Instituto de MATEMÁTICA E ESTATÍSTICA UFRGS

DEPARTAMENTO DE ESTATÍSTICA
E-mail: dest@mat.ufrgs.br

Trabalho de Conclusão de Curso

Uso de estratégias de reamostragem para correção de desbalanceamento entre classes em modelos de classificação

Demian B. O. Grams

25 de agosto de 2024

Demian B. O. Grams

Uso de estratégias de reamostragem para correção de desbalanceamento entre classes em modelos de classificação

Trabalho de Conclusão apresentado à comissão de Graduação do Departamento de Estatística da Universidade Federal do Rio Grande do Sul, como parte dos requisitos para obtenção do título de Bacharel em Estatística.

Orientador: Prof. Dr. João Henrique Ferreira Flores

Porto Alegre
Agosto de 2024

Demian B. O. Grams

Uso de estratégias de reamostragem para correção de desbalanceamento entre classes em modelos de classificação

Este Trabalho foi julgado adequado para obtenção dos créditos da disciplina Trabalho de Conclusão de Curso em Estatística e aprovado em sua forma final pela Orientador e pela Banca Examinadora.

Orientador:_____
Prof. Dr. João Henrique Ferreira Flores, UFRGS
Doutor pela Universidade Federal do Rio Grande do Sul, Porto Alegre, RS

Banca Examinadora:

Prof. Dr. João Henrique Ferreira Flores, UFRGS
Doutor pela Universidade Federal do Rio Grande do Sul, Porto Alegre, RS

Prof. Dr. Rodrigo Citton Padilha dos Reis, UFRGS
Doutor pela Universidade Federal de Minas Gerais, Belo Horizonte, MG

Porto Alegre
Agosto de 2024

Agradecimentos

Sou grato aos colegas, professores, e ao meu orientador, João Henrique F. Flores.

Agradeço ao meu pai, por todo incentivo e pelos conselhos.

À minha mãe, por acreditar em mim mais do que ninguém.

Aos meus amigos, pela parceria e pelas histórias que teremos para contar.

Em especial, agradeço e dedico este trabalho à minha avó, Professora Marina, por ter me ensinado, desde pequeno, o valor do estudo e da educação.

Resumo

A estatística tem como objeto central de estudo dados provenientes de eventos estocásticos, ou seja, de situações que envolvem incerteza ou aleatoriedade. Um tópico importante na área são os dados categóricos, onde, na prática, não é raro depararmonos com casos nos quais os eventos de interesse ocorrem com pouca frequência. Como exemplo, fraudes em cartão de crédito, diagnóstico de doenças raras e detecção de *spam*. Nestas situações ocorre o chamado problema de dados desbalanceados, que culminou no compo de *imbalanced learning*. O presente trabalho investiga o uso dos modelos de regressão logística e *Support Vector Machines* (SVMs) para fins de classificação em dados com classes desbalanceadas, em especial, no que diz respeito aos métodos de reamostragem quando utilizados no treinamento dos modelos, como forma de correção do desbalanceamento entre classes, e seu impacto na qualidade do ajuste, performance preditiva e calibração. Em particular, abordaremos estratégias de sobreamostragem, sendo elas a *Synthetic Minority Oversampling Technique* (SMOTE), *Borderline*-SMOTE1, a estratégia *Adaptive Synthetic* (ADASYN) e o próprio método de *Random Oversampling* (ROS), bem como as estratégias de subamostragem como *Random Undersampling* (RUS), *Cluster Centroids* e *Edited Nearest Neighbours* (ENN). Para avaliar o desempenho preditivo dos modelos optamos por utilizar como métricas a área sob a curva característica de operação do receptor (ROC AUC), *score* de Brier, precisão, *recall* e o F_1 *score*. Parte do trabalho preocupa-se com a calibração dos modelos, que será avaliada por meio de gráficos de calibração e o *score* de Brier.

Palavras-Chave: Aprendizado Supervisionado, Calibração, Classificação, Dados Desbalanceados, Reamostragem.

Abstract

Statistics has as its main subject of study data derived from stochastic events, meaning situations involving uncertainty or randomness. An important topic in the field is categorical data, where it is not uncommon to encounter cases in which the events of interest occur infrequently. Examples include credit card fraud, the diagnosis of rare diseases, and spam detection. In these situations, the so-called problem of imbalanced data arises, which has led to the development of the field known as imbalanced learning. This study investigates the use of logistic regression models and Support Vector Machines (SVMs) for classification in datasets with imbalanced classes, particularly regarding the use of resampling methods as a data-level solution for class imbalance correction, and their impact on model fit, predictive performance, and calibration. Specifically, we will discuss oversampling strategies such as the Synthetic Minority Oversampling Technique (SMOTE), Borderline-SMOTE, the Adaptive Synthetic (ADASYN) sampling strategy, and Random Oversampling (ROS), as well as undersampling strategies such as Random Undersampling (RUS), Cluster Centroids, and Edited Nearest Neighbours (ENN). To evaluate the predictive performance of the models, we chose as metrics the area under the Receiver Operating Characteristic Curve (ROC AUC), Brier score, precision, recall, and the F1 score. Part of the study focuses on model calibration, which will be assessed through calibration plots and the Brier score.

Keywords: Calibration, Classification, Imbalanced Data, Resampling, Supervised Learning.

Sumário

1 Introdução **11**
1.1 Considerações Iniciais . . . 11
1.2 Motivação . . . 12
1.3 Objetivos . . . 13

2 Fundamentação Teórica **14**
2.1 Notação e Definições Básicas . . . 14
2.2 Dados Desbalanceados . . . 15
2.3 Modelos de Classificação . . . 15
2.3.1 Classificadores Probabilísticos . . . 16
2.3.2 Perda e Risco . . . 16
2.3.3 Regressão Logística . . . 17
2.3.4 *Support Vector Machines* (SVMs) . . . 18
2.4 Métodos de Reamostragem . . . 20
2.4.1 Validação Cruzada . . . 21
2.4.2 *Oversampling* e *Undersampling* . . . 22
2.4.3 SMOTE . . . 22
2.4.4 *Borderline*-SMOTE1 . . . 24
2.4.5 ADASYN . . . 24
2.4.6 *Edited Nearest Neighbours (ENN)* . . . 25
2.4.7 *Cluster Centroids* . . . 26
2.5 Métricas de Desempenho . . . 26
2.5.1 Métricas Baseadas na Matriz de Confusão . . . 27
2.5.2 F-*score* . . . 27
2.5.3 Curva ROC . . . 28
2.5.4 Calibração . . . 29
2.5.5 *Score* de Brier . . . 30

3 Resultados **31**
3.1 Introdução . . . 31
3.2 Impacto da Reamostragem na Distribuição das Classes . . . 33
3.3 Impacto da Reamostragem nos Modelos de Classificação . . . 34
3.4 Desempenho no Treinamento . . . 34
3.5 Desempenho nos Dados de Teste . . . 36
3.6 Calibração nos Dados de Teste . . . 38

4 Considerações Finais **40**

Lista de Figuras

2.1 Conjunto de pontos num espaço bidimensional e diferentes retas candidatas a separadora das classes. (Cyc, Public domain, via Wikimedia Commons, 2008) 18
2.2 Exemplo de SVM com transformação de kernel em dados não linearmente separáveis no espaço original. (Machine Learner, CC BY-SA 4.0, via Wikimedia Commons, 2014) 20
2.3 Ilustração do método de validação cruzada com 5 lotes. 21
2.4 Exemplo do algoritmo SMOTE criando uma nova observação sintética. 23
2.5 Ilustração do *tradeoff* entre viés e variância no erro total do modelo. 27
2.6 Exemplo de curvas ROC de diferentes modelos em dados simulados. 29

3.1 Plots dos possíveis pares dentre as 4 covariáveis geradas pela função `make_classification` de acordo com os parâmetros fornecidos. 32
3.2 Impacto da reamostragem na distribuição dos dados, usando as covariáveis X_3 e X_4 como exemplo. De cima para baixo, da esquerda para a direita, temos: Os dados originais, ROS, SMOTE, *Borderline*-SMOTE1, ADASYN, RUS, *Cluster Centroids*, ENN, e SMOTEENN. 33
3.3 Impacto da reamostragem na região de decisão do modelo logístico considerando as variáveis X_3 e X_4. 35
3.4 Curvas de calibração do modelo logístico sob as diferentes estratégias de reamostragem nos dados de teste. 38
3.5 Curvas de calibração do modelo SVC sob as diferentes estratégias de reamostragem nos dados de teste. 39

Lista de Tabelas

2.1 Matriz de confusão no caso binário. 15
2.2 Exemplo fictício de *dataset* e *score* produzido por um classificador, junto da classe predita caso o limiar de decisão seja $\lambda = 0.8$. . . . 28

3.1 Sumário descritivo dos dados simulados pela função `make_classification` de acordo com os parâmetros fornecidos. 32
3.2 Distribuição das classes sob cada método de reamostragem no conjunto de treinamento. 34
3.3 Comparação de desempenho do modelo logístico sob diferentes estratégias de reamostragem. Médias e erros padrão (em parênteses) obtidos por meio de validação cruzada 5-*fold* estratificada. . 36
3.4 Comparação de desempenho do SVC sob diferentes estratégias de reamostragem. Médias e erros padrão (em parênteses) obtidos por meio de validação cruzada 5-*fold* estratificada. 36
3.5 Desempenho dos modelos e estratégias de reamostragem nos dados de teste. 37

1 Introdução

1.1 Considerações Iniciais

No contexto de dados categóricos, em especial no desenvolvimento de modelos preditivos (de classificação) o presente trabalho busca investigar o impacto das estratégias de reamostragem quando utilizadas como solução para o problema de classes desbalanceadas na fase de pré-processamento dos dados. Serão considerados o modelo de regressão logística e um classificador de *Support Vector Machine* (SVM), também chamado *Support Vector Classifier* (SVC), treinados em diferentes conjuntos de dados gerados a partir de diferentes algoritmos de reamostragem, por fim avaliando a performance dos modelos em termos de métricas estabelecidas na literatura.

Os artigos que mais se aproximam do tópico deste trabalho e que serviram de base foram He (2011), He e Garcia (2009) e Sun et al. (2009) pois tratam de revisões da área de classificação de dados desbalanceados, e van den Goorbergh et al. (2022), Tarawneh et al. (2022), Batista et al. (2004), Tarimo et al. (2021), Varotto et al. (2021) pois, de forma similar ao presente trabalho, buscam implementar e comparar métodos de reamostragem quando utilizados no treinamento de modelos de classificação no caso de dados desbalanceados.

Os pacotes *imbalanced-learn* (Lemaître et al., 2017) e *scikit-learn* (Pedregosa et al., 2011), bem como suas documentações, possibilitaram o desenvolvimento do trabalho pois contém uma vasta gama de funcionalidades para implementar modelos estatísticos e de aprendizagem de máquina, implementar os métodos de reamostragem, simular dados, gerar visualizações, entre outras coisas.

Alguns artigos fundamentais são Cortes e Vapnik (1995) que introduz as SVMs, Chawla et al. (2002) que introduz o algoritmo *Synthetic Minority Oversampling Technique* (SMOTE), He et al. (2008) que introduz a estratégia *Adaptive Synthetic* (ADASYN) e Han et al. (2005) por conta dos métodos de *Borderline*-SMOTE. O modelo de *k-nearest neighbours*, embora não seja abordado diretamente, é essencial na maioria dos algoritmos de reamostragem. Por fim, num escopo mais geral, foram referência alguns livros sobre modelos preditivos e aprendizado de máquina, sendo eles Izbicki e dos Santos (2020), James et al. (2013), Kuhn e Johnson (2013), Sugiyama (2016) e Murphy (2012).

Os dados utilizados foram gerados a partir da função `make_classification` do pacote *scikit-learn*, que permite ao usuário uma grande flexibilidade na geração de observações com resposta categórica. Para fins de reprodutibilidade foi utilizada uma semente de valor 42 para gerar de números pseudoaleatórios. Todas as análises

foram feitas utilizando a linguagem de programação Python, na versão 3.10, usando a plataforma Google Colaboratory.

1.2 Motivação

A situação de dados desbalanceados ocorre quando uma das classes é observada com frequência consideravelmente menor que as outras. No caso binário, um exemplo comum dessa situação é a classificação de transações de cartão de crédito em fraudulentas ou legítimas. Como a vasta maioria das transações não é fraude, um modelo que desconsidera a informação contida nos dados e classifica todas instâncias como pertencentes à classe majoritária (chamado classificador trivial) terá uma acurácia muito alta. Portanto em dados desse tipo, é possível que modelos ruins tenham performance boa, dependendo da métrica considerada. Nas palavras de He e Ma (2013), "[...] raridade relativa/classes desbalanceadas são um problema somente porque algoritmos de aprendizado não conseguem lidar efetivamente com dados deste tipo".

Os métodos de reamostragem, de acordo com James et al. (2013), tratam da obtenção de repetidas amostras de um conjunto de dados de treinamento e o ajuste do modelo de interesse em cada amostra, de modo a obter informações a mais sobre o modelo ajustado. Especificamente no contexto de classes desbalanceadas, o uso da reamostragem busca artificialmente equilibrar a distribuição da variável resposta, permitindo que os algoritmos/modelos detectem melhor os padrões da classe minoritária.

Na área de *imbalanced learning*, não há um consenso sobre o uso de estratégias de reamostragem como forma de pré-processamento dos dados no treinamento de modelos de classificação. Tanto em artigos de cunho técnico ou em materiais para fins aplicados, existe uma discordância sobre as vantagens e desvantagens desses métodos quando utilizados na mitigação do desbalanceamento entre classes. Por exemplo, na documentação oficial de bibliotecas como *imbalanced-learn* (Lemaître et al., 2017), tutoriais do TensorFlow, e nos diversos artigos introduzindo os diferentes métodos (SMOTE, ADASYN etc), é demonstrada uma melhora em diversas métricas de desempenho. Na mesma linha, Batista et al. (2004) mostra que métodos de *oversampling* no geral fornecem bons resultados na prática.

Por outro lado, Tarawneh et al. (2022), já no título do artigo, sugere que não se utilize *oversampling* para correção de desbalanceamento. Similarmente, van den Goorbergh et al. (2022) conclui que o uso de *random oversampling*, *random undersampling* e SMOTE, ocasionaram modelos mal calibrados, fortemente superestimando a probabilidade de uma dada instância pertencer a classe minoritária. De forma similar, Elkan (2001) no contexto de *cost-sensitive learning*, conclui que "[...] mudar a proporção de exemplos negativos e positivos no treinamento tem pouco efeito no classificador treinado".

