
| $\mathbf{X}^t$     | Binary matrix representing a partition of $\mathbf{X}$                            |
| $\neg\mathbf{X}^t$ | Binary matrix representing $\mathbf{X}^t$ with opposite values                    |
| $\mathbf{y}^t$     | Binary vector representing the classifications of the samples from $\mathbf{X}^t$ |
| $\neg\mathbf{y}^t$ | Binary vector representing $\mathbf{y}^t$ with opposite values                    |
| $\mathcal{E}^-$    | All $\mathbf{X}_i$ that have $y_i = 0$                                            |

|                            |                                                                                                                                               |
|----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| $\mathcal{E}^+$            | All $\mathbf{X}_i$ that have $y_i = 1$                                                                                                        |
| $ \mathcal{E}^- $          | Number of $\mathbf{X}_i$ that have $y_i = 0$                                                                                                  |
| $ \mathcal{E}^+ $          | Number of $\mathbf{X}_i$ that have $y_i = 1$                                                                                                  |
| $k$                        | Number of rules                                                                                                                               |
| $o$                        | Integer variable ranging from 1 to $k$                                                                                                        |
| $\mathbf{R}$               | Set of rules in DNF                                                                                                                           |
| $ \mathbf{R} $             | Number of labels in $\mathbf{R}$                                                                                                              |
| $R_o$                      | One rule from $\mathbf{R}$                                                                                                                    |
| $ R_o $                    | Number of labels in $R_o$                                                                                                                     |
| $\mathbf{R}(\mathbf{X}_i)$ | Application of $\mathbf{R}$ to $\mathbf{X}_i$                                                                                                 |
| $R_o(\mathbf{X}_i)$        | Application of $R_o$ to $\mathbf{X}_i$                                                                                                        |
| $\mathbf{M}$               | Set of rules in CNF                                                                                                                           |
| $ \mathbf{M} $             | Number of labels in $\mathbf{M}$                                                                                                              |
| $M_o$                      | One rule from $\mathbf{M}$                                                                                                                    |
| $\mathbf{M}(\mathbf{X}_i)$ | Application of $\mathbf{M}$ to $\mathbf{X}_i$                                                                                                 |
| $Cl$                       | Set of clauses in CNF                                                                                                                         |
| $w$                        | Number representing a weight                                                                                                                  |
| $\infty$                   | Infinite weight                                                                                                                               |
| $W(Cl) = w$                | Application of weight $w$ to all clauses in $Cl$                                                                                              |
| $l$                        | Maximum number of labels per rule                                                                                                             |
| $d$                        | Integer variable ranging from 1 to $l$                                                                                                        |
| $u_{o,d}^j$                | Boolean variable of the SQFSAT and IMLIB approaches representing that a certain label will be in a certain rule and at a certain position     |
| $p_{o,d}$                  | Boolean variable of the SQFSAT and IMLIB approaches representing whether a certain label will appear in the rule in an opposite manner or not |
| $u_{o,d}^*$                | Boolean variable of the SQFSAT and IMLIB approaches representing that a certain label of a certain rule will be skipped                       |

|                |                                                                                                                                                                                                            |
|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| $e_{o,d,i}$    | Boolean variable of the SQFSAT and IMLIB approaches representing that a certain label of a certain rule contributes to the correct classification of a certain sample                                      |
| $z_{o,i}$      | Boolean variable of the SQFSAT and IMLIB approaches representing that a certain rule contributes to the correct classification of a certain sample                                                         |
| $\lambda$      | Number representing a weight of the IMLI approach used to penalize classification errors                                                                                                                   |
| $v$            | Integer variable ranging from 1 to $2m$                                                                                                                                                                    |
| $b_o^v$        | Boolean variable of the IMLI approach representing the presence or absence of a label in a certain rule                                                                                                    |
| $\eta_i$       | Boolean variable of the IMLI approach representing whether a certain sample was classified correctly or not                                                                                                |
| $\mathbf{L}_o$ | Set of variables $b_o^v$ from a certain rule                                                                                                                                                               |
| $\cup$         | Concatenation operation between two samples                                                                                                                                                                |
| $@$            | Operation between two vectors of the same size that consists of performing a conjunction between the corresponding elements of the vectors and, subsequently, a disjunction between the resulting elements |

# Contents

|            |                                                  |           |
|------------|--------------------------------------------------|-----------|
| <b>1</b>   | <b>INTRODUCTION . . . . .</b>                    | <b>17</b> |
| <b>1.1</b> | <b>Motivation and Overview . . . . .</b>         | <b>17</b> |
| <b>1.2</b> | <b>General and Specific Objectives . . . . .</b> | <b>19</b> |
| <b>1.3</b> | <b>Scientific Production . . . . .</b>           | <b>19</b> |
| <b>1.4</b> | <b>Dissertation Structure . . . . .</b>          | <b>19</b> |
| <b>2</b>   | <b>PRELIMINARIES . . . . .</b>                   | <b>21</b> |
| <b>3</b>   | <b>RULE LEARNING METHODS . . . . .</b>           | <b>25</b> |
| <b>3.1</b> | <b>SQFSAT . . . . .</b>                          | <b>25</b> |
| <b>3.2</b> | <b>IMLI . . . . .</b>                            | <b>26</b> |
| <b>3.3</b> | <b>Iterative Version of IMLI . . . . .</b>       | <b>28</b> |
| <b>4</b>   | <b>IMLIB . . . . .</b>                           | <b>31</b> |
| <b>4.1</b> | <b>I-IMLIB . . . . .</b>                         | <b>35</b> |
| <b>5</b>   | <b>EXPERIMENTS AND RESULTS . . . . .</b>         | <b>37</b> |
| <b>6</b>   | <b>CONCLUSION . . . . .</b>                      | <b>45</b> |
|            | <b>BIBLIOGRAPHY . . . . .</b>                    | <b>47</b> |

# 1 Introduction

## 1.1 Motivation and Overview

The success of Machine Learning (ML) in recent years has led to a growing advancement in studies in this area (CARLEO et al., 2019; HUANG et al., 2021; JANIESCH; ZSCHECH; HEINRICH, 2021). Several applications have emerged with the aim of circumventing various problems and situations (GHASSEMI; OAKDEN-RAYNER; BEAM, 2021; KWEKHA-RASHID; ABDULJABBAR; ALHAYANI, 2021; SHARMA et al., 2020). One such problem is the lack of explainability of prediction models. This directly affects the reliability of using these applications in critical situations involving, for example, finance, autonomous systems, damage to equipment, the environment, and even lives (BIRAN; COTTON, 2017; GUNNING et al., 2019; YAN et al., 2020). That said, some works seek to develop approaches that bring explainability to their predictions (JIMÉNEZ-LUNA; GRISONI; SCHNEIDER, 2020; TJOA; GUAN, 2020; VILONE; LONGO, 2021).

Precise predictions with high levels of interpretability are often not a simple task. There are some works that try to solve this problem by balancing accuracy of predictions with interpretability (IGNATIEV et al., 2021; MITA et al., 2020; YU et al., 2020; GHOSH; MEEL, 2019; MALIOUTOV; MEEL, 2018; LAKKARAJU; BACH; LESKOVEC, 2016). It can be seen that some of these works use approaches based on the Boolean Satisfiability Problem (SAT) and the Maximum Boolean Satisfiability Problem (MaxSAT). The choice of these approaches to solve this problem has been increasingly recurrent in recent years. The reasons can be seen in the results obtained by these models.

SAT-based approaches have been proposed recently (ROCHA; MARTINS, 2019; ROCHA; MARTINS; FERREIRA, 2020) to learn quantifier-free first-order sentences from a set of classified strings. More specifically, given a set of classified strings, the goal is to find a first-order sentence over strings of minimum size that correctly classifies all the strings. One of the approaches demonstrated is SQFSAT (Synthesis of quantifier-free first-order sentences over strings with SAT). Upon receiving a set of classified strings, this approach generates a quantifier-free first-order sentence over strings in disjunctive normal form (DNF) with a given number of terms. What makes this method stand out is the fact that we can limit both the number of terms and the number of literals per term in the generated formula. In addition, as the approach generates formulas in DNF, each term of the formula can be seen as a rule. Then, for each rule, its explanation is the conjunction of formulas in the rule, which can be interesting for their interpretability (IGNATIEV et al., 2018; ROCHA; MARTINS, 2019). On the other hand, as the model is based on the SAT problem, in certain situations it may bring results that are not so interesting in terms of

interpretability and efficiency, such as in cases where the set of strings is large.

Ghosh, B. et al. created a classification model based on MaxSAT called IMLI (GHOSH; MEEL, 2019). The approach takes the number of rules and a set of classified samples, represented by vectors of numerical and categorical data, and generates a set of rules expressed in DNF or in conjunctive normal form (CNF) that correctly classifies as many samples as possible. In this work, we focus on using IMLI for learning rules in DNF. While the number of rules in the set of rules can be defined similarly to SQFSAT, IMLI does not consider the number of elements per rule. Although IMLI focuses on learning a sparse set of rules, it may obtain a combination of both large and small rules. IMLI also takes into account the option of defining a weighting for correct classifications. As the weighting increases, the accuracy of the model in the training data improves, but at the cost of an increase in the size of the generated set of rules. The smaller the weighting, the lower the accuracy of the model in the training data, but correspondingly, the generated set of rules tends to be smaller. Furthermore, IMLI uses an incremental approach to achieve better runtime performance. The incremental form consists of dividing the set of samples into partitions in order to generate a set of rules for each partition from the set of rules obtained in the previous partitions.

Ghosh, B. et al. also created an iterative version of IMLI (GHOSH; MALIOUTOV; MEEL, 2022). The iterative version involves training on a set of samples, continually attempting to generate a single rule. Samples correctly classified by the generated rule are then removed from the set of samples. Subsequently, the training is executed again with the remaining set of samples. The newly obtained rule is concatenated with the previously obtained rule. This process is repeated until there are no more samples in the set or until the maximum defined number of rules is reached. The idea is that by training while generating only one rule at a time, the model creates a smaller number of variables, making it easier for the MaxSAT solver to find a solution.

In this work, we aim to create a new approach for learning interpretable rules based on MaxSAT that unites SQFSAT with the incrementability of IMLI. The motivation for choosing SQFSAT is the possibility of defining the number of literals per clause, allowing us to generate smaller and more balanced rules. The choice of IMLI is motivated by its incrementability technique, which allows the method to train on large sets of samples efficiently. In addition, also based on IMLI, we propose a technique that reduces the size of the generated rules, removing possible redundancies, and an iterative version of the approach.

The experiments conducted on ten datasets showed promising results. The approach proposed in this work generates sets of rules that are more balanced, maintaining, and in various situations, surpassing the accuracy of IMLI. The iterative version of the proposed approach also achieved interesting results. It managed to reduce training time while

maintaining or surpassing the accuracy of its incremental version in most of the tested datasets.

## 1.2 General and Specific Objectives

This dissertation aims to enhance interpretable classification models based on SAT and MaxSAT in terms of interpretability, performance, and accuracy.

