UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

CAMILA MAIONE

# Balanceamento de dados com base em oversampling em dados transformados

Goiânia
2020

UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

# TERMO DE CIÊNCIA E DE AUTORIZAÇÃO (TECA) PARA DISPONIBILIZAR VERSÕES ELETRÔNICAS DE TESES E DISSERTAÇÕES NA BIBLIOTECA DIGITAL DA UFG

Na qualidade de titular dos direitos de autor, autorizo a Universidade Federal de Goiás (UFG) a disponibilizar, gratuitamente, por meio da Biblioteca Digital de Teses e Dissertações (BDTD/UFG), regulamentada pela Resolução CEPEC nº 832/2007, sem ressarcimento dos direitos autorais, de acordo com a Lei 9.610/98, o documento conforme permissões assinaladas abaixo, para fins de leitura, impressão e/ou download, a título de divulgação da produção científica brasileira, a partir desta data.

O conteúdo das Teses e Dissertações disponibilizado na BDTD/UFG é de responsabilidade exclusiva do autor. Ao encaminhar o produto final, o autor(a) e o(a) orientador(a) firmam o compromisso de que o trabalho não contém nenhuma violação de quaisquer direitos autorais ou outro direito de terceiros.

**1. Identificação do material bibliográfico**

[ ] Dissertação [ x ] Tese

**2. Nome completo do autor**

Camila Maione

**3. Título do trabalho**

Balanceamento de dados com base em oversampling em dados transformados

**4. Informações de acesso ao documento (este campo deve ser preenchido pelo orientador)**

Concorda com a liberação total do documento [ X ] SIM [ ] NÃO[1]

**[1]** Neste caso o documento será embargado por até um ano a partir da data de defesa. Após esse período, a possível disponibilização ocorrerá apenas mediante:
**a)** consulta ao(à) autor(a) e ao(à) orientador(a);
**b)** novo Termo de Ciência e de Autorização (TECA) assinado e inserido no arquivo da tese ou dissertação.
O documento não será disponibilizado durante o período de embargo.
Casos de embargo:
- Solicitação de registro de patente;
- Submissão de artigo em revista científica;
- Publicação como capítulo de livro;
- Publicação da dissertação/tese em livro.

**Obs. Este termo deverá ser assinado no SEI pelo orientador e pelo autor.**

Documento assinado eletronicamente por **Rommel Melgaço Barbosa**, **Professor do Magistério Superior**, em 05/10/2020, às 11:24, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **CAMILA MAIONE**, **Discente**, em 16/10/2020, às 20:53, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://sei.ufg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **1569876** e o código CRC **940A6A5D**.

CAMILA MAIONE

# Balanceamento de dados com base em oversampling em dados transformados

Tese apresentada ao Programa de Pós–Graduação do Instituto de Informática da Universidade Federal de Goiás, como requisito parcial para obtenção do título de Doutor em Computação.

**Área de concentração:** Ciência da Computação.

**Orientador:** Prof. Rommel Melgaço Barbosa

Goiânia
2020

Ficha de identificação da obra elaborada pelo autor, através do
Programa de Geração Automática do Sistema de Bibliotecas da UFG.

Maione, Camila
Balanceamento de dados com base em oversampling em dados transformados [manuscrito] / Camila Maione. - 2020.
135 f.: il.

Orientador: Prof. Dr. Rommel Melgaço Barbosa.
Tese (Doutorado) - Universidade Federal de Goiás, Instituto de Informática (INF), Programa de Pós-Graduação em Ciência da Computação em rede (UFG/UFMS), Goiânia, 2020.
Bibliografia. Apêndice.
Inclui gráfico, tabelas, algoritmos, lista de figuras, lista de tabelas.

1. mineração de dados. 2. classificação de dados. 3. aprendizagem de máquina. 4. balanceamento de dados. 5. transformação de dados. I. Melgaço Barbosa, Rommel , orient. II. Título.

CDU 004

UNIVERSIDADE FEDERAL DE GOIÁS

INSTITUTO DE INFORMÁTICA

**ATA DE DEFESA DE TESE**

Ata Nº **05/2020** da sessão de Defesa de Tese de **Camila Maione** que confere o título de Doutora em Ciência da Computação, na área de concentração em Ciência da Computação**.**

Aos dezessete dias do mês de agosto de dois mil e vinte, a partir das catorze horas, via sistema de weconferência da RNP, realizou-se a sessão pública de Defesa de Tese intitulada **"Balanceamento de dados com base em oversampling em dados transformados**". Os trabalhos foram instalados pelo Orientador, Professor Doutor Rommel Melgaço Barbosa (INF/UFG) com a participação dos demais membros da Banca Examinadora: Professora Doutora Ana Paula Cabral Seixas Costa (PPGEP/UFPE), membra titular externa; Professora Doutora Kátia Kelvis Cassiano Lozano (FIC/UFG), membra titular externa; Professor Doutor Plínio de Sá Leitão Júnior (INF/UFG), membro titular interno; e Professor Doutor Ronaldo Martins da Costa (INF/UFG), membro titular interno. A realização da banca ocorreu por meio de videoconferência, em atendimento à recomendação de suspensão das atividades presenciais na UFG emitida pelo Comitê UFG para o Gerenciamento da Crise COVID-19, bem como à recomendação de isolamento social da Organização Mundial de Saúde e do Ministério da Saúde para enfrentamento da emergência de saúde pública decorrente do novo coronavírus. Durante a argüição os membros da banca não fizeram sugestão de alteração do título do trabalho. A Banca Examinadora reuniu-se em sessão secreta a fim de concluir o julgamento da Tese tendo sido a candidata **aprovada** pelos seus membros. Proclamados os resultados pelo Professor Doutor Rommel Melgaço Barbosa, Presidente da Banca Examinadora, foram encerrados os trabalhos e, para constar, lavrou-se a presente ata que é assinada pelos Membros da Banca Examinadora, aos dezessete dias do mês de agosto de dois mil e vinte.

TÍTULO SUGERIDO PELA BANCA

Documento assinado eletronicamente por **Rommel Melgaço Barbosa**, **Professor do Magistério Superior**, em 17/08/2020, às 18:16, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Ana Paula Cabral Seixas Costa**, **Usuário Externo**, em 17/08/2020, às 18:18, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Plinio De Sa Leitão Junior**, **Professor do Magistério Superior**, em 17/08/2020, às 18:18, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Ronaldo Martins Da Costa**, **Professor do Magistério Superior**, em 17/08/2020, às 18:18, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **CAMILA MAIONE**, **Discente**, em 17/08/2020, às 18:31, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Katia Kelvis Cassiano Lozano**, **Professor do Magistério Superior**, em 17/08/2020, às 22:40, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://sei.ufg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **1458809** e o código CRC **D0C1C79D**.

**Referência:** Processo nº 23070.032961/2020-15 SEI nº 1458809

**Camila Maione**

Mestre e bacharela em Ciências da Computação pela Universidade Federal de Goiás (UFG), com ênfase na área de mineração de dados e descoberta de informação. Durante o mestrado e o doutorado, foi bolsista da CAPES e realizou diversas pesquisas na área, com foco em desenvolvimento e aplicação de técnicas de análise multivariada de dados, aprendizado de máquina supervisionado e não supervisionado e seleção de variáveis para análise de conjuntos de dados reais de diferentes domínios, tamanhos e complexidades, que lhe renderam 9 publicações acadêmicas em revistas científicas internacionais bem conceituadas.

À minha mãe, Zildete.

# Agradecimentos

Agradeço primeiramente ao meu orientador, Rommel Melgaço Barbosa, por me receber sob sua orientação, pela sua condução e por todo o conhecimento a mim transmitido no decorrer destes seis anos de pós-graduação, pela sua acessibilidade, solicitude e pela oportunidade de participar ativamente dos projetos de pesquisa descritos neste trabalho.

Agradeço a minha família e amigos por toda a motivação e incentivo. Agradeço também aos meus falecidos avós, Azíria e Hercídio, por todo o auxílio fundamental prestado em meus anos escolares e cuja lembrança por si só me motiva a dar sempre o melhor de mim.

Meus mais sinceros agradecimentos à minha melhor amiga, Aline Mikado. Não existem palavras capazes de descrever minha gratidão pela sua paciência, seu carinho, pelos seus mimos, por toda a motivação, por dividir comigo o júbilo de minhas conquistas e por me suportar nos meus piores momentos no decorrer desta longa jornada.

Agradeço, principalmente, à minha mãe Zildete, que é a minha maior motivação por trás de meus atos e a quem dedico todas as minhas conquistas. Obrigada pelo seu esmero em minha criação e por me garantir sempre os melhores recursos de aprendizado ao seu alcance, que foram de fundamental importância para que eu concretizasse esta etapa. Obrigada pelo seu eterno e incondicional suporte, emocional e financeiro, nos momentos de dificuldade encontrados no decorrer destes dois anos.

Obrigada, equipe do Centro Espírita Cabocla Jurema, especialmente Antônio Terra e Carmem Lúcia, por me acolherem e apaziguarem minha mente e espírito em meus momentos de maior dificuldade nos momentos derradeiros dos meus estudos, e por não me deixarem esquecer o valor da fé, da paciência, da auto-confiança e da caridade.

Agradeço aos meus colegas de orientação, Nattane, Diego, Guilherme e Márcio, pelas conversas descontraídas, pelos bons momentos, pelas experiências e ideias compartilhadas, pelo apreço e pela motivação.

Agradeço aos membros:

- do Laboratório de Toxicologia e Essencialidade de Metais da Universidade de São Paulo, Vanessa de Oliveira Souza, Fernando Barbosa Junior, Eloisa Silva de Paula, Matheus Gallimberti, Airton da Cunha Martins Junior e Ana Carolina Paulelli;

- do Laboratório de Química e Toxicologia Forense do Instituto de Criminalística de São Paulo, Loraine Rezende Togni e José Luiz da Costa;
- do Department of Chemistry da University of Central Florida, Andres Dobal Campiglia;
- do Centro de Ciências Naturais e Humanas da Universidade Federal do ABC, Fabiana Roberta Segura e Bruno Lemos Batista;
- do Centro de Energia Nuclear na Agricultura da Universidade de São Paulo, Christian Turra, Elisabete de Nadai Fernandes e Márcio Arruda Bacchi;
- do Departamento de Ciência do Solo da Universidade de São Paulo, Eloá Moura Araujo, Sabrina Novaes dos Santos-Araújo, Alexys Giorgia Friol Boim e Luís Reynaldo Ferracciú Alleoni;

pelas suas parcerias, pelas oportunidades dadas a mim e pelas contribuições nas pesquisas realizadas.

Agradeço toda a equipe técnica e administrativa do Instituto de Informática, especialmente Mirian e Mariana, por sua presteza, gentileza e assistência nas diversas dúvidas levantadas no decorrer do programa.

Agradeço à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelo auxílio financeiro.

E, finalmente, agradeço a todo o corpo docente do Instituto de Informática, pelo seu inestimável empenho pelo instituto e pelos seus alunos, e por todo o conhecimento e experiências compartilhados comigo desde os anos de graduação.

Há uma quantidade enorme de informações ocultas em bases de dados - informações que são potencialmente importantes mas ainda não foram descobertas ou articuladas. A nossa missões é trazê-las à tona.

**Ian H. Witten, Eibe Frank, Mark A. Hall**,
*Data Mining - Practical Machine Learning Tools and Techniques*.

# Resumo

Maione, Camila. **Balanceamento de dados com base em oversampling em dados transformados**. Goiânia, 2020. 137p. Tese de Doutorado. Instituto de Informática, Universidade Federal de Goiás.

**Introdução:** A eficiência e confiabilidade de análises de bases de dados dependem da qualidade da base de dados em questão. O processo de preparação de bases de dados para torná-las mais limpas, representativas e de melhor qualidade chama-se pré-processamento de dados, durante o qual também é realizado o balanceamento dos dados. A importância de balancear os dados jaz no fato de que diversos modelos de classificação utilizados em projetos coorporativos e acadêmicos são projetados para trabalhar com conjuntos de dados balanceados, e há diversos outros fatores degradadores de desempenho de classificação que estão associados ao desbalanceamento de dados.
**Objetivo:** Propõe-se uma nova abordagem para balanceamento de dados, baseada em transformação de dados combinada com *resampling* de dados transformados. A abordagem proposta transforma o conjunto de dados original através da transformação de suas variáveis descritoras, consequentemente alterando a posição das amostras de dados no plano dimensional, influenciando a escolha que algoritmos de resampling como o SMOTE fazem sobre as amostras iniciais, seus vizinhos mais próximos e onde posicionar as amostras sintéticas geradas.
**Métodos:** Uma implementação inicial baseada em análise de componentes principais (PCA) e SMOTE é apresentada, chamado PCA-SMOTE. Para testar a qualidade do balanceamento realizado pelo PCA-SMOTE, 12 bases de dados de teste foram balanceadas utilizando o PCA-SMOTE e outros três métodos de balanceamento populares na literatura, e o desempenho de três modelos de classificação diferentes treinados com tais bases foram avaliados e comparados.
**Resultados:** Diversos modelos de classificação treinados com bases balanceadas através do método proposto mostraram desempenho superior ou similar aos dos modelos treinados com bases balanceadas pelos outros algoritmos populares, como Borderline-SMOTE, Safe-Level-SMOTE e ADASYN, em diversos casos de teste.
**Conclusões:** Os resultados satisfatórios obtidos comprovam o potencial que o PCA-SMOTE possui para melhorar o aprendizado de classificadores sobre bases de dados desbalanceadas.

**Palavras–chave**

mineração de dados, classificação de dados, aprendizagem de máquina, balanceamento de dados, transformação de dados, pré-processamento de dados

# Abstract

Maione, Camila. **Data balancing based on oversampling on transformed data**. Goiânia, 2020. 137p. PhD. Thesis. Instituto de Informática, Universidade Federal de Goiás.

**Introduction:** The efficiency and reliability of data analyses depends heavily on the quality of the analyzed data. The fundamental process of preparing databases in order to make them cleaner, more representative and improve their quality is called data preprocessing, during which data balancing is also performed. The importance of data balancing lies in the fact that several classification models commonly employed in enterprises and academic projects are designed to work with balanced data sets, and there are several factors which hinder classification performance which are associated to data imbalance.

**Objective:** A new approach for data balancing based on data transformation combined with *resampling* of transformed data is proposed. The proposed approach transforms the original data set by transforming its input variables into new ones, therefore altering the data samples' position in the dimensional plane and consequently the choice that SMOTE-based resampling algorithms make over the initial samples, their nearest neighbours and where to place the generated synthetic samples.

**Methods:** An initial implementation based on Principal Component Analysis (PCA) and SMOTE is presented, called PCA-SMOTE. In order to test the quality of the balancing performed by PCA-SMOTE, twelve test data sets were balanced through PCA-SMOTE and three other popular data balancing methods, and the performance of three classification models trained on these balanced sets are assessed and compared.

**Results:** Several classification models trained on data sets which were balanced using the proposed method presented higher or similar performance measures in comparison to the same models trained on data sets that were balanced through the other evaluated algorithms, such as Borderline-SMOTE, Safe-Level-SMOTE and ADASYN.

**Conclusion:** The satisfactory results obtained prove the potential of the proposed algorithm to improve learning of classifiers on imbalanced data sets.

**Keywords**

data mining, data classification, machine learning, imbalanced data, data transformation, data preprocessing

# Sumário


Lista de Figuras **13**

Lista de Tabelas **15**

1 Apresentação **17**

2 Balanceamento de dados **19**
- 2.1 Visão geral 19
- 2.2 Problemas causados pelo desbalanceamento de dados 21
  - 2.2.1 Degradação de desempenho 22
  - 2.2.2 Métricas de desempenho adequadas 24
- 2.3 Métodos de sampling 26
  - 2.3.1 Random sampling 27
  - 2.3.2 Informed sampling 28
  - 2.3.3 Synthetic Minority Oversampling Technique (SMOTE) 28
  - 2.3.4 Algoritmos baseados em SMOTE 30
- 2.4 Aprendizagem cost-sensitive 33
  - 2.4.1 Métodos baseados em ponderação de amostras 34
  - 2.4.2 Métodos baseados em modificações de modelos de classificação 35
- 2.5 Conclusão 35

3 Classificação de dados **37**
- 3.1 Classificação de dados 38
- 3.2 Modelos de classificação 40
  - 3.2.1 Máquinas vetores de suporte 40
  - 3.2.2 Redes neurais artificiais 44
  - 3.2.3 Árvores de decisão 46
- 3.3 Medidas de desempenho de classificadores 49
- 3.4 Problemas de classificação de dados publicados 52
  - 3.4.1 Estudo dos elementos químicos mais importantes para a autenticação de folhas de laranja orgânicas 52
  - 3.4.2 Discriminação de tabletes de *ecstasy* apreendidos em Campinas e Ribeirão Preto, Brasil 54
  - 3.4.3 Diferenciação de tabletes de chocolate fabricados de maneira orgânica e convencional 55
- 3.5 Conclusão 56

4 Método proposto **57**
4.1 Considerações iniciais 57
4.1.1 Análise dos componentes principais (PCA) 60
4.2 Descrição do PCA-SMOTE 66

5 Testes realizados e resultados obtidos **70**
5.1 Metodologia de teste 70
5.2 Bases de dados utilizadas 72
5.3 Configurações do ambiente de teste 78
5.4 Resultados obtidos 78
5.4.1 Breast Cancer Coimbra 80
5.4.2 Arroz 82
5.4.3 Chocolate orgânico 84
5.4.4 Folhas de laranjeiras orgânicas 86
5.4.5 Suco de uva orgânico 88
5.4.6 Breast Cancer Wisconsin (Original) 91
5.4.7 Glass Identification 94
5.4.8 Wine Quality 96
5.4.9 Yeast 99
5.4.10 Pima Indians Diabetes 101
5.4.11 Ecoli 103
5.4.12 Thyroid Disease 105
5.5 Discussão 107

6 Considerações finais **111**

Referências Bibliográficas **113**

A Código: pca-smote.R **126**

B Códigos auxiliares **128**
B.1 main.R 128
B.2 classificationfunctions.R 129
B.3 main_classification.R 131
B.4 main_printresults1.R 133
B.5 main_printsigntests.R 134
B.6 performanceEval.R 135
B.7 smotefunctions.R 136

# Lista de Figuras


2.1 Exemplos de conjunto de dados balanceado *esquerda* e outro conjunto de dados desbalanceado *direita*. 19
2.2 Baixa densidade de dados no conjunto de dados (yeast4) [72]. 23
2.3 Exemplo de pequenos disjuntos em um conjunto de dados. 23
2.4 Conjunto de dados sem sobreposição de dados (à esquerda) e com sobreposição de dados (à direita) [23]. 24
2.5 Gráfico ROC contendo duas curvas referentes aos verdadeiros e falsos positivos obtidos por dois modelos de classificação A e B. [31]. 26
2.6 Funcionamento do algoritmo *EasyEnsemble*. MAJ e MIN referem-se aos conjunto de dados das classes majoritária e minoritária, respectivamente. $MAJ_1$, $MAJ_2$ e $MAJ3$ são partições independentes de $MAJ$. 28
2.7 Representação gráfica do algoritmo SMOTE [109]. 30
2.8 Representação gráfica do algoritmo *Borderline-SMOTE* [46]. (a) Distribuição original do conjunto de dados *Circle*. (b) Amostras da classe minoritária nas bordas do limite de decisão (quadrados preenchidos). (c) Amostras sintéticas geradas pelo *Borderline-SMOTE* (quadrados vazios). 31
2.9 Matriz de custo. 33

3.1 Processo de descoberta de conhecimento em bancos de dados (KDD) [119]. 38
3.2 Exemplo de classificador. 40
3.3 Possíveis limites de decisão para um conjunto de dados. 41
3.4 Margem do limite de decisão. 42
3.5 Limites de decisão, margens do limite de decisão e parâmetros do SVM. 43
3.6 Transformação de um conjunto de dados originalmente bidimensional ($\Re^2$) e não separáveis linearmente para um espaço tridimensional ($\Re^3$) onde eles são linearmente separáveis [57]. 43
3.7 Desenho arquitetural de um modelo perceptron básico. 44
3.8 Exemplo de um multilayer perceptron [119]. 45
3.9 Exemplo de estrutura de uma árvore de decisão [97]. 48
3.10 Execução do método de validação cruzada *k-fold*. 51

4.1 Exemplo de execução do SMOTE com sobreposição de dados. 59
(a) Conjunto de dados original. 59
(b) Seleção da amostra $s$ da classe minoritária e dos seus 3 vizinhos mais próximos $v1$, $v2$ e $v3$, também da classe minoritária. 59
(c) Amostras sintéticas $g1$, $g2$ e $g3$ da classe minoritária são geradas e posicionadas no segmento que liga a amostra $s$ aos seus vizinhos mais próximos. 59

4.2 Gráfico de um conjunto de dados bidiomensional. 61
4.3 Os mesmos dados mostrados na Fig. 4.2, expressos agora apenas em termos de $x$. 61
4.4 Quantidade de variância total dos dados expressada por cada componente principal em um exemplo arbitrário. PC1 expressa 35% da variância total, PC2 expressa 20% da variância total etc. 63
4.5 Gráfico PCA gerado para o conjunto de dados ilustrado na Fig. 4.2. 65
4.6 Exemplo de execução do SMOTE combinado com transformação de variáveis pelo PCA. 67
(a) Gráfico PCA gerado para o conjunto de dados ilustrado na Fig. 4.2. 67
(b) Amostra $s$ da classe minoritária e seus vizinhos da classe minoritária mais próximos $v1$, $v2$ e $v3$ selecionados. 67
(c) Amostras sintéticas $g1$, $g2$ e $g3$ da classe minoritária geradas e posicionadas nos segmentos entre $s$ e seus vizinhos. 67
4.7 Fluxo de execução do algoritmo PCA-SMOTE. 69

5.1 Base de dados **Breast Cancer Coimbra** expressa em termos de suas duas componentes principais antes e após a aplicação do PCA-SMOTE. 81
5.2 Base de dados arroz expressa em termos de suas duas componentes principais antes e após a aplicação do PCA-SMOTE. 83
5.3 Base de dados de chocolate orgânico expressa em termos de suas duas componentes principais antes e após a aplicação do PCA-SMOTE. 85
5.4 Base de dados de folhas de laranjeiras orgânicas expressa em termos de suas duas componentes principais antes e após a aplicação do PCA-SMOTE. 87
5.5 Base de dados de suco de uva orgânico expressa em termos de suas duas componentes principais antes e após a aplicação do PCA-SMOTE. 89
5.6 Base de dados **Breast Cancer Wisconsin Original** expressa em termos de suas duas componentes principais antes e após a aplicação do PCA-SMOTE. 92
5.7 Base de dados **Glass Identification** expressa em termos de suas duas componentes principais antes e após a aplicação do PCA-SMOTE. 95
5.8 Base de dados **Wine Quality** expressa em termos de suas duas componentes principais antes e após a aplicação do PCA-SMOTE. 97
5.9 Base de dados **Yeast** expressa em termos de suas duas componentes principais antes e após a aplicação do PCA-SMOTE. 100
5.10 Base de dados **Pima Indians Diabetes** expressa em termos de suas duas componentes principais antes e após a aplicação do PCA-SMOTE. 102
5.11 Base de dados **Ecoli** expressa em termos de suas duas componentes principais antes e após a aplicação do PCA-SMOTE. 104
5.12 Base de dados **Thyroid Disease** expressa em termos de suas duas componentes principais antes e após a aplicação do PCA-SMOTE. 106

# Lista de Tabelas


2.1 Exemplos de propostas sugeridas para os modelos de classificação SVM, redes neurais artificiais e KNN incorporando aprendizado *cost-sensitive*. 36

3.1 Exemplos de funções kernel para uso com o SVM. 44
3.2 Conjunto de treinamento [97]. 47
3.3 Estrutura de uma matriz de confusão gerada para um modelo de classificação binário. 52
3.4 Medidas de desempenho computadas para os modelos SVM e RNA construídos considerando todos os elementos químicos determinados. 53
3.5 Medidas de desempenho computadas para os modelos SVM e RNA. 54

4.1 Exemplo de matriz de covariância para o conjunto de variáveis $a_1, a_2, ..., a_m$. 64
4.2 Exemplo de conjunto de dados com 4 variáveis numéricas e duas classes possíveis, A e B. Classes A e B possuem 26 e 14 amostras respectivamente. 65

5.1 Métricas de desempenho calculadas para os modelos treinados com a base de dados **Breast Cancer Coimbra** sem balanceamento e balanceada de acordo com os algoritmos. 80
5.2 Testes pareado de Wilcoxon realizados sobre as métricas de desempenho obtidas pelos modelos. 80
5.3 Métricas de desempenho calculadas para os modelos treinados com a base de dados Arroz sem balanceamento e balanceada de acordo com os algoritmos. 82
5.4 Testes pareado de Wilcoxon realizados sobre as métricas de desempenho obtidas pelos modelos. 82
5.5 Métricas de desempenho calculadas para os modelos treinados com a base de dados de chocolate orgânico sem balanceamento e balanceada de acordo com os algoritmos. 84
5.6 Testes pareado de Wilcoxon realizados sobre as métricas de desempenho obtidas pelos modelos. 84
5.7 Métricas de desempenho calculadas para os modelos treinados com a base de dados de folhas de laranjeiras orgânicas sem balanceamento e balanceada de acordo com os algoritmos. 86
5.8 Testes pareado de Wilcoxon realizados sobre as métricas de desempenho obtidas pelos modelos. 86
5.9 Métricas de desempenho calculadas para os modelos treinados com a base de dados de suco de uva orgânico sem balanceamento e balanceada de acordo com os algoritmos. 88

5.10 Testes pareado de Wilcoxon realizados sobre as métricas de desempenho obtidas pelos modelos. 88
5.11 Métricas de desempenho calculadas para os modelos treinados com a base de dados **Breast Cancer Wisconsin Original** sem balanceamento e balanceada de acordo com os algoritmos. 91
5.12 Testes pareado de Wilcoxon realizados sobre as métricas de desempenho obtidas pelos modelos. 91
5.13 Métricas de desempenho calculadas para os modelos treinados com a base de dados **Glass Identification** sem balanceamento e balanceada de acordo com os algoritmos. 94
5.14 Testes pareado de Wilcoxon realizados sobre as métricas de desempenho obtidas pelos modelos. 94
5.15 Métricas de desempenho calculadas para os modelos treinados com a base de dados **Wine Quality** sem balanceamento e balanceada de acordo com os algoritmos. 96
5.16 Testes pareado de Wilcoxon realizados sobre as métricas de desempenho obtidas pelos modelos. 96
5.17 Métricas de desempenho calculadas para os modelos treinados com a base de dados **Yeast** sem balanceamento e balanceada de acordo com os algoritmos. 99
5.18 Testes pareado de Wilcoxon realizados sobre as métricas de desempenho obtidas pelos modelos. 99
5.19 Métricas de desempenho calculadas para os modelos treinados com a base de dados **Pima Indians Diabetes** sem balanceamento e balanceada de acordo com os algoritmos. 101
5.20 Testes pareado de Wilcoxon realizados sobre as métricas de desempenho obtidas pelos modelos. 101
5.21 Métricas de desempenho calculadas para os modelos treinados com a base de dados **Ecoli** sem balanceamento e balanceada de acordo com os algoritmos. 103
5.22 Testes pareado de Wilcoxon realizados sobre as métricas de desempenho obtidas pelos modelos. 103
5.23 Métricas de desempenho calculadas para os modelos treinados com a base de dados **New Thyroid** sem balanceamento e balanceada de acordo com os algoritmos. 105
5.24 Testes pareado de Wilcoxon realizados sobre as métricas de desempenho obtidas pelos modelos. 105
5.25 Melhores algoritmos de balanceamento estipulados de acordo com os modelos de classificação avaliados para cada base de dados de teste. 107

CAPÍTULO 1

# Apresentação

Processos de análise e classificação de dados estão no auge do interesse de pesquisadores do mundo inteiro, sem previsão de desfavorecimento. A evolução acentuada dos mecanismos de geração, processamento e armazenamento de dados que se deu nas últimas duas décadas somada à conscientização mundial a respeito do valor dos dados e descoberta de informação através destes instaurou a necessidade de que novos meios de análise e classificação de dados naturalmente surgissem para serem capazes de lidar com os dados da nova era, caracterizados não apenas pela sua quantidade gigante, como também complexidade, multidisciplinaridade e heterogeneidade. A literatura acadêmica recente têm sido inundada por novos métodos de análise, ciência e mineração de dados, utilizados não apenas em pesquisas de diversas áreas do conhecimento como também em ambientes coorporativos para diversos objetivos.

Para que uma análise sobre uma determinada base de dados seja de fato eficiente e renda resultados confiáveis, a base em questão precisa atender determindos requisitos que influenciam a sua qualidade. A quantidade de valores nulos e amostras anômalas por exemplo deve ser mínima possível, nula em cenários ideais. Variáveis com diferentes unidades de medida precisam ser padronizadas ou normalizadas, de maneira a evitar que variáveis com faixas de valores muito grandes dominem a análise e se sobreponham àquelas com faixas menores. O nome que se dá ao processo fundamental de preparação de bases de dados para torná-las mais limpas, representativas e de qualidade se chama pré-processamento de dados [95, 64, 37].

Parte do processo de pré-processamento de dados, balanceamento de dados refere-se ao processo de equiparar, para um determinado conjunto de dados, a quantidade de amostras de cada classe disponível para análise, tornando-as igualmente representadas diante do processo de aprendizagem. A importância deste processo jaz no fato de que diversos modelos de classificação utilizados frequentemente em projetos coorporativos e acadêmicos e que apresentam bons desempenhos de predição são projetados para trabalhar com conjuntos de dados balanceados. Entretanto, ao se trabalhar com conjuntos de dados e aplicações reais, o mais provável é que os dados disponíveis para análise estejam desbalanceados. Diversos estudos publicados na literatura mostram que conjuntos de da-

dos desbalanceados são inconvenientes e apresentam diversos desafios para a mineração e análise multivariada de dados [43, 48, 72, 56, 92, 15], e os motivos para a existência de desbalanceamento em dados são diversos, como será abordado no decorrer desta tese. Existem diversas estratégias relatadas na literatura recente para lidar com o problema do desbalanceamento de dados.

Neste trabalho, propõe-se uma nova abordagem para balanceamento de dados, baseada em transformação de dados combinada com *resampling* de dados transformados. Uma implementação inicial baseada em análise de componentes principais (PCA) e SMOTE é apresentada, e os resultados satisfatórios obtidos são detalhados.

Nos Capítulos 2 e 3, apresenta-se toda fundamentação teórica que embasa este trabalho. No Capítulo 3 estão detalhados conceitos básicos de mineração e classificação de dados e descoberta automática de conhecimento através de aprendizagem supervisionada. Descreve-se também os aspectos teóricos dos modelos e funções de classificação que foram utilizados no decorrer desta pesquisa, bem como as métricas de desempenho que foram utilizadas para avaliar os modelos. No Capítulo 2 é introduzido o conceito de balanceamento de dados. A visão geral por trás dos problemas causados pelo desbalanceamento de dados é discutida, e os métodos práticos mais conhecidos na literatura para lidar com o desbalanceamento (como métodos de *resampling* e aprendizagem *cost-sensitive*) são listados e descritos.

No Capítulo 4 a abordagem proposta para balanceamento de dados é apresentada. Discute-se sobre o objetivo da pesquisa, aspectos teóricos sobre *oversampling* de dados e como eles são afetados por alguns fatores inerentes a conjuntos de dados desbalanceados, aspectos teóricos sobre transformação de dados e a proposta deste processo como possível aprimorador de desempenho de algoritmos de *resampling*. O método análise de componentes principais (PCA) é descrito, e finalmente apresenta-se uma implementação prática da abordagem proposta, o algoritmo PCA-SMOTE.

No Capítulo 5 estão listados todos os resultados obtidos pelo PCA-SMOTE nos vários cenários e casos de teste investigados. Discorre-se também sobre a metodologia de teste empregada para atestar a eficiência do algoritmo, as bases de dados utilizadas para teste, e outros detalhes.

Finalmente, no Capítulo 6, faz-se as considerações finais do trabalho aqui relatado, destacando os bons resultados obtidos pela metodologia de balanceamento proposta, limitações encontradas e diretrizes para possíveis trabalhos futuros.

CAPÍTULO 2

# Balanceamento de dados

## 2.1 Visão geral

Diversos modelos de classificação utilizados frequentemente em projetos coorporativos e acadêmicos e que apresentam bons desempenhos de predição são projetados para trabalhar com conjuntos de dados balanceados. Dizemos que um conjunto de dados é balanceado quando a quantidade de amostras para todas as classes possíveis é igual ou diferente em apenas uma pequena porcentagem, de maneira que todas as classes estejam igualmente representadas por suas distribuições. Na Figura 2.1, podemos ver à esquerda um exemplo de conjunto de dados perfeitamente balanceado: tal conjunto é composto de oito amostras, sendo quatro delas rotuladas com a classe 'A' e as outras quatro rotuladas com a classe 'B', e portanto ambas as classes estão representadas pela mesma quantidade de amostras. Em contrapartida, o conjunto de dados ilustrado à direita possui apenas duas amostras da classe 'B' para quatro amostras da classe 'A', tecendo uma proporção 2:1 de A para B que evidencia o desbalanceamento de classes.

| Var1 | Var2 | Var3 | Var4 | Classe |
|---|---|---|---|---|
| 8.05 | 1 | 15.15 | 97.85 | A |
| 12.12 | 1 | 30.21 | 96.84 | A |
| 36.1 | 2 | 32.68 | 95.14 | A |
| 24.89 | 1 | 18.13 | 98.27 | A |
| 9.36 | 3 | 12.79 | 97.39 | B |
| 15.42 | 2 | 31.43 | 99.1 | B |
| 11.05 | 3 | 39.24 | 96.28 | B |
| 10.74 | 2 | 10.54 | 95.99 | B |

| Var1 | Var2 | Var3 | Var4 | Classe |
|---|---|---|---|---|
| 8.05 | 1 | 15.15 | 97.85 | A |
| 12.12 | 1 | 30.21 | 96.84 | A |
| 36.1 | 2 | 32.68 | 95.14 | A |
| 24.89 | 1 | 18.13 | 98.27 | A |
| 9.36 | 3 | 12.79 | 97.39 | B |
| 15.42 | 2 | 31.43 | 99.1 | B |

**Figura 2.1:** *Exemplos de conjunto de dados balanceado esquerda e outro conjunto de dados desbalanceado direita.*

Embora modelos e funções de classificação em geral sejam concebidos sobre a presunção de que serão aplicados em conjuntos de dados com distribuições de classes equilibradas, quando lidamos com problemas reais o mais provável é que os dados disponíveis para análise estejam desbalanceados. Existem vários motivos que levam a este desbalanceamento. Em diversas ocasiões comuns, a obtenção de dados de uma determinada classe pode:

- estar condicionada a um evento de ocorrência rara, de maneira que amostras de uma determinada classe são muito mais frequentes do que para outras classes;
- ser bastante cara do ponto de vista econômico, computacional e/ou de tempo, requerendo recursos que podem não estar facilmente disponíveis.

Como exemplo, considere os conjuntos de dados *Wine Quality* disponibilizados por [20] na *UCI Machine Learning Repository*. Os dois conjuntos de dados disponíveis consistem em amostras de variações tinto e branco do vinho português conhecido como Vinho Verde. Considere, especificamente, o conjunto de dados referente apenas às amostras de vinho tinto (*winequality-red.csv*). Ele é composto de 1599 amostras de vinho tinto, descritas por um total de 12 variáveis referentes aos aspectos fisicoquímicos das amostras analisadas, tais como acidez, pH, densidade, teor alcoólico, quantidade de sulfatos e outros. A variável de saída, i.e o rótulo de classe, é um valor inteiro que representa a qualidade da amostra de vinho com base em análises sensoriais conduzidas por testadores humanos. Este valor vai de 0 a 10, de maneira que o valor 0 reflete uma avaliação extremamente negativa e, de maneira inversa, o valor 10 reflete uma avaliação totalmente positiva. Desta forma, um classificador desenvolvido sobre estes dados poderia tentar predizer qual nota de qualidade seria possivelmente atribuída a uma determinada amostra de Vinho Verde tinto com base em suas características fisicoquímicas. Entretanto, observando todas as notas possíveis, aproximadamente 82% das amostras do conjunto de dados possuem valores 5 ou 6 para qualidade (681 e 638 amostras, respectivamente). Dez amostras receberam indicador 3, 53 amostras receberam indicador 4, 199 amostras receberam indicador 7 e as 18 amostras restantes receberam indicador 8. Este é um problema evidente de desbalanceamento de dado, uma vez que dois rótulos de classe possuem disparadamente muito mais amostras para representá-los do que os demais 8 rótulos possíveis.

Neste exemplo, vemos um tipo de desbalanceamento relativo: as amostras da classe minoritária (i.e amostras de vinho que receberam notas plenamente baixas ou altas) não são necessariamente raras de serem obtidas, mas são raras de serem obtidas em relação às amostras que receberam notas médias [48]. Assim, espera-se que, conforme mais amostras de vinho são analisadas, mais amostras com notas plenamente altas ou baixas (assim como amostras com com notas médias) surjam e sejam incluídas no conjunto de dados. Outro exemplo de desbalanceamento relativo pode ser observado ao coletarmos dados sobre rebanhos infectados por uma determinada bactéria. Aqui, a probabilidade de termos amostras de animais não infectados é muito maior do que aqueles infectados, não porque a ocorrência de infecções em animais seja um evento raro na natureza, mas sim porque a sua ocorrência é rara em relação à ocorrência de animais saudáveis.

Desbalanceamento de dados ocasionado devido à raridade das amostras pode ser verificado, por exemplo, na análise de dados hipotéticos de acidentes aéreos. Para

isso, registra-se os vôos realizados por uma determinada empresa de transporte aéreo nos últimos anos. Percebe-se que a quantidade de vôos que sofreram um acidente é irrisoriamente menor do que a quantidade de vôos que foram realizados com sucesso. Da mesma forma, desenvolver modelos capazes de diferenciar imagens de planetas de imagens de satélites para uma determinada galáxia requer que várias imagens de corpos celestes desta galáxia estejam disponíveis para o treinamento do classificador.

Em tais cenários onde dados são difíceis e/ou custosos de serem obtidos, e até mesmo em alguns casos de problemas cotidianos simples, depara-se frequentemente com a necessidade de trabalhar com conjuntos de dados onde amostras de uma determinada classe são bem mais ou menos numerosas do que das demais classes disponíveis. Existem diversas estratégias relatadas na literatura recente para lidar com o problema do desbalanceamento de dados. No decorrer deste capítulo, abordaremos duas categorias: métodos de amostragem (*sampling*) e algoritmos de aprendizagem sensível a custo (*cost-sensitive*).

Além disso, durante o programa de doutorado, a autora desta tese realizou e escreveu duas revisões da literatura relacionadas à aplicação de métodos de análise multivariada de dados e aprendizagem de máquina para a discriminação geral do arroz (aceito para publicação na revista *Critical Reviews in Food Science and Nutrition* com previsão para 2019 [73]) e do mel (publicado na revista *Computers and Electronics in Agriculture* em 2019 [77]). Durante o desenvolvimento de ambas revisões, foi constatado que muitos trabalhos que empregam técnicas de análise multivariada de dados e aprendizagem de máquina para a classificação de alimentos ignoram o desbalanceamento de dados e até mesmo utilizam conjuntos de dados muito pequenos, trazendo dúvidas a respeito da confiabilidade dos resultados publicados. Este fato serviu como incentivo para a busca por um método de balanceamento voltado para dados quimiométricos de alimentos cuja ideia e estrutura fossem simples e ao mesmo tempo fácil de empregar por qualquer pesquisador não especialista em mineração de dados, sendo ao mesmo tempo eficiente para o balanceamento.

## 2.2 Problemas causados pelo desbalanceamento de dados

Embora em um cenário ideal de análise seja esperado que as classes presentes em um conjunto de dados estejam similarmente representadas, ao trabalharmos com conjuntos de dados e aplicações reais o mais provável é que os dados disponíveis para análise estejam desbalanceados. Diversos estudos publicados na literatura mostram que conjuntos de dados desbalanceados são inconvenientes e apresentam diversos desafios para a mineração e análise multivariada de dados [43, 48, 72, 56, 92, 15].

### 2.2.1 Degradação de desempenho

O primeiro (e principal) problema enfrentado pela aprendizagem sobre dados desbalanceados trata-se do viés que modelos de classificação costumam apresentar neste cenário em direção aos dados da classe majoritária (aquela que possui a maior quantidade de amostras). Quando treinados por conjuntos de dados desbalanceados, os modelos de classificação mais populares na literatura tenderão a expressar um bom desempenho de classificação para as amostras da classe majoritária e um desempenho menor para amostras da classe minoritária. Diversos estudos na literatura mostram que esta queda de desempenho ocorre não necessariamente devido à diferença na representação das classes em si, mas devido a outros fatores inerentes ao desbalanceamento, como a presença de pequenos disjuntos (*small disjuncts*), baixa densidade de dados, sobreposição de dados e outros [48, 72].

A baixa densidade de dados comumente leva modelos a não detectar padrões raros nos dados. Quando há poucos dados através dos quais aprender, um algoritmo preditivo pode ter dificuldades para extrair e aprender padrões importantes que discriminem as classes do problema e estabelecer um limite de decisão confiável, o que interfere diretamente no desempenho do modelo para realizar generalizações futuras sobre amostras de dados desconhecidas. Em muitos problemas desbalanceados, a classe minoritária sofre de baixa densidade de dados e este problema é agravado na presença de desbalanceamento *within-class*, isto é, quando a classe minoritária possui outros subconceitos (ou subclasses) ocultos e estes subconceitos não possuem amostras suficientes para descrevê-los. A baixa densidade de dados também afeta a capacidade de muitos modelos de classificação de lidar com ruídos. Em situações em que a quantidade de dados de uma determinada classe é muito pequena, alguns algoritmos podem considerá-las como ruído ou *outliers*. A Figura 2.2 ilustra o cenário da baixa densidade de dados investigado por [72] no conjunto de dados *yeast4*. Ao particionar os dados e obter um subconjunto de treinamento com apenas 10% dos dados originais, quase não se constata a presença de amostras da classe minoritária, dificultando o aprendizado do algoritmo.

A existência de subconceitos em classes está diretamente ligada à presença de pequenos disjuntos. Pequenos disjuntos tratam-se de pequenos agrupamentos de amostras de uma determinada classe, normalmente em regiões dominadas por amostras de outra classe, como ilustrado na Figura 2.3. [48] descreve o problema em torno dos pequenos disjuntos da seguinte forma: um classificador tentará aprender um conceito existente nos dados através da criação de várias regras que descrevem o conceito; quando conceitos são homogêneos, o classificador consegue uni-los e criar regras que classificam corretamente uma grande porção de amostras pertencentes ao conceito principal formado; em contrapartida, conceitos heterogêneos levam à criação de pequenos disjuntos, que são regras específicas que cobrem apenas uma pequena porção de amostras pertencentes ao conceito

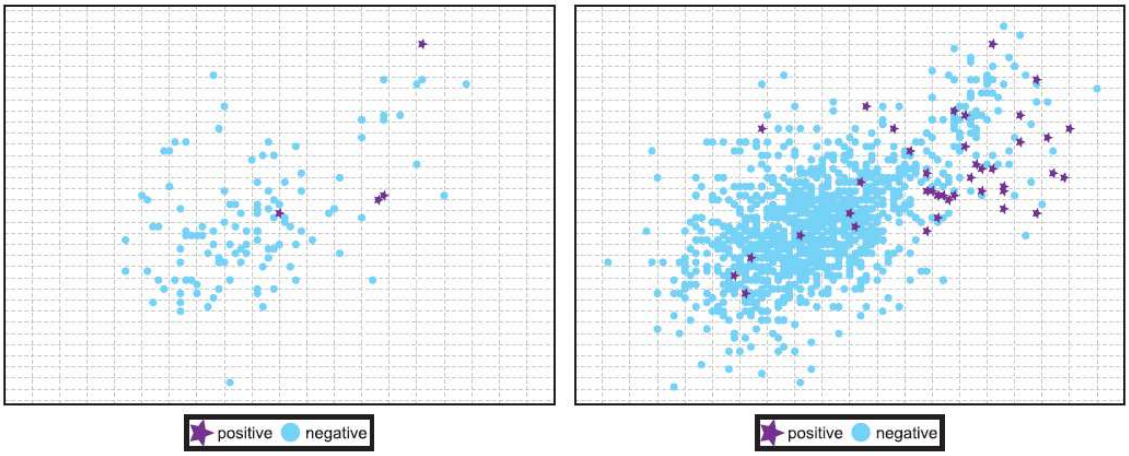



**Figura 2.2:** *Baixa densidade de dados no conjunto de dados (yeast4) [72].*

principal. Por cobrirem apenas uma pequena quantidade de amostras isoladas, estas regras específicas costumam ser maiores e mais complexas, o que as torna mais suscetíveis a erros do que regras de grandes disjuntos [56]. A presença de pequenos disjuntos aumenta a complexidade do conjunto de dados, uma vez que eles podem ser interpretados tanto como subconceitos mal representados da classe ou como ruídos, dependendo do algoritmo de aprendizado utilizado. Da mesma forma, por serem formados por uma quantidade menor de amostras, pequenos disjuntos,subconceitos e classes minoritárias em geral tendem a sofrer mais impacto de amostras ruidosas do que os demais casos.

**Figura 2.3:** *Exemplo de pequenos disjuntos em um conjunto de dados.*

A sobreposição de dados é outra característica presente em conjuntos de dados desbalanceados e que por si só está fortemente ligada à queda no desempenho de modelos de classificação. Como iustrado na Figura 2.4, ela ocorre quando exemplos de classes diferentes estão ”misturados” em determinadas regiões do espaço dimensional,

dificultando o estabelecimento de um limite de decisão confiável que separe as classes de maneira eficiente. Estudos mostram que o desempenho de classificadores é mais prejudicado pela sobreposição de dados do que pelo desbalanceamento de classes em si, como observado no experimento conduzido por [72], onde a árvore de decisão treinada foi incapaz de diferenciar de maneira satisfatória até mesmo amostras da classe majoritária.

