Luis Eduardo Boiko Ferreira

UMA ABORDAGEM BASEADA EM FUNÇÕES DE REGRESSÃO PARA TRATAMENTO DE FLUXOS CONTíNUOS DE DADOS DESBALANCEADOS

Curitiba

2018

Luis Eduardo Boiko Ferreira

UMA ABORDAGEM BASEADA EM FUNÇÕES DE REGRESSÃO PARA TRATAMENTO DE FLUXOS CONTÍNUOS DE DADOS DESBALANCEADOS

Dissertação de Mestrado apresentado ao programa de Pós-Graduação em informática da Pontifícia Universidade Católica do Paraná como requisito parcial para obtenção do título de mestre em Informática.

Pontifícia Universidade Católica do Paraná

Programa de Pós-Graduação em Informática

Orientador: Fabrício Enembreck

Curitiba

2018

Dados da Catalogação na Publicação
Pontifícia Universidade Católica do Paraná
Sistema Integrado de Bibliotecas – SIBI/PUCPR
Biblioteca Central
Giovanna Carolina Massaneiro dos Santos CRB 9/1911

Ferreira, Luis Eduardo Boiko
F383a Uma abordagem baseada em funções de regressão para tratamento de
2018 fluxos contínuos de dados balanceados / Luis Eduardo Boiko Ferreira ;
orientador: Fabrício Enembreck. – 2018.
96 f. : il. ; 30 cm

Dissertação (mestrado) – Pontifícia Universidade Católica do Paraná,
Curitiba, 2018
Bibliografia: f. 89-96

1. Processamento de dados. 2. Mineração de dados (Computação).
3. Algoritmos I. Enembreck, Fabrício. II. Pontifícia Universidade Católica do
Paraná. Programa de Pós-Graduação em Informática. III. Título.

CDD 20. ed. – 004

Pontifícia Universidade Católica do Paraná
Escola Politécnica
Programa de Pós-Graduação em Informática

PUCPR
GRUPO MARISTA

ATA DE SESSÃO PÙBLICA

DEFESA DE DISSERTAÇÃO DE MESTRADO Nº 02/2018

PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA – PPGIa
PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ - PUCPR

Em sessão pública realizada às 14h00 de **22 de Fevereiro de 2018**, no Auditório Guglielmo Marconi – Bloco 8, ocorreu a defesa da dissertação de mestrado intitulada **"Uma Abordagem baseada em Funções de Regressão para Tratamento de Fluxos contínuos de Dados Desbalanceados"** apresentada pelo aluno **Luis Eduardo Boiko Ferreira**, como requisito parcial para a obtenção do título de **Mestre em Informática**, na área de concentração **Ciência da Computação**, perante a banca examinadora composta pelos seguintes membros:

Prof. Dr. Fabricio Enembreck (Orientador)- PUCPR

Prof. Dr. Alceu de Souza Britto Junior – PUCPR

Prof.ª Dr.ª Deborah Ribeiro Carvalho – PUCPR/PPGTS

Após a apresentação da dissertação pelo aluno e correspondente arguição, a banca examinadora emitiu o seguinte parecer sobre a tese:

Membro	Parecer	
Prof. Dr. Fabricio Enembreck	(X) Aprovado	( ) Reprovado
Prof. Dr. Alceu de Souza Britto Junior	(X) Aprovado	( ) Reprovado
Prof.ª Dr.ª Deborah Ribeiro Carvalho	(X) Aprovado	( ) Reprovado

Portanto, conforme as normas regimentais do PPGIa e da PUCPR, a tese foi considerada:

(X) **APROVADO**
(aprovação condicionada ao atendimento integral das correções e melhorias recomendadas pela banca examinadora, conforme anexo, dentro do prazo regimental)

( ) **REPROVADO**

E, para constar, lavrou-se a presente ata que vai assinada por todos os membros da banca examinadora.
Curitiba, 22 de Fevereiro de 2018.

Prof. Dr. Fabricio Enembreck

Prof. Dr. Alceu de Souza Britto Junior

Prof.ª Dr.ª Deborah Ribeiro Carvalho

Programa de Pós-Graduação em Informática PPGIa PUCPR

Rua Imaculada Conceição 1155 Prado Velho CEP 80215-901 Curitiba Parana Brasil
Tel.: (41) 3271 1669 Fax.: (41) 3271 2121 www.ppgia.pucpr.br

Agradecimentos

Ao meu orientador, Prof. Fabrício Enembreck, que me deu a oportunidade de ingressar na pesquisa. Durante estes dois anos, foram muitas conversas, ideias e críticas, todas fundamentais para o meu crescimento no âmbito pessoal e profissional. Apesar de ser graduado em matemática, as sugestões do Prof. Fabrício referente as disciplinas a serem cursadas, tal como artigos a serem lidos, foram pontuais para preencher as lacunas existentes em minha formação. Além da formação de base, a produção acadêmica sempre foi incentivada, auxiliando sempre nas revisões. Agradeço imensamente pela confiança depositada em mim nestes anos de trabalho.

Agradeço a minha esposa Bárbara, que sempre me incentivou a buscar meus sonhos. Ela fez parte do planejamento e execução de todas as grandes conquistas que obtivemos ao longo da última década. Agradeço pelas incontáveis conversas sobre a qualificação, artigos e mais recentemente, a defesa. Sem ela, nada disso teria sentido. Agradeço também a minha filha Sarah, que assim como eu, sofreu com a distância física, imposta pelo mestrado, mas nunca deixou de me incentivar. Aos meus pais, Olmyr e Iara, que forneceram todo o subsídio necessário para que eu seguisse com meus estudos, e finalmente ao meu irmão, pelo companheirismo e amizade, que foram muito importantes nestes últimos anos.

Em especial aos meus amigos e colegas de laboratório Heitor Murilo Gomes e Jean Paul Barddal, que me ensinaram muito do que sei hoje. Agradeço pela paciência, dedicação, disponibilidade, orientações e parceria. Tive uma sorte imensa em tê-los por perto durante esta caminhada.

Aos professores Edson Emílio Scalabrin e Deborah Ribeiro Carvalho, que forneceram observações importantes que contribuíram para reflexões e o enriquecimento deste trabalho, e por fim, a Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelo financiamento.

Imagination will often carry us
to worlds that never were.
But without it we go
nowhere - Carl Sagan

Resumo

Uma das maiores dificuldades para a classificação em *streams* de dados do mundo real é a distribuição desbalanceada de instâncias entre as classes. Em algumas áreas, o desbalanceamento de dados é intrínseco: a natureza do problema faz com que alguns rótulos de classe sejam sobre-representados; enquanto em outros é extrínseco: o desbalanceamento não está relacionado com a natureza do conjunto de dados, como por exemplo, se o conjunto de dados estiver incompleto. Em ambos os casos, os algoritmos tradicionais de classificação são normalmente incapazes de induzir modelos úteis, uma vez que comumente trabalham sob a suposição de que os rótulos de classe são uniformemente distribuídos. Ao longo dos anos, uma variedade de técnicas de re-amostragem foi desenvolvida com a finalidade de prover uma distribuição de dados mais equilibrada, porém, as técnicas geralmente induzem a alguns problemas, como perda de informação, sobre-ajuste, geração incorreta de valores para alguns tipos de atributos e alta complexidade computacional, o que dificulta sua aplicação em contextos de *streams*. Além dos problemas supracitados, tratando-se de desbalanceamento intrínseco, as técnicas que geram instâncias sintéticas não conseguem aumentar a representatividade da classe minoritária no espaço de características. Neste trabalho, o método de re-amostragem Regression Functions Approach for *oversampling* (RFAO) é apresentado. Este método gera instâncias sintéticas seguindo o fluxo natural dos dados, garantindo que a geração ocorra em um lugar seguro no espaço de características. Ao contrário dos demais algoritmos da literatura, a abordagem proposta se baseia em atributos, o que possibilita a geração de valores via funções de regressão e probabilidade estatística, além da redução da complexidade computacional, se comparado a abordagens tradicionais baseadas em instâncias. Estudos empíricos mostram que a utilização do algoritmo RFAO proporciona melhora na performance de classificação em instâncias minoritárias, tanto em *streams* reais com desbalanceamento intrínseco quanto em *streams* simuladas.

Palavras-chave: mineração de fluxos contínuos de dados; desbalanceamento; re-amostragem.

Abstract

One of the biggest challenges for classification in real-world data streams is the imbalanced distribution of instances amongst class labels. In some areas, data imbalance is intrinsic: the problem domain causes some class labels to be overrepresented; while in others it is extrinsic: the imbalance is not related to the nature of the data set, like for instance, if the data set is incomplete. In both cases, standard classification algorithms are unable to induce useful models from imbalanced data sets since they usually work under the assumption that class labels are uniformly distributed. Throughout the years, a multitude of sampling techniques were developed to balance the class label distribution, however the techniques generally induce some problems, such as information loss, overfitting, incorrect generation of values for some types of attributes and high computational costs, which makes it difficult to apply in streams contexts. In addition to the aforementioned problems, in intrinsic imbalance problems, the techniques that generate synthetic instances can not increase the representativeness of the minority class in the feature space. In this work, the re-sampling method Regression Functions Approach for *oversampling* (RFAO) is proposed. This method generate synthetic instances following the natural flow of data, ensuring that the generation occurs in a safe place in the feature space. This approach allows the generation of values through regression functions and probability, as well as the reduction of computational complexity, if compared to traditional approaches based on instances. Empirical studies show that the use of the RFAO algorithm provides improved classification performance in minority instances, both in real streams with instrinsic imbalance and in simulated streams.

Keywords: data stream mining. class imbalance. resampling.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

AdaBoost	Adaptive Boosting
AM	Aprendizagem de máquina
AUROC	Area under the Receive Operating Characteristics curve
BEV	Bagging ensemble variation
CC	Cluster Centroids
DT	Decision Tree
HT	Hoeffding Tree
KNN	K-Nearest Neighbors
NIE	Nonstationary and Imbalanced Environments
NSE	Nonstationary Environments
MNS	Majority Noise Set
MOA	Massive On-line Analysis
MUTE	Majority Under-sampling Technique
NB	Naive Bayes
NM1	NearMiss-1
NM2	NearMiss-2
NM3	NearMiss-3
PSS	Parallel selective sampling
RFAO	Regression Functions Approach for Oversampling
RLS	Regressão linear simples
ROC	Receive Operating Characteristics curve
RUS	Random Undersampling
SERA	Selectively Recursive Approach algorithm

SMBL1	SMOTE-Borderline-1
SMBL2	SMOTE-Borderline-2
SMOTE	Synthetic Minority *oversampling* Technique
SMTK	SMOTE-TOMEK-LINKS
STDEV	Desvio padrão
SVM	Support Vector Machines
TK	Tomek Links
UCB	Uncorrelated Bagging

Lista de símbolos

$\mathcal{S}$	Fluxo contínuo de dados
S_{min}	Classe minoritária
S_{maj}	Classe majoritária
f_j	Atributo referente ao índice j
$\vec{x}_i$	Instância referente ao índice i
$\mathcal{W}$	Janela deslizante
$S_{balanced}$	Fluxo contínuo de dados balanceados
δ	Proporção de balanceamento esperada
λ	Correlação esperada
L_b	Lista de atributos binários
P_b	Tupla representando a probabilidade de ocorrência de cada valor do atributo binário
L_n	Lista de atributos normais
L_np	Lista de atributos não-normais
L_cn	Lista de combinações, dois a dois, de todos os atributos normais
L_cnp	Lista de combinações, dois a dois, de todos os atributos não-normais
p	Coeficiente de correlação de Pearson
s	Coeficiente de correlação de Spearman
L_{fp}	Lista final com atributos que possuem correlação de Pearson $\geq$ a esperada
L_{fs}	Lista final com atributos que possuem correlação de Spearman $\geq$ a esperada
$C_{(L_{fp[i]},2)}$	Combinação linear, dois a dois, entre os atributos de L_{fp}
$C_{(L_{fs[i]},2)}$	Combinação linear, dois a dois, entre os atributos de L_{fs}
L_{cn}	Lista das combinações retornadas de $C_{(L_{fp[i]},2)}$

L_{cnp}	Lista das combinações retornadas de $C_{(L_{fs[i]},2)}$
$L_{fdanger}$	Lista de atributos que não possuem correlação
Fc_{tupla}	Função de regressão obtida com a tupla
P_{min}	Par ordenado mínimo
P_{max}	Par ordenado máximo
$D_{(p_{min},p_{max})}$	Distancia euclidiana entre os pares ordenados máximo e mínimo
T_{extra}	Proporção de expansão no espaço de características
$N_{x_{max}}$	Nova coordenada x para o ponto máximo
$N_{y_{max}}$	Nova coordenada y para o ponto máximo
$N_{x_{min}}$	Nova coordenada x para o ponto mínimo
$N_{y_{min}}$	Nova coordenada y para o ponto mínimo
N_{pmax}	Novo ponto máximo
N_{pmin}	Novo ponto mínimo
ψ	Número de instâncias a ser gerado
ϖ	Tamanho da janela deslizante
C_{min}	Classe minoritária
C_{maj}	Classe majoritária
$L_{atributos}$	Lista com os atributos atuais na *stream*
L_{rls}	Dicionário que associa um par de atributos e a correlação
L_{val}	Lista com os valores possíveis para dado atributo nominal
L_{prob}	Lista com as probabilidades de ocorrência dos valores de L_{val}
$\vec{x}_{new}$	Nova instância gerada sintéticamente

Sumário

1 Introdução

O desenvolvimento recente da tecnologia e da ciência permitiu uma alta taxa de crescimento no volume dos dados tal como aumentou sua disponibilidade. De 2005 a 2020, espera-se que o universo digital crescerá por um fator de 300, de 130 exabytes para 40.000 exabytes, ou 40 trilhões de gigabytes (mais de 5.200 gigabytes por cada homem, mulher e criança) (GANTZ; REINSEL, 2012). Parte deste crescimento se deve ao grande volume de dados gerados por redes de computadores, *smartphones*, *wearables* além de uma vasta gama de sensores, que produzem dados em tempo real. Todos esses dados só são úteis se puderem ser processados eficientemente, de modo que indivíduos possam tomar decisões oportunas com base neles (GOMES, 2017).

Dados brutos não permitem identificação rápida de padrões de comportamento, e para tal, técnicas de aprendizagem de máquina (AM) são amplamente difundidas. Sua utilização ocorre em diversas áreas, do cotidiano de pessoas comuns à segurança nacional (HE; GARCIA, 2009). A aplicação de técnicas de AM na solução de problemas, além da eficiência em gerar modelos, deve-se ao fato de que pessoas são propensas a cometerem erros ao fazer certas análises ou, possivelmente, ao tentarem encontrar relações entre múltiplas características, o que geralmente não ocorre com AM (KOTSIANTIS; ZAHARAKIS; PINTELAS, 2007).

AM é um subcampo da ciência da computação que dá aos computadores a capacidade de aprender sem ser explicitamente programado para isto (SAMUEL, 1959). Algoritmos de AM possuem duas abordagens possíveis: **supervisionada** e **não supervisionada**. Na abordagem não supervisionada, destaca-se a tarefa de **agrupamento**, em que o indutor analisa os exemplos recebidos, e busca agrupá-las seguindo algum critério (CHEESEMAN et al., 1988). Já na abordagem supervisionada, o algoritmo de aprendizado (indutor) recebe uma base de dados $\mathcal{S}$ com m exemplos. Define-se $\mathcal{S} = \{(x_i, y_i)\}, i = [1, m]$ aonde $x_i \in X$ corresponde a uma instância no espaço de características n-dimensional $X = \{f_1, f_2, f_3, ..., f_n\}$ e $y_i \in Y = \{c_1, c_2, c_3, ..., c_n\}$ representa o rótulo de classe associado a x_i (HE; GARCIA, 2009). Caso o rótulo de classe a ser previsto seja **discreto**, têm-se um problema de **classificação**. Em oposição a este cenário, caso os rótulos de classe sejam **contínuo**, o problema em questão é de **regressão**.

Com bases de dados cada vez maiores, o limitador físico de processamento começa a ficar evidente, uma vez que, de modo tradicional, a base de dados é carregada na memória do computador com a finalidade de ser processada. Algoritmos tradicionais de AM (*batch*) podem ser capazes de processar bases de dados grandes, caso a mesma seja subdividida em bases menores, que serão processadas individualmente. Ainda sim, tais algoritmos não

conseguem processar um fluxo contínuo de dados. No cenário *batch* um modelo previamente induzido não é atualizado à medida que novas instâncias chegam. Existem situações aonde esta limitação não é desejada (BIFET et al., 2010). Neste contexto, o paradigma de *data streams* emergiu, postulando que bases de dados muito maiores do que a memória do computador poderiam ser processadas, através do processamento de um fluxo contínuo de dados. O princípio básico é que o algoritmo indutor não tem controle sobre a ordem em que os exemplos chegam, deve inspecionar cada exemplo uma única vez e depois descartá-lo, afim de liberar memória, e por fim, deve ser capaz de atualizar seu modelo de maneira incremental a cada nova instância processada, propiciando que a cada momento o indutor possa ser utilizado (BIFET et al., 2010).

Diversos fatores podem comprometer a performance de um classificador, tanto em cenários *batch* quanto em *stream*. Estes fatores geralmente incluem problemas relacionados à complexidade, separabilidade, dimensionalidade e desbalanceamento.

O foco deste trabalho está no problema de classificação envolvendo desbalanceamento, no contexto de *streams*. A importância do tratamento dos dados desbalanceados se deve ao fato de que algoritmos tradicionais de classificação tendem a focar nas instâncias com maior representatividade, e com isso acabam negligenciando instâncias minoritárias (GONG; KIM, 2017). Este problema fica ainda mais evidente no contexto de *streams*, pois de modo antagônico ao cenário *batch*, as instâncias são observadas individualmente, o que leva a uma observação inabitual de instâncias minoritárias. Tal fato pode impedir ou postergar a descoberta de quaisquer padrões existentes nesta classe. Além dos problemas supracitados, *streams* podem enfrentar mudanças de conceito ao longo de sua existência, o que poderia causar a alternância entre a classe majoritária e minoritária. Como aplicações de problemas envolvendo dados desbalanceados, elenca-se: doenças raras (VO; WON, 2007), fraudes de cartões de crédito (VO; WON, 2007), derramamentos de óleo no oceano (TOPOUZELIS, 2008), classificação espectral de raios gama (BELLINGER; JAPKOWICZ; DRUMMOND, 2015) e empréstimo pessoal entre pessoas (VEDALA; KUMAR, 2012; TSAI; RAMIAH; SINGH, 2014; MALEKIPIRBAZARI; AKSAKALLI, 2015; KUMAR et al., 2016).

Neste trabalho, o algoritmo *Regression Functions Approach for Oversampling* (RFAO) é apresentado. Este algoritmo foi desenvolvido para efetuar re-amostragem em fluxos contínuos de dados, respeitando as limitações de tempo de processamento impostas pelo ambiente de *streams*. Ainda, o algoritmo apresentado supera algumas das limitações encontradas nos principais algoritmos da literatura, como incapacidade de gerar valores em faixas corretas e aumento da representatividade da classe minoritária no espaço de características.

1.1 Objetivos

O objetivo principal do presente trabalho é o desenvolvimento de uma estratégia de re-amostragem baseada em funções de regressão para o problema de classificação supervisionada binária em *streams* de dados. Com a finalidade de cumprir o objetivo principal previamente estabelecido, os seguintes objetivos específicos foram definidos:

1. Criar uma classificação para as estratégias de re-amostragem existentes;
2. Definir e analisar o algoritmo de re-amostragem proposto;
3. Avaliar os resultados obtidos;

O objetivo 1 tem como finalidade situar o novo algoritmo proposto em relação aos existentes na literatura. O objetivo 2 foi definido para que a formalização teórica seja realizada, tal como a análise do algoritmo proposto. Por fim, no objetivo 3 definiu-se que uma avaliação empírica seja realizada, utilizando *streams* estacionárias reais e simuladas.

1.2 Motivação

Com a crescente utilização de técnicas de AM nos mais diversos cenários, problemas envolvendo dados desbalanceados estão cada vez mais frequentes e relevantes. Apesar de existirem várias soluções que se propõe a resolver o problema de desbalanceamento, grande parte delas foi desenvolvida para o ambiente *batch*, o que fornece capacidades limitadas para funcionar corretamente em ambiente de *streams*. Além do problema supracitado, a maioria das soluções existentes acabam induzindo uma relação inversa entre aumento da classificação correta das instâncias da classe minoritária com o aumento da classificação correta das instâncias da classe majoritária (HE; GARCIA, 2009). Tal comportamento se deve ao fato de estratégias de re-amostragem serem desenvolvidas com foco primário em aumentar a performance do classificador na classe com menor representatividade, tomando pouco ou nenhum cuidado em não comprometer a performance na classe majoritária, o que não é desejável em cenários aonde a classificação correta de instâncias é crítica para ambas as classes.

A motivação deste trabalho é tentar sobrepujar as limitações apontadas, inicialmente, apresentando uma maneira de balancear os dados através da geração de instâncias sintéticas que sigam o fluxo natural dos dados, fornecendo parâmetros para que o usuário adapte o método ao cenário pretendido.

1.3 Hipótese

Hipótese. *O algoritmo de re-amostragem proposto é capaz de aumentar a representatividade da classe minoritária no espaço de características, o que melhora a um nível competitivo a classificação correta de instâncias pertencentes a classe minoritária sem comprometer a performance na classe majoritária.*

A hipótese está baseada na ideia que aumentando a representatividade da classe minoritária no espaço de características, através de funções de regressão, exista uma melhora na taxa de classificação correta das instâncias pertencentes a classe minoritária, pois este processo pode possibilitar a descoberta de novos e interessantes padrões de classificação.

1.4 Resultados esperados e contribuição

Espera-se que este trabalho contribua para o avanço sobre o entendimento do problema de desbalanceamento em *streams* de dados, tal como forneça um método de re-amostragem destinado a este ambiente. Concretamente, foram coletadas evidências empíricas, sustentadas por testes estatísticos, de que o método proposto aumenta a performance de classificação tanto em *streams* reais quanto simuladas.

1.5 Organização

O presente trabalho está organizado da seguinte forma: o Capítulo 2 traz em detalhes uma discussão sobre a mineração de fluxos contínuos de dados e problema do desbalanceamento, tal como abordagens utilizadas para amenizar o impacto do desbalanceamento no contexto de fluxo contínuo de dados e *batch*, além de métricas de avaliação de performance adequadas na presença do desbalanceamento, também, para os dois cenários. O Capítulo 3 apresenta o método de re-amostragem proposto, trazendo um estudo de caso e discussões sobre os pontos fortes e fracos do método. O Capítulo 4 traz uma série de experimentos conduzidos a fim de testar o método proposto e compará-lo com alguns métodos concorrentes. O capítulo inicia definindo o protocolo experimental adotado, tal como estratégia de validação, justificativa na escolha dos classificadores e técnicas de re-amostragem utilizadas e um descritivo sobre bases de dados utilizadas nos experimentos. A seguir, os resultados são individualmente apresentados e discutidos. Por fim, no Capítulo 2 são apresentadas as conclusões do presente trabalho, seguido pelo capítulo final, onde encontram-se as referências dos trabalhos citados nesta pesquisa.

2 Revisão da literatura

Tanto no ambiente *batch* quanto no ambiente de *streams*, o objetivo da tarefa de classificação é prever um valor nominal (classe) de uma instância representada por um vetor de características. A diferença principal entre os dois ambientes se dá na etapa de treinamento, uma vez que em *batch* as instâncias estão disponíveis na forma de um *data set* contendo todos os exemplos, logo, o classificador tem acesso a todas as instâncias de treinamento antes de induzir o modelo. Já no ambiente de *streams*, as instâncias são apresentadas individualmente em um fluxo potencialmente infinito de dados.

As características do ambiente de *streams* requerem que os classificadores sejam capazes de processar um grande volume de dados e estejam prontos para predizer em todos os momentos. Alguns classificadores foram adaptados para o ambiente *stream*, como o Online Bagging (OZA, 2005), que é uma adaptação do algoritmo Bagging tradicional (BREIMAN, 1996), o que reforça a ideia de que abordagens de re-amostragem desenvolvidas para o ambiente *batch* podem tem suas ideias adaptadas para o ambiente de *streams*.

Nas seções seguintes, os conceitos fundamentais sobre mineração de *streams* de dados e desbalanceamento são apresentados. A seção 2.1 apresenta o processo de classificação em *streams* de dados. A seção 2.2 traz a definição formal do problema de desbalanceamento, seguido pelas seções 2.2.1, 2.2.2, 2.2.3 e 2.2.4, que apresentam alternativas para melhorar a performance de classificação na presença de desbalanceamento, sendo as duas ultimas seções dedicadas a trazer as técnicas de re-amostragem para *batch* e fluxos contínuos de dados, respectivamente. Por fim, as métricas e procedimentos de avaliação são apresentadas nas seções 2.3 e 2.4.

2.1 Mineração de fluxos contínuos de dados

Mineração de dados e AM são técnicas estabelecidas para o processamento de grandes quantidades de dados. Detecção de *spam* e sistemas de recomendação são exemplos de atividades executadas por algoritmos de AM presentes no dia-a-dia da maioria das pessoas. O *pipeline* usual da aplicação destas técnicas envolve retirar uma amostra dos dados, executar o pré-processamento para torna-los adequados para geração do modelo, treinar um modelo para a tarefa desejada, e finalmente utilizar este modelo para predição em produção (BIFET; MORALES, 2014). Uma das principais dificuldades encontra-se no fato deste *pipeline* necessitar de manutenção periódica para que o modelo esteja sempre atualizado.

Hoje em dia, a maioria dos dados é gerado na forma de uma *stream*. O modelo *batch*

pode naturalmente ser interpretado como um recorte dos dados de uma *stream*, obtidos em um dado intervalo de tempo. Um ambiente de aprendizado baseado em *streams* possui características diferentes do ambiente tradicional *batch*. O processo de classificação típico neste ambiente pode ser observado na Figura 1. As etapas deste processo são:

1. Processar uma instância por vez e inspeciona-la uma única vez (na maioria dos casos)
2. Executar o processamento da instância em uma quantidade limitada de tempo, atualizando o modelo quando necessário
3. Estar pronto para executar predições a qualquer momento

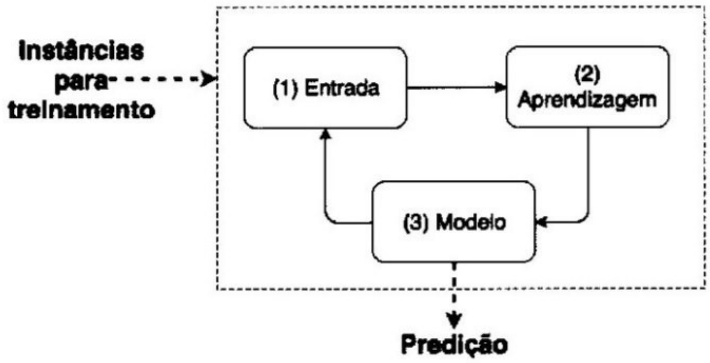

Figura 1 – Ciclo de classificação em *streams*, adaptado de (BIFET et al., 2010)

Formalmente, tem-se um fluxo contínuo de dados $\mathcal{S}$, que gera instâncias $\vec{x}_i$, com dimensionalidade d, de maneira serializada e potencialmente infinita: $\mathcal{S} = \{\vec{x}_1, \vec{x}_2, ..., \vec{x}_\infty\}$. Tal como no cenário *batch*, em *streams*, o problema de classificação refere-se ao problema de induzir uma função capaz de mapear instancias em uma de duas ou mais classes pré-definidas (GOMES, 2017).

Na maioria dos cenários de fluxos de dados, informações mais recentes tendem a possuir maior relevância, uma vez que podem trazer novos conceitos ou mudanças na distribuição dos dados (BARDDAL, 2015). A fim de solucionar este problema, abordagens baseadas em janelas móveis foram propostas, sendo estas: janela deslizante, janela *damped* e janela *landmark* (SILVA et al., 2013).

A abordagem de janela deslizante se baseia em reter apenas as instâncias mais recentes obtidas na *stream* em uma estrutura de tamanho fixo ou dinâmico $\mathcal{W}$. A fim de manter apenas os últimos registros, esta estrutura geralmente possui uma política de acesso FIFO (*first in first out*). Já a abordagem de janela *damped* associa pesos as instâncias recebidas, utilizando algum critério para que os mesmos decaiam com o passar do tempo. Por fim, a abordagem *landmark* baseia-se em reter uma janela de instâncias por um certo período, que pode ser determinado por um intervalo temporal ou um dado numero de instâncias, descartando todas as instâncias a cada ciclo.

Abordagens baseadas em janelas deslizantes, além de permitirem que o algoritmo de classificação foque-se nas instâncias mais recentes, possibilitam a aplicação de técnicas de re-amostragem. Uma vez que tais técnicas alteram a distribuição original dos dados antes de apresentá-los ao classificador, as janelas podem ser utilizadas como *mini-batches* de processamento. É possível também, através destes *mini-batches*, que algoritmos tradicionais de classificação sejam utilizados, porém, tal utilização poderia ser restritiva, uma vez que estes algoritmos não foram feitos para lidar com problemas oriundos do ambiente de *streams*, como mudanças de conceito (*concept drift*).

2.1.1 *Concept Drift*

Mudanças de conceito são frequentemente observadas na sociedade. Por exemplo, o conceito que certa população possui sobre determinado posicionamento político, pode ser afetado pelo aparecimento de novas informações sobre o mesmo. Outros exemplos de mudança de conceito incluem: a flutuação do mercado financeiro, a procura por determinados produtos ou o volume de vendas em determinadas épocas do ano, tal como a incidência de certas doenças, como a gripe. Todos estes exemplos são afetados pelo momento em que são observados (WIDMER; KUBAT, 1996).

