Uma Ferramenta para Extração de Padrões

LEANDRO MACIEL ALMEIDA[1]
THEREZA PATRÍCIA P. PADILHA[1]
FERNANDO LUIZ DE OLIVEIRA[1]
CONCEIÇÃO A. PREVIERO[2]

[1]CEULP – Centro Universitário Luterano de Palmas/ULBRA
Curso de Sistemas de Informação
Cx. Postal 160 - CEP: 77.054-970 Palmas (TO)
{leandro, thereza, nandoluiz}@ulbra-to.br

[2]CEULP – Centro Universitário Luterano de Palmas/ULBRA
Curso de Engenharia Agrícola
previero@ulbra-to.br

Resumo: O processo de extração de conhecimento (padrões) de dados é uma técnica com a qual objetiva-se construir hipóteses (regras de produção e árvore de decisão, segundo o paradigma simbólico) para auxiliar tarefas tais como a tomada de decisão em uma empresa. Este processo é constituído de cinco etapas que serão apresentadas no decorrer deste artigo. Diante disso, esse trabalho tem como objetivo apresentar o desenvolvimento de uma ferramenta para extração de padrões a partir de uma base de dados. Para analisar o desempenho dessa ferramenta em relação aos outros sistemas de aprendizado conhecidos na literatura (J48.J48 e J48.PART, por exemplo), foi aplicada uma base de dados formada por características do fruto pequi para cada um deles. O intuito é gerar conhecimento de forma que, a partir de algumas características do fruto, tal como largura e espessura, possa-se classificá-lo em frutos com um, dois ou três caroços.

Palavras-chave: Extração de padrões, Teoria geral da informação.

1 Introdução

Várias mudanças vieram com a tecnologia, sendo que uma delas, refere-se ao aumento da importância da informação no mundo científico e comercial. A informação tornou-se um grande diferencial, auxiliando, por exemplo, no caso de uma empresa, desde a tomada de decisão até a descoberta de fraudes ou perfis de consumidores [Fayyad, 1996]. No entanto, para que estas informações sejam extraídas corretamente, é necessária a utilização de técnicas e ferramentas que propiciem a descoberta ou mineração de padrões [Mittchel, 1997].

Na literatura existem várias ferramentas para esse fim, que, como em qualquer outra área, possuem as suas vantagens e desvantagens. As ferramentas que disponibilizam um grande número de recursos são consideradas complexas e, em geral, possuem um alto custo. Uma outra limitação dessas ferramentas é não possibilitar a comunicação direta com um banco de dados e, como a maioria das empresas possui seus dados armazenados num conjunto de tabelas, é necessário realizar uma adaptação dessas tabelas para um modelo de arquivo especificado pela ferramenta.

Este trabalho está situado no campo de extração de conhecimento a partir de dados (*Knowledge Discovery in Database* - KDD) e tem como objetivo apresentar o desenvolvimento de uma ferramenta para extração de padrões usando um Sistema Gerenciador de Banco de Dados (SGBD). Para demonstrar o desempenho da ferramenta implementada são apresentados experimentos comparativos com a ferramenta e com dois sistemas de aprendizado do paradigma simbólico J48.J48 e J48.PART, disponíveis na ferramenta *Waikato Environment Knowledge Analysis*

(Weka) [Witten, 1999]. Nos experimentos, foi utilizada uma base de dados do fruto pequi encontrado no cerrado de alguns estados brasileiros, principalmente no estado do Tocantins.

Este artigo está organizado da seguinte forma: a seção 2 descreve uma visão geral do processo de extração de conhecimento de dados. A seção 3 apresenta uma descrição dos sistemas de aprendizado analisados: J48.J48 e J48.PART. Na seção 4 são fundamentados os métodos baseados na Teoria Geral da Informação (TGI), empregados na implementação da ferramenta. Na seção 5 é apresentada a ferramenta para extração de padrões implementada. A seção 6 apresenta algumas avaliações e resultados obtidos pelos sistemas de aprendizado e a ferramenta implementada, utilizando o conjunto de dados de pequi. Na seção 7 são descritos os trabalhos futuros. Por fim, a seção 8 mostra as considerações finais.

2 Processo de Extração de Conhecimento de Dados

O processo de extração de conhecimento de dados é constituído por um conjunto de etapas com a finalidade de obter um conhecimento a respeito de um determinado domínio, a partir de uma base dados em estado bruto [Fayyad, 2002]. As etapas do processo de extração de conhecimento são: seleção, pré-processamento, transformação, mineração de dados (*Data Mining* - DM) e Interpretação/Avaliação, como pode ser ilustrado na Figura 1.

