UNIVERSIDADE DE BRASÍLIA
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
CURSO DE MESTRADO EM INFORMÁTICA

Um Estudo de Limpeza em Base de Dados Desbalanceada e com Sobreposição de Classes

EMERSON LOPES MACHADO

Dissertação submetida à avaliação
como requisito parcial para a obtenção do grau de
Mestre em Informática

Prof. Dr. Marcelo Ladeira
Orientador

Brasília, Distrito Federal

Abril de 2007

CIP - Catalogação na Publicação

Machado, Emerson Lopes

Um Estudo de Limpeza em Base de Dados Desbalanceada e com Sobreposição de Classes / Emerson Lopes Machado. - Brasília: CIC da UnB, 2007.

63p.: il.

Dissertação (mestrado) – Universidade de Brasília. Programa de Mestrado em Informática, Brasília, BR – DF, 2007. Orientador: Ladeira, Marcelo.

1. Base de dados desbalanceada. 2. Mineração de dados. 3. Aprendizagem de Máquina. 5. Representação do conhecimento. 6. Inteligência artificial. I. Ladeira, Marcelo.

Resumo

O objetivo geral desta pesquisa é analisar técnicas para aumentar a acurácia de classificadores construídos a partir de bases de dados desbalanceadas. Uma base de dados é desbalanceada quando possui muito mais casos de uma classe do que das outras, portanto possui *classes raras*. O desbalanceamento também pode ser em uma mesma classe se a distribuição dos valores dos atributos for muito assimétrica, levando à ocorrência de *casos raros*. Algoritmos classificadores são muito sensíveis a estes tipos de desbalanceamentos e tendem a valorizar as classes (ou casos) predominantes e a ignorar as classes (ou casos) de menor freqüência. Modelos gerados para bases de dados com classes raras apresentam baixa acurácia para estas classes, o que é problemático quando elas são classes de interesse (ou quando uma delas é a classe de interesse). Já os casos raros podem ser ignorados pelos algoritmos classificadores, o que é problemático quando tais casos pertencem à classe (ou às classes) de interesse. Uma nova proposição de algoritmo é o *Cluster-based Smote*, que se baseia na combinação dos métodos de *Cluster-based Oversampling* (*oversampling* por replicação de casos guiada por *clusters*) e no SMOTE (*oversampling* por geração de casos sintéticos). O método *Cluster-based Oversampling* visa melhorar a aprendizagem de pequenos disjuntos, geralmente relacionados a casos raros, mas causa *overfitting* do modelo ao conjunto de treinamento. O método SMOTE gera novos casos sintéticos ao invés de replicar casos existentes, mas não enfatiza casos raros. A combinação desses algoritmos, chamada de *Cluster-based Smote*, apresentou resultados melhores do que a aplicação deles em separado em oito das nove bases de dados utilizadas proposta nesta pesquisa. A outra abordagem proposta nesta pesquisa visa a diminuir a sobreposição de classes possivelmente provocada pela aplicação do método SMOTE. Intuitivamente, esta abordagem consiste em guiar a aplicação do SMOTE com a aprendizagem não supervisionada proporcionada pela clusterização. O método implementado sob esta abordagem, denominado de *C-clear*, resultou em melhora significativa em relação ao SMOTE em três das nove bases testadas e empatou nas demais. Foi também proposta uma nova abordagem para limpeza de dados baseada na aprendizagem não supervisionada, a qual foi incorporada ao *C-clear*. Esta limpeza somente surtiu melhora em uma base de dados, sendo este baixo desempenho oriundo possivelmente da escolha não adequada de seus parâmetros de limpeza. A aprendizagem destes parâmetros a partir dos dados ficou como trabalho futuro.

PALAVRAS-CHAVES: Mineração de dados, desbalanceamento de classes, sobreposição de classes, SMOTE, *Cluster-based Oversampling*, *Cluster-based Smote*, *C-clear*.

Abstract

It is intended in this work to research methods that improve the accuracy of classifiers applied to data set with class imbalance (high skew in class distribution causing rare classes) and within-class imbalance (high skew in data within-class distribution causing rare cases). Standard classifier algorithms are strongly affected by these characteristics and their generated model are biased to the majority classes (or cases), in detriment of classes (or cases) underrepresented. Generally, models generated with imbalanced data set suffer from low accuracy for the minority classes, which is a problem when the target class is one of them. Eventually, rare cases are likely of being ignored by inductors, which is a problem when they belong to the interesting class (or classes). A new method is proposed in this work, *Cluster-based Smote*, which combines the methods *Cluster-based Oversampling* (oversampling by replication of positive cases guided by clusters) and SMOTE (Synthetic Minority Oversampling Technique). *Cluster-based Oversampling* addresses small disjuncts, but overfits the model to the training set. The method SMOTE addresses the overfit problem of random oversampling, but does not treat rare cases. The combination of them proposed in this research, named *Cluster-based Smote*, presented better results in eight out of nine datasets, compared to the applying of them all alone. Another approach proposed in this research aims at reducing the class overlap problem possibly caused by applying SMOTE. The main idea is to guide the SMOTE process by non-supervised learning (with clustering techniques). The method implemented under this approach, named *C-clear*, resulted in significant improvement over SMOTE in three out of nine datasets. A cleaning method based in the non-supervised learning was also proposed and has been incorporated in the *C-clear* method. The cleaning method improved the results in only one dataset, probably because of the not so well values chosen as cleaning parameters. The learning of these parameters from the data is left as a future work.

KEYWORDS: Data mining, class imbalance, class overlap, SMOTE, *Cluster-based Oversampling*, *Cluster-based Smote*, *C-clear.*

Navegar é preciso, minerar não é preciso!
Marcelo Ladeira

Agradecimentos

Primeiramente, agradeço aos meus pais pelo incentivo e apoio constante e incondicional em toda trajetória de minha vida. Devo a vocês toda a minha gratidão e amor eternos. Não somente a eles, mas também a todos de minha família, principalmente a meus irmãos, pela crença em meus sonhos, que me fez mais forte para segui-los.

À Clarissa, minha amada, *mi prometida*, agradeço pelo carinho e compreensão aos inúmeros finais de semanas, festas e etc. em que lhe troquei pelo computador. Você será recompensada em breve.

À Susana, minha guru, que me mostrou a porta para o encontro de meus sonhos. Você é uma das principais responsáveis pela realização deste trabalho.

Ao Wagner, por ter me iniciado na pesquisa científica na área de IA e ter me incentivado a prosseguir na carreira acadêmica. Nem consigo me imaginar sem este começo precioso.

Ao meu orientador, Marcelo Ladeira, que foi um pai para mim durante dois preciosos anos de minha vida.

À Professora Carolina, pela paciência e compreensão na demora em receber esta dissertação – *Edward A. Murphy*...

Ao Bruno, do LABIC, pela gentileza de imprimir esta dissertação e entregar à Professora Carolina.

À CAPES, pelo importante apoio financeiro dado a esta pesquisa.

À mulher da minha vida, a qual eu já amava muito antes de conhecer. Esta que, mesmo momentaneamente longe, nunca esteve tão perto de meu coração. Esta que dita o ritmo de meu coração. Esta que me aquece a alma e me faz forte para vencer. *Te quiero muuuuuuuuuuuuuuuuuuuuuucho*!

Clarissa Solino Evelin
Sunshine of my Love

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1 Introdução

Neste capítulo, é apresentada a especificação geral do problema abordado, sua relevância e as áreas de pesquisas relacionadas.

1.1 Definição do Problema

Uma base de dados é dita desbalanceada, no domínio de classificação, quando existem muito menos casos de algumas classes do que de outras [Chawla, Japkowicz & Kotcz, 2004]. Estas classes com pouca representação são chamadas de *classes raras* [Weiss, 2004]. Outro tipo de desbalanceamento que tem despertado o interesse da comunidade é o desbalanceamento dentro de uma classe (*intraclasses*), na qual a distribuição dos valores dos atributos é muito assimétrica, levando à ocorrência de *casos raros* [Han, Wang & Mao, 2005]. A título de ilustração, sejam as classes fraude e não-fraude no domínio de transações de cartões de crédito. Um exemplo de desbalanceamento de classes é o número de fraudes ser muito menor do que o de não-fraudes, e um exemplo de casos raros é a ocorrência de fraudes milionárias. Algoritmos classificadores são muito sensíveis a estes dois tipos de desbalanceamento e tendem a valorizar as classes (casos) predominantes e a ignorar as classes (casos) de menor representação [Phua, Alahakoon & Lee, 2004]. Os classificadores resultantes de dados com classes raras apresentam altas taxas de falsos negativos para as classes raras, o que é problemático quando a classe de interesse é justamente uma das classes raras. Quando existem casos raros, estes casos não são aprendidos, o que é indesejável quando eles pertencem à classe de interesse. A premissa de se estudar estes dois tipos de desbalanceamento vem do fato de que o problema de classes raras freqüentemente acompanha o problema de casos raros [Weiss, 2004].

O problema de classes desbalanceadas é um problema relativamente recente que surgiu quando o aprendizado de máquina evoluiu de seu estado embrionário, puramente científico, para uma tecnologia aplicada, muito usada no mundo de negócios, indústrias e pesquisas científicas. Apesar deste problema já ter sido identificado há mais tempo, ele somente se fez presente no círculo de pesquisas há cerca de 10 anos. Sua importância cresceu quando foi percebido que bases de dados desbalanceadas prejudicavam a acurácia de modelos gerados por algoritmos classificadores [Chawla, Japkowicz & Kotcz, 2004]. O problema da aprendizagem em bases de dados desbalanceadas é tão importante que já foram realizados dois workshops, um em 2000 na conferência AAAI [Japkowicz, 2000] e outro em 2003 na conferência ICML [Chawla, Japkowicz & Kotcz, 2003], e foi lançada uma edição especial da ACM SIGKDD Explorations em 2004 que reuniu diversos trabalhos neste tema [Fayyad, 2004]. Em aplicações práticas, a relação entre a classe de menor e de maior representatividade pode ser de 1:100, 1:1.000, 1:10.000 ou ainda maior. Esta desproporção pode ser decorrente de um problema de amostragem, que poderia ser resolvida fazendo-se uma nova amostragem, mas em certos domínios este desbalanceamento é uma propriedade intrínseca do domínio do problema. Nestes casos, faz-se necessário o uso de técnicas específicas para classificação em bases de dados desbalanceadas.

1.2 Objetivo

O objetivo geral desta dissertação é explorar técnicas para aumentar a acurácia de algoritmos classificadores aplicados a bases de dados desbalanceadas. A literatura apresenta uma ampla gama de soluções propostas para o problema de desbalanceamento de classes e, durante o desenvolvimento desta pesquisa, foi dado foco apenas aos métodos de amostragem. Esta escolha foi motivada pela atenção dada pela comunidade a este tipo de abordagem [Chawla, Japkowicz, Kotcz, 2004], o que, de forma alguma, desmerece as outras abordagens existentes. O objetivo específico é propor novas técnicas de balanceamento e/ou limpeza de dados, baseadas na combinação de características dos algoritmos SMOTE e *Cluster-based Oversampling*.

1.3 Áreas de Pesquisas Relacionadas

A mineração de dados constitui uma área de pesquisa que utiliza técnicas de aprendizagem de máquina aplicadas a banco de dados para a construção de modelos e diversas técnicas da estatística. Ela também é vista como um dos passos do processo de extração de conhecimento em bases de dados (KDD - *Knowledge Discovery in Databases*) [Fayyad, 1997].

A área da aprendizagem de máquina estuda como construir programas de computador que melhoram seu desempenho com a experiência. Neste contexto, e para situações em que o desempenho em alguma tarefa possa ser medido, aprender pode ser definido como se segue: um programa computacional aprende a partir da experiência E em relação a uma classe de tarefas T, com medida de desempenho P, se seu desempenho nas tarefas T (medida por P) melhora com a experiência E [Mitchell, 1997]. Os métodos de aprendizagem de máquina têm sido utilizados em diversas aplicações como veículos autônomos que aprendem a dirigir em vias expressas, reconhecimento da fala, detecção de fraudes em cartões de crédito, estratégias para a construção de jogos, programas de mineração de dados que descobrem regras gerais em grandes bases de dados, etc.

1.4 Organização desta Dissertação

Esta dissertação está organizada da seguinte forma: no Capítulo 2, é apresentado o estado da arte em pré-processamento em base de dados desbalanceados. No Capítulo 3, as abordagens propostas são introduzidas. No Capítulo 4, os resultados obtidos com as abordagens propostas são apresentados e analisados. Por fim, conclusões e trabalho futuro são descritos no Capítulo 5. Alguns temas relevantes a esta dissertação, mas que não constitui o foco de pesquisa, são sucintamente abordados nos apêndices. No Apêndice A e no Apêndice B, são abordados, respectivamente, a fundamentação teórica para os classificadores e os métodos de avaliação utilizados nesta pesquisa. No Apêndice C, são apresentados os métodos de clusterização *k-means*, *Squeezer* e CEBMDC. E finalmente, no Apêndice D, são apresentadas as métricas de distância euclidiana, VDM e HVDM.

Capítulo 2 Pré-processamento em Base de Dados Desbalanceada

Neste capítulo, é apresentada uma revisão da literatura em técnicas de pré-processamento de dados para a mineração de dados em bases desbalanceadas. Este capítulo está organizado da seguinte maneira: na Seção 2.1, é apresentada uma taxonomia para métodos de balanceamento de classes. Na Seção 2.2 e Seção 2.3 são apresentados, respectivamente, os métodos de amostragem e os métodos de limpeza estudados nesta pesquisa; na Seção 2.4, são abordadas, de forma sucinta, outras categorias de métodos.

2.1 Introdução

Chawla, Japkowicz & Kotcz (2004) dividem as soluções propostas para o caso de desbalanceamento de classes em três subáreas: métodos de amostragem (englobam os métodos de classificadores híbridos, também chamados de múltiplos); métodos de aprendizagem de apenas uma classe, por exemplo, a classe de interesse; e métodos de seleção de atributos (do inglês, *feature selection*). Esta separação é apenas uma forma didática de se apresentar os diversos métodos existentes, visto que uma solução específica pode vir a se beneficiar da utilização conjunta de dois ou mais tipos de abordagens. Nesta pesquisa as abordagens a este problema são expostas de forma diferente, de forma a refletir o foco que foi dado à mesma, isto é:

1. Métodos de amostragem de dados, que visam diminuir o desbalanceamento das bases de dados (Seção 2.2);
2. Métodos de limpeza de dados, que atuam de forma a melhor definir o espaço de decisão aprendido pelos classificadores (Seção 2.3);
3. Outros métodos que são importantes para a solução do problema de classes desbalanceadas, mas que somente são apresentados a título de informação e não serão detalhados por fugirem do foco desta pesquisa (Seção 2.4).

2.2 Métodos de Amostragem para Balanceamento de Classes

Os métodos de amostragem visam mudar a distribuição dos dados de treinamento, de modo a aumentar a acurácia de seus modelos. Isto é alcançado com a eliminação de casos da classe majoritária (denominado na literatura como *undersampling*) ou replicação de casos da classe minoritária (denominado *oversampling*). De acordo com Jo e Japkowicz (2004), não há garantia de que a distribuição original dos dados de treinamento seja a mais adequada para a construção de classificadores. Os métodos de amostragem servem para alterar a distribuição dos dados de tal forma que seja possível gerar classificadores melhores para eles. Weiss (2004) subdivide-os em duas vertentes: métodos básicos e métodos avançados de amostragem.

Os *métodos básicos de amostragem* são métodos que não utilizam heurística na eliminação e na replicação de casos, ou seja, são métodos que visam balancear a distribuição de classes de forma aleatória. São eles o *undersampling* aleatório e o *oversampling* aleatório. Segundo Batista, Prati & Monard (2004), diversos autores

concordam que os métodos de amostragem que não utilizam heurísticas podem provocar distúrbios indesejados nos modelos gerados. A simples replicação de casos da classe minoritária pode causar *overfit*[1] do classificador aos dados de treinamento, ao passo que a eliminação aleatória de casos da classe majoritária pode remover casos importantes para o processo de aprendizagem.

Já os *métodos de amostragem avançados* [Weiss, 2004] utilizam heurística na eliminação de casos da classe majoritária e na replicação de casos da classe minoritária. Os seguintes métodos de amostragem avançados são discutidos a seguir: segmentação dos dados [Weiss, 2004]; uma variante do *Condensed Nearest Neighbor Rule* – CNN [Kubat & Matwin, 1997]; *One-sided Selection* – OSS [Kubat & Matwin, 1997]; SMOTE [Chawla *et al.*, 2002]; *Cluster-based Oversampling* [Nickerson, Japkowicz & Millos¸ 2001].

2.2.1 Segmentação dos Dados

O método de segmentação dos dados consiste em reduzir o número de casos da base de dados com a utilização de informação a priori. Suponha que se saiba que a classe de interesse está fortemente ligada a um determinado estado de alguma variável, por exemplo, a uma determinada hora do dia. Pode-se, então, filtrar a base de dados de forma a somente manter os casos que são pertinentes ao problema e, dessa forma, reduzir o desbalanceamento de classes.

2.2.2 Condensed Nearest Neighbor Rule (CNN)

O método CNN (do inglês, *Condensed Nearest Neighbor Rule*), proposto por Hart (1968), consiste em criar um subconjunto consistente C a partir de um conjunto de dados S. O subconjunto $C \subseteq S$ é designado *consistente* quando um classificador 1-NN (classificador k-NN com $k = 1$) (Apêndice A.2) consegue classificar de forma correta todo o conjunto S, utilizando o subconjunto C como conjunto de treinamento. Kubat e Matwin (1997) utilizam uma variante desse método para fazerem *undersampling* apenas dos casos da classe majoritária. A idéia é retirar os exemplos redundantes da classe majoritária. Para isso, o subconjunto C é criado com todos os casos da classe interesse e apenas os casos mais relevantes da classe majoritária do conjunto S. Entende-se por casos relevantes da classe majoritária aqueles que, com o classificador 1-NN, conseguem classificar corretamente todos os casos desta classe. A seguir, apresenta-se o algoritmo desta variação do CNN proposto por Kubat e Matwin:

Algoritmo 1 O método CNN modificado por Kubat e Matwin faz *undersampling* da classe majoritária. A idéia é construir um subconjunto C que seja consistente, a partir do conjunto de entrada S.

Descrição das variáveis:
S: conjunto de dados de entrada;
$C \subseteq S$: subconjunto consistente de S retornado pelo algoritmo CNN.

CNN
Entrada: S

---

[1] Um classificador é dito *overfitted* quando ele é muito específico para o conjunto de treinamento. Modelos muito específicos para o conjunto de treinamento possuem acurácia elevada para os casos conhecidos, mas não é geral o suficiente para a predição de novos casos.

Saída: *C*

```
1. Inicie C com os casos da classe de interesse e os retire de S;
2. Retire um caso da classe majoritária de forma aleatória de S e
   insira em C;
3. Para todo caso x_i da classe majoritária de S faça {
4.   Seja c_i' a classe retornada pelo k-NN com C e k = 1;
5.   Se c_i' diferir da classe de x_i, retire-o de C e o insira em S;
6.  }
7. Retornar C.
```

Observa-se que o subconjunto *C* criado com a variante do CNN de Kubat e Matwin não é necessariamente o menor subconjunto consistente do conjunto *S*. De acordo com os autores, "é satisfatório se o conjunto de casos negativos (da classe majoritária) for reduzido de forma razoável" [Kubat & Matwin, 1997].

