Incorporando Aprendizado Incremental com MaxSAT em um modelo de Aprendizado de Regras baseado em SAT

Antônio Carlos Souza F. Júnior * **Thiago Alves Rocha** *

* *Instituto Federal de Educação, Ciência e Tecnologia do Ceará, Brasil, antonio.carlos.souza60@aluno.ifce.edu.br, thiago.alves@ifce.edu.br*

Abstract: This article aims to describe a new incremental model for learning interpretable rules based on MaxSAT, called IKKRR. This new model was based on two other approaches, one based on SAT and the other on MaxSAT. The one based on MaxSAT, called IMLI, presents a technique to increase performance that consists in learning a set of rules by applying the model in a dataset incrementally. This work shows which adaptations were necessary for a model based on SAT to be transformed into one based on MaxSAT. It also shows what was done to make it incremental. Finally, IKKRR and IMLI are compared using diverse datasets. Despite having a smaller number of variables, the IKKRR obtained results comparable to the IMLI.

Resumo: Este artigo tem como objetivo descrever a criação de uma nova modelagem incremental de aprendizado de regras interpretáveis baseada no MaxSAT, chamada de IKKRR. A criação dessa abordagem teve como base duas outras abordagens, sendo uma delas baseada no SAT e a outra no MaxSAT. A baseada no MaxSAT, denominada de IMLI, apresenta uma técnica para aumento de performance que consiste em aprender um conjunto de regras aplicando o modelo em um conjunto de dados de maneira incremental. Dito isso, este trabalho mostra quais foram as adaptações necessárias para que uma modelagem baseada no SAT fosse transformada em uma baseada no MaxSAT. Além disso, mostra o que foi feito para torná-la incremental. Por fim, o IKKRR e o IMLI são comparados usando uma variedade de bases de dados. Apesar de ter uma quantidade de variáveis menor, o IKKRR obteve resultados equiparáveis ao IMLI.

Keywords: Interpretable Artificial Intelligence; Explainable Artificial Intelligence; Machine Learning; Computational Logic; Satisfiability.

Palavras-chaves: Inteligência Artificial Interpretável; Inteligência Artificial Explicável; Aprendizado de Máquina; Lógica Computacional; Satisfatibilidade.

1. INTRODUÇÃO

Recentemente ocorreram inúmeros avanços no desenvolvimento de técnicas de aprendizado de máquina (ML do inglês: *Machine Learning*) (Jordan and Mitchell, 2015; LeCun et al., 2015; Mnih et al., 2015). A área de ML trouxe uma infinidade de possibilidades para o desenvolvimento de aplicações em diversos ramos como indústria, governo, ciência e medicina (Chen et al., 2018; Bravo et al., 2021). Alcançando, assim, aplicações que envolvem danos graves a equipamentos, ao meio ambiente, a pessoas e até mesmo a vidas humanas. Por este motivo, além dos algoritmos de ML precisarem ter previsões precisas, é exigido que os mesmos forneçam explicações sobre essas previsões. Várias pesquisas (Che et al., 2016; Liu et al., 2018; Ignatiev et al., 2018; de La Torre et al., 2020) estão sendo feitas no intuito de elaborar abordagens que façam com que os algoritmos, além de prever, expliquem suas previsões para os usuários. O objetivo disso é aumentar a confiança em relação as previsões.

Deixar uma explicação tão clara ao ponto da preocupação do usuário ser apenas avaliar, ao invés de tentar entender a previsão, é crucial. Porém, fazer com que um algoritmo obtenha precisão e interpretabilidade nas suas previsões, muitas vezes, não é uma tarefa simples de se fazer. Vários trabalhos (Lakkaraju et al., 2016; Ignatiev et al., 2018; Malioutov and Meel, 2018; Ghosh and Meel, 2019; Mita et al., 2020) estão sendo desenvolvidos no intuito de solucionar o problema em questão que é o balanceamento da precisão das previsões dos algoritmos com a sua interpretabilidade.

Kamath et al. (1992) propuseram um modelo baseado no problema de satisfatibilidade Booleana (SAT). O objetivo do modelo, que chamaremos de KKRR (das iniciais dos autores Kamath, Karmarkar, Ramakrishna e Resend), é o aprendizado de regras ótimas utilizando variáveis proposicionais. O problema da regra ser ótima é o fato da interpretabilidade ser afetada em determinadas situações. Por outro lado, outros autores propuseram um modelo baseado no problema de máxima satisfatibilidade Booleana (MaxSAT) (Ghosh and Meel, 2019). O objetivo do modelo, chamado IMLI (*Incremental Framework for MaxSAT-based Learning of Interpretable Classification Rules*), é o equilíbrio entre a interpretabilidade e a acurácia por meio da aplicação de pesos. Além disso, o IMLI usa uma abordagem incremental para alcançar uma melhor performance de tempo de execução. A forma incremental

consiste em dividir o conjunto de dados em partições de forma a aprender um conjunto de regras para cada partição a partir das regras obtidas nas partições anteriores.

Dito isso, o objetivo desse trabalho é a criação de uma modelagem de aprendizado de regras que atenda, ao máximo, as exigências citadas anteriormente: previsões precisas, explicações para as mesmas e interpretabilidade. Para isso, utilizaremos duas abordagens: KKRR e IMLI. Implementaremos algumas técnicas do IMLI no KKRR, fazendo com que surja uma nova abordagem que denominamos de IKKRR (versão incremental do KKRR). E por fim, compararemos as abordagens incrementais, IKKRR e IMLI, em diversos conjuntos de dados. A modelagem do IKKRR possui uma quantidade de variáveis menor que a do IMLI, e isso pode produzir modelagens com melhor desempenho no tempo de execução. Entretanto, nos experimentos realizados, os dois métodos não apresentaram diferenças significativas de desempenho.

Este trabalho é dividido em 5 seções. Na Seção 2 são definidas as notações e como o conjunto de dados é tratado antes da aplicação das abordagens. Na Seção 3 são demonstradas as abordagens KKRR e IMLI, respectivamente. Na Seção 4 é apresentada a nossa contribuição: IKKRR. Na Seção 5 é descrito como os testes para a comparação foram feitos e os resultados das comparações. E na última seção apresentamos a conclusão do trabalho e indicamos trabalhos futuros.

2. PRELIMINARES

Consideramos o problema de classificação binária onde recebemos um conjunto de dados. O conjunto de dados é representado por uma matriz binária de tamanho $n \times m$ e um vetor binário de tamanho n. A matriz é representada por $\mathbf{X}$ e o vetor por $\mathbf{y}$. Cada linha de $\mathbf{X}$ é um exemplo do conjunto de dados e é representada por $\mathbf{X}_i$ com $i \in \{1, ..., n\}$. Cada coluna de $\mathbf{X}$ tem um rótulo que representa uma característica e o rótulo é simbolizado por x^j com $j \in \{1, ..., m\}$. Uma partição de $\mathbf{X}$ é representada por $\mathbf{X}^t$ com $t \in \{1, ..., p\}$, onde p é o número de partições. Logo, as partições do vetor $\mathbf{y}$ são representadas por $\mathbf{y}^t$.

Cada elemento de $\mathbf{X}_i$ descreve uma característica do exemplo i. Assim, representamos os valores das m características de $\mathbf{X}_i$ da seguinte maneira: x_i^j com $j \in \{1, ..., m\}$. O complemento do valor de uma característica x_i^j é representado por $\neg x_i^j$. Já o vetor $\mathbf{y}$ representa a classe de cada exemplo. Cada índice de $\mathbf{y}$, ou seja, cada previsão dos n exemplos é representada por: y_i com $i \in \{1, ..., n\}$. Usamos $\mathcal{E}^- = \{X_i \mid y_i = 0, 1 \leq i \leq n\}$, $\mathcal{E}^+ = \{X_i \mid y_i = 1, 1 \leq i \leq n\}$ e para representar o tamanho desses conjuntos, ou seja, o número de exemplos contido neles: $|\mathcal{E}^-|$ e $|\mathcal{E}^+|$.

