CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA

ADILSON PEREIRA DOS SANTOS

SÃO PAULO
MAIO/2008

ADILSON PEREIRA DOS SANTOS

USO DE ÁRVORES DE DECISÃO NA PREDIÇÃO DE FALHAS EM MOTORES ELÉTRICOS

UMA ABORDAGEM AUXILIADA POR UM SISTEMA DE BANCO DE DADOS

Dissertação apresentada como exigência parcial para obtenção do Título de Mestre em Tecnologia no Centro Estadual de Educação Tecnológica Paula Souza, no Programa de Mestrado em Tecnologia: Gestão, Desenvolvimento e Formação, sob orientação do Prof. Dr. Maurício Amaral de Almeida.

SÃO PAULO
Maio/2008

Santos, Adilson Pereira dos

S237u Uso de árvores de decisão na predição de falhas em motores elétricos: uma abordagem auxiliada por um sistema de banco de dados. -- São Paulo: CEETEPS, 2008.

119 f.

Dissertação (Mestrado) - Centro Estadual de Educação Tecnológica Paula Souza, 2008.

1. Tecnologia da Informação. 2. Árvores de decisão. 3. Manutenção industrial . I. Título.

CDU 681.3:007:658.58

ADILSON PEREIRA DOS SANTOS

USO DE ÁRVORES DE DECISÃO NA PREDIÇÃO DE FALHAS EM MOTORES ELÉTRICOS

UMA ABORDAGEM AUXILIADA POR UM SISTEMA DE BANCO DE DADOS

PROF. DR. MAURÍCIO AMARAL DE ALMEIDA

PROF ª DR ª MÁRCIA ITO

PROF. DR. REGIS ROSSI ALVES FARIA

São Paulo, 12 de maio de 2008.

DEDICATÓRIA

Aos meus pais, Pedro e Amélia, pelo incentivo e apoio que sempre deram durante todo o caminho até a realização deste trabalho, e cujo exemplo de vida tenho humildemente tentado seguir.

À minha esposa, Margareth, pela paciência e carinho, e também pelo incentivo nos momentos difíceis.

AGRADECIMENTOS

À Cia. do Metropolitano de São Paulo, em especial ao Eng. Jair Cândido Ferreira, ao então assessor da Gerência de Manutenção, Jorge Martins Secall, ao coordenador José Kenshiti Tuguimoto e ao Eng. Luiz Eduardo Argenton por viabilizarem a minha participação neste programa de mestrado.

Aos demais colegas e amigos da Cia. do Metropolitano de São Paulo pelas palavras de incentivo e pelas informações fornecidas.

Ao meu orientador, Prof. Dr. Maurício Amaral de Almeida, pelo seu apoio e orientação.

Aos membros da banca pelos seus comentários, críticas e sugestões, muito valiosas para a finalização deste trabalho.

A todos os colaboradores do programa de mestrado do Centro Paula Souza, e em especial à Cleonice pelo carinho, pela atenção e por sempre me lembrar das datas e prazos.

Aos professores e colegas do programa de mestrado pela oportunidade da troca de experiências e de conhecimento.

Um especial agradecimento ao colega Fábio A. Almeida pelas dicas sobre a linguagem Java, que muito me ajudaram da implementação do algoritmo de indução de árvores de decisão.

E, enfim a todos aqueles que acreditaram e acreditam em mim e que de alguma forma, direta ou indiretamente, contribuíram para a conclusão deste trabalho.

“Sábio é aquele que conhece os limites da própria ignorância.”

Sócrates

RESUMO

SANTOS, ADILSON P. **Uso de Árvores de decisão na Predição de Falhas em Motores Elétricos**: uma abordagem auxiliada por um sistema de banco de dados. 2008. 119 f. Dissertação (Mestrado em Tecnologia) – Centro Estadual de Educação Tecnológica Paula Souza, São Paulo, 2008.

Dispor de técnicas para identificar quais os fatores predisponentes para a ocorrência de falhas em motores elétricos de tração de sistemas metro-ferroviários e os seus inter-relacionamentos pode ajudar a aperfeiçoar os processos de manutenção e reduzir o número de falhas, aumentando a confiabilidade destes equipamentos. Neste tipo de equipamento, onde as intervenções realizadas podem ser bastante invasivas, os processos de manutenção podem ser fatores geradores de novas falhas. Possuir um bom histórico de manutenção juntamente com técnicas adequadas de mineração de dados pode indicar os pontos a serem melhorados nas atividades de manutenção realizadas. Neste trabalho é aplicada a técnica de indução de árvores de decisão sobre os dados históricos da manutenção em oficinas para descobrir a relação entres os processos de manutenção e a reincidência de falhas. São discutidos os aspectos teóricos envolvidos na tarefa de indução das árvores de decisão e o estado da arte sobre a integração de ferramentas de mineração de dados e sistemas de banco de dados. São abordadas também, as atividades realizadas na fase de pré-processamento dos dados e as soluções utilizadas para o tratamento dos atributos cujos valores são desconhecidos e a discretização dos atributos contínuos, usando uma técnica baseada na análise de histogramas. Na abordagem proposta, as informações necessárias para o processo de indução da árvore de decisão serão obtidas diretamente do banco de dados, com o uso de instruções SQL. Além dos dados do histórico de manutenção, os parâmetros de configuração do programa de indução e a árvore gerada pelo programa também são armazenados neste banco de dados. Por fim discute-se a validade dos resultados obtidos com a implementação desta abordagem.

ABSTRACT

SANTOS, ADILSON P. **Uso de Árvores de decisão na Predição de Falhas em Motores Elétricos**: uma abordagem auxiliada por um sistema de banco de dados. 2008. 119 f. Dissertação (Mestrado em Tecnologia) – Centro Estadual de Educação Tecnológica Paula Souza, São Paulo, 2008.

Identifying predisponent factors in electric motors fails and it is interconnections, on rail traction-systems, can help to improve maintenance process and to decrease fault occurrences, increasing reliability. In such equipments, servicing can demand deeper repairs actions and maintenance process may become a fault source. Ensuring a reliable maintenance historical data and handling proper data mining tools could point out improving opportunities in maintenance activities. In this work, induction decision trees techniques is applied on historical maintenance data to find out the connection between maintenance process and fault relapsing. Theorical arguments involved on induction decision trees tasks and state of art on integrating data mining tools and data base systems are discussed. Also, the pre processing activities and the solutions in use to handling attributes with missing values and the discretization process based on histogram analysis are treated. In this approach, the induction decision trees program require informations that are acquired from the data base system by using SQL statements. Besides historical maintenance data, the induction program configurations parameters and the decision tree generated are also stored on this data base. At last, checking produced results and implementation validity of this approach is preceded.

Lista de Abreviaturas e Siglas

API – Application Programming Interface

cv – cavalo-vapor

IDE – Integrated Development Environment

JDBC - Java Database Connectivity

MRDTL - Multirelational Decision Tree Learning Algorithm

KDD – Knowledge Discovery in Databases

Mohms – O equivalente a milhões de Ohms

Ohms – Unidade de medida da resistência elétrica

PC – Personal Computer

PL/pgSQL – Procedural Language/PostgreSQL

PL/SQL – Procedural Language/SQL

SGBD – Sistema Gerenciador de Bancos de Dados

SQL - Structured Query Language

V – Volts

Índice de Figuras

Lista de tabelas

Lista de Quadros

SUMÁRIO

1. Introdução

Muitos dos dados dos processos industriais ligados à manufatura são hoje armazenados em SGBDs – Sistemas Gerenciadores de Bancos de Dados. Desde o recebimento do pedido ou da ordem de produção até a entrega do produto final para o distribuidor ou cliente, todos os dados podem ser armazenados em um banco de dados.

Assim como na área de industrial, os sistemas de informação ligados à manutenção de equipamentos e sistemas estão se tornando cada vez mais importantes. O advento de normas como a ISO9000 tem exigido o registro e o controle de uma série de parâmetros presentes nos processos das empresas. Isto veio apenas reforçar uma tendência surgida muito tempo antes, quando se buscou a qualidade dos processos produtivos e dos produtos como um diferencial competitivo, visando o aumento da produtividade ou apenas a manutenção da sua posição no mercado.

Vê-se, portanto, que a manutenção de equipamentos e sistemas tem assumido uma função estratégica nas empresas. Dispor, hoje, de ferramentas informatizadas de apoio à manutenção pode significar a economia de recursos não apenas financeiros, mas também ambientais e humanos, já que uma falha de equipamento (MOUBRAY, 2000), pode afetar o meio ambiente e a segurança das pessoas.

Existem basicamente quatro tipos de manutenção segundo Pinto; Xavier (2001):

1. Manutenção corretiva:
   - Caracteriza-se pela atuação em um equipamento ou sistema para correção da falha ou do desempenho de funcionamento menor que o esperado;
2. Manutenção preventiva:
   - Caracteriza-se pela atuação em um equipamento ou sistema de forma a reduzir ou evitar a falha ou queda de desempenho. Procura prevenir a ocorrência de falhas através de um plano de manutenção previamente definido;
3. Manutenção preditiva:
   - Também conhecida por Manutenção Sob Condição ou Manutenção com Base no Estado do Equipamento, é baseada no acompanhamento sistemático de parâmetros de condição e desempenho para permitir a operação contínua do equipamento ou sistema pelo maior tempo possível. Uma vez detectados desvios nos aspectos monitorados um plano de ação é disparado para restabelecer as condições de funcionamento anteriores;

4. Manutenção detectiva:
    - Procura detectar falhas ocultas e não perceptíveis ao pessoal de operação e manutenção. Exemplos simples deste tipo de manutenção são os testes de sistemas de iluminação de emergência e de sistemas de detecção de incêndio.

Dentro do contexto da manutenção, a confiabilidade é uma característica muito importante em equipamentos e sistemas industriais e comerciais, sendo primordial a identificação de falhas ocultas para garantir a confiabilidade (PINTO, XAVIER, 2001).

Determinar os aspectos que comprometem a confiabilidade é essencial para:

- Evitar ou minimizar transtornos operacionais;
- Minimizar prejuízos por paradas não programadas em linhas de produção;
- Minimizar os transtornos causados pela não prestação de serviços ao público;
- Aumentar o grau de confiança dos consumidores ou usuário na marca;
- Reduzir custos operacionais pelo aprimoramento dos processos produtivos e de manutenção.

Estes são apenas alguns dos benefícios esperados, mas muitos outros poderiam ser listados.

A confiabilidade pode ser definida como: "a probabilidade que um item possa desempenhar sua função requerida, por um intervalo de tempo estabelecido, sob condições definidas de uso" (sic) (PINTO, XAVIER, 2001, p. 96). Em outras palavras, quanto maior a probabilidade de um item permanecer operando dentro de seus padrões normais de funcionamento maior será o seu nível de confiabilidade.

Matematicamente, a confiabilidade de um equipamento pode ser expressa como (PINTO, XAVIER, 2001) (1):

$$R(t) = e^{-\lambda t} \qquad (1)$$

Onde:

R(t) = confiabilidade a qualquer tempo t

e = base dos logaritmos naturais ou neperianos (e = 2,178...)

λ = taxa de falhas (número total de falhas por período de operação)

t = tempo previsto de operação.

Especificamente, na área de manutenção de equipamentos industriais, a confiabilidade é de vital importância para evitar prejuízos financeiros ou à imagem da empresa, gerados pela

interrupção da produção ou da paralisação na prestação de serviços.

Para aumentar os níveis de confiabilidade e aprimorar os processos de manutenção ou de produção, é possível a utilização os dados históricos existentes para:

- a determinação das causas das falhas ou suas condições predisponentes;
- a predição de quando este tipo de evento poderá ocorrer;
- estabelecer a inter-relação entre os parâmetros funcionais.

Dentre os dados históricos existentes pode-se citar:

- os parâmetros de funcionamento do equipamento;
- as intervenções ou manutenções realizadas;
- as condições ambientais a que os equipamentos estão expostos;
- o tempo de funcionamento do equipamento entre duas falhas consecutivas;
- os processos de manutenção aplicados.

Mas, para utilizar os dados históricos, é essencial que eles estejam registrados, preferencialmente, em algum meio magnético ou ótico para que sejam facilmente acessados por um computador.

Como muitas empresas têm buscado a obtenção e a manutenção das certificações da série ISO 9000, a obtenção de muitos dos dados necessários acaba sendo facilitada, já que uma das inúmeras exigências destas normas é o registro dos parâmetros de controle dos processos.

Enquanto a determinação dos fatores predisponentes para a ocorrência de falhas, pode ajudar na eliminação ou minimização da quantidade de problemas de funcionamento, a predição de falhas, pode evitar ou minimizar transtornos operacionais. O conhecimento destes fatores predisponentes, quando utilizada em conjunto com a manutenção preditiva, permite a atuação de forma antecipada ou programada nos equipamentos antes que eles apresentem problemas.

O desafio então é implementar uma solução confiável e precisa sem que especialistas humanos precisem manipular milhares de registros para obter padrões de comportamento. Uma das alternativas para a realização desta tarefa é o uso de técnicas de mineração de dados para obter os padrões de comportamento, que, provavelmente, existem nestes dados.

Dentre as técnicas existentes, a indução de árvores de decisão possui a grande vantagem de apresentar os seus resultados em um formato que é facilmente interpretado pelos especialistas humanos. Esta forma de representação, na forma de diagramas causa/efeito ou *check list*, já e conhecida dos técnicos e engenheiros ligados à área de manutenção, por constar nos manuais de manutenção de alguns equipamentos para auxiliar no diagnóstico de

falhas. Além disto, quando comparada a outras técnicas, os seus princípios são de simples entendimento e a sua implementação, relativamente simples, permite a adição de uma série de refinamentos para melhorar sua precisão e legibilidade.

Segundo Russel (2004) e Mitchell (1997) a indução de árvores de decisão é uma das formas mais usadas, estudadas e mais bem sucedidas, dentre os algoritmos de aprendizagem, seja pela sua praticidade, seja pela sua simplicidade. Além disto, são robustas a ruídos nos dados (MITCHELL, 1997). raha; Shmilovici (2003), por exemplo, ao optarem pelo uso de árvores de decisão para melhorar o processo litográfico, levaram em conta que os resultados são apresentados de forma compreensível para os usuários, já que podem ser representados por conjuntos de regras do tipo se-então (*if-then*).

Os algoritmos de geração de árvores de decisão de código aberto - *open source* – mais populares, como por exemplo o CART (BREIMAN, FRIEDMAN, 1984), o ID3 (QUINLAN, 1986), o C4.5 (QUINLAN, 1993), o Weka (WITTEN, FRANK, 2005) e o Tanagra (RAKOTOMALALA, 2005), foram concebidos para acessar os dados na forma de um sistema de arquivos pré-definido, geralmente do tipo texto. Isto funciona muito bem para acessar um volume pequeno de dados, mas para o tratamento de milhares, centenas de milhares, ou mesmo milhões de registros, esta estratégia pode não ser muito eficiente.

Nos sistemas baseados em texto, os dados a serem analisados devem estar na memória principal do sistema, mas isto, apesar do acesso extremamente rápido, acaba limitando a quantidade de registros analisados ao espaço de memória disponível, forçando a criação de estratégias ou de artifícios como o *windowing* (QUINLAN, 1993), diminuindo o desempenho do sistema (SATTLER, DUNEMANN, 2001).

Contudo, freqüentemente os dados estão armazenados em sistemas de banco de dados que dispõem das seguintes características:

- possuem mecanismos altamente eficientes para o armazenamento, o acesso e a extração dos dados;
- são capazes de manipular enormes quantidades de dados;
- possuem mecanismos de *backup* automático;
- segurança no acesso aos dados, dentre outras.

Além disto existem outras vantagens (BLOCKEEL et al., 2007):

- flexibilidade para consultas *ad-hoc*;
- os dados são analisados onde eles estão;
- possuem uma série de recursos que aumentam a escalabilidade e a flexibilidade;

- existe uma separação clara entre a camada física e a camada conceitual.

Blockeel et al. (2007) e Sattler; Dunenmann (2001), apontam que a maior desvantagem do uso dos SGBD em atividades de mineração de dados (item 3.1) está na baixa performance da linguagem padrão de acesso aos dados, a linguagem SQL – *Structured Query Language*. Com o uso desta linguagem, as operações de mineração de dados devem ser implementadas como uma série de consultas que são tratadas de forma "isolada e independente do SGBD". No entanto, Blockeel et al. (2007), afirmam que este recurso não deve ser descartado.

Qualquer que seja o caso, é essencial a existência de um histórico detalhado das condições a serem analisadas e testadas, para que os resultados tenham validade e representem um grande espectro de possibilidades e de variações nos parâmetros.

As oficinas de manutenção do Metrô de São Paulo são responsáveis pela manutenção dos equipamentos presentes nas quatro linhas que atendem a cidade de São Paulo. São dez oficinas de manutenção, distribuídas em três pátios de manutenção, que tem sob sua responsabilidade a manutenção dos equipamentos mecânicos, elétricos, eletrônicos e dos veículos usados pelas equipes de manutenção, totalizando mais de oito mil itens diferentes.

Do total de equipamentos para manutenção boa parte deles tem registrados algum tipo de histórico de manutenção, cujo nível de detalhamento depende da sua complexidade construtiva e estrutural, da complexidade dos processos de manutenção envolvidos e da sua importância dentro dos sistemas do qual fazem parte.

Em 1994, as oficinas de manutenção iniciaram um processo de informatização para aumentar o controle dos seus processos internos e garantir o registro histórico das atividades realizadas. Este primeiro processo de informatização se iniciou com o desenvolvimento interno de um sistema integrado de gestão. Este sistema passa atualmente por um processo de reformulação visando a sua atualização tecnológica e uma integração efetiva com os demais sistemas ligados à manutenção.

Dentro do contexto exposto, os equipamentos que apresentam um histórico de manutenção maior e mais bem detalhado são os motores elétricos de tração utilizados nos trens de forma semelhante ao motor de um carro. Este registro histórico, anterior ao processo de informatização, já foi transferido para o sistema informatizado para facilitar as consultas pelos gestores.

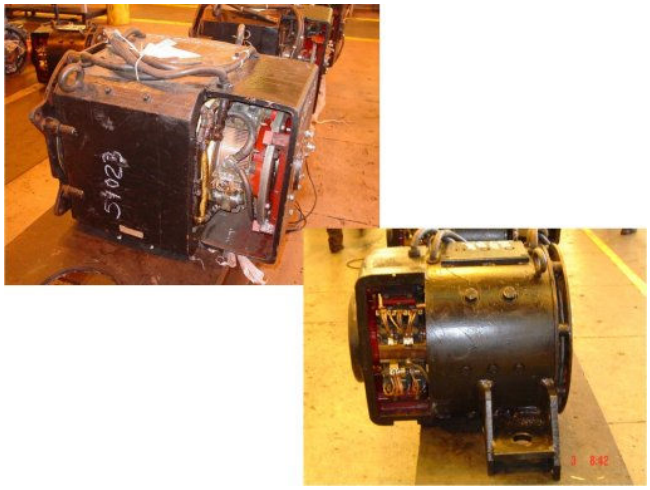


Figura 1 - Dois modelos dos motores de tração utilizados nos trens.

Os motores elétricos de tração utilizados nos trens (Figura 1), pesam aproximadamente 680kg e, dependendo do modelo do motor, funcionam alimentados por uma tensão nominal de 300 a 375 Volts em corrente contínua e geram de 170 a 190cv de potência nominal. Cada um deles funciona acoplado a uma caixa de redução responsável pela transferência do movimento gerado, para um conjunto eixo-rodas (rodeiro), que por sua vez está montado em uma estrutura metálica conhecida como truque (Figura 2). Cada carro (vagão) está apoiado sobre dois truques, com dois rodeiros cada um, sendo que cada rodeiro possui um motor de tração próprio. Portanto, em cada trem, compostos por 6 carros, existem vinte e quatro motores, quatro por carro, todos funcionando sincronizados sob o comando de equipamentos elétricos e circuitos eletrônicos.

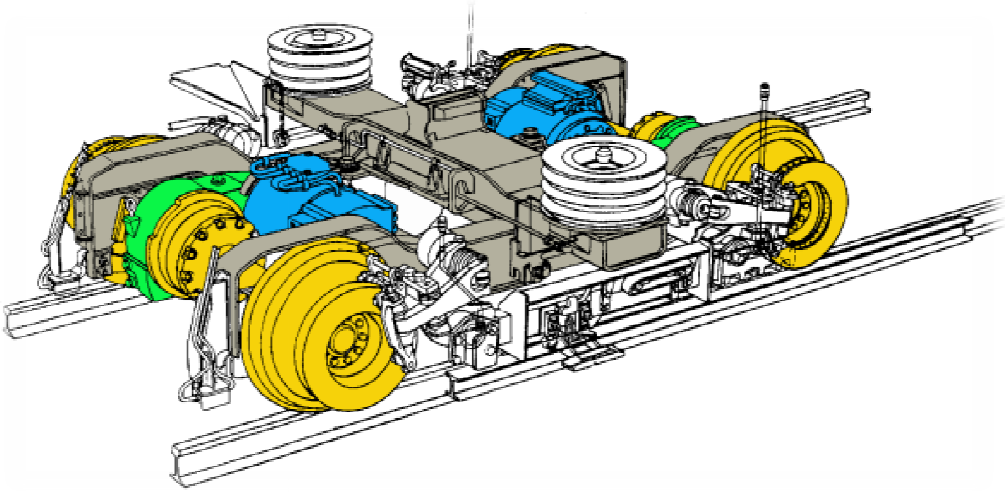


Figura 2 - Desenho do truque (cinza) com os seus diversos componentes. Nele podem ser identificados os motores de tração (azul), os conjuntos eixo-rodas (amarelo) e as caixas de redução (verde).

Figura 3 - Foto do truque durante processo de manutenção. Os motores de tração podem ser identificados com o auxílio da Figura 2.

Estes motores são removidos apenas em duas circunstâncias: para revisões programadas em oficina ou quando apresentam algum tipo de falha ou alteração nos seus parâmetros de funcionamento, caracterizando uma ação corretiva.

Falhas nestes equipamentos causam uma série de transtornos operacionais, como, por exemplo, o cancelamento de viagens programadas, provocadas pela retirada de trens de circulação. Como conseqüência adicional, este tipo de evento pode provocar a queda da qualidade na prestação do serviço à população devido à diminuição momentânea da disponibilidade de trens na via, enquanto este não é substituído por um trem reserva. Nos casos mais graves, a circulação dos demais trens pode ser prejudicada ou mesmo interrompida quando há o travamento das rodas do trem, por falha do motor.

Portanto, assegurar altos níveis de confiabilidade, não só dos motores de tração, mas também dos demais equipamentos e sistemas são de vital importância para garantir uma alta disponibilidade operacional. Mas, para que isto seja possível é necessário identificar claramente quais os aspectos materiais, humanos e metodológicos que interferem na confiabilidade.

2. Objetivo

Dada a importância de se minimizar os transtornos operacionais advindos de falhas operacionais nos equipamentos do Metrô de São Paulo, este trabalho tem como principal objetivo usar árvores de decisão como uma ferramenta para avaliar quais são os fatores predisponentes para a ocorrência de falhas em equipamentos, mais precisamente em motores elétricos de tração. Procura-se também, como parte do objetivo principal, avaliar com que grau de precisão a árvore de decisão gerada é capaz de predizer a ocorrência de uma nova falha, após o motor passar por um processo de manutenção em oficina.

Como objetivo secundário, pretende-se implementar um processo de geração de árvores de decisão que acesse diretamente o banco de dados onde os dados estão armazenados. Além disto, ele será utilizado também como um recurso adicional pelo algoritmo de indução para:

- levantar os dados necessários para a seleção dos atributos a serem testados pelo algoritmo;
- particionar o conjunto de dados durante a fase de crescimento da árvore;
- armazenar os metadados, ou seja, as informações sobre os dados;
- armazenar o resultado do processo de indução: a árvore de decisão.

3. Referencial Teórico

Antes de expor os aspectos teóricos envolvidos para atingir os objetivos propostos no capítulo 2, é conveniente apresentar algumas definições e conceitos que muitas vezes causam algum tipo de confusão, apesar de alguns já terem sido citados anteriormente.

Começa-se pelas definições de dado, informação e conhecimento.

Setzer (2005), descreve o dado como sendo "uma representação simbólica (isto é, feita por meio de símbolos), quantificada e qualificável", algo tangível, meramente sintático, simbólico.

Informação, por outro lado, pode ser interpretada como o dado adicionado de seu significado (ou semântica), de forma que o receptor da comunicação possa compreender o contexto onde este dado está inserido, o que ele representa e a que conceitos estão associados. Outra forma de definir informação poderia ser a interpretação de um dado dentro de um contexto. É importante mencionar que esta interpretação ou contextualização é muitas vezes cercada de subjetividade.

Segundo a definição clássica, o conhecimento consiste de crença verdadeira e justificada, sendo o conceito de crença verdadeira fruto da intersecção entre as crenças e a verdade dos fatos. Apesar, de tanto o conhecimento como a informação consistirem de declarações verdadeiras, o conhecimento distingue-se da mera informação porque está associado a um propósito ou a uma utilidade.

A seguir são descritos outros conceitos e definições:

Indutor: programa de aprendizado ou algoritmo de indução responsável pela extração do classificador a partir de um conjunto de exemplos;

Classificador: é o resultado do processo de indução. Deve ser capaz de, uma vez dado um novo exemplo cujo atributo alvo seja desconhecido, prever o valor deste atributo com base nos atributos previsores. Representa um conjunto de regras;

Atributo previsor: descreve uma característica ou um aspecto de um exemplo ou caso. Pode possuir valores discretos ou contínuos. Cada valor possível do atributo representa um galho na árvore de decisão;

Atributo alvo: também chamado de classe, representa o valor de saída de um exemplo ou caso e está relacionado a um ou mais atributos previsores. Na árvore de decisão é representado pela figura da folha;

Caso ou **exemplo**: é uma tupla de valores dos atributos previsores mais o atributo alvo. Representa o registro de uma tabela.

Conjunto de treinamento: conjunto de tuplas de valores usado pelo indutor para a geração do classificador;

Conjunto de teste: conjunto de tuplas de valores usado para validar o resultado gerado pelo classificador;

Folha: representa uma classe objetivo ou um valor possível do atributo alvo;

Nó de Decisão: representa o resultado do teste. Existe um galho para cada um dos possíveis resultados do teste feito no nó.

3.1 Mineração de dados

Com base nas definições, expostas no item anterior, de dado, informação e conhecimento, pode-se interpretar o significado do KDD - *Knowledge Discovery in Database* - como um processo pelo qual se procura extrair conhecimento de bancos de dados que tenham significado e aplicações práticas.

A mineração de dados ou *data mining* é um dos subprocessos deste processo denominado KDD; ele apresenta um escopo mais amplo, que busca o conhecimento embutido em grandes bases de dados, a partir da análise, da assimilação e da interpretação dos padrões encontrados no processo de *data mining*. Independentemente de estarem ou não associados a um processo de KDD, as técnicas de *data mining* podem ser usadas como apoio para sistemas de suporte à decisão.

Segundo Fayyad, Piatetsky-Shapiro e Smyth (1996), *apud* Rezende (2005), a mineração de dados ou a extração do conhecimento da base de dados é o processo de identificação de padrões válidos, novos, potencialmente úteis e compreensíveis embutidos nos dados.

O processo de mineração de dados pode ser resumido em 3 etapas: pré-processamento, extração de padrões e pós-processamento.

A etapa de pré-processamento compreende a escolha do conjunto de dados visando:

- realizar a extração de padrões;
- conhecer quais valores são válidos para os atributos, seus tipos e formatação;
- determinar quais são os atributos mais relevantes,
- determinar as relações entre os atributos quando estes estiverem contidos em mais de uma tabela;
- realizar a “limpeza” dos dados, removendo registros incompletos ou corrompidos;

- reduzir o número de dados, caso não haja espaço disponível suficiente ou o algoritmo de mineração utilizado apresente alguma limitação.Neste caso deve-se tomar o cuidado de não reduzi-los excessivamente, a ponto de comprometer a precisão e a qualidade do resultado.

Nesta fase, também pode ser feita a discretização - substituição de atributos contínuos ou com uma grande quantidade de valores distintos por outros - sem perda significativa da informação necessária para o processo de mineração de dados e interpretação posterior dos resultados. Este processo é discutido em detalhes no item 3.3.2.

A próxima etapa, a extração dos padrões, caracteriza-se pela extração do conhecimento da base de dados e pela escolha das atividades auxiliares para isto. Entre elas está a escolha da tarefa de mineração de dados e do algoritmo, ou dos algoritmos, a serem utilizados (REZENDE, 2005).

A Figura 4 ilustra as tarefas de mineração de dados classificados em dois grupos: atividades preditivas e atividades descritivas.

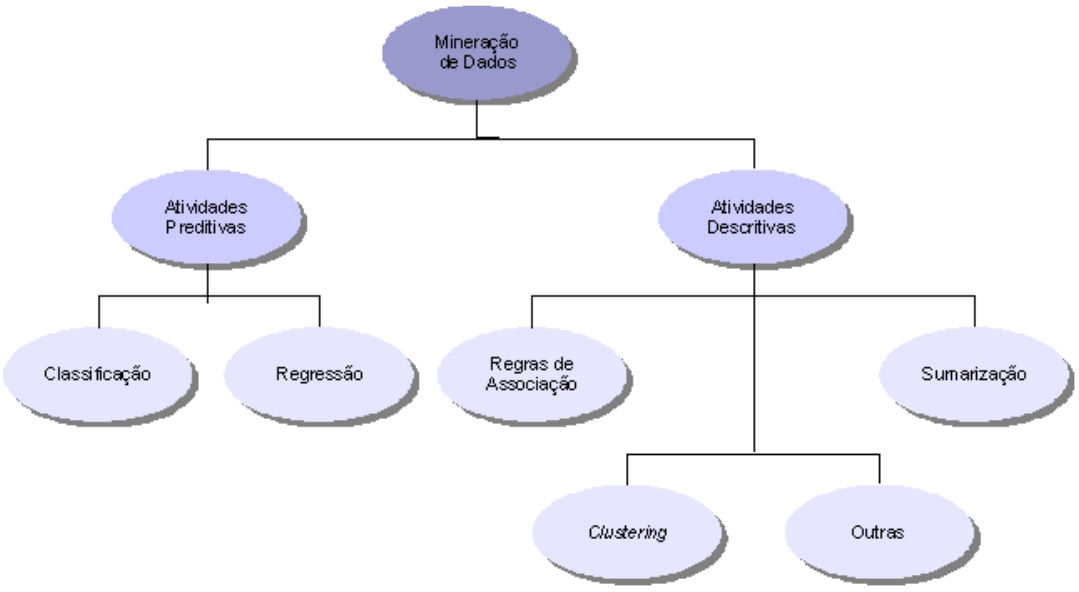


Figura 4 - Tarefas de mineração de dados (REZENDE, 2005).