2.2.3 One-sided Selection (OSS)

Basicamente, o método *OSS* (do inglês, *One-sided Selection*), proposto por Kubat e Matwin (1997), separa o conjunto original de dados *S* em dois subconjuntos: o subconjunto $C \subseteq S$, que agrupa os casos da classe de interesse, e o subconjunto $O \subseteq T$, que agrupa o restante dos casos. O subconjunto *O* é reduzido (*undersampled*) enquanto o conjunto *C* é mantido intacto. Este método utiliza uma variante do CNN para eliminar casos do subconjunto *O* que estão muito distantes do limiar de decisão (vide Seção 2.3) e, logo após, aplica o método *Tomek links* (Seção 2.3.1) para retirar os casos que são ruídos ou limítrofes. De acordo com esses autores, exemplos limítrofes são considerados "instáveis", pois uma pequena quantidade de ruído pode fazê-los cair no lado errado do limiar de decisão (Seção 2.3).

2.2.4 SMOTE

As simples técnicas de *oversampling* são amplamente atacadas pela comunidade científica, pois muitas delas apenas replicam casos positivos existentes. Meramente replicar casos já existentes da classe minoritária, realmente aumenta o víeis do classificador para esta classe. Entretanto, ocorre o efeito indesejado de modelos *overfitted*, ou seja, modelos muito específicos para estes casos replicados, prejudicando, dessa forma, seu poder de generalização para a classe de interesse. Chawla *et al*. (2002) interpretam este problema em termos de regiões de decisão no espaço de atributos. Geralmente, classes raras são acompanhadas de casos raros e estes casos raros são circundados de casos negativos. O simples ato de replicar um caso positivo propicia que classificadores reconheçam esta região, mas esta região será tão pequena que não conseguirá classificar corretamente novos casos da classe de interesse que venham a cair nas vizinhanças desta região.

Diante desta problemática do *oversampling*, Chawla *et al*. (2002) desenvolveram um método diferente de se fazer *oversampling* da classe minoritária, que consiste na geração de *casos sintéticos* (casos artificiais) para a classe de interesse a partir dos casos já existentes. Estes novos casos serão gerados na vizinhança de cada caso da classe minoritária, de forma a fazer crescer a região de decisão e, assim, aumentar o poder de generalização dos classificadores gerados para estes dados. Esses autores chamam este novo método de SMOTE (do inglês, *Synthetic Minority Oversampling Technique*).

Visualmente, no espaço amostral do conjunto de dados, estes novos casos sintéticos serão interpolados aleatoriamente ao longo do segmento de reta que liga cada caso da classe minoritária a um de seus *k* vizinhos mais próximos, escolhidos de forma aleatória. A seguir, o Algoritmo 2 apresenta o pseudocódigo do método SMOTE.

Algoritmo 2 O algoritmo SMOTE faz *oversampling* da classe minoritária com a criação de novos casos a partir dos casos já existentes.

Descrição das variáveis:
S: conjunto de dados de entrada;
k: número de vizinhos mais próximos considerados;
qtd: número de novos casos desejado para cada caso da classe minoritária;

SMOTE
Entrada: *S*, *qtd*, *k*
Saída: *S* balanceado

```
1. Para cada caso i pertencente à classe minoritária faça {
2.   Calcule os k vizinhos mais próximos do caso i;
3.   Faça qtd vezes {
4.     Para cada atributo contínuo de i faça {
5.       Crie um valor entre i e um dos k vizinhos;
6.     }
7.   }
8. }
```

2.2.5 Cluster-based Oversampling

Nenhum dos métodos apresentados até então enfocam o problema de desbalanceamento interno, que freqüentemente acompanha o problema de classes desbalanceadas. Um conjunto de dados se apresenta internamente desbalanceado quando ocorrem casos raros em comparação com o restante dos dados. De acordo com Weiss (2004), casos raros não são facilmente identificáveis, mas métodos de aprendizagem não supervisionada, como a clusterização, ajudam na identificação desses casos, sendo que eles podem se apresentar como pequenos *clusters*[2] dentro de uma classe. Em classificadores, casos raros potencialmente se apresentam como pequenos disjuntos, que são conceitos aprendidos que cobrem poucos casos do conjunto de treinamento. Em outras palavras, casos raros ocorrem quando alguns subconjuntos de uma classe são muito menores do que os outros [Han, Wang & Mao, 2005]. Ocorre que certos algoritmos classificadores, como o C4.5 *Backpropagation*, tendem a ignorar tais casos por eles não terem representatividade no conjunto de treinamento [Japkowicz, 2003], tornando-se problemático quando casos raros ocorrem na classe minoritária, e esta é a classe interesse.

Nickerson *et al*. (2001) propõem uma abordagem para minimizar este problema de casos raros, juntamente com a minimização do problema de classes raras. Eles partem do pressuposto de que um conjunto de dados retém algumas características originais do problema em questão, e usam estas características para fazer *oversampling* não somente da classe minoritária, mas também dos casos raros. A idéia deles é agrupar os dados de treinamento em *clusters* e, então, balancear a distribuição de seus casos. Este método é chamado *Cluster-based Oversampling*. Primeiramente, o método utiliza a técnica PDDP

---

[2] Aglomerações de dados que compartilham semelhanças entre si.

(do inglês, *Principal Direction Divisive Partitioning*) [Boley, 1998], que é uma técnica não-supervisionada de clusterização para formar os *clusters* de cada classe, e depois faz *oversampling* de cada um destes *clusters*. Este método é aplicável a bases de dados multiclasses. O pseudocódigo de *Cluster-based Oversampling* é apresentado no Algoritmo 3, a seguir.

Algoritmo 3 O algoritmo *Cluster-based Oversampling* faz *oversampling* de todos os *clusters* de todas as classes do conjunto de entrada. Primeiramente, os *clusters* da classe majoritária são balanceados, isto é, cada *cluster* tem seus casos replicados até que se atinja o tamanho do maior *cluster*. Em seguida, as outras classes são sobreamostradas (*oversampled*) até atingir o tamanho da classe majoritária, seguindo o mesmo princípio utilizado para a classe majoritária.

Descrição das variáveis:
S: conjunto de dados de entrada;
PDDP: método de clusterização de Boley (1998);

Cluster-based Oversampling
Entrada: *S*
Saída: *S* balanceado

```
 1. Construa os clusters da classe negativa de S com PDDP;
 2. max ← tamanho do maior cluster;
 3. Para cada cluster C_i faça {
 4.   Replique os casos de C_i até que o tamanho de C_i seja igual a max;
 5. }
 6. numCluster  ← quantidade de clusters classe negativa;
 7. max ← tamanho da classe negativa dividido por numCluster;
 8. Construa os clusters da classe positiva de S com PDDP;
 9. Para cada cluster C_i faça {
10.   Replique os casos de C_i até que o tamanho de C_i seja igual a max;
11. }
12. Retorna S
```

Notar que o algoritmo proposto pelos autores finaliza com o conjunto desbalanceado de entrada totalmente equilibrado, ou seja, com todas as classes e seus *clusters* possuindo a mesma quantidade de casos. Jo & Japkowicz (2004) propõem que os *clusters* utilizados no método *Cluster-based Oversampling* sejam computados com o método de clusterização *k-means* (Apêndice C).

2.3 Métodos de Limpeza de Dados

Casos reais podem apresentar ruídos nas suas variáveis (atributos e classe). Estes ruídos são, geralmente, provocados por erros aleatórios (incerteza na medição, por exemplo) ou por problemas na coleta dos dados (erros de digitação, por exemplo). De acordo com Kubat & Matwin (1997), os casos de uma base de dados podem ser divididos em quatro categorias: casos ruidosos quanto à classe (*class-label noise*), limítrofes (*borderlines*), redundantes e seguros. Essas quatro categorias são relativas ao espaço de decisão, no qual cada caso é visto como um ponto no espaço $\Re^n$, onde *n* é a quantidade de atributos da base de dados. Visto desta forma, o objetivo do classificador é aprender a delimitar as regiões de casos semelhantes, separando-as com superfícies de decisão. Um caso é considerado ruidoso quanto à classe quando ele se situa no lado errado do limiar de decisão, ou seja, ele diverge da classe de seus vizinhos mais próximos. Casos limítrofes são aqueles que se situam perto da borda de decisão. Os casos limítrofes não são confiáveis, pois mesmo uma quantidade pequena de ruído em seus atributos pode jogá-los para o lado errado de uma superfície de decisão. Casos redundantes são aqueles que

podem ser representados por seus vizinhos sem perda da expressividade do classificador resultante. Casos seguros são aqueles que devem ser mantidos para a geração do classificador. A seguir, a Figura 2.1 apresenta visualmente estas categorias. Os dados originais (Figura 2.1a) são rotulados (Figura 2.1b) de acordo com as categorias feitas por Kubat & Matwin (1997). Os rótulos na Figura 2.1b são: 1 - casos ruidosos quanto à classe; 2 - casos limítrofes; 3 - casos redundantes. Os casos não rotulados são os casos considerados seguros para a classificação. Na Figura 2.1c, são apresentados os dados limpos com a retirada dos casos rotulados de 1 a 3.

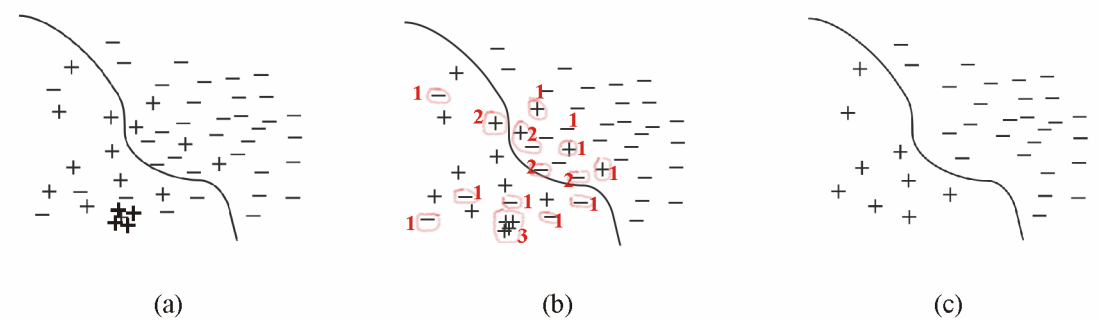

Figura 2.1: Limpeza de Dados.

A presença de dados ruidosos quanto à classe, limítrofes e redundantes pode vir a degradar o desempenho dos classificadores gerados para as bases que os contém. As técnicas de limpeza de dados existentes na literatura visam à eliminação destes tipos de dados. As seções seguintes apresentam algumas destas técnicas de limpeza de dados. Para melhor clareza, foi adotado o termo ruído, ou caso ruidoso, para casos que apresentam ruído quanto à classe.

2.3.1 Tomek links

Sejam x e y dois casos e $d(x,y)$ a distância entes eles no espaço dos seus atributos (variáveis que compõem a instância de dados). O par (x,y) é chamado de *Tomek link* [Tomek, 1976] se, e somente se, x e y pertencerem a classes diferentes, e não existir outro caso z, tal que $d(x,z) < d(x,y)$ e $d(y,z) < d(x,y)$. Intuitivamente, em um *Tomek link*, ou um dos casos é ruído, ou os dois são casos limítrofes. Os pares *Tomek links* podem ser utilizados tanto para se fazer *undersampling* da classe majoritária (remover ruído) quanto para se fazer limpeza em torno do limiar de decisão. No caso de *undersampling*, apenas o caso da classe majoritária de um *Tomek link* é eliminado. No caso de limpeza de dados, ambos são excluídos. O Algoritmo 4, a seguir, apresenta o pseudocódigo de *Tomek links*.

Algoritmo 4 O algoritmo Tomek links pode ser utilizado para se fazer undersampling da classe majoritária ou fazer limpeza em torno do limiar de decisão.

Descrição das variáveis:
S: conjunto de dados de entrada;
tipo: informa a forma que *Tomek links* irá operar. Pode ser "limpeza" ou "redução";

Tomek links
Entrada: *S*, *tipo*
Saída: *S* limpo ou reduzido (*undersampled*)

```
1. Para cada caso x_i de S faça {
2.   Para cada caso x_m de S faça {
```

```
 3.     Se as classes de x_i e x_m divergirem, x_i for o vizinho mais
    próximo de x_m e x_m for o vizinho mais próximo de x_i {
 4.       Se tipo = "redução" {
 5.         Se x_i for da classe majoritária {
 6.           Retire x_i de S;
 7.         } Senão {
 8.           Retire x_m de S;
 9.         }
10.       }
11.       Se tipo = "limpeza" {
12.         Retire x_i e x_m de S;
13.       }
14.     }
15.   }
16. }
17. Retornar S.
```

2.3.2 Edited Nearest Neighbor Rule (ENN)

O método ENN (do inglês, *Edited Nearest Neighbor Rule*) faz *undersampling* dos dados de forma contrária à maneira em que é feito o *undersampling* no método CNN. O método ENN faz *undersampling* do conjunto de entrada *S*, retirando todo caso $E_i \in S$ cuja classe divergir da classe predita pelos seus *k* vizinhos mais próximos (geralmente com $k = 3$). Se nem todos os *k* vizinhos são da mesma classe, a determinação da classe predita pode ser feita pela classe mais freqüente ou por uma ponderação das classes em função da importância relativa de cada uma delas. De acordo com Wilson e Martinez (2000), esse processo remove tanto os casos ruidosos quanto os casos limítrofes, de forma a propiciar um limiar de decisão mais suave. Assim, os dados são reduzidos com a eliminação de casos ruidosos ao mesmo tempo em que a vizinhança da classe de interesse é limpa. A seguir, o Algoritmo 5 apresenta o método ENN.

Algoritmo 5 O método ENN faz *undersampling* do conjunto de entrada *S*. Um caso x_i do conjunto de entrada *S* é eliminado se e somente se a classe de x_i divergir da classe predita pelos seus *k* vizinhos mais próximos, utilizando o classificador *k*-NN (geralmente, $k = 3$). Este método não é otimizado para bases de dados desbalanceadas.

Descrição das variáveis:
S : conjunto de dados de entrada;
k: número de vizinhos mais próximos considerados;

ENN
Entrada: *S*, *k* (geralmente, $k = 3$)
Saída: *S* reduzido

```
1. Para cada x_i pertencente a S faça {
2.   Seja c_i' a classe retornada pelo k-NN com S;
3.   Se c_i' diferir da classe de x_i, retire-o de S;
4. }
5. Retorna S.
```

2.3.3 Neighborhood Cleaning Rule (NCL)

De acordo com Wilson e Martinez (2000), o principal problema do OSS (*One-sided Selection*) é que o método CNN (*Condensed Nearest Neighbor Rule*) nele utilizado é

extremamente sensível a ruídos. Esta sensibilidade se dá porque casos ruidosos geralmente são classificados de forma errônea pelo 1-NN e, assim, são adicionados ao conjunto consistente *C* retornado pelo algoritmo (ver passos 3, 4 e 5 do Algoritmo 1). Mesmo removendo os casos ruidosos com *Tomek links*, os resultados com OSS não são satisfatórios possivelmente por causa do CNN [Laurikkala, 2001].

Laurikkala (2001) apresenta um método multiclasse que enfatiza mais a limpeza de dados do que a sua redução. O autor justifica esta posição pelo fato de que a qualidade de classificadores não reside necessariamente só na quantidade de casos que cada classe do conjunto de treinamento contém, pois a ocorrência de dados de má qualidade afeta de forma negativa o desempenho de classificadores. O método NCL, em seu primeiro passo, utiliza o método ENN de Wilson (1972) *apud* [Wilson & Martinez, 2000] para fazer *undersampling* dos casos que não fazem parte da classe de interesse. A seguir, será sumariamente apresentado o método ENN e, logo após, o método NCL.

O método NCL de Laurikkala utiliza a idéia do método ENN (*Edited Nearest Neighbor Rule*) para encontrar os dados ruidosos que não pertencem à classe de interesse e, logo em seguida, faz limpeza do limiar da vizinhança da classe de interesse. Primeiramente, o conjunto de entrada *S* é organizado em dois subconjuntos: o subconjunto $I \subseteq S$, contendo os casos da classe de interesse, e o subconjunto $M \subseteq S$, contendo o restante dos dados. Para identificar os casos ruidosos de *M*, utilizamos o método ENN para formar o subconjunto $R \subseteq M$. Então, é feita a limpeza da vizinhança de *I* (casos da classe de interesse) da seguinte forma: a partir de cada caso de *I*, encontrar seus três vizinhos mais próximos que pertencem a *M* (casos que não pertencem à classe de interesse) e movê-los para *L* (subconjunto de *S* com casos limítrofes). A fim de se evitar uma excessiva redução de casos de classes com baixa freqüência, somente serão considerados os casos que pertencerem às classes com freqüências maiores ou iguais à metade da freqüência da classe de interesse na construção do subconjunto *L*. A métrica HVDM (do inglês, *Heterogeneous Value Difference Metric*) (Apêndice D) é utilizada para o cálculo da distância entre casos (linhas 2 e 6 do Algoritmo 6).

Algoritmo 6 O método NCL de Laurikkala (2001) faz *undersampling* com foco na limpeza de dados. Primeiramente, são selecionados os casos ruidosos e, logo depois, são selecionados os casos limítrofes que não forem da classe de interesse. Do total dos casos do conjunto de dados de entrada, são eliminados todos os casos ruidosos e limítrofes que forem de classes cujo tamanho seja maior ou igual à metade do tamanho da classe de interesse.

Descrição das variáveis:
S: conjunto de dados de entrada;
k: número de vizinhos mais próximos considerados;
$I \subseteq S$: subconjunto de *S* que contêm todos os casos que pertencem à classe de interesse;
$R \subseteq M$: subconjunto de *S* utilizado para agrupar os casos ruidosos de *M*, iniciado com vazio;
$L \subseteq M$: subconjunto de *S* utilizado para agrupar os casos limítrofes de *M*, iniciado com vazio;

NCL
Entrada: *S*, *k* (geralmente, *k* = 3)
Saída: *C*

```
1. Para cada caso x_i de S que não seja da classe de interesse faça {
2.   Seja c_i' a classe retornada pelo k-NN com S;
3.   Se c_i' diferir da classe de x_i, insira-o em R;
4. }
5. Para cada caso x_i de S da classe de interesse faça {
6.   Seja D o conjunto dos k vizinhos mais próximos de x_i;
7.   Seja c_i' a classe mais freqüente em D;
```

```
 8.   Se c_i' divergir da classe de x_i {
 9.     Para cada caso x_d de D faça {
10.       Seja c_d a classe do caso x_d;
11.       Se o tamanho da classe c_d ≥ metade do tamanho de I {
12.         Insira x_d em L;
13.       }
14.     }
15.   }
16. Remova de S os casos contidos em R e L;
17. Retorna S.
```

2.3.4 SMOTE + Tomek links

Esta é uma idéia proposta por Batista, Prati & Monard (2004) que tem por objetivo fazer primeiramente *oversampling* com SMOTE dos casos positivos (casos da classe de interesse) e, logo após, fazer limpeza de dados. A motivação para limpeza de dados advém do fato de que, geralmente, os casos da classe positiva não estão bem agrupados e pode haver *sobreposição entre as classes*, isto é, podem existir casos negativos (positivos) que invadem a região de decisão de *clusters* de casos positivos (negativos). A aplicação do SMOTE, como método de *oversampling* da classe minoritária, fará crescer a região de decisão dos casos positivos, que é desejado, mas poderá gerar, ou mesmo aumentar, a ocorrência de casos ruidosos. A aplicação de *Tomek links* (Seção 2.2.1) logo após a aplicação do SMOTE visa remover esses casos ruidosos e, portanto, melhorar os agrupamentos de cada classe. Para este método de limpeza, o método Tomek *links* é aplicado tanto para os casos positivos quanto para os negativos. Os casos rotulados com o número 1 na Figura 2.1(a) são exemplos de casos removidos pela aplicação de Tomek *links*.

2.3.5 SMOTE + ENN

Batista, Prati & Monard (2004) também propõem a combinação de SMOTE + ENN. O funcionamento deste método é análogo ao SMOTE + Tomek *links*, sendo que a única diferença é que ENN (Seção 2.3.2) promove uma limpeza mais profunda. Assim como na abordagem SMOTE + *Tomek links*, a limpeza é aplicada aos casos de ambas as classes. O método ENN é utilizado com $k = 3$.