Exemplo 1. Seja o conjunto de dados $\mathbf{X} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix}$ e $\mathbf{y} = [1, 0, 0, 1]$. Os exemplos $\mathbf{X}_i$ são: $\mathbf{X}_1 = [0, 0, 1], ..., \mathbf{X}_4 = [1, 0, 0]$. As características x_i^j de cada exemplo são: $x_1^1 = 0, x_1^2 = 0, x_1^3 = 1, x_2^1 = 0, ..., x_4^3 = 0$. As previsões y_i são: $y_1 = 1, ..., y_4 = 1$. Podemos dividir $\mathbf{X}$ em duas partições de várias formas, uma delas é: $\mathbf{X}^1 = \begin{bmatrix} 0 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$, $\mathbf{y}^1 = [0, 0]$, $\mathbf{X}^2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ e $\mathbf{y}^2 = [1, 1]$.

Um conjunto de regras é representado por $\mathbf{R}$. Dessa forma, $\mathbf{R}(\mathbf{X}_i)$ representa a aplicação de $\mathbf{R}$ em $\mathbf{X}_i$. Cada regra de $\mathbf{R}$ é representada por R_l com $l \in \{1, ..., k\}$, onde k é o número de regras. A aplicação de R_l em $\mathbf{X}_i$ é representada por $R_l(\mathbf{X}_i)$. Uma regra é uma conjunção de uma ou mais características ou a negação de características. Para representar o número de características de um conjunto de regras $\mathbf{R}$, usamos a notação $|\mathbf{R}|$.

Exemplo 2. Sejam as características: $x^1 =$ É homem, $x^2 =$ Tem moto, $x^3 =$ Não tem carro. Seja o conjunto de regras $\mathbf{R} =$ (É homem) $\vee$ (Não tem carro $\wedge$ Tem moto). As regras R_l são: $R_1 =$ (É homem) e $R_2 =$ (Não tem carro $\wedge$ Tem moto). A aplicação de $\mathbf{R}$ em $\mathbf{X}_i$ é representada da seguinte maneira: $\mathbf{R}(\mathbf{X}_i) = x_i^1 \vee (x_i^3 \wedge x_i^2)$. Por exemplo, seja $\mathbf{X}$ do Exemplo 1, então: $\mathbf{R}(\mathbf{X}_1) = x_1^1 \vee (x_1^3 \wedge x_1^2) = 0 \vee (1 \wedge 0) = 0$. Seguindo a mesma ideia temos: $R_1(\mathbf{X}_1) = x_1^1 = 0$ e $R_2(\mathbf{X}_1) = (x_1^3 \wedge x_1^2) = (1 \wedge 0) = 0$.

Como assumimos um conjunto de dados binários, a discretização de dados usada neste trabalho é a mesma utilizada pela abordagem IMLI. Consiste em binarizar uma tabela de dados com características numéricas ou categóricas. O algoritmo divide as características em quatro tipos: constante, onde todos os exemplos possuem essa mesma característica; binária, onde só existem duas variações distintas dentre todos os exemplos para essa mesma característica; categórica, quando a característica não se enquadra em constante e binária e possui o objetivo de classificar os exemplos com três ou mais classificações; ordinal, quando a característica não se enquadra em constante e binária e tem valorações numéricas.

Quando o tipo da característica é constante, o algoritmo descarta essa característica. Isso acontece pelo fato de que uma característica comum a todos os exemplos não faz diferença nas regras geradas. Quando o tipo é binário, uma das variações da característica receberá 0 e a outra 1 como novos valores. Caso o tipo seja categórico, usamos *one-hot encoding*. Por fim, para característica do tipo ordinal, é feito uma quantização, ou seja, as variações dessa característica são divididas em quantis. Com isso, são atribuídos valores Booleanos para cada quantil de acordo com o valor original.

Usamos SAT e MaxSAT *solver* para implementação das modelagens. Um *solver* recebe uma fórmula na forma normal conjuntiva (FNC), por exemplo: $(p \vee q) \wedge (q \vee \neg p)$. Além disso, um MaxSAT *solver* recebe pesos que serão atribuídos em cada cláusula da fórmula. Os pesos são representados por $W(c) = w$ onde c é uma cláusula e w é um número que representa o peso atribuído. Sendo assim, para que a fórmula fique na FNC e atenda as exigências de entrada do *solver*, muitas modelagens exigem o uso de equisatisfatibilidade. Usamos como notação para equisatisfatibilidade o símbolo $\approx$. Assim, $f_1 \approx f_2$, onde f_1 e f_2 são fórmulas Booleanas, significa que f_2 é satisfatível se e somente se f_1 for satisfatível.

3. APRENDIZAGEM DE REGRAS COM SAT E MAXSAT

3.1 KKRR

O método KKRR é uma abordagem de aprendizado de regras exatas baseada no SAT. Isso quer dizer que, dado um conjunto de dados $\mathbf{X}$, $\mathbf{y}$ e um número de regras k, a abordagem tenta achar um conjunto de regras $\mathbf{R}$ com k regras que classifique, de maneira correta, todos os exemplos $\mathbf{X}_i$, ou seja, $\mathbf{R}(\mathbf{X}_i) = y_i$ para todo i. No geral, a abordagem recebe três parâmetros: $\mathbf{X}$, $\mathbf{y}$ e k. Constrói uma consulta SAT e aplica em um SAT *solver* de modo que a resposta do *solver* possa ser utilizada para obter $\mathbf{R}$.

A construção das cláusulas SAT na modelagem é definida por variáveis proposicionais: $p_{j,l}$ e $p'_{j,l}$. A característica x^j não estará na regra R_l de $\mathbf{R}$ se e somente se $p_{j,l} = 1$. A característica $\neg x^j$ não estará na regra R_l de $\mathbf{R}$ se e somente se $p'_{j,l} = 1$. Para cada exemplo $\mathbf{X}_i \in \mathcal{E}^+$, é acrescentada uma variável $cr_{l,i}$. Assim, $cr_{l,i} = 1$ se e somente se $R_l(\mathbf{X}_i) = 1$ com $\mathbf{X}_i \in \mathcal{E}^+$. Para representar a inserção de $p_{j,l}$ quando $x_i^j = 0$ ou a inserção de $p'_{j,l}$ quando $x_i^j = 1$, usamos a notação $s_{l,i}^j$. Dito isso, a seguir, mostraremos as restrições criadas pela modelagem para a construção da consulta SAT:

Conjunto de cláusulas que garante que não haverá características opostas em cada regra:

$$V_l^j = (p_{j,l} \vee p'_{j,l});\ para\ j \in \{1, ..., m\}\ e\ l \in \{1, ..., k\} \quad (1)$$

Agora vamos para o conjunto de cláusulas que garante que $\mathbf{R}(\mathbf{X}_i) = 0$ para todo i com $y_i = 0$. Seja $P_i^+ = \{j \mid x_i^j = 1, 1 \leq j \leq m\}$ e $P_i^- = \{j \mid x_i^j = 0, 1 \leq j \leq m\}$:

$$\begin{aligned} D_{l,i} = (\bigvee_{j \in P_i^+} \neg p'_{j,l} \vee \bigvee_{j \in P_i^-} \neg p_{j,l});\\ para\ l \in \{1, ..., k\}\ e\ \mathbf{X}_i \in \mathcal{E}^- \end{aligned} \quad (2)$$

A seguir, temos o conjunto de cláusulas que garante que $\mathbf{R}(\mathbf{X}_i) = 1$ para todo i com $y_i = 1$:

$$\begin{aligned} S_{l,i}^j =& (s_{l,i}^j \vee \neg cr_{l,i});\ para\ j \in \{1, ..., m\},\\ & l \in \{1, ..., k\}\ e\ \mathbf{X}_i \in \mathcal{E}^+ \end{aligned} \quad (3)$$

$$C_i = (\bigvee_{l \in \{1,...,k\}} cr_{l,i});\ para\ \mathbf{X}_i \in \mathcal{E}^+ \quad (4)$$

