UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Análise comparativa de métodos de seleção de variáveis em problemas de classificação

Luna Wagner Cunha

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

ICMC USP
SÃO CARLOS

Luna Wagner Cunha

Análise comparativa de métodos de seleção de variáveis em problemas de classificação

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

Área de Concentração: Estatística

Orientadora: Profa. Dra. Cibele Maria Russo Novelli

USP – São Carlos
Janeiro de 2025

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

Cunha, Luna Wagner
C972a Análise comparativa de métodos de seleção de variáveis em problemas de classificação / Luna Wagner Cunha; orientadora Cibele Maria Russo. -- São Carlos, 2024.
89 p.

Dissertação (Mestrado - Programa Interinstitucional de Pós-graduação em Estatística) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2024.

1. Seleção de variáveis. 2. Valores SHAP. 3. Lasso. 4. Aprendizado de máquina. I. Russo, Cibele Maria, orient. II. Título.

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

Luna Wagner Cunha

Comparative analysis of variable selection methods in classification problems

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

Concentration Area: Statistics

Advisor: Profa. Dra. Cibele Maria Russo Novelli

USP – São Carlos
January 2025

Dedico este trabalho primeiramente aos meus pais, cujo amor incondicional, apoio inabalável e dedicação incansável tornaram possível cada passo desta jornada acadêmica. Vocês foram a fonte inesgotável de inspiração e força ao longo dos anos, e este trabalho é um reflexo do valor que vocês sempre atribuíram à educação.

Dedico também à criança que um dia fui, cheia de sonhos e curiosidades, que via na educação o caminho para evoluir e fazer a diferença no mundo.

Que estas dedicatórias expressem meu profundo reconhecimento e gratidão por cada pessoa e momento que contribuíram para esta conquista.

AGRADECIMENTOS

Agradeço primeiramente aos meus pais, Djenane e Marcos, pela dedicação incansável, pelo incentivo constante e pelo exemplo de como a educação é transformadora. Sem o apoio de vocês, não teria alcançado este momento tão significativo em minha vida acadêmica.

À minha orientadora, Cibele Russo, que desde o meu TCC de graduação tem sido uma guia. Seu apoio, paciência e conhecimento foram fundamentais para o desenvolvimento desta dissertação. Aprendi muito com você e sou grata pela oportunidade de trabalhar sob sua orientação.

Às professoras Luciana Godoy e Daiane Zuanetti, cujas contribuições críticas e sugestões foram essenciais para aprimorar este trabalho e torná-lo mais relevante academicamente.

Agradeço também aos institutos do ICMC e da UFSCar pelo apoio contínuo ao longo de todo o processo de construção deste projeto.

Por fim, a todos que de alguma forma contribuíram para este trabalho, meu profundo agradecimento. Este momento não seria possível sem o suporte e a colaboração de cada um de vocês.

RESUMO

CUNHA, L. W. **Análise comparativa de métodos de seleção de variáveis em problemas de classificação**. 2025. 89 p. Dissertação (Mestrado em Estatística – Programa Interinstitucional de Pós-Graduação em Estatística) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2025.

Neste trabalho é apresentada uma comparação abrangente entre o método SHAP e o método Lasso para seleção de variáveis. As metodologias de ambas as técnicas são exploradas e justapostas, alavancando métricas de verificação de classificação, como precisão, recall, pontuação F1 e acurácia em cenários de banco de dados simulados e reais. Os testes aplicados mostram o SHAP como um bom competidor para métodos de seleção de variáveis, com resultados inclusive levemente superiores em relação ao Lasso nos casos aqui apresentados, tanto na base simulada quando na base real, mantendo níveis de acurácia competitivos em relação ao modelo completo. Apesar de ficar um pouco próximo na acurácia, o SHAP reduz notavelmente o espaço de variáveis, demonstrando sua proeza na seleção. Além disso, um estudo de robustez usando valores de perturbação em treinamento, teste e em ambos conjuntamente, confirma a resiliência das variáveis selecionadas pelo SHAP, particularmente em termos de acurácia. Essas análises destacam a eficácia do método SHAP como uma ferramenta versátil e poderosa para seleção de variáveis, prometendo melhor interpretabilidade e desempenho em aplicações de aprendizado de máquina.

Palavras-chave: Valores SHAP, Aprendizado de máquina, Seleção de variáveis, Lasso.

ABSTRACT

CUNHA, L. W. **Comparative analysis of variable selection methods in classification problems**. 2025. 89 p. Dissertação (Mestrado em Estatística – Programa Interinstitucional de Pós-Graduação em Estatística) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2025.

In this study, a comprehensive comparison between the SHAP method and the Lasso method for variable selection is presented. The methodologies of both techniques are explored and juxtaposed, utilizing key selection metrics such as precision, recall, F1 score, and accuracy in both simulated and real database scenarios. The tests applied show SHAP as a good competitor for variable selection methods, with results even slightly superior to Lasso in the cases presented here, both on a simulated and real basis, maintaining competitive accuracy levels in relation to the complete model. Despite being somewhat close in accuracy, SHAP notably reduces the variable space, demonstrating its selection prowess. Additionally, a robustness study involving perturbation values in training, testing, and combined datasets confirms the resilience of the variables selected by SHAP, particularly in terms of accuracy. These analyses underscore the efficacy of the SHAP method as a versatile and potent tool for variable selection, promising improved interpretability and performance in machine learning applications.

Keywords: SHAP values, Machine learning, Feature selection, Lasso.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

CAPÍTULO

1

INTRODUÇÃO

Algoritmos que utilizam aprendizado de máquina são atualmente aplicados em vários campos, incluindo finanças, medicina, economia, biologia e outros que exigem previsões baseadas em múltiplas características. Embora seja esperado que um conjunto maior de variáveis melhore a capacidade discriminatória do modelo e o aprendizado algorítmico geral, na realidade, vetores excessivamente grandes podem causar atrasos significativos no processo de aprendizado e levar à possibilidade de *overfitting*, (BREIMAN, 2001) termo utilizado quando o modelo se ajusta perfeitamente aos dados de treino mas não identifica o padrão, comprometendo a generalização do modelo, o que leve ao baixo desempenho em dados de teste.

Para evitar tais problemas, foram propostos métodos de seleção de variáveis, ou seja, processos de seleção de um subgrupo de características, identificando as mais interessantes e relevantes para o modelo, reduzindo, assim, a dimensão dos dados de entrada. A seleção de variáveis tem sido amplamente aplicada em diversos temas como, por exemplo, categorização de texto (FORMAN *et al.*, 2003; LEE; LEE, 2006), bioinformática (SAEYS; INZA; LARRANAGA, 2007) e astronomia (ZHENG; ZHANG, 2008).

Um dos mais novos métodos que são utilizados na seleção de variáveis utilizam a interpretabilidade e explicabilidade das variáveis dos modelos. Mas isso não é uma tarefa fácil, especialmente se falarmos em modelos de aprendizado de máquina, muitas vezes chamados de modelos “caixas pretas", por não permitirem identificar de forma explícita a função que é modelada por seus algoritmos, uma vez que o interesse maior está voltado a previsão.

Atualmente tem sido muito valorizado e até mesmo necessário conseguir que o modelo ajustado seja tanto preciso quanto interpretável, ou seja, entender o porquê de uma certa predição ou decisão. Dois conceitos simples (DOSHI-VELEZ; KIM, 2017) ajudam a entender essa necessidade: **justiça** (*fairness*), ou seja, a garantia que as previsões são imparciais, não viesadas e que não descriminem grupos sub-representados; e **confiança**, é mais fácil para seres humanos confiaram em um sistema que consegue explicar as decisões tomadas.

Entretanto, embora existam vários exemplos de aplicações com explicações globais e generalizadas de um modelo, interpretações locais, que revelam o impacto e a importância de cada variável em cada predição, não têm sido muito estudadas. Com essa motivação, propõe-se neste trabalho o estudo comparativo das técnicas SHAP e Lasso qeu serão brevemente descritas a seguir.

1.1 SHAP

O SHAP (*SHapley Additive exPlanations*), proposto por Lundberg e Lee (2017), surgiu como um método agregador e conciliador de diferentes técnicas de interpretabilidade de modelos, detalhando as importâncias das variáveis por observação. Desde então, de acordo com o Google, ao menos 14700 artigos citaram esse novo método.

Os valores do SHAP são baseados no valor de *Shapley*, um conceito da teoria dos jogos cooperativos que mede a contribuição de cada jogador para o retorno total de uma coalizão de jogadores. No contexto do aprendizado de máquina, os jogadores são as variáveis de entrada e o retorno é a previsão do modelo. Os valores SHAP medem a contribuição marginal de cada variável para a diferença entre o valor previsto na saída do modelo e o valor esperado, calculado como a previsão média do modelo em todas as combinações possíveis de variáveis.

A partir desse método surgiram outras metodologias que utilizam ou modificam esse cálculo de forma a encontrar a verdadeira contribuição de cada variável. Kwon e Zou (2022) identificaram uma limitação importante do valor de Shapley uma vez que ele atribui pesos uniformes às contribuições marginais no cálculo da pontuação de atribuição. Eles mostraram que isso pode levar a erros de atribuição quando diferentes contribuições marginais têm sinais e ruídos diferentes. Assim propuseram o *WeightedSHAP*, uma generalização do valor de *Shapley* que é mais flexível. Esse método usa uma média ponderada de contribuições marginais onde os pesos podem ser aprendidos a partir dos dados. Esse método não será trabalhada neste trabalho, será utilizado apenas o método SHAP original.

Vários estudos demonstraram a eficácia dos valores SHAP para a seleção de variáveis. Por exemplo, um estudo de Lundberg e Lee (2017) usou valores SHAP para selecionar recursos para uma tarefa de classificação de câncer de mama e obteve maior precisão do que outros métodos de seleção de variáveis. Outro estudo de Ghorbani e Zou (2019) usou valores SHAP para selecionar um subconjunto de variáveis para uma tarefa de detecção de spam e obteve maior precisão do que outros métodos de seleção.

1.2 LASSO

Um algoritmo bem conhecido atualmente para seleção de variáveis é o método Lasso, proposto por Tibshirani (1996), que possui a propriedade da *ridge regression* de reduzir o valor

das estimativas dos parâmetros. O Lasso é capaz de estimar os coeficientes das variáveis de um modelo iguais a zero, fazendo assim uma seleção do melhor subconjunto, descartando variáveis irrelevantes do modelo.

Este método já foi adaptado e utilizado em vários modelos, Fonti e Belitser (2017) trouxeram um exemplo do funcionamento do Lasso, primeiro em uma regressão linear e depois em uma regressão logística. Mais recentemente o Lasso tem sido utilizado como um pré-processamento para algoritmos de aprendizado de máquina. Ghosh *et al.* (2021) apresentou um exemplo de aplicação em uma base de dados relacionado a predição de doença cardiovascular, no qual método Lasso foi utilizado para reduzir a quantidade de covariáveis antes da aplicação de modelos como *Random Forest Bagging Method* (RFBM) e *Decision Tree Bagging Method* (DTBM).

Nesse trabalho serão estudados métodos de seleção de variáveis utilizando a metodologia SHAP e suas ramificações, comparando para referência com o método Lasso, muito utilizado para regressão e seleção de variáveis. Serão feitas aplicações primeiramente em uma base de dados simulada e depois em uma base de dados real, disponível no site Kaggle[1] (KAGGLE, 2022), referente a *Lending Club*, uma empresa americana de empréstimos *peer-to-peer*, com sede em São Francisco, Califórnia. Ela é a maior plataforma de empréstimos *peer-to-peer* do mundo. A empresa alega que US$ 15,98 bilhões em empréstimos foram originados por meio de sua plataforma até 31 de dezembro de 2015.

No Capítulo 2, as metodologias aplicadas serão descritas mais detalhamente. No Capítulo 3 serão apresentados as bases utilizadas nas comparações e os resultados obtidos em cada uma delas. Para o Capítulo 4 serão mostrados os resultados do estudo de robustez em ambas as bases. Por fim, o Capítulo 5 será apresentada uma pequena discussão das apurações feitas.

[1] Disponível em https://www.kaggle.com/datasets/ethon0426/lending-club-20072020q1?resource=download, Acessado em 12/06/2022

CAPÍTULO

2

METODOLOGIA

Neste capítulo serão apresentadas as metodologias aplicadas no trabalho, tanto referente a seleção de variáveis como nas técnicas utilizadas para avaliação e comparação do modelos escolhidos.

2.1 SHAP

Os valores SHAP têm várias vantagens sobre outros métodos de importância de variáveis. Primeiro, eles são agnósticos de modelo, o que significa que podem ser calculados para qualquer tipo de modelo, incluindo modelos de caixa-preta, como redes neurais. Em segundo lugar, eles podem lidar com relacionamentos não lineares entre variáveis independentes e a variável dependente, também chamada de variável alvo. Em terceiro lugar, eles podem levar em conta as interações entre essas variáveis, o que é importante em muitas aplicações do mundo real.

Lundberg *et al.* (2020) trouxeram uma nova aplicação em árvores sobre como, utilizando o SHAP, é possível partir de interpretações locais e chegar a compreensões globais. A Figura 1 exemplifica o resultado obtido utilizando esse novo método, no qual modelos, antes chamados de “caixas pretas”, são abertos e é encontrado um valor aproximado da relevância local que cada variável recebeu para que o modelo obtivesse a predição definida. Nesse exemplo é apresentado uma modelagem feita em cima de dados de saúde, tentando prever o risco de mortalidade para cada indivíduo.

Ao aplicar o *Tree Explainer* do método SHAP no modelo de “caixa-preta” ajustado, é possível visualizar as importâncias que cada variável independente teve no modelo, apresentado no quadrado da direita. Nesse caso, a variável “Pressão sanguínea” teve o maior impacto para aumentar a predição de risco de mortalidade, e a variável “Sexo” apresentou um peso negativo na predição empurrando o valor predito para baixo.

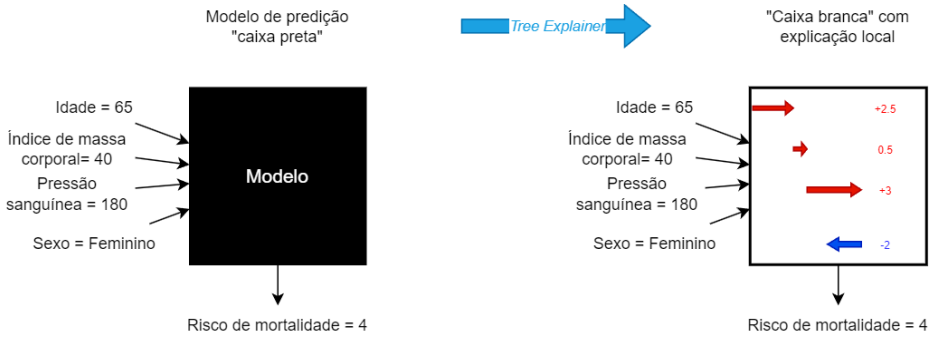


Figura 1 – Caixa Preta e Caixa Branca

Fonte: Lundberg *et al.* (2020).

(Adaptado para o português pela autora)

A Figura 2 exemplifica um outro resultado também encontrado através do SHAP. Utilizando as variáveis de entrada, seus valores e o modelo definido, aplica-se um *Explainer*. Nesse exemplo, apresenta-se o *Tree Explainer*, utilizado para modelos de árvores, de forma a encontrar os valores do SHAP com explicações locais e a partir deles são calculados os *insights* globais, ou seja, compreensões maiores identificadas pelo modelo, uma variável com maior importância para a base, possíveis sub-grupos, entre outros.

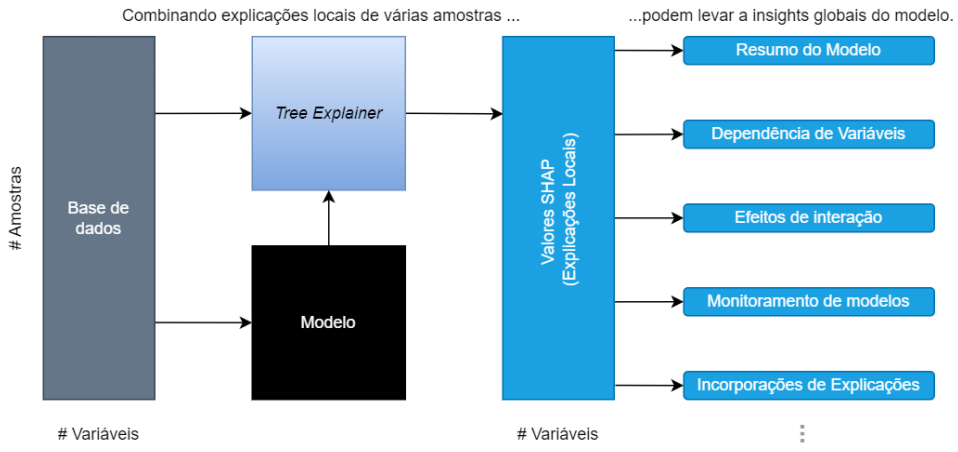


Figura 2 – Interpretabilidade Global

Fonte: Lundberg *et al.* (2020).

(Adaptado para português pela autora)

Os valores SHAP conseguem fornecer uma maneira de atribuir a diferença entre uma previsão individual e a previsão esperada para cada variável de maneira aditiva, garantindo que a soma de seus valores seja igual à diferença nas previsões. E o valor esperado para uma

determinada observação é a previsão média que o modelo faz em todas as combinações possíveis de valores das variáveis, fazendo com que o custo computacional seja alto.

Em seu artigo original para o método SHAP, Lundberg e Lee (2017) mostram que os valores SHAP fornecem uma medida única que avalia de forma aditiva a contribuição de cada variável independente para a predição obtida e que respeita importantes propriedades tais como: linearidade, precisão local, ausência e consistência. A seguir será apresentada a teoria que fundamenta o método SHAP como um método de atribuição de recursos aditivos e uma discussão detalhando as propriedades que ele satisfaz.

O enfoque aqui consiste em considerar métodos locais construídos para explicar uma predição $f(x)$ baseado em uma entrada x. Para isso considere g o modelo explicativo e x' uma entrada simplificada que mapeia a entrada original x tal que $x = h_x(x')$. Métodos locais tentam garantir $g(z') \approx f(h_x(z'))$ quando $z' \approx x'$:

Definição 1: Métodos de atribuição de recursos aditivos tem um modelo explicativo que é uma função linear de valores binários:

$$g(z') = \phi_0 + \sum_{i=1}^{M} \phi_i z'_i, \tag{2.1}$$

onde $z' \in \{0,1\}^M$, M é o número de variáveis de entrada simplificado, e $\phi_i \in \Re$.

Como mostrado por (LUNDBERG; LEE, 2017), métodos com modelos explicativos que respeitam a Definição 1 atribuem um efeito ϕ_i em cada variável e, somando todos os efeitos de todas as atribuições, aproxima-se do modelo original $f(x)$.

Linearidade é uma propriedade desejável que estabelece que a soma dos valores SHAP pra todas as variáveis independentes é igual a diferença entre a previsão do modelo para uma observação e a previsão esperada, ou seja, a previsão média do modelo em todas as combinações possíveis de variáveis.

Em seguida é apresentada a propriedade de **Precisão Local**, que demanda para um modelo original f, uma entrada específica x e M variáveis com ϕ_i a importância de cada variável, o modelo explicativo $g(x')$ deve pelo menos corresponder à saída de f para a entrada simplificada x (correspondente a entrada x). Formalmente,

Precisão local:

$$f(x) = g(x') = \phi_0 + \sum_{i=1}^{M} \phi_i x'_i. \tag{2.2}$$

O modelo explicativo $g(x')$ corresponde ao original $f(x)$ quando $x = h_x(x')$, onde $\phi_0 = f(h_x(0))$ representa a saída do modelo com todas as entradas simplificadas desativadas, ou seja, ausentes. Assim o modelo explicativo deve ser igual o original quando as observações de entrada são todas ausentes.

A terceira propriedade a ser satisfeita pelos valores SHAP é **Ausência**, implicando que, se uma variável estiver ausente (por exemplo, *NaN*), seu valor SHAP é a média dos valores em todas as observações presentes. Basicamente essa propriedade afirma que entradas de variáveis ausentes não deverão ter impacto no modelo. Formalmente,

Ausência:

$$x'_i = 0 \Rightarrow \phi_i = 0. \tag{2.3}$$

A propriedade de **Consistência** afirma que, se um modelo muda, de forma que a contribuição de alguma entrada original aumente ou permaneça igual, independentemente de outras, a importância dessa entrada não deve diminuir. Formalmente,

Consistência: Para $f_x(z') = f(h_x(z'))$ e $z' \setminus i$ denota que $z'_i = 0$. Para quaisquer dois modelos f e f', se

$$f'_x(z') - f'_x(z' \setminus i) \geq f_x(z') - f_x(z' \setminus i), \tag{2.4}$$

para todos as entradas $z' \in \{0,1\}^M$, então $\phi_i(f',x) \geq \phi_i(f,x)$.

Essas propriedades fornecem a base para a interpretação dos valores SHAP e garantem que eles possuam atributos desejáveis para a explicação do modelo.

Antes dos valores SHAP é necessário definir os valores Shapley, que seguem um conceito da teoria de jogos cooperativos, medindo a contribuição de cada jogador para o retorno (ganho) total de uma coalização de jogadores. No contexto de aprendizado de máquina, os jogadores são as variáveis de entrada e o retorno é a diferença entre a previsão do modelo para uma observação específica e a previsão esperada.

O Teorema (2.5) apresenta o cálculo dos valores Shapley ϕ_i que são os únicos a respeitarem as propriedades necessárias (Precisão Local, Ausência e Consistência) para uma solução única no cálculo de importância das variáveis e satisfaz a definição da classe proposta por Lundberg e Lee (2017) de métodos de atribuição aditiva de variáveis, mostrada na Definição 1 (2.1).