Streams de dados são obtidos no mundo real, logo apresentam as mesmas características dinâmicos encontradas na realidade. Formalmente, as mudanças de conceito podem ocorrer de forma abrupta, incremental, gradual ou recorrente. Quando estas mudanças ocorrem de maneira abrupta, são facilmente perceptíveis, uma vez que os erros de predição aumentam, tal como ocorre uma variação na distribuição dos dados no espaço de características em um curto período de tempo (GOMES, 2017). Mudanças incrementais são difíceis de serem detectadas, uma vez que representam os conceitos que evoluem lentamente ao longo do tempo. Caso uma mudança leve certo tempo para tornar-se persistente, ela é dita gradual. Por fim, a mudança recorrente refere-se a conceitos que possuem variação com certa regularidade, por exemplo, o aumento de vendas em épocas, como o natal.

Algoritmos de re-amostragem agem na etapa de pré-processamento, modificando a distribuição dos dados, a fim de prover uma distribuição tão equilibrada quanto desejada para o classificador. Como o foco deste trabalho está na classificação binária, assume-se que ambas as classes possam alternar a predominância ao longo da *stream*, porém, o desaparecimento e surgimento de novas classes não foi estudado.

2.2 Desbalanceamento

Dados desbalanceados são caracterizados por possuírem muito mais instâncias pertencendo a certa classe (classe majoritária) do que outras (classes minoritárias). Seja a classe minoritária denotada como $S_{min} \in S$ e a classe majoritária como $S_{maj} \in S$,

tal que $S_{min} \cap S_{maj} = \emptyset$ e $S_{min} \cup S_{maj} = S$. Como as instâncias pertencentes a S_{min} ocorrem raramente, as regras para classificação destas classes tendem a ser raras, não são descobertas, ou ainda, são ignoradas (SUN; WONG; KAMEL, 2009). Em oposição a este cenário, S_{maj} está sobre-representada, o que impede o classificador de focar-se na classificação das instâncias de S_{min} (GONG; KIM, 2017). O desbalanceamento pode ser categorizado em: intrínseco e extrínseco. O desbalanceamento intrínseco ocorre quando a natureza do problema sugere a existência do desbalanceamento, como por exemplo, em bases de diagnóstico médico (VO; WON, 2007), fraudes de cartões de crédito (PANIGRAHI et al., 2009), derramamento de óleo no oceano (TOPOUZELIS, 2008), classificação espectral de raios Gamma (BELLINGER; JAPKOWICZ; DRUMMOND, 2015) entre outras. Nas bases de diagnóstico médico, por exemplo, o número de indivíduos saudáveis é sempre maior do que os que apresentam certa patologia. Este comportamento é esperado neste tipo de situação. Caso o desbalanceamento não seja esperado no problema em questão, e ele ainda sim ocorrer, ele é chamado de desbalanceamento extrínseco. Diversos fatores podem causar o desbalanceamento extrínseco, entre eles, a interrupção de um sensor que capte dados de uma fonte por certo período de tempo. No contexto de *data streams*, as relações entre classes não são permanentes. O desbalanceamento pode ocorrer por diversos fatores, como o surgimento de novas classes e o desaparecimento de outras (KRAWCZYK, 2016).

A fim de resolver o problema de desbalanceamento, existem algumas alternativas, como: combinação de classificadores (GALAR et al., 2012; WANG; MINKU; YAO, 2014), abordagens baseadas em custos (GHAZIKHANI; MONSEFI; YAZDI, 2013; ZHANG; TAN; REN, 2016) e técnicas de re-amostragem. Todas as técnicas de re-amostragem são conhecidas por trabalhar a nível dos dados. Já abordagens de combinação de classificadores é executada a nível de algoritmo. Por fim, abordagens baseadas em custos podem ser tanto baseada em algoritmos, quanto a combinação destas com abordagens baseadas em dados.

A sub-seção 2.2.1 irá apresentar o funcionamento do tratamento do desbalanceamento segundo a combinação de classificadores (*ensemble*), seguido pela seção 2.2.2, que traz a abordagem baseada em custos. Por fim, nas seções 2.2.3 e 2.2.4 serão abordadas as técnicas de re-amostragem para *batch* e fluxos contínuos de dados, respectivamente.

2.2.1 Combinação de classificadores

A ideia de utilizar a combinação de classificadores se baseia em construir vários classificadores a partir dos dados originais de S e depois agregar suas previsões ao classificar amostras desconhecidas. A principal motivação para combinar classificadores, é melhorar sua capacidade de generalização, pois cada classificador possivelmente irá cometer erros diferentes, supondo que cada classificador tenha sido treinado com um número limitado de dados.

O efeito da combinação de conjuntos de classificadores treinados na mesma porção de S, também é estudado em termos dos conceitos estatísticos de viés e variância. O viés pode ser caracterizado como uma medida de sua capacidade de generalizar corretamente para um conjunto de teste, enquanto a variância pode ser similarmente caracterizada como uma medida em que a previsão do classificador é sensível aos dados em que foi treinada. A variância é então associada a superposição: se o método causa problemas de sobreposição, as previsões para uma única instância variarão entre amostras (KAMEL; WANAS, 2003). A melhora no desempenho decorrente de combinações de classificadores geralmente é o resultado de uma redução na variância. Isso ocorre porque o efeito usual da média de conjunto é reduzir a variância de um conjunto de classificadores. *Bagging*, Random Forest (BREIMAN, 2001) e Adaptive Boosting (AdaBoost) (SCHAPIRE, 2003) são técnicas de combinação de classificadores que demonstraram sucesso na redução da variância (SUN; WONG; KAMEL, 2009).

Uma técnica baseada em evolução que tem mostrado sucesso em aplicações biológicas que faz o uso de combinação de classificadores foi proposta em (YANG et al., 2009). Esta técnica baseia-se no principio de otimização de enxame de partículas (PSO). PSO faz parte de um novo grupo de algoritmos baseados na população que usa a ideia de comunicação social e comportamentos históricos para ajustar o processo de otimização (KENNEDY, 2011). A técnica proposta é uma variação binária do PSO, na qual a BPSO é hibridizada com vários classificadores e métricas para seleção de amostras de dados e classificação.

A técnica SMOTE-Boost (CHAWLA et al., 2003), baseia-se em re-amostragem combinada com AdaBoost, de modo a aumentar a classificação correta de instâncias da classe minoritária. Em (LI, 2007) foi proposto o *bagging ensemble variation* (BEV), o qual utiliza uma forma de *bagging* que treina classificadores com todas as instâncias minoritárias e apenas um sub-grupo de instâncias majoritárias. Em oposição a este método, existe o trabalho proposto por (GODASE; ATTAR, 2012), para *streams*, que é um baseado em k-vizinhos mais próximos combinado com oversampling, a fim de fornecer um melhor balanceamento entre as classes.

Ainda no contexto de *streams*, em (DITZLER; POLIKAR, 2010) foi proposto o método Learn^{++}.NIE (*Nonstationary and Imbalanced Environments*), que é uma abordagem baseada no método Learn^{++}.NSE(*Nonstationary Environments*) (ELWELL; POLIKAR, 2009) para cenários aonde existe desabancamento. O método tem como objetivo obter um conjunto de classificadores que possa reconhecer instâncias de ambas as classes, minoritária e majoritária, cujas distribuições podem estar experimentando *concept drift*. Em (LI et al., 2017), foi proposto o *multi-window based ensemble learning* (MWEL), que é baseado na utilização de múltiplas janelas. Cada janela armazena um certo conjunto de instâncias: o *batch* atual de dados, as instâncias minoritárias e as respectivas porções de dados utilizadas

para o treinamento de cada classificador do conjunto. A predição da classe ocorre através de votação majoritária ponderada, sendo que a cada vez que um dos classificadores do conjunto apresentar baixa performance (*precision*), o mesmo é removido e um novo modelo começa a ser treinado.

A próxima seção traz outra forma de melhorar a performance de classificadores em bases de dados desbalanceadas, utilizando uma abordagem baseada em custos.

2.2.2 Abordagens baseadas em custos

Enquanto as técnicas de amostragem tentam balancear a distribuição entre as classes considerando as proporções representativas dos exemplos de classe na distribuição, os métodos de aprendizagem baseados em custos consideram os custos associados com exemplos de classificação errada (ELKAN, 2001). Em vez de criar distribuições balanceadas de dados através de diferentes estratégias de amostragem, a aprendizagem baseada em custos tem como alvo o problema de aprendizagem desbalanceada, utilizando diferentes matrizes de custos que descrevem os custos para classificação para exemplos de S_{maj} e S_{min} (HE; GARCIA, 2009). A matriz de custos pode ser considerada como uma representação numérica da penalidade de classificar exemplos de uma classe para outra. Pesquisas indicam que existe uma forte ligação entre a aprendizagem sensível aos custos e a aprendizagem a partir de dados desbalanceados, portanto os algoritmos baseados em custos podem ser aplicados a problemas de dados desbalanceados (ZADROZNY; LANGFORD; ABE, 2003).

A combinação de classificadores também foi considerada na abordagem baseada em custos para lidar com dados desbalanceados. Vários métodos de combinação de classificadores foram adaptados com sucesso para incluir custos durante a fase de aprendizagem. No entanto, *boosting* foi o mais amplamente explorado (BRANCO; TORGO; RIBEIRO, 2016). AdaBoost é o algoritmo mais representativo da família *boosting*. Quando a classe alvo é desbalanceada, AdaBoost influencia a aprendizagem (através dos pesos) para a S_{maj}, pois contribui mais para a acurácia geral. Várias propostas apareceram que modificam o processo de atualização de peso do AdaBoost incorporando itens de custo para que exemplos de diferentes classes sejam tratados de forma desigual, podendo assim, favorecer a classificação de exemplos em S_{min}.

Tratando-se de abordagens baseadas em algoritmos, as soluções tentam adaptar os classificadores existentes para fortalecer o aprendizado com relação à S_{min}. As soluções de aprendizagem baseadas em custos que incorporam as abordagens a nível de dados e algoritmos assumem maiores custos de classificação incorreta para amostras de S_{min} e procuram minimizar os erros de alto custo (LING et al., 2004). Vários algoritmos de *boosting* também são relatados como meta-técnicas que são aplicáveis à maioria dos algoritmos de classificação. A ideia geral de uma abordagem de *boosting* é introduzir itens de custo em um *framework* de aprendizagem, como o AdaBoost (FREUND; SCHAPIRE, 1995) num

esforço para aumentar mais os pesos nas amostras associadas a uma maior importância de identificação (S_{min}) e eventualmente influenciar positivamente a aprendizagem em relação a elas (SUN; WONG; KAMEL, 2009).

2.2.3 Tratamento de desbalanceamento para *batch*

Diversas técnicas de re-amostragem foram desenvolvidas ao longo do tempo, como segue na tabela 1. Estas técnicas agem geralmente em três frentes: *oversampling*, *undersampling* e abordagens mistas. O *oversampling* consiste em inflar o número de instâncias da classe minoritária pela criação de novas instâncias sintéticas. Espera-se que com as novas instâncias, a classe minoritária exerça maior influência sobre a decisão a ser tomada pelo classificador, devido ao aumento de representatividade desta classe. Os métodos baseados em *undersampling*, por sua vez, buscam remover instâncias da classe majoritária, a fim de proporcionar um maior equilíbrio entre as distribuições das classes. Por fim, abordagens mistas baseiam-se na aplicação sucessiva de uma ou mais técnicas de *oversampling* ou *undersampling*, seguido da aplicação de uma ou mais técnicas com finalidade oposta. Por exemplo: aplicar uma técnica de *oversampling* para aumentar o número de instâncias da classe minoritária para 80% do tamanho da majoritária, seguido da aplicação de uma técnica de *undersampling* para reduzir o tamanho da classe majoritária ao tamanho da minoritária, conseguindo assim, uma distribuição balanceada entre as classes. Tal como combinação de classificadores e abordagens baseadas em custos, a abordagem baseada em re-amostragem pode ser usada tanto no modo *batch*, quanto em fluxos contínuos de dados, visto que uma das abordagens para processar estes fluxos baseia-se na construção de *mini-batches*, que são pequenos grupos de dados que são armazenados ao longo da stream. A próxima sub-seção irá abordar as técnicas de re-amostragem para o ambiente batch. As técnicas foram agrupadas segundo: *oversampling*, *undersampling* e abordagens mistas.

2.2.3.1 Oversampling

A maioria das técnicas de *oversampling* derivam do *Synthetic Minority Oversampling Technique* (SMOTE) (CHAWLA et al., 2002). Esta técnica gera instâncias sintéticas baseada nas observações de S_{min}. Ela é projetada para encontrar os K-vizinhos ($K \in N$) para cada instância $x_i \in S_{min}$. Então, um destes vizinhos é selecionado randomicamente ($\hat{x}_i$) e sua distância para x_i é multiplicada por um número randômico $\delta \in [0, 1]$ para então gerar um novo vetor (x_{new}), que se encontra entre x_i e o K-vizinho escolhido, conforme a equação 2.1.

$$x_{new} = x_i + \delta(\hat{x}_i - x_i) \tag{2.1}$$

Um dos maiores problemas introduzidos pelo SMOTE é o aumento na sobreposição entre as classes, devido a forma com que ele gera as instâncias sintéticas. Todas as

novas instâncias serão geradas em um lugar aleatório entre duas instâncias escolhidas aleatoriamente de S_{min}, sem se preocupar se existem instâncias de S_{maj} nestes locais.

A fim de resolver certos problemas induzidos pelo SMOTE, diversas técnicas propondo variações do algoritmo surgiram ao longo dos anos. Em (HAN; WANG; MAO, 2005) foram apresentadas duas técnicas: SMOTE-Borderline-1 e SMOTE-Borderline-2. Na primeira versão, para cada $x_i \in S_{min}$ os j-vizinhos mais próximos são selecionados. O número de exemplos entre os vizinhos mais próximos é denotado por j', $(0 \leq j' \leq j)$. Se $j' = j$, logo todos os j-vizinhos mais próximos pertencem a S_{maj} e x_i é então considerado um ruído, fazendo com que os próximos passos não sejam executados. Caso $\frac{j}{2} \leq j' < j$, o número dos vizinhos de S_{maj} mais próximos de x_i é maior do que o número de vizinhos mais próximos de S_{min}, o que aumenta a chance de x_i ser classificado incorretamente, então este exemplo deve ir para o grupo chamado de exemplos de risco (RS). Caso $0 \leq j' \leq \frac{j}{2}$, x_i é então considerado como seguro e não participa dos próximos passos. Os exemplos em RS são considerados como pertencentes a borda de S_{min}, o que implica que $RS \subseteq S_{min}$, então $RS = x'_1, x'_2, ..., x'_m$, com $0 \leq m < n$. No passo final, $v \times q$ exemplos sintéticos oriundos de RS são gerados ($V = [0, 1]$). Para cada x'_q os v vizinhos mais próximos dos K-vizinhos mais próximos de S_{min} são selecionados. Primeiramente a diferença $dif_w(1, 2, ..., v)$ entre $x'q$ e os v vizinhos mais próximos de S_{min} é computada. Então a dif_w é multiplicada por um número randômico entre zero e um, $rw, w = (1, 2, ..., v)$. Finalmente, a nova instância sintética v é gerada entre x'_q e seu vizinho mais próximo. Este processo é iterado para cada x'_q de RS, a fim de preencher os vq exemplos, como segue na equação 2.2.

$$wSintético = x'_w + r_w \times dif_w(w = 1, 2, \ldots, v) \tag{2.2}$$

O SMOTE-Borderline-2, por sua vez, segue todos os passos do algoritmo anterior, entretanto ele não gera exemplos sintéticos baseados somente no conjunto R de exemplos de risco, mas também gera novos exemplos considerando os vizinhos mais próximos de S_{maj}. Além disto, no último passo as diferenças são multiplicadas por um valor randômico entre 0 e 0.5, o que gera instâncias mais próximas umas das outras, se comparado a técnica anterior.

Outra variação do SMOTE proposta em (BUNKHUMPORNPAT; SINAPIROMSARAN; LURSINSAP, 2009), denominada SAFE-Level-SMOTE, também classifica cada instância de acordo com seu nível de segurança, antes de gerar novas instâncias. Cada instância sintética é posicionada mais próxima do maior nível seguro, de modo que todas elas são geradas somente em regiões seguras. O nível seguro (sl) leva em conta o número de instâncias positivas nos K-Vizinhos mais próximos, denominada $N_{k_{vizinhos}}$, e pode ser obtida de acordo com a equação 2.3.

$$sl = N_{k_{vizinhos}} \tag{2.3}$$

.

Caso o nível de segurança de uma instância seja próximo a zero, a instância provavelmente é um ruido. Caso ela seja próxima de k, ela é então considerada segura. A taxa de nível de segurança (tsl) precisa além do sl dos K-vizinhos, do sl das instâncias positivas, denominado sl_{pos} e é definida como na equação 2.4.

$$tsl = \frac{sl_{pos}}{sl_{k_{vizinhos}}} \tag{2.4}$$

.

Para descrever como o o algoritmo funciona, é necessário definir algumas variáveis. p é uma instância pertencente ao grupo D de todas as instâncias originais positivas. n representa os vizinhos mais próximos de p selecionados. s é uma instância sintética incluída no grupo de instâncias sintéticas positivas D'. sl_p , sl_n e sl_{taxa} representam o nível seguro de p , o nível seguro de n e a taxa de nível segura, respectivamente. *numattr* representa o número de atributos. dif é a diferencia entre os valores de n e p em um mesmo atributos. gap é uma fracão randômica de dif. $p[i], n[i]$ e $s[i]$ são os valores numéricos da instância no $i-ézinho$ atributo. Definidas as variáveis, pode-se dar início ao algoritmo.

Depois de atribuir os valores de sl_p e sl_n, o algoritmo calcula o sl_{taxa}. Existem cinco casos referentes aos possíveis valores que $sltaxa$ pode assumir.

No primeiro caso, o valor de sl_{taxa} é igual a ∞ e sl_p é igual a 0, o que implica que tanto p quanto n são ruídos. Caso isto ocorra, as instâncias sintéticas não serão geradas, pois o algoritmo não quer enfatizar a importância de regiões de ruído.

Já para o segundo caso, se sl_{taxa} for igual a ∞, porém sl_p seja diferente de 0, isto implica que n é um ruído. Caso esta situação ocorra, a nova instância sintética deverá ser gerada distante de n através da duplicação de p, a fim de evitar a instância ruidosa n.

No terceiro caso, sl_{taxa} é igual a 1, o que implica em sl_p e sl_n são iguais. Caso esta situação ocorra, a nova instância sintética sera gerada ao longo da linha entre p e n, pois p é tão seguro quanto n.

Representando o quarto caso, se $sl_{taxa} > 1$, então $sl_p > sl_n$. Caso isto ocorra, a nova instância é posicionada mais próxima de p, pois p é mais seguro do que n. A instância sintética vai ser gerada na faixa $[0, \frac{1}{sl_{taxa}}]$.

Para o último caso, $sl_{taxa} < 1$, o que implica que $sl_p < sl_n$. Neste caso, a nova instância sintética deverá ser posicionada mais próxima de n, pois n é mais segura do que p. A nova instância deverá ser gerada na faixa $[1 - sl_{taxa}, 1]$.

Após o algoritmo terminar, ele retorna um grupo de instâncias sintéticas D'. O algoritmo gera $|D| - t$ instâncias sintéticas, aonde $|D|$ representa o número de instâncias positivas em D e t é o número de instâncias que satisfazem o primeiro caso.

O SMOTE não leva em conta a distribuição das instâncias em S_{maj}. Apesar do SAFE-Level-SMOTE usar tal informação, o algoritmo pode não ser suficiente para resolver este problema, em particular se S_{min} é decomposta em várias sub-regiões de cardinalidades bastante pequenas. Esta situação refere-se ao problema de pequenos disjuntos, que é conhecida por ser um complicador maior do que o próprio nível de desbalanceamento para os classificadores. A técnica proposta por (MACIEJEWSKI; STEFANOWSKI, 2011), denominada LN-SMOTE é uma extensão da SAFE-Level-SMOTE, que considera a presença de outras instâncias de S_{maj} para geração de novas instâncias sintéticas. No SAFE-Level-SMOTE tradicional, caso um exemplo p seja um *outlier* e o seu vizinho n, selecionado de S_{maj} não tenha nenhum outro vizinho de S_{min}, $sl_p = 0$ e $sl_n = 1$, devido a presença do exemplo p na vizinhança local de n. A $sl_{taxa} = 0$, logo o novo exemplo vai ser gerado exatamente na posição de n, o que introduz a um problema de dados inconsistentes. Então, algumas modificações foram propostas, como o cálculo dos níveis de segurança dos vizinhos de S_{maj}. Caso o exemplo p seja identificado como pertencente aos k vizinhos mais próximos de n, ele não deve ser incluído em sl_n, ao invés disto é procurado pelo próximo $k+1$ vizinho. Um outro problema introduzido ao gerar instâncias sintéticas entre um exemplo $p \in S_{min}$ e seu vizinho $n \in S_{maj}$ é direcionar sua posição mais perto do representante de S_{min}, p do que para S_{maj}, representada por n. Métodos similares, como o SMOTE-Borderline-2, tentam restringir a área aonde as novas instâncias podem ser geradas. Assim, em alguns casos de níveis seguros não são considerados os limites direito do intervalo como 1, mas como limiar $t < 1$. O valor de *threshold* não é fixo, mas sim, é determinado dinamicamente de acordo com os níveis seguros das instâncias de S_{maj} consideradas. Caso sl_n seja relativamente baixo, isto implica que n está cercado por muitos exemplos de S_{maj}, logo o novo exemplo deve ser colocado próximo de p. Caso n esteja cercado por uma quantidade razoável de exemplos de S_{min} , o valor de sl_n é maior, logo um novo exemplo deveria ser alocado próximo de n, o que permite que o nível de expansão de S_{min} seja controlado dinamicamente, levando em conta a distribuição local dos exemplos.

O problema do desbalanceamento afeta diversos classificadores e a adoção de técnicas de *oversampling* podem induzir outros. Por exemplo o classificador *Support Vector Machines* (SVM) enfrenta tipicamente dois problemas ao ser utilizado após a aplicação de alguma técnica de oversampling: o *oversampling* aumenta o número de exemplos e a relação de distribuição dos dados é determinada segundo uma busca gulosa, passando por todo o conjunto de dados várias vezes até que seja possível estabelecer o melhor hiperplano de separação máxima. A fim de resolver este problema, uma adaptação do SMOTE foi proposta. O SMOTE-SVM não faz a re-amostragem de todos os dados pertencentes a S_{min}, mas ao invés disto, uma pequena porção de exemplos é selecionada. É esperado que uma

porção selecionada e representativa de S_{min} possa determinar a fronteira de decisão, o que pode levar a uma redução do tempo despendido para encontrar o hiperplano de separação máxima, com o SVM. A chave para este algoritmo está em encontrar a relação correta entre S_{min} e S_{maj}. Esta relação é denominada R_{smote}, e ela é encontrada empiricamente. Seja X_p os exemplos de S_{min} indexados por p e X_n os exemplos de S_{maj} indexados por n. Tipicamente o SVM tem uma complexidade computacional de $O((X_p + X_n))^3$ para aprender um modelo (BURGES, 1998). O SMOTE-SVM melhora a complexidade em $O((X_p \times (1 + R_{smote}) + X_n))^3$ (CHOI, 2010)

Existem ainda abordagens, como a proposta em (VERBIEST et al., 2012) que fazem a limpeza dos dados antes da aplicação do SMOTE. O algoritmo proposto se baseia em remover instâncias ruidosas da base de dados tal qual a qualidade das instâncias geradas pelo SMOTE seja melhor. Como não faz sentido assumir que apenas as instâncias de S_{min} sejam ruidosas, são removidas instâncias tanto de S_{min} quanto de S_{maj}. Para cada instância é calculado seu nível de ruído segundo a teoria fuzzy irregular dos conjuntos (DUBOIS; PRADE, 1990). Posteriormente, são removidas as instâncias que apresentarem ruído acima de um determinado *threshold*, que é determinado por um procedimento *wrapper* que avalia diferentes *thresholds* com base no Área sob a Curva ROC do treinamento nos sub-grupos de instâncias resultantes. Após este procedimento, o SMOTE é aplicado. Esta técnica de limpeza é chamada *Fuzzy Rough Imbalanced Prototype Selection* (FRIPS). Quando aplicada junto com o SMOTE é chamada FRIPS-SMOTE.

Apesar do grande sucesso do SMOTE, existem outros algoritmos baseados em distância que não são baseados nele, como por exemplo, o ADASYN e o DEAGO. O ADASYN usa um modelo para criar diferentes quantidades de exemplos sintéticos baseados em sua distribuição (HE et al., 2008). O número de exemplos sintéticos a ser gerado é calculado usando a equação 2.5.

$$G = (|S_{maj}| - |S_{min}|) \times \beta \tag{2.5}$$

aonde $\beta \in [0, 1]$ é um parâmetro usado para especificar o nível de balanceamento desejado após a geração das instâncias sintéticas. Para cada $X_i \in S_{min}$, os K-vizinhos mais próximos são selecionados de acordo com a distância euclidiana, então a relação Γ_i é computada seguindo 2.6.

$$\Gamma_i = \frac{\frac{\Delta_i}{K}}{Z}, i = 1, ..., S_{min} \tag{2.6}$$

onde Δ_i representa o número de exemplos nos k-vizinhos mais próximos de x_i pertencentes a S_{maj}, e Z é uma constante de normalização tal que Γ_i é uma função de distribuição $\sum \Gamma_i = 1$. O número de instâncias sintéticas que devem ser geradas para cada $x_i \in S_{min}$ é obtido segundo 2.7.

$$g_i = \Gamma_i \times G \tag{2.7}$$

Finalmente, $\forall x_i \in S_{min}, g_i$ exemplos são gerados sinteticamente de acordo com a equação 2.1.
A chave deste método é usar a distribuição de densidade Γ como um critério para determinar o número de exemplos sintéticos a serem gerados para cada exemplo de S_{min}, através da adaptação de diferentes pesos de diferentes exemplos de S_{min} a fim de compensar a distribuição desigual das classes.

O DEAGO é uma técnica de *oversampling* que é baseada nas capacidades de se modelar e se reconstruir observada em autodccodificadores (BELLINGER; JAPKOWICZ; DRUMMOND, 2015). Autodecodificadores são redes neurais que aprendem a construir versões mais limpas da entrada da rede na camada de saída. O algoritmo leva as instâncias de treino X como entrada e realiza a normalização das características antes de treinar o autodecodificador *dea*. *dea* é então treinado com h unidades ocultas para e épocas via gradiente descendente *backpropagation*. Após a criação e treinamento da rede, o conjunto de iniciação da amostra normalizada é mapeado via *dea* para a distribuição induzida.

Os próprios pontos de iniciação da amostra podem ter um impacto no conjunto sintético dependendo da forma como são selecionados. Isto se deve ao fato de que a função induzida $g(f(\tilde{x}))$ mapeia a entrada da distribuição de aprendizado $\tilde{x}$. Assim, se o conjunto de iniciação da amostra é distribuído numa pequena região em relação à distribuição aprendida, as amostras resultantes preencherão a região próxima da distribuição induzida. Isso pode ser uma maneira eficaz de influenciar o classificador induzido em favor da classe minoritária.

A saída Y, do algoritmo DEAGO é desnormalizada e retornada para uso em inflar o conjunto de treinamento minoritário. Em resultados empíricos é sugerido o uso de $X_{init} = X + N(\Delta)$ para um conjunto sintético que cobre todas as regiões da distribuição induzida. Alternativamente, o conjunto sintético pode ser tendencioso para a classe majoritária, utilizando as instâncias de S_{maj} no X_{init}. Finalmente, a complexidade e generalidade da função induzida por DEAGO é controlada pelo número de unidades ocultas u e o ruído de treinamento σ.

Além das estratégias baseadas em distância apresentadas acima, existem outras abordagens, como por exemplo, baseada em *clusters*, como o A-SUWO (NEKOOEIMEHR; LAI-YUEN, 2016). O A-SUWO consiste de três etapas principais: agrupamento semi-supervisionado, dimensionamento adaptativo do sub-conjunto, e a geração de instâncias sintéticas. Na primeira etapa, as instâncias de S_{min} são agrupadas usando uma abordagem de agrupamento hierárquico semi-supervisionado que agrupa iterativamente instâncias de S_{min}, evitando os exemplos de S_{maj} entre eles. O método de agrupamento hierárquico usa informações sobre as instâncias de S_{maj} para mesclar os subgrupos minoritários nomeados e evitar gerar sobreposição de instâncias sintéticas. Na etapa de dimensionamento do sub-grupo adaptativo, o tamanho a que cada subgrupo minoritário será aumentado é

determinado com base na sua complexidade ao ser classificado (erro de classificação). Uma nova medição baseada na taxa de erro médio padronizada foi proposta para determinar a complexidade do sub-grupo e é calculado usando validação cruzada. Finalmente, na etapa de geração de instância sintética, um novo sistema de ponderação foi proposto para atribuir pesos a instâncias de S_{min} com base na sua distância euclidiana média para seus vizinhos de S_{maj} mais próximos, de modo que as instâncias sintéticas são geradas com base nesses pesos.

Em áreas como a medicina, existem inúmeros problemas que envolvem dados desbalanceados. Em (LI et al., 2016) foi proposta uma estratégia baseada em nuvem de partículas combinada com o SMOTE, a fim de gerar instâncias sintéticas baseadas nas instâncias de S_{min} que obtivessem o menor Kappa. Kappa é uma medida de concordância usada em escalas nominais que nos fornece uma ideia do quanto as observações se afastam daquelas esperadas.