2.4 Outras Abordagens

Durante o início desta pesquisa, foram estudados vários métodos da literatura antes de se focar nos métodos de amostragem. Esta seção apresenta alguns destes métodos estudados. Cabe lembrar que estes métodos são aqui apresentados apenas a título de informação.

Oliveira & Neto (2004), propuseram um esquema de composição de classificadores, denominado *ExperText*, e aplicam-no no problema de classificação de textos, ambiente em que os dados são altamente desbalanceados. A idéia básica é utilizar amostragens distintas do conjunto de treinamento e classificá-las com diferentes técnicas de classificação de forma distribuída e incremental. Os modelos produzidos por cada um destes conjuntos são, então, combinados em um único modelo de classificação. De

acordo com Oliveira e Neto, o classificador combinado produz resultados superiores quando comparados a classificadores individuais.

Phua, Alahakoon & Lee (2004) propõem um esquema de classificador híbrido que mistura as qualidades do *Stacking* e *Bagging*, e o chamam de *Stacking-Bagging*. Os autores utilizam os algoritmos *Backpropagation* [Rumelhart & McClelland, 1986], *Naive Bayes* [Langley, Iba & Thompson, 1992] e C4.5 [Quinlan, 1993] em partições de dados derivados de *oversampling* com reposição. Eles utilizam um meta-classificador para escolher os melhores modelos base (*stacking*) e, então, combinam estes modelos base com votação (*bagging*).

Uma outra abordagem para minimizar os efeitos do desbalanceamento de classes é aprender apenas a classe de interesse [Weiss, 2004], sendo o classificador construído somente com os exemplos da classe minoritária. A idéia é utilizar a aprendizagem baseada em reconhecimento. Assim, para classificar um novo caso, basta calcular o nível de similaridade entre ele e a classe alvo, e depois compará-lo com o limiar de similaridade adotado [Chawla, Japkowicz & Kotcz, 2004]. De acordo com os autores, dentre os métodos que aprendem apenas a classe de interesse, um que tem apresentado bom desempenho é o SVM (do inglês, *Support Vector Machine*) [Raskutti & Kowalczyk, 2003] *apud* [Weiss, 2004].

Outra abordagem é a de seleção de atributos. A seleção de atributos é uma etapa que pode ter grande impacto na acurácia do classificador [Guyon & Elisseeff, 2003] *apud* [Chawla et al., 2004]. O processo de aprendizagem em base de dados com muitos atributos necessita de um alto poder computacional e, geralmente, resulta em baixa acurácia. De acordo com Chawla, Japkowicz & Kotcz, 2004, a existência de muitos atributos favorece a ocorrência de desbalanceamento de classes.

Capítulo 3 Abordagens Propostas

Este capítulo apresenta as duas abordagens propostas para o problema de base de dados desbalanceada. Na Seção 3.1 é apresentado o *Cluster-based Smote* e, na Seção 3.2, é apresentado o *C-clear*.

3.1 Cluster-based Smote

Após a revisão do estado da arte do problema de bases de dados desbalanceadas, observou-se uma maior atividade de pesquisa vinculada ao desbalanceamento de classes e a não existência de métodos específicos que tratam de casos raros ao mesmo tempo em que evitam a simples replicação de casos existentes. Geralmente, os pequenos disjuntos acabam por serem negligenciadas por algoritmos classificadores e a mera replicação de casos existentes pode causar *overfitting* [Daskalaki, Kopanas & Avouris, 2006].

Embora o método SMOTE seja utilizado para criar novos casos sintéticos, o seu foco não é o tratamento do problema de casos raros. Uma alteração potencialmente promissora desse método seria povoar as regiões menos densas com maior intensidade do que as regiões mais densas. Desta forma, seria criado um maior número de casos sintéticos para os casos de pequenos disjuntos, reduzindo a possibilidade de ocorrência de *overfitting*. Já o método *Cluster-based Oversampling* permite descobrir e povoar pequenos disjuntos, mas se limita a apenas replicar casos já existentes que, como anteriormente citado, pode causar *overfitting*.

Nesta seção, é proposta uma nova abordagem de *oversampling* que combina os métodos SMOTE de Chawla *et al*. (2002) e *Cluster-based Oversampling* de Nickerson *et al*. (2001). A idéia consiste em alterar o método *Cluster-based Oversampling* de forma que ele gere casos sintéticos com o SMOTE, ao invés de simplesmente replicá-los. Com esta nova abordagem, potencialmente, pode-se obter melhores resultados no balanceamento de base de dados do que apenas utilizar o SMOTE ou o *Cluster-based Oversampling de per si*. Este novo método foi nomeado *Cluster-based Smote*.

Diferentemente do *Cluster-based Oversampling*, o *Cluster-based Smote* foi implementado com a opção de se escolher a distribuição de classe resultante. Esta opção de projeto foi motivada pelo fato de que a distribuição de classe ótima dos dados de treinamento deve ser determinada de forma empírica [Phua *et al*., 1999 *apud* Phua, Alahakoon & Lee, 2004][Jo & Japkowicz, 2004]. Para melhor compreensão do funcionamento do *Cluster-based Smote*, esta opção não aparece no Algoritmo 7.

Por opção de projeto, o *Cluster-based Smote* funciona com qualquer método de clusterização desejado, pois ele utiliza um vetor de inteiros como parâmetro de entrada que indica a qual *cluster* cada caso pertence, ao invés de chamar um método de clusterização, como indicado no Algoritmo 7, a seguir.

O método SMOTE somente é válido para atributos contínuos. Para o *Cluster-based Smote* funcionar com bases que possuem atributos mistos, o SMOTE foi alterado para utilizar a métrica HVDM (Apêndice D) no cálculo das distâncias entre casos. Na geração de um caso sintético, os valores contínuos são interpolados (Seção 2.2.4) e os

valores categóricos são replicados. Originalmente, o método SMOTE é aplicável a todos os casos da classe positiva. Para ele se integrar no *Cluster-based Smote*, o SMOTE foi implementado para gerar casos sintéticos a partir dos casos indicados em um vetor de inteiros. Maiores detalhes de implementação na Seção 3.3.

Algoritmo 7 O algoritmo *Cluster-based Smote* faz *oversampling* de todos os *clusters* das classes do conjunto de entrada. Primeiramente, os *clusters* da classe majoritária são balanceados, isto é, são criados casos sintéticos para seus *clusters* até que todos tenham o mesmo tamanho. Em seguida, cada classe restante é igualmente balanceada até que se atinja o tamanho da classe majoritária.

Descrição das variáveis:
S: conjunto de dados de entrada;

Cluster-based Smote
Entrada: *S*
Saída: *S* balanceado

```
 1. Construa os clusters da classe negativa de S;
 2. max ← tamanho do maior cluster;
 3. Para cada cluster C_i faça {
 4.   Aplique SMOTE a C_i até que o tamanho de C_i seja igual a max;
 5. }
 6. numCluster ← quantidade de clusters classe negativa;
 7. max ← tamanho da classe negativa dividido por numCluster;
 8. Construa os clusters da classe positiva de S;
 9. Para cada cluster C_i faça {
10.   Aplique SMOTE a C_i até que o tamanho de C_i seja igual a max;
11. }
12. Retorna S.
```

3.2 C-clear

Prati, Batista & Monard (2004) apresentaram um estudo sobre o quanto que a sobreposição de classes pode afetar o desempenho de classificadores e Batista, Prati & Monard (2004) propuseram a aplicação de métodos de limpeza logo após a aplicação do método SMOTE (Seções 2.3.4 e 2.3.5). A motivação dessas abordagens consiste em remover casos gerados pelo SMOTE que possivelmente sejam ruídos (Seção 2.3), os quais degradam o desempenho de classificadores.

A nova abordagem proposta nesta seção visa guiar a aplicação do método SMOTE, de modo a diminuir a quantidade de ruído por ele gerado e, logo após, fazer limpeza de dados. Intuitivamente, esta abordagem consiste em aplicar o método SMOTE aos casos positivos que residem em regiões onde a freqüência da classe positiva seja maior do que certo limiar e, logo após, remover os casos negativos (positivos) das regiões onde a freqüência da classe positiva (negativa) seja maior do que certo limiar. Tais regiões são obtidas com aprendizagem não supervisionada. O método desenvolvido sob esta abordagem foi nomeado *C-clear*. O funcionamento dessa abordagem se baseia em duas premissas: primeiro, os dados tenderem a se agrupar em *clusters*, ao invés de se distribuirem uniformemente no espaço amostral e, segundo, a variável classe ser um atributo relevante a este agrupamento.

Sumariamente, o *C-clear* opera em três passos (Figura 3.1). No primeiro, os dados de entrada são agrupados (Figura 3.1a e Figura 3.1b). No segundo (Figura 3.1c), os *clusters* são rotulados como positivo e negativo, de acordo com o limiar adotado. No

terceiro (Figura 3.1d), é aplicado SMOTE aos *clusters* positivos e, opcionalmente, pode ser feita uma limpeza de dados com a remoção dos casos ruidosos (*i.e.*, casos com classes dissonantes do rótulo do *cluster* ao qual pertencem). O parâmetro **limiarSmote** indica qual a freqüência mínima de casos positivos para que um *cluster* seja considerado positivo. O **limiarSmote** varia no intervalo [0,1], sendo 0 para considerar todos os *clusters* como positivo e, portanto, aplicar SMOTE em todos os casos positivos, e 1 para não aplicar SMOTE. A intensidade da limpeza depende do parâmetro **limiarLimpezaPositivo** (**limiarLimpezaNegativo**), o qual indica qual a freqüência positiva (negativa) mínima admitida para que um *cluster* seja considerado positivo (negativo). Os parâmetros utilizados no *C-clear* variam no intervalo real [0,1], embora na prática poucos valores sejam de interesse. De fato, o número máximo de valores de interesse é igual à quantidade de *clusters* que possuem freqüências de classes positivas distintas. O menor valor de interesse para os parâmetros **limiarSmote** e **limiarLimpezaPositivo** (**limiarLimpezaNegativo**), depois do 0, é a menor freqüência relativa de casos positivos (negativos) dos *clusters.* Por analogia, o maior valor de interesse, antes do 1, para os parâmetros **limiarSmote** e **limiarLimpezaPositivo** (**limiarLimpezaNegativo**) é a maior freqüência relativa de casos positivos (negativos) dos *clusters*. O pseudocódigo do *C-clear* apresentado no Algoritmo 8, a seguir.

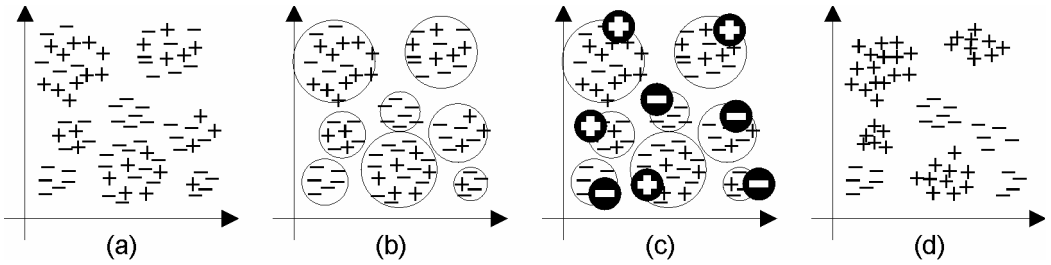

Figura 3.1: Funcionamento do método *C-clear*.

Algoritmo 8 O algoritmo *C-clear* aplica SMOTE somente aos *clusters* que possuem freqüência positiva maior que certo limiar e, opcionalmente, faz limpeza de dados.

Descrição das variáveis:
S: conjunto de dados de entrada;
limiarSmote: freqüência positiva mínima de um *cluster* para *oversampling*;
limiarLimpezaPositivo: freqüência positiva mínima de um *cluster* para limpeza da classe positiva;
limiarLimpezaNegativo: freqüência negativa mínima de um *cluster* para limpeza da classe negativa;

Entrada: *S*, *limiarSmote*, *limiarLimpezaPositivo*, *limiarLimpezaNegativo*
Saída: *S* limpo e balanceado

```
 1. Construa os clusters de S;
 2. Para cada cluster Ci faça {
 3.   Se a freqüência positiva de Ci ≥ limiarSmote {
 4.     Aplique SMOTE a Ci;
 5.   }
 6. Se a limpeza for desejada {
 7.   Construa os clusters de S;
 8.   Para cada cluster Ci faça {
 9.     Se a freqüência positiva de Ci ≥ limiarLimpezaPositivo {
10.       Remova todos os casos negativos de Ci;
11.     }
12.     Se a freqüência negativa de Ci ≥ limiarLimpezaNegativo {
13.       Remova todos os casos positivos de Ci;
14.     }
15.   }
```

```
16. }
17. Retorne S;
```

Da mesma forma que o método *Cluster-based Smote*, o *C-clear* foi implementado de forma a funcionar com qualquer método de clusterização desejado. Maiores detalhes de implementação na Seção 3.3, a seguir.

3.3 Implementação dos Métodos Propostos

Os métodos propostos *Cluster-based Smote* e *C-clear* e os métodos *Oversampling* aleatório, *Cluster-based Oversampling* e SMOTE foram implementados em Java 5 no módulo de pré-processamento do UnBMiner [Ladeira *et al.*, 2005]. O UnBMiner é composto de uma plataforma e API abertas e foi desenvolvido para facilitar a implementação, avaliação de técnicas de mineração de dados e construção de modelos. Esse software suporta parcialmente o modelo de referência CRISP-DM[3] [SPSS, 1999], abrangendo parte da fase de preparação dos dados (módulo pré-processamento do UnBMiner), fase de modelagem (criação de modelos baseados em *Naive* Bayes, CNM, árvore de decisão – algoritmos ID3 e C4.5 – e rede MLP *backpropagation*) e fase de avaliação (módulo de avaliação do UnBMiner).

[3] Do inglês, *CRoss Industry Standard Process for Data Mining*.

Capítulo 4 Avaliação

Este capítulo apresenta os resultados obtidos com a aplicação dos métodos propostos *Cluster-based Smote* e *C-clear* a bases de dados disponíveis na literatura e geralmente utilizadas para avaliação de desempenho de algoritmos de classificação em mineração de dados. Foram utilizadas bases de dados disponíveis no repositório da Universidade da Califórnia em Irvine (UCI). A Seção 4.1 apresenta a metodologia utilizada para a avaliação dos algoritmos propostos, a Seção 4.2 apresenta a análise dos resultados obtidos e a Seção 4.3 apresenta a conclusão das avaliações realizadas neste capítulo.

4.1 Metodologia de Avaliação

Os métodos propostos *Cluster-based Smote* e *C-clear* foram avaliados e comparados com o *oversampling* aleatório, *Cluster-based Oversampling* e SMOTE. Sumariamente, o processo de avaliação consiste em três passos:

1) A base de dados de entrada é divida em dois conjuntos, um de treinamento e outro de avaliação;
2) O conjunto de treinamento é pré-processado com os métodos propostos e é gerado um modelo para cada um dos conjuntos de treinamento pré-processados;
3) Os modelos são avaliados com a AUC (Apêndice B.4) e análise ROC (Apêndice B.2). Esses mesmos passos são feitos com os métodos *oversampling* aleatório, *Cluster-based Oversampling* e SMOTE.

O melhor modelo e, portanto, o melhor método de pré-processamento é aquele que obtiver maior medida AUC e sua curva ROC dominar as outras em todo espectro no gráfico ROC. Esta metodologia de avaliação está detalhada nas subseções seguintes.

4.1.1 Bases de dados

As bases de dados utilizadas para avaliação foram obtidas no repositório da UCI [Merz & Murphy, 1998]. A Tabela 4.1, a seguir, resume as principais características dessas bases. Nessa tabela, a segunda coluna apresenta a freqüência de casos positivos, a terceira apresenta a quantidade de atributos contínuos e discretos, a quarta contém as classes utilizadas e, finalmente, a última mostra o número de rodadas necessárias para a estabilidade numérica na validação cruzada (Seção 4.1.4).

Tabela 4.1: Bases de Dados Utilizadas

Base de Dados	Positivo (%)	Quantidade de casos	Atributos (Cont, Disc)	Classes (positiva, negativa)	Rodadas
Sonar	47%	208	60 (60,0)	(M, R)	18
Pima	35%	768	8 (8,0)	(1, 0)	40
Vehicle	24%	846	18 (18,0)	(van, *outras*)	2
Letter-vogal	20%	20000	16 (16,0)	(vogais, *outras*)	1
Vowel	9%	990	12 (10,2)	(0, *outras*)	12
Glass	8%	214	9 (9,0)	(Ve-win-float-proc, *outras*)	600
Forest	7%	38501	27 (10,17)	(4, 3)	1
Yeast	4%	483	8 (8,0)	(CYT, POX)	86
Letter-a	4%	20000	16 (16,0)	(a, *outras*)	1

A escolha destas bases de dados foi feita de forma a garantir que os métodos propostos pudessem ser avaliados em problemas reais e com diferentes tamanhos e distribuições de classe. De forma análoga a [Batista, Prati & Monard, 2004] e [Guo & Viktor, 2004], as bases de dados com mais de duas classes tiveram as classes não minoritárias colapsadas em uma classe, representadas na coluna *Classes* da Tabela 4.1, pelo vocábulo *outras*. Da base *Yeast* foram utilizadas apenas as classes *POX* e *CYT*, respectivamente tomadas como classe positiva e classe negativa, em consonância com [Guo & Viktor, 2004]. A base *Letter-vogal* foi construída a partir da base original *Letter*, utilizando as vogais como classe positiva e as restantes como classe negativa, de forma análoga a [Batista, Prati & Monard, 2004]. Essa mesma estrutura foi aplicada à base *Letter-a*. Da base *Forest* foram utilizadas apenas as classes *3* e *4*, assim como em [Chawla, Bowyer & Hall, 2002]. Originalmente, a base *Forest* possui 7 classes e 54 atributos, sendo 44 atributos binários. Com a remoção dos casos das outras classes, 27 atributos binários ficaram com apenas um estado possível e, portanto, foram removidos.

4.1.2 Pré-processamento

Foram utilizados os métodos *Oversampling* aleatório, *Cluster-based Oversampling* e SMOTE e os métodos propostos *Cluster-based Smote* e *C-clear*. De forma a simplificar a comparação dos métodos de *oversampling*, a distribuição de classe deles resultante foi fixada em 1:1, isto é, totalmente balanceada. De acordo com Weiss e Provost (2003), uma base de dados com distribuição de classe balanceada tende a apresentar bom desempenho quando a medida de qualidade AUC é utilizada.

A clusterização dos dados utilizada nesta pesquisa foi feita da seguinte maneira: para a obtenção dos *clusters* do conjunto de treinamento com atributos contínuos, foi utilizado o algoritmo *k-means* [MacQueen, 1967] (Apêndice C.1). Optou-se pelo *k-means* pela simplicidade de seu funcionamento e implementação, embora esse método requeira o informe do número de *clusters*. Nesse caso, foi feita uma análise de sensibilidade para a aplicação do método *C-clear* à base Pima, resultando no uso de $k = 15$ para o número de *clusters*. A métrica de distância utilizada para o *k-means* foi a distância euclidiana normalizada pela variância (Apêndice D). Para clusterização de atributos categóricos, optou-se pelo método *Squeezer* [He *et al.*, 2002] (Apêndice C.2) por ele sugerir uma heurística para cálculo do número de *clusters*, seu único parâmetro de entrada. A métrica de distância utilizada no *Squeezer* foi a VDM (Apêndice D). Como freqüentemente bases de dados reais possuem tanto atributos contínuos quanto categóricos, foi implementado o *framework* de clusterização de atributos mistos CEBMDC (Apêndice C.2) proposto por He *et al.* (2005). O *k-means* foi utilizado para os atributos contínuos e o *Squeezer* foi utilizado para os atributos categóricos e, também, como metaclusterizador. Esses métodos de clusterização foram implementados em Java 5 no módulo de aprendizagem não supervisionada do UnBMiner [Ladeira *et al.*, 2005].