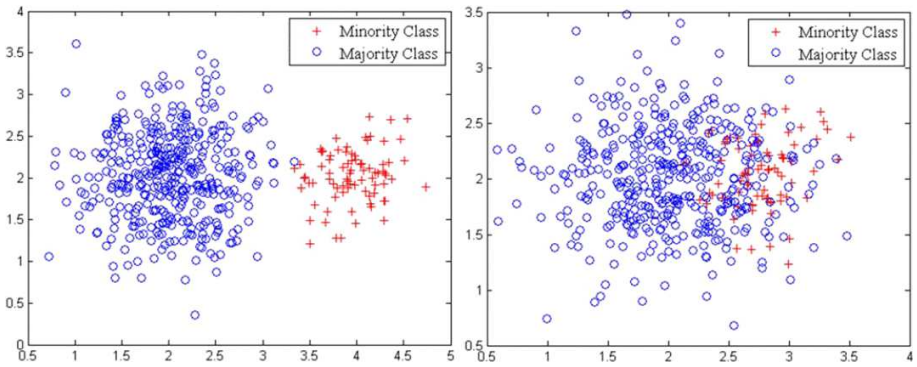



**Figura 2.4:** *Conjunto de dados sem sobreposição de dados (à esquerda) e com sobreposição de dados (à direita) [23].*

### 2.2.2 Métricas de desempenho adequadas

Outro problema enfrentado ao lidar com conjuntos de dados balanceados trata-se da seleção de métricas de desempenho adequadas para verificar o desempenho de classificadores. Métricas de desempenho baseadas em verdadeiros e falsos positivos e negativos, como acurácia, não são apropriadas para avaliar modelos de classificação desenvolvidos sobre dados desbalanceados, uma vez que grandes números de verdadeiros positivos tenderão a acobertar grandes números de falsos positivos e vice-versa, de maneira que um classificador que rotula muito bem amostras da classe majoritária e rotula de maneira mediana amostras da classe minoritária ainda apresentará um bom valor de acurácia.

Como exemplo deste cenário, considere um conjunto de dados sobre rebanhos infectados de tamanho 100 e cujo fator de debalanceamento seja 3:1. Após o treinamento de um modelo de classificação, constata-se que 95% dos animais não infectados pela bactéria (62 de 66 amostras) foram classificados corretamente como não portadores, mas apenas 30% dos animais infectados (9 de 34 amostras) foram classificados corretamente. Isso significa que 71 dos 100 casos investigados no conjunto de dados são corretamente classificados, gerando uma acurácia final para o modelo de 71% que, superficialmente, indica que ele tem um bom desempenho. Entretanto, este mesmo modelo possui aproximadamente 74% de chance de classificar erroneamente animais infectados como saudáveis, podendo esta decisão gerar uma série de consequências catastróficas para um empreendimento agropecuário, como perda de animais que não receberam o devido tratamento.

Modelos de classificação desenvolvidos sobre dados desbalanceados requerem métricas de desempenho diferentes dos métodos tradicionais, que estejam mais atentas às classes minoritárias.

Uma abordagem conhecida e recomendada para estes cenários é o uso de gráficos ROC, que expressam a relação entre sensitividade e falsos positivos obtidos por um classificador quando vários (*thresholds*) para a sua função de decisão são investigados. Através das curvas projetadas neste gráfico, é possível ver para um determinado modelo de classificação quais são os custos-benefícios entre verdadeiros e falsos positivos obtidos, facilitando enxergar a habilidade do classificador de classificar corretamente amostras de uma classe em relação às outras [31]. Como curvas ROC são construídas a partir de valores de verdadeiros e falsos positivos, elas são imunes a mudanças feitas na distribuição das classes. O cálculo da área abaixo da curva (AUC), feito pela equação 2-1 [72], gera um valor escalar no intervalo entre 0 e 1 que se refere à probabilidade do classificador de classificar uma amostra positiva melhor do que uma amostra negativa, e este valor pode ser utilizado como medida de desempenho de classificação.

$$AUC = \frac{1 + VP_{rate} - FP_{rate}}{2} \tag{2-1}$$

Quanto maior o valor de AUC computado para um classificador, melhor é considerado o seu desempenho. Em geral, assume-se que um classificador possui desempenho aceitável quando o seu valor de AUC produzido é superior a 0.5, considerando que o valor 0.5 reflete um classificador que possui desempenho semelhante a um classificador aleatório. A Figura 2.5 ilustra um exemplo de gráfico ROC para dois modelos de classificação, onde a diferença das áreas abaixo das curvas revela que o modelo B possui desempenho superior ao modelo A ($AUC_B > AUC_B$).

Outra métrica comumente utilizada é o cálculo da média geométrica (G-Mean) da sensitividade e especificidade obtidos pelo classificador. Essa métrica leva em consideração tanto os valores de verdadeiros positivos quanto de negativos obtidos, correlacionando o desempenho do classificador à sua habilidade de classificar corretamente tanto amostras da classe positiva quanto negativa. A fórmula para o cálculo do G-Mean é dada na equação 2-2.

$$GMean = \sqrt{\frac{VP}{VP+FN} \cdot \frac{VN}{FP+VN}} \tag{2-2}$$

Finalmente, *F-measure* (ou F1) também é uma métrica de desempenho eficiente para modelos de classificação construídos sobre dados desbalanceados. O valor de F1 representa a média harmônica entre o valor de precisão e o valor de sensitividade computados para o modelo, de acordo com a equação 2-3:

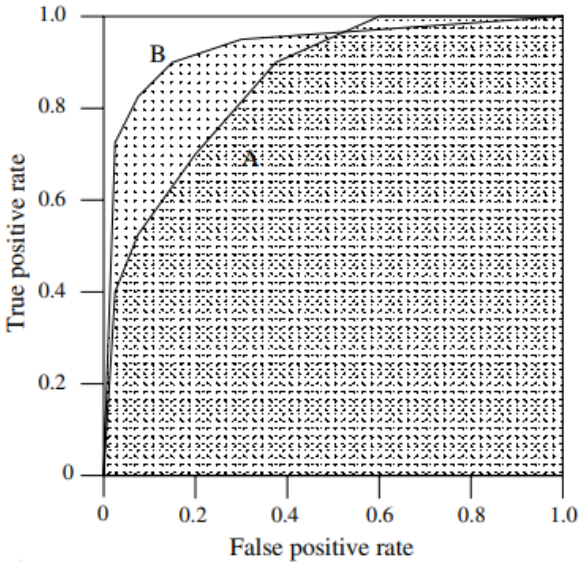



**Figura 2.5:** *Gráfico ROC contendo duas curvas referentes aos verdadeiros e falsos positivos obtidos por dois modelos de classificação A e B. [31].*

$$FMeasure = 2 \times \frac{precision \times sensitividade}{precision + sensitividade} \tag{2-3}$$

O valor de precisão, dado pela equação 2-4, determina, para todas as amostras que o classificador classificou como pertencendo à classe $Y_k$, quantas amostras realmente pertencem a esta classe. Algumas fontes se referem a estas amostras como ”amostras relevantes”. Quanto maior o valor de precisão obtido pelo modelo, menor é a taxa de falsos positivos gerados por ele [119]. Desta forma, a métrica F-1 tende a avaliar melhor os modelos de classificação que apresentam bons valores de precisão e sensitividade, isto é, não apenas tendem a classificar corretamente as amostras de uma determinada classe, como também faz poucos erros para as demais classes.

$$Precision = \frac{VP}{VP + FP} \tag{2-4}$$

## 2.3 Métodos de sampling

Métodos baseados em amostragem (*sampling*, ou *resampling*) são os métodos mais simples e mais antigos utilizados para o balanceamento de conjuntos de dados. Eles consistem na modificação da estrutura do conjunto de dados desbalanceado, de maneira a deixá-lo com quantidades equivalentes de amostras para as classes presentes, seja através da remoção (*undersampling*) ou adição (*oversampling*) de novas amostras. Estes tipos de métodos também são conhecidos como abordagens a nível de dados (*data level approach*) uma vez que toda a solução para o balanceamento dos dados é efetuada diretamente

sobre o conjunto de dados, isolando os modelos de classificação, que apenas recebem os dados já balanceados para treinamento. Como este processo normalmente ocorre antes que modelos de classificação sejam treinados, métodos de *sampling* estão relacionados à etapa de pré-processamento de dados.

Eles são populares e amplamente utilizados devido ao desempenho melhor que classificadores treinados com conjuntos de dados balanceados apresentam em relação a modelos treinados com dados desbalanceados, como discorrido na seção anterior. Em seu estudo, [110] concluíram que métodos simples de *undersampling*, como *random undersampling* e ENN (*Edited Nearest Neighbors* proposto por [126]) apresentaram desempenhos melhores do que outras técnicas de amostragem para todos os níveis de ruído e desbalanceamento investigados no conjunto de dados de teste, mesmo quando comparados ao SMOTE e suas variações. Além disso, os autores mostraram que os modelos de classificação mais robustos e mais capazes de lidar com dados desbalanceados e ruidosos sem perder muito desempenho foram SVMs e classificadores bayesianos.

Alguns dos métodos de *sampling* mais utilizados na literatura são descritos a seguir.

### 2.3.1 Random sampling

*Randomsampling*, ou amostragem aleatória, refere-se ao processo de eliminar o desbalanceamento de um conjunto de dados através da remoção aleatória de amostras da classe majoritária (*random undersampling*) ou replicação de amostras da classe minoritária (*random oversampling*). Embora ambos métodos sejam intuitivos e simples de serem executados, eles trazem consigo algumas desvantagens. A remoção de amostras do conjunto de dados acarreta em perda de dados e pode potencialmente privar o modelo de classificação de informações importantes a respeito da classe majoritária, prejudicando a sua capacidade de discriminá-las. Já a adição de amostras replicadas da classe minoritária pode levar o modelo de classificação a criar *overfitting*, prejudicando o seu desempenho para discriminar novas amostras, uma vez que o classificador possivelmente criará diversas regras de classificação para uma mesma amostra, tornando-a muito específica [48]. Contudo, quando o conjunto de dados dispõe de muitas amostras da classe minoritária, realizar o *undersampling* das amostras da classe majoritária ainda é uma escolha melhor do que aplicar *oversampling* nas amostras da classe minoritária devido ao tempo computacional [43] e ao desempenho superior dos algoritmos de *undersampling* em relação aos demais algoritmos de amostragem discutido por [110].

### 2.3.2 Informed sampling

Algoritmos de *informed sampling*, ou amostragem assistida, surgiram para amenizar a perda de informações causada pela remoção aleatória de amostras. Um critério de amostragem é utilizado, geralmente visando preservar no conjunto de dados as amostras mais informativas para a discriminação das classes. Um exemplo destes algoritmos é o *EasyEnsemble*, ilustrado na Figura 2.6. Este algoritmo faz com que as amostras da classe majoritária sejam divididas em vários subconjuntos independentes, preferencialmente com tamanho equivalente ao número de amostras da classe minoritária. Cada subconjunto é somado ao conjunto de amostras da classe minoritária e então o conjunto resultante, contido no conjunto de dados original, é utilizado para treinar um modelo de classificação. Desta forma, vários modelos são treinados de acordo com a quantidade de subconjuntos disponíveis.

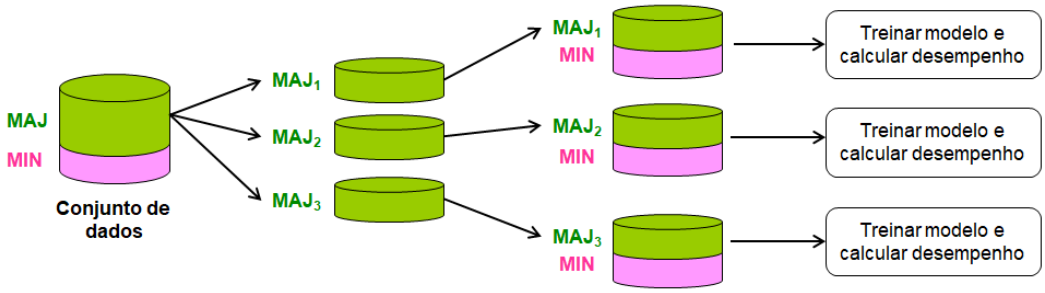



**Figura 2.6:** *Funcionamento do algoritmo EasyEnsemble. MAJ e MIN referem-se aos conjunto de dados das classes majoritária e minoritária, respectivamente. $MAJ_1$, $MAJ_2$ e MAJ3 são partições independentes de MAJ.*

Já o *BalanceCascade* é um algoritmo para a remoção de amostras da classe minoritária que são escolhidas por um conjunto de classificadores [70]. Outros estudos sugerem o uso do modelo dos vizinhos mais próximos (KNN) para computar e descartar as amostras da classe majoritária que estejam a uma distância média maior de amostras da classe minoritária, ou o uso de algoritmos de agrupamento como o K-means para dividir os dados da classe majoritária em grupos, mantendo apenas as amostras dos centros dos grupos (representativas) enquanto as demais são descartadas.

### 2.3.3 Synthetic Minority Oversampling Technique (SMOTE)

O algoritmo SMOTE surgiu em 2002, proposto por [16], e é atualmente o algoritmo de amostragem mais famoso e utilizado para balancear conjuntos de dados previamente à construção de modelos de classificação. A versão básica do SMOTE trata-se de uma técnica de *oversampling* que, ao invés de simplesmente replicar amostras do conjunto minoritário original, gera amostras sintéticas baseado nas similaridades entre

amostras no espaço n-dimensional de variáveis. Estas amostras sintéticas possibilitam aos classificadores identificar regiões de decisão maiores e menos específicas, ao invés de regiões menores e mais específicas criadas tipicamente pela simples replicação de amostras [15]. Para a sintetização das novas amostras, o algoritmo KNN é utilizado para selecionar as $k$ amostras da classe minoritária mais próximas de uma amostra também da classe minoritária, e amostras sintéticas são criadas e posicionadas no segmento entre elas.

Detalhadamente, o algoritmo SMOTE compreende os seguintes passos:

1. Escolha uma amostra da classe minoritária $s_i$;
2. Aplique o algoritmo dos $k$ vizinhos mais próximos (KNN) para identificar as $k$ amostras da classe minoritária cuja distância euclidiana em relação a $x_i$ sejam as menores;
3. Escolha, de maneira aleatória, uma das amostras retornadas pelo passo anterior e chame-a de $s_h$;
4. Calcule uma amostra sintética $s'_k$ a partir da equação 2-5. A amostra sintética é gerada a partir da soma dos valores de $s_i$ com os valores de $s_h$ multiplicados por um número $\sigma$ no intervalo $[0,1]$;
5. Adicione $s'_k$ no conjunto de dados;
6. Repita os passos anteriores até que todas as amostras originais da classe minoritária do conjunto de dados tenham gerado uma amostra sintética.

$$s'_k = s_i + s_h \times \sigma \mid \sigma \in [0,1] \tag{2-5}$$

A Figura 2.7 ilustra a execução de uma iteração do algoritmo SMOTE. Em (a), vemos o início do algoritmo com as amostras da classe minoritária e majoritária representadas por pontos verdes e azuis, respectivamente. Em (b), uma amostra da classe minoritária é selecionada (ponto preto) e os seus 3 vizinhos mais próximos (pontos amarelos) são identificados de acordo com o algoritmo KNN. Em (c), uma das amostras vizinhas é selecionada aleatoriamente (ponto marrom), e finalmente uma amostra sintética (ponto vermelho) é gerada e posicionada no segmento que conecta o ponto preto ao ponto marrom.

O sucesso do SMOTE para o balanceamento de dados tem sido comprovado em diversos trabalhos na literatura. Em seu artigo, [43] afirma que o uso do SMOTE é recomendado sobre outros métodos em cenários em que há a disponibilidade de apenas algumas dezenas de amostras da classe minoritária, ou em combinação com alguma técnica de *undersampling* quando o conjunto de dados é consideravelmente grande.

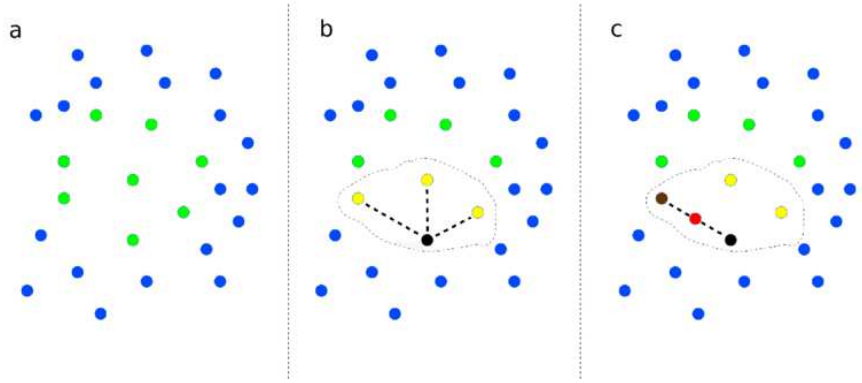



**Figura 2.7:** *Representação gráfica do algoritmo SMOTE [109].*

### 2.3.4 Algoritmos baseados em SMOTE

Até 2018, mais de 85 extensões para o SMOTE tinham sido publicadas na literatura, com a maior parte delas explorando e propondo melhorias na etapa de seleção das amostras iniciais e também na geração adaptativa das amostras sintéticas (ponderação de amostras com base na dificuldade que elas apresentam para serem aprendidas) [32]. Outras abordagens comumente envolvem a integração do SMOTE com processos de *undersampling*, definição de novos métodos para a geração das amostras sintéticas, combinação do SMOTE com métodos de redução de dimensionalidade, adição de filtros para remover amostras sintéticas ruidosas e outros. [32] traz em seu trabalho uma extensa lista de métodos propostos com base no SMOTE. Nesta subseção, citamos algumas das extensões mais conhecidas e utilizadas.

Um dos métodos mais conhecidos chama-se *Borderline-SMOTE*, uma adaptação do SMOTE proposta por [46] que parte do pressuposto de que amostras localizadas nos entornos do limite de decisão são mais informativas e mais propensas a erros de classificação do que as demais e por isso o processo de balanceamento deve ser focado sobre elas. De fato, o estudo conduzido por [85] mostrou que o desempenho de classificadores é fortemente afetado pela quantidade de amostras nas bordas do limite de decisão. Quanto melhor definidas são as bordas do limite de decisão, mais precisa é a discriminação entre as amostras das classes constantes no conjunto de dados [72]. O algoritmo *Borderline-SMOTE* é uma aplicação do SMOTE tradicional na região próxima ao limite de decisão, gerando amostras sintéticas da classe minoritária considerando segmentos apenas entre as amostras mais próximas do limite de decisão, como mostrado na Figura 2.8.

Amostras próxima ao limite de decisão também recebem ênfase no algoritmo SDSMOTE proposto por [67], que utiliza uma medida chamada de nível de suporte (*support degree*) para selecionar as amostras próximas ao limite de decisão e efetuar *oversampling* sobre elas.

Em contrapartida à afirmação base do *Borderline-SMOTE* e SDSMOTE, os au-

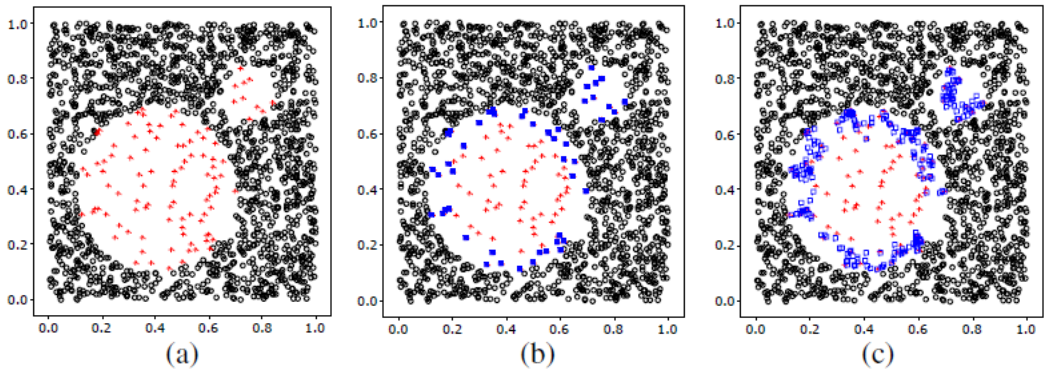



**Figura 2.8:** *Representação gráfica do algoritmo Borderline-SMOTE [46]. (a) Distribuição original do conjunto de dados Circle. (b) Amostras da classe minoritária nas bordas do limite de decisão (quadrados preenchidos). (c) Amostras sintéticas geradas pelo Borderline-SMOTE (quadrados vazios).*

tores de [104] argumentam que as amostras localizadas próximas ao limite de decisão são as mais difíceis de serem aprendidas por um modelo e que normalmente não contribuem para o desempenho da classificação, podendo serem descartadas para aprimorar o processo de classificação. Para lidar com este problema e também com o problema de amostras anômalas (ruídos), eles propõem o SMOTE-IPF, uma combinação do SMOTE tradicional com o filtro IPF (*Iterative-Partitioning Filter*) que, após a geração de amostras sintéticas da classe minoritária pelo SMOTE, remove amostras anômalas e amostras localizadas nas bordas do limite de decisão, tornando o conjunto de dados mais regular. A implementação do SMOTE-IPF consiste na aplicação do filtro IPF sobre o conjunto de dados resultante da aplicação prévia do SMOTE, o que permite que outras extensões do SMOTE também possam ser utilizadas sem alterar a proposta do algoritmo. O filtro IPF funciona de maneira iterativa até que a quantidade de ruído constatada no conjunto de dados seja inferior à uma determinada porcentagem do conjunto de dados, compreendendo os seguintes passos[104]:

1. Considerar $A = \emptyset$ o conjunto inicial de amostras ruído;
2. Dividir o conjunto de dados $E$ em $n$ subconjuntos de tamanho similar;
3. Para cada subconjunto, construir um modelo de classificação baseado em C4.5 e investigar o desempenho do modelo treinado a partir do conjunto $E$;
4. Identificar amostras ruído em $E$ e adicioná-las em $A$ de acordo com algum critério de votação;
5. Remover as amostras ruído: $E \leftarrow E \backslash A$

O algoritmo *Safe-Level-SMOTE* [12] é outra extensão do SMOTE bastante utilizada e que foca o processo de *oversampling* nas amostras que estão posicionadas

em regiões consideradas ”seguras”, isto é, dominadas majoritariamente pelas amostras da classe minoritária. O nível de segurança de uma amostra *a* é dado pela quantidade de amostras nos *k* vizinhos mais próximos a *a* que pertencem à classe minoritária, de maneira que quanto este nível é próximo a 0, a amostra *a* é praticamente considerada como ruído. Quanto mais próximo ao valor de *k* é o nível de segurança, mais segura é considerada a amostra *a*, e amostras sintéticas tenderão a ser posicionadas próximas a ela. A vantagem do *Safe-Level-SMOTE* em comparação ao *Borderline-SMOTE* é que, diferente do último, *Safe-Level-SMOTE* foca a geração de amostras em regiões confiáveis da classe minoritária e evita regiões com muitos ruídos ou com sobreposição de dados.

Atacando uma das principais vulnerabilidades do SMOTE que é a sua desatenção à distribuição dos dados, [130] propõem o *Abstention-SMOTE*, que foca o processo de *oversampling* apenas nas amostras de abstenção. Amostras de abstenção são identificadas por determinados tipos de modelos de classificação que, além de predizer a classe de uma amostra, também são capazes de se abster para algumas amostras que o modelo considera difíceis de serem rotuladas, ou fáceis de serem classificadas incorretamente. Os autores argumentam que o aumento sintético destas amostras pode melhorar o desempenho da classificação das amostras da classe minoritária como um todo. Os testes realizados compararam este método com o SMOTE tradicional e o *Borderline-SMOTE*, apresentando melhores resultados do que ambas as técnicas.

[94] propõem o *Weighted-SMOTE*, uma versão modificada do SMOTE tradicional na qual o *oversampling* de cada amostra da classe minoritária é realizado com base em valores de pesos associadas a elas. Estes valores de peso se referem à distância euclidiana da amostra em relação a todas as demais amostras da classe minoritária.

O algoritmo ROSE (*Random OverSampling Examples*) proposto por [82] é um *framework* sistemático que faz o *oversampling* das amostras da classe minoritária de acordo com uma abordagem *bootstrap* suavizada. Uma amostra da classe minoritária do conjunto de dados original é escolhida e uma amostra sintética é gerada em sua vizinhança, considerando que a amplitude desta vizinhança é dada por uma matriz de parâmetros escalares.

A ideia geral por trás do algoritmo ADASYN [47] é o uso de distribuições ponderadas para amostras de dados diferentes da classe minoritária dependendo do quão difícil é para tais amostras serem aprendidas por modelos. Desta forma, amostras que são mais difíceis de serem aprendidas contribuem mais para a geração de amostras sintéticas do que aquelas que são mais fáceis. Os autores da técnica argumentam que o uso de distribuiuções ponderadas do ADASYN melhora o processo de aprendizado das amostras ”difíceis” de duas formas: reduzindo o viés introduzido pelo desbalanceamento de dados, e alterando adaptativamente o limite de decisão dos classificadores em direção às amostras ”difíceis”.

## 2.4 Aprendizagem cost-sensitive

Enquanto métodos de *resampling* modificam estruturalmente o conjunto de dados de maneira a equilibrar a quantidade de amostras de cada classe, a ideia que embasa a aprendizagem *cost-sensitive* é assumir custos maiores para erros de classificação de amostras da classe minoritária e incorporar tais custos no processo de aprendizado dos algoritmos de classificação já conhecidos. Fontes específicas se referem a estes tipos de métodos como abordagens a nível algorítmico (*algorithmic level*). Geralmente, os custos para classificações erradas são definidos em matrizes de custo, cujos valores podem variar para cada domínio e de acordo com opiniões de especialistas. Embora diversos estudos mencionados em [48] sugiram que métodos *cost-sensitive* apresentam resultados superiores a métodos de amostragem, [43] revela em sua revisão da literatura que a grande maioria dos trabalhos publicados recentemente que lidam com o problema do desbalanceamento de dados preferiram usar métodos de *resampling* e um dos principais motivos para isso é a dificuldade de estabelecer valores para a matriz de custo que rege a aprendizagem *cost-sensitive*. Desta forma, simplesmente inserir ou remover amostras do conjunto de dados de maneira a equiparar a proporção das classes é uma estratégia mais amigável a cientistas que não são familiarizados com a aprendizagem de máquina.

A aprendizagem *cost-sensitive* gira em torno dos valores definidos na matriz de custo e o seu objetivo geral é alcançar a solução que minimiza o custo geral de treinamento. Por padrão, não há valores de custo associados a classificações corretas e classificações erradas de amostras da classe minoritária recebem mais custo do que amostras da classe majoritária, o que força os modelos de classificação a serem mais atentos para classificações erradas de amostras de classes menos representadas. A Fig. 2.9 mostra um exemplo de matriz de custo para um problema multiclasse, onde $C(y_i, y_j)$ se refere ao custo de uma classificação errada de uma amostra da classe $y_i$ como sendo da classe $y_j$.

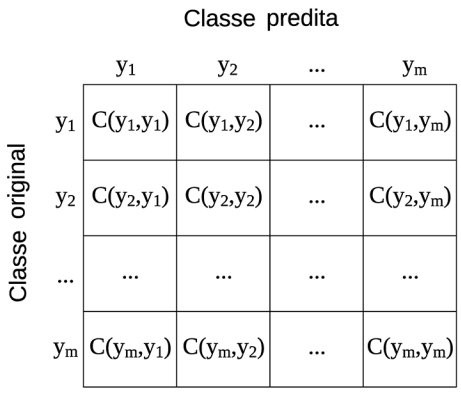



**Figura 2.9:** *Matriz de custo.*

Existem diferentes formas de implementar a aprendizagem *cost-sensitive* para

lidar com o problema do desbalanceamento de dados. Nas subseções abaixo, discorremos sobre duas abordagens comumente empregadas em trabalhos de sucesso na literatura, embora diversas outras possam ser encontradas pela literatura.

### 2.4.1 Métodos baseados em ponderação de amostras

Estratégias frequentemente utilizadas na literatura para implementar a aprendizagem *cost-sensitive* envolvem a atribuição de valores de peso às amostras do conjunto de dados de acordo com os valores da matriz de custo. Uma das maneiras mais tradicionais de realizar isto é através de *boosting*, utilizando algoritmos como o AdaBoost. Tal algoritmo assume que amostras classificadas incorretamente devem receber pesos maiores, forçando o modelo de classificação a aprendê-las melhor. Diversas extensões para o AdaBoost foram propostas na literatura recente para uso combinado com aprendizagem *cost-sensitive*, como o AdaBoost.M1. Este algoritmo opera em $T$ iterações e, em cada iteração $t$, a função de distribuição $D_i$ dos dados de treinamento é atualizada e utilizada para treinar um novo classificador, de acordo com a equação [48]:

$$D_{t+1}(i) = \frac{D_t(i)exp(-\alpha_t h_t(x_i)y_i)}{Z_t} \tag{2-6}$$

sendo $\alpha_t = \frac{1}{2}ln(\frac{1-\varepsilon_t}{\varepsilon_t})$ o parâmetro de atualização dos pesos das amostras, $h_t(x_i)$ é o valor predito pelo classificador $h_t$ para a amostra $x_i$, $\varepsilon_t = \sum_{i:h_t(x_i)\neq y_i} D_t(i)$ é o erro produzido pelo classificador $h_t$, e $Z_t$ é um fator de normalização que garante que $D_{t+1}$ será uma função de distribuição ($\sum_{i=1}^{m} D_{t+1}(i) = 1$). Os algoritmos AdaC1, AdaC2 e AdaC3 [118] baseiam-se neste cenário, e embutem seus valores de custos dentro da exponencial (AdaC1, equação 2-7), fora da exponencial (equação 2-8) e tanto dentro quanto fora da exponencial (equação 2-9), respectivamente. Em todas as três equações, $C_i$ refere-se ao valor de custo associado à classificação errada da amostra $x_i$.

$$D_{t+1}(i) = \frac{D_t(i)exp(-\alpha_t C_i h_t(x_i)y_i)}{Z_t} \tag{2-7}$$

$$D_{t+1}(i) = \frac{C_i D_t(i)exp(-\alpha_t h_t(x_i)y_i)}{Z_t} \tag{2-8}$$

$$D_{t+1}(i) = \frac{C_i D_t(i)exp(-\alpha_t C_i h_t(x_i)y_i)}{Z_t} \tag{2-9}$$

Outros algoritmos conhecidos e com abordagem similar são o AdaBoost.M2, AdaBoost.MR e AdaBoost.MH [34, 107], AdaCost [30] e outros.

### 2.4.2 Métodos baseados em modificações de modelos de classificação

O processo de aprendizado de modelos de classificação como SVM, árvore de decisão, redes neurais artificiais e outros podem ser modificados de maneira a incorporar custos para classificações erradas, assim aprimorando o desempenho da classificação sobre conjuntos de dados desbalanceados. Por exemplo, de acordo com [48], três tipos de mudanças podem ser feitas em árvores de decisão de maneira a torná-las *cost-sensitive*:

- Ajustes podem ser feitos no limite de decisão, considerando custos para erros de classificação, a distribuição das amostras de treinamento e a posição do limite de decisão;
- Custos podem ser implementados no critério de divisão das variáveis em cada nó da árvore;
- Esquemas de podas *cost-sensitive* podem ser aplicados à arvore. A poda de árvores de decisão faz a remoção de nós folhas que contenham estimativas de probabilidade de classes menor do que um certo limite, o que melhora o desempenho de classificação, em contrapartida tendendo a descartar nós folhas que contenham classes minoritárias.

Em seu artigo, os autores citam diversos trabalhos que desenvolvem soluções seguindo estas três abordagens. Além deles, outros aprimoramentos para modelos de classificação existentes que os tornam mais atentos às classes menos representadas foram publicadas na literatura. A Tabela 2.1 resume alguns exemplos de trabalhos que seguiram esta diretriz e propuseram modificações para os modelos SVM, redes neurais artificiais e KNN de maneira a torná-los *cost-sensitive*.

## 2.5 Conclusão

Neste capítulo, discorremos sobre o balanceamento de dados, a etapa de pré-processamento que visa igualar a quantidade de amostras de dados de cada classe disponível de maneira a tornar as classes igualmente representadas diante do algoritmo de aprendizado de modelos e funções de classificação. O desbalanceamento de dados pode acarretar, entre várias outras consequências, em viéses de classificação e queda de desempenho de modelos preditivos, prejudicando a eficiência e a confiabilidade de modelos. Desta forma, o balanceamento de dados, assim como a liquidação de valores nulos, de amostras ruidosas e a seleção de variáveis, são etapas fundamentais que devem ser implementadas pelo analista de dados que se preocupe com a confiabilidade de seus produtos de classificação. Os problemas, conceitos e técnicas descritos neste capítulo, em especial amostragem, *oversampling* e SMOTE, são as fundações da abordagem proposta nesta tese.

| Modelo base | Variações propostas |
|---|---|
| SVM | [131]: desenvolveram uma função de kernel escalável para o SVM que expande o espaço assimétrico do limite entre as classes de maneira a compensar o desbalanceamento dos dados;<br>[25]: propuseram uma variante chamada Near-Bayesian Support Vector machines (NBSVMs), que incorpora custos assimétricos para classificações erradas durante a construção do limite de decisão do SVM tradicional;<br>[129]: propuseram máquinas de hiper-esferas com margem máxima e volume mínimo com perda de pinball (Pin-M$^3$HM) para classificação de dados de duas classes e desbalanceados;<br>[17]: propuseram o algoritmo IBFSVM baseado vetores de suporte *fuzzy*, introduzindo fatores que amenizam ruídos e compensam o desbalanceamento das classes; |
| Redes neurais artificiais | [93]: propuseram o método CO$^2$RBFN, um modelo cooperativo e competitivo para projeto de redes de funções gaussianas posteriormente combinadas com SMOTE para pré-processamento dos dados;<br>[101]: integra diferentes métodos e funções de custo no treinamento de redes neurais convolucionais;<br>[22]: propuseram uma abordagem de pré-processamento dos dados anterior ao treinamento do modelo de classificação, descorrelacionando as variáveis ao aplicar PCA para transformá-las e em seguida aplicando ruído gaussiano aditivo em cada variável transformada; |
| K-nearest neighbors | [8]: propuseram um método de decomposição hierárquica baseado em algoritmos de agrupamento e detecção de *outliers*, sem utilizar hierarquias baseadas em variáveis e classes, que apresenta bom desempenho para dados desbalaceados mesmo quando ocorre muita sobreposição de amostras;<br>[62]: propuseram o $\alpha$-wkNN, uma modificação do algoritmo KNN ponderado que determina o grau de associação da classe através do $\alpha$-ésimo quantil da distribuição de probabilidade da classe estimada, assim mitigando o impacto do desbalanceamento dos dados;<br>[36]: propuseram um método para a escolha de números $k$ locais para diferentes regiões do espaço dimensional, aprimorando o desempenho do KNN tradicional tanto para dados balanceados como para dados desbalanceados; |

**Tabela 2.1:** *Exemplos de propostas sugeridas para os modelos de classificação SVM, redes neurais artificiais e KNN incorporando aprendizado cost-sensitive.*

CAPÍTULO 3

# Classificação de dados

Há algum tempo atrás, informações eram obtidas de conjuntos de dados através de métodos lineares e cálculos estatísticos simples, compondo um processo que ainda hoje chamamos de análise multivariada de dados. Contudo, naquela época, os conjuntos de dados eram impressionantemente menores e mais simples, e a extração de conhecimento se resumia basicamente a teste de hipóteses.

A última decada foi marcada por um avanço desenfreado de tecnologias de geração, processamento e armazenamento de dados, além da própria conscientização global em diversas áreas sobre o valor de dados e de possíveis informações valiosas ocultas neles. Conforme as bases de dados se tornaram gradativamente enormes, caras e complexas, contendo não apenas valores numéricos simples como também valores categóricos, imagens, áudios, séries temporais e outros tipos, técnicas de análise de dados e a própria ciência de dados precisaram evoluir para serem capazes de realizar estudos produtivos nestas bases e extrair informações úteis, observando questões como extração de informações relevantes, processamento de tipos de dados complexos, custo computacional, tempo de execução e outros.

A mineração de dados é um processo robusto para descoberta de informação em bases de dados grandes e complexas que surgiu através da incorporação de conceitos de inteligência artificial, otimização matemática e algoritmos de computação à já conhecida, antiga e amplamente utilizada análise multivariada de dados. Técnicas de mineração de dados são derivadas através de modelagem, otimização e/ou raciocínio probabilístico [53] e têm como objetivo a descoberta de conhecimento e padrões ocultos nos dados, ao invés dos simples testes de hipóteses utilizados em larga escala pela estatística tradicional. [119] define a mineração de dados como um processo de descoberta automática de informações úteis em grandes depósitos de dados, dispondo de técnicas para descoberta de informações em conjuntos de dados que podem passar inobservadas através de métodos de análise estatística tradicionais. A mineração de dados é uma parte integral da *descoberta de conhecimento em bancos de dados (KDD)*, que é o processo geral de conversão de dados brutos em informações úteis e consiste de uma série de etapas de transformação como as que estão dispostas na Figura 3.1 [119].

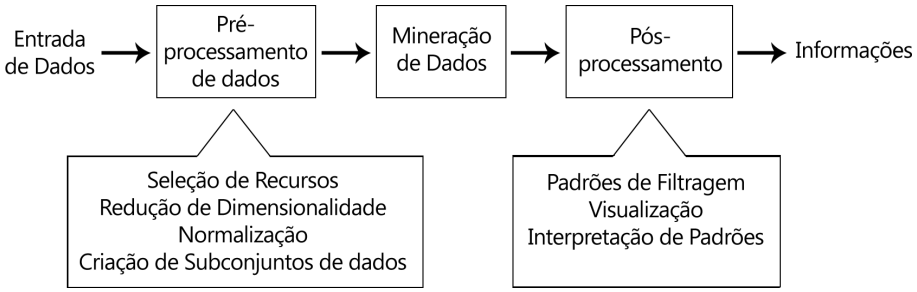



**Figura 3.1:** *Processo de descoberta de conhecimento em bancos de dados (KDD) [119].*

A aprendizagem de máquina (*machine learning*) é um procedimento fundamental para a realização da maior parte dos processos de mineração de dados. Derivada do campo de inteligência artificial, ela molda algoritmos capazes de fazer descobertas automáticas de padrões e informações em conjuntos de dados que levam a uma tomada de decisão [9]. No contexto de mineração de dados e reconhecimento de padrões, uma base (ou conjunto) de dados é definida como um conjunto finito de exemplos (ou amostras) $E = \{e_1, e_2, ..., e_n\}$ que representam combinações lineares de um determinado conjunto de variáveis $X = \{x_1, x_2, ..., x_m\}$. Em outras palavras, cada exemplo é descrito por um vetor finito de características e seus respectivos valores, que podem ser mapeados em diversos tipos, como números, cadeias de caracteres e outros. Desta forma, um conjunto de dados refere-se a uma matriz $n \times m$, sendo $n$ a quantidade de exemplos disponíveis e $m$ a quantidade de variáveis, ou características, que descrevem os exemplos. A aprendizagem de máquina nos permite então modelar algoritmos capazes de observar o comportamento dos exemplos de um determinado conjunto de dados e suas características e tentar predizer informações em exemplos novos e desconhecidos, ou mesmo identificar padrões e perfis entre os exemplos do conjunto de dados observado.

## 3.1 Classificação de dados

Atualmente, duas perspectivas possibilitam a descoberta de informações e padrões ocultos em bases de dados através de mineração: análise preditiva ou análise exploratória. Análises preditivas envolvem o uso de funções capazes de predizer determinadas informações em novos exemplos com base em dados similares previamente observados, como prognósticos médicos, clima, placares de partidas em jogos esportivos, tipos de alimentos e outros. O subprocesso da mineração de dados que oferece técnicas para a realização de predições é chamado de classificação de dados.

Considere um conjunto de dados arbitrário. Após a leitura de todos os exemplos deste conjunto e suas características, queremos ser capazes de inferir informações para novos exemplos desconhecidos que sejam descritos pelo mesmo vetor de características.

A classificação de dados implementa a *aprendizagem supervisionada*, cujo objetivo é o desenvolvimento de algoritmos capazes de generalizar dados novos e desconhecidos com base em um conjunto de dados previamente observados chamado de *conjunto de treinamento*. Os exemplos do conjunto de treinamento estão associados a uma característica especial chamada rótulo de classe (também conhecida como variável de resposta, ou variável dependente), que se refere à informação que se deseja predizer em novos exemplos. Desta forma, a aprendizagem supervisionada é utilizada para construir algoritmos capazes de predizer o rótulo de classe de novos exemplos com base nas características e nos rótulos de classe observados em outros exemplos similares.

O produto da aprendizagem supervisionada é um *classificador*, matematicamente definido como sendo uma função $\hat{f} : X \to \{true, false\}$ que utiliza o conjunto de exemplos $D$ de maneira que $\hat{f}(x) \cong f(x)$, sendo $D$ um conjunto de exemplos rotulados onde $D = \{(x,y) \mid x \in D \text{ e } y = f(x)\}$ para uma função de rotulação $f : X \to \{true, false\}$ [45]. Essencialmente, um classificador é uma função capaz de mapear um conjunto de valores de características em um rótulo de classe, e que é construído a partir da observação prévia de outros valores para as mesmas características e quais rótulos de classe estão associados a eles. A Figura 3.2 ilustra um exemplo de classificador para um conjunto de dados cujos exemplos são definidos por três variáveis: $x_1$ (Variável 1), $x_2$ (Variável 2) e um rótulo de classe $c$, referente a cor do exemplo no gráfico. Como $c$ só pode assumir dois valores (vermelho ou azul), dizemos que este é um problema de classificação binária. O classificador receberá como entrada os valores de $x_1$ e $x_2$ e retornará um valor predito para $c$. No exemplo dado, o classificador é definido pela função de classificação exposta na equação 3-1 e irá predizer a cor vermelha para exemplos desconhecidos cujo valor de $x_2$ for maior do que 5 e azul caso contrário.

$$f(x) = \begin{cases} \text{verde} & \text{se } x_2 < 5 \\ \text{vermelho} & \text{se } x_2 > 5 \end{cases} \tag{3-1}$$

A classificação de dados utilizando métodos de mineração de dados derivados de aprendizagem de máquina possui uma infinidade de aplicações de sucesso em diversas áreas. Alguns exemplos de trabalhos são:

- Detecção de mensagens de *spam* em e-mails e redes sociais [55, 39, 113, 84, 83, 114]
- Reconhecimento da origem geográfica de alimentos e bebidas [7, 1, 13, 74, 21, 77, 35, 96]
- Autenticação de alimentos orgânicos [76, 6, 5, 59, 78, 81, 122, 49, 103]
- Horticulturas [44, 50, 71, 99, 40, 121, 106]
- Ciências forenses [115, 65, 124]
- Negócios [111, 2, 90, 69, 60, 3]

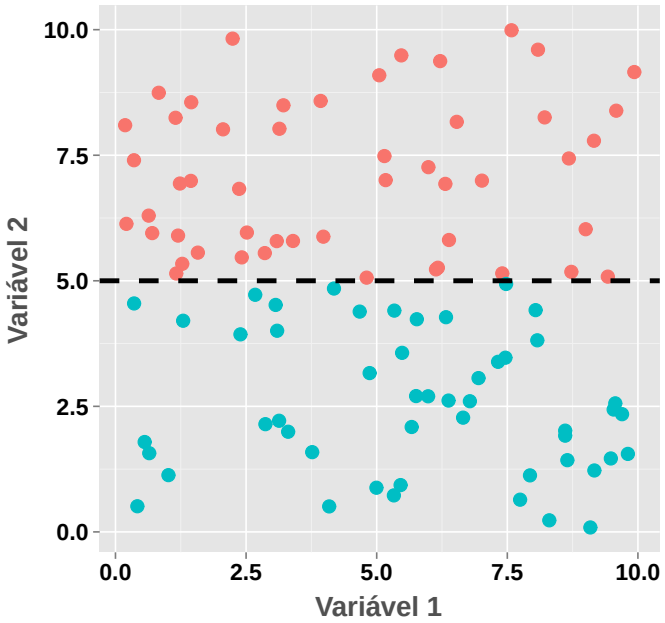



**Figura 3.2:** *Exemplo de classificador.*

- Esportes [26, 66, 41, 11]
- Classificação de textos [87, 120, 108]
- Diagnósticos médicos e psiquiátricos [88, 68, 91, 58, 29, 52, 100, 61, 128]
- E muitos outros.

## 3.2 Modelos de classificação

Nesta subseção, discorremos sobre os modelos de classificação que foram utilizados no decorrer da nossa pesquisa: máquinas vetores de suporte (SVM), redes neurais artificiais e árvores de decisão.

### 3.2.1 Máquinas vetores de suporte

Máquinas vetores de suporte (SVM) refere-se a um modelo de classificação introduzido por Corinna Cortes e Vladmir Vapnik em 1995 [19] e que tem se popularizado rapidamente na literatura sobre mineração de dados, predições de informações e aprendizado de máquina. Um dos motivos para a sua popularidade é a sua eficiência e sucesso empírico, comprovados através das altas porcentagens de predição com sucesso que o SVM tem mostrado em diversas aplicações de vários domínios.

