Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Departamento de Computação

Programa de Pós-Graduação em Ciência da Computação

Extração de forma compacta de Regras Fuzzy de uma Rede Bayesiana

Yin I Hsien

São Carlos - SP

Maio/2010

Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Departamento de Computação

Programa de Pós-Graduação em Ciência da Computação

Extração de regras fuzzy para explicação de forma compacta de uma rede Bayesiana

Yin I Hsien

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação, do Departamento de Computação, da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.
Área de concentração: Inteligência Artificial-IA

São Carlos – SP

Maio/2010

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar

Y51ef

Yin, I Hsien.
Extração de forma compacta de regras fuzzy de uma rede Bayesiana / I Hsien Yin. -- São Carlos : UFSCar, 2010.
156 f.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2010.

1. Data mining (Mineração de dados) - métodos estatísticos. 2. Redes Bayesianas. 3. Lógica nebulosa. 4. Seleção de atributos. 5. Inteligência artificial. I. Título.

CDD: 006.3 (20ª)

Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Programa de Pós-Graduação em Ciência da Computação

"Extração de forma compacta de Regras Fuzzy de uma Rede Bayesiana"

YIN I HSIEN

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação

Membros da Banca:

Prof. Dr. Estevam Rafael Hruschka Júnior
(Orientador - DC/UFSCar)

Profa. Dra. Heloisa de Arruda Camargo
(DC/UFSCar)

Profa. Dra. Maria do Carmo Nicoletti
(PPG-CC/UFSCar)

Prof. Dr. Nelson Francisco Favilla Ebecken
(COPPE/UFRJ)

São Carlos
julho/2010

AGRADECIMENTOS

Agradeço a oportunidade que esta universidade e o Departamento de Computação me concedeu para realizar este trabalho de pesquisa no programa de Pós-Graduação em Ciência da Computação.

Ao meu orientador Prof. Dr. Estevam Rafael Hruschka Junior por todo seu apoio e confiança, assim como à minha co-orientadora Profa. Dra. Heloísa de Arruda Camargo. Ambos possuem vasto conhecimento e bom caráter. Obrigado por compartilhar todos os seus conhecimentos, assim como também as suas formas de convivência. Agradeço também aos outros professores do departamento pelos conselhos e apoios recebidos.

À minha família, que mesmo estando distante, está sempre me apoiando e me dando dicas, trocando experiências e dando-me forças para continuar no caminho que escolhi e à minha namorada, que em todos os momentos esteve comigo me dando todo o tipo de apoio e ajudando-me a superar diversas dificuldades.

Aos meus amigos e colegas que sempre me ajudaram e apoiaram, compartilhando diversos conhecimentos e discutindo ideias que contribuíram para este trabalho.

Aos funcionários não somente do Departamento de Computação, mas de toda a universidade pelos seus esforços para manter o ambiente de pesquisa agradável.

À CAPES, pelo apoio financeiro no primeiro ano do programa e à empresa E-Biz Solution também pelo apoio financeiro e equipamentos de pesquisa assim como a oportunidade de poder trabalhar junto num projeto tão interessante.

Dedico este trabalho ao meu falecido pai,
pela sabedoria que me deu e que até hoje é de grande inspiração.

RESUMO

As ferramentas de apoio à decisão são importantes em diversos domínios da nossa sociedade. Porém há uma necessidade do usuário entender as decisões feitas por tais ferramentas para ter uma confiança maior sobre os resultados. Na literatura técnica, as redes Bayesianas são consideradas como um sistema probabilístico de classificação com bom desempenho em termos de precisão. Mas ainda necessitam de uma forma de apresentação mais compreensível para os usuários. Por outro lado, a lógica *fuzzy* oferece potencial para lidar com imprecisão e incerteza, assim como a representação linguística, o que facilita a compreensão dos usuários. A combinação das redes Bayesianas com a lógica *fuzzy* é proposta pelo método *BayesFuzzy* que utiliza regras *fuzzy* como explicação de uma rede Bayesiana, com o objetivo de obter uma ferramenta de apoio à decisão de bom desempenho e que seja fácil de ser compreendida pelos usuários. O *BayesFuzzy*, entretanto, apresenta limitações com relação ao número de regras geradas e isto torna seus resultados, muitas vezes, de difícil interpretação. Assim, neste trabalho de mestrado é proposto o método *Pruned BayesFuzzy* (PBF). O PBF tem como base o *BayesFuzzy* e incorpora algumas técnicas de minimização do número de regras para otimizar a compreensibilidade dos resultados gerados. Dentre as técnicas incorporadas estão o mínimo grau de certeza, a regra default e a poda *Rule Post-Pruning* como formas de selecionar dentre as regras geradas, as mais importantes para a classificação e ao mesmo tempo simplificando estas regras. Os resultados do PBF mostram que houve um ganho grande em relação à compreensibilidade, mas também uma perda na taxa de classificação correta. Porém o ganho de compreensibilidade é bastante promissor o que estimula a pesquisa e a seqüência dos trabalhos com o PBF. Além do PBF, este trabalho propõe também o *Pruned BayesFuzzy* 2 (PBF2) que é o PBF incorporando uma técnica de seleção de atributos baseado em *Markov Blanket*. Com a incorporação desta técnica, é possível lidar com situações que contém uma quantidade grande de variáveis dentro do *Markov Blanket* da variável classe. Os resultados mostram que houve perda na taxa de classificação correta, o que é de se esperar quando tentamos simplificar mais ainda o *Markov Blanket*. A viabilidade de poder resolver problemas reais de grande escala e com algumas características específicas é ainda algo a ser considerado.

ABSTRACT

The decision support tools are importants in many domains of our society. But there is a need to make users understand the decisions made by those tools in order to increase the faith of the users on the results. In the literature, Bayesian networks are considered as a probabilistic classification system with good performance. But it still need a better presentation of it results to make it more understandable to the users. In other hand, the fuzzy logic ofer potential to deal with imprecision and uncertainty, as well as a linguistic representation, which facilitates user's understanding. The combination of Bayesian networks and fuzzy logic is proposed by the method BayesFuzzy, which make use of fuzzy rules as a form of explanation of a Bayesian network, it aims to obtain a decision support tool with good performance and easy to be understood by users. So we are proposiing the method Pruned BayesFuzzy (PBF), it is a BayesFuzzy incorporated with minimum certainty degree, default rule and Rule Post-Pruning as a form to select the most important rules for classification between all the rules generated, it also simplifies those rules. The results of PBF show an improvement in understanding but a loss in correct classification rate. But the improvement in understanding is promising enough to further research and enhance of the PBF. Then beside PBF, we also propose the Pruned BayesFuzzy 2 (PBF2), which is PBF incorporated with a feature selection technique based on Markov Blanket. With the incorporation of this technique, it's possible to deal with situations that contains a large amount of variables inside of the Markov Blanket of the class variable. The results show a loss in correct classification rate, that is already expected when we try to simplify further more the Markov Blanket. However, the availability to be able to deal with big scale problems is something to be considered.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ALGORÍTMOS

1. INTRODUÇÃO

O Aprendizado de Máquina (AM) pode ser visto como uma área de pesquisa que busca o desenvolvimento de programas de computador que possam evoluir à medida que vão sendo expostos à novas experiências. Estas experiências podem ser vistas como dados armazenados em meios eletrônicos e que podem ser utilizados como fonte de informação para o aprendizado automático de programas de computador.

Dentre as formas de aprendizado de máquinas mais comuns, está o aprendizado supervisionado. Neste tipo de aprendizado, um conjunto de dados rotulados é fornecido para o treinamento de um classificador. No processo de treinamento do classificador, o conjunto de dados de treinamento serve como experiências que os algoritmos de aprendizado se baseiam para construir seus modelos de conhecimento.

Quando uma aplicação de AM é utilizada como ferramenta de apoio à decisão, é muito importante que além de fornecer resultados precisos, permitam que os usuários possam entender as razões para cada resultado gerado [3]. Assim, a precisão e a compreensibilidade dos resultados da classificação devem ser consideradas características importantes de uma ferramenta de apoio à decisão.

Em [8] os autores dividem os tipos de explicação em explicação diagnóstica e explicação preditiva. A explicação diagnóstica explica o problema voltado para uma entrada especificamente. Podemos assimilar como um médico diagnosticando os sintomas de um certo paciente. Isso representa o contrário do que estamos tentando realizar neste trabalho, que é a explicação preditiva, ou seja, explicar a lógica de funcionamento do problema e não em relação a uma certa entrada especificamente. Podemos entender como um professor ensinando os futuros médicos a como diagnosticar os pacientes.

Como pode ser observado na literatura, métodos Bayesianos têm sido utilizados para modelar problemas que envolvem a tarefa de classificação com bastante sucesso, principalmente quando se considera a precisão dos resultados obtidos. O conhecimento representado por classificadores Bayesianos não é, entretanto, tão compreensível como algumas outras formas de representação, tais como as regras de classificação. Tentando superar esta dificuldade, alguns trabalhos vêm sendo desenvolvidos buscando a melhoria na compreensibilidade dos resultados de classificadores Bayesianos [3][5][8].

Também com base na literatura técnica, pode-se constatar que a utilização de variáveis linguísticas, como as utilizadas em Classificadores *fuzzy*, facilitam a interpretação e aumentam a compreensibilidade dos resultados gerados por um classificador [20]. Assim, a integração de redes Bayesianas e classificadores *fuzzy* podem trazer ganhos em termos de compreensibilidade e precisão.

Portanto, este trabalho de pesquisa tem como objetivo, investigar, propor e implementar formas de explicação de um classificador Bayesiano utilizando regras de classificação *fuzzy*. Para atingir esse objetivo, o processo de extração de regras de redes Bayesianas será o foco principal da investigação. Neste sentido, busca-se identificar formas adequadas de se traduzir o conhecimento armazenado em uma rede Bayesiana para a forma de regras de classificação *fuzzy*.

Este trabalho está organizado da seguinte forma: o Capítulo 2 apresenta alguns conceitos básicos sobre as Redes Bayesianas; o Capítulo 3 aborda alguns conceitos básicos da teoria fuzzy utilizada neste trabalho; o Capítulo 4 aborda alguns trabalhos que se relacionam com o objetivo desta pesquisa; o Capítulo 5 descreve o método *Pruned BayesFuzzy*, que é um dos métodos propostos deste trabalho; o Capítulo 6 apresenta o método *Pruned BayesFuzzy*2, que é uma melhora do método *Pruned BayesFuzzy*; e finalmente o Capítulo 7 que apresenta as conclusões e aponta alguns trabalhos futuros.

2. REDES BAYESIANAS

Este capítulo, apresenta algumas noções básicas de redes Bayesianas para o melhor entendimento da pesquisa. O capítulo está organizado da seguinte forma: Inicia-se com uma explicação sucinta sobre o teorema de Bayes, que é a base das redes Bayesianas; Seção 2.1 Inferência - nesta seção, há a explicação de uma das formas clássicas de se realizar inferências numa rede Bayesiana; Seção 2.2 Aprendizado de redes Bayesianas - apresenta dois algoritmos clássicos que realizam aprendizado automático das redes Bayesianas; Seção 2.3 Naïve Bayes - apresenta uma introdução básica sobre os classificadores *Naïve Bayes* e Seção 2.4 Seleção de atributos - *Markov Blanket*, onde levanta a importância da realização do processo de seleção de atributos e uma introdução sobre o *Markov Blanket*, que é uma das formas clássicas de se realizar a seleção de atributos numa rede Bayesiana.

As Redes Bayesiana são a base para o método de extração de regras *fuzzy* proposto neste trabalho e, desta forma, utilizaremos um exemplo para explicar um pouco sobre estes modelos probabilísticos.

Considere um problema do domínio médico, onde queremos saber se um paciente está com gripe ou não. Pelos nossos conhecimentos sobre a gripe, sabemos que sintomas como cansaço e nariz congestionado são eventos que ocorrem, na maioria das vezes, quando estamos com gripe. Este conhecimento que temos, de antemão, é chamado de conhecimento *a priori*. Com este conhecimento *a priori*, podemos dizer que, quando um paciente apresenta sintomas como cansaço e nariz congestionado, existe uma grande probabilidade de que ele esteja com gripe. Ou seja, neste caso, os eventos de cansaço e nariz congestionado influenciam no aumento da probabilidade do paciente estar com gripe. Esta influência é nomeada de Dependência Condicional. A utilização da noção de dependência condicional para saber a probabilidade de um paciente estar com gripe ou não é chamada de Probabilidade Condicional. O conhecimento obtido a partir destas relações (dependências condicionais) é chamado de conhecimento *a posteriori*.

O teorema de *Bayes*, através da fórmula (1), pode ser utilizado para calcular a probabilidade condicional de um evento acontecer, dado que um outro evento aconteceu. Em nosso exemplo de tratamento médico, suponha que queremos descobrir qual a probabilidade do paciente estar ou não com gripe, dado que o paciente manifesta sintomas de cansaço e nariz congestionado.

$$P(X|Y) = \frac{P(Y|X)\,P(X)}{P(Y)} \qquad (1)$$

P(X|Y) – Probabilidade condicional do evento X acontecer dado que Y aconteceu

P(Y|X) – Probabilidade condicional do evento Y acontecer dado que X aconteceu,

é chamada também de verossimilhança.

P(X) – Probabilidade do evento X acontecer

P(Y) – Probabilidade do evento Y acontecer

Vamos estender um pouco o nosso exemplo para entendermos o teorema de *Bayes*. Considere que estamos resolvendo o problema de uma clínica especializada em tratamento de gripe. Assim, o nosso problema é descobrir se os pacientes que vêm para a clínica estão com gripe ou não. Suponha que a clínica já possui um histórico de 100 registros sobre pacientes que foram diagnosticados e tratados. Este histórico será utilizado para definir o nosso conhecimento *a priori*. Estes registros armazenam apenas duas informações: se o paciente está com gripe ou não e se o paciente está com o nariz congestionado. Suponha então que dentre esses 100 registros, 80 referem-se a pacientes que apresentam o nariz congestionado, 75 estão com gripe e 60 apresentam o nariz congestionado e gripe ao mesmo tempo. Então, quando um paciente chega à clínica, desejamos descobrir se ele está com gripe ou não, sendo que o paciente nos informou que está com o nariz congestionado. Utilizaremos o teorema *Bayes* para determinar qual a probabilidade do paciente estar com gripe. Na fórmula (1), o evento de paciente estar com gripe ou não, corresponde ao evento X. O evento de paciente estar com o nariz congestionado ou não corresponde ao evento Y. A probabilidade de um paciente estar com gripe corresponde à P(X) = 80/100 = 0,8 (probabilidade *a priori* de X). A probabilidade de um paciente estar com o nariz congestionado corresponde à P(Y) = 75/100 = 0,75 (probabilidade *a priori* de Y). A probabilidade de um paciente estar com o nariz congestionado dado que estava com gripe, ou seja, P(Y|X) é 60/75 = 0,8. A resposta que queremos descobrir corresponde a P(X|Y). Aplicando o teorema Bayes, teremos:

P(X|Y) = 0,8 * 0,8 / 0,75 = 0,853.

Ou seja, existe 85,3% (probabilidade a posteriori) de chance do paciente estar com gripe, dado que está com o nariz congestionado.

Este exemplo é apenas uma simplificação do problema de classificação, pois o problema real possui um grau de complexidade bem maior. Da mesma forma, muitos outros problemas reais não podem ser resolvidos simplesmente aplicando uma única vez o teorema *Bayes*, porque diversas variáveis estão envolvidas no problema e podem existir influências ou dependências condicionais entre as variáveis. Para esses casos, as redes Bayesianas podem auxiliar na definição do modelo probabilístico que seja mais adequado para o problema em questão.

As redes Bayesianas são modelos que representam as relações de dependências e independências condicionais entre as variáveis. Eles são

representados por um grafo direcionado acíclico, onde cada nodo representa uma variável do problema, cada arco representa uma dependência condicional entre dois nodos e, se entre dois nodos não existe nenhum arco, estes são considerados condicionalmente independentes entre si. Um arco direcionado que conecta da variável X à variável Y representa que X é o pai ou antecessor de Y, e Y é o filho ou descendente de X. Além disso, a cada nodo está associada uma tabela de probabilidade condicional que contém todas as possíveis combinações de valores entre o pai e o filho e as suas respectivas probabilidades condicionais. Veja um exemplo, na **Figura 1**. Neste exemplo, temos quatro variáveis associadas A, B, C e D sendo que o A é o pai dos nodos B e C. B é o pai de D assim como C também é pai de D. Para cada nodo, a tabela que se situa mais próxima a ele é a tabela de probabilidade condicional daquele nodo. A forma de se interpretar esta tabela de probabilidade condicional é, por exemplo, se olharmos para a tabela de probabilidade condicional do nodo B, o P(B = 0|A = 0) = 0,7, quer dizer que, dado que o nodo A assume o valor 0, a probabilidade do nodo B assumir o valor 0 é de 0,7 (70%). A mesma lógica se aplica às outras tabelas.

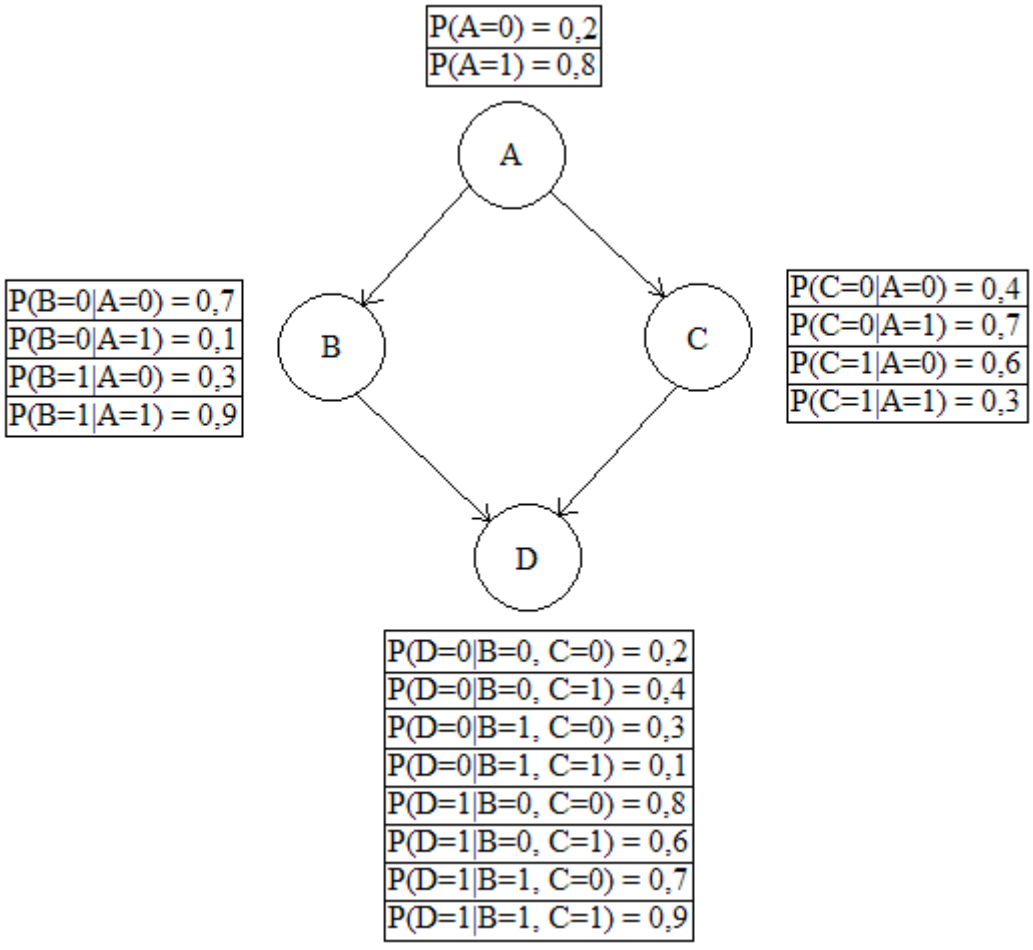

Figura 1 Exemplo de uma rede Bayesiana com as suas tabelas de probabilidade condicional.

2.1. Inferência

A inferência Bayesiana entre duas variáveis pode ser feita aplicando o teorema de *Bayes* (1), como vimos no exemplo dado no início do Capítulo 2. Porém, quando temos uma rede Bayesiana complexa, a simples aplicação do teorema de *Bayes* não poderá resolver o problema. Por exemplo, considera-se a rede Bayesiana apresentada na Figura 2 em que todas as variáveis são binárias e que foi providenciada uma evidência, valor 0, para a variável A. e então, para calcular, por exemplo, qual a probilidade de C assumir o valor 0, dados que a variável A é 0. Assim, não é possível aplicar uma única vez o teorema de Bayes. Porém, podemos calcular o P(C=0|A=0) da seguinte maneira [28]:

$$P(C=0|A=0) = P(C=0|B=0)P(B=0|A=0) + P(C=0|B=1)P(B=1|A=0)$$

Neste caso, estamos calculando a probabilidade da variável C assumir valor 0 dado que a variável A é 0 independentemente do valor que o B assume.

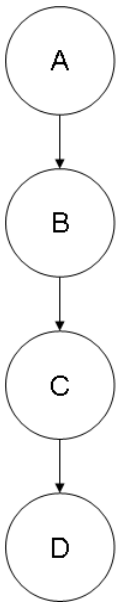

Figura 2 Exemplo de uma rede bayesiana na forma de uma lista conectada.

E se desejamos calcular a probabilidade condicional da variável D dado um valor da variável A, iremos repetir o processo e chegaremos ao resultado.

Entretanto, o processo apresentado acima só é válido para redes Bayesianas na forma de uma lista conectada ou uma árvore. Para uma rede Bayesiana como a da Figura 3, o mesmo processo já não consegue lidar.

Para esses casos, em [30] foi desenvolvido um algorítmo de passagem de

mensagem para realizar a inferência nas redes Bayesianas simples conectadas. Este algorítmo basicamente constrói dois tipos de mensagens: as mensagens λ, que são mandadas dos filhos para os pais e as mensagens π, que são mandadas dos pais para os filhos. Estas mensagens atualizam as probabilidades condicionais de cada variável de acordo com as evidências que a rede recebe. Neste algoritmo, ainda considera que cada possível valor de cada variável possui dois valores associados, chamados de valor λ e valor π, ambos utilizados no cálculo das mensagens λ e π.

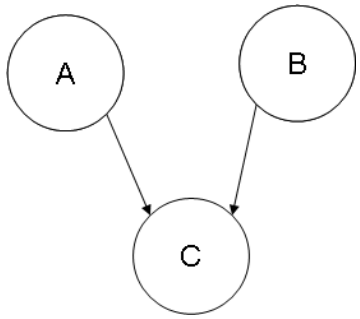

Figura 3 Exemplo de uma rede Bayesiana.

Tabela 1 Tabela de probabilidade condicional da rede Bayesiana da Figura 3.

Variável	Conhecimento a priori
A	P(A=0) = 0,3
	P(A=1) = 0,7
B	P(B=0) = 0,8
	P(B=1) = 0,2
C	P(C=0) = 0,6
	P(C=1) = 0,4
	P(C=0|A=0, B=0) = 0,4
	P(C=1|A=0. B=0) = 0,6
	P(C=0|A=0, B=1) = 0,5
	P(C=1|A=0. B=1) = 0,5
	P(C=0|A=1, B=0) = 0,2
	P(C=1|A=1. B=0) = 0,8
	P(C=0|A=1, B=1) = 0,9
	P(C=1|A=1. B=1) = 0,1

Para entender melhor, considere a rede Bayesiana da Figura 3. O algoritmo inicia com um passo de inicialização, onde o valor λ de todos os possíveis valores de todas as variáveis assumem o valor 1 e todas as mensagens λ e π de todos os possíveis valores de todas as variáveis também assumem o valor 1. Em nosso exemplo, o conjunto das variáveis instanciadas serão representadas pela A_e, que

conterá o nodo A e C, e o conjunto das evidências serão representadas pela a_e, onde o nodo A será instanciado com o valor 0 e o nodo C com o valor 1.

Em seguida, serão identificadas as variáveis raízes, ou seja, variáveis que não possuem nenhum pai, no nosso exemplo são as variáveis A e B, e definirá os valores π de cada possível valor destas variáveis raízes como sendo a probabilidade a priori daqueles valores. Então, no nosso exemplo:

$\lambda(A_0) = 1$
$\lambda(A_1) = 1$

$\pi(A_0) = 0,3$
$\pi(A_1) = 0,7$

$\lambda(B_0) = 1$
$\lambda(B_1) = 1$

$\pi(B_0) = 0,8$
$\pi(B_1) = 0,2$

No passo seguinte, as variáveis raizes enviarão as mensagens π (cada possível valor do nodo pai envia uma mensagem π independente para o nodo filho), que são calculadas usando a equação (2), para que os seus filhos calculem os valores π de cada possível valor de acordo com a equação (3):

$$\pi_X(z) = \pi(z) \; Y \in \text{CHz} - \{X\} \; \lambda_Y(z) \qquad (2)$$

Legenda da equação (2):

$\pi_X(z)$ = Mensagem π que um certo valor z do nodo Z manda para o nodo X;
$\pi(z)$ = O valor π de um certo valor z do nodo Z;
CHz = Todos os nodos filhos do Z;
$\lambda_Y(z)$ = Mensagem λ que o nodo Y manda para um certo valor do nodo Z

$$((x) = \sum_{k \in ZC} \{P(x|k) \prod_{i=1}^{j} (X(zi)\} \qquad (3)$$

Legenda da equação (3):

ZC = Todos as possíveis combinações de todos os possíveis valores de todos os

nodos pais do nodo X;

$\pi(x)$ = O valor π de um certo valor x do nodo X;

j = A quantidade dos nodos pais do nodo X;

$\pi_X(z_i)$ = Mensagem π que o valor do i-ésimo pai do X que compõe a combinação k manda para o nodo X.

Em nosso exemplo, as mensagens π que o nodo A manda para C são:

$\pi_C(A_0) = 0.3 * 1 = 0.3$ /*Neste caso, o nodo A não possui nenhum outro filho e, portanto, o resultado do produtório é 1*/

$\pi_C(A_1) = 0.7 * 1 = 0.7$

As mensagens π que o nodo B manda para C são:

$\pi_C(B_0) = 0.8 * 1 = 0.8$

$\pi_C(B1) = 0.2 * 1 = 0.2$

Neste ponto do exemplo, o conjunto A_e ainda é um conjunto vazio. Portanto, o nodo C não pertence a A_e e então calcularemos os valores π do nodo C:

$$\begin{aligned}\pi(C_0) &= P(C_0|A_0, B_0)\,\pi_C(A_0)\,\pi_C(B_0) + P(C_0|A_0, B_1)\,\pi_C(A_0)\,\pi_C(B_1) \\ &\quad + P(C_0|A_1, B_0)\,\pi_C(A_1)\,\pi_C(B_0) + P(C_0|A_1, B_1)\,\pi_C(A_1)\,\pi_C(B_1) \\ &= (0,4)(0,3)(0,8) + (0,5)(0,3)(0,2) + (0,2)(0,7)(0,8) + (0,9)(0,7)(0,2) \\ &= 0,364 \\ \pi(C_1) &= P(C_1|A_0, B_0)\,\pi_C(A_0)\,\pi_C(B_0) + P(C_1|A_0, B_1)\,\pi_C(A_0)\,\pi_C(B_1) \\ &\quad + P(C_1|A_1, B_0)\,\pi_C(A_1)\,\pi_C(B_0) + P(C_1|A_1, B_1)\,\pi_C(A_1)\,\pi_C(B_1) \\ &= (0,6)(0,3)(0,8) + (0,5)(0,3)(0,2) + (0,8)(0,7)(0,8) + (0,9)(0,7)(0,2) \\ &= 0,748\end{aligned}$$

No passo seguinte, atualiza os $P(x|a_e)$, onde x representa um dos possíveis valores que o nodo que recebeu a mensagem π, de acordo com a equação (4) e depois de atualizar o $P(x|a_e)$ para todos os possíveis valores do nodo, normalize os $P(x|a_e)$.

$$P(x|a_e) = \alpha\lambda(x)\pi(x) \qquad (4)$$

Legenda da equação (4):

α = Constante de normalização;

$\lambda(x)$ = O valor λ do valor x;
$\pi(x)$ = O valor π do valor x.

Em nosso exemplo, ainda não foi instanciado nenhum valor, portanto o conjunto $a_e = \{\}$:

$P(C_0|\{\}) = \alpha\lambda(C_0)\pi(C_0) = \alpha(1)(0,364) = 0,364\alpha$
$P(C_1|\{\}) = \alpha\lambda(C_1)\pi(C_1) = \alpha(1)(0,748) = 0,748\alpha$

Normalizando teremos:

$P(C_0|\{\}) = 0,364\alpha / (0,364\alpha + 0,748\alpha) = 0,327$
$P(C_1|\{\}) = 0,748\alpha / (0,364\alpha + 0,748\alpha) = 0,673$

No passo seguinte, serão enviadas as mensagens π para os nodos filhos e o processo continua até que o nodo que recebe a mensagem π não possua nenhum filho. Portanto, não há um alvo para enviar as mensagens π, ou o nodo que receberá a mensagem π é um nodo que representa uma das variáveis que foi instanciada, neste caso o nodo que receberá a mensagem π só atualizará os valores π sem mandar as mensagens para outros nodos.

No nosso exemplo, o nodo C não possui nenhum nodo filho e, portanto, não mandará nenhuma mensagem π.

Após este passo, caso o valor λ de todos os possíveis valores da variável seja 1, então não serão enviadas as mensagens λ. Caso contrário, serão mandadas as mensagens λ para todos os nodos pais tal que o nodo pai que receberá a mensagem λ não mande uma mensagem π para o nodo X ou que não seja um dos nodos que receberá uma evidência.

Em nosso exemplo, no passo da inicialização, o valor λ de todos os possíveis valores do nodo X é 1 e, portanto, as mensagens λ não serão mandadas.

Após o término do envio das mensagens π, a rede será atualizada com as evidências que foram providenciadas. A atualização da rede é feita de uma variável por vez, a variável que está atualizando a rede será representada pela V_e e o valor da evidência para a V_e é representada pela v_e.

O primeiro passo é atualizar todos os valores λ e π de V_e, tal que todos os possíveis valores de V_e que não são v_e, ou v_p, terão os seus valores λ e π assumindo o valor 0, assim como $P(v_p|a_e) = 0$. Enquanto os valores λ e π de v_e assumirão o valor 1 e o $P(v_e|a_e) = 1$.

Assume que no nosso exemplo, o V_e = A e $v_e = A_0$, então:

$A_e = A_e \cup \{A\} = \{A\}$
$a_e = a_e \cup \{A_0\} = \{A_0\}$

$\lambda(A_0) = 1$
$\pi(A_0) = 1$
$P(A_0|a_e) = 1$

$\lambda(A_1) = 0$
$\pi(A_1) = 0$
$P(A_1|a_e) = 0$

A seguir a V_e mandará as mensagens λ, que são calculadas pela equação (5), para os nodos pais de V_e que não pertencem a A_e (será enviada uma mensagem λ independente para cada possível valor do nodo pai) e os valores λ dos nodos pais de V_e serão atualizados pela equação (6).

$$\lambda_Y(x) = \sum_{m \in y} \{ \sum_{k \in ZC} [P(y|x, k) \prod_{i=1}^{j} \pi_Y(z_i)] \} \lambda(m)$$

Legenda da equação (5):

$\lambda_Y(x)$ = Mensagem λ que o nodo Y manda um certo valor x do nodo X;

y = Todos os possíveis valores do nodo Y;

ZC = Todos as possíveis combinações de todos os possíveis valores de todos os nodos pais do nodo Y;

j = A quantidade de nodos pais do nodo Y;

$\pi_Y(z_i)$ = Mensagem π que o valor do i-ésimo pai do Y que compõe a combinação k manda para o nodo Y;

$\lambda(m)$ = O valor λ do m.

$$\lambda(x) = \prod_{W \in CH_X} \lambda_W(x) \qquad (6)$$

Legenda da equação (6):

$\lambda(x)$ = O valor λ de um certo valor x do nodo X;

CH_X = Todos os nodos filhos do nodo X;

$\lambda_W(x)$ = A mensagem λ que o nodo filho W manda para um certo valor x do nodo X.

O nodo que recebe a mensagem λ, atualizará seus valores λ e mandará mensagens λ para seus pais que não pertencem a A_e. O processo continua até que o nodo que recebe a mensagem λ não possuir nenhum pai, ou que todos os pais pertençam a A_e. Quando todas as mensagens λ terminam de ser enviadas, os nodos que não puderam mandar as mensagens λ começarão a mandar as mensagens π para os seus filhos que não mandaram mensagem λ para eles. O processo para mandar as mensagens π são os mesmos que o envio das mensagens π do passo de inicialização.

No final de todo o processo, a probabilidade condicional das variáveis da rede será atualizada com a evidência da variável V_e. Este processo repetirá para todas as variáveis que pertencem a A_e.

No nosso exemplo, o nodo A não possui nenhum nodo pai, portanto não mandará mensagens λ e mandará as mensagens π para o nodo C:

$\pi_C(A_0) = 1 * 1 = 1$
$\pi_C(A_1) = 0 * 1 = 0$

$\pi(C_0) = P(C_0|A_0, B_0)\ \pi_C(A_0)\ \pi_C(B_0) + P(C_0|A_0, B_1)\ \pi_C(A_0)\ \pi_C(B_1)$
$+ P(C_0|A_1, B_0)\ \pi_C(A_1)\ \pi_C(B_0) + P(C_0|A_1, B_1)\ \pi_C(A_1)\ \pi_C(B_1)$
$= (0,4)(1)(0,8) + (0,5)(1)(0,2) + (0,2)(0)(0,8) + (0,9)(0)(0,2)$
$= 0,42$
$\pi(C_1) = P(C_1|A_0, B_0)\ \pi_C(A_0)\ \pi_C(B_0) + P(C_1|A_0, B_1)\ \pi_C(A_0)\ \pi_C(B_1)$
$+ P(C_1|A_1, B_0)\ \pi_C(A_1)\ \pi_C(B_0) + P(C_1|A_1, B_1)\ \pi_C(A_1)\ \pi_C(B_1)$
$= (0,6)(1)(0,8) + (0,5)(1)(0,2) + (0,8)(0)(0,8) + (0,1)(0)(0,2)$
$= 0,58$

$P(C_0|A_0) = \alpha\lambda(C_0)\pi(C_0) = \alpha(1)(0,42) = 0,42\alpha$
$P(C_1|A_0) = \alpha\lambda(C_1)\pi(C_1) = \alpha(1)(0,58) = 0,58\alpha$

Normalizando teremos:

$P(C_0|A_0) = 0,42\alpha / (0,42\alpha + 0,58\alpha) = 0,42$
$P(C_1|A_0) = 0,58\alpha / (0,42\alpha + 0,58\alpha) = 0,58$

No nosso exemplo, ainda foi instanciado o nodo C com o valor C_1, portanto:

$A_e = A_e \cup \{C\} = \{A, C\}$
$a_e = a_e \cup \{C_1\} = \{A_0, C_1\}$

$\lambda(C_0) = 0$
$\pi(C_0) = 0$
$P(C_0|a_e) = 0$

$\lambda(C_1) = 1$
$\pi(C_1) = 1$
$P(C_1|a_e) = 1$

No próximo passo, o nodo C mandará mensagens λ para os seus pais que não pertencem a A_e, que é o nodo B:

$$\lambda_C(B_0) = [P(C_0|B_0, A_0)\pi_C(A_0) + P(C_0|B_0, A_1)\ \pi_C(A_1)]\ \lambda(C_0)$$
$$+ [P(C_1|B_0, A_0)\pi_C(A_0) + P(C_1|B_0, A_1)\ \pi_C(A_1)]\ \lambda(C_1)$$
$$= [(0,4)(1) + (0,2)(0)](0) + [(0,6)(1) + (0,8)(0)](1)$$
$$= 0,6$$
$$\lambda_C(B_1) = [P(C_0|B_1, A_0)\pi_C(A_0) + P(C_0|B_1, A_1)\ \pi_C(A_1)]\ \lambda(C_0)$$
$$+ [P(C_1|B_1, A_0)\pi_C(A_0) + P(C_1|B_1, A_1)\ \pi_C(A_1)]\ \lambda(C_1)$$
$$= [(0,5)(1) + (0,9)(0)](0) + [(0,5)(1) + (0,1)(0)](1)$$
$$= 0,5$$

$\lambda\ (B_0) = \lambda_C(B_0) = 0,6$ /*Neste caso o nodo B só possui o nodo C como filho*/
$\lambda\ (B_1) = \lambda_C(B_1) = 0,5$

$P(B_0|\{A_0, C_1\}) = \alpha\lambda(B_0)\pi(B_0) = \alpha(0,6)(0,8) = 0,48\alpha$
$P(B_1|\{A_0, C_1\}) = \alpha\lambda(B_1)\pi(B_1) = \alpha(0,5)(0,2) = 0,1\alpha$

Normalizando teremos:

$P(B_0|\{A_0, C_1\}) = 0,48\alpha / (0,48\alpha + 0,1\alpha) = 0,828$
$P(B_1|\{A_0, C_1\}) = 0,1\alpha / (0,48\alpha + 0,1\alpha) = 0,172$

Como o nodo B não possui nenhum pai, não será mandada mensagens λ. O nodo B não possui nenhum outro filho a não ser o nodo C que mandou mensagens λ para o nodo B. Portanto, o nodo B não mandará nenhuma mensagem π também.

Como podemos ver, durante o processo de atualização da rede, as probabilidades condicionais $P(x|a_e)$ são atualizadas, o que possibilita a obtenção da

probabilidade condicional de qualquer valor de qualquer variável da rede depois que um conjunto de evidências é providenciado. Este é o processo de inferência numa rede simples conectada, proposta pela [30]. O Algoritmo 1 descreve o processo de inicialização, o Algoritmo 2 descreve o envio das mensagens λ, o Algoritmo 3 descreve o envio das mensagens π, o Algoritmo 4 descreve o processo de atualização da rede e o Algoritmo 5 descreve o processo de inferência.

```
Procedure inicialização;
input:   B: Uma rede Bayesiana (G,P) onde G = (V,E);
         A_e: Um conjunto de variáveis que receberão as evidências;
         a_e: Um conjunto de evidências;
output:  Nenhuma;
init:    Nennuma;
begin

    for cada X ∈ V do

    begin
        for cada possível valor x de X do
            λ(x) := 1;
        for cada nodo pai Z de X do
            for cada possível valor z de Z do
                λ_X(z) := 1;
        for cada nodo filho Y de X do
            for cada possível valor x de X do
                π_Y(x) := 1;
    end; {for}
    for cada nodo raíz R de G do
    begin
        for cada possível valor r de R do
        begin
            P(r|a_e) := P(r);
            π(r) := P(r);
        end; {for}
        for cada nodo filho Y de R do
            envia_mensagem_π(R, Y, A_e, a_e);
    end; {for}
end. {inicialização}
```

Algoritmo 1. O processo de inicialização do algoritmo de inferência.

Procedure envia_mensagem_λ;
input: Y: Nodo que está enviando a mensagem λ;
X: Nodo que está recebendo a mensagem λ;
A_e: Um conjunto de variáveis que receberão as evidências;
a_e: Um conjunto de evidências;
outpu: Nenhuma;
init: Nenhuma;
begin
for cada possível valor x de X **do**
begin
Utiliza a equação (5) para calcular o $\lambda_Y(x)$;
Utiliza a equação (6) para calcular o $\lambda(x)$;
Utiliza a equação (4) para calcular o $P(x|a_e)$;
end; {for}
normaliza $P(x|a_e)$;
for cada nodo pai Z de X tal que $Z \notin A_e$ **do**
envia_mensagem_$\lambda(X, Z, A_e, a_e)$;
for cada nodo filho W de X tal que $W \neq Y$ **do**
envia_mensagem_$\pi(X, W, A_e, a_e)$;
end. {envia_mensagem_λ}

Algoritmo 2. O processo para envio das mensagens λ

```
Procedure envia_mensagem_π;
input:   Z: Nodo que está enviando a mensagem π;
         X: Nodo que está recebendo a mensagem π;
         A_e: Um conjunto de variáveis que receberão as evidências;
         a_e: Um conjunto de evidências;
output:  Nenhuma;
init:    Nenhuma;
being
    for cada possível valor z do nodo Z do
        Utiliza a equação (2) para calcular π_X(z);
    if X ∉ A_e then
    begin
        for cada possível valor x do X do
        begin
            Utiliza a equação (3) para calcular π(x);
            Utiliza a equação (4) para calcular P(x|a_e);
        end; {for}
        normaliza P(x|a_e);
        for cada nodo filho Y do X do
            envia_mensagem_π(X, Y, A_e, a_e);
    end; {if-then}
    if algum λ(x) de todos os possíveis valores do X é diferente de 1 then
        for cada nodo pai W do X tal que W ≠ Z e W ∉ A_e do
            envia_mensagem_λ(X, W, A_e, a_e);
end. {envia_mensagem_π}
```

Algoritmo 3. O processo de envio das mensagens π.

Procedure atualiza;
input: B: Uma rede Bayesiana (G, P) onde G = (V, E);
A_e: Um conjunto de variáveis que receberão as evidências;
a_e: Um conjunto de evidências;
V_e: A variável que recebeu a nova evidência;
v_e: A nova evidência que chegou na rede;
output: Nenhuma;
init: $A_e := A_e \cup \{V_e\}$;
$a_e := a_e \cup \{v_e\}$;
$\lambda(v_e) := 1$;
$\pi(v_e) := 1$;
$P(v_e|a_e) := 1$;
for cada possível valor v do V_e **do**
begin
$\lambda(v) := 0$;
$\pi(v) := 0$;
$P(v|a_e) := 0$;
end; {for}
begin
for cada nodo pai Z do V_e tal que $Z \notin A$ **do**
envia_mensagem_$\lambda(V_e, Z, A_e, a_e)$;
for cada nodo filho Y do Ve **do**
envia_mensagem_$\pi(V_e, Y, A_e, a_e)$;
end. {atualiza}

Algoritmo 4. O processo de atualização da rede Bayesiana na inferência.

```
Procedure inferência;
input:   B: Uma rede Bayesiana (G, P) onde G = (V, E);
         e: Um conjunto de evidências;
output:  B': A rede Bayesiana B que foi atualizada com novas probabilidades
                    condicionais;
init:    A_e := {};
         a_e := {};
         B' := B;
         E := void;
begin
    inicialização(B', A_e, a_e);
    for cada valor e' de e do
    begin
        E := a variável que corresponde a e';
        atualiza(B', A_e, a_e, E, e');
    end; {for}
    return B';
end. {inferência}
```

Algoritmo 5. O algoritmo de inferência numa rede Bayesiana.

Em [28] foi discutido um método chamado *conditioning* para aplicar o algoritmo de inferência discutido acima numa rede multi conectada. A idéia do *conditioning* é achar os nodos que, ao serem retirados de uma rede multi conectada, tornaria a rede simples conectada. Estes nodos são chamados de *loop-cutset*. Depois de achar estes nodos, para cada possível combinação de todos os possíveis valores destes nodos, cria-se uma rede Bayesiana instanciando esses nodos com a combinação correspondente. Por exemplo, se na rede Bayesiana da Figura 3 tiver um nodo D, binário, que seja pai do nodo A e do nodo B, a rede se torna uma rede multi conectada.

Neste mesmo exemplo, se retirarmos o nodo D, a rede simples se tornaria conectada. Portanto, criaremos duas redes Bayesianas: uma instanciando o nodo D com seu primeiro valor e a outra instanciando o nodo D com seu segundo valor,. As probabilidades de A e B serão modificadas, onde na primeira rede, P(a1) = P(a1|d1), P(a2) = P(a2|d1), P(b1) = P(b1|d1) e P(b2) = P(b2|d1) e na segunda rede, P(a1) = P(a1|d2), P(a2) = P(a2|d2), P(b1) = P(b1|d2) e P(b2) = P(b2|d2). Despois das modificações, o nodo D será retirado destas duas redes, tornando-as simples conectadas. Desta forma é possível aplicar o Algoritmo 5 nestas duas redes para realizar a inferência.

Depois, para cada nodo $X \in V - \{E \cup C\}$, onde V são as variáveis do problema, E é o conjunto de nodos instanciados e C é o conjunto de *loop-cutset*, podemos utilizar a equação (7) para calcular as probabilidades de cada nodo.

$$P(x_i) = \sum_{c \in CV} P(x_i|e_v,c)\, P(c|e_v) \qquad (7)$$

Legenda da equação (7):

$P(x_i)$ = Probablidade de i-ésimo valor do nodo X após a inferência;

CV = Todas as possíveis combinações de todos os possíveis valores de todos os nodos que pertencem a C;

$P(x_i|e,c)$ = Probabilidade condicional de i-ésimo valor do nodo X dado as evidências e_v e a combinação c. Pode ser obtido aplicando o Algoritmo 5;

$P(c|e_v)$ = Probabilidade condicional da combinação c dado as evidências e_v. Pode ser calculado utilizando a igualdade $P(c|e_v) = \alpha P(e_v|c)\, P(c)$, onde se o conjunto C contém somente nodos raízes, o P(c) já tem a disposto.

Porém, como mencionado em [28], nem todos os casos o *loop-cutset* só conterá nodos raízes e achar um conjunto mínimo de *loop-cutset* é um problema NP difícil. Em [34] desenvolve métodos para computar o P(c) em casos gerais.

Além do método *conditioning*, existem outros métodos que também buscam a mesma finalidade, como o método *clustering* mencionado em [28].

2.2. Aprendizado de redes Bayesianas

O aprendizado de uma rede Bayesiana é um problema NP completo. Existem diversas abordagens na literatura técnica para tentar resolver este problema. Para esta pesquisa, focamos a nossa atenção para o algoritmo K2[28] e o algoritmo IC[22] que seguem direções distintas para aprender uma rede Bayesiana. Também vamos estudar o uso do classificador *Naïve Bayes* no método *BayesFuzzy*, porque o *Naïve Bayes* é um classificador Bayesiano mais direcionado para o problema de classificação. Abaixo apresentaremos um pouco sobre cada um deles.

2.2.1. Algoritmo K2

O K2[28] é um algoritmo de busca heurística que tem como objetivo encontrar a estrutura mais provável de uma rede Bayesiana. Para alcançar este objetivo, o

algoritmo procura maximizar $P(B_S/D)$ de uma rede, onde B_S denota a estrutura daquela rede Bayesiana, D denota o conjunto de dados (casos) relacionado àquela rede e $P(B_S/D)$ denota a probabilidade de B_S dado o conjunto de dados *D*.

Uma das entradas para o algoritmo K2, é uma ordenação das variáveis (ou vetor de variáveis), que é um dos fatores essenciais que determinará o desempenho do algoritmo. Para obter melhores resultados, esta ordenação deve ser dada da seguinte maneira: considere uma variável qualquer do vetor de variáveis, variável X. Ela é sempre um possível pai de todas as variáveis que são posteriores a ela e nenhuma variável posterior a X pode ser pai de X. Então a busca heurística se baseia nesta ordenação para aprender uma rede Bayesiana.