Para o *Cluster-based Oversampling*, os *clusters* das bases de dados com atributos contínuos foram construídos com o *k-means* com $k = 5$, da mesma forma que utilizado em [Japkowicz, 2002]. Já para as bases de dados com atributos categóricos foi utilizado o *Squeezer*, sendo que a quantidade de *clusters* foi determinada pela heurística definida em [He, Xu & Deng, 2002]. As bases com atributos mistos foram clusterizadas com o CEBMDC, utilizando $k = 5$ para os atributos contínuos. Esse mesmo procedimento foi

também aplicado para a obtenção dos *clusters* para o método proposto *Cluster-based Smote*, de forma a propiciar uma comparação justa entre os dois métodos.

O *C-clear* foi utilizado com **limiarSmote** fixado na freqüência de casos positivos (coluna 2 da Tabela 4.1), **limiarLimpezaPositivo** e **limiarLimpezaNegativo** fixados em 0,7, e k = 15 (quantidade de *clusters* resultantes do método *k-means*). A quantidade de *clusters* obtidos com o método *k-means* foi fixada em k = 15. Um estudo sobre o impacto dos valores desses parâmetros e uma forma de aprendê-los a partir dos dados foram deixados como trabalho futuro. A intuição quanto ao elevado número de *clusters* utilizados foi amenizar a chance de se excluir *clusters* com alta representatividade e, desta forma, prevenir uma possível degradação do classificador resultante. A mesma intuição foi utilizada para os parâmetros **limiarLimpezaPositivo** e **limiarLimpezaNegativo**. Um valor demasiadamente baixo para **limiarLimpezaPositivo** removeria muitos casos da classe negativa, enquanto que um valor muito alto para o parâmetro **limiarLimpezaNegativo** removeria muitos casos positivos considerados como ruído (Seção 3.2). O *C-clear* foi avaliado de duas formas: com e sem limpeza. Desta seção em diante, o método *C-clear* grafado com o símbolo '-' (como em *C-clear*) refere-se à limpeza de dados *habilitada* e, quando grafado com o símbolo '+' (como em *C+clear*), refere-se à limpeza de dados *desabilitada*.

4.1.3 Modelagem

Os classificadores foram induzidos com o C4.5, utilizando a correção de Laplace. Esta correção objetiva melhorar a precisão na estimação de probabilidades com base na freqüência observada dos casos positivos sobre o número total de casos [Ferri et al., 2002]. No estudo em questão, as probabilidades a serem utilizadas para a construção da curva ROC foram inferidas a partir do número de casos reportados pelo C4.5 em cada classe representada nas folhas da árvore. Se uma folha apresenta apenas 1 caso positivo e 0 caso negativo, e outra folha apresenta 50 casos positivos e 0 caso negativo, a probabilidade estimada por ambas as folhas é 1. No entanto, a probabilidade estimada a partir da folha que apresenta maior freqüência é mais precisa. Com a suavização, o cálculo para essas probabilidades passa a ser, respectivamente, 2/3 e 51/52. O C4.5 foi utilizado sem poda, pois de acordo com Batista, Prati & Monard (2004), a aplicação de poda raramente resulta em melhora na medida AUC.

4.1.4 Estrutura de Avaliação

Os métodos propostos *Cluster-Based Smote* e *C-clear* foram aplicados, durante a fase de pré-processamento, às bases de dados da UCI com o objetivo de reduzir o desbalanceamento de classes e de melhorar a aprendizagem de casos raros. As bases de dados resultantes foram utilizadas para a aprendizagem de classificadores com a aplicação dos formalismos de árvore de decisão, C4.5 (Apêndice A.1). Para efeito de comparação de desempenho dos classificadores obtidos, as mesmas bases de dados desbalanceadas iniciais foram pré-processadas com a aplicação dos métodos *Oversampling* aleatório, *Cluster-Based Oversampling* e SMOTE, considerados para efeito de controle. Similarmente, as bases de dados resultantes foram utilizadas para a aprendizagem de classificadores com base nos algoritmos C4.5. O processo de avaliação da performance dos classificadores obtidos utilizou a validação cruzada com dez dobras (do inglês, *10-fold cross validation*), indicada por Kohavi (1995) como a

forma de avaliação mais eficiente para a seleção de modelos. Resumidamente, a metodologia empregada consistiu nos seguintes passos:

1. Cada base de dados desbalanceada inicial é divida em dez subconjuntos com a utilização de amostragem estratificada, visando conservar a distribuição de classes para cada um dos dez subconjuntos;
2. O método *10-fold cross validation* é aplicado, consistindo nos seguintes passos:
   2.1. A cada uma das dez dobras da validação cruzada, a base de dados em questão é dividida em um subconjunto de treinamento, composto de nove dos dez subconjuntos descritos no passo 1, e um subconjunto de avaliação, composto do subconjunto restante;
   2.2. O conjunto de treinamento é balanceado com a aplicação dos métodos de pré-processamento propostos (*Cluster-based Smote* e *C-clear*) e dos métodos de pré-processamentos de controle (*oversampling* aleatório, *Cluster-based Oversampling* e SMOTE) sendo gerados dois modelos para cada um dos conjuntos de treinamento balanceados, com a aplicação do C4.5;
   2.3. Cada modelo gerado (correspondendo a um classificador) é avaliado quanto ao sucesso na classificação, sendo para isso calculada a matriz de confusão e, a partir dela, a curva ROC e a AUC, área sob essa curva, conforme descrito em [Fawcett, 2004];
   2.4. Ao final das dez dobras, são calculadas a média e o desvio padrão das curvas ROC e das AUC obtidas em cada uma das dobras;
3. Os passos 1 a 2.3 podem ser repetidos até que haja uma estabilidade numérica da média AUC computada no passo 2.4. Cada ciclo composto dos passos 1 a 2.3 é denominado uma rodada. Se houver mais de uma rodada, a média e o desvio padrão da AUC e dos pontos que constituem a curva ROC, feitos no passo 2.4, são calculados ao final de todas as rodadas.
4. O melhor modelo e, conseqüentemente, o melhor método de pré-processamento, é aquele que obtiver maior AUC e a sua curva ROC (Apêndice B.2) dominar as outras em todo espectro no gráfico ROC.

Esta forma de avaliação feita em lote permite que cada método de pré-processamento receba os mesmos dados de entrada e, portanto, garante uma comparação mais justa entre eles. A estabilidade numérica adotada no passo 3 foi a precisão na unidade da métrica AUC.

4.2 Resultados e Análise das Avaliações

Os valores AUC obtidos na avaliação dos classificadores, e seus respectivos desvios padrão, expressos entre parênteses, são apresentados na Tabela 4.2, a seguir. Nessa tabela e nas demais a seguir, os métodos são representados pelas seguintes siglas: *Oversampling* – Over; *Cluster-based Oversampling* – CBO; *Cluster-based Smote* – CBS.

Tabela 4.2: AUC das Bases Sonar, Pima, Vehicle, Letter-vogal e Vowel.

	Original	*Over*	SMOTE	CBO	CBS	*C+clear*	*C-clear*

Sonar	80,32 (9,80)	80,59 (9,53)	80,54 (10,20)	79,69 (10,44)	**81,71 (9,92)**	81,23 (8,83)	80,48 (10,18)
Pima	76,74 (5,09)	75,89 (5,46)	75,71 (5,33)	74,73 (5,64)	74,23 (5,54)	77,06 (5,36)	**78,37 (4,84)**
Vehicle	**98,09 (1,36)**	98,02 (1,43)	97,68 (1,30)	97,83 (1,47)	98,01 (1,19)	97,45 (1,24)	96,34 (2,01)
Letter-vogal	54,68 (0,85)	54,69 (0,86)	93,94 (0,50)	54,69 (0,86)	**94,47 (1,18)**	93,73 (0,56)	89,47 (2,12)
Vowel	97,91 (3,42)	97,94 (3,05)	98,16 (2,89)	97,69 (3,35)	97,93 (3,30)	**98,46 (2,38)**	96,67 (2,74)
Glass	72,67 (5,87)	72,67 (5,87)	73,86 (13,26)	72,67 (5,89)	**74,39 (13,06)**	74,20 (13,47)	73,86 (15,40)
Forest	95,85 (0,57)	95,31 (0,33)	97,62 (0,24)	95,28 (0,83)	**98,12 (0,29)**	98,10 (0,25)	93,31 (0,56)
Yeast	65,12 (22,27)	59,75 (22,10)	74,12 (20,57)	63,51 (23,07)	70,17 (21,36)	**77,94 (19,32)**	65,72 (22,29)
Letter-a	63,66 (1,06)	63,39 (1,15)	**99,71 (0,23)**	63,38 (1,12)	99,60 (0,41)	99,70 (0,26)	98,11 (1,98)

[Prati, Batista & Monard, 2004] plotam as AUC obtidas a partir das bases sem pré-processamento *versus* a proporção entre casos positivos e negativos de cada uma, concluindo que o desbalanceamento de classes não é necessariamente a causa da degradação do desempenho de classificadores. Esse mesmo procedimento foi realizado nesta pesquisa e os resultados são apresentados na Figura 4.1, a seguir. Nessa figura, também é observada a mesma conclusão desses autores e exemplos como as bases *Forest* e *Vowel* corroboram esta conclusão, pois resultam em altos valores AUC mesmo contendo apenas 7% e 9% de exemplos da classe positiva, respectivamente.

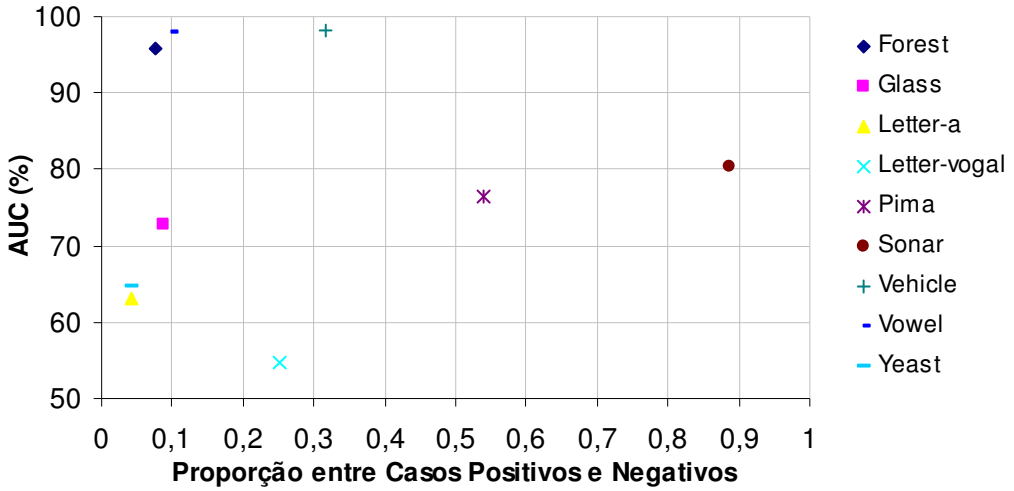

Figura 4.1: Distribuição de Classe Positiva/Negativa *versus* AUC

A Tabela 4.3, a seguir, apresenta o *ranking* dos métodos de pré-processamento para cada base de dados utilizada nesta pesquisa. Os métodos propostos nesta pesquisa estão em negrito. Assim como em [Batista, Prati & Monard, 2004], os melhores métodos foram comparados com o restante dos métodos com o intuito de se verificar se eles são significativamente melhores. Para tanto, foi utilizada a análise de variância (ANOVA) com nível de significância $\alpha = 0,05$. Os campos em cor cinza da Tabela 4.3 representam os métodos que são significativamente piores que os melhores métodos, estes apresentados na coluna '1º'.

Tabela 4.3: Ranking dos Métodos de Pré-processamento

Base	**1º**	**2º**	**3º**	**4º**	**5º**	**6º**	**7º**
Sonar	**CBS**	**C+clear**	Over	SMOTE	**C-clear**	Original	CBO
Pima	**C-clear**	**C+clear**	Original	Over	SMOTE	CBO	**CBS**
Vehicle	Original	Over	**CBS**	CBO	SMOTE	**C+clear**	**C-clear**
Letter-vogal	**CBS**	SMOTE	**C+clear**	**C-clear**	Over	CBO	Original
Vowel	**C+clear**	SMOTE	Over	**CBS**	Original	CBO	**C-clear**
Glass	**CBS**	**C+clear**	**C-clear**	SMOTE	CBO	Original	Over
Forest	**CBS**	**C+clear**	SMOTE	Original	Over	CBO	**C-clear**
Yeast	**C+clear**	SMOTE	**CBS**	**C-clear**	Original	CBO	Over
Letter-a	SMOTE	**C+clear**	**CBS**	**C-clear**	Original	Over	CBO

O *C-clear* (com limpeza) apresentou significativamente o melhor resultado para a base *Pima* e significativamente os piores resultados para três das demais bases. É provável que esses últimos resultados sejam efeitos da escolha não adequada para os parâmetros de limpeza **limiarLimpezaPositivo** e **limiarLimpezaNegativo**, produzindo uma remoção excessiva de casos de ambas as classes. O maior impacto produzido por essa limpeza foi observado na aplicação do *C-clear* à base de dados *Yeast*, na qual a limpeza de dados resultou na diminuição de 12 pontos percentuais em relação à não aplicação da limpeza (*C+clear* = 77,94% e *C-clear* = 65,72%). Por outro lado, o método *C+clear* apresentou bons resultados, sendo ele o melhor em duas bases e o segundo melhor em outras cinco. O método *Cluster-based Smote* também se mostrou competitivo, sendo o melhor em quatro bases, o terceiro melhor em três, e melhor que o *Cluster-based Oversampling* em oito das nove bases de dados utilizadas nesta pesquisa. O desempenho do *C+clear* foi significativamente superior em três bases em relação ao desempenho do SMOTE, como apresentado na Tabela 4.4, e o desempenho do *Cluster-based Smote* foi significativamente superior ao desempenho do *Cluster-based Oversampling* em cinco bases, como apresentado na Tabela 4.5.

Tabela 4.4: Bases em que o *C+clear* é melhor que o SMOTE.

Base	*Sonar*	*Pima*	*Vehicle*	*Letter-vogal*	*Vowel*	*Glass*	*Forest*	*Yeast*	*Letter-a*
Melhora?		×					×	×	

Tabela 4.5: Bases em que o *Cluster-based Smote* é melhor que *Cluster-based Oversampling*.

Base	*Sonar*	*Pima*	*Vehicle*	*Letter-vogal*	*Vowel*	*Glass*	*Forest*	*Yeast*	*Letter-a*
Melhora?	×			×			×	×	×

As figuras a seguir apresentam os gráficos ROC para as bases de dados utilizadas nesta pesquisa. Para cada curva nesses gráficos, são apresentados o nome do método utilizado e a respectiva AUC obtida com eles. Para maior clareza, os métodos serão referenciados pelo nome da curva do modelo gerado com ele. A curva do melhor método de cada gráfico ROC está pontilhada com '×' em vermelho.

Na Figura 4.2, gráfico ROC da base *Sonar*, observa-se que a curva do método *Cluster-based Smote* domina as demais curva até próximo ao ponto fpr = 0,3 e, cruza com as outras curvas a partir deste ponto até o ponto fpr = 0,7, onde passa novamente a dominar as outras curvas. A curva do método *Cluster-based Oversampling* foi dominada por todas as demais curvas do ponto fpr = 0,2 em diante.

Na Figura 4.3, é apresentado o gráfico ROC para a base de dados *Pima*. Diferentemente do gráfico da base *Sonar*, as curvas se apresentam um pouco mais afastadas umas das outras. Observa-se que a curva *C-clear* domina as demais curvas a partir do ponto fpr = 0,2 até fpr = 0,7.

O gráfico ROC da base *Vehicle* é apresentada na Figura 4.4. Nele, as curvas Original e *Cluster-based Smote* compartilham vários pontos da *convex hull*. Observa-se a curva *C-clear* sendo dominada desde fpr = 0 até pouco mais de fpr = 0,1.

A Figura 4.5 apresenta o gráfico ROC para a base *letter-vogal*. Nele, observa-se a dominância dos métodos baseados no SMOTE em relação aos métodos O*versampling* aleatório e *Cluster-based Oversampling*. Observa-se também o efeito negativo proporcionado pela limpeza feita com o *C-clear* e a dominância do método *Cluster-based Smote* em boa parte do espectro.

Na Figura 4.6, gráfico ROC da base *Vowel*, a curva *C-clear* é dominada por todas as outras até pouco antes do ponto fpr = 0,1 e, deste ponto em diante, passa a compartilhar pontos com a curva do método *C+clear*, que foi o melhor método para essa base.

A Figura 4.7 apresenta o gráfico ROC para a base *Glass*. O método *Cluster-based Smote* domina as demais curvas até o ponto fpr = 0,3 e passa a cruzar com as demais curvas no restante do espectro.

O gráfico ROC da base *Forest* é apresentada na Figura 4.8. Nele, os métodos *Cluster-based Smote* e *C+clear* dominam os demais métodos praticamente em todo espectro do gráfico. Observa-se com maior detalhe o efeito negativo da limpeza feita pelo *C-clear*, cuja curva é visivelmente dominada pelas demais curvas.

A Figura 4.9 apresenta o gráfico ROC para a base *Yeast*. A dominância do método *C+clear* é mais perceptível. Esta curva domina as demais até próximo do ponto fpr = 0,5 e, a partir desse ponto, compartilha a dominância com o método SMOTE. Observa-se o baixo desempenho do método *Oversampling* aleatório, sendo a curva deste dominada por todas as outras.

O gráfico ROC da base *Letter-a* é apresentado na Figura 4.10. Assim como no gráfico ROC da base *Letter-vogal*, apresentado na Figura 4.5, os métodos baseados no SMOTE dominam os métodos baseados no *Oversampling* aleatório. Novamente, a limpeza feita pelo método *C-clear* degrada o desempenho do classificador resultante, como se pode observar pela dominância dos outros métodos baseados no SMOTE em relação a ele.