Teorema 1: Somente um modelo explicativo possível g atende a definição 1 e as propriedades:

$$\phi_i(f,x) = \sum_{z' \subseteq x'} \frac{|z'|!(M - |z'| - 1)!}{M!} [f_x(z') - f_x(z' \setminus i)], \tag{2.5}$$

onde $|z'|$ é o número de entradas diferentes de zero em z', e $z' \subseteq x'$ representa todos os vetores z', entradas diferentes de zero que são um subgrupo das entradas diferentes de zero de x'. Essa equação representa a importância calculada utilizando a diferença entre os valores preditos e o valor esperado.

Assim se propôs o método SHAP como uma unificação das medidas de importância de variáveis. Os valores SHAP são os valores Shapley de uma função de expectativa condicional do

modelo original; assim eles são a solução para a Equação 2.5 onde $f_x(z') = f(h_x(z')) = E[f(z)|z_S]$ e S é o conjunto de índeces diferentes de zero em z', ou seja, no caso em que o valor esperado é a previsão média do modelo em todas as combinações possíveis de variáveis. Com isso, o SHAP apresenta uma única medida aditiva de importância que respeita as propriedades necessárias e usa expectativas condicionais para definir entradas simplificadas. Implícito nessa definição de valores SHAP está um mapeamento de entrada simplificada $h_x(z') = z_S$, onde z_S tem valores ausentes de entrada para variáveis que não estão presente no grupo S. Como a maioria dos modelos não consegue lidar com padrões arbitrários de valores ausentes, $f(z_s)$ é aproximado com $E[f(z)|z_s]$.

Em uma melhor interpretação, os valores SHAP podem ser resumidos através da seguinte equação:

$$SHAP(feature_i) = \sum \left[\sum \frac{(f(x) - f(x_M \cup feature_i))}{C(n-1, |M|)} \right], \tag{2.6}$$

em que:

- $f(x)$ é a predição do modelo para a observação x,
- $f(x_M \cup feature_i)$ é a predição do modelo quando o subconjunto de variáveis S é combinado com a $feature_i$,
- $C(n-1, |M|)$ representa o número de combinações possíveis que incluem a $feature_i$,
- A soma externa é feita em cima de todos os valores possíveis de subconjuntos M que não incluem a $feature_i$,
- A soma interna é feita sobre todas as combinações possíveis de M.

Em resumo, o valor SHAP mede a contribuição marginal média de cada variável em todas os possíveis subconjuntos de variáveis. Basicamente, apontando o quanto a inclusão dessa variável no modelo contribui para a diferença entre a previsão da observação e a previsão esperada, ou seja, a previsão média do modelo em todas as combinações possíveis de variáveis.

Lundberg e Lee (2017) apresentaram uma maneira consistente e aditiva de explicar previsões individuais, conseguindo quantificar a contribuição de cada variável. Nessa abordagem fundamentada na teoria dos jogos cooperativos, os cálculos garantem que a contribuição de cada variável seja atribuída com precisão e de forma que respeite as interações com outras variáveis.

2.1.1 SHAP para seleção de variáveis

Os valores SHAP também podem ser usados como uma técnica de seleção para identificar as variáveis mais importantes para uma determinada tarefa.

Para usar esses valores para seleção de variáveis, uma abordagem é classificá-las com base em seus SHAP absolutos médios em um grande número de amostras, esse valores são

calculados simplesmente fazendo uma média dos valores obtidos em cada variável. As variáveis com valores SHAP absolutos médios mais altos são consideradas mais importantes para as previsões do modelo. Outra abordagem é usar um limite para selecionar as variáveis com os valores SHAP absolutos mais altos, ou seja, definir a quantidade de variáveis que deseja ter no modelo.

Procedimentos comuns para seleção de variáveis costumam usar as importâncias fornecidas por um modelo, como os coeficientes de uma regressão linear ou as pontuações de importância de variáveis no caso de modelos baseados em árvore como *Random Forest* ou *XGBoost*. No entanto, esses métodos têm algumas limitações. Por exemplo, os modelos lineares assumem que a relação entre a variável resposta e uma ou mais variáveis indepedentes é linear nos parâmetros, o que pode não ser o caso na realidade. Os modelos baseados em árvore podem capturar relacionamentos não lineares, mas suas pontuações de importância de variável podem ser tendenciosas em relação àquelas que estão correlacionadas com a variável de destino.

Os valores SHAP fornecem um método flexível e preciso para calcular as importâncias das variáveis e podem ser usadas como uma técnica de seleção para identificar as mais importantes. Esse cálculo têm várias vantagens sobre outros métodos de importância de variáveis, como não assumir linearidade e conseguir lidar com possíveis interações entre variáveis, o que se mostrou eficaz em vários estudos.

2.2 Lasso

O *Least Absolute Shrinkage and Selection Operator* (Lasso) é um método popular para seleção de variáveis em aprendizado de máquina e estatística. É uma técnica de regressão linear que adiciona um termo de regularização à função objetivo. O termo de regularização é um parâmetro de penalidade que controla a complexidade do modelo e evita o *overfitting* ao reduzir os coeficientes de recursos menos importantes para zero.

O modelo de regressão linear Lasso é definido por:

$$\hat{y} = \beta_0 + \sum_{j=1}^{p} \beta_j x_j, \tag{2.7}$$

onde $\hat{y}$ é a variável de resposta prevista, x_j é a j-ésima variável preditora j^{th}, β_j é o coeficiente correspondente e β_0 é o intercepto.

Esse modelo de regressão assume uma relação linear entre as variáveis independentes e a variável resposta (ou dependente), o que pode não ser verdadeiro para todos os conjuntos de dados. A regressão logística, embora seja um caso particular do modelo linear generalizado (MLG), continua sendo um modelo linear em seus parâmetros. A inclusão de covariáveis polinomiais e termos de interação permite capturar relações complexas entre as variáveis preditoras, mas isso não transforma o modelo em não linear. Ele permanece linear nos coeficientes, o que significa

que, apesar de incorporar essas características, a estrutura do modelo mantém sua linearidade. Uma história detalhada da regressão logistica pode ser encontrada em Cramer (2002).

Em modelos de regressão linear, é modelada a média da variável resposta em função de um conjunto de variáveis independentes organizadas de forma linear, o chamado preditor linear. Na regressão logística mais simples, a variável resposta tem distribuição Bernoulli, o que implica na média da variável resposta ser a probabilidade de sucesso (p). Como 0 < p < 1 e o preditor linear varia nos reais, transformamos a probabilidade p na escala do preditor linear e modelo de regressão logística faz essa transformação através da função logit.

Enquanto a regressão linear modela resultados contínuos, a regressão logística modela resultados binários. A regressão linear assume uma relação linear entre as variáveis independentes e dependentes, enquanto a regressão logística modela as probabilidades logarítmicas da probabilidade da variável dependente.

Na regressão linear, com ligação identidade os coeficientes representam a mudança na média da variável dependente para uma mudança de uma unidade na variável independente. Na regressão logística, os coeficientes representam a mudança na função da média da variável dependente para uma mudança de uma unidade na variável independente.

Na regressão logística, a relação entre a variável dependente e as variáveis independentes é modelada usando a função logística, também conhecida como função sigmóide. A função logística transforma a combinação linear das variáveis independentes na escala de [0, 1], representando a probabilidade de sucesso de um resultado binário.

A função logito, denotada como $logit(p)$, é definida como:

$$logit(p) = \ln\left(\frac{p}{1-p}\right), \tag{2.8}$$

onde p representa a probabilidade de sucesso do resultado binário e ln denota o logaritmo natural.

A função logit mapeia a probabilidade p para a escala *log-odds*, que varia nos reais. Esta transformação é necessária porque a regressão linear assume um resultado contínuo, o que não é adequado para resultados binários.

Na regressão logística, a função logit é usada para modelar a relação entre as variáveis independentes e as probabilidades do resultado binário. O preditor linear, denotado como η, é a combinação linear das variáveis independentes:

$$\eta = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p, \tag{2.9}$$

onde:

- β_0 é o termo de intercepto,

- $\beta_1, \beta_2, \ldots, \beta_p$ são os coeficientes associados às variáveis independentes $(x_1, x_2, \ldots, x_p)$ respectivamente.

A função logit é aplicada ao preditor linear (η) para obter as probabilidades logarítmicas previstas do resultado binário:

$$logit(p) = \eta = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p. \tag{2.10}$$

A probabilidade prevista do resultado binário é então obtida aplicando o inverso da função logit, conhecida como função logística:

$$p = \frac{1}{1 + e^{-\eta}}, \tag{2.11}$$

onde e é o número de *Euler*.

A regressão logística funciona estimando os coeficientes $(\beta_0, \beta_1, \ldots, \beta_p)$ que maximizam a probabilidade de observar os resultados binários reais, dados os valores das variáveis independentes. Essa estimativa normalmente é feita usando algoritmos de otimização, como estimativa de máxima verossimilhança ou gradiente descendente. Uma vez estimados os coeficientes, eles podem ser usados para prever a probabilidade do resultado binário para novas observações com base nos valores das variáveis independentes.

Na seleção de recursos Lasso, é adicionada uma penalidade L1 à função objetivo a ser minimizada durante o treinamento do modelo. Essa penalidade incentiva o modelo a definir alguns dos coeficientes como zero, realizando efetivamente a seleção de variáveis.

A função objetivo a ser minimizada é dada por:

$$min_{\beta_0,\beta} \left\{ \frac{1}{2n} \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij})^2 + \alpha \sum_{j=1}^{p} |\beta_j| \right\}, \tag{2.12}$$

onde:

- n é o número de observações nos dados de treinamento,
- y_i é a variável de resposta observada para a i-ésima observação,
- x_{ij} é o valor observado da j-ésima variável preditora para a i-ésima observação,
- α é o parâmetro de regularização que controla a força da penalidade,
- $\sum_{j=1}^{p} |\beta_j|$ é o termo de penalidade L1.

A penalidade L1 encoraja alguns dos coeficientes a serem exatamente iguais a zero, o que corresponde a realizar a seleção de variáveis. Isso ocorre porque a penalidade L1 resulta em uma região de restrição em forma de "diamante" no espaço de coeficientes, onde as restrições

interceptam os eixos. Os cantos desta região em forma de diamante correspondem aos casos em que alguns dos coeficientes são iguais a zero. O Lasso basicamente minimiza a soma dos resíduos quadrados e o termo de penalidade.

A força da penalidade L1 é controlada pelo parâmetro de regularização α. Valores maiores de α resultam em penalidades mais fortes, que por sua vez resultam em mais coeficientes sendo zerados. Valores menores de α resultam em penalidades mais fracas, que por sua vez resultam na retenção de mais coeficientes. O valor ideal de α pode ser determinado usando validação cruzada ou outras técnicas de seleção de modelo.

Durante o treinamento do modelo, o algoritmo Lasso minimiza a função objetivo em relação aos coeficientes β_j, sujeito à restrição L1. Este problema de otimização pode ser resolvido usando vários algoritmos de otimização, como descida de coordenadas ou descida de gradiente.

O algoritmo Lasso tem várias vantagens sobre outras técnicas de seleção de variáveis. Uma das principais é poder manipular dados de alta dimensão com muitas variáveis, mesmo quando o número de amostras é menor que o número de variáveis. Isso ocorre porque o Lasso é um problema de otimização convexo que pode ser resolvido de forma eficiente usando vários algoritmos de otimização numérica. Ele pode lidar com conjuntos de dados com muitas variáveis correlacionadas e seleciona automaticamente um subconjunto das mais relevantes. Além disso, a dispersão do vetor de peso torna o modelo mais interpretável e reduz o risco de *overfitting*.

No entanto, o algoritmo Lasso possui algumas limitações, como a tendência de selecionar apenas uma variável de um grupo de variáveis altamente correlacionadas e a dificuldade de selecionar variáveis na presença de dados ruidosos, ou seja, dados que possuem algum tipo de erro e, ou uma certa variabilidade em determinadas variáveis.

O Lasso está intimamente relacionado à regressão rígida (*ridge*), que também adiciona um termo de penalidade à função de custo. No entanto, o termo de penalidade na regressão rígida é a soma dos valores quadrados dos coeficientes de regressão, em vez dos valores absolutos. O efeito desse termo de penalidade é reduzir os coeficientes de regressão para zero, mas não defini-los exatamente como zero. Isso significa que a regressão de Ridge não realiza uma seleção de variáveis.

Em *Python*, o Lasso pode ser implementado usando a biblioteca *scikit-learn*, que fornece uma classe Lasso que pode ser usada para regressão e seleção de variáveis. Nesse pacote é possível passar um parâmetro de regularização e ajustar o modelo aos dados. Em seguida, através do atributo $Lasso.coef\_$, são encontrados os coeficientes do modelo Lasso, conseguindo assim ver quais variáveis foram efetivamente selecionadas pelo modelo.

No geral, o algoritmo Lasso é um método poderoso para seleção de recursos, que pode melhorar a precisão e a interpretabilidade dos modelos de aprendizado de máquina.

2.3 Modelos Aplicados

Os modelos de *Boosting* foram os que melhor performaram nos ajustes iniciais da base de dados, portanto foi escolhido partir deles para as análises e comparação.

2.3.1 Extreme Gradient Boosting

XGBoost (eXtreme Gradient Boosting) é um algoritmo popular de aprendizado de máquina usado para problemas de regressão e classificação. Proposto por Chen e Guestrin (2016), é um método de aprendizado conjunto que combina várias árvores de decisão para fazer previsões.

É possível descrever como esse modelo se comporta começando pelo primeiro passo, no qual o *XGBoost* cria árvores de decisão. Uma árvore de decisão é um modelo semelhante a uma árvore em que cada nó interno representa um teste em uma variável, cada ramificação representa o resultado do teste e cada nó folha representa um rótulo de classe. O objetivo é criar um conjunto de árvores de decisão que possam prever a variável alvo com precisão.

Esse método utiliza ainda a técnica de *boosting*, que combina vários modelos fracos (árvores de decisão neste caso) para criar um modelo forte. O *boosting* funciona treinando uma série de árvores de decisão, cada uma tentando corrigir os erros cometidos pelas árvores anteriores. A previsão final é a soma ponderada das previsões de todas as árvores.

Uma variante de *boosting*, chamada aumento de gradiente, também é utilizada pelo *XGBoost*. No aumento de gradiente, cada nova árvore é treinada nos erros residuais das árvores anteriores. Isso significa que cada nova árvore tenta prever os erros residuais das árvores anteriores. Este processo continua até que o erro não possa mais ser reduzido ou um número especificado de árvores seja alcançado, esse valor pode ser parametrizado na criação do modelo.

O *XGBoost* também usa técnicas de regularização para evitar o *overfitting*, ela envolve a adição de um termo de penalidade à função de perda, incentivando o modelo a preferir modelos mais simples. O *XGBoost* usa dois tipos de regularização, a regularização L1 (Lasso) e a regularização L2 (*ridge*).

O *XGBoost* fornece um método para medir a importância de cada variável no modelo. Essa pontuação de importância é calculada com base em quantas vezes cada variável é utilizada para fazer uma divisão nas árvores de decisão e quanto ganho cada quebra está fornecendo. Para avaliar o desempenho do modelo *XGBoost* é muito utilizada a validação cruzada.

Além disso, é fácil integrar a metodologia SHAP ao algoritmo *XGBoost*, dado que foi um dos primeiros em *Python* a ser disponibilizado na biblioteca *shap*, fornecendo assim uma maneira de explicar as previsões de um modelo atribuindo-as a diferentes variáveis de entrada, além da possibilidade de utilizar as visualizações do pacote *shap* para entender a contribuição de cada variável em cada observação do modelo.

2.3.2 Light Gradient Boosting Machine

O *Light Gradient Boosting Machine* (*LightGBM*) é um método de aumento de gradiente popular e eficiente desenvolvido pela *Microsoft* (KE *et al.*, 2017). Ele foi projetado para alcançar alto desempenho e escalabilidade ao lidar com conjuntos de dados de grande escala. O *LightGBM* usa o algoritmo de aumento de gradiente, como o *XGBoost*, que combina sequencialmente modelos de previsão fracos para criar um modelo de previsão forte. O processo de treinamento minimiza uma função de perda adicionando iterativamente modelos fracos ao conjunto.

Esse método suporta várias funções de perda para diferentes tipos de problemas, como regressão, classificação binária e classificação multiclasse. A função objetivo quantifica a diferença entre os valores previstos e os alvos reais, o erro quadrático médio (MSE) funciona como um exemplo de função objetivo para regressão e entropia cruzada como um exemplo para classificação.

Durante o treinamento, o *LightGBM* otimiza o modelo de conjunto ajustando iterativamente modelos fracos aos gradientes negativos da função de perda. Ele usa uma abordagem de descida de gradiente ponderada, onde cada modelo fraco é ajustado aos gradientes negativos em relação às previsões do conjunto atual, o que vai melhorando gradualmente as previsões do modelo com o passar das iterações.

Esse método utiliza uma abordagem baseada em histograma para discretização de variáveis. Ele divide cada variável em compartimentos discretos, construindo histogramas de valores de variáveis. Ao encontrar pontos de divisão ideais nos histogramas, o *LightGBM* determina os melhores limites para a divisão de variáveis durante a construção da árvore. Essa abordagem reduz a complexidade de tempo e o uso de memória em comparação com outros métodos que exigem a classificação de todos os pontos de dados.

A *Gradient-based One-Side Sampling* (GOSS) é uma técnica usada no *LightGBM* para acelerar o treinamento. Ele amostra seletivamente as observações com base em seus gradientes para focar em amostras informativas e executa amostragem aleatória nas menos informativas, descartando algumas. Ao fazer isso, o método GOSS equilibra a distribuição dos dados. Essa estratégia ajuda a reduzir o custo computacional ao processar apenas uma fração dos dados.

O *Light Gradient Boosting Machine* também fornece opções de paralelização para computação eficiente. Ele suporta a construção de árvores paralelas, onde várias árvores podem ser construídas simultaneamente, explorando os recursos computacionais disponíveis. Além disso, o *LightGBM* pode distribuir os dados em várias máquinas para treinamento, permitindo um processamento mais rápido de grandes conjuntos de dados, sendo esse um dos principais recursos do *LightGBM*. Ao adotar uma abordagem baseada em histograma para discretização de variáveis, ele permite um treinamento mais rápido e menor consumo de memória em comparação com outras estruturas de aumento de gradiente.

Em termos de desempenho, o *LightGBM* demonstrou resultados competitivos em várias

tarefas de aprendizado de máquina, Fan *et al.* (2019) utilizaram esse método como um modelo de computação flexível eficiente para estimar a evapotranspiração de referência diária com dados meteorológicos locais e externos. Uma versão otimizada desse método foi utilizado por Taha e Malebary (2020) como uma abordagem inteligente para detecção de fraude de cartão de crédito.

Esse algoritmo atinge alta precisão construindo modelos de conjunto fortes por meio de *boosting* e emprega técnicas como regularização e parada antecipada para evitar *overfitting*. Seus recursos exclusivos, como discretização de variáveis baseada em histograma e amostragem baseada em gradiente, contribuem para um treinamento mais rápido e menor consumo de memória.

2.3.3 Gradient Boosting Classifier

O *Gradient Boosting Classifier* é um algoritmo de aprendizado de máquina popular que também pertence à família de aumento de gradiente. É amplamente utilizado para tarefas de classificação e tem demonstrado alto desempenho em vários domínios. Friedman (2001) propôs esse método em um artigo onde apresenta a estrutura geral de *Gradient Boosting* e discute especificamente sua aplicação a problemas de classificação binária. Desde então, o algoritmo ganhou popularidade significativa e foi desenvolvido e refinado pela comunidade de aprendizado de máquina. Zhang e Haghani (2015) utilizou esse método para melhorar a previsão do tempo de viagem, Friedman (2002) propôs uma abordagem estocástica de aumento de gradiente, obtendo resultados que indicavam que a precisão do aumento de gradiente pode ser substancialmente melhorada pela introdução de aleatoriedade ao treinar o modelo base em diferentes subconjuntos de dados selecionados em cada iteração.

Como os outros métodos de *boosting*, o GBC constrói o modelo de *ensemble* adicionando iterativamente classificadores fracos, com cada classificador subsequente visando corrigir os erros cometidos pelos anteriores. O algoritmo começa com uma previsão inicial, que pode ser uma estimativa simples, como a média dos valores alvo ou um valor constante e esse valor serve como base para iterações subsequentes.

Em cada nova iteração, o *Gradient Boosting Classifier* treina um classificador fraco nos dados de treinamento. O classificador fraco é tipicamente uma árvore de decisão com pouca profundidade ou um toco (uma árvore com apenas uma divisão). O classificador fraco é treinado para prever os erros ou resíduos cometidos pelo *ensemble* de classificadores nas iterações anteriores.

O algoritmo então calcula os gradientes, representando a direção e a magnitude dos erros, da função de perda em relação às previsões feitas pelo conjunto até o momento. A função de perda específica usada depende do tipo de problema em questão.

O classificador fraco é adicionado ao conjunto multiplicando-o por uma taxa de aprendizado ou um fator de encolhimento para controlar a sua contribuição. Esse dimensionamento

permite uma convergência mais suave e uma melhor generalização. Por fim, as previsões do conjunto são atualizadas adicionando as previsões ponderadas do novo classificador fraco.

O processo iterativo continua até que um número especificado de classificadores fracos (árvores) seja treinado ou um critério de parada predefinido seja atendido, ambos podem ser definidos na criação da classe do modelo. O critério de parada pode ser baseado, por exemplo, no número máximo de iterações, na melhoria da função de perda ou no desempenho em um conjunto de validação.

A previsão final do *Gradient Boosting Classifier* é obtida combinando as previsões de todos os classificadores fracos no conjunto. Para classificação binária, a predição pode ser obtida usando um limite, enquanto para classificação multiclasse, técnicas como *softmax* (GOODFELLOW; BENGIO; COURVILLE, 2016) ou *one vs rest* (BISHOP; NASRABADI, 2006) podem ser usadas.

