UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

LÓGICA DIFUSA NO TRATAMENTO DE ATRIBUTOS CONTÍNUOS NA INDUÇÃO DE ÁRVORE DE DECISÃO

SAMUEL LUCAS FINK

BLUMENAU
2011

2011/1-33

SAMUEL LUCAS FINK

LÓGICA DIFUSA NO TRATAMENTO DE ATRIBUTOS CONTÍNUOS NA INDUÇÃO DE ÁRVORE DE DECISÃO

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciência da Computação — Bacharelado.

Prof. Cláudio Ratke, Mestre - Orientador

BLUMENAU
2011

2011/1-33

LÓGICA DIFUSA NO TRATAMENTO DE ATRIBUTOS CONTÍNUOS NA INDUÇÃO DE ÁRVORE DE DECISÃO

Por

SAMUEL LUCAS FINK

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

	_____
Presidente:	Prof. Cláudio Ratke, Mestre – Orientador, FURB
	_____
Membro:	Prof. Mauro Marcelo Mattos, Doutor – FURB
	_____
Membro:	Prof. Roberto Heinzle, Doutor – FURB

Blumenau, 06 de julho de 2011

“Dedico este trabalho a minha família, em especial aos meus pais, por terem confiado em mim, e por não medirem esforços, auxiliando-me na busca do conhecimento com o qual alcancei esta conquista, e por terem me dado amor e carinho sempre.”

AGRADECIMENTOS

A Deus, pelo seu imenso amor e graça.

A meus pais, pela grande força e incentivo nesta jornada sempre ajudando e apoiando nos momentos difíceis.

Aos meus irmãos pelo incentivo a realização dos estudos.

Aos meus amigos, pelos empurrões e cobranças.

Ao meu orientador, Cláudio Ratke, pela paciência, dedicação e sabedoria na orientação deste trabalho mesmo com tantos obstáculos.

Seja você quem for, seja qual for a posição social que você tenha na vida, a mais alta ou a mais baixa, tenha sempre como meta muita força, muita determinação e sempre faça tudo com muito amor e com muita fé em Deus, que um dia você chega lá. De alguma maneira você chega lá.

Ayrton Senna

RESUMO

O objetivo da mineração de dados é prover a descoberta de conhecimento em bases de dados, de modo a obter conhecimento que não é facilmente descoberto a olho nu. Este conhecimento por sua vez pode ser utilizado a fim de auxiliar na tomada de decisão em processos gerenciais de organizações. Árvore de decisão é uma técnica utilizada que cria e organiza regras de classificação. Para maximizar a qualidade do processo de classificação, pode ser utilizada a técnica de lógica difusa. Este projeto apresenta o desenvolvimento de uma aplicação web, que permite executar o processo de mineração de dados. A aplicação utiliza-se da linguagem de programação Java com banco de dados Oracle. Como resultado destaca-se a aplicação da técnica de árvore de decisão juntamente com lógica difusa para realizar o processo de classificação.

Palavras-chave: Descoberta de conhecimento em bases de dados. Mineração de dados. Árvore de decisão. Lógica difusa.

ABSTRACT

The main objective of the data mining is give the discovery of the knowledge in data basis, in the target to obtain the knowledge that's not easily discovered within sight of it. This knowledge can be utilized to help to get decisions in management processes in companies. Decisions Tree is a technique used that create and organize rules of classification. To maximize the process's quality, can be used the fuzzy logic. This project presents the development of a web application, that allows make this data mining process. The application use the language of Java programming with Oracle database. As a result, applies the Tree technique along fuzzy logic to realize this classification process.

Key-words: Knowledge discovery in database. Data mining. Decisions' tree. Fuzzy logic.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

API – *Application Programming Interface*

CC – Ciências da Computação

DM – *Data Mining*

JDBC – *Java DataBase Connectivity*

KDD – *Knowledge Discovery in Database*

OO – Orientação a Objetos

RF – Requisitos Funcionais

RNF – Requisitos Não Funcionais

UML – *Unified Modeling Language*

PMML – *Predictive Model Markup Language*

SUMÁRIO

1 INTRODUÇÃO

Com os avanços tecnológicos de coleta e armazenamento de dados, os Sistemas de Informações (SI) das organizações acumulam diariamente um grande volume de dados em seus bancos de dados. No entanto, para algumas aplicações, muito destes dados armazenados não têm significado nenhum quando analisados isoladamente e sem nenhuma interpretação (CARVALHO, 2003, p. 10). Desta forma, com o passar do tempo, observou-se que é possível transformar os dados armazenados em conhecimento, a fim de auxiliar na tomada de decisão de processos gerenciais da organização.

Para a transformação dos dados armazenados em informações úteis, é necessário técnicas e ferramentas. Para isto, pode ser utilizado o processo denominado Descoberta de Conhecimento em Bases de Dados, ou em inglês *Knowledge Discovery in Database* (KDD) que, para Tan, Steinbach e Kumar (2009, p. 4), é o processo geral de conversão de dados brutos em informações úteis.

O KDD é composto por várias etapas, entre elas a Mineração de Dados, ou em inglês Data *Mining* (DM). Segundo Carvalho L. (2005, p. 3), DM compreende o uso de técnicas automáticas de exploração de grandes quantidades de dados de forma a descobrir novos padrões e relações que, devido ao volume de dados, não são facilmente descobertas a olho nu pelo ser humano. Para o desenvolvimento de DM, podem ser utilizadas técnicas como estatística, redes neurais artificiais e inteligência artificial simbolista (CARVALHO, L., 2005, p. 19).

Na utilização da inteligência artificial simbolista encontram-se a árvore de decisão que segundo Barbieri (2001, p. 170) é utilizada de forma satisfatória na tarefa de DM. Porém, ao lidar com atributos contínuos[1], as árvores de decisão são sensíveis a ruídos e instabilidades[2] nos dados analisados (PENG; FLACH, 2001, p. 109). Como proposta, Peng e Flach (2001, p. 110) sugeriram a utilização de conjuntos difusos para tratar os atributos contínuos.

Resumidamente este trabalho apresenta o desenvolvimento de uma biblioteca para a indução de árvore de decisão difusa que utiliza o algoritmo C4.5 (QUINLAN, 1993), juntamente com lógica difusa, a fim de lidar com os ruídos e instabilidades dos atributos contínuos.

---

[1] Atributos contínuos são aqueles que possuem muitos valores distintos (RATKE; JUSTINO; BORGES, 2003).
[2] Ruídos e instabilidades nos dados pode ser consideradas instâncias duplicadas, atributos faltantes e/ou valores não previstos.

O objetivo é utilizar a representação dos conjuntos difusos com a técnica de árvores de decisões, para obter vantagens como tratamento de incertezas, processamento gradual com compreensão, popularidade e facilidade de aplicação, a fim de melhorar a qualidade do conhecimento obtido no processo de KDD.

1.1 OBJETIVOS DO TRABALHO

Desenvolver uma biblioteca para a indução de árvores de decisão difusas, utilizando das técnicas de árvore de decisão e teoria dos conjuntos difusos, para obter vantagens no tratamento de incertezas.

Os objetivos específicos do trabalho são:

a) disponibilizar um modelo de classificação de dados para gerar conhecimento para tomada de decisão;
b) utilizar o algoritmo C4.5 para indução de árvore de decisão;
c) utilizar a teoria dos conjuntos difusos para lidar com atributos contínuos;
d) disponibilizar uma ferramenta *web* para indução de árvore de decisão difusa;
e) demonstrar o uso da ferramenta através do estudo de caso jogar golfe;
f) comparar a precisão do conhecimento gerado pela árvore de decisão difusa com o de uma árvore de decisão utilizando o software Weka 3 (MACHINE LEARNING GROUP AT UNIVERSITY OF WAIKATO, 2010);

1.2 ESTRUTURA DO TRABALHO

Este trabalho está organizado em quatro capítulos, sendo que, no primeiro capítulo foi apresentada a introdução, os objetivos e a estrutura do trabalho.

No segundo capítulo é apresentada à fundamentação teórica, bem como os assuntos que serviram de base para o desenvolvimento do trabalho e a apresentação de trabalhos correlatos.

No terceiro capítulo está descrito o desenvolvimento da aplicação proposta, as técnicas e ferramentas utilizadas bem como a elaboração de diagramas para auxiliar na compreensão

da aplicação.

E por fim, no quarto capítulo as conclusões, limitações e sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo aborda o processo de KDD, bem como as suas etapas e a importância de cada uma delas para se extrair informações úteis a partir de dados armazenados. O processo de DM com maior ênfase na tarefa de classificação utilizando da técnica de árvore de decisão. A teoria dos conjuntos difusos com destaque para a função de pertinência gaussiana e onde ela pode contribuir no processo de DM. E por fim trabalhos correlatos que mostram a utilização da técnica de árvore de decisão com ou sem a utilização da teoria dos conjuntos difusos.

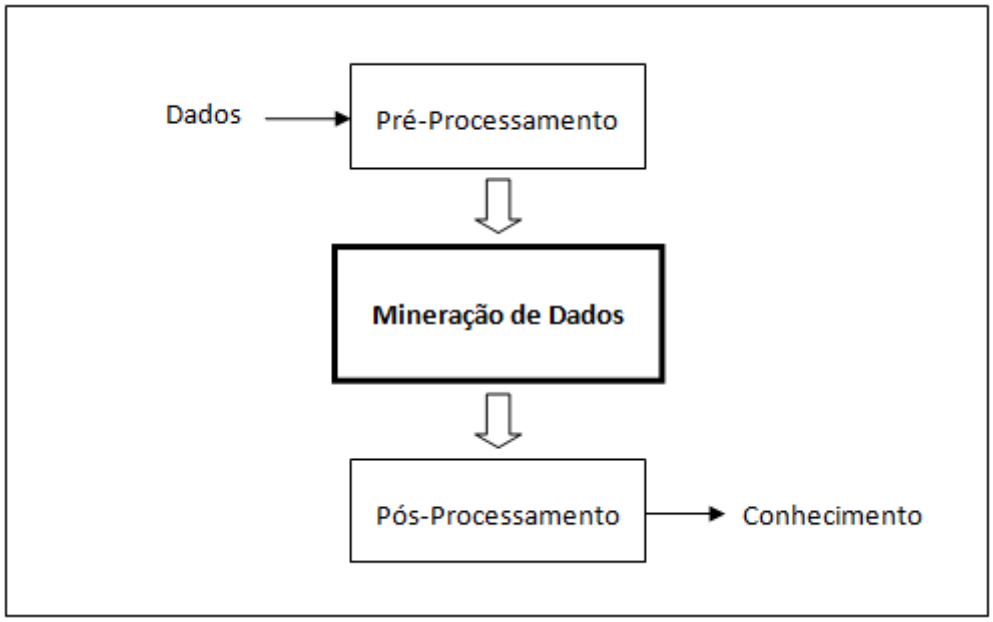
2.1 DESCOBERTA DE CONHECIMENTO EM BANCOS DE DADOS

O interesse que se tem pelo KDD, para Feldens (1996 apud COELHO, 2007, p. 4) fundamenta-se no fato de que grandes bases de dados podem ser uma fonte de conhecimento útil, porém, não explicitamente representado e cujo objetivo é desenvolver e validar técnicas, metodologias e ferramentas capazes de extrair conhecimento implícito nos dados e representá-lo de forma acessível aos usuários.

O KDD pode ser definido como um processo não trivial de identificar padrões válidos, novos e potencialmente úteis e compreensíveis (FAYYAD; PIATETSKY-SHAPIRO; SMITH, 1996, apud ARAÚJO, 2006, p. 13). Também como um processo geral de conversão de dados brutos em informações úteis (TAN, STEINBACH e KUMAR, 2009, p. 4).

Com base nos autores Tan, Stainbach e Kumar (2009, p.4) o processo consiste em três etapas essenciais, o pré-processamento, a mineração de dados e o pós-processamento, como podem ser observados na Figura 1.

Para obter bons resultados com o processo de KDD deve-se iniciar com o entendimento do domínio da aplicação e dos objetivos finais a serem atingidos (RODRIGUES FILHO; SHIMIZU, 2002). Este processo deve ser realizado por conhecedores da área de domínio do negócio, da base de dados e com apoio do especialista de KDD, para que, ao concluir o processo, a informação gerada possa ser de fácil entendimento.



Fonte: adaptado de Tan, Stainbach e Kumar (2009, p.4).

Figura 1 - O processo de KDD

A etapa do pré-processamento tem como propósito transformar os dados das entradas brutas em um formato apropriado para análises subseqüente (TAN; STEINBACH; KUMAR, 2009, p. 4). Como destaca Rodrigues Filho e Shimizu (2002), esta etapa é dividida em três passos:

a) Seleção de recursos: é onde ocorre a fusão dos dados de diferentes fontes (arquivos comuns, planilhas, bases de dados heterogêneas, *data warehouse*, etc.) que compreendem o domínio da área de aplicação;
b) Redução de dimensionalidade/normalização: é a remoção de ruídos, como, registros duplicados, atributos fora do domínio, dados ausentes, atributos exclusivos das regras de negócio, entre outros erros; e
c) Criação de subconjuntos de dados: consiste na adequação dos dados, a fim de facilitar o seu uso pelas técnicas de mineração de dados, pois cada técnica possui necessidades específicas, como por exemplo, entrada, variáveis auxiliares e categorização de atributos.

A etapa de mineração de dados pode ser considerada o núcleo do KDD, consistindo na aplicação de algoritmos de extração de padrões a partir dos dados armazenados (RODRIGUES FILHO; SHIMIZU, 2002).

Para a aplicação da tarefa de DM existem várias tarefas e técnicas. A classificação e a regressão são dois exemplos diferentes de tarefas utilizadas no processo de DM. Já dentro da classificação várias técnicas podem ser utilizadas como árvores de decisão, regras de indução, regras de associação, entre outras. Cada uma destas tarefas possui suas características e

funcionalidade especifica, onde o desempenho é dependente do problema e de seus dados correspondentes.

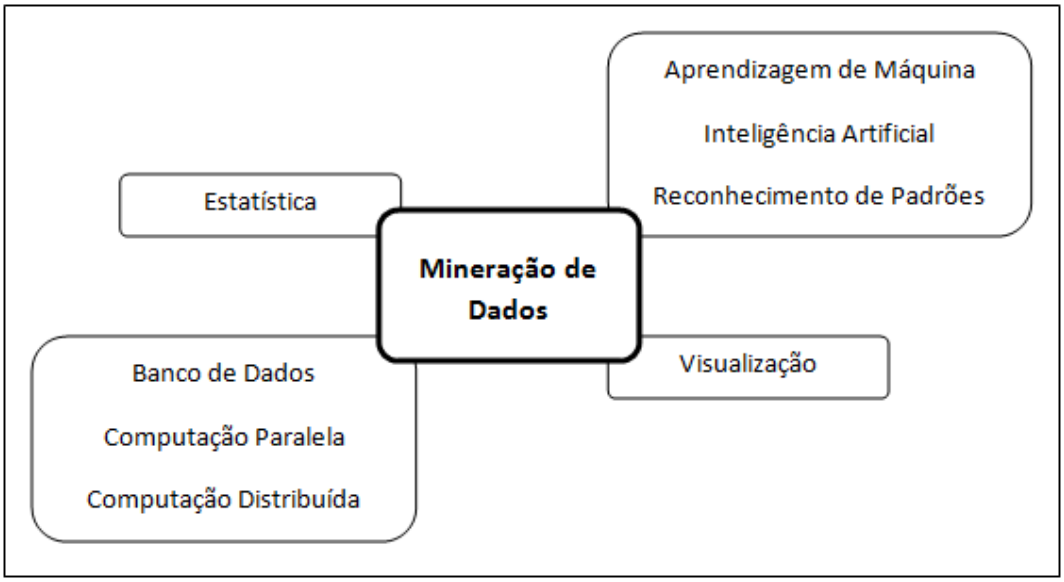
Estas tarefas normalmente provem de áreas como aprendizagem de máquina, reconhecimento de padrões, estatística e inteligência artificial. As técnicas quando combinadas podem gerar melhores resultados.

Na etapa de pós-processamento a informação é extraída e analisada em relação aos objetivos definidos no início do processo, a fim de identificar a validade ou relevância dos padrões encontrados, para posterior apresentação dos resultados. Caso os resultados sejam irrelevantes, deve-se retornar às etapas anteriores e refazê-las. Esta iteração pode ocorrer até se obter resultados aceitáveis ou concluir que não é possível extrair conhecimento do conjunto de dados processado.

2.2 MINERAÇÃO DE DADOS

Mineração de Dados pode ser definida como um processo para agir sobre grandes bancos de dados com o intuito de descobrir padrões úteis e recentes que poderiam de outra forma, permanecer ignorados (TAN; STEINBACH; KUMAR, 2009, p. 3). Também pode ser definida como a exploração e análise de grandes quantidades de dados, a fim de descobrir padrões e regras significativas (BERRY; LINOFF, 2004, p. 7).

Como pode ser observado nas definições de DM, trata-se de um processo que tem como desafios trabalhar com grandes quantidades e de diferentes tipos de dados, com vários tipos de algoritmos e de diferentes áreas de conhecimento. Além disto, a Ciências da Computação (CC) também desempenha um papel importante com o fornecimento de novas tecnologias com o intuito de se obter um processo mais eficaz. Os autores Tan, Steinbach e Kumar (2009, p.8) demonstram o relacionamento da mineração de dados com outras áreas de conhecimento, como poder ser observado na Figura 2.



