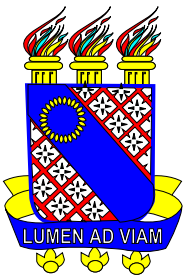

UNIVERSIDADE ESTADUAL DO CEARÁ

RAFAEL AUGUSTO FERREIRA DO CARMO

SELEÇÃO CONJUNTA DE ATRIBUTOS E EXEMPLOS EM MINERAÇÃO DE DADOS

FORTALEZA, CEARÁ

2010

RAFAEL AUGUSTO FERREIRA DO CARMO

SELEÇÃO CONJUNTA DE ATRIBUTOS E EXEMPLOS EM MINERAÇÃO DE DADOS

Dissertação apresentada no Curso de Mestrado Acadêmico em Ciência da Computação da Universidade Estadual do Ceará – UECE, como requisito parcial para obtenção de título de Mestre em Ciência da Computação.

Orientador: Jerffeson Teixeira de Souza

FORTALEZA, CEARÁ

2010

C287s Carmo, Rafael Augusto Ferreira do.
Seleção Conjunta de Atributos e Exemplos em Mineração de Dados / Rafael Augusto Ferreira do Carmo. – Fortaleza, 2010.
60p.;il.
Orientador: Prof. Dr. Jerffeson Teixeira de Souza
Dissertao (Mestrado Acadêmico em Ciência da Computação) - Universidade Estadual do Ceará, Centro de Ciências Tecnologia - CCT.
1. seleção 2. atributos 3. exemplos I. Universidade Estadual do Ceará, Centro de Ciências Tecnologia - CCT.

CDD:001.6

RAFAEL AUGUSTO FERREIRA DO CARMO

SELEÇÃO CONJUNTA DE ATRIBUTOS E EXEMPLOS EM MINERAÇÃO DE DADOS

Dissertação apresentada no Curso de Mestrado Acadêmico em Ciência da Computação da Universidade Estadual do Ceará – UECE, como requisito parcial para obtenção de título de Mestre em Ciência da Computação.

Aprovada em: __/__/____

BANCA EXAMINADORA

Prof. Dr. Jerffeson Teixeira de Souza
Universidade Estadual do Ceará - UECE
Orientador

Prof. Dr. Thelmo Pontes de Araújo
Universidade Estadual do Ceará - UECE

Prof. Dr. Marcelino Pereira dos Santos Silva
Universidade Estadual do Rio Grande do Norte - UERN

AGRADECIMENTOS

Uma dissertação como esta, apesar de nominalmente ser de uma única pessoa, é fruto de um árduo trabalho coletivo. Desta forma devo agradecer, não exclusivamente, à meus pais que por tantos anos têm se dedicado com todo afinco à educação minha e da minha irmã Maira. Minha mãe com seus dois ou até três empregos me deu todo o suporte necessário, e até mesmo bem mais que isso, para que sempre as melhores oportunidades fossem me dadas e praticamente tudo o que fiz e faço até hoje é fruto deste sacrifício hercúleo feito por ela. Meu pai apesar da distância física, sempre esteve presente em minha vida, com seus conselhos, orientações e ajuda me guiando por caminhos que invariavelmente se mostraram corretos, além de ser exemplo de competência e persistência para alcançar seus objetivos. Também não posso deixar de citar meus avôs e minhas avós, todos, todos sem exceção, com história de vida belíssimas que me fascinam e inspiram a continuar construindo esta história de vida tão rica. Como digo para 'Zé di Nezim', ainda vou escrever um livro sobre a vida de vocês. Todos os outros milhões de membros da minha família também fazem parte deste trabalho, contribuindo de uma forma ou de outra, mas sempre contribuindo.

Não posso deixar de citar meu orientador Jerffeson. Alguns anos atrás quando começamos esta relação eu não poderia pensar que ela seria tão duradoura e geraria tantos bons frutos. Obrigado por ter apostado em mim, espero que esta aposta tenha valido a pena e que possamos continuar trabalhando conjuntamente por bastante tempo. Falando em apostas, o querido professor Gustavo não pode passar despercebido. Em 2004 éramos todos novatos na UECE e ele também apostou em mim e em meus colegas. Anos muito bons estes de maratona, não fomos muito longe mas nos divertimos muito - quem sabe isso não foi mais importante do que qualquer resultado de ponta. Além deles, todos os outros professores merecem meu agradecimento. O pessoal da secretaria tanto da graduação quando da pós também merece estar aqui, eu tanto os perturbei e eles sempre foram muito solícitos comigo. Verdadeiros colegas em toda esta jornada.

Vinícius e Joyce sabem que sempre estarão em todo agradecimento que eu fizer. A equação é simples: em algum momento do futuro não muito distante eu e Vinícius dominaremos o mundo e Joyce será nossa opositora, e que opositora viu! Eu pensei em fazer a lista feijão com arroz e dar a antiga desculpa para não citar todos os nomes mas vou fazer diferente, prefiro setorizar todo mundo e não citar ninguém, aqueles que se identificarem com um grupo saberão que foram aqui citados. Assim, devo lembrar daqueles que desde o ensino médio-técnico me perseguem e me dão ótimos momentos, aqueles que conheci durante a graduação, os do grupo PET, os do Larces, do Laconi (tá bom, Fabrício e Daniel eu não posso deixar de falar :D), do Lamac, da dança, da banda, os ex-vizinhos, os do mestrado. É muita gente!

Obrigado a todos e a Deus por esta dádiva chamada vida.

"A mente que se abre a uma nova idéia jamais voltará ao seu tamanho original."
Albert Einstein

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

1NN	1-Nearest Neighbor
CBR	Case-based Reasoning
CRISP-DM	Cross Industry Standard Process for Data Mining
DemoIS	Democratic Instance Selection
FortalFS	Fortal Feature Selection
ID3	Iterative Dichotomiser 3
kNN	k-Nearest Neighbors
RNA	Rede Neural Artificial
SVM	Support Vector Machines
SSVM	Smooth Support Vector Machines

1NN CBR CRISP-DM DemoIS FortalFS ID3 kNN RNA SVM SSVM

RESUMO

Esta dissertação propõe uma nova abordagem para o problema de seleção de dados, uma questão-chave no campo de mineração de dados. Esta abordagem, que é baseada em um algoritmo de seleção de atributos e um método de seleção de exemplos, reduz o conjunto original em duas dimensões, selecionando atributos relevantes e retendo instâncias importantes. Os processos de busca pelos melhores subconjuntos de atributos e de exemplos, ocorrem de forma independente mas, devido à influência dos atributos na importâncias dos exemplos e vice-versa, tais processos são influenciados um pelo outro. Diversos experimentos validam a abordagem proposta, mostrando o seu desempenho competitivo, tanto em consumo de tempo quanto em qualidade de soluções geradas.

ABSTRACT

This dissertation proposes a new approach to the data selection problem, a key issue in the data mining field. This approach, which is based on a feature selection algorithm and one instance selection method, reduces the original dataset in two dimensions, selecting relevant features and retaining important instances. The search processes for the best feature and instance subsets occur independently yet, due to the influence of features in the importance of instances and vice versa, they bias one another. Several experiments validate the proposed approach, showing its competitive performance, both in time consumption and quality of generated solutions.

1 INTRODUÇÃO

Vivemos na chamada Sociedade da Informação, uma forma de arranjo social na qual a chave do desenvolvimento científico e tecnológico está altamente relacionado com a capacidade de se transformar dados facilmente coletados e armazenados em grande volume em informação escassa, não-trivial e complexa, que possa ser compreendida e que agregue valor à atividade desenvolvida seja por uma empresa, uma universidade, uma instituição qualquer ou mesmo um empreendedor autônomo.

Imaginemos um Sistema Tutorial Inteligente que auxilie estudantes do ensino fundamental (ou similar) no estudo de álgebra. Este sistema é capaz de capturar em arquivos de log toda a interação do estudante com o sistema, registrando quantas vezes o estudante solicita auxílio, quantas vezes não responde corretamente a solução de um problema, etc. Se estudarmos o comportamento da iteração estudante-tutor, por exemplo, utilizando estes arquivos de log como guia e conseguirmos uma melhora no processo de ensino, que o torne em torno de 10% mais rápido, enxergaremos números um tanto interessantes:

- Se os estudantes gastarem, em média, 50 horas por ano utilizando o sistema então ao aplicarmos a melhoria introduzida através da mineração dos logs, economizaremos 5 horas de esforço para cada estudante.
- Se 500.000 estudantes utilizam este sistema, então 2,5 milhões de hora-estudante serão economizadas por ano, horas estas que podem ser utilizadas das mais diversas formas, seja para aprendizado de outra matéria ou até mesmo como tempo livre para o indivíduo.

A relevância deste tipo de estudo é tão grande que este cenário e estes dados estão presentes no desafio da KDD Cup 2010[1], competição internacional bastante importante dentro da comunidade de Mineração de Dados e áreas afins, patrocinada por grandes corporações da área de Informática tais como IBM e Facebook.

Este é um exemplo retirado do meio acadêmico na qual a transformação de dados em informação é bastante valiosa, mas além deste existem diversos outros exemplos que podemos retirar do nosso dia-a-dia que demonstram quão importante é a capacidade de efetuar esta transformação de forma eficaz.

Disposição de produtos em prateleiras de supermercados (atividade exemplifica pela lenda urbana da relação cerveja-fralda[2]), detecção de transações bancárias fraudulentas, valoração

[1] https://pslcdatashop.web.cmu.edu/KDDCup/ - Sítio acessado em Abril/2010.
[2] http://www.dssresources.com/newsletters/66.php - Sítio acessado em Abril/2010.

de seguros de automóveis, sistemas de recomendação de compras, programas de reconhecimento de faces são outros exemplos que inundam a literatura com áreas nas quais a transformação de dados em informação agrega valor à atividades diversas.

1.1 Motivação

Infelizmente tal transformação não ocorre de forma trivial. Seja qual for a técnica utilizada, ela provavelmente será cara, demandará um bom tempo de trabalho e não garantirá *a priori* nenhum resultado proveitoso à organização que se propõe a gerar informação a partir de dados coletados por ela ou por terceiros. Mesmo assim, aplicando-se um método bem estudado, estável e coerente com o que pode ser extraído dos dados que se tem em mãos, a probabilidade de se extrair algo interessante passa a ser relativamente grande.

Caso a abordagem escolhida seja a de Mineração de Dados, diversas são as atividades que cooperam para a maximização desta probabilidade. Um modelo de processo de Mineração de Dados reconhecido internacionalmente como padrão industrial que define muitas destas atividades é o CRISP-DM (CHAPMAN et al., 2000). De uma forma geral, estas atividades são distribuídas e ocorrem de acordo com o esquema mostrado na Figura 1.1. Este modelo de processo é bastante grande e aborda diversas áreas que não são relevantes para o desenvolvimento deste trabalho, portanto o foco de discussão e estudo será direcionado às atividades de Preparação de Dados (Data Preparation) e Modelagem (Modeling).

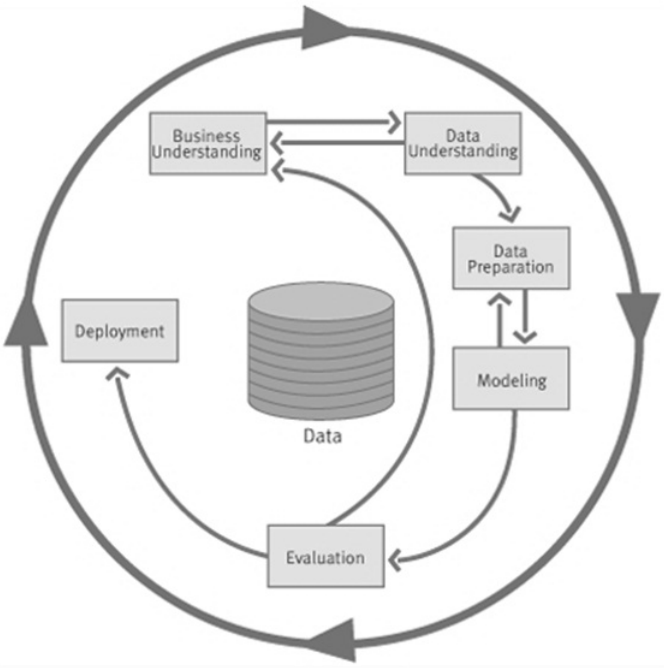

Figura 1.1: Fases do modelo de referência do CRISP-DM. Fonte: Chapman et al. (2000).

Diversas são as tarefas executadas nestas duas fases do processo de mineração de dados. Na fase de Preparação de Dados, deseja-se encontrar uma base de dados, ou seja, um conjunto de dados que represente de forma adequada o domínio do qual se quer obter novas

informações. Para tal, deve-se decidir que dados serão utilizados, listando razões pelas quais vai-se ou não incluí-los numa base de dados inicial, bem como executar tarefas automatizadas para tornar este conjunto o mais conciso possível para a tarefa de mineração. Já a fase de Modelagem busca-se produzir modelos computacionais que tragam à luz relações, fenômenos ou características presentes nestes dados. Tais modelos podem descrever relações entre objetos apresentados na base de dados (Regras de Associação), agrupar exemplos em conjuntos com alta similaridade (Clustering) ou deduzir uma função que descreva o conceito apresentado na base de dados (Classificação - área de desenvolvimento desta dissertação).

Voltando à fase de Preparação de Dados, uma das atividades chave desta fase é a atividade de seleção de dados que, como descrito em Chapman et al. (2000), consiste na escolha dos dados a serem analisados, levando-se em consideração a relevância para o objetivo da mineração de dados, qualidade destes dados e restrições técnicas e orçamentárias. Como dados utilizados para mineração são usualmente apresentados na forma de tabelas, esta seleção se dá tanto nas colunas (atributos) como nas linhas (exemplos).

É nesta atividade que será desenvolvida esta dissertação, através de uma descrição aprofundado da tarefa de seleção de dados, bem como – a partir do estudo das técnicas que realizam-na – a proposição uma abordagem para esta tarefa que possui características únicas quando comparadas a outras apresentadas na literatura.

1.2 Objetivos

Em linhas gerais, este trabalho visa a geração de uma nova abordagem para a atividade de seleção de dados para modelos de classificação em mineração de dados. Para atingir tal objetivo, alguns tópicos devem ser desenvolvidos:

- revisão da literatura referente à atividade de seleção de dados, bem como sua divisão em seleção de atributos e seleção de exemplos.
- descrição da problemática relacionado à esta atividade, mostrando as definições encontradas na literatura e abordando sua importância na tarefa de classificação.
- geração de uma nova abordagem, construída a partir da percepção de que há meios de se explorar este problema de uma forma diferente das já apresentadas na literatura.
- implementação de abordagens similares para comparação.
- execução de testes para validação da abordagem proposta.

1.3 Hipóteses e Questões de Pesquisa

A definição do problema de seleção de dados, de sua importância e características, bem como o estudo das abordagens propostas na literatura para resolução de tal problema mostram

que há possibilidades de resolver este problema que ainda não foram exploradas e que podem ser bastante interessantes para alguns cenários em mineração de dados.

O estudo da literatura mostra, como visto no capítulo 3, que as abordagens criadas até o momento, que lidam com seleção de dados como um todo, ou seja, realizam as tarefas de seleção de atributos e seleção de exemplos simultaneamente, estão intimamente ligadas ao uso de técnicas de resolução de problemas de otimização, tais como metaheurísticas, para guiar o processo de busca. Neste contexto, surge o questionamento "Por que não adaptarmos algoritmos de seleção de atributos e algoritmos de seleção de exemplos para construir novos algoritmos de seleção de dados?".

Nesta dissertação, trabalha-se fortemente com a hipótese que a conjunção de abordagens de seleção de atributos e abordagens de seleção de exemplos pode gerar novos algoritmos de seleção de dados tão, ou mais, competitivos que abordagens projetadas diretamente para seleção de dados. A adaptação de tais abordagens tem a grande vantagem de, caso a hipótese esteja certa, possuírem componentes já estudados e testados, portanto existe maior garantia quanto à qualidade final do novo algoritmo.

Resultados empíricos em Kuncheva e Jain (apud HO; LIU; LIU, 2002) mostram que a abordagem apresentada para seleção de dados é uma solução competitiva quando comparada à resolução sequencial de seleção de atributos e exemplos. Souza, Carmo e Lima (2008) reforçam esta ideia mostrando que nos testes realizados naquele trabalho, a solução de seleção de dados converge consistentemente mais velozmente que as outras soluções sequenciais. "Será que este padrão se repete, utilizando-se outras estratégias que não metaheurísticas? Ao se adaptar métodos de seleção de atributos, qual é a melhora (ou piora) causada pelo processo de busca em exemplos e vice-versa?".

A partir destes questionamentos, outro foco é a procura de um planejamento ideal (em termos de eficácia e eficiência) para a resolução do problema de seleção de dados, seja sequencialmente, executando seleção de atributos e após seleção de exemplos, ou vice-versa, ou ainda de forma conjunta.Uma última hipótese é que a utilização integrada de dois algoritmos, um de seleção de atributos e outro de seleção de exemplos, que claramente podem ser modificados para resolver conjuntamente o problema de seleção de dados são é uma abordagem, no mínimo, tão eficiente e eficaz quanto a execução sequencial de tais algoritmos.

1.4 Estrutura da dissertação

O restante desta dissertação é composto pelo capítulo 2, que traz um estudo sobre a fundamentação teórica necessária para um bom entendimento deste trabalho, definindo tarefas e algoritmos de aprendizado supervisionado bem como o problema de seleção de dados. O capítulo 3 faz uma revisão da literatura referente a este problema, mostrando a classificação das diversas abordagens que tratam tal problema, assim como os subproblemas de seleção de atributos e seleção de exemplos. O capítulo 4 descreve a nova abordagem proposta neste trabalho, apresentando quais características a diferenciam das demais encontradas na literatura. O capítulo 5 apresenta a configuração dos testes executados para validação da nova proposta, bem

como uma apresentação e discussão dos resultados gerados por tais testes. Por fim, o capítulo 6 conclui o trabalho, resumindo aquilo que foi apresentado e discutido na dissertação. Finalmente, o apêndice apresenta os resultados dos testes executados - de uma forma expandida.

2 FUNDAMENTAÇÃO TEÓRICA

Como dito anteriormente, o foco de discussão e estudo desta dissertação será direcionado às atividades realizadas nas fases de Preparação de Dados e Modelagem. Neste capítulo, será feita uma introdução aos conceitos, atividades e problemas presentes nestas duas fases. Neste trabalhos, utilizaremos a seguinte notação: uma base de dados $D = (A, I)$ é composta por um conjunto de exemplos $I = i_1, i_2, ..., i_m$ que são representados por um conjunto $A = a_1, a_2, ..., a_n$ de atributos, sendo que $a_n = y$, ou seja, a_n é o atributo-classe nessa base de dados. A explicação aqui apresentada começa pela fase de Modelagem e segue com Preparação de Dados.