O objetivo do SVM é encontrar um hiperplano que atuará como limite de decisão e que deverá ter a maior margem possível. Tal hiperplano é chamado de hiperplano de margem máxima. A Figura 3.3 mostra um exemplo de conjunto de dados linearmente separável e os vários hiperplanos capazes de classificar perfeitamente os dados, e o

classificador deverá escolher um destes hiperplanos para ser o limite de decisão com base em sua capacidade de separação. As margens de um hiperplano são definidas como hiperplanos paralelos ao limite de decisão que se afastam até tocar, cada um, os primeiros exemplos de cada classe. A Figura 3.4 ilustra dois limites de decisão, $L$ e $T$, sendo $m_1$ e $m_2$ margens de $L$ e $t_1$ e $t_2$ margens de $T$.

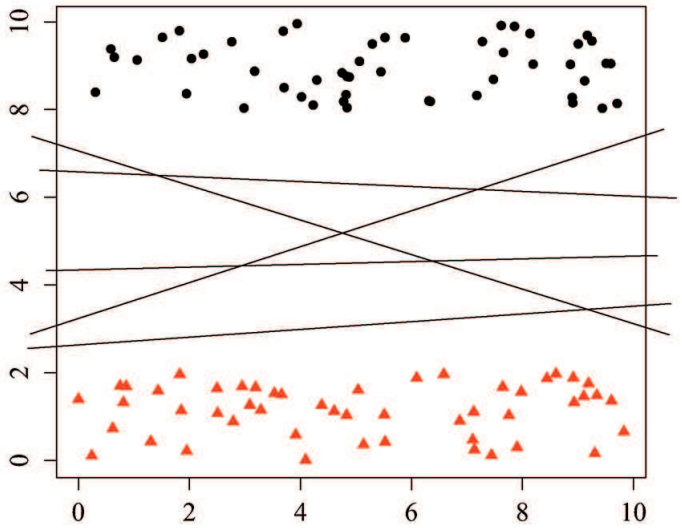



**Figura 3.3:** *Possíveis limites de decisão para um conjunto de dados.*

Classificadores com limites de decisão com margens pequenas são mais susceptíveis a *overfitting*, cenário em que um modelo ou função de classificação se adequa satisfatoriamente a apenas um conjunto limitado de amostras de dados mas apresenta baixa capacidade de generalização para amostras novas/desconhecidas. Além disso, tais classificadores possuem menor rendimento de classificação, uma vez que uma ligeira perturbação no limite de decisão pode ter um impacto significativo sobre o resultado da classificação [119]. Observando a Figura 3.4, vê-se que um exemplo desconhecido que ocorra imediatamente abaixo do limite de decisão $T$ em sua porção superior direita será classificado como triângulo, quando na verdade este exemplo demonstra maior similaridade com os exemplos representados pelo círculo; para o limite de decisão $L$, exemplos somente serão classificados como triângulo se estiverem pelo menos abaixo da metade do gráfico. Isto significa que, no pior caso, um exemplo classificado como triângulo pelo limite de decisão $L$ será mais similar aos demais exemplos triângulos de treinamento do que no caso do limite de decisão $T$.

Quando o SVM trabalha sobre *dados linearmente separáveis* o limite de decisão possui representação linear, como ilustrado na Figura 3.5 para um exemplo de conjunto de dados bidimensional arbitrário. Este limite de decisão é definido pela equação 3-2, sendo $x$ o conjunto de valores das variáveis de um exemplo arbitrário $e$, e $w$ é um conjunto de

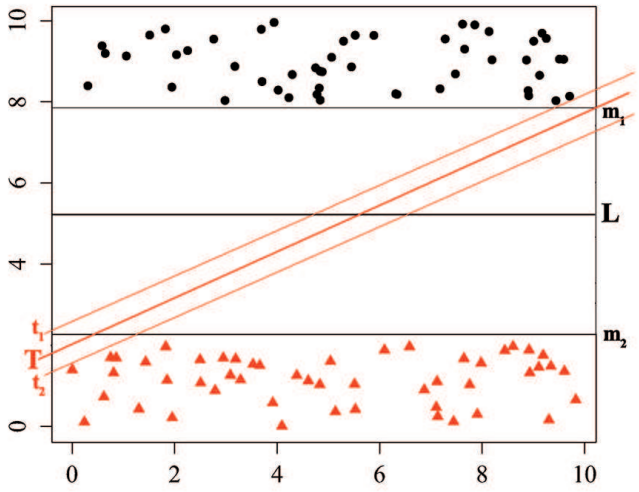



**Figura 3.4:** *Margem do limite de decisão.*

pesos cuja combinação linear com os valores de $x$ computa o valor do rótulo de classe $y$ predito para o exemplo $e$. Assim, se considerarmos que os dois possíveis valores de rótulos de classe para os exemplos são 1 e -1 (no exemplo da Figura 3.5, suponha que círculos possuem rótulo de classe 1 e triângulos possuem rótulo de classe -1), o SVM irá predizer o rótulo de classe de um novo exemplo como 1 se $w \cdot x + b > 0$ e -1 se $w \cdot x + b < 0$.

$$w \cdot x + b = 0 \tag{3-2}$$

Intuitivamente, a largura da margem do limite de decisão é dada pela distância entre os dois hiperplanos paralelos ao limite de decisão que tangenciam os exemplos de cada classe que estão mais próximos ao limite de decisão. Tais hiperplanos paralelos são conhecidos na literatura como vetores de suporte, e estão ilustrados na Figura 3.4, sendo $m_1$ e $m_2$ vetores de suporte para o limite de decisão $L$. A distância $d$ que representa a largura da margem do limite de decisão é calculada através da equação 3-3, e dado este valor, o objetivo geral do SVM é encontrar o limite de decisão cuja margem seja a maior possível. Isto significa minimizar a função objetivo descrita na equação 3-4.

$$d = \frac{2}{|| w ||} \tag{3-3}$$

$$\min_{w} \frac{|| w ||^2}{2} \tag{3-4}$$

Ao trabalhar com dados que não podem ser separados por um limite de decisão linear, o SVM utiliza um recurso chamado *truque do núcleo* que consiste em projetar os dados do seu espaço de coordenadas original descrito pelos valores de $x$ em um novo espaço $\Phi(x)$ onde os exemplos sejam separáveis por um limite de decisão linear, como

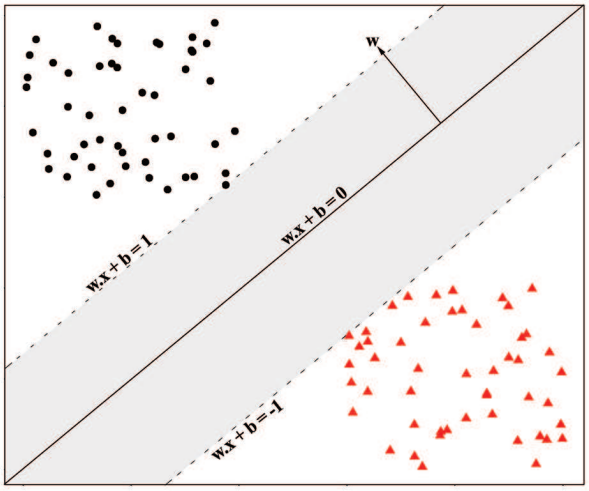



**Figura 3.5:** *Limites de decisão, margens do limite de decisão e parâmetros do SVM.*

mostrado na Figura 3.6. Este processo é feito através da aplicação de uma função de transformação sobre as variáveis do conjunto de dados, como a função exemplo descrita na equação 3-5. O limite de decisão linear no novo espaço será dado pela equação 3-2 adaptada para as variáveis transformadas, ou seja: $w \cdot \Phi(x) + b = 0$.

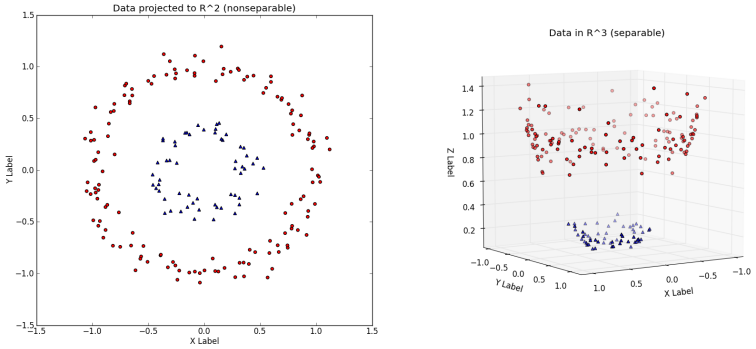



**Figura 3.6:** *Transformação de um conjunto de dados originalmente bidimensional ($\Re^2$) e não separáveis linearmente para um espaço tridimensional ($\Re^3$) onde eles são linearmente separáveis [57].*

$$\Phi : \Re^2 \rightarrow \Re^3 (x_1, x_2) \rightarrow (z_1, z_2, z_3) = (x_1^2, \sqrt{2} x_1 x_2, x_2^2) \tag{3-5}$$

Quando o conjunto de dados é muito grande, mapear todos os valores da matriz de dados para o novo espaço dimensional poderá requerer grandes quantidades de tempo, em alguns casos sendo um processo inviável. Uma alternativa para este cálculo é o uso das funções de núcleo, ou *funções kernel*, representadas por $K(x, y)$. Estas funções permitem expressar diretamente a similaridade entre dois exemplos no novo espaço transformado em função do produto de ponto entre eles sem que a transformação dos exemplos a

princípio seja de fato necessária. A Tabela 3.1 traz algumas das funções de kernel mais conhecidas e utilizadas em conjunto com o SVM.

| Kernel | Fórmula |
|---|---|
| Linear | $K(x,y) = x^T y + c$ |
| Polinomial | $K(x,y) = (\alpha x^T + c)^d$ |
| Gaussiana | $K(x,y) = exp(-\gamma \mid\mid x - y \mid\mid^2)$ |
| Laplaciana | $K(x,y) = exp(-\frac{\mid\mid x-y \mid\mid}{\sigma})$ |

**Tabela 3.1:** *Exemplos de funções kernel para uso com o SVM.*

### 3.2.2 Redes neurais artificiais

Redes neurais artificiais foram inspiradas no sistema cognitivo e nas funções neurológicas do cérebro humano, simulando os neurônios e seus ligamentos, responsáveis pelas transmissões de impulsos nervosos. Neurônios possuem os axônios, filamentos que conectam a outro neurônio através dos dendritos, e o ponto de conexão é chamado de sinapse. De maneira similar, uma rede neural artificial é um modelo de classificação composto de nós interconectados [119], comumente referidos também como *perceptron*. Atualmente, redes neurais artificiais tem sido amplamente empregadas em problemas de predição específicos caracterizados por conjuntos de dados amplos e descritos por grandes quantidades de variáveis, como reconhecimento de voz, escrita e face.

A Figura 3.7 mostra o desenho arquitetural de um modelo perceptron. Um perceptron é uma estrutura composta por $n$ nós de entrada, que simulam os neurônios. Assim como os neurônios humanos recebem estímulos, os nós da rede neural artificial recebem valores de entrada que serão processados e transferidos para nós em outras camadas da rede. Os nós localizados na camada inicial do modelo recebem e armazenam os valores de entrada das variáveis que descrevem um determinado exemplo. Cada nó desta camada inicial se conecta a outro nó, disposto na camada de saída do modelo, através de arestas que estão associadas a um valor de peso $w$, que simulam os axônios e a força da conexão da sinapse entre um nó de entrada e o nó de saída.

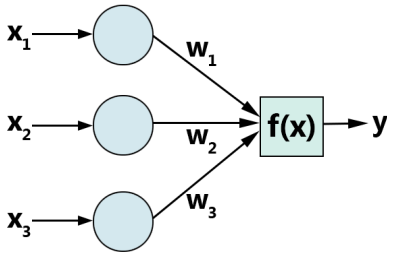



**Figura 3.7:** *Desenho arquitetural de um modelo perceptron básico.*

O nó de saída do modelo armazena a função que, utilizando os valores de entrada fornecidos e os pesos de suas respectivas arestas transmissoras, calcula o valor de saída $y$ do modelo, referente ao rótulo de classe para o exemplo investigado. Essa função envolve a soma ponderada dos valores recebidos pelos nós de entrada multiplicados pelo valor de peso das arestas correspondentes. O valor de saída é computado através de uma *função de ativação*, como por exemplo a função *sinal* descrita na equação 3-6, sendo $t$ um fator de tendência. Desta forma, $y$ no exemplo acima pode assumir dois valores distintos, $\{+1,-1\}$.

$$y = sign\left(\left(\sum_{k=1}^{d} x_d w_d\right) - t\right) \tag{3-6}$$

O *multilayer perceptron* (MLP) é uma extensão do modelo *perceptron* clássico que possui, além das camadas de entrada e saída, camadas intermediárias (também chamadas de camadas ocultas) formadas por nós chamados nós ocultos. A Figura 3.8 mostra um exemplo de estrutura de um MLP.

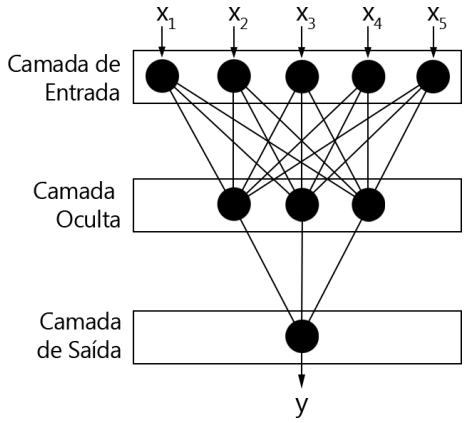



**Figura 3.8:** *Exemplo de um multilayer perceptron [119].*

Durante a fase de treinamento, o modelo MLP busca determinar valores para os pesos $w$ que minimize a soma total de erros quadrados, descrita pela equação 3-7. Desta forma, os valores de $w_1, w_2, ..., w_n$ são repetidamente ajustados conforme o modelo busca por um conjunto de valores ideal que reduza o erro de classificação o máximo possível.

$$E(w) = \frac{1}{2}\sum_{i=1}^{N}(y_i - \hat{y}_i) \tag{3-7}$$

Um algoritmo amplamento conhecido para o ajuste dos pesos é o *método de gradiente descendente*, que propõe a equação 3-8 para a atualização dos valores de $w$, onde $\lambda$ é chamada taxa de descoberta. Como a atualização do valor do peso depende do valor de saída obtido pelo classificador, para que os nós ocultos tenham acesso a este valor, o método de *propagação inversa* é utilizado. Este procedimento adiciona uma

fase no treinamento após o cálculo do valor de saída para um determinado exemplo de treinamento. Nesta fase adicional, o valor do erro de predição é propagado da camada de saída para as camadas anteriores, e os pesos dos nós nas camadas ocultas são atualizados em ordem reversa [119].

$$w_j = wj - \lambda \frac{\vartheta E(w)}{\vartheta w_j} \tag{3-8}$$

Finalmente, o algoritmo de treinamento com o método de propagação inversa nos modelos MLP pode ser resumido nos seguintes passos [38]:

1. Inicialize a rede neural com os seus valores de pesos;
2. Leia o primeiro exemplo de treinamento;
3. Propague os valores das variáveis do exemplo fornecido através da rede neural para que um valor de saída seja obtido;
4. Calcule o erro de saída através da comparação do valor de saída esperado com o valor de saída obtido;
5. Propague o erro de volta pela rede;
6. Ajuste os valores dos pesos de maneira a minimizar o erro geral de classificação;
7. Repita os passos 2-7 para um novo exemplo de treinamento, até que o erro geral seja satisfatoriamente pequeno.

O MLP tem sido utilizado para resolver uma variedade de problemas que podem ser categorizados em predição, aproximação de funções ou classificação de padrões [38]. Entretanto, existem alguns desafios inerentes ao uso deste classificador, como:

- Dificuldade de implementação e interpretação. Modelos baseados em redes neurais artificiais costumam ser vistos como ”caixa preta”, produzindo predições sem deixar explícito como exatamente elas foram obtidas, como as variáveis se relacionam e qual o impacto delas no valor final predito.
- A quantidade de nós e camadas ocultas a serem utilizadas são parâmetros que interferem grandemente no desempenho de classificação da rede neural, mas não possuem valores definidos. Embora alguns estudos sugiram valores padrões, cabe ao analista investigar e decidir qual serão os melhores valores que maximizam o desempenho da rede neural levando em conta as características do conjunto de dados investigado.

### 3.2.3 Árvores de decisão

Árvores de decisão [10, 97] se referem a um dos modelos de classificação mais antigos e mais populares devido à sua simplicidade, baixo custo computacional e rápida

generalização de novos exemplos. Árvores de decisão são relativamente fáceis de serem construídas e ainda mais fáceis de serem interpretadas, permitindo ao analista de dados enxergar com clareza como as variáveis se relacionam durante o processo de classificação.

Cada variável do conjunto de dados de treinamento é questionada individualmente, e estas questões e suas respostas formam regras de classificação. O objetivo da árvore de decisão é expressar as regras de classificação em uma estrutura em árvore. Cada nó da árvore corresponde a uma variável do conjunto de dados, e cada aresta que sai de um nó $x$ representa um valor, ou uma faixa de possíveis valores, para a variável $x$. Nós folhas armazenam os possíveis rótulos de classe, e correspondem ao ponto final do processo de classificação. A classificação de um exemplo desconhecido se dá através da checagem dos valores de suas variáveis, começando pelo nó raíz, analisando o resultado do teste da variável ali armazenada, e seguindo pela aresta que representa o valor obtido. Este processo iterativo gerará um percurso na árvore de decisão até que um dos nós folhas seja alcançado e o rótulo de classe associado a este nó folha será determinado como o rótulo de classe do exemplo fornecido [97].

A Figura 3.9 mostra um exemplo de árvore de decisão desenvolvida a partir do conjunto de treinamento mostrado na Tabela 3.2. O conjunto de treinamento possui 14 exemplos, descritos pelas variáveis "Aparência"que assume os possíveis valores ({ensolarado, nublado, chuvoso}), "Temperatura"({quente, moderado, frio}), "Umidade"({alta, normal}), "Ventania"({sim, não}), e o rótulo de classe "Classe"({P,N}).

| | Aparência | Temperatura | Umidade | Ventania | Classe |
|---|---|---|---|---|---|
| 1 | ensolarado | quente | alta | não | N |
| 2 | ensolarado | quente | alta | sim | N |
| 3 | nublado | quente | alta | não | J |
| 4 | chuvoso | moderado | alta | não | J |
| 5 | chuvoso | frio | normal | não | J |
| 6 | chuvoso | frio | normal | sim | N |
| 7 | nublado | frio | normal | sim | J |
| 8 | ensolarado | moderado | alta | não | N |
| 9 | ensolarado | frio | normal | não | J |
| 10 | chuvoso | moderado | normal | não | J |
| 11 | ensolarado | moderado | normal | sim | J |
| 12 | nublado | moderado | alta | sim | J |
| 13 | nublado | quente | normal | não | J |
| 14 | chuvoso | moderado | alta | sim | N |

**Tabela 3.2:** *Conjunto de treinamento [97].*

Várias árvores distintas podem ser construídas para resolver um único problema de classificação, e embora algumas árvores sejam mais precisas do que outras, encontrar a árvore ótima é computacionalmente inviável [119]. Uma das questões mais pertinentes a respeito do projeto de árvores de decisão e que influirá em seu desempenho de

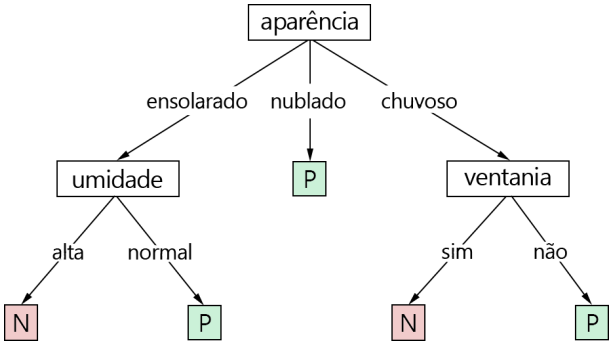



**Figura 3.9:** *Exemplo de estrutura de uma árvore de decisão [97].*

classificação e tempo de execução é a organização dos nós, isto é, a forma com a qual o questionamento de uma determinada variável irá particionar os dados para as demais variáveis nos nós seguintes. Em geral, deseja-se que uma determinada variável seja capaz de dividir os exemplos do conjunto de dados de maneira que exemplos de uma determinada classe tenham a maior frequência possível do que outra nos nós seguintes. A porcentagem de exemplos de uma mesma classe que estão alocados em uma partição está relacionada ao *grau de pureza* desta partição. Quanto maior for a porcentagem de exemplos de uma mesma classe alocados em uma partição gerada pelo questionamento de uma variável, mais pura será esta partição.

Um dos tipos de classificadores baseados em árvores de decisão mais conhecidos é o *ID3*, proposto por Quinlan [97]. O ID3 utiliza a chamada *entropia condicionada* como métrica de particionamento [86]. A função de entropia, utilizada para medir o grau de distorção da distribuição das classes $C = \{c_1, ..., c_j\}$, é definida pela equação 3-9, sendo $a$ um determinado atributo que descreve um exemplo de treinamento, $j$ a quantidade de rótulos de classe possíveis que o exemplo pode assumir, e $P(c_i \mid a)$ a probabilidade posterior do atributo $a$ na população $c_i$. A variável que apresentar o menor valor de entropia será aquela que divide os dados com maior grau de pureza [119, 86].

$$H(a) = -\sum_{i=1}^{j} p(c_i \mid a) log_2 p(c_i \mid a) \tag{3-9}$$

Desta forma, o algoritmo para a construção da árvore de decisão de acordo com o ID3 compreende os seguintes passos [86]:

1. Calcule o valor de entropia $H(a_i)$ para todas as variáveis $a_i$ que descrevem os exemplos e selecione a variável que apresentar o menor valor de entropia;
2. Divida o conjunto de dados de acordo com os possíveis valores de $a_i$ e gere seus correspondentes nós. O nó será terminal caso todos os exemplos pertençam a mesma classe;
3. Caso contrário, o nó gerado será não terminal. Para cada nó não terminal, escolha a variável $a_j$ que apresentar o menor valor de entropia $H(a_j)$.

4. Repita o passo 2 para a variável $a_j$.

A desvantagem principal do ID3 é que ele é sensível à quantidade de valores que as variáveis podem assumir, ou seja, seu custo computacional é altamente influenciado pela quantidade de valores possíveis das variáveis. O algoritmo *C4.5* é uma extensão do ID3 projetada para superar esta limitação [98]. O C4.5 utiliza a métrica do ganho de informação definida pela equação 3-10, onde $X$ é uma variável, $Y$ é o rótulo de classe, $H(Y)$ é o valor de entropia do rótulo de classe $Y$, $H(Y \mid X)$ é a entropia condicionada de $Y$ quando o valor de $X$ é conhecido [86]. A taxa de ganho de informação pode ser calculada através da fórmula 3-11, sendo $H(X)$ a entropia dos exemplos em relação ao atributo $X$.

$$I(Y \mid X) = H(Y) - H(Y \mid X) \tag{3-10}$$

$$TG(Y \mid X) = \frac{I(Y \mid X)}{H(X)}, \tag{3-11}$$

A vantagem da métrica da taxa de ganho de informação empregada pelo C4.5 em relação ao simples cálculo da entropia condicionada utilizada pelo ID3 é que a métrica da taxa de ganho de informação desencoraja a árvore de decisão a selecionar variáveis com grandes quantidades de valores distintos possíveis. Desta forma, ao trabalharmos com conjuntos de dados com muitas variáveis que podem assumir uma grande quantidade de valores distintos, o uso do algoritmo C4.5 é preferível.

Em suma, árvores de decisão possuem diversas vantagens. Árvores de decisão podem ser facilmente visualizadas e interpretadas. Através da observação das regras de decisão expostas nos caminhos da árvore, é possível gerar hipóteses a respeito da influência individual que cada variável exerce nos resultados de classificação e seus relacionamentos. Além disso, árvores de decisão apresentam baixo custo computacional não apenas em sua construção, como também para a generalização de novos exemplos [119].

## 3.3 Medidas de desempenho de classificadores

Avaliar o desempenho de um modelo de classificação significa averiguar o quão preciso é o modelo para predizer o rótulo de classe de novos exemplos. Um modelo de classificação perfeito adivinhará corretamente o rótulo de classe para todo e qualquer exemplo novo que seja fornecido, mas na prática, para a maioria dos conjuntos de dados reais, este cenário é bastante raro. Desta forma, o desempenho de um modelo de classificação também costuma ser medido pela sua taxa de erro. Ao passo que anular esta taxa de erro é uma tarefa difícil (e impossível em alguns problemas), é possível minimizá-la com diversas ações, como por exemplo:

- Ajustar repetidamente os parâmetros de um classificador e observar os resultados;
- Escolher um bom conjunto de variáveis para descrever os exemplos. Muitas vezes as variáveis escolhidas para representar o conjunto de dados são redundantes (correlacionadas) entre si, ou possuem pouca influência na informação que se deseja predizer (estas variáveis também são comumente chamadas de irrelevantes, ou com baixo poder discriminatório);
- Diminuir a quantidade de ruídos ou valores nulos nos dados. Ruídos são anomalias nos dados que podem ser causados por diversos fatores, como uma medição mal feita, interferência de sinal, danos causados durante a cópia ou leitura, entre outros. Valores nulos referem-se a medidas não coletadas ou inexistentes que farão com que o modelo de classificação desconheça o valor durante a leitura dos dados. Ruídos e valores nulos são prejudiciais ao desempenho do modelo de classificação. Ruídos, em particular, são difíceis de serem identificados e ruídos não tratados são averiguados como informação relevante durante o treinamento do classificador;
- Escolher um número satisfatório e equivalente de exemplos de treinamento para as classes distintas, de maneira que as classes fiquem igualmente representadas. Quando o conjunto de dados é desbalanceado, isto é, as classes não estão representadas de maneira equivalente ou similar, o classificador pode ignorar a importância das classes minoritárias e tender a fazer mais predições para as classes majoritárias, desta forma ocasionando um baixo desempenho de classificação de exemplos que pertençam às classes minoritárias. Um estudo detalhado dos efeitos nocivos provocados pelo desbalanceamento dos dados no desempenho dos modelos de classificação é provido no capítulo 2.
- Escolher um modelo de classificação adequado e otimizado para o tipo de problema com o qual se deseja trabalhar. Quantidade de variáveis descritoras, quantidade de exemplos de treinamento, tipos de variáveis, quantidade de possíveis valores para cada variável, são questões importantes a se fazer a respeito do conjunto de dados na hora de escolher o classificador ideal;
- entre outros.

A avaliação do desempenho de um classificador requer, além de um conjunto de exemplos de treinamento, um conjunto de exemplos de teste cujos rótulos de classes sejam conhecidos a princípio. Os exemplos de treinamento serão usados para construir o modelo e calibrar a função de decisão, enquanto os exemplos de teste serão utilizados para que o usuário obtenha informações sobre o quão preciso é o modelo desenvolvido para predizer o rótulo de classe de exemplos novos. Entretanto, esta abordagem apresenta dois problemas principais: um número muito pequeno de exemplos disponíveis no conjunto de dados torna a sua divisão em um subconjunto de exemplos para treinamento e outro para teste inviável [123], e o fato de que os resultados da avaliação dependem dos exemplos

que são aleatoriamente designados para o conjunto de treinamento ou para o conjunto de teste [63].

O método de *validação cruzada* é uma solução para o caso em que os conjuntos de treinamento e teste são formados por poucos exemplos [28]. No método de validação cruzada *k-fold*, o conjunto de dados $D$ é dividido de forma aleatória em $k$ subconjuntos mutuamente exclusivos $D_1, D_2, ..., D_k$ (os "folds", ou repartições) de tamanho igual ou similar. O classificador é treinado e testado em $k$ iterações, tal que a cada iteração $t \in \{1, 2, ..., k\}$ o classificador é treinado usando a repartição $D \setminus D_t$ e testado usando a repartição $D_t$. O método de validação cruzada calcula a acurácia como a quantidade geral de classificações feitas corretamente, dividida pelo número de exemplos total no conjunto de dados. A Figura 3.10 ilustra a execução do método de validação cruzada *k-fold*.

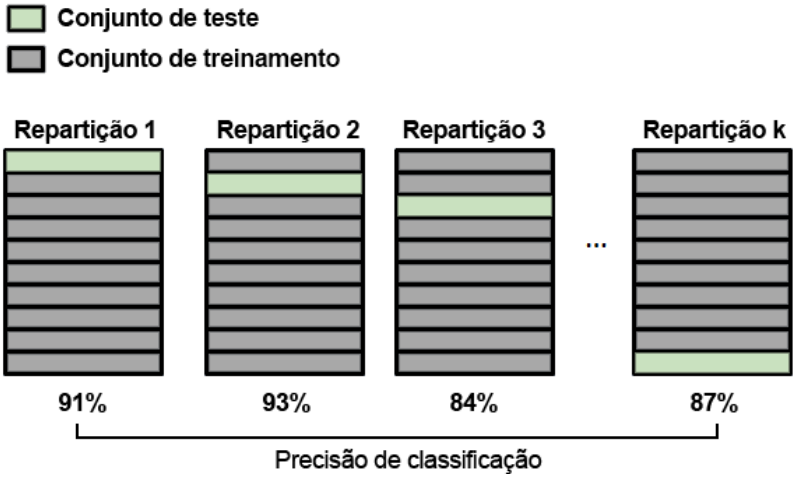



**Figura 3.10:** *Execução do método de validação cruzada k-fold.*

Os resultados de predição obtidos por um classificador podem ser organizados em uma *matriz de confusão*. Um exemplo de matriz de confusão está na Tabela 3.3. As linhas da matriz de confusão são os rótulos de classe esperados dos exemplos, enquanto as colunas referem-se aos rótulos previstos pelo classificador. Assim, um número $b_{ij}$ da matriz refere-se à quantidade de exemplos da classe associada à linha $i$ que foram classificados pelo modelo como pertencentes à classe associada à coluna $j$. De maneira intuitiva, todas as classificações corretas estão armazenadas na linha diagonal da matriz de confusão. Chamamos de *verdadeiros positivos* (VP) e *verdadeiros negativos* (VN) o número de exemplos da classe positiva e da classe negativa, respectivamente, que foram corretamente classificados, *falsos positivos* (FP) o número de exemplos da classe negativa que foram incorretamente classificados como sendo da classe positiva, e *falsos negativos* o número de exemplos da classe positiva que foram incorretamente classificados como sendo da classe negativa.

Os valores armazenados na matriz de confusão podem ser utilizados para obter outras medidas de desempenho populares, como *acurácia* (Equação 3-12), *sensitividade*

| Classe original | Classe predita | |
|---|---|---|
| | Positivo | Negativo |
| Positivo | Verdadeiro positivo (VP) | Falso negativo (FN) |
| Negativo | Falso positivo (FP) | Verdadeiro negativo (VN) |

**Tabela 3.3:** *Estrutura de uma matriz de confusão gerada para um modelo de classificação binário.*

ou *recall* (Equação 3-13) e *especificidade* (Equação 3-14). Acurácia refere-se à probabilidade geral do modelo classificar corretamente um exemplo de uma classe qualquer, dada pela razão do número de exemplos de teste classificados corretamente (verdadeiros positivos e verdadeiros negativos) sobre o número total de exemplos do conjunto de dados. Sensitividade refere-se a probabilidade geral do modelo classificar corretamente um exemplo arbitrário que pertença à classe positiva, e é dada pelo número de exemplos da classe positiva classificados corretamente (verdadeiros positivos) sobre o número total de exemplos da classe positiva no conjunto de dados (verdadeiros positivos e falsos negativos). Analogamente, especificidade estima a probabilidade geral do modelo classificar corretamente um exemplo arbitrário que pertença à classe negativa, dada pelo número de exemplos da classe negativa classificados corretamente (verdadeiros negativos) sobre o número total de exemplos da classe negativa no conjunto de dados (verdadeiros negativos e falsos positivos).

$$Acuracia(\%) = \frac{TP+TN}{TP+FP+FN+TN} \times 100 \tag{3-12}$$

$$Sensitividade(\%) = \frac{TP}{TP+FN} \times 100 \tag{3-13}$$

$$Especificidade(\%) = \frac{TN}{FP+TN} \times 100 \tag{3-14}$$

## 3.4 Problemas de classificação de dados publicados

### 3.4.1 Estudo dos elementos químicos mais importantes para a autenticação de folhas de laranja orgânicas

Neste estudo, foi realizada uma análise de folhas de laranjeiras orgânicas brasileiras com base na sua composição química. O produto resultante do trabalho foram modelos de classificação capazes de prever se uma laranjeira foi cultivada no sistema orgânico ou convencional através da análise dos níveis de concentração dos 14 elementos químicos encontrados em amostras de folhas de ambos os tipos de laranjeiras. O foco do trabalho foi empregar algoritmos de seleção de variáveis para entender o comportamento dos elementos químicos nas folhas e identificar aqueles mais relevantes que diferenciam

os dois tipos de folhas. Até onde sabemos, este foi o primeiro artigo que abordou o problema da classificação de folhas de laranja orgânicas com base na sua composição química. Este trabalho foi publicado em 2017 na revista acadêmica *Analytical Letters* [78].

Amostras de folhas de laranjeiras orgânicas e convencionais foram obtidas de fazendas orgânicas devidamente certificadas e próximas à cidade de Borborema, no estado de São Paulo, Brasil. As composições elementais das amostras foram determinadas via espectrômetro de massa por plasma indutivamente acoplado (ICP-MS). Um total de 14 elementos químicos foram encontrados nas folhas: Al, Cu, Mn, Rb, Sr, V, Cs, Ba, Cr, Co, Ni, Fe, Mg e Si. As concentrações destes elementos foram utilizadas como variáveis de entrada para as técnicas de mineração de dados empregadas no decorrer no estudo.

Para classificar as amostras de folhas entre orgânica ou convencional, foram desenvolvidos modelos de classificação baseados em máquinas vetores de suporte (SVM) e redes neurais artificiais (RNA). Para a avaliação dos elementos mais importantes nos dois tipos de folhas, os algoritmos de seleção de variáveis *Correlation-Based Feature Selection* (CFS), coeficiente chi-quadrado ($X^2$) e F-score foram empregados. A Tabela 3.4 resume os resultados obtidos pelos modelos de classificação baseados em SVM e RNA construídos para predizer o tipo da folha, quando as concentrações de todos os elementos químicos encontrados foram utilizados no treinamento. Ambos os modelos de classificação foram capazes de diferenciar as folhas orgânicas das convencionais com base na sua composição química com alto valor de acurácia, o que comprova que a composição química das folhas de laranja orgânicas e convencionais produzem padrões distintos e bastante distinguíveis.

| | SVM | RNA |
|---|---|---|
| Precisão (%) | 88.2 | 84.3 |
| Sensitividade (%) | 87.5 | 87.5 |
| Especificidade (%) | 77.8 | 81.5 |
| Valor Kappa | 0.7638 | 0.6866 |

**Tabela 3.4:** *Medidas de desempenho computadas para os modelos SVM e RNA construídos considerando todos os elementos químicos determinados.*

Os valores de F-score e coeficiente $X^2$ computados para cada elemento estimaram que as concentrações de magnésio, rubídio e manganês foram as variáveis mais relevantes para a diferenciação, e também concordaram em sua ordem de classificação. Por outro lado, o método CFS excluiu manganês do subconjunto das variáveis supostamente melhores. Para visualizar como esses elementos afetaram o desempenho dos modelos de classificação, novos modelos SVM e RNA foram treinados usando apenas as concentrações de Mg, Rb e Mn. Também foram construídos modelos adicionais utilizando apenas as concentrações de Mg e Rb. As medidas de desempenho para estes modelos são mostradas na Tabela 3.5. Em geral, o melhor modelo obtido foi o SVM usando todos os elementos

como variáveis de entrada, que apresentou acurácia de 88.2%, 87.5% de sensitividade e especificidade de 77.8%. Todavia, os elementos Mg e Rb sozinhos foram capazes de produzir modelos de classificação com acurácia muito satisfatória.

| Medida de desempenho | Todas as variáveis | | Mg, Rb, Mn | | Mg, Rb | |
|---|---|---|---|---|---|---|
| | SVM | RNA | SVM | RNA | SVM | RNA |
| Acurácia (%) | 88.2 | 84.3 | 82.3 | 76.5 | 82.3 | 76.5 |
| Sensitividade(%) | 87.5 | 87.5 | 79.2 | 75 | 83.3 | 75 |
| Especificidade(%) | 77.8 | 81.5 | 85.2 | 77.8 | 77.8 | 77.78 |

**Tabela 3.5:** *Medidas de desempenho computadas para os modelos SVM e RNA.*

### 3.4.2 Discriminação de tabletes de *ecstasy* apreendidos em Campinas e Ribeirão Preto, Brasil

Este projeto de pesquisa teve como objetivo realizar análise preditiva sobre comprimidos de ecstasy apreendidos pela polícia do estado de São Paulo, no período de Agosto de 2011 até Julho de 2012, em duas cidades, Campinas e Ribeirão Preto. Os parâmetros investigados foram a composição mineral da droga, determinada por ICP-MS. Desenvolveu-se modelos de classificação capazes de prever em qual das duas cidades uma amostra arbitrária de ecstasy seria mais provável ter sido apreendida. O trabalho foi publicado em 2018 na revista acadêmica *Neural Computing and Applications* [75].

O conjunto de dados analisado compreendeu amostras de tabletes de ecstasy apreendidos em duas cidades: Ribeirão Preto (17 amostras) e Campinas (21 amostras). A composição química das amostras foi determinada via ICP-MS, bem como as concentrações dos elementos em cada amostra, que foram utilizadas como variáveis de entrada para as técnicas de mineração de dados utilizadas no trabalho. Vinte e cinco elementos químicos foram encontrados nas amostras. Para a classificação das amostras, construiu-se modelos de classificação baseados em máquinas vetores de suporte (SVM) aliadas à função *kernel* gaussiana. Também foi empregado o algoritmo F-score para avaliação dos elementos e determinação daqueles mais importantes e mais irrelevantes para a classificação.

Através das avaliações obtidas pela aplicação do F-score, observou-se que as concentrações de selênio, molibdênio e manganês foram classificadas como os parâmetros mais importantes para a diferenciação das amostras de ecstasy apreendidas nas duas cidades. Estes elementos por si só foram capazes de gerar um modelo SVM que apresentou os melhores valores de acurácia (81.58%), sensibilidade (95.24%) e especificidade (64.71%). Os elementos menos discriminativos avaliados foram cádmio, magnésio e bário. Os elementos La, Ni, Te, Lu, Ca, Ce, Cu, Zn, Cs, Nd, Rb, Tl, Co, Ba, Mn e Cd, que

receberam os valores de F-score mais baixos, mostraram serem prejudiciais ao desempenho do modelo quando suas concentrações foram utilizadas para treiná-lo, causando uma queda nos valores das três métricas de desempenho investigadas.

### 3.4.3 Diferenciação de tabletes de chocolate fabricados de maneira orgânica e convencional

Neste trabalho, foi proposto um método simples para a diferenciação de amostras de chocolate orgânico de amostras de chocolate com ingredientes cultivados no sistema tradicional com base em sua composição química e métodos de análise multivariada de dados. Foi durante o desenvolvimento deste trabalho que as primeiras ideias a respeito do método SMOTE combinado com projeção de variáveis surgiram, uma vez que os dados analisados eram desbalanceados (24 amostras de chocolate convencional para apenas 12 amostras de chocolate orgânico) e foi preciso buscar estratégias para manter as duas classes (orgânico versus convencional) igualmente representadas. Neste trabalho, optou-se pelo uso do SMOTE, seguido pela aplicação de análise dos componentes principais (PCA) e análise discriminante linear (LDA) para diferenciar as amostras dos dois tipos de chocolate.

O conjunto de dados analisado consistiu em amostras de chocolate orgânico e convencional obtidas de diferentes supermercados nos estados brasileiros de São Paulo e Minas Gerais. Os elementos químicos e suas concentrações foram determinados nas amostras através de ICP-MS. Ao todo, 38 elementos foram identificados, tanto essenciais quanto tóxicos para o organismo humano.

O primeiro passo foi balancear a quantidade de amostras dos dois tipos de chocolate. A técnica SMOTE foi aplicada, gerando um novo conjunto de dados formado de mais 12 amostras sintéticas de chocolate orgânico além das 36 amostras originais, equiparando desta forma o número de amostras de ambas classes. Em seguida, PCA foi utilizado para identificar possíveis tendências e aproximações entre as amostras de chocolate. Finalmente, prosseguiu-se à diferenciação das amostras através do uso de análise discriminante linear. Metade das amostras foram utilizadas para treinar a função discriminante, enquanto a outra metade foi utilizada para testá-la. A função foi capaz de discriminar as amostras dos dois tipos de chocolate com 83% de precisão, classificando errôneamente apenas 4 amostras. Amostras de chocolate convencional apresentaram níveis maiores de alguns elementos essenciais ao organismo humano, especificamente ferro, zinco e magnésio, contrariando a ideia de que alimentos orgânicos são mais ricos em nutrientes. O trabalho foi publicado em 2018 na revista científica *Journal of Chemometrics* [59].

## 3.5 Conclusão

Neste capítulo, foi oferecida uma visão geral sobre o processo de análise preditiva e classificação de dados. A capacidade de inferir informações em amostras de dados a partir de dados previamente rotulados tornam os classificadores ferramentas de fundamental importância para o processo de descoberta de conhecimento, sendo amplamente empregados atualmente em problemas de domínio científico e coorporativo. Balanceamento e pré-processamento de dados em geral têm o objetivo suprassumo de tornar bases de dados mais limpas, confiáveis e significativas, eliminando riscos, inconsistências e amostras de dados ruidosas que possam prejudicar o desempenho de classificadores e do processo de descoberta de informação como um todo. Finalmente, o produto final desta tese trata-se de uma abordagem nova para balanceamento de dados, cujo desempenho será estimado indiretamente a partir do desempenho de modelos de classificação treinados com conjuntos de dados balanceados com o método proposto. Especificamente, os modelos escolhidos para tal fase de teste foram os três modelos descritos nas seções anteriores: máquinas vetores de suporte, árvores de decisão e redes neurais artificiais. Esta escolha é justificada não apenas pela grande popularidade destes modelos na literatura, auxiliando com sucesso na resolução de problemas de classificação de diversos domínios diferentes de conhecimento, mas também porque possuem estratégias de aprendizado distintas entre si.

CAPÍTULO 4

# Método proposto

## 4.1 Considerações iniciais

O objetivo desta tese é investigar e propor uma nova abordagem para balanceamento de dados. A primeira decisão de projeto foi adotar um método de *oversampling* como base. Tal escolha é justificada pelos seguintes motivos. Métodos de *resampling* são conceitualmente simples e intuitivos, uma vez que consistem em apenas alterar a estrutura do conjunto de dados, seja adicionando ou deletando amostras de dados. Isso os torna mais inteligíveis e amigáveis a não especialistas da área de aprendizagem de máquina, estatística e mineração de dados, sendo amplamente utilizados para balancear dados em problemas de diversos domínios. Além disso, aumentar a quantidade de amostras da classe minoritária através da adição de amostras sintéticas seria uma abordagem preferível a descartar amostras da classe majoritária, assim evitando o desperdício dos recursos investidos na obtenção destas amostras (um problema considerável especialmente quando a obtenção dos dados exigem recursos caros) e de possíveis informações discriminativas contidas nelas.

SMOTE é atualmente a estratégia padrão de *oversampling* de dados e também o algoritmo de balanceamento baseado em amostragem mais utilizado e mais estudado, com cerca de 85 extensões publicadas na literatura até o início de 2018 [32]. Diversas questões influenciam no desempenho do conjunto de dados resultante da aplicação do SMOTE, como por exemplo o conjunto de amostras escolhidas inicialmente. Um grande número de algoritmos baseados em SMOTE propostos na literatura recente procuram formas de avaliar e escolher as melhores amostras a serem utilizadas como ponto de partida para o *oversampling*. Enquanto estratégias como o *Borderline-SMOTE* focam em amostras nas bordas do limite de decisão, algoritmos como o *Safe-SMOTE* preferem focar o *oversampling* em amostras localizadas em regiões com mais densidade de amostras da classe minoritária. Entende-se que a escolha criteriosa dessas amostras iniciais pode contribuir para a redução de sobreposição de dados e a adição de ruídos resultantes da aplicação do SMOTE.

O modo como as amostras sintéticas serão criadas e onde exatamente serão

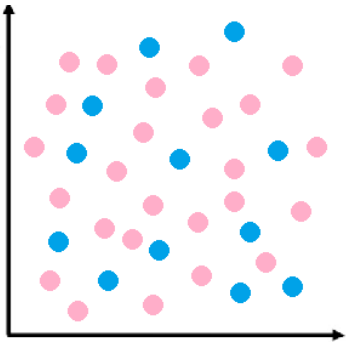
posicionadas é outra questão pertinente sobre o SMOTE. Por padrão, o SMOTE cria uma amostra sintética da classe minoritária e a coloca em alguma posição aleatória no segmento que une a amostra da classe minoritária selecionada inicialmente com o seu vizinho mais próximo (interpolação). Tal amostra sintética pode então ser posicionada tanto sobre a amostra inicial, como sobre a amostra vizinha, ou mais perto de uma do que de outra. [32] lista em seu trabalho diversas publicações recentes de extensões propostas para o SMOTE que atacam a forma de interpolação das amostras sintéticas, como impor restrições no alcance da interpolação, utilizar ponderação para decidir se a amostra sintética deve ficar mais próxima da amostra selecionada ou de sua vizinha mais próxima, utilizar interpolação baseada em agrupamento, utilizar interpolações múltiplas e outros.

Outro fator que influencia no desempenho do SMOTE é o nível de sobreposição das amostras de dados. Quando amostras de classes diferentes estão distribuídas de maneira muito próxima umas das outras, pode ser difícil para o SMOTE identificar grupos de amostras da classe minoritária próximas entre si de maneira que amostras sintéticas possam ser geradas e posicionadas no segmento entre elas sem que esta nova amostra se sobreponha à uma amostra da classe majoritária. Em alguns casos, quando amostras da classe minoritária estão cercadas de um número muito maior de vizinhos da classe majoritária do que da classe minoritária, estas amostras podem ser consideradas como *outliers* e ignoradas. A Fig. 4.1 ilustra este cenário. No caso ilustrado, as amostras sintéticas $g1$, $g2$ e $g3$ posicionadas no segmento entre a amostra $s$ e seus vizinhos mais próximos estão muito próximas ou sobrepostas a amostras da classe majoritária.

