UM ABORDAGEM PARA O PROBLEMA DE CLASSIFICAÇÃO UTILIZANDO PROGRAMAÇÃO INTEIRA

MARCOS HENRIQUE MUNIZ

UM ABORDAGEM PARA O PROBLEMA DE CLASSIFICAÇÃO UTILIZANDO PROGRAMAÇÃO INTEIRA

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Orientador: Geraldo Robson Mateus

Belo Horizonte

Setembro de 2010

Muniz, Marcos Henrique
D1234p Um abordagem para o problema de classificação utilizando programação inteira / Marcos Henrique Muniz. — Belo Horizonte, 2010
xxiv, 84 f. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal de Minas Gerais
Orientador: Geraldo Robson Mateus

1. Pesquisa Operacional. 2. Mineração de dados. 3. Classificação de dados. I. Título.

CDU 519.6*82.10

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

Uma abordagem para o problema de classificação de dados utilizando árvores de decisão e programação inteira

MARCOS HENRIQUE MUNIZ

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROF. GERALDO ROBSON MATEUS - Orientador
Departamento de Ciência da Computação - UFMG

PROF. MARTÍN GOMÉZ RAVETTI
Departamento de Engenharia de Produção - UFMG

PROF. WAGNER MEIRA JÚNIOR
Departamento de Ciência da Computação - UFMG

Belo Horizonte, 22 de setembro de 2010.

Agradecimentos

Em primeiro lugar quero agradecer ao professor Geraldo Robson Mateus que foi meu orientador ainda na graduação e sempre foi solícito quando tive qualquer dúvida. Gostaria de agradecer ao professor Martín Gómez Ravetti por ter me ajudado um pouco durante o trabalho e muito nas correções que melhoraram bastante o trabalho e ao professor Wagner Meira Júnior que esteve presente na minha defesa e foi meu professor na disciplina de mineração de dados que foi fundamental para este trabalho.

Não poderia deixar de fora os meus agradecimentos à Pollyanna Gomes Coelho que sempre esteve ao meu lado durante os momentos mais difícieis do mestrado e ainda me ajudou muito na parte escrita do trabalho, Polly o que você fez por mim jamais poderei agradecer o suficiente.

Aos meus familiares que me deram suporte em todos os momentos.

Ao meu amigo Alysson Rodrigues que foi o primeiro a dar a notícia que tinha sido aprovado no mestrado, acho que aquele foi um dos momentos mais felizes da minha vida.

Aos amigos Alessandro Liebman, Luiz Guilherme Pais que sempre estiverão por perto nessa caminhada.

Aos amigos Rafael Barra, Emmanuelly Barros e Vânia Duarte que me acompanharam nos estudos de PAA por algumas noites. Aos amigos Luíz Felipe Hussin, Rafael Barra, Henrique Baião, Fillipe Brandão, que fizeram as matérias de Otimização, tudo bem que depois o Hussin mudou pra Robótica, mas não antes de ver o rato parar a apresentação do trabalho de Otimização Combinatória porque estava com frio! Aquele momento ficará marcado.

Ao Livernull, time que sou fundador junto com mais alguns e que serviu para fortalecer algumas amizades e criar novas amizades, sem falar do melhor churrasco do DCC que é feito pelo Livernull.

Ao Synergia que me deu apoio antes, quando ainda fazia disciplina isolada, durante e depois do mestrado. Os anos que estive no Synergia foram muito proveitosos.

Enfim, gostaria de agradecer a todos que estiveram comigo antes, durante e depois

dessa jornada.

"I don't remember, you looking any better, then again I don't remeber you"
(John Mayer)

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

Devido ao avanço no processamento de máquinas nos últimos anos, muitas empresas têm armazenado diversas informações dos seus clientes, negócios e outros dados de controle interno. Porém, estes dados não são apenas armazenados para controle de históricos. A importância dos dados nas empresas aumentou significativamente, pois através deles têm-se conseguido obter informações relevantes relativas ao seu negócio, clientes, estratégias de mercado, entre outras informações. Esta mudança no paradigma de sistemas de informação se deve ao avanço das tecnologias de armazenamento de dados, que são capazes de armazenar uma quantidade enorme dos mesmos por um preço acessível se comparadas às tecnologias utilizadas no passado.

A partir destas informações, a tendência é que as empresas possam oferecer ao clientes: novos produtos, vantagens em programas de relacionamento, fidelização de clientes e até a remoção dos mesmos da cartela de negócios. Neste último caso, Pinheiro [2008] aborda situações em que abandonar um cliente seja a melhor escolha a ser feita por uma companhia.

Para auxiliar um tomador de decisões a encontrar a melhor solução para seu negócio em determinado tempo, os Sistemas de Gerenciamento de Relacionamento com o Cliente (SGRC), são fundamentais. Os SGRC são utilizados para uma empresa conhecer melhor as características e o perfil de comportamento de seus clientes e, a partir disso, encontrar informações que facilitem a tomada de decisões.

Para a construção destes SGRC várias técnicas são utilizadas, entre elas encontra-se a mineração de dados que é definida por Pinheiro [2008] como o processo de descoberta de padrões e tendências existentes em repositórios de dados. A mineração de dados é um campo da ciência da computação e geralmente está associada a descoberta do conhecimento. A descoberta do conhecimento segundo Fayyad et al. [1996], é o processo não trivial de identificar padrões em dados que sejam válidos, novos, poten-

cialmente úteis e compreensíveis em última instância. Além disso, a mineração de dados é um passo no processo da descoberta do conhecimento em dados, preocupado com meios algorítmicos no qual padrões ou modelos (estruturas) são enumerados a partir dos dados sobre um esforço computacional aceitável.

1.1 Motivação

A área de mineração de dados, em geral, têm muitas pesquisas nos campos de estatística e aprendizado de máquinas utilizando algumas técnicas como: redes neurais artificiais, árvores de decisão, algoritmos genéticos, redes bayesianas, modelos de regressão, regras de associação, entre outras. Problemas como classificação, regressão linear, visualização de dados e agrupamento, podem ser formulados como problemas de otimização, porém existem poucos trabalhos em mineração de dados que empregam técnicas de otimização na resolução dos problemas. De acordo com Bertsimas & Shioda [2007], existe uma crença estabelecida no início da década de 70, de que os métodos de programação inteira não eram tratáveis computacionalmente.

Com isto, seguindo a própria definição de mineração de dados, os métodos de programação inteira, foram deixados de lado nas pesquisas. Aliado a esta crença e ao sucesso na utilização das demais técnicas empregadas na resolução dos problemas de mineração de dados, técnicas de otimização não têm sido exploradas nos últimos anos para estes problemas. Alguns trabalhos recentes, como Bertsimas & Shioda [2007] , Boros et al. [2000] , Osuna et al. [1997] , Bradley et al. [1999] , Kunapuli et al. [2008] , entre outros, empregaram técnicas de otimização para resolver problemas de mineração de dados. Os mesmos obtiveram resultados satisfatórios e mostram que a crença de que técnicas de otimização não são viáveis para resolver este tipo de problema não é realidade. Com este cenário favorável, este trabalho tem como objetivo explorar técnicas de otimização com métodos exatos.

1.2 Objetivo

O objetivo deste trabalho é resolver o problema de classificação de duas classes através de um novo algoritmo que combina a utilização de árvores de decisão com um algoritmo de programação inteira visando melhorar a interpretabilidade da solução do algoritmo de programação inteira e aumentar a eficiência da árvore de decisão.

O problema de classificação de duas classes pode ser definido da seguinte maneira: suponha que a base de dados seja um conjunto de n observações (x_i, y_i), i = 1, ...,

n com $x_i \in \mathbb{R}^d$ e $y_i \in \{0, 1\}$, onde x_i é um ponto com d dimensões e y_i indica se o ponto x_i pertence a classe 0 ou 1. Neste trabalho, a base de dados é dividida em dois conjuntos, um para treinamento e outro para testes. O objetivo é minimizar o erro para determinar a classe de um ponto pertencente à base de testes utilizando o modelo construído a partir da base de treinamento.

1.3 Trabalhos relacionados

Nesta seção serão apresentados alguns trabalhos relacionados que foram separados de acordo com o tipo de técnica utilizada para facilitar o entendimento e comparação entre eles. Este trabalho utiliza as Árvores de decisão em conjunto com um algoritmo de programação inteira chamado CRIO (do inglês, *Classification and Regression via Integer Optimization*).

De acordo com estas informações esta seção está organizada da seguinte maneira. Primeiramente serão apresentados trabalhos que utilizam a Árvores de decisão. Em seguida trabalhos em que as Máquinas de Vetores de Suporte (SVM, do inglês, *Support Vectors Machines*) são utilizadas serão apresentados. A SVM é uma técnica de programação matemática na qual o CRIO pode ser comparado, por isso os trabalhos que utilizam esta técnica serão apresentados nesta seção.

1.3.1 Árvores de decisão

Árvores de decisão (AD) é um dos métodos mais famosos na área de mineração de dados e aprendizado de máquina devido à sua simplicidade, interpretabilidade e robustez. O modelo das AD é criado ao escolher, para cada nível, um atributo que melhor separe o conjunto de dados de acordo com a classe. A cada atributo escolhido é criada uma ramificação para cada valor presente na lista de possíveis valores do atributo. Para cada nova ramificação criada é verificado se todos os objetos separados pertencem à mesma classe, ou a grande maioria deles. Em caso positivo, uma folha é criada para ela. Caso contrário é verificado se ainda existem atributos que possam ser utilizados na ramificação. Se existirem, então o processo de construção da árvore é repetido até que não existam mais atributos disponíveis, ou todas as ramificações sejam associadas a uma folha. Caso todos os atributos sejam utilizados e ainda não seja possível determinar a classe de um conjunto de objetos deve-se escolher a classe majoritária para o conjunto e criar uma folha para a ramificação. Mais detalhes sobre as AD são discutidos no capítulo 2.

No trabalho de Li [2005] é proposta uma árvore de decisão capaz de trabalhar com bases de dados massivas no qual o tamanho excede a capacidade de memória de um computador. O algoritmo proposto incorpora a utilização de modelos de otimização lineares no processo de particionamento da árvore de decisão.

Já no trabalho de Chandra & Varghese [2009] é proposto um novo método para particionar a árvore de decisão que tem como objetivo diminuir o tamanho da árvore de decisão(da raiz até a folha). Os resultados obtidos mostram que a técnica obteve árvores de decisão mais compactas e com qualidade comparável às árvores de decisão que utilizam as formas de particionamento mais tradicionais como Critério de Gini e Ganho de Informação, que são descritas no capítulo 2.

No trabalho de Polat & Güneş [2009] um método baseado no algoritmo C4.5, apresentado no capítulo 2, combinado com a abordagem um contra todos é proposto para problemas de classificação multi-classe. Este método constróis M árvores de decisão binárias, onde M é o número de classes. Uma função é criada para cada árvore de decisão e uma nova instância será classificada pela função que tiver o maior valor.

No trabalho de Jenhani et al. [2008] é proposto uma árvore de decisão baseada no algortimo C4.5 porém capaz de lidar com incertezas na base de dados utilizando a teoria das possibilidades. Outro fator interessante neste trabalho é a utilização de mais de um atributo ao mesmo tempo para particionar a AD.

No trabalho de Chen et al. [2009b] é proposta uma técnica para construir árvores de decisão utilizando múltiplos atributos em cada nível através de uma técnica conhecida como restrição de custos. A restrição de custos exige que cada nível da árvore "pague"um determinado valor que não pode ser excedido na construção da árvore.

Já no trabalho de Zhou & Chen [2002] é proposta uma árvore de decisão híbrida que combina a utilização da árvore de decisão com redes neurais. Os atributos são classificados em qualitativos e quantitativos. A árvore de decisão é construída e as folhas nas quais a diversidade dos objetos esteja além de um determinado limiar são marcadas para serem substituídas pela solução que será encontrada pelas redes neurais utilizando somente atributos qualitativos.

Por fim, no trabalho de Kim & yin Loh [2001] dois métodos de divisão da árvore de decisão univariados e um método de divisão utilizando combinação linear são propostos para construir árvores de decisão com partições de múltiplas ramificações (2 ou mais ramos por atributo).

1.3.2 Máquinas de Vetor de Suporte

As Máquinas de Vetor de Suporte (do Inglês, *Support Vector Machines*(SVM)), desenvolvida por Vapnik [1995] utiliza os princípios da Teoria do Aprendizado Estatístico (TAE) Vapnik [1998]. Originalmente, esta técnica foi elaborada para resolver o problema de classificação de duas classes, mas atualmente é também utilizada em classificação multi-classe e regressão linear, como pode ser visto em Maimon & Rokach [2005]. Segundo Lorena [2003], a TAE visa estabelecer condições matemáticas que permitem escolher um classificador, com bom desempenho, para o conjunto de dados disponíveis para treinamento e teste. Na TAE, o aprendizado significa estimar uma função a partir de um conjunto de treinamento.

Na sua forma mais simples a SVM separa pontos de diferentes classes utilizando um único hiperplano. Nas versões mais sofisticadas da SVM um hiperplano de separação é construído em um espaço com mais dimensões tornando o problema de particionamento não linear. Em todos os casos, conforme Bertsimas & Shioda [2007], o hiperplano ótimo de separação é modelado como um problema de progamação quadrático convexo.

Em Luts et al. [2010] é apresentado um tutorial sobre métodos para classificação baseados em SVM na área de quimiometria. Três problemas reais são utilizados para demonstrar a aplicação da SVM em problemas de classificação. Os problemas que são resolvidos neste tutorial são classificação de tumor cerebral baseado em regiões anatômicas em tecidos utilizando imagem de massa espectral. Nos três casos a SVM obteve resultados satisfatórios.

Já no trabalho de Cervantes et al. [2008] é proposto um método capaz de trabalhar em bases de dados massivas através da utilização da SVM através de um algoritmo de agrupamento que delimita a área através de um esfera. O algoritmo, primeiramente, cria os agrupamentos, para que, em seguida a SVM seja aplicada considerando os centros de cada agrupamento como um ponto. Posteriormente, a SVM é executada novamente para os agrupamentos que contenham pontos de classes diferentes ou para aqueles em que seu centro seja um vetor de suporte. Esta técnica é bastante similar ao CRIO, considerando que o mesmo, inicialmente, realiza o agrupamento e em seguida executa o método de programação linear.

Uma técnica que combina a utilização de AD com SVM é apresentada por Chen et al. [2009a]. Neste trabalho, é utilizada uma AD binária para diminuir a complexidade computacional na utilização da SVM, diminuindo o número de vetores de suporte utilizados, em problemas de classificação multiclasse. Inicialmente são criadas várias SVM para separar cada classe do problema. A cada nível da AD, algumas SVMs, são

escolhidas para formar as ramificações da AD. O processo é repetido até que as folhas da AD sejam criadas. Este método obteve resultados semelhantes a outras SVM que trabalham com classificação multiclasse, porém obteve melhor desempenho.

Outro trabalho que combina a utilização das AD com a SVM foi proposto por Kumar & Gopal [2010]. Neste, o objetivo também é aumentar o desempenho das SVM, porém no problema de classificação de duas classes. Diferentemente da maioria dos trabalhos que procuram melhorar o desempenho das SVM, este trabalho não tem como foco diminuir o número de vetores de suporte. O objetivo é diminuir a quantidade de dados que são utilizados na criação das SVM. Inicialmente a AD é criada e para cada ramificação um limiar de acerto é determinado. Sempre que a ramificação da AD estiver acima do limiar, a SVM será utilizada para classificar estes dados.

Por último, no trabalho de Kianmehr & Alhajj [2008] é proposto um arcabouço que combina a utilição de regras de associação com a SVM. O objetivo deste trabalho é combinar a vantagem do conhecimento discriminativo das regras de associação com o poder de classificação da SVM gerando um modelo que é mais interpretativo que a SVM e mais eficiente que as regras de associação. A interpretabilidade citada como objetivo deste trabalho é para que o usuário possa identificar como as regras foram geradas e quais atributos podem ser mais relevantes.

Capítulo 2

Árvores de Decisão

Este capítulo tem como objetivo definir uma técnica conhecida como Árvores de Decisão(AD) que é bastante utilizada para criar modelos de classificação de dados. Esta técnica foi utilizada no trabalho por ser bastante popular em pesquisa de mineração de dados e aprendizado de máquina, fácil de ser interpretada, robusta e que pode ser combinada com a utilização de outra técnica.

Este capítulo está estruturado da seguinte maneira: primeiramente, é apresentada uma introdução sobre o que são as AD é apresentada. Em seguida é apresentada a origem das AD, como construí-las, os diferentes métodos de escolha da divisão das AD, técnicas para limitar o tamanho das AD, como trabalhar com atributos contínuos e por último como foi implementado o algoritmo de AD utilizado neste trabalho.

2.1 Introdução

AD é um método eficiente e de fácil interpretação para classificação de dados. A partir de um conjunto de dados de treinamento, essa técnica tem como objetivo criar um modelo que consiga identificar qual a classe de um determinado objeto de acordo com o conjunto de dados de treinamento. Para se utilizar este método espera-se que o conjunto de dados atenda as seguintes condições:

- Os dados de treinamento devem possuir um conjunto finito de classes pré-definidas.
- Os atributos dos objetos devem possuir um conjunto finito de valores. Caso os atributos possuam valores contínuos alguns métodos podem ser usados para transformar estes valores em discretos. Na seção 2.4 uma solução para este problema é apresentada.

- Deve-se ter exemplos suficientes para construir o modelo. A quantidade de exemplares necessária é afetada por fatores como o número de atributos e classes e a complexidade do modelo. Maiores detalhes podem ser vistos em Quinlan [1993]

Considerando que as condições citadas anteriormente sejam atendidas o método das AD funciona da seguinte maneira: inicialmente todos os objetos de treinamento são avaliados e o atributo que melhor separa os dados é escolhido para ser a raiz da árvore. A partir deste momento temos um nó (raiz) e n ramificações geradas a partir dos n possíveis valores que o atributo raiz pode assumir. Cada ramificação contém um conjunto de objetos que foram determinados a partir do valor do atributo da ramificação. Por exemplo, se a raiz pode assumir três valores, alto, médio e baixo, os objetos da 3 ramificações geradas serão separados de acordo com o seu valor para o atributo raiz da árvore, os objetos com valor "alto"pertencerão a uma ramificação, os de valor "médio"pertencerão a outra ramificação e assim por diante. Além de possuir um subconjunto de objetos, uma ramificação pode ser definida como folha ou então como nó. Uma folha é o ponto final de uma árvore e associada a ela está uma classe para o conjunto de objetos presentes naquela ramificação. Um nó é uma subárvore. Para cada nó o processo descrito anteriormente deve ser repetido até que todos os objetos de treinamento sejam associados a uma folha. Para determinar se uma ramificação é uma folha basta verificar se todos os objetos da ramificação possuem a mesma classe ou se apenas uma porcentagem bem pequena é de outra classe, ou caso não exista mais nenhum atributo para ser avaliado na subárvore a folha deve ser criada com a classe predominante na maioria dos objetos da ramificação.

Um exemplo simples de uma AD pode ser visto a seguir. No exemplo da figura 2.1, considere que temos que classificar 7 tipos de frutas: maça, uva, banana, cereja, laranja, lima e melão. Podemos classificar uma fruta a partir de algumas respostas sobre seus atributos como: cor, formato, tamanho e sabor. Estas perguntas são feitas até que encontremos a resposta para identificar a fruta. No caso da AD a resposta é uma folha e cada pergunta é um nó da árvore, os valores das ramificações são as respostas das perguntas que levam até uma folha ou outro nó da árvore. Quando chegamos em uma folha da árvore, temos a classificação do objeto. Este método é bem simples de se entender e, principalmente por isso, é considerado o mais popular de mineração de dados como pode ser visto em Chen et al. [2003].

Além da sua simplicidade, existem algumas vantagens deste método em relação aos demais: como citado anteriormente, este é fácil de interpretar, bem organizado, tem baixo custo computacional e é capaz de lidar com dados impuros, de acordo com Breiman et al. [1984].

Porém, tem que se atentar à presença de dados impuros nas bases de dados. Estes têm a característica de possuir valores que não atendem a um determinado padrão e, em muitos métodos, a sua pode afetar consideravelmente o desempenho e a qualidade das soluções.

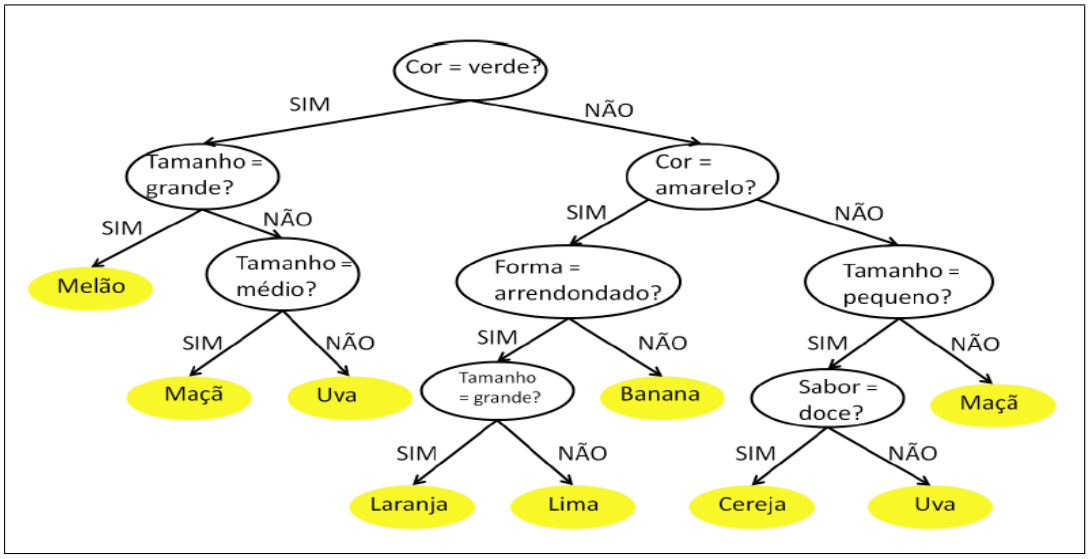

Figura 2.1. Exemplo de uma Árvore de Decisão para classificar uma fruta a partir de suas características.

2.1.1 Origem das Árvores de Decisão

Segundo Murthy [1998] a utilização das AD ocorreu devido à necessidade de explorar dados de pesquisa. A identificação de padrões utilizando AD foi motivada pela necessidade de interpretar imagens a partir de satélites de sensoriamento remoto, como o LANDSAT nos anos 70, Swain & Hauska [1977] . Maiores detalhes sobre a origem das AD podem ser encontrados em Breiman et al. [1984] .

2.2 Construção das Árvores de decisão

Um algoritmo básico de construção de uma AD deve decidir qual o atributo para a separação dos dados a cada nível. A forma mais usual, Breiman et al. [1984] , Quinlan [1993], é avaliar cada atributo utilizando algum critério, que pode ser através de algum método estatístico ou através de alguma heurística, para se determinar o melhor atributo.

Uma vez que o atributo foi escolhido, para cada ramificação da árvore, ou seja, cada valor possível do atributo raiz, devemos repetir o processo até atingir um critério de parada.

A principal decisão dos algoritmos de AD que criam a árvore a partir da raiz até as folhas, formato conhecido como de cima para baixo, é escolher qual atributo deve ser testado a cada nível da árvore. O problema é definir como avaliar o valor de cada atributo. A principal diferença dos métodos de construção de AD está geralmente na escolha de qual atributo deve ser utilizado para particionar o conjunto de dados. Existem vários algoritmos como o ID3 Quinlan [1986a] , CART Breiman et al. [1984] e C4.5 Quinlan [1993], que utilizam o mesmo estilo de construção da AD. A principal diferença entre eles é na fase da escolha de qual atributo deve ser utilizado para a separação dos dados.

Antes de introduzir alguns métodos de divisão da árvore é necessário definir o conceito de probabilidade condicional que é utilizado por alguns métodos.

Definição 2.2.1 *Dado um conjunto de elementos n e um elemento $j \in n$ a probabilidade condicional para o elemento j é dada por:*

$$1 - p(j|t) = \sum_{i \neq j}^{n} p(i|t) \tag{2.1}$$

A seguir, alguns critérios para a divisão da árvore serão detalhados.