2.2.3.2 Undersampling

As técnicas de *undersampling* excluem instâncias de S_{maj} a fim de fornecer uma distribuição mais equilibrada entre as classes. Elas agem de duas formas: **limpeza de** instâncias ruidosas introduzidas após a aplicação de uma técnica de ***oversampling* ou** simplesmente na redução de instâncias de S_{maj}. Uma das primeiras **técnicas para limpeza** dos dados que se tem notícia é o Tomek-Links (TK) (TOMEK, 1976). **TK visa reduzir** a sobreposição entre as classes que é induzida pelas técnicas de **oversampling. Ele pode** ser definido como um par de vizinhos mais próximos, pertencentes a **classes opostas, que** possuem distância mínima. Dado um par de instâncias (x_i, x_j), onde $x_i \in S_{min}, x_j \in S_{maj}$ e $d(x_i, x_j)$ é a distância entre x_i e x_j, o par de instâncias pode ser considerado um TK se não existe uma instância x_k tal que $d(x_i, x_k) < d(x_i, x_j)$ ou $d(x_j, x_k) < d(x_i, x_j)$. Com isto em mente, se um par de instância forma um TK, uma das instâncias é um ruído, ou ambas estão em uma região de borda. De qualquer forma, a redução destas instâncias tende a trazer um resultado melhor para classificação, pois *clusters* de classe bem definidos são utilizados para induzir os modelos de classificação, o que pode levar a regras de classificação melhores definidas, aumentando a performance do classificador em ambas as classes.

Uma outra técnica baseada em TK é a *Parallel selective sampling* (PSS). Esta técnica age dividindo S_{maj} em N (fornecido pelo usuário) sub-grupos diferentes. Para cada sub-grupo formado, são computados os TK referentes ao sub-grupo e S_{min} e todos os pontos que não representam TK são selecionados para um grupo R. Este processo é iterado é que a base de dados esteja balanceada. Quando o balanceamento ocorrer, o grupo R é utilizado para treinar o classificador.

Tal como o TK, a maioria das técnicas são baseadas em distância, como por exemplo, a estratégia proposta em (MANI; ZHANG, 2003). Baseado nas características

da distribuição dos dados, três técnicas foram propostas: NearMiss-1, NearMiss-2 and NearMiss-3. No NearMiss-1, amostras são selecionadas de S_{maj} tal que a distância média para os três vizinhos de S_{min} mais próximos, seja a menor. Se o NearMiss-1 procura os exemplos de S_{min} mais próximos, o NearMiss-2, por outro lado, procura os três exemplos mais distantes. O NearMiss-3 seleciona um dado número de exemplos de S_{maj} que estão mais próximos de cada exemplo de S_{min}, com a finalidade de garantir que cada exemplo de S_{min} esteja rodeado por algum exemplo de S_{maj}.

O *Majority Under-sampling Technique* (MUTE) é uma técnica de *undersampling* que foi inspirada no SAFE-Level-SMOTE, que define níveis seguros apenas em instâncias minoritárias antes de aplicar uma técnica de oversampling. Em contraste com o SMOTE de Nível Seguro, o MUTE aplica níveis seguros em instâncias majoritárias para fins de subamostragem (BUNKHUMPORNPAT; SINAPIROMSARAN; LURSINSAP, 2011). Um nível seguro de uma instância majoritária, obtido por 2.3, é calculado pelo número de instâncias minoritárias entre k-vizinhos mais próximos. Um exemplo de S_{maj} está localizado em uma região segura se um nível seguro for igual a 0. Por outro lado, se um nível seguro é igual a k, uma instância maioritária é considerada ruído. O objetivo do MUTE é eliminar o *Majority Noise Set* (MNS), a partir de um conjunto de dados. O MNS é definido por $N = n \in S|Sl_n = k$. Espera-se que com a aplicação desta técnica, apenas instâncias ruidosas sejam removidas de S_{maj}.

Um outro exemplo de técnica de *undersampling* bastante utilizado é o ***Random Undersampling*** (RUS). Esta técnica age selecionando um sub-grupo randômico E de S_{maj} e o exclui de S, causando $|S| = |S_{min}| + |S_{maj}| - |E|$. Como não existe critério sobre quais instâncias serão deletadas da base de dados, pode ocorrer perda de informação relevante, pois não existem garantias de que amostras representativas não serão excluídas de S_{maj}.

Um dos maiores problemas com a utilização de técnicas de *undersampling* é que elas podem levar a perda de informação, uma vez que são removidas instâncias de S_{maj}. A fim de resolver este problema, a técnica *Cluster Centroids* (CC) combina a utilização do algoritmo de agrupamento K-médias com a técnica de RUS (KUMAR et al., 2014). CC inicia dividindo S_{maj} em K grupos. As instâncias no ultimo grupo gerado são nomeadas MA_i. A relação entre o número de exemplos de S_{maj} para o número total de exemplos no ultimo grupo é definida como $r_i = \frac{MA_i}{|S_{maj}|}, 1 \leq i \leq k$. O número de exemplos mais próximos de S_{maj} baseados no centróide, que devem ser mantidos, é computado com a equação 2.8.

$$si = MA_i \times ri \tag{2.8}$$

2.2.3.3 Abordagens mistas

Ao contrário da FRIPS-SMOTE, que faz a limpeza dos dados antes de aplicar o SMOTE, o SMOTE-TOMEK-LINKS aplica o SMOTE e após faz a limpeza das instâncias

sintéticas que foram geradas em uma região imprópria do espaço de características, pela aplicação de TOMEK-LINKS (BATISTA; PRATI; MONARD, 2004).

A seção 2.2.4 traz técnicas para tratamento de desbalanceamento para fluxos contínuos de dados, seguindo os mesmos critérios de divisão da seção 2.2.3.

2.2.4 Tratamento de desbalanceamento em fluxos contínuos de dados

Embora a mineração de fluxos conínuos de dados tenha atraído bastante atenção recentemente, a atenção dada aos fluxos de dados desbalanceados tem sido bastante limitada (HE; GARCIA, 2009). Além disso, no que se refere à aprendizagem incremental a partir de fluxos dc dados desbalanceados, algumas questões importantes precisam ser abordadas, como: o balanceamento dos dados deve ser feito durante a aprendizagem incremental? Se sim, de que maneira?

Existem abordagens para detecção de *drift* em bases de dados desbalanceadas, como em (WANG; ABRAHAM, 2015), abordagens baseadas em combinação de classificadores e adaptação de modelos (GHAZIKHANI; MONSEFI; YAZDI, 2014), porém existe pouco material referente a técnicas de re-amostragem para o ambiente de fluxos contínuos de dados.

2.2.4.1 oversampling

As técnicas de re-amostragem para fluxos contínuos de dados se baseiam no armazenamento momentâneo das instâncias que estão chegando em um ***mini-batch***, a fim de que os dados possam ser pre-processados antes de apresentá-los ao classificador. Com isto, se não existir *concept drift* nos dados, as técnicas de re-amostragem tradicionais do ambiente *batch* podem ser aplicadas em fluxos contínuos de dados, porém existem abordagens desenvolvidas especificamente para este ambiente, como o algoritmo *Uncorrelated Bagging* (UCB). Ele é baseado em uma estrutura de bagging que treina classificadores em um subconjunto das instâncias de S_{maj} e a união de todas as instâncias de S_{min} vistas até o momento atual (atual + Exemplos positivos anteriores). Com cada novo lote de dados, instâncias de S_{min} são salvas e acumuladas para treinar os classificadores nas iterações futuras. No entanto, esta abordagem assume implicitamente que os dados de S_{maj} são estacionários. Esta suposição, quando violada, pode fazer com que o algoritmo interprete incorretamente o espaço de característica verdadeiro de S_{maj} em etapas de tempo subsequentes (Charles Ditzler, 2011).

No algoritmo UCB as instâncias minoritárias são salvas para utilização futura, porém no *Selectively Recursive Approach algorithm* (SERA) é utilizada uma medida de similaridade para selecionar exemplos de S_{min} anteriores que são mais semelhantes aos do conjunto de dados mais recente (CHEN; HE, 2009).

Em (SPYROMITROS-XIOUFIS, 2011) foi apresentado um método baseado em janelas que usa o classificador *K-Nearest Neighbors* (k-NN) para classificação multi-label para dados que contém *concept drift* e dados desbalanceados.

Mais recentemente, em (Charles Ditzler, 2011) foi proposta uma abordagem baseada em *mini-batches* que combina o SMOTE com o algoritmo $Learn^{++}.NSE$, que é um algoritmo versátil, capaz de acomodar uma variedade de cenários de *drift* de conceito, incluindo taxas de *drift* cíclicas ou variáveis, bem como a adição / remoção de classes que a maioria dos outros algoritmos não abordam.

Diferentemente das abordagens que existem para *batch*, no contexto de fluxos contínuos de dados não existem propostas de *undersampling* e, consequentemente, abordagens mistas. Parte disto se deve a detecção de *drift*, pois ao remover instâncias de S_{maj}, uma possível mudança de conceito pode não ser observada devido à ausência de informação para tal.

As técnicas de *oversampling* e de *undersampling* visam fornecer uma distribuição mais equilibrada entre as classes, a fim de facilitar o processo de classificação. Na tabela 1 são apresentadas todas as técnicas estudadas, para *batch* e fluxos contínuos de dados, segundo suas características. Porém, todas as técnicas estudadas induzem a problemas, que podem ser: perda de informação relevante, problema de sobreposição de classes ou gerar várias instâncias sintéticas que pouco contribuem na melhora da performance dos classificadores. Existe a necessidade de uma técnica que não induza a perda de informação e consiga gerar novas instâncias para S_{min} seguindo a tenência dos dados, a fim de aumentar a representatividade desta classe no espaço de características, possibilitando a descoberta de novos e interessantes padrões de classificação.

A fim de avaliar a performance de classificadores em bases de dados desbalanceadas, métricas específicas podem facilitar a visualização dos resultados, fornecendo um extrato honesto sobre a performance. Porém, existem métricas que devem ser evitadas em determinados contextos, pois podem fornecer uma falsa impressão sobre taxa de classificações corretas. A próxima seção irá trazer uma discussão sobre o assunto, tal como as métricas adequadas na presença de desbalanceamento.

2.3 Métricas para avaliação

Um dos problemas ao avaliar a performance em bases de dados desbalanceadas, é que muitas vezes o objeto de interesse na classificação é a performance do classificador nas instâncias de S_{min}, porém as métricas tradicionais de avaliação tendem a serem mais afetadas pelos casos mais frequentes (S_{maj}) (BRANCO; TORGO; RIBEIRO, 2016).

No cenário de classificação tradicional, cada instância $x_i \in S$ está associada a uma

Tabela 1 – Sumário dos artigos que propõe técnicas de re-amostragem para desbalanceamento

Categoria	Estratégia	Artigos
Oversampling	Baseados em Cluster	(CAO; TAN; PANG, 2014), (NEKOOEIMEHR; LAI-YUEN, 2016)
	Baseados em distância	(CHAWLA et al., 2002), (CHAWLA et al., 2003), (HAN; WANG; MAO, 2005), (HE et al., 2008), (BUNKHUMPORNPAT; SINAPIROMSARAN; LURSINSAP, 2009), (CHEN; HE, 2009), (CHOI, 2010), (ZHANG et al., 2011), (MACIEJEWSKI; STEFANOWSKI, 2011), (Charles Ditzler, 2011), (SPYROMITROS-XIOUFIS, 2011), (VERBIEST et al., 2012), (GODASE; ATTAR, 2012), (BELLINGER; JAPKOWICZ; DRUMMOND, 2015)
	Baseados em evolução	(CASTILLO; SERRANO, 2004), (LI et al., 2016)
Undersampling	Baseados em Cluster	(YEN; LEE, 2009), (KUMAR et al., 2014), (SOBHANI; VIKTOR; MATWIN, 2014)
	Baseados em distância	(TOMEK, 1976), (MANI; ZHANG, 2003), (D'ADDABBO; MAGLIETTA, 2015), (ANAND et al., 2010), (BUNKHUMPORNPAT; SINAPIROMSARAN; LURSINSAP, 2011)
	Baseados em evolução	(DOUCETTE; HEYWOOD, 2008)
Abordagens Mistas	-	(BATISTA; PRATI; MONARD, 2004), (YANG et al., 2009), (SEIFFERT et al., 2010), (RAMENTOL et al., 2012), (CAO et al., 2014), (DUBEY et al., 2014), (CATENI; COLLA; VANNUCCI, 2014), (JIAN; GAO; AO, 2016), (MAO et al., 2016)

classe $y_i \in Y$ e o classificador assinala uma classe $\hat{y}_i = h(x_i)$. A classe assinalada, geralmente é aquela que possui a maior probabilidade condicional $\hat{y}_i = argmax_y \hat{P}(Y = y|X = x_i)$. Seja $I()$ a função indicadora que retorna 1 caso o argumento seja verdadeiro e 0 caso seja falso, $n_c = \sum_{i=1}^{n} I(y_i = c)$ representa o total de instâncias que pertencem a classe c. A probabilidade a priori da classe c pode ser representada como $p(Y = c) = \frac{n_c}{n}$, tal como a probabilidade estimada de uma instância x_i pertencer a classe c pode ser estimada por $\hat{P}(c|x_i)$ (BRANCO; TORGO; RIBEIRO, 2016).

A maioria das métricas de avaliação podem ser obtidas através da matriz de confusão. Considerando um problema binário de classificação, aonde existe uma classe

Tabela 2 – Matriz de confusão

	Positivos	Negativos	Total
c_+	TP	FN	n_+
c_-	FP	TN	n_-

positiva $c_+(Y = +)$ e outra negativa $c_-(Y = -)$, a matriz de confusão pode ser representada como na Tabela 2. Os valores para TP, FN, FP e TN, podem ser obtidos segundo as equações 2.9, 2.10, 2.12 e 2.13, respectivamente. Como cálculo adicional, estão as somas $n_+ en_-$, que são obtidas de acordo com as equações 2.11 e 2.11.

$$TP = \sum_{i=1}^{n} I(y_i = +)I(\hat{y}_i = +) \tag{2.9}$$

$$FN = n_+ - TP \tag{2.10}$$

$$n_+ = \sum_{i=1}^{n} I(y_i = +) \tag{2.11}$$

$$FP = n_- - TN \tag{2.12}$$

$$TN = \sum_{i=1}^{n} I(y_i = -)I(\hat{y}_i = -) \tag{2.13}$$

$$n_- = \sum_{i=1}^{n} I(y_i = -) \tag{2.14}$$

A métrica mais comum para avaliação de classificadores é a acurácia, que pode ser obtida pela equação 2.15, porém, esta métrica não é adequada quando refere-se a problemas de desbalanceamento (LYON et al., 2014), pois pode fornecer uma falsa impressão de boa performance. Em um problema binário de classificação, aonde a classe majoritária possui 900 instâncias e a classe minoritária 100, se o classificador predizer corretamente todas as instâncias da classe majoritária, porém, errar todas as pertencentes a classe minoritária, a acurácia seria de 90 %. Neste contexto, existem métricas como o sensibilidade (taxa de TP) e a especificidade (taxa de TN), que podem ser obtidos através das equações 2.16 e 2.17, respectivamente. Estas métricas podem fornecer um estrato mais adequado da performance do classificador, pois apresentam o percentual de classificações corretas em cada classe. Se o objetivo for avaliar a performance em ambas as classes simultaneamente, é necessário cautela, pois a grande maioria destas métricas pode induzir a falsa sensação de boa performance, como a acurácia, devido ao fato de utilizarem os TP e TN simultaneamente para computar seu resultado, não se preocupando se existe um

desbalanceamento severo nos dados. Neste contexto métricas como a *Receive Operating Characteristics curve* (ROC) e *Area under the Receive Operating Characteristics curve* (AUROC) são mais adequadas, pois possibilitam ver as trocas entre os benefícios (taxa de TP) e os custos (taxa de FP) (PROVOST; FAWCETT; KOHAVI, 1998), conforme a figura 2. A área resultante do calculo do AUROC pode ser interpretada como a probabilidade de um positivo escolhido randomicamente ocorrer ser mais alta do que um exemplo negativo escolhido randomicamente, caso a região esteja situada acima da área definida pela função identidade no intervalo definido.

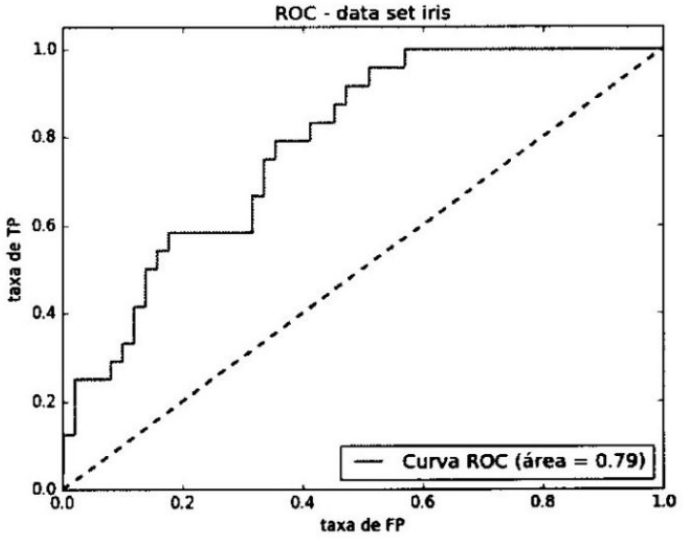

Figura 2 – Exemplo de curva ROC

$$Acurácia = \frac{TP + TN}{TP + FN + TN + FP} \tag{2.15}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.16}$$

$$Especificidade = \frac{TN}{TN + FP} \tag{2.17}$$

2.4 Procedimentos de avaliação para *streams*

Apesar do aumento no número de algoritmos de classificação para o ambiente de *streams*, as métricas e o planejamento destes experimentos para se ter acesso a qualidade dos modelos inferidos ainda é um problema em aberto, pois existe um fluxo contínuo de dados, ao invés de um número fixo de instâncias, como no cenário *batch*, e os modelos de classificação evoluem com o passar do tempo, ao invés de serem estáticos. (RODRIGUES, 2013).

O procedimento de avaliação trivial para o ambiente é o *test-then-train*, que testa o modelo preditivo a cada instância recebida, efetuando a adaptação do mesmo na sequência. Uma alternativa, caso exista um grande conjunto de dados, é a avaliação *holdout*, aonde o modelo atual de predição é exposto a um grupo de teste a cada intervalo regular de tempo (ou grupo de exemplos). Em um grupo de teste *holdout* com M exemplos, a perda é computada tal como segue na equação 2.18

$$H_e(i) = \frac{1}{M}\sum_{k=1}^{M} L(y_k, \hat{y_k}) = \frac{1}{M}\sum_{k=1}^{M} e_k \tag{2.18}$$

Uma outra alternativa é a avaliação *Predictive Sequential* (PREQUENTIAL). Para esta avaliação o erro de um modelo é computado de uma sequência de exemplos. Para cada exemplo recebido do fluxo contínuo de dados, o modelo atual faz uma predição baseada nos valores dos atributos do exemplo. Sabe-se que o erro estimado através de todo o fluxo contínuo de dados pode influenciar esta avaliação. Tendo em vista que os primeiros modelos tendem a ter uma performance inferior, uma vez que tiveram menos observações dos dados, isto pode levar a uma alta taxa de erro que irá se propagar por todo o processo de classificação. A fim de resolver este problema, fez-se necessário criar um mecanismo de esquecimento, o que pode ser obtido utilizando uma janela deslizante de tamanho infinito ou fatores de decaimento. Para ambos os casos, em *streams* estacionárias, tanto o PREQUENTIAL quanto o *holdout* convergem para o erro de Bayes (GAMA; SEBASTIÃO; RODRIGUES, 2013). Para a avaliação PREQUENTIAL não é necessário conhecer o valor real de y_i para todas as instâncias recebidas (RODRIGUES, 2013). O erro do PREQUENTIAL, computado em um tempo i, em uma janela deslizante de tamanho w, é baseado na soma acumulada da função de perda entre as predições e os valores observados segundo a equação 2.19.

$$P_e(i) = \frac{1}{i}\sum_{k=1-w+1}^{i} L(y_k, \hat{y_k}) = \frac{1}{i}\sum_{k=1-w+1}^{i} e_k \tag{2.19}$$

A avaliação prequential foi provada ser uma das melhores opções para estimar o erro em um fluxo contínuo de dados, pois pode usar mecanismos de esquecimento e fornecer sempre uma estimativa atual sobre o modelo de predição, além de propiciar uma resposta mais rápida na detecção de uma mudança de conceito (RODRIGUES, 2013).

2.5 Considerações finais

Este capítulo cobriu os principais aspectos relacionados a classificação no ambiente de *stream*, tal como trouxe uma análise sobre o problema de desbalanceamento, cobrindo as principais técnicas empregadas neste contexto e as métricas mais adequadas para um

estrato real de performance. Estes conceitos serviram como base para este trabalho, uma vez que esta sendo proposto um método de re-amostragem. Apesar da variedade de técnicas de re-amostragem existentes, poucas foram desenvolvidas para o ambiente de *stream*. Parte disto se deve ao fato de que técnicas populares, como o SMOTE e seus derivados, possuem alta complexidade computacional, o que inviabiliza sua utilização neste ambiente, e outras técnicas menos custosas, como RUS, podem induzir a perda de informação. O próximo capítulo traz a proposta de um novo método de *oversampling*, baseado na hipótese de que ao aumentar a representatividade de S_{min} no espaço de características, novos e interessantes padrões de comportamento possam ser observados, o que poderia melhorar a performance de classificação nesta classe.

3 Método

De todos os métodos de re-amostragem estudados no capítulo anterior, o SMOTE foi o que mais influenciou a criação de novos métodos. O SMOTE é um método baseado em distância que age a nível de instância, as tratando como vetores n-dimensionais posicionados no espaço de características. De forma sucinta, a geração de instâncias sintéticas por este método ocorre em duas etapas:

- Escolha de uma instância minoritária $\vec{x}_i$ aleatoriamente (pivô)
- Para cada vizinho $\vec{x}_j$ minoritário desta instância, criar uma nova instância $\vec{x}_{new} = \vec{x}_i + \delta(\vec{x}_j - \vec{x}_i)$, $\delta \in [0, 1]$

Graficamente, o funcionamento do SMOTE está representado na figura 3, sendo a classe minoritária representada por círculos vermelhos e a majoritária por estrelas verdes. A esquerda desta figura, estão as instâncias originais, já o recorte à direita demonstra a geração de instâncias, através de um pivô.

A abordagem baseada em operações com vetores pode induzir a geração de valores em faixas inaceitáveis para atributos nominais tal como impedir o aumento da representatividade da classe minoritária no espaço de característica, uma vez que as instâncias geradas tendem a aglomerar-se em uma pequena região.

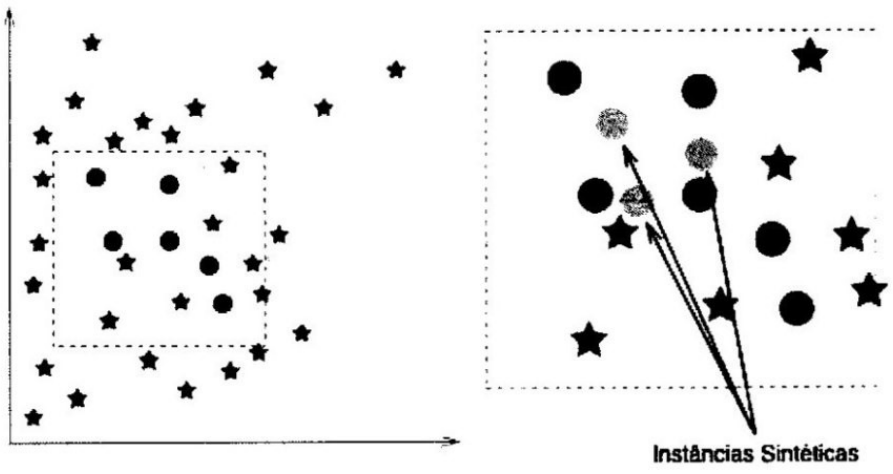

Figura 3 – Processo de geração de instâncias sintéticas com o SMOTE adaptado de (KARAHOCA, 2012)

Outras abordagens, como as baseadas em *undersampling* apresentadas na seção 2.2.3.2, apesar de menos custosas, podem induzir a perda de informação, uma vez que ao balancear os dados utilizando estas técnicas, uma grande quantidade de instâncias é descartada.

A hipótese do algoritmo proposto é de que ao aumentar a representatividade da classe minoritária no espaço de características, seguindo o fluxo natural dos dados, ocorreria uma melhora na performance de classificação na classe referida, sem que ocorra uma deterioração severa na performance da classe majoritária. O cumprimento destes requisitos resolveria os problemas observados com o SMOTE.

A próxima seção descreve uma nova abordagem para re-amostragem chamada *Regression Functions Approach for Oversampling* (RFAO).

3.1 Um método baseado em funções de regressão: RFAO para *batch* (protótipo)

As técnicas de *oversampling* propostas nas seções 2.2.3.1 e 2.2.4.1 agem a nível de instância. Tal comportamento causa impacto negativo no tempo de execução das mesmas, já que na maioria das vezes, cálculos de distância entre pares de instâncias são realizados. Como a abordagem proposta tem como foco o ambiente de *streams*, baseia-se em atributos, que uma vez processados, podem ser utilizados para geração de tantas instâncias sintéticas quanto se desejar. Sendo assim, o método aqui proposto sobrepuja uma das principais dificuldades encontradas na migração dos métodos tradicionais de *oversampling* para o ambiente de *streams*.

Em casos de desbalanceamento elevado, no contexto de *streams*, o intervalo de recebimento de instâncias minoritárias pode ser longo. Tal fato pode dificultar ou postergar o reconhecimento de padrões nesta classe. Com o objetivo de validar as hipóteses propostas, um protótipo foi desenvolvido para o ambiente *batch*, que por definição, torna disponível todas as instâncias nas etapas de treinamento e teste, facilitando o reconhecimento de padrões pelo método. Tal abordagem também permite a verificação e correção mais rápida de possíveis falhas no projeto, dada a abundância de recursos disponíveis para implementação e testes neste ambiente. O intuito desta implementação inicial é de servir como base para implementação final, no ambiente de *streams*.

A proposta inicial do método é de aumentar a representatividade da classe minoritária no espaço de características. Para tal, uma abordagem baseada em funções de regressão linear simples (RLS) foi adotada. Esta abordagem se baseia em induzir uma relação (função) entre duas variáveis, uma dependente (X) e outra independente (Y), tal que a variável X deve fornecer informações sobre o comportamento de Y (SILVA, 2017). Sua relação linear é definida pela equação 3.1, sendo β_0 o coeficiente linear, β_1 o coeficiente angular e X_i representa as n observações da variável explicativa. Os valores de β_0 e β_1 podem ser obtidos através das equações 3.2 e 3.3, respectivamente.

$$E[Y_i] = \beta_0 + \beta_1 X_i \tag{3.1}$$

$$\beta_0 = \frac{\sum Y \sum X^2 - \sum X(\sum XY)}{n(\sum X^2) - (\sum X)^2} \tag{3.2}$$

$$\beta_1 = \frac{n(\sum XY) - \sum X \sum Y}{n(\sum X^2) - (\sum X)^2} \tag{3.3}$$

A opção de utilizar a RLS, que resume sua aplicação a apenas duas variáveis por vez, se deve ao fato de que relações de dependência entre variáveis (atributos), intuitivamente, tendem a se tornar mais raras a medida que se aumenta o número de variáveis analisadas. De maneira mais direta, é mais plausível encontrar dois atributos correlacionados, do que cinco.

Antes de determinar se um certo par de atributos possui correlação, é preciso estudar o comportamento destes atributos individualmente, uma vez que os testes de correlação são direcionados de acordo com o tipo de atributo estudado (normal ou não-normal). Um dos testes de normalidade mais comuns é o desenvolvido por Shapiro e Wilk (SHAPIRO; WILK, 1965). Trata-se de um teste apropriado para detecção de desvios da normalidade devidos à assimetria e curtose. Este teste parece ser superior aos testes baseados em distância, como qui-quadrado e de Kolmogorov-Smirnov (D'AGOSTINO, 1971), porém, não foi estendido para um número de amostras maior do que 50, o que inviabiliza sua utilização em *mini-batches* com mais de 50 instâncias. O teste baseado na estatística D (DOWNTON, 1966) de D'Agostino (D'AGOSTINO, 1971) foi escolhido por ser aplicável para amostras de qualquer tamanho, não requerendo tabelas de pesos, e para amostras de tamanho 50 ou mais a sua distribuição nula pode ser aproximada por expansões Cornish-Fisher (CORNISH; FISHER, 1938).

Após realizar o teste de normalidade, cada atributo normal e não-normal deverá ter sua correlação testada com os demais atributos que possuem o mesmo comportamento. Os três testes de correlação mais populares são: Pearson (PEARSON, 1920), Spearman (SPEARMAN, 1904) e Kendall (KENDALL, 1938), sendo o primeiro deles destinado a dados com comportamento normal, e os dois últimos para dados não-normais (HAUKE; KOSSOWSKI, 2011). Para compor o método, apenas os dois primeiros foram utilizados, um para cada tipo de variável.

Em cenários do mundo real, nem sempre é possível encontrar correlações entre atributos existentes, o que impossibilitaria a geração de valores via RLS. A fim de cobrir estes casos, o método desenvolvido recebeu a capacidade de gerar valores de acordo com probabilidade de ocorrência e estatística. Atributos binários, por exemplo, sempre são gerados via probabilidade de ocorrência, uma vez que outro mecanismo de geração poderia

implicar na geração de faixas de valores inaceitáveis, e.g. uma variável binária recebendo um valor real. Os demais atributos que não possuem correlação são gerados de acordo com a moda somada a um valor randômico limitado pelo desvio padrão (STDEV).