Como na estatística não existe almoço grátis, é relevante entender melhor os cenários nos quais essas estratégias podem ser beneficiais, bem como suas limitações; faremos isso por meio da simulação de dados desbalanceados com propriedades conhecidas, avaliando o desempenho dos modelos com e sem reamostragem, por meio de métricas estabelecidas na literatura.

1.3 Objetivos

Este trabalho tem como principal objetivo investigar as vantagens e desvantagens das estratégias de reamostragem quando utilizadas como forma de correção de desbalanceamento entre classes no ajuste de modelos de classificação, expandindo o trabalho de van den Goorbergh et al. (2022). Além do modelo de regressão logística, consideraremos também um classificador SVM. Os métodos de reamostragem considerados serão *Random Oversampling*, *Random Undersampling*, SMOTE, *Borderline*-SMOTE1, ADASYN, *Edited Nearest Neighbours* (ENN), *Cluster Centroids*, e SMOTE utilizado conjuntamente com ENN. Já as métricas escolhidas para avaliação de performance foram F_1 score, que depende diretamente da precisão e do *recall*, área sob a curva ROC e o score de Brier, além de gráficos de calibração.

2 Fundamentação Teórica

2.1 Notação e Definições Básicas

A notação no geral seguirá o convencionado na literatura de estatística, denotaremos por letras maiúsculas as variáveis aleatórias (RVs) e observações destas RVs por letras minúsculas, por exemplo, a notação $\mathbb{P}(Y = y|\boldsymbol{X} = \boldsymbol{x})$ representa a probabilidade de Y assumir o valor y dado que $\boldsymbol{X} = \boldsymbol{x}$. Funções massa de probabilidade (PMF) e funções densidade de probabilidade (PDF) serão denotadas por $p(\boldsymbol{x})$, $p(\boldsymbol{x}, y)$ e de forma análoga para probabilidades condicionais. São sinônimos observações, instâncias e exemplos, quando se referem a realizações de uma variável aleatória.

Os modelos considerados no presente trabalho são denominados modelos de aprendizado supervisionado, mais especificamente, modelos de classificação, de modo que o objetivo principal é categorizar (classificar) uma **variável resposta** Y, baseando-se em observações desta e de um vetor $\boldsymbol{X} = (X_1, \ldots, X_K) \in \mathbb{R}^K$ de **variáveis preditoras** (também chamadas de variáveis explicativas, covariáveis ou atributos). No caso de aprendizado não supervisionado não há variável resposta diretamente observável (Hastie et al., 2009).

Iremos considerar somente o caso de resposta binária ou dicotômica, onde a variável resposta $y \in \{-1, +1\}$ ou $y \in \{0, 1\}$. A classe/categoria que ocorre com menor frequência será denominada **classe positiva** ou minoritária, e a que ocorre com maior frequência, **classe negativa** ou majoritária.

Um classificador genérico será denotado como uma função $h : \mathcal{X} \longrightarrow \mathcal{Y}$, tal que a classe atribuída a um padrão $\boldsymbol{x}$ é $h(\boldsymbol{x}) = y$, onde $h(\cdot)$ é a verdadeira função de classificação. Em alguns casos o classificador tem como *output* um *score* (possivelmente uma probabilidade), ou seja, $h : \mathcal{X} \longrightarrow \mathbb{R}$, de modo que $\boldsymbol{x} \mapsto h(\boldsymbol{x}) = p$. A partir do *score* e uma regra de decisão podemos obter uma classificação.

Na prática precisamos aproximar a função de classificação verdadeira a partir de um modelo proposto e dos dados observados, fazendo uma predição (classificação) a partir disso. Neste caso, usamos a notação de chapéu, $\hat{h}(\cdot)$, para indicar que a função foi aproximada a partir dos dados (os coeficientes foram estimados por máxima-verossimilhança, por exemplo), $\hat{p}$ para denotar uma probabilidade (ou *score*) estimada, e $\hat{y}$ para denotar a classe predita.

O vetor de covariáveis pode conter variáveis de natureza tanto contínua quanto discreta. Estaremos considerando, salvo menção ao contrário, observações independentes e identicamente distribuídas (IID) amostradas de algum processo gerador de dados (DGP), denotando-as por $(\boldsymbol{x}_i, y_i) \in \mathbb{R}^{K+1}$, com $i = 1, 2, \ldots, n$ indexando as

observações. As siglas que representam termos comuns estarão de acordo com a abreviação em inglês, por exemplo, DGP para *data generating process* e RV para *random variable*. Funções perda serão denotadas genericamente por $L(\cdot)$, já a função de verossimilhança será denotada por $\ell(\cdot)$. O logaritmo natural será denotado $\log(\cdot)$.

2.2 Dados Desbalanceados

Como mencionado anteriormente, o desbalanceamento ocorre quando uma das classes é sub-representada, não sendo observada com muita frequência. Na literatura não há uma proporção exata que caracterize a situação de desbalanceamento, porém, uma proporção de 1:9 ou inferior, significando que para cada observação da instância positiva temos 9 da instância negativa, é uma boa referência. Além disso, o problema desbalanceamento pode ser relativo ou absoluto (tamanho de amostra pequeno), embora ambos os casos possam gerar complicações, estaremos estudando o caso relativo.

Além do problema de o classificador trivial ter acurácia alta, frequentemente a classe de maior interesse prático é a minoritária, de modo que há uma preocupação em maximizar a taxa de verdadeiros positivos ou controlar os erros nessa classe. O problema então pode estar na métrica que usamos para avaliar os modelos, ou nas propriedades dos modelos de classificação, e não necessariamente nos dados; os métodos de reamostragem porém, atuam no nível dos dados (pré-processamento), pois alteram o conjunto de treinamento buscando um maior balanceamento entre as classes (He e Ma, 2013).

2.3 Modelos de Classificação

No contexto de aprendizado de máquina ou reconhecimento de padrões, modelos de classificação são modelos preditivos cuja variável resposta é categórica (qualitativa). O objetivo é construir **classificadores** com boas propriedades, e em especial, com capacidade de generalização; Fukunaga (2014) declara que, "De modo a construir classificadores, devemos estudar as características da distribuição de $\boldsymbol{X}$ em cada categoria e encontrar uma função de discriminação apropriada". Seguindo Sugiyama (2016), os dados são pares entrada-saída $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, usados para aproximar $h(\boldsymbol{x}) = y$, chamada função de classificação (ou de discriminação) verdadeira, ou seja, essa função leva um padrão $\boldsymbol{x}$ à sua classe y. Analogamente, o classificador estimado a partir dos dados é uma função $\hat{h} : \mathcal{X} \longrightarrow \mathcal{Y}$.

Existem diversas maneiras de se avaliar se um classificador tem "boas propriedades", um sumário muito comum é a matriz de confusão, que no caso binário é uma tabela de contingência 2 por 2, contendo a classe predita e a classe observada:

Tabela 2.1: Matriz de confusão no caso binário.

	Predito +1	**Predito -1**
Observado +1	TP	FN
Observado -1	FP	TN

Nas células temos os verdadeiros positivos (TP), verdadeiros negativos (TN), falsos positivos (FP) e falsos negativos (FN). A partir da matriz de confusão é possível construir diversas métricas como acurácia, taxa de verdadeiros positivos (sensibilidade/recall), taxa de verdadeiros negativos (especificidade) etc. Posteriormente iremos abordar essas métricas em mais detalhe.

A fase de treinamento do modelo usualmente envolve alguma forma de otimização, por exemplo a minimização do risco empírico de uma função de predição, sob alguma função perda. Além disso, e de forma similar, após o ajuste do modelo, busca-se calcular alguma medida de erro associada ao classificador. Nesse sentido, é muito importante que o conjunto de dados original seja particionado em, ao menos, duas partes: uma parcela usada apenas para treinar o classificador (conjunto de dados de treinamento), e outra parcela usada apenas para avaliar a performance desse classificador em dados "novos" (conjunto de dados de validação). Ainda sobre o particionamento dos dados, é essencial garantir que nenhuma informação do conjunto de treinamento vaze para o conjunto de validação ou teste, esse problema é chamado de *data leakage* (vazamento de dados) e acaba gerando estimativas otimistas demais.

2.3.1 Classificadores Probabilísticos

Os classificadores probabilísticos são regras (modelos, algoritmos) que classificam de acordo com a probabilidade estimada de cada classe, baseando-se em uma probabilidade à posteriori, tal como a regra do máximo à posteriori (MAP) dada por

$$\hat{y} = \underset{y}{\operatorname{argmax}}\, p(y|\boldsymbol{x})$$

Dizemos que um classificador probabilístico é do tipo **generativo** quando especifica-se um DGP para os dados, isto é, começamos propondo um modelo conjunto da forma $p(\boldsymbol{x}, y) = p(\boldsymbol{x}|y)p(y)$, estimando a probabilidade classe-condicional $p(\boldsymbol{x}|y)$ e a priori $p(y)$; o nome se deve ao fato de que a partir de $p(\boldsymbol{x}|y)$ podemos gerar observações de $\boldsymbol{x}$ dado que a classe é y. Já quando se estima diretamente a probabilidade classe-posteriori $p(y|\boldsymbol{x})$, o classificador é do tipo **discriminativo** (Sugiyama, 2016; Murphy, 2012).

2.3.2 Perda e Risco

Para se construir bons modelos preditivos é necessário um critério que quantifique o desempenho da função estimada. Isso pode ser feito por meio de uma função risco a ser otimizada, onde o problema pode ser de minimização ou maximização, a depender da função perda. No contexto de regressão linear, por exemplo, usualmente busca-se minimizar a perda quadrática esperada, onde os coeficientes do modelo são obtidos deste processo de minimização.

Seja $L(\hat{y}, y)$ a **perda** incorrida ao classificar um padrão $\boldsymbol{x}$ como $\hat{y}$ quando a classe verdadeira é y. A esperança matemática da função perda tomada com respeito à distribuição conjunta de Y e $\boldsymbol{X}$, dada por $\mathbb{E}[L(\hat{y}, y)]$, é o que se chama de **risco**. No geral, porém, não se sabe a distribuição conjunta dos dados, de modo que precisamos lidar com a versão empírica do risco. No contexto de classificação binária a perda zero-um é uma função naturalmente intuitiva, onde o risco estatístico de um classificador $h(\cdot)$ é portanto

$$R(h) := \mathbb{E}[\mathbb{I}(Y \neq h(\boldsymbol{X})] \tag{2.1}$$

sendo a função indicadora é definida por

$$\mathbb{I}(Y \neq h(\boldsymbol{X})) = \begin{cases} 1 & \text{se } y \neq h(\boldsymbol{x}), \\ 0 & \text{se } y = h(\boldsymbol{x}). \end{cases} \tag{2.2}$$

Neste caso, o risco é minimizado pelo classificador

$$h(\boldsymbol{x}) = \underset{y \in \mathcal{Y}}{\operatorname{argmax}}\, \mathbb{P}(Y = y|\boldsymbol{x}) \tag{2.3}$$

ou seja, o melhor classificador segundo o risco acima é aquele que classifica $\boldsymbol{x}$ como pertencente à classe com maior probabilidade à posteriori, chamado **classificador de Bayes** (Izbicki e dos Santos, 2020).

Na prática, limitamo-nos a considerar apenas funções de uma família $\mathcal{H}$, escolhendo o classificador $h : \mathcal{X} \longrightarrow \mathcal{Y}$ que **minimiza o risco empírico**, isto é, a perda média nos dados observados (conjunto de validação ou teste), matematicamente o risco empírico é dado por

$$\hat{R}(h) := \frac{1}{m} \sum_{i=1}^{m} L(h(\boldsymbol{x}_i), y_i) \tag{2.4}$$

Note que, como mencionado anteriormente, no caso de dados desbalanceados o risco de um classificador trivial sob a perda zero-um é baixo. Outras métricas para avaliar a performance do modelo, em especial na etapa de teste e validação, serão apresentadas posteriormente, porém o *framework* da minimização do risco empírico serve de base para falar sobre acurácia, matriz de confusão, validação cruzada, entre outras medidas.

2.3.3 Regressão Logística

Como exposto em Izbicki e dos Santos (2020), podemos estimar quantidades do tipo $P(Y = y|\boldsymbol{x})$ tomando a esperança de uma função indicadora, mais especificamente $\mathbb{E}[\mathbb{I}(Y = y)|\boldsymbol{x}]$. Portanto, podemos estimar probabilidades a partir de modelos de regressão. A conexão entre a esperança de variáveis indicadoras e probabilidades é chamada "ponte fundamental" por Blitzstein e Hwang (2019). A princípio poderíamos estimar uma probabilidade usando regressão linear, porém, possivelmente obteríamos valores fora do intervalo $[0, 1]$. O modelo de regressão logística não apresenta esse problema e pode ser usado para construir classificadores probabilísticos.

Seja a variável resposta $Y_i \sim \text{Bernoulli}(\pi_i)$, o modelo de regressão logística assume que o logaritmo da chance de um evento (função logito) é uma função linear das covariáveis, mais precisamente

$$\log\left(\frac{\pi_i}{1-\pi_i}\right) = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_{iK} x_{iK}$$

onde π_i é a probabilidade do evento para a observação i e K indica a quantidade de preditores no modelo. Podemos reescrever o modelo na seguinte forma paramétrica

$$P(Y_i = 1|\boldsymbol{x}_i) = \frac{\exp\{\beta_0 + \sum_{k=1}^{K} \beta_k x_{ik}\}}{1 + \exp\{\beta_0 + \sum_{k=1}^{K} \beta_k x_{ik}\}} \tag{2.5}$$

onde os parâmetros β podem ser estimados por máxima verossimilhança, por exemplo. Lembrando que nesse caso a função de verossimilhança a ser maximizada é dada por

$$\ell(y; \boldsymbol{x}, \boldsymbol{\beta}) = \prod_{i=1}^{n} \Big[P(Y_i = 1|\boldsymbol{x}_i, \boldsymbol{\beta})\Big]^{y_i} \Big[1 - P(Y_i = 1|\boldsymbol{x}_i, \boldsymbol{\beta})\Big]^{1-y_i}$$

vista como uma função de $\boldsymbol{\beta}$. A regressão logística nos dá um valor que pode ser interpretado como uma probabilidade, a partir do qual podemos criar uma regra de classificação do seguinte modo: se $P(Y = 1|\boldsymbol{x}) > 0.5$ então classifica $\boldsymbol{x}$ como 1, caso contrário como 0. A regra de classificação poderia considerar outros limiares, embora 0.5 seja o mais comum.