Utilizaremos as seguintes notações:

- i: índice do nodo na ordenação das variáveis;
- π_i: conjunto de pais do nodo x_i;
- ψ_i: conjunto de todas as combinações de todos os possíveis valores de todos os pais do nodo x_i. Ou seja, se $p_1...p_s$ são todos os pais possíveis do nodo x_i, e $v(p_i)$ denota todos os possíveis valores do pai p_i, então ψ_i é o produto Cartesiano de $v(p_1) \times ... \times v(p_s)$;
- $q_i = |\psi_i|$;
- V_i: o conjunto de todos os possíveis valores do nodo x_i;
- $r_i = |V_i|$;
- N_{ijk}: número de casos (i.e. instâncias) no conjunto de treinamento, onde o atributo x_i é instanciado com o seu k^{th} valor e os pais do x_i no conjunto π_i são instanciados com o j^{th} instanciação do ψ_i.
- N_{ij}: $\sum_{k=1}^{r_i} N_{ijk}$. Isto é, a quantidade de instâncias no conjunto de treinamento onde os pais do x_i são instanciados com a j^{th} instanciação do conjunto ψ_i.

Segue o pseudocódigo do algoritmo:

```
Procedure K2;
input:  x1...xn: Um conjunto de n variáveis ordenadas;
        u: O máximo de número de pais que um nodo possui;
        D: Conjunto de dados de treinamento;
output: R: A rede Bayesiana aprendida;
init:   π1... πn := {};
        Pold := void;
        Pnew := void;
        Continua := void;
        z := void;
        R := void;
begin
    for i := 1 to n do
    begin
        Pold := g(i, πi); /*Essa função é computado usando equação (2)*/
        Contiuna := true;
        while Continua and |πi| < u do
        begin
            z := O nodo em Pred(xi) -πi que maximiza g(i, πi ∪ {z});
            /*Pred(xi) são os nodos que precedem xi na ordenação dos nodos*/
            Pnew := g(i, πi ∪ {z}));
            if  Pnew > Pold then
            begin
              Pold := Pnew;
              πi := πi ∪ {z};
            end;{if-then}
            else
                Continua := false;
        end;{while}
        Estabelece arcos que direcionam dos pais do xi até o xi em R;
    end;{for}
    return R;
end.{K2}
```

Algoritmo 6. Pseudocódigo do algoritmo K2.

$$g(i, \pi_i) = \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}! \quad (8)$$

Para melhorar a velocidade de processamento e a precisão dos cálculos do algoritmo, as funções $g(i, \pi_i)$ e $g(i, \pi_i \cup \{z\})$ podem ser substituídas por $\log(g(i, \pi_i))$ e $\log(g(i, \pi_i \cup \{z\}))$, respectivamente. Na seção seguinte será apresentado um exemplo do uso do algoritmo K2 para melhorar o entendimento do algoritmo.

2.2.2. Exemplo de uso do algoritmo K2

Considere um problema com três variáveis chamados “Atributo 1”, “Atributo 2” e “Atributo 3”, onde os possíveis valores para o Atributo 1 e Atributo 2 são {0, 1}, e para o Atributo 3, os possíveis valores são {0, 1, 2}. A **Tabela 2** servirá como o conjunto de dados de treinamento D para este problema.

Tabela 2 Conjunto de treinamento do exemplo de aplicação de K2.

Instâncias	Atributo 1	Atributo 2	Atributo 3
1	0	0	1
2	1	0	0
3	1	1	0
4	0	1	2
5	1	1	2

Considere também que o vetor de ordenação das variáveis de entrada para o algoritmo seja {Atributo 2, Atributo 1, Atributo 3}, o máximo de pais seja 2 e que $Pred(x_i)$ denota os nodos que antecede ao nodo x_i. Então x_1 representa o Atributo 2, x_2 representa o Atributo 1 e x_3 representa o Atributo 3.

No processamento do primeiro nodo do vetor de entrada (Atributo 2):

Considere i = 1 e $r_1 = 2$, porque o x_1 (Atributo 2) possui dois valores possíveis {0, 1}.

1.: $\pi_1 = \{\}$; $q_1 = 0$;

2.: $P_{old} = g(1, \{\}) = \prod_{j=1}^{0} (2 - 1)! / (N_{1j} + 2 - 1)! \prod_{k=1}^{2} N_{1jk}!$

Nas situações onde q_i é 0, ou seja, o nodo atual não possui nenhum pai, o somatório que envolve o q_i será desconsiderado no cálculo, portanto a função $g(i, \pi_i)$

assume a forma:

$$g(i, \pi_i) = \frac{(r_i - 1)!}{(N_i + r_i - 1)!} \prod_{k=1}^{r_i} N_{ik}!$$

- $N_{1.1} = 2$; (Instâncias 1 e 2)
- $N_{1.2} = 3$; (Instâncias 3, 4 e 5)
- $N_{1.} = N_{1.1} + N_{1.2} = 5$

Portanto,

$P_{old} = (2 - 1)! / (N_{1.} + 2 - 1)! \prod_{k=1}^{2} N_{1.k}! = \{1! / (5 + 2 - 1)!\} * (2!) * (3!)$

$= \{1 / 6!\} * 2 * 6 = 1/720 * 2 * 6 = 1/60$

3.: Já que $Pred(x_1) = \{\}$, a iteração termina aqui e conclui que $\pi_1 = \{\}$;

No processamento do segundo nodo do vetor de entrada (Atributo 1):

Considere i = 2 e $r_2 = 2$, porque o x_2 (Atributo 1) possui dois valores possíveis {0, 1}.

1.: $\pi_2 = \{\}$; $q_2 = 0$;

2.: $P_{old} = g(2, \{\}) = \prod_{j=1}^{0} (2 - 1)! / (N_{2j} + 2 - 1)! \prod_{k=1}^{2} N_{2jk}!$

- $N_{2.1} = 2$; (Instâncias 1 e 4)
- $N_{2.2} = 3$; (Instâncias 2, 3 e 5)
- $N_{2.} = N_{1.1} + N_{1.2} = 5$

Portanto,

$P_{old} = (2 - 1)! / (N_{2.} + 2 - 1)! \prod_{k=1}^{2} N_{2.k}! = \{1! / (5 + 2 - 1)!\} * (2!) * (3!)$

$= \{1 / 6!\} * 2 * 6 = 1/720 * 2 * 6 = 1/60$

3.: $Pred(x_2) - \pi_2 = \{\text{Atributo 2}\}$, portanto o único nodo que maximiza a função $g(2, \pi_2 \cup \{z\})$ é o Atributo 2. $q_2 = 2$, porque só existem duas combinações possíveis.

$P_{new} = g(2, \{\text{Atributo 2}\}) = \prod_{j=1}^{2} \{(2 - 1)! / (N_{2j} + 2 - 1)!\} \prod_{k=1}^{2} N_{2jk}!$

- $N_{211} = 1$; (Instância 1)
- $N_{212} = 1$; (Instância 2)
- $N_{221} = 1$; (Instância 4)
- $N_{222} = 2$; (Instâncias 3 e 5)
- $N_{21} = N_{211} + N_{212} = 2$;
- $N_{22} = N_{221} + N_{222} = 3$;

Portanto,

$P_{new} = \prod_{j=1}^{2} (2 - 1)! / (N_{2j} + 2 - 1)! \prod_{k=1}^{2} N_{2jk}!$

$= \{(2 - 1)! / (N_{21} + 2 - 1)!\} * (1!) * (1!) * \{(2 - 1)! / (N_{22} + 2 - 1)!\} * (1!)$

* (2!)

= (1/6) * 1 * 1 * (1/24) * 1 * 2 = 2/144 = 1/72

4.: $P_{new} < P_{old}$ portanto, $\pi_2 = \{\}$.

5.: $Pred(x_2) - \pi_2 = \{\}$. Não existem mais elementos no $Pred(x_2) - \pi_2$, portanto, termina a interação;

No processamento do terceiro nodo do vetor de entrada (Atributo 3):

Neste ciclo, i = 3, r_3 é igual a três, porque o x_3 (Atributo 3) possui três valores possíveis {0, 1, 2}.

1.: $\pi_3 = \{\}$; $q_3 = 0$;

2.: $P_{old} = g(3, \{\}) = \prod_{j=1}^{0} (3 - 1)! / (N_{3j} + 3 - 1)! \prod_{k=1}^{3} N_{3jk}!$

- $N_{3.1} = 2$; (Instâncias 2 e 3)
- $N_{3.2} = 1$; (Instância 1)
- $N_{3.3} = 2$; (Instâncias 4 e 5)
- $N_{3.} = N_{3.1} + N_{3.2} + N_{3.3} = 5$

Portanto,

P_{old} = $(3 - 1)! / (N_{3.} + 3 - 1)! \; \Pi^{3}_{k=1} N_{3.k}!$ = {2! / (5 + 3 – 1)!} * (2!) * (1!) * (2!)

= {2 / 7!} * 2 * 1 * 2 = 1/2520 * 4 = 1/630

3.: $Pred(x_3) - \pi_3$ = {Atributo 2, Atributo 1}

$g(3, \pi_3 \cup \{\text{Atributo 2}\}) = \prod_{j=}^{2} (3 - 1)! / (N_{3j} + 3 - 1)! \prod_{k=1}^{2} N_{3jk}!$

- $N_{311} = 1$; (Instância 2)
- $N_{312} = 1$; (Instância 1)
- $N_{313} = 0$;
- $N_{321} = 1$; (Instância 3)
- $N_{322} = 0$;
- $N_{323} = 2$; (Instâncias 4 e 5)
- $N_{31} = N_{311} + N_{312} + N_{313} = 2$;
- $N_{32} = N_{321} + N_{322} + N_{323} = 3$;

$g(3, \pi_3 \cup \{\text{Atributo 2}\})$ = {(2!)/(2 + 3 – 1)!} * (1!) * (1!) * (0!) * {(2!)/(3 + 3 – 1)!} * (1!) * (0!) * (2!)

= {2/(4!)} * 1 * 1 * 1 * {2/(5!)} * 1 * 1 * 2 = (1/12) * (1/60) * 2 = 1/360

$g(3, \pi_3 \cup \{\text{Atributo 1}\}) = \prod_{j=1}^{2} (3 - 1)! / (N_{3j} + 3 - 1)! \prod_{k=1}^{3} N_{3jk}!$

- $N_{311} = 0$;
- $N_{312} = 1$; (Instância 1)

- N_{313} = 1; (Instância 4)
- N_{321} = 2; (Instâncias 2 e 3)
- N_{322} = 0;
- N_{323} = 1; (Instância 5)
- $N_{31} = N_{311} + N_{312} + N_{313}$ = 2;
- $N_{32} = N_{321} + N_{322} + N_{323}$ = 3;

g(3, $\pi_3 \cup$ {Atributo 1})

= {(2!)/(2 + 3 – 1)!} * (0!) * (1!) * (1!) * {(2!)/(3 + 3 – 1)!} * (2!) * (0!) * (1!)
= {2/(4!)} * 1 * 1 * 1 * {2/(5!)} * 2 * 1 * 1 = (1/12) * (1/60) * 2
= 1/360

Como os dois empatam, então será escolhido o primeiro nodo como sendo z. Caso tivessem resultados diferentes, aquele nodo que, adicionado ao π_i obtivesse a maior probabilidade, seria considerado o z.

- P_{new} = g(3, {Atributo 2}) =1/360

4.: $P_{new} > P_{old}$ portanto, π_3 = {Atributo 2}. P_{old} = 1/360;

5.: Pred(x_3) -π_i = {Atributo 1}, portanto o único nodo que maximiza a função g(3, $\pi_3 \cup$ {z}) é o Atributo 2. q_2 = 4, porque existem quatro combinações possíveis, {(0, 0); (0, 1); (1, 0); (1, 1)}

$$P_{new}(x_3) = g(3, \pi_3 \cup \{\text{Atributo 1}\})$$
$$= \prod_{j=1}^{2} (3-1)! \,/\, (N_{3j} + 3 - 1)! \prod_{k=1}^{3} N_{3jk}!$$

- N_{311} = 0; (Instâncias que possuem valores (0, 0, 0), neste caso nenhuma)
- N_{312} = 1; (Instâncias que possuem valores (0, 0, 1), neste caso instância 1)
- N_{313} = 0; (Instâncias que possuem valores (0, 0, 2), neste caso nenhuma)
- N_{321} = 0; (Instâncias que possuem valores (0, 1, 0), neste caso nenhuma)
- N_{322} = 0; (Instâncias que possuem valores (0, 1, 1), neste caso nenhuma)
- N_{323} = 1; (Instâncias que possuem valores (0, 1, 2), neste caso instância 4)
- N_{331} = 1; (Instâncias que possuem valores (1, 0, 0), neste caso instância 2)

- N_{332} = 0; (Instâncias que possuem valores (1, 0, 1), neste caso nenhuma)
- N_{333} = 0; (Instâncias que possuem valores (1, 0, 2), neste caso nenhuma)
- N_{341} = 1; (Instâncias que possuem valores (1, 1, 0), neste caso instância 3)
- N_{342} = 0; (Instâncias que possuem valores (1, 1, 1), neste caso nenhuma)
- N_{343} = 1; (Instâncias que possuem valores (1, 1, 2), neste caso instância 5)
- $N_{31} = N_{311} + N_{312} + N_{313} = 1$;
- $N_{32} = N_{321} + N_{322} + N_{323} = 1$;
- $N_{33} = N_{331} + N_{332} + N_{333} = 1$;
- $N_{34} = N_{341} + N_{342} + N_{343} = 2$;

g(3, $\pi_3 \cup$ {Atributo 1})

= {(2!)/(1 + 3 – 1)!} * (0!) * (1!) * (0!) * {(2!)/(1 + 3 – 1)!} * (0!) * (0!) * (1!) * {(2!)/(1 + 3 – 1)!} * (1!) * (0!) * (0!) * {(2!)/(2 + 3 – 1)!} * (1!) * (0!) * (1!)
= {2/(3!)} * 1 * 1 * 1 * {2/(3!)} * 1 * 1 * 1 * {2/(3!)} * 1 * 1 * 1 * {2/(4!)} * 1 * 1 * 1 = (1/3) * (1/3) * (1/3) * (1/12) = 1/108

6.: $P_{new} > P_{old}$ portanto, π_3 = {Atributo 2, Atributo 1}. P_{old} = 1/108;

7.: Pred(x_3) $-\pi_3$ = {}, não existem mais elementos no Pred(x_3) $-\pi_3$, portanto termina a iteração;

A rede aprendida é mostrada na **Figura 4**.

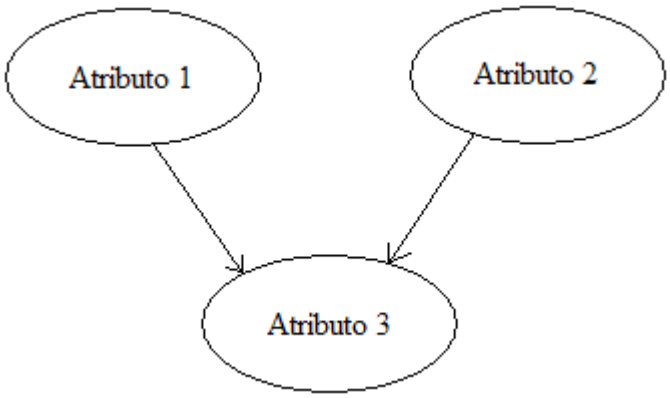

Figura 4 Rede Bayesiana aprendida pelo algoritmo K2 no exemplo da Seção 2.2.2.

2.2.3. Algoritmo IC

O IC[22] é um algoritmo de aprendizagem de rede Bayesiana que se baseia nos conceitos de independência condicional. Ele realiza uma busca exaustiva para cada par de variáveis do problema para determinar se eles podem ser considerados condicionalmente independentes ou não. O algoritmo pode ser descrito em três passos, que são:

1. Encontrar uma estrutura não direcionada e incluir arcos não direcionados de dois a dois nodos, caso os dois nodos sejam condicionalmente dependentes, ou seja, não existem separadores entre os dois nodos;
2. Para cada tripla de nodos conectados da forma A-B-C sendo A, B e C variáveis do problema, formar estruturas-V A->B<-C se A e C são condicionalmente dependentes dado o B;
3. Direcionar os demais arcos evitando a formação de novas estruturas-V e não tornar o grafo cíclico;

O algoritmo apresentado, ignora o esforço computacional envolvido no processo; alguns outros algoritmos desta classe se baseiam neste algoritmo para implementar uma forma mais prática e que seja computacionalmente viável. O Algoritmo 7 mostra um dos possíveis pseudocódigos do algoritmo IC:

```
Procedure IC;
input:  x1...xn: Um conjunto de n variáveis;
        D: Conjunto de dados de treinamento;
output: R: A rede Bayesiana aprendida;
init:   R := Uma rede Bayesiana com n nodos onde todos os nodos conectam
entre si;
begin
    for i := 1 to n do /*Para cada par de nodos*/
        for j := 1 to n do
            if i ≠ j
        and xi e xj são condicionalmente independentes de acordo com D then
                Remove arco que conecta entre xi e xj na R;

    for i := 1 to n do /*Para cada tripla de nodo*/
        for j := 1 to n do
            if i ≠ j then
                for k := 1 to n do
                    if k ≠ i and k ≠ j
                and xi, xj e x= estão conectados na forma xi – xj – xk then
                        Forma a estrutura V, xi -> xj <- xk em R;

    for i := 1 to n do /*Para cada par de nodos*/
        for j := 1 to n do
            if o arco entre xi e xj não está direcionado then
                Direciona o arco entre xi e xj sem formar novas estruturas-V e
                não transformar R cíclico;
    return R;
```

Algoritmo 7. Pseudocódigo do algoritmo IC.

2.2.4. Exemplo de uso do algoritmo IC

Considera-se um problema com 5 variáveis, A, B, C, D e E, onde o A é condicionalmente dependente com C e D, o B é condicionalmente dependente com D, C é condicionalmente dependente com A e E, o D é condicionalmente dependente com A, B e E, e E é condicionalmente dependente com C e D. Portanto, o resultado do passo 1 é mostrado na **Figura 5**.

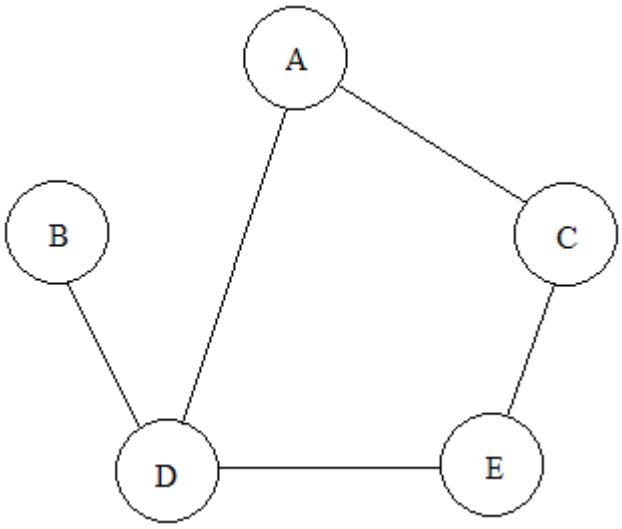

Figura 5 Resultado do passo 1 de exemplo de aplicação do algoritmo IC.

Suponha que A e B são dependentes do dado D, e C e D são condicionalmente dependentes do dado E. Portanto, o resultado do passo 2 resulta na estrutura apresentada na **Figura 6**.

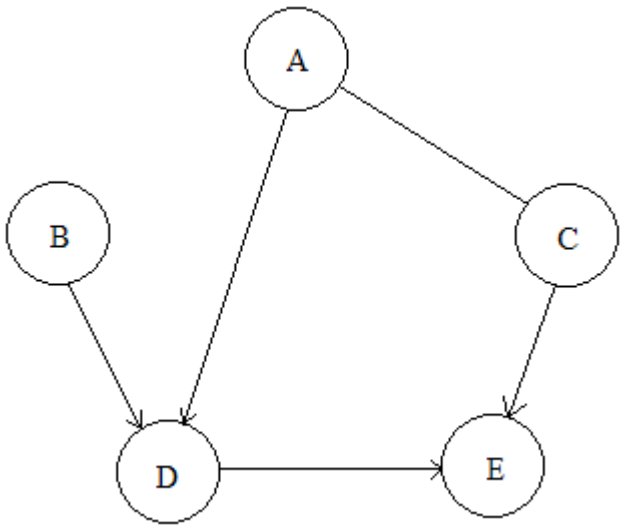

Figura 6 Resultado do passo 2 de exemplo de aplicação do algoritmo IC.

No passo 3 do algoritmo os arcos restantes são direcionados de forma a não formar novas estruturas V e não formar círculos direcionados. Portanto um dos possíveis resultados é apresentado na **Figura 7**.

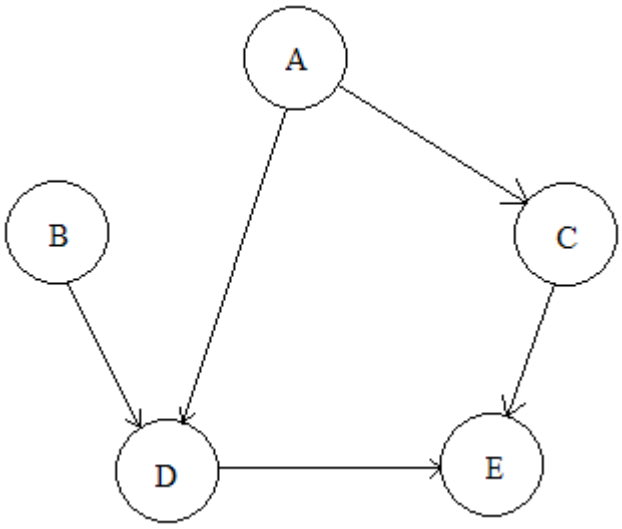

Figura 7 Resultado do passo 3 de exemplo de aplicação do algoritmo IC.

Existem propostas na literatura técnica para realizar cada um destes passos de forma mais eficiente ou de forma mais prática. Por exemplo, uma das formas de se realizar o passo 1 é iniciar o grafo conectando todos os nodos entre si. Então, retirar os arcos entre os nodos que são condicionalmente independentes testando os possíveis separadores, como os seus vizinhos mais próximos. Esta abordagem é para tentar resolver o problema de poder existir uma grande quantidade de possíveis conjuntos de separadores para serem testados para cada dois nodos que estamos tentando determinar, se devemos conectar um arco entre eles, ou seja, se são condicionalmente independentes.

2.3. *Naive Bayes*

Os classificadores Bayesianos buscam normalmente a distribuição da probabilidade real de todas as variáveis do problema. Existe, entretanto, um tipo de classificador Bayesiano que se especializa na classificação de uma única variável. Esse tipo de classificador é conhecido como Classificador *Naïve Bayes* (NB). Um classificador Bayesiano comum sempre tentará atribuir o valor mais provável para a variável da classe num processo de classificação. Porém, para realizar esta tarefa, é necessário identificar as relações entre as variáveis. Esta identificação dos relacionamentos muitas vezes não é otimizada para a tarefa de classificação de uma única variável. Portanto, os classificadores *Naïve Bayes* utilizam uma abordagem diferente para encontrar este valor mais provável.

A abordagem NB considera que todos os nodos são condicionalmente independentes entre si (dado o valor da classe) e que todos eles influenciam

diretamente a variável da classe. Ou seja, todos os nodos de uma rede *Naïve Bayes*, são filhos da variável classe. Desta forma, não é necessário se preocupar com as relações entre os nodos e sim apenas com a relação de cada um dos nodos com a variável da classe, melhorando o efeito de classificação. A **Figura 8** mostra a versão *Naïve Bayes* da rede Bayesiana mostrada na **Figura 1.**

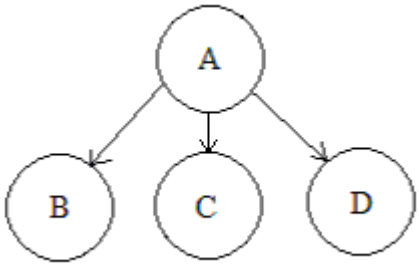

Figura 8 Exemplo de uma rede Naïve Bayes.

2.4. Seleção de atributos - *Markov Blanket*

Em algumas situações reais, a quantidade de variáveis que descreve o problema é grande. Isto geralmente torna o processamento da rede Bayesiana computacionalmente ineficiente ou computacionalmente intratável. Portanto, pode ser interessante desconsiderar variáveis que não possuem uma grande influência para a solução do problema. Considerando novamente o exemplo dado na **Figura 1**, suponha que desejamos focar o nosso problema na classificação do nodo D. Para este objetivo, podemos retirar o nodo A da rede para tornar o processamento mais eficiente. A remoção do nodo A é uma aplicação da definição de *Markov Blanket*.

Numa rede Bayesiana um dado nodo X pode ser influenciado pela instanciação de um nodo distante Z. Porém, a instanciação de alguns nodos que se situam mais perto de X, serve como uma “barreira” para estas influências fazendo com que a instanciação de um nodo que não faz parte desta “barreira”, não tenha mais influências sobre o nodo X. Esta é justamente a idéia principal do *Markov Blanket*, ou seja, a de localizar esta “barreira” dado um nodo X qualquer de uma rede Bayesiana.

A definição do *Markov Blanket* é: dado um nodo X, o *Markov Blanket* do nodo X é composto por todos os nodos que sejam pais de X, filhos de X e pais dos filhos de X. Veja o exemplo da **Figura 9**, onde temos 8 variáveis, A, B, C, D, E, F, G e H. Considere que desejamos encontrar os nodos que pertencem ao *Markov Blanket* do nodo F.

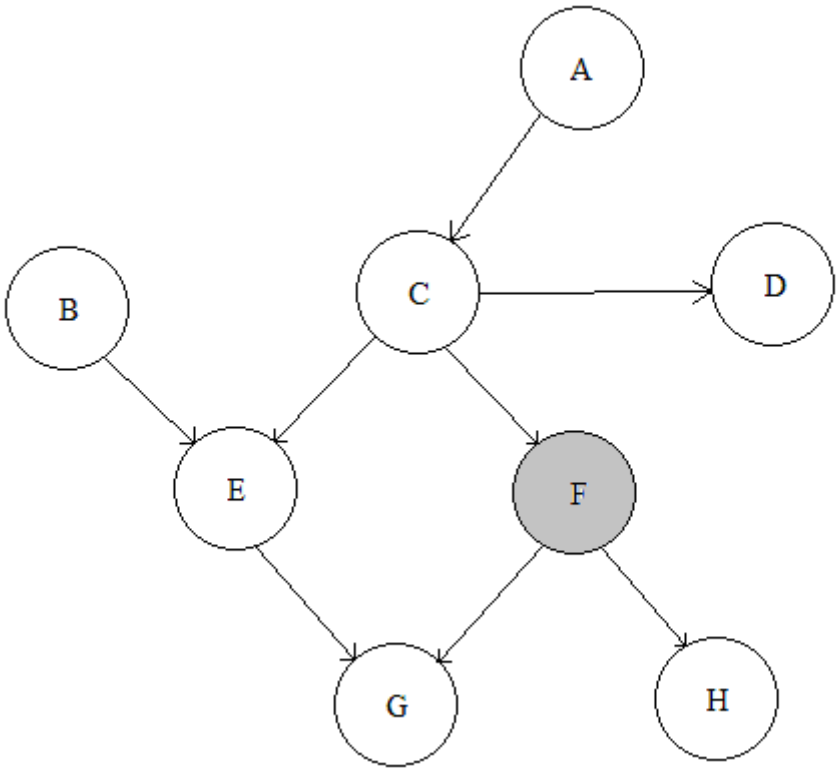

Figura 9 Exemplo de uma rede Bayesiana antes da identificação de Markov Blanket no nodo F.

Os nodos que satisfazem à definição do *Markov Blanket* do nodo F são, C, E, G e H, como mostra a **Figura 10**.

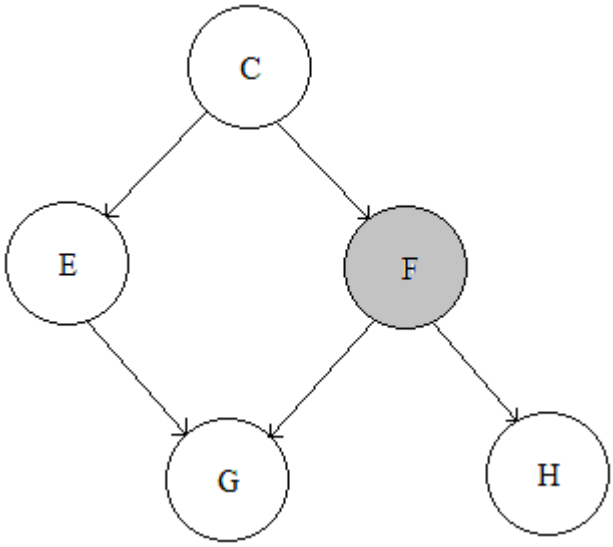

Figura 10 Markov Blanket do nodo F do exemplo dado na **Figura 9**.

O nodo C faz parte do *Markov Blanket* de F porque é o pai de F. Os nodos G e H

também porque são filhos de F e o nodo E é membro do Markov Blanket porque ele é o pai de G que é filho de F. Desta forma, se tivermos um problema que pode ser representado pela rede Bayesiana da **Figura 9**, conseguiremos reduzir este problema de 8 variáveis para 5 variáveis diminuindo o espaço de busca e tempo computacional.

3. LÓGICA *FUZZY*

A lógica *fuzzy* é conhecida pela sua capacidade de tratar incerteza e imprecisão assim como de permitir um ganho na compreensão ao utilizar valores linguísticos. Essas características são bastante importantes para esta pesquisa, pois pretendemos aumentar a compreensibilidade da saída de um classificador Bayesiano.

Os principais conceitos que serão utilizados na pesquisa são: fuzzificação, um processo que transforma valores clássicos (e.g. inteiros, reais) em valores *fuzzy*; regras *fuzzy*, que é a forma que acreditamos ser o melhor no momento para explicar as saídas de um classificador Bayesiano; e Sistema de Classificação *Fuzzy* (SCF), que servirá como o classificador principal desta pesquisa. Nas seções seguintes, será dada uma visão geral sobre cada um destes conceitos.

3.1. Fuzzificação

O processo conhecido como fuzzificação possui o propósito de transformar valores *crisp*, que são valores clássicos e que normalmente pertecem ao conjunto dos números reais, em valores *fuzzy*, valores que são definidos por conjuntos fuzzy. Para tanto, primeiro deve-se determinar quais conjuntos *fuzzy* representarão o domínio da variável. Existem alguns métodos para achar os conjuntos *fuzzy* que melhor representam o domínio da variável, por exemplo o *fuzzy C-Means*[17], ou simplesmente obter os conjuntos *fuzzy* através de testes empíricos para determinar qual é a melhor forma de cada conjunto *fuzzy* e a quantidade de conjuntos *fuzzy* que deverão ser utilizados para representar a variável.

Para fuzzificar um certo valor *crisp*, é preciso calcular todos os graus de pertinência deste valor em todos os conjuntos *fuzzy* daquele domínio. Mostraremos abaixo um exemplo de fuzzificação:

Considere uma variável contínua que possui os limites entre 0 e 10. Considere também que os conjuntos *fuzzy* que serão utilizados para definir esta variável sejam :

$$\mu_{BAIXO}(x) = \begin{cases} 0, \text{ se } x < 0 \text{ ou } x >= 4 \\ (4 - x)/4, \text{ se } x >= 0 \text{ e } x < 4 \end{cases}$$

$$\mu_{MEDIO}(x) = \begin{cases} 0, \text{ se } x <= 1 \text{ ou } x >= 9 \\ (9 - x)/4, \text{ se } x >= 5 \text{ e } x < 9 \\ (x - 1)/4, \text{ se } x < 5 \text{ e } x > 1 \end{cases}$$

$$\mu_{ALTO}(x) = \begin{cases} 0, \text{ se } x <= 6 \text{ ou } x > 10 \\ (x - 6)/4, \text{ se } x > 6 \text{ e } x <= 10 \end{cases}$$

Então, aplicando o processo de fuzzificação em um valor *crisp* q = 6,5 desta variável teremos o seguinte resultado :

$\mu_{BAIXO}(q) = 0$;
$\mu_{MÉDIO}(q) = (9 – 6,5)/4 = 2,5/4 = 0,625$;
$\mu_{ALTO}(q) = (6,5 – 6)/4 = 0,5/4 = 0,125$;

Então podemos dizer que o valor q pertence ao conjunto *fuzzy* MÉDIO com 0,625 de grau de pertinência e pertence também ao conjunto *fuzzy* ALTO com 0,125 de grau de pertinência. Note que isso não quer dizer que o valor *fuzzy* do valor q seja o conjunto MÉDIO, mas sim que ele é considerado mais MÉDIO do que ALTO.

3.2. Classificação *fuzzy*

Um sistema de classificação *fuzzy* utiliza as regras *fuzzy* como uma base de regras. O sistema então utiliza a base de regras para classificar certa instância. Dois tipos de raciocínios podem ser aplicados para obter a classe que a instância pertence: o Raciocínio Clássico e o Raciocínio Geral. A seguir mostraremos mais detalhes sobre os cálculos envolvidos no processo de classificação e os dois tipos de raciocínios.

3.2.1. Regras *fuzzy*

As regras *fuzzy* são regras no formato “Se <antecedente> Então <consequente>”, e que possuem alguns ou todos os antecedentes e/ou alguns ou todos os consequentes definidos pelas variáveis *fuzzy*. Para este trabalho, focamos nas regras *fuzzy* que são utilizadas nos problemas de classificação, ou seja, são regras que possuem alguns ou todos os seus antecedentes sendo variáveis linguísticas que representam um conjunto *fuzzy* . Na parte de consequente está presente o valor da variável classe, o qual pode ser associado a um conjunto *fuzzy*. Cada regra *fuzzy* pode ter um grau de certeza associado indicando o peso da regra na classificação. A seguir apresentaremos um exemplo de uma regra no formato utilizado neste trabalho.

se A é alto e B é baixo então Classe é c1 com 0,8 de certeza

Neste trabalho, se a classe se associa a um conjunto *fuzzy*, ela se transforma em valor nominal representado pelo valor linguístico do conjunto fuzzy que a classe possui o maior grau de pertinência.

3.2.2. Raciocínio *fuzzy* clássico e geral

Os sistemas de classificação *fuzzy* que utilizaremos neste trabalho são baseados em regras *fuzzy*, ou seja, utilizam uma base de regras no processo de classificação. Existem dois tipos de raciocínio que podemos utilizar na hora de realizar os cálculos para classificação: o raciocínio *fuzzy* clássico e o raciocínio *fuzzy* geral. Os cálculos que são feitos utilizando o raciocínio *fuzzy* clássico podem ser descritos pelo pseudocódigo mostrado no **Algoritmo 8**.

```
Procedure Raciocínio Fuzzy Clássico;
input:  R1...Rn: Uma base de regras com n regras, cada regra com um grau de
        certeza  (GC) associado;
        X2...Xm: As variáveis do problema;
        X1: Variável classe;
        e2...em: As evidências fornecidas;
output: Vc: O valor da classe mais provável;
init:   CP2...CPm := void;
        C := void;
        H1...Hn := void;
        Vc := void;
begin
    for i := 1 to n do
    begin
        for j := 2 to m do
            if Xj de Ri não é conjunto fuzzy then
                if ej = o valor de Xj de Ri then
                    CPj := 1;
                else
                    CPj := 0;
            else
                CPj := Grau de pertinência do ej no conjunto fuzzy Xj de Ri;
        C := t_norma(CP2,...,CPm);
        Hi := C * GCi; /*Calcular o grau de associação*/
    end;{for}
    Rk := A regra com maior grau de associação;
    Vc := O valor de X1 definido na Rk;
    return Vc;
end.{RaciocinioFuzzyClassico}
```

Algoritmo 8. Raciocínio fuzzy clássico.

Como pode ser visto no pseudo código dado no **Algoritmo 8**, o raciocínio *fuzzy* clássico se baseia na idéia da regra vencedora, porque ele considera apenas o resultado da regra que possui o maior grau de associação e ignora os possíveis conhecimentos que podem existir nas outras regras. Ou seja, se existe uma quantidade alta de regras que indicam o padrão sendo de certa classe, porém nenhuma delas é vencedora. Este raciocínio dará a prioridade para a regra vencedora ignorando o resultado de todas as outras regras. Contrário do raciocínio

fuzzy geral, que pode ser descrito pelo pseudocódigo **Algoritmo 9**.

```
Procedure Raciocínio Fuzzy Geral;
input:  R1...Rn: Uma base de regras com n regras, cada regra com um grau de
        certeza (GC) associado;
        X2...Xm: As variáveis do problema;
        X1: Variável classe;
        e2...em: As evidências fornecidas;
output: Vc: O valor da classe mais provável;
init:   CP2...CPm := void;
        C := void;
        H1...Hn := void;
        k := void;
        p := Quantidade de possíveis valores que X1 pode assumir;
        Y1...Yp := 0;
begin
    for i := 1 to n do
    begin
        for j := 2 to m do
            if Xj de Ri não é conjunto fuzzy then
                if ej = o valor de Xj de Ri then
                    CPj := 1;
                else
                    CPj := 0;
            else
                CPj := Grau de pertinência do ej no conjunto fuzzy Xj de Ri;
        C := t_norma(CP2,...,CPm);
        Hi := C * GCi; /*Calcular o grau de associação*/
        k := Índice que representa o valor de X1 definido no Ri;
        /*Y é o grau de classificação, é um agrupamento dos graus de
        associação pelo valor da classe e f é um operador de associação*/
        Y k := f(Yk + Hi);
    end;{for}
    Yk := A classe com maior grau de classificação;
    Vc := O valor da classe associada ao Yk;
    return Vc;
end.{RaciocinioFuzzyGeral}
```

Algoritmo 9. Raciocínio fuzzy geral.

O raciocínio *fuzzy* geral valoriza os resultados de todas as regras e não somente da regra vencedora. Com esta característica, o raciocínio *fuzzy* geral pode extrair mais conhecimentos de todas as regras no processo de classificação. Neste trabalho estamos utilizando o raciocínio *fuzzy* geral no sistema de classificação *fuzzy*, como t-norma, o mínimo e o operador de associação como média aritmética.

4. TRABALHOS RELACIONADOS

Neste capítulo apresentamos alguns trabalhos que também buscam melhorar a compreensão de uma rede Bayesiana e trabalhos relacionados a seleção de atributos em redes Bayesianas. Os trabalhos abordados neste capítulo são o método *BayesFuzzy* [5], que trabalha com regras *fuzzy* e com a explicação preditiva que é a base do trabalho deste mestrado; a abordagem *Ghim-Eng* [8], que trabalha com regras de classificação e explicação diagnóstica e possui um objetivo semelhante ao este trabalho, mas com uma abordagem diferente; abordagem *Extended Markov Blanket* [23], que se baseia no *Markov Blanket* para desenvolver um método de seleção de atributo mais eficiente. É um trabalho que nos auxilía a ter uma nova visão sobre a seleção de atributos que utilizamos neste trabalho; e também a abordagem *JCFO* [24], que busca ao mesmo tempo, construir um classificador Bayesiano e também selecionar os atributos mais relevantes.

Tendo estes trabalhos citados acima, podemos comparar qual dos métodos está mais próximo do nosso objetivo e quais pontos ainda não foram trabalhados por eles. Será mostrado também um método clássico (C4.5) que utiliza árvores de decisão para tratar um problema e expressar os resultados em forma de regras, que além de ser semelhante com o nosso objetivo, também servirá como uma base de comparação. Nas seções seguintes será mostrado um pouco sobre cada um deles.

4.1. *BayesFuzzy*

Tradicionalmente, redes Bayesianas são aprendidas a partir de dados discretos e não possuem a capacidade de tratar dados contínuos[1]. Portanto, os autores de *BayesFuzzy* [5] propuseram o uso do processo de fuzzificação para transformar os dados contínuos em dados discretos (processo de discretização). Desta forma, após o processo de fuzzificação, é possível aprender uma rede Bayesiana sobre qualquer conjunto de dados. Por isso, esta fuzzificação, para discretizar as variáveis, é o primeiro passo do método *BayesFuzzy*. Em seguida, é utilizado um algoritmo de aprendizagem de redes Bayesianas para construir um classificador Bayesiano baseado nos dados fuzzificados.

Em [5], foi utilizado o algoritmo K2 como algoritmo de aprendizagem. Após a rede ser aprendida, as regras são extraídas desta rede de acordo com o pseudocódigo dado no **Algoritmo 10**.

[1] É possível se trabalhar com redes Bayesianas que manipulam dados contínuos, mas tais redes são normalmente chamadas “redes Gaussianas” [27]. A maior dificuldade em se utilizar tais modelos probabilísticos é a necessidade de se conhecer a distribuição de probabilidade de todas as variáveis do problema. Por este motivo, a discretização acaba sendo um método mais adequado, na maioria das vezes[41].

As regras extraídas da rede são então utilizadas como uma base de regras de um SCF. Portanto, podemos sumarizar o *BayesFuzzy* nos 4 passos seguintes:

1) "Discretizar" o conjunto de dados original D usando o processo de fuzzificação, gerando o conjunto de dados qualitativo D';
2) Aprender um Classificador Bayesiano (CB) usando D';
3) Extrair regras do CB gerado no passo 2 usando o **Algoritmo 10**;
4) Utilizar as regras geradas no passo 3 como uma base de regras para um SCF;

Os resultados apresentados pelo *BayesFuzzy* mostram que esta abordagem é adequada para a tarefa de explicação de uma rede Bayesiana, ou seja, consegue transformar os conhecimentos contidos numa rede Bayesiana numa forma que a maioria das pessoas possam entender estes conhecimentos com maior facilidade. Em [5], os autores comparam os resultados obtidos pelo *BayesFuzzy* com o método de geração automática de regras *fuzzy* de Wang e Mendel [39]. Em muitos casos, o *BayesFuzzy* produziu menos regras e com uma taxa de classificação próxima à taxa de classificação obtida pelas regras fuzzy geradas pelo método de Wang e Mendel. O *BayesFuzzy* gera explicações do tipo preditiva. A **Figura 11** mostra o esquema geral do *BayesFuzzy*.

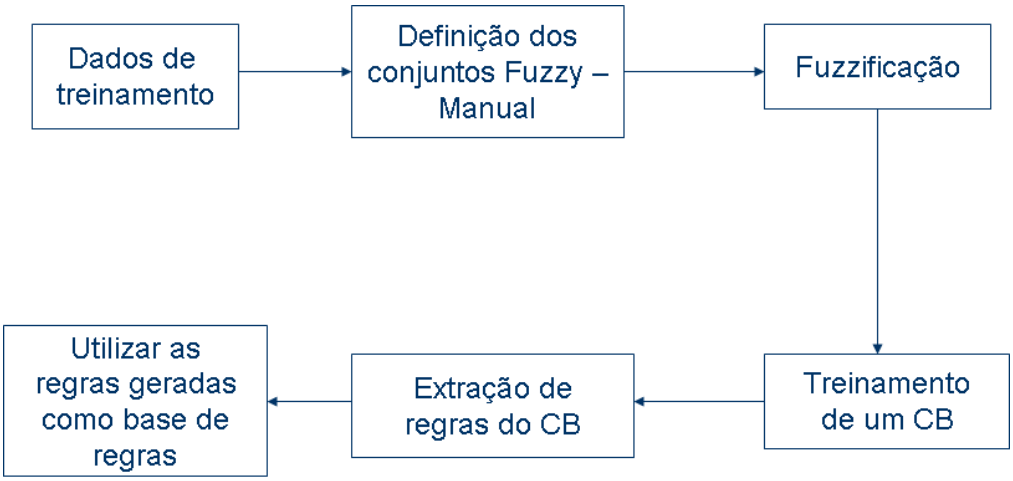

Figura 11 Esquema geral do método *BayesFuzzy*.

```
procedure BC_Extrair_Regras;
input:   R: Rede bayesiana com m variáveis;
         X1: Variáveis da classe;
output:  CRR: Conjunto de regras reduzidas;
init:    CRR := {};
         MBC := void;
         n := void;
         V2...Vn := void;
         ji...jn := void;
begin
    MBC := MB(R, X1); /*Markov Blanket do nodo X1*/
    n := |Nodos de MBC|;
    for i := 2 to n do
        Vi := todos os possíveis valores que o Xi pode assumir;
    for i := 2 to n do
        ji := |Vi|;
    RI := 1; /*Índice da regra*/
    for k2 := 1 to j2 do
        for k3 := 1 to j3 do
            …
            for kn := 1 to jn do
            begin
                Forma antecedentes das regras como X2 := V2k2 and X3 := V3k3
                and ... and Xn := Vnkn;
                Propaga os antecedentes acima sobre a MBC obtendo Vc como
                resultado da classificação;
                Define RRI como X2 é V2k2 e X3 é V3k3 e ... e Xn é Vnkn então X1 é
                Vc;
                CRR := CRR ∪ RRI;
                RI := RI + 1;
            end;{for}
    CRR := remover_regras_superfluas(CRR);
    return CRR;
```

Algoritmo 10. Extrair regras de um BC (Parte do *BayesFuzzy*).

4.2. Abordagem *Ghim-Eng*

Existem diversos métodos que foram propostos para tentar explicar uma rede Bayesiana de forma mais intuitiva, porém poucos ou nenhum deles conseguem explicar a forma com que a rede Bayesiana trata a falta de informações ou informações errôneas, tais como erro na entrada dos dados, assim como alguns trabalhos não expressam a interação entre os pais do nodo da classe [8].

Em [8] é proposta a utilização de regras extraídas da rede Bayesiana para explicar a decisão da rede. Esta explicação não é apenas referente às relações de cada valor de entrada com o valor da classe, mas também é referente às relações do conjunto de entradas com o valor da classe e explica como os erros ou a falta de informações da rede é compensada. Para não estender o problema mais do que o necessário, os autores limitam esta explicação para o *Markov Blanket* do nodo da classe, porque o *Markov Blanket* do nodo já é suficiente para explicar totalmente o resultado da inferência referente ao nodo da classe.

Estas regras são extraídas a partir de árvores de decisão construídas para cada possível valor da classe e associam o grau de certeza das regras geradas com a probabilidade condicional calculada utilizando uma instância que corresponde à regra em questão, ou seja, a parte antecedente de cada regra gerada é a parte das evidências utilizadas na rede Bayesiana para a geração das regras. Já a parte consequente da regra, é o valor da classe. Portanto, utilizando estas evidências podemos calcular a probabilidade condicional $P(x|e)$ onde x representa o valor da classe (que corresponde à parte consequente da regra) e e representa as evidências utilizadas na rede para a extração da regra. Assim, pode-se utilizar o $P(x|e)$ como o grau de certeza da regra gerada.