Com base nas definições supracitadas, a execução do RFAO para *batch* foi dividida em duas etapas: exploratória e geração de instâncias sintéticas. Na etapa exploratória, o algoritmo analisa apenas a porção de dados referente a S_{min}. Para cada atributo, as seguintes verificações são realizadas:

1. É binário?
2. Possui comportamento normal?
3. Possui correlação $\geq$ a estipulada com algum outro atributo?
4. Existe mais algum atributo correlacionado com o atual?

A etapa exploratória tem como objetivo sinalizar de que maneira os valores devem ser gerados, para cada atributo, com base em seu comportamento. Por fim, na etapa de geração de instâncias sintéticas, os seguintes passos são executados:

1. Caso seja binário, gerar de acordo com a probabilidade de ocorrência
2. Após testar normalidade, analisar correlação com outros atributos que possuam o mesmo comportamento
3. Caso possua correlação $\geq$ a estipulada, identificar par de atributos como elegível para geração via RLS, caso contrário, gerar valores via moda $+-$ um valor aleatório $\in [-STDEV, +STDEV]$
4. Entre os pares de atributos da lista de elegíveis para geração via RLS, caso um atributo esteja presente em mais de um par, manter apenas o par com maior correlação, os demais devem ser gerados via moda $+-$ um valor aleatório $\in [-STDEV, +STDEV]$

A execução completa do RFAO pode ser visualizada através do diagrama 4, onde a cor verde escura representa a entrada e saída do algoritmo, verde claro representa ações triviais, como seleção de dados, a cor azul representa testes realizados e vermelho representa situações não favoráveis para o algoritmo.

Dois parâmetros foram incluídos no método, um referente a proporção de balanceamento δ que S_{min} deve apresentar em relação a S_{maj} e outro referente a correlação mínima λ esperada entre atributos para que sejam gerados via RLS. O primeiro atributo possui faixa de variação $[0, 100]$, aonde 100% geraria instâncias sintéticas até S_{min} possuir mesmo número de instâncias de S_{maj}. Nota-se que o balanceamento somente é executado caso a

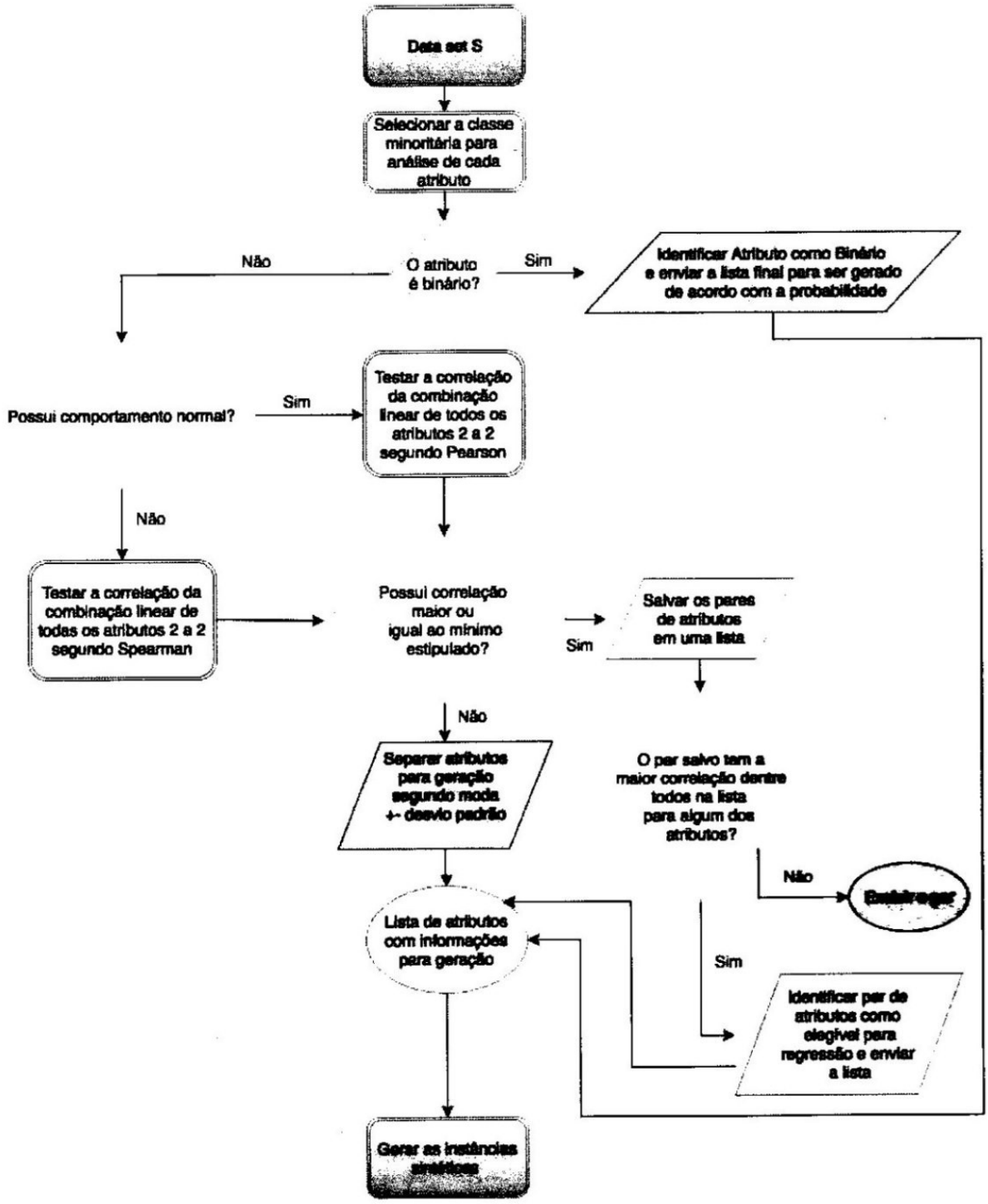

Figura 4 Processo de geração de instâncias sintéticas com o RFAO

proporção δ informada seja $\geq$ do que o balanceamento atual dos dados. Por fim, λ possui variação entre $[0, 1]$, aonde 1 representa a correlação máxima.

A primeira fase do algoritmo consiste em separar os atributos de acordo com seu comportamento e é executada pela função representada em pseudo-código no Algoritmo 1. A função recebe um *batch* S e a lista de atributos referentes a este *batch* . Caso o atributo seja binário, a geração sintética é feita segundo probabilidade de ocorrência de cada um dos dois valores possíveis, portanto é necessário manter uma lista (L_b) contendo cada

atributo binário junto com sua respectiva tupla de probabilidades (P_b). Caso o atributo não seja binário, é realizado um teste de normalidade a fim de estabelecer se os valores do atributo apresentam comportamento gaussiano. Se o atributo apresentar comportamento normal, deve ser adicionado a lista L_n e caso não apresente, à L_{np}.

Algorithm 1 RFAO

```
1: função TipoAtributo(S_min, X)
2:     P_b ← recebe uma tupla de probabilidades
3:     L_b ← recebe uma lista de tuplas. Cada tupla contém o nome do atributo e P_b
4:     L_n ← recebe uma lista atributos que possuem comportamento normal
5:     L_np ← recebe uma lista atributos que não possuem comportamento normal
6:     para cada f_i ∈ X faça
7:         se f_i é binário então
8:             P_b ← (p_valor[0], p_valor[1])
9:             L_b ← L_b ∪ (f_i, P_b)
10:        senão
11:            se f_i apresenta comportamento normal então
12:                L_n ← L_n ∪ f_i
13:            senão
14:                L_np ← L_np ∪ f_i
15:            fim se
16:        fim se
17:    fim para
18:    retorne (L_b, L_n, L_np)
19: fim função
```

Para finalizar a etapa exploratória (Algoritmo 2), são adicionadas a lista L_{cn} todas as combinações possíveis, dois a dois, dos elementos de L_n, e à lista L_{cn} são adicionadas todas as combinações entre os elementos de L_{np}. Para as tuplas de atributos de L_{cn}, é executado o teste de correlação de Pearson (p) a fim de estabelecer se o par de atributos é elegível (se $p \geq \lambda$) para a segunda etapa do algoritmo. Se o par for elegível, sua correlação p e o par são adicionados à lista L_{cn}. Já para as tuplas de L_{cnp}, é executado o teste de Spearman (s) para verificar a elegibilidade do par (se $s \geq \lambda$). Caso seja elegível, a correlação s e o par são adicionados à L_{fnp}. Uma etapa adicional executa a limpeza de L_{fnp}, aonde se um atributo faz parte de mais de um par, apenas o par com maior correlação é mantido. Ao final, os atributos que não forem elegíveis, em ambos os casos, são removidos das respectivas listas e ao final, concatenados em uma lista L_{danger} de atributos que não são binários e não apresentam correlação. S_{danger} dá origem a uma nova lista $L_{fdanger}$, que contém todos os atributos de L_{danger}, sua moda e seu desvio padrão. Este processo é realizado pela função descrita em pseudo-código no Algoritmo 2.

O próximo algoritmo a ser executado necessita de duas outras funções auxiliares para seu correto funcionamento.

A primeira delas é representada pelo Algoritmo 3, que calcula os pares-ordenados

Algorithm 2 RFAO

função TESTECORRELACAO(L_n, L_{np})
  $p \leftarrow$ coeficiente de correlação de Pearson
  $s \leftarrow$ coeficiente de correlação de Spearman
  $\lambda \leftarrow$ correlação mínima esperada
  $L_{fp} \leftarrow$ lista final de elementos com correlação de Pearson $\geq \lambda$
  $L_{fs} \leftarrow$ lista final de elementos com correlação de Spearman $\geq \lambda$
  $L_{cn} \leftarrow C_{(L_{n[i]},2)}$
  $L_{cnp} \leftarrow C_{(L_{np[i]},2)}$
  $L_{fdanger} \leftarrow$ lista final de elementos que não possuem correlação $\geq \lambda$
  para cada $par_n \in L_{cn}$ **faça**
    se $p(par_n) \leq \lambda$ **então**
      $L_{cn} = L_{cn} - par_n$
    senão
      se não existe outro par $\in L_{fn}$ contendo algum dos atributos do par_n **então**
        $L_{fn} = L_{fn} \cup (p, par_n)$
      senão
        manter apenas o par com maior p
      fim se
    fim se
  fim para
  para cada $par_{np} \in L_{cnp}$ **faça**
    se $s(par_{np}) \leq \lambda$ **então**
      $L_{cnp} = L_{cnp} - par_{np}$
    senão
      se não existe outro par $\in L_{fnp}$ contendo algum dos atributos do par_{np} **então**
        $L_{fnp} = L_{fnp} \cup (s, par_{np})$
      senão
        manter apenas o par com maior s
      fim se
    fim se
  fim para
  $L_{danger} = L_{cn} + L_{cnp}$
  para cada $tupla_d \in L_{danger}$ **faça**
    $L_{fdanger} = L_{fdanger} \cup (tupla_d[0], tupla_d[0].moda, tupla_d[0].desvio\_padrao)$
    $L_{fdanger} = L_{fdanger} \cup (tupla_d[1], tupla_d[1].moda, tupla_d[1].desvio\_padrao)$
  fim para
  retorne (L_{danger}, L_{fn}, L_{fnp})
fim função

máximos e mínimos formados por cada tupla de atributos elegíveis para regressão. Esta função utiliza as seguintes variáveis para seu funcionamento: Fc_{tupla}, P_{min}, P_{max}, $D_{(p_{min},p_{max})}$, T_{extra}, $N_{x_{max}}$, $N_{y_{max}}$, $N_{x_{min}}$, $N_{y_{min}}$, N_{pmax} e N_{pmin}. Elas representam: a função de regressão obtida com a tupla, o par-ordenado mínimo e o máximo, a distância entre eles, o tamanho que a distribuição irá expandir no espaço de características, as novas coordenadas para os novos pontos máximos e mínimos e por fim, o novo ponto máximo e mínimo, respectivamente.

A segunda função que algoritmo necessita, é exposta no Algoritmo 4, que recebe como entrada a saída do Algoritmo 3, ou seja: $P_{min}, P_{max}, N_{pmin}, N_{pmax}$, além de outros valores, como ψ, a tupla de atributos em questão e a função de regressão obtida no Algoritmo 3 e retorna duas listas, cada lista contendo ψ valores gerados sinteticamente para a tupla de atributos informada, sendo que cada lista contém metade dos atributos gerados acima do par-ordenado máximo e outra metade abaixo do par-ordenado mínimo.

Tendo definidas as duas funções auxiliares, o Algoritmo 5 pode ser executado. Inicialmente é calculado o número ψ de instâncias sintéticas que serão geradas. De acordo com as características dos atributos, são geradas as instâncias sintéticas S_{synth}, que por fim, são adicionadas à S_{maj} e retornadas no formado de um *batch* balanceado $S_{balanced}$, de acordo com o algoritmo 5. S_{synth} é um dicionário, aonde os índices representam os atributos, por exemplo: $S_{synth}[A1]$ retorna todos os valores referentes ao atributo A1.

Algorithm 3 RFAO

função NovosPontosMaxMin(*tupla*)
  $Fc_{tupla} \leftarrow$ função de regressão entre os elementos da *tupla*
  $T_{extra} \leftarrow$ o tamanho que a distribuição deverá expandir
  $N_{xmax} \leftarrow$ coordenada x do novo ponto máximo
  $N_{ymax} \leftarrow$ coordenada y do novo ponto máximo
  $N_{xmin} \leftarrow$ coordenada x do novo ponto mínimo
  $N_{ymin} \leftarrow$ coordenada y do novo ponto mínimo
  $N_{pmax} \leftarrow$ novo ponto máximo
  $N_{pmin} \leftarrow$ novo ponto mínimo
  $P_{min} \leftarrow$ o ponto mínimo de $(tupla[0], tupla[1])$
  $P_{max} \leftarrow$ o ponto máximo de $(tupla[0], tupla[1])$
  $D_{(p_{min},p_{max})} =$ distância euclidiana entre P_{min} e P_{max}
  $T_{extra} = D_{(p_{min},p_{max})} \times \delta \div 100$
  $N_{x_{max}} = P_{max}.x + \frac{T_{extra}}{2} \times \cos(\hat{a}ngulo(Fc_{tupla}))$
  $N_{y_{max}} = P_{max}.y + \frac{T_{extra}}{2} \times \sin(\hat{a}ngulo(Fc_{tupla}))$
  $N_{x_{min}} = P_{min}.x + \frac{T_{extra}}{2} \times \cos(180 - \hat{a}ngulo(Fc_{tupla}))$
  $N_{y_{min}} = P_{min}.y + \frac{T_{extra}}{2} \times \sin(180 - \hat{a}ngulo(Fc_{tupla}))$
  $N_{pmax} = (N_{x_{max}}, N_{y_{max}})$
  $N_{pmin} = (N_{x_{min}}, N_{y_{min}})$
  retorne $((P_{min}, P_{max}), (N_{pmin}, N_{pmax}), Fc_{tupla})$
fim função

Algorithm 4 RFAO

função CRIARSYNTH(P_{min}, P_{max}, N_{pmin}, N_{pmax}, ψ, $Tupla$, $RegFunc$)
    $L_{at0} \leftarrow$ irá receber uma lista de valores sintéticos gerados para o primeiro atributo da tupla
    $L_{at1} \leftarrow$ irá receber uma lista de valores sintéticos gerados para o segundo atributo da tupla
    aux e $aux2 \leftarrow$ são variáveis para cálculos auxiliares
    para i $\leftarrow$ 1 **até** $int(\frac{\psi}{2})$ **faça**
        aux = número aleatório $\in [P_{max}[0], N_{pmax}[0]]$
        $aux2$ = valor recebido em $RegFunc(aux)$
        $aux2 = aux2 +-$ um valor aleatório $\in$ [0, $Tupla[1].desvio\_padrao$]
        $L_{at1} = L_{at1} \cup aux2$
        aux = número aleatório $\in [P_{max}[1], N_{pmax}[1]]$
        $aux2$ = valor recebido em $RegFunc(aux)$
        $aux2 = aux2 +-$ um valor aleatório $\in$ [0, $Tupla[0].desvio\_padrao$]
        $L_{at0} = L_{at0} \cup aux2$
        aux = número aleatório $\in [P_{min}[0], N_{pmin}[0]]$
        $aux2$ = valor recebido em $RegFunc(aux)$
        $aux2 = aux2 +-$ um valor aleatório $\in$ [0, $Tupla[1].desvio\_padrao$]
        $L_{at1} = L_{at1} \cup aux2$
        aux = número aleatório $\in [P_{min}[1], N_{pmin}[1]]$
        $aux2$ = valor recebido em $RegFunc(aux)$
        $aux2 = aux2 +-$ um valor aleatório $\in$ [0, $Tupla[0].desvio\_padrao$]
        $L_{at0} = L_{at0} \cup aux2$
    fim para
    retorne (L_{at0}, L_{at1})
fim função

Por fim, o Algoritmo 6 faz a chamada de todas as outras funções e retorna a ***stream*** balanceada.

Com estes passos é possível cobrir todos os casos de atributos numéricos, gerando sinteticamente atributos binários, pares que apresentam correlação e atributos não correlacionados.

O próxima seção irá trazer um estudo de caso utilizando o método em uma base fictícia, a fim de demonstrar como as instâncias são geradas pelo RFAO.

3.1.1 Estudo de caso para a versão em *batch*

A fim de demonstrar o funcionamento do RFAO, esta seção irá apresentar o algoritmo passo a passo em uma base de dados fictícia como segue na tabela 3. Em detalhes, podem ser visualizados os atributos e seus respectivos valores na tabela 4.

Seja esta a base de dados S. A base apresenta o nível de desbalanceamento 1:1.5. Supondo que o nível δ de balanceamento desejado seja fixado em 100, ou seja, a proporção

Algorithm 5 RFAO

1: **função** GERARSYNTH($L_b, L_{danger}, L_{fn}, L_{fnp}$)
2: ψ representa a quantidade de instâncias sintéticas serão geradas
3: S_{synth} é um dicionário que contém todas as instâncias sintéticas geradas, os índices representam os atributos e contém os valores definidos para os mesmos
4: $\psi = |S| * \delta \div 100 - |S_{min}|$
5: **para cada** $tupla_b \in L_b$ **faça**
6: $S_{synth}[tupla_b[0]] \leftarrow \psi$ valores para de acordo com as probabilidades em $tupla_b[1]$
7: **fim para**
8: **para cada** $tupla_{fd} \in L_{fdanger}$ **faça**
9: $S_{synth}[tupla_{fd}[0]] \leftarrow \psi$ valores para de acordo com a $tupla_{fd}[1]$ (moda) +− um número aleatório $\in [0, tupla_{fd}[2]]$
10: **fim para**
11: **para cada** $tupla_n \in L_{fn} \cup L_{fnp}$ **faça**
12: $Np = NovosPontosMaxMin(tupla_n)$
13: $Gerar_{synth} = CriarSynth(Np[0][0], Np[1][0], Np[0][1], Np[1][1], \psi, tupla_n, Np[2])$
14: $S_{synth}[tupla_n[0]] = Gerar_{synth}[0]$
15: $S_{synth}[tupla_n[1]] = Gerar_{synth}[1]$
16: **fim para**
17: **para cada** $elemento \in L_{danger}$ **faça**
18: $S_{synth}[elemento][0] \leftarrow \psi$ instâncias sintéticas no geradas de acordo com $elemento[1]$ + − um valor aleatório $\in [0, elemento[2]$
19: **fim para**
20: **retorne** (S_{synth})
21: **fim função**

Algorithm 6 RFAO

1: **função** RFAO(S)
2: $S \leftarrow$ representa uma base ou *stream*
3: Separar S em S_{min} e S_{maj}
4: $tipos = TipoAtributo(S_{min}$, lista atributos de S)
5: $correlacoes = TesteCorrelacao(tipos[1], tipos[2])$
6: $S_{synth} = GerarSynth(tipos[0], correlacoes[0], correlacoes[1], correlacoes[2])$
7: **retorne** ($S \cup S_{synth}$)
8: **fim função**

Tabela 3 – Descrição da base

Instâncias	Attributes	Classe 0	Classe 1
20	4	12	8

Tabela 4 – Base de teste

A1	A2	A3	A4	Classe
0	1	7.3	7	0
0	2	8.6	6	0
0	32	64.0	1	1
1	36	72.0	1	1
0	3	9.9	2	0
0	4	11.2	4	0
0	5	12.5	4	0
1	29	58.0	1	1
0	32	64.0	4	1
0	6	13.8	2	0
0	27	54.0	3	1
0	7	15.1	3	0
0	8	16.4	1	0
0	9	17.7	8	0
1	38	76.0	5	1
1	43	86.0	12	1
0	10	19.0	5	0
0	11	20.3	14	0
0	45	90.0	12	1
1	12	24	11	0

desejada é de 1:1, e o nível mínimo estipulado para correlação λ seja de 80% (0.8). Quando apresentada para o RFAO, no primeiro teste, que é referente ao tipo de atributo, ele retorna que o atributo A1 é binário, contendo 50% de chance de ocorrência para o valor 0 e para o valor 1 em S_{min}. A2, A3 e A4, não apresentam comportamento normal, apresentando os seguintes valores no teste de normalidade: 0.13, 0.13 e 0.21, respectivamente, sendo inferior ao λ estipulado. No segundo teste, responsável por estabelecer se existe correlação entre os atributos, os atributos A2 e A3, que não apresentam comportamento normal, tem correlação de Spearman de 100%, logo, devem ser separados para geração segundo função de regressão.

A4 não apresenta λ acima do estipulado, logo deve ser gerada segundo moda e desvio padrão. Sua relação com A2 e A3 pode ser observada na Figura 5 e Figura 6, respectivamente. A moda tem valor 6, como pode ser observado no histograma apresentado na Figura 7 e o desvio padrão tem valor aproximado de 4. Definida a forma com que vão ser gerados os valores para os atributos e coletadas a devidas informações é iniciada a etapa de geração das instâncias sintéticas.

Para etapa de geração de instâncias sintéticas, primeiramente se deve calcular a quantidade ψ de instâncias a serem geradas. Para S, ψ é 4. O processo de geração é iniciado pela criação de atributos binários. Como os valores sintéticos de A1 devem ser criados com probabilidade de 50% de ocorrência, o algoritmo retornou 2 valores 0 e 2 valores 1. O par-ordenado mínimo de A2 e A3 para S_{min} é $(27, 54.0)$ e o máximo $(45, 90.0)$. A distância entre estes pares ordenados é aproximadamente 40. Como $|S_{min}|$, que representa aproximadamente $\frac{2}{3}$ de $|S_{maj}|$ e segundo parâmetro λ informado, deve representar 100%, logo deve possuir $\frac{4}{3}$ de seu tamanho original, o que implica em crescer para o sentido positivo e negativo aproximadamente 7 unidades de distância. Como a função de regressão encontrada é $y = x + 2$, O ângulo formado pelos valores de A2 e A3 é de 45 graus, logo o novo par-ordenado máximo é (50, 100) e o mínimo é (22, 44). Os novos valores para os atributos A2 e A3 devem ser gerados nesta faixa. Ao executar o algoritmo, foram gerados os seguintes valores para A2: 24, 25, 54 e 56. Para A3 foram gerados: 48.0, 50.0, 108.0 e 112.0. Por fim, para A4 os atributos são gerados a partir da moda +- um valor entre zero e o desvio padrão. Aplicando o algoritmo os valores gerados foram: 2, 4, 7 e 10.

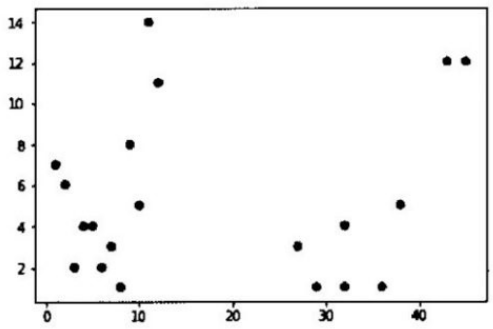

Figura 5 – Atributos A2 e A4

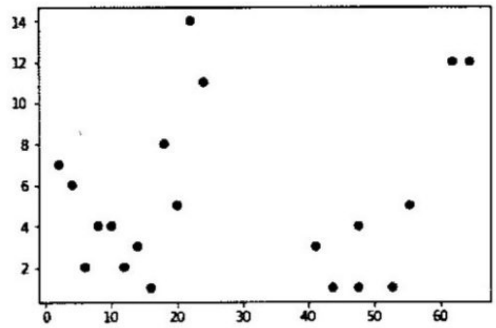

Figura 6 – Atributos A3 e A4

Com todos os atributos sintéticos gerados, o RFAO devolve a base S balanceada. O resultado pode ser observado na Tabela 5, que apresenta a distribuição desejada de 1:1.

A distribuição das instâncias da base original tal como a distribuição após aplicação do RFAO podem ser observadas graficamente na Figura 8.

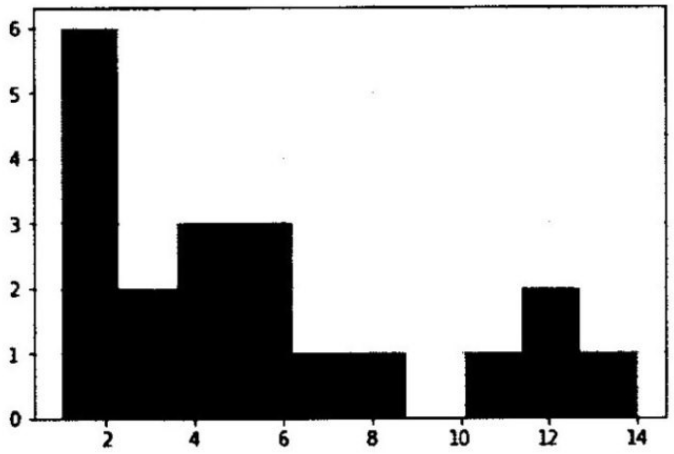

Figura 7 – Histograma de freqência de A4

Tabela 5 – Base de teste

A1	A2	A3	A4	Classe
0	1	7.3	7	0
0	2	8.6	6	0
0	32	64.0	1	1
1	36	72.0	1	1
0	3	9.9	2	0
0	4	11.2	4	0
0	5	12.5	4	0
1	29	58.0	1	1
0	32	64.0	4	1
0	6	13.8	2	0
0	27	54.0	3	1
0	7	15.1	3	0
0	8	16.4	1	0
0	9	17.7	8	0
1	38	76.0	5	1
1	43	86.0	12	1
0	10	19.0	5	0
0	11	20.3	14	0
0	45	90.0	12	1
1	12	24	11	0
1	24	48.0	2	1
0	25	50.0	4	1
1	26	52.0	7	1
0	27	64.0	10	1

3.1.2 Discussão e limitações da implementação em *batch*

O RFAO se propõe resolver alguns problemas do SMOTE, como por exemplo, aumento da representatividade e correta geração de atributos binários, o que ocorreu no estudo de caso apresentado na seção 3.1.1. Quando as classes são perfeitamente separáveis,

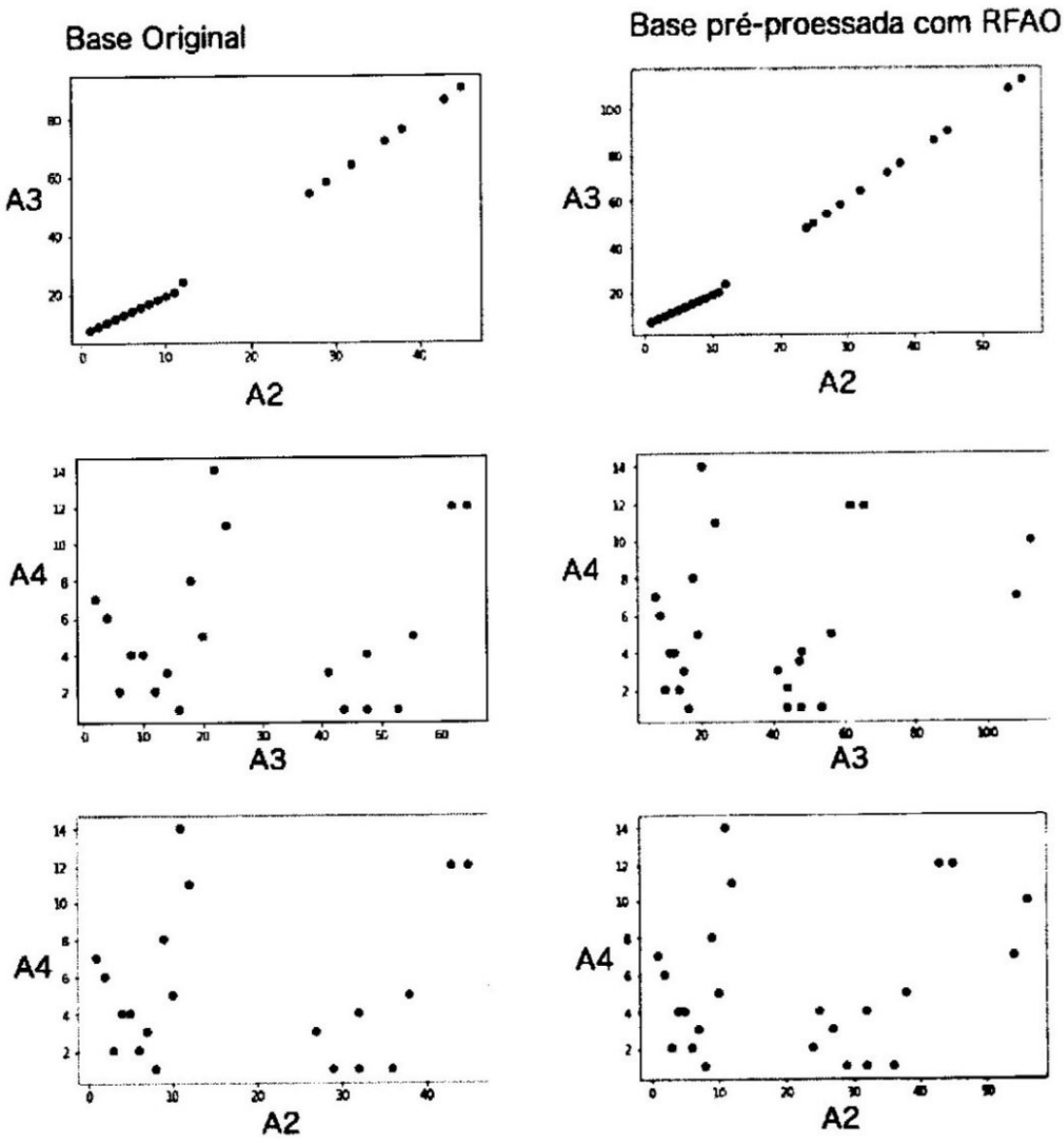

Figura 8 – Distribuição das instâncias por pares no espaço de características por feature

a maneira com que são geradas as instâncias sintéticas pelo RFAO não traz consequências para a correta classificação das instâncias de S_{maj}, porém, caso exista sobreposição, pode existir deterioramento da performance em S_{maj}. O RFAO consegue trabalhar de mancira probabilística com dados binários ou que não apresentam correlação entre as características, logo pode ter sua performance deteriorada caso existam *outliers*, caso contrário, os valores para os atributos gerados continuarão dentro da faixa de valores aceitáveis. Outro fator a ser observado é que existe a necessidade de se estabelecer uma faixa de variação para a geração de valores segundo a função de regressão, o que aumentaria ainda mais a possibilidade de aumento na representatividade de S_{min} (aumento da área coberta pela região do espaço de características), porém, isto aumentaria a chance de sobreposição entreas classes.