Therefore, the specific objectives are:

- Adapt the SQFSAT model to enable the application of IMLI’s incremental training technique, generating a new approach;
- Apply pruning techniques to the set of rules generated by the new approach;
- Apply the iterative training technique to the new approach;
- Evaluate the results obtained by the new approach and its iterative version, comparing them with IMLI and its iterative version in terms of: number of rules, number of features in the set of rules, size of the largest rule, accuracy on test data, training time and rule size balancing;

## 1.3 Scientific Production

During this master’s project, the following scientific papers were accepted and published:

- **Júnior, A. C. S. F., & Rocha, T. A., (2022, October). Incorporando Aprendizado Incremental com MaxSAT em um modelo de Aprendizado de Regras baseado em SAT.** In Congresso Brasileiro de Automática-CBA (Vol. 3, No. 1);
- **Ferreira Júnior, A. C. S., & Rocha, T. A., (2023, September). An Incremental MaxSAT-Based Model to Learn Interpretable and Balanced Classification Rules.** In Brazilian Conference on Intelligent Systems (pp. 227-242). Cham: Springer Nature Switzerland.

In addition to the works already published, we have another in the writing process that will still be submitted.

## 1.4 Dissertation Structure

This work is divided into 6 chapters. In Chapter 2, we define the general notions and notations. Since all methods presented in this work use Boolean logic, we also explain in Chapter 2 how these methods binarize datasets with numerical and categorical data.

In Chapter 3, SQFSAT, IMLI, and their iterative versions are presented, respectively. We present SQFSAT in the context of our work, where samples consist of binary vectors instead of strings, and elements of rules are boolean literals instead of are first-order sentences over strings. In Chapter 4, our contribution, IMLIB and its iterative version, are presented. In Chapter 5, we describe the experiments conducted and the results for the comparison of our approach against IMLI and its iterative version. Finally, in the last chapter, we present the conclusions and outline future work.

## 2 Preliminaries

We consider binary classification problems where we are given a set of samples and their classifications. The set of samples is represented by a binary matrix of size  $n \times m$  and their classifications by a vector of size  $n$ . We call the matrix  $\mathbf{X}$  and the vector  $\mathbf{y}$ . Each row of  $\mathbf{X}$  is a sample of the set and we will call it  $\mathbf{X}_i$  with  $i \in \{1, \dots, n\}$ . To represent a specific value of  $\mathbf{X}_i$ , we will use  $x_{i,j}$  with  $j \in \{1, \dots, m\}$ . Each column of  $\mathbf{X}$  has a label representing a feature and the label is symbolized by  $x^j$ . To represent a specific value of  $\mathbf{y}$ , we will use  $y_i$ .

To represent the opposite value of  $y_i$ , that is, if it is 1 the opposite value is 0 and vice versa, we use  $\neg y_i$ . Therefore, we will use the symbol  $\neg \mathbf{y}$  to represent  $\mathbf{y}$  with all opposite values. To represent the opposite value of  $x_{i,j}$ , we use  $\neg x_{i,j}$ . Therefore, we will use the symbol  $\neg \mathbf{X}_i$  to represent  $\mathbf{X}_i$  with all opposite values. Each label also has its opposite label which is symbolized by  $\neg x^j$ .

A partition in the samples of  $\mathbf{X}$  is represented by  $\mathbf{X}^t$  with  $t \in \{1, \dots, p\}$ , where  $p$  is the number of partitions. Therefore, the partitions of vector  $\mathbf{y}$  are represented by  $\mathbf{y}^t$ . Each element of  $\mathbf{y}$  is symbolized by  $y_i$  and represents the class value of sample  $\mathbf{X}_i$ . Partitions must maintain the association of each  $\mathbf{X}_i$  with its respective  $y_i$ . We use  $\mathcal{E}^- = \{\mathbf{X}_i \mid y_i = 0, 1 \leq i \leq n\}$  and  $\mathcal{E}^+ = \{\mathbf{X}_i \mid y_i = 1, 1 \leq i \leq n\}$ . To represent the size of these sets, that is, the number of samples contained in them, we use the notations:  $|\mathcal{E}^-|$  and  $|\mathcal{E}^+|$ .

**Example 1.** Let  $\mathbf{X}$  be the set of samples

$$\mathbf{X} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

and their classifications  $\mathbf{y} = [1, 0, 0, 1]$ . The samples  $\mathbf{X}_i$  are:  $\mathbf{X}_1 = [0, 0, 1]$ , ...,  $\mathbf{X}_4 = [1, 0, 0]$ . The values of each sample  $x_{i,j}$  are:  $x_{1,1} = 0, x_{1,2} = 0, x_{1,3} = 1, x_{2,1} = 0, \dots, x_{4,3} = 0$ . The class values  $y_i$  of each sample are:  $y_1 = 1, \dots, y_4 = 1$ . We can divide  $\mathbf{X}$  into two partitions in several different ways, one of which is:  $\mathbf{X}^1 = \begin{bmatrix} 0 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$ ,  $\mathbf{y}^1 = [0, 0]$ ,  $\mathbf{X}^2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$  e  $\mathbf{y}^2 = [1, 1]$ .

**Example 2.** Let  $\mathbf{X}$  be the set of samples from Example 1, then

$$\neg \mathbf{X} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix}$$

and  $\neg \mathbf{y} = [0, 1, 1, 0]$ . The samples  $\neg \mathbf{X}_i$  are:  $\neg \mathbf{X}_1 = [1, 1, 0], \dots, \neg \mathbf{X}_4 = [0, 1, 1]$ . The values of each sample  $\neg x_{i,j}$  are:  $\neg x_{1,1} = 1, \neg x_{1,2} = 1, \neg x_{1,3} = 0, \neg x_{2,1} = 1, \dots, \neg x_{4,3} = 1$ . The class values of each sample  $\neg y_i$  are:  $\neg y_1 = 0, \dots, \neg y_4 = 0$ . We can divide  $\neg \mathbf{X}$  in partitions as in

Example 1:  $\neg \mathbf{X}^1 = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix}$ ,  $\neg \mathbf{y}^1 = [1, 1]$ ,  $\neg \mathbf{X}^2 = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix}$  e  $\neg \mathbf{y}^2 = [0, 0]$ .

We define a set of rules being a disjunction of rules and is represented by  $\mathbf{R}$ . A rule is a conjunction of one or more features  $x^j$ . Each rule in  $\mathbf{R}$  is represented by  $R_o$  with  $o \in \{1, \dots, k\}$ , where  $k$  is the number of rules. Moreover,  $\mathbf{R}(\mathbf{X}_i)$  represents the application of  $\mathbf{R}$  to  $\mathbf{X}_i$ . The notations  $|\mathbf{R}|$  and  $|R_o|$  are used to represent the number of features in  $\mathbf{R}$  and  $R_o$ , respectively.

**Example 3.** Let  $x^1 = \text{Man}$ ,  $x^2 = \text{Smoke}$ ,  $x^3 = \text{Hike}$  be labels of features. Let  $\mathbf{R}$  be the set of rules  $\mathbf{R} = (\text{Man}) \vee (\text{Smoke} \wedge \neg \text{Hike})$ . The rules  $R_o$  are:  $R_1 = (\text{Man})$  and  $R_2 = (\text{Smoke} \wedge \neg \text{Hike})$ . The application of  $\mathbf{R}$  to  $\mathbf{X}_i$  is represented as follows:  $\mathbf{R}(\mathbf{X}_i) = x_{i,1} \vee (x_{i,2} \wedge \neg x_{i,3})$ . For example, let  $\mathbf{X}$  be the set of samples from Example 1, then:  $\mathbf{R}(\mathbf{X}_1) = x_{1,1} \vee (x_{1,2} \wedge \neg x_{1,3}) = 0 \vee (0 \wedge 0) = 0$ . Moreover, we have that  $|\mathbf{R}| = 3$ ,  $|R_1| = 1$  and  $|R_2| = 2$ .

As we assume a set of binary samples, we need to perform some preprocessing for binarizing sets of samples with numerical or categorical values. The algorithm divides the features into four types: constant, where all samples have the same value; binary, where there are only two distinct variations among all the samples for the same value; categorical, when the feature does not fit in constant and binary and its values are three or more categories; ordinal, when the feature does not fit into constant and binary and has numerical values.

When the feature type is constant, the algorithm discards that feature. This happens due to the fact that a feature common to all samples makes no difference in the generated rules. When the type is binary, one of the feature variations will receive 0 and the other 1 as new values. If the type is categorical, we employ the widely known technique of one-hot encoding. Finally, for the ordinal type feature, a quantization is performed, that is, the variations of this feature are divided into quantiles. With this, Boolean values are assigned to each quantile according to the original value.

We use SAT and MaxSAT solvers to implement the methods presented in this work. A solver receives a formula in CNF, for example:  $(p \vee q) \wedge (q \vee \neg p)$ . Furthermore, a MaxSAT

solver receives weights that will be assigned to each clause in the formula. A clause is the disjunction of one or more literals. The weights are represented by  $W(Cl) = w$  where  $Cl$  is one or more clauses and  $w$  represents the weight assigned to each one of them. A SAT solver tries to assign values to the literals in such a way that all clauses are satisfied. If this is not possible, the solver determines that there is no solution. Informally, a MaxSAT solver assigns values to the literals in a way that the sum of the weights of satisfied clauses is maximum. Clauses with numerical weights are considered *soft*. The greater the weight, the greater the priority of the clause to be satisfied. Clauses assigned a weight of  $\infty$  are considered *hard* and must be satisfied. Therefore, a MaxSAT solver returns a solution that maximizes the sum of the weights of satisfied clauses. For details on propositional logic, SAT solvers and MaxSAT solvers, see (BIERE et al., 2021).

## 3 Rule Learning Methods

### 3.1 SQFSAT

SQFSAT is a SAT-based approach that, given  $\mathbf{X}$ ,  $\mathbf{y}$  and positive integers  $k$ ,  $l$ , tries to find a set  $\mathbf{R}$  with  $k$  rules and at most  $l$  features per rule that correctly classify all samples  $\mathbf{X}_i$ , that is,  $\mathbf{R}(\mathbf{X}_i) = y_i$  for all  $i$  (ROCHA; MARTINS, 2019; ROCHA; MARTINS; FERREIRA, 2020). In general, the approach takes its parameters  $\mathbf{X}$ ,  $\mathbf{y}$ ,  $k$  and  $l$  and constructs a CNF formula to apply it to a SAT solver, which returns an answer that is used to get  $\mathbf{R}$ .

The construction of the SAT clauses is defined by propositional variables:  $u_{o,d}^j$ ,  $p_{o,d}$ ,  $u_{o,d}^*$ ,  $e_{o,d,i}$  and  $z_{o,i}$ , for  $d \in \{1, \dots, l\}$ . If the valuation of  $u_{o,d}^j$  is true, it means that  $j$ th feature label will be the  $d$ th feature of the rule  $R_o$ . Furthermore, if  $p_{o,d}$  is true, it means that the  $d$ th feature of the rule  $R_o$  will be  $x^j$ , in other words, will be positive. Otherwise, it will be negative:  $\neg x^j$ . If  $u_{o,d}^*$  is true, it means that the  $d$ th feature is skipped in the rule  $R_o$ . In this case, we ignore  $p_{o,d}$ . If  $e_{o,d,i}$  is true, then the  $d$ th feature of rule  $R_o$  contributes to the correct classification of the  $i$ th sample. If  $z_{o,i}$  is true, then the rule  $R_o$  contributes to the correct classification of the  $i$ th sample. That said, below, we will show the constraints formulated in the model for constructing the SAT clauses.