As atividades de predição procuram, a partir de exemplos ou de um conjunto de dados de aprendizado, estabelecer correlação entre estes dados e as respostas conhecidas para predizer ou determinar quais as respostas para um novo conjunto de dados onde elas não são conhecidas.

As atividades de descrição, por sua vez, não procuram respostas, mas sim comportamentos embutidos nestes dados.

Uma vez definida a tarefa de mineração a ser realizada deve-se escolher os algoritmos ou técnicas mais adequadas aos objetivos almejados e cujos resultados sejam mais facilmente interpretados pelos especialistas e analistas de negócio, se eles forem os usuários finais.

Um estudo mais aprofundado das principais técnicas de mineração de dados, representadas na Figura 5, mostra que elas são fruto de fundamentações e abordagens diferentes. No entanto, isto que não significa que o emprego de uma técnica exclua a utilização de outra como complementar ou auxiliar. Isto é facilmente justificável já que nos casos encontrados no mundo real inexistem situações que possam ser classificadas ou analisadas inequivocamente.

Figura 5 - Técnicas usadas em mineração de dados. Adaptado de (REZENDE, 2005).

Uma vez realizada a extração de padrões, a etapa de pós-processamento deverá validar os resultados obtidos com o auxílio de um especialista no domínio analisado. Esta validação visa averiguar se os padrões extraídos realmente representam o conhecimento buscado e se os padrões são úteis e compreensíveis para o usuário, bem como se são precisos e se tem um desempenho compatível com a tarefa e o volume de dados.

Com as facilidades para a coleta de dados, a redução constante dos custos do seu armazenamento e a disseminação crescente do uso de computadores em todos os níveis de utilização, os sistemas de mineração devem:

- ser aperfeiçoados para garantir escalabilidade;
- apresentar os resultados de forma compreensível para os usuários;
- ser interativos;
- poder interagir e se integrarem a outros sistemas;
- estar preparados para novas tecnologias de armazenamento e de distribuição de informação;
- conseguir manipular outros tipos de dados, como imagens e sons.

3.2 Banco de dados

Atualmente, o modelo de banco de dados relacional é o mais difundido e o mais implementado, seja através de soluções proprietárias, seja *open source* ou *freeware*.

Mas, apesar das diferenças que possam existir quanto ao desempenho, a escalabilidade e a forma como foram implementados, este modelo de banco de dados, têm em comum a organização dos dados em tabelas normalizadas e relacionadas entre si.

Aliado a isto têm-se a existência de uma linguagem de manipulação padronizada e suportada pela quase totalidade dos sistemas gerenciadores de bancos de dados - a linguagem SQL. Ela permite que o acesso aos dados seja feito de forma simples e transparente para os usuários e sistemas que interagem com os SGBD.

Ao longo dos anos, os SGBDs têm passado por contínuos aperfeiçoamentos e inclusão de novas tecnologias. Dentre eles pode-se citar:

- o uso de variadas técnicas de indexação permite que os dados sejam acessados e alterados de forma rápida e transparente para o usuário;
- a sintaxe das linguagens de consulta permite que o usuário concentre sua atenção no modelo conceitual do banco de dados, sem se preocupar com os detalhes físicos da sua implementação;
- os volumes de informação que os SGBD conseguem manipular e armazenar são cada vez maiores;
- os tipos de dados armazenados não estão mais restritos a número, códigos ou palavras. Atualmente, tipos especiais de dados conseguem armazenar e manipular, por exemplo, imagens e sons.

A linguagem SQL apresenta grande parte de sua sintaxe amplamente suportada pelos SGBDs, mas cada fornecedor inclui construções adicionais na linguagem procedural ou cria APIs dedicadas para a realização de determinadas tarefas, como, por exemplo, a mineração de

dados. Entretanto, quando se deseja garantir a portabilidade de uma solução deve-se evitar ao máximo o uso de comandos e funções que não compõem a linguagem padrão. Exceções podem ser feitas se estas extensões ou APIs derem suporte para mais de um SGBD, mas mesmo assim, existe o risco delas comprometerem ou limitarem a portabilidade.

3.3 Árvores de decisão

Antes de se definir o que é uma árvore de decisão, suas aplicações e a sua forma de implementação, é conveniente situar esta técnica dentro do conceito de aprendizagem de máquina (*machine learning*), um das linhas de pesquisa da área de inteligência artificial.

Inicialmente, a pergunta que os pesquisadores ligados à aprendizagem de máquina procuram responder é: "Como podemos construir sistemas de computadores que automaticamente melhorem com a experiência, e quais são as leis fundamentais que governam todos os processos de aprendizagem?" (MITCHELL, 2006).

Esta é uma pergunta ambiciosa, cuja resposta é complexa e interdisciplinar, eis que envolve diferentes áreas de pesquisa, como, por exemplo, a estatística, a engenharia de computação, a psicologia e a neurociência, só para citar algumas.

Uma das formas mais fáceis de se aprender é a partir de observações ou de exemplos. Este tipo de aprendizagem é conhecido como indutivo, pois é baseado na inferência indutiva, um recurso utilizado pelo cérebro humano para derivar conhecimento novo (REZENDE, 2005). O aprendizado indutivo pode ser dividido em três tipos: aprendizado não supervisionado, supervisionado e por reforço ou recompensa (RUSSEL, 2004).

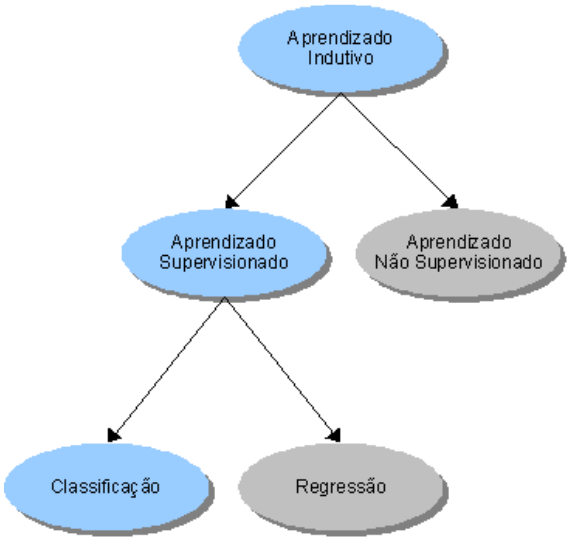


Figura 6 - Hierarquia dos tipos de aprendizado (REZENDE, 2005).

No aprendizado não supervisionado não se conhece *a priori* qual o conjunto de dados de saída. São fornecidos apenas os dados de entrada e neles são procurados padrões de comportamento, tentando agrupá-los de alguma maneira. Estes agrupamentos ou *clusters* são analisados posteriormente para determinar se os mesmos têm algum significado e revelam alguma informação até então desconhecida.

O aprendizado supervisionado utiliza a inferência indutiva para determinar, a partir de um conjunto de exemplos, uma função ou um classificador que represente a saída, com base nos dados de entrada. Para isto, faz uso de um algoritmo de aprendizado - ou indutor - que opera em um conjunto de exemplos, para inferir um classificador. Uma vez determinado este classificador, novos casos, onde não se conheçam as saídas, poderão ser examinados.

O aprendizado supervisionado pode ainda ser subdividido em:

- Classificação, quando os possíveis valores do atributo alvo forem discretos ou;
- Regressão, quanto os possíveis valores para atributo forem contínuos.

A Figura 6 ilustra a hierarquia dos tipos de aprendizado.

A essência da tarefa de indução de árvores de decisão é obter um conjunto de regras que classifique corretamente, não apenas os exemplos de treinamento, mas também novos casos novos cujos valores dos atributos sejam diferentes dos existentes no conjunto de treinamento. O processo de classificação usando árvore de decisão está representado na Figura 7.

As árvores de decisão "representam disjunções de conjunções de condições restritivas sobre os atributos do caso" (MITCHELL, 1997), onde as disjunções correspondem ao operador booleano "*or*" e representam a árvore e as sub-árvores, enquanto as conjunções correspondem ao operador booleano "*and*" e representam o caminho da raiz até a folha.

A representação do processo de geração de um classificador do tipo árvore de decisão pode ser visto na

Figura 8.

Nesta figura, o conjunto de treinamento está representado como um conjunto de exemplos composto por variáveis independentes (atributos), juntamente as variáveis dependentes (atributos alvo ou classes).

Quilan (1996) atribui mais uma qualidade as árvores de decisão:

> "Se os atributos são adequados, é sempre possível construir uma árvore de decisão que classifica corretamente cada objeto no conjunto de treinamento, e usualmente existem muitas árvores de decisão corretas".

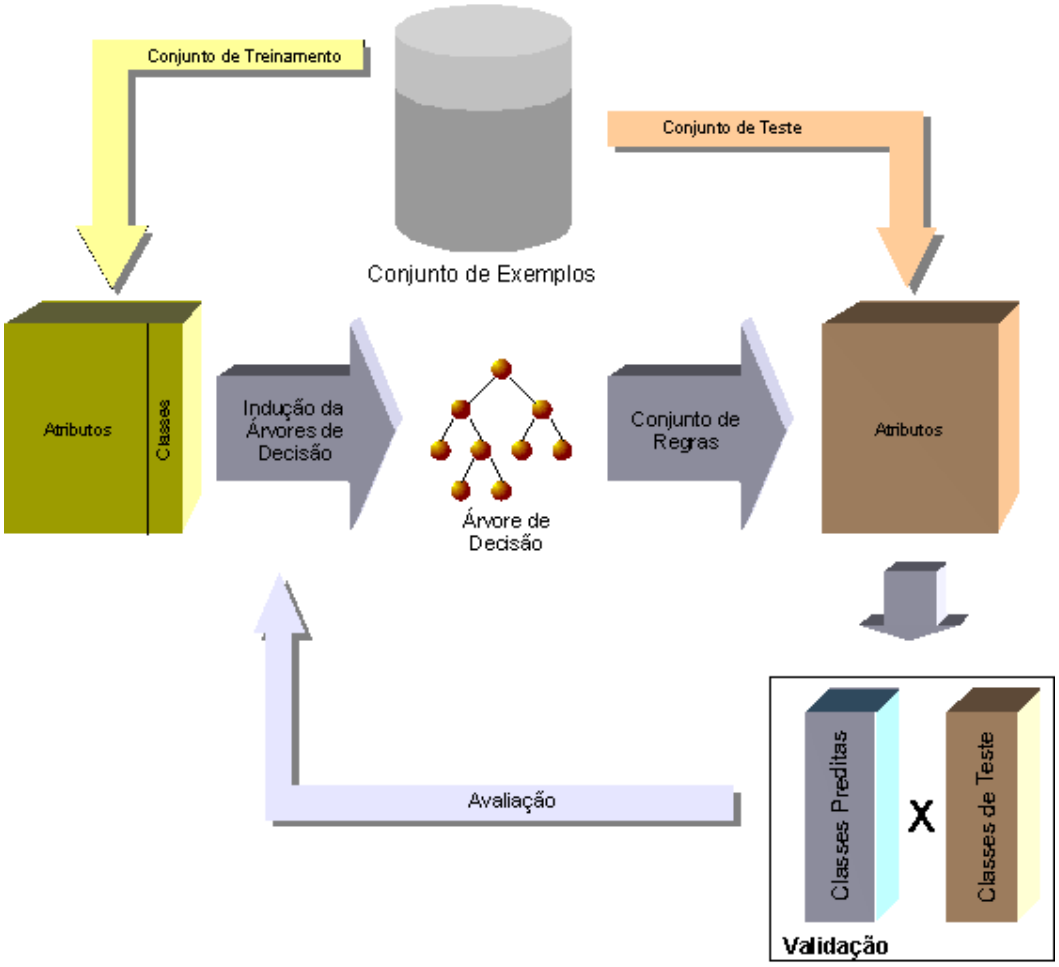


Figura 7 – Processo de classificação usando árvores de decisão. Os dados do conjunto de treinamento, já arranjados em tuplas compostas por atributos e classes, são processados pelo indutor de árvores de decisão. O conjunto de regras resultantes é aplicado ao conjunto de teste e as classes preditas são avaliadas quanto a sua precisão. Se o resultado estiver aquém do esperado o processo de indução é ajustado e realizado novamente.

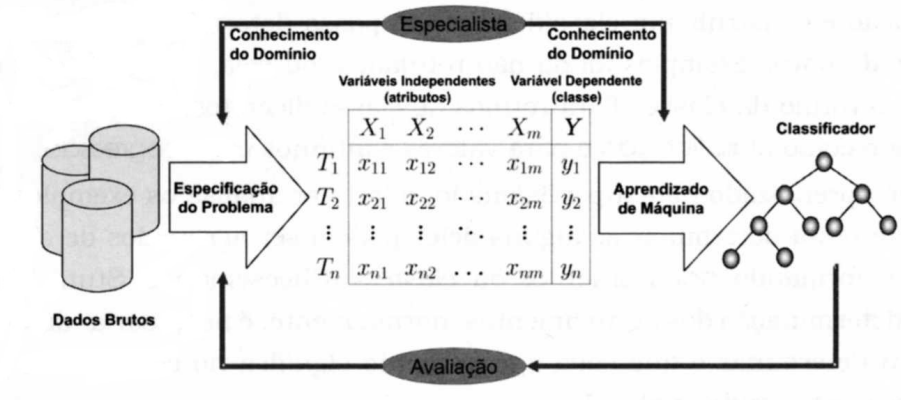


Figura 8 - Processo de geração do classificador tipo árvore de decisão. Extraído de (REZENDE, 2005).

As árvores de decisão são adequadas para problemas com as seguintes características:

- os casos são representados por pares atributo-valor;
- o alvo é representado por valores discretos de saída;
- a descrição na forma de disjunção for requerida;
- os dados de treinamento podem conter erros;
- os dados de treinamento podem conter atributos com valor desconhecido.

Muitos dos algoritmos, utilizam como estratégia uma busca *top-down* através de espaço de estado para construir as árvores de decisão. O espaço de estado representa, neste caso, todas as possíveis árvores de decisão. Este processo, essencialmente iterativo, envolve a decisão sobre qual atributo deve ser testado e quando a busca deve parar.

A escolha do atributo a ser testado deverá levar em conta o quão bom ele classifica sozinho os exemplos de treinamento em função do atributo alvo. Esta escolha é baseada em uma medida da informação, contida em cada conjunto de dados, e é feita para cada nó da árvore de decisão, desconsiderando os atributos já testados em nós anteriores. Como conseqüência disto, o nó raiz é sempre o melhor atributo[1], ou seja, o que melhor divide o conjunto de treinamento.

Uma vez selecionada a raiz da árvore ela é testada com base nos possíveis valores do atributo. Para cada teste realizado é associado um galho ao nó raiz. Os galhos assim gerados separam o conjunto de treinamento em igual número de subconjuntos; cada um dos subconjuntos com o seu conjunto de atributos que devem ser testados para a escolha daquele que melhor particiona o respectivo subconjunto. Após a seleção do atributo, para cada resultado do teste é associado um galho. Este processo continua sendo realizado até que não seja mais possível particionar os subconjuntos, pois todos os atributos alvo têm o mesmo valor. Nestas condições, o nó se transforma em uma folha que representa o valor do atributo alvo. O

Quadro 1 ilustra este processo na forma de um algoritmo baseado no ID3.

Pelo exposto até o momento podemos estabelecer as seguintes equivalências, conforme ilustrado na Figura 9:

- cada nó de decisão representa um atributo, A_n, a ser testado;
- cada galho representa um dos possíveis valores do atributo;
- o melhor atributo, ou seja, o que melhor particiona o conjunto está na raiz;

[1] O termo melhor atributo está relacionado apenas a capacidade que este tem de dividir o conjunto de treinamento em sub-conjuntos com atributos alvo distintos. De forma alguma este atributo é o mais importante para a tomada de decisão quando a árvore induzida é convertida em um conjunto de regras.

- as folhas representam os possíveis valores dos atributos alvo, C_m.

```
Indutor (Exemplos, Atributo_alvo, Atributos) {
        criar um novo nó Raiz para a árvore;
        se (todos os membros de Exemplos são da mesma classe C)
                Raiz = (árvore de um único nó com rótulo = C);
        caso contrário se (Atributos é vazio)
                Raiz = (árvore de um único nó com rótulo = valor mais comum do Atributo_alvo nos
                Exemplos)
        senão {
                A = membro dos Atributos que maximiza o Ganho(Exemplos, A);
                A é o atributo de decisão para a Raiz;
                para cada possível valor v de A {
                        adicionar um novo galho abaixo da Raiz, testando A = v;
                        Exemplos_v = subconjunto de Exemplos com A = v;
                        se (Exemplos_v é vazio) {
                                abaixo do novo galho adicionar uma folha com rótulo = valor mais
                                comum do Atributo_alvo nos Examplos;
                        } senão {
                                abaixo do novo galho adicionar uma sub-árvore
                                Indutor(Exemplos_v, Atributo_alvo, Atributos -{A});
                        }
                }
        }
        retorna Raiz;
}
```

Quadro 1- Algoritmo de geração de uma árvore de decisão.

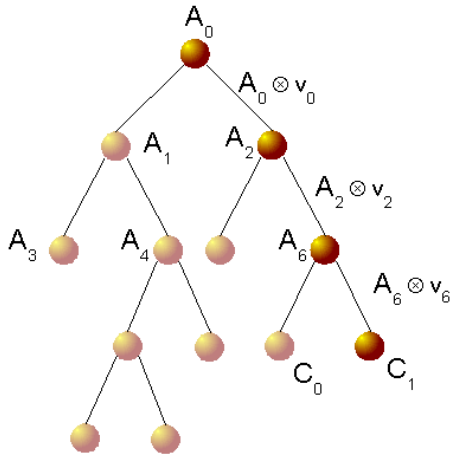


Figura 9 – O caminho da raiz da árvore até uma de suas folhas passa por diversos nós e galhos. Os nós representam os atributos testados e os galhos os possíveis resultados do teste. O caractere ⊗ representa um dos operadores relacionais (>, <, =, ...).

3.3.1 Critérios de seleção de atributos

Uma das tarefas mais importantes a ser realizada pelo algoritmo de indução é escolher qual dos atributos melhor particiona o conjunto de exemplos em cada iteração. Existem vários critérios para subsidiar a escolha deste atributo. Algumas possibilidades são (REZENDE, 2005):

- escolha aleatória;
- menor quantidade de valores;
- maior quantidade de valores;
- ganho de informação (*information gain*);
- índice Gini;
- razão de ganho (*gain ratio*).

Serão abordados neste trabalho apenas os critérios de ganho de informação (*information gain*) e de razão de ganho (*gain ratio*), por serem os utilizados no C4.5 e no seu antecessor, o ID3.

Ambos os critérios têm como base de formulação uma grandeza chamada de entropia e as proporções entre os elementos dos conjuntos.

A entropia é um conceito oriundo da termodinâmica e, originalmente, está associada ao grau de desordem de um sistema termodinâmico. Do ponto de vista da teoria da informação, a entropia está associada à pureza ou impureza de um conjunto aleatório de dados. Assim, quanto mais homogênea for a distribuição dos valores dentro do conjunto,

menor será a entropia associada a este conjunto. Para conjuntos onde a distribuição do número de elementos semelhantes é uniforme, a entropia apresenta o seu valor máximo, caracterizando a maior dispersão dos valores. Em outras palavras, a entropia mede a homogeneidade de um conjunto de elementos.

Matematicamente, a entropia de um conjunto S é definida como (2):

$$E(S) = \sum_{i=1}^{c} -p_i \,.\, log_2\, p_i \qquad (2)$$

Sendo:

E(S) a entropia do conjunto S;

c o número de valores distintos dentro do conjunto S;

p_i a proporção de elementos de S que possuem um valor distinto *i.*

Como a teoria da informação mede o conteúdo de informação em bits, a entropia especifica o número mínimo de bits necessários para codificar a classificação de um elemento arbitrário de um conjunto.

Para estabelecer um critério para a seleção do atributo que melhor particiona o conjunto de exemplos é necessário medir a efetividade do atributo em classificar um conjunto de dados. Esta medida, que é chamada de ganho de informação, é definida como (3):

$$gain(S, A) = E(S) - \sum_{v \in Dom(A)} \frac{|S_v|}{|S|} . E(S_v) \qquad (3)$$

Onde:

gain(S, A) é o ganho do atributo *A* no conjunto *S*;

v representa os possíveis valores de *A*;

S_v é o subconjunto de S onde A tem valor *v*;

$|S_v|$ é o tamanho do subconjunto S_v;

$|S|$ é o tamanho do conjunto *S*.

A razão $|S_v|/|S|$ representa a fração dos exemplos que pertencem a S_v e funciona como um de peso aplicado sobre a entropia calculada para cada subconjunto S_v. A somatória

passa então a contabilizar o valor esperado da entropia após S ter sido particionado com base no atributo *A*.

Apesar de usado com sucesso como critério de seleção pelo ID3 (QUINLAN, 1986), o ganho de informação apresenta uma séria deficiência relatada por Quinlan (1993). Esta deficiência surge quando a quantidade de subconjuntos gerados pelo atributo testado é grande. Nestas condições o valor representado pela somatória tende a ser bastante reduzido[2], maximizando o ganho de informação deste atributo. Como resultado existirá uma grande preferência para selecionar atributos com esta característica, o que prejudica o processo da predição.

Para corrigir esta deficiência do ganho de informação como critério de seleção, Quinlan (1993) propõe a inserção de um fator de correção, chamado de *split info*, para realizar uma espécie de normalização e minimizar esta tendência.

Por analogia à definição de entropia, o termo *split info(S, A)* é definido como (4):

$$splitInfo\ (S, A) = \sum_{v \in Dom(A)} \frac{|S_v|}{|S|} \cdot \log_2 \frac{|S_v|}{|S|} \qquad (4)$$

Isto, segundo ele, representa a informação potencial gerada pela divisão de *S* em *n* subconjuntos, enquanto que o ganho de informação mede a informação relevante para a classificação que surge da mesma divisão.

A correção proposta gerou um novo critério de seleção conhecido como *gain ratio* ou razão de ganho cujo valor é dado por (5):

$$gainRatio\ (S, A) = \frac{gain(S, A)}{splitInfo\ (S, A)} \qquad (5)$$

Apesar de o critério *gain ratio* ser robusto e tipicamente dar resultados mais consistentes do que o critério *gain information,* há referências que apontam para uma tendência a favor das divisões desbalanceadas na qual um subconjunto S_v é muito menor que os outros (QUINLAN, 1993).

[2] Este valor, em condições especiais pode ser igual a zero (QUINLAN, 1993, p.23).

3.3.2 Atributos contínuos

A estratégia mais simples para se lidar com atributos contínuos é discretizar estes valores na forma de novos atributos qualificadores (por exemplo: baixo, médio, alto, acima do peso, abaixo do peso), que representariam faixas dentro do espectro dos valores possíveis que o atributo testado pode assumir. O problema desta abordagem é que nem sempre isto é possível ou então os qualificadores não representariam de forma adequada a natureza do atributo. Se este não for o caso, é uma solução simples e válida.

É interessante notar que apesar dos valores de um atributo serem classificados como contínuos, existem dentro do conjunto dos dados, apenas um número finito destes valores que efetivamente são atribuídos ao atributo *A* dentro do conjunto. Estes valores podem então ser definidos como um conjunto discreto e ordenado de *n* valores, denotados por $\{v_1, v_2, v_3, ..., v_n\}$ que pode ser interpretado como o domínio deste atributo, *Dom(A)*.

Qualquer valor limiar situado entre v_i e v_{i+1} terá o igual efeito de dividir os casos naqueles cujos valores do atributo *A* situam-se em $\{v_1, v_2, ..., v_i\}$ e naqueles cujos valores estão em $\{v_{i+1}, v_{i+2}, ..., v_m\}$. A partir desta observação, a discretização dos atributos contínuos pode ser feita dinamicamente, para a partir destes valores, obter um ou mais valores limiares que vão, então, identificar um conjunto discreto de intervalos para a análise do algoritmo de geração da árvore de decisão. Os valores assim obtidos podem ser facilmente utilizados para particionar os dados.

Uma das formas, e provavelmente a mais simples, de determinar os valores limiares (*threshold*), é identificar em quais valores do atributo previsor ocorre uma mudança no valor do atributo alvo (MITCHELL, 1997). Uma vez identificados os valores adjacentes onde isto ocorre, pode-se tomar como candidatos a valor limiar, as médias obtidas para cada um destes pares de valores. Os valores candidatos, podem, então ser avaliados para determinar qual deles proporciona o maior ganho de informação; este será o valor escolhido como *threshold*. Uma extensão desta abordagem para a criação de múltiplos valores é discutida por Fayyad; Irani (1993) *apud* Mitchell (1997).

Outra forma de determinar os valores de limiar (*threshold*) é usar como critério de seleção o *gain* ou o *gain ratio* (QUINLAN, 1993). Nesta abordagem, dado um conjunto de valores do atributo *A* devidamente ordenados, qualquer valor situado entre v_i e v_{i+1} terá igual efeito de dividir os casos naqueles cujos valores do atributo *A* situam-se em $\{ v_1, v_2, ..., v_i\}$ e naqueles cujos valores estão em $\{ v_{i+1}, v_{i+2}, ..., v_m\}$. O usual é escolher o ponto médio de cada intervalo como um valor de limiar representativo. Existiriam, portanto, *m-1* valores

candidatos a serem examinados.

No C4.5, ao invés de se usar o ponto médio como valor candidato escolhe-se o maior valor dentro do intervalo de modo que ele não exceda o ponto médio dado por (6):

$$\frac{v_i + v_{i+1}}{2} \qquad (6)$$

Isto assegura que os valores de *threshold* constantes da árvore ou das regras ocorrem nos dados.

Para determinar qual ou quais os valores candidatos se transformarão ou serão usados como valores de *threshold* calcula-se para cada um deles o ganho de informação ou o *gain ratio*. Com estes valores, analisa-se o resultado para verificar a distribuição de picos. Os valores *threshold* a serem escolhidos são os valores que geraram os picos nos valores de *gain* ou *gain ratio*, conforme ilustrado na Figura 10.

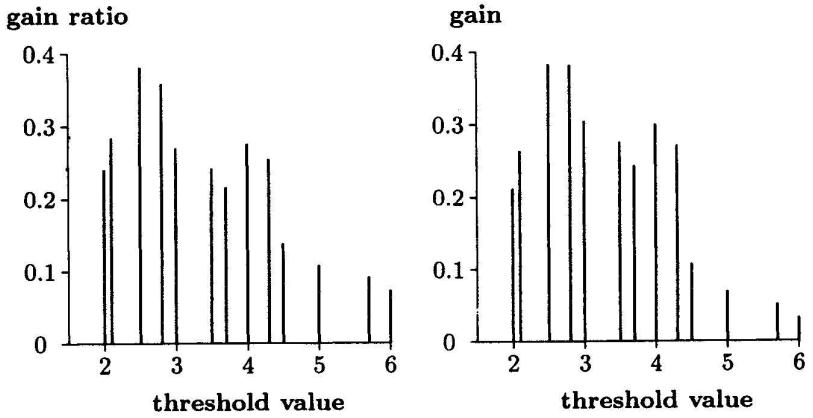


Figura 10 – Dentre os candidatos a threshold, os valores 2,5 e 4 apresentam os maiores valores para gain ratio e gain e portanto serão os escolhidos. Extraído de (QUINLAN, 1993).

Os valores obtidos serão os valores de *threshold*, a serem usados na avaliação e expansão da árvore de decisão.

No entanto, a escolha feita por estes testes seria parcial em favor dos atributos contínuos com valores distintos numerosos (QUINLAN, 1995). Para corrigir este desvio, Quinlan (1995) propõe uma correção, que consiste em usar o princípio MDL – *Minimum Description Length* - para criar um ganho ajustado (*adjusted gain*) e o *gain*, ao invés do *gain ratio*, para escolher os valores de *threshold*, dentre os valores candidatos; mas, uma vez escolhido os *thresholds*, a seleção do atributo para teste continua tendo como base o critério

do *gain ratio*, usando, no entanto, o ganho ajustado.

Existem muito outros métodos e abordagens para fazer o algoritmo lidar com os atributos contínuos. Perner e Troutzsch (1998) descrevem métodos de discretização para múltiplos intervalos para serem usados em árvores de decisão usando histogramas. Outras possibilidades podem ser vistas em Frank; Witten (1996).

3.3.3 Atributos com valores desconhecidos

A existência de atributos sem valor é comum quando se usam dados reais. Isto pode ser conseqüência do processo de pré-processamento, que descartou o dado por ele estar corrompido ou fora do formato esperado. Outra possibilidade é ele simplesmente nunca ter existido, seja pela sua inexistência ou por falha no processo de entrada dos dados.

Assim, como no caso dos atributos contínuos, esta situação também precisa ser tratada, já que, como mencionado anteriormente, classificar um caso usando uma árvore de decisão requer tomar o galho apropriado para cada nó, e uma vez que cada teste é baseado em um único atributo, o resultado do teste não poderá ser determinado se o valor deste atributo não for conhecido.

Caso o valor do atributo em questão seja desconhecido, têm-se duas opções:

- descartar esta parcela de dados e assumir que alguns casos de teste são tomados como não classificáveis. Isto pode significar que uma parcela considerável dos dados pode não ser utilizada e, como conseqüência, muitos casos deixarão de ser classificados. Tem-se, então, um comprometimento da própria precisão do resultado de classificação e da capacidade de encontrar padrões;
- modificar o algoritmo para que ele possa ser capaz de lidar com atributos sem valores.