Na literatura técnica, existem alguns métodos que utilizam a representação gráfica para explicar a rede Bayesiana. Fora a fácil compreensão da representação gráfica, estes métodos muitas vezes não conseguem examinar a distribuição de uma única variável e ainda é difícil compreender todo o processo de inferência.

Chajewska e Draper [3] propõem a procura por uma solução boa do problema em vez de se buscar a melhor solução. Com isso, é possível diminuir a complexidade da busca. Além disso, definem uma medida de qualidade da explicação. Contudo, ainda existe a dificuldade de examinar se possui uma explicação na rede que consiga atingir um certo grau de qualidade requerida, porque para isso seria necessário examinar todas as possibilidades novamente.

Em [8], são apontados 2 dos maiores problemas causadores de falha nos métodos que existem na literatura técnica:

1. Alta quantidade de variáveis, que pode chegar até milhares de variáveis;
2. Alto custo computacional para analisar todas as possíveis evidências. Muitos dos métodos se limitam apenas à análise de uma única variável e não à interação entre as variáveis;

Portanto, podemos organizar alguns pontos importantes no problema de explicação da rede Bayesiana :

1. Explicar a importância de cada variável em relação à classe;
2. Explicar a interação entre as variáveis;
3. Considerar que pode existir uma grande quantidade de possíveis evidências;
4. Considerar o custo computacional;
5. Medida da qualidade da explicação.

O método que *Ghim-Eng* et al. propuseram pode ser descrito pelo pseudocódigo dado no Algoritmo 11.

Procedure *Ghim-Eng*;
input: R : Uma rede Bayesiana treinada com n variáveis;
X_1: Variável classe;
$V_1..V_m$: Todos os possíveis valores da variável classe;
$e_2...e_n$: As evidências oferecidas pelo usuário;
d: Valor booleano indicando se o usuário deseja explicação sobre valores faltantes;
output: Ex: Uma explicação textual com algumas regras dentro;
init: MBC := {};
$AD_1...AD_m$:= **void**;
R := {};
Ex := **void**;
beginc
MBC := MB(R, X_1); /**Markov Blanket* do atributo classe*/
MBC := *Arc_reversal*(MBC, X_1); /*Aplicar *arc reversal* no atributo classe*/
for i := 1 **to** m **do**
/*Gerar uma árvore de decisão para cada possível valor da classe*/
AD_i := gerar_arvore(MBC, V_i);
/*Extrair regras das árvores geradas de acordo com as evidências oferecidas*/
R := extrair_regras(AD, $e_2...e_n$);
Ex := gerar_explicação(R);
if d **then**
/*Incorporar a explicação textual sobre como BC trata valores faltantes na explicação final*/
Ex := Ex + gerar_exp_val_faltante(MBC);
return Ex;

Algoritmo 11. Abordagem de *Ghim-Eng.*

No método descrito pelo Algoritmo 11, a geração das explicações é on-line, ou seja, será gerada uma explicação específica para cada entrada que o usuário fornece, portanto é uma explicação do tipo diagnóstica.

4.3. Abordagem *Extended Markov Blanket*

Análises proteômicas são excelentes para a identificação de algumas doenças no estágio prematuro e são simples, baratas e minimamente invasivas.

Entre as tecnologias de arquivação do soro, a *low-resolution surface-enhanced laser desorption/ionization time-of-flight* (*SELDI-TOF*) e a espectrometria de massa (*MS*) são utilizadas para detectar padrões das proteínas. Porém, as proteínas num, escopo menor, contém informações abundantes sobre a classificação das proteínas, mas para visualizaá-las num escopo menor, é necessário a análise de espectrometria de massa com alta resolução.

Diversos algoritmos foram desenvolvidos para *SELDI-TOF*, mas para *MS* de alta resolução, os algoritmos tradicionais de aprendizagem da máquina não possuem capacidade para analisar uma entrada com tão alta dimensionalidade. Portanto, há uma necessidade de redução da dimensionalidade para os problemas *MS* de alta resolução.

Baseado nestas necessidades, esta abordagem desenvolve um algoritmo de seleção de atributos multi-passo chamado *Extended Markov Blanket* (*EMB*). Este método, busca dois subconjuntos de atributos que são relevantes para cada atributo: são os subconjuntos de atributos baixo-correlacionado (*LCFS*) e alto-correlacionado (*HCFS*). O subconjunto alto-correlacionado é a definição do *Markov Blanket* obtido pelo cálculo dos coeficientes de correlação de *Pearson* e contribui para a remoção dos atributos redundantes. E, por outro lado, o subconjunto baixo-correlacionado é utilizado para atribuir pesos para cada atributo em relação a influência deles no resultado da classificação.

A contribuição ou novidade deste algoritmo, está justamente no uso do subconjunto *LCFS* para atribuir pesos para cada atributo. Os pesos dos atributos são obtidos com base na função de vetor de peso do *Support Vector Machine* (*SVM*) linear que utiliza uma pequena porção do conjunto de dados para servir como dados de treinamento e teste para *SVM* linear. E como o algoritmo de *SVM* linear é treinado apenas com uma pequena porção dos dados de treinamento, seu desempenho pode ser bastante limitado. Assim, tanto o peso absoluto do *SVM* linear quanto a precisão são fatores importantes para determinar o peso final do atributo. Então, quando a

precisão é maior do que 0.5, será tratado na função de cálculo de peso como sendo um peso positivo. Caso contrário, será considerado uma penalidade que gera um peso negativo.

Depois de determinar os pesos dos atributos, é removido o atributo com menor coeficiente de correlação de *Pearson* de acordo com as regras de *Markov Blanket*. E então atualizam-se os subconjuntos *HCFS* e *LCFS* e calculam-se os pesos novamente. Este processo será repetido até que sobre apenas um atributo ou então até atingir um número de atributos pré-determinado.

No passo seguinte, será aplicado um *Backward Feature Selection* para selecionar o melhor atributo. O Algoritmo 12 descreve o algoritmo *Extented Markov Blanket* e o Algoritmo 13 descreve o algoritmo *Backward Feature Selection*.

$$z_k = \begin{cases} \dfrac{|w_k|}{\sum_{j=1}^{d+1} |w_j|}\,\beta & \text{se } \beta > 0{,}5 \\ -\left(1 - \dfrac{|w_k|}{\sum_{j=1}^{d+1} |w_j|}\right)(0{,}5 - \beta) & \text{caso contrário} \end{cases} \qquad (9)$$

Legenda da equação (9):

z_k = Peso final do k-ésimo atributo;

w_k = Peso do k-ésimo atributo calculado por um SVM linear;

d = A quantidade de atributos presente no LCFS;

β = A taxa de classificação obtida pelo k-ésimo atributo dentro do SVM linear.

$$\Delta(F_i|M_i) = \sum_{j=1}^{n} P_j(M_i, F_i) \sum_{l=1}^{k} P_j(c_l|M_i, F_i) \log\left(\frac{P_j(c_l|M_i, F_i)}{P_j(c_l|M_i)}\right) \qquad (10)$$

Legenda da equação (10):

$\Delta(F_i|M_i)$ = Entropia cruzada do nodo F_i dado o *Markov Blanket* M_i do F_i;

n = Quantidade de amostras;

k = Quantidade de possíveis valores do atributo classe;

c_l = i-ésimo possível valor do atributo classe.

Procedure Extended_Markov_Blanket;
input: $A_1...A_n$: Um conjunto de atributos;
A_q: Atributo classe;
max: Quantidade desejada de atributos que devem restar no LCFS;
d: Quantidade de nodos que inicialmente pertencem a LCFS e HCFS;
T: Um conjunto de dados de treinamento;
output: Resultado: Um conjunto de atributos selecionados;
init: HCFS := d atributos mais correlacionados com o A_q de acordo com o cálculo do coeficiente correlacionado de *Pearson*;
LCFS := HCFS;
Removidos := {};
$w_1...w_d$:= **void**; /*Os pesos calculados pelo SVM*/
$z_1...z_d$:= **void**; /*Os pesos finais após o cálculo da equação (9)*/
$e_1...e_d$:= **void**; /*As entropias cruzadas de cada atributo*/
$c_1...c_d$:= 0; /*Os pesos finais acumulados*/
begin
while |HCFS| > max;**do**
begin
for cada $A_k \in$ HCFS **and** $A_k \notin$ Removidos **do**
begin
e_k := Resultado da equação (10) onde F_i é o A_k;
z_k := Resultado da equação (9);
w_k := O peso do atributo A_k que um algoritmo de SVM linear calculou com os dados de treinamento T;
$c_k := c_k + z_k$;
end; {for}
Remove o atributo com a menor entropia cruzada do HCFS de acordo com as regras de *Markov Blanket*;
LCFS := HCFS;
Removidos := Removidos $\cup$ {Atributo com a menor entropia cruzada };
end; {while}
for i := 1 **to** d **do**
$c_i := c_i$ / |Removidos|;
return BFS($A_1...A_d$, $c_1...c_d$, T);
end. {Extended_Markov_Blanket}

Algoritmo 12. Algoritmo *Extented Markov Blanket.*

```
Procedure BFS;
input:  A1...An: Um conjunto de atributos;
        c1...cn: Os pesos relacionados aos A1...An;
        T: Um conjunto de dados de treinamento;
output: Resultado: Um conjunto de atributos selecionados;
init:   r1...r60 := 0;
        k := 1;
        Resultado := {A1,...,An};
begin
    Ordena os A1...An de acordo com os seus pesos c;
    for i := 1 to 60 do
    begin
        ri := Taxa de classificação de um SVM linear treinado em T utilizando os
              primeiros i atributos na ordenação;

        if ri ≥ rk then

            k := i;
    end; {for}
    while |Resultado| > 1 do
    begin
        Remove do resultado o atributo que após ser removido causa maior
            queda no erro ou que só aumenta o erro por um pouco;
    end; {while}
    return Resultado;
end. {BFS}
```

Algoritmo 13. Algoritmo *Backward Feature Selection.*

4.4. Abordagem *JCFO*

O objetivo dos problemas de aprendizado supervisionado binário é aprender como distinguir os exemplos em duas classes. Nesta abordagem, os autores desenvolveram o algoritmo *JCFO* para construir um classificador e também fazer a seleção de atributos nos problemas de aprendizado supervisionado binário.

Uma das idéias fundamentais que suportam *JCFO* é de um fator não negativo relacionado com cada um dos atributos, θ_k. Estes fatores são estimados a partir dos dados, com prioridade sobre os valores que são significantemente grandes ou exatamente zero. Os autores deste trabalho focam em *probabilistic kernel classifier* da forma da equação (11), buscando esparsidade nas funções bases e esparsidade

nos atributos.

No contexto do trabalho, a similaridade entre os registros depende dos fatores que se associam com cada um dos atributos, θ_k. Se θ_k possui um valor zero, então o atributo associado é removido do classificador, como podemos observar nas equações (12) e (13), servindo como a seleção de atributo.

Para encorajar a esparcidade nos fatores escalares da função base e de atributo (β e θ respectivamente), os autores adotaram *Laplacian prior* para ambos os casos. Porém, *Laplacian prior* demanda um recurso computacional grande, tornando a aplicação inviável. Portanto, os autores utilizam uma formulação hierárquica alternativa que corresponde a *Laplacian prior* e que demanda menos recurso computacional.

O objetivo do algoritmo é estimar o Máximo a Posteriori (MAP) dos fatores β e θ. Para este fim, o trabalho adota um algoritmo de Expectativa-Maximização (EM).

O algoritmo completo do JCFO pode ser descrito pelo Algoritmo 14. Caso deseje realizar uma seleção de atributos, basta configurar todos os θ_k por uma constante igual para todos e a definição de $h_\theta(x)$ (que pode ser encontrada na legenda da equação (15)) passa a ser $h_\theta(x) = [1, x_1, x_2, ..., x_p]^T$, onde x_i é o valor da i-ésima variável da instância x.

Os primeiros testes empíricos são realizados em dados de expressão gênica. Estes dados são conhecidos por terem muitas variáveis irrelevantes. Os resultados mostram que a seleção de atributos realizada pelo *JCFO* ajudou bastante a identificar as variáveis que são realmente importantes e que já são bastante referenciadas em domínio de saúde e em relação a classificação correta, *JCFO* mostra ser melhor do que *Adaboost*, *SVM*, *RVM*, *SPR* e *logistic regression* pelo teste binomial com um valor $p <= 0.12$ em relação a *SPR* e $p <= 0.03$ em relação aos outros.

Também realizaram testes em bases de dados de *benchmark*, onde *JCFO* teve as melhores performances ou quase melhores em todas as bases testadas, porém os testes estatísticos não mostram que são estatisticamente significativos.

Como conclusão, o *JCFO* atingiu o objetivo da pesquisa com boa performance de classificação e seleção automática dos atributos. Porém estes ganhos aumenta a complexidade do modelo e também aumenta o esforço computacional. Portanto, o *JCFO* ainda não está preparado para ser aplicado num dado de treinamento com centenas e milhares de variáveis e amostras.

$$P(y=1|x) = \Phi\left(\beta_0 + \sum_{i=1}^{n} \beta_i K_\theta(x,x^{(i)}) \right) \qquad (11)$$

Legenda da equação (11):

$\Phi(x)$ = Função Gaussiana de distribuição cumulativa, calculada pela equação (12);

β_i = Fator de escala associado a i-ésima instância dos dados de treinamento;

K((x,x(i)) = Função Kernel calculando a similaridade entre a instância de teste x e a i-ésima instância de treinamento x(i), pode ser calculada pela função (13) ou (14).

Legenda da equação (12):

$\Phi(z)$ = Probabilidade de uma variável randômica cair dentro do intervalo $(-\infty,z]$.

$$K_\theta(x,x^{(i)}) = \left(1 + \sum_{k=1}^{p} \theta_k x_k x_k^{(i)} \right)^r \qquad (13)$$

Legenda da equação (13):

x_k = O valor do k-ésimo atributo da instância de teste x;

$x_k^{(i)}$ = O valor do k-ésimo atributo da i-ésima instância de treinamento $x^{(i)}$;

θ_k = Fator de escala associado ao k-ésimo atributo;

r = Ordem polinomial utilizado;

p = Quantidade de atributos.

$$K_\theta(x,x^{(i)}) = \exp\left(- \sum_{k=1}^{p} \theta_k (x_k - x_k^{(i)})^2 \right) \qquad (14)$$

Legenda da equação (14):

x_k = O valor do k-ésimo atributo da instância de teste x;

$x_k^{(i)}$ = O valor do k-ésimo atributo da i-ésima instância de treinamento $x^{(i)}$;

θ_k = Fator de escala associado ao k-ésimo atributo;
p = Quantidade de atributos.

$$P(y=1|x,\beta,\theta) = \Phi(h_\theta^T(x)\beta) \qquad (15)$$

Legenda da equação (15):
x = Instância de teste;
$h_\theta^T(x)$ = A matriz transposta de $h_\theta(x)$. $h_\theta(x) = [1, K_\theta(x, x^{(1)}), ..., K_\theta(x, x^{(1)})]^T$.

$$\hat{\beta}_\theta = S(I + SH^T_\theta H_\theta S)^{-1} SH^T_\theta \upsilon \qquad (16)$$

Legenda da equação (16):

$\hat{\beta}_\theta$ = O conjunto de fatores escalares β ótimo correspondente ao conjunto de fatores escalares θ dado;

S = Matrix diagonal onde a diagonal principal é composta pelos elementos [s_0, ..., s_n], n é a quantidade de dados de treinamento;

$s_i = \gamma_1^{-1/2} |\hat{\beta}_i^{(t)}|^{1/2}$, onde γ_1 = parâmetro de regularização do β, obtido pela validação cruzada, $\beta_i^{(t)} = \beta$ estimado no passo t correspondente a i-ésima instância do conjunto de dados de treinamento;

$$\upsilon_i \equiv h_\theta^T(x^{(i)})\beta^{(t)} + \frac{l^{(i)} N(h_\theta^T(x^{(i)})\beta^{(t)}|0,1)}{\Phi(l^{(i)} h_\theta^T(x^{(i)})\beta^{(t)})} \qquad (17)$$

Legenda da equação (17):
υ_i = O valor esperado das variáveis faltantes da i-ésima instância $z^{(i)}$, onde z representa as variáveis faltantes da forma $[z^{(1)}, ..., z^{(n)}]^T$. $z^{(i)} = z(x^{(i)}, \beta, \theta) = h_\theta^T(x^{(i)})\beta + (, (\sim N(0,1)$;
$l(i) = 2y(i) - 1$;

!!

$$\omega_i \equiv \gamma_1 |\hat{\beta}_i^{(t)}|^{-1} \qquad (18)$$

Legenda da equação (18):

ω_i = O valor esperado das variáveis faltantes da i-ésima instância (i-1, onde (representa as variáveis faltantes da forma [(0, ..., (n]T.

$$\delta_k \equiv \gamma_2 (\theta_k^{(t)})^{-1} \qquad (19)$$

Legenda da equação (19):

δ_k = O valor esperado da k-ésima variável faltante (k-1, onde (representa as variáveis faltantes da forma [(1, ..., (k]T;

(2 = Parâmetro de regularização de (, obtido pela validação cruzada.

$$\hat{\beta}^{(t+1)}, \hat{\theta}^{(t+1)} = \arg\max_{\beta,\theta} Q(\beta,\theta|\hat{\beta}^{(t)},\hat{\theta}^{(t)}) \qquad (20)$$

Legenda da equação (20):

Q(x) = A função Q que pode ser calculada pela equação (21).

$$Q(\beta, \theta|\hat{\beta}^{(t)}, \hat{\theta}^{(t)}) = - \beta^T H_\theta{}^T H_\theta \beta + 2\beta^T H_\theta{}^T \upsilon - \beta^T \Omega \beta - \theta^T \Delta \theta \qquad (21)$$

Legenda da equação (21):

$H_\theta{}^T$ = A matriz transposta de H_θ. $H_\theta = [h_\theta(x^{(1)}), ..., h_\theta(x^{(n)})]^T$;

$\upsilon = [\upsilon_1, ..., \upsilon_n]^T$;

Ω = Matriz diagonal onde a diagonal principal é composta pelos elementos (ω_0, ... ω_n);

Δ = Matriz diagonal onde a diagonal principal é composta pelos elementos (δ_1, ..., δ_p);

Procedure JCFO;
input: D: Um conjunto de dados de treinamento com n instâncias;
$X_1...X_p$: Os atributos do problema;
X_k: O atributo classe;
$x_{t1}...x_{tp}$: Os valores dos atributos $X_1...X_p$ da instância de teste;
output: y_R: A classe mais provável da instância de teste;
init: y_R := **void**;
p_1 := **void**;
β := {};
θ := {};
υ := {};
Ω := {};
Δ := {};
begin
/*Algoritmo EM*/
while (β e θ não convergem)
begin
/*Passo E*/
/*Definição de υ, Ω e Δ pode ser encontrada na legenda da equação (21)*/
Calcula υ, Ω e Δ de acordo com as equações (17), (18) e (19) respectivamente;
/*Passo M*/
Utiliza um algoritmo de gradiente conjugado para achar o θ;
Utiliza o θ para achar o ótimo β correspondente utilizando a equação (16);
Calcula os novos β e θ de acordo com a equação (20);
end; {while}
/*Término do algoritmo EM*/
p1 := $P(X_k = 1|x_t, \beta, \theta)$; /* $P(X_k = 1|x_t, \beta, \theta)$ pode ser calculada pela equação (15)*/
if p1 $\geq$ 0,5 **then**
y_R := 1;

Algoritmo 14. Algoritmo de JCFO.

4.5. Árvore de Decisão C4.5

As árvores de decisão apresentam bons resultados para os problemas de

classificação. Uma das formas tradicionais de aprendizagem de uma árvore de decisão é utilizar o algoritmo C4.5 [19]. Este algoritmo adota uma abordagem *top-down* e uma busca do tipo *greedy* pelo espaço de possíveis árvores de decisão. A idéia geral do algoritmo é iniciar escolhendo o nodo que melhor classifica o conjunto de treinamento sozinho. A partir deste nodo, será criado uma quantidade de filhos que corresponde aos possíveis valores que este nodo pode assumir. Então para cada filho, repete-se o processo excluindo os nodos que já foram incluídos na árvore. Por fim, as folhas são as classes mais prováveis. O Algoritmo 15 apresenta um pseudocódigo do C4.5.

Procedure C4.5**;**
input: D: conjunto de dados de treinamento;
X_1: variável classe;
X_2...X_n: variáveis do problema;
output: A: árvore de decisão gerada;
init: Z := {};
M := {};
A := árvore com nenhum nodo;
X_k := **void**;
begin
Z := Z ∪ {X_2, X_3, …, X_n}**;**
while Z ≠ {} **do**
begin
X_k := a variável em Z que classifica melhor sozinho sobre D;
M := todos os possíveis valores de qualquer folha de A;
if A = {} **then**
torna X_k em raiz de A;
e**lse**
cria |M| folhas com valor X_k para cada folha de A e rotula os novos arcos como cada valor de M;
Z := {X_2, ..., X_{k-1}, X_{k+1}, ..., X_n};
end;{while}
criar uma folha para cada folha em A com valor como sendo a classe mais provável em D onde as instâncias possuem exatamente os valores iguais aqueles seguindo a raiz até a folha em questão;
return A**;**
end.{C4.5}

Algoritmo 15. Árvore de decisão C4.5.

A **Figura 12** mostra o exemplo de uma árvore de decisão C4.5 construída.

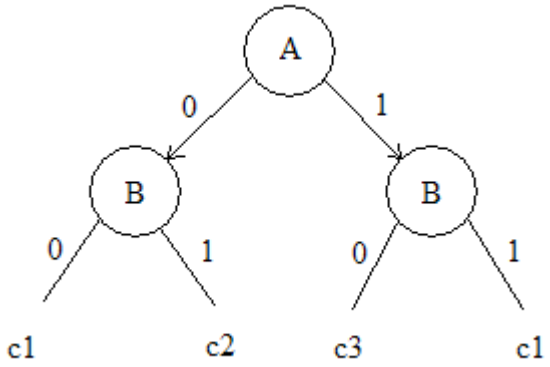

Figura 12 Exemplo de uma árvore de decisão C4.5.

Para otimizar os resultados apresentados pela árvore de decisão, as técnicas de poda podem ser aplicadas para obter melhores taxas de classificação e uma estrutura mais compacta. Uma técnica de poda bastante comum se chama *Reduced Error Pruning* [36]. A idéia principal desta poda é verificar quais nodos ao serem removidos trazem uma melhora ou mantém a mesma taxa de classificação. Então remove estes nodos e reconstrói a árvore de decisão de acordo. Na prática, existe um outro método de poda equivalente ao *Reduced Error Pruning*, que é o *Rule Post-Pruning*[36]. Esta poda trabalha com as regras e não diretamente na árvore gerada. Isso traz como benefício, menor tempo de execução, porque não há necessidade de reconstruir a árvore para cada nodo retirado, podendo esta poda ser aplicada para qualquer conjunto de regras não se limitando nas árvores de decisão. A *Rule Post-Pruning* pode ser descrita pelo pseudocódigo dado no Algoritmo 16:

```
Procedure Rule Post Pruning;
input:   R: um conjunto de n regras;
         D: um conjunto de dados de "treinamento"
output:  R': R podada;
init:    R' := {};
         Continua := void;
         X := void;
begin
    for i := 1 to n do
    begin
        Continua := true;
        while Continua do
        begin
            X := o antecedente de Ri que após ser retirado de Ri, a taxa de
            classificação da Ri em D mantém ou aumenta;
            if X = void then
                Continua := false;
            else
                remove X da Ri;
        end;{while}
        R' := R' ∪ Ri;
    end;{for}
    return R';
end.{Rule Post-Pruning}
```

Algoritmo 16. *Rule Post-Pruning.*

É possível aplicar esta poda em qualquer conjunto de regras, o que a torna mais interessante para o nosso trabalho de mestrado. Veremos a seguir um exemplo desta poda aplicada em uma única regra. Considere que temos a seguinte regra para ser podada:

se A é alto e B é baixo e C é baixo então Classe é c1 com 0,8 de certeza

considere também que, ao removermos a variável A, a regra cai na sua taxa de classificação, a remoção de B trás um ganho na taxa de classificação e a remoção de C não trás nenhuma diferença na taxa de classificação. Então, a regra podada será:

se A é alto então Classe é c1 com 0.8 de certeza

porque a remoção de B é benéfica para a regra em relação à taxa de classificação e a remoção de C, embora não traga ganhos na taxa de classificação, torna a regra mais curta, logo, será removida também. Na Seção 5.4, vamos apresentar alguns resultados comparando o *Pruned BayesFuzzy* com o método J48 [19] que é um método utilizado para a geração da árvore C4.5.

5. ALGORITMO *PRUNED BAYESFUZZY*

Com o objetivo de descobrir uma forma de explicar uma rede Bayesiana, de uma forma mais intuitiva para os seres humanos, a utilização da lógica *fuzzy* parece ser uma boa direção. Partindo desta suposição, decidimos dar sequência ao trabalho iniciado e descrito em [5], que é o método *BayesFuzzy*, o qual possuiu o mesmo objetivo e preocupação deste trabalho de mestrado. Neste trabalho desevolvemos uma estenção do *Bayes Fuzzy*: o *Pruned BayesFuzzy*.

A definição completa do algoritmo *Pruned BayesFuzzy* foi gradativa de acordo com os problemas que identificamos em cada experimento empírico que executamos. Então, para explicar a presença de cada novo elemento do *Pruned BayesFuzzy*, incluiremos os resultados empíricos obtidos em cada fase de *Pruned BayesFuzzy* neste capítulo.

Os dados utilizados para obter os resultados empíricos são dados sintéticos produzidos com o propósito de ter um controle maior sobre as características dos dados. Com este propósito, e também para garantir a aleatoriedade que deverá existir nos dados, foi desenvolvido um gerador de dados que considera apenas um único atributo classe *crisp*. Este gerador pode ser descrito pelo pseudocódigo dado no Algoritmo 17.

Em todos os testes realizados, os atributos que possuem valores contínuos são agrupados de acordo com a definição de três conjuntos *fuzzy*, sendo que cada um possui um formato triangular e são distribuídos uniformemente sobre o domínio.

A partir da Seção 5.2, teremos os resultados que obtivemos ao introduzir os novos elementos no método *BayesFuzzy*. Na Seção 5.1, apresentaremos um exemplo de aplicação deste gerador de dados.Também utilizamos a validação cruzada de 10 pastas (*10-fold cross-validation)*, ou seja, os resultados apresentados são médias dos resultados obtidos nas 10 pastas.

Procedure Gerador_dados;
input: R: um conjunto de n regras, onde cada regra possui porcentagem associada para cada possível valor que a classe pode assumir;
p: quantidade de instâncias que serão geradas;
$X_1...X_m$: variáveis do problema;
output: D: um conjunto de dados gerados;
init: D := {};
Inst := **void**;
q := **void**;
begin
for i := 1 **to** p **do**
begin
Inst := uma instância do problema;
q = i % m;
for j := 1 **to** n **do**
if X_j está definido em R_q **then**
if X_j não é fuzzy **then**
X_j de Inst assume o valor de X_j de R_q;
else
gera um valor aleatório para X_j de Inst em que este valor possui um grau de pertinência maior ou igual a 0.8 no conjunto de X_j definido em R_q
else
gera um valor aleatório para X_j de Inst;
gera o valor para o atributo classe de Inst de acordo com as porcentagens de R_q;
D := D ∪ Inst.;
end;{for}
return D;

Algoritmo 17. Geração de dados a partir de um conjunto de regras.

5.1. Exemplo de aplicação do gerador de dados

Considere a seguinte definição do problema :

1. 3 variáveis, *a1*, *a*2 e *a3*. Sendo *a3*, a classe;
2. *a1* é uma variável nominal que pode assumir valores *v1*, *v2*, *v3* e *v4*;
3. *a*2 é uma variável contínua que pode assumir valores de 0 a 2. Ela

pode ser representada por 3 conjuntos *fuzzy* triangulares, f_{baixo} (0) = 1, f_{baixo} (1) = 0, f_{baixo} (2) = 0; $f_{médio}$ (0) = 0, $f_{médio}$ (1) = 1, $f_{médio}$ (2) = 2 e f_{alto} (0) = 0, f_{alto} (1) = 0, f_{alto} (2) = 1;

4. *a3* é a classe que pode assumir os valores *c1*, *c*2 e *c3*;

Considere também as seguintes regras como sendo as regras que servirão para definir os padrões dos dados :

1. Regra 0 : Se *a1* é *v1* e *a*2 é *médio* então *a3* é (*c1* = 1; *c*2 = 0; *c3* = 0);
2. Regra 1 : Se *a1* é *v*2 então *a3* é (*c1* = 0,2; *c*2 = 0,8, *c3* = 0);
3. Regra 2 : Se *a*2 é *alto* então *a3* é (*c1* = 0,1; *c*2 = 0,2; *c3* = 0.7);

Neste caso, o n_R é 3, porque possui 3 regras e supondo também que o *n* seja 4, ou seja, serão gerados 4 instâncias a partir destas regras.

Iniciando a iteração *i* = 0 :

- Gera os dados de acordo com a regra $i\%n_R$, 0%3 = 0. Ou seja, a regra 0, Se *a1* é *v1* e *a*2 é *médio* então *a3* é (*c1* = 1*; c*2 = 0*; c3* = 0);
- Considere a primeira variável do problema. *a1* é uma variável nominal, ou seja, não *fuzzy* e é definida na regra 0. Portanto, o valor de *a1* será *v1* na nova instância gerada;
- Considere a segunda variável do problema. *a*2 é uma variável contínua, *fuzzy* e é definida na regra 0. Portanto, seu valor será um número aleatório que esteja entre os intervalo dos valores que possuem o grau de pertinência no conjunto *médio*, maiores ou iguais a 0,8. Assume que *x* é o valor que possui grau de pertinência igual a 0,8 no conjunto *médio* e, portanto, pela igualdade triangular: (*x*-0)/(0,8-0) = (1-0)/(1-0) e (2-*x*)/(0,8-0) = (2-1)/(1-0), o x = 0,8 e 1,2. Será gerado então, um número aleatório que esteja entre 0,8 e 1,2. Suponhamos que o valor aleatório seja 0,92, então o valor de *a*2 da nova instância gerada será 0,92;
- Considere a última variável do problema. *a3* é a variável da classe e é definida na regra 0. O valor de *a3* da nova instância gerada será *c1*, porque a probabilidade do *c1* é 1,0, ou seja, 100% de ocorrência em todas as gerações utilizando a regra 0. Portanto, nunca a variável *a3* de uma instância gerada a partir da regra 0 assumirá o valor de *c*2 ou *c3*;
- A nova instância gerada possuirá os seguintes valores : *a1* = *v1*, *a*2 = 0,92 e *a3* = *c1*;

Iniciando a iteração *i* = 1 :

- Gere os dados de acordo com a regra $i\%n_R$, 1%3 = 1. Ou seja, a regra 1: se *a1* é *v*2 então *a3* é (*c1* = 0,2; *c*2 = 0,8*, c3* = 0*)*;

- Considere a primeira variável do problema. *a1* é uma variável nominal, ou seja, não *fuzzy* e é definida na regra 1. Portanto o valor de *a1* será *v2* na nova instância gerada;
- Considere a segunda variável do problema. *a*2 é uma variável contínua, *fuzzy* e não é definida na regra 1. Portanto, seu valor será um número aleatório que esteja entre os intervalo dos valores que a variável *a*2 pode assumir, isto é, de 0 a 2. Suponhamos que o valor aleatório seja 1,57, então o valor de *a*2 da nova instância gerada será 1,57;
- Considere a última variável do problema. *a3* é a variável da classe e é definida na regra 1. O valor de *a3* da nova instância gerada será *c1* ou *c*2, porque a probabilidade do *c1* é 0,2, ou seja, 20% de ocorrência em todas as gerações, onde *c*2 é 0,8. Portanto, 80% de ocorrência em todas as gerações e não será *c3*, pois este tem a probabilidade 0,0. Nunca a variável *a3* de uma instância gerada a partir da regra 1 assumirá o valor de *c3*. Suponhamos que a seleção aleatória, de acordo com a probabilidade dada, selecionou o valor *c*2;
- A nova instância gerada possuirá os seguintes valores : *a1* = *v2*, *a*2 = 1,57 e *a3* = *c*2;

Iniciando a iteração *i* = 2 :

- Gere os dados de acordo com a regra $i\%n_R$, 2%3 = 2. Ou seja, a regra 2, Se *a*2 é *alto* então *a3* é (*c1* = 0,1; *c*2 = 0,2; *c3* = 0,7*)*;
- Considere a primeira variável do problema. *a1* é uma variável nominal, ou seja, não *fuzzy* e não é definida na regra 1. Portanto, o valor de *a1* será aleatório entre os possíveis valores que a *a1* pode assumir. Suponhamos que o valor aleatório seja *v4*, então, na nova instância gerada, a *a1* possuirá o valor *v4*;
- Considere a segunda variável do problema. *a*2 é uma variável contínua *fuzzy* e é definida na regra 1. Portanto, seu valor será um número aleatório que esteja entre os intervalo dos valores que possuem grau de pertinência maiores ou iguais a 0,8 no conjunto *alto*. Ou seja, entre 1,8 e 2. Suponhamos que o valor aleatório seja 1,93. Então, o valor de *a*2 da nova instância gerada será 1,93;
- Considere a última variável do problema. *a3* é a variável da classe e é definida na regra 1. O valor de *a3* da nova instância gerada será *c1*, *c*2 ou *c3*, porque a probabilidade do *c1* é 0,1, ou seja, 10% de ocorrência em todas as gerações, onde *c*2 é 0,2. Portanto, 20% de ocorrência em todas as gerações: *c3* é 0,7. Logo, 70% de ocorrência em todas as gerações. Suponhamos que a seleção aleatória, de acordo com a

probabilidade dada, selecionou o valor *c3*;

- A nova instância gerada possuirá os seguintes valores : *a1* = *v4*, *a*2 = 1,93 e *a3* = *c3*;

Iniciando a iteração *i* = 3 :

- Gere os dados de acordo com a regra $i\%n_R$, 3%3 = 0. Ou seja, a regra 0, Se *a1* é *v1* e *a*2 é *médio*, então *a3* é (*c1* = 1*; c*2 = 0*; c3* = 0);
- Considere a primeira variável do problema. *a1* é uma variável nominal, ou seja, não *fuzzy* e é definida na regra 0. Portanto, o valor de *a1* será *v1* na nova instância gerada;
- Considere a segunda variável do problema. *a*2 é uma variável contínua, *fuzzy* e é definida na regra 0. Portanto, seu valor será um número aleatório que esteja entre os intervalos dos valores que possuem grau de pertinência maiores ou iguais a 0,8 no conjunto *alto*. Ou seja, entre 0,8 e 1,2. Suponhamos que o valor aleatório seja 1,14. e então, o valor de *a*2 da nova instância gerada será 1,14;
- Considere a última variável do problema. *a3* é a variável da classe e é definida na regra 0. O valor de *a3* da nova instância gerada será *c1*, porque a probabilidade do *c1* é 1,0, ou seja, 100% de ocorrência em todas as gerações. Portanto, nunca a variável *a3* de uma instância gerada a partir da regra 0 assumirá o valor de *c*2 ou *c3*;
- A nova instância gerada possuirá os seguintes valores : *a1* = *v1*, *a*2 = 1,14 e *a3* = *c1*;

Portanto foram geradas *n* = 4 instâncias :

1. *a1* = *v1*, *a*2 = 0,92 e *a3* = *c1*;
2. *a1* = *v*2, *a*2 = 1,57 e *a3* = *c*2;
3. *a1* = *v4*, *a*2 = 1,93 e *a3* = *c3*;
4. *a1* = *v1*, *a*2 = 1,14 e *a3* = *c1*;

5.2. Mínimo grau de certeza

Este é o primeiro elemento que tentamos introduzir no método *BayesFuzzy*, um mínimo para o grau de certeza das regras *fuzzy* geradas. Acreditamos que ao estabelecer um mínimo grau de certeza podemos diminuir a quantidade de regras geradas pelo método *BayesFuzzy*, ou seja, as regras que não possuem o grau de certeza acima deste mínimo serão descartadas.

Como vimos na Seção 4.1 sobre o método *BayesFuzzy*, para extrair uma regra de uma rede Bayesiana, necessitamos utilizar a própria rede para classificar uma certa instância e então construir uma regra. Se neste cálculo, associarmos a probabilidade condicional obtida no processo de classificação com o grau de certeza da regra *fuzzy* a ser produzida no processo, conseguiremos atribuir um grau de certeza para as regras *fuzzy* geradas. Desta maneira, todas as regras geradas que não possuem os seus graus de certeza acima do mínimo estabelecido, serão descartadas. Com os resultados empíricos com relação à introdução do mínimo de grau de certeza, queremos descobrir se a utilização de mínimo grau de certeza é efetiva para selecionar as melhores regras e qual é o comportamento do método, de uma maneira geral.

Os testes realizados aqui utilizam uma base de dados criada manualmente para ter um controle sobre como os conjuntos *fuzzy* são distribuídos e sobre as características destes dados. Foram geradas 30 instâncias para os primeiros testes. Mesmo com pouca quantidade de instâncias, podemos tirar algumas conclusões que, mais tarde, em outros experimentos, foram confirmadas, principalmente na Seção 5.4, que são experimentos feitos com conjunto de dados bem mais elaborados e planejados.

Como mencionado anteriormente, para este teste, foram usados dados criados manualmente. O número total das instâncias são: 30 instâncias e 7 variáveis, sendo que 3 delas são relevantes para a classificação e 4 delas não são relevantes para a classificação. A Tabela 2 descreve as variáveis para este conjunto de dados sendo que a variável Idosa é a variável classe.

Os testes envolvem diversas combinações de variáveis relevantes com irrelevantes, aplicação ou não do *Markov Blanket*, utilizando diferentes quantidades de conjuntos *fuzzy* no processo de fuzzificação e também, diferentes mínimos graus de certeza das regras. Os resultados detalhados podem ser encontrados no 0.

Tabela 3 Variáveis contidas nos dados utilizados nos testes da Seção 5.2.

Nome da variável	Possíveis valores	Qt. de conj. fuzzy
Altura (em metros)	Número real	3
Velocidade de caminhar (em metros por segundo)	Número real	3
Sexo	Masculino, Feminino	n/a
Nível de animação	Desanimado, Animado, Muito animado	n/a
Peso (em kilogramas)	Número real	3
Fumante	Verdadeiro, Falso	n/a
Visitas a hospital (por ano)	Número inteiro	3
Idosa	Verdadeiro, Falso	n/a

Foi utilizado o seguinte método para conduzir os testes:

- Realizar uma bateria de testes (definido no item seguinte) utilizando dois, três e quatro conjuntos *fuzzy* triangulares na fuzzificação dos dados, como no exemplo citado no gerador dos dados, realizaremos uma bateria de testes assumindo que a variável *a*2 pode ser representada por 2 conjuntos *fuzzy*, *baixo* e *alto*. Depois, mais duas baterias de testes assumindo que *a*2 pode ser representada por 3 conjuntos *fuzzy* ou 4 conjuntos *fuzzy*. Caso tenha mais do que uma variável *fuzzy*, assumir que todas as variáveis podem ser representadas por 2 conjuntos *fuzzy* na hora de fazer a bateria de testes para 2 conjuntos *fuzzy* e assim por diante;
- Cada bateria de teste é realizada com o objetivo de se obter a quantidade de regras geradas e a taxa de classificação média para os mínimos graus de certeza sendo 0,8, 0,825, 0,85, 0,875, 0,9, 0,925 e 0,95. Estes valores foram definidos arbitrariamente;

A partir dos resultados, podemos perceber os seguintes pontos :

- Ao aumentar a quantidade de variáveis irrelevantes, a quantidade de regras cresceu e a taxa de classificação se manteve quase a mesma, com pequenas variações. Isso ocorre devido ao fato da forma com que as regras foram geradas: gerar uma regra para cada possível valor das variáveis que estão no *Markov Blanket* da variável classe. Portanto, quanto mais variáveis se localizam no *Markov Blanket*, maior a quantidade de regras que serão geradas. Mas isso não necessariamente irá influenciar a qualidade das regras, como acontece de fato com os resultados obtidos, porque a qualidade das regras não é determinada pela quantidade de variáveis irrelevantes, mas sim, pela quantidade de variáveis

relevantes;

- Os dados utilizados possuem 3 variáveis discretas e 4 variáveis numéricas. Todas as variáveis numéricas foram criadas para serem melhor representadas por 2 ou 3 conjuntos *fuzzy*. Portanto, como esperado, podemos notar que quando utilizamos 3 conjuntos *fuzzy*,, obtivemos taxas de classificação melhores. Então podemos afirmar que escolher conjuntos *fuzzy* que melhor representam o padrão de uma variável ajuda a obter uma taxa de classificação mais satisfatória;
- Ao aumentar a quantidade de conjuntos *fuzzy* que se utiliza para representar as variáveis numéricas, aumenta-se também a quantidade de regras. Isso também é causado pela forma com que as regras são geradas, porque uma maior quantidade de conjuntos *fuzzy*, significa que existem mais combinações possíveis, portanto, mais regras são geradas;
- Com o aumento do mínimo grau de certeza das regras, a quantidade de regras geradas decresce, isso é o que se espera que aconteça, já que o propósito é justamente de remover as regras que não possuem esse mínimo;
- Com o aumento do mínimo grau de certeza, pode decair, aumentar ou até manter a taxa de classificação. Isso ocorre devido a utilização do sistema de classificação *fuzzy* geral, pois considera o resultado de todas as regras e não apenas da regra vencedora.

  Neste caso, ao removermos algumas regras, mesmo que possuam menor grau de certeza, podem ainda influenciar o resultado final. Então a perda destas regras “não vencedoras” pode causar uma perda na taxa de classificação.

  Isso pode ser observado, por exemplo na **Figura 13**, quando utilizamos 3 conjuntos *fuzzy* para representar as variáveis numéricas e configuramos o mínimo de grau de confiança de 0,8 a 0,85. As regras são removidas mas não são suficientes para modificar o resultado da classificação. Mas, quando configuramos para 0,875, depois das regras serem removidas, a taxa de classificação aumentou, porém, quando configuramos para 0,9, a taxa de classificação cai. Ou seja, podemos concluir que existem regras benéficas que possuem o grau de certeza entre as faixas de 0,875 e 0,9;

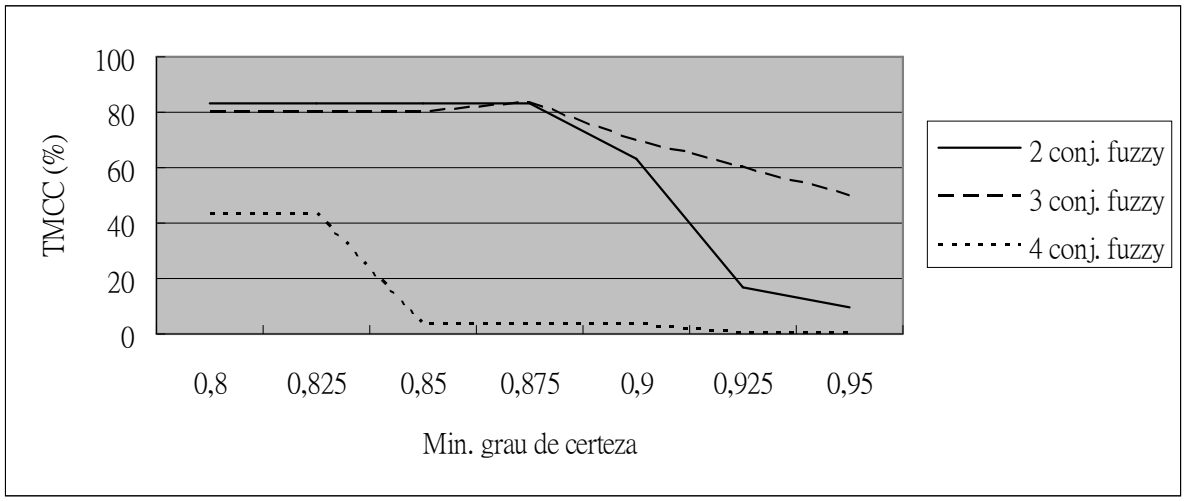

Figura 13 Taxa Média de Classificação Correta (TMCC) de teste sobre uso de mínimo grau de certeza. Teste considerando 2 variáveis relevantes e 1 variável irrelevante no conjunto de dados de treinamento.

- Podemos perceber que, de acordo com o mínimo estabelecido, pode-se atingir um balanceamento bom entre a quantidade de regras geradas com a taxa de classificação. Mas, como achar um mínimo que possui um bom balanceamento é, ainda, uma questão a ser estudada;
- Pode acontecer de existir dezenas de variáveis localizadas no *Markov Blanket* da variável classe; quando isso acontece, será gerada uma grande quantidade de regras e a redução da quantidade das regras por grau de certeza não será um método eficiente para diminuir a quantidade de regras, já que pode acontecer que todas as regras possuam graus de certeza semelhantes, dificultando a definição do mínimo grau de certeza a ser utilizado;
- Há a possibilidade de, após a remoção das regras, as regras resultantes não cubrirem todas as possíveis instanciações do problema, ou seja, existirão instâncias que as regras resultantes não conseguirão classificar. Então, a remoção, por mínimo grau de certeza, ainda precisará ser melhor trabalhada. Uma forma seria, por exemplo, separar as regras pela parte consequente e filtrar cada grupo individualmente tendo diferentes mínimos de certeza, de forma a deixar pelo menos uma regra em cada grupo ou, então, inserir uma regra default no conjunto de regras geradas. Para melhor entender sobre esta abordagem, veja o exemplo a seguir:

Considere um problema com duas variáveis A e B, sendo A binária e B que pode assumir valores de {0, 1, 2}. Considere ainda a variável classe C sendo binária e também que as seguintes regras foram geradas após a aplicação do

mínimo grau de certeza:

se A é 0 e B é 0 então C é 1

se A é 0 e B é 1 então C é 1

Considere também que a seguinte evidência é fornecida para classificação.

A é 1 e B é 2

Para esta evidência, o grau de disparo para as duas regras será 0. Ou seja, nenhuma das regras consegue classificar esta evidência aparecendo uma falha no sistema de classificação;

- O não uso do mínimo grau de certeza aumenta drasticamente o tempo de execução, porque todas as possíveis combinações das variáveis gerarão uma regra, ou seja, caso tenhamos N variáveis, onde cada uma pode ter até M valores diferentes, será gerado um total de M^N regras. Portanto, o uso do mínimo grau de certeza possui um efeito adicional de minimizar o tempo de execução do método.

A **Figura 14** mostra o esquema geral do BayesFuzzy (dado na Seção 4.1), mas agora após a inserção do mínimo grau de certeza.