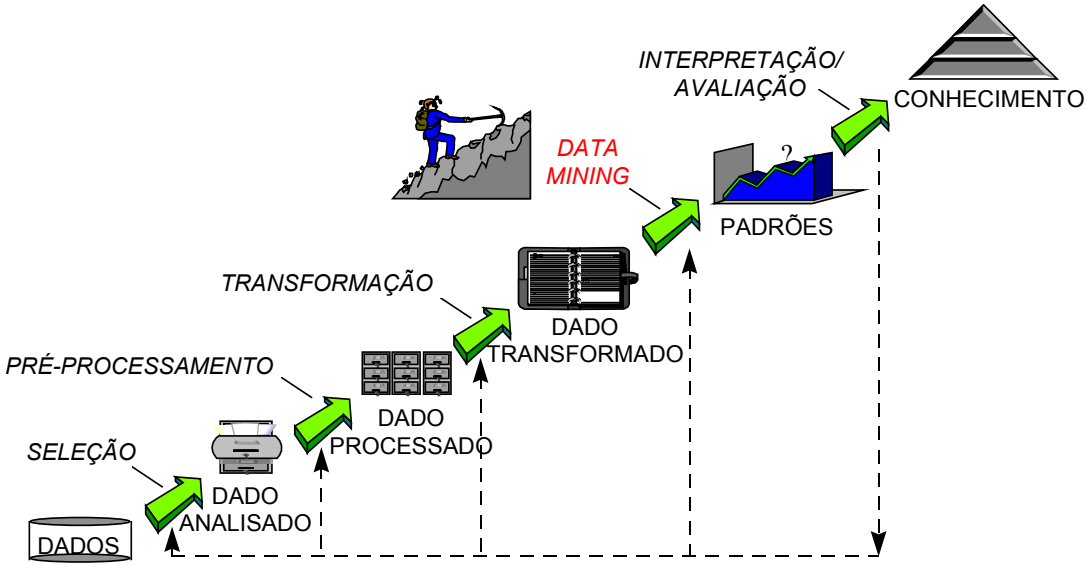

Figura 1: Etapas do processo de extração de conhecimento de dados [Fayyad, 2002].

2.1 Seleção

A etapa de seleção tem como objetivo selecionar um conjunto de dados, pertencentes a um domínio, para que, a partir de um critério definido pelo especialista do domínio, possa ser analisado.

2.2 Pré-processamento

Após a seleção dos dados, inicia-se a etapa de pré-processamento dos dados. Nesta etapa deverão ser realizadas tarefas que eliminem ou tratem os ruídos ou registros com dados ausentes. Outra tarefa importante é a verificação de predominância de classes, sendo que nestes casos, deve-se eliminar alguns dos registros da classe predominante ou acrescentar registros das outras classes. O objetivo é balancear a base de dados de tal forma que, no processo do aprendizado, uma certa classe não seja favorecida, evitando que o sistema fique tendencioso.

Para que a base de dados possa se tornar consistente, faz-se necessário que os ruídos e os dados ausentes sejam eliminados ou, com o auxílio do especialista do domínio, tratados/recuperados. Ruídos referem-se a situações em que dois ou mais registros, composto por atributos que possuem os mesmos valores e que chegam a classes distintas. Já dados ausentes, correspondem a registros que não possuem todos os valores dos atributos preenchidos. Em ambas situações, os registros redundantes ou mal formados devem ser eliminados, ou modificados, de tal forma que tenham

a mesma classe ou todos seus valores preenchidos, respectivamente. A presença do especialista do domínio nesta etapa é de extrema relevância.

2.3 Transformação

Após serem pré-processados, os dados necessitam ser armazenados e formatados adequadamente, para que os algoritmos de aprendizado possam ser aplicados. Esta etapa, por algum tempo, foi considerada como um obstáculo, porque o usuário (engenheiro de conhecimento) tinha que ficar formatando os dados, ou seja, adaptando-os em arquivos necessários para a execução de cada algoritmo de aprendizado escolhido.

2.4 Data Mining

Esta etapa, por sua vez, tem como objetivo construir hipóteses (considerando o paradigma simbólico: regras de produção e árvores de decisão). Para isto, existem diversos algoritmos ou sistemas de aprendizado, tais como: C4.5, CN2, *AutoClass* e BKD. Um sistema de aprendizado é um programa de computador que toma decisões baseadas em experiências acumuladas contidas em exemplos resolvidos com sucesso [Weiss, 1991]. Além disso, existem algumas ferramentas de *data mining* que implementam inúmeros algoritmos com o intuito de facilitar a execução de vários algoritmos a partir de um mesmo formato de arquivo, que são: a biblioteca MLC++ (*Machine Learning Library* in C++) [Kohavi, 1997] e WEKA [Witten, 1999].

2.5 Interpretação/Avaliação

Durante esta etapa, o conhecimento adquirido (por exemplo, árvores de decisão e regras de produção) será analisado. Para que esta análise seja feita corretamente, é fundamental que esta etapa seja realizada em conjunto com o(s) especialista(s) do domínio. Caso o conhecimento gerado não seja satisfatório, o processo de extração de conhecimento pode ser recomeçado (como pode ser observado na Figura 1), modificando o conjunto de dados selecionado ou alterando os parâmetros utilizados na ferramenta de DM, por exemplo.

3 Sistemas de Aprendizado Analisados

Weka é uma ferramenta de *data mining* com uma crescente utilização que foi escolhida por várias razões: encontra-se disponível na Internet (http://www.cs.waikato.ac.nz/~ml/weka/), facilidade de instalação e implementação feita em Java torna o sistema portável. A ferramenta Weka é formada por um conjunto de pacotes, que são: *attribute selection* (seleção de atributos de uma base de dados), *classifiers* (implementação de algoritmos de aprendizagem supervisionada), c*lustering* (implementação de algoritmos de aprendizagem não supervisionada) e *filters* (seleção de instâncias de uma base).

Essa ferramenta dispõe também de métodos de meta aprendizagem, que são utilizados para a construção de conjuntos de classificadores. Dentre os métodos disponíveis, destacam-se: *Bagging* e *Boosting*. O método *Bagging* constrói os classificadores a partir de sucessivos e independentes conjuntos de amostras de dados. Estas amostras são geradas a partir de um conjunto de dados de treinamento. Randomicamente são extraídas *n* instâncias com substituição a partir do conjunto original, ou seja, ocorrendo à repetição de instâncias nas amostras. Por outro lado, no método *Boosting*, cada instância de treinamento tem um certo peso associado. Ao induzir o primeiro classificador todas as instâncias são equiprováveis, isto é, tem o mesmo peso. Após ter induzido o primeiro classificador, os pesos das instâncias de treinamento classificadas incorretamente são alterados baseado nos classificadores que foram construídos anteriormente [Witten, 1999].