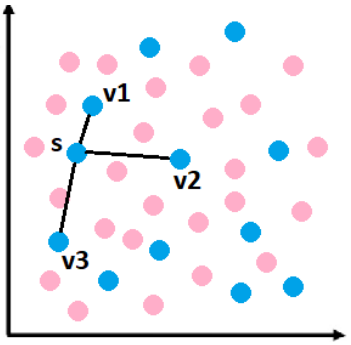
O método SMOTE básico, algoritmos baseados nele e outros algoritmos de *over-sampling* em sua maioria trabalham com distâncias entre amostras no plano dimensional para identificar vizinhos mais próximos. Cada algoritmo utiliza um critério para escolher qual vizinho mais próximo, ou qual ponto específico no plano dimensional, será utilizado como base para a interpolação da amostra sintética.

Assim, prossegue-se ao ponto chave deste projeto. Neste trabalho, propõe-se transformar os dados, ou projetá-los em um outro plano dimensional, antes que técnicas de *resampling* sejam aplicadas para balanceá-los.

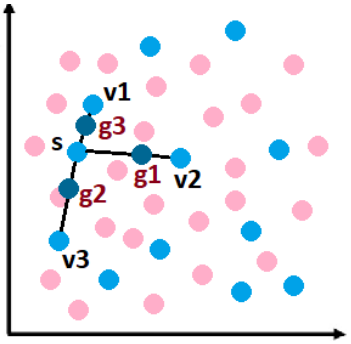
Transformar os dados significa projetá-los em um novo plano dimensional, transformando as variáveis originais $A$ que descrevem as amostras em um novo conjunto de variáveis $A'$ com base em algum cálculo realizado sobre as variáveis originais ($A$). Neste novo plano, as amostras de dados estarão reorganizadas em uma nova distribuição. Desta forma, tal transformação interfere diretamente no cálculo das distâncias entre amostras feitos por técnicas de *resampling*, uma vez que os vizinhos mais próximos de uma determinada amostra no espaço original possivelmente não serão os mesmos no novo espaço gerado pelas variáveis transformadas. *Portanto, a proposta deste trabalho*



(a) *Conjunto de dados original.*




(b) *Seleção da amostra s da classe minoritária e dos seus 3 vizinhos mais próximos v1, v2 e v3, também da classe minoritária.*




(c) *Amostras sintéticas g1, g2 e g3 da classe minoritária são geradas e posicionadas no segmento que liga a amostra s aos seus vizinhos mais próximos.*

**Figura 4.1:** *Exemplo de execução do SMOTE com sobreposição de dados.*

*é investigar até que ponto a* transformação prévia dos dados*, projetando as amostras do conjunto de dados em um outro espaço dimensional, descrito por novas variáveis, onde amostras da classe majoritária e as amostras da classe minoritária estejam supostamente melhor separadas, ou melhor agrupadas com base em algum padrão de similaridade, aprimoram o resultado final de técnicas de balanceamento de dados baseadas em resampling*.

O conjunto transformado pode ser maior do que o conjunto original (caso que leva ao aumento da dimensionalidade do conjunto de dados, como o *truque do kernel* realizado pelo SVM) ou menor (como realizado pelo algoritmo PCA, que reduz a dimensionalidade do conjunto de dados com base em combinações lineares entre as variáveis, facilitando a visualização dos dados). Assim, é necessário encontrar um novo $\Re^k$, sendo $k$ maior ou menor do que a dimensionalidade do conjunto de variáveis original, junto com as $k$ variáveis transformadas que montam um plano dimensional onde as amostras de cada classe estejam melhor agrupadas, o mais separadas possível por um

limite de decisão linear, ou agrupadas de acordo com algum padrão de similaridade.

Após a projeção dos dados neste novo plano dimensional descrito por novas variáveis transformadas, a ideia é proceder à execução do *oversampling* sobre as amostras das classes minoritária. Como no novo plano transformado as amostras estão reorganizadas de maneira diferente, cada amostra possuirá vizinhos mais próximos diferentes daqueles computados no plano original. Desta forma, o algoritmo proposto modifica a forma como o SMOTE escolhe os vizinhos da amostra selecionada durante o processo de *oversampling*. Além disso, técnicas baseadas em *kernel* e principalmente PCA são conhecidas por facilitarem a identificação de *outliers* [112, 105, 33, 117, 54], e desta forma a aplicação do SMOTE seria potencialmente restrita a amostras consideradas confiáveis.

Neste trabalho, optou-se por seguir o caminho da transformação de dados combinada com redução de dimensionalidade. Para isso, foi empregado o algoritmo PCA (análise dos componentes principais) como forma de transformar os dados antes da aplicação do SMOTE. Uma descrição detalhada deste método é colocada a seguir.

### 4.1.1 Análise dos componentes principais (PCA)

A presença de conjuntos de dados descritos por grandes números de variáveis é comum em problemas de mineração, ciência e classificação de dados atuais. Este número pode variar desde dezenas até milhares, dependendo do tamanho, complexidade e investimento por trás da aplicação. Intuitivamente, parece correta a ideia de que quanto mais variáveis são incorporadas à um conjunto de dados para análise, mais informações a respeito das amostras são observadas, facilitando a derivação de novos conhecimentos a respeito delas. Entretanto, na realidade, é comum que variáveis que em vista do analista humano pareçam valiosas para a extração de informações sobre as amostras sejam na verdade irrelevantes ou redundantes para a análise. Por exemplo, um modelo de classificação criado para um supermercado para predizer qual produto, entre cerveja e sorvete, venderá mais em um determinado mês, não se beneficiará de dados de temperatura sobre amostras de treinamento, uma vez que tanto cerveja quanto sorvete são produtos mantidos e vendidos em baixíssimas temperaturas. Neste caso, ”temperatura” seria uma coluna irrelevante no conjunto de dados de treinamento. Em geral, diz-se que uma variável é importante para a classificação de dados quando ela é relevante e irredundante. Variáveis redundantes são aquelas que possuem relativamente alto grau de dependência de outra(s), de maneira que a informação contida poderia ser igualmente obtida se apenas uma ou menos destas variáveis fossem utilizadas [14]. Variáveis irrelevantes são aquelas cuja informação contida não é útil para a geração de qualquer hipótese sobre as amostras com respeito aos seus rótulos de classe, isto é, a variável é independente do rótulo de classe. Para que uma variável seja importante, ela

precisa ser independente das demais variáveis que descrevem os dados, e ao mesmo tempo mostras dependência do rótulo de classe [14].

Como exemplo, considere o gráfico expresso na Fig. 4.2. Este gráfico representa um conjunto de dados bidimensional expresso em termos de suas duas variáveis, *x* e *y*.

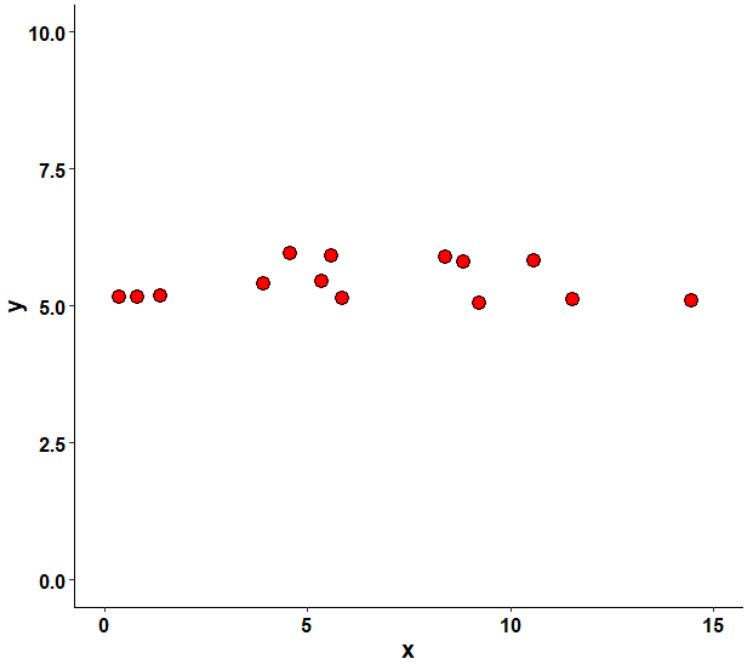



**Figura 4.2:** *Gráfico de um conjunto de dados bidiomensional.*

É possível ver que as amostras se concentram na região central do eixo *y*, se estendendo por todo o eixo *x* aproximadamente naquela altura. Em outras palavras, enquanto há uma grande variação nos valores das amostras para a variável *x*, há pouca variação para os valores da variável *y*, de maneira que os pontos de dados expressos no gráfico foram uma região similar a uma reta. Achatando o gráfico, isso é, descartando o eixo *y* e expressando estes mesmos dados apenas em termos de *x*, o resultado é a reta mostrada na Fig. 4.3. Porque o eixo *y* expressa pouca variação dos dados, é possível representar os dados em forma de uma reta em termos apenas do eixo *x*, responsável pela maior parte da variação nos dados, sem perda considerável de informação. A distribuição dos dados nesta reta é visualmente bastante similar à distribuição mostrada no gráfico dimensional. É possível enxergar o mesmo conjunto de dados, porém em uma quantidade reduzida de dimensões.

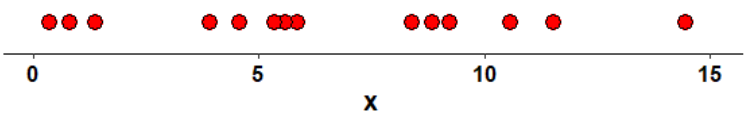



**Figura 4.3:** *Os mesmos dados mostrados na Fig. 4.2, expressos agora apenas em termos de x.*

Esta é a ideia principal por trás do PCA (análise dos componentes principais). PCA é uma técnica muito antiga e ainda bastante utilizada na literatura de análise e mineração de dados, cujo objetivo é reduzir a dimensionalidadede um conjunto de dados para facilitar a visualização e identificação de possíveis agrupamentos, tendências e padrões relevantes nos dados. Para isso, o PCA gera um novo conjunto de dados a partir do conjunto de dados original em que as variáveis são combinações lineares das variáveis do conjunto original que expressam a maior variação possível nos dados. Essas combinações lineares que posam como novas variáveis são chamadas de *componentes principais*. PCA gera os componentes principais de maneira que as variáveis em cada componente sejam descorrelacionadas e a informação contida nos dados seja comprimida nas primeiras componentes principais. Isso significa gerar os componentes principais em ordem decrescente de variância explicada, de maneira que o primeiro componente principal expressará a maior variação nos dados, o segundo componente principal expressará a segunda maior variação nos dados, e assim sucessivamente, como mostrado na Fig. 4.4. Desta forma, os dados trasformados podem ser visualizados em um gráfico agora em termos de suas componentes principais ao invés das variáveis originais. Idealmente espera-se que apenas duas ou três componentes principais sejam suficientes para mostrar os dados sem grandes perdas de informação, embora a perda em si seja praticamente inevitável, porém ao ganho de simplicidade em compensação. Resumidamente e de maneira informal, o método PCA organiza os dados em novos eixos (os componentes principais) que oferecem um ângulo melhor para que os dados sejam observados e analisados.

[102] descreve o problema por trás do PCA da seguinte forma: como obter uma nova base que seja uma combinação linear da base original que seja capaz de re-expressar os dados de maneira ótima? Considerando a fórmula de mudança de base da álgebra linear, isso significa que, dada a matrix $X$ de dimensões $n \times m$ composta pelos dados originais ($m$ variáveis e $n$ amostras), deseja-se encontrar uma matrix $P$ de dimensões $m \times m$ cuja multiplicação pela matrix $X$ gere uma outra matrix $Y$ de iguais dimensões ($Y = PX$), onde $Y$ contém os dados transformados e $P$ seja formado pelas direções das componentes principais.

O processo do PCA envolve os seguintes passos [116]:

1. Centralização dos dados: O conjunto/matriz de dados original $X$ é centralizado, isto é, para cada variável $y_m$ do conjunto de dados, calcula-se a média $\mu_{y_m}$ da variável e substrai-se esse valor da variável. A centralização é necessária devido à heterogeneidade de unidades de medidas das variáveis de entrada, de maneira que cada variável pode ser uma faixa de valores muito diferente (maiores/menores) do que outras, o que causa viés no cálculo do PCA. Chama-se $\hat{X}$ o conjunto $X$ centralizado.

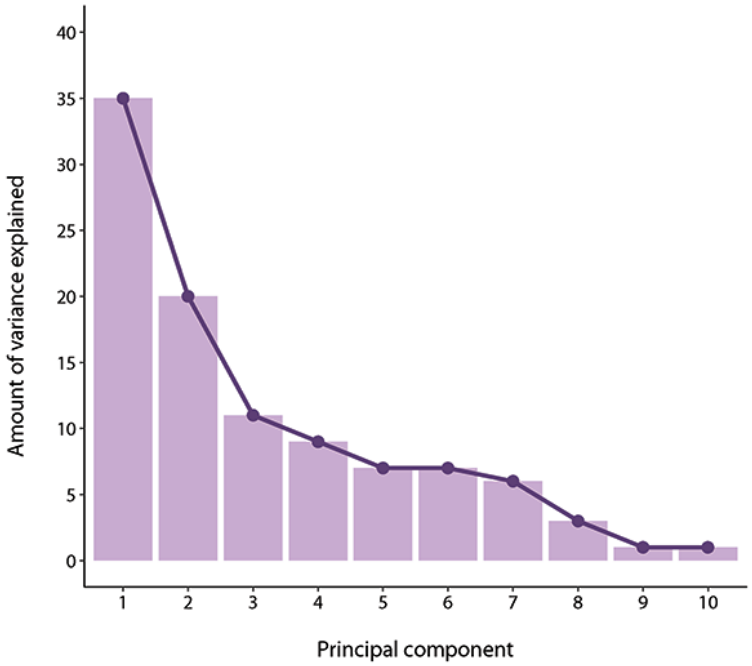



**Figura 4.4:** *Quantidade de variância total dos dados expressada por cada componente principal em um exemplo arbitrário. PC1 expressa 35% da variância total, PC2 expressa 20% da variância total etc.*

2. Cálculo da matriz de covariância: Trata-se de uma matriz simétrica $m \times m$ onde cada elemento $(i, j)$ da matriz é o valor de covariância entre a variável $i$ e a variável $j$, como mostrado na Tabela 4.1. A matriz de covariância guarda todos os valores de covariância entre pares de variáveis. Estes valores permitem identificar como as variáveis do conjunto de dados variam e correlacionam entre si. Se o valor de covariância entre $i$ e $j$ é positivo, então o valor de $i$ cresce conforme $j$ cresce e vice-versa; se o valor for negativo, os valores de $i$ crescem quando os valores de $j$ decrescem e vice-versa. Uma propriedade especial da matriz de covariância é que, intuitivamente, a sua diagonal contém as variâncias individuais de cada variável (uma vez que variância é um caso especial de covariância quando as duas variáveis são idênticas [102]). Por ser uma matriz simétrica, os valores constantes no triângulo superior e o triângulo inferior à diagonal são idênticos. PCA tenta construir componentes principais o mais descorrelacionados possível, ou seja, as variáveis de cada combinação linear devem apresentar valores de covariância o mais próximo possível de 0. Por outro lado, os componentes principais devem explicar a maior quantidade de variância possível nos dados, o que significa buscar valores grandes para a diagonal da matriz de covariância.
3. Cálculo dos autovetores e autovalores da matriz de covariância: Geometricamente, os componentes principais representam as direções da variação nos dados e são gerados pelo PCA em ordem decrescente, isto é, o primeiro componente principal

| | $a_1$ | $a_2$ | ... | $a_m$ |
|---|---|---|---|---|
| $a_1$ | $Cov(a_1,a_1)$ | $Cov(a_1,a_2)$ | ... | $Cov(a_1,a_m)$ |
| $a_2$ | $Cov(a_2,a_1)$ | $Cov(a_2,a_2)$ | ... | $Cov(a_2,a_m)$ |
| ... | ... | ... | ... | ... |
| $a_m$ | $Cov(a_m,a_1)$ | $Cov(a_m,a_2)$ | ... | $Cov(a_m,a_m)$ |

**Tabela 4.1:** *Exemplo de matriz de covariância para o conjunto de variáveis $a_1, a_2, ..., a_m$.*

representará a linha que captura a maior dispersão nos dados, o segundo componente principal representará a linha que captura a segunda maior dispersão nos dados e assim sucessivamente. A restrição é que, como os componentes principais são descorrelacionados entre si, a linha representada pelo primeiro componente principal deve ser perpendicular à linha representada pelo segundo componente principal e assim em diante. O cálculo dos autovetores da matriz de covariância determina as direções de maior variância, enquanto o cálculo dos autovalores determina a quantidade de variância explicada por cada autovetor. Ao final, basta ordenar os autovetores e seus respectivos autovalores de maneira decrescente para que se tenha formado o conjunto de componentes principais $P$ para um conjunto de dados em ordem inversa de variância explicada. Assim, $P = (av_1, av_2, ..., av_m)$, sendo $av_i$ o autovetor com a $i$-ésima maior variação explicada.

4. Transformar os dados originais de acordo com os componentes principais determinados: O conjunto de dados transformado $\hat{X}'$ pelo PCA para um novo espaço dimensional em termos dos componentes principais é obtidos através da multiplicação:

$$\hat{X}' = P_T \hat{X}_T \tag{4-1}$$

Aplicando a equação 4-1, tem-se agora os dados originais transformados, isto é, expressos em termos dos autovetores computados. A Fig. 4.5 mostra um exemplo de gráfico PCA gerado para o conjunto de dados exemplificado na Tabela 4.2.

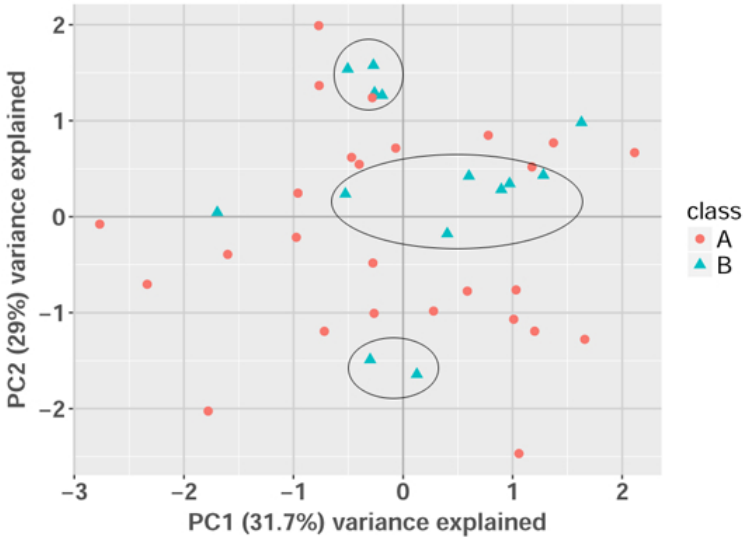



**Figura 4.5:** *Gráfico PCA gerado para o conjunto de dados ilustrado na Fig. 4.2.*

| | var1 | var2 | var3 | var4 | class | | var1 | var2 | var3 | var4 | class |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 3.86 | 65.35 | 3.4 | 7.04 | B | **21** | 7.11 | 55.35 | 5.64 | 7.18 | A |
| **2** | 3.04 | 83.89 | 10.51 | 7.19 | A | **22** | 6.01 | 93.66 | 5.84 | 5.64 | B |
| **3** | 2.9 | 51 | 5.83 | 5.81 | A | **23** | 5.79 | 27 | 8.87 | 10.81 | A |
| **4** | 5.96 | 73.56 | 6.76 | 7.47 | B | **24** | 0.45 | 46.71 | 6.39 | 4.98 | B |
| **5** | 1.6 | 107.64 | 6.9 | 8.5 | A | **25** | 13 | 61.7 | 9.62 | 5.24 | A |
| **6** | 12.09 | 110.95 | 10.61 | 7.31 | B | **26** | 8.03 | 40.68 | 5.66 | 6.16 | A |
| **7** | 3.64 | 45.57 | 9.12 | 5.38 | A | **27** | 4.42 | 25.46 | 4.44 | 6.3 | B |
| **8** | 4.49 | 122.69 | 5.13 | 4.99 | A | **28** | 8.24 | 83.19 | 11.97 | 6.22 | A |
| **9** | 19.56 | 25.16 | 9.61 | 7.7 | B | **29** | 15.78 | 113.38 | 3.2 | 8.02 | A |
| **10** | 17.76 | 58.18 | 8.14 | 5.57 | A | **30** | 8.82 | 47.52 | 11.34 | 5.5 | A |
| **11** | 7.1 | 108.55 | 4.78 | 3.02 | A | **31** | 8.27 | 31.08 | 10.32 | 8.31 | A |
| **12** | 12.02 | 57.15 | 2.81 | 4.86 | A | **32** | 20.66 | 98.95 | 10.58 | 6.62 | A |
| **13** | 6.27 | 78.11 | 14.57 | 6.03 | A | **33** | 6.42 | 20.28 | 3.64 | 7.41 | A |
| **14** | 6.66 | 105.41 | 14.98 | 9.3 | A | **34** | 4.81 | 115.27 | 2.73 | 6.43 | A |
| **15** | 11 | 96.49 | 9.09 | 6.7 | A | **35** | 4.06 | 42.94 | 5.49 | 5.71 | B |
| **16** | 15.12 | 118.38 | 6.34 | 6.78 | A | **36** | 14.94 | 66.01 | 12.53 | 2.48 | A |
| **17** | 8.49 | 90.97 | 7.14 | 6.22 | B | **37** | 9.42 | 67.68 | 6.17 | 4.66 | B |
| **18** | 3.48 | 16.81 | 2.79 | 6.91 | A | **38** | 11.35 | 50.6 | 3.18 | 3.83 | B |
| **19** | 5 | 31.48 | 10.43 | 7.81 | B | **39** | 13.13 | 49.9 | 4.35 | 4.7 | B |
| **20** | 3.85 | 42.1 | 14 | 8.43 | A | **40** | 7.36 | 131.21 | 2.67 | 7.18 | B |

**Tabela 4.2:** *Exemplo de conjunto de dados com 4 variáveis numéricas e duas classes possíveis, A e B. Classes A e B possuem 26 e 14 amostras respectivamente.*

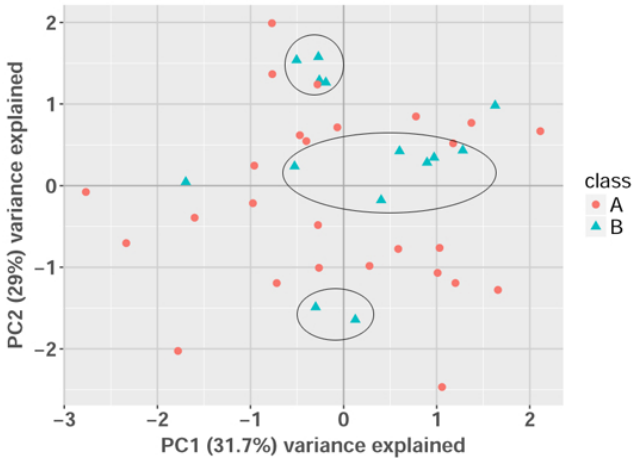
## 4.2 Descrição do PCA-SMOTE

Para averiguar o potencial da transformação de dados como fator de aprimoramento do desempenho de algoritmos de balanceamento de dados, propõe-se um algoritmo prático com três etapas principais:

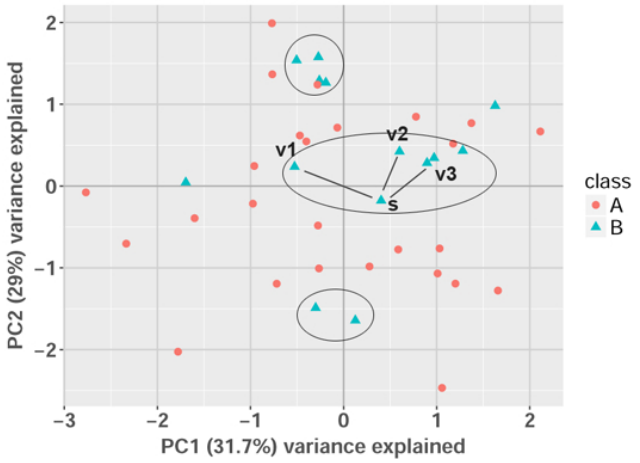
1. Transformação dos dados originais, ou projeção destes em um novo plano dimensional descrito por variáveis transformadas de acordo com alguma função ou critério;
2. Aplicação de um algoritmo de *oversampling* sobre os dados transformados;
3. Restauração dos dados para o plano original.

Para a etapa 1, investiga-se inicialmente o método PCA para transformar os dados. Como descrito na seção anterior, o resultado final da aplicação do PCA sobre um conjunto de dados é um novo conjunto de dados expresso em termos de padrões existentes entre eles, em forma de combinações das contribuições de cada variável original [116]. Tomando como exemplo o conjunto de dados da Tabela 4.2, temos na Fig. 4.6 o resultado da aplicação do PCA. Para facilitar a visualização dos dados, este exemplo utiliza apenas as duas primeiras componentes principais para expressar os dados, formando assim um plano bidisimensional. As regiões circuladas no gráfico são os pequenos agrupamentos formados pelas amostras da classe minoritária no espaço transformado expresso em termos destas duas componentes principais.

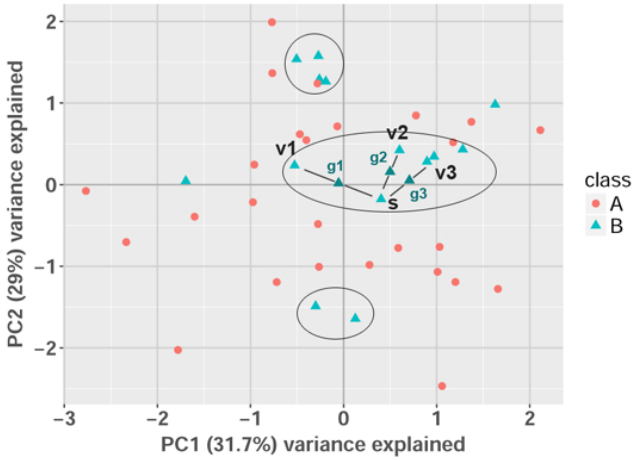
Para a etapa 2, opta-se inicialmente por utilizar o algoritmo SMOTE para realizar o *oversampling* dos dados transformados. O SMOTE é aplicado sobre as amostras reposicionadas no novo plano dimensional construído de acordo com as novas variáveis (componentes principais) computadas pelo PCA. As figuras 4.6(b) e 4.6(c) ilustram a aplicação do SMOTE no espaço transformado pelo PCA. Em 4.6(b), uma amostra $s$ no espaço transformado é selecionada e as 3 amostras vizinhas mais próximas também da classe minoritária são identificadas ($v_1, v_2 e v_3$). Em seguida, em 4.6(c), amostras sintéticas são geradas e posicionadas no segmento que liga a amostra $s$ às suas vizinhas mais próximas. Facultativamente, pode-se manter todas as amostras sintéticas geradas para cada vizinho ou então selecionar apenas uma destas amostras sintéticas e descartar as demais. Todo o processo é então repetido até que uma quantidade satisfatória de amostras da classe minoritária tenha sido gerada.




(a) *Gráfico PCA gerado para o conjunto de dados ilustrado na Fig. 4.2.*




(b) *Amostra s da classe minoritária e seus vizinhos da classe minoritária mais próximos v1, v2 e v3 selecionados.*




(c) *Amostras sintéticas g1, g2 e g3 da classe minoritária geradas e posicionadas nos segmentos entre s e seus vizinhos.*

**Figura 4.6:** *Exemplo de execução do SMOTE combinado com transformação de variáveis pelo PCA.*

Finalmente, após geradas as amostras sintéticas no espaço transformado, é necessário revertê-lo à forma original, descrito pelas variáveis originais de entrada. Isso significa basicamente computar para cada amostra sintética gerada os seus valores em termos das variáveis originais e então anexá-las ao conjunto de dados original, dispensando o tempo de processamento para reverter todo o conjunto de dados. Lembrando que a transformação dos dados originais em pontuações obtidas pelo PCA é obtida através da equação 4-1, temos então:

$$\hat{X}_T = P_T^{-1}\hat{X}' \tag{4-2}$$

Sendo $P_T^{-1}$ a matriz inversa de $P_T$, que será de fato igual a $P_{T_T}$ quando todos os componentes principais foram utilizados na transformação. Substituindo na fórmula, tem-se que:

$$\hat{X}_T = P\hat{X}' \tag{4-3}$$

A equação 4-3 faz o cálculo dos valores $\hat{X}$, que são os dados originais centralizados. Para computar os valores dos dados originais antes da centralização, é necessário adicionar os valores das médias novamente nas variáveis de $\hat{X}$. Ou seja:

$$X_T = P\hat{X}' + \mu \tag{4-4}$$

Assim, finalmente tem-se o conjunto de dados original reconstruído. Como dito anteriormente, é possível simplificar a reconstrução dos dados originais simplesmente reconstruindo as amostras sintéticas geradas e adicionando-as ao conjunto de dados original, evitando que todo o conjunto de dados tenha que ser reprocessado. Para cada amostra de dados $p$ no espaço transformado $P$, tal amostra pode ser mapeada para o espaço original através da fórmula:

$$s = pP_T + \mu \tag{4-5}$$

O algoritmo proposto PCA-SMOTE pode então ser sumarizado nos seguintes passos, também ilustrados através da Fig. 4.7:

1. Considere $X$ o conjunto de dados a ser analisado e $A$ o conjunto finito de variáveis que descrevem $X$ de tamanho $m$;
2. Aplique o método PCA em $X$ e extraia as pontuações (*scores*) e as componentes principais e seus respectivos carregamentos (*loadings*) $PC = \{PC_1, ..., PC_m\}$ obtidas. Considere $A_{X'} = PC$ como o conjunto de variáveis transformadas e $X' = scores$ como o novo conjunto de dados transformado;
3. Para cada amostra $s'$ em $X'$ da classe minoritária, calcule seus $k$ vizinhos mais próximos também da classe minoritária $V = \{v_1, v_2, ..., v_k\}$;
4. Escolha aleatoriamente $v_i \in V$ e compute o segmento $S(s', v_i)$ que liga a amostra $s'$ ao seu vizinho $v_i$ mais próximo;

5. Calcule uma amostra sintética $s'_s$ a partir da soma dos valores de $s'$ com os valores de $v_i$ multiplicados por um número $\sigma$ no intervalo $[0, 1]$;
6. Calcule $s_s$, que é a reconstrução de $s'_s$ para o espaço dimensional original (descrito pelas variáveis $A$);
7. Adicione $s_s$ ao conjunto de dados original $X$.
8. Repita os passos 3-7 até que $X$ esteja balanceado de maneira satisfatória.

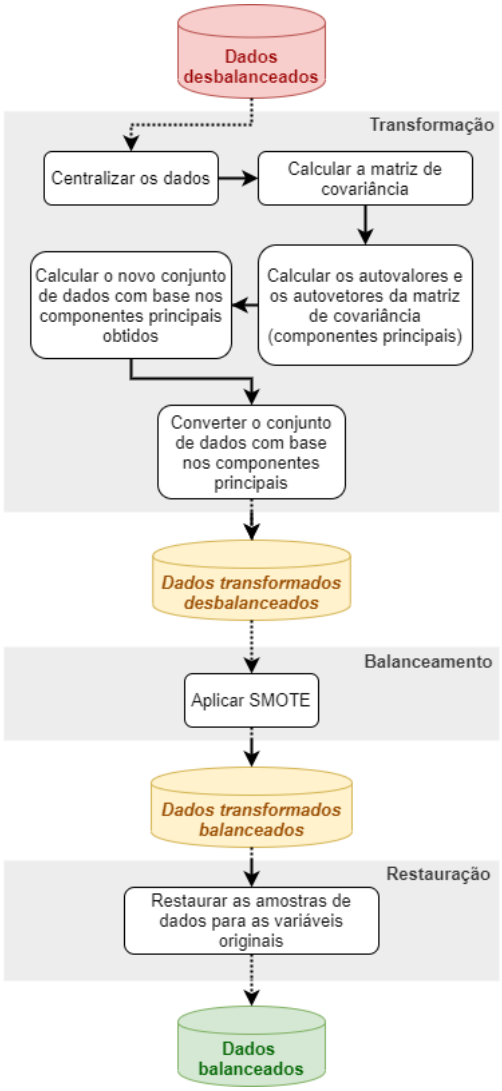



**Figura 4.7:** *Fluxo de execução do algoritmo PCA-SMOTE.*

CAPÍTULO 5

# Testes realizados e resultados obtidos

Neste capítulo estão detalhados todos os resultados obtidos durante os testes conduzidos sobre o algoritmo proposto. As bases de dados utilizadas, a metodologia de teste empregada, as características do ambiente de teste bem como as métricas de desempenho computadas para modelos de classificação desenvolvidos antes e após o balanceamento de dados utilizando o algoritmo proposto estão descritos em detalhes. Finalmente, discute-se os resultados obtidos.

## 5.1 Metodologia de teste

O algoritmo foi implementado na linguagem R, utilizando o software RStudio versão 1.1. Os modelos de classificação necessários para testar o balanceamento de dados também foram desenvolvidos nessa linguagem, bem como todas as demais estruturas de dados necessárias para a implementação e condução dos testes. Para os algoritmos de balanceamento já existentes contra os quais o desempenho do PCA-SMOTE foi comparado, especificamente *Borderline-SMOTE*, *Safe-Level-SMOTE*, ADASYN e o SMOTE tradicional, foram utilizadas as implementações disponibilizadas por Romero Barata em seu pacote *bimba* [4]. O código implementado para o algoritmo PCA-SMOTE está disponível no Apêndice A. Adicionalmente, foram também inclusos no Apêndice B os cógidos dos demais recursos do sistema de teste implementado.

Para o teste do algoritmo PCA-SMOTE, foram utilizadas as bases de dados descritas na Seção 5.2. Cada base de dados de teste $X$ foi padronizada e então balanceada de acordo com os algoritmos PCA-SMOTE, *Borderline-SMOTE*, *Safe-Level-SMOTE*, ADASYN e o SMOTE tradicional, gerando-se desta forma cinco versões diferentes de balanceamento ($X_{\text{smote}}$, $X_{\text{pca-smote}}$, $X_{\text{borderline-smote}}$, $X_{\text{safesmote}}$, $X_{\text{adasyn}}$), além da versão desbalanceada. O objetivo era comparar o desempenho obtido por modelos de classificação desenvolvidos sobre cada uma dessas versões, permitindo assim a obtenção de uma estimativa da qualidade do algoritmo proposto em comparação a outros algoritmos populares na literatura, incluindo o próprio SMOTE básico. Assim como os modelos de classificação, tais algoritmos de balanceamento foram escolhidos como concorrentes do método

proposto porque, além de serem métodos de balanceamento conhecidos e muito empregados em trabalhos na literatura, possuem embasamentos e raciocínios diferentes a respeito de como e onde efetuar o oversampling das amostras.

Como mencionado na seção 3.2, os modelos de classificação escolhidos para o teste foram SVM, árvores de decisão (C4.5) e redes neurais artificiais da família *multilayer perceptron* (MLP). Tais modelos implementam algoritmos de aprendizado diferentes, cada qual com seu conjunto de vantagens e desvantagens específicos, além de serem modelos amplamente utilizados com sucesso na literatura para a classificação de diversos conjuntos de dados de diferentes domínios. Foram utilizadas as implementações disponíveis no pacote *caret*, que trazem por padrão as seguintes configurações de parâmetros para cada modelo:

- Modelo SVM: função kernel gaussiana (RBF), com valor de C escolhido por padrão através de busca em grade nos valores $C = \{0.25, 0.5, 1\}$
- Modelo C4.5: fator de confidência 0.2 e mínimo de 3 objetos por nó-folha
- Modelo MLP: 1 camada oculta contendo $(m+2)/2$ nós ocultos, sendo $m$ o número de variáveis no conjunto de dados

As métricas escolhidas para expressar o desempenho dos modelos foram acurácia, sensitividade, especificidade, AUC (área abaixo da curva ROC), média geométrica da precisão pela sensitividade (*g-mean*) e *f-measure*.

Para cada versão balanceada, os modelos de classificação foram treinados e testados através do método de validação cruzada 5-*fold*. O uso de apenas cinco partições é justificado pelo tamanho pequeno de alguns dos conjuntos de dados de teste, o que tornaria inviável as suas divisões em muitas partições. Para evitar viéses possivelmente causados pela divisão aleatória das amostras de dados em cinco partições, cada validação cruzada foi repetida 10 vezes, computando-se desta forma dez valores para cada métrica de precisão. Os valores finais expressos para cada métrica de desempenho ($\text{Acc}_f$, $\text{Sens}_f$, $\text{Espec}_f$, $\text{AUC}_f$, $\text{G-mean}_f$ e $\text{F-measure}_f$) foram as médias aritméticas entre estes dez valores:

- $\text{Acc}_f = \frac{acc_1+...+acc_10}{10}$
- $\text{Sens}_f = \frac{sens_1+...+sens_10}{10}$
- $\text{Espec}_f = \frac{esp_1+...+esp_10}{10}$
- $\text{AUC}_f = \frac{a_1+...+a_10}{10}$
- $\text{G-mean}_f = \frac{g_1,...,g_10}{10}$
- $\text{G-mean}_f = \frac{f_1,...,f_10}{10}$

O procedimento de teste pode então ser resumido da seguinte forma:

1. Para cada base de dados X:
   (a) Substitua possíveis valores nulos pelo valor da média da variável
   (b) Padronize os valores de X para terem média 0 e desvio padrão 1
   (c) Compute as versões balanceadas $X_{\text{smote}}$, $X_{\text{pca-smote}}$, $X_{\text{borderline-smote}}$, $X_{\text{safesmote}}$, $X_{\text{adasyn}}$
   (d) Para $X$ e também para cada versão balanceada $X_i$:
      i. Treine um modelo SVM através de validação cruzada 5-*fold*
      ii. Calcule acurácia, sensitividade, especificidade, valor AUC, *G-mean* e *F-measure* para o modelo SVM obtido
      iii. Treine um modelo C4.5 através de validação cruzada 5-*fold*
      iv. Calcule acurácia, sensitividade, especificidade, valor AUC, *G-mean* e *F-measure* para o modelo C4.5 obtido
      v. Treine um modelo MLP através de validação cruzada 5-*fold*
      vi. Calcule acurácia, sensitividade, especificidade, valor AUC, *G-mean* e *F-measure* para o modelo MLP obtido
      vii. Repetir o procedimento (b) mais nove vezes

Para cada base de dados testada, testes de hipóteses pareados de Wilcoxon também foram conduzidos entre os valores de $\text{Acc}_f$, $\text{Sens}_f$, $\text{Espec}_f$, $\text{AUC}_f$, $\text{G-mean}_f$ e $\text{F-measure}_f$ obtidos pelos modelos para cada versão balanceada. Este teste não-paramétrico foi usado para verificar se havia de fato diferenças significativas entre os valores de desempenho obtidos quando cada base de dados é balanceada de acordo com determinado algoritmo. O valor $p$ (*p-value*) produzido por cada teste representa o menor nível de significância que uma hipótese pode assumir para que ela seja rejeitada, permitindo verificar se dois algoritmos realmente produzem resultados diferentes [104]. Tomando como base a metodologia de teste empregada para o algoritmo SMOTE-IPF [104], considerou-se que o valor máximo que poderia ser atribuído ao *p-value* para que a diferença entre médias fosse considerada significativa foi de 0.1.

## 5.2 Bases de dados utilizadas

Todas as bases de dados utilizadas durante os testes descritos nesta tese estão disponíveis gratuitamente para pesquisa no repositório UCI Machine Learning Repository [27]. Todas elas referem-se a dados reais obtidos de fontes confiáveis, e nenhuma base de dados artificial foi utilizada. As bases de dados selecionadas possuem diferentes dimensões e variáveis descritoras, de maneira a investigar o potencial do algoritmo para balancear conjuntos de dados de várias naturezas.

## Breast Cancer Coimbra

O conjunto de dados *Breast Cancer Coimbra* publicado por [89] consiste em dados clínicos de 116 pacientes de câncer de mama, sendo 64 destas amostras referentes a pacientes com a doença e as 52 amostras restantes referentes a pacientes saudáveis. As variáveis analisadas são numéricas e referem-se a dados antropométricos que podem ser obtidos através de análise sanguínea de rotina, especificamente: idade do paciente; índice de massa corpórea (kg/m$^2$), glucose (mg/dL), insulina ($\mu$U/mL), índice de Homa, leptina (ng/mL), adiponectina ($\mu$g/mL), resistina (ng/mL), nível de MCP-1 (pg/dL), classe (1 para paciente saudável, 2 para paciente com câncer presente).

## Arroz

Trata-se da base de dados utilizada no trabalho [74]. Contém as concentrações de 20 elementos químicos determinados via espectrometria de massa por plasma indutivamente acoplado (ICP-MS) em amostras de arroz do tipo *Oriza Sativa* produzidas nos estados de Goiás e Rio Grande do Sul, que utilizam sistemas de plantio irrigado e cerqueiro, respectivamente. Os elementos encontrados foram cobre, zinco, magnésio, boro, fósforo, molibdênio, arsênio, chumbo, cádmio, manganês, selênio, cobalto, cromo, bário, rubídio, ferro, potássio, cálcio, lantânio e cério, e suas unidades de medida no conjunto de dados estão misturadas entre ng/g, mg/kg e g/100g. O rótulo de classe e alvo da diferenciação das amostras é o estado de origem da amostra de arroz (Goiás ou Rio Grande do Sul). A base de dados possui originalmente 31 amostras de arroz, sendo 12 amostras produzidas em Goiás e as demais 19 amostras produzidas no Rio Grande do Sul. Para este trabalho, 4 amostras de Goiás foram aleatoriamente descartadas de maneira a agravar o desbalanceamento da base.

## Chocolate orgânico

Base de dados utilizada no estudo [59]. Contém as concentrações de 40 elementos químicos determinados via ICP-MS em amostras de chocolate orgânico e convencional obtidos de diferentes marcas e supermercados nos estados de São Paulo e Minas Gerais. Os elementos listados nas variáveis são arsênio, alumínio, escândio, cádmio, cobre, ferro, magnésio, manganês, níquel, chumbo, vanádio, urânio, zinco, lantânio, disprósio, praseodímio, estanho, neodímio, gadolínio, bário, antimônio, berílio, selênio, molibdênio, cobalto, tálio, érbio, cério, itérbio, lutécio, tório, hólmio, germânio, nióbio, samário, térbio, túlio e európio, medidos em ng/g e $\mu$g/g. Das 36 amostras contidas na base, 12 são de chocolate orgânico e as demais 24 de chocolates convencionais, todas com conteúdo de cacao normalizado em 70%. Todas as variáveis são numéricas. O rótulo de classe e alvo

da diferenciação das amostras é o tipo de chocolate (orgânico ou convencional). Ressalta-se que o desbalanceamento do conjunto de dados exigiu dos autores do trabalho original [59] o uso de algoritmos de balanceamento para que os modelos construídos apresentassem bons desempenhos para predizer os dois tipos de chocolate. O algoritmo empregado por eles foi o SMOTE tradicional.

## Folhas de laranja orgânicas

Esta base de dados foi utilizada no trabalho [78] para o estudo dos elementos químicos mais importantes que diferenciam folhas de laranja cultivadas de maneira orgânica daquelas cultivadas de maneira tradicional. A base original contém 52 amostras de folhas de laranja do tipo Valencia(*Citrus sinensis* [L.] Osbeck), sendo que 28 foram colhidas de fazendas orgânicas e as demais 24 de fazendas tradicionais. As variáveis da base consistem em medições em $\mu$g/g das concentrações dos 14 elementos químicos determinados nas folhas através de ICP-MS, especificamente alumínio, bário, cobalto, crômio, césio, cobre, ferro, magnésio, manganês, rubídio, níquel, silício, estrôncio e vanádio. A décima quinta variável, também considerada como o rótulo de classe, é o tipo de cultivo da folha (orgânica ou convencional). Para este trabalho, 18 amostras orgânicas foram aleatoriamente descartadas de maneira a desbalancear o conjunto de dados. O conjunto de teste resultante possui 10 amostras orgânicas e 24 amostras convencionais.

## Suco de uva orgânico

Base de dados utilizada no estudo [76], que investiga o desempenho de diferentes modelos de classificação para diferenciar amostras de suco de uva orgânico de amostras convencionais. A base de dados original possui 37 amostras de suco de uva obtidos de diferentes estados brasileiros, sendo 19 amostras referentes a sucos de uva orgânicos certificados e as demais 18 amostras de sucos de uva comuns. As variáveis são todas numéricas e correspondem às medições em ng/g e $\mu$g/g de 44 elementos químicos determinados nas amostras via ICP-MS. A quadragésima quinta variável e rótulo de classe das amostras é o tipo da amostra (orgânica ou convencional). Para este trabalho, 10 amostras orgânicas foram aleatoriamente descartadas de maneira a desbalancear o conjunto de dados. O conjunto de teste resultante possui 9 amostras orgânicas e 18 amostras convencionais.