2.1 Modelagem

Nesta fase, é feita a seleção dos algoritmos que serão utilizados na tentativa de extração de informação a partir dos dados. Tal seleção deve considerar todas as características destes algoritmos, tais como tipo de dados para os quais eles se adaptam de uma melhor forma, distribuição probabilistica dos dados, tipo de informação que deseja-se extrair, entre outros.

Através de mineração de dados, uma das formas de se extrair informação a partir de dados ocorre por meio da técnica de aprendizado supervisionado. Esta técnica é aplicável quando um dos atributos presentes na base de dados é o atributo especial y, chamado de *classe*, e caracteriza-se como sendo a saída (ou resultado) de um exemplo contendo todas as outros atributos apresentados. Desta forma, o problema em aprendizado supervisionado é deduzir uma função que, dados os valores dos atributos de um exemplo como entrada, prediga corretamente o valor do atributo *classe*, inicialmente desconhecido. Caso o valor desta classe seja uma variável contínua, temos então um problema de *regressão*, caso contrário, o problema é chamado de *classificação*.

Algoritmos de indução são aqueles utilizados quando deseja-se construir modelos de aprendizado supervisionado a partir de bases de dados. Existem diversos de algoritmos na literatura, porém, neste trabalho, utilizaremos três dos algoritmos de classificação mais conhecidos: o algoritmo de árvore de decisão C4.5, o algoritmo de aprendizado Bayesiano Naive Bayes e o algoritmo de aprendizado por exemplos k-Nearest Neighbours.

Árvores de decisão do tipo ID3 e sua sucessora C4.5 (QUINLAN, 1993 apud MITCHELL, 1997) são algoritmos de indução gulosos e recursisvos que constróem seus modelos, ou seja, as árvores de decisão, utilizando a valoração de um atributo como teste lógico para subdividir a

árvore em ramos, um para cada valor possível do dado atributo e assim continuando o processo recursivamente em cada ramo gerado. Este atributo é escolhido através da resposta à pergunta "Qual é o atributo que melhor classifica esta base de dados?", resposta esta que utiliza o cálculo estatístico de *ganho de informação* para informar qual atributo deve ser utilizado.

O algoritmo de indução Naive Bayes (JOHN; LANGLEY, 1995) é um classificador probabilistico simples que faz uso do teorema de Bayes e de suposições fortes de independência de atributos, ou seja, que não há correlação entre os atributos e estes contribuem independentemente para a classificação de um dado exemplo. Desta forma, o modelo gerado por este algoritmo classifica um dado exemplo calculando a probabilidade deste exemplo pertencer a todas as possíveis classes e, ao final, escolhe aquela classe com a maior probabilidade de rotular o dado exemplo.

O algoritmo kNN (k-Nearest Neighbours) (AHA; KIBLER, 1991) é um método de classificação baseado em exemplos pertencente à classe de algoritmos baseados em instâncias. Este algoritmo classifica um dado exemplo de acordo com a classificação dos seus *k* vizinhos mais próximos, definindo *proximidade* através do cálculo da distância entre dois exemplos, como por exemplo a distância Euclidiana ou a distância de Hamming.

Além de escolher o algoritmo de induçao utilizado, deve-se escolher um mecanismo de avaliação da qualidade do modelo gerado. Em geral, é comum utilizar estimativas de taxa de erro como medidas de qualidade para modelos de aprendizado supervisionado. O método *hold out* divide a base de dados em dois subconjuntos, um de treinamento e outro de teste. Assim, a estimativa da taxa de erro de um modelo é calculada utilizando-se a base de treinamento para construir o modelo de aprendizado e a base de teste para calcular esta taxa. Já o método *k-fold cross validation* faz este cálculo de forma diferente. A base de dados inicial é dividida aleatoriamente em *k* subconjuntos distintos, utilizando um dos *k* subconjuntos como conjunto de teste e os outros *k - 1* conjuntos como base de treinamento. Este procedimento é repetido *k* vezes, e cada subconjunto é utilizado uma vez como conjunto de teste. Desta forma, a taxa de erro é calculada como sendo a taxa de erro média destes *k* testes.

2.2 Preparação de dados

A fase de Preparação de Dados visa produzir uma base de dados que será utilizada durante a fase de Modelagem, a de Avaliação e de quaisquer outras tarefas executadas posteriormente. Atividades como seleção, limpeza, construção, integração e formatação de dados são exemplos de tarefas que podem ser executadas durante esta fase. A seguir será discutida a tarefa de seleção de dados, visto que esta tarefa é o foco do presente trabalho.

2.3 Seleção de Dados

Neste contexto, a seleção de dados consiste na escolha dos dados a serem analisados, levando-se em consideração a relevância para o objetivo da mineração de dados, qualidade

destes dados e restrições técnicas e orçamentárias. Esta escolha produz diversos efeitos importantes e interessantes para o processo de mineração de dados.

Modelos gerados a partir de bases de dados que foram submetidas à esta seleção são menos complexos que modelos gerados a partir de bases que não passaram por este processo e, portanto, mais fáceis de serem compreendidos por aqueles interessados na extração de informação. Além disso também são possivelmente mais próximos da explicação real do fenômeno existente nos dados (via Occam's razor[1]) e menos propensos a *overfitting*, que é a condição na qual, ao invés de ocorrer uma generalização a partir dos exemplos apresentados, este modelo memoriza tais exemplos, perdendo assim a função a qual foi proposto.

Além destes, o mais desejado e principal efeito deste processo de seleção é a construção de modelos que, quando testados, mostram-se mais eficazes e mais eficientes que os modelos gerados sem passar por este processo. Desta forma, a comparação destes modelos é um bom comparativo *baseline* para verificar a qualidade de uma estratégia de seleção de dados.

A literatura atual descreve e estuda Seleção de Dados dividindo-a em duas tarefas parecidas e interligadas: Seleção de Atributos, que se dá nas colunas da base de dados e Seleção de Exemplos, que se dá nas linhas desta base. A seguir a definição destas duas atividades será apresentada, bem como a descrição de alguns dos problemas encontrados para realizá-las.

2.3.1 Seleção de Atributos

Atributos são descritores de dados, caracterizando-os quanto ao tipo (textual, nominal, categórico, numérico), quanto à cardinalidade do seu domínio (finito ou infinito), etc. O capítulo 5 de Theodoridis e Koutroumbas (2006) faz uma boa introdução ao problema de seleção de atributos e é nela que basearemos nossa descrição.

Adequar-se a restrições computacionais é uma razão clara para se fazer seleção de atributos. As bases de dados atuais possuem conjuntos de atributos cuja cardinalidade varia desde poucas dezenas até dezenas de milhares (bases de dados de expressão de genes, por exemplo). Utilizar todos os atributos para a geração do modelo computacional pode se tornar uma tarefa demasiadamente complexa, custosa e desnecessária.

Em uma base de dados utilizada em um sistema de Aprendizado Supervisionado, por exemplo, um atributo pode isoladamente ser um classificador, ou seja, ser altamente correlacionada com a classe a ser descrita. Porém, ao se combinar este atributo com outro correlacionado a ele, o poder de classificação desde conjunto não aumenta em relação ao poder individual do atributo inicial mas a complexidade do modelo sim.

Além disto, quanto mais atributos houver em uma base de dados, maior será a necessidade de parametrização do modelo gerado (maior número de nós em uma árvore de decisão ou maior o número de dimensões no hiperplano gerado por uma SVM, por exemplo). Estas são apenas algumas das razões que demonstram a importância de se selecionar atributos. Assim, para produzir uma boa seleção, deve-se levar em consideração as razões que fazem um atributo

[1] http://en.wikipedia.org/wiki/Occam's_razor - Sítio acessado em Abril/2010.

importante para uma determinada tarefa de mineração. Para tal, escolheremos o problema de classificação (área de aprendizado supervisionado) como exemplo.

Variadas são as explicações de importância de atributos e definições do problema de seleção de atributos encontradas na literatura. Theodoridis e Koutroumbas (2006) define este problema fazendo a pergunta "Dada uma quantidade de atributos, como pode-se selecionar os mais importantes deles para reduzir o seu número e ao mesmo tempo manter o máximo possível de sua informação de discriminação de classe?"[2] e define que deve-se buscar a seleção de atributos que gerem uma distância inter-classes grande bem como uma variância intra-classe pequena.

John, Kohavi e Pfleger (1994) descrevem este problema através de definições de relevância de atributos em tarefas de aprendizado supervisionado. Primeiramente, ele reproduz definições existentes na literatura, mostra que existem cenários nos quais estas definições falham e propõe dois graus de relevância para atributos. Atributos altamente relevantes são aqueles que, quando adicionados ou removidos do conjunto que é utilizado para definir a classe, modificam a probabilidade desta, ou seja, a probabilidade condicional da classe dado o atributo é diferente da probabilidade da classe, demonstrando assim haver correlação entre o atributo classe e o atributo relevante. Já atributos fracamente relevantes são aqueles que não são altamente relevantes e que contribuem ocasionalmente com a precisão de um classificador.

Kohavi e John (1997) define o problema de seleção de atributos como sendo o problema de se encontrar um subconjunto de atributos de modo que um indutor construído a partir de dados que contenham apenas este subconjunto produza um classificador cuja precisão é a máxima possível. Além disto, este trabalho mostra alguns problemas relacionados com esta busca, tais como o problema de *bias-variance tradeoff*[3] e demonstra que há cenários nos quais o conjunto ótimo de atributos, ou seja, aquele que maximiza a precisão de um classificador, não necessariamente é aquele que contém apenas atributos considerados relevantes.

2.3.2 Seleção de Exemplos

Exemplos são instâncias dos fenômenos representados em uma base de dados, ou seja, são objetos com os quais são construídos modelos de aprendizado, como por exemplo os utilizados para predição. Usualmente, estes elementos são expressos através de vetores de valores de atributo singulares, ou seja, sem relação direta explícita ou implícita com outros registros na base de dados, mas outras representações mais complexas podem ser utilizadas, tais como representações que expressem relações entre instâncias ou representações nas quais mais de uma instância representa um objeto.

Blum e Langley (1997) listam três razões básicas para se selecionar exemplos em tarefas de aprendizado de máquina. A primeira é que, se é utilizado um algoritmo de aprendizado cuja computação é custosa e se é provida uma quantidade mais que suficiente de exemplos

[2] Tradução livre do trecho "Given a number of features, how can one select the most important of them so as to reduce their number and at the same time retain as much as possible of their class discriminatory information?"

[3] Fonte: http://videolectures.net/stanfordcs229f08_ng_lec09/ - Sítio acessado em Abril/2010.

de treinamento, então é interessante que se foque o aprendizado em apenas alguns exemplos, a fim de tornar o processo de aprendizagem menos dispendioso. Além disto, se o custo de se classificar um exemplo é alto mas diversos exemplos não classificados são facilmente conseguidos, então selecionar exemplos para posterior classificação e utilização em um sistema de aprendizado se torna uma tarefa extremamente relevante. Por fim, deve-se selecionar exemplos buscando aumentar a taxa de aprendizagem, focando a atenção em exemplos informativos que sejam importantes para a pesquisa no espaço de hipóteses.

Liu e Motoda (2002) listam três funções fundamentais da seleção de exemplos. Segundo o trabalho, é natural que qualquer algoritmo, seja de mineração de dados ou de outra área, possua limitações quanto ao volume, tipo e formato de dados que consegue tratar. Assim, a primeira função que deve estar presente em um processo de seleção de exemplos é a capacidade de habilitar (*enable*) a utilização destes algoritmos em bases de dados bastante grandes. Outra função é a de focar (*focusing*), visto que quando dados são coletados a partir de observações em um domínio, esta coleta captura dados que não necessariamente são importantes para uma determinada tarefa de mineração de dados, portanto, focar a pesquisa nos dados relevantes torna-se algo desejável. Por fim, a função de limpeza (*cleaning*) também é fundamental, pois não importa quão bom ou complexo seja um método de aprendizado, se forem apresentados dados de má qualidade a este método, muito provavelmente o resultado gerado por ele também será de má qualidade (princípio "Garbage In Garbage Out"[4]).

Ainda em Liu e Motoda (2002) é definida a tarefa de seleção de exemplos como a tarefa de escolher um subconjunto dos dados que alcance o objetivo original da tarefa de mineração de dados, da mesma forma que se fosse utilizada toda a base de dados disponível. Além disto, o trabalho define que o resultado ideal de um processo de seleção de exemplos é uma base de dados que seja independente do modelo de aprendizado utilizado, a menor possível e que possa ser utilizada nas mesmas tarefas que a base inicial com o mínimo de deterioração no desempenho destas tarefas.

Apesar de ser claramente um problema análogo ao de seleção de atributos, seleção de exemplos possui algumas características adicionais que devem ser levadas em consideração para que haja qualidade no processo de seleção. Um exemplo simples é que, enquanto um único atributo pode ser suficiente para um descritor de um classe em um problema de aprendizagem de conceito, um único exemplo terá pouca, ou mesmo nenhuma, relevância para a mesma descrição. Outro problema é que a distribuição de classes é uma variável que influencia sensivelmente o processo de aprendizagem e, quando modificada de forma imprudente por um algoritmo de seleção, pode levar à construção de modelos de baixa qualidade.

[4] http://en.wikipedia.org/wiki/Garbage_In,_Garbage_Out – Sítio acessado em Abril/2010.

3 TRABALHOS RELACIONADOS

O capítulo 2 discutiu o problema de seleção de dados e sua divisão em seleção de atributos e seleção de exemplos. Neste capítulo serão listadas algumas abordagens encontradas na literatura que tratam estes problemas. Inicialmente será discutido o problema de seleção de atributos, dando enfoque a algoritmos relevantes quanto ao tipo de abordagem escolhida. Algoritmos para seleção de exemplos podem ser classificados em dois ramos: o de seleção de exemplos classificado (onde o objetivo é ”otimizar”a base de dados) e o de seleção de exemplos não classificados (onde objetiva-se criar uma base de dados já ”ótima”). Para seleção de exemplos, o foco deste trabalho será a discussão de abordagens que tratam de seleção de exemplos classificados. Por fim serão discutidas abordagens que trabalham com seleção de dados de forma integral, tratando o problema de seleção de atributos e o de exemplos de forma simultânea.

A literatura divide os algoritmos de seleção de atributos e seleção de exemplos em quatro classificações distintas:

Filtros Filtros são aqueles algoritmos que realizam a tarefa de seleção em um momento de pré-processamento separado e de forma totalmente independente do algoritmo de indução utilizado no sistema de aprendizagem. Desta forma, estes algoritmos utilizam métricas próprias para o processo de seleção, tais como correlação ou ganho de informação (atributos) e conceitos de vizinhança (exemplos). Tais algoritmos caracterizam-se por serem computacionalmente baratos e gerarem conjuntos com menor qualidade que os gerados por outras classes de abordagens.

Wrappers Wrappers são técnicas que utilizam um algoritmo de indução como parte da tarefa de seleção, de forma que a qualidade de um dado atributo/exemplo ou de um subconjunto deles é mensurada através do algoritmo de indução utilizado. Esta utilização se dá executando o algoritmo de indução no conjunto de treinamento e utilizando a precisão estimada como medida de qualidade do subconjunto testado. Tal classe de algoritmos é computacionalmente mais dispendioso que os filtros mas geram conjuntos com maior qualidade. Como visto a seguir, o maior diferencial entre algoritmos wrappers é a forma como tais algoritmos efetuam a busca pelo espaço de soluções.

Híbridos Abordagens híbridas são algoritmos que realizam a seleção em duas fases: uma inicial na qual são utilizados filtros para pré-selecionar dados e assim diminuir o espaço de busca utilizado na segunda fase, na qual a busca é efetuada com a utilização de wrappers. Desta forma, estas abordagens tentam combinar a qualidade de seleção gerada por

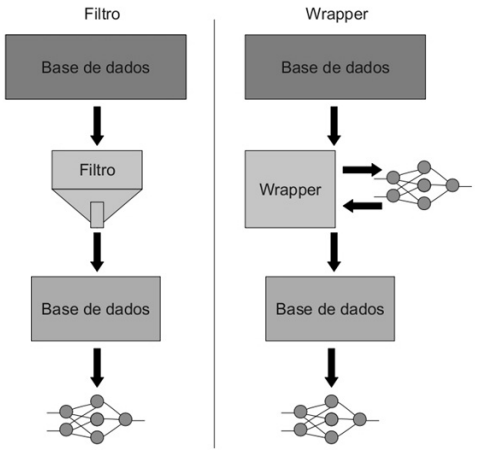

Figura 3.1: Filtros e Wrappers para seleção de dados.

wrappers com o baixo custo computacional dos filtros.

Incorporados Um refinamento possível é efetuado por algoritmos incorporados de seleção. Estes algoritmos agregam o conhecimento da estrutura de funcionamento do algoritmo de aprendizado utilizado e a função de classificação utilizada por ele e assim combinam o passo de seleção com a geração do modelo de indução. Desta forma, tais algoritmos perdem a generalidade, podendo ser utilizados com apenas um algoritmo específico de aprendizado, porém ganham em performance e funcionam de forma transparente, conjuntamente com o algoritmo de indução.

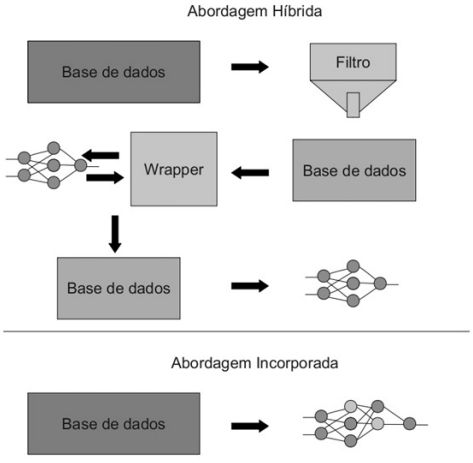

Figura 3.2: Algoritmos Híbridos e Incorporados para seleção de dados.

3.1 Abordagens para Seleção de Atributos

A literatura já descreveu variados algoritmos para seleção de atributos que foram listados e estudados em diversos trabalhos (BLUM; LANGLEY, 1997; KOHAVI; JOHN, 1997;

GUYON; ELISSEEFF, 2003; SOUZA, 2004). A seguir serão listados alguns algoritmos, seguindo a classificação descrita na seção anterior.

