CAPÍTULO

2

Descoberta do Conhecimento em Base de Dados

Neste capítulo será apresentado o embasamento teórico para auxiliar a compreensão do processo de Descoberta do Conhecimento em Base de Dados (Knowledge Discovery in Databases). Também serão apresentadas as 5 fases desse processo: coleta, pré-processamento, transformação, mineração e avaliação e interpretação dos resultados; bem como as principais tarefas e técnicas encontradas em cada uma destas fases.

2.1 Contextualização

Atualmente é alarmante a distância crescente entre a geração de dados e a capacidade de analisá-los e compreende-los. À medida que o volume de dados aumenta, a proporção dos dados que é analisada e entendida pelas pessoas diminui e escondido entre todo este volume de dados está a informação potencialmente útil (Batista, 2003). Existe, portanto, a necessidade de técnicas e ferramentas que possibilite os analistas humanos compreender estas grandes bases de dados. Tais técnicas e ferramentas são objetivos de estudo de uma área de pesquisa chamada de *Descoberta do Conhecimento em Base de Dados* (DCBD).

Existem diversas definições para DCBD (Fayyad et al., 1996), (Batista, 2003), (Neves, 2003), uma das mais utilizadas é a proposta por Fayyad et al. (1996), que define Descoberta do Conhecimento em Base de Dados como um processo não trivial para identificar padrões válidos, novos, potencialmente úteis e compreensíveis em dados existentes. Por não trivial é entendido que alguma busca ou inferência é utilizada, isto é, DCBD não é um simples cálculo de quantidades

pré-definidas, como calcular média de um conjunto de valores; Os padrões descobertos pelo processo de DCBD devem ser validados com novos dados. Esses padrões também devem ser novos e potencialmente úteis, isto é, devem levar algum benefício ao usuário ou a aplicação. Por fim, os padrões devem ser compreensíveis aos analistas humanos, se não imediatamente, ao menos após algum pós-processamento.

Segundo Batista (2003), frequentemente três atores estão envolvidos no DCBD: o analista de dados, o especialista de domínio e o usuário:

O Analista de Dados: é aquele que entende das técnicas envolvidas no processo de DCBD. Este ator tem conhecimento sobre o funcionamento dos algoritmos e das ferramentas utilizadas no processo, mas não necessariamente conhece o domínio ao qual os dados pertencem.

O Especialista no Domínio: conhece o domínio no qual o processo de descoberta de conhecimento será aplicado. Por exemplo, pode-se utilizar DCBD para encontrar padrões existentes entre os dados de uma Operadora de Plano de Saúde, neste caso o especialista no domínio pode ser um médico, ou alguém com grande conhecimento dos dados e funcionamento dessa operadora.

O Usuário: é aquele que irá utilizar o resultado do processo de DCBD. Normalmente o usuário não é somente uma pessoa, mas uma instituição, uma empresa ou um departamento.

Esses três atores denotam diferentes habilidades, mas não são, necessariamente pessoas diferentes. Por exemplo, frequentemente os papéis de usuário e de especialista de domínio são exercidos por uma mesma pessoa quando o usuário possui conhecimento detalhado do domínio de aplicação (Saitta e Neri, 1998).

Influenciado pelas várias metodologias de DCBD existentes como as apresentadas por Fayyad et al. (1996), Batista (2003), Neves (2003), considera-se neste trabalho que o processo de descoberta do conhecimento é estruturado em 5 (cinco) fases: a) coleta, b) pré-processamento, c) transformação, d) mineração de dados, e) avaliação e interpretação dos resultados, as quais podem ser repetidas em múltiplas iterações; É importante comentar que não existe nenhum rigor quanto a sequência em que estas fases devem ocorrer, porém é de fundamental importância a presença do especialista do domínio do problema durante todo o processo. A Figura 2.1 ilustra as fases do processo de DCBD e os detalhamentos de cada uma são apresentados a seguir.

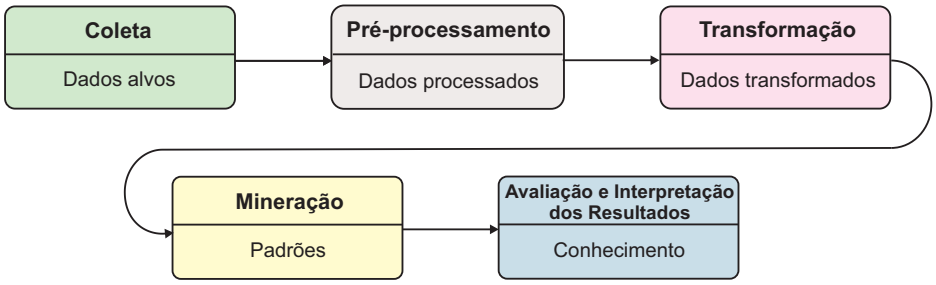

Figura 2.1: Visão geral das fases do processo de descoberta do conhecimento (adaptado de Fayyad et al. (1996)).

2.2 Coleta

A fase de coleta envolve inicialmente entrevistas com o especialista do domínio a fim de identificar os objetivos do projeto. Levantamento dos recursos disponíveis (pessoal, dados, hardware, software, financeiros, etc) e a verificação da existência de qualquer conhecimento prévio que possa contribuir ao alcance dos objetivo também são realizados nesta fase (Neves, 2003).

Uma vez que o projeto foi definido e entendido, é necessário identificar quais dados serão utilizados no desenvolvimento do projeto. O especialista no domínio pode fornecer ao analista de dados quais as informações são, na sua opinião, mais relevantes para a criação do modelo proposto. Entretanto, para não limitar a originalidade do conhecimento descoberto, sempre que possível, o analista de dados deve adicionar novas informações e verificar a importância destas no conhecimento gerado. Também é importante verificar se essas novas informações existem nos bancos de dados da empresa ou se elas podem ser encontradas em fontes de dados externas, como arquivos físicos por exemplo (Batista, 2003).

Coletar dados é uma atividade crítica porque os dados podem não estar disponíveis em um formato apropriado para serem utilizados no processo de DCBD e algumas vezes, mesmo que disponíveis, eles podem precisar ser rotulados com o auxílio de um especialista, como relatado por Provost e Danyluk (1995). Muitos dos sistemas de gerenciamento de dados que estão funcionando hoje foram criados há muitos anos, quando as técnicas de Engenharia de Software ainda não estavam bem desenvolvidas e/ou pouco difundidas. Como resultado, muitos desses sistemas são proprietários e possuem documentação insatisfatória, o que faz com que o processo de coleta de dados se torne difícil e dependente do especialista de domínio.

Alguns dos principais desafios encontrados nesta fase são: problemas legais e éticos, motivos estratégicos, problemas de conectividade (ver Figura 2.2).

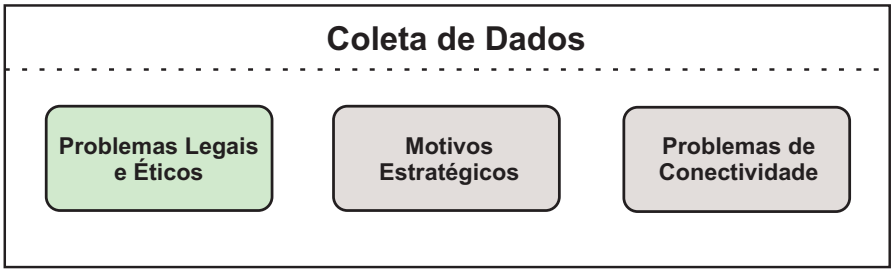

Figura 2.2: Principais desafios encontrados na fase de coleta.