2.2.1 Critério da impuridade

O Critério da impuridade foi definido por Breiman et al. [1984] e visa a minimização da impuridade de cada nó da árvore. A impuridade pode ser interpretada como a homogeneidade dos dados em determinado conjunto. A partir da homogeneidade dos dados o que as AD procuram uma função de impureza que separe os dados de maneira que, a cada divisão, os dados de uma partição sejam mais homogêneos. A função de impuridade foi definida como:

Definição 2.2.2 *Uma função de impuridade é uma função Φ definida sobre o conjunto de todas as J-tuplas de números $(p_1, \ldots, p_j)$ que satisfazem $p_j \geq 0$, $j = 1, \ldots, J$, $\sum_j p_j = 1$ com as propriedades:*

(i) Φ é máximo somente no ponto $(\frac{1}{j}, \frac{1}{j}, \ldots, \frac{1}{j})$,

(ii) Φ alcança seu valor mínimo somente nos pontos $(1, 0, \ldots, 0)$, $(0, 1, \ldots, 0)$, ..., $(0, 0, \ldots, 0, 1)$,

(iii) Φ é uma função simétrica de $p_1, \ldots, p_j$.

Como é evidente, a impureza de um nó é máxima, quando o número de exemplares de cada classe é o mesmo; e mínima, quando todos os exemplares pertencem somente a uma classe. A partir da definição da função de impuridade temos a seguinte definição para a medida de impuridade.

Definição 2.2.3 *Dada uma função de impuridade Φ, defina a medida de impuridade $i(t)$ de qualquer nó t como:*

$$i(t) = \Phi(p(1|t), p(2|t), \ldots, p(j|t)). \tag{2.2}$$

Se uma divisão s de um nó t divide com uma proporção de p_R dos dados em t para t_R e a proporção p_L para t_L, a função de decréscimo na impureza pode ser definida como

$$\Delta i(s, t) = i(t) - p_R i(t_R) - p_L i(t_L) \tag{2.3}$$

Baseado nas seguintes definições, podemos considerar a qualidade da divisão $\Phi(s, t)$ como sendo $\Delta i(s, t)$.

2.2.2 Critério de Gini

Segundo o critério de Gini, também definido por Breiman et al. [1984] , a impuridade num dado nó é dada por:

$$\sum_{j \neq i} p(i|t)p(j|t) \tag{2.4}$$

Este critério, aplicável a árvores n-árias, conforme da F. Lisboa [1994], pode ser interpretado como usar uma regra que associe um objeto selecionado aleatoriamente de um nó para a classe i com a probabilidade $p(i|t)$. A probabilidade estimada de que o item seja na realidade da classe j é $p(j|t)$. Dessa maneira, a probabilidade de erro de classificação sobre esta regra é o índice de Gini.

O critério de Gini trabalha com uma classe de cada vez, tentando separar os objetos da classe mais frequente dos demais objetos a cada teste realizado.

2.2.3 Critério de Twoing

O critério de Twoing separa, em cada nó da árvore o conjunto de todas as classes em dois subconjuntos de modo a obter o mínimo de impureza desse nó. Ele fornece informação sobre a semelhança entre as classes, ou seja, as classes são agrupadas,em cada nó, de acordo com as que apresentam maior semelhança. Este critério agrupa, perto da raiz da árvore, um grande número de classes que são semelhantes em alguma característica, e com o crescimento da árvore, estas classes são separadas.

Teorema 2.2.4 *Sobre o critério de duas classes $p(1|t)p(2|t)$, para uma dada divisão s, uma superclasse C_1 que maximize*

$$\Delta i(s,t,C_1) \tag{2.5}$$

é

$$C_1(s) = \{j : p(j|t_L) \geq p(j|t_R)\} \tag{2.6}$$

e

$$max_{C_1}\Delta i(s,t,C_1) = \frac{p_L p_R}{4}[\sum_j |p(j|t_L) - p(j|t_R)|]^2 \tag{2.7}$$

Então a melhor partição de twoing $s^*(C_1{}^*)$ é dada pela partição s^* que maximize $\Phi(s,t)$ onde C_1^* é dado por

$$C_1^*(s) = \{j : p(j|t_L^*) \geq p(j|t_R^*)\} \tag{2.8}$$

onde t_L^*, t_R^* são os nós dado pela divisão s^*

Este método tem uma vantagem do ponto de vista do usuário final. Ele agrupa as classes semelhantes na raiz da árvore fazendo com que os tomadores de decisão possam identificar algum padrão entre as classes.

Um exemplo de um problema que pode ser resolvido efetivamente pelo critério de Twoing é o problema de reconhecimento de palavras faladas. Dado 100 palavras, cada palavra sendo uma classe, a primeira divisão de palavras poderia separar as que são monosilábicas das demais. As próximas divisões devem ocorrer de acordo com outro padrão e assim por diante. No final, pode-se navegar na árvore e identificar padrões de similaridade entre as palavras no decorrer dos níveis da árvore.

O critério de Twoing, assim como o índice de Gini também é utilizado pelo CART. Ao compará-los, verifica-se que ambos, em geral, apresentam resultados semelhantes. Os dois são utilizados para construir árvores binárias e diferem na separação dos dados.

Por outro lado, resultados apresentados por Breiman et al. [1984] mostram que geralmente o critério de Gini é melhor do que o índice de Twoing. Por isso, o critério

de Gini é o método padrão utilizado pelo algoritmo CART. Somente em casos onde o número de possíveis valores de um atibuto é muito grande o critério de Twoing apresentou melhores resultados que o índice de Gini. Sendo, portanto, recomendado nestes casos.

2.2.4 Entropia

A entropia caracteriza a impureza de um conjunto arbitrário de exemplos. Segundo Luenberger [2006] entropia é a medida de informação que esperamos receber no futuro. É a informação média tomada com relação a todos os possíveis resultados. Dado um conjunto S, contendo somente exemplos positivos e negativos (um problema de classificação binário, por exemplo), a entropia do conjunto S é definida como:

$$\text{Entropia(S)} = -p_p \log_2 p_p - p_n \log_2 p_n \tag{2.9}$$

onde p_p é a proporção de exemplos positivos em S e p_n é a proporção de exemplos negativos em S.

Em todos os cálculos envolvendo entropia, definimos $\log_0 = 0$. Por exemplo, suponha que S é um conjunto com 25 exemplares, destes 15 positivos e 10 negativos. A entropia de S é:

$$\text{Entropia(S)} = -(15/25)\log_2(15/25) - (10/25)\log_2(10/25) = 0.970 \tag{2.10}$$

Perceba que se a entropia é 0 todos os exemplares de S pertencem a mesma classe. Por exemplo, se todos os exemplares são positivos ($p_p = 1$), então a Entropia(S) $= -1\log_2(1) - 0\log_2 0 = 0$. Por outro lado, a entropia será 1 (valor máximo), quando o conjunto for separado igualmente em dois ou mais conjuntos. A figura 2.2 mostra a forma como a função de entropia varia para o exemplo binário citado anteriormente.

Uma interpretação para a entropia bastante utilizada na teoria da informação é que a entropia determina o número mínimo de bits necessários para codificar uma mensagem binária. Generalizando, a entropia, para quando um atributo pode ter c valores, ao invés de apenas 2, é:

$$\text{Entropia(S)} = -\sum_{i=1}^{c} p_i \log p_i \tag{2.11}$$

onde p_i é a proporção de S pertencente a classe i.

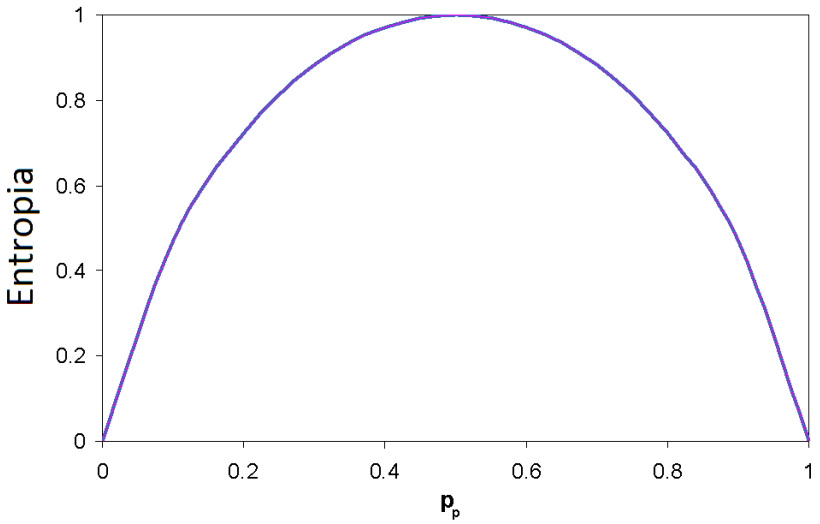

Figura 2.2. Função de entropia para dados binários.

2.2.5 Ganho de informação

A entropia tem a desvantagem de favorecer atributos que particionem os dados no maior número de intervalos conforme da F. Lisboa [1994], ou seja, um nó que seja dividido no maior número de ramificações. Devido, a esta deficiência, Quinlan [1986a] propôs um método estatístico chamado de ganho de informação.

A partir da definição da entropia podemos medir a eficácia de um atributo na separação dos exemplos de treinamento para construção da AD. O ganho de informação, é a expectativa de redução na entropia causada pelo particionamento dos exemplos de acordo com um atributo. Mais precisamente, o ganho de informação, Ganho(S,A) de um atributo A, relativo ao conjunto de exemplos S é definido como:

$$\text{Ganho(S,A)} = \text{Entropia(S)} - \sum_{v \in A} \frac{S_v}{S} \text{Entropia(S)} \tag{2.12}$$

onde $v \in A$ é o conjunto de possíveis valores para o atributo A, e S_v é o subconjunto de S para o qual o atributo A possui o valor v.

O primeiro termo na equação é justamente a entropia do conjunto original, e o segundo termo é a entropia esperada de S, após o conjunto ser particionado utilizando o atributo A. A entropia esperada, descrita pelo segundo termo, é simplesmente a soma das entropias de cada subconjunto S_v, ponderado pela fração dos exemplos $\frac{S_v}{S}$ que pertencem a S_v. O Ganho(S,A) é, portanto, a redução esperada na entropia por conhecer o valor do atributo A.

Desta maneira, para se escolher em qual atributo devemos particionar o conjunto

de dados, basta encontrar o atributo que gere o maior ganho de informação.

Muitos trabalhos já tentaram comparar os diversos algoritmos para a escolha da divisão da árvore porém nenhum trabalho conseguiu encontrar um algoritmo que se sobressaia em relação aos demais. No trabalho de Raileanu & Stoffel [2004] é realizado um experimento para comparar os métodos Ganho de Informção e Critério de Gini. O resultado encontrado não é suficiente para determinar qual dos dois métodos é o melhor. Outros trabalhos também procuram investigar qual o melhor dos métodos para a separação dos dados, porém são inconclusivos. Os métodos em geral são especialistas em algum tipo de problema, como problemas com muitos valores ou somente valores binários, e por isso é difícil encontrar um método que seja melhor em todos os casos. Mais detalhes podem ser encontrados nos trabalhos de: De Mántaras [1991], Gama & Brazdil [1995] e Lim et al. [2000].

2.3 Limitação na dimensão das Árvores de Decisão

Conforme foi mostrado nas seções anteriores, as AD são construídas a partir dos dados de treinamento da raiz até as folhas com basicamente duas condições utilizadas como critério de parada:

(i) Todos os elementos de uma ramificação pertencem a uma mesma classe, logo deve-se criar uma folha para a ramificação com a classe dos elementos.

(ii) Todos os atributos foram avaliados, porém não foi possível separar os dados totalmente. Neste caso, a classe mais frequente entre os elementos da ramificação deve ser escolhida para criar uma folha.

Porém, um modelo de classificação baseado nestas premissas pode ser muito especialista para a base de treinamento utilizada na construção da árvore. Por exemplo, anomalias na base de dados podem participar do modelo de classificação, mesmo possuindo apenas poucos elementos.

Duas soluções são encontradas na literatura para este problema. A primeira é prever quando o crescimento de uma ramificação da árvore deve ser interrompido, e a segunda é podar a árvore depois que ela esteja totalmente construída.

2.3.1 Métodos de limitação durante a construção

Alguns métodos foram propostos com o objetivo de interromper o crescimento da árvore evitando esforço computacional desnecessário e generalizando a árvore de modo a obter

melhores resultados. Vários critérios foram utilizados para decidir quando a construção da árvore deve ser interrompida. A seguir, serão descritos alguns métodos utilizados para a limitação do tamanho da árvore durante a construção.

2.3.1.1 Critério de interrupção baseado na informação mútua

Há um critério de interrupção da dimensão das AD baseado na informação mútua. Segundo este critério, se o ganho de informação obtido com o melhor atributo na divisão da partição for inferior a um determinado δ a partição deve ser considerada uma folha e a classe predominante deve ser escolhida como a classe da folha.

$$\text{Ganho(S)} = -\sum_{1}^{c} p_i . \log_2 p_i \tag{2.13}$$

$$\text{Informação Mútua(S,A)} = \text{Ganho(S)} - \text{Ganho(S,A)} < \delta \tag{2.14}$$

O valor de δ é o fator crucial para o sucesso deste método. Breiman et al. [1984] mostrou que este tipo de definição não é fácil. Se o valor de δ for muito alto a árvore terá seu crescimento interrompido sem que os benefícios das futuras partições seja evidente e se o valor de δ for muito baixo a simplificação da árvore será praticamente insignificante.

2.3.1.2 Critério de interrupção baseado no teste de independência estatística ou teste do qui-quadrado

Também, há um critério de interrupção de dimensão das AD baseado no teste de independência estatística ou teste χ^2 (qui-quadrado). Considere a existência de um conjunto de treino T e de um atributo ω que possibilite a partição $\{t_1, t_2, \ldots, t_n\}$. Sejam p e n as frequências esperadas para um conjunto de dados, então p'_i e n'_i são dadas por:

$$p'_i = p.\frac{p_i + p_n}{p + n} \tag{2.15}$$

$$n'_i = n.\frac{p_i + p_n}{p + n} \tag{2.16}$$

O valor aproximado do χ^2 com $n - 1$ graus de liberdade é dado pela equação:

$$\sum_{i=1}^{n} \frac{(p_i - p'_i)^2}{p'_i} + \frac{(n_i - n'_i)^2}{n'_i} \tag{2.17}$$

Este valor pode ser usado para a determinação da confiança com que se rejeita a hipótese do atributo ω ser independente da classe de objetos em T.

Quinlan [1986b] utilizou o critério de parada descrito acima. Apesar de encontrar resultados satisfatórios em alguns domínios, estes eram irregulares e por isso o próprio autor descartou utilizar a técnica futuramente no algoritmo $C4.5$, Quinlan [1993].

Maiores detalhes sobre critérios de parada na construção da árvore, descritos nesta seção, podem ser vistos em da F. Lisboa [1994].

2.3.2 Métodos de redução da árvore

A poda da árvore é o método mais utilizado para reduzir as AD, garantindo que a mesma seja a mais generalista possível. Uma árvore generalista é aquela que tem nos seus atributos os mais relevantes para todo o conjunto e não somente para parte dos exemplos, ela visa um ótimo global ao invés do ótimo local.

A idéia básica da poda é a seguinte:

(i) Construa a árvore da raiz até as folhas sem aplicar nenhum critério de interrupção. Ou seja, o crescimento da árvore é interrompido quando não houverem mais atributos para serem avaliados, ou todos os exemplos de treinamento em uma determinada partição pertencerem a uma mesma classe.

(ii) Para cada folha, avalie se a sua remoção melhora a estimativa de erro da árvore. Algumas técnicas para realizar esta avaliação serão apresentadas no decorrer deste capítulo.

2.3.2.1 Poda utilizando heurística

Vários métodos usados para a poda da árvore utilizam algum tipo de heurística para avaliar quando se deve podar uma ramificação de uma árvore. Conforme foi mostrado anteriormente, definir um critério de parada é uma tarefa bastante complicada. A solução para este problema é conseguir ajustar o limiar do critério de parada ou então podar a árvore depois que ela esteja toda construída. Já foi mostrado em vários estudos, que os métodos de poda da árvore podem aumentar a generalização de uma AD, principalmente em domínios que contenham dados com ruídos (ou anomalias).

Existem várias técnicas que utilizam heurísticas para a poda da árvore. Nas subseções seguintes estas técnicas serão detalhadas.

2.3.2.2 Poda baseada no custo de complexidade

A poda baseada no custo de complexidade (também conhecida por poda baseada no erro de complexidade) foi proposta por Breiman et al. [1984] e deve ser executada em três fases:

(i) Construa uma árvore com o tamanho necessário para produzir um valor de erro estimado suficientemente baixo.

(ii) Crie um conjunto de árvores a partir do conjunto de dados de treinamento T_0, T_1, ..., T_k, onde T_0 é a árvore original sem nenhuma poda e T_k é a raiz da árvore.

(iii) Uma das k árvores deve ser escolhida como a melhor árvore baseada no erro de estimativa generalizado.

A idéia do método é simples, porém algumas questões precisam ser avaliadas. Uma delas é como as árvores podadas serão geradas e como cada uma será avaliada.

Cada árvore T_{i+1} é obtida substituindo uma ou mais sub-árvores da árvore predecessora T_i. As sub-árvores que são removidas da árvore predecessora são aquelas que obtém o menor ganho na taxa de erro aparente por folha podada:

$$\alpha = \frac{\epsilon(\text{podar(T,t),S}) - \epsilon(T,S)}{\mid \text{folhas(T)} \mid - \mid \text{folhas(podar(T,t)} \mid} \tag{2.18}$$

onde $\epsilon(T,S)$ indica a taxa de erro da árvore T sobre o conjunto de dados S, e $folhas(T)$ retorna o número de folhas da árvore T, e $podar(T,t)$ representa uma árvore obtida podando a ramificação t na árvore T por uma folha adequada.

Para se construir as k árvores como citado anteriormente, deve-se escolher a cada iteração a ramificação que tenha o menor ganho na taxa de erro (α). Encontrado o menor α deve-se criar uma nova árvore reduzida a partir da função $podar(T,t)$ para o menor α. O processo deve ser repetido a partir da árvore podada na iteração anterior até que seja encontrada uma árvore contendo somente o nó raiz. Dessa forma, tem-se uma sequência de tamanho de ramificações e valores de α da seguinte maneira:

$$T_0 > T_1 > T_{...} > T_k \qquad\qquad \alpha_1 < \alpha_2 < \alpha_{...} < \alpha_k \tag{2.19}$$

Pode-se entender dessa sequência que, inicialmente, sub-árvores com mais ramificações tendem a serem eliminadas da árvore, porém, nas iterações seguintes, a tendência é que o número de ramificações eliminadas seja menor.

A partir das árvores geradas deve-se escolher qual a melhor árvore. Para isso precisamos medir o quão generalizadas nossas árvores estão. Tem-se basicamente três métodos para medir isso.

O primeiro é desenvolvido a partir do conjunto de treinamento que avalia o percentual de dados classificados de forma errônea. O seu uso não é recomendado, pois todos os elementos de treinamento são utilizados para construir o modelo e, caso o conjunto de dados possua anomalias, elas irão refletir no modelo tornando-o especialista ao invés de generalista.

O segundo método consiste em dividir o conjunto de dados em um conjunto de treinamento e outro de validação. O seu uso tende a funcionar melhor que o anterior, pois os dados de validação não foram utilizados para criar as árvores em teste, logo, as árvores especialistas ao conjunto de dados de treinamento serão menos precisas. Este mesmo é recomendado apenas quando a quantidade de elementos para a criação do modelo for suficientemente grande para a divisão do conjunto de dados em treinamento em validação. Recomenda-se utilizar $2/3$ dos dados para treinamento e $1/3$ para validação.

Já o terceiro método consiste em dividir o conjunto de dados de treinamento em n partições, técnica que é conhecida como validação cruzada. Para cada n, o conjunto de dados de treinamento deve conter $n-1$ partições e o conjunto de dados de validação deve conter 1 partição. Desta maneira, o conjunto de dados inteiro será avaliado. O problema deste método é que n conjuntos de árvores serão criados ao invés de um. É recomendado utilizá-lo quando o conjunto de dados for pequeno e não for viável dividir o conjunto de dados de treinamento e validação com uma quantidade suficiente de elementos.

Estes três métodos de avaliação das árvores podem ser empregados nas demais técnicas, principalmente os dois últimos. Maiores detalhes sobre estas técnicas de avaliação podem ser encontrados em Breiman et al. [1984] e Rokach & Maimon [2008].

2.3.2.3 Poda baseada no erro reduzido

A poda baseada no erro reduzido, proposta por Quinlan [1987], trabalha das folhas até a raiz verificando em cada nível se a substituição da ramificação por uma folha com a classe mais frequente não reduz a acurácia da árvore. A ramificação é removida se a acurácia da árvore não for reduzida. O procedimeno deve ser repetido até que qualquer outra poda diminua a acurácia da árvore.

Recomenda-se utilizar um conjunto de validação ou validação cruzada para obter resultados satisfatórios com está técnica.

Pode ser demonstrado que esta técnica termina com a menor sub-árvore com maior acurácia sobre um conjunto de dados utilizado para realizar a poda da árvore.

2.3.2.4 Poda pessimista

A poda pessimista, proposta por Quinlan [1987], tem a vantagem de não precisar da utilização de parte do conjunto de dados somente para ser feita a validação. Ao invés disso, um teste de correlação estatística é utilizado para determinar a melhor poda.

A idéia básica deste método é que a taxa de erro encontrada, utilizando o conjunto de treinamento, não é confiável o suficiente(devido as razões citadas anteriormente). Para resolver este problema, uma medida mais realista, conhecida como correção de continuidade para a distribuição binomial, é utilizada:

$$\epsilon'(\mathrm{T,S}) = \epsilon(\mathrm{T,S}) + \frac{\mid \mathrm{folhas(T)} \mid}{2. \mid \mathrm{S} \mid} \tag{2.20}$$

Apesar de mais realista que a taxa de erro baseada no conjunto de treinamento, esta correção ainda produz uma taxa de erro otimista. Consequentemente, Quinlan [1993] sugeriu podar uma ramificação t se a sua taxa de erro está dentro de um erro padrão de uma árvore de referência:

$$\epsilon'(\mathrm{T,S}) \leq \epsilon'(T,S) + \sqrt{\frac{\epsilon'(T,S).(1-\epsilon'(T,S))}{\mid S \mid}} \tag{2.21}$$

A última condição é baseada no intervalo de confiança estatística para proporções. Normalmente, a última condição é usada de maneira que T se refira a uma subárvore cuja raiz é a ramificação t e S denota a parcela do conjunto de treinamento que se refere ao nó t.

O procedimento de poda pessimista faz de "cima para baixo", ao longo do percurso de ramificações. Se uma ramificação é podada, então todos os seus descendentes são removidos do processo de poda, resultando em uma poda relativamente rápida.

2.3.2.5 Poda baseada no erro

Proposta por Quinlan [1993], a poda baseada no erro, é uma evolução da Poda pessimista. Assim como na poda pessimista, a taxa de erro é estimada utilizando o limite superior do intervalo de confiança estatístico para proporções, não sendo necessário a utilização de um subconjunto de dados separados somente para a validação da poda da árvore.

Numa dada ramificação que classifique N casos do conjunto de treinamento, sendo e o número de casos classificados incorretamente, o erro aparente será $\frac{e}{N}$. Pode-se interpretar estes números como representando uma experiência com e eventos em N experiências. Esta perspectiva é pouco correta estatisticamente mas permite obter bons resultado práticos. Apesar de que a partir destes dados, não é possível calcular a probabilidade de ocorrência de erro de forma exata, ela possui uma distribuição de probabilidades a *posteriori* que é normalmente representada por um par de limites de confiança.

Para um dado fator de confiança α, o limite superior desta probabilidade pode ser encontrado a partir dos limites de confiança da distribuição binomial. Estes limites inferior e superior, que se denotam respectivamente por p_i e p_s, representam os valores de probabilidade que permitem dizer que a probabilidade real de um acontecimento (que ocorreu e vezes em N experiências) estar fora do intervalo definido por estes valores é de $1 - \alpha$.

Estes limites inferior e superior podem ser calculados segundo Geigy [1978] com base na distribuição binomial dada por 2.22, de uma forma aproximada, pela expressão 2.24.

$$p(N, x) = \binom{N}{x} p^x (1-p)^{N-x} \tag{2.22}$$

$$p(p_i < p < p_s \mid x = e, N) = 1 - 2\alpha \; ; \; \alpha < 0.5 \tag{2.23}$$

$$p_i, p_s = \frac{x \pm \frac{1}{2} + \frac{c^2}{2} \pm \mid c \mid \sqrt{\left(\pm\frac{x}{2}\right)\left(1 - \frac{x \pm 0.5}{N}\right) + \frac{c^2}{4}}}{N + c^2} \tag{2.24}$$

O valor de c em 2.24 é uma constante que depende do valor de α. Para $\alpha = 0.125$ que é um valor típico, tem-se segundo a tabela da pagina 28 de Geigy [1978], um valor de $c = 1.1503$.