3.1.1 Filtros

Um dos filtros clássicos encontrados na literatura é o FOCUS (ALMUALLIM; DIETTERICH, 1991). Este algoritmo faz uma busca completa, começando a partir de atributos individuais, seguindo para conjuntos de dois atributos, depois três e assim por diante, até que seja encontrado o menor subconjunto de atributos que gere uma base dados sem inconsistência, ou seja, na qual não ocorram exemplos com os mesmos valores de atributos mas que são classificadas em classes diferentes.

O filtro LVF (LIU; SETIONO, 1996b) consiste em um algoritmo que, a cada iteração, gera subconjuntos de atributos de forma aleatória e utiliza a cardinalidade deste conjunto bem como um valor de inconsistência que este subconjunto de atributos produz na base de dados para aceitar ou não tal subconjunto aleatório como nova solução. Uma nova solução só é aceita caso a cardinalidade do conjunto de atributos selecionados seja menor ou igual à atual e o valor de inconsistência calculado seja menor que um limiar pré-definido. Além disto, este algoritmo (1) registra soluções durante o processo de busca, podendo produzir diversas soluções em uma execução.

Algoritmo 1: Algoritmo LVF

```
Data: base de dados D, inteiro num_iter e real limiar
Result: Subconjunto de atributos
S_best = Todos os Atributos de D; Inicialize pesos W_i = 0;
for i = 1 to num_iter do
    S_i = Escolha aleatoriamente um exemplo de D;
    if Cardinalidade(S_i) ≤ Cardinalidade(S_best) e Inconsistência(S_i, D) < limiar
    then
        Registre S_i;
        if Cardinalidade(S_i) < Cardinalidade(S_best) then
            S_best = S_i;
        end
    end
end
return S_best;
```

Kira e Rendell (1992) apresentam o algoritmo Relief, outro filtro bastante conhecido. Este algoritmo constrói uma técnica para mensurar numericamente a importância de cada atributo, onde esta importância recebe valor zero e é atualizada a cada iteração do algoritmo. Em cada iteração, um exemplo é escolhido aleatoriamente e a partir dele são selecionados os exemplos mais próximo da mesma classe e os exemplos mais próximos da outra classe (supondo um problema de aprendizagem de conceito) para que a partir deles, o algoritmo possa atualizar a importância de cada atributo de acordo com sua capacidade de discriminação entre o exemplo-base e o de classe contrária e sua capacidade de reconhecer o exemplo de mesma classe. Ao

final da execução do algoritmo, pode-se apenas utilizar esta medida numérica como parâmetro para o indutor, dando "pesos" para cada atributo ou selecionar aqueles como maior valor.

Em Frank (2002), um filtro específico para o algoritmo Naive Bayes é construído calculando-se fórmulas que definem o limite de erro de um modelo construído com um subconjunto de atributos e utilizando o algoritmo *Branch and Bound* para fazer uma pesquisa completa pelo espaço de subconjuntos de atributos.

Filtros podem ser apresentados como uma primeira tentativa de abordagem para este problema de seleção de atributos. A premissa de que as características dos atributos por si só possui relevância em uma tarefa de classificação, por exemplo, é correta e inteligente, porém, como visto anteriormente o conjunto ótimo para uma dada tarefa não é necessariamente aquele onde há apenas atributos relevantes. Além disto, algoritmos de aprendizagem possuem natureza e parametrização diversas, fazendo com que atributos que são interessantes para se utilizar em um algoritmo deixam de ser interessantes quando utilizados em outros. Assim, abordagens que utilizam tais algoritmos como parte do processo de seleção tornam-se abordagens também interessantes.

3.1.2 Wrappers

Forward Selection e Backward Selection são dois algoritmos genéricos e análogos que podem ser classificados como wrappers. Tais algoritmos partem de um subconjunto inicial de atributos, vazio no caso de forward selection e completo no caso de backward selection, e efetuam o processo de busca através da inclusão/remoção de atributos, seguido do cálculo da qualidade do novo subconjunto e decisão de prosseguimento da busca.

LVW (LIU; SETIONO, 1996a) é um wrapper que processa a busca de forma aleatória, afim de evitar ótimos locais e a execução de buscas bastante custosas. Este algoritmo repete o processo de geração de uma nova solução aleatória (subconjunto de atributos) e avaliação de qualidade até que um número predefinido de iterações ocorra sem que seja encontrada alguma solução com maior qualidade que a melhor encontrada até o momento.

Metaheurísticas como Algoritmos Genéticos (GOLDBERG, 1989) e Têmpera Simulada (KIRKPATRICK et al., 1983) podem ser facilmente utilizadas como wrappers para seleção de atributos. Codificar uma solução para este problema como sendo um vetor de valores binários (está/não está incluso) é uma abordagem natural e, além disto, procedimentos diversos como geração de soluções vizinhas, mutação de soluções e combinação também são facilmente implementadas quanse se utiliza este tipo de representação. Assim, diversos são os trabalhos que utilizam tais ferramentas para lidar com a tarefa de seleção de atributos (VAFAIE; JONG, 1992; YANG; HONAVAR, 1998; EMMANOUILIDIS; HUNTER; MACINTYRE, 2000; OLIVEIRA et al., 2002; SUN et al., 2002; BERITELLI et al., 2005; HUANG; CAI; XU, 2007; PEDRYCZ; PARK; PIZZI, 2009; WANG; HUANG, 2009).

Wrappers são uma tentativa de otimizar o processo de busca por um subconjunto ótimo de atributos, utilizando a estimativa de performance de um algoritmo de indução como guia para este processo. Desta forma, há um maior acoplamento entre a seleção de atributos e o processo

de aprendizagem, porém, este acoplamento ainda é um tanto fraco, visto que a única informação utilizada para conectar as duas partes é a estimativa de performance.

3.1.3 Híbridos

Um refinamento da idéia central dos wrappers é apresentada nos algoritmos híbridos, onde características internas dos algoritmos são utilizadas para tornar o acoplamento entre o processo de seleção de atributos e a aprendizagem ainda maior. Xing, Jordan e Karp (2001) apresentam uma abordagem híbrida composta por três fases, testando-a em uma base de dados de expressão de genes composta por 72 exemplos e 7130 atributos. Esta abordagem inicia com um filtro estatístico que avalia a viabilidade da execução de seleção de atributos por filtros e discretiza os atributos para a segunda fase, que é responsável por ordenar os atributos de acordo com uma medida de ganho de informação. A terceira fase utiliza uma estratégia de *Markov blanket filtering* (SAEED, 2008) em subconjuntos de atributos que são passados para um indutor, avaliados e no fim o melhor é retornado.

Souza (2004) propõe o FortalFS, algoritmo híbrido que consta de uma fase na qual filtros são executados e o subconjunto de exemplos gerados por cada execução é levada em consideração na criação de um vetor que serve como base para a busca aleatória da segunda fase. Este vetor é um contador do número de vezes que cada atributo é selecionado numa execução do filtro. A segunda fase consta de uma busca aleatória utilizando a probabilidade gerada pela distribuição mostrada no vetor criado na fase anterior, sendo esta busca predisposta a selecionar subconjuntos que construam modelos que sejam melhor avaliados e subconjuntos com menor cardinalidade. O Algoritmo 2 descreve o pseudocódigo do FortalFS.

Algoritmo 2: Algoritmo FortalFS

```
Data: base de dados D e inteiro num_iter
Result: Subconjunto de atributos
O = Execute filtro (D);
Adão = Calcule Adão (O);
for i = 1 to num_iter do
    S = Gere Subconjunto (Adão);
    if Taxa de erro(S, D) < Taxa de erro (S_best , D) then
        S_best = S;
    else
        if Taxa de erro(S, D) = Taxa de erro (S_best, D) && |S| < |S_best| then
            S_best = S;
        end
    end
end
return S_best;
```

3.1.4 Incorporados

Em Guyon et al. (2006) são encontradas formas com as quais se pode incorporar um algoritmo de seleção de atributos a um algoritmo de aprendizado. Em linhas gerais, a primeira caracteriza-se pela capacidade de, durante o processo de geração do modelo, incluir ou excluir um atributo levando-se em consideração cálculos de como a função de classificação será atualizada com a inserção ou remoção de um dado atributo, antes que esta operação seja efetivamente executada. A segunda forma caracteriza-se pela transformação (ou relaxamento) do problema de seleção em um problema de ponderação (*weighting*) de atributos e resolução do problema de minimização da taxa de erro do modelo gerado e a terceira caracteriza-se por, a partir de suposições quanto à convexidade da função do modelo gerado e de uma função sobre os atributos, reduzir o problema de seleção a uma nova versão de minimização de um parâmetro sobre tais funções[1].

Árvores de decisão, tais como ID3 (QUINLAN, 1986), implementam métodos de seleção de atributos de forma natural. O processo de geração do modelo neste algoritmo de indução é realizado pela criação de nós que são testes lógicos sobre valores de um dado atributo. Assim, a seleção de atributos se dá a partir do cálculo do ganho de informação dado pela escolha de um atributo, ou seja, será selecionado aquele atributo que retornar o maior ganho de informação ao modelo (ou, em outras palavras, minimizará a entropia do mesmo). Desta forma, seleciona-se atributos e constrói-se modelos simultaneamente.

3.2 Abordagens para Seleção de Exemplos

Assim como em seleção de atributos, há alguns trabalhos que listam e descrevem algoritmos e técnicas de seleção de exemplos (BLUM; LANGLEY, 1997; WILSON; MARTINEZ, 2000; BRIGHTON; MELLISH, 2002; LIU; MOTODA, 2002; REINARTZ, 2002). Tratando-se de seleção de exemplos classificados (o foco deste trabalho), tais algoritmos também podem ser classificados nas quatro categorias apresentadas para seleção de atributos. Apesar de poderem ser classificados nas quatro categorias, aqui não será apresentada nenhuma abordagem classificada como híbrida, visto que não foi encontrado na literatura nenhum trabalho que pudesse receber tal classificação, o que é justificado através da percepção que, em seleção de exemplos para mineração de dados, filtros são abordagens que ou são puramente estatísticas ou, mesmo indiretamente, têm alguma ligação com o algoritmo de aprendizado utilizado no processo de mineração.

Algoritmos de seleção de exemplos podem ser divididos em duas grandes categorias: os métodos de seleção de protótipos e as abordagens para seleção do conjunto de treinamento. Métodos de seleção de protótipos são algoritmos especializados para funcionar de forma ótima com a a classe de algoritmos de vizinhos mais próximos, enquanto abordagens para seleção de conjunto de treinamento são tecnicas que selecionam exemplos para a geração de modelos

[1] Esta explicação tenta transferir para uma discussão textual o que é apresentado através de funções matemáticas em Guyon et al. (2006). Para um entendimento e explicação mais detalhadas, ler o documento original.

quaisquer, não só aqueles baseados em vizinhos mais próximos.

3.2.1 Filtros

Técnicas de amostragem, trazidas do campo da estatística, são exemplos clássicos de abordagens que podem ser classificadas como filtros. Retirar uma amostra de uma população (no contexto deste trabalho a população é a base de dados) consiste basicamente em selecionar um subconjunto desta população que seja escolhido aleatoriamente ou represente uma cópia imparcial da população, ou seja, possua as mesmas características da população inicial.

Uma das técnicas mais simples de amostragem é a de amostragem aleatória simples. Nesta técnica cada indivíduo da população (exemplo numa base de dados) tem igual probabilidade de ser escolhido como parte da amostra selecionada. Esta seleção pode ocorrer com ou sem repetição, ou seja, pode-se "clonar" exemplos na nova base de dados, possibilidade interessante quando há desbalanceamento de classes na base de dados inicial (JAPKOWICZ, 2000).

A amostragem aleatória estratificada ocorre em duas fases. Na primeira, a população inicial é separada em estratos, que são subconjuntos distintos desta população. Tais conjuntos são escolhidos de forma que os componentes de cada estrato tenham características semelhantes. Já na segunda fase, em cada estrato é selecionado um conjunto de elementos que farão parte da amostragem. Esta seleção ocorre de forma independente em cada estrato. Desta forma, busca-se que gerar uma amostra imparcial da população inicial.

O capítulo 11 de Bishop (2007) fornece uma explicação interessante sobre o tema de amostragem aleatória, tratando-o matematicamente e utilizando conceitos de aprendizagem para justificar e mostram métodos de amostragem. No contexto de Mineração de Dados, filtros genéricos utilizados para seleção de exemplos são mais raros e menos utilizados que filtros para seleção de atributos, visto que a importância de um dado exemplo é dada pelo modo como o algoritmo de indução constrói o seu modelo de aprendizado, utilizando protótipos que são construídos artificialmente a partir de uma base ou exemplos reais da mesma, ou modelos que transformam atributos em regras e assim por diante. Assim sendo, a escolha de exemplos torna-se bem mais dependente do algoritmo utilizado para indução e desta forma, filtros genéricos tornam-se estratégias não muito interessantes.

O filtro apresentado em Brodley e Friedl (1999), aqui chamado MVF (*Majority Voting Filter*), é um algoritmo que utiliza um conjunto de indutores como filtro para remover exemplos que possivelmente são *ruídos*. Para tal, ele treina este conjunto de indutores utilizando toda a base de dados apresentada e remove todos aqueles exemplos que não são classificados corretamente pela maioria destes indutores. O Algoritmo 3 mostra o pseudocódigo desta abordagem.

O filtro DemoIS, apresentado em García-Osorio, Haro-García e García-Pedrajas (2010) - uma evolução do trabalho apresentado em Haro-García e García-Pedrajas (2009) - é um trabalho interessante que reforça esta ideia. Este trabalho desenvolve uma estratégia similar com a de *emsemble learning*, na qual diversos indutores simples (no caso 1-NN) são utilizados em combinação para, em conjunto funcionarem como um único indutor. Desta forma, o algoritmo

Algoritmo 3: Algoritmo MVF

Data: base de dados D, conjunto de indutores A
Result: Subconjunto de exemplos
for $i = 1$ **to** $|A|$ **do**
  Treine A_i utilizando D;
end
$S = D$;
for $i = 1$ **to** D **do**
  if I_i *não é classificado corretamente pela maioria dos indutores de A* **then**
    Remova I_i de S;
  end
end
return S;

desenvolvido utiliza-se de seguidas execuções de algoritmos de seleção de exemplos em subconjuntos da base de dados inicial que votam quais exemplos devem ser selecionados, e assim, seleciona-se os exemplos na base de dados inicial. Apesar de ser classificado como um filtro, pois não há utilização direta da performance de um algoritmo de aprendizado para guiar o processo de busca, um dos parâmetros do algoritmo é justamente um algoritmo de seleção de exemplos que pode estar ligado a indutor e portanto, fazer com que, mesmo indiretamente, a proposta para seleção de exemplos utilize-se da performance deste indutor para selecionar exemplos. O Algoritmo 4 descreve o pseudocódigo do DemoIS.

Algoritmo 4: Algoritmo DemoIS

Data: base de dados D, inteiro s e inteiro num_iter
Result: Subconjunto de exemplos
for $i = 1$ **to** *num_iter* **do**
  Divida a base de dados em n / s subconjuntos distintos;
  for $j = 1$ **to** n / s **do**
    Execute algoritmo de seleçao de exemplos (D_j);
    Conte quais instâncias foram removidas (D_j);
  end
end
Obtenha limiar de votos v;
return Todas as instâncias de D que foram removidas menos de v vezes;

3.2.2 Wrappers

Kuncheva (1995) apresenta a utilização de um algoritmo genético para criação do conjunto de referência (instâncias que são utilizadas para classificar uma outra instância qualquer) de um algoritmo da classe k-NN. Apesar de tal trabalho validar a abordagem especificando o algoritmo de aprendizagem utilizado, assim como em seleção de atributos, quaisquer metaheurísticas aplicadas para esta tarefa podem ser classificadas como wrappers.

Canoa, Herrera e Lozano (2003) faz um estudo sobre a utilização de quatro diferentes

instâncias de algoritmos genéticos para a tarefa de seleção de exemplos focando-se na seleção de protótipos (para utilização do indutor 1-nn) e na redução do tamanho de bases de dados de treinamento (para utilização de um indutor qualquer). Os resultados comprovam que tais abordagens são mecanismos interessantes para se lidar com esta tarefa.

Continuando com a utilização de metaheurísticas para seleção de exemplos, Kim (2006) também utiliza um algoritmo genético para selecionar exemplos em uma base de dados para predição financeira e utiliza como método de indução a técnica de rede neural artificial. García-Pedrajas, Castillo e Ortiz-Boyer (2009) apresenta a utilização de um algoritmo genético cooperativo coevolucionário como wrapper para seleção de protótipos, mas que, como expresso pelos autores, pode ser transformado em ferramenta para redução do tamanho de bases de dados.

Novamente, este estudo da literatura reforça a afirmação anterior de que para seleção de exemplos é bastante interessante que haja um acoplamento entre a técnica de seleção e o algoritmo de indução utilizados. Mesmo em trabalhos que claramente utilizam wrappers para tal seleção, a construção destes algoritmos e teste dos mesmos se dá tendo por base um algoritmo de indução fixo. Tal fato pode ser coincidência mas é uma demonstração empírica do fato citado anteriormente.

3.2.3 Incorporados

Assim como em seleção de atributos, há algoritmos de aprendizado de máquina que já tratam naturalmente o problema de seleção de exemplos de forma incorporada. Como mostrado em Blum e Langley (1997), há métodos, tais como *perceptron* (ROSENBLATT, 1958) e *edited nearest-neighbor* que somente aprendem, ou seja, modificam seu modelo de conhecimento quando um exemplo apresentado é classificado erroneamente por este modelo. Estes métodos incorporam a tarefa de seleção de exemplos ignorando todos os exemplos que são corretamente classificados pela hipótese atual expressa no modelo criado pelo algoritmo de aprendizagem.

A construção de um modelo de aprendizagem para o algoritmo 1-NN se dá através da armazenagem dos exemplos que serão utilizados como base para classificação das outras instâncias e é desta forma que Marchiori (2010) desenvolve sua abordagem incorporada ao modelo do algoritmo 1-NN. Esta abordagem é desenvolvida em um processo de duas fases, para o qual a primeira fase consiste em um método que busca remover da base de dados aqueles exemplos que sejam *outliers*, estejam isolados ou muito próximos da fronteira de decisão do 1-NN, tentando ampliar esta fronteira e assim, reduzir o erro empírico apresentado pelo algoritmo. Após isto, a segunda fase remove exemplos que não sejam importantes para a nova fronteira construída na primeira fase.

Lee e Mangasarian (2001) apresenta uma abordagem que modifica a construção da superfície de separação para SVMs. Esta modificação se dá através da utilização de kernels não-lineares e subjacente transformação do modelo de programação de SVMs para um modelo no qual o kernel, ao invés de calcular a similaridade entre todos os exemplos da base de dados, calcula a similaridade entre estes exemplos com apenas uma amostra bastante pequena retirada da base de dados. Após a superfície ser construída, os exemplos que não estão na amostra são

descartados e apenas os outros são utilizados na função de classificação de novos exemplos.