Problemas Legais e Éticos: Podem existir barreiras legais ou éticas que impeçam que dados sejam disponibilizados para análise. Por exemplo, em uma Operadora de Plano de Saúde existem barreiras legais que, sob algumas circunstâncias, impedem que dados referentes a identificação de pessoas com determinadas doenças sejam disponibilizados.

Motivos Estratégicos: Podem haver motivos estratégicos que impeçam o acesso a parte dos dados ou até mesmo a algumas estatísticas sobre eles. Por exemplo Chan e Stolfo (1998) descrevem uma análise para identificação de operações fraudulentas em cartões de créditos na qual a distribuição das classes foi alterada. A proporção de operações fraudulentas e não fraudulentas é uma informação estratégica mantida em absoluto segredo pelas companhias de cartão de crédito.

Problemas de Conectividade: os problemas de conectividade surgem quado os sistemas armazenam os dados de forma descentralizada e existe a necessidade da conexão entre esses dados, pois é necessário que eles estejam conectados ao sistema que irá analisá-los. Por exemplo, se uma operadora de plano de saúde possuir filiais em diferentes cidades e cada uma destas filiais possuir seu próprio banco de dados local de forma independente e desconectada das demais, o problema de conectividade está caracterizado.

2.3 Pré-processamento

É nesta fase que uma análise inicial dos dados é feita para se ter sólidas definições sobre estrutura das tabelas, valores dos atributos, formatos e tipos de dados, além de toda e qualquer operação necessária a escolha dos dados relevantes aos objetivos do projeto (Neves, 2003).

De uma forma geral, a etapa de pré-processamento de dados é um processo semiautomático. Por semiautomático entende-se que esta fase depende da capacidade do analista de dados em identificar os problemas presentes nos dados, além da natureza desses problemas, e utilizar os métodos mais apropriados para solucionar cada um dos problemas.

Algumas das principais tarefas encontrados nesta fase são: tratamento de valores desconhecidos, tratamento de classes desbalanceadas e seleção de atributos (ver Figura 2.3).

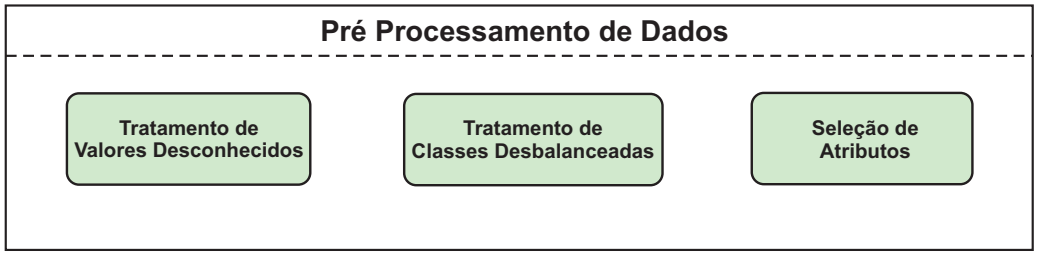

Figura 2.3: Principais tarefas encontradas na fase de pré-processamento.

2.3.1 Tratamento de Valores Desconhecidos

Um problema relevante em qualidade de dados é a presença de valores desconhecidos ou valores ausentes. Valores desconhecidos ou ausentes consistem no não preenchimento dos valores de um atributo para determinados casos. Esse problema pode originar-se de diversas fontes, como recusa por parte dos entrevistados em responder determinadas perguntas, não disponibilidade da

informação no momento da entrada de dados, dentre outras. Apesar da frequente ocorrência de valores desconhecidos em conjuntos de dados, muitos analistas de dados tratam os valores desconhecidos de forma bastante simplista. Entretanto, o tratamento de valores desconhecidos deve ser cuidadosamente pensado, caso contrário, distorções podem ser introduzidas no conhecimento induzido (Batista, 2003).

Algumas das principais técnicas para o tratamento do problema de valores desconhecidos são: ignorar ou descartar dados, imputação, conhecimento do domínio, *hot deck* e *cold deck* e modelo de predição (ver Figura 2.4).

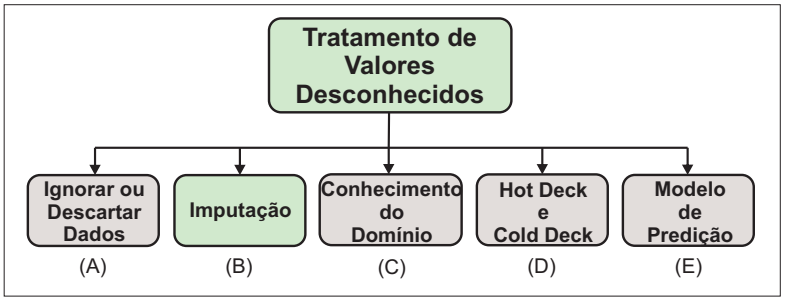

Figura 2.4: Principais formas de tratamento do problema dos valores desconhecidos.

2.3.1.(A) - Ignorar ou descartar dados

Existem duas abordagens mais utilizadas para desconsiderar dados com valores desconhecidos: *descarte de casos e/ou atributos* e *análise de casos completos*. A primeira abordagem consiste em descartar qualquer caso que possua um ou mais valores desconhecidos, já a segunda, consiste em determinar a extensão dos valores desconhecidos em cada caso e em cada atributo, para em seguida remover os casos e/ou atributos com grandes quantidades de valores desconhecidos.

2.3.1.(B) - Imputação

Imputação consiste em substituir valores desconhecidos por valores estimados por meio de alguma informação extraída do conjunto de dados. Duas das principais técnicas de imputação são: *substituição de casos* e *imputação pela média ou moda*. A primeira técnica consiste em substituir um caso com valores desconhecidos por outro caso que não estava presente na amostra inicial, já a segunda, consiste em substituir o valor desconhecido da informação de um determinado caso pela média (em atributos quantitativos) ou moda (em atributos qualitativos) da mesma informação de todos os outros casos.

2.3.1.(C) - Conhecimento do domínio

Conhecimento de domínio pode ser utilizado pelo especialista de domínio para substituir os valores desconhecidos por valores estimados por meio da experiência do especialista. De uma forma

geral, esse procedimento é seguro quando o especialista está familiarizado com a aplicação, pois o conjunto de dados é grande e o número de valores desconhecidos é pequeno. Alternativamente, o especialista no domínio pode discretizar um atributo quantitativo (por exemplo: o atributo **renda** pode ser discretizado em Classe A, Classe B, Classe C e Classe D) de forma que se possa predizer com mais confiança em qual categoria está o caso com valor desconhecido. Sendo assim, a variável discreta pode substituir a variável quantitativa na análise, levando-se em consideração que existe uma perda de informação nesta transformação.

Esse método possui o mérito de ser menos conservador que a simples remoção de casos ou atributos. Estando familiarizado com a aplicação, o especialista no domínio será capaz de estimar os valores com uma precisão maior que uma substituição pela média. Entretanto, essa substituição é manual, portanto ela fica restrita a pequenas quantidades de valores desconhecidos. Ainda, as estimativas do especialista são limitadas ao conhecimento existentes sobre os dados, o que de certa forma pode direcionar o conhecimento a ser aprendido.

