UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

AMOSTRAGEM ALEATÓRIA E EXTENSÕES PARA PREDIÇÃO DE EVENTOS RAROS

Richard Guilherme dos Santos

Dissertação de Mestrado do Programa Interinstitucional de Pós-Graduação em Estatística (PIPGEs)

ICMC USP SÃO CARLOS

Richard Guilherme dos Santos

AMOSTRAGEM ALEATÓRIA E EXTENSÕES PARA PREDIÇÃO DE EVENTOS RAROS

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP e ao Departamento de Estatística – DEs-UFSCar, como parte dos requisitos para obtenção do título de Mestre em Estatística – Programa Interinstitucional de Pós-Graduação em Estatística. *EXEMPLAR DE DEFESA*

Área de Concentração: Estatística

Orientador: Prof. Dr. Carlos Alberto Diniz

USP – São Carlos
Novembro de 2024

Richard Guilherme dos Santos

RANDOM SAMPLING AND EXTENSIONS FOR RARE EVENT PREDICTION

Master dissertation submitted to the Institute of Mathematics and Computer Sciences – ICMC-USP and to the Department of Statistics – DEs-UFSCar, in partial fulfillment of the requirements for the degree of the Master Interagency Program Graduate in Statistics.
EXAMINATION BOARD PRESENTATION COPY

Concentration Area: Statistics

Advisor: Prof. Dr. Carlos Alberto Diniz

USP – São Carlos
November 2024

Independentemente das nossas limitações, nós sempre podemos ser de alguma utilidade. Nosso poder pode parecer insignificante, mas ele pode se revelar útil no grande esquema das coisas. Até um pequeno poder poderia mudar o destino do mundo. É por isso que devemos estar sempre focados e não deixar o momento passar.

AGRADECIMENTOS

Gostaria de expressar minha profunda gratidão a minha família, especialmente aos meus pais, Érica e Michel, por me proporcionarem todas as oportunidades que me permitiram cursar o ensino superior e trilhar o caminho até a pós-graduação. Sou igualmente grato aos meus avós e tios, cujo apoio e carinho foram fundamentais em cada etapa dessa jornada, sempre estando presentes nos momentos em que mais precisei.

Agradeço ainda à CAPES pelo suporte financeiro que viabilizou esta pesquisa durante os dois anos de desenvolvimento. Registro também meu reconhecimento aos professores que participaram da minha banca de defesa e, em especial, ao meu orientador, Carlos Diniz, por sua orientação e pela liberdade concedida durante todo o processo de elaboração deste trabalho.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001

“Independentemente das nossas limitações, nós sempre podemos ser de alguma utilidade. Nosso poder pode parecer insignificante, mas ele pode se revelar útil no grande esquema das coisas.”

(Shikamaru Nara)

RESUMO

SANTOS, R.G. **AMOSTRAGEM ALEATÓRIA E EXTENSÕES PARA PREDIÇÃO DE EVENTOS RAROS**. 2024. 65 p. Dissertação (Mestrado em Estatística – Programa Interinstitucional de Pós-Graduação em Estatística) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2024.

Em problemas de classificação, a predição de eventos raros costuma ser uma questão de difícil resolução. Versões clássicas de algoritmos sofrem diversos problemas ao serem treinados quando a variável resposta é desbalanceada, além de certas métricas, como a acurácia, perderem valor na comparação de diferentes modelos. Neste trabalho, apresentamos diferentes técnicas de *random sampling* e suas aplicações em extensões de técnicas *ensemble* que propõem resolver tal dilema. Embora existam extensões para a maioria dos métodos utilizados em problemas de multiclasse, focamos na sua utilização para problemas dicotômicos. Além disso, realizamos simulações em bases de dados buscando observar vantagens e lacunas dos métodos utilizados, com destaque em uma base de dados de concessão de crédito, onde o desbalanceamento é severo.

Palavras-chave: classificação, eventos raros, machine learning, random sampling.

ABSTRACT

SANTOS, R.G. **RANDOM SAMPLING AND EXTENSIONS FOR RARE EVENT PREDICTION**. 2024. 65 p. Dissertação (Mestrado em Estatística – Programa Interinstitucional de Pós-Graduação em Estatística) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2024.

In classification problems, the prediction of rare events is often a difficult issue to solve. Classical versions of algorithms suffer several problems when being trained when the response variable is unbalanced, and certain metrics, such as accuracy, lose value when comparing different models. In this paper, we present different random sampling techniques and their applications in extensions of ensemble techniques that propose to solve this dilemma. Although extensions exist for most methods used in multi-class problems, we focus on their use for dichotomous problems. In addition, we performed simulations on databases seeking to observe advantages and shortcomings of the methods used, with emphasis on a credit concession database, where the imbalance is severe.

Keywords: classification, rare events, machine learning, random sampling.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

CAPÍTULO

1

INTRODUÇÃO

Em modelos de classificação binária a variável resposta admite apenas dois resultados, chamados usualmente de evento e não-evento. Buscamos prever a probabilidade de ocorrência de um evento utilizando outras variáveis de interesse, chamadas de covariáveis. Em diversos problemas dicotômicos reais, como de detecção de fraudes em cartão de crédito (TAHA; MALEBARY, 2020), análises médicas (YU *et al.*, 2012), reconhecimento facial (YANG *et al.*, 2004), detecção de câncer de mama (NASSER; YUSOF, 2023) e detecção de spam (GUZELLA; CAMINHAS, 2009), a distribuição dos dados é desbalanceada em relação a uma certa classe. Dizemos que um conjunto de dados é desbalanceado quando as categorias das classes alvo estão representadas de forma desigual, sendo chamadas de classe minoritária e classe majoritária aquelas que estiverem em menor e maior proporção, respectivamente. Definimos como *imbalance ratio* (IR) a razão da quantidade de elementos da classe minoritária pela classe majoritária:

$$\text{IR} = \frac{\text{quantidade de elementos da classe minoritária}}{\text{quantidade de elementos da classe majoritária}}.$$

Embora qualquer conjunto em que as classes não estejam igualmente representadas possa tecnicamente ser considerado desbalanceado, é consenso que o desbalanceamento se torna um problema quando a razão é baixa. No Capítulo 2, abordaremos quando o desbalanceamento passa a ser um problema.

Diversos algoritmos de aprendizado de máquina sofrem diferentes problemas ao serem ajustados a este tipo de conjuntos de dados. Em sua estrutura, os algoritmos normalmente esperam que as classes estejam igualmente representadas e atribuam pesos iguais para o erro de classificação de ambas (FERNÁNDEZ *et al.*, 2018). Dentre os problemas mais comuns, temos:

- Classificadores clássicos como regressão logística, *support vector machine (SVM)* e árvores de decisão tendem a obter um bom poder preditivo para a classe majoritária mas distorcer resultados em relação a classe minoritária (LÓPEZ *et al.*, 2013).

- Diversos algoritmos são designados com o objetivo de minimizar erros de classificação de forma a minimizar a acurácia, métrica que possui um viés em direção a classe majoritária quando o conjunto de dados é desbalanceado, levando a algoritmos com baixo poder preditivo em relação à classe minoritária (LOYOLA-GONZÁLEZ *et al.*, 2016).

- Eventos da classe minoritária podem ser considerados como ruído durante o ajuste do modelo. De forma análoga, dados ruidosos ppodem ser erroneamente identificados como instâncias da classe minoritária, uma vez que ambos tem pouca representação no espaço amostral (BEYAN; FISHER, 2015)

Com o crescimento do *Big Data* e o advento de bases de dados cada vez maiores, também tivemos o surgimento de contextos onde o desbalanceamento era cada vez mais presente. Diferentes abordagens foram desenvolvidas para lidar com essa questão. Entre as técnicas mais utilizadas estão as de amostragem aleatória, conhecidas como *random sampling*, que incluem métodos como *random undersampling* e *random oversampling*. Essas técnicas têm como objetivo modificar o conjunto de treinamento para reduzir o nível de desbalanceamento entre as classes. O *random undersampling* atua removendo aleatoriamente amostras da classe majoritária, enquanto o *random oversampling* adiciona cópias aleatórias de amostras da classe minoritária.

Embora eficazes na redução do desbalanceamento, essas abordagens podem levar a problemas como perda de informações importantes ou características representativas da classe majoritária, e *overfitting* causado pela duplicação excessiva de amostras da classe minoritária. Além disso, essas técnicas muitas vezes não consideram a distribuição intrínseca dos dados, o que pode resultar em um conjunto de treinamento menos informativo e, consequentemente, em uma redução na performance do modelo (CHAWLA *et al.*, 2002).

Para superar essas limitações, várias extensões e métodos alternativos foram propostos, visando aprimorar o critério de seleção das instâncias a serem selecionadas e modificadas. Técnicas como *SMOTE (Synthetic Minority Over-sampling Technique)* e suas variações utilizam métodos baseados em interpolação para criar novas amostras sintéticas da classe minoritária, ao invés de simplesmente duplicar amostras existentes, o que ajuda a preservar a diversidade do conjunto de dados. Outras abordagens avançadas incluem métodos baseados em aprendizagem ativa e algoritmos de agrupamento que selecionam instâncias de maneira mais estratégica (HAIXIANG *et al.*, 2017). Essas técnicas têm mostrado serem mais eficazes na manutenção das características relevantes do conjunto de treinamento, ao mesmo tempo que abordam o desbalanceamento de forma mais inteligente e robusta.

Combinações de algoritmos de baixo poder preditivo, frequentemente denominadas técnicas de *ensemble*, são amplamente utilizadas para enfrentar problemas de desbalanceamento de classes. Essas técnicas consistem na combinação de múltiplos modelos fracos, chamados de *weak learners*, para criação de um modelo mais robusto e preciso. Dentre as técnicas mais utilizadas, é possível atribuir pesos distintos para cada classe, permitindo que o modelo final dê

mais importância à elementos da classe minoritária durante o treinamento.

Além disso, muitas variações dessas técnicas incorporam métodos de *random sampling* durante as iterações de treinamento, de forma que em cada, o algoritmo é treinado em uma base de dados cuja proporção de classes é mais equilibrada (CHAWLA *et al.*, 2003).

Técnicas como *Bagging* e *Boosting* são exemplos populares de algoritmos *ensemble* que se beneficiam dessa integração com métodos de amostragem. Enquanto o *Bagging* ajuda a reduzir a variância do modelo combinando várias *weak learners* independentes (BREIMAN, 1996), o *Boosting* foca em reduzir o viés ao ajustar iterativamente modelos fracos, que corrigem os erros de seus predecessores (FRIEDMAN, 2002). Essas abordagens avançadas não apenas melhoram a precisão geral, mas também aumentam a sensibilidade às classes minoritárias, fornecendo um equilíbrio eficaz entre precisão e robustez em conjuntos de dados desbalanceados.

Modificações de algoritmos que atribuem pesos distintos a cada classe, ajustando o erro de classificação de acordo com a importância de cada uma, são conhecidas como *cost-sensitive learning*. Essas técnicas buscam modificar algoritmos já existentes para lidar com o desbalanceamento das classes de dados e estão presentes em diversos algoritmos clássicos. Apesar disso, é uma abordagem frequentemente menos encontrada na literatura (WEISS; MCCARTHY; ZABAR, 2007), com um dos prováveis motivos sendo a necessidade de definir uma matriz de pesos apropriada para o problema de acordo com a importância relativa de cada classe, muitas vezes sendo feita de forma empírica, introduzindo variabilidade e incerteza nos resultados encontrados.

Dada a ampla variedade de técnicas disponíveis, comparamos diferentes abordagens voltadas para modificações de técnicas de *ensemble* e de *random sampling*. Através da análise, buscamos identificar as técnicas que oferecem melhor desempenho em métricas que são mais robustas em relação a eventos raros, como a *Area Under the ROC Curve (ROCAUC)* e a área da curva de Precisão-Recall. Essas métricas são essenciais para avaliar a performance de modelos em cenários onde há desbalanceamento, pois consideram tanto a precisão quanto a sensibilidade, proporcionando uma visão mais completa da capacidade do modelo em detectar corretamente a classe minoritária.

Além disso, abordaremos algumas atenções necessárias para utilização dos algoritmos de *random sampling*, sendo a principal, a definição do melhor *imbalance ratio*. Tal fato é relevante pois tratar problemas cuja origem é desbalanceada em bases totalmete balanceadas leva a uma modelagem imprecisa, incapaz de generalizar o problema para novas instâncias.

Este trabalho está organizado da seguinte forma. No capítulo 2 revisamos a teoria por trás dos algoritmos *ensemble* e de técnicas de *random sampling*, assim como as extensões de ambas ferramentas. No capítulo 3 apresentamos a metodologia utilizada para o ajuste dos hiperparâmetros dos algoritmos e cuidados a serem tomados na aplicação dos métodos analisados. Nos capítulos 4 e 5 as técnicas são aplicadas em problemas de classificação gerados artificial e

em uma base real de concessão de crédito, visando entender como e quais técnicas apresentam bons resultados em problemas genéricos e práticos. Por fim, no capítulo 6 apresentamos os resultados e considerações finais sobre o estudo.

CAPÍTULO

2

REVISÃO TEÓRICA

Qualquer conjunto de dados com distribuição desigual é tecnicamente desbalanceado. Porém, é consenso é consenso a utilização do termo quando uma classe é consideravelmente menos presente que as demais (FERNÁNDEZ *et al.*, 2018). A estes eventos, que estão em menor proporção, damos o nome de eventos raros.