O *Gradient Boosting Classifier*, como outros modelos de aumento de gradiente, é conhecido por sua capacidade de lidar com relacionamentos complexos entre as covariáveis e variáveis alvo e por ser um bom modelo contra *overfitting*. Ele consegue capturar padrões não lineares nos dados e lidar efetivamente com espaços de variáveis de alta dimensão.

No entanto, é importante ajustar os hiperparâmetros, como a taxa de aprendizado, a profundidade da árvore e os parâmetros de regularização para otimizar o desempenho do *Gradient Boosting Classifier*, o pacote *sklearn* permite alterar livremente cada um desses hiperparâmetros. Além disso, o algoritmo pode exigir mais recursos computacionais e tempo de treinamento em comparação com classificadores mais simples devido à sua natureza iterativa.

2.4 Análise dos Modelos

2.4.1 Matriz de confusão e Acurácia

Uma matriz de confusão é uma tabela usada para avaliar o desempenho de um modelo de classificação de aprendizado de máquina. É uma ferramenta simples, mas poderosa, que resume os rótulos de classe previstos e reais de um problema de classificação.

Esse método consiste em quatro resultados diferentes:

- Verdadeiro Positivo (TP): O modelo previu corretamente que uma observação pertence à classe positiva.
- Falso Positivo (FP): O modelo previu que uma observação pertence à classe positiva, mas na verdade ela pertence à classe negativa.
- Verdadeiro Negativo (TN): O modelo previu corretamente que uma observação pertence à classe negativa.

- Falso Negativo (FN): O modelo previu que uma observação pertence à classe negativa, mas na verdade ela pertence à classe positiva.

Geralmente é apresentada em um formato de tabela com os rótulos das classes previstas nas colunas e os rótulos das classes reais nas linhas.

A partir dessa matriz, várias métricas de desempenho podem ser calculadas, como acurácia, previsão, especificidade, sensibilidade e pontuação F1, que são medidas de desempenho construídas a partir da teoria de probabilidade condicional. Por exemplo, a acurácia (a) é calculada como $(TP+TN)/(TP+FP+TN+FN)$, a precisão (pr) como $TP/(TP+FP)$, a especificidade (e) como $TN/(TN+FP)$, a sensibilidade (s) como $TP/(TP+FN)$ e a pontuação F1 é como $(2*pr*s)/(pr+s)$. Na literatura, a sensibilidade também é denominada recall ou revocação, enquanto a precisão também pode ser denominada valor preditivo positivo (VPP).

As matrizes de confusão também podem ser visualizadas usando várias técnicas, como mapas de calor ou gráficos de barras empilhadas, para entender melhor o desempenho do modelo de classificação.

2.4.2 Curva ROC

A curva Receiver Operating Characteristic (ROC) é uma representação gráfica comumente usada em tarefas de classificação binária para avaliar o desempenho de um classificador de aprendizado de máquina. Ilustra o compromisso entre a taxa de verdadeiros positivos (sensibilidade) e a taxa de falsos positivos (1 - especificidade) em diferentes limites de decisão. Essa curva foi utilizada neste trabalho como uma das formas de avaliação dos resultados dos modelos.

A curva ROC é criada traçando a taxa de verdadeiros positivos (sensibilidade) contra a taxa de falsos positivos (1 - especificidade) para diferentes limites de decisão. Cada ponto da curva representa o desempenho do classificador em um limite específico. Uma linha diagonal de (0,0) a (1,1) representa uma adivinhação aleatória, enquanto uma curva que se curva em direção ao canto superior esquerdo indica melhor desempenho.

A área sob a curva ROC (AUC-ROC) quantifica o desempenho geral do classificador em todos os limites de decisão possíveis. Varia de 0 a 1, onde o valor 1 indica desempenho de classificação perfeito e o valor 0,5 representa adivinhação aleatória. Uma pontuação AUC-ROC mais alta indica melhor discriminação entre instâncias positivas e negativas.

Em resumo, a curva ROC fornece uma visualização abrangente do desempenho de um classificador, permitindo que os profissionais avaliem sua capacidade de distinguir entre instâncias positivas e negativas em diferentes limites de decisão.

2.4.3 Score da média geométrica

O score de média geométrica, também conhecida como média geométrica das pontuações de classificação, é outra métrica de avaliação comumente usada em aprendizado de máquina, especialmente para problemas de classificação desbalanceados, o que a tornou uma métrica interessante de ser analisada neste trabalho. Ao contrário da média aritmética, que é a soma dos valores dividida pelo número de valores, a média geométrica é a enésima raiz do produto de n valores. No contexto da classificação, é calculado com base na média harmônica de sensibilidade (taxa de verdadeiro positivo) e especificidade (taxa de verdadeiro negativo).

O score média geométrica fornece uma medida equilibrada do desempenho de um classificador em classes positivas e negativas. Leva em conta tanto a sensibilidade como a especificidade, tornando-o adequado para conjuntos de dados desbalanceados onde uma classe pode ter uma quantidade bem superior a outra. Uma pontuação média geométrica mais alta indica melhor desempenho geral da classificação.

2.4.4 Gráficos do método SHAP

SHAP (SHapley Additive exPlanations) é um pacote que fornece uma variedade de ferramentas de visualização para interpretar as previsões de modelos de aprendizado de máquina. Serão utilizados uma série de gráficos para fazer uma análise e compreensão dos modelos, procurando por relações entre variáveis e/ou observações *outliers*.

O *dependence plot* é gráfico de dependência que mostra a relação entre uma única variável e as previsões do modelo. O eixo x representa os valores do recurso e o eixo y representa os valores SHAP. Cada ponto representa uma amostra no conjunto de dados e a cor do ponto representa o valor de outra variável que pode estar interagindo com a variável de interesse. Os gráficos de dependência são úteis para entender a direção e a força da relação entre as variáveis e as previsões do modelo.

O gráfico de cascata ou *waterfall* é um gráfico que expõe como cada variável contribui para a previsão do modelo para uma única amostra. O gráfico começa com o valor esperado da previsão do modelo e, em seguida, mostra o impacto de cada variável na previsão, com contribuições positivas mostradas em azul e contribuições negativas mostradas em vermelho. Gráficos de cascata são úteis para entender a importância relativa de cada variável para uma única previsão.

O *summary plot* traz a importância global de cada variável em todo o conjunto de dados. O gráfico classifica as variáveis por seu valor SHAP absoluto médio e mostra a distribuição dos valores SHAP para cada uma. O *summary plot* é útil para identificar as variáveis mais importantes para as previsões do modelo.

O gráfico de *interaction values* possui os valores de interação e mostra como duas variáveis interagem para influenciar as previsões do modelo. O eixo x representa os valores

SHAP para uma variável e o eixo y representa os valores SHAP para o outra. Cada ponto representa uma amostra no conjunto de dados e a cor do ponto representa o valor da previsão do modelo. Os gráficos de valores de interação são úteis para entender como as previsões do modelo dependem das interações entre os recursos.

No geral, o SHAP fornece uma variedade de ferramentas de visualização que podem ajudar a interpretar as previsões dos modelos de aprendizado de máquina, e esses gráficos podem ser usados para obter informações sobre o relacionamento entre as variáveis e as previsões do modelo.

2.4.5 Recursos Computacionais

2.4.5.1 Google Colab

O advento da computação em nuvem no início do século 21 foi significativo na evolução dos recursos computacionais. Soluções baseadas em nuvem, como Google Workspace, Microsoft 365 e Salesforce, ofereceram escalabilidade, flexibilidade e recursos colaborativos, permitindo trabalho remoto e colaboração em tempo real entre equipes geograficamente dispersas. Assim os recursos computacionais utilizados nas aplicações apresentadas, foram diretos e de fácil alcance.

Ao longo desta pesquisa de dissertação, o Google Colab foi uma ferramenta fundamental para tarefas de programação e análise de dados. Google Colab é uma plataforma baseada em nuvem fornecida pelo Google, oferecendo um ambiente conveniente e colaborativo para executar códigos em Python, executar modelos de aprendizado de máquina e analisar dados usando bibliotecas populares como NumPy e Pandas. A integração da plataforma com o Google Drive permite acesso contínuo a arquivos e conjuntos de dados, facilitando a colaboração eficiente e o controle de versões entre pesquisadores e membros da equipe.

Uma das principais vantagens do Google Colab é o fornecimento de recursos gratuitos de GPU e TPU (Tensor Processing Unit), permitindo computação acelerada para tarefas de aprendizado de máquina. Aproveitando esses recursos, algoritmos complexos e modelos de aprendizagem profunda podem ser treinados de forma mais rápida e eficiente em comparação com a computação tradicional baseada em CPU,. Vale ressaltar que esses métodos mais robustos não foram necessários nas aplicações realizadas, utilizando apenas CPU, os tempos de execução foram razoáveis. Além disso, o Google Colab fornece acesso a ambientes com muita RAM, o que é particularmente vantajoso para lidar com grandes conjuntos de dados e operações com uso intensivo de memória.

O ambiente de tempo de execução padrão, que foi o utilizado para as aplicações, normalmente inclui um único núcleo de CPU e uma quantidade moderada de RAM, adequado para tarefas básicas de programação e análise leve de dados (12,5 GB).

A facilidade de uso da plataforma, agilizou o processo de pesquisa e facilitou a exploração de tarefas computacionais complexas. Além disso, a integração com o Google Drive proporcionou

acesso contínuo a dados e ferramentas de colaboração, aumentando a produtividade e a eficiência do trabalho.

2.4.5.2 Pycaret

O pacote PyCaret (ALI, 2020) serviu como ferramenta principal para a aplicação e análise de modelos de aprendizado de máquina neste projeto de pesquisa. PyCaret é uma biblioteca de aprendizado de máquina de código aberto em Python, projetada para agilizar o fluxo de trabalho, desde a preparação de dados até a implantação do modelo. Uma das principais vantagens do PyCaret é sua simplicidade e facilidade de uso, permitindo que pesquisadores e profissionais criem protótipos, comparem e ajustem rapidamente vários modelos de aprendizado de máquina com esforço mínimo de codificação.

O fluxo de trabalho normalmente começa com o carregamento do conjunto de dados em um Pandas DataFrame, após o qual o PyCaret lida automaticamente com valores ausentes, codificação categórica e dimensionamento de variáveis. Este pré-processamento automatizado garante consistência e reprodutibilidade em diferentes experimentos.

Após o pré-processamento, o PyCaret facilita a criação de modelos de aprendizado de máquina por meio de uma interface simples e intuitiva. Os usuários podem escolher entre uma ampla variedade de algoritmos, incluindo modelos lineares, modelos baseados em árvore, métodos de conjunto e algoritmos de aumento de gradiente. PyCaret fornece uma API unificada para treinamento de modelo, facilitando a comparação do desempenho de diferentes algoritmos em um determinado conjunto de dados.

Para avaliação e seleção de modelo, PyCaret oferece um conjunto de funções para gerar métricas importantes de desempenho, como acurácia, especificidade, sensibilidade, pontuação F1 e área sob a curva ROC (AUC-ROC). Além disso, o PyCaret automatiza o processo de validação cruzada, permitindo aos usuários avaliar o desempenho de generalização de modelos em múltiplas dobras dos dados.

Os gráficos da curva de aprendizagem ajudaram a avaliar a convergência do modelo e a identificar possíveis problemas, como *overfitting* ou *underfitting*, enquanto os gráficos da matriz de confusão forneceram uma visão do desempenho da classificação do modelo em diferentes classes. No geral, o PyCaret facilitou a experimentação e análise eficientes de modelos de aprendizado de máquina, contribuindo para a conclusão.

2.4.5.3 Tempos de execução

Com as configurações e pacotes definidos acima, as tabelas Tabela 1, Tabela 2 apresentam os tempos de execuções gastos nos principais processos de análise e aplicação dos modelos, primeiramente para a base de dados simulada e em sequência para a base de dados real. Mais detalhes dos modelos apresentados serão discutidos no Capítulo 3.

Na base de dados simulada o tempo para seleção de variáveis foi maior utilizando o método SHAP, o tempo de aplicação dos modelos também foi maior comparado ao Lasso, mas ambos foram mais rápidos do que a aplicação sem seleção.

Tabela 1 – Tempo de execução, em segundos, dos processos em base simulada

Processos	Sem seleção	Seleção método Lasso	Seleção método SHAP
Seleção de Variáveis	0	15.20	32,85
Light Gradient Boosting Machine	17,67	4,1	5,16
Extreme Gradient Boosting	19,50	4,98	5,01
Gradient Boosting Classifier	28,25	7,4	7,91

Fonte: Dados da pesquisa.

Para a base de dados real, o tempo de seleção foi bem maior no método SHAP, mas na aplicação dos modelos ele foi mais rápido que o método Lasso.

Tabela 2 – Tempo de execução, em segundos, dos processos em base real

Processos	Sem seleção	Seleção método Lasso	Seleção método SHAP
Seleção de Variáveis	0	152.19	880.35
Light Gradient Boosting Machine	19.55	11.36	8.63
Extreme Gradient Boosting	103.95	38.50	16,09
Gradient Boosting Classifier	104.74	58.79	26.03

CAPÍTULO

3

APLICAÇÃO

3.1 Aplicação em base de dados Simulada

Primeiramente foi feita uma aplicação em um banco de dados simulado para compreender, em um ambiente controlado, as propostas de comparação. Uma maneira popular de criar um banco de dados é usar a função *make classification* em Python.

A função *make classification* faz parte do módulo *sklearn.datasets* e pode ser usada para gerar um problema aleatório de classificação multiclasse ou binária. Ele permite especificar o número de amostras, número de variáveis, número de classes, número de variáveis informativas, número de variáveis redundantes e número de *clusters* por classe, ou seja, os possíveis subgrupos dentro de uma das categorias da variável alvo.

Então é possível usar a função para gerar um banco de dados com as propriedades desejadas. Para este trabalho foi criado um banco de dados com 5.000 unidades amostrais, 300 variáveis, nomeadas de *"Feature_i"* para i de 1 a 300, sendo dessas 100 variáveis informativas, 20 variáveis redundantes que são combinações lineares das 100 informativas e portanto 180 variáveis irrelevantes. A variável dependente possui 2 classes com proporções de 40% e 60% .

A função inicialmente cria *clusters* de pontos normalmente distribuídos (std=1) sobre os vértices de um hipercubo com dimensão igual ao número de variáveis informativas com lados de comprimento $2 * class\_step$, fator que multiplica o tamanho do hipercubo. Valores maiores espalham os *clusters*/classes e facilitam a tarefa de classificação; e atribui um número igual de *clusters* para cada classe. Ele introduz a interdependência entre essas variáveis e adiciona vários tipos de ruído adicional aos dados. A matriz X resultante contém as variáveis de entrada para cada amostra, enquanto a variável y contém os rótulos de classe de destino correspondentes.

No geral, usar a função *make classification* em Python é uma maneira conveniente de gerar dados sintéticos para teste e experimentação. Ao ajustar os parâmetros da função, podem ser criados bancos de dados com uma ampla gama de propriedades e características, tornando-a

uma ferramenta útil para pesquisa e desenvolvimento de aprendizado de máquina.

Depois de criar um conjunto de dados sintético, foram ajustados algoritmos de aprendizado de máquina para construir modelos preditivos sobre esses dados. Em geral foram mais utilizados algoritmos de árvores e *boosting*. Um dos principais é o *XGBoost*, que é um poderoso algoritmo de aumento de gradiente usado para problemas de classificação e regressão.

Para melhorar o desempenho dos modelos aplicados e reduzir o *overfitting*, entram as técnicas de seleção de variáveis para identificar aquelas mais importantes no conjunto de dados. Neste caso, foram utilizados dois métodos diferentes para seleção de variáveis: valores SHAP e o método Lasso.

Os valores SHAP classificaram as variáveis com base em sua importância média e assim foi possível selecionar as principais para o modelo, definindo um *threshold* de importância mínima a ser selecionada. Ao fazer isso, é possível reduzir a dimensionalidade do conjunto de dados e melhorar o desempenho do modelo.

O método Lasso é usado para seleção de atributos ao penalizar os coeficientes das variáveis menos importantes, resultando em um modelo esparso com apenas as variáveis mais importantes. Com ele foram selecionadas as principais variáveis com base em seus coeficientes e aplicadas nos modelos escolhidos.

Depois de aplicar os dois métodos de seleção de variáveis, foi possível comparar o desempenho dos ajustes com o conjunto de dados completo versus ambos conjuntos de dados reduzidos. Os modelos foram avaliados usando métricas como acurácia, precisão , sensibilidade e F1-score. Procurando identificar se o desempenho do conjunto de dados reduzido é semelhante ao do conjunto de dados completo, assim as variáveis selecionados seriam suficientes para o modelo prever novas observações.

Importante destacar que este capítulo foi apresentado em formato de pôster no *Latin American Congresso f Probability and Mathematical Statistics, 2023* , que ocorreu em São Paulo entre os dias 10 e 14 de julho de 2023, gerando uma publicação em Cunha e Russo (2023)

3.1.1 Sem seleção de variáveis

Os modelos foram escolhidos a partir de uma comparação feita utilizando a função *compare models* do pacote *pycaret*, ela traz várias opções de modelos a partir de uma dada base de dados. Essa função separa a base em treino (70%) e teste (30%), ajusta os modelos utilizando *Cross Validation*, nesse caso com 5 *folds*, possibilitando também obter os resultados na base de treino.

Os três modelos que melhor performaram no ajuste com base sem seleção de variáveis na base de treino estão mostrados na Tabela 3, eles serão então utilizados para comparar a performance das metodologias de seleção de variáveis.

Também foram testados os modelos: Ridge Classifier, Ada Boost, Random Forecast, Logistic Regression, Extra Trees, Decision Tree, SVM, Naive Bayes, K Neighbors, Quadratic Discriminant Analysis. Mas não obtiveram resultados interessantes para serem analisados a fundo neste trabalho.

Tabela 3 – Comparação de Modelos sem seleção de Variáveis

Modelo	Acurácia	AUC	Sensibilidade	Precisão	F1
Extreme Gradient Boosting	88.91%	94.83%	94.76%	87.75%	91.11%
Light Gradient Boosting Machine	88.29%	95.13%	94.28%	87.29%	90.62%
Gradient Boosting Classifier	83.71%	91.50%	92.85%	82.30%	87.24%

Fonte: Dados da pesquisa.

O gráfico da curva de aprendizado é uma ferramenta usada no aprendizado de máquina para visualizar o desempenho de um modelo à medida que o tamanho do conjunto de dados de treinamento aumenta. Ele mostra a relação entre o tamanho do conjunto de treinamento e a métrica de desempenho, como precisão ou taxa de erro. O gráfico é útil para entender o desempenho de um modelo e para identificar quando um modelo atingiu seu potencial máximo.

Esse gráfico mede o desempenho através do método de validação cruzada ou *CrossValidation*. O objetivo principal da validação cruzada é avaliar a capacidade de um modelo de generalizar para dados novos e não vistos. O processo envolve particionar o conjunto de dados em vários subconjuntos ou dobras. O modelo é treinado em um subconjunto de dados e validado nos dados restantes, utilizando uma métrica desejada, dependendo se é uma tarefa de regressão, classificação ou outro tipo, normalmente são utilizadas valores de acurácia, F1, entre outros. Este processo é repetido várias vezes, com diferentes subconjuntos usados para treinamento e validação em cada iteração. A forma mais comum de validação cruzada é a validação cruzada k-fold, onde o conjunto de dados é dividido em k subconjuntos e o modelo é treinado e avaliado k vezes, cada vez usando uma dobra diferente para validação. A média dos resultados é então calculada para fornecer uma estimativa mais robusta do desempenho do modelo.

A validação cruzada ajuda a identificar problemas potenciais, como *overfitting* ou subajuste, fornecendo uma avaliação mais precisa do desempenho do modelo em dados invisíveis.

A curva de aprendizado tem o tamanho do conjunto de treinamento no eixo x e a métrica de desempenho no eixo y, nesse caso a acurácia. O gráfico geralmente tem duas linhas - uma para as pontuações do treinamento e outra para as pontuações do teste. A linha de pontuações de treinamento mostra o desempenho do modelo no conjunto de treinamento à medida que o tamanho do conjunto de treinamento aumenta. A linha de pontuações de teste mostra como o modelo funciona em um conjunto de teste separado à medida que o tamanho do conjunto de treinamento aumenta.

Ao aumentar o tamanho do conjunto de treinamento, o desempenho do modelo nesse conjunto geralmente diminui, pois o modelo tem mais dados para aprender. No entanto, o

desempenho no conjunto de teste geralmente aumenta junto com o tamanho do conjunto de treinamento, pois o modelo tem mais dados para generalizar. O objetivo é encontrar o ponto onde o desempenho no conjunto de teste é mais alto e o desempenho no conjunto de treinamento está próximo da convergência. Isso indica que o modelo aprendeu tudo o que pode com os dados e adicionar mais dados não melhorará seu desempenho.

A Figura 3 traz o gráfico de curva de aprendizado para o melhor modelo sem seleção de variáveis. A linha de treinamento não chega a cair conforme a quantidade de observações vai aumentando, o modelo está prevendo um score alto para essa base de dados, isso é um indício de *overfitting* na base de treino. No conjunto de testes a curva vai aumentando conforme o conjunto de treino aumenta.

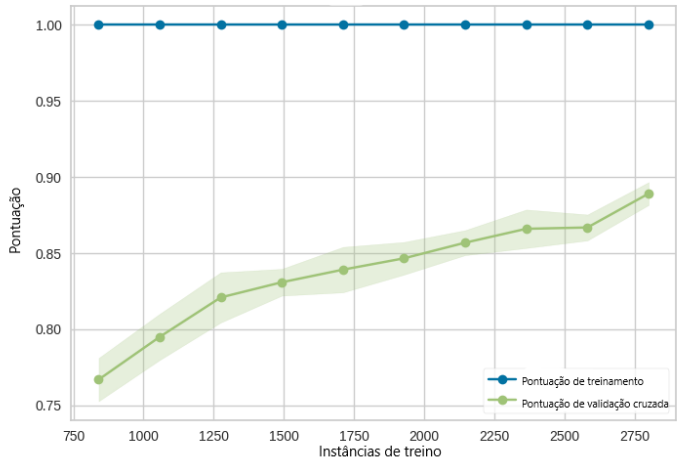


Figura 3 – Curva de aprendizado para o modelo sem seleção de *features*

Fonte: Elaborado pela autora.