## Breast Cancer Wisconsin (Original)

O conjunto de dados *Breast Cancer Wisconsin Original* foi coletado por William H. Wolberg, médico associado ao *University of Wisconsin Hospitals*, e doado para

o repositório após a publicação de alguns de seus trabalhos conduzidos sobre eles [80, 127, 79]. Esta base reúne 699 amostras de casos clínicos de câncer de mama tratados por Dr. Wolberg, sendo 241 amostras refentes a pacientes portadores do câncer malígno e as demais 458 amostras referentes a pacientes portadores da versão benígna do câncer. As 11 variáveis descritoras consistem em informações celulares de amostras do câncer colhidas do paciente, especificamente: identificação do paciente, espessura do amontoado de células, uniformidade do tamanho das células, uniformidade do formato das células, adesão da margem, tamanho das células epiteliais únicas, núcleos, cromatina branda, nucléolos normais, mitoses e o rótulo de classe (valor 2 para benígno e 4 para malígno). Com exceção do rótulo de classe, todas as variáveis são numéricas inteiras na faixa de 1 a 10, mapeando a informação contida.

## Glass Identification

O conjunto de dados *Glass Identification* foi coletado por B. German, associado ao *Central Research Establishment*, *Home Office Forensic Science Service* e doado ao repositório por Vina Spiehler, associada a *Diagnostic Products Corporation*, em 1 de Setembro de 1987. Essa base de dados contém informações sobre seis tipos de vidros definidos em termos da porcentagem de óxidos em seus conteúdos, especificamente: sódio, magnésio, alumínio, silicone, potássio, cálcio, bário e ferro. Os tipos de vidro, representados no rótulo de classe das amostras, referem-se à sua origem: janelas de edifícios (processados), janelas de edifícios (não processados), janelas de veículos (processados), janelas de veículos (não processados), recipientes, talheres e lamparinas. A base de dados possui um total de 214 amostras de vidro.

Para este trabalho, os rótulos de classe das amostras foram modificados. Amostras de vidro oriundas de veículos (processadas e não processadas) foram associadas ao rótulo de classe ("*vehicle*"), enquanto os demais tipos foram unificados no rótulo de classe "outro"("*other*"). Desta forma, criou-se uma base de dados com 214 amostras, sendo 146 amostras da classe *window* e as demais 68 pertencentes à classe *other*.

## Wine Quality

A base de dados *Wine Quality* foi coletado por Paulo Cortez, associado à Universidade de Minho, e doada ao repositório em 7 de Outubro de 2009 após a publicação de sua pesquisa [20]. O objetivo da coleta dos dados foi modelar a qualidade do vinho verde produzido em Portugal baseado em testes fisicoquímicos. O rótulo de classe, *quality*, é um valor inteiro entre 0 e 10 que representa a nota associada a cada amostra de vinho, mapeando a sua qualidade. As variáveis descritoras das amostras são suas propriedades fisicoquímicas, especificamente acidez fixa, acidez volátil, níveis de

ácido cítrico, açúcar residual, cloridos, dióxidos de enxofre livres, total de dióxidos de enxofre, densidade, pH, sulfatos e álcool. A base reúne amostras de vinho verde em suas versões tinto e branco. Para este trabalho, foram consideradas apenas amostras de vinho tinto (total de 1599 amostras). Além disso, os rótulos de classe foram modificados de maneira a tornar a variável de classe binária. Amostras de vinho que receberam nota 7 e 8 (as maiores notas registradas nos dados originais) receberam rótulo *good*, enquanto as demais amostras receberam rótulo *other*. Desta forma, resultou-se em um conjunto de dados com 12 variáveis, 217 amostras de vinho de classe *good* e 1382 amostras de classe *other*.

## Yeast

O conjunto de dados *Yeast* foi coletado e mantido por Kenta Nakai, associado ao Instituto de Biologia Molecular e Celular da Universidade de Osaka, durante o desenvolvimento de sua pesquisa envolvendo a predição de localizações de proteínas em células [51]. A base reúne 1484 amostras de proteínas, cada uma associada a um rótulo de classe correspondente à sua localização na célula, que pode ser: citólica ou citoesqueletal (CYT), nuclear (NUC), mitocondrial (MIT), membrana sem sinal N-terminal (ME3), membrana sem sinal de clivagem (ME2), membrana com sinal de clivagem (ME1), extracelular (EXC), vacuolar (VAC), peroxissomal (POX) e endoplásmica (ERL). As variáveis que descrevem as amostras representam: número de sequência, método de McGeoch para reconhecimento da sequência de sinal, método de von Heijne para reconhecimento da sequência de sinal, pontuação obtida no programa de predição da região de abrangência da membrana ALOM, pontuação obtida na análise discriminante do conteúdo de aminoácidos da região N-terminal (20 resíduos de comprimento) de proteínas mitocondriais e não mitocondriais, presença de cadeia "HDEL", sinal peroxissomal no C-terminus, pontuação obtida na análise discriminante do conteúdo de aminoácidos em proteínas vacuolares e extracelulares, e a pontuação obtida na análise discriminante da localização nuclear de proteínas nucleares e não nucleares.

Pàra este trabalho, os rótulos de classe das amostras foram modificados de maneira a tornar a variável da classe binária. Os rótulos de classe ME1, ME2 e ME3 foram subtituídos por apenas *membrane*, enquanto os demais rótulos foram susbtituídos por *other*. Desta forma, resultou-se em um conjunto de dados com 9 variáveis (desconsiderando a variável de identificação das amostras, que foi descartada durante os testes), 258 amostras de proteína da classe *membrane* e 1226 amostras da classe *other*.

## Pima Indians Diabetes

O conjunto de dados *Pima Indians Diabetes* foi coletado pelo Instituto Nacional de Diabetes e Doenças Digestivas e de Rins, EUA, e tem como objetivo diagnosticar a presença de diabetes em mulheres indígenas do povo Pima com base nas diversas variáves descritoras contidas na base. Tais variáveis incluem: número de gestações passadas, concentração plasmática de glicose obtida após teste oral de tolerância a glucose, pressão arterial, espessura da pele, quantidade de insulina, índice de massa corpórea, função de pedigree de diabetes, idade, e finalmente a presença (1) ou ausência (0) de diabetes como rótulo de classe. A base de dados possui 768 amostras de mulheres indígenas avaliadas, sendo 268 amostras diabéticas e as 500 amostras restantes sem diabetes registrada.

## Ecoli

O conjunto de dados *Ecoli* também foi coletado por Kenta Nakai e utilizado em sua pesquisa envolvendo predição de localizações de proteínas em células [51]. Possui 336 amostras de proteínas, cada uma associada a um rótulo de classe correspondente à sua localização na célula, que pode ser: citoplasma (cp), membrana interna sem sequência de sinal (pp), membrana interna com sequência de sinal não clivável (imU), membrana interna com sequência de sinal clivável (imS), membrana externa (om), membrana externa lipoproteína (omL) ou membrana interna lipoproteína (imL). Pàra este trabalho, os rótulos de classe das amostras foram modificados de maneira a tornar a variável da classe binária. Os rótulos de classe "imU"foram mantidos, enquanto os demais foram subtituídos por *other*. Desta forma, resultou-se em um conjunto de dados com 8 variáveis (desconsiderando a variável de identificação das amostras, que foi descartada durante os testes), 35 amostras de proteína da classe *imU* e 301 amostras da classe *other*.

## New Thyroid

A base de dados *New Thyroid* foi coletada por Danny Coomans, associado ao Departamento de Matemática e Estatística da Uiversidade James Cook, e doada ao repositório UCL por Stefan Aeberhard, associado ao Departamento de Ciências da Computação da mesma universidade. Foi utilizada por Coomans em seu trabalho no qual foram comparadas 16 técnicas diferentes de discriminação para a predição do estado da glândula tireóide [18]. A base reúne dados de 215 pacientes associados a rótulos de classe que mapeiam a presença de hipertireoidismo, hipotireoidismo ou nenhum em tais pacientes. As 5 variáveis analisadas incluem resultados de teste de captação de T3 em resina, tiroxina sérica total, triiodotironina sérica total, nível basal de hormônio estimulador da tireoide (TSH), diferença absoluta máxima do valor do TSH após a injeção

de 200 microgramas de hormônio liberador de tireotropina em comparação ao valor basal, e o rótulo de classe (normal para pacientes normais, *hyper* para pacientes com hipertireoidismo e *hypo* para pacientes com hipotireoidismo).

Para este trabalho, foram mantidos os rótulos de classe *hyper* enquanto os demais foram substituídos por *other*, dessa forma resultando em um conjunto de dados com 30 amostras da classe *hypo* e 185 amostras da classe *other*.

## 5.3 Configurações do ambiente de teste

Os testes foram conduzidos em um computador pessoal do tipo *desktop* com as seguintes configurações:

- Processador: Intel® Core™ i5-8400 @ 2.80 GHz
- Memória RAM: 16GB DDR4 2400MHz (2 x 8GB 2400MHz DDR4, dual channel)
- Placa-mãe: ASRock H310M-DGS
- Placa de vídeo: NVIDIA GeForce GTX 1060 6GB GDDR5
- Fonte: Corsair CX430 430W
- Disco rígido 1: Solid State Drive Samsung 850 EVO (250GB)
- Disco rígido 2: Hard Disk Drive ST3500413AS (500GB)
- Sistema operacional: Microsoft Windows 10 Pro
- Monitor AOC 24’ G2460P

Alternativamente utilizou-se também um computador pessoal do tipo *notebook* com as seguintes configurações:

- Processador: Intel® Core™ i5-
- Memória RAM: 4GB DDR3
- Placa-mãe:
- Disco rígido 1: Hard Disk Drive (500GB)
- Sistema operacional: Microsoft Windows 7 Pro

## 5.4 Resultados obtidos

Nas subseções seguintes, estão detalhados os resultados quantitativos obtidos pelo algoritmo PCA-SMOTE e os demais algoritmos de balanceamento escolhidos para comparação aplicados às base de dados de teste selecionadas. Cada subseção resume os resultados obtidos por uma base de dados diferente. Para cada base de dados, está disponibilizada uma tabela que mostra as métricas de desempenho (acurácia, sensitividade, especificidade, valor AUC, *G-mean* e *F-measure*) computadas para os modelos SVM,

C4.5 e MLP construídos a partir da base de dados sem balanceamento (-) e balanceada de acordo com o SMOTE tradicional, o *Safe-Level-SMOTE* (SLSMOTE), *Borderline-SMOTE* (B-SMOTE), ADASYN e o algoritmo proposto PCA-SMOTE (Tabelas 5.1, 5.3, 5.5, 5.7, 5.9, 5.11, 5.13, 5.15, 5.17, 5.19, 5.21 e 5.23).

Para facilitar a comparação do desempenho do PCA-SMOTE com o dos demais algoritmos, estão marcados de vermelho todos os valores de desempenho obtidos pelos algoritmos que foram inferiores aos valores obtidos pelo PCA-SMOTE para cada modelo de classificação. Além disso, para atestar os resultados obtidos para cada base de dados, foram disponibilizados os valores obtidos pelos testes de significância pareados de Wilcoxon aplicados sobre as médias de desempenho obtidas entre pares de modelos/algoritmos (Tabelas 5.2, 5.4, 5.6, 5.8, 5.10, 5.12, 5.14, 5.16, 5.18, 5.20, 5.22 e 5.24).

Finalmente, também estão inclusos os gráficos que mostram as distribuições das amostras das bases de dados, expressas em termos de suas duas componentes principais extraídas pelo PCA, antes e após a aplicação do PCA-SMOTE (Figuras 5.1, 5.2, 5.3, 5.4, 5.5, 5.6, 5.7, 5.8, 5.9, 5.10, 5.11, 5.12).

### 5.4.1 Breast Cancer Coimbra

| Modelo | Medida | - | SMOTE | SLSMOTE | B-SMOTE | ADASYN | PCA-SMOTE |
|---|---|---|---|---|---|---|---|
| SVM | Acurácia (%) | 77.43 | **88.74** | 86.19 | 85.75 | 88.1 | 88.57 |
| | Sensitividade (%) | 17.4 | 95.96 | 89.12 | 89.99 | 93.68 | **95.81** |
| | Especificidade (%) | **99.04** | 81.01 | 83 | 81.21 | 82.68 | 80.78 |
| | AUC (%) | 0.58 | 0.88 | 0.86 | 0.86 | 0.88 | 0.88 |
| | G-mean (%) | 0.31 | 0.88 | 0.86 | 0.85 | 0.88 | 0.88 |
| | F-measure (%) | 0.25 | 0.9 | 0.87 | 0.87 | 0.89 | 0.9 |
| C4.5 | Acurácia (%) | 75.47 | 83.55 | 78.04 | 77.84 | 81.12 | **83.89** |
| | Sensitividade (%) | 48.2 | 86.97 | 82.88 | 80.81 | 85.83 | **87.68** |
| | Especificidade (%) | **85.59** | 79.88 | 72.82 | 74.74 | 76.59 | 79.85 |
| | AUC (%) | 0.67 | 0.83 | 0.78 | 0.78 | 0.81 | **0.84** |
| | G-mean (%) | 0.6 | 0.83 | 0.77 | 0.77 | 0.81 | 0.83 |
| | F-measure (%) | 0.49 | 0.85 | 0.8 | 0.79 | 0.82 | 0.85 |
| MLP | Acurácia (%) | 74.91 | 85.98 | **86.4** | 85.19 | 84.21 | 84.52 |
| | Sensitividade (%) | 48.1 | 90.2 | **92.33** | 89.48 | 87.99 | 90.15 |
| | Especificidade (%) | **84.54** | 81.29 | 79.94 | 80.46 | 80.63 | 78.54 |
| | AUC (%) | 0.66 | **0.86** | **0.86** | 0.85 | 0.84 | 0.84 |
| | G-mean (%) | 0.58 | 0.85 | **0.86** | 0.84 | 0.84 | 0.84 |
| | F-measure (%) | 0.47 | 0.87 | **0.88** | 0.86 | 0.84 | 0.86 |

**Tabela 5.1:** *Métricas de desempenho calculadas para os modelos treinados com a base de dados* ***Breast Cancer Coimbra*** *sem balanceamento e balanceada de acordo com os algoritmos.*

| | Métodos | AUC | G-mean | F-measure |
|---|---|---|---|---|
| $p_{wilcoxon}$ SVM | PCA-SMOTE vs - | 0.001 | 0.001 | 0.001 |
| | PCA-SMOTE vs SMOTE | 0.6875 | 0.6875 | 0.6152 |
| | PCA-SMOTE vs SLSMOTE | 0.0137 | 0.0244 | 0.0029 |
| | PCA-SMOTE vs B-SMOTE | 0.0049 | 0.0029 | 0.0029 |
| | PCA-SMOTE vs ADASYN | 0.5 | 0.5391 | 0.0322 |
| $p_{wilcoxon}$ C4.5 | PCA-SMOTE vs - | 0.001 | 0.001 | 0.001 |
| | PCA-SMOTE vs SMOTE | 0.3994 | 0.3848 | 0.3848 |
| | PCA-SMOTE vs SLSMOTE | 0.001 | 0.0029 | 0.001 |
| | PCA-SMOTE vs BMOTE | 0.001 | 0.001 | 0.001 |
| | PCA-SMOTE vs ADASYN | 0.0098 | 0.0137 | 0.002 |
| $p_{wilcoxon}$ MLP | PCA-SMOTE vs - | 0.001 | 0.001 | 0.001 |
| | PCA-SMOTE vs SMOTE | 0.9473 | 0.958 | 0.9199 |
| | PCA-SMOTE vs SLSMOTE | 0.9932 | 0.9932 | 0.998 |
| | PCA-SMOTE vs BMOTE | 0.8389 | 0.8389 | 0.7539 |
| | PCA-SMOTE vs ADASYN | 0.3125 | 0.4229 | 0.0801 |

**Tabela 5.2:** *Testes pareado de Wilcoxon realizados sobre as métricas de desempenho obtidas pelos modelos.*

O conjunto de dados *Breast Cancer Coimbra* adaptado para o teste possui dimensões $87 \times 10$, sendo 23 amostras pertencentes à classe minoritária "1"(pacientes saudáveis) e 64 amostras pertencentes à classe "2"(pacientes com câncer), configurando desta forma um fator de desbalanceamento de aproximadamente 1:2. Ao todo, 29 amostras de pacientes saudáveis foram removidas aleatoriamente do conjunto de dados para criar o desbalanceamento. Os algoritmos de balanceamento incluíram 46 (PCA-SMOTE, SMOTE tradicional, *Borderline-SMOTE* e SLSMOTE) e 39 (ADASYN) amostras sintéticas para a classe "1".

De acordo com os testes de significância pareados de Wilcoxon expostos na Tabela 5.2, o algoritmo PCA-SMOTE apresentou em geral valores de desempenho

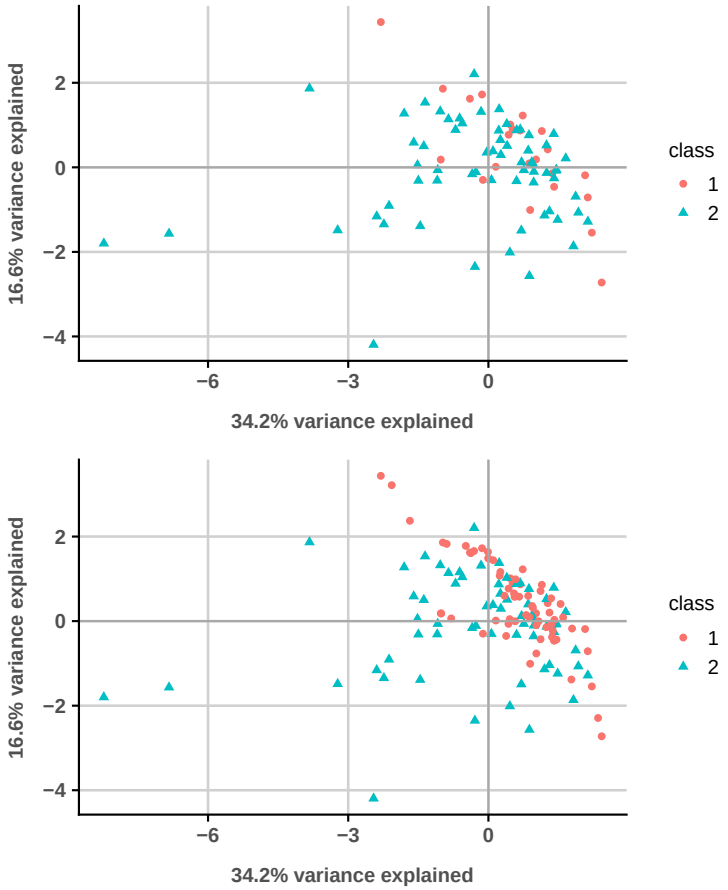



**Figura 5.1:** *Base de dados* ***Breast Cancer Coimbra*** *expressa em termos de suas duas componentes principais antes e após a aplicação do PCA-SMOTE.*

melhores do que os demais algoritmos de balanceamento para os modelos SVM e C4.5 avaliados pelas três métricas AUC, *G-mean* e *F-measure* (valor de $p$ máximo obtido de 0.0322), com exceção do SMOTE básico e do ADASYN combinado ao modelo SVM avaliado pelas métricas AUC e *G-mean*. Já para o modelo MLP, estima-se pelos valores na Tabela 5.1 que o melhor algoritmo de balanceamento investigado tenha sido o *Safe-Level-SMOTE*, que apresentou para todas as métricas de desempenho (exceto especificidade) valores maiores ou iguais do que os demais algoritmos.

Conclui-se desta forma que o PCA-SMOTE foi de fato o algoritmo de balanceamento que trouxe os melhores resultados para o aprendizado dos modelos SVM e C4.5 sobre a base de dados *Breast Cancer Coimbra*. Já o modelo MLP obtém melhores resultados quando combinado ao algoritmo *Safe-Level-SMOTE* para balancear as amostras da base.

### 5.4.2 Arroz

| Modelo | Medida | - | SMOTE | SLSMOTE | B-SMOTE | ADASYN | PCA-SMOTE |
|---|---|---|---|---|---|---|---|
| SVM | Acurácia (%) | 88.77 | 95.14 | 91.18 | 89.29 | **96.23** | 90.54 |
| | Sensitividade (%) | 65 | 92.5 | 82.5 | 79.5 | **95.3** | 81.67 |
| | Especificidade (%) | 99.5 | 98 | 99.5 | 99 | 97.33 | **100** |
| | AUC (%) | 0.82 | 0.95 | 0.91 | 0.89 | **0.96** | 0.91 |
| | G-mean (%) | 0.7 | 0.94 | 0.89 | 0.87 | **0.96** | 0.9 |
| | F-measure (%) | 0.69 | 0.94 | 0.88 | 0.86 | **0.96** | 0.88 |
| C4.5 | Acurácia (%) | 80.57 | 83.71 | 84.17 | 77.55 | **86.48** | 81.6 |
| | Sensitividade (%) | 68 | 82 | 83.17 | 77.33 | **88.2** | 80 |
| | Especificidade (%) | 84 | **85.5** | 85.17 | 77.83 | 84.83 | 82.83 |
| | AUC (%) | 0.77 | 0.84 | 0.84 | 0.78 | **0.87** | 0.81 |
| | G-mean (%) | 0.65 | 0.83 | 0.82 | 0.76 | **0.85** | 0.8 |
| | F-measure (%) | 0.61 | 0.83 | 0.83 | 0.77 | **0.87** | 0.81 |
| MLP | Acurácia (%) | 86.07 | 92.86 | 90 | **93** | 92.6 | 92.74 |
| | Sensitividade (%) | 70 | **97.5** | 89.67 | 96 | 96.6 | 93 |
| | Especificidade (%) | **93.83** | 88 | 90.5 | 90.17 | 88.5 | 92.67 |
| | AUC (%) | 0.82 | 0.93 | 0.9 | 0.93 | 0.93 | 0.93 |
| | G-mean (%) | 0.76 | 0.92 | 0.89 | 0.92 | 0.92 | 0.92 |
| | F-measure (%) | 0.71 | **0.94** | 0.89 | 0.93 | 0.93 | 0.93 |

**Tabela 5.3:** *Métricas de desempenho calculadas para os modelos treinados com a base de dados Arroz sem balanceamento e balanceada de acordo com os algoritmos.*

| | Métodos | AUC | G-mean | F-measure |
|---|---|---|---|---|
| $p_{wilcoxon}$ SVM | PCA-SMOTE vs - | 0.001 | 0.0029 | 0.0029 |
| | PCA-SMOTE vs SMOTE | 0.9955 | 0.9955 | 0.9946 |
| | PCA-SMOTE vs SLSMOTE | 0.5 | 0.5472 | 0.5 |
| | PCA-SMOTE vs BMOTE | 0.0462 | 0.0654 | 0.0654 |
| | PCA-SMOTE vs ADASYN | 0.9979 | 0.9979 | 0.9979 |
| $p_{wilcoxon}$ C4.5 | PCA-SMOTE vs - | 0.0654 | 0.0029 | 0.002 |
| | PCA-SMOTE vs SMOTE | 0.9157 | 0.9346 | 0.8838 |
| | PCA-SMOTE vs SLSMOTE | 0.9756 | 0.8838 | 0.8125 |
| | PCA-SMOTE vs BMOTE | 0.1377 | 0.0967 | 0.0654 |
| | PCA-SMOTE vs ADASYN | 0.9968 | 1 | 1 |
| $p_{wilcoxon}$ MLP | PCA-SMOTE vs - | 0.0029 | 0.0029 | 0.001 |
| | PCA-SMOTE vs SMOTE | 0.7232 | 0.7234 | 0.9382 |
| | PCA-SMOTE vs SLSMOTE | 0.117 | 0.0776 | 0.0378 |
| | PCA-SMOTE vs BMOTE | 0.5407 | 0.5 | 0.6523 |
| | PCA-SMOTE vs ADASYN | 0.3176 | 0.2968 | 0.6166 |

**Tabela 5.4:** *Testes pareado de Wilcoxon realizados sobre as métricas de desempenho obtidas pelos modelos.*

O conjunto de dados Arroz adaptado para o teste possui dimensões $27 \times 21$, sendo 8 amostras pertencentes à classe minoritária "GO"(amostras de arroz obtidas do estado de Goiás) e 19 amostras pertencentes à classe "RS"(amostras de arroz obtidas do estado do Rio Grande do Sul), configurando desta forma um fator de desbalanceamento de aproximadamente 1:2,3. Os algoritmos de balanceamento incluíram 11 (PCA-SMOTE), 12 (SMOTE tradicional), 10 (*Borderline-SMOTE* e SLSMOTE) e 13 (ADASYN) amostras sintéticas para a classe "GO".

A distribuição das amostras em termos das duas primeiras componentes principais extraídas pelo PCA pode ser vista na Fig. 5.2. Embora não haja sobreposição de amostras das duas classes, a base de dados sofre de baixa densidade de dados, o que pode

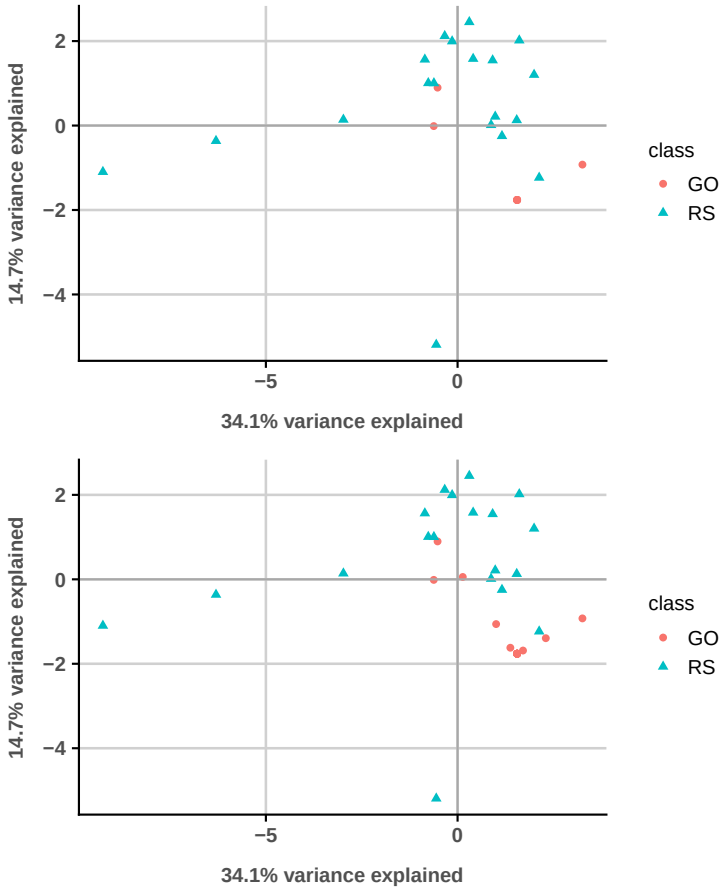



**Figura 5.2:** *Base de dados arroz expressa em termos de suas duas componentes principais antes e após a aplicação do PCA-SMOTE.*

prejudicar a capacidade de generalização dos modelos de classificação para amostras novas.

Para esta base de teste em particular, o algoritmo de balanceamento que teve o melhor desempenho geral foi o ADASYN, perdendo apenas em termos de F-measure para o SMOTE tradicional combinado com o modelo MLP. Embora o algoritmo proposto não esteja entre os melhores para este conjunto de dados em específico, é possível perceber através dos testes de significância expostos na Tabela 5.4 que o PCA-SMOTE ainda apresentou desempenho melhor do que o a base de dados desbalanceada, para todos os modelos de classificação (valor de $p$ entre 0.001 e 0.0654), e também melhor do que o *Borderline-SMOTE*, para o modelo SVM e para o modelo C4.5 quando avaliado via *G-mean* (valor de $p$ igual a 0.0967) e *F-measure* (valor de $p$ igual a 0.0654). Além disso, o algoritmo PCA-SMOTE obteve valores aparentemente melhores do que o **Safe-Level-SMOTE** quando combinado aos modelos SVM e C4.5. Os valores de especificidade obtidos pelo PCA-SMOTE combinado com o modelo MLP também foram melhores do que os de todos os demais algoritmos quando combinados com este mesmo modelo.

### 5.4.3 Chocolate orgânico

| Modelo | Medida | - | SMOTE | SLSMOTE | B-SMOTE | ADASYN | PCA-SMOTE |
|---|---|---|---|---|---|---|---|
| SVM | Acurácia (%) | 69.48 | 85.05 | 83.82 | 85.66 | 84.57 | **87.01** |
| | Sensitividade (%) | 36.67 | 86.8 | 81.5 | 86.7 | 84.7 | **89.9** |
| | Especificidade (%) | **86.8** | 83.4 | 86.1 | 84.8 | 84.2 | 84.4 |
| | AUC (%) | 0.62 | 0.85 | 0.84 | 0.86 | 0.84 | **0.87** |
| | G-mean (%) | 0.41 | 0.84 | 0.82 | 0.85 | 0.84 | **0.86** |
| | F-measure (%) | 0.36 | 0.85 | 0.82 | 0.86 | 0.83 | **0.87** |
| C4.5 | Acurácia (%) | 69.39 | 78.46 | 77.62 | 79.54 | 75.29 | **81.84** |
| | Sensitividade (%) | 54 | 82.9 | 83.7 | 84.4 | 80.4 | **84.6** |
| | Especificidade (%) | 77.9 | 74.2 | 71.7 | 74.9 | 70.9 | **78.9** |
| | AUC (%) | 0.67 | 0.79 | 0.78 | 0.8 | 0.76 | **0.82** |
| | G-mean (%) | 0.55 | 0.77 | 0.76 | 0.78 | 0.74 | **0.8** |
| | F-measure (%) | 0.49 | 0.78 | 0.78 | 0.8 | 0.75 | **0.82** |
| MLP | Acurácia (%) | 79.88 | 89.32 | 87.79 | 89.91 | 85.96 | **90.25** |
| | Sensitividade (%) | 76 | 97.6 | 94.9 | **98** | 91.5 | 97.2 |
| | Especificidade (%) | 82.2 | 81.2 | 80.5 | 81.9 | 80.9 | **83.7** |
| | AUC (%) | 0.79 | 0.89 | 0.88 | 0.9 | 0.86 | 0.9 |
| | G-mean (%) | 0.76 | 0.89 | 0.86 | 0.89 | 0.85 | **0.9** |
| | F-measure (%) | 0.71 | 0.9 | 0.89 | 0.91 | 0.86 | 0.91 |

**Tabela 5.5:** *Métricas de desempenho calculadas para os modelos treinados com a base de dados de chocolate orgânico sem balanceamento e balanceada de acordo com os algoritmos.*

| | Métodos | AUC | G-mean | F-measure |
|---|---|---|---|---|
| $p_{wilcoxon}$ SVM | PCA-SMOTE vs - | 0.001 | 0.001 | 0.001 |
| | PCA-SMOTE vs SMOTE | 0.0801 | 0.0527 | 0.0967 |
| | PCA-SMOTE vs SLSMOTE | 0.0322 | 0.0378 | 0.0244 |
| | PCA-SMOTE vs BMOTE | 0.1565 | 0.1377 | 0.1377 |
| | PCA-SMOTE vs ADASYN | 0.0294 | 0.0322 | 0.0244 |
| $p_{wilcoxon}$ C4.5 | PCA-SMOTE vs - | 0.001 | 0.001 | 0.001 |
| | PCA-SMOTE vs SMOTE | 0.0767 | 0.0527 | 0.1162 |
| | PCA-SMOTE vs SLSMOTE | 0.0527 | 0.0322 | 0.0654 |
| | PCA-SMOTE vs BMOTE | 0.1377 | 0.1377 | 0.2783 |
| | PCA-SMOTE vs ADASYN | 0.0137 | 0.0137 | 0.0098 |
| $p_{wilcoxon}$ MLP | PCA-SMOTE vs - | 0.001 | 0.001 | 0.001 |
| | PCA-SMOTE vs SMOTE | 0.2204 | 0.1539 | 0.2158 |
| | PCA-SMOTE vs SLSMOTE | 0.002 | 0.001 | 0.0098 |
| | PCA-SMOTE vs BMOTE | 0.2703 | 0.2376 | 0.3848 |
| | PCA-SMOTE vs ADASYN | 0.0029 | 0.001 | 0.001 |

**Tabela 5.6:** *Testes pareado de Wilcoxon realizados sobre as métricas de desempenho obtidas pelos modelos.*

O conjunto de dados do chocolate orgânico possui dimensões $36 \times 40$ e fator de desbalanceamento 1:2, sendo 12 amostras da classe "organic"(amostras de chocolate orgânico) e 24 amostras pertencentes à classe "conventional"(amostras de chocolate comum). Todos os algoritmos de balanceamento incluíram 12 amostras para a classe "organic"no conjunto de dados, exceto o ADASYN, que gerou 9 amostras sintéticas. A distribuição das amostras geradas pelo PCA-SMOTE em termos das duas componentes principais pode ser vista na Fig. 5.2.

Esta base de dados tem como principal característica a esparsidade, visível na Fig. 5.3, que mostra as amostras de dados em um gráfico bidimensional expresso em termos das duas primeiras componentes principais computadas pelo PCA. Embora as

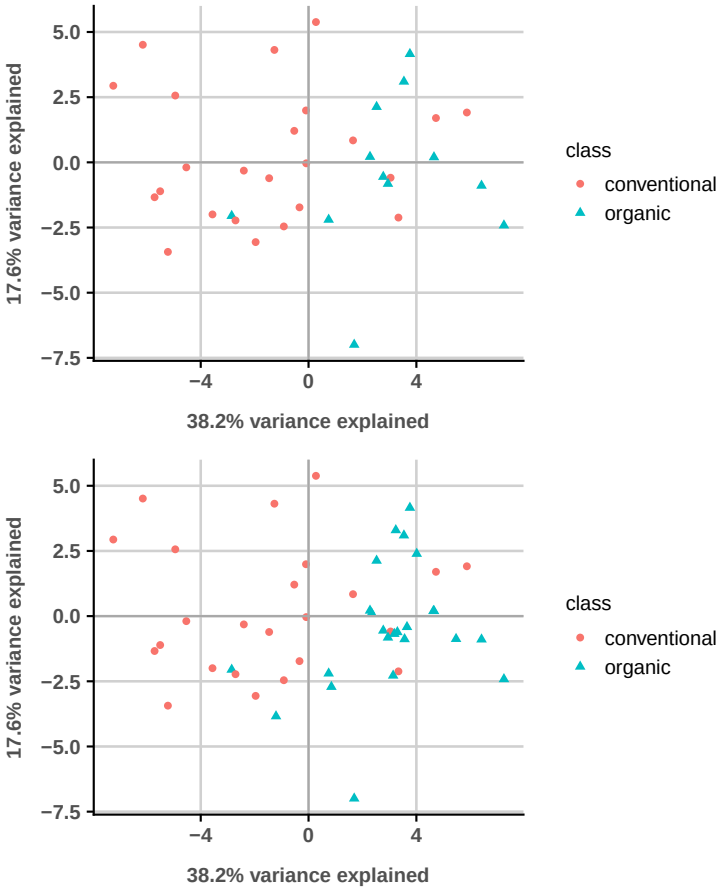



**Figura 5.3:** *Base de dados de chocolate orgânico expressa em termos de suas duas componentes principais antes e após a aplicação do PCA-SMOTE.*

amostras estejam espalhadas por toda a região visível do gráfico, as amostras da classe *"organic"* se concentram na parte direita do gráfico, enquanto as amostras da classe *"conventional"* se concentram primariamente na região esquerda.

Como evidenciado nas Tabelas 5.5 e 5.6, o algoritmo PCA-SMOTE foi o algoritmo de balanceamento que teve o melhor desempenho em combinação com os três modelos de classificação avaliados, para praticamente todas as métricas consideradas. É derrotado apenas pelo *Borderline-SMOTE* (conclusão visível apenas através dos testes de significância de Wilcoxon na Tabela 5.6, uma vez que a Tabela 5.5 mostra valores médios de métricas de desempenho levemente inferiores aos obtidos pelo PCA-SMOTE), e também pelo SMOTE básico combinado com o modelo MLP.

### 5.4.4 Folhas de laranjeiras orgânicas

| Modelo | Medida | - | SMOTE | SLSMOTE | B-SMOTE | ADASYN | PCA-SMOTE |
|---|---|---|---|---|---|---|---|
| SVM | Acurácia (%) | 88.14 | **94.61** | 92.23 | 93.06 | 91.26 | 92.53 |
| | Sensitividade (%) | 64 | **98.47** | 91.2 | 95.6 | 87.67 | 94.8 |
| | Especificidade (%) | **98.4** | 90.5 | 93.2 | 90.4 | 94.4 | 90.2 |
| | AUC (%) | 0.81 | **0.94** | 0.92 | 0.93 | 0.91 | 0.92 |
| | G-mean (%) | 0.76 | **0.94** | 0.92 | 0.92 | 0.9 | 0.92 |
| | F-measure (%) | 0.73 | **0.95** | 0.92 | 0.94 | 0.89 | 0.93 |
| C4.5 | Acurácia (%) | 93.52 | 90.67 | **95.65** | 94.57 | 95.37 | 91.2 |
| | Sensitividade (%) | 80 | 91.2 | 91.4 | **97.67** | 89 | 86.4 |
| | Especificidade (%) | 99.2 | 90.2 | **100** | 91.1 | **100** | 96.4 |
| | AUC (%) | 0.9 | 0.91 | **0.96** | 0.94 | 0.94 | 0.91 |
| | G-mean (%) | 0.85 | 0.9 | **0.95** | 0.94 | 0.94 | 0.91 |
| | F-measure (%) | 0.84 | 0.91 | **0.95** | **0.95** | 0.93 | 0.9 |
| MLP | Acurácia (%) | 81.62 | 87.13 | 86.36 | 90.38 | 84.01 | **90.56** |
| | Sensitividade (%) | 60 | 88.87 | 88.33 | 93.47 | 78.67 | **93.6** |
| | Especificidade (%) | 90.5 | 85.3 | 84.3 | 86.9 | 87.3 | **87.5** |
| | AUC (%) | 0.75 | 0.87 | 0.86 | 0.9 | 0.83 | **0.91** |
| | G-mean (%) | 0.67 | 0.86 | 0.86 | 0.9 | 0.81 | **0.9** |
| | F-measure (%) | 0.61 | 0.88 | 0.87 | 0.91 | 0.79 | **0.91** |

**Tabela 5.7:** *Métricas de desempenho calculadas para os modelos treinados com a base de dados de folhas de laranjeiras orgânicas sem balanceamento e balanceada de acordo com os algoritmos.*

| | Métodos | AUC | G-mean | F-measure |
|---|---|---|---|---|
| $p_{wilcoxon}$ SVM | PCA-SMOTE vs - | 0.0029 | 0.001 | 0.001 |
| | PCA-SMOTE vs SMOTE | 0.9793 | 0.9814 | 0.9814 |
| | PCA-SMOTE vs SLSMOTE | 0.4594 | 0.5406 | 0.1875 |
| | PCA-SMOTE vs BMOTE | 0.7428 | 0.6875 | 0.7842 |
| | PCA-SMOTE vs ADASYN | 0.0767 | 0.0654 | 0.0049 |
| $p_{wilcoxon}$ C4.5 | PCA-SMOTE vs - | 0.1465 | 0.0124 | 0.0124 |
| | PCA-SMOTE vs SMOTE | 0.1177 | 0.1377 | 0.8125 |
| | PCA-SMOTE vs SLSMOTE | 0.9979 | 1 | 1 |
| | PCA-SMOTE vs BMOTE | 0.9955 | 0.9951 | 1 |
| | PCA-SMOTE vs ADASYN | 0.9979 | 0.9979 | 0.9968 |
| $p_{wilcoxon}$ MLP | PCA-SMOTE vs - | 0.0029 | 0.001 | 0.001 |
| | PCA-SMOTE vs SMOTE | 0.0089 | 0.0137 | 0.0322 |
| | PCA-SMOTE vs SLSMOTE | 0.0029 | 0.001 | 0.001 |
| | PCA-SMOTE vs BMOTE | 0.5771 | 0.5391 | 0.6875 |
| | PCA-SMOTE vs ADASYN | 0.001 | 0.001 | 0.001 |

**Tabela 5.8:** *Testes pareado de Wilcoxon realizados sobre as métricas de desempenho obtidas pelos modelos.*

A base de dados das folhas de laranjeiras orgânicas possui dimensões $34 \times 16$, onde 10 amostras são da classe "organic"(amostras de folhas de laranjeiras orgânicas) e 24 amostras pertencentes à classe "conventional"(amostras de folhas de laranjeiras cultivadas em sistema tradicional), configurando desta forma um fator de desbalanceamento de aproximadamente 1:2,4. O algoritmo PCA-SMOTE incluiu 15 amostras sintéticas da classe "organic"no conjunto de dados, enquanto os demais algoritmos incluíram 16 amostras, com exceção do ADASYN, que incluiu 7 amostras sintéticas.

Para esta base de teste, o algoritmo PCA-SMOTE apresentou desempenho superior aos demais algoritmos de balanceamento quando combinado com o modelo MLP, com exceção do *Borderline-SMOTE*. Quando combinado ao SVM, o algoritmo

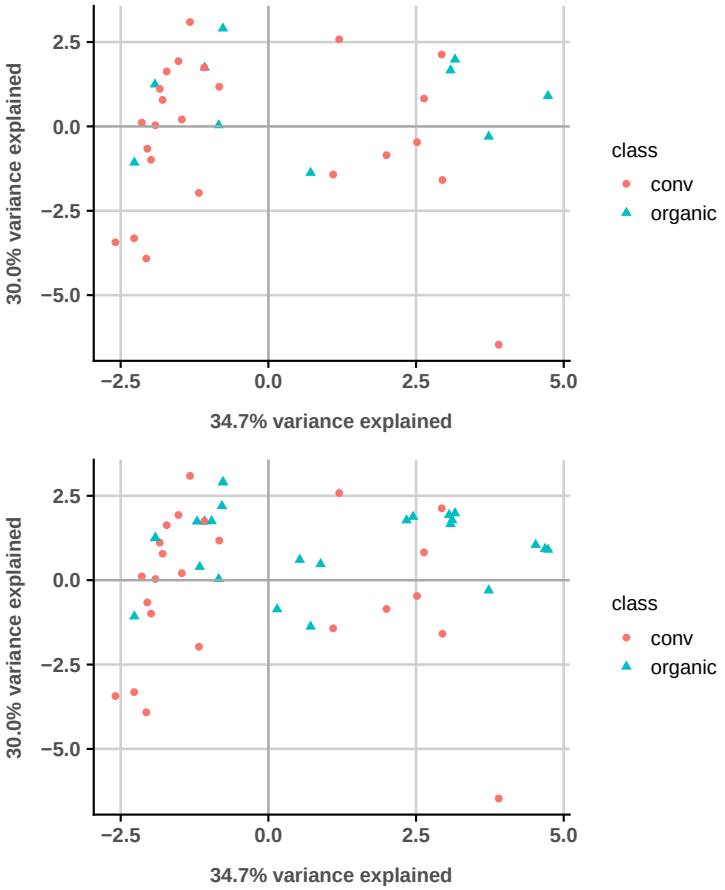



**Figura 5.4:** *Base de dados de folhas de laranjeiras orgânicas expressa em termos de suas duas componentes principais antes e após a aplicação do PCA-SMOTE.*

proposto também apresenta acurácia, sensitividade, AUC, *G-mean* e *F-measure* melhores do que os obtidos pelo ADASYN, e valores de acurácia e sensitividade melhores do que aqueles obtidos pelo *Safe-Level-SMOTE*. Para os demais casos, o desempenho do algoritmo mostrou-se inferior ou similar aos dos demais algoritmos.

### 5.4.5 Suco de uva orgânico

| Modelo | Medida | - | SMOTE | SLSMOTE | B-SMOTE | ADASYN | PCA-SMOTE |
|---|---|---|---|---|---|---|---|
| SVM | Acurácia (%) | 70.83 | 91.67 | **95.71** | 91.21 | 89.96 | 93.05 |
| | Sensitividade (%) | 14 | 91.83 | **96.83** | 91.17 | 90 | 93.83 |
| | Especificidade (%) | **98.83** | 91.5 | 94.67 | 92 | 89.83 | 92.67 |
| | AUC (%) | 0.56 | 0.92 | **0.96** | 0.92 | 0.9 | 0.93 |
| | G-mean (%) | 0.19 | 0.91 | **0.95** | 0.9 | 0.89 | 0.91 |
| | F-measure (%) | 0.18 | 0.91 | **0.96** | 0.9 | 0.89 | 0.92 |
| C4.5 | Acurácia (%) | 70.13 | 80.24 | 79.99 | **81.95** | 81.61 | 81.3 |
| | Sensitividade (%) | 51 | 84.83 | 89.17 | 87.5 | **91.83** | 88.83 |
| | Especificidade (%) | **79.33** | 76.17 | 70.5 | 76.5 | 72.5 | 74.17 |
| | AUC (%) | 0.65 | 0.8 | 0.8 | 0.82 | 0.82 | 0.82 |
| | G-mean (%) | 0.5 | 0.78 | 0.76 | 0.79 | 0.79 | 0.79 |
| | F-measure (%) | 0.45 | 0.81 | 0.82 | 0.82 | **0.83** | 0.82 |
| MLP | Acurácia (%) | 84.33 | 91.64 | 91.4 | 91.73 | 91.96 | **93.58** |
| | Sensitividade (%) | 74 | 99.5 | 98.83 | 100 | 99.33 | 100 |
| | Especificidade (%) | **89.33** | 83.67 | 83.67 | 83.5 | 85 | 87.33 |
| | AUC (%) | 0.82 | 0.92 | 0.91 | 0.92 | 0.92 | **0.94** |
| | G-mean (%) | 0.74 | 0.91 | 0.9 | 0.91 | 0.91 | **0.93** |
| | F-measure (%) | 0.7 | 0.93 | 0.92 | 0.93 | 0.93 | **0.94** |

**Tabela 5.9:** *Métricas de desempenho calculadas para os modelos treinados com a base de dados de suco de uva orgânico sem balanceamento e balanceada de acordo com os algoritmos.*

| | Métodos | AUC | G-mean | F-measure |
|---|---|---|---|---|
| $p_{wilcoxon}$ SVM | PCA-SMOTE vs - | 0.001 | 0.001 | 0.001 |
| | PCA-SMOTE vs SMOTE | 0.2461 | 0.3125 | 0.3125 |
| | PCA-SMOTE vs SLSMOTE | 0.9045 | 0.9384 | 0.7969 |
| | PCA-SMOTE vs BMOTE | 0.0961 | 0.1792 | 0.0527 |
| | PCA-SMOTE vs ADASYN | 0.0295 | 0.1377 | 0.0513 |
| $p_{wilcoxon}$ C4.5 | PCA-SMOTE vs - | 0.0029 | 0.0049 | 0.001 |
| | PCA-SMOTE vs SMOTE | 0.1611 | 0.3178 | 0.0963 |
| | PCA-SMOTE vs SLSMOTE | 0.3606 | 0.2461 | 0.3125 |
| | PCA-SMOTE vs BMOTE | 0.4528 | 0.4229 | 0.4609 |
| | PCA-SMOTE vs ADASYN | 0.6825 | 0.5771 | 0.6875 |
| $p_{wilcoxon}$ MLP | PCA-SMOTE vs - | 0.001 | 0.001 | 0.001 |
| | PCA-SMOTE vs SMOTE | 0.0089 | 0.0149 | 0.0537 |
| | PCA-SMOTE vs SLSMOTE | 0.0064 | 0.0104 | 0.0045 |
| | PCA-SMOTE vs BMOTE | 0.0768 | 0.0616 | 0.0961 |
| | PCA-SMOTE vs ADASYN | 0.0332 | 0.0324 | 0.1377 |

**Tabela 5.10:** *Testes pareado de Wilcoxon realizados sobre as métricas de desempenho obtidas pelos modelos.*

A base de dados de suco de uva orgânico possui dimensões $27 \times 45$, onde 9 amostras são da classe "organic"(amostras de sucos de uva orgânicos) e 18 amostras pertencentes à classe "conventional"(amostras de sucos de uva comuns), configurando desta forma um fator de desbalanceamento de 1:2. Os algoritmos PCA-SMOTE, o SMOTE básico e o *Safe-Level-SMOTE* incluíram 9 amostras sintéticas da classe "organic"no conjunto de dados, enquanto os demais algoritmos incluíram 8 amostras. Este conjunto de dados tem a particularidade do grande número de dimensões superando a quantidade de amostras, mesmo após balanceado. A alta dimensionalidade de bases de dados é um dos desafios naturais para o aprendizado de modelos de classificação.