3.3 Abordagens para Seleção de Dados

Blum e Langley (1997) utiliza-se de outros trabalhos para demonstrar a clara ligação entre os problemas de seleção de atributos e seleção de exemplos. Littlestone e Mesterharm (1997) mostra uma variante para aprendizado online do algoritmo Naive Bayes que atualiza seu modelo quando dá uma classificação incorreta a um exemplo e comporta-se melhor com atributos irrelevantes que a versão padrão do algoritmo. Assim, é interessante que técnicas de seleção de dados tentam modelar e resolver estes problemas de forma relacionada, fazendo com que seleção de atributos potencialize as qualidades de seleção de exemplos e vice-versa.

Um modo natural de se abordar seleção de dados é utiliza-se de algoritmos de seleção de atributos e seleção de exemplos sequencialmente. Desta forma, primeiro reduz-se o tamanho da base de dados em uma das dimensões e depois, a partir da base de dados reduzida, reduze-se a outra dimensão da mesma.

Li, Pal e Shiu (2006) desenvolve uma abordagem sequencial para classificadores do tipo CBR. A primeira fase consiste do processo de seleção de atributos. Utilizando rough sets e suas definições matemáticas de dispensabilidade e indispensabilidade, os autores adicionam um parâmetro β a estas definições e assim desenvolvem dois algoritmos que selecionam os atributos que são mais importantes seguindo estas definições. Para a segunda fase, são desenvolvidos quatro algoritmos, um que baseia sua seleção em definições de cobertura e acessibilidade de exemplos, outro baseado em similaridade de exemplos e mais dois que são a aplicação do princípio de k-NN nestes dois algoritmos. Além de executar sequencialmente estes algoritmos, também é proposta uma abordagem interativa para seleção de atributos e seleção de exemplos, visto que todos os algoritmos criados neste trabalho são iterativos e retornam respostas parciais a cada iteração.

Liu, Motoda e Yu (2004) não apresentam uma proposta para seleção de dados, mas sim uma abordagem para seleção de atributos condicionada através de seleção de exemplos. O algoritmo criado no trabalho inicialmente separa os exemplos de uma base de dados em diversos subconjuntos cujos exemplos tem características similares. Esta separação se dá pela construção de kd-trees nas quais os atributos que subdividem cada nó são escolhidos através do cálculo da variância destes atributos. Por fim, uma modificação do Relief é executada, e a base de dados utilizada para calcular o "peso" de cada atributo é construída escolhendo-se aleatoriamente um exemplo de cada subconjunto.

Fragoudis, Meretakis e Likothanassis (2002) propõe um algoritmo específico para o problema de seleção de dados para classificação de textos, onde os textos são representados através de "Bag of Words" e a classificação dos textos é binária. Para exemplificação, utilizemos A e ˜A para representar uma classe e sua negação. Este algoritmo opera em duas fases, primeiramente selecionando todas as palavras que são as melhores preditoras para a classe A, ou seja, aquelas palavras que aparecem em uma maior proporção nos textos classificados como A do que nos textos classificados como ˜A. Ao selecionar estas palavras, ele seleciona todos

os textos nos quais estas palavras aparecem e descarta os outros. A partir dai a segunda fase do algoritmo seleciona dentre todas as palavras inicialmente listadas quais são as que melhor distinguem os textos classificados em ˜A, utilizando-se apenas do subconjunto dos textos selecionado na primeira fase. Por fim o algoritmo retorna todas estas palavras selecionadas nas duas fases, juntamente com os textos selecionados na primeira fase.

A utilização de metaheurísticas diversas, em especial de algoritmos genéticos, também é abundante em abordagens para seleção de dados. Isto se dá pela facilidade de se representar soluções para este problema, representando-as através de um vetor de valores binários de tamanho igual ao número de atributos mais o número de exemplos. Por exemplo, Skalak (1994) apresenta a utilização da técnica de otimização Subida de Encosta (Hill Climbing) para a seleção de dados, removendo atributos e selecionando protótipos, tendo como função objetivo a performance de um classificador do tipo 1-NN.

Ishibuchi e Nakashima (2000) apresenta uma abordagem baseada em um algoritmo genético elitista para seleção simultânea de atributos e de exemplos. O problema de seleção de dados é então formulado como sendo um problema multi-objetivo de minimização do número de atributos, minimização do número de exemplos selecionados maximização da performance do subconjunto gerado (ou seja, maximização da qualidade do modelo gerado por um indutor construído a partir deste subconjunto). Partindo desta formulação, o problema é transformado em uma versão mono-objetivo na qual a função objetivo é igual à soma dos valores de cada objetivo inicial, multiplicados pelo seu peso (importância) individual na tarefa. Apesar da solução representar conjuntamente os subconjuntos de atributos e de exemplos, cada região é tratada separadamente, havendo um operador de mutação especial para a região referente a exemplos. Este operador é propenso a gerar mutações que diminuam o número de exemplos selecionados.

Ho, Liu e Liu (2002) desenvolve uma abordagem também baseada em algoritmo genético elitista, porém constrói a função objetivo utilizando um único valor de peso para a minimização do número de atributos e do número de exemplos adicionado à maximização da precisão do classificador. Esta precisão é calculada utilizando-se um conjunto de avaliação, classificado através de 1-NN. A cada classificação correta, soma-se 1 ao valor da precisão, a cada errada 0. Além disto, o algoritmo genético proposto utilizam uma variação do crossover comum, chamada pelos autores de crossover inteligente, método este que é construído com idéias de modelagem de experimentos, trazida do campo da estatística.

Ros et al. (2008) desenvolve um algoritmo genético bastante específico para o problema de seleção de dados no contexto de exploração automática de bases de dados para companhias farmacêuticas. A especificidade já começa na formulação do problema, formulação esta que busca maximizar a performance da classificação, minimizar o número de atributos e maximizar a quantidade de protótipos (exemplos). O algoritmo desenvolvido neste trabalho possui duas fases. A primeira fase é responsável por promover diversidade entre as soluções da população e remover atributos totalmente desnecessários à tarefa de classificação, para assim preparar a segunda fase, que é responsável por melhorias locais nas soluções encontradas pela fase anterior. A função objetivo maximizada por este algoritmo é a composição da performance de um classificador multiplicada por duas funções sobre o número de exemplos e o número de atributos, funções estas que dão valores de piso (> 0) e de teto (≤ 1) para estas quantidades.

Além disto, operadores de crossover, mutação e busca local são desenvolvidos pensando-se no contexto apresentado.

Além destas, outras aplicações de algoritmos genéticos podem ser encontradas facilmente na literatura (KUNCHEVA; JAIN, 1999; SIERRA et al., 2001; CHEN; CHEN; HO, 2005; RAMIREZ-CRUZ et al., 2006; DERRAC; GARCıA; HERRERA, 2010). Souza, Carmo e Lima (2008) também utiliza uma metaheurística como wrapper para seleção de dados, no caso Têmpera Simulada, e diferencia-se dos trabalhos anteriores fazendo uma separação maior nos processos de seleção de atributos e de seleção de exemplos. O método apresentado neste trabalho consiste em duas instâncias da Têmpera Simulada, uma para atributos e outra para exemplos, que ocorrem separadamente e simultaneamente, comunicando-se toda vez que um subconjunto de atributos (ou exemplos) tem que ser avaliado. Esta é a forma pensada no trabalho para se relacionar os dois problemas. Quando há esta avaliação, a solução guia do outro processo é utilizada para cálculo da função objetivo do subconjunto que tem que ser avaliado.

4 FORTALDEMODS: SELEÇÃO DE DADOS VIA ADAPTAÇÃO DE TÉCNICAS DE SELEÇÃO DE ATRIBUTOS E DE SELEÇÃO DE EXEMPLOS

Neste capítulo será proposto um algoritmo de seleção de dados, aqui chamado FortalDemoDS - acrônimo para *Fortal Democratic Data Selection*, construído a partir da junção e adaptação do algoritmo de seleção de atributos FortalFS e do algoritmo de seleção de exemplos DemoIS. A chave para a construção deste algoritmo é a crença que o processo de seleção de atributos contribui positivamente na execução do processo de seleção de exemplos e vice-versa, portanto deseja-se que esta junção possibilite a geração de um algoritmo no mínimo tão bom quanto a execução sequencial do FortalFS e do DemoIS.

4.1 O Algoritmo FortalDemoDS

O Algoritmo 5 descreve o pseudo-código do FortalDemoDS. A primeira parte do algoritmo é a mesma fase inicial do algoritmo FortalFS. Um filtro de atributos é utilizado para gerar o vetor *Adão* que servirá de guia para o processo de seleção de atributos. Este vetor armazena o número de vezes que cada atributo foi selecionado na execução deste filtro que pode ser do tipo do LVF, que gera diversas soluções diferentes em uma única execução, ou um que gere apenas uma única solução (mas que seja não-determinístico). Desta forma, no momento de geração aleatória de subconjuntos de atributos, cada atributo pode ser selecionado utilizando o teste lógico $att_i = 1$, se $Adão(i) > random(k)$ e $att_i = 0$ caso contrário, sendo att_i o i-ésimo atributo, $Adão(i)$ o número de vezes que este atributo foi selecionado e k o número de execuções do filtro (ou de soluções geradas por ele).

Após esta primeira fase é que o algoritmo se diferencia das implementações anteriores, utilizando soluções intermediárias do processo de seleção de atributos como parte integrante do processo de seleção de exemplos e vice-versa. A cada iteração do algoritmo, um subconjunto a_i de atributos é selecionado, utilizando-se como guia o vetor *Adão* e os exemplos presentes em D são divididos em subconjuntos distintos $e_1, ..., e_n$ de cardinalidade *tamanho*. Estes são os procedimentos presentes nas iterações FortalFS e do DemoIS, respectivamente.

Inicialmente, cada subconjunto e_j possui todos os atributos apresentados em D, mas, no FortalDemoDS, todos os atributos que não estão presentes em a_i são removidos e esta nova base de dados contendo os a_i atributos e e_j exemplos é apresentada ao algoritmo de seleção de

Algoritmo 5: Algoritmo FortalDemoDS

```
Data: base de dados D, inteiro tamanho e inteiro iterações
Result: base de dados D'(D' ⊂ D)
Origem = Execute filtro de atributos (D);
Adão = Calcule Adão (Origem);
for i = 1 to iterações do
    a_i = Gere subconjunto de atributos(Adão);
    E = Divida a D em subconjuntos distintos cardinalidade tamanho | ∪_i e_i = D;
    taxa de erro de a_i = 0;
    for e_j ∈ E do
        Remova atributos não selecionados por a_i em e_j;
        Execute algoritmo de seleçao de exemplos (e_j);
        Conte quais exemplos foram removidos de e_j;
        d = Delete exemplos removidos em e_j;
        taxa de erro de a_i += taxa de erro(a_i, d);
    end
    taxa de erro de a_i = a_i/|E|;
    if taxa de erro de a_i < taxa de erro de melhores atributos then
        melhores atributos = a_i;
    else
        if taxa de erro de a_i = taxa de erro de melhores atributos &&
        |a_i| < |melhores atributos| then
            melhores atributos = a_i;
        end
    end
end
v = Obtenha limiar de votos para remoção de exemplos;
e = Exemplos que foram removidas menos de v vezes;
D' = Construa base de dados(D, melhores atributos, e);
return D';
```

exemplos. Esta é a principal diferença do algoritmo proposto e do DemoIS e é desta forma que o processo de seleção de atributos influencia o de seleção de exemplos. Após isto, a execução do algoritmo segue da mesma forma que no DemoIS, com a utilização de um selecionador de exemplos neste subconjunto da base de dados para selecionar os exemplos importantes e por fim, atualizar o contador de todos os exemplos que foram removidos nesta iteração.

Passada esta fase, deve-se estimar a qualidade do subconjunto a_i de atributos, como no algoritmo FortalFS. Para isto, é construída uma nova base de dados d contendo todos os a_i atributos e apenas os exemplos de e_i que foram selecionados pelo algoritmo de seleção executado na fase anterior. A partir desta nova base d é que é calculada a taxa de erro do subconjunto de atributos a_i. Como para cada subconjunto de atributos a_i existem diversos subconjuntos de exemplos e_j, a taxa de erro média é que é levada em consideração para se comparar dois subconjuntos de atributos. É através da divisão da base de dados e da remoção de exemplos supostamente não importantes que o processo de seleção de exemplos influencia o de seleção de atributos, diferenciando esta nova abordagem do algoritmo FortalFS.

Por fim, deve-se calcular o limiar de remoção de exemplos, ou seja, o número máximo de vezes que um exemplo pode ter sido removido durante o processo que ocorre na linha 9. Este cálculo é proposto no algoritmo DemoIS como sendo calculado a partir da função

$$f(l) = \alpha * e_t(l) + (1 - \alpha) * m(l)$$

onde l é o valor do limiar, que pode variar no intervalo [1..*iterações*], a função $e_t(l)$ é a taxa de erro, $m(l)$ é a porcentagem de exemplos retidos e α é um valor no intervalo [0..1] que mede a relevância relativa das duas funções. A função $e_t(l)$ é calculada retirando uma pequena porção da base de dados para utilizá-la para estimar a taxa de erro do indutor construído a partir da base de dados selecionada pelo algoritmo. Assim, para calcular o valor de l ótimo, são feitos testes com todos os valores de l no intervalo [1..*iterações*] e aquele valor que minimiza a função $f(l)$ é então escolhido. Então, a nova base de dados contendo o subconjunto de atributos com a melhor taxa de erro média e os exemplos que se encaixam no teste do limiar é retornada pelo algoritmo. A Figura 4.1 ilustra o processo de uma iteração do algoritmo FortalDemoDS.

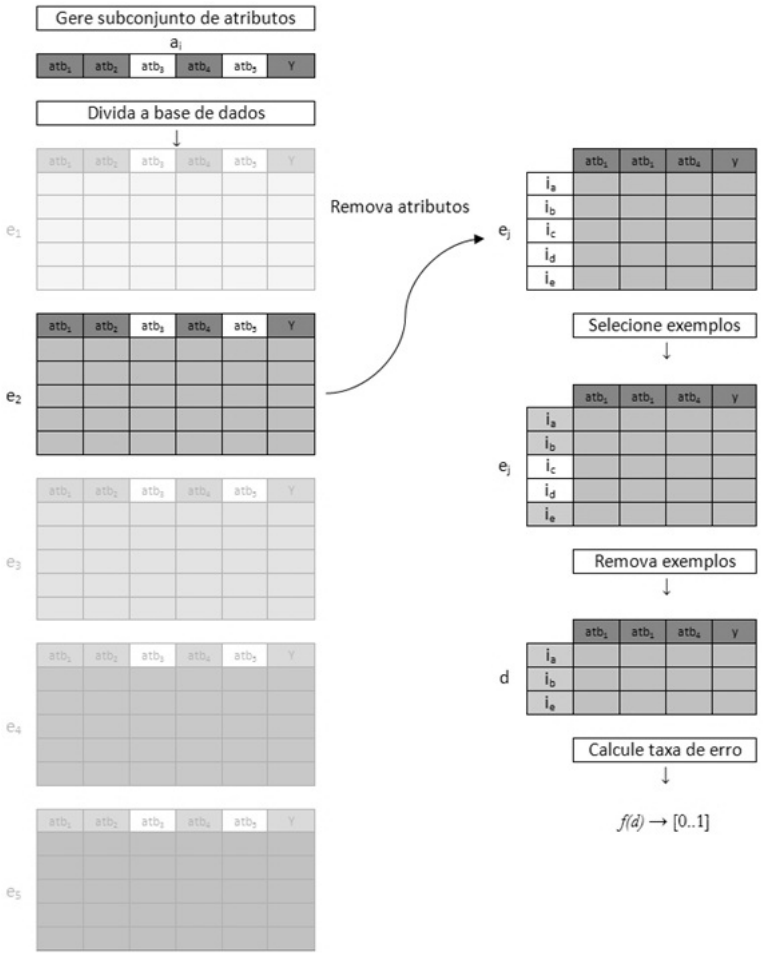

Figura 4.1: Representação gráfica de uma iteração do FortalDemoDS.

Como visto, o FortalDemoDS é bastante parecido tanto com o FortalFS quanto com o DemoIS. De fato, ele pode se comportar como os dois algoritmos. Se colocarmos como valor da variável *tamanho* a quantidade de exemplos presentes em *D* então o algoritmo se comportará

como sendo o FortalFS. Para tornarmos o comportamento do algortimo igual ao do DemoIS basta construir um filtro de seleção de atributos que sempre retorne todos o conjunto de atributos apresentado a ele, fazendo com que a execução do comando $a_i =$ **Gere subconjunto de atributos(***Adão***)** sempre retorne um conjunto a_i de atributos igual ao conjunto inicial A. Esta é a dualidade pretendida na construção do algoritmo: mesmo fazendo seleção conjunta de atributos e exemplos, o algoritmo possui características dos algoritmos "separados" que serviram como base para a geração deste novo algoritmo e modifica estas características para que um processo influencie o outro. No FortalDemoDS, a função de avaliação de um conjunto de atributos deixa de utilizar todo o conjunto de dados para cálculo da taxa de erro, como executado no FortalFS, e utiliza apenas alguns exemplos para este cálculo. Também no FortalDemoDS a avaliação da importância de um exemplo leva em consideração apenas um subconjunto de atributos, diferentemente do DemoIS que leva em consideração todo o conjunto de atributos.

4.2 Complexidade da Abordagem

Calcular complexidade desta nova abordagem é um tanto difícil, visto que alguns dos componentes do algoritmo não são definidos, tal como o filtro utilizado no processo de seleção de atributos, e portanto, não têm sua complexidade conhecida *a priori*. Portanto, aqui trataremos a formalização da complexidade do algoritmo em função da complexida destes componentes.

A complexidade de tempo do algoritmo FortalDemoDS é dependente da (a) complexidade do filtro de seleção de atributos utilizado na fase inicial do processo, (b) complexidade do algoritmo de aprendizado utilizado para avaliar a qualidade do subconjunto de atributos, (c) complexidade do processo de geração de subconjuntos de exemplos e (d) complexidade do algoritmo de seleção de exemplos utilizado. Estas são dependências já apresentadas nos algoritmos FortalFS e DemoIS.

A complexidade do algoritmo FortalFS é $O(max(O(f), \mathit{Iterações}{*}O(l)))$, onde $O(f)$ é a complexidade do filtro utilizado na primeira fase do algoritmo e $O(l)$ é a complexidade do algoritmo de aprendizado utilizado para avaliar a qualidade do subconjunto de atributos. Já a complexidade do algoritmo DemoIS é linear quanto ao número de exemplos $|e|$ apresentados na base de dados, quando é utilizado especificamente com o algoritmo kNN e é $O(max(O(|e|), \mathit{Iterações}{*}O(a)))$, onde a é o algoritmo de seleção de exemplos utilizado, quando um algoritmo qualquer é utilizado.