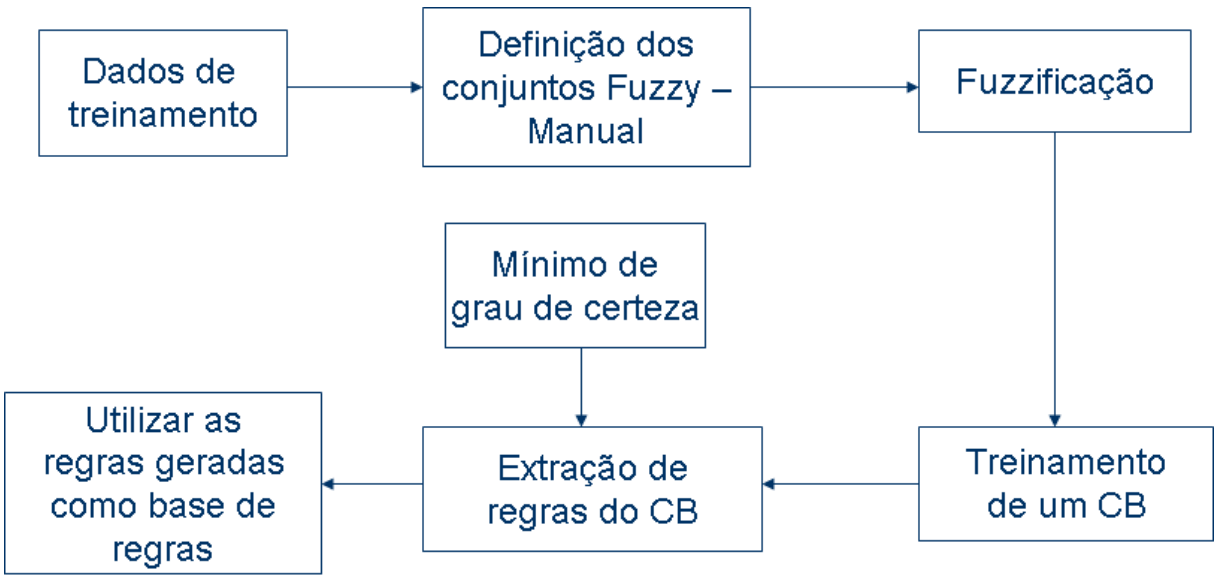

Figura 14 Esquema geral do *BayesFuzzy* após a inserção do mínimo grau de certeza.

5.3. Remoção das regras supérfluas

Até esta etapa, os resultados obtidos já nos mostraram alguns problemas que

existem com a introdução do mínimo grau de certeza e uma possível solução. Mas não foi aplicado a remoção de regras supérfluas e isto gerou uma quantidade grande de regras. Então realizamos novamente os experimentos com a remoção das regras supérfluas.

A remoção das regras supérfluas é realizada de acordo com o Algoritmo 20.

Procedure Redução_regras;
input: R: um conjunto de n regras, cada uma das regras possui um grau de certeza associada;
output: D: um conjunto de regras não redundantes;
init: D := R;
begin
  for i := 1 **to** n **do**
  begin
    for j := i **to** n **do**
      if i ≠ j **then**
      begin
        if D_i possui mesmo consequente que D_j **and**
        D_i possui pelo menos um antecedente igual a qualquer um dos antecedentes de D_j
        then begin
          Constrói D_{new} com mesmos consequentes de D_i e D_j, com os antecedentes que D_i e D_j possuem em comum e associa o maior grau de certeza entre D_i e D_j com D_{new};
          $D_i := D_{new}$;
          Remove D_j do D;
        end;{if-then}
      end;{if-then}
    end;{for}
  end;{for}
  return D;
end.{ Remoção_regras_redundantes}

Algoritmo 18. Remoção das regras.

```
Procedure Remoção_regras_conflitantes_redundantes;
input:  R: um conjunto de n regras, cada uma das regras possui um grau de
certeza associada;
output: D: um conjunto de regras não conflitantes;
init:   D := R;
        D_aux := void;
begin
    for i := 1 to n do
    begin
        for j := 1 to n do
            if i ≠ j then
            begin
                if D_i possui mesmos antecedentes que D_j then
                begin
                    D_aux := a que possui maior grau de certeza entre D_i e D_j;
                    Remove D_j do D;
                    D_i = D_aux;
                end;{if-then}
            end;{if-then}
        end;{for}
    end;{for}
    return D;
end.{ Remoção_regras_conflitantes_redundantes}
```

Algoritmo 19. Remoção das regras conflitantes e redudantes.

```
Procedure Remoção_regras_supérfluas;
input:  R: um conjunto de n regras, cada uma das regras possui um grau de
certeza associada;
output: D: um conjunto de regras não supérfluas;
init:   D := {};
begin
    D := Remoção_regras_conflitantes_redudantes(Redução_regras(R));
    return D;
end.{ Remoção_regras_supérfluas}
```

Algoritmo 20. Remoção das regras supérfluas.

O objetivo deste teste é o de mensurar a diferença na aplicação ou não da remoção das regras supérfluas. Foi utilizado o mesmo método utilizado na Seção 5.2

para conduzir os testes.

Pelos resultados que este teste nos apresenta, a utilização de remoção das regras supérfluas trouxe uma grande diferença na quantidade de regras geradas. Por exemplo, se compararmos os resultados entre o uso ou não desta remoção ao aplicarmos o mínimo grau de certeza em 0.9 e supondo 3 conjuntos *fuzzy*, é produzido em média 210.3 regras com 79% de taxa de classificação correta, quando a remoção não é aplicada, e 19.9 regras com 63% de taxa de classificação correta, quando a remoção é aplicada, como mostra a **Figura 15**, **Figura 16**, **Figura 17** e a **Figura 18**.

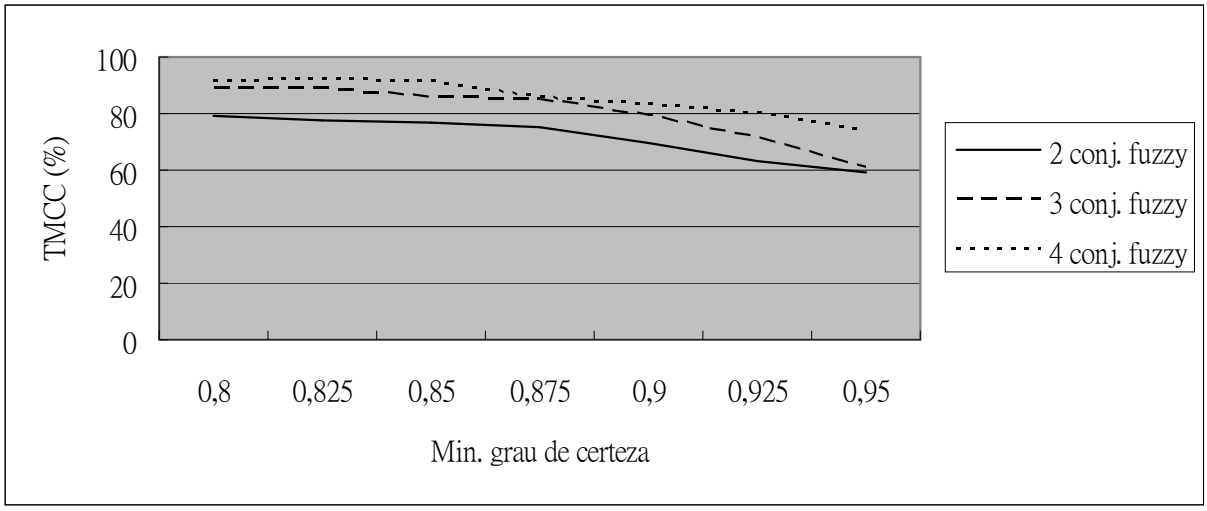

Figura 15 Taxa Média de Classificação Correta (TMCC) de teste sobre remoção das regras supérfluas. Teste não considerando o uso da remoção das regras supérfluas.

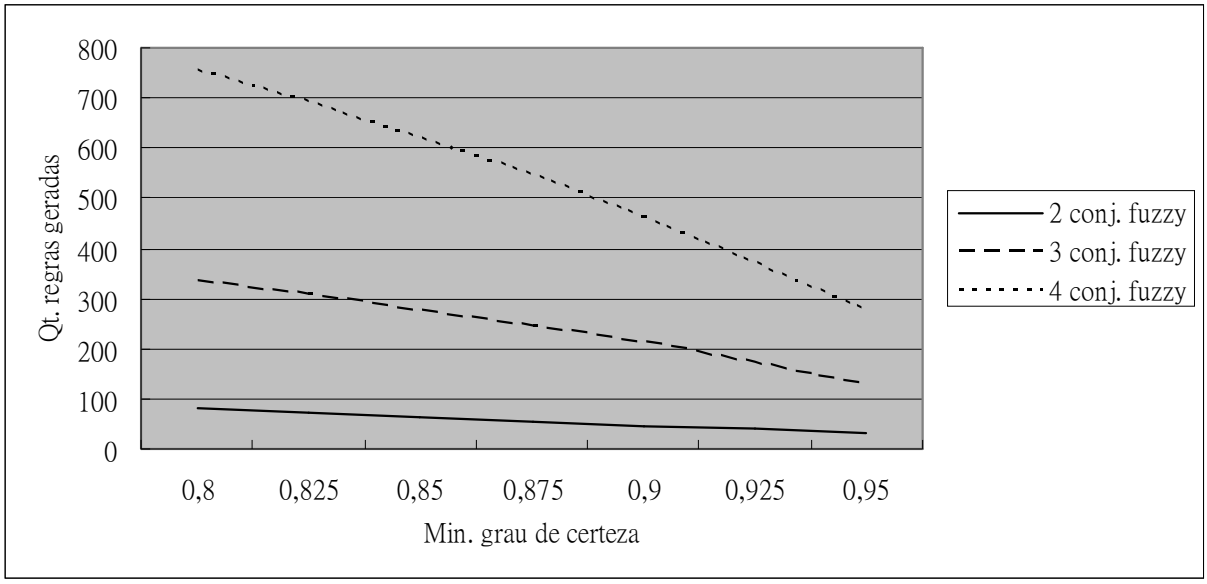

Figura 16 Quantidade de regras geradas de teste sobre remoção das regras supérfluas. Teste não considerando o da remoção das regras supérfluas.

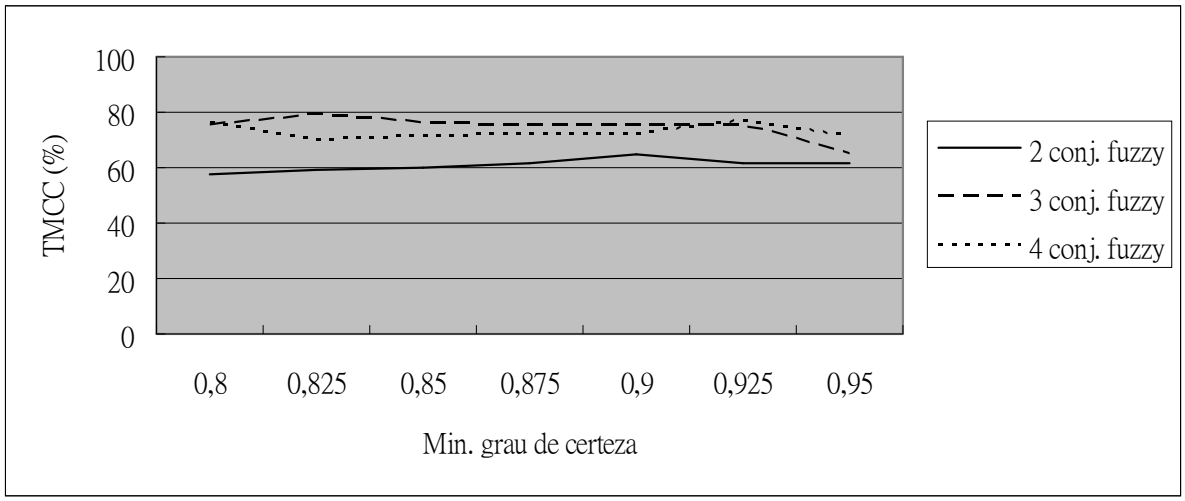

Figura 17 Taxa Média de Classificação Correta (TMCC) de teste sobre remoção das regras supérfluas. Teste considerando o uso da remoção das regras supérfluas.

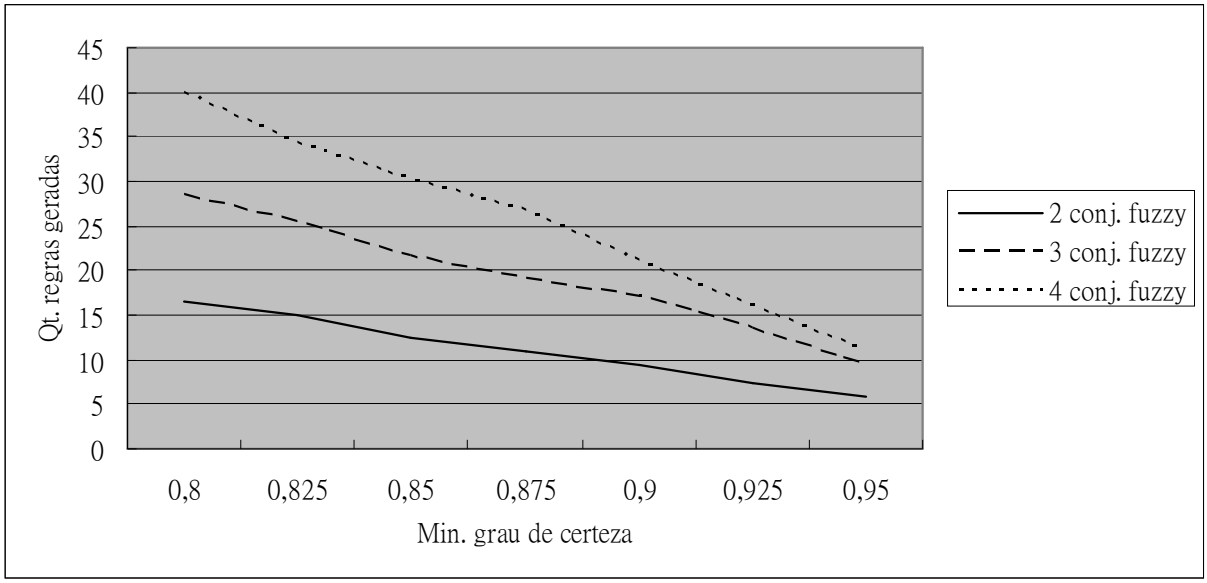

Figura 18 Quantidade de regras geradas de teste sobre remoção das regras supérfluas. Teste considerando o da remoção das regras supérfluas.

Então, podemos verificar que a aplicação de remoção das regras supérfluas traz um grande benefício na quantidade de regras produzidas pelo método. Mas houve uma queda na taxa de classificação que pode ser causada pelo fato de que as regras estão mais gerais e não tão específicas como anteriormente. Estamos utilizando o raciocínio de classificação *fuzzy* geral, então quanto maior a quantidade de regras que reforçam uma certa classe, mesmo sendo redundantes, mais se elevará a chance de certa instância ser classificada como aquela classe.

A partir desta análise, decidimos tomar o rumo da nossa pesquisa para seleção

de atributos, que pode melhorar a taxa de classificação e manter uma baixa quantidade de regras e um baixo número de antecedentes. Porque, se podemos selecionar somente as variáveis que possuam maior influência na variável classe e definir estas variáveis como antecedentes das regras geradas, então teremos regras mais curtas e com maior precisão.

A **Figura 19** mostra o esquema geral da Seção anterior após a inserção da remoção das regras supérfluas.

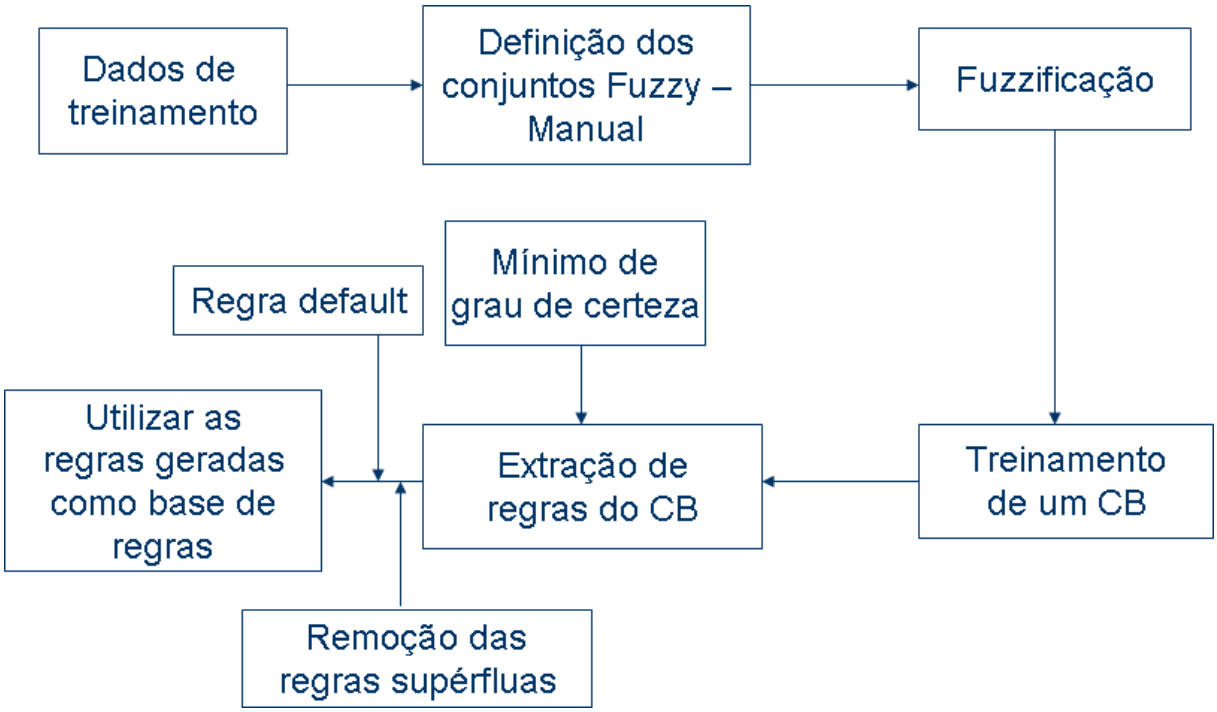

Figura 19 Esquema geral do método *BayesFuzzy* com mínimo grau de certeza, regra default e remoção das regras supérfluas.

5.4. *Rule Post-Pruning*

Para verificar o desempenho do método *Pruned BayesFuzzy*, foi realizada uma análise comparativa com base na implementação J48 [19]. O J48 é uma implementação do algoritmo C4.5 e que utiliza uma árvore de decisão para gerar regras de classificação. Para este teste, temos dois objetivos: saber qual o desempenho do *Pruned BayesFuzzy* quando comparado com um algoritmo (C4.5) clássico e reconhecido de extração de regras de classificação e descobrir se podemos incorporar alguma parte da implementação J48 no método que estamos desenvolvendo. Foi utilizado o mesmo método da Seção 5.2 para conduzir os testes.

Em resumo, os resultados obtidos mostram que o J48, utilizando a técnica de poda, produz menos regras com menos antecedentes e com maior taxa de

classificação. A quantidade de regras e a quantidade de antecedentes que o J48 gera possuem valores bem próximos do PBF e uma grande diferença na taxa de classificação. Os resultados mostram que, ao utilizarmos conjuntos que não são similares com os conjuntos que os dados originalmente representam, as taxa de classificação tendem a baixar e podem gerar uma quantidade de regras maior.

Mesmo observando que, nos experimentos realizados, o J48 apresentou melhores resultados na quantidade de regras geradas, no número de antecedente e na taxa de classificação, não se pode afirmar que o J48 é superior em todos os aspectos em relação ao PBF. O mesmo não necessariamente acontecerá para outros conjunto de dados, pois é necessário realizar mais testes em bases de dados com diferentes características para podemos fazer esta afirmação. Porém, podemos afirmar que, a partir deste teste, percebemos que os dois métodos possuem comportamentos distintos para o conjunto de dados utilizado nos testes. Ou seja, a incorporação do *Rule Post-Pruning* das árvores de decisão C4.5 no PBF poderá trazer benefícios. Os resultados detalhados deste teste localizam-se no 0.

Portanto, os resultados acima mencionados serviram de motivação para a incorporação da técnica de poda utilizada no J48 em *BayesFuzzy*,.Esta poda é a *Rule Post-Pruning*. Portanto o PBF pode ser descrito pelo Algoritmo 21.

A análise de ganho com a incorporação da poda no BayesFuzzy foi realizada de maneira um pouco diferente dos experimentos anteriores. Na validação cruzada utilizada nos experimentos anteriores, 10% dos dados eram utilizados como dados de teste e os 90% restantes dos dados eram utilizados como os dados de treinamento para cada pasta. No caso da análise da poda, utilizaremos 10% dos dados como os dados de teste, outros 10% dos dados como os dados utilizados no processo da poda e os 80% restantes dos dados como os dados de treinamento. Além disso, nestes experimentos, o Bayesfuzzy com poda é testado com alguns algoritmos de aprendizagem da rede Bayesiana diferentes, comparando com os resultados que o método J48 obteve nas mesmas bases de dados.

```
Procedure PBF;
input:   D: um conjunto de dados de treinamento;
         X1: atributo classe;
         min: mínimo grau de certeza;
output:  R: uma base de regras para um sistema de classificação fuzzy;
init:    R := {};
         BC := void;
         MBC := void;
         m := void;
begin
    D := fuzzificacao(D);
    BC := aprender_BC(D);
    MBC := MB(BC, X1);
    R := BC_extrair_regras_com_grau_certeza(MBC);
    m := |R|;
    for i := 1 to m do
        if grau de certeza de Ri < min then
            R := {R1, ..., Ri – 1, Ri + 1, ..., Rm};
    R := remover_regras_superfluas(R);
    R := R ∪ Regra default;
    R := Rule_post_pruning(R);
    R := remover_regras_superfluas(R);
    return R;
end.{PBF}
```

Algoritmo 21. PBF (Pruned BayesFuzzy).

Os algoritmos de aprendizagem utilizados nestes experimentos são K2 e IC, que são algoritmos baseados em busca heurística e em independência condicional respectivamente. Ou seja, são algoritmos que possuem diferentes características e que, por isso, podem apresentar comportamentos diferentes ao cooperarem com o método *BayesFuzzy*. Também é utilizado o *Naïve Bayes* que é um método focado em classificação. Mas a característica principal de uma rede *Naïve* Bayes é que todos os nodos são filhos da classe, ou seja, todos eles pertencem ao *Markov Blanket* da classe, o que aumenta a quantidade de nodos que participarão do processo de geração das regras. Para resolver este problema, estamos introduzindo uma nova técnica que chamaremos de *NaïveMarkov*. Essa técnica utilizará o algoritmo K2 para aprender uma rede Bayesiana e descobrir quais nodos fazem parte do *Markov Blanket* do nodo da classe. A seguir gera-se uma rede *Naïve* Bayesiana que

removerá todos os nodos que não fazem parte do *Markov Blanket* da rede aprendida pelo algoritmo K2. Porém, para este teste, todos os nodos estão presentes no *Markov Blanket* da classe, mostrando nenhuma diferença na aplicação ou não do *NaïveMarkov*. Mas futuramente faremos testes que mostrarão esta diferença.

Serão utilizados sete diferentes tipos de bases de dados, cada uma delas apresentando uma característica específica, o que nos ajudará a descobrir o comportamento do método que estamos desenvolvendo em diferentes situações. Todas as bases, exceto o *Baseline*, possuem versões diferentes com características semelhantes, criadas para medir até quanto o método é influenciado por uma certa característica. Por exemplo, serão geradas para as bases de dados "Variáveis Irrelevantes" em três versões distintas: uma para quando existirem poucas variáveis irrelevantes, uma para quando existir uma quantidade mediana de variáveis irrelevante e uma última para quando existir uma quantidade elevada de variáveis irrelevantes. Para cada versão das bases de dados, serão geradas quatro bases com quantidade de instâncias distintas. A primeira com 1000 instâncias, a segunda com 10000 instâncias, a terceira com 30000 e a quarta e última com 50000 instâncias. Estes números são definidos para medir qual quantidade de dados será suficiente para definir a real distribuição do problema. Estas bases são geradas pelo gerador de dados descritos no início deste capítulo. Os tipos de bases e suas características são :

1. Base de dados "*Baseline*"

   Características :

   - Todas as variáveis são relevantes;
   - Todas as regras que irão gerar os dados possuem todas as variáveis definidas na parte de antecedente. Ou seja, não existem ruídos na base de dado;
   - Não existem regras repetitivas na geração dos dados. Ou seja, não reforça nenhum padrão em específico;
   - Não existem regras que sejam a forma mais geral de alguma outra regra;
   - Possui quantidade de variáveis contínuas igual à quantidade de variáveis discretas;
   - A quantidade de instâncias por classe é balanceada;
   - Não contém informações incertas;
   - Um exemplo do conjunto de regras utilizado é dado no 0.

   Objetivo :

   - Supor uma situação ideal para servir como base de comparação para

outras bases de dados.

Versões das bases :

- *Baseline* com 1000 (*BaselineS*), 10000 (*BaselineM*), 30000 (*BaselineL*) e 50000 (*BaselineXL*) instâncias.

2. Base de dados “Variáveis Irrelevantes” (VI)

Características :

- Três bases de dados, cada uma com uma quantidade diferente de variáveis irrelevantes de forma a ter no final uma base de dados com todas as variáveis irrelevantes; Suponha, por exemplo, que se tenha um conjunto de N variáveis. Assim, teremos uma primeira base de dados onde M (M<N) variáveis são irrelevantes, uma segunda base onde M*C (C é uma constante e M*C<N) variáveis são irrelevantes e assim, sucessivamente até atingirmos a N-ésima base de dados que terá todas as N variáveis irrelevantes.
- Todas as regras de geração excluirão as mesmas variáveis dos seus antecedentes;
- Não existem regras repetitivas na geração dos dados. Ou seja, não reforça nenhum padrão em específico;
- Não existem regras que sejam a forma mais geral de alguma outra regra;
- Possui quantidade de variáveis contínuas igual à quantidade de variáveis discretas;
- A quantidade de instâncias por classe é balanceada;
- Não contém informações incertas;
- Um exemplo do conjunto de regras utilizado é dado no 0.

Objetivo :

- Observar o comportamento do método em relação às variáveis irrelevantes presentes na base de dados.

Versões das bases :

- VI1 com 1000 (VI1S), 10000 (VI1M), 30000 (VI1L) e 50000 (VI1XL) instâncias. VI1 possui duas variáveis irrelevantes;
- VI2 com 1000 (VI2S), 10000 (VI2M), 30000 (VI2L) e 50000 (VI2XL) instâncias. VI2 possui quatro variáveis irrelevantes;
- VI3 com 1000 (VI3S), 10000 (VI3M), 30000 (VI3L) e 50000 (VI3XL)

instâncias. VI3 possui todas as variáveis irrelevantes.

3. Base de dados “Regras Repetitivas” (RR)

   Características :

   - Todas as variáveis são relevantes;
   - Todas as regras que irão gerar os dados, possuem todas as variáveis definidas na parte de antecedente;
   - Possui cópias de uma das regras ocupando mais de 10% da base de dados toda;
   - Não existem regras que sejam a forma mais geral de alguma outra regra. Ou seja, para não reforçar nenhum padrão em específico;
   - Possui quantidade de variáveis contínuas igual à quantidade de variáveis discretas;
   - A quantidade de instâncias por classe não necessariamente é balanceada;
   - Não contém informações incertas;
   - Um exemplo do conjunto de regras utilizado é dado no 0.

   Objetivo :

   - Observar o comportamento do método em relação à situação onde existe uma forte presença de algum padrão.

   Versões das bases :

   - RR com 1000 (RRS), 10000 (RRM), 30000 (RRL) e 50000 (RRXL) instâncias. RR possui uma regra repetitiva.

4. Base de dados “Regras Mais Gerais” (RMG)

   Características :

   - Todas as variáveis são relevantes;
   - Possui uma regra que é a forma mais geral de uma outra regra;
   - Não existem regras repetitivas na geração dos dados. Ou seja, não reforça nenhum padrão em específico;
   - Metade das regras são as formas mais gerais de alguma outra regra. Ou seja, reforça os padrões definidos nas regras mais gerais;
   - Possui quantidade de variáveis contínuas igual à quantidade de variáveis discretas;
   - A quantidade de instâncias por classe é balanceada;
   - Não contém informações incertas;
   - Um exemplo do conjunto de regras utilizado é dado no 0.

Objetivo :

- Objetivo principal é de observar o comportamento do método em relação às regras que são formas mais gerais das outras regras;
- Também pode observar um pouco sobre o tratamento de ruídos do método, porém esse será o objetivo principal de uma outra base de dados.

Versões das bases :

- RMG com 1000 (RMGS), 10000 (RMGM), 30000 (RMGL) e 50000 (RMGXL) instâncias. RMG possui uma regra que é mais geral do que a outra.

5. Base de dados “Ruídos”

Características :

- Todas as variáveis são relevantes;
- Todas as regras de geração possuem 2 variáveis definidas na parte antecedente e nenhum par de regras possuem as mesmas variáveis definidas na parte antecedente. Isso simularia a presença de ruídos nos dados;
- Não existem regras repetitivas na geração dos dados. Ou seja, não reforça nenhum padrão em específico;
- Não existem regras que sejam a forma mais geral de alguma outra regra;
- Possui quantidade de variáveis contínuas igual à quantidade de variáveis discretas;
- A quantidade de instâncias por classe é balanceada;
- Não contém informações incertas;
- Um exemplo do conjunto de regras utilizado é dado no 0.

Objetivo :

- Observar o comportamento do método em relação aos ruídos que existem na base de dados.

Versões das bases :

- Ruidos1 com 1000 (Ruidos1S), 10000 (Ruidos1M), 30000 (Ruidos1L) e 50000 (Ruidos1XL) instâncias. Todas as regras que são utilizadas para definir a base Ruidos1 possuem 2 variáveis de ruído, ou seja, não são definidos na regra;

- Ruidos2 com 1000 (Ruidos2S), 10000 (Ruidos2M), 30000 (Ruidos2L) e 50000 (Ruidos3XL) instâncias. Todas as regras que são utilizadas para definir a base Ruidos2 possuem 4 variáveis de ruído, ou seja, não são definidos na regra;
- Ruidos3 com 1000 (Ruidos3S), 10000 (Ruidos3M), 30000 (Ruidos3L) e 50000 (Ruidos3XL) instâncias. Todas as regras que são utilizadas para definir a base Ruidos3 possuem 5 variáveis de ruído, ou seja, não são definidos na regra.

6. Base de dados “Incerteza”

Características :

- Todas as variáveis são relevantes;
- Todas as regras de geração possuem todas as variáveis definidas na parte de antecedente;
- Não existem regras repetitivas na geração dos dados. Ou seja, não reforça nenhum padrão em específico;
- Não existem regras que sejam a forma mais geral de alguma outra regra;
- Possui quantidade de variáveis contínuas igual à quantidade de variáveis discretas;
- A quantidade de instâncias por classe é balanceada;
- Contém informações incertas;
- Um exemplo do conjunto de regras utilizado é dado no 0.

Objetivo :

- Observar o comportamento do método em relação à presença de informações incertas contidas na base de dados.

Versões das bases :

- Incerteza1 com 1000 (Incerteza1S), 10000 (Incerteza1M), 30000 (Incerteza1L) e 50000 (Incerteza1XL) instâncias. Incerteza1 possui pouca incerteza inserida na base, ou seja, existe um certo domínio de algum padrão;
- Incerteza2 com 1000 (Incerteza2S), 10000 (Incerteza2M), 30000 (Incerteza2L) e 50000 (Incerteza2XL) instâncias. Incerteza2 possui um nível mediano de incerteza inserido na base, ou seja, o domínio de algum padrão não é tão forte quanto a Incerteza1;
- Incerteza3 com 1000 (Incerteza3S), 10000 (Incerteza3M), 30000 (Incerteza3L) e 50000 (Incerteza3XL) instâncias. Incerteza3 possui um

nível alto de incerteza inserido na base, ou seja, quase não existe domínio de algum padrão.

7. Base de dados “Classe Desbalanceada”

Características:

- Todas as variáveis são relevantes;
- Todas as regras de geração possuem todas as variáveis definidas na parte de antecedente;
- Não existem regras repetitivas na geração dos dados. Ou seja, não reforça nenhum padrão em específico;
- Não existem regras que sejam a forma mais geral de alguma outra regra;
- Possui quantidade de variáveis contínuas igual à quantidade de variáveis discretas;
- A quantidade de instâncias por classe é desbalanceada;
- Não contém informações incertas;
- Um exemplo do conjunto de regras utilizado é dado no 0.

Objetivo :

- Observar o comportamento do método quando a base é desbalanceada.

Versões das bases :

- Desb1 com 1000 (Desb1S), 10000 (Desb1M), 30000 (Desb1L) e 50000 (Desb1XL) instâncias. Desb1 possui a distribuição dos valores da classe *c1*, *c*2 e *c3* como sendo respectivamente 2:4:6;
- Desb2 com 1000 (Desb2S), 10000 (Desb2M), 30000 (Desb2L) e 50000 (Desb2XL) instâncias. Desb2 possui a distribuição dos valores da classe *c1*, *c*2 e *c3* como sendo respectivamente 2:6:10;
- Desb3 com 1000 (Desb3S), 10000 (Desb3M), 30000 (Desb3L) e 50000 (Desb3XL) instâncias. Desb3 possui a distribuição dos valores da classe *c1*, *c*2 e *c3* como sendo respectivamente 2:10:14.

Por conta do número de bases de dados gerados, nestes experimentos muitos resultados são gerados. Analisando-se, entretanto, o comportamento dos algoritmos nas bases com 1000, 3000, 5000 e 10000 registros observou-se um comportamento bem próximo para as bases contendo 3, 5 e 10 mil registros. Assim, para facilitar a sequencia das análises, foram utilizadas apenas as bases com 10000 registros. Também foram utilizadas (nas análises subsequentes) as versões de bases de dados

que apresentam forte presença de certa característica, que são as versões 2, mas que não levam esta característica ao extremo, como as versões 3.

Serão apresentados aqui os resultados para as bases *Baseline*M, VI2M, RRM, RMGM, Ruídos2M, Incerteza2M e Desbalanceada2M. Mas, mesmo assim, existe uma grande quantidade de resultados sendo gerados. Portanto, após a análise, percebemos que o algoritmo *NaïveMarkov* não trouxe nenhuma alteração com relação aos resultados obtidos com o NaiveBayes, isso ocorre devido ao fato de que todos os nodos destas bases fazem parte do *Markov Blanket* da classe. Por este motivo os resultados com o NaïveMarkov não são reportados.

Observa-se também que os testes que variam o mínimo grau de certeza entre 0.8 a 0.95 mostram poucas diferenças nos resultados. Portanto, nos gráficos a seguir, não estão presentes resultados que mostram esta variação. são apresentados apenas os resultados utilizando o mínimo grau de certeza de 0.8.

Foi utilizado o mesmo método da Seção 5.2 para conduzir os testes, mas os mínimos de grau de certeza utilizados neste teste são 0.8, 0.85, 0.9 e 0.95.

A **Figura 20** mostra os resultados de Taxa Média de Classificação Correta (TMCC) e a quantidade média de regras geradas (#Regras). A partir dela podemos concluir que:

- Podemos perceber que na maioria dos casos, ao aplicar a poda, a TMCC cai. A única exceção é em relação ao K2, porque o K2 é um algoritmo que se preocupa com a distribuição de probabilidade real de todas as variáveis do problema. Mas, quando se aplica a poda, a rede aprendida pelo K2 passa a ser um pouco mais específica para o problema de classificação, o que pode ter causado um ganho na TMCC em alguns casos.
- Na teoria, a aplicação da poda *Rule Post-Pruning* deve trazer ganhos na TMCC, mas, como podemos ver, em muitos casos isto não acontece. Isso pode ser causado devido ao seguinte fato: a aplicação da *Rule Post-Pruning* reduz o número de antecedentes da regra, e portanto, a regra passa a ter capacidade de classificar mais instâncias. Esta capacidade de poder classificar mais instâncias trouxe uma TMCC mais alta na hora de decidir a regra que se resulta do *Rule Post-Pruning*.

  Porém, as formas mais gerais das regras não necessariamente possuem o mesmo potencial de classificação que a forma mais específica das regras e, portanto, quando são utilizadas na classificação real, a TMCC cai.
- Podemos notar também que o #Regras cai na maioria dos casos após a aplicação da poda, ou seja, um ganho na compreensibilidade. Mas na base de dados VI, o J48 teve um comportamento bem diferente. Após a

aplicação da poda, houve um aumento do #Regras. Uma das possíveis causas é a de que com a poda, um novo conjunto de regras é gerado pela árvore de decisão. Este novo conjunto de regras pode não ter tantas regras supérfluas a serem removidas, causando um #Regras mais elevado.

A queda na TMCC em troca da melhora na compreensão da rede, é um comportamento esperado, porém esta troca não parece ser tão vantajosa nestes primeiros resultados por se estar perdendo muita precisão na TMCC com pouco ganho na redução do #Regras.

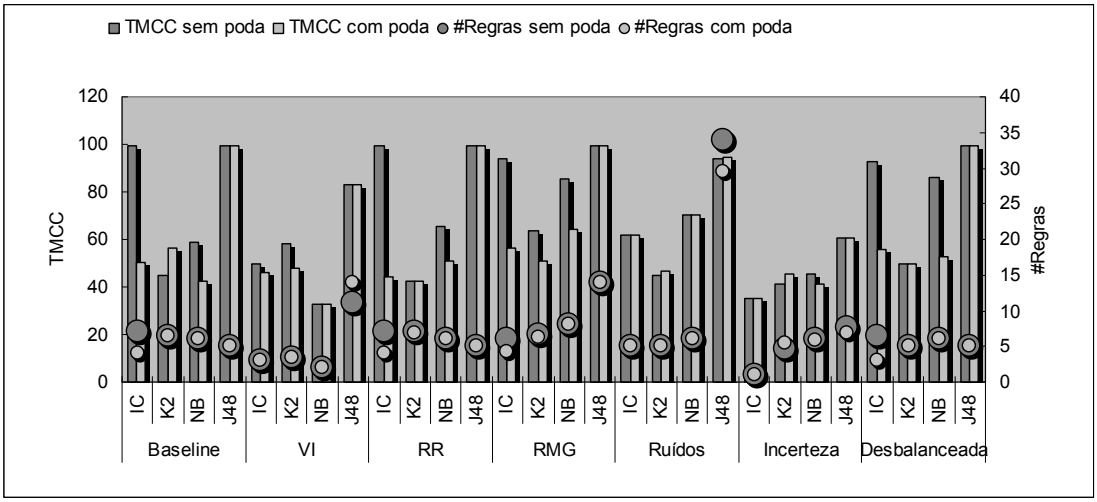

Figura 20 Gráfico mostrando os resultados obtidos em relação à Taxa Média de Classificação Correta (TMCC) e quantidade média de regras (#Regras).

A **Figura 21** mostra os resultados da TMCC com quantidade média de antecedentes (#Ant.). A partir destes resultados podemos concluir que:

- Este gráfico nos mostra que houve uma queda significativa no #Ant. em muitos casos, o que demonstra um grande ganho em relação à compreensibilidade da rede. É uma melhora muito mais significativa em relação ao #Regras, como mostra a **Figura 22**.

  Podemos considerar também que um ganho no #Ant. é mais favorável do que um ganho no #Regras na mesma proporção. Isso porque, uma regra com menos antecedentes representa uma regra na sua forma mais geral, o que servirá para explicar diversos casos. Também consideramos que saber mais sobre os casos gerais facilita na compreensão de um problema do que saber mais sobre os casos específicos.

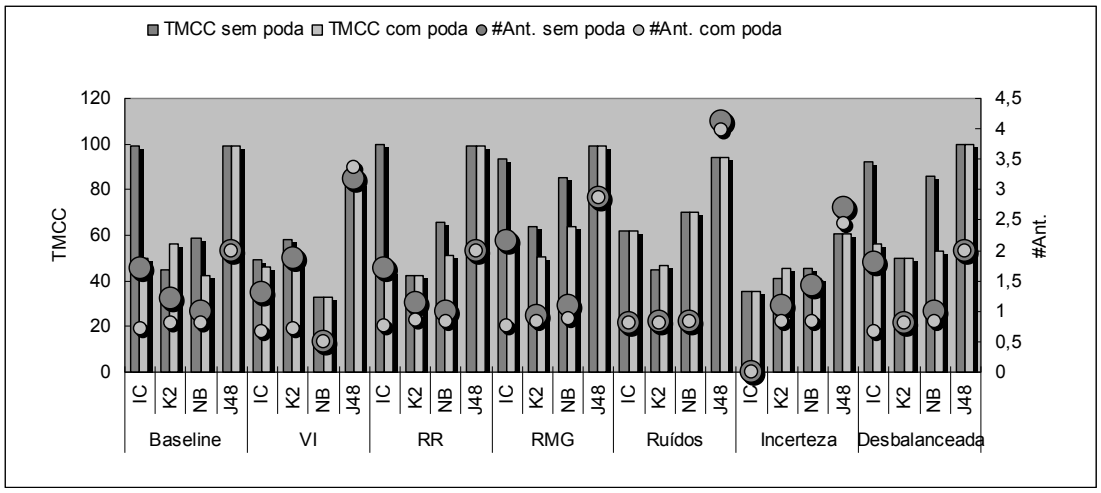

Figura 21 Gráfico mostrando os resultados obtidos em relação à Taxa Média de Classificação Correta (TMCC) e quantidade média de antecedentes (#Ant.).

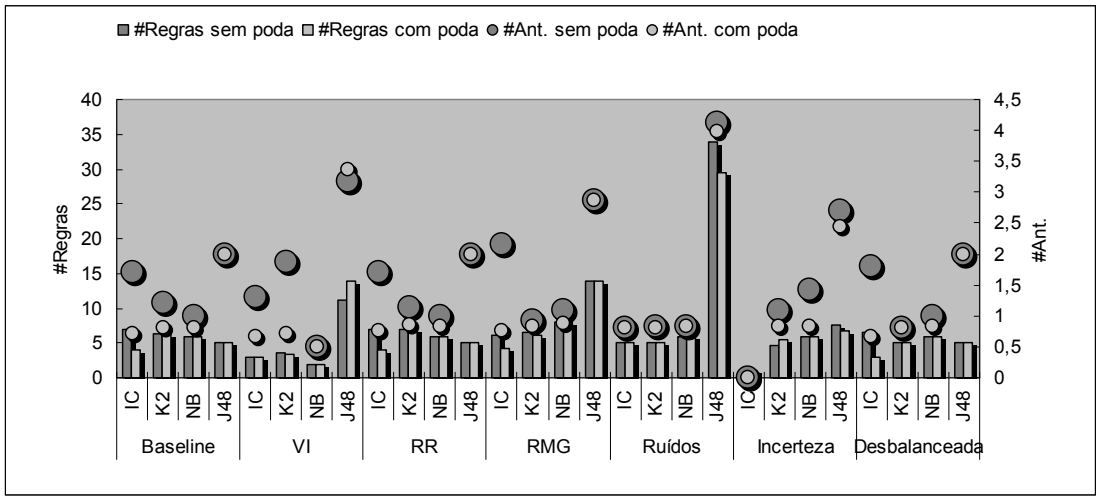

Figura 22 Gráfico mostrando os resultados obtidos em relação à quantidade média de regras geradas (#Regras) e quantidade média de antecedentes (#Ant.).

- Podemos dizer também que, a queda no #Ant. tem como uma possível consequência, uma queda no #Regras devido ao processo de remoção das regras supérfluas. Por exemplo, considere que temos duas regras:

  se A é v1 e B é v2 então Classe é c1 com 0,9 de certeza
  se A é v1 e B é v1 então Classe é c2 com 0,8 de certeza

  e suponha que, após o processo de redução das antecedentes das regras pela poda *Rule Post-Pruning*, a variável B das duas regras seja removida, então as regras passam a ser:

se A é v1 então Classe é c1 com 0,9 de certeza
se A é v1 então Classe é c2 com 0,8 de certeza

aplicando a remoção das regras supérfluas, teremos como resultado, uma única regra:

se A é v1 então Classe é c1 com 0,9 de certeza

Ou seja, neste caso, uma redução na quantidade de antecedentes causou uma redução na quantidade de regras também. Mas não podemos afirmar que a redução da quantidade de antecedentes, TMCC, aumenta, como mostra o IC na base Baseline e K2 nesta mesma base também. O IC obteve menos antecedentes, mas com um TMCC menor também. Já o K2 teve menos antecedentes, mas com um TMCC maior.

A **Figura 23** mostra o esquema geral do PBF. Note que a remoção das regras supérfluas é aplicada novamente após a poda *Rule Post-Pruning*, porque existe a possibilidade de que, após as regras serem podadas, novas regras supérfluas apareçam.

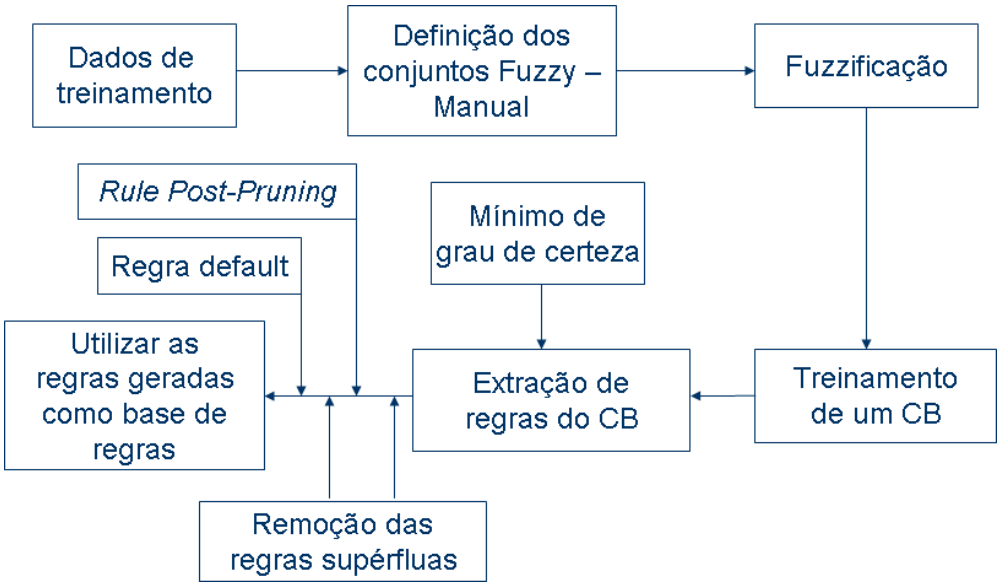

Figura 23 Esquema geral do *Pruned BayesFuzzy.*

6. ALGORITMO *PRUNED BAYESFUZZY 2*

O algoritmo *Pruned BayesFuzzy2* (PBF2) tem como base o PBF, herdando assim todas as suas funcionalidades e características. A principal mudança é a incorporação de um novo método de seleção de atributo desenvolvido especificamente para otimizar o PBF. Este novo método de seleção de atributos se baseia no conceito de *Markov Blanket*, e será chamado de "Seleção por Força de Relação de *Markov Blanket*" (SFRMB).

