UNIVERSIDADE FEDERAL FLUMINENSE

RAFAEL BARROS PEREIRA

SELEÇÃO LAZY DE ATRIBUTOS PARA A TAREFA DE CLASSIFICAÇÃO

NITERÓI

2009

UNIVERSIDADE FEDERAL FLUMINENSE

RAFAEL BARROS PEREIRA

SELEÇÃO LAZY DE ATRIBUTOS PARA A TAREFA DE CLASSIFICAÇÃO

Dissertação de Mestrado submetida ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Mestre. Área de concentração: Inteligência Artificial.

Orientador:
Bianca Zadrozny

Co-orientador:
Alexandre Plastino de Carvalho

NITERÓI

2009

SELEÇÃO LAZY DE ATRIBUTOS PARA A TAREFA DE CLASSIFICAÇÃO

Rafael Barros Pereira

Dissertação de Mestrado submetida ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Mestre.

Aprovada por:

Bianca Zadrozny, Ph.D. / IC-UFF
(Presidente)

Alexandre Plastino de Carvalho, D.Sc. / IC-UFF

Ana Cristina Bicharra Garcia, Ph.D. / IC-UFF

Wagner Meira Jr., Ph.D. / DCC-UFMG

Nelson Francisco Favilla Ebecken, D.Sc. / COPPE-UFRJ

Niterói, 07 de julho de 2009.

Agradecimentos

A MINHA FAMÍLIA, pelo apoio constante e ajuda desde o início do mestrado.

AOS COLEGAS DE TRABALHO, que participaram e incentivaram o longo percurso até a conclusão do curso, em especial aos amigos do BANCO CENTRAL: José Roberto, Guilherme, André Castanheira, Gabriela, Miriam e Manuel; e aos estimados companheiros do BNDES: Carmen, Luciano, João Marcos, Cássio, Nicolaas, José Marcos, Marcelo, Eduardo, Fabiano, Luis Filipe, Vanessa e Ricardo.

AOS MEUS ORIENTADORES, Bianca Zadrozny e Alexandre Plastino, pela paciência, compreensão e competência.

AOS PROFESSORES do curso, em especial à Simone, Satoru e Celso, pelos valiosos ensinamentos e oportunidades de aprendizado, e também aos professores da UERJ, Paulo Eustáquio, Vera Werneck, Eduardo Galúcio e David, pelas mesmas razões.

AOS COLEGAS Luiz Merschmann e Alex Freitas, pela cooperação, companherismo e troca de idéias.

A TODOS que de uma maneira ou de outra contribuíram para que este trabalho pudesse ser concluído.

Resumo

Seleção de atributos é tradicionalmente uma etapa de pré-processamento dos dados que visa identificar atributos relevantes para a tarefa de classificação. O seu objetivo é remover atributos da base de dados que não contribuam para a classificação ou que possam prejudicar a capacidade preditiva do classificador.

Neste trabalho, é proposta uma nova estratégia de seleção de atributos – baseada em uma abordagem *lazy* – que adia a identificação de atributos relevantes até o momento em que uma instância é submetida ao classificador. Essa estratégia é fundamentada na hipótese de que considerar os valores de cada atributo de uma instância a ser classificada pode contribuir para a identificação dos melhores atributos para a correta classificação dessa instância específica. Resultados experimentais com 40 bases de dados utilizando os classificadores k-NN, Naive Bayes e HiSP confirmam que atrasar a seleção de atributos até o momento da classificação pode ser uma boa estratégia.

Palavras-chave: seleção de atributos, técnicas de classificação, mineração de dados.

Abstract

Attribute selection is a data preprocessing step which aims at identifying relevant attributes for the classification task. It attempts to remove from the data set attributes which do not contribute to, or that can even reduce, the predictive ability of a classifier.

In this work, we propose a new attribute selection strategy – based on a lazy approach – which postpones the identification of relevant attributes until an instance is submitted for classification. Our strategy relies on the hypothesis that taking into account the attribute values of an instance to be classified may contribute to identifying the best attributes for the correct classification of that particular instance. Comprehensive experimental results for 40 datasets using the k-NN, Naive Bayes and HiSP classifiers show that, indeed, delaying attribute selection to classification time can be a good strategy.

Keywords: attribute selection, classification, data mining.

Glossário

CBA	:	*Classification Based on Association*;
CFS	:	*Correlation-based Feature Selection*;
HiSP	:	***Hi**ghest **S**ubset **P**robability*;
IC	:	Instituto de Computação;
k-NN	:	*k-Nearest Neighbor*;
NB	:	*NaiveBayes*;
SVM	:	*Support Vector Machines*;
UCI	:	*University of California, Irvine*;
UFF	:	Universidade Federal Fluminense;
WEKA	:	*Waikato Environment for Knowledge Analysis*;

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Mineração de dados é o processo de descoberta de informações relevantes a partir de um grande conjunto de dados [18, 38]. A evolução dos computadores e meios de transmissão de informações digitais possibilitou o acúmulo de grandes quantidades de dados, que por sua vez motivou o desenvolvimento de conceitos e técnicas automatizadas de mineração de dados.

Dentre as tarefas da área de mineração de dados, destaca-se a tarefa de classificação, que tem sido alvo de grande esforço de estudo e pesquisa devido à sua aplicabilidade e sucesso em diversos domínios. Representam importantes aplicações da tarefa de classificação: detecção de fraudes, diagnóstico médico, estimativa de desempenho, entre outras com objetivos preditivos nas áreas de educação, finanças e biologia molecular [18, 38].

A partir de uma base de dados na qual cada instância é caracterizada por um conjunto de atributos e pela classe à qual pertence, o problema da classificação consiste em estimar a classe à qual pertence uma nova instância a partir dos valores de seus atributos.

As técnicas de classificação são tradicionalmente categorizadas como *eager* ou *lazy*. As estratégias de classificação do tipo *eager* utilizam um conjunto de dados de treinamento para construir um modelo de classificação que define um mapeamento de instâncias para classes. No momento da classificação, este modelo é utilizado para predizer a qual classe pertence uma nova instância. Entre os classificadores do tipo *eager* mais conhecidos estão as árvores de decisão [31, 32], as redes neurais [34], os classificadores associativos [23] e a técnica SVM (*Suport Vector Machines*) [5].

Estratégias *lazy*, por outro lado, não constroem um modelo de classificação explícito e adiam a maior parte do processamento dos dados de treinamento até o momento em que se conheça a instância a ser classificada. Uma das técnicas de classificação *lazy*

mais conhecidas é o algoritmo k-NN (*k-Nearest Neighbors*) [6, 7], no qual a classe de uma instância é estimada a partir da informação das classes dos seus k vizinhos mais próximos (elementos mais semelhantes). Outro algoritmo que pode ser utilizado como um classificador *lazy* é o Naive Bayes [11], que se baseia no teorema de Bayes, pois o cálculo das probabilidades condicionais pode ser adiado até o momento da classificação. Além desses, também existe uma versão *lazy* das árvores de decisão [13], em que uma árvore é construída apenas no momento em que uma nova instância é submetida à classificação, e também algoritmos *lazy* de classificação associativa [36], em que regras de classificação são extraídas apenas no momento da classificação.

Um dos maiores desafios relacionados à classificação é desenvolver algoritmos que sejam eficientes, precisos e capazes de lidar com bases de dados que contenham um grande número de atributos e instâncias. Sabe-se que o desempenho de técnicas de classificação está diretamente relacionado, entre outros fatores, à qualidade dos dados da base de treinamento. Atributos redundantes e irrelevantes podem não somente prejudicar a acurácia de um classificador, mas também tornar o processo de construção do modelo ou a execução do algoritmo de classificação mais lento.

A fim de tentar evitar esses problemas, são utilizadas técnicas de seleção de atributos, que visam eliminar da base de treinamento atributos que não contribuem ou mesmo prejudicam o desempenho das estratégias de classificação [16, 24]. Tradicionalmente, técnicas de seleção de atributos são executadas na fase de pré-processamento, ou preparação, dos dados e suas decisões são definitivas para a fase de construção do modelo ou classificação propriamente dita. A construção de um modelo ou o processo de classificação propriamente dito são realizados, portanto, a partir da base de dados reduzida.

Neste trabalho, propõe-se uma nova abordagem de seleção de atributos, cuja característica principal é adiar a seleção dos atributos relevantes – de forma *lazy* – ao momento em que uma instância for submetida ao processo de classificação, em vez de se fazer a seleção previamente. Tem-se como hipótese para essa proposta que o conhecimento dos valores dos atributos da instância a ser classificada pode contribuir para a identificação dos melhores atributos para aquela instância em particular.

Dessa forma, para diferentes instâncias a serem classificadas, subconjuntos distintos de atributos poderão ser selecionados. Supõe-se que o conhecimento das características particulares de uma instância ajude na escolha do subconjunto de atributos mais adequado e, consequentemente, permita que o classificador atinja melhores valores de acurácia preditiva. Cabe ressaltar que por adiar a seleção dos atributos ao momento da classificação,

a abordagem proposta é direcionada para classificadores também do tipo *lazy*.

O restante desta dissertação está organizado como especificado a seguir. O Capítulo 2 contém uma revisão sobre a tarefa de seleção de atributos. O Capítulo 3 apresenta a proposta central desta dissertação – a abordagem *lazy* de seleção de atributos –, além de descrever uma instanciação dessa abordagem que utiliza o critério de entropia para ordenar os atributos a serem selecionados. A implementação dessa estratégia e os resultados obtidos são relatados no Capítulo 4. Por fim, o Capítulo 5 apresenta as conclusões deste trabalho e propostas de trabalhos futuros.

Capítulo 2

Técnicas de Seleção de Atributos

2.1 Introdução

Um problema importante na área de mineração de dados é a utilização de bases de dados com um grande número de atributos. A tarefa de seleção de atributos é uma das maneiras utilizadas para lidar com esse problema. Trata-se de um procedimento em que um subconjunto dos atributos da base de dados é selecionado para aplicação de um algoritmo de mineração. Nesta dissertação, será abordado o tópico de seleção de atributos especificamente para a tarefa de classificação.

De maneira geral, nem todos os atributos da base de dados são necessários para discriminar a classe de maneira precisa, e incluí-los no modelo de classificação pode até mesmo gerar resultados inferiores do que seriam obtidos se eles fossem removidos da base [9].

De acordo com [15], as técnicas de seleção de atributos são a princípio empregadas para identificar atributos relevantes e com informações essenciais. Em geral, além desse objetivo principal, existem outras importantes motivações: o aperfeiçoamento da acurácia preditiva do classificador, a redução e simplificação do conjunto de dados, a maior agilidade na tarefa de classificação e a simplificação do modelo de classificação gerado. Neste trabalho, a principal motivação da estratégia de seleção de atributos proposta é melhorar a acurácia preditiva do classificador.

Com esse objetivo, o subconjunto selecionado deve conter idealmente o menor subconjunto de atributos que levam à melhor acurácia possível. Em outras palavras, deseja-se selecionar o subconjunto de atributos que contenha o total ou grande parte das informações disponíveis na base de dados que sejam relevantes para a tarefa de classificação.

Do ponto de vista teórico, pode ser demonstrado que uma seleção de atributos ótima para problemas de classificação requer uma busca exaustiva de todos os possíveis subconjuntos de atributos [33], tornando-se impraticável quando o número de atributos é muito alto. Por esse motivo foram desenvolvidas diferentes técnicas de seleção de atributos, que utilizam critérios de seleção e algoritmos de busca distintos para avaliar e encontrar de forma heurística o subconjunto de atributos mais adequado.

Essas técnicas de seleção de atributos geralmente são divididas em três categorias: *embedded*, *wrapper* e filtro [25].

As estratégias do tipo *embedded* são diretamente incorporadas no algoritmo responsável pela indução do modelo de classificação. As estratégias *wrapper* e filtro são executadas em uma fase de pré-processamento dos dados, e procuram pelo conjunto de atributos mais adequado para ser utilizado pelo algoritmo de classificação ou no processo de indução do modelo de classificação.

Em linhas gerais, a diferença entre algoritmos de seleção de atributos do tipo filtro e do tipo *wrapper* é que na primeira o subconjunto de atributos é avaliado por uma medida independente, enquanto na última é utilizado o próprio algoritmo de classificação para essa avaliação.

Em geral, as técnicas *wrapper* selecionam subconjuntos de atributos que atingem uma acurácia preditiva maior que as estratégias do tipo filtro, já que elas avaliam os atributos utilizando o mesmo algoritmo que será utilizado na fase de classificação. Contudo, como as estratégias *wrapper* necessitam de várias execuções do algoritmo de classificação, os seus custos computacionais tendem a ser bem maiores que os custos das estratégias do tipo filtro.