A presença de desbalanceamento nos dados compromete a performance de diversos algoritmos de aprendizado de máquina (HE; GARCIA, 2009), como regressão logística, árvores de decisão, *support vector machines*, redes neurais e *K-Nearest Neighbors* (BATISTA; PRATI; MONARD, 2004). Tais algoritmos assumem o balanceamento da variável resposta ou um erro de classificação igual para cada classe. Dessa forma, quando dados desbalanceados estão presentes , os algoritmos tendem a ter problemas com a previsão dos eventos de principal interesse.

Nesse cenário, métricas usuais em problemas de classificação, como a acurácia, passam a ter certos problemas na avaliação de performance de diferentes modelos. Esses problemas acontecem em conjuntos em que o desbalanceamento é extremo, como em casos dicotômicos onde 98% dos dados pertencem a uma única classe. Um algoritmo simples que prevê todos os dados pertencente ao grupo majoritário obtém 98% de acurácia, mesmo sendo incapaz de distinguir os eventos da variável resposta (CHAWLA, 2009). Dessa forma, são necessárias métricas diferentes e mais informativas em relação à qualidade do modelo, como *ROCAUC*, *f1-score*, precisão e *recall*.

No geral, existem diversas abordagens que visam solucionar problemas em relação ao desbalanceamento dos dados. Estas podem ser divididas em três estratégias principais (HAIXIANG *et al.*, 2017):

1. **Cost sensitive learning:** buscam adaptar classificadores para incluir um viés com a classe minoritária através de modificações na estrutura dos algoritmos ou atribuição de pesos a cada instância. Dessa forma, eventos da classe minoritária passam a ter mais relevância no

treinamento.

2. **Data level:** tem como objetivo balancear a distribuição da variável resposta no conjunto de treinamento através da remoção ou adição de novas instâncias a partir de determinadas regras específicas.

3. **Métodos Ensemble:** Consistem da combinação de vários algoritmos para a criação de um mais estável. Tais técnicas são chamadas de *ensemble* e englobam algoritmos como *Bagging, AdaBoost, Random Forests* e *Gradient Boosting*.

A seguir, introduzimos a teoria por trás dos principais algoritmos relacionados à segunda e terceira estratégia, assim como algoritmos utilizados como base durante o trabalho.

2.1 Regressão Logística

O modelo de regressão logística é um caso particular de modelos lineares generalizados. Seja π_i a probabilidade da observação i assumir o valor 1, podemos definir a função de probabilidade de Y da forma

$$f_Y(y_i|\pi_i) = \pi_i^{y_i}(1-\pi_i)^{1-\pi_i}, 0 < \pi_i < 1. \tag{2.1}$$

Reescrevendo a Equação (2.1) no formato da família exponencial:

$$f_Y(y_i|\pi_i) = \exp\left[y_i \log\left(\frac{\pi_i}{1-\pi_i}\right) - (-\ln(1-\pi_i))\right].$$

Para calcular a probabilidade de uma determinada observação pertencer a uma classe, a regressão logística utiliza a função de ligação logit, obtendo o modelo de regressão logística com p covariáveis:

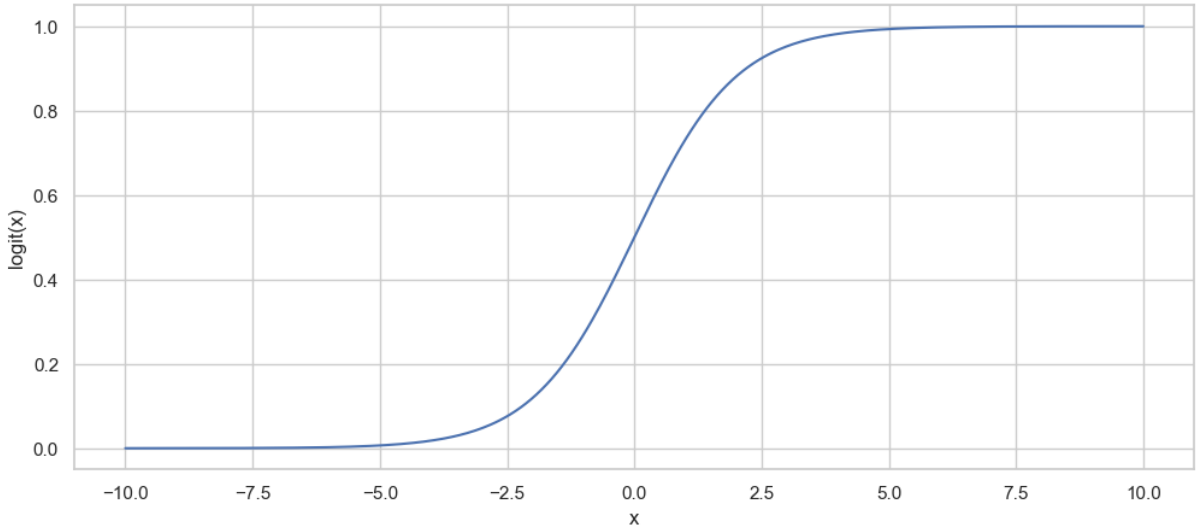
$$\ln\left(\frac{\pi_i}{1-\pi_i}\right) = \sum_{i=0}^{p} \beta_i x_i, \quad x_0 = 1.$$

Alternativamente, o modelo pode ser expresso da seguinte forma:

$$P(Y_i = 1|x_i) = \frac{\exp(\sum_{i=0}^{p} \beta_i x_i)}{1 + \sum_{i=0}^{p} \beta_i x_i}.$$

A regressão logística possuí como característica o uso de uma função de ligação simétrica, que pode não ser ideal para problemas desbalanceados. Os trabalhos (LEMONTE; BAZÁN, 2018) e (BAZÁN; ROMEO; RODRIGUES, 2014) destacam outras funções de ligação adequadas para problemas de classificação em dados desbalanceados, que, no entanto, não foram aplicadas neste trabalho devido ao escopo delimitado.

Figura 1 – A simetria da função logit, usada como função de ligação na regressão logística, implica que a transformação dos valores ajusta as probabilidades de forma equilibrada em torno de 0.5. Essa característica faz com que a função logit trate a probabilidade de evento e não-evento como igualmente distantes dos extremos (0 e 1), assumindo uma distribuição simétrica dos dados ao redor desse ponto.



Fonte: Imagem elaborada pelo autor.

2.2 Técnicas Ensemble

Weak learners são algoritmos de aprendizado de máquina que no geral tendem a possuir uma precisão pouco acima de uma escolha aleatória, sendo os mais comuns as árvores de decisão e o algoritmo *Naive Bayes*. Métodos *Ensemble* consistem em combinar tais algoritmos em um único, buscando melhorar a precisão e diminuir a variância do modelo final utilizando diferentes estratégias (HASTIE *et al.*, 2009). Dentre as técnicas mais utilizadas estão *Bagging, Random Forests, AdaBoost* e *Gradient Boosting*.

Bagging ou *Bootstrap Aggregating* é um algoritmo que se baseia na construção de modelos independentes a partir de amostras aleatórias do conjunto de treinamento (BREIMAN, 1996). Do conjunto de treinamento original, novas bases de mesmo tamanho, ou menor, são criadas. Frequentemente, as amostras são retiradas com reposição, permitindo a ocorrência de observações repetidas em cada base visando aumentar a variabilidade dos modelos que os compõem. Em seguida, cada novo conjunto de treinamento é utilizado para ajustar um *weak learner*, com os modelos podendo ser do mesmo tipo ou distintos, tornando o *ensemble* ainda mais diversificado. Por fim, para a predição de novas instâncias, agregamos as predições de cada modelo que compõe o *bagging* e utilizamos as técnicas de *majority voting* ou *soft voting* de forma a resumir todas as predições do conjunto dos modelos utilizados, que podem ser encontradas na tabela 1.

Tabela 1 – Diferentes métodos de agregação de predições para o *Bagging*.

Majority voting	**Soft voting**
Prevê a classe mais frequente dentre todas as previstas pelos modelos	Prevê a classe que possuir maior probabilidades entre as previstas pelos modelos

Random Forests é uma modificação do método *Bagging* que visa aumentar ainda mais a variabilidade dos modelos que o compõem. Tal fato é importante pois parte da premissa de que quanto maior for a variabilidade dos modelos, melhor será a capacidade de previsão do modelo final, visto que diferentes modelos conseguirão explicar características que não são captadas pelos demais (BREIMAN, 1996). Árvores de decisão são ajustadas em amostras *bootstrap* com apenas uma parte das variáveis exploratórias utilizadas para a criação de cada árvore, obtendo uma menor correlação entre as árvores que compõem o modelo final e, consequentemente, reduzindo a variância geral do algoritmo (HASTIE *et al.*, 2009). O Algoritmo 1 exemplifica os passos para sua execução.

Algoritmo 1 – *Random Forests*

1. Para $m = 1$ até M:
   a) Selecione uma amostra *bootstrap*, $T_{bootstrap}$, do conjunto de treinamento.
   b) Selecione q covariáveis das p presentes em $T_{bootstrap}$.
   c) Ajuste uma árvore de decisão h_m utilizando apenas as q covariáveis selecionas no passo anterior;
2. Saída: Combinação de M árvores de decisão, $\{h_m\}_{m=1}^{M}$.

Boosting é uma estratégia genérica que busca aprimorar o desempenho de qualquer algoritmo de aprendizado. Tem como ideia principal o treinamento sucessivo de *weak learners*, dando mais ênfase a exemplos classificados incorretamente pelos classificadores anteriores. Com isso, cada algoritmo concentra-se nos erros anteriores, corrigindo-os e melhorando o desempenho geral do modelo. Tal método foi inicialmente proposto para problemas de classificação através do algoritmo *AdaBoost* (FREUND; SCHAPIRE; ABE, 1999) e aperfeiçoado com a técnica *Gradient Boosting* (FRIEDMAN, 2001).

Considere $Y \in \{-1, 1\}$ e $h(x)$ um classificador qualquer. Defina a taxa de erro da amostra no treinamento por:

$$\text{err} = \frac{1}{n}\sum_{i=1}^{n} I(y_i \neq h(x_i)),$$

onde I é a função indicadora de evento.

O algoritmo *AdaBoost* ajusta os modelos sequencialmente ao conjunto de treinamento a versões com pesos distintos para cada instância, produzindo classificadores $h_m(x), m = 1,2,\ldots,M$ e combinando-os de forma que o final seja uma soma ponderada destes:

$$h(x) = \text{sign}\left(\sum_{m=1}^{M} \alpha_m h_m(x)\right),$$

onde $\alpha_1,\ldots,\alpha_M$ são as contribuições de cada classificador calculados pelo Algoritmo 2.

As instâncias são modificados a cada iteração através da associação aos pesos $w_1, w_2, \ldots, w_n$. Inicialmente, todas as observações possuem a mesma relevância, $w_i = \frac{1}{n}$. A cada iteração, as instâncias que foram erroneamente classificadas tem seus pesos incrementados, de forma que as de mais difícil classificação ganhem influência ao decorrer do treinamento, fazendo com que cada próximo classificador se concentre nos erros cometidos pelos anteriores.

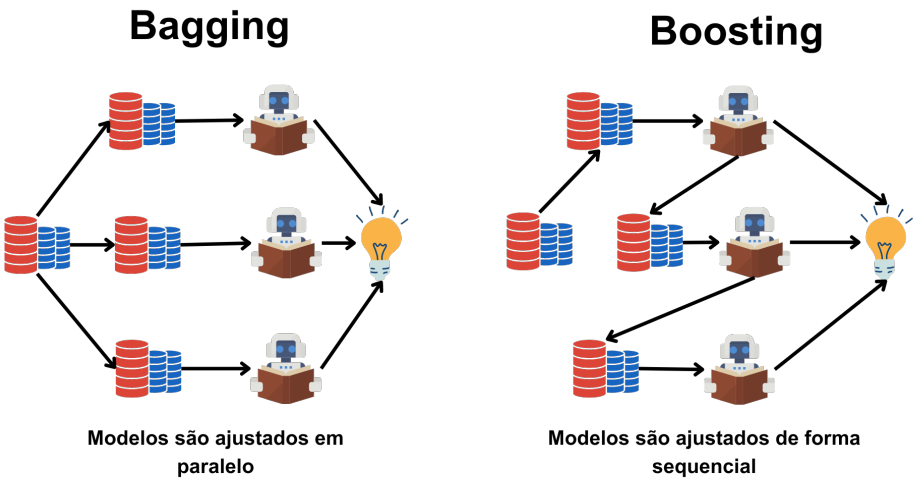
Algoritmo 2 – *AdaBoost.M1*

1. Iniciar com $Y \in \{-1,1\}$.
2. Ajustar os pesos das observações, $w_i = \frac{1}{n}, i = 1,2,\ldots,n$.
3. Para $m = 1$ até M:
   a) Ajuste o classificador $h_m(x)$ no conjunto de treinamento utilizando os pesos w_i.
   b) Calcule
   $$\text{err}_m = \frac{\sum_{i=1}^{n} w_i I(y_i \neq h_m(x_i))}{\sum_{i=1}^{n} w_i}.$$
   c) Calcule $\alpha_m = \ln\left(\frac{1-\text{err}_m}{\text{err}_m}\right)$.
   d) Faça $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq h_m(x_i))], i = 1,2,\ldots,n$.
4. Saída: $h(x) = \text{sign}[\sum_{m=1}^{M} \alpha_m h_m(x)]$.