A necessidade de incorporar esta técnica de seleção de atributos no PBF ocorre pelo fato de que em experimentos realizados com uma base, dados que descrevem um problema real, observou-se que a utilização do *Markov Blanket* para a seleção de variáveis não foi capaz de diminuir (de maneira satisfatória) a quantidade de atributos a serem utilizados nas regras. Assim, a grande quantidade de atributos gera uma grande quantidade de regras e demanda grande esforço e recurso computacional. Assim, pode-se concluir que mesmo com todo esforço de tentar minimizar a quantidade de regras que esta sendo gerada pela aplicação das podas, ainda não temos garantia de que em todas as situações, estas podas possam nos garantir uma quantidade viável de regras.

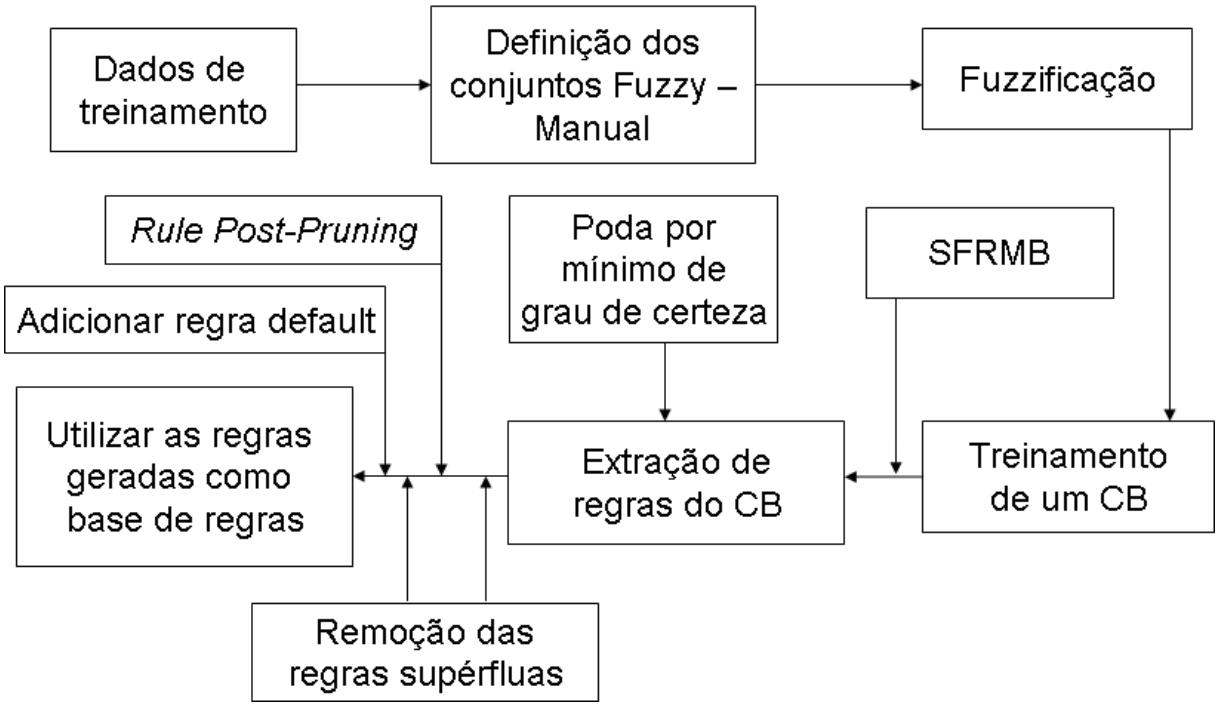

Figura 24 Esquema geral do *Pruned BayesFuzzy* 2.

Portanto, há uma necessidade extra de selecionar entre os atributos contidos no *Markov Blanket* da classe, aqueles que são mais relevantes do que os outros. E não somente identificá-los, mas sim, de acordo com os possíveis valores que os atributos

mais relevantes podem assumir, estimar a quantidade de regras que serão geradas e estabelecer um limitante nesta quantidade. Desta forma, podemos estimar a quantidade de atributos que serão considerados no processo de geração das regras.

A **Figura 24** é o esquema geral de PBF2 após a inclusão de SFRMB e o PBF2 pode ser descrito pelo Algoritmo 22. Na Seção seguinte, discutiremos com mais detalhes o SFRMB.

```
Procedure PBF2;
input:   D: um conjunto de dados de treinamento;
         X1: atributo classe;
         min: mínimo grau de certeza;
         maxReg: máximo de regras a serem geradas;
output:  R: uma base de regras para um sistema de classificação fuzzy;
init:    R := {};
         BC := void;
         MBC := void;
         SFMBC := void;
         m := void;
begin
    D := fuzzificacao(D);
    BC := aprender_BC(D);
    MBC := MB(BC, X1);
    SFMBC := SFRMB(MBC, maxReg);
    R := BC_extrair_regras_com_grau_certeza(SFMBC);
    m := |R|;
    for i := 1 to m do
        if grau de certeza de Ri < min then
            R := {R1, ..., Ri – 1, Ri + 1, ..., Rm};
    R := remover_regras_superfluas(R);
    R := R ∪ Regra default;
    return R;
end.{PBF}
```

Algoritmo 22. PBF2 (Pruned BayesFuzzy 2).

6.1. Seleção pela força de relação de *Markov Blanket*

Como mencionado na Seção anterior, o objetivo de SFRMB é limitar a quantidade de atributos que participarão do processo de geração das regras. Para

atingir este objetivo, buscou-se otimizar a seleção de atributos que já estava sendo aplicada no PBF, que tinha como base o *Markov Blanket*.

Tomando como base a definição de Markov Blanket dada na Seção 2.4, pode-se dizer que os atributos que estão no *Markov Blanket* da variável classe são todos os atributos que possuem uma influência significativa no valor da classe. Portanto, uma das idéias é determinar qual entre estes atributos, possui maior influência. Para saber isso, observem o exemplo abaixo:

Considere 3 variáveis binárias, v_a, v_b e v_c, onde v_a é a variável classe, v_b e v_c são os filhos de v_a e são condicionalmente independentes entre si, como mostra a Figura 25. As tabelas de probabilidade condicional deles são:

v_a: $P(v_a = 0) = 0.3$
$P(v_a = 1) = 0.7$

v_b: $P(v_b = 0|v_a = 0) = 0.8$
$P(v_b = 0|v_a = 1) = 0.1$
$P(v_b = 1|v_a = 0) = 0.2$
$P(v_b = 1|v_a = 1) = 0.9$

v_c: $P(v_c = 0|v_a = 0) = 0.3$
$P(v_c = 0|v_a = 1) = 0.4$
$P(v_c = 1|v_a = 0) = 0.7$
$P(v_c = 1|v_a = 1) = 0.6$

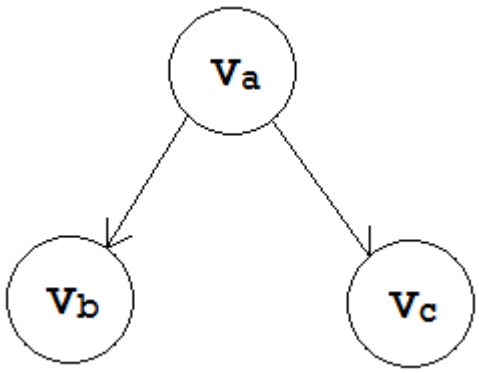

Figura 25 Uma rede Bayesiana utilizada no exemplo de SFRMB.

Neste exemplo, podemos considerar que, mesmo sabendo que tanto v_b quanto v_c pertencem ao Markov Blanket da classe, v_b tem uma relação mais forte com v_a do

que v_c com v_a. Isto porque quando v_a modifica o seu valor (de 0 para 1 ou de 1 para 0), a probabilidade condicional de v_b sofre uma maior modificação em relação a v_c, $P(v_b = 0|v_a = 0) = 0.8$ para $P(v_b = 0|v_a = 1) = 0.1$ e $P(v_c = 0|v_a = 0) = 0.3$ para $P(v_c = 0|v_a = 1) = 0.4$, ou seja, uma modificação em v_a causou um maior impacto na probabilidade condicional do v_b.

Mas, como se pode perceber, este conceito só se aplica para as variáves que possuem uma relação direta entre si. Portanto, para facilitar o nosso problema, o primeiro passo do SFRMB é considerar apenas os pais e os filhos da classe. Os pais dos filhos da classe que originalmente são considerados pelo *Markov Blanket* são removidos.

Com base nesta idéia, a seguinte fórmula é aplicada para calcular a força de relação de *Markov Blanket* das variáveis que estão no *Markov Blanket* da classe.

$$St_x = \sum_{i=1}^{n-1} \{ \sum_{j=i+1}^{n} [\sum_{k=1}^{q} (|p_{ik} - p_{jk}|) / q] / (n - i)\} / (n-1) \qquad (3)$$

onde:

St_x = Força da relação entre a variável *x* e a variável da classe;

n = Número de possíveis valores que a variável da classe pode assumir;

q = Número de possíveis valores que a *x* pode assumir;

p_{ik} = Probabilidade P(X=A|classe=B,C), onde, se a variável da classe é pai da variável *x*, então A = i-ésimo valor da variável *x*, B = k-ésimo valor da variável da classe e C = o(s) primeiro(s) valor(es) de todas as outras variáveis que são pais da *x* ou se a variável *x* é pai da variável da classe, A = i-ésimo valor da variável da classe, B = k-ésimo valor da variável *x* e C = o(s) primeiro(s) valor(es) de todas as outras variáveis que são pais da variável da classe;

SFRMB recebe duas entradas: uma que é o Markov Blanket encontrado e a outra que é o máximo de regras a serem geradas. Então podemos descrever o SFRMB em três passos:

1. Eliminar todos os atributos que não são pais ou filhos da classe;
2. Calcular a força de relação dos atributos restantes e ranqueá-los;
3. Considerar q(xi) sendo a quantidade de possíveis valores que o nodo xi pode assumir e que a quantidade de variáveis que está sendo estimada é N, então a estimativa da quantidade de regras a ser gerada será $\prod_{i=1}^{N} q(xi)$, a seguir, eliminar as variáveis com menos força de acordo com o ranking obtido no passo 2 até que a estimativa esteja abaixo do máximo de regras a ser gerada.

Para entender melhor SFRMB, considere o exemplo da **Figura 26**:

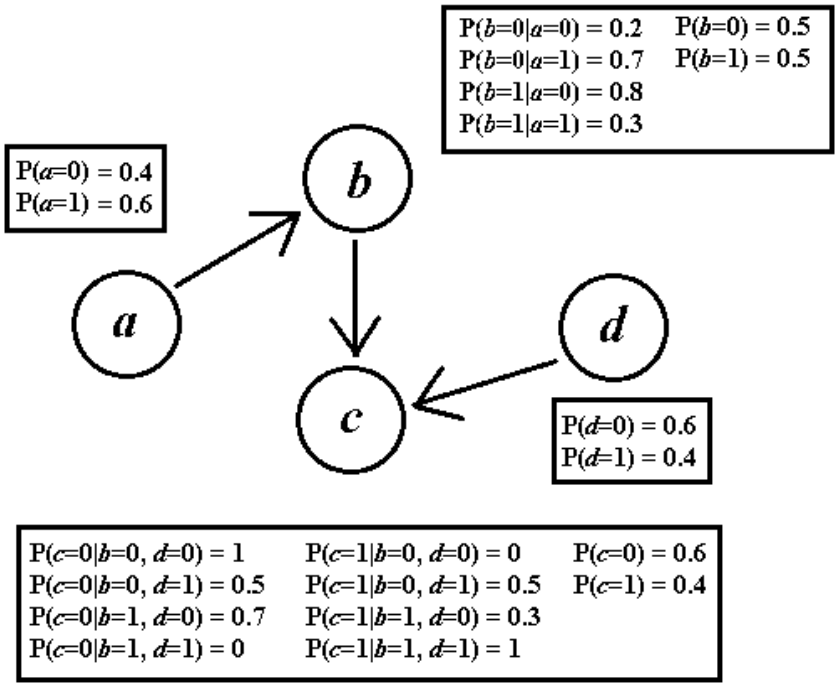

Figura 26 Exemplo para SFRMB.

Onde todas as variáveis são binárias e *b* é a variável classe onde estabelecemos que no máximo serão gerados 3 regras.

No passo 1 de SFRMB, eliminaremos a variável *d*, já que ele não possui uma relação direta com o *b*.

No passo 2, utilizaremos a Fórmula (3) para obter St_a e St_c:

$$St_a = \sum_{i=1}^{1} \{ \sum_{j=1+1}^{2} [\sum_{k=1}^{2} (|p_{ik} - p_{jk}|) / 2] / (2 - 1)\} / 1$$

$$= \sum_{i=1}^{1} \{ \sum_{j=2}^{2} [(|P(a=0|b=0) - P(a=0|b=1)| + |P(a=1|b=0) - P(a=1|b=1)|)/ 2]$$

$$/ 1\} / 1$$

Aplicando o teorema de *Bayes*:

$$St_a = \{[|P(b=0|a=0)*P(a=0)/P(b=0) - P(b=1|a=0)*P(a=0)/P(b=1)| +$$
$$|P(b=0|a=1)*P(a=1)/P(b=0) - P(b=1|a=1)*P(a=1)/P(b=1)|] / 2\}$$
$$= (|0.2*0.4/0.5 - 0.8*0.4/0.5| + |0.7*0.6/0.5 - 0.3*0.6/0.5|) / 2$$
$$= 0.48$$

$$St_c = \sum_{i=1}^{1} \{ \sum_{j=1+1}^{2} [\sum_{k=1}^{2} (|p_{ik} - p_{jk}|) / 2] / (2 - 1)\} / 1$$

$$= \sum_{i=1}^{1} \{ \sum_{j=2}^{2} [(|P(c=0|b=0, d=0) - P(c=0|b=1, d=0)| +$$
$$|P(c=1|b=0, d=0) - P(c=1|b=1, d=0)|) / 2] / 1\} / 1$$

$$= [|1 - 0.7| + |0 - 0.3|] / 2 = 0.3$$

Temos então, $St_a = 0.45$ e $St_c = 0.3$, o ranking deles será {*a*, *c*}, onde da esquerda para direita é a ordem das variáveis com maior força de relação para as variáveis com menor força de relação.

No passo 3, consideramos o máximo estabelecido de 3 regras. A estimativa de regras que será gerada é de 4 (quantidade de possíveis valores da variável *a* multiplicada pela quantidade de possíveis valores da variável *c*). Esta quantidade é maior do que o máximo estabelecido, então eliminaremos a última variável no ranking e estimamos novamente a quantidade de regras que será gerada. Neste caso, a variável *c* é eliminada sobrando apenas a variável *a* com estimativa de gerar 2 regras. Esta quantidade é menor do que o máximo estabelecido, então nenhuma variável será eliminada e então as variáveis resultantes são *b* e *a*.

6.2. Resultados obtidos

Os testes realizados com o PBF2, além de permitir a análise do comportamento do SFRMB, também tem como objetivo, descobrir o quanto PBF e PBF2 são influenciados pelo algoritmo de aprendizado de redes Bayesianas.

Para estes testes, foram utilizados dados sintéticos oferecidos pela empresa E-Biz Solution e adaptados pelo aluno para o contexto do trabalho. Estes dados possuem 45 variáveis e 50000 instâncias. As variáveis podem assumir até dezenas de milhares de possíveis valores, por exemplo, bairro onde uma pessoa mora.

Consideramos que este ambiente não é apropriado para aplicar um teste, fora a imensa quantidade de possíveis valores, a quantidade de instâncias não é suficiente para que um padrão possa ser identificado. Portanto a adaptação feita selecionou

manualmente 8 variáveis consideradas mais relevantes, limitou a quantidade de possíveis valores que cada variável pode assumir para não mais que 10 possíveis valores para, a partir daí, construir uma rede e gerar dados a partir desta rede utilizando a ferramenta *GeNIe*.

Nesta rede construída, inserimos padrões artificiais, desta forma, temos mais certeza se as regras que foram geradas pelo método estão próximas dos padrões inseridos. Além disso, os testes também foram executados nas bases com mesmas características que as bases nas quais o PBF foi executado.

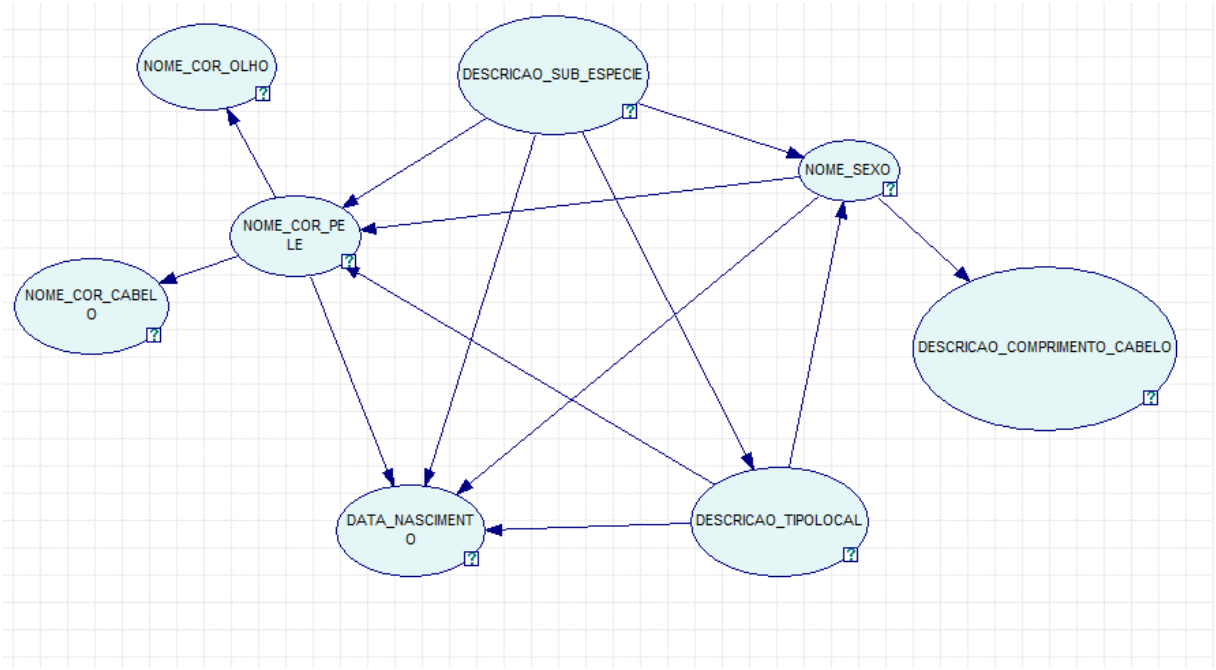

Figura 27 Rede original utilizada para a geração de dados a partir da ferramenta *GeNIe.*

A **Figura 27** mostra a rede na sua forma original e a **Figura 28** mostra a rede aprendida pelo algoritmo IC utilizada na PBF e PBF2. Como podemos ver, as duas redes são bem semelhantes, sendo que há apenas um arco adicional (NOME_COR_OLHO -> DESCRICAO_COMPRIMENTO_CABELO) e um arco a menos (DESCRICAO_TIPO_LOCAL -> NOME_SEXO) na rede aprendida pelo IC e também dois arcos com direções inversas (DESCRICAO_TIPO_LOCAL - DESCRICAO_SUB_ESPECIE e NOME_SEXO - DESCRICAO_SUB_ESPECIE).

Portanto, podemos considerar que o algoritmo de aprendizagem de redes Bayesiana teve uma boa performance e, portanto, podemos analisar os resultados como sendo comportamentos da extração de regras e das podas que foram aplicadas.

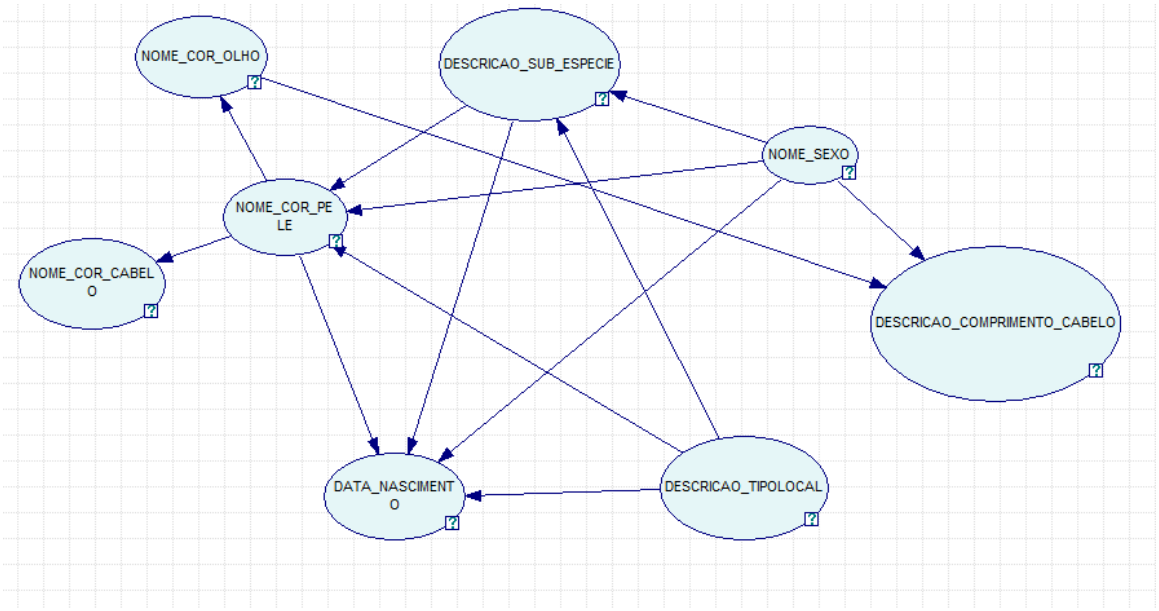

Figura 28 Rede aprendida pelo algoritmo IC.

Mas como podemos ver na **Tabela 4**, os resultados não são ideais tanto para PBF, PBF2 e para J48. Podemos concluir que os dados não foram projetados de forma adequada, ou que a quantidade de 500000 instâncias não é suficiente para capturar os padrões existentes, ou ainda, que o algoritmo J48 não foi capaz de identificar padrões existentes nos dados. Porém, a rede Bayesiana foi construída de forma adequada. Na sequência é feita a análise com relação às bases de dados já utilizadas nos experimentos relativos ao PBF (veja Seção 5.4).

Tabela 4. Resultados da aplicação do PBF2 nos dados sintéticos oferecido pela E-Biz e modificado para as necessidades dos testes realizados.

Config.	TMCC	#Regras	#Ant.	URD
NPMB_0.8	24,4	6,8	2,97	6
PMB_0.8	25,87	4,1	0,75	3
NPSFRMB_10_0.8	15,02	1	0	50000
PSFRMB_10_0.8	15,02	1	0	50000
NPSFRMB_30_0.8	15,02	2	1	15
PSFRMB_30_0.8	15,02	2	0,5	2
NPSFRMB_50_0.8	15,02	2	1	15
PSFRMB_50_0.8	15,02	2	0,5	2
NPSFRMB_100_0.8	27,63	3	1,67	6
PSFRMB_100_0.8	28,4	3	0,67	2
NPJ48	13.97	1	1	NaN
PJ48	13.97	1	1	NaN

Legenda para **Tabela 4**:

- Config. = Configuração que foi utilizada.
- TMCC = Taxa média de classificação correta.
- #Regras = Quantidade média de regras que foram geradas durante os 10 folds.
- #Ant. = Quantidade média de antecedentes das regras que foram geradas durante os 10 folds.
- URD = Quantidade média de vezes que a regra default foi utilizada.
- (P|NP)MB_X = P significa o uso do *Rule-Post Pruning (RPP)* e NP significa o não uso do *RPP*; MB significa que a seleção de atributos que foi utilizada foi *Markov Blanket*; X significa o mínimo de grau de certeza que foi utilizada. Estes são aplicações de PBF.
- (P|NP)SFRMB_Y_X = P, NP e X tem os mesmos significados que o item anterior; SFRMB é o tipo de seleção de atributos que foi utilizado; Y é o máximo de regras que devem ser geradas. Estes são aplicações de PBF2.
- (P|NP)J48 = P e NP tem os mesmos significados que os anteriores; J48 é o método que foi utilizado para a geração das regras.

Nos resultados que obtivemos nas bases com mesmas características que as bases nas quais o PBF foi aplicado, além de permitir uma análise comparativa com o PBF2, tivemos também novas conclusões sobre o uso do mínimo de grau de certeza. Os resultados dos testes envolvendo o PBF2 que discutiremos aqui possuem

configurações de 0.8 de mínimo de grau de certeza e 30 de máximo de regras geradas. A razão de utilizar 0.8 de mínimo de grau de certeza é para poder comparar com os resultados da Seção 5.4 e a razão de utilizar 30 de máximo de regras geradas é porque estes refletem os melhores resultados obtidos. Podemos perceber que os resultados em relação a PBF e J48 são diferentes daqueles que foram apresentados na Seção 5.4, isto é, porque as bases que foram geradas possuem as mesmas características, mas com padrões muito mais complexos. O motivo de aumentar a complexidade das bases é para saber se os métodos podem ter boa performance, mesmo em bases mais complexas. A partir dos resultados apresentados na **Figura 29**, podemos concluir:

- Em quase todos os casos, o PBF2 teve um desempenho melhor que PBF em termos de TMCC. A mesma situação ocorre em comparação com o J48, que além de PBF2 ter TMCC próximo ou melhor que J48, ainda possui menos #Regras;
- Uma outra informação dos resultados não foi mostrada nas figuras devido a grande quantidade de informções, mas a quantidade das vezes em que a regra default foi utilizada durante o processo da classificação não passa de 10 vezes na maioria dos casos, ou nos casos em que somente foi gerada a regra default, o uso dela é 100% das vezes. Ou seja, podemos considerar que, para os casos que possuem mais regras, além da regra default, o TMCC que obtivemos não sofre tanta influência da regra default;
- Podemos perceber que quase em todos os casos, o uso da poda não é tão benéfico. Seu uso trouxe um aumento no #Regras, mas também uma perda grande na TMCC, algumas exceções são J48 em *Baseline*, Incerteza, RR e VI assim como o PBF em VI. Isso pode ser causado porque o conjunto de regras possui partes das regras que não são relevantes e, portanto, a poda trouxe um benefício, mas como isso parece variar bastante de caso para caso, neste momento podemos considerar como um elemento não constante;
- Em algumas bases, como a de Incerteza ou Ruidos, ambo PBF e PBF2 tiveram desempenhos ruins, sendo que apenas a regra default foi gerada. Mesmo o J48 não teve bons resultados, mas mais tarde mostraremos resultados comparativos nestas bases com diferentes mínimos de grau de certeza e análises sobre eles;

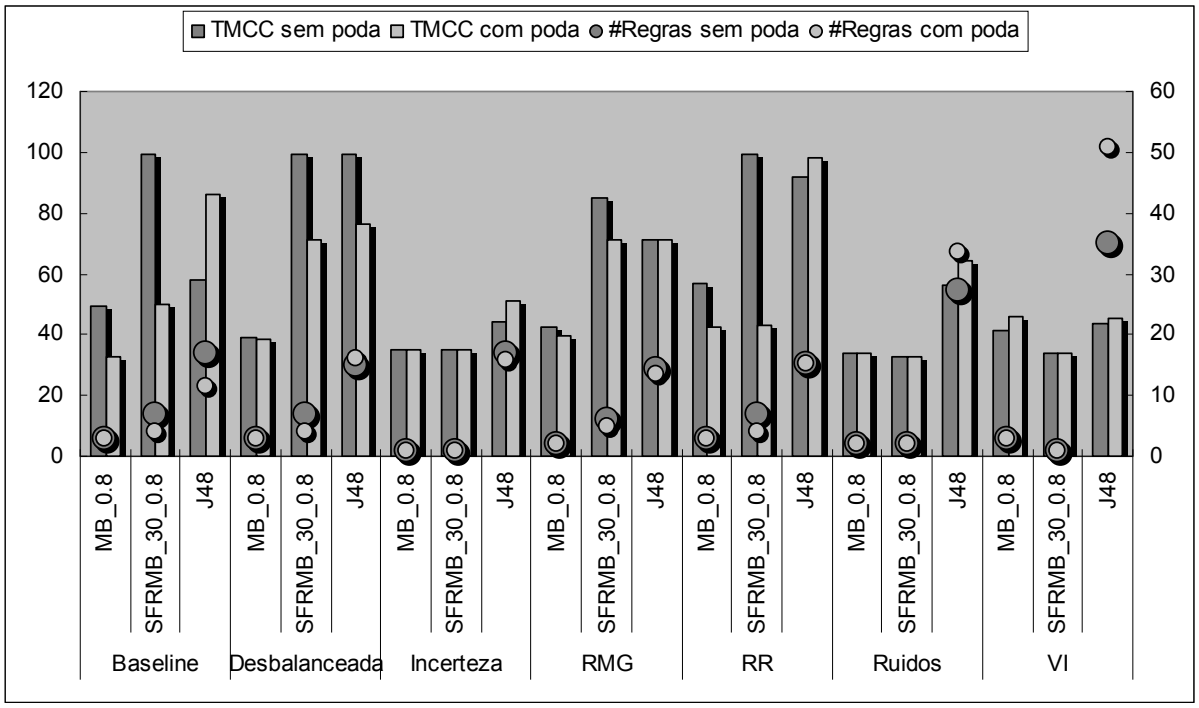

Figura 29 Resultados comparativos de TMCC e #Regras entre PBF, PBF2 e J48.

Nas **Figura 30** e **Figura 31**, há resultados envolvendo TMCC com #Ant. e #Regras com #Ant. respectivamente. Mas podemos tirar as conclusões semelhantes àquelas que tiramos com a **Figura 29**.

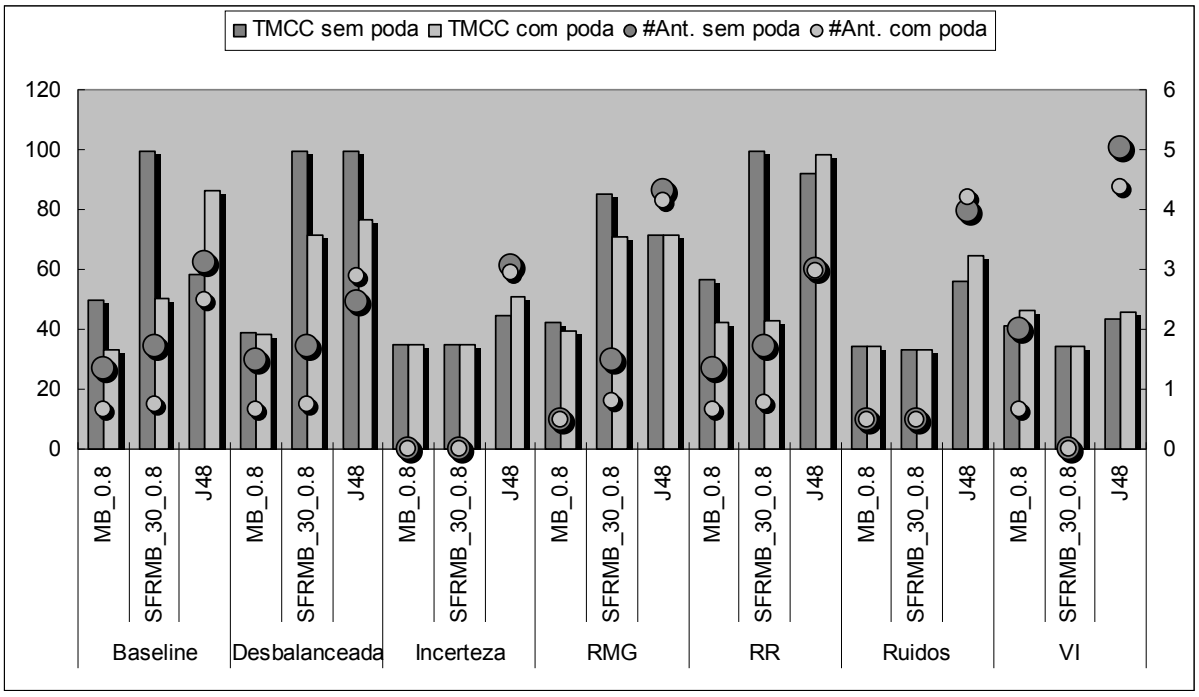

Figura 30 Resultados comparativos de TMCC e #Ant. entre PBF, PBF2 e J48.

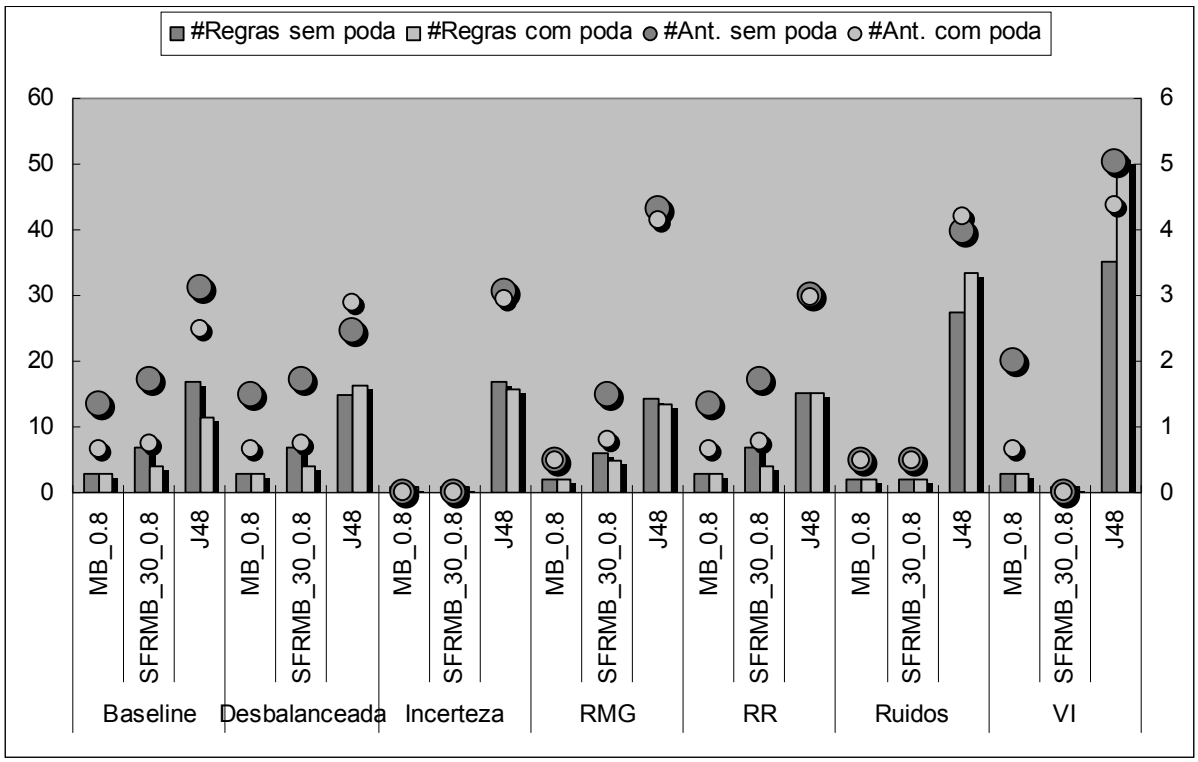

Figura 31 Resultados comparativos de #Regras e #Ant. entre PBF, PBF2 e J48.

Como mencionado anteriormente, o uso de diferentes graus de certeza possui novos comportamentos, por exemplo, na base de Incerteza, onde os algoritmos tiveram desempenho ruim, outros graus de certeza produziram comportamentos diferentes, como mostra a **Tabela 5**. A partir dela podemos concluir que:

- Quanto menor o mínimo de grau de certeza utilizada, mais regras serão geradas além de regras default;
- Na base de Incerteza, sabemos que sua característica é ter bastante informações conflitantes e incertas, o que pode fazer com que o grau de certeza associado a todas as regras seja mais baixo, ou seja, se diminuir o mínimo de grau de certeza, podemos extrair mais regras que podem ajudar a classificação, como mostra o resultado de SFRMB_0.6 em relação a SFRMB_0.7, por exemplo. A mesma situação ocorre com a base de Ruidos e VI.

  Portanto, podemos dizer que, em situações onde existe bastante incerteza na base, ou bastante ruído e variáveis irrelevantes, o mínimo de grau de certeza que devemos aplicar deve ser menor em relação a outros tipos de bases;

Tabela 5 Resultados com diferentes mínimo de graus de certeza.

Configuração	Poda	TMCC	#Regras	#Ant.
MB_0.6	Sim	34,4	1,1	0,27
	Não	33,15	1,5	0,33
MB_0.7	Sim	35,11	1	0
	Não	35,11	1	0
MB_0.8	Sim	35,11	1	0
	Não	35,11	1	0
MB_0.9	Sim	35,11	1	0
	Não	35,11	1	0
SFRMB_0.6	Sim	37,68	3,5	1,43
	Não	38,05	3,5	0,71
SFRMB_0.7	Sim	35,11	1	0
	Não	35,11	1	0
SFRMB_0.8	Sim	35,11	1	0
	Não	35,11	1	0
SFRMB_0.9	Sim	35,11	1	0
	Não	35,11	1	0

Uma diferença no mínimo de grau de certeza pode trazer um impacto nos resultados obtidos. Mas será que uma diferença no máximo de regras que são geradas, também traga um grande impacto? A **Figura 32** mostra um gráfico comparativo de TMCC com #Regras entres diferentes máximo de regras geradas (MaxReg) que o SFRMB utilizou. A partir destes resultados, podemos concluir que:

- Em todos os casos, o uso de MaxReg = 30 teve melhores TMCC e #Regras. Qualquer outro MaxReg utilizado teve resultados piores. Podemos dizer que, o MaxReg utilizado traz um grande impacto nos resultados obtidos, mas ainda não temos uma forma de definir a MaxReg ideal para ser utilizada;
- A escolha de MaxReg influencia não somente a TMCC, mas também o #Regras. Para MaxReg = 30 foram geradas mais regras, como mostra a base de *Baseline* da **Figura 34**. Para o teste de MaxReg = 50, ele tem aproximadamente 57% de #Regras em relação a MaxReg = 30 e aproximadamente 50% de TMCC em relação a MaxReg = 30. Podemos dizer que, com o uso de MaxReg = 30 houve um ganho um pouco maior em TMCC do que a perda em #Regras nesta análise simples;
- Uma possível solução rápida para descobrir o máximo ideal a ser aplicado com SFRMB é realizar testes empíricos para cada máximo que vai incluir

uma variável a mais. Por exemplo, se tivemos 4 variáveis binárias, os testes empíricos poderiam ser com máximos de 2, 4, 8 e 16 de máximo de regras geradas, ou seja, em cada um destes testes, temos exatamente uma variável a mais. Desta maneira, os teste empíricos, para descobrir o máximo ideal, são mais objetivos;

- A **Figura 33** e a **Figura 34** mostram os gráficos comparativos de TMCC com #Ant. e #Regras com #Ant.. Ambos conseguem chegar nas mesmas conclusões que tivemos nos itens anteriores, sendo que o #Ant. possui um comportamento bem semelhante ao #Regras nestes casos. O aumento de #Ant. do MaxReg = 30 em relação a MaxReg = 50 é de 112,5%, o aumento de #Regras é de 75% e o aumento de TMCC é de 100%. e então, podemos dizer que o ganho em TMCC compensa a perda em #Ant. e #Regras, porque além de TMCC ter atingido um valor quase ideal (100%), em termos absolutos, o créscimo de 3 regras fuzzy e 1 antecedente não parece piorar a compreensibilidade de forma drástica;

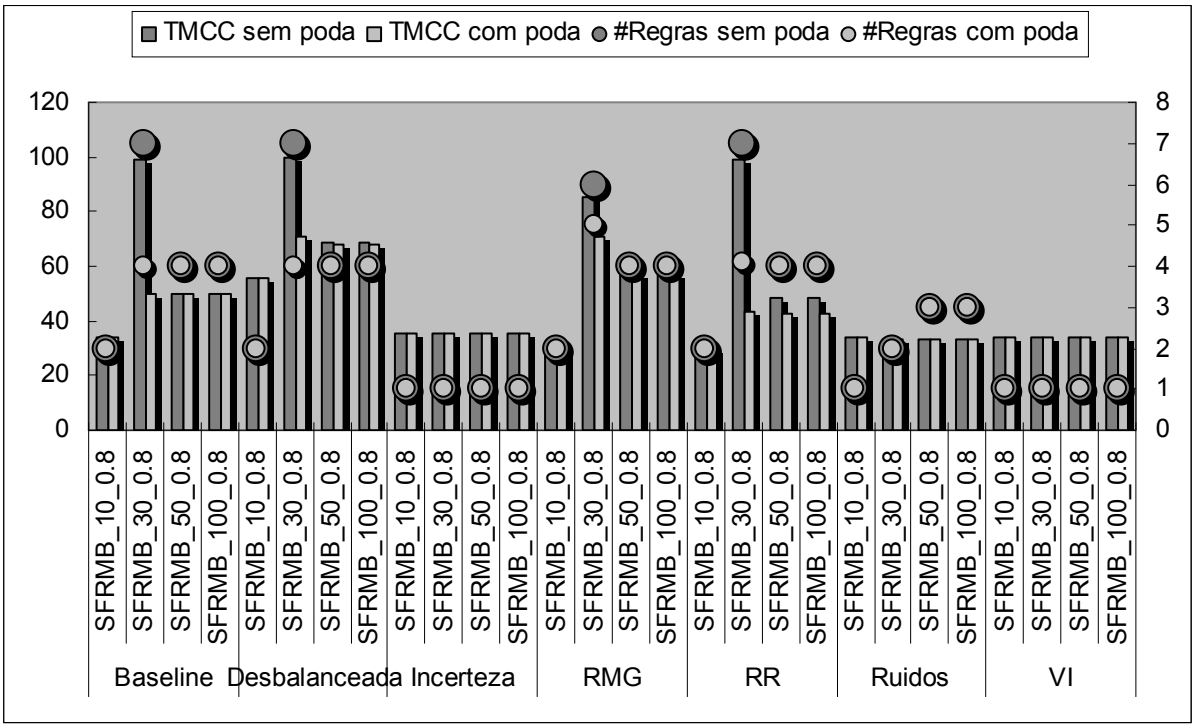

Figura 32 Resultado comparando o uso de diferentes máximo de regras no PBF2 (TMCC x #Regras).

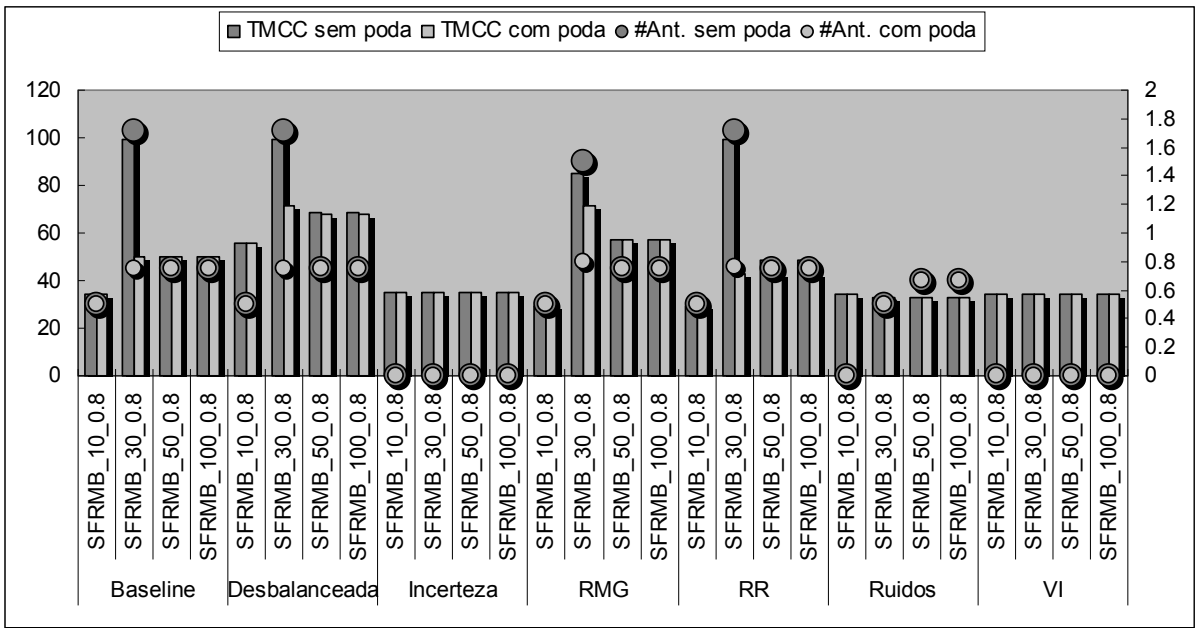

Figura 33 Resultado comparando o uso de diferentes máximo de regras no PBF2 (TMCC x #Ant.).

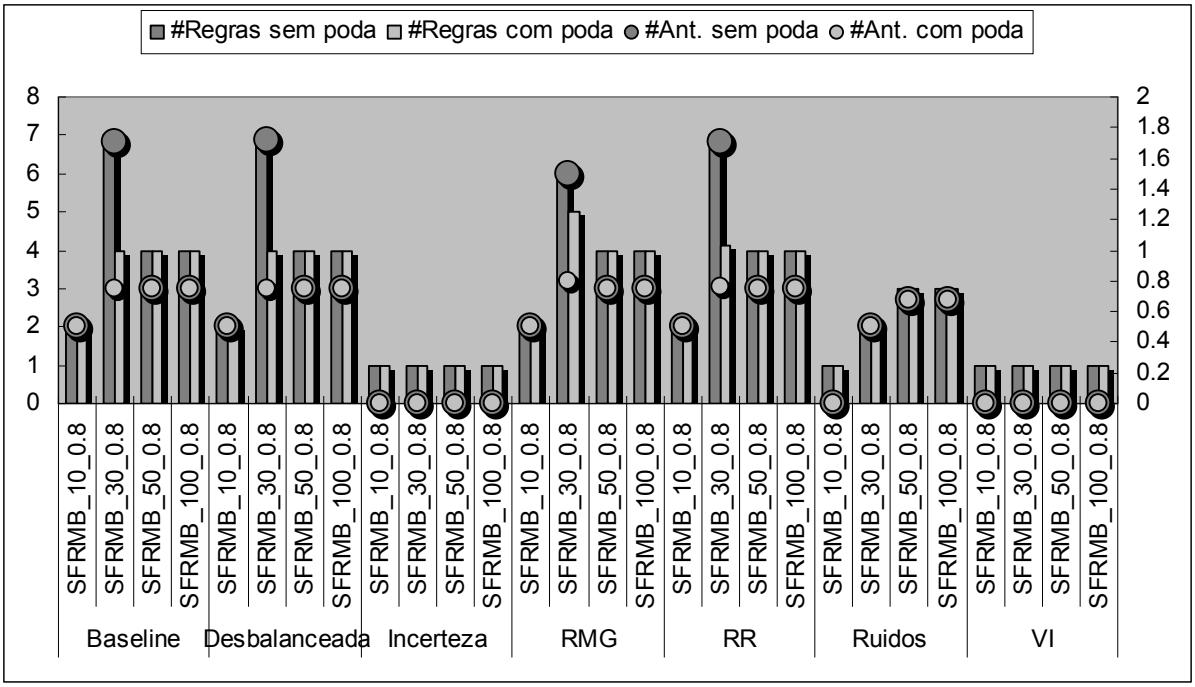

Figura 34 Resultado comparando o uso de diferentes máximo de regras no PBF2 (#Regras x #Ant.).