E finalmente temos a consulta que será enviada para o *solver*. Seja Q todas as cláusulas SAT construídas pela modelagem, temos:

$$\begin{aligned} Q = \bigwedge_{\substack{l \in \{1,...,k\} \\ j \in \{1,...,m\}}} V_l^j \ \wedge \bigwedge_{\substack{l \in \{1,...,k\} \\ i \in \mathcal{E}^-}} D_{l,i} \ \wedge \\ \bigwedge_{\substack{j \in \{1,...,m\} \\ l \in \{1,...,k\} \\ i \in \mathcal{E}^+}} S_{l,i}^j \ \wedge \bigwedge_{i \in \mathcal{E}^+} C_i \end{aligned} \quad (5)$$

3.2 IMLI

O método IMLI é uma abordagem incremental baseada no MaxSAT para aprendizagem de regras interpretáveis. Um dos seus maiores focos é o equilíbrio entre a acurácia e a interpretabilidade das regras. A métrica utilizada para a definição de interpretabilidade é quantidade de literais em todas as regras. Regras com poucos literais são mais simples de serem entendidas e avaliadas por humanos.

Uma possibilidade para obter máxima acurácia seria buscar um conjunto de regras $\mathbf{R}$ que classifica todos os exemplos corretamente, ou seja, $\mathbf{R}(\mathbf{X}_i) = y_i$ para todo i. A ideia é que, de todas as possibilidades que satisfaçam isso, ela retorne a menor delas: $\min_R\{|\mathbf{R}| \mid \mathbf{R}(\mathbf{X}_i) = y_i$ para todo $i\}$.

Obter o menor conjunto de regras $\mathbf{R}$ que classifique todos os exemplos de $\mathbf{X}$ corretamente pode gerar regras grandes e, consequentemente, difíceis de interpretar. Dito isso, a modelagem do IMLI permite erros de classificação com uma constante λ de penalização. O intuito é equilibrar a interpretabilidade com a acurácia. Sendo assim, quanto maior o valor de λ, mais acurácia o conjunto de regras vai ter, porém sua interpretabilidade será afetada negativamente. Quanto menor for o valor de λ, menor será a acurácia, mas a interpretabilidade será afetada positivamente. Matematicamente o objetivo é:

$$\min_R\{|\mathbf{R}| + \lambda|\mathcal{E}_R| \mid \mathcal{E}_R = \{\mathbf{X}_i \mid \mathbf{R}(\mathbf{X}_i) \neq y_i\}\} \quad (6)$$

No geral, o IMLI recebe quatro parâmetros: $\mathbf{X}$, $\mathbf{y}$, k e λ. Sendo $\mathbf{X}$ uma matriz binária contendo colunas barradas para cada característica. Constrói uma consulta MaxSAT e aplica ela em um MaxSAT *solver* de modo que a resposta do *solver* possa ser utilizada para obter $\mathbf{R}$. A construção das cláusulas MaxSAT na modelagem é definida a partir de $k \times m$ variáveis proposicionais: $b_1^1, b_1^2, ..., b_1^m, ..., b_k^m$. A característica x_i^j estará na regra R_l de $\mathbf{R}$ se e somente se $b_l^j = 1$. Para cada exemplo $\mathbf{X}_i$, é introduzido uma variável η_i que representa se $\mathbf{R}$ não classifica corretamente o exemplo, ou seja, se $\mathbf{R}(\mathbf{X}_i) \neq y_i$. Essas variáveis serão utilizadas nas cláusulas *soft* da modelagem. As cláusulas *soft* são cláusulas que podem não serem satisfeitas pela solução, caso necessário. Mostraremo-as a seguir:

$$N_i = (\neg\eta_i);\ W(N_i) = \lambda;\ para\ i \in \{1, ..., n\} \quad (7)$$

$$\begin{aligned} V_l^j = (\neg b_l^j);\ W(V_l^j) = 1;\ para\ j \in \{1, ..., m\}\ e\\ l \in \{1, ..., k\} \end{aligned} \quad (8)$$

Observe que os dois conjuntos de cláusulas são inseridos com o símbolo de negação. Isso se deve pelo fato de que a resposta ideal esperada é uma regra vazia, pois seria a mais fácil possível de ser entendida, se seguirmos a métrica de interpretabilidade comentada anteriormente. Além disso, também é desejável que o máximo de exemplos sejam acertados, por isso as variáveis η_i também são negadas. E o equilíbrio entre os tamanhos das regras e o número de acertos é controlado pelos seus respectivos pesos $W(N_i)$ e $W(V_l^j)$. O objetivo do MaxSAT *solver* é valorar as variáveis de uma forma que o máximo de cláusulas sejam satisfeitas.

O terceiro conjunto de cláusulas é um conjunto de cláusulas *hard*, que são cláusulas que devem obrigatoriamente serem satisfeitas pela solução, e consiste em: se $\eta_i = 0$, ou seja, se o exemplo $\mathbf{X}_i$ deve ser corretamente classificado, então $y_i = \mathbf{R}(\mathbf{X}_i)$ tem que ser verdade. Para forçar que essa igualdade aconteça, sabendo que y_i é uma constante enviada por parâmetro, é feita uma operação, representada pelo operador @, entre o exemplo $\mathbf{X}_i$ e o vetor

$\mathbf{B}_l = \{b_l^j \mid j \in \{1,...,m\}\}$ com $l \in \{1,...,k\}$. Semelhante ao produto escalar de vetores, essa operação consiste em efetuar uma conjunção de cada índice x_i^j com o índice b_l^j correspondente. Com o vetor resultante dessa operação é feito uma disjunção entre cada elemento contido nele. Para facilitar o entendimento vamos mostrar dois exemplos:

Exemplo 3. Seja $\mathbf{X}_i = [0,1,0]$ e $\mathbf{B}_l = [b_l^1, b_l^2, b_l^3]$. Temos que:

$\mathbf{X}_i \,@\, \mathbf{B}_l = b_l^2$, pois $(x_i^1 \wedge b_l^1) \vee (x_i^2 \wedge b_l^2) \vee (x_i^3 \wedge b_l^3) \equiv (0 \wedge b_l^1) \vee (1 \wedge b_l^2) \vee (0 \wedge b_l^3) \equiv b_l^2$

Exemplo 4. Seja $\mathbf{X}_i = [1,0,0]$, $\mathbf{B}_1 = [b_1^1, b_1^2, b_1^3]$ e $\mathbf{B}_2 = [b_2^1, b_2^2, b_2^3]$. Temos que:

$(\mathbf{X}_i \,@\, \mathbf{B}_1) \wedge (\mathbf{X}_i \,@\, \mathbf{B}_2) = b_1^1 \wedge b_2^1$, pois $(x_i^1 \wedge b_1^1) \vee (x_i^2 \wedge b_1^2) \vee (x_i^3 \wedge b_1^3) \wedge (x_i^1 \wedge b_2^1) \vee (x_i^2 \wedge b_2^2) \vee (x_i^3 \wedge b_2^3) \equiv (1 \wedge b_1^1) \vee (0 \wedge b_1^2) \vee (0 \wedge b_1^3) \wedge (1 \wedge b_2^1) \vee (0 \wedge b_2^2) \vee (0 \wedge b_2^3) \equiv b_1^1 \wedge b_2^1$

Note que apenas as características que servirão para que a regra classifique o exemplo como positivo são postas na cláusula resultante. Matematicamente o terceiro conjunto de cláusulas MaxSAT fica da seguinte forma:

$$\begin{aligned} &D_i = (\neg\eta_i \rightarrow (y_i \leftrightarrow \bigwedge_{l \in \{1,...,k\}} (\mathbf{X}_i \,@\, \mathbf{B}_l))); \\ &W(D_i) = \infty;\ \textit{para } i \in \{1,...,n\} \end{aligned} \tag{9}$$

Seja Q todas as cláusulas MaxSAT construídas pela modelagem, temos:

$$\begin{aligned} Q = \bigwedge_{i \,\in\, \{1,...,n\}} N_i \ \wedge \bigwedge_{\substack{l \,\in\, \{1,...,k\} \\ j \,\in\, \{1,...,m\}}} V_l^j \ \wedge \\ \bigwedge_{i \,\in\, \{1,...,n\}} D_i \end{aligned} \tag{10}$$

Após Q ser convertida para FNC, temos a consulta e os pesos que serão enviados para o *solver*. O número de variáveis dessa consulta é: $|\mathcal{E}^-| + |\mathcal{E}^+| + m \cdot k + |\mathcal{E}^-| \cdot k$. O número de cláusulas, no pior caso, é: $|\mathcal{E}^-| + |\mathcal{E}^+| + m \cdot k + |\mathcal{E}^+| + |\mathcal{E}^-| \cdot (m \cdot k + 1)$.