Existem diversas variantes e otimizações do algoritmo *AdaBoost.M1*, como *AdaBoost.M2*, *Real AdaBoost* e *Gentle AdaBoost*. Neste trabalho, utilizaremos a versão *SAMME.R*, padrão da biblioteca *sklearn* da linguagem *Python*, com o Algoritmo 3 representando sua execução para problemas binários. A principal mudança é relacionada a forma em que os pesos são atualizados, utilizando a função de perda *log-loss* em vez do erro de classificação e a saída do classificador final passando a ser uma probabilidade, tornando a convergência do algoritmo mais rápida (ZHU *et al.*, 2006).

A generalização do método *AdaBoost*, chamado de *Gradient Boosting*, tem como estratégia para prever a relação f entre X e Y minimizar uma função de perda diferenciável $L(y, f)$

Figura 2 – Diferenças no treinamento das técnicas *ensemble* de *Bagging* e *Boosting*. Na primeira, cada modelo pode ser treinado de forma individual e independe dos demais. No segundo, cada modelo é dependente de seus anteriores, buscando prever erros de classificação que foram ignorados.



Fonte: Imagem adaptada de Towards Data Science.

utilizando o método de Gradiente Descendente. Buscamos encontrar $\hat{f}$ de modo que:

$$\hat{f} = \arg\min_{f} \sum_{i=1}^{n} L(y_i, f(x_i)). \tag{2.2}$$

Para isto, ajustamos modelos $h_1(x), h_2(x), \ldots, h_m(x)$ de forma iterativa. No primeiro passo, definimos uma constante $f_0(x)$ que minimize a Equação 2.2 como previsão inicial de cada instância. A partir disso, atualizamos a variável resposta como o oposto do gradiente da função de perda considerando o conjunto $(x_i, f_0(x_i))_{i=1}^{N}$, isto é:

$$r_i^{(1)} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f=f_0}, i = 1, \ldots, n.$$

Ajustamos o modelo $h_1(x)$ no conjunto $\left(x_i, r_i^{(1)}\right)_{i=1}^{N}$ para que este possa corrigir os erros na predição realizada por $f_0(x)$. Atualizamos então o classificador:

$$f_1(x) = f_0(x) + h_1(x).$$

Repetimos o procedimento um número máximo de M iterações ou até um critério de parada. Dessa forma, o $m-$ésimo classificador busca ajustar os erros realizados pelos seus anteriores, com a previsão para a $i-$ésima instância sendo dada por:

$$y_i^{(m)} = f_0(x_i) + \sum_{i=1}^{m} h_i(x_i) = f_m(x). \tag{2.3}$$

Algoritmo 3 – AdaBoost SAMME.R

1. Inicializar com $Y \in \{1,2\}$.
2. Ajustar os pesos das observações, $w_i = \frac{1}{n}, i = 1,2,\ldots,n$.
3. Para $m = 1$ até M:
   a) Ajustar um classificador no conjunto de treinamento utilizando os pesos w_i.
   b) Obter a probabilidade associada a classe pelo estimador do passo anterior:
   $$p_k^{(m)}(x) = P(k|x), k = 1,2.$$
   c) Defina
   $$h_k^{(m)}(x) = 0.5 \left(\ln p_k^{(m)} - 0.5 \sum_{k' \neq k} \ln p_{k'}^{(m)}(x) \right), k = 1,2.$$
   d) Atualizar os pesos:
   $$w_i = \begin{cases} w_i \exp\left(0.5 \ln p_1^{(m)}(x)\right), y_i = 1 \\ w_i \exp\left(0.5 \ln p_2^{(m)}(x)\right), y_i = 2 \end{cases}$$
   e) Normalizar w_i.
4. Saída:
   $$h(x) = \arg\max_k \sum_{m=1}^{M} h_k^{(m)}(x).$$

Para evitar *overfitting*, duas técnicas são bastante utilizadas. *Shrinkage* consiste da adição de uma taxa de aprendizado η a cada termo $h_i(x)$ na equação (2.3), com objetivo de controlar a velocidade de convergência do algoritmo, suavizando as atualizações efetuadas a cada iteração. Tal parâmetro pode ser constante ou atualizar baseado nos erros do modelo, sendo a primeira mais comum devido a sua simplicidade. A segunda é uma modificação do algoritmo chamada *Stochastic Gradient Boosting*, em que cada *weak learner* ajustado, consideramos amostras *bootstrap* de tamanho menor do conjunto de treinamento original, visando aumentar a independências entre os *weak learners* e, consequentemente, diminuir a variância do modelo final.

O Algoritmo 4 exemplifica os passos para execução do *Gradient Boosting* com uma taxa de aprendizado constante. Para a versão original basta definir $\eta = 1$.

Algoritmo 4 – *Gradient Boosting*

1. Iniciar $f_0(x) = \arg\min_\gamma \sum_{i=1}^{N} L(y_i, \gamma)$
2. Para $m = 1$ até M:
   a) Para $i = 1 \ldots, n$ calcule:
   $$r_i^{(m)} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x_i)=f_{m-1}(x_i)}$$
   b) Ajuste $h_m(x)$ nas instâncias $\left(x_i, r_i^{(m)}\right)_{i=1}^{N}$
   c) Atualize o m-ésimo classificador:
   $$f_m(x) = f_{m-1}(x) + \eta h_m(x)$$
3. Saída: $\hat{f}(x) = f_M(x)$

2.3 Random Sampling

Diversos algoritmos de classificação são criados sob a hipótese de que o conjunto de treinamento seja balanceado, perdendo performance quando ajustados em dados cuja proporção de classes seja assimétrica (FERNÁNDEZ *et al.*, 2018). Assim, abordagens que alteram o conjunto de treinamento de forma a balancear as classes são criadas, existindo três grupos principais que podem ser vistas na tabela 2.

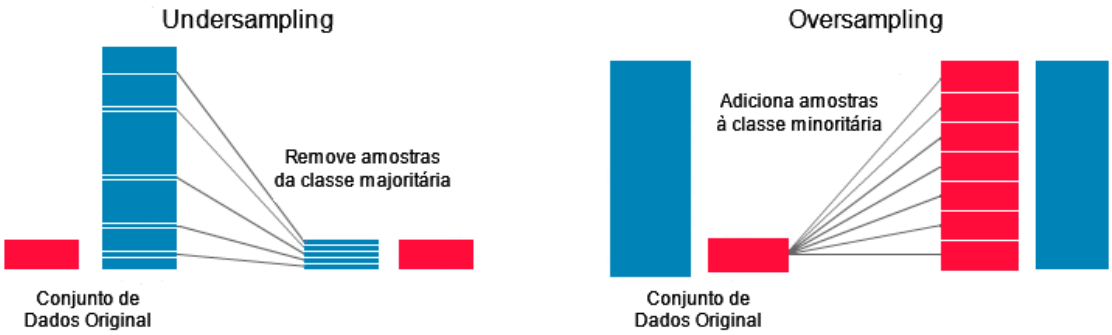
Tabela 2 – Diferentes métodos de Random Sampling.

Undersampling	**Oversampling**	**Híbridos**
Cria um subconjunto do original eliminando instâncias da classe majoritária	Cria um novo conjunto do original replicando instâncias da classe minoritária	Combinam ambas as abordagens anteriores

As estratégias mais clássicas para as duas primeiras abordagens são as técnicas de *Random Undersampling (RUS)* e *Random Oversampling (ROS)*. Na figura 3, temos um exemplo de sua aplicação a um conjunto desbalanceado. No geral, o *imbalance ratio* do conjunto resultante é 1, porém qualquer proporção desejada de classes pode ser atingida, visando manter certa assimetria que muitas vezes é inerente ao problema.

- **Random Undersampling (RUS)**: Exclui amostras da classe majoritária de forma aleatória até obter um conjunto de treinamento com o balanceamento desejado.
- **Random Oversampling (ROS)**: Equilibra a distribuição das classes replicando aleatoriamente exemplos da classe minoritária.

Figura 3 – Técnicas de *random undersampling* e *random oversampling* sendo aplicado em um conjunto de dados.



Fonte: Imagem adaptada de ResearchGate.

Ambas as abordagens são bastante debatidas na literatura, com diversas questões e certas limitações técnicas sendo frequentemente discutidas. Em *random undersampling*, existe a possibilidade de excluir dados potencialmente úteis e relevantes para o processo de ajuste do modelo, fazendo com que ele sofra de *underfitting* por faltar informação relevante no conjunto de treinamento. Enquanto isso, a técnica *random oversampling* realiza a adição de cópias exatas de instâncias, além de conflitar com a hipótese a independência entre observações no conjunto de treinamento necessária em certos algoritmos, a estratégia pode levar ao *overfitting* do modelo, de modo que ele aprenda exatamente o comportamento dos exemplos do treinamento, mas não consiga se adaptar a novas instâncias. Nesta técnica, o tempo de ajuste do modelo também aumenta consideravelmente, podendo ser inviável na prática dependendo da volumetria da base de dados utilizada. Por fim, o próprio *imbalance ratio* do conjunto em que o método foi aplicado não é um consenso, sendo estabelecido de forma empírica ou obtendo um conjunto resultante balanceado pela implementação usual do algoritmo (FERNÁNDEZ *et al.*, 2018). Desta forma, diversas técnicas foram elaboradas na literatura para solucionar os problemas citados anteriormente.

2.3.1 NearMiss

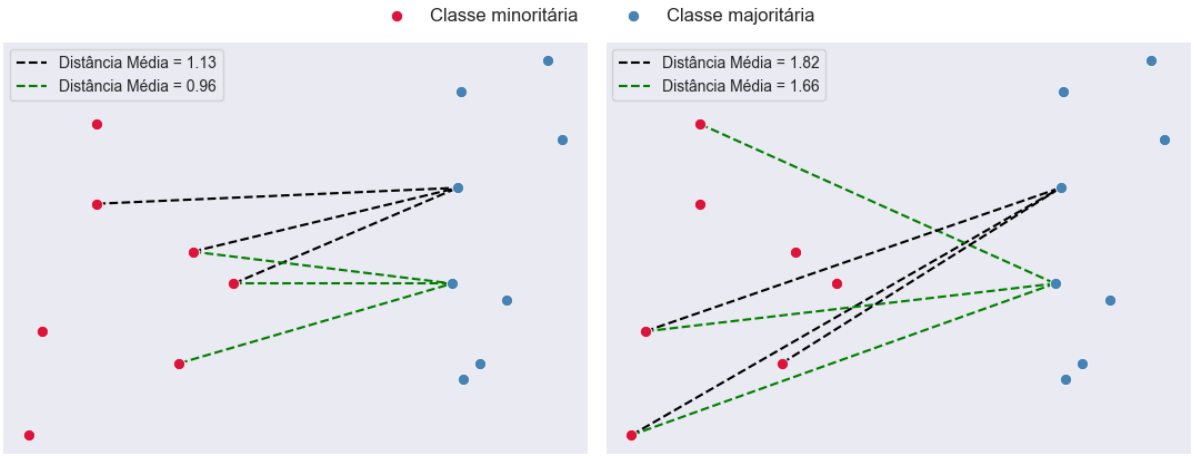
O algoritmo *NearMiss* é uma técnica de *random undersampling* baseada no algoritmo *K-Nearest Neighbors* (*K-NN*), selecionando instâncias mais relevantes a partir de um determinado padrão de acordo com os vizinhos de exemplos do conjunto de dados (MANI; ZHANG, 2003). A técnica possui duas versões principais que ditam diferentes estratégias para escolha dos pontos mantidos no conjunto de treinamento, sendo elas:

- **NearMiss 1:** Para cada vetor de covariáveis da classe majoritária, encontramos seus k vizinhos mais próximos que pertencem a classe minoritária. Calculamos a média da distância euclidiana entre todos os vizinhos para cada elemento da classe majoritária.

Mantemos as instâncias no conjunto de treinamento cuja média das distâncias seja a menor possível até obtermos uma proporção desejada.

- **NearMiss 2:** Para cada vetor de covariáveis da classe majoritária, encontramos os k vetores de covariáveis da classe minoritária cuja distância seja a maior. Calculamos a média da distância euclidiana entre todos os vizinhos para cada elemento da classe majoritária. Mantemos as instâncias no conjunto de treinamento cuja média das distâncias dos k vetores encontrados seja a menor possível.

Figura 4 – Exemplo dos algoritmos *NearMiss-1* e *NearMiss-2* da esquerda para a direita, respectivamente. Na primeira imagem selecionamos dois exemplos da classe majoritária e seus três vizinhos mais próximos. Mantemos as instâncias cuja distância média é a menor, neste caso referente a linha verde. Na segunda imagem, o procedimento é parecido. Agora, para cada elemento da classe majoritária, os vizinhos mais distantes são escolhidos. Por fim, selecionamos o ponto que possui a menor distância média entre tais vizinhos para participar do conjunto de treinamento. O processo é realizado em toda a base de forma a obter um conjunto de treinamento com um *imbalance ratio* alvo.



Fonte: Imagem adaptada de imblanced-learn.

