UNIVERSIDADE FEDERAL DE CAMPINA GRANDE

CENTRO DE CIÊNCIAS E TECNOLOGIA

DEPARTAMENTO DE SISTEMAS E COMPUTAÇÃO

CURSO DE MESTRADO EM INFORMÁTICA

DISSERTAÇÃO DE MESTRADO

PROJETO E CONSTRUÇÃO DE UM AMBIENTE PARA EXTRAIR CONHECIMENTO DE BANCOS DE DADOS DA PETROBRÁS

ESTUDO DE CASO: IDENTIFICAÇÃO AUTOMÁTICA DE LITOFÁCIES EM POÇOS DE PETRÓLEO

JOSUÉ TOEBE

(MESTRANDO)

MARCUS COSTA SAMPAIO

(ORIENTADOR)

CAMPINA GRANDE
DEZEMBRO – 2002

UNIVERSIDADE FEDERAL DE CAMPINA GRANDE

CENTRO DE CIÊNCIAS E TECNOLOGIA

DEPARTAMENTO DE SISTEMA E COMPUTAÇÃO

CURSO DE MESTRADO EM INFORMÁTICA

PROJETO E CONSTRUÇÃO DE UM AMBIENTE PARA EXTRAIR CONHECIMENTO DE BANCOS DE DADOS DA PETROBRÁS

ESTUDO DE CASO: IDENTIFICAÇÃO AUTOMÁTICA DE LITOFÁCIES EM POÇOS DE PETRÓLEO

JOSUÉ TOEBE

Dissertação submetida à Coordenação de Pós-Graduação em Informática do Centro de Ciências e Tecnologia da Universidade Federal de Campina Grande, como requisito parcial para obtenção do grau de Mestre em Informática.

Área de Concentração: Ciência da Computação

Linha de Pesquisa: Sistemas de Informações e Banco de Dados

MARCUS COSTA SAMPAIO

(ORIENTADOR)

CAMPINA GRANDE
Dezembro – 2002

TOEBE, Josué

T641P

Projeto e Construção de um Ambiente para Extrair Conhecimento de Bancos de Dados da Petrobrás

Dissertação (Mestrado), Universidade Federal de Campina Grande, Centro de Ciências e Tecnologia, Coordenação de Pós Graduação em Informática,

Campina Grande, Dezembro de 2002.

132 p.

Orientador: Marcus Costa Sampaio

Palavras-chaves:

1. Banco de Dados
2. Mineração de Dados
3. Processo Automatizado
4. Heurísticas
5. Poços de Petróleo
6. Framework

CDU – 681.3.07B

"PROJETO E CONSTRUÇÃO DE UM AMBIENTE PARA EXTRAIR CONHECIMENTO DE BANCOS DE DADOS DA PETROBRÁS. ESTUDO DE CASO: IDENTIFICAÇÃO AUTOMÁTICA DE LITOFÁCIES EM POÇOS DE PETRÓLEO"

JOSUÉ TOEBE

DISSERTAÇÃO APROVADA EM 17.12.2002

PROF. MARCUS COSTA SAMPAIO, Dr.
Orientador

PROF. JACQUES PIERRE LOUIS ROBIN, Ph.D
Examinador

OLINTO GOMES SOUZA JÚNIOR, Ph.D
Examinador

CAMPINA GRANDE – PB

Dedico esta dissertação aos meus pais, João Carlos e Rosa Maria, pelo seu amor e dedicação, ao meu irmão Elton e minhas irmãs Flavia e Caroline. O incentivo de vocês foi de fundamental importância para a realização deste trabalho.

Agradecimentos

Ao apoio financeiro recebido da Agência Nacional do Petróleo (ANP) e da Financiadora de Estudos e Projetos (FINEP) por meio do Programa de Recursos Humanos da ANP para o Setor Petróleo e Gás – PRH-ANP/MME/MCT.

Ao professor Marcus Costa Sampaio, pela excelente orientação dedicada à realização deste trabalho.

Aos meus colegas e amigos, em especial ao Eder, Fabio e Paulo, pela inestimável ajuda.

Aos professores e funcionários do Departamento de Sistemas e Computação da Universidade Federal de Campina Grande.

Resumo

Um processo de mineração de dados é uma série de passos que objetivam obter conhecimento dos dados. Esses passos incluem a preparação dos dados, seleção de amostras, indução de conhecimento das amostras usando algoritmos de indução e a estimativa da qualidade do conhecimento induzido. A complexidade do processo depende destas várias variáveis relatadas. Nesta dissertação, nós propomos um processo de mineração e dados automatizado para guiar o minerador em sua tarefa. O processo automatizado considera a existência de uma diversidade de técnicas para cada etapa do processo. Com o objetivo de reduzir os custos, o processo é guiado por heurísticas que ajudam a descobrir a combinação de técnicas que obtém os melhores resultados para um dado banco de dados. O processo é concebido como um framework para mineração de dados, provendo uma infraestrutura bem projetada para quando novas técnicas forem criadas (um novo algoritmo de indução, uma nova técnica de preparação de dados, ou uma nova técnica de amostragem), elas possam ser incorporadas ao framework com um mínimo impacto. Para ilustrar sua aplicação, experimentos que seguem o processo são descritos.

Abstract

A data mining process is a series of steps to obtain knowledge from data. These steps include data preparation, random data sampling, knowledge induction from samples using induction algorithms, and estimation of the exactness of the knowledge induced. The complexity of the process is due to these various related variables. In this work, we propose a data mining process to guide the miner through the mining task. It takes into account the diversity of sample selection and preparation techniques, as well as the existence of various induction algorithms. In order to reduce its costs, the process is driven by heuristics that help to discover the combination of techniques that fits best for a given dataset for mining. The process is conceived as an object-oriented process-driven framework, providing a well-designed and well-thought-out infrastructure so that when new pieces are created (a new inducer, or a new preparation technique, or a new sampling technique), they can be incorporated with minimal impact on the other pieces in the framework. To illustrate its application, experiments that follow the process are described.

Índice

Índice de Figuras

Índice de Tabelas

Capítulo I

Introdução

Este trabalho de dissertação de mestrado foi desenvolvido no contexto do Programa Interdepartamental de Tecnologia em Petróleo e Gás - PRH (25). Esse programa tem como objetivo a formação de profissionais especializados em petróleo e gás e faz parte do Programa de Recursos Humanos da Agência Nacional do Petróleo (ANP) para o setor Petróleo e Gás, (PRH-ANP/ MCT).

O PRH-ANP/MCT, voltado para o nível superior, é uma parceria entre o Ministério da Ciência e Tecnologia e universidades brasileiras cujo objetivo é viabilizar e incentivar a criação de programas específicos para formação de especialistas na área de petróleo e gás natural, nos níveis de graduação, mestrado e doutorado. Para tanto, o PRH-ANP/MCT concede apoio financeiro (taxa de bancada) às instituições de ensino e bolsas de estudos aos alunos previamente selecionados.

A Universidade Federal da Paraíba, através do PRH (25), recebeu nos últimos dois anos cerca de 56 bolsas distribuídas pelos mais variados cursos, dentre os quais destacamos o curso de Mestrado em Informática, no qual este trabalho foi desenvolvido. De acordo com cada curso existem diferentes especializações. Este projeto se encontra associado à especialização intitulada Engenharia do Conhecimento, cujo principal objetivo é o de formar profissionais visando suprir deficiências existentes nesse setor, como por exemplo, o desenvolvimento de ferramentas avançadas de informática para inferir conhecimento "escondido" nos mais diversos bancos de dados.

Para a realização deste trabalho, temos a disposição o CD-ROM intitulado Campo Escola de Namorado fornecido pela Agência Nacional do Petróleo. Deste CD-ROM retiramos dados para a realização de um estudo de caso. O estudo de caso com

dados reais visa gerar conhecimento para a identificação automática de litofácies em poços de petróleo utilizando técnicas avançadas de Mineração de Dados.

1.1 Mineração de Dados

O grande desenvolvimento da área de banco de dados propiciou o armazenamento de enormes quantidades de dados operacionais em empresas. Bancos de dados podem ser caracterizados como verdadeiras minas de conhecimento, apenas parcialmente exploradas pelas consultas rotineiras dos usuários [Agrawal et al., 1993]. *Mineração de Dados* (MD) é uma área de pesquisa/desenvolvimento que se preocupa em como melhor explorar essas minas. Seu objetivo é descobrir *padrões* (ou *conhecimento*) nos dados. O conhecimento minerado deve ser não trivial, compreensível e de fácil assimilação [Fayyad et al., 1996].

Mineração de Dados agrega conceitos de diversas áreas de pesquisa como Estatística, Inteligência Artificial e Banco de Dados. Sua evolução deu-se a partir da década de 80, com a conscientização da existência de conhecimento não explorado e o surgimento dos primeiros algoritmos de indução[1] de conhecimento. Entre os algoritmos pioneiros estão *ID3* da Inteligência Artificial e *Naïve Bayes* da Estatística. Com o passar dos anos, diversos algoritmos foram propostos e aperfeiçoados visando o aumento na qualidade e diversidade do conhecimento minerado.

Dependendo do conhecimento inferido, ele pode ser representado de diferentes formas: regras de classificação e regras de associação, entre outras. Uma *regra de classificação* é do formato *IF antecedente THEN conseqüente*, em que o conseqüente é composto de apenas um par *atributo* = *valor*. Um classificador é um conjunto de regras de classificação em que o atributo do conseqüente é sempre o mesmo. Em se tratando de *regras de associação*, o conseqüente pode apresentar mais que um par *atributo* = *valor* e os atributos podem variar. Tratamos nesta dissertação exclusivamente da inferência de conhecimento na forma de classificadores.

Um *processo* de MD para inferência de classificadores é muito mais complexo do que a simples utilização de um algoritmo de indução sobre um banco de dados. Um

[1] Neste contexto, as palavras inferir e induzir são considerados sinônimos e significam: “deduzir; tirar por conclusão”.

processo de MD é composto por uma série de passos igualmente importantes que objetivam inferir conhecimento 'escondido' nos dados e onde os algoritmos de indução figuram apenas com uma das etapas. Esses passos incluem:

?? *Preparação dos dados* [Pyle, 1999] [Fayyad et al., 1997] ? normalmente bancos de dados apresentam problemas como dados *desconhecidos* ou *inexistentes* e dados '*sujos*' que são transmitidos para a amostra e precisam ser corrigidos antes da extração do conhecimento. Dados desconhecidos são aqueles que de fato existem, mas que por algum motivo não estão presentes no banco de dados. Por exemplo, em um cadastro de clientes, o campo *data_de_nascimento*, para vários clientes, poderia não ter sido preenchido. Já dados inexistentes são aqueles que não estão presentes no banco de dados e que no contexto realmente não existem. Por exemplo, no caso de uma tabela de clientes de um hospital, o campo com informações sobre o número de partos realizados em um cliente pode não estar preenchido, no cadastro de um cliente do sexo masculino. A aplicação de uma técnica para correção de dados desconhecidos, neste caso pode trazer inconsistência ao banco de dados, já que esta informação realmente não existe. Na maioria dos casos é muito difícil diferenciar dados desconhecidos de dados inexistentes sem a presença de um especialista no domínio dos dados. Dados 'sujos' são informações corrompidas que causam inconsistência a um banco de dados. As causas de dados 'sujos' são várias. Os exemplos mais comuns são oriundos da fase de aquisição dos dados, como erros de digitação. Também na fase de preparação da amostra pode ser necessário transformar um domínio de dados de valores contínuos para valores discretos. Para ilustrar, considere uma variável contínua de um banco de dados que armazene a idade de clientes cadastrados. Pode-se transformar essa variável em discreta utilizando faixas de valores (criança, jovem, adulto, idoso).

?? *Seleção de amostra* [Pyle, 1999] [Brumem et al., 2001] ? a importância da seleção de amostras representativas de uma 'população' de dados está no alto custo de processamento dos algoritmos de indução se todos os dados disponíveis fossem processados.

?? *Fragmentação do conjunto-amostra* em subconjuntos [Kohavi, 1995] [Witten and Eibe, 1999] [Jain and Dubes, 1997] ? Suponha a existência de uma amostra já preparada para MD. Se toda a amostra fosse submetida a um algoritmo de indução de conhecimento, não seria possível avaliar a qualidade do conhecimento inferido. Já que a avaliação do conhecimento é feita sobre o conjunto de teste, que deve ser composto por casos diferentes dos contidos no conjunto de treinamento. Por esse motivo, a amostra é dividida em sub-conjuntos de treinamento e teste. O conjunto de treinamento é utilizado pelo algoritmo de indução para inferir o conhecimento e o conjunto de teste para avaliação do conhecimento inferido.

?? *Algoritmo de Indução do Conhecimento* [Bramer, 2000] [Thrun et al., 1991] [Quinlan, 1993] ? nesta etapa é utilizada um dos diferentes algoritmos de indução para inferir o conhecimento 'escondido' no conjunto de treinamento. O conhecimento inferido é apresentado sob um modelo de conhecimento. Um modelo de conhecimento é um padrão formal para representação do conhecimento e depende diretamente do algoritmo de indução utilizado. Existem vários modelos para representação de conhecimento, e para cada modelo existem algoritmos de indução. O algoritmo *Prism* [Cendrowska, 1987], por exemplo, induz o conhecimento na forma de regras de classificação; a família de algoritmos TDIDT [Quinlan, 1993] (*Top-Down Induction of Decision Trees*) induz o conhecimento na forma de árvores de decisão.

?? *Avaliação do conhecimento* inferido [Witten and Eibe, 1999] ? o conhecimento inferido é avaliado com o conjunto de teste e uma medida de sua qualidade é estimada. A qualidade do conhecimento pode ser expressa por sua *acurácia*, que indica o quanto o conhecimento é verdadeiro para o conjunto de testes e dá uma estimativa do quanto aquele conhecimento é verdadeiro para o conjunto de execução. O conjunto de execução é formado por casos 'novos', isto é, que não estão presentes nos conjuntos de treinamento e testes. Considere a existência de um banco de dados com informações meteorológicas. Uma amostra é retirada e preparada para MD.

A amostra é dividida em subconjuntos de treinamento e teste. O conhecimento é inferido do conjunto de treinamento e sua acurácia é medida com o conjunto de teste. Então esse conhecimento é utilizado para prever futuras condições do tempo, que ainda não estão presentes no banco de dados ? o conjunto de execução.

Como o leitor pode deduzir, a confiança na qualidade do conhecimento extraído depende fortemente do processo de MD subjacente. Infelizmente, a literatura, pródiga em algoritmos de indução é, no entanto, pobre quanto à discussão do processo de MD como um todo.

Para cada um dos passos descritos anteriormente, diversas técnicas foram propostas. Desta forma, temos diversas técnicas de seleção de amostras, diversas técnicas de fragmentação de amostras e inúmeros algoritmos de indução de conhecimento. A escolha de qual técnica permitirá obter os melhores resultados em cada um dos passos, depende do banco de dados a ser minerado. A complexidade de um processo de MD deve-se a essas múltiplas variáveis em jogo.

A experiência acumulada com processos de MD "ad hoc" mostra que não existe uma técnica de amostragem que seja ótima em todos os casos. A escolha adequada de qual técnica de amostragem utilizar depende do banco de dados, da técnica de fragmentação de amostras e do algoritmo de indução.

	Banco de Dados	Técnica de amostragem	Técnica de Fragmentação da Amostra	Algoritmo de Indução	Conhecimento Induzido.
1º	C_1	Am_1	F_1	A_1	X_1
2º	C_1	Am_2	F_1	A_1	X_2
3º	C_1	Am_1	F_2	A_1	X_3
4º	C_1	Am_1	F_1	A_2	X_4

Tabela 1.1: Casos de DM

A tabela 1.1 apresenta cinco casos de MD. No primeiro caso, com o uso da técnica de amostragem Am_1 foi extraída uma amostra do banco de dados C_1 já

devidamente preparado para MD. Essa amostra foi fragmentada com F_1. Em seguida o algoritmo de indução A_1 foi aplicado, inferindo o conhecimento X_1. No segundo caso, tudo se passou como no primeiro, com exceção da técnica de amostragem que foi substituída por Am_2, resultando em um conhecimento X_2 ? X_1. Essa instabilidade no conhecimento produzida pela mudança na técnica de amostragem conduz a duas questões: Qual dos dois conhecimentos é o melhor, X_1 ou X_2? Qual a técnica de amostragem que nos leva ao melhor conhecimento para o banco de dados C_1, Am_1 ou Am_2?

Da mesma forma, no terceiro caso todas as técnicas utilizadas para minerar o banco de dados C_1 são idênticas as do primeiro caso, com exceção da técnica de fragmentação que foi substituída por F_2. Como resultado foi induzido o conhecimento X_3 ? X_1. Também com alteração da técnica de fragmentação existe instabilidade no conhecimento induzido. O que leva a outras questões: Qual dos conhecimentos é o melhor, X_1 ou X_3? Qual técnica de fragmentação nos leva ao melhor conhecimento para o banco de dados C_1, F_1 ou F_2?

No quarto caso, novamente todas as técnicas utilizadas para minerar C_1 são as mesmas que no primeiro caso, com exceção do algoritmo de indução, que no primeiro caso é A_1 e foi substituído por A_2 no quarto caso. Como resultado temos o conhecimento X_4 ? X_1. A alteração no algoritmo de indução também produz instabilidade. Outras questões podem ser formuladas: Qual dos dois conhecimentos é o melhor, X_1 ou X_4? Qual o algoritmo de indução induz o melhor conhecimento do banco de dados C_1, A_1 ou A_2?

Sobretudo, se o banco de dados a ser minerado C_1 for substituído por C_2, as respostas para as diversas questões anteriores serão, provavelmente, diferentes.

Esta instabilidade no conhecimento induzido torna complexo o trabalho do minerador[2], que objetiva extrair um bom resultado, mas tem dificuldade em fazer as escolhas acertadas, já que um bom resultado depende de uma combinação acertada de técnica de amostragem, técnica de fragmentação e algoritmo de indução. Atualmente inexiste na literatura uma proposta de processo automatizado de MD que auxilie o minerador nesta tarefa.

[2] Agente do processo de Mineração de Dados.

1.2 Objetivos da Dissertação

Este trabalho tem como objetivo principal propor um processo automatizado de mineração de dados para a inferência de classificadores. O processo agrega diversas técnicas para cada uma das etapas do processo de mineração vistas na seção 1.2 selecionando um conjunto de técnicas que infere o conhecimento com a melhor acurácia para um banco de dados sendo minerado. Com a ajuda do processo, a complexidade do trabalho do minerador fica enormemente reduzida.

Além disto, a adição de novas técnicas ao processo é facilitada pela implementação realizada na forma de um framework[3]. Frameworks são hoje o estado-da-arte em engenharia de software no que tange o desenvolvimento de software reutilizável e facilmente extensível. O desenvolvimento na forma de um framework traz inúmeros benefícios, já que ele permite um amplo reuso de projeto e código. Permitindo desta forma a rápida e fácil adição de novas técnicas para qualquer que seja a etapa do processo de MD.

1.3 Estrutura da Dissertação

No capítulo 2 apresentamos alguns conceitos e técnicas relacionadas à MD que são úteis para a compreensão das seções seguintes. Dentre os conceitos que definimos estão: o que é um banco de dados classificado, os conjuntos de treinamento, teste e execução, algoritmos de indução, classificador, acurácia e amostra. Apresentaremos também algumas técnicas de seleção de amostras, fragmentação de amostras e indução de conhecimento que são utilizadas no processo de MD proposto.

No capítulo 3 o processo de Mineração de Dados para inferência de classificadores é formalmente proposto em duas versões. A primeira versão denominada *Naïve_Classifier_Inducer* faz pesquisa exaustiva dentre as possíveis combinações de técnicas visando selecionar as melhores, para o banco de dados sendo minerado.

[3] Um framework é um conjunto de classes (no sentido da orientação a objetos) coesas que colaboram entre si para compor um projeto reutilizável para um escopo específico de aplicações [Gamma et al., 1994].

Todavia esta primeira versão apresenta custo computacional muito elevado. A segunda versão denominada *Expert_Classifier_Inducer* faz uso de heurísticas para diminuir o custo de processamento.

No capítulo 4 mostraremos uma implementação para os algoritmos. Nesse capítulo são apresentados detalhes de análise, projeto e implementação na forma de um framework orientado a objeto.

No capítulo 5, uma avaliação experimental é realizada com o uso de dados provenientes de diversas áreas, disponíveis no repositório UCI [Blake and Merz, 2002]. A avaliação visa mostrar a eficácia dos algoritmos propostos na extração de conhecimento a partir de bancos de dados.

Um estudo de caso é apresentado no capítulo 6. Para isto utilizamos o banco de dados disponível no CD-ROM Campo Escola de Namorado. O estudo visa inferir conhecimento para a identificação automática de litofácies em poços de petróleo.

As conclusões e algumas perspectivas de trabalhos futuros são apresentadas no capítulo 7.

Capítulo II

Conceitos e Técnicas Relacionados a Mineração de Dados

Neste capítulo, são formalizados conceitos e apresentadas técnicas de mineração de dados que são úteis para a compreensão do processo automatizado proposto. O processo de mineração de dados será detalhado a partir do capítulo 3.

2.1 Conceitos Fundamentais

2.1.1 Banco de Dados Classificado

Seja *Dom(A)* o domínio do atributo *A* de um banco de dados relacional *X*. X_c ? *X* é um *banco de dados classificado* se cada instância deste BD for formada por um conjunto de valores associados a $v+1$ atributos ? $A_1, ..., A_v, A_{v+1}$? , com $Dom(A_{v+1}) = \{ c_1, ..., c_m \}$, em que c_i é uma classe e *m* é o número de classes. A_{v+1} é então chamado de *atributo de classificação.*

Para ilustrar, considere um problema fictício de vazamento em tubulações de transporte de óleo combustível. Considera-se a possibilidade de condições climáticas influenciarem a ocorrência desses vazamentos. A tabela 2.1 apresenta as informações sobre condições do tempo e os eventuais vazamentos das tubulações. Constam do BD quatorze vistorias realizadas nas tubulações, cada uma correspondendo a uma linha da tabela. Para cada vistoria, cinco informações foram computadas, correspondendo aos atributos Estado, Temperatura, Umidade, Vento e Vazamento. O quinto atributo ? Vazamento ? é o atributo de classificação. *Dom(Vazamento)* é composto por duas classes: *{sim, não}*, que indicam ou não a ocorrência de vazamentos, nas condições climáticas apresentadas. O problema de MD neste caso é descobrir em que condições do

tempo ocorrem com mais freqüência os vazamentos, para, em casos futuros, saber com antecedência se irão ocorrer vazamentos ou não.

Estado	Temperatura	Umidade	Vento	Vazamento
ensolarado	alta	Alta	não	não
ensolarado	alta	Alta	sim	não
nublado	média	Alta	não	sim
chuvoso	baixa	Alta	não	sim
chuvoso	baixa	normal	não	sim
chuvoso	baixa	normal	sim	não
nublado	média	normal	sim	sim
ensolarado	baixa	Alta	não	não
ensolarado	baixa	normal	não	sim
chuvoso	média	normal	não	sim
ensolarado	média	normal	sim	sim
nublado	média	Alta	sim	sim
nublado	alta	normal	não	sim
chuvoso	média	Alta	sim	não

Tabela 2.1: BD com informações sobre vazamentos

A tabela 2.1 representa então um BD classificado, contendo informações sobre vazamentos.

2.1.2 Conjuntos de Treinamento, Teste e Execução

Considere novamente o banco de dados X. X_{ctr} ? X_c é um *conjunto de treinamento* e X_{cts} ? X_c é um *conjunto de teste,* X_{ctr} ? $X_{cts} = X_c$ e X_{ctr} ? X_{cts} = ? , assim como X_c ? X_e = ? e X_c ? $X_e = X$. X_e é um *conjunto de execução*, ou um conjunto de instâncias não classificadas. A figura 2.1 ilustra estes conjuntos para MD.

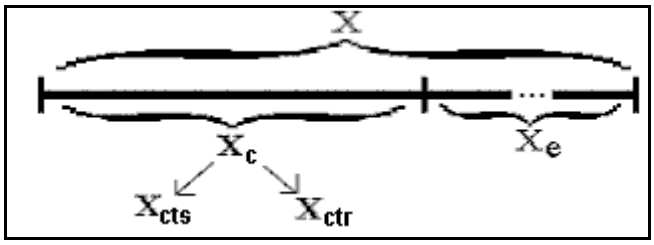

Figura 2.1: Conjuntos para MD

Retomando o nosso exemplo, temos o BD de vazamentos X_c, com X_c ? X. X_{ctr} e X_{cts}, com X_{ctr} ? $X_{cts} = X_c$ e X_{ctr} ? $X_{cts} = ?$, dividem respectivamente as quatorze instâncias de X_c, sem elementos em comum. X_e, com $X = X_c$? X_e, é o conjunto de execução, ou seja, composto por instâncias ainda não classificadas ? os casos futuros. As tabelas 2.2 e 2.3 apresentam uma possível divisão de X_c em X_{ctr} e X_{cts}, em que o conjunto de treinamento é composto por 10 instâncias selecionadas de forma aleatória, e o conjunto de teste com 4.

Estado	**Temperatura**	**Umidade**	**Vento**	**Vazamento**
ensolarado	alta	alta	não	não
ensolarado	alta	alta	sim	não
chuvoso	baixa	alta	não	sim
chuvoso	baixa	normal	não	sim
nublado	média	normal	sim	sim
ensolarado	baixa	normal	não	sim
chuvoso	média	normal	não	sim
ensolarado	média	normal	sim	sim
nublado	média	alta	sim	sim
chuvoso	média	alta	sim	não

Tabela 2.2: Conjunto de treinamento X_{ctr}

Estado	Temperatura	Umidade	Vento	Vazamento
nublado	Média	alta	não	sim
chuvoso	Baixa	normal	sim	não
ensolarado	Baixa	alta	não	não
nublado	Alta	normal	não	sim

Tabela 2.3: Conjunto de teste X_{cts}

O conjunto X, com os subconjuntos X_{ctr}, X_{cts} *e* X_e serão também utilizados para ilustrar as próximas definições.

2.1.3 Algoritmo de Indução de Conhecimento e Classificador

Um *algoritmo de indução de conhecimento* infere um classificador, segundo um modelo de conhecimento, do conjunto de treinamento X_{ctr}. Um *classificador* é uma função que mapeia uma instância não classificada de um conjunto de execução X_e para uma classe.

De um modo geral, o *modelo de conhecimento* é uma regra *if T then C*, em que *T* é uma conjunção de *termos* <atributo ? valor>, ? = {'=', '<', '>', '?', '?', '?'}, e *C* é um termo <atributo_de_classificação ? classe>.

Entre os mais importantes algoritmos de indução, citamos *Prism* [Cendrowska, 1987] [Witten and Eibe, 1999] e a família de algoritmos TDIDT [Quinlan, 1993] (*Top-Down Induction of Decision Trees*). TDIDT induz regras de classificação indiretamente, na forma de árvores de decisão.

Voltemos ao exemplo dos vazamentos. A figura 2.2 mostra um possível classificador, induzido do conjunto de treinamento da tabela 2.3 pelo algoritmo de indução *Prism*.

```
if Estado = ensolarado and Umidade = alta then
        Vazamento = não
if Estado = chuvoso and Vento = sim then
        Vazamento = não
if Estado = nublado then
        Vazamento = sim
if Umidade = normal then
        Vazamento = sim
else Vazamento = sim
```

Figura 2.2: Classificador induzido pelo algoritmo *Prism*

Já a figura 2.3 mostra um classificador na forma de árvore de decisão induzido por um algoritmo da família TDIDT do mesmo conjunto de treinamento.

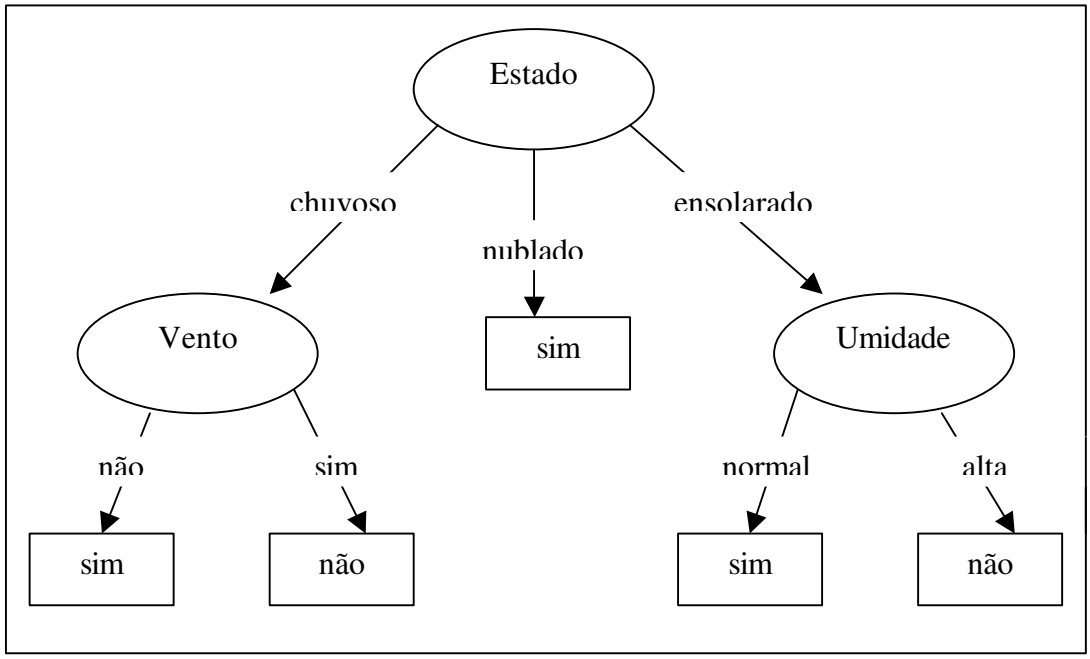

Figura 2.3: Classificador induzido por um algoritmo da família TDIDT na forma de árvore de decisão

A figura 2.4 mostra o conhecimento na forma de regras de associação, inferido pelo algoritmo *Apriori* do conjunto de treinamento na tabela 2.2.

```
if temperatura = baixa then
      umidade = normal
if umidade = normal and ventania = não then
      umidade = alta
if estado = ensolarado and vazamento = não then
      umidade = alta
if ventania = não and vazamento = não then
      estado = ensolarado and umidade = alta
```

Figura 2.4: Conhecimento induzido pelo algoritmo *Apriori*, na forma de regras de associação

A seção 2.4 deste capítulo apresenta em detalhes alguns algoritmos de indução de conhecimento utilizados na implementação do processo de MD proposto a partir do capítulo 3.

2.1.4 Acurácia com Conjunto de Teste

Antes de classificar X_e o classificador induzido a partir de X_{ctr} deve ser testado com cada instância do conjunto de teste X_{cts}. O teste com uma instância é bem sucedido quando o classificador mapeia a instância para uma classe que é a mesma classe da instância no conjunto de teste, senão o teste com a instância é considerado mal sucedido. A *acurácia com conjunto de teste* ? ou simplesmente *acurácia de teste* ? , acc_{ts}, é definida pela equação:

$$acc_{ts} ? \frac{? (testes\_bem\_sucedidos)s}{? (testes\_mal\_sucedidos)s? ? (testes\_bem\_sucedidos)s}, ? \text{ instância } ? X_{cts}.$$

Observe que a acurácia de teste varia com alterações nos conjuntos treinamento e teste, de modo que, a acurácia de teste é em geral mais adequadamente calculada com a média de diversas acc_{ts}, cada uma obtida com um diferente par de conjuntos treinamento-teste. A seção 2.3 deste capítulo discute algumas técnicas de fragmentação de amostras em conjuntos de treinamento e teste.

No exemplo dos vazamentos, temos apenas um par de conjuntos X_{ctr} - X_{cts} ? tabelas 2.2 e 2.3. O classificador apresentado na figura 2.2 foi induzido pelo algoritmo de indução de conhecimento *Prism* a partir do conjunto de treinamento disposto na tabela 2.2. O respectivo conjunto de teste é composto por quatro instâncias e o classificador consegue classificar corretamente todas elas. A acurácia de teste deste classificador é dada por:

$$acc_{ts} ? \frac{4}{0 ? 4}$$

$$acc_{ts} ? 1$$

O classificador induzido consegue 100% de acurácia de teste. Contudo, em casos reais, um classificador útil nem sempre obtém 100% de acurácia de teste. Em muitos casos, regras com 100% de acurácia podem ser descartadas. Esse descarte ocorre quando a cobertura da regra é muito pequena ? fenômeno conhecido na literatura como "overfitting" [Pyle, 1999], que ocorre quando o algoritmo de indução gera regras a partir de casos isolados, com acurácia elevada, mas com pouca representatividade no banco de dados ? . Em outros casos, regras com acurácia baixa, podem ser perfeitamente confiáveis se elas foram induzidas de uma quantidade significativa de casos no conjunto de treinamento.

2.1.5 Acurácia com Conjunto de Execução

O conjunto de execução X_e é formado por dados ainda não classificados. A *acurácia com conjunto de execução* ? ou simplesmente *acurácia de execução,* acc_e ? é a probabilidade estimada de um classificador classificar corretamente uma instância de X_e, selecionada de forma aleatória. Espera-se que a acurácia de execução seja muito

próxima da acurácia de teste. Dificilmente, entretanto, a acurácia de execução será exatamente igual a acurácia de teste, visto que o conjunto de execução provavelmente terá algumas características diferentes do conjunto de teste.

O problema que se coloca então é o de saber, com um certo grau de confiança, o quanto a acurácia de execução pode variar em relação a acurácia de teste.

A solução para este problema vem da estatística. Em estatística, uma sucessão de eventos independente é chamada de um processo de Bernoulli [Witten and Eibe, 1999]. A medida da acurácia de teste é modelada como um processo de Bernoulli, já que os testes com cada instância do conjunto de teste são independentes. Considerando que uma sucessão de testes tem uma margem de erro ou acerto, a equação que mede a variabilidade da taxa de acerto de um processo de Bernoulli ? e, portanto, dos testes com um classificador ? é apresentada a seguir.

$$acc_e \; ? \; \frac{2 ? N ? acc_{ts} \; ? \; z^2 \; ? \; \sqrt{4 ? N ? acc_{ts} \; ? \; z^2 \; ? \; 4 ? N ? acc_{ts}^2}}{2 ? (N \; ? \; z^2)}$$

Na fórmula apresentada, acc_e é calculado em função de acc_{ts}, considerando um determinado grau de confiança. Mais precisamente, N indica o número de instâncias utilizadas para o cálculo da acurácia de teste acc_{ts}, e z expressa o grau de confiança do resultado. Os valores de z, bem como os graus de confiança correspondentes são apresentados na tabela 2.4.