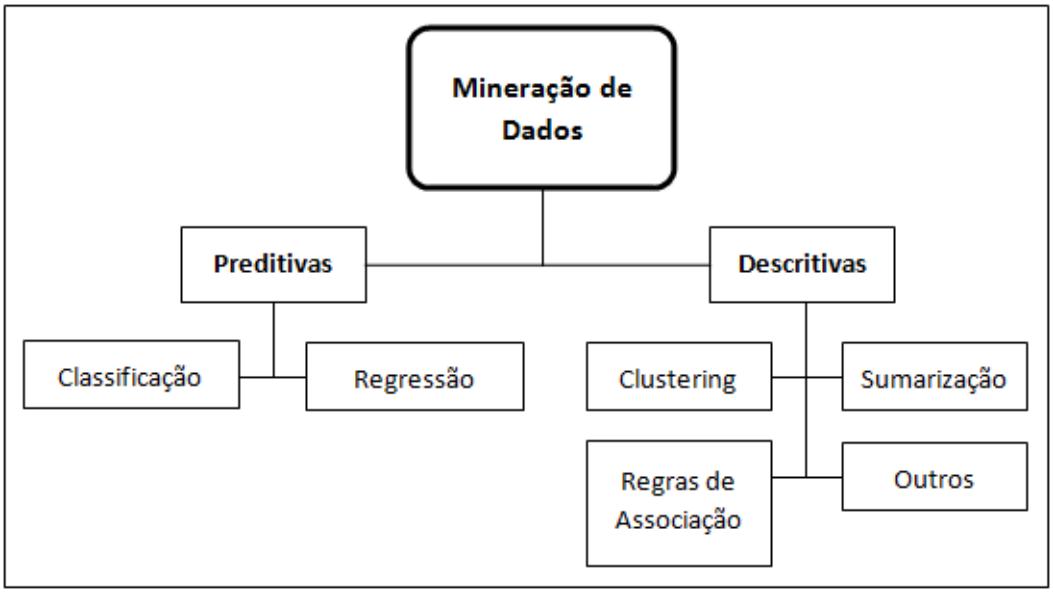
Fonte: adaptado de Tan, Stainbach e Kumar (2009, p.8).

Figura 2 - Mineração de Dados como uma confluência de muitas disciplinas

Para a execução do processo de DM existem diversas tarefas cada qual com suas técnicas e objetivos. As tarefas podem ser classificadas como descritivas e preditivas (BERRY; LINOFF, 1997, p. 50):

a) Descritivas: consiste na descoberta de padrões interpretáveis por humanos que descrevem os fatos cadastrados na base de dados; e as
b) Preditivas: consiste em utilizar determinadas variáveis para prever valores desconhecidos de outras variáveis de interesse.

Com base nos autores Berry e Linoff (1997, p.50) e Rezende (2003, p. 318) é possível classificar as tarefas como descritivas ou preditivas conforme a Figura 3.



Fonte: adaptado de Berry e Linoff (1997, p.50) e Rezende (2003, p. 318).

Figura 3 - Classificação das Tarefas de Mineração de Dados

As características da predição e da descrição são atendidas através das tarefas de DM. A seguir é descrita a tarefa de Classificação e a técnica de Árvore de Decisão.

2.3 CLASSIFICAÇÃO

A classificação é uma tarefa de DM que, segundo Barbieri (2001, p.187), é usada para definir grupos ou classes de elementos, baseados em certos parâmetros pré-estabelecidos. Também pode ser definido como uma tarefa de encontrar um modelo (ou função) que descreva e distingue classes de dados e conceitos, com o objetivo de se utilizar para prever a classe de um objeto cujo rotulo é desconhecido (HAN; KAMBER, 2006, p. 24).

De forma simplificada, a tarefa de classificação procura construir um modelo a partir de um conjunto de dados de um determinado evento, cujo objetivo é prever a ocorrência deste evento, de acordo com a análise de suas características. Por exemplo, em um sistema de concessão de crédito, onde o cliente pode ser aceito ou rejeitado de acordo com os padrões estabelecidos de atraso de pagamento, idade, classe social, entre outros.

Diversos algoritmos de classificação foram propostos por pesquisadores de diferentes áreas como estatística, aprendizagem de máquina e reconhecimento de padrões. Estes algoritmos por sua vez utilizam técnicas como árvores de decisão, redes neurais, método bayesiano (probabilidade de características), regras determinísticas e regras probabilísticas.

Para Breiman et al (1984, p. 43), um classificador extraído de um conjunto de dados serve a dois propósitos:

a) predição de um valor;
b) compreensão da relação existente entre os atributos previsores e a classe.

Para atender os propósitos do classificador, além de classificar os dados, o conhecimento descoberto deve ser representado de forma compreensível. Uma das possíveis formas de representação da árvore de decisão é a utilização de regras de produção no formato "se"..(condições).."então"..(conclusão).., onde a interpretação é: "se" os valores satisfazem as condições da regra "então" o exemplo pertence à classe prevista pela regra.

Com o objetivo de se ter bons resultados, segundo Hand (1997, p.73), as regras de classificação encontradas devem ser avaliadas, segundo três critérios, que são precisão preditiva, a compreensibilidade e o grau de interesse do conhecimento descoberto.

A precisão preditiva é normalmente medida com o número de exemplos de teste

classificados corretamente dividido pelo número total de exemplos de teste. Existem formas mais sofisticadas de se medir a precisão preditiva (HAND, 1997, p.74), mas por ser uma forma simples, em sua essência, é a forma mais utilizada na prática. Já a compreensibilidade, geralmente é medida pela simplicidade em relação ao número de regras descobertas e do número de condições por regra. Quanto maior estes números, menos compreensível é o conjunto de regras descoberto. O grau de interesse do conhecimento descoberto, mesmo que seja correto do ponto de vista estatístico, pode não ser de fácil compreensão, pois o conjunto pode ser muito grande para ser analisado, conter redundâncias ou ainda representar um relacionamento já conhecido. Desta forma, também se torna importante validar o conhecimento pelo usuário.

Uma boa prática para a classificação de dados é a divisão dos dados em dados de treinamento e dados de teste. Assim, um conjunto de dados de treinamento é utilizado para a construção do modelo de classificação. Com modelo de classificação construído, iniciam-se os testes de modo a classificar os dados de testes. Ao final, o modelo construído é avaliado como um bom modelo, do ponto de vista preditivo, somente se uma alta percentagem dos dados de teste for classificada corretamente.

2.4 ÁRVORE DE DECISÃO

Árvore de decisão é uma técnica que a partir de um conjunto de dados, cria e organiza regras de classificação em um formato simples de diagrama de árvore (BARBIERI, 2001, p.190). As regras de classificação são calculadas através de algoritmos de aprendizagem de máquina que realizam cálculos sobre os atributos de um conjunto de dados. Para Dias (2001, p. 12), ID3, C4.5 e CART são exemplos de algoritmos para indução de árvores de decisão.

Na execução desta técnica, assim como em outras técnicas de classificação, o usuário deve escolher um atributo que deseja avaliar e o algoritmo procura os atributos mais correlacionados a ele e cria uma árvore de decisão com várias ramificações (WITTEN; FRANK, 2000). No contexto deste trabalho este atributo será denominado de atributo meta.

As regras de classificação encontradas em relação ao atributo meta são representadas em uma estrutura de árvore, que é organizada de forma que cada:

a) nó representa o teste de um atributo;
b) ramo representa um dos possíveis valores do atributo;

c) folha representa a resposta de acordo com o atributo meta.

Pode-se exemplificar uma árvore de decisão que procure definir se uma pessoa deve ou não jogar golfe (TAN, STEINBACH e KUMAR, 2009, p. 147). Para indução de árvore de decisão os atributos tempo, temperatura, umidade, vento e o atributo meta que é jogar, são avaliados. Com a indução de árvore de decisão as seguintes regras de classificação podem ser apresentadas (Figura 4):

a) se (Tempo = Sol) e (Umidade <= 82) então (Jogar = Sim);

b) se (Tempo = Sol) e (Umidade > 82) então (Jogar = Não);

c) se (Tempo = Nublado) então (Jogar = Sim);

d) se (Tempo = Chuva) e (Vento = Sim) então (Jogar = Não);

e) se (Tempo = Chuva) e (Vento = Não) então (Jogar = Sim);

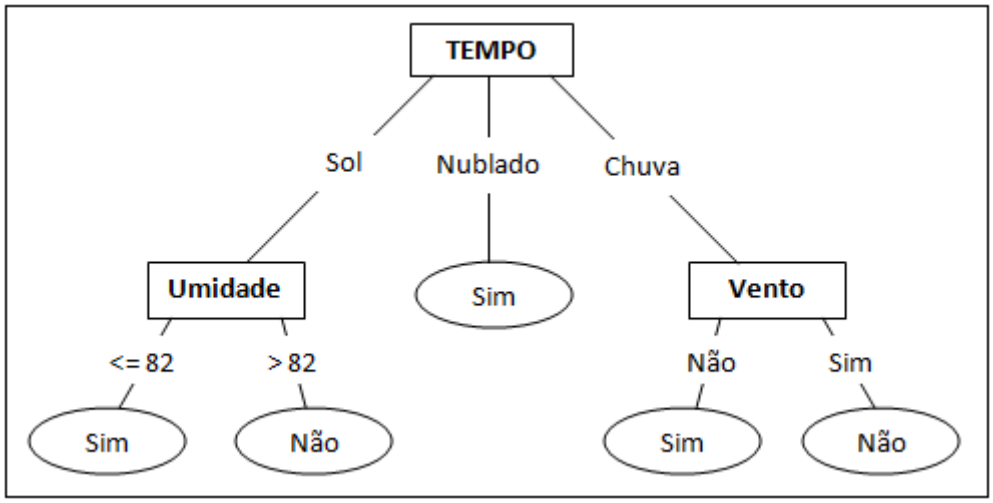


Figura 4 - Exemplo de árvore de decisão

A maioria dos algoritmos desenvolvidos para indução de árvores de decisão é uma variação de algoritmo *top-down* (MITCHELL, 1997, p. 55), ou seja, este tipo de algoritmo utiliza a estratégia de dividir problemas complexos em subproblemas mais simples. O algoritmo divide recursivamente um conjunto de dados, até que cada subconjunto contenha casos de uma única classe.

Com base em Pimenta (2004, p. 18), os algoritmos de indução de árvore de decisão que utilizam a técnica *top-down* podem ser representados de forma genérica pelo pseudo-algoritmo do Quadro 1.

```
ENTRADA: Conjunto de dados (D)
SAÍDA  : Árvore de Decisão

Função GeraÁrvore(D)

        Se atingido critério de parada

                Cria um nó folha

        Se Não

                Seleciona o atributo Ai que maximiza a função de seleção

                Para cada valor do atributo Ai

                        Aplica  GeraÁrvore(Dk)

                Fim Para

        Fim Se
```

Fonte: adaptado de Pimenta (2004, p. 18).

Quadro 1 - Pseudo-algoritmo de indução de árvore de decisão

Mesmo sendo um algoritmo genérico é possível destacar características que diferenciam os algoritmos de indução de árvore de decisão:

a) cada nó pode possuir mais de um valor possível;
b) os testes nos nós podem possuir mais de um atributo como filtro;
c) os atributos podem ser discretos ou contínuos. Existem algoritmos que têm a capacidade de lidar com os dois tipos de atributo, enquanto outros algoritmos lidam somente com atributos discretos.

Em 1993, Ross Quinlan propôs o algoritmo C4.5 que foi desenvolvido a partir do algoritmo ID3. Neste novo algoritmo é possível destacar algumas melhorias:

a) manipulação de atributos contínuos e discretos;
b) manipulação de conjuntos de dados com valores faltantes;
c) manipulação de atributos com custos diferentes;
d) poda da árvore após a sua indução.

Neste algoritmo o principal passo é a escolha de um atributo para rotular o nó atual da árvore. Para isto, deve-se escolher o atributo que possuí o maior poder de discriminação entre as classes para o exemplo do nó atual. Para calcular o poder de discriminação dos atributos, são utilizadas medidas baseadas na teoria da informação (COVER; THOMAS, 1991; QUINLAN, 1993, apud CARVALHO, D., 2005, p.13). Nesse algoritmo são utilizadas as medidas de entropia e do ganho de informação.

A entropia (QUINLAN, 1993) é uma medida que indica o grau de aleatoriedade do atributo. Para Quinlan (1993), quanto menor o grau de aleatoriedade do atributo, melhor é a capacidade de previsão. A entropia é calculada utilizando a expressão do Quadro 2, onde $p(D,j)$ é a proporção de casos em D que pertencem a classe j e C é o número total de classes.

$$Entropia\ \ (D) = -\sum_{j=1}^{C} p(D, j) \times \log_2 (p(D, j))$$

Quadro 2 – Expressão de calculo da entropia

O ganho de informação é uma medida que representa a redução da entropia causada pela partição dos exemplos de acordo com os valores do atributo (MITCHELL, 1997, p.56). O ganho de informação é calculado utilizando a fórmula do Quadro 3, onde k é a quantidade de classes e Di são os casos por classe.

$$Ganho\ \ (D, T) = Entropia\ \ (D) - \sum_{i=1}^{k} \frac{|D_i|}{|D|} \times Entropia\ \ (D_i)$$

Quadro 3 - Expressão de calculo do ganho de informação

Ao utilizar atributos que possuem informação numérica, é necessário transformar os dados para forma discreta. Uma das formas é a utilização da média aritmética simples da classe, conforme expressão do Quadro 4, onde *n* é a quantidade de classes e D_i são os casos por classe.

$$Discretiza\ \ (D) = \frac{\sum_{i}^{n} D_i}{n}$$

Quadro 4 - Expressão de calculo da média aritmética simples

Tomando-se o conjunto de dados de treinamento da Tabela 1, que se refere ao exemplo de jogar golfe, pode-se ilustrar a aplicação da árvore de decisão.

Tabela 1 - Conjunto de Dados Jogar Golfe

CONJUNTO DE DADOS JOGAR GOLFE				
Tempo	**Temperatura**	**Umidade**	**Vento**	**Jogar**
Sol	85	85	Não	Não
Sol	80	90	Sim	Não
Nublado	83	86	Não	Sim
Chuva	70	96	Não	Sim
Chuva	68	80	Não	Sim
Chuva	65	70	Sim	Não
Nublado	64	65	Sim	Sim
Sol	72	95	Não	Não
Sol	69	70	Não	Sim
Chuva	75	80	Não	Sim
Sol	75	71	Sim	Sim
Nublado	72	90	Sim	Sim
Nublado	81	75	Não	Sim
Chuva	71	91	Sim	Não

Para se identificar o nó raiz, primeiramente deve-se calcular a entropia da classe meta (“Jogar”), conforme Quadro 5.

```
Entropia (Jogar) = [p(Joga, Sim) * log2(p(Joga, Sim))] + [p(Joga, Não) * log2(p(Joga, Não))]
Entropia (Jogar) = [9/14 * log2(9/14)] + [5/14 * log2(5/14)]
Entropia (Jogar) = 0,940285
```

Quadro 5 - Cálculo de entropia para a classe meta "Jogar"

Em seguida deve-se calcular ganho de informação dos demais atributos, de tal forma que, o atributo que possuir o maior ganho será o nó raiz. Para o atributo “Vento” conforme Quadro 6.

```
D(Vento=sim; Jogar=sim) = 3/6 * log2(3/6) = -0,5
D(Vento=sim; Jogar=não) = 3/6 * log2(3/6) = -0,5
D(Vento=não; Jogar=sim) = 6/8 * log2(6/8) = -0,311278124
D(Vento=não; Jogar=não) = 2/8 * log2(2/8) = -0,5
Ganho (Joga, Vento) = Entropia (Joga) – |{[6/14 * (-0,5 + -0,5)] + [8/14 * (-0,3113 + -0,5) ] }|
Ganho (Joga, Vento) = 0,04813
```

Quadro 6 - Cálculo do ganho de informação para o atributo "Vento"

Para calcular os atributos “Temperatura” e “Umidade”, deve-se para cada atributo obter dados em formato discreto. Assumindo que os dados encontram-se de forma discreta, então para o atributo “Umidade”, conforme Quadro 7. Para os demais atributos conforme Quadro 8.

```
Ganho (Joga, Umidade) = Entropia (Joga) – |{[7/14 * (-0,4010 + -0,1906)] + [7/14 * (-0,5239 + -
0,4613) ] }|
Ganho (Joga, Umidade) = 0,1518
```

Quadro 7 - Cálculo do ganho de informação para o atributo "Umidade"

```
Ganho (Joga, Temperatura) = 0,0013
Ganho (Joga, Tempo) = 0,2467
```

Quadro 8 - Resultado do cálculo de ganho de informação para os demais atributos

Como pode ser observado, o atributo que possui maior ganho de informação é o “Tempo”, portando este será o nó raiz. Este processo e executado recursivamente para todos os ramos da árvore, até se encontrar uma condição de parada válida (como pode ser observado no pseudo-algoritmo do Quadro 1).

A vantagem da árvore de decisão está na fácil compreensão dos resultados gerados quando comparado com outras técnicas, porém, o algoritmo não exclui a possibilidade de erros nas classificações ao lidar com atributos contínuos, instabilidades ou ruídos nos dados.

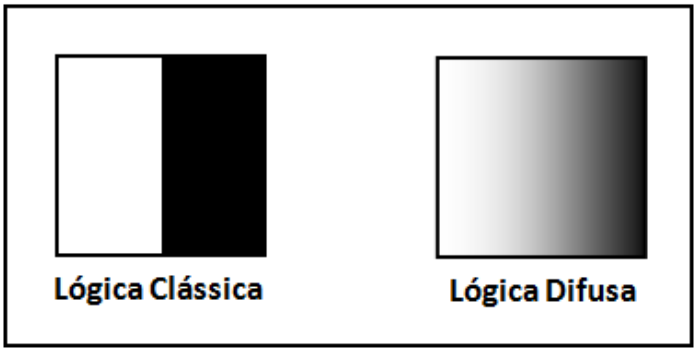
2.5 LÓGICA DIFUSA

Para Tah e Carr (2000, p. 492), a lógica difusa é um sistema onde é possível caracterizar uma determinada afirmativa não somente como certa ou errada, mas podendo ser

parcialmente certa ou parcialmente errada. Sendo assim, um determinado elemento pode pertencer a mais de um conjunto, a fim de eliminar a mudança brusca de um conjunto para outro.

Diferente da lógica tradicional, onde os elementos dos problemas assumem a possibilidade de dois valores, o verdadeiro ou o falso, de acordo com Zadeh, Jamshidi e Titli (1997), a lógica difusa considera os elementos pertencentes a um determinado conjunto com certo grau de pertinência.