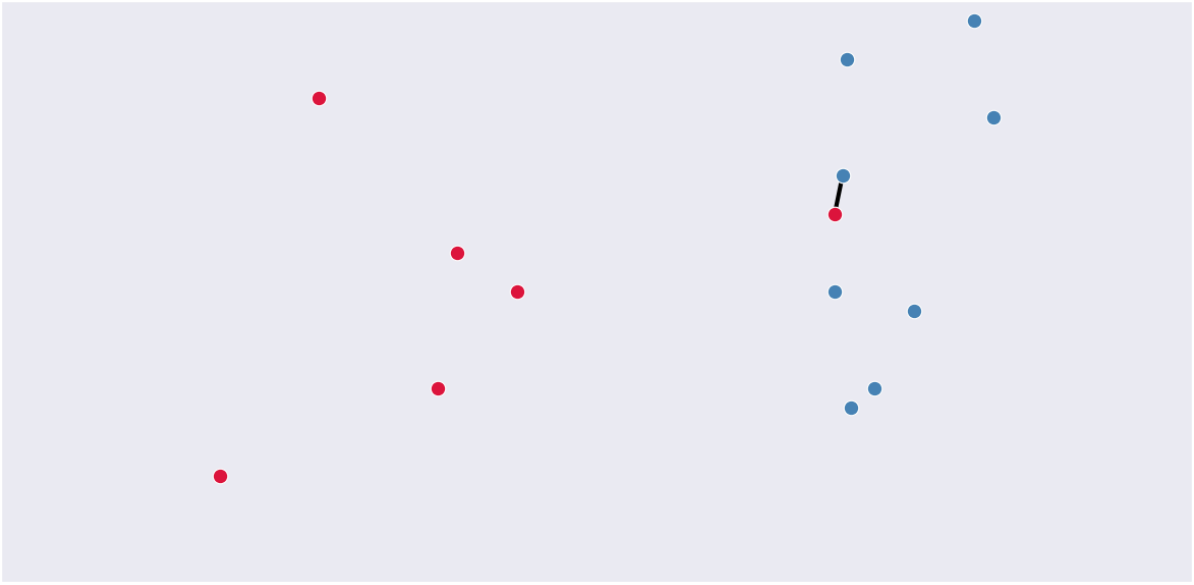
2.3.2 TomekLinks

Técnica de *random undersampling* que busca eliminar os *Tomek links* entre duas amostras (IVAN, 1976). Um *Tomek link* entre duas instâncias de classes distintas x_1 e x_2 é definido de forma que estas sejam vizinhas mais próximas uma da outra, isto é, para qualquer observação z:

$$d(x_1,x_2) < d(x_1,z) \text{ e } d(x_1,x_2) < d(x_2,z).$$

onde $d(\cdot)$ é a distância euclidiana. No método, instâncias da classe majoritária que possuem *Tomek's links* são removidas do conjunto de treinamento.

Figura 5 – Exemplo de *Tomek's Link*. Pontos na extremidade do segmento de reta definem um *Tomek's Link*. No exemplo, a instância da classe majoritária, referente a cor vermelha, será removida do conjunto de treinamento.

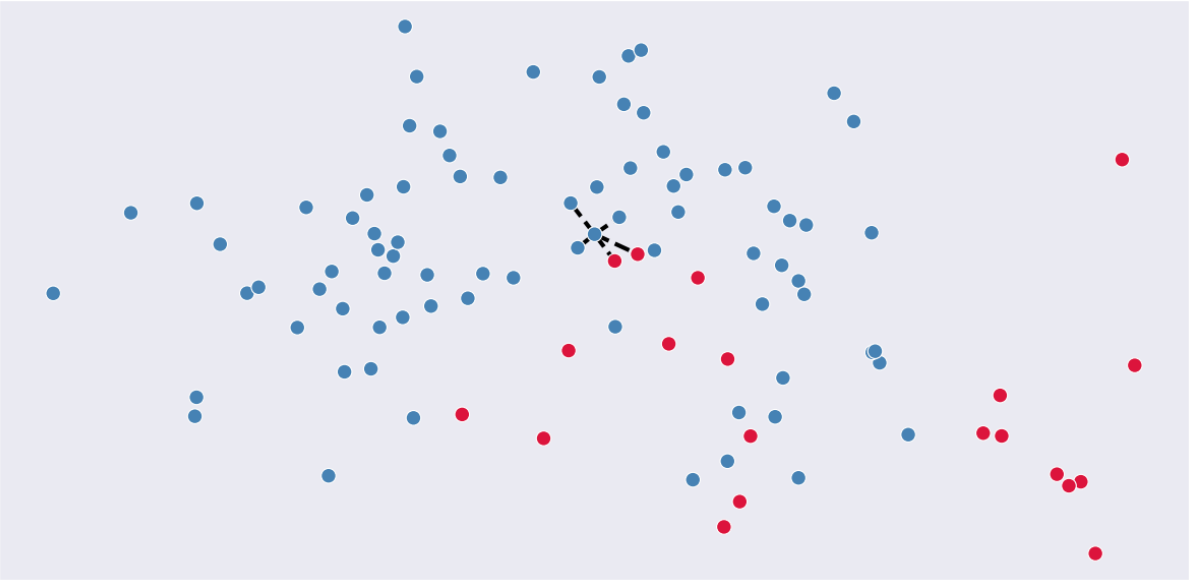


Fonte: Imagem adaptada de imblearn.

2.3.3 EditedNearestNeighbours

Técnica de *random undersampling* tem como estratégia a remoção de observações que não se assemelham o suficiente com seus k vizinhos mais próximos (WILSON, 1972). Para cada instância da classe majoritária, observa-se a moda de seus k vizinhos mais próximos e caso estas sejam diferentes, a instância é tratada como ruído e removida do conjunto de treinamento. O método ainda tem uma segunda versão, na qual há a necessidade da concordância de classes entre os vizinhos ser total, fazendo com que todos tenham que pertencer a mesma classe para permanência da observação no conjunto de treinamento. Esta, por sua vez, será a utilizada durante as simulações.

Figura 6 – Utilização da Técnica *Edited Nearest Neighbours*. Selecionamos um ponto da classe majoritária e seus 5 vizinhos mais próximos. No exemplo, caso a moda seja considerada como regra o ponto permanecerá no conjunto de treinamento, pois esta é condizente com a classe da instância. Caso todas as classes das vizinhanças precisem ser iguais, então este será removido do treinamento.

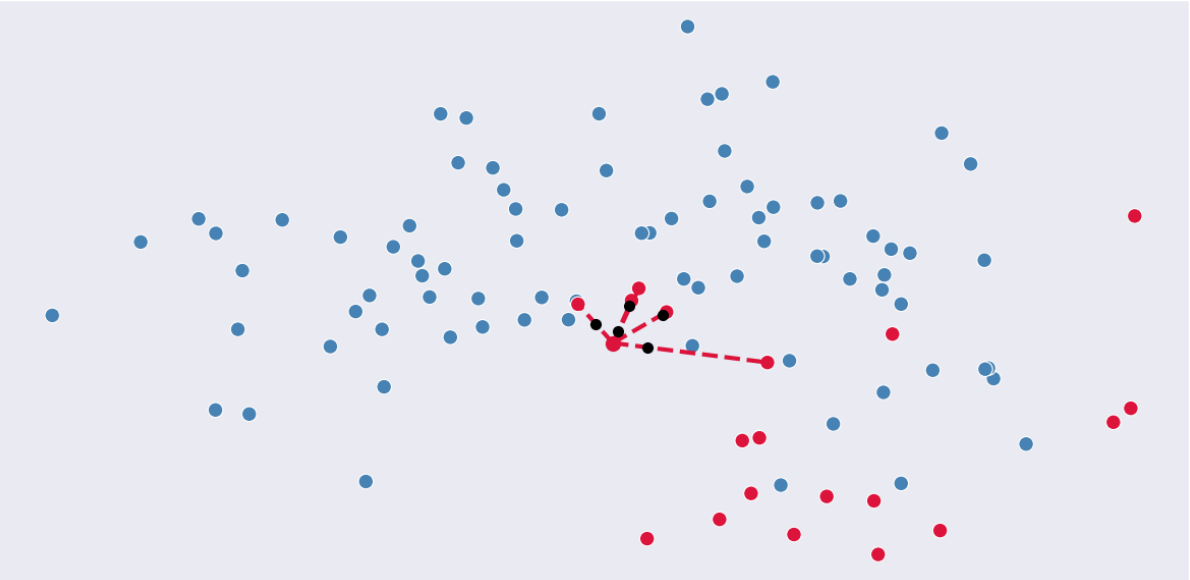


Fonte: Imagem elaborada pelo autor.

2.3.4 Synthetic Minority Over-sampling Technique (SMOTE)

Técnica de *oversampling* que introduz novos exemplos da classe minoritária de forma sintética, criados a partir da interpolação dos vizinhos mais próximos dos elementos da classe minoritária.

Figura 7 – Exemplo de utilização da técnica SMOTE. Selecionamos aleatoriamente um elemento da classe minoritária e seus k vizinhos mais próximos, também pertencentes a classe minoritária. Para cada vizinho, um ponto aleatório no segmento de reta é selecionado e então é adicionado ao conjunto de treinamento até que um *imbalance ratio* pré-fixado seja atingido.



Fonte: Imagem elaborada pelo autor.

O método consiste em selecionar aleatoriamente instâncias da classe minoritária e seus k vizinhos mais próximos. Em seguida, selecionamos aleatoriamente um ponto do segmento formado por estes dois e o adicionamos ao conjunto de treinamento. Repetimos o processo até que um certo *imbalance ratio* definido inicialmente aconteça (CHAWLA *et al.*, 2002). Na figura 7, temos o exemplo de seu funcionamento.

2.3.5 Extensões do SMOTE

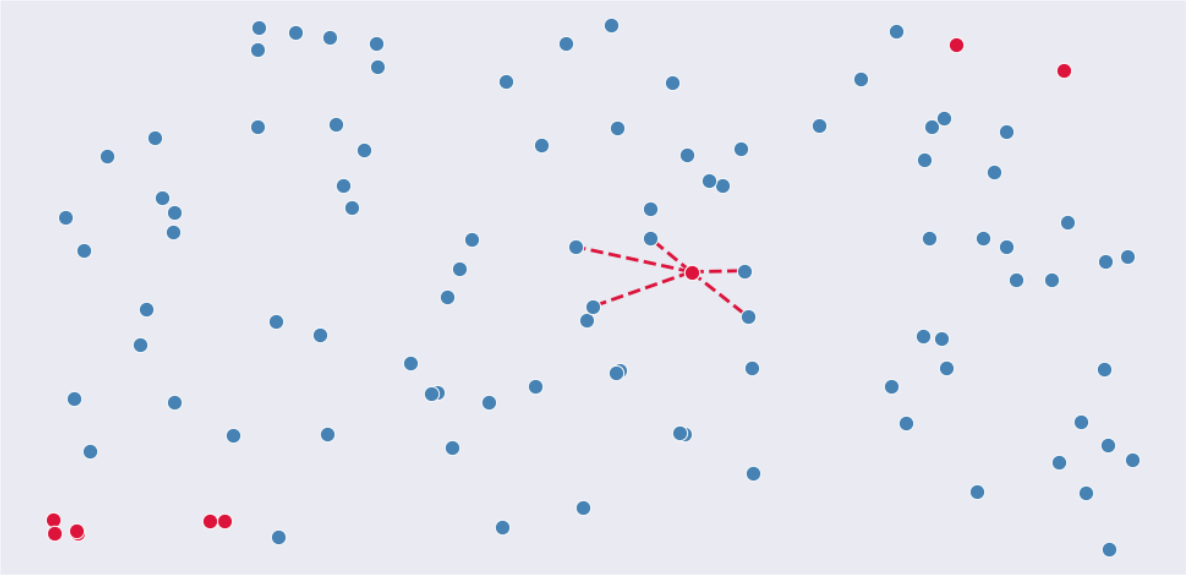
A técnica *SMOTE* se popularizou devido a sua capacidade de gerar novas instâncias parecidas com as anteriores em vez de simplesmente duplicar exemplos existentes como na técnica de *random oversampling*. Com o tempo, diversas variantes foram criadas com objetivo de contornar certas limitações do método que poderiam afetar a performance de algoritmos preditivos. Em geral, as extensões evitam com que a seleção de instâncias a serem criadas durante o algoritmo *SMOTE* sejam aleatórias, buscando aqueles exemplos que são consideradas mais difíceis de serem classificados. Em geral, os principais problemas em selecionar instâncias de forma aleatória são:

- Novas instâncias podem ser criadas de modo que estas sigam padrões próximos dos demais da já existente, fazendo com que o algoritmo ganhe pouco ou nenhum poder preditivo em troca de um aumento considerável no tempo de treinamento (HE *et al.*, 2008).

- Instâncias da classe minoritária na fronteira da tomada de decisão muitas vezes são considerados exemplos de classificação mais difícil. Replicar amostras longe da fronteira, não interfere no poder preditivo do algoritmo, mas aumenta seu tempo computacional (HAN; WANG; MAO, 2005).

- Diferentes dados ruidosos estão presentes nos membros da classe minoritária. Permitir que a técnica gere exemplos sintéticos a partir destes elementos faz com que novos ruídos sejam desnecessariamente criados no conjunto de treinamento (LAST; DOUZAS; BACAO, 2017).

Figura 8 – Exemplo de problema da técnica SMOTE. Em instâncias de evento isoladas, a técnica replica ruídos de maneira artificial, dificultando a capacidade do modelo em realizar predições.



Fonte: Imagem elaborada pelo autor.

Adaptative synthetic sampling (ADASYN) é uma técnica que tem como estratégia gerar uma quantidade maior de dados sintéticos das instâncias consideradas de mais difícil classificação (HE *et al.*, 2008). Isto é feito através da distribuição de densidade das amostras, com o Algoritmo 5 representando seu funcionamento. Para cada instância da classe minoritária, consideramos esta sendo de difícil classificação quanto maior for a quantidade de elementos da classe majoritária entre seus k vizinhos mais próximos. Dessa forma, geramos uma quantidade maior de exemplos sintéticos pela técnica *SMOTE* de dados que possuem uma maior quantidade de elementos da classe majoritária em suas vizinhanças.