2.3.1.(D) - Hot deck e cold deck

No método *hot deck*, um valor desconhecido é substituído por um valor obtido por meio de uma distribuição estimada a partir dos dados disponíveis. O método *hot deck* é tipicamente implementado em dois estágios. No primeiro estágio, o conjunto de dados é particionado em *clusters* utilizando um método de Aprendizado Supervisionado. No segundo estágio, cada exemplo com valores desconhecidos é associado a um dos *clusters*. Os exemplos completos no *clusters* são utilizados para estimar os valores desconhecidos. Uma forma de estimar os valores desconhecidos é por meio do cálculo da média ou da moda do atributo, utilizando somente os exemplos membros do *cluster*. A diferença entre os métodos *hot deck* e *cold deck* é que eles geram os *clusters* de maneira diferente. O primeiro utiliza todos os dados para gerar o agrupamento, enquanto que o segundo utiliza apenas os exemplos que não possuem valores desconhecidos.

2.3.1.(E) - Modelo de predição

Modelos de predição são procedimentos sofisticados para tratar valores desconhecidos. Esses métodos consistem na criação de um modelo preditivo para estimar valores que vão substituir os valores desconhecidos. O atributo com valores desconhecidos é utilizado como atributo classe, e os demais atributos são utilizados como entrada para o modelo de predição. Um importante argumento a favor desta abordagem é que, frequentemente, os atributos possuem correlações entre si. Dessa forma, essas correlações podem ser utilizadas para criar um modelo preditivo de classificação ou regressão dependendo do tipo do atributo com valores desconhecido. Uma limitação importante desta abordagem é que os valores estimados são geralmente mais bem comportados do que os valores reais (não conhecidos) seriam, ou seja, os valores preditos são mais consistentes com o conjunto de atributos do que os valores reais seriam. Uma segunda limitação é a necessidade

por correlações entre os atributos. Se não existem correlações entre um ou mais atributos em um conjunto de dados, então o modelo preditivo não será preciso em estimar os valores desconhecidos.

2.3.2 Tratamento de Conjuntos com Classes Desbalanceadas

O problema de classes desbalanceadas corresponde a domínios nos quais uma classe é representada por um grande número de exemplos, enquanto que a outra é representada por poucos exemplos. A maioria dos algoritmos de aprendizagem de máquina têm dificuldades em criar um modelo que classifique com precisão os exemplos da classe minoritária (Batista, 2003). O problema de classes desbalanceadas é de grande importância uma vez que conjuntos de dados com essa características podem ser encontrados em diversos domínios. Por exemplo, nos dados da Operadora de Plano de Saúde utilizados neste trabalho a quantidade de procedimentos "Não Autorizado" era bem menor em relação a quantidade de procedimentos "Autorizados".

Muitos sistemas de aprendizado assumem que as classes estão balanceadas e, dessa forma, esses sistemas falham em induzir um classificador que seja capaz de predizer a classe minoritária com precisão na presença de dados com classes desbalanceadas. Frequentemente, o classificador possui uma boa precisão para a classe majoritária, mas uma precisão inaceitável para a classe minoritária. O problema agrava-se ainda mais quando o custo da classificação incorreta da classe minoritária é muito maior do que o custo de classificação incorreta da classe majoritária. No mundo real, essa é a norma para a maior parte das aplicações com conjuntos de dados com classes desbalanceadas, uma vez que essas aplicações visam traçar o perfil de um pequeno conjunto de entidades valiosas que estão dispersas em um grande grupo de entidades "pouco interessantes".

Segundo Japkowicz e Stephen (2002) vários pesquisadores têm analisado o problema de aprender a partir de conjuntos de dados com classes desbalanceadas. Dentre os diversos métodos propostos por esses pesquisadores, duas abordagens principais têm sido utilizadas com maior frequência: *Under-sampling* e *Over-sampling*. No *Under-sampling* alguns exemplos das classes mais populosas são removidos, enquanto que no *Over-sampling* são inseridos mais exemplos nas classes menos populosas. Essas duas abordagens possuem alguns sub-métodos para tratar o problema das classes desbalanceadas, tais como: *under* e *over sampling* aleatório, ligações tomek, seleção unilateral, limpeza de vizinhança e SMOTE (ver Figura 2.5).

2.3.1.(A) - *Under* e *Over sampling* aleatório

Em suas versões mais simples, a adição/remoção de exemplos nas classes é feita de maneira aleatória. Nesses casos, os métodos são geralmente chamados de *over-sampling* aleatório e *under-sampling* aleatório. O *Over-sampling* aleatório é um método não heurístico que replica aleatoriamente exemplos da classe minoritária. Já o *Under-sampling* aleatório também é um método não heurístico, porém este elimina de forma aleatória exemplos da classe majoritária. Por serem aleatórios, ambos os métodos possuem limitações conhecidas. *Under-sampling* aleatório pode eliminar dados potencialmente úteis, e *over-sampling* aleatório pode aumentar a probabilidade de

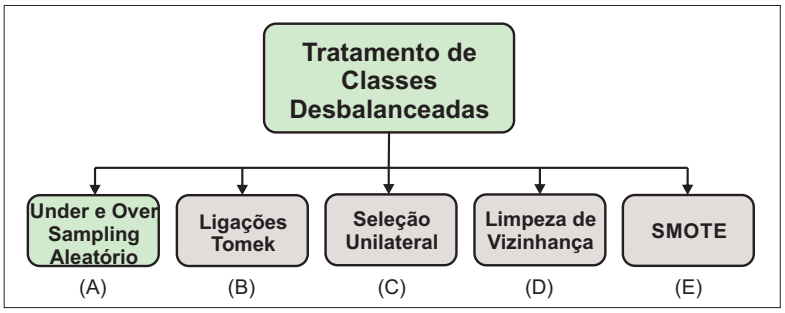

Figura 2.5: Principais métodos de tratamento do problema dos conjuntos com classes desbalanceadas.

ocorrer *overfitting*, uma vez que a maioria dos métodos de *over-sampling* fazem cópias exatas dos exemplos pertencentes à classe minoritária (Prati, 2006).

2.3.1.(B) - Ligações Tomek

Casos próximos a borda e ruídos podem ser identificados por meio das ligações Tomek, e removidos do conjunto de dados. Uma ligação Tomek pode ser definida da seguinte maneira:

Sejam e_i e e_j dois exemplos de classes diferentes. Seja $d(.)$ uma função de distância entre exemplos. Um par de exemplos (e_i, e_j) constitui uma ligação Tomek se não existir um exemplo e_k tal que a distância $d(e_k, e_i) < d(e_i, e_j)$ ou $d(e_k, e_j) < d(e_i, e_j)$.

Se dois exemplos $d(e_i, e_j)$ formam uma ligação Tomek, então, ou e_i e e_j são exemplos próximos a borda de decisão, ou um desses exemplos é possivelmente ruído. No uso de ligações Tomek para o balanceamento de conjuntos de dados, apenas os exemplos da classe majoritária que possuem ligações Tomek são removidos.

2.3.1.(C) - Seleção unilateral

É um método que consiste da aplicação do algoritmo para a identificação de ligações Tomek seguido da aplicação do *K-Nearest Neighbor* (KNN). Essa abordagem é conhecida como seleção unilateral. Ligações Tomek são utilizadas para verificar exemplos da classe majoritária que se sobrepõem a classe minoritária e o KNN é utilizado para remover exemplos muitos distantes da fronteira de decisão. O restante dos exemplos da classe majoritária e todos os exemplos da classe minoritária são usados para a aprendizagem (Kubat e Matwin, 1997).

