Universidade Federal de São Carlos
Centro de Ciências Exatas e de Tecnologia
Departamento de Estatística

Um estudo sobre diferentes métodos de classificação para dados extremamente desbalanceados sob a ótica do aprendizado sensível ao custo

João Pedro Modesto Angélico

Trabalho de Conclusão de Curso

Universidade Federal de São Carlos
Centro de Ciências Exatas e de Tecnologia
Departamento de Estatística

Um estudo sobre diferentes métodos de classificação para dados extremamente desbalanceados sob a ótica do aprendizado sensível ao custo

João Pedro Modesto Angélico
Orientador(a): Prof. Dr. Carlos Alberto Ribeiro Diniz

Trabalho de Conclusão de Curso apresentado como parte dos requisitos para obtenção do título de Bacharel em Estatística.

São Carlos
Fevereiro de 2025

Sumário

Capítulo 1

Introdução

Com o contínuo aumento na disponibilidade de dados em sistemas de grande escala, complexos e em rede, como vigilância, segurança, Internet e finanças, torna-se crucial avançar na compreensão fundamental do tratamento e análise de dados a partir de informações não processadas, a fim de apoiar os processos de tomada de decisão. Embora as técnicas estabelecidas de engenharia de dados tenham demonstrado grande êxito em diversas aplicações reais, o desafio de aprender a partir de dados desbalanceados é uma questão que tem atraído cada vez mais atenção tanto da comunidade acadêmica quanto da indústria (He e Garcia, 2009).

O desbalanceamento em um conjunto de dados é um problema comum em modelagem preditiva, principalmente em aplicações em cenários reais. Ele ocorre quando a distribuição de observações entre as classes não é igual. Essa distribuição pode variar desde um leve viés até um desbalanceamento severo. É comum descrever o desbalanceamento em um conjunto de dados por meio de porcentagens entre as classes. Por exemplo, um problema de classificação multiclasse pode ter 80% das observações em uma classe, 18% na segunda classe e 2% na última. O desbalanceamento pode afetar negativamente o desempenho dos modelos de aprendizado de máquina por tenderem a se concentrar na predição da classe majoritária, ignorando a minoritária (Brownlee, 2020, 4-5).

O aprendizado em cenários desbalanceados refere-se ao desafio enfrentado por modelos estatísticos ao lidar com dados onde uma classe está significativamente sub-representada, especialmente em casos de desbalanceamento extremo, como quando a classe minoritária representa apenas 1% das observações. Devido à natureza complexa de conjuntos de dados desbalanceados, foram desenvolvidas técnicas e abordagens específicas para possibilitar sua utilização em modelos preditivos e estatísticos. Este trabalho abordará algumas dessas metodologias, incluindo a **regressão logística**, **florestas aleatórias** e **boosting**, adaptadas para incorporar variações

baseadas no custo, promovendo maior eficiência na classificação.

Adicionalmente, o conceito de custo desempenha um papel fundamental na modelagem com dados desbalanceados. Em contextos onde a penalidade de um erro varia de acordo com a classe, a incorporação de custos diferenciais nos algoritmos de aprendizado torna-se essencial para refletir a importância relativa dos erros de classificação. Por exemplo, em situações em que a classe minoritária representa apenas uma pequena fração dos dados, é comum atribuir um peso consideravelmente maior aos erros associados a essa classe, de forma a compensar sua baixa representatividade e minimizar impactos críticos, como diagnósticos médicos equivocados ou falhas na detecção de fraudes. Essa abordagem, denominada aprendizado sensível ao custo, integra os custos diretamente na função de perda, influenciando a atualização dos coeficientes durante o treinamento e promovendo um equilíbrio entre sensibilidade e especificidade.

A abordagem de classificação sensível ao custo tem sido amplamente estudada para enfrentar os desafios impostos pelo desbalanceamento de classes, situação em que a distribuição desigual dos dados pode prejudicar o desempenho dos modelos tradicionais. Estudos fundamentais, como os de Breiman *et al.* (1984), estabeleceram as bases para a construção de árvores de decisão e métodos *ensemble*, os quais foram aprimorados para corrigir o viés em cenários com classes desproporcionais. Em problemas onde a classe minoritária é de maior relevância prática, Japkowicz (2000) destaca a necessidade de tratar de forma diferenciada os erros de classificação, enquanto Kubat e Matwin (1997) propôs técnicas de *one-sided sampling* para mitigar esse viés. Além disso, as pesquisas de Zadrozny e Elkan (2001a,b) demonstram que a incorporação de custos na função de perda, através do ajuste de pesos e calibração das probabilidades, melhora a capacidade dos modelos em identificar corretamente a classe menos representada. Complementarmente, Drummond e Holte (2000) investigou a sensibilidade dos critérios de divisão nas árvores aos custos, e Weiss e Provost (2001) mostrou que a distribuição das classes influencia diretamente o desempenho dos classificadores. Esse conjunto de estudos fornece um embasamento teórico robusto que justifica a adaptação dos modelos preditivos para refletirem as reais implicações dos erros, sobretudo em cenários de desbalanceamento severo.

Este TCC iniciou-se com uma introdução que contextualizou o problema do desbalanceamento em aprendizado de máquina e apresentou a relevância de abordagens sensíveis ao custo. No Capítulo 2, analisaram-se as causas desse desbalanceamento e os fundamentos do aprendizado sensível ao custo, incluindo revisão de métodos clássicos e contemporâneos. O Capítulo 3 descreveu em detalhes os modelos relevantes para esse trabalho — Regressão Logística, Florestas Aleatória e XGBoost — e apresentou as principais métricas de avaliação, da matriz de

confusão à curva ROC, além de técnicas de validação cruzada. No Capítulo 4, aplicaram-se esses métodos a um conjunto real de transações de cartão de crédito, comparando cenários com e sem ponderação de custos e demonstrando ganhos expressivos em sensibilidade para detecção de fraudes. Por fim, a conclusão sintetizou as contribuições principais, discutiu limitações e sugeriu direções futuras, consolidando a eficácia do aprendizado sensível ao custo em cenários desbalanceados.

Capítulo 2

Desbalanceamento e sensibilidade ao custo

O desbalanceamento de classes é um problema recorrente e desafiador na modelagem preditiva, especialmente em contextos onde as proporções entre as categorias da variável-alvo são extremamente discrepantes (Megahed *et al.*, 2021). Situações desse tipo são encontradas em diversas áreas, como na detecção de fraudes financeiras (Araujo, 2024), diagnóstico de doenças raras e identificação de falhas em sistemas industriais (Zheng e Jin, 2020). A principal dificuldade está no fato de que a classe minoritária é sub-representada. Isso pode levar os modelos a apresentar um desempenho insatisfatório na identificação dessa classe, concentrando-se quase exclusivamente na classe majoritária.

Os impactos do desbalanceamento tornam-se mais evidentes à medida que aumenta a disparidade entre as classes. Em cenários de alto desbalanceamento, modelos treinados sem técnicas específicas para lidar com esse problema podem apresentar métricas de avaliação aparentemente boas, como alta acurácia, mas, na prática, falham em detectar adequadamente os casos da classe minoritária. Essa questão é especialmente crítica quando o foco da análise está em prever eventos raros, como falências ou complicações médicas, nos quais a falha em identificar esses eventos pode ter consequências graves.

O desbalanceamento pode ser descrito de diferentes maneiras, como pela proporção entre as classes ou em termos percentuais. Por exemplo, em um conjunto de dados com desbalanceamento 1:100, para cada observação da classe minoritária, existem 100 da classe majoritária. Outra forma de descrever o problema é considerar cenários de múltiplas classes, onde algumas categorias podem representar a maior parte das observações, enquanto outras possuem representatividade marginal.

Neste capítulo, serão discutidas as possíveis causas que levam ao desbalanceamento nos conjuntos de dados e os desafios que ele apresenta à modelagem preditiva . Também será abordada

a necessidade de métodos alternativos para avaliar o desempenho dos modelos, considerando que as métricas tradicionais, como a acurácia, podem ser enganosas em cenários desbalanceados.

2.1 Possíveis causas de desbalanceamento

A existência de desbalanceamento em conjuntos de dados pode originar-se de duas fontes principais: amostragem e natureza intrínseca dos dados. Em casos nos quais as observações estão desproporcionalmente distribuídas, a causa pode residir na coleta dos dados, evidenciando-se especialmente quando a amostragem é enviesada ou quando ocorrem erros durante esse processo.

Por exemplo, a coleta de observações de uma região específica ou de um período limitado pode resultar em uma distribuição desigual entre as classes. Além disso, erros na atribuição de rótulos de classe durante a classificação podem contribuir para o desbalanceamento. Em situações em que o desequilíbrio é ocasionado por viés de amostragem ou erros de medição, métodos aprimorados de amostragem e correção de erros podem ser aplicados para reequilibrar o conjunto de dados.

Por fim, o desbalanceamento pode ser inerente ao domínio de interesse, sendo uma característica natural dos dados. Isso pode ocorrer quando o processo gerador de observações em uma classe demanda mais recursos, como tempo, custo, computação, entre outros. Exemplos na área médica (diagnóstico de doenças raras), ciências biológicas (detecção de espécies ameaçadas) ou do mercado financeiro (fraudes), tendem a observar fenômenos que ocorrem em menor frequência, gerando dados naturalmente com desbalanços de classe severos.

Diante desses desafios, diversas estratégias podem ser empregadas para lidar com o desbalanceamento, como reamostragem, técnicas de *ensemble* e modelagem sensível ao custo.

2.2 Aprendizado sensível ao custo

Em geral, algoritmos de aprendizado de máquina tomam todos os erros de classificação com uma mesma importância, porém nem todos os erros são iguais, especialmente em problemas de classificação desbalanceados onde a classe minoritária é a mais relevante para análise, porém é pouco classificada. Pode-se ilustrar esse argumento com alguns exemplos reais como diagnósticos médicos (Sterner, 2024) ou identificação de fraudes em cartões de crédito.

O conceito de Aprendizado Sensível ao Custo (*Cost-Sensitive Learning*) incorpora custos

associados a diferentes tipos de erros durante o treinamento do modelo. Isso implica que, em vez de considerar todos os erros igualmente, o modelo é otimizado para minimizar o custo total das previsões incorretas, considerando as consequências assimétricas dos falsos positivos e falsos negativos.

A relação entre o Aprendizado Sensível ao Custo e dados desbalanceados é intrínseca: técnicas como ajuste de pesos de classe, meta-algoritmos como *MetaCost* (Domingos, 1999), e modificações em funções de custo de modelos são amplamente utilizadas para mitigar vieses em conjuntos desbalanceados. Essas abordagens são particularmente críticas em domínios como saúde, onde o custo de um falso negativo (e.g., falha em diagnosticar uma doença) supera em ordens de magnitude o custo de um falso positivo (Araf *et al.*, 2024).

2.2.1 Algumas definições em aprendizado sensível ao custo

Para lidar com cenários em que os erros de classificação têm impactos econômicos ou operacionais distintos, o aprendizado sensível ao custo formaliza o problema por meio de penalidades atribuídas a diferentes decisões. A seguir, apresentam-se algumas definições para custo, definidas por Turney (2002).

Custo de obtenção (Test-cost)

Em áreas como a de diagnóstico médico, cada exame ou atributo possui um custo associado à sua execução. Sempre que o custo de um teste excede o benefício da redução de erros, torna-se racional omitir determinadas avaliações; por outro lado, se o custo de erro for muito alto, todos os testes relevantes devem ser realizados.

Custo de ensino (Teacher-cost)

Algumas aplicações exigem supervisão especializada para rotular corretamente exemplos de treinamento. Nesses casos, incorpora-se um *teacher-cost*, representando o esforço ou recurso necessários para obter rótulos precisos. Estratégias de *active learning* podem minimizar esse custo selecionando apenas instâncias críticas para anotação.

Custo computacional

A complexidade de um classificador—em tempo de treinamento, tempo de inferência ou uso de memória—também pode ser modelada como um custo. Métodos sensíveis a esse critério

buscam equilibrar a acurácia preditiva com restrições de recursos computacionais.

Custo de classificação errônea

O tipo mais estudado em cost-sensitive learning, o *misclassification cost*, atribui penalidades assimétricas a falsos positivos e falsos negativos. Essa matriz de custo pode ser fixa (estacionária) ou depender de características específicas de cada instância, orientando o classificador a priorizar a minimização dos erros mais críticos.

Outros custos

Existem outras definições para custo, como *intervention costs* (ações corretivas após uma predição), *instability costs* (variabilidade do modelo), e *unwanted achievement costs* (resultados indesejados de predições corretas). No Capítulo 4 desse trabalho, será apresentada uma análise focada na minimização dos custos de classificação errônea, relevante para a detecção de eventos raros em bases desbalanceadas.

2.2.2 Matriz de custo estática e função de custo

Grande parte dos métodos de aprendizado sensível ao custo assume que, para um problema com M classes, existe uma matriz de custo $C \in \mathbb{R}^{M \times M}$ fixa durante todo o processo de treinamento e decisão. Cada elemento $C(i, j)$ corresponde ao custo de predizer a classe i quando a verdadeira é j.

Para o caso binário ($M = 2$), a matriz estática adota a forma da Tabela 2.1:

Tabela 2.1: Matriz de custo para duas classes

	Real negativo	**Real positivo**
Predito negativo	C_{00} (correto)	C_{01} (falso negativo)
Predito positivo	C_{10} (falso positivo)	C_{11} (correto)

Nessa configuração, os custos de *falso positivo* e *falso negativo* são, respectivamente, C_{10} e C_{01}. Para refletir a gravidade dos erros, espera-se que

$$C_{10} > C_{00} \quad \text{e} \quad C_{01} > C_{11},$$

assegurando penalização maior para classificações incorretas do que para as corretas (Elkan, 2001).