Algoritmo 5 – *Adaptive synthetic sampling* - ADASYN

1. Calcular o número de instâncias sintéticas necessárias para serem geradas:

$$R = \frac{n_{\text{maj}} - n_{\text{min}}}{\text{IR}},$$

onde IR é o *imbalance ratio* desejado no conjunto de treinamento resultante, n_{maj} e n_{min} são a quantidade de elementos da classe majoritária e minoritária, respectivamente.

2. Para cada instância x_i da classe minoritária, encontrar as suas k vizinhanças mais próximas.

3. Calcular a razão:

$$r_i = \frac{k_{\text{maj},i}}{k}, i = 1, 2, \ldots, n_{\text{min}}$$

sendo k_{maj} o número de exemplos da classe majoritária pertencentes a K vizinhança mais próxima de x_i.

4. Normalizar r_i de forma que este passe a ser uma distribuição de densidade:

$$\hat{r}_i = \frac{r_i}{\sum_{i=1}^{n_{\text{min}}} r_i},$$

isto é, $\sum_{i=1} \hat{r}_i = 1$.

5. Para cada instância da classe minoritária x_i gerar $\hat{r}_i \cdot R$ amostras sintéticas utilizando a técnica *SMOTE* com k vizinhanças.

Para atingir uma melhor precisão, diversos algoritmos buscam aprender a dividir a fronteira de cada classe no espaço de covariáveis da melhor forma possível durante o processo de treinamento, sendo os exemplos da fronteira, e seus mais próximos, os mais prováveis de serem classificados erroneamente daqueles longe desta e, consequentemente, de mais importante classificação. Baseado nisso, a técnica *Borderline-SMOTE* foi desenvolvida para gerar exemplos sintéticos apenas nas regiões fronteiriças, também chamada de *borderline*, da classe minoritária (HAN; WANG; MAO, 2005). O Algoritmo 6 exemplifica a execução da técnica *Borderline-SMOTE1* .

Figura 9 – Exemplo de problema da técnica Borderline-SMOTE. Apenas para os dados da fronteira de decisão a técnica SMOTE é utilizada.

Fonte: Imagem elaborada pelo autor.

Algoritmo 6 – *Borderline SMOTE-1*

1. Para cada instância da classe minoritária x_i, calcular seus m vizinhos mais próximos.
2. Seja m_{maj} a quantidade de vizinhos de x_i pertencentes a classe majoritária:
   - Se $m_{maj} = m$ então x_i é considerado como ruído.
   - Se $\frac{m}{2} \leq m_{maj} < m$ então x_i é considerado como ponto da fronteira e adicionado a um conjunto suporte R.
   - Caso $m_{maj} < \frac{m}{2}$ então x_i é considerado como uma instância de fácil classificação.
3. Os exemplos em R são os da fronteira do classe minoritária. Para cada exemplo, aplicamos a técnica *SMOTE* considerando os k vizinhos mais próximos.

A técnica ainda possui uma segunda versão chamada *Borderline-SMOTE2*, onde a estratégia *SMOTE* não apenas interpola as instâncias da classe minoritária entre si, mas também com os da majoritária para criação dos exemplos sintéticos. Em contrapartida, estes novos exemplos ficarão mais próximos da classe minoritária, interpolando os exemplos artificias por um valor entre 0 e 0.5.

2.3.6 Combinações de Técnicas

Todas as extensões da técnica *SMOTE* visam solucionar problemas relacionados à escolha das novas amostras geradas. Técnicas que selecionam os melhores exemplos seguindo algum critério específico foram definidas, contudo é possível combinar técnicas de *random undersampling* de forma que tais ruídos sejam eliminados antes da aplicação da técnica *SMOTE*

e o conjunto de treinamento tenha parte de seus ruídos removidos.

Assim, as técnicas *Tomek's link* e *Edited Nearest-Neighbours* foram utilizadas para limpeza das bases de dados antes da execução da técnica *SMOTE*, visando obter um conjunto de dados mais parcimonioso. Tais combinações são chamadas de *SMOTETomek* e *SMOTEENN*, respectivamente.

2.4 Modificações de Técnicas Ensemble

Técnicas *ensemble* são bastante utilizadas em conjuntos desbalanceados por, no geral, terem um bom poder preditivo e possuírem modelos que conseguem atribuir pesos a instâncias da classe minoritária. Além disso, diversas extensões foram criadas com objetivo de combinar técnicas de *random sampling* no ajuste dos *weak learners* que os compõem. Nesta seção estudaremos tais extensões.

EasyEnsemble é uma combinação do algoritmo *bagging* onde classificadores *AdaBoost* são ajustados em conjuntos de dados balanceados utilizando a técnica *random undersampling* para então serem combinados em um único classificador por *soft voting* (LIU; WU; ZHOU, 2008). O Algoritmo 7 exemplifica seu funcionamento, onde M classificadores *AdaBoost* são ajustados em conjuntos balanceados. Para cada classificador, s_m *weak learners* são treinados e então combinados por *soft voting*. Por fim, todos os *weak learners* são combinados para formar o classificador final.

Algoritmo 7 – *EasyEnsemble*

1. Para $m = 1$ até M:
   a) Selecione um subconjunto balanceado T_m do conjunto de treinamento original utilizando a técnica de *random undersampling*.
   b) Ajuste um modelo *AdaBoost SAMME.R* h_m com s_m *weak learners* $h_{m,i}$ e $\alpha_{m,i}$ seus pesos correspondentes:

$$h_m(x) = \text{sign}\left(\sum_{i=1}^{s_m} \alpha_{m,i} h_{m,i}(x)\right)$$

2. Resultado final um classificador *Bagging* com M classificadores *AdaBoost*:

$$h(x) = \text{sign}\left(\sum_{m=1}^{M}\sum_{i=1}^{s_m} \alpha_{m,i} h_{m,i}(x)\right).$$

As demais extensões se baseiam em estratégias parecidas, buscando modificar o conjunto de treinamento de forma a este ser mais balanceado durante as iterações de algum algoritmo *ensemble*, fazendo com que o viés da classe majoritária diminua. Temos uma certa preferência

a técnicas que diminuam a correlação dos classificados, uma vez que quanto maior for a variabilidade dos modelos que compõem as técnicas *ensemble*, melhor tende a ser seu desempenho (HASTIE *et al.*, 2009). Dentre as extensões consideramos:

- **RUSBoost**: Técnica onde o *AdaBoost* tem cada *weak learner* ajustado em base balanceada com a técnica *random undersampling* (SEIFFERT *et al.*, 2009).
- **OverBoost**: Técnica onde o *AdaBoost* tem cada *weak learner* ajustado em base balanceada com a técnica *random oversampling* (CHAWLA *et al.*, 2003).
- **SMOTEBoost**: Técnica onde o *AdaBoost* tem cada *weak learner* ajustado em base balanceada com a técnica *SMOTE* (CHAWLA *et al.*, 2003).
- **BalancedBagging**: Modelo de *Bagging* onde cada *weak learner* é ajustado em uma base balanceada com a técnica *random undersampling*. Caso os *weak learners* escolhidos sejam modelos *AdaBoost* então trabalhamos com um modelo *EasyEnsemble*.
- **BalancedRandomForests**: Modelo de *Random Forests* onde cada árvore de decisão é ajustada em uma base balanceada com a técnica de *random undersampling*.

Restringimos as modificações a um leque menor das técnicas de *random sampling* devido a dois motivos principais: sua implementação em bibliotecas populares de aprendizado de máquina e para facilitar a quantidade de métodos a serem comparados e ajustados durante as simulações.

Por fim, destacamos que técnicas como *Tomek's links* e *Condensed nearest neighbors* são evitadas, uma vez que em diferentes execuções não há modificação do conjunto de treinamento resultante destas. Assim, de certa forma há um ganho de correlação nos *weak learners* que compõem o *ensemble*.

Além disto, modificações onde o balanceamento é realizado por meio de amostras *bootstrap* são possíveis com *BalancedBaggingg* e *BalancedRandomForests* (LÓPEZ *et al.*, 2013) sendo modificações das técnicas de *Bagging* e *Random Forests* em que para cada iteração, as amostras *boostrap* são selecionadas de forma balanceada, visando diminuir a correlação entre os classificadores que compõem o *ensemble*.

2.5 Métricas de Validação

A performance de algoritmos de classificação é realizada através de informações contidas em uma tabela chamada matriz de confusão, vista na tabela 3. As linhas da matriz consistem das previsões para as classes enquanto as colunas são seus valores reais.

Como elementos que compõem a matriz temos:

Tabela 3 – Matriz de Confusão

		Real	
		Sim	Não
Previsto	Sim	Verdadeiro Positivo (TP)	Falso Positivo (FP)
	Não	Falso Negativo (FN)	Verdadeiro Negativo (TN)

- **Verdadeiro Negativo (TN)**: Número de não eventos corretamente classificados.
- **Falso Positivo (FP):** Número de não eventos incorretamente classificados como eventos.
- **Falso Negativo (FN):** Número de eventos incorretamente classificados como não eventos.
- **Verdadeiro Positivo (TP):** Número de eventos corretamente classificados.

Uma das métricas mais utilizadas em problemas de classificação é a proporção total de acertos realizada pelo modelo, também conhecida como acurácia:

$$\text{Acurácia} = \frac{\text{TP+TN}}{\text{TP+FP+TN+FP}}.$$

Porém, esta métrica sofre de problemas em dados desbalanceados, sendo incapaz de informar se um modelo consegue separar as classes das instâncias observadas. Como exemplo, suponhamos um caso onde apenas 1% das instâncias sejam referentes a classe de interesse, um modelo que prevê todas as instâncias referentes a classe majoritária obtém 99% de acurácia, mas não consegue distinguir os dados de forma alguma.

A partir disso, da tabela 3 derivamos outras métricas chamadas de precisão e *recall*, que medem a quantidade de eventos corretamente classificados e a taxa de previsões de eventos classificada corretamente pelo modelo, respectivamente.

$$\text{Precisão} = \frac{TP}{TP+FP} \quad \text{e} \quad \text{Recall} = \frac{TP}{TP+FN}.$$

Em conjuntos desbalanceados, buscamos aumentar o *recall* sem diminuir a precisão (CHAWLA, 2009). Porém, como aumentar um normalmente costuma diminuir o outro, sendo um caso de *trade-off* (BUCKLAND; GEY, 1994), podemos otimizar a média harmônica entre estas, também conhecido como *f1-score*:

$$F_1 = 2 \cdot \frac{\text{Precisão} \cdot \text{Recall}}{\text{Precisão} + \text{Recall}}.$$

2.5.1 Curva ROC

Para entender a performance do modelo em diferentes pontos de corte, utilizamos a área da curva *Receiver Operating Characteristic (ROC)*. Tal curva é um gráfico do desempenho das

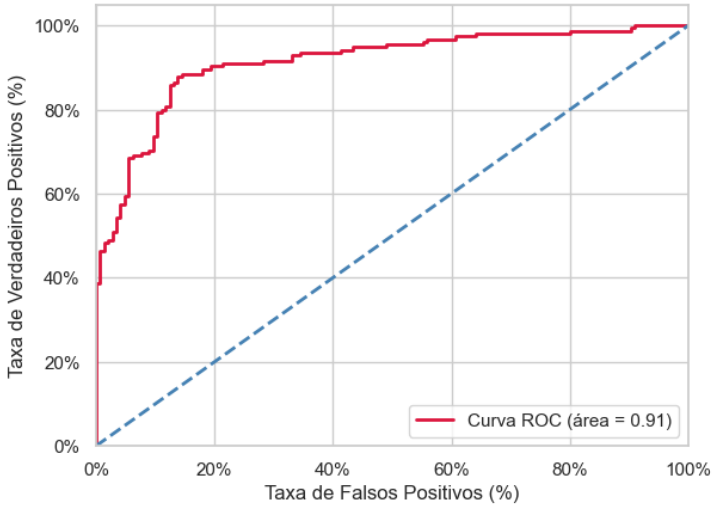
taxas de falso positivo e verdadeiro positivo do modelo ao variar diferentes pontos de corte. Estas por sua vez, são definidas como:

$$\text{Taxa de verdadeiro positivo} = \text{TPR} = \frac{TP}{TP+FN}, \quad \text{Taxa de falso positivo} = \text{FPR} = \frac{FP}{FP+TN}.$$

A área sob a curva ROC (AUC - *Area Under the Curve*) fornece uma medida única da capacidade do modelo de classificar corretamente os exemplos, sendo um valor próximo a 1 indicativo de excelente performance e valores próximos a 0,5 indicando que o modelo não é melhor do que o acaso.

Ponto mais próximo do canto superior esquerdo: Um método comum é identificar o ponto na curva que está mais próximo do canto superior esquerdo do gráfico (0,1). Esse ponto representa a melhor combinação de TPR e FPR.

Figura 10 – Curva ROC.



Fonte: Imagem elaborada pelo autor.