Com o objetivo de fazer uma revisão das técnicas de seleção de atributos mais conhecidas ou relevantes para este trabalho, serão delineadas nas Seções 2.2, 2.3 e 2.4, as técnicas que se enquadram em cada uma das categorias descritas anteriormente.

2.2 Técnicas de Seleção de Atributos do Tipo Filtro

Técnicas de seleção de atributos do tipo filtro selecionam o subconjunto de atributos na fase de pré-processamento, de forma independente do classificador utilizado [14].

O nome "filtro" deriva da ideia de que os atributos irrelevantes são *filtrados* da base de dados antes da aplicação do algoritmo de classificação [2]. Os filtros usam as informações

da própria base de treinamento para escolher os atributos a serem utilizados posteriormente.

Técnicas do tipo filtro podem avaliar os atributos individualmente e escolher os melhores, como exemplificado pelas técnicas *Information Gain Attribute Ranking* [40] e *Relief* [19, 20]; ou podem avaliar subconjuntos de atributos, buscando de forma heurística o melhor subconjunto. As técnicas mais conhecidas desse último grupo são a *Correlation-based Feature Selection* [17] e a *Consistency-based Feature Selection* [26].

2.2.1 Técnicas de Ranqueamento de Atributos

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.

Uma maneira de se medir a qualidade de um atributo para classificação é avaliar o seu grau de associação com a classe. Para determinar esse tipo de associação, foram propostas diversas métricas, que serão descritas a seguir. Essas métricas são empregadas em uma base de dados $D(A_1, A_2, ..., A_n, C)$, $n \geq 1$, com $n+1$ atributos, onde C é o atributo classe e o seu domínio é $\{c_1, c_2, ..., c_m\}$, $m \geq 2$. Assume-se que os valores dos atributos e da classe são discretos.

- *Ganho de Informação*: é baseado no conceito de *entropia* [31], 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:

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

  A entropia da classe, representada por $H(C)$, pode ser calculada da mesma forma, considerando p_j a razão entre o número de instâncias em que o valor c_j da classe, $1 \leq j \leq m$, ocorre na base e o número total de instâncias.

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 calculada usando a fórmula:

$$H(C|A) = -\sum_{i=1}^{k}\sum_{j=1}^{m}[p_{j|i} * log(\frac{p_{j|i}}{p_i})] \tag{2.2}$$

Quanto mais informativo um atributo A for em relação à classe C, menor será a sua entropia condicional $H(C|A)$. Pelas Fórmulas 2.1 e 2.2, verifica-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 C. 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 fórmula:

$$GANHO(C|A) = H(C) - H(C|A) \tag{2.3}$$

- *Gain Ratio*: a métrica de ganho de informação tende a superestimar a qualidade de atributos com muitos valores. Para evitar esse viés, a métrica *Gain Ratio*[31] 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 fórmula:

$$GainRatio(C|A) = \frac{GANHO(C|A)}{H(A)} = \frac{H(C) - H(C|A)}{H(A)} \tag{2.4}$$

- *Incerteza Simétrica*: a métrica de incerteza simétrica [39] (*Symmetrical Uncertainty*) é uma outra tentativa de normalizar o ganho de informação de um atributo em relação à classe, por meio da fórmula:

$$SymmU(C|A) = 2 * \frac{GANHO(C|A)}{H(A) + H(C)} = 2 * \frac{H(C) - H(C|A)}{H(A) + H(C)} \tag{2.5}$$

- *Chi-Square*: a métrica *Chi-Square* (ou χ^2) avalia a qualidade de um atributo de acordo com a sua correlação com a classe, por meio de um teste estatístico χ^2. Para cada valor a_i do atributo A ($1 \leq i \leq k$) e para cada valor c_j da classe C ($1 \leq j \leq m$), existe uma frequência esperada quando ($A = a_i$) e ($C = c_j$), que pode ser calculada pela fórmula [18]:

$$e_{ij} = \frac{count(A = a_i) * count(C = c_j)}{N}, \tag{2.6}$$

onde N é o número total de instâncias, $count(A = a_i)$ é o número de instâncias em que ocorre o valor a_i do atributo A, e $count(C = c_j)$ é o número de instâncias que pertencem à classe c_j. A partir da frequência esperada de todas as combinações de valores i e j, pode-se calcular a métrica χ^2 pela fórmula:

$$\chi^2 = \sum_{i=1}^{k} \sum_{j=1}^{m} [\frac{o_{ij} - e_{ij}}{e_{ij}}], \tag{2.7}$$

onde o_{ij} é a frequência observada da combinação $(A = a_i)(C = c_j)$, ou seja, a razão entre o número de instâncias em que ocorre o valor i do atributo A simultaneamente com o valor j da classe C, e o número total de instâncias da base. O teste estatístico pode ser aplicado para determinar se o atributo A e a classe C são independentes, usando o grau de liberdade $(k - 1) * (m - 1)$, que é proporcional ao número de ocorrências distintas de valores em A multiplicado pela cardinalidade da classe C. Se essa hipótese puder ser rejeitada, de acordo com um nível de significância estatística pré-determinado e uma distribuição χ^2, significa que o atributo é fortemente relacionado à classe [18].

- *Gini index*: o coeficiente de *Gini* é uma medida estatística de desigualdade (ou impureza), comumente utilizada para calcular a desigualdade de distribuição de renda, embora seja aplicável a qualquer distribuição. O índice *gini* é utilizado para a tarefa de seleção de atributos no algoritmo CART [22] de árvores de decisão. A idéia do algoritmo do *Gini Index* é a seguinte: deseja-se encontrar o atributo que contém a melhor partição de valores, considerando o coeficiente de índice *gini*. Esse coeficiente é máximo (pior) quando as instâncias relativas a uma determinada partição de valores estão igualmente distribuídas entre todas as classes, e o coeficiente é mínimo (melhor) quando todas as instâncias pertencem a uma única classe. Os atributos com as melhores partições de valores, entre os subconjuntos de valores possíveis, apresentam um coeficiente de *gini* baixo, que é dado por [21]:

$$Gini(C|A) = \sum_{i=1}^{k} \sum_{j=1}^{m} p_i * p_{j|i}^2 - \sum_{j=1}^{m} p_j^2, \tag{2.8}$$

onde $p_{j|i}$ é a probabilidade da instância pertencer à classe c_j $(1 \leq j \leq m)$ quando ocorre o valor a_i do atributo A $(1 \leq i \leq k)$.

- *Relief* e *ReliefF*: O algoritmo *Relief* foi proposto por [19] 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 onde existem dependências fortes entre atributos [20].

  A idéia 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). Deste modo, o algoritmo é capaz de avaliar o quão bem os valores do atributo distinguem a classe entre instâncias que são próximas umas das outras [21]

  O funcionamento da versão original do *Relief* é o seguinte: define-se um vetor de pesos W[A], 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 a 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 fórmula:

$$W[a] = W[a] - \frac{dist(a, R, H)}{t} + \frac{dist(a, R, M)}{t}, \tag{2.9}$$

  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 término do algoritmo, tem-se um vetor de pesos W[A] atualizado para todos os atributos da base: quanto maior o peso, mais relevante é o atributo.

  Esse algoritmo consegue lidar com atributos discretos e contínuos, mas é limitado a problemas de duas classes. O algoritmo foi estendido em [20], e entre as seis extensões propostas, a versão denominada *Relief-F* praticamente substituiu o método *Relief* original, pois é capaz de lidar com conjuntos de dados incompletos, com ruídos e com mais de duas classes.

2.2.2 Consistency-based Feature Selection

A métrica baseada em consistência proposta em [26] avalia a qualidade de um subconjunto de atributos pelo nível de consistência dos valores da classe quando as instâncias de treinamento são projetadas no subconjunto de atributos.

Um subconjunto de atributos é consistente se, para todas as instâncias com a mesma combinação de valores desses atributos, a classe é a mesma. Por outro lado, caso existam pelo menos duas instâncias com as mesmas combinações de valores de atributos, porém com classes diferentes, esse subconjunto de atributos é dito inconsistente. Entre esses dois extremos é possível calcular valores intermediários de consistência para um determinado subconjunto de atributos.

A métrica de consistência utilizada em [26] é dada pela fórmula:

$$Consistency(S) = 1 - \frac{\sum_{i=0}^{j} |D_i| - |M_i|}{N}, \tag{2.10}$$

onde S é um subconjunto de atributos, j é o número de combinações distintas dos valores dos atributos do subconjunto S, $|D_i|$ é o número de ocorrências da combinação i dos valores dos atributos, $|M_i|$ é a cardinalidade da classe majoritária para a mesma combinação i de valores e N é número total de instâncias na base de dados.

A consistência de um subconjunto nunca será menor que a do conjunto total de atributos, logo a prática usual é utilizar essa métrica em conjunto com uma busca exaustiva ou heurística que procure pelos menores subconjuntos com a consistência mais próxima possível da consistência do conjunto total de atributos. Uma comparação experimental entre métodos de busca aplicados para a medida de consistência de atributos é feita em [8].

2.2.3 Correlation-based Feature Selection

A técnica de seleção de atributos *Correlation-based Feature Selection* (CFS) proposta em [17] parte da hipótese de que um bom subconjunto de atributos é aquele que contém atributos com alta correlação com a classe, mas com uma baixa correlação entre si.

Dessa forma, o CFS avalia o subconjunto de atributos considerando a capacidade preditiva individual de cada um em conjunto com o grau de redundância entre eles. Subconjuntos de atributos que são muito correlacionados com a classe ao mesmo tempo em que possuem uma baixa correlação entre si são preferidos para a seleção.

Para caracterizar a força de uma associação entre um atributo e uma classe, é possível utilizar diversos coeficientes de correlação entre duas variáveis. O mais conhecido é o coeficiente de correlação de Pearson, que é obtido dividindo-se a covariância das variáveis pelo produto dos seus desvios padrões, representado pela fórmula [37]:

$$r(x,y) = \frac{\sum(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_i (x_i - \overline{x})^2}\sqrt{\sum_i (y_i - \overline{y})^2}} \tag{2.11}$$

O valor absoluto de r varia de 0 (se não existir nenhuma correlação entre as duas variáveis x e y) a 1 (caso as duas variáveis sejam linearmente dependentes). A relevância de um subconjunto de atributos pode ser definida como [17]:

$$CFS(X_k, C) = \frac{k * r_{kc}}{\sqrt{k + (k-1)r_{kk}}}, \tag{2.12}$$

onde $r_{kc} = \overline{r}(X_k, C)$ é o coeficiente de correlação médio entre os k atributos de um subconjunto X_k e a classe C, e $r_{kk} = \overline{r}(X_k, X_k)$ é o coeficiente de correlação médio entre os diferentes atributos [39].

2.3 Métodos de Seleção de Atributos do tipo Wrapper

Métodos do tipo *Wrapper* ("Empacotados") utilizam o próprio algoritmo de classificação adotado como uma caixa preta para avaliar os subconjuntos de atributos de acordo com a sua capacidade preditiva [14].

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 [17]. Porém, como esse algoritmo é executado diversas vezes para subconjuntos distintos de atributos, *wrappers* podem ter um custo computacional muito elevado e se tornarem inviáveis em bases de dados com um número grande de atributos.

Existem inúmeros 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 [27].

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 [10].

O algoritmo OBVILION, proposto em [29], 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 a acurácia estimada decline [2]. Esse método de seleção de atributos é considerado um *wrapper*, pois envolve a execução do algoritmo do k-NN para uma parte dos dados de treinamentos, utilizando um sistema de validação cruzada *leave-one-out* [18] para estimar a acurácia de cada subconjunto de atributos.

Outros algoritmos foram desenvolvidos em conjunto 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 [2].

Também existem estratégias híbridas que tentam combinar as vantagens das abordagens filtro e *wrapper* [27]. Uma técnica muito utilizada é o uso de filtros para reduzir o conjunto inicial de atributos, viabilizando o uso de um *wrapper* em seguida.

2.4 Métodos de Seleção do tipo Embedded

Métodos *Embedded* ("Embutidos") selecionam o conjunto de atributos no próprio processo de construção do modelo de classificação, durante a fase de treinamento, e são geralmente específicos para um dado algoritmo de classificação [14].

A seguir serão descritos dois métodos em que a seleção de atributos é inerente ao algoritmo de classificação: árvores de decisão e *Random Forests*.