Confiança	**z**
20%	0,25
40%	0,84
80%	1,28
90%	1,65
98%	2,33
99%	2,58
99,8%	3,09

Tabela 2.4: Limites de confiança para equação de Bernoulli

Prosseguindo com o exemplo dos vazamentos, o classificador da figura 2.2 obteve 100% de acurácia de teste com um conjunto de teste de apenas quatro instâncias.

Objetivando descobrir a acurácia de execução, com um grau de confiança de 80%, as variáveis da equação ficam assim instanciadas:

$$acc_e \in \frac{2 \cdot 4 \cdot 1 \cdot 1{,}28^2 \pm \sqrt{4 \cdot 4 \cdot 1 \cdot 1{,}28^2 \cdot 4 \cdot 4 \cdot 1^2}}{2 \cdot (4 \cdot 1{,}28^2)}$$

$$acc_e \in \{70{,}94\% \ldots 100\%\}$$

O resultado obtido é uma acurácia de execução entre 70,94% e 100%, com uma confiança de 80%. A faixa de valores é grande, mas é explicado pelo baixo número de instâncias do conjunto de teste.

2.1.6 Amostra para MD

Uma *amostra X_c para MD* é um conjunto de casos retirado de uma 'população' *X*. Como explicado anteriormente, X_c deve ser fragmentado em conjunto de treinamento e conjunto de teste, $X_c = X_{ctr} \cup X_{cts}$. No exemplo do tempo, X_c é uma amostra com 14 casos (tabela 2.1). Certamente, não é uma amostra representativa, dado o número reduzido de casos. Contudo, a seleção de uma amostra representativa de uma 'população', não é uma tarefa fácil. Algumas das técnicas de seleção de amostras existentes são apresentadas na próxima seção.

2.2 Técnicas de Amostragem para MD

Nesta seção, são discutidas duas técnicas para seleção de amostras de bancos de dados para MD, *Adaptive Incremental Framework* e *Convergência*.

2.2.1 Adaptive Incremental Framework

O objetivo da técnica *Adaptive Incremental Framework* [Brumen et al., 2001] é selecionar, de um banco de dados, a menor amostra possível, de modo que a acurácia[4] do conhecimento inferido da amostra satisfaça os requerimentos prescritos. A idéia básica é construir uma pequena parte inicial de uma *curva de performance*, que mostra a variabilidade da acurácia de um classificador relativamente ao tamanho do conjunto de treinamento. O restante da curva é estimado por uma função matemática. Dois tipos de funções podem ser utilizados para estimar o restante da curva ? funções logarítmicas e funções de potência.

Função logarítmica: $y ? A ? \log(x) ? B$

Função de potência: $y ? A ? x^b$

Em que y representa a acurácia e x representa o tamanho do conjunto de treinamento utilizado, A é a acurácia inicial enquanto B é a taxa de incremento do conjunto de treinamento [Frey and Fisher, 1999].

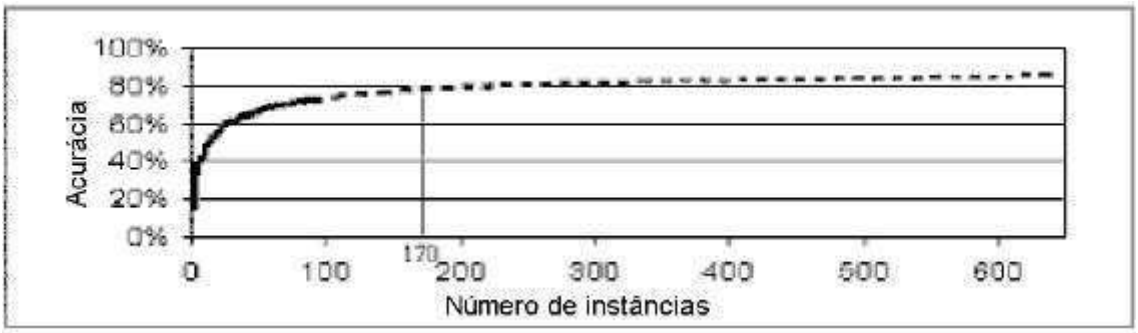

Figura 2.5: Uma curva de performance

A figura 2.5 mostra uma curva de performance em que os 15% iniciais (linha contínua) foram construídos com a ajuda de um algoritmo de indução, e os 85% restantes (linha pontilhada) foi estimada por uma função de potência.

Analisando a figura, percebe-se uma grande instabilidade da curva de performance para pequenos conjuntos de treinamento ? menores que 100 instâncias. A partir de 170 instâncias, a curva torna-se praticamente estável em torno 80% de

[4] Acurácia de teste.

acurácia. Como resultado, o minerador pode escolher um conjunto de treinamento com 170 instâncias e obterá uma acurácia em torno de 80%, resultado praticamente idêntico ao de um conjunto com 500 instâncias, por exemplo. O conjunto de teste é uma fração do conjunto de treinamento (veja seção 2.3).

A grande vantagem desta técnica é que a maior parte da curva de performance é estimada por uma função. A confiança da estimativa depende do algoritmo de indução utilizado e necessita ser validada com um conjunto de teste. Além disto, esta técnica é influenciada por particularidades do banco de dados, com a possibilidade de trabalhar bem para alguns bancos de dados e não trabalhar muito bem para outros bancos de dados.

2.2.2 Convergência

A técnica da *Convergência* [Pyle, 1999] também permite a seleção de amostras de um banco de dados para MD. A idéia subjacente é a seguinte: dado um banco de dados e uma amostra deste banco de dados, as curvas de distribuição dos valores de cada atributo na amostra e no banco de dados devem estatisticamente convergir. A medida da aproximação entre amostra e banco de dados é chamada variabilidade. Quanto menor for a variabilidade, maior será a convergência.

Com o objetivo de alcançar um bom tamanho de amostra, uma pequena amostra inicial é progressivamente[5?] adicionada de outras instâncias, e concomitantemente a variabilidade é calculada. Quando a variabilidade é suficientemente baixa, então a distribuição de valores dos atributos da amostra convergiu para a distribuição da 'população' e a amostra é selecionada.

Infelizmente, as amostras escolhidas através da técnica de convergência não são necessariamente boas para MD. O resultado depende de características particulares do banco de dados e também algoritmo de indução de conhecimento, da seguinte forma: a amostra pode ser boa para um algoritmo de indução, mas pode não ser tão boa para outro algoritmo.

Na seção seguinte, discutimos técnicas de fragmentação de amostras em conjunto de treinamento e conjunto de teste.

[5] Consiste em outra técnica incremental
?

2.3 Técnicas de Fragmentação

Serão descritas aqui três técnicas de fragmentação de amostras em conjunto de treinamento e conjunto de teste: *Holdout*, *K-fold Cross Validation* e *Bootstrap*.

2.3.1 Holdout

A técnica *Holdout* [Kohavi, 1995], também conhecida na literatura como teste para estimar a amostra, divide o conjunto amostra em dois subconjuntos mutuamente excludentes: conjunto de treinamento e conjunto de teste. Para o conjunto de treinamento é designado 2/3 das instâncias, enquanto os remanescentes 1/3 são destinados ao conjunto de teste. O conjunto de treinamento é submetido a um algoritmo de indução, o classificador induzido é então testado com o conjunto de teste.

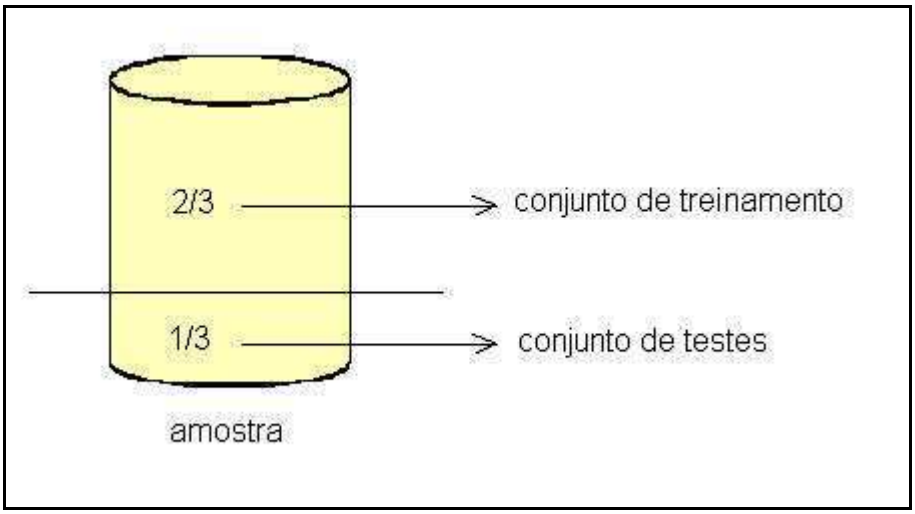

Figura 2.6: A técnica *Holdout*

Assumindo que a acurácia do classificador induzido aumenta com o aumento do conjunto de treinamento, a técnica *Holdout* é muito pessimista no cálculo da acurácia, dado que o conjunto de treinamento é apenas 66% da amostra.

2.3.2 K-fold Cross Validation

K-fold Cross Validation [Witten and Eibe, 1999] é uma técnica de fragmentação de amostras que funciona do seguinte modo: seja *D* uma amostra de tamanho *n*. *D* é aleatoriamente dividido em *k* subconjuntos D_i *1* ? *i* ? *k*, todos eles com o mesmo tamanho *n/k*. Assim *k* classificadores são induzidos com um algoritmo de indução da seguinte maneira: o i-ésimo classificador é induzido do conjunto de treinamento (*D* - D_i) e testado com D_i. A acurácia do classificador resultante (qualquer um dos *k* classificadores) é obtido calculando a média aritmética ? das acurácias dos *k* classificadores, com o desvio padrão ?. A figura 2.7 ilustra o funcionamento da técnica *K-fold Cross Validation*.

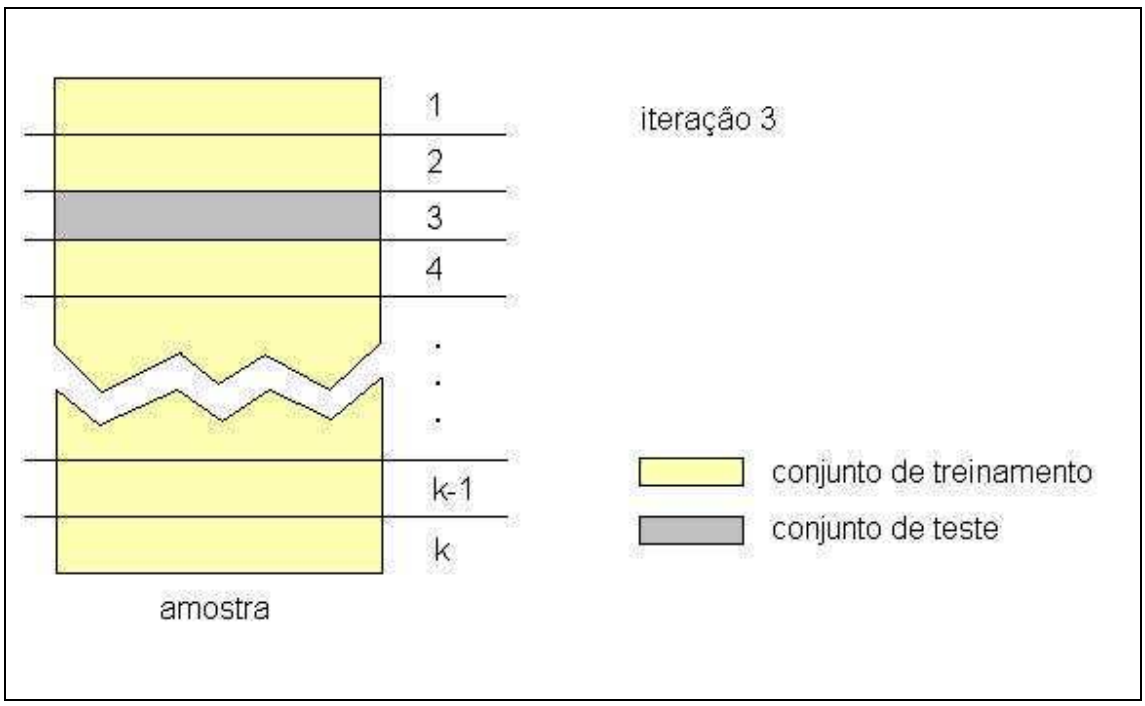

Figura 2.7: A técnica *K-fold Cross Validation*

O estudo apresentado em [Kohavi, 1995] conclui que *k=10* é uma é uma ótima escolha. Para *k<10*, a instabilidade dos classificadores é alta, já que uma grande quantidade das instâncias disponíveis é reservada para teste em cada iteração. Para *k>10*, os conjuntos de teste são excessivamente pequenos, o que pode resultar em uma

acurácia falsamente maior que a real, já que muitos casos representativos podem não estar presentes neste conjunto de teste pequeno.

Uma derivação desta técnica utiliza o conceito de estratificação. *Estratificação* é um meio de manter a mesma proporção de instâncias classificadas com cada classe, em cada um dos *k* subconjuntos [Kohavi, 1995]. Desta maneira é possível obter classificadores com acurácia média mais elevada e com desvio padrão menor. Considere um exemplo: se em uma determinada amostra para MD, a distribuição das instâncias nas classes é como a apresentada na tabela 2.5, cada um dos *k* subconjuntos deve manter a mesma proporcionalidade.

Classe	**Distribuição**
A	30%
B	20%
C	50%

Tabela 2.5: Distribuição das classes em uma amostra

A técnica (com ou sem estratificação) assume que as diferenças entre os *k* classificadores induzidos são mínimas, ou que o conhecimento induzido é estável. Infelizmente, isto é freqüentemente falso.

2.3.3 Bootstrap

Bootstrap [Jain and Dubes, 1997] é outra técnica para fragmentação da amostra que trabalha da seguinte forma: A partir de uma amostra com *n* instâncias, *n* instâncias são aleatoriamente selecionadas para formar o conjunto de treinamento, com recolocação. Conseqüentemente o conjunto de treinamento será também de tamanho *n*, mas certamente apresentará instâncias repetidas, sendo então provavelmente diferente da amostra original. As instâncias da amostra original que não tenham sido selecionadas para formar o conjunto de treinamento são destinadas ao conjunto de teste.

A probabilidade de uma instância da amostra não ser selecionada para formar o conjunto de treinamento após *n* seleções é de $(1-1/n)^n \approx e^{-1} \approx 0{,}368$. Assim, o número de diferentes instâncias que compõem o conjunto de treinamento é aproximadamente 63%

de *n*. Esta é uma porcentagem bem pequena, se comparado a *Ten-fold Cross Validation*, onde o tamanho do conjunto de treinamento é 90% de *n* (para cada 10 subconjunto de uma amostra, 9 compõe o conjunto de treinamento e 1 o conjunto de teste). Com o objetivo de compensar esta pequena porcentagem de instâncias diferentes, o processo é repetido *b* vezes[6]. Conseqüentemente *b* classificadores são induzidos e a acurácia do classificador escolhido (qualquer um dos *b* classificadores) é a média ponderada, calculada da seguinte forma:

$$Acurácia = \frac{1}{b} \sum_{i=1}^{b} (0{,}632 \cdot acc_{cts_i} + 0{,}368 \cdot acc_{ctr_i})$$

Em que, acc_{ts} é a acurácia do classificador medida com o conjunto de teste e acc_{ctr} é a acurácia do classificador medida com o conjunto de treinamento. O desvio padrão das acurácias pode também ser medido para verificar a instabilidade dos classificadores.

Esta técnica pode apresentar vantagens se a amostra disponível for pequena, dado que ela permite 'maximizar' o conjunto de treinamento. Por outro lado, se a amostra for muito grande, o custo de indução do classificador pode ser alto. Outra desvantagem é a impossibilidade de estratificação. Por fim, a acurácia do classificador escolhido também depende de características do banco de dados e do algoritmo de indução.

Segue-se uma discussão sobre alguns algoritmos de indução de conhecimento de conjuntos de treinamento, o conhecimento induzido sendo testado por meio de conjuntos de teste.

2.4 Algoritmos de Indução de Conhecimento

Nesta seção, são descritos quatro algoritmos de indução de conhecimento: *OneR*, *Naïve Bayes*, *ID3* e *Prism*. Destamos este quatro algoritmos, que serão utilizados pelo processo automatizado proposto, por serem amplamente discutidos e utilizados na literatura.

[6] O valor de *b* é fixado pelo minerador.

2.4.1 OneR

OneR [Witten and Eibe, 1999] é o algoritmo de indução de conhecimento que gera regras consideradas rudimentares. Basicamente o conhecimento inferido é representado na forma de uma árvore de decisão de um único nível, que é expressa na forma de um conjunto de regras para cada valor de um determinado atributo. Apesar de rudimentar, em diversos casos os classificadores induzidos por *OneR* podem atingir alto grau de acurácia, já que muitos bancos de dados do mundo real são também rudimentares e com apenas um atributo é possível determinar a classe de uma instância acertadamente. Para uma melhor compreensão, a figura 2.8 apresenta o algoritmo *OneR* em pseudocódigo, conforme retirado de [Witten and Eibe, 1999].

```
Para cada atributo:
        Para cada valor do atributo, construa uma regra da seguinte forma:
                conte quantas vezes o valor aparece em cada classe;
                encontre a classe mais freqüente para este valor;
                construa uma regra, assinalando a classe mais freqüente ao valor;
        Calcule a taxa de erro para as regras;
Escolha o conjunto de regras com menor taxa de erro.
```

Figura 2.8: Pseudocódigo de *OneR*

A idéia básica é a seguinte: Construir uma árvore de decisão para cada atributo e verificar a acurácia dessas árvores para o conjunto de treinamento. A árvore que apresentar melhor acurácia para o conjunto de treinamento será a escolhida.

Considere novamente o exemplo dos vazamentos. A tabela 2.6 mostra em detalhes a aplicação do algoritmo da figura 2.4 ao conjunto de treinamento da tabela 2.2.

Atributo	Regras	Erros	Total de Erros
Estado	ensolarado ✍ não*	2/4	3/10
	chuvoso ✍ sim	1/4	
	nublado ✍ sim	0/2	
Temperatura	**alta ✍ não**	0/2	**1/10**
	média ✍ sim	1/5	
	baixa ✍ sim	0/3	
Umidade	alta ✍ não	2/5	2/10
	normal ✍ sim	0/5	
Vento	sim ✍ sim	2/5	3/5
	não ✍ sim	1/5	

Tabela 2.6: Resultados da execução de *OneR*

A tabela mostra quatro conjuntos de regras, uma para cada atributo do conjunto de treinamento. O * nas regras indica que ouve empate na escolha da classe majoritária, e a escolha foi feita aleatoriamente. Observando o total de erros, o melhor classificador escolhido é do atributo Temperatura ? em negrito na tabela ? que obteve a menor taxa de erros, apenas 1/10, conseqüentemente obtendo a maior acurácia com o conjunto de treinamento.

2.4.2 Naïve Bayes

Naïve Bayes [Elder and Pregibon, 1996] é um algoritmo que utiliza modelagem estatística para inferir um classificador a partir de um conjunto de treinamento. Na modelagem estatística todos os atributos são considerados igualmente importantes e independentes um do outro. Essas suposições podem parecer irrealistas, já que no mundo real os atributos de um banco de dados certamente não são igualmente importantes, nem independentes uns dos outros. Contudo *Naïve Bayes* consegue excelentes resultados em muitos casos.

A idéia subjacente é verificar a probabilidade de cada par *atributo-valor* ocorrer em cada classe. O processo para verificação das probabilidades é simples e consiste em contar quantas vezes cada par *atributo-valor* ocorre em cada classe.

Considere à execução de *Naïve Bayes* na indução de um classificador probabilístico para o problema dos vazamentos. Inicialmente temos o conjunto de treinamento da tabela 2.2. A tabela 2.7 mostra as ocorrências de cada par *atributo-valor* em cada classe, juntamente com a probabilidade dessa ocorrência.

Atributos	**Valores**	**Classe**			
		Sim		**não**	
		Ocorrência	**Probabilidade**	**Ocorrência**	**Probabilidade**
Estado	ensolarado	2	2/7	2	2/3
	chuvoso	3	3/7	1	1/3
	nublado	2	2/7	0	0/3
Temperatura	alta	2	2/9	0	0/1
	média	4	4/9	1	1/1
	baixa	3	3/9	0	0/1
Umidade	alta	2	2/7	3	3/3
	normal	5	5/7	0	0/3
Vento	sim	3	3/7	2	2/3
	não	4	4/7	1	1/3
Vazamento	sim	7	7/10	0	-
	não	0	-	3	3/10

Tabela 2.7: Ocorrência e probabilidades para o problema do tempo

Observando a linha *Estado* na tabela 2.7 verifica-se que das 10 instâncias do conjunto de treinamento, 7 tem classificação *sim* e 3 a classificação *não.* Das 7 instâncias classificados como *sim*, em 2 *Estado = ensolarado*, em 3 *Estado=chuvoso* e em outras 2 *Estado=nublado*. Dessa forma as probabilidades apresentadas para a classe *sim*, na primeira linha da tabela foram: 2/7 para *Estado = ensolarado*, 3/7 para *Estado=chuvoso*, e 2/7 para *Estado=nublado*.

A classificação de uma instância é dada pela multiplicação das probabilidades correspondentes a cada par *atributo=valor* da instância, para cada uma das classes. A classe cuja multiplicação de probabilidades obtiver o maior resultado será a classificação resultante.

2.4.3 ID3

O algoritmo de extração do conhecimento *ID3* [Oates and Jensen, 1997] é um dos algoritmos da família TDIDT que induz classificadores na forma de árvores de decisão. Seu funcionamento é como segue: inicialmente, seleciona-se um atributo para ser o nodo raiz da árvore. Cada valor deste atributo torna-se um ramo da árvore. Isto decompõe o conjunto de treinamento em vários subconjuntos, um para cada valor do atributo.

O processo é então repetido recursivamente para cada ramo da árvore. Quando todos as instâncias do ramo tiverem a mesma classificação o processo é encerrado para aquele ramo. A figura 2.3 mostra um exemplo de árvore de decisão gerada pelo algoritmo *ID3* para o exemplo dos vazamentos.

Um problema que se apresenta aqui é saber escolher acertadamente quais são os melhores atributos para se tornarem raiz. Dependendo dos atributos escolhidos, podem ser geradas árvores muito 'largas', ou ainda árvores muito 'altas', o que é indesejável. Árvores muito 'altas' geram regras muito complexas, já árvores muito 'largas' podem levar a uma explosão de regras. O ideal é induzir a menor árvore possível.

Para a escolha de um atributo para a posição de nodo raiz, utiliza-se então uma medida denominada entropia. A *entropia* mede a desordem dos subconjuntos repassados a cada ramo e é inversamente proporcional ao *ganho* obtido com o nodo. Quanto maior a entropia, menor a capacidade do nodo em separar as instâncias com classificações diferentes. O atributo com menor entropia e conseqüentemente maior ganho na separação das classes é escolhido em detrimento dos outros. A entropia de um ramo do nodo é calculada da seguinte forma:

$$entropia(p_1, p_2, ..., p_n) = \frac{[-p_1 \log(p_1) - p_2 \log(p_2) ... - p_n \log(p_n)]}{p_1 + p_2 + ... + p_n}$$

em que p_1 é o número de instâncias do subconjunto de treinamento repassado ao ramo da classe 1, p_2 é o número de instâncias da classe 2 e assim por diante.

2.4.4 Prism

O algoritmo *Prism* [Bramer, 2000] infere regas de classificação com uma abordagem "botton-up", diferentemente dos algoritmos da família TDIDT que utilizam uma abordagem "top-down". A idéia do algoritmo *Prism* é abordar uma classe de cada vez, encontrando regras que cubram todas as instâncias do conjunto de treinamento com aquela classe. *Prism* é também é conhecido como um algoritmo de "convering" [Witten and Eibe, 1999].

O algoritmo, como apresentado em [Cendrowska, 1987] é mostrado na figura 2.9. Ele assume que existem pelo menos duas classes.

Para cada classe c_i:

(1) calcule a probabilidade de ocorrência da classe c_i para cada par *atributo=valor*;

(2) selecione o par *atributo=valor* com a probabilidade máxima de ocorrência e crie um subconjunto a partir do conjunto de treinamento (entrada), compreendendo todas as instâncias que contenham o par selecionado;

(3) repita os passos 1 e 2 para este subconjunto até o momento em que ele apresente apenas instâncias da classe c_i. A regra induzida é então a conjunção de todos os pares *atributo=valor* selecionados na criação deste subconjunto homogêneo;

(4) Remova todas as instâncias que satisfaçam a regra formada do conjunto de treinamento. Repita os passos 1 a 4 até que todas as instâncias do conjunto de treinamento tenham sido removidas.

Figura 2.9: O algoritmo *Prism*

A abordagem "button-up" de *Prism*, no entanto, gera altos custos de processamento e a acurácia dos classificadores gerados dependem de características do banco de dados a ser minerado. A figura 2.2 mostra o classificador induzido por *Prism* para o conjunto de treinamento do problema dos vazamentos.

2.5 Trabalhos Relacionados

Nesta seção destacamos três trabalhos relacionados, que são amplamente utilizados na inferência de conhecimento de bancos de dados. São eles: *DBMiner*, *IBM Inteligent Miner* e *WEKA*

2.5.1 DBMiner

Este sistema para descoberta de conhecimento em bancos de dados relacionais implementa várias tarefas de Mineração de Dados, tais como: regras de associação, regras de classificação, "clustering", etc. A integração com o Sistema Gerenciador de Banco de Dados (SGBD) é efetuada através da linguagem DMQL (*Data Mining Query Language – SQL based*) e é baseado na filosofia cliente/servidor, permitindo o acesso a uma BD via ODBC. Além disto, *DBMiner* [DBMiner, 2002] permite a mineração de bancos de dados OLAP.

Em se tratando de regras de classificação, *DBMiner* implementa apenas um algoritmo. Esse algoritmo pertence à família TDIDT, ou seja, árvores de decisão. *DBMiner* não implementa algoritmos de fragmentação, solicitando do usuário que determine a porcentagem dos dados disponíveis para formar o conjunto de teste. *DBMiner* também não implementa técnicas de amostragem.

A principal vantagem de utilizar *DBMiner* está em suas excelentes interfaces gráficas para visualização do conhecimento inferido. A figura 2.10 ilustra as interfaces gráficas.

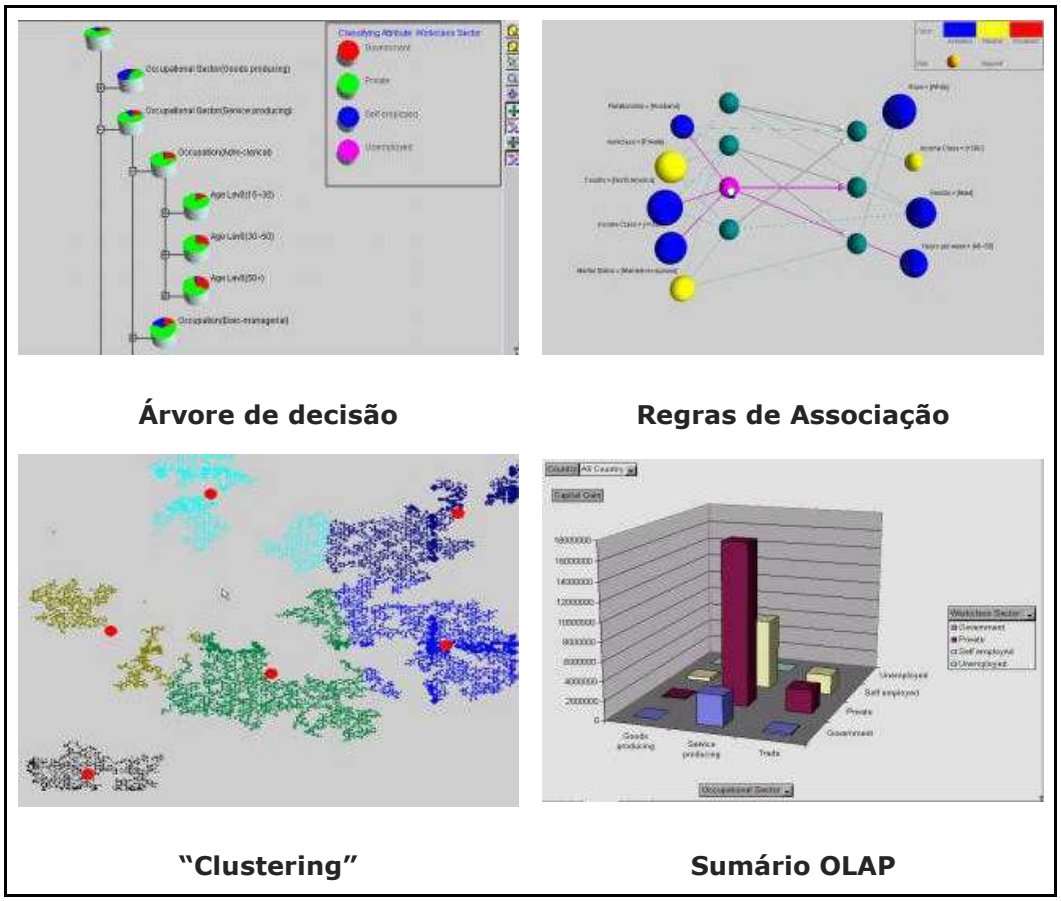

Figura 2.10: Interface gráfica de *DBMiner*

2.5.2 IBM Inteligent Miner

IBM Inteligent Miner [IBM, 2002] desenvolvido pela IBM é acoplado ao SGBD DB2. Ele implementa algoritmos para regras de associação, regras de classificação e "clustering", entre outros. Em se tratando de regras de classificação, implementa apenas um algoritmo da família TDIDT. Além disto, utiliza o paradigma de Redes Neurais Artificiais para prover classificadores e "clustering". Todos as técnicas implementadas estão disponíveis na forma de funções, as quais executam buscando dados diretamente no DB2.

Um dos pontos fortes de *IBM Inteligent Miner* são suas funções para preparação dos dados, antes da mineração. Essas funções permitem a totalização de valores, converter domínios de atributos, eliminar dados desconhecidos, substituir valores ausentes por constantes, tornar discretos domínios de atributos, filtrar atributos em busca de inconsistências, entre outros.

Em se tratando de amostragem, *IBM Inteligent Miner* permite a seleção de amostras aleatórias, desde que o tamanho da amostra seja indicado pelo usuário. Ele não implementa técnicas de fragmentação.

Outro ponto forte deste software é suas interfaces gráficas para visualização do conhecimento inferido. As quais facilitam a compreensão do usuário. A figura 3.11 ilustra a interface de visualizado de classificadores.

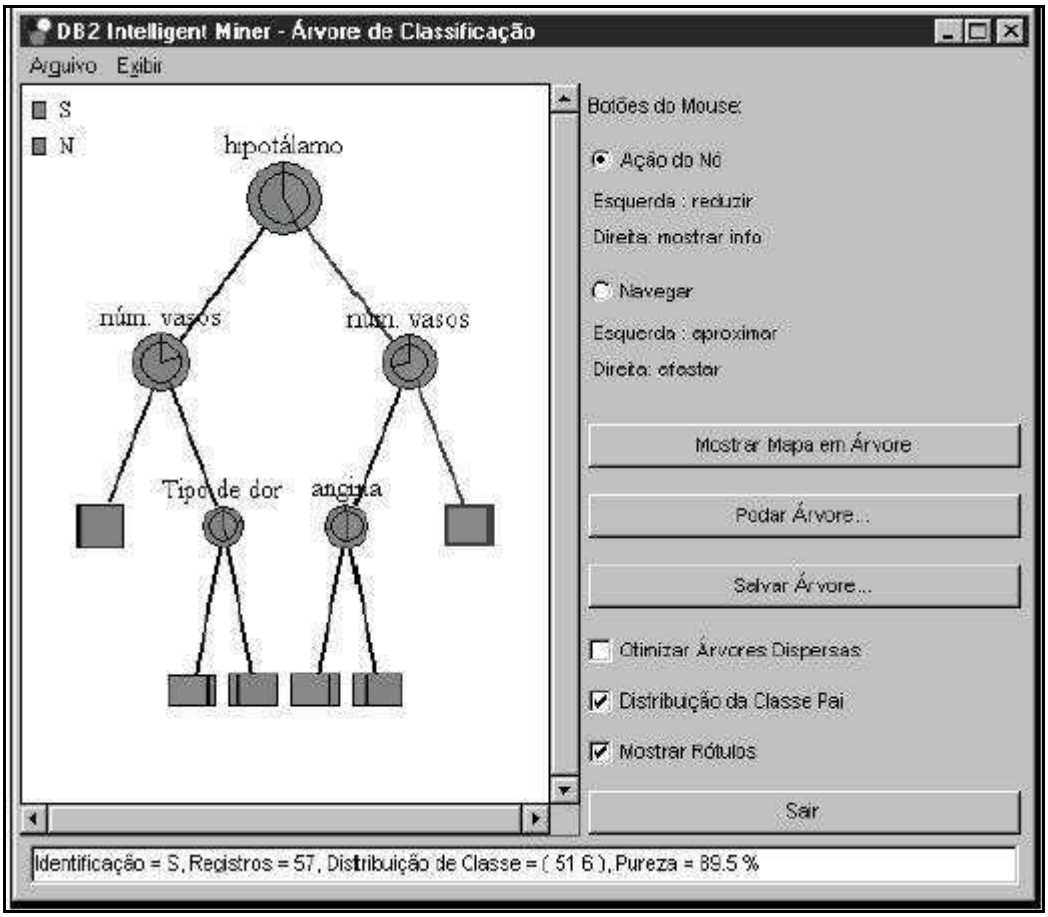

Figura 2.11: Interface gráfica de *IBM Inteligent Miner*

2.5.3 Framework WEKA

O framework *WEKA* [Witten and Eibe, 1999] (Waikato Environment for Knowledge Analisys) agrega uma variedade de algoritmos de indução de conhecimento. Dentre as dezenas de algoritmos para inferência de classificadores implementados em *WEKA*, estão *ID3, Prism, OneR* e *Naïve Bayes*, descritos na seção 2.4 deste capítulo.

Além disto, *WEKA* implementa o algoritmo de regras de associação *Apriori*, algoritmos para "clustering", entre outros. Em se tratando de técnicas de fragmentação de amostras, *WEKA* implementa apenas *K-fold Cross Validation*. *WEKA* não implementa técnicas de amostragem.