2.5.2 Kolmogorov-Smirnov (KS)

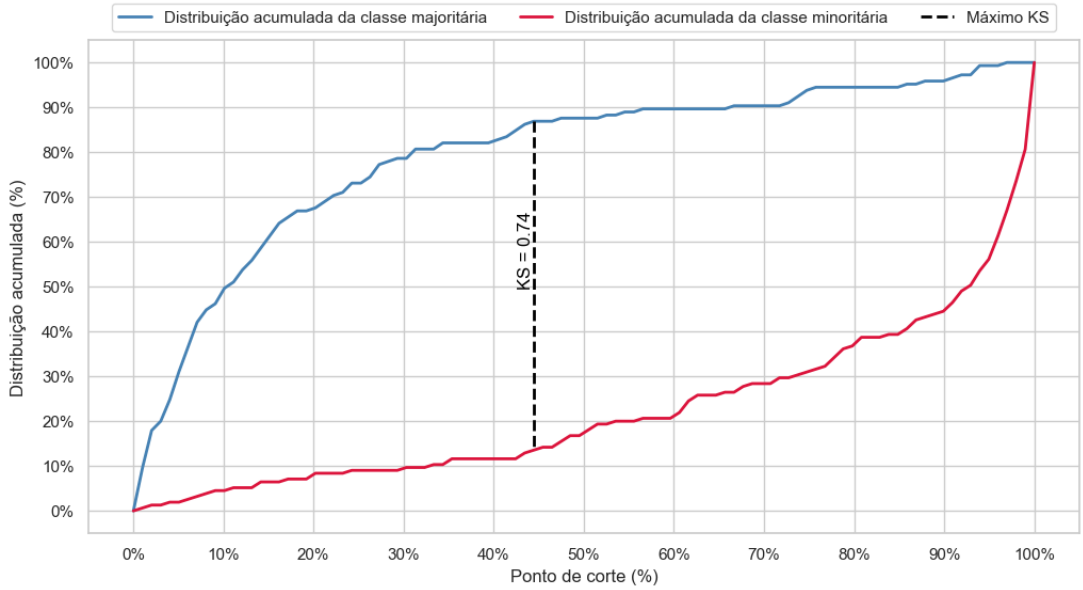
A métrica Kolmogorov-Smirnov (KS) é uma ferramenta estatística amplamente utilizada no mercado de crédito para confecção de modelos de *scoring*. O teste de Kolmogorov-Smirnov é uma técnica não paramétrica que mede a discrepância entre duas distribuições empíricas, permitindo avaliar a capacidade de um modelo em distinguir entre duas classes diferentes.

A estatística KS é definida como a máxima diferença absoluta entre as funções de distribuição acumulada (CDFs) de duas amostras. Em um contexto de classificação binária, essas amostras representam as distribuição de probabilidades atribuídas as classes de evento e não-evento. A fórmula da estatística KS é:

$$D = \max |F_1(x) - F_2(x)|$$

Em que $F_1(x)$ e $F_2(x)$ são as funções de distribuição acumulada da classe positiva e da classe negativa, respectivamente. O valor de D varia entre 0 e 1, em que 0 indica que as distribuições são idênticas e 1 indica que elas são completamente diferentes.

Figura 11 – Estatística Kolmogorov-Smirnov (KS).



Fonte: Imagem elaborada pelo autor.

O valor da estatística KS fornece uma medida intuitiva da separabilidade entre as classes. Em geral, valores de KS acima de 0,3 são considerados indicativos de uma boa separação entre as classes, enquanto valores abaixo de 0,1 podem indicar que o modelo não está conseguindo distinguir adequadamente entre elas.

2.6 Analisando o problema de desbalanceamento

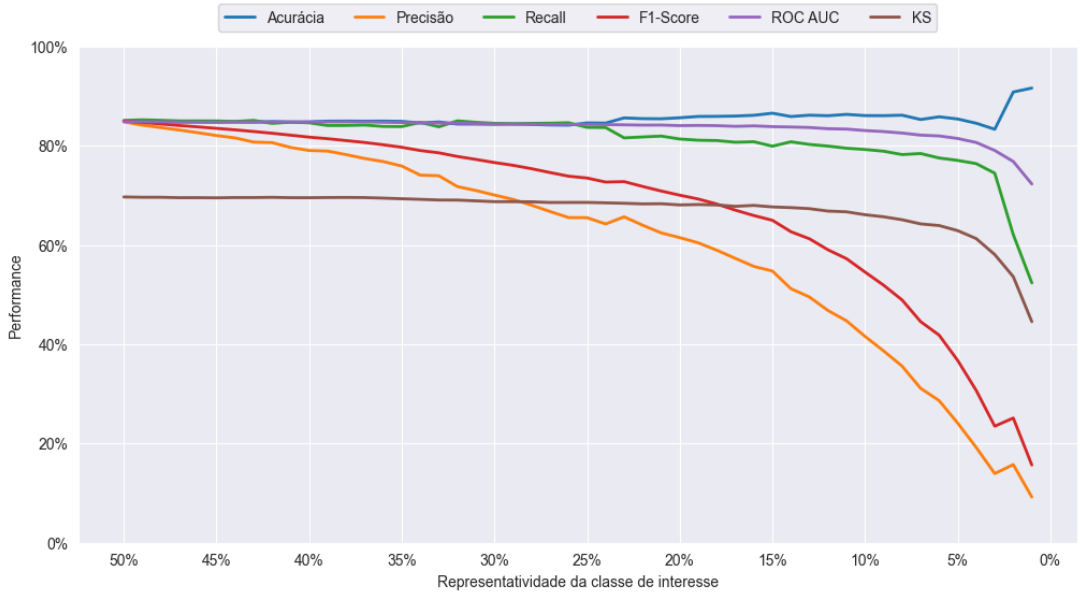
Até o momento usamos o termo ”problema de desbalanceamento“ como uma afirmação verdadeira, mas é necessário analisar porque e quando dados desbalanceados podem ser um problema para análise de dados.

Para isto, simulamos bases de dados com 100.000 observações e 10 covariáveis com diferentes níveis de *imbalance ratio*, utilizando a função *make_classification* do *sk-learn*. Separamos as bases em treinamento, e validação com as proporções 70-30, respectivamente e para cada ajustamos um modelo de regressão logística com os pontos de corte sendo otimizados de acordo com a curva ROC.

A figura 12 mostra a performance do modelo para bases de dados com diferentes níveis de proporção para a variável resposta. Observamos que quanto menor a representatividade da classe de interesse, menor é o desempenho do modelo em relação a precisão e, consequentemente, *f1-score*. *Recall* por sua vez apresenta perda de performance apenas para bases cujo evento de interesse é menor do que 3%.

Além disso, é possível perceber a estabilidade da ROC-AUC para dados desbalanceados.

Figura 12 – Performance do modelo de regressão logística em bases de dados com diferentes níveis de desbalanceamento.

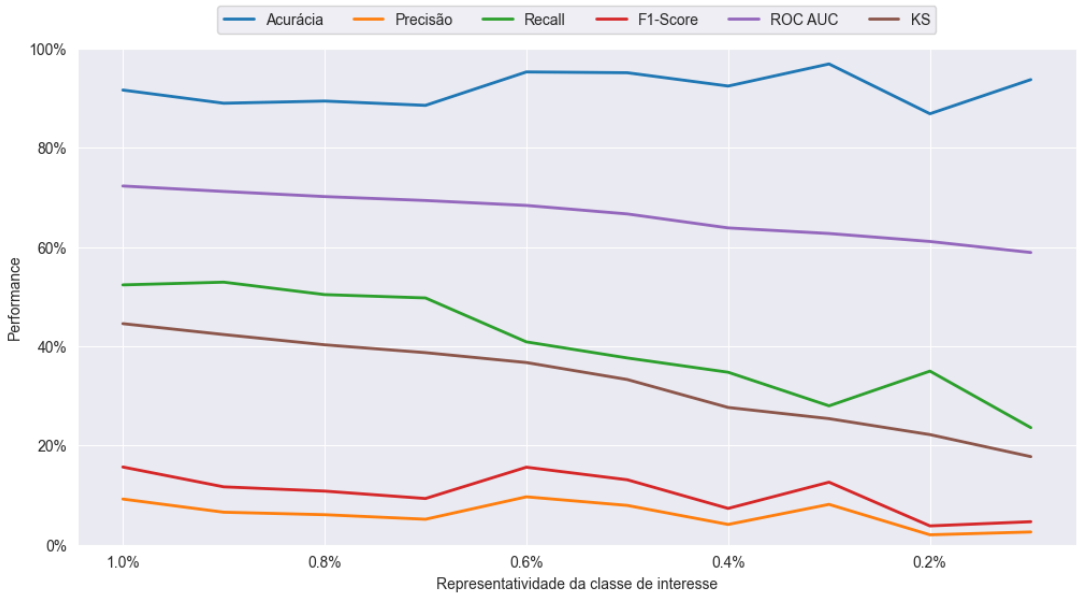


Fonte: Imagem elaborada pelo autor.

Na figura 13, aumentamos o desbalanceamento e a perda de performance continua progressivamente.

Por fim, apresentamos diferentes algoritmos para lidar com o desbalanceamento de dados a partir da premissa de que utilizar o *random sampling* para balancear a base de dados pode ser algo a melhorar o desempenho dos modelos clássicos, mas não apresentamos nenhuma demonstração desse efeito. Para isto, utilizaremos um caso particular onde cinco bases de dados diferentes foram simuladas. As bases possuem diferente nível de representatividade para o evento de interesse, variando de 5% até 1% e para cada uma delas uma regressão logística foi ajustada com e sem *random undersampling*.

Figura 13 – Performance do modelo de regressão logística quando o desbalanceamento é extremo.



Fonte: Imagem elaborada pelo autor.

CAPÍTULO

3

METODOLOGIA

3.1 Simulação das Bases

A função *make_classification*, presente na biblioteca *sk-learn*, é uma ferramenta poderosa para a geração de conjuntos de dados sintéticos destinados a problemas de classificação. Utilizada amplamente para experimentação e validação de algoritmos, essa função permite a criação de bases de dados que simulam diversas estruturas de dados, com possibilidade de ajuste de ruído e distribuição. Utilizaremos esta função para analisar o desempenho das técnicas e modelos em dados sintéticos.

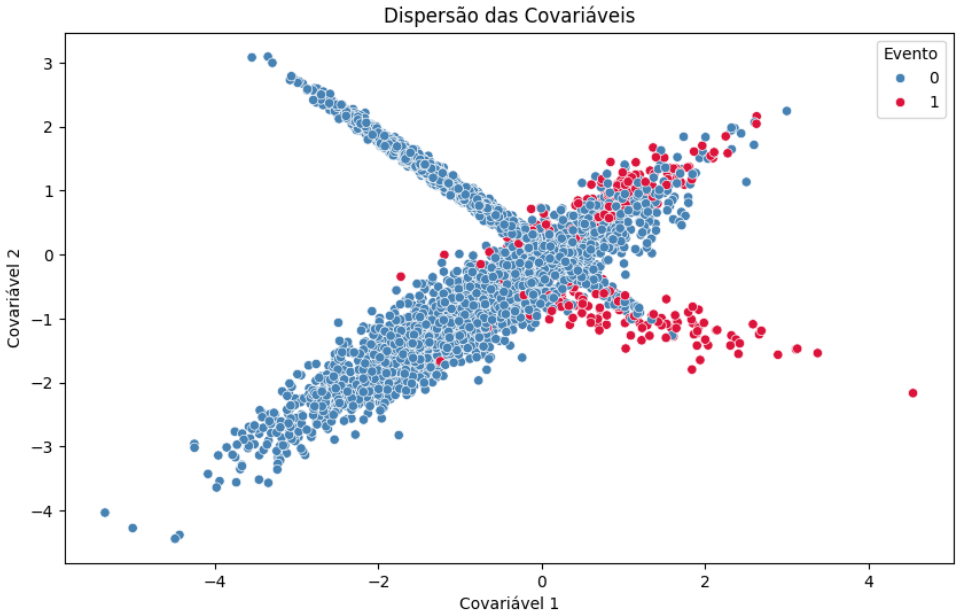
A função gera as bases utilizando uma combinação de variáveis informativas, redundantes e repetidas, cada um com um papel específico para o problema. A primeira são aquelas que possuem de fato relação com a variável resposta, enquanto as segundas são utilizadas para trazer complexidade para o problema, seja gerando combinação linear das variáveis informativas ou simplesmente repetindo variáveis já existentes na simulação. Os dados gerados são todos contínuos, distribuídos normalmente em torno de vértices de um hipercubo n-dimensional.

Além disso, o processo introduz interdependência entre as características, gerando correlações significativas entre elas. Por fim, introduzimos ruído de forma que uma parcela baixa de instâncias pertencentes a variável resposta tenham seu rótulo aleatoriamente atribuído. A figura 14 exemplifica uma base gerada através desse método para duas dimensões.

3.2 Validação Cruzada

A validação cruzada, também conhecida como *cross-validation*, é uma técnica essencial em aprendizado de máquina e estatística para avaliar o desempenho de modelos de predição. Em essência, a validação cruzada visa verificar a capacidade de generalização e estabilidade de um modelo, ou seja, sua aptidão para fazer previsões precisas em novos dados não vistos,

Figura 14 – Na esquerda, dados gerados utilizando a função *make-classification* utilizando 2-*clusters*.