2.4.1 Árvores de decisão

As árvores de decisão são estruturas simples, porém eficientes, usadas para produzir classificadores a partir de bases de dados, sendo largamente utilizadas devido à sua apresentação intuitiva e compreensível para a análise do problema em questão.

As árvores de decisão podem ser vistas como detentoras de uma técnica de seleção de atributos incorporada ao seu algoritmo de indução. Esse algoritmo seleciona internamente um subconjunto de atributos para compor a árvore, de forma que cada nó de decisão seja responsável por testar um atributo, e cada ramo descendente corresponda a um possível valor ou conjunto de valores desse atributo.

Existem vários métodos que analisam os atributos de uma base de treinamento e constroem a árvore de decisão, utilizando em cada etapa uma função para avaliar qual é o atributo com a maior capacidade de discriminação da classe [2]. A árvore é estendida até que os atributos se esgotem, ou até que nenhuma discriminação seja possível, ao avaliar o subconjunto restante de atributos.

Exemplos de algoritmos que utilizam métodos de particionamento para a indução do modelo de classificação são: ID3 [30], C4.5 [32], e CART [22].

2.4.2 Random Forests

O método de classificação conhecido como *Random Forests* foi proposto em [4], e consiste em uma técnica de agregação de classificadores do tipo árvore, construídos de forma que a sua estrutura seja composta de maneira aleatória. Para determinar a classe de uma instância, o método combina o resultado de várias árvores de decisão, por meio de um mecanismo de votação. O classificador é baseado no método *Bagging* [3].

Para cada árvore gerada é utilizado um conjunto de treinamento diferente, formado por n instâncias de treinamento escolhidas aleatoriamente (i.e., uma amostra ou *bootstrap*). Para cada nó da árvore gerada, são escolhidos aleatoriamente m atributos que orientam o direcionamento do nó, baseado na melhor discriminação de classes do conjunto de treinamento, de acordo com uma métrica. Em geral, o valor de m deve ser bem menor que o total de atributos da base, de maneira que possam ser geradas árvores distintas, que são combinadas para classificar uma nova instância. O modelo gerado elege a classe mais frequente entre as opções individuais de cada árvore.

Dessa forma a seleção de atributos é feita no instante de construção do modelo de classificação, caracterizando a seleção do tipo *embedded*. A vantagem desse classificador é que ele permite bases de dados com um número grande de atributos, contudo é suscetível a *overfitting* em determinadas bases [35].

2.5 Sumário

O Capítulo 2 descreveu a tarefa de seleção de atributos para o problema de classificação, a sua motivação e objetivos. Além disso, foram delineadas as três categorias em que as estratégias de seleção são classificadas: filtro, *wrapper* ou *embedded*.

Os exemplos das técnicas enquadradas nessas três categorias fazem a seleção de maneira prévia, ou seja, em uma fase de pré-processamento anterior ao processo de classificação. Dessa forma, além de descartar a informação contida nos atributos que não são mantidos na base, a seleção também não considera os valores da instância a ser classificada.

Conforme será visto no próximo capítulo, a técnica de seleção de atributos desenvolvida nesta dissertação busca resolver esses problemas, avaliando os valores de cada atributo da instância para orientar a escolha dos atributos relevantes para a classificação.

Capítulo 3

Seleção Lazy de Atributos

3.1 Considerações Iniciais

Em estratégias de seleção de atributos convencionais, os atributos são selecionados em uma fase de pré-processamento dos dados, e aqueles não selecionados são descartados da base de dados e não participam mais do processo de classificação.

Nesta dissertação, propõe-se uma estratégia de seleção de atributos do tipo *lazy*, baseada na hipótese de que adiar a seleção de atributos até o ponto em que a instância é submetida ao classificador pode contribuir na identificação dos melhores atributos para a correta classificação dessa instância em particular. Para cada diferente instância a ser classificada, é possível selecionar um subconjunto de atributos distinto e mais apropriado para classificá-la.

A seguir é dado um exemplo para ilustrar o fato de que a classificação de certas instâncias pode se beneficiar de atributos provavelmente descartados por estratégias convencionais de seleção de atributos. Além disso, ilustra-se que alguns atributos selecionados pelas estratégias convencionais podem ser irrelevantes para a classificação de outras instâncias. Em outras palavras, o exemplo evidencia que atributos podem ser úteis, ou não, dependendo dos valores dos atributos da instância a ser classificada.

Na Tabela 3.1, a mesma base de dados, descrita por três atributos – X, Y, e a classe C – é representada duas vezes. A relação da esquerda está ordenada pelos valores do atributo X e a da direita está ordenada pelos valores de Y. Na relação da esquerda, pode ser observado que os valores de X estão fortemente correlacionados com os valores da classe, indicando que este é um atributo útil para a classificação. Apenas o valor 4 não discrimina bem a classe. Por outro lado, como mostrado na relação da direita, o

atributo Y seria um forte candidato a ser eliminado da base já que os seus valores não discriminam bem a classe. Porém há uma forte correlação do valor 4 do atributo Y com o valor B da classe, que seria perdido se esse atributo fosse eliminado. A classificação de uma instância com o valor 4 no atributo Y poderia claramente beneficiar-se com a presença desse atributo.

Tabela 3.1: Exemplo de Base de Dados: Motivação

Base ordenada por X			Base ordenada por Y		
– X –	– Y –	– C –	– X –	– Y –	– C –
1	2	B	2	1	A
1	3	B	3	1	B
1	4	B	4	1	A
2	1	A	1	2	B
2	2	A	2	2	A
2	3	A	3	2	B
3	1	B	1	3	B
3	2	B	2	3	A
3	4	B	4	3	B
4	1	A	1	4	B
4	3	B	3	4	B
4	4	B	4	4	B

Uma estratégia de seleção de atributos convencional – que será denominada a partir de agora como uma estratégia de seleção *"eager"* – provavelmente selecionaria o atributo X em detrimento do atributo Y, independentemente das instâncias que fossem submetidas para classificação.

De acordo com [25], uma vantagem importante de abordagens do tipo *lazy* é que elas podem responder a condições inesperadas de maneiras que não são possíveis para os classificadores *eager*, pois elas não perdem informações cruciais que podem ser utilizadas para gerar predições mais acuradas.

Logo, a principal motivação da seleção *lazy* de atributos proposta é a capacidade de acessar os valores dos atributos da instância a ser classificada, e utilizar essa informação para selecionar atributos que discriminem bem as classes para esses valores particulares. Como resultado, para cada instância é possível selecionar atributos que são úteis para a classificação dessa instância em particular.

3.2 Estratégia Proposta

Nesta seção, apresenta-se a técnica *lazy* de seleção de atributos proposta neste trabalho.

Os parâmetros de entrada da técnica *lazy* proposta são: uma base de dados D com n atributos, uma instância $I = (v_1, v_2, ..., v_n)$ a ser classificada de acordo com os valores dos seus atributos, e um número r, $1 \leq r < n$, que representa a quantidade de atributos a serem selecionados.

O conceito de entropia é usado para avaliar a qualidade de cada valor de atributo na classificação de uma instância. Especificamente, a entropia é utilizada para medir quão bem os valores dos atributos de uma instância determinam a classe. A entropia é bastante utilizada para medir a relevância de um atributo em estratégias *eager* do tipo filtro, que avaliam atributos individualmente [16, 40]. Por esse motivo, e também pela sua relativa simplicidade de adaptação, optou-se pelo seu uso na adequação de uma medida para a seleção de atributos *lazy*.

Seja $D(A_1, A_2, ..., A_n, C)$, $n \geq 1$, uma base de dados com $n+1$ atributos, onde C é o atributo classe. Seja $\{c_1, c_2, ..., c_m\}$, $m \geq 2$, o domínio de C. A entropia da distribuição da classe em D, representada por $H(D)$, é definida por:

$$H(D) = -\sum_{i=1}^{m}[p_i * log(p_i)], \tag{3.1}$$

onde p_i é a probabilidade que uma instância arbitrária em D pertença à classe c_i.

Seja $\{a_{j1}, a_{j2}, ..., a_{jk_j}\}$, $k_j \geq 1$, o domínio do atributo A_j, $1 \leq j \leq n$. Seja D_{ji}, $1 \leq j \leq n$ e $1 \leq i \leq k_j$, a partição de D composta por todas as instâncias cujo valor de A_j seja igual a a_{ji}. A entropia da distribuição de classes em D, restrita aos valores do atributo A_j, $1 \leq j \leq n$, representada por $H(D|A_j)$, é definida por:

$$H(D|A_j) = \sum_{i=1}^{k_j}[(\frac{|D_{ji}|}{|D|}) * H(D_{ji})]. \tag{3.2}$$

Nesta proposta, define-se a entropia da distribuição de classes em D, restrita pelo valor a_{ji}, $1 \leq i \leq k_j$, do atributo A_j, $1 \leq j \leq n$, representada por $H(D|A_j = a_{ji})$, por meio da fórmula:

$$H(D|A_j = a_{ji}) = H(D_{ji}). \tag{3.3}$$

O conceito definido na Fórmula 3.2 equivalente ao utilizado pela técnica *eager* conhecida como *Information Gain Attribute Ranking* [40] para medir a capacidade de um atributo discriminar os valores da classe. Essa técnica foi descrita no Capítulo 2 (Seção 2.2.1).

A Fórmula 3.3 será utilizada na estratégia proposta de seleção *lazy* para medir a capacidade de discriminação da classe de um valor específico a_{ji} de um atributo particular A_j. Quanto mais próxima de zero é a entropia $H(D|A_j = a_{ji})$, maior a chance que o valor a_{ji} do atributo A_j seja um bom discriminador da classe.

Para selecionar os r melhores atributos para classificar a instância $I = (v_1, v_2, ..., v_n)$, os n atributos são avaliados baseados na medida *lazy* proposta ($LazyH$), definida na Fórmula 3.4, que estabelece que, para cada atributo A_j, se a capacidade discriminatória do valor específico v_j de A_j ($H(D|A_j = v_j)$) é melhor que (menor que) a capacidade de discriminação geral do atributo A_j ($H(D|A_j)$), então a primeira será considerada como a medida *lazy* do atributo A_j.

Dessa forma, a medida proposta para avaliar a qualidade de cada atributo A_j é definida por:

$$LazyH(D|A_j = v_j) = Min(H(D|A_j = v_j), H(D|A_j)), \tag{3.4}$$

onde $Min()$ retorna o menor entre os seus parâmetros.

Após calcular o valor $LazyH(D|A_j = v_j)$ para cada atributo A_j, a estratégia *lazy* selecionará os r atributos que apresentam os r menores valores de $LazyH$.

Além de ser caracterizada pela forma *lazy* de selecionar os atributos relevantes, a técnica proposta poderia ser classificada como do tipo filtro, com avaliação individual dos atributos.

Capítulo 4

Resultados Experimentais

A implementação da técnica proposta está baseada no conceito de entropia e, buscando avaliar a característica *lazy* da proposta, optou-se por compará-la com a estratégia *eager* mais próxima que também utiliza o conceito de entropia.

A acurácia preditiva foi o critério utilizado na avaliação comparativa entre o método proposto e a outra técnica de seleção de atributos. Os valores de acurácia preditiva obtidos pelos classificadores foram avaliados a partir de um procedimento de validação cruzada com 10 partições [18], na qual cada partição foi obtida de maneira aleatória. A acurácia correspondente a cada execução do classificador é uma média dos valores obtidos em cada partição. As mesmas partições foram utilizadas nos experimentos de cada técnica empregada.

A seleção *lazy* ocorre quando o classificador recebe uma nova instância a ser classificada. Como para cada nova instância um subconjunto distinto de atributos deve ser considerado pelo classificador, os atributos não selecionados pela estratégia *lazy* para uma dada instância não são removidos da base de dados, mas apenas desconsiderados pelo procedimento de classificação.

Para se realizar a comparação com a implementação *lazy* proposta, foi utilizada uma técnica de seleção de atributos *eager* correspondente, no caso, à técnica *Information Gain Attribute Ranking* [40]. Essa técnica está disponível na ferramenta Weka com o nome "InfoGainAttributeEval". Ambas utilizam o conceito de entropia para dispor os atributos em um *ranking*, além de serem estratégias supervisionadas e, portanto, necessitam de antemão do número de atributos a serem selecionados.

A avaliação comparativa das técnicas foi realizada empregando um total de 40 bases pertencentes ao repositório de dados *UCI Machine Learning Repository* [1]. Essas bases de

dados estão relacionadas a áreas de aplicações distintas e, portanto, possuem uma grande variedade em termos de conteúdo, tamanho e complexidade. A Tabela 4.1 apresenta as seguintes informações sobre essas bases: nome, número de atributos, número de classes e número de instâncias.