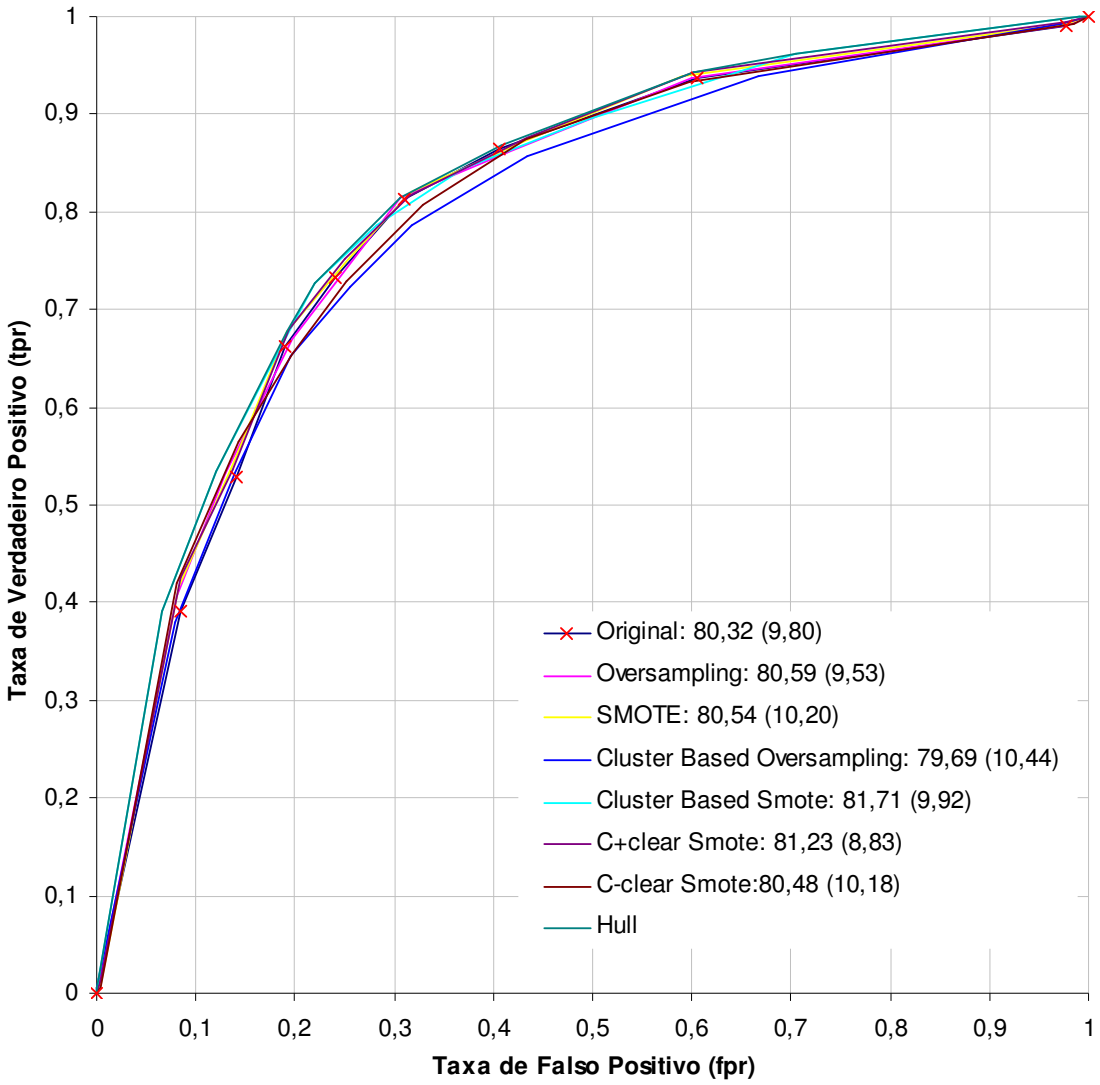

Figura 4.2: Gráfico ROC para a Base de Dados Sonar.

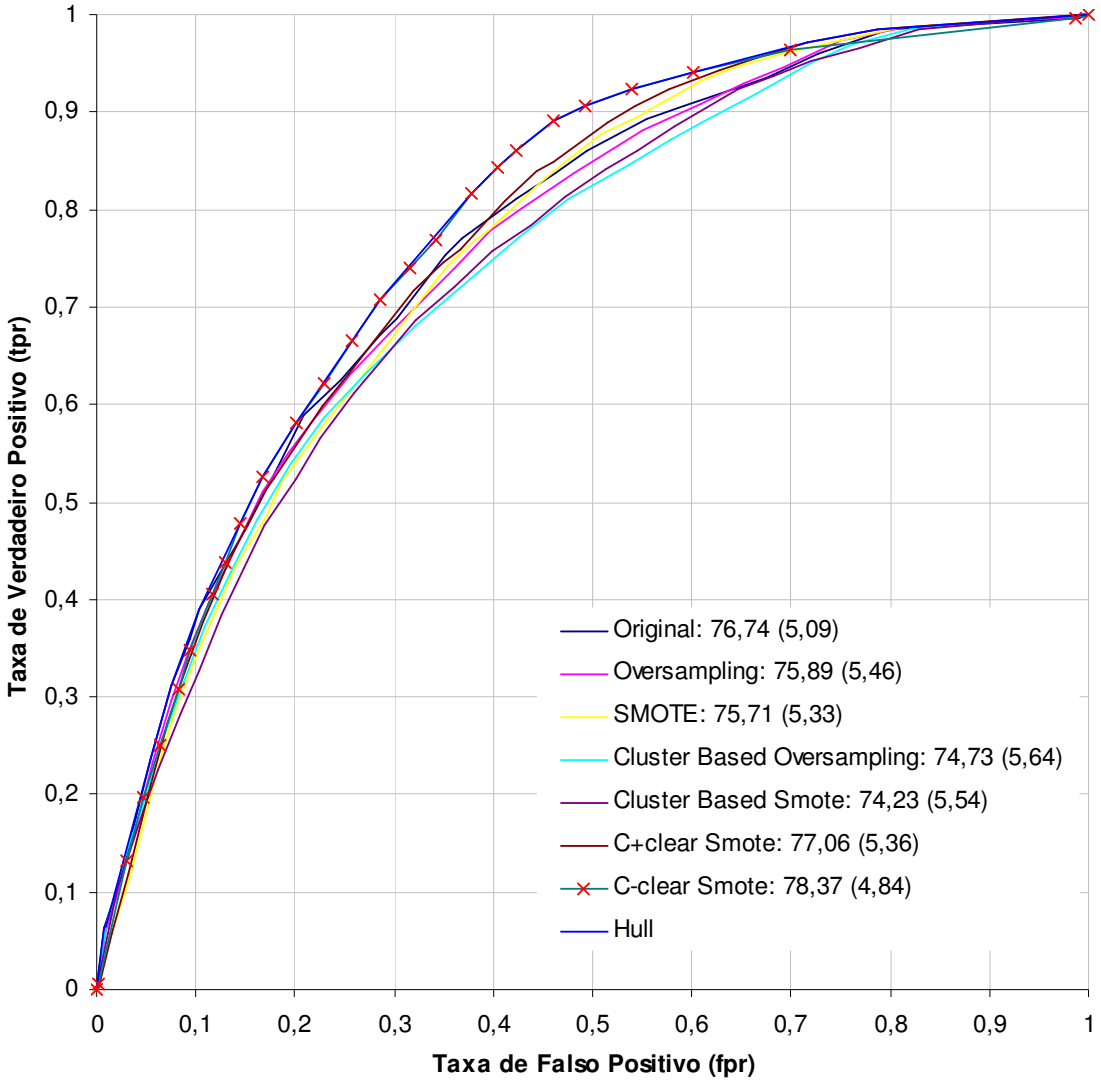

Figura 4.3: Gráfico ROC para a Base de Dados Pima.

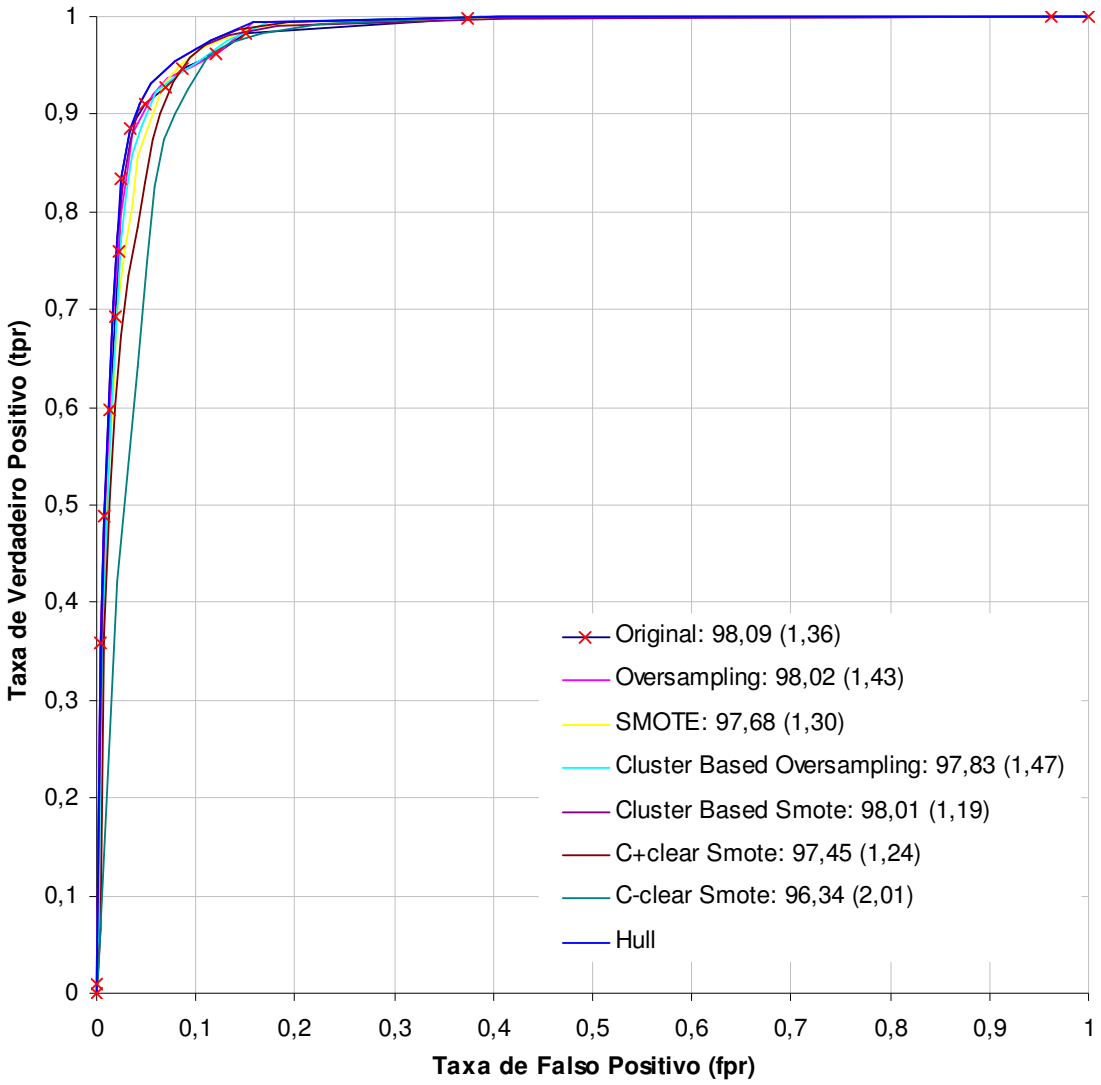

Figura 4.4: Gráfico ROC para a Base de Dados Vehicle.

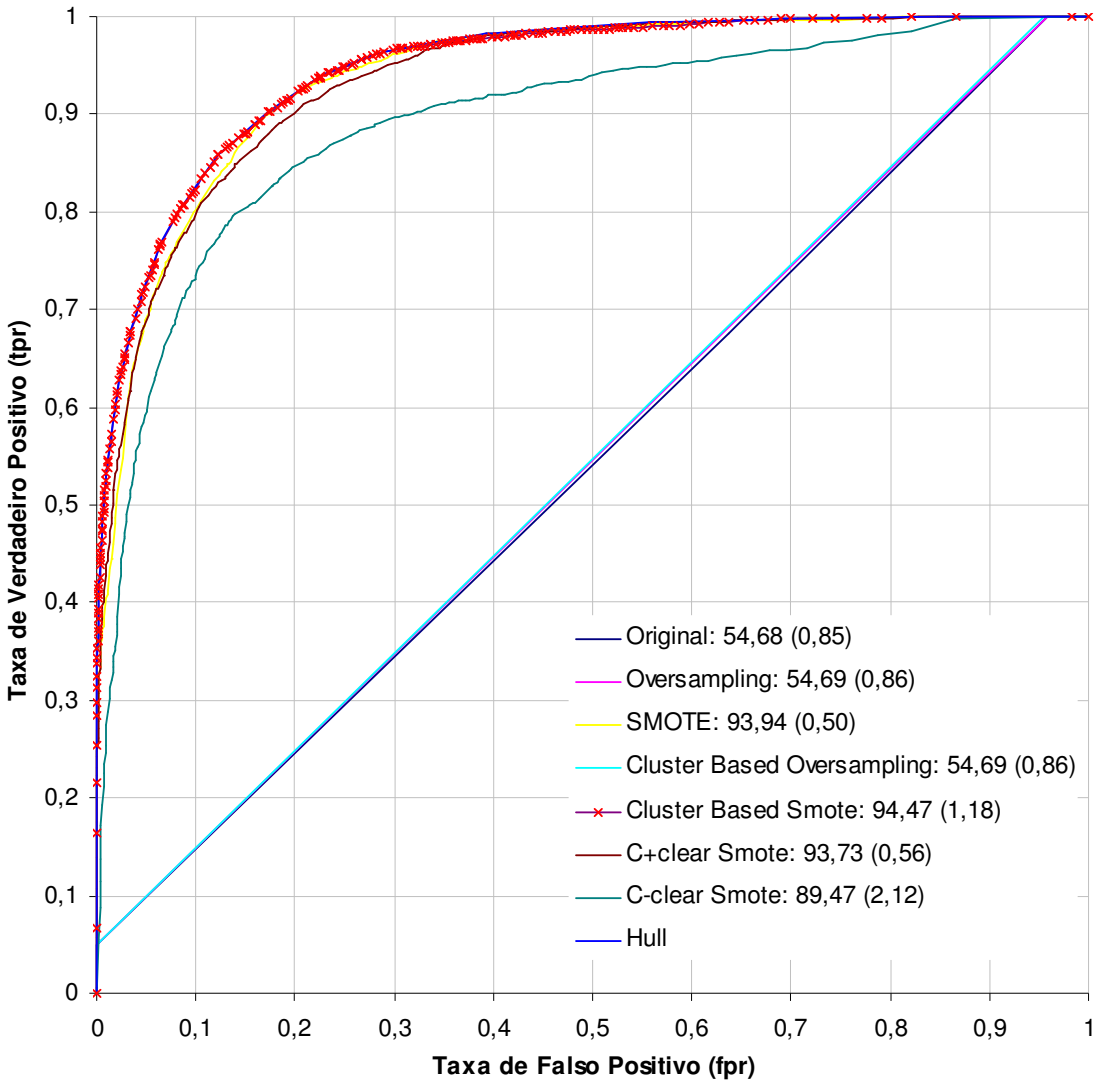

Figura 4.5: Gráfico ROC para a Base de Dados Letter-vogal.

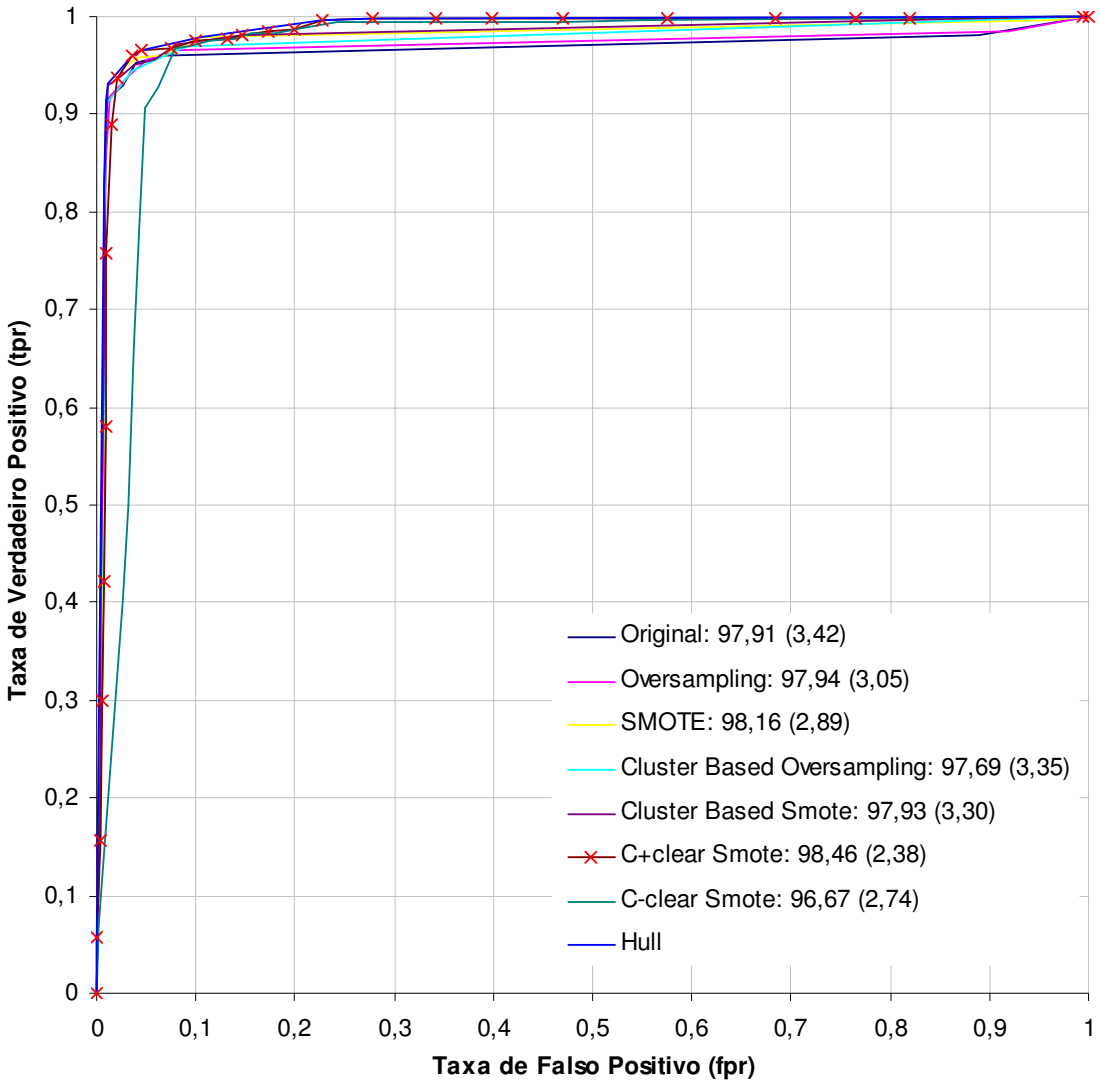

Figura 4.6: Gráfico ROC para a Base de Dados Vowel.

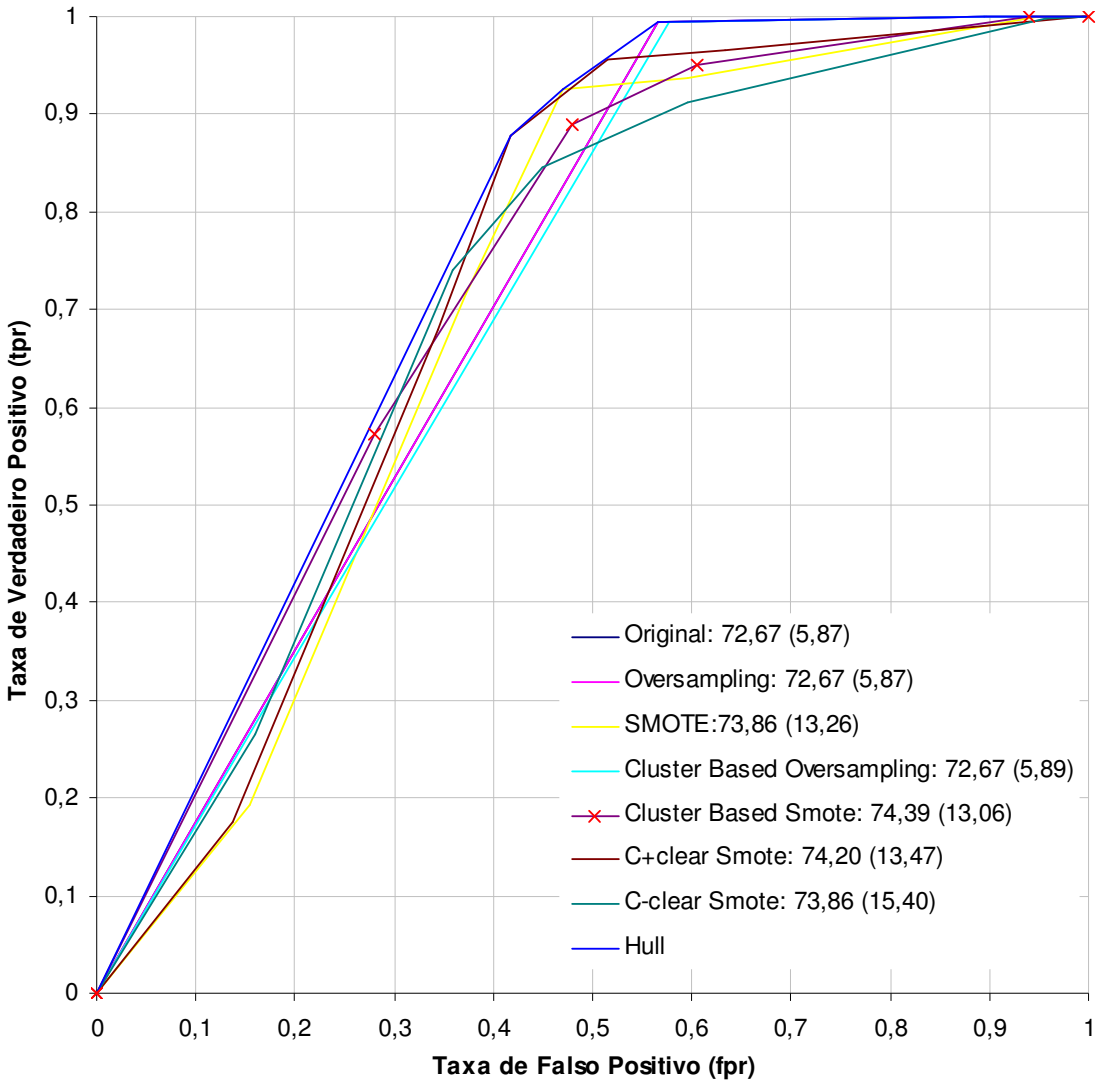

Figura 4.7: Gráfico ROC para a Base de Dados Glass.