Para este trabalho, foram selecionados dois algoritmos de classificação, que são: J48.J48 e J48.PART. Uma breve descrição desses algoritmos é apresentada a seguir:

3.1 J48.J48

O algoritmo J48.J48 é uma implementação do algoritmo C4.5 *release* 8 que, gera árvore de decisão (última publicação da família de algoritmos que geram árvores de decisão antes do C5.0, versão mais recente e disponível apenas comercialmente) e é considerado o mais popular algoritmo da Weka. O J48.J48 constrói um modelo de árvore de decisão baseado num conjunto de dados de treinamento, sendo que esse modelo é utilizado para classificar as instâncias do conjunto de teste.

Durante o processo de utilização do algoritmo J48.J48 é interessante conhecer alguns parâmetros que podem ser modificados para proporcionar melhores resultados, como, por exemplo, o uso de podas na árvore, o número mínimo de instâncias por folha e a construção de árvore binária.

3.2 J48.PART

O algoritmo J48.PART é uma variação do J48.J48 que constrói regras de produção a partir da árvore de decisão. O processo de geração de regras de produção atua em dois estágios: regras são induzidas inicialmente de uma árvore e posteriormente são refinadas. Para cada regra criada é estimada a cobertura das instâncias da base. Isso ocorre repetidamente até que todas as instâncias estejam cobertas. As regras com coberturas mais altas são apresentadas para o usuário e as demais são descartadas.

4 Teoria Geral da Informação

A Teoria Geral da Informação surgiu com pesquisas no campo da telegráfica e telefonia realizadas por Claude E. Shannon e Warrem A. Weaver, com ênfase na engenharia da transmissão da cadeia de mensagens de baixo custo e alta confiabilidade [Ferreira, 2002]. Foram desenvolvidos vários métodos para medir e calcular a quantidade de informação, dentre esses métodos destaca-se a entropia. Baseado no cálculo da entropia, surgiram os métodos *Gain*, *Split* e *Gain Ratio* voltados para obter o ganho da informação.

A entropia é uma grandeza que mede a desordem, tanto de objetos físicos quanto de informações. Quanto maior o grau da entropia maior é a desordem e, quanto menor o grau da entropia melhor a organização. Sendo assim, o grau de entropia mais apropriado é o de menor valor [Ferreira, 2002]. O cálculo da entropia é obtido através da seguinte fórmula:

$$E(A = v_j) = -\sum_{i=1}^{n} p(i) \times \log_2(p(i))$$

na qual $A = v_j$ significa que o atributo A tem o valor v_j, n é o número de classes diferentes $c_1,c_2,...,c_n$ e $p(i)$ é a probabilidade de um registro pertencer à classe c_n. Após realizar o cálculo com todos os valores possíveis de um atributo, deve-se relacionar todos esses valores para encontrar o ganho de informação de um atributo a partir da seguinte fórmula:

$$Gain(D,T) = Info(D) - \sum_{i=1}^{k} \frac{|Di|}{|D|} \times Info(Di)$$

na qual *Info(D)* representa a entropia da base como um todo, k denota a quantidade de subconjuntos do atributo T, *Info(Di)* denota a entropia de cada subconjunto $D_1,D_2, D_3,...,D_N$, D_i denota a quantidade de exemplos de cada subconjunto e o D a quantidade de exemplos contidos na base. O melhor ganho de informação é determinado pelo maior valor encontrado em relação aos demais valores.

Apesar do método *Gain* apresentar bons resultados, os atributos com muitos valores são favorecidos, já que a maioria, ou todos subconjuntos gerados, irá possuir apenas uma classe, e com E(X)=0 o ganho da informação por particionar esse conjunto será máxima, sendo isto uma inverdade. Para contornar esse problema é necessário calcular o S*plit Info* com a seguinte fórmula:

$$Split(D,T) = -\sum_{i=1}^{k} \frac{|Di|}{|D|} \times \log_2\left(\frac{|Di|}{|D|}\right)$$

na qual k denota a quantidade de subconjuntos do atributo T, Di representa a quantidade de exemplos de cada subconjunto de T e D representa a quantidade de exemplos contidos na tabela. Por fim, é necessário aplicar a seguinte formula:

$$GainRatio(D,T) = \frac{Gain(D,T)}{Split(D,T)}$$

com isso tem-se o completo cálculo do ganho de informação do atributo T.

5 Ferramenta para Extração de Padrões Implementada

Devido ao fato de que alguns sistemas de aprendizado possuem certas limitações para a extração de padrões, uma ferramenta foi implementada utilizando recursos alternativos para tornar o processo parcialmente mais simples. Essa ferramenta possibilita a comunicação com uma base de dados situada num SGBD e aplica métodos da TGI. A junção de todos esses métodos resulta no completo e mais coerente possível cálculo do ganho de informação, o *Gain Ratio,* responsável por revelar a quantidade de informação relevante em um determinado atributo [Quinlan, 1996].

A ferramenta implementada utiliza como técnica de *Data Mining* a indução de regras, para que o processo de extração de padrões obtenha um resultado coerente em relação à tabela disponibilizada. É necessário, inicialmente, efetuar o cálculo da entropia do atributo classe, o qual classifica os registros de acordo com as informações existentes nos demais atributos. Como dito anteriormente, a entropia mede a desordem, portanto quanto maior o grau da entropia maior é a desordem. Sendo assim, o melhor grau de entropia é o que possui valor mais baixo. Depois de encontrar o valor da entropia, todas as fórmulas apresentadas na seção 4 são utilizadas para obter o valor do ganho de informação.

Para a utilização dos métodos apresentados é necessário identificar *a priori* qual o tipo de dado definido em cada coluna da tabela especificada, ou seja, se o dado é do tipo discreto ou numérico (contínuo). Como o MySQL é o SGBD utilizado pela ferramenta, esta identificação é realizada de acordo com o tipo especificado na criação da tabela no SGBD, caso seja do tipo TINYINT, SMALLINT, MEDIUMINT, INT, BIGINT, FLOAT, DOUBLE ou DECIMAL é considerado um atributo do tipo contínuo, caso contrário é discreto.