Os valores de $C(i, j)$ são tipicamente expressos em termos monetários ou operacionais e permanecem proporcionais se todos forem escalonados por um mesmo fator. Podem ainda assumir valores negativos para representar benefícios, exigindo análise cuidadosa na construção da matriz (Margineantu, 2009).

Dado um exemplo x, a predição ótima é aquela que minimiza o custo esperado:

$$L(x, y) = \sum_{j=1}^{n} P(j \mid x)\, C(i, j).$$

Assim, mesmo que uma classe seja a mais provável, a decisão pode recair sobre outra que reduza o custo total esperado, caracterizando a essência da tomada de decisão sensível ao custo (Elkan, 2001). .

2.2.3 Abordagens para Aprendizado Sensível ao Custo

O aprendizado sensível ao custo costuma ser dividido em três grandes categorias de técnicas: (1) modificação do conjunto de dados de treinamento (*data modification*), (2) modificação do algoritmo de aprendizagem (*algorithm modification*) e (3) pós-processamento da saída do modelo treinado (*output post-processing*). Cada uma dessas estratégias busca incorporar as diferenças de custo de erro nas diferentes classes de formas distintas, conforme ilustrado a seguir.

Modificação do conjunto de dados de treinamento

Nesta abordagem de *pré-processamento*, altera-se a distribuição das classes no conjunto de treino para refletir os custos associados. Isso pode ser feito, por exemplo, por *oversampling* ou *undersampling*: replica-se instâncias (ou gera-se exemplos sintéticos) da classe minoritária de maior custo, ou remove-se exemplos da classe majoritária, equilibrando assim a proporção efetiva de exemplos (Wang, 2013).

Alternativamente, usa-se *ponderação de instâncias*: atribui-se pesos maiores às amostras cuja classe incorreta implicaria custo elevado. Esses pesos fazem com que algoritmos de aprendizado padrão (que podem aceitar pesos, como árvores de decisão ou modelos bayesianos) tratem casos mais custosos com maior importância durante o treinamento.

Neste trabalho, por exemplo, parâmetros como `class_weight` (em bibliotecas como scikit-learn) e `scale_pos_weight` (em modelos XGBoost) realizam essa ponderação, equivalente a reequilibrar o conjunto de dados de treino.

Modificação do algoritmo de aprendizagem

A segunda estratégia consiste em incorporar custos diretamente no processo de modelagem (*in-processing*). Nesse caso, altera-se a função de treinamento ou os critérios internos do algoritmo. Por exemplo, em árvores de decisão pode-se modificar o critério de divisão para levar em conta os custos de erro de cada classe, em vez de usar somente a impureza padrão. De maneira geral, diversos algoritmos clássicos têm versões sensíveis a custos: Naive Bayes, SVM e redes neurais podem incluir pesos na função de perda; algoritmos de decisão podem modificar suas heurísticas internas. No trabalho atual, as configurações de `class_weight` na regressão logística e `scale_pos_weight` no XGBoost representam exatamente essa modificação algorítmica, pois ajustam o critério de otimização do modelo para priorizar corretamente a classe minoritária de maior custo.

Pós-processamento da saída do modelo

Nesta categoria, ajusta-se a decisão final do modelo após o treinamento, sem alterar nem os dados nem o algoritmo. Comumente, usa-se a probabilidade predita pelo modelo para definir um novo limiar de classificação que minimize o custo esperado. Por exemplo, dada uma estimativa de probabilidade para cada classe, escolhe-se o ponto de corte que equilibra falsos positivos e falsos negativos segundo seus custos relativos (técnica conhecida como *threshold-moving*). Elkan (2001) recomenda manter o modelo aprendido e depois computar explicitamente o limiar ótimo usando as probabilidades. Além disso, métodos de avaliação como as *cost curves* de Drummond e Holte (2006) permitem visualizar o desempenho do classificador ao longo de toda a gama de possíveis limiares e distribuições de custos. Essas ferramentas auxiliam na escolha do limiar mais adequado ao contexto de custos do problema. Note que, diferentemente dos parâmetros ajustados em tempo de treinamento (como `class_weight`), o pós-processamento atua apenas sobre a saída do modelo já treinado, realocando sua fronteira de decisão de acordo com as penalidades desejadas.

2.3 Classificação desbalanceada sensível ao custo

A classificação desbalanceada sensível ao custo é uma abordagem que busca resolver os desafios associados à modelagem preditiva em cenários onde as classes apresentam distribuições extremamente desiguais. Esses desafios decorrem do fato de que, em problemas desbalanceados, modelos de aprendizado de máquina tendem a privilegiar a classe majoritária, resultando em

desempenho inferior na classe minoritária.

O foco dessa abordagem está em reconhecer que os erros de classificação têm impactos assimétricos, dependendo das classes envolvidas. Enquanto um erro ao classificar uma instância da classe majoritária pode ter consequências mínimas, um erro na classe minoritária pode acarretar implicações graves, como perdas financeiras significativas ou diagnósticos médicos errados. Para lidar com isso, modelos sensíveis ao custo atribuem penalidades diferenciadas aos erros, priorizando a minimização dos mais críticos.

A implementação dessa estratégia envolve a inclusão explícita de custos no processo de treinamento, seja por ajustes na função de perda ou por ponderação de classes. Por exemplo, em tarefas onde a classe minoritária representa 1% dos dados, atribui-se um peso cem vezes maior aos seus erros. Essa adaptação força o modelo a reequilibrar internamente o viés dos dados.

Essa abordagem apresenta vantagens notáveis. Primeiramente, ela possibilita que os modelos se adaptem a diferentes contextos e priorizem o que realmente importa para o problema em questão. Além disso, ao incluir os custos diretamente no treinamento, evita-se a necessidade de balancear o conjunto de dados por meio de técnicas como *oversampling* ou *undersampling*, que podem introduzir outros desafios, como *overfitting* ou perda de variabilidade. Entretanto, a definição precisa dos custos pode ser uma tarefa complexa, especialmente em contextos onde as consequências dos erros não são facilmente quantificáveis.

A classificação sensível ao custo é amplamente aplicável em problemas desbalanceados que demandam alta sensibilidade para identificar casos da classe minoritária. Por exemplo, na área da saúde, erros em diagnósticos de doenças graves devem ser minimizados, mesmo que isso implique um pequeno aumento nos falsos positivos. Da mesma forma, na detecção de fraudes, é aceitável investigar um número maior de transações legítimas para garantir que fraudes reais não sejam ignoradas.

2.4 Estudos relacionados

Diversos estudos foram realizados para lidar com o problema dos conjuntos de dados desbalanceados.Costa e Nascimento (2016) realizaram um estudo que utiliza modelos de aprendizado de máquina para prever a falha de um componente específico do Sistema de Pressão de Ar. Nele foram avaliados quatro algoritmos de aprendizado de máquina: *K-Nearest Neighbors* (KNN), Regressão Logística (LR), Máquina de Vetores de Suporte (SVM), Árvore de Decisão

(*DT*) e Floresta Aleatória (*RF*). Para lidar com o conjunto desbalanceado, empregou-se uma abordagem de aprendizado sensível ao custo, ajustando os pesos do *SVM* e da *LR* de forma inversamente proporcional à fração de casos da respectiva classe. No caso do *RF* e do *KNN*, o limiar de classificação foi ajustado empiricamente, aumentando-o para melhorar o desempenho, tendo o *RF* alcançado os melhores resultados, com índices de 3,74% de Falsos Positivos e 3,7% de Falsos Negativos.

Altinger *et al.* (2017) focaram em uma abordagem de reamostragem (*oversampling* e *undersampling*) para a previsão de falhas em *software* automotivo utilizando conjuntos de dados altamente desbalanceados, analisando diferentes algoritmos de classificação com o F1-Score e a G-Measure como métricas de desempenho. Observou-se que o *undersampling* apresentou bons resultados apenas com *SVM* utilizando *kernel* de função de base radial, enquanto o *oversampling* demonstrou eficácia para *Naive Bayes* (*NB*), AdaBoost M1 com *NB* e o *SVM* do tipo *divide and conquer*. Os autores também concluíram que o XGBoost mostrou melhorias com *oversampling* maciço, enquanto o desempenho do *SVM* diminuiu com altos níveis de *oversampling*. Quanto à previsibilidade por amostragem, os algoritmos de *boosting* (XGBoost, AdaBoost M1) revelaram-se os mais imprevisíveis, sendo o *NB* o preditor mais estável, alcançando os melhores resultados tanto com *undersampling* quanto com *oversampling*.

Pereira *et al.* (2021) analisaram o tratamento de dados altamente desbalanceados a partir de oito conjuntos de dados de uma empresa de manufatura automotiva, considerando quatro algoritmos de aprendizado de máquina (RF, dois métodos de AutoML e AutoEncoder) e cinco técnicas de balanceamento, em uma comparação de desempenho realizada em duas etapas. Na primeira etapa, todos os algoritmos foram analisados considerando dois produtos e três estratégias de balanceamento: Nenhuma, Synthetic Minority Oversampling Technique (SMOTE) e Gaussian Copula (GC), com o RF alcançando os melhores resultados em termos de classificação geral e esforço computacional. Na segunda fase, os conjuntos de dados foram avaliados com cinco métodos de balanceamento: Nenhum, SMOTE, GC, Random Undersampling, aprendizado sensível ao custo e ajuste de limiar no processo de liberação de lotes associado a Tomek Links para uma análise mais aprofundada do RF. Nesse cenário, os autores obtiveram os melhores resultados ao combinar o RF com Gaussian Copula, atingindo uma Área Sob a Curva (AUC) média de 67,31%.

Domingos (1999) propôs o MetaCost, um meta-algoritmo capaz de transformar qualquer classificador em sensível ao custo por meio de reclassificações estocásticas baseadas em custos esperados. Em testes com diversos conjuntos de dados do UCI, o MetaCost mostrou reduções

significativas no custo total de misclassificação, comprovando sua versatilidade e robustez.

Fan *et al.* (1999) desenvolveram o AdaCost, uma extensão do AdaBoost que incorpora explicitamente custos de misclassificação na fórmula de atualização de pesos durante o processo de boosting. Aplicações em detecção de fraudes e diagnósticos médicos revelaram que o AdaCost consegue reduzir de forma consistente o custo cumulativo em comparação ao AdaBoost clássico, sem aumentar a complexidade computacional.

Zhou e Liu investigaram o uso de redes neurais sensíveis ao custo, comparando estratégias de reamostragem (oversampling e undersampling), threshold-moving e ensembles suaves. Seus experimentos em 21 bases de dados do UCI evidenciaram que o threshold-moving e ensembles de redes neurais alcançam o melhor equilíbrio entre sensibilidade e especificidade, enquanto métodos puramente baseados em amostragem podem comprometer a generalização em cenários multiclasses (Zhou e Liu, 2006).

Drummond e Holte introduziram as cost curves, uma técnica de visualização e comparação de classificadores que varia custos de falso positivo e falso negativo simultaneamente. As cost curves permitem avaliar o desempenho esperado sob diferentes distribuições de classes e custos, oferecendo insights mais completos que as tradicionais curvas ROC, especialmente em aplicações onde os custos operacionais flutuam (Drummond e Holte, 2006).

Capítulo 3

Modelos e métricas

Aqui serão discutidos os modelos e métricas que serão utilizados na análise de dados no Capítulo 4.

3.1 Regressão logística sensível ao custo

Em muitos conjuntos de dados do mundo real, como detecção de fraudes, diagnósticos médicos ou falhas industriais, as classes da variável resposta são altamente desbalanceadas. Por exemplo, casos de fraude podem representar apenas 1% das transações. Nesses contextos, a regressão logística tradicional trata todos os erros da mesma forma, o que frequentemente leva a um modelo que acerta quase sempre na classe majoritária, mas erra nas classes minoritárias, que geralmente são as mais importantes. Para contornar esse problema, utiliza-se a regressão logística sensível ao custo (cost-sensitive logistic regression). Essa abordagem atribui pesos diferentes aos erros de cada classe. Assim, os erros cometidos sobre a classe minoritária, ou de maior interesse, têm maior influência na função de custo do modelo, forçando o algoritmo a prestar mais atenção a esses casos. O efeito prático dessa modificação é que o modelo deixa de minimizar apenas a média simples dos erros (em que cada observação tem o mesmo peso), e passa a minimizar um erro ponderado, em que cada classe contribui para o treinamento de forma proporcional à sua importância. Em outras palavras, erros em classes mais importantes (geralmente as menos frequentes) pesam mais na função de perda, orientando o modelo a equilibrar melhor suas previsões, ou, ainda com mais detalhes: Na regressão logística tradicional, o procedimento de estimação dos parâmetros é baseado na maximização da verossimilhança (ou, equivalentemente, na minimização da log-verossimilhança negativa). Nessa formulação, todas as observações contribuem igualmente para a função objetivo, independentemente da classe à

qual pertencem. Isso pode ser problemático em conjuntos de dados desbalanceados, pois o modelo tende a favorecer a classe majoritária, este procedimento também é conhecido como "média simples dos erros": o erro de uma observação da classe 0 vale o mesmo que o erro de uma da classe 1. Se a base de dados estiver desbalanceada (por exemplo, 95% de classe 0 e 5% de classe 1), o modelo pode aprender a prever tudo como classe 0 e ainda assim parecer "bom", já que ele acerta a maioria, mas ignora a minoria, que geralmente é a mais importante. Uma solução usual é considerar custos. Ao tornar o modelo sensível ao custo, você atribui um peso maior aos erros da classe minoritária. Com isso, o modelo não está mais minimizando o erro médio simples, mas sim um erro que leva em conta a importância (peso) de cada classe. Isso é o que chamamos de erro ponderado.