Tabela 4.1: Bases de dados do Repositório do UCI

Base de Dados	**Attributos**	**Classes**	**Instâncias**
anneal	38	5	898
audiology	69	24	226
autos	25	6	205
breast-cancer	9	2	286
breast-w	9	2	699
chess-Kr-vs-Kp	36	2	3196
credit-a	15	2	690
diabetes	8	2	768
flags	29	8	194
glass	9	6	214
heart-cleveland	13	2	303
heart-hungarian	13	2	294
hepatitis	19	2	155
horse-colic	27	2	368
hypothyroid	29	4	3772
ionosphere	34	2	351
labor	16	2	57
letter-recognition	16	26	20000
lymph	18	4	148
mol-bio-promoters	57	2	106
mol-bio-splice	60	3	3190
mushroom	22	2	8124
optdigits	64	10	5620
pendigits	16	10	10992
postoperative	8	3	90
primary-tumor	17	21	339
solar-flare1	12	6	323
solar-flare2	12	6	1066
sonar	60	2	208
soybean-large	35	19	683
spambase	57	2	4601
statlog-heart	13	2	270
statlog-segment	19	7	2310
statlog-vehicle	18	4	846
thyroid-sick	29	2	3772
vote	16	2	435
vowel	13	11	990
waveform-5000	40	3	5000
wine	13	3	178
zoo	17	7	101

A medida de entropia utilizada para avaliar a qualidade de cada atributo na técnica proposta necessita de valores de atributos discretos. Logo foi adotado um método supervisionado de discretização baseado em ganho de informação e proposto em [12] para discretizar os atributos contínuos.

A estratégia de seleção de atributos *lazy* foi incorporada a três técnicas de classificação *lazy* distintas. Na Seção 4.1, são apresentados os resultados obtidos com o classificador k-NN (*k-Nearest Neighbor*), na Seção 4.2, são apresentados os resultados obtidos com o classificador Naive Bayes e, na Seção 4.3, estão os resultados do classificador HiSP.

4.1 Resultados Experimentais com o k-NN

O algoritmo k-NN basicamente atribui o rótulo da classe majoritária entre as k instâncias mais próximas de uma nova instância a ser classificada, a partir de uma base de treinamento [6, 7]. A distância entre cada instância de treinamento e a nova instância é calculada por uma função definida pelos valores dos seus atributos. Logo, a utilização de uma seleção de atributos *lazy* implica que o cálculo dessas distâncias poderá ser feito utilizando subconjuntos diferentes de atributos para diferentes instâncias a serem classificadas.

Os resultados da implementação original do k-NN, disponível na ferramenta Weka com o nome IBk, foram comparados com os resultados da versão adaptada da implementação IBk que executa a seleção de atributos *lazy* antes de classificar cada instância de teste. Ambas foram executadas com diferentes valores do parâmetro k: 1, 3 e 5. Em todos os experimentos relatados neste trabalho, os parâmetros não mencionados foram configurados para os seus valores *default* na ferramenta Weka, que são: distanceWeighting=No, meanSquared=false, noNormalization=false e windowSize=0.

Inicialmente, são reportados os resultados obtidos pelas estratégias de seleção *eager* e *lazy* quando aplicadas em três bases de dados específicas: *Wine*, *Heart-Hungarian* e *Vowel*. O principal objetivo dessa primeira análise é apresentar alguns resultados em detalhes, para que a metodologia experimental seja melhor entendida. Além disso, são apresentadas evidências de que a estratégia *lazy* pode superar a estratégia *eager* para determinadas bases de dados.

As Tabelas 4.2, 4.3 e 4.4 mostram as acurácias preditivas obtidas pelo algoritmo k-NN, utilizando tanto a estratégia de seleção *eager* quanto a seleção *lazy*, com o parâmetro k igual a 1 (colunas 2 e 3), k igual a 3 (colunas 4 e 5), e k igual a 5 (colunas 6 e 7), para as bases de dados *Wine*, *Heart-Hungarian* e *Vowel*, respectivamente. Cada linha dessas tabelas representa a execução das estratégias de seleção com um número fixo de atributos a serem selecionados. A primeira coluna indica o número de atributos selecionados a partir de uma percentagem do número total de atributos na base de dados – o número absoluto de atributos é indicado entre parênteses. A última linha (100%) representa a execução utilizando o conjunto total de atributos, ou seja, quando não é feita nenhuma seleção de atributos. Cada valor de acurácia é obtido utilizando um procedimento de validação cruzada com dez partições, conforme descrito no início do capítulo. Na comparação das duas estratégias de seleção de atributos, valores em negrito indicam o melhor resultado.

A Tabela 4.2 mostra os resultados da base *Wine*. É possível observar que a estratégia *lazy* apresentou resultados melhores que a estratégia *eager*, independente do número de atributos selecionados. Os resultados com a estratégia *lazy* foram inclusive melhores que os obtidos sem seleção de atributos. A estratégia *lazy* alcançou a acurácia de 100% quando 3 dos 13 atributos foram selecionados, com o valor de k igual a 3 e 5. Esses resultados são uma evidência de que a estratégia *lazy* pode ser melhor que uma abordagem *eager* e melhor também do que a classificação sem seleção de atributos.

Tabela 4.2: Acurácias para a base *Wine*

Atributos	**1-NN**		**3-NN**		**5-NN**	
Selecionados	**Eager**	**Lazy**	**Eager**	**Lazy**	**Eager**	**Lazy**
10% (1)	82,6	**95,5**	82,6	**95,5**	82,6	**95,5**
20% (3)	95,5	**97,8**	94,4	**100,0**	94,4	**100,0**
30% (4)	95,5	**98,3**	93,3	**98,9**	93,3	**98,9**
40% (5)	97,2	**98,9**	93,3	**98,9**	93,3	**98,9**
50% (7)	97,8	**98,3**	96,6	**98,3**	96,6	**98,3**
60% (8)	98,9	98,9	96,6	**98,3**	96,6	**98,3**
70% (9)	97,2	**97,8**	94,9	**97,8**	94,9	**97,8**
80% (10)	97,2	**98,3**	94,4	**97,2**	94,4	**97,2**
90% (12)	96,6	**97,8**	96,1	**96,6**	96,1	**96,6**
100% (13)	98,3		96,1		96,1	

A Tabela 4.3 apresenta os resultados obtidos com a base *Heart-Hungarian*. Neste experimento, a estratégia *eager* obteve um comportamento superior ao da estratégia *lazy* quando o número de atributos selecionados foi inferior a 60%. As duas estratégias apresentaram as mesmas acurácias quando o número de atributos selecionados foi superior a 60%. Além disso, as duas estratégias foram capazes de obter acurácias melhores que as obtidas pela classificação sem seleção de atributos.

Tabela 4.3: Acurácias para a base *Heart-Hungarian*

Atributos	**1-NN**		**3-NN**		**5-NN**	
Selecionados	**Eager**	**Lazy**	**Eager**	**Lazy**	**Eager**	**Lazy**
10% (1)	**80,3**	78,6	**80,3**	79,3	**80,3**	79,3
20% (3)	**82,0**	81,3	**84,0**	83,0	**84,0**	83,0
30% (4)	**82,7**	81,3	**83,0**	82,7	**83,0**	82,7
40% (5)	**80,6**	79,6	**81,6**	81,0	**81,6**	81,0
50% (7)	80,6	80,6	**83,0**	82,7	**83,0**	82,7
60% (8)	81,0	**81,6**	82,7	**83,3**	82,7	**83,3**
70% (9)	80,3	80,3	83,0	83,0	83,0	83,0
80% (10)	80,3	80,3	82,3	82,3	82,3	82,3
90% (12)	80,3	80,3	82,3	82,3	82,3	82,3
100% (13)	80,3		82,3		82,3	

A Tabela 4.4 mostra os resultados da base *Vowel*, em que a seleção de atributos não contribui para o processo de classificação. Os resultados para essa base de dados indicam que nem a estratégia *lazy* nem a *eager* foram capazes de superar os resultados obtidos

pela classificação sem seleção de atributos. Isso indica que todos os atributos da base de dados contribuem para o desempenho geral do classificador. Também é possível observar que quanto menor o número de atributos selecionados, menor é a acurácia do classificador, para as duas estratégias.

Tabela 4.4: Acurácias para a base *Vowel*

Atributos	**1-NN**		**3-NN**		**5-NN**	
Selecionados	**Eager**	**Lazy**	**Eager**	**Lazy**	**Eager**	**Lazy**
10% (1)	26,5	**30,5**	26,5	**30,5**	26,5	**30,5**
20% (3)	51,6	**53,1**	50,1	**50,9**	50,1	**50,9**
30% (4)	**59,7**	59,3	52,2	**52,9**	52,2	**52,9**
40% (5)	**69,9**	68,9	**59,3**	58,0	**59,3**	58,0
50% (7)	76,5	**77,6**	64,2	**64,6**	64,2	**64,6**
60% (8)	80,1	**80,7**	**69,6**	68,1	**69,6**	68,1
70% (9)	82,7	82,7	71,2	71,2	71,2	71,2
80% (10)	86,3	86,3	74,0	74,0	74,0	74,0
90% (12)	89,8	89,8	**78,3**	78,1	**78,3**	78,1
100% (13)	89,8		78,6		78,6	

Na Tabela 4.5, estão sumarizados os resultados obtidos pelas estratégias *eager* e *lazy* usando os classificadores 1-NN, 3-NN e 5-NN, para as 40 bases de dados. Para cada classificador e base de dados, foram comparadas as acurácias das estratégias ao se variar o percentual de atributos selecionados de 10% a 90% do total de atributos, com um incremento regular de 10%, considerando as nove execuções distintas das duas estratégias. As colunas "Eager" e "Lazy" indicam o número de vezes em que cada estratégia obteve a maior acurácia preditiva, considerando as nove comparações. O melhor comportamento é indicado por valores em negrito. A coluna "Empate" representa o número de vezes em que as estratégias obtiveram a mesma acurácia, ou seja, o mesmo número de acertos. A última linha mostra a soma total de melhores resultados obtidos por cada estratégia.

Como pode ser observado na última linha da Tabela 4.5, para a maior parte das execuções, a estratégia *lazy* alcançou um número maior de resultados superiores do que a estratégia *eager*, independente do valor de k. Quando k é igual a 1, a estratégia *lazy* obteve o melhor resultado 220 vezes contra 87 vezes para a estratégia *eager*, com 57 empates. O comportamento foi semelhante para k igual a 3 e 5.

Para as bases *Letter, Mol-Bio-Splice, Pendigits* e *Wine*, a estratégia *lazy* obteve os melhores resultados em todos os testes, independente do número de atributos escolhidos ou do parâmetro k do classificador. Para as bases *Chess-kr-vs-kp, Ionosphere, Labor, Primary-Tumor* e *Spambase*, para pelo menos um dos parâmetros de k a estratégia *lazy* alcançou o melhor resultado em todos os nove testes. O comportamento oposto não ocorreu, ou seja, uma prevalência da estratégia *eager* em todos os nove testes, para alguma

Tabela 4.5: Número de execuções em que cada estratégia obteve o melhor resultado utilizando o classificador k-NN

Base	**1-NN**			**3-NN**			**5-NN**		
	Eager	**Lazy**	**Empate**	**Eager**	**Lazy**	**Empate**	**Eager**	**Lazy**	**Empate**
anneal	1	**5**	3	0	**6**	3	1	**5**	3
audiology	0	**5**	4	3	**5**	1	4	4	1
autos	**6**	1	2	**6**	2	1	**6**	2	1
breast-cancer	4	**5**	0	**5**	4	0	3	**6**	0
breast-w	1	**7**	1	3	**5**	1	4	4	1
chess-kr-vs-kp	0	**9**	0	1	**8**	0	2	**7**	0
credit-a	4	**5**	0	4	**5**	0	3	**6**	0
diabetes	3	3	3	3	3	3	**4**	2	3
flags	**7**	2	0	**7**	2	0	**6**	2	1
glass	0	**7**	2	0	**7**	2	0	**7**	2
heartcleveland	2	**4**	3	**4**	2	3	3	3	3
hearthungarian	**4**	2	3	2	**5**	2	**5**	1	3
hepatitis	1	**8**	0	2	**7**	0	2	**7**	0
horse-colic	4	**5**	0	4	**5**	0	4	**5**	0
hypo-thyroid	1	**8**	0	1	**8**	0	1	**8**	0
ionosphere	0	**9**	0	3	**6**	0	4	**5**	0
labor	0	**9**	0	1	**7**	1	2	**6**	1
letter	0	**9**	0	0	**9**	0	0	**9**	0
lymph	3	**6**	0	1	**8**	0	2	**7**	0
mol-bio-promot	4	**5**	0	4	**5**	0	4	**5**	0
mol-bio-splice	0	**9**	0	0	**9**	0	0	**9**	0
mushroom	0	3	**6**	0	3	**6**	0	3	**6**
optdigits	1	**6**	2	2	**6**	1	3	**5**	1
pendigits	0	**9**	0	0	**9**	0	0	**9**	0
postoperative	2	**4**	3	0	**6**	3	0	**5**	4
primary-tumor	3	**6**	0	3	**6**	0	0	**9**	0
solar-flare1	**7**	2	0	4	**5**	0	3	**5**	1
solar-flare2	**4**	3	2	2	**6**	1	1	**6**	2
sonar	**7**	2	0	3	**6**	0	2	**7**	0
soybean-large	1	**7**	1	1	**7**	1	0	**8**	1
spambase	0	**9**	0	0	**9**	0	3	**6**	0
statlog-heart	3	3	3	3	3	3	**4**	2	3
statlog-segment	1	**8**	0	1	**8**	0	1	**8**	0
statlog-vehicle	2	**7**	0	1	**8**	0	2	**7**	0
thyroid-sick	**5**	2	2	**5**	3	1	**5**	3	1
vote	2	**6**	1	1	**7**	1	1	**7**	1
vowel	2	**4**	3	2	**5**	2	3	**4**	2
waveform-**5**000	1	3	**5**	1	3	**5**	1	3	**5**
wine	0	**9**	0	0	**9**	0	0	**9**	0
zoo	1	4	4	4	4	1	**5**	3	1
TOTAIS	87	**220**	53	87	**231**	42	94	**219**	47

base de dados.