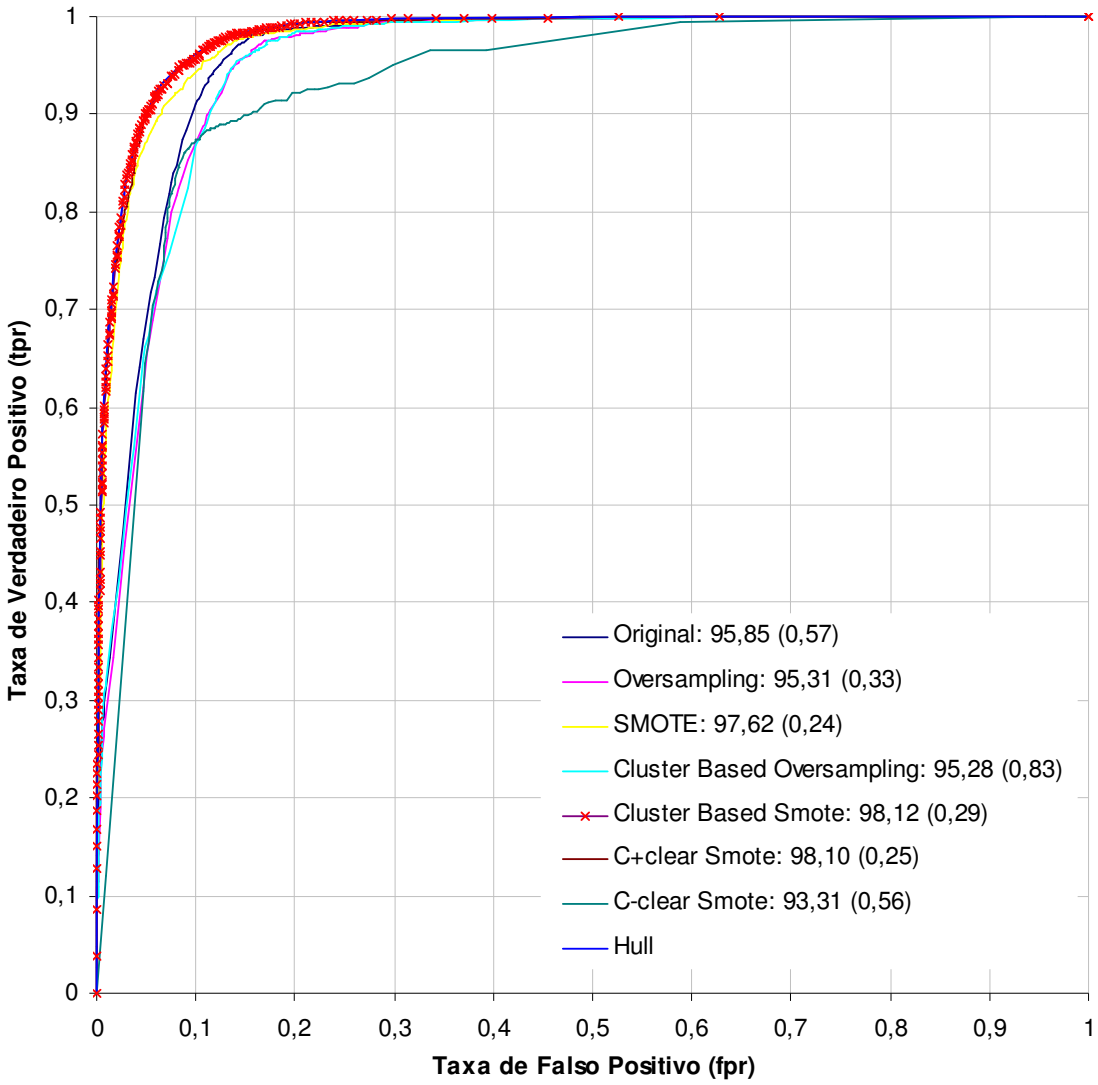

Figura 4.8: Gráfico ROC para a Base de Dados Forest.

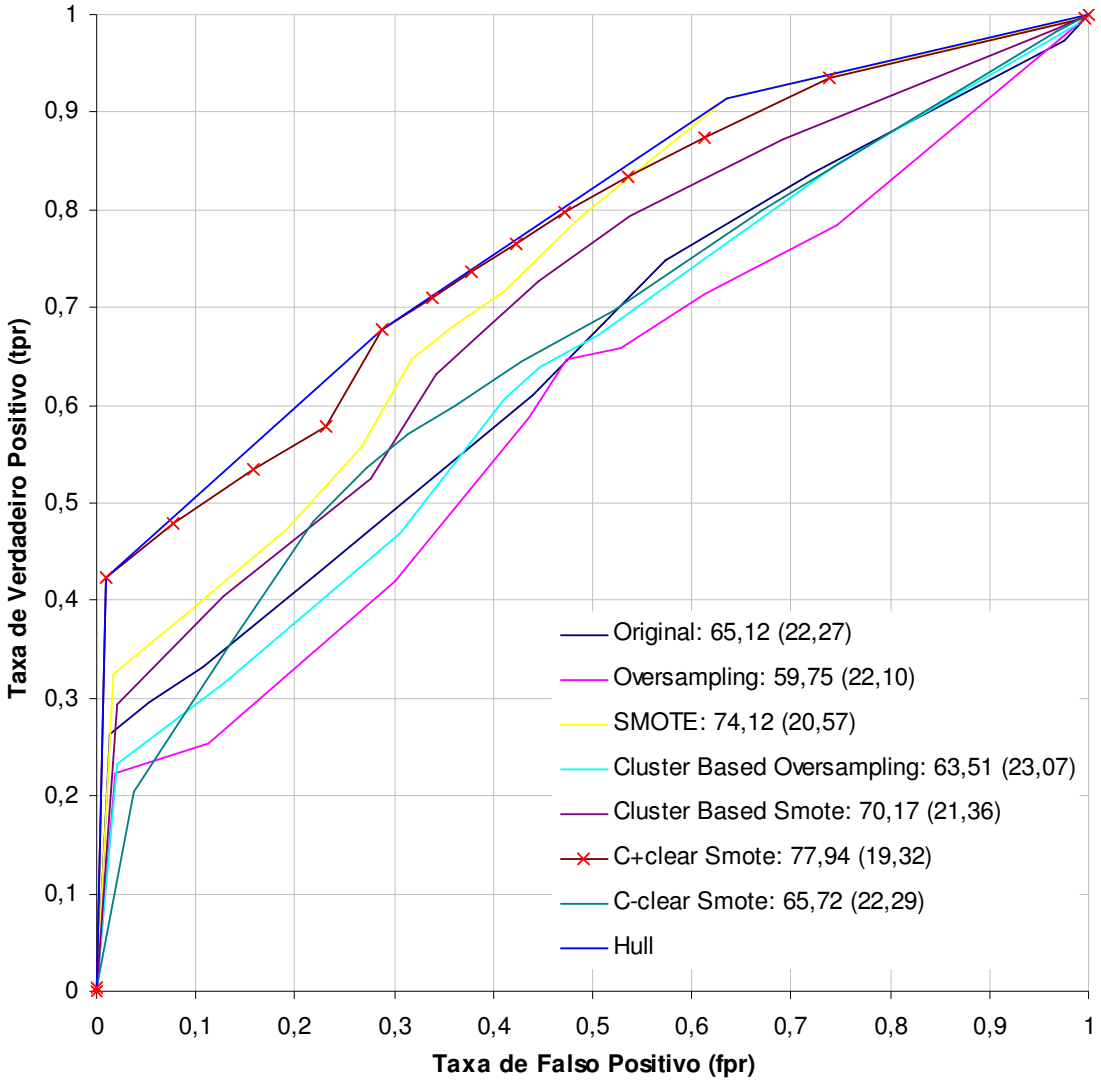

Figura 4.9: Gráfico ROC para a Base de Dados Yeast.

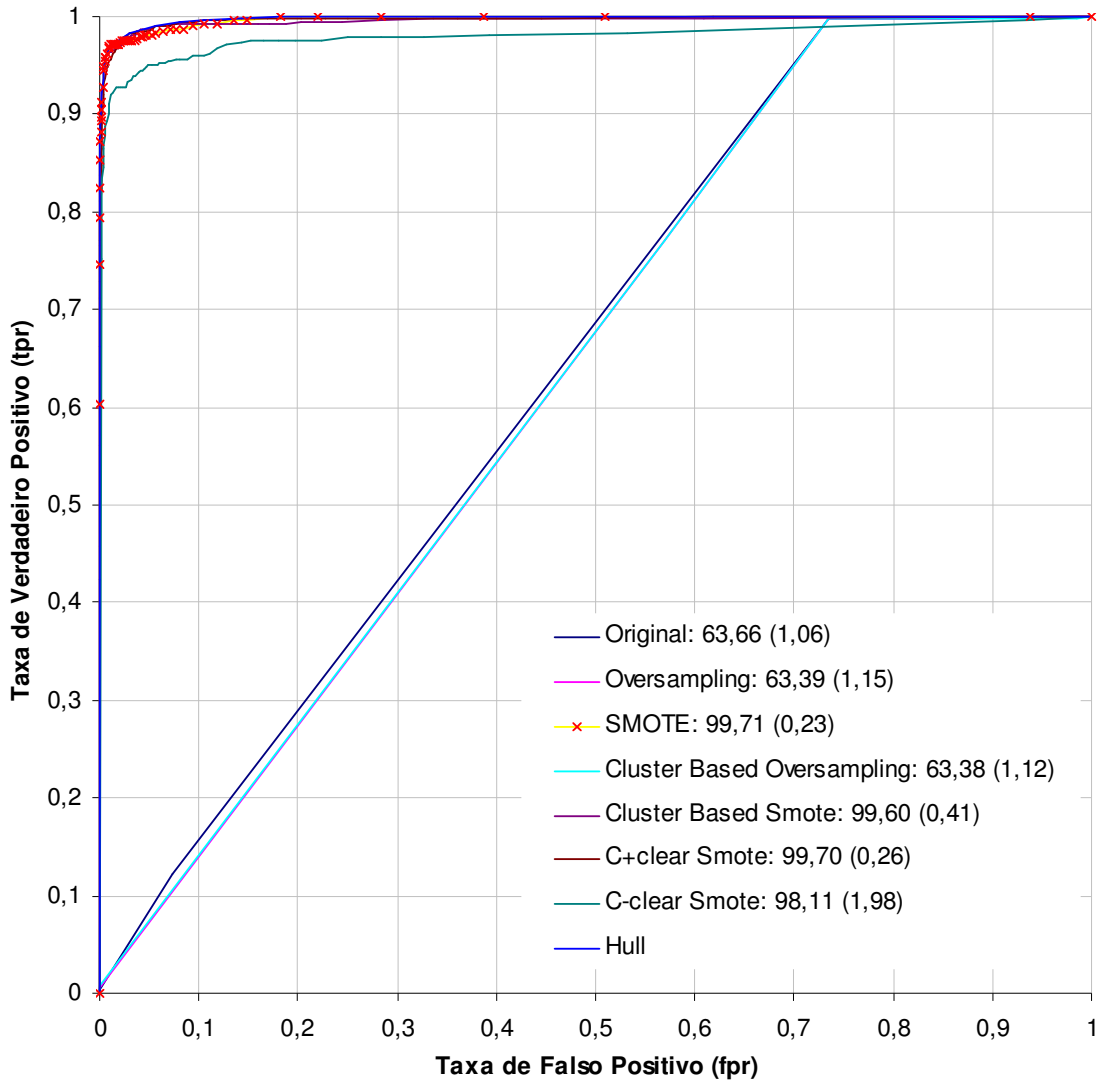

Figura 4.10: Gráfico ROC para a Base de Dados Letter-a.

4.3 Conclusão das Avaliações

A aplicação do método SMOTE guiada pela aprendizagem não supervisionada feita pelo *C+clear* mostrou-se competitiva em relação à aplicação do SMOTE *de per si*. Em três bases o *C+clear* foi significativamente superior que o SMOTE. Nas duas bases em que o SMOTE foi superior ao *C+clear*, *Letter-a* e *Letter-vogal*, o ganho não foi significativo.

A limpeza feita com o método *C-clear* somente obteve bom desempenho na base *Pima*. É provável que este comportamento do *C-clear* seja resultado da escolha inadequada dos parâmetros de limpeza **limiarLimpezaPositivo** e **limiarLimpezaNegativo** , e do número de *clusters* (k = 15) para as demais bases. Possivelmente, aprender esses parâmetros a partir dos próprios dados se mostre eficiente. Esse estudo fica como trabalho futuro.

A proposta de se substituir a simples replicação do método *Cluster-based Oversampling* pela geração de casos sintéticos com o SMOTE resultou em melhora significativa na métrica AUC. O método *Cluster-based Smote*, que une essas duas abordagens, superou significativamente o método *Cluster-based Oversampling* em oito das nove bases

utilizadas nesta pesquisa, e foi significativamente o melhor método em quatro das nove bases.

A análise de curvas ROC permitiu a verificação do comportamento dos métodos de pré-processamento em todo espectro do limiar de decisão. Porém, os melhores métodos dominaram os demais somente em determinadas faixas do espectro do gráfico ROC. Entretanto, segundo Prati, Batista & Monard (2003), mesmo que esta dominância do melhor método não seja estatisticamente significativa, a utilização de uma distribuição de classe ou custo de classificação baseada nesta faixa do espectro, onde o melhor método domina os demais, pode resultar na escolha de um modelo significativamente melhor do que os proporcionados pelos demais métodos.

Capítulo 5 Conclusão

Esse capítulo descreve as conclusões e resultados obtidos e as linhas de trabalho futuro. Para melhor contextualização, é apresentada uma breve recapitulação do objetivo geral, da motivação, dos resultados esperados e das linhas de ação adotadas nesta pesquisa. Logo após, são apresentadas as condições de término desta pesquisa, e explicitadas as restrições admissíveis para a solução proposta.

5.1 Resumo dos Objetivos

O objetivo geral é explorar técnicas para aumentar a acurácia de algoritmos classificadores aplicados a bases de dados desbalanceadas com foco nos métodos de amostragem. O objetivo específico é propor novas técnicas de balanceamento e/ou limpeza de dados, baseadas na combinação de características dos algoritmos SMOTE e *Cluster-based Oversampling*.

5.2 Resultados Obtidos

Foram propostas duas abordagens baseadas no funcionamento dos métodos de pré-processamento SMOTE e *Cluster-based Oversampling*.

A primeira abordagem, *Cluster-based Smote*, utiliza a geração de casos sintéticos feita pelo método SMOTE para povoar os pequenos disjuntos encontrados com o método *Cluster-based Oversampling*, ao invés de simplesmente replicá-los. A motivação foi melhorar a aprendizagem de pequenos disjuntos e, ao mesmo tempo, evitar o *overfitting* do modelo a eles. Os resultados apresentados na Seção 4.2 indicam que, realmente, o método *Cluster-based Oversampling* gera modelos muito ajustados (*overfitted*) para os dados de treinamento e, por conseguinte, para os pequenos disjuntos. A aplicação do SMOTE realizada pelo *Cluster-based Smote*, proporcionou resultados superiores ao do método *Cluster-based Oversampling* em oito das nove bases utilizadas nesta pesquisa (Tabela 4.5), e foi o método que apresentou o melhor desempenho em quatro bases (Tabela 4.3).

A segunda abordagem foi proposta para amenizar a sobreposição de classes que possivelmente venha a ser causada pela aplicação do SMOTE. Essa abordagem consiste em guiar o método SMOTE pela aprendizagem não supervisionada, aplicando-o somente aos *clusters* onde a freqüência de casos positivos seja maior que certo limiar e, logo após, aplicar ou não a limpeza de dados. Essa limpeza consiste na eliminação dos casos negativos (positivos) de todo *cluster* cuja freqüência positiva (negativa) seja maior que certo limiar. O *C-clear*, método desenvolvido sob essa abordagem, foi testado com e sem a aplicação de limpeza (*C-clear* com limpeza; *C+clear* sem limpeza). Os resultados obtidos com o *C+clear* indicam que a aplicação do SMOTE orientada pela aprendizagem não supervisionada atenua o problema de sobreposição de classes gerada, ou mesmo, intensificada pelo SMOTE. O *C+clear* apresentou bons resultados, sendo ele o melhor em duas bases e o segundo melhor em outras cinco (Tabela 4.3). Além disso, o desempenho do *C+clear* foi significativamente superior ao desempenho do SMOTE em três bases, como apresentado na Tabela 4.4 e o SMOTE não apresentou nenhum resultado significativamente melhor que o *C+clear*. Já a

limpeza efetuada pelo *C-clear* somente surtiu melhora na base *Pima* e rendeu os piores resultados para três das demais bases. Possivelmente, a causa desta degradação esteja vinculada a uma escolha não adequada dos parâmetros de limpeza **limiarLimpezaPositivo** e **limiarLimpezaNegativo**, que produziu uma remoção excessiva de casos de ambas as classes.

5.3 Trabalho Futuro

Para esta pesquisa, foram utilizadas algumas heurísticas, tais como o número de *clusters* do algoritmo *k-means* fixo em $k = 15$ para o *C-clear*, o parâmetro **limiarSmote** do método *C-clear* fixo no valor da freqüência da classe positiva, e os limiares de limpeza **limiarLimpezaPositivo** e **limiarLimpezaNegativo** fixados nos valores 0,7. Entretanto, os valores estipulados para esses parâmetros possivelmente não são ótimos para todas as bases. Os resultados obtidos com a limpeza promovida pelo *C-clear*, por exemplo, corroboram essa hipótese, pois somente a base *Pima* apresentou ganho de desempenho com este método, e as três demais resultaram em degradação significativa no desempenho. Portanto, um trabalho futuro seria estender o *C-clear* para aprender seus parâmetros a partir dos próprios dados de treinamento, como feito em [Rakotomamonjy, 2004][Sing, Beerenwinkel, & Lengauer, 2004][Brefeld & Scheffer, 2005][Prati & Flash, 2005][Ataman, Street & Zhang, 2006]. Para tanto, a estrutura de validação implementada no UnBMiner (Seção 4.1.4) foi projetada de forma que essa extensão possa ser facilmente implementada.

Bibliografia

Ataman, K.; Street, W.N. & Zhang, Y. (2006). *Learning to rank by maximizing AUC with linear programming*. **IEEE International Joint Conference on Neural Networks IJCNN'2006**. p.123-129.