A seguir, no Exemplo 5, vamos mostrar como a modelagem monta toda a consulta dado os respectivos parâmetros requeridos. Vale salientar que, ao contrário do KKRR, e embora o exemplo a seguir também não mostre por motivos didáticos, o IMLI utiliza o conjunto de dados $\mathbf{X}$ contendo as colunas barradas de cada característica:

Exemplo 5. Seja $\mathbf{X} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$, $\mathbf{y} = [1,0,0]$, $k = 2$ e $\lambda = 10$:

$N_1 = (\neg\eta_1);\ N_2 = (\neg\eta_2);\ N_3 = (\neg\eta_3);\ W(N_i) = 10;$

$V_1^1 = (\neg b_1^1);\ V_1^2 = (\neg b_1^2);\ V_1^3 = (\neg b_1^3);$

$V_2^1 = (\neg b_2^1);\ V_2^2 = (\neg b_2^2);\ V_2^3 = (\neg b_2^3);\ W(V_l^j) = 1;$

$D_1 = (\neg\eta_1 \rightarrow (y_1 \leftrightarrow ((b_1^1 \vee b_1^2 \vee b_1^3) \wedge (b_2^1 \vee b_2^2 \vee b_2^3))));$

$D_2 = (\neg\eta_2 \rightarrow (y_2 \leftrightarrow ((b_1^1 \vee b_1^3) \wedge (b_2^1 \vee b_2^3))));$

$D_3 = (\neg\eta_3 \rightarrow (y_3 \leftrightarrow (b_1^1 \wedge b_2^1)));$ $\qquad W(D_i) = \infty;$

Para que o *solver* aceite essa consulta, é preciso que algumas adaptações sejam feitas, pois um *solver* recebe consulta na FNC. Observe que D_1, D_2, D_3 não atendem a esse formato. Para isso, iremos aplicar equivalência lógica e também, em D_2 e D_3, equisatisfatibilidade ($\approx$):

$D_1 = (\neg\eta_1 \rightarrow (1 \leftrightarrow ((b_1^1 \vee b_1^2 \vee b_1^3) \wedge (b_2^1 \vee b_2^2 \vee b_2^3)))) \equiv$
$(\neg\eta_1 \rightarrow ((b_1^1 \vee b_1^2 \vee b_1^3) \wedge (b_2^1 \vee b_2^2 \vee b_2^3))) \equiv$
$(\eta_1 \vee ((b_1^1 \vee b_1^2 \vee b_1^3) \wedge (b_2^1 \vee b_2^2 \vee b_2^3))) \equiv$
$(\eta_1 \vee ((\neg b_1^1 \vee \neg b_1^2 \vee \neg b_1^3) \vee (\neg b_2^1 \vee \neg b_2^2 \vee \neg b_2^3))) \equiv$
$(\eta_1 \vee \neg b_1^1 \vee \neg b_1^2 \vee \neg b_1^3 \vee \neg b_2^1 \vee \neg b_2^2 \vee \neg b_2^3);$

$D_2 = (\neg\eta_2 \rightarrow (0 \leftrightarrow ((b_1^1 \vee b_1^3) \wedge (b_2^1 \vee b_2^3)))) \equiv$
$(\neg\eta_2 \rightarrow ((\neg b_1^1 \wedge \neg b_1^3) \vee (\neg b_2^1 \wedge \neg b_2^3))) \equiv$
$(\eta_2 \vee ((\neg b_1^1 \wedge \neg b_1^3) \vee (\neg b_2^1 \wedge \neg b_2^3))) \equiv$
$(\eta_2 \vee (\neg b_1^1 \wedge \neg b_1^3) \vee (\neg b_2^1 \wedge \neg b_2^3)) \approx$
$(\eta_2 \vee z_1 \vee z_2) \wedge (z_1 \rightarrow (\neg b_1^1 \wedge \neg b_1^3)) \wedge (z_2 \rightarrow (\neg b_2^1 \wedge \neg b_2^3)) \equiv$
$(\eta_2 \vee z_1 \vee z_2) \wedge (\neg z_1 \vee (\neg b_1^1 \wedge \neg b_1^3)) \wedge (\neg z_2 \vee (\neg b_2^1 \wedge \neg b_2^3)) \equiv$
$(\eta_2 \vee z_1 \vee z_2) \wedge (\neg z_1 \vee \neg b_1^1) \wedge (\neg z_1 \vee \neg b_1^3) \wedge (\neg z_2 \vee \neg b_2^1) \wedge (\neg z_2 \vee \neg b_2^3);$
$W(\eta_2 \vee z_1 \vee z_2) = W(\neg z_1 \vee \neg b_1^1) = W(\neg z_1 \vee \neg b_1^3) = W(\neg z_2 \vee \neg b_2^1) =$
$W(\neg z_2 \vee \neg b_2^3) = \infty;$

$D_3 = (\neg\eta_3 \rightarrow (0 \leftrightarrow (b_1^1 \wedge b_2^1))) \equiv$
$(\neg\eta_3 \rightarrow (\neg b_1^1 \vee \neg b_2^1)) \equiv$
$(\eta_3 \vee (\neg b_1^1 \vee \neg b_2^1)) \equiv$
$(\eta_3 \vee \neg b_1^1 \vee \neg b_2^1) \approx$
$(\eta_3 \vee z_3 \vee z_4) \wedge (z_3 \rightarrow \neg b_1^1) \wedge (z_4 \rightarrow \neg b_2^1) \equiv$
$(\eta_3 \vee z_3 \vee z_4) \wedge (\neg z_3 \vee \neg b_1^1) \wedge (\neg z_4 \vee \neg b_2^1);$
$W(\eta_3 \vee z_3 \vee z_4) = W(\neg z_3 \vee \neg b_1^1) = W(\neg z_4 \vee \neg b_2^1) = \infty;$

E ao final de tudo temos que:

$Q_{FNC} = \neg\eta_1 \wedge \neg\eta_2 \wedge \neg\eta_3 \wedge \neg b_1^1 \wedge \neg b_1^2 \wedge \neg b_1^3 \wedge \neg b_2^1 \wedge \neg b_2^2 \wedge \neg b_2^3 \wedge (\eta_1 \vee \neg b_1^1 \vee \neg b_1^2 \vee \neg b_1^3 \vee \neg b_2^1 \vee \neg b_2^2 \vee \neg b_2^3) \wedge (\eta_2 \vee z_1 \vee z_2) \wedge (\neg z_1 \vee \neg b_1^1) \wedge (\neg z_1 \vee \neg b_1^3) \wedge (\neg z_2 \vee \neg b_2^1) \wedge (\neg z_2 \vee \neg b_2^3) \wedge (\eta_3 \vee z_3 \vee z_4) \wedge (\neg z_3 \vee \neg b_1^1) \wedge (\neg z_4 \vee \neg b_2^1)$

A matriz de entrada $\mathbf{X}$, no IMLI, pode ser dividida em p partições: $\mathbf{X}^1$, $\mathbf{X}^2$, ..., $\mathbf{X}^p$. Com isso, o vetor $\mathbf{y}$ particionado em: $\mathbf{y}^1$, $\mathbf{y}^2$, ..., $\mathbf{y}^p$, de acordo com as partições de $\mathbf{X}$. Cada partição contém a mesma quantidade, ou a quantidade mais próxima possível, de exemplos $\mathbf{X}_i \in \mathcal{E}^-$ e $\mathbf{X}_i \in \mathcal{E}^+$. Além disso, os exemplos são distribuídos de maneira aleatória entre as partições. O particionamento é uma das diferenças entre o IMLI e sua versão anterior MLIC (*MaxSAT-Based framework for learning interpretable classification rules*) (Malioutov and Meel, 2018).