3.1.1 Formulação matemática com pesos por classe

Na regressão logística tradicional, os parâmetros $\beta \in \mathbb{R}^p$ e $\beta_0 \in \mathbb{R}$ são estimados por máxima verossimilhança. Dado um conjunto de observações $\{(x_i, y_i)\}_{i=1}^n$, com $x_i \in \mathbb{R}^p$ e $y_i \in \{0, 1\}$, assume-se que as variáveis Y_i são independentes e seguem distribuições Bernoulli com parâmetro p_i, dado por

$$p_i = P(Y_i = 1 \mid x_i) = \frac{1}{1 + \exp\big(-(\beta_0 + x_i^\top \beta)\big)}.$$

O logaritmo da função de verossimilhança correspondente é

$$\ell(\beta, \beta_0) = \sum_{i=1}^{n} \big[y_i \log(p_i) + (1 - y_i) \log(1 - p_i)\big].$$

A estimação por máxima verossimilhança consiste em encontrar os valores dos parâmetros que maximizam $\ell(\beta, \beta_0)$. Alternativamente, pode-se minimizar a log-verossimilhança negativa

$$J(\beta, \beta_0) = -\frac{1}{n}\, \ell(\beta, \beta_0),$$

frequentemente chamada de *log-loss*.

3.1.2 Modelo logístico ponderado por classe

Em muitas aplicações práticas, como detecção de fraudes ou diagnóstico de doenças raras, os erros cometidos pelo modelo não têm todos o mesmo impacto. Errar uma fraude pode ser bem mais grave do que errar uma transação legítima, por exemplo. Para lidar com esse tipo

de situação, podemos adaptar a regressão logística tradicional para que ela dê mais atenção às classes mais importantes.

Uma forma simples e eficaz de fazer isso é atribuir um peso w_j para cada classe $j \in \{0, 1\}$. Esses pesos funcionam como um ajuste fino na função de verossimilhança: observações da classe com maior peso passam a ter mais influência no processo de estimação dos parâmetros. Ou seja, o modelo aprende a se preocupar mais com os casos que realmente importam. Dado o modelo logístico com

$$p_i = P(Y_i = 1 \mid x_i) = \frac{1}{1 + \exp\big(-(\beta_0 + x_i^\top \beta)\big)},$$

a log-verossimilhança ponderada torna-se

$$\ell_{\text{ponderada}}(\beta, \beta_0) = \sum_{i=1}^{n} w_{y_i} \left[y_i \log(p_i) + (1 - y_i) \log(1 - p_i) \right],$$

em que

$$w_{y_i} = \begin{cases} w_1 & \text{se } y_i = 1, \\ w_0 & \text{se } y_i = 0. \end{cases}$$

O uso dos pesos w_0 e w_1 modifica diretamente a forma como o modelo "aprende". Em vez de tratar todas as observações da mesma maneira, a função objetivo passa a considerar o peso de cada classe. Isso faz com que os erros cometidos em observações de uma classe mais importante (com maior peso) tenham um impacto maior na função que o modelo está tentando otimizar. Na prática, isso significa que queremos maximizar uma versão ponderada da log-verossimilhança. Equivalente a isso, podemos pensar em minimizar a log-verossimilhança negativa ponderada, dada por

$$J_{\text{ponderada}}(\beta, \beta_0) = -\frac{1}{n} \sum_{i=1}^{n} w_{y_i} \left[y_i \log(p_i) + (1 - y_i) \log(1 - p_i) \right],$$

em que os pesos w_{y_i} são definidos de acordo com a classe da observação:

$$w_{y_i} = \begin{cases} w_1 & \text{se } y_i = 1, \\ w_0 & \text{se } y_i = 0. \end{cases}$$

Com isso, o modelo passa a dar mais atenção, e mais peso, para os erros nas observações que realmente interessam no contexto da aplicação.

3.1.3 Como escolher os pesos?

Existem duas formas comuns de definir os pesos w_0 e w_1, definição manual e automática (balanceamento).

- **Definição manual:** quando você tem conhecimento prévio sobre o problema e sabe, por exemplo, que errar uma observação da classe 1 (como uma fraude) é muito mais grave do que errar uma da classe 0. Nesse caso, você pode definir os pesos manualmente, como no exemplo abaixo:

  ```
  class_weight = {0: 1, 1: 10}
  ```

  Assim, o modelo será incentivado a detectar com mais cuidado as observações da classe 1.

- **Definição automática (balanceamento):** caso não haja um conhecimento específico sobre os custos, uma boa estratégia é deixar que o próprio modelo balanceie os pesos com base na frequência das classes. No scikit-learn, isso pode ser feito com:

  ```
  class_weight = 'balanced'
  ```

  Nesse caso, o peso da classe j é calculado como:

  $$w_j = \frac{n}{2\,n_j},$$

  onde n é o número total de observações e n_j é o número de observações na classe j. Esse tipo de balanceamento ajuda o modelo a lidar com desbalanceamentos severos nos dados.

3.1.4 Pesos de Classe e regularização: dois mecanismos complementares

Na regressão logística implementada pelo scikit-learn, dois elementos importantes podem ser combinados para melhorar o desempenho do modelo em situações reais: a ponderação por classe e a regularização. Embora ambos afetem o processo de estimação, eles atuam de forma separada na função de perda e têm propósitos distintos.

Tornando o modelo sensível às classes: `class_weight`

O argumento `class_weight` permite atribuir pesos diferentes a cada classe da variável resposta, de modo que o modelo preste mais atenção aos casos que realmente importam. Isso significa que erros cometidos na classe com maior peso afetam mais o processo de otimização. Dessa forma, o modelo é ajustado para priorizar o acerto das classes de maior interesse, mesmo que sejam minoritárias.

Evitando Overfitting: Regularização L2

A regularização é outro componente essencial do modelo. Ela entra como um termo extra na função de perda, penalizando soluções com coeficientes muito grandes. No caso da regularização L2 (ou Ridge), adicionamos à função objetivo:

$$\frac{\lambda}{2}\|\beta\|_2^2,$$

onde $\lambda = 1/C$ é o parâmetro de regularização controlado pelo usuário (via o argumento `C`). Com isso, a função de perda completa torna-se:

$$J_{\text{total}}(\beta, \beta_0) = J_{\text{ponderada}}(\beta, \beta_0) + \frac{\lambda}{2}\|\beta\|_2^2.$$

Esse segundo termo não tem relação direta com as classes, mas atua restringindo o tamanho dos coeficientes, o que ajuda a reduzir a variância do modelo e aumentar sua generalização.

Conclusão

Ao combinar esses dois mecanismos — pesos de classe e regularização — conseguimos construir modelos que não apenas aprendem a distinguir padrões relevantes em contextos desbalanceados, mas também evitam ajustes excessivos aos dados de treinamento. São ferramentas complementares que ajudam a melhorar tanto a robustez quanto a sensibilidade do modelo em aplicações práticas.

3.2 Floresta aleatória sensível ao custo

A Floresta Aleatória (*Random Forest*) consiste na aplicação do *bagging* utilizando árvores de decisão como classificadores individuais, porém com uma importante extensão: além de

reamostrar os registros, existe também a reamostragem das variáveis, o que resulta em uma grande diversidade no processo.

O algoritmo *random forest* é formalmente apresentado abaixo:

1. Tire uma subamostra A através de *bootstrap* com reposição dos indivíduos.
2. Para a primeira divisão, amostre $p < P$ variáveis aleatoriamente sem reposição.
3. Para cada uma das variáveis amostradas $X_{j(1)}, X_{j(2)}, \ldots, X_{j(p)}$, aplique o algoritmo de divisão:
   (a) Para cada valor $s_{j(k)}$ de $X_{j(k)}$:
      i. Divida os registros da subamostra A com $X_{j(k)} < s_{j(k)}$ como uma repartição, e os registros restantes, $X_{j(k)} \geq s_{j(k)}$, como outra repartição.
      ii. Meça a homogeneidade de classes dentro de cada repartição de A definida no passo anterior.
   (b) Selecione o valor de $s_{j(k)}$ que produza a máxima homogeneidade de classes dentro da subamostra.
4. Selecione a variável $X_{j(k)}$ e o valor de divisão $s_{j(k)}$ que produzam a máxima homogeneidade de classes dentro da repartição.
5. Prossiga à próxima divisão e repita os passos anteriores, começando com o Passo 2.
6. Continue com divisões adicionais seguindo o mesmo procedimento até que a árvore tenha crescido.
7. Volte ao Passo 1, tire outra subamostra *bootstrap*, e recomece todo o processo.

3.2.1 Tornando o modelo sensível ao custo dos erros

Quando as classes são desbalanceadas, a Floresta Aleatória tende a favorecer a classe majoritária, pois maximiza a acurácia global. Para tornar o modelo sensível ao custo dos erros, introduzimos pesos de classe que penalizam diferentemente cada tipo de erro. Operacionalmente, isso significa multiplicar o peso de cada amostra pelo valor associado à sua classe durante o treinamento das árvores. Observações da classe minoritária (ou de maior custo) recebem peso maior, de modo que erros nessas observações influenciam mais a função objetivo. Como resultado, a árvore é induzida a criar divisões que melhorem a separação da classe minoritária,

mitigando o viés em favor da classe majoritária. Em essência, em vez de minimizar o erro simples médio, minimizamos uma perda ponderada que reflete a importância relativa de cada classe.

3.2.2 Formulação matemática com pesos por classe

Nas árvores de decisão (e, por consequência, na Floresta Aleatória), a seleção de um split (divisão) em um nó é feita otimizando um critério de impureza, como o índice de Gini. No cenário ponderado por classe, cada classe i recebe um peso w_i, aplicado às observações dessa classe. Se tivermos um nó com n_i amostras da classe i, o Gini ponderado neste nó é calculado usando as frequências efetivas ponderadas $w_i n_i$. Por exemplo, para duas classes a fórmula do índice de Gini ponderado fica:

$$Gini_{ponderado} = 1 - \sum_{i=1}^{2} \left(\frac{w_i n_i}{\sum_{j=1}^{2} w_j n_j} \right)^2 .$$

Ou seja, primeiro soma-se o peso total de cada classe, e depois calcula-se a impureza da proporção ponderada de cada classe. Essa expressão indica que, ao escolher um ponto de corte, as proporções calculadas para cada nó filho são afetadas pelos pesos. Quanto maior o peso de uma classe, maior a variação do índice de Gini quando o split separa essa classe de outras. Essa formulação (usada em cada nó de cada árvore) faz com que o algoritmo privilegie divisões que melhorem a pureza dos nós em relação à classe de maior peso. Em outras palavras, o critério de divisão torna-se sensível aos pesos: um nó com muitos exemplos de classe majoritária mas com peso baixo pode ter impureza efetiva pequena, enquanto um nó pequeno da classe minoritária com peso alto contribui significativamente para o ganho de informação. Essa adaptação matemática garante que o processo de construção das árvores da Floresta Aleatória incorpore a ponderação de classes diretamente no critério de divisão.

3.2.3 Modelo de floresta aleatória ponderado por classe

A Floresta Aleatória ponderada por classe é simplesmente o uso de árvores de decisão treinadas com a consideração de pesos de classe, combinadas em um `ensemble`. Na prática, em implementações como o scikit-learn, isso é feito passando o parâmetro `class_weight` ao construtor de `RandomForestClassifier`. Por exemplo, definindo `class_weight='balanced'`, cada classe i recebe peso proporcional a $\frac{N}{K n_i}$ (de forma similar ao modelo de regressão logística, discutido anteriormente). Durante o treino, esses pesos são convertidos em pesos de amostra

passados a cada árvore, alterando a forma como os critérios de divisão (Gini, entropia, etc.) são calculados em cada nó. Além de `balanced`, o scikit-learn oferece `balanced_subsample`, que calcula esses pesos separadamente em cada amostra bootstrap usada para gerar cada árvore. Assim, mesmo em diferentes amostras de treinamento internas, a árvore continua sensível ao custo das classes. No conjunto, o modelo resultante é uma média (voto majoritário ou probabilidade média) dos classificadores ponderados, refletindo a ênfase desejada nas classes de maior custo.

Tornando o modelo sensível às classes: `class_weight`

De forma análoga ao que foi discutido na seção de regressão logística, o parâmetro `class_weight` permite tornar o modelo sensível às classes, assim podendo penalizar mais a função de custo global, forçando cada árvore a tentar acertar melhor essas classes. Em outras palavras, essa estratégia atua diretamente no critério de divisão das árvores: divisões que isolam exemplos da classe ponderada negativamente (mais importante nesse exemplo) geram grande redução de impureza efetiva e são preferidas. Dessa forma, a Floresta Aleatória resultante tende a oferecer maior sensibilidade (recall) para a classe de interesse, ainda que isso possa reduzir um pouco a acurácia global ou precisão na classe majoritária. Note que o `class_weight` funciona multiplicativamente sobre `sample_weight`: portanto, se for fornecido um vetor de pesos de instância, ele será combinado com os pesos de classe especificados.

Evitando overfitting: estratégias de controle da complexidade em floresta aleatória

Além da ponderação de classes, a Floresta Aleatória incorpora mecanismos próprios de regularização que controlam sua complexidade e ajudam a evitar overfitting. Esses mecanismos são parâmetros de hiperparâmetro que limitam o crescimento das árvores individuais ou ajustam a diversidade do `ensemble`. Entre os principais, temos:

- **Número de árvores (`n_estimators`):**quantidade de árvores no `ensemble`. Mais árvores reduzem a variância do modelo (tornando-o mais estável) mas aumentam o custo computacional. Um número muito baixo pode não representar bem o conjunto, enquanto um número muito alto tende a saturar o ganho em desempenho.
- **Profundidade máxima (`max_depth`):**limita o tamanho de cada árvore. Árvores muito profundas podem memorizar ruído; ao restringir a profundidade, reduz-se a complexidade

do modelo. Se `max_depth=None`, as árvores crescem até que cada folha seja pura ou atinja `min_samples_split`.