2.3.1.(D) - Limpeza da vizinhança

O método de limpeza da vizinhança (Laurikkala, 2001) (*Neighborhood Cleaning Rule*) utiliza a regra do vizinho mais próximo ENN (*Edited Nearest Neighbor Rule*) de Wilson (1972) para remover exemplos da classe majoritária. O ENN remove exemplos cuja classe difere da classe por, pelo menos, 2 dos seus 3 vizinhos mais próximos.

2.3.1.(E) - SMOTE

A abordagem adotada no método SMOTE (*Synthetic Minority Over-sampling Technique*) consiste em gerar casos sintéticos para a classe de interesse a partir dos casos existentes. Os novos casos são gerados na vizinhança de cada caso da classe minoritária com o intuito de crescer o espaço de decisão desta classe (região do $\Re^n$) e aumentar o poder de generalização dos classificadores. Visualmente, no espaço amostral do conjunto de dados, os novos casos sintéticos são interpolados aleatoriamente ao longo do segmento de reta que une cada caso da classe minoritária a um de seus k vizinhos mais próximos, escolhidos de forma aleatória (Chawla et al., 2002).

2.3.3 Seleção de Atributos

Nesta seção, de pré-processamento, já foram descritos os métodos de tratamento dos problemas de valores desconhecidos e classes desbalanceadas, agora serão descritos os principais métodos do processo de seleção de atributos.

O processo de seleção de atributos é necessário para identificar e eliminar inconsistências presentes nos dados, como: poluição de dados, valores preenchidos de forma *default* e informações duplicadas e redundantes (Batista, 2003).

Poluição de dados: presença de dados distorcidos, os quais não representam os valores verdadeiros. Uma possível fonte de poluição de dados é a tentativa, por parte dos usuários do sistemas, de utilizar o sistema além das suas funcionalidades originais. Por exemplo, imagine uma operadora de plano de saúde cujo banco de dados possuía um campo *sexo* para armazenar o sexo dos seus clientes. Entretanto, alguns valores assumiam o valor 'PJ' para este atributo, o qual, posteriormente, descobriu-se que este valor correspondia a informação *Pessoa Jurídica*. Originalmente, o sistema tinha sido projetado somente para cadastrar pessoas físicas, porém quando empresas começaram a contratar planos de saúde, não havia um campo especifico para identificar que aquele cadastro era de uma empresa. Essa informação foi então armazenada no campo *sexo*.

Valores preenchidos de forma *default*: consiste no preenchimento, com o mesmo valor, para determinados casos que as informações não foram especificadas. Esse problema acontece porque a maioria dos sistemas gerenciadores de banco de dados permite a inserção de valores *defaults* para alguns atributos. Um exemplo desses problema na área médica acontece com o atributo *mês de gravidez*, que por padrão é preenchido com "0", quando essa informação não é especificada. Porém, este atributo pode ser preenchido de maneira incorreta para uma mulher grávida por desatenção do usuário.

Informações duplicadas e redundantes: consiste no armazenamento de informações idênticas em diversos atributos. Um exemplo disso é a presença dos atributos *data de nascimento*, *data de aniversário* e *idade* em uma tabela, ou seja, a partir da data de nascimento pode-se obter facilmente a data de aniversário e a idade de uma pessoa. O maior dano causado pela redundância é um aumento no tempo de processamento dos algoritmos de aprendizado.

Embora também possam ser utilizados todos os atributos de uma base de dados, a maioria dos autores consultados defende a seleção de atributos relevantes e a eliminação dos que são totalmente desnecessários, na tentativa de diminuir a complexidade do problema e alcançar um desempenho ótimo de aprendizagem (Neves, 2003).

Complementarmente pode-se afirmar que a seleção de atributos desempenha um papel importante na preparação de dados para a mineração, pois quanto mais se remove dados redundantes e irrelevantes, mais se reduz a dimensionalidade do espaço de atributos (Liu e Yu, 2002).

Segundo Riano (1997) existem várias razões que justificam a redução do número de atributos: alguns atributos podem ser inúteis ou podem refletir informações que já estão embutidas em outros atributos. A redução do número de atributos reflete na criação de estruturas menores que permitem um melhor entendimento do domínio. Outra vantagem é que o custo tempo-espaço dos algoritmos de indução está diretamente relacionado ao número de atributos considerado, ou seja, a redução do número de atributos ocasiona na redução do custo de tempo-espaço do algoritmo.

Assim, os efeitos imediatos da seleção de atributos para a aplicação são: a execução mais rápida dos algoritmos utilizados nas fases seguintes e a melhoria da qualidade dos dados, que consequentemente conduz a um melhor desempenho no processo de DCBD e no aumento da compreensibilidade dos resultado obtidos (Liu e Yu, 2002).

Segundo Kohavi (1997), pode-se dividir as abordagens de seleção de atributos em três categorias: filtro, *wrapper* e *embedded* (ver Figura 2.6). Outros autores, como Bloedorn e Michalski (1998), também consideram a construção de atributos como parte da seleção de atributos.

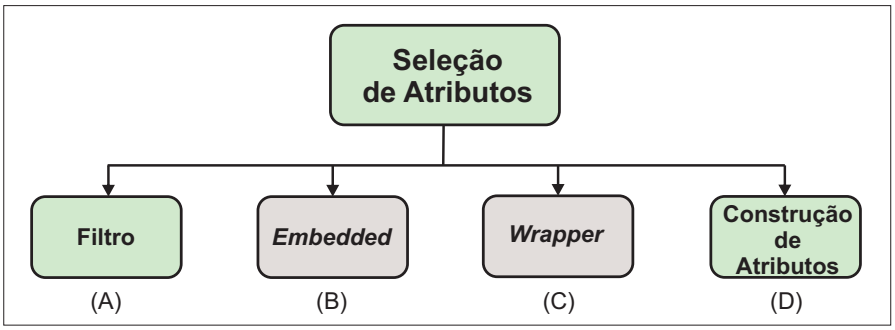

Figura 2.6: Principais categorias do processo de seleção de atributos.

2.3.3.(A) - Técnicas de seleção de atributos do tipo filtro

As técnicas de seleção do tipo filtro selecionam o subconjunto de atributos na fase de pré-processamento de forma independente do classificador utilizado (Guyon e Elisseff, 2003).

O nome "filtro" deriva da ideia de que os atributos irrelevantes são *filtrados* da base de dados antes da aplicação do algoritmo de classificação (Blum e Langley, 1997). Os filtros usam as informações da própria base de treinamento para escolher os atributos a serem utilizados.

Uma forma de realizar a seleção de atributos de uma base de dados é utilizar uma métrica para avaliar a qualidade de cada um dos atributos individualmente. Dessa forma, é possível ordenar o conjunto de atributos em um *ranking*, que pode então ser utilizado para selecionar todos os atributos cujo valor da métrica esteja acima de um determinado limiar (*threshold*), ou mesmo um número fixo de melhores atributos, em termos absolutos ou percentuais (Pereira, 2009).

Uma maneira de medir a qualidade de um atributo para a classificação é avaliar o seu grau de associação com a classe. Para determinar esse tipo de associação, a literatura apresenta diversas métricas, que serão descritas a seguir (ver Figura 2.7).

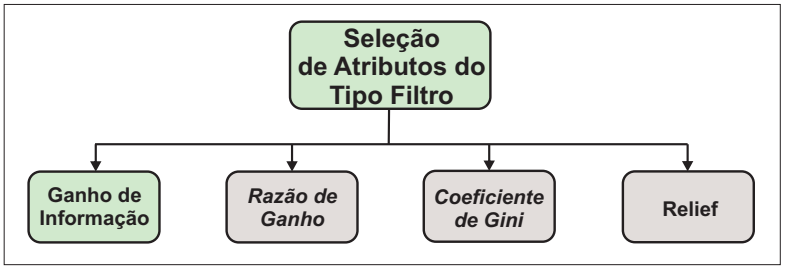

Figura 2.7: Métricas usada no processo de seleção de atributos do tipo filtro.