Sua condição de framework permite o fácil desenvolvimento de aplicações utilizando seus componentes, além disto, permite o acoplamento de novas técnicas rapidamente. A principal dificuldade de um usuário ao utilizar *WEKA* está na determinação da melhor técnica para seu problema de MD. Essa dificuldade só é resolvida com a experimentação manual realizada com cada técnica e comparação dos resultados obtidos.

2.6 Considerações Finais

Neste capítulo, foram apresentados diversos conceitos e técnicas de mineração de dados que são importantes para a compreensão dos capítulos seguintes. Dentre os conceitos definidos estão: banco de dados classificado, conjunto de treinamento, conjunto de teste, conjunto de execução, algoritmo de indução de conhecimento, classificador, acurácia de teste, acurácia de execução e amostra para MD. As técnicas discutidas foram: *Adaptive Incremental Framework* e *Convergência* (técnicas de seleção de amostras); *Holdout*, *K-fold Cross Validation* e *Bootstrap* (técnicas de fragmentação de amostras); *OneR*, *Naïve Bayes*, *ID3* e *Prism* (algoritmos de indução de conhecimento). Além disto, apresentamos os trabalhos relacionados: *IBM Inteligent Miner*, *DBMiner* e o framework *WEKA*.

No próximo capítulo, a automatização do processo de mineração de dados para inferência de classificadores é discutida.

Capítulo III

Um Processo Automatizado de Mineração de Dados para Inferência de Classificadores

O processo de MD para inferência de classificadores apresenta grande complexidade. Pesquisadores da área vêm aplicando seus esforços na diminuição dessa complexidade. Os esforços, no entanto, pecam por dois motivos:

1. Visões localizadas ou parciais do processo de mineração.
2. Pouca atenção à instabilidade dos algoritmos de mineração.
    a. Dependência de:
        i. Banco de dados a minerar;
        ii. Técnica de amostragem;
        iii. Técnica de fragmentação de amostras;
        iv. Algoritmo de indução de conhecimento;

Como ilustração, considere um exemplo. Dois mineradores diferentes têm a disposição um mesmo banco de dados, devidamente classificado, com informações sobre o cultivo de plantações de soja. O banco de dados não apresenta dados sujos, dados desconhecidos ou inexistentes e nem dados contínuos. A tarefa dos mineradores é inferir um classificador que determine, dado as características das plantas, qual a doença que está afetando a plantação.

Para solucionar o problema, Minerador1 utiliza a técnica de amostragem *Convergência*, a técnica de fragmentação *K-fold Cross Validation* e o algoritmo de indução de conhecimento *Naïve Bayes*.

Minerador2 utiliza a técnica de amostragem *Adaptive Incremental Framework*, a técnica de fragmentação *Bootstrap* e o algoritmo de indução *ID3*. A figura 3.1 apresenta os mineradores.

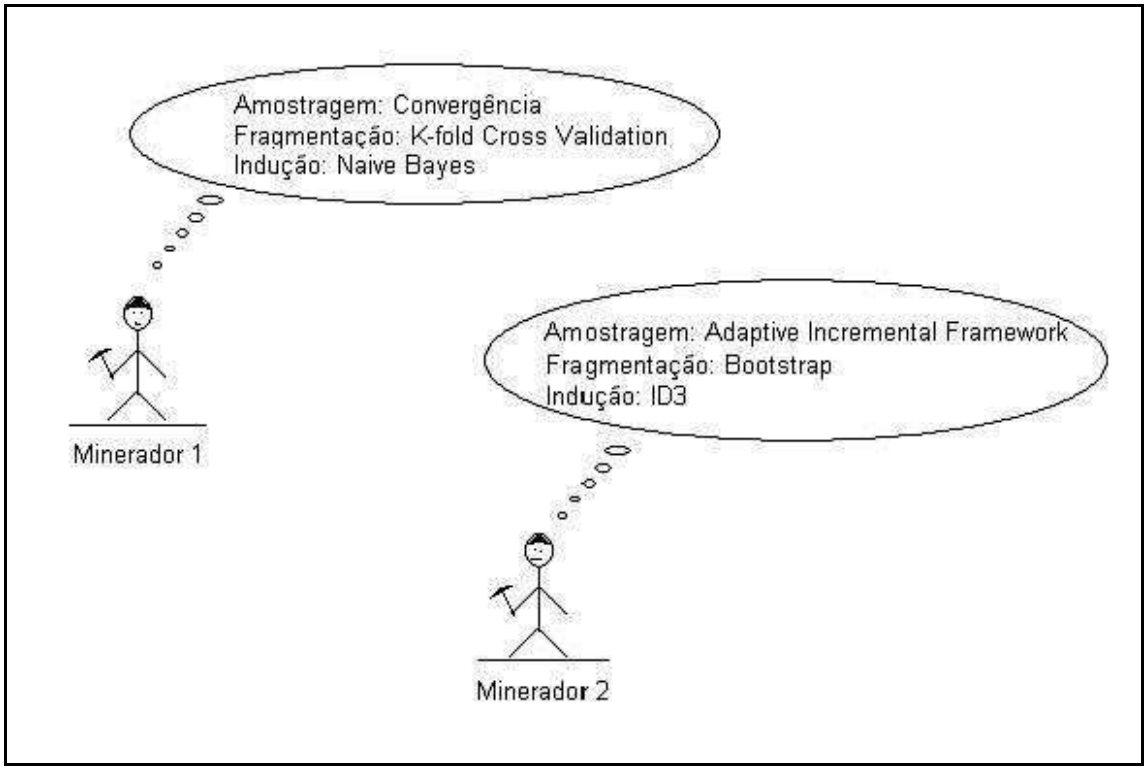

Figura 3.1: Diferentes mineradores com o mesmo problema

Como resultado, Minerador1 obteve o classificador *C1*, enquanto que Minerador2 obteve o classificador *C2*. O classificador *C1* é diferente do classificador *C2*. A acurácia de teste do classificador *C1* é de 91%. A acurácia de teste do classificador *C2* é de 67%.

Pior, mesmo que os mineradores escolhessem as mesmas técnicas de fragmentação e indução, a escolha de uma técnica de amostragem diferente resultaria em classificares diferentes, com acurácias diferentes. O mesmo vale para a técnica de fragmentação e o algoritmo de indução.

Em se tratando de bancos de dados diferentes para minerar, os resultados obtidos por técnicas variam muito. Essa instabilidade no resultado faz com que o minerador tenha diversas dúvidas após ter em mãos um classificador:

1. Esse é o melhor classificador para o problema?

2. Se eu tivesse utilizado outras técnicas não poderia obter um classificador melhor?
3. Para obter resultados melhores, quais etapas eu devo refazer?
4. Para obter resultados melhores, quais técnicas eu devo utilizar?

O que propomos é uma ferramenta que, considerando diferentes técnicas de amostragem, diferentes técnicas de fragmentação e diferentes algoritmos de indução, garanta ao minerador o melhor classificador possível para o problema. A escolha do melhor classificador pode passar por um processo de busca exaustiva dentre todas as possíveis combinações das técnicas disponíveis. A busca exaustiva é eficaz, entretanto, é impraticável quando a mineração ocorre manualmente. Mesmo se tratando de um processo automatizado, os custos de processamento são altos. Faz-se necessário, então, a busca de heurísticas para diminuir esse custo de processamento. Heurísticas tornam possível a implementação de versões mais otimistas do processo.

Relembrando o que já foi discutido na introdução, as etapas: preparação dos dados, seleção de amostras, fragmentação da amostra, inferência de conhecimento e avaliação do conhecimento são apresentadas novamente, em mais detalhes.

?? *Preparação dos dados* [Pyle, 1999] [Fayyad et al., 1997]: normalmente bancos de dados apresentam problemas como dados desconhecidos ou inexistentes e domínios de atributos contínuos que são transmitidos para a amostra e precisam ser corrigidos antes da extração do conhecimento. O tratamento de dados desconhecidos pode ser feito pela simples eliminação das instâncias que contém esse problema. Essa é uma abordagem adequada, pela grande dificuldade em diferenciar automaticamente dados desconhecidos de dados inexistentes. Outras técnicas para resolução de dados desconhecidos conduzem a utilização de algoritmos de indução para preencher valores de atributos desconhecidos, ou a utilização de constates para ocupar o lugar desses valores. Essas últimas duas abordagens levam a resultados errôneos se utilizados em casos onde é difícil diferenciar dados inexistentes de dados desconhecidos, podendo inviabilizar os resultados. Também na fase de preparação da amostra pode ser necessário transformar

um domínio de dados de valores contínuos para valores discretos. Para ilustrar, considere uma variável contínua de um banco de dados que armazene a idade de clientes cadastrados. Pode-se transformar essa variável em discreta utilizando faixas de valores (criança, jovem, adulto, idoso).

?? *Seleção de amostra* [Pyle, 1999] [Brumem et al., 2001]: A tarefa do minerador, nesta etapa do processo de MD, é determinar uma amostra do banco de dados a ser minerado que possa representar toda a 'população' durante as etapas posteriores do processo de MD. A importância da seleção de amostras representativas está no alto custo de processamento dos algoritmos de indução se todos os dados disponíveis no banco de dados são processados. Contudo, nenhuma das técnicas disponível atualmente garante bons resultados para todos os possíveis bancos de dados. Dessa forma, o grande problema do minerador é escolher qual das possíveis técnicas é a melhor para o banco de dados que ele quer minerar. Os resultados obtidos por uma técnica de amostragem dependem de características do banco de dados a ser minerado e de associações adequadas com técnicas de outras etapas do processo.

?? *Fragmentação do conjunto-amostra* em subconjuntos [Kohavi, 1995] [Witten and Eibe, 1999] [Jain and Dubes, 1997]: A tarefa do minerador nesta etapa é a divisão do conjunto amostra em subconjuntos de treinamento e teste. A dificuldade nesta etapa está em dividir a amostra de modo que, o conjunto treinamento seja suficientemente grande para manter a maioria das características presentes na amostra, e conseqüentemente, presentes na população, mas por outro lado, que o conjunto de teste seja suficientemente grande para determinar a qualidade do classificador inferido. Todavia, nenhuma das técnicas de fragmentação disponíveis atualmente garante bons resultados para qualquer que seja o banco de dados. Tornando difícil ao minerador, determinar qual das possíveis técnicas utilizar nesta etapa. Os resultados obtidos por uma técnica de fragmentação dependem de

características do banco de dados a ser minerado e de associações adequadas com técnicas de outras etapas do processo.

?? *Indução do Conhecimento* [Bramer, 2000] [Thrun et al., 1991] [Quinlan, 1993]: Algoritmos de indução de conhecimento são utilizados sobre o conjunto de treinamento para inferência de um classificador. O grande problema encontrado pelo minerador nesta etapa é escolher qual das possíveis técnicas ele deve utilizar. A qualidade do classificador inferido por um algoritmo de indução depende fortemente das características do banco de dados sendo minerado e de associações adequadas com técnicas de outras etapas do processo.

?? *Avaliação do conhecimento* inferido [Witten and Eibe, 1999]: Nesta etapa a tarefa do minerador é avaliar o conhecimento inferido através da acurácia de teste e da acurácia de execução. A acurácia de execução é calculada com a utilização da equação de Bernoulli. Diversas questões podem surgir para o minerador durante esta tarefa. Como por exemplo: O conhecimento inferido é o melhor possível para este banco de dados? Se tivesse utilizado outras técnicas, não seria possível obter resultados melhores?

A figura 3.2, onde estão dispostas as etapas típicas de um processo de MD. Existem várias técnicas para cada uma das etapas. De um modo geral, a grande dificuldade do minerador é determinar as técnicas que geram os melhores resultados para o particular banco de dados a ser minerado, já que é difícil comparar manualmente os resultados de duas técnicas quaisquer. Bancos de dados apresentam características que os diferenciam uns dos outros, e algumas técnicas podem gerar excelentes resultados para alguns bancos de dados e, gerar resultados inferiores para outros bancos de dados.

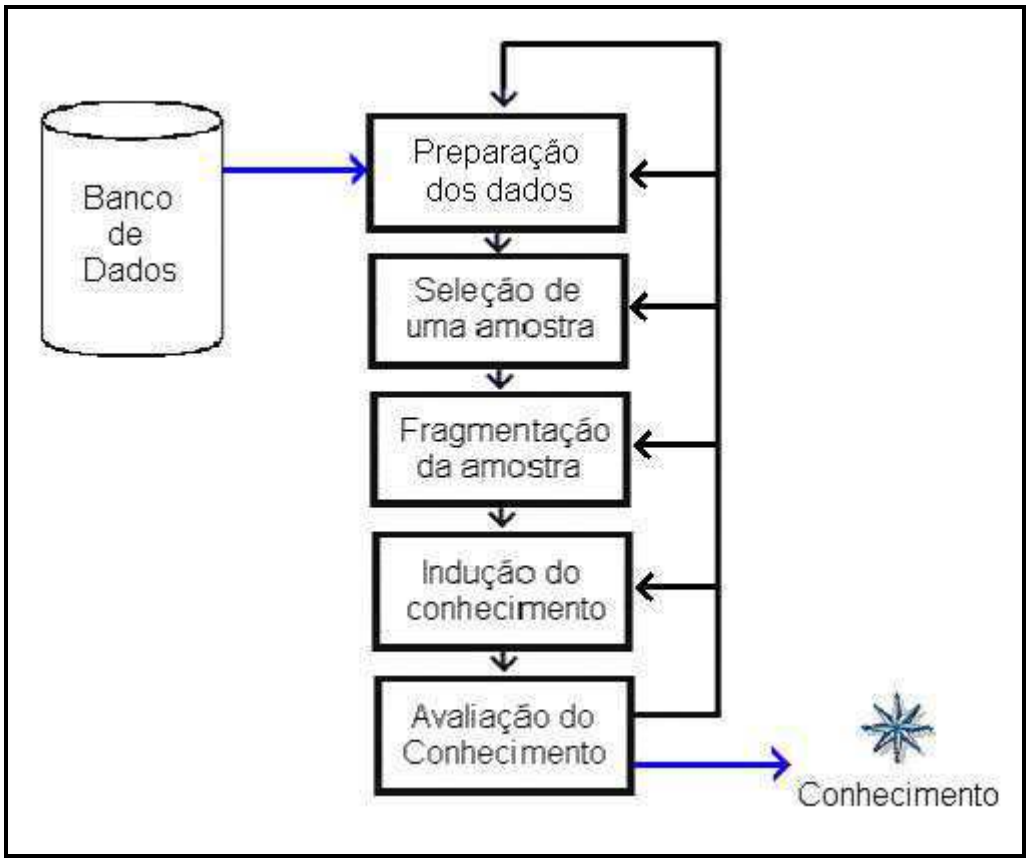

Figura 3.2: Etapas típicas de um processo de MD

Um importante ponto é que cada técnica é tipicamente mais adequada do que outras para um determinado grupo de banco de dados. Assim, não há técnicas que sejam universais e não existem critérios para decidir quais técnicas usar em cada circunstância. Essa falta de critérios torna o processo iterativo, como apresentado na figura 3.2. Quando o resultado final de uma execução seqüencial de técnicas ? técnica de amostragem, técnica de fragmentação, algoritmo de indução ? for um classificador de baixa qualidade, surge então, a necessidade de repetir o processo utilizando outras técnicas.

3.1 Requisitos de um processo de MD automatizado

Um processo de MD automatizado para inferência de classificadores deve considerar uma serie de requisitos fundamentais, para que sua utilização seja facilitada.

A seguir são descritos os requisitos básicos levantados, juntamente com uma breve explicação de cada um.

R1 ? **O processo deve contemplar todas as etapas apresentadas na figura 3.2.**

Na figura 3.2 podem ser visualizadas as cinco etapas do processo: preparação dos dados, seleção de amostra, fragmentação da amostra, inferência do conhecimento e avaliação do conhecimento. Todas essas etapas são igualmente importantes para a qualidade do conhecimento inferido e devem estar presentes no processo.

R2 ? **O processo deve iteragir dentre uma variedade de técnicas de amostragem diferentes.**

Como não existe uma técnica de amostragem universal, que obtenha bons resultados para todos os bancos de dados, o processo deve utilizar uma coleção de técnicas de amostragem, além de determinar qual técnica de amostragem é a mais adequada para o banco de dados sendo minerado. Algumas técnicas de amostragem são descritas na seção 2.2 do capítulo 2.

R3 ? **O processo deve iteragir dentre uma variedade de técnicas de fragmentação de amostras.**

Também não existe uma técnica de fragmentação de amostras que seja universal. Por esse motivo, o processo deve considerar a existência de várias técnicas de fragmentação diferentes, além de indicar qual delas obtém os melhores resultados para o banco de dados sendo minerado. A seção 2.3 no capítulo 2 trata das técnicas de fragmentação.

R4 ? **O processo deve iteragir dentre uma variedade de algoritmos de indução de conhecimento.**

Dezenas de algoritmos de indução de conhecimento estão disponíveis. A seção 2.4 no capítulo 2 detalha alguns desses algoritmos. O processo deve considerar a existência dessa diversidade de algoritmos e indicar qual o algoritmo de indução que obtém o melhor resultado para o banco de dados sendo minerado.

R5 ? **O processo deve selecionar um conjunto de técnicas, uma de cada**

etapa, que são as melhores para o banco de dados minerado.

Uma técnica de amostragem, uma técnica de fragmentação e um algoritmo de indução devem ser considerados os mais adequados para tratar o banco de dados do usuário.

R6 ? **O processo deve induzir o melhor classificador possível, com a utilização das técnicas disponíveis.**

O resultado de uma execução do processo automatizado deve ser o melhor classificador possível para o problema. Esse classificador deve ser inferido com as melhores técnicas selecionadas, dentre as técnicas disponíveis.

R7 ? **O processo deve estimar a acurácia de execução do melhor classificador.**

Após a determinação do melhor classificador, o processo automatizado deve informar a acurácia de execução estimada para o classificador. Para o cálculo da acurácia de execução deve ser utilizada a equação de Bernulli (seção 2.1 do capítulo 2).

3.2 O algoritmo *Naïve_Classifier_Inducer*

A figura 3.3 ilustra o algoritmo *Naïve_Classifier_Inducer*, em pseudocódigo, de acordo com os requisitos enumerados. Ele é "naïve" porque emprega exaustivamente toda a gama de técnicas, de variados tipos, que lhe são disponíveis, ou seja, vários algoritmos de indução, diversas técnicas de amostragem e várias técnicas de fragmentação.

A entrada para o algoritmo é um banco de dados classificado para ser minerado. A função *Prepare* "encapsula" diversos aspectos da preparação de dados ? limpeza de dados 'sujos', tratamento dos dados desconhecidos, etc ? , retornando o banco de dados 'limpo'.

São consideradas três coleções não vazias de técnicas. A primeira coleção, denominada *Coleção I,* é composta por algoritmos de indução de conhecimento. A segunda coleção, denominada *Coleção A* é composta de técnicas de amostragem. Os

elementos da terceira coleção, *Coleção F*, são técnicas de fragmentação. Finalmente, a *Coleção C* armazena os classificadores induzidos.

***Naïve_Classifier_Inducer*(banco de dados classificado *D*)**
Prepare(*D*)
Para cada *algoritmo de indução* em *Coleção I*
    Para cada *técnica de amostragem* em *Coleção A*
        Para cada *técnica de fragmentação* em *Coleção F*
            Selecione uma amostra *S* de *D*
            Para cada par$_i$ $D_{ctr} - D_{cts}$ de *S*
                Infira um classificador$_i$ a partir de D_{ctr}
                Calcule a acurácia$_i$ do classificador$_i$ com D_{cts}
            Fim_Para
            Calcule a média ? e o desvio padrão ? das acurácias
            Selecione um dos classificadores e atribua a acurácia ? - ? a ele
            Armazene o classificador selecionado em *Coleção C*
        Fim_Para
    Fim_Para
Fim_Para
Ordene os classificadores em *Coleção C* considerando ? - ?
Calcule a acurácia estimada para o classificador com melhor ? - ?
Retorne o classificador com melhor ? - ?, a acurácia estimada e a *Coleção C*.
Fim_Naïve_Classifier_Inducer

Figura 3.3: O algoritmo *Naïve_Classifier_Inducer*

O algoritmo retorna o melhor classificador encontrado ? etapa de avaliação de conhecimento ? , que é, o que obtiver a melhor ? - ?, juntamente com sua acurácia estimada. A acurácia estimada, ou acurácia de execução é calculada pela fórmula de Bernulli. Além disto, a *Coleção C* que contém o ranking dos classificadores também é retornada.

Os fatores ? e ? representam respectivamente a acurácia média de um classificador e sua estabilidade ? desvio padrão. Quanto maior a acurácia media e menor o desvio padrão, melhor será o classificador.

Sejam C_1 e C_2 dois classificadores. O classificador C_1 é melhor que C_2 se:

$$(C_1? - C_1?) ? (C_2? - C_2?)$$

Em que $C_1?$ é a acurácia média atribuída ao classificador C_1, $C_2?$ é a acurácia média atribuída ao classificador C_2, $C_1?$ é o desvio padrão da média atribuído ao classificador C_1 e $C_2?$ é o desvio padrão atribuído ao classificador C_2.

Como ilustração, considere que a acurácia média ? e o desvio padrão ? atribuídos ao classificador C_1 sejam respectivamente 92% e 3%. E para o classificador C_2 sejam respectivamente 93% e 5%. Nesse caso, (92%-3%) ? (93%-5%) e o classificador C_1 seria considerado o melhor.

O algoritmo *Naïve_Classifier_Inducer* é pessimista no sentido que investiga exaustivamente todas as combinações de técnicas de amostragem, técnicas de fragmentação e algoritmos de indução. Como assinalado anteriormente, essa característica é o que faz o algoritmo ser chamado de 'ingênuo'. Como todo algoritmo 'ingênuo', a grande vantagem é que ele escolhe o 'melhor' classificador, no sentido matemático do termo, dentre as diversas técnicas disponíveis ao algoritmo. Por outro lado, todo algoritmo 'ingênuo' tende a ter um alto custo de processamento, devido à pesquisa exaustiva. A preocupação com o alto custo potencial do nosso algoritmo nos levou a pensar em uma nova versão do mesmo, baseada em heurísticas que possam ajudar a diminuir significativamente os custos do algoritmo, e ainda selecionando um classificador de qualidade.

3.3 Heurísticas

Após uma intensa experimentação realizada com o algoritmo *Naïve_Classifier_Inducer*, empregando nove bancos de dados, duas técnicas de amostragem, três técnicas fragmentação e quatros algoritmos de indução, foi possível

chegar a três heurísticas, que serão apresentadas nesta seção. As heurísticas permitiram a construção de uma nova versão do algoritmo, *Expert_Classifier_Inducer*, de menor custo computacional em relação à versão *Naïve_Classifier_Inducer*. Os testes com o *Expert_Classifier_Inducer* permitiram verificar que o mesmo induziu classificadores com praticamente a mesma qualidade dos induzidos com o *Naïve_Classifier_Inducer*, para as mesmas técnicas. O capítulo 5 é dedicado aos experimentos com as duas versões do algoritmo. A seção 5.2 do capítulo 5 apresenta detalhadamente os experimentos realizados para observação das heurísticas.

Segue, uma lista com três heurísticas.

Heurística 1: Se a técnica de amostragem *A1* tem resultados melhores que as outras técnicas de amostragem para o banco de dados *X* em uma certa combinação de técnicas de fragmentação e indução, então a vantagem de *A1* sobre as outras técnicas de amostragem pode ser assumida para *X*, independentemente das outras combinações com as demais técnicas de fragmentação e indução.

Heurística 2: Considerando a melhor técnica de amostragem (Heurística 1). Se a técnica de fragmentação *F1* tem resultados melhores que as outras técnicas de fragmentação para o banco de dados *X* em uma certa combinação que esteja presente a melhor técnica de amostragem para *X*, a vantagem de *F1* sobre as outras técnicas de fragmentação pode ser assumida para *X*, independentemente dos algoritmos de indução.

Heurística 3: Considerando a melhor técnica de amostragem (Heurística 1) e a melhor técnica de fragmentação (Heurística 2). Se o algoritmo de indução *I1* tem resultados melhores que os outros algoritmos de indução para o banco de dados *X* em uma certa combinação que esteja presente a melhor técnica de amostragem e a melhor técnica de fragmentação para *X*, pode-se então assumir que *I1* é o melhor algoritmo de indução para *X*.

3.4 O Algoritmo *Expert_Classifier_Inducer*

A figura 3.4 apresenta o algoritmo *Expert_Classifier_Inducer* em pseudocódigo. O algoritmo foi desenvolvido com a utilização das heurísticas apresentadas na seção anterior. Assim como para a versão ingênua do algoritmo, a função *Prepare* implementa alguns aspectos da preparação de dados.

Expert_Classifier_Inducer **(classified dataset *D*)**

Prepare(*D*)

/* Etapa #1:Determinando melhor técnica de amostragem para *D* */

Selecione aleatoriamente um algoritmo de indução de *Coleção I* e armazene em *Z*
Selecione aleatoriamente uma técnica de fragmentação de *Coleção F* e armazena em *Y*
Para cada técnica de amostragem X_i em *Coleção A*
    Infira um classificador de *D* com a combinação X_i, *Y, Z* e adicione em *Coleção C*
Determine a melhor técnica de amostragem comparando os classificadores de *Coleção C* através dos fatores ? -? e armazene em *X*
Limpe a *Coleção C*

/* Etapa #2:Determinando melhor técnica de fragmentação para *D* */

Para cada técnica de fragmentação Y_i em *Coleção F*
    Infira um classificador de *D* com a combinação *X,* Y_i, *Z* e adicione em *Coleção C*
Determine a melhor técnica de fragmentação comparando os classificadores de *Coleção C* através dos fatores ? -? e armazene em *Y*
Limpe a *Coleção C*

/* Etapa #3: Determinando melhor algoritmo de indução para o D */
Para cada algoritmo de indução Z_i em *Coleção I*
    Infira um classificador de *D* com a combinação *X, Y,* Z_i e adicione em *Coleção C*
Determine o melhor algoritmo de indução comparando os classificadores de *Coleção C* através dos fatores ? -?

Calcule a acurácia estimada para o melhor classificador presente em *Coleção C*

Retorne o melhor classificador e a acurácia estimada

End_Expert_Classifier_Inducer

Figura 3.4: O algoritmo *Expert_Classifier_Inducer*

O algoritmo *Expert_Classifier_Inducer* é 'inteligente', no sentido que utiliza três heurísticas para diminuir o custo de determinar quais as melhores técnicas de amostragem, fragmentação e indução para um determinado banco de dados, evitando assim os testes com todas as combinações possíveis das técnicas, ou evitando a pesquisa exaustiva.

A comparação dos classificadores induzidos para a escolha do 'melhor' é realizada de forma idêntica ao algoritmo *Naïve_Classifier_Inducer*.

3.5 Diferença Computacional Entre os Algoritmos *Naive_Classifier_Inducer* e *Expert_Classifier_Inducer*

Note que, o número de combinações computadas por *Naive_Classifier_Inducer* é igual a ?conjunto de técnicas de amostragem?? ?conjunto de técnicas de fragmentação? ? ?conjunto de algoritmos de indução?. Esse número pode ser muito grande, se alguns dos conjuntos forem grandes. Com o auxílio das heurísticas, o algoritmo *Expert_Classifier_Inducer* escolhe apenas algumas das combinações possíveis, ou seja, ?conjunto de técnicas de amostragem? + ?conjunto de técnicas de fragmentação? + ?conjunto de algoritmos de indução?.

Considere um exemplo, caso o número de técnicas de amostragem, fragmentação e indução componentes dos conjuntos sejam respectivamente 10, 10 e 10, o número de combinações computadas por *Naive_Classifier_Inducer* será 10x10x10=1000, já o número de combinções computadas por *Expert_Classifier_Inducer* será 10+10+10=30.

3.6 Considerações Finais

Neste capítulo, inicialmente foi discutida a complexidade do processo de MD para inferência de classificadores. Para isso, um exemplo de motivação foi apresentado. O exemplo concluiu a existência de grande instabilidade nos resultados obtidos por técnicas diferentes em um mesmo banco de dados e grande instabilidade nos resultados

das mesmas técnicas empregadas em bancos de dados diferentes. Na seqüência foi apresentado um algoritmo para automatizar esse processo ? o algoritmo *Naïve_Classifier_Inducer* ? que induz classificadores para cada combinação possível de técnicas amostragem – fragmentação – indução. Fica evidente que, em presença de um grande número de técnicas, o custo computacional do algoritmo é inaceitável.

Três heurísticas foram apresentadas visando diminuir o custo computacional do algoritmo. A descoberta das heurísticas foi realizada com a análise dos resultados obtidos pelo algoritmo *Naïve_Classifier_Inducer*. A avaliação experimental é apresentada em detalhes no capítulo 5. Com a utilização das heurísticas, uma nova versão do algoritmo foi proposta ? *Expert_Classifier_Inducer*. O custo computacional de *Expert_Classifier_Inducer* deve ser muito inferior a versão 'ingênua', especialmente com a utilização de um grande número de técnicas de amostragem, técnicas de fragmentação e algoritmos de indução de conhecimento. O custo computacional inferior, deve ser confirmado na seção de experimentações.

O próximo capítulo é sobre a implementação dos dois algoritmos.

Capítulo IV

Um Framework para Mineração de Dados

Este capítulo descreve as etapas de análise e projeto de um software que automatiza o processo de mineração de dados. Inicialmente são apresentados os requisitos do software. Na seqüência a análise é discutida e por fim a arquitetura do software é apresentada.

4.1 Requisitos

Para o desenvolvimento do software, foram levantados diversos requisitos funcionais e não funcionais.

4.1.1 Requisitos Funcionais

O software deve:

RF1 ? **Implementar os dois algoritmos apresentados no capítulo 3.**
No capítulo anterior, dois algoritmos foram propostos visando automatizar o processo de MD para inferência de classificadores. Esses dois algoritmos devem ser implementados, constituindo o núcleo do software. A escolha de qual algoritmo utilizar deve ser feita pelo minerador.

RF2 ? **Implementar diversas técnicas de amostragem.**
Uma coleção de técnicas de amostragem deve ser implementada.

Essa coleção deve conter pelo menos duas técnicas diferentes, como por exemplo *Adaptive Incremental Framework* e *Convergência*.

RF3 ? **Permitir a fácil adição de novas técnicas de amostragem ao processo de MD.**

Constantemente, novas técnicas de amostragem são elaboradas e aperfeiçoadas. O software desenvolvido deve permitir a rápida e fácil inclusão dessas novas técnicas de amostragem ao processo de MD.

RF4 ? **Implementar técnicas de fragmentação de amostra.**

O software deve implementar uma coleção de técnicas de fragmentação de amostras, com pelo menos as técnicas *Holdout*, *K-Fold Cross Validation* e *Bootstrap*.

RF5 ? **Permitir a fácil adição de novas técnicas de fragmentação ao processo.**

Assim como as técnicas de amostragem, novas técnicas de fragmentação de amostras podem ser propostas e é importante que a inclusão dessas novas técnicas seja facilitada.

RF6 ? **Implementar algoritmos de indução de conhecimento.**

A literatura é pródiga em algoritmos de indução de conhecimento. O software deve implementar pelo menos quatro desses algoritmos: *OneR*, *ID3*, *Naïve Bayes* e *Prism*.

RF7 ? **Permitir a fácil adição de novos algoritmos de indução de conhecimento ao processo.**

Constantemente novos algoritmos de indução de conhecimento são desenvolvidos e o software deve permitir a rápida e fácil inclusão desses novos algoritmos ao processo.

RF8 ? **Implementar técnicas para o tratamento de dados desconhecidos e técnicas para tornar discretos domínios de variáveis contínuas.**

Um componente de preparação de dados deve ser implementado. Esse componente deve ser responsável pelo tratamento de dados desconhecidos e inexistentes, bem como, pela "discretização" de domínios de variáveis, caso necessário.

RF9 ? **Permitir a fácil adição de novas técnicas de preparação de dados**

ao processo.

Assim como as demais técnicas, novas técnicas responsáveis pela preparação da amostra podem ser propostas. O software deve permitir a inclusão dessas novas técnicas ao processo de forma facilitada.

RF10 ? **Permitir que o minerador selecione o banco de dados a ser minerado.**

O banco de dados a ser minerado deve ser indicado pelo minerador ao software. Esse banco de dados deve ser classificado e o atributo de classificação também deve ser indicado pelo minerador.

RF11 ? **Permitir que o minerador selecione as técnicas de amostragem que quer utilizar no processo.**

A cada execução do processo, o minerador deve indicar quais técnicas de amostragem deseja que sejam utilizadas. O objetivo desse requisito é diminuir custo de processamento em casos onde o minerador saiba de antemão quais são as melhores técnicas de amostragem para aquele banco de dados ? casos em que ele já tenha executado o processo anteriormente sobre o mesmo banco de dados.

RF12 ? **Permitir que o minerador selecione as técnicas de fragmentação que deseja que o processo utilize.**

Assim como para as técnicas de amostragem, o minerador pode diminuir o custo de processamento caso saiba de antemão as técnicas de fragmentação que geram os melhores resultados para um determinado banco de dados que ele quer minerar.

RF13 ? **Permitir que o minerador selecione os algoritmos de indução que deseja que o processo utilize.**

Dado uma coleção de algoritmos de indução que estarão implementados, o minerador pode saber antecipadamente quais os algoritmos que geram os melhores resultados para o banco de dados que ele quer minerar e selecionar apenas estes ? também em casos em que ele já tenha executado anteriormente o processo sobre o banco de dados. O objetivo é diminuir o custo de processamento.

RF14 ? **Permitir que o minerador estabeleça uma margem mínima de confiança para a acurácia do classificador.**

A confiança da acurácia estimada para o conjunto de execução deve ser indicada pelo minerador. Essa confiança deve ser utilizada para a escolha do parâmetro *z* na equação de Bernoulli.

RF15 ? **Permitir que o minerador encerre a execução do processo.**

Em qualquer momento da execução do processo, o minerador terá a opção de abortá-lo. Isso é necessário em casos que considere o custo de processamento muito alto.

4.1.2 Requisitos Não Funcionais

RNF1 ? **Interface do minerador.**

Faz-se necessária à existência de uma interface gráfica que auxilie os mineradores na configuração e execução do processo.

RNF2 ? **Plataformas de execução.**

Deve existir independência de plataforma por parte do software, isto é, sua implementação deve ser realizada em uma linguagem de programação que não esteja condicionada a uma plataforma específica.