Do exposto, podemos calcular a complexidade do algoritmo FortalDemoDS. No processo referente à seleção de exemplos, não há diferença nenhuma, portanto esta parte das dependências do algoritmo permanece a mesma. Quanto à parte referente a seleção de atributos, ao invés de criarmos um modelo a cada iteração, criamos $s = |e|/\mathit{tamanho}$ modelos. Portanto, a complexidade do algoritmo FortalDemoDS é $O(max(O(f), s{*}\ \mathit{Iterações}{*}O(l), O(|e|), \mathit{Iterações}{*}O(a)))$.

5 AVALIAÇÃO DO FORTALDEMODS

Para avaliar o algoritmo FortalDemoDS, foi implementado um pequeno conjunto de algoritmos de seleção de atributos e exemplos. Os algoritmos utilizados foram o LVF, o FortalFS, o MVF e o DemoIS. A comparação do novo algoritmo com as abordagens para seleção de atributos e seleção de exemplos se dá pois, assim como listado no início deste trabalho, uma das hipóteses iniciais é que a seleção de dados, ou seja, seleção conjunta de atributos e exemplos, é competitiva com a seleção sequencial dos mesmos. Assim, nesta dissertação não executamos testes comparativos com abordagens focadas em seleção de dados.

5.1 Metodologia

Visando ter resultados empíricos que corroboram as hipóteses de pesquisa, executamos testes diversos na seguinte forma:

Seleção de Atributos A base de dados inicial passa por um processo de seleção de atributos e então este processo de seleção é avaliado.

Seleção de Exemplos A base de dados inicial passa por um processo de seleção de exemplos e então este processo de seleção é avaliado.

Seleção de Atributos + Seleção de Exemplos A base de dados inicial passa por um processo de seleção de atributos e então este processo de seleção é avaliado. Após isto, os atributos não selecionados por este processo são removidos da base de dados e esta nova base (contendo um subconjunto de atributos e todos os exemplos) passa pelo processo de seleção de exemplos que também é avaliado. Por fim, todo o processo de seleção de dados é avaliado.

Seleção de Exemplos + Seleção de Atributos A base de dados inicial passa por um processo de seleção de exemplos e então este processo de seleção é avaliado. Após isto, os exemplos não selecionados por este processo são removidos da base de dados e esta nova base (contendo um subconjunto de exemplos e todos os atributos) passa pelo processo de seleção de atributos que também é avaliado. Por fim, todo o processo de seleção de dados é avaliado.

Seleção de Dados A base de dados inicial passa por um processo de seleção de dados, selecionando atributos e exemplos simultaneamente, No fim, este processo é avaliado.

Para esta avaliação, as métricas *tempo de execução*, *precisão* e *redução* foram utilizadas. A métrica *tempo de execução* leva em consideração apenas o processo de seleção propriamente dito, excluindo deste cálculo a avaliação final do subconjunto gerado. Esta avaliação final é calculada como sendo a estimativa da *precisão* ou *taxa de erro* de classificador construído a partir do subconjunto gerado pelo processo de seleção, seja um subconjunto de atributos, exemplos ou dados. Para tal cálculo, utilizamos validação cruzada *10-fold*, a exceção, de uma das bases de dados presente na avaliação (a base *Texture*), visto que ela já possuía um conjunto separado para teste. Vale ressaltar que este modo de validação é propenso a overfitting Reunanen (apud SOUZA, 2004), porém como o intuito da comparação não é predizer a qualidade dos modelos gerados e sim avaliar as técnicas de seleção de dados, esta torna-se uma avaliação válida. A métrica *redução* é simplesmente a razão entre o número de atributos ou exemplos selecionados e o valor inicial destas variáveis.

Foram utilizados três diferentes algoritmos no processo de indução: C4.5, Naive Bayes e 1NN. Além disto, estes são os algoritmos que fazem parte do processo de seleção de exemplos executado no algoritmo MVF. Durante o processo de aprendizado, toda vez que se fez necessário a geração de um modelo e cálculo da taxa de erro de tal modelo (fase wrapper do FortalFS, cálculo do limiar de votos do DemoIS e por consequência, no FortalDemoDS) foi utilizada a técnica de *hold-out* com 70% dos exemplos sendo utilizados para treinamento e o restante para teste.

Todos os algoritmos implementados neste trabalho foram escritos utilizando-se a linguagem Java e a API provida pelo framework de aprendizado de máquina Weka (HALL et al., 2009). As bases de dados utilizadas neste trabalho foram retiradas do repositório UCI (ASUNCION; NEWMAN, 2007), a exceção da base *Texture* que foi removida do projeto Elena[1], e são:

Chess Base de dados descrevendo o final de um jogo de xadrez, no qual é o turno do jogador branco, jogador este que possui o rei e uma torre e o jogador preto possui o rei e um peão a uma casa de se tornar rainha. A partir deste cenário, a classe descreve se o jogador branco consegue ou não ganhar esta partida - 36 atributos e 3196 exemplos.

Libras Base de dados contendo exemplos de movimentos gestuais em Libras (Língua Brasileira de Sinais), como meio de comunicação para surdos, capturados a partir da movimentação das mãos de um inidivíduo. A classe indica o tipo de movimento executado - 90 atributos e 360 exemplos.

Satellite Base de dados contendo pequenas amostras de imagens de satélite. A classe representa que tipo de solo ocorre naquela dada região mapeada - 36 atributos e 6435 exemplos.

Spam Base de dados que contém exemplos de emails classificados como spam e como mensagens normais. Os atributos representam a ocorrência (frequência) de palavras-chave e caracteres especiais - 57 atributos e 4597 exemplos.

[1] ftp://ftp.dice.ucl.ac.be/pub/neural-nets/ELENA/databases - Sítio acessado em Abril/2010.

Splice Base de dados contendo dois tipos diferentes de cruzamentos de cadeias de DNA - 60 atributos e 3190 exemplos.

Texture Base de dados contendo texturas, sendo caracterizada por sequências de pixels - 40 atributos e 5500 exemplos.

5.2 Configuração dos Experimentos

A tabela 5.2 apresenta os parâmetros utilizados em cada execução dos diferentes algoritmos implementados para esta dissertação. Esta parametrização foi escolhida buscando um equilíbrio entre tempo de execução e a parametrização ótima encontrada na literatura.

LVF
Limiar de inconsistência = inconsistência inicial da base de dados inicial$*10$ Número de iterações: $\|F\|^2$

FortalFS
Número de iterações: $10 * \|F\|$ Filtro utilizado: LVF $k = 10$

MVF
Algoritmos indutores: C4.5, Naive Bayes e 1NN

DemoIS
Número de iterações: 10 Tamanho do subconjunto = 1000 se $\|I\| >= 2000$ ou $\|I\|/2$

FortalDemoDS
S = Tamanho do subconjunto = 1000 se $\|I\| >= 2000$ ou $\|I\|/2$ caso contrário[2] Número de iterações: $10 * \|F\| * S/\|I\|$ Filtro de atributos utilizado: LVF Seletor de exemplos utilizado: MVF

Tabela 5.1: Configuração dos algortimos. |F| é o número de atributos e |I| é o número de instâncias na base de dados original.

A configuração do algoritmo LVF foi um pouco diferente da parametrização apresentada no artigo que definiu tal algoritmo (LIU; SETIONO, 1996b). Ao invés de executar o laço do algortimo $77 * |F|^5$ vezes, como indicado, ele foi executado apenas $|F|^2$ vezes, visto que pela grande cardinalidade dos conjuntos de atributos das bases de dados apresentadas neste trabalho, excutar o algoritmo $77 * |F|^5$ se tornaria extremamente custoso. Além disto, o limiar de inconsistência é igual ao limiar de inconsistência da base de dados inicial (valor indicado no trabalho) multiplicado por 10, visto que as bases de dados utilizadas neste estudo possuem inconsistência

bastante baixas, próximas a 0. Este relaxamento da restrição se deu, dado que, com valores tão baixos, a execução inicial do LVF não retornou soluções que respeitassem o limiar ideal.

Para o FortalFS, foi utilizada a melhor configuração entre as apresentadas em Souza (2004). Na fase filtro da abordagem, foi utilizado o algoritmo LVF para gerar 10 ($k = 10$) soluções que são a semente para construção do vetor Adão e o número de iterações da fase wrapper foi calculado como sendo igual ao número de atributos presentes na base de dados multiplicado por 10.

Para o MVF utilizamos os três algoritmos de indução listados anteriormente: C4.5, Naive Bayes e 1NN. No DemoIS, foi utilizada a configuração apresentada em García-Osorio, Haro-García e García-Pedrajas (2010), com 10 iterações e o tamanho dos subconjuntos igual a 1000 se o número de exemplos na base de dados é maior ou igual a 2000 ou igual ao número de exemplos dividido por 2 caso contrário.

Por fim, toda a configuração do FortalDemoDS foi a mesma dos algoritmos FortalFS e DemoIS. Vale ressaltar que alguns cálculos foram executados para tornar a comparação um tanto "justa". A execução sequencial do FortalFS e DemoIS (ou vice-versa) gera um número fixo de iterações, subconjuntos e modelos de conhecimento gerados como apresentado na tabela 5.2. Assim, é desejável que a execução do FortalDemoDS gere um número parecido de iterações, subconjuntos e modelos. Desta forma, fixamos o número de modelos gerados numa execução do FortalDemoDS e calculamos o valor das outras variáveis a partir deste valor.

FortalFS + DemoIS
Número de iterações: $10 * \|F\|$
Número de subconjuntos: $10 * \|F\| + 10 * \|I\|/S$
Número de modelos: $10 * \|F\|$

FortalDemoDS
Número de iterações: $10 * \|F\| * S/\|I\|$
Número de subconjuntos: $10 * \|F\| * S/\|I\| + 10 * \|F\|$
Número de modelos: $10 * \|F\|$

Tabela 5.2: Características das execuções do FortalFS+DemosIS e FortalDemoDS. O valor S é igual ao parâmetro *tamanho* do algoritmo DemoIS.

Esta escolha se deu levando em consideração a configuração que menos se distanciaria de uma execução sequencial dos algoritmos. Caso quiséssemos utilizar a configuração ótima do FortalFS, então cada exemplo na base de dados seria votado $10 * |F|$ vezes mais do que numa execução do DemoIS. Caso quiséssemos utilizar a configuração ideal do DemoIS, então apenas 10 subconjuntos de dados seriam avaliados. Portanto, estes cálculos foram realizados e notou-se que caso escolhêssemos fixar o mesmo número de iterações da execução sequencial, então o número de subconjuntos visitados e o número de modelos gerados seria muito maior. Se escolhêssemos fixar o número de subconjuntos visitados, então o o número de subconjuntos de atributos avaliados decresceria mais e o número de votos de cada exemplo aumentaria mais do que se utilizássemos o mesmo número de modelos gerados, que é a configuração que menos

difere do comportamento de uma execução sequencial de tais algoritmos.

5.3 Resultados e Análises dos Experimentos

Nesta seção são apresentados e discutidos os resultados obtidos com a execução dos algoritmos apresentados anteriormente. Todas as tabelas e figuras fazem referência à execução de um algoritmo de seleção de atributos, um algoritmo de seleção de exemplos, um algoritmo de seleção de atributos seguido por um algoritmo de seleção de exemplos (e vice-versa) e um algoritmo de seleção de dados. Portanto quando houver a expressão *algoritmo A* **+** *algoritmo B*, isto significa que houve a execução do algoritmo A em uma base inicial e, a partir da solução gerada, uma nova base menor foi criada para ser utilizada como entrada para a execução do algortimo B. Por exemplo, *LVF + MVF* indica que primeiro ocorreu a execução do algoritmo LVF e em seguida o algoritmo MVF.

Foi executado um total de 90 experimentos, tendo em vista que foram realizadas 5 repetições de cada algoritmo base nas combinações de 6 bases de dados e 3 algoritmos de indução. Foram executadas 13 combinações de algoritmos de seleção de atributos e de exemplos: LVF, LVF + MVF, LVF + DemoIS, FortalFS, FortalFS + MVF, FortalFS + DemoIS, MVF, MVF + LVF, MVF + FortalFS, DemoIS, DemoIS + LVF, DemoIS + FortalFS e FortalDemoDS. Além destes experimentos, também calculamos a taxa de erro dos algoritmos Naive Bayes, C4.5 e INN com bases de dados sem nenhum tipo de pré-seleção.

5.3.1 Análise - Naive Bayes

Nesta subseção analisaremos o comportamento apresentado por todas as abordagens de seleção de dados quando foi utilizado o algoritmo de indução Naive Bayes. A Figura 5.1 apresenta o erro médio obtido por elas em cada base de dados. Como pode ser visto, as abordagens de seleção de dados (todas aquelas com '+' e o FortalDemoDS) tiveram comportamento **médio** similar em termos de erro obtido. Além disto, salvo algumas exceções, a execução sequencial dos algoritmos de seleção de atributos e exemplos levou a melhores taxas de erro do que a execução de apenas um destes passos.

Apesar de obter comportamento médio similar, a Figura 5.2 mostra que em 17 dos 30, isto é, 56,67% dos experimentos executados, o algoritmo FortalDemoDS alcançou a melhor taxa de erro dentre as 13 abordagens concorrentes. Além disto, também é percebida a presença dos algoritmos FortalFS e DemoIS em quase todas as outras abordagens que obtiveram desempenho vencedor. Novamente, abordagens que selecionam apenas atributos ou apenas exemplos foram as menos bem sucedidas, havendo apenas uma ocorrência do MVF como abordagem vencedora.

Já a Figura 5.3 mostra a taxa média de remoção de atributos para cada abordagem implementada. Obviamente, LVF e FortalFS não obteram nenhum grau de remoção de exemplos e MVF e DemoIS não removeram nenhum atributo. A partir deste gráfico pode-se perceber que houve uma pequena variação na taxa de remoção de atributos nas abordagens (na média

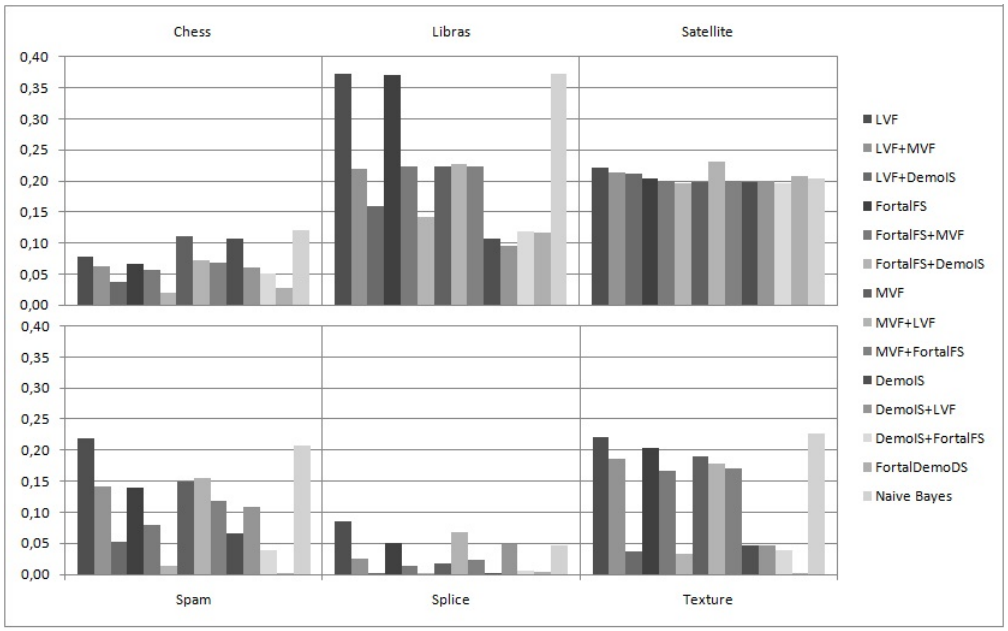

Figura 5.1: Taxa de erro média [0..1]. A coluna *Naive Bayes* indica que a base de dados inicial foi apresentada ao algoritmo.

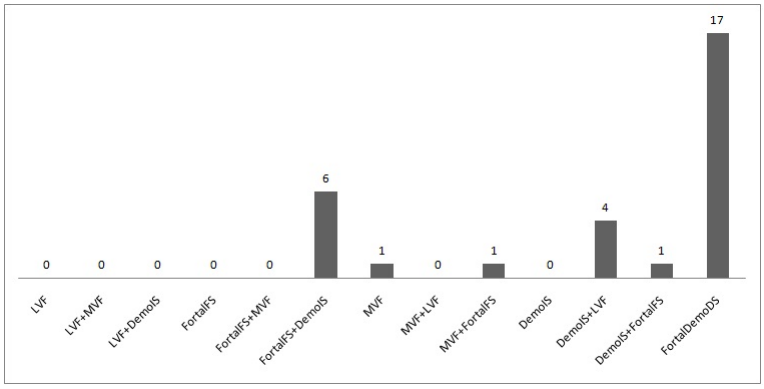

Figura 5.2: Quantidade de experimentos nos quais a abordagem alcançou a menor taxa de erro.

todas ficaram entre 50% e 60% de remoção de atributos). Porém, tratando-se de remoção de exemplos, o algoritmo FortalDemoDS obteve uma taxa expressivamente maior que as demais, alcançando 55% enquanto a mais próxima, obtida por LVF + DemoIS, alcançou 29%.

Em se tratando de tempo de execução, a Figura 5.4 mostra que o algoritmo FortalDemoDS foi consideravelmente mais lento que as demais abordagens. Isto se deu visto que, apesar do FortalDemoDS construir o mesmo número de modelos que uma execução sequencial dos algoritmos FortalFS e DemoIS, no FortalDemoDS a cada vez que se constrói um modelo também se seleciona exemplos, tarefa que não ocorre na abordagem sequencial.