As consultas são criadas em cima das partições, uma de cada vez. Na primeira partição, a modelagem constrói a consulta da mesma forma descrita no Exemplo 5. Na segunda em diante, as valorações obtidas pelo *solver* na partição anterior para as restrições V_l^j são reaproveitadas. A motivação disso é tentar manter a regra obtida na partição anterior para a próxima partição. Sendo assim, a partir da segunda partição, as restrições V_l^j, para $j \in \{1,...,m\}$ e $l \in \{1,...,k\}$, Restrição (8), são substituídas pelas restrições a seguir:

$$\begin{aligned} &V_l^j = \begin{cases} b_l^j, \text{ se } b_l^j = 1 \text{ na solução da partição anterior;} \\ \neg b_l^j, \text{ caso contrário;} \end{cases} \\ &W(V_l^j) = 1; \\ &\textit{para } j \in \{1,...,m\} \textit{ e } l \in \{1,...,k\} \end{aligned} \tag{11}$$

A modelagem do IMLI apresentada obtém regras na FNC. Apesar disso, o IMLI consegue gerar regras na FNC e na

forma normal disjuntiva (FND). Para que o modelo gere as regras na FND é preciso que o parâmetro de entrada **y** seja invertido e que o conjunto de regras obtido seja negado.

Outro procedimento realizado pelo IMLI é uma técnica para diminuir o tamanho das regras geradas. Criada no intuito de remover redundâncias, como a do Exemplo 6 a seguir.

Exemplo 6. Seja o seguinte conjunto de regras abaixo:

$$(\text{Idade} > 18 \wedge \text{Idade} > 20) \vee (\text{Altura} \leq 2)$$

Veja que esse conjunto de regras acima pode ser simplificado removendo a redundância da primeira regra:

$$(\text{Idade} > 20) \vee (\text{Altura} \leq 2)$$

A técnica para remover redundâncias é aplicada logo após o *solver* encontrar as valorações obtidas pelas consultas em cada partição. O funcionamento da técnica consiste em averiguar se alguma característica, que seja do tipo ordinal, aparece duas ou mais vezes. Caso apareça, ele detecta se o operador relacional das características repetidas é $\leq$ ou $>$. Caso seja $\leq$, a característica com o menor valor se mantém, enquanto as demais são excluídas. E se for $>$, a característica com o maior valor se mantém, enquanto as demais são excluídas. O Exemplo 7 demonstra o funcionamento dessa técnica, dado uma regra com redundância.

Exemplo 7. Seja o conjunto de regras $\mathbf{R}$ = (Sexo = Masculino $\wedge$ Idade $\leq 15 \wedge$ Idade $\leq 18) \vee$ (Salário $> 1400 \wedge$ Salário $> 1700 \wedge$ Salário ≤ 2000). Os seguintes passos são realizados:

1. Identificar as características do tipo ordinal que se repetem em cada regra de $\mathbf{R}$ e que possuem o mesmo operador relacional. Será guardado em vetor(es) r_l, onde $l \in \{1, ..., k\}$, as respectivas repetições:

$r_1 = [\text{Idade} \leq 15, \text{Idade} \leq 18]$; $r_2 = [\text{Salário} > 1400, \text{Salário} > 1700]$

2. As características em r_l, caso haja, são consideradas redundantes. Sendo assim, neste passo, deve ser escolhida apenas uma delas. Se o operador for $\leq$, a que tiver o menor valor. Se $>$, a que tiver o maior:

$r_1 = [\text{Idade} \leq 15]$; $r_2 = [\text{Salário} > 1700]$

3. Insere a(s) característica(s) restante(s) de r_l na respectiva regra R_l de $\mathbf{R}$:

$\mathbf{R}$ = (Sexo = Masculino $\wedge$ Idade $\leq 15) \vee$ (Salário $> 1700 \wedge$ Salário ≤ 2000)

4. ABORDAGEM INCREMENTAL PARA O MÉTODO KKRR

Nesta seção vamos apresentar nosso método IKKRR que é uma versão incremental e baseada no MaxSAT da modelagem KKRR. Além disso, o IKKRR usa a técnica de remoção de redundâncias do IMLI. Portanto, a modelagem passou a ter particionamento dos dados de entrada, acréscimo de restrições, peso em suas restrições e aplicação de técnica para redução de redundâncias nas regras geradas. Com isso, a nossa abordagem recebe quatro parâmetros de entrada: $\mathbf{X}$, $\mathbf{y}$, k e λ.

A primeira mudança foi em relação a como os dados de entrada, ou matriz de entrada $\mathbf{X}$, são tratados antes de serem aplicados na modelagem. O mesmo processo de particionamento usado no IMLI é aplicado nos dados de entrada. Dessa forma, as consultas serão criadas em cima de cada partição. Antes de demonstrarmos como ficarão estas consultas em um exemplo, mostraremos como ficam as restrições da abordagem, sendo a primeira delas uma das acrescidas na modelagem:

Conjunto de cláusulas que tenta minimizar o número de características em $\mathbf{R}$:

$$N_l^j = (p_{j,l});\ W(N_l^j) = 1;\ \textit{para}\ j \in \{1, ..., m\}\ e\ l \in \{1, ..., k\} \tag{12}$$

$$L_l^j = (p'_{j,l});\ W(L_l^j) = 1;\ \textit{para}\ j \in \{1, ..., m\}\ e\ l \in \{1, ..., k\} \tag{13}$$

Conjunto de cláusulas que garante que não haverá características opostas em cada regra:

$$V_l^j = (p_{j,l} \vee p'_{j,l});\ W(N_i) = \infty;\ \textit{para}\ j \in \{1, ..., m\}\ e\ l \in \{1, ..., k\} \tag{14}$$

Conjunto de cláusulas que garante que $\mathbf{R}(\mathbf{X}_i) = 0$, para todo i com $y_i = 0$. Sejam P^+ e P^- definidos como na Restrição (2):

$$D_{l,i} = (\bigvee_{j \in P^+} \neg p'_{j,l} \vee \bigvee_{j \in P^-} \neg p_{j,l});\ W(N_i) = \lambda;\ \textit{para}\ l \in \{1, ..., k\}\ e\ \mathbf{X}_i \in \mathcal{E}^- \tag{15}$$

Conjunto de cláusulas que garante que $\mathbf{R}(\mathbf{X}_i) = 1$ para todo i com $y_i = 1$:

$$S_{l,i}^j = (s_{l,i}^j \vee \neg cr_{l,i});\ W(N_i) = \infty;\ \textit{para}\ j \in \{1, ..., m\},\ l \in \{1, ..., k\}\ e\ \mathbf{X}_i \in \mathcal{E}^+ \tag{16}$$

$$C_i = (\bigvee_{l \in \{1,...,k\}} cr_{l,i});\ W(N_i) = \lambda;\ \textit{para}\ \mathbf{X}_i \in \mathcal{E}^+ \tag{17}$$

Por fim, temos a consulta que será enviada para o *solver*. Seja Q todas as cláusulas MaxSAT construídas pela modelagem, temos:

$$Q = \bigwedge_{\substack{l \in \{1,...,k\} \\ j \in \{1,...,m\}}} (N_l^j \wedge L_l^j) \wedge \bigwedge_{\substack{l \in \{1,...,k\} \\ j \in \{1,...,m\}}} V_l^j \wedge \bigwedge_{\substack{l \in \{1,...,k\} \\ i \in \mathcal{E}^-}} D_{l,i} \wedge \bigwedge_{\substack{j \in \{1,...,m\} \\ l \in \{1,...,k\} \\ i \in \mathcal{E}^+}} S_{l,i}^j \wedge \bigwedge_{i \in \mathcal{E}^+} C_i \tag{18}$$

O número de variáveis de Q é: $2 \cdot m \cdot k + |\mathcal{E}^+| \cdot k$. O número de cláusulas é: $3 \cdot m \cdot k + |\mathcal{E}^-| \cdot k + m \cdot |\mathcal{E}^+| \cdot k + |\mathcal{E}^+|$.