RNF3 ? **Perfil dos usuários.**

Dois tipos de usuários devem ser considerados. O primeiro tipo caracteriza os usuários que não conhecem programação de software e tampouco conhecem mineração de dados; porém, têm conhecimento sobre o domínio e semântica dos dados, e desejam utilizar o software para inferir classificadores. O segundo tipo de usuários refere-se a pessoas que têm experiência em programação de sistemas e mineração de dados.

4.2 Casos de Uso

Com base na especificação dos requisitos funcionais e não funcionais, foram definidos diversos casos de uso. O modelo de casos de uso tem o propósito de representar as possíveis interações que podem ocorrer entre o sistema e seus atores externos ? os usuários do sistema.

Considerando o conjunto dos requisitos funcionais e não funcionais, podemos identificar dois atores: O *Minerador* e o *Construtor de Componentes*. Um ator é um usuário com os papéis assumidos na sua interação com o sistema.

A figura 4.1 ilustra o comportamento do ator Construtor de Componentes.

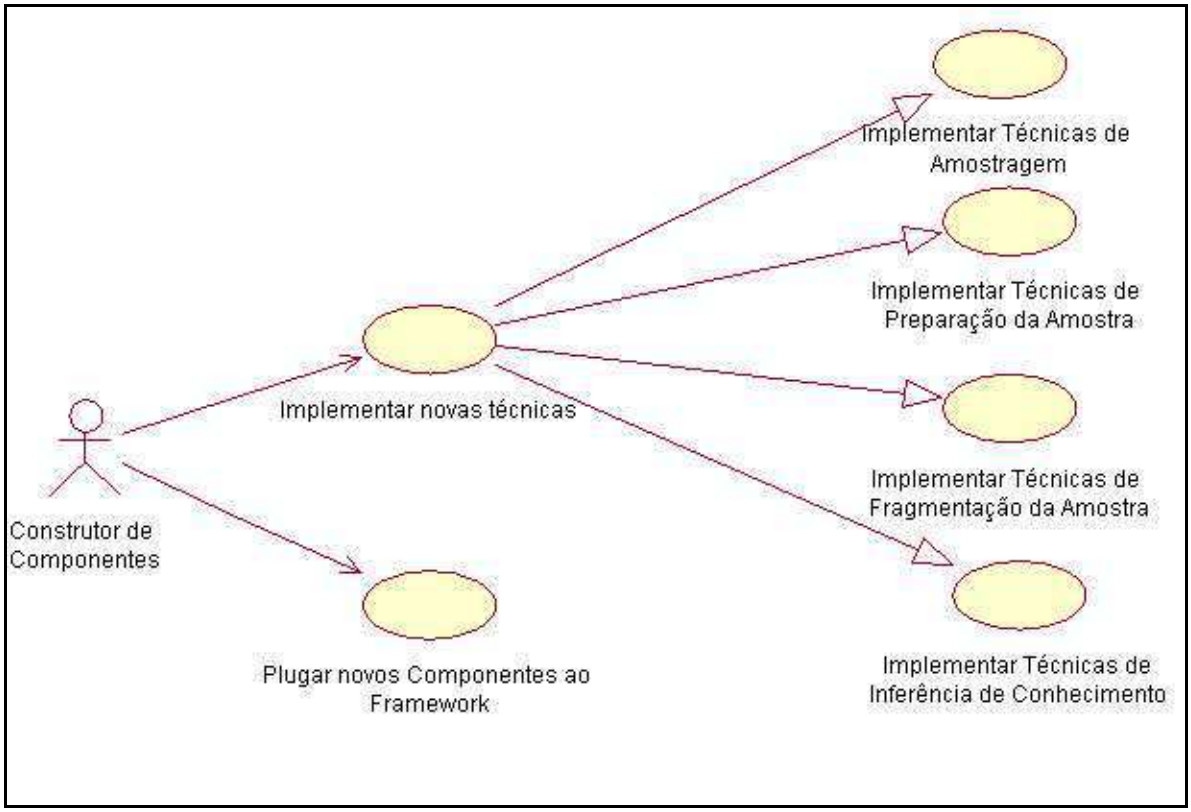

Figura 4.1: Diagrama de casos de uso do ator Construtor de Componentes

CU1 - *Implementar novas técnicas*: Permite ao Construtor de Componentes implementar novas técnicas na forma de componentes de software. As técnicas que podem ser implementadas pertencem a quatro famílias diferentes: técnicas de preparação de dados, técnicas de amostragem,

técnicas de fragmentação da amostra e algoritmos de indução de conhecimento. Essa implementação deverá obedecer a interfaces específicas que definirão o acoplamento entre o componente desenvolvido e o software. Essas interfaces serão definidas na seção 4.3 deste capítulo. Esse caso de uso esta diretamente relacionado com os requisitos funcionais RF3, RF5, RF7, RF9.

CU2 - *Conectar novos componentes ao framework*: Consiste na ação de adicionar ao software um novo componente implementado pelo Construtor de Componentes visando permitir que esse novo componente seja utilizado na execução do processo.

A figura 4.2 ilustra o modelo de casos de uso do ator Minerador.

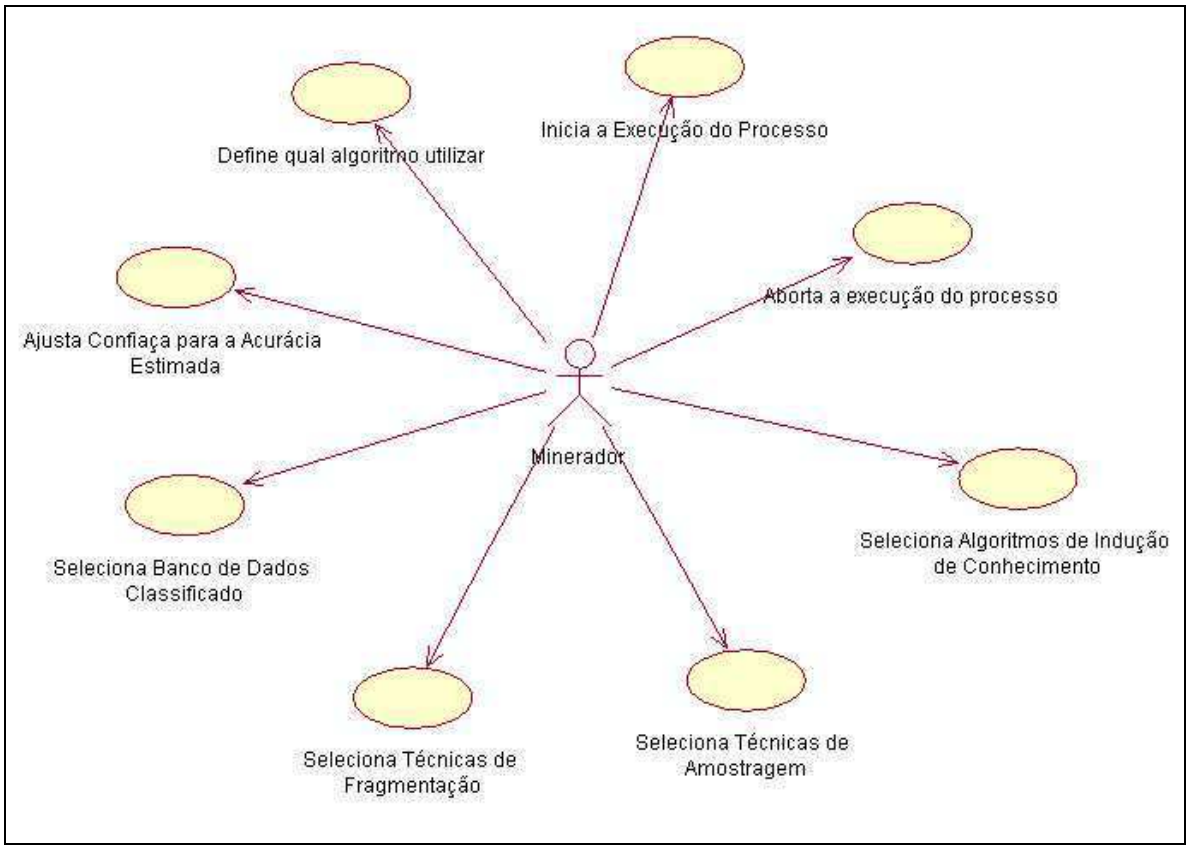

Figura 4.2: Diagrama de casos de uso do ator Minerador

CU3 - *Seleciona técnicas de amostragem*: Uma das interações possíveis entre o Minerador e o software consiste na seleção das técnicas de amostragem que

farão parte da coleção de técnicas utilizadas durante a execução do processo. Esse caso de uso está diretamente relacionado com o requisito RF11.

CU4 - *Seleciona técnicas de fragmentação*: Esse caso de uso visa atender o requisito funcional RF12, possibilitando ao Minerador estabelecer quais técnicas de fragmentação farão parte da coleção de técnicas de fragmentação que o processo utilizará durante sua execução.

CU5 - *Seleciona algoritmos de indução de conhecimento***:** Os algoritmos que automatizam o processo de MD utilizado como núcleo do software necessitam de uma coleção de algoritmos de indução de conhecimento. A seleção dos algoritmos que estarão presentes nessa coleção deve ser feita pelo Minerador. Esse caso de uso visa atender ao requisito funcional RF13.

CU6 - *Seleciona banco de dados classificado***:** É tarefa do Minerador indicar o banco de dados a ser minerado, bem como o atributo de classificação do mesmo. Esse caso de uso está relacionado com o requisito funcional RF10.

CU7 - *Ajusta confiança para acurácia estimada***:** Outra interação entre o software e o Minerador deve permitir que o usuário especifique a confiança sobre a acurácia estimada para o conjunto de execução ? RF14.

CU8 - *Define qual algoritmo utilizar*: estarão disponíveis ao Minerador dois algoritmos diferentes, que automatizam o processo: *Naïve_Classifier_Inducer* e *Expert_Classifier_Inducer*. O Minerador deve selecionar um desses algoritmos.

CU9 - *Inicia a execução do processo***:** Após o Minerador ter indicado o banco de dados classificado para mineração, ter selecionado todas as técnicas utilizadas em cada uma das etapas do processo e ter ajustado a confiança sobre a acurácia estimada para o conjunto de execução, o Minerador deve ter a opção de iniciar a execução do processo.

CU10 - *Aborta a execução do processo***:** O Minerador deve ter a opção de encerrar a execução do processo a qualquer momento. Essa é uma opção que pode ser utilizada em casos que considere os custos computacionais muito elevados e queira abandonar aquela execução. Esse caso de uso visa atender o requisito funcional RF15.

CU11 - *Solicita ranking de classificadores*: Um ranking de classificadores é gerado durante a execução do algoritmo *Naïve_Classifier_Inducer*. Esse caso de uso visa permitir que o Minerador tenha a opção de visualizar esse ranking após a execução do processo.

4.3 Arquitetura do Software

Visando atender os casos de uso CU1 e CU2 do Construtor de Componentes, os quais conduzem ao desenvolvimento de um software reutilizável e facilmente extensível, a análise do software foi realizada visando construir um framework orientado a objeto. No framework apresentado aqui, a adição de novos componentes se dá através do mecanismo de herança. Esse tipo de framework também é conhecido na literatura como "White Box" [Gamma et al., 1994].

O projeto do software baseou-se em uma arquitetura de três camadas conforme ilustra a figura 4.3. As três camadas propostas são: Apresentação, Aplicação e Dados.

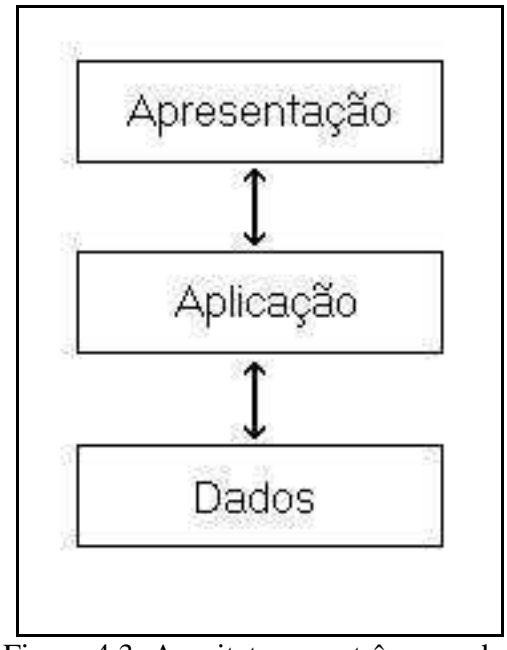

Figura 4.3: Arquitetura em três camadas

4.3.1 A Camada de Dados

A camada de *Dados* é onde se encontram os dados a serem minerados pela camada de aplicação. Estes podem estar dispostos em diferentes formatos, como arquivos texto ou até mesmo um Sistema de Gerência de Banco de Dados (SGBD). A figura 4.4 apresenta, em notação UML, o diagrama das classes que representa a camada de dados no framework.

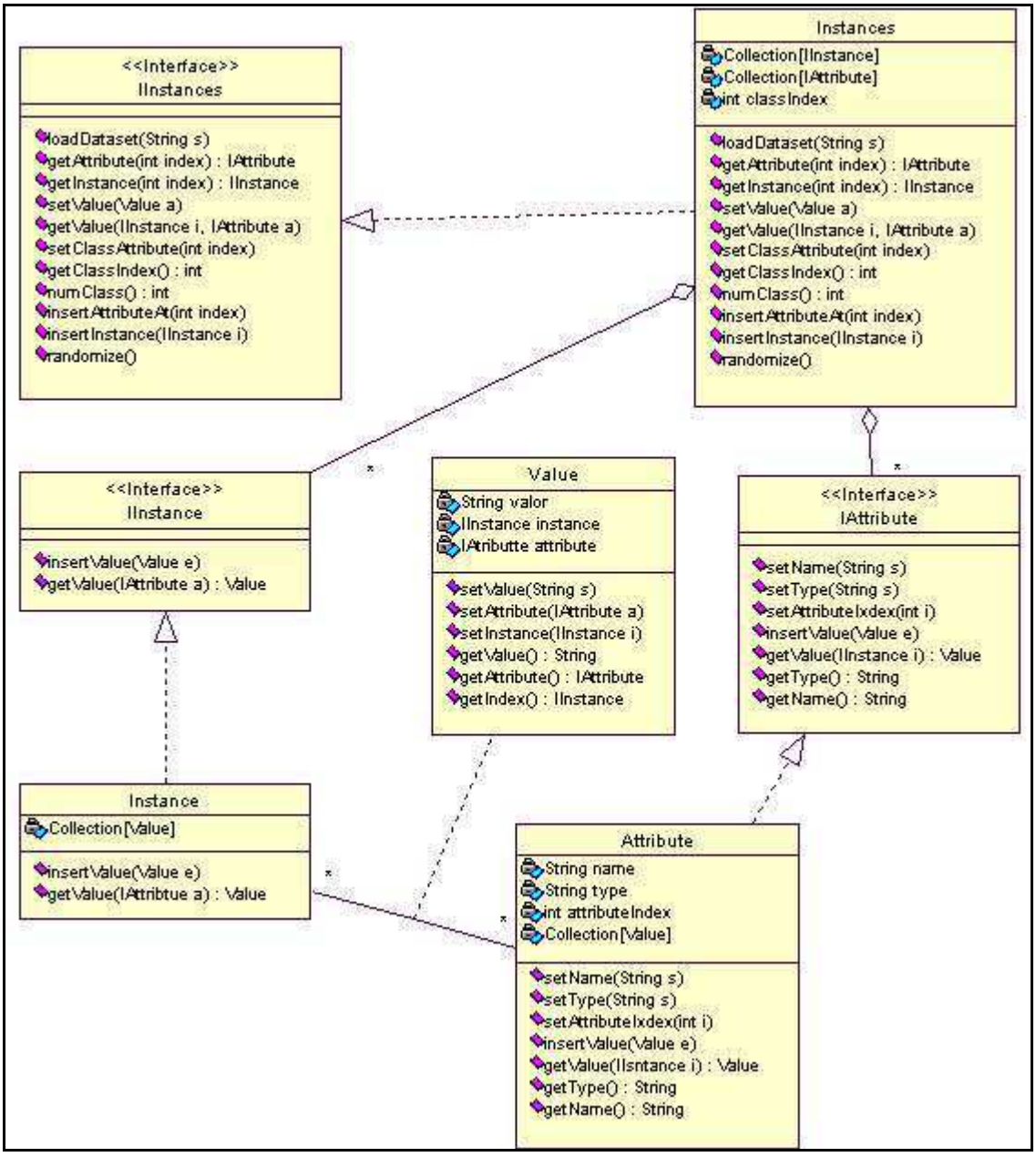

Figura 4.4: Diagrama das classes que representam o banco de dados

Três interfaces foram especificadas no diagrama: *IInstances*, *IInstance* e *IAttribute*. A classe *Instances* implementa *IInstances* e agrega duas coleções ? uma coleção de objetos de classes que implementam *IInstance* e uma coleção de objetos instanciados a partir de classes que implementam a interface *IAttribute* ? . As classes *Instance* e *Attribute* herdam de *IInstance* e *IAttribute* respectivamente. Além das duas coleções, *Instances* armazena o índice do atributo de classificação do banco de dados na variável do tipo *inteiro classIndex*.

Cada par de objetos *Instance – Atributo* compartilham um objeto da classe *Value*. Um objeto da classe *Value* armazena uma *String* que representa um valor de um atributo em uma instância do banco de dados. Além disso, guarda referências para um objeto da classe *Attribute* e um objeto da classe *Instance*, as quais pertence ? pelos atributos *instance* e *attribute*.

Além de uma coleção de *Value*, a classe *Attribute* registra o nome do atributo, forma de uma *String name*, o tipo do atributo, na *String type* e o índice do atributo no banco de dados, pelo *inteiro attributeIndex.*

São métodos definidos na interface *IInstance*:

?? *insertValue(Value e)*: permite a inserção de um valor na instância. O parâmetro *e* deve ter seu valor e a referência para o objeto da classe *Attribute* já ajustado.

?? *getValue(IAttribute a):Value*: Retorna o *Value* da instância para o atributo especificado no paramento *a*.

São métodos definidos na interface *IAttribute*:

?? *setName(String s)*: Define o nome do atributo como sendo o parâmetro *s*.

?? *setType(String s)*: O parâmetro *s* informa o tipo do atributo. Os possíveis tipos são: “nominal” e “numeric”.

?? *setAttributoIndex(int i)*: Esse método define o índice do atributo, que é informado pelo parâmetro *i*.

?? *insertValue(value e)*: permite a inserção de um valor ao atributo. O parâmetro *e* deve ter seu valor e a referência para o objeto da classe *Instance* já ajustados.

?? *getValue(Instance i):Value*: Retorna o *Value* do atributo para uma determinada instância recebida no parâmetro *i*

?? *getType():String*: Retorna o tipo do atributo.

?? *getName():String*: Retorna o nome do atributo.

São métodos definidos na interface *IInstances*:

?? *loadDataset(String s)*: O parâmetro *s* informa o caminho para o banco de dados a ser minerado. O método deve ler os dados do banco de dados no caminho especificado e gerar as coleções de instâncias e atributos com seus respectivos valores.

?? *getAttribute(int index): IAttribute*: Retorna o atributo cujo índice é igual ao especificado no parâmetro *index*.

?? *getInstance(int index): IInstance*: Retorna a instância localizada na posição da coleção indicada pelo parâmetro *index.*

?? *setValue(Value a)*: permite a inserção de um valor ao conjunto de instâncias. O parâmetro *a* deve conter além de seu valor as referências para os objetos *:Attribute* e *:Instance* já ajustados.

?? *getValue(IInstance i, IAttribute a): Value*: retorna o *Value* referenciado por *i* e *a.*

?? *setClassAttribute(int index)*: Determina o atributo de classificação como sendo o parâmetro *index.*

?? *getClassIndex(): int*: Retorna o índice do atributo de classificação.

?? *numClass():int*: Retorna o número de valores diferentes encontrados no atributo cujo índice é igual ao atributo de classificação.

?? *insertAttribute(IAttribute a)*: Adiciona o atributo *a* na coleção de atributos.

?? *insertInstance(IInstance ia)*: Adiciona a instância *i* à coleção de instâncias.

?? *randomize()*: Ordena aleatoriamente a coleção de instâncias.

4.3.2 A Camada de Aplicação

A camada de *Aplicação* consiste da implementação dos dois algoritmos propostos no capítulo 3 desta dissertação, juntamente com os componentes que implementam técnicas e algoritmos necessários para suas execuções.

Visando diminuir o tempo de desenvolvimento do software, a implementação do framework foi acoplada a dois outros softwares que implementam técnicas especificas, como representado pela figura 4.5 que ilustra a camada de aplicação.

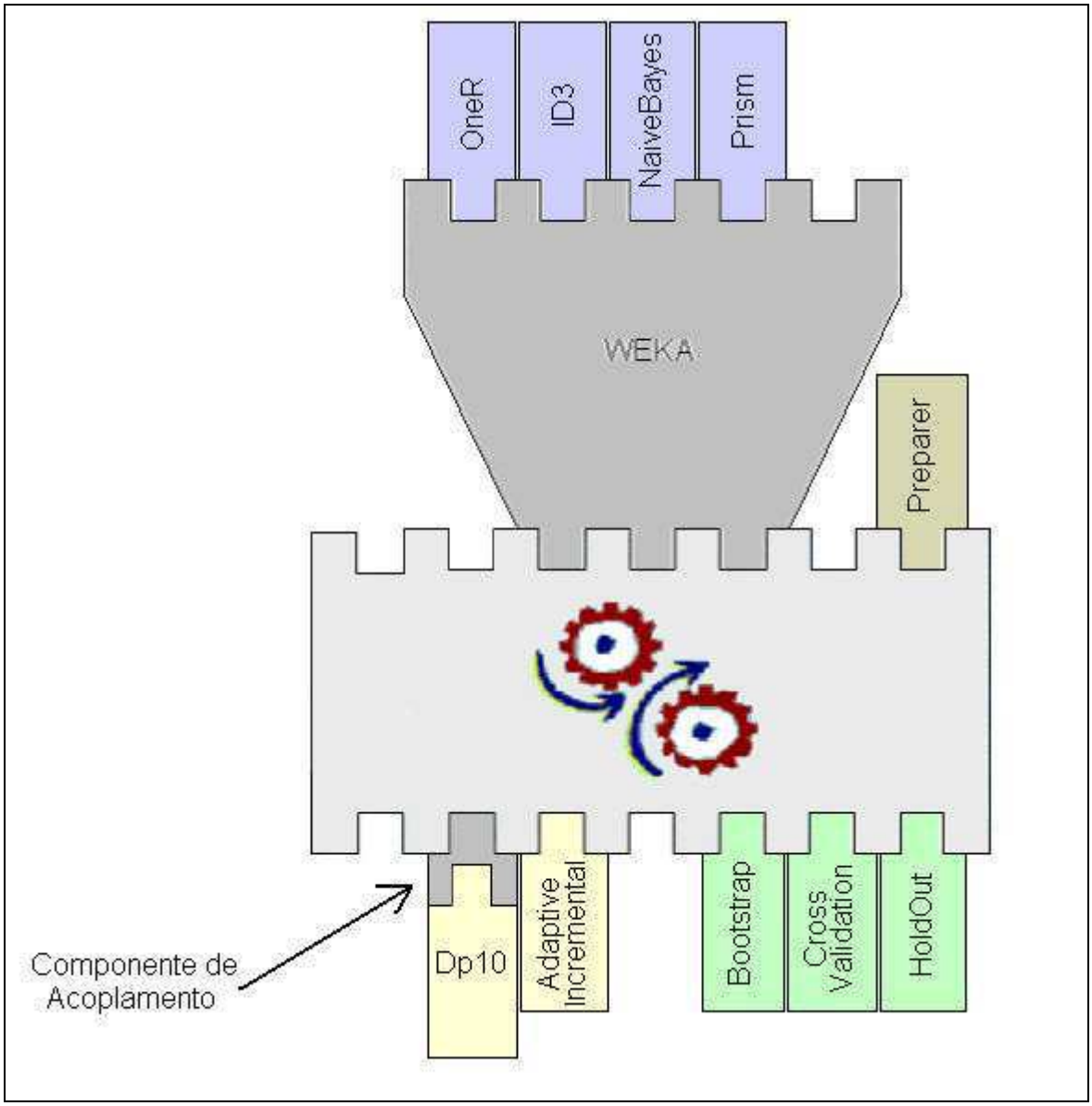

Figura 4.5: Arquitetura da Camada de Aplicação

O pacote[7] WEKA (Waikato Enviroment for Knowledge Analisys) foi discutido no capítulo 2 desta dissertação, e é por si só um framework para MD. WEKA agrega uma variedade de algoritmos de indução de conhecimento que podem ser utilizados em

[7] Um pacote é simplesmente um diretório contendo uma coleção de classes relacionadas.

nosso framework. Dentre os algoritmos que ele implementa estão o *ID3*, *Prism*, *OneR* e *Naïve Bayes*.

Dp10 é um software que implementa a técnica de amostragem denominada *Convergência*. Com o acoplamento dessa ferramenta ao framework, o tempo de implementação do software é diminuído. Torna-se necessário, no entanto, a construção de uma classe que permita o acoplamento entre a ferramenta e o framework, já que *Dp10* é uma ferramenta implementada na linguagem *C* e a implementação do framework é feita em JAVA.

Um método da classe de acoplamento projetada alimenta os arquivos de entrada do software *Dp10* com os parâmetros necessários e executa uma versão compilada da ferramenta. As saídas de *Dp10* são armazenadas em arquivos texto. Esses arquivos são lidos por um método da classe e posteriormente repassados ao framework.

4.3.2.1 Análise da Camada de Aplicação

A figura 4.6 descreve a parte central da camada de aplicação, usando a notação UML. O núcleo do framework é a classe abstrata *Classifier_Inducer*. As classes *Naïve_Classifier_Inducer* e *Expert_Classifier_Inducer* herdam da classe *Classifier_Inducer* e redefinem o método *runProcess()*. Os métodos *runProcess()* implementam os algoritmos definidos no capítulo anterior, conforme especifica o requisito funcional RF1. *Classifier_Inducer* implementa a interface *IClassifier_Inducer*, a qual define os métodos básicos de uma classe que representa um processo de MD deve implementar.

A classe *Classifier_Inducer* agrega três coleções. A primeira coleção é composta por objetos de classes que implementam a interface *IInducer*. A interface *IInducer* define o acoplamento entre os objetos instanciados a partir de classes que implementam algoritmos de indução de conhecimento e os demais objetos do sistema. A segunda coleção é composta por objetos de classes que implementam *ISampleTechnique*. A interface *ISampleTechnique* define o acoplamento entre objetos instanciados a partir de classes que implementam técnicas de amostragem e os demais objetos do sistema. A terceira coleção é composta por objetos instanciados a partir de classes que implementam *ISplittingTechnique*. A interface *ISplittingTechnique* define o

acoplamento entre objetos instanciados a partir de classes que implementam técnicas de fragmentação de amostras e os demais objetos do sistema.

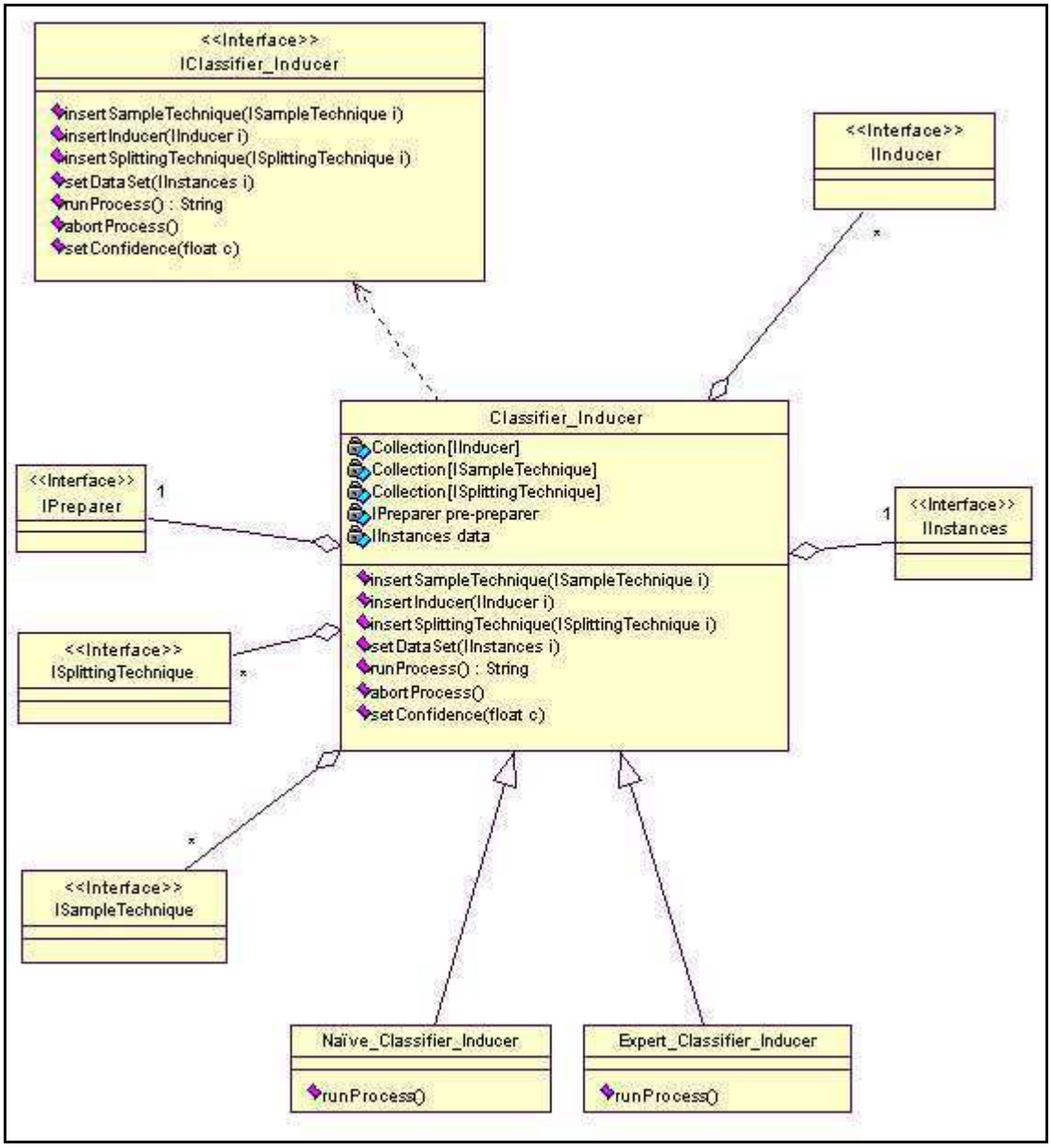

Figura 4.6: Diagrama de classes do núcleo do framework

Além das três coleções, compõem a classe *Classifier_Inducer* um objeto instanciado de uma classe que implementa *IInstances* e um objeto instanciado a partir de uma classe que implementa *IPrepare*. Os objetos instanciados a partir de classes que implementam a interface *IInstances* representam o banco de dados a ser minerado. A

interface *IPrepare* define o acoplamento entre os objetos instanciados a partir de classes que implementam técnicas de preparação de amostras e os demais objetos. As classes que implementam a interface *IPrepare* agregam técnicas para tratamento de dados desconhecidos e algoritmos para tornar discretos domínios de variáveis contínuos.

Os métodos presentes na interface *IClassifier_Inducer*, implementados pela classe abstrata *Classifier_Inducer* foram especificados levando em consideração os casos de uso do Minerador detalhados na seção 4.2 deste capítulo. Para cada caso de uso, um método foi especificado.

A seguir cada um dos métodos especificados em *IClassifier_Inducer* é descrito brevemente.

?? *insertSampleTechnique(ISampleTechnique i)*: adiciona uma técnica de amostragem *i* que é recebida através do parâmetro à coleção de técnicas de amostragem que será utilizada na execução do processo. Esse método implementa o caso de uso CU3.

?? *insertSplittingTechnique(ISplittingTechnique i)*: recebe como parâmetro um objeto de uma classe que implementa *ISplittingTechnique* e adiciona esse objeto na coleção de técnicas de fragmentação que serão utilizadas pelo processo. Esse método implementa o caso de uso CU4.

?? *insertInducer(IInducer i)*: esse método recebe como parâmetro um objeto instanciado a partir de uma classe que implementa *IInducer* e o adiciona na coleção de algoritmos de indução para que seja utilizado durante a execução do processo. Esse método implementa o caso de uso CU5.

?? *setDataSet(IInstances i)*: recebe como parâmetro um banco de dados *i* classificado para ser minerado. Esse método implementa o caso de uso CU6.

?? *setConfidence(float c)*: esse método recebe o parâmetro *c* que contém a confiança desejada sobre a acurácia estimada para o conjunto de execução. Esse método implementa o caso de uso CU7.

?? *runProcess(): String*: implementa um algoritmo que contempla o processo MD. Retorna uma *string* contendo o melhor classificador encontrado para o banco de dados, juntamente com sua acurácia estimada

para o conjunto de execução. Esse método implementa o caso de uso CU8.

?? *abortProcess()*: encerra a execução do processo liberando a memória alocada. Esse método implementa o caso de uso CU9.

?? *getConfusionMatrix()*: *String*: retorna uma *string* contendo a matriz de confusão [Witten and Eibe, 1999] para o melhor classificador induzido. A matriz de confusão demonstra a acurácia do classificador para cada uma das classes em separado.

A figura 4.7 ilustra o grupo de classes que representam os algoritmos de indução de conhecimento.