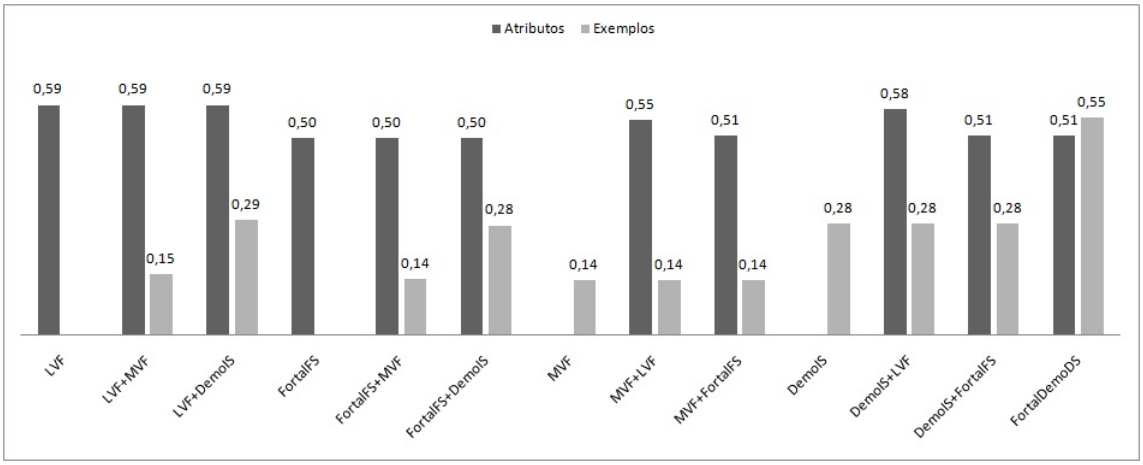

Figura 5.3: Taxa média [0..1] de remoção de atributos e exemplos.

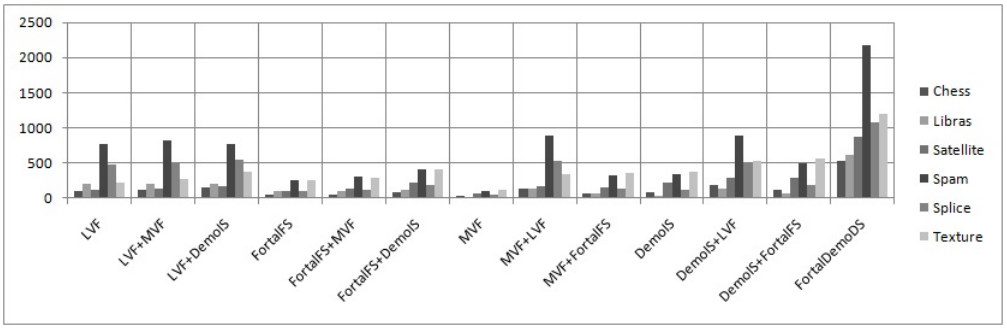

Figura 5.4: Tempo médio (em segundos) gasto por cada algoritmo.

5.3.2 Análise - C4.5

Nesta subseção analisaremos o comportamento apresentado por todas as abordagens de seleção de dados quando foi utilizado o algoritmo de indução C4.5. A Figura 5.5, que refere-se à taxa de erro média apresentada pelas abordagens implementadas, possui algumas características interessantes: a primeira é que o comportamento, em linhas gerais, é bastante parecido com o apresentado nos testes utilizando o algoritmo *Naive Bayes*, ou seja, abordagens de seleção de dados foram geralmente melhores que abordagens de seleção ou de exemplos ou de atributos e as abordagens sequenciais apresentaram um ganho efetivo ao se executar a segunda fase de seleção, salvo alguns casos onde houve piora da taxa de erro, tal como na abordagem LVF + DemoIS na base de dados *Satellite*. Além disto, a utilização do algoritmo de indução C4.5 causou uma diminição geral das taxas de erro apresentadas pelos algoritmos.

Diferentemente do apresentado na análise anterior, a Figura 5.6 mostra que houve uma maior distribuição do número de abordagens com o melhor desempenho em um dado experimento, mas mesmo assim o algoritmo FortalDemoDS apresentou um número grande de "vitórias". Novamente as abordagens de seleção de atributos ou seleção de exemplos não obtiveram bom desempenho, a exceção da abordagem DemoIS que em três experimentos obteve a melhor taxa de erro.

Em se tratando de remoção de atributos e exemplos (Figura 5.7), mais uma vez houve

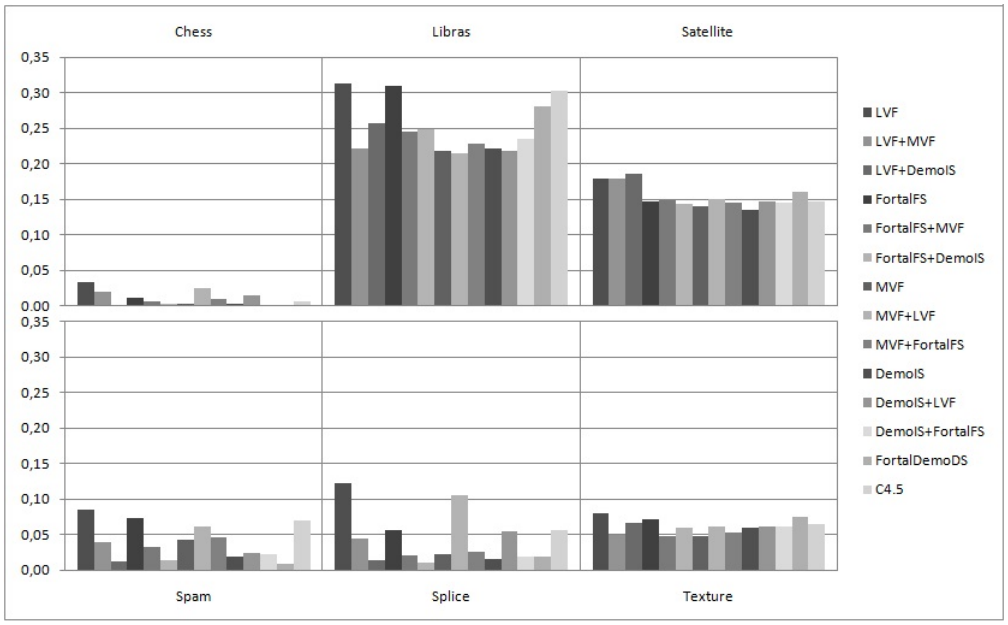

Figura 5.5: Taxa de erro média [0..1]. A coluna *C4.5* indica que a base de dados inicial foi apresentada ao algoritmo.

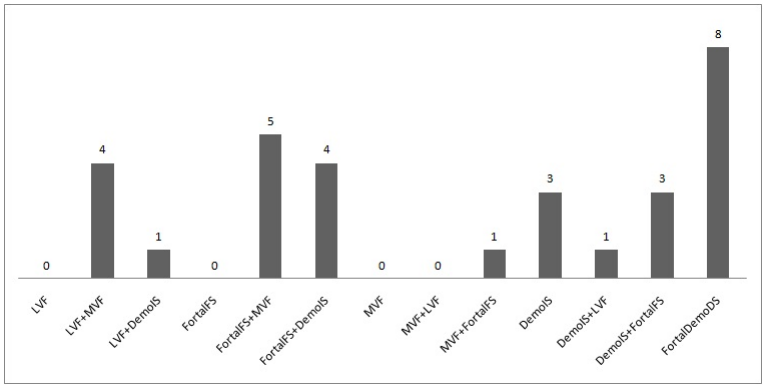

Figura 5.6: Número de experimentos nos quais a abordagem alcançou a melhor taxa de erro.

grande semelhança entre a taxa média de remoção de atributos de todas as abordagens e, novamente, o algoritmo FortalDemoDS obteve um grau de remoção de exemplos expressivamente maior que os demais. Uma possível explicação para este fenômeno é que, comparando-se como o DemoIS, a parte de seleção do FortalDemoDS é executada muito mais vezes, assim há um refinamento maior dos exemplos selecionados.

Novamente, o algoritmo FortalDemoDS foi mais lento que os demais, como visto na Figura 5.8.

5.3.3 Análise - 1NN

Nesta subseção analisaremos o comportamento apresentado por todas as abordagens de seleção de dados quando foi utilizado o algoritmo de indução 1NN. A Figura 5.9 refere-se à taxa de erro média, que novamente apresentou o mesmo tipo de comportamento visto nos testes

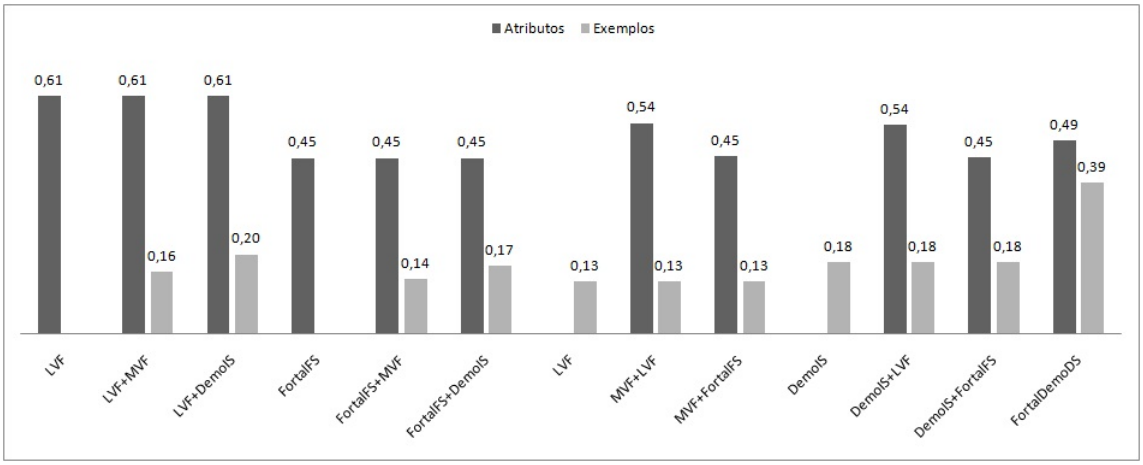

Figura 5.7: Taxa média [0..1] de remoção de atributos e exemplos.

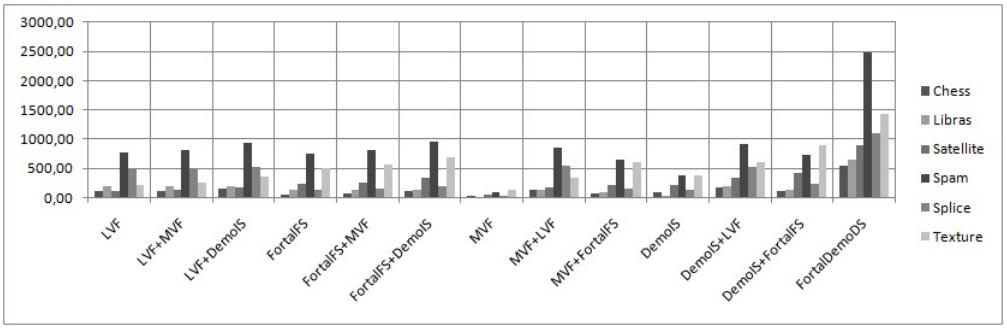

Figura 5.8: Tempo médio (em segundos) gasto por cada algoritmo.

com os outros algoritmos de indução, com abordagens de seleção de dados sendo geralmente melhores que abordagens de seleção ou de exemplos ou de atributos apenas, e algumas exceções nas quais a execução sequencial ocasionou piora na taxa de erro. Além disto, a utilização do algoritmo de indução 1NN na base de dados *Splice* se mostrou bastante ruim, obtendo taxas de erro bem maiores que as anteriores.

Mais uma vez o algoritmo FortalDemoDS foi a abordagem que mais vezes conseguiu alcançar a melhor taxa de erro 5.10. Vale ressaltar também, o bom desempenho do algoritmo DemoIS, presente em quase todas as demais abordagens vencendoras.

Como visto na Figura 5.12, novamente houve um comportamento parecido quanto ao número de atributos removidos por todas as abordagens e o FortalDemoDS obteve um grau de redução bem maior que as abordagens concorrentes.

Por fim, a Figura **??** mostra o comportamento dos algoritmos em termos de tempo de execução.

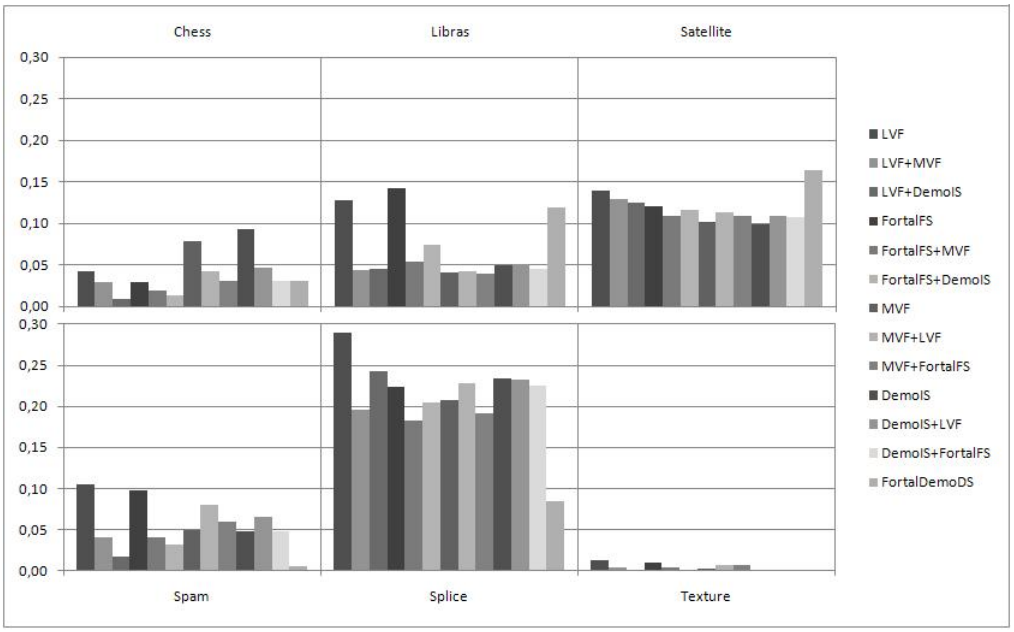

Figura 5.9: Taxa de erro média [0..1]. A coluna *1NN* indica que a base de dados inicial foi apresentada ao algoritmo.

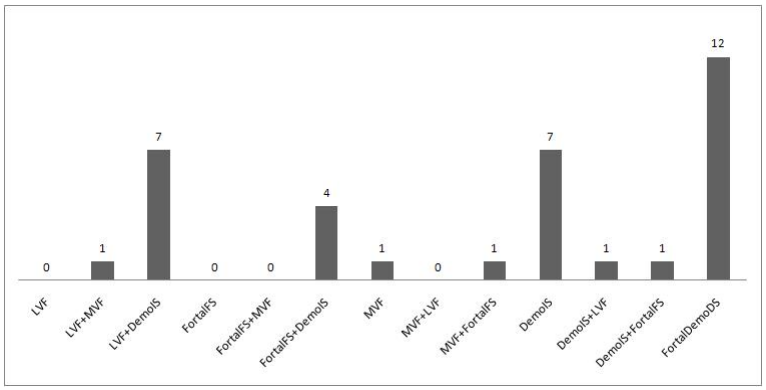

Figura 5.10: Número de experimentos nos quais a abordagem alcançou a melhor taxa de erro.

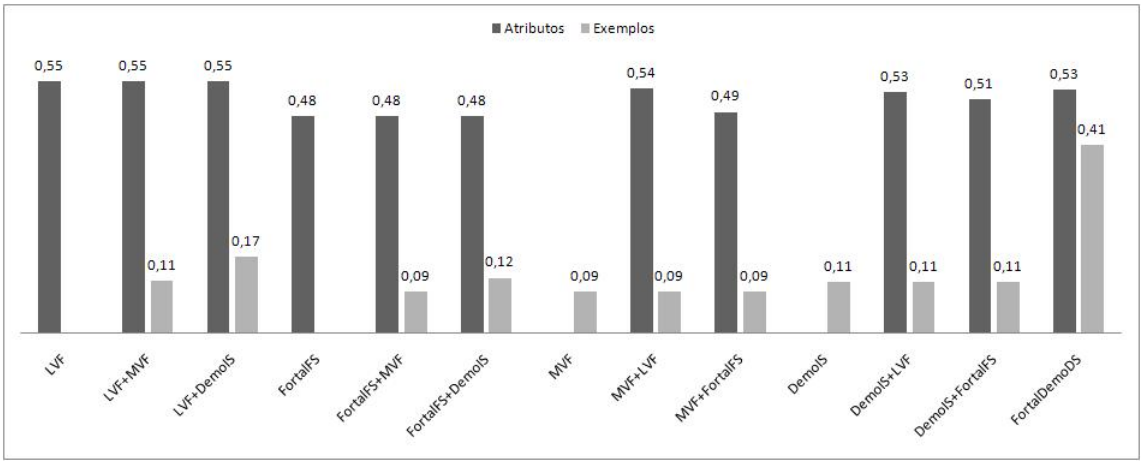

Figura 5.11: Taxa média [0..1] de remoção de atributos e exemplos.

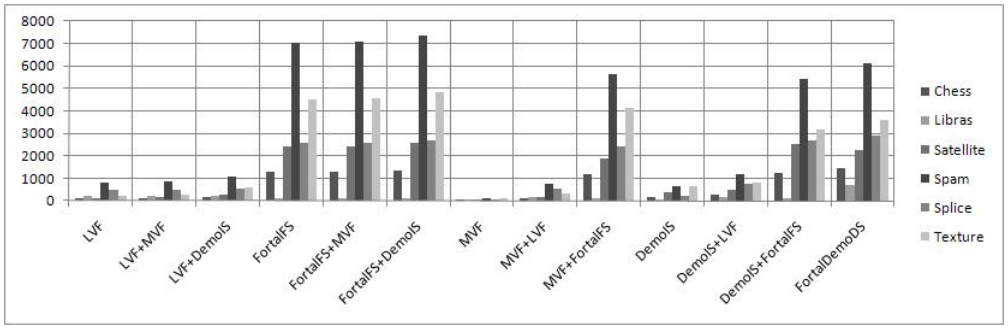

Figura 5.12: Tempo médio (em segundos) gasto por cada algoritmo.

5.4 Comparação Pareada

Através de uma comparação pareada entre o algoritmo FortalDemoDS e a execução sequencial dos algoritmos FortalFS e DemoIS nós podemos calcular a melhora obtida na execução combinada dos dois algoritmos no FortalDemoDS em comparação com a execução sequencial dos mesmos, em FortalFS + DemoIS e DemoIS + FortalFS.

Para tal comparação, utilizamos o teste estatístico *student t-test*. Como percebido na Figura 5.13, as duas abordagens sequenciais e o algoritmo FortalDemoDS possuem comportamento parecido, assim podemos dizer que tais abordagens obtêm resultados similares. Porém ao analisarmos a taxa de redução do FortalDemoDS, vemos que este algoritmo obteve uma taxa parecida de redução de atributos e expressivamente maior de exemplos, portanto apesar do "empate", pode-se afirmar que o algoritmo FortalDemoDS processa uma seleção melhor, visto que obtém uma taxa estatisticamente similar às outras abordagens e ao mesmo tempo consegue reduzir mais a base de dados.