Na Figura 8, para os atributos A2 e A3, é possível observar que S_{min} se torna

mais representativa, aumentando sua densidade no espaço de características, sem mudar seu comportamento padrão. Ao analisar o par de atributos A3 e A4, nota-se que A3 tornou-se mais denso, apesar de ter dois novos atributos com valor maior do que 100. Estes valores altos foram induzidos pela maneira com que são geradas as instâncias que possuem correlação, ou seja, a partir do par-ordenado máximo e mínimo. Uma alternativa para contornar este comportamento seria gerar os novos valores a partir do par-ordenado médio. Este comportamento pode ser observado também nos valores gerados para A2, porém, devido a distribuição estar delimitada em uma região menor, o impacto não foi tão evidente.

3.2 RFAO para *streams*

Após o protótipo ser desenvolvido para o ambiente de *batch*, buscou-se fazer a adaptação do método para o ambiente de *streams*. Uma das maiores dificuldades neste processo, se deve ao fato de que no ambiente *batch*, existe acesso a todas as instâncias minoritárias na etapa de geração das instâncias sintéticas, já em fluxos contínuos de dados, o acesso as instâncias ocorre de maneira gradual, o que adiciona um complicador extra para aplicação de técnicas de re-amostragem.

A fim de solucionar este problema, foi adotada a estratégia de manter uma janela deslizante (*window*), contendo as últimas instâncias observadas. Para problemas de desbalanceamento severo, o número de instâncias minoritárias encontrado em cada janela pode ser muito pequeno para que padrões de comportamento sejam observados. Esta implementação oferece a possibilidade de carregar uma segunda janela deslizante, contendo todas as instâncias minoritárias observadas na *stream* (*keepMinorityBatch*). Com a estratégia de janelamento, é possível simular *mini-batches* de dados, e assim, realizar as análises necessárias para o funcionamento do método.

Após os resultados obtidos com a versão *batch*, apresentados na seção 4.2, decidiu-se que a versão para *streams* deve seguir o mesmo modelo da implementação para *batch*, recebendo o tamanho que a classe minoritária deve possuir em relação a classe majoritária em porcentagem (*ratio* δ) e a correlação esperada para geração via funções de regressão (*expectedCorrelation* λ), porém algumas funcionalidades novas foram adicionadas. Na seção 3.1, foi definido que todos os atributos que não possuem correlação com outro par, deveriam ser gerados segundo moda somada a uma constante definida na faixa de valores correspondente a $[-STDEV, +STDEV]$, porém, após os resultados preliminares, uma verificação extra foi adicionada: caso os atributos sejam nominais e não sejam correlacionados com qualquer outro atributo, os valores devem ser gerados segundo probabilidade de ocorrência observada na classe minoritária, na janela deslizante atual. A expansão dos atributos gerados via RLS, anteriormente calculada dinamicamente, como mostra o algoritmo 3,

Tabela 6 – Parâmetros e valores *default* do algoritmo RFAO para *streams*

Parâmetro	Valor *default*
baseLearner	HoeffdingTree
window	500
keepMinorityBatch	False
ratio	1.0
expectedCorrelation	0.7
expandCorrelatedAttributes	0.1
applyBalance	True
ensureNeighborhood	True
knn	3
minNeighbors	2
randomUndersampling	False
percentOfMajorityInstancesToKeep	0.75

foi adaptada para ser recebida como parâmetro *expandCorrelatedAttributes*($\varsigma \in [0, 1]$). Além disto, foi incluída ainda na etapa de geração de sintéticas, a possibilidade de verificar se a instância gerada está posicionada em uma região segura do espaço de características (*ensureNeighborhood*), utilizando o algoritmo K-NN. Caso o usuário opte por executar esta estapa, o número de vizinhos que a instância deve ser comparada (*knn*) tal como o número mínimo de vizinhos pertencentes a classe minoritária próximos de cada nova instância gerada (*minNeighbors*) deve ser informado. Por fim, parâmetros para controlar a expansão dos atributos gerados via função de regressão (*expandCorrelatedAttributes*) e a possibilidade de executar o RUS (*randomUndersampling*) da classe majoritária foram adicionados, tornando o RFAO um método híbrido de re-amostragem. A tabela 6 traz todos os parâmetros do método e seus respectivos valores *default*.

O diagrama de funcionamento do método para *streams* foi dividido em duas etapas: processo de treinamento e *oversampling*. O início e fim do algoritmo estão indicados. As decisões, por sua vez, estão indicados pela cor amarela. Laços de repetição em laranjado, e por fim, o treinamento está indicado em azul e os demais processos em cinza.

O processo de treinamento, observado no diagrama 9, é um processo cíclico que é executado a cada nova instância recebida. Este processo é baseado em criar uma janela deslizante com tamanho ϖ definido pelo usuário, e utiliza-la para extrapolar padrões nos dados. A atualização do modelo preditor, tal como a geração de instâncias sintéticas ocorre a cada intervalo de observação de instâncias, baseado no tamanho de ϖ. O processo de treinamento pode ser observado no algoritmo *TrainOnInstance* 7.

Uma série de funções auxiliares são executadas no algoritmo *TrainOnInstance*. A primeira delas, *verificarClasseMajoritariaEm*, detalhada no algoritmo 8, verifica quem é a classe majoritária e minoritária dentro do *batch* de dados atual, recebendo como entrada, o *batch* que deve ser analisado. A próxima função executada, recebe o *batch* a ser

Algorithm 7 RFAOStream

Entrada: *baseLearner*, ϖ, *keepMinorityBatch*, δ, , *expandCorrelatedAttributes*, *applyBalance*, *ensureNeighborhood*, *knn*, *minNeighboors*, *randomUndersampling*, *percentOfMajorityInstancedToKeep*

1: $dicionarioEstatisticas \leftarrow \emptyset$
2: $instanciasObservadas \leftarrow 0$
3: $instanciasAGerar \leftarrow 0$
4: **função** TRAINONINSTANCE($\vec{x}_i$)
5: **se** $InstanciasObservadas == 0$ **então**
6: **Instanciar** *batch*, $batchMin$, $batchMaj$, $windowMin$ e $batchSynth$
7: **fim se**
8: **se** $InstanciasObservadas$ **MOD** $\varpi == 0$ **então**
9: $verificarClasseMajoritariaEm(batch)$
10: **se** *applyBalance* **então**
11: **se** *randomUndersampling* **então**
12: treine o classificador com $rus(batchMaj)$
13: **fim se**
14: $instanciasAGerar = calcularInstanciasAGerar()$
15: **se** $instanciasAGerar > 0$ **então**
16: $getStatistics(\vec{x}_i)$
17: $correlationTest(atributosNormais, tipo = Normal)$
18: $correlationTest(atributosNaoNormais, tipo = NaoNormal)$
19: $setProbabilities()$
20: $generateSynthInstances()$
21: **fim se**
22: **senão**
23: treine o classificador com *batch*
24: **fim se**
25: **fim se**
26: $instanciasObservadas++$
27: $atualizarJanela()$
28: **fim função**

reduzido e realiza o RUS. Seu algoritmo está apresentado em 9. Na sequência, calcula-se $instanciasAGerar$ com base no algoritmo 10. As funções seguintes são responsáveis por realizar o procedimento de *oversampling*, visível também, através do diagrama CITAR. Este processo se inicia obtendo as informações estatísticas de cada atributo f_j, demonstrado pelo algoritmo 11, que recebe como entrada uma instância $\vec{x}_i$, para referência dos atributos correntes. Como na implementação apresentada na seção anterior, testes de correlação devem ser executados, a fim de determinar quais atributos serão gerados via RLS. Este procedimento é realizado pelo algoritmo 12. Como última etapa antes da geração de sintéticas, a função *setProbabilities*, representada pelo algoritmo 13, é executada. O objetivo desta função é determinar a probabilidade de ocorrência dos valores possíveis, observados até o momento, para cada atributo nominal que não seja correlacionado com nenhum outro atributo. A geração de instâncias sintéticas, representada no algoritmo 14,

representa o fim do processo de *oversampling*. Por fim, a janela deslizante, representada pela função *atualizarJanela* é mostrada no algoritmo 15.

Algorithm 8 RFAOStream

Entrada: *batch* de dados

função $verificarClasseMajoritariaEm(batch)$
  $L_{classes} \leftarrow$
  $Freq_0 \leftarrow 0$
  $Freq_1 \leftarrow 0$
  para cada $instancia \in batch$ **faça**
    se $instancia.classe \notin L_{classes}$ **então**
      $L_{classes} \cup instancia.classe$
    fim se
  fim para
  para cada $instancia \in batch$ **faça**
    se $instancia.classe == L_{classes}[0]$ **então**
      $Freq_0 + +$
    senão
      $Freq_1 + +$
    fim se
  fim para
  se $Freq_0 > Freq_1$ **então**
    $C_{maj} = L_{classes}[0]$
    $C_{min} = L_{classes}[1]$
  senão
    $C_{maj} = L_{classes}[1]$
    $C_{min} = L_{classes}[0]$
  fim se
fim função

Algorithm 9 RFAOStream

Entrada: *batch* de dados a ser reduzido

função $rus(batch)$
  $numeroDeInstanciasADeletar \leftarrow |batchMaj| - (int)(|batchMaj| * percentOfMajorityInstancedToKeep)$
  $indiceAleatorio \leftarrow 0$
  para $i = 0 \leftarrow i < numeroDeInstanciasADeletar$ **até** $i + +$ **faça**
    $indiceAleatorio =$ inteiro aleatório $\in [0, |batch|]$
    $batch = batch - batch[indiceAleatorio]$
  fim para
  retorne *batch*
fim função

3.2.1 Estudo de caso para versão em *streams*

Com a finalidade de demonstrar o funcionamento do método em *streams*, a base Pima Indians Diabetes foi escolhida, pois apresenta atributos correlacionados e desbalan-

Algorithm 10 RFAOStream

1: **função** *calcularInstanciasAGerar*
2: $instanciasAGerar = (int)(|batchMaj| * ratio)$
3: **se** $instanciasAGerar <= |batchMin|$ **então**
retorne 0
4: **senão**
retorne $instanciasAGerar$
5: **fim se**
6: **fim função**

Algorithm 11 RFAOStream

Entrada: $\vec{x}_i$

1: **função** $getStatistics(\vec{x}_i)$
2: $L_{atributos} \leftarrow$ atributos contidos em $\vec{x}_i$
3: $L_n \leftarrow \emptyset$
4: $L_{np} \leftarrow \emptyset$
5: **para cada** $f_i \in L_{atributos}$ **faça**
6: **se** $f_i \neq$ classe **então**
7: $dicionarioEstatisticas \cup ((\vec{x}_i)$,moda, média, desvio padrão, mínimo e máximo de f_j observado em $batchMin$ ou $windowMin$, caso $keepMinorityBatch$
8: **fim se**
9: **se** f_j apresenta comportamento **normal então** $L_n \cup f_i$
10: **senão**$L_{np} \cup f_i$
11: **fim se**
12: **fim para**
13: **fim função**

ceamento. Como esta base possui apenas 8 atributos, todos numéricos e 768 instâncias, sendo 500 pertencentes a classe majoritária (*tested_negative*) e 268 pertencentes a classe minoritária (*tested_positive*), o tamanho de janela definido para o RFAO foi de 50 instâncias. Este foi o único parâmetro alterado da configuração *default* do RFAO. De acordo com este tamanho de janela, a distribuição dos dados em cada batch pode ser observada na Tabela 7. Nota-se que o maior desbalanceamento encontrado se dá no décimo batch (1:4) e o menor no primeiro batch, aonde não ocorreu desbalanceamento. O desbalanceamento médio encontrado durante a *stream* foi de 1:2.

Durante a *stream*, dois pares de atributos apresentaram correlação: *Skin* e *Insu* apresentaram correlação média de 35% e *Preg* e *Age* apresentaram 10%, alcançando 89.8% e 76.7% em determinados *batches*, conforme pode ser observado nas Figuras 11 e 12.

Como o RFAO foi utilizado na configuração *default*, as instâncias geradas sinteticamente devem ter sua vizinhança assegurada com instâncias originais. Na figura 13, as variáveis *Skin* e *Insu* da base original contendo todas as instâncias existentes antes do tratamento, estão plotadas no espaço de características. A classe minoritária está representada em vermelho e a majoritária em azul. Após o RFAO ser utilizado na base,

Algorithm 12 RFAOStream

Entrada: $L_{atributos}, tipoDeCorrelacarATestar$

1: **função** $correlationTest(\vec{x}_i, tipoDeCorrelaçãoATestar)$
2: **para** $i = 0 \leftarrow i < (|L_{atributos}| - 1)$ **até** $i++$ **faça**
3: **para** $j = 1 \leftarrow j < (|L_{atributos}| - 1)$ **até** $j++$ **faça**
4: **se** $tipoDeCorrelacaoATestar == Normal$ **então**
5: **se** $p(f_i, f_j) > 0.05$ **então**
6: $L_{rls} \cup ((f_i, f_j), p)$
7: **fim se**
8: **senão**
9: **se** $s(f_i, f_j) > 0.05$ **então**
10: $L_{rls} \cup ((f_i, f_j), s)$
11: **fim se**
12: **fim se**
13: **fim para**
14: **fim para**
15: **para** $i = 0 \leftarrow i < (|L_{rls}| - 1)$ **até** $i++$ **faça**
16: **para** $j = i + 1 \leftarrow j < |L_{rls}|$ **até** $j++$ **faça**
17: **se** $\exists f_i \vee f_j \in L_{rls}[i][0] \wedge L_{rls}[j][0]$ **então**
18: **se** $L_{rls}[i][1] > L_{rls}[j][1]$ **então**
19: $L_{rls} - L_{rls}[j]$
20: **senão**
21: $L_{rls} - L_{rls}[i]$
22: **fim se**
23: **fim se**
24: **fim para**
25: **fim para**
26:
27: **fim função**

Algorithm 13 RFAOStream

Entrada: $\vec{x}_i$

1: **função** $setProbabilities(\vec{x}_i)$
2: $L_{val} \leftarrow$
3: $L_{prob} \leftarrow$
4: $dictValProb(L_{val}, L_{prob}) \leftarrow (,)$
5: **para cada** $atributo \in \vec{x}_i$ **faça**
6: **se** $atributo \cap L_{rls} = \emptyset$ **então**
7: **se** $atributo \neq classe$ **então**
8: **se** $atributo$ é nominal **então**
9: salvar os possíveis valores em L_{val}
10: salvar as probabilidades de ocorrência destes valores em L_{probl}
11: adicionar registro a $dictValProb(L_{val}, L_{prob})$
12: **fim se**
13: **fim se**
14: **fim se**
15: **fim para**
16: **fim função**

Algorithm 14 RFAOStream

função *generateSynth*
  enquanto$|batchSynth| < instanciasAGerar$
  para $i = 0 \leftarrow i < |L_{atributos}$ **até** $i++$ **faça**
    se $L_{atributos}[i] \in L_{rls}$ **então**
      Fixar $L_{atributos}[i]$ como Y e segundo atributo correlacionado em L_{rls} como X
      se correlação do par < 0 **então**
        $X_i \leftarrow minimo \times (1 + expandCorrelatedAttributes)$
      senão
        $X_i \leftarrow maximo \times (1 + expandCorrelatedAttributes)$
      fim se
      $valor \leftarrow$ regressao correspondente a X_i
    senão
      se $L_{atributos}[i]$ é nominal **então**
        $valor \leftarrow$ gerado de acordo com a probabilidade de ocorrência
      senão
        $valor \leftarrow$ gerado de acordo com a moda e $STDEV$
      fim se
    fim se
    Adicionar valor a nova instância $\vec{x}_{new}$
  fim para
  se *ensureNeighborhood* **então**
    se Vizinhança é segura **então**
      $batchSynth \cup \vec{x}_{new}$
    fim se
  senão
    $batchSynth \cup \vec{x}_{new}$
  fim se
  fim enquanto
fim função

a classe minoritária ganhou muito mais representatividade, seguindo o fluxo natural dos dados, como pode ser observado na figura 14. O mesmo ocorreu com os atributos *Preg* e *Age*, que na *stream* original possuíam o comportamento mostrado na figura 15 e após o balanceamento o comportamento mostrado na figura 16. Nota-se que em ambos os casos havia sobreposição de classes no espaço de características.

Com a finalidade de analisar o comportamento de outro par de atributos que não tivesse sido gerado por funções de regressão, os atributos *Pres* e *Skin* foram analisados. Seu comportamento na *stream* original pode ser observado na figura 17, e após o balanceamento, na figura 18. Como ocorreu para os atributos gerados por função de regressão, este par se expandiu no espaço de características mantendo traços de suas características originais e também possui sobreposição entre as classes desde a *stream* original.

Uma análise estatística dos atributos utilizados para geração via regressão, revelou

Algorithm 15 RFAOStream

Entrada: $\vec{x}_i$

função $atualizarJanela(\vec{x}_i)$
  se $\varpi \leq instanciasObservadas$ **então**
    $primeiraInstancia \leftarrow batch[0]$
    se $primeiraInstancia.classe == C_{min}$ **então**
      $batchMin = batchMin - batchMin[0]$
      $batchMin = batchMin \cup \vec{x}_i$
    senão
      $batchMaj = batchMaj - batchMaj[0]$
      $batchMaj = batchMaj \cup \vec{x}_i$
    fim se
    $batch = batch - batch[0]$
    $batch = batch \cup \vec{x}_i$
  senão
    $batch = batch \cup \vec{x}_i$
  fim se
  se $keepMinorityBatch$ **então**
    se $windowMin >= batch$ **então**
      se $primeiraInstancia.classe == C_{min}$ **então**
        $windowMin = windowMin - windowMin[0]$
        $windowMin = windowMin \cup \vec{x}_i$
      fim se
    fim se
  fim se
fim função

Tabela 7 – Pima Indians Diabetes - Numero de instâncias por batch

Batch	Majoritárias	Minoritárias	Sintéticas geradas	Total observado
1	25	25	0	50
2	38	12	26	100
3	34	16	18	150
4	28	22	6	200
5	29	21	8	250
6	32	18	14	300
7	30	20	10	350
8	32	18	14	400
9	30	20	10	450
10	40	10	30	500
11	37	13	24	550
12	37	13	24	600
13	35	15	20	650
14	32	18	14	700
15	31	19	12	750

que com a geração de instâncias sintéticas, todos os desvios padrão diminuíram, como pode ser observado na tabela 8. Este comportamento fornece indícios de que a maioria das

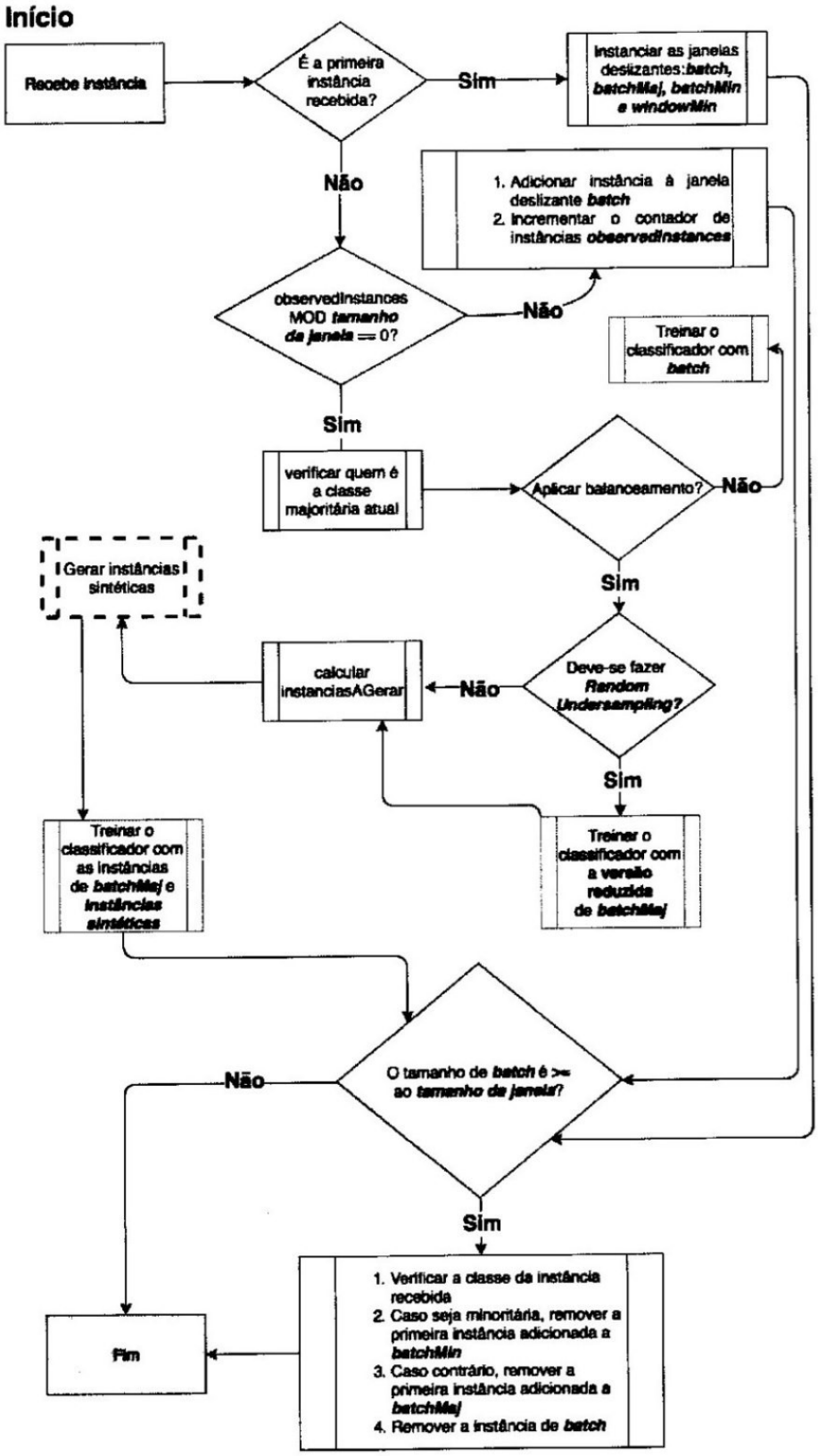

Figura 9 – Processo de treinamento do RFAO em *streams*

instâncias geradas encontra-se dentro da faixa de valores originais da *stream*. O atributo *Insu* obteve variação no valor mínimo, se comparado com a base original, porém esta variação está dentro da margem de expansão *default* do RFAO (10%).

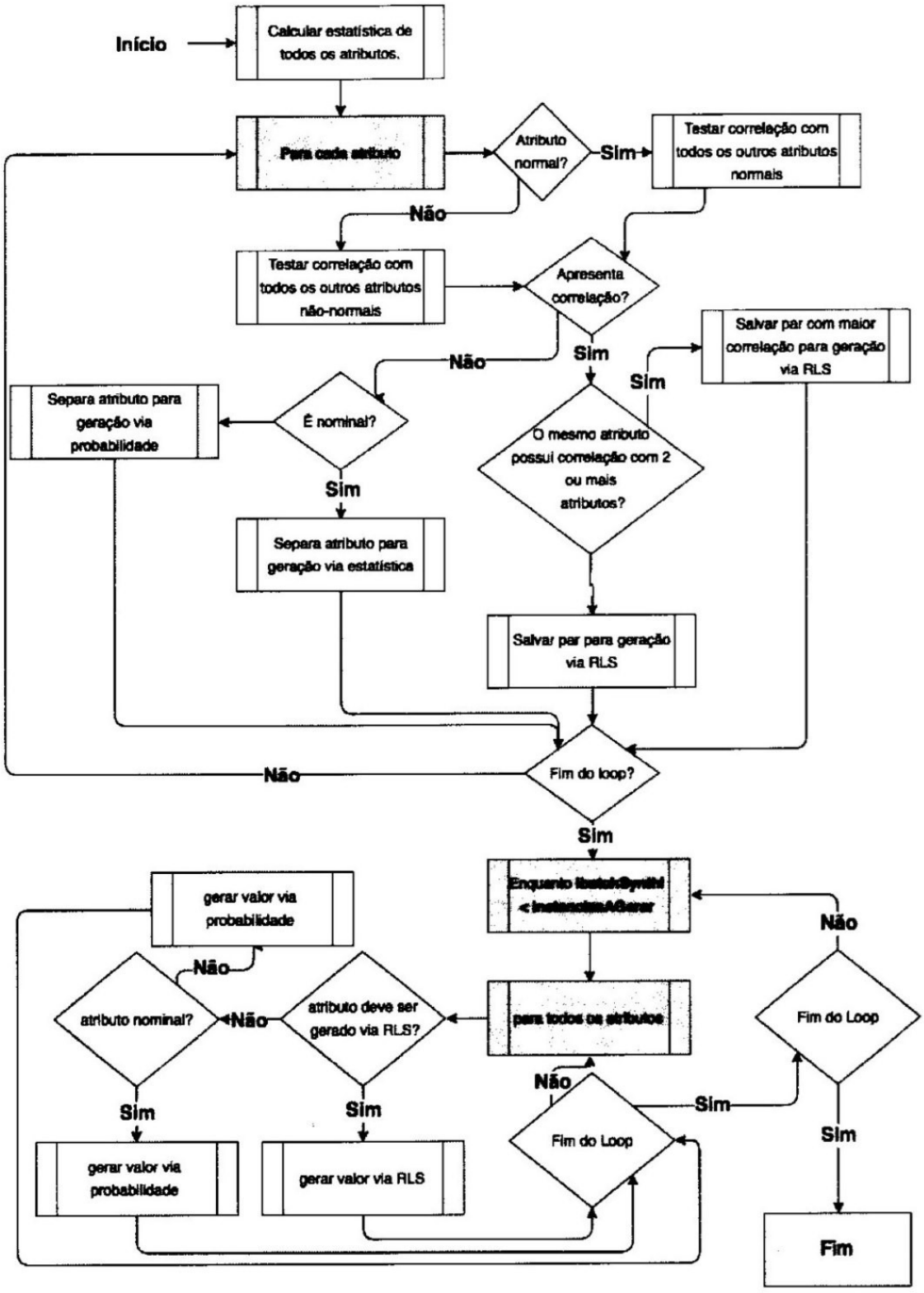

Figura 10 – Processo de *oversampling* do RFAO para *streams*

3.2.2 Discussão e limitações da implementação em *streams*

Para o estudo de caso da versão protótipo do RFAO, uma base de dados fictícia foi concebida. Como este protótipo foi desenvolvido para o ambiente *batch*, a base criada não precisava possuir muitas instâncias, ou mesmo atributos, para que fosse possível demonstrar