2.3.4 *Support Vector Machines* (SVMs)

Os modelos de SVM foram introduzidos por Cortes e Vapnik (1995), são muito populares e bastante estudados por serem intuitivos e apresentarem boas propriedades teóricas, com uma sólida teoria matemática por trás (Vapnik, 1999).

Um classificador baseado em SVMs usa diretamente a noção de hiperplano separador, ou seja, um plano no espaço p-dimensional dos dados e que separa as classes. Para melhor ilustrar essa noção, podemos limitar a exposição ao caso de um espaço bidimensional, onde o equivalente ao "plano" separador é uma reta.

Sejam X_1, X_2 duas covariáveis (atributos), de modo que uma instância (*data-point*) é caracterizada por um vetor/ponto num espaço bidimensional. Na figura 2.1, H_1, H_2 são retas que separam as classes (pontos em preto são de uma classe, em branco de outra), já H_3 não consegue separar os pontos adequadamente.

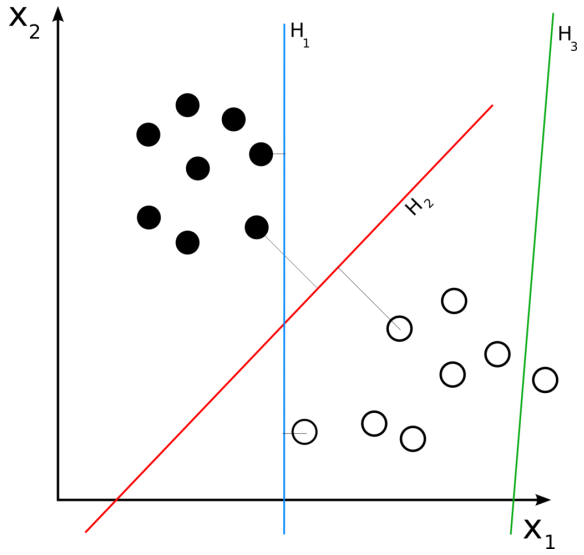

Figura 2.1: Conjunto de pontos num espaço bidimensional e diferentes retas candidatas a separadora das classes. (Cyc, Public domain, via Wikimedia Commons, 2008)

A reta H_1, embora separe corretamente os pontos de cada classe, tem uma **margem** pequena, já H_2 apresenta uma margem de separação maior, ilustrada pelas retas em cinza, perpendiculares à H_2 e em direção a um ponto de cada classe, esses

pontos no contexto de SVMs são os **vetores de suporte**, os pontos mais próximos do limiar de decisão mas com maior margem quando comparados a outros pontos. O nome "vetor de suporte" vem do fato de que uma pequena movimentação nesses pontos também moveria a reta de maior margem (James et al., 2013).

Possivelmente as nuvens de pontos não são linearmente separáveis, de modo que não existe um hiperplano que separa perfeitamente as classes, o *support vector classifier* permite que a construção do **hiperplano de margem máxima** seja tal que alguns dos pontos podem não ficar do lado correto, ou seja, permite algumas classificações erradas durante o treinamento, por isso é denominado um classificador de margem suave.

Uma das formas de definir o *Support Vector Classifier* (SVC) é a seguinte. Sejam $\lambda > 0$ um hiperparâmetro, $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_K$ covariáveis definidas em um espaço $\mathcal{X}$, $y_i \in \{-1, 1\}$ a variável resposta, onde $i = 1, \ldots, n$ indexa as observações. Então, o SVC é obtido por

$$h(\boldsymbol{X}) = \beta_0 + \beta_1 \boldsymbol{X}_1 + \beta_2 \boldsymbol{X}_2 + \cdots + \beta_K \boldsymbol{X}_K \tag{2.6}$$

$$\underset{\beta_0, \beta_1, \ldots, \beta_K}{\text{minimize}} \quad \sum_{i=1}^{n} \max\{0, 1 - y_i h(\boldsymbol{x}_i)\} + \lambda \sum_{j=1}^{K} \beta_j^2 \tag{2.7}$$

onde o problema de minimização, pela expressão 2.7, envolve a ***hinge loss*** dada por $L(y_i, h(\boldsymbol{x}_i)) = \max\{0, 1 - y_i h(\boldsymbol{x}_i)\}$ e uma penalização *ridge*, onde maiores valores de λ permitem que mais observações fiquem no lado errado da margem. Essa definição do SVC, embora não seja a mais comum, demonstra similaridades na forma funcional com o modelo de regressão logística.

Também é possível que os pontos sejam linearmente separáveis apenas em um espaço de dimensão maior, o chamado *kernel trick* que, de forma engenhosa, busca uma solução nesse sentido. Esse artifício gera um espaço de dimensão maior onde pode ser possível encontrar um hiperplano separador. Modelos de SVM são caracterizados por essa "expansão" do espaço das covariáveis, o que permite lidar com uma gama maior de problemas. A figura 2.2 ilustra isso, onde à esquerda temos a distribuição dos dados no espaço original das covariáveis, onde os dados não são linearmente separáveis, já à direita, no espaço de dimensão maior, os dados podem ser separados por um hiperplano. Outro ponto importante de ser mencionado é que, por construção, um classificador baseado em SVM não produz um *score* calibrado ou uma probabilidade, porém, utilizando o método chamado *Platt Scaling* (Platt, 1999) é possível obter *scores* calibrados a partir de um SVC. A ideia básica é ajustar um modelo logístico nos *scores*, ou seja, busca-se estimar

$$P(y = 1|\boldsymbol{x}) = \frac{1}{1 + \exp(\alpha + \beta h(\boldsymbol{x}))}$$

onde $h(\boldsymbol{x})$ é o *score*, e não a decisão $y = \text{sign}(h(\boldsymbol{x}))$ utilizada por padrão em um SVC; omitindo alguns detalhes sobre como transformar as respostas binárias em probabilidades e como estimar os parâmetros.

A teoria matemática por trás dos modelos de SVM é um pouco mais complexa do que o que foi exposto até o momento, o leitor interessado pode consultar Cortes e Vapnik (1995) e James et al. (2013). Neste trabalho, tomaremos este modelo e sua implementação computacional, bem como a regressão logística, como fornecidos.

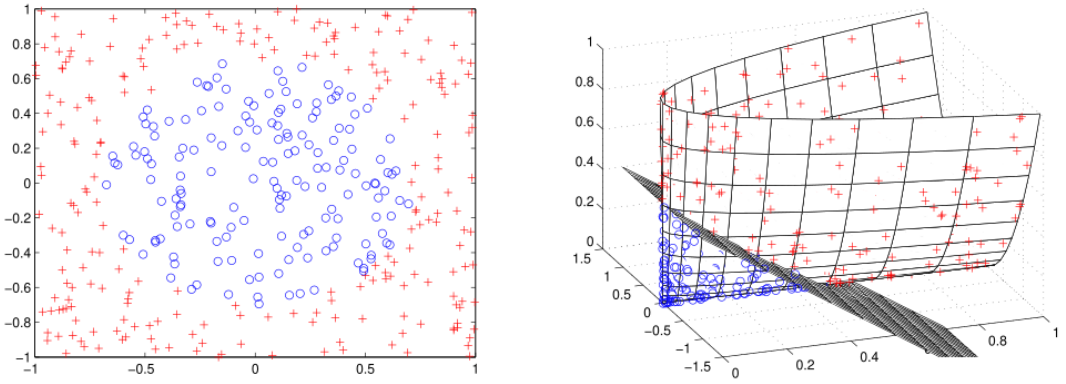

Figura 2.2: Exemplo de SVM com transformação de kernel em dados não linearmente separáveis no espaço original. (Machine Learner, CC BY-SA 4.0, via Wikimedia Commons, 2014)

2.4 Métodos de Reamostragem

Os métodos ou estratégias de reamostragem consistem na obtenção de novas observações à partir da amostra inicial e, quando usados no treinamento de modelos, são considerados uma solução *data-level* pois atuam na fase de pré-processamento dos dados, já as soluções a nível de algoritmo são aquelas que modificam diretamente o modelo ou algoritmo de aprendizagem.

As principais estratégias de reamostragem são a sobreamostragem e subamostragem. A primeira trata da coleta de observações selecionadas aleatoriamente com reposição da classe minoritária. Já a segunda envolve aleatoriamente descartar observações da classe majoritária. Além disso, é possível utilizar estratégias híbridas, que combinam os dois tipos. Vale ressaltar também que existem métodos de reamostragem, como a validação cruzada, que são utilizados para obter uma medida de erro dos modelos, ou na otimização de hiperparâmetros.

Para ilustrar melhor o conceito de *data-level approach*, em notação matemática, podemos denotar o conjunto de dados de treinamento original por $\mathcal{D}_{\text{train}}$, e para fins de exemplificação, dois outros *datasets* $\mathcal{D}_{\text{RUS}}$ e $\mathcal{D}_{\text{ROS}}$ obtidos do conjunto de dados original por *random undersampling* e *random oversampling*, respectivamente. Deste modo, para um mesmo modelo de classificação (regressão logística, por exemplo), teríamos 3 versões, $\mathcal{M}_{\text{train}}, \mathcal{M}_{\text{RUS}}, \mathcal{M}_{\text{ROS}}$, a serem comparadas em termos de performance preditiva.

Essas estratégias, embora populares, apresentam algumas limitações, por exemplo, *undersampling* efetivamente desperdiça informação ao descartar observações, e *oversampling* pode ocasionar *overfitting*, ou seja, o modelo se ajusta bem demais aos dados observados mas perde capacidade de generalização, pois uma mesma instância pode aparecer múltiplas vezes (He e Ma, 2013). Outra limitação das estratégias se dá na calibração dos modelos. Idealmente, para as previsões que um modelo estima uma probabilidade de ocorrência de, digamos, 30%, espera-se que cerca de 30% dessas previsões de fato ocorram; um modelo desses é dito **calibrado**. Iremos aprofundar melhor esse ponto na seção 2.5 e no capítulo 3.

2.4.1 Validação Cruzada

O método de validação cruzada (Stone, 1974), abreviado por CV, é ubíquo na prática e teoria, usado para obter estimativas de erro ou da habilidade de generalização de modelos preditivos, otimização de hiperparâmetros e seleção de modelos no geral. A ideia básica por trás do método é dividir os dados de treinamento em k parcelas (k-*fold* CV), uma delas utilizada para validação e o restante para treinar o modelo, fazendo isso iterativamente em cada uma das parcelas. A versão chamada *leave-one-out cross-validation* (LOOCV) envolve considerar um "lote" de uma única instância, realizando o mesmo procedimento. A figura 2.3 ilustra o método.

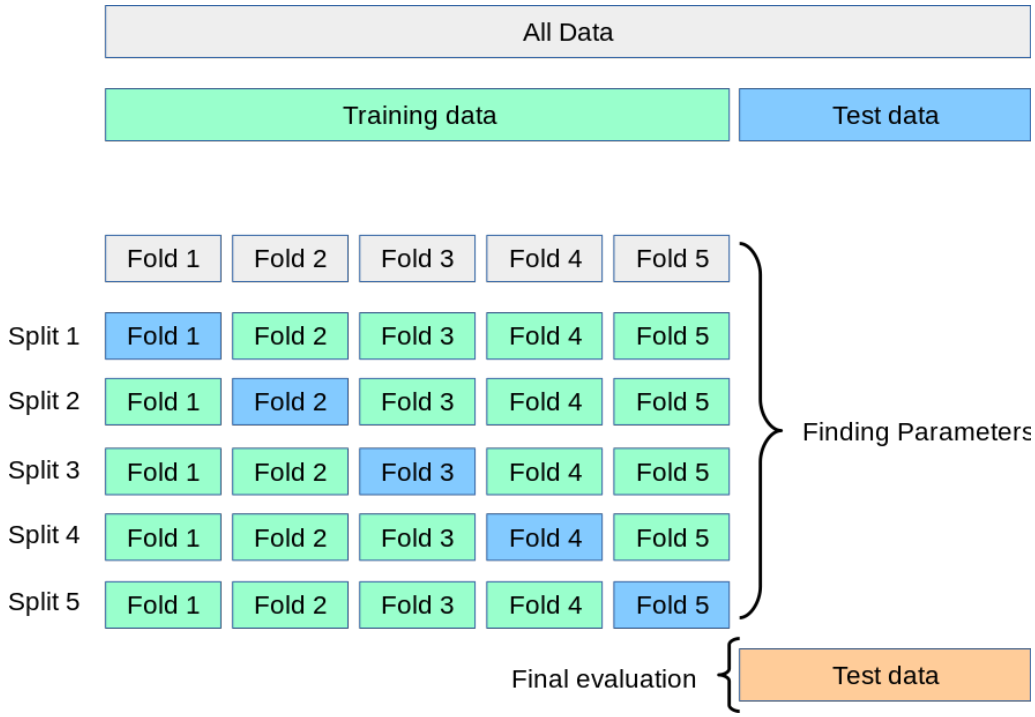

Figura 2.3: Ilustração do método de validação cruzada com 5 lotes.

Em cada *split* um dos lotes não é utilizado no treinamento mas sim na validação. No caso de 5 lotes, teremos também 5 *splits*. Veja que particionamos em lotes o conjunto "original" de dados de treinamento, de modo que o lote usado para validação também é parte do conjunto de dados de treinamento. A parcela "original" de dados de teste não é utilizada na validação cruzada, mas sim em uma avaliação final do modelo. O procedimento básico é, portanto:

- Treinamento do modelo em $k-1$ parcelas do conjunto de treinamento;
- Validaçao do modelo na parcela restante (por exemplo, cálculo da acurácia).

Não entramos em detalhes, mas o particionamento do conjunto de treinamento é feito de modo a obter lotes de tamanho aproximadamente iguais. Na prática é comum o particionamento em 5 ou 10 lotes (He e Ma, 2013). No contexto de dados desbalanceados, dependendo da prevalência da classe minoritária, é possível que a distribuição das classes em cada um dos lotes seja desigual; pode ser que alguns lotes contenham poucas ou nenhuma observação da classe rara. Uma mitigação

desse problema, especificamente para o caso de validação cruzada, é o uso de um esquema de amostragem estratificada, e não amostragem aleatória simples.

Em termos simples, a estratificação pela variável resposta (a classe) busca amostrar aleatoriamente elementos de **ambas** as classes, garantindo que os lotes tenham uma distribuição equilibrada. Um estudo conduzido por Kohavi et al. (1995) recomenda que se utilize validação cruzada 10-*fold* com estratificação, concluindo ser o melhor método para seleção de modelos.