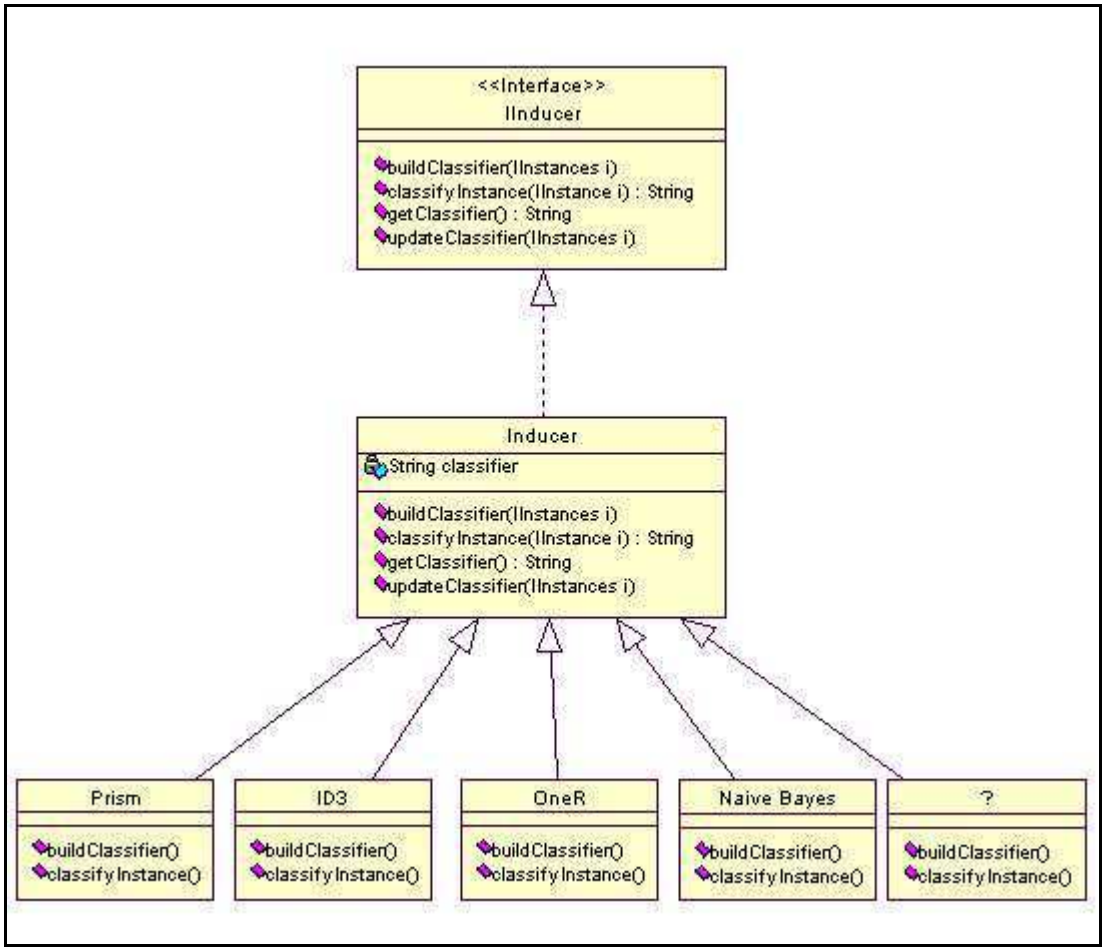

Figura 4.7: Diagrama de classes dos algoritmos de indução de conhecimento

A interface *IInducer* estabelece o acoplamento entre os objetos instanciados a partir de classes que implementam algoritmos de indução e demais os objetos. *Inducer* é

uma classe abstrata que implementa a interface *IInducer*. O atributo *classifier* da classe *Inducer* é responsável por armazenar o classificador induzido pelo algoritmo na forma de uma *String*. Todos os algoritmos de indução de conhecimento acoplados ao framework precisam necessariamente herdar da classe *Inducer* e redefinir apenas dois métodos: *buildClassifier()* e *classifyInstance()*.

No diagrama apresentado na figura 4.7, as classes que implementam os algoritmos de indução *ID3*, *OneR*, *Naïve Bayes* e *Prism* herdam de *Inducer* e redefinem os métodos necessários. A classes "?" indica a possibilidade de inclusão de novos algoritmos de indução ao framework pela simples adição de uma nova classe que herde de *Inducer* e redefine os métodos *buildClassifier()* e *classifyInstance()*.

Quatro são os métodos definidos na interface *IInducer*. A seguir é apresentado brevemente cada um desses métodos.

- *buildClassifier(IInstances i)*: Esse método recebe como parâmetro um objeto de uma classe que implementa *IInstances*, o qual representa um banco de dados classificado. Utilizando as características próprias do algoritmo de indução que a classe implementa, o método infere um classificador do banco de dados.
- *classifyInstance(Instance i)*: *String*: Com a instância *i* recebida como parâmetro, o método verifica junto ao classificador inferido a classificação indicada para a instância e a retorna na forma de uma *string*.
- *updateClassifier(IInstances i)*: Dado um banco de dados classificado *i* que é recebido pelo parâmetro, o método deve inferir um novo classificador e substituir o classificador corrente.
- *getClassifier(): String*: Retorna o classificador inferido na forma de uma *string*.

A figura 4.8 ilustra o grupo de classes que representam as técnicas de fragmentação de amostras. A interface *ISplittingTechnique* define o acoplamento entre os objetos instanciados a partir de classes que implementam esta interface e os demais objetos.

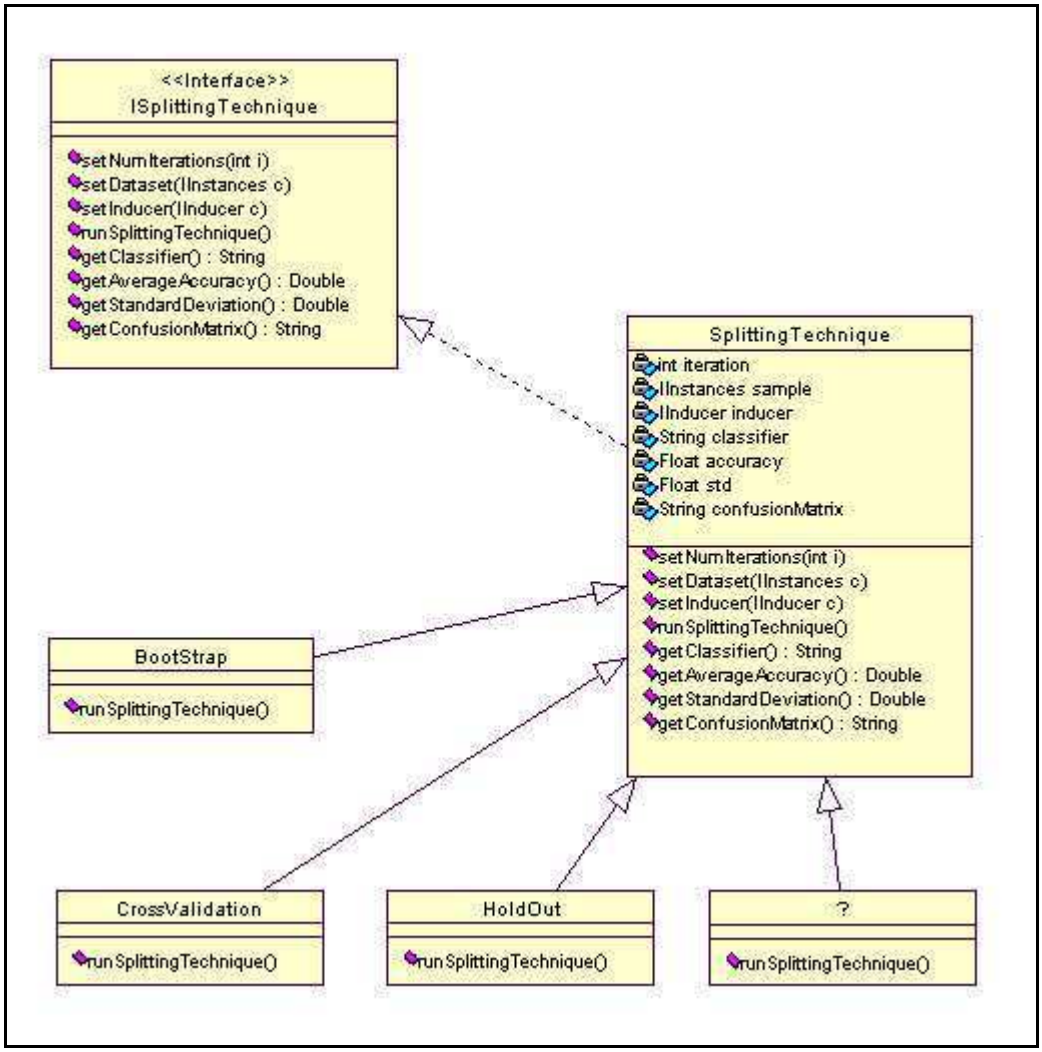

Figura 4.8: Diagrama de classes das técnicas de fragmentação de amostras

A classe *SplittingTechnique* é uma classe abstrata que implementa a interface *ISplittingTechnique.* Seus atributos são: *iteration* que é responsável por armazenar o número de iterações da técnica ? para *K-fold Cross Validation* indica o *k*, para *Bootstrap* indica o *b*, por exemplo, veja a seção 2.3 do capítulo 2. ? , *sample* é amostra que será fragmentada, *inducer* é o algoritmo de indução ao qual os fragmentos gerados serão submetidos, *classifier* armazena o classificador induzido em uma das iterações da técnica, *accuracy* é a acurácia média obtida nas iterações da técnica, *std* registra o desvio padrão da média, e *confusionMatrix* guarda a matriz de confusão do classificador.

As classes que implementam técnicas de fragmentação de amostras *K-fold Cross Validation*, *Bootstrap* e *Holdout* herdam da classe *SplittingTechnique* e redefinem o método *runSplittingTechnique.* A classe rotulada por "?" expressa a possibilidade de inclusão de novas técnicas de fragmentação ao framework pela simples adição de uma classe que herde da classe abstrata S*plittingTechnique* e redefina o método *runSplittingTechnique.*

A seguir é descrito brevemente cada um dos métodos especificados na interface *ISplittingTechnique*.

- *setNumIterations(int i)*: Ajusta o número de iterações da técnica segundo o parâmetro *i*.
- *setDataset(IInstances c)*: Esse método ajusta o parâmetro *c* como a amostra dos dados a ser fragmentada.
- *setInducer(IInducer c)*: Ajusta o algoritmo de indução ao qual os fragmentos da amostra serão submetidos.
- *runSplittingTechnique()*: Utilizando as características próprias da técnica de fragmentação que a classe implementa, o método fragmenta a mostra segundo o número de iterações definido. Com o algoritmo de indução ele infere classificadores a partir dos conjuntos de treinamento e os testa com os respectivos conjuntos de teste. No fim da execução do método, os atributos *classifier*, *accuracy, std* e *confusionMatrix* são ajustados.
- *getClassifier(): String*: Retorna o classificador inferido na forma de uma *string*
- *getAverageAcurracy(): Float*: Retorna a media das acurácia medidas na forma de um *float*.
- *getStandardDeviation(): Float*: Retorna o desvio padrão da média na forma de um *float*.
- *getConfusionMatrix(): String*: Retorna a matriz de confusão do classificador na forma de uma *string*.

A figura 4.9 apresenta o diagrama das classes que implementam as técnicas de amostragem. Nesse diagrama, a interface *ISampleTechnique* define o acoplamento entre os objetos instanciados a partir de classes que implementam técnicas de amostragem e

os demais objetos do sistema. *SampleTechnique* é uma classe abstrata que implementa a interface *ISampleTechnique*. São atributos da classe *SampleTechnique*: *dataset* que armazena o banco de dados do qual será retirada a amostra, *inducer* que registra o algoritmo de inferência ao qual a amostra será submetida após ser fragmentada e *sample* que armazena a amostra retirada.

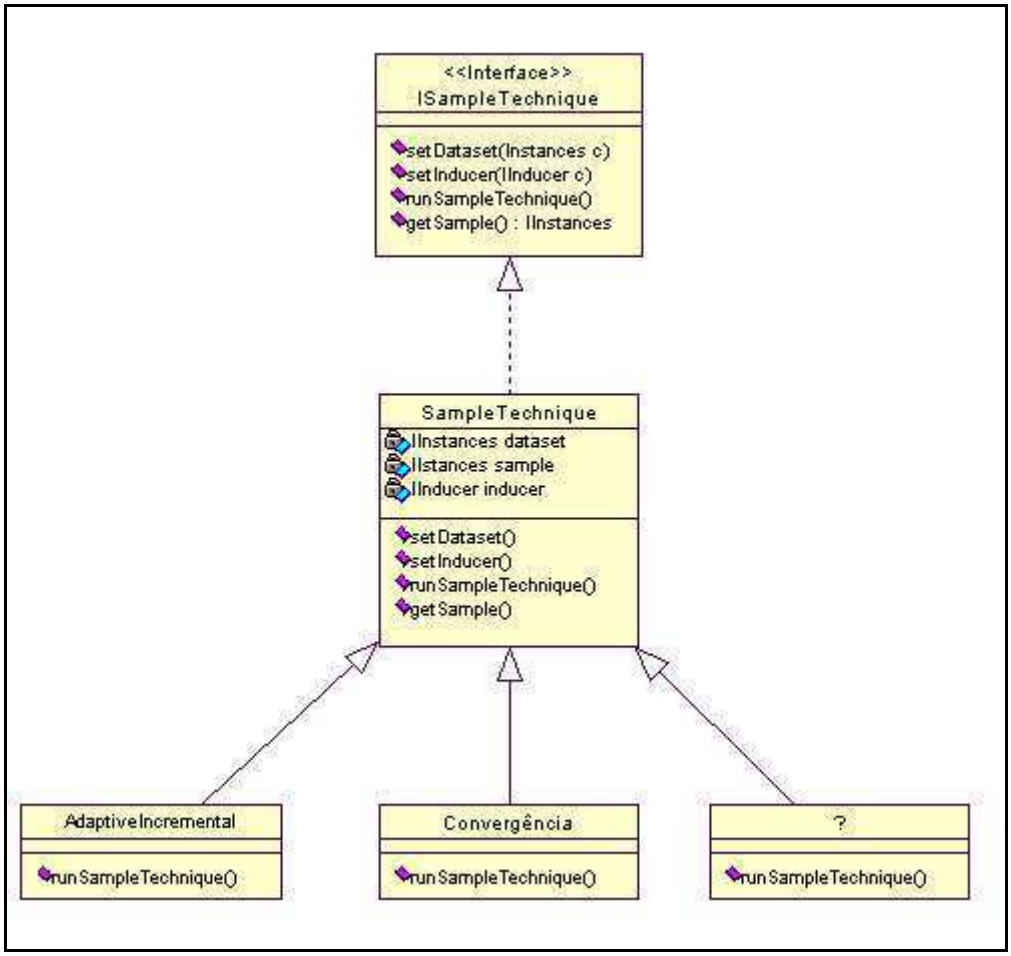

Figura 4.9: Diagrama de classes das técnicas de amostragem

Todas as classes que implementam técnicas de amostragem inseridas no framework herdam da classe abstrata *SampleTechnique* e redefinem o método *runSampleTechnique()*. No diagrama estão representadas duas técnicas de amostragem: *Adaptive Incremental Framework* e *Convergência*. A classe denominada "?" representa a possibilidade de inserção de novas técnicas de amostragem ao framework, pela simples adição de uma classe herdando de *SampleTechnique* e redefinindo o método *runSampleTechnique()* a sua maneira.

A interface *ISampleTechnique* especifica quatro métodos, que são os seguintes:

- ?? *setDataset(IInstances c)*: Esse método define o parâmetro *c* como o banco de dados a ser extraída a amostra.
- ?? *setInducer(IInducer c)*: Define o algoritmo de indução ao qual a amostra será submetida após ser retirada.
- ?? *runSampleTechnique()*: Executa a técnica de amostragem sobre o banco de dados. Cada classe que utiliza características próprias e específicas para isto. No final da execução deste método, o atributo *sample* é ajustado com a amostra.
- ?? *getSample(): IInstances*: Retorna a amostra do banco de dados retirada com a técnica que a classe implementa.

O diagrama de classes apresentado na figura 4.10 ilustra as classes que implementam as técnicas de preparação de dados. A interface *IPrepare*, apresentada nesse diagrama, define o acoplamento entre os objetos instanciados a partir de classes que implementam as técnicas de preparação e os demais objetos do sistema. A classe denominada "?" representa a possibilidade de uma nova classe de preparação de dados ser acoplada ao framework, pela simples adição dessa classe implementando a interface *IPreprare* e redefinindo o método *preprareDataset()* a sua maneira. A classe *Pre-Prepare* implementa a interface *IPrepare* e seu único atributo é *dataset*, que é utilizado para armazenar a amostra do banco de dados a ser preparada.

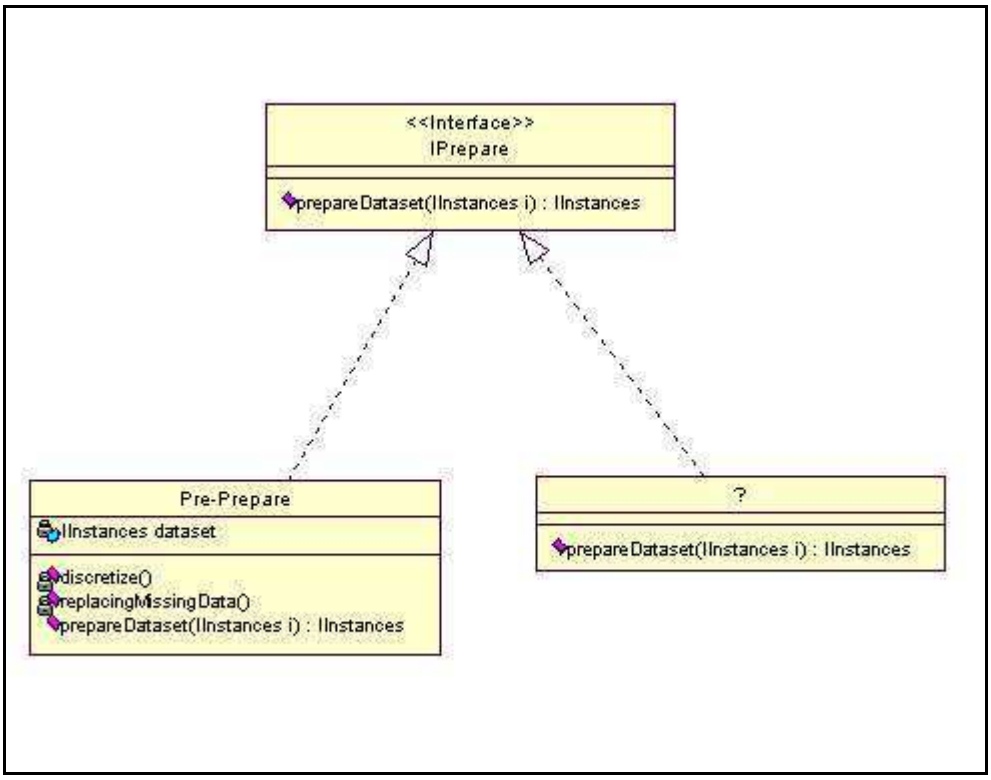

Figura 4.10: Diagrama de classes das técnicas de preparação de dados

Três métodos estão presentes em *Pre-Prepare:*

?? *discretize()*: A função desse método é verificar a necessidade de tornar discretos os domínios das variáveis presentes no banco de dados. Caso necessário ele aplica um algoritmo para isto. Esse método é privado, isto é, só pode ser invocado por outros métodos que sejam internos à classe que o implementa.

?? *replacingMissingData()*: Verifica a existência de dados desconhecidos ou inexistentes no banco de dados. Caso existam, ele aplica uma técnica para tratamento desses dados. A técnica consiste em remover as instâncias que apresentam esses problemas. Esse também é um método privado à classe.

?? *prepareDataset(IInstances i):IInstances*: Esse método é definido pela interface *IPrepare*. Ele recebe como parâmetro um banco de dados classificado para ser preparado e em seguida aplica as técnicas de preparação de amostras, para isso ele invoca os outros dois métodos

definidos anteriormente: *replacingMissingData()* e *discretize()*. Por fim retorna o banco de dados preparado.

4.3.3 A Camada de Apresentação

A camada de *Apresentação* consiste em uma interface gráfica de comunicação com o framework. Nela todos os casos de uso definidos para o Minerador são contemplados e a utilização do framework para inferência de conhecimento é facilitada. A figura 4.11 apresenta a interface gráfica construída.

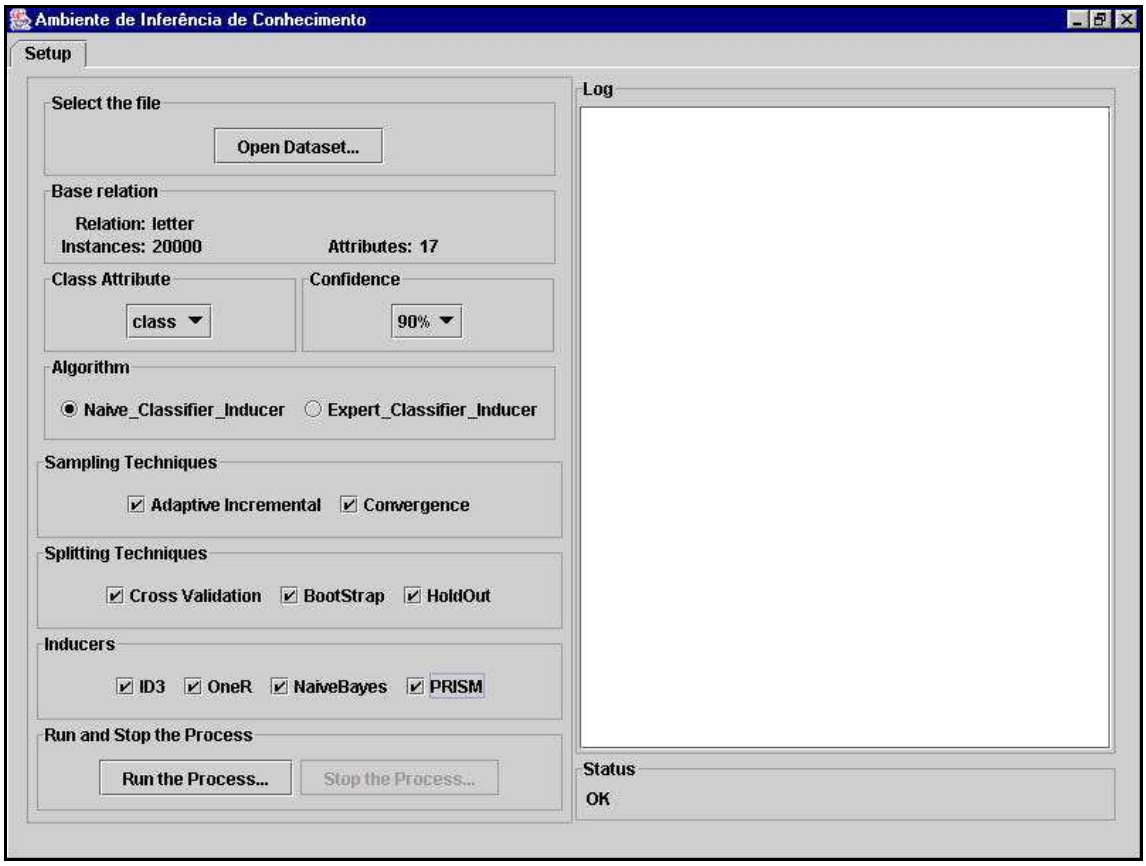

Figura 4.11: Interface Gráfica do Framework

4.4 Considerações Finais

Este capítulo apresenta a análise e projeto de um framework que implementa os algoritmos *Expert_Classifier_Inducer* e *Naïve_Classifier_Inducer*. Além destes algoritmos, o framework implementa diversas técnicas de amostragem, diversas técnicas de fragmentação de amostras e diversos algoritmos de indução de conhecimento, bem como um componente de preparação de dados para mineração.

No próximo capítulo, uma avaliação experimental é descrita. A avaliação experimental visa avaliar os algoritmos *Naïve_Classifier_Inducer* e *Expert_Classifier_Inducer* na inferência do melhor conjunto de técnicas, em uma variedade de problemas de MD diferentes.

Capítulo V

Avaliação Experimental

Para um julgamento mais amplo a respeito da qualidade dos resultados gerados pelo software, este capítulo foi dividido em três partes. A primeira avalia o algoritmo *Naïve_Classifier_Inducer*. A segunda parte discute a observação das heurísticas que permitiram a construção do algoritmo *Expert_Classifier_Inducer*. A terceira parte visa comprovar que o algoritmo *Expert_Classifier_Inducer* consegue resultados satisfatórios, isto é, induz classificadores confiáveis, com um custo menor de processamento que *Naïve_Classifier_Inducer*.

Para a execução de todos os experimentos foram selecionados quatros algoritmos de indução de conhecimento: *Naïve Bayes* (*Naïve*), *OneR*, *ID3* e *Prism*; duas técnicas de amostragem: *Convergência* (*Conv*) e *Adaptive Incremental Framework* (*AIF*) ? com acurácia mínima para seleção da amostra de 90% ? ; três técnicas de fragmentação de amostras: *HoldOut* (*HO*) ? dez iterações ? , *K-Fold Cross Validation* (*CV*) ? dez iterações ? e *Bootstrap* (*BS*) ? dez iterações ? . O componente de preparação de dados acoplado ao framework remove todas as instâncias que contêm dados desconhecidos ou inexistentes e para os atributos numéricos, torna seus domínios discretos em cinco faixas.

5.1 Experimentos com *Naïve_Classifier_Inducer*

Para avaliar o algoritmo *Naïve_Classifier_Inducer*, nós escolhemos nove bancos de dados de diversas áreas diferentes. Os bancos de dados escolhidos são: *Letter, Splice*, *Kr-vs-Kp*, *Mushroom*, *Soybean*, *Cars*, *Titanic*, *Connect-4* e *Cmc*. Todos os bancos de dados foram obtidos ao acaso, dentre os disponíveis no repositório UCI

[Blake and Merz, 2002]. A tabela 5.1 apresenta uma descrição dos bancos de dados utilizados.

BD	Descrição	Atributos	Instâncias	Classes
Letter	Caracteres manuscritos	17	20000	26
Splice	Seqüências de DNA	61	3100	3
Mushroom	Espécies de cogumelos	22	8124	2
Soybean	Cultivo de soja	35	683	19
Titanic	Naufrágio do transatlântico Titanic	4	2201	2
Connect-4	Partidas de Connect-4	43	36991	3
Cmc	Escolha do método anticoncepcional	10	1473	3
Kr-vs-Kp	Partidas de Xadrez	36	3196	2
Cars	Informações sobre automóveis	10	404	3

Tabela 5.1: Bancos de Dados para Experimentos com *Naïve_Classifier_Inducer*

A tabela 5.2 é uma síntese dos melhores classificadores inferidos pela execução do algoritmo *Naïve_Classifier_Inducer* sobre os bancos de dados mencionados, através do método *runProcess()* da interface *IClassifier_Inducer*, implementado pela classe *Naïve_Classifier_Inducer* ? capítulo 4. A coluna ? contém os valores das acurácias para os melhores classificadores inferidos, com o respectivo desvio padrão na coluna ?. A coluna $acc_e(90\%)$ contém os valores da acurácia estimada dos classificadores, calculada através da fórmula de Bernoulli, na seção 2.1 do capítulo 2.

Melhor classificador para	?	?	**acc_e (90%)**	**Acurácia máxima relatada por UCI**
Letter	77.20%	0.43%	75.77% ? 78.18%	80,00%
Splice	92.55%	1.27%	89.79% ? 93.75%	93,80%
Cars	99.39%	0.32%	94.40% ? 99.82%	Não relatada
Soybean	92.03%	1.50%	86.73% ? 95.28%	97,10%
Kr-vs-Kp	99.58%	0.08%	97.99% ? 99.50%	99,70%
Mushroom	100.00%	0.00%	99.79% ? 100.00%	99,41%
Titanic	98.18%	5.45%	97.26% ? 98.54%	Não relatada
Cmc	55.14%	0.84%	52.14% ? 57.82%	57,00
Connect-4	96.95%	0.98%	95.12% ? 96.72%	Não relatada

Tabela 5.2: Síntese dos Experimentos com *Naïve_Classifier_Inducer*

A última coluna da tabela apresenta a melhor acurácia relatada na literatura para os bancos de dados utilizados, segundo o repositório UCI. Essa informação não foi relatada para três dos nove bancos de dados utilizados.

Percebemos, comparando a coluna da acurácia média obtida pelo melhor classificador inferido por *Naive_Classifier_Inducer* e a coluna da melhor acurácia relatada na literatura, que os classificadores inferidos para *Letter, Splice, Soybean, Kr-vs-kp, Mushroom* e *CMC,* foram muito próximos dos melhores relatados na literatura. A diferença foi maior para *Soybean*, que se aproximou dos 5%. Para *Mushroom*, por outro lado, a acurácia obtida foi maior que a relatada. Com esses resultados, podemos concluir que os classificadores inferidos por *Naive_Classifier_Inducer* são bons para os problemas de MD apresentados.

Como podemos explicar que os classificadores inferidos para *Letter* e *CMC* tenham acurácia não superior a 80%?

Lembre-se que *Letter* é um banco de dados que contém informações sobre caracteres manuscritos. Caracteres diferentes manuscritos por uma mesma pessoa apresentam características semelhantes, por outro lado, caracteres iguais manuscritos por pessoas diferentes podem apresentar grandes diferenças, o que torna difícil à

indução de padrões genéricos para *Letter*. Essa dificuldade em gerar padrões genéricos fez com que os algoritmos de indução acoplados ao framework não obtivessem bons resultados.

Cmc apresenta repetidas instâncias com diferentes classes ? o que consiste em ruído. A tabela 5.3 mostra um fragmento de *Cmc*, com 4 instâncias. O atributo de classificação é *"Contraceptive Method Used*". As primeiras duas instâncias tem os mesmos valores para todos os atributos, exceto o atributo de classificação. O mesmo acontece com a 3ª e 4ª instâncias. Essa presença de instâncias idênticas em diferentes classes complica a tarefa dos algoritmos de indução.

	Wife age	Wife Education	Husband Education	Children	Religion	Working	Husband Occupation	Standard of Living	Media Exponsure	Contraceptive Method Used
1ª	31	2	2	4	1	1	3	3	0	**1**
2ª	31	2	2	4	1	1	3	3	0	**3**
3ª	44	4	4	5	1	0	1	4	0	**2**
4ª	44	4	4	5	1	0	1	4	0	**3**

Tabela 5.3: Um fragmento do banco de dados *Cmc* com ruído

A tabela 5.4 apresenta um comparativo dos ranking de classificadores inferidos com as execuções do algoritmo *Naïve_Classifier_Inducer*. Para *Letter*, a melhor combinação de técnicas foi: técnica de amostragem *AIF*, a técnica de fragmentação de amostras *BS* e o algoritmo de indução de conhecimento *Prism*, Enquanto que *Conv-CV-OneR* foi a melhor combinação de técnicas para *Cars*, e assim por diante.

Observando a tabela, fica clara a instabilidade do conhecimento inferido por combinações de técnicas diferentes para um mesmo banco de dado, assim como, a instabilidade do conhecimento inferido por uma mesma combinação de técnicas para bancos de dados diferentes. Por exemplo, a combinação *Prism – AIF – BS* que para o banco de dados *Letter* inferiu o melhor classificador, para *Mushroom* inferiu o pior.

BD / Técnicas	*Letter*	*Splice*	*Cars*	*Soybean*	*Kr-vs-Kp*	*Mushroom*	*Titanic*	*Cmc*	*Connect-4*
ID3 – AIF– CV	7º	2º	8º	15º	7º	13º	7º	19º	11º
ID3 – AIF – BS	3º	3º	18º	16º	8º	9º	11º	4º	6º
ID3 – AIF – HO	8º	5º	21º	18º	14º	21º	6º	23º	15º
ID3 – Conv – CV	14º	19º	4º	3º	**1º**	5º	8º	20º	19º
ID3 – Conv – BS	13º	16º	5º	5º	2º	4º	12º	5º	12º
ID3 – Conv – HO	14º	17º	6º	8º	3º	6º	9º	24º	20º
Naïve – AIF – CV	4º	**1º**	16º	11º	16º	24º	3º	7º	**1º**
Naïve – AIF – BS	2º	6º	12º	13º	15º	22º	2º	3º	8º
Naïve – AIF – HO	6º	4º	17º	17º	17º	23º	**1º**	8º	10º
Naïve – Conv – CV	10º	11º	11º	**1º**	9º	18º	5º	10º	16º
Naïve – Conv – BS	9º	10º	14º	2º	11º	17º	10º	6º	9º
Naïve – *Conv – HO*	11º	12º	15º	4º	12º	19º	4º	9º	17º
OneR – AIF – CV	20º	13º	23º	21º	19º	14º	15º	12º	2º
OneR – AIF – BS	19º	14º	22º	24º	24º	11º	17º	11º	4º
OneR – AIF – HO	21º	15º	24º	23º	23º	15º	13º	15º	3º
OneR – Conv – CV	22º	22º	**1º**	19º	20º	8º	16º	14º	5º
OneR – Conv – BS	23º	18º	2º	20º	22º	7º	17º	13º	7º
OneR – Conv – HO	24º	20º	3º	22º	21º	10º	14º	16º	14º
Prism – AIF – CV	5º	7º	10º	10º	10º	16º	21º	17º	23º
Prism – AIF – BS	**1º**	8º	19º	12º	13º	12º	23º	**1º**	17º
Prism – AIF – HO	15º	9º	20º	14º	18º	20º	19º	21º	21º
Prism – Conv – CV	16º	24h	7º	6º	4º	2º	22º	18º	24º
Prism – Conv – BS	12º	21º	13º	7º	5º	**1º**	24º	2º	13º
Prism – Conv – HO	18º	23º	9º	9º	6º	3º	20º	22º	22º

Tabela 5.4: Os ranking de classificadores

Nossos experimentos permitem concluir que o algoritmo *Naïve_Classifier_Inducer* é eficaz na inferência do 'melhor' classificador, no sentido matemático do termo. A tabela 5.4 confirma que diferentes técnicas de amostragem, diferentes técnicas de fragmentação e diferentes algoritmos de indução utilizados em cada etapa do processo de MD têm influência sobre as acurácias dos classificadores inferidos. Em resumo, uma técnica que é boa para um banco de dados, pode não ser boa para outro banco de dados, e vice-versa.

5.2 Observação das Heurísticas

A seção 3.3 do capítulo 3 apresentou três heurísticas que foram utilizadas para construção do algoritmo *Expert_Classifier_Inducer*. A observação das heurísticas foi feita com base nos ranking dos classificadores inferidos para os nove bancos de dados ilustrados na tabela 5.2.

A figura 5.1 apresenta o ranking dos classificadores inferidos para o banco de dados *Mushroom*, com a execução do algoritmo *Naïve_Classifier_Inducer*. Mais precisamente, a figura exibe, para o banco de dados *Mushroom*, as acurácias médias obtidas ? menos os desvios padrões ? para cada combinação de técnicas de amostragem, de fragmentação e de indução. A acurácia média subtraída o desvio padrão, expressa a acurácia mais pessimista.