- **Número mínimo de amostras em folha ou em split (`min_samples_leaf`, `min_samples_split`):** controlam o tamanho mínimo de um nó. Aumentar esses valores evita divisões que criariam folhas com poucos exemplos, suavizando a árvore e prevenindo que padrões espúrios no treinamento dominem a modelagem.

- **Número de atributos por divisão (`max_features`):** define quantas variáveis são consideradas aleatoriamente em cada tentativa de split.Valores menores de `max_features` aumentam a aleatoriedade (tornando as árvores menos correlacionadas), o que costuma melhorar a generalização do `ensemble`. O padrão `'sqrt'` (raiz do total de atributos) já introduz essa regularização inerente.

- **Bootstrap e amostragem de instâncias:** Por padrão, cada árvore é treinada em uma amostra bootstrap dos dados, o que por si só é uma forma de regularização (`bagging`). Há ainda o parâmetro `max_samples` que restringe o tamanho dessa amostra de cada árvore.

A justaposição de `class_weight` com esses hiperparâmetros garante que o modelo não apenas foque nas classes minoritárias, mas também mantenha boa capacidade de generalização. Em geral, ajusta-se `n_estimators`, `max_depth`, `max_features` etc., preferencialmente via validação cruzada, para balancear entre reduzir viés e controlar a variância do modelo. Esse ajuste conjunto é essencial para obter um bom desempenho final em dados desbalanceados.

Conclusão

O uso de Florestas Aleatórias sensíveis ao custo combina o poder do `ensemble` com a flexibilidade de ponderar classes. A ponderação por classe atua diretamente nas divisões das árvores, via critério de impureza ajustado, garantindo atenção aos casos minoritários ou de alto custo. Paralelamente, parâmetros de regularização naturais à Floresta Aleatória (número de árvores, profundidade máxima, número de atributos por split etc.) controlam sua complexidade e evitam overfitting. Dessa forma, essas duas abordagens — pesos de classe e ajustes de hiperparâmetros — são complementares: juntas constroem modelos que aprendem a distinguir os padrões relevantes mesmo em cenários desbalanceados, mantendo robustez e capacidade de generalização. Em implementações como o scikit-learn, tudo isso pode ser feito de forma integrada através de parâmetros como `class_weight` e os hiperparâmetros de floresta, per-

mitindo ao usuário adaptar o modelo às necessidades específicas do problema (por exemplo, maximizando a sensibilidade da classe minoritária)

3.3 Extreme Gradient Boosting - XGBoost

O XGBoost é um método de *ensemble* que combina características do *boosting* com técnicas de amostragem similares às utilizadas em métodos de *bagging*, como os encontrados em Florestas Aleatórias. Em sua essência, o algoritmo constrói árvores de decisão de forma sequencial, onde cada nova árvore é ajustada para corrigir os erros das anteriores, utilizando, para isso, uma amostragem dos dados que lembra o processo de *bagging*.

3.3.1 Formulação matemática com pesos por classe

Formalmente, seja $\ell(y_i, \hat{y}_i)$ a perda de log-verossimilhança no ponto x_i com rótulo y_i e predição $\hat{y}_i$ no modelo atual (por exemplo, para classificação binária logística, $\ell_i = -\big[y_i \log(p_i) + (1-y_i)\log(1-p_i)\big]$ com $p_i = \frac{1}{1+e^{-\hat{y}_i}}$.) Em um modelo ponderado, cada exemplo i recebe um peso w_i (por classe), e a função objetivo regularizada do XGBoost para a árvore f_t no passo t torna-se:

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} w_i \, \ell\big(y_i,\ \hat{y}_i^{(t-1)} + f_t(x_i)\big) \ + \ \Omega(f_t),$$

onde $\Omega(f)$ é o termo de regularização que penaliza a complexidade das árvores (incluindo termos L1/L2 e custo por folha).Aplicando expansão em série de Taylor de segunda ordem à perda, definimos o gradiente $g_i = \frac{\partial \ell}{\partial \hat{y}_i}$ e o hessiano $h_i = \frac{\partial^2 \ell}{\partial \hat{y}_i^2}$. Com pesos, cada gradiente e hessiano é multiplicado pelo peso w_i, da respectiva instância, isto é $g_i \leftarrow w_i \, g_i$, e $h_i \leftarrow w_i \, h_i$. No caso da perda logística binária, isso equivale a $g_i = w_i \, (p_i - y_i)$, e $h_i = w_i \, p_i(1-p_i)$.

Ao agrupar os termos por folha da árvore, seja I_j o conjunto de índices que caem na folha j. Somando gradientes e hessianos ponderados dentro de cada folha, definimos $G_j = \sum_{i \in I_j} g_i$, e $H_j = \sum_{i \in I_j} h_i$.

Considerando a regularização L2 de intensidade λ e o custo fixo por folha γ, a parte do objetivo referente à folha j é

$$G_j \, w_j \ + \ \tfrac{1}{2} \, (H_j + \lambda) \, w_j^2 \ + \ \gamma,$$

onde w_j é o peso (valor predito) atribuído pela folha j. O valor ótimo w_j^* que minimiza essa

função quadrática é

$$w_j^* = -\frac{G_j}{H_j + \lambda}.$$

Esse resultado mostra que a decisão final de cada folha (e, consequentemente, a divisão da árvore) é moldada pelos somatórios dos gradientes e hessianos ponderados pelos pesos de classe. Assim, ao alterar w_i, alteramos diretamente a contribuição de cada instância na função objetivo, tornando o modelo sensível aos custos definidos.

3.3.2 Modelo XGBoost ponderado por classe

Na prática, a biblioteca `xgboost` (Python) expõe o parâmetro `scale_pos_weight` na classe `XGBClassifier` para controlar os pesos das classes em problemas binários. Esse parâmetro define um fator α aplicado a todas as instâncias positivas na função de perda (equivalente a atribuir $w_i = \alpha$ para $y_i = 1$ e $w_i = 1$ para $y_i = 0$). Em código:

```
from xgboost import XGBClassifier
model = XGBClassifier(scale_pos_weight=ratio)
```

Aqui `ratio` pode ser, por exemplo, o número de instâncias negativas dividido pelo número de positivas. Internamente, isso faz com que os gradientes associados às instâncias positivas sejam multiplicados por `ratio`, aumentando a correção do modelo nesses casos. Como resultado, durante o treinamento o XGBoost prioriza a minimização do erro na classe minoritária.

Para problemas multiclasse (ou quando se deseja controle mais fino), pode-se fornecer explicitamente um vetor de pesos por amostra ao chamar `model.fit(X, y, sample_weight=pesos)`. Ou seja, atribui-se manualmente um peso w_i para cada observação (por classe, se desejado). Em suma, `scale_pos_weight` é recomendado para classificação binária; em multiclasses, usa-se tipicamente o argumento `sample_weight` no método `fit`, definindo pesos equivalentes para cada classe.

3.3.3 Evitando overfitting: regularização no XGBoost

Além dos pesos de classe, o XGBoost dispõe de vários mecanismos de regularização para conter a complexidade das árvores. O termo $\Omega(f)$ inclui L2 nos pesos das folhas (`reg_lambda`) e L1 (`reg_alpha`).A regularização L2 penaliza pesos muito grandes (fazendo o modelo mais conservador), enquanto a L1 tende a zerar pesos de pouca importância.Outros hiperparâmetros também ajudam a prevenir overfitting: o parâmetro `gamma` (ou `min_split_loss`) exige um

ganho mínimo para qualquer divisão de nó;valores maiores de `gamma` tornam o algoritmo mais conservador, impedindo splits sem melhoria substancial. O limite de `max_depth` controla a profundidade máxima das árvores;árvores muito profundas ajustam-se demais ao ruído dos dados. O parâmetro `min_child_weight` exige um valor mínimo da soma de hessianos em cada folha para que ocorra uma divisão adicional,evitando folhas muito "puras" (com poucos exemplos). Em conjunto, esses mecanismos garantem que, mesmo com pesos de classe elevados, o modelo não cresça de forma descontrolada. Em síntese, os pesos de classe focam o aprendizado nos erros mais importantes, enquanto a regularização L1/L2 e os limites de estrutura evitam que o modelo sofra overfitting.

Conclusão

Em suma, o XGBoost pode ser adaptado a cenários de custo assimétrico atribuindo-se pesos às classes. O parâmetro `scale_pos_weight` provoca a modificação da função de perda ao multiplicar os gradientes das instâncias de cada classe pelos fatores especificados, tornando o modelo sensível às diferenças de frequência ou custo. A formulação matemática que incorpora esses pesos altera diretamente o treinamento das árvores: os somatórios de gradientes/hessianos (agora ponderados) determinam os valores das folhas e as divisões escolhidas. Ao mesmo tempo, é fundamental usar as técnicas de regularização disponibilizadas (L1, L2, `gamma`, limites de profundidade etc.) para controlar a complexidade do modelo e evitar que o foco na classe minoritária leve a sobreajuste.Dessa forma, XGBoost sensibilizado ao custo combina ajustes no objetivo de erro com penalidades estruturais, resultando em classificadores mais equilibrados em bases desbalanceadas.

3.3.4 Medidas de desempenho

Algumas medidas são importantes para avaliar o desempenho do modelo. Tradicionalmente, a precisão é a mais comumente utilizada. Porém, para classificação de classes desbalanceadas, em que há uma classe minoritária, por exemplo, fraudes, e uma classe predominante, não fraudes, a precisão não é mais uma medida interessante. Isso porque a classe minoritária tem muito pouco impacto sobre a precisão em comparação com a classe predominante (Horta, 2010).

3.3.5 Matriz de confusão

Em um modelo com variável resposta binária, como ocorre no caso de um Credit Scoring, busca-se classificar os clientes em uma das categorias consideradas, ou seja, em não fraudes ou fraudes, e obter um bom grau de acerto nessas classificações.

Geralmente, nas amostras de validação, conhece-se a resposta dos clientes em relação à sua real condição de crédito. Estabelecendo critérios em que se classifiquem esses clientes em não fraudes e fraudes, torna-se possível comparar essa classificação obtida com a verdadeira condição creditícia dos clientes. Essa comparação é feita através da matriz de confusão ou matriz de erros. Essa matriz descreve uma tabulação cruzada entre a classificação predita através de um único ponto de corte e a condição real e conhecida de cada indivíduo, onde os valores da diagonal principal compreendem as classificações corretas e os valores fora dessa diagonal correspondem aos erros de classificação. A Tabela 3.1 ilustra uma matriz de confusão.

Tabela 3.1: Matriz de confusão

	fraude (Predito)	**não fraude (Predito)**	**Total**
fraude (Real)	a	b	$a+b$
não fraude (Real)	c	d	$c+d$
Total	$a+c$	$b+d$	$a+b+c+d$

Os elementos de entrada dessa matriz têm os seguintes significados:

- a: quantidade de clientes classificados como fraudes dado que eles são fraudes.
- b: quantidade de clientes classificados como não fraudes dado que os clientes são fraudes.
- c: quantidade de clientes classificados como fraudes dado que eles são não fraudes.
- d: quantidade de clientes classificados como não fraudes dado que eles são não fraudes.

Algumas medidas comumente utilizadas em problemas de classificação binária, obtidas a partir de uma matriz de confusão, são:

- **Acurácia (AC)**: é a proporção do total de predições corretas.

$$AC = \frac{a+d}{a+b+c+d}$$

A acurácia é enganosa em cenários desbalanceados. Por exemplo, em um dataset com proporção 1:100 entre classes, um modelo que classifica todos os exemplos como majoritários

alcança 99% de acurácia, ignorando completamente a classe minoritária (Brownlee, 2020, 48-55). Esse fenômeno, conhecido como *paradoxo da acurácia*, evidencia a necessidade de métricas complementares como F1-score e AUC-ROC para avaliação justa.

- **Taxa de positivos-corretos (PC)** ou sensibilidade ou **Recall**: é a proporção de clientes classificados pelo modelo como não fraudes dado que eles são não fraudes.

$$PC = \frac{d}{c+d}$$

- **Taxa de falsos positivos (FP)**: é a proporção de clientes classificados pelo modelo como não fraudes dado que eles são fraudes.

$$FP = \frac{b}{a+b}$$

- **Taxa de negativos corretos (NC)** ou especificidade: é a proporção de clientes classificados pelo modelo como fraudes dado que eles são fraudes.

$$NC = \frac{a}{a+b}$$

- **Taxa de falsos-negativos (FN)**: é a proporção de clientes classificados pelo modelo como fraudes dado que eles são não fraudes.

$$FN = \frac{c}{c+d}$$

- **Precisão (P)**: representa a proporção de todas as classificações corretas realizadas pelo modelo em relação ao total de predições efetuadas. Diferentemente da precisão por classe (PV+ e PV-), que avalia acertos condicionados a uma classe específica, a precisão geral considera o desempenho global do classificador:

$$P = \frac{a+d}{a+b+c+d}$$

onde a e d correspondem às classificações corretas (verdadeiros negativos e verdadeiros positivos, respectivamente), e $a+b+c+d$ representa o total de classificações realizadas pelo modelo.

3.3.6 Precisão vs. Sensibilidade

Em problemas de classificação com dados desbalanceados, a escolha entre priorizar *precisão* ou *sensibilidade* depende dos custos associados aos erros de classificação. A precisão, definida como a proporção de verdadeiros positivos entre todas as predições positivas é crítica em contextos onde falsos positivos (FP) têm consequências custosas. Por outro lado, o sensibilidade, que mede a capacidade de identificar todos os casos relevantes é prioritário em aplicações como diagnóstico médico, onde falsos negativos (FN) podem ser catastróficos.