Esta base de dados também tem como principal característica a esparsidade, visível na Fig. 5.5, que mostra as amostras de dados em um gráfico bidimensional

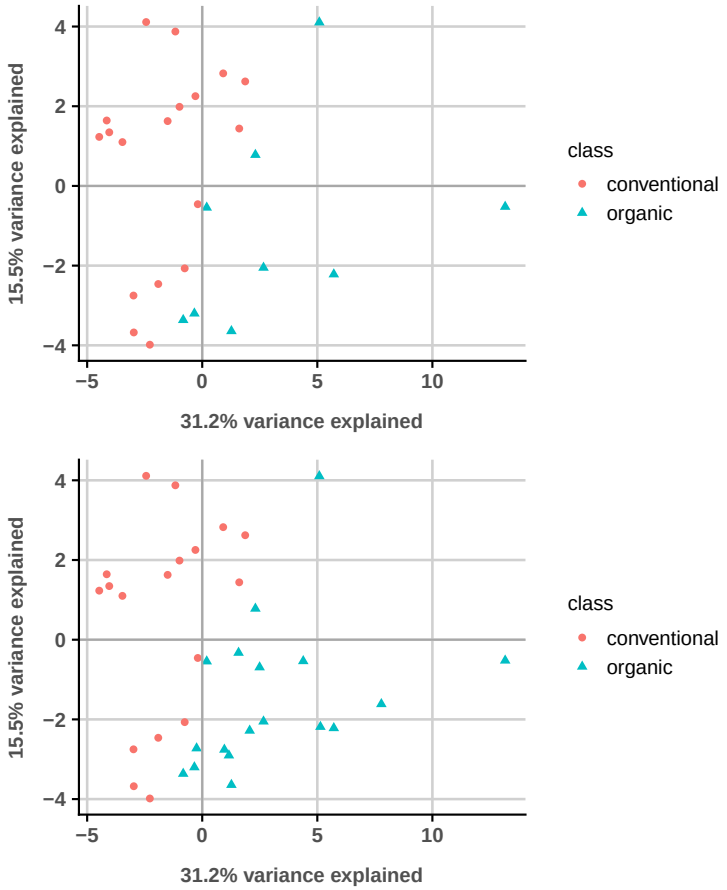



**Figura 5.5:** *Base de dados de suco de uva orgânico expressa em termos de suas duas componentes principais antes e após a aplicação do PCA-SMOTE.*

expresso em termos das duas primeiras componentes principais computadas pelo PCA. Embora as amostras estejam espalhadas por toda a região esquerda do gráfico, as amostras da classe *"organic"* se concentram na parte direita da região preenchida, enquanto as amostras da classe *"conventional"* se concentram primariamente na região esquerda.

Conforme exposto na Tabela 5.10 de testes de significância de Wilcoxon, o algoritmo PCA-SMOTE apresentou valores de AUC, *G-mean* e *F-measure* maiores do que o dos demais algoritmos de balanceamento para o modelo MLP. Para o modelo SVM, o PCA-SMOTE apresentou valores médios de AUC e *F-measure* maiores do que *Borderline-SMOTE* e ADASYN. Para todas as demais comparação pareadas entre o PCA-SMOTE e os demais algoritmos, os valores de $p$ obtidos foram superiores a 0.1. Contudo, observado a Tabela 5.9, vê-se para o PCA-SMOTE valores de AUC, *G-mean* e *F-measure* maiores ou iguais aos obtidos pelos outros algoritmos de balanceamento para a maioria dos cenários. A exceção foram os valores de AUC e *G-mean* obtidos pelo SLSMOTE quando combinado com o modelo SVM (0.96 e 0.95, respectivamente) que supera em 0.3 e 0.4 os valores obtido pelo PCA-SMOTE, e o valor de *F-measure* obtido pelo

ADASYN combinado ao modelo C4.5, 0.1 ponto maior do que o valor obtido pelo PCA-SMOTE. Também é possível ver na Tabela 5.9 alguns casos em que, para o modelo C4.5, o algoritmo proposto combinado ao modelo C4.5 se equiparou ao *Borderline-SMOTE* e ADASYN, em termos de *G-mean*, e ao *Borderline-SMOTE* e ao *Safe-Level-SMOTE*, em termos de *F-measure*. O SMOTE básico também obteve valores *G-mean* aliado ao modelo SVM igual ao obtido pelo PCA-SMOTE.

### 5.4.6 Breast Cancer Wisconsin (Original)

| Modelo | Medida | - | SMOTE | SLSMOTE | B-SMOTE | ADASYN | PCA-SMOTE |
|---|---|---|---|---|---|---|---|
| SVM | Acurácia (%) | 95.42 | 97.22 | 97.19 | 97.22 | 96.96 | **97.36** |
| | Sensitividade (%) | 93.8 | **94.98** | 94.76 | 94.87 | 94.41 | 94.91 |
| | Especificidade (%) | 98.51 | 99.42 | 99.54 | 99.51 | 99.63 | **99.76** |
| | AUC (%) | 0.96 | 0.97 | 0.97 | 0.97 | 0.97 | 0.97 |
| | G-mean (%) | 0.96 | 0.97 | 0.97 | 0.97 | 0.97 | 0.97 |
| | F-measure (%) | 0.96 | 0.97 | 0.97 | 0.97 | 0.97 | 0.97 |
| C4.5 | Acurácia (%) | 94.45 | 95.6 | 96.03 | 95.82 | **96.29** | 96 |
| | Sensitividade (%) | **95.39** | 94.59 | 94.65 | 94.85 | 95.31 | 94.52 |
| | Especificidade (%) | 92.66 | 96.6 | 97.36 | 96.77 | 97.32 | **97.45** |
| | AUC (%) | 0.94 | 0.96 | 0.96 | 0.96 | 0.96 | 0.96 |
| | G-mean (%) | 0.94 | 0.96 | 0.96 | 0.96 | 0.96 | 0.96 |
| | F-measure (%) | 0.96 | 0.96 | 0.96 | 0.96 | 0.96 | 0.96 |
| MLP | Acurácia (%) | 96.11 | **97.47** | 97.19 | 97.41 | 97.42 | 97.34 |
| | Sensitividade (%) | **95.94** | 95.83 | 95.66 | 95.81 | 95.59 | 95.91 |
| | Especificidade (%) | 96.43 | 99.08 | 98.67 | 98.98 | **99.33** | 98.73 |
| | AUC (%) | 0.96 | 0.97 | 0.97 | 0.97 | 0.97 | 0.97 |
| | G-mean (%) | 0.96 | 0.97 | 0.97 | 0.97 | 0.97 | 0.97 |
| | F-measure (%) | 0.97 | 0.97 | 0.97 | 0.97 | 0.97 | 0.97 |

**Tabela 5.11:** *Métricas de desempenho calculadas para os modelos treinados com a base de dados* ***Breast Cancer Wisconsin Original*** *sem balanceamento e balanceada de acordo com os algoritmos.*

| | Métodos | AUC | G-mean | F-measure |
|---|---|---|---|---|
| $p_{wilcoxon}$ SVM | PCA-SMOTE vs - | 0.001 | 0.001 | 0.001 |
| | PCA-SMOTE vs SMOTE | 0.0527 | 0.0967 | 0.0967 |
| | PCA-SMOTE vs SLSMOTE | 0.0137 | 0.0098 | 0.0049 |
| | PCA-SMOTE vs BMOTE | 0.0244 | 0.0244 | 0.0244 |
| | PCA-SMOTE vs ADASYN | 0.0029 | 0.0049 | 0.0098 |
| $p_{wilcoxon}$ C4.5 | PCA-SMOTE vs - | 0.001 | 0.001 | 0.2461 |
| | PCA-SMOTE vs SMOTE | 0.0137 | 0.0137 | 0.0186 |
| | PCA-SMOTE vs SLSMOTE | 0.5391 | 0.5391 | 0.5 |
| | PCA-SMOTE vs BMOTE | 0.1875 | 0.1875 | 0.1611 |
| | PCA-SMOTE vs ADASYN | 0.9951 | 0.9951 | 0.999 |
| $p_{wilcoxon}$ MLP | PCA-SMOTE vs - | 0.001 | 0.001 | 0.1162 |
| | PCA-SMOTE vs SMOTE | 0.9033 | 0.9033 | 0.8838 |
| | PCA-SMOTE vs SLSMOTE | 0.3125 | 0.2783 | 0.2783 |
| | PCA-SMOTE vs BMOTE | 0.8125 | 0.8125 | 0.8125 |
| | PCA-SMOTE vs ADASYN | 0.8623 | 0.8623 | 0.8623 |

**Tabela 5.12:** *Testes pareado de Wilcoxon realizados sobre as métricas de desempenho obtidas pelos modelos.*

A base de dados *Breast Cancer Wisconsin Original* possui dimensões $699 \times 10$, sendo 241 amostras da classe "4"(pacientes com câncer maligno) e as demais 458 amostras pertencentes à classe "2"(pacientes com câncer benigno), configurando desta forma um fator de desbalanceamento de aproximadamente 1:2. Os algoritmos de balanceamento incluíram 225 (PCA-SMOTE), 224 (SMOTE básico), 229 (*Borderline-SMOTE*), 232 (*Safe-Level-SMOTE*) e 195 (ADASYN) amostras sintéticas da classe minoritária "2"no conjunto de dados. As duas primeiras componentes principais computadas pelo PCA explicam juntas aproximadamente 74% da variância total dos dados, como mostrado na Fig. 5.6. É possível ver que as amostras da classe majoritária "2"formam um grupo denso na região direita do gráfico onde amostras se sobrepõem, enquanto as amostras da classe

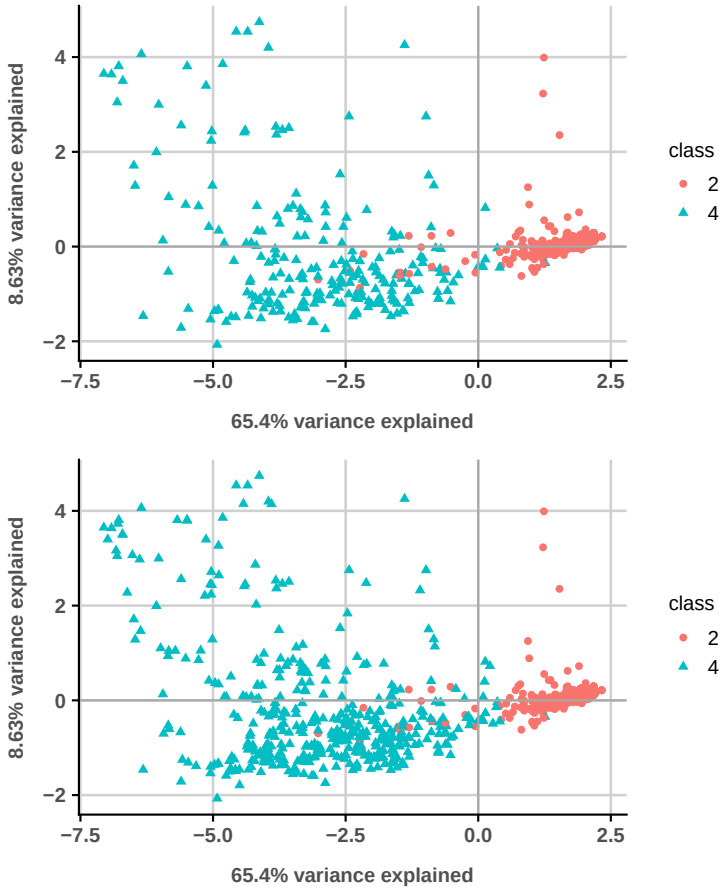



**Figura 5.6:** *Base de dados* ***Breast Cancer Wisconsin Original*** *expressa em termos de suas duas componentes principais antes e após a aplicação do PCA-SMOTE.*

"4"estão mais esparsas e se distribuem por toda a região central e esquerda do gráfico, podendo causar a falsa conclusão de que há mais amostras da classe "4"do que da classe "2"na base de dados.

Como esperado de uma base de dados com classes bem separadas, os três modelos de classificação testados apresentaram ótimos valores de desempenhos mesmo quando treinados com a base de dados desbalanceada, como evidenciado na Tabela 5.11. O balanceamento da base através do algoritmo PCA-SMOTE melhora os valores de acurácia, sensitividade e especificidade obtidos pelos três modelos, e considerando estas três métricas, o PCA-SMOTE se sobressai:

- a todos os demais algoritmos, para o modelo SVM (com exceção do SMOTE básico, que apresenta 94.98% de sensitividade, 0.07% superior ao obtido pelo PCA-SMOTE);
- ao SMOTE básico e ao *Borderline-SMOTE*, para o modelo C4.5 em termos de acurácia;
- a todos os demais algoritmos, para o modelo MLP em termos de sensitividade;

- a todos os demais algoritmos para o modelo C4.5 em termos de especificidade;
- ao algoritmo *Safe-Level-SMOTE* para o modelo MLP em termos de especificidade.

Os modelos de classificação apresentaram valores idênticos de AUC, *G-mean* e *F-measure* quando treinados com todas as versões balanceadas da base de dados de acordo com os algoritmos investigados. Entretanto, na Tabela 5.12, pode-se atestar através dos testes de significância que o modelo PCA-SMOTE se sobressai a todos os demais algoritmos de balanceamento para o modelo SVM, e ao SMOTE básico e ao *Safe-Level-SMOTE* para o modelo C4.5.

### 5.4.7 Glass Identification

| Modelo | Medida | - | SMOTE | SLSMOTE | B-SMOTE | ADASYN | PCA-SMOTE |
|---|---|---|---|---|---|---|---|
| SVM | Acurácia (%) | 85.78 | 85.57 | **87.73** | 85.52 | 86.52 | 86.64 |
| | Sensitividade (%) | 66.97 | 78.41 | 85.42 | 79.74 | **85.72** | 82.11 |
| | Especificidade (%) | **94.59** | 92.67 | 90.13 | 91.26 | 87.24 | 91.22 |
| | AUC (%) | 0.81 | 0.86 | **0.88** | 0.85 | 0.86 | 0.87 |
| | G-mean (%) | 0.79 | 0.85 | **0.88** | 0.85 | 0.86 | 0.86 |
| | F-measure (%) | 0.74 | 0.84 | **0.87** | 0.84 | 0.86 | 0.86 |
| C4.5 | Acurácia (%) | 81.88 | 84.29 | **86.7** | 83.98 | 85.24 | 83.12 |
| | Sensitividade (%) | 66.21 | 83.66 | 86.09 | 83.77 | **88.36** | 84.46 |
| | Especificidade (%) | **89.17** | 84.93 | 87.31 | 84.19 | 82.38 | 81.78 |
| | AUC (%) | 0.78 | 0.84 | **0.87** | 0.84 | 0.85 | 0.83 |
| | G-mean (%) | 0.76 | 0.84 | **0.87** | 0.84 | 0.85 | 0.83 |
| | F-measure (%) | 0.7 | 0.84 | **0.87** | 0.84 | 0.85 | 0.83 |
| MLP | Acurácia (%) | 83.16 | 85.97 | 86.19 | 86.23 | **87.57** | 87.42 |
| | Sensitividade (%) | 63.3 | 83.72 | 86.48 | 85.81 | **89.86** | 86.7 |
| | Especificidade (%) | **92.43** | 88.19 | 85.91 | 86.64 | 85.5 | 88.17 |
| | AUC (%) | 0.78 | 0.86 | 0.86 | 0.86 | **0.88** | 0.87 |
| | G-mean (%) | 0.76 | 0.86 | 0.86 | 0.86 | 0.87 | 0.87 |
| | F-measure (%) | 0.7 | 0.85 | 0.86 | 0.86 | 0.87 | 0.87 |

**Tabela 5.13:** *Métricas de desempenho calculadas para os modelos treinados com a base de dados* ***Glass Identification*** *sem balanceamento e balanceada de acordo com os algoritmos.*

| | Métodos | AUC | G-mean | F-measure |
|---|---|---|---|---|
| $p_{wilcoxon}$ SVM | PCA-SMOTE vs - | 0.001 | 0.001 | 0.001 |
| | PCA-SMOTE vs SMOTE | 0.0049 | 0.0049 | 0.0029 |
| | PCA-SMOTE vs SLSMOTE | 0.9951 | 0.9971 | 0.999 |
| | PCA-SMOTE vs BMOTE | 0.0322 | 0.0244 | 0.0186 |
| | PCA-SMOTE vs ADASYN | 0.4229 | 0.5 | 0.4609 |
| $p_{wilcoxon}$ C4.5 | PCA-SMOTE vs - | 0.001 | 0.001 | 0.001 |
| | PCA-SMOTE vs SMOTE | 0.9033 | 0.9033 | 0.7217 |
| | PCA-SMOTE vs SLSMOTE | 1 | 1 | 1 |
| | PCA-SMOTE vs BMOTE | 0.9199 | 0.8623 | 0.7539 |
| | PCA-SMOTE vs ADASYN | 0.9902 | 0.9902 | 0.9678 |
| $p_{wilcoxon}$ MLP | PCA-SMOTE vs - | 0.001 | 0.001 | 0.001 |
| | PCA-SMOTE vs SMOTE | 0.0068 | 0.0098 | 0.0068 |
| | PCA-SMOTE vs SLSMOTE | 0.0137 | 0.0137 | 0.0654 |
| | PCA-SMOTE vs BMOTE | 0.0527 | 0.0801 | 0.0967 |
| | PCA-SMOTE vs ADASYN | 0.7217 | 0.7539 | 0.5391 |

**Tabela 5.14:** *Testes pareado de Wilcoxon realizados sobre as métricas de desempenho obtidas pelos modelos.*

A base de dados *Glass Identification* possui dimensões $214 \times 10$, sendo 146 amostras da classe *"window"* (amostras de vidro oriundos de janelas de veículos) e 68 amostras pertencentes à classe minoritária *"other"* (amostras de vidro oriundos de outras fontes), configurando desta forma um fator de desbalanceamento de aproximadamente 1:2,4. Os algoritmos de balanceamento incluíram 80 (PCA-SMOTE), 77 (SMTE básico), 76 (*Borderline-SMOTE*), 83 (*Safe-Level-SMOTE*) e 65 (ADASYN) amostras sintéticas da classe minoritária.

As duas primeiras componentes principais computadas pelo PCA expliquem apenas aproximadamente 50% da variância total dos dados, como mostrado na Fig. 5.7. Em termos delas, as amostras da classe *"window"* se concentram na região central inferior

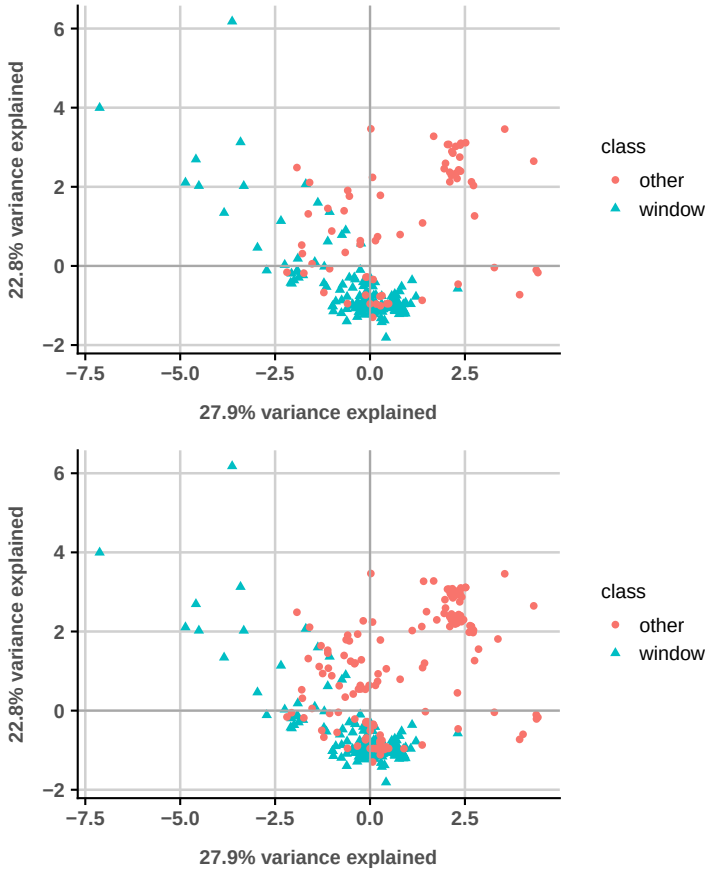



**Figura 5.7:** *Base de dados* ***Glass Identification*** *expressa em termos de suas duas componentes principais antes e após a aplicação do PCA-SMOTE.*

do gráfico, formando um grupo denso, com algumas poucas amostras se espalhando pela região esquerda central. Já as amostras da classe *"other"* formam um grupo pequeno na região direita central do gráfico, mas com grande parte das amostras distribuídas de maneira esparsa pela região central, sendo visível certa sobreposição com as amostras da classe *"window"* - principalmente após a aplicação do PCA-SMOTE, que realiza o *oversampling* de maneira equilibrada entre as várias regiões do gráfico dominadas pelas amostras da classe minoritária *"other"*.

Como mostrado nos testes de significância na Tabela 5.14, o algoritmo PCA-SMOTE mostrou valores de AUC, *G-mean* e *F-measure* superiores aos obtidos pelos demais algoritmos para o modelo MLP, exceto ADASYN. Ele também supera o *Borderline-SMOTE* e o SMOTE básico quando combinados ao modelo SVM. Além disso, pode-se constatar através da Tabela 5.13 que o PCA-SMOTE produz valores iguais aos obtidos pelos algoritmo ADASYN, para o modelo SVM em termos de *G-mean* e *F-measure*.

### 5.4.8 Wine Quality

| Modelo | Medida | - | SMOTE | SLSMOTE | B-SMOTE | ADASYN | PCA-SMOTE |
|---|---|---|---|---|---|---|---|
| SVM | Acurácia (%) | 88.49 | 87.08 | **89.2** | 87.27 | 87.22 | 87.17 |
| | Sensitividade (%) | 27.82 | 90.9 | **94.13** | 91.3 | 91.3 | 90.91 |
| | Especificidade (%) | **98.02** | 83.26 | 84.23 | 83.22 | 83.73 | 83.4 |
| | AUC (%) | 0.63 | 0.87 | **0.89** | 0.87 | 0.88 | 0.87 |
| | G-mean (%) | 0.52 | 0.87 | **0.89** | 0.87 | 0.87 | 0.87 |
| | F-measure (%) | 0.39 | 0.88 | **0.9** | 0.88 | 0.87 | 0.88 |
| C4.5 | Acurácia (%) | 87.62 | 89.62 | **91.72** | 89.41 | 91.43 | 88.73 |
| | Sensitividade (%) | 44.61 | 92.68 | 94.04 | 92.86 | **94.35** | 91.78 |
| | Especificidade (%) | **94.37** | 86.56 | 89.38 | 85.95 | 88.94 | 85.65 |
| | AUC (%) | 0.69 | 0.9 | **0.92** | 0.89 | **0.92** | 0.89 |
| | G-mean (%) | 0.65 | 0.9 | **0.92** | 0.89 | **0.92** | 0.89 |
| | F-measure (%) | 0.49 | 0.9 | **0.92** | 0.9 | 0.91 | 0.89 |
| MLP | Acurácia (%) | 87.53 | 85.03 | 87.03 | 85.59 | **87.65** | 85.92 |
| | Sensitividade (%) | 32.46 | 86.48 | **88.78** | 86.47 | 87.74 | 87.58 |
| | Especificidade (%) | **96.17** | 83.58 | 85.27 | 84.7 | 87.58 | 84.26 |
| | AUC (%) | 0.64 | 0.85 | 0.87 | 0.86 | **0.88** | 0.86 |
| | G-mean (%) | 0.54 | 0.85 | 0.87 | 0.86 | **0.88** | 0.86 |
| | F-measure (%) | 0.4 | 0.85 | **0.87** | 0.86 | **0.87** | 0.86 |

**Tabela 5.15:** *Métricas de desempenho calculadas para os modelos treinados com a base de dados **Wine Quality** sem balanceamento e balanceada de acordo com os algoritmos.*

| | Métodos | AUC | G-mean | F-measure |
|---|---|---|---|---|
| $p_{wilcoxon}$ SVM | PCA-SMOTE vs - | 0.001 | 0.001 | 0.001 |
| | PCA-SMOTE vs SMOTE | 0.3477 | 0.3477 | 0.2158 |
| | PCA-SMOTE vs SLSMOTE | 1 | 1 | 1 |
| | PCA-SMOTE vs BMOTE | 0.7539 | 0.7217 | 0.7539 |
| | PCA-SMOTE vs ADASYN | 0.9971 | 0.9971 | 0.001 |
| $p_{wilcoxon}$ C4.5 | PCA-SMOTE vs - | 0.001 | 0.001 | 0.001 |
| | PCA-SMOTE vs SMOTE | 0.9951 | 0.9951 | 0.998 |
| | PCA-SMOTE vs SLSMOTE | 1 | 1 | 1 |
| | PCA-SMOTE vs BMOTE | 0.999 | 0.998 | 0.999 |
| | PCA-SMOTE vs ADASYN | 1 | 1 | 1 |
| $p_{wilcoxon}$ MLP | PCA-SMOTE vs - | 0.001 | 0.001 | 0.001 |
| | PCA-SMOTE vs SMOTE | 0.002 | 0.0029 | 0.0029 |
| | PCA-SMOTE vs SLSMOTE | 0.999 | 0.999 | 0.9951 |
| | PCA-SMOTE vs BMOTE | 0.0186 | 0.0527 | 0.0068 |
| | PCA-SMOTE vs ADASYN | 0.9951 | 0.9902 | 0.958 |

**Tabela 5.16:** *Testes pareado de Wilcoxon realizados sobre as métricas de desempenho obtidas pelos modelos.*

A base de dados *Wine Quality* processada para o teste possui dimensões finais $1599 \times 12$, sendo 217 amostras da classe minoritária *"good"* (amostras de vinho verde tinto classificadas com boas notas) e 1382 amostras pertencentes à classe*"other"* (amostras de vinho verde tinto que receberam notas médias de qualidade), configurando desta forma um fator de desbalanceamento de aproximadamente 1:6,4. Os algoritmos de balanceamento incluíram 1169 (PCA-SMOTE), 1171 (SMTE básico), 1175 (*Borderline-SMOTE*), 1182 (*Safe-Level-SMOTE*) e 962 (ADASYN) amostras sintéticas da classe minoritária na base de dados de teste.

Esta base de dados de teste é caracterizada, além do fator alto de desbalanceamento, por também um alto nível de sobreposição de dados, como evidenciado na Fig.

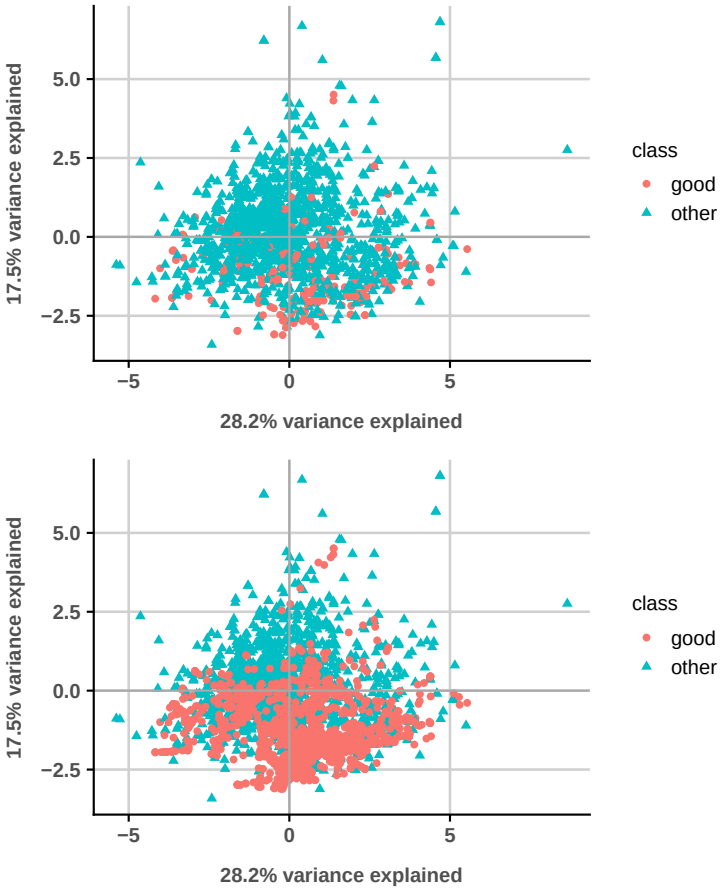



**Figura 5.8:** *Base de dados **Wine Quality** expressa em termos de suas duas componentes principais antes e após a aplicação do PCA-SMOTE.*

5.8, principalmente após a geração das amostras sintéticas. As duas primeiras componentes principais extraídas pelo PCA explicam apenas 45.7% da variância total das amostras. Ainda assim, o gráfico bidimensional foi mantido para facilitar a visualização dos dados. Em termos das duas componentes principais, as amostras da classe minoritária *"good"* se concentram mais na região central inferior do gráfico, tornando-se mais esparsas e eventualmente desaparecendo conforme os valores da segunda componente principal crescem. Já as amostras da classe *"other"* formam um grande grupo que ocupa de maneira quase uniforme toda a região central do gráfico, com esparsidade observada nas bordas do grupo. Após o balanceamento, a região inferior central do gráfico passa a ser abastecida por mais amostras da classe minoritária, agravando a sobreposição. Contudo, a região superior central do grupo de dados ainda é majoritariamente dominada por amostras da classe *"other"*.

Como mostrado nos testes de significância na Tabela 5.16, o algoritmo PCA-SMOTE mostrou valores de AUC, *G-mean* e *F-measure* superiores aos obtidos pelo SMOTE básico e pelo *Borderline-SMOTE* para o modelo MLP. Novamente, embora este tenha sido o único caso de superioridade explícita do algoritmo proposto, podemos ver na

Tabela 5.15 que o PCA-SMOTE produz valores iguais aos obtidos pelos algoritmos:

- *Borderline-SMOTE*, para todos os modelos, de acordo com as métricas AUC, *G-mean* e *F-measure*;
- SMOTE básico, para o modelo SVM, de acordo com as três métricas;
- ADASYN, para o modelo SVM avaliado pela métrica *G-mean*.

### 5.4.9 Yeast

| Modelo | Medida | - | SMOTE | SLSMOTE | B-SMOTE | ADASYN | PCA-SMOTE |
|---|---|---|---|---|---|---|---|
| SVM | Acurácia (%) | 92.7 | 93.57 | **94.92** | 93.62 | 94.13 | 94.19 |
| | Sensitividade (%) | 73.82 | 95.21 | 96.16 | 95.23 | 96.28 | **96.55** |
| | Especificidade (%) | **96.66** | 91.91 | 93.67 | 91.98 | 92.45 | 91.79 |
| | AUC (%) | 0.85 | 0.94 | **0.95** | 0.94 | 0.94 | 0.94 |
| | G-mean (%) | 0.84 | 0.94 | **0.95** | 0.94 | 0.94 | 0.94 |
| | F-measure (%) | 0.78 | 0.94 | **0.95** | 0.94 | 0.94 | 0.94 |
| C4.5 | Acurácia (%) | 92.01 | 93.09 | **94.67** | 93.09 | 93.99 | 93.29 |
| | Sensitividade (%) | 80.7 | 95.35 | **97.34** | 95.86 | 96.03 | 96.26 |
| | Especificidade (%) | **94.4** | 90.8 | 91.97 | 90.28 | 92.4 | 90.27 |
| | AUC (%) | 0.88 | 0.93 | **0.95** | 0.93 | 0.94 | 0.93 |
| | G-mean (%) | 0.87 | 0.93 | **0.95** | 0.93 | 0.94 | 0.93 |
| | F-measure (%) | 0.78 | 0.93 | **0.95** | 0.93 | 0.93 | 0.94 |
| MLP | Acurácia (%) | 92.34 | 91.56 | **93.23** | 91.52 | 92.1 | 92.01 |
| | Sensitividade (%) | 75.55 | 91.93 | **93.81** | 91.02 | 91.62 | 92.9 |
| | Especificidade (%) | **95.87** | 91.18 | 92.64 | 92.02 | 92.48 | 91.1 |
| | AUC (%) | 0.86 | 0.92 | **0.93** | 0.92 | 0.92 | 0.92 |
| | G-mean (%) | 0.85 | 0.92 | **0.93** | 0.91 | 0.92 | 0.92 |
| | F-measure (%) | 0.77 | 0.92 | **0.93** | 0.92 | 0.91 | 0.92 |

**Tabela 5.17:** *Métricas de desempenho calculadas para os modelos treinados com a base de dados **Yeast** sem balanceamento e balanceada de acordo com os algoritmos.*

| | Métodos | AUC | G-mean | F-measure |
|---|---|---|---|---|
| $p_{wilcoxon}$ SVM | PCA-SMOTE vs - | 0.001 | 0.001 | 0.001 |
| | PCA-SMOTE vs SMOTE | 0.001 | 0.001 | 0.001 |
| | PCA-SMOTE vs SLSMOTE | 1 | 1 | 1 |
| | PCA-SMOTE vs BMOTE | 0.001 | 0.001 | 0.001 |
| | PCA-SMOTE vs ADASYN | 0.958 | 0.9678 | 0.001 |
| $p_{wilcoxon}$ C4.5 | PCA-SMOTE vs - | 0.001 | 0.001 | 0.001 |
| | PCA-SMOTE vs SMOTE | 0.042 | 0.042 | 0.0098 |
| | PCA-SMOTE vs SLSMOTE | 1 | 1 | 1 |
| | PCA-SMOTE vs BMOTE | 0.0801 | 0.0801 | 0.0527 |
| | PCA-SMOTE vs ADASYN | 1 | 1 | 0.0527 |
| $p_{wilcoxon}$ MLP | PCA-SMOTE vs - | 0.001 | 0.001 | 0.001 |
| | PCA-SMOTE vs SMOTE | 0.0068 | 0.0068 | 0.0049 |
| | PCA-SMOTE vs SLSMOTE | 1 | 1 | 1 |
| | PCA-SMOTE vs BMOTE | 0.0049 | 0.0049 | 0.0049 |
| | PCA-SMOTE vs ADASYN | 0.7217 | 0.6875 | 0.001 |

**Tabela 5.18:** *Testes pareado de Wilcoxon realizados sobre as métricas de desempenho obtidas pelos modelos.*

A base de dados *Yeast* processada para o teste possui dimensões finais $1484 \times 9$, sendo 258 amostras da classe minoritária *"membrane"* (amostras de proteínas localizadas em membrana celular) e 1226 amostras pertencentes à classe*"other"* (amostras proteínas em outras localizações em células), configurando desta forma um fator de desbalanceamento de aproximadamente 1:4,7. Os algoritmos de balanceamento incluíram 983 (PCA-SMOTE e SMOTE básico), 964 (*Borderline-SMOTE*), 984 (*Safe-Level-SMOTE*) e 699 (ADASYN) amostras sintéticas da classe minoritária na base de dados de teste.

Esta base de dados de teste também é caracterizada por fator alto de desbalanceamento e alto nível de sobreposição de dados, como evidenciado na Fig. 5.8, principalmente após a geração das amostras sintéticas. As duas primeiras componentes principais extraídas pelo PCA explicam apenas 38.6% da variância total das amostras, e aqui, o

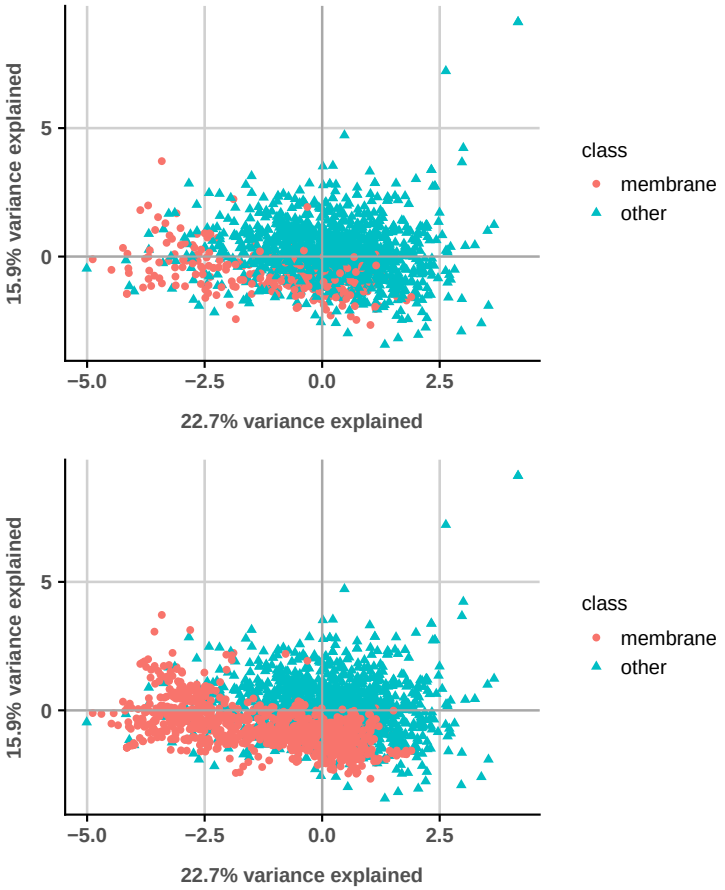



**Figura 5.9:** *Base de dados* ***Yeast*** *expressa em termos de suas duas componentes principais antes e após a aplicação do PCA-SMOTE.*

gráfico bidimensional também foi mantido para facilitar a visualização dos dados. As amostras de dados de ambas as classes formam um grande grupo denso na porção central do gráfico. Amostras da classe minoritária *"membrane"* dominam a região esquerda e inferior deste grupo, sendo que na região esquerda a sobreposição de dados aparenta ser substancialmente menor. Após o balanceamento, a região inferior central do gráfico passa a ser abastecida por mais amostras da classe minoritária, agravando a sobreposição. Contudo, a região superior central e direita do grupo de dados ainda é majoritariamente dominada por amostras da classe *"other"*.

Como mostrado nos testes de significância na Tabela 5.18, o algoritmo PCA-SMOTE mostrou valores de AUC, *G-mean* e *F-measure* superiores aos obtidos pelo SMOTE básico e pelo *Borderline-SMOTE* para os três modelos de classificação avaliados. Ainda, de acordo com a Tabela 5.17, o PCA-SMOTE produziu valores iguais aos obtidos pelo algoritmo ADASYN para o modelo MLP, em termos de AUC e *G-mean*, e para o modelo SVM, em termos de *G-mean* e *F-measure*.

### 5.4.10 Pima Indians Diabetes

| Modelo | Medida | - | SMOTE | SLSMOTE | B-SMOTE | ADASYN | PCA-SMOTE |
|---|---|---|---|---|---|---|---|
| SVM | Acurácia (%) | 76.32 | 77.53 | **78.78** | 77.85 | 77.32 | 78.24 |
| | Sensitividade (%) | 53.39 | 80.81 | 81.84 | 82.06 | 79.42 | **82.96** |
| | Especificidade (%) | **88.6** | 74.2 | 75.68 | 73.56 | 75.5 | 73.36 |
| | AUC (%) | 0.71 | 0.78 | **0.79** | 0.78 | 0.77 | 0.78 |
| | G-mean (%) | 0.69 | 0.77 | **0.79** | 0.78 | 0.77 | 0.78 |
| | F-measure (%) | 0.61 | 0.78 | 0.79 | 0.79 | 0.76 | 0.79 |
| C4.5 | Acurácia (%) | 73.68 | 75.82 | **77.84** | 75.88 | 76.67 | 76.87 |
| | Sensitividade (%) | 58.52 | 80.72 | 81.32 | 79.2 | 78.49 | **82.19** |
| | Especificidade (%) | **81.8** | 70.84 | 74.32 | 72.5 | 75.08 | 71.38 |
| | AUC (%) | 0.7 | 0.76 | **0.78** | 0.76 | 0.77 | 0.77 |
| | G-mean (%) | 0.69 | 0.75 | **0.78** | 0.76 | 0.77 | 0.76 |
| | F-measure (%) | 0.6 | 0.77 | **0.79** | 0.77 | 0.76 | 0.78 |
| MLP | Acurácia (%) | 75.28 | 76.38 | **78.78** | 76.6 | 76.52 | 76.69 |
| | Sensitividade (%) | 57.88 | 75.92 | **83.73** | 79.04 | 79.05 | 78.77 |
| | Especificidade (%) | **84.6** | 76.86 | 73.76 | 74.12 | 74.34 | 74.54 |
| | AUC (%) | 0.71 | 0.76 | **0.79** | 0.77 | 0.77 | 0.77 |
| | G-mean (%) | 0.69 | 0.75 | **0.78** | 0.76 | 0.76 | 0.76 |
| | F-measure (%) | 0.61 | 0.76 | **0.8** | 0.77 | 0.76 | 0.77 |

**Tabela 5.19:** *Métricas de desempenho calculadas para os modelos treinados com a base de dados* ***Pima Indians Diabetes*** *sem balanceamento e balanceada de acordo com os algoritmos.*

| | Métodos | AUC | G-mean | F-measure |
|---|---|---|---|---|
| $p_{wilcoxon}$ SVM | PCA-SMOTE vs - | 0.001 | 0.001 | 0.001 |
| | PCA-SMOTE vs SMOTE | 0.0068 | 0.0068 | 0.001 |
| | PCA-SMOTE vs SLSMOTE | 0.9902 | 0.9902 | 0.5391 |
| | PCA-SMOTE vs BMOTE | 0.1875 | 0.2158 | 0.0244 |
| | PCA-SMOTE vs ADASYN | 0.0244 | 0.0654 | 0.001 |
| $p_{wilcoxon}$ C4.5 | PCA-SMOTE vs - | 0.001 | 0.001 | 0.001 |
| | PCA-SMOTE vs SMOTE | 0.0322 | 0.042 | 0.002 |
| | PCA-SMOTE vs SLSMOTE | 0.9678 | 0.9756 | 0.8623 |
| | PCA-SMOTE vs BMOTE | 0.0967 | 0.1162 | 0.042 |
| | PCA-SMOTE vs ADASYN | 0.5 | 0.6152 | 0.001 |
| $p_{wilcoxon}$ MLP | PCA-SMOTE vs - | 0.001 | 0.001 | 0.001 |
| | PCA-SMOTE vs SMOTE | 0.4229 | 0.2783 | 0.1611 |
| | PCA-SMOTE vs SLSMOTE | 1 | 1 | 0.999 |
| | PCA-SMOTE vs BMOTE | 0.5 | 0.5771 | 0.3848 |
| | PCA-SMOTE vs ADASYN | 0.3477 | 0.3125 | 0.0049 |

**Tabela 5.20:** *Testes pareado de Wilcoxon realizados sobre as métricas de desempenho obtidas pelos modelos.*

A base de dados *Pima Indians Diabetes* possui dimensões $768 \times 9$, sendo 268 amostras da classe minoritária "1"(amostras de mulheres indígenas da linhagem Pima que possuem diabetes) e 500 amostras pertencentes à classe "0"(amostras de mulheres indígenas da linhagem Pima que não possuem diabetes), configurando desta forma um fator de desbalanceamento de aproximadamente 1:8,6. Os algoritmos de balanceamento incluíram 249 (PCA-SMOTE), 240 (SMOTE básico), 242 (*Borderline-SMOTE*), 238 (*Safe-Level-SMOTE*) e 165 (ADASYN) amostras sintéticas da classe minoritária na base de dados de teste.