- *Ganho de Informação* (Quinlan, 1986): é baseado no conceito de *entropia*, que tem origem na área de teoria da informação. Dado um atributo *a*, cujo domínio é $\{a_1, a_2, ..., a_k\}, k \geq 1$, define-se a probabilidade $p_i, 1 \leq i \leq k$, de cada valor a_i do atributo como a razão entre o número de instâncias da base em que ocorre o valor a_i para o atributo *a* e o número total de instâncias. A entropia desse atributo, representada por $h(a)$ é dada por:

$$h(a) = -\sum_{i=1}^{k}[p_i * \log_2(p_i)], \tag{2.1}$$

  já a entropia da classe, representada por $h(c)$, pode ser calculada de forma semelhante a de um atributo, considerando p_j a razão entre o número de instâncias em que o valor c_i da classe, $1 \leq j \leq m$, ocorre na base e o número total de instâncias.

  Assim, seja a probabilidade $p_{j|i}$ a razão entre o número de instâncias da base que pertencem à classe c_j em que ocorre o valor a_i do atributo *a*, e o número total de instâncias da base, a entropia condicional da classe *c*, dado o atributo *a*, é expressa por:

$$h(c|a) = -\sum_{i=1}^{k}\sum_{j=1}^{m}[p_{j|i} * log\frac{(p_{j|i})}{p_i}]. \tag{2.2}$$

  Ressalta-se que quanto mais informação um atributo *a* tiver em relação à classe *c*, menor será a sua entropia condicional $h(c|a)$. Pelas equações 2.1 e 2.2 percebe-se que a entropia de *c* dado *a* nunca será maior que a entropia de *c* apenas, pois o conhecimento do atributo *a*

pode ser usado para determinar o valor de a. Logo, o ganho de informação de um atributo a em relação à classe pode ser calculado como a diferença entre a entropia da classe e a entropia condicional da classe dado o atributo a, indicado na equação:

$$gain(c|a) = h(c) - h(c|a). \tag{2.3}$$

- *Razão de Ganho* (Quinlan, 1986): a métrica de ganho de informação tende a superestimar a qualidade de atributos com muitos valores. Para evitar esse viés, a Razão de Ganho pondera a fórmula original, calculando a razão do ganho de informação do atributo a em relação à classe c e a entropia do atributo, como indicado na equação:

$$gain\ Ratio(c|a) = \frac{gain(c|a)}{h(a)} = \frac{h(c) - h(c|a)}{h(a)} \tag{2.4}$$

- *Coeficiente de Gini*(Kononenko, 1995): o coeficiente de *Gini* (*Gini index*) é uma medida estatística de desigualdade (ou impureza), comumente utilizada para calcular a desigualdade de distribuição de renda, embora seja aplicada a qualquer distribuição. O índice *Gini* é utilizado para a tarefa de seleção de atributos no algoritmo *CART* (Breiman, 1993). A ideia do algoritmo do *Gini index* é: deseja-se encontrar o atributo que contém a melhor participação de valores, considerando o coeficiente de índice *Gini*. Esse coeficiente é máximo (pior) quando as instâncias relativas a uma determinada participação de valores estão igualmente distribuídas entre todas as classes, e o coeficiente é mínimo (melhor) quando todas as instâncias pertencem a uma única classe. O cálculo do coeficiente *Gini* é expresso pela equação:

$$gini(c|a) = \sum_{i=1}^{k} \sum_{j-1}^{m} p_i * p_{j|i}^2 - \sum_{j=1}^{m} p_j^2, \tag{2.5}$$

onde $p_{j|i}$ é a probabilidade da instância pertencer à classe c_i $(1 \leq i \leq k)$.

- *Relief* (Kononenko, 1994): O algoritmo *Relief* foi proposto por Kira e L. (1992) para estimar a qualidade dos atributos de acordo com as dependências entre eles. As medidas apresentadas nesta seção assumem que os atributos são independentes entre si, e portanto não são aplicáveis em domínios ondem existem dependências fortes entre atributos.

  A ideia central do algoritmo *Relief* é estimar a qualidade dos atributos de acordo com a maneira como os seus valores distinguem entre as instâncias que estão próximas entre si. Para cada instância, o algoritmo busca por dois de seus vizinhos mais próximos: um que possui a mesma classe (chamado de *hit* mais próximo) e outro de classe diferente (chamado de *miss* mais próximo). Desse modo, o algoritmo é capaz de avaliar o quão bem os valores dos atributos distinguem as classes entre instâncias que são próximas umas das outras (Kononenko, 1995).

O funcionamento da versão original do *Relief* é: define-se um vetor de pesos $W \mid a \mid$, inicializado com o valor zero para cada atributo da base, e o valor t, que corresponde ao número de instâncias de treinamento que devem ser selecionadas aleatoriamente da base. Para cada uma dessas instâncias R, são encontrados o *hit* e o *miss* mais próximos, denominados H e M, respectivamente. Um *hit* é uma instância cuja classe é a mesma da instância selecionada R, enquanto um *miss* é uma instância cuja classe é diferente da classe de R. Para cada um dos atributos da instância, é ajustado o peso desse atributo de acordo com a soma das diferenças entre a instância selecionada e as instâncias *hit* e *miss*, de acordo com a equação:

$$W \mid a \mid = W \mid a \mid -\frac{dist(a, R, H)}{t} + \frac{dist(a, R, M)}{t}, \tag{2.6}$$

onde a função $dist$ é usada para calcular a diferença entre os valores de um atributo para duas instâncias, variando de 0 a 1. Para atributos discretos, são aplicados apenas os rótulos 0 (valores diferentes) e 1 (valores iguais). Após o termino do algoritmo, tem-se um vetor de pesos $W \mid a \mid$ atualizados para todos os atributos da base: quanto maior o peso, mais relevante o atributo.

2.3.3.(B) - Métodos de seleção de atributos do tipo *wrapper*

Os métodos do tipo *wrapper* utilizam o próprio algoritmo de classificação adotado como uma caixa preta para avaliar os subconjuntos de atributos de acordo com a sua capacidade preditiva (Guyon e Elisseff, 2003).

Assim como as abordagens do tipo filtro que avaliam subconjuntos de atributos, as abordagens *wrapper* precisam realizar uma busca entre os possíveis subconjuntos a serem avaliados. Nos filtros os subconjuntos são avaliados por meio de uma métrica; já nos *wrappers* o algoritmo de classificação é executado para cada subconjunto e a avaliação geralmente é feita em termos da acurácia preditiva retornada pelo algoritmo.

Wrappers geralmente produzem resultados melhores (em termos de acurácia preditiva de classificação) do que filtros, porque a seleção de atributos é guiada pelo próprio algoritmo de classificação que será usado em seguida (Hall, 2000). Porém, como esse algoritmo é executado diversas vezes para subconjuntos distintos de atributos, *wrappers* tendem a ter um custo computacional muito elevado e se tornarem inviáveis em bases de dados com um número grande de atributos.

Existem vários algoritmos de seleção de atributos do tipo *wrapper* que além de empregarem diferentes classificadores para avaliar o subconjunto de atributos também possuem critérios de parada e estratégias de busca distintos (Liu e Yu, 2005).