A relação entre essas métricas é intrinsecamente antagônica: otimizar uma geralmente degrada a outra. Por exemplo, em dados desbalanceados, aumentar o sensibilidade da classe minoritária frequentemente exige a redução do limiar de classificação, o que introduz mais falsos positivos e reduz a precisão (Brownlee, 2020, 63). Esse fenômeno é visualizado na *curva de precisão-sensibilidade*, que ilustra o compromisso (*trade-off*) entre as duas métricas para diferentes limiares.

Para superar essa dicotomia, métricas compostas, como o *F1-score*, combinam precisão e sensibilidade em um único valor harmônico:

$$F1 = 2 \times \frac{\text{Precisão} \times \text{sensibilidade}}{\text{Precisão} + \text{sensibilidade}}. \tag{3.1}$$

O F1-score é particularmente útil para avaliação holística em cenários desbalanceados, pois penaliza modelos que negligenciam uma das métricas He e Garcia (2009). No entanto, sua interpretação deve considerar a natureza do problema: em contextos onde uma métrica é mais relevante, variações como o *Fβ-score* permitem ponderar precisão e sensibilidade de forma assimétrica:

$$F_\beta = (1 + \beta^2) \times \frac{\text{Precisão} \times \text{sensibilidade}}{(\beta^2 \times \text{Precisão}) + \text{sensibilidade}}. \tag{3.2}$$

Aqui,β controla a importância relativa do sensibilidade em relação à precisão Han *et al.* (2012).

Modelos sensíveis ao custo integram esse equilíbrio diretamente em sua arquitetura. Técnicas como *weighted loss functions* ajustam os pesos das classes durante o treinamento, enquanto métodos pós-treinamento, como a otimização de limiares, recalibram a relação precisão-sensibilidade conforme a necessidade operacional Elkan (2001). Essas abordagens não apenas melhoram o desempenho na classe minoritária, mas também preservam a robustez estatística do modelo.

3.3.7 Curva ROC (Receiver Operating Characteristic)

A Curva ROC é uma ferramenta fundamental para avaliar a capacidade discriminativa de modelos, especialmente na análise de risco de crédito, baseando-se nos conceitos de sensibilidade e especificidade. Para sua construção, calcula-se a sensibilidade (taxa de verdadeiros positivos) e a especificidade (taxa de verdadeiros negativos) em diversos limiares de decisão. Ao se plotar a sensibilidade em função de $1 -$ especificidade, obtém-se a curva, a qual facilita a identificação do ponto de corte que melhor equilibra ambas as métricas, geralmente situado na região do "ombro"da curva.

Em problemas de classificação binária, é comum utilizar um limiar de 0.5. No contexto de análise de crédito, se a probabilidade estimada para uma empresa ou pessoa física ultrapassar esse valor, o sistema a classifica como não fraude, resultando na concessão de crédito; caso contrário, é considerada fraude, levando à recusa.

A área sob a curva ROC (AUC) quantifica a capacidade de discriminação do modelo: quanto maior a AUC, melhor o desempenho na distinção entre não fraudes e fraudes. Uma forma de interpretar essa área é através da estatística U de Mann-Whitney. Sejam n_1 o número de indivíduos fraudes ($Y = 1$) e n_0 o número de indivíduos não fraudes; formando-se $n_1 \times n_0$ pares, a AUC equivale à proporção de vezes em que o indivíduo fraude apresenta uma probabilidade maior que o não fraude.

Conforme descrito por Hosmer & Lemeshow (Hosmer e Lemeshow, 1989), as interpretações gerais são:

- Se AUC $= 0.5$: ausência de discriminação;
- Se $0.7 \leq$ AUC ≤ 0.8: discriminação aceitável;
- Se $0.8 \leq$ AUC ≤ 0.9: discriminação excelente;
- Se AUC ≥ 0.9: discriminação excepcional.

Modelos com elevado poder discriminativo se concentram no canto superior esquerdo da curva ROC, onde a sensibilidade é alta com mínima perda de especificidade. Por outro lado, modelos com desempenho inferior tendem a se aproximar da diagonal, que representa o resultado esperado de um modelo sem qualquer informação relevante sobre a inadimplência – como se as decisões fossem tomadas ao acaso (Carballo, 2002).

3.4 Validação cruzada

Durante o treinamento de modelos, é frequente ocorrer o fenômeno de overfitting, em que o modelo se ajusta demasiadamente aos dados de treinamento, comprometendo sua capacidade de generalização para novos conjuntos. Para detectar e evitar esse problema, além de comparar o desempenho dos modelos, emprega-se a técnica de validação cruzada.

A validação cruzada consiste em dividir a base de dados em duas partes: uma destinada ao treinamento e outra à avaliação do desempenho do modelo. Uma abordagem comum é o k-fold Cross Validation, na qual o conjunto de dados é particionado em k subconjuntos, denominados *folds*. Em cada iteração, um *fold* é reservado para teste, enquanto os $k - 1$ restantes são utilizados para treinar o modelo. Esse processo se repete k vezes, garantindo que cada *fold* seja usado uma vez como conjunto de teste. A Figura 3.1 exemplifica esse procedimento.

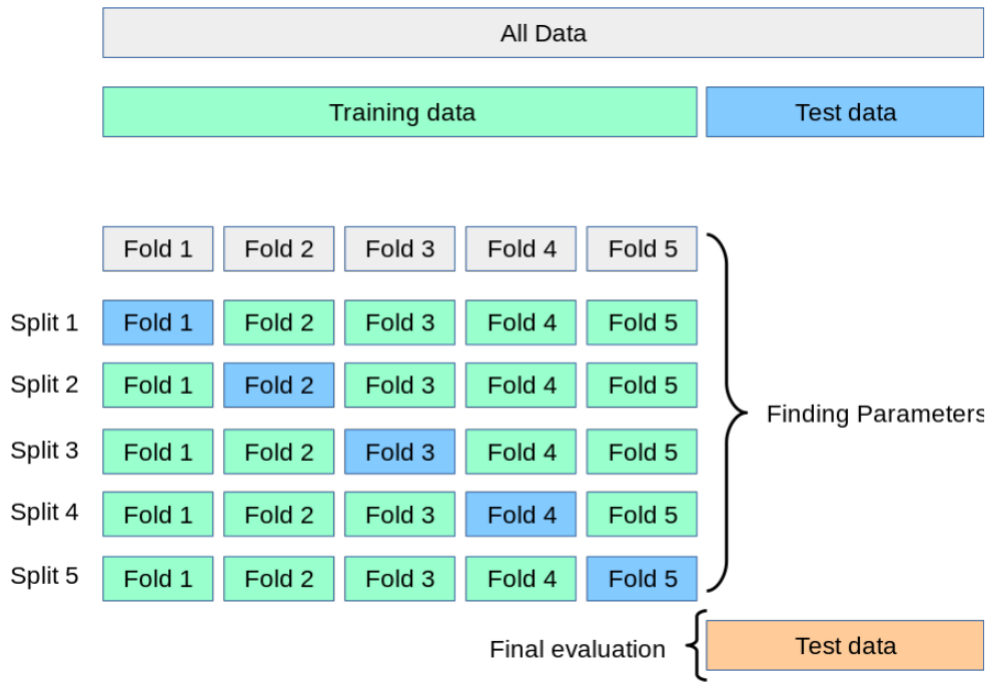

Figura 3.1: Exemplo visual da metodologia k-fold Cross Validation

Uma das principais vantagens dessa metodologia é a obtenção de múltiplas métricas de desempenho, permitindo o cálculo de estatísticas, como média e desvio-padrão, que fornecem uma visão robusta sobre o comportamento do modelo.

Em problemas com dados desbalanceados, é essencial ajustar o k-fold Cross Validation para que cada *fold* contenha uma proporção representativa das classes. Esse ajuste é realizado por meio da amostragem estratificada, que preserva a distribuição das classes em cada subconjunto.

Para a implementação prática dessa abordagem neste trabalho, foram utilizadas as classes

`RepeatedStratifiedKFold` e `cross_val_score` da biblioteca `scikit-learn` do Python.

3.5 Relatório de classificação

O relatório de classificação é uma ferramenta usada para avaliar o desempenho de modelos de classificação. Ele apresenta as principais métricas de desempenho por classe, fornecendo uma visão mais detalhada do comportamento do modelo do que a simples acurácia global. Essa visão por classe ajuda a identificar desbalanceamentos e pontos fracos do modelo. Na prática, o relatório é gerado a partir das previsões do modelo (após aplicar um limiar de decisão) e pode ser obtido facilmente com a função `classification_report` da biblioteca scikit-learn, que retorna um sumário textual dessas métricas.

- *Precisão por classe* (Precision_i) – é a proporção de clientes que realmente são não fraudes dado que eles foram preditos pelo modelo como não fraudes (PV+) e a proporção de clientes que realmente são fraudes dado que eles foram preditos pelo modelo como fraudes (PV-), sendo, respectivamente:

$$PV+ = \frac{d}{b+d}$$

$$PV- = \frac{a}{a+c}$$

- *Sensibilidade por classe* (Recall_i) – fração dos exemplos da classe i que foram corretamente identificados.

- *F_1-score por classe* ($F_{1,i}$) – média harmônica entre Precision_i e Recall_i,

$$F_{1,i} \;=\; 2\,\frac{\text{Precision}_i \;\times\; \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i}\,.$$

Além dos valores individuais por classe, o relatório pode incluir médias das métricas:

- *Média macro* – média aritmética simples das métricas por classe, tratando cada classe igualmente;

- *Média ponderada* – média das métricas, ponderada pelo suporte de cada classe, refletindo o desequilíbrio de frequência.

Na prática, esse sumário é obtido diretamente pela função `classification_report` da biblioteca `scikit-learn`, que organiza essas informações em formato tabular ou textual. Ao apresentar cada métrica por classe, o relatório de classificação permite identificar se o modelo está favorecendo alguma classe em particular, evidenciando assim padrões de viés ou fragilidades em cenários desbalanceados.

Em resumo, o relatório de classificação fornece uma visão abrangente dos acertos e erros do modelo em cada classe, complementando a matriz de confusão. Ele destaca, para cada classe, quão preciso e completo é o modelo ao prever cada categoria. Esse conjunto de métricas facilita a comparação entre modelos de classificação, permitindo escolher aqueles que apresentam valores mais elevados e balanceados de precisão, sensibilidade e F1-score.

Capítulo 4

Análise com banco de dados reais

4.1 Dados

O conjunto de dados utilizado refere-se a transações realizadas com cartões de crédito por titulares europeus durante o mês de setembro de 2013(Machine Learning Group - ULB, 2018), os códigos utilizados estão contidos em Modesto Angelico (2025). O objetivo principal desse conjunto é auxiliar na detecção de transações fraudulentas, permitindo que empresas de cartão de crédito evitem cobranças indevidas aos clientes por compras que não realizaram.

O dataset é composto por 284.807 transações, das quais 492 são identificadas como fraudulentas, representando aproximadamente 0,172% do total. Essa característica torna o conjunto de dados altamente desbalanceado, o que influencia diretamente a escolha das métricas de avaliação dos modelos preditivos.

Os dados fornecidos já vieram com um tratamento, foram transformados por meio de Análise de Componentes Principais (PCA), o motivo especificado em Machine Learning Group - ULB (2018) é a necessidade de manter a anonimidade das observações, resultando em variáveis numéricas identificadas como V1 até V28. Além dessas, há duas variáveis adicionais não transformadas:

- "Time": representa o tempo, em segundos, decorrido entre cada transação e a primeira transação registrada no dataset.
- "Amount": indica o valor monetário envolvido em cada transação, o que pode ser útil especialmente em abordagens de aprendizado sensíveis ao custo das transações.

Por fim, a variável de resposta "Class" indica se uma transação é fraudulenta (valor igual a 1) ou legítima (valor igual a 0).

4.2 Escolhendo os pesos

Devido ao forte desbalanceamento (0,172% de fraudes), foi necessário ajustar o critério de treinamento para que cada classe tivesse igual influência no processo de otimização. Adotamos duas abordagens:

1. **Parâmetro `class_weight="balanced"`**: em modelos da `scikit-learn` (Regressão Logística e Random Forest), usamos o modo `balanced`, que automaticamente atribui a cada classe o peso inversamente proporcional à sua frequência:

   $$w_i \;=\; \frac{N}{2 \times N_i}\,,$$

   onde $N = 284\,807$ é o total de observações, $N_0 = 284\,315$ o número de transações legítimas e $N_1 = 492$ o número de transações fraudulentas. Na prática isso resulta em

   $$w_0 \;=\; \frac{284\,807}{2 \times 284\,315} \;\approx\; 0.50 \quad \text{e} \quad w_1 \;=\; \frac{284\,807}{2 \times 492} \;\approx\; 289.43.$$

   Dessa forma, a classe minoritária (fraude) recebe peso aproximadamente 578 vezes maior que a classe majoritária, mas normalizado para que $w_0 + w_1 = 1$, equilibrando a importância de ambas no processo de otimização.

2. **Cálculo manual (`scale_pos_weight`)**: em modelos de `XGBoost`, configuramos explicitamente

   $$\texttt{scale\_pos\_weight} \;=\; \frac{N_0}{N_1} \;=\; \frac{284\,315}{492} \;\approx\; 578.5.$$

   Para garantir balanceamento equivalente a 50/50, converti este fator em pesos $w_0 = 0.50$ (classe não-fraude) e $w_1 = 289.43$ (classe fraude), de modo que

   $$\frac{w_1}{w_0} \;\approx\; \frac{N_0}{N_1} \;=\; 578.5 \quad \text{e} \quad w_0 + w_1 = 1.$$

   Assim, cada erro em fraude "custa" 578 vezes mais do que um erro em não-fraude, mas com a soma dos pesos ajustada para 1.