7. CONCLUSÕES

Como destacado no Capítulo 1, a precisão e a compreensibilidade dos resultados de um classificador são características importantes, quando este é utilizado como ferramenta de apoio à decisão. Assim, este trabalho de pesquisa investigou formas de explicação de um classificador Bayesiano utilizando regras de classificação *fuzzy*. Neste sentido, buscou-se identificar formas adequadas de se traduzir o conhecimento armazenado em uma rede Bayesiana para a forma de regras de classificação *fuzzy* e manter o balanço de precisão e compreensibilidade.

Como resultado desta investigação, foram propostos dois algoritmos, o *Pruned BayesFuzzy* (PBF) e o *Pruned BayesFuzzy2* (PBF2).

Os experimentos realizados permitem afirmar que:

- Em alguns casos, o mínimo grau de certeza pode trazer grandes diferenças e em outros casos, traz poucas diferenças. Porém, a presença do grau de certeza é ainda importante, pois o seu uso torna o método mais rápido;
- A adição da regra default na base de regras resolve um dos problemas levantados na Seção 5.2 sobre a incapacidade das regras resultantes classificarem certos padrões dos dados. Porém a classificação que a regra default realiza não é uma classificação real e dependerá muito dos dados de treinamento. Ou seja, é importante evitar o uso da regra default, porque ela pode não representar a real distribuição do problema;
- A incorporação da poda *Rule Post-Pruning* trouxe um benefício significativo para a compreensibilidade da rede Bayesiana gerada. Porém, ao mesmo tempo, diminuiu ainda mais a taxa de classificação correta, o que coloca em dúvida se o seu uso será benéfico ou não para o método;
- O *Markov Blanket,* aplicado neste problema, não fez uma seleção muito efetiva, então ainda há a possibilidade de melhora em relação à seleção de atributos. Também estamos dando preferência para a redução de antecedentes das regras do que a redução na quantidade de regras geradas, como foi mencionado na Seção 5.4. A redução de antecedentes das regras pode ser vista como uma forma de seleção de atributos também;
- Podemos perceber também que os conjuntos *fuzzy* utilizados na fuzzificação também trazem uma diferença nos resultados da classificação, então é possível considerar novas formas de fuzzificação e encontrar os conjuntos *fuzzy* que melhor representam os dados;
- O PBF conseguiu atingir o objetivo da pesquisa utilizando o algoritmo IC como algoritmo de treinamento e 0.8 de mínimo de grau de certeza, ou seja, o PBF conseguiu obter um balanceamento entre a classificação correta e a compreensibilidade, mesmo que ainda não seja um resultado com bastante

satisfação, mas já suficiente para termos a motivação para melhorar o PBF.

Já o PBF2, pode ser definido pelo Algoritmo 22. Após a inclusão de SFRMB, o PBF2 realmente teve uma melhora em relação ao PBF em termos de desempenho na classificação e também em termos de compreensibilidade, caso o máximo ideal seja aplicado. E, com base nos experimentos executados, podemos concluir que:

- O SFRMB conseguiu selecionar os atributos mais importantes para a classificação, mas precisa de uma quantidade certa de variáveis para que tenha uma boa classificação, ou seja, precisa ter um máximo ideal definido;
- O processo para descobrir o máximo, neste primeiro momento, através de uma forma ingênua, foi sugerido na Seção 6.2. Em outras palavras, deve-se incluir as variáveis mais fortes uma a uma de acordo com o rank de SFRMB até atingir o máximo ideal;
- Como mencionado na Seção 6.2, o mínimo de grau de certeza em algumas situações parece ter comportamentos bem interessantes e, definindo seu valor de acordo com a situação em que encontramos, pode melhorar a performance do método. Por enquanto, podemos afirmar que, nos casos em que houver incerteza, ruído e variáveis irrelevantes, o uso de um mínimo de grau de certeza menor parece trazer ganhos em relação aos mínimos mais altos;
- O não uso da poda *Rule Post-Pruning* parece ser uma melhor decisão, então ela será retirada do PBF2;
- PBF2 atingiu seu objetivo de melhorar a performance de classificação e compreensibilidade de PBF atingindo um balanceamento bom entre estes dois fatores.

Alguns trabalhos futuros que podem ser realizados são:

- Definir melhor como achar um mínimo de grau de certeza ideal;
- Definir uma melhor forma de definir o máximo de regras geradas usada em SFRMB;
- Aplicar métodos de agrupamento *fuzzy* para achar os conjuntos *fuzzy* ideais. Este trabalho implementou o *fuzzy C-Means* mas não teve testes suficientes para que possam ser reportados aqui.

Para incorporar o SFRMB no proceso de aprendizagem de redes Bayesianas, o PBF2, é preciso aprender uma rede Bayesiana completa primeiro para que possa iniciar o processo de SFRMB, porém isso nem sempre é possível. Alguns problemas demandam um esforço e recurso computacional muito grande ou chega a ser inviável aprender uma rede completa.

REFERÊNCIAS

[1] Arnold Neumaier. 2004. "Clouds, Fuzzy Sets, and Probability Intervals". Reliable Computing, vol. 10, number 4, pages 249-272. Netherland: Springer.

[2] Aliferis C.F., Tsamardinos T. and Statnikov A.. 2003. "HITON: a novel Markov blanket algorithm for optimal variable selection". AMIA Annual Symposium, pages 21-25.

[3] Chajweska U. and Draper D. L. 1998. “Explaining predictions in Bayesian networks and influence diagrams”. AAAI Spring Symposium series: Interactive and Mixed-Initiative Decision-Theoretic Systems, pages 23-32.

[4] Daphne Koller and Mehran Sahami. 1996. "Toward Optimal Feature Selection". International Conference on Machine Learning.

[5] Estevam Rafael Hruschka Jr. et al. 2007. “*BayesFuzzy : using a Bayesian Classifier to Induce a fuzzy Rule Base*”. Fuzzy Systems Conference, pages 1–6.

[6] Giulianella Coletti and Romano Scozzafava. 2006. "Conditional probability and fuzzy information". Computational Statistics & Data Analysis, volume 51, issue 1, pages 115-132.

[7] George J. Klir and Bo Yuan. 1995. "Fuzzy Sets and Fuzzy Logic Theory and Applications". New Jersey: Prentice Hall, 1st edition.

[8] Ghim-Eng Yap, Ah-Hwee Tan and Hwee-Hwa Pang. 2008. *“Explaining Inferences in Bayesian Networks*”, Applied Intelligence, vol. 29, pages 263-278.

[9] Gregory F. Cooper and Edward Herskovits. 1992. “*A Bayesian Method for the Induction of Probabilistic Networks from Data*”. Machine Learning, vol. 9, pages 309-347.

[10] H.J. Suermondt and Gregory F. Cooper. 1993. "An Evaluation of Explanations of Probabilistic Inference". Computers and Biomedical Research, vol. 26, issue 3, pages 242-254.

[11] Hisao Ishibuchi and Tomoharu Nakashima. 2001. "Effect of rule weights in fuzzy rule-based classification systems". IEEE Transactions on Fuzzy Systems, vol. 9, pages 506-515.

[12] Hossein Rahnama, Alireza Sadeghian and William Melek. 2006. "Fuzzy Bayesian models for Classification and Diagnosis in Generalized Cardiology Discipline". World Automation Congress, pages 1-7.

[13] Hanchuan Peng, Fuhui Long and Chris Ding. 2005. "Feature Selection Based on Mutual Information: Criteria of Max-Dependency, Max-Relevance, and Min-Redundancy". Pattern Analysis and Machine Intelligence, vol. 27, issue 8, pages 1226-1238.

[14] I-Hsien Yin, Estevam R. Hruschka Jr. and Heloisa de A. Camargo. 2009. "The

Impact of Pruning BayesFuzzy Rule Set". Intelligent System Design and Applications, pages 456-461.
[15] Imme Ebert-Uphoff. 2009. "A Probability-Based Approach to Soft Discretization for Bayesian Networks". Research Report.
[16] Isabelle Guyon and André Elisseeff. 2003. "An Introduction to Variable and Feature Selection". Journal of Machine Learning Research, vol. 3, pages 1157-1182.
[17] J. C. Bezdek. 1981. "Pattern recognition with fuzzy objective function algorithms". Norwell: Kluwer Academic.
[18] J. R. Quinlan. 1996. "Improved Use of Continuous Attributes in C4.5". Journal of Artificial Intelligent Research 4, pages 77-90.
[19] J. R. Quinlan. 1993. "C4.5: Programs for Machine Learning". San Mateo, Calif.: Morgan Kaufmann.
[20] Jan van den Berg, Uzay Kaymak and Williem-Max van den Bergh. 2002. "Fuzzy Classification Using Probability-Based Rule Weighting". Fuzzy Systems, vol. 2, pages 991-996.
[21] Jim F. Baldwin and Enza Di Tomaso. 2003. "Inference and Learning in Fuzzy Bayesian Networks". Fuzzy Systems, vol. 1, pages 630-635.
[22] Judea Pearl and T.S. Verma. 1991. "A theory of inferred causation". Knowledge Representation, pages 441-452.
[23] Jung Hun Oh, et. al. 2009. "An Extended Markov Blanket Approach to Proteomic Biomarker Detection From High-Resolution Mass Spectrometry Data". Information Technology in Biomedicine, vol. 13, issue 2, pages 195-206.
[24] Krishnapuram, B. et. al. 2004. "A Bayesian approach to joint feature selection and classifier design". Pattern Analysis and Machine Intelligence, vol. 26, issue 9, pages 1105-1111.
[25] M. E. Cintra, H. A. Camargo and M. C. Monard. 2008. "A study on techniques for the automatic generation of membership functions for pattern recognition". C3N - Congresso da Academia Trinacional de Ciências, issue 1, pages 1-10.
[26] Marcelo Andrade Teixeira and Gerson Zaverucha. 2002. "Fuzzy Markov Predictor in Electric Load Forecasting". Neural Networks, vol. 3, pages 2416-2421.
[27] Mathias Johansson and Tomas Olofsson. 2007. "Bayesian Model Selection for Markov, Hidden Markov, and Multinomial Models". Signal Processing Letters, vol. 14, issue 2, pages 129-132.
[28] Neapolitan R.E. 2003. "Learning Bayesian Networks". New York: Prentice Hall.
[29] Pang-Ning Tan, Vipin Kumar and Jaideep Srivastava. 2002. “Selecting the Right Interestingness Measure for Association Patterns”. Knowledge Discovery and Data Mining, pages 32-41.
[30] Pearl, J. 1986. "Fusion, Propagation, and Structuring in Belief Networks". Artificial

Intelligence, vol. 29, issue 3, pages 241-288.
[31] Shewchuk, J.R. 1994. "An Introduction to the Conjugate Gradient Method Without the Agonizing Pain". Technical Report CMU-CS-94-125, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania.
[32] Shixin Yu and Paul Scheunders. 2001. "Fuzzy Markov Chains Approach to Feature Selection for High Dimensional Remote Sensing Data". Geoscience and Remote Sensing Symposium, vol. 7, pages 3306-3308.
[33] Shuangcheng Wang, Cuiping Leng and Ruijie Du. 2009. " Feature Subset Selection Based on Bayesian networks". Fuzzy Systems and Knowledge Discovery, vol. 1, pages 184-187.
[34] Suermondt, H. J. and Cooper, G. F. 1991. "Initialization for the Method of Conditioning in Bayesian Belief Networks". Artificial Intelligence, vol. 50, issue 1, pages 83-94.
[35] Te-Shun Chou, et. al. 2007. "Correlation-Based Feature Selection for Intrusion Detection Design". Military Communications Conference, pages 1-7.
[36] Tom M. Mitchell. 1997. "Machine Learning". McGraw Hill, 1st edition.
[37] Urzula Chajewska and Joseph Y. Halpern. 1997. "Defining Explanation in Probabilistic Systems". Uncertainty in Artificial Intelligence, pages 62-71.
[38] Verma T. and Pearl J. 1990. “Equivalance and Synthesis of Causal Models”. Sixth Conference in Artificial Intelligence, pages 220-227.
[39] Wang, L.X. and Mendel J.M.. 1992. "Generating fuzzy rules by learning from example". IEEE Trans. on Systems, Man and Cybernetics, vol. 22, issue 6, pages 1414-1427.
[40] Xue Bai, et al. 2004. "PCX: Markov Blanket Classification for Large Data Sets with Few Cases". Technical Report CMU-CALD-04-102.
[41] Y. Yang and G. Webb. 2002. "A comparative study of discretization methods for naive-Bayes classifiers". In Proceedings of the Pacific Rim Knowledge Acquisition Workshop, pages 159-173.
[42] Yongchuan Tang and Yang Xu. 2005. "Application of fuzzy Naive Bayes and a real-valued genetic algorithm in identification of fuzzy model". Information Sciences, vol. 169, issues 3-4, pages 205-226.
[43] Yongchuan Tang, Wuming Pan, Haiming Li and Yang Xu. 2002. "Fuzzy Naive Bayes Classifier Based on Fuzzy Clustering". System, Man and Cybernetics, vol. 5, 6 page.
[44] Zakaria Nouir, et al. 2007. "Use of Fuzzy Bayesian Clustering to Enhance Generalization Capacity of Radio Network Planning Tool". International Conference on Communications, pages 338-343.

APÊNDICES

Apêndice 1. Resultados detalhados com a introdução do mínimo grau de certeza

Os seguintes resultados foram obtidos pelos testes realizados sem a aplicação do *Markov Blanket*.

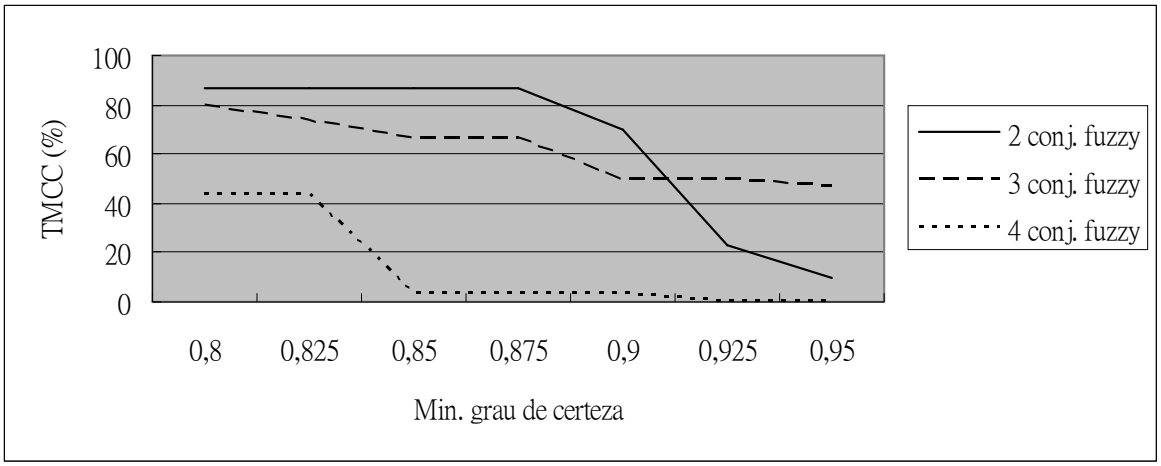

Figura 35 Taxa Média de Classificação Correta (TMCC) de teste sobre o uso de mínimo grau de certeza. Teste considerando 2 variáveis relevantes e 0 variável irrelevante no conjunto de dados de treinamento.

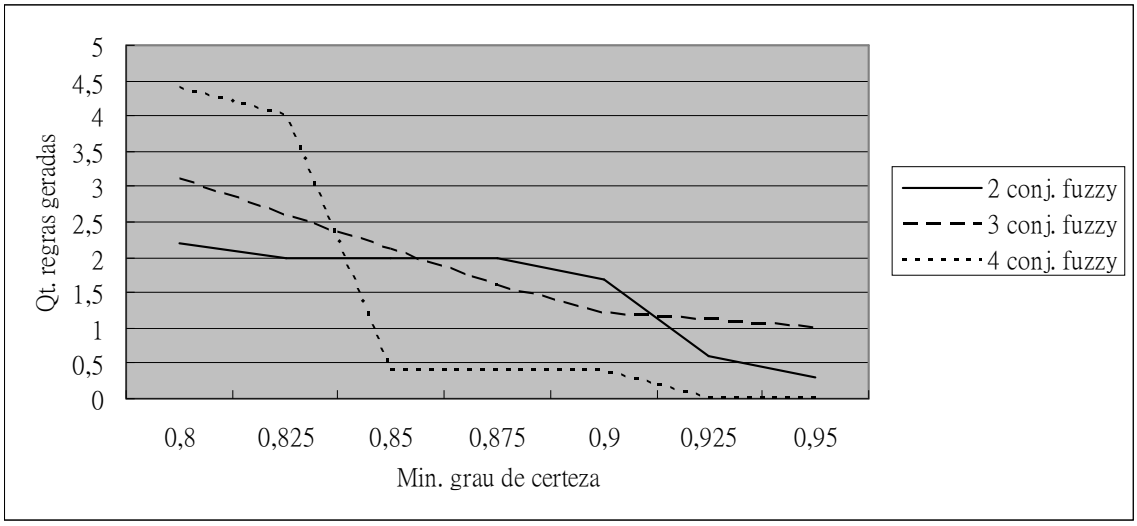

Figura 36 Quantidade de regras geradas de teste sobre o uso de mínimo grau de certeza. Teste considerando 2 variáveis relevantes e 0 variável irrelevante no conjunto de dados de treinamento.

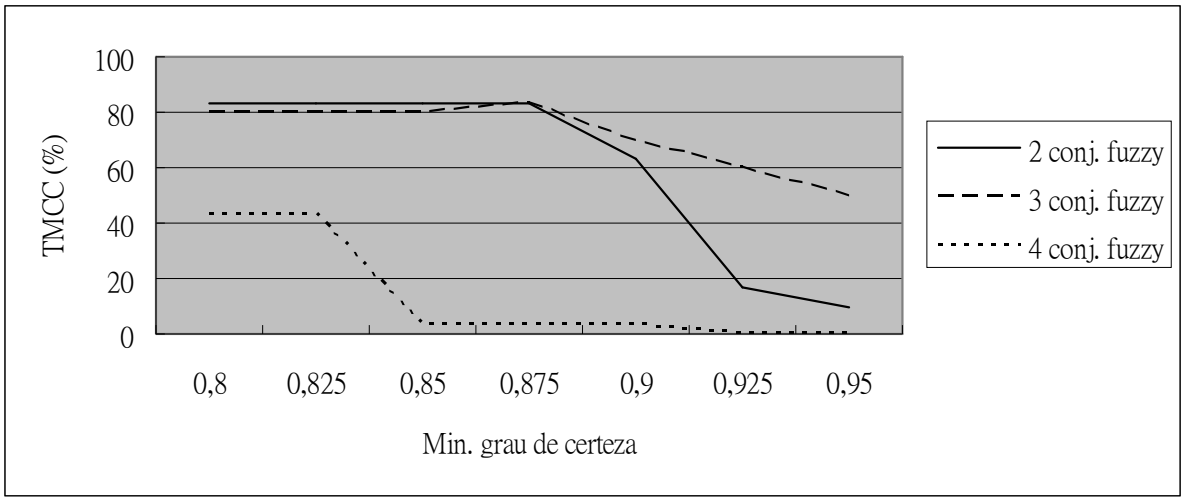

Figura 37 Taxa Média de Classificação Correta (TMCC) de teste sobre o uso de mínimo grau de certeza. Teste considerando 2 variáveis relevantes e 1 variável irrelevante no conjunto de dados de treinamento.

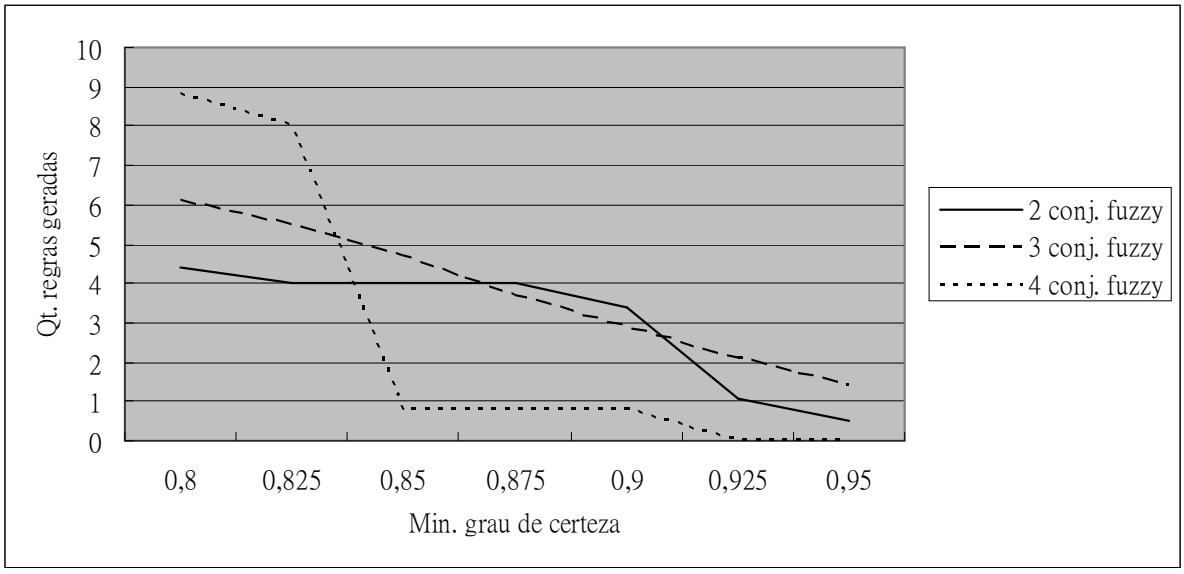

Figura 38 Quantidade de regras geradas de teste sobre o uso de mínimo grau de certeza. Teste considerando 2 variáveis relevantes e 1 variável irrelevante no conjunto de dados de treinamento.

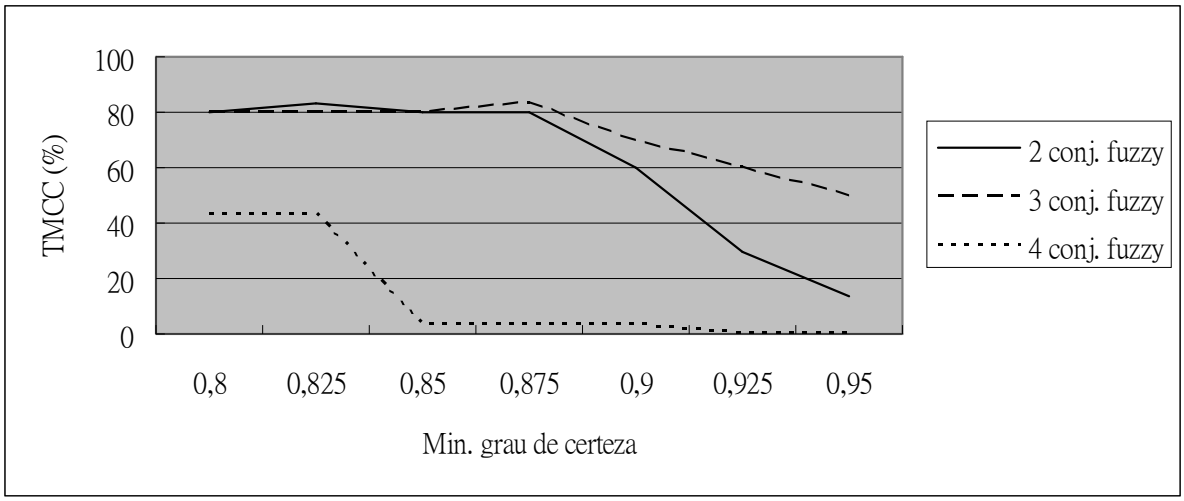

Figura 39 Taxa Média de Classificação Correta (TMCC) de teste sobre p uso de mínimo grau de certeza. Teste considerando 2 variáveis relevantes e 2 variáveis irrelevantes no conjunto de dados de treinamento.

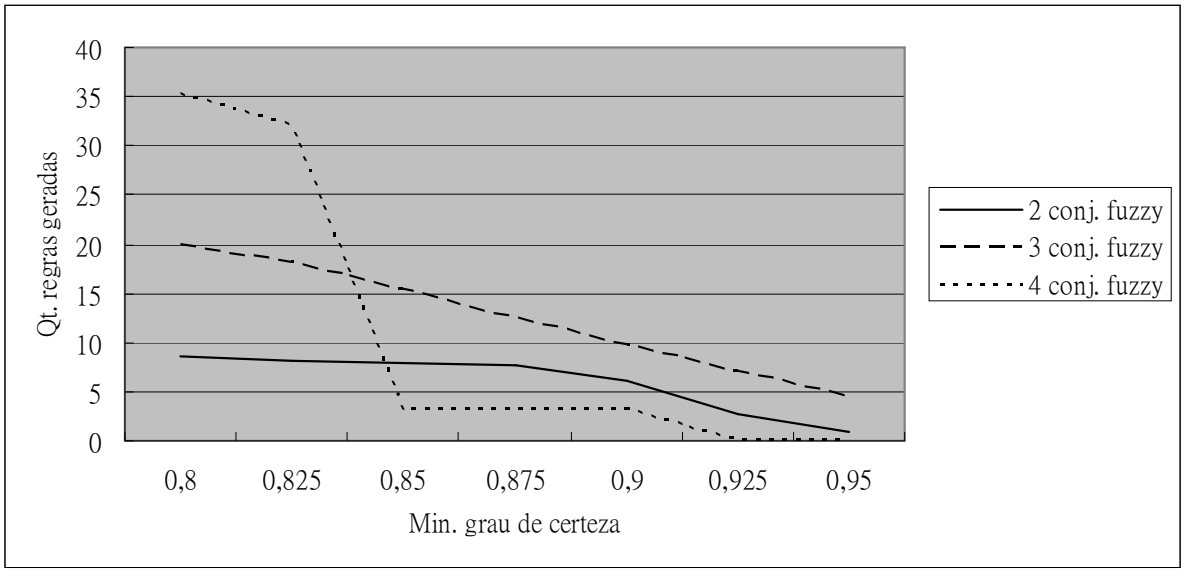

Figura 40 Quantidade de regras geradas de teste sobre o uso de mínimo grau de certeza. Teste considerando 2 variáveis relevantes e 2 variáveis irrelevantes no conjunto de dados de treinamento.

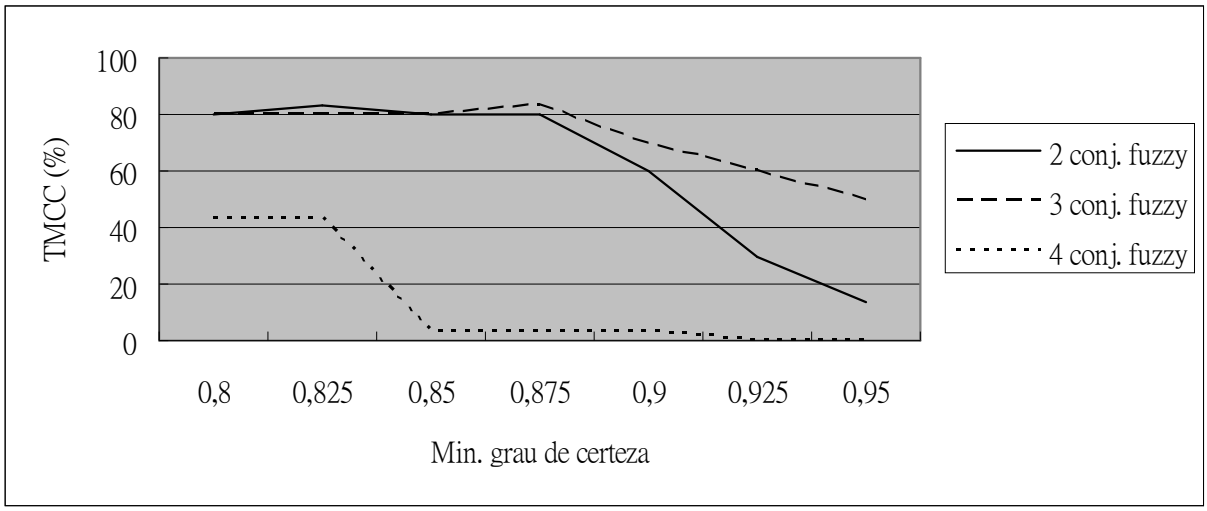

Figura 41 Taxa Média de Classificação Correta (TMCC) de teste sobre o uso de mínimo grau de certeza. Teste considerando 2 variáveis relevantes e 3 variáveis irrelevantes no conjunto de dados de treinamento

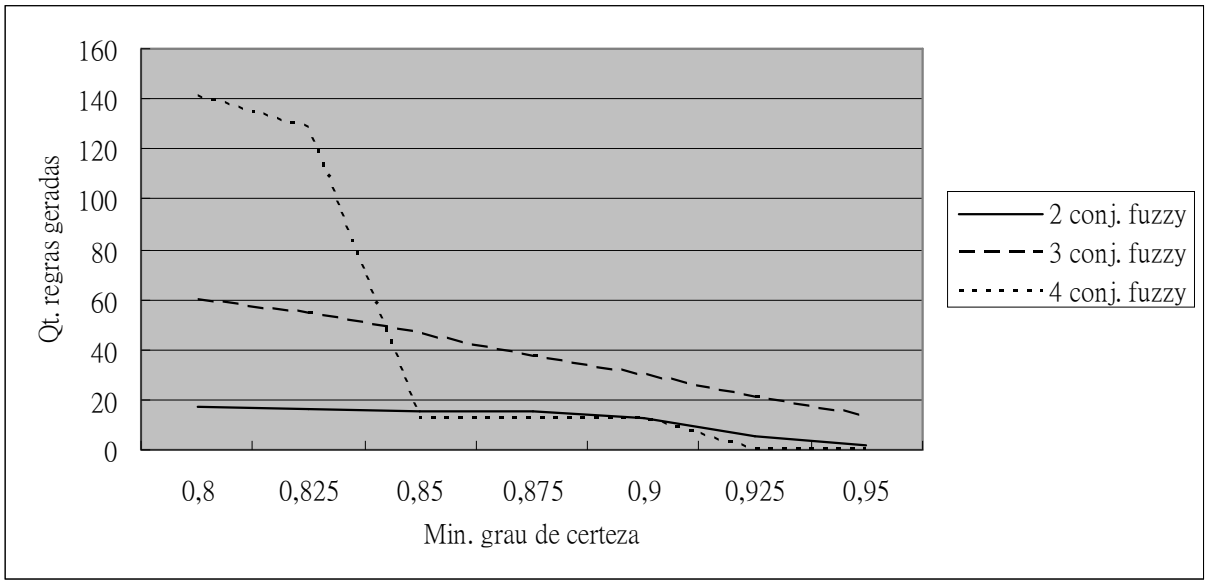

Figura 42 Quantidade de regras geradas de teste sobre o uso de mínimo grau de certeza. Teste considerando 2 variáveis relevantes e 3 variáveis irrelevantes no conjunto de dados de treinamento.

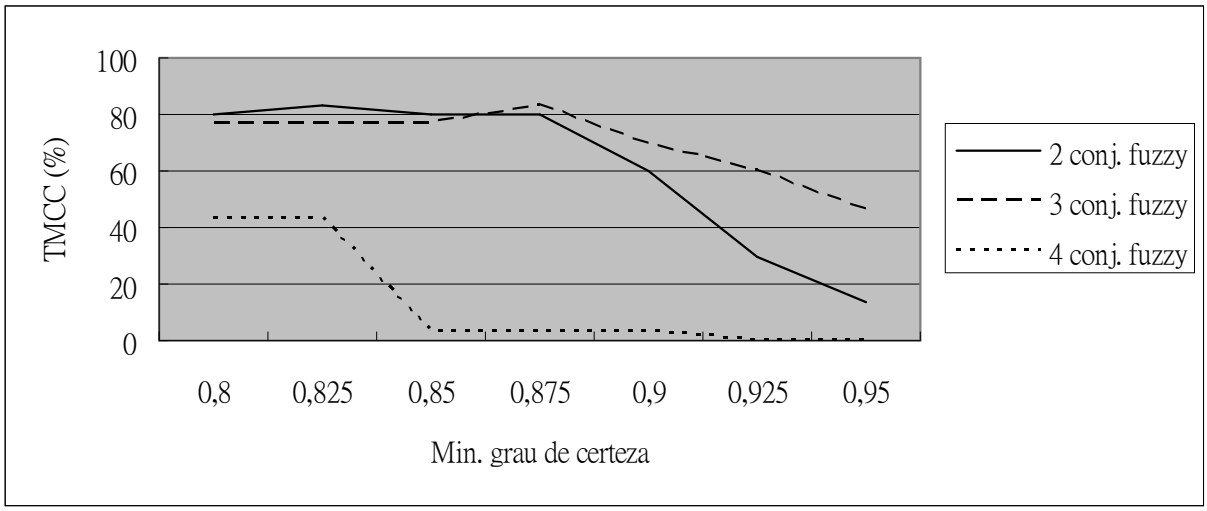

Figura 43 Taxa Média de Classificação Correta (TMCC) de teste sobre o uso de mínimo grau de certeza. Teste considerando 2 variáveis relevantes e 4 variáveis irrelevantes no conjunto de dados de treinamento.

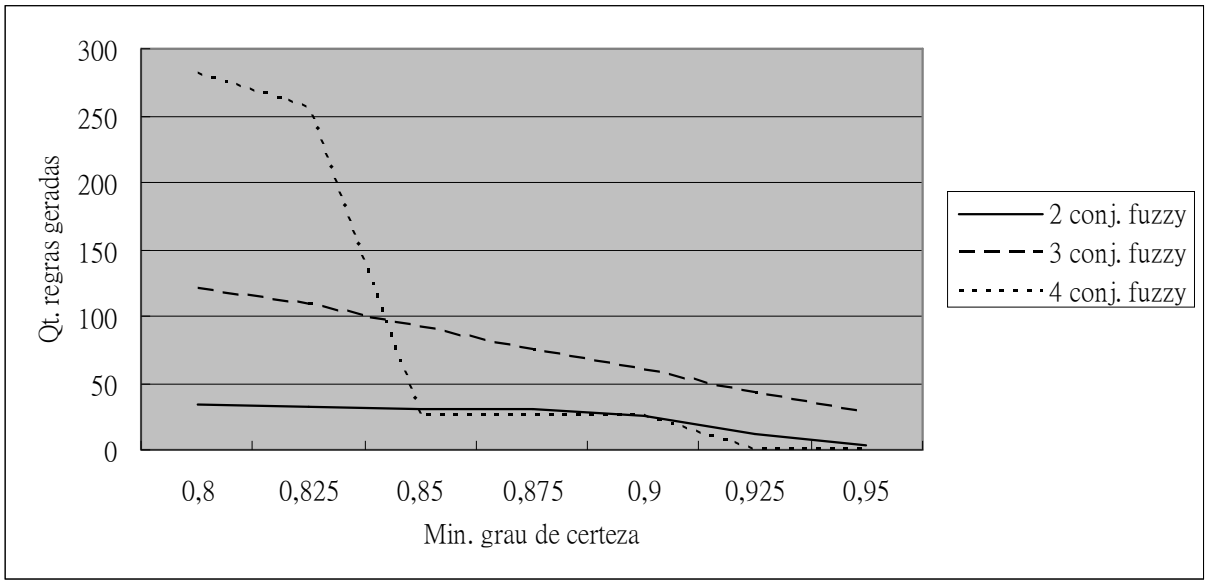

Figura 44 Quantidade de regras geradas de teste sobre o uso de mínimo grau de certeza. Teste considerando 2 variáveis relevantes e 4 variáveis irrelevantes no conjunto de dados de treinamento.

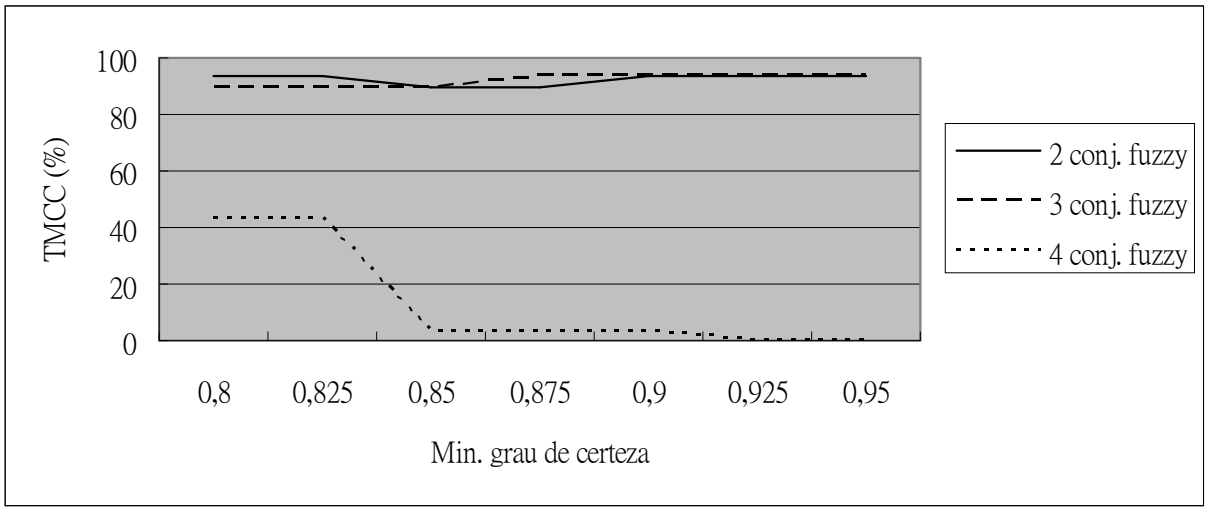

Figura 45 Taxa Média de Classificação Correta (TMCC) de teste sobre o uso de mínimo grau de certeza. Teste considerando 3 variáveis relevantes e 0 variável irrelevante no conjunto de dados de treinamento.

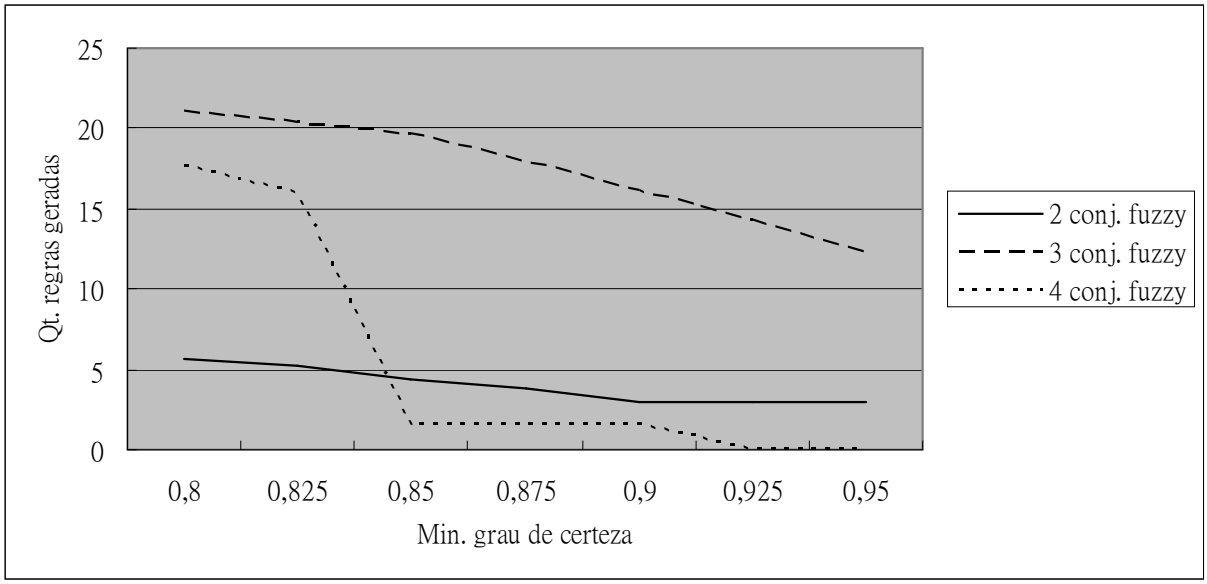

Figura 46 Quantidade de regras geradas de teste sobre o uso de mínimo grau de certeza. Teste considerando 3 variáveis relevantes e 0 variável irrelevante no conjunto de dados de treinamento.

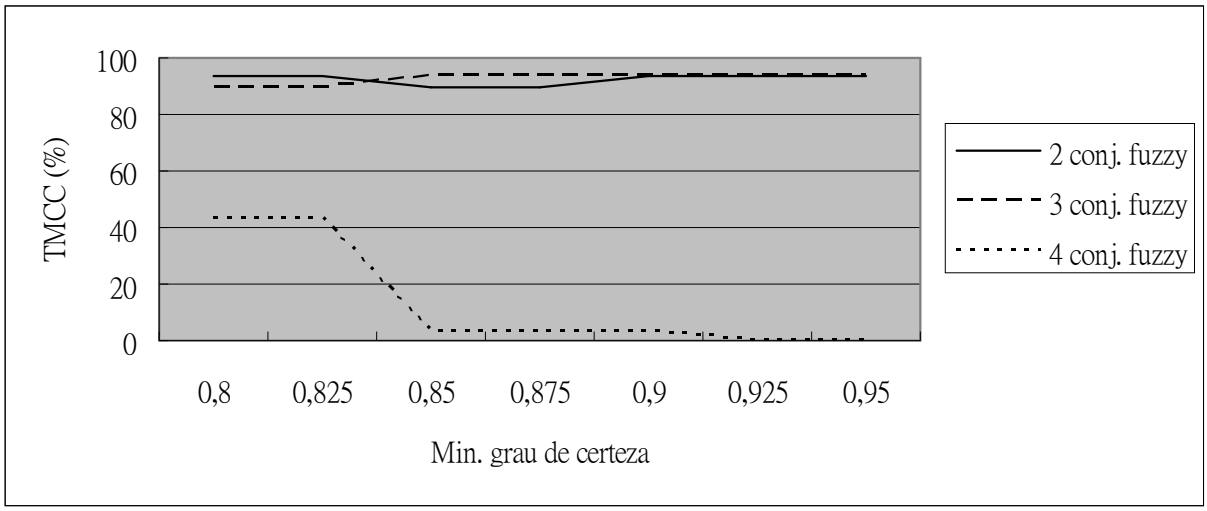

Figura 47 Taxa Média de Classificação Correta (TMCC) de teste sobre o uso de mínimo grau de certeza. Teste considerando 3 variáveis relevantes e 1 variável irrelevante no conjunto de dados de treinamento.

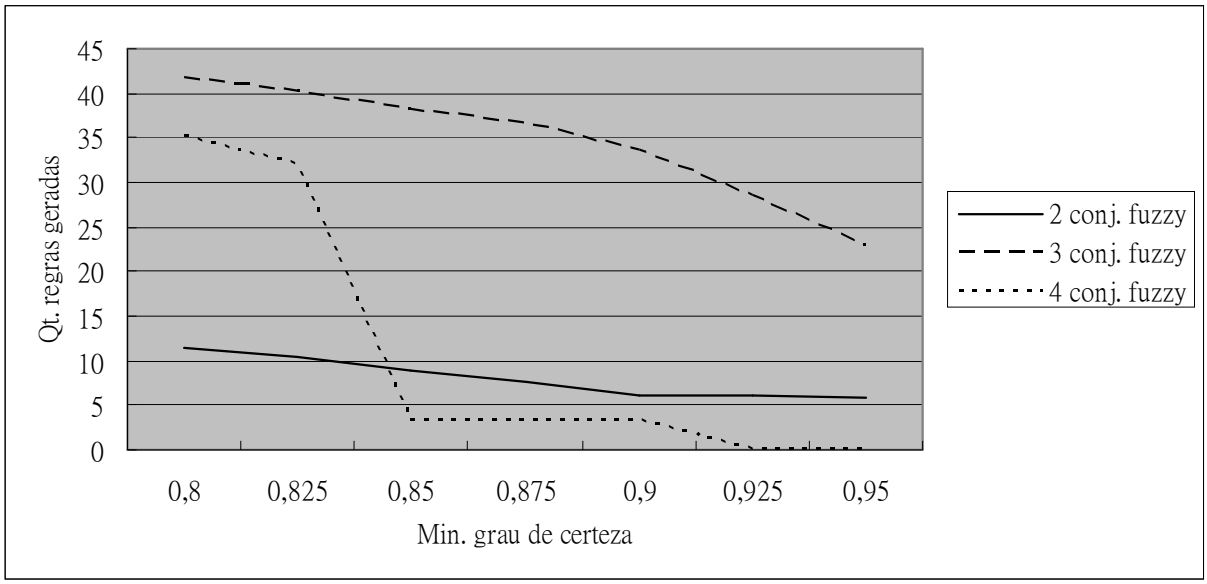

Figura 48 Quantidade de regras geradas de teste sobre o uso de mínimo grau de certeza. Teste considerando 3 variáveis relevantes e 1 variável irrelevante no conjunto de dados de treinamento.