(Adaptado para português pela autora)

Depois do treino o modelo foi ajustado diretamente nas bases de treino e teste obtendo uma matriz de confusão, usada para avaliar o desempenho de modelos de classificação de aprendizado de máquina. A Figura 4 apresenta os resultados obtidos e com isso obtemos os valores para a acurácia como sendo 89.53%, uma precisão de 88.73% e sensibilidade de 94.55%. Gerando um $F1$ de 91.55%.

Após o ajuste do modelo, o pacote SHAP foi aplicado levantando as principais variáveis relevantes para o modelo. Para isso é apresentado o gráfico de *summary plot* (Figura 5) que traz a importância global de cada recurso em todo o conjunto de dados. Esse gráfico mostra os valores de importância da variável para cada observação no modelo, com as variáveis ordenados do mais ao menos importante.

Neste gráfico a cor representa a variação de escala de cada variável (baixa ou alta) e qual o impacto que cada valor tem na saída do modelo (eixo x). A interpretação dos resultados do *summary plot* pode nos ajudar a entender quais variáveis são mais importantes para as previsões

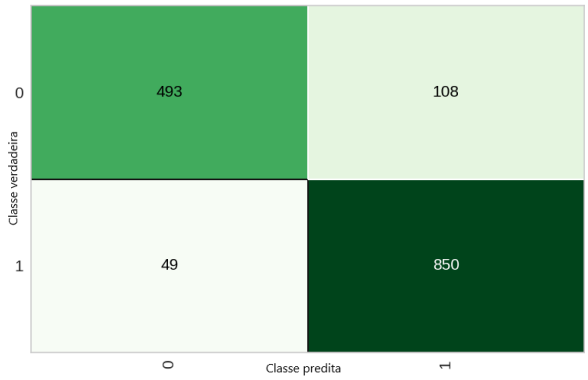


Figura 4 – Matriz de confusão para o modelo sem seleção de *features*

Fonte: Elaborado pela autora.

(Adaptado para português pela autora)

do modelo. As variáveis com um valor SHAP alto e uma barra larga são as mais importantes, enquanto as variáveis com um valor SHAP baixo e uma barra estreita são as menos importantes.

Com isso é possível notar que a variável 137 se mostra a mais relevante para o modelo, onde valores altos dessa variável tem um alto impacto negativo para o modelo, levando a predição para 0 e valores baixos tem um impacto positivo menor, levando a predição a 1. Não foi possível identificar se as variáveis escolhidas pelo método estavam entre as 100 informativas definidas no começo da simulação.

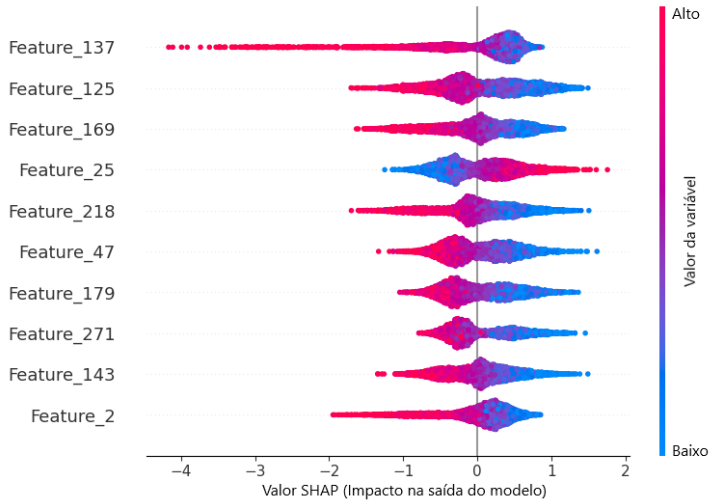


Figura 5 – SHAP Summary Plot para o modelo sem seleção *features*

Fonte: Elaborado pela autora.

(Adaptado para português pela autora)

3.1.2 Com seleção de variáveis pelo método Lasso

Para aplicar o método LASSO para uma seleção de variáveis é importante escolher o valor apropriado para o parâmetro de penalidade, representado por α. Assim o método foi aplicado utilizando a função *LassoCV* com 5 *folds*, variando os valores de α de 0.01 a 1 em intervalos de 0.01. Foi escolhido o parâmetro que trouxe a maior acurácia para a regressão ajustada, para isso, fixado um valor de alpha foi obtida a probabilidade resultante da regressão e o valor foi arredondado para 0 ou 1, com um ponto de corte em 0.5. A Figura 6 traz em verde os resultados obtidos para essa escolha, mostrando também quantas variáveis permaneceram no modelo com coeficiente diferente de 0, em cinza e eixo z.

É possível notar que o primeiro corte com α pequeno já seleciona menos de 100 variáveis para o modelo, a acurácia chega a subir um pouco nas primeiras iterações mas perto de 0.1 ela começa a cair.

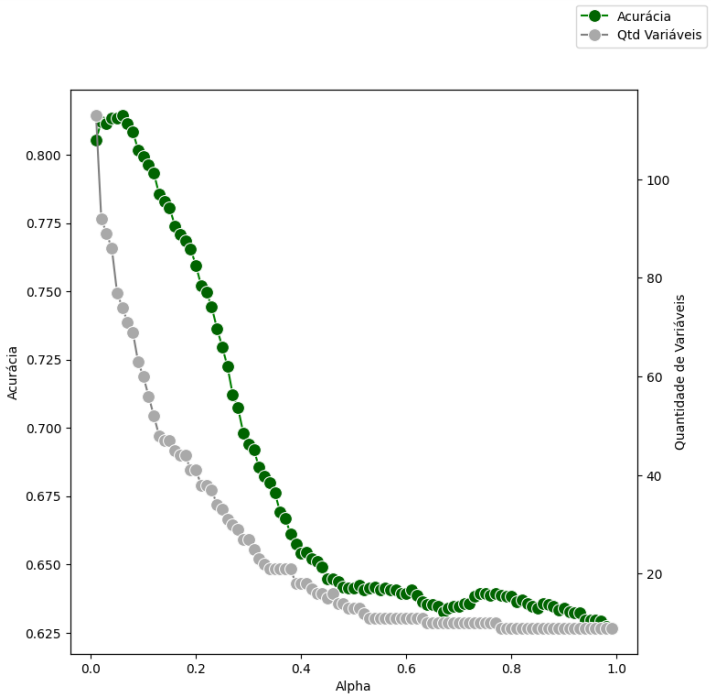


Figura 6 – Escolha do valor de α para o LASSO

Fonte: Elaborado pela autora.

Escolhido o parâmetro de regularização como $\alpha = 0.06$, foi possível ajustar os 3 modelos escolhidos com as 74 variáveis selecionadas, lembrando que dentre as 300 disponíveis existem 100 informativas e 20 variáveis redundantes. Aqui, também não foi possível identificar a natureza das 74 variáveis selecionadas, se são informativas, reduntantes ou irrelevantes. Os resultados do *Cross Validation* com 5 *folds* são apresentados na Tabela 4 a seguir.

Para o melhor modelo escolhido foi feito o gráfico de curva de aprendizado representado na Figura 7. Muito parecida com o modelo ajustado sem seleção de *features* é possível notar

Tabela 4 – Comparação de Modelos com seleção de Variáveis Lasso

Modelo	Acurácia	AUC	Sensibilidade	Precisão	F1
Light Gradient Boosting Machine	88.03%	94.80%	93.52%	87.42%	90.35%
Extreme Gradient Boosting	87.74%	94.65%	93.09%	87.33%	90.10%
Gradient Boosting Classifier	83.37%	91.19%	92.80%	81.89%	87.00%

Fonte: Dados da pesquisa.

apenas que a curva em treino não chega tão perto de 90% quanto o modelo anterior.

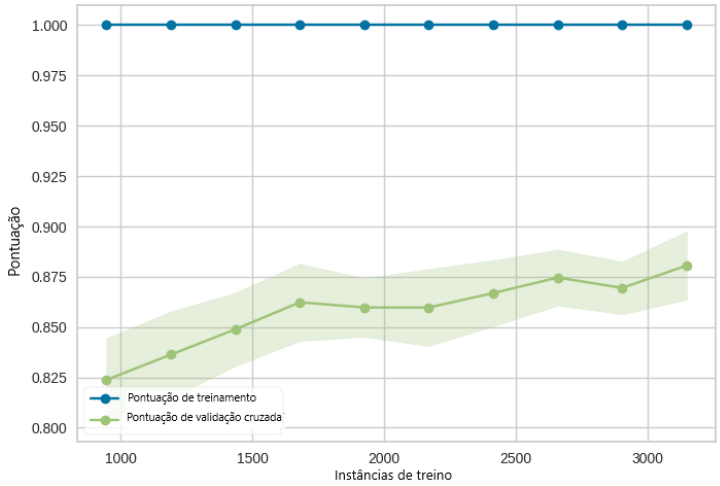


Figura 7 – Curva de aprendizado para o modelo com seleção de *features* pelo LASSO

Fonte: Elaborado pela autora.

Esse modelo com seleção pelo LASSO também foi ajustado diretamente na base de teste obtendo uma matriz de confusão. A Figura 8 apresenta os resultados obtidos e com isso obtemos os valores para a acurácia como sendo $88,67\%$, uma precisão de $88,82\%$ e sensibilidade de $92,77\%$. Gerando um $F1$ de $90,75\%$. Abaixo dos valores gerados pelo modelo com todas as variáveis, indicando que talvez a seleção não tenha sido tão eficaz.

Na Figura 9 é apresentado o *summary plot* encontrado a partir dessa seleção de variáveis, em que é possível notar que ele possui várias das mesmas variáveis do modelo anterior. Mas o destaque ocorre na troca de ordenação de relevância, há várias trocas, em especial no modelo sem seleção a variável 125 se encontrava no segundo lugar do gráfico, para o modelo utilizando LASSO a variável 25 se torna mais relevante, com um alto impacto negativo para valores altos e um alto impacto positivo em valores menores. A *Feature_125* ainda se encontra presente no gráfico, mas na terceira posição.

Vale notar que esse summary plot é equivalente ao mostrado anterior, algumas vezes o pacote altera o estilo automaticamente para ajudar na visualização e compreensão do modelo. Essa visualização não é tão informativa quanto a outra forma do gráfico, aqui se peder a densidade.

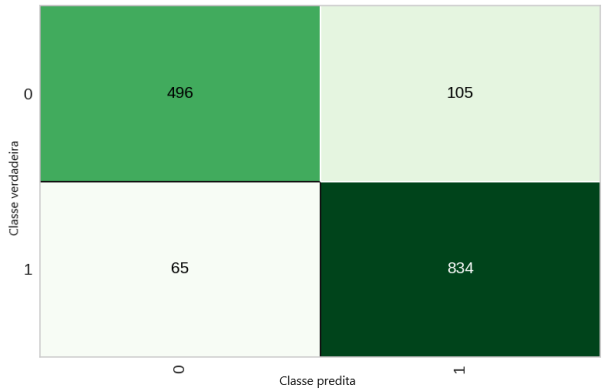


Figura 8 – Matriz de confusão para o modelo com seleção de *features* pelo LASSO

Fonte: Elaborado pela autora.

(Adaptado para português pela autora)

A largura da barra nesse caso representa o peso que cada variável terá para cada predição.

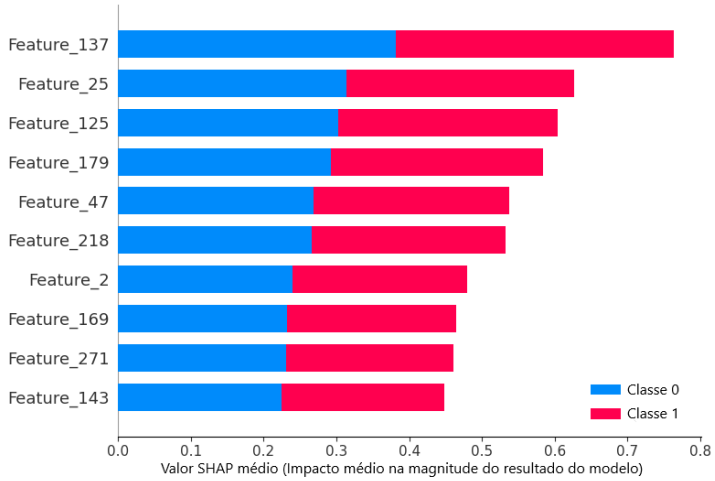


Figura 9 – SHAP Summary Plot para o modelo com seleção de *features* via LASSO

Fonte: Elaborado pela autora.

(Adaptado para português pela autora)

3.1.3 Com seleção de variáveis pelo método SHAP

Existem algumas formas diferentes de se montar uma seleção de variáveis utilizando os valores SHAP, para esse trabalho foi escolhido ordenar as variáveis pelo seu valor SHAP absoluto médio e selecionar aquelas acima de um valor de *threshold* definido. Então foram necessários alguns testes para escolher esse limite.

Esses testes ocorreram da seguinte forma, foi ajustado um modelo *XGBoost* e encontradas as importâncias através do pacote SHAP. Então dado um *threshold* foram selecionadas as

variáveis que possuiam importância acima do valor desejado. Por fim, um modelo *XGBoost* era novamente aplicado utilizando as variáveis selecionadas. O *threshold* que trouxe maior acurácia no modelo final aplicado foi selecionado.

A Figura 10 traz os resultados desses testes. Em azul temos as acurácias encontradas para o *threshold* variando entre os valores: 0.01, 0.03, 0.05, 0.07, 0.09, 0.1, 0.2. Esses números foram escolhidos com base em análises e também para garantir uma importância mínima de 1% para cada *feature*. Em vermelho relativo ao eixo z está mostrado a quantidade de variáveis selecionadas para cada valor.

O primeiro corte em 1% já seleciona apenas metade do total disponível (cerca de 180 variáveis), a partir dele com os cortes mais agressivos a acurácia sobe e depois começa a cair, sendo que acima de 10% ela fica abaixo dos valores encontrados pelos modelos anteriores.

Também já é possível notar nesse gráfico a comparação com os modelos anteriores, para 4 pontos de seleção do SHAP a acurácia está acima do modelo sem seleção e para 5 pontos está acima do modelo com seleção via método LASSO.

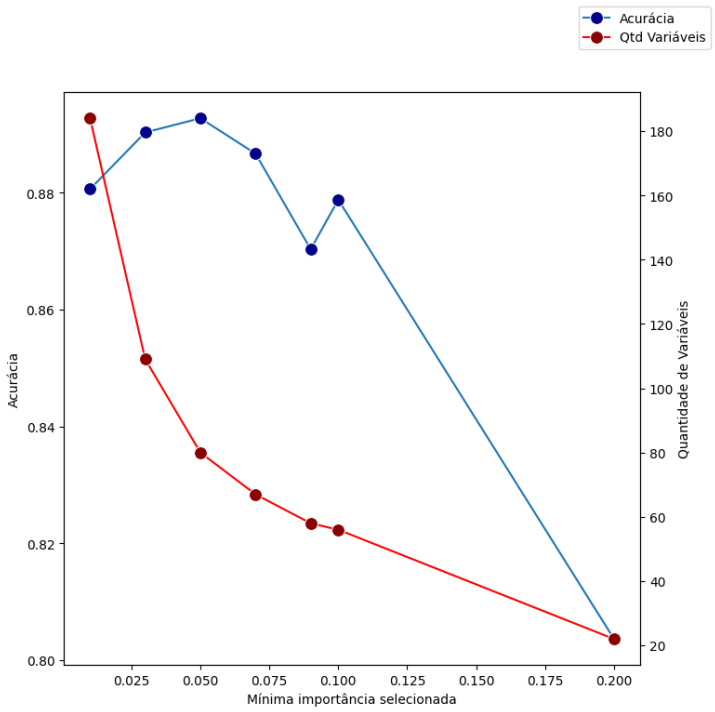


Figura 10 – Escolha do valor da mínima importância de variável para seleção

Fonte: Elaborado pela autora.

O *threshold* foi definido em 5%, com o SHAP escolhendo 80 variáveis. Esse valor se mostra mais próximo da quantidade de variáveis influentes no modelo definidas na criação da base de dados (100 informativas e 20 redundantes). Assim, foram aplicados os 3 modelos na base com 80 variáveis, em um *Cross Validation* com 5 *folds*, obtendo os valores apresentados na Tabela 5 a seguir.

Tabela 5 – Comparação de Modelos com seleção de Variáveis pelo SHAP

Modelo	Acurácia	AUC	Sensibilidade	Precisão	F1
Extreme Gradient Boosting	89.17%	95.74%	94.04%	88.64%	91.24%
Light Gradient Boosting Machine	88.77%	95.52%	94.33%	87.89%	90.97%
Gradient Boosting Classifier	84.34%	91.84%	93.18%	82.90%	87.72%

Fonte: Dados da pesquisa.

A Figura 11 traz a curva de aprendizado para o melhor modelo ajustado com as variáveis selecionadas pelo SHAP, é possível notar que a curva alcança valores altos antes de utilizar toda a base de treino, já com bases a partir de 2250 observações.

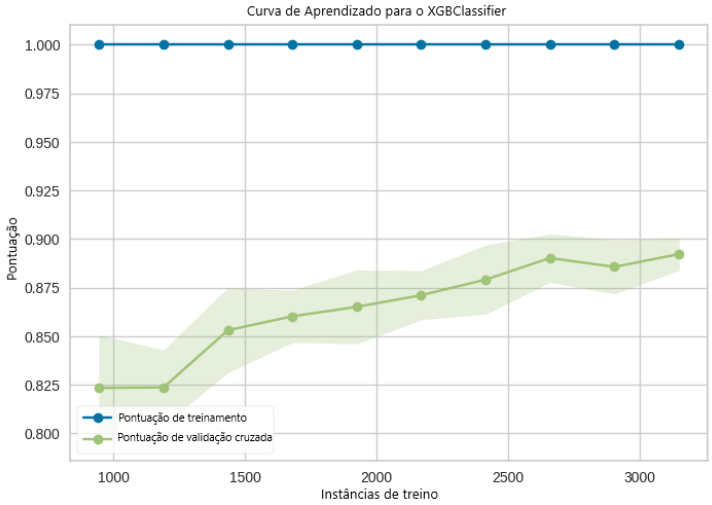


Figura 11 – Curva de aprendizado para o modelo com seleção de *features* pelo SHAP

Fonte: Elaborado pela autora.

(Adaptado para português pela autora)

A matriz de confusão encontrada nesse ajuste também se mostrou a melhor entre as opções. Apresentada na Figura 12 ela traz uma acurácia de 89.93% uma precisão de 89.45% e sensibilidade de 94.33%, alcançando um $F1$ de 91.82%.

A Tabela 6 traz os valores comparados para os modelos completos, com seleção Lasso e com seleção SHAP. É possível notar que a acurácia e precisão do modelo com seleção via SHAP é a maior entre os 3, com uma leve queda na sensibilidade quando comparado com o modelo sem seleção de variáveis. Mas o $F1$, métrica que traz o peso entre a relação de precisão e sensibilidade teve um aumento de 1% em relação ao método anterior. Além do ganho de acerto, um ajuste com menos variáveis traz consigo um tempo de execução menor, ou seja, depois de selecionada as variáveis de interesse o tempo de execução do modelo cai pela metade nesse exemplo.

A partir do modelo ajustado com valores SHAP é possível tirar outros *insighs* com os

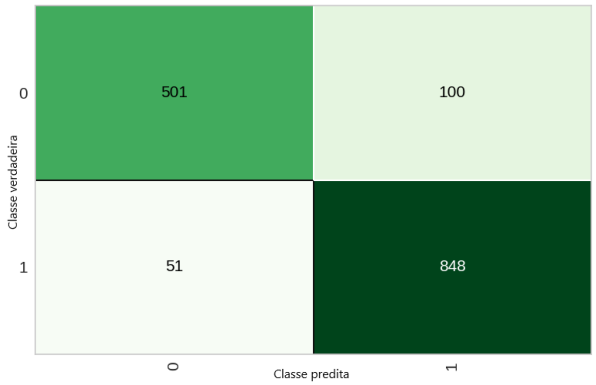


Figura 12 – Matriz de confusão para o modelo com seleção de *features* pelo SHAP

Fonte: Elaborado pela autora.

(Adaptado para português pela autora)

Tabela 6 – Resultados dos ajuste dos modelos em base simulada

Resultados em Teste	Sem seleção	Seleção método LASSO	Seleção método SHAP
Quantidade de variáveis	300	74	80
Acurácia	89.53%	88.67%	89.93%
Precisão	88.73%	88.82%	89.45%
Sensibilidade	94.55%	92.77%	94.33%
F1-Score	91.55%	90.75%	91.82%
Escore média geométrica	0.881	0.875	0.887

Fonte: Dados da pesquisa.

gráficos de importância de variáveis, começando com o *summary plot* já comentado anteriormente para os outros modelos. Da Figura 13, nota-se que a variável 137 permanece como a mais importante para o modelo e a 125 manteve seu segundo lugar, mas também ocorreram algumas trocas na ordenação e foram adicionadas algumas variáveis.