A criação das consultas na primeira partição da matriz de entrada $\mathbf{X}$, seguindo as restrições, é demonstrada no Exemplo 8. A partir da segunda partição, com o intuito de tentar manter a regra obtida na partição anterior, as Restrições (12) e (13) são substituídas pelas restrições a seguir:

$$N_l^j = \begin{cases} \neg p_{j,l}, \text{ se } p_{j,l} = 0 \text{ na solução da partição anterior;} \\ p_{j,l}, \text{ caso contrário;} \end{cases}$$
$$W(N_l^j) = 1$$
$$\textit{para}\ j \in \{1, ..., m\}\ e\ l \in \{1, ..., k\} \tag{19}$$

$$L_l^j = \begin{cases} \neg p'_{j,l}, \text{ se } p'_{j,l} = 0 \text{ na solução da partição anterior;} \\ p'_{j,l}, \text{ caso contrário;} \end{cases}$$
$$W(L_l^j) = 1$$
$$para\ j \in \{1, ..., m\}\ e\ l \in \{1, ..., k\} \tag{20}$$

Exemplo 8. Seja $\mathbf{X} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix}$, $\mathbf{y} = [1, 0, 0]$, $k = 2$ e $\lambda = 10$:

$N_1^1 = (p_{1,1});\ N_1^2 = (p_{2,1});\ N_1^3 = (p_{3,1});$

$N_2^1 = (p_{1,2});\ N_2^2 = (p_{2,2});\ N_2^3 = (p_{3,2});$

$W(N_l^j) = 1;$

$L_1^1 = (\neg p_{1,1});\ L_1^2 = (\neg p_{2,1});\ L_1^3 = (\neg p_{3,1});$

$L_2^1 = (\neg p_{1,2});\ L_2^2 = (\neg p_{2,2});\ L_2^3 = (\neg p_{3,2});$

$W(L_l^j) = 1;$

$V_1^1 = (p_{1,1} \vee p'_{1,1});\ V_1^2 = (p_{2,1} \vee p'_{2,1});\ V_1^3 = (p_{3,1} \vee p'_{3,1});$

$V_2^1 = (p_{1,2} \vee p'_{1,2});\ V_2^2 = (p_{2,2} \vee p'_{2,2});\ V_2^3 = (p_{3,2} \vee p'_{3,2});$

$W(V_l^j) = \infty;$

$D_{1,2} = (\neg p_{1,1} \vee \neg p_{2,1} \vee \neg p'_{3,1});\ D_{1,3} = (\neg p'_{1,1} \vee \neg p_{2,1} \vee \neg p'_{3,1});$

$D_{2,2} = (\neg p_{1,2} \vee \neg p_{2,2} \vee \neg p'_{3,2});\ D_{2,3} = (\neg p'_{1,2} \vee \neg p_{2,2} \vee \neg p'_{3,2});$

$W(D_{l,i}) = \lambda;$

$S_{1,1}^1 = (p'_{1,1} \vee \neg cr_{1,1});\ S_{1,1}^2 = (p'_{2,1} \vee \neg cr_{1,1});\ S_{1,1}^3 = (p'_{3,1} \vee \neg cr_{1,1});$

$S_{2,1}^1 = (p'_{1,2} \vee \neg cr_{2,1});\ S_{2,1}^2 = (p'_{2,2} \vee \neg cr_{2,1});\ S_{2,1}^3 = (p'_{3,2} \vee \neg cr_{2,1});$

$W(S_{l,i}^j) = \infty;$

$C_1 = (cr_{1,1} \vee cr_{2,1});\ W(C_i) = \lambda;$

E ao final de tudo temos que:

$$Q = p_{1,1} \wedge p_{2,1} \wedge p_{3,1} \wedge p_{1,2} \wedge p_{2,2} \wedge p_{3,2} \wedge \neg p_{1,1} \wedge \neg p_{2,1} \wedge \neg p_{3,1} \wedge \neg p_{1,2} \wedge \neg p_{2,2} \wedge \neg p_{3,2} \wedge (p_{1,1} \vee p'_{1,1}) \wedge (p_{2,1} \vee p'_{2,1}) \wedge (p_{3,1} \vee p'_{3,1}) \wedge (p_{1,2} \vee p'_{1,2}) \wedge (p_{2,2} \vee p'_{2,2}) \wedge (p_{3,2} \vee p'_{3,2}) \wedge (\neg p_{1,1} \vee \neg p_{2,1} \vee \neg p'_{3,1}) \wedge (\neg p'_{1,1} \vee \neg p_{2,1} \vee \neg p'_{3,1}) \wedge (\neg p_{1,2} \vee \neg p_{2,2} \vee \neg p'_{3,2}) \wedge (\neg p'_{1,2} \vee \neg p_{2,2} \vee \neg p'_{3,2}) \wedge (p'_{1,1} \vee \neg cr_{1,1}) \wedge (p'_{2,1} \vee \neg cr_{1,1}) \wedge (p'_{3,1} \vee \neg cr_{1,1}) \wedge (p'_{1,2} \vee \neg cr_{2,1}) \wedge (p'_{2,2} \vee \neg cr_{2,1}) \wedge (p'_{3,2} \vee \neg cr_{2,1}) \wedge (cr_{1,1} \vee cr_{2,1})$$

A última mudança do IKKRR com relação ao KKRR é a aplicação da redução do tamanho das regras geradas. Esta técnica é a mesma utilizada no IMLI para redução de redundâncias, que foi explicada no Exemplo 7.

5. EXPERIMENTOS

Nesta seção apresentamos os experimentos que conduzimos para comparar os métodos IMLI e IKKRR. As duas modelagens foram implementadas [1] em Python e com o MaxSAT *solver* MiFuMaX (Janota, 2014). Os experimentos foram feitos em uma máquina com as seguintes configurações: processador Intel(R) Core(TM) i5-4460 de 3.20GHz, e 12GB de memória RAM. Foram utilizadas 8 bases de dados do repositório UCI (Dua and Graff, 2017) para comparar o IKKRR com IMLI em termos de tamanho do conjunto de regras, tamanho da maior regra, acurácia no teste e tempo de treinamento. O tamanho do conjunto de regras e o tamanho da maior regra podem ser utilizadas como métricas de interpretabilidade. Por exemplo, um conjunto de regras com poucas regras e com regras pequenas é mais interpretável que um outro com muitas regras grandes.