Fonte: Imagem elaborada pelo autor

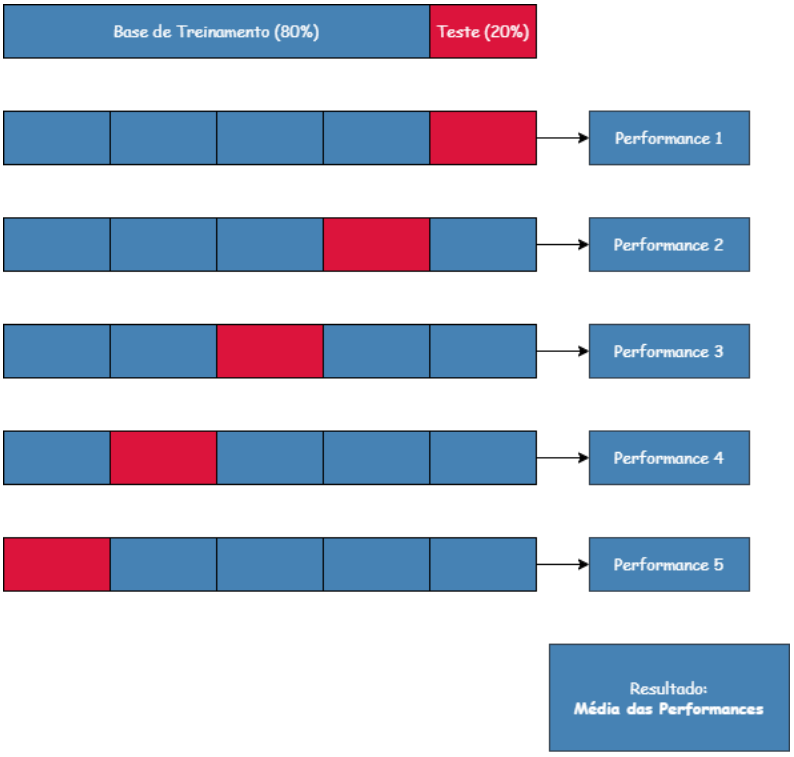
a partir de um conjunto de dados limitado, evitando efeitos aleatórios como a seleção de um conjunto de treinamento específico. Essa técnica é especialmente útil para evitar problemas como o *overfitting*, quando o modelo ajusta-se excessivamente aos dados de treinamento, capturando ruído e detalhes específicos ao invés de padrões generalizáveis.

O principal objetivo da validação cruzada é obter uma estimativa mais robusta e confiável do desempenho do modelo, sem depender de uma única divisão entre dados de treino e teste. Em muitas situações, dividir o conjunto de dados em duas partes, uma para treino e outra para teste, não é ideal, pois pode levar a uma avaliação enviesada, especialmente em conjuntos de dados pequenos. A validação cruzada resolve essa limitação, permitindo que cada dado contribua tanto para o treinamento quanto para a validação do modelo, o que resulta em uma avaliação mais equilibrada.

A abordagem mais comum de validação cruzada é a validação cruzada *k-fold*, em que o conjunto de dados é dividido em k subconjuntos, ou *"folds"*, de tamanho aproximadamente igual. O processo pode ser visto na figura 15 e é exemplificado abaixo:

- O conjunto de dados é dividido em k partes ou *folds*;
- Para cada parte, separamos uma base de treinamento e uma base de teste, de forma que as bases de treinamento entre as partes não possuam elementos repetidos;
- O modelo é ajustado em cada conjunto de treinamento e calculamos as métricas de interesse para o respectivo conjunto de validação;

Figura 15 – Exemplo do Método de Validação Cruzada com 5−*folds*.



Fonte: Imagem elaborada pelo autor.

- O desempenho do modelo para o problema é definido como a média dos desempenhos nas bases de validação.

No contexto de bases desbalanceadas, utilizamos um caso particular da validação cruzada, a validação cruzada estratificada *(Stratified Cross-Validation)*. Nesse contexto, é importante que a proporção das classes seja preservada em cada *fold* para uma avaliação mais precisa. A validação cruzada estratificada garante que cada *fold* tenha uma distribuição similar das classes, evitando que a variabilidade entre as bases de treinamento afete a avaliação do modelo.

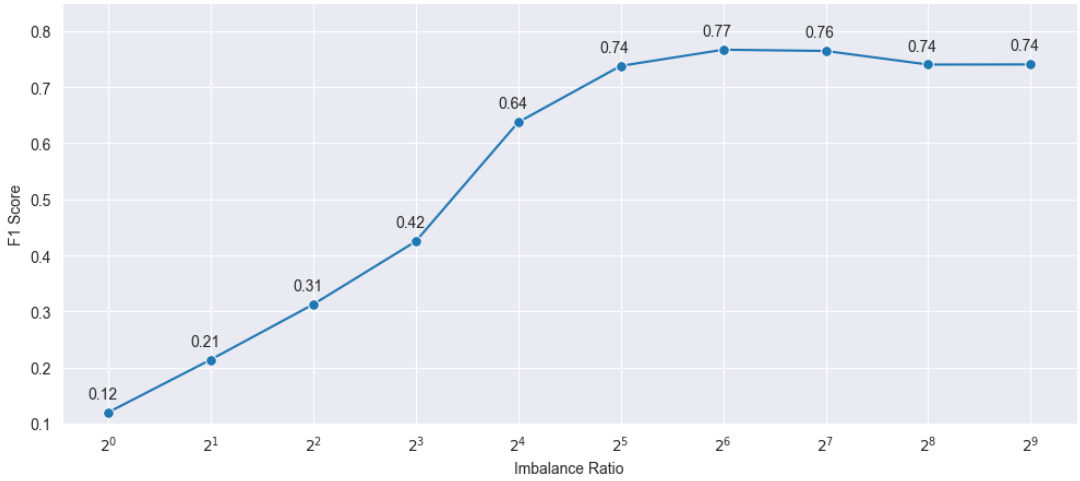
3.3 Seleção dos Melhores Modelos

Para avaliar o desempenho das técnicas utilizamos os modelos de Árvore de Decisão, *AdaBoost*, *Bagging*, *Random Forest*, Regressão Logística e *Gradiente Boosting* na sua implementação pela biblioteca *XGBoost*.

Todos os modelos foram treinados em sua versão padrão da biblioteca *sk-learn* sem e com as técnicas de *random sampling* apresentadas. Apenas o parâmetro de *imbalance-ratio* foi otimizado da seguinte maneira definindo:

$$K = \{1, \ldots, \text{Imbalance Ratio da base}\}.$$

Figura 16 – Escolha do melhor *Imbalance Ratio* para simulação



Fonte: Imagem elaborada pelo autor.

Onde K é um conjunto de tamanho pré-definido, definimos $\#K$ modelos cada um com um *imbalance ratio* diferente de K. O modelo final escolhido será aquele que otimize o *f1-score* durante a validação cruzada.

Este processo é necessário devido ao fato de não ser necessariamente verdade o fato de que treinar um modelo em uma base balanceada melhore sua performance. Na figura 16 ajustamos um modelo de Regressão Logística utilizando a técnica *SMOTE* com diferentes proporções para a base resultante. Como resultado, obtemos que o algoritmo com melhor desempenho[1] ainda é ajustado em uma base desbalanceada, mesmo que em menor proporção que a base inicial.

Esta metodologia é aplicada apenas para as técnicas de *random sampling*. Destacamos que o mesmo poderia ser utilizado para as extensões de *boosting* por usarem essas técnicas em sua composição, mas devido a limitações nas bibliotecas optamos por não realizá-los.

[1] Melhor seguindo a lógica de aumento do *f1-score*.

CAPÍTULO

4

SIMULAÇÃO EM BASES DE DADOS SIMULADOS

4.1 Simulação

Para analisar o comportamento das técnicas de *random sampling* e as extensões das técnicas *ensemble*, simulamos uma base de dados com 10.000 observações e apenas 1% de representatividade para o evento de interesse. A base possui 20 covariáveis contínuas com metade sendo informativas e a outra metade combinação linear das demais. Além disso, atribuímos aleatoriamente 1% das instâncias da base à variável resposta com objetivo de introduzir ruído na base e dificultar o problema de classificação.

4.2 Resultados

As tabelas 4, 5 e 6 mostram o desempenho dos modelos clássicos, com aplicação de técnicas de *random sampling* e extensões de técnicas *ensemble*, respectivamente. Estas podem ser encontradas completas no GitHub do autor.

A escolha da métrica adequada para avaliação de modelos está intrinsecamente ligada ao problema em questão e, geralmente, não ocorre de forma isolada. Aspectos como estabilidade das covariáveis, nível de significância e outras características específicas também são considerados durante esse processo. No entanto, para simplificação, neste trabalho, priorizamos a análise do desempenho dos modelos com base na validação cruzada.

Os resultados obtidos evidenciam padrões relevantes. A aplicação de técnicas de *random oversampling* resultou em um aumento significativo no desempenho dos modelos, especialmente em métricas como *F1-Score*, ROC AUC e KS, quando aplicadas aos modelos *Random Forest* e *XGBoost*. Observamos, ainda, que na maior parte dos casos os modelos com melhores perfor-

mances continuaram sendo ajustados em bases desbalanceadas, embora com menor desequilíbrio em relação à base original.

No contexto das extensões de técnicas *ensemble*, verificou-se que o impacto mais expressivo ocorreu em relação à métrica KS nos modelos derivados de extensões da técnica de *Bagging*, enquanto para as demais métricas os resultados foram similares.

Assim, o uso de técnicas de *random sampling* para este caso sugere uma relevância para aplicações em casos de *credit scoring*, onde a métrica KS desempenha papel central na avaliação da capacidade do modelo em separar as duas classes. Além disso, foi possível observar melhorias significativas no desempenho dos modelos em casos específicos, com destaque para a técnica *BorderlineSMOTE-2*, que proporcionou incremento em todas as métricas avaliadas, incluindo o F1-Score, sem a necessidade de comprometer o equilíbrio entre precisão e *recall*.

Tabela 4 – Desempenho dos modelos clássicos em bases simuladas.

Algoritmo	Precisão	Recall	F1-Score	ROC AUC	KS
XGBoost	9,56%	64,82%	16,35%	80,06%	54,16%
RandomForest	7,69%	61,27%	13,51%	78,64%	49,56%
LogisticRegression	8,06%	62,66%	14,10%	76,12%	50,70%
AdaBoost	5,50%	62,71%	9,94%	72,62%	43,94%
Bagging	11,64%	46,50%	17,52%	70,98%	39,51%
DecisionTree	23,15%	23,23%	23,18%	61,06%	22,12%

Tabela 5 – Desempenho nos modelos clássicos com técnicas de *Random sampling* em bases Simuladas. Apenas os 10 modelos com melhor ROC AUC.

Algoritmo	Técnica de Sampling	Precisão	Recall	F1-Score	ROC AUC	KS	Imbalance Ratio
XGBoost	BorderlineSMOTE-2	13,27%	67,70%	20,04%	81,67%	56,84%	27,38%
RandomForest	BorderlineSMOTE-1	10,77%	66,92%	18,50%	81,41%	58,79%	89,63%
XGBoost	BorderlineSMOTE-1	17,36%	59,16%	25,64%	80,87%	53,25%	53,31%
XGBoost	RandomOverSampler	12,04%	61,29%	19,34%	80,84%	53,03%	84,44%
RandomForest	RandomOverSampler	10,81%	61,21%	18,19%	80,50%	53,19%	6,63%
RandomForest	SMOTEEN	8,50%	64,01%	14,69%	80,33%	52,83%	22,19%
XGBoost	SMOTE	10,76%	56,96%	18,00%	80,29%	50,03%	89,63%
XGBoost	ADASYN	12,41%	61,86%	18,43%	80,27%	50,85%	32,56%
RandomForest	BorderlineSMOTE-2	16,02%	57,73%	24,98%	80,16%	53,22%	11,82%
XGBoost	ENN	7,78%	66,15%	13,70%	80,05%	53,40%	100,00%

Tabela 6 – Desempenho das extensões de técnicas *Ensemble* em bases simuladas.

Algoritmo	Precisão	Recall	F1-Score	ROC AUC	KS
SMOTEBagging	9,65%	59,84%	16,51%	78,32%	51,26%
OverBagging	6,55%	65,48%	11,75%	77,59%	49,30%
BalancedRandomForestClassifier	11,99%	59,19%	19,15%	76,67%	51,12%
BalancedBaggingClassifier	4,51%	64,04%	8,34%	75,72%	42,01%
SMOTEBoost	6,25%	61,20%	11,26%	75,35%	47,61%
OverBoost	7,40%	59,18%	12,82%	74,51%	44,73%
EasyEnsemble	6,56%	63,45%	11,43%	74,41%	45,71%
RUSBoost	4,81%	40,90%	8,53%	64,40%	27,80%

CAPÍTULO

5

SIMULAÇÃO EM BASES DE DADOS REAIS

5.1 Análise Exploratória

Com objetivo de analisar o comportamento das técnicas para problemas mais complexos, também realizamos o estudo em uma base de concessão de crédito real com covariáveis enriquecidas pelo produto *Outbox* da empresa *StepWise*. As covariáveis utilizadas foram selecionadas manualmente como sendo as mais relevantes para o problema em si devido a sua grande quantidade. A base de modelagem analisada possuía 18.167 observações e 5% de desbalanceamento, com um todo de 34 covariáveis, sendo elas 29 contínuas e 5 categóricas.

5.2 Resultados

As tabelas 7, 8 e 9 mostram o desempenho dos modelos clássicos, com aplicação de técnicas de *random sampling* e extensões de técnicas *ensemble*, respectivamente.

Tabela 7 – Desempenho dos modelos clássicos em um dados reais.