2.4.2 *Oversampling* e *Undersampling*

O método de *Random Oversampling* (ROS) consiste em aumentar artificialmente a classe minoritária. Para implementar os processos de reamostragem basta considerarmos o conjunto $\mathcal{I} = \{1, 2, \ldots, n\}$ que indexa cada uma das observações no conjunto de treinamento, ou seja, o número 1 indexa uma unidade, o número 2 indexa outra, e assim por diante, não importando a ordem. O método de ROS consiste em obter repetidas amostras **com reposição** do conjunto $\mathcal{I}_-$ dos índices das observações que pertencem à classe minoritária, tendo cada índice a mesma probabilidade de ser selecionado. Por exemplo, considere o conjunto $\mathcal{I} = \{1, 2, \ldots, 10\}$ de 10 observações, das quais $\mathcal{I}_- = \{2, 7, 9\}$, indexam instâncias da classe minoritária, então, uma possível amostra (nesse caso de tamanho igual a seis) obtida por ROS é a lista $(2, 2, 7, 2, 9, 9)$. Veja que originalmente tínhamos apenas 3 observações da classe minoritária, e a nova amostra contém 6, por conta de termos reamostrado algumas observações.

De forma parecida, *Random Undersampling* (RUS) busca sub-representar a classe com mais observações de modo a obter um maior grau de balanceamento entre as classes. Seguindo a notação acima, teríamos um conjunto de índices $\mathcal{I}_+$ de onde removeríamos observações aleatoriamente até obter uma nova amostra do tamanho desejado. Por exemplo, para o mesmo conjunto de 10 observações acima, as instâncias da classe majoritária são $\mathcal{I}_+ = \{1, 3, 4, 5, 6, 8, 10\}$ de modo que uma possível subamostra é o conjunto $\{3, 4, 10\}$.

Note que no caso de ROS não criamos nenhuma informação que já não estava presente nos dados originais e no RUS descartamos informação ao ignorar parte da amostra, o que ilustra as limitações desses métodos. De qualquer maneira, as ideias de *oversampling* e *undersampling* servem de base para os outros métodos de reamostragem, podendo ser combinados e modificados para criar diferentes técnicas.

2.4.3 SMOTE

Este método foi introduzido por Chawla et al. (2002) e consiste em uma estratégia de sobreamostragem da classe minoritária que gera dados sintéticos, isto é, observações novas que não estão na amostra original. As novas observações são geradas levando em conta a vizinhança de cada ponto. Nas palavras dos autores: "A classe minoritária é sobre-amostrada tomando cada observação da classe minoritária e introduzindo observações sintéticas ao longo do segmento de reta que liga algum dos/todos os k vizinhos mais próximos [das observações] da classe minoritária." Portanto, o método se baseia na ideia de k-vizinhos mais próximos (kNN, *k-nearest neighbours*) (Stone, 1977), no qual a noção de proximidade entre dois pontos quaisquer é medida por uma função distância $d(\cdot, \cdot)$, usualmente a distância

euclidiana.

O passo a passo do algoritmo, baseando-se no pseudocódigo que os autores fornecem no artigo, é:

1. Especifica-se o percentual de reamostragem a ser aplicado e o número k de vizinhos a ser considerado;

2. Para cada instância $\boldsymbol{x}_i$ da classe minoritária encontram-se os k vizinhos mais próximos;

3. Sorteia-se com probabilidade uniforme discreta um dos k vizinhos mais próximos, denotamos esse vizinho por $\boldsymbol{x}_{zi}$;

4. Sorteia-se com probabilidade uniforme contínua um valor $\lambda \in [0, 1]$;

5. Gera-se uma instância sintética na reta que liga a instância ao vizinho sorteado, pela fórmula $\boldsymbol{x}_{\text{new}} = \boldsymbol{x}_i + \lambda(\boldsymbol{x}_{zi} - \boldsymbol{x}_i)$.

Fica implícito que os passos são repetidos para cada instância da classe minoritária, embora alguns *edge cases* não tenham sido contornados. Por exemplo, no passo (2), se o percentual de reamostragem a ser aplicado for menor que 100% significa que a iteração não pode passar por todas instâncias da classe minoritária, sendo necessário um critério (usualmente aleatorização) para escolher quais instâncias considerar. O leitor pode consultar o artigo original (Chawla et al., 2002) caso necessite do pseudocódigo completo. A lógica do algoritmo pode ser representada pela figura 2.4, retirada diretamente da documentação do pacote *imbalanced-learn*. Dentro do

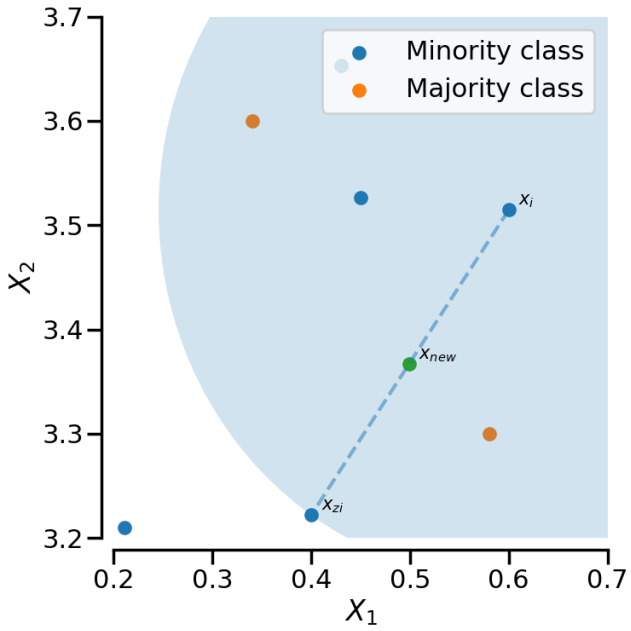

Figura 2.4: Exemplo do algoritmo SMOTE criando uma nova observação sintética. (Lemaître et al., 2017)

círculo azul temos os 3 vizinhos mais próximos da instância $\boldsymbol{x}_i$, o vizinho sorteado $\boldsymbol{x}_{zi}$ (que acabou sendo o mais distante dos 3), e a nova instância $\boldsymbol{x}_{\text{new}}$, gerada no

segmento de reta ligando a instância original e o vizinho sorteado. O algoritmo SMOTE, assim como *oversampling* e *undersampling*, serve de base para boa parte dos outros métodos que veremos. A maioria destes métodos prioriza instâncias de acordo com certos critérios, e após isso aplica SMOTE nas instâncias priorizadas.

2.4.4 *Borderline*-SMOTE1

Esse método é baseado no SMOTE e foi introduzido por Han et al. (2005). No artigo os autores introduzem duas versões ligeiramente diferentes, a versão *Borderline*-SMOTE2 está além do escopo deste trabalho. A ideia do método é identificar um limiar que separa as classes e reamostrar em maior quantidade as instâncias próximas ao limiar, pois supõe-se que sejam mais importantes de se aprender. O SMOTE não prioriza nenhuma instância. Este limiar é encontrado construindo um conjunto que os autores denominam DANGER do seguinte modo:

1. Para cada instância da classe minoritária encontra-se os k' vizinhos mais próximos;

2. Se todos os k' vizinhos de uma dada instância minoritária são da classe oposta então essa instância é considerada ruído e nada é feito;

3. Se a maioria (mas não todos) dos vizinhos k' de uma dada instância for da classe oposta então essa instância é considerada difícil de classificar e é posta no conjunto DANGER, caso contrário é considerada fácil e nada é feito;

4. As instâncias em DANGER formam o limiar mencionado. Para cada uma delas encontram-se os k vizinhos mais próximos e prossegue-se de acordo com o algoritmo SMOTE.

Veja que a diferença quando comparado ao SMOTE é que nesse caso primeiro temos um passo que considera k' vizinhos e busca obter as instâncias "difíceis", após isso o algoritmo é o mesmo.

2.4.5 ADASYN

O método de ADASYN significa "estratégia de amostragem *Adaptive Synthetic*". Foi introduzido por He et al. (2008) e é similar ao *Borderline*-SMOTE. A similaridade se dá em ser uma estratégia que gera observações sintéticas por interpolação, aumentando a classe minoritária, usando o método de k-vizinhos mais próximos, priorizando instâncias difíceis. Mais precisamente, o ADASYN gera observações baseadas em uma dada instância $\boldsymbol{x}_i$ proporcionalmente ao número de instâncias vizinhas que não são da mesma classe. Nas palavras dos autores, "[...] a ideia do ADASYN é usar uma distribuição ponderada para as diferentes observações da classe minoritária de acordo com seu nível de dificuldade na aprendizagem, onde mais dados sintéticos são gerados para observações da classe minoritária que são mais difíceis de aprender [...]" (He et al., 2008).

O algoritmo pode ser assim descrito:

1. Especifica-se o número total G de observações a ser gerado e o número k de vizinhos a ser considerado;

2. Para cada instância $\boldsymbol{x}_i$ da classe minoritária encontram-se os k vizinhos mais próximos e calcula-se $r_i = \Delta_i/k \in [0, 1]$ onde Δ_i é a quantidade de instâncias dentre os k vizinhos que pertence à classe majoritária;

3. Normaliza-se cada r_i de modo que a soma sobre todas instâncias da classe minoritária seja 1, ou seja, toma-se $\pi_i = r_i / \sum_i r_i$;

4. Calcula-se o número $g_i = \pi_i \times G$ de instâncias sintéticas a serem geradas a partir de cada $\boldsymbol{x}_i$, cada instância tem um respectivo g_i;

5. Tendo obtido os valores anteriores, para cada instância $\boldsymbol{x}_i$ da classe minoritária geram-se g_i novas instâncias tomando-se aleatoriamente um vizinho $\boldsymbol{x}_i$ dentre os k mais próximos, gerando as instâncias pela fórmula $\boldsymbol{x}_{\text{new}} = \boldsymbol{x}_i + \lambda(\boldsymbol{x}_{zi} - \boldsymbol{x}_i)$ do mesmo modo que se faz no SMOTE.

Veja que r_i (e consequentemente a "densidade" π_i) é uma medida do grau de dificuldade de aprender uma dada instância, sendo igual a zero se nenhum dos k vizinhos é da classe oposta, ou 1 se todos os vizinhos são. Baseado nisso, g_i será maior quanto maior for o número de vizinhos da classe oposta.

2.4.6 *Edited Nearest Neighbours (ENN)*

Este método baseia-se em Wilson (1972) e tem similaridades com o ADASYN. A ideia é "limpar" o conjunto de dados de treinamento removendo observações da classe negativa, mais precisamente, instâncias próximas ao limiar de decisão são removidas se a maioria dos vizinhos mais próximos for da classe oposta. Na verdade o método é mais flexível do que isso. É possível aplicar a qualquer instância, não só da classe minoritária. A proporção de vizinhos da classe oposta não necessariamente precisa constituir uma maioria, e até é possível aplicar em um problema multiclasse, porém, não iremos nos aprofundar no método. O passo a passo do ENN é:

1. Especifica-se o número k de vizinhos mais próximos. Usualmente 3 ou 5;

2. Especifica-se qual a classe a ser considerada (minoritária, majoritária ou ambas);

3. Para cada instância da classe especificada encontra-se seus k vizinhos mais próximos;

4. Compara-se a classe da instância com a classe predominante nos vizinhos mais próximos;

5. Se a maioria dos vizinhos for da classe oposta, então remove essa instância do *dataset*, caso contrário mantém.

Note que o algoritmo tende a preservar o limiar de decisão a menos que a instância esteja "mais do lado oposto" do que do lado correto, baseado na sua vizinhança. As instâncias "ruído", que claramente estão no meio de um conglomerado da classe oposta também são removidas. Uma estratégia mista que iremos utilizar é o SMOTE conjuntamente com o ENN, que denotaremos SMOTEENN.

2.4.7 *Cluster Centroids*

Este método diminui a classe majoritária substituindo conglomerados de observações pelo seu centroide de acordo com o algoritmo *k-means* (MacQueen et al., 1967). No *k-means*, os conglomerados são inicializados aleatoriamente, após isso, para cada ponto é verificado qual o *cluster* mais próximo, baseado no centroide atual daquele *cluster*, e então esse ponto é considerado pertencente ao *cluster*, o passo é repetido, recalculando os centroides (média do conglomerado de pontos) até que não haja melhoria na separação dos conglomerados. Não iremos abordar o algoritmo a fundo, basta saber que no contexto de reamostragem o método substitui um conjunto de pontos pelo seu centroide, dessa forma reduzindo a classe majoritária.

2.5 Métricas de Desempenho

Uma parte importante na construção de qualquer modelo estatístico é a avaliação da qualidade ou desempenho do modelo. Para tal fim, precisamos definir alguma métrica a ser otimizada. Essa escolha envolve uma certa subjetividade, além de depender do contexto do problema sendo estudado. Uma possível abordagem é considerar pares de métricas que levam em conta o contexto do problema e o risco de cada decisão, por exemplo, dois pares comumente utilizados são precisão e *recall*, ou sensibilidade e especificidade (Cullerne Bown, 2024).

No contexto de previsões probabilísticas, temos as *scoring rules*, como o *score* de Brier e a *log-loss*, que, de acordo com Gneiting e Raftery (2007), servem para avaliar a qualidade de previsões probabilísticas, fornecendo um valor numérico (um *score*) baseado na distribuição preditiva e no evento ou valor que se observa. Uma *proper scoring rule* é otimizada (maximizada ou minimizada, dependendo da definição) quando as probabilidades preditas coincidem com as probabilidades reais (Silva Filho et al., 2023).

Num ponto de vista mais geral, um conceito muito relevante na construção de modelos preditivos é o *tradeoff* viés-variância, relacionado ao que se chama de sobreajuste. Em termos simples, o sobreajuste ocorre quando um dado modelo se ajusta muito bem aos dados de treinamento mas mal às novas observações, isto é, não generaliza bem. De acordo com Kohavi et al. (1996), o custo esperado (sob perda quadrática) de um algoritmo de aprendizagem pode ser decomposto na soma de três quantidades não negativas:

- Ruído intrínseco à variável resposta.
- Viés (ao quadrado). Uma quantidade que mede o quão próxima a estimativa média (sob todos possíveis conjuntos de treinamento de um dado tamanho) de um algoritmo está do verdadeiro valor da variável resposta.
- Variância. Quantidade que mede o quanto as estimativas do algoritmo de aprendizagem variam/flutuam em diferentes conjuntos de treinamento de um dado tamanho.

No artigo, os autores introduzem uma decomposição similar para o caso de perda zero-um, a mais comum em classificação binária, porém nos basta entender os conceitos de sobreajuste e *tradeoff* viés-variância como ilustra a figura 2.5.