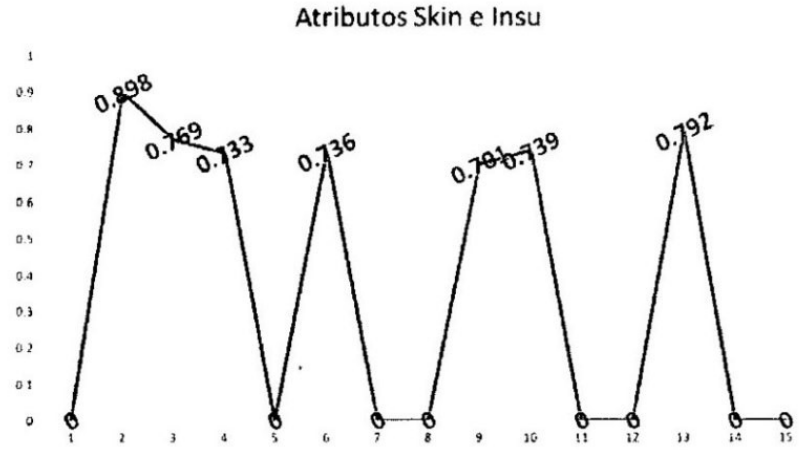

Figura 11 – Correlação entre os atributos observada durante a *stream*

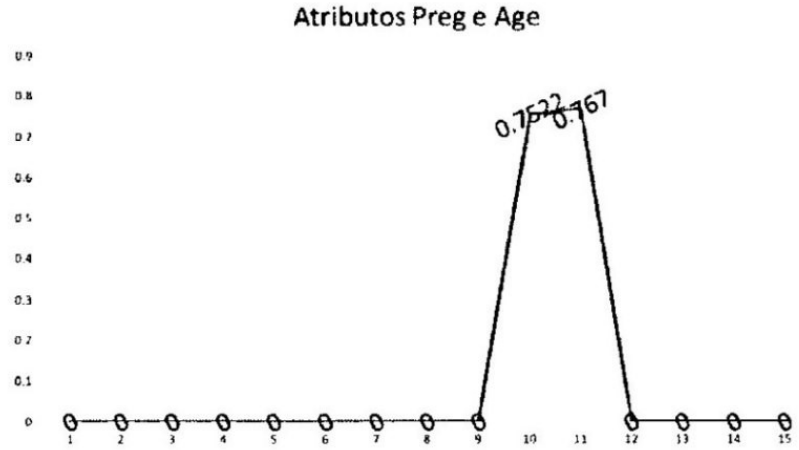

Figura 12 – Correlação entre os atributos observada durante a *stream*

Tabela 8 – Estatística dos atributos correlacionados

Atributo	Stream	Mínimo	Máximo	Média	Desvio Padrão
Skin	Original	0	99	20.53	15.95
Skin	Balaneada	−3.8	99	19.9	14.78
Insu	Original	0	846	79.79	115.24
Insu	Balanceada	−60.5	846	82.9	107.88
Preg	Original	0	17	3.84	3.37
Preg	Balanceada	0	17	3.81	3.24
Age	Original	21	81	33.24	11.76
Age	Balanceada	21	81	34.6	11.08
Pres	Original	0	122	69.1	19.35
Pres	Balanceada	0	122	69.18	18.73

o funcionamento do método. Já para a versão final, desenvolvida para o ambiente de *streams*, preferiu-se utilizar uma base de dados real com desbalanceamento intrínseco, simulada em forma de *stream*. Tal escolha se deve ao fato de que em bases reais existem problemas de separabilidade entre as classes, tal como as correlações existentes entre os

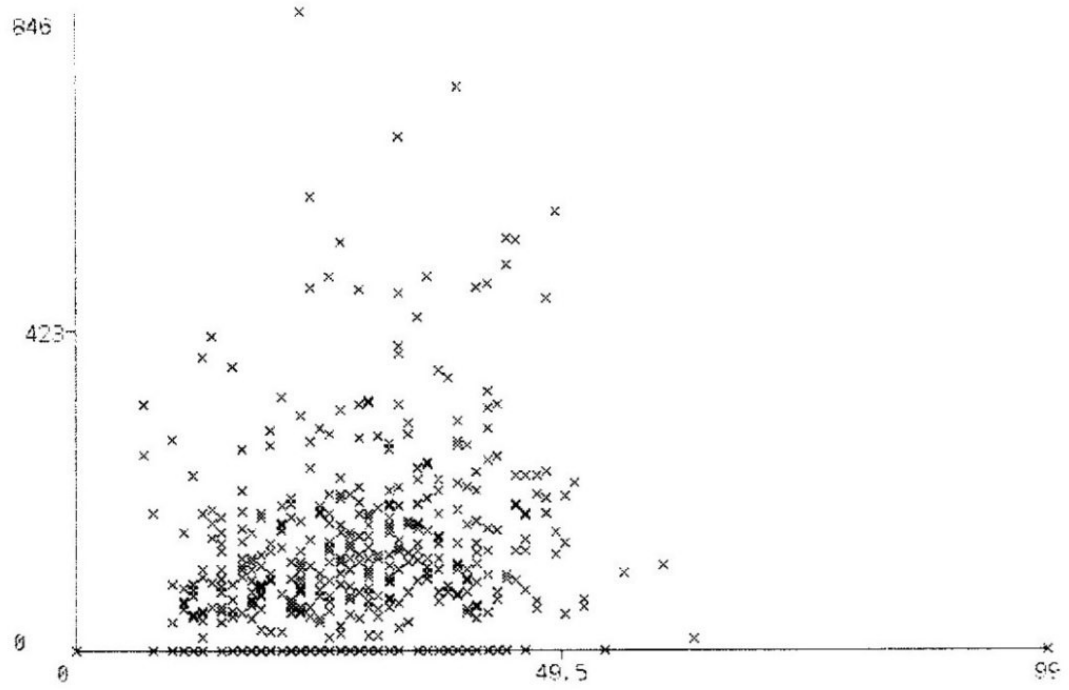

Figura 13 – Distribuição das variáveis *Skin* e *Insu* no espaço de características na base original

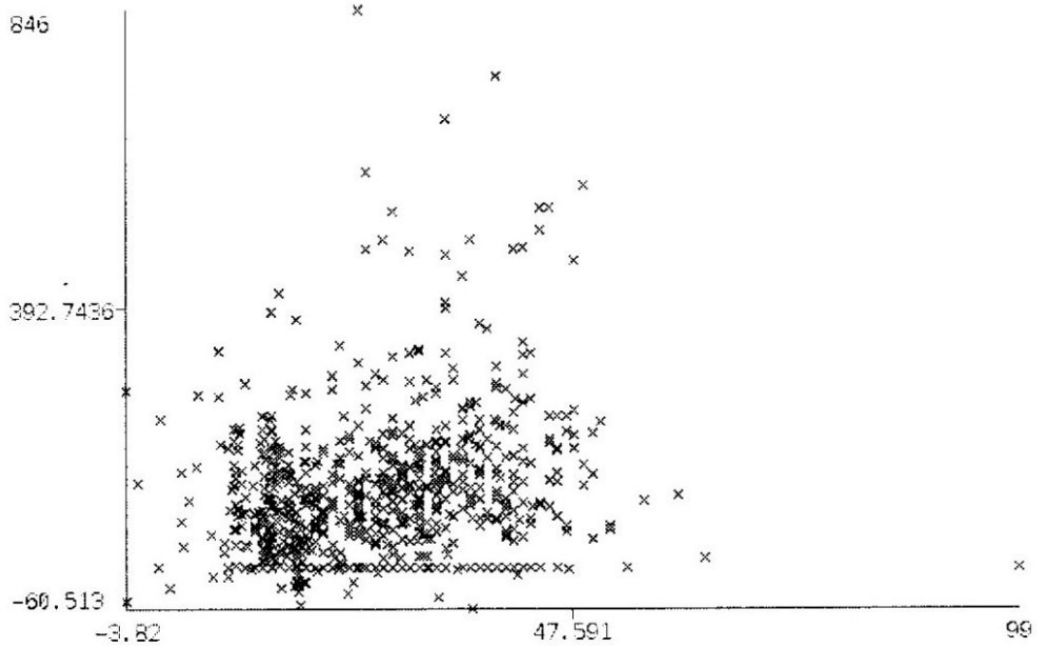

Figura 14 Distribuição das variáveis *Skin* e *Insu* no espaço de características na base balanceada

atributos são menores, o que na prática fornece um estrato mais próximo da realidade de como o método se comportará no mundo real. Outro fator que contribuiu para esta decisão foi o fato de que com uma base de dados maior, um fluxo contínuo de dados seria simulado por mais tempo, proporcionando uma análise mais detalhada. Porém, a base de dados utilizada não possui atributos nominais, o que inviabilizou a geração de valores de

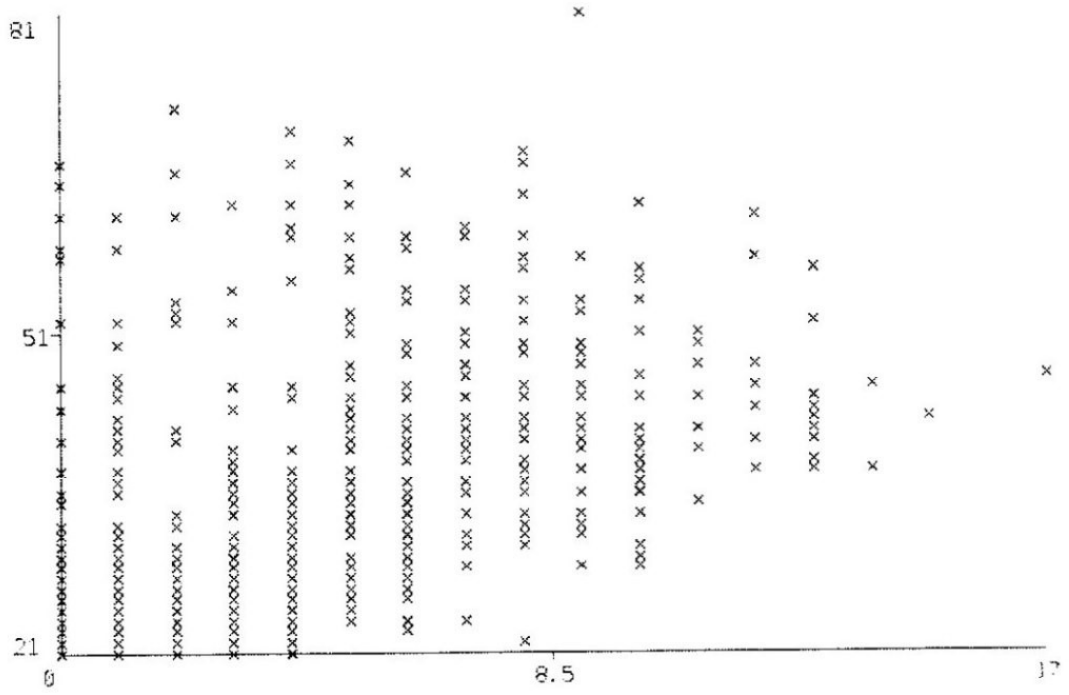

Figura 15 – Distribuição das variáveis *Preg* e *Age* no espaço de características na base original

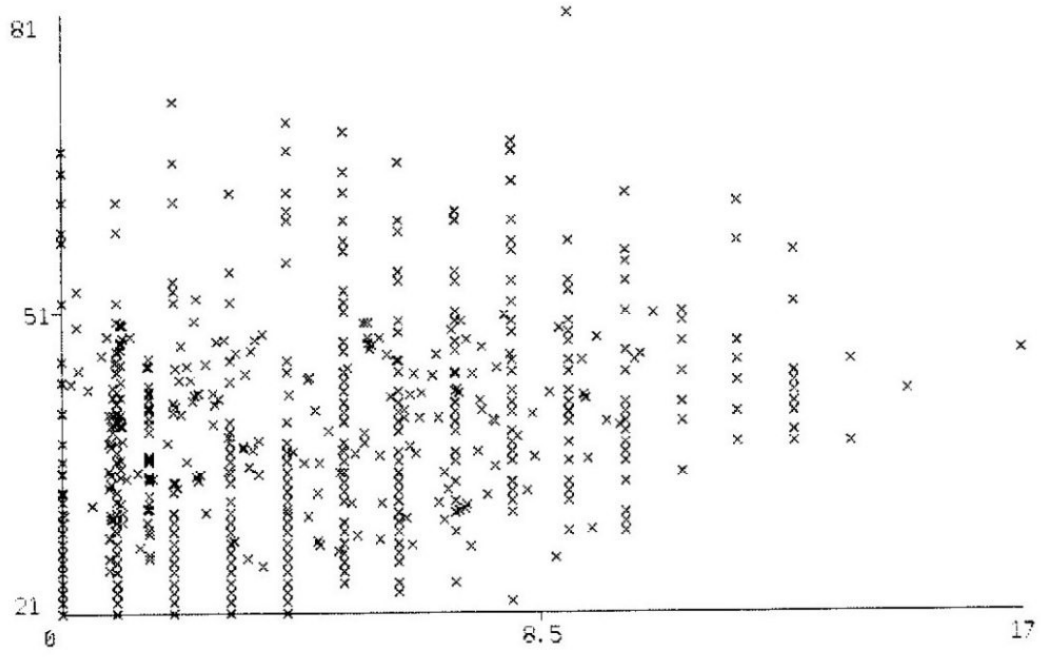

Figura 16 – Distribuição das variáveis *Preg Age* no espaço de características na base balanceada

atributo por probabilidade de ocorrência.

Problemas com mais de três variáveis não possuem representação gráfica adequada, devido a limitação em representar mais de três dimensões graficamente. Devido ao problema supracitado, apenas um par de atributos além dos pares que possuíam correlação maior

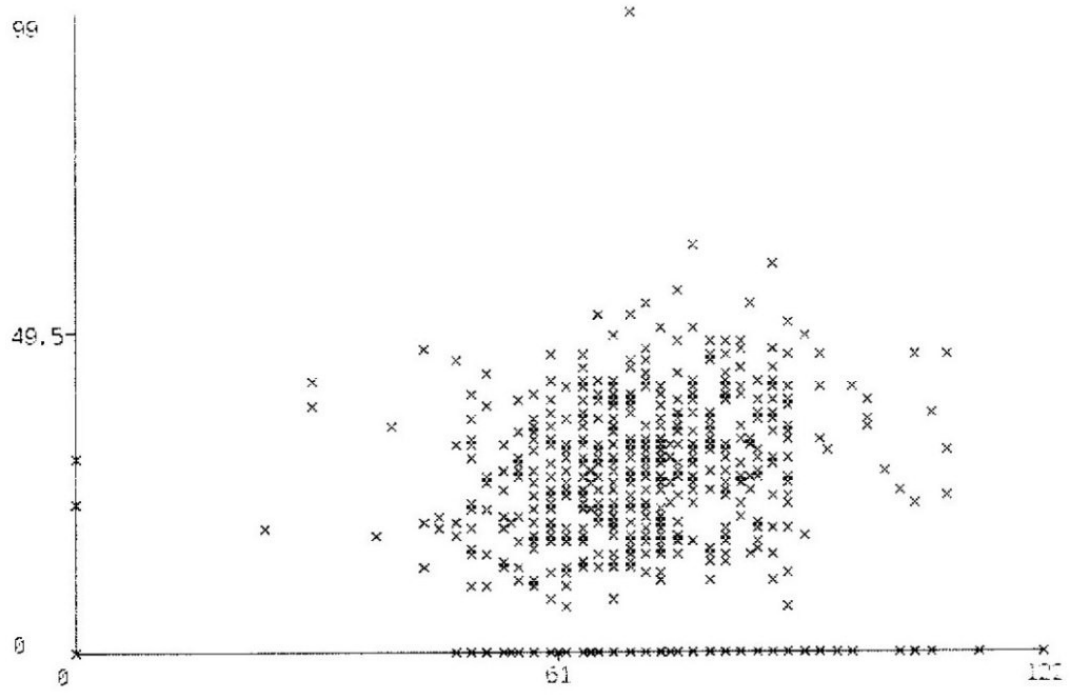

Figura 17 – Distribuição das variáveis *Pres* e *Skin* no espaço de características na base original

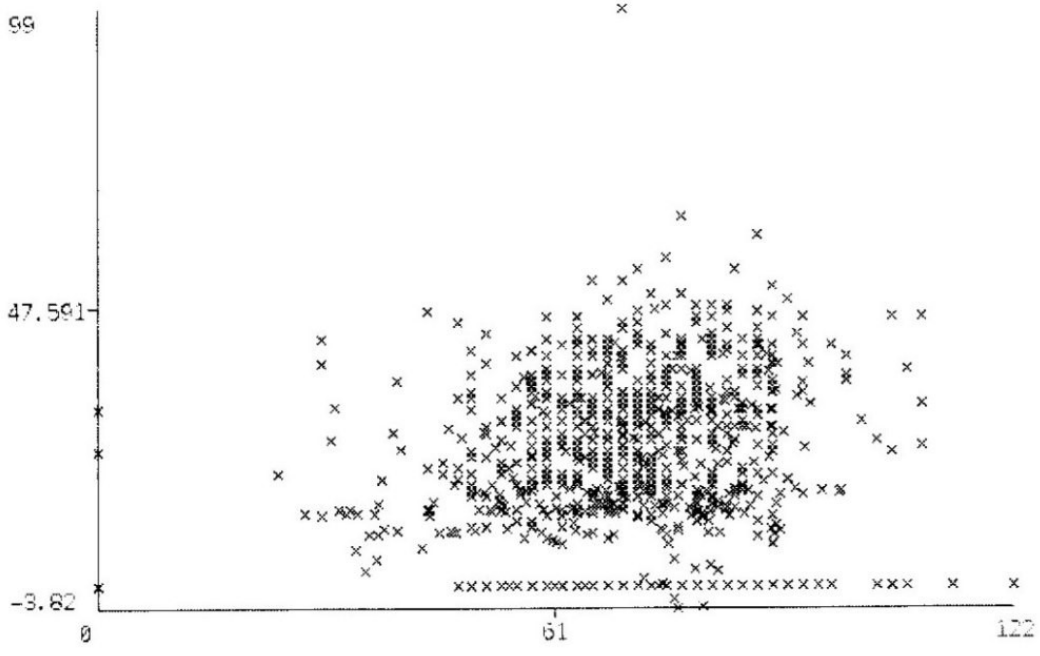

Figura 18 – Distribuição das variáveis *Pres Skin* no espaço de características na base balanceada

ou igual a 70% foi utilizado para análise.

Neste capítulo, o desenvolvimento do método RFAO é apresentado em duas etapas: protótipo e implementação final. Ambas as etapas incluíram um estudo de caso detalhado, mostrando como o método funciona em determinadas situações, a fim de validar as hipóteses previamente levantadas. O algoritmo desenvolvido na etapa de prototipação está totalmente

documentado, em forma de pseudo-código e diagrama de execução. A implementação final utilizou muito da lógica previamente desenvolvida, executando modificações para o ambiente de *stream* tal como certas melhorias, o que está documentado em forma de *diagramas*. Ao final, é possível ver evidências de que o RFAO pode ser utilizado para qualquer *stream* numérica que apresente qualquer tipo de comportamento nos atributos, não sendo necessário que exista correlação entre seus atributos. O próximo capítulo traz os experimentos e resultados preliminares, comparando o RFAO com algumas das técnicas estudadas no capítulo de revisão da literatura.

4 Resultados

Neste capítulo, são apresentados os resultados dos experimentos conduzidos com o RFAO. Inicialmente, são apresentados os resultados dos experimentos conduzidos com a primeira versão do método, desenvolvida para o ambiente *batch*. Uma das maiores vantagens do método ser implementado neste ambiente se dá pela quantidade de algoritmos de re-amostragem implementados no mesmo, o que enriquece os experimentos, fornecendo mais base de comparação, que é de fundamental importância para o aperfeiçoamento do método. Por fim, são apresentados os resultados obtidos no ambiente de *stream*, com a versão final do RFAO. Devido a exiguidade de técnicas de re-amostragem desenvolvidas para este ambiente, não foram encontradas implementações que viabilizassem a comparação com o RFAO, sendo assim, apenas o impacto do balanceamento na qualidade do modelo de predição gerado foi estudado.

A próxima seção detalha as características da primeira etapa dos experimentos, cobrindo o protocolo experimental, estratégia de validação, escolha de classificadores, técnicas de re-amostragem utilizadas para comparação e a descrição das bases utilizadas.

4.1 Versão *batch*

Nesta primeira etapa, os experimentos focaram em determinar se a aplicação das técnicas de amostragem existentes implica em uma melhora na performance de classificação na classe minoritária, se comparado a não utilização das mesmas. Para tal, o *baseline* de cada base de dados foi definido pelo classificador que melhor performou, entre os classificadores testados, sem a utilização de re-amostragem. Além da comparação com o *baseline*, uma comparação entre as técnicas de re-amostragem também foi realizada, com a finalidade de verificar pontos fortes que possam ser incorporados na implementação final do RFAO.

4.1.1 Protocolo experimental

Esta seção traz a discussão em torno das etapas seguidas para realização e avaliação dos testes. Os testes preliminares foram realizados no ambiente *batch*, devido a facilidade de implementação e testes, para ver como o RFAO se comporta tendo toda a base de dados disponível para análises, para então, expandir a solução para o ambiente de fluxos contínuos de dados. Como o RFAO não faz detecção de *drift* automaticamente, mas sim, dependeria de um algoritmo externo ou supor que o mesmo não acontece, os testes realizados em *batch* podem ser estendidos para o ambiente *stream*, ao processar *mini-batches*.

4.1.1.1 Validação

A estratégia de validação utilizada foi o *cross-validation* com 5 *folds*. A escolha de manter 5 ao invés dos 10 que são geralmente utilizados, se deve ao fato de que assim, haveriam mais instâncias minoritárias em cada *fold*. Todas as bases de dados foram divididas em cinco, mantendo a proporção de desbalanceamento inicial em cada *fold*. Após a divisão, os *folds* foram então combinados, 4 a 4, e em cada combinação foram aplicadas as técnicas de amostragem, para então, apresentar esta porção ao classificador. O *fold* restante era então utilizado para testar o modelo aprendido. O resultado final foi obtido pela média dos resultados obtidos.

4.1.1.2 Classificadores

É amplamente aceito na comunidade que nenhum classificador irá superar todos os outros sempre. Cada classificador funciona melhor em alguns cenários específicos. NB é um classificador que pertence à família probabilística. Utiliza a probabilidade a priori e a probabilidade a posteriori para determinar qual é a classe mais provável para uma nova instância. Este algoritmo assume que existe uma forte independência entre as características (ZHANG, 2004).

Classificadores baseados em árvores de decisão (DT) usam nós para representar os atributos. Os nós são escolhidos com base em algum critério, como impureza ou entropia, que é utilizado para dividir os dados por suas características (MAGEE, 1964). DT são fáceis de interpretar, uma vez que se tem acesso as regras de classificação, e por serem não paramétricos, trabalham bem com valores em faltantes. No entanto, eles tendem a sofrer do problema de sobre-ajuste facilmente (DIETTERICH, 1995).

O *Support Vector Machine* (SVM), por outro lado, é um classificador binário que dificilmente sofre do problema de sobre-ajuste devido a forma com que ele funciona, criando um hiperplano de margem máxima, que visa separar as duas classes (CORTES; VAPNIK, 1995). A função que gera o hiperplano não é linear em alguns casos, mas existem adaptações do *kernel* para quase todas as formas de distribuição de dados.

Para garantir a diversidade dos classificadores, DT, NB e SVM foram escolhidos para os experimentos. Todos esses classificadores são comuns. Não há nenhuma modificação em nível de algoritmo para melhor prever as classes minoritárias.

Foram utilizados os classificadores implementados em (PEDREGOSA et al., 2011) com algumas modificações. A DT usada neste trabalho usa entropia como critério de divisão, em vez de *gini-index*, que é o padrão. Além disso, a profundidade máxima que a DT pode crescer foi definida como até sete, ao invés do padrão, onde não há limite de crescimento. Esta modificação foi feita para evitar o sobre-ajuste, uma vez que nenhuma regra de poda foi definida. Para o SVM, o kernel linear foi escolhido, já que é o comportamento mais

Tabela 9 – Descrição das bases utilizadas

Nome	Instâncias	Atributos	Classe 0	Classe 1
Santander	76020	371	73012	3008
German	1000	20	700	300
D.C.C.Clients	30000	24	23364	6636
JapaneseC.S.	653	15	357	296
Australian	690	16	383	307
Pima Indians Diabetes	768	8	500	268

comum para um conjunto de dados. Para os algoritmos de amostragem, o parâmetro responsável pelo novo tamanho da amostra foi definido para 90% do tamanho de S_{maj}, a fim de adquirir um conjunto de dados quase perfeitamente balanceado.

4.1.1.3 Técnicas de re-amostragem utilizadas

O SMOTE e suas variações compõe o espectro de técnicas de *oversampling* mais utilizadas para tratamento amostral. Para os experimentos, o SMOTE, SMOTE-Borderline-1, SMOTE-Borderline-2, SMOTE-SVM e SMOTE+TK foram selecionados para servir como base de comparação ao RFAO. Todos estes algoritmos foram executados com os parâmetros *default* da implementação em (LEMAITRE; NOGUEIRA; ARIDAS, 2016). Para ser viável a utilização do RFAO, o mesmo deve atingir resultados competitivos se comparado a estas técnicas. Para os testes, o parâmetro de correlação mínima esperada foi definido como 50%.

4.1.2 Bases de dados

Quatro bases com desbalanceamento intrínseco foram utilizadas. A base Santander, oriunda do Kaggle challenge [1] é a maior entre as escolhidas tal como é a que apresenta o maior grau de desbalanceamento. As outras bases foram obtidas na UCI (LICHMAN, 2013). Todos os atributos são inteiros, booleanos ou reais. Alguns atributos eram originalmente categóricos nos conjuntos de dados originais, mas foram convertidos em booleanos utilizando *onehotencoding*.

A próxima seção irá trazer os resultados preliminares tal como a discussão.

4.2 Resultados versão *batch*

Nesta seção são apresentados os primeiros resultados do RFAO em ambiente *batch*, a fim de comparar sua performance com os os algoritmos concorrentes. Cada sub-seção traz as análises referentes a base em questão.

[1] <https://www.kaggle.com/c/santander-customer-satisfaction>

4.2.1 Santander

Esta base de dados possui o maior grau de desbalanceamento entre as classes (1:24). A *baseline* estipulada para esta base era de 0.641 para sensibilidade e 0.6836 para AUROC, alcançado pelo NB, como mostrado na tabela 10.

Após a aplicação das técnicas de amostragem, o RFAO obteve a melhor performance referente a sensibilidade com os classificadores NB (0.662) e DT (0.4153), porém, o melhor valor global foi obtido pela combinação do SVM com o SMOTE-Borderline-1 (0.9). O fato do SVM ter acertado apenas 0.0007 das instâncias de S_{min} sem a aplicação das técnicas de amostragem, sugere que a separação entre as classes não seja linear, o que impossibilitou que separação fosse adequada pelo kernel escolhido.

Já para AUROC, que fornece um extrato global da performance, não ouve melhora em relação ao *baseline* estipulado, o que sugere a aplicação das técnicas de *oversampling* podem influenciar negativamente na correta classificação de instâncias de S_{maj}.

O desempenho dos classificadores também foi severamente prejudicado pela 'maldição da dimensionalidade', uma vez que este conjunto de dados é descrito por 371 atributos. Destes atributos, a maioria deles é binário, não existem atributos com comportamento normal, tal como não existem atributos que possuam correlação maior do que a estipulada. A combinação destes fatores não favorece a utilização do RFAO.

4.2.2 Japanese Credit Screening (Japanese C. S.)

Esta base de dados tem o menor número de instâncias e atributos, entre as testadas. O *baseline* foi estipulado pelo classificador DT, obtendo 0.7695 para sensibilidade e 0.8188 para AUROC, como é obsevado na Tabela 11. O melhor resultado obtido para sensibilidade (0.8647) e AUROC (0.8496), foi obtido ainda com o DT, porém, em combinação com o SMOTE+TK, oferecendo indícios que existe problemas de sobreposição entre as classes.

O RFAO não obteve boa performance nesta base, possivelmente devido ao problema de sobreposição as instâncias sintéticas tenham sido geradas em uma região não favorável. Sua melhor performance foi obtida com o DT (0.7433), porém foi um resultado pior do que seus concorrentes. Existe um par de atributos que apresenta correlação de 99.99%, porém, existem apenas três posições para os pares-ordenados destes atributos se fixarem, o que não ajuda o RFAO a criar novas instâncias representativas.

4.2.3 German

Esta base de dados apresenta um grau de desbalanceamento maior do que a *Japanese Credit Screening* (1:3,5) e também mais atributos (20). Sua *baseline* para sensibilidade foi estipulada pelo SVM (0.6067) e para AUROC pelo DT (0.6617). A melhor performance

Tabela 10 – Resultados para base Santander

Classificador	Estratégia	Sensibilidade	AUROC
NB	-	0.641	0.6836
NB	$SMOTE$	0.6187	0.6249
NB	$SMTK$	0.6366	0.6366
NB	$SMBL1$	0.6503	0.6486
NB	$SMBL2$	0.2234	0.5028
NB	$SMOTE-SVM$	0.191	0.5238
NB	$RFAO$	0.662	0.5273
SVM	-	0.0007	0.4994
SVM	$SMOTE$	0.6627	0.5708
SVM	$SMTK$	0.5372	0.5795
SVM	$SMBL1$	0.9002	0.5562
SVM	$SMBL2$	0.4364	0.5865
SVM	$SMOTE-SVM$	0.8676	0.6413
SVM	$RFAO$	0.672	0.5711
DT	–	0.0037	0.5014
DT	$SMOTE$	0.4046	0.6594
DT	$SMTK$	0.4092	0.6612
DT	$SMBL1$	0.4089	0.6616
DT	$SMBL2$	0.357	0.6437
DT	$SMOTE-SVM$	0.4235	0.6111
DT	$RFAO$	0.4153	0.6728

para sensibilidade foi obtida pelo SVM, em combinação com o SMOTE-SVM, seguido pelo SMOTE e RFAO, obtendo 0.8, 0.7667 e 0.7133, respectivamente. Já para o AUROC, o SVM + SMOTEBorderline-1 obteve o melhor resultado (0.6779) seguido pelo RFAO (0.663). Nesta base foi possível observar o SMOTE-SVM ser realmente efetivo em combinação com o classificador SVM. Os resultados completos são encontrados na Tabela 12.