Para o tratamento de dados discretos utilizou-se a criação dinâmica de sentenças *Structured Query Language* (SQL) para o agrupamento de exemplos e identificação de subconjuntos de cada atributo da tabela. Quanto aos dados contínuos, são necessários realizar certos passos para definir a discretização dos mesmos, ou seja, para o cálculo do ganho de informação de um atributo contínuo necessita-se identificar o melhor *threshold* (limiar) contido no atributo. Os atributos contínuos possuem um grande diferencial em relação aos discretos, pois os contínuos apresentam um conjunto extensivo de valores possíveis.

Sendo o atributo contínuo A pertencente a uma tabela D, o processo para encontrar o melhor limiar inicia-se com a ordenação dos exemplos nos valores do atributo A. Para cada valor do atributo A, situado em uma transição de classe, é aplicada a seguinte fórmula [Galiano, 2002]:

$$t_i = \max\left\{ v \mid v \leq \frac{v_i + v_{i+1}}{2} \right\}$$

na qual t_i representa um limiar possível do atributo, v_i denota um exemplo contido no atributo e o V_{i+1} representa o seu conseqüente.

Após encontrar todos os limiares candidatos para o atributo A, a escolha do melhor limiar é feita utilizando as duas primeiras fórmulas apresentadas nesta seção, isto é, para cada valor candidato é calculado o ganho de informação e, o valor que apresentar o maior ganho é o limiar escolhido para discretizar o atributo A. Dependendo da organização dos exemplos de um atributo contínuo, podem ser detectados diversos valores candidatos, podendo ocasionar pequenos subconjuntos, o que não é aconselhável, pois esses pequenos subconjuntos apresentam um bom ganho de informação em relação aos demais, o que não representa o melhor limiar a ser utilizado para a identificação de padrões. Para sanar este problema, é necessário forçar o número mínimo de valores candidatos possíveis.

De um modo geral, para os dois tipos de atributos, a fase posterior refere-se ao processo de construção da árvore de decisão, partindo da identificação do melhor atributo. Os valores únicos existentes nesse atributo, juntamente com os valores do atributo classe serão utilizados para identificar se existe um padrão no nível atual de detalhe, caso isso não seja comprovado pela aplicação, realiza-se então uma nova busca por um atributo que melhor revele o padrão existente, utilizando como restrição o valor do melhor atributo e o valor respectivo do atributo classe. Todo esse processo é baseado no algoritmo dividir para conquistar [Hunt, 1966]:

Tendo um conjunto de treinamento T, composto pelas classes $\{c_1, c_2,...,c_n\}$, existem três possibilidades para T:

- 1ª: T não contém exemplos, procura-se por outras informações no pai desse nó;
- 2ª: T contém um ou mais exemplos, todos pertencendo a uma mesma classe c_y, a árvore de decisão para T é uma folha identificando a classe c_y;

- 3ª: T contém vários exemplos, pertencendo a uma miscelânea de classes. O conjunto T é dividido em subconjuntos que se classifiquem nas possibilidades citadas anteriormente. Essa divisão é efetuada com base em um atributo que possua valores mutuamente exclusivos $\{v_i, v_2, v_3,...,v_n\}$. T é particionado em subconjuntos $T_1, T_2, T_3,...,T_n$, no qual T_i contém todos os casos com valores v_i. A árvore de decisão para T consiste de um nó de decisão identificando o teste para o atributo e um galho para cada valor do atributo. Recursivamente cada subconjunto T_i é visto como T, que irá se encaixar em uma das três possibilidades.

No terceiro passo do algoritmo utilizado, a escolha pelo atributo que possua valores mutuamente exclusivos é realizada utilizando todas as fórmulas descritas nesta seção. Essas etapas são realizadas de forma recursiva, a cada finalização de construção de um ramo da árvore é verificado se para este é possível realizar a poda, com intuito de eliminar redundâncias, ao contrário dos demais sistemas de extração de padrões, que realizam a poda somente no término da construção da árvore [Eloamaa, 2001].

A árvore de decisão gerada pela ferramenta serve de base para a construção das regras de produção, que são armazenadas em um arquivo. Cada regra possui uma porcentagem indicando o quanto àquela regra é coerente e o quanto essa regra possui prioridade em relação à quantidade de exemplos que ela acopla.

A implementação da ferramenta foi realizada com a linguagem de programação JAVA, pelo fato de oferecer um *driver* simples/funcional de conexão a uma base de dados e objetos que implementam listas dinâmicas [Deitel, 2001]. O processo de extração de padrões consiste na conexão da ferramenta a uma base de dados, através de classes do *Java Database Connectivity* (JDBC). A ferramenta obtém todos os metadados necessários para o processo, realiza a formulação de sentenças SQL para coleta, classificação e posterior manipulação dos dados na memória. Em toda a extensão da ferramenta são utilizadas listas dinâmicas para suportar o trabalho com uma base de dados, que pode de alguma forma sofrer atualizações.

A ferramenta desenvolvida foi baseada no sistema de aprendizado C4.5 [Quinlan, 1993], porém há diferenças consideráveis no tratamento de dados discretos e contínuos, bem como na metodologia para identificar outros atributos necessários para construção de uma árvore consistente. Além dessas modificações/melhoramentos na ferramenta, houve uma redução de etapas, necessárias para a construção da árvore de decisão e formalização de regras, em conseqüência da previsão da continuação do processo de avaliação de atributos importantes para a construção da árvore. Algumas melhorias podem ser adicionas, como a extensão para o tratamento de dados do tipo DATE e binário, além de desenvolver uma interface visual que forneça uma melhor utilização da ferramenta.