Base de dados	E	C	Métricas	IMLI	IKKRR
Parkinsons	195	22	TC	**4.40 ± 2.33**	5.10 ± 2.88
			TMR	**2.30 ± 0.78**	2.80 ± 1.08
			AC	**0.76 ± 0.10**	**0.76 ± 0.08**
			TT	1.11 ± 0.23	**1.01 ± 0.18**
Ionosphere	351	33	TC	**7.60 ± 4.56**	8.10 ± 5.52
			TMR	4.50 ± 1.36	**4.40 ± 1.62**
			AC	**0.83 ± 0.06**	0.82 ± 0.12
			TT	**2.01 ± 0.47**	2.08 ± 0.50
Iris	150	4	TC	6.60 ± 2.46	**6.40 ± 2.83**
			TMR	4.10 ± 1.04	**3.80 ± 0.60**
			AC	**0.85 ± 0.08**	**0.85 ± 0.06**
			TT	**0.91 ± 0.25**	1.04 ± 0.10
Lung cancer	59	6	TC	**3.90 ± 0.54**	4.00 ± 0.63
			TMR	**3.40 ± 0.66**	3.60 ± 0.80
			AC	**0.95 ± 0.08**	0.93 ± 0.08
			TT	**0.26 ± 0.03**	**0.26 ± 0.06**
Mushroom	8124	22	TC	13.40 ± 4.96	**12.40 ± 3.98**
			TMR	9.10 ± 0.83	**8.50 ± 1.36**
			AC	**0.99 ± 0.00**	**0.99 ± 0.00**
			TT	**29.07 ± 6.35**	29.40 ± 9.78
WDBC	569	30	TC	**4.80 ± 3.51**	6.50 ± 3.53
			TMR	**2.90 ± 1.22**	4.00 ± 1.67
			AC	0.83 ± 0.10	**0.85 ± 0.07**
			TT	**3.52 ± 0.92**	3.83 ± 0.91
Toms	28179	96	TC	**6.90 ± 3.93**	7.20 ± 4.35
			TMR	**4.50 ± 2.01**	5.30 ± 2.00
			AC	**0.91 ± 0.02**	0.89 ± 0.04
			TT	**134.74 ± 37.83**	136.31 ± 47.03
Twitter	49999	77	TC	10.20 ± 6.00	**6.70 ± 3.92**
			TMR	5.80 ± 2.40	**4.40 ± 1.43**
			AC	0.90 ± 0.03	**0.91 ± 0.02**
			TT	276.75 ± 86.30	**264.25 ± 82.86**

Tabela 1. Comparação entre as modelagens IKKRR e IMLI em diferentes bases de dados. As colunas E e C representam, para cada base de dados, a quantidade de exemplos e a de características, respectivamente. Nas últimas duas colunas, o primeiro valor representa o tamanho do conjunto de regras (TC), o segundo representa o tamanho da maior regra (TMR), o terceira valor representa a acurácia (AC) e o quarto valor representa o tempo de treinamento (TT) em segundos. Destaca-se em negrito o melhor valor e o empate entre ambas as modelagens.

O experimento foi repetido 10 vezes em cada base de dados. Em cada repetição os dados foram separados em: 80% para treino, 10% para validação e 10% para teste. Os mesmos dados resultantes dessa divisão foram utilizados para ambos os métodos. O conjunto de validação foi usado para escolher a configuração de cada modelagem que obteve maior acurácia. As configurações foram compostas pela combinação de: $k \in \{1, 2, 3\}$, $\lambda \in \{5, 10\}$ e $lp \in \{8, 16\}$, onde k é o número de regras, λ é a constante de penalização e lp é o número máximo de exemplos por partição. Tendo a melhor configuração de cada abordagem, o conjunto de treino e o de validação foram utilizados

[1] O código fonte do IKKRR e a implementação dos testes executados podem ser encontrados no link: https://github.com/cacajr/Abordagens-Para-Aprendizado-de-Regras-de-Classificacao-Interpretavel

Base de dados	IKKRR
Parkinsons	(Maximum vocal fundamental frequency $\leq$ 200.41 and DFA $\leq$ 0.76) or (RPDE $>$ 0.46 and spread2 $>$ 0.20)
Iris	(Petal length $\leq$ 5.32 and Petal length $>$ 3.90 and Petal width $\leq$ 1.90) or (Petal length $\leq$ 4.64 and Petal width $>$ 1.16)
Lung cancer	(Age $>$ 58.40) or (Age $>$ 27.60 and AreaQ $\leq$ 8.00 and Alkhol $>$ 2.0)
WDBC	(largest radius $>$ 16.00 and largest concave points $>$ 0.08) or (radius $>$ 17.06 and smoothness $>$ 0.09)

Tabela 2. Exemplos de conjuntos de regras geradas pelo IKKRR em algumas bases de dados.

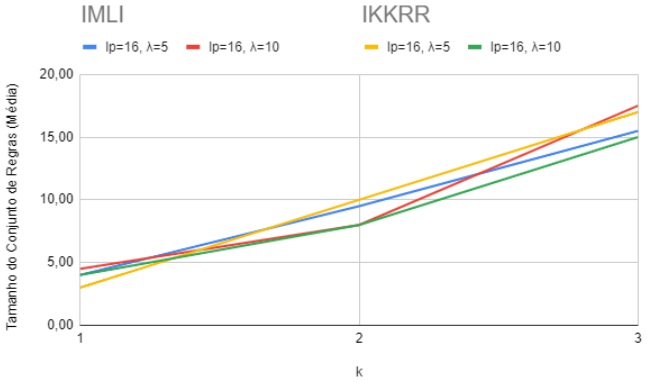

Figura 1. Comportamento do tamanho do conjunto de regras ao variar k com lp e λ fixos na base de dados Toms.

para obter o conjunto de regras com essa configuração. O conjunto de regras obtido foi aplicado no conjunto de teste para obter os resultados de acurácia. A Tabela 1 mostra a média dos resultados das 10 repetições das duas modelagens nas respectivas bases de dados.

Na Tabela 1 podemos observar que, em algumas situações, o IKKRR consegue se destacar. Os exemplos que mostram isso são os da base de dados Mushroom e Twitter. Na base Mushroom, o IKKRR teve não só um conjunto de regras menor como também teve regras menores, obtendo a mesma acurácia ao custo de um pouco mais de tempo de treinamento. Na base Twitter, a maior base dentre as testadas, o IKKRR se destacou em todos os aspectos, incluindo o tempo de treinamento.

O fato do IKKRR gerar uma quantidade de variáveis menor que o IMLI não contribuiu para o aumento significativo de performance. O que não era o esperado tendo em vista que quanto menos variáveis um *solver* recebe, menos combinações de valorações existem para serem testadas. Entretanto, é possível que as restrições do IMLI ajudem o *solver* a encontrar a solução ótima mesmo com as variáveis adicionais.

Alguns exemplos de conjunto de regras geradas pelo IKKRR podem ser vistas na Tabela 2. Esses exemplos indicam que o IKKRR consegue obter conjuntos de regras interpretáveis.

Também usamos o experimento acima para comparamos os dois métodos com as mesmas configurações em cada conjunto de dados. Dessa forma, para cada configuração de k, λ e lp, usamos os mesmos 80% do conjunto de dados para obter as regras com os dois métodos. O objetivo é

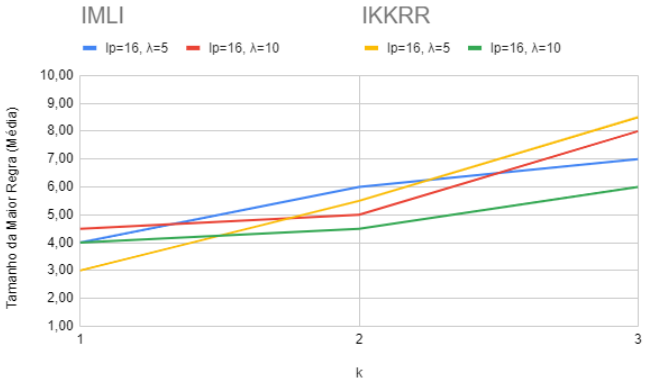

Figura 2. Comportamento do tamanho da maior regra ao variar k com lp e λ fixos na base de dados Toms.

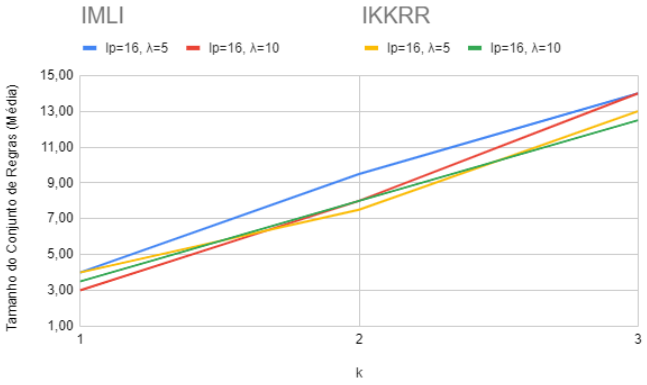

Figura 3. Comportamento do tamanho do conjunto de regras ao variar k com lp e λ fixos na base de dados Ionosphere.