A segunda opção pode ser realizada em uma fase de pré-processamento, antes da aplicação do algoritmo de indução, e levanta três pontos importantes que devem ser analisados (QUINLAN, 1989) (QUINLAN, 1993):

- o critério de escolha do atributo pode requerer a comparação do resultado de testes baseados em atributos com diferentes valores desconhecidos;
- uma vez selecionado um teste sobre um atributo A, os casos com valores desconhecidos para A não podem ser associados com um resultado ou atributo alvo. Como tratar estes casos durante a divisão do conjunto de testes em

subconjuntos é uma questão que deve ser respondida;

- quando a árvore de decisão for utilizada para classificar um caso novo, deve-se pensar em como proceder quando for encontrado um teste para o qual o valor do atributo é desconhecido.

Além disto, tratar adequadamente os atributos com valores desconhecidos é importante em 3 momentos distintos, que exigem abordagens diferentes:

- avaliação dos atributos para teste;
- particionamento do conjunto de teste;
- classificação de novos casos.

Muitas das abordagens propostas para responder estas questões se resumem basicamente em preencher o valor desconhecido com:

- o atributo mais provável;
- um valor de atributo baseado na inter-relação entre os valores de diferentes atributos;
- analisando a distribuição dos valores do atributo.

Em Quinlan (1989), algumas abordagens são comparadas empiricamente e o resultado mostrou que, apesar de algumas serem visivelmente inferiores, nenhuma delas é "uniformemente superior".

O C4.5 (QUINLAN, 1993), quando avalia os atributos para teste durante a fase de treinamento, reduz o aparente ganho de informação do atributo de teste *A* proporcionalmente ao número de casos com valores desconhecidos deste atributo. A justificativa é que os valores desconhecidos do atributo *A* não produzem informação.

Esta adaptação do critério de avaliação do atributo pode ser expressa da seguinte maneira: supondo a fração *F* de valores conhecidos para o atributo *A* dentro do conjunto de teste *S* a nova versão do ganho pode ser escrita como (7):

$$gain'(S, A) = F \cdot gain(S, A) \qquad (7)$$

De forma semelhante o *split info* (S, A) pode ser alterado para "corrigir" também o *gain ratio*, considerando que os casos com valores desconhecidos devem ser tratados como um grupo adicional. Ou seja, se o teste tem "n" resultados, o valor *split information* deve ser calculado como se o teste dividisse o conjunto em (n + 1) subconjuntos.

O particionamento do conjunto de treinamento no C4.5 adota uma abordagem baseada na probabilidade de um caso, com valor do atributo desconhecido, pertencer a um

subconjunto. São utilizados nesta situação as definições modificadas para o ganho e *split info* para a seleção do teste.

Suponha-se que o atributo de teste escolhido A, com resultados R_1, R_2,...,R_n tem resultados desconhecidos em alguns dos casos do conjunto de treinamento S. Quando um caso do conjunto de treinamento S, com resultado conhecido R_i, é atribuído ao subconjunto S_i, a probabilidade deste caso pertencer ao subconjunto S_i é 1 e em todos os outros é 0. Entretanto, quando o resultado não é conhecido, somente uma inferência probabilística pode ser feita. Para isto, é associado a cada caso do subconjunto S_i um peso estatístico representando a probabilidade do caso pertencer ao subconjunto. Se o caso tem um resultado conhecido o seu peso é um, caso contrário o peso é a probabilidade do resultado R_i neste ponto. Cada subconjunto S_i é agora uma coleção de frações que representam probabilidades de casos e $|S_i|$ deve ser reinterpretado como a soma dos pesos fracionais dos casos no conjunto.

A probabilidade final é estimada como a soma dos pesos dos casos em S com resultado conhecido ser igual a R_i dividido pela soma dos pesos dos casos em S com resultados conhecidos.

Uma abordagem similar é utilizada quando a árvore de decisão é usada para classificar um caso ainda não visto. Nesta situação, se em um nó de decisão é encontrado um atributo relevante cujo valor é desconhecido, e o resultado do teste não pode ser determinado, o sistema explora todos os possíveis resultados e combina as classificações resultantes aritmeticamente.

Outra possibilidade para tratar atributos desconhecidos é tratar o valor desconhecido como outro possível valor para o atributo (FRANK, WITTEN, 1996).

Esta mesma possibilidade é abordada em Bruha; Franek (1996) que, ao comparar várias rotinas tradicionais para processar atributos desconhecidos, concluem que esta abordagem supera todas as demais analisadas. Foram utilizadas nestes experimentos quatro bases de dados diferentes muito utilizadas em pesquisas de inteligência artificial.

3.3.4 Under e Overfitting

Durante o processo de indução da árvore de decisão pode ocorrer uma adequação excessiva do resultado ao conjunto de treinamento produzindo árvores complexas com um grande número de regras (ou nós de decisão). Todavia este excesso de precisão é indesejável, pois gera soluções mais complexas, com um grande número de nós de decisão e é muito específica para o conjunto de dados de treinamento, prejudicando a capacidade de

generalização esperada para o resultado.

Este fenômeno, conhecido como *overffiting*, implica que quanto mais ajustadas estiverem as regras ao conjunto de treinamento, piores os resultados obtidos quando se aplicam as regras ao conjunto de teste. Como conseqüência, os erros de classificação de novos exemplos tendem a aumentar proporcionalmente ao aumento do *overfitting*. A Figura 11 ilustra este fenômeno.

Pode parecer estranho esta adequação excessiva provocar uma imprecisão na classificação de novos exemplos, tendo em vista, que os exemplos do conjunto de treinamento e de teste são partes do mesmo conjunto de dados. Uma explicação é que o conjunto de dados pode conter erros aleatórios nos dados (ruídos) e, como uma parcela destes erros está presente no conjunto de treinamento, o algoritmo de indução, ao analisar os dados e gerar as regras, acaba levando estes erros em consideração. Logo, é preciso evitar o *overfitting*, para melhorar a capacidade de predição da árvore de decisão para outros conjuntos de dados.

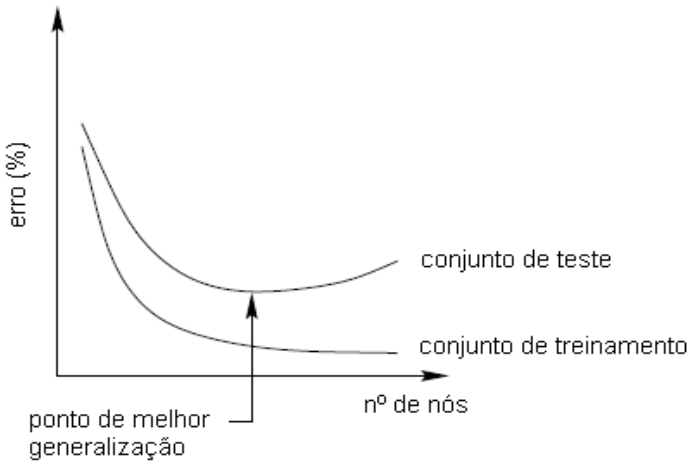


Figura 11 - Fenômeno do *overffiting*. Uma árvore com um número muito grande de nós tende a aumentar o erro de predição no conjunto de teste.

Existem duas possibilidades para tratar o *overfitting*:

- parar o crescimento da árvore precocemente antes dela classificar perfeitamente os dados de treinamento;
- permitir o *overfitting* e então fazer uma simplificação na árvore, retirando sub-árvores para aumentar a sua capacidade de generalização.

A primeira possibilidade, chamada de *pre-prunning*, apesar de ser mais fácil de implementar, pode não obter bons resultados, já que é difícil estimar precisamente o momento para se parar o crescimento da árvore. Se o ponto limiar da poda não for bem escolhido, pode-

se ter:

- uma pequena simplificação, devido a uma parada tardia no processo de indução, ou;
- o ocultamento de divisões (sub-árvores), que poderiam ser interessantes.

Quinlan (1986) propõe como possível critério para determinar o limiar de poda, o teste de χ^2 (leia-se: chi quadrado)[3]. Mas, apesar de bastante satisfatório em alguns domínios, os resultados usando esta técnica se mostraram irregulares, o que provocou o abandono desta abordagem (QUINLAN, 1993).

Já o ocultamento de sub-árvores, tem uma implementação mais complexa, pois é preciso determinar quais sub-árvores podem ser descartadas, sem perda de informação e sem comprometer a capacidade de classificação. Esta segunda possibilidade, chamada de *post-prunning*, alcança na prática maiores níveis de sucesso que a primeira (MITCHELL, 1997).

Independentemente do método escolhido, o processo de poda procura remover partes da árvore que não contribuem para a precisão na classificação de novos casos. Como resultado tem-se um conjunto de regras mais simples sem grande perda na capacidade de predição.

Uma vez escolhido o tipo de poda, a próxima questão é determinar o tamanho final da árvore que produz melhores resultados. Este ponto é justamente aquele onde se consegue a menor taxa de erros com a maior quantidade de nós na árvore, o ponto de melhor generalização, mostrado na Figura 11.

Para isto existem pelo menos três possibilidades (MITCHELL, 1997):

1. usar conjuntos de dados separados, um para a fase de treinamento e outro para a fase de testes, para avaliar a pós-poda (*post-prunning*);
2. usar todos os dados de treinamento e aplicar um teste estatístico para estimar se a expansão ou poda da árvore aumenta a performance da árvore de decisão;
3. aplicar o princípio MDL ao avaliar a complexidade de codificar os exemplos de treinamento e a árvore de decisão para parar o processo de crescimento quando o tamanho for mínimo.

A primeira abordagem é a mais usual e consiste em separar os dados em dois conjuntos:

- conjunto de treinamento: usado na indução da árvore de decisão;
- conjunto de teste ou validação: usado para avaliar a precisão da classificação

[3] O teste de χ^2 é usado no tratamento estatístico de dados experimentais para avaliar o "quanto os valores estão espalhados em torno da verdadeira curva" (HELENE, 1981).

após o processo de poda.

Existem muitas variações desta abordagem, sendo que as principais e mais conhecidas são: *reduced error prunning* e *rule post-prunning*.

O *reduced error prunning*, proposta por (QUINLAN, 1997), é um processo iterativo que considera cada nó de decisão como um candidato a ser removido e que continua até o ponto em que podas adicionais diminuem a precisão da árvore sobre o conjunto de testes. A escolha do nó a ser removido é baseada na capacidade que este nó tem de incrementar, quando removido, a precisão da árvore sobre o conjunto de teste ou validação.

O *rule post-prunning*, por sua vez, é um método de bastante sucesso que avalia a performance da árvore no próprio conjunto de testes, usando uma estimativa pessimista, para compensar a tendência de super ajustamento - *overfitting* - sobre os dados de treinamento. Este método pode ser resumido em 4 passos (MITCHELL, 1997):

1. usando o conjunto de treinamento, construir a árvore de decisão com a maior precisão possível permitindo o *overfitting*;
2. uma vez criada a árvore convertê-la em um conjunto de regras;
3. podar cada regra removendo qualquer pré-condição que aumente a precisão estimada (generalização);
4. classificar as regras podadas por sua precisão estimada e as considerar nesta ordem quando classificar novos exemplos.

O C4.5 (QUINLAN, 1993) usa uma variação deste método para realizar a poda na árvore de decisão.

4. Trabalhos Relacionados

O uso de ferramentas e métodos de inteligência artificial para melhorar a confiabilidade e permitir a predição de falhas em equipamentos e sistemas é cada vez mais utilizado no dia-a-dia.

Em Murray; Hughes; Delgado (2005), por exemplo, são utilizados métodos de aprendizagem de máquina para prever ou predizer falhas no funcionamento de discos rígidos usados em computadores. Os dados utilizados ficam armazenados no próprio dispositivo e representam as condições de funcionamento do dispositivo, sendo normalmente, monitorados por um algoritmo simples, que verifica quando determinados limiares são atingidos para então sinalizar, ao sistema operacional ou a algum aplicativo de terceiros, que o disco rígido está na eminência de falhar, permitindo então que o usuário possa fazer o *backup* dos seus dados. No entanto, dada à diversidade da natureza dos parâmetros, esta solução não é precisa, gerando um número de falsos alarmes maior que o aceitável pelos fabricantes. Os autores, com base na comparação do desempenho de diversos métodos para detectar a ocorrência de eventos raros, propõem um novo algoritmo para múltiplas instâncias, baseado em um classificador Bayesiano simples, especialmente desenhado para controlar a taxa de falsos alarmes. Este novo algoritmo conseguiu um desempenho consideravelmente melhor, mas que, segundo os autores, ainda poderia ser melhorado.

Na linha de auxílio ao diagnóstico, Cardoso Junior, Rolim e Zürn (2004), discutem o uso de técnicas de inteligência artificial para o diagnóstico de falhas em sistemas de potência da malha de distribuição de energia elétrica. Segundo os autores, tais falhas em sistemas de distribuição de alta potência, dadas a sua complexidade construtiva e operacional e a grande variedade de dispositivos de proteção e de segurança envolvidos, tornam o diagnóstico e a localização do ponto ou equipamento com falha uma tarefa não trivial, sendo essencial minimizar os riscos causados por uma má interpretação dos eventos sinalizados. A tudo isto se soma, ainda, a necessidade de restabelecer o fornecimento de energia o mais rapidamente possível. Dentre as técnicas de inteligência artificial existentes, foram analisados pelos autores os sistemas especialistas, as redes neurais artificiais, os algoritmos genéticos, as redes de Petri e a lógica difusa ou *fuzzy*, destacando-se as vantagens e desvantagens de cada uma delas, quando aplicadas ao problema estudado. Por fim, os autores concluem que a aplicabilidade de uma ou outra técnica "irá depender do tipo de dados disponíveis, e de como o problema é formulado ou modelado" (sic).

Usada muito intensamente na área comercial e de serviços, a indução de árvores de

decisão e a mineração de dados estão cada vez mais presentes nos setores industriais, que buscam formas eficientes de aumentar a qualidade dos seus produtos e aperfeiçoar os processos produtivos que, muitas vezes, são extremamente complexos e associados ao uso de altas tecnologias.

Neste contexto, Braha; Shmilovici (2003), dão um bom exemplo disto ao descreverem o uso de árvores de decisão para melhorar o processo litográfico presente na indústria de semicondutores. Esta indústria possui um processo produtivo bastante complexo no qual garantir altos níveis de produtividade com alta qualidade é um fator importante neste competitivo segmento industrial, pois pequenas variações durante a manufatura dos dispositivos afetam o seu funcionamento, gerando, no caso dos circuitos integrados de alta densidade, até 80% de perdas.

O uso de árvores de decisão em detrimento de outros métodos, como as redes neurais, foi motivado pela facilidade de interpretação dos resultados pelo corpo técnico, já que o seu resultado pode ser facilmente expresso em regras do tipo "*if - then*".

Para melhorar o entendimento das intrincadas interações entre os diferentes processos e para obter conhecimento de alto nível que pudesse ser usado para melhorar a qualidade do processo, foi empregado pelos autores para a indução das árvores de decisão, um produto comercial baseado no C4.5, chamado de KnowledgeSEEKER. No experimento foi utilizado um conjunto de 1054 registros de produção - que representam os 45 subprocessos de um processo litográfico usado na linha de produção. Mesmo com as limitações detectadas, os benefícios obtidos classificaram o experimento como um sucesso, pois este permitiu:

- detectar variações de qualidade associadas a períodos de produção e ao uso de maquinários específicos;
- determinar as relações causa-efeito associadas aos processos litográficos;
- otimizar a receita do método produtivo de processos específicos;
- localizar defeitos nos equipamentos de produção de semicondutores.

Em casos com uma pequena quantidade de dados como mencionado por Braha; Shmilovici (2003) no exemplo da indústria de semicondutores, o uso de um sistema de arquivos para armazenar os dados pode ser suficiente. Mas, para trabalhar grandes conjuntos de dados, que contém dezenas ou centenas de milhares de registros armazenados, em sistemas de arquivos e que não podem ser alocados simultaneamente na memória, é necessária outra solução. Como os dados necessários para a mineração de dados ou simplesmente a geração de árvores de decisão estão muitas vezes armazenados em banco de dados, a solução encontrada é a integração entre os bancos de dados e os mecanismos de mineração de dados. Esta

integração tem sido cada vez maior mais estudada, motivando a publicação de uma série de trabalhos nesta área. Foram identificadas basicamente cinco abordagens do suporte dos sistemas gerenciadores de banco de dados, SGBD, para operações de mineração de dados:

1. adicionar novas construções à linguagem SQL;
2. utilizar APIs específicas para mineração de dados ou usar tipos definidos pelo usuário;
3. usar operadores ou primitivas especiais úteis para a mineração de dados, mas que não implementam uma tarefa particular de mineração de dados;
4. usar extensões da linguagem procedural dos SGBDs;
5. criar *frameworks* que não aumentem a quantidade de instruções da linguagem SQL.

No entanto, segundo Netz et al. (2001), construir aplicações de mineração de dados em banco de dados relacionais não é trivial e requer um trabalho significativo por parte dos desenvolvedores de aplicações. Para incrementar o uso desta solução em ambientes empresariais é preciso facilitar a preparação do modelo e dos dados, trabalhando diretamente na base de dados relacional, Netz et al. (2001), propõem a utilização de uma API chamada "*OLE DB for Data Mining*", que, segundo os autores, é independente do fornecedor do SGBD e suporta conceitos de mineração de dados.

Os componentes básicos da solução proposta são os dados de entrada na forma de um conjunto de casos e o modelo de mineração de dados. O conjunto de casos pode ser interpretado como uma *view*, onde todos os dados relacionados a uma instância estão juntos em registros de uma única "tabela". Já o modelo de mineração de dados é composto pelo conjunto de atributos a ser predito, pelo conjunto de atributos usados na predição e pelos metadados destes atributos. Dentre os benefícios apontados estão:

1. a facilidade de aplicar os algoritmos tradicionais de mineração de dados;
2. o aumento da escalabilidade;
3. a facilidade de uso pelo desenvolvedores das corporações, pois as sintaxes utilizadas são muito semelhantes as de outros produtos já utilizados nas empresas.

Em uma outra abordagem, Blockeel et al. (2007), propõem um *framework* para integrar o processo de mineração de dados ao banco de dados, sem o uso de extensões à linguagem SQL, de tal forma que possam ser utilizados diversos modelos de mineração de dados sem que sejam necessárias alterações significativas na representação dos dados. Em contrapartida, eles propõem uma extensão ao esquema de tabela do banco de dados com a

adição de novas tabelas - na verdade *views* (*mining views*) – nas quais estarão contidas, por exemplo, regras de associação, árvores de decisão e outros modelos preditivos ou descritivos necessários. Neste *framework* todas as tuplas que satisfazem uma condição especificada (conceito) estão representadas em uma tabela chamada de tabela de conceitos, que juntamente com a de dados, podem representar, quando utilizadas em conjunto com a representação do modelo na forma de uma *view* (*virtual mining view*), uma série de modelos de mineração de dados. As *virtual mining views* são construídas de acordo com o modelo de mineração a ser utilizado.

Silva; Xexéo (1999), após identificar a tendência do uso de primitivas das linguagens de consultas dos SGBDs - ou mesmo uma extensão destas - em um grande número de algoritmos de mineração de dados para auxiliar na tarefa de classificação e da coleta de dados para a montagem de sumários estatísticos, propõem a descrição de uma primitiva para dar suporte à tarefa de mineração de dados, usando uma extensão à linguagem SQL. Normalmente, para obter as informações necessárias para a classificação, o SGBD faz várias requisições ao servidor, sendo os resultados destas requisições agrupados e disponibilizados ao solicitante. A primitiva proposta tem por objetivo produzir uma matriz que contenha estes dados através de uma única requisição ao servidor numa tentativa de melhorar o desempenho das consultas. A solução encontrada para a implementação desta primitiva foi criar uma extensão à cláusula "*group by*" da linguagem SQL. Nesta extensão, a cláusula passa a aceitar um atributo virtual que representa simultaneamente todos os atributos a serem agrupados. No entanto, existem restrições quando os domínios dos atributos previsores são de natureza distinta (discretos e contínuos). Conforme ficou demonstrado nos experimentos realizados, em decorrência da forma como a extensão trabalha, ela se beneficia de sistemas que trabalham em paralelo, retornando os resultados com mais eficiência.

Nesta mesma linha de trabalho, Sattler; Dunenmann (2001), ressaltam as vantagens de se realizar a mineração diretamente no banco de dados e alertam para a desvantagem do uso da linguagem SQL como ferramenta de mineração devido à sua baixa performance, já que as operações necessárias devem ser implementadas usando uma série de consultas escritas em SQL. Neste artigo, além de serem apresentados os resultados do trabalho sobre primitivas em bancos de dados para classificadores do tipo árvore de decisão, é feita a implementação de uma extensão à linguagem SQL para um SGBD comercial (o Oracle 8i). Todos os operadores desenvolvidos são utilizados em conjunto com um indutor de árvores de decisão - uma implementação do ID3 - cuja representação foi armazenada em uma tabela no banco de dados. Além disto, também é apresentada uma estrutura especial de índice para otimizar o acesso aos

dados pelas consultas implementadas. A fase de poda foi implementada usando o princípio do *Minimum Description Length,* MDL (QUINLAN, 1993). Os experimentos realizados mostraram os bons resultados obtidos com o uso das soluções propostas, quando comparadas aos mesmos experimentos realizados sem estas soluções. Por fim, os autores concluem, com base nos experimentos, que o uso das primitivas implementadas trouxe benefícios e que estes poderiam ser maiores se fossem utilizadas APIs dedicadas e se a integração com o sistema de banco de dados fosse mais estreita.

Como normalmente os dados estão armazenados em sistemas de bancos de dados relacionais é interessante que os métodos de mineração de dados possam atuar em estruturas multi-relacionais, conforme proposto por Atramentov; Leiva; Honavar (2003). A implementação proposta, chamada de MRDTL-2 – *Multi-Relational Decision Tree Learning Algorithm*, é um aperfeiçoamento do mesmo algoritmo apresentado em Leiva et al. (2002). Em ambos os trabalhos são desenvolvidos algoritmos que permitem a geração das declarações SQL para pesquisa no banco de dados, a partir da descrição de gráficos de seleção que representam o relacionamento entre as entidades do banco de dados. As declarações geradas não apresentam qualquer cláusula fora do padrão da linguagem SQL e, portanto, podem ser portadas para qualquer sistema de banco de dados relacional que suporte esta linguagem. A árvore de decisão gerada "pode ser visualizada como um conjunto de declarações SQL associadas com os gráficos de seleção". Os resultados obtidos, no tocante a precisão, são semelhantes aos reportados por outros estudos e o desempenho, quando comparado com a versão anterior do MRDTL, teve um significativo aumento, conforme relatado em (ATROMENTOV, 2003), mostrando, segundo os autores, que a abordagem proposta é competitiva com as outras implementações de algoritmos de aprendizado de classificadores.

Por outro lado, Onoda; Ebecken (2001), avaliam a implementação de um algoritmo próprio de árvore de decisão implementado de duas formas diferentes: uma utilizando a linguagem Java acessando um banco de dados Oracle, utilizando JDBC, e outra utilizando a linguagem de programação do banco de dados, PL/SQL. Os resultados dos testes, no tocante à precisão da precisão árvore, mostram-se dentro do esperado. O desempenho computacional, no entanto, ressaltou a semelhança dos tempos de execução entre os dois métodos estudados, fato que não era esperado, pois se acreditava que o programa em Java fosse ter um desempenho inferior ao implementado usando a linguagem PL/SQL. Este resultado mostrou que o uso de uma implementação em Java se mostra competitiva não só devido ao bom desempenho, mas também a portabilidade propiciada por esta linguagem de programação.

Nota-se, pelo exposto, que o uso de ferramentas de inteligência artificial está cada dia

mais integrado às necessidades dos ambientes empresariais. No caso específico do uso de árvores de decisão e de outras ferramentas de mineração de dados acoplados a sistemas de banco de dados, percebe-se que muitas pesquisas estão sendo feitas no sentido de tornar esta integração uma realidade cada vez mais presente e acessível.

5. Implementação

No decorrer deste estudo serão apresentadas as diversas etapas do processo de geração de árvores de decisão, desde a seleção dos dados até a implementação do algoritmo de indução para alcançar os objetivos mencionados no capítulo 2:

- usar árvores de decisão como uma ferramenta para avaliar quais são os fatores predisponentes para a ocorrência de falhas em equipamentos, mais precisamente em motores elétricos de potência;
- implementar um processo de geração de árvores de decisão que acesse diretamente o banco de dados onde os dados estão armazenados.

Durante a fase de implementação foram usados preferencialmente, ferramentas e aplicativos *freeware* ou *open source*. Não foi utilizado também qualquer tipo funcionalidade que prejudicasse a portabilidade ou a escalabilidade da solução. Com base nestes requisitos chegou-se a seguinte configuração:

- banco de dados: PostgreSQL versão 8.2[4];
- administrador de banco de dados: pgAdmin III, versão 1.6.3[5];
- linguagem de programação para a implementação do algoritmo de indução e da rotina de pré-processamento: Java2 release 1.5[6];
- linguagem de acesso ao banco de dados: SQL padrão;
- algoritmo de indução usado como base para a implementação: C 4.5;
- ambiente de desenvolvimento: IDE Eclipse[7]
- sistema operacional: Windows XP – SP2[8].

5.1 Breve descrição da rotina de manutenção nas oficinas

Qualquer que seja a circunstância que tenha provocado a remoção do motor, quando eles chegam às oficinas, os dados informados pelas equipes de campo são registrados no sistema informatizado de gestão de oficinas. Os principais são:

- o tipo (motivo) de entrada do motor (por corretiva ou revisão programada);
- a descrição da falha ou a atividade a ser realizada;

[4] Disponível para *download* em: http://www.postgresql.org/
[5] Disponível para *download* em : http://www.pgadmin.org/download/
[6] Disponível para *download* em: http://www.java.com/pt_BR/
[7] Ambiente de desenvolvimento integrado para o desenvolvimento de aplicações em Java e em outras linguagens. Disponível para *download* em: http://www.eclipse.org/downloads/
[8] *Service Pack 2*. Mais informações em: www.microsoft.com/brasil/windowsxp/sp2/default.mspx

- o número de série;
- a data de entrada em oficina;
- a data de remoção do motor.

Após o reparo ou revisão do motor, a data de liberação para o usuário ou de envio para o almoxarifado, é registrada no banco de dados deste mesmo sistema. Este sistema de gestão das oficinas foi desenvolvido com a linguagem Clipper, usa como banco de dados um conjunto de tabelas no formato dBase III e é executado em microcomputadores do tipo PC.

As tabelas que contém estes dados estão representadas na Figura 12.

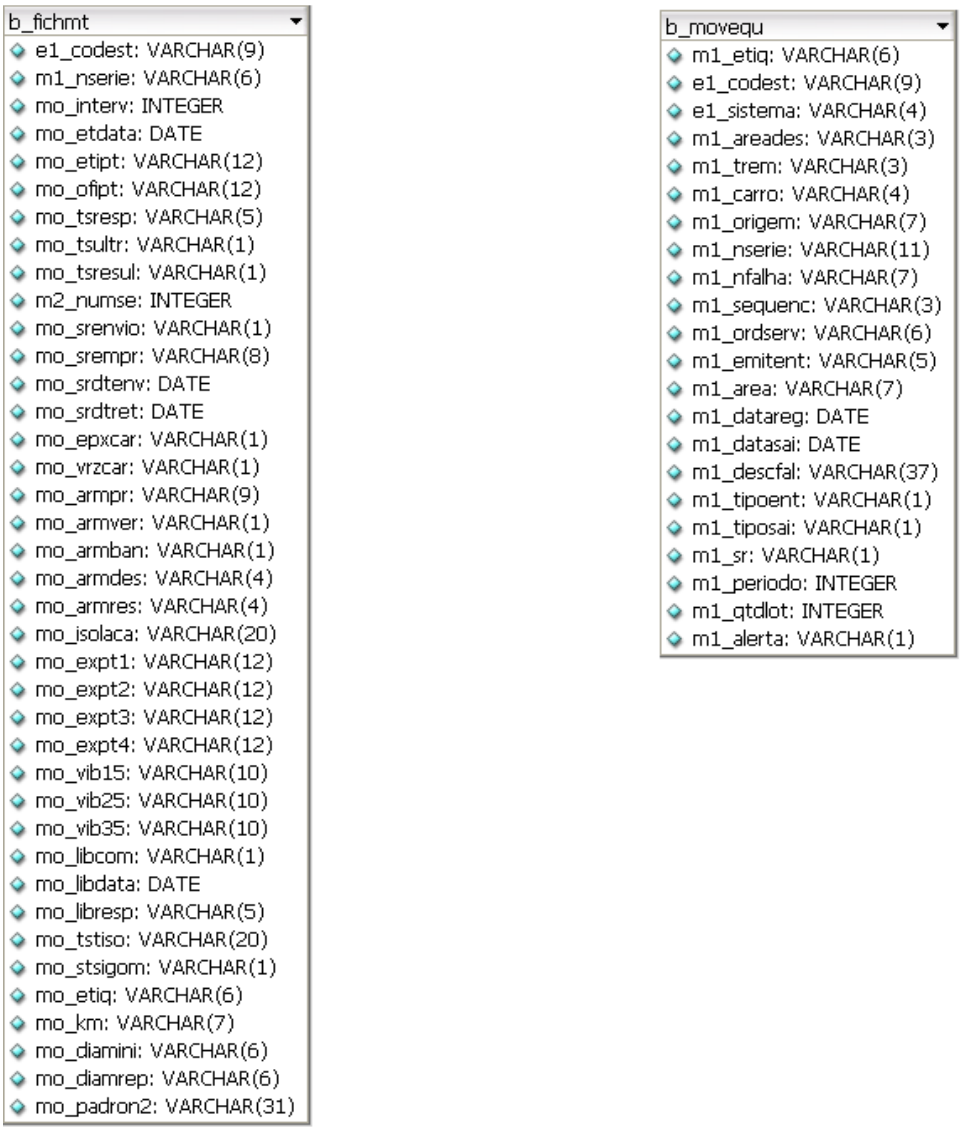


Figura 12 -Tabelas utilizadas pelo sistema de registro de intervenções responsáveis pelo registro do histórico de manutenção dos motores.