Alguns dos primeiros trabalhos com essa abordagem de seleção de atributos focaram nos classificadores de Árvores de Decisão, analisando diferentes estratégias de busca para selecionar o subconjunto de atributos candidatos, além de comparar o impacto de cada estratégia no comportamento final do indutor da árvore (Doak, 1992).

O algoritmo OBVILION, proposto em (Langley, 1994), combina a abordagem *wrapper* com o método de classificação k-NN. Na seleção de atributos do OBVILION, é feita uma busca conhecida como *backward elimination*, que começa a execução com todos os atributos e retira gradualmente aqueles julgados menos relevantes, ou seja, que não fazem com que o valor da acurácia estimada diminua (Blum e Langley, 1997). Esse método de seleção de atributos é considerado um *wrapper*, pois envolve a execução do algoritmo k-NN para uma parte dos dados de treinamentos, utilizando um sistema de validação cruzada *leave-one-out* (Han e Kamber, 2006) para estimar a acurácia de cada subconjunto de atributos.

Outros algoritmos foram desenvolvidos em conjuntos com o método k-NN, mas utilizando-se estratégias distintas para compor o subconjunto de atributos a ser avaliado, como: buscas aleatórias, *hill climbing* e algoritmos genéticos (Blum e Langley, 1997).

Também existem estratégias híbridas que tentam combinar as vantagens das abordagens filtro e *wrapper* (Liu e Yu, 2005). Uma técnica muito utilizada é o uso de filtros para reduzir o conjunto inicial de atributos, viabilizando o uso de um *wrapper* em seguida.

2.3.3.(C) - Métodos de seleção do tipo *Embedded*

Os métodos *Embedded* (Embutidos) selecionam o conjunto de atributos no próprio processo de construção do modelo de classificação durante a fase de treinamento e não são geralmente específicos para um dado algoritmo de classificação (Guyon e Elisseff, 2003).

Para exemplificar uma abordagem que utiliza um método de seleção do tipo *Embedded* consideramos o exemplo da Árvore e Decisão.

As Árvores de Decisão podem ser vistas como detentoras de uma técnica de seleção de atributos incorporada ao seu algoritmo de indução. Este algoritmo seleciona internamente um subconjunto de atributos para compor a árvore, utilizando em cada etapa uma função para avaliar qual é o atributo com a maior capacidade de discriminação da classe (Blum e Langley, 1997). Exemplos de Árvores de Decisão que utilizam métodos de particionamento para a indução do modelo de classificação são: ID3 (Quinlan, 1983), C4.5 (Quinlan, 193) e CART (Breiman, 1993).

2.3.3.(D) - Construção de Atributos

Atributos fracamente, indiretamente ou condicionalmente relevantes podem ser individualmente inadequados, entretanto, esses atributos podem ser convenientemente combinados gerando novos atributos que podem mostrar-se altamente representativos para a descrição de um conceito. O processo de construção de novos atributos é conhecido como *construção de atributos* ou *construção indutiva* (Bloedorn e Michalski, 1998).

Assim, construção de atributos é o processo de composição de atributos ditos primitivos, produzindo-se novos atributos possivelmente relevantes para a descrição de um conceito.

De uma forma bastante ampla, o processo de indução construtiva pode ser dividido em duas abordagens: a automática e a guiada pelo usuário. A *indução automática* consiste em um processo de construção de atributos guiada automaticamente pelo método de construção. Geralmente, os atributos construídos são avaliados em relação aos dados, e podem ser descartados ou integrados ao conjunto de dados. A *indução guiada pelo usuário* utiliza o conhecimento do usuário ou especialista no domínio para guiar a composição dos atributos (Batista, 2003).

2.4 Transformação dos Dados

O objetivo da etapa de transformação de dados é modificar a forma com que os dados estão representados, pois os algoritmos utilizados na fase de Mineração de Dados não são capazes de analisar certos tipos de dados como do tipo data e hora, por exemplo. Desta forma, frequentemente atributos com esses tipos de dados são transformados em outro atributo com a mesma informação, mas com um formato que os algoritmos sejam capazes de tratar. Um atributo do tipo data e hora pode ser transformado em um atributo do tipo inteiro, o qual representa o número de dias decorridos a partir de uma data fixa (Batista, 2003). Segundo Batista (2003) algumas das transformações de dados mais comuns são: normalização de dados, transformação de atributos qualitativos em quantitativos e tratamento de tipos de dados complexos (ver Figura 2.8).

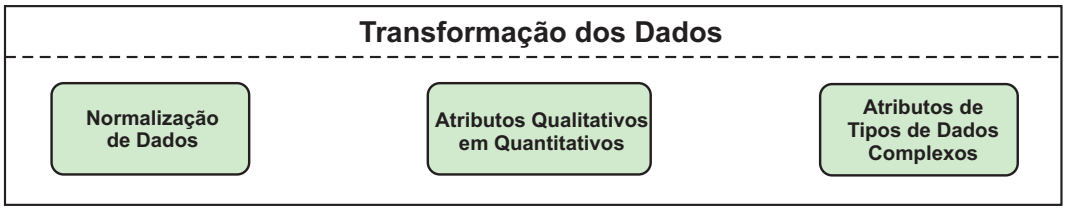

Figura 2.8: Principais transformações de dados.

Normalização de Dados

Segundo Han e Kamber (2006), a normalização consiste em ajustar as faixas de valores dos atributos para o mesmo intervalo, tais como: de -1 a 1 ou de 0 a 1. Esta técnica possui grande utilidade quando os atributos possuem grande intervalo de valores, sendo de grande utilidade para algoritmos de classificação, principalmente os baseados em distância. As principais formas de normalização são: min-max, Z-score e escalonamento decimal.

Na normalização min-max, o valor normalizado ($v^{'}$) pode ser encontrado através da Equação:

$$v^{'} = \frac{v - min_A}{max_A - min_A}, \tag{2.7}$$

onde v é o valor original para o atributos A e min_A e max_A sao respectivamente o menor e o maior valor encontrados para o atributo A.

Na normalização Z-score, o valor normalizado ($v^{'}$) pode ser encontrado através da Equação:

$$v^{'} = \frac{v - media_A}{desvio_A}, \tag{2.8}$$

onde v é o valor original para o atributos A e $media_A$ e $desvio_A$ sao respectivamente a média e o desvio padrão encontrados para os valores do atributo A.

Na normalização escalonamento decimal, o valor normalizado ($v^{'}$) pode ser encontrado através da Equação:

$$v^{'} = \frac{v}{10^j}, \tag{2.9}$$

onde v é o valor original para o atributos A e j é o menor inteiro tal que $Max(|v^{'}|) < 1$.

Transformação de Atributos Qualitativos em Quantitativos

Alguns algoritmos não são capazes de manipular atributos qualitativos que possuem apenas valores não numéricos. Dessa forma, é necessário converter os atributos qualitativos em atributos quantitativos. Existem diversas abordagens para realizar essa transformação dependendo das características e limitações de cada algoritmo. Atributos qualitativos em ordem, tal como *pequeno*, *médio* e *grande*, podem ser mapeados para valores numéricos de forma a manter a ordem dos valores, por exemplo *pequeno* = 1, *médio* = 2 e *grande* = 3. Porém, atributos qualitativos sem ordem inerente, tais como *verde*, *amarelo*, *azul* e *vermelho*, quando mapeados para números criam uma ordem nos valores do atributos que não é real. Portanto, uma forma de mapeamento destes atributos é através da criação de bits de representação. Por exemplo, para mapear o atributo do tipo cor seriam criados dois bits, assim estes poderiam representar até quatro valores diferentes: verde = 00, amarelo = 01, azul = 10 e vermelho = 11 (Batista, 2003).