A Fig. 5.10 mostra as amostras da base expressas em termos das duas primeiras componentes principais extraídas pelo PCA, que explicam juntas um total de 47.8% da

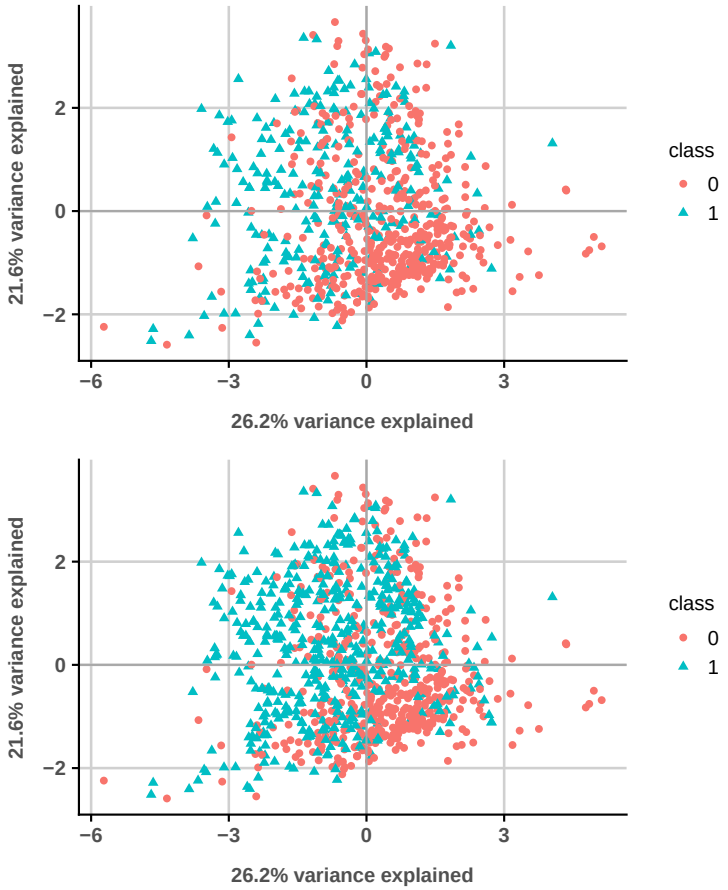



**Figura 5.10:** *Base de dados* ***Pima Indians Diabetes*** *expressa em termos de suas duas componentes principais antes e após a aplicação do PCA-SMOTE.*

variância dos dados. Além de desbalaceamento e sobreposição acentuada de dados, esta base também é caracterizada pela esparsidade, principalmente por parte das amostras da classe "1". Já as amostras da classe majoritária formam um agrupamento visível na parte inferior direita da região dos dados, onde aparentemente há pouca sobreposição com amostras da classe "0". Conforme os valores da primeira componente principal diminuem, amostras da classe majoritária tendem a se tornar escassas e amostras da classe minoritária formam um grande agrupamento, principalmente após o balanceamento pelo PCA-SMOTE.

Como mostrado nos testes de significância na Tabela 5.20, o algoritmo PCA-SMOTE mostrou valores de AUC, *G-mean* e *F-measure* superiores aos obtidos pelo *Borderline-SMOTE*, para os modelos SVM e C4.5, e pelo SMOTE básico, para o modelo C4.5. Ainda, de acordo com a Tabela 5.19, o PCA-SMOTE produziu valores iguais aos obtidos pelos algoritmos:

- ADASYN, em termos de valor AUC e *G-mean*, para os modelos MLP e C4.5;
- *Safe-Level-SMOTE*, para o modelo SVM avaliado pela métrica *F-measure*.

### 5.4.11 Ecoli

| Modelo | Medida | - | SMOTE | SLSMOTE | B-SMOTE | ADASYN | PCA-SMOTE |
|---|---|---|---|---|---|---|---|
| SVM | Acurácia (%) | 93.09 | 94.38 | **96** | 94.27 | 95.15 | 94.52 |
| | Sensitividade (%) | 60 | 98.55 | 97.85 | 98.43 | 98.19 | **98.71** |
| | Especificidade (%) | **96.94** | 90.37 | 94.15 | 90.13 | 92.92 | 90.43 |
| | AUC (%) | 0.78 | 0.94 | **0.96** | 0.94 | **0.96** | 0.95 |
| | G-mean (%) | 0.75 | 0.94 | **0.96** | 0.94 | 0.95 | 0.94 |
| | F-measure (%) | 0.64 | 0.95 | **0.96** | 0.95 | 0.95 | 0.95 |
| C4.5 | Acurácia (%) | 91.61 | 93.59 | **95.78** | 93.75 | 93.93 | 94.49 |
| | Sensitividade (%) | 49.71 | 95.83 | **97.35** | 94.61 | 95.7 | 97.14 |
| | Especificidade (%) | **96.48** | 91.43 | 94.21 | 92.89 | 92.63 | 91.89 |
| | AUC (%) | 0.73 | 0.94 | **0.96** | 0.94 | 0.94 | 0.95 |
| | G-mean (%) | 0.68 | 0.94 | **0.96** | 0.94 | 0.94 | 0.94 |
| | F-measure (%) | 0.54 | 0.94 | **0.96** | 0.94 | 0.93 | 0.95 |
| MLP | Acurácia (%) | 92.29 | 94.25 | **95.79** | 94.38 | 95.54 | 94.65 |
| | Sensitividade (%) | 60 | 97 | **97.75** | 97.7 | 97.64 | 97.34 |
| | Especificidade (%) | **96.04** | 91.59 | 93.82 | 91.09 | 93.99 | 92.03 |
| | AUC (%) | 0.78 | 0.94 | **0.96** | 0.94 | **0.96** | 0.95 |
| | G-mean (%) | 0.74 | 0.94 | **0.96** | 0.94 | **0.96** | 0.95 |
| | F-measure (%) | 0.6 | 0.94 | **0.96** | 0.95 | 0.95 | 0.95 |

**Tabela 5.21:** *Métricas de desempenho calculadas para os modelos treinados com a base de dados **Ecoli** sem balanceamento e balanceada de acordo com os algoritmos.*

| | Métodos | AUC | G-mean | F-measure |
|---|---|---|---|---|
| $p_{wilcoxon}$ SVM | PCA-SMOTE vs - | 0.001 | 0.001 | 0.001 |
| | PCA-SMOTE vs SMOTE | 0.3848 | 0.3848 | 0.2461 |
| | PCA-SMOTE vs SLSMOTE | 1 | 1 | 1 |
| | PCA-SMOTE vs BMOTE | 0.0801 | 0.0801 | 0.0801 |
| | PCA-SMOTE vs ADASYN | 1 | 1 | 0.1875 |
| $p_{wilcoxon}$ C4.5 | PCA-SMOTE vs - | 0.001 | 0.001 | 0.001 |
| | PCA-SMOTE vs SMOTE | 0.0186 | 0.0186 | 0.0137 |
| | PCA-SMOTE vs SLSMOTE | 0.9971 | 0.9971 | 0.9971 |
| | PCA-SMOTE vs BMOTE | 0.001 | 0.001 | 0.001 |
| | PCA-SMOTE vs ADASYN | 0.1377 | 0.1377 | 0.001 |
| $p_{wilcoxon}$ MLP | PCA-SMOTE vs - | 0.001 | 0.001 | 0.001 |
| | PCA-SMOTE vs SMOTE | 0.1611 | 0.2158 | 0.1611 |
| | PCA-SMOTE vs SLSMOTE | 0.998 | 0.998 | 0.998 |
| | PCA-SMOTE vs BMOTE | 0.1611 | 0.1377 | 0.2461 |
| | PCA-SMOTE vs ADASYN | 1 | 1 | 0.6523 |

**Tabela 5.22:** *Testes pareado de Wilcoxon realizados sobre as métricas de desempenho obtidas pelos modelos.*

A base de dados *Ecoli* processada para teste possui dimensões finais 336 $\times$ 8, sendo 35 amostras da classe minoritária "imU"(amostras de proteínas localizadas nas membranas celulares internas, sem sequência de sinal clivável) e as 301 amostras restantes pertencentes à classe *"other"* (amostras de proteínas em outras localizações), configurando desta forma um fator de desbalanceamento de aproximadamente 1:8,6. Os algoritmos de balanceamento incluíram 259 (PCA-SMOTE), 255 (SMOTE básico), 264 (*Borderline-SMOTE*), 267 (*Safe-Level-SMOTE*) e 186 (ADASYN) amostras sintéticas da classe minoritária na base de dados de teste.

As duas primeiras componentes principais extraídas pelo PCA explicaram juntas um total de 52.4% da variância total dos dados, de acordo com a Fig. 5.11. No gráfico bidimensional formado em termos de ambas componentes, vê-se que as amostras de dados

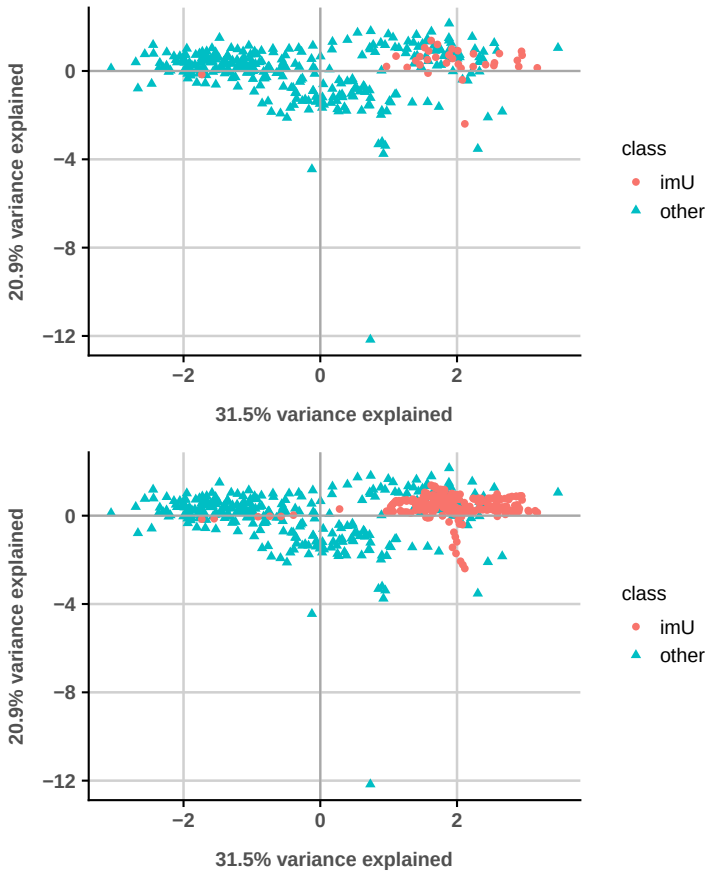



**Figura 5.11:** *Base de dados* ***Ecoli*** *expressa em termos de suas duas componentes principais antes e após a aplicação do PCA-SMOTE.*

de ambas as classes formam grupos distintos na região superior que se estendem pelos valores da primeira componente principal, sendo que as amostras da classe minoritária "imU"ocupam sumariamente a região direita do grupo, onde há sobreposição de dados com as amostras da outra classe, enquanto estas últimas ocupam sumariamente a região esquerda do gráfico com praticamente nenhuma sobreposição de dados. Por isso, espera-se que as amostras da classe majoritária sejam mais fáceis de serem aprendidas pelos classificadores do que as amostras da classe minoritária, não apenas devido ao desbalanceamento como também à sobreposição de dados.

De acordo com os testes de significância de Wilcoxon na Tabela 5.22, o algoritmo PCA-SMOTE mostrou valores de AUC e *G-mean* superiores aos obtidos por todos os demais algoritmos para todos os modelos avaliados. O algoritmo proposto também apresenta valores de *F-measure* superiores aos obtidos pelos algoritmos *Safe-Level-SMOTE* e *ADASYN* para o modelo SVM, e por todos os algoritmos de balanceamento para o modelo MLP. Além disso, de acordo com a Tabela 5.21, PCA-SMOTE apresentou valores superiores de acurácia e especificidade em relação a todos os algoritmos testados combinados com o modelo SVM e MLP.

### 5.4.12 Thyroid Disease

| Modelo | Medida | - | SMOTE | SLSMOTE | B-SMOTE | ADASYN | PCA-SMOTE |
|---|---|---|---|---|---|---|---|
| SVM | Acurácia (%) | 95.35 | 99.25 | 99.01 | 99.19 | 97.14 | **99.38** |
| | Sensitividade (%) | 68.33 | 99.95 | 98.72 | **100** | 96 | 99.89 |
| | Especificidade (%) | **99.73** | 98.54 | 99.3 | 98.38 | 98 | 98.86 |
| | AUC (%) | 0.84 | 0.99 | 0.99 | 0.99 | 0.97 | 0.99 |
| | G-mean (%) | 0.82 | 0.99 | 0.99 | 0.99 | 0.97 | 0.99 |
| | F-measure (%) | 0.79 | 0.99 | 0.99 | 0.99 | 0.97 | 0.99 |
| C4.5 | Acurácia (%) | 96.33 | 96.42 | 97.15 | 97.25 | **97.91** | 96.58 |
| | Sensitividade (%) | 85.67 | 97.22 | 97.81 | **98.17** | 96.93 | 97.15 |
| | Especificidade (%) | 98.05 | 95.62 | 96.49 | 96.32 | **98.65** | 96 |
| | AUC (%) | 0.92 | 0.96 | 0.97 | 0.97 | **0.98** | 0.97 |
| | G-mean (%) | 0.91 | 0.96 | 0.97 | 0.97 | **0.98** | 0.97 |
| | F-measure (%) | 0.87 | 0.96 | 0.97 | 0.97 | **0.98** | 0.97 |
| MLP | Acurácia (%) | 98.47 | 99 | 98.98 | 99 | 98.89 | **99.22** |
| | Sensitividade (%) | 94.33 | 99.84 | 99.68 | 99.84 | 100 | 100 |
| | Especificidade (%) | **99.14** | 98.16 | 98.27 | 98.16 | 98.05 | 98.43 |
| | AUC (%) | 0.97 | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 |
| | G-mean (%) | 0.97 | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 |
| | F-measure (%) | 0.94 | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 |

**Tabela 5.23:** *Métricas de desempenho calculadas para os modelos treinados com a base de dados **New Thyroid** sem balanceamento e balanceada de acordo com os algoritmos.*

| | Métodos | AUC | G-mean | F-measure |
|---|---|---|---|---|
| $p_{wilcoxon}$ SVM | PCA-SMOTE vs - | 0.0029 | 0.001 | 0.001 |
| | PCA-SMOTE vs SMOTE | 0.0599 | 0.021 | 0.1181 |
| | PCA-SMOTE vs SLSMOTE | 0.021 | 0.0165 | 0.0049 |
| | PCA-SMOTE vs BMOTE | 0.0217 | 0.0122 | 0.1162 |
| | PCA-SMOTE vs ADASYN | 0.001 | 0.001 | 0.001 |
| $p_{wilcoxon}$ C4.5 | PCA-SMOTE vs - | 0.001 | 0.001 | 0.001 |
| | PCA-SMOTE vs SMOTE | 0.5 | 0.5391 | 0.5391 |
| | PCA-SMOTE vs SLSMOTE | 0.985 | 0.9932 | 0.9756 |
| | PCA-SMOTE vs BMOTE | 0.9839 | 0.9863 | 0.9902 |
| | PCA-SMOTE vs ADASYN | 1 | 1 | 1 |
| $p_{wilcoxon}$ MLP | PCA-SMOTE vs - | 0.004 | 0.002 | 0.001 |
| | PCA-SMOTE vs SMOTE | 0.0493 | 0.117 | 0.0411 |
| | PCA-SMOTE vs SLSMOTE | 0.0391 | 0.033 | 0.0378 |
| | PCA-SMOTE vs BMOTE | 0.0685 | 0.0914 | 0.0626 |
| | PCA-SMOTE vs ADASYN | 0.0207 | 0.0286 | 0.0029 |

**Tabela 5.24:** *Testes pareado de Wilcoxon realizados sobre as métricas de desempenho obtidas pelos modelos.*

A base de dados *Thyroid Disease* processada para teste possui dimensões finais $215 \times 6$, sendo 30 amostras da classe minoritária "hypo"(amostras de pacientes portadores de hipotireoidismo) e as 185 amostras restantes pertencentes à classe *"other"* (amostras de pacientes normais ou portadores de hipertireoidismo), configurando desta forma um fator de desbalanceamento de aproximadamente 1:6. Os algoritmos de balanceamento incluíram 156 (PCA-SMOTE, SMOTE básico e *Borderline-SMOTE*), 157 (*Safe-Level-SMOTE*) e 110 (ADASYN) amostras sintéticas da classe minoritária na base de dados de teste.

As duas primeiras componentes principais extraídas pelo PCA explicaram juntas um total de 74.1% da variância total dos dados, de acordo com a Fig. 5.12. É possível

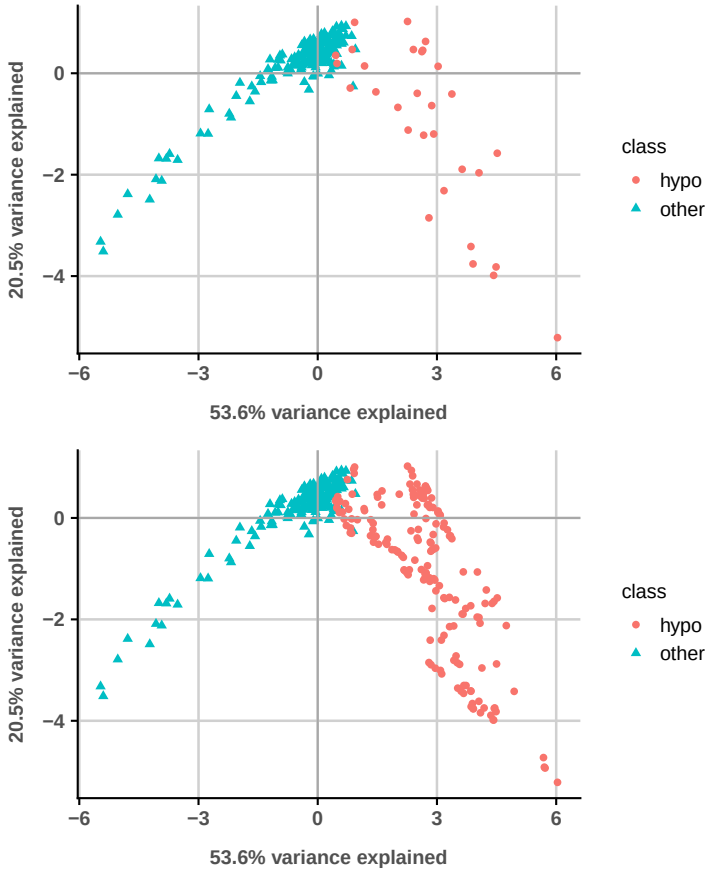



**Figura 5.12:** *Base de dados* ***Thyroid Disease*** *expressa em termos de suas duas componentes principais antes e após a aplicação do PCA-SMOTE.*

ver que as amostras de dados formam dois grupos distintos nas regiões esquerda e direita do gráfico, com praticamente nenhuma sobreposição de dados entre elas, mesmo após o balanceamento. Amostras da classe majoritária *"other"* se concentram principalmente na região superior do gráfico, tornando-se mais esparsar conforme os valores da primeira componente principal decrescem, enquanto as amostras da classe minoritária por sua vez estão espalhadas de maneira quase uniforme pela região direita do gráfico.

Este cenário favorece o aprendizado das amostras por parte dos classificadores, o que é confirmado pelos valores mostrados na Tabela 5.23, que mostra que os três modelos de classificação obtiveram alto desempenho mesmo quando treinados com a base de dados desbalanceada. O uso de algoritmos de balanceamento aprimora estes resultados. De acordo com os testes de significância de Wilcoxon na Tabela 5.24, o algoritmo PCA-SMOTE mostrou valores superiores dos que os demais algoritmos de balanceamento para os modelos SVM e MLP, para as três métricas de desempenho consultadas. Ainda de acordo com Tabela 5.23, para o modelo C4.5, os valores de desempenho obtidos pelo PCA-SMOTE são iguais àqueles obtidos pelos algoritmos *Safe-Level-SMOTE* e *Borderline-SMOTE*.

## 5.5 Discussão

| *Conjunto de dados* | *Modelo* | *Algoritmos com melhor desempenho* | | |
|---|---|---|---|---|
| | | *AUC* | *G-mean* | *F-measure* |
| Breast Cancer Coimbra | SVM | *PCA-SMOTE* | *PCA-SMOTE* | *PCA-SMOTE* |
| | C4.5 | *PCA-SMOTE* | *PCA-SMOTE* | *PCA-SMOTE* |
| | MLP | SLSMOTE | SLSMOTE | SLSMOTE |
| Arroz | SVM | ADASYN | ADASYN | ADASYN |
| | C4.5 | ADASYN | ADASYN | ADASYN |
| | MLP | Todos exceto SLSMOTE | Todos exceto SLSMOTE | SMOTE |
| Chocolate orgânico | SVM | *PCA-SMOTE* | *PCA-SMOTE* | *PCA-SMOTE* |
| | C4.5 | *PCA-SMOTE* | *PCA-SMOTE* | *PCA-SMOTE* |
| | MLP | SMOTE | SMOTE e B-SMOTE | SMOTE |
| Folhas de laranjeiras orgânicas | SVM | SMOTE | SMOTE | SMOTE |
| | C4.5 | SLSMOTE | SLSMOTE | SLSMOTE e B-SMOTE |
| | MLP | *PCA-SMOTE* | *PCA-SMOTE* | *PCA-SMOTE* |
| Suco de uva orgânico | SVM | SLSMOTE | SLSMOTE | SLSMOTE |
| | C4.5 | B-SMOTE, ADASYN e *PCA-SMOTE* | B-SMOTE, ADASYN e *PCA-SMOTE* | B-SMOTE |
| | MLP | *PCA-SMOTE* | *PCA-SMOTE* | *PCA-SMOTE* |
| Breast Cancer Wisconsin | SVM | *PCA-SMOTE* | *PCA-SMOTE* | *PCA-SMOTE* |
| | C4.5 | Todos | Todos | Todos |
| | MLP | Todos | Todos | Todos |
| Glass Identification | SVM | SLSMOTE | SLSMOTE | SLSMOTE |
| | C4.5 | SLSMOTE | SLSMOTE | SLSMOTE |
| | MLP | *PCA-SMOTE* | *PCA-SMOTE* | *PCA-SMOTE* |
| Wine Quality | SVM | SLSMOTE | SLSMOTE | SLSMOTE |
| | C4.5 | SLSMOTE | SLSMOTE | SLSMOTE |
| | MLP | ADASYN | ADASYN | SLSMOTE e ADASYN |
| Yeast | SVM | SLSMOTE | SLSMOTE | SLSMOTE |
| | C4.5 | SLSMOTE | SLSMOTE | SLSMOTE |
| | MLP | SLSMOTE | SLSMOTE | SLSMOTE |
| Pima Indians Diabetes | SVM | SLSMOTE | SLSMOTE | SLSMOTE |
| | C4.5 | SLSMOTE | SLSMOTE | SLSMOTE |
| | MLP | SLSMOTE | SLSMOTE | SLSMOTE |
| Ecoli | SVM | SLSMOTE e ADASYN | SLSMOTE | SLSMOTE |
| | C4.5 | SLSMOTE | SLSMOTE | SLSMOTE |
| | MLP | SLSMOTE e ADASYN | SLSMOTE e ADASYN | SLSMOTE |
| Thyroid Disease | SVM | *PCA-SMOTE* | *PCA-SMOTE* | *PCA-SMOTE* |
| | C4.5 | ADASYN | ADASYN | ADASYN |
| | MLP | *PCA-SMOTE* | *PCA-SMOTE* | *PCA-SMOTE* |

**Tabela 5.25:** *Melhores algoritmos de balanceamento estipulados de acordo com os modelos de classificação avaliados para cada base de dados de teste.*

Para todas as bases de dados testadas, é possível ver que os modelos de classificação apresentaram, em geral, valores de desempenho substancialmente melhores quando treinados com as bases de dados balanceadas pelo algoritmo proposto PCA-SMOTE do que quando treinados sobre as mesmas bases de dados em suas versões desbalanceadas. As exceções foram para os valores de especificidade, que para a maioria dos casos de teste (combinação de modelo de classificação com base de teste) foram maiores quando os modelos foram treinados com as bases desbalanceadas - mesmo em comparação com os demais algoritmos de balanceamento investigados. Desta forma, pode-se concluir que o algoritmo PCA-SMOTE cumpre o seu papel inicial de aprimorar o aprendizado dos modelos de classificação investigados e os seus desempenhos para classificar conjuntos de dados desbalanceados.

A Tabela 5.25 resume, para cada base de dados abordada, quais foram os melhores algoritmos de balanceamento investigados para cada modelo de classificação, considerando as métricas de desempenho AUC, *G-mean* e *F-measure*. Considera-se como o melhor algoritmo aquele que obteve os melhores valores de desempenho no maior número de casos de teste para cada base, ainda que para alguns casos de teste isolados outros algoritmos possam ter mostrado valores melhores. Na Tabela, verifica-se que o algoritmo proposto PCA-SMOTE foi o que teve o melhor desempenho em um número de casos de teste. Especificamente, o PCA-SMOTE combinado com o modelo SVM mostrou o melhor desempenho para as bases de dados *Breast Cancer Coimbra*, chocolate orgânico, *Breast Cancer Wisconsin* e *Thyroid Disease*. Combinado com o modelo C4.5, mostrou o melhor desempenho para as bases de dados *Breast Cancer Coimbra*, chocolate orgânico, suco de uva orgânico (em conjunto com o *Borderline-SMOTE* e ADASYN). Finalmente, combinado com o modelo MLP, mostrou o melhor desempenho para as bases de dados folhas de laranjeiras orgânicas, suco de uva orgânico, *Glass Identification* e *Thyroid Disease*.

É interessante notar que o PCA-SMOTE está entre os melhores algoritmos de balanceamento avaliados para um ou mais dos três modelos de classificação para as três bases de dados de produtos orgânicos analisadas (folhas de laranjeiras, sucos de uva e chocolate). Além do domínio, as três bases têm em comum: a natureza das variáveis descritoras, que são concentrações de elementos químicos encontrados nas amostras através ICP-MS; a quantidade de amostras (34, 27 e 36, respectivamente); fator de desbalanceamento, de aproximadamente 1:2; o alto grau de esparsidade dos dados, onde as amostras se distribuem por praticamente todo o gráfico bidimensional formado pelas duas primeiras componentes principais computadas pelo PCA; o baixo (praticamente nulo) grau de sobreposição de dados, entre outras possíveis características ocultas não identificadas neste estudo. Estima-se desta forma que o PCA-SMOTE tenha potencial favorável para balancear conjuntos de dados de amostras de produtos orgânicos,

principalmente aqueles que demonstrem estas mesmas características.

Com relação à natureza das variáveis descritoras, a base de dados *Glass Identification* também é composta por variáveis referentes a concentrações de elementos químicos, embora elas caracterizem amostras de vidros ao invés de produtos orgânicos. Coincidentemente, o algoritmo PCA-SMOTE foi o que mostrou melhores valores de desempenho quando combinado ao modelo MLP para predizer amostras neste conjunto de dados.

As bases de dados *Breast Cancer Coimbra* e *Thyroid Disease* têm quantidade similar de variáveis (6 e 10, respectivamente). As variáveis de ambas as bases dizem respeito a componentes que podem ser encontrados e medidos no sangue de pacientes: níveis de glucose, insulina, índice de Homa, leptina, adiponectina e resistina, para a base de dados *Breast Cancer Coimbra*, e níveis de T3, tiroxina sérica total, triiodotironina sérica total e de TSH para a base de dados *Thyroid Disease*. Em ambas as bases, o algoritmo PCA-SMOTE foi o que apresentou melhores valores de desempenho para dois de três modelos de classificação avaliados (sendo o SVM necessariamente um deles nos dois casos). Desta forma, estima-se que o PCA-SMOTE tenha um bom potencial para balancear conjuntos de dados deste domínio.

Todos estes testes foram conduzidos utilizando uma implementação inicial do algoritmo PCA-SMOTE embasado no uso do PCA como método de transformação dos dados, e o SMOTE tradicional para balanceamento dos dados transformados. A eficiência do método depende da resolução de algumas questões fundamentais.

A primeira delas refere-se à qual algoritmo de transformação de dados utilizar para os dados. Neste texto, foi apresentada uma versão que utiliza PCA para transformar as variáveis originais do conjuntos de dados em combinações lineares construídas de maneira a explicar a maior variância possível nos dados. Então o algoritmo SMOTE básico é utilizado para gerar amostras sintéticas da classe minoritária neste novo espaço transformado, e por fim as amostras geradas são processadas para o conjunto de variáveis original. Uma outra abordagem que poderia ser investigada seria o uso de funções de *kernel*. Modelos de classicação baseados em SVM utilizam estas funções para projetar um conjunto de dados, que originalmente não é linearmente separável, em outros espaços dimensionais onde as amostras das classes estejam linearmente separáveis. O objetivo da transformação de dados resume-se em encontrar novas variáveis para um conjunto de dados a partir das variáveis originais, contemplando relacionamentos existentes entre elas, onde as amostras das mesmas classes estejam distribuídas de maneira mais agrupada possível enquanto amostras de classes diferentes estejam o mais distantes possível entre si.

Outro fator a ser analisado é qual algoritmo de *oversampling* utilizar sobre os dados transformados. A princípio, utilizou-se a versão tradicional do SMOTE para gerar amostras sintéticas da classe minoritária para as bases de dados de teste. Entretanto, existe

na literatura uma infinidade de algoritmos de *oversampling* de dados que poderiam ser testados. Mesmo o SMOTE possui dezenas de extensões que implementam estratégias diferentes de escolha de amostras iniciais ou dos vizinhos mais próximos a serem considerados. O algoritmo *Borderline-SMOTE*, por exemplo, foca o *oversampling* nas regiões de dados mais próximas do limite de decisão, enquanto o *Safe-Level-SMOTE* foca nas regiões mais seguras e densas das amostras da classe minoritária.

Finalmente, é necessário considerar os modelos de classificação utilizados para testar a qualidade das amostras geradas a partir do algoritmo proposto. Nos testes discorridos neste texto, foram utilizados três modelos de classificação populares e amplamente utilizados na literatura para classificação de dados para vários propósitos e domínios: SVM, árvores de decisão baseadas em C4.5 e redes neurais artificiais baseadas em *Multilayer Perceptron*. Tais modelos implementam algoritmos de aprendizagem diferentes, cada um com suas vantagens e desvantagens específicas. Enquanto o objetivo do uso dos modelos neste texto foi expressar o desempenho do algoritmo PCA-SMOTE para balancear dados, e compará-lo a outros algoritmos de balanceamento, em um trabalho de análise de dados de uma base específica é necessário levar em consideração características da base de dados em análise (como, por exemplo, quantidade de variáveis e/ou amostras, tipo de dados, variabilidade e nível de sobreposição nos dados etc) para se escolher um modelo de classificação propício. Exemplos de outros modelos que poderiam ser investigados em par com transformação de dados são redes bayesianas, florestas aleatórias, métodos de *ensemble learning*. Assim como ocorre com métodos de balanceamento, há uma grande quantidade de modelos de classificação publicados na literatura, e uma grande quantidade de modificações propostas para os modelos já existentes. A parametrização é outro fator importante quando se trata de modelos de classificação. Nos testes conduzidos neste texto foi utilizada majoritariamente parametrização estática, isto é, os modelos C4.5 e MLP mostrados receberam valores fixos pré-determinados para os seus parâmetros. Em testes mais robustos, outros métodos de parametrização como busca em grade poderiam ser considerados, de maneira a se obter os modelos de fato de maior desempenho possível.

CAPÍTULO 6

# Considerações finais

Neste trabalho, propõe-se o uso de transformação de dados combinada com amostragem para aprimorar a aprendizagem de modelos de classificação sobre conjuntos de dados desbalanceados.

Numa versão inicial, tal abordagem foi implementada utilizando o método de análise dos componentes principais (PCA) para transformar os dados, e então o algoritmo de balanceamento SMOTE básico para realizar *oversampling* sobre as amostras transformadas. O método PCA utiliza ortogonalização de vetores e cálculo de autovetores e autovalores para transformar um conjunto de dados arbitrário em um novo conjunto de dados onde cada nova variável é uma combinação linear das variáveis originais, e tais combinações lineares são geradas de maneira a explicar a maior variância possível nos dados. Desta forma, a primeira componente principal (primeira variável do novo conjunto de dados) explica a maior variância existente nos dados, a segunda componente principal (segunda variável) explica a segunda maior variância e assim sucessivamente. Como resultado, o PCA gera um novo conjunto de dados expresso em termos de padrões existentes entre eles, em forma de combinações das contribuições de cada variável original [116]. Finalmente, o balanceamento através do SMOTE é realizado sobre estas amostras transformadas, e as amostras sintéticas geradas são convertidas para as variáveis originais.

Os resultados satisfatórios apresentados mostram o bom potencial que o algoritmo proposto batizado de PCA-SMOTE possui para melhorar o aprendizado de classificadores sobre bases de dados desbalanceadas, com o seu desempenho tendo superado os de outros algoritmos populares na literatura em vários casos de teste.

No decorrer do trabalho, algumas dificuldades foram enfrentadas, sendo a maior parte delas inerentes às limitações iniciais que o algoritmo proposto possui. Inicialmente, trabalha-se apenas com conjuntos de dados numéricos. Tanto o método PCA quanto o método SMOTE são limitados a este tipo de dados, por motivos diferentes: o PCA trabalha com ortogonalização de vetores e operações de transformações numéricas sobre os dados, enquanto o SMOTE utiliza medidas de distância (por padrão distâncias euclidianas) para identificar os vizinhos mais próximos de amostras escolhidas como ponto de partida do algoritmo. Desta forma, nenhum dos dois métodos originalmente possui suporte a

variáveis categóricas ou mais complexas, sendo necessário pré-processar esses tipos de variáveis quando encontradas em algum problema e transformá-las em numéricas. Uma melhoria a ser feita em versões futuras deste trabalho seria a inclusão de estratégias de transformação e balanceamento de dados que possam lidar de maneira satisfatória com a presença de dados categóricos. Além disso, neste trabalho, para a implementação inicial da abordagem proposta, apenas problemas de classificação binária foram investigados. Algumas bases de dados que originalmente contavam com vários rótulos de classe, como por exemplo *Thyroid Disease* e *Wine Quality*, tiveram suas variáveis de classe convertidas para variáveis categóricas binárias. Embora tanto o SMOTE básico quanto o PCA sejam conhecidos por suas capacidades de trabalhar com conjuntos de dados multi-classe, outros testes são necessários para confirmar o potencial do algoritmo proposto para estes tipos de bases.

Outro ponto a ser considerado em trabalhos futuros é a inclusão de seleção de variáveis no algoritmo de balanceamento. Seleção de variáveis refere-se a um processo de mineração de dados responsável por avaliar o poder discriminativo das variáveis descritoras, de maneira que variáveis mais relevantes para a tomada de decisão do classificador possam ser mantidas na análise enquanto variáveis consideradas redundantes ou irrelevantes possam ser descartadas. A remoção de variáveis que apresentam influência baixa ou nula na informação mapeada na classe pode render vantagens como melhora no desempenho de predição, redução de dimensionalidade, redução do tempo de construção e execução de modelos de classificação e outras [24, 42, 119]. O método PCA é comumente utilizado na literatura para este fim, uma vez que se trata de um redutor de dimensionalidade natural, onde variáveis com maior poder discriminativo (relacionadas à maior variância nos dados) tendem a receber altos coeficientes de combinação linear nas primeiras componentes principais computadas. Desta forma, seria interessante investigar o potencial do PCA para não apenas transformar os dados, como também estimar e remover variáveis desimportantes do conjunto de treinamento antes que o balanceamento de dados fosse aplicado nas amostras transformadas, provendo outros benefícios para os modelos de classificação além da melhora no desempenho de generalização. Outras abordagens, como o uso métodos filtro, também poderiam ser aplicadas. Métodos filtro são comumente utilizados como etapa de pre-processamento e visam a remoção de variáveis desimportantes de maneira independente e prévia à construção de modelos de classificação [125], são algoritmicamente simples e possuem baixo custo computacional. Uma sugestão seria utilizá-los para reduzir a dimensionalidade dos dados antes de prosseguir à etapa de transformação.

# Referências Bibliográficas


[1] ALCÁZAR, Á.; JURADO, J. M.; PALACIOS-MORILLO, A.; DE PABLOS, F.; MARTÍN, M. J. **Recognition of the geographical origin of beer based on support vector machines applied to chemical descriptors**. *Food Control*, 23(1):258 – 262, 2012.

[2] AMANKWAH-AMOAH, J.; ADOMAKO, S. **Big data analytics and business failures in data-rich environments: An organizing framework**. *Computers in Industry*, 105:204 – 212, 2019.

[3] ANSHARI, M.; ALMUNAWAR, M. N.; LIM, S. A.; AL-MUDIMIGH, A. **Customer relationship management and big data enabled: Personalization & customization of services**. *Applied Computing and Informatics*, 2018.

[4] BARATA, R. **Sampling Algorithms for Two-Class Imbalanced Data Sets**, 2019. https://rdrr.io/github/RomeroBarata/bimba/.

[5] BARBOSA, R. M.; BATISTA, B. L.; BARIÃO, C. V.; VARRIQUE, R. M.; COELHO, V. A.; CAMPIGLIA, A. D.; BARBOSA, F. **A simple and practical control of the authenticity of organic sugarcane samples based on the use of machine-learning algorithms and trace elements determination by inductively coupled plasma mass spectrometry**. *Food Chemistry*, 184:154–159, 10 2015.

[6] BARBOSA, R. M.; BATISTA, B. L.; VARRIQUE, R. M.; COELHO, V. A.; CAMPIGLIA, A. D.; BARBOSA, F. **The use of advanced chemometric techniques and trace element levels for controlling the authenticity of organic coffee**. *Food Research International*, 61:246–251, 7 2014.

[7] BATISTA, B. L.; DA SILVA, L. R. S.; ROCHA, B. A.; RODRIGUES, J. L.; BERRETTA-SILVA, A. A.; BONATES, T. O.; GOMES, V. S. D.; BARBOSA, R. M.; BARBOSA, F. **Multi-element determination in brazilian honey samples by inductively coupled plasma mass spectrometry and estimation of geographic origin with data mining techniques**. *Food Research International*, 49:09–215, 2012.

[8] BEYAN, C.; FISHER, R. **Classifying imbalanced data sets using similarity based hierarchical decomposition**. *Pattern Recognition*, 48:1653–1672, 2015.

[9] BISHOP, C. M. **Pattern Recognition and Machine Learning**. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[10] BREIMAN, L.; FRIEDMAN, J.; OLSHEN, R.; STONE, C. **Classification and Regression Trees**. Wadsworth and Brooks, Monterey, CA, 1984.

[11] BUNKER, R. P.; THABTAH, F. **A machine learning framework for sport result prediction**. *Applied Computing and Informatics*, 15(1):27 – 33, 2019.

[12] BUNKHUMPORNPAT, C.; SINAPIROMSARAN, K.; LURSINSAP, C. **Safe-level-smote: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem**. In: Theeramunkong, T.; Kijsirikul, B.; Cercone, N.; Ho, T.-B., editors, *Advances in Knowledge Discovery and Data Mining*, p. 475–482, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[13] CEBALLOS-MAGAÑA, S. G.; JURADO, J. M.; MUÑIZ-VALENCIA, R.; ALCÁZAR, A.; DE PABLOS, F.; MARTÍN, M. J. **Geographical authentication of tequila according to its mineral content by means of support vector machines**. *Food Analytical Methods*, 5(2):260–265, 2012.

[14] CHANDRASHEKAR, G.; SAHIN, F. **A survey on feature selection methods**. *Computers & Electrical Engineering*, 10(1):16–28, 2014.

[15] CHAWLA, N. V. **Data mining for imbalanced datasets: An overview**. In: Maimon, O.; Rokach, L., editors, *Data Mining and Knowledge Discovery Handbook*, p. 853–867. Springer US, Boston, MA, 2005.

[16] CHAWLA, N. V.; BOWYER, K. W.; HALL, L. O.; KEGELMEYER, W. P. **Smote: Synthetic minority over-sampling technique**. *J. Artif. Int. Res.*, 16(1):321–357, 2002.

[17] CHENG, J.; LIU, G.-Y. **Affective detection based on an imbalanced fuzzy support vector machine**. *Biomedical Signal Processing and Control*, 18:118–126, 2015.

[18] COOMANS, D.; BROECKAERT, I.; JONCKHEER, M.; MASSART, D. L. **Comparison of multivariate discrimination techniques for clinical data–application to the thyroid functional state**. *Methods of Information in Medicine*, p. 93–101, 1983.

[19] CORTES, C.; VAPNIK, V. **Support-vector networks**. *Machine Learning*, 20(3):273–297, 1995.

[20] CORTEZ, P.; CERDEIRA, A.; ALMEIDA, F.; MATOS, T.; REIS, J. **Modeling wine preferences by data mining from physicochemical properties**. *Decision Support Systems*, 47:547–553, 2009.

[21] COSTA, N. L.; CASTRO, I. A.; BARBOSA, R. M. **Classification of cabernet sauvignon from two different countries in south america by chemical compounds and support vector machines**. *Applied Artificial Intelligence*, 30:679–689, 2016.

[22] DA SILVA, I. B. V.; ADEODATO, P. J. L. **Pca and gaussian noise in mlp neural network training improve generalization in problems with small and unbalanced data sets**. In: *The 2011 International Joint Conference on Neural Networks*, p. 2664–2669, 2011.

[23] DAS, B.; KRISHNAN, N. C.; COOK, D. J. **Handling class overlap and imbalance to detect prompt situations in smart homes**. In: *Proceedings of the 2013 IEEE 13th International Conference on Data Mining Workshops*, ICDMW '13, p. 266–273, Washington, DC, USA, 2013. IEEE Computer Society.

[24] DASH, M.; LIU, H. **Feature selection for classification**. *Intelligent Data Analysis*, 1(1-4):131–156, 1997.

[25] DATTA, S.; DAS, S. **Near-bayesian support vector machines for imbalanced data classification with equal or unequal misclassification costs**. *Neural Networks*, 70:39–52, 2015.

[26] DELEN, D.; COGDELL, D.; KASAP, N. **A comparative analysis of data mining methods in predicting {NCAA} bowl outcomes**. *International Journal of Forecasting*, 28(2):543 – 552, 2012.

[27] DUA, D.; GRAFF, C. **UCI machine learning repository**. http://archive.ics.uci.edu/ml, 2017.

[28] DUDA, R. O.; HART, P. E.; STORK, D. G. **Pattern Classification (2Nd Edition)**. Wiley-Interscience, 2000.

[29] EXARCHOS, T.; RIGAS, G.; BIBAS, A.; KIKIDIS, D.; NIKITAS, C.; WUYTS, F.; IHTIJAREVIC, B.; MAES, L.; CENCIARINI, M.; MAURER, C.; MACDONALD, N.; BAMIOU, D.-E.; LUXON, L.; PRASINOS, M.; SPANOUDAKIS, G.; KOUTSOURIS, D.; FOTIADIS, D. **Mining balance disorders' data for the development of diagnostic decision support systems**. *Computers in Biology and Medicine*, 77:240 – 248, 2016.

[30] FAN, W.; STOLFO, S. J.; ZHANG, J.; CHAN, P. K. **Adacost: Misclassification cost-sensitive boosting**. In: *Proceedings of the Sixteenth International Conference on*

*Machine Learning*, ICML '99, p. 97–105, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[31] FAWCETT, T. **An introduction to roc analysis**. *Pattern Recognition Letters*, 27:861–874, 2006.

[32] FERNÁNDEZ, A.; GARCÍA, S.; HERRERA, F.; CHAWLA, N. V. **Smote for learning from imbalanced data: Progress and challenges, marking the 15-year anniversary**. *Journal of Artificial Intelligence Research*, 61:863–905, 2018.

[33] FILZMOSER, P.; HRON, K.; REIMANN, C. **Principal component analysis for compositional data with outliers**. *Environmetrics*, 20:621–632, 2009.

[34] FREUND, Y.; SCHAPIRE, R. E. **Experiments with a new boosting algorithm**. In: *Proceedings of the Thirteenth International Conference on International Conference on Machine Learning*, ICML'96, p. 148–156, San Francisco, CA, USA, 1996. Morgan Kaufmann Publishers Inc.

[35] GAIAD, J. E.; HIDALGO, M. J.; NE, R. N. V.; MARCHEVSKY, E. J.; PELLERANO, R. G. **Tracing the geographical origin of argentinean lemon juices based on trace element profiles using advanced chemometric techniques**. *Microchemical Journal*, 129:243–248, 2016.

[36] GARCÍA-PEDRAJAS, N.; DEL CASTILLO, J. A. R.; CERRUELA-GARCÍA, G. **A proposal for local k values for k-nearest neighbor rule**. *IEEE Transactions on Neural Networks and Learning Systems*, 28:470–475, 2017.

[37] GARCÍA, S.; LUENGO, J.; HERRERA, F. **Data Preprocessing in Data Mining**, volume 72 de **Intelligent Systems Reference Library**. Springer International Publishing, 1 edition, 2015.

[38] GARDNER, M.; DORLING, S. **Artificial neural networks (the multilayer perceptron) - a review of applications in the atmospheric sciences**. *Atmospheric Environment*, 32(14-15):2627–2636, 1998.

[39] GAYATHRI, G. V.; JYOTHI, B. S. **A comparative study of classification algorithms on spam detection**. *International Journal for Research in Applied Science & Engineering Technology*, 6, 2018.

[40] GHOLIPOOR, M.; NADALI, F. **Fruit yield prediction of pepper using artificial neural network**. *Scientia Horticulturae*, 250:249 – 253, 2019.

[41] GU, W.; FOSTER, K.; SHANG, J.; WEI, L. **A game-predicting expert system using big data and machine learning**. *Expert Systems with Applications*, 2019.