Maiores detalhes sobre este método podem ser conferidos em Quinlan [1993] e da F. Lisboa [1994].

2.4 Trabalhando com atributos contínuos

Conforme mencionado anteriormente, a maioria dos métodos de construção de AD trabalha somente com atributos discretos. Para solucionar este problema um método bastante simples é utilizado por algoritmos como CART Breiman et al. [1984] e C4.5 Quinlan [1993]. O algoritmo tem como objetivo transformar os atributos contínuos em

discretos com vários valores que o discretizem. Para isso, inicialmente todos os valores de um atributo A são ordenados em ordem crescente $\{v_1, v_2, v_3, \ldots, v_m\}$. Para cada par de valores em sequência um novo valor é criado para o atributo. Usualmente, o valor médio de cada intervalo é definido como:

$$\frac{v_i + v_{i+1}}{2} \tag{2.25}$$

Desta maneira, após executar o método, tem-se para o atributo A os seguintes valores "$v_1; \frac{v_1+v_2}{2}$", "$\frac{v_1+v_2}{2}; \frac{v_2+v_3}{2}$","...", "$\frac{v_{m-1}+v_m}{2}; v_m$".

Outro método que pode ser utilizado é definir, manualmente, sub-intervalos ou categorias para um conjunto de dados. Porém, tem-se a desvantagem de requerer alguém que conheça a base de dados e realize a divisão dos intervalos. Em contrapartida, este método pode ser eficiente uma vez que menos intervalos serão gerados em relação ao método anterior.

Outros métodos podem ser conferidos em Fayyad & Irani [1992], Fayyad & Irani [1993], Murthy & Pazzani [1994] e Brodley & Utgoff [1995].

2.5 O algoritmo implementado

O algoritmo implementado neste trabalho foi baseado no C4.5, proposto por Quinlan [1993]. A decisão de utilizar uma implementação do C4.5 ao invés de outro algoritmo de árvore de decisão foi arbitrária. O principal motivo para a escolha do C4.5 foi a popularidade do método. A implementação do trabalho foi comparada com a implementação do weka(algoritmo J48) e os resultados foram os mesmos atestando que a implementação do algoritmo estava correta. Foi implementado uma nova versão do algoritmo devido à integração com o CRIO, porém outras implementações do C4.5 podem ser utilizadas.

Algoritmo 2.1: Algoritmo C4.5 implementado neste trabalho

Input: exemplos,atributoBase,atributos
Output: Árvore de Decisão ajustada aos exemplos

```
1  Crie um nó raiz para a árvore
2  if Todos os exemplos são de uma mesma classe then
3  |  retorne a árvore somente com a raiz, com a classe única dos elementos.
4  end
5  if atributos é uma lista vazia then
6  |  retorne a árvore somente com a raiz, com a classe igual ao valor mais comum do atributoBase nos exemplos.
7  end
8  else
9  |  A ← o atributo da lista de atributos que melhor classifica os exemplos de acordo com o ganho
   |  de informação;
10 |  Atributo de decisão da raiz ← A;
11 |  for cada possível valor, v_i, de A do
12 |  |  Adicione uma nova ramificação abaixo da árvore, correspondente ao teste A = v_i
   |  |  Torne Exemplos_{v_i} o subconjunto de exemplos que tem o valor v_i para A if
   |  |  Exemplos_{v_i} é vazio then
13 |  |  |  Adicione uma folha com a classe mais comum do atributo base no conjunto de
   |  |  |  exemplos.
14 |  |  end
15 |  |  else
16 |  |  |  Adicione a sub-árvore (C4.5(exemplos_{v_i}, atributoBase, atributos - {A})) na
   |  |  |  ramificação.
17 |  |  end
18 |  end
19 end
```

O algoritmo 2.1 trabalha com uma ramificação por vez. Inicialmente é verificado se os exemplos são apenas de uma classe. Caso exista exemplos de classes diferentes é necessário identificar qual o atributo produz o maior ganho de informação. Este atributo é escolhido como o atributo base (raiz) e para cada valor deste atributo é verificado se será criada uma folha ou se o processo de construção será repetido recursivamente para os demais exemplos e atributos. O algoritmo é repetido até que todos os exemplares sejam classificados em folhas ou então todos os atributos sejam utilizados.

Após a construção da árvore é necessário realizar a poda para torná-la mais generalista. O algoritmo 2.2 trabalha a partir da raiz até as folhas. A cada nível é verificado se a taxa de erros da subárvore é maior que a taxa de erros da árvore como um todo. Caso a taxa de erros da subárvore seja maior então está subárvore deve ser substituída por uma folha. O processo é repetido até que todas as possíveis subárvores sejam avaliadas. A árvore restante é a árvore final deste algoritmo.

Algoritmo 2.2: Algoritmo de poda

```
   Input: Uma ramificação
   Output: Ramificação podada
 1 ErrosSubárvore ← calcularErrosDaSubárvore;
 2 ErrosÁrvore ← calcularErrosDaÁrvore;
 3 if ErrosSubárvore ≥ ErrosÁrvore then
 4 |   Remova a ramificação e suas subramificações e substitua por uma folha.
 5 end
 6 else
 7 |   for cada subramificação da ramificação atual do
 8 |   |   algoritmoDePoda(subramificacao);
 9 |   end
10 end
```

Para facilitar o seu entendimento será utilizado um exemplo simples, devido a simplicidade deste problema a poda não foi aplicada.

Para demonstrar o funcionamento do algoritmo, será utilizado o mesmo exemplo descrito em alguns livros como Quinlan [1993], Mitchell [1997]. Suponha que se tem um conjunto de dados já coletados com estatísticas sobre o treinamento de um tenista. Este conjunto de treinamento contém quatro atributos : Condição climática, Temperatura, Umidade, Vento e duas classes: joga ou não joga. Os dados utilizados neste exemplo estão ilustrados na tabela 2.1

Tabela 2.1. Conjunto de dados utilizado no exemplo.

Dia	Condição climática	Temperatura	Umidade	Vento	Joga?
D1	Ensolarado	Quente	Alta	Fraco	Não
D2	Ensolarado	Quente	Alta	Forte	Não
D3	Nublado	Quente	Alta	Fraco	Sim
D4	Chuvoso	Amena	Alta	Fraco	Sim
D5	Chuvoso	Frio	Normal	Fraco	Sim
D6	Chuvoso	Frio	Normal	Forte	Não
D7	Nublado	Frio	Normal	Forte	Sim
D8	Ensolarado	Amena	Alta	Fraco	Não
D9	Ensolarado	Frio	Normal	Fraco	Sim
D10	Chuvoso	Amena	Normal	Fraco	Sim
D11	Ensolarado	Amena	Normal	Forte	Sim
D12	Nublado	Amena	Alta	Forte	Sim
D13	Nublado	Quente	Normal	Fraco	Sim
D14	Chuvoso	Amena	Alta	Forte	Não

Tabela 2.2. Conjunto de dados utilizado no exemplo depois de agrupado.

Dia	Condição climática	Temperatura	Umidade	Vento	Joga?
D4	Chuvoso	Amena	Alta	Fraco	Sim
D5	Chuvoso	Frio	Normal	Fraco	Sim
D6	Chuvoso	Frio	Normal	Forte	Não
D10	Chuvoso	Amena	Normal	Fraco	Sim
D14	Chuvoso	Amena	Alta	Forte	Não
D1	Ensolarado	Quente	Alta	Fraco	Não
D2	Ensolarado	Quente	Alta	Forte	Não
D8	Ensolarado	Amena	Alta	Fraco	Não
D9	Ensolarado	Frio	Normal	Fraco	Sim
D11	Ensolarado	Amena	Normal	Forte	Sim
D3	Nublado	Quente	Alta	Fraco	Sim
D7	Nublado	Frio	Normal	Forte	Sim
D12	Nublado	Amena	Alta	Forte	Sim
D13	Nublado	Quente	Normal	Fraco	Sim

Inicialmente, o objetivo é escolher, entre os quatro atributos, qual é o melhor para a raiz da árvore. Realizando o teste do ganho de informação para os quatro atributos, temos os seguintes valores:

(i) Condição climática = 0.127869204956554

(ii) Temperatura = 0.00970034371883056

(iii) Umidade = 0.106444642548075

(iv) Vento = 0.107013936606488

Portanto, o atributo "Condição climática"é o mais indicado para ser escolhido como a raiz da árvore. Observa-se na tabela 2.2, que os dados são separados em três subconjuntos de dados agrupados pelo valor do atributo "Condição climática". Observe que o último subconjunto, valor igual a "Nublado", contém elementos somente de uma classe, ou seja, uma folha deve ser criada para o último subconjunto. Como restam mais atributos para serem avaliados e os outros subconjuntos não estão totalmente separados, deve-se então continuar expandindo a árvore. A árvore até este momento está ilustrada na figura 2.3.

Para a ramificação com "Condição climática"com o valor "Ensolarado"o melhor atributo, de acordo com o critério do Ganho de Informação, para separar os dados é

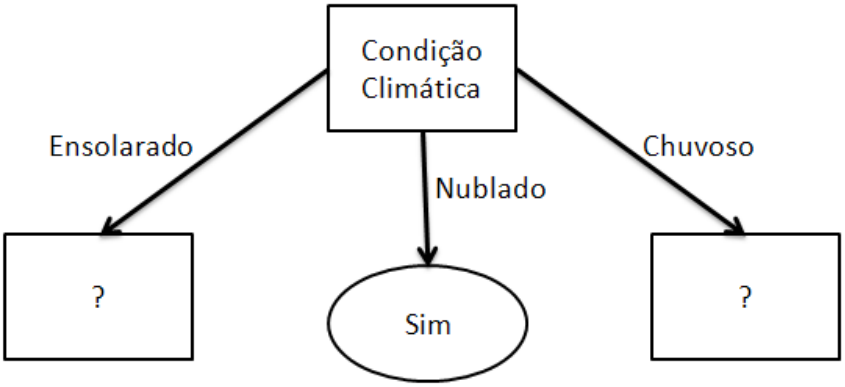

Figura 2.3. Exemplo de construção da árvore, passo inicial.

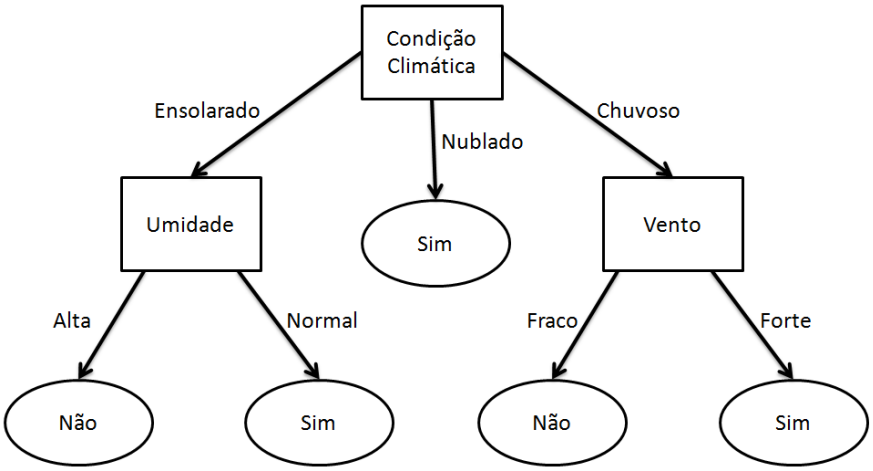

Figura 2.4. Exemplo de construção da árvore, árvore completa.

a "Umidade". Verifique que o atributo "Umidade"divide os dados em dois subconjuntos. O primeiro subconjunto reune todos os exemplos que tem "Umidade"com o valor "Alta"e que a classe é não jogam, enquanto o segundo subconjunto contrém os demais exemplos que tem "Umidade"com o valor "Normal"e pertencem à classe dos que jogam. Portanto mais duas folhas devem ser criadas para nossa árvore.

Por último, vamos avaliar a ramificacão com o valor "Chuvoso". Assim, como na ramificação anterior, para este conjunto de dados tem-se um atributo que divide os objetos em dois subconjuntos distintos. Os exemplos que tem o atributo Vento com o valor igual a "Fraco"são classificados como não jogam, enquanto os que possuem o atributo vento com o valor igual a "Forte"pertencem à classe dos que jogam. A figura 2.4 mostra como a AD para os dados apresentados deve ser.

Após a construção da AD o algoritmo da Poda baseada no erro que foi citado anteriormente na seção 2.3.2.2 é aplicado deixando a AD mais generalista.

Capítulo 3

Resolução do problema de classificação utilizando técnicas de programação linear

3.1 Algoritmo de classificação e regressão linear utilizando técnicas de otimização

Esta seção tem como objetivo descrever o algoritmo CRIO, do inglês "Classification and Regression via Interger Optimization", proposto por Bertsimas & Shioda [2007], toda esta seção é inteiramente baseada no mesmo trabalho.O CRIO é capaz de resolver os problemas de classificação binária e regressão linear utilizando técnicas de programação linear. Para este trabalho somente a parte que resolve o problema de classificação binária foi implementada seguindo o algoritmo proposto.

3.1.1 Visão geral do CRIO

O algoritmo CRIO tem como objetivo resolver os problemas de classificação binária e regressão linear. O problema de regressão linear não será abordado neste trabalho. Para solucionar o problema de classificação binária o CRIO utiliza algumas formulações matemáticas. Devido à complexidade das formulações é necessário agrupar inicialmente os dados em grupos de maneira que o problema possa ser resolvido para cada grupo ao invés de resolvê-lo para cada indivíduo do grupo.

A base de dados de treinamento consiste em n observações (x_i, y_i), i = 1, ..., n com $x_i \in \mathbb{R}^d$ e $y_i \in \{0, 1\}$, onde x_i é um ponto com d dimensões e y_i indica se o ponto

x_i pertence a classe 0 ou 1. Considere m_0 e m_1 como o número de pontos de classe 0 e classe 1, respectivamente. Os pontos de classe 0 e classe 1 são definidos da seguinte forma: x_i^0, i = 1, ..., m_0 e x_i^1, i = 1, ..., m_1. Seguindo o mesmo raciocíno, define-se $M_0 = \{1, ..., m_0\}$ e $M_1 = \{1, ..., m_1\}$ e $K = \{1, ..., k\}$, onde k é um parâmetro que pode ser utilizado para limitar o número de grupos. Cada grupo contém um conjunto de pontos de mesma classe que podem ser agrupados sem que exista nenhum outro ponto de outra classe dentro deste grupo.

Para classificar os dados, o CRIO busca particionar pontos de classe 1 em K grupos disjuntos, de forma que não exista nenhum ponto de classe 0 que possa ser expressado como uma combinação combinatória destes pontos (pontos de classe 1, pertencentes a um K específico).

Considere G_k como o conjunto de índices de pontos de classe 1 que estão associados ao grupo k, onde $\bigcup_{k \in K} G_k = M_1$ e $G_k \bigcap G_{k'} = \emptyset$, k, $k' \in K$, $k \neq k'$. Desta maneira, o seguinte sistema é inviável para todo $i \in M_0$ e $k \in K$:

$$\begin{aligned} \sum_{j \in G_k} \lambda_j x_j^1 &= x_i^0 \\ \sum_{j \in G_k} \lambda_j &= 1 \\ \lambda_j \geq 0, j &\in G_k \end{aligned} \tag{3.1}$$

Em 3.1 x_j^1 e x_i^0 são,respectivamente, os pontos de classe 1 e classe 0, e λ_j é a variável do sistema. Este sistema verifica se existe algum ponto x_i^0 , $i \in M_0$, que seja uma combinação linear de algum ponto x_j^1. Pelo lema de Farkas, o sistema de equações 3.1 é inviável se, e somente se, o sistema a seguir for viável:

$$\begin{aligned} p' x_i^0 + q &< 0 \\ p' x_j^1 + q &\geq 0, j \in G_k \end{aligned} \tag{3.2}$$

No sistema 3.2 p', q são as variáveis. Agora considere o seguinte problema de

otimização:

$$\begin{aligned} z_{k,i} = \text{Maximizar} \quad & \epsilon \\ \text{sujeito a :} \quad & p'x_i^0 + q \leq -\epsilon, \\ & p'x_j^1 + q \geq 0, \qquad j \in G_k \\ & 0 \leq \epsilon \leq 1 \end{aligned} \tag{3.3}$$

Se $z_{k,i} > 0$, conclui-se que o problema 3.3 é viável e o sistema 3.2 é inviável. Se $z_{k,i} = 0$, problema 3.3 é inviável e o sistema 3.2 é viável. Isto significa que existe pelo menos um ponto x_1^0 presente no fecho convexo dos pontos de x_j^1, $j \in G_k$. Expandindo o problema 3.3 para todo $k \in \overline{K}$ e $i \in M_0$ temos:

$$\begin{aligned} z = \text{Maximizar} \quad & \delta \\ \text{sujeito a :} \quad & p'_{k,i}x_i^0 + q_{k,i} \leq -\delta, \qquad i \in M_0; k \in \overline{K}, \\ & p'_{k,i}x_j^1 + q_{k,i} \geq 0, \qquad i \in M_0; k \in \overline{K}; j \in G_k \\ & 0 \leq \delta \leq 1 \end{aligned} \tag{3.4}$$

Para determinar se pontos de classe 1 podem ser associados em K grupos, considerando z > 0, para $k \in \overline{K}$ e $j \in M_1$ são definidas as seguintes variáveis:

$a_{k,j} = 1$, se x_j^1 está associado ao grupo k; 0, caso contrário.
Pode-se verificar se os pontos de classe 1 podem ser particionados em K grupos disjuntos de maneira que não exista nenhum ponto de classe 0 que esteja no seu fecho convexo ao resolver o seguinte problema:

$$\begin{aligned} z = \text{Maximizar} \quad & \delta \\ \text{sujeito a :} \quad & p'_{k,i}x_i^0 + q_{k,i} \leq -\delta, \qquad i \in M_0; k \in \overline{K}, \\ & p'_{k,i}x_i^1 + q_{k,i} \geq a_{k,j} - 1, \qquad i \in M_0; k \in \overline{K}; j \in M_1 \\ & \sum_{k=1}^{k} a_{k,j} = 1, \qquad j \in M_1 \\ & 0 \leq \delta \leq 1 \\ & a_{k,j} \in \{0, 1\} \end{aligned} \tag{3.5}$$

Se $z^* > 0$, a partição em K grupos é viável; enquanto se $z^* = 0$ a partição não é viável.

Desta forma deve-se incrementar o valor de K.

O problema 3.5 tem $Km_0(d+1)+1$ variáveis contínuas, Km_1 variáveis binárias e $Km_0 + Km_0m_1 + m_1$ linhas, onde d é o número de dimensões dos pontos pertencentes a m_0 e m_1. Para grandes números de pontos, o problema 3.5 torna-se de difícil solução(muito esforço computacional para o problema ser resolvido). Ao invés de tratar o problema para cada ponto, pode-se resolvê-lo para agrupamentos de pontos. A partir desta idéia foi criado um algoritmo para o agrupamento dos pontos, que tem como objetivo encontrar pontos que sejam semelhantes de alguma maneira. A diferença entre os problemas de agrupamento e classificação é que no primeiro caso o aprendizado não é supervisionado, diferentemente do segundo caso. Para facilitar o entendimento, no problema de classificação, a classe dos objetos de treinamento é conhecida a priori. Já no problema de agrupamento, busca-se justamente, encontrar os objetos que podem ser agrupados através de alguma semelhança entre eles, ou seja, conhecer quais são as possíveis relações entre os objetos.

3.1.2 O algoritmo de agrupamento

O algoritmo de agrupamento desenvolvido aplica a estratégia de agrupamento hierárquico, onde pontos ou agrupamentos com a menor distância são unidos em um agrupamento maior até que o número desejado de agrupamentos seja alcançado, ou até que não seja mais possível realizar o agrupamento.

Para garantir que não existe ponto de classe 0(classe 1) na junção de dois pontos ou agrupamentos de classe1 (classe 0), foi proposto o seguinte problema de otimização linear, onde r e s são os agrupamentos candidatos a serem unidos:

$$\begin{aligned} \delta^* = \text{Maximizar} \quad & \delta \\ \text{sujeito a :} \quad & p_i' x_i^0 + q_i \leq -\delta, \qquad i \in M_0; \\ & p_i' x_j^1 + q_i \geq \delta, \qquad j \in C_r \cup C_s \end{aligned} \tag{3.6}$$

C_r e C_s são, respectivamente, os índices dos pontos de classe 1 nos agrupamentos r e s. Se $\delta^* > 0$, os agrupamentos r e s podem ser combinados; enquanto se $\delta^* = 0$, os mesmos não podem ser combinados pois existe pelo menos um ponto de classe 0 no fecho convexo da combinação dos agrupamentos.
Para escolher quais agrupamentos serão combinados é utilizada a distância estatística para encontrar os dois agrupamentos mais próximos. A distância estatística é definida a seguir:

Dada um coleção $\mathscr{F}$ de pontos, a distância estatística entre os pontos $x \in \mathscr{F} \subseteq \mathbb{R}^d$ e $z \in \mathscr{F} \subseteq \mathbb{R}^d$ é definida como:

$$d_{\mathscr{F}}(x,z) = \sqrt{\sum_{j=1}^{d} \frac{(x_j - z_j)^2}{s_j^2}} \tag{3.7}$$

onde s_j^2 é a variância da amostra da j-ésima coordenada de todos os pontos em $\mathscr{F}$. A distãncia estatística é amplamente aceita na comunidade de mineração de dados para medir a proximidade dos pontos Bertsimas & Shioda [2007].

O algoritmo de agrupamento 3.1 tem como objetivo agrupar pontos de classe 0(classe 1) de maneira que o agrupamento criado possa ser tratado como um único ponto nas demais etapas do CRIO. Ele deve ser executado para cada conjunto de pontos de classe 0 e de classe 1. Em cada execução, o objetivo é unir o maior número de pontos de mesma classe em grupos, quanto menos grupos melhor. Inicialmente cada ponto é tratado como um grupo e os grupos mais próximos, definido pela distância estatística, são escolhidos para testar se a união é possível. Quando a união for possível um novo grupo é criado unindo os dois grupos. O processo é repetido até que não seja possível unir nenhum grupo.

Algoritmo 3.1: Algoritmo de agrupamento de pontos de acordo com sua classe

Input: Conjunto de pontos M_0 e M_1
Output: Conjunto de agrupamentos K_0 e K_1

```
1  K:= m_1, k:=0
2  while k < K do
3  |  Encontre os agrupamentos (r e s) com a menor distância estatística.
4  |  Resolva o problema 3.6 nos agrupamentos r e s.
5  |  if δ* = 0 then
6  |  |  k:=k+1
7  |  end
8  |  else
9  |  |  Combine os agrupamentos r e s
10 |  |  K:= K-1, k:=0
11 |  end
12 |  k:=k+1
13 end
```

No início do algoritmo, cada ponto é considerado um agrupamento. Desta maneira, $K = M_1$. Na linha 3, a menor distância estatística é calculada a entre os

centros de todos os agrupamentos. Os centros dos mesmos são definidos como a média aritmética de todos os pontos que pertencem ao agrupamento. Na combinação deles, o centro do novo agrupamento é atualizado. Este algoritmo anterior é para encontrar agrupamentos de classe 1, o mesmo deve ser feito para os pontos de classe 0, apenas mudando os parâmetros. A figura 3.1 ilustra o resultado esperado após a execução do algoritmo 3.1.

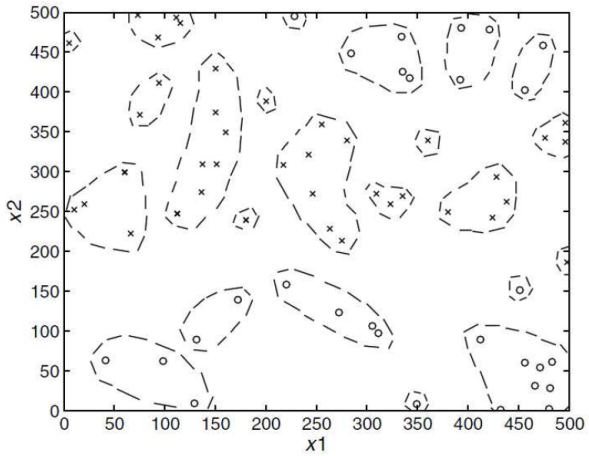

Figura 3.1. Exemplo de agrupamentos. Figura originalmente encontrada em Bertsimas & Shioda [2007]

3.1.3 Remoção de anomalias

Na presença de anomalias, é provável que seja necessário a utilização de um número maior de grupos (K), gerando, possivelmente a sobreposição de agrupamentos. Um ponto pode ser considerado como anomalia caso ele esteja significativamente longe de qualquer outro ponto da mesma classe. A figura 3.2 ilustra um problema que apresenta anomalias. A partir dele tem-se duas abordagens que podem ser feitas: remoção dos agrupamentos que possuem baixa cardinalidade(exemplo, menos de 1% do tamanho total do número de elementos) ou através de uma formulação que tem como objetivo remover estes pontos.