Com base em Kohagura (2007, p. 6), a idéia da lógica difusa, é não ficar restrita apenas ao verdadeiro e falso, mas sim aos vários níveis entre o verdadeiro e o falso. De modo figurativo, enquanto a lógica clássica enxerga somente o preto e o branco, a lógica difusa é capaz de enxergar vários tons de cinza, como demonstrado na Figura 5.



Fonte: adaptado de Kohagura (2007, p. 6)

Figura 5 - Comparativo lógica clássica e lógica difusa

A lógica difusa é caracterizada por uma função de pertinência, pois é ela que determina o quanto o elemento pertence ao conjunto (ZIMMERMAN, 1991 apud KOHAGURA, 2007, p. 23). Existem diferente tipos de função de pertinência como triangular, trapezoidal, gaussiana, cauchy e sigmóide.

De acordo com Morais (2007, p. 7), a função gaussiana tem como característica a distribuição normal, ou seja, tende a zero para valores muito maiores ou muito menores que a média. É calculada utilizando a expressão do Quadro 9, onde x é o valor de entrada, μ é a média do conjunto de dados do valor de entrada e σ é o desvio padrão do conjunto de dados do valor de entrada.

$$G(x, \mu, \sigma) = e^{-\frac{(x-\mu)^2}{\sigma^2}}$$

Quadro 9 - Expressão de cálculo da função gaussiana

Utilizando o conjunto de dados da Tabela 1, como exemplo, a Figura 6, representa a função gaussiana para o atributo “Temperatura” em relação ao atributo meta “Jogar”.

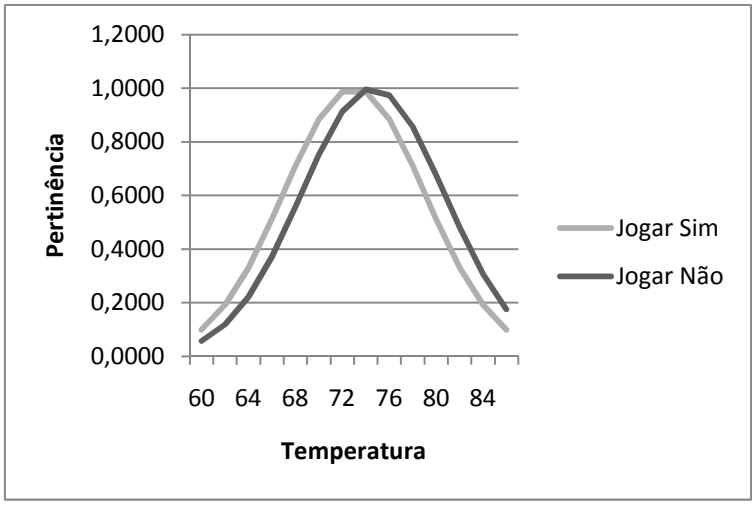
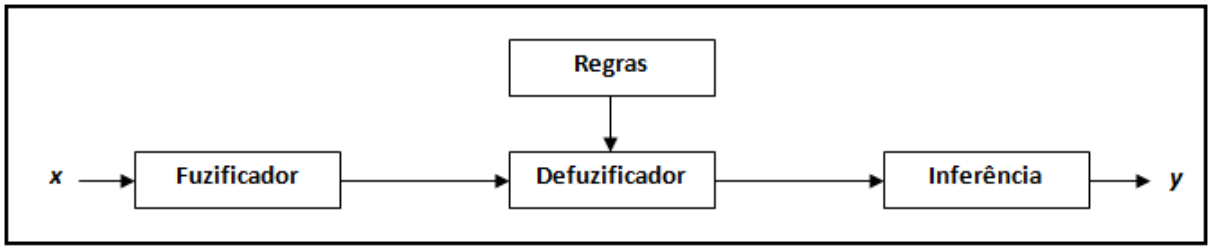


Figura 6 - Gráfico da Função Gaussiana de Temperatura x Jogar

A lógica difusa é composta por um processo de inferência que possui três etapas o fuzificador, a inferência e o defuzificador, como pode ser observado na Figura 7.

Este processo de acordo com Moraes (2007, p. 3) mapeia um valor escalar em um número entre 0 e 1 (Figura 6) indicando o grau de pertinência desse valor ao conjunto. Por sua vez os valores 0 e 1, representam, respectivamente, o menor e o maior grau de probabilidade de pertencer ao conjunto.



Fonte: adaptado de Morais (2007, p. 3)

Figura 7 – Processo de inferência da lógica difusa

Conforme Moraes (2007, p. 4) a lógica difusa passa por um processo com diferentes etapas onde:

a) o fuzificador é responsável pela transformação das variáveis de entrada do problema em valores difusos. Neste momento é aplicada a função de pertinência para cada valor de entrada;
b) as regras são fornecidas ou extraídas de dados numéricos utilizando, por exemplo, a capacidade de aprendizagem da árvore de decisão. Para a elaboração das regras é importante seguir alguns conceitos, como:
   - variáveis lingüísticas: com elas é possível nomear os conjuntos,
   - conexões lógicas do tipo E/OU para criar relações entre variáveis;
   - implicações do tipo Se A Então B;

c) a inferência aplica as regras sobre os valores de entrada fazendo o mapeamento da lógica difusa em conjuntos difusos.
d) e por último, o defuzificador converte os resultados do processo de inferência em valores discretos para a variável de saída do processo.

A utilização da lógica difusa na implementação de sistemas de controle ou de tomada de decisão, permite uma aproximação muito maior da realidade que é marcada por um conjunto de dados com muitas variáveis e valores ambíguos e inexatos (KOHAGURA, 2007, p. 48).

2.6 ÁRVORE DE DECISÃO DIFUSA

A árvore de decisão difusa utiliza a lógica difusa no tratamento de atributos contínuos para a indução de uma árvore de decisão, beneficiando-se desta forma da habilidade de se tratar informações vagas ou imprecisas (RATKE; JUSTINO; BORGES, 2003). O algoritmo, de acordo com Araújo (2006, p. 40) aplica a lógica difusa na formulação dos nós de decisão. Desta forma, para Ratke, Justino e Borges (2003), cada nó de decisão de um atributo contínuo possui conjuntos difusos associados a cada ramo da árvore (Figura 8). Assim mais de um caminho pode ser percorrido na árvore e, portanto, pode-se obter mais de uma classe como resposta.

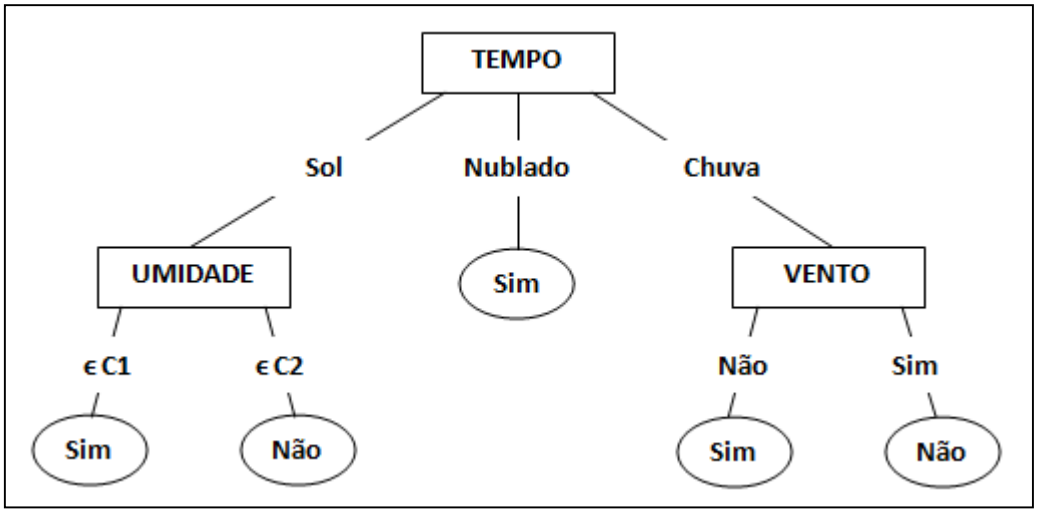


Figura 8 - Exemplo de árvore de decisão difusa

O algoritmo para a indução de árvore de decisão difusa é bastante similar ao algoritmo original, como apresentado no Quadro 1. Contudo, com a introdução da lógica difusa,

alterações precisam ser realizadas, (RATKE, JUSTINO e BORGES; 2003):

a) a regra de produção de um atributo contínuo passa a utilizar o operador pertence ao invés dos operadores > e <=;
b) como um atributo pode pertencer a mais de um conjunto difuso, mais de um caminho pode ser percorrido na árvore, e desta forma pode-se obter mais de uma classe como resposta;
c) o resultado apresentado pela árvore deve ser uma combinação dos graus de pertinência obtidos desde o nó raiz até o nó folha.

2.7 TRABALHOS CORRELATOS

Estudos semelhantes ao proposto já foram desenvolvidos, cada qual com suas características e aplicado a uma área de domínio específica. Entre eles estão os trabalhos desenvolvidos por: Compolt (1999), Ratke, Justino e Borges (2003) e Araújo (2006).

Compolt (1999) apresenta um modelo de classificação e segmentação de dados utilizando a técnica de indução de árvore de decisão. Para a demonstração do modelo foi desenvolvido um protótipo de um sistema de informação que possui as seguintes características:

a) Utilização de oito atributos no modelo de classificação;
b) Definição de prioridade de cada atributo, para os cálculos de entropia;
c) Exibição dos resultados por visualização SE/ENTÃO e visualização por nível;
d) Limitado ao domínio de aplicação de concessão de crédito a fornecedores;

Ratke, Justino e Borges (2003) demonstram a utilização da técnica com o objetivo de obter melhores resultados em relação à limitação da árvore de decisão de lidar com atributos contínuos. Para a indução de árvore de decisão difusa é utilizado o algoritmo C4.5 adaptado para utilizar a teoria dos conjuntos difusos através da função de pertinência sigmóide. Conforme Ratke, Justino e Borges (2003), isto permite melhorar a generalização das regras ao introduzir o tratamento de informações imprecisas. Para demonstrar a utilização deste modelo de extração de conhecimento, foi utilizado um conjunto de dados onde o objetivo é prever se o jogador deve ou não jogar golfe conforme as condições do tempo.

Araújo (2006) apresenta um algoritmo de aprendizagem de máquina que utiliza árvores de decisão difusa para ser aplicado sobre um conjunto de dados extraído de imagens, a fim de

definir a idade gestacional de recém nascidos. O algoritmo desenvolvido é chamado de *Soft Decision Tree* (SDT) que é baseado no algoritmo de Peng e Flach (2001, p. 114). O SDT gera uma árvore de decisão difusa utilizando discretização suave dos atributos para transformar nós de decisão em nós de decisão difusa (ARAÚJO, 2006, p. 42). Além do SDT, outros algoritmos como J4.8, REPTree e LMTree, são disponibilizados para utilização na aplicação proposta no processo de indução de árvore de decisão, porém o único que utiliza lógica difusa é o SDT.

3 DESENVOLVIMENTO DA FERRAMENTA

Este capítulo detalha as etapas de desenvolvimento da ferramenta FTree. São apresentados os requisitos, a especificação e desenvolvimento da ferramenta, destacando as técnicas e ferramentas utilizadas. Também são comentadas questões referente a operacionalidade e os resultados obtidos.

3.1 REQUISITOS DO SISTEMA

O Quadro 10 apresenta os principais Requisitos Funcionais (RF) previstos para o sistema FTree que é proposto neste trabalho.

REQUISITOS FUNCIONAIS
RF01 – Importar arquivo com formato *comma-separated values* (CSV)
RF02 – Permitir a escolha do atributo meta a ser avaliado no processo de classificação
RF03 – Permitir a escolha dos atributos a serem utilizados no processo de classificação
RF04 – Permitir a escolha do número máximo de níveis da árvore
RF05 – Permitir a escolha do valor de pertinência aceito
RF06 – Permitir a utilização da lógica difusa
RF05 – Permitir criar um modelo de classificação
RF06 – Permitir salvar o modelo de classificação criado

Quadro 10 - Requisitos funcionais

O Quadro 11 apresenta os principais Requisitos Não Funcionais (RNF.

REQUISITOS NÃO FUNCIONAIS
RNF01 – Utilizar a técnica de indução de árvore de decisão como modelo de classificação
RNF02 – Utilizar o algoritmo C4.5 para indução de árvore de decisão
RNF03 – Utilizar a teoria dos conjuntos difusos, para lidar com atributos contínuos
RNF04 – Utilizar o modelo de regra de produção “se então” para representar as regras de classificação
RNF05 – Ser implementado utilizando a linguagem de programação Java
RNF06 – Ser implementado utilizando o ambiente de desenvolvimento Oracle JDeveloper 11g
RNF07 – Utilizar o banco de dados Oracle Database Express Edition
RNF08 – Utilizar a *Application Programming Interface* (API) *Java DataBase Connectivity* (JDBC) para se conectar ao banco de dados

Quadro 11 - Requisitos não funcionais

3.2 ESPECIFICAÇÃO DO SISTEMA

Nesta seção é apresentada a especificação do sistema que utiliza conceitos, de Orientação a Objetos (OO) juntamente com *Unified Modeling Language* (UML) para a especificação dos diagramas de caso de uso, diagrama de atividades e diagramas de classe. E o conceito de Modelo de Entidade e Relacionamento (MER) para a especificação do modelo de dados. O desenvolvimento das especificações foi utilizada a ferramenta Oracle JDeveloper 11g, que permite desde o desenvolvimento de diagramas UML e MER, até o desenvolvimento da programação.

Também é especificado o formato que o arquivo CSV deve possuir para a correta importação da base de dados.

3.2.1 Diagrama de casos de uso

A Figura 9 apresenta o diagrama de casos de uso do sistema.

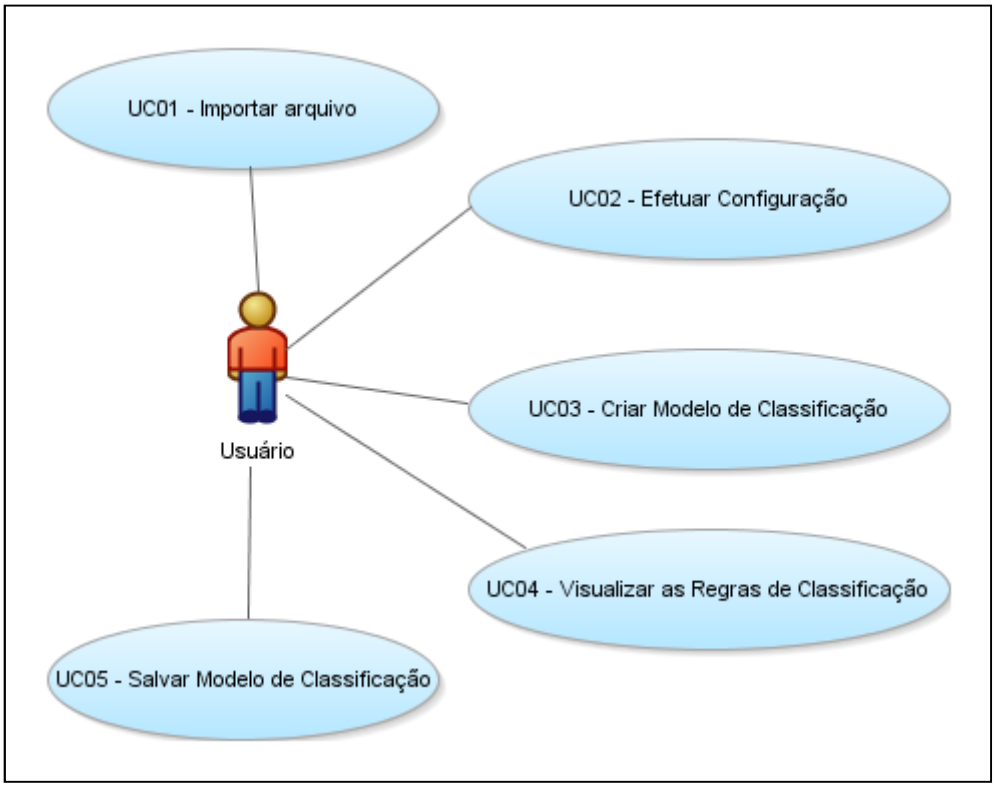


Figura 9 - Diagrama de casos de uso

Nos casos de uso apresentados no diagrama não é possível definir um caso principal,

todos por sua vez são importantes para o correto funcionamento do sistema. A ordem de execução é identificada pela própria numeração dos casos de uso, que se inicia ao importar os dados (UC01), em seguida são efetuadas as configurações (UC02), e finaliza-se com a criação (UC03) e visualização das regras de classificação (UC04), e caso o usuário desejar as regras de classificação é possível salvar (UC05).

No Quadro 12 é apresentado de forma detalhado cada caso de uso presente no diagrama da Figura 9.

UC01 – Importar arquivo	
Pré-condições	Não há pré-condições.
Cenário Principal	01) O usuário inicia o sistema FTree. 02) O usuário seleciona o arquivo que deseja importado. 03) O usuário pressiona o botão importar. 04) O sistema importa o arquivo.
Fluxo Alternativo 01	No passo 04, se o usuário optar por cancelar: 04.1) O sistema FTree cancela a importação de desfaz as alterações.
Exceção 01	No passo 04, se o arquivo de entrada for inválido: 04.1) O sistema FTree exibe uma mensagem de erro. 04.2) O sistema FTree cancela o processo.
Pós-condição	De acordo com o cabeçalho do arquivo será criada uma tabela e em seguida todos os dados serão importados nesta tabela.