Durante o processo de manutenção, todas as atividades realizadas nos motores e as medições realizadas são registradas pelos eletricistas e técnicos em uma ficha que acompanha cada motor durante o seu período dentro da oficina. Ao término do processo de manutenção, o conteúdo desta ficha é transferido para o sistema de gestão e passa a fazer parte do histórico de manutenção destes equipamentos.

Das tabelas existentes neste sistema, duas são essenciais para atingir os objetivos propostos neste trabalho (Figura 12). São elas:

b_movequ: contém os dados relativos à solicitação de reparo, como, por exemplo, a data de retirada do equipamento de operação, a descrição da falha, origem do equipamento, o registro funcional do técnico que atuou ou removeu o equipamento e o mais importante, o identificador do tipo de entrada – manutenção corretiva ou preventiva. Este atributo foi o escolhido como atributo alvo para a árvore de decisão:

b_fichmt: contém os detalhamentos das medidas realizadas e dos processos pelos quais passou o motor. Representa a transcrição da ficha que acompanha o motor desde o início do seu processo de manutenção em oficina. Nesta tabela estão concentrados os atributos que são analisados durante a geração da árvore de decisão.

5.2 Pré-processamento

Como os conceitos de chave primária e chave estrangeira não estão presentes nas tabelas que constituem o banco de dados do sistema de gestão das oficinas, o uso de uma abordagem multirelacional semelhante à proposta por Leiva; Atramentov; Honavar (2002) e Atramentov; Leiva; Honavar (2003), onde o processo de mineração de dados é feito diretamente nas tabelas operacionais normalizadas, sem a necessidade de agrupar todos os dados em uma única tabela não normalizada.

Como o objetivo deste trabalho é utilizar os recursos disponíveis em um SGBD para auxiliar no processo de indução de uma árvore de decisão e devido à não existência de um SGBD nativo para o formato dBase III, criou-se um banco de dados em PostgreSQL e os dados utilizados foram migrados para este novo banco, usando um programa conhecido como Datapump[9]. Obteve-se, após este processo de cópia de registros, uma réplica das tabelas

[9] Programa freeware que acompanha o ambiente gráfico de desenvolvimento Delphi 6, da Borland Software Corporation cuja finalidade é copiar tabelas de um banco de dados para outro.

originais (Figura 12) sem a criação de qualquer tipo de relacionamento entre elas.

Esta migração, além de permitir o uso de funcionalidades e facilidades disponibilizadas pela linguagem SQL para extrair os dados necessários para o processo de indução da árvore de decisão, permitiu testar e desenvolver as declarações SQL utilizadas pelo algoritmo de indução da árvore de decisão. Este desenvolvimento, testes e criação de tabelas auxiliares foi feito com o auxílio do PGAdmin[10], uma ferramenta gráfica de administração do PostgreSQL .

Após a migração para o PostgreSQL, os dados passaram por um processo de pré-processamento composto por duas fases.

5.2.1 Primeira fase – Inspeção dos dados

A primeira fase caracteriza-se por uma inspeção dos dados, avaliando os seus significados, suas características, sua qualidade, sua completeza, a existência de valores corrompidos e qual a melhor forma de disponibilizar os dados para o algoritmo de indução.

mo_expt1 character varying(12)	mo_tstiso character varying(20)	mo_padron2 character varying(31)
02,002,002,5	04000007500040000800	DRG.RPR
00,500,500,5	06000000500000000000	DAC.RRE
02,002,002,5	07000060000900005000	DRG.RPR
02,002,002,0	04000050000001002000	DIC.RRP
01,301,301,3	04000003500050001000	DCE.IIV.RPR
01,201,301,0	00680007800065000500	DCE.DLS.IRU.IUS.RSZ
00,000,000,0	01500025000500000150	DRG.RPR
01,501,501,5	10000001501000000700	DAC.RRE
01,001,002,0	05000100000500005000	DCE.RPR
03,003,003,0	04000040000125001000	DRG.RPR.RTP
00,000,000,0	05000002000020000005	DRG.RPR
01,001,001,0	05000050000150001000	DBR.RRB
00,500,500,5	10000050000600000000	DAC.RRE
06,004,004,5	00250001000015000150	DRG.IUS.RPR.RTP
01,801,502,0	00500001500250002000	DCE.RPR
02,002,002,0	50000500005000020000	DBR.IUS.RRB

Figura 13 - Amostra dos dados concatenados presentes na tabela b_fichmt

Nesta fase foram observados os seguintes aspectos:

1. O tamanho e a característica dos campos comuns entre as duas tabelas não eram compatíveis. Existiam alguns campos importantes como, por exemplo, o

[10] Uma das mais populares plataformas de administração e desenvolvimento para a plataforma PostgreSQL. Disponível para *download* em http://www.pgadmin.org/.

número de série do equipamento, que estavam definidos com tamanhos diferentes em cada uma das duas tabelas. Um dos campos, por exemplo, em uma das tabelas estava definido como um *varchar* de tamanho 11 e na outra como um *varchar* de tamanho 6. Neste e nos demais casos semelhantes, optou-se por usar a menor definição de tamanho desde que não houvesse perda de informação;

2. A tabela **b_fichmt** – que continha os dados oriundos da ficha do motor – possuía campos com conteúdos formados pela concatenação dos dados informados na tela de registro de fichas do sistema de gestão das oficinas. Com o uso da tela de consulta de fichas deste sistema de gestão, exibida no Anexo 1, foi possível identificar os dados concatenados, o seu significado e os seus formatos originais, como o número de dígitos e de casas decimais. Um exemplo destes dados é mostrado na Figura 13, nas colunas "mo_expt1" e "mo_tstiso", que representam respectivamente os valores coletados nos testes de excentricidade e isolação em diversos pontos do motor. Para resolver o problema, a solução encontrada foi decompor cada uma destas colunas com *n* valores concatenados, em *n* novas colunas;
3. De forma semelhante ao aspecto 2, os códigos de defeitos encontrados e dos reparos realizados estavam armazenados em um único campo separados por um ponto na coluna "mo_padron2", conforme pode ser visto na Figura 13. Levantado o domínio destes códigos chegou-se a um universo de 76 valores distintos, que combinados de seis em seis, resultam, de acordo com a análise combinatória, em um número de combinações dado por (8):

$$C_{n,k} = \frac{n!}{k!(n-k)!} \qquad (8)$$

Onde *k* é o número de elementos em cada subconjunto, no caso 6, e *n* é o número de elementos, no caso 76. Na prática este valor é maior, pois o campo pode conter de um a seis códigos combinados distintamente. Como o valor de combinações possíveis é muito alto, o uso destes dados em sua forma original tornaria inviável a sua utilização pelo algoritmo de indução, pois seria necessário implementar um número de testes equivalente ao número de combinações.

Para facilitar o trabalho do algoritmo de indução a solução encontrada foi criar um campo do tipo *boolean* para cada um dos elementos do domínio,

onde o valor '*true*' significa a existência do valor na coluna original e o valor '*false*' a inexistência. Os novos campos foram batizados com as três letras que identificam os códigos. Uma amostra da tabela resultante pode ser vista na figuraFigura 16;

4. Como os dados originais não estavam implementados dentro de uma estrutura de um banco de dados relacional e inexistia a figura da chave primária como forma de estabelecer um relacionamento entre as tabelas utilizadas, uma abordagem multirelacional semelhante à de (ATRAMENTOV; LEIVA; HONAVAR, 2003) ficou inviável. O caminho escolhido foi projetar uma única tabela ou *view* com todos os valores dos atributos previsores e os respectivos atributos alvos;
5. Existia uma grande quantidade de atributos cujos valores eram desconhecidos. Este fato pode ser conseqüência de dois fatores: o dado não foi digitado, fato pouco provável dentro do contexto de trabalho da oficina, ou o dado inexistia, pois a atividade realizada não exigiu o seu levantamento;
6. Alguns dados concatenados tinham problemas de formatação que puderam ser facilmente corrigidos durante a fase de tratamento dos dados. No entanto, também existiam dados corrompidos que não puderam ser recuperados. Neste último caso a tupla da qual faziam parte foi desconsiderada;
7. O tipo de entrada do motor em oficina – reparo ou preventiva - deve estar relacionado à tupla da última manutenção realizada, para que possa ser utilizado como atributo alvo. A explicação do porque desta observação será discutido adiante, ainda neste capítulo;
8. O tempo que o motor ficou em operação não estava disponível de forma direta.

Com base nestas observações e estando os dados necessários para a indução da árvore de decisão no novo banco de dados, o próximo passo a ser realizado foi corrigir e adequar os dados para que as declarações SQL pudessem ser aplicadas.

5.2.2 Segunda fase – Tratamento dos dados

Como mencionado no início deste capítulo, uma abordagem multirelacional se mostrou inviável devido à inexistência de uma forma simples de relacionar os registros. Este relacionamento é usualmente feito com o uso de chaves primária e estrangeira e a inexistência

deste mecanismo torna as declarações SQL muito complexas, devido à necessidade de se acrescentar uma série de condições adicionais para relacionar inequivocamente os registros.

O único campo que as duas tabelas tinham em comum era o número de série do motor, mas este aparecia mais de uma vez em cada tabela e sozinho não era capaz de relacionar as duas tabelas.

A solução para contornar este problema foi usar um critério de desempate usando os campos de data de liberação, presente em "b_fichmt", e o campo de data de envio para o almoxarifado, presente em "b_movequ". No entanto, como estas datas podem não ser as mesmas, já que uma vez liberado o motor pode esperar alguns dias para ser enviado, assumiu-se um intervalo de até 3 dias de atraso para fazer a comparação e estabelecer o relacionamento entre os registros. Este intervalo de três dias foi escolhido através de simulações que demonstraram que intervalos maiores não alteravam o número de registros relacionados de forma significativa.

Pensou-se inicialmente em criar uma *view* usando as funções padrão da linguagem SQL para fazer os ajustes de tamanhos de campos e separação dos campos concatenados. As primeiras experiências mostraram que tal opção poderia ser extremamente trabalhosa, devido à complexidade das declarações SQL.

Esta abordagem também se mostrou limitada, para a realização de algumas tarefas, como por exemplo:

- eliminar registros e dados inconsistentes;
- padronizar os tamanhos dos campos;
- padronizar e os tipos dos dados (*varchar*, *number* e *date*);
- corrigir os outros problemas encontrados.

Para a realização destas tarefas seria necessária a utilização de funções da linguagem procedural do banco de dados, o PL/pgSQL, o que limitaria a portabilidade da solução.

Estas limitações e dificuldades impuseram a necessidade de criação de uma nova tabela e o desenvolvimento de uma aplicação Java adicional para fazer todo o pré-processamento dos dados. Como resultado, obteve-se uma simplificação significativa das declarações SQL utilizadas pelo programa de indução da árvore de decisão e uma melhor qualidade dos dados.

O que se busca com esta tarefa de pré-processamento é um conjunto de exemplos S composto por k tuplas ordenadas cronologicamente $\{T_1, T_2, T_3, \ldots, T_k\}$, onde cada tupla T_i, com $0 < i \leq k$, possui atributos previsores $\{A_1, A_2, A_3, \ldots, A_j\}$, $j > 0$, com respectivos valores $\{a_{i1}, a_{i2}, \ldots, a_{kj}\}$ e atributo alvo C com valor c_i, $0 < i \leq k$. A representação deste conjunto é ilustrada na

Figura 14.

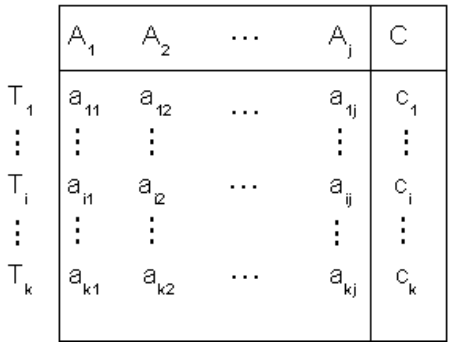


Figura 14 – Representação do conjunto de exemplos S.

Supõe-se neste trabalho que fatores como: as intervenções realizadas nos motores, as condições de funcionamento em que ele foi liberado e o tempo em que ficou em operação são importantes para prever a ocorrência da próxima falha, usando as regras obtidas com o uso de uma árvore de decisão. Ou seja, os atributos que representam estas informações podem indicar o tipo de entrada do motor na próxima vez que ele retornar à oficina, evidenciando quais os atributos que foram responsáveis por esta nova falha, se for o caso.

Quando um motor chega à oficina para reparo uma das informações que é registrada no sistema de gestão é justamente o tipo de entrada (C - corretiva ou P - preventiva), refletindo desta forma o motivo da remoção. Supondo-se por exemplo, que um motor tenha entrado para a realização de uma ação corretiva, os fatores que determinaram este tipo de entrada podem ter sido influenciados pelas condições operacionais que o equipamento apresentava quando foi liberado pela oficina, da última vez que sofreu um processo de manutenção, e também pelo do tempo em operação no trem.

Assim, dada uma tupla com os dados de manutenção de um determinado motor, para que a relação de causa e efeito possa ser estabelecida é preciso alterar o valor do tipo de entrada original para o valor que ele assumiu quando o equipamento retornou à oficina para uma nova manutenção.

Este processo pode ser melhor visualizado com o auxílio da Figura 15. Dado o conjunto de dados *S* como definido anteriormente e seja um motor de número de série *N*, que entrou para reparo em oficina *j* vezes e sejam também as tuplas T_n e T_m, com $j > m > n$, representando duas entradas quaisquer consecutivas deste motor nas datas $data_n$ e $data_m$, onde $data_n < data_m$. Para que a tupla T_n tenha o seu atributo alvo *C* representando o valor do futuro tipo de entrada do equipamento, este deve assumir o valor do atributo alvo para a tupla T_m, ou

seja, c_m. Na Figura 15(a) está representado o conjunto de dados antes do reposicionamento do atributo alvo de valor c_m da tupla T_m. Para que a relação de causa e efeito seja estabelecida o valor do atributo alvo C relacionado à tupla T_n deve ser substituído pelo valor do atributo alvo desta tupla T_m, ou seja, c_m. Após esta substituição a tupla T_n passa a ter um novo valor para o atributo alvo, conforme ilustrado na Figura 15(b). Procedendo-se desta forma consegue-se associar a cada tupla do conjunto de exemplos o tipo de entrada (atributo alvo) correto do ponto de vista necessário para a indução da árvore de decisão, estabelecendo-se a relação causa e efeito.

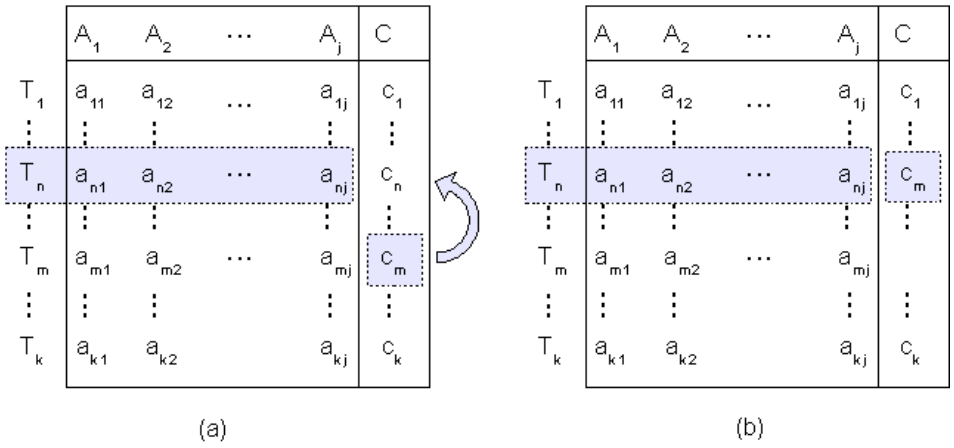


Figura 15 – Representação do conjunto de exemplos antes (a) e depois do acerto do valor do atributo alvo (b).

mo_expt1_1 numeric(4,0)	mo_expt1_2 numeric(4,0)	mo_expt1_3 numeric(4,0)	mo_tstiso_1 numeric(5,0)	mo_tstiso_2 numeric(5,0)	mo_tstiso_3 numeric(5,0)	mo_tstiso_4 numeric(5,0)	dgr boolean	rpr boolean
1	1	1	0	0	0	0	f	f
2	2	2	20000	4000	7500	1250	f	t
2	2	2	25000	5000	2000	10000	f	f
2	2	1	7500	850	1500	5000	f	f
3	3	3	5000	2000	500	100	f	t
2	2	2	7500	5000	5000	20	f	t
2	2	2	2500	1500	1500	5000	f	f
2	3	3	5000	2500	5000	75	f	f
1	1	1	2500	750	1500	5	f	f
2	2	2	1500	0	2600	200	f	t
2	2	2	10000	10000	10000	15	f	f
1	1	1	750	500	300	500	f	f
0	0	0	50000	50000	50000	50000	f	t
1	1	1	25000	22000	25000	25000	f	f
2	2	2	2500	4000	10000	3000	f	t
2	3	3	10000	750	2500	5000	f	t
2	2	2	25000	15000	25000	25000	f	t
2	2	2	5000	5000	10000	10000	f	f
3	3	3	10000	10000	10000	10000	f	f
1	1	1	10000	1000	5000	2000	f	t

Figura 16 - Amostra do conjunto de exemplos (tabela fichamotor) após o desmembramento dos campos com dados concatenados. A condição anterior destes dados pode ser visto na Figura 13.

Como mencionado na introdução deste trabalho, um dos fatores mais importantes para avaliar a confiabilidade de um equipamento é o tempo em que ele fica em operação sem apresentar falhas. No conjunto de dados analisado, este tempo não existia e as tentativas de obtê-lo em outros sistemas mostraram-se infrutíferas.

Para contornar a ausência deste dado optou-se pela utilização do intervalo, em dias, entre a data de liberação do motor e a data imediatamente posterior de sua nova entrada na oficina para manutenção deste mesmo motor.

O inconveniente desta opção é que o intervalo de tempo obtido representa, não só o tempo que o equipamento ficou efetivamente instalado, mas também o tempo que ele ficou parado no almoxarifado aguardando a instalação. Isto, apesar de não comprometer o resultado do processo de indução da árvore, deve ser levado em conta numa análise posterior do resultado com vistas à melhoria nos processo de manutenção.

O programa desenvolvido para o pré-processamento foi responsável por:

- inserir os registros na tabela de exemplos, a partir dos dados de outras tabelas;
- verificar e padronizar tipos diferentes de dados;
- separar as colunas que apresentam dados concatenados em novas colunas. Uma amostra destas novas colunas pode servir vista na Figura 16;
- excluir os registros cujos dados estejam corrompidos;
- calcular o intervalo entre duas datas para determinar o tempo que o equipamento ficou disponível para uso e armazenar este valor em uma nova coluna;
- fazer o acerto dos valores dos atributos alvos para que eles correspondam ao valor da futura entrada, excluindo as tuplas que ficaram sem valor para este atributo.

Finalizada a etapa de pré-processamento, obteve-se uma tabela com 2382 registros de um total inicial de 4043, devido à exclusão de registros com dados corrompidos ou cujo novo valor para o tipo de entrada não pôde ser determinado. Além disto, a quantidade de colunas saltou de 39 para 150 devido à expansão de “mo_padron2.

O *script* de criação desta nova tabela, chamada de 'fichamotor', é mostrado no Apêndice 1, enquanto o algoritmo da rotina de pré-processamento é mostrado no Apêndice 2.

5.3 Declarações SQL

O uso de declarações SQL para integrar o processo de mineração de dados, conforme

visto no capítulo 3, está baseado no fato de que os algoritmos não precisam dos dados em si para fazer o seu trabalho, mas apenas de uma medida da informação contida em cada conjunto de dados. Na maioria dos casos esta medida está relacionada ao número de vezes que um determinado atributo alvo aparece no conjunto pesquisado.

Dentre as cinco abordagens mencionadas no capítulo 4, optou-se por utilizar a abordagem que não estende a linguagem SQL, utilizando apenas a sintaxe padrão.

As declarações SQL foram responsáveis por fazer parte do trabalho do algoritmo. Este trabalho retirado do algoritmo corresponde às rotinas de agrupamento e levantamento das distribuições dos atributos no conjunto.

Ao algoritmo coube a tarefa de criar e solicitar a execução destas declarações ao SGBD e extrair os dados do conjunto retornado, no momento em que necessitar. Exemplos destas declarações SQL são mostrados a seguir.

```
select atributo_alvo as alvo,
       count(*) as distribuicao
from tabela_exemplos
group by alvo
```

(a)

```
select m1_tipoent as alvo,
       count(*) as distribuicao
from fichamotor
group by alvo
```

alvo character var	distribuicao bigint
P	1217
C	1165

(b)

Figura 17- Declarações SQL para determinar as freqüências dos valores dos atributos alvo na sua forma genérica (a) e um exemplo de aplicação com resultado obtido (b).

A declaração SQL exibida na Figura 17 foi gerada e utilizada pelo programa de indução toda vez que necessitou dos dados para calcular o valor da entropia de algum conjunto.

Já a declaração da Figura 18 foi gerada para determinar as freqüências dos valores dos atributos alvo para cada valor do atributo previsor, necessárias para o cálculo do *information gain* e do *gain ratio*, caso ele seja discreto. Na Figura 19 é apresentado um exemplo para o caso de atributos contínuos.

Quando da avaliação dos nós, o programa de indução utilizou a declaração SQL

ilustrada na Figura 20. Esta declaração representa a união de declarações menores para cada um dos testes realizados com os valores que o atributo previsor pode assumir.

```
select text 'atributo_ previsor  as atributo,
       atributo_previsor  as valor,
       atributo_alvo as alvo,
       count(*) as frequencia
from tabela_exemplos
where atributo_ previsor  in
       (select atributo_ previsor
        from  (select distinct atributo_ previsor ,
                               atributo_alvo,
                               count(*)
               from tabela_exemplos
               group by atributo_ previsor , atributo_alvo
               order by atributo_ previsor ) as temp
        group by atributo_ previsor
        having count(*) > 1
        order by atributo_ previsor )
group by valor, alvo
order by valor
```

(a)

```
select text 'rpr' as atributo,
       rpr as valor,
       m1_tipoent as alvo,
       count(*) as frequencia
from fichamotor
where rpr in
       (select rpr
        from  (select distinct rpr,
                               m1_tipoent,
                               count(*)
               from fichamotor
               group by rpr, m1_tipoent
               order by rpr) as temp
        group by rpr
        having count(*) > 1
        order by rpr)
group by valor, alvo
order by valor
```

atributo text	valor boolean	alvo character var	frequencia bigint
rpr	f	C	677
rpr	f	P	630
rpr	t	C	488
rpr	t	P	587

(b)

Figura 18- Declaração SQL para determinar as freqüências dos valores dos atributos alvo, para cada valor do atributo previsor (discreto), na sua forma genérica (a) e um exemplo de aplicação com resultado obtido (b).

```
select text 'atributo_ previsor' as atributo,
       atributo_previsor as valor,
       atributo_alvo as alvo,
       count(*) as frequencia
from tabela_exemplos
where atributo_ previsor in
       (select atributo_ previsor
        from (select distinct atributo_previsor ,
                              atributo_alvo ,
                             count(*)
              from fichamotor
              group by atributo_previsor , atributo_alvo
              order by atributo_previsor ) as temp
        group by atributo_ previsor
        having count(*) > 1
        order by atributo_ previsor )
group by valor, alvo
order by valor
```

(a)

```
select text 'm1_dispon' as atributo,
       m1_dispon as valor,
       m1_tipoent as alvo,
       count(*) as frequencia
from fichamotor
where m1_dispon in
       (select m1_dispon
        from (select distinct m1_dispon,
                              m1_tipoent,
                              count(*)
              from fichamotor
              group by m1_dispon, m1_tipoent
              order by m1_dispon) as temp
        group by m1_dispon
        having count(*) > 1
        order by m1_dispon)
group by valor, alvo
order by valor
```

atributo text	valor integer	alvo character var	frequencia bigint
m1_dispon	758	C	1
m1_dispon	758	P	1
m1_dispon	763	C	1
m1_dispon	763	P	3
m1_dispon	764	C	1
m1_dispon	764	P	1
m1_dispon	766	C	1
m1_dispon	766	P	1
m1_dispon	769	C	1
m1_dispon	769	P	1
m1_dispon	773	C	1
m1_dispon	773	P	2
m1_dispon	796	C	2
m1_dispon	796	P	1
m1_dispon	797	C	1
m1_dispon	797	P	2

(b)

Figura 19- Declaração SQL para determinar as freqüências dos valores dos atributos alvo, para cada valor do atributo previsor (contínuo) , na sua forma genérica (a) e um exemplo de aplicação com resultado obtido (b).

```
(select temp.previsor,
        temp.condicao,
        temp.alvo,
        count (*) as ocorrencias
from (select text 'atributo_previsor' as previsor,
             text '>=valor' as condicao,
             atributo_alvo as alvo
      from fichamotor where atributo_previsor >= valor) as temp
group by previsor, condicao, alvo
order by condicao, alvo)

union all

(select temp.previsor,
        temp.condicao,
        temp.alvo,
        count (*) as ocorrencias
from (select text 'atributo_previsor' as previsor,
             text '<valor' as condicao,
             atributo_alvo as alvo
      from fichamotor where atributo_previsor < valor) as temp
group by previsor, condicao, alvo
order by condicao, alvo)

union all

   . . .
```

(a)

```
(select temp.previsor,
        temp.condicao,
        temp.alvo,
        count (*) as ocorrencias
from (select text 'm1_dispon' as previsor,
             text '>=500' as condicao,
             m1_tipoent as alvo
      from fichamotor where m1_dispon >= '500') as temp
group by previsor, condicao, alvo
order by condicao, alvo)

union all

(select temp.previsor,
        temp.condicao,
        temp.alvo,
        count (*) as ocorrencias
from (select text 'm1_dispon' as previsor,
             text '<500' as condicao,
             m1_tipoent as alvo
      from fichamotor where m1_dispon < '500') as temp
group by previsor, condicao, alvo
order by condicao, alvo)
```

previsor text	condicao text	alvo character var	ocorrencias bigint
m1_dispon	>=500	C	783
m1_dispon	>=500	P	834
m1_dispon	<500	C	382
m1_dispon	<500	P	383

(b)

Figura 20- Declaração SQL para determinar as freqüências dos valores dos atributos alvo para cada teste do atributo previsor na sua forma genérica (a) e um exemplo de aplicação com resultado obtido (b).

5.4 Tabelas auxiliares

Além da tabela com o conjunto de exemplos, foram implementadas mais duas tabelas auxiliares, ‘metadados’ e ‘valteste’.

A tabela ‘metadados’ (Figura 21) continha os metadados de cada um dos atributos previsores e do atributo alvo. Os campos da tabela ‘metadados’ são:

- **atributo**: nome da coluna na tabela de exemplos (fichamotor) onde estão os valores do atributo;

- **tipo**: tipo de dado do atributo ((S) string, (D) double, (I) integer ou (B) boolean);
- **tratamento**: (D) discreto ou (C) contínuo;
- **categoria**: (P) previsor ou (A) alvo.

atributo character varying(20)	tipo character varying(1)	tratamento character(1)	categoria character(1)
m1_tipoent	C	D	A
m1_dispon	D	C	P
mo_etipt_1	D	C	P
mo_etipt_2	D	D	P
mo_etipt_3	D	C	P
mo_ofipt_1	D	C	P
mo_ofipt_2	D	C	P
mo_ofipt_3	D	C	P
mo_epxcar	C	D	P
mo_armpr_1	I	C	P
mo_armpr_2	I	C	P
mo_armpr_3	I	C	P

Figura 21- Exemplo do conteúdo da tabela ‘ metadados’.

A segunda tabela, ‘valtestes’ (Figura 22), continha os valores de teste que foram utilizados em conjunto com cada um dos atributos previsores. As colunas desta tabela são:

- **atributo**: nome do atributo;
- **valor**: valor de teste do atributo.

atributo character varying(20)	valor character varying(10)
m1_tipoent	P
m1_tipoent	C
m1_dispon	720
m1_dispon	1080
mo_etipt_1	5.0
mo_etipt_1	12.5
mo_etipt_2	5.1
mo_etipt_2	11.8
mo_etipt_3	6.7
mo_etipt_3	14.8

Figura 22- Exemplo do conteúdo da tabela ‘valtestes’.

Nesta tabela também estavam os valores do atributo alvo, ‘m1_tipoent’. Estes valores dependem do motivo de entrada na oficina e podem ser:

- ‘P’, para os casos em que os equipamentos entraram na oficina para preventiva;

- ‘C’, para os casos em que os equipamentos entraram para manutenção corretiva.

5.5 Tratamento dos atributos contínuos

Como já discutido, para que os atributos contínuos possam ser avaliados durante o processo de indução da árvore de decisão é necessária a discretização destes valores. No capítulo 3 foram discutidas diversas técnicas de discretização, ficando a escolha de qual ou quais delas adotar, em função da que melhor se ajusta às características dos dados.

Esta necessidade de análise de qual técnica de discretização melhor se ajusta às características dos dados, foi notada quando se tentou utilizar o método baseado no ganho (QUILAN, 1993), para discretizar os valores de tempo que os motores ficaram disponíveis para uso. Com o uso deste método tanto os valores obtidos para o *information gain* quanto os obtidos para o *gain ratio* não permitiram identificar claramente os valores de *threshold*. Isto exigiu a escolha de uma outra técnica de discretização.

O resultado desta análise prévia identificou basicamente dois comportamentos possíveis para a distribuição dos valores, que dependem essencialmente da natureza da grandeza representada.