A Figura 14 apresenta os gráficos de dependência ou *"Dependence Plots"* do SHAP, gráficos de dispersão que mostram o efeito que uma única variável tem nas previsões feitas pelo modelo. O eixo x representa os valores do recurso e o eixo y representa os valores SHAP. Cada ponto representa uma amostra no conjunto de dados e a cor do ponto representa o valor de outra variável que pode estar interagindo com a variável de interesse. O gráfico foi feito para a variável 137 e o pacote selecionou automaticamente a variável 16 como uma possível interação no modelo. É possível notar que quanto maior os valores para a 137, menores são os valores SHAP, e o impacto é negativo para valores acima de 0. Com a variável 16 é difícil notar alguma relação direta, mas aparenta ser inversa, ou seja, para valores até 0 da variável 137, altos valores da 16 levam a uma chance menor do modelo prever 1 e acima de 0 valores altos da 16 levam a SHAP maior a chance da previsão ser 1.

Também é possível comentar dos valores de interação SHAP e, para calculá-los, cada

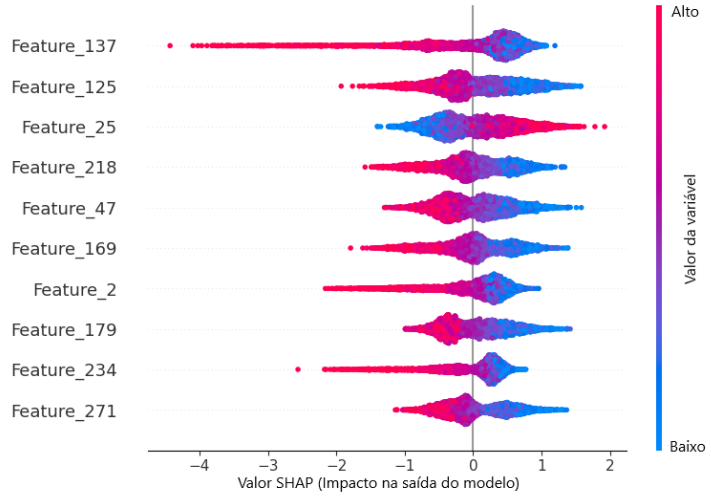


Figura 13 – SHAP Summary Plot para o modelo com seleção de *features* via SHAP

Fonte: Elaborado pela autora.

(Adaptado para português pela autora)

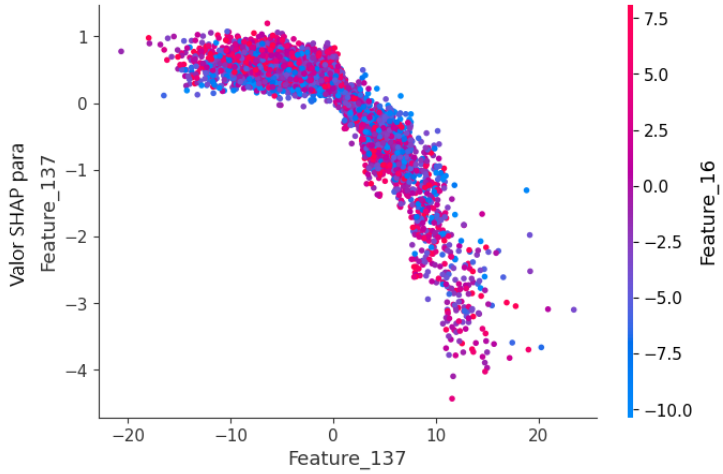


Figura 14 – *Dependence Plot* para o modelo com seleção de *features* via SHAP

Fonte: Elaborado pela autora.

(Adaptado para português pela autora)

contribuição é dividida em contribuições principais e de interação. A computação desses valores retorna uma matriz para cada previsão, em que os efeitos principais estão na diagonal e os efeitos de interação estão fora da diagonal. Os efeitos principais são semelhantes aos valores SHAP que seriam obtidos para um modelo linear, e os efeitos de interação capturam todas as interações de ordem superior e as dividem entre os termos de interação em pares. Observa-se que a soma de toda a matriz de interação é a diferença entre a saída atual do modelo e a saída esperada e, portanto, os efeitos da interação na diagonal são divididos ao meio (já que existem dois de cada).

Ao colocar os efeitos de interação em um gráfico, o pacote SHAP multiplica automaticamente os valores fora da diagonal por dois para obter o efeito de interação completo (LUNDBERG *et al.*, 2020). A Figura 15, portanto, possui os valores de interação e mostra como duas variáveis interagem para influenciar as previsões do modelo. O eixo x representa os valores SHAP para uma variável e o eixo y representa os valores SHAP para o outra. Cada ponto representa uma amostra no conjunto de dados e a cor do ponto representa o valor da previsão do modelo. No caso da base de simulação há poucas possíveis interações entre as variáveis, o mais relevante é uma pequena interação entre as variáveis 137 e 2.

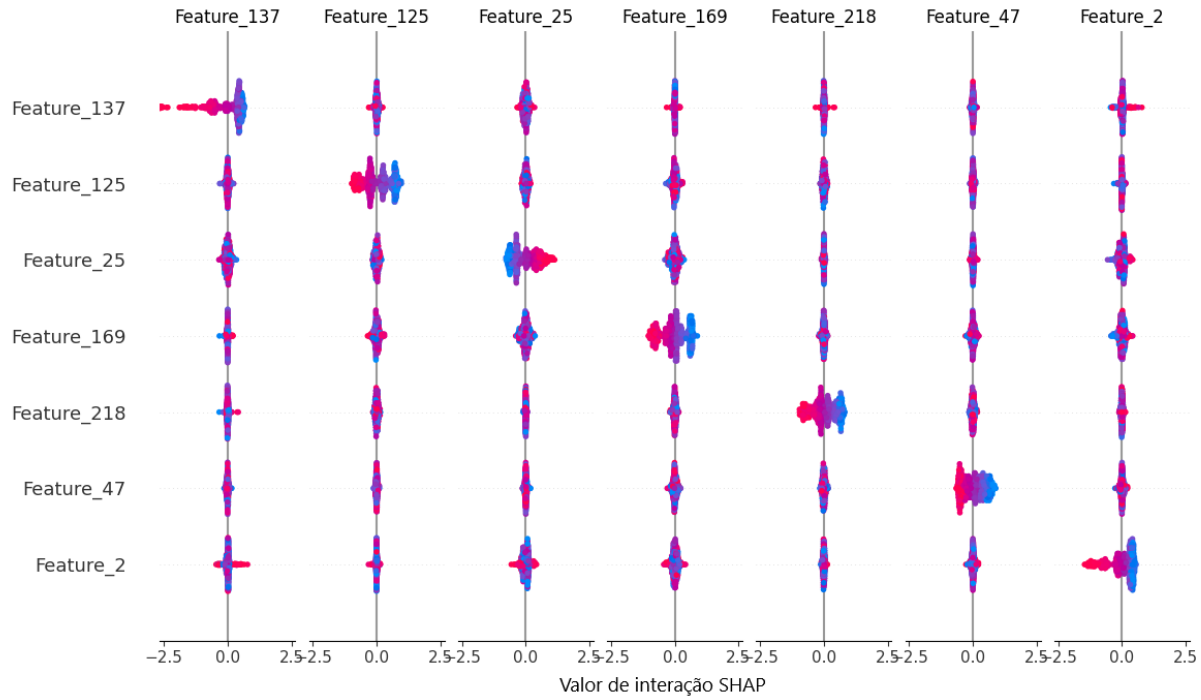


Figura 15 – *SHAP Interation Value* para XGBoost com seleção de *features* via SHAP

Fonte: Elaborado pela autora.

(Adaptado para português pela autora)

3.2 Aplicação em Base Real

3.2.1 Base de dados - Lending Club

O banco de dados (KAGGLE, 2022) é composto de empréstimos feitos no *Lending Club* entre 2007 e 2020, possui 142 variáveis relacionadas aos usuários, assim como o status do empréstimo (pago, atrasado, não pago, etc) que será o alvo da predição. No total são 1.879.204 observações, cada uma representando uma transação já finalizada, mas nem todas foram utilizadas por uma questão computacional, para ajustar um modelo a uma base como essa é necessário uma máquina com muita memória e com um bom processamento, o que vai além de um computador

comum. Portanto algumas decisões foram tomadas para limpar e arrumar a base de forma que um modelo possa ser ajustado em um computador convencional.

A descrição completa de cada variável se encontra no Anexo 1 e, a seguir, estão mais detalhes da variável alvo. O resultados dos empréstimos são mostrados na Tabela 7 onde é observado que a base de dados é desbalanceada, ou seja, há uma concentração de resultados na classe “Totalmente Pago", que se refere a empréstimos já finalizados e completamente pagos.

Tabela 7 – Valores Gerais Variável Alvo

Status do Empréstimo	Observações
Totalmente Pago	1497783
Não pago	362548
Atrasado (31-120 dias)	16154
Atrasado (16-30 dias)	2719
Corrente	1031016
Omisso	433
Não atende à política de crédito. Status: Cobrado	761
Não atende à política de crédito. Status: Cobrado	1988
Em período de carência	10028
Emitido	2062

Fonte: Dados da pesquisa.

(Adaptado para o português pela autora)

Como são de interesse apenas o status nas transações finalizadas, objetivando encontrar predições para “Será Pago"ou “Não será pago", foram utilizadas apenas observações com resultado como *“Totalmente Pago"* e *“Não pago"*, além disso, as transações atrasadas, *“Atrasado (31-120 dias)"* e *“Atrasado (16-30 dias)"*, foram consideradas como *“Não pago"*, já que elas também representam resultados negativos para o empréstimo, e ajudarão com o desbalanceamento da base.

Em seguida foram analisadas as variáveis em si, e, com uma pesquisa em trabalhos anteriores, foram encontradas algumas colunas que seriam altamente correlacionadas com a variável alvo, gerando *“leak"* no modelo, ou seja, um possível vazamento de informações incorreto, dados ao qual o modelo em uma predição futura não teria acesso, ou uma variável independente altamente correlacionada com a variável alvo. com isso foram removidas as colunas: *“out_prncp"; “out_prncp_inv"; “total_pymnt"; “total_pymnt_inv"; “total_rec_prncp"; “total_rec_int"; “total_rec_late_fee"; “recoveries"; “collection_recovery_fee"; “last_pymnt_d"; “last_pymnt_amnt"; “next_pymnt_d"; “last_credit_pull_d"; “debt_settlement_flag"*.

Uma análise de quantidade de valores nulos mostrou que algumas variáveis possuíam um percentual muito alto de observações faltantes, com isso foram removidas todas as variáveis com mais de 60% de valores nulos. Além disso foi feita uma análise de correlação entre as covariáveis quantitativas e removidas uma variável de cada par com mais de 0.6 de correlação entre si .

Duas variáveis, “issue_d"e “earliest_cr_line", são do tipo “datetime", com isso foi necessário um tratamento para transformá-las em covariáveis utilizáveis, elas foram alteradas para representarem a quantidade de dias entre a data em questão e o dia atual.

Mesmo com todos os tratamentos anteriores, a base ainda era relativamente grande para o nível de processamento disponível, com isso foi decidido remover observações que ainda continham algum valor nulo ou faltante, utilizando a função *dropna*, em vez de buscar algum método de preenchimento de dados faltantes.

Após os ajustes, a base de dados ficou com 358.669 observações, onde 78% são empréstimos “Totalmente Pago"e 22% “Não Pago". Assim gerou-se algumas análises gerais para entender o comportamento da variável alvo. A própria empresa *LendingClub* já possui uma classificação entre bons e maus pagadores, representada no banco por duas variáveis *“Grade"* e *“Subgrade"*, onde A simboliza a melhor classe e G a última, além disso, cada classe é quebrada em níveis de 1 a 5.

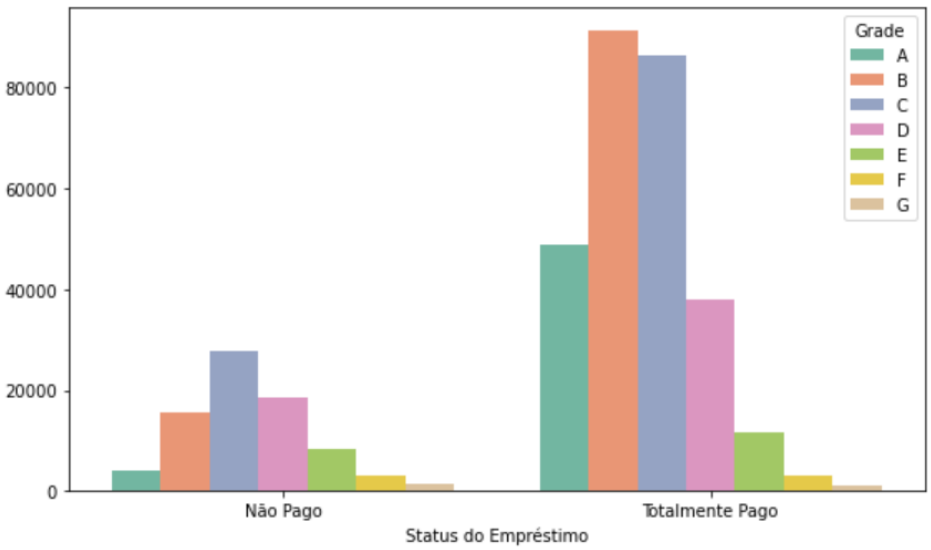


Figura 16 – Grupos definidos da variável alvo por Grade

Fonte: Elaborado pela autora.

A Figura 16 mostra a relação entre a variável alvo e a primeira classificação, nela é possível notar que os empréstimos concluídos com sucesso estão concentrados nas classes A, B, C e D. Além disso as categorias F e G possuem quase a mesma quantidade de *“Totalmente Pago"* e *“Não Pago"*, é possível se aprofundar nelas utilizando a variável subgrade.

A Figura 17 traz um detalhamento das classes F e G em relação ao resultado de empréstimos. Na classe G, a quantidade de *“Não Pago"* ultrapassa a quantidade de *“Totalmente Pago"*, o gráfico da direita ainda mostra especificamente em quais sub-classes os maus pagadores ultrapassam a quantidade de bons pagadores. Como essas variáveis são feitas a partir de resultados de empréstimos anteriores na plataforma, elas são correlacionadas com a predição e causariam

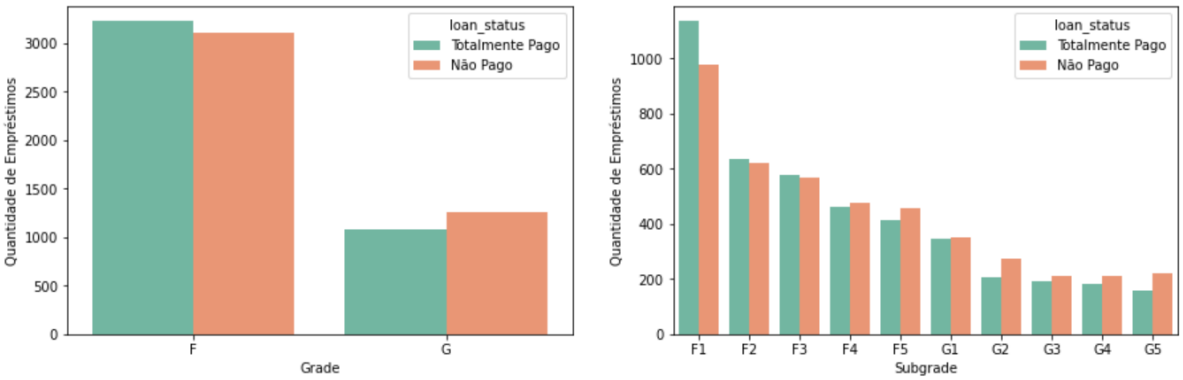


Figura 17 – Grupos definidos da variável alvo por Grade e Subgrade

Fonte: Elaborado pela autora.

leak, como explicado anteriormente, na predição, portanto elas foram removidas da base antes do ajuste dos modelos.

Assim a base de dados está limpa e bem definida de forma que é possível seguir com um ajuste de modelo. Além disso, já foi possível conhecer um pouco do comportamento geral dessa base de dados, em específico da resposta alvo.

Os modelos testados para essa base foram modelos de aprendizado de máquina sem seleção de *features*, com seleção via método LASSO e com seleção pelo SHAP. Como a maioria desses programas e pacotes não comportam variáveis categóricas, foi necessário utilizar a função *get_dummies* nas colunas: *"verification_status"*; *"purpose"*; *"initial_list_status"*; *"application_type"*; *"home_ownership"*; *"emp_length"*; *"addr_state"*; *"hardship_flag"*. Além disso, as variáveis "*term*"e "*int_rate*"foram tratadas para serem utilizadas como variáveis quantitativas.

O conjunto de dados foi dividido em bases de treino e teste, com este representando 30% do total e mantendo uma proporção aproximada da base completa (78% vs 22%). É importante notar que como a classe dominante (*Totalmente Pago*) possui 78% das observações, um bom modelo deve ter uma acurácia maior que esse valor. Por fim, a variável alvo foi transformada em 0, 1, onde o estado *Totalmente Pago* é representado pelo valor 1.

3.2.2 Sem seleção de variáveis

Para gerar alguns processamentos dos modelos foi necessário colher uma amostra da quantidade de observações total (aproximadamente 350 mil, aqui chamada de base completa) por quesitos computacionais. Assim foi selecionado 20% da base (aproximadamente 70 mil), separado em treino (70%) e teste (30%), aqui chamada de base de amostra, para escolher quais modelos seriam testados e para definir tanto o parâmetro de regularização do LASSO, quanto o *threshold* de importância das variáveis para o SHAP. Essa amostra foi selecionada de forma estratificada, ou seja, os percentuais das classes da variável alvo se mantiveram. Em seguida,

com os modelos e parâmetros escolhidos, a base completa de 350 mil foi utilizada dividida em treino e teste e utilizada para calcular as métricas preditivas.

Comparando modelos através do pacote *pycaret*, com um *Cross Validation* de 5 *folds* e utilizando os dados da amostra, foram escolhidos os seguintes modelos por terem maior acurácia *Light Gradient Boosting Machine* (LGBM), *Gradient Boosting Classifier* e *Linear Discriminant Analysis*. Além deles, foi também ajustado o modelo *Extreme Gradient Boosting* para comparação, por ser um modelo bem relacionado com o pacote SHAP.

Em seguida, foram aplicados os modelos na base completa de 350 mil observações, para a escolha do melhor modelo ajustado, o modelo de *Linear Discriminant Analysis* ficou pior entre os 4 modelos para todos os ajustes na base completa, então não foi detalhado nesse trabalho. A Tabela 8 traz os resultados desse ajuste, trazendo o LGBM como o melhor modelo escolhido.

Tabela 8 – Comparação de Modelos Sem seleção de Variáveis

Modelo	Acurácia	AUC	Sensibilidade	Precisão	F1
Light Gradient Boosting Machine	91.04%	95.39%	94.02%	94.49%	94.25%
Gradient Boosting Classifier	90.94%	95.26%	93.96%	94.42%	94.19%
Extreme Gradient Boosting	90.91%	95.23%	94.09%	94.26%	94.18%
Linear Discriminant Analysis	90.82%	95.12%	94.10%	94.16%	94.13%

Fonte: Dados da pesquisa.

A Figura 18 traz a curva de aprendizado do melhor modelo sem seleção de variáveis ajustado. Como esperado, em uma base real, a acurácia na base de treino decai conforme a quantidade de observações aumenta, na base de teste ela chega a aumentar, especialmente a partir de 30 mil.

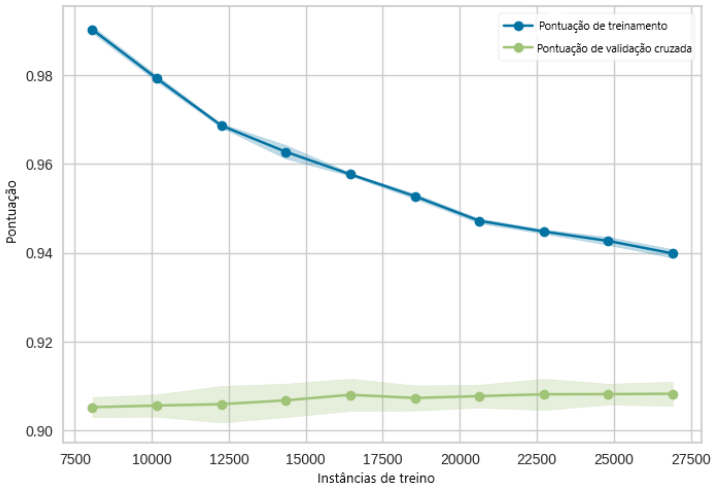


Figura 18 – Curva de aprendizado para o modelo sem seleção de *features*

Fonte: Elaborado pela autora.

(Adaptado para português pela autora)

Para a matriz de confusão o modelo foi ajustado diretamente na base completa, tendo na base de teste cerca de 118 mil observações. A Figura 19 traz os valores para o modelo sem seleção, apontando uma acurácia de 91.11%, precisão de 94.56% e sensibilidade de 94.02%, gerando um $F1$ de 94.29%.

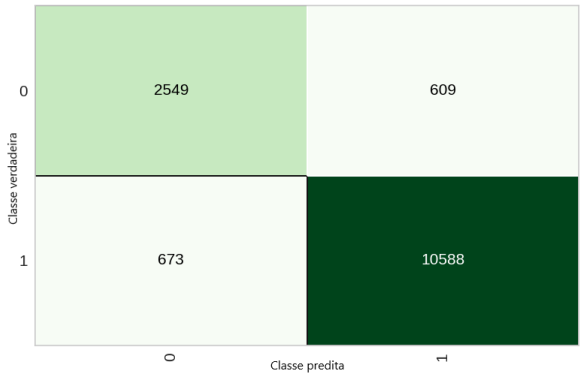


Figura 19 – Matriz de confusão para o modelo sem seleção de *features*

Fonte: Elaborado pela autora.