Algoritmo	Precisão	Recall	F1-Score	ROC AUC	KS
AdaBoost	8,28%	65,31%	14,67%	66,87%	26,73%
RandomForest	8,17%	63,01%	14,47%	65,23%	25,60%
LogisticRegression	8,08%	65,41%	14,19%	64,56%	23,49%
XGBoost	7,47%	59,84%	13,23%	63,54%	20,15%
Bagging	6,93%	52,47%	12,24%	58,81%	15,24%
DecisionTree	8,60%	10,65%	9,51%	52,31%	4,70%
EasyEnsemble	6,56%	63,45%	11,43%	74,41%	45,71%
RUSBoost	4,81%	40,90%	8,53%	64,40%	27,80%

No contexto do problema real analisado, as técnicas clássicas apresentaram desempenhos próximos entre si. Contudo, ao incorporar técnicas de *random sampling* no ajuste dos modelos, observou-se um leve incremento no desempenho das técnicas *Random Forest* e *AdaBoost* em

métricas como *F1-Score*, ROC AUC e KS. De maneira semelhante, as extensões de técnicas *ensemble* demonstraram uma performance ligeiramente superior em comparação às suas versões clássicas.

Embora o aumento de desempenho obtido não seja expressivo, ele é significativo o suficiente para destacar a utilização de técnicas de amostragem como um potencial hiperparâmetro de ajuste nos modelos. Assim como a taxa de aprendizagem desempenha um papel crucial nas técnicas de *boosting* ou a profundidade das árvores influencia o desempenho de *Random Forests* e são otimizadas durante o ajuste dos modelos, as estratégias de *random sampling* podem ser exploradas para otimizar os resultados em cenários específicos.

Tabela 8 – Desempenho dos modelos clássicos com técnicas de *random sampling* em dados reais. Apenas os 10 modelos com melhor ROC AUC.

Algoritmo	Técnica de Sampling	Precisão	Recall	F1-Score	ROC AUC	KS	Imbalance Ratio
RandomForest	SMOTEEN	9,46%	59,49%	16,29%	68,32%	29,02%	75,07%
RandomForest	RandomUnderSampler	8,85%	61,36%	15,44%	67,79%	27,84%	80,06%
AdaBoost	ENN	8,68%	62,78%	15,24%	67,79%	27,84%	100,00%
RandomForest	BorderlineSMOTE-1	9,36%	56,09%	16,00%	67,67%	27,15%	100,00%
AdaBoost	RandomOverSampler	8,95%	62,57%	15,62%	67,64%	28,55%	90,03%
AdaBoost	BorderlineSMOTE-1	8,70%	62,44%	15,12%	67,51%	26,68%	55,13%
RandomForest	SMOTETomek	9,43%	56,31%	16,09%	67,37%	27,38%	75,07%
AdaBoost	TomekLinks	8,60%	64,64%	15,10%	67,33%	27,42%	100,00%
AdaBoost	BorderlineSMOTE-2	8,14%	66,07%	14,44%	67,18%	26,29%	35,19%
RandomForest	SMOTE	9,48%	56,64%	16,07%	67,10%	26,83%	65,10%

Tabela 9 – Desempenho das extensões de técnicas *ensemble* em dados reais.

Algoritmo	Precisão	Recall	F1-Score	ROC AUC	KS
EasyEnsemble	9,2%	62,2%	16,0%	68,9%	29,4%
OverBoost	8,8%	66,0%	15,4%	68,6%	28,6%
BalancedRandomForestClassifier	8,7%	62,3%	15,3%	68,3%	27,9%
SMOTEBagging	8,1%	60,9%	14,2%	65,8%	24,0%
SMOTEBoost	7,7%	71,2%	13,8%	66,3%	25,2%
OverBagging	8,0%	49,3%	13,7%	62,1%	18,8%
RUSBoost	7,4%	64,8%	13,2%	63,1%	21,5%
BalancedBaggingClassifier	7,3%	63,3%	13,0%	63,2%	20,1%

CAPÍTULO

6

CONSIDERAÇÕES FINAIS

A predição de eventos raros é um problema cuja resolução muitas vezes acontece através da premissa de que os modelos terão melhor performance quando a base estiver balanceada com as mais diversas técnicas sendo inventadas para melhorar a performance de algoritmos quando o desbalanceamento é presente. Nesta dissertação, apresentamos diferentes métodos de *random sampling* para ajuste de diversos modelos clássicos. Contextualizamos quando o desbalanceamento passa a ser um problema e afeta a performance de certos algoritmos e algumas soluções para lidar com esses casos.

Além disso, as técnicas são de fácil implementação e podem ser ajustadas como um possível hiperparâmetro para os modelos clássicos, como a taxa de aprendizagem ou a profundidade das árvores acontecem em modelos de *boosting*. Em particular observamos um certo destaque para extensões da técnica *SMOTE*, que resolvem alguns dos possíveis problemas da técnica e ainda demonstraram aumento do desempenho dos modelos.

Há ainda certos refinamentos que poderiam ser realizados para este projeto, dentre eles uma maior variabilidade do tipo de covariáveis nas bases de simulação, uma vez que utilizamos apenas covariáveis contínuas. A performance para bases com desbalanceamento maior e menor do que 1% também poderia ser encaixada para entender o aumento de desempenho para diferentes níveis de *imbalance ratio*. Por fim, destacamos a possibilidade de utilização de funções de ligação assimétricas para o modelo de regressão (BAZÁN; ROMEO; RODRIGUES, 2014), as quais não foram utilizadas devido ao escopo do projeto.

Por fim, este trabalho realiza um resumo das principais técnicas de *random sampling* disponíveis para predição de eventos raros nas principais bibliotecas de ciência de dados na linguagem *python* e diversos detalhes sobre sua utilização. Este trabalho surge para guiar pesquisadores e cientistas em algumas das principais ferramentas. Destacamos ainda sobre a infinidade de técnicas existentes e uma aparente falta de comparação entre os métodos. Trazemos possíveis *insights* sobre os métodos mais relevantes para esse contexto e que podem aumentar o

desempenho dos modelos clássicos.

REFERÊNCIAS

BATISTA, G. E.; PRATI, R. C.; MONARD, M. C. A study of the behavior of several methods for balancing machine learning training data. **ACM SIGKDD explorations newsletter**, ACM New York, NY, USA, v. 6, n. 1, p. 20–29, 2004. Citado na página 25.

BAZÁN, J. L.; ROMEO, J. S.; RODRIGUES, J. Bayesian skew-probit regression for binary response data. 2014. Citado nas páginas 26 e 61.

BEYAN, C.; FISHER, R. Classifying imbalanced data sets using similarity based hierarchical decomposition. **Pattern Recognition**, Elsevier, v. 48, n. 5, p. 1653–1672, 2015. Citado na página 22.

BREIMAN, L. Bagging predictors. **Machine learning**, Springer, v. 24, p. 123–140, 1996. Citado nas páginas 23, 27 e 28.

BUCKLAND, M.; GEY, F. The relationship between recall and precision. **Journal of the American society for information science**, Wiley Online Library, v. 45, n. 1, p. 12–19, 1994. Citado na página 43.

CHAWLA, N. V. Data mining for imbalanced datasets: An overview. **Data mining and knowledge discovery handbook**, Springer, p. 875–886, 2009. Citado nas páginas 25 e 43.

CHAWLA, N. V.; BOWYER, K. W.; HALL, L. O.; KEGELMEYER, W. P. Smote: synthetic minority over-sampling technique. **Journal of artificial intelligence research**, v. 16, p. 321–357, 2002. Citado nas páginas 22 e 37.

CHAWLA, N. V.; LAZAREVIC, A.; HALL, L. O.; BOWYER, K. W. Smoteboost: Improving prediction of the minority class in boosting. In: SPRINGER. **Knowledge Discovery in Databases: PKDD 2003: 7th European Conference on Principles and Practice of Knowledge Discovery in Databases, Cavtat-Dubrovnik, Croatia, September 22-26, 2003. Proceedings 7**. [S.l.], 2003. p. 107–119. Citado nas páginas 23 e 42.

FERNÁNDEZ, A.; GARCÍA, S.; GALAR, M.; PRATI, R. C.; KRAWCZYK, B.; HERRERA, F. **Learning from imbalanced data sets**. [S.l.]: Springer, 2018. v. 10. Citado nas páginas 21, 25, 32 e 33.

FREUND, Y.; SCHAPIRE, R.; ABE, N. A short introduction to boosting. **Journal-Japanese Society For Artificial Intelligence**, JAPANESE SOC ARTIFICIAL INTELL, v. 14, n. 771-780, p. 1612, 1999. Citado na página 28.

FRIEDMAN, J. H. Greedy function approximation: a gradient boosting machine. **Annals of statistics**, JSTOR, p. 1189–1232, 2001. Citado na página 28.

____. Stochastic gradient boosting. **Computational statistics & data analysis**, Elsevier, v. 38, n. 4, p. 367–378, 2002. Citado na página 23.

GUZELLA, T. S.; CAMINHAS, W. M. A review of machine learning approaches to spam filtering. **Expert Systems with Applications**, Elsevier, v. 36, n. 7, p. 10206–10222, 2009. Citado na página 21.

HAIXIANG, G.; YIJING, L.; SHANG, J.; MINGYUN, G.; YUANYUE, H.; BING, G. Learning from class-imbalanced data: Review of methods and applications. **Expert systems with applications**, Elsevier, v. 73, p. 220–239, 2017. Citado nas páginas 22 e 25.

HAN, H.; WANG, W.-Y.; MAO, B.-H. Borderline-smote: a new over-sampling method in imbalanced data sets learning. In: SPRINGER. **Advances in Intelligent Computing: International Conference on Intelligent Computing, ICIC 2005, Hefei, China, August 23-26, 2005, Proceedings, Part I 1**. [S.l.], 2005. p. 878–887. Citado nas páginas 38 e 39.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. H.; FRIEDMAN, J. H. **The elements of statistical learning: data mining, inference, and prediction**. [S.l.]: Springer, 2009. v. 2. Citado nas páginas 27, 28 e 42.

HE, H.; BAI, Y.; GARCIA, E. A.; LI, S. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In: IEEE. **2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)**. [S.l.], 2008. p. 1322–1328. Citado nas páginas 37 e 38.

HE, H.; GARCIA, E. A. Learning from imbalanced data. **IEEE Transactions on knowledge and data engineering**, Ieee, v. 21, n. 9, p. 1263–1284, 2009. Citado na página 25.

IVAN, T. Two modifications of cnn. **IEEE transactions on Systems, Man and Communications, SMC**, v. 6, p. 769–772, 1976. Citado na página 34.

LAST, F.; DOUZAS, G.; BACAO, F. Oversampling for imbalanced learning based on k-means and smote. 11 2017. Citado na página 38.

LEMONTE, A. J.; BAZÁN, J. L. New links for binary regression: an application to coca cultivation in peru. **Test**, Springer, v. 27, p. 597–617, 2018. Citado na página 26.

LIU, X.-Y.; WU, J.; ZHOU, Z.-H. Exploratory undersampling for class-imbalance learning. **IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)**, IEEE, v. 39, n. 2, p. 539–550, 2008. Citado na página 41.

LÓPEZ, V.; FERNÁNDEZ, A.; GARCÍA, S.; PALADE, V.; HERRERA, F. An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. **Information sciences**, Elsevier, v. 250, p. 113–141, 2013. Citado nas páginas 21 e 42.

LOYOLA-GONZÁLEZ, O.; MARTÍNEZ-TRINIDAD, J. F.; CARRASCO-OCHOA, J. A.; GARCÍA-BORROTO, M. Study of the impact of resampling methods for contrast pattern based classifiers in imbalanced databases. **Neurocomputing**, Elsevier, v. 175, p. 935–947, 2016. Citado na página 22.

MANI, I.; ZHANG, I. knn approach to unbalanced data distributions: a case study involving information extraction. In: ICML. **Proceedings of workshop on learning from imbalanced datasets**. [S.l.], 2003. v. 126, p. 1–7. Citado na página 33.

NASSER, M.; YUSOF, U. K. Deep learning based methods for breast cancer diagnosis: A systematic review and future direction. **Diagnostics**, Multidisciplinary Digital Publishing Institute, v. 13, n. 1, p. 161, 2023. Citado na página 21.

SEIFFERT, C.; KHOSHGOFTAAR, T. M.; HULSE, J. V.; NAPOLITANO, A. Rusboost: A hybrid approach to alleviating class imbalance. **IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans**, IEEE, v. 40, n. 1, p. 185–197, 2009. Citado na página 42.

TAHA, A. A.; MALEBARY, S. J. An intelligent approach to credit card fraud detection using an optimized light gradient boosting machine. **IEEE Access**, IEEE, v. 8, p. 25579–25587, 2020. Citado na página 21.

WEISS, G. M.; MCCARTHY, K.; ZABAR, B. Cost-sensitive learning vs. sampling: Which is best for handling unbalanced classes with unequal error costs? **Dmin**, v. 7, n. 35-41, p. 24, 2007. Citado na página 23.

WILSON, D. L. Asymptotic properties of nearest neighbor rules using edited data. **IEEE Transactions on Systems, Man, and Cybernetics**, SMC-2, n. 3, p. 408–421, 1972. Citado na página 35.

YANG, P.; SHAN, S.; GAO, W.; LI, S. Z.; ZHANG, D. Face recognition using ada-boosted gabor features. In: IEEE. **Sixth IEEE International Conference on Automatic Face and Gesture Recognition, 2004. Proceedings.** [S.l.], 2004. p. 356–361. Citado na página 21.

YU, H.; NI, J.; DAN, Y.; XU, S. Mining and integrating reliable decision rules for imbalanced cancer gene expression data sets. **Tsinghua Science and technology**, TUP, v. 17, n. 6, p. 666–673, 2012. Citado na página 21.

ZHU, J.; ROSSET, S.; ZOU, H.; HASTIE, T. Multi-class adaboost. **Statistics and its interface**, v. 2, 02 2006. Citado na página 29.


ICMC
USP
SÃO CARLOS