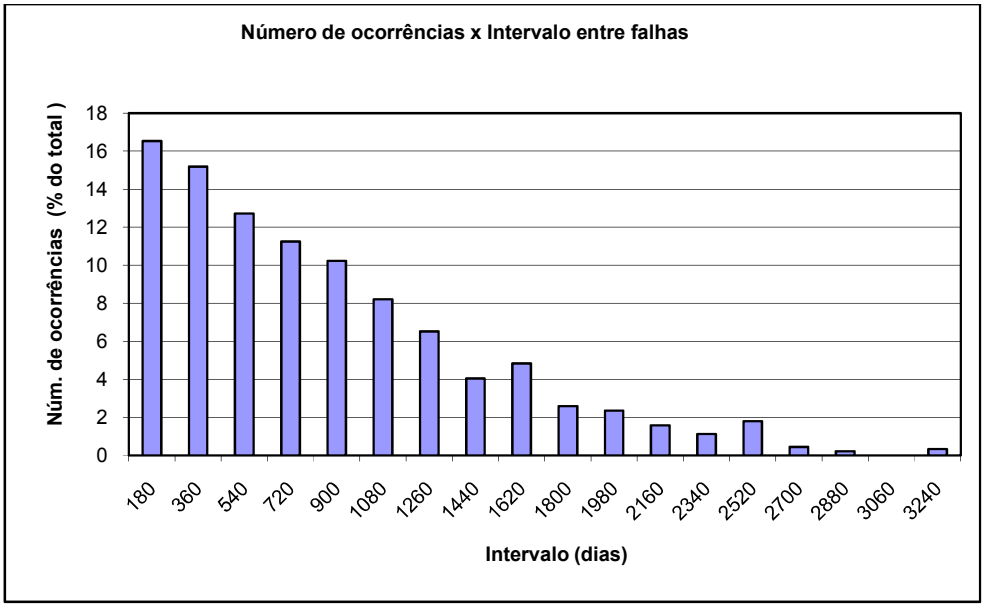


Figura 23 - Distribuição do número de entradas de motores para reparo em função do intervalo entre entradas.

O primeiro tipo de comportamento, ilustrado na Figura 23, é observado na distribuição dos valores de tempo que o motor ficou disponível para uso. Este padrão de comportamento

do número de falhas é condizente com o tipo de equipamento e com o tipo de manutenção realizada – complexa e invasiva (MOUBRAY, 2000, p. 247).

O histograma da figura Figura 23, cujo intervalo de amostra é de 180 dias, tem o comportamento que representa a região de "Mortalidade Infantil" de uma curva conhecida como "Curva Característica da Vida de Equipamentos" ou "Curva da Banheira" (Figura 24). Esta curva, freqüentemente está relacionada a um índice de falhas acentuado logo após a fabricação de um equipamento ou a um processo de manutenção mais complexo, decaindo à medida que o tempo passa (PINTO, XAVIER, 2001, p. 99). Com o passar do tempo ela pode apresentar novo crescimento quando o final da vida útil do equipamento se aproxima. Esta curva é bem conhecida nos meios produtivos e de manutenção por representar o ciclo de vida útil de equipamentos e sistemas.

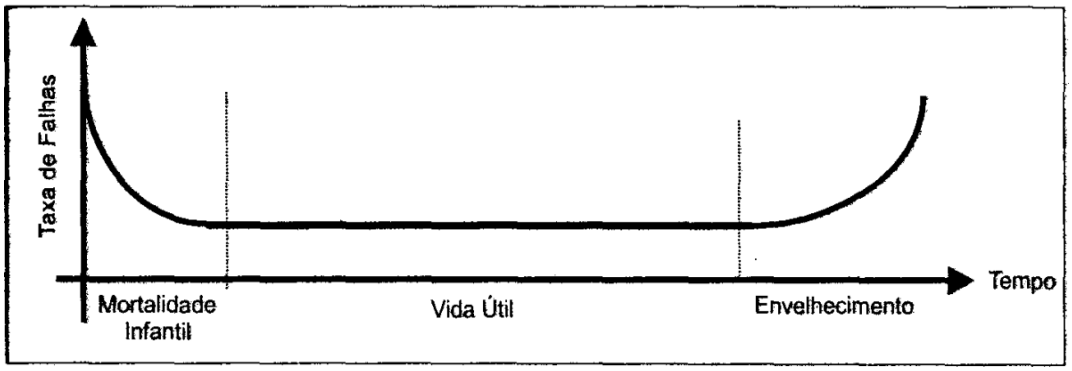


Figura 24 - Curva característica da vida de equipamentos (Curva da Banheira) (PINTO, XAVIER, 2001, p. 99).

Analisando o histograma para a distribuição do número de entradas em função do intervalo entre falhas (Figura 23), observa-se que nos primeiros 720 dias (2 anos) pouco mais de 50% dos motores liberados pela oficina retornam para novo reparo, enquanto que em 1080 dias (3 anos) quase 75% dos motores reparados retornaram.

Tanto Pinto; Xavier (2001) quanto Moubray (2000) destacam um ponto nesta curva, baseado na função da densidade de probabilidade de Weibull[11], onde ocorrem 63% das falhas. Tal situação corresponde ao ponto onde o tempo até a falha inicial ou vida mínima se iguala

[11] Desenvolvida pelo engenheiro sueco Hjalmar Waloddi Weibull, esta função "mostrou-se bastante adequada à análise de falha em equipamentos e, desde então, foi eleita como um das ferramentas para análise de confiabilidade. [...] é um método estatístico que correlaciona dados específicos de falha com uma distribuição particular, podendo indicar se a falha é um evento prematuro (mortalidade infantil), randômico (aleatório) ou ocasionado por desgaste (final de vida econômica)." (PINTO, XAVIER, 2001, p. 283).

ao tempo de vida característico do equipamento.

Baseado neste valor, foram escolhidos inicialmente como *threshold* os valores de 720 e 1080 dias que correspondem respectivamente a 50% e 75% dos retornos, por delimitarem a região onde o valor de 63% esta incluído.

O outro comportamento encontrado pode ser visto no gráfico da Figura 25, para o caso dos atributos relacionados às medidas de grandezas físicas. Neste exemplo, esta distribuição se refere aos valores obtidos para o parâmetro conhecido, pela equipe de oficina, como "isolação da bobina". Para os demais casos de atributos associados a grandezas físicas, o que variou foi a posição, a quantidade e a amplitude dos picos.

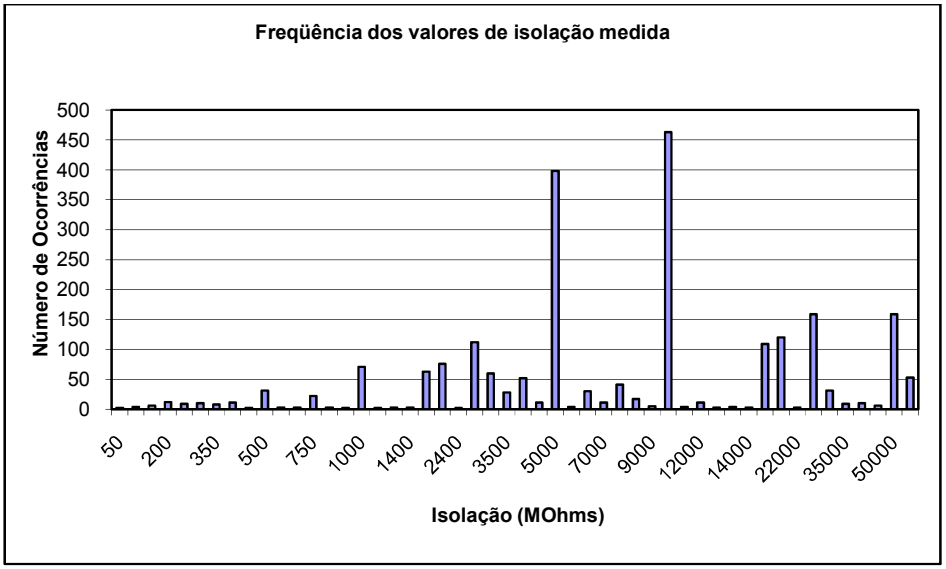


Figura 25 - Distribuição das freqüências dos valores medidos para a isolação da bobina do motor.

A abordagem utilizada para a discretização dos atributos contínuos, nestes casos foi similar a proposta por Perner; Trautzsch (1998) e se baseia na análise do histograma de distribuição do número de ocorrências dos valores presentes no domínio do atributo. Nesta proposta, a escolha do valor ou dos valores a serem utilizados para discretizar o conjunto de dados é feita baseada na identificação dos picos presentes na distribuição dos valores. Uma vez identificados os valores onde ocorrem os picos da distribuição, os valores de *threshold* do atributo podem ser escolhidos como o ponto médio entre dois picos adjacentes. No Apêndice 3 estão listados os valores de *threshold* determinados por este método.

Uma variação deste critério, é calcular o ganho de informação dos valores para os atributos previsores entre os dois picos e escolher aquele que apresenta o maior valor, como valor de *threshold*.

5.6 Tratamento dos atributos desconhecidos

Como as medições de algumas grandezas físicas só são realizadas quando o procedimento de manutenção utilizado na realização da atividade assim determinar ou, quando elas forem necessárias para auxiliar o diagnóstico da falha, a inexistência de valores para alguns atributos, é uma característica do critério de coleta destes valores. Assim, tentar preencher estes atributos com valores baseados nas técnicas descritas no capítulo 3 além de não resolver o problema, poderia criar outro mais sério, a inserção de valores irreais.

A solução adotada neste caso, foi considerar a inexistência deste valor como mais um valor a ser testado (o valor nulo), conforme a proposta de Frank; Witten (1996). Esta solução foi escolhida, inicialmente, devido à simplicidade e aos bons resultados obtidos por Bruha; Franek (1996), que consideraram esta solução a melhor dentre as que eles testaram.

Outra possibilidade, também discutida no item 3.3.3, é descartar as tuplas com atributos cujos valores sejam desconhecidos ou vazios. Esta técnica só foi utilizada quando a quantidade de dados descartados era muito inferior à quantidade de tuplas disponíveis.

5.7 Poda

O programa de indução da árvore de decisão utiliza como estratégia de poda, para evitar o *overfitting*, a pré-poda ou *pre-prunning*. Para que este processo de poda tenha efeito, para cada nó, no momento de uma nova expansão, é feita uma avaliação da razão entre freqüência de ocorrência dos atributos alvo para este novo teste. Quando esta razão, chamada de pureza, atingir o limite pré-estabelecido, os atributos previsores restantes para este nó não são mais avaliados, sendo o nó convertido em uma folha, cujo valor corresponde ao atributo alvo de maior valor para o nó. Por exemplo, sejam *freq*(*P)* e *freq*(*C)* os números de ocorrências para os possíveis valores dos atributos alvo *P* e *C* e seja o valor da pureza igual a 0,98. Quando uma das razões:

$$\frac{freq(P)}{freq(P)+freq(C)} \quad \text{, se freq(P) > freq(C)} \qquad (9)$$

ou,

$$\frac{freq(C)}{freq(P)+freq(C)} \quad \text{, se freq(P) < freq(C)} \qquad (10)$$

for maior ou igual à pureza (0,98), o crescimento da árvore é interrompido e nó é então convertido para um nó folha cujo valor representa o atributo de maior freqüência.

O *post-prunning* não é realizado pelo programa de indução e a sua implementação foi feita manualmente para tentar aumentar a capacidade previsão das regras obtidas.

5.8 Representação do resultado

Durante o processo de indução da árvore de decisão, os resultados gerados pela expansão dos nós em novos galhos foram armazenados em uma tabela que, ao fim do processo continha todos os nós e galhos gerados. Nesta tabela também foi armazenada a probabilidade de ocorrência do atributo alvo que apresenta maior freqüência de ocorrência em cada galho. Esta probabilidade é dada por:

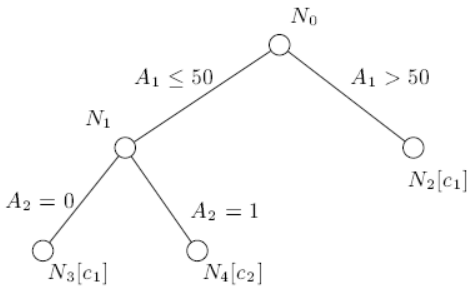
$$\text{———————} \quad \text{, se freq(P) > freq(C)} \qquad (11)$$

ou,

$$\text{———————} \quad \text{, se freq(P) < freq(C)} \qquad (12)$$

A representação do resultado foi feita de três formas. A primeira utilizando a tabela gerada pelo programa de indução, a segunda na forma de um conjunto de regas obtida a partir da tabela que representa a árvore de decisão e a terceira na forma de um diagrama semelhante ao apresentado nas Figura 26(a) e Figura 27(a).

A representação da árvore de decisão na forma de uma tabela tem uma aparência semelhante à desenvolvida por Sattler; Dunemann (2001), conforme ilustrado na Figura 26.

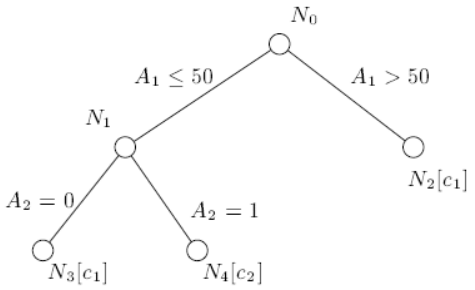


(a) *Árvore de decisão*

node	*parent*	*attrib*	*minval*	*maxval*	*class*	*prob*
N_0	$\perp$	$\perp$	$\perp$	$\perp$	c_1	0.70
N_1	N_0	A_1	MIN	50	c_1	0.86
N_2	N_0	A_1	50	MAX	c_2	0.66
N_3	N_1	A_2	-1	0	c_1	0.98
N_4	N_1	A_2	0	1	c_2	0.60

(b) *Representação na tabela*

Figura 26- Exemplo da representação da árvore de decisão na forma de uma tabela. Adaptada de Sattler; Dunemann (2001).



(a) *Árvore de decisão*

nó	*nó pai*	*atributo*	*condição*	*alvo*	*prob*
N_0	$\perp$	$\perp$		c_1	0.70
N_1	N_0	A_1	$A_1 \leq 50$	c_1	0.86
N_2	N_0	A_1	$A_1 > 50$	c_2	0.66
N_3	N_1	A_2	$A_2 = 0$	c_1	0.98
N_4	N_1	A_2	$A_2 = 1$	c_2	0.60

(b) *Representação na tabela*

Figura 27- Exemplo da representação proposta baseada no exemplo da Figura 26.

Neste trabalho a representação da árvore na forma de uma tabela difere um pouco da apresenta na Figura 26. Na representação utilizada aqui, os campos "minval" e "maxval" foram substituídos pelo campo "condição", por tornar mais clara a equivalência entre a representação na forma gráfica (Figura 26a) com a em forma de tabela (Figura 26b).

A representação em forma de um conjunto de regras é feita a partir dos resultados na forma de tabela, podendo ser obtida de duas formas:

- manualmente;
- ou de forma automática com a utilização de um programa adicional.

Neste trabalho foi utilizada a forma manual para uma avaliação preliminar dos resultados e a sua validade.

De forma semelhante, a representação da árvore de decisão na forma de um diagrama foi feita manualmente, com base na tabela gerada pelo programa de indução da árvore de decisão.

5.9 Geração da árvore de decisão

O programa de geração da árvore de decisão é baseado no algoritmo ilustrado no

Quadro 1 e, como discutido Capítulo 2, a implementação deste programa foi feita usando a linguagem Java, sendo o acesso aos dados realizado via JDBC - *Java Database Connectivity* -com uso de declarações SQL.

No programa desenvolvido todas as declarações SQL necessárias são geradas dinamicamente à medida que se tornam necessárias.

6. Resultados

Após a conclusão com sucesso das fases de pré-processamento e a obtenção do conjunto de dados, foi necessário definir os conjuntos de dados de treinamento e teste, implementar o processo de discretização dos atributos contínuos e avaliar a melhor forma de tratar os atributos com valores desconhecidos, antes de gerar as árvores de decisão e verificar o seus desempenho no conjunto de teste.

6.1 Geração dos conjuntos de treinamento e de teste

Conforme mencionado no capítulo 3, para a realização do processo de classificação usando árvores de decisão são necessários dois conjuntos de dados: um de treinamento, utilizado no processo de aprendizagem da árvore e um de teste para avaliar a precisão do conjunto de regras obtido no processo de aprendizagem. Desta forma, o conjunto original de dados disponível após a segunda fase da etapa de pré-processamento foi dividido em dois subgrupos: um de treinamento, com 20% dos dados, e outro de teste, com os 80% restantes.

Dentre as inúmeras formas de separação dos dados de treinamento foram analisadas as seguintes opções:

1. separar um subconjunto contíguo, iniciando em uma posição aleatória do conjunto inicial;
2. separar os dados aleatoriamente, até que o subconjunto de treinamento atinja o tamanho desejado;
3. separar o conjunto de treinamento a partir de tuplas distribuídas uniformemente ao longo do conjunto de dados.

A primeira opção, apesar de ser a mais fácil de implementar, gera um subconjunto que pode representar aspectos sazonais, já que as tuplas são dispostas em ordem cronológica. Caso algum aspecto sazonal esteja presente, este método de criação dos subconjuntos pode introduzir alguma distorção no resultado obtido após o processo de indução da árvore.

Apesar de não apresentar os problemas mencionados pela primeira opção, já que as tuplas utilizadas são colhidas aleatoriamente, para a implementação da segunda solução se faz necessária a criação de uma rotina, dentro do programa de pré-processamento, para realizar a seleção aleatória dos registros.

A terceira opção, retira amostras uniformemente distribuídas ao longo do tempo, garantindo a produção de um conjunto uniforme de dados dentro do período de análise

considerado, minimizando qualquer efeito sazonal porventura existente. Alem disto, esta opção não exige a criação de uma rotina adicional para a escolha aleatória das tuplas.

Com o uso da terceira opção, os conjuntos podem ser criados com o uso de duas declarações SQL, uma para gerar o subconjunto de treinamento e outra para o subconjunto de teste, que podem ou não ser executadas como parte do programa de pré-processamento.

Neste trabalho, optou-se pela terceira opção e pela execução das declarações SQL diretamente no banco de dados com o uso do PGAdmin.

As declarações utilizadas podem ser vistas nos Quadro 2 eQuadro 3. Nota-se que para o conjunto de treinamento foram utilizados os dados posicionados em tuplas múltiplas de cinco, enquanto que para o conjunto de teste foram utilizados os demais registros, garantindo a proporção de 20% dos dados para o conjunto de treinamento e 80% para o conjunto de teste.

```
insert into ficha_treino
select *
from fichamotor
where (nr_ficha % 5 = 0)
```

Quadro 2 - Declaração SQL utilizada para a geração do conjunto de treinamento. Nota-se que somente as tuplas do conjunto de dados (tabela 'fichamotor'), que possuem o campo 'nr_ficha' múltiplo de cinco são incorporadas ao conjunto de treinamento. O campo 'nr_ficha' contém uma seqüência numérica criada apenas para identificar univocamente cada registro.

```
insert into ficha_teste
select *
from fichamotor
where (nr_ficha % 5 != 0)
```

Quadro 3 - Declaração SQL utilizada para a geração do conjunto de teste. Nota-se que somente as tuplas do conjunto de dados (tabela 'fichamotor'), que possuem o campo 'nr_ficha' não múltiplo de cinco são incorporadas ao conjunto de teste. O campo 'nr_ficha' contém uma seqüência numérica criada apenas para identificar univocamente cada registro.

Apesar de terem sido usados neste trabalho apenas dois conjuntos de dados, um de treinamento e outro de teste, outra forma de análise e geração da árvore de decisão com mais de dois conjuntos é possível. Esta nova abordagem consiste em aumentar o número de subconjuntos e para cada um deles induzir uma árvore de decisão, que seria depois testada no conjunto de treinamento da outra e assim sucessivamente. Procedendo desta forma seriam obtidas tantas árvores de decisão quanto o número de subconjuntos. O resultado final seria

obtido a partir da busca de estruturas ou regras em comum.

6.2 Discretização dos atributos contínuos

O processo de discretização dos atributos contínuos, conforme mencionado no item 3.3.2, produziu o conjunto de valores de *threshold* que consta no Apêndice 3.

Observou-se que, durante o processo de discretização, para a maioria dos atributos discretizáveis, os gráficos gerados para o levantamento dos valores de *threshold* apresentavam picos bem definidos, facilitando a tarefa de discretização. Alguns poucos, no entanto, não apresentavam picos tão bem definidos, o que dificultou a determinação dos valores de *threshold*, sugerindo que para estes casos específicos a abordagem devesse ser diferente.

Apesar disso, os valores obtidos não foram utilizados, conforme discutido no próximo item.

6.3 Seleção dos atributos a serem testados

Dos cento e trinta e cinco atributos disponíveis, nem todos foram considerados atributos previsores a serem utilizados pelo processo de indução da árvore. A seleção dos atributos seguiu os seguintes critérios:

- os campos discretos representando ou não a presença de um determinado tipo de defeito não foram classificados como atributos previsores. Para esta categoria de atributos foram utilizados apenas aqueles que representam reparos ou intervenções realizadas, pois se supôs que as intervenções ou reparos realizados poderiam influenciar o surgimento de uma falha no equipamento;
- não foram classificados como previsores os campos auxiliares utilizados para o gerenciamento dos processos de manutenção;
- os campos contendo datas também não foram considerados para análise.

Além destes atributos, os com conteúdo contínuo também não foram selecionados como previsores. A justificativa para o não uso deste tipo de atributo se deveu ao fato de os valores medidos durante o processo de manutenção dos motores, estarem muito acima dos valores mínimos exigidos pelos procedimentos de manutenção ou pelas especificações dos fabricantes. Os valores encontrados nestes campos eram, em alguns casos, no mínimo 40 vezes melhores que o especificado, o que leva a crer que tenham pouca ou nenhuma influência como geradores de novas falhas.

A única exceção, dentre os atributos contínuos, foi o tempo de disponibilidade, pois,

conforme discutido no Capítulo 1, este atributo é um parâmetro muito importante na análise de confiabilidade de equipamentos.

6.4 Atributos com valores desconhecidos

Como grande parte dos valores desconhecidos estava concentrada nos atributos contínuos, a não utilização deste tipo de atributo dispensou o uso de um tratamento específico para contornar a ausência destes valores, conforme discutido nos itens 3.3.3 e 5.6.

Para os poucos casos onde os atributos apresentavam valores desconhecidos, a solução adotada foi desconsiderar a tupla. Este procedimento só foi aplicado pois o descarte destas tuplas causou pouco impacto na quantidade de dados disponíveis.

6.5 Árvore de decisão

Após definir o tratamento a ser dado aos atributos com valores nulos e quais os atributos que seriam analisados, o programa de indução foi executado utilizando os dados do conjunto de treinamento. Foram geradas quatro árvores de decisão, que serão analisadas a seguir.

A primeira árvore foi gerada utilizando um valor de 0,98 para o valor da pureza, conforme definido no item 5.7, e o resultado encontrado pode ser visto na Tabela 1.

Esta tabela é uma reprodução do resultado produzido pelo programa de indução na tabela ‘arvore’ do banco de dados. A coluna ‘Regra’ não faz parte do resultado gerado pelo programa de indução e foi adicionada apenas para facilitar a associação com a tabela que contém as regras (Tabela 2). O significado de cada uma das colunas é descrito a seguir:

- **nr_id**: número seqüencial de identificação do nó;
- **nó pai**: nome do atributo pai da sub-árvore;
- **condição**: condição de teste do atributo;
- **nó**: nome do atributo testado;
- **alvo**: valor do atributo alvo (nó folha);
- **prob**: razão entre a freqüência do atributo alvo e o número de exemplos analisado no nó folha;
- **regra**: número da regra associada à cada sub-árvore.

O programa de indução gerou também outra tabela denominada ‘regras’, onde estavam contidas as regras para cada um dos nós da árvore armazenada na tabela ‘arvore’. Estas regras foram expressas na forma de uma cadeia de caracteres, onde cada predicado (atributo +

condição) é concatenado com o próximo por meio uso do operador relacional "*and*". A sentença lógica obtida desta forma pode representar as condições necessárias para se obter o valor do atributo alvo especificado na coluna 'alvo'. A coluna 'nr_id' é o número da regra que corresponde ao valor da coluna 'regra' da tabela 'arvore'.

As regras contidas na Tabela 2 correspondem à árvore armazenada na Tabela 1.

nr_id	nó pai	condição	nó	condição	alvo	prob	freq1	freq2	Regra
1	null	null	itr	= t	C	1	2	0	1
2	null	null	itr	= f			232	242	
3	itr	= f	mo_armres	= 2			32	23	
4	mo_armres	= 2	ror	= t	P	1	1	0	2
5	mo_armres	= 2	ror	= f			32	22	
6	ror	= f	ius	= t	C	1	4	0	3
7	ror	= f	ius	= f			28	22	
8	ius	= f	rrb	= t	P	1	1	0	4
9	ius	= f	rrb	= f			28	21	
10	rrb	= f	rre	= t	C	1	2	0	5
11	rrb	= f	rre	= f			26	21	
12	rre	= f	rsz	= t	C	1	2	0	6
13	rre	= f	rsz	= f			24	21	
14	rsz	= f	mo_epxcar	= R	C	1	1	0	7
15	rsz	= f	mo_epxcar	= A			23	21	
16	mo_epxcar	= A	m1_dispon	<= 1080			12	16	
17	m1_dispon	<= 1080	rsc	= t	C	1	1	0	8
18	m1_dispon	<= 1080	rsc	= f			11	16	
19	rsc	= f	rtp	= t	P	1	3	0	9
20	rsc	= f	rtp	= f			11	13	
21	rtp	= f	mo_libcom	= O			10	13	
22	mo_libcom	= O	rrp	= t	P	1	1	0	10
23	mo_libcom	= O	rrp	= f			10	12	
24	rrp	= f	ivc	= t			7	5	
25	ivc	= t	mo_armban	= R			6	5	
26	mo_armban	= R	mo_armver	= A	C	1	2	0	11
27	mo_armban	= R	mo_armver	= R	P	0,555555556	4	5	12
28	ivc	= t	mo_armban	= A	C	1	1	0	13
29	rrp	= f	ivc	= f	P	0,7	3	7	14
30	rtp	= f	mo_libcom	= R	C	1	1	0	15
31	mo_epxcar	= A	m1_dispon	> 1080	C	0,6875	11	5	16
32	mo_epxcar	= A	m1_dispon	<= 720	P	0,6	8	12	17
33	mo_epxcar	= A	m1_dispon	> 720	C	0,625	15	9	18
34	itr	= f	mo_armres	= 1			48	26	
35	mo_armres	= 1	mo_armdes	= 1			47	26	
36	mo_armdes	= 1	rsb	= t	P	0,5	1	1	19
37	mo_armdes	= 1	rsb	= f	C	0,647887324	46	25	20
38	mo_armres	= 1	mo_armdes	= 2	C	1	1	0	21

Tabela 1- Representação gerada pelo programa de indução para a primeira árvore de decisão (tabela 'arvore'). A coluna 'Regra' foi adicionada apenas para facilitar a associação com a tabela que contem as regras (Tabela 2).

Com o uso destas regras na cláusula WHERE de uma consulta SQL é possível avaliar o desempenho das regras e conseqüentemente da árvore gerada, quando aplicadas aos conjuntos de treinamento e de teste. Esta avaliação procura verificar a capacidade de predição através do percentual de acertos do valor do atributo alvo.

Tomando-se como exemplo a regra 2 da Tabela 2, a declaração SQL para avaliar a precisão desta regra tem a forma apresentada no Quadro 4. Como resultado desta consulta, tem-se o valor do atributo alvo e da sua freqüência de ocorrência.

Num. Regra	alvo	Regra
1	C	itr = 't'
2	P	itr = 'f' and mo_armres = 2 and ror = 't'
3	C	itr = 'f' and mo_armres = 2 and ror = 'f' and ius = 't'
4	P	itr = 'f' and mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 't'
5	C	itr = 'f' and mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 'f' and rre = 't'
6	C	itr = 'f' and mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 'f' and rre = 'f' and rsz = 't'
7	C	itr = 'f' and mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 'f' and rre = 'f' and rsz = 'f' and mo_epxcar = 'R'
8	C	itr = 'f' and mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 'f' and rre = 'f' and rsz = 'f' and mo_epxcar = 'A' and m1_dispon <= 1080 and rsc = 't'
9	P	itr = 'f' and mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 'f' and rre = 'f' and rsz = 'f' and mo_epxcar = 'A' and m1_dispon <= 1080 and rsc = 'f' and rtp = 't'
10	P	itr = 'f' and mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 'f' and rre = 'f' and rsz = 'f' and mo_epxcar = 'A' and m1_dispon <= 1080 and rsc = 'f' and rtp = 'f' and mo_libcom = 'O' and rrp = 't'
11	C	itr = 'f' and mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 'f' and rre = 'f' and rsz = 'f' and mo_epxcar = 'A' and m1_dispon <= 1080 and rsc = 'f' and rtp = 'f' and mo_libcom = 'O' and rrp = 'f' and ivc = 't' and mo_armban = 'R' and mo_armver = 'A'
12	P	itr = 'f' and mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 'f' and rre = 'f' and rsz = 'f' and mo_epxcar = 'A' and m1_dispon <= 1080 and rsc = 'f' and rtp = 'f' and mo_libcom = 'O' and rrp = 'f' and ivc = 't' and mo_armban = 'R' and mo_armver = 'R'
13	C	itr = 'f' and mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 'f' and rre = 'f' and rsz = 'f' and mo_epxcar = 'A' and m1_dispon <= 1080 and rsc = 'f' and rtp = 'f' and mo_libcom = 'O' and rrp = 'f' and ivc = 't' and mo_armban = 'A'
14	P	itr = 'f' and mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 'f' and rre = 'f' and rsz = 'f' and mo_epxcar = 'A' and m1_dispon <= 1080 and rsc = 'f' and rtp = 'f' and mo_libcom = 'O' and rrp = 'f' and ivc = 'f'
15	C	itr = 'f' and mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 'f' and rre = 'f' and rsz = 'f' and mo_epxcar = 'A' and m1_dispon <= 1080 and rsc = 'f' and rtp = 'f' and mo_libcom = 'R'
16	C	itr = 'f' and mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 'f' and rre = 'f' and rsz = 'f' and mo_epxcar = 'A' and m1_dispon > 1080
17	P	itr = 'f' and mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 'f' and rre = 'f' and rsz = 'f' and mo_epxcar = 'A' and m1_dispon <= 720
18	C	itr = 'f' and mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 'f' and rre = 'f' and rsz = 'f' and mo_epxcar = 'A' and m1_dispon > 720
19	P	itr = 'f' and mo_armres = 1 and mo_armdes = 1 and rsb = 't'
20	C	itr = 'f' and mo_armres = 1 and mo_armdes = 1 and rsb = 'f'
21	C	itr = 'f' and mo_armres = 1 and mo_armdes = 2

Tabela 2- Conjunto de regras geradas pelo programa de indução. Estas regras estão associadas à arvore contida na Tabela 1.