Conjunction of clauses that guarantees that exactly one  $u_{o,d}^j$  is true for the  $d$ th feature of the rule  $R_o$ :

$$A = \bigwedge_{\substack{o \in \{1, \dots, k\} \\ d \in \{1, \dots, l\}}} \bigvee_{j \in \{1, \dots, m, *\}} u_{o,d}^j \quad (3.1)$$

$$B = \bigwedge_{\substack{o \in \{1, \dots, k\} \\ d \in \{1, \dots, l\} \\ j, j' \in \{1, \dots, m, *\}, j \neq j'}} (\neg u_{o,d}^j \vee \neg u_{o,d}^{j'}) \quad (3.2)$$

Conjunction of clauses that ensures that each rule has at least one feature:

$$C = \bigwedge_{o \in \{1, \dots, k\}} \bigvee_{d \in \{1, \dots, l\}} \neg u_{o,d}^* \quad (3.3)$$

We will use the symbol  $s_{o,d,i}^j$  to represent  $e_{o,d,i}$  or  $\neg e_{o,d,i}$  depending on the value of the  $i$ th sample in the  $j$ th feature label of  $\mathbf{X}$ . If this value is 1, it means that if the  $j$ th feature label is in the  $d$ th position of the rule  $R_o$ , then it contributes to the correct classification of the  $i$ th sample. Therefore,  $s_{o,d,i}^j = e_{o,d,i}$ . Otherwise,  $s_{o,d,i}^j = \neg e_{o,d,i}$ . That said, the following conjunction of formulas guarantees that  $e_{o,d,i}$  is true if the  $j$ th feature

in the  $o$ th rule contributes to the correct classification of the sample  $\mathbf{X}_i$ :

$$D = \bigwedge_{\substack{o \in \{1, \dots, k\} \\ d \in \{1, \dots, l\} \\ j \in \{1, \dots, m\} \\ i \in \{1, \dots, n\}}} u_{o,d}^j \rightarrow (p_{o,d} \leftrightarrow s_{o,d,i}^j) \quad (3.4)$$

Conjunction of formulas guaranteeing that if the  $d$ th feature of a rule is skipped, then the classification of this rule is not interfered by this feature:

$$E = \bigwedge_{\substack{o \in \{1, \dots, k\} \\ d \in \{1, \dots, l\} \\ i \in \{1, \dots, n\}}} u_{o,d}^* \rightarrow e_{o,d,i} \quad (3.5)$$

Conjunction of formulas indicating that  $z_{o,i}$  will be set to true if and only if all the features of rule  $R_o$  contribute to the correct classification of sample  $\mathbf{X}_i$ :

$$F = \bigwedge_{o \in \{1, \dots, k\}} \bigwedge_{i \in \{1, \dots, n\}} z_{o,i} \leftrightarrow \bigwedge_{d \in \{1, \dots, l\}} e_{o,d,i} \quad (3.6)$$

Conjunction of clauses that guarantees that  $\mathbf{R}$  will correctly classify all samples:

$$G = \bigwedge_{i \in \mathcal{E}^+} \bigvee_{o \in \{1, \dots, k\}} z_{o,i} \quad (3.7)$$

$$H = \bigwedge_{i \in \mathcal{E}^-} \bigwedge_{o \in \{1, \dots, k\}} \neg z_{o,i} \quad (3.8)$$

Next, the formula  $Q$  below is converted to CNF and given as input to a SAT solver. If the solver returns an assignment that satisfies the input formula, then  $\mathbf{R}$  can be constructed based on the values of variables  $u_{o,d}^j$ ,  $p_{o,d}$  and  $u_{o,d}^*$ .

$$Q = A \wedge B \wedge C \wedge D \wedge E \wedge F \wedge G \wedge H \quad (3.9)$$

## 3.2 IMLI

IMLI is an incremental approach based on MaxSAT for learning interpretable rules (GHOSH; MEEL, 2019). Informally, given  $\mathbf{X}$ ,  $\mathbf{y}$ , a positive integer with at most  $k$ , and a weight  $\lambda$ , the model aims to obtain the smallest set  $\mathbf{M}$  with at most  $k$  rules in CNF that correctly classifies as many samples as possible, penalizing each classification errors with  $\lambda$ . In general, the method solves the optimization problem  $\min_{\mathbf{M}} \{|\mathbf{M}| + \lambda|\mathcal{E}_M| \mid \mathcal{E}_M = \{\mathbf{X}_i \mid \mathbf{M}(\mathbf{X}_i) \neq y_i\}\}$ , where  $|\mathbf{M}|$  represents the number of features in  $\mathbf{M}$  and  $\mathbf{M}(\mathbf{X}_i)$  denotes the application of the set of rules  $\mathbf{M}$  to  $\mathbf{X}_i$ . Therefore, the approach takes its parameters  $\mathbf{X}$ ,  $\mathbf{y}$ ,  $k$  and  $\lambda$  and constructs a MaxSAT query to apply it to a MaxSAT solver, which returns an answer that is used to generate  $\mathbf{M}$ . Note that IMLI generates set of rules in

CNF, whereas our objective in this work is to obtain sets of rules in DNF. For that, we will have to use as parameter  $\neg \mathbf{y}$  instead of  $\mathbf{y}$  and negate the set of rules  $\mathbf{M}$  to obtain a set of rules  $\mathbf{R}$  in DNF.

The construction of the MaxSAT clauses is defined by propositional variables:  $b_o^v$  and  $\eta_i$ , for  $v \in \{1, \dots, 2m\}$ . The  $v$  ranges from 1 to  $2m$ , as it also considers opposite features. If the valuation of  $b_o^v$  is true and  $v \leq m$ , it means that feature  $x^v$  will be in the rule  $M_o$ , where  $M_o$  is the  $o$ th rule of  $\mathbf{M}$ . If the valuation of the  $b_o^v$  is true and  $v > m$ , it means that feature  $\neg x^{v-m}$  will be in the rule  $M_o$ . If the valuation of  $\eta_i$  is true, it means that sample  $\mathbf{X}_i$  is not classified correctly, that is,  $\mathbf{M}(\mathbf{X}_i) \neq y_i$ . That said, below, we will show the constraints for constructing MaxSAT clauses.

Constraints that represent that the cost of a misclassification is  $\lambda$ :

$$A = \bigwedge_{i \in \{1, \dots, n\}} \neg \eta_i, W(A) = \lambda \quad (3.10)$$

Constraints that represent that the model tries to insert as few features as possible in  $\mathbf{M}$ , taking into account the weights of all clauses:

$$B = \bigwedge_{\substack{v \in \{1, \dots, 2m\} \\ o \in \{1, \dots, k\}}} \neg b_o^v, W(B) = 1 \quad (3.11)$$

Even though the constraints in 3.11 prioritize learning sparse rules, they do so by directing attention to the overall set of rules, i.e. in the total number of features in  $\mathbf{M}$ . Then, IMLI may generate a set of rules that comprises a combination of both large and small rules. In our approach presented in Chapter 4, we address this drawback by limiting the number of features in each rule.

We will use  $\mathbf{L}_o$  to represent the set of variables  $b_o^v$  of a rule  $M_o$ , that is,  $\mathbf{L}_o = \{b_o^v \mid v \in \{1, \dots, 2m\}\}$ , for  $o \in \{1, \dots, k\}$ . To represent the concatenation of two samples, we will use the symbol  $\cup$ . We also use the symbol  $@$  to represent an operation between two vectors of the same size. The operation consists of applying a conjunction between the corresponding elements of the vectors. Subsequently, a disjunction between the elements of the result is applied. The following example illustrates how these definitions will be used:

**Example 4.** Let be  $\mathbf{X}_4$  as in Example 1,  $\mathbf{X}_4 \cup \neg \mathbf{X}_4 = [1, 0, 0, 0, 1, 1]$  and  $\mathbf{L}_o = [b_o^1, b_o^2, b_o^3, b_o^4, b_o^5, b_o^6]$ . Therefore,  $(\mathbf{X}_4 \cup \neg \mathbf{X}_4) @ \mathbf{L}_o = (x_{4,1} \wedge b_o^1) \vee (x_{4,2} \wedge b_o^2) \vee \dots \vee (\neg x_{4,6} \wedge b_o^6) = (1 \wedge b_o^1) \vee (0 \wedge b_o^2) \vee \dots \vee (1 \wedge b_o^6) = b_o^1 \vee b_o^5 \vee b_o^6$ .

The objective of this operation is to generate a disjunction of variables that indicates if any of the features associated with these variables are present in  $M_o$ , then sample  $\mathbf{X}_i$  will be correctly classified by  $M_o$ . Now, we can show the formula that guarantees that if  $\eta_i$  is false, then  $\mathbf{M}(\mathbf{X}_i) = y_i$ :

$$C = \bigwedge_{i \in \{1, \dots, n\}} \neg \eta_i \rightarrow (y_i \leftrightarrow \bigwedge_{o \in \{1, \dots, k\}} ((\mathbf{X}_i \cup \neg \mathbf{X}_i) @ \mathbf{L}_o)), W(C) = \infty \quad (3.12)$$

We can see that  $C$  is not in CNF. Therefore, formula  $Q$  below must be converted to CNF. With that, finally, we have the MaxSAT input that is sent to the solver. Then,  $\mathbf{M}$  can be constructed based on the values of variables  $b_o^v$ .

$$Q = A \wedge B \wedge C \quad (3.13)$$

The set of samples  $\mathbf{X}$ , in IMLI, can be divided into  $p$  partitions:  $\mathbf{X}^1, \mathbf{X}^2, \dots, \mathbf{X}^p$ . Each partition, but the last one, contains  $\lceil \frac{|\mathcal{E}^-|}{p} \rceil$  samples  $\mathbf{X}_i$  such that  $y_i = 0$  and  $\lceil \frac{|\mathcal{E}^+|}{p} \rceil$  samples  $\mathbf{X}_i$  such that  $y_i = 1$ . Also, the samples are randomly distributed across the partitions. Partitioning aims to make the model perform better in generating the set of rules  $\mathbf{M}$ . Thus, the conjunction of clauses will be created from each partition  $\mathbf{X}^t$  in an incremental way, that is, the set of rules  $\mathbf{M}$  obtained by the current partition will be reused for the next partition. In the first partition, constraints (3.10), (3.11), (3.12) are created in exactly the same way as described. From the second onwards, (3.11) is replaced by the following constraints:

$$B' = \bigwedge_{\substack{v \in \{1, \dots, 2m\} \\ o \in \{1, \dots, k\}}} \begin{cases} b_o^v, & \text{if } b_o^v \text{ is true in the previous partition;} \\ \neg b_o^v, & \text{otherwise.} \end{cases}, W(B') = 1 \quad (3.14)$$

The IMLI also has a technique for reducing the size of the generated set of rules. The technique removes possible redundancies in ordinal features as the one in Example 5. In the original implementation of the model, the technique is applied at the end of each partition. In our implementation for the experiments in Chapter 5, this technique is applied only at the end of the last partition. The reason for this is training performance.