Batista, G.E.A.P.A.; Prati, R.C. & Monard, M.C. (2004). *A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data*. **SIGKDD Explorations**, v.6 p.20-29.

Brefeld, U. & Scheffer, T. (2005). *AUC Maximizing Support Vector Learning*. **Proceedings of the ICML Workshop on ROC Analysis in Machine Learning**.

Boley, D.L. (1998). *Principal Direction Divisive Partitioning*. **Data Mining and Knowledge Discovery**, v.2, n.4, p.325-344.

Castillo, M. & Serrano, J. (2004). *A Multistrategy Approach for Digital Text Categorization from Imbalanced Documents*. **SIGKDD Explorations**, v.6 p.70-79.

Chawla, N.V.; Bowyer, K.W.; Hall, L.O. & Kegelmeyer, W.P. (2002). *SMOTE: Synthetic Minority Over-sampling Technique*. **JAIR**, v.16, p.321–357.

Chawla, N.V.; Japkowicz, N. & Kotcz, A. (2003). (Editors) *Proceedings of the ICML'2003 Workshop on Learning from Imbalanced Data Sets*.

Chawla, N.V.; Japkowicz, N.; Kotcz, A. (2004) (Editors) *Editorial: Special Issue on Learning from Imbalanced Data Sets*. **ACM SIGKDD Explorations**. v.6. p.1-6.

Dasarathy, B.; Sanchez, J. & Townsend, S. (2000). *Nearest Neighbor Editing and Condensing Tools – Synergy Exploitation*. **Pattern Analysis and Applications**. v.3, n.1, p.19-30.

Daskalaki, S.; Kopanas, I. & Avouris, N. (2006). *Evaluation of Classifiers for an Uneven Class Distribution Problem*. **Applied Artificial Intelligence**. v.20, p.381-417

Fayyad, U.M. (1997). *Editorial: Data Mining and Knowledge Discovery*. v.1 p.5-10.

Fayyad, U.M. (2004). (Editor). *Special Issue on Learning from Imbalanced Data Sets*. **ACM SIGKDD Explorations**. v.6.

Fawcett, T. (2004) *ROC Graphs - Notes and Practical Considerations.* **Machine Learning**.

Ferri, C.; Flach, P. & Hernández-Orallo, J.H. (2002). *Learning Decision Trees using the Area under the ROC curve*. **In C.S.A. Hoffman, editor, Nineteenth International Conference on Machine Learning (ICML'2002)**. Morgan Kaufmann Publishers. p.139–146.

Gama, J. & Brazdil, P. (2000) Cascade Generalization. **Machine Learning**. v.41 n.3 p.315-343.

Guo, H. & Viktor, H.L. (2004). *Learning from Imbalanced Data Sets with Boosting and Data Generation: The DataBoost-IM Approach*. **SIGKDD Explorations**. v.6 p.30-39

Han, H.; Wang, W.Y. & Mao, B.H. (2005) *Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning*. **In Proceedings of ICIC**. Hefei. p.878-887.

Hart, P. E. (1968). *The Condensed Nearest Neighbor Rule*. **IEEE Transactions on Information Theory IT-14**. p.515–516.

He, Z; Xu, X. & Deng, S. (2002). *Squeezer: an Efficient Algorithm for Clustering* Categorical Data. **Journal of Computer Science and Technology**. v.17, n.5, p.611-625

He, Z; Xu, X. & Deng, S. (2005). *Clustering Mixed Numeric and Categorical Data: a Cluster Ensemble Approach*. **ArXiv Computer Science e-prints**. (Acesso em 12/12/2006. Disponível em: http://arxiv.org/ftp/cs/papers/0509/0509011.pdf)

Japkowicz, N. (2002). *Supervised Learning with Unsupervised Output Separation*. **In Proceedings of the IASTED International Conference on Artificial Intelligence and Soft Computing (ASC)**. p.321-325.

Japkowicz, N. (2003). *Class imbalances: Are we Focusing on the Right Issue?* **In Proceedings of the ICML'03 Workshop on Learning from Imbalanced Data Sets**.

Jo, T. & Japkowicz, N. (2004). *Class Imbalances versus Small Disjuntcs*. **SIGKDD Explorations**, v.6 p.40-49.

Kohavi, R. (1995). *A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection*. **International Joint Conference on Artificial Intelligence (IJCAI'1995)**. p.1137-1145

Kubat, M., Holte, R. & Matwin, S. (1998). *Machine Learning for the Detection of Oil Spills in Radar Images*. **Machine Learning**, v.30 p.195-215.

Kubat, M. & Matwin, S. (1997). *Addressing the Curse of Imbalanced Training Sets: One-sided Selection*. **In Proceedings of ICML**. Nashville: p.179–86.

Lachiche, N. & Flach, P.A. (2003). *Improving Accuracy and Cost of Two-class and Multi-class Probabilistic Classifiers Using ROC Curves*. **ICML'2003**. p.416-423

Ladeira, M; Vieira, M.H.P; Prado, H.A; Noivo, R.M & Castanheira, D.B.S (2005). *UnBMiner - Ferramenta Aberta Para Mineração de Dados*. ***Revista Tecnologia da Informação***, Brasília-DF, v.5, n.1, p.45-63.

Langley, P.; Iba, W. & Thompson, K. (1992). *An Analysis of Bayesian Classifiers*. **In Proceedings of the 10th National Conference on Artificial Intelligence**. AAAI Press and MIT Press. p.223-228.

Laurikkala, J. (2001). *Improving Identification of Difficult Small Classes by Balancing Class Distribution*. **(TR A-2001-2)**. University of Tampere.

MacQueen, J.B. (1967). *Some Methods for Classification and Analysis of Multivariate Observations*. **Proceedings of Fifth Berkeley Symposium on Mathematical Statistics and Probability**. Berkeley, University of California Press, v.1, p.281-297.

Merz, C.J. & Murphy, P.M. (1998) UCI Repository of Machine Learning Datasets. http://www.ics.uci.edu/~mlearn/MLRepository.html. (Acesso em 20/01/2007).

Mitchell, T. (1997). **Machine Learning**. New York. McGraw Hill

Nickerson, A.; Japkowicz, N. & Millos, E. (2001). *Using Unsupervised Learning to Guide Resampling in Imbalanced Data Sets*. **In Proceedings of the 8th International Workshop on AI and Statistics**. Key West. p.261-65.

Oliveira, G.L. & Neto, M.G.M. (2004). *ExperText: Uma Ferramenta de Combinação de Múltiplos Classificadores Naive Bayes*. **Anales de la 4ª Jornadas Iberoamericanas de Ingeniería de Software e Ingeniería de Conocimiento**. Madrid. v.1, p.317-32.

Phua, C.; Alahakoon, D. & Lee, V. (2004). *Minority Report in Fraud Detection: Classification of Skewed Data*. **ACM SIGKDD Explorations**. v.6. p.50-59.

Prati, R.C.; Batista, G.E.A.P.A. & Monard, M.C (2003). *Uma experiência no Balanceamento Artificial de Conjuntos de Dados para Aprendizado com Classes Desbalanceadas utilizando Análise ROC*. **IV Workshop de Inteligência Artificial ATAI'2003**.

Prati, R.C.; Batista, G.E.A.P.A. & Monard, M.C (2004). *Class Imbalances versus Class Overlapping: an Analysis of a Learning System Behavior*. **In MICAI**. p.312-321.

Prati, R.C. & Flash, P.A. (2005). *ROCCER: An algorithm for rule learning based on ROC analysis*. **In Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCI'2005)**. p.823-828. Edinburgh, Scotland, UK.

Prati, R.C. (2006). *Novas abordagens em aprendizado de máquina para a geração de regras, classes desbalanceadas e ordenação de casos*. Tese de Doutorado. Disponível em: www.teses.usp.br/teses/disponiveis/55/55134/tde-01092006-155445.

Quinlan, J. R. (1986). *Induction of Decision Trees*. **Machine Learning**, 1(1), p.81-106.

Quinlan, J.R. (1993). *C4.5 Programs for Machine Learning*. **Morgan Kaufmann**, San Mateo, CA.

Sanches, M. K. & Monard, M. C. (2004). *Proposta de um Algoritmo de Clustering Semi-supervisionado para Rotular Exemplos a Partir de Poucos Exemplos Rotulados*. **In Workshop in Artificial Intelligence, Jornadas Chilenas de**

Computación. Sociedad Chilena de Ciencias de la Computación. Arica, Chile. v.1, p.1-9.

Sing, T.; Beerenwinkel, N. & Lengauer, T. (2004). *Learning Mixtures of Localized Rules by Maximizing the Area Under the ROC Curve*. **European Conference on Artificial Intelligence – ROCAI'2004. Workshop on ROC Analysis in AI**. p.89-96.

Raskutti, B. & Kowalczyk, A. (2003). *Extreme Re-balancing for SVMs: a Case Study*. **In Proceedings of Workshop on Learning from Imbalanced Data Sets II**. Washington, DC.

Rakotomamonjy, A. (2004). *Optimizing Area Under Roc Curve with SVMs*. **In First Workshop on ROC Analysis in AI**. Valencia, Spain.

SPSS Inc.; NCR Systems Engineering Copenhagen & DaimlerChrysler AG (1999). *CRISP-DM 1.0 – Step-by-step Data Mining Guide*. **SPSS & CRISP-DM Consortium**. (Disponível em www.crisp-dm.org/CRISPWP-0800.pdf. Acesso em 26/04/2006).

Tomek, I. (1976). *Two Modifications of CNN*. **IEEE Transactions on Systems Man and Communications SMC**. v.6. p.769–772.

Van Rijsbergen, C. J. (1979). **Information Retrieval**. 2ª Edição, London, Butterworths.

Wilson, D.R. & Martinez, T.R. (2000). *Reduction Techniques for Exemplar-Based Learning Algorithms*. **Machine Learning**. v.38, n.3, p 257-286.

Weiss, G. (2004) *Mining with Rarity: A Unifying Framework*. **ACM SIGKDD Explorations** v.6. p.7-19.

Apêndice A Classificadores

Para efeito de construção de classificadores, um conjunto de *n* variáveis pode ser dividido em *n*-1 variáveis atributos e uma variável classe. Por existir apenas uma variável classe, é usual se referir aos estados desta variável como as classes possíveis. Um caso é uma instância deste conjunto de *n* variáveis atributos, isto é, um caso e sua classe formam um registro numa base de dados. Um classificador é um modelo construído a partir de um conjunto de instâncias de atributos-classe (casos e suas respectivas classes), ou seja, é uma função que mapeia os valores das variáveis atributos nos possíveis estados da variável classe.

A.1 C4.5

A aprendizagem por árvores de decisão é um dos métodos mais usados e práticos para inferência indutiva. É um método para obter funções que retornam valores discretos (nominais) a partir dos dados de entrada, na qual a função aprendida é representada por uma árvore de decisão. Cada folha representa uma classe (isto é, um valor possível para os estados da variável de classe), cada nó de decisão especifica um teste a ser feito sobre um atributo que pode levar a uma folha ou alguma outra sub-árvore. O nó raiz contém a variável de atributo mais informativo para a definição do estado da classe da variável de classe. Árvores de decisão podem ser representadas como conjuntos de regras de classificação para melhorar o entendimento de uma pessoa que esteja avaliando os dados. Cada caminho iniciando no nó raiz e finalizando em um nó folha, representa uma regra de classificação. Muitos algoritmos foram desenvolvidos para a construção de árvores de decisão a partir de um algoritmo base que emprega uma construção *top-down*, com procura gulosa pelo espaço das possíveis árvores de decisão. Os algoritmos implementados no UnBMiner foram o ID3 [Quinlan, 1986] e o seu sucessor C4.5 [Quinlan, 1993]. Notar que esses algoritmos não são os originais de Quinlan, e sim uma reimplementação em Java dos originais. O usuário do sistema pode realizar inferências com a árvore de decisão, entrando com um valor para cada um dos nós não folha e percorrendo um caminho do atributo raiz até uma folha. Cada valor induz caminhos a percorrer até se atingir uma folha na qual está representada a classe correspondente.

Basicamente algoritmo ID3 constrói uma árvore recursivamente, como se segue:

1. Um atributo é selecionado como nó raiz e arcos são criados para cada valor de atributo possível.
2. As instâncias de dados são divididas em subconjuntos (um para cada arco que se estende do nó).
3. Pare se todas as instâncias associadas a um arco possuem a mesma classe
4. Repita os passos 1 a 3 para cada arco que possua mais de uma classe, usando apenas as instâncias a ele associadas.

A questão a ser analisada é qual atributo deve ser selecionado no passo 1. Deve ser selecionado aquele capaz de produzir a menor árvore de decisão. Uma boa heurística é escolher o atributo que produz os nós mais puros. Um nó é puro se as instâncias associadas a ele possuem apenas uma classe. O grau de impureza de um nó é máximo quando todas as classes são igualmente distribuídas. Frequentemente, entropia é utilizada como medida de pureza e o maior ganho de informação como critério para selecionar o atributo a ser utilizado. A entropia, medida da Teoria da Informação, mede

o grau de homogeneidade de um conjunto S de instâncias de dados do atributo de interesse (no domínio da classificação, o atributo de classe), variando de 0 (todas as instâncias possuem o mesmo valor) a 1 (todos os valores são igualmente distribuídos) e é dada pela fórmula (2), onde p_i representa a proporção de instâncias de S da classe i.

$$Entropia(S) \equiv \sum_{i=1}^{c} - p_i \log_2 p_i \tag{2}$$

O ganho de informação mede o quanto um atributo divide um conjunto de instâncias S para produzir subconjuntos mais puros em relação a uma variável de interesse, sendo sua fórmula dada pela equação (3).

$$Ganho(S, A) \equiv Entropia(S) - \sum_{v \in val(A)} \frac{|S_v|}{|S|} Entropia(S_v), \tag{3}$$

onde $val(A)$ é o conjunto dos valores possíveis para o atributo A e S_v é o subconjunto de instâncias de S com valor v para o atributo A. Assim, o ganho de informação é a diferença entre a entropia do conjunto S e o valor esperado da entropia depois que S é particionado segundo o atributo A.

O algoritmo C4.5 estende o algoritmo ID3 com a consideração de taxa de ganho de informação, dados faltantes, atributos numéricos e poda da árvore. Suas vantagens são: representação compacta e de fácil visualização, que permite a decomposição de decisões complexas em decisões elementares, e a fácil detecção de atributos redundantes ou irrelevantes. A árvore de decisão pode vir a ser complexa e a classificação correta de novas instâncias depende da representatividade dos dados utilizados no treinamento.

A.2 *K*-NN

Usualmente, um algoritmo classificador tem como entrada um conjunto de treinamento e, a partir deste, gera um modelo que é utilizado para classificar casos novos. O classificador *k*-NN (do inglês, *Nearest Neighbor* – vizinho mais próximo) é um tipo diferente de classificador que não necessita gerar modelo, pois a classificação de novos casos é feita de forma direta. Os casos de um conjunto de dados são interpretados como pontos distribuídos no espaço euclidiano. Quando um novo caso é apresentado ao classificador *k*-NN, ele o classifica (rotula a sua variável classe) com a classe de maior freqüência entre seus *k* vizinhos mais próximos. O algoritmo a seguir mostra os passos deste classificador:

Algoritmo 9 O algoritmo classificador *k*-NN rotula novos casos com a classe de maior freqüência entre seus *k* vizinhos mais próximos. Este algoritmo não cria um classificador para os dados de entrada, ele classifica um novo caso diretamente com os dados já conhecidos.

Descrição das variáveis:
S: conjunto de entrada;
x': novo caso a ser classificado;
k: número de vizinhos mais próximos considerados;

k-NN
Entrada: *S*, *x*', *k*
Saída: *c*'
1. Calcular todas as distâncias entre o caso *x*' e os casos de *S*;

5. Faça $D_{x'}^{k}$ ← conjunto dos *k* vizinhos mais próximos de *x'*;
6. Faça *c'* ← a classe mais freqüente em $D_{x'}^{k}$;
7. Retornar *c'*.

Existem outras variantes mais interessantes do passo 3, como, por exemplo, utilizar pesos distintos para o voto de cada vizinho, conforme a distância entre ele e o caso sendo classificado. Batista, Prati & Monard (2004) utilizam esta idéia como forma de penalizar a contribuição de vizinhos mais longínquos, multiplicando seus votos pelo inverso do quadrado de suas respectivas distâncias em relação ao caso que se quer classificar. A equação (1), a seguir, expressa esta variante do *k*-NN:

$$\mathrm{h}(E_q) = \arg\max_{c \in C} \sum_{i=1}^{k} \omega_i \delta(c, f(\hat{E}_i)) \qquad \omega_i = \frac{1}{d(E_q, \hat{E}_i)^2} \quad (1)$$

onde:

- E_q é o novo caso a ser classificado, e $\hat{E} = \{\hat{E}_1, \hat{E}_2, \ldots, \hat{E}_k\}$ é o conjunto dos *k* casos mais próximos de E_q (de acordo com a métrica escolhida no passo 1 do algoritmo *k*-NN);
- $f(\hat{E}_i)$ é a classe do caso $\hat{E}_i$;
- A função $\delta(a, b) \leftarrow 1$ se $a = b$ e $\delta(a, b) \leftarrow 0$ caso contrário;
- *C* é o conjunto de classes do domínio do problema;
- $\mathrm{h}(E_q)$ retorna a classe que rendeu maior valor à multiplicação de ω_i por δ.

Apêndice B Métodos de Avaliação

Este apêndice apresenta os métodos de avaliação utilizados nesta dissertação, análise ROC e AUC. A Seção B.1 introduz a avaliação de classificadores e o relaciona com o problema de desbalanceamento de classes; o Apêndice B.2 apresenta a análise ROC; o Apêndice B.3 apresenta a *convex hull*; e o Apêndice B.4 apresenta a métrica escalar AUC.

B.1 Introdução

Comparar o desempenho de diferentes algoritmos de mineração de dados para um problema requer a realização de testes e validação (para inferir parâmetros de validade e confiabilidade nos modelos gerados), e o cálculo de estatísticas (para auxiliar a análise dos resultados). Considere o problema de se obter uma amostra representativa a partir de um conjunto de dados. Um método de divisão de dados reservaria certa quantidade para avaliação e usaria o restante para treinamento. Em termos práticos, se o conjunto de dados é representativo do domínio, é comum reservar um 1/3 dos dados para avaliação e os 2/3 restantes para treinamento. Deve-se garantir que a amostragem aleatória para obtenção do conjunto de treinamento e de avaliação seja estratificada. Uma maneira mais geral de avaliação consiste em repetir o processo completo de divisão várias vezes, nos conjuntos de treinamento e de avaliação, cada vez com diferentes amostras aleatórias estratificadas. Em cada iteração, uma proporção dos dados, em geral 9/10, é selecionada para treinamento e o restante, em geral 1/10, é usado para avaliação. Os índices de desempenho do modelo são obtidos como a média dos índices de desempenho nas diferentes iterações. Este processo é conhecido como validação cruzada de dez dobras (do inglês, *10-fold cross-validation*).

No caso de classificadores, é natural medir o desempenho em termos de erro e de sucesso na classificação. Em algumas situações práticas, o custo de uma classificação errada pode ser muito elevado e deve ser levado em consideração. Para classificação multiclasses, as classificações no conjunto de avaliação são apresentadas na forma de uma matriz bidimensional com uma linha e uma coluna para cada classe. Cada elemento desta matriz, chamada *matriz de confusão*, apresenta o número de casos avaliados, na qual a classe real é a linha e a classe predita é a coluna. Bons resultados são caracterizados por valores altos na diagonal principal e valores nulos para elementos fora da diagonal principal. Suponha que um classificador está sendo utilizado para classificar uma instância que possui apenas duas classes, *positivo* e *negativo*. A Tabela B.1 ilustra a nomenclatura utilizada para as quatro situações possíveis.