Dessa forma, todos os algoritmos trataram as duas classes com igual prioridade, garantindo que o modelo não "ignore" a minoria de fraudes durante o treinamento. Esses ajustes se mostraram fundamentais para aumentar drasticamente o *recall* na detecção de fraudes, sem comprometer excessivamente a precisão.

4.3 Cenários

Para avaliação do desempenho dos modelos, serão considerados dois cenários distintos:

- **Cenários base (sem custo):** Para o primeiro cenário adotamos a função de perda binária tradicional (0-1), que atribui o mesmo peso a todos os erros cometidos pelo modelo. Esta função é dada por:

$$L(y, \hat{y}) = \begin{cases} 0, & \text{se } y = \hat{y} \\ 1, & \text{se } y \neq \hat{y}, \end{cases}$$

  onde y representa o valor real da classe (0 para não fraude e 1 para fraude), e $\hat{y}$ é o valor predito pelo modelo.

  - (I) O primeiro cenário utilizará uma divisão dos dados em treino e teste, na proporção de 70% para treino e 30% para teste. Neste cenário serão avaliados os seguintes modelos: Regressão logística, Floresta aleatória e XGBoost, utilizando as métricas AUC-ROC, F1-score, Precisão e Sensibilidade (*Recall*).
  - (II) O segundo cenário adotará validação cruzada com a técnica StratifiedKFold para preservar a proporção entre as classes. Os mesmos modelos e métricas utilizados no primeiro cenário serão aplicados, permitindo uma análise comparativa da robustez e desempenho dos modelos em diferentes abordagens de validação.

- **Cenários com custo:** Considerando o forte desbalanceamento dos dados, optamos por introduzir pesos diferenciados na função de perda, visando penalizar mais severamente os erros relacionados à classe minoritária (fraude). Com os pesos definidos como $w_0 = 0.50$ para a classe não fraude e $w_1 = 289.43$ para a classe fraude, a função de perda ponderada fica expressa por:

$$L_w(y, \hat{y}) = \begin{cases} 0, & \text{se } y = \hat{y} \\ w_0 = 0.50, & \text{se } y = 0 \text{ e } \hat{y} = 1 \\ w_1 = 289.43, & \text{se } y = 1 \text{ e } \hat{y} = 0, \end{cases}$$

  desta forma, a função de perda ponderada permite ao modelo focar na correta identificação

dos casos fraudes, minimizando custos operacionais ou financeiros decorrentes de falsos negativos.

- (III) O primeiro cenário utilizará uma divisão dos dados em treino e teste, na proporção de 70% para treino e 30% para teste. Neste cenário serão avaliados os seguintes modelos: Regressão logística, Floresta aleatória e XGBoost, utilizando as métricas AUC-ROC, F1-score, Precisão e Sensibilidade com custo diferentes para cada classes da variável resposta, o vetor de custo utilizado é baseado na distribuição empiríca obsevada, para nosso caso vamos utilizar: {0: 0.50, 1: 289.43}.
- (IV) O segundo cenário adotará validação cruzada com a técnica StratifiedKFold para preservar a proporção entre as classes. Os mesmos modelos e métricas utilizados no primeiro cenário serão aplicados, permitindo uma análise comparativa da robustez e desempenho dos modelos em diferentes abordagens de validação.

4.4 Comparação de cenários

Nesta Seção apresentamos os resultados dos quatro modelos ajustados nos cenários em estudo. A Tabela 4.1 apresenta as métricas: Acurácia, Sensibilidade, Precisão, F1-Score e AUC-ROC para cada modelo nos quatro diferentes cenários.

Tabela 4.1: Métricas para os modelos nos cenários em estudo.

Cenário	**Métrica**	**Logistica**	**Random Forest**	**XGBoost**
Data-split (70-30) sem custo	Acurácia	0.999262	0.99957	0.999602
Data-split (70-30) sem custo	Recall	0.716216	0.810810	0.824324
Data-split (70-30) sem custo	Precisão	0.834645	0.9375	0.938461
Data-split (70-30) sem custo	F1-Score	0.770909	0.869565	0.877697
Data-split (70-30) sem custo	AUC-ROC	0.870	0.980	0.990
K-Fold estratificado sem custo	Acurácia	0.999197	0.999027	0.999508
K-Fold estratificado sem custo	Recall	0.635574	0.784884	0.802326
K-Fold estratificado sem custo	Precisão	0.867384	0.701751	0.903878
K-Fold estratificado sem custo	F1-Score	0.728252	0.738113	0.848833
K-Fold estratificado sem custo	AUC-ROC	0.950	0.972	0.9855
Data-split (70-30) com custo	Acurácia	0.999345	0.999544	0.999462
Data-split (70-30) com custo	Recall	0.743243	0.868108	0.891892
Data-split (70-30) com custo	Precisão	0.859375	0.885862	0.869815
Data-split (70-30) com custo	F1-Score	0.797101	0.876894	0.880715
Data-split (70-30) com custo	AUC-ROC	0.870	0.980	0.990
K-Fold estratificado com custo	Acurácia	0.9730945	0.999027	0.999594
K-Fold estratificado com custo	Recall	0.9011628	0.808751	0.819748
K-Fold estratificado com custo	Precisão	0.8570488	0.943884	0.937247
K-Fold estratificado com custo	F1-Score	0.878552	0.871108	0.868699
K-Fold estratificado com custo	AUC-ROC	0.979	0.974	0.987

Os modelos avaliados—Logística, Floresta aleatória e XGBoost — apresentaram excelente desempenho em termos de acurácia (todos acima de 99% em praticamente todos os cenários), o que é esperado devido ao alto desbalanceamento dos dados no conjunto "creditcard.csv".

Contudo, métricas mais adequadas para este contexto, como Sensibilidade, Precisão, F1-score e AUC-ROC, revelaram diferenças mais marcantes. XGBoost, em geral, obteve resultados ligeiramente superiores na maioria dos critérios avaliados, destacando-se especialmente pelo elevado AUC-ROC, acima de 0.985 na maior parte dos cenários testados.

Ao analisar o cenário sem custo, podemos notar que a técnica de treinamento influenciou principalmente a Sensibilidade, Precisão e F1-score. O cenário "Data-split (70-30)"apresentou valores consistentemente superiores ao "K-Fold estratificado", especialmente para os modelos Floresta aleatória e XGBoost, que tiveram significativa redução da precisão e do F1-score ao utilizar o treinamento estratificado (de 0.9375 para 0.7017 e 0.9385 para 0.9039, respectivamente).

Nos cenários com custo, os modelos novamente demonstraram uma variação clara conforme a técnica de treinamento. Notavelmente, a regressão logística obteve um aumento substancial na Sensibilidade ao mudar de "Data-split (70-30)"(0.7432) para "K-Fold estratificado"(0.9012), embora tenha havido um pequeno decréscimo na precisão e na acurácia. Floresta aleatória e XGBoost também mostraram leve variação nas métricas, mas mantiveram desempenho estável e elevado, com destaque para a alta precisão obtida pela Floresta aleatória (0.9439 no K-Fold estratificado).

Ao analisar os resultados do método Data-split com e sem custo, verifica-se claramente o impacto positivo da introdução do custo nas classes minoritárias. Por exemplo, a Sensibilidade apresentou melhoria significativa para todos os modelos, evidenciando maior sensibilidade na identificação das fraudes. O XGBoost conseguiu manter alta performance tanto no cenário sem custo quanto com custo, indicando excelente robustez.

Na comparação do método K-Fold estratificado entre cenários com e sem custo, destaca-se especialmente a Regressão Logística que obteve uma considerável melhora na Sensibilidade (0,636 sem custo para 0,901 com custo) e no F1-Score (0,728 sem custo para 0,879 com custo), mesmo apresentando leve queda na Precisão (de 0,867 para 0,857). A Floresta aleatória também obteve melhoria expressiva no cenário com custo, especialmente em Precisão e F1-Score, enquanto o XGBoost demonstrou métricas consistentes, com melhorias modestas de Sensibilidade e F1-Score. Assim, o cenário com custo claramente favorece um melhor balanceamento entre as métricas de desempenho, especialmente na identificação das classes minoritárias no esquema K-Fold estratificado.

4.5 Relatório de classificação - melhor modelo de cada cenário (XGBoost)

Nesta Seção apresentamos o relatório de classificação para o modelo Xgboost no quatro diferentes cenários de estudo. Nosso objetivo é identificar o padrão de classificação e ganho apresentado pelo modelo.

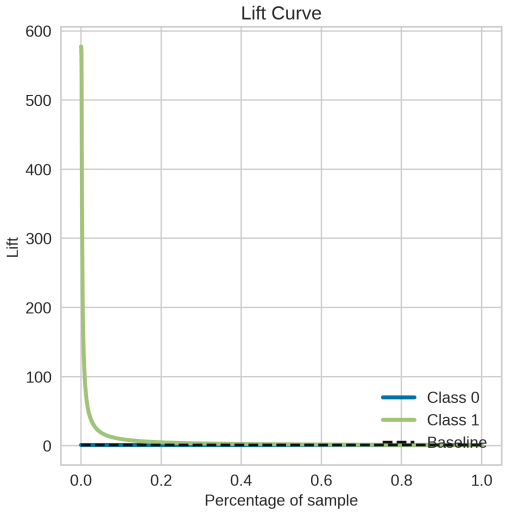

Figura 4.1: Curva Lift XGboost

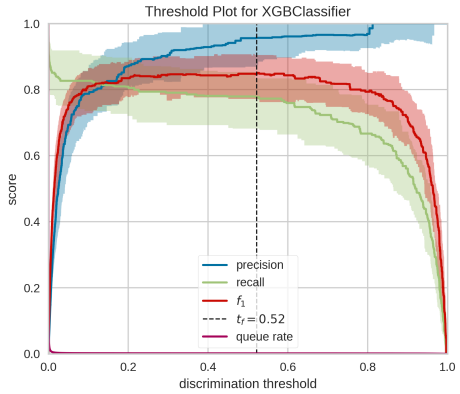

Figura 4.2: gráfico de limiar XGBoost

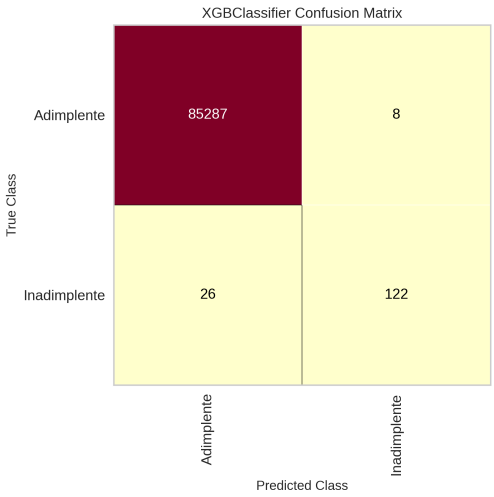

Figura 4.3: Matriz de confusão XGBoost

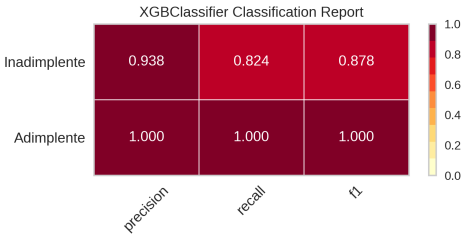

Figura 4.4: Relatório de classificação XGBoost

Figura 4.5: Comparação de cenário 1.

A curva Lift indica o quanto o modelo é eficiente ao identificar casos positivos (fraudes, classe 1) em comparação à seleção aleatória. A alta elevação inicial (próxima a 600 vezes no começo da curva) evidencia que o modelo XGBoost possui excelente capacidade para priorizar

corretamente casos positivos nos primeiros percentuais da amostra. Esse resultado é especialmente relevante em contextos como fraudes ou inadimplências, onde o objetivo é capturar rapidamente o maior número possível de eventos raros.

O gráfico de limiar mostra como as métricas (Precisão, Sensibilidade, F1-score) variam com diferentes valores do limiar de decisão. O limiar ótimo encontrado foi de aproximadamente 0,52, representado pela linha pontilhada. Neste ponto, há equilíbrio adequado entre precisão (aproximadamente 0,94) e Sensibilidade (aproximadamente 0,82), refletindo um valor ideal de F1-score próximo a 0,88. A métrica "queue rate"próxima de zero sugere que poucos casos são classificados como positivos erroneamente quando o limiar ideal é adotado, confirmando o bom desempenho do modelo.

A matriz de confusão evidencia um desempenho sólido do modelo XGBoost. O número de verdadeiros positivos (fraudes corretamente detectados) é de 122, enquanto há apenas 26 falsos negativos. No caso dos não fraudes, o modelo acertou 85.287 casos e cometeu apenas 8 erros (falsos positivos). Esses números indicam excelente desempenho na classificação, especialmente considerando o contexto desbalanceado dos dados, já que o modelo conseguiu classificar corretamente a grande maioria dos casos em ambas as classes.

O relatório de classificação resume o desempenho geral do modelo XGBoost com métricas específicas por classe. Para a classe "fraude", o modelo apresentou precisão elevada (0,938), Sensibilidade satisfatório (0,824) e F1-score robusto (0,878), indicando bom equilíbrio na classificação da classe minoritária. A classe majoritária "não fraude"teve todas as métricas iguais a 1, mostrando desempenho perfeito devido à alta proporção e facilidade em identificar corretamente esses casos.