Para k igual a 1, em 28 de 40 bases de dados, a estratégia *lazy* obteve um número superior de melhores resultados do que o número de empates e do que o número em que a estratégia *eager* prevaleceu. Isso ocorreu com a estratégia *eager* em apenas sete bases de dados. As ocorrências correspondentes a esses casos estão indicadas por números em negrito na tabela. Nos outros cinco casos, ocorreu um número igual ou maior de empates entre as estratégias *eager* e *lazy*.

Para k igual a 3 e 5, esses números também foram 28 vezes para a estratégia *lazy* e sete para a *eager*, indicando uma regularidade nos resultados e que o comportamento superior da estratégia *lazy* não é dependente do valor do parâmetro k.

Até aqui, as duas estratégias foram comparadas sempre considerando o mesmo número de atributos a serem selecionados, variando percentualmente de 10% a 90% do total de atributos da base. Na próxima análise, os resultados são avaliados considerando que a melhor opção de número de atributos seria adotada por cada estratégia, dentro das faixas apresentadas anteriormente. Na prática, essa percentagem de atributos poderia ser estimada de maneira independente para cada estratégia, utilizando-se, por exemplo, um procedimento de validação cruzada.

Na Tabela 4.6, são relatadas, para cada base de dados, as melhores acurácias obtidas com cada estratégia de seleção de atributos, considerando as nove diferentes quantidades percentuais de atributos. Para os classificadores 1-NN, 3-NN e 5-NN, são apresentadas nas colunas "Eager" e "Lazy" as melhores acurácias obtidas por cada estratégia. O número entre parênteses indica a percentagem de atributos selecionados em que essa acurácia foi obtida. A coluna "SemSel" representa a acurácia obtida quando nenhuma seleção de atributos é realizada. Para cada base de dados e execução do k-NN, os valores em negrito indicam os melhores resultados obtidos. Na última linha da tabela, estão sumarizados os totais de bases em que cada estratégia de seleção obteve o melhor resultado para as 40 bases de dados.

É possível observar que, nesse cenário, a estratégia *lazy* tende a alcançar acurácias superiores às da estratégia *eager*, independente do parâmetro k. Para o valor de k igual a 1, a estratégia *lazy* obteve a melhor acurácia 28 vezes, enquanto a *eager* obteve a melhor acurácia 21 vezes. Para k igual a 3, os resultados apontaram 29 melhores resultados da estratégia *lazy* contra 15 da *eager* e, para k igual a 5, o resultado foi 26 a favor da estratégia *lazy* contra 17 da estratégia *eager*.

Tabela 4.6: As melhores acurácias preditivas alcançadas por cada estratégia utilizando o classificador k-NN

Base	**1-NN**			**3-NN**			**5-NN**		
	Eager	**Lazy**	**SemSel**	**Eager**	**Lazy**	**SemSel**	**Eager**	**Lazy**	**SemSel**
anneal	99,3 (60)	**99,6** (30)	99,2	98,0 (70)	**98,4** (20)	98,0	97,2 (80)	**97,8** (20)	97,2
audiology	75,7 (30)	**77,0** (20)	76,1	68,6 (20)	**72,1** (20)	65,9	69,5 (30)	**73,0** (20)	60,6
autos	**88,3** (50)	87,3 (50)	85,9	**81,5** (90)	**81,5** (90)	**81,5**	**77,1** (90)	**77,1** (90)	**77,1**
breast-cancer	72,7 (70)	**74,1** (30)	69,9	72,7 (70)	**73,4** (30)	70,3	74,1 (90)	**75,2** (70)	74,1
breast-w	**97,3** (70)	97,1 (40)	97,1	**97,0** (80)	**97,0** (90)	96,9	**97,1** (80)	**97,1** (90)	97,0
chess-kr-kp	96,4 (40)	**96,8** (40)	96,6	96,1 (40)	96,2 (40)	**96,6**	95,7 (40)	95,6 (40)	**96,1**
credit-a	**85,5** (10)	**85,5** (30)	82,3	85,5 (10)	**85,9** (30)	84,2	**85,9** (80)	85,8 (80)	84,6
diabetes	77,3 (60)	**78,0** (50)	76,4	78,0 (60)	**78,3** (50)	76,8	**78,1** (60)	78,0 (60)	77,2
flags	**63,4** (10)	60,8 (30)	59,8	**63,4** (40)	61,3 (30)	55,2	62,4 (30)	**63,4** (30)	57,7
glass	**77,1** (80)	**77,1** (80)	**77,1**	**75,7** (80)	**75,7** (80)	**75,7**	**73,8** (80)	**73,8** (80)	**73,8**
heart-clevela	**84,5** (20)	82,8 (20)	80,5	**84,2** (20)	82,8 (20)	82,5	**83,5** (20)	82,8 (90)	82,8
heart-hungari	**82,7** (30)	81,6 (60)	80,3	83,0 (60)	**83,7** (60)	83,0	**84,0** (20)	83,3 (60)	82,3
hepatitis	83,9 (40)	**86,5** (70)	83,9	**86,5** (50)	**86,5** (60)	83,9	84,5 (50)	**85,2** (50)	84,5
horse-colic	83,4 (20)	**83,7** (30)	78,5	**83,2** (20)	82,9 (30)	77,2	**83,2** (20)	82,1 (20)	77,4
hypo-thyroid	94,6 (20)	**97,0** (10)	91,5	94,8 (20)	**97,5** (10)	93,2	94,6 (20)	**97,3** (10)	93,3
ionosphere	93,4 (60)	**94,6** (40)	92,6	91,5 (50)	**92,3** (20)	90,6	90,9 (50)	**92,3** (10)	89,7
labor	96,5 (60)	**100** (60)	96,5	96,5 (90)	**98,2** (90)	96,5	**96,5** (80)	**96,5** (60)	91,2
letter	91,9 (70)	**91,9** (70)	91,9	90,4 (90)	**90,6** (90)	90,6	89,7 (90)	89,8 (90)	**89,8**
lymph	**85,1** (70)	**85,1** (90)	82,4	83,1 (80)	**83,8** (90)	**83,8**	84,5 (90)	**85,1** (90)	83,8
mol-bio-promo	**90,6** (10)	89,6 (10)	80,2	**91,5** (10)	88,7 (10)	80,2	87,7 (10)	**89,6** (30)	79,2
mol-bio-splic	89,7 (10)	**90,7** (10)	73,3	88,5 (10)	**91,2** (10)	77,4	88,2 (10)	**91,2** (10)	79,4
mushroom	**100** (40)	**100** (30)	**100**	**100** (40)	**100** (30)	**100**	**100** (40)	**100** (40)	**100**
optdigits	94,8 (70)	**94,8** (70)	94,3	95,4 (60)	**95,5** (60)	95,3	**95,6** (80)	95,5 (80)	95,5
pendigits	96,3 (90)	96,8 (90)	**97,1**	96,0 (90)	96,5 (90)	**96,7**	95,5 (90)	96,1 (90)	**96,5**
postoperative	**71,1** (20)	**71,1** (20)	63,3	**71,1** (10)	**71,1** (10)	68,9	**71,1** (10)	**71,1** (10)	**71,1**
primary-tumor	**42,8** (40)	42,5 (20)	38,3	44,2 (70)	**44,5** (30)	43,7	45,7 (90)	**46,6** (80)	46,3
solar-flare1	**72,8** (20)	71,5 (20)	65,9	**73,1** (20)	72,1 (20)	65,3	**71,2** (20)	70,9 (50)	66,6
solar-flare2	75,9 (20)	**76,3** (20)	73,5	75,7 (20)	**76,0** (20)	74,0	75,7 (20)	**76,2** (20)	73,8
sonar	**88,5** (60)	86,5 (60)	86,5	88,0 (90)	**88,5** (40)	86,1	83,7 (30)	**85,6** (80)	84,6
soybean-large	**93,4** (80)	**93,4** (80)	92,2	**92,4** (90)	**92,4** (90)	91,5	**91,9** (90)	**91,9** (90)	90,8
spambase	93,1 (90)	**93,7** (30)	93,0	93,3 (90)	**93,6** (90)	93,3	93,2 (90)	**93,3** (90)	93,2
statlog-heart	84,8 (30)	**85,2** (50)	84,1	**84,8** (30)	84,4 (50)	80,7	85,6 (50)	**85,9** (40)	82,2
statlog-segme	**94,8** (90)	94,7 (90)	94,7	93,9 (90)	93,9 (90)	**94,0**	**93,1** (90)	92,9 (90)	**93,1**
statlog-vehic	**71,4** (50)	**71,4** (90)	70,9	71,4 (90)	**71,5** (80)	71,3	70,6 (90)	**71,0** (70)	70,7
thyroid-sick	**97,7** (40)	97,5 (60)	97,5	**97,5** (10)	97,3 (20)	97,1	**97,6** (40)	97,5 (20)	96,6
vote	95,2 (50)	**96,1** (10)	92,2	95,2 (50)	**95,6** (20)	92,4	95,2 (10)	**95,4** (10)	92,9
vowel	**89,8** (90)	**89,8** (90)	**89,8**	84,1 (90)	84,1 (90)	**84,6**	78,3 (90)	78,1 (90)	**78,6**
waveform-5000	74,6 (40)	**75,5** (10)	73,8	79,0 (40)	**79,1** (40)	78,6	80,8 (40)	**80,8** (40)	79,7
wine	**98,9** (60)	**98,9** (40)	98,3	97,2 (50)	**98,9** (20)	96,6	96,6 (50)	**100** (20)	96,1
zoo	**97,0** (80)	**97,0** (80)	96,0	**97,0** (80)	**97,0** (80)	93,1	**93,1** (90)	92,1 (80)	**93,1**
TOTAIS	21	28	4	15	29	8	17	26	10

Também é possível identificar que, em apenas algumas bases de dados, a seleção de atributos parece não ser útil para a classificação. A execução sem seleção de atributos – indicado na coluna "SemSel" – alcançou o melhor resultado nove vezes, considerando todos os valores de k. Outro resultado interessante é que, para algumas bases, como a *Wine, Audiology, Hypo-Thyroid* e *Ionosphere*, a acurácia máxima foi obtida com um número relativamente baixo de atributos selecionados.

Nos resultados apresentados até agora, foram comparadas as acurácias, obtidas por uma média de dez valores, sem levar em conta a sua significância estatística. Em seguida, é empregado o teste estatístico de comparação de médias *t–Student*, bi-caudal e pareado, com o objetivo de identificar quais acurácias preditivas comparadas são significativamente diferentes.

Nas comparações conduzidas anteriormente, cada valor de acurácia preditiva foi calculado como uma média de um procedimento de validação cruzada com dez partições. Para aplicar a análise do teste *t–Student*, será considerado cada um dos resultados individuais em cada uma das dez partições.