(Adaptado para português pela autora)

Então, seguindo com o ajuste feito, aplicamos o método SHAP, utilizando a função *TreeExplainer*. Com isso, foi obtida a Figura 20 a seguir, mostrando o impacto das principais variáveis em cada observação. Fica bem evidente que a variável *"last_fico_range_high"* se mostra a mais importante, seus valores altos levam a valores SHAP maiores, ou seja, maior a chance de o empréstimo ser pago. Pesquisando sobre a origem dessa variável essa interpretação faz sentido. O "FICO"é um score geral utilizado para determinar a credibilidade de pessoas, sua pontuação de crédito, muito utilizado por instituições financeira para determinar o quanto de crédito pode ser oferecido e a que taxa de juros, com a pontuação variando entre 300 e 850, indicando que quanto maior esse score, maior a confiabilidade de pagamento do indivíduo.

Em segundo lugar temos a variável *loan_amnt* que se refere ao valor total do empréstimo, impactando inversamente a predição, ou seja, empréstimos maiores possuem um valor SHAP negativo, levando a predição a 0 (Não pago) e valores menores possuem um SHAP positivo tendendo a prever a observação como pago.

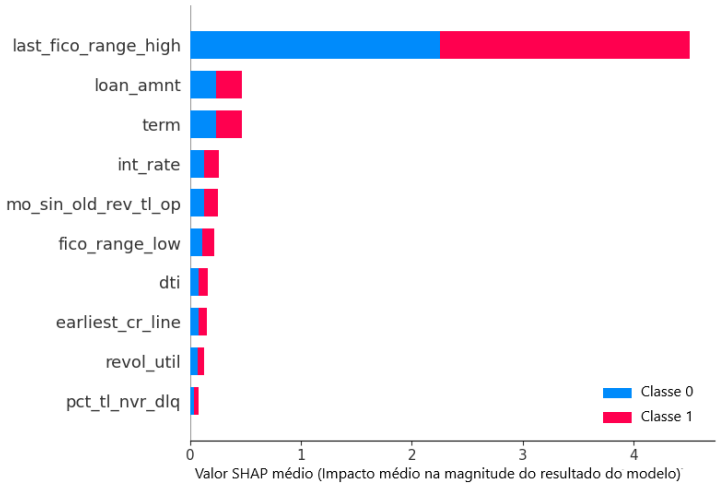


Figura 20 – SHAP Summary Plot para o modelo com seleção de *features* via SHAP

Fonte: Elaborado pela autora.

(Adaptado para português pela autora)

3.2.3 Com seleção de variáveis pelo método Lasso

No ajuste do método LASSO também foi utilizada a função *LassoCV* com 5 *folds* variando os valores de α de 0.01 a 1 em intervalos de 0.01, e aplicado na base de amostra de 70 mil observações por quesitos computacionais. Novamente foi escolhido o parâmetro que trouxe a maior acurácia para a regressão ajustada. A Figura 21 traz em verde a acurácia para cada α selecionado, referente ao eixo y, e em cinza a quantidade de variáveis com coeficientes diferentes de 0, referente ao eixo z.

A partir do gráfico nota-se que o primeiro corte já seleciona apenas cerca de 25 variáveis das 123 disponíveis na amostra. Há uma pequena variação de acurácias no início do gráfico mas logo ele já começa a decair. O α escolhido foi justamente o primeiro, de 0.01.

Dado o parâmetro de regularização como $\alpha = 0.01$ foi aplicado um LASSO na base completa e selecionado um total de 23 variáveis. Assim foram aplicados os modelos escolhidos anteriormente, com um *Cross Validation*, na base de treino com todas as observações e com essas 23 variáveis definidas. A Tabela 9 traz os resultados, comparando os 4 modelos ajustados.

Tabela 9 – Comparação de Modelos com seleção de Variáveis Lasso

Modelo	Acurácia	AUC	Sensibilidade	Precisão	F1
Light Gradient Boosting Machine	91.02%	95.34%	93.98%	94.50%	94.24%
Gradient Boosting Classifier	90.93%	95.24%	93.95%	94.41%	94.18%
Extreme Gradient Boosting	90.89%	95.15%	94.04%	94.29%	94.16%
Linear Discriminant Analysis	90.76%	95.08%	94.06%	94.12%	94.09%

Fonte: Dados da pesquisa.

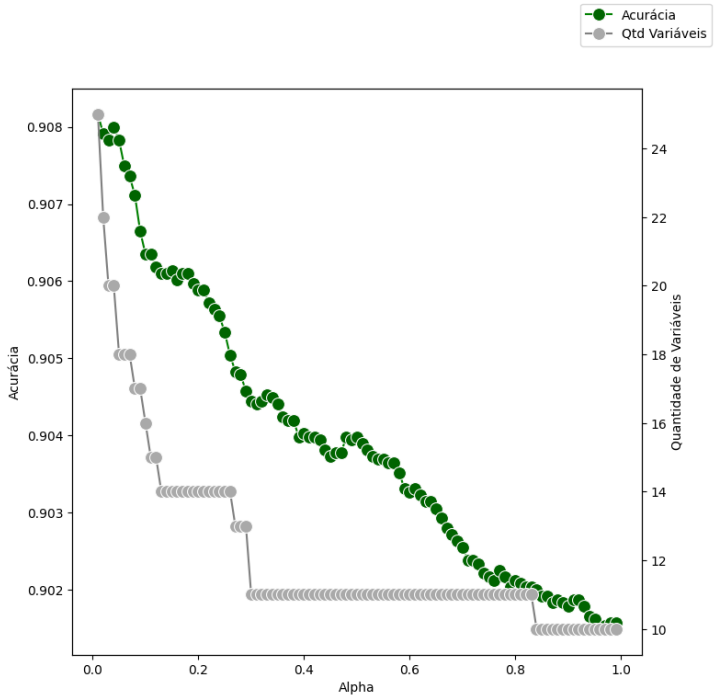


Figura 21 – Escolha do valor de α para o LASSO

Fonte: Elaborado pela autora.

Em seguida gerou-se a curva de aprendizado apresentada na Figura 22, na qual é possível notar o aprendizado de treino cai, como esperado, com o aumento da base e no *Cross-validation* a curva chega a subir, mas com menor intensidade do que no modelo ajustado sem seleção.

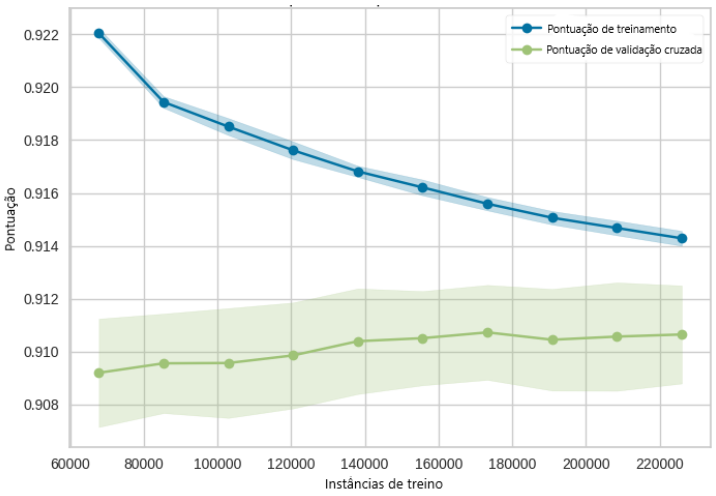


Figura 22 – Curva de aprendizado para o modelo com seleção de *features* pelo LASSO

Fonte: Elaborado pela autora.

(Adaptado para português pela autora)

A matriz de confusão foi obtida com o modelo ajustado a base completa. Apresentada na

Figura 23 ela mostra uma acurácia de 91.04%, precisão de 94.49% e sensibilidade de 94.01%, gerando um F1 de 94.25%. O equilíbrio de acertos e erros, representado pelas cores do gráfico se mantém o mesmo em relação ao modelo ajustado sem seleção.

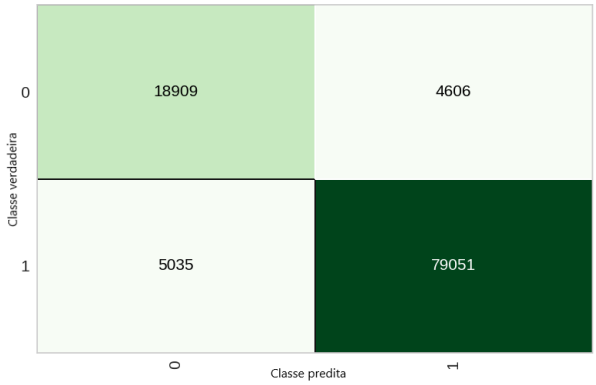


Figura 23 – Matriz de confusão para o modelo com seleção de *features* pelo LASSO

Fonte: Elaborado pela autora.

(Adaptado para português pela autora)

Após o ajuste, o método SHAP traz as importâncias das variáveis pelo *summary plot* na Figura 24. Muito parecido com o modelo anterior a variável *"last_fico_range_high"* continua sendo a mais importante, juntamente com a variável referente ao valor do empréstimo e a variável *term*, que mostra a quantidade de parcelas escolhidas para o pagamente (30 ou 60). Para o resto das variáveis houve algumas alterações na ordem de importância.

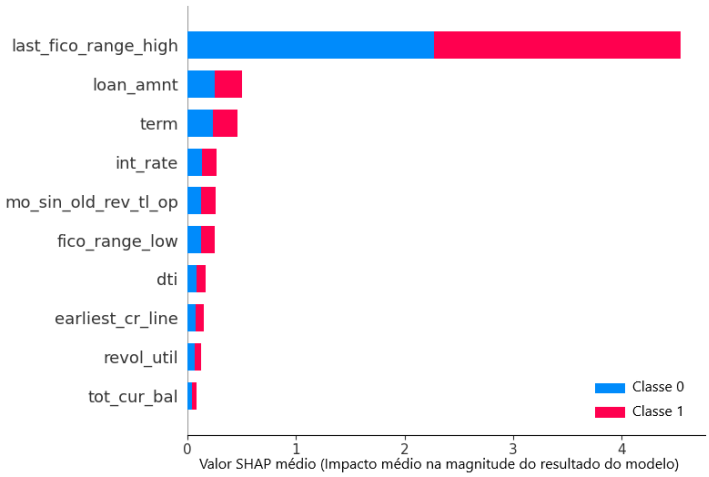


Figura 24 – SHAP Summary Plot para o modelo com seleção de *features* via Lasso

Fonte: Elaborado pela autora.

(Adaptado para português pela autora)

3.2.4 Com seleção de variáveis pelo método SHAP

A escolha do *threshold* de importância mínima para a seleção de variáveis foi feita na base de amostra dado o custo computacional. Ela também foi feita entre os valores de: 0.01, 0.03, 0.05, 0.07, 0.09, 0.1, 0.2. A Figura 25 traz em azul as acurácias encontradas e em vermelho a quantidade de variáveis selecionadas para cada valor.

O primeiro corte em 1% seleciona cerca de 40 variáveis das 123 disponíveis, a partir dele os cortes são mais agressivos. A acurácia começa relativamente baixa mas acaba por ter uma boa variação para alguns valores próximos de 10%.

Nesse gráfico já é possível notar a primeira comparação com os métodos anteriores. Em 2 pontos de seleção do SHAP há uma superioridade na acurácia tanto em relação ao modelo sem seleção quanto ao modelo no qual foi utilizado o método LASSO.

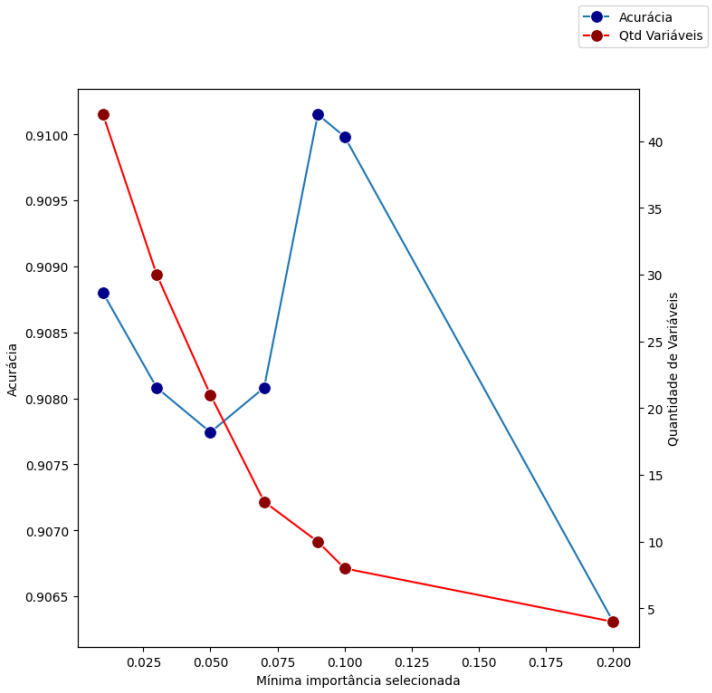


Figura 25 – Escolha do valor da mínima importância de variável para seleção

Fonte: Elaborado pela autora.

O valor escolhido de importância mínima foi de 9%, selecionando assim 10 variáveis a partir da base amostral. Com isso foram aplicados os modelos escolhidos na seção anterior, com um *Cross Validation* e com a base de treino selecionando apenas essas 10 variáveis. Assim foi montada a Tabela 10, definindo o modelo LGBM como o melhor, mas com uma acurácia um pouco acima da anterior.

Com isso foi gerada a curva de aprendizado desse modelo, apresentada na Figura 26, na qual repete-se o comportamento do gráfico para o treino, mas a curva de *Cross-validation*

Tabela 10 – Comparação de modelos com seleção de variáveis pelo SHAP

Modelo	Acurácia	AUC	Sensibilidade	Precisão	F1
Light Gradient Boosting Machine	91.03%	95.36%	94.02%	94.47%	94.25%
Extreme Gradient Boosting	90.93%	95.22%	94.10%	94.28%	94.19%
Gradient Boosting Classifier	90.92%	95.24%	93.96%	94.39%	94.18%
Linear Discriminant Analysis	90.72%	95.06%	94.00%	94.12%	94.06%

Fonte: Dados da pesquisa.

tem um aumento levemente mais significativo se comparado aos dois métodos apresentados anteriormente.

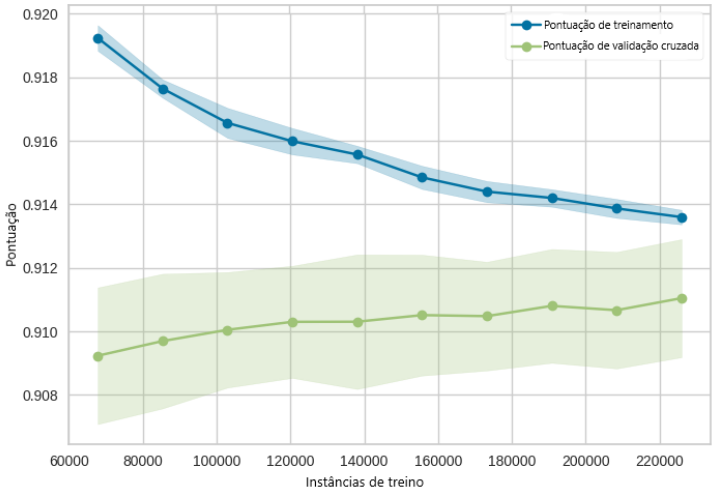


Figura 26 – Curva de aprendizado para o modelo com seleção de *features* pelo SHAP

Fonte: Elaborado pela autora.

(Adaptado para português pela autora)

Também foi obtida a matriz de confusão da base de teste apresentada na Figura 27. Assim a acurácia alcançada foi de 91.04%, precisão de 94.49% e sensibilidade de 94.01%, gerando um F1 de 94.25%. O equilíbrio de acertos e erros, representado pelas cores do gráfico também se manteve o mesmo em relação aos modelos ajustados anteriormente.

Também foi obtida a curva ROC para o modelo aplicando a seleção de variáveis utilizando o SHAP (Figura 28). Na qual é possível notar que até aproximadamente 0,1 de *False Positive Rate* a curva referente a classe de “Totalmente pago"é superior a referente a classe de “Não Pago", e esse comportamento se inverte para valores acima de 0,1.

A Tabela 11 traz uma comparação para os resultados encontrados para os melhores modelos selecionados de cada versão. É possível notar que a acurácia e sensibilidade do modelo com seleção via SHAP é a maior entre que do método Lasso, com uma leve queda na precisão em relação ao modelo com todas as variáveis. E o $F1$, métrica que traz o peso entre a relação de precisão e sensibilidade teve um aumento em relação aos métodos anteriores. Além disso, houve

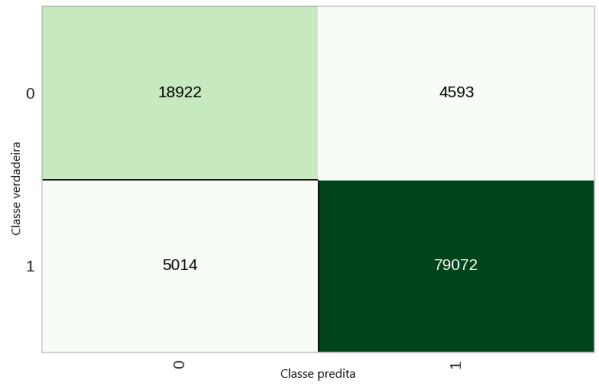


Figura 27 – Matriz de confusão para o modelo com seleção de *features* pelo SHAP

Fonte: Elaborado pela autora.

(Adaptado para português pela autora)

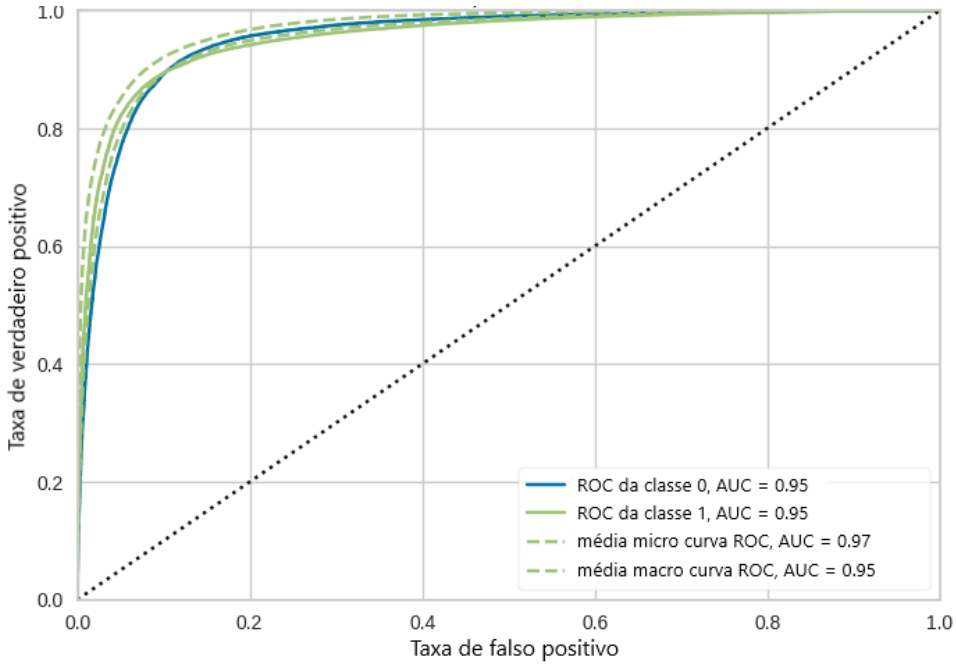


Figura 28 – Curva ROC para modelo com seleção via SHAP

Fonte: Elaborado pela autora.

(Adaptado para português pela autora)

uma redução significativa da quantidade de variáveis, levando a um tempo de execução menor, ou seja, depois de selecionada as variáveis de interesse o tempo de execução do modelo caiu muito no caso apresentado.

Por fim, escolhido o modelo SHAP com seleção como sendo o mais adequado foram aplicados mais gráficos de importância de variáveis do pacote, iniciando novamente pelo *summary plot*. A variável relacionada ao score FICO, que como apresentado anteriormente é um score

Tabela 11 – Resultados dos ajuste dos modelos

Resultados	Sem seleção	Seleção método LASSO	Seleção método SHAP
Quantidade de variáveis	123	23	10
Acurácia	91.11%	91.04%	91.07%
Precisão	94.56%	94.49%	94.51%
Sensibilidade	94.02%	94.01%	94.04%
F1-Score	94.29%	94.25%	94.27%
Escore média geométrica	0.8711	0.8694	0.8698

Fonte: Dados da pesquisa.

geral utilizado para determinar a credibilidade de pessoas, sua pontuação de crédito, continua como sendo a mais importante para o modelo, e o SHAP manteve a *loan_amnt* no segundo lugar. E a terceira mais importante é relativa a quantidade de parcelas do empréstimo.

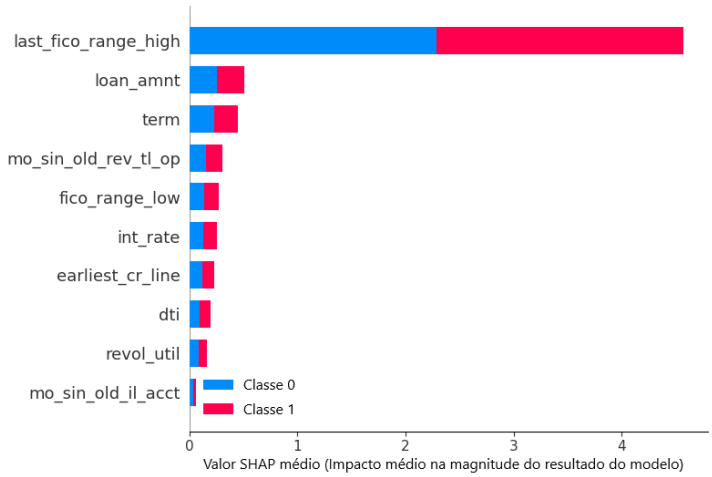


Figura 29 – SHAP Summary Plot para modelo com seleção de *features* via SHAP

Fonte: Elaborado pela autora.