**Example 5.** Let  $\mathbf{R}$  be the following set of rules with redundancy in the same rule:

$$(Age > 18 \wedge Age > 20) \vee (Height \leq 2).$$

Then, the technique removes the redundancy and the following set of rules is obtained:

$$(Age > 20) \vee (Height \leq 2).$$

### 3.3 Iterative Version of IMLI

In this section, we will discuss the iterative version of IMLI (GHOSH; MALIOUTOV; MEEL, 2022). The main advantage of iterative learning is the creation of smaller MaxSAT input. In incremental learning, the model learns all clauses in a CNF formula together, while in iterative learning, the model learns a single clause of a CNF in each iteration. By combining incremental and iterative learning, we achieve a reduction in the number of samples in the training data and a reduction in the number of clauses to be sent to the MaxSAT solver.

The iterative version of IMLI still receives the parameters  $\mathbf{X}$ ,  $\mathbf{y}$ ,  $k$ , and a weight  $\lambda$ . The difference begins in constraints (3.11), (3.12), and (3.14), more specifically in the index  $o$ . In the iterative version, the index  $o$  is replaced by the value 1. In other words, the model is trained to find  $\mathbf{R}$  with one rule. However, the training is done  $k$  times iteratively, combining all rules obtained in each iteration into one set of rules. In each iteration, the set of samples  $\{X_i \mid \mathbf{R}(X_i) = y_i, 1 \leq i \leq n\}$  and its predictions are removed from  $\mathbf{X}$  and  $\mathbf{y}$ , respectively, before the next iteration. Thus, iterations can stop in three situations: when the  $k$  iterations are completed, when  $\mathbf{X}$  and  $\mathbf{y}$  are empty, or when the set of rules obtained in the current iteration does not correctly classify any remaining samples. Note that in this way, the model generates a smaller number of variables for the solver. This assists the solver in finding solutions faster, potentially reducing training time.

## 4 IMLIB

In this chapter, we will present our method IMLIB which is an adaptation of SQFSAT for tabular data with an incremental approach based on MaxSAT. Like IMLI, IMLIB also has a technique to reduce the size of the set of generated rules. Since IMLIB is incremental, our approach also partitions the set of samples  $\mathbf{X}$ . Moreover, our method has one more constraint (4.3 bellow) than SQFSAT and a weight on all clauses. With that, informally, our approach receives five input parameters  $\mathbf{X}$ ,  $\mathbf{y}$ ,  $k$ ,  $l$ ,  $\lambda$  and tries to obtain the smallest  $\mathbf{R}$  that correctly classifies as many samples as possible, penalizing classification errors with  $\lambda$ . More formally, our method solves the optimization problem  $\min_{\mathbf{R}} \{|\mathbf{R}| + \lambda|\mathcal{E}_R| \mid \mathcal{E}_R = \{\mathbf{X}_i \mid \mathbf{R}(\mathbf{X}_i) \neq y_i\}\}$ . That said, below, we will show the constraints of our approach for constructing MaxSAT clauses.

Constraints that guarantee that exactly only one  $u_{o,d}^j$  is true for the  $d$ th feature of the rule  $R_o$ :

$$A = \bigwedge_{\substack{o \in \{1, \dots, k\} \\ d \in \{1, \dots, l\}}} \bigvee_{j \in \{1, \dots, m, *\}} u_{o,d}^j, W(A) = \infty \quad (4.1)$$

$$B = \bigwedge_{\substack{o \in \{1, \dots, k\} \\ d \in \{1, \dots, l\} \\ j, j' \in \{1, \dots, m, *\}, j \neq j'}} \neg u_{o,d}^j \vee \neg u_{o,d}^{j'}, W(B) = \infty \quad (4.2)$$

Constraints representing that the model will try to insert as few features as possible in  $\mathbf{R}$ :

$$C = \bigwedge_{\substack{o \in \{1, \dots, k\} \\ d \in \{1, \dots, l\} \\ j \in \{1, \dots, m\}}} \neg u_{o,d}^j \wedge \bigwedge_{\substack{o \in \{1, \dots, k\} \\ d \in \{1, \dots, l\}}} u_{o,d}^*, W(C) = 1 \quad (4.3)$$

Conjunction of clauses that guarantees that each rule has at least one feature:

$$D = \bigwedge_{o \in \{1, \dots, k\}} \bigvee_{d \in \{1, \dots, l\}} \neg u_{o,d}^*, W(D) = \infty \quad (4.4)$$

The following conjunction of formulas ensures that  $e_{o,d,i}$  is true if the  $j$ th feature label in the  $o$ th rule contributes to correctly classify sample  $\mathbf{X}_i$ :

$$E = \bigwedge_{\substack{o \in \{1, \dots, k\} \\ d \in \{1, \dots, l\} \\ j \in \{1, \dots, m\} \\ i \in \{1, \dots, n\}}} u_{o,d}^j \rightarrow (p_{o,d} \leftrightarrow s_{o,d,i}^j), W(E) = \infty \quad (4.5)$$

Conjunction of formulas guaranteeing that the classification of a specific rule will not be interfered by skipped features in the rule:

$$F = \bigwedge_{\substack{o \in \{1, \dots, k\} \\ d \in \{1, \dots, l\} \\ i \in \{1, \dots, n\}}} u_{o,d}^* \rightarrow e_{o,d,i}, W(F) = \infty \quad (4.6)$$

Conjunction of formulas indicating that the model assigns true to  $z_{o,i}$  if all the features of rule  $R_o$  support the correct classification of sample  $\mathbf{X}_i$ :

$$G = \bigwedge_{o \in \{1, \dots, k\}} \bigwedge_{i \in \{1, \dots, n\}} z_{o,i} \leftrightarrow \bigwedge_{d \in \{1, \dots, l\}} e_{o,d,i}, W(G) = \infty \quad (4.7)$$

Conjunction of clauses designed to generate a set of rules  $\mathbf{R}$  that correctly classify as many samples as possible:

$$H = \bigwedge_{i \in \mathcal{E}^+} \bigvee_{o \in \{1, \dots, k\}} z_{o,i}, W(H) = \lambda \quad (4.8)$$

$$I = \bigwedge_{i \in \mathcal{E}^-} \bigwedge_{o \in \{1, \dots, k\}} \neg z_{o,i}, W(I) = \lambda \quad (4.9)$$

Finally, after converting formula  $Q$  below to CNF, we have the MaxSAT query that is sent to the solver.

$$Q = A \wedge B \wedge C \wedge D \wedge E \wedge F \wedge G \wedge H \wedge I \quad (4.10)$$

Next, in Example 6, we will illustrate how the constraints are defined for a specific example:

**Example 6.** Let  $\mathbf{X}$  be the set of samples

$$\mathbf{X} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

and their classifications  $\mathbf{y} = [0, 1, 0]$ . Furthermore, let  $k = 2$  and  $l = 2$ . Then, formulas (4.1)-(4.9) for this instance are:

$$A = (u_{1,1}^1 \vee u_{1,1}^2 \vee u_{1,1}^3 \vee u_{1,1}^*) \wedge (u_{1,2}^1 \vee u_{1,2}^2 \vee u_{1,2}^3 \vee u_{1,2}^*) \wedge \\ (u_{2,1}^1 \vee u_{2,1}^2 \vee u_{2,1}^3 \vee u_{2,1}^*) \wedge (u_{2,2}^1 \vee u_{2,2}^2 \vee u_{2,2}^3 \vee u_{2,2}^*)$$

$$B = (\neg u_{1,1}^1 \vee \neg u_{1,1}^2) \wedge (\neg u_{1,1}^1 \vee \neg u_{1,1}^3) \wedge (\neg u_{1,1}^1 \vee \neg u_{1,1}^*) \wedge (\neg u_{1,1}^2 \vee \neg u_{1,1}^3) \wedge \\ (\neg u_{1,1}^2 \vee \neg u_{1,1}^*) \wedge (\neg u_{1,1}^3 \vee \neg u_{1,1}^*) \wedge (\neg u_{1,2}^1 \vee \neg u_{1,2}^2) \wedge (\neg u_{1,2}^1 \vee \neg u_{1,2}^3) \wedge \\ (\neg u_{1,2}^1 \vee \neg u_{1,2}^*) \wedge (\neg u_{1,2}^2 \vee \neg u_{1,2}^3) \wedge (\neg u_{1,2}^2 \vee \neg u_{1,2}^*) \wedge (\neg u_{1,2}^3 \vee \neg u_{1,2}^*) \wedge \\ (\neg u_{2,1}^1 \vee \neg u_{2,1}^2) \wedge (\neg u_{2,1}^1 \vee \neg u_{2,1}^3) \wedge (\neg u_{2,1}^1 \vee \neg u_{2,1}^*) \wedge (\neg u_{2,1}^2 \vee \neg u_{2,1}^3) \wedge \\ (\neg u_{2,1}^2 \vee \neg u_{2,1}^*) \wedge (\neg u_{2,1}^3 \vee \neg u_{2,1}^*) \wedge (\neg u_{2,2}^1 \vee \neg u_{2,2}^2) \wedge (\neg u_{2,2}^1 \vee \neg u_{2,2}^3) \wedge \\ (\neg u_{2,2}^1 \vee \neg u_{2,2}^*) \wedge (\neg u_{2,2}^2 \vee \neg u_{2,2}^3) \wedge (\neg u_{2,2}^2 \vee \neg u_{2,2}^*) \wedge (\neg u_{2,2}^3 \vee \neg u_{2,2}^*)$$

$$C = \neg u_{1,1}^1 \wedge \neg u_{1,1}^2 \wedge \neg u_{1,1}^3 \wedge \neg u_{1,2}^1 \wedge \neg u_{1,2}^2 \wedge \neg u_{1,2}^3 \wedge \neg u_{2,1}^1 \wedge \neg u_{2,1}^2 \wedge \neg u_{2,1}^3 \wedge \neg u_{2,2}^1 \wedge \neg u_{2,2}^2 \wedge \neg u_{2,2}^3 \wedge u_{1,1}^* \wedge u_{1,2}^* \wedge u_{2,1}^* \wedge u_{2,2}^*$$

$$D = (\neg u_{1,1}^* \vee \neg u_{1,2}^*) \wedge (\neg u_{2,1}^* \vee \neg u_{2,2}^*)$$