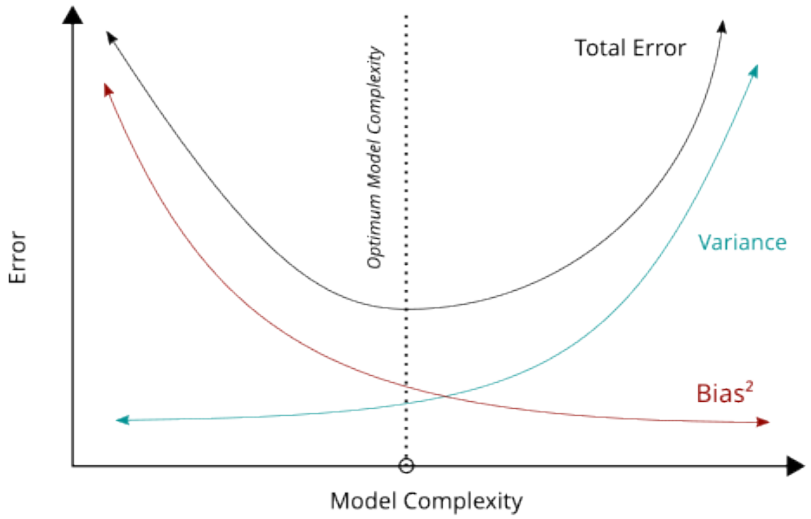

Figura 2.5: Ilustração do *tradeoff* entre viés e variância no erro total do modelo. (Bigbossfarin, CC0, via Wikimedia Commons, 2021)

Algumas métricas dependem diretamente do limiar de decisão, sendo exemplos a acurácia, precisão, *recall* e F-*score*. No geral, quanto menor o limiar maior a chance de classificar uma dada instância como positiva, aumentando também a taxa de falsos positivos. Como não existe um limiar "correto", pode ser vantajoso utilizar métricas que independem do limiar, como a curva ROC; ou otimizar o ponto de corte a ser utilizado. De qualquer modo, todas as métricas servem para distinguir a correta classificação entre diferentes classes (Sokolova et al., 2006).

2.5.1 Métricas Baseadas na Matriz de Confusão

A partir da matriz de confusão introduzida anteriormente, é possível construir diversas métricas. Sejam TP, TN, FP, FN os verdadeiros positivos, verdadeiros negativos, falsos positivos e falsos negativos, respectivamente, então:

1. A **acurácia** é definida como $\frac{\text{TP+TN}}{\text{TP+TN+FP+FN}}$
2. A **precisão** é $\frac{TP}{\text{TP+FP}}$
3. A taxa de verdadeiros positivos (**TPR**) ou ***recall*** é $\frac{\text{TP}}{\text{TP+FN}}$
4. A taxa de falsos positivos (**FPR**) é $\frac{\text{FP}}{\text{FP+TN}}$

Note que a precisão tem como denominador o total de observações que foram classificadas (preditas) como positivas, já o *recall* tem como denominador as observações que de fato são positivas. Similarmente, a FPR tem como denominador o total de instâncias que são de fato negativas.

2.5.2 F-*score*

Esta medida, também denotada F_β *score*, é uma média harmônica ponderada da precisão e *recall*, introduzidas na seção anterior, e é dada pela expressão

$$F_\beta = (1 + \beta^2)\frac{PR}{\beta^2 P + R} \tag{2.8}$$

onde P denota a precisão, e R o *recall* (Van Rijsbergen, 1979). O coeficiente β prioriza o *recall*, quanto maior for (Derczynski, 2016). Neste trabalho utilizaremos a versão chamada F_1 *score*, com $\beta = 1$, de modo a dar o mesmo peso à precisão e *recall*, definida como

$$F_1 = 2\frac{PR}{P+R} \tag{2.9}$$

Como regra de bolso, podemos priorizar a precisão quando o objetivo é evitar os falsos positivos, ou *recall* quando o foco é evitar falsos negativos. De modo geral, em dados desbalanceados, há uma preocupação maior em detectar os padrões que caracterizam a classe minoritária (detectar uma fraude, ou presença de doença), de modo que o *recall* tem relevância maior.

2.5.3 Curva ROC

A **curva característica de operação do receptor** representa graficamente as taxas de falsos positivos (FPR) no eixo x e as taxas de verdadeiros positivos (TPR) no eixo y ao longo de diferentes limiares de decisão, portanto, serve para comparar os benefícios e custos de um dado classificador (Fawcett, 2004). Matematicamente, seja $h(\cdot)$ um modelo de classificação que classifica de acordo com a seguinte regra

$$y_{\text{pred}} = \begin{cases} 1, & \text{se } h(\boldsymbol{x}) > \lambda, \\ 0, & \text{se } h(\boldsymbol{x}) \leq \lambda. \end{cases}$$

Então, a curva característica de operação é definida por $\lambda \mapsto (\text{FPR}(\lambda), \text{TPR}(\lambda))$, ao variarmos o limiar λ no intervalo $(-\infty, +\infty)$.

A tabela 2.2 ilustra um classificador inventado que, para um limiar de decisão de 0.8, tem TPR (recall) de 3/4, sendo $i = 1, 2, 7, 10$ as quatro instâncias da classe positiva, das quais as instâncias $i = 1, 2, 10$ foram corretamente classificadas; e que tem FPR igual a 1/6, pois dentre as 6 instâncias negativas, classificou apenas a instância $i = 9$ como positiva erroneamente.

Tabela 2.2: Exemplo fictício de *dataset* e *score* produzido por um classificador, junto da classe predita caso o limiar de decisão seja $\lambda = 0.8$.

i	**Classe**	***Score***	**Classificação**
1	1	0.90	1
2	1	0.83	1
3	0	0.50	0
4	0	0.42	0
5	0	0.74	0
6	0	0.17	0
7	1	0.66	0
8	0	0.70	0
9	0	0.81	1
10	1	0.93	1

Mesmo que um dado classificador não produza um *score*, apenas uma classificação discreta (uma decisão), é possível ao menos representar esse classificador por num ponto no ”espaço ROC”.

A métrica mais comum baseada na ROC é tomar a área sob a curva (chamada ROC AUC, AUROC, ou apenas AUC se o contexto deixar claro), pois sumariza o comportamento geral do modelo com um único número e tem boas propriedades. Essa métrica é considerada robusta na avaliação do desempenho de classificadores, já que independe do limiar de decisão e da prevalência *a priori* das classes (Bradley, 1997). Além disso, uma propriedade estatística relevante dessa métrica é que a AUC é equivalente à probabilidade de um classificador ordenar (de acordo com o *score*) uma instância aleatória da classe positiva acima de uma instância aleatória da classe negativa, ou seja, tem relação com o conceito de dominância/superioridade estocástica.

A figura 2.6 ilustra o gráfico das "curvas" de diferentes classificadores em dados simulados, apenas para fins ilustrativos. Para um numero finito de observações, a curva é na verdade uma função escada.

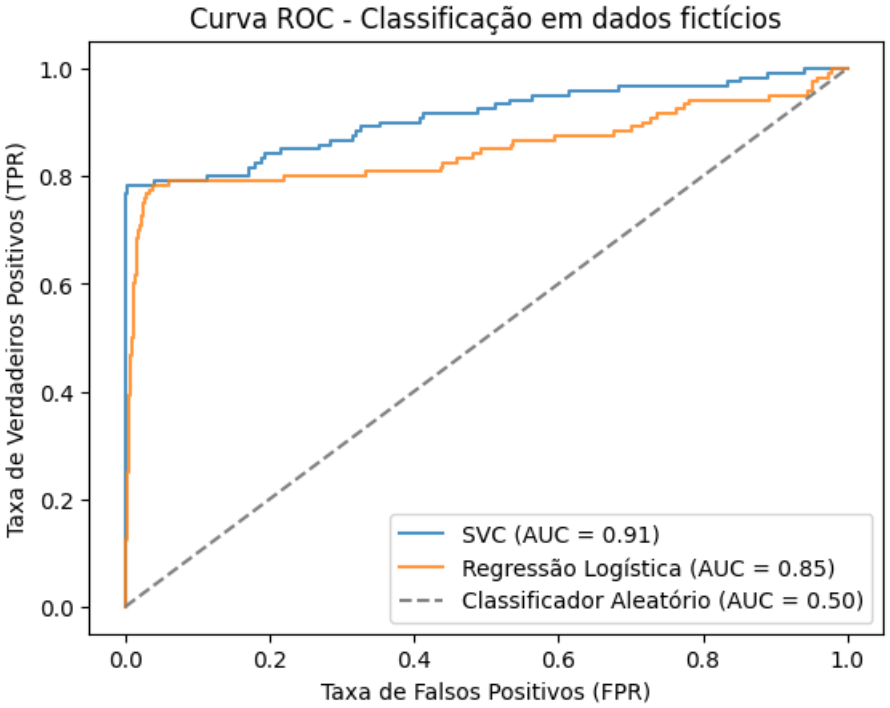

Figura 2.6: Exemplo de curvas ROC de diferentes modelos em dados simulados.

Um classificador considerado perfeito se situaria no ponto $(0, 1)$ do espaço FPR$\times$ TPR. O classificador aleatório é aquele que atribui uma instância à uma classe aleatoriamente, sem considerar a informação presente nas covariáveis. Note ainda que um classificador abaixo da diagonal $\text{TPR}(\lambda) = \text{FPR}(\lambda)$ pode ser considerado um classificador que tem informação útil mas a utiliza da forma errada, já que se tomássemos o oposto de suas classificações teríamos um valor acima da diagonal.

2.5.4 Calibração

Um *forecaster*, por exemplo um meteorologista, ou o próprio modelo de classificação, é dito **calibrado**, de acordo com Dawid (1982), se, por exemplo, dentre os eventos aos quais ele atribui a probabilidade 30%, a proporção de ocorrência [destes eventos] no longo prazo é 30%. Essa propriedade é relevante pois permite interpretar

scores como probabilidades. Porém, como diz (Spiegelhalter, 1986), calibração por si só não implica que as previsões sejam úteis, também é necessário que as previsões sejam correlacionadas com a variável resposta. Mais precisamente, um classificador probabilístico $h : \mathcal{X} \longrightarrow [0, 1]$ é dito calibrado se, para qualquer $p \in [0, 1]$, vale que

$$P(Y = 1 \mid h(\boldsymbol{X}) = p) = p \tag{2.10}$$

Em outras palavras, o vetor de probabilidades estimadas deve ser aproximadamente igual ao vetor de probabilidades empiricamente observadas (Silva Filho et al., 2023). O caso multiclasse envolve uma generalização não trivial da noção de calibração, na verdade nesse caso há mais de uma definição de calibração. Como informado anteriormente, neste trabalho limitamo-nos a classificadores binários.

Uma das vantagens de um classificador calibrado é permitir combinar e comparar medidas na mesma escala e definir regras de decisão padronizadas. Por exemplo, uma regra de decisão comum em classificação binária é fazer a previsão $y = 1$ quando $p > 0.5$. Se essa probabilidade estimada p não for calibrada, fica difícil justificar a regra. A noção de calibração também nos permite dizer se um dado classificador é *overconfident* ou *underconfident*, isto é, se um classificador "pensa" ser melhor ou pior do que realmente é, respectivamente, onde o que o classificador "pensa" se reflete no vetor de probabilidades estimadas (Silva Filho et al., 2023).

2.5.5 *Score* de Brier

O *score* de Brier é uma métrica que avalia tanto a discriminação quanto a calibração do modelo, e nada mais é do que um erro quadrático médio, portanto no nosso contexto assume a seguinte forma

$$B(\boldsymbol{y}, \hat{\boldsymbol{p}}) = n^{-1} \sum_{i=1}^{n} (y_i - \hat{p}_i)^2 \tag{2.11}$$

onde $\boldsymbol{y}$ é um vetor de realizações de variáveis aleatórias $Y_i \sim Ber(\pi_i)$, e $\hat{\boldsymbol{p}}$ um vetor de probabilidades preditas. Podemos decompor essa quantidade em um termo de "calibração" e um termo de "discriminação" dado por

$$B(\boldsymbol{y}, \hat{\boldsymbol{p}}) = n^{-1} \sum_{i=1}^{n} (y_i - \hat{p}_i)(1 - 2\hat{p}_i) + n^{-1} \sum_{i=1}^{n} \hat{p}_i(1 - \hat{p}_i) \tag{2.12}$$

onde o primeiro termo mede falta de calibração, com média zero sob a hipótese nula de calibração perfeita. O segundo termo mede falta de dispersão nas previsões (Spiegelhalter, 1986; Steyerberg et al., 2010).

Essa métrica pode ser usada para comparar a performance relativa de diferentes modelos de classificação, porém, um *score* menor não necessariamente implica um modelo melhor calibrado (Rufibach, 2010). Um teste estatístico para averiguar a calibração de diferentes modelos, baseado no *score* de Brier, é a estatística z de Spiegelhalter, introduzida no mesmo artigo do autor citado no parágrafo anterior, dada por

$$Z(\boldsymbol{y}, \hat{\boldsymbol{p}}) = \frac{\sum_{i=1}^{n} (y_i - \hat{p}_i)(1 - 2\hat{p}_i)}{\sqrt{\sum_{i=1}^{n} (1 - 2\hat{p}_i)^2 \hat{p}_i(1 - \hat{p}_i)}} \tag{2.13}$$

cuja hipótese nula de calibração, referente a $Y_i \sim Ber(\pi_i)$, de que $\hat{p}_i = \pi_i$, é rejeitada ao nível α se $|Z(\boldsymbol{y}, \hat{\boldsymbol{p}})| > q_{1-\alpha/2}$, sendo $q_{1-\alpha/2}$ o quantil α de uma distribuição normal padrão.

3 Resultados

3.1 Introdução

Os dados foram gerados a partir da função `make_classification` do pacote *scikit-learn*, cujo algoritmo é baseado em Guyon (2003). De acordo com a documentação, os dados são gerados criando conglomerados de pontos a partir da distribuição normal padrão nos vértices de um hipercubo num espaço de dimensão a ser especificado pelo usuário, introduzindo correlação por meio de combinações lineares das covariáveis.