6 Experimentos Realizados

Os experimentos realizados com os algoritmos descritos anteriormente e com a ferramenta implementada têm como objetivo investigar seus comportamentos usando um conjunto de dados do pequi. Nesta seção, as etapas que compõem o processo KDD serão aplicadas para esse domínio.

6.1 Descrição do Domínio da Base de Dados

O domínio escolhido baseia-se em um conjunto de variáveis que descrevem as dimensões de um fruto conhecido como pequi, possibilitando que ele possa ser classificado em um, dois ou três caroços. O fruto pequi (*Caryocar brasiliense* Camb.) é do tipo *drupaceo* e, normalmente, contém de um até quatro caroços ou *putamens*. O mesocarpo interno de cor amarela, branca ou rósea é comestível e é o verdadeiro atrativo da planta. No interior dos estados de Goiás, Minas Gerais, Mato Grosso e Tocantins a importância deste fruto na dieta das pessoas, que lá habitam, cresce substancialmente, segundo [Previero, 2000]. Além de sua importância alimentar, o pequi é uma planta totalmente aproveitada como, por exemplo, o uso da casca e folhas para a extração de corantes [Almeida, 1994].

Para a construção da base de dados, diversos frutos foram colhidos e analisados no laboratório de Pré-processamento do Curso de Engenharia Agrícola do CEULP/ULBRA, tendo como características físicas avaliadas o comprimento, largura, espessura, volume, diâmetro equivalente e esfericidade. Estas características serão posteriormente descritas mais detalhadamente. No geral, foram selecionados 183 registros (instâncias), divididos em 3 classes possíveis. O critério adotado para a análise destes dados consiste nos valores que um conjunto de variáveis (atributos) devem apresentar para que o registro possa ser classificado em uma destas classes.

6.2 Análise da Base de Dados

A distribuição dos 183 registros que compõem a base de dados em relação às três classes são: 61 registros pertencem a classe 1, 61 a classe 2 e 61 a classe 3. Cada registro é formado por seis atributos e uma classe, que estão descritos na Tabela 1.

Nome do Atributo	Descrição	Valores Possíveis
Comp (mm)	Indica o comprimento do fruto	Contínuo
Larg (mm)	Indica a largura do fruto	Contínuo
Espes (mm)	Indica a espessura do fruto	Contínuo
Deq (mm)	Indica o diâmetro equivalente	Contínuo
Vol (g/ml)	Indica o volume	Contínuo
Esferic	Indica a esfericidade	Contínuo
Classe	Identifica o número de caroços do fruto	1, 2 e 3

Tabela 1: Descrição dos atributos da base de dados

Para a base de dados selecionada, não foi necessário realizar nenhum tratamento de ruídos, dados ausentes ou balanceamento dos registros em relação a uma classe porque esses "problemas" não foram identificados.

6.3 Formatação dos Dados

Neste trabalho, foi adotado e adaptado o paradigma de estimativa de erro *Holdout*, sendo 65% dos dados para treinamento e 35% para teste. A divisão dos registros em conjuntos de treinamento e de teste foi feita pelo próprio engenheiro de conhecimento de forma aleatória. Sendo assim, 118 registros foram selecionados para treinamento e 65 para teste.

Para a ferramenta Weka, foram criados dois arquivos (um para treinamento e outro para teste), esses arquivos são formados pelo nome e os valores possíveis de cada um dos atributos (cabeçalho) e pelos registros de cada conjunto.

A ferramenta implementada exige para o seu correto funcionamento que o especialista do domínio informe qual coluna da tabela, contida em uma base de dados, será o atributo classe, caso não seja informado a ferramenta assume como atributo classe a última coluna da tabela especificada. Na experiência realizada neste trabalho utilizou-se uma tabela contida no MySQL, um SGBD gratuito, neste caso é necessário informar para a ferramenta a base de dados e a tabela. Em alguns SGBDs é preciso criar um *Data Source Name* (DSN) no sistema operacional Windows, em outros bancos de dados não existe a necessidade de criação de um DSN, pois os *drivers* de conexão/manipulação desses bancos provêem o correto funcionamento, independente da plataforma.

6.4 Resultados Obtidos

Com relação aos resultados obtidos, vale salientar que os algoritmos J48.J48 e J48.PART foram executados com seus valores padrões para cada parâmetro disponível. Estes algoritmos foram aplicados separadamente, chegando, portanto, a um determinado resultado. Após estes experimentos foram utilizados os métodos de meta aprendizagem (*Bagging* e *Boosting*), associados com estes mesmos algoritmos, na tentativa de se chegar a melhores resultados. Tanto os algoritmos quanto à ferramenta proposta receberam, como entrada, o mesmo conjunto de registros para treinamento e teste.

Dependendo do algoritmo e método usados, as regras de produção e árvore de decisão geradas sofrem alterações, com relação ao número de regras e número de níveis da árvore. Este trabalho terá enfoque maior na análise das matrizes de confusão obtidas, mas serão apresentadas algumas regras de produção geradas. O objetivo da análise das matrizes é verificar a precisão das hipóteses geradas (regras de produção e árvores de decisão) utilizando as instâncias não vistas durante o processo de treinamento, ou seja, o conjunto destinado para o teste.

As tabelas seguintes apresentam os valores das matrizes de confusão obtidas aplicando o algoritmo J48.PART separadamente e em conjunto com os métodos de meta aprendizagem. Os valores das taxas de erro apresentados foram obtidos da própria matriz de confusão.