A performance do RFAO foi razoável nesta base de dados, mesmo não existindo correlação entre os atributos, o que fez com que todos fossem gerados de maneira probabilística.

4.2.4 Default of Credit Card Clients (Default C. C. Clients)

Esta base de dados apresenta desbalanceamento leve (1:3,32) tal como não possui muitos atributos (24), porém é a segunda em número de instâncias (30.000).

A *baseline* foi estipulada pelo NB, obtendo 0.497 para sensibilidade e 0.6735 para AUROC. A melhor combinação para sensibilidade encontrada foi com o SVM + SMOTEBorderline-1 (0.8164) seguido por SVM + SMOTEBorderline-2 (0.7686), e para AUROC, NB + SMOTE-SVM (0.6803), como é observado na Tabela 13.

Não existe correlação entre os atributos, logo o RFAO teve que gerar todas as

Tabela 11 - Resultados para base Japanese Credit Screening

Classificador	Estratégia	Sensibilidade	AUROC
NB	–	0.681	0.7269
NB	$SMOTE$	0.6842	0.7244
NB	$SMTK$	0.7008	0.7131
NB	$SMBL1$	0.681	0.7269
NB	$SMBL2$	0.681	0.734
NB	$SMOTE-SVM$	0.6904	0.722
NB	$RFAO$	0.6861	0.7192
SVM	–	0.6654	0.7514
SVM	$SMOTE$	0.6766	0.6953
SVM	$SMTK$	0.3718	0.6662
SVM	$SMBL1$	0.5011	0.6732
SVM	$SMBL2$	0.6373	0.7343
SVM	$SMOTE-SVM$	0.6514	0.65
SVM	$RFAO$	0.6442	0.663
DT	–	0.7694	0.8188
DT	$SMOTE$	0.7995	0.8227
DT	$SMTK$	0.8647	0.8496
DT	$SMBL1$	0.8405	0.839
DT	$SMBL2$	0.8305	0.8396
DT	$SMOTE-SVM$	0.8105	0.8324
DT	$RFAO$	0.7433	0.7811

instâncias de maneira probabilística. Para esta base, o RFAL conseguiu aumentar o ***baseline*** quando combinado com o SVM, obtendo 0.5873.

4.2.5 Australian

Esta base de dados apresenta o menor grau de desbalanceamento apresentado (1:1,24) e apenas 16 atributos. O *baseline* estipulado para sensibilidade foi obtido com o classificador SVM (0.7741), já para AUROC foi obtido com o DT (0.817), como pode ser observado na Tabela 14. A melhor combinação para sensibilidade se deu com o DT + SMTK (0.8179), deixando o RFAO (0.795) em quarto lugar. Já para AUROC, DT + SMOTEBorderline-1 foi o melhor (0.8462), deixando o RFAO em quinto lugar (0.8167). DT + SMOTEBorderline-1 mostrou-se a melhor combinação para esta base, porém, a maioria dos algoritmos obteve performance similar. Novamente o RFAO teve que gerar as instâncias de modo probabilístico, pois não haviam atributos correlacionados.

4.2.6 Pima Indians Diabetes

Esta base de dados apresenta o menor número de atributos (8), tal como a menor quantidade de instâncias em S_{min}. Apenas dois atributos possuem correlação, A1 e A8, com 52.76% de correlação. O *baseline* foi estabelecido com DT, para sensibilidade (0.5816)

Tabela 12 – Resultados para a base German

Classificador	Estratégia	Sensibilidade	AUROC
NB	–	0.15	0.5436
NB	$SMOTE$	0.4	0.565
NB	$SMTK$	0.4067	0.5583
NB	$SMBL1$	0.3967	0.5633
NB	$SMBL2$	0.3933	0.5631
NB	$SMOTE-SVM$	0.44	0.565
NB	$RFAO$	0.33	0.5264
SVM	–	0.6067	0.6405
SVM	$SMOTE$	0.7667	0.6512
SVM	$SMTK$	0.5267	0.6476
SVM	$SMBL1$	0.65	0.6779
SVM	$SMBL2$	0.66	0.6021
SVM	$SMOTE-SVM$	0.8	0.6557
SVM	$RFAO$	0.7133	0.663
DT	–	0.5033	0.6617
DT	$SMOTE$	0.5167	0.6262
DT	$SMTK$	0.5267	0.634
DT	$SMBL1$	0.5733	0.6424
DT	$SMBL2$	0.62	0.6521
DT	$SMOTE-SVM$	0.5	0.6271
DT	$RFAO$	0.38	0.6107

e AUROC (0.6922). O melhor resultado para sensibilidade foi obtido com o SVM + RFAO (0.7712), seguido pelo SVM + SMOTEBorderline-2 (0.7538). Para AUROC o melhor resultado foi obtido com DT + RFAO (0.7168), seguido por DT + SMOTEBorderline-1 (0.7093). Os resultados na integra podem ser observados na Tabela 15

A utilização de técnicas de re-amostragem, teve impacto negativo na classificação de instâncias de S_{maj} em alguns casos, como NB + SMOTEBorderline-2, DT + SMOTE e DT + SMOTE-SVM, que tiveram a performance em ambas as classes, representada pelo AUROC, reduzida segundo a *baseline*.

Ao final dos experimentos preliminares, conclui-se que apesar das bases de dados testadas apresentarem correlações baixas ou inexistentes entre os atributos, o que acabou forçando o RFAO a gerar a maioria dos valores de maneira probabilística, logo, não obtendo seu potencial total, em todas as bases testadas o RFAO obteve melhora na *baseline*. Observou-se que para alguns classificadores, em algumas bases de dados, o RFAO tem performance muito similar aos seus concorrentes. Para a base Pima Indians Diabetes, alcançou performance superior em ambos os quesitos analisados, porém, em certos casos, como na base Default of Credit Card Clients, a performance fica aquém do desejado, pois apesar de ter aumentando a *baseline* em quase 10%, ainda fica muito atrás do primeiro colocado.

Tabela 13 – Resultados para base Default of C. C. Clients

Classificador	Estratégia	Sensibilidade	AUROC
NB	–	0.497	0.6735
NB	$SMOTE$	0.4494	0.6678
NB	$SMTK$	0.4498	0.6679
NB	$SMBL1$	0.4494	0.6678
NB	$SMBL2$	0.4656	0.6746
NB	$SMOTE-SVM$	0.5081	0.6803
NB	$RFAO$	0.493	0.665
SVM	–	0.2472	0.4995
SVM	$SMOTE$	0.6005	0.5527
SVM	$SMTK$	0.6258	0.5226
SVM	$SMBL1$	0.8164	0.5258
SVM	$SMBL2$	0.7686	0.5078
SVM	$SMOTE-SVM$	0.3173	0.5651
SVM	$RFAO$	0.5873	0.5423
DT	–	0.3624	0.6553
DT	$SMOTE$	0.4325	0.6762
DT	$SMTK$	0.4391	0.6749
DT	$SMBL1$	0.4209	0.6692
DT	$SMBL2$	0.3823	0.6608
DT	$SMOTE-SVM$	0.4122	0.6688
DT	$RFAO$	0.4277	0.6708

4.3 Versão *streams*

Nesta seção são detalhados os experimentos para o ambiente de *streams*. Tal como na seção anterior, o *baseline* de cada base de dados foi estabelecida pelo performance do classificador base, sem a utilização de re-amostragem. A comparação da implementação para *batch* com as técnicas de re-amostragem concorrentes foi de fundamental importância para determinar os pontos fracos e limitações do método, para que pudessem ser corrigidas ou contornadas na implementação final.

4.3.1 Protocolo experimental

A fim de determinar qual é a melhor combinação de valores de parâmetros para cada *stream* analisada, diversas combinações foram testadas, conforme tabela 16. Apesar do processo assemelhar-se ao *GridSearch*, comum no ambiente *batch* para determinar os parâmetros ótimos para determinada base de dados, no ambiente *stream* este processo serve apenas para definir os parâmetros, que **em média**, são melhores, uma vez que é impossível determinar uma combinação de valores de atributos que não seja influenciada por possíveis mudanças que ocorram na distribuição ou padrões de comportamento da *stream* observada.

Tabela 14 - Resultados para base Australian

Classificador	Estratégia	Sensibilidade	AUROC
NB	–	0.7125	0.8119
NB	*SMOTE*	0.7125	0.8119
NB	*SMTK*	0.7191	0.7982
NB	*SMBL1*	0.7125	0.804
NB	*SMBL2*	0.7092	0.8052
NB	*SMOTE – SVM*	0.7012	0.7988
NB	*RFAO*	0.7028	0.8133
SVM	–	0.7741	0.6949
SVM	*SMOTE*	0.5648	0.6384
SVM	*SMTK*	0.6258	0.5226
SVM	*SMBL1*	0.7282	0.715
SVM	*SMBL2*	0.6539	0.7457
SVM	*SMOTE – SVM*	0.67	0.5651
SVM	*RFAO*	0.5181	0.6578
DT	–	0.7723	0.817
DT	*SMOTE*	0.7754	0.8214
DT	*SMTK*	0.8179	0.8319
DT	*SMBL1*	0.8173	0.8462
DT	*SMBL2*	0.8017	0.8291
DT	*SMOTE – SVM*	0.7922	0.8187
DT	*RFAO*	0.795	0.8167

O classificador base utilizado nos experimentos foi o Hoeffding Tree (HT) (DOMINGOS; HULTEN, 2000), que é um classificador incremental baseado na estrutura de árvore, capaz de realizar predições a qualquer momento e trabalhar com *streams* massivas, assumindo que a distribuição não mude com o passar do tempo.

Como citado no início do capítulo, não existem implementações de algoritmos de re-amostragem disponíveis publicamente para o ambiente de *streams*, logo, os experimentos se basearam em efetuar a comparação da performance do o *baseline*, estabelecido pela performance do HT, com a performance do mesmo classificador na presença do RFAO. O procedimento de avaliação *test-then-train* foi adotado, uma vez que utiliza todas as instâncias recebidas para teste e posteriormente, treino, pode-se tirar o maior proveito dos dados recebidos. As mesmas métricas do utilizadas para testes no ambiente *batch*, sensibilidade e AUROC, foram utilizadas. Por fim, o teste de Wilcoxon (WILCOXON, 1945) foi utilizado para determinar se a utilização do RFAO implica em uma melhora estatística significativa, se comparado ao *baseline*, para cada *stream* analisada.

4.3.2 Streams

O *framework Massive On-line Analysis* (MOA) (BIFET et al., 2010) foi utilizado configuração e execução destes experimentos. Este *framework* permite a implementação,

Tabela 15 – Resultados para base Pima Indians Diabetes

Classificador	Estratégia	Sensibilidade	AUROC
NB	–	0.03	0.5067
NB	*SMOTE*	0.4469	0.5273
NB	*SMTK*	0.4243	0.5254
NB	*SMBL1*	0.5190	0.5240
NB	*SMBL2*	0.6540	0.4724
NB	*SMOTE – SVM*	0.4631	0.5102
NB	*RFAO*	0.6106	0.5433
SVM	–	0.2182	0.5263
SVM	*SMOTE*	0.6605	0.5878
SVM	*SMTK*	0.4821	0.6015
SVM	*SMBL1*	0.7413	0.5916
SVM	*SMBL2*	0.7538	0.6124
SVM	*SMOTE – SVM*	0.7421	0.5813
SVM	*RFAO*	0.7712	0.6432
DT	–	0.5816	0.6922
DT	*SMOTE*	0.60	0.6756
DT	*SMTK*	0.6630	0.6946
DT	*SMBL1*	0.69	0.7093
DT	*SMBL2*	0.6789	0.7025
DT	*SMOTE – SVM*	0.6391	0.6653
DT	*RFAO*	0.7118	0.7168

Tabela 16 – Parâmetros e valores do algoritmo RFAO para *streams* testados

Parâmetro	Valores
baseLearner	HoeffdingTree
window	[70, 500, 1000, 5000, 10000]
keepMinorityBatch	[True, False]
ratio	[0.8, 0.9, 1.0]
expectedCorrelation	[0.6, 0.7, 0.8, 0.9, 1.0]
expandCorrelatedAttributes	[0.1, 0.2, 0.3]
applyBalance	[True, False]
ensureNeighborhood	[True, False]
knn	[3, 5, 7]
minNeighbors	[2, 3, 4, 5, 6, 7]
randomUndersampling	[True, False]
percentOfMajorityInstancesToKeep	[0.6, 0.7, 0.8, 0.9, 1.0]

testes e avaliação de algoritmos de aprendizado para o ambiente o ambiente *streams*.

As bases de dados *Pima Indians Diabetes*, *Japanese C. S.*, *German* e *Santander* foram utilizadas para os testes no ambiente *stream*, uma vez que podem ser simuladas como uma fluxo contínuo de dados. Além destas, foram incluídas outras duas *streams*, que utilizam algoritmos geradores para sintetizar os dados. O fato destas duas *streams* serem geradas via algoritmo, fornece mecanismos para que várias configurações diferentes dos

dados sejam testados, por exemplo, diferentes níveis de balanceamento ou ruído.

4.3.2.1 Agrawal Generator

O gerador Agrawal generator (AGRAWAL; IMIELINSKI; SWAMI, 1993) sintetiza instâncias que possuem nove atributos, seis com valores nominais e três contínuos, além da classe binária. Existe a possibilidade de utilizar dez funções diferentes para geração das instâncias, além de poder adicionar um fator de perturbação, que adiciona ruído aos valores de atributos, definido através de uma distribuição randômica uniforme. Apesar de existir a possibilidade de gerar dados desbalanceados, não existem parâmetros para definir qual o nível desejado. Para os experimentos, quatro configurações distintas foram utilizadas: AGRW-90, AGRW-60, AGRW-30 e AGRW-0. Na primeira configuração um fator de perturbação foi definido para 90%, nas seguintes para 60, 30 e 0, respectivamente, com os outros parâmetros nas configurações *default*.

4.3.2.2 SEA Generator

O SEA generator (STREET; KIM, 2001) é um gerador de *streams* que gera instâncias contendo três atributos contínuos, que assumem valores $\in [0, 10]$, além da classe binária. Dos três atributos, apenas dois são relevantes para classificação. Além da possibilidade de adicionar ruído na distribuição de classes, a implementação utilizada fornece a possibilidade de informar explicitamente o balanceamento desejado. Para os experimentos, quatro configurações distintas desta *stream* foram utilizadas: SEA-40, SEA-30, SEA-20 e SEA-10. Na primeira delas, a proporção que a classe minoritária representa em relação a majoritária é de 40%, na segunda 30%, na terceira 20%, e por fim, na última a classe minoritária representa apenas 10%.

4.3.3 Resultados obtidos

Os resultados obtidos são mostrados de acordo com a *stream* analisada. Na tabela 17 os resultados obtidos com as variações do gerador Agrawal são apresentados, sendo os melhores resultados representados em negrito. A evolução da sensibilidade ao longo das *streams* AGRW-90, AGRW-60, AGRW-30 e AGRW-0 pode ser observada nas figuras 19, 20, 21 e 22, respectivamente. Na tabela 18 os resultados obtidos com o gerador SEA são apresentados, seguindo o mesmo modelo de apresentação supracitado. Como praticamente não existiu variação de performance entre o *baseline* e a versão com o RFAO, os gráficos de evolução de sensibilidade não foram incluídos. Por fim, os resultados obtidos nas *streams* reais Santander, Pima Indians Diabetes, German e Japanese C. S. são apresentados na tabela 19, tal como os gráficos de evolução de sensibilidade são apresentados nas figuras 23, 24, 26 e 25, respectivamente.

Tabela 17 – Resultados RFAO com gerador Agrawal

Stream	Classificador	Sensibilidade	AUROC
AGRW-90	HT	0	0.49
AGRW-90	HT + RFAO	**0.05**	**0.5**
AGRW-60	HT	0	0.51
AGRW-60	HT + RFAO	**0.18**	**0.58**
AGRW-30	HT	0.01	**0.73**
AGRW-30	HT + RFAO	**0.58**	**0.73**
AGRW-0	HT	**0.99**	0.98
AGRW-0	HT + RFAO	**0.99**	**0.99**

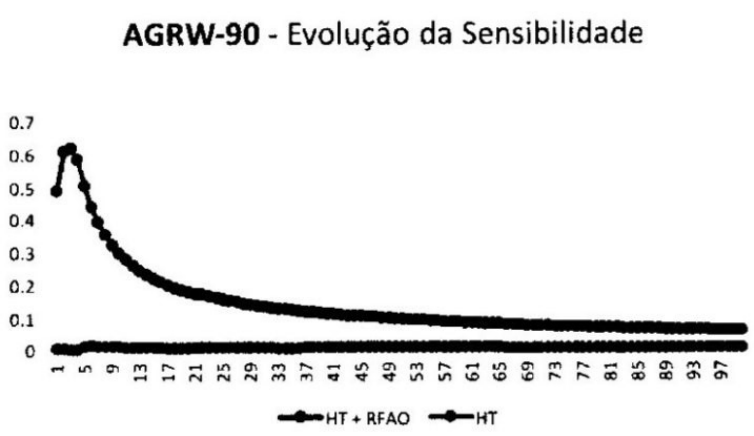

Figura 19 – Evolução da sensibilidade em AGRW-90 a cada 10.000 instâncias

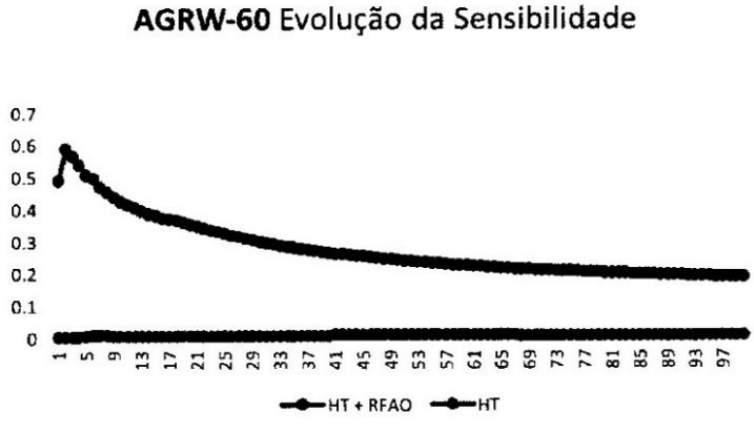

Figura 20 Evolução da sensibilidade em AGRW-60 a cada 10.000 instâncias

Como diversas variações de parâmetros foram testadas, esperava-se que existisse maior diversidade entre os mesmos para cada *streams* analisada, porém, com exceção da *stream* Santander, que obteve parametrização diferente, sendo a única que utilizou o RUS (0.7), todas as outras testadas obtiveram melhor AUROC com os parâmetros *default*, variando apenas o tamanho da janela ϖ, que para as *streams* Pima Indians Diabetes, Japanece C. S e German obtiveram o tamanho de 70 instâncias, enquanto as outras *streams*

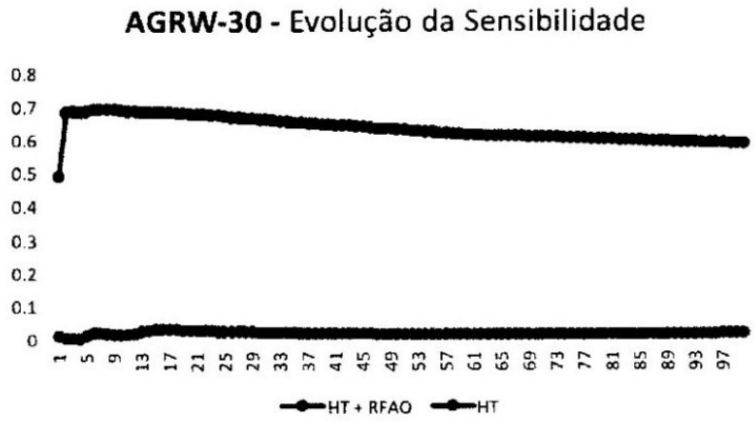

Figura 21 – Evolução da sensibilidade em AGRW-30 a cada 10.000 instâncias

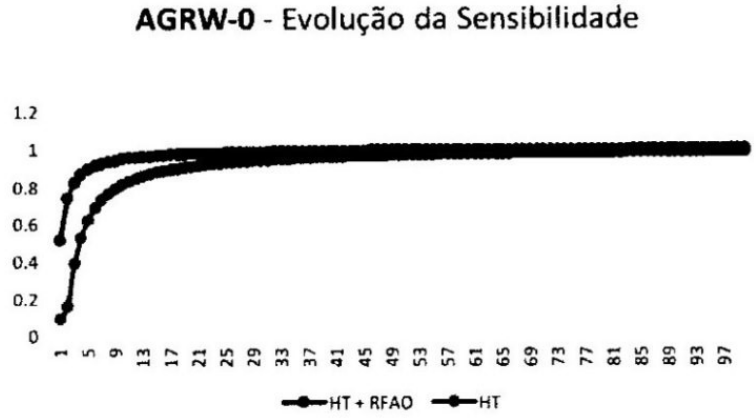

Figura 22 – Evolução da sensibilidade em AGRW-0 a cada 10.000 instâncias

Tabela 18 – Resultados RFAO com gerador SEA

Stream	Classificador	Sensibilidade	AUROC
SEA-40	HT	0.98	0.99
SEA-40	HT + RFAO	0.98	0.99
SEA-30	HT	0.98	0.99
SEA-30	HT + RFAO	**0.99**	0.98
SEA-20	HT	0.99	0.98
SEA-20	HT + RFAO	0.99	0.98
SEA-10	HT	0.97	0.99
SEA-10	HT + RFAO	0.97	0.99

maiores, utilizaram $\varpi = 5000$. Para todas as *streams* testadas ocorreu diferença estatística significativa, referente a sensibilidade, entre os resultados obtidos pelo RFAO em relação ao *baseline*, comprovando a eficiência do método.

Tabela 19 – Resultados RFAO *stream* Santander e Pima Indians Diabetes

Stream	Classificador	Sensibilidade	AUROC
Santander	HT	0	**0.63**
Santander	HT + RFAO	**0.56**	0.52
Japanese C. S.	HT	0.48	0.59
Japanese C. S.	HT + RFAO	**0.67**	**0.64**
German	HT	0.05	0.56
German	HT + RFAO	**0.51**	**0.71**
Pima Indians Diabetes	HT	0.42	0.72
Pima Indians Diabetes	HT + RFAO	**0.6**	**0.73**

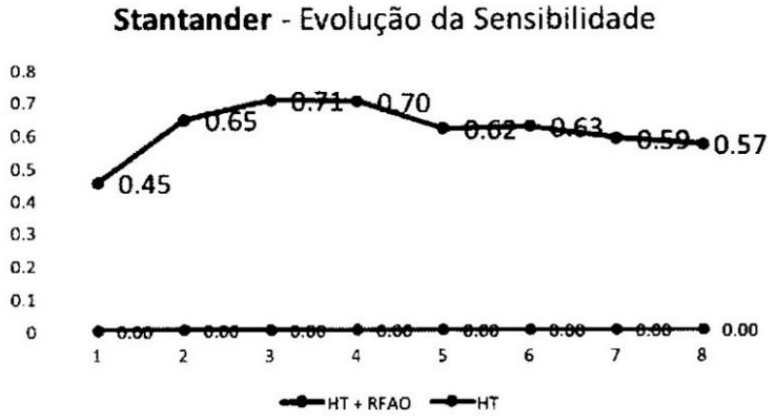

Figura 23 – Evolução da sensibilidade em Santander a cada 10.000 instâncias

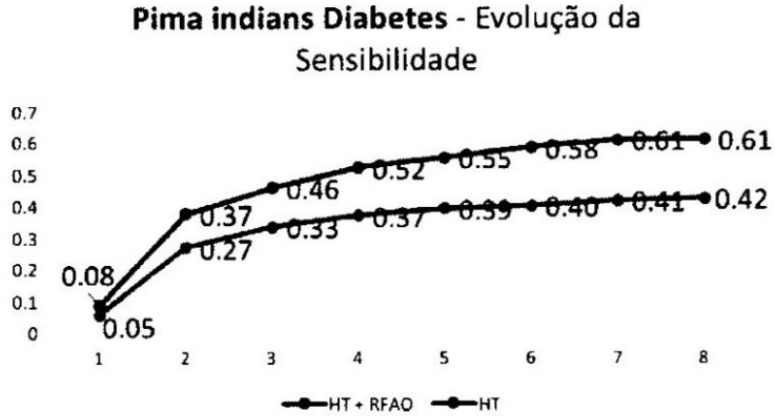

Figura 24 – Evolução da sensibilidade em Pima Indians Diabetes a cada 70 instâncias

4.3.4 Análise dos resultados e discussão

Os resultados obtidos com a versão para *stream* do RFAO mostram empiricamente que o método cumpre o objetivo principal, que se baseia em aumentar a performance de classificação na classe minoritária. Em praticamente todas as situações nota-se também, ganhos referentes a performance geral, observados pelo aumento do AUROC.

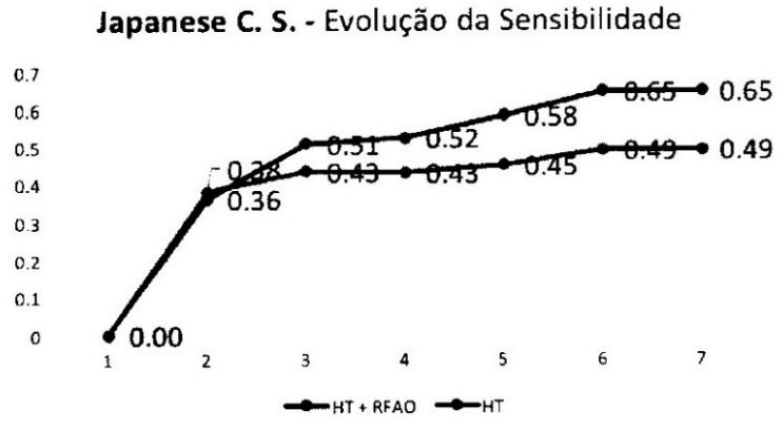

Figura 25 – Evolução da sensibilidade em Japanese C. S. a cada 70 instâncias

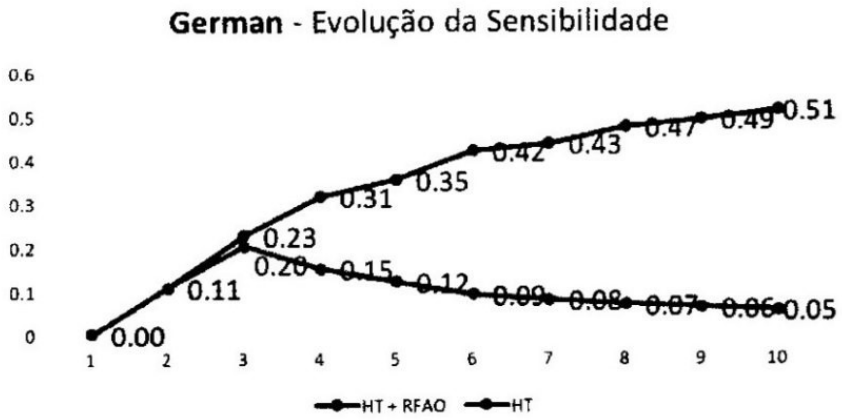

Figura 26 – Evolução da sensibilidade em German a cada 70 instâncias

Nas *streams* reais, como *German*, o resultado referente ao AUROC (0.71) foi melhor do que o melhor resultado obtido no ambiente *batch* (0.67). O mesmo foi observado na *stream* Pima Indians Diabetes, que obteve para AUROC 0.73, contra 0.71 no ambiente *batch*, mostrando que as melhorias realizadas no método, após os resultados preliminares, foram eficazes. Ainda sobre a *stream German*, a partir da figura 26 é possível observar que após a terceira janela de dados ocorreu uma queda na sensibilidade referente ao *baseline*, em decorrência da diminuição brusca na incidência de instâncias minoritárias. Apesar disso, quando utilizado o RFAO, a sensibilidade continuou aumentando durante toda a *stream*. Nenhum atributo possuiu correlação, não sendo possível gerar valores via RLS. Tal feito levanta evidências sobre a qualidade das instâncias geradas pelo RFAO, que além de melhorar a performance global, aumenta a performance na classe minoritária, mesmo quando os valores de atributos são gerados via probabilidade e estatística. Para a Pima Indians Diabetes também é possível verificar a melhora na sensibilidade a cada nova janela, como mostra a figura 24. A partir da segunda janela, a diferença entre a as sensibilidades da versão balanceada para o *baseline* é de 10%, chegando a 19% na última.

Para esta *stream*, dois pares de atributos possuem correlação em determinados momentos, possibilitando expandir a classe minoritária no espaço de caraterísticas.

Ressalta-se que no ambiente de *streams*, a baixa observação de instâncias minoritárias pode influenciar ainda mais o modelo induzido pelo classificador do que no ambiente *batch*, como fica evidente nos experimentos com as *streams German*, previamente discutida, e Santander. Na figura 23 nota-se que a abordagem sem balanceamento, representada na cor laranja, não conseguiu classificar corretamente instâncias minoritárias, dada a baixa relevância que possuem para o indutor, devido a distribuição. Ainda nesta *stream*, o resultado obtido referente a sensibilidade (0.56) foi superior ao melhor resultado obtido pelo classificador baseado em árvore (DT + SMOTE-SVM) utilizado nos testes preliminares (0.42), tendo alcançado 0.71 na terceira janela. Destaca-se que existem evidências empíricas de que a baixa incidência de instâncias minoritárias, além de contribuir para baixa performance do classificador, contribui diretamente para a deterioração das instâncias geradas sinteticamente, uma vez que entre a segunda e quarta janela, o número de instâncias minoritárias observado foi o maior durante toda a *stream*, o que implicou diretamente nas maiores sensibilidades observadas. Por fim, em Japanese C. S., para a menor *stream* em número de atributos e desbalanceamento, a utilização do RFAO implicou em uma melhora de até 16% na sensibilidade. Devido às características desta base, é possível que existam problemas de sobreposição entre as classes, como discutido nos experimentos em batch, pois existe um par de atributos correlacionados tanto em S_{min} quanto em S_{maj}. Tal comportamento nos dados dificulta o reconhecimento de padrões por parte do classificador e do algoritmo de re-amostragem.