Os métodos de reamostragem foram implementados de acordo com as funcionalidades prontas da biblioteca *imbalanced-learn*. Essa biblioteca foi construída de modo a ser totalmente compatível com *scikit-learn*. O código utilizado para ajustar cada um dos modelos (considerando a reamostragem) está no apêndice, porém vale explicitar no texto o código utilizado para gerar os dados:

```
X, y = make_classification(n_samples=10000,
                           n_features=4,
                           n_informative=4,
                           n_redundant=0,
                           n_repeated=0,
                           weights=(0.97, 0.03),
                           flip_y=0.02,
                           class_sep=1.5,
                           scale=10,
                           random_state=SEED)
```

Onde os principais parâmetros são a semente `SEED = 42`, as 10 mil observações solicitadas, 4 covariáveis, 3% de prevalência da classe minoritária, e um ruído (inversão da classe) de 2%, portanto obtivemos 414 instâncias da classe minoritária, e não os 300 equivalente a 3%. A vantagem de simular os dados de maneira mais controlada é, além de poder compreender o processo gerador, evitar despender tempo na parte de *feature engineering* e pré-processamento dos dados (a não ser, é claro, no que diz respeito à reamostragem). Além disso, utilizando apenas 4 covariáveis, das quais todas são relevantes, simplifica-se a parte de visualização dos dados.

A tabela 3.1 contém algumas estatísticas descritivas dos dados gerados. No nosso caso, as particularidades dos dados não são tão relevantes quanto seriam em dados reais, exceto no que diz respeito ao desbalanceamento e separabilidade das classes.

Tabela 3.1: Sumário descritivo dos dados simulados pela função `make_classification` de acordo com os parâmetros fornecidos.

	X1	**X2**	**X3**	**X4**	**y**
Média	-0.011	0.043	-14.431	-14.645	0.041
Desvio	18.942	19.589	12.190	10.781	0.185
Mediana	-1.261	2.510	-14.560	-15.055	0
Min	-55.161	-70.495	-58.599	-56.023	0
Max	58.085	58.561	41.883	52.030	1

Para fins de treinamento e teste dos modelos foi escolhido um *split* 70-30, sendo 70% dos dados usado para treinamento e 30% para teste. Isso pode ser feito usando a função `train_test_split` do mesmo pacote, usando a mesma semente e estratificando pela variável resposta. Por conta da estratificação, tanto o conjunto de treinamento quanto teste também obtiveram uma prevalência da classe minoritária muito próxima de 4.14%. Ressaltamos aqui a importância desse particionamento dos dados, e da necessidade de evitar *data leakage*. Os métodos de reamostragem, por exemplo, só são aplicados no conjunto de treinamento.

A figura 3.1 traz os gráficos de dispersão dos possíveis pares dentre as 4 variáveis preditoras, em alguns casos as classes estão razoavelmente bem separadas, em outros não. Lembrando que o espaço $\mathcal{X}$ das covariáveis nesse caso tem dimensão 4, então é possível que o "grau de separabilidade" seja diferente do que se vê nos gráficos bidimensionais.

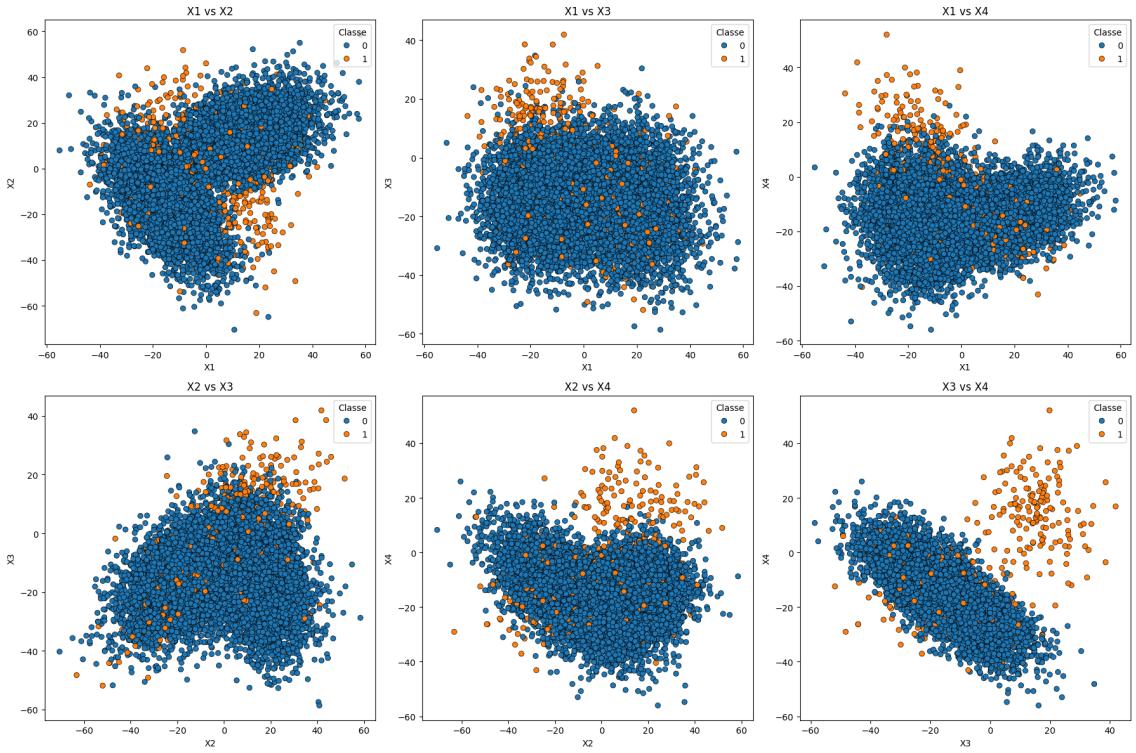

Figura 3.1: Plots dos possíveis pares dentre as 4 covariáveis geradas pela função `make_classification` de acordo com os parâmetros fornecidos.

3.2 Impacto da Reamostragem na Distribuição das Classes

Cada método impacta de forma diferente no espaço das covariáveis, e portanto na região de decisão. Tomemos como exemplo as covariáveis X_3 e X_4, aplicando a reamostragem no conjunto de treinamento, que conta com 7000 observações, das quais 290 são da classe positiva. Podemos visualizar melhor o comportamento de cada método pela figura 3.2. Note que alguns métodos claramente capturam muito do ruído, gerando conglomerados implausíveis sob o ponto de vista de um processo gerador de dados.

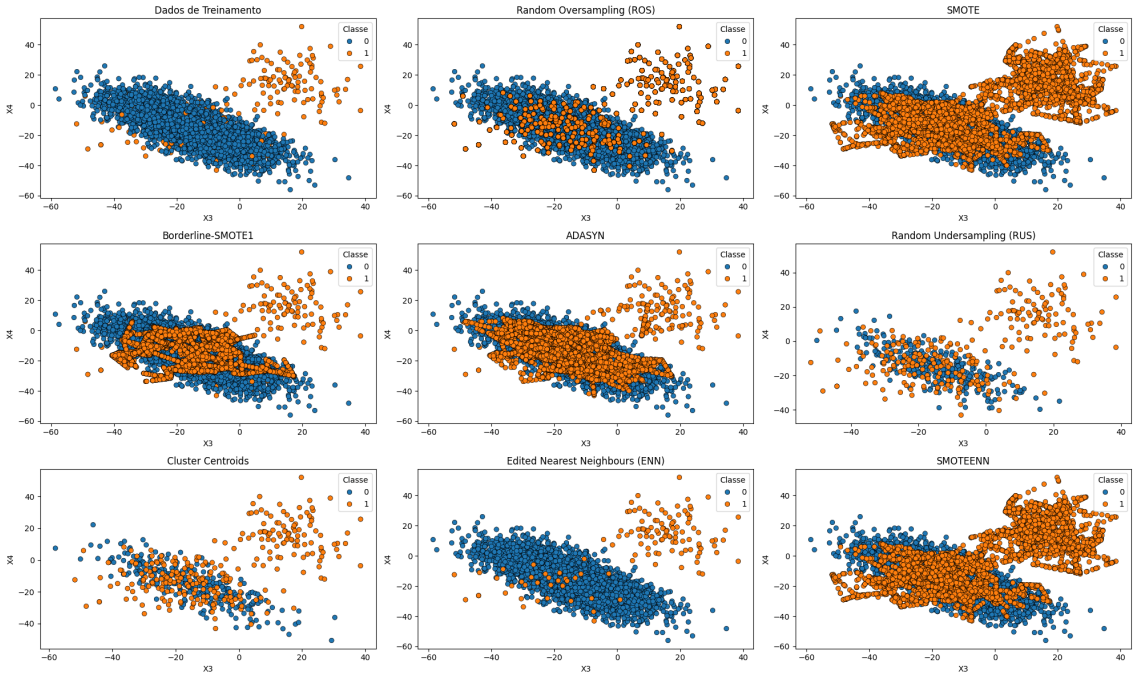

Figura 3.2: Impacto da reamostragem na distribuição dos dados, usando as covariáveis X_3 e X_4 como exemplo. De cima para baixo, da esquerda para a direita, temos: Os dados originais, ROS, SMOTE, *Borderline*-SMOTE1, ADASYN, RUS, *Cluster Centroids*, ENN, e SMOTEENN.

A tabela 3.2 sumariza a distribuição das classes após cada método de reamostragem ser aplicado. Note que nem todos os métodos buscam a proporção 1:1. O método ENN por exemplo, talvez contra-intuitivamente, aumenta o desbalanceamento. Porém, como o algoritmo foca em remover as instâncias difíceis (ruído, ou no limiar de decisão) é possível que um maior desbalanceamento melhore a separação entre as classes. No nosso caso, devido ao ruído de 2%, várias observações da classe positiva tiveram a classe trocada, portanto ficaram na nuvem de pontos da classe negativa, sendo removidos pelo ENN e métodos similares, que consideram a vizinhança de cada ponto para determinar se este ponto deve ser removido.

Tabela 3.2: Distribuição das classes sob cada método de reamostragem no conjunto de treinamento.

Reamostragem	**#Negativa**	**#Positiva**	**Proporção**
Nenhuma	6710	290	4.14%
ROS	6710	6710	50%
SMOTE	6710	6710	50%
Borderline-SMOTE1	6710	6710	50%
ADASYN	6710	6768	50.22%
RUS	290	290	50%
Cluster Centroids	290	290	50%
ENN	6371	135	2.08%
SMOTEENN	5516	6236	46.94%

3.3 Impacto da Reamostragem nos Modelos de Classificação

Para comparar o impacto da reamostragem no treinamento de modelos de classificação, iremos considerar a regressão logística e o SVC, treinados com os dados originais após a separação em treino-teste. Os parâmetros escolhidos podem ser consultados no apêndice e correspondem aos padrões dos respectivos modelos na implementação do *scikit-learn*. Diferentemente de van den Goorbergh et al. (2022) e Elor e Averbuch-Elor (2022), não iremos investigar diferentes graus de desbalanceamento, diferentes maneiras de otimizar os modelos nem qualquer outro procedimento referente a *model tuning*.

No caso da regressão logística, utilizar o limiar padrão de 0.5 afeta a calibração inclusive do caso sem reamostragem, principalmente por conta da prevalência a priori das classes. Inicialmente tomando como exemplo o modelo de regressão logística e as covariáveis X_3 e X_4 podemos vizualizar o impacto da reamostragem no limiar de decisão por meio da figura 3.3. No que todos os métodos movem o limiar de decisão de modo a englobar mais pontos da classe positiva, mesmo considerando apenas gráficos bidimensionais.

Um comportamento similar é observado sob outros método de reamostragem e outros modelos de classificação, incluindo o SVC. Essa mudança do limiar, decorrente da reamostragem, também ocorre quando o limiar não é um hiperplano. Ou seja, o impacto da reamostragem é similar a uma mudança no limiar de decisão, conclusão similar ao que mostra Assunção et al. (2023).

3.4 Desempenho no Treinamento

Podemos comparar o impacto da reamostragem no modelo de regressão logística, de acordo com as métricas de interesse por meio da tabela 3.3. As estimativas, incluindo o erro padrão (em parênteses), foram obtidas usando validação cruzada 5-*fold* com estratificação pela variável resposta. Vale ressaltar mais uma vez que no processo de validação cruzada é importante que a reamostragem seja aplicada somente após o particionamento dos dados de treinamento em lotes, caso contrário

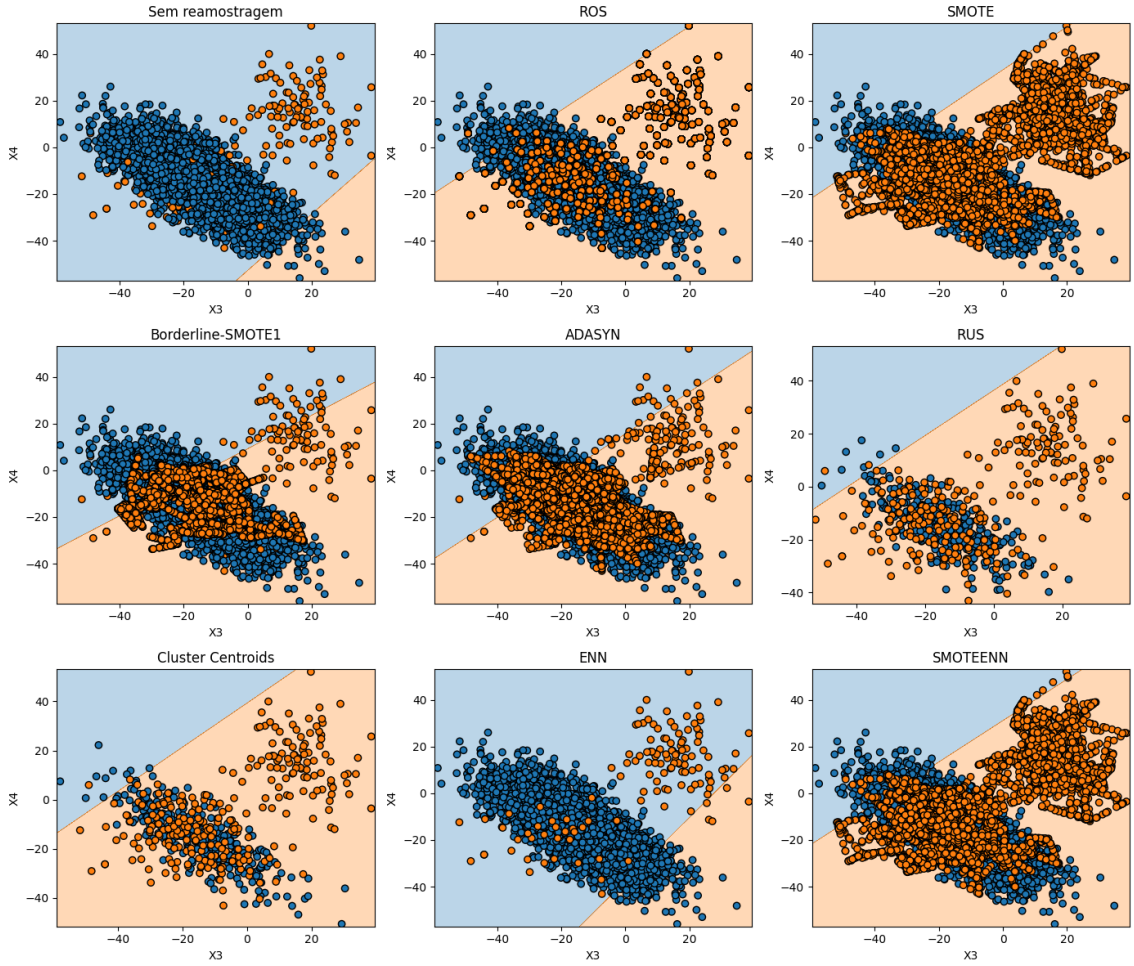

Figura 3.3: Impacto da reamostragem na região de decisão do modelo logístico considerando as variáveis X_3 e X_4.

há vazamento de informação de um lote para outro e os modelos ficariam sobreajustados. A escolha da validação cruzada 5-*fold* se deu por três motivos principais. Testamos a versão 10-*fold*, sugerida por Kohavi et al. (1995), e não encontramos valores muito distintos da versão 5-*fold* nos nossos dados. Além disso, o custo computacional é menor e a versão com 5 lotes é extremamente comum na literatura, sendo o padrão da biblioteca *scikit-learn*.