3.1.3.1 Remoção de anomalias por tamanho de agrupamentos

O algoritmo de agrupamento da seção anterior pode encontrar muitos agrupamentos que não podem ser combinados, devido à presença de anomalias. Estas anomalias tem comportamento semelhante, já que estão muito distantes de pontos da mesma classe e não conseguem ser combinadas com muitos pontos. Porém, os agrupamentos de pontos

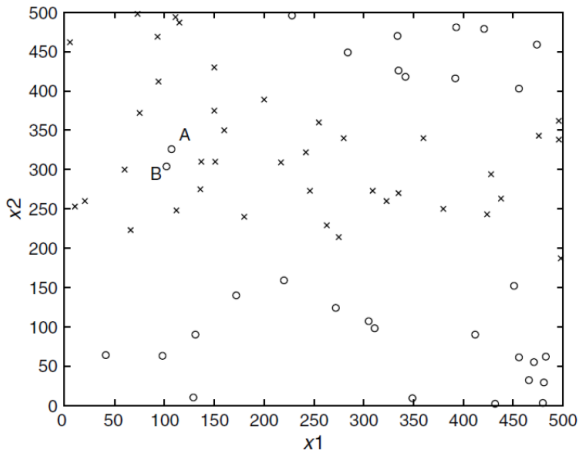

Figura 3.2. Dois outliers A e B entre pontos de outra classe. Figura originalmente encontrada em Bertsimas & Shioda [2007]

que não são anomalias conseguirão ser combinados, apesar de que o agrupamento não será o melhor devido à presença de anomalias.

Devido a estas características, pode-se remover agrupamentos de baixa cardinalidade considerando que estes agrupamentos são anômalos. De acordo com Bertsimas & Shioda [2007], na maioria dos casos os agrupamentos anômalos podem conter apenas um ou poucos pontos, por exemplo 1% do total. Portanto a chance de haver erros na remoção de anômalos é pequena.

3.1.3.2 Remoção de anomalias por programação linear inteira

O problema de otimização 3.8 tem como objetivo remover todas as anomalias que existam no mesmo. As variáveis e_0 e e_1 são associadas,respectivamente, a cada ponto de classe 0 e classe 1. Caso o valor destas variáveis seja maior do que 1, o ponto associado é uma anomalia e deve ser removido.

$$
\begin{aligned}
\text{Minimizar} \quad & \sum_{i=1}^{m_0} e_i^0 + \sum_{i=1}^{m_1} e_j^0 \\
\text{sujeito a:} \quad & p'_{k,t} x_i^0 + q_{k,t} \leq -1 + e_i^0, \qquad i \in C_t^0; t \in \overline{K}_0; k \in \overline{K}, \\
& p'_{k,t} x_i^1 + q_{k,t} \geq -M + (M+1) a_{k,r} - e_j^1, \\
& t \in \overline{K}_0; k \in \overline{K}; r \in \overline{K}_1; j \in C_r^1; \\
& \sum_{k=1}^{K} a_{k,r} = 1 \qquad r \in \overline{K}_1, \\
& a_{k,r} \in \{0,1\}, e_i^0 \geq 0, e_j^1 \geq 0
\end{aligned} \tag{3.8}
$$

A primeira restrição do problema 3.8 exige que $p'_{k,t}x^0_i + q_{k,t}$ seja estritamente negativo, para todo ponto de classe 0. De qualquer maneira, se um ponto x^0_i não puder satisfazer esta restrição, o problema 3.8 permite que a restrição seja violada, por exemplo, $p'_{k,t}x^0_i + q_{k,t}$ pode ser positiva se $e^0_i > 1$. De forma semelhante, $p'_{k,t}x^1_i + q_{k,t}$ deveria ser não negativo quando $a_{k,r} = 1$ e arbítrário quando $a_{k,r} = 0$. Porém utilizando e^1_j a restrição $p'_{k,t}x^1_i + q_{k,t}$ pode ser negativa, desde que e^1_j seja maior que 1. Desta maneira, permitindo que e^0_i e e^1_j sejam maiores que 1 quando necessário o problema 3.8 garante que sempe K grupos de classe 1 serão retornados , ignorando os pontos que, inicialmente, não permitiam o agrupamento. Os pontos com $e^0_i > 1$ e $e^1_j > 1$ podem ser considerados anomalias e serem eliminados.

3.1.4 Associação dos grupos para poliedros

A solução do problema 3.8 resulta em K grupos disjuntos de classe 1, de maneira que não exista qualquer ponto de classe 0 no fecho convexo de qualquer destes grupos. O objetivo agora é representar, geometricamente, cada um destes k grupos com um poliedro P_k. Uma escolha inicial para P_k poderia ser o resultado do problema 3.8:

$$P_k = \{x \in \mathbb{R}^d \mid p'_{k,t}x \geq -q_{k,t} \ , \ k \in \overline{K} \ ; \ t \in \overline{K}_0\} \tag{3.9}$$

Baseado no sucesso das SVM, foi proposto uma solução para o problema. A mesma utiliza hiperplanos que separam os pontos de cada classe de maneira que a menor distância euclidiana de qualquer ponto para o hiperplano seja maximizada. Esta abordagem previne sobreposição de agrupamentos. O objetivo é encontrar o hiperplano:

$$\pi'_{k,t}x = \alpha_{k,t} \tag{3.10}$$

para cada grupo k, $k \in K$, de classe 1 e para cada t, $t \in K_0$, de maneira que todos os pontos no agrupamento t sejam separados de cada ponto no grupo k de modo que a menor distância entre cada ponto e o hiperplano seja maximizada. A distância, $d(x, \pi'_{k,t}x, \alpha_{k,t})$ entre um ponto x e o hiperplano $\pi'_{k,t}x = \alpha_{k,t}$ é $d(x, \pi'_{k,t}x, \alpha_{k,t}) = \frac{|\gamma|}{||\pi_{k,t}||}$, onde $\gamma = \pi'_{k,t}x - \alpha_{k,t}$. Desta maneira podemos maximizar $d(x, \pi'_{k,t}x, \alpha_{k,t})$ fixando $|\gamma|$ e minimizando $||\pi_{k,t}||^2 = \pi'_{k,t}\pi_{k,t}$. Desta maneira basta resolver o problema de otimização quadrático 3.11 .

$$\begin{aligned} w_{k,t_0} = \text{Maximizar} \quad & \pi'_{k,t}\pi_{k,t} \\ \text{sujeito a:} \quad & \pi'_{k,t}x_i^0 \leq \alpha_{k,t}, \qquad i \in C_t^0. \\ & \pi'_{k,t}x_j^1 \leq \alpha_{k,t} + 1, \qquad j \in G_k. \end{aligned} \tag{3.11}$$

Deve-se resolver o problema 3.11 para cada $t \in \overline{K}_0$ e $k \in \overline{K}$, encontrando KK_0 hiperplanos. Desta maneira, para cada grupo k, o poliedro que descreve a região do grupo k é:

$$P_k = \{x \in \mathbb{R}^d \mid \pi'_{k,t}x \leq \alpha_{k,t} \ , \ t \in \overline{K}_0\} \tag{3.12}$$

A figura 3.3 ilustra a solução que pode ser encontrada após a resolução do problema 3.11. É possível verificar que existem restrições redundantes neste problema. Para resolvê-lo,um algoritmo foi proposto,também por Bertsimas & Shioda [2007], sendo descrito na seção seguinte.

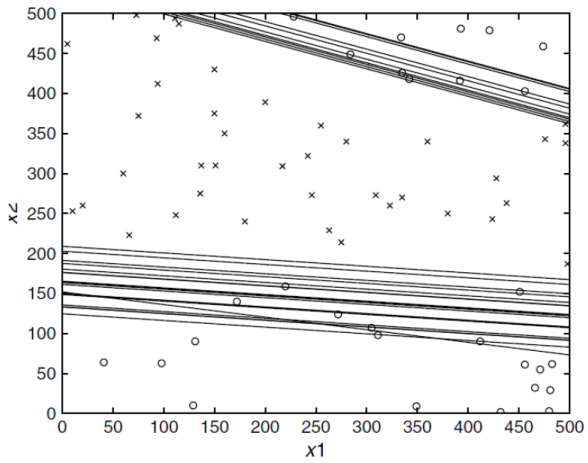

Figura 3.3. Exemplo de restrições encontradas após a resolução do problema 3.11. Figura originalmente encontrada em Bertsimas & Shioda [2007]

3.1.4.1 Remoção de restrições redundantes

Após a definição dos poliedros P_k, $k \in K$, dado pela equação 3.12, pode-se verificar, quando a restrição

$$\pi'_{k,t_0} x \leq \alpha_{k,t_0} \tag{3.13}$$

é redundante e pode ser eliminada resolvendo o seguinte o problema de otimização 3.14 .

$$\begin{aligned} w_{k,t_0} = \text{Maximizar} \quad & \pi'_{k,t_0} x \\ \text{sujeito a:} \quad & \pi'_{k,t} x \leq \alpha_{k,t}, \quad t \in K_0 \backslash \{t_0\}, \\ & \pi'_{k,t_0} x \leq \alpha_{k,t_0} + 1. \end{aligned} \tag{3.14}$$

No problema 3.14, é importante ressaltar que a variável é x, uma vez que os valores de π'_{k,t_0} e $\alpha_{k,t}$, são dados pela equação 3.12. Se $w_{k,t_0} \leq \alpha_{kt0}$, então a restrição 3.13 é coberta pelas demais restrições e portanto, é redundante. Por outro lado, se $w_{k,t_0} > \alpha_{kt0}$, a restrição 3.13 é necessária para descrever o poliedro P_k. O algoritmo 3.2 resume o processo descrito nesta seção.

Algoritmo 3.2: Algoritmo de remoção de restrições redundantes

Input: Conjunto de restrições com possíveis restrições redundantes
Output: Conjunto de restrições sem redundância

```
for k = 1 até K do
    for t_0 = 1 até K_0 do
        Resolva o problema de remoção de restrições redundantes.
        if w_{k,t_0} ≤ α_{k,t_0} then
            Elimine a restrição π'_{k,t_0} x ≤ α_{k,t_0}
        end
    end
end
```

Após a execução do algoritmo tem-se cada poliedro P_k definido sem nenhuma restrição redundante. Para verificar se um ponto qualquer informado pertence a classe 1 ou a classe 0, basta verificar se o ponto atende todas as restrições de um poliedro P_k. Caso atenda o ponto é da classe 1, caso contrário, o ponto é de classe 0.

3.1.5 Visão geral do algoritmo de classificação

O algoritmo para classificação do CRIO é o seguinte:

(i) **Pré-processamento:** utilize o algoritmo de agrupamento detalhado na seção 3.1.2 para encontrar os agrupamentos. Elimine os agrupamentos com cardinalidade menor que 1% de $m_0(m_1)$ para a agrupamentos de classe 0(classe 1).

(ii) **Associar agrupamentos para grupos:** resolva o problema de programação-mista 3.8 para associar agrupamentos de pontos de classe 1 para grupos, enquanto elimina-se prováveis anomalias.

(iii) **Associar grupos para poliedros:** resolva o problema de otimização quadrática 3.11 para encontrar os hiperplanos que definam o poliedro de cada grupo de classe 1.

(iv) **Elimine as restrições redundantes:** remova as restrições redundantes dos poliedros, seguindo o algoritmo descrito na seção 3.1.4.1 .

Após o CRIO determinar o conjunto de restrições não redundantes que determinam os poliedros. O modelo deve ser utilizado para identificar a classe de novos pontos. Se o ponto satisfazer as restrições de qualquer um dos K poliedros, pode-se classificar o ponto como de classe 1. Se o ponto não atende as restrições de qualquer poliedro, deve-se classificá-lo como de classe 0.

Capítulo 4

Algoritmo ADCRIO

4.1 Introdução

A partir dos algoritmos AD e CRIO descritos,respectivamente, nos capítulos 2 e 3, foi desenvolvido um algoritmo que combina o uso das AD com CRIO chamado de ADCRIO. O ADCRIO é bem simples e fácil de ser interpretado, assim como as AD. Uma ilustração em pseudo-código do algoritmo é descrita no algoritmo 4.1.

Algoritmo 4.1: Algoritmo ADCRIO

Input: base de dados
Output: Vários CRIO construídos a partir de uma AD
Construa a AD para a base de dados de treinamento.
Identifique cada ramificação única da AD.
for *Para cada ramificação única da AD* **do**
    Remova os atributos da base de dados de treinamento que não são utilizados pela ramificação. Utilize o CRIO para resolver o problema na base de treinamento tratada no passo anterior. **if** *O CRIO não resolveu o problema* **then**
        Aumente o valor de K e execute o CRIO novamente.
    end
    Associe a solução do CRIO encontrada no passo anterior a todas as folhas que tenham a mesma ramificação atual.
end
for *Para cada instância da base de treinamento* **do**
    Utilize a AD para identificar qual problema do CRIO deve determinar a classe da instância Após encontrar qual solução do CRIO deve determinar a classe, remova os atributos que não são utilizados pela ramificação. Utilize o CRIO para determinar a classe da instância
end

Inicialmente, constrói-se a AD (com ou sem poda da árvore). Em seguida, cria-se um conjunto de ramificações sem que haja repetição a partir da AD criada. Cada uma delas corresponde à lista de atributos utilizada a partir da raiz da AD até uma folha.

Para identificar unicamente uma ramificação basta verificar se o conjuto de atributos dela não é repetido por outra, independente da ordem em que o atributo apareça nela.

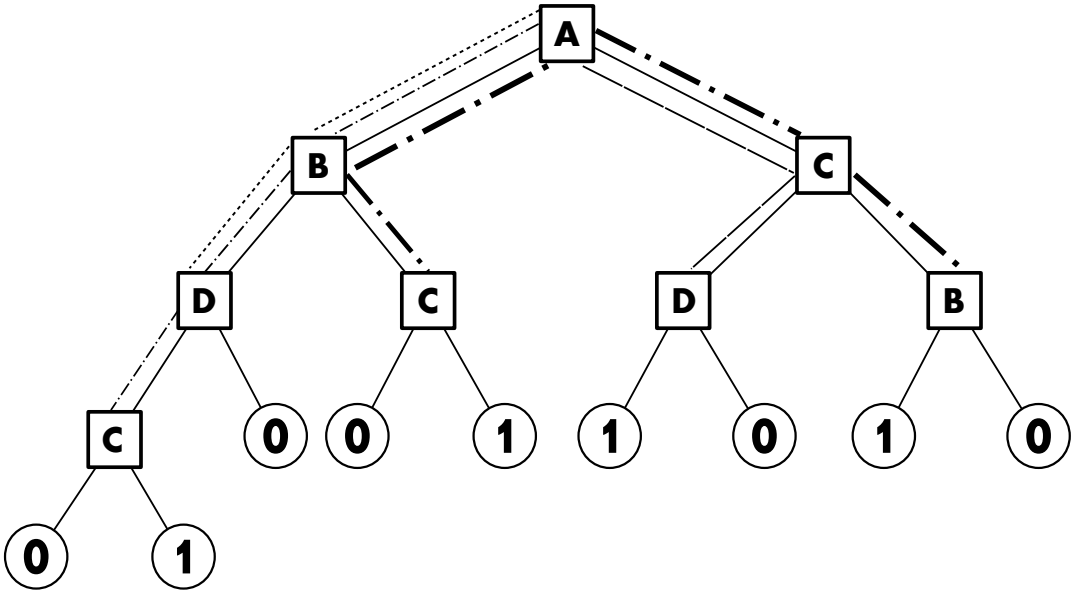

Figura 4.1. Exemplo de uma Árvore de Decisão com ramificações repetidas.

Na figura 4.1 tem-se uma AD com 5 ramificações : {A-B-D-C; A-B-D;A-B-C;A-C-B e A-C-D}, destas 5 ramificações, há uma que se repete (A-B-C e A-C-B). Logo, nosso conjunto de ramificações únicas teria as ramificações {A-B-D-C; A-B-D;A-B-C e A-C-D}. O modelo do CRIO criado para a ramificação A-B-C também deve ser utilizado para a ramificação A-C-B. A figura 4.2 mostra como o exemplo da figura 4.1 ficará após a execução do algoritmo 4.1.

Os principais objetivos do ADCRIO em relação às AD e o CRIO são:

(i) Aumentar a eficácia da AD, uma vez que o CRIO apresenta melhores resultados, do ponto de vista de taxa de acerto dos objetos na base de testes, conforme pode ser visto no trabalho de Bertsimas & Shioda [2007].

(ii) Facilitar a interpretação do resultado encontrado pelo CRIO,pois utilizando a AD torna o resultado interpretável, enquanto no CRIO o resultado é uma caixa preta fornecendo apenas a classe sem nenhuma informação de como ela foi obtida.

(iii) Analisar a eficácia do CRIO utilizando um número reduzido de atributos do problema original.

Existem alguns problemas para integrar as duas técnicas. O primeiro problema é que as AD trabalham com atributos nominais, enquanto o CRIO trabalha com atributos numéricos. Existem formas de resolver este problema transformando a base de

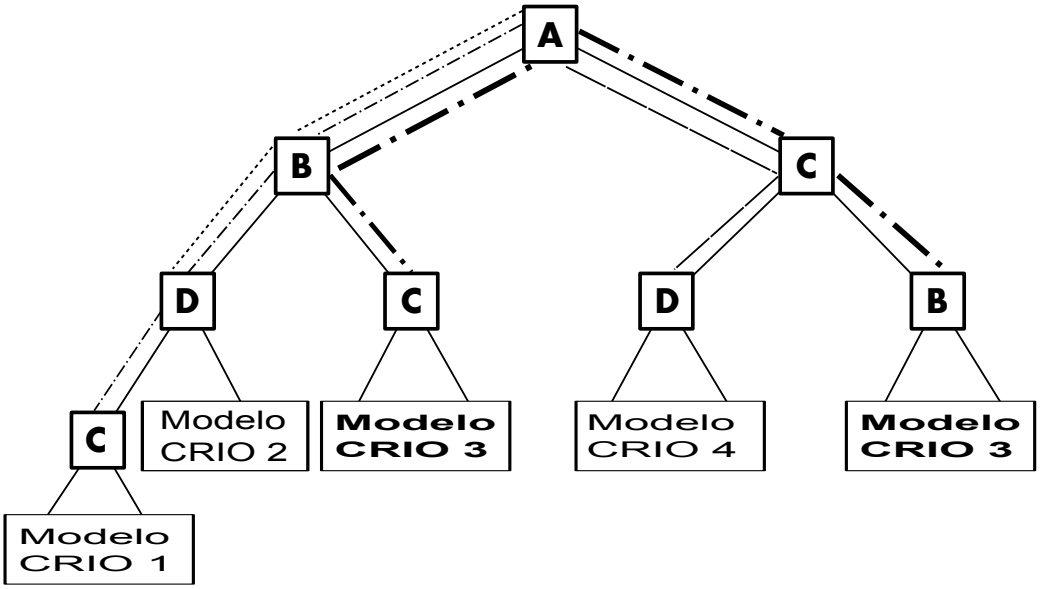

Figura 4.2. Árvore de decisão combinada com o CRIO após execução do algoritmo.

dados numérica para nominal. Desta forma, a AD e o CRIO podem solucionar o mesmo problema sem nenhum empecilho na conversão da base de dados. Porém, esta abordagem pode dificultar o problema uma vez que o processo de identificar os intervalos não é trivial, podendo gerar ruídos na base de dados. Outra questão a ser considerada é que o CRIO ficará limitado apenas ao espaço nominal das AD o que pode dificultar com que o CRIO consiga separar os dados com a mesma eficiência que separa em bases de dados numéricas.

A abordagem utilizada neste trabalho pode ser interpretada de duas maneiras: Do ponto de vista da pesquisa operacional onde quanto menos variáveis mais simples é o problema; e a outra abordagem é do ponto de vista da mineração de dados onde a remoção dos atributos dificulta o problema por conter menos informação que o problema original.

Como o CRIO é baseado em técnicas de programação linear a abordagem adotada neste trabalho visou diminuir a complexidade do problema no número de variáveis uma vez que é preferível executar vários problemas menores do que executar um problema com um alto número de variáveis.

Porém, os sub-problemas resolvidos pelo CRIO são mais complexos que o problema original pois possuem menos variáveis e consequentemente o conhecimento da base de dados dos sub-problemas é menor que o do problema original.

Capítulo 5

Resultados Computacionais e Conclusões

5.1 Introdução

Este capítulo apresenta os resultados obtidos pela AD, pelo CRIO e pelo algoritmo que combina as duas técnicas. Para avaliar as três técnicas o testes foram preparados da seguinte maneira:

(i) Testar bases nominais convertendo cada valor nominal para um número inteiro e único por valor.

(ii) Testar bases nominais e numéricas, convertendo os atributos numéricos para valores binários e os atributos nominais para números inteiros e únicos.

(iii) Testar bases numéricas, convertendo os atributos para binários.

O primeiro item tem como objetivo avaliar o desempenho dos algoritmos sem que problemas de conversão das bases de dados influenciem no resultado. O segundo item visa comparar os resultados em uma base de dados mista onde os atributos numéricos sejam divididos em duas faixas de valores e os atributos nominais não sejam alterados. O último caso é o pior caso para se comparar os dois algoritmos já que todas as instâncias terão seus atributos com apenas dois valores(binários) de forma que o CRIO não poderá aproveitar das características numéricas das instâncias e as AD dependerão do algoritmo de conversão das instâncias para obter bons resultados.

Além de testar diferentes tipos de bases de dados, os métodos são comparados em relação à quantidade dos dados utilizados para treinamento e teste. Não foram utilizados dados para validação pois as AD não tem nenhum parâmetro que precisava

ser ajustado e o CRIO foi executado sempre com o menor valor de K possível devido ao tempo de execução do algoritmo. Para os testes cada base foi dividida em um conjunto com 60% dos dados para treinamento e 40% para testes. Depois os dados de testes foram utilizados para treinamento e os de treinamento para testes. Desta maneira temos um outro parâmetro de comparação em relação à quantidade de dados necessária para obter boa qualidade dos classificadores. Cada base de dados foi separada 5 vezes com os dados de treinamento e testes sendo selecionados de forma aleatória da base original. Os resultados apresentados abaixo apresentam o valor da média encontrada nos 5 testes realizados em cada base de dados.

O restante do capítulo está organizado da seguinte maneira. Primeiramente as bases de dados utilizadas serão apresentadas, em seguidas os resultados obtidos pelas AD, pelo CRIO, pelo algoritmo que combina as AD com o CRIO e por último será realizada a comparação entre as três técnicas. Finalmente serão apresentadas as conclusões e trabalhos futuros.

5.2 Bases de dados

O objetivo desta seção é descrever as bases de dados utilizadas nos experimentos e o formato de dados de cada uma delas para facilitar na interpretação dos resultados obtidos.

5.2.1 Single Proton Emission Computed Tomography (SPECT)

Esta base foi criada por Lukasz A. Kurgan & Goodenday [2001] a partir de exames de pacientes com problemas cardíacos no *Medical College of Ohio (MCO)*. Os exames são realizados utilizando uma tomografia e cada paciente foi classificado por médicos em duas classes: normais e anormais. A partir de 267 imagens foi utilizado um algoritmo para extrair as principais características da imagem e então foram encontrados 45 atributos que são utilizados nesta base para cada registro. O atributo que identifica o diagnóstico do paciente é representado de forma binária, todos os demais atributos são representados por números reais. Das 267 imagens 148 são da classe normal(classe 0) e 119 são da classe anormal(classe 1). Outra base de dados foi criada pelos mesmos autores. Esta base é chamada de SPECTF e contém 349 registros cada um com 23 atributos, todos eles são binários. A divisão das classes para esta base tem 95 registros da classe normal(classe 0) e 254 registros da classe anormal(classe 1). As duas bases são utilizadas nos testes e não possuem nenhuma instância com dados incompletos.