[illegible]

$$F = (\neg u_{1,1}^* \vee e_{1,1,1}) \wedge (\neg u_{1,1}^* \vee e_{1,1,2}) \wedge (\neg u_{1,1}^* \vee e_{1,1,3}) \wedge$$

$$(\neg u_{1,2}^* \vee e_{1,2,1}) \wedge (\neg u_{1,2}^* \vee e_{1,2,2}) \wedge (\neg u_{1,2}^* \vee e_{1,2,3}) \wedge$$

$$(\neg u_{2,1}^* \vee e_{2,1,1}) \wedge (\neg u_{2,1}^* \vee e_{2,1,2}) \wedge (\neg u_{2,1}^* \vee e_{2,1,3}) \wedge$$

$$(\neg u_{2,2}^* \vee e_{2,2,1}) \wedge (\neg u_{2,2}^* \vee e_{2,2,2}) \wedge (\neg u_{2,2}^* \vee e_{2,2,3})$$

$$\begin{aligned}
G = & (\neg z_{1,1} \vee e_{1,1,1}) \wedge (\neg z_{1,1} \vee e_{1,2,1}) \wedge (z_{1,1} \vee \neg e_{1,1,1} \vee \neg e_{1,2,1}) \wedge \\
& (\neg z_{1,2} \vee e_{1,1,2}) \wedge (\neg z_{1,2} \vee e_{1,2,2}) \wedge (z_{1,2} \vee \neg e_{1,1,2} \vee \neg e_{1,2,2}) \wedge \\
& (\neg z_{1,3} \vee e_{1,1,3}) \wedge (\neg z_{1,3} \vee e_{1,2,3}) \wedge (z_{1,3} \vee \neg e_{1,1,3} \vee \neg e_{1,2,3}) \wedge \\
& (\neg z_{2,1} \vee e_{2,1,1}) \wedge (\neg z_{2,1} \vee e_{2,2,1}) \wedge (z_{2,1} \vee \neg e_{2,1,1} \vee \neg e_{2,2,1}) \wedge \\
& (\neg z_{2,2} \vee e_{2,1,2}) \wedge (\neg z_{2,2} \vee e_{2,2,2}) \wedge (z_{2,2} \vee \neg e_{2,1,2} \vee \neg e_{2,2,2}) \wedge \\
& (\neg z_{2,3} \vee e_{2,1,3}) \wedge (\neg z_{2,3} \vee e_{2,2,3}) \wedge (z_{2,3} \vee \neg e_{2,1,3} \vee \neg e_{2,2,3})
\end{aligned}$$

$$H = (z_{1,2} \vee z_{2,2})$$

$$I = (\neg z_{1,1}) \wedge (\neg z_{2,1}) \wedge (\neg z_{1,3}) \wedge (\neg z_{2,3})$$

IMLIB can also partition the set of samples  $\mathbf{X}$  in the same way as IMLI. Therefore, all constraints described above are applied in the first partition. Starting from the second partition, the constraints in (4.3) are replaced by the following constraints:

$$C' = \bigwedge_{\substack{o \in \{1, \dots, k\} \\ d \in \{1, \dots, l\} \\ j \in \{1, \dots, m, *\}}} \begin{cases} u_{o,d}^j, & \text{if } u_{o,d}^j \text{ is true in the previous} \\ & \text{partition;} \\ \neg u_{o,d}^j, & \text{otherwise.} \end{cases}, W(C') = 1 \quad (4.11)$$

IMLIB also has a technique for reducing the size of the generated set of rules as demonstrated in Example 5. Moreover, we added two more cases which are described in Example 7 and Example 8.

**Example 7.** Let  $\mathbf{R}$  be the following set of rules with opposite features in the same rule:

$$(Age > 20) \vee (Height \leq 2 \wedge Height > 2) \vee (Hike \wedge Not\ Hike).$$

Therefore, the technique removes rules with opposite features in the same rule obtaining the following set of rules:

$$(Age > 20).$$

**Example 8.** Let  $\mathbf{R}$  be the following set of rules with the same feature occurring twice in a rule:

$$(Hike \wedge Hike) \vee (Age > 20).$$

Accordingly, our technique for removing redundancies eliminates repetitive features, resulting in the following set of rules:

$$(Hike) \vee (Age > 20).$$

## 4.1 I-IMLIB

The iterative version of IMLIB was inspired by the iterative version of IMLI described in Section 3.3. In the case of IMLIB, all constraints are altered in their iterative version. More specifically, in all of them, the index  $o$  is replaced by the value 1. In addition, the iterations will have the same stopping possibilities: when the  $k$  iterations are completed, when  $\mathbf{X}$  and  $\mathbf{y}$  are empty, or when the set of rules obtained in the current iteration does not correctly classify any remaining samples.

## 5 Experiments and Results

In this chapter, we present the experiments we conducted to compare our methods IMLIB and its iterative version I-IMLIB with IMLI and its iterative version I-IMLI, as well as the results obtained. All models were implemented<sup>1</sup> in Python and with the MaxSAT solver RC2 (IGNATIEV; MORGADO; MARQUES-SILVA, 2019). The experiments were performed on a machine with the following configurations: Intel(R) Core(TM) i7-1165G7 2.80GHz processor and 16GB of RAM. Ten datasets from the UCI repository (DUA; GRAFF, 2017) were used to compare the approaches. Information about the datasets can be seen in Table 1. Datasets with more than two classes were adapted, considering that both models are designed for binary classification. For comparison purposes, we measured the following metrics: number of rules, size of the set of rules, size of the largest rule, accuracy on test data, and training time. The number of rules, size of the set of rules, and size of the largest rule can be used as interpretability metrics. For example, a set of rules with few rules and small rules is more interpretable than one with large or many rules.

To collect the averages shown in Table 2, we performed 10 runs on each dataset. In each run, we split the dataset into 90% for training and 10% for testing. However, before training the models, we performed a validation to choose the best configuration. The validation involves splitting the training dataset into 90% for validation training and 10% for validation testing. All models are trained with the validation training data, and the configuration that achieves the highest accuracy on the validation testing data is used for training on the whole training data. Consequently, the computed results for each run come from training with the chosen configuration and evaluating the set of rules on the run's testing data.

For all models, we considered parameter configurations obtained by combining values of:  $k \in \{1, 2, 3\}$ ,  $\lambda \in \{5, 10\}$ , and  $lp \in \{8, 16\}$ , where  $lp$  is the number of samples per partition. Since IMLIB and I-IMLIB have the maximum number of features per rule  $l$  as an additional parameter, for each parameter configuration of IMLI, I-IMLI, and their corresponding sets of rules, we considered  $l$  ranging from 1 to one less than the size of the smallest rule among the largest rule in the set obtained by IMLI and the largest rule in the set obtained by I-IMLI. Thus, the value of  $l$  that resulted in the best accuracy on the validation testing data was chosen. Our objective with this is to evaluate whether IMLIB and I-IMLIB can achieve higher accuracy on the testing data compared to IMLI and I-IMLI by introducing smaller and more balanced rules. Furthermore, it is worth noting that this process does not exclude the possibility of our methods generating sets of

---

<sup>1</sup> The source code for IMLIB, I-IMLIB and the implementation of the conducted tests can be found at the link: <[https://github.com/cacajr/decision\\_set\\_models](https://github.com/cacajr/decision_set_models)>

Table 1 – Datasets information.

| Datasets    | Samples | $ \mathcal{E}^- $ | $ \mathcal{E}^+ $ | Features |
|-------------|---------|-------------------|-------------------|----------|
| lung cancer | 59      | 31                | 28                | 6        |
| iris        | 150     | 100               | 50                | 4        |
| parkinsons  | 195     | 48                | 147               | 22       |
| ionosphere  | 351     | 126               | 225               | 33       |
| wdbc        | 569     | 357               | 212               | 30       |
| transfusion | 748     | 570               | 178               | 4        |
| pima        | 768     | 500               | 268               | 8        |
| titanic     | 1309    | 809               | 500               | 6        |
| depressed   | 1429    | 1191              | 238               | 22       |
| mushroom    | 8112    | 3916              | 4208              | 22       |

rules and rules with sizes larger than those generated by IMLI and I-IMLI. For example, if IMLI or I-IMLI generates a set of rules with 3 rules of sizes 4, 5, 10. IMLIB and I-IMLIB can generate, among other possibilities, a set of rules with 3 rules of sizes 9, 9, 9.

Analyzing Table 2, we can see that, in terms of the size of the generated set of rules and the size of the largest rule, our approaches stand out in all datasets. Particularly, I-IMLIB achieved the best result more frequently. Although I-IMLI significantly improved these metrics compared to IMLI, the difference with I-IMLIB remains prominent. Additionally, our approaches maintained accuracy very close to, and even surpassed, IMLI and its iterative version, as seen in the lung cancer, parkinsons, ionosphere, transfusion, and depressed datasets. In terms of training time, I-IMLIB outperformed IMLIB in most datasets, but the spotlight is on IMLI and I-IMLI. Notably, I-IMLI achieved the best result in all datasets. This was expected, as the model generates fewer variables than IMLIB, I-IMLIB and is iterative.

To analyze the balance of the generated rules more accurately between the models, we conducted a second experiment. The experiment involved selecting the dataset where the models achieved the highest averages in terms of the size of the largest rule in Table 2, in this case, the mushroom dataset. Subsequently, we performed 10 runs for each configuration, using the same configurations described earlier in this chapter and employing the same process for choosing the parameter  $l$ . In each run, the chosen dataset was divided into 80% for training and 20% for testing. The configuration that obtained the highest average accuracy among all models was chosen to have its runs for each model exposed in Table 3 and Table 4. The idea of this experiment is to assess which of the models generates more balanced sets of rules, given the same configurations.

In Table 3, we can observe that the set of rules generated by the IMLIB approach are more balanced than those generated by IMLI. This characteristic can be interesting for the interpretability of a rule. For instance, in run 5, IMLI generated a set of rules with one rule containing 9 features. On the other hand, IMLIB generated a set of rules with 3 rules,