(Adaptado para português pela autora)

A Figura 30 traz o *dependence plot* para a variável *"last_fico_range_high"*, mostrando o efeito que essa variável tem nas previsões feitas pelo modelo. Nesse gráfico é possível observar claramente que valores expressivos tem um grande impacto positivo, ou seja, quanto maior a variável, maior será o valor SHAP e maior será a chance de o pagador quitar a dívida, e vice-versa.

Além disso, foi escolhida como variável de possível relação a *term*, relacionada a quantidade de parcelas do empréstimo. Para quem possui um FICO score abaixo de aproximadamente 600, valores altos de parcelas, ou seja, quem selecionou a quantidade igual a 60 tem menor probabilidade de pagar a dívida em relação a quem escolheu 30 parcelas.

Complementar a isso, a Figura 31 traz o gráfico para a variável *"loan_amnt* mostrando

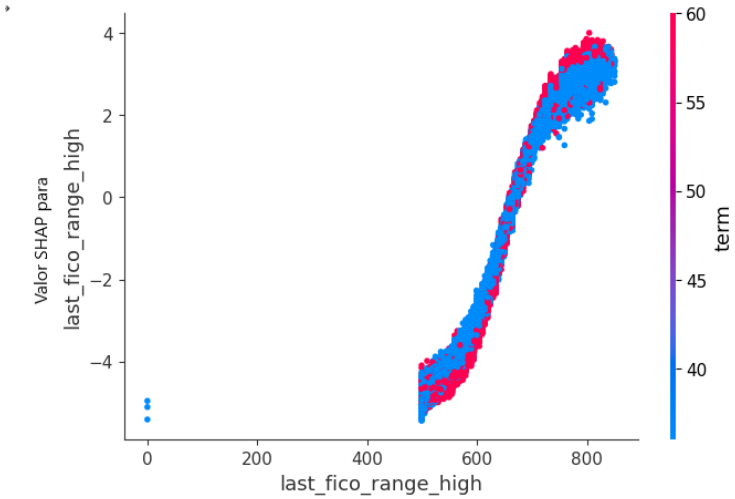


Figura 30 – *Dependence Plot* para a variável *"last_fico_range_high"*

Fonte: Elaborado pela autora.

(Adaptado para português pela autora)

que o valor do empréstimo tem uma influência direta na predição do modelo, quanto maior o valor do empréstimo, o SHAP começa a ter um impacto negativo assim, menor a chance da do empréstimo ser pago. Além disso, mostra melhor ainda a relação com a variável *"last_fico_range_high"*, na qual até valores perto de $\$10000,00$ quanto maior o score, menor a chance de pagamento; e para valores acima de $\$10000,00$ quanto maior o score, maior a chance de pagamento.

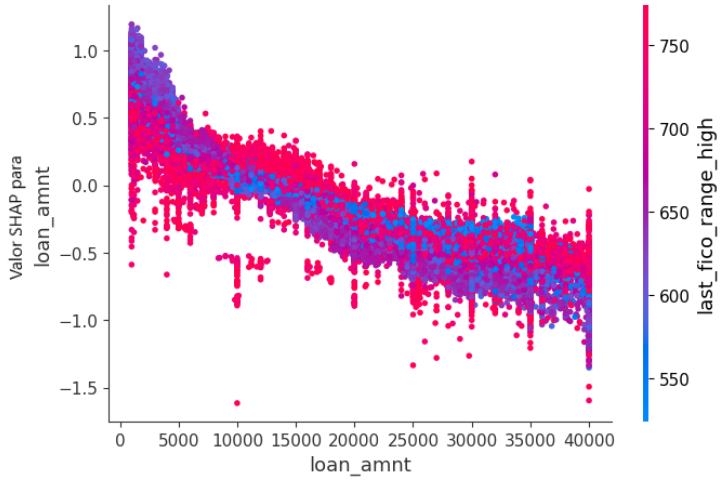


Figura 31 – *Dependence Plot* para a variável *loan_amnt*

Fonte: Elaborado pela autora.

(Adaptado para português pela autora)

A Figura 32 traz um resumo com os principais valores de interação encontrados. Apesar

de não ser visível nenhuma relação muito expressiva, especialmente pelo gráfico ser um resumo de várias variáveis, é possível notar uma leve variação nas duas variáveis relativas a abertura de contas e linhas de crédito, são elas “*earliest_cr_line*"e “*mo_sin_old_rev_tl_op*".

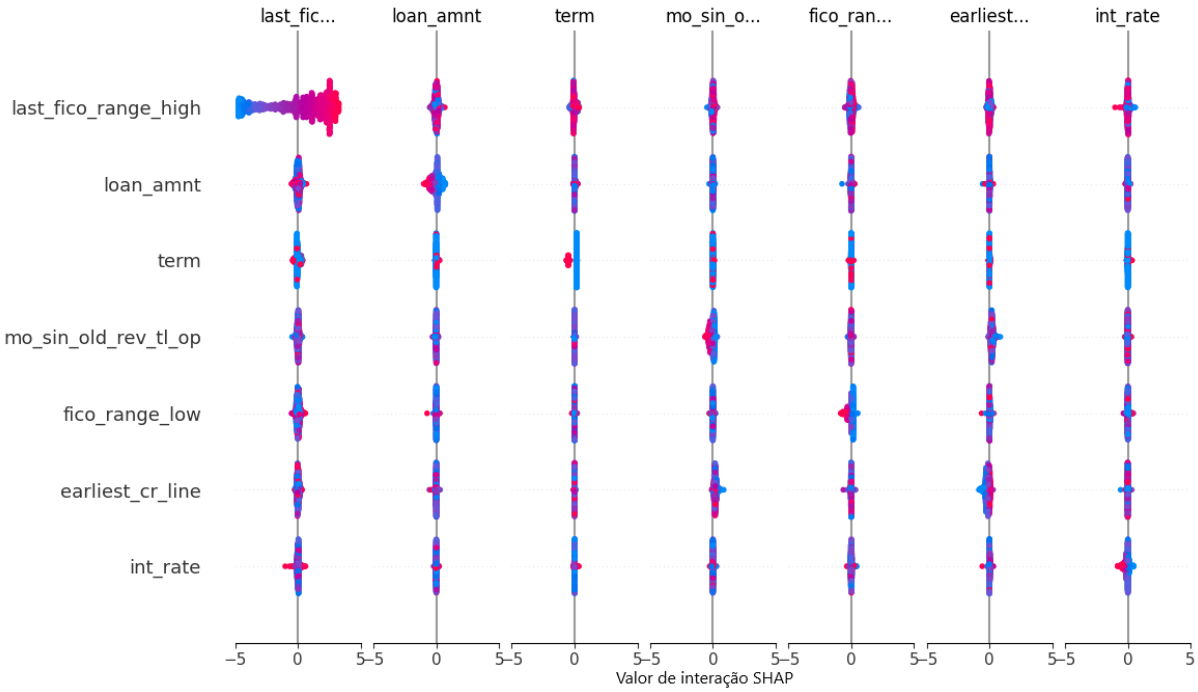


Figura 32 – *SHAP Interation Value* para XGBoost com seleção de *features* via SHAP

Fonte: Elaborado pela autora.

(Adaptado para português pela autora)

Dado os valores encontrados no gráfico da Figura 31, que mostrava uma possível relação entre a variável do score FICO e o valor do empréstimo, foi gerado um *dependence plot* a partir dos valores de interação para essas variáveis, o resultado é mostrado na Figura 33.

A partir dela nota-se que provavelmente deve existir alguma relação entre as duas variáveis no modelo, dado que a relação entre elas não parece ser linear. Os valores altos da variável relativa ao score FICO parecem ter um comportamento de subida conforme os valores de empréstimos vão aumentando, especialmente até os valores próximos a 10.000, enquanto valores menores da variável “last_fico_range_high"tem um comportamento logístico ao aumentarmos o valor do empréstimo.

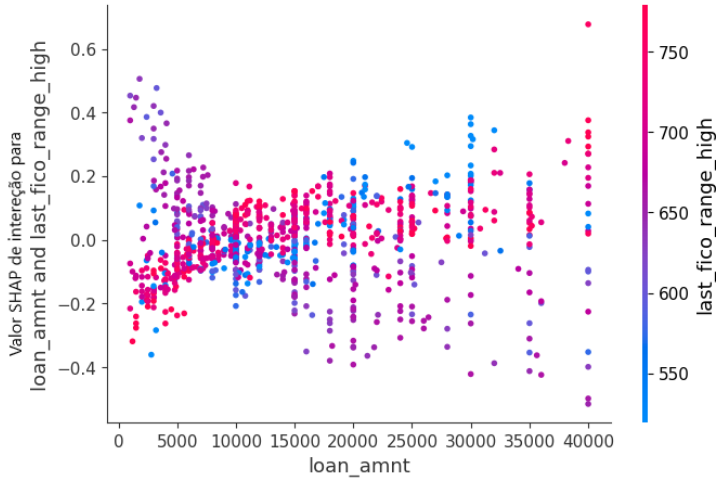


Figura 33 – SHAP Interation Value - *Dependence Plot*

Fonte: Elaborado pela autora.

(Adaptado para português pela autora)

CAPÍTULO

4

ANÁLISE ROBUSTA

Uma análise robusta da variável alvo é uma abordagem crítica na avaliação da estabilidade e confiabilidade de um modelo de classificação. Essa técnica avalia o desempenho do modelo quando a variável alvo possui ruídos, é perturbada ou sujeita a incertezas. Com essa técnica é possível simular cenários do mundo real, onde o alvo pode não ser totalmente preciso devido a vários fatores, por exemplo erros de medição ou ruídos. Neste estudo foi realizada uma análise de robustez no problema de classificação apresentado anteriormente, usando o melhor modelo escolhido (*LGBM*), com as diferentes combinações de covariáveis selecionadas, com foco em perturbar a variável alvo separadamente nos bancos de dados de treino e teste, bem como perturbar os dois bancos conjuntamente. Esse estudo foi feito tanto nos dados simulados quanto para os dados de empréstimos.

Primeiramente foram aplicados diferentes percentuais de perturbações a variável alvo dentro do banco de dados de treinamento e comparadas as performances dos 3 modelos aplicados, com todas as variáveis, com as variáveis escolhidas pelo LASSO e finalmente com as variáveis escolhidas pelo SHAP. O objetivo é avaliar a capacidade dos modelos de generalizar e se adaptar às incertezas durante o processo de treinamento. As perturbações são aplicadas a uma determinada porcentagem dos rótulos-alvo, mantendo o restante inalterado. O modelo é então treinado no conjunto de dados perturbado e seu desempenho é avaliado em relação ao conjunto de teste original.

Nas partes subsequentes da análise, o foco foi ampliado para perturbar a variável de destino no banco de dados de teste e nos bancos de dados de treinamento e teste combinados, utilizando os mesmos percentuais aplicados anteriormente. Ao perturbar os rótulos de teste, é possível investigar o quão bem o modelo mantém sua precisão quando confrontado com valores-alvo incertos ou errôneos durante a fase de avaliação. Além disso, as perturbações combinadas examinam a robustez geral do modelo, levando em consideração os estágios de treinamento e avaliação. Com isso foi possível avaliar como o modelo se degrada em vários cenários de

incerteza de dados e fornecem orientações essenciais para aprimorar a confiabilidade do modelo e os recursos de generalização em aplicativos do mundo real com dados imperfeitos.

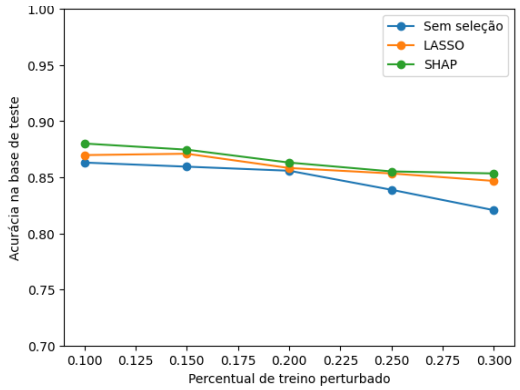
Para essa aplicação foi utilizado diretamente o modelo LGBM com 5 *folds* para todos os testes, não foi possível utilizar a biblioteca do *pycaret* pois são necessárias alterações nas bases de treino e teste, o que não é permitido com o pacote, dado que é necessário fazer a definição das bases que serão utilizadas em todo o processamento uma única vez no início da aplicação.

4.1 Análise em base de dados simulada

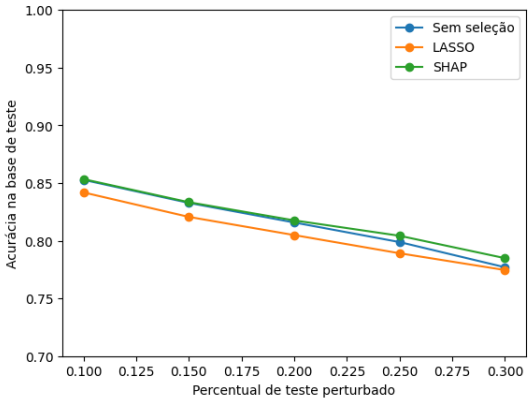
Para aumentar a confiabilidade do estudo foram utilizados 5 percentuais diferentes de perturbação nas diferentes bases, foram utilizados os valores de: 10%, 15%, 20%, 25% e 30%, apenas na variável alvo. Essa pertubação ocorreu da seguinte forma, selecionados os X% da base comentados acima, era gerado um valor de 0 ou 1, aleatoriamente, com 50% de probabilidade como variável alvo para essa linha, adicionando assim um erro aleatório nesse pedaço da base de dados.

A primeira Figura 34 traz uma comparação entre as acurácias para as três diferentes conjuntos de variáveis e para alterações somente em treino, somente em teste e treino e teste conjuntos. A Figura 34a traz os valores da acurácia na base de teste para perturbações na base de treino, nela é possível ver que o ajuste com apenas as variáveis selecionadas pelo SHAP foram superiores em todos os níveis testados, seguido pelo ajuste feito com as variáveis selecionadas pelo algoritmo LASSO.

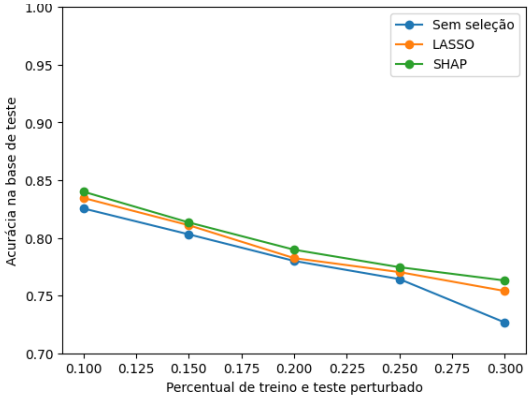
A Figura 34b traz os valores de acurácia para as perturbações feitas apenas na base de teste, continuando com o modelo que contém apenas as variáveis escolhidas pelo SHAP superior a versão que contém todas as variáveis possíveis, e ambos acima do modelo ajustado com as variáveis escolhidas pelo LASSO. Por fim a Figura 34c traz os valores para perturbações feitas tanto em treino quanto em teste, mostrando o SHAP como superior tanto ao LASSO quanto com o modelo sem seleção de variáveis.



(a) Acurácia em teste para perturbações na base de treino



(b) Acurácia em teste para perturbações na base de teste



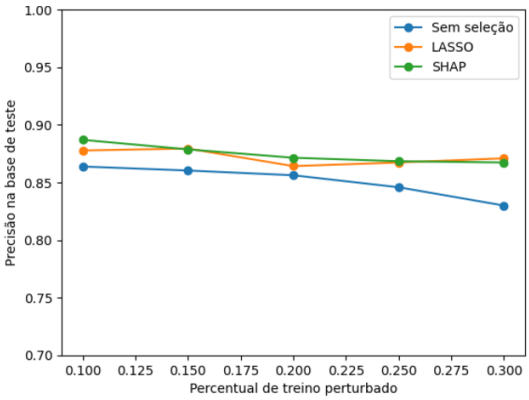
(c) Acurácia em teste para perturbações na bases de treino e teste

Figura 34 – Acurácia na análise de robustez

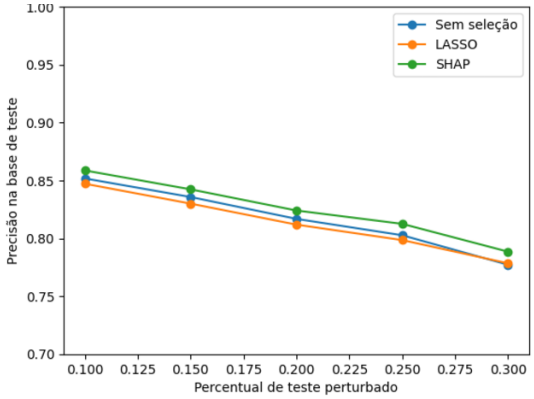
Fonte: Elaborado pela autora.

Complementando os valores de acurácia a Figura 35 traz a precisão encontrada ao aplicar os modelos com perturbações na base de teste. É possível notar que ao perturbar apenas a base de teste, (Figura 35b) as variáveis selecionadas pelo SHAP possuem superioridade para todos os valores, seguido pelo ajuste feito com todas as variáveis.

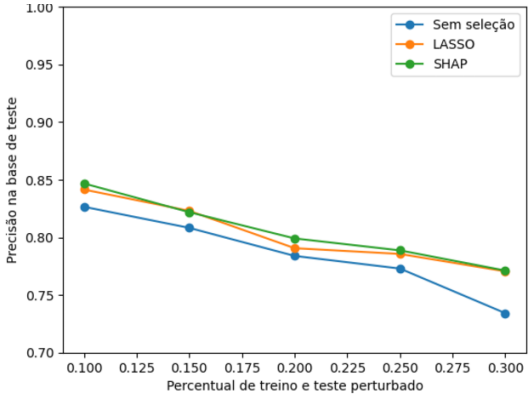
Para o distúrbio feito apenas em treino (Figura 35a) o SHAP está com precisão acima do LASSO em 3 dos 5 pontos, e ambos estão acima do modelo sem seleção de variáveis. Finalmente para perturbações feitas simultaneamente em treino e teste, apresentado na Figura 35c, possui um comportamento muito próximo ao comentado anteriormente, com o SHAP possuindo valores acima tanto do LASSO quanto da versão com todas as variáveis, quase sendo ultrapassado em 2 pontos pelo modelo ajustado com variáveis selecionadas pelo LASSO.



(a) Precisão em teste para perturbações na base de treino



(b) Precisão em teste para perturbações na base de teste



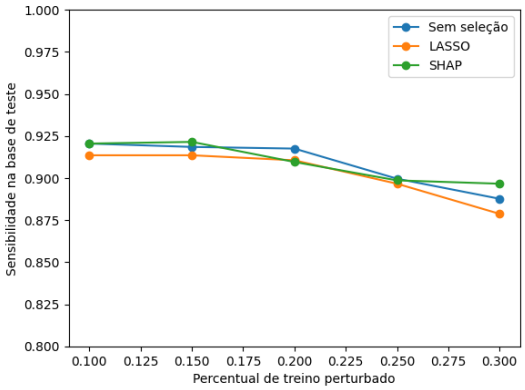
(c) Precisão em teste para perturbações na base de treino e teste

Figura 35 – Precisão na análise de robustez

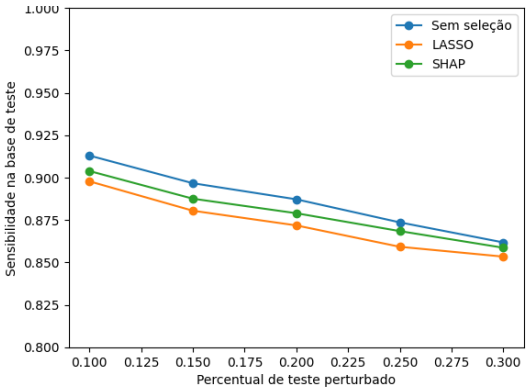
Fonte: Elaborado pela autora.

A Figura 36 apresenta a última medida de classificação apresentada para mostrar os resultados obtidos nessa análise de robustez, a sensibilidade. É possível notar que para os 3 gráficos o ajuste feito com todas as variáveis se mostrou levemente melhor que as outras opções. Especialmente para as perturbações feitas apenas em treino, apresentado na Figura 36a, o modelo com as variáveis selecionadas pelo SHAP se mostram melhores em dois momentos, com 15% e com 30% de perturbação.

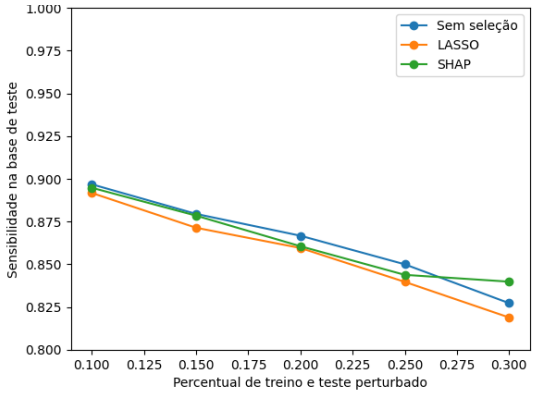
Nas outras aplicações, em teste (Figura 36b) e em teste e treino conjuntamente (Figura 36c) o modelo completo está acima em quase todos os valores, mas o ajuste feito com as variáveis escolhidas pelo SHAP está acima em todos os percentuais perturbados do modelo ajustado com variáveis selecionadas pelo LASSO.



(a) Sensibilidade em teste para perturbações na base de treino



(b) Sensibilidade em teste para perturbações na base de teste



(c) Sensibilidade em teste para perturbações na base de treino e teste

Figura 36 – Sensibilidade na análise de robustez

Fonte: Elaborado pela autora.