Forma geral:

```
select atributo_alvo, count(*)
from tabela
where regra
group by atributo_alvo
order by atributo_alvo
```

Exemplo usando a regra 2:

```
select m1_tipoent, count(*)
from ficha_teste
where itr = 'f' and mo_armres = 2 and ror = 't'
group by m1_tipoent
order by m1_tipoent
```

Quadro 4 – Exemplo da declaração SQL utilizada para avaliar a precisão da regra em sua forma geral e um exemplo usando a regra de número 2 (em destaque).

Os resultados obtidos nesta avaliação para os conjuntos de treinamento e de teste estão resumidos nas tabelasTabela 3 eTabela 4, respectivamente, cuja descrição das colunas é feita a seguir:

- **C**: número de ocorrências do valor ‘C’ para o atributo alvo obtidos com o uso da regra;
- **P**: número de ocorrências do valor ‘P’ para o atributo alvo obtidos com o uso da regra;
- **acertos**: quantidade de valores para o atributo alvo preditos corretamente pela árvore;
- **erros**: quantidade de valores para o atributo alvo preditos erroneamente pela regra;
- **% ERRO**: percentual de valores preditos erroneamente dentre o total de casos analisados para cada regra;
- **% ACERTO**: percentual de valores preditos corretamente dentre o total de casos analisados para cada regra;

A última linha das tabelasTabela 3 eTabela 4 totaliza os valores encontrados e apresenta o percentual total estimado de erros e de acertos para a árvore de decisão gerada. Como esperado o desempenho das regras para o conjunto de treinamento foi melhor que o obtido para o conjunto de teste devido ao *overfitting*. O nível do *overfitting* é determinado pelo valor da pureza (item 5.7), que nesta primeira árvore, foi estipulado em 0,98.

Num. Regra	Alvo	REGRAS APLICADAS AO CONJUNTO DE TREINAMENTO					
		C	P	acertos	erros	% ERRO	% ACERTO
1	C	2	0	2	0	0,00	100,00
2	P	1	0	1	0	0,00	100,00
3	C	4	0	4	0	0,00	100,00
4	P	1	0	1	0	0,00	100,00
5	C	2	0	2	0	0,00	100,00
6	C	2	0	2	0	0,00	100,00
7	C	1	0	1	0	0,00	100,00
8	C	1	0	1	0	0,00	100,00
9	P	3	0	3	0	0,00	100,00
10	P	1	0	1	0	0,00	100,00
11	C	2	0	2	0	0,00	100,00
12	P	0	0	5	4	44,44	55,56
13	C	1	0	1	0	0,00	100,00
14	P	0	0	7	3	30,00	70,00
15	C	1	0	1	0	0,00	100,00
16	C	0	0	11	5	31,25	68,75
17	P	12	8	12	8	40,00	60,00
18	C	15	9	15	9	37,50	62,50
19	P	1	1	1	1	50,00	50,00
20	C	46	25	46	25	35,21	64,79
21	C	1	0	1	0	0,00	100,00
Totais		97	43	120	55	31,43	68,57

Tabela 3 - Resultado da avaliação das regras aplicadas ao conjunto de treinamento.

Num. Regra	Alvo	REGRAS APLICADAS AO CONJUNTO DE TESTE					
		C	P	acertos	erros	% ERRO	% ACERTO
1	C	2	0	2	0	0,00	100,00
2	P	0	1	1	0	0,00	100,00
3	C	8	5	8	5	38,46	61,54
4	P	5	0	0	5	100,00	0,00
5	C	6	1	6	1	14,29	85,71
6	C	6	0	6	0	0,00	100,00
7	C	2	3	2	3	60,00	40,00
8	C	2	0	2	0	0,00	100,00
9	P	3	3	3	3	50,00	50,00
10	P	3	2	2	3	60,00	40,00
11	C	4	1	4	1	20,00	80,00
12	P	12	9	9	12	57,14	42,86
13	C	0	0	0	0	0,00	0,00
14	P	17	18	18	17	48,57	51,43
15	C	3	1	3	1	25,00	75,00
16	C	39	11	39	11	22,00	78,00
17	P	28	27	27	28	50,91	49,09
18	C	58	18	58	18	23,68	76,32
19	P	2	1	1	2	66,67	33,33
20	C	202	100	202	100	33,11	66,89
21	C	0	1	0	1	100,00	0,00
Totais		402	202	393	211	34,93	65,07

Tabela 4 - Resultado da avaliação das regras aplicadas ao conjunto de teste.

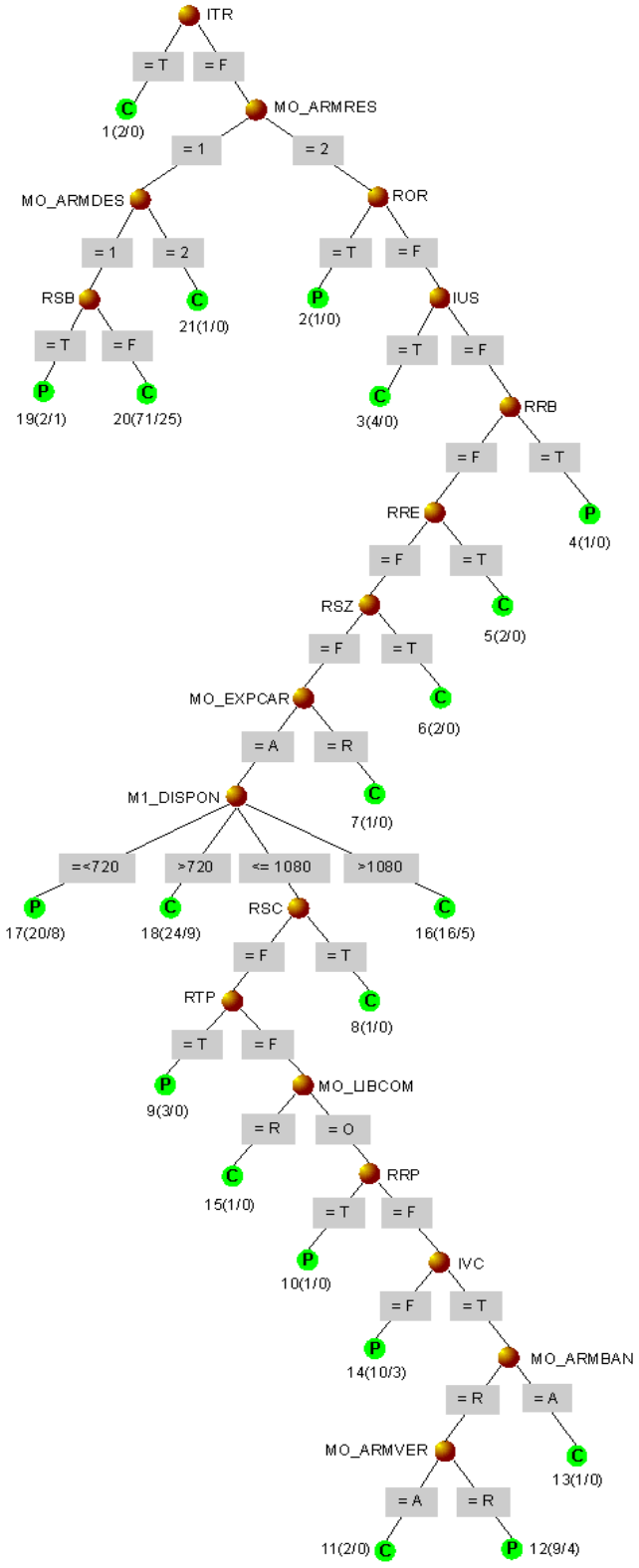


Figura 28- Árvore de decisão obtida a partir da Tabela 1. Os números próximos aos nós folha, na forma R(N/E), representam o número da regra (R) que gera o nó folha, o número de casos avaliados (N) e o número de casos avaliados erroneamente.

Com base nas tabelasTabela 1 eTabela 3 foi possível montar o diagrama que representa a árvore induzida e que pode ser visto na Figura 28. Neste diagrama estão representados graficamente os atributos avaliados ou nós, os ramos com as condições de teste que eles representam e os nós folha ou atributos alvo. Além disto, foi adicionado junto a cada nó folha, um valor na forma *R(N/E)*, onde *R* representa o número da regra, *N* a quantidade total de casos classificados no nó folha e *E* o número de casos classificados erroneamente.

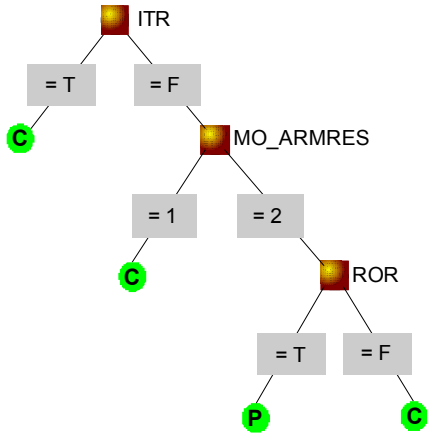


Figura 29 - Árvore de decisão após o *post-prunning*.

Num. Regra	Alvo	Regra
1	C	itr = 't'
2	P	itr = 'f' and mo_armres = 2 and ror = 't'
3	C	itr = 'f' and mo_armres = 2 and ror = 'f'
4	C	itr = 'f' and mo_armres = 1

Tabela 5 - Conjunto de regras após o processo de *post-prunning*.

Num. Regra	Alvo	REGRAS APLICADAS AO CONJUNTO DE TREINAMENTO						REGRAS APLICADAS AO CONJUNTO DE TESTE					
		C	P	acertos	erros	% ERRO	% ACERTO	C	P	acertos	erros	% ERRO	% ACERTO
1	C	2	0	2	0	0,00	100,00	2	0	2	0	0,00	100,00
2	P	0	1	1	0	0,00	100,00	2	1	1	2	66,67	33,33
3	C	32	22	32	22	40,74	59,26	110	54	110	54	32,93	67,07
4	C	48	26	48	26	35,14	64,86	205	102	205	102	33,22	66,78
		82	49	83	48	36,64	63,36	319	157	318	158	33,19	66,81

Tabela 6 - Avaliação do novo conjunto de regras

Sub-conjunto de dados	Antes do *post-prunning*		Depois do *post-prunning*	
	% ERRO	% ACERTO	% ERRO	% ACERTO
Treinamento	31,43	68,57	36,64	63,36
Teste	34,93	65,07	33,19	66,81

Tabela 7- Comparação do desempenho da árvore antes e depois do processo de *post-prunning* para os conjuntos de treinamento e de teste

Foram geradas também árvores com valores de pureza iguais a 0,7 e 0,8 para avaliar a precisão da predição usando diferentes valores de *pre-prunning*. Os resultados obtidos para estes valores de pureza geraram árvores idênticas à obtida usando o valor de pureza igual a 0,98, para o mesmo conjunto de treinamento. As tabelas 'arvore' para estes casos podem ser vistas nos Apêndice 5 eApêndice 6 respectivamente.

nr_id	nó pai	condição	nó	condição	alvo	prob	freq1	freq2	Regra
1	null	null	itr	= t	C	1	2	0	1
2	null	null	itr	= f		0,510548523	232	242	
3	itr	= f	mo_armres	= 2		0,581818182	32	23	
4	mo_armres	= 2	ror	= t	P	1	1	0	2
5	mo_armres	= 2	ror	= f		0,592592593	32	22	
6	ror	= f	ius	= t	C	1	4	0	3
7	ror	= f	ius	= f		0,56	28	22	
8	ius	= f	rrb	= t	P	1	1	0	4
9	ius	= f	rrb	= f		0,571428571	28	21	
10	rrb	= f	rre	= t	C	1	2	0	5
11	rrb	= f	rre	= f		0,553191489	26	21	
12	rre	= f	rsz	= t	C	1	2	0	6
13	rre	= f	rsz	= f		0,533333333	24	21	
14	rsz	= f	mo_epxcar	= R	C	1	1	0	7
15	rsz	= f	mo_epxcar	= A		0,522727273	23	21	
16	mo_epxcar	= A	m1_dispon	<= 1080		0,571428571	12	16	
17	m1_dispon	<= 1080	rsc	= t	C	1	1	0	8
18	m1_dispon	<= 1080	rsc	= f		0,592592593	11	16	
19	rsc	= f	rtp	= t	P	1	3	0	9
20	rsc	= f	rtp	= f		0,541666667	11	13	
21	rtp	= f	mo_libcom	= O		0,565217391	10	13	
22	mo_libcom	= O	rrp	= t	P	1	1	0	10
23	mo_libcom	= O	rrp	= f		0,545454545	10	12	
24	rrp	= f	ivc	= t		0,583333333	7	5	
25	ivc	= t	mo_armban	= R		0,545454545	6	5	
26	mo_armban	= R	mo_armver	= A	C	1	2	0	11
27	mo_armban	= R	mo_armver	= R	P	0,555555556	4	5	12
28	ivc	= t	mo_armban	= A	C	1	1	0	13
29	rrp	= f	ivc	= f	P	0,7	3	7	14
30	rtp	= f	mo_libcom	= R	C	1	1	0	15
31	mo_epxcar	= A	m1_dispon	> 1080	C	0,6875	11	5	16
32	mo_epxcar	= A	m1_dispon	<= 720	P	0,6	8	12	17
33	mo_epxcar	= A	m1_dispon	> 720	C	0,625	15	9	18
34	itr	= f	mo_armres	= 1		0,648648649	48	26	
35	mo_armres	= 1	mo_armdes	= 1		0,643835616	47	26	
36	mo_armdes	= 1	rsb	= t	P	0,5	1	1	19
37	mo_armdes	= 1	rsb	= f	C	0,647887324	46	25	20
38	mo_armres	= 1	mo_armdes	= 2	C	1	1	0	21

Tabela 8 - Os valores de probabilidade para os nós intermediários omitidos na Tabela 1 são apresentados para explicar a não ocorrência do *pre-prunning*. Foram destacados os valores menores que 1.

O resultado desta avaliação mostrou que o processo de *pre-prunning* (itens 3.3.4 e 5.7) foi, neste caso, ineficaz para melhorar a precisão das regras obtidas quando aplicadas ao conjunto de teste, já que a diminuição do valor da pureza equivale a aumentar o nível do *pre-prunning*, confome discutido no item 5.7. O valor mínimo da pureza foi limitado a 0,7 pois se esperava da árvore uma precisão da ordem de 70%.

Esta ineficácia do processo de *pre-prunning* pode ser entendida com o uso da Tabela 8. Nesta tabela estão exibidos todos os valores de probabilidade que foram omitidos na Tabela 1, para os nós que não representavam folhas. Ao observar os valores para estes nós intermediários, nota-se que nenhum deles é superior ao menor valor de pureza utilizado. Isto significa que, para que a poda aconteça, o valor utilizado para a pureza deverá ser inferior a 0,7.

Esta limitação do processo de *pre-prunning* pode ser melhor entendida quando se toma como exemplo os nós que correspondem à regra de número 2, (as linhas correspondentes a 'nr_id' igual a 2, 3 e 4), na Tabela 8. Na linha correspondente a 'nr_id' igual a 2, tem-se o valor de probabilidade igual a 0,51, ou seja,

$$\frac{242}{(232+242)} \qquad (13)$$

Portanto, quando este nó é testado para definir se ele deve ser expandido ou não em uma nova subárvore, o resultado do teste indica que a expansão do nó deve ser realizada e novas subárvores poderão ser geradas. O mesmo acontece para a linha correspondente a 'nr_id' igual a 3, cujo valor de probabilidade é aproximadamente igual a 0,58.

Para a linha correspondente a 'nr_id' igual a 4 o valor da probabilidade assume o valor 1 e o nó é transformado em nó folha, já que não é mais possível continuar a particionar o subconjunto.

O que acontece neste exemplo se repete até o final da tabela, com valores de probabilidade sempre maiores ou iguais a 0,7.

Diante destes resultados insatisfatórios, a saída encontrada para aumentar a capacidade de generalização da árvore de decisão foi usar o resultado com maior grau de *overfitting* – a árvore com valor de pureza igual a 0,98 – e aplicar técnicas de *post-prunning*. Foi escolhida a árvore de elevado valor de *overfitting* pois, nesta abordagem, esta é uma característica desejada, conforme discutido no item 3.3.4.

Para o processo de *post-prunning* utilizou-se uma versão simplificada da técnica conhecida por *Reduced-error Prunning* descrita por Quinlan (1993) e mencionada no item 3.3.4.

Nesta versão simplificada, iniciou-se o processo de poda pelos galhos/folhas mais externas, assumindo-se que os ramos cujos nós folha apresentem erro igual ou superior a 35% serão podados e o nó correspondente substituído pelo valor do atributo alvo com menor erro de predição. A escolha deste valor de erro foi feita com base no erro total da árvore, quando aplicada ao conjunto de teste (conforme a Tabela 4), por ser a pior estimativa obtida. Este procedimento de poda foi realizado usando a árvore obtida para o conjunto de dados de treinamento, assim como realizado por Quinlan (1993).

O resultado obtido foi uma árvore bastante simplificada (Figura 29) com poucos nós e regras mais simples (Tabela 5). O desempenho desta nova árvore pode ser visualizado na Tabela 6. Quando este resultado é comparado com os obtidos para a árvore original nota-se um ligeiro aumento na capacidade de predição quando aplicada ao conjunto de dados de teste, e uma conseqüente diminuição do nível de acertos quando aplicada ao conjunto de treinamento (Tabela 7).

A quarta árvore de decisão gerada pelo programa de indução serviu para avaliar como a retirada de um dos atributos previsores impacta o resultado final. Não existiu aqui a intenção de fazer uma análise mais aprofundada dos impactos da remoção ou inclusão de atributos previsores, mas apenas ter uma amostra dos efeitos deste tipo de procedimento que pode ser realizado na etapa de pós-processamento, por um especialista no domínio do problema analisado.

Dentre os atributos existentes foi escolhido o de nome ITR, por ser o nó raiz das árvores anteriores. Foram mantidos todos os demais parâmetros de configuração utilizados na primeira árvore induzida. O resultado para esta nova árvore está representado na Tabela 9. Comparando este resultado com o anterior (Tabela 1) percebe-se que as diferenças ocorrem no início, devido a retirada do atributo ITR e no final da árvore como conseqüência das alterações nos valores das freqüências de ocorrência dos valores do atributo alvo. O novo conjunto de regras produzido nesta nova condição pode ser visto na Tabela 10.

As Tabela 11 eTabela 12 mostram o resultado da avaliação das regras feita com o uso dos conjuntos de treinamento e de teste, respectivamente. Nota-se que os resultados obtidos são muito próximos dos resultados anteriores, no tocante ao percentual de erros e acertos.

O diagrama representando esta nova árvore pode ser visto na Figura 30. Neste diagrama fica mais fácil perceber as modificações na conformação da árvore com a retirada

do atributo que estava anteriormente na raiz (ITR).

nr_id	nó pai	condição	nó	condição	alvo	prob	freq1	freq2	Regra
1	null	null	mo_armres	= 2			32	23	
2	mo_armres	= 2	ror	= t	P	1	1	0	1
3	mo_armres	= 2	ror	= f			32	22	
4	ror	= f	ius	= t	C	1	4	0	2
5	ror	= f	ius	= f			28	22	
6	ius	= f	rrb	= t	P	1	1	0	3
7	ius	= f	rrb	= f			28	21	
8	rrb	= f	rre	= t	C	1	2	0	4
9	rrb	= f	rre	= f			26	21	
10	rre	= f	rsz	= t	C	1	2	0	5
11	rre	= f	rsz	= f			24	21	
12	rsz	= f	mo_epxcar	= R	C	1	1	0	6
13	rsz	= f	mo_epxcar	= A			23	21	
14	mo_epxcar	= A	m1_dispon	<= 1080			12	16	
15	m1_dispon	<= 1080	rsc	= t	C	1	1	0	7
16	m1_dispon	<= 1080	rsc	= f			11	16	
17	rsc	= f	rtp	= t	P	1	3	0	8
18	rsc	= f	rtp	= f			11	13	
19	rtp	= f	mo_libcom	= O			10	13	
20	mo_libcom	= O	rrp	= t	P	1	1	0	9
21	mo_libcom	= O	rrp	= f			10	12	
22	rrp	= f	ivc	= f			3	7	
23	ivc	= f	mo_armver	= A			3	6	
24	mo_armver	= A	mo_armban	= R			2	3	
25	mo_armban	= R	rmv	= t	C	1	1	0	10
26	mo_armban	= R	rmv	= f	P	0,75	1	3	11
27	mo_armver	= A	mo_armban	= A	P	0,75	1	3	12
28	ivc	= f	mo_armver	= R	P	1	1	0	13
29	rrp	= f	ivc	= t			7	5	
30	ivc	= t	iva	= t	P	0,571428571	3	4	14
31	ivc	= t	iva	= f	C	0,8	4	1	15
32	rtp	= f	mo_libcom	= R	C	1	1	0	16
33	mo_epxcar	= A	m1_dispon	> 1080	C	0,6875	11	5	17
35	mo_epxcar	= A	m1_dispon	<= 720	P	0,6	8	12	18
36	mo_epxcar	= A	m1_dispon	> 720	C	0,625	15	9	19
37	null	null	mo_armres	= 1			49	26	
38	mo_armres	= 1	mo_armdes	= 2	C	1	1	0	20
39	mo_armres	= 1	mo_armdes	= 1			48	26	
40	mo_armdes	= 1	rsb	= t	P	0,5	1	1	21
41	mo_armdes	= 1	rsb	= f	C	0,652777778	47	25	22

Tabela 9 -Representação gerada pelo programa de indução para a árvore de decisão sem o atributo ITR (troca de rolamento). A coluna ‘regra’ foi adicionada apenas para facilitar a associação com a tabela que contem as regras (Tabela 10).

nr_id	alvo	Regra
1	P	mo_armres = 2 and ror = 't'
2	C	mo_armres = 2 and ror = 'f' and ius = 't'
3	P	mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 't'
4	C	mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 'f' and rre = 't'
5	C	mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 'f' and rre = 'f' and rsz = 't'
6	C	mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 'f' and rre = 'f' and rsz = 'f' and mo_epxcar = 'R'
7	C	mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 'f' and rre = 'f' and rsz = 'f' and mo_epxcar = 'A' and m1_dispon <= 1080 and rsc = 't'
8	P	mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 'f' and rre = 'f' and rsz = 'f' and mo_epxcar = 'A' and m1_dispon <= 1080 and rsc = 'f' and rtp = 't'
9	P	mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 'f' and rre = 'f' and rsz = 'f' and mo_epxcar = 'A' and m1_dispon <= 1080 and rsc = 'f' and rtp = 'f' and mo_libcom = 'O' and rrp = 't'
10	C	mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 'f' and rre = 'f' and rsz = 'f' and mo_epxcar = 'A' and m1_dispon <= 1080 and rsc = 'f' and rtp = 'f' and mo_libcom = 'O' and rrp = 'f' and ivc = 'f' and mo_armver = 'A' and mo_armban = 'R' and rmv = 't'
11	P	mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 'f' and rre = 'f' and rsz = 'f' and mo_epxcar = 'A' and m1_dispon <= 1080 and rsc = 'f' and rtp = 'f' and mo_libcom = 'O' and rrp = 'f' and ivc = 'f' and mo_armver = 'A' and mo_armban = 'R' and rmv = 'f'
12	P	mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 'f' and rre = 'f' and rsz = 'f' and mo_epxcar = 'A' and m1_dispon <= 1080 and rsc = 'f' and rtp = 'f' and mo_libcom = 'O' and rrp = 'f' and ivc = 'f' and mo_armver = 'A' and mo_armban = 'A'
13	P	mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 'f' and rre = 'f' and rsz = 'f' and mo_epxcar = 'A' and m1_dispon <= 1080 and rsc = 'f' and rtp = 'f' and mo_libcom = 'O' and rrp = 'f' and ivc = 'f' and mo_armver = 'R'
14	P	mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 'f' and rre = 'f' and rsz = 'f' and mo_epxcar = 'A' and m1_dispon <= 1080 and rsc = 'f' and rtp = 'f' and mo_libcom = 'O' and rrp = 'f' and ivc = 'f' and ivc = 't' and iva = 't'
15	C	mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 'f' and rre = 'f' and rsz = 'f' and mo_epxcar = 'A' and m1_dispon <= 1080 and rsc = 'f' and rtp = 'f' and mo_libcom = 'O' and rrp = 'f' and ivc = 'f' and ivc = 't' and iva = 'f'
16	C	mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 'f' and rre = 'f' and rsz = 'f' and mo_epxcar = 'A' and m1_dispon <= 1080 and rsc = 'f' and rtp = 'f' and mo_libcom = 'R'
17	C	mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 'f' and rre = 'f' and rsz = 'f' and mo_epxcar = 'A' and m1_dispon > 1080
18	P	mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 'f' and rre = 'f' and rsz = 'f' and mo_epxcar = 'A' and m1_dispon <= 720
19	C	mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 'f' and rre = 'f' and rsz = 'f' and mo_epxcar = 'A' and m1_dispon > 720
20	C	mo_armres = 1 and ror = 'f' and mo_armres = 2
21	P	mo_armres = 1 and mo_armdes = 1 and rsb = 't'
22	C	mo_armres = 1 and mo_armdes = 1 and rsb = 'f'

Tabela 10 - Conjunto de regras geradas pelo programa de indução sem o atributo ITR. Estas regras estão associadas à arvore contida na Tabela 9.

Num. Regra	alvo	REGRAS APLICADAS AO CONJUNTO DE TREINAMENTO					
		C	P	acertos	erros	% ERRO	% ACERTO
1	P	1	0	0	1	100,00	0,00
2	C	4	0	4	0	0,00	100,00
3	P	1	0	0	1	100,00	0,00
4	C	2	0	2	0	0,00	100,00
5	C	2	0	2	0	0,00	100,00
6	C	1	0	1	0	0,00	100,00
7	C	1	0	1	0	0,00	100,00
8	P	1	0	0	1	100,00	0,00
9	P	1	0	0	1	100,00	0,00
10	C	1	0	1	0	0,00	100,00
11	P	3	1	1	3	75,00	25,00
12	P	3	1	1	3	75,00	25,00
13	P	1	0	0	1	100,00	0,00
14	P	4	3	3	4	57,14	42,86
15	C	4	1	4	1	20,00	80,00
16	C	1	0	1	0	0,00	100,00
17	C	11	5	11	5	31,25	68,75
18	P	12	8	8	12	60,00	40,00
19	C	15	9	15	9	37,50	62,50
20	C	1	0	1	0	0,00	100,00
21	P	1	1	1	1	50,00	50,00
22	C	47	25	47	25	34,72	65,28
Totais		**118**	**54**	**104**	**68**	**39,53**	**60,47**

Tabela 11 – Resultado da avaliação das regras (sem o atributo ITR) aplicadas ao conjunto de treinamento.

Num. Regra	alvo	REGRAS APLICADAS AO CONJUNTO DE TESTE					
		C	P	acertos	erros	% ERRO	% ACERTO
1	P	0	1	1	0	0,00	100,00
2	C	8	5	8	5	38,46	61,54
3	P	5	0	0	5	100,00	0,00
4	C	6	1	6	1	14,29	85,71
5	C	6	0	6	0	0,00	100,00
6	C	2	3	2	3	60,00	40,00
7	C	2	0	2	0	0,00	100,00
8	P	3	3	3	3	50,00	50,00
9	P	3	2	2	3	60,00	40,00
10	C	0	0	0	0	0,00	0,00
11	P	7	7	7	7	50,00	50,00
12	P	6	11	11	6	35,29	64,71
13	P	4	0	0	4	100,00	0,00
14	P	0	0	0	0	0,00	0,00
15	C	0	0	0	0	0,00	0,00
16	C	3	1	3	1	25,00	75,00
17	C	39	11	39	11	22,00	78,00
18	P	28	27	27	28	50,91	49,09
19	C	55	18	55	18	24,66	75,34
20	C	0	0	0	0	0,00	0,00
21	P	2	1	1	2	66,67	33,33
22	C	203	100	203	100	33,00	67,00
Totais		**382**	**191**	**376**	**197**	**34,38**	**65,62**

Tabela 12 - Resultado da avaliação das regras (sem o atributo ITR) aplicadas ao conjunto de teste.