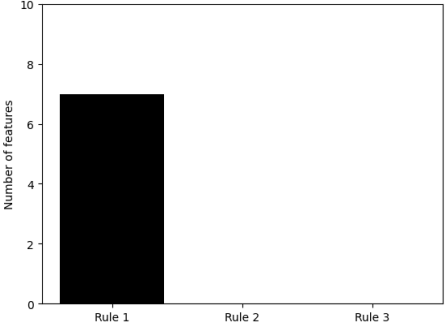
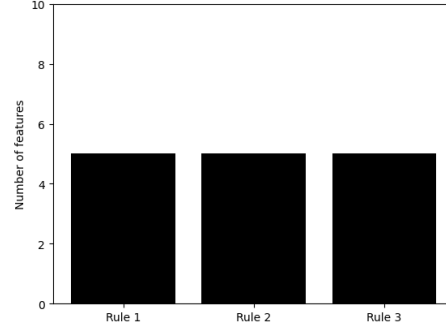
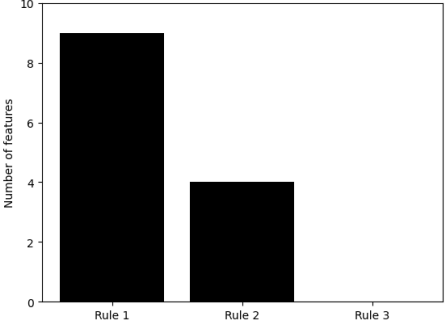
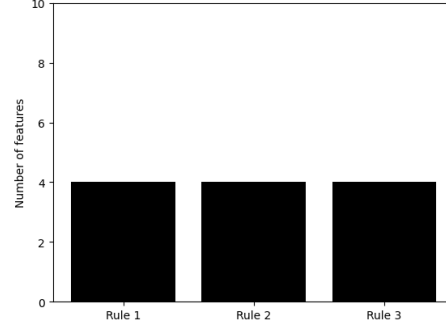
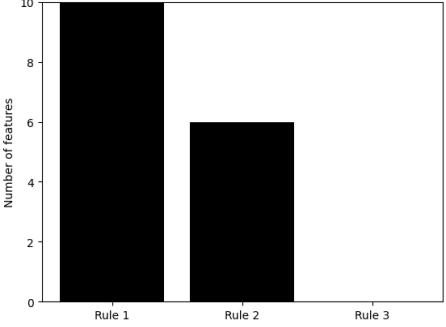
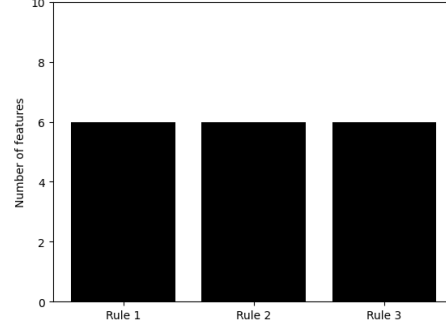
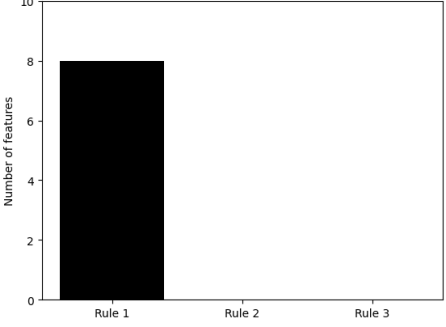
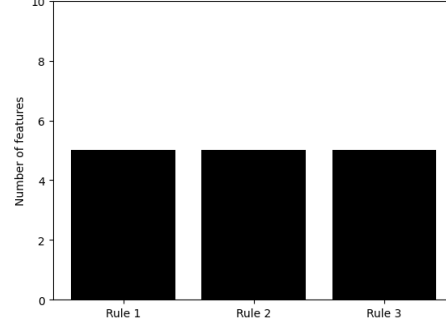
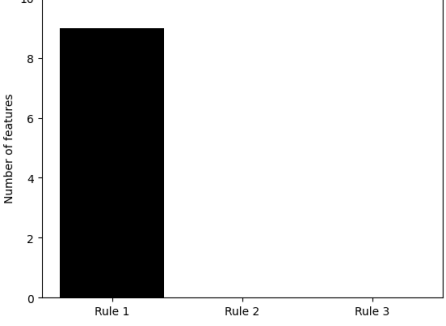
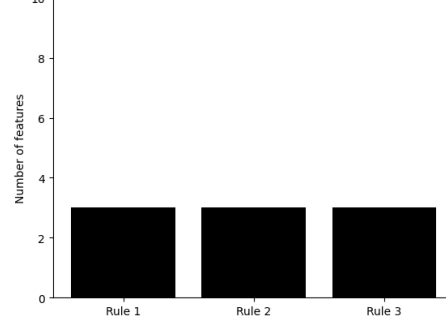
Table 2 – Comparison between IMLI, I-IMLI, IMLIB and I-IMLIB in different datasets. The column Training time represents the training time in seconds.

| Datasets    | Models  | Number of rules    | R                  | Largest rule size  | Accuracy           | Training time          |
|-------------|---------|--------------------|--------------------|--------------------|--------------------|------------------------|
| lung cancer | IMLI    | 1.70 ± 0.82        | 4.00 ± 1.56        | 3.10 ± 1.10        | 0.88 ± 0.11        | 0.0049 ± 0.0022        |
|             | I-IMLI  | <b>1.00 ± 0.00</b> | 2.90 ± 1.10        | 2.90 ± 1.10        | 0.90 ± 0.12        | <b>0.0029 ± 0.0005</b> |
|             | IMLIB   | 1.40 ± 0.52        | 1.70 ± 0.68        | 1.30 ± 0.48        | 0.87 ± 0.11        | 0.0055 ± 0.0027        |
|             | I-IMLIB | 1.10 ± 0.32        | <b>1.30 ± 0.48</b> | <b>1.20 ± 0.42</b> | <b>0.92 ± 0.09</b> | 0.0050 ± 0.0017        |
| iris        | IMLI    | 1.30 ± 0.68        | 4.90 ± 1.52        | 4.10 ± 0.74        | <b>0.87 ± 0.09</b> | 0.0028 ± 0.0012        |
|             | I-IMLI  | 1.20 ± 0.42        | 4.80 ± 1.62        | 4.20 ± 0.79        | 0.85 ± 0.11        | <b>0.0026 ± 0.0009</b> |
|             | IMLIB   | 1.40 ± 0.70        | 3.80 ± 1.62        | <b>2.90 ± 0.88</b> | 0.86 ± 0.10        | 0.0081 ± 0.0037        |
|             | I-IMLIB | <b>1.00 ± 0.00</b> | <b>2.90 ± 0.74</b> | <b>2.90 ± 0.74</b> | 0.83 ± 0.12        | 0.0065 ± 0.0009        |
| parkinsons  | IMLI    | 2.20 ± 0.63        | 5.50 ± 2.37        | 3.60 ± 1.27        | 0.77 ± 0.11        | 0.0171 ± 0.0041        |
|             | I-IMLI  | 1.70 ± 0.68        | 2.60 ± 1.35        | 1.80 ± 0.63        | 0.75 ± 0.14        | <b>0.0153 ± 0.0046</b> |
|             | IMLIB   | 2.10 ± 0.88        | 3.50 ± 2.32        | 1.70 ± 0.95        | 0.74 ± 0.15        | 0.0301 ± 0.0167        |
|             | I-IMLIB | <b>1.40 ± 0.52</b> | <b>1.50 ± 0.71</b> | <b>1.10 ± 0.32</b> | <b>0.79 ± 0.13</b> | 0.0203 ± 0.0191        |
| ionosphere  | IMLI    | 2.60 ± 0.52        | 8.80 ± 3.01        | 4.20 ± 1.23        | 0.76 ± 0.09        | 0.0525 ± 0.0116        |
|             | I-IMLI  | <b>1.20 ± 0.42</b> | 3.80 ± 2.39        | 3.20 ± 1.40        | 0.74 ± 0.11        | <b>0.0314 ± 0.0085</b> |
|             | IMLIB   | 1.80 ± 0.92        | 4.40 ± 4.50        | <b>2.20 ± 1.40</b> | 0.74 ± 0.11        | 0.1243 ± 0.1103        |
|             | I-IMLIB | 1.30 ± 0.48        | <b>2.90 ± 0.99</b> | 2.50 ± 0.97        | <b>0.77 ± 0.11</b> | 0.0919 ± 0.0347        |
| wdbc        | IMLI    | 1.80 ± 0.79        | 6.00 ± 3.16        | 3.60 ± 1.43        | <b>0.87 ± 0.04</b> | 0.0416 ± 0.0126        |
|             | I-IMLI  | 1.30 ± 0.68        | 4.10 ± 2.56        | 3.30 ± 1.25        | <b>0.87 ± 0.05</b> | <b>0.0410 ± 0.0189</b> |
|             | IMLIB   | 1.70 ± 0.48        | 2.90 ± 1.97        | <b>1.60 ± 0.84</b> | 0.84 ± 0.06        | 0.0813 ± 0.0487        |
|             | I-IMLIB | <b>1.10 ± 0.32</b> | <b>2.00 ± 1.63</b> | 1.70 ± 0.95        | 0.84 ± 0.08        | 0.0770 ± 0.0600        |
| transfusion | IMLI    | 1.90 ± 0.99        | 5.70 ± 3.53        | 3.30 ± 1.25        | 0.66 ± 0.08        | 0.0290 ± 0.0097        |
|             | I-IMLI  | <b>1.40 ± 0.52</b> | <b>3.40 ± 1.71</b> | 2.60 ± 0.97        | 0.67 ± 0.10        | <b>0.0265 ± 0.0080</b> |
|             | IMLIB   | 1.90 ± 0.74        | 4.90 ± 2.89        | 2.60 ± 0.97        | 0.63 ± 0.12        | 0.1310 ± 0.1317        |
|             | I-IMLIB | <b>1.40 ± 0.52</b> | <b>3.40 ± 2.17</b> | <b>2.40 ± 0.97</b> | <b>0.68 ± 0.09</b> | 0.1405 ± 0.1135        |
| pima        | IMLI    | 1.90 ± 0.88        | 8.90 ± 4.89        | 5.50 ± 2.27        | <b>0.64 ± 0.08</b> | 0.0502 ± 0.0170        |
|             | I-IMLI  | <b>1.20 ± 0.42</b> | 4.70 ± 2.41        | 4.10 ± 1.52        | 0.61 ± 0.11        | <b>0.0413 ± 0.0146</b> |
|             | IMLIB   | 1.90 ± 0.88        | 4.40 ± 3.10        | 2.50 ± 1.58        | 0.62 ± 0.06        | 0.3627 ± 0.2822        |
|             | I-IMLIB | 1.70 ± 0.82        | <b>3.20 ± 2.49</b> | <b>2.00 ± 1.25</b> | 0.63 ± 0.07        | 0.2721 ± 0.1924        |
| titanic     | IMLI    | 1.60 ± 0.84        | 9.80 ± 4.21        | 6.80 ± 1.03        | 0.67 ± 0.10        | 0.0687 ± 0.0277        |
|             | I-IMLI  | <b>1.40 ± 0.52</b> | 9.20 ± 3.74        | 6.60 ± 0.97        | <b>0.70 ± 0.09</b> | <b>0.0630 ± 0.0193</b> |
|             | IMLIB   | 1.80 ± 0.79        | <b>4.60 ± 3.24</b> | <b>2.70 ± 1.95</b> | 0.63 ± 0.11        | 1.6199 ± 1.4272        |
|             | I-IMLIB | 1.90 ± 0.88        | 7.30 ± 6.80        | 3.80 ± 2.66        | 0.69 ± 0.11        | 1.3627 ± 1.4872        |
| depressed   | IMLI    | 2.00 ± 0.94        | 9.10 ± 5.69        | 5.50 ± 2.42        | 0.66 ± 0.06        | 0.1308 ± 0.0482        |
|             | I-IMLI  | <b>1.10 ± 0.32</b> | 4.30 ± 2.21        | 4.10 ± 2.13        | 0.68 ± 0.06        | <b>0.0864 ± 0.0190</b> |
|             | IMLIB   | 2.10 ± 0.88        | 8.30 ± 6.33        | 3.80 ± 2.15        | <b>0.70 ± 0.06</b> | 0.5179 ± 0.3231        |
|             | I-IMLIB | 1.20 ± 0.42        | <b>3.80 ± 2.82</b> | <b>3.30 ± 2.41</b> | <b>0.70 ± 0.10</b> | 0.3561 ± 0.3245        |
| mushroom    | IMLI    | 2.40 ± 0.70        | 15.80 ± 7.24       | 7.80 ± 2.15        | 0.99 ± 0.01        | 0.2662 ± 0.1288        |
|             | I-IMLI  | 1.10 ± 0.32        | 9.30 ± 1.49        | 8.90 ± 0.74        | <b>1.00 ± 0.00</b> | <b>0.1177 ± 0.0369</b> |
|             | IMLIB   | 1.60 ± 0.84        | 9.30 ± 4.79        | <b>6.10 ± 0.99</b> | 0.96 ± 0.05        | 1.3706 ± 0.6522        |
|             | I-IMLIB | <b>1.00 ± 0.00</b> | <b>6.20 ± 0.79</b> | 6.20 ± 0.79        | 0.90 ± 0.10        | 2.2118 ± 1.2587        |

each having 3 features. Since each rule is independent, we argue that when interpreting a set of rules, it is easier to comprehend 3 rules of size 3 compared to a single rule of size 9.