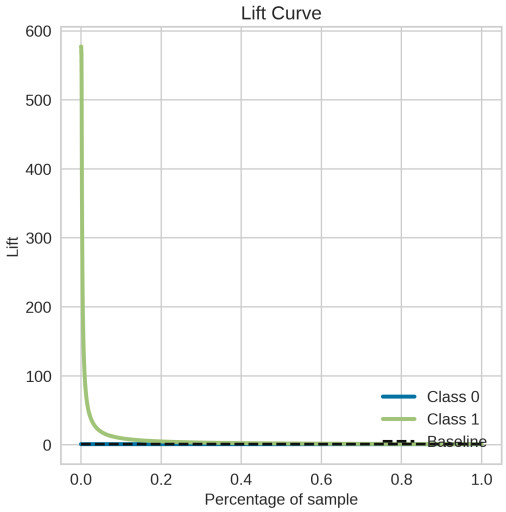

Figura 4.6: Curva Lift XGboost

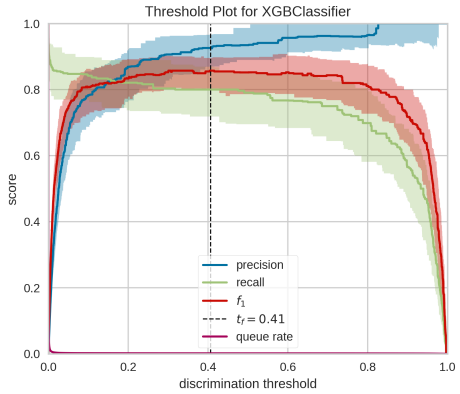

Figura 4.7: Gráfico de limiar XGBoost

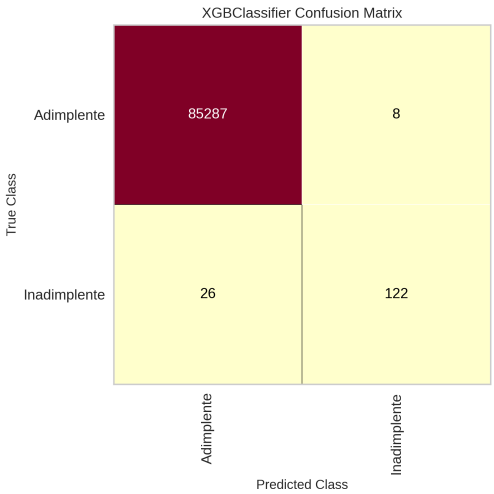

Figura 4.8: Matriz de confusão XGBoost

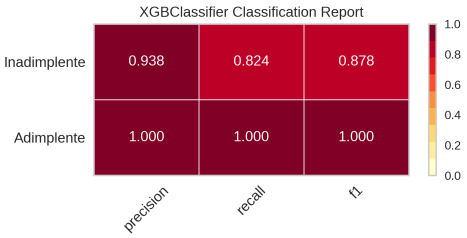

Figura 4.9: Relatório de classificação XGBoost

Figura 4.10: Comparação de cenário 2.

A curva Lift no cenário 2 também demonstra excelente desempenho do modelo XGBoost para a identificação antecipada dos casos da classe positiva (fraude). Observa-se novamente uma elevação inicial extremamente alta (próximo de 600 vezes acima da baseline), indicando que o modelo identifica rapidamente grande parte dos fraudes nos primeiros percentuais da amostra. Tal desempenho reforça a capacidade preditiva robusta do modelo, especialmente valiosa em contextos altamente desbalanceados.

Nesse cenário, o limiar ótimo de discriminação está definido em aproximadamente 0,41, um pouco menor em comparação ao cenário anterior (0,52). Esse limiar ajustado indica uma preferência por Sensibilidade mais elevado (capturar mais fraudes), possivelmente às custas de

uma pequena redução na precisão. As métricas no limiar escolhido apresentam valores altos e equilibrados: precisão em torno de 0,94, Sensibilidade em torno de 0,82 e F1-score próximo a 0,88. Isso sugere que o ajuste do limiar foi eficaz em manter excelente equilíbrio entre as métricas, especialmente quando se deseja capturar mais casos positivos.

A matriz de confusão confirma a eficácia do modelo XGBoost neste cenário, sendo muito semelhante ao cenário anterior. Observa-se um alto número de verdadeiros negativos (85.287), com pouquíssimos falsos positivos (8 casos). Em relação aos fraudes, há novamente 122 casos identificados corretamente (verdadeiros positivos), com apenas 26 falsos negativos. Esses resultados consolidam a capacidade do modelo em classificar corretamente ambas as classes com alta confiabilidade.

O relatório de classificação deste cenário indica novamente alta performance do modelo. Para a classe "fraude"(classe minoritária), o modelo atingiu uma precisão de 0,938, Sensibilidade de 0,824 e F1-score de 0,878, valores consistentemente elevados e praticamente idênticos ao cenário anterior. Para a classe majoritária (não fraude), o modelo manteve uma performance perfeita em todas as métricas, com valores iguais a 1, evidenciando que o modelo não tem dificuldades em identificar corretamente a classe dominante.

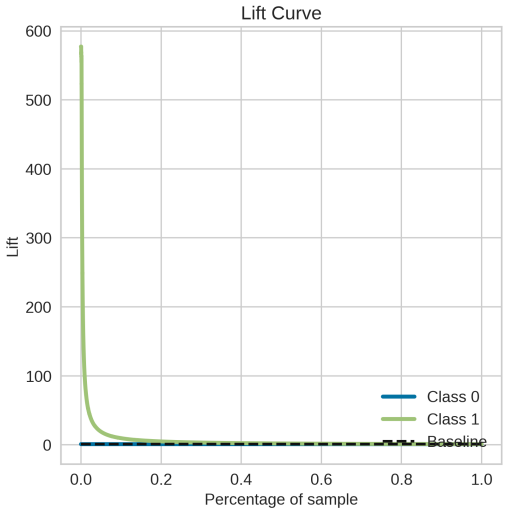

Figura 4.11: Curva Lift XGboost

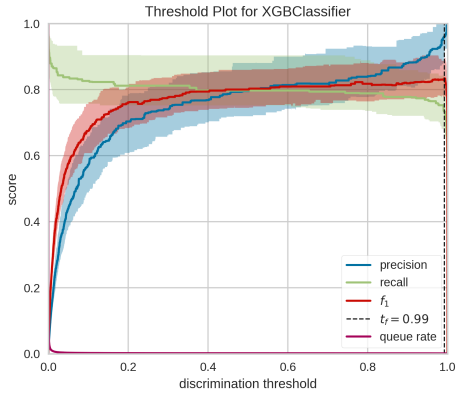

Figura 4.12: Gráfico de limiar XGBoost

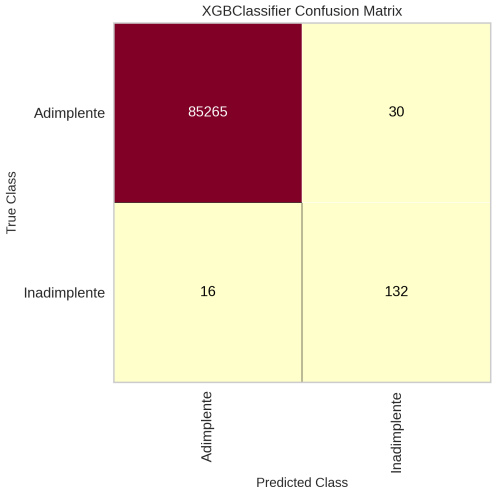

Figura 4.13: Matriz de confusão XGBoost

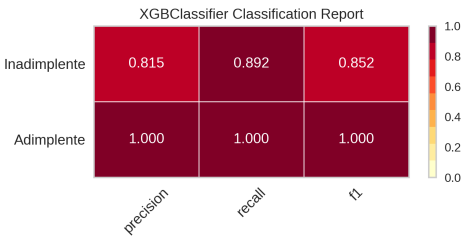

Figura 4.14: Relatório de classificação XGBoost

Figura 4.15: Comparação de cenário 3.

A curva Lift do cenário 3 apresenta novamente uma alta elevação inicial, indicando a excelente capacidade do modelo em priorizar corretamente a classe fraude nos primeiros percentuais da amostra. O valor inicial próximo a 600 vezes a baseline reforça a eficácia do modelo em concentrar corretamente a maioria dos fraudes, permitindo uma identificação precoce e eficaz destes casos críticos.

Neste cenário, o limiar ideal identificado é significativamente mais alto (0,99). Isso implica uma preferência acentuada por precisão sobre Sensibilidade, com o objetivo de reduzir ao máximo os falsos positivos. Nesse limiar, nota-se que a precisão alcança valores máximos (acima de 0,90), porém, a Sensibilidade permanece relativamente alto, próximo de 0,89, resultando

em um F1-score estável (cerca de 0,85). Esse ajuste de limiar revela uma estratégia mais conservadora, ideal para situações onde a classificação incorreta de casos normais como fraudes é altamente custosa.

A matriz de confusão confirma os efeitos da estratégia conservadora adotada neste cenário. O modelo conseguiu identificar corretamente 132 fraudes (verdadeiros positivos), reduzindo significativamente o número de falsos negativos para 16. No entanto, houve um leve aumento nos falsos positivos (30 casos erroneamente classificados como fraudes). O número de verdadeiros negativos continua elevado (85.265). Assim, este cenário destaca-se pela capacidade de capturar mais fraudes reais, mesmo aumentando ligeiramente os falsos alarmes.

O relatório de classificação revela claramente o impacto do limiar alto na precisão e na Sensibilidade. Para a classe fraude, o modelo atingiu precisão de 0,815, Sensibilidade de 0,892 e um F1-score de 0,852, refletindo um equilíbrio ainda sólido. A classe majoritária mantém as métricas perfeitas (1,0), indicando um modelo eficaz na classificação correta dos não fraudes.

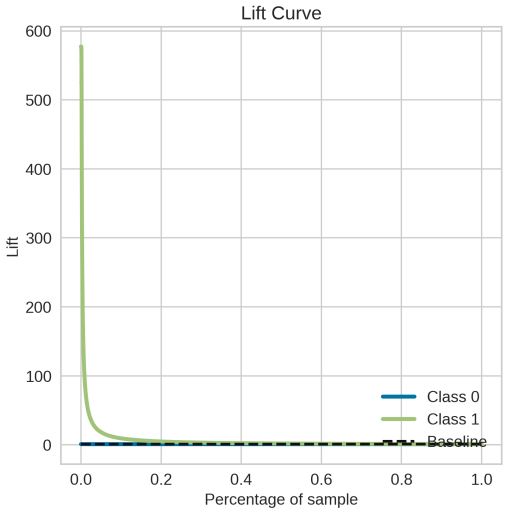

Figura 4.16: Curva Lift XGboost

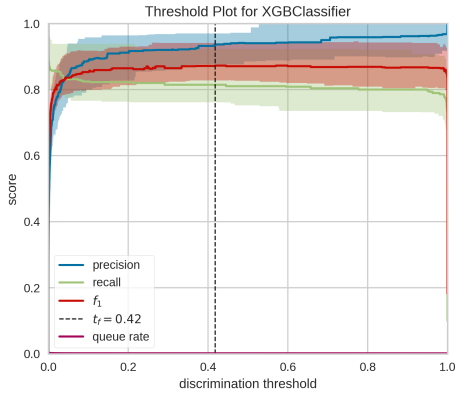

Figura 4.17: Gráfico de limiar XGBoost

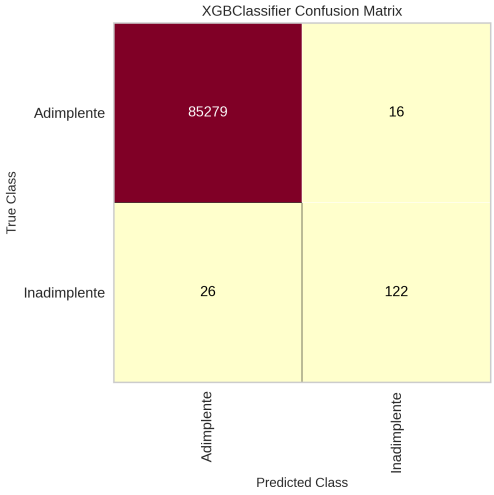

Figura 4.18: Matriz de confusão XGBoost

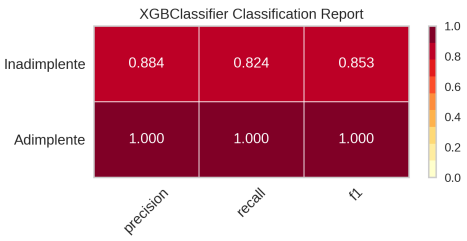

Figura 4.19: Relatório de classificação XGBoost

Figura 4.20: Comparação de cenário 4.

A curva Lift apresentada no cenário 4 mantém uma excelente performance inicial. A elevação observada no início da curva, próxima a 600 vezes em relação à baseline, confirma que o modelo continua sendo altamente eficaz em priorizar os fraudes corretamente. Esse comportamento indica que, mesmo com mudanças nos critérios de decisão do modelo (limiar), o XGBoost mantém uma capacidade robusta e constante para selecionar prioritariamente casos críticos.

O gráfico do limiar de discriminação neste cenário identifica o limiar ótimo em aproximadamente 0,42, o que revela um equilíbrio mais intermediário entre precisão e Sensibilidade em relação ao cenário anterior. Neste ponto, observa-se uma precisão em torno de 0,88, um Sensibilidade próximo de 0,82 e F1-score estável em torno de 0,85. Esse limiar oferece uma boa

combinação, com desempenho consistente em todas as métricas, adequado para situações onde há necessidade de balancear a captura de fraudes e a minimização dos falsos alarmes.

A matriz de confusão reflete claramente o equilíbrio encontrado pelo limiar escolhido. O modelo detectou corretamente 122 casos fraudes (verdadeiros positivos) e teve 26 falsos negativos, mantendo uma proporção similar aos cenários anteriores. Por outro lado, houve uma redução considerável nos falsos positivos (16 casos), mantendo o número elevado de verdadeiros negativos (85.279). Essa matriz evidencia que a estratégia adotada no cenário 4 busca equilibrar ao máximo a precisão e Sensibilidade.