A Tabela 4.7 apresenta o resultado dessa análise estatística. Cada linha representa os resultados obtidos por uma aplicação distinta do k-NN, com k igual a 1, 3 e 5. As colunas "Eager" e "Lazy" destacam os números relatados na última coluna da Tabela 4.5, que é o número de vezes em que cada estratégia obteve o melhor valor de acurácia. Entre parênteses está o número de vezes em que cada estratégia obteve a melhor acurácia, considerando a significância estatística com um valor de $p \leq 0,05$, que significa que a probabilidade da diferença de desempenho das estratégias ter sido causada por fatores aleatórios é menor que 5%. A última coluna mostra o número de casos em que as duas acurácias foram iguais.

Observa-se que os resultados de acurácia com a estratégia *lazy* foram novamente superiores aos da estratégia *eager*. Para as execuções do 1-NN, a estratégia *lazy* obteve os melhores resultados com significância estatística em 87 vezes, contra apenas sete vezes da estratégia *eager*. Para os classificadores 3-NN e 5-NN, os resultados foram similares e mostram um desempenho superior da estratégia *lazy*.

Após levar em consideração o teste de significância estatística, a estratégia *lazy* alcançou uma acurácia igual ou melhor do que outra estratégia em 24 das 40 bases de dados, independente do valor de k ou do número de atributos selecionados. O mesmo só ocorreu para a estratégia *eager* em quatro bases, e nas 12 restantes ora a *lazy* foi melhor, ora a *eager* prevaleceu. Isso indica que selecionar atributos de modo *lazy* é geralmente uma

Tabela 4.7: Resultados do teste T-Student com nível de significância de 0,05

Classifier	Eager	Lazy	Empate
1-NN	87 (7)	220 (87)	53
3-NN	87 (10)	231 (84)	42
5-NN	94 (8)	219 (78)	47

opção melhor do que executar a seleção *eager* na fase de pré-processamento.

4.2 Resultados Experimentais com o Naive Bayes

De forma similar aos experimentos conduzidos com o k-NN, a estratégia *lazy* foi utilizada em conjunto com o classificador Naive Bayes, que é uma técnica de classificação baseada no teorema de Bayes. O classificador emprega esse teorema assumindo que os atributos contribuem de maneira independente para determinar o valor da classe – e apesar dessa premissa não ser sempre precisa, o classificador geralmente produz resultados aceitáveis na prática.

A seleção de atributos *lazy* foi incorporada ao algoritmo no passo anterior ao da classificação. Dessa forma, o Naive Bayes considera apenas os r atributos selecionados de maneira *lazy* para calcular as suas probabilidades, para cada instância de teste. De forma análoga aos experimentos com o k-NN, isso implica que para diferentes instâncias serão utilizados subconjuntos distintos de atributos. A implementação foi adaptada da versão do classificador Naive Bayes presente na ferramenta Weka, sem alterações além da seleção de atributos. As execuções foram conduzidas com os parâmetros default desse classificador, que são: useKernelEstimator=false e userSupervisedDiscretization=false.

A Tabela 4.8 mostra o número de vezes em que cada estratégia, *eager* ou *lazy*, alcançou uma acurácia superior à outra, para as nove execuções em que foi variada a percentagem de atributos escolhidos entre 10% e 90%, em incrementos de 10%. Como mostrado na última linha "Totais", os resultados mostram uma prevalência da estratégia *lazy* sobre a estratégia *eager*, apesar dessa superioridade ser menos acentuada do que a observada com o k-NN.

Na Tabela 4.9, as colunas "Eager" e "Lazy" indicam as melhores acurácias obtidas para cada estratégia de seleção pelo classificador Naive Bayes, de forma similar ao que foi mostrado na Tabela 4.6 para o classificador k-NN. O número entre parênteses indica a percentagem de atributos selecionados em que essa acurácia foi obtida. A coluna "SemSel"

Tabela 4.8: Número de execuções em que cada estratégia obteve o melhor resultado utilizando o classificador Naive Bayes

Base	Naive Bayes		
	Eager	Lazy	Empate
anneal	3	**4**	2
audiology	0	**8**	1
autos	**7**	2	0
breast-cancer	**7**	2	0
breast-w	2	**6**	1
chess-kr-vs-kp	**5**	4	0
credit-a	4	**5**	0
diabetes	**4**	2	3
flags	**6**	1	2
glass	2	**5**	2
heart-cleveland	3	**4**	2
heart-hungarian	3	1	**5**
hepatitis	1	**7**	1
horse-colic	**5**	4	0
hypo-thyroid	**8**	1	0
ionosphere	2	**7**	0
labor	2	**4**	3
letter	0	**9**	0
lymph	2	**6**	1
molecular-biology-promoters	**6**	3	0
molecular-biology-splice	**9**	0	0
mushroom	3	**5**	1
optdigits	0	**8**	1
pendigits	0	**9**	0
postoperative	1	**5**	3
primary-tumor	1	**8**	0
solar-flare1	**5**	4	0
solar-flare2	**5**	3	1
sonar	2	**7**	0
soybean-large	1	**7**	1
spambase	0	**9**	0
statlog-heart	3	3	3
statlog-segment	1	**7**	1
statlog-vehicle	1	**8**	0
thyroid-sick	1	**5**	3
vote	**5**	3	1
vowel	**4**	3	2
waveform-5000	1	3	**5**
wine	0	**8**	1
zoo	3	**5**	1
TOTAIS	118	**195**	47

representa a acurácia obtida quando nenhuma seleção de atributos é realizada. Para cada base de dados, os valores em negrito indicam os melhores resultados. Na última linha da tabela, estão sumarizados os totais de bases de dados em que cada estratégia de seleção obteve o melhor resultado.

A estratégia *lazy* atingiu a melhor acurácia em 27 casos e a técnica *eager* em 24 casos. Esses resultados confirmam que um número relevante de bases de dados pode se beneficiar da seleção *lazy*.

Finalmente, um teste estatístico *t–Student* foi executado para os resultados obtidos com o classificador Naive Bayes, com os mesmos parâmetros utilizados anteriormente para os resultados do k-NN, ou seja, $p = 0{,}05$, com uma configuração bi-caudal e pareada. Os

Tabela 4.9: As melhores acurácias preditivas alcançadas por cada estratégia utilizando o classificador Naive Bayes

Base de Dados	**Eager**		**Lazy**		**Sem Sel.**
anneal	94,9	(80)	**96,3**	(10)	94,9
audiology	74,3	(40)	**76,6**	(60)	73,0
autos	**81,0**	(20)	77,6	(20)	72,7
breast-cancer	**73,1**	(40)	72,4	(30)	71,7
breast-w	**97,6**	(90)	**97,6**	(90)	97,0
chess-kr-vs-kp	89,6	(20)	**91,2**	(20)	87,7
credit-a	**86,7**	(70)	86,5	(70)	86,4
diabetes	78,8	(60)	**79,0**	(60)	78,1
flags	**62,9**	(60)	61,9	(90)	61,3
glass	**74,3**	(80)	**74,3**	(80)	**74,3**
heart-cleveland	**84,5**	(20)	84,2	(50)	82,8
heart-hungarian	**84,0**	(20)	**84,0**	(50)	**84,0**
hepatitis	85,8	(50)	**86,5**	(40)	83,9
horse-colic	**85,6**	(20)	84,5	(20)	82,1
hypo-thyroid	**95,3**	(60)	94,8	(80)	**95,3**
ionosphere	92,0	(50)	**92,6**	(10)	90,9
labor	**98,3**	(60)	**98,3**	(70)	**98,3**
letter	74,3	(70)	**74,7**	(70)	74,0
lymph	**86,5**	(70)	**86,5**	(80)	**86,5**
mol-bio-promot	**94,3**	(10)	93,4	(80)	93,4
mol-bio-splice	**96,1**	(60)	95,3	(90)	95,5
mushroom	98,8	(10)	**99,4**	(10)	95,5
optdigits	**92,5**	(90)	**92,5**	(70)	**92,5**
pendigits	87,2	(90)	87,5	(90)	**87,8**
postoperative	**71,1**	(20)	**71,1**	(20)	70,0
primary-tumor	49,6	(80)	**50,7**	(80)	49,3
solar-flare1	**72,8**	(30)	72,1	(30)	65,0
solar-flare2	75,2	(30)	**75,7**	(10)	74,8
sonar	68,3	(10)	**71,6**	(10)	67,8
soybean-large	89,9	(80)	**90,0**	(30)	89,9
spambase	90,5	(20)	**91,6**	(10)	90,2
statlog-heart	**84,4**	(50)	**84,4**	(40)	83,0
statlog-segment	**91,6**	(90)	**91,6**	(90)	**91,6**
statlog-vehicle	62,9	(60)	**63,0**	(70)	62,2
thyroid-sick	**97,2**	(10)	**97,2**	(30)	97,0
vote	**95,2**	(10)	94,3	(10)	90,1
vowel	**57,7**	(70)	**57,7**	(70)	52,7
waveform-5000	**81,1**	(40)	81,1	(40)	80,7
wine	98,9	(90)	**100,0**	(30)	98,9
zoo	**96,0**	(80)	**96,0**	(80)	93,1
TOTAIS	24		27		8

resultados são os seguintes: a estratégia *eager* prevaleceu 118 vezes sobre a estratégia *lazy*, mas apenas 25 vezes com significância estatística. A estratégia *lazy* superou a *eager* 195 vezes, e dessas, 80 vezes com significância estatística. Em 47 testes ocorreu um empate.

Mais uma vez, a estratégia *lazy* se mostrou mais eficiente, em termos de acurácia, do que a estratégia comparada para diversas bases de dados, reforçando a sua aplicabilidade a diferentes tipos de classificadores.

4.3 Resultados Experimentais com o HiSP

Um terceiro classificador com características *lazy* foi utilizado para validar a estratégia de seleção de atributos proposta. O HiSP (*Highest Subset Probability*) [28] é uma técnica de classificação que se baseia no teorema de Bayes, e avalia todos os subconjuntos de valores de atributos específicos da instância que se deseja classificar. Os subconjuntos com as maiores probabilidades de pertencerem a determinadas classes definem a classificação da nova instância.

No entanto, dependendo das dimensões da base de dados e das características da instância, o custo computacional do processamento desses subconjuntos pode ser muito elevado. Assim, para viabilizar a utilização do HiSP para certas bases de dados, é necessário reduzi-las por meio de uma etapa de seleção de atributos. Em [28], foram empregadas três técnicas *eager* de seleção de atributos para avaliar 18 bases de dados que continham mais de 15 atributos.

Como para a maioria das bases o HiSP só pode ser executado em um tempo razoável com 16 ou menos atributos, os experimentos com esse classificador foram conduzidos de forma diferente do que os experimentos com os demais: as bases de dados foram divididas em dois grupos: o Grupo 1 com bases de até 16 atributos, e o Grupo 2 com bases de mais de 16 atributos. O número de atributos a serem selecionados variou de 10% a 90%, em incrementos de 10%. Para as bases do Grupo 1 foram executados todos os nove testes. Para as bases do Grupo 2, foram realizadas tantas execuções quanto possíveis, limitadas pelo limite de 16 atributos. Consequentemente, o número total de execuções com o classificador HiSP foi menor do que com os demais classificadores: 270 experimentos para as mesmas 40 bases de dados. As análises dessas execuções serão feitas separadamente.

Ao se comparar o número de vezes em que cada estratégia, *eager* ou *lazy*, alcançou uma acurácia superior à da outra, para cada execução individual em uma base de dados com um valor distinto de atributos selecionados, a estratégia *lazy* mostrou novamente resultados favoráveis, expressos nas Tabelas 4.10 e 4.11, para as bases do Grupo 1 e 2, respectivamente.