No geral, no classificador baseado em regressão logística, o uso da reamostragem impactou de negativamente o *score* F_1, exceto no caso do ENN, que teve o mesmo desempenho em relação à *baseline* (sem reamostragem). A métrica AUC permaneceu igual ou melhorou de forma marginal. O maior aumento foi de 0.085 na estimativa pontual, obtida pelo método SMOTEENN. O *score* de Brier, por sua vez, piorou em quase todos os métodos, exceto ENN. O método ADASYN, no geral, teve o pior desempenho. Note que o *score* de Brier, por ser uma perda (quadrádica), quanto maior pior. Portanto há uma tendência de piora na calibração dos modelos quanto se utiliza reamostragem.

De forma análoga, considerando agora o SVC, temos na tabela 3.4 o sumário do desempenho dos métodos de reamostragem. Nota-se que o SVC já parte de uma *baseline* ligeiramente melhor em todas as métricas, quando comparado ao modelo logístico. O comportamento geral dos métodos aparentemente é similar, porém nota-

Tabela 3.3: Comparação de desempenho do modelo logístico sob diferentes estratégias de reamostragem. Médias e erros padrão (em parênteses) obtidos por meio de validação cruzada 5-*fold* estratificada.

	Regressão Logística		
Estratégia	F1 *Score*	AUC	Brier
Sem reamostragem	0.437 (0.037)	0.641 (0.014)	0.028 (0.001)
ROS	0.175 (0.007)	0.704 (0.014)	0.177 (0.001)
SMOTE	0.180 (0.007)	0.712 (0.013)	0.176 (0.002)
Borderline-SMOTE1	0.168 (0.016)	0.686 (0.031)	0.175 (0.004)
ADASYN	0.127 (0.009)	0.645 (0.026)	0.231 (0.002)
RUS	0.178 (0.009)	0.700 (0.014)	0.175 (0.002)
Cluster Centroids	0.173 (0.010)	0.687 (0.017)	0.174 (0.0009)
ENN	0.437 (0.037)	0.641 (0.014)	0.029 (0.001)
SMOTEENN	0.177 (0.006)	0.726 (0.013)	0.186 (0.002)

se que o impacto da reamostragem não prejudicou severamente o F_1 *score*, nem as outras métricas. A variação na AUC continuou nula ou pequena. O *score* de Brier piorou, principalmente no ADASYN.

Mencionamos brevemente no trabalho, no contexto de SVMs, que é possível transformar *scores* em valores calibrados. Isso é um fato mais geral, não sendo limitado ao contexto de SVMs onde chamamos de Platt Scaling. O pacote *scikit-learn* utiliza dessa calibração quando solicitamos que o *ouput* do SVC seja uma probabilidade. Além disso, também por padrão, o SVC utiliza um *radial basis function kernel*, permitindo obter regiões de decisão não lineares.

Tabela 3.4: Comparação de desempenho do SVC sob diferentes estratégias de reamostragem. Médias e erros padrão (em parênteses) obtidos por meio de validação cruzada 5-*fold* estratificada.

	Support Vector Classifier		
Estratégia	F1 *Score*	AUC	Brier
Sem reamostragem	0.599 (0.043)	0.717 (0.021)	0.020 (0.001)
ROS	0.530 (0.014)	0.813 (0.009)	0.088 (0.0006)
SMOTE	0.527 (0.022)	0.809 (0.010)	0.088 (0.0008)
Borderline-SMOTE1	0.416 (0.018)	0.796 (0.009)	0.070 (0.003)
ADASYN	0.227 (0.009)	0.759 (0.005)	0.171 (0.003)
RUS	0.484 (0.026)	0.810 (0.009)	0.108 (0.003)
Cluster Centroids	0.492 (0.015)	0.809 (0.010)	0.106 (0.002)
ENN	0.610 (0.036)	0.722 (0.018)	0.020 (0.001)
SMOTEENN	0.430 (0.027)	0.801 (0.010)	0.100 (0.003)

3.5 Desempenho nos Dados de Teste

Na seção anterior obtivemos estimativas da média das métricas de interesse e seu erro padrão para cada um dos métodos de reamostragem em ambos os modelos

considerados nos dados de treinamento. Porém, parte crucial no desenvolvimento de modelos preditivos é avaliar os modelos ajustados em dados novos, desse modo é possível quantificar o poder preditivo do modelo e não sua capacidade de "memorização". Para avaliarmos os modelos em dados "novos", a tabela 3.5 traz um sumário dos resultados obtidos no *split* de teste. Optamos por trazer também a precisão e *recall*, medidas que compõem o F_1 *score*, para ilustrar melhor as nuances nos modelos e nas métricas.

Tabela 3.5: Desempenho dos modelos e estratégias de reamostragem nos dados de teste.

Modelo	**Estratégia**	**Métricas**				
		F_1 *Score*	AUC	Brier	Precisão	*Recall*
Regressão Logística	Sem reamostragem	0.469	0.814	0.027	1.000	0.306
	ROS	0.213	0.857	0.176	0.122	0.831
	SMOTE	0.212	0.857	0.176	0.121	0.831
	Borderline-SMOTE1	0.180	0.776	0.176	0.104	0.669
	ADASYN	0.139	0.744	0.235	0.077	0.750
	RUS	0.220	0.854	0.174	0.127	0.815
	Cluster Centroids	0.214	0.854	0.175	0.124	0.790
	ENN	0.469	0.817	0.027	1.000	0.306
	SMOTEENN	0.204	0.856	0.186	0.116	0.847
SVC	Sem reamostragem	0.649	0.861	0.018	0.984	0.484
	ROS	0.595	0.893	0.083	0.480	0.782
	SMOTE	0.615	0.886	0.083	0.511	0.774
	Borderline-SMOTE1	0.454	0.889	0.065	0.320	0.782
	ADASYN	0.265	0.881	0.173	0.159	0.815
	RUS	0.574	0.893	0.097	0.453	0.782
	Cluster Centroids	0.574	0.900	0.102	0.448	0.798
	ENN	0.656	0.861	0.017	0.984	0.492
	SMOTEENN	0.544	0.886	0.089	0.419	0.774

No geral, podemos notar um comportamento similar às estimativas obtidas nos dados de treinamento. Especialmente o *score* de Brier, que tanto calibração quanto discriminação, ficou muito similar. Baseado nessa métrica, há uma piora na calibração dos modelo sob reamostragem. Já a área sob a curva ROC, em todas as estratégias, partiu de uma *baseline* melhor (do que no treinamento). A regressão logística sem reamostragem, por exemplo, atingiu uma AUC de 0.861, o que é considerado bom. Para esse mesmo modelo, o método ADASYN foi o pior, apresentando uma AUC de 0.744. Veja que nos dados de teste, mesmo o ADASYN tendo apresentado a menor AUC, ainda é superior à todas estimativas de AUC nos dados de treinamento (para o modelo logístico).

Considerando o SVC, notamos que os F_1 *scores* foram quase sempre superiores aos do modelo logístico, embora os métodos de reamostragem não tenham melhorado essa métrica comparada à *baseline* do próprio SVC. A métrica AUC ficou consistentemente acima de 0.88 e o *score* de Brier abaixo de 0.10.

As métricas de precisão e *recall* permitem entender melhor as particularidades de cada modelo. Podemos notar que o SVC e a regressão logística tem *recall* similar

porém, o modelo logístico tem precisão muito menor, exceto no caso sem reamostragem e ENN. Sob estas estratégias, ambos tiveram precisão bem alta, em especial o classificador logístico que, dentre as previsões positivas, acertou todas. Em ambos os modelos, a maioria das estratégias prejudica a precisão mas melhora o *recall*. Isso pode ser útil na prática, em situações de dados desbalanceados, como mencionamos na seção 2.5. Em contraste à regressão logística, o SVC apresenta um balanceamento maior entre as duas métricas, com precisão sempre superior, independente da estratégia.

3.6 Calibração nos Dados de Teste

Podemos entender melhor o impacto da reamostragem na calibração dos modelos por meio dos diagramas de confiabilidade (ou curva de calibração). No eixo x temos a probabilidade média estimada em cada *bin*, onde o número de *bins* escolhido foi 5. Já no eixo y, temos a frequência observada de instâncias da classe positiva. Quanto à escolha do número de *bins*, do mesmo modo que em histogramas, no geral a escolha é subjetiva. No nosso caso, escolhemos 5 para não degenerar o gráfico por falta de instâncias positivas em alguma faixa, e para não capturar muito ruído. Podemos visualizar o comportamento do modelo logístico sob as diferente estratégias pela figura 3.4.

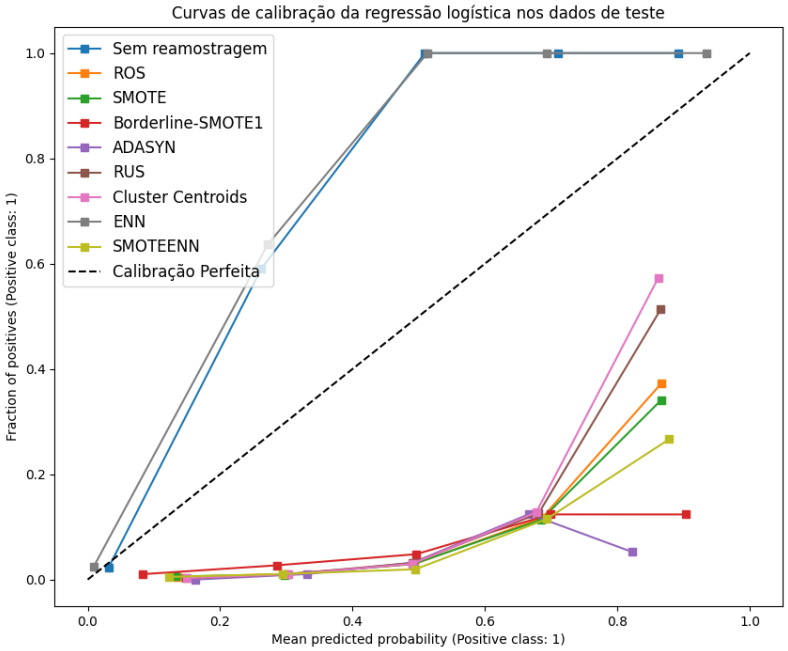

Figura 3.4: Curvas de calibração do modelo logístico sob as diferentes estratégias de reamostragem nos dados de teste.

Podemos notar que quase todos os métodos tendem a produzir probabilidades excessivamente altas, com um comportamento similar em quase todas as faixas. Nesse caso diríamos que o classificador é *overconfident*. Por outro lado, no caso sem reamostragem e ENN a tendência é de se subestimar as probabilidades. Tomando como exemplo o intervalo de 0.8 a 1.0. O método *Cluster Centroids* (em rosa) gerou estimativas com média próxima de 0.85 (85%), quando na verdade menos de 60%

das instâncias nessa faixa eram da classe positiva. O ADASYN (em lilás), ainda pior, estimou uma média também próxima a 0.8, sendo que a verdadeira frequência de instâncias da classe positiva foi praticamente 0%.

O mesmo comportamento geral pode ser observado no modelo SVC, como mostra a figura 3.5. Neste modelo, nas faixa de predições entre 0.8 e 1.0, o modelo tem uma melhora na calibração, embora ainda *overconfident*. Nas outras faixas a calibração é péssima. O caso sem reamostragem e o ENN, similar ao modelo logístico, apresentaram estimativas *underconfident*. Além disso, há um comportamento errático na faixa de 0.6 a 0.8.

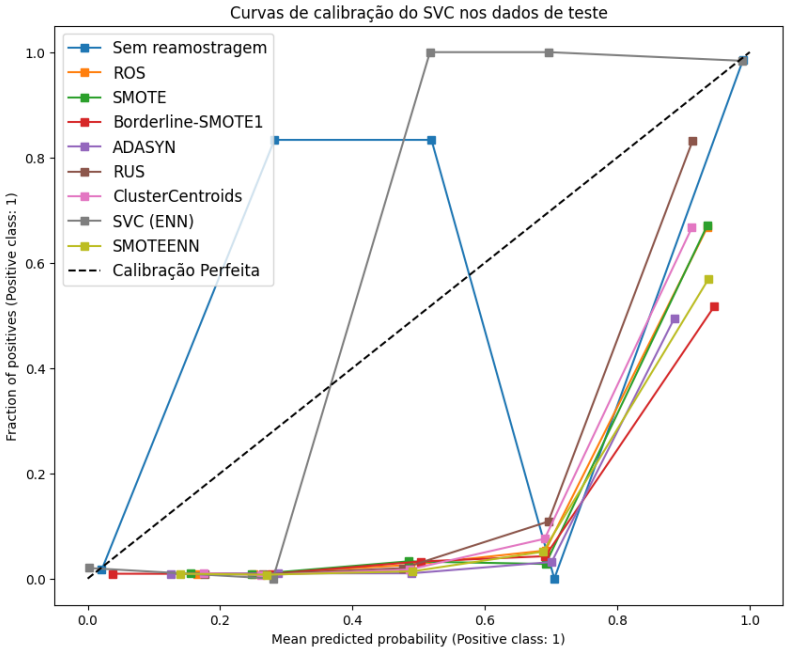

Figura 3.5: Curvas de calibração do modelo SVC sob as diferentes estratégias de reamostragem nos dados de teste.