A base SPECTF teve todos os seus atributos transformados em binários utilizando o algoritmo *Discretize* presente no software Weka, Witten et al. [1999], este algoritmo divide os valores de um atributo em uma faixa de valores que pode ser encontrada pelo próprio algoritmo ou informada pelo usuário. Para os testes foi informado o valor 2 para que apenas 2 faixas de valores fossem criadas para cada atributo. A base SPECT não teve nenhuma transformação da base de dados original,ela já era binária. Portanto ela pôde ser utilizada pelo CRIO e as AD sem nenhuma conversão.

5.2.2 Aprovação de crédito

Esta base foi proposta por Quinlan [1987] e contém registros de aplicações de cartão de crédito. Esta base contém atributos numéricos, nominais e reais e difere um pouco das demais bases utilizadas neste trabalho por utilizar atributos nominais com poucos valores (2 e 3) e atributos nominais com muitos valores(9 e 14).

Nesta base, 6 atributos são numéricos e 8 nomimais e a base possui 690 instâncias, 468 delas são de créditos aprovados (classe 0) e 210 de créditos reprovados(classe 1). Além disso, 37 instâncias(5% das instâncias) possuem dados incompletos em pelo menos um atributo. Porém, neste trabalho estes dados foram substituídos pelo valor mais frequente utilizando o algoritmo *AddValues* do weka que utiliza o valor mais frequente onde não exista um valor informado para o atributo.

5.2.3 BUPA

Esta base de dados, criada pelo *BUPA Medical Research Ltd.*, é utilizada para identificar problemas clínicos no fígado de pacientes. Foram utilizados exames de sangue que são considerados sensíveis à problemas no fígado que podem ser originados a partir do consumo excessivo de alcóol.

Esta base contém 345 instâncias e nehuma contém dados incompletos. Das 345 instâncias, 145 são de pacientes que provavelmente terão problemas clínicos(classe 0) e 200 são de pacientes que não terão (classe 1). Esta base de dados contém 6 atributos dos quais 5 são valores reais que representam resultados de exames de sangue de pacientes. O sexto atributo correspondente à quantidade de drinques de bebidas alcoólicas que cada paciente consome por dia sem uma faixa de valores pré-definida. Ou seja, esta base é totalmente numérica. Para utilizá-la nos experimentos foi utilizado o algoritmo *Discretize* do Weka para que a base tivesse cada atributo real convertido para um atributo discreto com apenas dois valores.

5.2.4 Câncer de mama

Esta base de dados, utilizada pela primeira vez no trabalho de Mangasarian & Wolberg [1990], é utilizada para identificar câncer de mama em pacientes. Ela foi construída a partir de imagens de exames de câncer realizados em mamas de pacientes que tinham câncer de tumores. O objetivo é identificar qual o tipo do tumor cada paciente tem: maligno ou beligno.

Esta base contém 683 instâncias e 9 atributos que auxiliam no dianóstico de câncer de mama em duas classe, malígno(444 instâncias que correspondem a 65.0% do total) e benigno (239 instâncias que correspondem a 35% do total). Existe outra base do mesmo autor que possui os mesmos dados mais 16 instâncias, porém estas 16 instâncias possuem dados incompletos e por isso não foram utilizadas neste trabalho.

5.3 Resultados utilizando o algoritmo Árvore de decisão

Esta seção apresenta os resultados computacionais para o problema de classificação obtidos pelo algoritmo de AD. Em todas as avaliações cada base foi dividida em dois conjuntos: um com 60% dos dados para treinamento e 40% para testes e outro com 40% para treinamento e 60% para testes.

5.3.1 SPECT

A tabela de confusão 5.1 exibe os resultados para a base SPECT com 60% dos dados utilizados para treinamento. A poda da AD apresentou um pequeno aumento na taxa de acerto. A taxa total de acerto para a AD sem poda foi de 93.77% e após a poda a taxa de acerto geral aumentou para 93.96%.

Para a AD sem poda a taxa de acerto para a classe 0 foi de 96.01% e para a classe 1 a taxa de acerto foi de 91.34%. Para a AD com poda a taxa de acerto para a classe 0 foi de 95.65% e para a classe 1 a taxa de acerto foi de 92.13%.

Resultados similares forma encontrados para os testes com o modelo criado com 40% da base de dados, porém a poda não teve efeito nos resultados, que foram os mesmos. A tabela de confusão 5.2 exibe os resultados para a base SPECT com 40% dos dados utilizados para treinamento. A taxa total de acerto foi de 92.75%. A taxa

Tabela 5.1. Tabela de confusão para a base SPECT com 60% da base para criação do modelo. Para a classe 0 temos 57.4 objetos e para a classe 1 temos 48.6 objetos, o número de objetos por classe é calculado através da média dos valores nas 5 instâncias. O resultado da AD podada foi um pouco melhor que o resultado da AD sem poda. Para a AD sem poda a taxa total de acerto foi de 93.77% e após a poda da AD a taxa de acerto aumentou para 93.96%. Para a classe 0 a taxa de acerto foi de 96.01% para a AD sem poda e 95.65% para a AD com poda. Para a classe 1 a taxa de acerto foi de 91.34% para a AD sem poda e 92.13% para a AD podada.

	Árvore normal		Árvore podada		Total de itens
	Classe 0	Classe 1	Classe 0	Classe 1	
Classe 0	53.0	4.4	4.0	52.8	57.4
Classe 1	2.2	46.4	2.4	46.8	48.6
Percentual de acerto por classe	96.01	91.34	95.65	92.13	-
Percentual de acerto por instância	93.77		93.96		-

de acerto para a classe 0 foi de 93.88% enquanto para a classe 1 a taxa de acerto foi de 91.23%.

Tabela 5.2. Tabela de confusão para a base SPECT com 40% da base para criação do modelo. Para a classe 0 temos 92.0 objetos e para a classe 1 temos 68.0 objetos, o número de objetos por classe é calculado através da média dos valores nas 5 instâncias. Os resultados da AD podada foram os mesmos da AD sem poda. A taxa total de acerto foi de 92.75%. Para a classe 0 a taxa de acerto foi de 93.88% e para a classe 1 a taxa foi de 91.23%.

	Árvore normal		Árvore podada		Total de itens
	Classe 0	Classe 1	Classe 0	Classe 1	
Classe 0	86.0	6.0	86.0	6.0	92.0
Classe 1	5.6	62.4	5.6	62.4	68.0
Percentual de acerto por classe	93.88	91.23	93.88	91.23	-
Percentual de acerto por instância	92.75		92.75		-

Pode-se notar que na tabela de confusão 5.2 os resultados encontrados foram um pouco inferiores aos resultados encontrados na tabela de confusão 5.1, o que era esperado uma vez que mais objetos de treinamento foram utilizados para construir o modelo da tabela de confusão 5.1.

5.3.2 SPECTF

A tabela de confusão 5.3 exibe os resultados para a base SPECTF com 60% dos dados utilizados para treinamento. A taxa total de acerto para a árvore normal é de 66.48%, sendo aumentada para 74.10% para arvore podada.

Para a árvore normal, temos a taxa de acerto de 74.72% e 63.55% para as classes 0 e 1, respectivamente. Após a poda da árvore pode-se notar o mesmo comportamento dos testes com 60% da base de dados utilizado para treinamento.

A poda da árvore fez com que a classe 1, que possui 73.81% dos dados, fosse a solução para quase todos os casos tornando a taxa de acerto da classe 1 mais alta, 98.44% contra apenas 5.50% de acerto para a classe 0. Este comportamento fez com que a taxa de acerto geral fosse superior à taxa de acerto da árvore normal, porém pode não refletir a realidade do problema, uma vez que a classe 1 foi escolhido como a melhor apenas optando para a classe 1 nos casos de dúvida, isto pode ser observado pela baixa taxa de acerto da classe 0 para a árvore podada.

Tabela 5.3. Tabela de confusão para a base SPECTF com 60% da base para criação do modelo. Para a classe 0 temos 36.4 objetos e para a classe 1 temos 102.6 objetos, o número de objetos por classe é calculado através da média dos valores nas 5 instâncias. O resultados da árvore podada foi superior ao da árvore normal. Para a classe 0 a taxa de acerto foi de 74.72% para a árvore normal e 5.50% para a árvore podada e para a classe 1 a taxa foi de 63.55% para a árvore normal e 98.44% para a árvore podada.

Instância	Árvore normal		Árvore podada		Total de itens
	Classe 0	Classe 1	Classe 0	Classe 1	
Classe 0	27.2	37.4	2.0	1.6	36.4
Classe 1	9.2	65.2	34.4	101.0	102.6
Percentual de acerto por classe	74.72	63.55	5.50	98.44	-
Percentual de acerto por instância	66.48		74.10		-

A tabela de confusão 5.4 exibe os resultados para a base SPECTF com 40% dos dados utilizados para treinamento. A taxa total de acerto para a árvore normal é de 60.38%, sendo aumentada para 74.48% após a arvore ser podada.

Para a árvore normal a taxa de acerto para foi de 92.11% para a classe 0 e 39.23% para a classe 1. Este comportamento foi alterado drasticamente na poda da árvore.

A poda da árvore fez com que a classe 1, que possui 73.81% dos dados para a construção do modelo, fosse a solução para quase todos os casos de dúvidas aumentando dessa maneira a taxa de acertos da classe 1, de 39.23% para 73.14% e diminuindo a

taxa de acerto da classe 0 de 92.11% para 6.16% . Como a classe 1 é a que contém o maior número de exemplares, a taxa de acertos total foi aumentada de 60.38% para 74.48%.

Pode-se concluir, para os dois testes na base SPECTF, que mesmo aumentando o valor da taxa de acerto geral para este problema a poda da árvore apenas generalizou a classe 1 como a classe mais frequente e ignorou a classe 0 na criação do modelo. Desta maneira se no futuro mais exemplares de classe 0 forem testados a chance do modelo errar é considerável, enquanto se uma nova instância de classe 1 surgir a chance de acerto é alta, porém com baixo grau de confiança uma vez que a classe 1 será escolhida na maioria dos casos.

Tabela 5.4. Tabela de confusão para a base SPECTF com 40% da base para criação do modelo. Para a classe 0 temos 48.2 objetos e para a classe 1 temos 161.8 objetos, o número de objetos por classe é calculado através da média dos valores nas 5 instâncias. O resultado da árvore podada foi superior ao da árvore normal. Para a classe 0 a taxa de acerto foi de 92.11% para a árvore normal e 6.16% para a árvore podada e para a classe 1 a taxa foi de 39.23% para a árvore normal e 73.14% para a árvore podada. A poda da árvore aumentou a taxa de acertos de uma maneira geral, porém diminui consideravelmente a taxa de acertos da classe 0.

Instância	Árvore normal		Árvore podada		Total de itens
	Classe 0	Classe 1	Classe 0	Classe 1	
Classe 0	44.4	79.4	2.8	8.2	48.2
Classe 1	3.8	82.4	45.4	153.6	161.8
Percentual de acerto por classe	92.11	39.23	6.16	73.14	-
Percentual de acerto por instância	60.38		74.48		-

5.3.3 BUPA

A tabela de confusão 5.5 exibe os resultados para a base BUPA com 60% dos dados utilizados para treinamento. A taxa total de acerto para a árvore normal é de 45.51%, sendo aumentada para 53.3% para arvore podada.

Analisando os resultados por classe nota-se grande disparidade nos acertos por cada uma. Para a árvore normal tem-se um acerto em 97.27% dos objetos de classe 0. Porém, para a classe 1, tem-se apenas 7.01% de taxa de acerto. Apesar da classe 0 possuir menos objetos nesta base (41.64% dos objetos) ela foi generalizada pela árvore de decisão.

Já para a árvore de decisão podada, o comportamento muda um pouco. Para os objetos de classe 0 a taxa de acerto foi de apenas 33.7% contra 97.27% para a árvore normal. Além disso, observa-se um aumento na taxa de acerto dos objetos de classe 1, de 7.01% para 68.0% do total de objetos. Dessa forma, conclui-se que a poda da árvore fez com que os erros de classe 1 diminuíssem. Porém, teve-se como consequecia o aumentou dos erros de classe 0. Como a maioria da base de dados correspondem a objetos de classe 1, a taxa total de acertos subiu de 43.7% para 53.3%. Mesmo tendo aumentando o número de acertos da classe 1 e diminuindo o número de acertos da classe 0 a poda da árvore não generalizou apenas uma classe como ocorreu nos testes da base de dados SPECTF.

Tabela 5.5. Tabela de confusão para a base BUPA com 60% da base para criação do modelo. Os resultados apresentados são para os exemplares de teste. Para a classe 0 temos 58.8 objetos e para a classe 1 temos 79.2 objetos, o número de objetos por classe é calculado através da média dos valores nas 5 instâncias. A poda da árvore melhorou os resultados da árvore normal. A taxa total de acerto foi de 45.50% para a árvore normal e 53.33% para a árvore podada. Para a classe 0 a taxa de acerto foi de 97.28% com a árvore normal e 33.67% com a árvore podada, já para a classe 1 a de acerto foi de 7.07% para a árvore normal e 67.38% para a árvore podada.

Instância	Árvore normal		Árvore podada		Total de itens
	Classe 0	Classe 1	Classe 0	Classe 1	
Classe 0	57.2	73.6	19.8	25.4	58.8
Classe 1	1.6	5.6	39.0	53.8	79.2
Percentual de acerto por classe	97.28	7.07	33.67	67.93	-
Percentual de acerto por instância	45.50		53.33		-

A tabela de confusão 5.6 exibe os resultados para a base de dados BUPA com 40% dos dados para treinamento. Para a árvore de decisão sem poda a taxa total de acerto é de 44.25%, sendo aumentada para 54.01% na AD podada.

Para a árvore normal foram acertados 97.2% dos objetos de classe 0. Para os objetos de classe 1 apenas 6.46% foram acertados. Estes resultados mostram que apesar da classe 0 ser minoria nesta base a árvore de decisão generalizou os objetos para a classe 0, o que também ocorreu para a base com 60% de treinamento.

Quando a árvore foi podada os resultados melhoram de forma geral. Para a classe 0 temos a taxa de acerto menor, 38,9% contra 97,22%. Para a classe 1 a taxa de acerto

aumenta de 6.5% para 64.7%. A árvore podada fica mais equilibrada porém favorece a classe 1, que é a classe predominante nesta base de dados.

De forma geral, para os dois testes realizados a poda melhora o resultado obtido pela árvore de decisão sem poda. Esta melhora pode ser atribuída à generalização da classe 1 como classe predominante, desta forma aumentaram os erros de classe 0 e diminuiram o erros de classe 1, na equação final o número de erros totais ficou menor. Comparando os testes com 40% e 60% dos dados utilizados para a criação do modelo, obteve-se melhor resultado para o modelo gerado com 40% dos dados após a poda e com a árvore normal o modelo com 60% dos dados obteve um resultado melhor.

Tabela 5.6. Tabela de confusão para a base BUPA com 40% da base para criação do modelo. Os resultados apresentados são para os exemplares de teste. Para a classe 0 temos 86.2 objetos e para a classe 1 temos 120.8 objetos, o número de objetos por classe é calculado através da média dos valores nas 5 instâncias. A poda da árvore melhorou os resultados da árvore normal. A taxa total de acerto foi de 44.25% para a árvore normal e 54.01% para a árvore podada. Para a classe 0 a taxa de acerto foi de 97.21% com a árvore normal e 38.98% com a árvore podada, já para a classe 1 a de acerto foi de 6.46% para a árvore normal e 64.74% para a árvore podada.

Instância	Árvore normal		Árvore podada		Total de itens
	Classe 0	Classe 1	Classe 0	Classe 1	
Classe 0	83.8	113.0	33.6	42.6	86.2
Classe 1	2.4	7.8	52.6	78.2	120.8
Percentual de acerto por classe	97.21	6.46	38.98	64.74	-
Percentual de acerto por instância	44.25		54.01		-

5.3.4 Câncer de mama

A tabela de confusão 5.7 mostra os resultados obtidos pela árvore de decisão para a base de dados de câncer de mama com 60% dos dados utilizados para treinamento. Observa-se que para a árvore sem poda a taxa de acerto para a classe 0 foi de 96.69% e para a classe 1 a taxa de acerto foi de 78.28%. Considerando que a classe predominante é a classe 0 que possui 65.51% do total da base os números apresentados foram satisfatórios e se considerar todos os objetos a taxa total de acerto foi de 90.11%.

Após a poda da árvore, os resultados foram ainda melhores, já que a taxa total de acerto subiu para 94.51%. Mesmo com a diminuição da taxa de acerto diminui de 96.69% para 96.12% para a classe 0,por outro lado, obteve-se um ganho na taxa de acerto da classe 1 que subiu de 78.28% para 91.60%. Os números comprovam que a

poda neste problema foi muito efetiva e não só priorizou a classe mais frequente como encontrou bons resultados para a classe menos frequente.

Tabela 5.7. Tabela de confusão para a base de Câncer de mama com 60% da base para criação do modelo. Para a classe 0 temos 175.4 objetos e para a classe 1 temos 97.6 objetos, o número de objetos por classe é calculado através da média dos valores nas 5 instâncias. O resultado da árvore podada foi superior ao da árvore normal, 90.11% para a árvore normal contra 94.51% da árvore podada. Para a classe 0 a taxa de acerto foi de 96.69% para a árvore normal e 96.12% para a árvore podada e para a classe 1 a taxa foi de 78.28% para a árvore normal e 91.60% para a árvore podada. A poda da árvore foi bastante efetiva não apenas generalizando a classe majoritária.

Instância	Árvore normal		Árvore podada		Total de itens
	Classe 0	Classe 1	Classe 0	Classe 1	
Classe 0	169.6	21.2	168.6	8.2	175.4
Classe 1	5.8	76.4	6.8	89.4	97.6
Percentual de acerto por classe	96.69	78.28	96.12	91.60	-
Percentual de acerto por instância	90.11		94.51		-

A tabela de confusão 5.8 mostra os resultados obtidos pela árvore de decisão para a base de dados de câncer de mama com 40% dos dados utilizados para treinamento.

Para a árvore normal a taxa de acerto para a classe 0 foi de 96.58% e para a classe 1 a taxa de acerto foi de 87.41% e a taxa total de acerto foi de 93.41%.

Para a árvore podada os resultados foram melhores, como já tinha acontecido no modelo anterior. A taxa total de acerto foi de 95.27%. Para a classe 0 a taxa de acerto foi de 95.76% e para a classe 1 a taxa de acerto ficou em 94.3%. A poda apresentou resultado satisfatório novamente para esta base mesmo com a quantidade de objetos de treinamento menor. Os resultados encontrados para a base com 40% foram superiores tanto na árvore normal quanto na árvore normal, após a poda da árvore a taxa de acerto foi praticamente a mesma para objetos de classe 0 e classe 1, comprovando a eficiência da poda da árvore.

5.3.5 Aprovação de crédito

A tabela de confusão 5.9 mostra os resultados obtidos pela árvore de decisão para a base de dados de câncer de mama com 60% dos dados utilizados para treinamento.

Tabela 5.8. Tabela de confusão para a base de Câncer de mama com 40% da base para criação do modelo. Para a classe 0 temos 268.6 objetos e para a classe 1 temos 141.4 objetos, o número de objetos por classe é calculado através da média dos valores nas 5 instâncias. O resultado da árvore podada foi superior ao da árvore normal, 93.41% para a árvore normal contra 95.27% da árvore podada. Para a classe 0 a taxa de acerto foi de 96.58% para a árvore normal e 95.76% para a árvore podada e para a classe 1 a taxa foi de 87.41% para a árvore normal e 94.34% para a árvore podada. A poda da árvore foi bastante efetiva o que pode ser visto na diferença da taxa de acerto da classe 0 e classe 1 que ficou em apenas 1.49%.

Instância	Árvore normal		Árvore podada		Total de itens
	Classe 0	Classe 1	Classe 0	Classe 1	
Classe 0	259.4	17.8	257.2	8.0	268.6
Classe 1	9.2	123.6	11.4	133.4	141.4
Percentual de acerto por classe	96.58	87.41	95.76	94.34	-
Percentual de acerto por instância	93.41		95.27		-

Para a árvore normal a taxa de acerto geral foi de 79.71%. Para a classe 0 foi de 83.44% e para a classe 1 a taxa de acerto foi de 76.60%.

Após a poda da árvore, os resultados foram piores do que os da árvore normal. A taxa de acerto geral que era de 79.71% caiu para 73.55%. Para a classe 0, a taxa de acerto foi de 85.51% e para a classe 1 a taxa de acerto foi de 63.56%. A taxa de acertos aumentou para objetos de classe 0 de 83.44% para 85.51% e diminuiu para objetos de classe 1, de 76.57% para 63.56%.

Tabela 5.9. Tabela de confusão para a base Aprovação de crédito com 60% da base para criação do modelo. Os resultados apresentados são para os exemplares de teste. Para a classe 0 temos 125.6 objetos e para a classe 1 temos 150.4 objetos, o número de objetos por classe é calculado através da média dos valores nas 5 instâncias. A poda da árvore piorou os resultados da árvore normal. A taxa total de acerto foi de 79.71% para a árvore normal e 73.55% para a árvore podada. Para a classe 0 a taxa de acerto foi de 83.44% com a árvore normal e 85.51% com a árvore podada, já para a classe 1 a de acerto foi de 76.60% para a árvore normal e 63.56% para a árvore podada.

Instância	Árvore normal		Árvore podada		Total de itens
	Classe 0	Classe 1	Classe 0	Classe 1	
Classe 0	104.8	35.2	107.4	54.8	125.6
Classe 1	20.8	115.2	18.2	95.6	150.4
Percentual de acerto por classe	83.44	76.60	85.51	63.56	-
Percentual de acerto por instância	79.71		73.55		-

A tabela de confusão 5.10 mostra os resultados obtidos pela árvore de decisão para a base de dados de aprovação de crédito com 40% dos dados utilizados para treinamento. Para a árvore sem poda a taxa de acerto geral foi de 80.82%. Para a classe 0 a taxa de acerto foi de 85.10% e para a classe 1 a taxa de acerto foi de 77.49%.

Para a árvore de decisão podada os resultados também foram piores, como já tinha acontecido no modelo anterior. A taxa total de acerto foi de 78.31%. Para a classe 0 a taxa de acerto foi de 76.71% e para a classe 1 a taxa de acerto ficou em 79.55%.

Para a base de dados de aprovação de crédito a poda da árvore apresentou piores resultados que a árvore normal. Não foi possível estabelecer um padrão na poda da árvore, uma vez que com 60% dos dados utilizados para treinamento a classe 0 teve a taxa de acerto aumentada enquanto a classe 1 teve a taxa de acertos diminuída e para o modelo criado com 40% dos dados para treinamento o comportamento foi inverso.

Tabela 5.10. Tabela de confusão para a base Aprovação de crédito com 40% da base para criação do modelo. Os resultados apresentados são para os exemplares de teste. Para a classe 0 temos 181.2 objetos e para a classe 1 temos 232.8 objetos, o número de objetos por classe é calculado através da média dos valores nas 5 instâncias. A poda da árvore piorou os resultados da árvore normal. A taxa total de acerto foi de 80.82% para a árvore normal e 78.31% para a árvore podada. Para a classe 0 a taxa de acerto foi de 85.10% com a árvore normal e 76.71% com a árvore podada, já para a classe 1 a de acerto foi de 77.49% para a árvore normal e 79.55% para a árvore podada.

Instância	Árvore normal		Árvore podada		Total de itens
	Classe 0	Classe 1	Classe 0	Classe 1	
Classe 0	154.2	52.4	139.0	47.6	181.2
Classe 1	27.0	180.4	42.2	185.2	232.8
Percentual de acerto por classe	85.10	77.49	76.71	79.55	-
Percentual de acerto por instância	80.82		78.31		-

5.4 Resultados utilizando o algoritmo ADCRIO

Esta seção apresenta os resultados computacionais para o problema de classificação obtidos pelo algoritmo de ADCRIO. Em todas as avaliações cada base foi dividida em dois conjuntos: um com 60% dos dados para treinamento e 40% para testes e outro com 40% para treinamento e 60% para testes.

5.4.1 SPECT

Para a base de dados SPECT a poda não apresentou melhoria. Tanto na base com 60% dos dados para treinamento quanto na base com 40% dos dados para treinamento os valores encontrados pelo ADCRIO com poda e sem poda são os mesmos.

A tabela de confusão 5.11 exibe os resultados para a base SPECT com 60% dos dados utilizados para treinamento.A taxa total de acerto foi de 91.70%. A taxa de acerto para a classe 0 foi de 89.85% enquanto para a classe 1 a taxa de acerto foi de 93.70%.