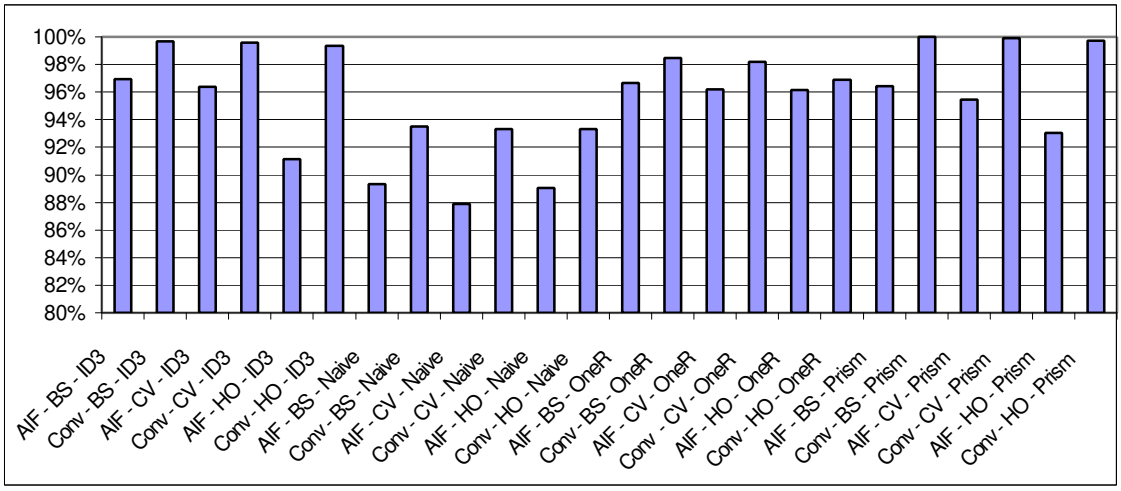

Figura 5.1: Ranking dos classificadores inferidos para o banco de dados *Mushroom*

A combinação de técnicas que inferiu o melhor classificador para o banco de dados *Mushroom* é *Convergência* (*Conv*), *BootStrap* (*BS*) e *Prism.* Note que, *Conv* combinado com *BS* e *Prism* apresenta resultados superiores a combinação de *Adaptive Incremental Framework* (*AIF*) com as mesmas técnicas de fragmentação e indução ? *Bootstrap* e *ID3*. Esta vantagem de *Conv* sobre *AIF* pode ser verificada também nas combinações com as técnicas *K-fold Cross Validation* (*CV*), de fragmentação, e *ID3*, de indução. De fato, como a figura evidencia, a técnica *Conv* é superior a *AIF* para todas as combinações possíveis de técnicas de fragmentação e indução, para *Mushroom*. Portanto, para a descoberta da melhor técnica de amostragem para o banco de dados *Mushroom*, é necessário apenas descobrir qual a melhor técnica de amostragem em uma das combinações possíveis.

Esse mesmo raciocínio pode ser aplicado a técnica de fragmentação e ao algoritmo de indução. Veja, por exemplo, que a técnica de fragmentação *BS* combinada a *Conv* e *Naïve Bayes* (*Naïve*) é melhor que *Holdout (HO)* na mesma combinação. E essa vantagem de *BS* sobre *HO* pode ser verificada nas outras combinações, em que *Conv* esteja presente.

Para o banco de dados *Titanic*, a melhor combinação de técnicas é *AIF*, *HO*, *Naïve*. *AIF* vence *Conv* associado à *HO* e *Naïve*, além disso, *AIF* também vence *Conv* em qualquer outra combinação de técnicas. Observe na figura 5.2. Por outro lado, *Holdout* vence *CV* e *BS* associado a *AIF* e a *Naïve Bayes*. Além disso, *Holdout* também vence *CV* e *BS* em qualquer outra combinação em que *AIF* esteja presente. *Naïve Bayes* vence os outros algoritmos de indução combinada a *Holdout* e *AIF*.

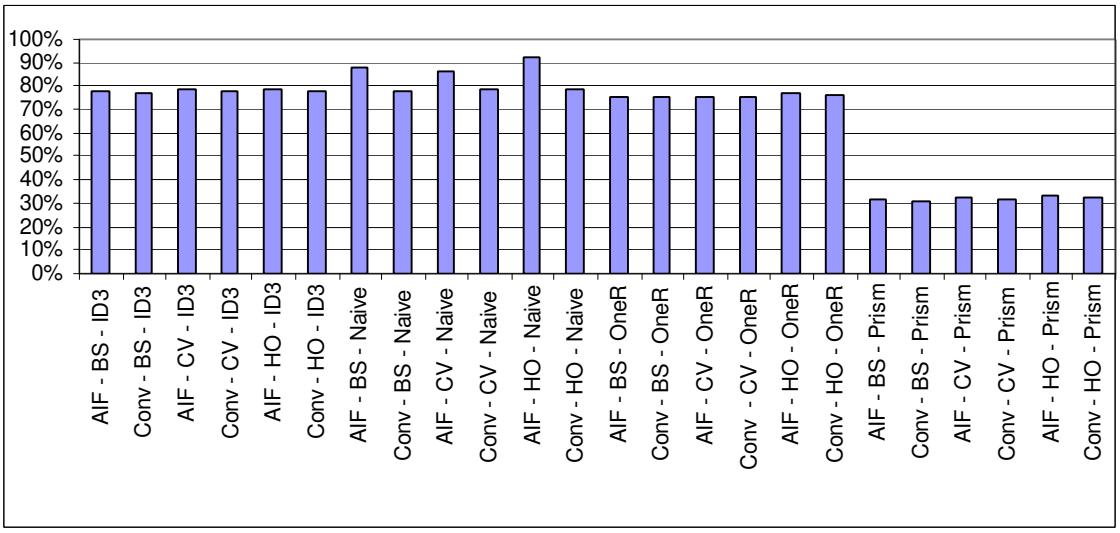

Figura 5.2: Ranking dos classificadores inferidos para o banco de dados *Titanic*

A figura 5.3 ilustra o ranking de classificadores para o banco de dados *Letter*. A melhor combinação de técnicas para *Letter* é: a técnica de amostragem *AIF*, a técnica de fragmentação *Bootstrap* e o algoritmo de indução *Prism*. Observe que *AIF* é melhor que *Convergência* em qualquer uma das combinações de técnicas. *Bootstrap* é melhor que *CV* e *HO*, em qualquer uma das combinações em que *AIF* esteja presente. *Prism* é o melhor que os outros algoritmos de indução, combinado a *AIF* e *Bootstrap*.

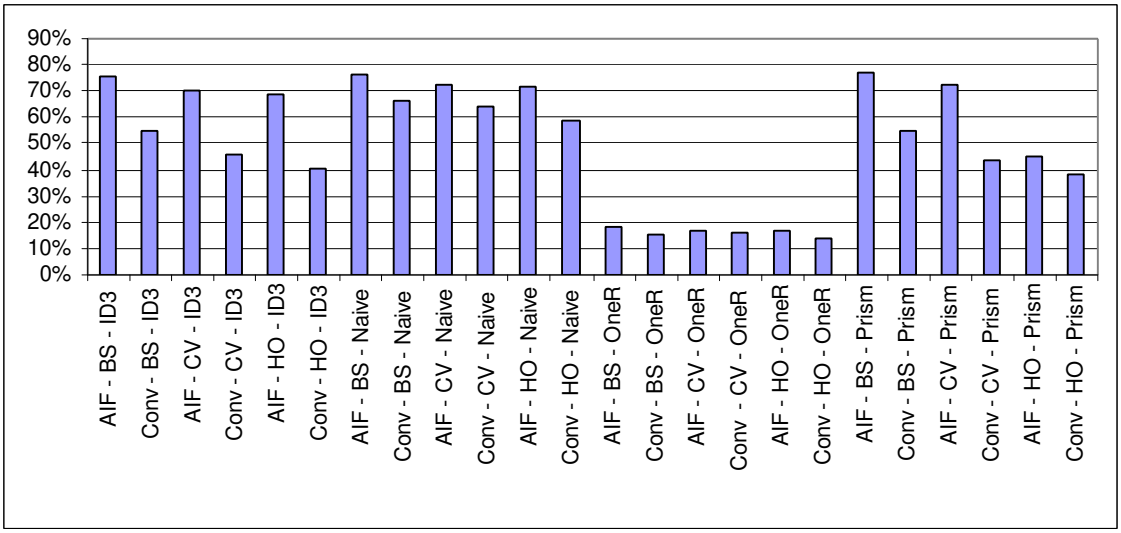

Figura 5.3: Ranking dos classificadores inferidos para o banco de dados *Letter*

Para o banco de dados *Splice*, a melhor combinação de técnicas é: a técnica de amostragem *AIF*, a técnica de fragmentação *CV* e o algoritmo de indução *Naïve Bayes* ? figura 5.4. *AIF* é melhor que *Conv* em qualquer combinação de técnicas. *CV* é melhor que as outras técnicas de fragmentação em qualquer combinação de técnicas em que *AIF* esteja presente. *Naïve* é o melhor que os outros algoritmos de indução, combinado a *AIF e CV*.

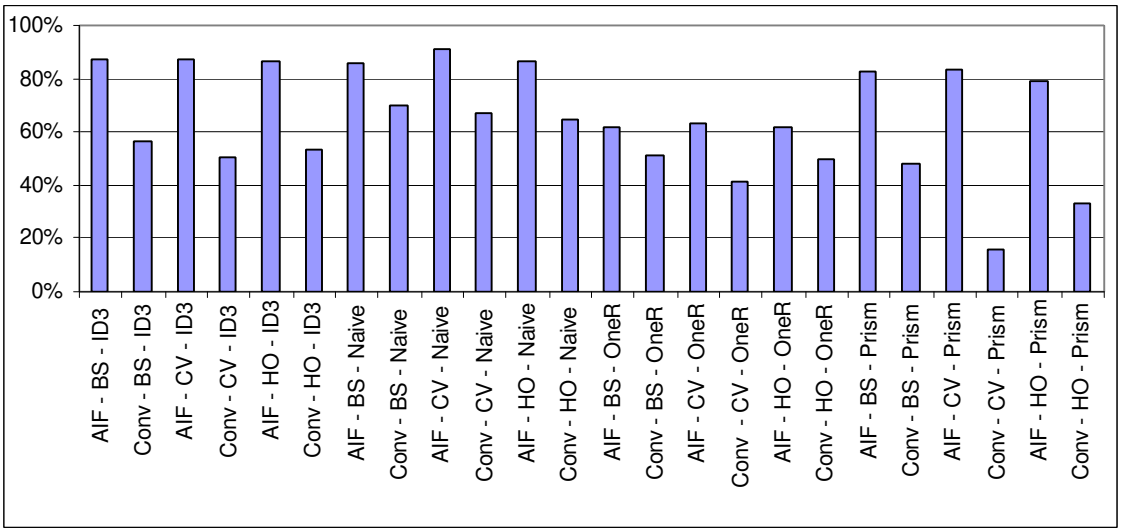

Figura 5.4: Ranking dos classificadores inferidos para o banco de dados *Splice*

Para o banco de dados *Kr-vs-Kp*, veja figura 3.8, a melhor combinação de técnicas é: a técnica de amostragem *Convergência*, a técnica de fragmentação *CV* e o algoritmo de indução *ID3*. *Convergência* é melhor que *AIF* em qualquer uma das combinações, exceto a combinação com *CV* e *OneR*. *CV* é melhor que as outras técnicas de fragmentação em todas as combinações de técnicas em que *Convergência* esteja presente. *ID3* é melhor que os outros algoritmos de indução, para esse banco de dados, combinado a *Conv* e *CV*.

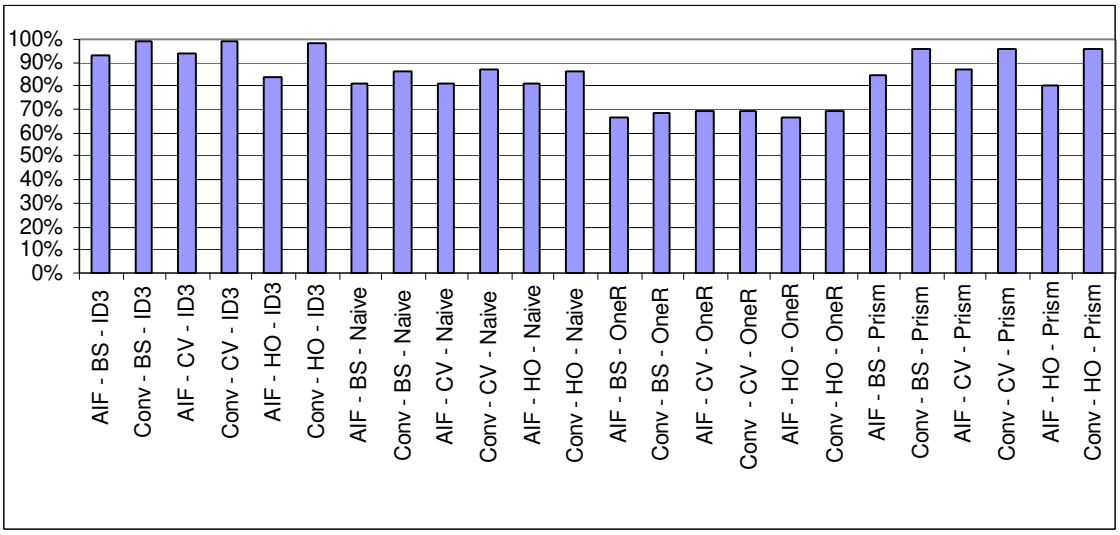

Figura 5.5: Ranking dos classificadores inferidos para o banco de dados *Kr-vs-Kp*

Para o banco de dados *Soybean*, figura 5.6, a melhor combinação de técnicas é a técnica de amostragem *Convergência*, a técnica de fragmentação *CV* e o algoritmo de indução *Naïve Bayes*. A técnica de amostragem *Convergência* é melhor que as outras técnicas de amostragem para qualquer uma das combinações possíveis. *CV* é melhor que as outras técnicas de fragmentação em qualquer uma das combinações em que *Conv* esteja presente. *Naïve Bayes* é o melhor algoritmo de indução, combinado a *Conv* e *CV*.

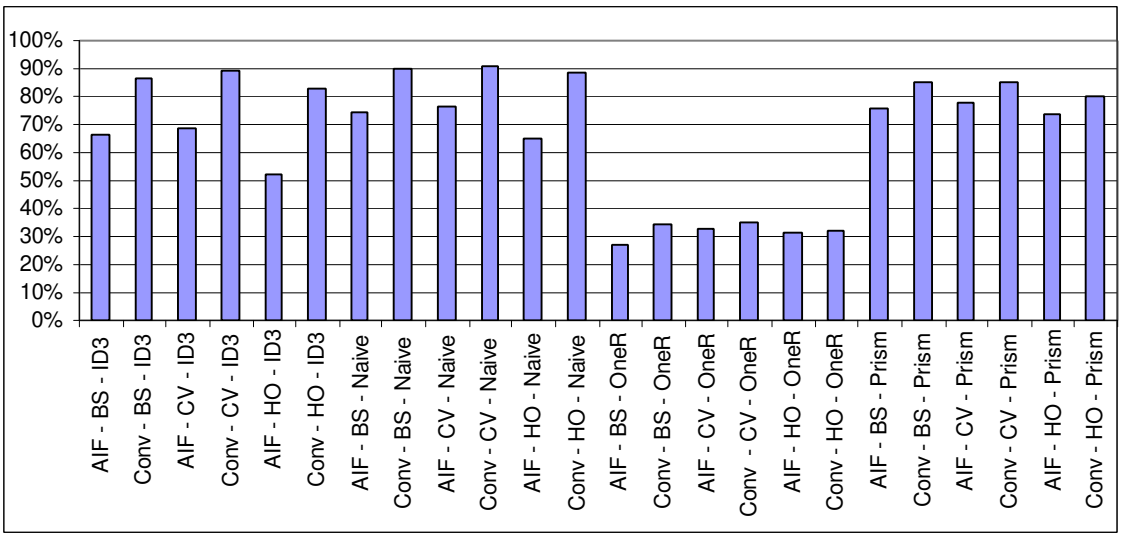

Figura 5.6: Ranking dos classificadores inferidos para o banco de dados *Soybean*

A figura 5.7 ilustra o ranking de classificares para o banco de dados *Connect-4*. A melhor combinação de técnicas para *Connect-4* é a técnica de amostragem *AIF*, a técnica de fragmentação *CV* e o algoritmo de indução *Naïve Bayes*. *AIF* é melhor que *Conv* em todas as possíveis combinações de técnicas, exceto a combinação com *BS* e *Prism*. *CV* é a melhor técnica de fragmentação em todas as combinações em que *AIF* esteja presente, exceto duas combinações: *AIF* com *ID3* e *AIF* com *Prism*. O algoritmo de indução *Naïve Bayes* é melhor que os outros algoritmos de indução, combinado a *AIF* e *CV*.

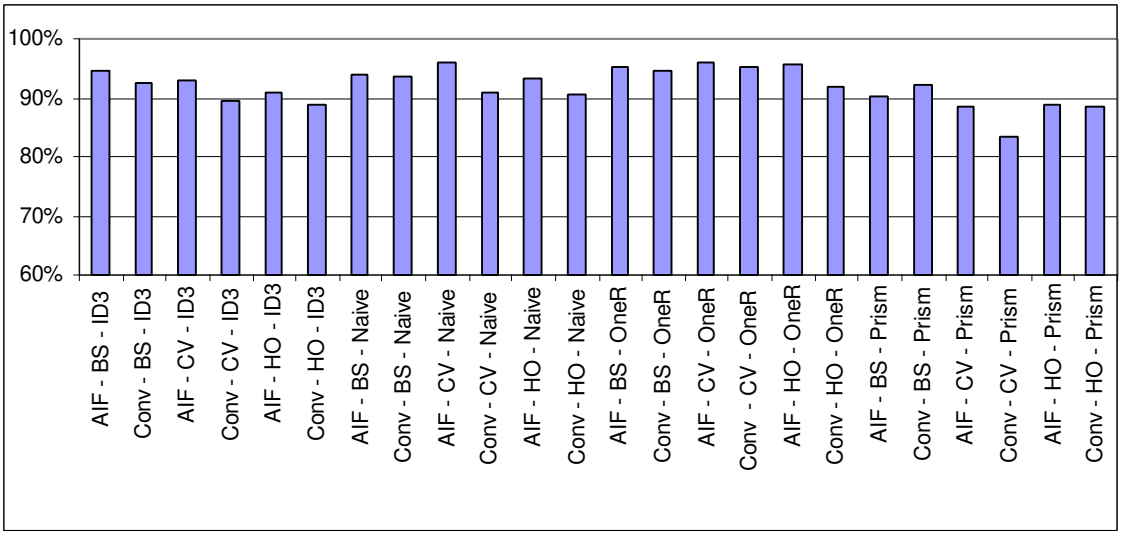

Figura 5.7: Ranking dos classificadores inferidos para o banco de dados *Connect-4*

Para o banco de dados *Cmc*, figura 5.8, o melhor conjunto de técnicas é: *AIF*, *Bootstrap* e *Prism*. *AIF* é melhor que *Conv*, para esse banco de dados, em todas as combinações de técnicas. *Bootstrap* é melhor que as outras técnicas de fragmentação em qualquer umas das possíveis combinações em que *AIF* esteja presente. *Prism* é melhor que os outros algoritmos de indução, combinado a *AIF* e *Bootstrap*.

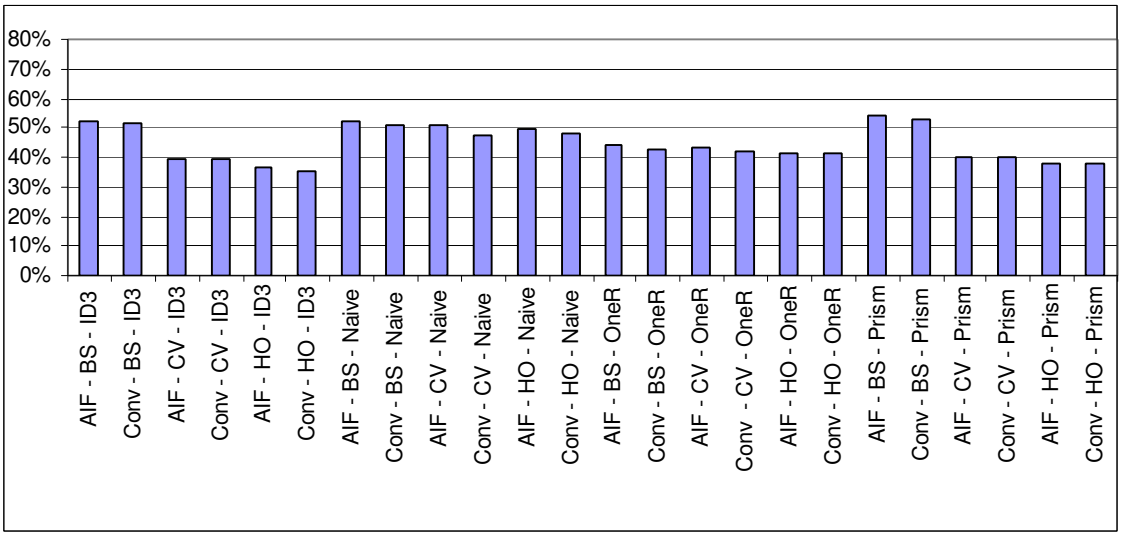

Figura 5.8: Ranking dos classificadores inferidos para o banco de dados *Cmc*

A figura 5.9 apresenta o ranking dos classificadores para o banco de dados *Cars.* A melhor combinação de técnicas para esse banco de dados é *Conv – CV – OneR*. *Conv* é melhor que *AIF* em qualquer combinação de técnicas, exceto a combinação com *BS* e *Naïve Bayes*. *CV* é a melhor técnica de fragmentação em qualquer uma das combinações possíveis em que *Conv* está presente. *OneR* é o melhor algoritmo de indução combinado a *CV* e *Conv*.

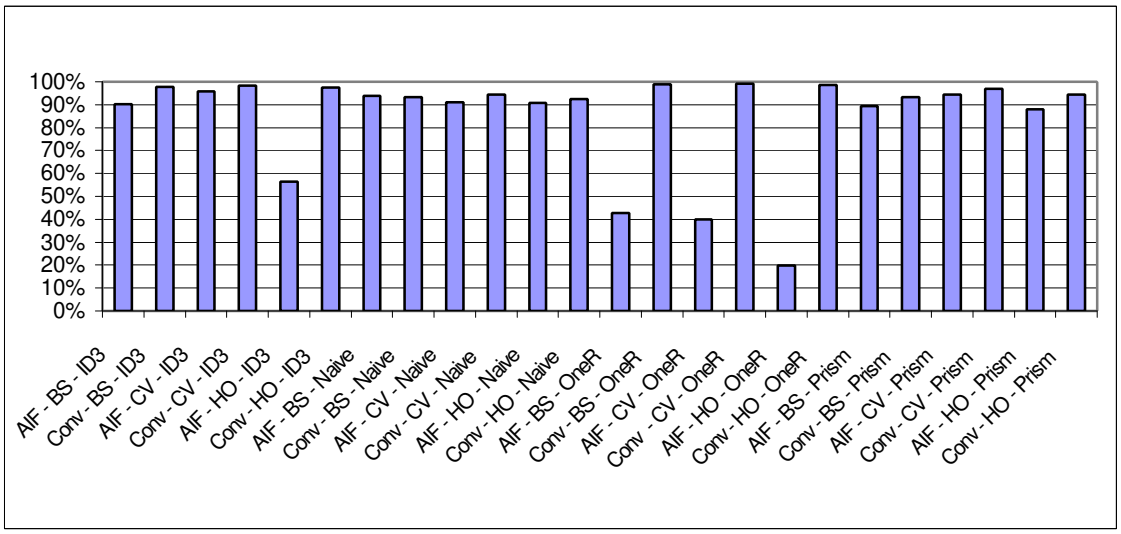

Figura 5.9: Ranking dos classificadores inferidos para o banco de dados *Cars*

Generalizando o que foi constatado para cada um dos ranking de classificadores, pode-se dizer que:

1. Se a técnica de amostragem *A1* tem resultados melhores que as outras técnicas de amostragem para o banco de dados *X* em uma certa combinação de técnicas de fragmentação e indução, então a vantagem de *A1* sobre as outras técnicas de amostragem pode ser assumida para *X*, independentemente das outras combinações das demais técnicas de fragmentação e indução.

2. Considerando a melhor técnica de amostragem (Heurística 1). Se a técnica de fragmentação *F1* tem resultados melhores que as outras técnicas de fragmentação para o banco de dados *X* em uma combinação em que esteja presente a melhor técnica de amostragem para *X*, a vantagem de *F1* sobre as outras técnicas de fragmentação pode ser assumida para *X*, independentemente das técnicas de indução.

3. Considerando a melhor técnica de amostragem (Heurística 1) e a melhor técnica de fragmentação (Heurística 2). Se o algoritmo de indução *I1* tem resultados melhores que os outros algoritmos de indução para o banco de dados *X* em uma combinação que esteja presente a melhor técnica de amostragem a melhor técnica de fragmentação para *X*, então pode-se assumir que *I1* é o melhor algoritmo de indução para *X*.

Apesar das três heurísticas serem verificadas em todos os nove bancos de dados verificados, elas não são 100% verdadeiras. Por exemplo, para *cars* e *connect-4* os raciocínios tiveram exceções. A próxima seção visa determinar se o algoritmo *Expert_Classifier_Inducer,* que utiliza as três heurísticas encontradas, consegue resultados equivalentes ao algoritmo *Naïve_Classifier_Inducer*.

5.3 Experimentos com *Expert_Classifier_Inducer*

Visando realizar uma avaliação do algoritmo *Expert_Classifier_Inducer*, selecionamos outros três banco de dados. Esses três bancos de dados também estão disponíveis junto ao repositório UCI A tabela 5.5 apresenta os bancos de dados.

BD	**Descrição**	**Atributos**	**Instâncias**	**Classes**
Cars-Evolution	Avaliação de automóveis	8	1728	4
Nursery	Escola Infantil	8	12960	5
Tic-tac-toe	Jogo *Tic-tac-toe*	10	953	2

Tabela 5.5: Bancos de Dados para Experimentos com *Expert_Classifier_Inducer*

A tabela 5.6 é uma síntese das acurácias obtidas pelos melhores classificadores inferidos para os três bancos de dados. As execuções dos algoritmos foram realizadas em um microcomputador com processador Pentium II 600MHz com 128MB de memória RAM. Os componentes acoplados ao framework para esta experimentação são os mesmos utilizados na experimentação anterior.

	Naïve_Classifier_Inducer			***Expert_Classifier_Inducer***		
O melhor classificador para	**Melhores técnicas**	? -?	**Tempo**	**Melhores técnicas**	? -?	**Tempo**
Cars-Evolution	*AIF – BS – Prism*	89,76%	1’56’’	*AIF – CV – ID3*	87,08	1’10’’
Nursery	*AIF – CV – ID3*	90,56%	9’52	*AIF – CV – ID3*	89,89%	6’23’’
Tic-tac-toe	*AIF – CV - Prism*	97,05%	2’56’’	*AIF – CV – Prism*	97,08%	1’49’’

Tabela 5.6: Síntese dos Experimentos com *Expert_Classifier_Inducer*

As colunas ? -? apresentam a acurácia média obtida pelo classificador subtraída o desvio padrão. Constata-se que os classificadores inferidos com a utilização de

Naïve_Classifier_Inducer e *Expert_Classifier_Inducer* tem acurácias muito próximas. Como explicar as pequenas variações apresentadas por execuções diferentes das mesmas técnicas? Dentre as técnicas de amostragem e fragmentação utilizadas na experimentação, diversas utilizam algum tipo de escolha aleatória ? por exemplo, BootStrap, que utiliza seleção aleatória de instâncias para compor o conjunto de treinamento ? . Essas escolhas aleatórias tornam muito baixa a probabilidade de duas execuções consecutivas, da mesma técnica, ter resultado idêntico. Contudo os resultados são muito próximos.

Quanto ao tempo de processamento dos dois algoritmos, *Naïve_Classifier_Inducer* apresentou em todas as execuções tempos superiores a *Expert_Classifier_Inducer*. O que comprova a diminuição do custo computacional. O ganho ? em termos de tempo de processamento ? não é muito grande para este número de técnicas, mas com o aumento do número de técnicas em cada conjunto, o ganho tenho a ser maior.

Para *Nursery* e *Tic-tac-toe*, os algoritmos *Expert_Classifier_Inducer* e *Naïve_Classifier_Inducer* selecionaram igualmente o melhor conjunto de técnicas. Para *Cars-Evolution* os algoritmos divergiram quanto ao melhor conjunto de técnicas, entretanto, sem grande perda na qualidade do melhor classificador.

Nossos experimentos permitem concluir que o algoritmo *Expert_Classifier_Inducer* consegue resultados iguais ao algoritmo *Naïve_Classifier_Inducer* em muitos casos. Nos casos onde o resultado não é igual, o melhor classificador inferido por *Expert_Classifier_Inducer* é muito próximo do melhor classificador inferido por *Naïve_Classifier_Inducer*, isto é, está entre os melhores. Tal resultado confirma que as três heurísticas utilizadas não construção do algoritmo *Expert_Classifier_Inducer* são válidas.

5.4 Considerações Finais

Este capítulo apresentou uma avaliação experimental do framework desenvolvido. Inicialmente foi feita uma avaliação do algoritmo *Naïve_Classifier_Inducer*, o qual se mostrou eficaz na determinação do melhor conjunto de técnicas e na inferência do melhor classificador, em todos os experimentos

realizados. Também ficou evidente a instabilidade dos classificadores inferidos por técnicas diferentes para os mesmos bancos de dados. Na seqüência a observação das heurísticas utilizadas na elaboração do algoritmo *Expert_Classifier_Inducer* foi discutida.

Por fim, uma avaliação do algoritmo *Expert_Classifier_Inducer* foi realizada. Essa avaliação concluiu que os resultados obtidos por *Expert_Classifier_Inducer* são equivalentes a versão exaustiva, com custos de processamento inferiores.

No próximo capítulo um estudo de caso é realizado, visando avaliar a ferramenta desenvolvida com o problema da identificação automática de litofácies em poços de petróleo.

Capítulo VI

Estudo de Caso: Identificação Automática de Litofácies em Poços de Petróleo

O objetivo deste estudo de caso é avaliar a eficácia do framework desenvolvido na inferência de conhecimento a partir de bancos de dados. O estudo de caso é realizado com dados reais de um campo produtor de petróleo. Os dados foram disponibilizados pela Agência Nacional do Petróleo sob a forma de um CD-ROM denominado Campo Escola de Namorado.

Este capítulo discute inicialmente alguns conceitos importantes da industria de petróleo, para facilitar a compreensão do estudo de caso realizado. Na seqüência são apresentados os resultados obtidos pela experimentação, bem como o conhecimento inferido. Por fim algumas considerações sobre o estudo são apresentadas.

6.1 O Petróleo

Petróleo é o nome genérico dado às misturas naturais de hidrocarbonetos. Quimicamente qualquer petróleo é uma mistura extremamente complexa de hidrocarbonetos, outros compostos de carbono e mais algumas impurezas como oxigênio, nitrogênio, enxofre e metais. Dependendo da temperatura e da pressão a que está submetido, o petróleo pode se apresentar no estado sólido, líquido ou gasoso. Geralmente, o petróleo é encontrado no estado líquido ? também chamado de óleo cru, ou simplesmente óleo ? , ou no estado gasoso ? conhecido como gás natural ? , ou em ambos os estados ? parte no estado líquido e parte no estado gasoso ? , em equilíbrio. O petróleo no estado líquido é uma substância oleosa, inflamável, menos densa que a água, com cheiro característico e cor variando de acordo com sua origem, oscilando entre o negro e o âmbar [Popp, 1988]. A Tabela 6.1 mostra a análise elementar do óleo.

Hidrogênio	11% - 14%
Carbono	83% - 87%
Enxofre	0,06% - 8%
Nitrogênio	0,11% - 1,7%
Oxigênio	0,1% - 2%
Metais	Até 0,3%

Tabela 6.1: Análise elementar do óleo cru típico[8]

A alta porcentagem de carbono e hidrogênio existente no petróleo mostra que os seus principais constituintes são os hidrocarbonetos ? compostos químicos orgânicos formados por carbono e hidrogênio. Independente da origem, todos os petróleos contêm substancialmente os mesmos hidrocarbonetos em diferentes quantidades, o que resulta em diferentes características dos tipos de petróleo. A variação do petróleo de acordo com seus constituintes indica o tipo de derivado produzido: querosene de aviação, diesel, lubrificantes, gasolina, solvente, asfalto, etc.

6.2 Geologia do Petróleo

O petróleo tem origem a partir da matéria orgânica depositada junto com os sedimentos. A interação dos fatores ? matéria orgânica, sedimentos e condições termoquímicas apropriadas ? é fundamental para o início da cadeia de processos que leva à formação do petróleo [Popp,1988].

Após o processo de geração, é necessário que ocorra a migração do óleo e que esta tenha seu caminho interrompido pela existência de algum tipo de armadilha geológica ou trapa, para ter a acumulação do petróleo. A rocha onde o petróleo é gerado é chamada *geradora* ou fonte e onde se acumula, reservatório. Para que uma rocha se constitua em um reservatório, esta deve apresentar espaços vazios no seu interior (porosidade) que devem estar interconectados, conferindo-lhe a característica de permeabilidade [Popp, 1988].

[8] Tabela obtida a partir de [Thomas,2001]

Para que ocorra a acumulação do petróleo, é necessário que alguma barreira se interponha no seu caminho. Esta barreira é produzida pela *rocha selante*, cuja característica principal é sua baixa permeabilidade. A descoberta de uma jazida de petróleo em uma nova área é uma tarefa que envolve um longo e dispendioso estudo e análise de dados geofísicos e geológicos das *bacias sedimentares* [Popp, 1988].

6.3 Testemunho

A operação de *testemunhagem* é uma das operações especiais que podem ocorrer durante a perfuração de um poço de petróleo. A testemunhagem é o processo de obtenção de uma amostra de rocha de sub-superfície, chamada testemunho. Com a análise deste testemunho obtém-se informações valiosas sobre a geologia da formação (tais como litologia, textura, porosidade, permeabilidade, saturação de óleo e água etc), que serão utilizadas pela engenharia de reservatórios [Thomas, 2001].

Quando o geólogo quer obter uma amostra da formação que está sendo perfurada, a equipe de sonda coloca uma coroa de testemunho no barrilete. A *coroa de testemunho* é uma broca com um furo no meio que permite que a broca corte o testemunho. O *barrilete de testemunho* é um tubo especial que geralmente mede 9, 18 ou 27 metros. O barrilete, que é aonde irá se alojar o testemunho, é colocado na parte interna da coluna de perfuração. Durante a operação, à medida que a coroa avança, o cilindro de rocha não fragmentado é encamisado pelo barrilete interno e posteriormente trazido à superfície.