In Table 4, the iterative version of IMLIB exhibited a different behavior. While some balance in the size of rules still appeared in runs 5, 7, and 9, I-IMLIB generated sets of rules with a few rules of small sizes. This can be explained by the fact that I-IMLIB always seeks to find a set of rules with size 1, adding more rules only when it is not possible to cover the maximum number of samples in the training set. This characteristic becomes evident in runs 6, 8, and 10. The clear reduction in rule sizes in the iterative version of IMLIB brought very interesting results for interpretability of sets of rules when compared to the other models in the same run, as we can see in runs 2, 3, and 4.

Table 3 – Comparison between IMLI and IMLIB in terms of balancing of the set of rules on the mushroom dataset. The configuration used was  $k = 3$ ,  $\lambda = 10$ , and  $lp = 16$ .

| Runs | $l$ | IMLI                                                                                | IMLIB                                                                                |
|------|-----|-------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|
| 1    | 5   |    |    |
| 2    | 4   |   |   |
| 3    | 6   |  |  |
| 4    | 5   |  |  |
| 5    | 3   |  |  |

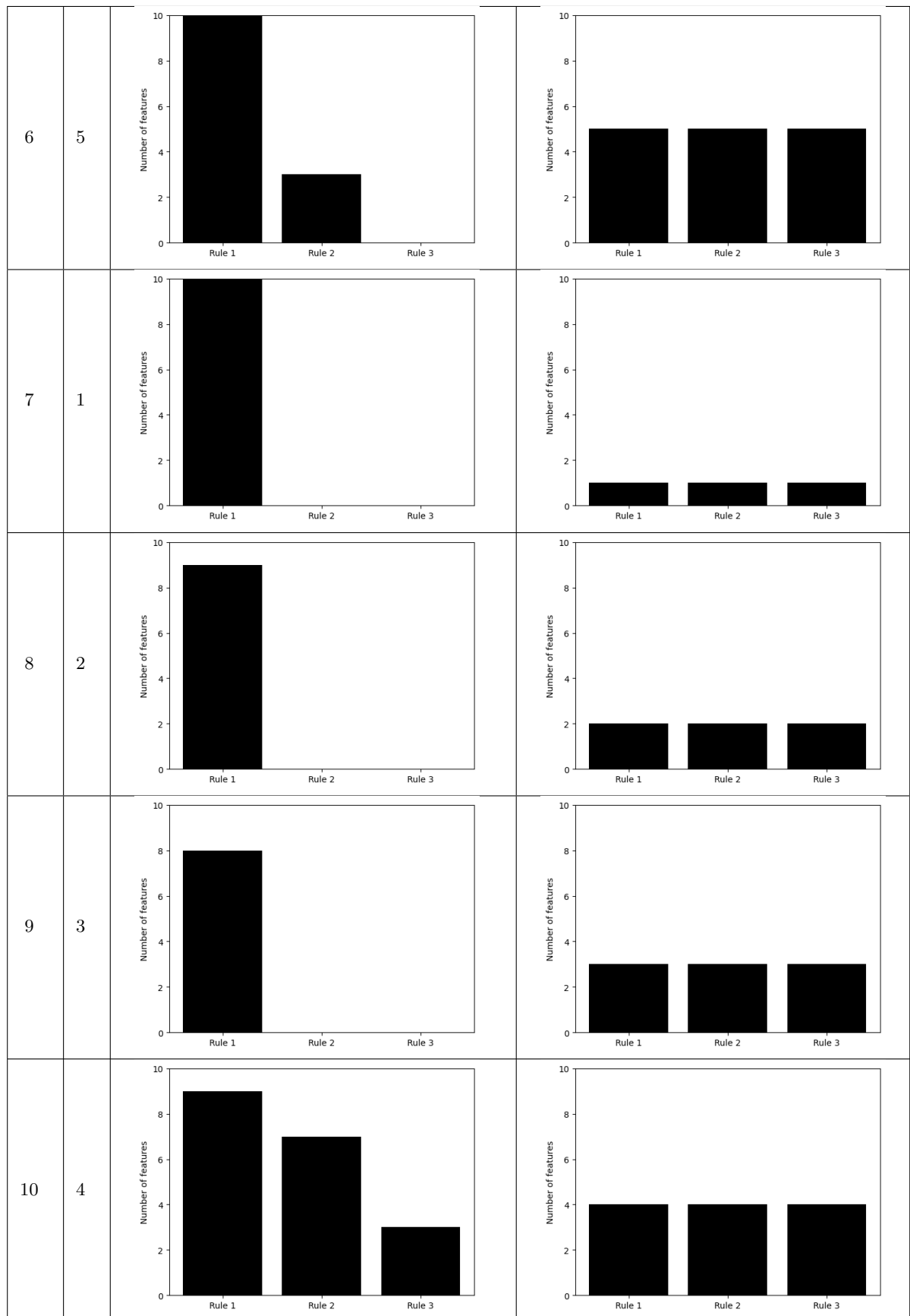
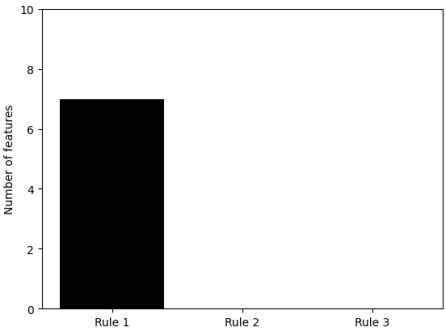
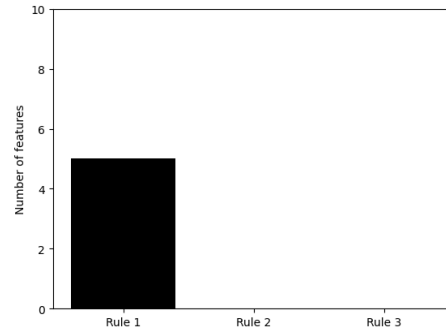
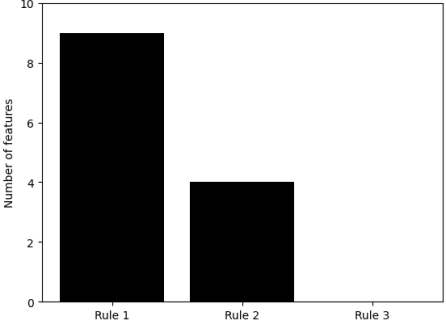
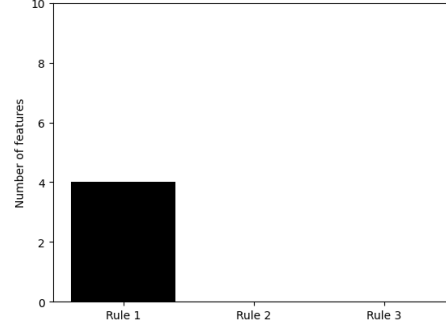
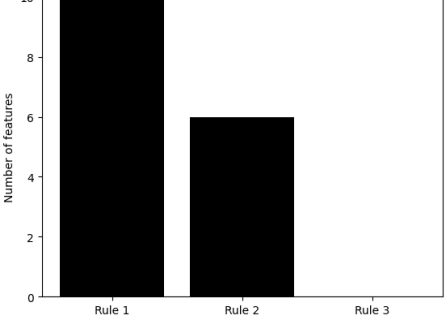
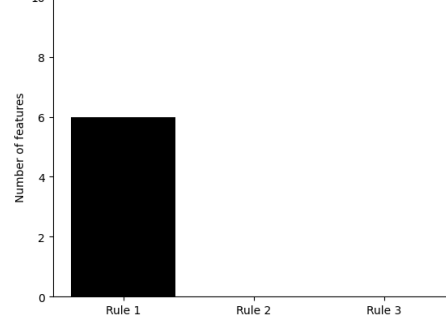
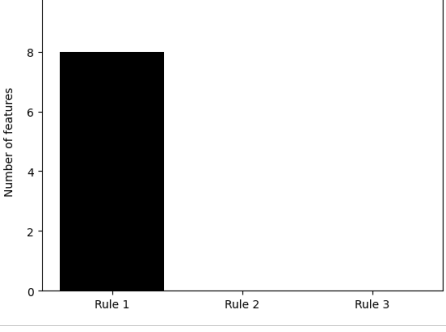
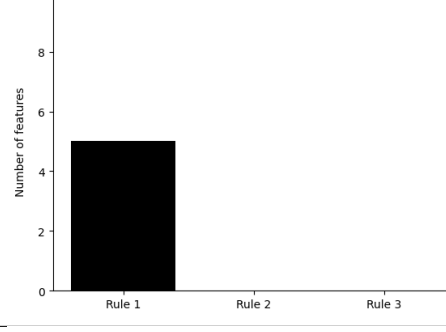
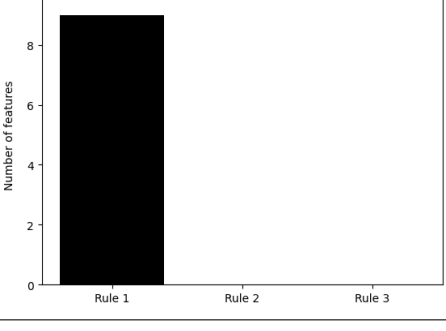
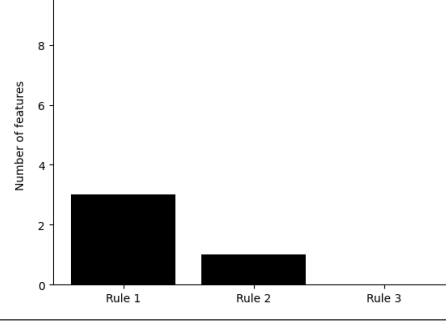
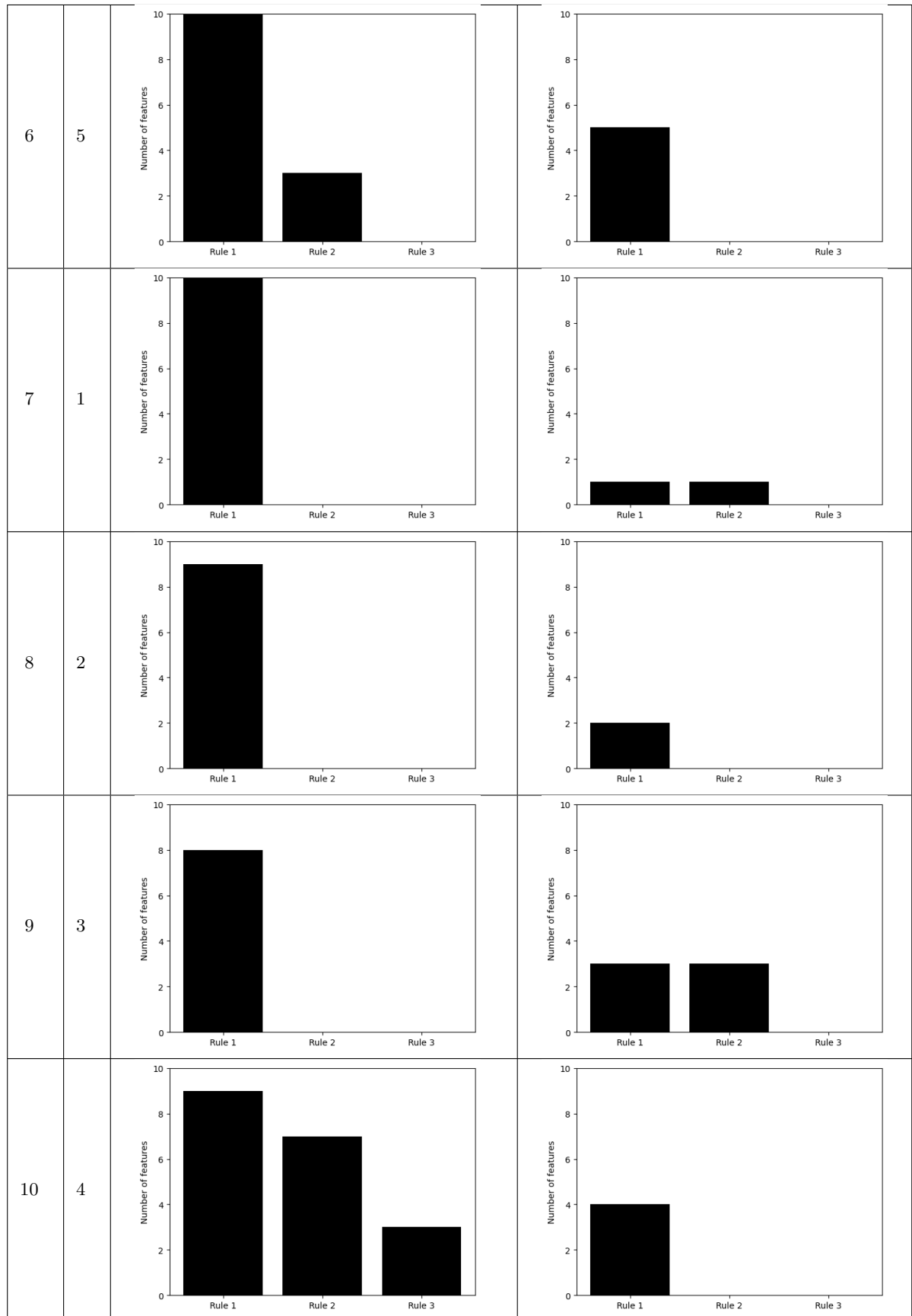