Matriz de Confusão			
C1asse 1	**Classe 2**	**Classe 3**	
19	0	*1*	**Classe 1**
0	14	*12*	**Classe 2**
0	*4*	15	**Classe 3**
Legenda:	**Acertos** | ***Erros***		

Taxa de Erro		
Classes	**Acertos**	**Erros**
Classe 1	95%	5%
Classe 2	54%	46%
Classe 3	79%	21%
Geral	74%	26%

Tabela 2: Matriz de confusão apresentada usando o algoritmo J48.PART

Matriz de Confusão			
C1asse 1	**Classe 2**	**Classe 3**	
19	0	*1*	**Classe 1**
0	18	*8*	**Classe 2**
0	*5*	14	**Classe 3**
Legenda:	**Acertos** | ***Erros***		

Taxa de Erro		
Classes	**Acertos**	**Erros**
Classe 1	95%	5%
Classe 2	69%	31%
Classe 3	73%	27%
Geral	80%	20%

Tabela 3: Matriz de confusão apresentada usando o método *Bagging* com o algoritmo J48.PART como classificador

Matriz de Confusão			
C1asse 1	**Classe 2**	**Classe 3**	
19	*1*	0	**Classe 1**
0	20	*6*	**Classe 2**
0	*3*	16	**Classe 3**
Legenda:	**Acertos** | ***Erros***		

Taxa de Erro		
Classes	**Acertos**	**Erros**
Classe 1	95%	5%
Classe 2	77%	23%
Classe 3	84%	16%
Geral	84%	16%

Tabela 4: Matriz de confusão apresentada usando o método *Boosting* com o algoritmo J48.PART como classificador

Com relação aos números apresentados pôde-se verificar que a utilização dos métodos de meta aprendizagem diminuiu sensivelmente a taxa de erro geral, passando de 26% (apenas o algoritmo) para 16% (*Boosting* + algoritmo). Os resultados obtidos da mesma forma que os apresentados pela utilização do J48.PART melhoraram com a utilização dos métodos de meta aprendizagem, mas especificamente com o método *Boosting*, com uma taxa de erro de 14%.

A quantidade de regras geradas em cada experimento foi: 7 (J48.PART), 7 (J48.PART com o método *Boosting*) e 10 (J48.PART com o método *Bagging*). Na Tabela 8 é apresentado um dos conjuntos de regras de produção gerado pelo algoritmo J48.PART usando o método *Boosting*. Os valores apresentados entre parênteses representam os pesos de instâncias classificadas corretamente e incorretamente (em negrito), respectivamente. O valor da classe (1, 2 ou 3), que correspondem ao número de caroços no fruto, em cada regra é apresentado após o símbolo de dois-pontos (:).

Ordem	Regras
1	comp > 7.8 AND espess <= 8.45 AND esferic > 0.82 AND vol > 149: 2 (81.99/***5.64***)
2	deq > 5.924 AND vol > 124.6 AND deq <= 6.655: 3 (27.18/***0.86***)
3	comp <= 7.195 AND larg > 4.785: 2 (17.5)
4	vol > 84.3 AND comp > 7.3 AND esferic > 0.86 AND larg <= 7.515: 3 (34.54/***2.94***)
5	comp > 7.3 AND espess > 6.67: 2 (29.59/***1.71***)
6	deq > 5.882: 3 (21.53/***0.05***)
Default	:1(2.68/***0.97***)

Tabela 8: Regras de produção geradas pelo J48.PART com o método *Boosting.*

As matrizes geradas pela utilização do algoritmo J48.J48, bem como suas análises serão apresentadas a seguir.

Matriz de Confusão			
C1asse 1	**Classe 2**	**Classe 3**	
19	0	*1*	**Classe 1**
0	16	*10*	**Classe 2**
0	*4*	15	**Classe 3**
Legenda:	**Acertos** | ***Erros***		

Taxa de Erro		
Classes	**Acertos**	**Erros**
Classe 1	95%	5%
Classe 2	61%	39%
Classe 3	79%	21%
Geral	77%	23%

Tabela 5: Matriz de confusão apresentada usando o algoritmo J48.J48

Matriz de Confusão			
C1asse 1	**Classe 2**	**Classe 3**	
19	*1*	0	**Classe 1**
0	19	*7*	**Classe 2**
0	*4*	15	**Classe 3**
Legenda:	**Acertos** | ***Erros***		

Taxa de Erro		
Classes	**Acertos**	**Erros**
Classe 1	95%	5%
Classe 2	73%	27%
Classe 3	79%	21%
Geral	81%	19%

Tabela 6: Matriz de confusão apresentada usando o método *Bagging* com o algoritmo J48.J48 como classificador

Matriz de Confusão			
C1asse 1	**Classe 2**	**Classe 3**	
19	*1*	0	**Classe 1**
0	20	*6*	**Classe 2**
0	*2*	17	**Classe 3**
Legenda:	**Acertos** | ***Erros***		

Taxa de Erro		
Classes	**Acertos**	**Erros**
Classe 1	95%	5%
Classe 2	77%	23%
Classe 3	89%	11%
Geral	86%	14%

Tabela 7: Matriz de confusão apresentada usando o método *Boosting* com o algoritmo J48.J48 como classificador

Com relação aos valores apresentados utilizando o algoritmo J48.J48, pode-se verificar que as taxas de erro gerais têm valores menores quando são usados os métodos de meta aprendizagem. Verifica-se também que, para o domínio estudado, o conhecimento produzido pela utilização do método *Boosting*, tendo o algoritmo J48.J48 como classificador, atingiu o melhor resultado entre os apresentados.