O relatório confirma a boa performance geral observada nas outras figuras. A classe fraude apresentou uma precisão alta (0,884), um Sensibilidade consistente (0,824) e F1-score robusto (0,853). Para a classe não fraude, o desempenho permanece ideal (precisão, Sensibilidade e F1-score iguais a 1), mostrando que o modelo mantém alta eficiência na classificação dos casos majoritários.

Comparação crítica com outros trabalhos usando o mesmo banco de dados

Apesar dos excelentes resultados do TCC (AUC-ROC acima de 0,97 para Floresta aleatória e XGBoost), diversos estudos que usaram exatamente o mesmo dataset (transações europeias de 2013) reportam métricas ainda superiores em recall e F1. Por exemplo, El Bazi et al. (El Bazi *et al.*, 2024) empregaram um ensemble de stacking (XGBoost+CatBoost+LightGBM com otimização bayesiana dos hiperparâmetros) e obtiveram recall = 0,86 e F1 = 0,87 (AUC = 0,988), superando claramente valores típicos de modelos básicos. De forma similar, Chu et al. (Chu *et al.*, 2023) construíram um SVM treinado diretamente no dataset original (sem amostragem) e alcançaram pontuações muito elevadas; eles relatam que seu modelo supera em cerca de 3% o F1 de modelos de regressão logística anteriores, mantendo acurácia comparável e ficando apenas 2–3% abaixo do melhor recall/AUC reportado em outro modelo híbrido ("AllKNN-CatBoost"). Kabane (Kabane, 2024) demonstra outro caso extremo: ao aplicar SMOTE antes da divisão treino/teste, o XGBoost praticamente acertou todas as fraudes ($recall \approx 100\%, F1 \approx 0,9997, AUC \approx 0,9997$). Quando a amostragem é feita somente no conjunto de treino (cenário correto), o mesmo autor obteve recall na faixa de 92-93% e F1 em torno de 94-95%, mais consistentes com o que se espera. Em contrapartida, Sharma (Sharma, 2024) usando SMOTE e modelos clássicos reportou recall de 88,14% (F1 = 0,12) para Regressão Logística e recall de

80,00% (F1 = 0,81) para Floresta aleatória. Esse resultado exemplifica o trade-off: a regressão logística alcança alta sensibilidade (recall) mas com precisão baixíssima (6,44%), resultando em F1 muito baixo, enquanto a floresta aleatória achou compromisso melhor (recall menor, $F1 \approx 0,81$).

Essas diferenças metodológicas justificam os ganhos observados em recall/F1. Muitos estudos aplicam técnicas de reamostragem para balancear as classes: oversampling (SMOTE, CGAN) ou undersampling, o que tende a aumentar a sensibilidade do modelo às fraudes. Por exemplo, Kabane (Kabane, 2024) mostra que SMOTE pre-split eleva recall drasticamente (ao custo de superajuste), enquanto Du et al. (Du *et al.*, 2023) observaram que, no seu modelo autoencoder+LightGBM (AED-LGB), a reamostragem não melhorou a performance geral, indicando que o ganho nem sempre é garantido. Outros autores optam por não usar amostragem alguma, justamente para evitar viés: Chu et al. (Chu *et al.*, 2023) destacam que subamostragem pode eliminar informações úteis e oversampling pode causar overfitting, por isso trabalharam com o dataset original.

Outro fator crítico é o limiar de decisão: escolher um threshold de classificação diferente de 0,5 pode aumentar recall às custas de precisão. Du et al. (Du *et al.*, 2023) ajustaram manualmente o limiar do LightGBM para 0,05 (em vez de 0,5), elevando o verdadeiro positivo rate (recall) para 89,29%. Em geral, reduzir o limiar aumenta recall mas reduz precisão, o que é refletido em F1 e MCC. A afinagem de hiperparâmetros também altera resultados: El Bazi et al. (El Bazi *et al.*, 2024) usaram otimização bayesiana em seu ensemble, obtendo melhorias notáveis em métricas como recall e AUC em comparação aos modelos não afinados. Kabane ajustou parâmetros de regularização (e escalas de peso de classe) e mostrou que isso pode elevar métricas globais (por exemplo, F1 atingiu 0,95 em um caso com CGAN + cost-sensitive).

Finalmente, modelos mais sofisticados (ensembles complexos, redes neurais, etc.) tendem a capturar padrões que modelos simples não captam, elevando recall/F1. Isso explica por que o stacking de El Bazi et al. superou cada técnica isolada e por que abordagens híbridas com autoencoder+LGBM de Du et al. reportaram MCC e TPR mais altos que LGBM puro. Em resumo, variações no pré-processamento (SMOTE, undersampling), no ajuste de hiperparâmetros, na escolha do limiar decisório e na complexidade do ensemble são responsáveis pelas discrepâncias de métricas observadas entre a análise realizada aqui e outros estudos usando o mesmo dataset.

Capítulo 5

Conclusão

Esta pesquisa explorou o desafio de classificar conjuntos de dados altamente desbalanceados por meio de técnicas sensíveis ao custo, avaliando desde as bases teóricas até a aplicação em dados reais de transações de cartão de crédito. No capítulo 1, contextualizamos o problema do desbalanceamento e apresentamos os objetivos e hipóteses do trabalho, justificando a relevância acadêmica e prática do tema. No capítulo 2, discutimos as causas do desbalanceamento, o paradigma do aprendizado sensível ao custo e revisamos estudos clássicos e contemporâneos sobre reamostragem e algoritmos ponderados. O capítulo 3 detalhou os modelos implementados (Regressão Logística, Florestas Aleatória e XGBoost), bem como as métricas fundamentais de avaliação—da matriz de confusão à curva ROC e validação cruzada estratificada. Finalmente, no capítulo 4, aplicamos esses métodos a um conjunto real de 284.807 transações, demonstrando que técnicas sensíveis ao custo melhoram significativamente o recall na detecção de fraudes, ainda que impusessem trade-offs em precisão.

Contribuições Principais

1. **Recapitulação e alcance de objetivos**: Todos os objetivos gerais e específicos definidos na introdução foram alcançados, confirmando a hipótese de que ajustes de custo melhoram a identificação da classe minoritária.

2. **Avaliação metodológica**: O uso de validação cruzada estratificada e comparação entre data-split e k-fold demonstrou a robustez dos resultados, destacando XGBoost como o melhor modelo para nosso cenário.

3. **Ferramentas práticas**: Implementamos as análises em Python, utilizando `scikit-learn`

e `xgboost`, o que fornece um roteiro replicável para aplicações futuras.

Limitações e Perspectivas Futuras

Apesar do alto desempenho observado, percebe-se que:

- **Definição de custos**: A escolha de pesos exige conhecimento de domínio e pode ser refinada com técnicas de otimização automática.
- **Generalização**: Testes em outros domínios (p.ex., diagnóstico médico) são necessários para validar a aplicabilidade das conclusões.
- **Métricas complementares**: Explorar a curva de precisão-sensibilidade e AUPRC, sugerida para cenários severamente desbalanceados, pode enriquecer a avaliação.

Considerações Finais

Em síntese, este trabalho reforça que o aprendizado sensível ao custo é uma abordagem eficaz para mitigar enviesamentos em classificadores treinados com dados desbalanceados, especialmente quando o objetivo principal é maximizar a detecção da classe minoritária. A integração de diferentes técnicas de ensemble, aliada a uma rigorosa validação estatística e à análise em dados reais, comprovou a robustez do método. Recomenda-se, para trabalhos futuros, investigar estratégias de ajuste dinâmico de pesos e estender o estudo a fluxos de dados em tempo real, consolidando a contribuição deste TCC para a área de estatística aplicada e ciência de dados.

Referências Bibliográficas

Altinger, H., Herbold, S., Schneemann, F., Grabowski, J. e Wotawa, F. (2017). Performance tuning for automotive software fault prediction. Em *2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, páginas 526–530. IEEE.

Araf, I., Idri, A. e Chairi, I. (2024). Cost-sensitive learning for imbalanced medical data: a review. *Artificial Intelligence Review*, **57**, 80.

Araujo, I. (2024). Desafios na construção de classificadores robustos em conjuntos de dados desbalanceados.

Breiman, L., Friedman, J. H., Olshen, R. A. e Stone, C. J. (1984). *Classification and Regression Trees*. Wadsworth International Group.

Brownlee, J. (2020). *Imbalanced Classification with Python: Better Metrics, Balance Skewed Classes, Cost-Sensitive Learning*. Machine Learning Mastery.

Carballo, M. T. (2002). Predição da macrossomia fetal através da regressão logística e de redes neurais artificiais.

Chu, Y. B., Lim, Z. M., Keane, B., Kong, P. H., Elkilany, A. R. e Abusetta, O. H. (2023). Credit card fraud detection on original european credit card holder dataset using ensemble machine learning technique. *Journal of Cyber Security*, **5**, 33–46.

Costa, C. F. e Nascimento, M. A. (2016). Ida 2016 industrial challenge: Using machine learning for predicting failures. Em *International Symposium on Intelligent Data Analysis*, páginas 381–386. Springer, Cham.

Domingos, P. (1999). Metacost: A general method for making classifiers cost-sensitive. Em *Proceedings of the Fifth ACM SIGKDD*, páginas 155–164.

Drummond, C. e Holte, R. C. (2000). Exploiting the cost (in)sensitivity of decision tree splitting criteria. Em *Proceedings of the Seventeenth International Conference on Machine Learning*, páginas 239–246.

Drummond, C. e Holte, R. C. (2006). Cost curves: An improved method for visualizing classifier performance. *Machine Learning*, **65**(1), 95–130.

Du, H., Lv, L., Guo, A. e Wang, H. (2023). Autoencoder and lightgbm for credit card fraud detection problems. *Symmetry*, **15**(4), 870.

El Bazi, A., Chrayah, M., Aknin, N. e Bouzidi, A. (2024). Enhancing credit card fraud detection using a stacking model approach and hyperparameter optimization. *International Journal of Advanced Computer Science and Applications*, **15**(10), 1081–1087.

Elkan, C. (2001). The foundations of cost-sensitive learning. *International Joint Conference on Artificial Intelligence*, páginas 973–978.

Fan, W., Stolfo, S. J., Zhang, J. e Chan, P. K. (1999). Adacost: Misclassification cost-sensitive boosting. Em *Proceedings of the Sixteenth International Conference on Machine Learning*, páginas 97–105. Morgan Kaufmann.

Han, H., Wang, W. e Mao, B. (2012). Borderline-smote: A new over-sampling method in imbalanced data sets learning. *Advances in Intelligent Systems and Computing*, **164**, 878–887.

He, H. e Garcia, E. A. (2009). Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, **21**(9), 1263–1284.

Horta, R. A. M. (2010). *Uma metodologia de Mineração de Dados para a previsão de insolvência de empresas brasileiras de capital aberto*. Doutorado em engenharia civil, Universidade Federal do Rio de Janeiro, Rio de Janeiro.

Hosmer, D. W. e Lemeshow, S. (1989). *Applied logistic regression*. John Wiley & Sons, Inc., New York.

Japkowicz, N. (2000). The class imbalance problem: Significance and strategies. Em *Proceedings of the International Conference on Artificial Intelligence*. Pages not available.

Kabane, S. (2024). Impact of sampling techniques and data leakage on xgboost performance in credit card fraud detection.

Kubat, M. e Matwin, S. (1997). Addressing the curse of imbalanced training sets: One-sided sampling. Em *Proceedings of the Fourteenth International Conference on Machine Learning*, páginas 179–186. Morgan Kaufmann.

Machine Learning Group - ULB (2018). Credit card fraud detection. https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud. Anonymized credit card transactions by European cardholders in September 2013. Acesso em: 22 abril 2025.

Margineantu, D. (2009). Methods for cost-sensitive learning.

Megahed, F. M., Chen, Y. J., Megahed, A. *et al.* (2021). The class imbalance problem. *Nature Methods*, **18**, 1270–1272.

Modesto Angelico, J. P. (2025). Credit card fraud detection notebook. https://colab.research.google.com/drive/1IUp1iCo-NaXQBSuDNzByF-D5W61FdwmQ?usp=sharing. Google Colab notebook. Acesso em: 22 abril 2025.

Pereira, P. J., Pereira, A., Cortez, P. e Pilastri, A. (2021). A comparison of machine learning methods for extremely unbalanced industrial quality data. Em *EPIA Conference on Artificial Intelligence*, páginas 561–572. Springer, Cham.

Sharma, P. (2024). A study on detection of credit card fraud using machine learning techniques. *International Journal for Multidisciplinary Research (IJFMR)*, **6**(6), 1–10.

Sterner, T. e. a. (2024). Cost-sensitive learning for imbalanced medical data: a review. *Artificial Intelligence Review*, **57**(3).

Turney, P. (2002). Types of cost in inductive concept learning. *Journal of Machine Learning Research*, **3**, 1–33.

Wang, T. (2013). Efficient techniques for cost-sensitive learning with multiple cost considerations. University of Technology, Sydney. Faculty of Engineering and Information Technology. Microsoft Open Source Code of Conduct.

Weiss, G. M. e Provost, F. (2001). The effect of class distribution on classifier learning. Relatório Técnico ML-TR 43, Department of Computer Science, Rutgers University.

Zadrozny, B. e Elkan, C. (2001a). Learning and making decisions when costs and probabilities are both unknown. Relatório Técnico CS2001-0664, University of California, San Diego.

Zadrozny, B. e Elkan, C. (2001b). Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. Em *Proceedings of the Eighteenth International Conference on Machine Learning*.

Zheng, W. e Jin, M. (2020). The effects of class imbalance and training data size on classifier learning. *SN Computer Science*, **1**, 71.

Zhou, Z. e Liu, X. (2006). Training cost-sensitive neural networks with methods addressing the class imbalance problem. *IEEE Transactions on Knowledge and Data Engineering*, **18**(1), 63–77.