Tabela B.1: Matriz de Confusão com duas Classes.

		Classe Predita	
		positivo	**negativo**
Classe Real	**positivo**	Verdadeiro Positivo (TP)	Falso Negativo (FN)
	negativo	Falso Positivo (FP)	Verdadeiro Negativo (TN)

A partir da matriz de confusão, podem ser calculados índices de desempenho, dentre eles os apresentados na Tabela B.2, a seguir, onde N é o número de casos utilizados. Todos eles assumem valores no intervalo [0,1]. A sensibilidade é a *taxa de verdadeiro positivo* (*tpr*); a especificidade é a *taxa de verdadeiro negativo* (*tnr*) e a acurácia é a

probabilidade de acerto do classificador. As fórmulas da Tabela B.2 podem ser facilmente estendidas para mais de duas classes.

Tabela B.2: Índices para Discriminação entre Classificadores Dicotômicos.

Sensibilidade (S)	Especificidade (E)	Acurácia (Ac)	Erro Global
S = TP / (TP+FN)	E = TN / (TN+FP)	Ac = (TP+TN) / N	Erro = (FP + FN) / N

As métricas apresentadas na Tabela B.2 somente são interessantes quando o custo de erro de classificação das duas classes é igual. Considere, por exemplo, o erro de diagnóstico de se classificar um paciente doente como sadio. Esse erro pode custar a vida do paciente. Nesse domínio, o custo de erro de classificação da classe doente é muito superior à da classe sadio. A idéia é similar no domínio de desbalanceamento de classes, quando a classe de interesse é a classe minoritária. Um classificador enviesado para a classe majoritária poderá errar para todos os casos da classe minoritária e, mesmo assim, obter baixo índice de erro global. As métricas *F-measure* [Van Rijsbergen, 1979] e *G-mean* [Kubat, Holte & Matwin, 1998], por exemplo, tentam capturar esses problemas, penalizando os erros relativos a cada classe de forma separada. Entretanto, elas não são aplicáveis a problemas onde não se conhece os custos de erros para cada classe, ou quando não se conhece a melhor distribuição de classes a *priori* para a aprendizagem.

B.2 Análise ROC

A análise ROC (do inglês, *Receiver Operating Characteristics*), também conhecida como análise de curvas ROC, é uma técnica para a visualização, organização e seleção de classificadores baseadas em seus desempenhos. Esta análise vem sendo utilizada por longo tempo na teoria de detecção de sinais e pela comunidade médica para representar ganhos e perdas na escolha entre a taxa de detecção de acertos e a taxa de alarmes falsos providas por classificadores [Rakotomamonjy, 2004]. De acordo com Prati (2006), esta análise é também utilizada nas áreas de psicologia, economia e previsão do tempo e, recentemente (final dos anos 90), tem recebido maior atenção da comunidade de aprendizagem de máquina [Fawcett, 2004].

Um gráfico ROC é um gráfico bidimensional, no qual a taxa de detecção (*tpr*) é plotada no eixo das ordenadas e a taxa de falso alarme (*fpr*) é plotada no eixo das abscissas. Nesse gráfico, cada classificador avaliado é representado por um ponto e o ponto (1,0) representa o classificador perfeito e o ponto (0,1) representa o pior classificador possível. Portanto, um classificador X é melhor do que um classificador Y se, e somente se, X se encontrar acima e à esquerda de Y [Fawcett, 2004]. Classificadores próximos à diagonal principal são considerados estocásticos por possuírem taxas de acertos e de erros semelhantes. A Figura B.1, a seguir, apresenta um gráfico ROC para cinco classificadores, rotulados de A a E. Dessa figura, pode-se inferir que D é o melhor classificador, C tem desempenho aleatório e E é o pior classificador.

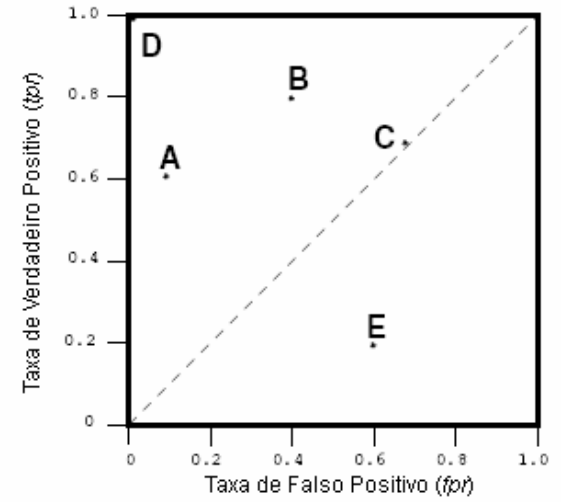

Figura B.1: Gráfico ROC com cinco classificadores binários [Fawcett, 2004].

Os classificadores apresentados na Figura B.1 são classificadores discretos, pois predizem apenas um resultado binário para a classe. Outros classificadores, como o *Naive* Bayes e Redes Neurais, produzem, para cada classe, uma probabilidade ou escore, que indica o grau de pertinência de um caso à classe positiva e negativa. Se o valor predito para a classe positiva for maior que um determinado limiar (geralmente 0.5), a classe predita pelo classificador é positiva (negativa, caso contrário). A avaliação destes classificadores com o limiar em 0.5 equivale a considerar um custo de erro de classificação uniforme, ou mesmo, considerar que a distribuição real de classes é balanceada [Lachiche & Flach, 2003]. Uma forma de se obter diferentes pontos no gráfico ROC, e com eles construir uma curva, é variar o limiar entre $(-\infty,+\infty)$ e, a cada passo, avaliar o modelo e plotar os pontos resultantes. Fawcett (2004) apresenta um método mais eficiente de se calcular esse pontos. A curva resultante da união desses pontos pode ser vista como várias avaliações do mesmo classificador utilizando-se diferentes custos ou diferentes distribuições de classe a *priori* [Prati, 2006]. Essa é uma característica importante, quando se necessita comparar dois ou mais classificadores em todo o espectro de custos de erro de classificação ou de distribuição de classe [Fawcett, 2004]. A Figura B.2, a seguir, apresenta um caso onde um classificador é melhor que outro independente do limiar utilizado.

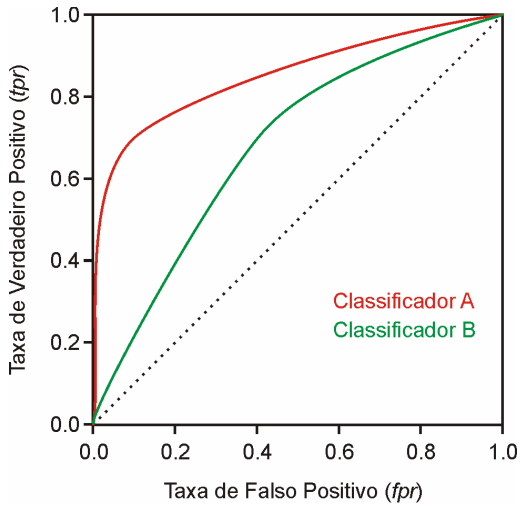

Figura B.2: Gráfico ROC com Dominância de Curva.

B.3 Convex Hull

Convex hull (casca convexa) é a envoltória dos pontos das curvas dos classificadores em um gráfico ROC. A Figura B.3, a seguir, apresenta um exemplo de uma *convex hull*. Um classificador é potencialmente ótimo se, e somente se, sua curva ROC se sobrepõe à *convex hull* [Barber *et al*., 1993] apud [Fawcett, 2004]. O gráfico ROC apresentado na Figura B.3 não possui classificadores potencialmente ótimos. Já no gráfico ROC da Figura B.2, a curva do classificador A se sobrepõe à *convex hull* e, portanto, ele é um classificador potencialmente ótimo. A existência de um classificador potencialmente ótimo garante que os demais classificadores presentes no mesmo gráfico ROC possam ser seguramente descartados [Fawcett, 2004].

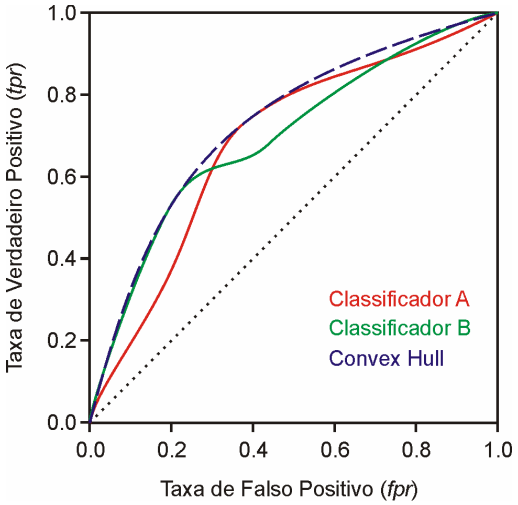

Figura B.3: Convex Hull.

B.4 Área sob a Curva ROC

A área sob a curva ROC (do inglês, *Area Under the ROC Curve*) poder ser utilizada como um estimador adimensional de desempenho de classificadores. Ela varia entre [0,1], mas na prática apenas se considera o intervalo [0.5, 1]. AUC facilita a comparação de classificadores, mas não substitui a análise de curvas ROC. Considere o gráfico ROC apresentado na Figura B.4, a seguir. O classificador A possui AUC maior do que o classificador B e, considerando apenas AUC como métrica de qualidade, o classificador A seria considerado melhor do que o classificador B. Entretanto, analisando a curva do classificador A, percebe-se que ele tem desempenho melhor somente enquanto a taxa de falso positivo (*fpr*) é menor do que 0.4. Para valores *fpr* maiores do que 0.4, o classificador B tem desempenho melhor.

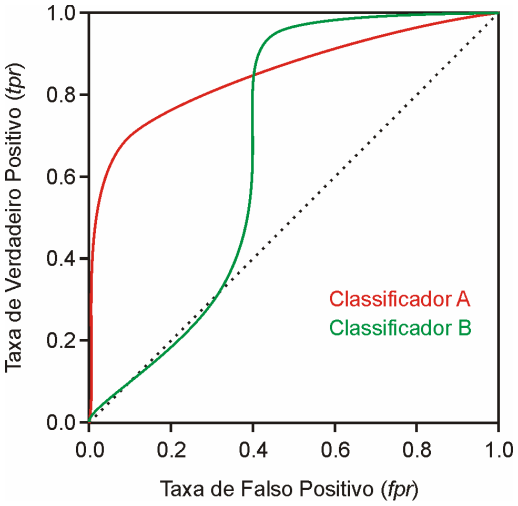

Figura B.4: Gráfico ROC com Intersecção de duas Curvas.

Apêndice C Clusterização

C.1 K-means

Neste método, cada caso é visto como um ponto no plano cartesiano. Visto desta forma, o método *k-means* consiste em dividir a base de dados em *k* subconjuntos de casos mais próximos. Inicialmente, *k* casos da base de dados são eleitos, de forma aleatória, para serem os centróides c_1, c_2, ..., c_k de cada *cluster* cl_1, cl_2, ..., cl_k, respectivamente. Um centróide é o ponto central de um *cluster*. Um caso *x* pertencerá a um *cluster* cl_i se, e somente se, a distância $d(x, c_i)$ for menor do que a distância $d(x, c_m)$ $\forall m$, $m \neq i$. Depois de agrupado todos os casos em seus respectivos *clusters*, o centróide de cada *cluster* é recalculado. Para cada *cluster* cl_i, a média aritmética de seus casos é atribuída ao seu centróide c_i. Desta forma, cada *cluster* cl_i tem seu centróide c_i deslocado para o centro geométrico de seus casos. Novamente, todos os casos da base de dados são reagrupados ao *cluster* com centróide mais perto. Este processo é repetido até que não haja mais mudanças em nenhum valor dos *k* centróides.

Algoritmo 10 O algoritmo *k-means* agrupa os casos da base de dados de entrada em *k clusters* e retorna os *clusters* formados.

Descrição das variáveis:
S: conjunto de dados de entrada;
k: quantidade de *clusters* desejada;

k-means
Entrada: *S*, *k*
Saída: conjunto dos c*lusters* formados

```
 1. Eleja k casos de S de forma aleatória para serem os centróides;
 2. Enquanto houver mudança nos centróides faça {
 3.   Para cada caso x_i de S faça {
 4.     Insira x_i no cluster cujo centróide esteja mais perto;
 5.   }
 6.   Para cada cluster Cl_i faça {
 7.     Atualize o centróide do cluster Cl_i com a média de seus casos;
 8.   }
 9. }
10. Retorne o conjunto dos clusters formados.
```

C.2 Squeezer

O método *Squeezer* [He, Xu & Deng, 2002] é utilizado para a clusterização de dados categóricos. Intuitivamente, *Squeezer* lê um caso da base de dados por vez e, ao ler o primeiro caso, este é utilizado para formar o primeiro *cluster*. Os demais casos lidos, ou são colocados nos *clusters* existentes, ou são utilizados para formar um novo *cluster*. Essa decisão é tomada com base na similaridade entre o caso lido e os *clusters* existentes. Se a maior similaridade entre o caso e os *clusters* existentes for menor do que certo limiar, o caso lido será colocado no *cluster* de maior similaridade a ele. Se, por outro lado, essa maior similaridade for menor do que o limiar adotado, um novo *cluster* é formado com esse caso. O processo se repete para todos os casos da base de dados. A similaridade entre um caso *c* e um *cluster Cl* é computada da seguinte forma: Seja $A_i(x)$ o valor do atributo A_i no caso *x* e $\{A_i(x_i)\}$ o conjunto dos valores do atributo

A_i dos casos $x_i \in Cl$. A similaridade entre x e Cl é a soma das freqüências de $A_i(x)$ em $\{A_i(x_i)\}$.

Algoritmo 11 O algoritmo *Squeezer* clusteriza os casos de base de dados com atributos categóricos.

Descrição das variáveis:
S: conjunto de dados de entrada;
limiarSimilaridade: limiar de similaridade;
simMax: maior similaridade entre um caso e um *cluster*;
$\{A_i(x_i)\}$: conjunto dos valores do atributo A_i dos casos $x_i \in Cl$;

Squeezer
Entrada: *S*, sim
Saída: conjunto dos *clusters* formados

```
 1. Retire o primeiro caso de S e forme o primeiro cluster com ele;
 2. Para cada caso x_i de S faça {
 3.   Para cada cluster Cl_m faça {
 4.     Calcule a similaridade entre x_i e Cl_m;
 5.   }
 6.   simMax ← maior similaridade calculada;
 7.   m ← índice do cluster com maior similaridade a x_i;
 8.   Se simMax ≥ limiarSimilaridade faça {
 9.     Insere x_i no cluster Cl_m;
10.   Senão {
11.     Cria um novo cluster com x_i;
12.   }
13. }
14. Retorna o conjunto dos clusters formados.
```

similaridade
Entrada: x_i, *Cl*
Saída: similaridade entre x_i e *Cl*

```
1. sim ← 0;
2. Para cada valor a_i do atributo A_i de x_i faça {
3.   sim ← sim + freqüência de a_i em {A_i(x_i)}, c_i ∈ Cl;
4. }
5. Retorna sim.
```

C.3 CEBMDC

He, Xu & Deng, (2005) propõem esse *framework* para clusterizar dados com atributos categóricos e contínuos. O *framework* CEBMDC (*Cluster Ensemble Based Mixed Data Clustering*) pode ser caracterizado como uma metaclusterização dos *clusters* para os atributos categóricos e contínuos feitos de forma separada. Primeiramente, a base de dados de entrada é separada em dois subconjuntos, um contendo os atributos categóricos e o outro os atributos contínuos. Em seguida, esses dois subconjuntos são clusterizados com qualquer método de clusterização desejado que retorne um vetor de valores indicando a qual *cluster* cada caso pertence. Finalmente, esses dois vetores são utilizados para construir uma nova base de dados, cada vetor representando um atributo, sendo esta base clusterizada com algum método de clusterização para atributos categóricos. Essa última clusterização é chamada de metaclusterização e o vetor de valores resultante indica a qual *cluster* cada caso pertence. A Figura B.1 sumariza o funcionamento do CEBMDC.

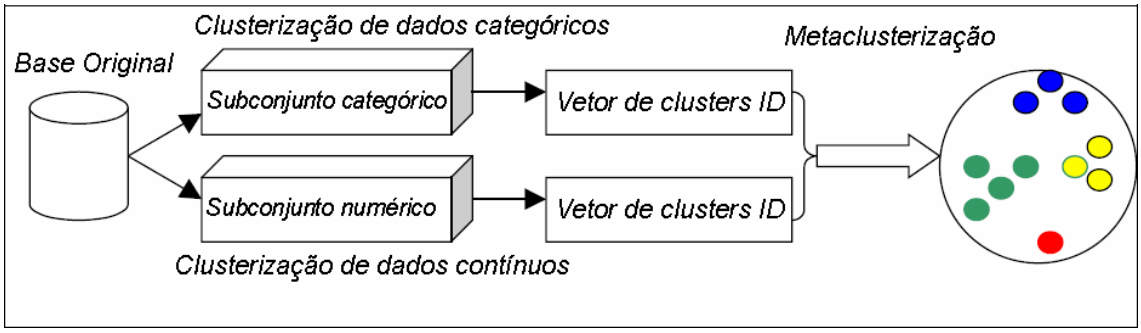

Figura C.1: Funcionamento do *framework* de clusterização CEBMDC.

Apêndice D Métricas de Distância

A distância euclidiana entre os pontos x e y pertencentes ao espaço Euclidiano $\Re^n$ é dada pela fórmula:

$$d = |\vec{x} - \vec{y}| = \sqrt{\sum_{i=1}^{n} |x_i - y_i|^2}$$

Como a distância euclidiana é aplicável apenas a variáveis numéricas e/ou ordinais, foram propostas outras métricas derivadas para o caso de atributos nominais. Uma primeira tentativa para aplicação da distância euclidiana a atributos nominais consiste na aplicação da métrica de sobreposição, onde se atribui o valor 0 quando os valores comparados são iguais, e 1 em caso contrário. Para atributos nominais, Stanfill (1986) *apud* Batista *et al.* (2004) utilizam uma modificação da distância euclidiana, denominada VDM (do inglês, *Value Difference Metric*), na qual se leva em consideração a similaridade entre os casos (Wilson & Martinez, 2000). Para bases de dados com atributos nominais e ordinais, Batista *et al*. (2004) utilizam uma métrica mais elaborada, chamada de HVDM (do inglês, *Heterogeneous Value Difference Metric*), proposta por Wilson e Martinez (2000). A distância entre dois casos é calculada da seguinte forma: para atributos ordinais, é função da soma das distâncias euclidiana *normalizadas por um fator da variância entre os atributos* (os autores utilizam o valor 4 para esse fator), e, para atributos nominais, é função da distribuição de freqüência relativa por classe dos atributos. A equação (1), a seguir, partida em duas equações para melhor compreensão, apresenta o cálculo da distância HVDM:

$$HVDM(x,y) = \sqrt{\sum_{a=1}^{m} d_a{}^2(x_a, y_a)} \qquad d_a(x,y) = \begin{cases} 1, \text{ se x ou y é desconhecido} \\ vdm_a(x,y), \text{ se a é nominal} \\ \dfrac{|x-y|}{4\sigma_a}, \text{ se a é numérico} \end{cases} \quad (1)$$

onde σ_a é o desvio padrão dos valores do atributo *a*.

A distância HVDM, como mostrado anteriormente, utiliza o cálculo da distância VDM, que é apresentada na equação (2) a seguir.

$$vdm_a(x,y) = \sum_{c=1}^{C} \left(\frac{N_{a,x,c}}{N_{a,x}} - \frac{N_{a,y,c}}{N_{a,y}} \right)^2 \quad (2)$$

onde:

$N_{a,x}$: freqüência que o valor *x* ocorre no atributo *a*;
$N_{a,x,c}$: freqüência que o valor *x* ocorre no atributo *a* para os casos da classe *c*;
C: quantidade total de casos.