Quadro 12 - Caso de uso 01

UC02 – Efetuar Configuração	
Pré-condições	O arquivo já deve estar importado no sistema FTree.
Cenário Principal	01) O usuário visualiza o quadro de configurações. 02) O sistema exibe os atributos no quadro. Ligeiramente após a importação do arquivo com as configurações padrão. 03) O usuário altera os atributos a serem utilizados na classificação, se assim desejar. 04) O usuário marca um atributo como meta. 05) O usuário altera o número máximo de níveis da árvore, se assim desejar.
Exceção 01	No passo 03, se o usuário desmarcar todos os atributos: 03.1) O sistema FTree exibe uma mensagem de alerta.
Exceção 02	No passo 04, o usuário marca como meta um atributo que não está selecionado para a classificação: 04.1) O sistema FTree exibe uma mensagem de alerta.
Exceção 03	No passo 05, o usuário informa um número inválido de níveis: 05.1) O sistema exibe uma mensagem de alerta.
Pós-condição	As configurações devem estar estabelecidas.

Quadro 13 - Caso de uso 02

UC03 – Criar Modelo de Classificação	
Pré-condições	As configurações devem estar estabelecidas.
Cenário Principal	01) O usuário pressiona o botão criar. 02) O sistema confere as configurações. 03) O sistema exibe no painel de resultado o modelo de classificação criado.
Exceção 01	No passo 02, se estiver faltando alguma configuração: 02.1) O sistema FTree exibe uma mensagem de alerta. 02.2) O sistema FTree cancela o processo.
Pós-condição	O modelo de classificação deve estar criado.

Quadro 14 - Caso de uso 03

UC04 – Visualizar as Regras de classificação	
Pré-condições	O modelo de classificação deve estar criado
Cenário Principal	01) O modelo de classificação será exibido no painel de resultados
Exceção 01	No passo 01, ocorreu erro ao criar o modelo de classificação: 01.1) O sistema FTree exibe uma mensagem no painel de resultado.
Pós-condição	O modelo é exibido no painel de resultado

Quadro 15 - Caso de uso 04

UC05 – Salvar Modelo de Classificação	
Pré-condições	O modelo de classificação deve estar criado
Cenário Principal	01) O usuário informa o arquivo a ser salvo. 02) O sistema salva o modelo no arquivo.
Exceção 01	No passo 02, se o arquivo for inválido: 02.1) O sistema FTree exibe uma mensagem de alerta. 02.2) O sistema cancela o processo.
Pós-condição	O modelo deve estar salvo em um arquivo de texto.

Quadro 16 - Caso de uso 05

3.2.2 Diagrama de atividades

O diagrama de atividades da Figura 10 representa o fluxo da ferramenta desenvolvida, para se obter um modelo de classificação.

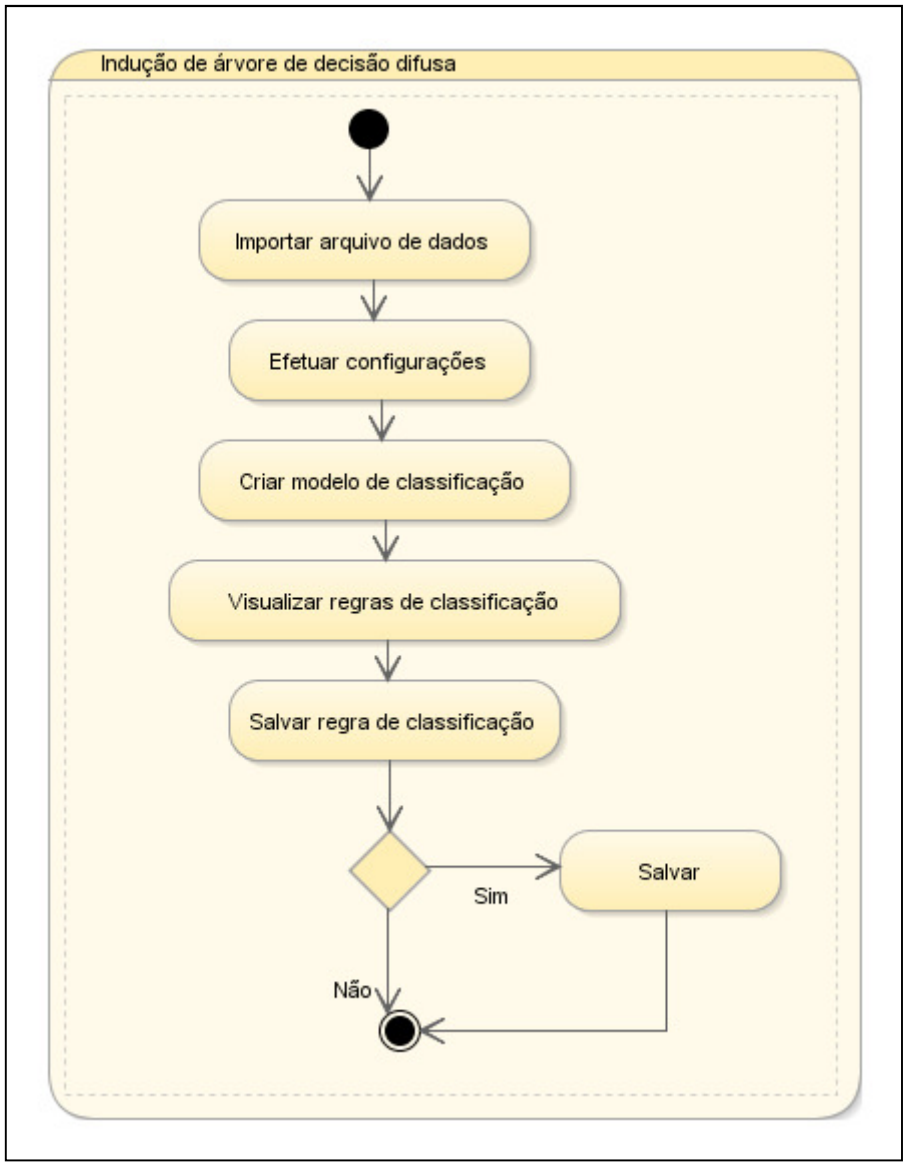


Figura 10 - Diagrama de Atividades

3.2.3 Diagramas de classe

O diagrama de classe é uma representação da estrutura e da relação das classes de um sistema. Esta subseção apresenta às classes desenvolvidas no projeto, na Figura 11 de forma compacta as classes são apresentadas. Na Figura 12 são apresentadas as classes do pacote `dao`, que são responsáveis pela conexão com o banco de dados e com a criação da base de dados.

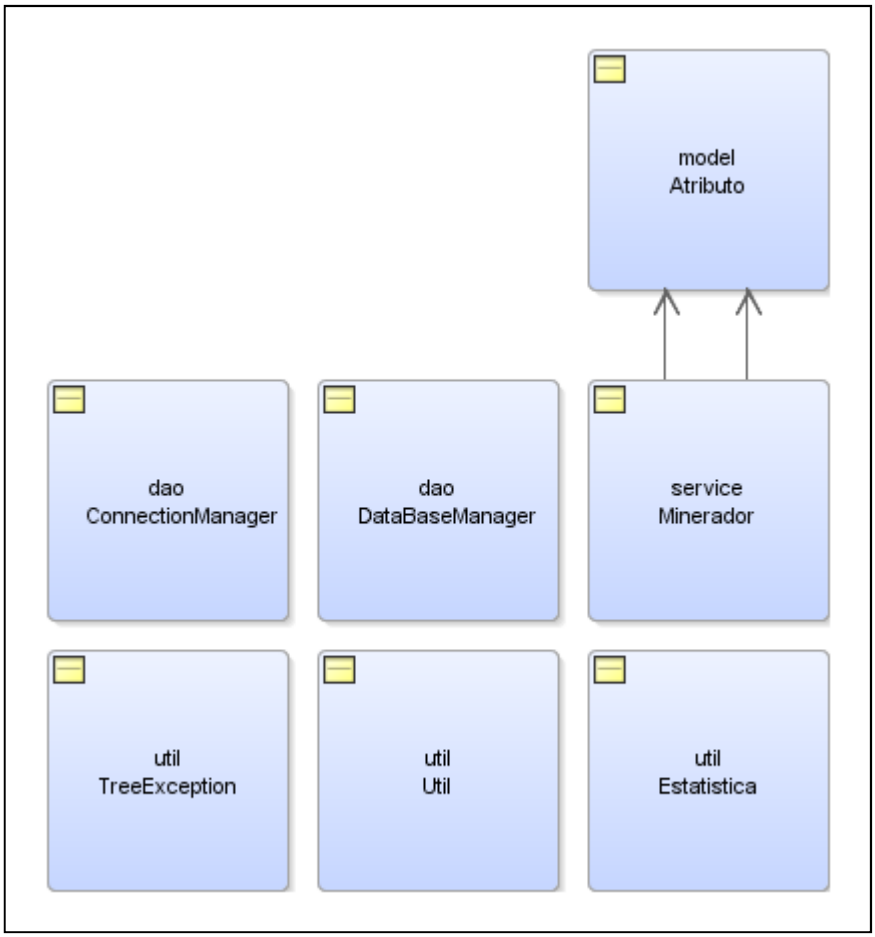


Figura 11 - Diagrama de Classes

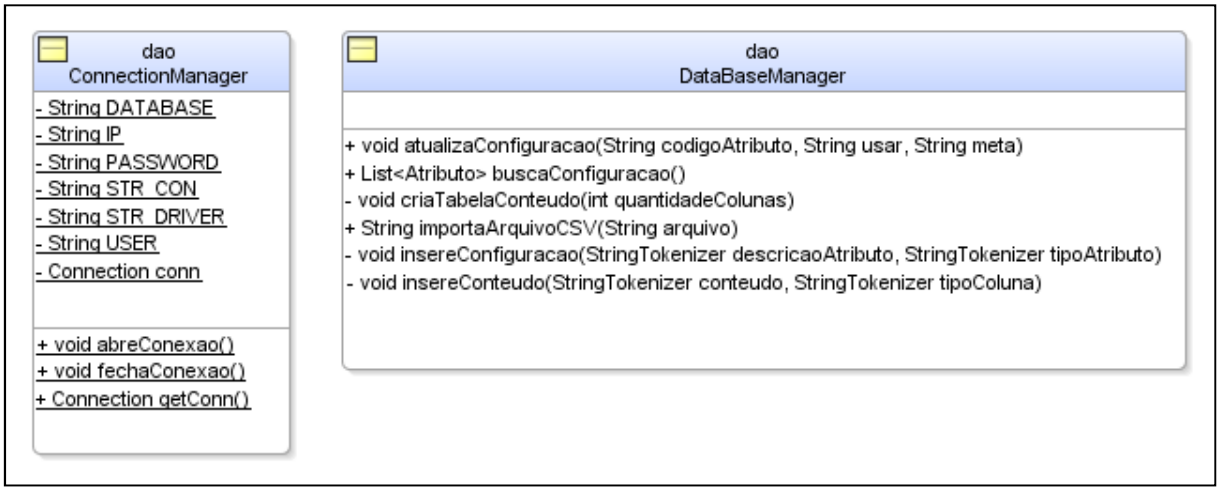


Figura 12 - Diagrama de classes do pacote `dao`

A classe `ConnectionManager`, é responsável por fazer o acesso a base de dados. Por se tratar de uma classe que possui atributos e método estáticos, não é necessário instanciá-la. No momento em que for necessário acessar a base de dados devem-se invocar os métodos correspondentes. Seus principais atributos e métodos são explicados no Quadro 17.

ConnectionManager	
Atributo	Descrição
DATABASE	Nome da base de dados.
IP	Servidor do banco de dados
PASSWORD	Senha para acessar a base de dados
STR_DRIVE	*Driver* de acessar a base de dados
USER	Usuário para acessar a base de dados
Método	Descrição
abreConexao	Abre a conexão com a base de dados
fechaConecao	Fecha a conexão com a base de dados
getConexao	Retorna a conexão com o banco de dados

Quadro 17- Principais métodos e atributos da classe `ConnectionManager`

A classe "DataBaseManager", é responsável por preparar a base de dados para a execução do processo de indução de árvore de decisão. Seus principais métodos são explicados no Quadro 18.

DataBaseManager	
Método	Descrição
atualizaConfiguracao	Atualiza registro de configuração na base de dados
buscaConfiguracao	Lista as configurações da base de dados
criaTabelaConteudo	Cria a tabela de conteúdo conforme a especificação do arquivo CSV
importaArquivoCSV	Importa o arquivo CSV para a base de dados
insereConfiguracao	Adiciona registros de configuração na base de dados
insereConteudo	Adiciona registros de conteúdo na base de dados

Quadro 18 - Principais métodos da classe `DataBaseManager`

Na Figura 13 são apresentadas as classes do pacote `util`, que são responsáveis por salvar um arquivo de texto, aplicar dados numéricos a fórmulas estatísticas e em personalizar exceções do sistema.

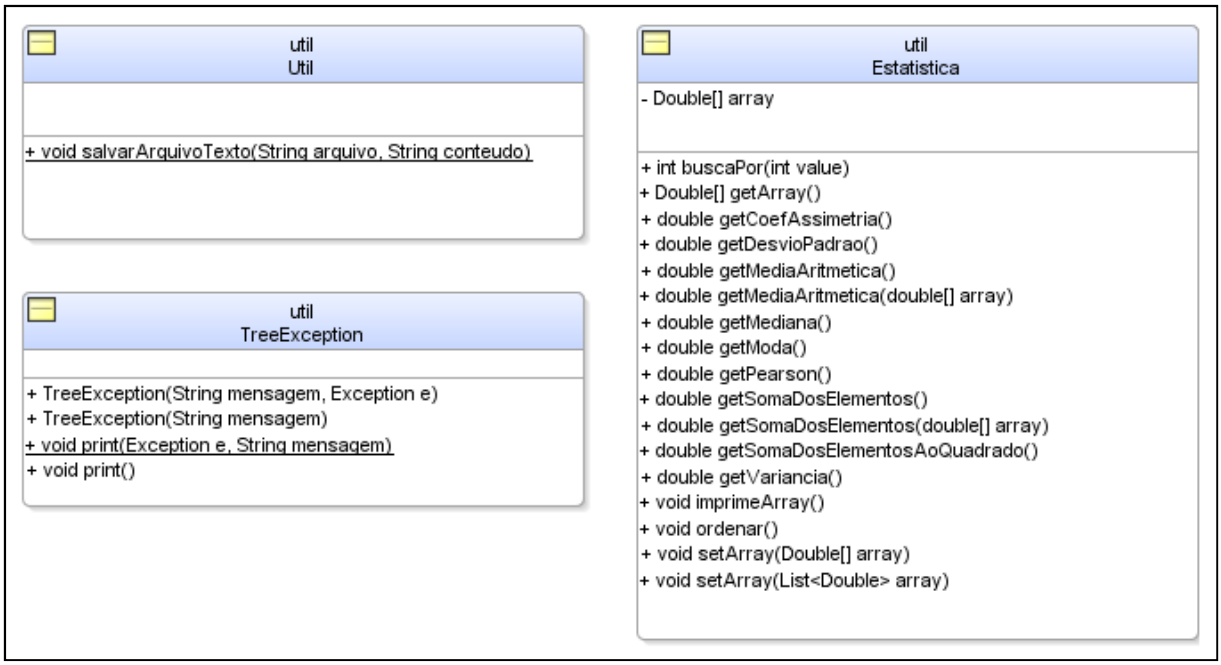


Figura 13 - Diagrama de classes do pacote `util`

A classe `Util` é responsável por salvar um arquivo de texto. Por ser uma classe estática não é necessário ser instanciada. Quando se deseja salvar um texto em arquivo basta invocar seu método `salvarArquivo`.

Na Figura 14 é apresentada a classe do pacote `model` que é responsável por representar o metadados do sistema. Também é apresentada sua relação com a classe do pacote `service`.

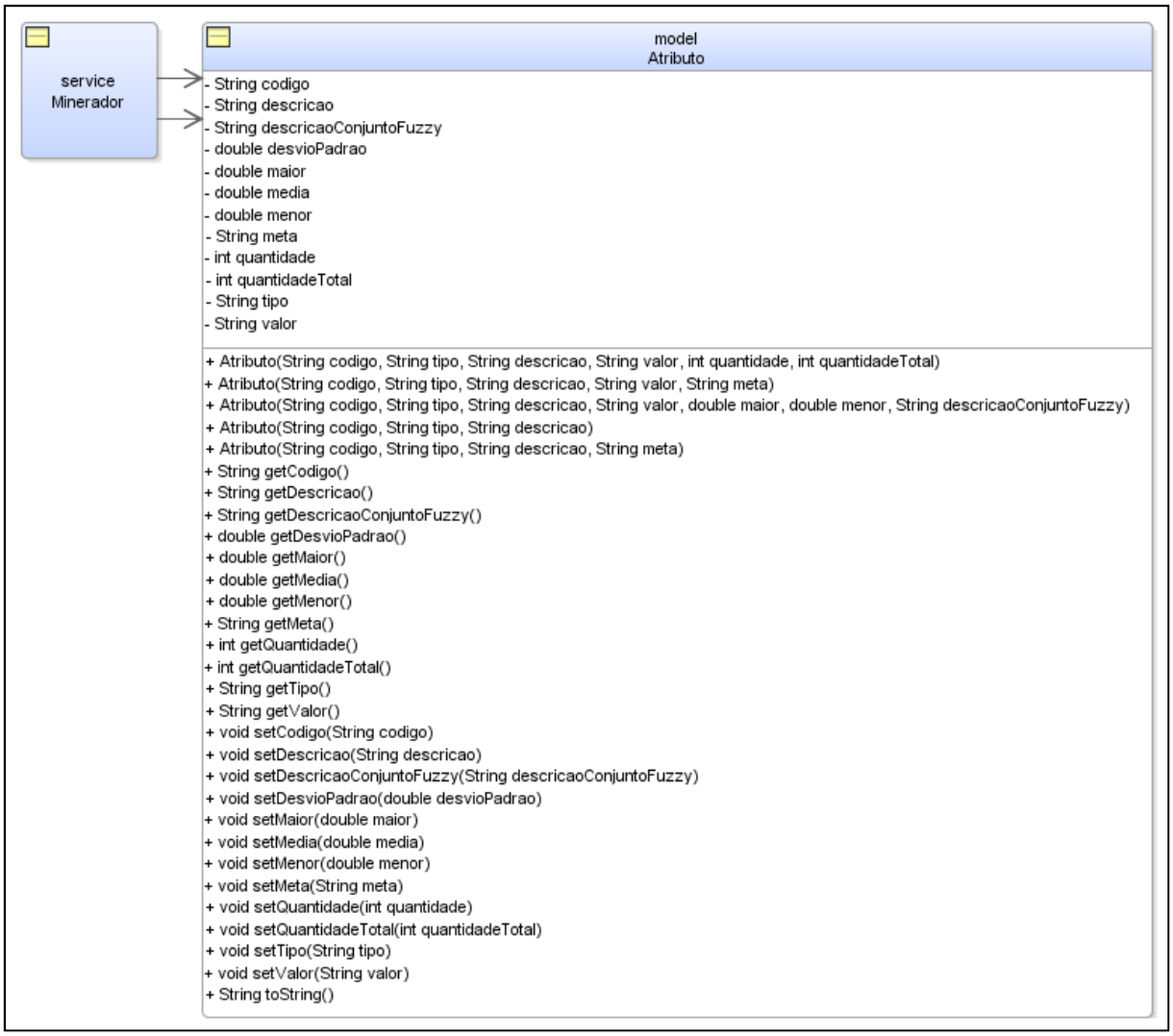


Figura 14 - Diagrama de classes do pacote `model`

A classe `Atributo` é responsável por representar as características dos atributos durante a execução do sistema. Seus atributos são explicados no Quadro 19.