Tabela 5.11. Tabela de confusão para a base SPECT com 60% da base para criação do modelo. Para a classe 0 temos 57.4 objetos e para a classe 1 temos 48.6 objetos, o número de objetos por classe é calculado através da média dos valores nas 5 instâncias. O resultado do ADCRIO podado foi o mesmo do ADCRIO normal, com a taxa de acerto global de 91.70%. Para a classe 0 a taxa de acerto foi de 89.85% e para a classe 1 a taxa foi de 93.70%.

Instância	Árvore normal		Árvore podada		Total de itens
	Classe 0	Classe 1	Classe 0	Classe 1	
Classe 0	49.6	3.2	49.6	3.2	57.4
Classe 1	5.6	47.6	5.6	47.6	48.6
Percentual de acerto por classe	89.85	93.70	89.85	93.70	-
Percentual de acerto por instância	91.70		91.70		-

Resultados similares forma encontrados para os testes com o modelo criado com 40% da base de dados. A tabela de confusão 5.12 exibe os resultados para a base SPECT com 40% dos dados utilizados para treinamento. A taxa total de acerto foi de 81.75%. A taxa de acerto para a classe 0 foi de 72.93% enquanto para a classe 1 a taxa de acerto foi de 93.57%.

Os resultados para os modelos com 60% e 40% dos dados utilizados para treinamento foram bem parecidos. Nos dois casos a poda do ADCRIO não apresentou nenhuma melhoria. A taxa de acerto geral para o ADCRIO gerado com 60% dos dados foi superior ao do ADCRIO com 40% dos dados. Este comportamento é esperado uma vez que a quantidade de dados para a criação do modelo foi superior.

Tabela 5.12. Tabela de confusão para a base SPECT com 40% da base para criação do modelo. Para a classe 0 temos 120 objetos e para a classe 1 temos 40 objetos, o número de objetos por classe é calculado através da média dos valores nas 5 instâncias. O resultado do ADCRIO podado foi o mesmo do ADCRIO sem poda. A taxa de acerto geral foi de 81.75%. Para a classe 0 a taxa de acerto foi de 72.93% e para a classe 1 a taxa foi de 93.57%.

Instância	Árvore normal		Árvore podada		Total de itens
	Classe 0	Classe 1	Classe 0	Classe 1	
Classe 0	66.8	4.4	66.8	4.4	92.0
Classe 1	24.8	64.0	24.8	64.0	68.0
Percentual de acerto por classe	72.93	93.57	72.93	93.57	-
Percentual de acerto por instância	81.75		81.75		-

5.4.2 SPECTF

A tabela de confusão 5.13 exibe os resultados para a base SPECTF com 60% dos dados utilizados para treinamento. A taxa total de acerto para o ADCRIO sem poda foi de 70.21%, sendo aumentada para 73.95% para o ADCRIO podado.

Para o ADCRIO sem poda tem-se um acerto em 25.27% dos objetos de classe 0 e 86.16% para objetos de classe 1.

Para o ADCRIO podado a classe 1 foi generalizada. A taxa de acerto para objetos de classe 0 foi de apenas 4.95% enquanto para a classe 1 a taxa de acerto foi de 98.44%. Isto indica que a classe 1, que era a classe majoritária no modelo de treinamento com 75.00% dos dados, foi generalizada. Apesar da taxa de acerto de 98.44% para a classe 1, esta abordagem não é a ideal uma vez que somente uma classe é priorizada e em casos difíceis de resolver a classe 1 é escolhida por padrão.

A tabela de confusão 5.14 exibe os resultados para a base de dados SPECTF com 40% dos dados para treinamento. Para o ADCRIO sem poda a taxa total de acerto é de 71.14%, sendo aumentada para 75.57% no AD podado.

Para o ADCRIO sem poda tem-se um acerto em 49.80% dos objetos de classe 0 e para a classe 1 a taxa de acerto foi de 77.50%.

Para o ADCRIO podado a classe 1 foi generalizada. A taxa de acerto para objetos de classe 0 foi de apenas 5.81% enquanto para a classe 1 a taxa de acerto foi de 95.06%. O comportamento do modelo com 40% da base de treinamento foi semelhante ao modelo anterior.

Tabela 5.13. Tabela de confusão para a base SPECTF com 60% da base para criação do modelo. Para a classe 0 temos 36.4 objetos e para a classe 1 temos 102.6 objetos, o número de objetos por classe é calculado através da média dos valores nas 5 instâncias. O resultados do ADCRIO podado foram superiores aos do ADCRIO normal. A taxa de acerto geral foi de 70.21% para o ADCRIO normal e 73.95% para o ADCRIO podado. Para a classe 0 a taxa de acerto foi de 25.27% para o ADCRIO normal e 4.95% para o ADCRIO podado e para a classe 1 a taxa de acerto foi de 86.16% para o ADCRIO normal e 98.44% para o ADCRIO podado.

Instância	Árvore normal		Árvore podada		Total de itens
	Classe 0	Classe 1	Classe 0	Classe 1	
Classe 0	9.2	14.2	1.8	1.6	36.4
Classe 1	27.2	88.4	34.6	101.0	102.6
Percentual de acerto por classe	25.27	86.16	4.95	98.44	-
Percentual de acerto por instância	70.21		73.95		-

Tabela 5.14. Tabela de confusão para a base SPECTF com 40% da base para criação do modelo. Para a classe 0 temos 48.2 objetos e para a classe 1 temos 161.8 objetos. Para o ADCRIO sem poda a taxa de acerto foi de 71.14% enquanto para o ADCRIO podado a taxa de acerto aumentou para 75.57%. Para a classe 0 a taxa de acerto foi de 49.80% para o ADCRIO sem poda e 5.81% para o ADCRIO podado. Para a classe 1 a taxa de acerto foi de 77.50% para o ADCRIO sem poda e 95.06% para o ADCRIO podado. Assim como no modelo anterior a classe 1 foi generalizada e como é a classe mais frequente a taxa total de acerto foi aumentada.

Instância	Árvore normal		Árvore podada		Total de itens
	Classe 0	Classe 1	Classe 0	Classe 1	
Classe 0	24.0	36.4	33.6	2.8	48.2
Classe 1	24.2	125.4	45.4	153.8	161.8
Percentual de acerto por classe	49.80	77.50	5.81	95.06	-
Percentual de acerto por instância	71.14		75.57		-

O mesmo comportamento encontrado pelas árvores de decisão foi encontrado pelo ADCRIO na base SPECTF. A generalização da classe 1 melhorou o resultado geral porém o resultado é pouco confiável devido à alta taxa de erro da classe 0.

5.4.3 BUPA

A tabela de confusão 5.15 exibe os resultados para a base BUPA com 60% dos dados utilizados para treinamento. Os resultados obtidos pelo ADCRIO sem poda e com poda foram os mesmos.

A taxa total de acerto foi de 57.97%. Para a classe 0 a taxa de acerto foi de apenas 1.70% e para a classe 1 a taxa de acerto foi de 99.75%. A classe 1 possui a maioria dos objetos de treinamento (58.39%) e essa maioria foi utilizada para criar um modelo que trate a grande maioria dos objetos como de classe 1. A taxa de acerto está diretamente vinculada com a porcentagem de objetos de classe 1 na base de testes, o que comprova que o modelo gerado escolheu uma classe como padrão ignorando a outra classe.

Tabela 5.15. Tabela de confusão para a base BUPA com 60% da base para criação do modelo. Para a classe 0 temos 58.8 objetos e para a classe 1 temos 79.2 objetos, o número de objetos por classe é calculado através da média dos valores nas 5 instâncias. Os resultados para o ADCRIO com e sem poda foram os mesmos. A taxa total de acerto foi de 57.97%. Para a classe 0 a taxa de acerto foi de 1.70% e para a classe 1 a de acerto foi de 99.75%.

Instância	Árvore normal		Árvore podada		Total de itens
	Classe 0	Classe 1	Classe 0	Classe 1	
Classe 0	1.0	0.2	1.0	0.2	58.8
Classe 1	57.8	79.0	57.8	79.0	79.2
Percentual de acerto por classe	1.70	99.75	1.70	99.75	-
Percentual de acerto por instância	57.97		57.97		-

A tabela de confusão 5.16 exibe os resultados para a base de dados BUPA com 40% dos dados para treinamento. Para o ADCRIO sem poda a taxa total de acerto foi de 56.81% e para o ADCRIO podado a taxa total de acerto foi de 57.68%.

Analisando o ADCRIO sem poda temos 2.55% de taxa de acerto para a classe 0 e 95.53% de acerto para a classe 1.

Para o ADCRIO podado a taxa de acerto para a classe 0 foi de 1.16% e para a classe 1 a taxa de acerto foi de 98.01%. A taxa geral de acerto na poda foi superior devido ao aumento na taxa de acerto da classe 1, que é a a classe predominante, 57.40% dos dados de treinamento, neste problema.

Nos dois testes realizados para a base de testes BUPA pode-se notar que o ADCRIO generalizou a classe mais frequente, classe 1, e não criou um modelo capaz de identificar diferentes tipos de objetos. O ADCRIO apenas apostou na classe que tem a maior probabilidade de ocorrer.

Tabela 5.16. Tabela de confusão para a base BUPA com 40% da base para criação do modelo. Para a classe 0 temos 86.2 objetos e para a classe 1 temos 120.8 objetos, o número de objetos por classe é calculado através da média dos valores nas 5 instâncias. A poda do ADCRIO melhorou os resultados do ADCRIO sem poda. A taxa total de acerto foi de 56.81% para o ADCRIO normal e 57.68% para o ADCRIO após a poda. Para a classe 0 a taxa de acerto foi de 2.55% com o ADCRIO normal e 1.16% para o ADCRIO podado. Para a classe 1 a de acerto foi de 95.53% para o ADCRIO sem poda e 98.01% para o ADCRIO podado.

Instância	Árvore normal		Árvore podada		Total de itens
	Classe 0	Classe 1	Classe 0	Classe 1	
Classe 0	2.2	5.4	2.4	1.0	86.2
Classe 1	84.0	115.4	85.2	118.4	120.8
Percentual de acerto por classe	2.55	95.53	1.16	98.01	-
Percentual de acerto por instância	56.81		57.68		-

Neste problema a escolha de uma classe como a "ùnica classe"do problema foi bem nítida. Tanto no modelo com 60% dos dados para treinamento quanto no modelo com 40% dos dados para treinamento o comportamento foi o mesmo. Este tipo de abordagem não pode ser considerada confiável pois a chance de resultados falso positivos é elevada.

5.4.4 Câncer de mama

A tabela de confusão 5.17 mostra os resultados obtidos pela árvore de decisão para a base de dados de câncer de mama com 60% dos dados utilizados para treinamento.

Observa-se que para o ADCRIO sem poda a taxa de acerto geral foi de 91.28%. Para a classe 0 a taxa de acerto foi de 96.12% e para a classe 1 a taxa de acerto foi de 82.58%.

Após a poda da árvore, o ADCRIO obteve resultados piores. A taxa total de acerto diminui para 72.23%. Para a classe 0 a taxa de acerto foi de 58.95% e para a classe 1 a taxa de acerto foi de 96.11%.

A poda do ADCRIO fez com que a classe 1 fosse privilegiada, mesmo a classe 1 sendo minoria neste problema, 34.49% das instâncias de treinamento, o ADCRIO priorizou a classe 1 como a mais frequente, aumentando a taxa de acerto da classe 1 porém diminuindo bastante a taxa de acerto da classe 0, de 96.12% sem a poda para 58.95% com a poda.

Tabela 5.17. Tabela de confusão para a base de Câncer de mama com 60% da base para criação do modelo. Para a classe 0 temos 175.4 objetos e para a classe 1 temos 97.6 objetos, o número de objetos por classe é calculado através da média dos valores nas 5 instâncias. O resultado do ADCRIO sem poda foi superior ao do ADCRIO podado, 91.28% para o ADCRIO sem poda contra 72.23% para o ADCRIO podado. Para a classe 0 a taxa de acerto foi de 96.12% para o ADCRIO sem poda e 58.95% para o ADCRIO podado e para a classe 1 a taxa de acerto foi de 85.58% para o ADCRIO sem poda e 96.11% para o ADCRIO podado.

Instância	Árvore normal		Árvore podada		Total de itens
	Classe 0	Classe 1	Classe 0	Classe 1	
Classe 0	168.6	17.0	103.4	3.8	175.4
Classe 1	6.8	80.6	72.0	93.8.4	97.6
Percentual de acerto por classe	96.12	85.58	58.95	96.11	-
Percentual de acerto por instância	91.28		72.23		-

A tabela de confusão 5.18 mostra os resultados obtidos pelo ADCRIO para a base de dados de câncer de mama com 40% dos dados utilizados para treinamento. Para o ADCRIO sem poda a taxa de acerto geral foi de 92.00% e 82.34% para o ADCRIO após a poda.

Para o ADCRIO sem poda a taxa de acerto para a classe 0 foi de 93.89% e para a classe 1 a taxa de acerto foi de 88.40%.

Para o ADCRIO com poda a taxa de acerto para a classe 0 foi de 75.28 e 95.76 para a classe 1. A poda do ADCRIO, assim como no modelo anterior, melhorou os resultados da classe 1, classe menos frequente, e piorou os resultados da classe 0. Isso fez com que o resultado geral fosse pior para o ADCRIO podado em relação ao ADCRIO sem poda.

5.4.5 Aprovação de crédito

A tabela de confusão 5.19 mostra os resultados obtidos pela árvore de decisão para a base de dados de câncer de mama com 60% dos dados utilizados para treinamento. A taxa de acerto geral do ADCRIO sem poda foi de 82.39% e para o ADCRIO com poda a taxa de acerto geral foi de 70.36%. mais uma vez a poda foi ineficiente para o ADCRIO.

Para o ADCRIO sem poda a taxa de acerto para a classe 0 foi de 82.01% e para a classe 1 a taxa de acerto foi de 82.71%.

Após a poda da árvore, o ADCRIO obteve 85.99% de acerto para para a classe 0 e apenas 57.31 para a classe 1.

Tabela 5.18. Tabela de confusão para a base de Câncer de mama com 40% da base para criação do modelo. Para a classe 0 temos 268.6 objetos e para a classe 1 temos 141.4 objetos, o número de objetos por classe é calculado através da média dos valores nas 5 instâncias. O resultado do ADCRIO sem poda podada foi superior ao do ADCRIO podado, 92.00% para o ADCRIO sem poda contra 95.27% da árvore podada. Para a classe 0 a taxa de acerto foi de 93.89% para o ADCRIO sem poda e 75.28% para o ADCRIO podado e para a classe 1 a taxa de acerto foi de 88.40% para o ADCRIO sem poda e 95.76% para o ADCRIO podado.

Instância	Árvore normal		Árvore podada		Total de itens
	Classe 0	Classe 1	Classe 0	Classe 1	
Classe 0	252.2	16.4	202.2	6.0	268.6
Classe 1	16.4	125.0	66.4	135.4	141.4
Percentual de acerto por classe	93.89	88.40	75.28	95.76	-
Percentual de acerto por instância	92.00		82.34		-

Tabela 5.19. Tabela de confusão para a base Aprovação de crédito com 60% da base para criação do modelo. Os resultados apresentados são para os exemplares de teste. Para a classe 0 temos 125.6 objetos e para a classe 1 temos 150.4 objetos, o número de objetos por classe é calculado através da média dos valores nas 5 instâncias. A poda da árvore piorou os resultados da árvore normal. A taxa total de acerto foi de 79.71% para a árvore normal e 73.55% para a árvore podada. Para a classe 0 a taxa de acerto foi de 83.44% com a árvore normal e 85.51% com a árvore podada, já para a classe 1 a de acerto foi de 76.60% para a árvore normal e 63.56% para a árvore podada.

Instância	Árvore normal		Árvore podada		Total de itens
	Classe 0	Classe 1	Classe 0	Classe 1	
Classe 0	103.0	26.0	108.0	64.2	125.6
Classe 1	22.6	124.4	17.6	86.2	150.4
Percentual de acerto por classe	82.01	82.71	85.99	57.31	-
Percentual de acerto por instância	82.39		70.36		-

A tabela de confusão 5.20 mostra os resultados obtidos pela árvore de decisão para a base de dados de câncer de mama com 40% dos dados utilizados para treinamento. A poda do ADCRIO piorou o resultado geral, da mesma maneira que tinha acontecido no modelo anterior, a taxa de acerto caiu de 82.37% para 78.36%.

Para o ADCRIO sem poda a taxa de acerto para a classe 0 foi de 83.00% e para a classe 1 a taxa de acerto foi de 81.87%.

Para o ADCRIO podado a taxa total de acerto para a classe 0 a taxa de acerto foi de 73.28% e para a classe 1 a taxa de acerto ficou em 82.30%. Novamente a classe

majoritária foi favorecida, a classe 1 possui 54.49% dos objetos utilizados pelo modelo de treinamento.

Tabela 5.20. Tabela de confusão para a base Aprovação de crédito com 40% da base para criação do modelo. Para a classe 0 temos 181.2 objetos e para a classe 1 temos 232.8 objetos, o número de objetos por classe é calculado através da média dos valores nas 5 instâncias. A poda da árvore piorou os resultados do ADCRIO. A taxa total de acerto foi de 82.37% para o ADCRIO sem poda e 78.36% para o ADCRIO com poda. Para a classe 0 a taxa de acerto foi de 83.00% para o ADCRIO sem poda e 73.28% com poda, já para a classe 1 a de acerto foi de 81.87% para o ADCRIO sem poda e 82.30% para o ADCRIO com poda.

Instância	Árvore normal		Árvore podada		Total de itens
	Classe 0	Classe 1	Classe 0	Classe 1	
Classe 0	150.4	42.2	132.8	41.2	181.2
Classe 1	30.8	190.6	48.4	191.6	232.8
Percentual de acerto por classe	83.00	81.87	73.28	82.30	-
Percentual de acerto por instância	82.37		78.36		-

5.5 Resultados utilizando o algoritmo CRIO

Esta seção apresenta os resultados computacionais para o problema de classificação obtidos pelo algoritmo de CRIO. Assim como nas comparações com os algortimos AD e ADCRIO em todas as avaliações cada base foi dividida em dois conjuntos: um com 60% dos dados para treinamento e 40% para testes e outro conjunto com 40% para treinamento e 60% para testes.

5.5.1 SPECT

A tabela de confusão 5.21 exibe os resultados encontrados para a instância SPECT utilizando 60% da base de dados para treinamento. A taxa de acerto total para esta base foi de 94.71%. Para a classe 0 a taxa de acerto foi de 94.60% e para a classe 1 a taxa de acerto foi de 94.84%. Pode-se atribuir o bom resultado do CRIO para esta base ao não favorecimento de apenas uma classe na construção do modelo.

A tabela de confusão 5.22 apresenta os resultados encontrados para os testes com a base SPECT utilizando 40% da base de dados para criação do modelo. A taxa total

Tabela 5.21. Tabela de confusão para a base SPECT com 60% da base para criação do modelo. Para a classe 0 temos 55.2 objetos e para a classe 1 temos 50.8 objetos, o número de objetos por classe é calculado através da média dos valores nas 5 instâncias. A taxa de acerto para o CRIO foi de 94.71% para a instância e 94.60% para a classe 0 e 94.84% para a classe 1.

-	Classe 0	Classe 1	Total de itens
Classe 0	52.6	2.6	55.2
Classe 1	3.0	47.8	50.8
Percentual de acerto por classe	94.60	94.84	-
Percentual de acerto por instância	94.71		-

de acerto foi de 90.86%. Para a classe 0 a taxa de acerto foi de 92.68 e para a classe 1 a taxa de acerto foi de 88.54%. Apesar de obter um bom resultado este modelo foi inferior ao modelo criado com 60% dos dados. Este comportamento é esperado uma vez que mais informação foi fornecida para o modelo com 60% dos dados.

Tabela 5.22. Tabela de confusão para a base SPECT com 40% da base para criação do modelo. Para a classe 0 temos 91.6 objetos e para a classe 1 temos 68.4 objetos, o número de objetos por classe é calculado através da média dos valores nas 5 instâncias. A taxa de acerto para o CRIO foi de 90.86% para a instância e 92.68% para a classe 0 e 88.54% para a classe 1.

-	Classe 0	Classe 1	Total de itens
Classe 0	83.6	6.0	91.6
Classe 1	6.6	61.8	68.4
Percentual de acerto por classe	92.68	88.54	-
Percentual de acerto por instância	90.86		-

5.5.2 SPECTF

A tabela de confusão 5.23 exibe os resultados encontrados para a base SPECTF utilizando 60% dos dados para a criação do modelo. A taxa total de acerto foi de 72.37%. Para a classe 0 a taxa de acerto foi de 47.06% e para a classe 1 a taxa de acerto foi de 80.57%.

A tabela de confusão 5.24 exibe os resultados encontrados para a base de dados SPECTF para o modelo criado com 40% dos dados. A taxa total de acerto foi de

Tabela 5.23. Tabela de confusão para a base SPECTF com 60% da base para criação do modelo. Para a classe 0 temos 36.4 objetos e para a classe 1 temos 102.6 objetos, o número de objetos por classe é calculado através da média dos valores nas 5 instâncias. A taxa de acerto para o CRIO foi de 72.37% para a instância e 47.06% para a classe 0 e 80.57% para a classe 1.

-	Classe 0	Classe 1	Total de itens
Classe 0	16.0	20.4	36.4
Classe 1	18.0	84.6	102.6
Percentual de acerto por classe	47.06	80.57	-
Percentual de acerto por instância	72.37		-

68.47%. Para a classe 0 a taxa de acerto foi de 32.40% e para a classe 1 a taxa de acerto foi de 72.61%.

Nos dois modelos ficou claro que a classe 1 foi generalizada. A classe 1 possui a maioria dos dados, 72.09% para o modelo com 60% dos dados e 73.81% para o modelo com 40% dos dados. Esta generalização refletiu diretamente na taxa de acertos da classe 0.

Tabela 5.24. Tabela de confusão para a base SPECTF com 40% da base para criação do modelo. Para a classe 0 temos 90.2 objetos e para a classe 1 temos 69.8 objetos, o número de objetos por classe é calculado através da média dos valores nas 5 instâncias. A taxa de acerto para o CRIO foi de 90.86% para a instância e 92.68% para a classe 0 e 88.54% para a classe 1.

-	Classe 0	Classe 1	Total de itens
Classe 0	7.0	51.6	58.6
Classe 1	14.6	136.8	151.4
Percentual de acerto por classe	32.40	72.61	-
Percentual de acerto por instância	68.47		-

5.5.3 BUPA

A tabela de confusão 5.25 apresenta os resultados para a base de dados BUPA com 60% dos dados utilizados para treinamento. A taxa total de acerto para este modelo foi de 57.39%. Para a classe 0 a taxa de acerto foi de 50.00 e para a classe 1 a taxa de acerto foi de 57.59%.

Tabela 5.25. Tabela de confusão para a base BUPA com 60% da base para criação do modelo. Para a classe 0 temos 58.8 objetos e para a classe 1 temos 79.2 objetos, o número de objetos por classe é calculado através da média dos valores nas 5 instâncias. A taxa de acerto para o CRIO foi de 57.39% para a instância e 50.00% para a classe 0 e 57.59% para a classe 1.

-	Classe 0	Classe 1	Total de itens
Classe 0	1.8	57.0	58.8
Classe 1	1.8	77.4	79.2
Percentual de acerto por classe	50.00	57.59	-
Percentual de acerto por instância	57.39		-

A tabela 5.26 exibe os resultados para a base de dados BUPA utilizando 40% dos dados para treinamento. A taxa total de acerto foi de 57.00%. Para a classe 0 a taxa de acerto foi de 26.67% e para a classe 1 a taxa de acerto foi de 57.91%.

Tabela 5.26. Tabela de confusão para a base BUPA com 40% da base para criação do modelo. Para a classe 0 temos 86.2 objetos e para a classe 1 temos 120.8 objetos, o número de objetos por classe é calculado através da média dos valores nas 5 instâncias. A taxa de acerto para o CRIO foi de 57.00% para a instância e 26.67% para a classe 0 e 57.91% para a classe 1.

-	Classe 0	Classe 1	Total de itens
Classe 0	1.6	84.6	86.2
Classe 1	4.4	116.4	120.8
Percentual de acerto por classe	26.67	57.91	-
Percentual de acerto por instância	57.00		-

Nos dois modelos a taxa de acerto foi muito baixa. Apesar dos resultados para a classe 1 terem sido ruins houve uma generalização do modelo para a classe 1. Como o modelo tem em torno de 58% dos objetos de classe 1, a taxa geral de acertos foi baixa.