Os testemunhos permitem que os geólogos analisem uma amostra da rocha da formação. No testemunho com barrilete convencional, ao final de cada corte de um testemunho é necessário trazer a coluna à superfície através de uma manobra, o que aumenta o tempo e o custo da operação. Assim, foi desenvolvida o testemunho a cabo, onde o barrilete interno pode ser removido até à superfície sem a necessidade de se retirar à coluna [Thomas, 2001].

Algumas vezes pode haver a necessidade de se testemunhar alguma formação já perfurada. Nestes casos, emprega-se o método de testemunho lateral onde cilindros ocos, presos por cabos de aço a um canhão, são arremessados contra a parede da formação para retirar amostras da rocha. Ao se retirar o canhão até a superfície, são arrastados os cilindros contendo as amostras retiradas da formação [Thomas, 2001].

A figura 6.1 apresenta um exemplo dos dados de testemunho, onde a coluna 1 representa a profundidade em relação ao perfil, a coluna 2 representa a profundidade em relação à ferramenta de testemunho, a coluna 3 representa o número da caixa de armazenamento do testemunho, a coluna 4 representa o tamanho das caixas em que o testemunho foi armazenado, a coluna 5 representa a granulometria, a coluna 6 representa as estruturas das rochas e, finalmente, a coluna 7 representa as litofácies encontradas.

Foram descritos 28 tipos de litofácies nos testemunhos dos poços do Campo Escola de Namorado. A tabela 6.2 apresenta os nomes das litofácies utilizadas nos experimentos. Os principais tipos de rocha no contexto da exploração de óleo são os arenitos e os folhelhos. Na maioria das vezes as rochas geradoras de óleo são folhelhos. Este tipo de rocha também constitui as rochas selantes, rochas que garantem o aprisionamento do óleo devido a sua baixa permeabilidade. Excepcionalmente, os folhelhos podem reter petróleo e os arenitos não.

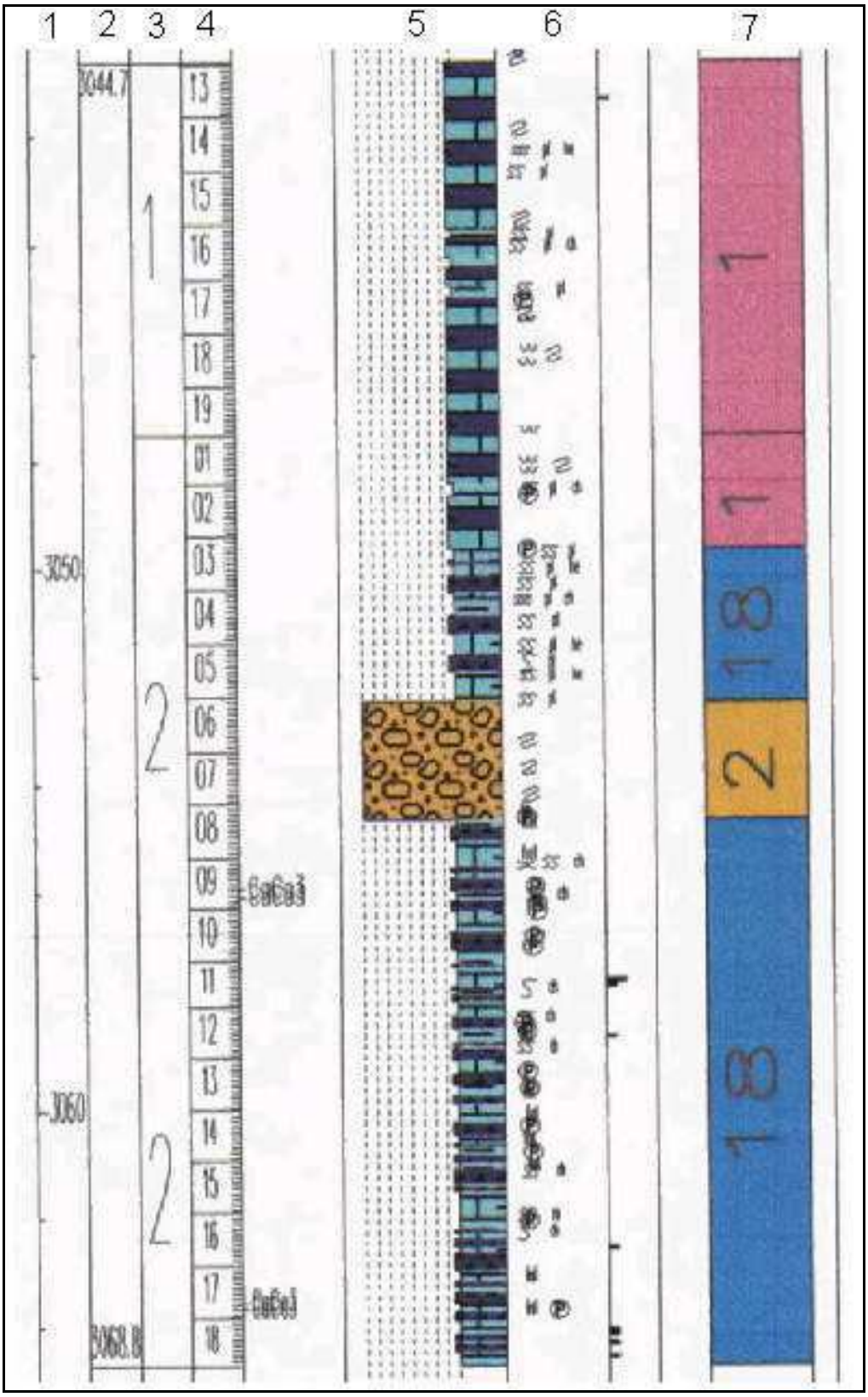

Figura 6.1: Descrição de um testemunho

Litofácies	Nome
1	Interlaminado Lamoso Deformado
2	Conglomerado e Brechas Carbonáticas
3	Diamictito Areno Lamoso
4	Conglomerados Residuais
5	Arenito com Intraclastos Argilosos
6	Arenito Grosso Amalgamado
7	Arenito Médio Laminado
8	Arenito Médio Maciço Gradado
9	Arenito Médio Cimentado
10	Arenito / Folhelho Interestratificado
11	Arenito / Folhelho Finamente Interestratificado
12	Siltito Argiloso Estratificado
13	Interlaminado Siltito Argiloso e Marga
14	Folhelho Radioativo
15	Interlaminado Arenoso Bioturbado
16	Interlaminado Siltito e Folhelho Bioturbado
17	Marga Bioturbada
18	Ritmito
19	Arenito Glauconítico
20	Folhelho com Níveis de Marga Bioturbados
21	Arenito Cimentado com Intraclastos
22	Siltito Argiloso / Arenito Deformado
23	Arenito Médio / Fino Laminado Cimentado
24	Interestratificado Siltito / Folhelho Intensamente Bioturbados
25	Folhelho Carbonoso
26	Arenito Maciço muito fino
27	Siltito Areno-Argiloso
28	Interlaminado Siltito Folhelho

Tabela 6.2: Litofácies presentes nos testemunhos do Campo Escola de Namorado

6.4 Perfilagem

Após a perfuração de uma fase do poço, geralmente são descidas várias ferramentas com a finalidade de medir algumas propriedades das formações, fundamentais para caracterização e avaliação econômica. Este processo é conhecido como perfilagem. A perfilagem permite obter informações importantes a respeito das formações atravessadas pelo poço, tais como litologia (tipo de rocha), porosidade, prováveis fluidos existentes nos poros e suas saturações. A maior limitação da perfilagem é a pequena extensão de seu raio de investigação lateral, de modo que apenas a vizinhança do poço é analisada pela perfilagem [Thomas, 2001].

A perfilagem pode revelar a existência de óleo e gás suficientes para justificar os gastos de completação do poço. Nas sondas terrestres a companhia contratada envia uma unidade de perfilagem montada em um caminhão, enquanto no mar a unidade é fixa na sonda, instalada num pequeno abrigo. A unidade de perfilagem é equipada com computadores, guinchos e controles que executam a operação. O perfil de um poço é a imagem visual, em relação à profundidade, de uma ou mais características ou propriedades das rochas perfuradas (resistividade elétrica, potencial eletroquímico natural, tempo de trânsito de ondas mecânicas, radioatividade natural ou induzida etc). Tais perfis, obtidos através do deslocamento contínuo de um sensor de perfilagem (sonda) dentro do poço, são denominados genericamente de perfis elétricos, independentemente do processo físico de medição utilizado [Thomas, 2001]. A ferramenta de perfilagem é descida no poço em um cabo condutor até a profundidade desejada. A unidade puxa a ferramenta que sobe pelo poço detectando certos aspectos da formação por onde ela passa. A informação é enviada à superfície pelo cabo condutor e registrada pelos computadores. O registro é impresso para posterior análise.

Existem vários tipos de perfis utilizados para as mais diversas aplicações, todos com o objetivo de avaliar melhor as formações geológicas quanto à ocorrência de uma jazida comercial de hidrocarbonetos. Os perfis mais comuns são: Raios Gama, Neutrônico, Resistividade, Sônico, Densidade.

?? **Raios Gama (GR)**: permite detectar e avaliar a radioatividade total da formação geológica. Utilizado na identificação da litologia, identificação de minerais

radioativos e para o cálculo do volume de argilas ou argilosidade. Também pode ser útil para interpretação de ambientes deposicionais e na investigação da subida do contato óleo-água em reservatórios fraturados.

?? **Neutrônico (NPHI)**: os perfis mais antigos medem a quantidade de raios gama de captura após excitação artificial através de bombardeio dirigido de nêutrons rápidos. Os mais modernos medem a quantidade de nêutrons epitermais e/ou termais da rocha após o bombardeio. São utilizados para estimativas de porosidade, determinação do volume de argila, pode auxiliar na identificação da litologia e dos fluidos da formação e detecção de hidrocarbonetos leves ou gás.

?? **Resistividade (ILD)**: fornece leitura aproximada da resistividade, através da medição de campos elétricos e magnéticos induzidos nas rochas. A resistividade é a propriedade da rocha permitir ou não a passagem de uma corrente elétrica.

?? **Sônico (DT)**: mede a diferença nos tempos de trânsito de uma onda mecânica através das rochas. É utilizado para estimativa de porosidade, identificação de litologia, correlação poço a poço, estimativas do grau de compactação das rochas ou estimativa das constantes elásticas, detecção de fraturas e apoio à sísmica para a elaboração do sismograma sintético.

?? **Densidade (RHOB)**: detecta os raios gama defletidos pelos elétrons orbitais dos elementos componentes das rochas, após terem sido emitidos por uma fonte colimada situada dentro do poço. Além da densidade das camadas, permite o cálculo da porosidade e a identificação das zonas de gás. É utilizado também como apoio à sísmica para o cálculo do sismograma sintético.

Em [Doveton, 1994] podem ser encontrados mais detalhes sobre os tipos de perfis mencionados acima.

O CD-ROM do Campo Escola de Namorado apresenta 56 arquivos de perfis de poços. Os dados de perfis se encontram no formato LAS, que é composto basicamente por um cabeçalho contendo informações sobre o poço e colunas numéricas, em que cada

coluna representa um perfil. A primeira coluna indica a profundidade em que a propriedade foi medida, conforme ilustrado na Figura 6.2.

```
~VERSION INFORMATION
 VERS.          2.0: CWLS Log ASCII Standard - Version 2.0
 WRAP.           NO: One Line per Depth Step
~WELL INFORMATION
#MNEM.UNIT              DATA            DESCRIPTION OF MNEMONIC
#----------             ----            -----------------------
 STRT.M          2950.0000              :Start Depth
 STOP.M          3200.0000              :Stop  Depth
 STEP.M             0.2000              :Step
 NULL.             -99999.0             :Null Value
 COMP   .   PETROLEO BRASILEIRO S/A     :Company
 WELL   .   3NA  0001A RJS              :Well
 FLD    .   NAMORADO                    :Field
 LOC    .                               :Location
 STAT   .   RIO DE JANEIRO              :State
 SRVC   .                               :Service Company
 DATE   .                               :Log Date
 API    .   742810040300                :API Code
~CURVE INFORMATION
 DEPT.M                                 :Measured Depth
 DT.                                    :01
 GR.                                    :02
 ILD.                                   :03
 NPHI.                                  :04
 RHOB.                                  :05
~ASCII LOG DATA
  2952.400    95.4766    74.3750    1.1445    25.7249    2.4929
  2952.600    94.8945    78.6992    1.1443    25.7249    2.4480
  2952.800    92.9727    80.0625    1.1831    25.7617    2.4355
  2953.000    91.8281    81.0391    1.2468    26.0015    2.4119
  2953.200    91.0703    81.8203    1.3105    25.3672    2.4176
  2953.400    90.0586    83.1836    1.3596    25.7031    2.4678
  2953.600    88.6602    83.6914    1.4053    26.9023    2.4412
  2953.800    88.1382    82.2656    1.4685    25.9531    2.4116
  2954.000    87.9492    79.0586    1.5327    24.0820    2.4775
  2954.200    88.5703    78.6836    1.5730    24.3438    2.5206
  2954.400    87.6211    81.9766    1.6030    24.8853    2.5220
  2954.600    85.3672    84.8750    1.6511    23.9453    2.5193
  2954.800    84.1523    85.9375    1.7231    23.0938    2.4961
  2955.000    84.1436    87.3555    1.7820    22.2891    2.4873
```

Figura 6.2: Fragmento de um arquivo de perfil

As informações contidas no cabeçalho, referentes ao poço, são as seguintes: profundidade do início da perfilagem (caso o poço seja marítmo, a lâmina d'água não entra no cálculo da profundidade); profundidade final da perfilagem, intervalo de medição do perfil (em metros), representação para valores nulos, nome da companhia,

nome do poço, nome do campo, localização, estado, nome da companhia contratada para realizar a perfilagem, data em que a perfilagem foi realizada. Em seguida, o cabeçalho contém o nome das colunas de dados, ou seja, a profundidade e os perfis que foram medidos.

6.5 O Problema de Identificação das Litofácies de um Reservatório de Petróleo

Para construir o modelo de um reservatório de forma confiável, é necessário fazer uma caracterização precisa do mesmo [Cunha and Gomes, 2002]. O testemunho é uma das técnicas mais antigas e ainda praticadas para extrair amostras das rochas de um reservatório. No entanto, testemunhar todos os poços em um campo muito grande pode ser economicamente inviável, além disso, o tempo consumido pode ser muito grande.

Dessa forma, consideramos como problema de mineração de dados inferir conhecimento que possa determinar automaticamente, isto é, apenas com a utilização dos perfis, as litofácies presentes nos poços de petróleo.

6.6 Dados Selecionados para Experimentação

Visando inferir o conhecimento necessário para identificar automaticamente as litofácies dos poços, unimos os dados de testemunho com os dados de perfis de vários poços diferentes do campo de Namorado. Desta forma foi formado um único grande banco de dados. Os poços do campo que não possuem testemunho foram descartados, bem como aqueles poços que não contêm todos os cinco perfis ? Raios Gama (GR), Sônico (DT), Indução (ILD), Densidade (RHOB) e Porosidade Neutrônica (NPHI).

Como apresentado na figura 6.1, os dados de testemunho encontram-se em uma planilha construída manualmente, denominada *descrição do testemunho*. Essa forma de apresentação torna difícil a identificação, em muitos casos, do tipo de litofácies de uma determinada região do testemunho, ou ainda, difícil definir com exatidão a profundidade exata de uma determinada litofácies no poço ? camadas muito finas ? assim como determinar com exatidão a profundidade de troca de um tipo de litofácies para outra.

Nestes casos, os dados das camadas muito finas foram removidos, bem como aqueles trechos de troca de tipo de litofácies, pois poderiam acarretar em ruído durante o processo de MD e dificultar o trabalho dos algoritmos de indução. Por exemplo, a figura 6.3 apresenta um intevalo testemunhado onde não é possível determinar, mesmo ampliando a figura, qual a litofácies presente entre as litofácies 13 e 17.

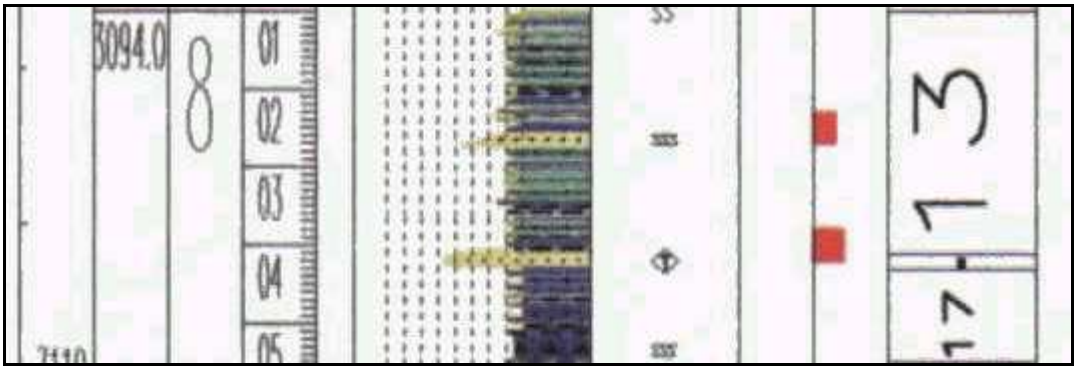

Figura 6.3: Intervalo testemunhado

6.7 Resultados

Para a execução do experimento foram selecionados quatro algoritmos de indução de conhecimento: *Naïve Bayes* (*Naïve)*, *OneR*, *ID3* e *Prism*; duas técnicas de amostragem: *Convergência* (*Conv*) e *Adaptive Incremental Framework* (*AIF*) ? com acurácia mínima para seleção da amostra de 90% ? ; três técnicas de fragmentação de amostras: *HoldOut* (*HO*) ? dez iterações ? , *K-fold Cross Validation* (*CV*) ? dez iterações ? e *Bootstrap* (*BS*) ? dez iterações ? . O componente de preparação de amostras acoplado ao framework removia todas as instâncias das amostras que continham dados faltando e para os atributos numéricos, tornava seus domínios discretos em cinco faixas.

Os experimentos foram realizados em duas etapas. Na primeira utilizamos o banco de dados com todos os atributos disponíveis, isto é: *Profundidade*, *DT*, *ILD*, *NPHI*, *RHOB* e *GR*. Na segunda etapa, subtraímos do banco de dados o atributo profundidade, visando inferir padrões mais genéricos para as litofácies, ou seja, padrões independentes da profundidade. A tabela 6.3 apresenta o resultado obtido pela execução dos algoritmos *Naive_Classifier_Inducer* e *Expert_Classifier_Inducer* sobre o

banco de dados completo. A tabela 6.4 relata os resultados obtidos pela execução dos algoritmos sobre o banco de dados de litofácies sem o atributo *Profundidade*.

Algoritmo	**Acurácia**	**Acc_e(90%)**	**Técnicas**
Naïve_Classifier_Inducer	90,60%	(89,19% ? 91,30%)	*AIF – CV – ID3*
Expert_Classifier_Inducer	90,38%	(89,17% ? 91,28%)	*AIF – CV – ID3*

Tabela 6.3: Acurácia obtida pelo melhor classificador inferido para banco de dados de Litofácies, com o atributo *Profundidade*

Algoritmo	**Acurácia**	**Acc_e(90%)**	**Técnicas**
Naïve_Classifier_Inducer	73,82%	(71,80% ? 74,20%)	*AIF – CV – ID3*
Expert_Classifier_Inducer	73,90%	(71,90% ? 74,30%)	*AIF – CV – ID3*

Tabela 6.4: Acurácia obtida pelo melhor classificador para o banco de dados de Litofácies, sem o atributo *Profundidade*

Tanto o algoritmo *Naïve_Classifier_Inducer* quanto o algoritmo *Expert_Classifier_Inducer* selecionaram o melhor conjunto de técnicas como sendo: a técnica de amostragem *Adaptive Incremental Framework*, a técnica de fragmentação *K-fold Cross Validation* e o algoritmo de indução de conhecimento *ID3*, para ambos os casos. A acurácia média do classificador para o problema, com a utilização do atributo *Profundidade*, superou os 90%. Resultado que pode ser considerado muito bom. Para a execução sem a utilização do atributo *Profundidade*, a acurácia ficou em torno de 73%.

O ranking de classificadores, que demonstra o resultado obtido por cada combinação de técnicas, é apresentado na figura 6.4. Ele foi obtido durante a execução do algoritmo *Naïve_Classifier_Inducer* sobre o banco de dados sem *Profundidade*.

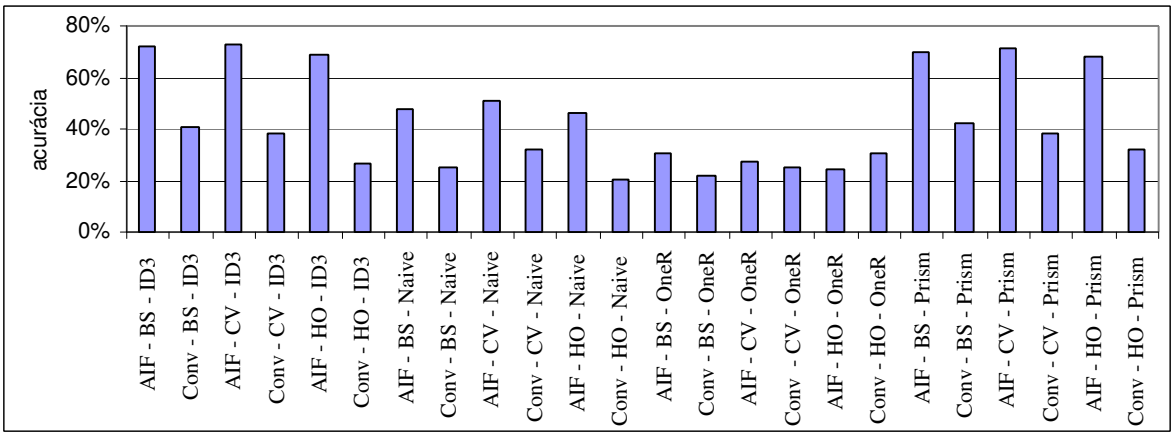

Figura 6.4: Ranking de classificadores para o banco de dados de *Litofácies*, sem o atributo *Profundidade*

Considerando que o melhor classificador para o problema, sem *Profundidade*, obteve acurácia em torno de 73%, uma interessante questão é determinar para que tipos de litofácies o classificador obtém mais acertos, e em que casos comete mais erros. A matriz de confusão apresentada na tabela 6.5 objetiva esclarecer essa questão.

a	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**	**j**	**k**	**l**	**m**	**n**	**o**	**p**	**q**	**r**	**s**	
258	0	2	1	1	0	3	0	0	4	0	2	0	0	10	1	0	0	0	**a=1**
4	**11**	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	2	0	**b=2**
2	0	**21**	0	2	0	6	0	2	1	2	1	0	3	2	0	0	1	0	**c=3**
1	1	0	**4**	0	0	0	0	0	0	0	1	0	1	1	0	2	0	0	**d=4**
1	0	4	0	**45**	0	2	0	0	0	1	1	0	0	14	0	1	0	0	**e=6**
1	0	0	1	0	**2**	0	0	0	0	0	0	0	0	0	0	0	0	0	**f=7**
5	0	7	0	4	0	**181**	0	0	0	13	3	0	0	5	0	0	0	0	**g=8**
0	0	0	0	0	0	0	**2**	0	0	1	3	0	0	0	0	0	0	0	**h=9**
0	0	3	0	0	0	2	0	**13**	0	1	2	0	0	0	0	1	0	0	**i=10**
4	0	0	0	0	0	1	0	0	**10**	0	2	0	1	0	0	1	0	0	**j=11**
0	0	3	0	0	0	12	0	0	2	**22**	3	2	6	1	0	0	0	0	**k=12**
2	0	3	1	1	0	5	0	1	0	4	**47**	0	0	3	0	5	0	0	**l=13**
0	1	0	0	1	0	0	0	1	0	3	0	**0**	0	0	0	0	1	0	**m=15**
2	0	0	1	2	0	0	0	0	0	3	0	0	**10**	6	0	0	0	0	**n=16**
8	1	1	0	4	0	1	0	0	1	3	3	0	1	**166**	0	1	0	0	**o=17**
41	0	0	0	0	0	0	0	0	0	0	3	0	0	0	**120**	0	0	0	**p=18**
1	0	0	1	2	0	0	0	0	0	0	6	0	0	0	0	**45**	0	0	**q=20**
3	3	7	0	1	0	1	0	0	0	0	0	1	0	0	0	0	**0**	0	**r=21**
0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	**0**	**s=22**

Tabela 6.5: Matriz de confusão para o problema das Litofácies

Na matriz, a letra *a* indica a litofácies do tipo 1, a letra *b* indica a litofácies do tipo 2, e assim por diante, conforme indica a última coluna da tabela. Todos os valores na diagonal principal da tabela indicam os acertos e estão em negrito. Considere a

primeira linha, litofácies do tipo 1, 258 casos do conjunto de teste foram classificados corretamente, 2 casos foram classificados como *c* ? ou seja, litofácies do tipo 3 ? , outros 10 casos foram classificados como *o* ? ou seja, litofácies do tipo 17 ? , e assim por diante.

Com a matriz podemos concluir que o classificados acerta mais para as litofácies cuja representação era maior no banco de dados minerado, como por exemplo as litofácies 1, 8 e 17. Para outros casos, onde a representação era menor, os erros foram maiores, como para as litofácies 12, 15 e 21.

A seguir o melhor classificador inferido para o problema, através da execução do algoritmo *Naïve_Classifier_Inducer*, para o banco de dados sem o atributo *Profundidade,* é apresentado na forma de uma árvore de decisão. O classificador se encontra neste modelo, pois sua inferência foi realizada pelo algoritmo *ID3* da família TDIDT.