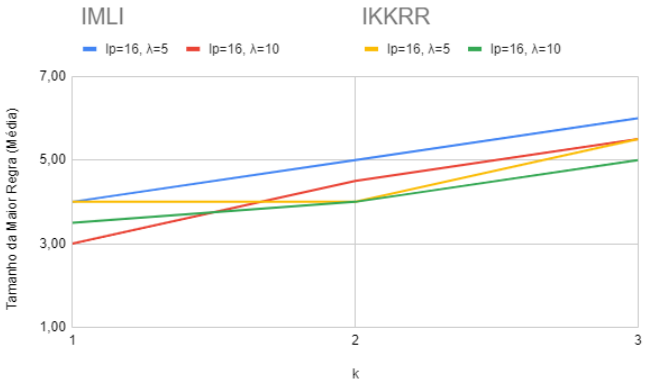

Figura 4. Comportamento do tamanho da maior regra ao variar k com lp e λ fixos na base de dados Ionosphere.

comparar os resultados dos métodos com as mesmas configurações. Analisamos a média do tamanho do conjunto de regras e do tamanho da maior regra de acordo com o aumento do número de regras k. Selecionamos três bases de dados para discutirmos os resultados. Na base de dados Toms, a configuração $lp = 16$ e $\lambda = 10$ do IKKRR teve melhor desempenho em termos do tamanho do conjunto de regras e tamanho da maior regra (Figuras 1 e 2). Nas Figuras 3 e 4 vemos que as duas configurações do IKKRR obtiveram melhor desempenho no tamanho do conjunto

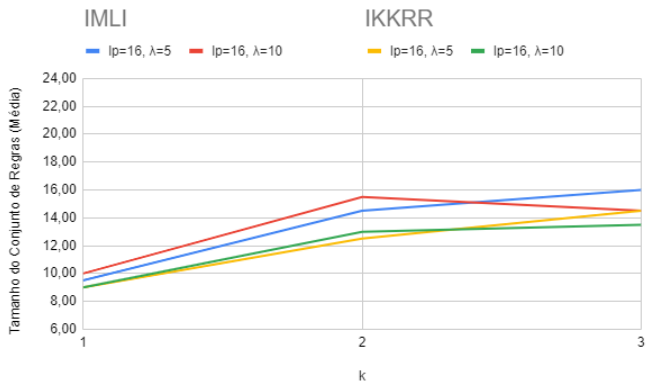

Figura 5. Comportamento do tamanho do conjunto de regras ao variar k com lp e λ fixos na base de dados Mushroom.

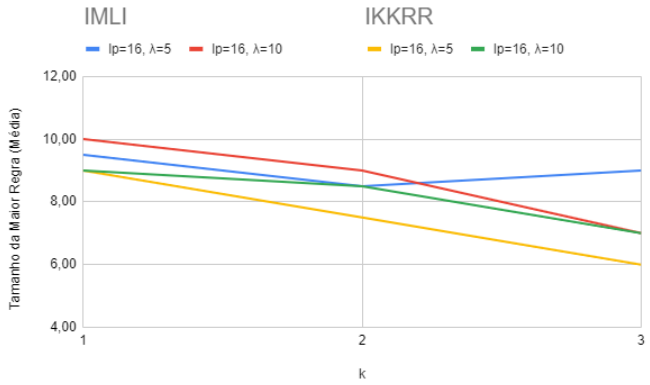

Figura 6. Comportamento do tamanho da maior regra ao variar k com lp e λ fixos na base de dados Mushroom.

de regras e no tamanho da maior regra na base de dados Ionosphere. Também observamos que, nessas duas bases de dados, o tamanho do conjunto de regras e o tamanho da maior regra aumentaram a medida que k aumentou em ambos os métodos e cada configuração. Já na base de dados Mushroom, o tamanho da maior regra diminuiu com o aumento de k nas configurações do IKKRR (Figura 6). Enquanto que esse tamanho aumentou para uma das configurações do IMLI. Na base de dados Mushroom, as configurações do IKKRR também obtiveram melhor desempenho no tamanho do conjunto de regras e no tamanho da maior regra (Figuras 5 e 6).

6. CONCLUSÃO

Neste trabalho, apresentamos o IKKRR, uma nova modelagem incremental de aprendizado de regras baseada no MaxSAT. Com o experimento aplicado utilizando as bases fornecidas pelo UCI, podemos averiguar que os resultados do IKKRR em relação ao IMLI foram similares, como descrito na Seção 5, mas interessantes, pois em algumas situações houveram melhorias. Uma dessas situações foi na maior base de dados testada e em todos os aspectos. Além disso, a criação do IKKRR demonstra a possibilidade de aprimoramento em diversos modelos de aprendizado de regras implementando as técnicas utilizadas no IMLI. Vale destacar que a menor quantidade de variáveis gerada pelo IKKRR não trouxe ganhos consideráveis na performance do modelo em relação ao IMLI. Como trabalho futuro, faremos uma versão incremental do método denominado MinDS (Ignatiev et al., 2018), aplicando as mesmas técnicas do IMLI que foram aplicadas no KKRR. Com isso, faremos a comparação entre: IMLI, IKKRR e a versão incremental do MinDS.

REFERÊNCIAS

Bravo, G.J.R., Maza, J.d.J.E., Issasi, A.M., Álvarez, Á.S., and Rodríguez, L.A.R. (2021). Aplicación de machine learning en la indutria 4.0 en tiempos de pandemia. *Interconectando Saberes*, 6(11).

Che, Z., Purushotham, S., Khemani, R., and Liu, Y. (2016). Interpretable deep models for ICU outcome prediction. In *AMIA Annual Symposium Proceedings*, volume 2016, 371.

Chen, Y., Wang, J., and Cai, Z. (2018). Study on the application of machine learning in government service: Take consumer protection service as an example. In *15th ICSSSM*, 1–5.

de La Torre, J., Valls, A., and Puig, D. (2020). A deep learning interpretable classifier for diabetic retinopathy disease grading. *Neurocomputing*, 396, 465–476.

Dua, D. and Graff, C. (2017). UCI machine learning repository. URL `http://archive.ics.uci.edu/ml`.

Ghosh, B. and Meel, K.S. (2019). IMLI: An incremental framework for MaxSAT-based learning of interpretable classification rules. In *AIES'19*, 203–210.

Ignatiev, A., Pereira, F., Narodytska, N., and Marques-Silva, J. (2018). A SAT-based approach to learn explainable decision sets. In *IJCAR*, 627–645. Springer.

Janota, M. (2014). MiFuMax—a literate MaxSAT solver. *Journal on Satisfiability, Boolean Modeling and Computation*, 9(1), 83–88.

Jordan, M.I. and Mitchell, T.M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245), 255–260.

Kamath, A.P., Karmarkar, N.K., Ramakrishnan, K., and Resende, M.G. (1992). A continuous approach to inductive inference. *Mathematical programming*, 57(1-3), 215–238.

Lakkaraju, H., Bach, S.H., and Leskovec, J. (2016). 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*, 1675–1684.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 436–444.

Liu, S., Yao, J., Zhou, C., and Motani, M. (2018). SURI: Feature selection based on unique relevant information for health data. In *BIBM*, 687–692.

Malioutov, D. and Meel, K.S. (2018). MLIC: A MaxSAT-based framework for learning interpretable classification rules. In *CP*, 312–327. Springer.

Mita, G., Papotti, P., Filippone, M., and Michiardi, P. (2020). LIBRE: Learning interpretable boolean rule ensembles. In *AISTATS*, 245–255. PMLR.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540), 529–533.