5.5.4 Câncer de mama

A tabela de confusão 5.27 exibe os resultados para a base de dados de Câncer de mama utilizando 60% dos dados para criar o modelo. A taxa de acerto geral foi de 97.73. Para a classe 0 a taxa de acerto foi de 98.29% e para a classe 1 a taxa de acerto foi de 96.74%.

Tabela 5.27. Tabela de confusão para a base Câncer de mama com 60% da base para criação do modelo. Para a classe 0 temos 175.2 objetos e para a classe 1 temos 97.8 objetos, o número de objetos por classe é calculado através da média dos valores nas 5 instâncias. A taxa de acerto para o CRIO foi de 97.73% para a instância e 98.29% para a classe 0 e 96.74% para a classe 1.

-	Classe 0	Classe 1	Total de itens
Classe 0	172.0	3.2	175.2
Classe 1	3.0	94.8	97.8
Percentual de acerto por classe	98.29	96.74	-
Percentual de acerto por instância	97.73		-

A tabela de confusão 5.28 exibe os resultados para a base de dados de Câncer de mama utilizando 40% dos dados para criar o modelo. A taxa de acerto geral foi de 96.40. Para a classe 0 a taxa de acerto foi de 97.39% e para a classe 1 a taxa de acerto foi de 94.52%.

Tabela 5.28. Tabela de confusão para a base Câncer de mama com 40% da base para criação do modelo. Para a classe 0 temos 268.6 objetos e para a classe 1 temos 141.4 objetos, o número de objetos por classe é calculado através da média dos valores nas 5 instâncias. A taxa de acerto para o CRIO foi de 96.40% para a instância e 97.39% para a classe 0 e 94.52% para a classe 1.

-	Classe 0	Classe 1	Total de itens
Classe 0	260.8	7.8	268.6
Classe 1	7.0	134.4	141.4
Percentual de acerto por classe	97.39	94.52	-
Percentual de acerto por instância	96.40		-

Tanto para o modelo com 60% dos dados quanto para o modelo com 40% dos dados os resultados foram satisfatórios. Não houve generalização de nenhuma classe mesmo com a classe 0 possuindo aproximadamente 65% dos dados desta base de dados.

5.5.5 Aprovação de crédito

A tabela de confusão 5.29 exibe os resultados para a base de dados de Aprovação de crédito utilizando 60% dos dados para criar o modelo. A taxa de acerto geral foi de 85.72. Para a classe 0 a taxa de acerto foi de 81.15% e para a classe 1 a taxa de acerto foi de 90.37%.

Tabela 5.29. Tabela de confusão para a base Aprovação de crédito com 60% da base para criação do modelo. Para a classe 0 temos 126.0 objetos e para a classe 1 temos 150.0 objetos, o número de objetos por classe é calculado através da média dos valores nas 5 instâncias. A taxa de acerto para o CRIO foi de 85.72% para a instância e 81.15% para a classe 0 e 90.37% para a classe 1.

-	Classe 0	Classe 1	Total de itens
Classe 0	112.8	13.2	126.0
Classe 1	26.2	123.8	150.0
Percentual de acerto por classe	81.15	90.37	-
Percentual de acerto por instância	85.72		-

A tabela de confusão 5.30 exibe os resultados para a base de dados de Aprovação de crédito utilizando 40% dos dados para criar o modelo. A taxa de acerto geral foi de 83.53. Para a classe 0 a taxa de acerto foi de 77.67% e para a classe 1 a taxa de acerto foi de 89.23%.

Tabela 5.30. Tabela de confusão para a base Aprovação de crédito com 40% da base para criação do modelo. Para a classe 0 temos 181.2 objetos e para a classe 1 temos 232.8 objetos, o número de objetos por classe é calculado através da média dos valores nas 5 instâncias. A taxa de acerto para o CRIO foi de 83.53% para a instância e 77.67% para a classe 0 e 89.23% para a classe 1.

-	Classe 0	Classe 1	Total de itens
Classe 0	158.6	22.6	181.2
Classe 1	45.6	187.2	232.8
Percentual de acerto por classe	77.67	89.23	-
Percentual de acerto por instância	83.53		-

Os resultados também são satisfatórios. Porém pode-se notar um ligeiro favorecimento para a classe 1, que é a classe majoritária nos dois modelos com aproximadamente 55% dos dados. Este favorecimento pode ser explicado através da taxa de acerto para a classe 1 que foi superior à taxa de acerto da classe 0 nos dois modelos.

5.6 Comparação entre os algoritmos

Nesta seção os algoritmos serão comparados por cada base de dados testada e no final um sumário das comparações será apresentado. Cada seção apresentará um gráfico com os respectivos resultados encontrados pelos algoritmos em cada base de dados.

5.6.1 Comparações na base de dados SPECT

Os resultados para os testes com 60% da base utilizada como treinamento são exibidos na figura 5.1. O algoritmo CRIO apresentou os melhores resultados seguido pela AD podada, AD sem poda e por último o ADCRIO(com e sem poda). A diferença entre o melhor e o pior algoritmo para esta base, CRIO e ADCRIO respectivamente, foi de apenas 3.17%.

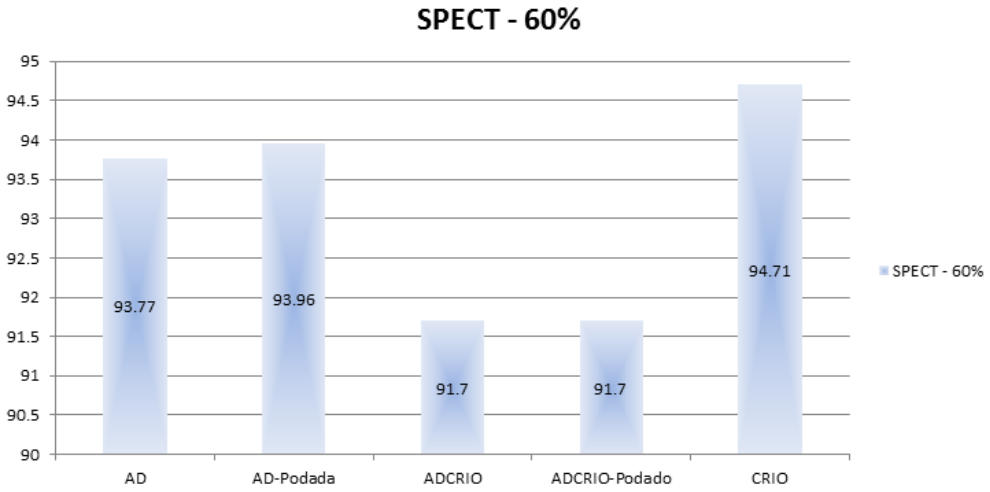

Figura 5.1. Comparação dos resultados para a base SPECT com 60% dos dados utilizados para criar o modelo. O algoritmo CRIO apresentou os melhores resultados seguido pelos algoritmos AD podada, AD sem poda, ADCRIO com e sem poda.

A figura 5.2 apresenta os resultados para os testes utilizando 40% dos dados para treinamento a AD(normal e podada) obteve os melhores resultados seguida pelo algoritmo CRIO e por último o ADCRIO(normal e podado). A diferença entre a melhor solução, AD, e a pior solucão, ADCRIO, foi de 11.86%. A diferença entre os resultados da AD e do CRIO foi de apenas 2.03%. Pode-se atribuir o baixo desempenho do ADCRIO ao favorecimento da classe 1 e consequentemente resultados inferiores para a classe 0. É interessante ressaltar que a classe mais frequente para esta base é classe 0 com 54.15% e que em geral a classe mais frequente é a classe generalizada, o que não ocorreu neste caso.

Era esperado que o CRIO obtivesse o melhor desempenho, e foi o que aconteceu com o modelo gerado com 60% da base para treinamento, porém para o modelo gerado com 40% da base para treinamento a AD superou o CRIO. O ADCRIO obteve resultados insatisfatórios para o modelo com 40% dos dados. Isto pode ser atribuído ao número reduzido de atributos utilizados para a geração do modelo.

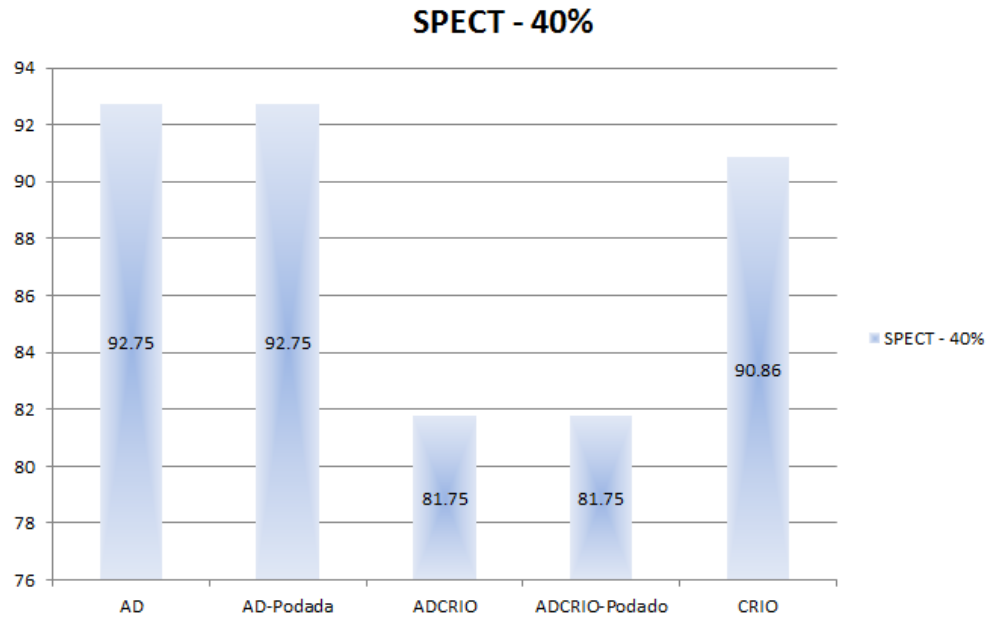

Figura 5.2. Comparação dos resultados para a base SPECT com 40% dos dados utilizados para criar o modelo. O algoritmo AD(com e sem poda) apresentou os melhores resultados seguido pelos CRIO e ADCRIO com e sem poda.

O algoritmo que obteve os melhores resultados para este problema foi o CRIO para o modelo criado com 60% da base de dados e a AD para o modelo com 40% da base de dados. De forma geral, como a diferença entre o CRIO e a AD foi pequena, apenas 0.79% para a AD com poda, é recomendado a utilização da AD para a base de dados SPECT uma vez que a AD é o algoritmo mais eficiente entre os algoritmos comparados.

5.6.2 Comparações na base de dados SPECTF

A figura 5.3 apresenta os resultados para os testes utilizando 60% dos dados para treinamento. O algoritmo AD podada obteve os melhores resultados. Em seguida vieram os algoritmos ADCRIO podado, CRIO, ADCRIO sem poda e AD sem poda. É interessante notar que o melhor algoritmo e o pior são o mesmo, a diferença é apenas a poda que fez com que o algoritmo sem poda fosse 10.28% inferior ao mesmo algoritmo com poda. O algoritmo ADCRIO sem poda foi 5.25% pior que o algoritmo AD com poda. O algoritmo CRIO ficou 2.33% inferior ao AD com poda e por último o ADCRIO com poda ficou apenas 0.20% inferior ao AD com poda.

A figura 5.4 apresenta os resultados para os testes utilizando 40% dos dados para treinamento o ADCRIO podado obteve o melhor resultado. Em seguida vieram os algoritmos AD com poda, ADCRIO sem poda, CRIO e por último a AD sem poda.

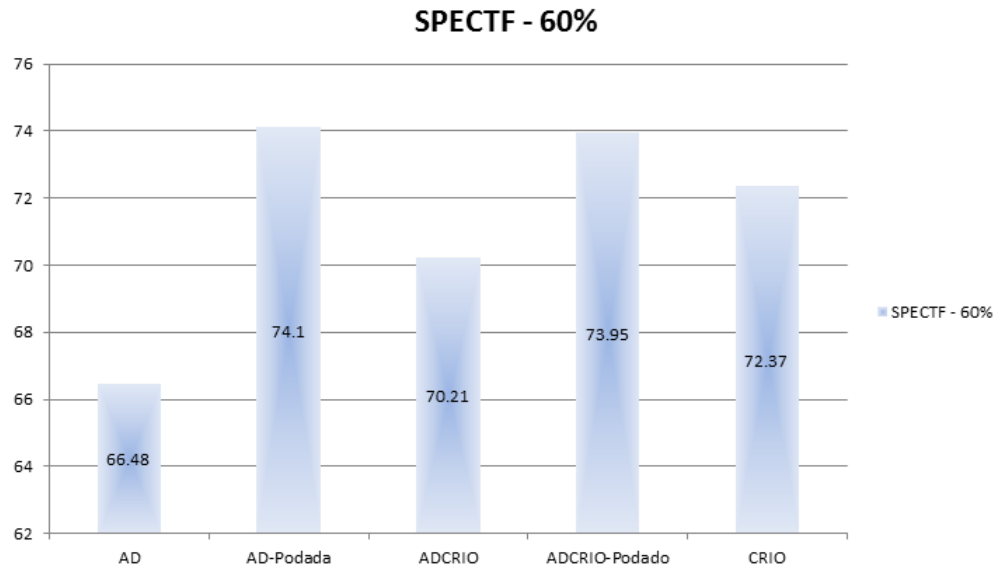

Figura 5.3. Comparação dos resultados para a base SPECTF com 60% dos dados utilizados para criar o modelo. O algoritmo AD com poda apresentou os melhores resultados seguido pelos algoritmos ADCRIO com poda, CRIO, ADCRIO sem poda e AD sem poda.

Novamente a diferença entre o ADCRIO com poda e a AD com poda foi pequena, apenas 1.31%. Porém a diferença entre o melhor e o pior algorimto praticamente dobrou. A diferença entre o ADCRIO podado e a AD sem poda foi de 20.10%. O algoritmo CRIO também não apresentou bons resultados, ficando 9.58% abaixo do ADCRIO com poda. E por último a diferença para o ADCRIO com e sem poda foi de 5.86%.Pode ser observado que mesmo sem a poda o ADCRIO foi um algoritmo com bom rendimento, superando o CRIO mesmo sem a poda.

Analisando os dois modelos para a base de testes SPECTF pode-se concluir que o algoritmo AD com poda é o mais indicado para este problema, assim como tinha ocorrido na base SPECT. Mesmo o ADCRIO com poda obtendo o melhor resultado no modelo com 40% dos dados, a diferença para a AD com poda foi pequena. Além da pequena diferença o algoritmo da AD é o mais eficiente dos três, logo a AD é o algoritmo recomendado para a base de dados SPECTF.

5.6.3 Comparações na base de dados BUPA

O resultados dos testes da base de dados BUPA são apresentados na tabela 5.5. Neste problema o ADCRIO(com e sem poda) ganhou nos testes realizados sendo seguido pelo CRIO, AD com poda e AD sem poda.

A diferença entre o ADCRIO(com e sem poda) e o CRIO foi de apenas 0.97%.

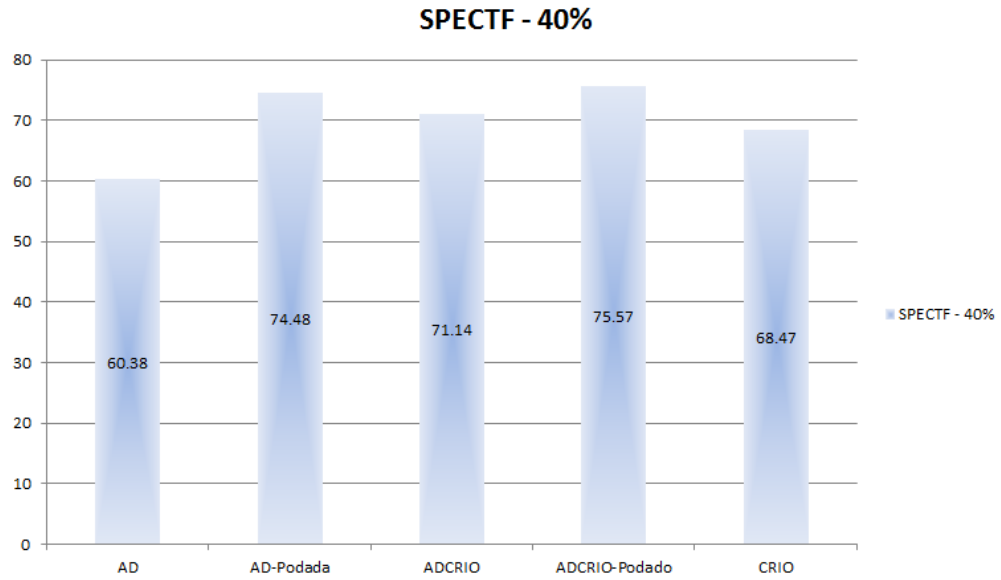

Figura 5.4. Comparação dos resultados para a base SPECTF com 40% dos dados utilizados para criar o modelo. O algoritmo ADCRIO com poda apresentou os melhores resultados seguido pelos algoritmos AD com poda, ADCRIO sem poda, CRIO e AD sem poda.

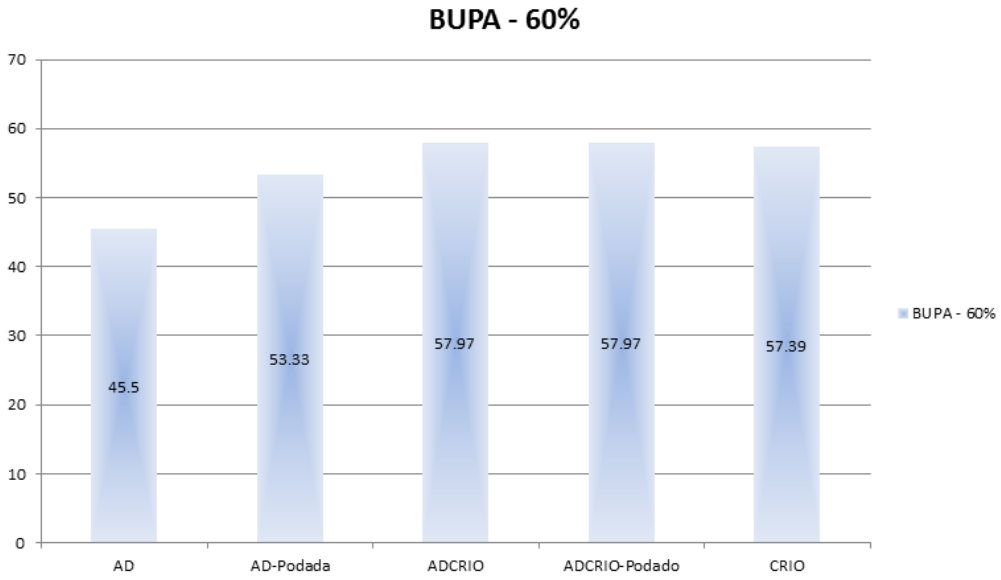

Figura 5.5. Comparação dos resultados para a base BUPA com 60% dos dados utilizados para criar o modelo. O algoritmo ADCRIO com e sem poda apresentou os melhores resultados seguido pelos algoritmos CRIO, AD com poda e AD sem poda.

Em seguida a AD sem poda obteve 7.97% de diferença para o ADCRIO e por último a AD com poda obteve 21.48% de diferença para o ADCRIO. Um comportamento interessante pode ser notado para esta base: apesar do ADCRIO ser relacionado com

a AD, a poda da árvore não afetou os resultados do ADCRIO. Por outro lado, a AD teve diferença de 14.68% para as versões com e sem poda.

Para os testes com 40% da base utilizada como treinamento o algoritmo ADCRIO com poda apresentou os melhores resultados, como já tinha acontecido para os testes com 60% da base de dados, seguido pelo CRIO, ADCRIO sem poda, AD com poda e AD sem poda. A diferença entre o ADCRIO com poda e o CRIO foi de 1.18%, entre as duas versões do ADCRIO a diferença foi de 1.50%. Na comparação com o AD com poda a diferença foi de 6.37% e por último a diferença para o AD sem poda foi de 23.28%. Através destes resultados observa-se que a poda da AD é fundamental neste problema. Para o ADCRIO a diferença entre as versões com poda e sem poda é pequena, porém para a AD, assim como já tinha acontecido no teste anterior, a diferença nesta base de dados foi bastante elevada. A diferença entre a taxa de acerto entra a AD com poda e a AD sem poda foi de 18.07%.

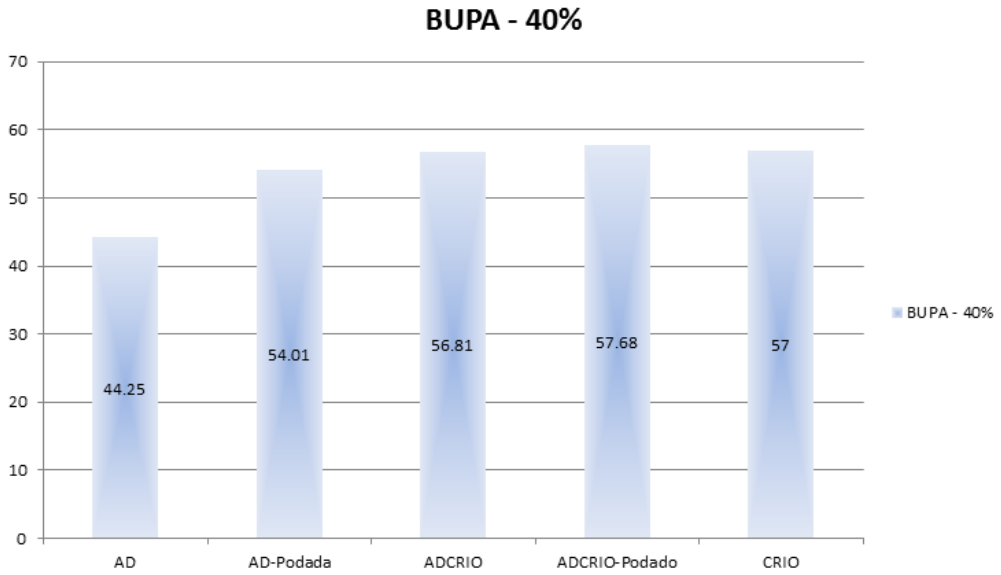

Figura 5.6. Comparação dos resultados para a base BUPA com 40% dos dados utilizados para criar o modelo. O algoritmo ADCRIO com poda foi o melhor. Em seguida vieram os algoritmos CRIO, ADCRIO sem poda, AD com poda e AD sem poda.

Para este problema a melhor solução encontrada foi a do algoritmo ADCRIO com poda. Em seguida veio o algoritmo CRIO. Como o ADCRIO é mais eficiente, é recomendável a utilização dele para este problema.

5.6.4 Comparações na base de dados Câncer de mama

O resultados dos testes da base de câncer de mama são apresentados na tabela 5.7. Para os testes com 60% da base utilizada como treinamento o algoritmo CRIO apresentou os melhores resultados seguido pelo AD com poda, ADCRIO sem poda, ADC sem poda e por último o ADCRIO podado. A diferença entre o CRIO e a AD com poda foi de 3.29% e do CRIO para o ADCRIO sem poda a diferença foi de 6.60%. Para a AD sem poda a diferença ficou em 7.79% e para o ADCRIO com poda a diferença subiu para 26.09%. Este teste apresenta um comportamento interessante. A poda da árvore favoreceu o AD, aumentando a taxa de acerto em 4.88%. Porém para o ADCRIO a taxa de acerto caiu em 20.87%. Essa queda na taxa de acertos pode ser atribuída ao número reduzido de atributos que são utilizados para o CRIO resolver cada subproblema gerado pelo ADCRIO.