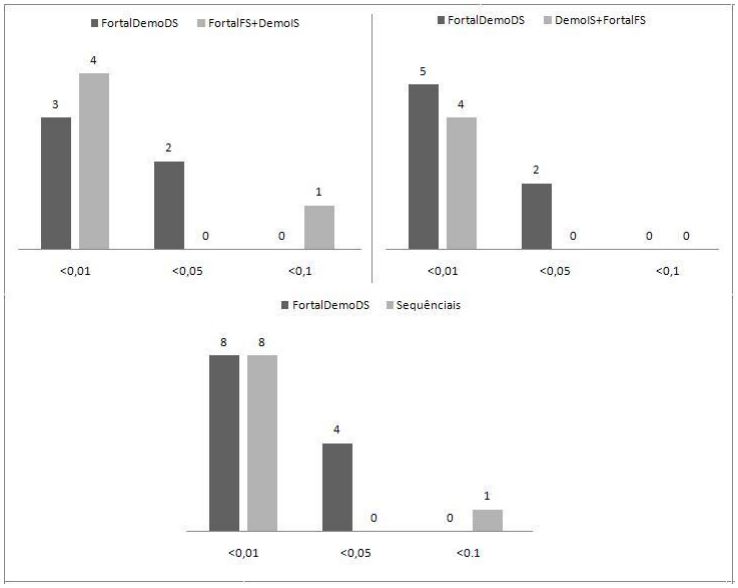

Figura 5.13: Número de ocorrências em que cada algoritmo obteve performance estatisticamente melhor, separado por grau de confiança.

5.5 Análise - Tempo Limitado

A fim de verificar se a qualidade dos subconjuntos gerados pelo algoritmo FortalDemoDS está ligada à qualidade do processo de busca ou ao custo computacional elevado apresentado em relação às demais abordagens, foi realizada mais uma série de testes. Nesta série, o código-fonte do algoritmo foi modificado para que seu tempo de execução fosse limitado. Assim, ao alcançar o limite de tempo máximo permitido, o algoritmo termina a iteração que estava executando e retorna a nova base de dados.

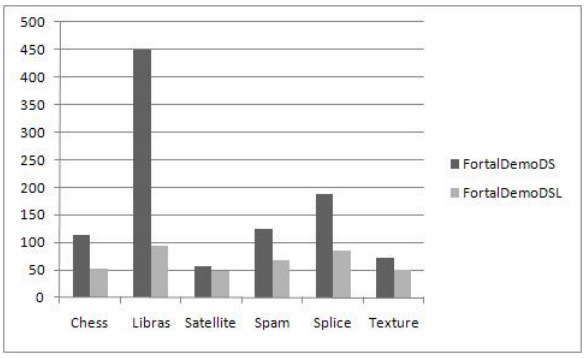

Figura 5.14: Número médio de iterações executadas por cada versão do algoritmo.

Para tal comparação, utilizaremos a sigla FortalDemoDSL para referirmos-nos à versão do FortalDemoDS com limite de tempo. Este limite de tempo foi dimensionado como a média entre o tempo médio despendido pela abordagem FortalFS + DemoIS e pela abordagem DemoIS + FortalFS. Assim, como apresentado na Figura 5.14, o número médio de iterações executadas pelo FortalDemoDSL foi bastante menor que o número de execuções do FortalDemoDS.

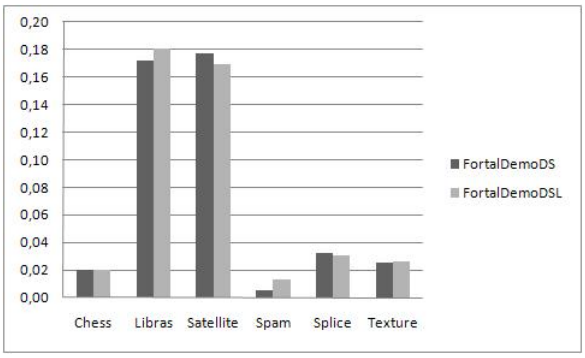

Figura 5.15: Erro médio obtido por cada algoritmo.

Mesmo assim, como visto na Figura 5.15, a taxa de erro média apresentada pelas duas versões do algoritmo é bastante parecida, não sendo relevante e não chegando a 2% de diferença.

Tal fenômeno também ocorre com o percentual de atributos (Figura 5.16) e exemplos (Figura 5.17) removido por cada abordagem. Esta diferença insignificante e o fato de que o FortalDemoDS é competitivo comparando-se com as abordagens sequenciais do FortalFS e do DemoIS demonstram a rápida convergência do algoritmo, não sendo necessário gerar o mesmo

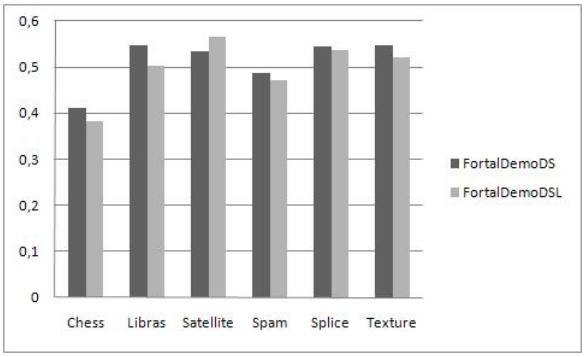

Figura 5.16: Taxa média de atributos removidos por cada algoritmo.

número de modelos que a abordagem sequencial do FortalFS e DemoIS para se obter boas taxas de erro, remoção de atributos e remoção de exemplos.

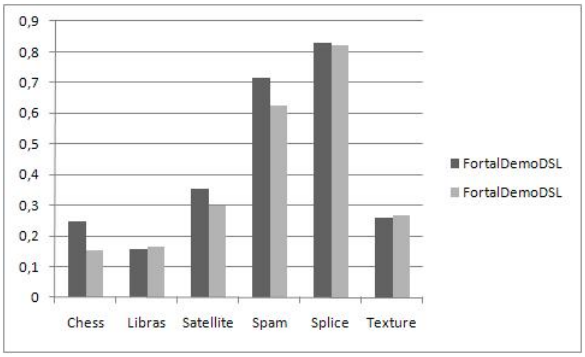

Figura 5.17: Taxa média de exemplos removidos por cada algoritmo.

6 CONCLUSÃO

6.1 Resumo Expandido

Vivemos na chamada Sociedade da Informaçãoo, uma forma de arranjo social na qual a chave do desenvolvimento científico e tecnológico está altamente relacionada com a capacidade de se transformar dados facilmente coletados e armazenados em grande volume em informação escassa, não-trivial e complexa, que possa ser compreendida e que agregue valor à atividade desenvolvida seja por uma empresa, uma universidade, uma instituicao qualquer ou mesmo um empreendedor autônomo.

Diversas são as abordagens que tentam automatizar este processo de transformação de dados em informação e Mineracao de Dados e um ramo da Inteligência Artificial que o faz através do uso extensivo de técnicas, algoritmos e ferramentas de Aprendizagem de Máquina, sendo extremamente útil em ambientes onde há grande volume de dados (na casa dos Giga ou até mesmo Terabytes) e a aplicação de técnicas estatísticas ou do trabalho de um especialista se tornam atividades complexas demais e pouco produtivas.

Dentre as várias atividades executadas durante um processo de Mineração de Dados, destaca-se na fase de pré-processamento a tarefa de Seleção de Dados. Este trabalho faz uma descrição aprofundado desta tarefa, bem como – a partir do estudo das técnicas que realizam-na – propõe um modo se construir abordagens para tal tarefa.

A literatura descreve o problema relacionado à seleção de dados dividindo esta tarefa em seleção de atributos e seleção de exemplos. Diversas são as abordagens que tratam seleção de atributos e outras tantas tratam seleção de exemplos, sem levar em consideração a relação existente entre estas duas entidades. Das abordagens que tratam seleção de dados como um todo, resolvendo seleção de atributos e seleção de exemplos simultaneamente, poucas são aquelas que tentam resolver tal problema reutilizando conceitos, características e partes de algoritmos que já resolvem este problema.

Desta forma, nesta dissertação, propõe-se o algoritmo para seleção de dados FortalDemoDS. Trata-se de um algoritmo híbrido que seleciona atributos de forma bem parecida com o modo do algoritmo de seleção de atributos FortalFS e seleciona exemplos de forma similar ao algoritmo de seleção de exemplos DemoIS. Este novo algoritmo trata estas duas seleções simultaneamente, fazendo com que o processo de seleção de atributos tenha influência direta sobre o processo de seleção de exemplos e vice-versa.

Alguns algoritmos foram implementados para servirem de comparação a este novo

algoritmo e os resultados empíricos dos testes executados mostraram que esta nova abordagem para seleção de dados é competitiva, tanto em relação ao esforço despendido como em qualidade de soluções geradas, quando comparada com abordagens sequenciais para este problema.

6.2 Contribuições da Dissertação

Fazendo um apanhado do que foi desenvolvido durante esta dissertação, podemos citar como contribuições:

Uma nova abordagem para seleção de dados O FortalDemoDS é um algoritmo híbrido para seleção de dados que faz uso de dois outros algoritmos, o FortalFS de seleção de atributos e o DemoIS de seleção de exemplos, fazendo com que o processo de busca pelo melhor conjunto de atributos seja influenciado pelo processo de busca pelo melhor conjunto de exemplos e vice-versa. Esta característica de ser construído a partir de dois outros algoritmos diferentes e resolver o problema de seleção de dados conjuntamente é uma característica única e que se mostrou bastante promissora.

Uma novo modo de se construir algoritmos para seleção de dados Salvo engano, nenhuma outra abordagem apresentada na literatura atual foi construída, de forma tão clara e transparente, através da junção e adaptação de um algoritmo de seleção de atributos e outro de seleção de exemplos. Desta forma, esta idéia de se reaproveitar abordagens de seleção de atributos e de exemplos para se construir algoritmos de seleção de dados pode contribuir fortemente para a evolução da área de pesquisa em seleção de dados.

6.3 Trabalhos Futuros

Dentre as diversas possibilidades criadas através do estudo realizado para a construção desta dissertação, podemos citar como trabalhos futuros:

Maior número de análises Foram executados diversos experimentos a fim de validar a hipótese de que efetuar seleção de dados de forma simultânea é competitivo com efetuar seleção de atributos e seleção de exemplos de forma sequencial, o que é a principal contribuição do trabalho. Porém, para uma melhor validação do algoritmo FortalDemoDS em si, é necessário também que ele seja comparado com outras abordagens de seleção de dados, como por exemplo, aquelas apresentadas no capítulo de trabalhos relacionados.

Análise da parametrização do algoritmo Neste trabalho, a parametrização do algoritmo FortalDemoDS se deu de forma que houvesse uma comparação "justa" com os outros algoritmos implementados para o trabalho. Os resultados obtidos mostram que ao utilizar esta parametrização o algoritmo obtém boas soluções mas não necessariamente esta parametrização faz com que o algoritmo gere soluções ótimas. Um refinamento da parametrização do mesmo pode trazer este benefício.

Construção de novos algoritmos O algoritmo FortalDemoDS pode ser modificado e alterado para a construção de novas versões do mesmo, porém, algo muito interessante que pode ser desenvolvido é a idéia de se reutilizar algoritmos de seleção de atributos e algoritmos de seleção de exemplos para a construção de novos algoritmos de seleção de dados. Esta idéia tem apoio em resultados empíricos anteriormente publicados e pode levar à construção de algoritmos de seleção de dados de alta qualidade, visto que tais novos algoritmos possuiriam características específicas para atributos e para exemplos e também juntariam estas duas tarefas, fazendo uso do elo existente entre estas duas entidades.

REFERÊNCIAS BIBLIOGRÁFICAS

AHA, D.; KIBLER, D. Instance-based learning algorithms. *Machine Learning*, v. 6, p. 37–66, 1991.

ALMUALLIM, H.; DIETTERICH, T. G. Learning with many irrelevant features. In: *Proceedings of the AAAI-91*. [S.l.: s.n.], 1991.

ASUNCION, A.; NEWMAN, D. *UCI Machine Learning Repository*. 2007. Disponível em: <http://www.ics.uci.edu/∼mlearn/MLRepository.html>.

BERITELLI, F. et al. A genetic algorithm feature selection approach to robust classification between ”positive”and ”negative”emotional states in speakers. In: *Proceedings of the Thirty-Ninth Asilomar Conference on Signals, Systems and Computers*. [S.l.: s.n.], 2005.

BISHOP, C. M. *Pattern Recognition and Machine Learning*. [S.l.]: Springer, 2007.

BLUM, A. L.; LANGLEY, P. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, v. 97, p. 245–271, December 1997.

BRIGHTON, H.; MELLISH, C. Advances in instance selection for instance-based learning algorithms. *Data Mining and Knowledge Discovery*, v. 6, n. 2, p. 153–172, April 2002.

BRODLEY, C. E.; FRIEDL, M. A. Identifying mislabeled training data. *Journal of Artificial Intelligence Research*, v. 11, p. 131–167, 1999.

CANOA, J. R.; HERRERA, F.; LOZANO, M. Using evolutionary algorithms as instance selection for data reduction in kdd: an experimental study. *IEEE Transactions on Evolutionary Computation*, v. 7, n. 6, p. 561–575, December 2003.

CHAPMAN, P. et al. *CRISP-DM 1.0 Step-by-step data mining guide*. [S.l.], 2000.

CHEN, J.-H.; CHEN, H.-M.; HO, S.-Y. Design of nearest neighbor classifiers multi-objective approach. *International Journal of Approximate Reasoning*, v. 40, n. 1-2, p. 3–22, July 2005.

DERRAC, J.; GARCıA, S.; HERRERA, F. Ifs-coco: Instance and feature selection based on cooperative coevolution with nearest neighbor rule. *Pattern Recognition*, v. 43, p. 2082–2105, 2010.

EMMANOUILIDIS, C.; HUNTER, A.; MACINTYRE, J. A multiobjective evolutionary setting for feature selection and acommonality-based crossover operator. In: *Proceedings of the 2000 Congress on Evolutionary Computation*. [S.l.: s.n.], 2000. v. 1, p. 309–316.

FRAGOUDIS, D.; MERETAKIS, D.; LIKOTHANASSIS, S. Integrating feature and instance selection for text classification. In: *Proceedings of the eighth ACM SIGKDD international conference on Knowledge Discovery and Data Mining*. [S.l.]: ACM, 2002. p. 501–506. Poster paper.

FRANK, A. *A New Branch and Bound Feature Selection Algorithm*. Dissertação (Mestrado) — Israel Institute of Technology, June 2002.

GARCíA-OSORIO, C.; HARO-GARCíA, A. de; GARCíA-PEDRAJAS, N. Democratic instance selection: A linear complexity instance selection algorithm based on classifier ensemble concepts. *Artificial Intelligence*, 2010.

GARCíA-PEDRAJAS, N.; CASTILLO, J. A. R. del; ORTIZ-BOYER, D. A cooperative coevolutionary algorithm for instance selection for instance-based learning. *Machine Learning*, v. 78, n. 3, p. 381–420, 2009.

GOLDBERG, D. E. *Genetic Algorithms in Search, Optimization, and Machine Learning*. 1. ed. [S.l.]: Addison-Wesley Professional, 1989. ISBN 0201157675.

GUYON, I.; ELISSEEFF, A. An introduction to variable and feature selection. *The Journal of Machine Learning Research*, v. 3, p. 1157–1182, 2003.

GUYON, I. et al. *Feature Extraction Foundations and Applications*. [S.l.]: Springer, 2006.

HALL, M. et al. The weka data mining software: An update. *SIGKDD Explorations*, v. 11, 2009.

HARO-GARCíA, A. de; GARCíA-PEDRAJAS, N. A divide-and-conquer recursive approach for scaling up instance selection algorithms. *Data Mining and Knowledge Discovery*, v. 18, n. 3, p. 392–418, 2009.

HO, S.-Y.; LIU, C.-C.; LIU, S. Design of an optimal nearest neighbor classifier using an intelligent genetic algorithm. *Pattern Recognition Letters*, v. 23, n. 13, p. 1495–1503, November 2002.

HUANG, J.; CAI, Y.; XU, X. A hybrid genetic algorithm for feature selection wrapper based on mutual information. *Pattern Recognition Letters*, v. 28, p. 1825–1844, 2007.

ISHIBUCHI, H.; NAKASHIMA, T. Multi-objective pattern and feature selection by a genetic algorithm. In: *Proceedings of Genetic and Evolutionary Computation Conference (GECCO'2000)*. [S.l.]: Morgan Kaufmann, 2000. p. 1069–1076.

JAPKOWICZ, N. The class imbalance problem: Significance and strategies. In: *Proceedings of the 2000 International Conference on Artificial Intelligence*. [S.l.: s.n.], 2000.

JOHN, G. H.; KOHAVI, R.; PFLEGER, K. Irrelevant features and the subset selection problem. In: *Proceedings of the 11th International Conference on Machine Learning*. [S.l.]: Morgan Kaufmann, 1994. p. 121–129.

JOHN, G. H.; LANGLEY, P. Estimating continuous distributions in bayesian classifiers. In: *Eleventh Conference on Uncertainty in Artificial Intelligence*. San Mateo: Morgan Kaufmann, 1995. p. 338–345.

KIM, K. jae. Artificial neural networks with evolutionary instance selection for financial forecasting. *Expert Systems with Applications*, v. 30, n. 3, p. 519–526, April 2006.

KIRA, K.; RENDELL, L. A. The feature selection problem: Traditional methods and a new algorithm. In: *Proceedings of the Tenth National Conference on Artificial lntelligence*. [S.l.: s.n.], 1992.

KIRKPATRICK, S. et al. Optimization by simulated annealing. *Science*, v. 220, p. 671–680, 1983.

KOHAVI, R.; JOHN, G. H. Wrappers for feature subset selection. *Artificial Intelligence*, v. 97, n. 1-2, p. 273–324, December 1997.

KUNCHEVA, L. I. Editing for the k-nearest neighbors rule by a genetic algorithm. *Pattern Recognition Letters*, v. 16, n. 8, p. 809–814, August 1995.

KUNCHEVA, L. I.; JAIN, L. C. Nearest neighbor classifier: Simultaneous editing and feature selection. *Pattern Recognition Letters*, v. 20, n. 11-13, p. 1149–1156, November 1999.

LEE, Y.-J.; MANGASARIAN, O. L. Rsvm: Reduced support vector machines. In: *Proceedings of the First SIAM International Conference on Data Mining*. [S.l.: s.n.], 2001.