Com relação às *streams* oriundas de geradores, ocorreram duas situações distintas. Para a SEA e todas as suas variações testadas, SEA-40, SEA-30, SEA-20 e SEA-10, não existiu diferença estatística significativa na performance do *baseline* para a versão balanceada, mesmo na maior proporção de desbalanceamento testada. Isto se deve ao fato das instâncias serem geradas com apenas três atributos, sendo que destes, apenas dois influenciam a classe, o que reduz a complexidade do problema, pois diminui as chances de sobreposição de classe, se comparado a um problema com mais dimensões. Já para as *streams* geradas através do Agrawal, que mantém a proporção de balanceamento em média de 1:2, a utilização do RFAO foi mais eficiente. Apesar do pouco desbalanceamento observado, o fato de perturbação induzido nas diferentes configurações do Agrawal testadas, gera problemas para o classificador, à medida que o mesmo aumenta. Com o fator de perturbação zero, a performance do *baseline* e da versão balanceada possui comportamento assintótico referente a sensibilidade, como pode ser visto na figura 22. Porém, à medida que o fator aumenta, como nas *streams* AGRW-60 e AGRW-90, a sensibilidade é comprometida para ambos os casos. Apesar da influência exercida pela sobreposição na geração de instâncias sintéticas, com o fator de perturbação em 30% nota-se que a versão balanceada consegue manter a sensibilidade até 64% melhor do que o *baseline*, como pode ser visto

na figura 21. Para todas as variações da Agrawal testadas, o RFAO foi estatisticamente melhor do que o *baseline*.

4.4 Considerações finais

Neste capítulo foram apresentados os experimentos realizados com o intuito de comparar a utilização do RFAO como mecanismo de re-amostragem. Primeiramente foram realizados os experimentos no ambiente *batch*, com a finalidade de comparar o protótipo do método com diversas técnicas de re-amostragem estudadas na etapa de revisão de literatura. Tal comparação foi possível devido à quantidade de técnicas implementadas para este ambiente. Estes experimentos serviram como base para sinalizar falhas e limitações da versão inicial, o que levou a melhorias comprovadas no método final, para *streams*.

Os resultados permitem afirmar que o algoritmo proposto é melhor do que os demais concorrentes, uma vez que resolve alguns problemas recorrentes das abordagens mais utilizadas, como geração incorreta de valores para determinados atributos e aumento da representatividade de S_{min} no espaço de características. Além disto, por utilizar uma abordagem baseada em atributos, consome menos tempo de processamento do que abordagens tradicionais baseadas em instância, o que é fundamental para o ambiente de fluxos contínuos de dados.

O próximo capítulo apresenta as conclusões do presente trabalho, tal como trabalhos futuros.

5 Conclusão

Algoritmos de re-amostragem são utilizados com a finalidade de prover uma distribuição tão equilibrada quanto se desejar, baseado em uma amostra contendo dados desbalanceados. Um dos principais problemas apontados na utilização destas técnicas, está na possível indução de problemas de sobreposição entre as classes, valores para atributos gerados em faixas inaceitáveis, perda de informação e alta complexidade computacional.

As técnicas de re-amostragem existentes, possuem abordagem baseadas em: *oversampling*, *undersampling* ou abordagens mistas, que tem como objetivo aumentar o número de instâncias de S_{min}, diminuir o número de instâncias de S_{maj}, ou utilizar as duas abordagens simultaneamente, respectivamente. A maioria das técnicas utiliza estratégias baseada em distância, que induzem a um alto custo computacional. Além das técnicas baseadas em distância, existem técnicas baseadas em *cluster* e evolução.

Um dos maiores problemas encontrados durante o desenvolvimento do método se deu na escassez de estratégias de re-amostragem desenvolvidas para o ambiente de *streams*, o que torna difícil obter resultados comparativos para o método proposto, tal como um ponto de partida para o seu desenvolvimento. Apesar disso, as informações levantadas na etapa de revisão de literatura, mesmo que para o ambiente *batch*, forneceram subsídio suficiente para o desenvolvimento do RFAO.

Neste trabalho foi apresentado o algoritmo RFAO, em duas versões: *batch* e *stream*. Este algoritmo utiliza probabilidade, estatística e RLS para gerar instâncias sintéticas. O principal diferencial do método desenvolvido com relação aos concorrentes, se dá na capacidade de gerar instâncias que seguem o fluxo natural dos dados, através da utilização de RLS, aumentando a representatividade de S_{min} no espaço de características, garantindo que cada instância gerada esteja em um lugar seguro do espaço, evitando problemas de sobreposição.

Uma série de parâmetros foi adicionada ao método a fim de cobrir os mais variados cenários: do desbalanceamento brando ao desbalanceamento severo enfrentando problemas de sobreposição. Porém, apesar de existir uma grande quantidade de problemas binários de classificação, poucas bases de dados e *streams* desbalanceados encontram-se disponíveis publicamente. Destas poucas, existe um número pequeno que apresenta atributos correlacionados na composição de suas instâncias. A proposta inicial do método dava bastante ênfase para geração de valores via RLS, porém, ao perceber o comportamento supracitado, uma atenção extra foi dedicada à geração via probabilidade e estatística, estendendo a proposta inicial, que utilizava probabilidade apenas para geração de valores para atributos binários, para todos os atributos nominais.

O método proposto confirmou a hipótese inicial, de que é possível aumentar a performance de classificação correta de instâncias pertencentes a S_{min}, sem deteriorar a performance em S_{maj}. Ocorreram casos em que ocorreu deterioração da performance em S_{maj}, porém, o aumento de performance na classe oposta foi sempre superior.

Uma das principais limitações deste trabalho esta na falta de comparação com outras técnicas no ambiente *stream*. Tal fato é justificado pela quase inexistência de técnicas de re-amostragem desenvolvidas para este ambiente e a falta de disponibilidade das existentes para comparações. Outra limitação está em determinar os parâmetros ótimos para o RFAO para cada momento da *stream*.

Com a finalidade de sobrepujar as limitações supracitadas, os trabalhos futuros incluem a implementação de algumas técnicas de re-amostragem desenvolvidas para o ambiente *stream*, com a finalidade de comparação com o RFAO, tal como novos experimentos variando o classificador base. Com a finalidade de determinar os parâmetros ótimos, o conceito de combinação de classificadores pode ser justaposto ao RFAO. Esta ideia se baseia na geração de um dado número de *batchs* balanceados, cada um utilizando um conjunto de parâmetros diferentes, que serão utilizados para o treinamento de um conjunto de classificadores. Tal abordagem pode ser utilizada para determinar dinamicamente quais são os atributos ótimos para determinado momento, elevando a performance geral ao longo da *stream* e flexibilizando o método.

Por fim, a versão inicial do método, desenvolvida para o ambiente *batch*, deverá ser adaptada com as melhorias implementadas na versão final, um estudo detalhado deverá ser conduzido para determinar quais as características presentes nas bases de dados ou *streams* são mais favoráveis para aplicação do RFAO tal como o método será expandido para problemas multiclasse.

Referências

AGRAWAL, R.; IMIELINSKI, T.; SWAMI, A. Database mining: A performance perspective. *IEEE transactions on knowledge and data engineering*, IEEE, v. 5, n. 6, p. 914–925, 1993. Citado na página 80.

ANAND, A. et al. An approach for classification of highly imbalanced data using weighting and undersampling. *Amino acids*, Springer, v. 39, n. 5, p. 1385–1391, 2010. Citado na página 36.

BARDDAL, J. P. Agrupamento online: Uma abordagem baseada na teoria de redes sociais. 2015. Citado na página 21.

BATISTA, G. E.; PRATI, R. C.; MONARD, M. C. A study of the behavior of several methods for balancing machine learning training data. *ACM Sigkdd Explorations Newsletter*, ACM, v. 6, n. 1, p. 20–29, 2004. Citado 2 vezes nas páginas 34 e 36.

BELLINGER, C.; JAPKOWICZ, N.; DRUMMOND, C. Synthetic oversampling for advanced radioactive threat detection. In: IEEE. *Machine Learning and Applications (ICMLA), 2015 IEEE 14th International Conference on*. [S.l.], 2015. p. 948–953. Citado 4 vezes nas páginas 17, 23, 31 e 36.

BIFET, A. et al. Moa: Massive online analysis. *Journal of Machine Learning Research*, v. 11, n. May, p. 1601–1604, 2010. Citado 4 vezes nas páginas 8, 17, 21 e 78.

BIFET, A.; MORALES, G. D. F. Big data stream learning with samoa. In: IEEE. *Data Mining Workshop (ICDMW), 2014 IEEE International Conference on*. [S.l.], 2014. p. 1199–1202. Citado na página 20.

BRANCO, P.; TORGO, I. S.; RIBEIRO, R. P. A Survey of Predictive Modeling on Imbalanced Domains. v. 49, n. 2, p. 1–50, 2016. Citado 3 vezes nas páginas 25, 35 e 36.

BREIMAN, L. Bagging predictors. *Machine learning*, Springer, v. 24, n. 2, p. 123–140, 1996. Citado na página 20.

BREIMAN, L. Random forests. *Machine learning*, Springer, v. 45, n. 1, p. 5–32, 2001. Citado na página 24.

BUNKHUMPORNPAT, C.; SINAPIROMSARAN, K.; LURSINSAP, C. Safe-level-smote: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem. In: SPRINGER. *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. [S.l.], 2009. p. 475–482. Citado 2 vezes nas páginas 27 e 36.

BUNKHUMPORNPAT, C.; SINAPIROMSARAN, K.; LURSINSAP, C. Mute: Majority under-sampling technique. In: IEEE. *Information, Communications and Signal Processing (ICICS) 2011 8th International Conference on*. [S.l.], 2011. p. 1–4. Citado 2 vezes nas páginas 33 e 36.

BURGES, C. J. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, Springer, v. 2, n. 2, p. 121–167, 1998. Citado na página 30.

CAO, H.; TAN, V. Y.; PANG, J. Z. A parsimonious mixture of gaussian trees model for oversampling in imbalanced and multimodal time-series classification. *IEEE transactions on neural networks and learning systems*, IEEE, v. 25, n. 12, p. 2226 2239, 2014. Citado na página 36.

CAO, P. et al. Ensemble-based hybrid probabilistic sampling for imbalanced data learning in lung nodule cad. *Computerized Medical Imaging and Graphics*, Elsevier, v. 38, n. 3, p. 137–150, 2014. Citado na página 36.

CASTILLO, M. D. D.; SERRANO, J. I. A multistrategy approach for digital text categorization from imbalanced documents. *ACM SIGKDD Explorations Newsletter*, ACM, v. 6, n. 1, p. 70–79, 2004. Citado na página 36.

CATENI, S.; COLLA, V.; VANNUCCI, M. A method for resampling imbalanced datasets in binary classification tasks for real-world problems. *Neurocomputing*, Elsevier, v. 135, p. 32–41, 2014. Citado na página 36.

Charles Ditzler, G. Incremental Learning of Concept Drift from Imbalanced Data. p. 204, 2011. ISSN 1098-6596. Citado 3 vezes nas páginas 34, 35 e 36.

CHAWLA, N. V. et al. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, v. 16, p. 321–357, 2002. Citado 2 vezes nas páginas 26 e 36.

CHAWLA, N. V. et al. Smoteboost: Improving prediction of the minority class in boosting. In: SPRINGER. *European Conference on Principles of Data Mining and Knowledge Discovery*. [S.l.], 2003. p. 107–119. Citado 2 vezes nas páginas 24 e 36.

CHEESEMAN, P. et al. Autoclass: A bayesian classification system. In: *Machine Learning Proceedings 1988*. [S.l.]: Elsevier, 1988. p. 54–64. Citado na página 16.

CHEN, S.; HE, H. Sera: selectively recursive approach towards nonstationary imbalanced stream data mining. In: IEEE. *Neural Networks, 2009. IJCNN 2009. International Joint Conference on*. [S.l.], 2009. p. 522–529. Citado 2 vezes nas páginas 34 e 36.

CHOI, J. M. A selective sampling method for imbalanced data learning on support vector machines. Digital Repository@ Iowa State University, 2010. Citado 2 vezes nas páginas 30 e 36.

CORNISH, E. A.; FISHER, R. A. Moments and cumulants in the specification of distributions. *Revue de l'Institut international de Statistique*, JSTOR, p. 307–320, 1938. Citado na página 43.

CORTES, C.; VAPNIK, V. Support-vector networks. *Machine learning*, Springer, v. 20, n. 3, p. 273–297, 1995. Citado na página 71.

D'AGOSTINO, R. B. An omnibus test of normality for moderate and large size samples. *Biometrika*, JSTOR, p. 341–348, 1971. Citado na página 43.

DIETTERICH, T. Overfitting and undercomputing in machine learning. *ACM computing surveys (CSUR)*, ACM, v. 27, n. 3, p. 326–327, 1995. Citado na página 71.

DITZLER, G.; POLIKAR, R. An ensemble based incremental learning framework for concept drift and class imbalance. In: IEEE. *Neural Networks (IJCNN), The 2010 International Joint Conference on*. [S.l.], 2010. p. 1–8. Citado na página 24.

DOMINGOS, P.; HULTEN, G. Mining high-speed data streams. In: ACM. *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. [S.l.], 2000. p. 71–80. Citado na página 78.

DOUCETTE, J.; HEYWOOD, M. I. Gp classification under imbalanced data sets: Active sub-sampling and auc approximation. In: SPRINGER. *European Conference on Genetic Programming*. [S.l.], 2008. p. 266–277. Citado na página 36.

DOWNTON, F. Linear estimates with polynomial coefficients. *Biometrika*, JSTOR, v. 53, n. 1/2, p. 129–141, 1966. Citado na página 43.

DUBEY, R. et al. Analysis of sampling techniques for imbalanced data: an n= 648 adni study. *NeuroImage*, Elsevier, v. 87, p. 220–241, 2014. Citado na página 36.

DUBOIS, D.; PRADE, H. Rough fuzzy sets and fuzzy rough sets. *International Journal of General System*, Taylor & Francis, v. 17, n. 2-3, p. 191–209, 1990. Citado na página 30.

D'ADDABBO, A.; MAGLIETTA, R. Parallel selective sampling method for imbalanced and large data classification. *Pattern Recognition Letters*, Elsevier, v. 62, p. 61–67, 2015. Citado na página 36.

ELKAN, C. The foundations of cost-sensitive learning. In: LAWRENCE ERLBAUM ASSOCIATES LTD. *International joint conference on artificial intelligence*. [S.l.], 2001. v. 17, n. 1, p. 973–978. Citado na página 25.

ELWELL, R.; POLIKAR, R. Incremental learning of variable rate concept drift. In: SPRINGER. *International Workshop on Multiple Classifier Systems*. [S.l.], 2009. p. 142–151. Citado na página 24.

FREUND, Y.; SCHAPIRE, R. E. A desicion-theoretic generalization of on-line learning and an application to boosting. In: SPRINGER. *European conference on computational learning theory*. [S.l.], 1995. p. 23–37. Citado na página 25.

GALAR, M. et al. A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, IEEE, v. 42, n. 4, p. 463–484, 2012. Citado na página 23.

GAMA, J.; SEBASTIÃO, R.; RODRIGUES, P. P. On evaluating stream learning algorithms. *Machine learning*, Springer, v. 90, n. 3, p. 317–346, 2013. Citado na página 39.

GANTZ, J.; REINSEL, D. The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east. *IDC iView: IDC Analyze the future*, v. 2007, n. 2012, p. 1–16, 2012. Citado na página 16.

GHAZIKHANI, A.; MONSEFI, R.; YAZDI, H. S. Online cost-sensitive neural network classifiers for non-stationary and imbalanced data streams. *Neural Computing and Applications*, Springer, v. 23, n. 5, p. 1283–1295, 2013. Citado na página 23.

GHAZIKHANI, A.; MONSEFI, R.; YAZDI, H. S. Online neural network model for non-stationary and imbalanced data stream classification. *International Journal of Machine Learning and Cybernetics*, Springer, v. 5, n. 1, p. 51–62, 2014. Citado na página 34.

GODASE, A.; ATTAR, V. Classifier ensemble for imbalanced data stream classification. In: ACM. *Proceedings of the CUBE International Information Technology Conference.* [S.l.], 2012. p. 284–289. Citado 2 vezes nas páginas 24 e 36.

GOMES, H. M. ADVANCES IN NETWORK-BASED ENSEMBLE OF CLASSIFIERS FOR DATA STREAMS. 2017. Citado 3 vezes nas páginas 16, 21 e 22.

GONG, J.; KIM, H. Rhsboost: Improving classification performance in imbalance data. *Computational Statistics & Data Analysis*, Elsevier, v. 111, p. 1–13, 2017. Citado 2 vezes nas páginas 17 e 23.

HAN, H.; WANG, W.-Y.; MAO, B.-H. Borderline-smote: a new over-sampling method in imbalanced data sets learning. In: SPRINGER. *International Conference on Intelligent Computing.* [S.l.], 2005. p. 878–887. Citado 2 vezes nas páginas 27 e 36.

HAUKE, J.; KOSSOWSKI, T. Comparison of values of pearson's and spearman's correlation coefficients on the same sets of data. *Quaestiones geographicae*, De Gruyter Open Sp. z oo, v. 30, n. 2, p. 87, 2011. Citado na página 43.

HE, H. et al. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In: IEEE. *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on.* [S.l.], 2008. p. 1322–1328. Citado 2 vezes nas páginas 30 e 36.

HE, H.; GARCIA, E. A. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, v. 21, n. 9, p. 1263–1284, 2009. ISSN 10414347. Citado 4 vezes nas páginas 16, 18, 25 e 34.

JIAN, C.; GAO, J.; AO, Y. A new sampling method for classifying imbalanced data based on support vector machine ensemble. *Neurocomputing*, Elsevier, v. 193, p. 115–122, 2016. Citado na página 36.

KAMEL, M. S.; WANAS, N. M. Data dependence in combining classifiers. In: SPRINGER. *International Workshop on Multiple Classifier Systems.* [S.l.], 2003. p. 1–14. Citado na página 24.

KARAHOCA, A. Advances in data mining knowledge discovery and applications. InTech, 2012. Citado 2 vezes nas páginas 8 e 41.

KENDALL, M. G. A new measure of rank correlation. *Biometrika*, JSTOR, v. 30, n. 1/2, p. 81–93, 1938. Citado na página 43.

KENNEDY, J. Particle swarm optimization. In: *Encyclopedia of machine learning.* [S.l.]: Springer, 2011. p. 760–766. Citado na página 24.

KOTSIANTIS, S. B.; ZAHARAKIS, I.; PINTELAS, P. *Supervised machine learning: A review of classification techniques.* 2007. Citado na página 16.

KRAWCZYK, B. Learning from imbalanced data: open challenges and future directions. *Progress in Artificial Intelligence*, v. 5, n. 4, p. 221–232, 2016. ISSN 2192-6360. Disponível em: <http://dx.doi.org/10.1007/s13748-016-0094-0>. Citado na página 23.

KUMAR, N. S. et al. Undersampled k-means approach for handling imbalanced distributed data. *Progress in Artificial Intelligence*, Springer, v. 3, n. 1, p. 29–38, 2014. Citado 2 vezes nas páginas 33 e 36.

KUMAR, V. et al. Credit risk analysis in peer-to-peer lending system. In: IEEE. *Knowledge Engineering and Applications (ICKEA), IEEE International Conference on.* [S.l.], 2016. p. 193–196. Citado na página 17.

LEMAITRE, G.; NOGUEIRA, F.; ARIDAS, C. K. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *CoRR*, abs/1609.06570, 2016. Disponível em: <http://arxiv.org/abs/1609.06570>. Citado na página 72.

LI, C. Classifying imbalanced data using a bagging ensemble variation (bev). In: ACM. *Proceedings of the 45th annual southeast regional conference.* [S.l.], 2007. p. 203–208. Citado na página 24.

LI, H. et al. Multi-window based ensemble learning for classification of imbalanced streaming data. *World Wide Web*, Springer, v. 20, n. 6, p. 1507–1525, 2017. Citado na página 24.

LI, J. et al. *WITHDRAWN: Adaptive Swarm Balancing Algorithms for rare-event prediction in imbalanced healthcare data.* [S.l.]: Elsevier, 2016. Citado 2 vezes nas páginas 32 e 36.

LICHMAN, M. UCI machine learning repository. 2013. Disponível em: <http://archive.ics.uci.edu/ml>. Citado na página 72.

LING, C. X. et al. Decision trees with minimal costs. In: ACM. *Proceedings of the twenty-first international conference on Machine learning.* [S.l.], 2004. p. 69. Citado na página 25.

LYON, R. J. et al. Hellinger distance trees for imbalanced streams. *CoRR*, abs/1405.2278, 2014. Disponível em: <http://arxiv.org/abs/1405.2278>. Citado na página 37.

MACIEJEWSKI, T.; STEFANOWSKI, J. Local neighbourhood extension of smote for mining imbalanced data. In: IEEE. *Computational Intelligence and Data Mining (CIDM), 2011 IEEE Symposium on.* [S.l.], 2011. p. 104–111. Citado 2 vezes nas páginas 29 e 36.

MAGEE, J. F. *Decision trees for decision making.* [S.l.]: Harvard Business Review, 1964. Citado na página 71.

MALEKIPIRBAZARI, M.; AKSAKALLI, V. Risk assessment in social lending via random forests. *Expert Systems with Applications*, Elsevier, v. 42, n. 10, p. 4621–4631, 2015. Citado na página 17.

MANI, I.; ZHANG, I. knn approach to unbalanced data distributions: a case study involving information extraction. In: *Proceedings of workshop on learning from imbalanced datasets.* [S.l.: s.n.], 2003. Citado 2 vezes nas páginas 32 e 36.

MAO, W. et al. Two-stage hybrid extreme learning machine for sequential imbalanced data. In: *Proceedings of ELM-2015 Volume 1.* [S.l.]: Springer, 2016. p. 423–433. Citado na página 36.

NEKOOEIMEHR, I.; LAI-YUEN, S. K. Adaptive semi-unsupervised weighted oversampling (a-suwo) for imbalanced datasets. *Expert Systems with Applications*, Elsevier, v. 46, p. 405–416, 2016. Citado 2 vezes nas páginas 31 e 36.

OZA, N. C. Online bagging and boosting. In: IEEE. *Systems, man and cybernetics, 2005 IEEE international conference on*. [S.l.], 2005. v. 3, p. 2340 2345. Citado na página 20.

PANIGRAHI, S. et al. Credit card fraud detection: A fusion approach using dempster–shafer theory and bayesian learning. *Information Fusion*, Elsevier, v. 10, n. 4, p. 354–363, 2009. Citado na página 23.

PEARSON, K. Notes on the history of correlation. *Biometrika*, JSTOR, v. 13, n. 1, p. 25–45, 1920. Citado na página 43.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, v. 12, n. Oct, p. 2825- 2830, 2011. Citado na página 71.

PROVOST, F. J.; FAWCETT, T.; KOHAVI, R. The case against accuracy estimation for comparing induction algorithms. In: *ICML*. [S.l.: s.n.], 1998. v. 98, p. 445–453. Citado na página 38.

RAMENTOL, E. et al. Smote-rsb*: a hybrid preprocessing approach based on oversampling and undersampling for high imbalanced data-sets using smote and rough sets theory. *Knowledge and information systems*, Springer, v. 33, n. 2, p. 245–265, 2012. Citado na página 36.

RODRIGUES, P. On evaluating stream learning algorithms. v. 90, n. 3, p. 317–346, 2013. Citado 2 vezes nas páginas 38 e 39.

SAMUEL, A. L. Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, IBM, v. 3, n. 3, p. 210–229, 1959. Citado na página 16.

SCHAPIRE, R. E. The boosting approach to machine learning: An overview. In: *Nonlinear estimation and classification*. [S.l.]: Springer, 2003. p. 149–171. Citado na página 24.

SEIFFERT, C. et al. Rusboost: A hybrid approach to alleviating class imbalance. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, IEEE, v. 40, n. 1, p. 185–197, 2010. Citado na página 36.

SHAPIRO, S. S.; WILK, M. B. An analysis of variance test for normality (complete samples). *Biometrika*, Biometrika Trust, v. 52, n. 3-4, p. 591–611, 1965. Citado na página 43.

SILVA, J. A. et al. Data stream clustering: A survey. *ACM Computing Surveys (CSUR)*, ACM, v. 46, n. 1, p. 13, 2013. Citado na página 21.

SILVA, J. P. B. C. d. *Modelos de regressão linear e logística utilizando o software R*. Tese (Doutorado), 2017. Citado na página 42.

SOBHANI, P.; VIKTOR, H.; MATWIN, S. Learning from imbalanced data using ensemble methods and cluster-based undersampling. In: SPRINGER. *International Workshop on New Frontiers in Mining Complex Patterns*. [S.l.], 2014. p. 69–83. Citado na página 36.

SPEARMAN, C. The proof and measurement of association between two things. *The American journal of psychology*, JSTOR, v. 15, n. 1, p. 72–101, 1904. Citado na página 43.

SPYROMITROS-XIOUFIS, E. *Dealing with concept drift and class imbalance in multi-label stream classification*. Tese (Doutorado) — Department of Computer Science, Aristotle University of Thessaloniki, 2011. Citado 2 vezes nas páginas 35 e 36.

STREET, W. N.; KIM, Y. A streaming ensemble algorithm (sea) for large-scale classification. In: ACM. *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. [S.l.], 2001. p. 377–382. Citado na página 80.

SUN, Y.; WONG, A. K. C.; KAMEL, M. S. Classification of imbalanced data: A review. *International Journal of Pattern Recognition and Artificial Intelligence*, v. 23, n. 4, p. 687–719, 2009. ISSN 0218-0014. Disponível em: <http://www.worldscientific.com/doi/abs/10.1142/S0218001409007326>. Citado 3 vezes nas páginas 23, 24 e 26.

TOMEK, I. Two modifications of cnn. *IEEE Trans. Systems, Man and Cybernetics*, v. 6, p. 769–772, 1976. Citado 2 vezes nas páginas 32 e 36.

TOPOUZELIS, K. N. Oil spill detection by sar images: dark formation detection, feature extraction and classification algorithms. *Sensors*, Molecular Diversity Preservation International, v. 8, n. 10, p. 6642–6659, 2008. Citado 2 vezes nas páginas 17 e 23.

TSAI, K.; RAMIAH, S.; SINGH, S. *Peer Lending Risk Predictor*. [S.l.]: Stanford University, 2014. Citado na página 17.

VEDALA, R.; KUMAR, B. R. An application of naive bayes classification for credit scoring in e-lending platform. In: IEEE. *Data Science & Engineering (ICDSE), 2012 International Conference on*. [S.l.], 2012. p. 81–84. Citado na página 17.

VERBIEST, N. et al. Improving smote with fuzzy rough prototype selection to detect noise in imbalanced classification data. In: SPRINGER. *Ibero-American Conference on Artificial Intelligence*. [S.l.], 2012. p. 169–178. Citado 2 vezes nas páginas 30 e 36.

VO, N. H.; WON, Y. Classification of unbalanced medical data with weighted regularized least squares. In: IEEE. *Frontiers in the Convergence of Bioscience and Information Technologies, 2007. FBIT 2007*. [S.l.], 2007. p. 347–352. Citado 2 vezes nas páginas 17 e 23.

WANG, H.; ABRAHAM, Z. Concept drift detection for imbalanced stream data. *arXiv Preprint*, 2015. Citado na página 34.

WANG, S.; MINKU, L. L.; YAO, X. A multi-objective ensemble method for online class imbalance learning. In: IEEE. *Neural Networks (IJCNN), 2014 International Joint Conference on*. [S.l.], 2014. p. 3311–3318. Citado na página 23.

WIDMER, G.; KUBAT, M. Learning in the presence of concept drift and hidden contexts. *Machine learning*, Springer, v. 23, n. 1, p. 69–101, 1996. Citado na página 22.

WILCOXON, F. Individual comparisons by ranking methods. *Biometrics bulletin*, JSTOR, v. 1, n. 6, p. 80–83, 1945. Citado na página 78.

YANG, P. et al. A particle swarm based hybrid system for imbalanced medical data sampling. *BMC genomics*, BioMed Central, v. 10, n. 3, p. S34, 2009. Citado 2 vezes nas páginas 24 e 36.

YEN, S.-J.; LEE, Y.-S. Cluster-based under-sampling approaches for imbalanced data distributions. *Expert Systems with Applications*, Elsevier, v. 36, n. 3, p. 5718–5727, 2009. Citado na página 36.

ZADROZNY, B.; LANGFORD, J.; ABE, N. Cost-sensitive learning by cost-proportionate example weighting. In: IEEE. *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on.* [S.l.], 2003. p. 435–442. Citado na página 25.

ZHANG, C.; TAN, K. C.; REN, R. Training cost-sensitive deep belief networks on imbalance data problems. In: IEEE. *Neural Networks (IJCNN), 2016 International Joint Conference on.* [S.l.], 2016. p. 4362–4367. Citado na página 23.

ZHANG, D. et al. A novel improved smote resampling algorithm based on fractal. *Journal of Computational Information Systems*, v. 7, n. 6, p. 2204–2211, 2011. Citado na página 36.

ZHANG, H. The optimality of naive bayes. *AA*, v. 1, n. 2, p. 3, 2004. Citado na página 71.