Table 4 – Comparison between I-IMLI and I-IMLIB in terms of balancing the set of rules on the mushroom dataset. The configuration used was  $k = 3$ ,  $\lambda = 10$ , and  $lp = 16$ .

| Runs | $l$ | I-IMLI                                                                              | I-IMLIB                                                                              |
|------|-----|-------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|
| 1    | 5   |    |    |
| 2    | 4   |   |   |
| 3    | 6   |  |  |
| 4    | 5   |  |  |
| 5    | 3   |  |  |



## 6 Conclusion

In this work, we present a new incremental model for learning interpretable and balanced rules: IMLIB. Our method leverages the strengths of SQFSAT, which effectively constrains the size of rules, while incorporating techniques from IMLI, such as incrementability, cost for classification errors, and minimization of the set of rules. Additionally, we introduce an iterative version of the new approach, which we call I-IMLIB. The creation of this version was inspired by the iterative version of IMLI.

Our experiments demonstrate that the proposed approaches generate smaller and more balanced rules than IMLI and its iterative version, while maintaining comparable or even superior accuracy in many cases. We argue that large sets such that all rules are small seem more interpretable when compared to sets with a few large rules. The experiments also showed that, although I-IMLIB improved training performance compared to IMLIB, the iterative version of IMLI outperformed our approaches in all tested datasets. What was expected, as the model generates fewer variables than IMLIB, I-IMLIB, and is iterative.

In future work, we intend to define a measure to determine the balance of a set of rules. Common metrics such as Entropy, Gini Index, Standard Deviation are not sufficient. This is due to certain situations where these metrics would quantify a set of rules as equally balanced. Furthermore, we aim to apply all the techniques described in this work to other SAT-based approaches that can classify more than two classes (IGNATIEV et al., 2018; YU et al., 2020). Thus, developing new incremental and iterative models based on MaxSAT.

# Bibliography

BIERE, A. et al. *Handbook of Satisfiability*. [S.l.]: IOS Press, 2021. v. 336. (Frontiers in Artificial Intelligence and Applications, v. 336). Citado na página 23.

BIRAN, O.; COTTON, C. Explanation and justification in machine learning: A survey. In: *IJCAI-17 workshop on explainable AI (XAI)*. [S.l.: s.n.], 2017. v. 8, p. 8–13. Citado na página 17.

CARLEO, G. et al. Machine learning and the physical sciences. *Reviews of Modern Physics*, APS, v. 91, n. 4, p. 045002, 2019. Citado na página 17.

DUA, D.; GRAFF, C. *UCI Machine Learning Repository*. 2017. Disponível em: <<http://archive.ics.uci.edu/ml>>. Citado na página 37.

GHASSEMI, M.; OAKDEN-RAYNER, L.; BEAM, A. L. The false hope of current approaches to explainable artificial intelligence in health care. *The Lancet Digital Health*, Elsevier, v. 3, n. 11, p. e745–e750, 2021. Citado na página 17.

GHOSH, B.; MALIOUTOV, D.; MEEL, K. S. Efficient learning of interpretable classification rules. *Journal of Artificial Intelligence Research*, v. 74, p. 1823–1863, 2022. Citado 2 vezes nas páginas 18 and 28.

GHOSH, B.; MEEL, K. S. IMLI: An incremental framework for MaxSAT-based learning of interpretable classification rules. In: *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*. [S.l.: s.n.], 2019. p. 203–210. Citado 3 vezes nas páginas 17, 18, and 26.

GUNNING, D. et al. XAI—Explainable artificial intelligence. *Science robotics*, American Association for the Advancement of Science, v. 4, n. 37, p. eaay7120, 2019. Citado na página 17.

HUANG, H.-Y. et al. Power of data in quantum machine learning. *Nature communications*, Nature Publishing Group UK London, v. 12, n. 1, p. 2631, 2021. Citado na página 17.

IGNATIEV, A. et al. Reasoning-based learning of interpretable ML models. In: *IJCAI*. [S.l.: s.n.], 2021. p. 4458–4465. Citado na página 17.

IGNATIEV, A.; MORGADO, A.; MARQUES-SILVA, J. RC2: an efficient MaxSAT solver. *Journal on Satisfiability, Boolean Modeling and Computation*, IOS Press, v. 11, n. 1, p. 53–64, 2019. Citado na página 37.

IGNATIEV, A. et al. A SAT-based approach to learn explainable decision sets. In: SPRINGER. *Automated Reasoning: 9th International Joint Conference, IJCAR 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14-17, 2018, Proceedings 9*. [S.l.], 2018. p. 627–645. Citado 2 vezes nas páginas 17 and 45.

JANIESCH, C.; ZSCHECH, P.; HEINRICH, K. Machine learning and deep learning. *Electronic Markets*, Springer, v. 31, n. 3, p. 685–695, 2021. Citado na página 17.

JIMÉNEZ-LUNA, J.; GRISONI, F.; SCHNEIDER, G. Drug discovery with explainable artificial intelligence. *Nature Machine Intelligence*, Nature Publishing Group UK London, v. 2, n. 10, p. 573–584, 2020. Citado na página 17.

KWEKHA-RASHID, A. S.; ABDULJABBAR, H. N.; ALHAYANI, B. Coronavirus disease (covid-19) cases analysis using machine-learning applications. *Applied Nanoscience*, Springer, p. 1–13, 2021. Citado na página 17.

LAKKARAJU, H.; BACH, S. H.; LESKOVEC, J. Interpretable decision sets: A joint framework for description and prediction. In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. [S.l.: s.n.], 2016. p. 1675–1684. Citado na página 17.

MALIOUTOV, D.; MEEL, K. S. MLIC: A MaxSAT-based framework for learning interpretable classification rules. In: SPRINGER. *Principles and Practice of Constraint Programming: 24th International Conference, CP 2018, Lille, France, August 27-31, 2018, Proceedings*. [S.l.], 2018. p. 312–327. Citado na página 17.

MITA, G. et al. LIBRE: Learning interpretable boolean rule ensembles. In: PMLR. *AISTATS*. [S.l.], 2020. p. 245–255. Citado na página 17.

ROCHA, T. A.; MARTINS, A. T. Synthesis of quantifier-free first-order sentences from noisy samples of strings. In: IEEE. *2019 8th Brazilian Conference on Intelligent Systems (BRACIS)*. [S.l.], 2019. p. 12–17. Citado 2 vezes nas páginas 17 and 25.

ROCHA, T. A.; MARTINS, A. T.; FERREIRA, F. M. Synthesis of a DNF formula from a sample of strings using Ehrenfeucht–Fraïssé games. *Theoretical Computer Science*, Elsevier, v. 805, p. 109–126, 2020. Citado 2 vezes nas páginas 17 and 25.

SHARMA, A. et al. Machine learning applications for precision agriculture: A comprehensive review. *IEEE Access*, IEEE, v. 9, p. 4843–4873, 2020. Citado na página 17.

TJOA, E.; GUAN, C. A survey on explainable artificial intelligence (XAI): Toward medical XAI. *IEEE transactions on neural networks and learning systems*, IEEE, v. 32, n. 11, p. 4793–4813, 2020. Citado na página 17.

VILONE, G.; LONGO, L. Notions of explainability and evaluation approaches for explainable artificial intelligence. *Information Fusion*, Elsevier, v. 76, p. 89–106, 2021. Citado na página 17.

YAN, L. et al. An interpretable mortality prediction model for covid-19 patients. *Nature machine intelligence*, Nature Publishing Group UK London, v. 2, n. 5, p. 283–288, 2020. Citado na página 17.

YU, J. et al. Computing optimal decision sets with SAT. In: SPRINGER. *Principles and Practice of Constraint Programming: 26th International Conference, CP 2020, Louvain-la-Neuve, Belgium, September 7–11, 2020, Proceedings 26*. [S.l.], 2020. p. 952–970. Citado 2 vezes nas páginas 17 and 45.