LI, Y.; PAL, S. K.; SHIU, S. C. K. Combining feature reduction and case selection in building cbr classifiers. *IEEE Transactions on Knowledge and Data Engineering*, v. 18, n. 3, p. 415–429, March 2006.

LITTLESTONE, N.; MESTERHARM, C. An apobayesian relative of winnow. *Advances in Neural Information Processing Systems*, v. 9, 1997.

LIU, H.; MOTODA, H. On issues of instance selection. *Data Mining and Knowledge Discovery*, v. 6, n. 2, p. 115–130, April 2002.

LIU, H.; MOTODA, H.; YU, L. A selective sampling approach to active feature selection. *Artificial Intelligence*, v. 159, n. 1-2, p. 49–74, November 2004.

LIU, H.; SETIONO, R. Feature selection and classification - a probabilistic wrapper approach. In: *Proceedings of the 9th International Conference on Industrial and Engineering Applications of AI and ES*. [S.l.: s.n.], 1996.

LIU, H.; SETIONO, R. A probabilistic approach to feature selection - a filter solution. In: . [S.l.: s.n.], 1996.

MARCHIORI, E. Class conditional nearest neighbor for large margin instance selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 32, p. 364–370, 2010.

MITCHELL, T. *Machine Learning*. [S.l.]: McGraw-Hill, 1997.

OLIVEIRA, L. S. et al. Feature selection using multi-objective genetic algorithms for handwritten digit recognition. In: *Proceedings of the 16 th International Conference on Pattern Recognition (ICPR'02)*. [S.l.]: IEEE Computer Society, 2002.

PEDRYCZ, W.; PARK, B.; PIZZI, N. Identifying core sets of discriminatory features using particle swarm optimization. *Expert Systems with Applications: An International Journal of Computer Science and Technology*, v. 36, p. 4610–4616, 2009.

QUINLAN, J. R. Induction of decision trees. *Machine Learning*, v. 1, p. 81–106, 1986.

QUINLAN, J. R. *C4.5: Programs for Machine Learning*. [S.l.]: Morgan Kaufmann Publishers, 1993.

RAMIREZ-CRUZ, J.-F. et al. Instance selection and feature weighting using evolutionary algorithms. In: *Proceedings of the 15th International Conference on Computing*. [S.l.]: IEEE Computer Society, 2006. p. 73–79.

REINARTZ, T. A unifying view on instance selection. *Data Mining and Knowledge Discovery*, v. 6, n. 2, p. 191–210, April 2002.

REUNANEN, J. Overfitting in making comparisons between variable selection methods. *Journal of Machine Learning Research*, JMLR.org, v. 3, p. 1371–1382, 2003. ISSN 1532-4435.

ROS, F. et al. Hybrid genetic algorithm for dual selection. *Pattern Analysis & Applications*, v. 11, n. 2, p. 179–198, May 2008.

ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, v. 65, n. 6, p. 386–408, 1958.

SAEED, M. Bernoulli mixture models for markov blanket filtering and classification. *Journal of Machine Learning Research*, JMLR.org, v. 3, 2008. ISSN 1532-4435.

SIERRA, B. et al. Prototype selection and feature subset selection by estimation of distribution algorithms. a case study in the survival of cirrhotic patients treated with tips. In: *Proceedings of the 8th Conference on AI in Medicine in Europe: Artificial Intelligence Medicine*. [S.l.]: Springer-Verlag, 2001. (Lecture Notes In Computer Science, v. 2101), p. 20–29.

SKALAK, D. B. Prototype and feature selection by sampling and random mutation hill climbing algorithms. In: *Proceedings of the Eleventh International Conference (ICML'94)*. [S.l.]: Morgan Kaufmann, 1994. p. 293–301.

SOUZA, J. T. de. *Feature Selection with a General Hybrid Algorithm*. Tese (Doutorado) — School of Information Technology and Engineering (SITE), University of Ottawa, 2004.

SOUZA, J. T. de; CARMO, R. A. F. do; LIMA, G. A. C. de. A novel approach for integrating feature and instance selection. In: *Proceedings of the 2008 International Conference on Machine Learning and Cybernetics*. [S.l.: s.n.], 2008. v. 1, p. 374–379.

SUN, Z. et al. Genetic feature subset selection for gender classification: A comparison study. In: *Proceedings of the Sixth IEEE Workshop on Applications of Computer Vision (WACV'02)*. [S.l.: s.n.], 2002.

THEODORIDIS, S.; KOUTROUMBAS, K. *Pattern Recognition*. [S.l.]: Academic Press, 2006.

VAFAIE, H.; JONG, K. D. Genetic algorithms as a tool for feature selection in machine learning. In: *Proceedings of the 1992 IEEE International Conference on Tools with AI*. [S.l.: s.n.], 1992.

WANG, C.-M.; HUANG, Y.-F. Evolutionary-based feature selection approaches with new criteria for data mining: A case study of credit approval data. *Expert Systems with Applications: An International Journal*, v. 36, n. 3, p. 5900–5908, April 2009.

WILSON, R.; MARTINEZ, T. R. Reduction techniques for instance-based learning algorithms. *Machine Learning*, v. 38, n. 3, p. 257–286, March 2000.

Apêndice

Neste apêndice são apresentadas as tabelas que indicam o comportamento médio da execução de cada algoritmo implementado durante a construção desta dissertação. Cada linha da tabela possui uma sigla dentre as possibilidades: NA - número de atributos, NE - número de exemplos, ER - taxa de erro, TT - tempo total e TE - tempo de execução. Para mostrar como se deve fazer a leitura destes valores, utilizaremos as linhas referentes à execução da estratégia LVF e LVF + MVF, utilizando o Naive Bayes como algoritmo de indução e aplicando tais estratégias na base de dados Chess. A tabela inicia com o relatório referente à execução do LVF unicamente, ou seja, são apresentados os dados número de atributos (NA - 19,6), taxa de erro (ER - 0,078411) e tempo total (TT - 101487,2). Logo a seguir está apresentado o resultado da execução da estratégia LVF + MVF. Esta estratégia executa primeiramente o algoritmo de seleção de atributos LVF e após isto o de seleção de exemplos MVF. Assim, a execução anterior do LVF é reaproveitada e a solução gerada por este algoritmo é utilizada para reduzir o número de atributos da base de dados que foi apresentada ao algoritmo MVF. Desta forma, para apresentar os resultados da estratégia LVF + MVF, restam apenas três dados que precisam ser indicados: número de exemplos selecionados (NE - 3118,6), taxa de erro calculada após todo o processo (ER - 0,06217), tempo de execução do segundo algoritmo de seleção (TE - 12532,8) e tempo total da abordagem (TT - 114020). O mesmo processo ocorre para a abordagem seguinte (LVF + DemoIS) e todas as outras apresentadas aqui, excetuando-se a abordagem FortalDemoDS que tem seus resultados apresentados de forma contígua nas tabelas.

Resultados Experimentais – Naive Bayes

		Chess	Libras	Satellite	Spam	Splice	Texture
LVF	NA	19,6	38,6	*8,6	22,8	28	18,4
	ER	0,078411	0,371667	0,2223	0,218996	0,085392	0,221782
	TT	101487,2	193952,2	116295,2	773902,6	482207,6	214855,6
LVF+MVF	NE	3118,6	272,2	3734,4	4164,2	2910,2	5197
	ER	0,06217	0,219802	0,2138	0,140861	0,024309	0,185452
	TE	12532,8	1421,4	15852	40265,2	16338,2	50038,6
	TT	114020	195373,6	132147,2	814167,8	498545,8	264894,2
LVF+DemoIS	NE	2970,8	236	3006,2	3237	2406,6	4008,4
	ER	0,03821	0,15961	0,2116	0,053139	0,002538	0,037774
	TE	44317,4	10828	48090,8	136288,4	55574	160775,4
	TT	145804,6	204780,2	164386	773902,7	537781,6	375631
FortalFS	NA	19	41,6	18,4	28,2	35	17,6
	ER	0,065457	0,37	0,2047	0,1391	0,04953	0,203309
	TT	44266,4	97069,6	102859,8	251332	94981,4	245406,6
FortalFS+MVF	NE	3153,8	266,8	3886,6	4255	3027,2	5221,2
	ER	0,055754	0,223416	0,1992	0,079145	0,014412	0,166735
	TE	11948,8	1348,6	31518,2	47436	21017,6	47164
	TT	56215,2	98418,2	134378	298768	115999	292570,6
FortalFS+DemoIS	NE	2946,8	207,2	3160,8	3555,4	2612,8	4111,2
	ER	*0,020822	0,14132	*0,1953	0,013194	*0,001078	0,032751
	TE	39910,6	12281,8	109376	155300,4	82689,6	162939,6
	TT	84177	109351,4	212235,8	406632,4	177671	408346,2
MVF	NE	3099,8	271,8	3907,2	4302,4	3038,2	5252,4
	ER	0,110712	0,222938	0,1991	0,148796	0,017511	0,190311
	TT	*24507,8	*3234	*61699,4	*107093	*39929	*113561
MVF+LVF	NA	18,4	*36,8	18,8	22	30,4	17,6
	ER	0,071997	0,227351	0,231	0,155709	0,0682	0,178814
	TE	105443,8	131950	102464,4	777243,4	496083,8	223537,4
	TT	129951,6	135184	164163,8	884336,4	536012,8	337098,4
MVF+FortalFS	NA	20	39,8	18,2	25,4	33	19,6
	ER	0,068381	0,222376	0,2004	0,119001	0,022764	0,171271
	TE	34900,4	63200,4	87942	217179,2	91202,4	236975,4
	TT	59408,2	66434,4	149641,4	324272,2	131131,4	350536,4
DemoIS	NE	3053,8	*190,2	3172	3470,2	2597,6	4129,2
	ER	0,107682	0,106412	0,1988	0,065353	0,002848	0,046152
	TT	88288	29578,2	217447	341322,8	119513	365187,6
DemoIS+LVF	NA	*17,8	40,2	16,4	*20,8	*27,6	*15,4
	ER	0,06102	*0,096281	0,1992	0,109165	0,048097	0,04722
	TE	103119,2	102432	76564,6	550814,6	393699,4	157625,4
	TT	191407,2	132010,2	294011,6	892137,4	513212,4	522813
DemoIS+FortalFs	NA	20,4	43,2	17,4	27	31,6	18,4
	ER	0,05083	0,11816	0,1959	0,038337	0,006699	0,039528
	TE	35676,8	43836,4	78946,2	153670,6	72996,2	192532,4
	TT	123964,8	73414,6	296393,2	494993,4	192509,2	557720
FortalDemoDS	NA	19,4	43,8	18,8	28,8	32,2	17,2
	NE	*2483,6	234,4	*2366,4	*1236,8	*344,8	*2769,6
	ER	0,027022	0,116133	0,2085	*0,002887	0,003746	*0,000948
	TT	523423,8	618358,6	875299	2167302	1085381	1199263

Resultados Experimentais – C4.5

		Chess	Libras	Satellite	Spam	Splice	Texture
LVF	NA	*15,6	34,6	*8,6	26,4	23,4	*15
	ER	0,033667	0,313333	0,1797	0,084982	0,122821	0,079309
	TT	104675,8	195022,4	117098,4	777342,2	484175,6	215096,4
LVF+MVF	NE	3128,2	*269	3692,4	4190,6	2834,6	*5176,2
	ER	0,01969	0,221255	0,1798	0,03938	0,044137	0,050187
	TE	13315,6	1244,4	15851,6	45811,2	14343,8	45801,8
	TT	117991,4	196266,8	132950	823153,4	498519,4	260898,2
LVF+DemoIS	NE	2870,2	328,8	*3563,6	3264	2291	5415
	ER	0,001319	0,256563	0,1864	0,011611	0,014245	0,066928
	TE	40907,8	10587,2	53648,6	160172	45191,2	141346,8
	TT	145583,6	205609,6	170747	937514,2	529366,8	356443,2
FortalFS	NA	27	42	17,8	33,6	33,6	19,2
	ER	0,012265	0,309444	0,1465	0,073593	0,055611	0,071309
	TT	62443,2	124722,2	234113,4	746592,4	135287,6	509597,8
FortalFS+MVF	NE	3137,2	270	3887,6	4228,2	3024,6	5235
	ER	0,005539	0,245586	0,1481	0,032728	0,020028	0,048095
	TE	15265,4	1378,8	30310	61357,4	24927,8	53554,8
	TT	77708,6	126101	264423,4	807949,8	160215,4	563152,6
FortalFS+DemoIS	NE	2853	316,6	4204	3372,8	2605,4	5404,2
	ER	0,003093	0,24933	0,1445	0,014655	*0,011047	0,0605
	TE	56014,2	12431	112808,8	204758,4	70201,6	179621,4
	TT	118457,4	137153,2	346922,2	951350,8	205489,2	689219,2
MVF	NE	3108,2	276,4	3915,6	4302,8	3042,4	5253,8
	ER	0,002573	0,217761	0,1406	0,042302	0,022482	*0,047316
	TT	*23862,2	*3282,2	*62238	*98150,2	*39291,8	*125682
MVF+LVF	NA	19,4	35,8	19,4	27,2	*22,2	19
	ER	0,025344	*0,215687	0,1503	0,061087	0,105785	0,06144
	TE	102782,2	136330,8	103406,8	755156,6	498621	212982,6
	TT	126644,4	139613	165644,8	853306,8	537912,8	338664,6
MVF+FortalFS	NA	26,6	43	17,8	27,8	33,4	23,8
	ER	0,009528	0,229323	0,1449	0,045689	0,026425	0,052721
	TE	52814,6	90965,4	161641,4	546346,2	123105	474063,8
	TT	76676,8	94247,6	223879,4	644496,4	162396,8	599745,8
DemoIS	NE	2659,4	304,2	4256,6	3467,6	2591,4	5423,2
	ER	0,002409	0,222531	*0,1359	0,018396	0,015754	0,059164
	TT	86230,8	28359,2	223608,4	389264,2	131485,8	390075,6
DemoIS+LVF	NA	18,6	*34,2	20,6	*26,2	24,2	20,4
	ER	0,015195	0,217572	0,1471	0,024403	0,054743	0,061547
	TE	85284,8	166131,6	116037	532815,8	393575,4	228041,8
	TT	171515,6	194490,8	339645,4	922080	525061,2	618117,4
DemoIS+FortalFS	NA	22,4	46,4	19	28,4	34	23,2
	ER	0,001728	0,234767	0,1446	0,021867	0,019918	0,062121
	TE	35374	103785,4	202533,6	339045,8	96070,6	499711,2
	TT	121604,8	132144,6	426142	728310	227556,4	889786,8
FortalDemoDS	NA	25	37,2	17,4	32,4	28,4	17,6
	NE	*1727,4	325,8	3925,4	*1451,4	*988,8	5481,6
	ER	*0,001098	0,281088	0,1604	*0,008452	0,018278	0,075014
	TT	536433,8	655892,8	889659,4	2476663	1107077	1434004

Resultados Experimentais – 1NN

		Chess	Libras	Satellite	Spam	Splice	Texture
LVF	NA	*18,4	35,8	*14	27,6	*22,4	23
	ER	0,042178	0,128333	0,1397	0,10489	0,290219	0,012545
	TT	103049,2	198096,6	116408	783034	482114,4	213413,4
LVF+MVF	NE	3147	270,8	3799,2	4181,2	2738,4	5224,6
	ER	0,028722	0,044331	0,1293	0,040148	0,195522	0,004557
	TE	11454,8	1491,4	27363,2	49517,2	15192,2	63729,8
	TT	114504	199588	143771,2	832551,2	497306,6	277143,2
LVF+DemoIS	NE	*2949,6	*262,6	4191,8	3233,6	3024,8	4054
	ER	*0,008286	0,045585	0,1255	0,016354	0,242185	0,000749
	TE	78961	10274	141331	293264,4	78508,4	361658,2
	TT	182010,2	208370,6	257739	1076298	560622,8	575071,6
FortalFS	NA	23,4	42,2	18,4	31,2	29,4	21,4
	ER	0,029412	0,142222	0,12	0,098109	0,223887	0,010073
	TT	1263990	120890,4	2390783	7024272	2556456	4513786
FortalFS+MVF	NE	3165,8	274	3900,6	4238	3015	5230,4
	ER	0,019651	0,053897	0,1088	0,039952	0,182837	0,004206
	TE	15215,6	1559,2	32672,2	56597,4	17874,4	62048,6
	TT	1279206	122449,6	2423455	7080869	2574330	4575835
FortalFS+DemoIS	NE	3074,2	316,6	4021	3675,2	3114,4	4090,8
	ER	0,013888	0,074095	0,1158	0,031588	0,204007	0,000585
	TE	95657,8	12377,6	180167	334358,6	115905,8	340214,6
	TT	1359648	133268	2570950	7358631	2672362	4854001
MVF	NE	3112	269,8	3911	4304,2	3036,8	5243,4
	ER	0,078727	0,040745	0,1025	0,048791	0,206994	0,002937
	TT	*23649	*3096,8	*68728,8	*106733,8	*38240,2	*120851,6
MVF+LVF	NA	19	38	17,2	*26,2	25,6	20,6
	ER	0,042618	0,043024	0,114	0,080202	0,227785	0,007288
	TE	102906	136989,6	102099	658326	494964	202234,6
	TT	126555	140086,4	170827,8	765059,8	533204,2	323086,2
MVF+FortalFS	NA	20,2	38,6	21,6	28,8	32,2	20,6
	ER	0,030011	*0,039986	0,1097	0,060277	0,191316	0,006255
	TE	1167484	77754,2	1812316	5525369	2353740	3999890
	TT	1191133	80851	1881044	5632103	2391980	4120742
DemoIS	NE	3186,2	291,8	4261,8	3892,8	3148,6	4139,2
	ER	0,093524	0,050038	*0,0988	0,047349	0,234514	0,000723
	TT	139080	26821,2	382406,6	631664,2	225000,4	635778,8
DemoIS+LVF	NA	19,2	*32,8	21	27	25	20,2
	ER	0,046996	0,05094	0,1091	0,064835	0,232549	0,001643
	TE	106272,6	155671,8	115071,4	568602,4	513579,6	150757,4
	TT	245352,6	182493	497478	1200267	738580	786536,2
DemoIS+FortalFs	NA	22,8	36	20,2	27,2	30	*16,2
	ER	0,030072	0,045951	0,1078	0,047705	0,22581	0,001546
	TE	1209346	84349	2135000	4808630	2460060	2523636
	TT	1209365	111170,2	2517406	5440295	2685060	3159415
FortalDemoDS	NA	20,8	42,2	15,6	27,8	23,2	19,8
	NE	3010	349,2	*2313,6	*1262,4	*358,6	*3741
	ER	0,031421	0,119505	0,1636	*0,004833	*0,083939	*0,000424
	TT	1432173	679355,4	2249636	6097611	2894321	3571738