O tamanho da árvore e o número de folhas geradas em cada experimento foram: 11 e 6 (J48.J48), 25 e 13 (J48.J48 com o método *Boosting*) e 21 e 11 (J48.J48 com o método *Bagging*). Na Figura 2 é apresentada a árvore gerada pelo algoritmo J48.J48. Assim como em regras de produção, os valores entre parênteses que aparecem nas folhas da árvore representam os pesos de instâncias classificadas corretamente e incorretamente (negrito), respectivamente. A partir dessa árvore, por exemplo, pode-se verificar que existe um custo adicional para definir o padrão de pequi com 2 e 3 caroços, pois necessita de um relacionamento entre vários atributos.

```
espess <= 5: 1 (58.0)
espess > 5
|   esferic <= 0.89
|   |   vol <= 187.5
|   |   |   espess <= 8.05: 3 (45.0/16.0)
|   |   |   espess > 8.05: 2 (10.0)
|   |   vol > 187.5
|   |   |   deq <= 7.219: 2 (4.0/1.0)
|   |   |   deq > 7.219: 3 (24.0)
|   esferic > 0.89: 2 (42.0/9.0)
```

Figura 2: Árvore gerada pelo algoritmo J48.J48.

Com relação aos experimentos realizados com a ferramenta implementada, observou-se que a classe 1 apresentou uma excelente detecção do padrão, atingindo a taxa de 100% de acerto, sendo considerada a maior entre todas as taxas encontradas pelos classificadores em suas diferentes configurações. A classe 3 apresentou também uma taxa de acerto consideravelmente boa, atingindo 85%, permanecendo na média dos classificadores. Por sua vez, na classe 2, nota-se um aumento da taxa de erro em relação aos demais, ficando um pouco acima da média. Na realidade, no fruto pequi da classe 2 para a identificação da quantidade de caroços é mais difícil, porque é necessário ponderar várias medidas e também por não possuir uma característica marcante.

De um modo geral, a taxa de acerto total da ferramenta permaneceu próxima da média encontrada pelos outros classificadores, obtendo até a melhor taxa de acerto. No entanto, quando a ferramenta é comparada aos classificadores utilizando o método *Bagging* ou *Boosting*, pode se observar que o seu desempenho tem regredido.

Matriz de Confusão			
C1asse 1	**Classe 2**	**Classe 3**	
20	*0*	*0*	**Classe 1**
0	8	*10*	**Classe 2**
0	*4*	23	**Classe 3**
Legenda:	**Acertos** | ***Erros***		

Taxa de Erro		
Classes	**Acertos**	**Erros**
Classe 1	100%	0%
Classe 2	44%	56%
Classe 3	85%	15%
Geral	79%	21%

Tabela 4: Matriz de Confusão apresentada usando a ferramenta implementada

O número de nós-folha e o tamanho da árvore de decisão gerada pela ferramenta foram 5 e 8, respectivamente, como pode ser observado na Figura 3. Nota-se uma menor quantidade de níveis e nós-folha em relação ao algoritmo J48.J48 ilustrado na Figura 2. A ferramenta apresentou uma menor quantidade de nós-folha devido às políticas de paradas adotadas, que se encontram ainda em fase de aprimoramento. Seguindo o mesmo padrão, os valores apresentados em cada nó-folha da árvore (Figura 3) são o número de exemplos classificados corretamente e incorretamente (negrito). A partir desta árvore nota-se que o custo adicional para definir o padrão de pequi com 2 e 3 caroços é menor em relação a árvore da Figura 2.

```
espess<=5.0-->1.0(58.0)
espess>5.0
    esferic<=0.89
        vol<=187.5
            espess<=7.625 --> 3 (34.0/9.0)
            espess>7.625 --> 2.0(21.0/4.0)
        vol>187.5 --> 3.0(28.0/3.0)
    esferic>0.89 --> 2.0(42.0/9.0)
```

Figura 3: Árvore gerada pela ferramenta implementada.

7 Trabalhos Futuros

Como mencionado anteriormente, a ferramenta implementada possui alguns melhoramentos a serem feitos, assim para trabalhos futuros serão pesquisadas e estudadas novas políticas de paradas para a construção da árvore como também otimizar o trabalho da ferramenta em base de dados de médio a grande porte. Para isso será desenvolvida a adaptação da ferramenta para trabalhar com dados DATE e binário de forma a utilizar com eficiência as informações contidas nesses tipos de dados, ou seja, dados do tipo DATE. Normalmente, esses dados são compostos por dia, mês e ano, logo existem mais informações úteis encapsuladas em um dado.

Com o objetivo de utilizar a ferramenta implementada para a extração de padrões de bases de dados, será desenvolvida uma interface visual também em JAVA. Essa interface conterá uma organização de componentes visuais para iniciantes no KDD bem como um *help* para auxiliar o manuseio da ferramenta. Pelo fato de que a versão atual da ferramenta é uma API (*Application Program Interface*), a adição de componentes visuais possibilitará a utilização mais eficiente das funções por usuários no KDD.

Ainda como trabalho futuro será realizada a integração da ferramenta ao ambiente de aprendizado colaborativo RESOLVE que está sendo desenvolvido no Laboratório de Inteligência Computacional (LINC) do Centro Universitário Luterano de Palmas (CEULP/ULBRA) [Padilha, 2003]. Essa integração tem por objetivo realizar uma

avaliação do aprendizado de grupos de alunos no processo de aprendizagem, sendo utilizada também a tecnologia de agentes inteligentes, assim os passos do processo de extração de dados são realizados por um agente inteligente [Almeida, 2003].

8 Considerações Finais

Este artigo apresentou a implementação de uma ferramenta para a extração de padrões, bem como experimentos com a ferramenta implementada e com os algoritmos de classificação J48.J48 e J48.PART disponíveis na ferramenta Weka, para a extração de conhecimento de dados do fruto pequi. Este trabalho serviu para consolidar os conceitos envolvidos no processo KDD e verificar a viabilidade de extrair conhecimento em áreas que possuem um grande volume de dados. A partir deste trabalho foi adquirida uma habilidade em trabalhar com a ferramenta Weka, bastante utilizada no meio acadêmico. Essa ferramenta possibilita uma facilidade em realizar experimentos distintos rapidamente, porém a sua performance é deteriorada com o uso de grandes conjuntos de dados, pois requer muita memória.