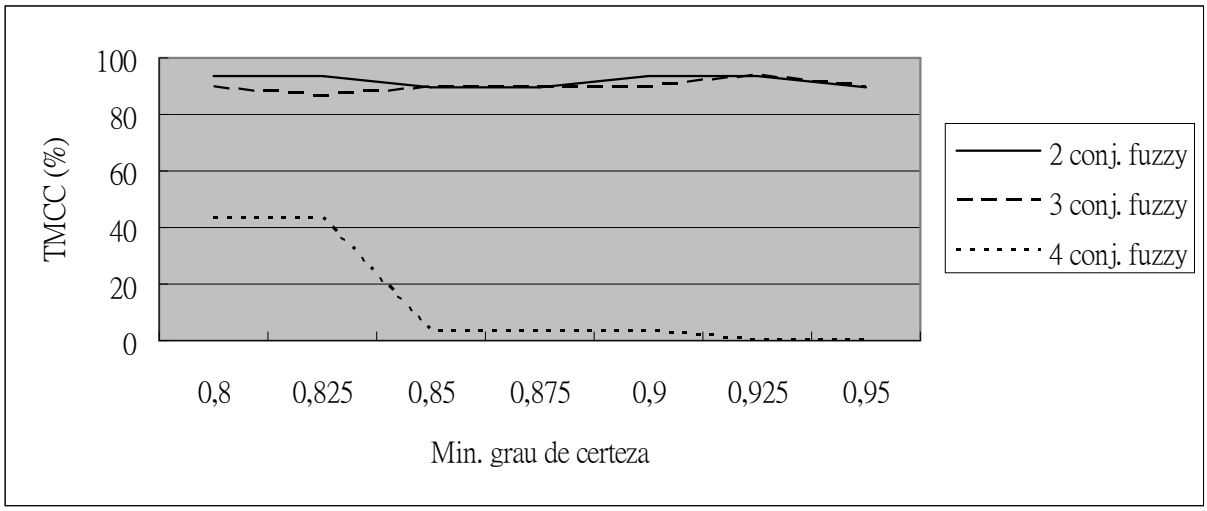

Figura 49 Taxa Média de Classificação Correta (TMCC) de teste sobre o uso de mínimo grau de certeza. Teste considerando 3 variáveis relevantes e 2 variáveis irrelevantes no conjunto de dados de treinamento.

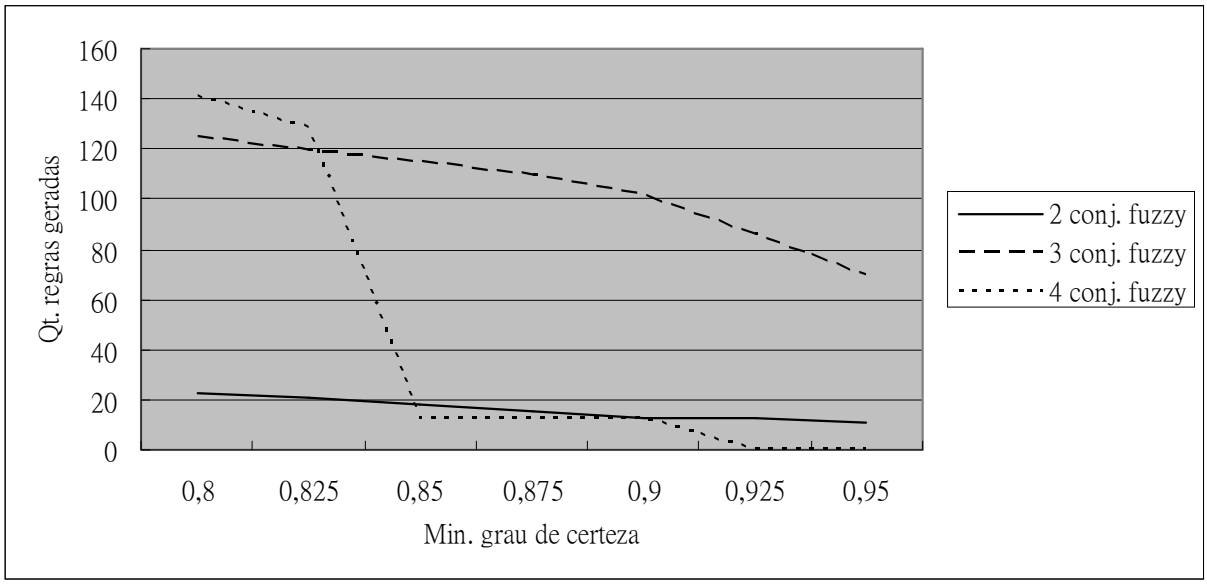

Figura 50 Quantidade de regras geradas de teste sobre o uso de mínimo grau de certeza. Teste considerando 3 variáveis relevantes e 2 variáveis irrelevantes no conjunto de dados de treinamento.

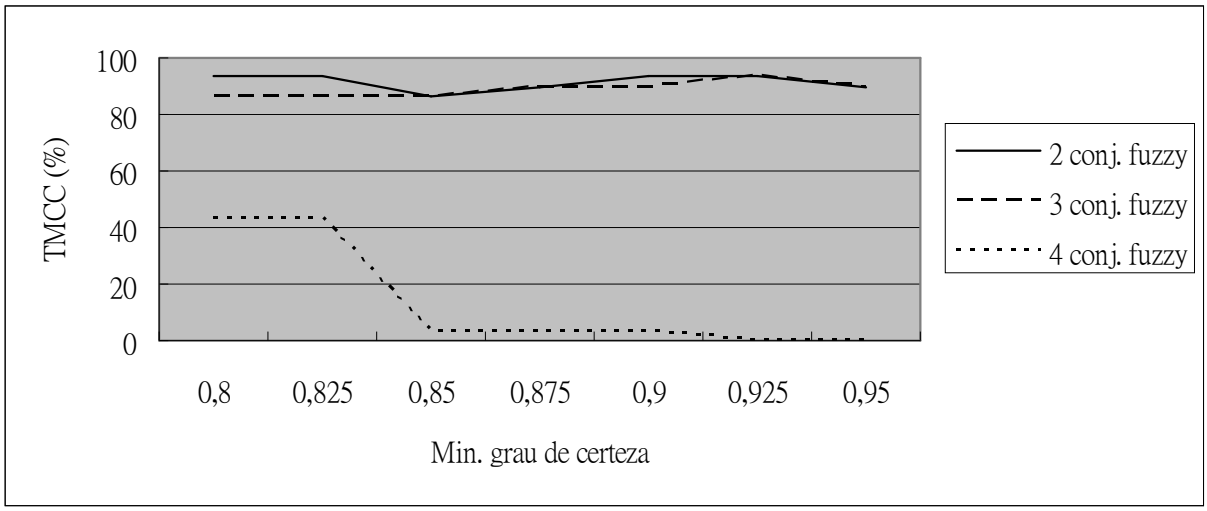

Figura 51 Taxa Média de Classificação Correta (TMCC) de teste sobre o uso de mínimo grau de certeza. Teste considerando 3 variáveis relevantes e 3 variáveis irrelevantes no conjunto de dados de treinamento.

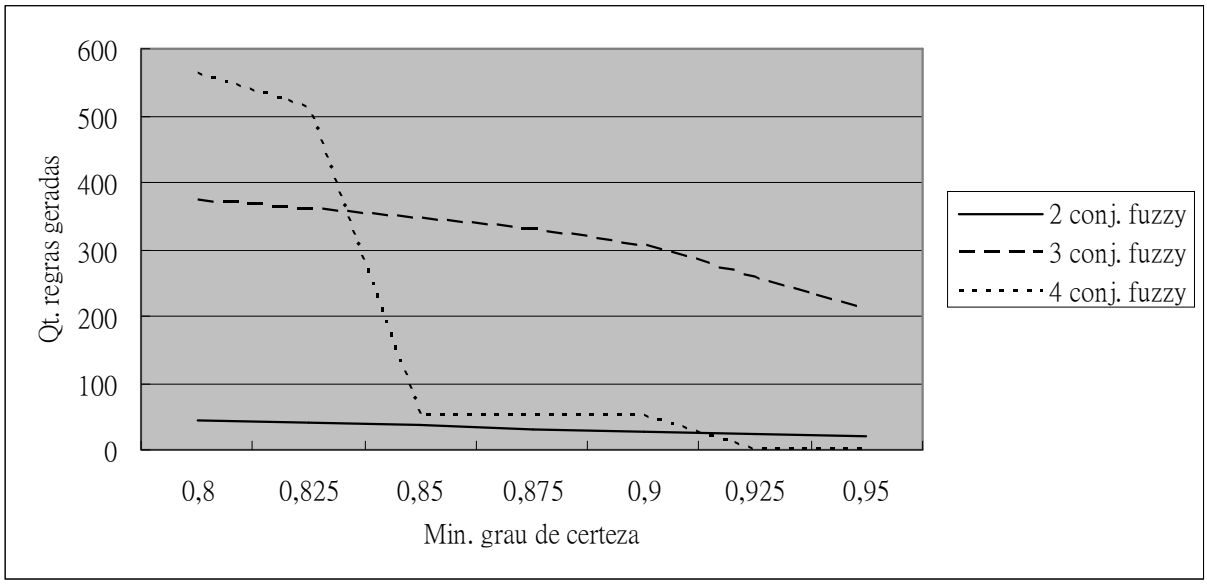

Figura 52 Quantidade de regras geradas de teste sobre o uso de mínimo grau de certeza. Teste considerando 3 variáveis relevantes e 3 variáveis irrelevantes no conjunto de dados de treinamento.

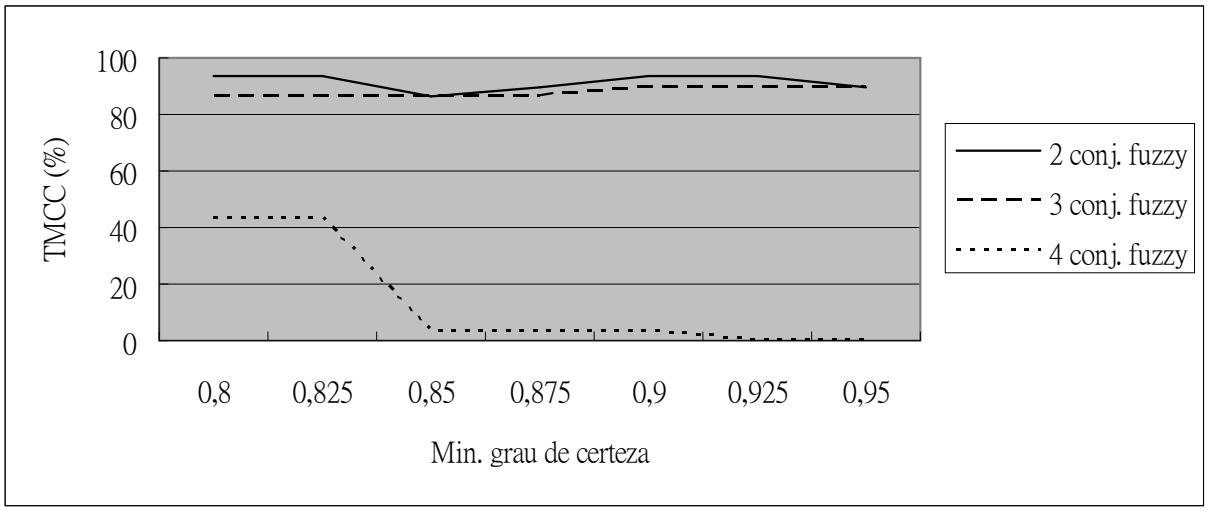

Figura 53 Taxa Média de Classificação Correta (TMCC) de teste sobre o uso de mínimo grau de certeza. Teste considerando 3 variáveis relevantes e 4 variáveis irrelevantes no conjunto de dados de treinamento.

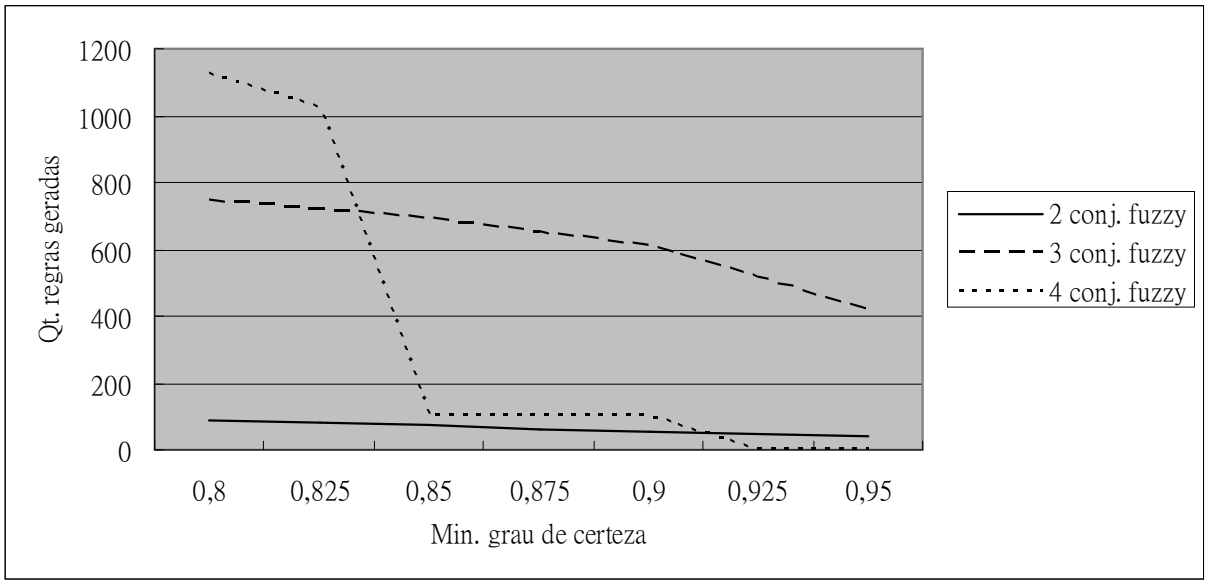

Figura 54 Quantidade de regras geradas de teste sobre o uso de mínimo grau de certeza. Teste considerando 3 variáveis relevantes e 4 variáveis irrelevantes no conjunto de dados de treinamento.

Os seguintes resultados foram obtidos pelos testes realizados com a aplicação do *Markov Blanket*.

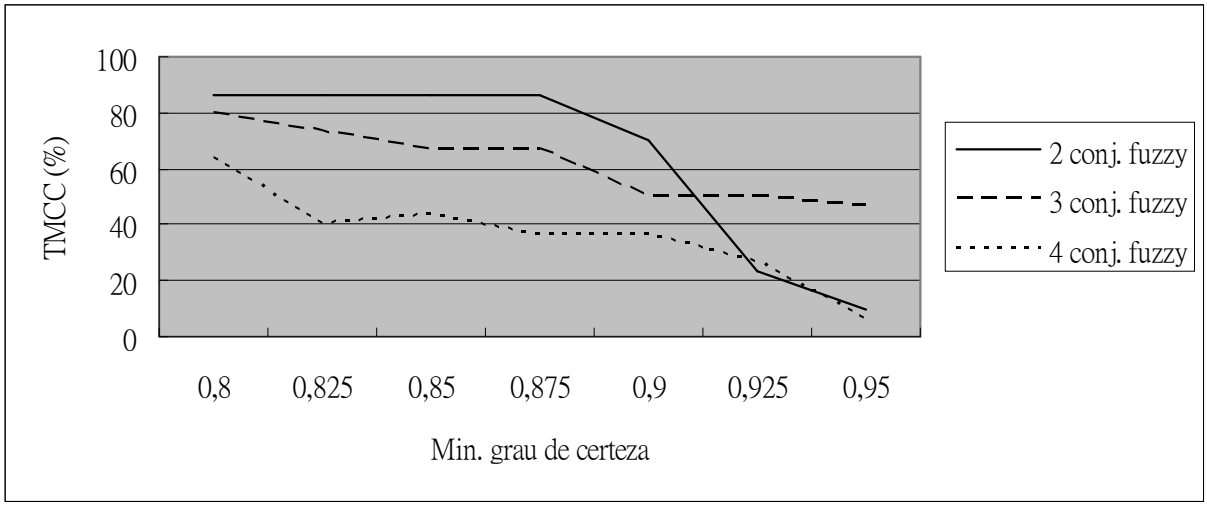

Figura 55 Taxa Média de Classificação Correta (TMCC) de teste sobre o uso de mínimo grau de certeza. Teste considerando 2 variáveis relevantes e 0 variável irrelevante no conjunto de dados de treinamento.

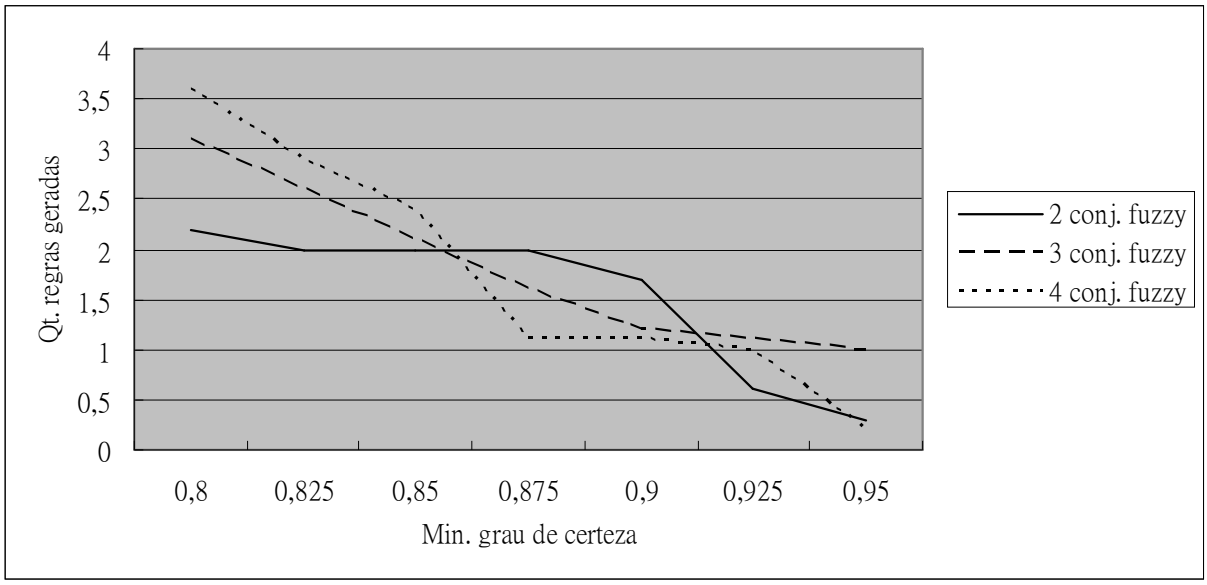

Figura 56 Quantidade de regras geradas de teste sobre o uso de mínimo grau de certeza. Teste considerando 2 variáveis relevantes e 0 variável irrelevante no conjunto de dados de treinamento.

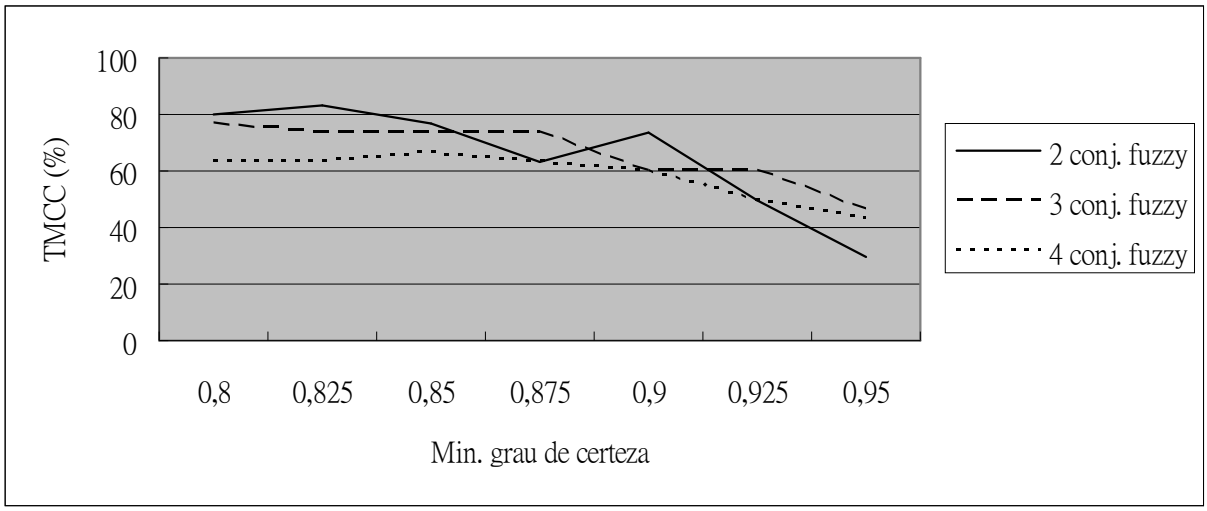

Figura 57 Taxa Média de Classificação Correta (TMCC) de teste sobre o uso de mínimo grau de certeza. Teste considerando 2 variáveis relevantes e 1 variável irrelevante no conjunto de dados de treinamento.

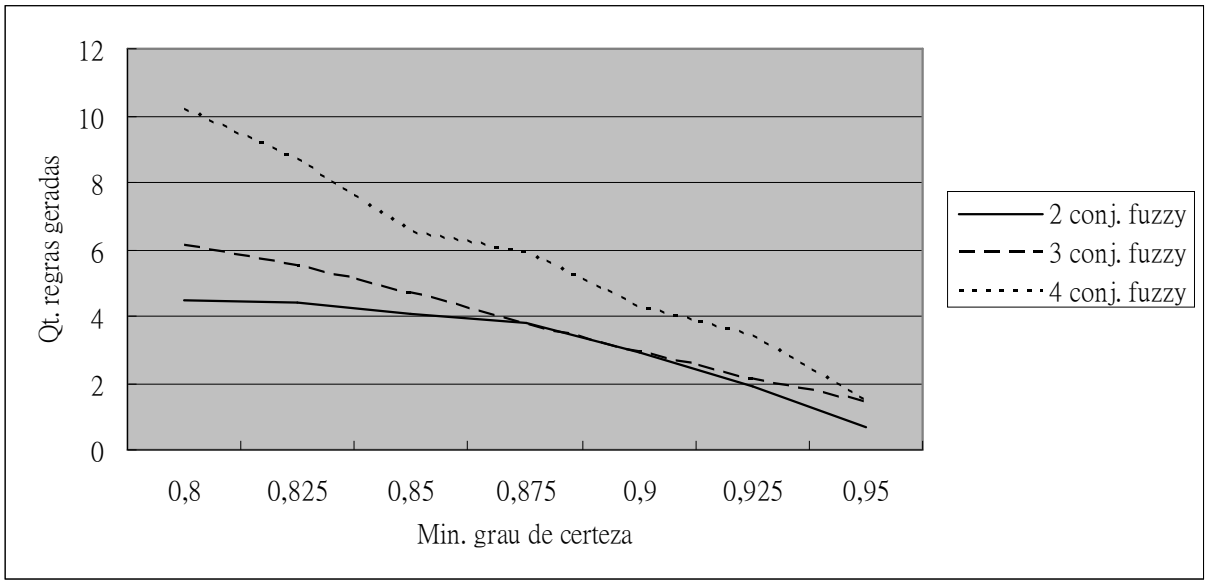

Figura 58 Quantidade de regras geradas de teste sobre o uso de mínimo grau de certeza. Teste considerando 2 variáveis relevantes e 1 variável irrelevante no conjunto de dados de treinamento.

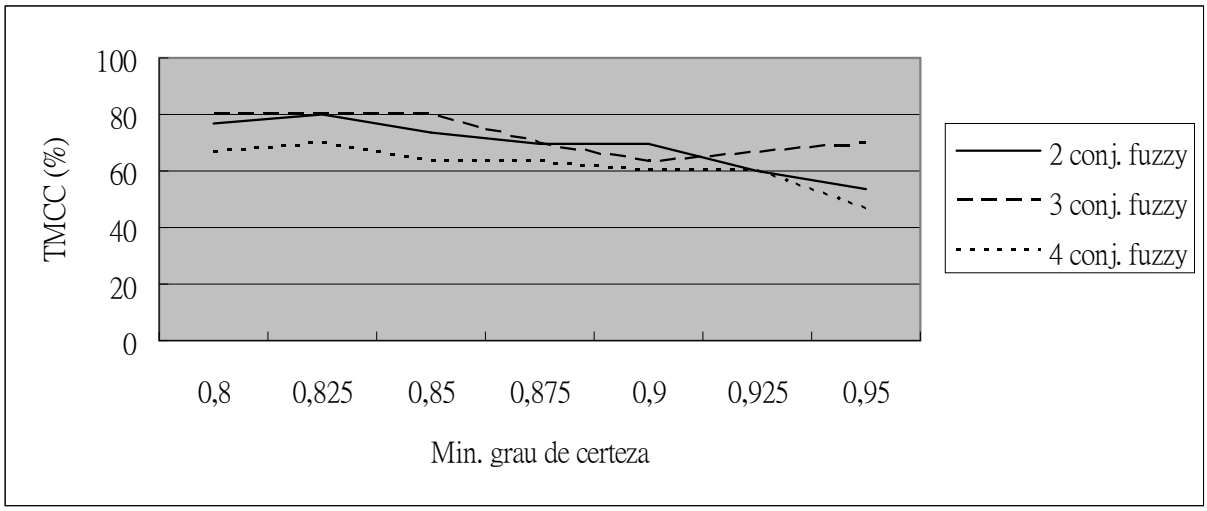

Figura 59 Taxa Média de Classificação Correta (TMCC) de teste sobre o uso de mínimo grau de certeza. Teste considerando 2 variáveis relevantes e 2 variáveis irrelevantes no conjunto de dados de treinamento.

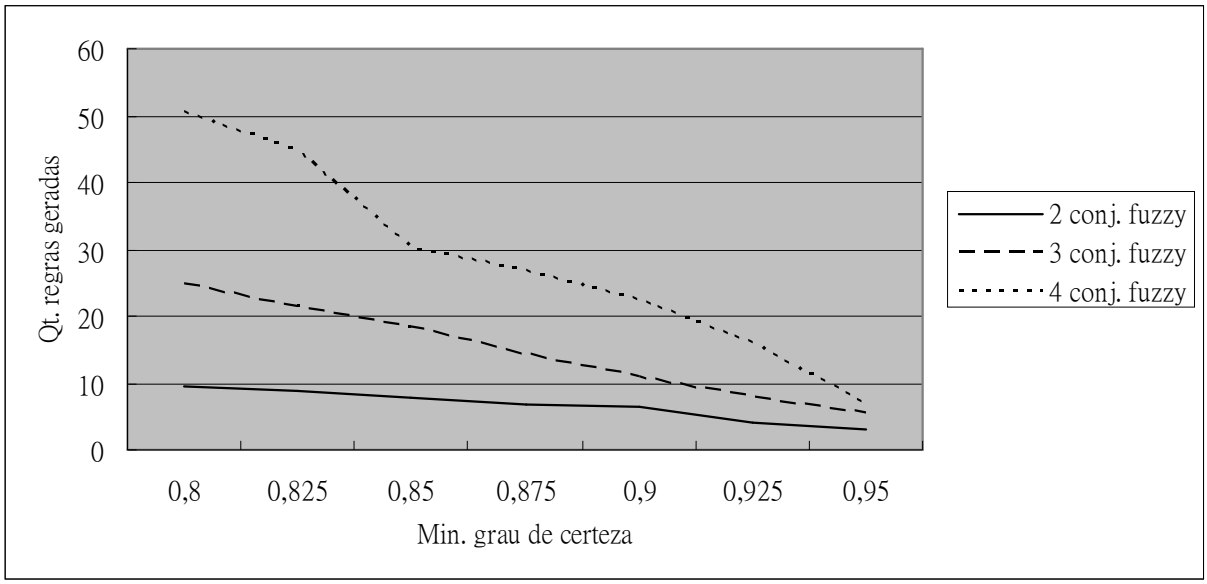

Figura 60 Quantidade de regras geradas de teste sobre o uso de mínimo grau de certeza. Teste considerando 2 variáveis relevantes e 2 variáveis irrelevantes no conjunto de dados de treinamento.

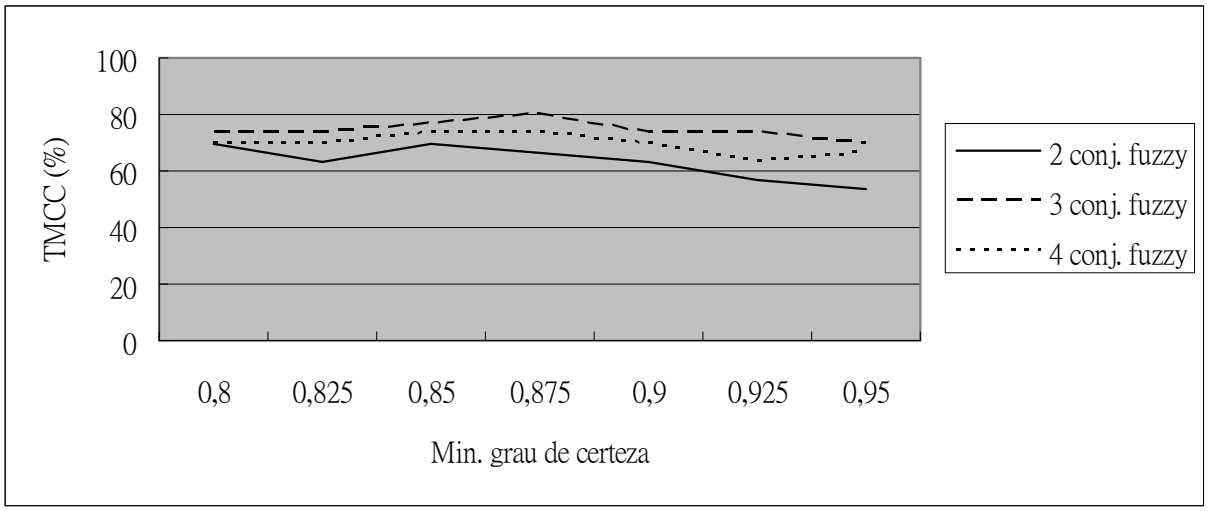

Figura 61 Taxa Média de Classificação Correta (TMCC) de teste sobre o uso de mínimo grau de certeza. Teste considerando 2 variáveis relevantes e 3 variáveis irrelevantes no conjunto de dados de treinamento.

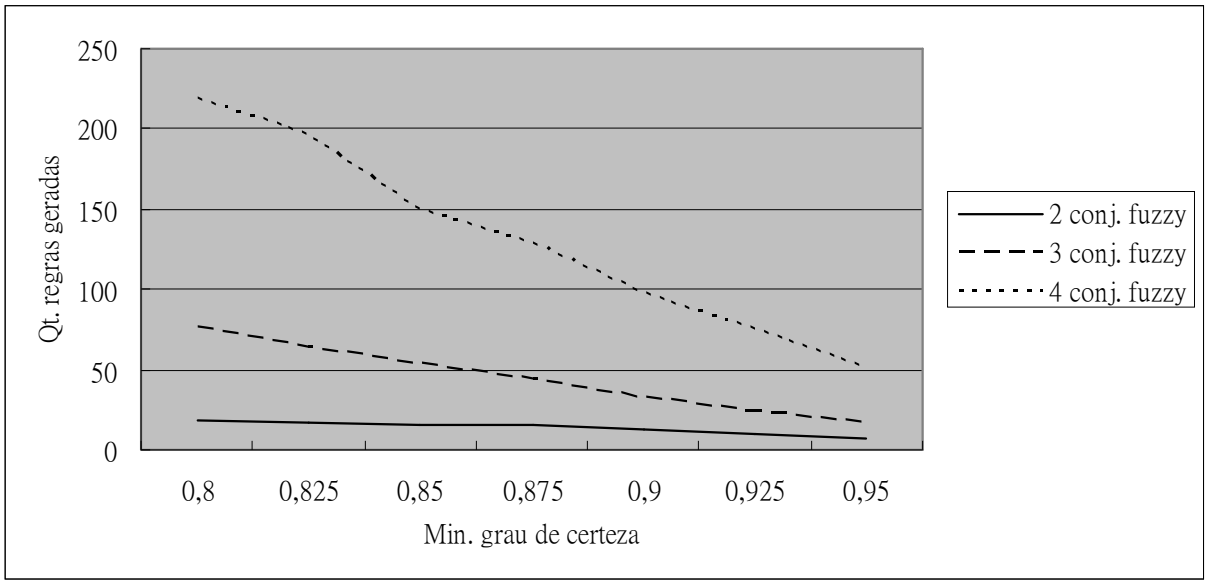

Figura 62 Quantidade de regras geradas de teste sobre o uso de mínimo grau de certeza. Teste considerando 2 variáveis relevantes e 3 variáveis irrelevantes no conjunto de dados de treinamento.

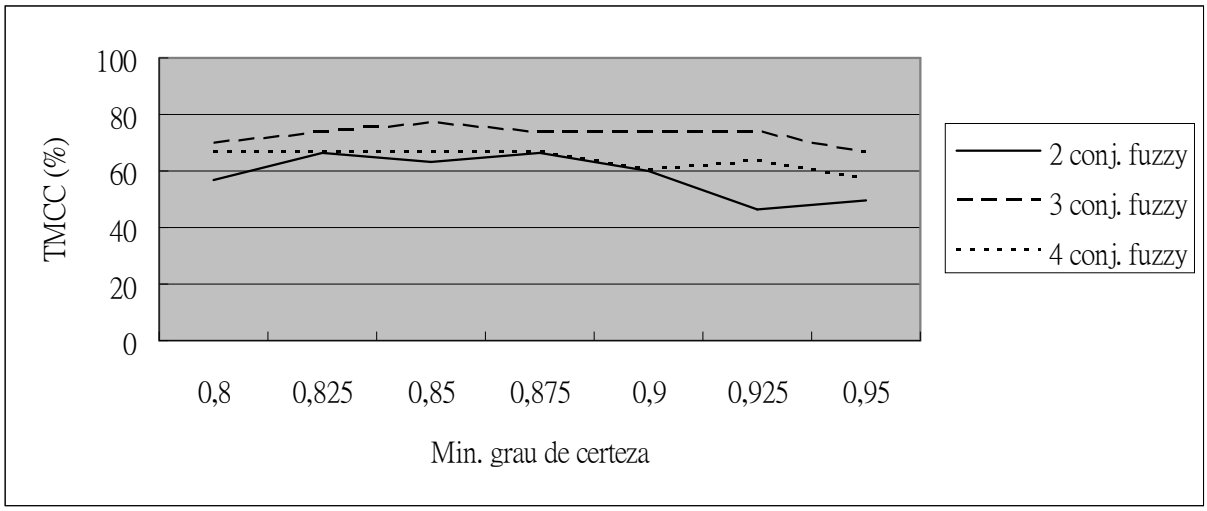

Figura 63 Taxa Média de Classificação Correta (TMCC) de teste sobre o uso de mínimo grau de certeza. Teste considerando 2 variáveis relevantes e 4 variáveis irrelevantes no conjunto de dados de treinamento.

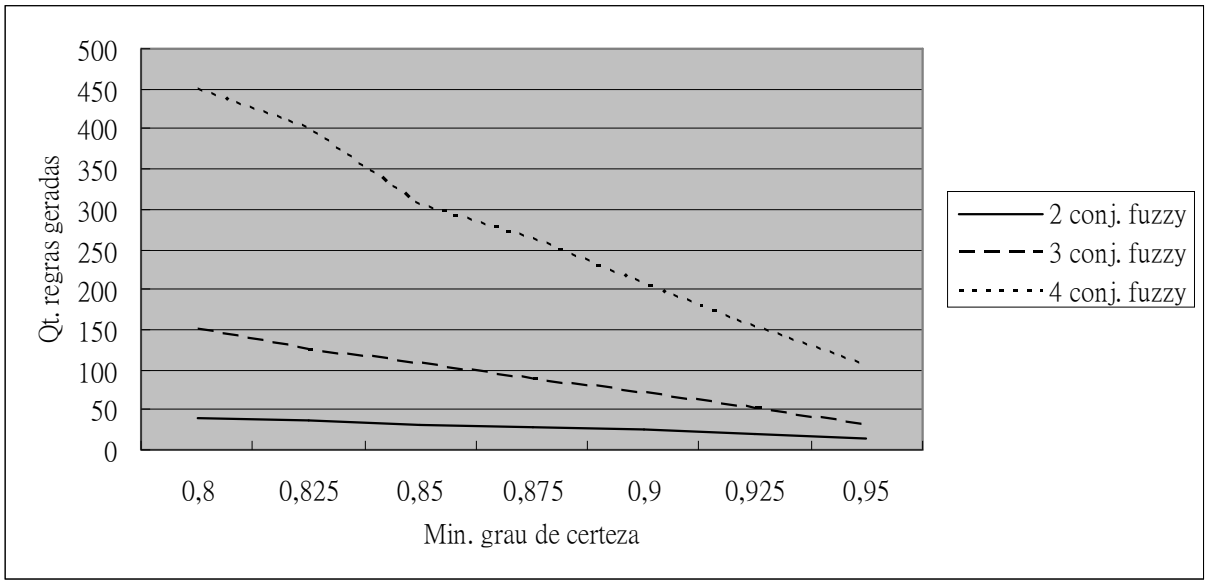

Figura 64 Quantidade de regras geradas de teste sobre o uso de mínimo grau de certeza. Teste considerando 2 variáveis relevantes e 4 variáveis irrelevantes no conjunto de dados de treinamento.

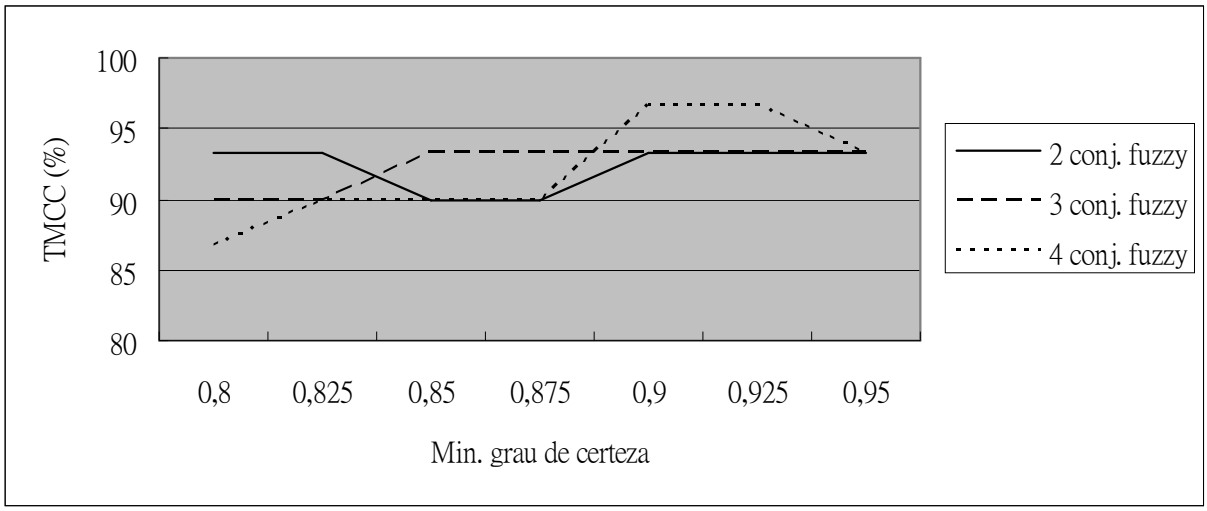

Figura 65 Taxa Média de Classificação Correta (TMCC) de teste sobre o uso de mínimo grau de certeza. Teste considerando 3 variáveis relevantes e 0 variável irrelevante no conjunto de dados de treinamento.

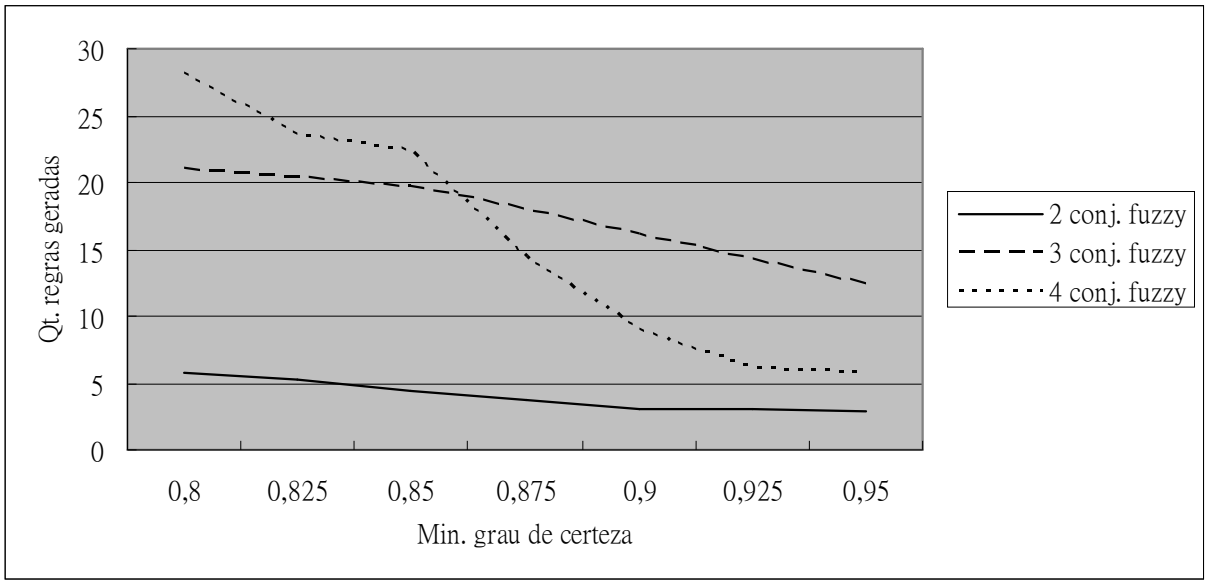

Figura 66 Quantidade de regras geradas de teste sobre o uso de mínimo grau de certeza. Teste considerando 3 variáveis relevantes e 0 variável irrelevante no conjunto de dados de treinamento.

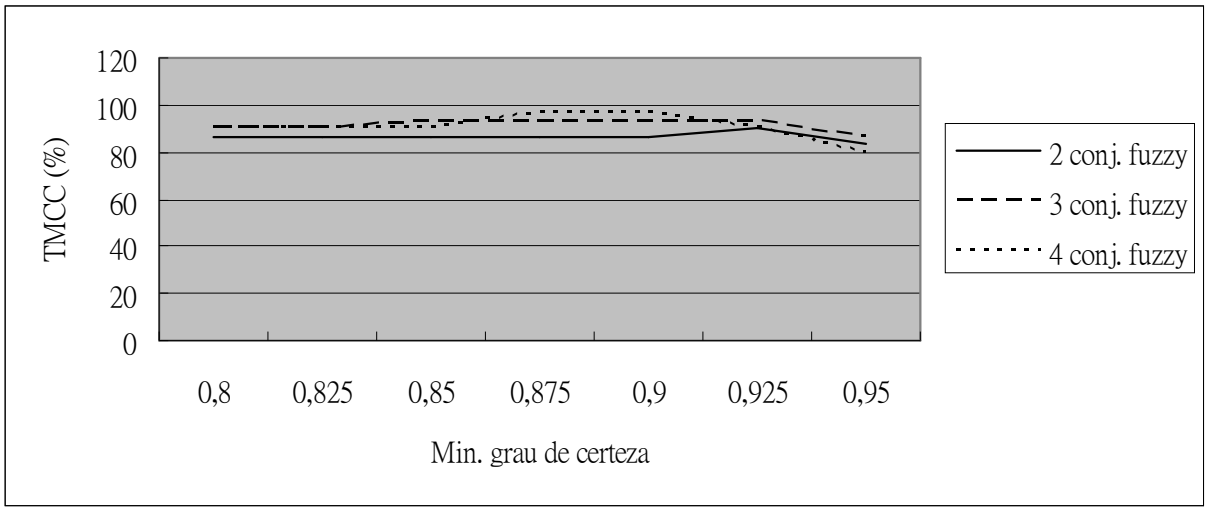

Figura 67 Taxa Média de Classificação Correta (TMCC) de teste sobre o uso de mínimo grau de certeza. Teste considerando 3 variáveis relevantes e 1 variável irrelevante no conjunto de dados de treinamento.

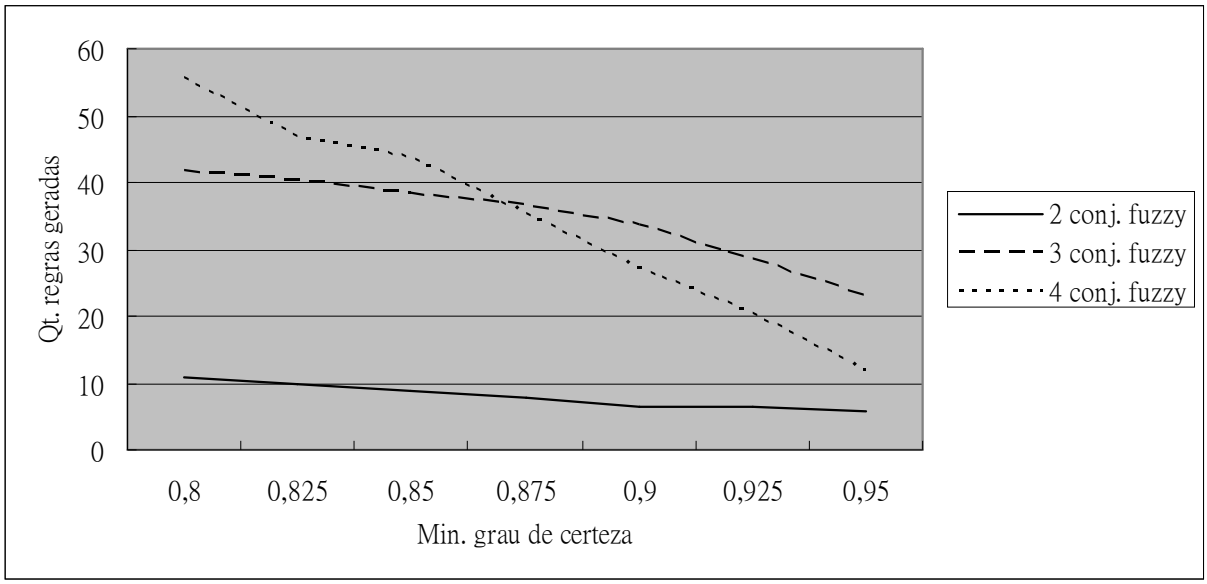

Figura 68 Quantidade de regras geradas de teste sobre o uso de mínimo grau de certeza. Teste considerando 3 variáveis relevantes e 1 variável irrelevante no conjunto de dados de treinamento.