Atributos de Tipos de Dados Complexos

Um atributo possui tipo de dado complexo quando ele não é considerado qualitativo nem quantitativo, como por exemplo data e hora. Esses atributos não são assimilados pela maioria dos algoritmos de classificação. Assim, atributos com esta característica devem ser tratados a fim de otimizarem o desempenho do modelo. No caso especifico dos tipos de data e hora, a escolha mais simples é pela conversão para o tipo inteiro. Isto pode ser feito calculando-se a diferença em dias, meses, ou qualquer outra unidade de tempo, entre os valores das datas do atributo em questão e uma data fixa. Por exemplo, um atributo *data de nascimento* pode ser convertido para *idade* calculando-se a diferença entre os valores do atributo *data de nascimento* e *data atual*.

2.5 Mineração de Dados

Segundo Ribeiro (2004) o objetivo da fase de Mineração de Dados é a verificação de uma hipótese ou obtenção de novos padrões sobre os dados. Assim, as tarefas de mineração podem ser de *predição* ou *descrição*. Enquanto as tarefas de predição constroem modelos para predizer o comportamento de dados futuros, as tarefas de descrição revelam padrões e propriedades presentes nos dados analisados. Para a aplicação de algoritmos de mineração sobre os dados é necessária a definição de dois fatores: *Atributos relevantes para análise* e *Tarefa de mineração adequada*.

Atributos relevantes para análise: frequentemente, a análise de todos os atributos de uma base de dados resulta em uma grande quantidade de padrões desinteressantes e sem significado. Assim, recomenda-se que somente os atributos de interesse sejam previamente selecionados para serem submetidos ao processo de mineração. A seleção de atributos relevantes é feita na fase de pré-processamento através de técnicas de seleção de atributos (ver Seção 2.3.3).

Tarefa de mineração adequada: existe uma grande variedade de tarefas que podem ser realizadas na fase de Mineração de Dados: associação, classificação, regressão, predição e agrupamento (ver Figura 2.9). Assim, é importante que o objetivo do processo de DCBD esteja bem definido para que sejam utilizadas as técnicas de mineração mais adequadas ao cumprimento do objetivo deste processo.

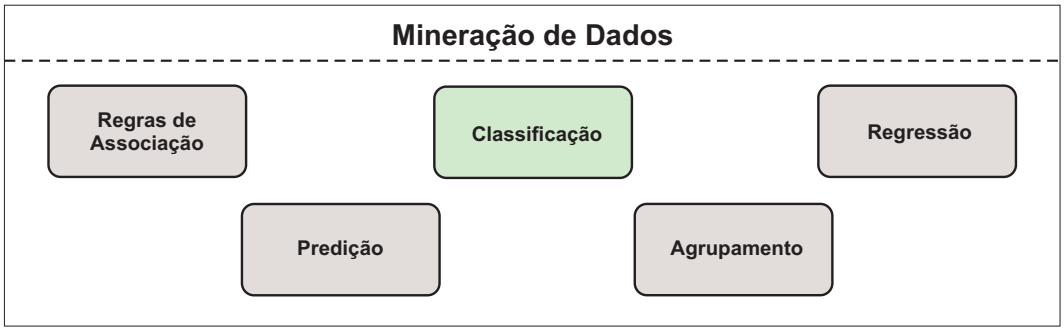

Figura 2.9: Principais tarefas da fase de MD.

Regras de Associação

A tarefa de associação permite relacionar a ocorrência de um determinado conjunto de itens. Uma regra de associação é da forma $X \rightarrow Y$, onde X e Y são conjuntos de itens, significando que se X ocorre em uma transação da base de dados, Y também tende a ocorrer. A análise da associação em um banco de dados pode gerar uma grande quantidade de regras de associação. Algumas dessas regras podem não ser interessante, pois ocorrem com baixa frequência nos dados. Para contornar esse problema devem ser definidos parâmetros (medidas de interesse) que determinam quais associações são interessantes ou não para o usuário. Assim, para uma regra ser considerada satisfatória ela deve satisfazer alguma medida de interesse (Ribeiro, 2004).

Classificação

A tarefa de classificação permite agrupar dados em uma hierarquia de classes de acordo com os valores de seus atributos. Os registros agrupados em classes são formados por um atributos alvo (ou atributo classe) que determina a classe do registro, e um conjunto de atributos de predição. O objetivo é descobrir as relações existentes entre os atributos de predição e o atributo alvo, utilizando registros cuja classificação é conhecida. A tarefa de classificação é portanto uma tarefa de predição, uma vez que ela prediz os valores dos atributos alvo (Ribeiro, 2004). Árvores de Decisão, Redes Neurais, Raciocínio Baseado em Exemplos e algoritmos Probabilísticos são técnicas bastantes utilizadas para o processo de classificação (Camargo, 2002).

Na Figura 2.10 é ilustrado o processo de classificação. A mineração de regras de classificação é feito em duas etapas. Na primeira etapa um algoritmo de classificação é aplicado sobre uma amostra do BD, que é chamada de conjunto de treinamento. Esse conjunto contém registros do banco cuja classificação é conhecida, ou seja, apresenta dados com o valor do atributo alvo preenchido. O resultado da execução da primeira etapa é um conjunto de regras de classificação. Na segunda etapa, as regras de classificação mineradas anteriormente são utilizadas para classificar a base de dados (Ribeiro, 2004).

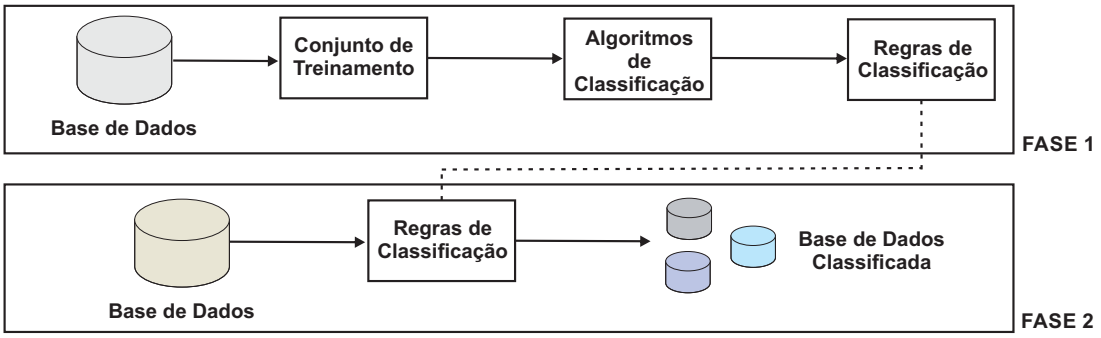

Figura 2.10: As duas fases do processo da classificação.

Regressão

O processo de regressão é semelhante ao de classificação, a principal diferença entre eles é que a classificação lida com valores discretos enquanto que e regressão com valores contínuos. Como consequência disso temos que através do processo de regressão é possível ordenar registros individualmente. Por exemplo, se pelo processo de classificação só é possível classificar registros como 0 ou 1, pelo processo de regressão é possível classificarmos registros com qualquer valor real entre 0 e 1 (Camargo, 2002).

Predição

O processo de predição também é semelhante aos processos anteriores, exceto pelo fato de que os registros possuem dados temporais e serão classificados de acordo com alguma predição de comportamento futuro ou predição de algum valor futuro. Tanto classificação como regressão podem ser adaptadas para uso em predição através dos exemplos de treinamento onde os valores passados das variáveis a serem preditas são conhecidos, de acordo com os dados históricos (que são utilizados para a construção do modelo que explica o comportamento corrente observado) para estes exemplos (Camargo, 2002).