Com relação aos experimentos realizados, pôde-se observar que as hipóteses geradas, na aplicação dos algoritmos e dos métodos, foram válidas visto os percentuais de erro apresentados. De um modo geral, o melhor resultado para domínio estudado refere-se ao método *Boosting*, aplicado com o algoritmo J48.J48. Além disso, o especialista do domínio detectou uma grande consistência das regras e árvores geradas pela ferramenta Weka na identificação de caroços do fruto pequi.

A ferramenta implementada apresentou um bom desempenho em relação aos demais algoritmos, porém existe a necessidade de buscar um aperfeiçoamento para a condição de parada da identificação de padrões, para que haja uma melhor identificação de novos atributos relevantes para o crescimento ou não da árvore. Além dessa melhoria, existe também o intuito de aplicar os melhoramentos descritos na seção 4, assim como buscar a aplicação de novos métodos de *data mining* ou aprimoramento com novas tecnologias disponíveis.

As imperfeições existentes na ferramenta podem e devem ser corrigidas, porém a mesma apresenta um diferencial em relação ao gerenciamento das regras encontradas a partir da árvore de decisão. Isso se torna mais visível para uma aplicação da tecnologia de agentes inteligentes, que precisam controlar o histórico dos acontecimentos encontrados [Wilkins, 2003]. Para isso está em desenvolvimento o armazenamento das regras em uma tabela contida na mesma base de dados. Como cada regra possui um valor, que é utilizado para criar um *ranking* e ainda utilizando as sentenças SQL, o processo de manipulação é facilitado de forma consideravelmente alta.

Os resultados encontrados com a ferramenta implementada possuem algumas incorreções, pois ainda não está totalmente definida e requer certos aperfeiçoamentos. Isso ocorre porque o processo de conhecimento sobre técnicas de *data mining* é longo, sendo necessário realizar buscas constantes no cenário nacional e/ou internacional sobre essas técnicas.

9 Referências Bibliográficas

[Almeida, 1994] Almeida, S. P. & Silva, J.A. “Pequi e buriti – importância alimentar para a população dos Cerrados”. Planaltina: EMBRAPA-CPAC, 38p, 1994.

[Almeida, 2003] Almeida, L.M. & Padilha, T.P.P. “Um Modelo do Aprendizado de Grupos de Alunos em Ambientes Colaborativos”. V Encontro de Estudantes de Informática do Estado do Tocantins, p.361-370, Palmas, 2003.

[Deitel, 2001] Deitel, H. M. & Deitel, P.J. “Java Como Programar”. Bookman, Porto Alegre, 2001.

[Eloamaa, 2001] Eloamaa, T. & Kääriäinen, M. “An Analysis of Reduced Error Pruning”. Journal of Artificial Intelligence Research: Volume 15, p. 163-187, 2001

[Ferreira, 2002] Ferreira, V. “Teoria Geral dos Sistemas”. IPEP: São Paulo, 2002.

[Fayyad, 1996] Fayyad, U., Shapiro, G.P. & Smyth, P. “From Data Mining to Knowledge Discovery in Databases”. AAAIMIT Press: p.37-54, 1996.

[Fayyad, 2002] Fayyad, U. M. & Uthurusamy, R. “Evolving Data Mining into Solutions for Insights”, Comminications of the ACM: vol.45, Nº 8, p. 28-31, 2002.

[Galiano 2002] Galiano, F. B. “ART Un Método Alternativo para la Construcción de Árboles de Decisión”, Universidad de Granada E.T.S. Ingeniería Informática, Tesis Doctoral, 2002.

[Hunt , 1966] Hunt, E. B., Marin, J., & Stone, P. J. “Experiments in Induction”, Academic Press, New York, 1966.

[Kohavi, 1997] Kohavi, R., Sommerfield, D. & Dougherty, J. “Data Mining using MLC++, a Machine Learning Library in C++”, International Journal of Artificial Intelligence Tools: vol. 6, Nº 4, p. 537-566, 1997.

[Mittchel, 1997] Mittchel, T. M. “Machine Learning”. McGraw-Hill, New York, 1997.

[Padilha, 2003] Padilha, T. P. P. “Um Ambiente de Aprendizado Colaborativo para Resolução de Problemas”, Monografia de Qualificação de Doutorado – UFSC, 2003.

[Previero, 2000] Previero, C.A., Riccioppo, M.T.Q., Guimarães, R.C.O., Correa, G.A. & Reys, M.A. “Propriedades físicas do fruto pequi (*Caryocar brasiliensi* camb.)”. In: XXIX Congresso Brasileiro de Engenharia Agrícola, Fortaleza/CE. Anais, CD, 2000.

[Quinlan, 1993] Quinlan J. R. “C4.5: Programs for Machine Learning”. San Mateo, CA: Morgan Kaufmann, 1993.

[Quinlan, 1996] Quinlan J.R. “Improved Use of Continuous Attributes in C4.5”. Journal of Artificial Intelligence Research: p.77-90, 1996

[Weiss, 1991] Weiss, S. M. & Kulikowski, C. A. “Computer Systems That Learn Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning and Expert Systems”, Morgan Kaufmann Publishers, 1991.

[Wilkins, 2003] Wilkins, D.E. & Lee, T.J. & Berry, P. “Interactive Execution Monitoring of Agent Teams”. Journal of Artificial Intelligence Research: Volume 18, p. 217-261, 2003.

[Witten, 1999] Witten, Ian H. & Frank, Eibe. “Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations”. Morgan Kaufmann, 1999.