Atributo	
Atributo	Descrição
codigo	Código do atributo na base de dados.
descricao	Descrição do atributo na base de dados.
descricaoConjuntoFuzzy	Descrição do conjunto difuso
desvioPadrao	Desvio padrão do atributo na base de dados.
maior	Maior atributo na base de dados
media	Média do atributo na base de dados.
menor	Menor atributo na base de dados
meta	Identificação se é atributo meta.
quantidade	Quantidade de registros da classe do atributo na base de dados.
quantidadeTotal	Quantidade total de registros na do atributo na base de dados.
tipo	Tipo de dado do atributo.
valor	Valor do atributo.

Quadro 19- Principais atributos da classe `Atributo`

Na Figura 15 é apresentada a classe do pacote `service` que é responsável por executar o processo de indução de árvore de decisão difusa. Também é apresentada sua relação com a classe do pacote `model`.

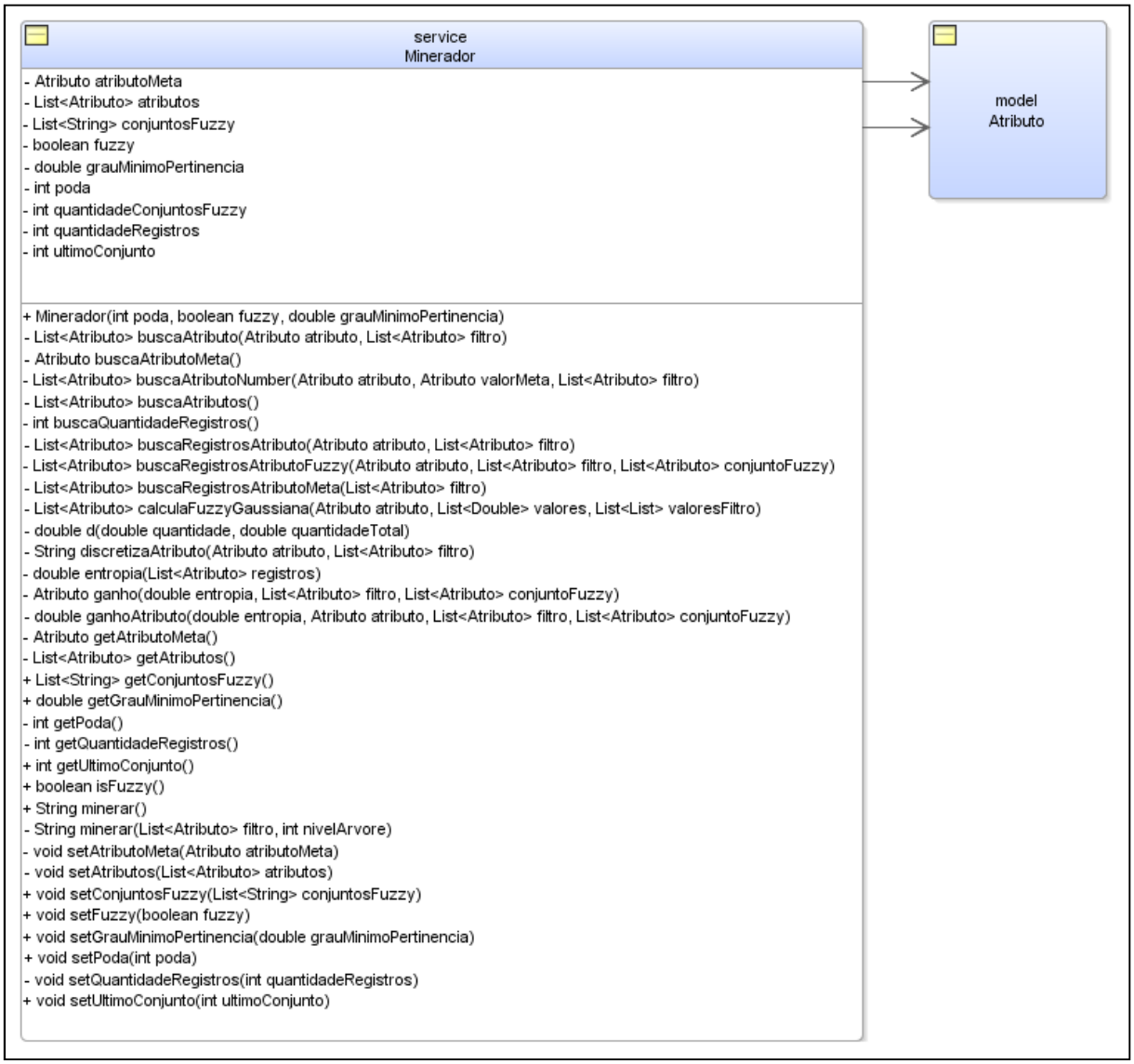


Figura 15 - Diagrama de classes do pacote `service`

A classe `Minerador` é responsável pela execução do processo de indução da árvore de decisão difusa. Seus principais atributos e métodos são explicados no Quadro 20.

Minerador	
Atributo	Descrição
atributoMeta	Uma instância do atributo meta.
atributos	Lista com os atributos que são utilizados no processo de indução de árvore de decisão.
conjuntosFuzzy	Lista com os conjuntos difusos utilizados no processo de indução de árvore de decisão.
fuzzy	Indica se é utilizado a teoria dos conjuntos difusos no processo de indução da árvore de decisão.
grauMinimoPertinencia	Grau de pertinência aceito no processo de indução de árvore de decisão difusa.
poda	Valor máximo de níveis que a árvore pode possuir.
quantidadeConjuntosFuzzy	Quantidade de conjuntos difusos utilizados no processo de indução de árvore de decisão difusa.
ultimoConjunto	Ultimo conjunto difuso utilizado no processo de indução de árvore de decisão.
quantidadeRegistros	Quantidade total de registros no banco de dados.
conjuntosFuzzy	Armazena os conjuntos difusos criados na execução.
ultimoConjunto	Indica que foi o último conjunto difuso criado.
Método	Descrição
buscaAtributo	Busca um atributo de acordo com a sua classe.
buscaRegistrosAtributo	Busca os registros de um atributo de acordo com a sua classe.
buscaAtributoMeta	Busca o atributo meta de acordo com a sua classe.
buscaRegistrosAtributoMeta	Busca os registros do atributo meta de acordo com a sua classe.
discretizaAtributo	Discretiza um atributo de acordo com a sua classe.
entropia	Calcula a entropia de uma lista de atributos.
ganho	Calcula o ganho dos atributos.
ganhoAtributo	Calcula o ganho de um atributo.

Quadro 20 - Principais atributos e métodos da classe `Minerador`

3.2.4 Modelo de entidades e relacionamentos

A Figura 16 apresenta o modelo de entidade e relacionamento que representa as unidades que serão persistidas no banco de dados.

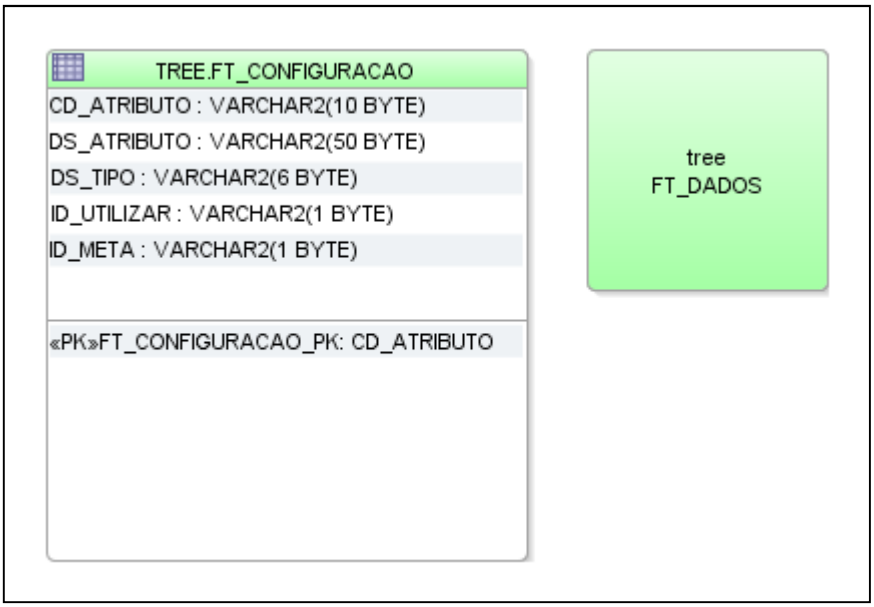


Figura 16 - Modelo de entidade e relacionamento

A entidade `FT_CONFIGURACAO` é responsável por armazenar e manter as informações de configuração do conjunto de dados da entidade `FT_DADOS`. Para cada coluna existente na entidade `FT_DADOS` existe um registro de configuração na entidade `FT_CONFIGURACAO`. Suas colunas são explicadas no Quadro 21.

FT_CONFIGURACAO	
Coluna	Descrição
CD_ATRIBUTO	Chave primaria da tabela.
DS_ATRIBUTO	Descrição do atributo.
DS_TIPO	Tipo do atributo (NUMBER ou CHAR).
ID_UTILIZAR	Identifica se o atributo deve ser utilizado Y ou N.
ID_META	Identifica de atributo meta deve ser utilizado Y ou N.

Quadro 21 - Colunas da tabela `FT_CONFIGURACAO`

A entidade `FT_DADOS` é responsável por armazenar o conjunto de dados que será submetido ao processo de indução da árvore de decisão. Trata-se de uma entidade dinâmica, pois suas colunas são criadas conforme a necessidade do conjunto de dados que será armazenado. Suas colunas são criadas com nome de `ATRIBUTO` seguido de um número de 1 até n-1, conforme a necessidade. O tipo de dado das colunas criadas é por padrão `VARCHAR2` com tamanho de 50 bytes.

3.2.5 Arquivo de Dados CSV

O arquivo com o conjunto de dados a ser importado na base de dados do sistema “FTree”, deve ser um arquivo no formato CSV e que deve possuir as seguintes características:

a) os *tokens* do arquivo devem estar separados por ponto e vírgula (“;”).
b) a primeira linha do arquivo deve conter uma descrição de cada *token*.
c) a segunda linha do arquivo deve conter o tipo de dado de cada *token*. O tipo de dados é restrito a “NUMBER” e “CHAR”.
d) a partir da terceira linha as informações devem estar dispostas em quantas linhas forem necessárias.

Um exemplo de arquivo é apresentado no Quadro 22.

```
TEMPO;TEMPERATURA;UMIDADE;VENTO;JOGA
CHAR;NUMBER;NUMBER;CHAR;CHAR
Sol;85;85;Não;Não
Sol;80;90;Sim;Não
Nublado;83;86;Não;Sim
Chuva;70;96;Não;Sim
Chuva;68;80;Não;Sim
Chuva;65;70;Sim;Não
Nublado;64;65;Sim;Sim
Sol;72;95;Não;Não
Sol;69;70;Não;Sim
Chuva;75;80;Não;Sim
Sol;75;71;Sim;Sim
Nublado;72;90;Sim;Sim
Nublado;81;75;Não;Sim
Chuva;71;91;Sim;Não
```

Quadro 22 - Exemplo de arquivo de dados CSV

3.3 IMPLEMENTAÇÃO

Nesta seção são apresentadas informações sobre as ferramentas utilizadas no desenvolvimento do sistema `FTree`, juntamente com a operacionalidade da implementação.

3.3.1 Técnicas e ferramentas utilizadas

Para desenvolver o sistema foram utilizadas as ferramentas:

- Oracle JDeveloper 11g;
- como *container web*, o Apache TomCat;
- Oracle Express Edition como banco de dados;
- como *front-end* foi utilizado Adobe Flex;
- BlazeDS para a comunicação entre o Java e o Adobe Flex.

3.3.2 Funcionamento da implementação

Nesta seção é apresentada a correta seqüência de operações necessárias para realizar a indução de uma árvore de decisão difusa. Também serão apresentados fragmentos de código fonte de rotinas do sistema.

Ao acessar a aplicação, o usuário é direcionado a tela principal como pode ser observado na Figura 17.

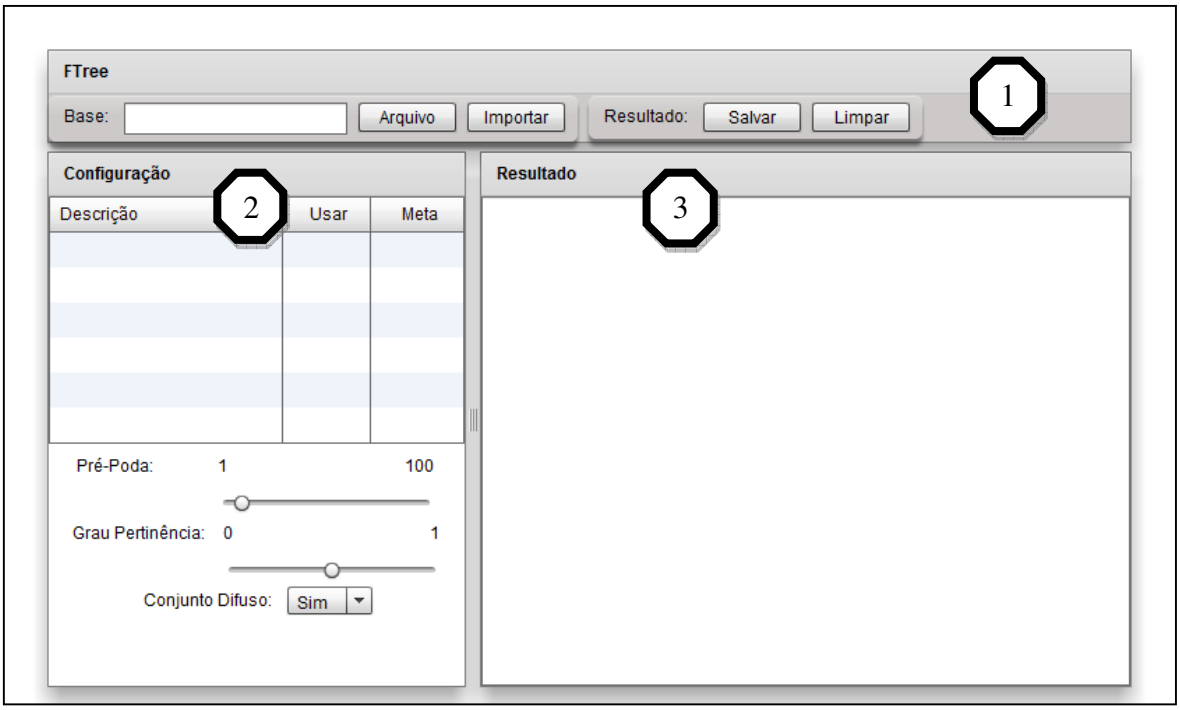


Figura 17 - Tela principal da aplicação

Esta tela é composta por três diferentes áreas, o cabeçalho (1), painel de configuração (2) e painel de resultados (3).

No cabeçalho (1), além do título há uma pequena barra de ferramentas. Nesta barra é possível localizar um arquivo (Figura 18) e importá-lo. Também é possível salvar em um arquivo de texto o conteúdo do painel de resultado (Figura 19).