Nessas duas tabelas, a segunda e a terceira coluna mostram o número de vezes em que cada estratégia, *eager* ou *lazy*, alcançou uma acurácia superior à outra, para as nove execuções em que foi variada a percentagem de atributos escolhidos entre 10% e 90%, em incrementos de 10%. A quarta coluna mostra o número de vezes em que ocorreu

Tabela 4.10: Número de execuções em que cada estratégia obteve o melhor resultado utilizando o classificador HiSP, para as bases do Grupo 1

Base	HiSP Grupo 1		
	Eager	Lazy	Empate
breast-cancer	4	**5**	0
breast-w	2	**6**	1
credit-a	**6**	3	0
diabetes	**6**	0	3
glass	0	**7**	2
heart-cleveland	3	3	3
heart-hungarian	3	**4**	2
labor	0	**8**	1
letter	0	**9**	0
pendigits	0	**9**	0
postoperative	0	0	**9**
solar-flare1	**5**	4	0
solar-flare2	**5**	2	2
statlog-heart	2	**4**	3
vote	3	**5**	1
vowel	2	**4**	3
wine	0	**8**	1
TOTAIS	41	**81**	31

Tabela 4.11: Número de execuções em que cada estratégia obteve o melhor resultado utilizando o classificador HiSP, para as bases do Grupo 2

Base	HiSP Grupo 2		
	Eager	Lazy	Empate
anneal	0	**4**	0
audiology	0	**2**	0
autos	**4**	2	0
chess-kr-vs-kp	0	**4**	0
flags	2	**3**	0
hepatitis	3	**5**	0
horse-colic	3	3	0
hypo-thyroid	0	**5**	0
ionosphere	0	**4**	0
lymph	**7**	2	0
mol-bio-promoters	1	1	0
mol-bio-splice	**2**	0	0
mushroom	0	**4**	3
optdigits	0	**2**	0
primary-tumor	2	**7**	0
sonar	1	1	0
soybean-large	0	**4**	0
spambase	0	**2**	0
statlog-segment	0	**8**	0
statlog-vehicle	2	**7**	0
thyroid-sick	**4**	0	0
waveform-5000	1	**3**	0
zoo	0	**6**	3
TOTAIS	32	**79**	6

um empate na execução das duas estratégias. A última linha, "Totais", confirma uma prevalência da estratégia *lazy* sobre a estratégia *eager*: para as 17 bases do Grupo 1, que seguiram o mesmo modelo de execução efetuado para os demais classificadores, a estratégia *lazy* apresentou o melhor desempenho 81 vezes, enquanto a *eager* obteve um resultado melhor 41 vezes. Ocorreram também 31 empates.

Para as demais 23 bases do Grupo 2, foram executadas tantos experimentos quanto possíveis, de acordo com o número de atributos selecionados. A estratégia *lazy* alcançou resultados superiores 79 vezes, contra 32 da *eager*. Em 6 execuções ocorreu um empate. É importante observar que nesse experimento específico, o número de execuções totais é diferente para cada base de dados.

Totalizando os resultados das tabelas, os seguintes resultados foram alcançados: do total de 270 execuções, a *lazy* foi melhor em 160 vezes contra 73 da *eager*, e 37 empates. Se forem considerados apenas os resultados com significância estatística, a exemplo dos experimentos conduzidos com os outros classificadores, a estratégia *lazy* prevaleceu em 60 execuções, contra apenas 10 da *eager*, e os demais 200 casos foram iguais ou com uma diferença sem significância estatística.

As Tabelas 4.12 e 4.13 comparam os resultados de acurácia obtidos com o HiSP para as bases de dados do Grupo 1 e do Grupo 2, respectivamente. Para cada base de dados do experimento, as colunas "Eager" e "Lazy" indicam a melhor acurácia preditiva obtida com o classificador HiSP após a seleção de atributos com a técnica *lazy* proposta neste trabalho e a técnica *eager*, também empregada no trabalho original do HiSP [28] para as bases com muitos atributos, *Information Gain Attribute Ranking*. Entre parênteses é indicado o número percentual de atributos em que essa acurácia foi obtida. Nas bases do Grupo 1 estão indicados os resultados sem seleção de atributos na coluna "Sem Sel.". Para as bases do Grupo 2 não foi possível executar o experimento sem seleção de atributos, por exceder o limite imposto de 16 atributos.

Tabela 4.12: As melhores acurácias preditivas alcançadas por cada estratégia utilizando o classificador HiSP para as bases do Grupo 1

Base de Dados	**Eager**		**Lazy**		**Sem Sel.**
breast-cancer	**76,22%**	(90%)	75,87%	(70%)	**76,22%**
breast-w	**97,14%**	(70%)	97,00%	(90%)	**97,14%**
credit-a	**87,25%**	(90%)	86,96%	(80%)	86,23%
diabetes	**77,99%**	(60%)	77,86%	(70%)	77,86%
glass	73,36%	(80%)	73,36%	(30%)	73,36%
heart-cleveland	**84,49%**	(20%)	83,50%	(40%)	82,18%
heart-hungarian	**83,33%**	(20%)	82,99%	(20%)	82,65%
labor	98,25%	(60%)	**100,00%**	(60%)	100,00%
letter	93,90%	(90%)	**94,02%**	(90%)	93,98%
pendigits	97,23%	(90%)	**97,64%**	(90%)	97,89%
postoperative	71,11%	(20%)	71,11%	(20%)	71,11%
solar-flare1	72,14%	(30%)	**73,68%**	(30%)	70,28%
solar-flare2	**75,05%**	(40%)	74,20%	(30%)	73,55%
statlog-heart	85,19%	(70%)	85,19%	(40%)	85,19%
vote	**95,17%**	(30%)	94,94%	(90%)	94,25%
vowel	90,40%	(90%)	90,40%	(90%)	90,20%
wine	99,44%	(90%)	99,44%	(40%)	99,44%
TOTAIS	8		4		2

Tabela 4.13: As melhores acurácias preditivas alcançadas por cada estratégia utilizando o classificador HiSP para as bases do Grupo 2

Base de Dados	Eager		Lazy	
anneal	98,44%	(40%)	**98,55%**	(40%)
audiology	72,12%	(20%)	**76,55%**	(20%)
autos	**89,27%**	(60%)	88,29%	(60%)
chess-kr-vs-kp	95,93%	(40%)	**96,15%**	(40%)
flags	61,86%	(40%)	**63,40%**	(30%)
hepatitis	85,81%	(40%)	**86,45%**	(70%)
horse-colic	82,61%	(20%)	82,61%	(40%)
hypo-thyroid	93,43%	(10%)	**93,98%**	(10%)
ionosphere	93,16%	(40%)	93,16%	(30%)
lymph	**87,16%**	(70%)	85,81%	(70%)
mol-bio-promoters	**94,34%**	(10%)	90,57%	(20%)
mol-bio-splice	**95,14%**	(20%)	90,00%	(10%)
mushroom	100,00%	(50%)	100,00%	(30%)
optdigits	90,57%	(20%)	**91,42%**	(20%)
primary-tumor	**47,79%**	(70%)	47,20%	(90%)
sonar	71,63%	(20%)	71,63%	(10%)
soybean-large	86,09%	(40%)	**89,75%**	(40%)
spambase	92,39%	(20%)	**92,50%**	(20%)
statlog-segment	95,76%	(70%)	**95,84%**	(70%)
statlog-vehicle	71,75%	(50%)	**72,22%**	(90%)
thyroid-sick	**96,29%**	(10%)	95,15%	(10%)
waveform-5000	**84,02%**	(40%)	83,72%	(40%)
zoo	97,03%	(80%)	97,03%	(80%)
TOTAIS	7		11	

Os resultados de melhor acurácia do HiSP utilizando a seleção *lazy* indicam uma supremacia em relação à seleção *eager* para as bases do Grupo 2, com mais atributos, o que não ocorre para as bases do Grupo 1, com menos atributos. Isso indica que a estratégia *lazy* pode ser adequada para bases com muitos atributos, em que certamente a seleção de atributos é importante para o HiSP ou mesmo outros classificadores.

Portanto, a estratégia *lazy* permanece como uma opção interessante para viabilizar o uso do classificador HiSP para bases de dados de alta dimensionalidade.

4.4 Previsão da eficiência da estratégia lazy

Em relação aos resultados apresentados nas seções anteriores, pode-se perceber que, para várias bases, a seleção *lazy* obteve um melhor desempenho, porém, para outras a seleção *eager* se mostrou mais adequada. Consequentemente, seria interessante dispor de um método para estimar em que condições a seleção *lazy* seria mais eficiente.

Intuitivamente, a estratégia *lazy* é mais oportuna quando há uma grande variação na entropia de cada valor de um atributo. Quando esse é o caso, alguns atributos têm mais chance de serem adequados para algumas instâncias, mas não para outras. Por conseguinte, propõe-se avaliar, para cada atributo A_j, $1 \leq j \leq n$, de uma base

$D(A_1, A_2, ..., A_n, C)$, $n \geq 1$, a média dos módulos (valores absolutos) das diferenças entre o valor $H(D, A_j)$ e cada um dos valores $H(D, A_j, a_{ji})$, $1 \leq i \leq k_j$, para cada valor a_{ji} do atributo A_j. Esse valor médio é definido por:

$$V(D, A_j) = \frac{1}{k_j} * \sum_{i=1}^{k_j} [|H(D, A_j, a_{ji}) - H(D, A_j)|]. \quad (4.1)$$

Para cada base de dados, é possível calcular a métrica $V(D)$ tomando a média dos valores $V(D, A_j)$ para cada um dos atributos A_j. Quanto maior esse valor, mais provável será o benefício introduzido pela estratégia *lazy* para a base D.

Porém, para definir uma métrica mais específica e apropriada, é necessário levar em conta o número de atributos a serem selecionados. Portanto, para uma base de dados D e uma percentagem x de atributos a serem selecionados, calcula-se a métrica $V(D, x)$ pela média dos valores $V(D, A_j)$ do conjunto de atributos com os $x\%$ maiores valores de $V(D, A_j)$.

De forma análoga, quanto maior o valor de $V(D, x)$, mais provável será a aplicabilidade da estratégia *lazy* para a base D, ao se realizar a seleção de $x\%$ dos atributos.

A Figura 4.1 mostra que um valor alto para essa métrica realmente implica em uma convicção maior da superioridade da seleção *lazy* de atributos. A análise é baseada em todas as execuções das estratégias *eager* e *lazy* com o classificador 1-NN, considerando todas as 40 bases (D) e a variação de percentagem (x) de atributos selecionados de 10% a 90%, em incrementos de 10%, o que representa um número máximo de 360 execuções por estratégia.

Para cada limiar do eixo horizontal, o valor correspondente no eixo vertical, na Curva 1, representa o percentual de casos em que a estratégia *lazy* obteve uma acurácia preditiva superior a da estratégia *eager*, considerando entre os 360 casos aqueles em que o respectivo valor $V(D, x)$ é maior ou igual do que o valor correspondente do eixo horizontal. Na Curva 2 é feita a comparação apenas para os casos com significância estatística, também entre as 360 execuções.

O ponto mais à esquerda da Curva 1 indica, por exemplo, que, para todas as combinações de D e x em que a métrica $V(D, x) \geq 0$, a estratégia *lazy* alcançou, em 61,1% das vezes (220 de 360 execuções), resultados melhores do que a estratégia *eager*. Quando os experimentos foram restritos às combinações de D e valores x em que a métrica $V(D, x) \geq 0,2$, a estratégia *lazy* prevaleceu em 64,9% dos casos (148 de 228 execuções),

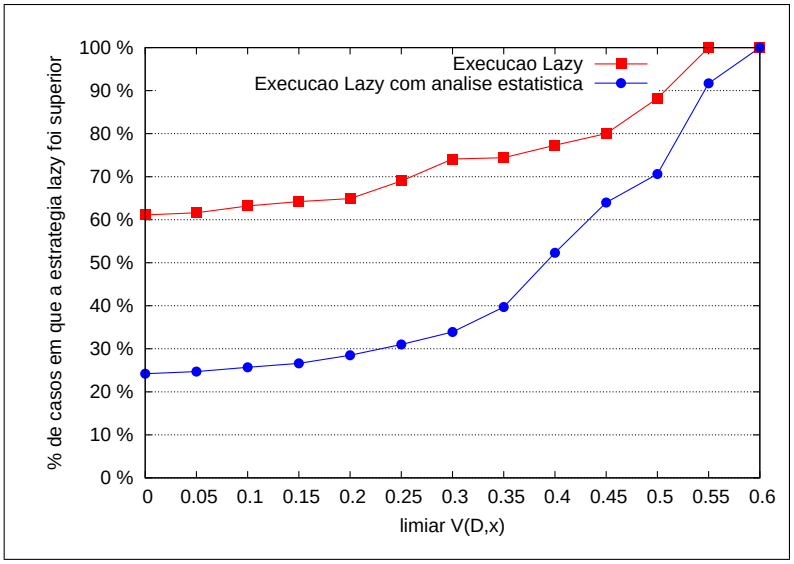

Figura 4.1: Avaliação da prevalência Lazy com a métrica $V(D, x)$ para o 1-NN.

e para $V(D, x) \geq 0,4$, os resultados indicam 77,3% de execuções em que a *lazy* superou a *eager* (ou 34 de 44 execuções totais).

O mesmo comportamento ocorre para os experimentos conduzidos com a análise de significância estatística, representada pela Curva 2: quanto maior o valor de $V(D, x)$, mais provável a chance de a estratégia de seleção *lazy* alcançar um resultado melhor que a estratégia *eager*.