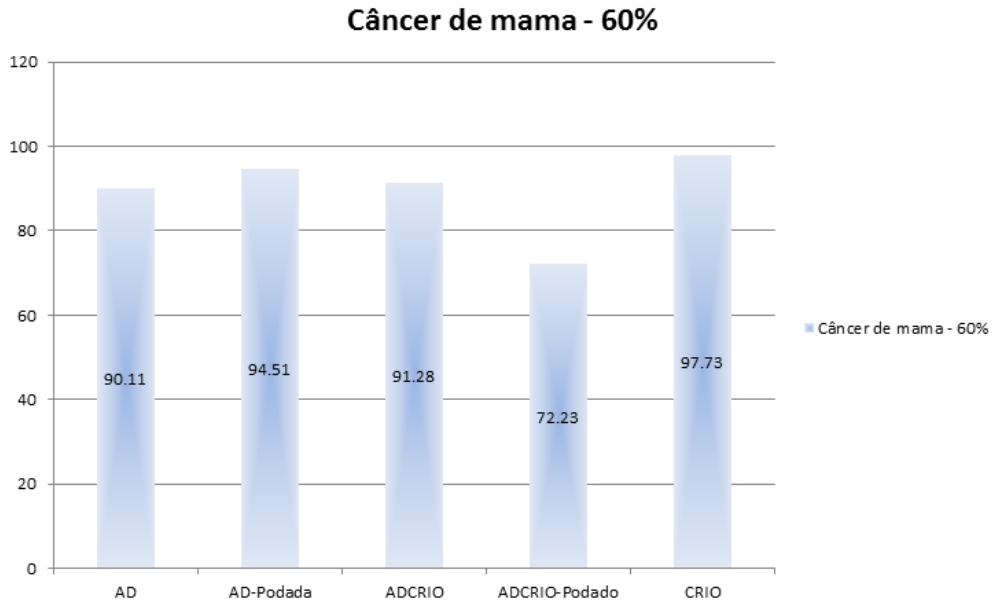

Figura 5.7. Comparação dos resultados para a base de Câncer de mama com 60% dos dados utilizados para criar o modelo. O algoritmo CRIO obteve os melhores resultados seguido pelos algoritmos AD com poda, ADCRIO sem poda, AD sem poda e ADCRIO com poda.

Para os testes com 40% da base utilizada como treinamento, o comportamento foi semelhante ao do teste anterior. O algoritmo o algoritmo CRIO apresentou os melhores resultados seguido pelo AD com poda, AD sem poda, ADCRIO sem poda e por último o ADCRIO com poda. A diferença entre a solução do CRIO e do AD com poda foi de apenas 1.72%. Para a AD sem poda a diferença foi de 3.10%. Em seguida o ADCRIO sem poda teve 4.56% de diferença para o CRIO e por último o ADCRIO com poda

que teve diferença de 14.58% para a solução do CRIO. Novamente a poda favoreceu o AD e desfavoreceu o ADCRIO.

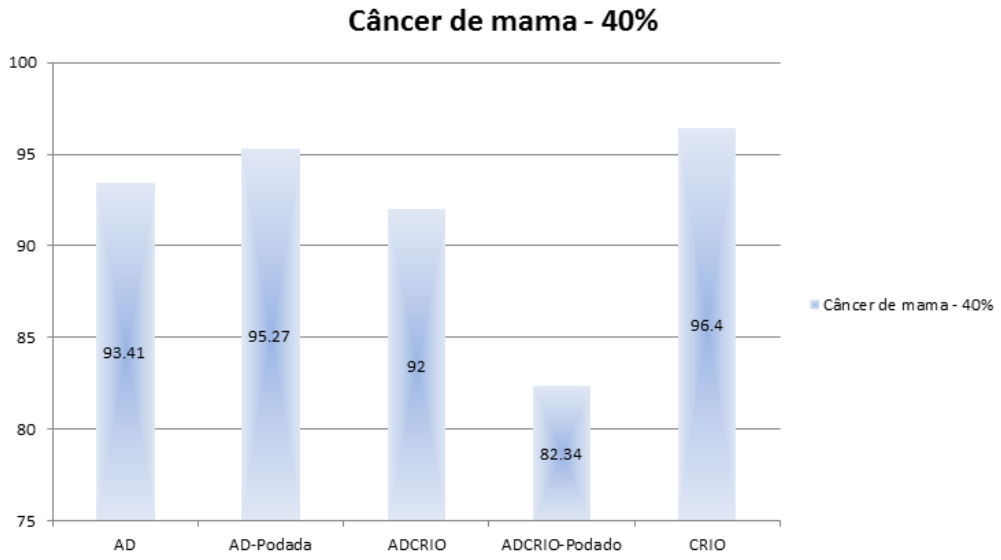

Figura 5.8. Comparação dos resultados para a base de Câncer de mama com 40% dos dados utilizados para criar o modelo. O algoritmo CRIO obteve os melhores resultados seguido pelos algoritmos AD com poda, AD sem poda, ADCRIO sem poda e ADCRIO com poda.

Para esta base de dados o algoritmo CRIO foi o melhor nos dois testes, seguido pelo AD com poda, AD sem poda, ADCRIO sem poda e ADCRIO com poda. A poda da árvore obteve efeito negativo para o ADCRIO nos dois testes realizados indicando que atributos importantes foram removidos da árvore de decisão dificultando os subproblemas que o CRIO resolve dentro do ADCRIO.

5.6.5 Comparações na base de dados Aprovação de crédito

O resultados dos testes da base de Aprovação de crédito com 60% dos dados utilizados para treinamento são apresentados na tabela 5.9. Nesta base de dados o algoritmo CRIO obteve o melhor resultado seguido pelos algoritmos ADCRIO sem poda, AD sem poda, AD com poda e por último o ADCRIO com poda. A diferença entre a solução do CRIO e do ADCRIO sem poda foi de 3.88%. Para o AD sem poda a diferença foi de 7.01%. Em seguida veio o AD com poda com diferença de 14.20% e por último o ADCRIO com poda com 17.92% de diferença. A poda das árvores do AD e ADCRIO piorou os resultados dos respectivos algoritmos. Este comportamento não era esperado para o algoritmo AD.

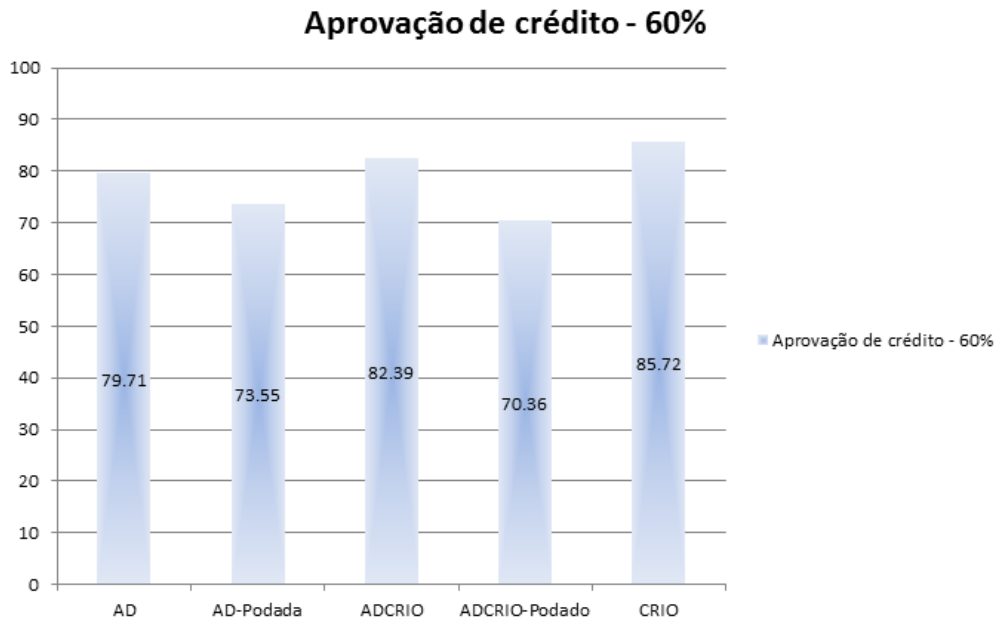

Figura 5.9. Comparação dos resultados para a base de Aprovação de crédito com 60% dos dados utilizados para criar o modelo. O algoritmo CRIO obteve os melhores resultados seguido pelos algoritmos ADCRIO sem poda, AD sem poda, AD com poda e ADCRIO com poda.

Para os testes com 40% da base utilizada como treinamento o algoritmo CRIO também apresentou os melhores resultados seguido pelo ADCRIO sem poda, AD sem poda, ADCRIO com poda e por último AD com poda.A diferença entre o CRIO e os demais foi de 1.38% para o ADCRIO sem poda, 3.24% para o AD sem poda, 6.19% para o ADCRIO com poda e 6.25% para o AD com poda. Novamente a poda da árvore piorou os resultados dos algoritmos AD e ADCRIO.

O CRIO foi o melhor algoritmo para esta base de dados. Em seguida o ADCRIO sem poda foi o melhor algoritmo. Mais testes são necessários para identificar porque a poda da árvore piorou o algoritmo AD, que normalmente é beneficiado com a poda da árvore.

5.6.6 Comparação geral

O CRIO foi o algoritmo que obteve os melhores resultados na maioria das bases de dados quando a base de treinamento continha 60% dos dados, dos 5 testes em 3 o CRIO foi o melhor algoritmo. Somente nas bases SPECTF e BUPA que o CRIO foi superado pelos demais algoritmos. Mesmo assim a maior diferença entre a solução encontrada pelo CRIO e a melhor solução foi de apenas 2.33%.

O ADCRIO obteve apenas um melhor resultados para a base de dados BUPA.

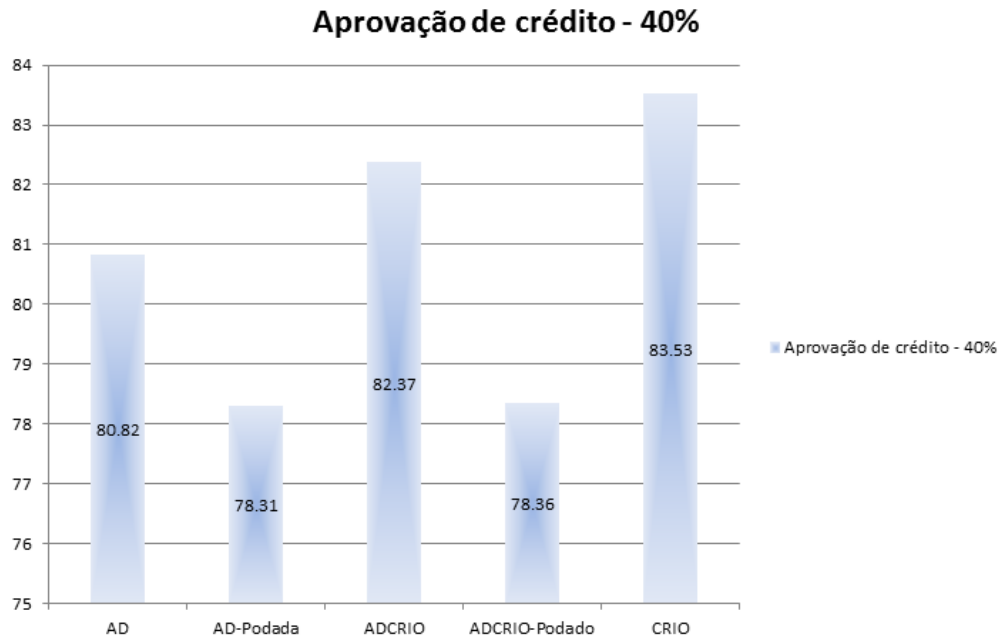

Figura 5.10. Comparação dos resultados para a base de Aprovação de crédito com 40% dos dados utilizados para criar o modelo. O algoritmo CRIO obteve os melhores resultados seguido pelos algoritmos ADCRIO sem poda, AD sem poda, ADCRIO com poda e AD com poda.

Nas demais bases de dados está técnica apresentou bons resultados quando não houve a poda da árvore, ficando no máximo a 6.59% da melhor solução. Quando a poda foi aplicada o pior caso ficou 26.10% abaixo da melhor solução, um péssimo resultado. Logo, a poda se mostrou prejudicial ao algoritmo ADCRIO, o que faz sentido uma vez que os subproblemas resolvidos são mais complexos pois possuem dados mais especializados e menos atributos para serem utilizados na geração do modelo.

As AD obtiveram um desempenho médio em todos os casos exceto o de Aprovação de crédito, no qual foi a pior técnica. Para a base de dados SPECTF a AD foi a melhor técnica. E somente na base de Aprovação de crédito a poda da árvore piorou o resultado da AD normal. Logo, a AD deve ser utilizada com a poda, exceto em raros casos como o da Aprovação de crédito.

Nos testes utilizando 40% da base de dados para treinamento os resultados foram um pouco diferentes.

O algoritmo CRIO foi o melhor em 2 bases de dados, Câncer de mama e Aprovação de crédito, porém para as bases de dados em que o CRIO não foi o melhor algoritmo a diferença entre a melhor solução e a solução do CRIO aumentou em relação ao modelo com 60% dos dados. Por exemplo, para a base de dados SPECTF a diferença entre a melhor solução e a solução do CRIO que era de 2.14% subiu para 9.39%. Como o

CRIO ainda foi a melhor técnica nas outras 3 bases de dados este aumento não indica que o CRIO só é capaz de trabalhar quando a base de treinamento é grande, porém sofre o impacto com o número reduzido de dados, assim como os demais algoritmos.

O ADCRIO obteve resultados melhores aos encontrados anteriormente. O ADCRIO foi o melhor em duas base de dados, SPECTF e BUPA. Para a base de dados BUPA a versão com poda foi a melhor e para a base de dados SPECTF a versão sem poda foi a melhor, sendo impossível indicar se a poda é ou não favorável ao ADCRIO.

As AD apresentaram resultados semelheantes. Na instância SPECT ela obteve o melhor resultado enquanto nas demais instâncias os resultados foram bastante semelhantes aos encontrados nos testes anteriores.

Pelos resultados encontrados não é possível indicar qual o melhor algoritmo para todos os problemas. Para os problemas testados neste trabalho o algoritmo CRIO obteve uma pequena vantagem em relação aos demais. Na comparação entre o ADCRIO e a AD o ADCRIO obteve melhores resultados porém foi mais instável em relação aos resultados e à poda da árvore. Mais testes são necessário para interpretar a necessidade ou não da poda da árvore para o ADCRIO.

5.7 Conclusão e trabalhos futuros

5.7.1 Conclusão

O algoritmo ADCRIO conseguiu demonstrar que é possível diminuir a dimensão do algoritmo CRIO sem afetar os resultados do CRIO de forma significativa. Somente nas base de dados de Câncer de mama e Aprovação de crédito que a versão do ADCRIO podado apresentou um resultado bastante inferior ao CRIO. Nas demais bases de dados o ADCRIO obteve resultados bem próximos ao CRIO e até superior em alguns casos. Além do benefício de reduzir a dimensão do problema temos também o benefício de facilidade para interpretar o resultado encontrado pelo ADCRIO que apresenta os resultados da mesma maneira que as AD. Estes são os principais benefícios do ADCRIO.

O tempo de execução do ADCRIO, apesar de não ter sido registrado neste trabalho, foi superior ao do CRIO. O ADCRIO executa vários problemas do CRIO, porém o ADCRIO consome menos memória uma vez que somente parte dos atributos é utilizada na construção do modelo. Testes em base de dados maiores são necessários para estabelecer um padrão de quando o ADCRIO é mais eficiente. Com bases de dados pequenas em que o CRIO pode ser executado o ADCRIO é um algoritmo inferior.

A poda da AD que em geral incrementa os resultados da AD não teve o mesmo efeito para o ADCRIO. Talvez algum outro método para escolher os atributos da AD e o processo de poda da AD possam melhorar a qualidade das soluções do ADCRIO. Em alguns casos a poda até melhorou o resultado no ADCRIO mas este comportamento não foi um padrão.

Os resultados dos testes foram inconclusivos para apontarmos qual a melhor técnica. O CRIO obteve vantagem na maioria dos testes. Porém em alguns testes a AD e o ADCRIO superaram o CRIO. O ADCRIO apresentou uma alternativa interessante para as AD, mas não obteve o desempenho esperado nos resultados para poder substituir as AD nem o CRIO devido a falta de regularidade nos resultados obtidos. Em alguns casos o ADCRIO obteve resultados expressivos e em outros obteve o pior resultados entre as técnicas comparadas.

No geral, pode-se concluir que o ADCRIO pode ser uma alternativa para o CRIO desde que outro método para escolher os atributos utilizados pelo ADCRIO seja aplicado. Este método pode ser uma AD com outro método de partição de dados diferente do ganho de informação ou até mesmo outra técnica que possa decidir quais são os melhores atributos para o ADCRIO ser executado.

5.7.2 Trabalhos futuros

A partir deste trabalho algumas possibilidades para trabalhos futuros podem ser investigadas. A primeira possibilidade é realizar um estudo com bases de dados maiores em quantidade de atributos e exemplares. Este estudo seria capaz de apontar se em bases de dados maiores a redução no número de atributos poderia manter a qualidade das soluções em tempo computacional menor.

No contexto das AD, alguns métodos para a escolha dos atributos também podem ser testados para comparar o desempenho do ADCRIO. Talvez uma AD que trabalhe escolhendo mais atributos a cada nível possa apresentar resultados melhores para o ADCRIO ou outro processo de poda possa incrementar os resultados do ADCRIO.

Em relação ao CRIO, outro trabalho que pode ser investigado é escolher um conjunto de atributos reduzidos para executar o CRIO. Diferente do ADCRIO que executa vários CRIO para cada subconjunto de atributos o CRIO pode ser testado sendo executado uma única vez com um conjunto reduzido de atributos. Assim como no ADCRIO é necessário que outro algoritmo faça a pré-seleção dos atributos que devem ser utilizados pelo CRIO.

Referências Bibliográficas

Bertsimas, D. & Shioda, R. (2007). Classification and regression via integer optimization. *Operations Research*, 55(2):252–271.

Boros, E.; Hammer, P. L.; Ibaraki, T.; Kogan, A.; Mayoraz, E. & Muchnik., I. (2000). An implementation of logical analysis of data. *IEEE Transactions on Knowledge and Data Engineering*, 12(2):292396.

Bradley, P. S.; Fayyad, U. M. & Mangasarian, O. L. (1999). Mathematical programming for data mining: Formulations and challenges. *INFORMS: Journal of Computing, special issue on Data Mining*, 11(3):217–238.

Breiman, L.; Friedman, J.; Stone, C. J. & Olshen, R. (1984). *Classification and Regression Trees*. Chapman & Hall.

Brodley, C. & Utgoff, P. (1995). Multivariate decision trees. *Machine Learning*, 19:45–77.

Cervantes, J.; Li, X.; Yu, W. & Li, K. (2008). Support vector machine classification for large data sets via minimum enclosing ball clustering. *Neurocomputing*, 71(4-6):611–619.

Chandra, B. & Varghese, P. P. (2009). Moving towards efficient decision tree construction. *Information Sciences*, 179:1059–1069.

Chen, J.; Wang, C. & Wang, R. (2009a). Adaptative binary tree for fast svm multiclass classification. *Neurocomputing*, 72:3370–3375.

Chen, Y.; Hsu, C. & Chou, S. (2003). Constructing a multi-valued and multi-labeled decision tree. *Expert Systems with Applications*, 25(2):199–209.

Chen, Y.-L.; Wu, C.-C. & Tang, K. (2009b). Building a cost-constrained decision tree with multiple condition attributes. *Inf. Sci.*, 179(7):967–979.

da F. Lisboa, J. M. M. R. (1994). *Indução de Árvores de Decisão, HistClass - Proposta de um algoritmo não paramétrico*. PhD thesis, Universidade Nova de Lisboa.

De Mántaras, R. L. (1991). A distance-based attribute selection measure for decision tree induction. *Machine Learning*, 6:81–92.

Fayyad, U. M. & Irani, K. B. (1992). On the handling in decision tree of continuous-valued attributes generation. *Machine Learning*, 8:87–102.

Fayyad, U. M. & Irani, K. B. (1993). Multi-interval discretization of continuos-valued attributes for classification learning. In Bajcsy, R., editor, *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pp. 1022–1027. Morgan-Kaufmann.

Fayyad, U. M.; Piatetsky-Shapiro, G. & Smyth, P. (1996). Knowledge discovery and data mining: Towards a unifying framework. In *In Proceedings of the 2nd ACMinternational conference on knowledge discovery and data mining (KDD)*, p. 8288.

Gama, J. & Brazdil, P. (1995). Characterization of classification algorithms. In *Proceedings of the 7th Portuguese Conference on Artificial Intelligence: Progress in Artificial Intelligence*, pp. 189–200, London, UK. Springer-Verlag.

Geigy (1978). Tables scientifiques. Technical report, CIBA-GEIGY SA.

Jenhani, I.; Amor, N. B. & Elouedi, Z. (2008). Decision trees as possibilistic classifiers. *Int. J. Approx. Reasoning*, 48(3):784–807.

Kianmehr, K. & Alhajj, R. (2008). Carsvm: A class association rule-based classification framework and its application to gene expression data. *Artif. Intell. Med.*, 44(1):7–25.

Kim, H. & yin Loh, W. (2001). Classification trees with unbiased multiway splits. *Journal of the American Statistical Association*, 96:589–604.

Kumar, M. A. & Gopal, M. (2010). A hybrid svm based decision tree. *Pattern Recognition*, 43:3977–3987.

Kunapuli, G.; Bennett, K. P.; J.Hu & J.Pang. (2008). Bilevel model selection for support vector machines. *CRM Proceedings and Lecture Notes*, 45:129–158.

Li, X.-B. (2005). A scalable decision tree system and its application in pattern recognition and intrusion detection. *Decision Support Systems*, 41:112–130.

Lim, T.-S.; Loh, W.-Y. & Shih, Y.-S. (2000). A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Mach. Learn.*, 40:203--228.

Lorena, A. C. Carvalho, A. C. P. (2003). Introdução aos classificadores de margens largas (large margen classifiers). Technical report, São Carlos - SP.

Luenberger, D. G. (2006). *Information Science*. Princeton University Press.

Lukasz A. Kurgan, Krzysztof J. Cios, R. T. M. O. & Goodenday, L. S. (2001). Knowledge discovery approach to automated cardiac spect diagnosis. *Artificial Intelligence in Medicine*, 23:149–169.

Luts, J.; Ojeda, F.; de Plas, R. V.; de Moor, B.; Huffel, S. V. & Suykens, J. A. K. (2010). A tutorial on support vector machine-based methods for classification problems in chemometrics. *Analytica Chimica Acta*, 665:129–145.

Maimon, O. & Rokach, L., editores (2005). *The Data Mining and Knowledge Discovery Handbook*. Springer.

Mangasarian, O. L. & Wolberg, W. H. (1990). Cancer diagnosis via linear programming. *SIAM News*, 23(5):1–18.

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill Science.

Murthy, P. & Pazzani, M. J. (1994). Exploring the decision forest: An empirical investigation of occam's razor in decision tree induction. *Journal of Artificial Intelligence Research*, 1:257–275.

Murthy, S. K. (1998). Automatic construction of decision trees from data: A multidisciplinary survey. *Data Mining and Knowledge Discovery*, 2:345–389.

Osuna, E.; Freund, R. & Girosi, F. (1997). Training support vector machines: An application to face detection. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Pinheiro, C. A. R. (2008). *Inteligência Analítica: Mineração de Dados e Descoberta de Conhecimento*. Ciência Moderna.

Polat, K. & Güneş, S. (2009). A novel hybrid intelligent method based on c4.5 decision tree classifier and one-against-all approach for multi-class classification problems. *Expert Syst. Appl.*, 36:1587–1592.

Quinlan, J. (1987). Simplifying decision trees. *International Journal of Man-Machine Studies*, 27:221–234.

Quinlan, J. R. (1986a). Induction of decision trees. *Machine Learning*, 1(1):81–106.

Quinlan, J. R. (1986b). *Machine Learning: An artificial intelligence Approach (Vol. 2)*, chapter The effect of noise on concept learning, pp. 149–166. Morgan Kaufmann.

Quinlan, J. R. (1993). *C4.5 : Programs for Machine Learning*. Morgan Kaufmann.

Raileanu, L. E. & Stoffel, K. (2004). Theoretical comparison between the gini index and information gain criteria. *Annals of Mathematics and Artificial Intelligence*, 41:77–94.

Rokach, L. & Maimon, O. (2008). *Data Mining With Decision Trees, Theory and Applications*, volume 69 of *Series in Machine Perception and Artificial Intelligence*. World Scientific Publishing Co. Pte. Ltd.

Swain, P. & Hauska, H. (1977). The decision tree classifier design and potential. *IEEE Trans. on Geoscience and Electronics*, GE-15:142–147.

Vapnik, V. (1995). *The Nature of Statistical Learning Theory*. Springer, New York.

Vapnik, V. N. (1998). *Statistical Learning Theory*. Wiley-Interscience.

Witten, I. H.; Frank, E.; Trigg, L.; Hall, M.; Holmes, G. & Cunningham, S. J. (1999). *Weka: Practical Machine Learning Tools and Techniques with Java Implementations*. ACM, New York, NY, USA.

Zhou, Z.-H. & Chen, Z.-Q. (2002). Hybrid decision tree. *Knowledge-Based Systems*, 15:515–528.