Em linhas gerais, obtivemos resultados similares aos de van den Goorbergh et al. (2022), em especial no que diz respeito à calibração. Por outro lado, algumas métricas apresentaram melhoria quando utilizamos reamostragem no treinamento dos modelos, especialmente o *recall*. Como a precisão ficou baixa no geral, o F_1 *score* não apresentou valores bons em nenhum caso.

4 Considerações Finais

Este trabalho buscou expandir o artigo de van den Goorbergh et al. (2022), considerando 2 modelos de classificação, 9 métodos de reamostragem e 3 diferentes métricas de avaliação de performance, de modo a trazer mais robustez às conclusões dos autores, mas também investigar o efeito dos métodos de reamostragem como possível solução ao desbalanceamento de classes. Optamos por considerar dados simulados, com uma prevalência de 4.14% da classe positiva, a partir destes dados foram treinados 2 tipos de modelos de classificação, a regressão logística e um *support vector classifier*, conjuntamente com os métodos de reamostragem.

Algo que não fizemos no trabalho foi otimizar o limiar de decisão. Como citado anteriormente, essa otimização é, de certo modo, equivalente a utilizar de métodos reamostragem ou algoritmos de aprendizagem sensíveis a custo. A equivalência se dá, pois, os métodos de reamostragem também mudam a região de decisão e portanto o limiar. Como ambas abordagens têm efeitos similares, deixamos como sugestão para trabalhos futuros estudar a otimização do limiar de decisão, dado que o foco deste trabalho eram as técnicas de reamostragem.

No geral, o comportamento observado foi de melhora considerável no *recall* alinhado à uma piora na precisão. O método ENN foi praticamente idêntico ao caso sem reamostragem. O método ADASYN teve o pior desempenho, considerando quase todas métricas. Considerando a AUC, vimos uma tendência de melhora, principalmente nas estimativas por CV nos dados de treinamento, e principalmente no modelo SVC. Já o *score* F_1 teve tendência de piora, principalmente por conta da precisão deteriorada. O SVC se mostrou mais robusto à essa deterioração no F_1 *score* em comparação à regressão logística.

O *score* de Brier no geral aumentou, o que indica uma piora na calibração. Para averiguar mais a fundo esse ponto, utilizamos diagramas de confiabilidade, de modo que pudemos confirmar visualmente que as estratégias impactaram negativamente e de forma acentuada na calibração dos modelos de classificação. Embora existam métodos de pós-calibração, van den Goorbergh et al. (2022), os considera contraproducentes quando utilizados conjuntamente com métodos de reamostragem.

A propriedade de calibração talvez não seja considerada essencial, mas fica difícil, se não impossível, interpretar o *output* de um modelo que produz um *score* como uma probabilidade, se este valor não possui propriedades que uma probabilidade possuiria. Essas questões entram um pouco na distinção entre as duas culturas de modelagem estatística, expostas no influente artigo de Breiman (2001). Isto é, contextualizando com nosso problema, faz sentido utilizar um modelo mal calibrado que faz boas previsões, ou vice-versa? Depende se o propósito é entender melhor

o processo gerador de dados ou fazer boas previsões. Nos dados que simulamos, o *recall* e AUC melhoraram, já a precisão e os *scores* de Brier e F_1 pioraram.

Por fim, vale mencionar que até mesmo a escolha das métricas, não só no contexto de dados desbalanceados, mas também no geral, é uma área de estudo ativa. O recente artigo de Cullerne Bown (2024) identifica mais de 50 aspectos fundamentais na escolha de métricas de desempenho. As métricas que optamos por considerar já são tradicionais na literatura de modelos preditivos, inclusive em dados desbalanceados (McDermott et al., 2024). Na prática, porém, a escolha das métricas é guiada pelo propósito do modelo e pelos custos das possíveis decisões, o que nos leva naturalmente a um ponto levantado ao longo do texto: se é necessário de qualquer modo considerar de forma aprofundada o problema sendo modelado, e todo seu contexto, seja para escolher um limiar de decisão adequado, ponderando os custos dos diferentes erros, seja para construir um modelo probabilístico, então parece fazer mais sentido utilizar algoritmos de *cost sensitive learning* (por exemplo), pois desse modo o usuário (pesquisador, ou quem quer que esteja modelando um dado problema) tem maior controle sobre as decisões sendo tomadas (Elkan, 2001).

Nessa linha de raciocínio, as *guidelines* mínimas do TRIPOD+AI (Collins et al., 2024), que atualiza o *Transparent Reporting of a multivariable prediction model for Individual Prognosis Or Diagnosis* (TRIPOD), exige a motivação, justificativa e descrição das estratégias de reamostragem, bem como do uso de subsequente de métodos de calibração, sempre que forem utilizados. Concluímos portanto, que os métodos de reamostragem, embora equivalentes a outras abordagens como otimização do limiar de decisão ou algoritmos de *cost-sensitive learning*, tendem a ofuscar a maneira que as regiões de decisão mudam ao tomar repetidas amostras ou remover observações dos dados originais, exigindo cuidado na utilização.

Referências Bibliográficas

Assunção, G. O., Izbicki, R., e Prates, M. O. (2023). Is augmentation effective to improve prediction in imbalanced text datasets?

Batista, G. E., Prati, R. C., e Monard, M. C. (2004). A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD explorations newsletter*, 6(1):20–29.

Bigbossfarin, CC0, via Wikimedia Commons (2021). Bias and variance contributing to total error. Accessed: 2024-08-07.

Blitzstein, J. K. e Hwang, J. (2019). *Introduction to probability*. CRC Press.

Bradley, A. P. (1997). The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159.

Breiman, L. (2001). Statistical modeling: The two cultures (with comments and a rejoinder by the author). *Statistical science*, 16(3):199–231.

Chawla, N. V., Bowyer, K. W., Hall, L. O., e Kegelmeyer, W. P. (2002). Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*.

Collins, G. S., Moons, K. G. M., Dhiman, P., Riley, R. D., Beam, A. L., Van Calster, B., Ghassemi, M., Liu, X., Reitsma, J. B., van Smeden, M., Boulesteix, A.-L., Camaradou, J. C., Celi, L. A., Denaxas, S., Denniston, A. K., Glocker, B., Golub, R. M., Harvey, H., Heinze, G., Hoffman, M. M., Kengne, A. P., Lam, E., Lee, N., Loder, E. W., Maier-Hein, L., Mateen, B. A., McCradden, M. D., Oakden-Rayner, L., Ordish, J., Parnell, R., Rose, S., Singh, K., Wynants, L., e Logullo, P. (2024). Tripod+ai statement: updated guidance for reporting clinical prediction models that use regression or machine learning methods. *BMJ*, 385.

Cortes, C. e Vapnik, V. N. (1995). Support-vector networks. *Machine Learning*.

Cullerne Bown, W. (2024). Sensitivity and specificity versus precision and recall, and related dilemmas. *Journal of Classification*, pages 1–25.

Cyc, Public domain, via Wikimedia Commons (2008). Svm separating hyperplanes. Accessed: 2024-07-13.

Dawid, A. P. (1982). The well-calibrated bayesian. *Journal of the American Statistical Association*, 77(379):605–610.

Derczynski, L. (2016). Complementarity, f-score, and nlp evaluation. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 261–266.

Elkan, C. (2001). The foundations of cost-sensitive learning. In *International joint conference on artificial intelligence*, volume 17, pages 973–978. Lawrence Erlbaum Associates Ltd.

Elor, Y. e Averbuch-Elor, H. (2022). To smote, or not to smote? *arXiv preprint arXiv:2201.08528*.

Fawcett, T. (2004). Roc graphs: Notes and practical considerations for researchers. *Machine learning*, 31(1):1–38.

Fukunaga, K. (2014). *Introduction to Statistical Pattern Recognition, 2nd ed.* Elsevier Science.

Gneiting, T. e Raftery, A. E. (2007). Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378.

Guyon, I. (2003). Design of experiments of the nips 2003 variable selection benchmark. In *NIPS 2003 workshop on feature extraction and feature selection*, volume 253, page 40.

Han, H., Wang, W.-Y., e Mao, B.-H. (2005). Borderline-smote: a new over-sampling method in imbalanced data sets learning. In *International conference on intelligent computing*, pages 878–887. Springer.

Hastie, T., Tibshirani, R., e Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer.

He, H., Bai, Y., Garcia, E. A., e Li, S. (2008). Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pages 1322–1328.

He, H. e Garcia, E. (2009). Learning from imbalanced data. *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*.

He, H. e Ma, Y. (2013). *Imbalanced Learning: Foundations, Algorithms, and Applications*. Wiley.

He, J. (2011). Analysis of rare categories. *Springer, Berlin, Heidelberg*.

Izbicki, R. e dos Santos, T. M. (2020). *Aprendizado de máquina: uma abordagem estatística*. UICLAP.

James, G., Witten, D., Hastie, T., e Tibshirani, R. (2013). *An Introduction to Statistical Learning*. Springer.

Kohavi, R. et al. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145. Montreal, Canada.

Kohavi, R., Wolpert, D. H., et al. (1996). Bias plus variance decomposition for zero-one loss functions. In *ICML*, volume 96, pages 275–283. Citeseer.

Kuhn, M. e Johnson, K. (2013). *Applied Predictive Modeling*. Springer.

Lemaître, G., Nogueira, F., e Aridas, C. K. (2017). Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18(17):1–5.

Machine Learner, CC BY-SA 4.0, via Wikimedia Commons (2014). Svm separating hyperplanes. CC BY-SA 4.0, via Wikimedia Commons. License: https://creativecommons.org/licenses/by-sa/4.0. Accessed: 2024-07-26.

MacQueen, J. et al. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA.

McDermott, M. B. A., Hansen, L. H., Zhang, H., Angelotti, G., e Gallifant, J. (2024). A closer look at auroc and auprc under class imbalance.

Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. MIT Press.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., e Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Platt, J. (1999). Probabilistic outputs for svms and comparisons to regularized likehood methods. *Advances in Large Margin Classifiers*.

Rufibach, K. (2010). Use of brier score to assess binary predictions. *Journal of clinical epidemiology*, 63(8):938–939.

Silva Filho, T., Song, H., Perello-Nieto, M., Santos-Rodriguez, R., Kull, M., e Flach, P. (2023). Classifier calibration: a survey on how to assess and improve predicted class probabilities. *Risk Management and Healthcare Policy*.

Sokolova, M., Japkowicz, N., e Szpakowicz, S. (2006). Beyond accuracy, f-score and roc: a family of discriminant measures for performance evaluation. *American Association for Artificial Intelligence*.

Spiegelhalter, D. J. (1986). Probabilistic prediction in patient management and clinical trials. *Statistics in medicine*, 5(5):421–433.

Steyerberg, E. W., Vickers, A. J., Cook, N. R., Gerds, T., Gonen, M., Obuchowski, N., Pencina, M. J., e Kattan, M. W. (2010). Assessing the performance of prediction models: a framework for traditional and novel measures. *Epidemiology*, 21(1):128–138.

Stone, C. J. (1977). Consistent nonparametric regression. *The annals of statistics*, pages 595–620.

Stone, M. (1974). Cross-validatory choice and assessment of statistical predictions. *Journal of the royal statistical society: Series B (Methodological)*, 36(2):111–133.

Sugiyama, M. (2016). *Introduction to Statistical Machine Learning*. Morgan Kaufmann.

Sun, Y., Wong, A. K. C., e Kamel, M. S. (2009). Classification of imbalanced data: A review. *International Journal of Pattern Recognition and Artificial Intelligence*.

Tarawneh, A. S., Hassanat, A. B., Altarawneh, G. A., e Almuhaimeed, A. (2022). Stop oversampling for class imbalance learning: A review. *IEEE Access*, 10:47643–47660.

Tarimo, C. S., Bhuyan, S. S., Li, Q., Ren, W., Mahande, M. J., e Wu, J. (2021). Combining resampling strategies and ensemble machine learning methods to enhance prediction of neonates with a low apgar score after induction of labor in northern tanzania. *Risk Management and Healthcare Policy*.

van den Goorbergh, R., van Smeden, M., Timmerman, D., e Van Calster, B. (2022). The harm of class imbalance corrections for risk prediction models: illustration and simulation using logistic regression. *Journal of the American Medical Informatics Association*.

Van Rijsbergen, C. J. (1979). Information retrieval. 2nd. newton, ma.

Vapnik, V. N. (1999). An overview of statistical learning theory. *IEEE TRANSACTIONS ON NEURAL NETWORKS*.

Varotto, G., Susi, G., Tassi, L., Gozzo, F., Franceschetti, S., e Panzica, F. (2021). Comparison of resampling techniques for imbalanced datasets in machine learning: Application to epileptogenic zone localization from interictal intracranial eeg recordings in patients with focal epilepsy. *Frontiers in Neuroinformatics*.

Wilson, D. L. (1972). Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man, and Cybernetics*, (3):408–421.

Apêndice - Modelos e Parâmetros

Função Auxiliar

Essa função serve para simplificar o processo de ajuste de vários modelos utilizando reamostragem.

```
def fit_model_with_resampling(X_train, y_train, model, resampler=None):
    if resampler:
        X_res, y_res = resampler.fit_resample(X_train, y_train)
        model.fit(X_res, y_res)
    else:
        model.fit(X_train, y_train)
    return model
```

Regressão Logística

```
model_lr = fit_model_with_resampling(X_train,
                                     y_train,
                                     LogisticRegression(solver='liblinear',
                                     penalty='l2',
                                     random_state=SEED))
```

Support Vector Classifier

```
model_svm = fit_model_with_resampling(X_train,
                                      y_train,
                                      SVC(probability=True,
                                      kernel='rbf',
                                      random_state=SEED))
```

Reamostragem

Podemos criar um objeto que armazena um dicionário com cada método de reamostragem e seus parâmetros. Para verificar os argumentos padrão de cada método consultar a documentação da biblioteca *imbalanced-learn*.

```
samplers = {
    'Sem reamostragem': None,
    'ROS': RandomOverSampler(random_state=SEED),
    'SMOTE': SMOTE(k_neighbors=5, random_state=SEED),
    'Borderline-SMOTE1': BorderlineSMOTE(kind='borderline-1',
                                         random_state=SEED),
    'ADASYN': ADASYN(n_neighbors=5, random_state=SEED),
    'RUS': RandomUnderSampler(random_state=SEED),
    'Cluster Centroids': ClusterCentroids(random_state=SEED),
    'ENN': EditedNearestNeighbours(n_neighbors=3),
    'SMOTEENN': SMOTEENN(random_state=SEED)
}
```