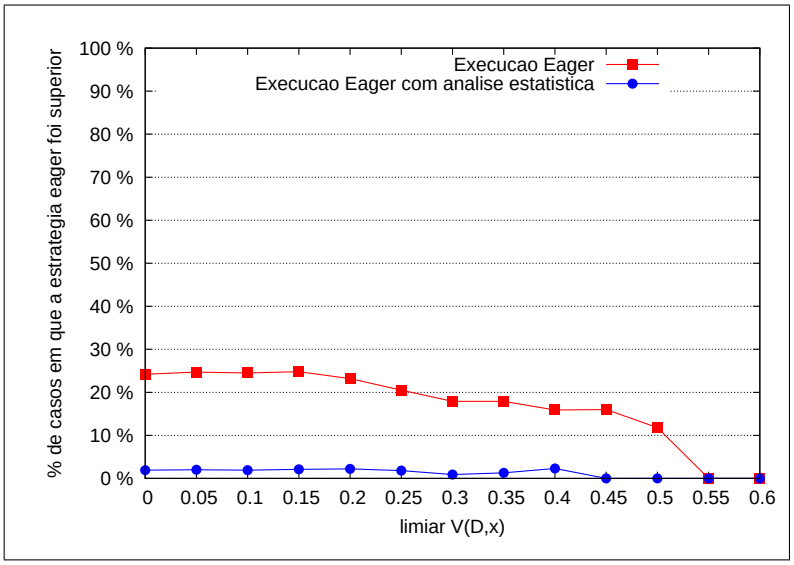

Figura 4.2: Avaliação da prevalência Eager com a métrica $V(D, x)$ para o 1-NN.

A Figura 4.2, de forma análoga à figura anterior, representa o percentual de casos em que a estratégia *eager* obteve uma acurácia preditiva superior à da estratégia *lazy*, considerando as mesmas execuções com o classificador 1-NN. As duas curvas indicam que,

quanto maior o valor da métrica $V(D, x)$, menor a eficácia da estratégia *eager* em relação à *lazy*.

Resultados similares foram obtidos com os demais classificadores. Isso indica que, de fato, a métrica pode ser utilizada para estimar com mais convicção se a estratégia *lazy* proposta é capaz de alcançar um resultado superior para uma base de dados específica, dada a percentagem de atributos a serem selecionados.

4.5 Avaliação do tempo computacional

A estratégia de seleção de atributos *eager* faz a seleção em uma etapa de pré-processamento dos dados, e portanto não adiciona tempo computacional na classificação dos exemplos (pelo contrário, diminui). Já na seleção *lazy* não há pré-processamento, porém existe um custo adicional de se fazer a seleção de atributos para cada exemplo. O objetivo dessa seção é examinar o impacto desse tempo computacional no tempo total de classificação.

Todos os experimentos foram conduzidos em uma máquina Intel Core 2 Duo CPU 4400 2.0 GHz com 2 Gbytes de RAM.

O tempo de execução médio do procedimento de seleção *lazy* para cada instância variou de 0,08 milisegundos (para a base de dados *Autos*) até 0,31 milisegundos (para a base *Lymph*). Esse tempo pode ser considerado desprezível, pois para essas mesmas bases de dados, o tempo de execução da classificação de uma instância utilizando o 3-NN (por exemplo) foi de 1,7 e 10,4 milisegundos, respectivamente.

Portanto, o custo computacional introduzido pela estratégia *lazy* na tarefa de classificação do k-NN, Naive Bayes e HiSP pode ser considerado irrelevante.

No caso do HiSP, a seleção de atributos viabiliza a classificação de instâncias pertencentes a bases com um número elevado de atributos. A estratégia de seleção *lazy*, além de permitir a utilização do HiSP para determinadas bases de dados, pode ser uma alternativa mais interessante do que estratégias *eager*.

Capítulo 5

Conclusões

Nesta dissertação, foi proposto o uso de uma estratégia *lazy* de seleção de atributos para o problema de classificação. Embora a estratégia seja genérica, foi implementada uma versão que se baseia no ranqueamento de atributos utilizando a medida de entropia. Essa implementação foi comparada com a estratégia *eager* análoga.

A avaliação da estratégia foi realizada a partir de quarenta bases de domínio público adotadas frequentemente em experimentos de mineração de dados, obtidas do repositório de dados UCI *Machine Learning Repository*.

Os resultados experimentais mostraram que adiar a escolha dos atributos ao momento em que uma nova instância está pronta para ser classificada, melhorou a acurácia preditiva da classificação em relação à estratégia *eager*, para a maioria dos casos testados. As técnicas de classificação k-NN, Naive Bayes e HiSP foram empregadas nos experimentos. Esses foram submetidos a um teste de análise de significância estatística para validar a sua superioridade em relação aos resultados *eager*. Os experimentos mostraram também a viabilidade do método em relação ao custo computacional adquirido pela incorporação do procedimento *lazy* à classificação.

Neste trabalho, também foi proposta uma métrica para estimar se uma base de dados específica pode se favorecer da estratégia de seleção de atributos *lazy*. Os resultados experimentais demonstraram que um valor alto dessa métrica está correlacionado com uma chance maior da base de dados se beneficiar da estratégia *lazy* em relação à estratégia *eager*.

Portanto, os resultados obtidos nos experimentos deste trabalho demonstraram que a seleção de atributos *lazy* é uma estratégia que pode ser utilizada de forma eficiente para diferentes tipos de bases de dados.

A implementação da técnica *lazy* desta dissertação foi desenvolvida tomando por base uma técnica *eager* que utiliza o conceito de entropia para avaliar os atributos de maneira individual. Apesar de a entropia ser, em geral, uma boa medida para avaliar a relevância de um atributo, seria interessante avaliar a eficácia da adaptação para a abordagem *eager* de outras medidas de ranqueamento para a seleção de atributos, tais como *gain ratio*, *chi square* ou *gini index*. Desse modo, como trabalho futuro, está planejada a condução de experimentos com outras medidas de avaliação de atributos.

Além disso, pretende-se estender a implementação da seleção *lazy* para estratégias filtro que avaliam subconjuntos de atributos, ao invés de examiná-los de maneira individual. Planeja-se também a extensão da concepção *lazy* para técnicas de seleção de atributos do tipo *wrapper*, que tradicionalmente alcançam resultados superiores do que a seleção filtro, em termos de acurácia preditiva do classificador.

Finalmente, propõe-se como extensão deste trabalho um procedimento para estimar o melhor parâmetro r, que determina o número de atributos a serem selecionados pela técnica *lazy*. Esse parâmetro influi nos resultados da classificação e, embora não seja um problema específico da estratégia *lazy*, espera-se que um procedimento automático para estimá-lo possa aperfeiçoar o processo de classificação.

Referências

[1] Asuncion, A., Newman, J. UCI machine learning repository. http://www.ics.uci.edu/∼mlearn/MLRepository.html, 2007. University of California, Irvine.

[2] Blum, A. L., Langley, P. Selection of relevant features and examples in machine learning. *Artificial Intelligence 97* (1997), 245–271.

[3] Breiman, L. Bagging predictors. *Machine Learning 24* (1996), 123–140.

[4] Breiman, L. Random forests. *Machine Learning 45* (2001), 5–32.

[5] Burges, C. J. C. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery 2* (1998), 121–168.

[6] Cover, T., Hart, P. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory 13* (1967), 21–27.

[7] Dasarathy, B. V. *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*. IEEE Computer Society Press, 1991.

[8] Dash, M., Liu, H. Consistency-based search in feature selection. *Artificial Intelligence 151* (2003), 155–176.

[9] D.J. Hand, H. M., Smyth, P. *Principles of Data Mining*. Bradford Book, Cambridge, 2000.

[10] Doak, J. An evaluation of feature-selection methods and their application to computer security. *Technical Report CSE-92-18 18* (1992).

[11] Duda, R. O., Hart, P. E., Stork, D. G. *Pattern Classification*, 2nd ed. John Wiley & Sons, 2001.

[12] Fayyad, U. M., Irani, K. B. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence* (Chambéry, France, 1993), Morgan Kaufmann, p. 1022–1029.

[13] Friedman, J. H., Kohavi, R., Yun, Y. Lazy decision trees. In *Proceedings of the National Conference on Artificial Intelligence*.

[14] Guyon, I., Elisseeff, A. An introduction to variable and feature selection. *Journal of Machine Learning Research 3* (2003), 1157–1182.

[15] Guyon, I., Elisseeff, A. An introduction to feature extraction. In *Feature Extraction, Foundations and Applications*, I. Guyon, S. Gunn, M. Nikravesh, and L. Zadeh, Eds. Springer, 2006, p. 1–24.

[16] Guyon, I., Gunn, S., Nikravesh, M., Zadeh, L., Eds. *Feature Extraction, Foundations and Applications*. Springer, 2006.

[17] Hall, M. A. A correlation-based feature selection for discrete and numeric class machine learning. In *Proceedings of the 17th International Conference on Machine Learning (ICML'00)* (2000).

[18] Han, J., Kamber, M. *Data Mining: Concepts and Techniques*, 2nd ed. Morgan Kaufmann, 2006.

[19] Kira, K., Rendell, L. A practical approach to feature selection. In *Proceedings of the 9th International Conference on Machine Learning (ICML'92)* (1992), p. 249–256.

[20] Kononenko, I. Estimating attributes: Analysis and extensions of relief. In *Proceedings of the 7th European Conference on Machine Learning (ECML'94)* (1994), p. 171–182.

[21] Kononenko, I. On biases in estimating multi-valued attributes: Analysis and extensions of relief. In *Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI 95)* (1995), p. 1034–1040.

[22] L. Breiman, J. H. Friedman, R. A. O., Stone, C. J. Classification and regression trees.

[23] Liu, B., Hsu, W., Ma, Y. Integrating classification and association rule mining. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD'98)* (1998), p. 80–86.

[24] Liu, H., Motoda, H. *Computational Methods of Feature Selection*. Chapman & Hall/CRC, 2008.

[25] Liu, H., Motoda, H. Less is more. In *Computational Methods of Feature Selection*, H. Liu and H. Motoda, Eds. Chapman & Hall/CRC, 2008, p. 3–17.

[26] Liu, H., Setiono, R. A probabilistic approach to feature selection: A filter solution. In *Proceedings of the 13th International Conference on Machine Learning (ICML'96)* (1996), p. 319–327.

[27] Liu, H., Yu, L. Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on Knowledge and Data Engineering 17*, 4 (2005).

[28] Merschmann, L. H. C. *Classificação Probabilística baseada em Análise de Padrões*. PhD thesis, Universidade Federal Fluminense, Agosto 2007.

[29] P. Langley, S. S. Oblivious decision trees and abstract cases. In *Working Notes of the AAAI94 Workshop on Case-Based Reasoning* (1994), p. 113–117.

[30] Quinlan, J. R. Learning efficient classification procedures and their application to chess end games. In *Machine learning: An artificial intelligence approach*, . T. M. M. R. S. Michalski, J. G. Carbonell, Ed. Morgan Kaufmann, San Francisco, CA, 1983, p. 3–17.

[31] Quinlan, J. R. Induction of decision trees. *Machine Learning 1* (1986), 81–106.

[32] Quinlan, J. R. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[33] Reunanen, J. Overfitting in making comparisons between variable selection methods. *The Journal of Machine Learning Research 3* (2003), 1371–1382.

[34] Ripley, B. D. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.

[35] Segal, M. R. Machine learning benchmarks and random forest regression. http://www.ics.uci.edu/∼mlearn/MLRepository.html, 2004. Center for Bioinformatics & Molecular Biostatistics.

[36] Veloso, A., Meira Jr., W., Zaki, M. J. Lazy associative classification. In *Proceedings of the Sixth International Conference on Data Mining (ICDM'06)* (Washington, DC, USA, 2006), IEEE Computer Society, p. 645–654.

[37] W.H. Press, S.A. Teukolsky, W. V., Flannery, B. *Numerical recipes in C. The art of scientific computing*. Cambridge University Press, 1988.

[38] Witten, I. H., Frank, E. *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd ed. Morgan Kaufmann, 2005.

[39] Wu, Y., Zhang, A. Feature selection for classifying high-dimensional numerical data. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2004)* (2004), vol. 2, p. 251–258.

[40] Yang, Y., Pedersen, J. O. A comparative study on feature selection in text categorization. In *Proceedings of the 14th International Conference on Machine Learning (ICML'97)* (1997), p. 412–420.