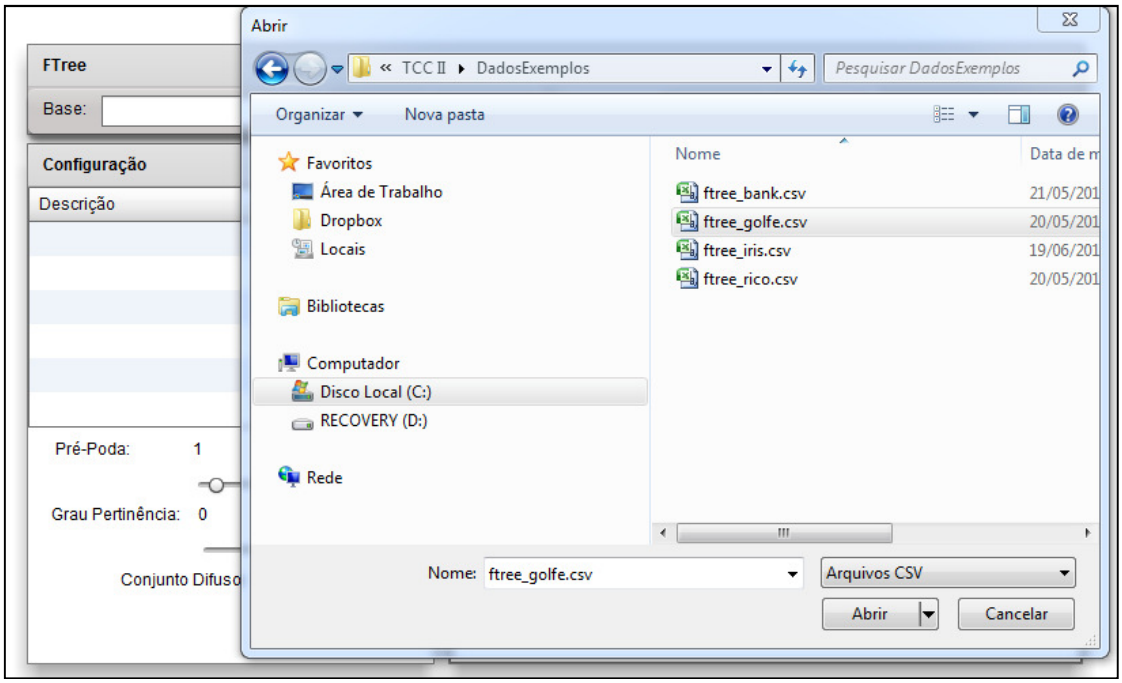


Figura 18 - Selecionar arquivo

O Quadro 23 demonstra trecho do código executado para importar o arquivo CSV para a base de dados da aplicação.

```
BufferedReader br = new BufferedReader(new FileReader(file));

//pega a primeira linha do arquivo que deve conter o cabeçalho das colunas
descricaoAtributo = br.readLine();
descricaoAtributoToken = new StringTokenizer(descricaoAtributo, token);
//pega a segunda linha do arquivo que deve conter o tipo das colunas
tipoAtributo = br.readLine();
tipoAtributoToken = new StringTokenizer(tipoAtributo, token);

//verifica se o cabeçalho do arquivo está correto
if (descricaoAtributoToken.countTokens() > 0 && tipoAtributoToken.countTokens() > 0 &&
    descricaoAtributoToken.countTokens() == tipoAtributoToken.countTokens()) {
    //numero de colunas que cada linha deve possuir
    totalColunas = descricaoAtributoToken.countTokens();
    //insere as configurações
    this.insereConfiguracao(descricaoAtributoToken, tipoAtributoToken);
} else {
    throw new TreeException("Problema no cabeçalho do arquivo!");
}

//antes de importa o arquivo cria a tabela
this.criaTabelaConteudo(totalColunas);

//importa os dados do arquivo
linha = br.readLine();
while (linha != null) {
    linhaToken = new StringTokenizer(linha, token);
    if (linhaToken.countTokens() != totalColunas) {
        throw new TreeException("Problema no conteúdo do arquivo!");
    }
    this.insereConteudo(linhaToken, new StringTokenizer(tipoAtributo, token));
    linha = br.readLine();
}
br.close();
```

Quadro 23 - Fragmento código fonte que importa um arquivo

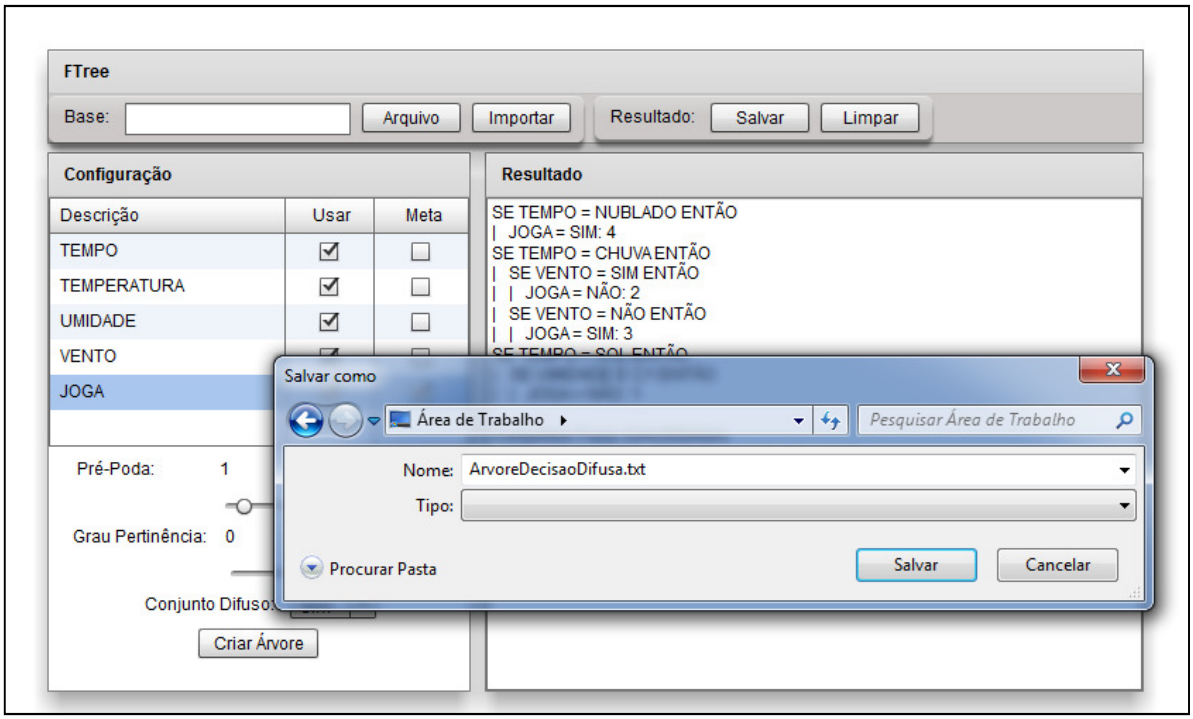


Figura 19 - Salvar painel de resultado em arquivo de texto

No painel de configuração (Figura 20), após importar um arquivo para a base de dados da aplicação, são executadas as configurações para a execução do processo de indução de árvore de decisão difusa. A única configuração obrigatória é selecionar o atributo meta. Outras configurações como escolher os atributos que serão utilizados pelo processo de indução de árvore de decisão difusa, definir um número de pré-poda (número máximo de níveis que a árvore pode atingir), e definir o grau de pertinência que será utilizado para avaliação dos conjuntos difusos, também devem ser realizados neste momento se necessário.

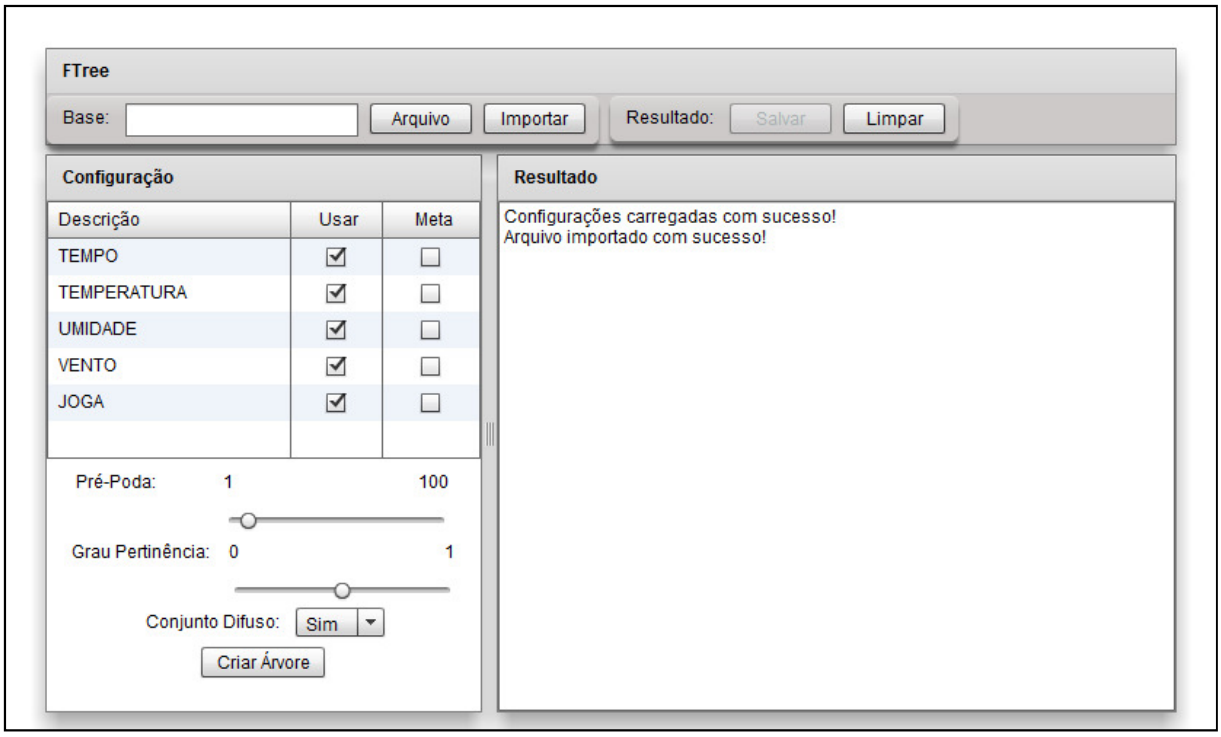


Figura 20 - Tela principal painel configuração

O Quadro 24 demonstra fragmento do código que é executado ao realizar modificações nas configurações.

```
boolean minhaConexao = false;
try {
    minhaConexao = ConnectionManager.abreConexao();
    if (meta.toUpperCase().equals("Y")) {
        strUpdate = new StringBuffer();
        strUpdate.append("UPDATE ft_configuracao");
        strUpdate.append("   SET id_meta = 'N'");

        ps = ConnectionManager.getConexao().prepareStatement("SQL");
        ps.executeUpdate(strUpdate.toString());
        ps.close();
    }
    //monta o update
    strUpdate = new StringBuffer();
    strUpdate.append("UPDATE ft_configuracao");
    strUpdate.append("   SET id_utilizar = '" + usar.toUpperCase() + "'");
    strUpdate.append("     , id_meta     = '" + meta.toUpperCase() + "'");
    strUpdate.append(" WHERE cd_atributo = '" + codigoAtributo + "'");

    ps = ConnectionManager.getConexao().prepareStatement("SQL");
    ps.executeUpdate(strUpdate.toString());
    ps.close();

} catch (SQLException e) {
    e.printStackTrace();
} catch (TreeException e) {
    e.printStackTrace();
} finally {
    ConnectionManager.fechaConexao(minhaConexao);
}
```

Quadro 24 - Fragmento de código fonte que atualiza as configurações

Neste painel também se encontra o botão para executar o processo de indução da árvore de decisão difusa. Se ao acionar o botão o atributo meta não estiver marcado o usuário será notificado conforme a Figura 21.

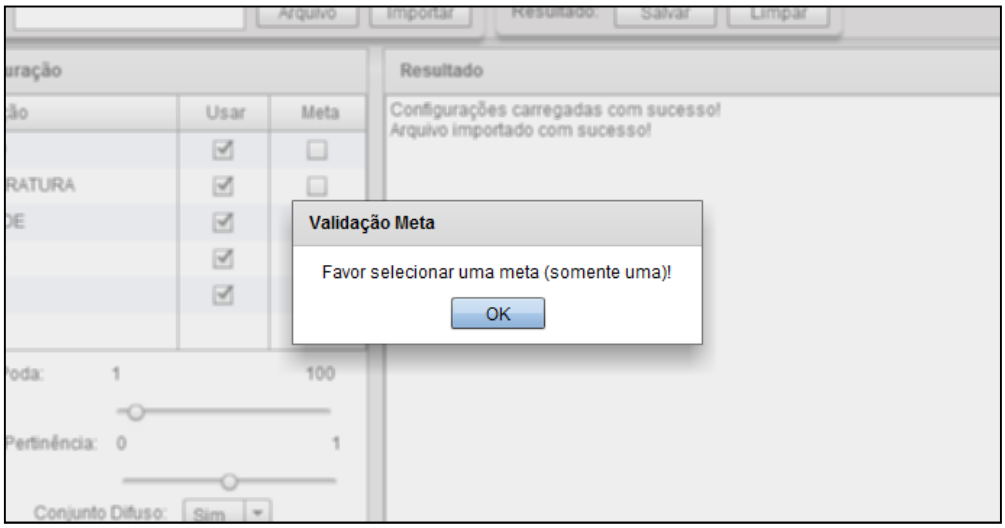


Figura 21 - Notificação ao usuário de atributo meta

O painel de resultados tem como objetivo apresentar a árvore de decisão difusa

encontrada na execução do processo e os conjuntos difusos pertencente à árvore de decisão difusa. A representação dos conjuntos difusos ocorre através de seu atributo e o nome do conjunto, seguido da média e do desvio padrão que são utilizados para definir os elementos do conjunto (Figura 22). Além disto, também serve para apresentar mensagens ao usuário durante utilização da aplicação (Figura 20).

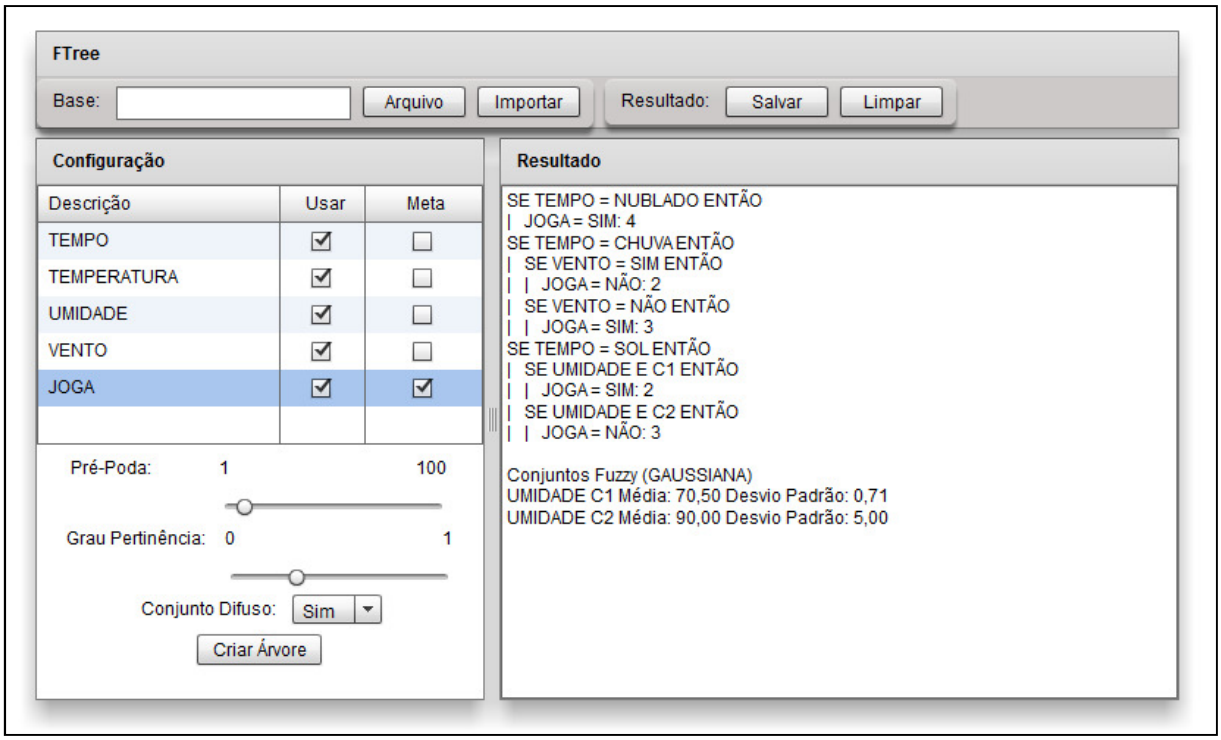


Figura 22 - Painel de resultado

O Quadro 25 demonstra um fragmento do código que é executado para iniciar a indução de árvore de decisão difusa.

```
public String minerar() {
    //variaveis para auxiliar calculo
    List<Atributo> registrosEntropia = null;
    double valorEntropia = 0;
    Atributo a = null;
    StringBuffer sbResultado = new StringBuffer();

    try {
        ConnectionManager.abreConexao();


        //seta atributos utilizados na mineração da arvore
        this.setAtributoMeta(this.buscaAtributoMeta());
        this.setAtributos(this.buscaAtributos());
        this.setQuantidadeRegistros(this.buscaQuantidadeRegistros());

        // ***********************************************
        // ******* INICIA A MINERAÇÃO NESTE PONTO *******
        // ***********************************************

        sbResultado.append(minerar(null, 0));

        // ***********************************************
        // ******* TERMINA A MINERAÇÃO NESTE PONTO *******
        // ***********************************************

        if (this.isFuzzy()) {
            sbResultado.append("\n");
            sbResultado.append("Conjuntos Fuzzy (GAUSSIANA)\n");
            for (String s : this.getConjuntosFuzzy()) {
                sbResultado.append(s + "\n");
            }
        }
    } catch (TreeException e) {
        e.printStackTrace();
    } finally {
        ConnectionManager.fechaConexao();
    }
    return sbResultado.toString();
}
```

Quadro 25- Fragmento de código fonte que executa o inicio da indução de árvore de decisão difusa