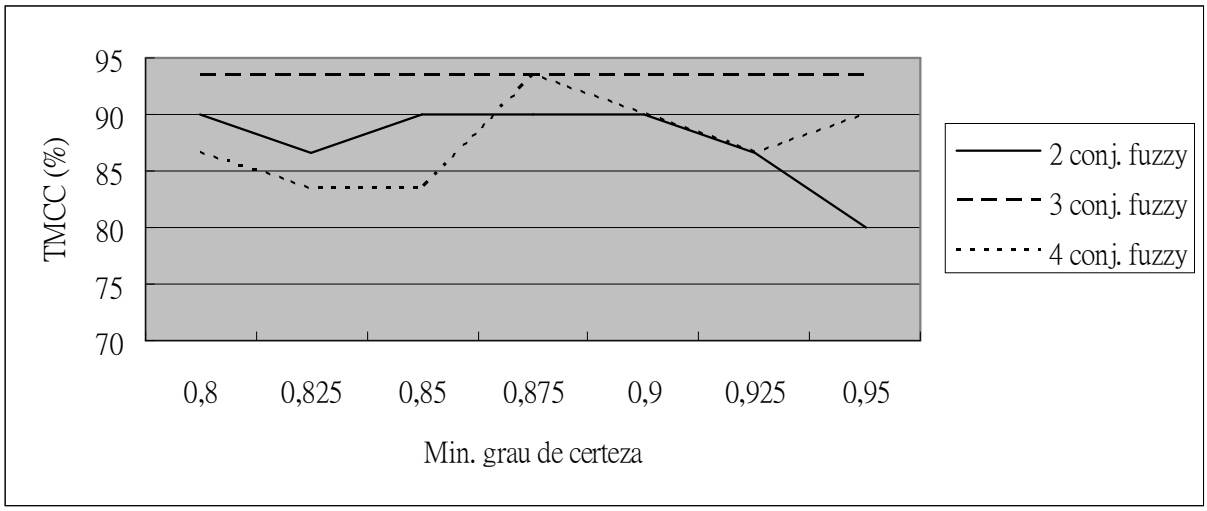

Figura 69 Taxa Média de Classificação Correta (TMCC) de teste sobre o uso de mínimo grau de certeza. Teste considerando 3 variáveis relevantes e 2 variáveis irrelevantes no conjunto de dados de treinamento.

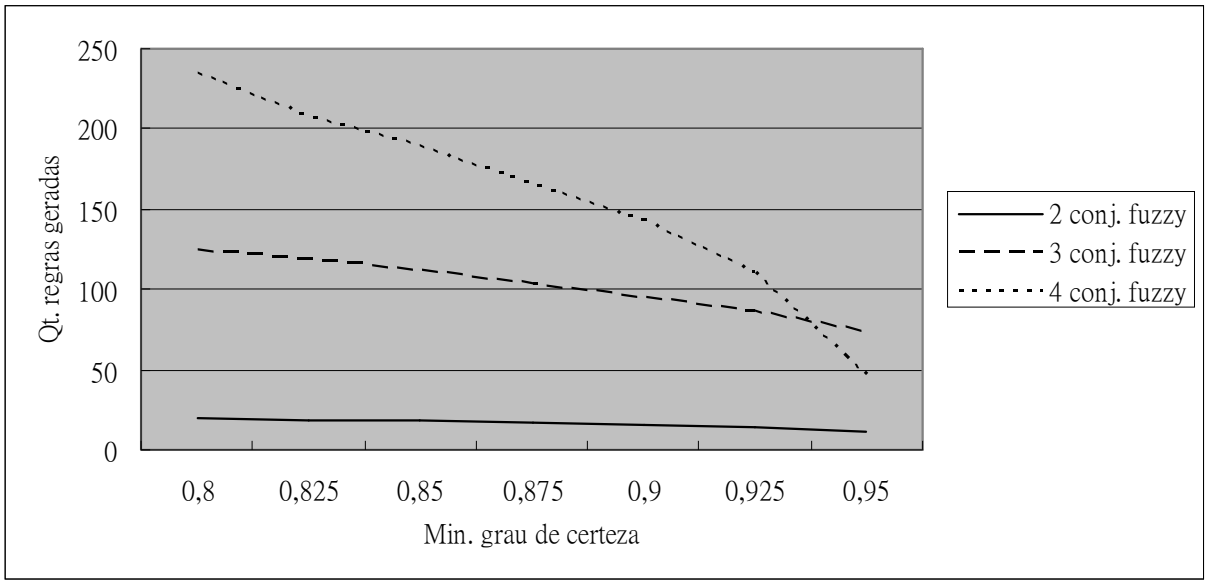

Figura 70 Quantidade de regras geradas de teste sobre o uso de mínimo grau de certeza. Teste considerando 3 variáveis relevantes e 2 variáveis irrelevantes no conjunto de dados de treinamento.

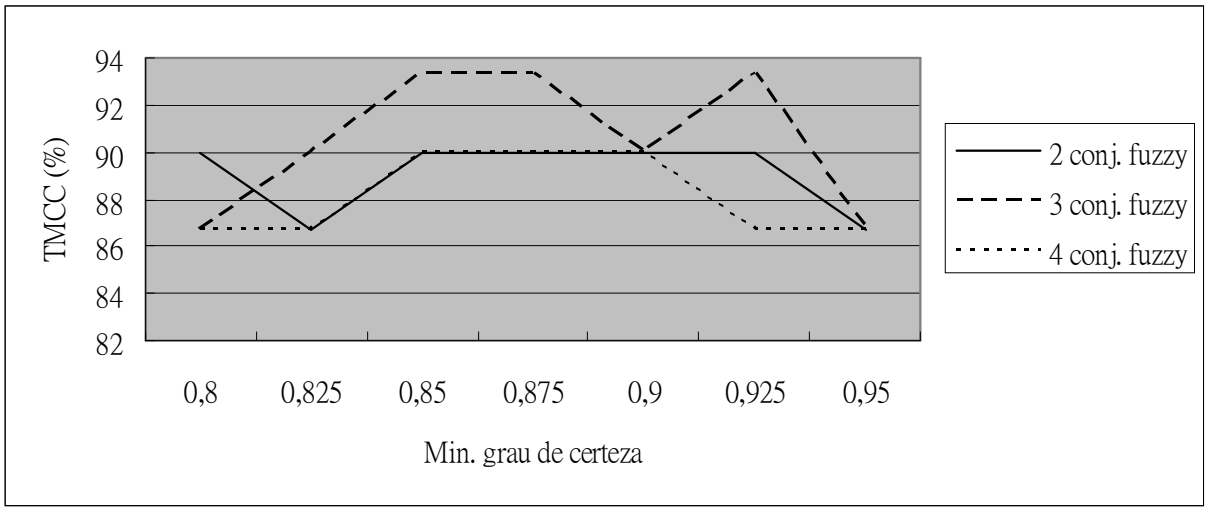

Figura 71 Taxa Média de Classificação Correta (TMCC) de teste sobre o uso de mínimo grau de certeza. Teste considerando 3 variáveis relevantes e 3 variáveis irrelevantes no conjunto de dados de treinamento.

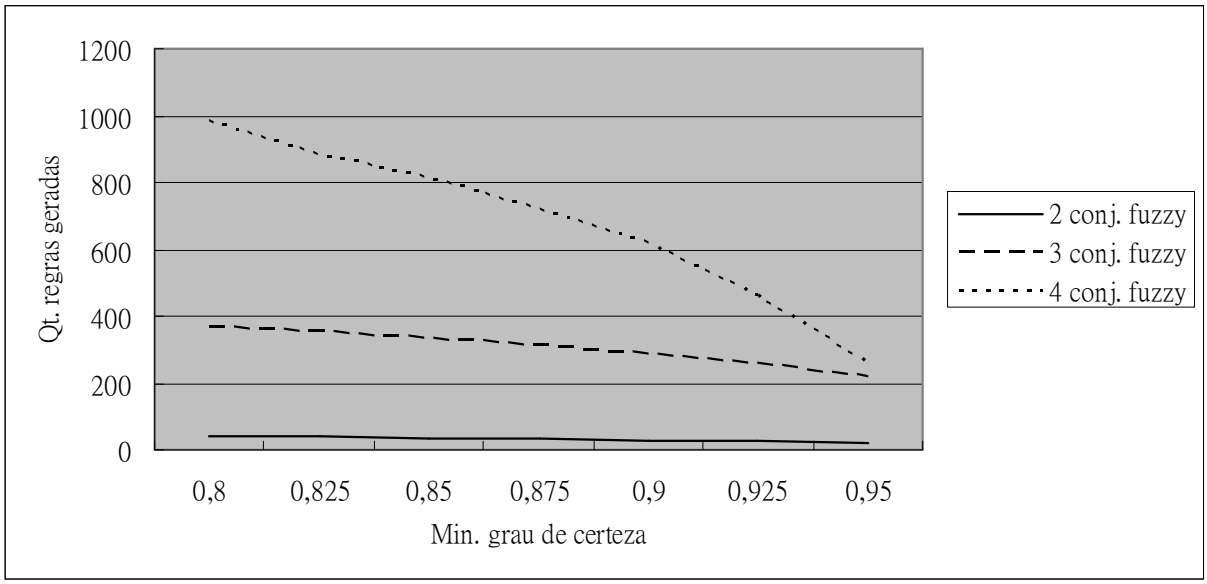

Figura 72 Quantidade de regras geradas de teste sobre o uso de mínimo grau de certeza. Teste considerando 3 variáveis relevantes e 3 variáveis irrelevantes no conjunto de dados de treinamento.

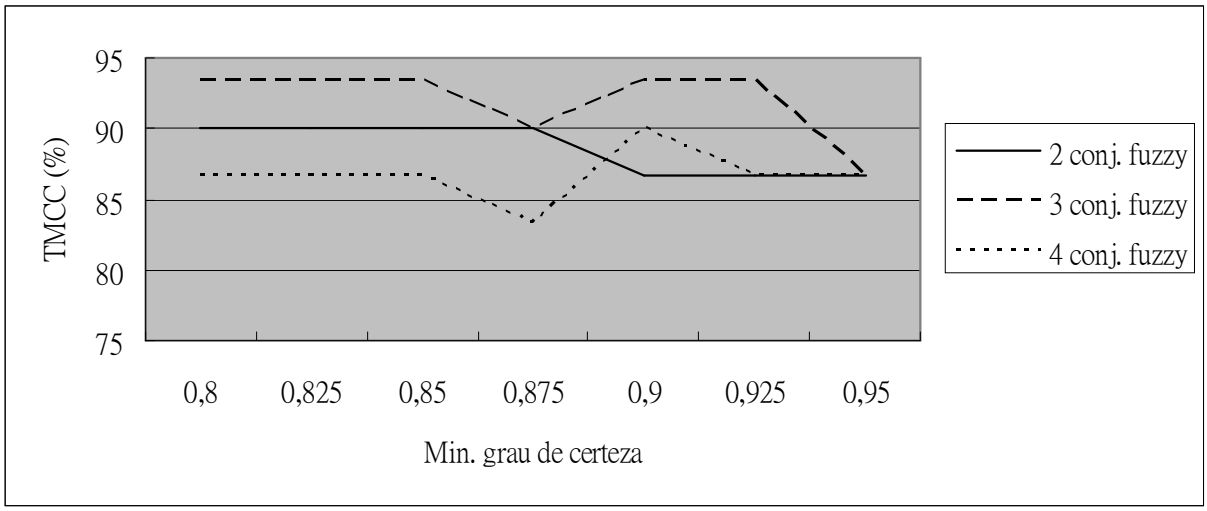

Figura 73 Taxa Média de Classificação Correta (TMCC) de teste sobre o uso de mínimo grau de certeza. Teste considerando 3 variáveis relevantes e 4 variáveis irrelevantes no conjunto de dados de treinamento.

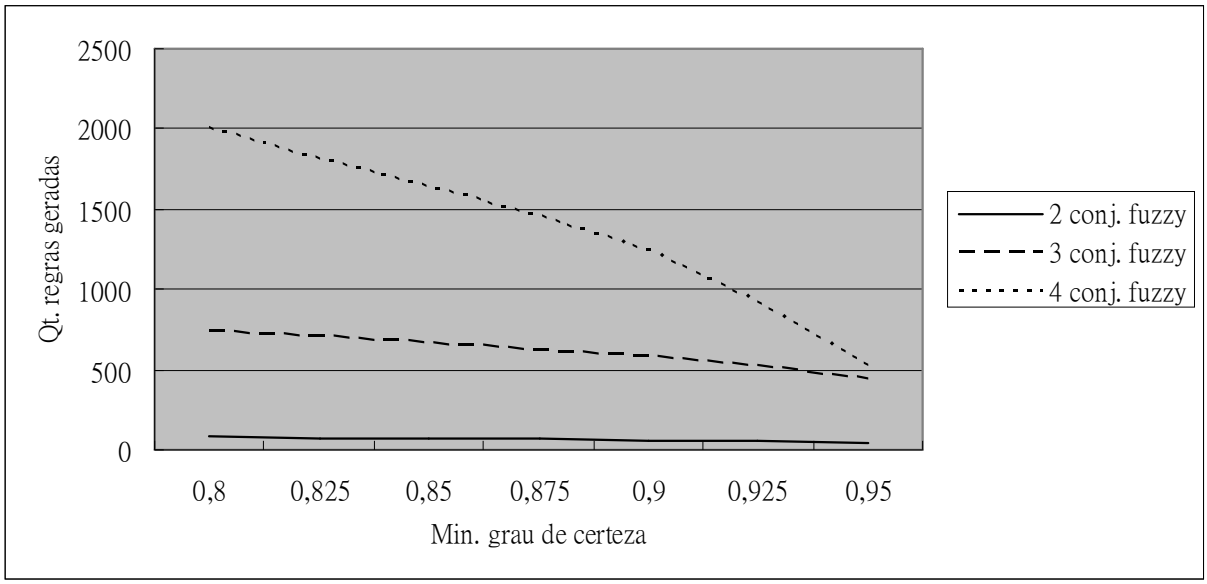

Figura 74 Quantidade de regras geradas de teste sobre o uso de mínimo grau de certeza. Teste considerando 3 variáveis relevantes e 4 variáveis irrelevantes no conjunto de dados de treinamento.

Apêndice 2. Resultados detalhados com a introdução da regra default

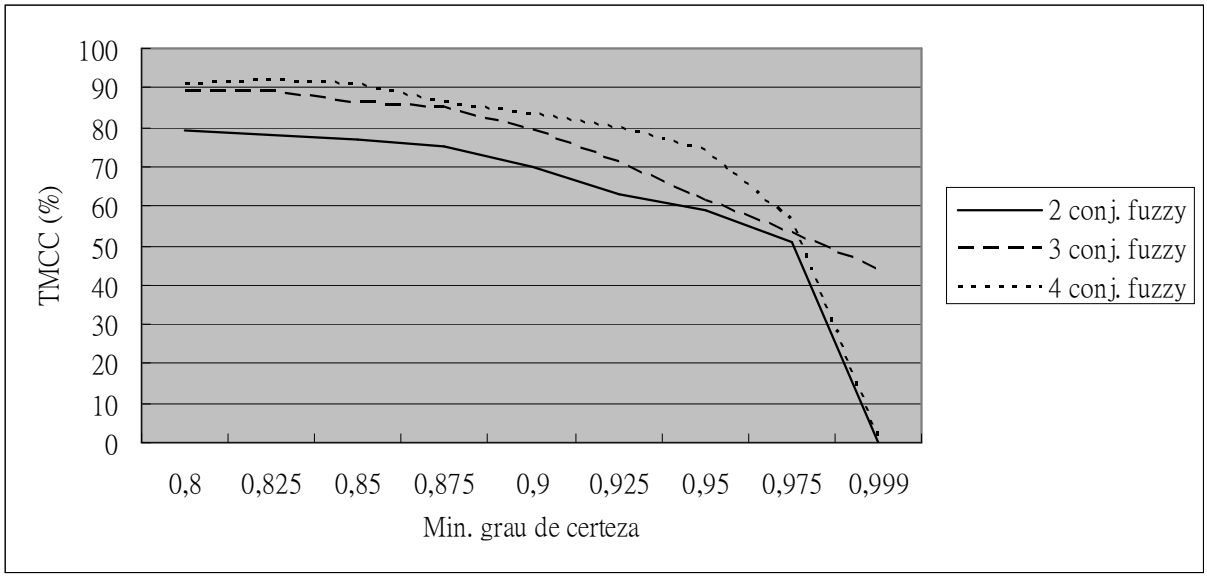

Figura 75 Taxa Média de Classificação Correta (TMCC) de teste sobre a inclusão da regra default. Teste não considerando o uso da regra default.

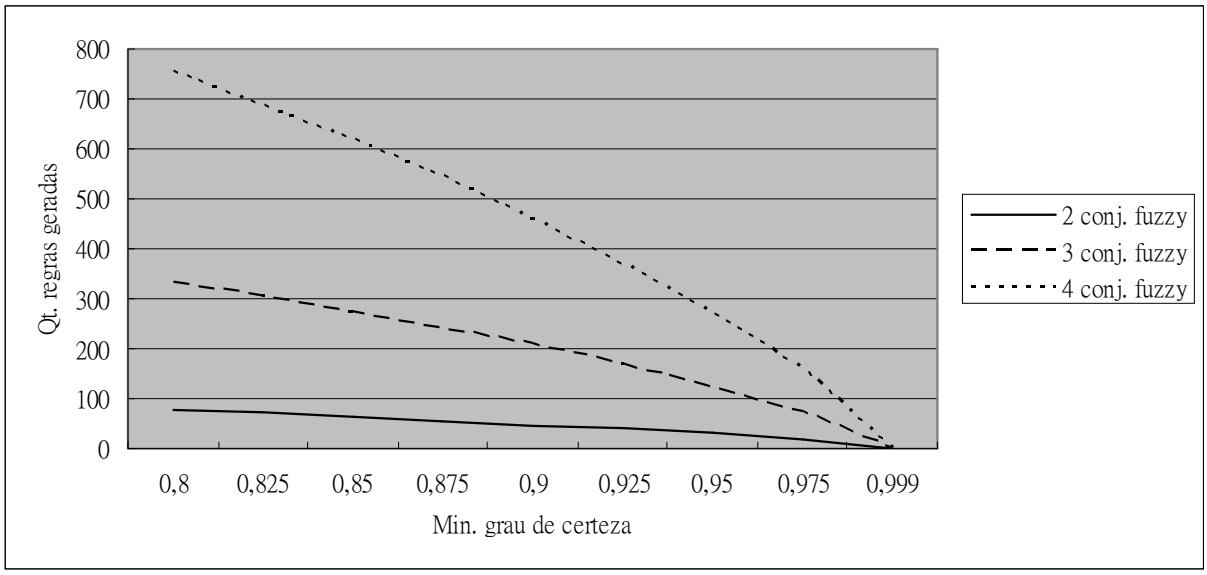

Figura 76 Quantidade de regras geradas de teste sobre a inclusão da regra default. Teste não considerando o uso da regra default.

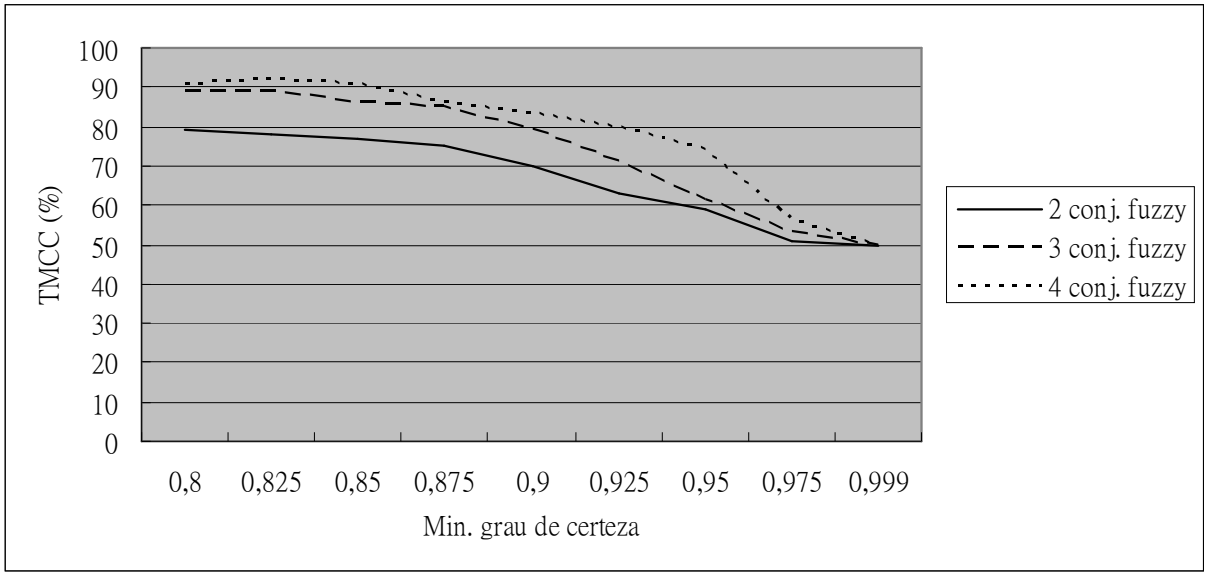

Figura 77 Taxa Média de Classificação Correta (TMCC) de teste sobre a inclusão da regra default. Teste considerando o uso da regra default.

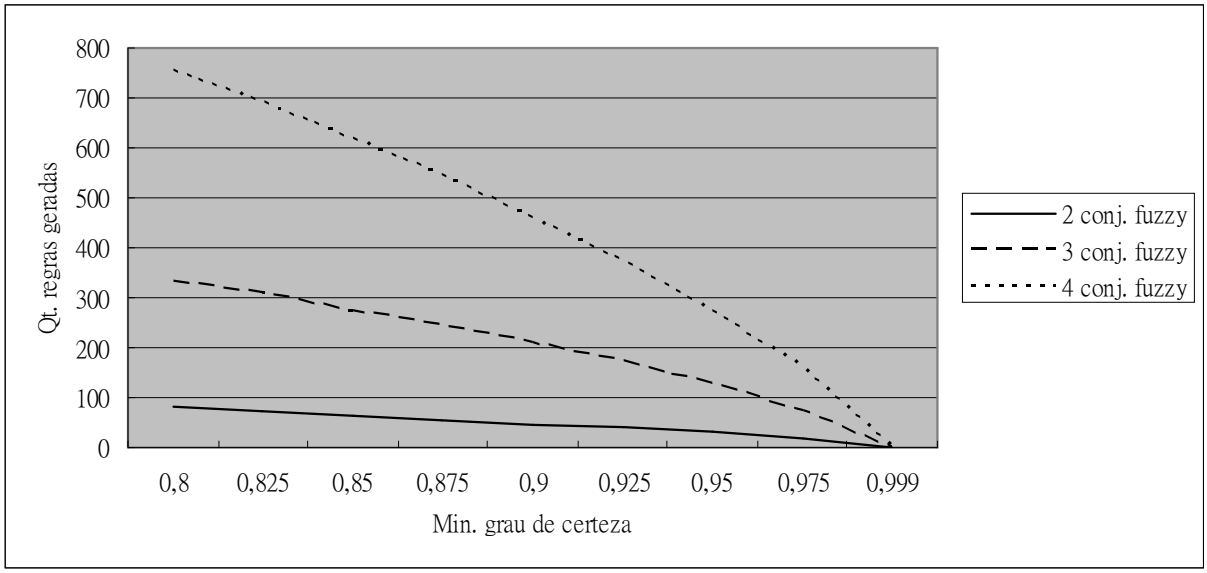

Figura 78 Quantidade de regras geradas de teste sobre a inclusão da regra default. Teste considerando o uso da regra default.

Apêndice 3. Resultados detalhados comparando com o método J48

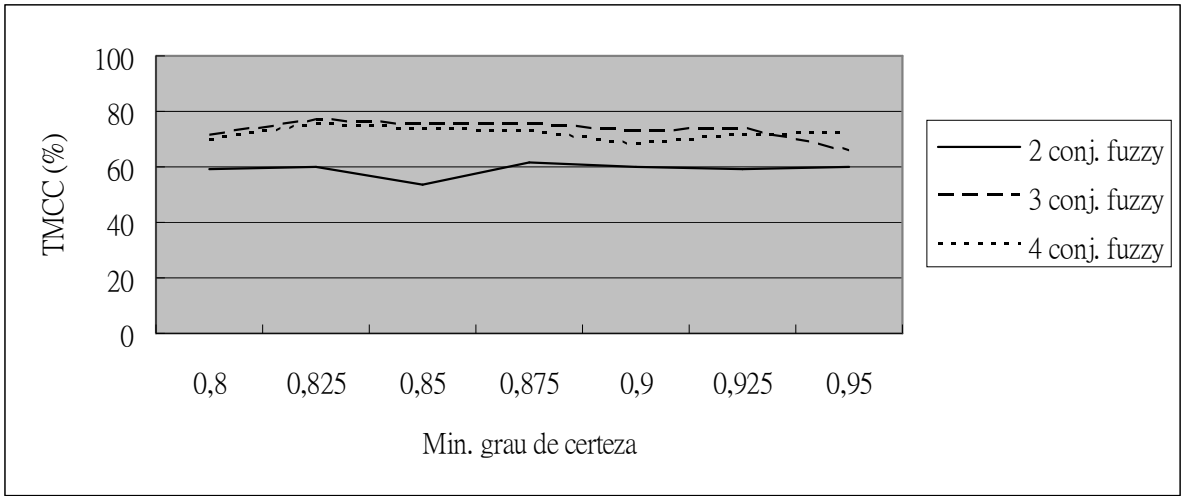

Figura 79 Taxa Média de Classificação Correta (TMCC) de teste comparativo com o método J48. Teste considerando o uso da regra default e a remoção das regras supérfluas.

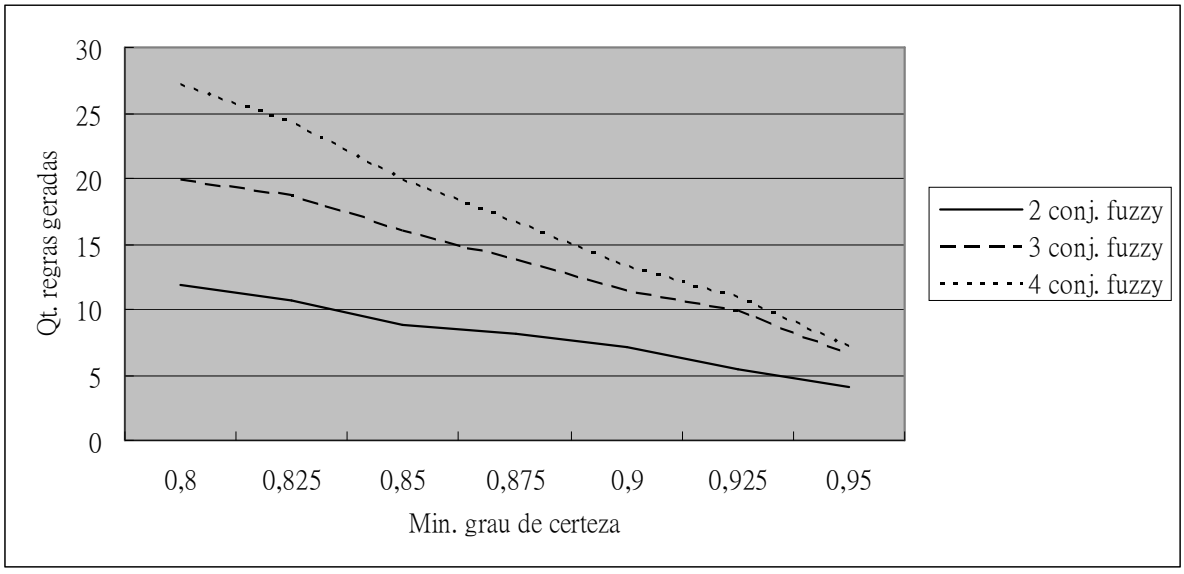

Figura 80 Quantidade de regras geradas de teste comparativo com o método J48. Teste considerando o uso da regra default e a remoção das regras supérfluas.

Qtd. CF	TMCC.(%)	#Reg.
2	84	6,1
3	89	6,5
4	91	7,7

Tabela 6 Taxa Média de Classificação Correta (TMCC) e quantidade de regras geradas (#Reg.) pelo método J48 considerando que a base de dados que foi utilizada no treinamento passou pelo processo de fuzzificação utilizando 2, 3 e 4 conjuntos fuzzy.

.

Apêndice 4. Resultados detalhados com a introdução da poda e a comparação entre diferentes algoritmos

A seguinte legenda será usada para as tabelas a serem apresentadas neste apêndice:

- Base: a base de dados que foi utilizada para obter os resultados;
- Alg.: algoritmo que foi utilizado para aprender uma rede Bayesiana ou se não for uma rede Bayesiana, então será o algoritmo que foi utilizado como comparação;
- Poda: a utilização da poda *Rule Post-Pruning*;
- NM: aplicação do método *Naïve Markov* apenas para os classificadores *Naïve Bayes*;
- Min. GC: mínimo grau de certeza que foi aplicado;
- TMCC: taxa média de classificação correta obtida;
- #Reg.: quantidade média de regras geradas pelos métodos;
- #Ant.: quantidade média de antecedentes das regras geradas;

Tabela 7 Resultados comparativos entre os algoritmos e a aplicação ou não da poda na base *Baseline*.

Base	Alg.	Poda	NM	Mín. GC	TMCC	#Reg.	#Ant.
Baseline	IC	Não		0,8	99,25	7	1,714
				0,85	99,25	7	1,714
				0,9	99,25	7	1,714
				0,95	99,25	7	1,714
		Sim		0,8	50,17	4	0,75
				0,85	50,17	4	0,75
				0,9	50,17	4	0,75
				0,95	50,17	4	0,75
	K2	Não		0,8	45,04	6,4	1,28
				0,85	48,35	6,8	1,29
				0,9	50,02	7	1,3
				0,95	35,14	6,3	1,19
		Sim		0,8	56,29	6,4	0,84
				0,85	56,18	7	0,86
				0,9	56,18	7	0,86
				0,95	34,97	6,3	0,84
	Naive Bayes	Não	Não	0,8	59,01	6	1
				0,85	59,01	6	1
				0,9	50,25	6	1,3
				0,95	50,25	6	1,3
			Sim	0,8	59,01	6	1
				0,85	59,01	6	1
				0,9	50,25	6	1,3
				0,95	50,25	6	1,3
		Sim	Não	0,8	42,35	6	0,83
				0,85	42,35	6	0,83
				0,9	42,35	6	0,83
				0,95	42,35	6	0,83
			Sim	0,8	42,35	6	0,83
				0,85	42,35	6	0,83
				0,9	42,35	6	0,83
				0,95	42,35	6	0,83
	J48	Não			99,25	5	2
		Sim			99,24	5	2

Tabela 8 Resultados comparativos entre os algoritmos e a aplicação ou não da poda na base VI.

Base	Alg.	Poda	NM	Mín. GC	TMCC	#Reg.	#Ant.
VI	IC	Não		0,8	49,46	3	1,3
				0,85	49,46	3	1,3
				0,9	33,02	2	1
				0,95	33,02	2	1
		Sim		0,8	46,13	3	0,67
				0,85	46,13	3	0,67
				0,9	33,02	2	0,5
				0,95	33,02	2	0,5
	K2	Não		0,8	57,95	3,5	1,86
				0,85	51,15	3,1	1,45
				0,9	46,37	2,9	1,48
				0,95	33,02	2	1
		Sim		0,8	47,73	3,4	0,71
				0,85	46,79	3,2	0,69
				0,9	41,2	2,7	0,63
				0,95	33,02	2	0,5
	Naive Bayes	Não	Não	0,8	33,02	2	0,5
				0,85	33,02	2	0,5
				0,9	33,02	2	0,5
				0,95	330,2	2	0,5
			Sim	0,8	33,02	2	0,5
				0,85	33,02	2	0,5
				0,9	33,02	2	0,5
				0,95	330,2	2	0,5
		Sim	Não	0,8	33,02	2	0,5
				0,85	33,02	2	0,5
				0,9	33,02	2	0,5
				0,95	33,02	2	0,5
			Sim	0,8	33,02	2	0,5
				0,85	33,02	2	0,5
				0,9	33,02	2	0,5
				0,95	33,02	2	0,05
	J48	Não			83,04	11,2	3,18
		Sim			82,82	13,9	3,37

Tabela 9 Resultados comparativos entre os algoritmos e a aplicação ou não da poda na base RR.

Base	Alg.	Poda	NM	Mín. GC	TMCC	#Reg.	#Ant.
RR	IC	Não		0,8	99,6	7	1,71
				0,85	99,6	7	1,71
				0,9	99,6	7	1,71
				0,95	99,6	7	1,71
		Sim		0,8	44	4,1	0,76
				0,85	44	4,1	0,76
				0,9	44	4,1	0,76
				0,95	44	4,2	0,76
	K2	Não		0,8	42,6	7	1,14
				0,85	42,6	7	0,89
				0,9	42,6	7	0,86
				0,95	42,8	7	1,14
		Sim		0,8	42,5	6,9	0,86
				0,85	42,5	6,9	0,86
				0,9	41,2	7	0,86
				0,95	41,3	7	0,86
	Naive Bayes	Não	Não	0,8	65,4	6	1
				0,85	65,4	6	1
				0,9	60,8	6	1,12
				0,95	58,7	6	1,33
			Sim	0,8	65,4	6	1
				0,85	65,4	6	1
				0,9	60,8	6	1,12
				0,95	58,7	6	1,33
		Sim	Não	0,8	51,09	6	0,83
				0,85	51,09	6	0,83
				0,9	51,09	6	0,83
				0,95	54,1	6,2	0,84
			Sim	0,8	51,1	6	0,83
				0,85	51,1	6	0,83
				0,9	51,1	6	0,83
				0,95	54,1	6,2	0,84
	J48	Não			99,4	5	2
		Sim			99,4	5	2

Tabela 10 Resultados comparativos entre os algoritmos e a aplicação ou não da poda na base RMG.

Base	Alg.	Poda	NM	Mín. GC	TMCC	#Reg.	#Ant.
RMG	IC	Não		0,8	93,66	6,1	2,15
				0,85	93,66	6,1	2,15
				0,9	93,66	6,1	2,15
				0,95	93,66	6,1	2,15
		Sim		0,8	56,22	4,2	0,76
				0,85	56,22	4,2	0,76
				0,9	56,22	4,2	0,76
				0,95	56,22	4,2	0,76
	K2	Não		0,8	63,65	6,6	0,92
				0,85	76,23	6,4	0,98
				0,9	79,34	6,4	1,03
				0,95	83,61	5,9	1
		Sim		0,8	50,83	6,2	0,84
				0,85	53,27	6,2	0,84
				0,9	54,14	6,3	0,84
				0,95	64,51	5,8	0,083
	Naive Bayes	Não	Não	0,8	85,42	8	1,08
				0,85	85,42	8	1,08
				0,9	85,42	8	1,15
				0,95	61,6	7	1,03
			Sim	0,8	85,42	8	1,08
				0,85	85,42	8	1,08
				0,9	85,42	8	1,15
				0,95	61,6	7	1,03
		Sim	Não	0,8	64,09	8	0,88
				0,85	64,09	8	0,88
				0,9	62,7	8	0,88
				0,95	65,31	7	0,86
			Sim	0,8	64,09	8	0,88
				0,85	64,09	8	0,88
				0,9	62,7	8	0,88
				0,95	65,31	7	0,86
	J48	Não			99,15	14	2,86
		Sim			99,15	14	2,86

Tabela 11 Resultados comparativos entre os algoritmos e a aplicação ou não da poda na base Ruídos.

Base	Alg.	Poda	NM	Mín. GC	TMCC	#Reg.	#Ant.
Ruídos	IC	Não		0,8	61,79	5	0,8
				0,85	61,79	5	0,8
				0,9	69,82	5,6	0,82
				0,95	72,27	5	0,8
		Sim		0,8	61,79	5	0,8
				0,85	61,79	5	0,8
				0,9	72,27	5	0,8
				0,95	72,27	5	0,8
	K2	Não		0,8	44,69	5	0,82
				0,85	44,69	5	0,86
				0,9	50,27	6	1,02
				0,95	52,25	6	1,17
		Sim		0,8	46,77	5	0,8
				0,85	46,77	5	0,8
				0,9	48,73	6	0,83
				0,95	48,73	6	0,83
	Naive Bayes	Não	Não	0,8	70,01	6	0,83
				0,85	49,99	4	0,75
				0,9	57,98	3	0,67
				0,95	57,98	3	0,67
			Sim	0,8	70,01	6	0,83
				0,85	49,99	4	0,75
				0,9	57,98	3	0,67
				0,95	57,98	3	0,67
		Sim	Não	0,8	70,02	6	0,83
				0,85	49,99	4	0,75
				0,9	57,98	3	0,67
				0,95	57,98	3	0,67
			Sim	0,8	70,02	6	0,83
				0,85	49,99	4	0,75
				0,9	57,98	3	0,67
				0,95	57,98	3	0,67
	J48	Não			94,2	34	4,11
		Sim			94,25	29,4	3,97

Tabela 12 Resultados comparativos entre os algoritmos e a aplicação ou não da poda na base Incerteza.

Base	Alg.	Poda	NM	Mín. GC	TMCC	#Reg.	#Ant.
Incerteza	IC	Não		0,8	35,11	1	0
				0,85	35,11	1	0
				0,9	35,11	1	0
				0,95	35,11	1	0
		Sim		0,8	35,11	1	0
				0,85	35,11	1	0
				0,9	35,11	1	0
				0,95	35,11	1	0
	K2	Não		0,8	41,07	4,6	1,09
				0,85	56,94	5	1,6
				0,9	56,42	4,8	2,73
				0,95	32,45	1,8	2,67
		Sim		0,8	45,27	5,5	0,82
				0,85	45,98	5	0,8
				0,9	40,39	4,2	0,76
				0,95	32,32	1,9	0,47
	Naive Bayes	Não	Não	0,8	45,73	6	1,42
				0,85	35,11	4,2	1,55
				0,9	35,11	4	2,33
				0,95	35,11	2	0,95
			Sim	0,8	45,73	6	1,42
				0,85	35,11	4,2	1,55
				0,9	35,11	4	2,33
				0,95	35,11	2	0,95
		Sim	Não	0,8	41,01	5,9	0,83
				0,85	34,18	4	0,75
				0,9	34,53	4	0,75
				0,95	35,11	2	0,5
			Sim	0,8	41,01	5,9	0,83
				0,85	34,18	4	0,75
				0,9	34,53	4	0,75
				0,95	35,11	2	0,5
	J48	Não			60,45	7,6	2,71
		Sim			60,43	6,8	2,44

Tabela 13 Resultados comparativos entre os algoritmos e aplicação ou não da poda na base Desbalanceada.

Base	Alg.	Poda	NM	Mín. GC	TMCC	#Reg.	#Ant.
Desbalanceada	IC	Não		0,8	92,43	6,5	1,8
				0,85	92,43	6,5	1,8
				0,9	92,43	6,5	1,8
				0,95	92,43	6,5	1,8
		Sim		0,8	55,95	3	0,67
				0,85	55,95	3	0,67
				0,9	55,95	3	0,67
				0,95	55,95	3	0,67
	K2	Não		0,8	49,79	5	0,8
				0,85	49,79	5	0,8
				0,9	49,79	5	0,8
				0,95	49,79	5	0,8
		Sim		0,8	49,79	5	0,8
				0,85	49,79	5	0,8
				0,9	49,79	5	0,8
				0,95	49,79	5	0,8
	Naive Bayes	Não	Não	0,8	86,07	6	1
				0,85	86,07	6	1
				0,9	86,07	6	1
				0,95	86,07	6	1
			Sim	0,8	86,07	6	1
				0,85	86,07	6	1
				0,9	86,07	6	1
				0,95	86,07	6	1
		Sim	Não	0,8	52,84	6	0,83
				0,85	52,84	6	0,83
				0,9	52,84	6	0,83
				0,95	52,84	6	0,83
			Sim	0,8	52,84	6	0,83
				0,85	52,84	6	0,83
				0,9	52,84	6	0,83
				0,95	52,84	6	0,83
	J48	Não			99,54	5	2
		Sim			99,52	5	2

Apêndice 5. Exemplos de regras de geração

Neste apêndice, apresentaremos as regras que foram utilizadas para definir os padrões das bases sintéticas e algumas informações sobre estas bases. As regras terão o formato de:

se <atributo> é <valor> e ... então <classe> é (<valor da classe> =
<porcentagem que aparece>)

Onde <atributo> é o atributo que está sendo considerado em questão. Para estas bases sintéticas terão variáveis a1, a2, a3, a4, a5, a6 e a7, sendo que a7 é a classe.

<valor> é um dos possíveis valores que o atributo em questão pode assumir para as regras que possui valor linguístico, porque os dados serão gerados de acordo com o conjunto fuzzy definido para a geração. Os valores que cada variável pode assumir são:

a1: {v1, v2, v3, v4}
a2: {0 ~ 10}, conjuntos fuzzy em formas triangulares: {baixo, médio, alto}
a3: {5 ~ 12}, conjuntos fuzzy em formas triangulares: {baixo, médio, alto}
a4: {v1, v2, v3}
a5: {0, 1}
a6: {2 ~ 3}, conjuntos fuzzy em formas triangulares: {baixo, médio, alto}
a7: {c1, c2, c3}

<porcentagem que aparece> representa quantos por cento dos dados que são gerados utilizando a regra em questão assumirão aquele valor e será apresentado num valor entre 0(0%) a 1(100%).

Será listado na Tabela 14 as regra de cada base e, por questão de espaço, as variáveis linguísticas terão os seguintes valores correspondentes, baixo = b, médio = m e alto = a:

Tabela 14 Regras de geração.

Base de dados	a1	a2	a3	a4	a5	a6	a7(c1)	a7(c2)	a7(c3)
Baseline	v1	m	b	v3	0	a	1	0	0
	v2	m	m	v2	1	b	1	0	0
	v2	m	a	v3	1	a	0	1	0
	v1	a	b	v2	1	a	0	1	0
	v2	a	a	v1	1	m	0	0	1
	v4	b	a	v2	0	a	0	0	1
VI	-	-	b	v3	0	a	1	0	0
	-	-	m	v2	1	b	1	0	0
	-	-	a	v3	1	a	0	1	0
	-	-	b	v2	1	a	0	1	0
	-	-	a	v1	1	m	0	0	1
	-	-	a	v2	0	a	0	0	1
RR	v1	m	b	v3	0	a	1	0	0
	v2	m	m	v2	1	b	1	0	0
	v1	m	b	v3	0	a	1	0	0
	v2	m	a	v3	1	a	0	1	0
	v1	a	b	v2	1	a	0	1	0
	v2	a	a	v1	1	m	0	0	1
	v4	b	a	v2	0	a	0	0	1
RMG	v1	m	b	v3	0	a	1	0	0
	v2	m	m	v2	1	b	1	0	0
	v1	m	-	-	-	-	1	0	0
	v2	m	a	v3	1	a	0	1	0
	v1	a	b	v2	1	a	0	1	0
	v2	a	a	v1	1	m	0	0	1
	v4	b	a	v2	0	a	0	0	1
Ruídos	v1	m	-	-	-	-	1	0	0
	v2	-	-	-	-	b	1	0	0
	-	-	-	-	1	a	0	1	0
	-	-	-	v2	1	-	0	1	0
	-	-	a	v1	-	-	0	0	1
	-	-	a	-	0	-	0	0	1

Incerteza	v1	m	b	v3	0	a	0.6	0.2	0.2
	v2	m	m	v2	1	b	0.6	0.2	0.2
	v2	m	a	v3	1	a	0.2	0.6	0.2
	v1	a	b	v2	1	a	0.2	0.6	0.2
	v2	a	a	v1	1	m	0.2	0.2	0.6
	v4	b	a	v2	0	a	0.2	0.2	0.6
Desbalanceada	v1	m	b	v3	0	a	1	0	0
	v2	m	m	v2	1	b	1	0	0
	v2	m	a	v3	1	a	0	1	0
	v1	a	b	v2	1	a	0	1	0
	v2	a	a	v1	1	m	0	0	1
	v2	a	b	v3	1	b	0	0	1
	v3	m	a	v1	0	m	0	0	1
	v4	a	a	v1	1	a	0	0	1
	v1	b	b	v1	0	b	0	0	1
	v4	b	a	v2	0	a	0	0	1
	v4	b	a	v2	1	b	0	0	1
	v4	m	m	v1	1	b	0	0	1

As seguintes informações são sobre todas as variáveis de todas as bases de dados. Para variáveis discretas, será demonstrado da seguinte forma:

<nome da variável> : <Primeiro possível valor>(<Quantidade de instâncias que possui este valor>), ..., <Último possível valor>(<Quantidade de instâncias que possui este valor>).

Para variáveis contínuas, será demonstrado da seguinte forma:

<nome da variável> : Min = <O valor mínimo>; Max = <O valor máximo>; Média = <O valor médio de todas as instâncias>.

Baseline:

a1 : v1(3334), v2(5000), v3(0), v4(1666).

a2 : Min = 0,01; Max = 9,999; Média = 6,752.

a3 : Min = 5,001; Max = 11,999; Média = 9,553.

a4 : v1(1666), v2(5000), v3(3334).

a5 : 0(3333), 1(6667).

a6 : Min = 2; Max = 2,999; Média = 2,775.

a7 : c1(3295), c2(3299), c3(3406).

VI:

a1 : v1(5012), v2(1647), v3(1701), v4(1640).
a2 : Min = 0,001; Max = 9,994; Média = 2,845.
a3 : Min = 5; Max = 11,998; Média = 6,989.
a4 : v1(4923), v2(2527), (2550).
a5 : 0(3333), 1(6667).
a6 : Min = 2; Max = 2,999; Média = 2,775.
a7 : c1(3300), c2(3295), c3(3405).

RR:

a1 : v1(4286), v2(4286), v3(0), v4(1428).
a2 : Min = 0,004; Max = 9,999; Média = 6,696.
a3 : Min = 5; Max = 11,999; Média = 9,095.
a4 : v1(1428), v2(4285), (4287).
a5 : 0(4286), 1(5714).
a6 : Min = 2; Max = 2,999; Média = 2,801.
a7 : c1(4241), c2(2835), c3(2924).

RMG:

a1 : v1(4286), v2(4286), v3(0), v4(1428).
a2 : Min = 0,003; Max = 9,999; Média = 6,725.
a3 : Min = 5; Max = 11,999; Média = 9,158.
a4 : v1(2122), v2(4655), (3223).
a5 : 0(3576), 1(6424).
a6 : Min = 2; Max = 2,999; Média = 2,705.
a7 : c1(4237), c2(2822), c3(2941).

Ruídos:

a1 : v1(5027), v2(2721), v3(1140), v4(1112).
a2 : Min = 0,001; Max = 9,999; Média = 3,441.
a3 : Min = 5; Max = 11,999; Média = 8,504.
a4 : v1(5020), v2(3324), (1656).
a5 : 0(4173), 1(5827).
a6 : Min = 2; Max = 2,999; Média = 2,379.
a7 : c1(3289), c2(3296), c3(3415).

Incerteza:

a1 : v1(3334), v2(5000), v3(0), v4(1666).

a2 : Min = 0,002; Max = 9,999; Média = 6,741.

a3 : Min = 5,001; Max = 11,999; Média = 9,543.

a4 : v1(1666), v2(5000), (3334).

a5 : 0(3333), 1(6667).

a6 : Min = 2; Max = 2,999; Média = 2,776.

a7 : c1(3258), c2(3251), c3(3511).

Desbalanceada:

a1 : v1(2224), v2(5001), v3(0), v4(2775).

a2 : Min = 0,001; Max = 9,999; Média = 6,576.

a3 : Min = 5,002; Max = 11,999; Média = 10,366.

a4 : v1(2777), v2(4999), (2224).

a5 : 0(3331), 1(6669).

a6 : Min = 2; Max = 2,999; Média = 2,825.

a7 : c1(1100), c2(3305), c3(5595).