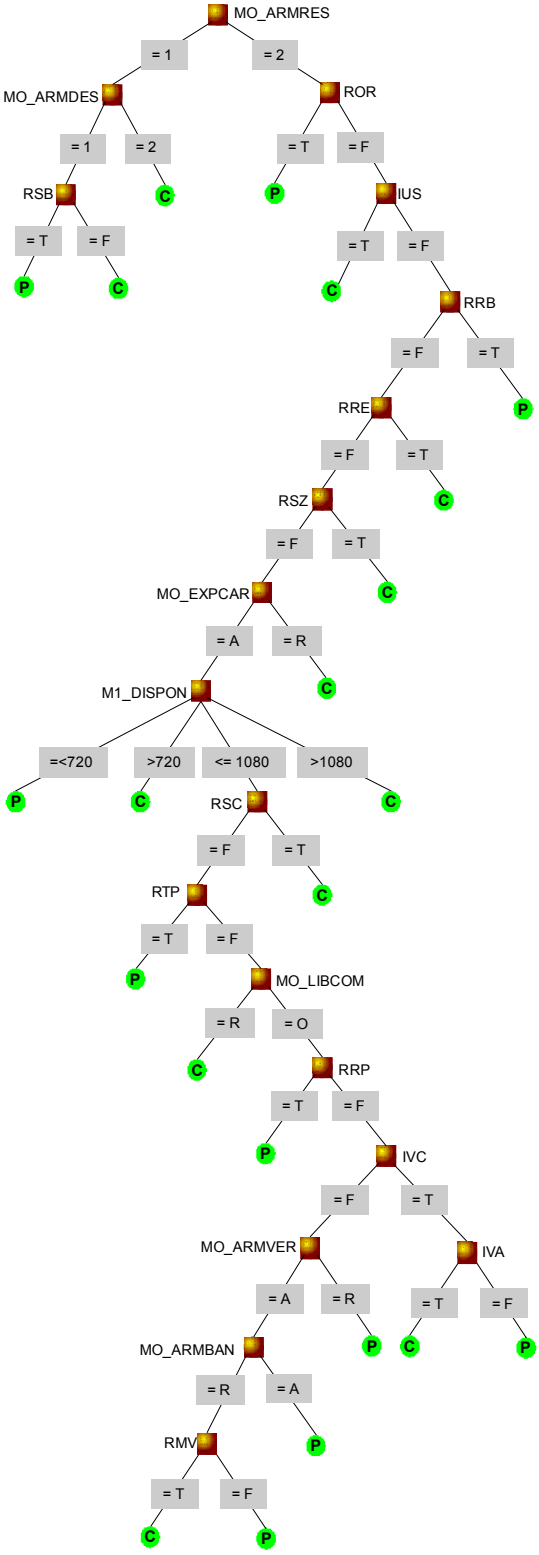


Figura 30 - Árvore de decisão gerada após a retirada do atributo ITR.

Assim como foi feito na primeira árvore induzida, esta nova árvore também passou por um processo de *post-prunning* numa tentativa de aumentar a capacidade de generalização e predição de novos casos. Neste processo de poda foram seguidos os mesmos critérios e procedimentos realizados anteriormente e os resultados obtidos podem ser vistos na Figura 31 e na Tabela 13. A Tabela 14 apresenta a avaliação deste novo conjunto de regras após a poda.

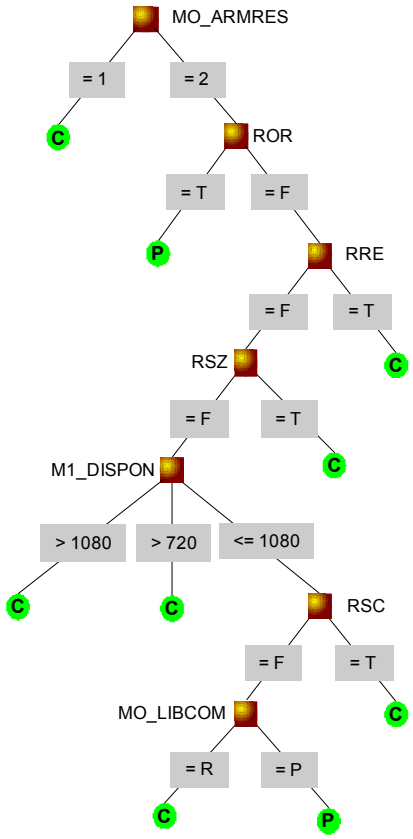


Figura 31 - Árvore de decisão sem o atributo ITR após o *post-prunning*.

Num. Regra	alvo	Regra
1	C	mo_armres = 1
2	P	mo_armres = 2 and ror = 't'
3	C	mo_armres = 2 and ror = 'f' and rre = 't'
4	P	mo_armres = 2 and ror = 'f' and rre = 'f' and rsz = 't'
5	C	mo_armres = 2 and ror = 'f' and rre = 'f' and rsz = 'f' and m1_dispon > 720
6	C	mo_armres = 2 and ror = 'f' and rre = 'f' and rsz = 'f' and m1_dispon > 1080
7	C	mo_armres = 2 and ror = 'f' and rre = 'f' and rsz = 'f' and m1_dispon <= 1080 and rsc = 't'
8	C	mo_armres = 2 and ror = 'f' and rre = 'f' and rsz = 'f' and m1_dispon <= 1080 and rsc = 'f' and mo_libcom = 'R'
9	P	mo_armres = 2 and ror = 'f' and rre = 'f' and rsz = 'f' and m1_dispon <= 1080 and rsc = 'f' and mo_libcom = 'O'

Tabela 13 - Conjunto de regras após o processo de *post-prunning* (sem o atributo ITR).

Num. Regra	alvo	REGRAS APLICADAS AO CONJUNTO DE TREINAMENTO						REGRAS APLICADAS AO CONJUNTO DE TESTE					
		C	P	acertos	erros	% ERRO	% ACERTO	C	P	acertos	erros	% ERRO	% ACERTO
1	C	49	26	49	26	34,67	65,33	206	102	206	102	33,12	66,88
2	P	0	1	1	0	0,00	100,00	0	1	1	0	0,00	100,00
3	C	3	0	3	0	0,00	100,00	9	2	9	2	18,18	81,82
4	P	4	0	0	4	100,00	0,00	10	3	3	10	76,92	23,08
5	C	17	9	17	9	34,62	65,38	60	20	60	20	25,00	75,00
6	C	13	5	13	5	27,78	72,22	42	13	42	13	23,64	76,36
7	C	1	0	1	0	0,00	100,00	3	1	3	1	25,00	75,00
8	C	1	3	1	3	75,00	25,00	6	2	6	2	25,00	75,00
9	P	10	14	14	10	41,67	58,33	40	33	33	40	54,79	45,21
Totais		98	58	99	57	36,54	63,46	376	177	363	190	34,36	65,64

Tabela 14 - Avaliação do novo conjunto de regras (sem o atributo ITR).

Sub-conjunto de dados	Antes do *post-prunning*		Depois do *post-prunning*	
	% ERRO	% ACERTO	% ERRO	% ACERTO
Treinamento	39,53	60,47	36,54	63,46
Teste	34,38	65,62	34,36	65,64

Tabela 15 - Comparação do desempenho da árvore antes e depois do processo de *post-prunning* para os conjuntos de treinamento e de teste.

O desempenho da árvore para os conjuntos de treinamento e teste antes e depois do processo de *post-prunning* está resumido na Tabela 15. Para a árvore sem o atributo ITR, diferentemente do que ocorreu na árvore anterior, os resultados para o conjunto de treinamento melhoram, enquanto os obtidos para o conjunto de teste praticamente se mantiveram.

Um comparativo dos resultados de desempenho para as duas árvores analisadas pode ser visto na Tabela 16. Nesta tabela observa-se que os valores %ERRO e %ACERTO praticamente não foram afetados pela ausência do atributo ITR entre os atributos previsores, sugerindo a sua baixa influência no resultado.

Árvore	Sub-conjunto de dados	Antes do *post-prunning*		Depois do *post-prunning*	
		% ERRO	% ACERTO	% ERRO	% ACERTO
com ITR	Treinamento	31,43	68,57	36,64	63,36
	Teste	34,93	65,07	33,19	66,81
sem ITR	Treinamento	39,53	60,47	36,54	63,46
	Teste	34,38	65,62	34,36	65,64

Tabela 16 - Comparativo dos desempenhos das árvores com e sem o atributo ITR.

7. Conclusões

Os dados analisado neste trabalho, nos Capítulos 5 e 6, estavam armazenados em tabelas de um banco de dados como ocorrem na maioria das situações. No entanto, quando são utilizados programas tradicionais de indução de árvores de decisão que trabalham diretamente com os dados em arquivos texto, esta forma de armazenagem exige um trabalho adicional para a conversão, das tabelas do banco de dados, para um único arquivo no formato de texto.

Este formato, apesar de conferir alto desempenho computacional, pode exigir o uso de técnicas especiais para lidar com estes arquivos quando o volume de dados é muito grande, o que aumenta a complexidade do programa de indução (QUINLAN, 1993).

Mesmo existindo algumas ressalvas quanto ao desempenho computacional, há uma tendência no sentido da integração entre as técnicas de mineração de dados e os sistemas de banco de dados, conforme discutido no capítulo 4.

Como conseqüência desta crescente integração, muitos esforços são realizados para aprimorar os recursos disponibilizados pelos gerenciadores de banco de dados como, por exemplo, implementar novas funções na linguagem SQL para mineração de dados e melhorar o desempenho dos algoritmos de acesso e recuperação destes dados.

Apesar da alegação de que a abordagem utilizando SGBDs penaliza o desempenho, ela traz uma série de outras vantagens tais como:

- a possibilidade de trabalhar com uma grande quantidade de dados sem a necessidade de converter tabelas para arquivos no formato texto;
- a possibilidade de acessar os dados dentro de uma arquitetura multirelacional com a mesma estrutura do ambiente transacional;
- simplificar o programa de indução, pois este não precisa mais agregar funções para realizar a busca e o agrupamento de registros;
- poder trabalhar com um grande volume de dados sem a necessidade de implementar técnicas especiais para manipular arquivos texto;
- contar com o contínuo aprimoramento dos SGBDs, pois os fornecedores tem investido pesadamente na melhoria de seus algoritmos de busca e recuperação de dados;
- novas funcionalidades relacionadas às necessidades da tarefa de mineração de dados tem sido desenvolvidas pelos fornecedores de SGBDs.

Neste trabalho a principal vantagem obtida, com o uso de um SGBD, foi a

simplificação das rotinas do programa de indução da árvore de decisão, já que não foi necessário implementar funcionalidades adicionais para manipular arquivos texto. Ao invés disto foi necessário apenas gerar as declarações SQL, enviá-las ao SGBD para execução e depois manipular o conjunto de dados ou os valores retornados.

O principal objetivo deste trabalho - a determinação da relação entre as falhas em motores elétricos de tração e os processos de manutenção a que foram submetidos em oficina - não ficou completamente estabelecido, pois o desempenho das regras obtidas apresentou um resultado aquém do esperado. Esperava-se um erro de predição da ordem de 10 a 15% para o conjunto de teste, mas o melhor percentual de erro obtido neste trabalho foi da ordem de 33%. Esta expectativa de erro teve como base os resultados apresentados em outros trabalhos, os quais, apesar de terem utilizados dados de natureza diferente, serviram como referência:

- Leiva (2002) obteve um erro médio 25% utilizando conjuntos de dados da área médica e do estudo de proteínas;
- Atramentov (2003) obteve um erro médio de 15% a partir do aperfeiçoamento do algoritmo desenvolvido por Leiva (2002), utilizando, para comparação, a mesma base de dados da área médica e do estudo de proteínas;
- Perner; Trautzsch (1998), com dados de imagens de satélites, obtiveram taxas de erro de 2,7 a 26 % usando o C4.5 (QUINLAN, 1993).

O fato de não se ter obtido melhores resultados levanta alguns pontos que requerem uma análise mais aprofundada, já que podem influenciar no resultado:

1. os atributos utilizados podem não ser os ideais para este tipo de análise. Os dados históricos existentes e usualmente registrados pela equipe de manutenção da oficina de motores podem ser os mais adequados para o controle dos processos de manutenção, mas não para o ponto de vista da predição. Neste caso, talvez devam ser obtidos e adicionados aos conjuntos de treinamento e teste, os dados das condições operacionais e ambientais, como, por exemplo, a carga (esforço que o motor realiza durante o seu funcionamento), tempo de funcionamento ou ciclo de trabalho e as temperaturas ambiente e do motor.
2. a ocorrência de falhas/defeitos é um fenômeno de natureza aleatória, não existindo um padrão de comportamento;
3. a técnica empregada para a análise ou mineração pode não ser a mais adequada e outras técnicas devem ser testadas;
4. não se tem clareza de qual o conjunto mínimo de atributos necessários para

análise deste tipo de equipamento, sendo necessário determinar quais são imprescindíveis;

5. a retirada de um dos atributos previsores, apesar de não alterar significativamente o resultado em termos de erro de predição, modificou de forma acentuada a configuração da árvore após o *post-prunning* (Figura 31 - Árvore de decisão sem o atributo ITR após o *post-prunning*.). Isto leva a crer que são necessários vários testes, inserindo e retirando atributos, até se chegar a uma versão da árvore, que melhor represente o fenômeno estudado. Como resultado destes testes pode-se chegar ao conjunto mínimo de atributos mencionado no ponto anterior.

Para se quantificar a influência destes pontos é necessária a realização de testes adicionais, cuja análise dos resultados deverá ser auxiliada por um especialista em motores elétricos de corrente contínua.

Pelo exposto até o momento, fica claro que o sucesso no uso de técnicas de mineração de dados para a obtenção de padrões de comportamento ou regras de classificação depende da experiência e do conhecimento de um especialista no assunto a que se referem os dados. A este especialista cabe não só as funções de analisar quais campos devem ser tratados como atributos previsores e a melhor forma de discretizar os atributos contínuos mas, principalmente, avaliar os resultados obtidos quanto à sua relevância e utilidade prática, já que disto podem resultar ações para aumentar a confiabilidade do equipamento ou melhorar a eficiência dos processos de manutenção.

Os resultados obtidos mostraram que o uso de árvores de decisão em conjunto com um sistema de banco de dados facilita a tarefa de agrupamento das tuplas e a determinação da freqüência das ocorrências dos valores do atributo alvo.

Além disto, caso se façam, em tempo de execução, ajustes semelhantes aos descritos na fase de pré-processamento, o programa de indução poderá acessar diretamente uma réplica da base de dados transacional.

Apesar do desempenho computacional da implementação não ter sido o foco neste trabalho, algumas possibilidades de comparação podem ser avaliadas no futuro:

- comparar o desempenho computacional com os obtidos por programas como o C4.5 (QUINLAN, 1993), Weka (WITTEN, FRANK, 2005) e Tanagra (RAKOTOMALALA, 2005);
- avaliar o desempenho de predição de resultados entre o programa desenvolvido e os citados no item anterior;

- levantar outros programas de indução que também utilizam o acesso aos dados de um SGBD para comparação do desempenho, tanto computacional quanto da precisão da predição;
- comparar o desempenho do algoritmo acessando um sistema de arquivos texto contra um sistema de banco de dados. Uma comparação deste tipo serviria para, por exemplo, verificar qual deles tem melhor desempenho ao trabalhar ou analisar um grande volume de dados.

Embora o uso de tabelas seja uma boa forma de armazenagem do resultado produzido pelo programa de indução, esta não é a melhor forma de apresentação e exigirá trabalho adicional para a geração de uma representação gráfica, feita de forma automatizada e de fácil leitura e interpretação.

Por fim, apesar do desempenho das regras obtidas ter ficado abaixo do esperado, os resultados obtidos mostram que o uso da abordagem proposta apresenta uma série de vantagens, necessitando de pequenos ajustes para que possa ser utilizado em outros tipos de equipamentos e sistemas.

8. Referências bibliográficas

ATRAMENTOV, A.; LEIVA, H.; HONAVAR, V.. A multi-relational decision tree learning algorithm – Implementation and experiments. Thirteenth International Conference on Inductive Logic Programming, Berlin, **Proceedings**. Springer-Verlag, 2003.

BLOCKEEL H.; CALDERS T.; FROMONT E.; GOETHALS B.; PRADO A. Mining views: database views for data mining. In: ECML/PKDD-2007 International Workshop on Constraint-Based Mining and Learning (CMILE), Warsaw, Poland. **Proceedings**. Springer-Verlag, September/2007

BRAHA, D.; SHMILOVICI, A. On the use of decision tree induction for discovery of interactions in a photolithographic process. **IEEE Transactions on Semiconductor Manufacturing**, vol. 16, no. 4, p. 644-652, November/2003.

BRUHA, I.; FRANEK, F. Comparison of Various Routines for Unknown Attribute Value Processing: Covering Paradigm**. International Journal Pattern Recognition and Artificial Intelligence**, vol. 10, no. 8, p. 939-955, 1996.

CARDOSO JUNIOR, G.; ROLIM, J. G.; ZÜRN, H. H. Diagnóstico de faltas em sistemas de potência: definição do problema e abordagens via inteligência artificial. **Revista Controle & Automação**, v. 15, n. 2, p. 215-229, abr./maio/jul. 2004.

CHEN, M.; ZHENG, A.X.; LLOYD, J.; JORDAN, M.I.; BREWER, E. Failure diagnosis using decision trees. In: First International Conference on Autonomic Computing, New York City, New York. **Proceedings,** p. 36- 43, May/2004.

FAYYAD, U. M., & IRANI, K. B. Multi-interval discretization of continuous-valued attributes for classification learning. In: 13th International Joint Conference on Artificial Intelligence, Chambery, France. **Proceedings**. Morgan-Kaufmann, p. 1022-1027, 1993.

FRANK, E.; WITTEN, I. H. (1996). Selecting multiway splits in decision trees. Working paper 96/31. Department of Computer Science, University of Waikato, December. URL: www.cs.waikato.ac.nz/~ml/publications/1996/Frank-Witten96.pdf

HELENE, O. A. M.; VANIN, V. R. Tratamento estatístico de dados em física experimental. São Paulo: Edgard Blücher, 1981.

LEIVA, H.A., MRDTL: A Multi-Relational Decision Tree Learning Algorithm, M.Sc. Thesis, Iowa State University, Ames, Iowa, USA, 2002.

LEIVA, H.; ATRAMENTOV, A.; HONAVAR, V. Experiments with MRDTL - A Multirelational Decision Tree Learning Algorithm. In: Workshop on Multi-Relational Decision Tree Learning. KDD-2002. **Proceedings of the ACM/SIGKDD**, 2002.

MITCHELL, T. M. **Machine Learning**, USA: McGraw-Hill, 1997.

MOUBRAY, J. **Manutenção Centrada em Confiabilidade**, United Kingdown: Aladon Ltd, 2000.

MURRAY, J. F.; HUGHES, G. F.; KREUTZ-DELGADO, K. Machine learning methods for predicting failures in hard drives: a multiple-instance application. **Journal of Machine Learning Research**, vol. 6, p. 783-816, May/2005.

NETZ A.; BERNHARDT J.; CHAUDHURI S.; FAYYAD U. integrating data mining with SQL databases: OLE DB for data mining. *In: 17th International Conference on Data Engineering, Heidelberg, Germany. Proceedings, 2001.*

ONODA, M.; EBECKEN, N. F. F. Implementação em Java de um Algoritmo de Árvore de Decisão Acoplado a um SGBD Relacional. XVI Simpósio Brasileiro de Banco de Dados, Rio de Janeiro, Brasil. **Anais/Proceedings**. COPPE/UFRJ Outubro/2001 p. 55-64

PERNER, P.; TRAUTZSCH, S. Multinterval Discretization For Decision Tree Learning. Advances In Pattern Recognition, LNCS 1451, Springer Verlag 1998, p. 475-482.

PINTO, A. K.; XAVIER, J. A. N. **Manutenção**: função estratégica, 2ª edição, Rio de Janeiro: Qualitymark Ed., 2001.

QUINLAN, J. R. Induction of Decision Trees. **Machine Learning**, Boston, v.1, p. 81-106, March/1986.

QUINLAN, J. R. Unknown Attribute Value in Induction. In: Sixth Machine Learning Workshop, San Mateo, CA. **Proceedings**. Morgan Kaufmann, p. 164-168, March/1996.

QUINLAN, J. R. **C4.5: Programs form machine learning**, London, England: Morgan Kaufmann, 1993.

QUINLAN, J. R. Improved Use of Continuous Attributes in C4.5. **Journal of Artificial Intelligence Research**, v.4, p. 77-90, March/1996.

REZENDE, S. O. (coord). **Sistemas inteligentes: fundamentos e aplicações**, Barueri, São Paulo: Manole, 2005.

RAKOTOMALALA, R. Tanagra: a free software for research and academic purposes. In: EGC2005 - European Grid Conference RNTI-E-3, Amsterdam, The Netherlands. **Proceedings**, vol. 2, p. 697-702, February/2005. (in French)

RUSSELL, S. J.; NORVIG, P. **Inteligência Artificial**: tradução da segunda edição. Rio de Janeiro: Elsevier, 2004.

SATTLER K.; DUNEMANN O.. SQL database primitives for decision tree classifiers. In: Tenth International Conference on Information and Knowledge Management, Buckhead, Atlanta, USA. **Proceedings.** ACM Press, p. 379-386, November/2001.

SETZER, V.W.; SILVA, F.S.C. **Banco de Dados**: aprenda o que são, melhore seu conhecimento, construa os seus, 1ª edição, São Paulo: Edgard Blücher, 2005.

SILVA, E. B.; XEXÉO G. B. Uma Primitiva para dar Suporte à Obtenção de Resumos Estatísticos para Classificação em Mineração de Dados. In: XIV SIMPÓSIO BRASILEIRO DE BANCO DE DADOS, Florianópolis, Santa Catarina, Brasil, Outubro 1999, **Anais/Proceedings**. UFSC, p. 333-347, 1999.

WITTEN, I. H.; FRANK, E. **Data Mining**: Practical machine learning tools and techniques, 2nd Edition, Morgan Kaufmann, San Francisco, 2005.

9. Glossário

Atributo alvo: também chamado de classe, representa o valor de saída de um exemplo ou caso e está relacionado a um ou mais atributos previsores. Na árvore é representado pela figura da folha.

Atributo previsor: descreve uma característica ou um aspecto de um exemplo ou caso. Pode possuir valores discretos ou contínuos.

dBase III: primeiro SGBD largamente utilizado industrialmente.

Caixa de redução: conjunto de engrenagens que funcionam em conjunto de forma a reduzir a rotação e aumentar o torque.

Caso ou **exemplo**: é uma tupla de valores dos atributos previsores mais o atributo alvo. Representa o registro de uma tabela.

Classificador: Representa um conjunto de regras que deve ser capaz de uma vez dado um novo exemplo, cujo atributo alvo seja desconhecido, prever o valor deste atributo com base nos atributos previsores. É o resultado do processo de indução.

Classificador Bayesiano simples: tipo de classificador que considera que o efeito do valor de um atributo previsor sobre um determinado atributo alvo é independente dos valores dos outros atributos previsores.

Clipper: linguagem de programação criada em 1984 com o propósito de ser um compilador para o banco de dados dBase.

Conjunto de teste: conjunto de tuplas de valores usado para validar o resultado gerado pelo classificador.

Conjunto de treinamento: conjunto de tuplas de valores usado pelo indutor para a geração do classificador.

Discretização: processo de representar na forma de valores discretos um conjunto de dados composto por valores contínuos sem perda significativa de informação.

Escalabilidade: se refere ao quão bem um hardware ou software pode se adaptar ao aumento de demanda.

Folha: representa uma classe objetivo ou um valor possível do atributo alvo.

Indutor: programa de aprendizado ou algoritmo de indução responsável pela extração do classificador a partir de um conjunto de exemplos.

Java: linguagem de programação orientada a objetos e dotada de grande portabilidade, desenvolvida nos anos 90, pela Sun Microsystems.

Linguagem procedural: linguagem de programação na qual o elemento básico de programação é a *procedure* (uma seqüência de instruções – rotina, sub-rotina ou função – associadas a um nome próprio).

Metadados: informações relativas aos dados.

Nó de Decisão: representa o resultado do teste. Existe um galho para cada um dos possíveis resultados do teste feito no nó.

Portabilidade: no contexto da informática é a capacidade que uma solução (um programa ou aplicativo) tem de ser executada em diversas plataformas e arquiteturas.

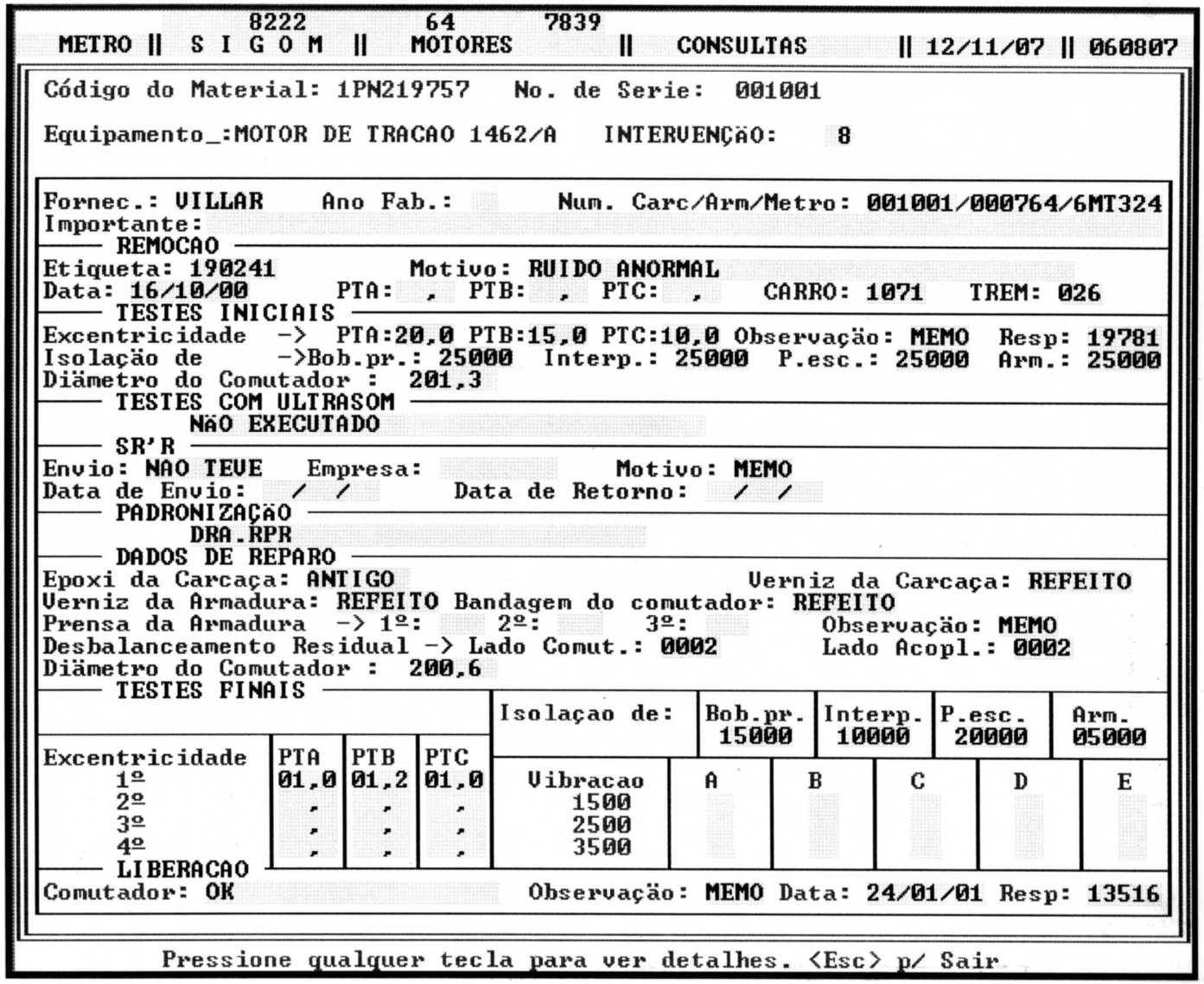
Oracle: sistema gerenciador de banco de dados surgido no final dos anos 70 desenvolvido pela Oracle Corporation.

Primitivas SQL: operadores básicos da linguagem SQL a partir dos quais se geram declarações para a definição e a manipulação das entidades de um banco de dados.

Ruídos: erros aleatórios nos dados, sem causa aparente.