O fragmento de código fonte do Quadro 26, é responsável pela execução do processo recursivo de indução de árvore de decisão difusa. Este código executa os seguintes passos:

a) teste da condição de parada;

b) calcula o nó da árvore;

c) adiciona os ramos na árvore;

d) adiciona folhas na árvore.

```
private String minerar(List<Atributo> filtro, int nivelArvore) {
    //variaveis para auxiliar calculo
    List<Atributo> registros = null;
    double valorEntropia = 0;
    Atributo melhorGanho = null;
    ArrayList<Atributo> novoFiltro = null;
    List<Atributo> valoresMelhorGanho = null;
    StringBuffer sb = new StringBuffer();
    List<Atributo> conjuntoFuzzy = new ArrayList<Atributo>();
    //busca os registro para calculo do nó
    registros = (ArrayList<Atributo>)this.buscaRegistrosAtributoMeta(filtro);
    //se não encontrou um informaçoes para continuar
    if (registros == null) {
        return "";
    }
    //verifica o numero maximo de niveis e se encontro o fim
    if (nivelArvore < this.getPoda() && registros.size() > 1) {
        //calcula a entropia do atributo meta
        valorEntropia = this.entropia(registros);
        //calcula o melhor ganho
        melhorGanho = this.ganho(valorEntropia, filtro, conjuntoFuzzy);
        if (isFuzzy() && melhorGanho.getTipo().equals("NUMBER")) {
            valoresMelhorGanho = new ArrayList<Atributo>();
            if (conjuntoFuzzy != null) {
                for (Atributo a : conjuntoFuzzy) {
                    String numeroConjunto = "C" + ++quantidadeConjuntosFuzzy;
                    String descricaConjunto = melhorGanho.getDescricao() + " " +
                                              numeroConjunto + " Média: " +
                                              (new DecimalFormat("0.00").format(a.getMedia())) +
                                              " Desvio Padrão: " +
                                              (new DecimalFormat("0.00").format(a.getDesvioPadrao()));
                    valoresMelhorGanho.add(new Atributo(melhorGanho.getCodigo(), melhorGanho.getTipo(),
                                                        melhorGanho.getDescricao(), a.getValor(),
                                                        a.getMaior(), a.getMenor(), numeroConjunto));
                    this.getConjuntosFuzzy().add(descricaConjunto);
                }
            }
        } else {
        //itera pelos ramos do nó
        for (Iterator i = valoresMelhorGanho.iterator(); i.hasNext(); ) {
            if (filtro != null) {
                novoFiltro = new ArrayList<Atributo>(filtro);
            } else {
                novoFiltro = new ArrayList<Atributo>();
            }
            Atributo a = (Atributo)i.next();
            novoFiltro.add(a);
            for (int ii = 0; ii < nivelArvore; ii++) {
                sb.append("|   ");
            }
            if (a.getTipo().equals("NUMBER")) {
                if (isFuzzy()) {
                    sb.append("SE " + a.getDescricao() + " E " + a.getDescricaoConjuntoFuzzy() + " ENTÃO\n");
                } else {
                    sb.append("SE " + a.getDescricao() + " " + a.getValor().replace("'", " ") + " ENTÃO\n");
                }
            } else {
                sb.append("SE " + a.getDescricao() + " = " + a.getValor() + " ENTÃO\n");
            }
            //chamada recursiva
            sb.append(this.minerar(novoFiltro, nivelArvore + 1));
        }
        //se for o último nível da árvore ou se houver somente um ramo
    } else {
        for (Iterator i = registros.iterator(); i.hasNext(); ) {
            Atributo a = (Atributo)i.next();
            for (int ii = 0; ii < nivelArvore; ii++) {
                sb.append("|   ");
            }
            if (a.getTipo().equals("NUMBER")) {
                if (isFuzzy()) {
                    sb.append(a.getDescricao() + " E " + a.getValor() + ": " + a.getQuantidade() + "\n");
                } else {
                    sb.append(a.getDescricao() + " " + a.getValor() + ": " + a.getQuantidade() + "\n");
                }
            } else {
                sb.append(a.getDescricao() + " = " + a.getValor() + ": " + a.getQuantidade() + "\n");
            }
        }
    }
    return sb.toString();
}
```

Quadro 26 - Fragmento de código fonte que executa a indução de árvore de decisão difusa

3.3.3 Cálculo de ganho de informação com lógica difusa

Nesta seção é apresentado o cálculo de ganho de informação utilizando a lógica difusa para suportar atributos contínuos. Para ilustrar será utilizado o atributo da "Temperatura" do conjunto de dados da Tabela 1.

Para executar o cálculo as seguintes etapas são executadas:

a) calcular o valor da média e desvio padrão para cada classe (Quadro 27), neste exemplo, por se estar no nó raiz, todo o conjunto de dados será utilizado, desta forma resultando em dois conjuntos o da classe Jogar = Sim e da classe Jogar = Não;

Conjunto 1 (Jogar = Sim) = {média = 73; desvio padrão = 6,1644}
Conjunto 2 (Jogar = Não) = {média = 74,6; desvio padrão = 7,8930}

Quadro 27 – Cálculo dos conjuntos

b) este é momento que ocorre a etapa fuzificador (Figura 7), que é calcular para cada conjunto, obtido no passo anterior, o grau de pertinência de cada elemento pertencente ao conjunto de dados em avaliação (Quadro 28), utilizando a fórmula do Quadro 9;

Variável:
e = logaritmo neperiano.

Conjunto 1 (x) = {e ^ -[(x - média)^2]/[desvio padrão ^ 2]}
Conjunto 1 (x = 83; Joga = Sim) = {e ^ -[(83 - 73)^2]/[6,1644 ^ 2]} = 0,0719
Conjunto 1 (x = 70; Joga = Sim) = {e ^ -[(70 - 73)^2]/[6,1644 ^ 2]} = 0,7891
Conjunto 1 (x = 68; Joga = Sim) = {e ^ -[(68 - 73)^2]/[6,1644 ^ 2]} = 0,5179
Conjunto 1 (x = 64; Joga = Sim) = {e ^ -[(64 - 73)^2]/[6,1644 ^ 2]} = 0,1186
Conjunto 1 (x = 69; Joga = Sim) = {e ^ -[(69 - 73)^2]/[6,1644 ^ 2]} = 0,6563
Conjunto 1 (x = 75; Joga = Sim) = {e ^ -[(75 - 73)^2]/[6,1644 ^ 2]} = 0,9000
Conjunto 1 (x = 75; Joga = Sim) = {e ^ -[(75 - 73)^2]/[6,1644 ^ 2]} = 0,9000
Conjunto 1 (x = 72; Joga = Sim) = {e ^ -[(72 - 73)^2]/[6,1644 ^ 2]} = 0,9740
Conjunto 1 (x = 81; Joga = Sim) = {e ^ -[(81 - 73)^2]/[6,1644 ^ 2]} = 0,1855
Conjunto 1 (x = 85; Joga = Não) = {e ^ -[(85 - 73)^2]/[6,1644 ^ 2]} = 0,0226
Conjunto 1 (x = 80; Joga = Não) = {e ^ -[(80 - 73)^2]/[6,1644 ^ 2]} = 0,2754
Conjunto 1 (x = 65; Joga = Não) = {e ^ -[(65 - 73)^2]/[6,1644 ^ 2]} = 0,1855
Conjunto 1 (x = 72; Joga = Não) = {e ^ -[(72 - 73)^2]/[6,1644 ^ 2]} = 0,9740
Conjunto 1 (x = 71; Joga = Não) = {e ^ -[(71 - 73)^2]/[6,1644 ^ 2]} = 0,9000

Conjunto 2 (x) = {e ^ -[(x - média)^2]/[desvio padrão ^ 2]}
Conjunto 2 (x = 83; Joga = Sim) = {e ^ -[(83 - 74,6)^2]/[7,8930 ^ 2]} = 0,3221
Conjunto 2 (x = 70; Joga = Sim) = {e ^ -[(70 - 74,6)^2]/[7,8930 ^ 2]} = 0,7120
Conjunto 2 (x = 68; Joga = Sim) = {e ^ -[(68 - 74,6)^2]/[7,8930 ^ 2]} = 0,4969
Conjunto 2 (x = 64; Joga = Sim) = {e ^ -[(64 - 74,6)^2]/[7,8930 ^ 2]} = 0,1647
Conjunto 2 (x = 69; Joga = Sim) = {e ^ -[(69 - 74,6)^2]/[7,8930 ^ 2]} = 0,6044
Conjunto 2 (x = 75; Joga = Sim) = {e ^ -[(75 - 74,6)^2]/[7,8930 ^ 2]} = 0,9974
Conjunto 2 (x = 75; Joga = Sim) = {e ^ -[(75 - 74,6)^2]/[7,8930 ^ 2]} = 0,9974
Conjunto 2 (x = 72; Joga = Sim) = {e ^ -[(72 - 74,6)^2]/[7,8930 ^ 2]} = 0,8971
Conjunto 2 (x = 81; Joga = Sim) = {e ^ -[(81 - 74,6)^2]/[7,8930 ^ 2]} = 0,5181
Conjunto 2 (x = 85; Joga = Não) = {e ^ -[(85 - 74,6)^2]/[7,8930 ^ 2]} = 0,1762
Conjunto 2 (x = 80; Joga = Não) = {e ^ -[(80 - 74,6)^2]/[7,8930 ^ 2]} = 0,6262
Conjunto 2 (x = 65; Joga = Não) = {e ^ -[(65 - 74,6)^2]/[7,8930 ^ 2]} = 0,2277
Conjunto 2 (x = 72; Joga = Não) = {e ^ -[(72 - 74,6)^2]/[7,8930 ^ 2]} = 0,8971
Conjunto 2 (x = 71; Joga = Não) = {e ^ -[(71 - 74,6)^2]/[7,8930 ^ 2]} = 0,8121

Quadro 28 – Cálculo de pertinência dos elementos da classe para cada conjunto

c) neste momento ocorre a etapa defuzificador (Figura 7), que é a avaliação dos elementos pertencentes a cada conjunto através da aplicação da regra de aceitação definida, neste trabalho é o grau mínimo de pertinência aceito. Os resultados são subdividindo-os por classe (Quadro 29);

Variáveis:
y = 0,3 (grau de pertinência configurável, ver Figura 20).
Regra:
Se (grau de pertinência > y) então (pertence) senão (não pertence).

Conjunto 1 (x = 83; Joga = Sim) = 0,0719 > y = não pertence
Conjunto 1 (x = 70; Joga = Sim) = 0,7891 > y = pertence
Conjunto 1 (x = 68; Joga = Sim) = 0,5179 > y = pertence
Conjunto 1 (x = 64; Joga = Sim) = 0,1186 > y = não pertence
Conjunto 1 (x = 69; Joga = Sim) = 0,6563 > y = pertence
Conjunto 1 (x = 75; Joga = Sim) = 0,9000 > y = pertence
Conjunto 1 (x = 75; Joga = Sim) = 0,9000 > y = pertence
Conjunto 1 (x = 72; Joga = Sim) = 0,9740 > y = pertence
Conjunto 1 (x = 81; Joga = Sim) = 0,1855 > y = não pertence
Conjunto 1 (x = 85; Joga = Não) = 0,0226 > y = não pertence
Conjunto 1 (x = 80; Joga = Não) = 0,2754 > y = não pertence
Conjunto 1 (x = 65; Joga = Não) = 0,1855 > y = não pertence
Conjunto 1 (x = 72; Joga = Não) = 0,9740 > y = pertence
Conjunto 1 (x = 71; Joga = Não) = 0,9000 > y = pertence

Totais por classe:
Conjunto 1 (Joga = Sim) = 6
Conjunto 1 (Joga = Não) = 2

Conjunto 2 (x = 83; Joga = Sim) = 0,3221 > y = pertence
Conjunto 2 (x = 70; Joga = Sim) = 0,7120 > y = pertence
Conjunto 2 (x = 68; Joga = Sim) = 0,4969 > y = pertence
Conjunto 2 (x = 64; Joga = Sim) = 0,1647 > y = não pertence
Conjunto 2 (x = 69; Joga = Sim) = 0,6044 > y = pertence
Conjunto 2 (x = 75; Joga = Sim) = 0,9974 > y = pertence
Conjunto 2 (x = 75; Joga = Sim) = 0,9974 > y = pertence
Conjunto 2 (x = 72; Joga = Sim) = 0,8971 > y = pertence
Conjunto 2 (x = 81; Joga = Sim) = 0,5181 > y = pertence
Conjunto 2 (x = 85; Joga = Não) = 0,1762 > y = não pertence
Conjunto 2 (x = 80; Joga = Não) = 0,6262 > y = pertence
Conjunto 2 (x = 65; Joga = Não) = 0,2277 > y = não pertence
Conjunto 2 (x = 72; Joga = Não) = 0,8971 > y = pertence
Conjunto 2 (x = 71; Joga = Não) = 0,8121 > y = pertence

Totais por classe:
Conjunto 2 (Joga = Sim) = 8
Conjunto 2 (Joga = Não) = 3

Quadro 29 – Cálculo dos elementos que pertencem aos conjuntos da classe

d) obter o ganho de informação do atributo (Quadro 30);

D(Conjunto 1; Jogar = Sim) = 6/8 * log2(6/8) = -0,3112
D(Conjunto 1; Jogar = Não) = 2/8 * log2(2/8) = -0,5000
D(Conjunto 2; Jogar = Sim) = 8/11 * log2(8/11) = -0,3341
D(Conjunto 2; Jogar = Não) = 3/11 * log2(3/11) = -0,5112
Ganho (Joga, Temperatura) = Entropia (Joga) – |{[8/14 * (-0,3112 + -0,5000)] + [11/14 * (-0,3341 + -0,5112)] }|
Ganho (Joga, Temperatura) = -0,1875

Quadro 30 – Ganho de informação com lógica difusa

Este processo deve ser executado para todos os atributos contínuos existentes no

conjunto de dados analisado, para que, ou fim do processo de indução de árvore de decisão se obtenha uma árvore de decisão difusa.

3.3.4 Árvore de decisão difusa x J4.8 do Weka

Nesta seção é apresentada uma breve comparação entre o algoritmo de indução de árvore de decisão difusa desenvolvido neste trabalho (Figura 24) com o algoritmo de indução de árvore de decisão J4.8 (Figura 23) disponível no software Weka 3 (MACHINE LEARNING GROUP AT UNIVERSITY OF WAIKATO, 2010). Para realizar a comparação foi utilizado o conjunto de dados *Iris data set* (MACHINE LEARNING REPOSITORY, 2007) que possui dados referentes a características de diferentes tipos de flores.

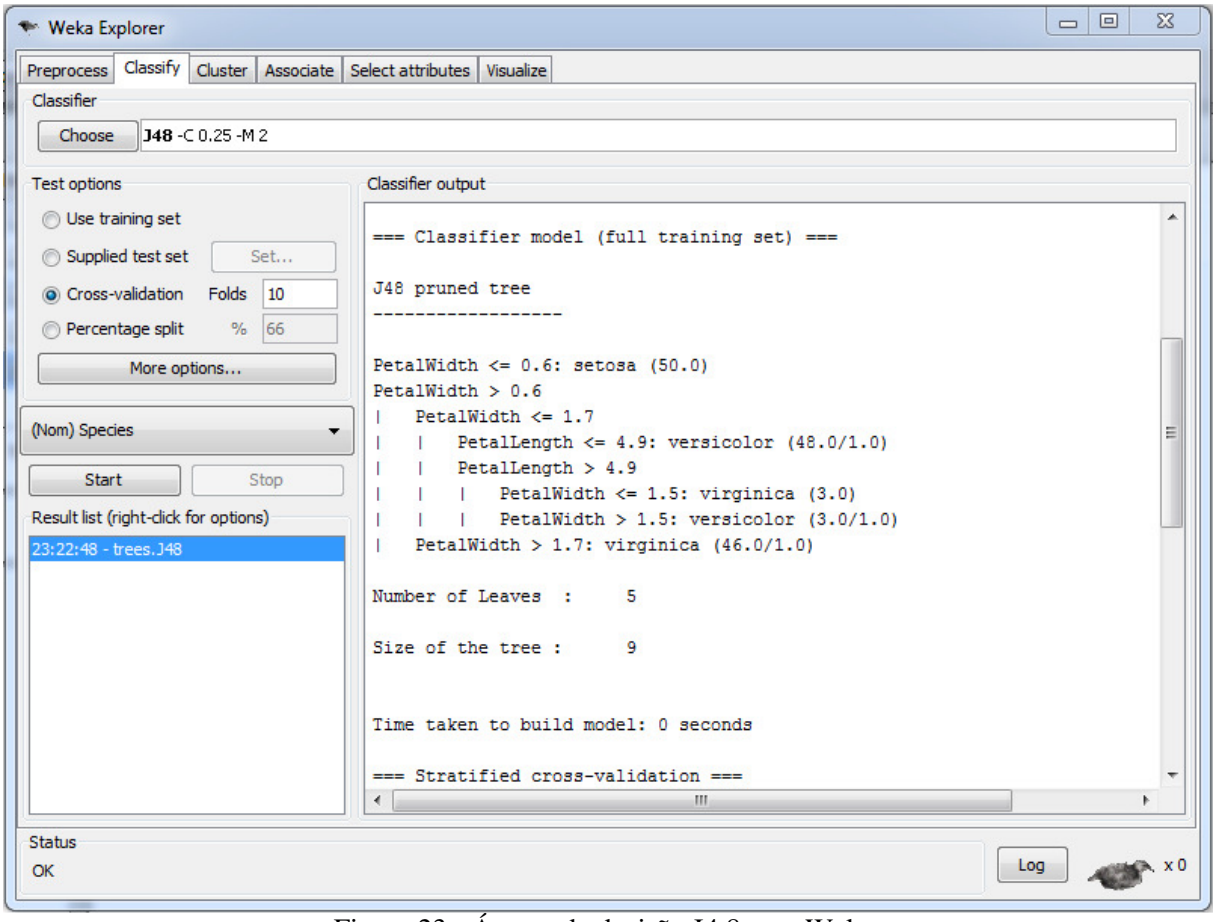


Figura 23 - Árvore de decisão J4.8 com Weka

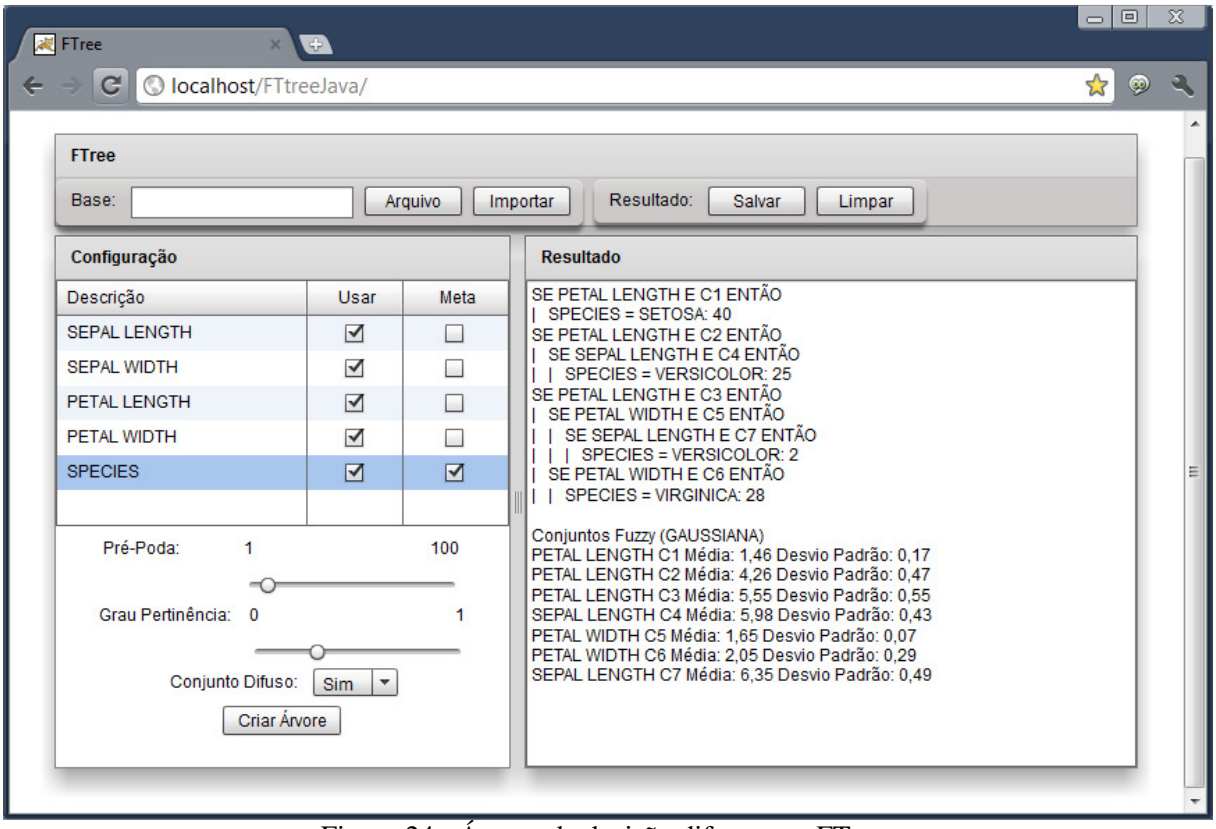


Figura 24 - Árvore de decisão difusa com FTree

Pode-se observar que a árvore gerada pelo J4.8 (Figura 23) possui uma classificação abrupta. Observa-se que o valor não é flexível em comparação com o conjunto C1 produzido pelo algoritmo de indução de árvore de decisão difusa (Figura 24) que tem como base a função de pertinência gaussiana para formar a suavização e proporcionar a generalização.