Agrupamento (*Clustering*)

O processo de agrupamento consiste em dividir uma população heterogênea em um número de subgrupos mais homogêneos. Estes grupos não são pré-definidos e também não há exemplos já rotulados, assim como acontece na classificação (Camargo, 2002).

Considerando que os dados podem ser representados através de pontos, o agrupamento permite agrupar esses pontos em partições de acordo com a distância entre eles. A distância entre dois pontos indica o quanto eles são similares, com base nos valores de determinados atributos. A distância entre os pontos do mesmo grupo deve ser sempre menor que a distância entre pontos de grupos distintos (Ribeiro, 2004).

2.6 Avaliação e Interpretação dos Resultados

Após a fase de Mineração de Dados, o processo de DCBD entra na etapa de avaliação e interpretação dos resultados e esta etapa deve envolver todos os participantes do processo. O analista de dados tenta descobrir se os algoritmos utilizados no processo atingiram as expectativas, avaliando os resultados de acordo com algumas métricas, como taxa de acerto por exemplo. O especialista no domínio irá verificar a compatibilidade dos resultados usando o seu conhecimento sobre o problema em questão. E, por fim, é de responsabilidade do usuário dar o julgamento final sobre a aplicabilidade do processo de DCBD (Batista, 2003). A Figura 2.11 ilustra algumas formas de avaliação dos resultados, tais como: avaliação dos algoritmos utilizados e extração das regras obtidas pelos classificadores e avaliação do conhecimento.

Do ponto de vista de classificação, as métricas de avaliação utilizadas neste trabalho foram: Precisão, *Recall*, Acurácia, *F-Measure*, Área Sob a Curva ROC e Índice *Kappa*. Todas essas medidas são detalhadas no Capítulo 5.

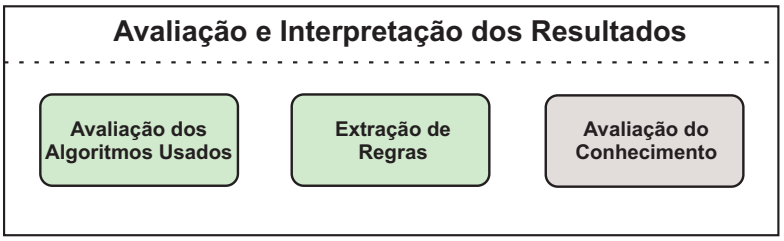

Figura 2.11: Principais tarefas da fase de avaliação e interpretação dos resultados.

2.7 Considerações Finais

Neste capítulo foi apresentado o embasamento teórico para o auxílio a compreensão do processo de Descoberta do Conhecimento em Base de Dados. Também foram apresentadas as 5 fases desse processo: coleta, pré-processamento, transformação, mineração e avaliação e interpretação dos resultados; juntamente com as principais tarefas e técnicas encontradas em cada uma dessas fases. A Figura 2.12 sintetiza o apresentado neste capítulo.

Para maior detalhes sobre os temas abordados a sugerimos: Descoberta do Conhecimento em Base de Dados - Batista (2003), Ribeiro (2004) e Han e Kamber (2006); Fases do processo de DCBD - Fayyad et al. (1996) e Han e Kamber (2006); Técnicas de pré-processamento de dados - Neves (2003); Processo de seleção de atributos - Pereira (2009); Medidas para avaliação de classificadores e extração de regras - Gomes (2002b) e Chimieski e Fagundes (2013).

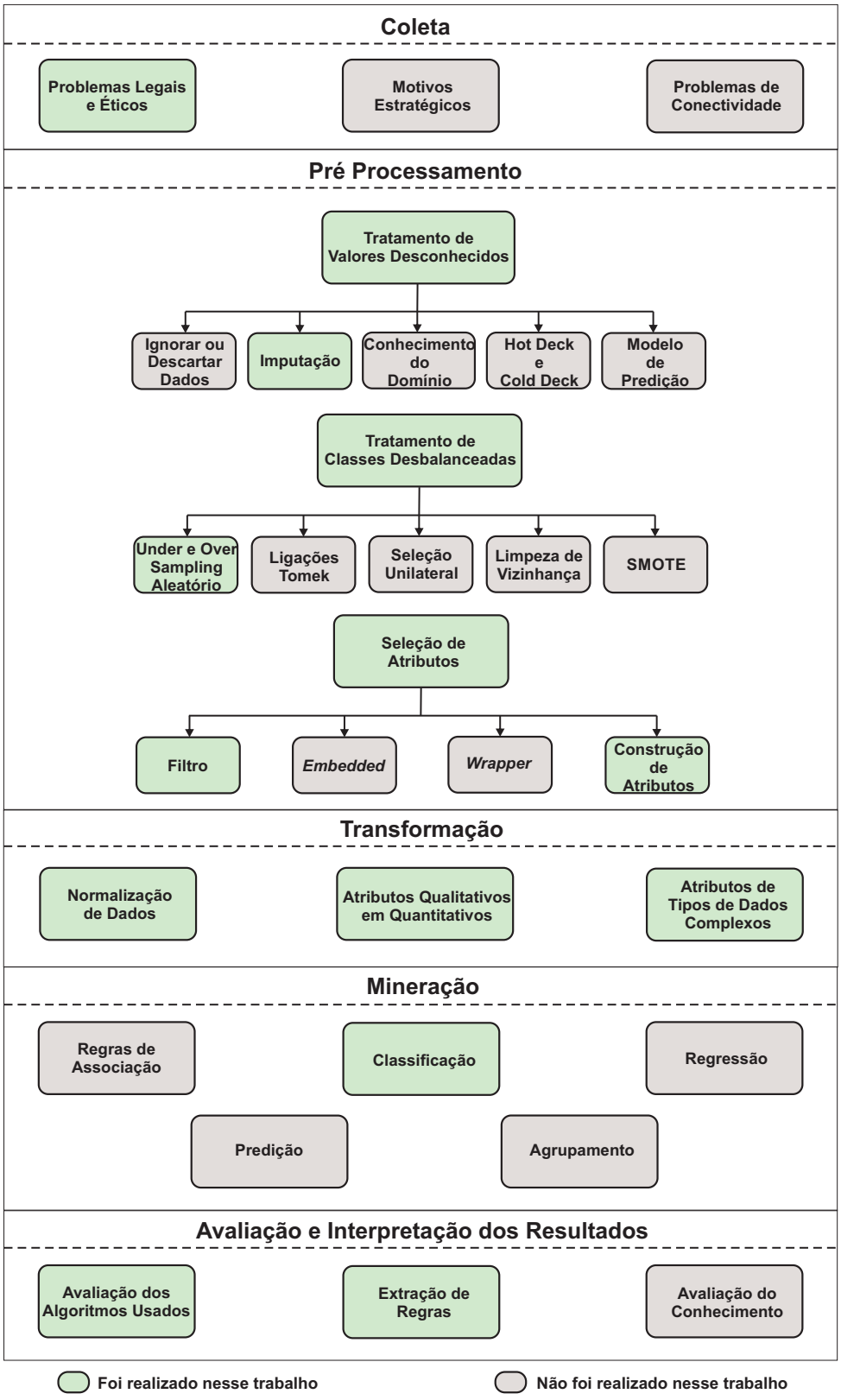

Figura 2.12: Fases do processo de DCBD juntamente com os principais problemas encontrados em cada fase.