[42] GUYON, I.; ELISSEEFF, A. **An introduction to variable and feature selection**. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

[43] HAIXIANG, G.; YIJING, L.; SHANG, J.; MINGYUN, G.; YUANYUE, H. **Learning from class-imbalanced data: Review of methods and applications**. *Expert Systems with Applications*, 73:220–239, 2017.

[44] HAMEED, K.; CHAI, D.; RASSAU, A. **A comprehensive review of fruit and vegetable classificationÂ techniques**. *Image and Vision Computing*, 80:24 – 44, 2018.

[45] HAMEL, L. H. **Knowledge Discovery with Support Vector Machines**. Wiley-Interscience, New York, NY, USA, 2009.

[46] HAN, H.; WANG, W.-Y.; MAO, B.-H. **Borderline-smote: A new over-sampling method in imbalanced data sets learning**. In: *Proceedings of the 2005 International Conference on Advances in Intelligent Computing - Volume Part I*, ICIC'05, p. 878–887, Berlin, Heidelberg, 2005. Springer-Verlag.

[47] HE, H.; BAI, Y.; GARCIA, E. A.; LI, S. **Adasyn: Adaptive synthetic sampling approach for imbalanced learning**. In: *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, p. 1322–1328, 2008.

[48] HE, H.; GARCIA, E. A. **Learning from imbalanced data**. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, 2009.

[49] HIDALGO, M. J.; POZZI, M. T.; FURLONG, O. J.; MARCHEVSKY, E. J.; PELLERANO, R. G. **Classification of organic olives based on chemometric analysis of elemental data**. *Microchemical Journal*, 142:30–35, 2018.

[50] HOOGEVEEN, H.; TOMCZYK, J.; VAN DER ZANDEN, T. C. **Flower power: Finding optimal plant cutting strategies through a combination of optimization and data mining**. *Computers & Industrial Engineering*, 127:39 – 44, 2019.

[51] HORTON, P.; NAKAI, K. **A probabilistic classification system for predicting the cellular localization sites of proteins**. *Intelligent Systems in Molecular Biology*, 4:109–115, 1996.

[52] ITANI, S.; LECRON, F.; FORTEMPS, P. **Specifics of medical data mining for diagnosis aid: A survey**. *Expert Systems with Applications*, 118:300 – 314, 2019.

[53] IZENMAN, A. J. **Modern Multivariate Statistical Techniques: Regression, Classification, and Manifold Learning**. Springer Publishing Company, Incorporated, 1 edition, 2008.

[54] JACKSON, D. A.; CHEN, Y. **Robust principal component analysis and outlier detection with ecological data**. *Environmetrics*, 15:129–139, 2004.

[55] JIN, X.; XIDE LIN, C.; LUO, J.; HAN, J. **Socialspamguard: A data mining-based spam detection system for social media networks**. *Proceedings of the VLDB Endowment*, 4:1458–1461, 08 2011.

[56] JO, T.; JAPKOWICZ, N. **Class imbalances versus small disjuncts**. *ACM SIGKDD Explorations Newsletter*, 6:40–49, 2004.

[57] JORDAN, M. I.; THIBAUX, R. **Cs281bstat241b: Advanced topics in learning & decision making - the kernel trick**. http://www.cs.berkeley.edu/~jordan/courses/281B-spring04/lectures/lec3.pdf, 2004.

[58] JOTHI, N.; RASHID, N. A.; HUSAIN, W. **Data mining in healthcare – a review**. *Procedia Computer Science*, 72:306 – 313, 2015.

[59] JUNIOR, A. C. M.; MAIONE, C.; BARBOSA, R. M.; GALLIMBERTI, M.; PAULELLI, A. C. C.; SEGURA, F. R.; SOUZA, V. C.; BATISTA, B. L.; BARBOSA JR, F. **Elemental fingerprint profiling with multivariate data analysis to classify organic chocolate samples**. *Journal of Chemometrics*, 32(8):e3036, 2018.

[60] KARA, M. E.; FIRAT, S. U. O.; GHADGE, A. **A data mining-based framework for supply chain risk management**. *Computers & Industrial Engineering*, 2018.

[61] KAVAKIOTIS, I.; TSAVE, O.; SALIFOGLOU, A.; MAGLAVERAS, N.; VLAHAVAS, I.; CHOUVARDA, I. **Machine learning and data mining methods in diabetes research**. *Computational and Structural Biotechnology Journal*, 15:104 – 116, 2017.

[62] KIM, S.; KIM, H.; NAMKOONG, Y. **Ordinal classification of imbalanced data with application in emergency and disaster information services**. *IEEE Intelligent Systems*, 31:50–56, 2016.

[63] KOHAVI, R. **A study of cross-validation and bootstrap for accuracy estimation and model selection**. In: *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'95, p. 1137–1143, 1995.

[64] KOTSIANTIS, S. B.; KANELLOPOULOS, D.; PINTELAS, P. E. **Data preprocessing for supervised leaning**. *International Journal of Computer Science*, 1:1306–4428, 2006.

[65] KUMAR, R.; SHARMA, V. **Chemometrics in forensic science**. *TrAC Trends in Analytical Chemistry*, 105:191 – 201, 2018.

[66] LEUNG, C. K.; JOSEPH, K. W. **Sports data mining: Predicting results for the college football games**. *Procedia Computer Science*, 35:710 – 719, 2014.

[67] LI, K.; ZHANG, W.; LU, Q. **An improved smote imbalanced data classification method based on support degree**. In: *International Conference on Identification, Information and Knowledge in the Internet of Things*, IIKI 2014, p. 34–38, New Jersey, USA, 2014. IEEE.

[68] LIBRENZA-GARCIA, D.; KOTZIAN, B. J.; YANG, J.; MWANGI, B.; CAO, B.; LIMA, L. N. P.; BERMUDEZ, M. B.; BOEIRA, M. V.; KAPCZINSKI, F.; PASSOS, I. C. **The impact of machine learning techniques in the study of bipolar disorder: A systematic review**. *Neuroscience & Biobehavioral Reviews*, 80:538 – 554, 2017.

[69] LIU, H.; XU, J. H. **Web service framework research of data mining in e-business**. *Procedia Engineering*, 15:1968 – 1972, 2011.

[70] LIU, X.-Y.; WU, J.; ZHOU, Z.-H. **Exploratory undersampling for class-imbalance learning**. *Trans. Sys. Man Cyber. Part B*, 39(2):539–550, 2009.

[71] LOGAN, T.; MCLEOD, S.; GUIKEMA, S. **Predictive models in horticulture: A case study with royal gala apples**. *Scientia Horticulturae*, 209:201 – 213, 2016.

[72] LÓPEZ, V.; FERNÁNDEZ, A.; GARCÍA, S.; PALADE, V.; HERRERA, F. **An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics**. *Information Sciences*, 250:113–141, 2013.

[73] MAIONE, C.; BARBOSA, R. M. **Recent applications of multivariate data analysis methods in the authentication of rice and the most analyzed parameters: A review**. *Critical Reviews in Food Science and Nutrition*, 2019.

[74] MAIONE, C.; BATISTA, B. L.; CAMPIGLIA, A. D.; JR, F. B.; BARBOSA, R. M. **Classification of geographic origin of rice by data mining and inductively coupled plasma mass spectrometry**. *Computers and Electronics in Agriculture*, 121:101 – 107, 2016.

[75] MAIONE, C.; DE OLIVEIRA SOUZA, V. C.; TOGNI, L. R.; DA COSTA, J. L.; CAMPIGLIA, A. D.; JR, F. B.; BARBOSA, R. M. **Establishing chemical profiling for ecstasy tablets based on trace element levels and support vector machine**. *Neural Computing and Applications*, 30:947–955, 2018.

[76] MAIONE, C.; DE PAULA, E. S.; GALLIMBERTI, M.; BATISTA, B. L.; CAMPIGLIA, A. D.; JR, F. B.; BARBOSA, R. M. **Comparative study of data mining techniques for the authentication of organic grape juice based on icp-ms analysis**. *Expert Systems with Applications*, 49:60–73, 2016.

[77] MAIONE, C.; JR, F. B.; BARBOSA, R. M. **Predicting the botanical and geographical origin of honey with multivariate data analysis and machine learning techniques: A review**. *Computers and Electronics in Agriculture*, 157:436–446, 2019.

[78] MAIONE, C.; TURRA, C.; FERNANDES, E. A. D. N.; BACCHI, M. A.; JR, F. B.; BARBOSA, R. M. **Finding the most significant elements for the classification of organic orange leaves: A data mining approach**. *Analytical Letters*, 50:2292–2307, 2017.

[79] MANGASARIAN, O. L.; SETIONO, R.; WOLBERG, W. H. **Pattern recognition via linear programming: Theory and application to medical diagnosis**. In: Coleman, T. F.; Li, Y., editors, *Large-scale numerical optimization*, p. 22–30. SIAM Publications, Philadelphia, 1990.

[80] MANGASARIAN, O. L.; WOLBERG, W. H. **Cancer diagnosis via linear programming**. *SIAM News*, 23(5):1–18, 1990.

[81] M.BARBOSA, R.; DE PAULA, E. S.; PAULELLI, A. C.; MOORE, A. F.; SOUZA, J. M. O.; BATISTA, B. L.; CAMPIGLIA, A. D.; JR, F. B. **Recognition of organic rice samples based on trace elements and support vector machines**. *Journal of Food Composition and Analysis*, 45:95–100, 2016.

[82] MENARDI, G.; TORELLI, N. **Training and assessing classification rules with imbalanced data**. *Data Mining and Knowledge Discovery*, 28:92–122, 2014.

[83] MÉNDEZ, J. R.; NEZ, T. R.-Y.; RUANO-ORDÁS, D. **A new semantic-based feature selection method for spam filtering**. *Applied Soft Computing*, 76:89–104, 2019.

[84] NAEM, A. A.; GHALI, N. I.; SALEH, A. A. **Antlion optimization and boosting classifier for spam email detection**. *Future Computing and Informatics Journal*, 3:436–442, 2018.

[85] NAPIERALA, K.; STEFANOWSKI, J.; WILK, S. **Learning from imbalanced data in presence of noisy and borderline examples**. In: *Proceedings of the 7th International Conference on Rough Sets and Current Trends in Computing*, RSCTC'10, p. 158–167, Berlin, Heidelberg, 2010. Springer-Verlag.

[86] NAVADA, A.; ANSARI, A.; PATIL, S.; SONKAMBLE, B. **Overview of use of decision tree algorithms in machine learning**. In: *Control and System Graduate Research Colloquium (ICSGRC), 2011 IEEE*, p. 37–42, June 2011.

[87] NIGAM, K.; MCCALLUM, A. K.; THRUN, S.; MITCHELL, T. **Text classification from labeled and unlabeled documents using em**. *Machine Learning*, 39(2):103–134, 2000.

[88] ORRÙ, G.; PETTERSSON-YEO, W.; MARQUAND, A. F.; SARTORI, G.; MECHELLI, A. **Using support vector machine to identify imaging biomarkers of neurological and psychiatric disease: A critical review**. *Neuroscience & Biobehavioral Reviews*, 36(4):1140 – 1152, 2012.

[89] PATRÍCIO, M.; PEREIRA, J.; CRISÓSTOMO, J.; MATAFOME, P.; GOMES, M.; SEIÇA, R.; CARAMELO, F. **Using resistin, glucose, age and bmi to predict the presence of breast cancer**. *BMC Cancer*, 18:1, 2018.

[90] PERAL, J.; MATÉ, A.; MARCO, M. **Application of data mining techniques to identify relevant key performance indicators**. *Computer Standards & Interfaces*, 54:76 – 85, 2017.

[91] POLAT, K.; GÜNEŞ, S. **Breast cancer diagnosis using least square support vector machine**. *Digital Signal Processing*, 17(4):694 – 701, 2007.

[92] PRATI, R. C.; BATISTA, G. E. A. P. A.; MONARD, M. C. **Class imbalances versus class overlapping: An analysis of a learning system behavior**. In: Monroy, R.; Arroyo-Figueroa, G.; Sucar, L. E.; Sossa, H., editors, *MICAI 2004: Advances in Artificial Intelligence*, p. 312–321, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[93] PÉREZ-GODOY, M. D.; FERNÁNDEZ, A.; RIVERA, A. J.; DEL JESUS, M. J. **Analysis of an evolutionary rbfn design algorithm, co2rbfn, for imbalanced data sets**. *Pattern Recognition Letters*, 31:2375–2388, 2010.

[94] PRUSTY, M. R.; JAYANTHI, T.; VELUSAMY, K. **Weighted-smote: A modification to smote for event classification in sodium cooled fast reactors**. *Progress in Nuclear Energy*, 100:355–364, 2017.

[95] PYLE, D. **Data Preparation for Data Mining**. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 1999.

[96] QIN, H.; HUO, D.; ZHANG, L.; YANGA, L.; ZHANG, S.; YANG, M.; SHEN, C.; HOU, C. **Colorimetric artificial nose for identification of chinese liquor with different geographic origins**. *Food Research International*, 45:45–51, 2012.

[97] QUINLAN, J. R. **Induction of decision trees**. *Machine Learning*, 1:81–106, 1985.

[98] QUINLAN, J. R. **C4.5: Programs for Machine Learning**. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[99] RAD, M. R. N.; KOOHKAN, S.; FANAEI, H. R.; RAD, M. R. P. **Application of artificial neural networks to predict the final fruit weight and random forest to select important variables in native population of melon (cucumis melo l.)**. *Scientia Horticulturae*, 181:108 – 112, 2015.

[100] RADY, E.-H. A.; ANWAR, A. S. **Prediction of kidney disease stages using data mining algorithms**. *Informatics in Medicine Unlocked*, 15:100178, 2019.

[101] RAJ, V.; MAGG, S.; WERMTER, S. **Towards effective classification of imbalanced data with convolutional neural networks**. In: Schwenker, F.; Abbas, H. M.; Gayar, N. E.; Trentin, E., editors, *Artificial Neural Networks in Pattern Recognition*, Lecture Notes in Artificial Intelligence, p. 150–162, Ulm, Germany, 2016. Springer International Publishing.

[102] RICHARDSON, M. **Principal component analysis**. Special topic, University of Oxford, 05 2009.

[103] RODRÍGUEZ-BERMÚDEZ, R.; LÓPEZ-ALONSO, M.; MIRANDA, M.; FOUZ, R.; ORJALES, I.; HERRERO-LATORRE, C. **Chemometric authentication of the organic status of milk on the basis of trace element content**. *Food Chemistry*, 240:686–693, 2018.

[104] SÁEZ, J. A.; LUENGO, J.; STEFANOWSKI, J.; HERRERA, F. **Smote-ipf: Addressing the noisy and borderline examples problem in imbalanced classification by a re-sampling method with filtering**. *Information Sciences*, 291:184–203, 2015.

[105] SAHA, P.; ROY, N.; MUKHERJEE, D.; SARKAR, A. K. **Application of principal component analysis for outlier detection in heterogeneous traffic data**. *Procedia Computer Science*, 83:107–114, 2016.

[106] SANTOS, A. L.; GONZÁLEZ, J. A. T.; JIMÉNEZ, A. D. J. M.; RAMÍREZ, J. S.; NA URIBE, G. D. J. P.; MARTÍNEZ, O. V.; ÁNGEL GARCÍA MARÍN, M.; BARRIOS, J. L. G.; SALGADO, J. R. H.; ÁVILA, J. G. A. **Assessing the culture of fruit farmers from calvillo, aguascalientes, mexico with an artificial neural network: An approximation of sustainable land management**. *Environmental Science & Policy*, 92:311 – 322, 2019.

[107] SCHAPIRE, R. E.; SINGER, Y. **Improved boosting algorithms using confidence-rated predictions**. *Machine Learning*, 37:297–336, 1999.

[108] SCHNEIDER, K.-M. **A comparison of event models for naive bayes anti-spam e-mail filtering**. In: *Proceedings of the Tenth Conference on European Chapter of the Association for Computational Linguistics - Volume 1*, EACL '03, p. 307–314, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[109] SCHUBACH, M.; RE, M.; ROBINSON, P.; VALENTINI, G. **Imbalance-aware machine learning for predicting rare and common disease-associated non-coding variants**. *Scientific Reports*, 7, 2017.

[110] SEIFFERT, C.; KHOSHGOFTAAR, T. M.; HULSE, J. V.; FOLLECO, A. **An empirical study of the classification performance of learners on imbalanced and noisy software quality data**. *Information Sciences*, 259:571–595, 2014.

[111] SENG, J.-L.; CHEN, T. **An analytic approach to select data mining for business decision**. *Expert Systems with Applications*, 37(12):8042 – 8057, 2010.

[112] SERNEELS, S.; VERDONCK, T. **Principal component analysis for data containing outliers and missing elements**. *Computational Statistics & Data Analysis*, 52:1712–1727, 2008.

[113] SHARMA, A. K.; SAHNI, S. **A comparative study of classification algorithms for spam email data analysis**. *International Journal on Computer Science and Engineering*, 3:1890–1895, 2011.

[114] SHRIVASTAVA, S.; ANJU, R. **Spam mail detection through data mining techniques**. In: *2017 International Conference on Intelligent Communication and Computational Techniques (ICCT)*, p. 61–64, 2017.

[115] SINGARAVELAN, S.; ARUN, R.; ARUNSHUNMUGAM, D.; JOY, S. J. C.; MURUGAN, D. **Inner interruption discovery and defense system by using data mining**. *Journal of King Saud University - Computer and Information Sciences*, 2017.

[116] SMITH, L. I. **A tutorial on principal components analysis**. Computer science technical report no. oucs-2002-12, Department of Computer Science, University of Otago, 05 2002.

[117] STANIMIROVA, I.; DASZYKOWSKI, M.; WALCZAK, B. **Dealing with missing values and outliers in principal component analysis**. *Talanta*, 72:172–178, 2007.

[118] SUN, Y.; KAMEL, M. S.; WONG, A. K. C.; WANG, Y. **Cost-sensitive boosting for classification of imbalanced data**. *Pattern Recognition*, 40:3358–3378, 2007.

[119] TAN, P.-N.; STEINBACH, M.; KUMAR, V. **Introduction to Data Mining, (First Edition)**. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.

[120] TONG, S.; KOLLER, D. **Support vector machine active learning with applications to text classification**. *Journal of Machine Learning Research*, 2:45–66, 2002.

[121] TORKASHVAND, A. M.; AHMADI, A.; NIKRAVESH, N. L. **Prediction of kiwifruit firmness using fruit mineral nutrient concentration by artificial neural network (ann) and multiple linear regressions (mlr)**. *Journal of Integrative Agriculture*, 16:1634 – 1644, 2017.

[122] TURRA, C.; DE LIMA, M. D.; FERNANDES, E. A. D. N.; BACCHI, M. A.; JR, F. B.; BARBOSA, R. M. **Multielement determination in orange juice by icp-ms associated with data mining for the classification of organic samples**. *Information Processing in Agriculture*, 4:199–205, 2017.

[123] VARMA, S.; SIMON, R. **Bias in error estimation when using cross-validation for model selection**. *BMC Bioinformatics*, 7(91), 2006.

[124] VIDAKI, A.; KAYSER, M. **Recent progress, methods and perspectives in forensic epigenetics**. *Forensic Science International: Genetics*, 37:180 – 195, 2018.

[125] WESTON, J.; MUKHERJEE, S.; CHAPELLE, O.; PONTIL, M.; POGGIO, T.; VAPNIK, V. **Feature selection for svms**. In: Leen, T. K.; Dietterich, T.; Tresp, V., editors, *Proceedings of the 13th International Conference on Neural Information Processing Systems*, NIPS'00, p. 647–653, Cambridge, MA, USA, 2000. MIT Press.

[126] WILSON, D. L. **Asymptotic properties of nearest neighbor rules using edited data**. *Analyst*, SMC-2:408–421, 1972.

[127] WOLBERG, W. H.; MANGASARIAN, O. L. **Multisurface method of pattern separation for medical diagnosis applied to breast cytology**. *Proceedings of the National Academy of Sciences*, 87:9193–9196, 1990.

[128] WU, C.-C.; YEH, W.-C.; HSU, W.-D.; ISLAM, M. M.; NGUYEN, P. A.; POLY, T. N.; WANG, Y.-C.; YANG, H.-C.; LI, Y.-C. J. **Prediction of fatty liver disease using machine learning algorithms**. *Computer Methods and Programs in Biomedicine*, 170:23 – 29, 2019.

[129] XU, Y.; YANG, Z.; ZHANG, Y.; PAN, X.; WANG, L. **A maximum margin and minimum volume hyper-spheres machine with pinball loss for imbalanced data classification**. *Knowledge-Based Systems*, 95:75–85, 2016.

[130] ZHANG, C.; CHEN, Y.; LIU, X.; ZHAO, X. **Abstention-smote: An over-sampling approach for imbalanced data classification**. In: *Proceedings of the 2017 International Conference on Information Technology*, ICIT 2017, p. 17–21, New York, NY, USA, 2017. ACM.

[131] ZHANG, Y.; FU, P.; LIU, W.; CHEN, G. **Imbalanced data classification based on scaling kernel-based support vector machine**. *Neural Computing and Applications*, 25:927–935, 2014.

APÊNDICE A

# Código: pca-smote.R

```
balance_with_pcasmote <- function(data, pover, neigh) {

        m = length(data)

        ###############################################################################
        ### PCA
        ###############################################################################
        pca = prcomp(data[,-m], scale = FALSE)
        loadings = pca$rotation
        scores = pca$x
        prop.pca = pca$sdev^2/sum(pca$sdev^2)

        ###############################################################################
        ### Aplicar SMOTE nas amostras transformadas (scores) do PCA
        ###############################################################################

        # Construindo o conjunto de dados com amostras transformadas pelos PCs
        data_transformed = as.data.frame(scores)
        data_transformed$class = data[,m]

        # Aplicando SMOTE nos dados transformados
        data_transformed_balanced = bimba::SMOTE(data_transformed, perc_over = pover, k = neigh)

        syn = nrow(data_transformed_balanced) - nrow(data_transformed)
        print(paste("Amostras criadas:", syn, sep = " "))
        print(table(data_transformed_balanced$class))

        l = length(data_transformed_balanced)

        # Gráfico dos dados transformados (scores) antes do balanceamento
        before = ggplot(data_transformed) + geom_point(aes(data_transformed[,1],
                        data_transformed[,2], colour = class, shape = class)) +
                labs(x = paste(percent(prop.pca[1]), " variance explained", sep = ""),
                        y = paste(percent(prop.pca[2]), " variance explained", sep = "")) +
                theme(panel.grid.major=element_line(colour = "#D0D0D0", size = .6)) +
                theme(legend.text=element_text(size = 10)) +
                theme(legend.title=element_text(size = 10)) +
                theme(plot.title=element_text(face = "bold", hjust = -.08, vjust = 2,
                        colour = "#3C3C3C", size = 14)) +
                theme(axis.text.x=element_text(size = 10, colour = "#535353", face = "bold")) +
                theme(axis.text.y=element_text(size = 10, colour = "#535353", face = "bold")) +
                theme(axis.title.y=element_text(size = 10, colour = "#535353", face = "bold",
                        vjust = 1.5)) +
                theme(axis.title.x=element_text(size = 10, colour = "#535353", face = "bold",
```

```
45                          vjust = -.5)) +
46                  geom_vline(xintercept = 0, colour = "darkgray") +
47                  geom_hline(yintercept = 0, colour = "darkgray")
48
49          # Gráfico dos dados transformados (scores) depois do balanceamento
50          after = ggplot(data_transformed_balanced) + geom_point(aes(data_transformed_balanced[,1],
51                          data_transformed_balanced[,2], colour = class, shape = class)) +
52                  labs(x = paste(percent(prop.pca[1]), " variance explained", sep = ""),
53                          y = paste(percent(prop.pca[2]), " variance explained", sep = "")) +
54                  theme(panel.grid.major=element_line(colour = "#D0D0D0", size = .6)) +
55                  theme(legend.text=element_text(size = 10)) +
56                  theme(legend.title=element_text(size = 10)) +
57                  theme(plot.title=element_text(face = "bold", hjust = -.08, vjust = 2,
58                          colour = "#3C3C3C", size = 14)) +
59                  theme(axis.text.x=element_text(size = 10, colour = "#535353", face = "bold")) +
60                  theme(axis.text.y=element_text(size = 10, colour = "#535353", face = "bold")) +
61                  theme(axis.title.y=element_text(size = 10, colour = "#535353", face = "bold",
62                          vjust = 1.5)) +
63                  theme(axis.title.x=element_text(size = 10, colour = "#535353", face = "bold",
64                          vjust = -.5)) +
65                  geom_vline(xintercept = 0, colour = "darkgray") +
66                  geom_hline(yintercept = 0, colour = "darkgray")
67
68          print(plot_grid(before, after, labels = NULL, nrow = 2))
69
70          ##############################################################################
71          ### Revertendo o conjunto de dados transformado e balanceado para valores
72          ### descritos pelas variaveis originais
73          ##############################################################################
74          data_transformed_balanced2 = as.matrix(data_transformed_balanced[,-1])
75          data_reconstructed = data_transformed_balanced2 %*% t(loadings)
76          data_reconstructed = as.data.frame(data_reconstructed)
77          data_reconstructed$class = data_transformed_balanced[,1]
78          data_balanced_pcasmote = data_reconstructed
79
80          return (data_balanced_pcasmote)
81  }
```

APÊNDICE B

# Códigos auxiliares

## B.1 main.R

```
require(ggplot2)
require(ggfortify)
require(DMwR)
require(scales)
require(dplyr)
require(cowplot)
require(caret)
require(pROC)
library(bimba)
library(beepr)

source("pca-smote.R")
source("smotefunctions.R")
source("classificationfunctions.R")
source("performanceEval.R")

#################################################################################################
### Data input and processing
#################################################################################################

data = read.csv("datasets/breastcancercoimbra.csv", sep = ",", dec = ".")

m = length(data) # num. variables
n = nrow(data)   # num. samples
minorityclassvalue = "1"
positiveValue = minorityclassvalue

# Rename class column
data$class = as.factor(data[,m])
data[,m] = NULL

# Change missing values for column mean
#for(i in 1:(m-1)){
#  data[is.na(data[,i]), i] = mean(data[,i], na.rm = TRUE)
#}

# Scaling
data_scaled = scale(data[,-m])
data_scaled = as.data.frame(data_scaled)
data_scaled$class = data$class

```

```
table(data$class)

#################################################################################################
### Data balancing and testing
#################################################################################################

pover = 200   # Valor "perc.over" a ser passado para o SMOTE
neigh = 5     # Valor "k" a ser passado para o SMOTE

# Get balanced datasets according to each SMOTE extension
data_balanced_pcasmote = balance_with_pcasmote(data_scaled, pover, neigh) # PCA-SMOTE
data_balanced_smote = balance_with_smote(data_scaled, pover, neigh) # SMOTE
data_balanced_bordersmote = balance_with_smote(data_scaled, pover, neigh) # Borderline-SMOTE
data_balanced_safesmote = balance_with_safesmote(data_scaled, pover, neigh) # Safe-SMOTE
data_balanced_adasyn = balance_with_adasyn(data_scaled, pover, neigh) # ADASYN

# Testing balanced datasets with classification models
test_repeat = 10
source("main_classification.R")
beep("fanfare")
source("main_printresults1.R")
source("main_printsigntests.R")
```

## B.2 classificationfunctions.R

```
#################################################################################################
## Support vector machines
#################################################################################################

svm_classification <- function (data, test_repeat, positiveValue) {
        svm_performance = matrix(ncol = 6) # Metrics of the model to be returned

        # Repeating the 5-fold CV for test_repeat times
        for (i in 1:test_repeat) {
                print(paste(i, "/", test_repeat, sep = ""))
                # Overall metrics of the model for a single execution of the 5-fold CV
                performance = 0
                # Generating a data partition with 5 folds
                created_folds = createFolds(y = data$class, k = 5, list = TRUE)
                # Evaluating SVM for each test fold
                for(testfold in 1:5) {
                        test_data = created_folds[[testfold]]
                        test_data = data[test_data,]
                        train_data = created_folds[-testfold]
                        train_data = c(train_data[[1]], train_data[[2]], train_data[[3]],
                                train_data[[4]])
                        train_data = data[train_data,]

                        ctrl <- trainControl(method = "none")
                        model = train(class ~ ., data = train_data, method = "svmRadial")
                        performance = performance +
                                performanceEval(model, test_data, positiveValue)/5
                }
                # Calculating final metrics of the model after test_repeat executions
                svm_performance = rbind(svm_performance, t(as.matrix(performance)))
        }
```

```
        svm_performance = svm_performance[-1,]
        return (svm_performance)
}

#############################################################################################
## Multilayer Perceptron
#############################################################################################

mlp_classification <- function (data, test_repeat, positiveValue) {
        hunits = (n + 2)/2
        mlp_performance = matrix(ncol = 6) # Metrics of the model to be returned

        # Repeating the 5-fold CV for test_repeat times
        for (i in 1:test_repeat) {
                print(paste(i, "/", test_repeat, sep = ""))
                # Overall metrics of the model for a single execution of the 5-fold CV
                performance = 0
                # Generating a data partition with 5 folds
                created_folds = createFolds(y = data$class, k = 5, list = TRUE)
                # Evaluating SVM for each test fold
                for(testfold in 1:5) {
                        test_data = created_folds[[testfold]]
                        test_data = data[test_data,]
                        train_data = created_folds[-testfold]
                        train_data = c(train_data[[1]], train_data[[2]], train_data[[3]],
                                train_data[[4]])
                        train_data = data[train_data,]

                        ctrl <- trainControl(method = "none")
                        model = train(class ~ ., data = train_data, method = "mlp",
                        tuneGrid = expand.grid(.size=hunits))
                        performance = performance +
                                performanceEval(model, test_data, positiveValue)/5
                }
                # Calculating final metrics of the model after test_repeat executions
                mlp_performance = rbind(mlp_performance, t(as.matrix(performance)))
        }
        mlp_performance = mlp_performance[-1,]
        return (mlp_performance)
}

#############################################################################################
## J48
#############################################################################################

j48_classification <- function (data, test_repeat, positiveValue) {
        j48_performance = matrix(ncol = 6) # Metrics of the model to be returned
        confidenceFactor = 0.2
        minNumObj = 3

        # Repeating the 5-fold CV for test_repeat times
        for (i in 1:test_repeat) {
                print(paste(i, "/", test_repeat, sep = ""))
                # Overall metrics of the model for a single execution of the 5-fold CV
                performance = 0
                # Generating a data partition with 5 folds
                created_folds = createFolds(y = data$class, k = 5, list = TRUE)
                # Evaluating SVM for each test fold
```

```
90                  for(testfold in 1:5) {
91                          test_data = created_folds[[testfold]]
92                          test_data = data[test_data,]
93                          train_data = created_folds[-testfold]
94                          train_data = c(train_data[[1]], train_data[[2]], train_data[[3]],
95                                  train_data[[4]])
96                          train_data = data[train_data,]
97
98                          ctrl <- trainControl(method = "none")
99                          model = train(class ~ ., data = train_data, method = "J48",
100                                 tuneGrid = expand.grid(.C=confidenceFactor, .M=minNumObj))
101                         performance = performance +
102                                 performanceEval(model, test_data, positiveValue)/5
103                 }
104                 # Calculating final metrics of the model after test_repeat executions
105                 j48_performance = rbind(j48_performance, t(as.matrix(performance)))
106         }
107         j48_performance = j48_performance[-1,]
108         return (j48_performance)
109 }
```

## B.3 main_classification.R

```
1   ##############################################################################################
2   ### Testing balanced datasets with classification models
3   ##############################################################################################
4
5   test_balanced_datasets <- function (model) {
6           if (model == "svm")
7                   classification = svm_classification
8           else if (model == "mlp")
9                   classification = mlp_classification
10          else if (model == "j48")
11                  classification = j48_classification
12
13          print(paste("Evaluating imbalanced dataset with ", toupper(model), sep = ""))
14          # Training with imbalanced data
15          nobalance_performance = classification(data_scaled, test_repeat, positiveValue)
16          print(paste("Evaluating PCA-SMOTE dataset with ", toupper(model), sep = ""))
17          # Training data balanced by SMOTE-PCA
18          pcasmote_performance = classification(data_balanced_pcasmote, test_repeat, positiveValue)
19          print(paste("Evaluating SMOTE dataset with ", toupper(model), sep = ""))
20          # Training data balanced by common SMOTE
21          smote_performance = classification(data_balanced_smote, test_repeat, positiveValue)
22          print(paste("Evaluating SafeSMOTE dataset with ", toupper(model), sep = ""))
23          # Training data balanced by SLSMOTE
24          safesmote_performance = classification(data_balanced_safesmote, test_repeat, positiveValue)
25          print(paste("Evaluating BorderlineSMOTE dataset with ", toupper(model), sep = ""))
26          # Training data balanced by BorderlineSMOTE
27          bordersmote_performance = classification(data_balanced_bordersmote, test_repeat, positiveValue)
28          print(paste("Evaluating ADASYN dataset with ", toupper(model), sep = ""))
29          # Training data balanced by ADASYN
30          adasyn_performance = classification(data_balanced_adasyn, test_repeat, positiveValue)
31
32          # Compute mean accuracy for the test_repeat cross validations done
33          acc = data.frame(nobalance_performance[,1], smote_performance[,1], safesmote_performance[,1],
```

```
                bordersmote_performance[,1], adasyn_performance[,1], pcasmote_performance[,1])
        colnames(acc) = c("No balance", "SMOTE", "SLSMOTE", "Borderline-SMOTE", "ADASYN",
                "PCA-SMOTE")

        # Compute mean sensitivity for the test_repeat cross validations done
        sens = data.frame(nobalance_performance[,2], smote_performance[,2],
                safesmote_performance[,2], bordersmote_performance[,2], adasyn_performance[,2],
                pcasmote_performance[,2])
        colnames(sens) = c("No balance", "SMOTE", "SLSMOTE", "Borderline-SMOTE", "ADASYN",
                "PCA-SMOTE")

        # Compute mean specificity for the test_repeat cross validations done
        spec = data.frame(nobalance_performance[,3], smote_performance[,3],
                safesmote_performance[,3], bordersmote_performance[,3], adasyn_performance[,3],
                pcasmote_performance[,3])
        colnames(spec) = c("No balance", "SMOTE", "SLSMOTE", "Borderline-SMOTE", "ADASYN",
                "PCA-SMOTE")

        # Compute mean AUC for the test_repeat cross validations done
        auc = data.frame(nobalance_performance[,4], smote_performance[,4],
                safesmote_performance[,4], bordersmote_performance[,4], adasyn_performance[,4],
                pcasmote_performance[,4])
        colnames(auc) = c("No balance", "SMOTE", "SLSMOTE", "Borderline-SMOTE", "ADASYN",
                "PCA-SMOTE")

        # Compute mean g-mean for the test_repeat cross validations done
        gmean = data.frame(nobalance_performance[,5], smote_performance[,5],
                safesmote_performance[,5], bordersmote_performance[,5], adasyn_performance[,5],
                pcasmote_performance[,5])
        colnames(gmean) = c("No balance", "SMOTE", "SLSMOTE", "Borderline-SMOTE", "ADASYN",
                "PCA-SMOTE")

        # Compute mean f-measure for the test_repeat cross validations done
        fmeasure = data.frame(nobalance_performance[,6], smote_performance[,6],
                safesmote_performance[,6], bordersmote_performance[,6], adasyn_performance[,6],
                pcasmote_performance[,6])
        colnames(fmeasure) = c("No balance", "SMOTE", "SLSMOTE", "Borderline-SMOTE", "ADASYN",
                "PCA-SMOTE")

        measures = list(acc, sens, spec, auc, gmean, fmeasure)
        names(measures) = c("Accuracy", "Sensitivity", "Specificity", "AUC", "Gmean", "Fmeasure")

        return (measures)
}

# SVM
svm_measures = test_balanced_datasets(model = "svm")
svm_acc = svm_measures$Accuracy
svm_sens = svm_measures$Sensitivity
svm_spec = svm_measures$Specificity
svm_auc = svm_measures$AUC
svm_gmean = svm_measures$Gmean
svm_fmeasure = svm_measures$Fmeasure

# J48
j48_measures = test_balanced_datasets(model = "j48")
j48_acc = j48_measures$Accuracy
j48_sens = j48_measures$Sensitivity
```

```
92  j48_spec = j48_measures$Specificity
93  j48_auc = j48_measures$AUC
94  j48_gmean = j48_measures$Gmean
95  j48_fmeasure = j48_measures$Fmeasure
96
97  # MLP
98  mlp_measures = test_balanced_datasets(model = "mlp")
99  mlp_acc = mlp_measures$Accuracy
100 mlp_sens = mlp_measures$Sensitivity
101 mlp_spec = mlp_measures$Specificity
102 mlp_auc = mlp_measures$AUC
103 mlp_gmean = mlp_measures$Gmean
104 mlp_fmeasure = mlp_measures$Fmeasure
```

## B.4 main_printresults1.R

```
1   ##############################################################################################
2   ### Printing mean results in a single table (models x datasets)
3   ##############################################################################################
4
5   prettyprint <- function(df, metric) {
6           if (metric == "auc")
7                   print("AUC values")
8           else if (metric == "gmean")
9                   print("GMean values")
10          else if (metric == "f1")
11                  print("F-measure values")
12          else if (metric == "acc")
13                  print("Accuracy values")
14          else if (metric == "sens")
15                  print ("Sensitivity values")
16          else if (metric == "spec")
17                  print ("Specificity values")
18          print(df)
19  }
20
21  test_results_grid <- function (svm_measures, j48_measures, mlp_measures) {
22          results = data.frame(round(svm_measures, 2), round(j48_measures, 2),
23                  round(mlp_measures, 2))
24          colnames(results) = c("SVM", "J48", "MLP")
25          results = t(results)
26  }
27
28  # Accuracy
29  test_results_acc = test_results_grid(colMeans(svm_acc)*100, colMeans(j48_acc)*100,
30          colMeans(mlp_acc)*100)
31  prettyprint(test_results_acc, metric="acc")
32
33  # Sensitivity
34  test_results_sens = test_results_grid(colMeans(svm_sens)*100, colMeans(j48_sens)*100,
35          colMeans(mlp_sens)*100)
36  prettyprint(test_results_sens, metric="sens")
37
38  # Specificity
39  test_results_spec = test_results_grid(colMeans(svm_spec)*100, colMeans(j48_spec)*100,
40          colMeans(mlp_spec)*100)
```

```
prettyprint(test_results_spec, metric="spec")

# AUC
test_results_auc = test_results_grid(colMeans(svm_auc), colMeans(j48_auc),
        colMeans(mlp_auc))
prettyprint(test_results_auc, metric="auc")

# Gmean
test_results_gmean = test_results_grid(colMeans(svm_gmean), colMeans(j48_gmean),
        colMeans(mlp_gmean))
prettyprint(test_results_gmean, metric="gmean")

# F-measure
test_results_f1 = test_results_grid(colMeans(svm_fmeasure), colMeans(j48_fmeasure),
        colMeans(mlp_fmeasure))
prettyprint(test_results_f1, metric="f1")

write.table(test_results_acc, "overall_results.csv", col.names=TRUE, sep=",")
write.table(test_results_sens, "overall_results.csv", col.names=TRUE, sep=",", append=TRUE)
write.table(test_results_spec, "overall_results.csv", col.names=TRUE, sep=",", append=TRUE)
write.table(test_results_auc, "overall_results.csv", col.names=TRUE, sep=",", append=TRUE)
write.table(test_results_gmean, "overall_results.csv", col.names=TRUE, sep=",", append=TRUE)
write.table(test_results_f1, "overall_results.csv", col.names=TRUE, sep=",", append=TRUE)
```

## B.5 main_printsigntests.R

```
compute.wilcoxon.tests <- function(measures) {
        nobalance.versus.pcasmote = wilcox.test(measures$`No balance`, measures$`PCA-SMOTE`,
                paired = TRUE, alternative = "less")
        smote.versus.pcasmote = wilcox.test(measures$SMOTE, measures$`PCA-SMOTE`,
                paired = TRUE, alternative = "less")
        slsmote.versus.pcasmote = wilcox.test(measures$SLSMOTE, measures$`PCA-SMOTE`,
                paired = TRUE, alternative = "less")
        bsmote.versus.pcasmote = wilcox.test(measures$`Borderline-SMOTE`, measures$`PCA-SMOTE`,
                paired = TRUE, alternative = "less")
        adasyn.versus.pcasmote = wilcox.test(measures$ADASYN, measures$`PCA-SMOTE`,
                paired = TRUE, alternative = "less")

        wilcoxon.tests = c(nobalance.versus.pcasmote$p.value, smote.versus.pcasmote$p.value,
                slsmote.versus.pcasmote$p.value, bsmote.versus.pcasmote$p.value,
                adasyn.versus.pcasmote$p.value)
        names(wilcoxon.tests) = c("versus No Balance", "versus SMOTE", "versus SLSMOTE",
                "versus BSMOTE", "versus ADASYN")
        return (wilcoxon.tests)
}

##########################################################################################
# Computing two-sample Wilcoxon signed rank tests on AUC values obtained by SVM
##########################################################################################

auc.wilcoxon.tests = compute.wilcoxon.tests(svm_auc) # AUC values
gmean.wilcoxon.tests = compute.wilcoxon.tests(svm_gmean) # G-mean values
fmeasure.wilcoxon.tests = compute.wilcoxon.tests(svm_fmeasure) # F-measure values

# Formatting results
svm.wilcoxon.tests = data.frame(AUC = round(auc.wilcoxon.tests, 4),
```

```
        Gmean = round(gmean.wilcoxon.tests, 4),
        F1 = round(fmeasure.wilcoxon.tests, 4))

##############################################################################################
# Computing two-sample Wilcoxon signed rank tests on AUC values obtained by J48
##############################################################################################

auc.wilcoxon.tests = compute.wilcoxon.tests(j48_auc) # AUC values
gmean.wilcoxon.tests = compute.wilcoxon.tests(j48_gmean) # G-mean values
fmeasure.wilcoxon.tests = compute.wilcoxon.tests(j48_fmeasure) # F-measure values

# Formatting results
j48.wilcoxon.tests = data.frame(AUC = round(auc.wilcoxon.tests, 4),
        Gmean = round(gmean.wilcoxon.tests, 4),
        F1 = round(fmeasure.wilcoxon.tests, 4))

##############################################################################################
# Computing two-sample Wilcoxon signed rank tests on AUC values obtained by MLP
##############################################################################################

auc.wilcoxon.tests = compute.wilcoxon.tests(mlp_auc) # AUC values
gmean.wilcoxon.tests = compute.wilcoxon.tests(mlp_gmean) # G-mean values
fmeasure.wilcoxon.tests = compute.wilcoxon.tests(mlp_fmeasure) # F-measure values

# Formatting results
mlp.wilcoxon.tests = data.frame(AUC = round(auc.wilcoxon.tests, 4),
                                Gmean = round(gmean.wilcoxon.tests, 4),
                                F1 = round(fmeasure.wilcoxon.tests, 4))


write.table(svm.wilcoxon.tests, "wilcoxon_tests.csv", col.names=TRUE, sep=",")
write.table(j48.wilcoxon.tests, "wilcoxon_tests.csv", col.names=TRUE, sep=",",
        append=TRUE)
write.table(mlp.wilcoxon.tests, "wilcoxon_tests.csv", col.names=TRUE, sep=",",
        append=TRUE)

rm(auc.wilcoxon.tests, gmean.wilcoxon.tests, fmeasure.wilcoxon.tests)
```

## B.6 performanceEval.R

```
performanceEval <- function(model, test_data, positiveValue) {
        pred = predict(model, test_data)
        cf = confusionMatrix(pred, test_data$class, mode = "everything",
                positive = positiveValue)
        auc = pROC::auc(as.numeric(test_data$class), as.numeric(pred))
        accuracy = cf$overall[1]
        sensitivity = cf$byClass[1]
        specificity = cf$byClass[2]
        precision = ifelse(cf$byClass[5] == "NaN", 0, cf$byClass[5])
        recall = cf$byClass[6]
        fmeasure = ifelse((precision == 0 || recall == 0), 0,
                ((2*precision*recall)/(precision+recall)))
        gmean = sqrt(sensitivity*specificity)

        perfmeasures = c(accuracy, sensitivity, specificity, auc, gmean, fmeasure)
        names(perfmeasures) = c("Accuracy", "Sensitivity", "Specificity",
```

```
                "AUC", "GMean", "F-measure")
        return (perfmeasures)
}
```

## B.7 smotefunctions.R

```
#############################################################################################
# SMOTE
#############################################################################################

balance_with_smote <- function(data, pover, neigh) {

        smote = bimba::SMOTE(data, perc_over = pover, k = neigh)
        syn = nrow(smote) - nrow(data)
        print(paste("Amostras criadas:", syn, sep = " "))
        print(table(smote$class))
        return (smote)
}

#############################################################################################
# Borderline-SMOTE
#############################################################################################

balance_with_borderlinesmote <- function(data, pover, neigh) {

        bordersmote = bimba::BDLSMOTE(data, perc_over = pover, k = neigh,
                m = 2 * (neigh + 1), borderline = 1)
        syn = nrow(bordersmote) - nrow(data)
        print(paste("Amostras criadas:", syn, sep = " "))
        print(table(bordersmote$class))
        return (bordersmote)
}

#############################################################################################
# Safe-SMOTE
#############################################################################################

balance_with_safesmote <- function(data, pover, neigh) {

        safesmote = bimba::SLSMOTE(data, perc_over = pover, k = neigh)
        syn = nrow(safesmote) - nrow(data)
        print(paste("Amostras criadas:", syn, sep = " "))
        print(table(safesmote$class))
        return (safesmote)
}


#############################################################################################
# ADASYN
#############################################################################################

balance_with_adasyn <- function(data, pover, neigh) {

        adasyn = bimba::ADASYN(data, perc_over = pover, k = neigh)
        syn = nrow(adasyn) - nrow(data)
        print(paste("Amostras criadas:", syn, sep = " "))
```

```
51          print(table(adasyn$class))
52          return (adasyn)
53  }
```