Ao analisar a Figura 25, pode-se verificar que existem elementos (em destaque) que pertencem a conjuntos difusos distintos, neste caso ao conjunto C2 e C3. Ou seja, serão classificados em mais de um conjunto com graus de pertinência distintos.

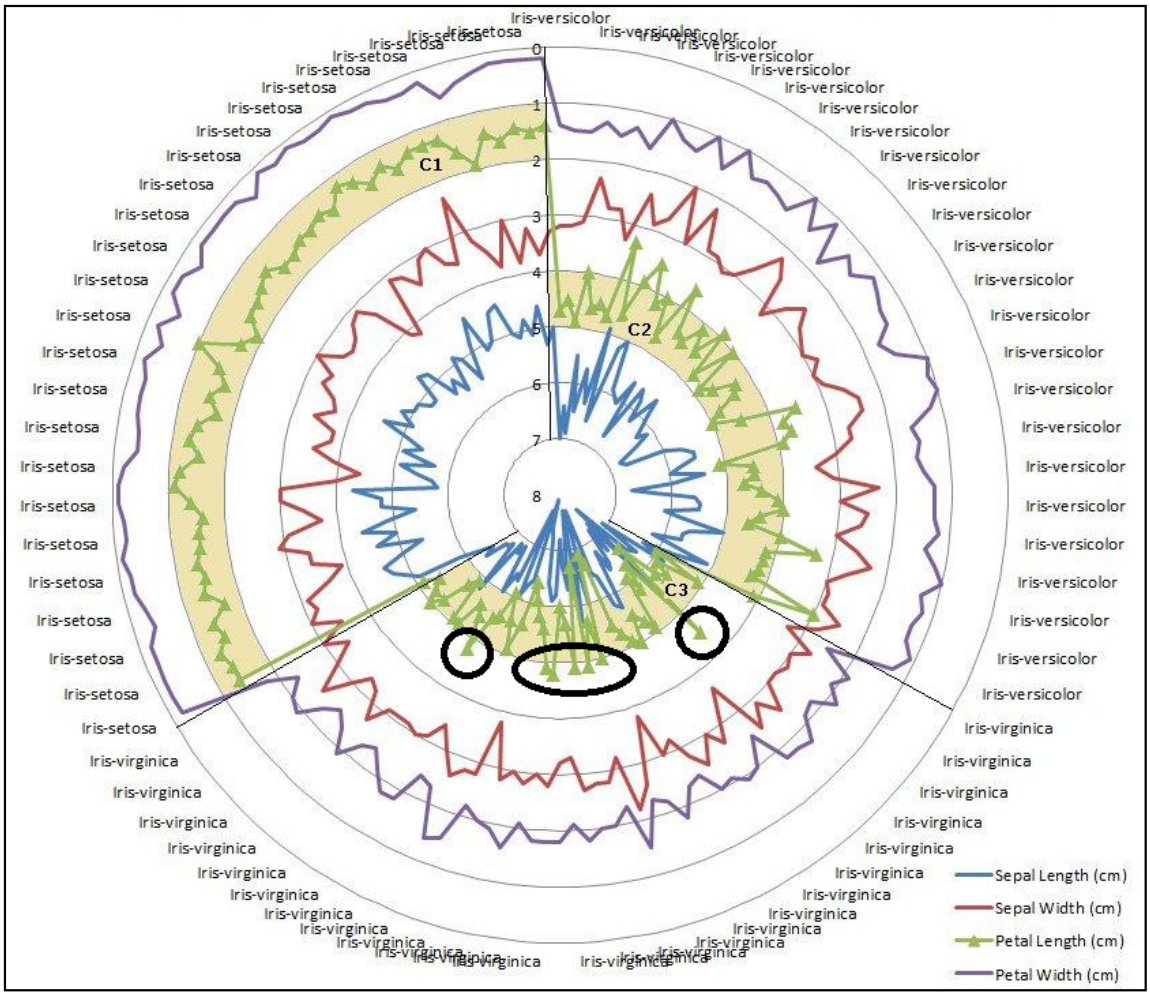


Figura 25 - Gráfico de distribuição dos valores do *Iris data set*

3.4 RESULTADOS E DISCUSSÃO

O desenvolvimento deste trabalho permitiu apresentar uma variação do algoritmo C4.5 utilizando o conceito de lógica difusa para obter vantagens no tratamento de atributos contínuos. Também permitiu apresentar de modo simplificado o processo de descoberta de conhecimento em bases de dados através da utilização da ferramenta "FTree" que possibilita de forma simples e prática criar modelos de classificação utilizando a técnica de árvores de decisão difusa.

Outras características relevantes como:

a) fácil alteração da generalização da árvore através da configuração do grau de pertinência;

b) utilização de banco de dados para auxiliar a aplicação do algoritmo de indução;

c) a simplicidade de criação de conjunto de dados com CSV torna prática à utilização da ferramenta.

No entanto, em relação ao trabalho de Compolt (1999), em que foi desenvolvida uma ferramenta para classificação e segmentação de dados utilizando a técnica de árvore de decisão, podem ser observadas algumas melhoras em relação à ferramenta desenvolvida na época, como por exemplo:

a) livre de domínio de aplicação;
b) não há um limite de atributos para análise;
c) não existe a necessidade de definir prioridades em relação aos atributos, para executar os cálculos do algoritmo.

Já em relação ao trabalho desenvolvido por Ratke, Justino e Borges (2003), que demonstra a utilização da técnica de árvore de decisão combinada com lógica difusa, foi possível observar o funcionamento do algoritmo de indução de árvore de decisão e sua adaptação para utilizar lógica difusa através do uso da função de pertinência sigmóide.

E com o trabalho de Araújo (2006) que apresenta o desenvolvimento de um algoritmo de indução de árvore de decisão difusa para ser aplicado sobre um conjunto de dados extraído de imagens foi possível observar as diferentes áreas de aplicação da técnica de mineração de dados.

Com a aplicação desenvolvida e projetada para ser utilizada com os mais diversos conjuntos de dados e também por ser possível disponibilizar a sua utilização através da *internet* e *intranet*, mostra que a proposta do trabalho pode ser integrada à estrutura de qualquer organização sem alterações significativas. Com isto as mais diversas áreas da organização teriam acesso à aplicação de modo a ser utilizada como fonte de conhecimento independente do processo gerencial da área.

4 CONCLUSÕES

O acúmulo de grande quantidade de informações nas bases de dados das organizações e o surgimento de técnicas de descoberta de conhecimento em bases de dados torna possível a transformação de informações em conhecimento que pode ser útil nos processos gerenciais das organizações. Entretanto, a utilização das ferramentas atuais, que disponibilizam diversos recursos de mineração de dados, necessita de especialistas para execução deste processo.

Neste trabalho se propôs o desenvolvimento de uma aplicação denominada “FTree”, capaz de apoiar a classificação de dados a partir de um conjunto de dados. A principal característica da aplicação é a realização de classificação de dados utilizando a técnica de árvore de decisão difusa.

Este trabalho apresentou os passos que devem ser realizados e as configurações necessárias para se realizar a criação e visualização do modelo de classificação utilizando-se da técnica de árvore de decisão difusa.

Os principais ganhos com o uso desta aplicação é criação de modelos de classificação de fácil entendimento por parte do usuário, e que por ser uma ferramenta simples pode ser utilizada por usuários que não são especialistas da área de mineração de dados.

4.1 EXTENSÕES

Embora a aplicação desenvolvida neste trabalho ofereça a funcionalidade de indução de árvore de decisão difusa, outras funcionalidades e técnicas de poda poderiam ser incluídas ou aprimoradas.

Sugere-se que sejam desenvolvidos trabalhos futuros:

a) desenvolvimento de funcionalidade para realizar o teste de classificação a partir de um exemplo de dados;
b) especializar as regras de pré-poda já existentes no algoritmo de indução de árvore de decisão difusa;
c) especializar o algoritmo de indução de árvore de decisão difusa para também utilizar regras de pós-poda;
d) especializar o algoritmo de indução de árvore de decisão difusa para utilizar outras

funções de pertinência, como sigmóide e triangular;

e) criar funcionalidade de abertura da aplicação para facilitar a integração com bases de dados.

f) outra sugestão proposta é utilizar o padrão *Predictive Model Markup Language* (PMML) para representar as regras de classificação, a fim de possibilitar a utilização da árvore de decisão difusa encontrada, por outras aplicações.

REFERÊNCIAS BIBLIOGRÁFICAS

ARAÚJO, Anderson V. **Árvore de decisão fuzzy na mineração de imagens do sistema FootScanAge.** 2006. 99 f. Dissertação (Mestrado em Informática) - Programa de Pós-Graduação em Informática, Universidade Federal do Paraná, Curitiba. Disponível em: <http://en.scientificcommons.org/13047004>. Acesso em: 18 out. 2010.

BARBIERI, Carlos. **BI – Business Intelligence:** modelagem & tecnologia. Rio de Janeiro: Axcel Books, 2001.

BERRY, Michael J. A.; LINOFF, G. **Data mining techniques for marketing, sales, and customer support**. 2. ed. Indianapolis, Indiana: Wiley Publishing, Inc., 2004.

BREIMAN, Leo. et al. **Classification and regression trees**. Monterey, CA: Wadsworth and Brooks, 1984. 368p.

CARVALHO, Deborah. R. **Árvore de decisão:** algoritmo genético para tratar o problema de pequenos disjuntos em classificação de dados. 2005. 162 f. Tese (Doutorado em Ciências em Engenharia Civil) - Universidade Federal do Rio de Janeiro, Rio de Janeiro. Disponível em: <http://www.ipardes.gov.br/biblioteca/docs/tese_deborah_carvalho.pdf >. Acesso em: 15 abr. 2011.

CARVALHO, Deborah. R. **Data mining através de introdução de regras e algoritmos genéricos**. 2003. 126 f. Dissertação (Mestrado em Informática Aplicada) - Pontifícia Universidade Católica do Paraná, Curitiba. Disponível em: <http://www.ppgia.pucpr.br/teses.html>. Acesso em: 30 set. 2010.

CARVALHO, Luis A. V. **Datamining**: a mineração de dados no marketing, medicina, economia, engenharia e administração. Rio de Janeiro: Ciência Moderna, 2005.

COELHO, Éden O. P. **Descoberta de conhecimento sobre o processo seletivo da UFLA.** 2007. 46 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) - Departamento de Ciência da Computação, Universidade Federal de Lavras, Lavras. Disponível em: <http://www.bcc.ufla.br/monografias/2007/Descoberta_de_conhecimento_sobre_o_processo_seletivo_da_UFLA.pdf >. Acesso em: 31 ago. 2010.

COMPOLT, Geandro L. **Sistema de informação executiva baseado em um data mining utilizando a técnica de árvore de decisão.** 1999. 52 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

DIAS, Maria M. **Um modelo de formalização do processo de desenvolvimento de sistemas de descoberta de conhecimento em banco de dados**. 2001. 197 f. Tese (Doutorado em Engenharia de Produção) - Programa de Pós-Graduação em Engenharia de Produção, Universidade Federal de Santa Catarina, Florianópolis. Disponível em: <http://www.din.uem.br/~mmdias/documentos/tese_Madalena.pdf >. Acesso em: 19 out. 2010.

FAYYAD, Usama M.; PIATETSKY-SHAPIRO, Grogory; SMYTH, Padhraic. From data mining to knowledge discovery: an overview. In: FAYYAD, Usama M. et al. **Advances in knowledge discovery and data mining**. Menlo Park: American Association for Artificial Intelligence, 1996. p. 1-34.

HAN, Jiawei; KAMBER, Micheline. **Data mining:** concepts and techniques. San Francisco, CA: Morgan Kaufmann, 2006. 743p.

HAND, David. J. **Construction and assessment of classification rules.** Chichester, UK: Wiley, 1997. 232p.

KOHAGURA, Tiago. **Lógica fuzzy e suas aplicações.** 2007. 49 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Departamento de Computação, Universidade Estadual de Londrina, Londrina.

MACHINE LEARNING GROUP AT UNIVERSITY OF WAIKATO. **Weka 3:** data mining software in java. [S.l.], 2010. Disponível em: <http://www.cs.waikato.ac.nz/ml/weka/index.html>. Acesso em: 30 ago. 2010.

MACHINE LEARNING REPOSITORY. **Iris data set**. [S.I.], 2007. Disponível em: <http://archive.ics.uci.edu/ml/datasets/Iris>. Acesso em: 01 jun. 2011.

MITCHELL, Tom. M. **Machine learning**. New York: McGraw Hill, 1997. 414p.

MORAIS, Cláudio M. M. Fuzzycom **-** componente de lógica fuzzy In: XV SEMINÁRIO DE INICIAÇÃO CIENTÍFICA PUC-RIO, 2007, Rio de Janeiro. **Anais...** Rio de Janeiro: Puc-Rio, 2007. 15p.

PIMENTA, Edgar M.C. Abordagens para decomposição de problemas multi-classe: os códigos de correção de erros de saída. Porto, 2004. Tese (Mestrado em Informática). Departamento de Ciência de Computadores, Faculdade de Ciências da Universidade do Porto. Disponível em: < http://repositorio-aberto.up.pt/bitstream/10216/14321/3/5306_TM_01_P.pdf>. Acesso em: 22 maio 2011.

PENG, Yonghong; FLACH, Peter A. Soft discretization to enhance the continuous decision tree induction. In: GIRAUD-CARRIER, Christophe; LAVRAC, Nada; MOYLE, Steve (Ed.). **Integrating aspects of data mining, decision support and meta-learning**. Freiburg: [s.n.], 2001. p. 109-118. Disponível em: <http://www.cs.bris.ac.uk/Publications/pub_master.jsp?id=1000563>. Acesso em: 20 out. 2010.

QUINLAN, Ross. **C4.5:** programs for machine learning. San Mateo: Morgan Kaufmann, 1993. 302p.

QUINLAN, Ross. Induction of desicion trees. **Machine learning**, Boston, v.1, n.1, p. 81-106, Mar. 1986.

RATKE, Claúdio; JUSTINO, Gilvan; BORGES, Paulo S. S. Uso da função sigmóide para pertinência em árvores de decisão difusas In: ENCONTRO DE CIÊNCIA E TECNOLOGIA, 2003, Lages. **Anais...** Lages: Uniplac, 2003. Não paginado.

REZENDE, Solange O. **Sistemas inteligentes:** fundamentos e aplicações. Barueri, SP: Manole, 2003. 525p.

RODRIGUES FILHO, José A. F.; SHIMIZU, Tamio. **Data mining**: conceitos básicos e aplicações. São Paulo: EPUSP, 2002. Não paginado.

TAH, Joseph H. M.; CARR, Virginia. A proposal for construction project risk assessment using fuzzy logic. In: HUGHES, Will (Ed.). **Construction management and economics**. [S.l.]: Routledge, 2000. p. 491-500. Disponível em: <http://www.informaworld.com/smpp/content~content=a713763603~db=all>. Acesso em: 15 set. 2010.

TAN, Pang-Ning; STEINBACH, Michael; KUMAR, Vipin. **Introdução ao datamining:** mineração de dados. Tradução Acauan P. Fernandes. Rio de Janeiro: Ciência Moderna, 2009.

WITTEN, Ian H.; FRANK, Eibe. **Data mining:** pratical machine learning tools and techniques with Java implementations. San Francisco: Morgan Kaufmann, 2000.

ZADEH, Lofti A.; JAMSHIDI, Mohammad; TITLI, André. **Applications of fuzzy logic:** towards high machine intelligence quotient systems (Prentice Hall Series on Environmental and Intelligent Manufacturing). Oklahoma: Prentice Hall, 1997.