A interpretação deste classificador pode ser feita da seguinte forma: nas regiões dos poços deste campo onde o valor do perfil *RHOB* entre 2,1575 e 2,1978 e o valor do perfil *GR* está entre 45,7542 e 49,3975, a litofácies presente é do tipo 09 (Arenito Médio Cimentado). Já caso o perfil *RHOB* entre 2,1575 e 2,1978 e o valor do perfil *GR* está entre 49,3945 e 53,6563 e o valor do perfil *ILD* estiver entre 3,21 e 7,7439, a litofácies presente é do tipo 01 (Interlaminado Lamoso Deformado) e assim por diante.

```
rhob = [2,1575 - 2,1978]
|  gr = [45,7542 - 49,3945]: litofácies = 09
|  gr = [49,3945 - 53,6563]
|  |  ild = [3,21 - 7,7439]: litofácies = 01
|  |  ild = [7,7439 - 32,4375]
|  |  |  dt = [92,375 - 95,4141]: litofácies = 03
|  |  |  dt = [95,4141 - 99,3562]
|  |  |  |  nphi = [24,1016 - 26,8633]: litofácies = 08
|  |  |  |  nphi = [26,8633 - 28,3757]: litofácies = 01
|  |  ild = [32,4375 - 40,964]: litofácies = 08
|  gr = [53,6563 - 57,875]
|  |  dt = [86,9844 - 92,375]
|  |  |  ild = [3,21 - 7,7439]: litofácies = 01
|  |  |  ild = [7,7439 - 32,4375]: litofácies = 08
|  |  |  ild = [32,4375 - 40,964]: litofácies = 03
|  |  dt = [92,375 - 95,4141]
|  |  |  nphi = [22,6267 - 24,1016]
|  |  |  |  ild = [7,7439 - 32,4375]: litofácies = 06
```

```
|   |   |   |  ild = [32,4375 - 40,964]: litofácies = 08
|   |   |  nphi = [24,1016 - 26,8633]
|   |   |   |  ild = [3,21 - 7,7439]: litofácies = 17
|   |   |   |  ild = [7,7439 - 32,4375]: litofácies = 03
|   |   |   |  ild = [32,4375 - 40,964]: litofácies = 08
|   |   |  nphi = [26,8633 - 28,3757]: litofácies = 01
|   |  dt = [95,4141 - 99,3562]
|   |   |  ild = [7,7439 - 32,4375]
|   |   |   |  nphi = [22,6267 - 24,1016]: litofácies = 08
|   |   |   |  nphi = [24,1016 - 26,8633]: litofácies = 10
|   |   |   |  nphi = [26,8633 - 28,3757]: litofácies = 08
|   |   |  ild = [32,4375 - 40,964]
|   |   |   |  nphi = [22,6267 - 24,1016]: litofácies = 08
|   |   |   |  nphi = [24,1016 - 26,8633]: litofácies = 08
|   |   |   |  nphi = [26,8633 - 28,3757]: litofácies = 08
|  gr = [57,875 - 65,3125]
|   |  nphi = [22,6267 - 24,1016]: litofácies = 12
|   |  nphi = [24,1016 - 26,8633]
|   |   |  ild = [1,0452 - 2,0781]: litofácies = 20
|   |   |  ild = [7,7439 - 32,4375]: litofácies = 13
|   |   |  ild = [32,4375 - 40,964]
|   |   |   |  dt = [86,9844 - 92,375]: litofácies = 17
|   |   |   |  dt = [92,375 - 95,4141]: litofácies = 17
|   |  nphi = [26,8633 - 28,3757]
|   |   |  dt = [80,6530 - 83,0884]: litofácies = 08
|   |   |  dt = [86,9844 - 92,375]
|   |   |   |  ild = [7,7439 - 32,4375]: litofácies = 10
|   |   |   |  ild = [32,4375 - 40,964]: litofácies = 08
|   |   |  dt = [92,375 - 95,4141]: litofácies = 13
|   |   |  dt = [95,4141 - 99,3562]
|   |   |   |  ild = [1,0452 - 2,0781]: litofácies = 20
|   |   |   |  ild = [3,21 - 7,7439]: litofácies = 10
|   |   |   |  ild = [7,7439 - 32,4375]: litofácies = 08
|   |   |   |  ild = [32,4375 - 40,964]: litofácies = 08
|  gr = [65,3125 - 72,2875]
|   |  ild = [3,21 - 7,7439]: litofácies = 10
|   |  ild = [7,7439 - 32,4375]
|   |   |  dt = [86,9844 - 92,375]: litofácies = 13
|   |   |  dt = [95,4141 - 99,3562]: litofácies = 01
|   |  ild = [32,4375 - 40,964]
|   |   |  nphi = [24,1016 - 26,8633]: litofácies = 15
|   |   |  nphi = [26,8633 - 28,3757]
|   |   |   |  dt = [92,375 - 95,4141]: litofácies = 02
|   |   |   |  dt = [95,4141 - 99,3562]: litofácies = 02
rhob = [2,1978 - 2,2776]
|  ild = [1,0452 - 2,0781]
|   |  gr = [45,7542 - 49,3945]
|   |   |  dt = [83,0884 - 86,9844]: litofácies = 17
|   |   |  dt = [86,9844 - 92,375]: litofácies = 02
```

```
|   |   |  dt = [92,375 - 95,4141]: litofácies = 17
|   |   |  dt = [95,4141 - 99,3562]: litofácies = 17
|   |   gr = [49,3945 - 53,6563]
|   |   |  nphi = [20,4688 - 22,6267]
|   |   |  |  dt = [83,0884 - 86,9844]: litofácies = 17
|   |   |  |  dt = [86,9844 - 92,375]: litofácies = 06
|   |   |  nphi = [22,6267 - 24,1016]
|   |   |  |  dt = [83,0884 - 86,9844]: litofácies = 17
|   |   |  |  dt = [86,9844 - 92,375]: litofácies = 13
|   |   |  nphi = [26,8633 - 28,3757]: litofácies = 17
|   |   gr = [53,6563 - 57,875]
|   |   |  nphi = [20,4688 - 22,6267]
|   |   |  |  dt = [80,6530 - 83,0884]: litofácies = 06
|   |   |  |  dt = [83,0884 - 86,9844]: litofácies = 06
|   |   |  |  dt = [86,9844 - 92,375]: litofácies = 17
|   |   |  nphi = [22,6267 - 24,1016]
|   |   |  |  dt = [80,6530 - 83,0884]: litofácies = 06
|   |   |  |  dt = [83,0884 - 86,9844]: litofácies = 06
|   |   |  |  dt = [86,9844 - 92,375]: litofácies = 17
|   |   |  |  dt = [92,375 - 95,4141]: litofácies = 13
|   |   |  |  dt = [95,4141 - 99,3562]: litofácies = 20
|   |   |  nphi = [24,1016 - 26,8633]
|   |   |  |  dt = [80,6530 - 83,0884]: litofácies = 06
|   |   |  |  dt = [83,0884 - 86,9844]: litofácies = 13
|   |   |  |  dt = [86,9844 - 92,375]: litofácies = 08
|   |   |  |  dt = [92,375 - 95,4141]: litofácies = 06
|   |   |  |  dt = [95,4141 - 99,3562]: litofácies = 06
|   |   |  nphi = [26,8633 - 28,3757]
|   |   |  |  dt = [83,0884 - 86,9844]: litofácies = 08
|   |   |  |  dt = [86,9844 - 92,375]: litofácies = 13
|   |   |  |  dt = [92,375 - 95,4141]: litofácies = 08
|   |   gr = [57,875 - 65,3125]
|   |   |  nphi = [18,8721 - 20,4688]
|   |   |  |  dt = [86,9844 - 92,375]: litofácies = 13
|   |   |  |  dt = [95,4141 - 99,3562]: litofácies = 08
|   |   |  nphi = [20,4688 - 22,6267]: litofácies = 20
|   |   |  nphi = [22,6267 - 24,1016]
|   |   |  |  dt = [83,0884 - 86,9844]: litofácies = 13
|   |   |  |  dt = [86,9844 - 92,375]: litofácies = 13
|   |   |  |  dt = [92,375 - 95,4141]: litofácies = 20
|   |   |  |  dt = [95,4141 - 99,3562]: litofácies = 20
|   |   |  nphi = [24,1016 - 26,8633]
|   |   |  |  dt = [83,0884 - 86,9844]: litofácies = 08
|   |   |  |  dt = [86,9844 - 92,375]: litofácies = 13
|   |   |  |  dt = [92,375 - 95,4141]: litofácies = 13
|   |   |  nphi = [26,8633 - 28,3757]: litofácies = 08
|   |   gr = [65,3125 - 72,2875]
|   |   |  nphi = [20,4688 - 22,6267]: litofácies = 20
|   |   |  nphi = [22,6267 - 24,1016]: litofácies = 20
```

```
|   |   |  nphi = [24,1016 - 26,8633]
|   |   |   |  dt = [92,375 - 95,4141]: litofácies = 17
|   |   |   |  dt = [95,4141 - 99,3562]: litofácies = 03
|   |   |  nphi = [26,8633 - 28,3757]: litofácies = 08
|   ild = [2,0781 - 3,21]
|   |   dt = [80,6530 - 83,0884]
|   |   |  gr = [49,3945 - 53,6563]: litofácies = 01
|   |   |  gr = [57,875 - 65,3125]: litofácies = 13
|   |   dt = [83,0884 - 86,9844]
|   |   |  gr = [45,7542 - 49,3945]: litofácies = 17
|   |   |  gr = [49,3945 - 53,6563]
|   |   |   |  nphi = [20,4688 - 22,6267]: litofácies = 17
|   |   |   |  nphi = [22,6267 - 24,1016]: litofácies = 01
|   |   |   |  nphi = [24,1016 - 26,8633]: litofácies = 01
|   |   |  gr = [53,6563 - 57,875]
|   |   |   |  nphi = [18,8721 - 20,4688]: litofácies = 17
|   |   |   |  nphi = [20,4688 - 22,6267]: litofácies = 01
|   |   |   |  nphi = [22,6267 - 24,1016]: litofácies = 17
|   |   |   |  nphi = [24,1016 - 26,8633]: litofácies = 13
|   |   |  gr = [57,875 - 65,3125]: litofácies = 17
|   |   |  gr = [65,3125 - 72,2875]: litofácies = 06
|   |   dt = [86,9844 - 92,375]
|   |   |  gr = [45,7542 - 49,3945]: litofácies = 17
|   |   |  gr = [49,3945 - 53,6563]
|   |   |   |  nphi = [22,6267 - 24,1016]: litofácies = 07
|   |   |   |  nphi = [24,1016 - 26,8633]: litofácies = 17
|   |   |  gr = [57,875 - 65,3125]
|   |   |   |  nphi = [22,6267 - 24,1016]: litofácies = 20
|   |   |   |  nphi = [24,1016 - 26,8633]: litofácies = 18
|   |   |  gr = [65,3125 - 72,2875]: litofácies = 13
|   |   dt = [92,375 - 95,4141]
|   |   |  gr = [45,7542 - 49,3945]: litofácies = 08
|   |   |  gr = [49,3945 - 53,6563]: litofácies = 20
|   |   |  gr = [53,6563 - 57,875]: litofácies = 04
|   |   |  gr = [57,875 - 65,3125]: litofácies = 20
|   |   |  gr = [65,3125 - 72,2875]
|   |   |   |  nphi = [20,4688 - 22,6267]: litofácies = 20
|   |   |   |  nphi = [22,6267 - 24,1016]: litofácies = 20
|   |   dt = [95,4141 - 99,3562]
|   |   |  nphi = [22,6267 - 24,1016]: litofácies = 16
|   |   |  nphi = [26,8633 - 28,3757]: litofácies = 08
|   ild = [3,21 - 7,7439]
|   |   dt = [80,6530 - 83,0884]
|   |   |  gr = [45,7542 - 49,3945]: litofácies = 08
|   |   |  gr = [53,6563 - 57,875]: litofácies = 18
|   |   |  gr = [57,875 - 65,3125]: litofácies = 04
|   |   |  gr = [65,3125 - 72,2875]: litofácies = 12
|   |   dt = [83,0884 - 86,9844]
|   |   |  gr = [45,7542 - 49,3945]: litofácies = 08
```

```
|   |   |  gr = [49,3945 - 53,6563]: litofácies = 17
|   |   |  gr = [53,6563 - 57,875]
|   |   |  |  nphi = [20,4688 - 22,6267]: litofácies = 16
|   |   |  |  nphi = [22,6267 - 24,1016]: litofácies = 04
|   |   |  gr = [57,875 - 65,3125]: litofácies = 01
|   |   |  gr = [65,3125 - 72,2875]
|   |   |  |  nphi = [18,8721 - 20,4688]: litofácies = 01
|   |   |  |  nphi = [20,4688 - 22,6267]: litofácies = 01
|   |   |  |  nphi = [22,6267 - 24,1016]: litofácies = 17
|   |  dt = [86,9844 - 92,375]
|   |   |  gr = [45,7542 - 49,3945]: litofácies = 08
|   |   |  gr = [49,3945 - 53,6563]
|   |   |  |  nphi = [22,6267 - 24,1016]: litofácies = 17
|   |   |  |  nphi = [24,1016 - 26,8633]: litofácies = 08
|   |   |  gr = [53,6563 - 57,875]
|   |   |  |  nphi = [22,6267 - 24,1016]: litofácies = 04
|   |   |  |  nphi = [24,1016 - 26,8633]: litofácies = 01
|   |   |  |  nphi = [26,8633 - 28,3757]: litofácies = 01
|   |   |  gr = [57,875 - 65,3125]: litofácies = 17
|   |   |  gr = [65,3125 - 72,2875]
|   |   |  |  nphi = [20,4688 - 22,6267]: litofácies = 06
|   |   |  |  nphi = [22,6267 - 24,1016]: litofácies = 06
|   |   |  |  nphi = [24,1016 - 26,8633]: litofácies = 06
|   |  dt = [92,375 - 95,4141]
|   |   |  nphi = [20,4688 - 22,6267]: litofácies = 11
|   |   |  nphi = [22,6267 - 24,1016]: litofácies = 16
|   |   |  nphi = [24,1016 - 26,8633]
|   |   |  |  gr = [49,3945 - 53,6563]: litofácies = 01
|   |   |  |  gr = [57,875 - 65,3125]: litofácies = 08
|   |   |  |  gr = [65,3125 - 72,2875]: litofácies = 11
|   |   |  nphi = [26,8633 - 28,3757]
|   |   |  |  gr = [45,7542 - 49,3945]: litofácies = 08
|   |   |  |  gr = [49,3945 - 53,6563]: litofácies = 07
|   |   |  |  gr = [57,875 - 65,3125]: litofácies = 12
|   |   |  |  gr = [65,3125 - 72,2875]: litofácies = 16
|   |  dt = [95,4141 - 99,3562]
|   |   |  nphi = [18,8721 - 20,4688]: litofácies = 15
|   |   |  nphi = [22,6267 - 24,1016]: litofácies = 17
|   |   |  nphi = [24,1016 - 26,8633]
|   |   |  |  gr = [57,875 - 65,3125]: litofácies = 12
|   |   |  |  gr = [65,3125 - 72,2875]: litofácies = 15
|   |   |  nphi = [26,8633 - 28,3757]
|   |   |  |  gr = [49,3945 - 53,6563]: litofácies = 17
|   |   |  |  gr = [57,875 - 65,3125]: litofácies = 10
|   |   |  |  gr = [65,3125 - 72,2875]: litofácies = 08
|  ild = [7,7439 - 32,4375]
|   |  gr = [45,7542 - 49,3945]
|   |   |  nphi = [18,8721 - 20,4688]: litofácies = 13
|   |   |  nphi = [20,4688 - 22,6267]: litofácies = 01
```

```
|   |   |  nphi = [22,6267 - 24,1016]
|   |   |   |  dt = [83,0884 - 86,9844]: litofácies = 08
|   |   |   |  dt = [86,9844 - 92,375]: litofácies = 01
|   |   |  nphi = [24,1016 - 26,8633]: litofácies = 01
|   |  gr = [49,3945 - 53,6563]
|   |   |  dt = [80,6530 - 83,0884]: litofácies = 15
|   |   |  dt = [83,0884 - 86,9844]
|   |   |   |  nphi = [18,8721 - 20,4688]: litofácies = 08
|   |   |   |  nphi = [20,4688 - 22,6267]: litofácies = 01
|   |   |  dt = [86,9844 - 92,375]: litofácies = 08
|   |   |  dt = [92,375 - 95,4141]: litofácies = 01
|   |   |  dt = [95,4141 - 99,3562]: litofácies = 08
|   |  gr = [53,6563 - 57,875]
|   |   |  nphi = [22,6267 - 24,1016]: litofácies = 08
|   |   |  nphi = [24,1016 - 26,8633]
|   |   |   |  dt = [86,9844 - 92,375]: litofácies = 11
|   |   |   |  dt = [92,375 - 95,4141]: litofácies = 10
|   |   |   |  dt = [95,4141 - 99,3562]: litofácies = 10
|   |   |  nphi = [26,8633 - 28,3757]: litofácies = 08
|   |  gr = [57,875 - 65,3125]
|   |   |  dt = [80,6530 - 83,0884]: litofácies = 11
|   |   |  dt = [83,0884 - 86,9844]
|   |   |   |  nphi = [20,4688 - 22,6267]: litofácies = 21
|   |   |   |  nphi = [22,6267 - 24,1016]: litofácies = 08
|   |   |   |  nphi = [26,8633 - 28,3757]: litofácies = 08
|   |   |  dt = [86,9844 - 92,375]
|   |   |   |  nphi = [22,6267 - 24,1016]: litofácies = 12
|   |   |   |  nphi = [24,1016 - 26,8633]: litofácies = 12
|   |   |  dt = [92,375 - 95,4141]
|   |   |   |  nphi = [22,6267 - 24,1016]: litofácies = 08
|   |   |   |  nphi = [24,1016 - 26,8633]: litofácies = 08
|   |   |  dt = [95,4141 - 99,3562]
|   |   |   |  nphi = [24,1016 - 26,8633]: litofácies = 12
|   |   |   |  nphi = [26,8633 - 28,3757]: litofácies = 08
|   |  gr = [65,3125 - 72,2875]
|   |   |  nphi = [20,4688 - 22,6267]
|   |   |   |  dt = [80,6530 - 83,0884]: litofácies = 03
|   |   |   |  dt = [83,0884 - 86,9844]: litofácies = 03
|   |   |   |  dt = [86,9844 - 92,375]: litofácies = 12
|   |   |   |  dt = [92,375 - 95,4141]: litofácies = 17
|   |   |  nphi = [22,6267 - 24,1016]
|   |   |   |  dt = [83,0884 - 86,9844]: litofácies = 03
|   |   |   |  dt = [86,9844 - 92,375]: litofácies = 03
|   |   |   |  dt = [92,375 - 95,4141]: litofácies = 13
|   |   |  nphi = [24,1016 - 26,8633]
|   |   |   |  dt = [86,9844 - 92,375]: litofácies = 12
|   |   |   |  dt = [92,375 - 95,4141]: litofácies = 12
|   |   |   |  dt = [95,4141 - 99,3562]: litofácies = 17
|   |   |  nphi = [26,8633 - 28,3757]
```

```
|   |   |   |  dt = [86,9844 - 92,375]: litofácies = 08
|   |   |   |  dt = [92,375 - 95,4141]: litofácies = 17
|   |   |   |  dt = [95,4141 - 99,3562]: litofácies = 03
|   |  ild = [32,4375 - 40,964]
|   |   |  nphi = [18,8721 - 20,4688]
|   |   |   |  dt = [80,6530 - 83,0884]: litofácies = 03
|   |   |   |  dt = [83,0884 - 86,9844]: litofácies = 06
|   |   |   |  dt = [86,9844 - 92,375]: litofácies = 04
|   |   |  nphi = [20,4688 - 22,6267]
|   |   |   |  gr = [49,3945 - 53,6563]: litofácies = 10
|   |   |   |  gr = [53,6563 - 57,875]: litofácies = 10
|   |   |   |  gr = [57,875 - 65,3125]: litofácies = 04
|   |   |  nphi = [22,6267 - 24,1016]
|   |   |   |  dt = [83,0884 - 86,9844]: litofácies = 03
|   |   |   |  dt = [86,9844 - 92,375]: litofácies = 08
|   |   |   |  dt = [92,375 - 95,4141]
|   |   |   |   |  gr = [53,6563 - 57,875]: litofácies = 06
|   |   |   |   |  gr = [57,875 - 65,3125]: litofácies = 06
|   |   |   |   |  gr = [65,3125 - 72,2875]: litofácies = 02
|   |   |  nphi = [24,1016 - 26,8633]
|   |   |   |  gr = [53,6563 - 57,875]: litofácies = 21
|   |   |   |  gr = [57,875 - 65,3125]: litofácies = 08
|   |   |   |  gr = [65,3125 - 72,2875]: litofácies = 02
|   |   |  nphi = [26,8633 - 28,3757]: litofácies = 13
rhob = [2,3932 - 2,4858]
|  nphi = [18,8721 - 20,4688]
|   |  gr = [45,7542 - 49,3945]
|   |   |  ild = [2,0781 - 3,21]: litofácies = 17
|   |   |  ild = [3,21 - 7,7439]: litofácies = 18
|   |   |  ild = [7,7439 - 32,4375]: litofácies = 13
|   |  gr = [49,3945 - 53,6563]
|   |   |  dt = [80,6530 - 83,0884]: litofácies = 06
|   |   |  dt = [83,0884 - 86,9844]: litofácies = 03
|   |  gr = [53,6563 - 57,875]: litofácies = 11
|   |  gr = [57,875 - 65,3125]: litofácies = 17
|  nphi = [20,4688 - 22,6267]
|   |  ild = [1,0452 - 2,0781]
|   |   |  gr = [45,7542 - 49,3945]: litofácies = 18
|   |   |  gr = [49,3945 - 53,6563]
|   |   |   |  dt = [83,0884 - 86,9844]: litofácies = 17
|   |   |   |  dt = [86,9844 - 92,375]: litofácies = 17
|   |   |  gr = [53,6563 - 57,875]
|   |   |   |  dt = [83,0884 - 86,9844]: litofácies = 17
|   |   |   |  dt = [86,9844 - 92,375]: litofácies = 17
|   |  ild = [2,0781 - 3,21]
|   |   |  gr = [45,7542 - 49,3945]: litofácies = 17
|   |   |  gr = [49,3945 - 53,6563]: litofácies = 01
|   |   |  gr = [53,6563 - 57,875]: litofácies = 11
|   |   |  gr = [57,875 - 65,3125]: litofácies = 11
```

```
|   |  ild = [3,21 - 7,7439]
|   |   |  gr = [49,3945 - 53,6563]: litofácies = 01
|   |   |  gr = [53,6563 - 57,875]
|   |   |   |  dt = [80,6530 - 83,0884]: litofácies = 01
|   |   |   |  dt = [83,0884 - 86,9844]: litofácies = 16
|   |   |  gr = [57,875 - 65,3125]: litofácies = 16
|   nphi = [22,6267 - 24,1016]
|   |  gr = [49,3945 - 53,6563]: litofácies = 17
|   |  gr = [53,6563 - 57,875]: litofácies = 17
|   |  gr = [57,875 - 65,3125]: litofácies = 01
|   |  gr = [65,3125 - 72,2875]: litofácies = 20
|   nphi = [24,1016 - 26,8633]: litofácies = 13
rhob = [2,4858 - 2,6042]
|   ild = [1,0452 - 2,0781]
|   |  gr = [45,7542 - 49,3945]: litofácies = 17
|   |  gr = [49,3945 - 53,6563]: litofácies = 17
|   |  gr = [57,875 - 65,3125]: litofácies = 01
|   ild = [2,0781 - 3,21]
|   |  dt = [80,6530 - 83,0884]: litofácies = 01
|   |  dt = [83,0884 - 86,9844]: litofácies = 01
|   |  dt = [86,9844 - 92,375]: litofácies = 20
|   ild = [3,21 - 7,7439]
|   |  gr = [45,7542 - 49,3945]
|   |   |  nphi = [18,8721 - 20,4688]
|   |   |   |  dt = [80,6530 - 83,0884]: litofácies = 01
|   |   |   |  dt = [83,0884 - 86,9844]: litofácies = 01
|   |   |   |  dt = [92,375 - 95,4141]: litofácies = 01
|   |   |   |  dt = [95,4141 - 99,3562]: litofácies = 01
|   |   |  nphi = [20,4688 - 22,6267]: litofácies = 01
|   |  gr = [49,3945 - 53,6563]
|   |   |  dt = [80,6530 - 83,0884]: litofácies = 01
|   |   |  dt = [83,0884 - 86,9844]: litofácies = 01
|   |   |  dt = [92,375 - 95,4141]: litofácies = 01
|   |   |  dt = [95,4141 - 99,3562]: litofácies = 01
|   |  gr = [53,6563 - 57,875]
|   |   |  dt = [80,6530 - 83,0884]: litofácies = 01
|   |   |  dt = [83,0884 - 86,9844]: litofácies = 18
|   |  gr = [57,875 - 65,3125]
|   |   |  nphi = [18,8721 - 20,4688]: litofácies = 01
|   |   |  nphi = [20,4688 - 22,6267]: litofácies = 06
|   |  gr = [65,3125 - 72,2875]: litofácies = 01
|   ild = [7,7439 - 32,4375]
|   |  gr = [45,7542 - 49,3945]: litofácies = 18
|   |  gr = [49,3945 - 53,6563]: litofácies = 09
|   ild = [32,4375 - 40,964]
|   |  dt = [80,6530 - 83,0884]
|   |   |  gr = [45,7542 - 49,3945]: litofácies = 15
|   |   |  gr = [49,3945 - 53,6563]: litofácies = 06
|   |  dt = [86,9844 - 92,375]: litofácies = 21
```

6.8 Considerações Finais

Este capítulo apresentou um estudo de caso visando demonstrar a eficácia do framework de mineração de dados proposto. Inicialmente foram discutidos alguns conceitos importantes da industria de petróleo, para facilitar a compreensão do estudo de caso realizado. Na seqüência foram apresentados os resultados obtidos pela experimentação, bem como o conhecimento inferido. A experimentação foi realizada em duas etapas. A primeira utilizando todos os atributos disponíveis no banco de dados. A segunda, geologicamente mais interessante, retirando do banco de dados o atributo *Profundidade*.

Os algoritmos *Naïve_Classifier_Inducer* e *Expert_Classifier_Inducer* selecionaram igualmente o melhor conjunto de técnicas para o problema, demonstrando que seus resultados são equivalentes. O melhor classificador inferido obteve acurácia em torno de 90%, com o atributo *Profundidade*, e acurácia em torno de 73%, sem o atributo *Profundidade.*

O próximo capítulo descreve as conclusões deste trabalho e as perspectivas de trabalhos futuros.

Capítulo VII

Sumário e Conclusões

Esta dissertação de mestrado descreve uma proposta de automatização para o processo de inferência de classificadores utilizando técnicas de mineração de dados (MD). A automatização do processo visa diminuir as dificuldades encontradas pelo minerador na tarefa de obter bons classificadores para seus problemas de MD. Essas dificuldades são devidas à instabilidade do conhecimento inferido, variando segundo as diferentes técnicas empregadas.

O primeiro capítulo discutiu o contexto do trabalho e fez uma breve introdução a Mineração de Dados, com destaque para o processo de MD. O processo de MD pode ser decomposto em várias etapas, dentre as quais estão: a preparação dos dados, a seleção de amostras, a fragmentação da amostra, a indução do conhecimento e a avaliação do conhecimento. O primeiro capítulo também enumerou os objetivos da dissertação.

No capítulo 2, foram apresentados diversos conceitos e técnicas MD que são fundamentais para uma melhor compreensão da nossa proposta. Dentre os conceitos definidos, citamos: banco de dados classificado, conjunto de treinamento, conjunto de teste, conjunto de execução, algoritmo de indução de conhecimento, classificador, acurácia de teste, acurácia de execução e amostra para MD. As técnicas discutidas foram: *Adaptive Incremental Framework* e *Convergência* (seleção de amostras); *Holdout*, *K-fold Cross Validation* e *Bootstrap* (fragmentação de amostras); *OneR*, Naïve *Bayes*, *ID3* e *Prism* (algoritmos de indução de conhecimento).

O capítulo 3 foi dedicado a nossos dois algoritmos que automatizam o processo de MD para inferência de classificadores. O primeiro algoritmo, denominado *Naïve_Classifier_Inducer* determina a melhor combinação de técnicas de amostragem, fragmentação e indução para inferir o melhor classificador para o problema, dentre as

técnicas disponíveis. A escolha da melhor combinação é exaustiva, isto é, o algoritmo testa todas as possíveis combinações de técnicas. Apesar de atingir o objetivo proposto, qual seja o de encontrar o melhor classificador possível, esse algoritmo tem um custo de processamento potencialmente elevado. Dependendo do número de técnicas utilizadas em cada etapa do processo, o custo de processamento do algoritmo *Naïve_Classifier_Inducer* pode ser inviável.

Através da observação de inúmeros experimentos com *Naïve_Classifier_Inducer*, pudemos descobrir três heurísticas no sentido de inferir classificadores de qualidade, sem no entanto escolher exaustivamente a melhor combinação de técnicas de amostragem, fragmentação e indução. As heurísticas permitiram a construção de uma versão do algoritmo de menor custo computacional, *Expert_Classifier_Inducer*.

O capítulo 4 tratou da implementação das duas versões do algoritmo, na forma de um framework orientado a objetos. Essa forma de implementação foi escolhida porque permite a fácil adição de novos componentes ao software ? requisito de flexibilidade. Dessa forma, novas técnicas de amostragem, fragmentação, ou indução, podem ser facilmente acopladas ao software.

O capítulo 5 refere-se à avaliação experimental dos algoritmos *Naïve_Classifier_Inducer* e *Expert_Classifier_Inducer*. Na primeira fase dos experimentos, utilizamos 9 bancos de dados de diversas áreas para avaliar o algoritmo *Naïve_Classifier_Inducer*. Ficou evidente a instabilidade do conhecimento inferido por técnicas diferentes para o mesmo problema. Técnicas que obtiveram bons resultados para alguns bancos de dados, apresentaram resultados até péssimos para outros bancos de dados. Entretanto, o algoritmo conseguiu determinar o melhor conjunto de técnicas para cada problema, para inferir o melhor classificador possível com as técnicas disponíveis.

Para os testes com o algoritmo *Expert_Classifier_Inducer*, foram utilizados três bancos de dados diferentes dos nove bancos de dados utilizados para a descoberta das três heurísticas, com o auxílio do algoritmo *Naïve_Classifier_Inducer*. Os resultados obtidos com a versão "expert" foram então comparados com os da versão "naïve", para os mesmos três novos bancos de dados. Apesar de não fazer pesquisa exaustiva, o algoritmo *Expert_Classifier_Inducer* conseguiu resultados idênticos a

Naïve_Classifier_Inducer para dois dos três bancos de dados, ou seja, a mesma combinação de técnicas para a inferência de classificadores com acurácias aproximadamente iguais. Para o outro bancos de dados, os melhores conjuntos de técnicas selecionados pelos algoritmos foram diferentes; no entanto, a acurácia do melhor classificador inferido pelo *Expert_Classifier_Inducer* foi muito próxima da acurácia obtida pelo melhor classificador inferido pelo algoritmo *Naïve_Classifier_Inducer*.

O capítulo 6 apresentou um estudo de caso, onde o framework desenvolvido foi utilizado para a classificação automática de litofácies em poços de petróleo, como parte de um projeto de pesquisa financiado pela Petrobrás, através da Agência Nacional de Petróleo. A identificação de litofácies é uma das etapas fundamentais do processo de caracterização de um reservatório de óleo e gás natural. Para tanto, inicialmente foram apresentados conceitos do setor de petróleo e gás, para uma melhor compreensão do problema. Em seguida os resultados obtidos pela experimentação foram expostos. Tanto *Naïve_Classifier_Inducer* quanto *Expert_Classifier_Inducer* selecionaram o mesmo conjunto de técnicas para o problema: a técnica de amostragem *Adaptive Incremental Framework*, a técnica de fragmentação *K-Fold Cross Validation* e o algoritmo de indução de conhecimento *ID3*. O classificador inferido por essas técnicas atingiu 90% de acurácia de teste.

Nossos experimentos permitem concluir que o algoritmo *Expert_Classifier_Inducer* consegue resultados (classificadores e suas respectivas acurácias) aproximadamente iguais ao algoritmo *Naïve_Classifier_Inducer* em todos os casos, e a um custo bem menor. Tal fato serve como validação das três heurísticas utilizadas pelo algoritmo *Expert_Classifier_Inducer*.

7.1 Limitações e Trabalhos Futuros

Algumas limitações deste trabalho, que podem gerar trabalhos futuros são destacadas a seguir:

?? Os processos automatizados propostos na forma dos algoritmos *Naive_Classifier_Inducer* e *Expert_Classifier_Inducer*, não consideram outras tarefas de mineração, como por exemplo: as regras de associação e

"clustering". Um possível trabalho futuro poderia acoplar tais tarefas ao framework e propor algoritmos que automatizem essas tarefas.

?? O framework proposto trabalha exclusivamente com banco de dados classificados. Um trabalho futuro poderia propor a utilização de outros tipos de bancos de dados, como por exemplo: bancos de dados orientado a objeto e multidimensional.

?? Os algoritmos propostos ? *Naive_Classifier_Inducer* e *Expert_Classifier_Inducer* ? escolhem a melhor combinação de técnicas, apenas considerando a acurácia dos classificadores inferidos. Um possível trabalho futuro poderia propor outras métricas para escolha do melhor classificador, como por exemplo: concisão e legibilidade do conhecimento inferido.

?? O software projetado não considera técnicas de combinação de classificadores. Poder-se-ia acoplar ao framework algumas técnicas de deste tipo, como por exemplo: *Bagging* e *Boosting*.

?? A interface gráfica desenvolvida apresenta o conhecimento de maneira textual simplificada. Um trabalho futuro poderia propor e acoplar ao software diferentes formas de apresentação do conhecimento, tornando deste modo, mais atrativo e legível aos usuários.

?? Os algoritmos propostos poderiam ser acoplados a um SGBD, visando a diminuição do tempo gasto em suas execuções.

?? Esforços adicionais são necessários visando diminuir ainda mais o custo de processamento do algoritmo *Expert_Classifier_Inducer*. Para tanto, outras heurísticas precisam ser encontradas.

?? Visando aumentar a acurácia do melhor classificador inferido para o problema da identificação automática de litofácies em poços de petróleo, o número de litofácies poderia ser menor, mediante o agrupamento de litofácies com características semelhantes. Com o agrupamento das litofácies, o número de casos de cada grupo será maior e com isso a possibilidade de ocorrer "overfitting" será menor. "Overfitting" ocorre quando o algoritmo de indução gera regras a partir de casos isolados, com pouca capacidade de generalização. Esse grupamento só pode ser

realizado com a ajuda de um geólogo especialista no domínio dos dados. Tal união não trás prejuízo algum para a qualidade dos resultados sobre o ponto de vista dos geólogos que trabalham com a prospecção de campos de petróleo e gás.

?? Finalmente, é possível explorar técnicas de paralelismo (hardware e software) para desenvolver uma versão distribuída dos algoritmos *Naive-Classifier_Inducer* e *Expert_Classifier_Inducer*, com o objetivo de diminuir ainda mais o seu custo de processamento.

Referências Bibliográficas e Bibliografia

[Agrawal et al., 1993] Agrawal, R.; Imielinski T.; Swami A.: "Database Mining: A Performance Perspective". IEEE Transactions on Knowledge and Data Engineering, Special Issue on Learning and Discovery in Knowledge-Based Databases, Vul. 5, No. 6, December 1993, pp. 914 – 925.

[Bayardo, 1997] Bayardo R. J.: "Brute-force Mining of High-Confidence Classification Rules". Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97), AAAI Press, 1997, pp. 364 – 389.

[Blake and Merz, 2002] Blake C. L.; Merz C. J.: "UCI Repository of Machine Leraning Databases [http://www.ics.uci.edu/~mlearn/MLRepository.html]". Irvine, CA: University of California, Department of Information and Computer Science, 2002.

[Bramer, 2000] Bramer M. A.: "Automatic Induction of Classification Rules from Examples Using N-*Prism*". Proceedings in Research and Development in Intelligent Systems XVI, Springer-Verlag, 2000, pp. 99 – 121.

[Brause et al.,1999] Brause R.; Langsdorf T. Hepp M.: "Neural Data Mining for Credit Card Fraud Detection". Proceedings of the 11th IEEE International Conference on Tools with Artificial Intelligence, 1999, pp. 79-95.

[Brumem et al., 2001] Brumen B.; Welzer T.; Jaakkola H.: "*Adaptive Incremental Framework* for Performance Driven Data Mining". Advances in Databases and Information Systens (ADBIS 2001), Lithuania, September 2001, pp. 193 – 203.

[Cendrowska, 1987] Cendrowska, K: "*Prism*: An Algorithm for Inducing Modular Rules". International Journal of Man-Machine Studies, Number 4, Volume 27, 1987, pp. 349 – 370.

[Cunha and Gomes, 2002] Cunha, E. S.; Gomes, H. M.: "Identificação de litofácies de poços de petróleo utilizando um método baseado em redes neurais". Proceedings of the I Workshop de Teses e Dissertações em Inteligência Artificial, 2002.

[DBMiner, 2002] DBMiner Technology Inc: "DBMiner 2.0 (EnterPrise) User Manual", 2002, 70 p.

[Doveton, 1994] Doveton J. H.: "Geologic Log Interpretation". SEPM Short Course, Socity of Sedimentary Geology, Tulsa, Oklahoma, nº 29, 1994.

[Elder and Pregibon, 1996] Elder, J. F.; Pregibon, D.: "A Statistical Perpective on Knowledge Discovery in Databases". Proceedings of the Advances in Discovery and Data Mining, AAAI/MIT Press, 1996, pp. 83 – 113.

[Fayyad et al., 1996] Fayyad Usama; Shapiro-Piatetsky Gregory; Smyth Padhraic; Uthurusamy Ramasamy: "Adavances in Knowledge Discovery and Data Mining". AAAI Press, 1996, 611 p.

[Fayyad et. al, 1997] Fayyad, Usama; Piatetski-Shapiro, Gregory; Padhraic, Smyth: "From Data Mining to Knowledge Discovery in Databases". AI Magazine, 1996, pp. 38-54.

[Frawley et al., 1992] Frawley, W. J. et al.: "Knowledge Discovery in Databases: An Overview". AI Magazine 13(3), 1992, pp. 57 – 70.

[Frey and Fisher, 1999] Frey L. J.; Fisher D. H.: "Modeling Decision Tree Performance with the Power Law. Proceedings of the 7th International Workshop on Artificial Intelligence and Statistics, 1999, pp. 59 – 65.

[Gamma et al., 1994] Gama E.; Helm R.; Vlissides J.: "Design Patterns: Elements of Reusable Object-Oriented Software". Addison-Wesly Publishers, 1994.

[Giffrida et al., 2000] Giovanni et. Al.: "Mining Classification Rules from Datasets withLarge Number of Many-Valued Attributes". Proceedings of the 7th International Conference on Extending Database Technology (EDBT), Konstanz, Germany, 2000, pp. 335 – 349.

[IBM, 2002] IBM Inteligent Miner for Data: "Utilizando o Inteligent Miner for Data Versão 6 Release 1", 2002, 410 p.

[Jain and Dubes, 1997] Jain A. K.; Dubes R. C.: "*Bootstrap* Techniques for error Estimation". IEEE Transactions on Pattern Analysis and Machine Intelligence, 1997, pp. 628 – 633.

[Kamber, et al., 1997] Kamber M.; Winstone L.; Cheng S.; Han J.: "Generalization and Decision Tree Induction: Efficient Classification in Data Mining". Proceedings of 1997 International Workshop on Research Issues on Data Engineering (RIDE'97), Birmingham, England, April 1997, pp. 111 – 120.

[Kohavi, 1995] Kohavi Ron: "A Study of Cross-Validation and *Bootstrap* for Accuracy Estimation and Model Selection". Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), 1996, pp. 1137 – 1143.

[Lee et all. 1995] Lee, H. Y.; Ong, H. L.; Quek L. H.: "Exploiting Vizualization in Knowledge Dsicovery. Proceedings 1st International Conference of Discovery and Data Mining (KDD-95), 1995, pp. 198-203

[Oates and Jensen, 1997] Oates T.; Jensen D.: "The Effects of Training Set Size on de Decision Tree Complexity". Proceedings of the Fourteenth Internation Conference on Machine Learning, 1997, pp. 379 – 390.

[Piatetsky-Shapiro, 1991] Piatetsky-Shapiro G.: "Knowledge Discovery in Real Databases". AI Magazine, Vol 11, No 5, Special issue, January 1991, pp. 68 – 70.

[Popp, 1988] Popp J. H.: "Geologia Geral". Editora: Livros Técnicos e Científicos, 1988.

[Provost et al., 1999] Provost, F.; Jensen, D.; and Oates, T.: "Efficient Progressive Sampling". Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining, San Diego, CA: ACM SIGKDD, 1999, pp. 23-32.

[Pyle, 1999] Pyle Dorian: "Data Preparation for Data Mining". San Francisco: Morgan Kaugmann Publishers, 1999, 540 p.

[Quinlan, 1993] Quinlan J. R.: "C4.5: Programs for Machine Learning". San Mateo: Morgan Kaufmann Publishers, 1995.

[Rastogi and Shim, 1998] Rastogi, R.; Shim K.: "A Decision Tree Classifier that Intergrates Building and Pruning". Proceedings of the 24th International Conference on Very large DataBases, San Francisco: Morgan Kaugmann Publishers, 1998, pp. 404 – 415.

[Theodoridis, 1999] Theodoridis S.: "Pattern Recognition". Academic Pess, 1st Esition, 1999.

[Thomas, 2001] Thomas J. E.: "Fundamentos de Engenharia de Petróleo". Editora: Interciência, 2001.

[Thrun et al., 1991] Thrun S. B.; Bala J.; Bloendorn E.; Bratko I.: "The Monk's Problems – A Performance Comparison of Different Learning Algorithms". Technical Report CMU-CS-91-197, Carnegie Mellon University, 1991.

[Tok et al., 2001] Tok W. H., et al.: "Predator-Miner: Ad hoc Mining of Association Rules Within a Database Management Ststem". Proceeding of the 18th Internacional Conference on Data Engeneering, 2001, pp. 327 – 328.

[Vasconcelos and Sampaio, 2002] Vasconcelos, B.; Sampaio, M. C.: "Mineração Eficiente de Regras de Classificação com Sistemas de Banco de Dados Objeto Relacional". Proceedings of the 17th Simpósio Brasileiro de Banco de Dados, Gramado – RS, October 2002.

[Witten and Eibe, 1999] Witten Ian H.; Eibe Frank: "Data Mining: Pratical Machine Learning Tools and Techniques with Java Implementations". San Diego: Morgan Kaufmann Publishers, 1999, 369 p.

[Xiao-Bai, 1999] Xiao-Bai, Li: "A Baysian Method for Estimating and Replacing Missing Data. IEEE". AI Magazine, 1999, pp. 29 – 34.