Anexo 1

Tela de consulta das fichas de motores no Sistema Integrado de Gestão da Oficinas de Manutenção



```
                    8222          64       7839
METRO || S I G O M  ||  MOTORES        ||  CONSULTAS      || 12/11/07 || 060807

Código do Material: 1PN219757   No. de Serie:  001001

Equipamento_:MOTOR DE TRACAO 1462/A   INTERVENÇÃO:    8

Fornec.: VILLAR    Ano Fab.:        Num. Carc/Arm/Metro: 001001/000764/6MT324
Importante:
 REMOCAO
Etiqueta: 190241          Motivo: RUIDO ANORMAL
Data: 16/10/00      PTA:  ,  PTB:  ,  PTC:  ,    CARRO: 1071    TREM: 026
 TESTES INICIAIS
Excentricidade  ->  PTA:20,0 PTB:15,0 PTC:10,0 Observaçäo: MEMO  Resp: 19781
Isolaçäo de     ->Bob.pr.: 25000  Interp.: 25000  P.esc.: 25000  Arm.: 25000
Diämetro do Comutador :  201,3
 TESTES COM ULTRASOM
       NÄO EXECUTADO
 SR'R
Envio: NAO TEVE   Empresa:          Motivo: MEMO
Data de Envio:   /  /      Data de Retorno:   /  /
 PADRONIZAÇÃO
       DRA.RPR
 DADOS DE REPARO
Epoxi da Carcaça: ANTIGO                    Verniz da Carcaça: REFEITO
Verniz da Armadura: REFEITO Bandagem do comutador: REFEITO
Prensa da Armadura  -> 1º:     2º:      3º:          Observaçäo: MEMO
Desbalanceamento Residual -> Lado Comut.: 0002       Lado Acopl.: 0002
Diämetro do Comutador :  200,6
 TESTES FINAIS
                                  Isolaçao de:  Bob.pr.  Interp.  P.esc.  Arm.
                                                15000    10000    20000   05000
Excentricidade  PTA   PTB   PTC
      1º        01,0  01,2  01,0   Vibracao     A        B        C       D       E
      2º          ,     ,     ,      1500
      3º          ,     ,     ,      2500
      4º          ,     ,     ,      3500
 LIBERACAO
Comutador: OK                  Observaçäo: MEMO Data: 24/01/01 Resp: 13516

        Pressione qualquer tecla para ver detalhes. <Esc> p/ Sair
```

Anexo 2

Tabela de códigos de reparo

TABELA DE CÓDIGOS DE REPARO

MOTORES DE TRAÇÃO	
D10	REVISÃO DOS 10 ANOS
DAC	ARMADURA EM CURTO
DBI	BAIXA ISOLAÇÃO
DBR	BANDAGEM ROMPIDA
DCB	CABOS E/OU CONEXÕES COM DEFEITO
DCD	COMUTADOR COM DESVIO
DCE	COMUTADOR EXCÊNTRICO
DCG	COMUTADOR GASTO
DCR	COMUTADOR RISCADO
DCT	COMUTADOR TRINCADO
DDA	DANOS POR ARCO
DDE	DESGASTE DE ESCOVA
DEI	DEFEITO NO EIXO
DEQ	BOBINA DE EQUALIZACÃO
DFO	FLASH OVER
DIC	ISOLAÇÃO DO COLARINHO
DLS	LAMINA SALTADA
DMA	MICA ALTA
DOD	OUTROS DEFEITOS
DQE	QUEBRANDO ESCOVA
DRA	RUÍDO ANORMAL
DRD	ROLAMENTO DANIFICADO
DRG	REVISÃO GERAL
DRQ	ROLAMENTO QUEBRADO
DSP	SUBST. PROGRAMADA DE COMUTADOR
DTE	TRIÂNGULO DE FIXAÇÃO DO MOTOR EMPENADO
DTF	TAMPA COM FOLGA
DTT	TRIÂNGULO DE FIXAÇÃO DO MOTOR TRINCADO
DVB	VIBRAÇÃO EXCESSIVA
IAM	APLICADO MASSA P/ NIVELAMENTO DO TRIÂNGULO
IIE	INSPEÇÃO DE TRIÂNGULO
IIV	INJETADO VERNIZ

MOTORES DE TRAÇÃO	
IRC	ROLAMENTO COLADO
IRU	ROLAMENTO USADO
ISA	SUBST. ARMADURA
ISO	TRIÂNGULO SOLDADO
ITR	TROCA DE ROLAMENTO
IUS	ULTRA SOM
IUT	TRIÂNGULO USINADO
IVA	VERNIZA ARMADURA
IVC	VERNIZA CARCAÇA
IBL	BALANCEAMENTO
ISC	SUBST. DE CABOS
ISR	SUBST. DE ROLAMENTO
ICO	COLARINHO REFEITO
IRM	REBAIXAMENTO DE MICA
RBI	REISOALAÇÃO BOB. DE INTERPOLO
RBP	REISOALAÇÃO BOB. DE POLO
RME	METALIZAÇÃO DO EIXO
RMV	REPARO MEIA VIDA
ROR	OUTROS REPAROS (INTERNO)
RCM	REPARO CONE DE MICA
RPR	PROCESSO
RRB	REPARO DE BANDAGEM
RRE	REENROLADO
RRP	REPARO EXTERNO
RSB	SERVIÇO BÁSICO
RSC	SUBST. COMUTADOR
RSE	SUBST. EIXO
RSI	SUBST. BOB. DE INTERPOLO
RSP	SUBST. BOB. DE POLO
RSZ	SAZONAMENTO
RTP	RETIFICA COM PEDRA

Apêndice 1

Script SQL para a criação da tabela principal – “fichamotor”

```
CREATE TABLE fichamotor
(
  nr_ficha serial NOT NULL,
  e1_codest character varying(9),
  m1_nserie character varying(6),
  mo_interv smallint,
  mo_etdata date,
  mo_etipt_1 numeric(3,1),
  mo_etipt_2 numeric(3,1),
  mo_etipt_3 numeric(3,1),
  mo_ofipt_1 numeric(3,1),
  mo_ofipt_2 numeric(3,1),
  mo_ofipt_3 numeric(3,1),
  mo_tsresp character varying(5),
  mo_tsultr character varying(1),
  mo_tsresul character varying(1),
  m2_numse integer,
  mo_srenvio character varying(1),
  mo_srempr character varying(8),
  mo_srdtenv date,
  mo_srdtret date,
  mo_epxcar character varying(1),
  mo_vrzcar character varying(1),
  mo_armpr_1 numeric(3),
  mo_armpr_2 numeric(3),
  mo_armpr_3 numeric(3),
  mo_armver character varying(1),
  mo_armban character varying(1),
  mo_armdes numeric(4),
  mo_armres numeric(4),
  mo_isolaca_1 numeric(5),
  mo_isolaca_2 numeric(5),
  mo_isolaca_3 numeric(5),
  mo_isolaca_4 numeric(5),
  mo_expt1_1 numeric(4),
  mo_expt1_2 numeric(4),
  mo_expt1_3 numeric(4),
  mo_expt2_1 numeric(4),
  mo_expt2_2 numeric(4),
  mo_expt2_3 numeric(4),
  mo_expt3_1 numeric(4),
  mo_expt3_2 numeric(4),
  mo_expt3_3 numeric(4),
  mo_expt4_1 numeric(4),
  mo_expt4_2 numeric(4),
  mo_expt4_3 numeric(4),
  mo_vib15_1 numeric(2),
  mo_vib15_2 numeric(2),
  mo_vib15_3 numeric(2),
  mo_vib15_4 numeric(2),
  mo_vib15_5 numeric(2),
  mo_vib25_1 numeric(2),
  mo_vib25_2 numeric(2),
  mo_vib25_3 numeric(2),
  mo_vib25_4 numeric(2),
  mo_vib25_5 numeric(2),
  mo_vib35_1 numeric(2),
  mo_vib35_2 numeric(2),
  mo_vib35_3 numeric(2),
  mo_vib35_4 numeric(2),
  mo_vib35_5 numeric(2),
  mo_libcom character varying(1),
  mo_libdata date,
  mo_libresp character varying(5),
  mo_tstiso_1 numeric(5),
  mo_tstiso_2 numeric(5),
  mo_tstiso_3 numeric(5),
  mo_tstiso_4 numeric(5),
  mo_stsigom character varying(1),
  mo_etiq character varying(6),
  mo_km numeric(7),
  mo_diamini numeric(6,1),
  mo_diamrep numeric(6,1),
  m1_dispon integer,
  d10 boolean DEFAULT false,
  dac boolean DEFAULT false,
  das boolean DEFAULT false,
  dbi boolean DEFAULT false,
  dbr boolean DEFAULT false,
  dca boolean DEFAULT false,
  dcb boolean DEFAULT false,
  dcd boolean DEFAULT false,
  dce boolean DEFAULT false,
  dcg boolean DEFAULT false,
  dci boolean DEFAULT false,
  dcq boolean DEFAULT false,
  dcr boolean DEFAULT false,
  dct boolean DEFAULT false,
  dda boolean DEFAULT false,
  dde boolean DEFAULT false,
  ddf boolean DEFAULT false,
  dei boolean DEFAULT false,
  dea boolean DEFAULT false,
  "dec" boolean DEFAULT false,
  deq boolean DEFAULT false,
  det boolean DEFAULT false,
  dfo boolean DEFAULT false,
  dgr boolean DEFAULT false,
  dic boolean DEFAULT false,
  dls boolean DEFAULT false,
  dma boolean DEFAULT false,
  dod boolean DEFAULT false,
  dqe boolean DEFAULT false,
  dra boolean DEFAULT false,
  drb boolean DEFAULT false,
  drc boolean DEFAULT false,
  drd boolean DEFAULT false,
  drg boolean DEFAULT false,
  drq boolean DEFAULT false,
  dsp boolean DEFAULT false,
  dsz boolean DEFAULT false,
  dte boolean DEFAULT false,
  dtg boolean DEFAULT false,
  dtt boolean DEFAULT false,
  dvb boolean DEFAULT false,
  iam boolean DEFAULT false,
  iie boolean DEFAULT false,
  iiv boolean DEFAULT false,
  inc boolean DEFAULT false,
  iru boolean DEFAULT false,
  irc boolean DEFAULT false,
  isa boolean DEFAULT false,
  iso boolean DEFAULT false,
  itr boolean DEFAULT false,
  iuc boolean DEFAULT false,
  ius boolean DEFAULT false,
  iut boolean DEFAULT false,
  iva boolean DEFAULT false,
  ivc boolean DEFAULT false,
  ive boolean DEFAULT false,
  ivs boolean DEFAULT false,
  prp boolean DEFAULT false,
  r10 boolean DEFAULT false,
  rbr boolean DEFAULT false,
  rde boolean DEFAULT false,
  rmu boolean DEFAULT false,
  rmv boolean DEFAULT false,
  ror boolean DEFAULT false,
  rpp boolean DEFAULT false,
  rpr boolean DEFAULT false,
  rpt boolean DEFAULT false,
  rrb boolean DEFAULT false,
  rre boolean DEFAULT false,
  rrp boolean DEFAULT false,
  rsb boolean DEFAULT false,
  rsc boolean DEFAULT false,
  rsp boolean DEFAULT false,
  rsz boolean DEFAULT false,
  rtp boolean DEFAULT false,
  rtr boolean DEFAULT false,
  rvc boolean DEFAULT false,
  rwv boolean DEFAULT false,
  m1_tipoent character varying(1),
  CONSTRAINT fichamotor_pkey PRIMARY
               KEY (nr_ficha)
)
```

Apêndice 2

Algoritmo da rotina de pré-processamento

```
Preprocessamento (tabela_de_fichas, tabela_de_entradas) tabela_de_saída {
	Criar a tabela_de_saída;
	Para cada registro da tabela_de_fichas {
		Procurar a data de liberação na tabela_de_entradas;
		Se existir a data de liberação na tabela_de_entradas {
			Calcular o tempo que o motor ficou disponível;
			Substituir o tipo de entrada original pelo tipo da próxima entrada;
			Se for coluna com valores concatenados {
				Fazer consistências;
				Decompor coluna;
			} Se for a coluna 'mo_padron2' {
				Fazer consistências;
				para cada sigla em 'mo_padron2'{
					preencher o a coluna da sigla com 'true';
				}
			} Senão {
				fazer demais consistências;
			}

		} Se não existir {
			Descartar registro;
		}
		Incluir novo registro na tabela_de_saída;
	}
}
```

Apêndice 3

Resultado do processo de discretização dos atributos contínuos

atributo	valor
mo_etipt_1	5
mo_etipt_2	5
mo_etipt_3	5
mo_ofipt_1	2,5
mo_ofipt_1	5
mo_ofipt_2	2,5
mo_ofipt_2	6
mo_ofipt_3	2,5
mo_ofipt_3	5,5
mo_isolaca_1	7500
mo_isolaca_2	7500
mo_isolaca_3	7500
mo_isolaca_4	7500
mo_expt1_1	1,5
mo_expt1_1	2,5
mo_expt1_2	1,5
mo_expt1_2	2,5
mo_expt1_3	1,5
mo_expt1_3	2,5
mo_expt2_1	1
mo_expt2_1	2,5
mo_expt2_2	1
mo_expt2_2	2,5
mo_expt2_3	1
mo_expt3_1	1
mo_expt3_2	1
mo_expt3_3	1
mo_expt4_1	1
mo_expt4_2	1
mo_expt4_3	1
mo_vib15_1	7
mo_vib15_1	11
mo_vib15_1	16
mo_vib15_2	7

atributo	valor
mo_vib15_2	11
mo_vib15_2	16
mo_vib15_3	7
mo_vib15_3	11
mo_vib15_3	16
mo_vib15_4	6,5
mo_vib15_4	15
mo_vib15_4	25,5
mo_vib15_5	4
mo_vib35_2	11
mo_vib35_3	4,5
mo_vib35_3	11
mo_vib35_4	4,5
mo_vib35_4	10
mo_vib35_5	4,5
mo_vib35_5	10
mo_tstiso_1	7500
mo_tstiso_1	30000
mo_tstiso_2	7500
mo_tstiso_2	30000
mo_tstiso_3	7500
mo_tstiso_3	30000
mo_tstiso_4	7500
mo_tstiso_4	30000
mo_diamini	200,5
mo_diamini	201,5
mo_diamini	202,5
mo_diamini	203,5
mo_diamrep	195,5
mo_diamrep	200,5
mo_diamrep	201,5
mo_diamrep	202,5
m1_dispon	720
m1_dispon	1080

Apêndice 4

Tabela de metadados

atributo	tipo	tratamento	categoria	avaliar
m1_tipoent	S	D	A	S
mo_etipt_1	D	C	P	N
mo_etipt_2	D	C	P	N
mo_etipt_3	D	C	P	N
mo_ofipt_1	D	C	P	N
mo_ofipt_2	D	C	P	N
mo_ofipt_3	D	C	P	N
mo_tsultr	S	D	P	N
mo_tsresul	S	D	P	N
mo_srenvio	S	D	P	N
mo_srempr	S	D	P	N
mo_epxcar	S	D	P	S
mo_armpr_1	I	C	P	N
mo_armpr_2	I	C	P	N
mo_armpr_3	I	C	P	N
mo_armver	S	D	P	S
mo_armban	S	D	P	S
mo_armdes	I	D	P	S
mo_armres	I	D	P	S
mo_isolaca_1	I	C	P	N
mo_isolaca_2	I	C	P	N
mo_isolaca_3	I	C	P	N
mo_isolaca_4	I	C	P	N
mo_expt1_1	D	C	P	N
mo_expt1_2	D	C	P	N
mo_expt1_3	D	C	P	N
mo_expt2_1	D	C	P	N
mo_expt2_2	D	C	P	N
mo_expt2_3	D	C	P	N
mo_expt3_1	D	C	P	N
mo_expt3_2	D	C	P	N
mo_expt3_3	D	C	P	N
mo_expt4_1	D	C	P	N
mo_expt4_2	D	C	P	N
mo_expt4_3	D	C	P	N
mo_vib15_1	D	C	P	N
mo_vib15_2	D	C	P	N
mo_vib15_3	D	C	P	N
mo_vib15_4	D	C	P	N
mo_vib15_5	D	C	P	N
mo_vib25_1	D	C	P	N
mo_vib25_2	D	C	P	N
mo_vib25_3	D	C	P	N
mo_vib25_4	D	C	P	N
mo_vib25_5	D	C	P	N
mo_vib35_1	D	C	P	N
mo_vib35_2	D	C	P	N
mo_vib35_3	D	C	P	N
mo_vib35_4	D	C	P	N

atributo	tipo	tratamento	categoria	avaliar
mo_vib35_5	D	C	P	N
mo_libcom	S	D	P	S
mo_tstiso_1	I	C	P	N
mo_tstiso_2	I	C	P	N
mo_tstiso_3	I	C	P	N
mo_tstiso_4	I	C	P	N
mo_diamini	D	C	P	N
mo_diamrep	D	C	P	N
m1_dispon	I	C	P	S
d10	B	D	P	N
dac	B	D	P	N
das	B	D	P	N
dbi	B	D	P	N
dbr	B	D	P	N
dca	B	D	P	N
dcb	B	D	P	N
dcd	B	D	P	N
dce	B	D	P	N
dcg	B	D	P	N
dci	B	D	P	N
dcq	B	D	P	N
dcr	B	D	P	N
dct	B	D	P	N
dda	B	D	P	N
dde	B	D	P	N
ddf	B	D	P	N
dei	B	D	P	N
dea	B	D	P	N
dec	B	D	P	N
deq	B	D	P	N
det	B	D	P	N
dfo	B	D	P	N
dgr	B	D	P	N
dic	B	D	P	N
dls	B	D	P	N
dma	B	D	P	N
dod	B	D	P	N
dqe	B	D	P	N
dra	B	D	P	N
drb	B	D	P	N
drc	B	D	P	N
drd	B	D	P	N
drg	B	D	P	N
drq	B	D	P	N
dsp	B	D	P	N
dsz	B	D	P	N
dte	B	D	P	N
dtg	B	D	P	N
dtt	B	D	P	N

atributo	tipo	tratamento	categoria	avaliar
dvb	B	D	P	N
iam	B	D	P	S
iie	B	D	P	S
iiv	B	D	P	S
inc	B	D	P	S
iru	B	D	P	S
irc	B	D	P	S
isa	B	D	P	S
iso	B	D	P	S
itr	B	D	P	S
iuc	B	D	P	S
ius	B	D	P	S
iut	B	D	P	S
iva	B	D	P	S
ivc	B	D	P	S
ive	B	D	P	S
ivs	B	D	P	S
prp	B	D	P	S
r10	B	D	P	S
rbr	B	D	P	S
rde	B	D	P	S
rmu	B	D	P	S
rmv	B	D	P	S
ror	B	D	P	S
rpp	B	D	P	S
rpr	B	D	P	S
rpt	B	D	P	S
rrb	B	D	P	S
rre	B	D	P	S
rrp	B	D	P	S
rsb	B	D	P	S
rsc	B	D	P	S
rsp	B	D	P	S
rsz	B	D	P	S
rtp	B	D	P	S
rtr	B	D	P	S
rvc	B	D	P	S
rwv	B	D	P	S

Apêndice 5

Regras e avaliação de desempenho para fator de pureza igual 0,7

nr_id	alvo	Regra	REGRAS APLICADAS AO CONJUNTO DE TESTE					
			C	P	acertos	erros	% ERRO	% ACERTO
1	C	itr = 't'	2	0	2	0	0,00	100,00
2	P	itr = 'f' and mo_armres = 2 and ror = 't'	0	1	1	0	0,00	100,00
3	C	itr = 'f' and mo_armres = 2 and ror = 'f' and ius = 't'	8	5	5	8	61,54	38,46
4	P	itr = 'f' and mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 't'	5	0	0	5	100,00	0,00
5	C	itr = 'f' and mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 'f' and rre = 't'	6	1	6	1	14,29	85,71
6	C	itr = 'f' and mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 'f' and rre = 'f' and rsz = 't'	6	0	6	0	0,00	100,00
7	C	itr = 'f' and mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 'f' and rre = 'f' and rsz = 'f' and mo_epxcar = 'R'	2	3	2	3	60,00	40,00
8	C	itr = 'f' and mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 'f' and rre = 'f' and rsz = 'f' and mo_epxcar = 'A' and m1_dispon <= 1080 and rsc = 't'	2	0	2	0	0,00	100,00
9	P	itr = 'f' and mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 'f' and rre = 'f' and rsz = 'f' and mo_epxcar = 'A' and m1_dispon <= 1080 and rsc = 'f' and rtp = 't'	3	3	3	3	50,00	50,00
10	P	itr = 'f' and mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 'f' and rre = 'f' and rsz = 'f' and mo_epxcar = 'A' and m1_dispon <= 1080 and rsc = 'f' and rtp = 'f' and mo_libcom = 'O' and rrp = 't'	3	2	2	3	60,00	40,00
11	C	itr = 'f' and mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 'f' and rre = 'f' and rsz = 'f' and mo_epxcar = 'A' and m1_dispon <= 1080 and rsc = 'f' and rtp = 'f' and mo_libcom = 'O' and rrp = 'f' and ivc = 't' and mo_armban = 'R' and mo_armver = 'A'	4	1	4	1	20,00	80,00
12	P	itr = 'f' and mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 'f' and rre = 'f' and rsz = 'f' and mo_epxcar = 'A' and m1_dispon <= 1080 and rsc = 'f' and rtp = 'f' and mo_libcom = 'O' and rrp = 'f' and ivc = 't' and mo_armban = 'R' and mo_armver = 'R'	12	9	9	12	57,14	42,86
13	C	itr = 'f' and mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 'f' and rre = 'f' and rsz = 'f' and mo_epxcar = 'A' and m1_dispon <= 1080 and rsc = 'f' and rtp = 'f' and mo_libcom = 'O' and rrp = 'f' and ivc = 't' and mo_armban = 'A'	0	0	0	0		
14	P	itr = 'f' and mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 'f' and rre = 'f' and rsz = 'f' and mo_epxcar = 'A' and m1_dispon <= 1080 and rsc = 'f' and rtp = 'f' and mo_libcom = 'O' and rrp = 'f' and ivc = 'f'	17	18	17	18	51,43	48,57
15	C	itr = 'f' and mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 'f' and rre = 'f' and rsz = 'f' and mo_epxcar = 'A' and m1_dispon <= 1080 and rsc = 'f' and rtp = 'f' and mo_libcom = 'R'	3	1	3	1	25,00	75,00
16	C	itr = 'f' and mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 'f' and rre = 'f' and rsz = 'f' and mo_epxcar = 'A' and m1_dispon > 1080	39	11	39	11	22,00	78,00
17	P	itr = 'f' and mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 'f' and rre = 'f' and rsz = 'f' and mo_epxcar = 'A' and m1_dispon <= 720	28	27	27	28	50,91	49,09
18	C	itr = 'f' and mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 'f' and rre = 'f' and rsz = 'f' and mo_epxcar = 'A' and m1_dispon > 720	55	18	55	18	24,66	75,34
19	P	itr = 'f' and mo_armres = 1 and mo_armdes = 1 and rsb = 't'	2	1	1	2	66,67	33,33
20	C	itr = 'f' and mo_armres = 1 and mo_armdes = 1 and rsb = 'f'	202	100	202	100	33,11	66,89
21	C	itr = 'f' and mo_armres = 1 and mo_armdes = 2	0	1	0	1	100,00	0,00
		Totais	399	202	386	215	35,77	64,23

Apêndice 6

Regras e avaliação de desempenho para fator de pureza igual 0,8

nr_id	alvo	Regra	REGRAS APLICADAS AO CONJUNTO DE TESTE					
			C	P	acertos	erros	% ERRO	% ACERTO
1	C	itr = 't'	2	0	2	0	0,00	100,00
2	P	itr = 'f' and mo_armres = 2 and ror = 't'	0	1	1	0	0,00	100,00
3	C	itr = 'f' and mo_armres = 2 and ror = 'f' and ius = 't'	8	5	5	8	61,54	38,46
4	P	itr = 'f' and mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 't'	5	0	0	5	100,00	0,00
5	C	itr = 'f' and mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 'f' and rre = 't'	6	1	6	1	14,29	85,71
6	C	itr = 'f' and mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 'f' and rre = 'f' and rsz = 't'	6	0	6	0	0,00	100,00
7	C	itr = 'f' and mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 'f' and rre = 'f' and rsz = 'f' and mo_epxcar = 'R'	2	3	2	3	60,00	40,00
8	C	itr = 'f' and mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 'f' and rre = 'f' and rsz = 'f' and mo_epxcar = 'A' and m1_dispon <= 1080 and rsc = 't'	2	0	2	0	0,00	100,00
9	P	itr = 'f' and mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 'f' and rre = 'f' and rsz = 'f' and mo_epxcar = 'A' and m1_dispon <= 1080 and rsc = 'f' and rtp = 't'	3	3	3	3	50,00	50,00
10	P	itr = 'f' and mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 'f' and rre = 'f' and rsz = 'f' and mo_epxcar = 'A' and m1_dispon <= 1080 and rsc = 'f' and rtp = 'f' and mo_libcom = 'O' and rrp = 't'	3	2	2	3	60,00	40,00
11	C	itr = 'f' and mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 'f' and rre = 'f' and rsz = 'f' and mo_epxcar = 'A' and m1_dispon <= 1080 and rsc = 'f' and rtp = 'f' and mo_libcom = 'O' and rrp = 'f' and ivc = 't' and mo_armban = 'R' and mo_armver = 'A'	4	1	4	1	20,00	80,00
12	P	itr = 'f' and mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 'f' and rre = 'f' and rsz = 'f' and mo_epxcar = 'A' and m1_dispon <= 1080 and rsc = 'f' and rtp = 'f' and mo_libcom = 'O' and rrp = 'f' and ivc = 't' and mo_armban = 'R' and mo_armver = 'R'	12	9	9	12	57,14	42,86
13	C	itr = 'f' and mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 'f' and rre = 'f' and rsz = 'f' and mo_epxcar = 'A' and m1_dispon <= 1080 and rsc = 'f' and rtp = 'f' and mo_libcom = 'O' and rrp = 'f' and ivc = 't' and mo_armban = 'A'	0	0	0	0		
14	P	itr = 'f' and mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 'f' and rre = 'f' and rsz = 'f' and mo_epxcar = 'A' and m1_dispon <= 1080 and rsc = 'f' and rtp = 'f' and mo_libcom = 'O' and rrp = 'f' and ivc = 'f'	17	18	18	17	48,57	51,43
15	C	itr = 'f' and mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 'f' and rre = 'f' and rsz = 'f' and mo_epxcar = 'A' and m1_dispon <= 1080 and rsc = 'f' and rtp = 'f' and mo_libcom = 'R'	3	1	3	1	25,00	75,00
16	C	itr = 'f' and mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 'f' and rre = 'f' and rsz = 'f' and mo_epxcar = 'A' and m1_dispon > 1080	39	11	39	11	22,00	78,00
17	P	itr = 'f' and mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 'f' and rre = 'f' and rsz = 'f' and mo_epxcar = 'A' and m1_dispon <= 720	28	27	27	28	50,91	49,09
18	C	itr = 'f' and mo_armres = 2 and ror = 'f' and ius = 'f' and rrb = 'f' and rre = 'f' and rsz = 'f' and mo_epxcar = 'A' and m1_dispon > 720	58	18	58	18	23,68	76,32
19	P	itr = 'f' and mo_armres = 1 and mo_armdes = 1 and rsb = 't'	2	1	1	2	66,67	33,33
20	C	itr = 'f' and mo_armres = 1 and mo_armdes = 1 and rsb = 'f'	202	100	202	100	33,11	66,89
21	C	itr = 'f' and mo_armres = 1 and mo_armdes = 2	0	1	0	1	100,00	0,00
		Totais	402	202	390	214	35,43	64,57

Apêndice 7

Tabela ‘árvore’ para fator de pureza igual 0,7

nr_id	nó pai	condição	nó	condição	alvo	prob	freq1	freq2	Regra
1	null	null	itr	= t	C	1	2	0	1
2	null	null	itr	= f			232	242	
3	itr	= f	mo_armres	= 2			32	23	
4	mo_armres	= 2	ror	= t	P	1	1	0	2
5	mo_armres	= 2	ror	= f			32	22	
6	ror	= f	ius	= t	C	1	4	0	3
7	ror	= f	ius	= f			28	22	
8	ius	= f	rrb	= t	P	1	1	0	4
9	ius	= f	rrb	= f			28	21	
10	rrb	= f	rre	= t	C	1	2	0	5
11	rrb	= f	rre	= f			26	21	
12	rre	= f	rsz	= t	C	1	2	0	6
13	rre	= f	rsz	= f			24	21	
14	rsz	= f	mo_epxcar	= R	C	1	1	0	7
15	rsz	= f	mo_epxcar	= A			23	21	
16	mo_epxcar	= A	m1_dispon	<= 1080			12	16	
17	m1_dispon	<= 1080	rsc	= t	C	1	1	0	8
18	m1_dispon	<= 1080	rsc	= f			11	16	
19	rsc	= f	rtp	= t	P	1	3	0	9
20	rsc	= f	rtp	= f			11	13	
21	rtp	= f	mo_libcom	= O			10	13	
22	mo_libcom	= O	rrp	= t	P	1	1	0	10
23	mo_libcom	= O	rrp	= f			10	12	
24	rrp	= f	ivc	= t			7	5	
25	ivc	= t	mo_armban	= R			6	5	
26	mo_armban	= R	mo_armver	= A	C	1	2	0	11
27	mo_armban	= R	mo_armver	= R	P	0,555555556	4	5	12
28	ivc	= t	mo_armban	= A	C	1	1	0	13
29	rrp	= f	ivc	= f	P	0,7	3	7	14
30	rtp	= f	mo_libcom	= R	C	1	1	0	15
31	mo_epxcar	= A	m1_dispon	> 1080	C	0,6875	11	5	16
32	mo_epxcar	= A	m1_dispon	<= 720	P	0,6	8	12	17
33	mo_epxcar	= A	m1_dispon	> 720	C	0,625	15	9	18
34	itr	= f	mo_armres	= 1			48	26	
35	mo_armres	= 1	mo_armdes	= 1			47	26	
36	mo_armdes	= 1	rsb	= t	P	0,5	1	1	19
37	mo_armdes	= 1	rsb	= f	C	0,647887324	46	25	20
38	mo_armres	= 1	mo_armdes	= 2	C	1	1	0	21

Apêndice 8

Tabela ‘árvore’ para fator de pureza igual 0,8

nr_id	nó pai	condição	nó	condição	alvo	prob	freq1	freq2	Regra
1	null	null	itr	= t	C	1	2	0	1
2	null	null	itr	= f			232	242	
3	itr	= f	mo_armres	= 2			32	23	
4	mo_armres	= 2	ror	= t	P	1	1	0	2
5	mo_armres	= 2	ror	= f			32	22	
6	ror	= f	ius	= t	C	1	4	0	3
7	ror	= f	ius	= f			28	22	
8	ius	= f	rrb	= t	P	1	1	0	4
9	ius	= f	rrb	= f			28	21	
10	rrb	= f	rre	= t	C	1	2	0	5
11	rrb	= f	rre	= f			26	21	
12	rre	= f	rsz	= t	C	1	2	0	6
13	rre	= f	rsz	= f			24	21	
14	rsz	= f	mo_epxcar	= R	C	1	1	0	7
15	rsz	= f	mo_epxcar	= A			23	21	
16	mo_epxcar	= A	m1_dispon	<= 1080			12	16	
17	m1_dispon	<= 1080	rsc	= t	C	1	1	0	8
18	m1_dispon	<= 1080	rsc	= f			11	16	
19	rsc	= f	rtp	= t	P	1	3	0	9
20	rsc	= f	rtp	= f			11	13	
21	rtp	= f	mo_libcom	= O			10	13	
22	mo_libcom	= O	rrp	= t	P	1	1	0	10
23	mo_libcom	= O	rrp	= f			10	12	
24	rrp	= f	ivc	= t			7	5	
25	ivc	= t	mo_armban	= R			6	5	
26	mo_armban	= R	mo_armver	= A	C	1	2	0	11
27	mo_armban	= R	mo_armver	= R	P	0,555555556	4	5	12
28	ivc	= t	mo_armban	= A	C	1	1	0	13
29	rrp	= f	ivc	= f	P	0,7	3	7	14
30	rtp	= f	mo_libcom	= R	C	1	1	0	15
31	mo_epxcar	= A	m1_dispon	> 1080	C	0,6875	11	5	16
32	mo_epxcar	= A	m1_dispon	<= 720	P	0,6	8	12	17
33	mo_epxcar	= A	m1_dispon	> 720	C	0,625	15	9	18
34	itr	= f	mo_armres	= 1			48	26	
35	mo_armres	= 1	mo_armdes	= 1			47	26	
36	mo_armdes	= 1	rsb	= t	P	0,5	1	1	19
37	mo_armdes	= 1	rsb	= f	C	0,647887324	46	25	20
38	mo_armres	= 1	mo_armdes	= 2	C	1	1	0	21