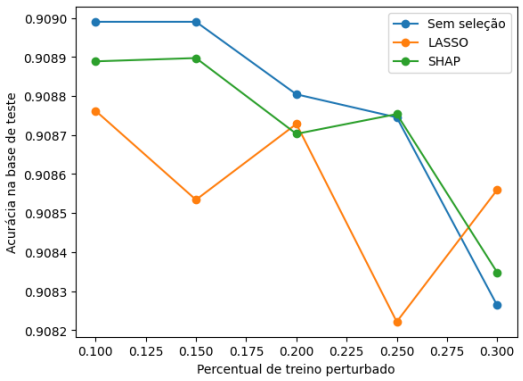
4.2 Análise em base de dados real

Na base de dados real também foram aplicados 5 percentuais de perturbação diferentes para gerar mais evidências para o estudo. Em seguida foram aplicados nos três conjuntos de variáveis diferentes utilizando um *cross validation* com 5 *folds*. É importante salientar que para os gráficos da base real foi escolhido manter os valores de limites do eixo y diferentes entre os gráficos, para uma melhor visualização do comportamento das medidas.

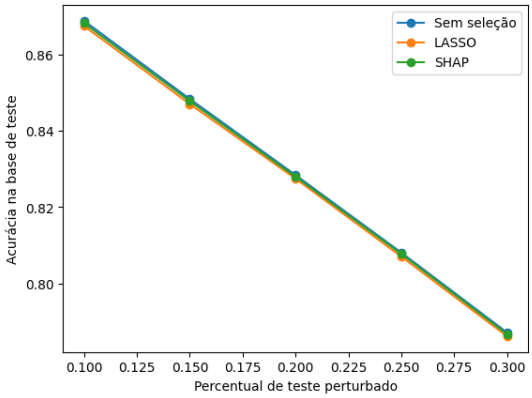
Iniciando com a medida de acurácia, os valores encontrados nos testes são apresentados a Figura 37. Para as perturbações em treino (Figura 37a) o modelo ajustado com todas as variáveis se mostra superior em 4 dos 5 percentuais perturbados seguido pelo modelo que utiliza as variáveis selecionadas pelo SHAP, que alcança o melhor resultado para 30% de perturbações entre as 3 bases comparadas.

Os valores obtidos para as perturbações feitas em teste e simultaneamente em teste e

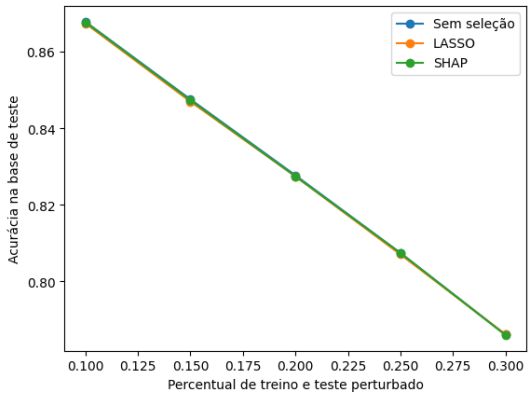
treino foram muito próximos entre as 3 bases de variáveis utilizadas, mostrando basicamente uma relação linear inversamente proporcional. Para as perturbações em teste, apresentadas na Figura 37b é possível notar um leve descolamento pra baixo no ajuste feito com a base de dados selecionada pelo modelo LASSO.



(a) Acurácia em teste para perturbações na base de treino



(b) Acurácia em teste para perturbações na base de teste



(c) Acurácia em teste para perturbações na bases de treino e teste

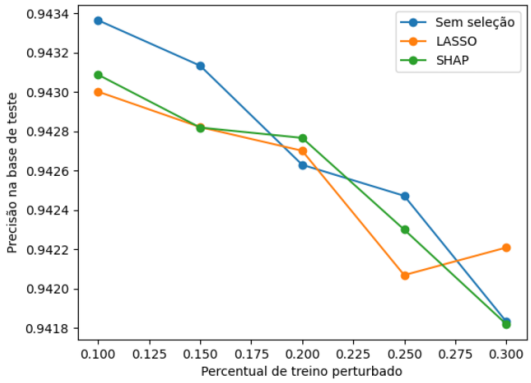
Figura 37 – Precisão na análise de robustez

Fonte: Elaborado pela autora.

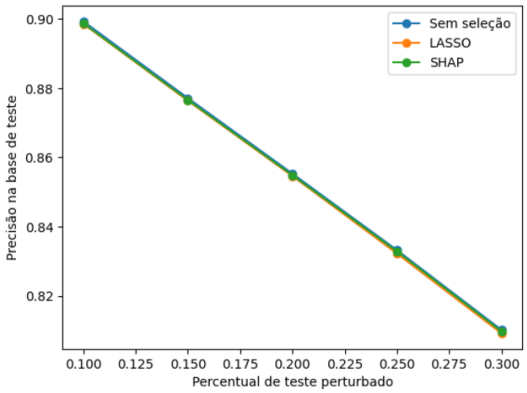
Em seguida é possível acompanhar os valores de precisão na Figura 38, mostrando resultados muito parecidos com os da acurácia. Para as perturbações na base de treino a superioridade fica entre os modelos ajustados com todas as variáveis ou com as variáveis selecionadas pelo SHAP, exceto com 30% da base perturbada, onde o modelo com as variáveis selecionadas pelo LASSO está acima dos outros 2. São mudanças sutis, de quarta casa decimal, mas o SHAP obteve esses resultados utilizando muito menos variáveis.

Para os outros gráficos (Figura 38b e Figura 38c) os valores novamente são muito próximos entre as 3 bases ajustadas. Além disso, também é possível notar o leve descolamento que acontece nas perturbações feita na base de teste, especialmente para valores percentuais mais

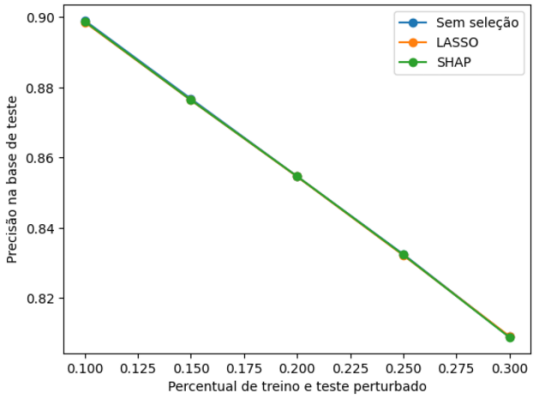
altos.



(a) Precisão em teste para perturbações na base de treino



(b) Precisão de teste para perturbações na base de teste



(c) Precisão em teste para perturbações na base de treino e teste

Figura 38 – Precisão na análise de robustez

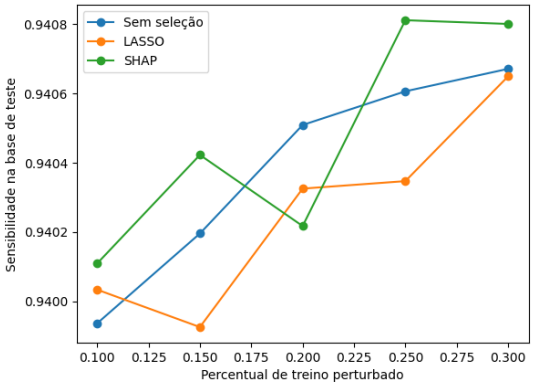
Fonte: Elaborado pela autora.

Os valores de sensibilidade encontrados (Figura 39) apresentaram resultados um pouco diferentes. Em especial os valores para as perturbações na base de treino, mostrados na Figura 39a, apresentaram um crescimento diretamente proporcional ao percentual da base perturbada, mas é importante notar que a diferença está na terceira casa decimal, ou seja, os valores estão muito próximos.

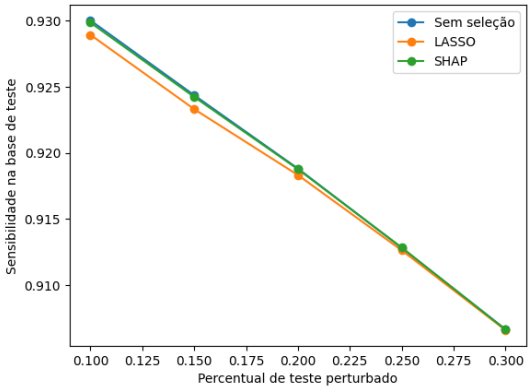
Nas perturbações feitas na base de teste, apresentadas na Figura 39b, os valores de sensibilidade somente com as variáveis selecionadas pelo SHAP estão muito próximos aos valores encontrados com a base completa, apenas o modelo ajustado com as variáveis LASSO se mostrou inferior especialmente nos quatro primeiros percentuais testados.

Por fim, para o caso onde foram provocadas perturbações tanto na base de treino quanto na de teste os valores de sensibilidade se mostraram bem próximos entre as três bases de variáveis, com um leve descolamento para baixo em 3 valores para a base com variáveis selecionadas pelo

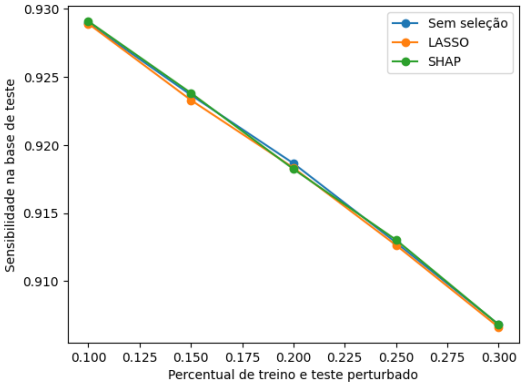
algorimo LASSO.



(a) Sensibilidade em teste para perturbações na base de treino



(b) Sensibilidade em teste para perturbações na base de teste



(c) Sensibilidade em teste para perturbações na base de treino e teste

Figura 39 – Sensibilidade na análise de robustez

Fonte: Elaborado pela autora.

CAPÍTULO

5

DISCUSSÃO

Este trabalhou explorou dois métodos diferentes para seleção de variáveis em aprendizado de máquina: o método SHAP (*SHapley Additive exPlanations*) e o método Lasso (*Least Absolute Shrinkage and Selection Operator*). O objetivo principal foi entender e comparar melhor esses métodos, para esclarecer seus pontos fortes, limitações e conseguir apontar se o SHAP é um método válido para seleção de variáveis. As metodologias de SHAP e Lasso foram esclarecidas, proporcionando uma compreensão mais profunda de seus princípios subjacentes e métodos estatísticos. A abordagem SHAP, baseada na teoria dos jogos cooperativos, demonstrou sua habilidade em determinar a importância de cada variável a partir das previsões do modelo. Enquanto isso, o método Lasso, que usa a regularização para reduzir as dimensões, demonstrou sua eficácia na seleção de variáveis apropriadas, suprimindo os menos informativos.

Uma análise comparativa detalhada foi realizada, considerando várias medidas de validade de classificação, incluindo precisão, sensibilidade, F1 score e acurácia. Essa análise foi realizada em bancos de dados simulados e reais. O teste de banco de dados simulado revelou que o método SHAP teve um desempenho melhor do que o método Lasso como seletor de variáveis, enfatizando sua capacidade de identificar com mais precisão os fatores influenciadores.

Fazendo a transição para um banco de dados do mundo real, o método SHAP mostrou resultados levemente superiores ao método Lasso, embora com valores de acurácia ligeiramente inferiores aos do modelo completo. No entanto, a notável façanha de reduzir um complexo espaço compreendendo 142 variáveis para apenas 10, mantendo a proximidade com a precisão do modelo completo, destacou a importância do SHAP como uma ferramenta de seleção de variáveis.

Além disso, foi realizado um estudo da robustez dos modelos e variáveis selecionados, introduzindo valores de perturbação variando de 10% a 30% nos conjuntos de treinamento, teste e em ambos conjuntamente. Esta análise foi realizada em bancos de dados simulados e reais. Os resultados evidenciaram a resiliência das variáveis escolhidas pelo método SHAP, principalmente

em termos de acurácia. A estabilidade e consistência exibidas por essas variáveis, mesmo sob perturbações, acentuam a confiabilidade do SHAP em cenários do mundo real.

Em conclusão, foi mostrado o papel fundamental da seleção de variáveis no domínio da interpretabilidade do modelo de aprendizado de máquina e melhoria de desempenho. Por meio de uma exploração comparativa, o método SHAP emergiu como um forte concorrente, destacando-se não apenas em ambientes simulados, mas também nas complexidades dos bancos de dados do mundo real. Embora o método Lasso também tenha demonstrado seu mérito, a capacidade do método SHAP de capturar efetivamente a importância da variável, especialmente para relações não lineares, e produzir modelos compactos, porém precisos, é notável.

É essencial reconhecer os caminhos para pesquisas futuras que possam surgir a partir dessa pesquisa. Uma direção promissora reside em melhorar a interpretabilidade dos próprios valores SHAP. Pesquisas futuras poderiam aprofundar o desenvolvimento de novas técnicas de visualização e explicações intuitivas para os valores SHAP, tornando-os ainda mais acessíveis para não especialistas e auxiliando na transparência dos modelos de aprendizado de máquina. Além disso, explorar a adaptação de valores SHAP a diferentes domínios e tipos de dados, como séries temporais ou dados de linguagem natural, tem um potencial imenso. O desenvolvimento de extensões SHAP específicas de domínio poderia abrir caminho para uma adoção mais ampla desta poderosa ferramenta de interpretabilidade.

Outro caminho intrigante para trabalhos futuros envolve a integração das metodologias SHAP e Lasso em estruturas híbridas de seleção de variáveis. A combinação dos pontos fortes de ambos os métodos poderia produzir técnicas de seleção ainda mais robustas e precisas. Investigar a otimização de hiperparâmetros e critérios de seleção dentro dessas abordagens híbridas apresenta um terreno fértil para futuras explorações. Além disso, estender esta pesquisa a dados multimodais, onde diversas fontes de dados são combinadas, poderia abrir novas possibilidades para técnicas de seleção de variáveis que sejam adaptáveis a outras aplicações complexas do mundo real. Em última análise, essas direções futuras prometem avançar na compreensão dos métodos de seleção de variáveis, melhorando a interpretabilidade e o desempenho dos modelos de aprendizado de máquina em uma ampla gama de domínios.

Métodos alternativos de avaliação dos modelos também podem ser pesquisados, para avançar e detalhar mais a comparação entre os concorrentes. Um exemplo seriam o valor preditivo positivo (VPP) que é a probabilidade de uma observação prevista ter como resultado como "Pago"e o empréstimo ter realmente sido pago; e o valor preditivo negativo (VPN), analogamente é a probabilidade da previsão resultar em "Não pago"e o empréstimo não ser realmente pago.

Os resultados aqui obtidos e discutidos, têm boas ramificações e oferecem orientações interessantes para novas pesquisas e aprofundamentos no cenário de seleção de variáveis. À medida que o campo de aprendizado de máquina continua a evoluir, as descobertas aqui apresentadas devem servir como base, facilitando o desenvolvimento de modelos mais interpretáveis,

eficientes e robustos.

REFERÊNCIAS

ALI, M. **PyCaret: An open source, low-code machine learning library in Python**. [S.l.], 2020. PyCaret version 1.0.0. Disponível em: <https://www.pycaret.org>. Citado na página 39.

BISHOP, C. M.; NASRABADI, N. M. **Pattern recognition and machine learning**. [S.l.]: Springer, 2006. v. 4. Citado na página 35.

BREIMAN, L. Random forests. **Machine learning**, Springer, v. 45, p. 5–32, 2001. Citado na página 19.

CHEN, T.; GUESTRIN, C. Xgboost: A scalable tree boosting system. In: **Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining**. [S.l.: s.n.], 2016. p. 785–794. Citado na página 32.

CRAMER, J. S. The origins of logistic regression. Tinbergen institute working paper, 2002. Citado na página 29.

CUNHA, L. W.; RUSSO, C. M. Comparative analysis of variables selection methods in classification problems. **Book of Abstratics**, 2023. Citado na página 42.

DOSHI-VELEZ, F.; KIM, B. Towards a rigorous science of interpretable machine learning. **arXiv preprint arXiv:1702.08608**, 2017. Citado na página 19.

FAN, J.; MA, X.; WU, L.; ZHANG, F.; YU, X.; ZENG, W. Light gradient boosting machine: An efficient soft computing model for estimating daily reference evapotranspiration with local and external meteorological data. **Agricultural water management**, Elsevier, v. 225, p. 105758, 2019. Citado na página 34.

FONTI, V.; BELITSER, E. Feature selection using lasso. **VU Amsterdam research paper in business analytics**, Business Analytics Master Amsterdam, The Netherland, v. 30, p. 1–25, 2017. Citado na página 21.

FORMAN, G. *et al.* An extensive empirical study of feature selection metrics for text classification. **J. Mach. Learn. Res.**, v. 3, n. Mar, p. 1289–1305, 2003. Citado na página 19.

FRIEDMAN, J. H. Greedy function approximation: a gradient boosting machine. **Annals of statistics**, JSTOR, p. 1189–1232, 2001. Citado na página 34.

_____. Stochastic gradient boosting. **Computational statistics & data analysis**, Elsevier, v. 38, n. 4, p. 367–378, 2002. Citado na página 34.

GHORBANI, A.; ZOU, J. Data shapley: Equitable valuation of data for machine learning. In: PMLR. **International Conference on Machine Learning**. [S.l.], 2019. p. 2242–2251. Citado na página 20.

GHOSH, P.; AZAM, S.; JONKMAN, M.; KARIM, A.; SHAMRAT, F. J. M.; IGNATIOUS, E.; SHULTANA, S.; BEERAVOLU, A. R.; BOER, F. D. Efficient prediction of cardiovascular disease using machine learning algorithms with relief and lasso feature selection techniques. **IEEE Access**, IEEE, v. 9, p. 19304–19326, 2021. Citado na página 21.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep learning**. [S.l.]: MIT press, 2016. Citado na página 35.

KAGGLE. **Lending Club 2007-2020Q3**. 2022. <https://www.kaggle.com/datasets/ethon0426/lending-club-20072020q1?resource=download>. Accessed: 2022-06-12. Citado nas páginas 21 e 53.

KE, G.; MENG, Q.; FINLEY, T.; WANG, T.; CHEN, W.; MA, W.; YE, Q.; LIU, T.-Y. Lightgbm: A highly efficient gradient boosting decision tree. **Advances in neural information processing systems**, v. 30, p. 3146–3154, 2017. Citado na página 33.

KWON, Y.; ZOU, J. Weightedshap: analyzing and improving shapley based feature attributions. **arXiv preprint arXiv:2209.13429**, 2022. Citado na página 20.

LEE, C.; LEE, G. G. Information gain and divergence-based feature selection for machine learning-based text categorization. **Information processing & management**, Elsevier, v. 42, n. 1, p. 155–165, 2006. Citado na página 19.

LUNDBERG, S. M.; ERION, G.; CHEN, H.; DEGRAVE, A.; PRUTKIN, J. M.; NAIR, B.; KATZ, R.; HIMMELFARB, J.; BANSAL, N.; LEE, S.-I. From local explanations to global understanding with explainable ai for trees. **Nature Machine Intelligence**, Nature Publishing Group, v. 2, n. 1, p. 2522–5839, 2020. Citado nas páginas 23, 24 e 53.

LUNDBERG, S. M.; LEE, S.-I. A unified approach to interpreting model predictions. In: GUYON, I.; LUXBURG, U. V.; BENGIO, S.; WALLACH, H.; FERGUS, R.; VISHWANATHAN, S.; GARNETT, R. (Ed.). **Advances in Neural Information Processing Systems 30**. Curran Associates, Inc., 2017. p. 4765–4774. Disponível em: <http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf>. Citado nas páginas 20, 25, 26 e 27.

SAEYS, Y.; INZA, I.; LARRANAGA, P. A review of feature selection techniques in bioinformatics. **bioinformatics**, Oxford University Press, v. 23, n. 19, p. 2507–2517, 2007. Citado na página 19.

TAHA, A. A.; MALEBARY, S. J. An intelligent approach to credit card fraud detection using an optimized light gradient boosting machine. **IEEE Access**, IEEE, v. 8, p. 25579–25587, 2020. Citado na página 34.

TIBSHIRANI, R. Regression shrinkage and selection via the lasso. **Journal of the Royal Statistical Society: Series B (Methodological)**, Wiley Online Library, v. 58, n. 1, p. 267–288, 1996. Citado na página 20.

ZHANG, Y.; HAGHANI, A. A gradient boosting method to improve travel time prediction. **Transportation Research Part C: Emerging Technologies**, Elsevier, v. 58, p. 308–324, 2015. Citado na página 34.

ZHENG, H.; ZHANG, Y. Feature selection for high-dimensional data in astronomy. **Advances in Space Research**, Elsevier, v. 41, n. 12, p. 1960–1964, 2008. Citado na página 19.

APÊNDICE

A

DICIONÁRIO VARIÁVEIS

LoanStatNew	Description
acc_now_delinq	The number of accounts on which the borrower is now delinquent.
acc_open_past_24mths	Number of trades opened in past 24 months.
addr_state	The state provided by the borrower in the loan application
all_util	Balance to credit limit on all trades
annual_inc	The self-reported annual income provided by the borrower during registration.
annual_inc_joint	The combined self-reported annual income provided by the co-borrowers during registration
application_type	Indicates whether the loan is an individual application or a joint application with two co-borrowers
avg_cur_bal	Average current balance of all accounts
bc_open_to_buy	Total open to buy on revolving bankcards.
bc_util	Ratio of total current balance to high credit/credit limit for all bankcard accounts.
chargeoff_within_12_mths	Number of charge-offs within 12 months
collection_recovery_fee	post charge off collection fee
collections_12_mths_ex_med	Number of collections in 12 months excluding medical collections
delinq_2yrs	The number of 30+ days past-due incidences of delinquency in the borrower's credit file for the past 2 years
delinq_amnt	The past-due amount owed for the accounts on which the borrower is now delinquent.
desc	Loan description provided by the borrower

dti	A ratio calculated using the borrower's total monthly debt payments on the total debt obligations, excluding mortgage and the requested LC loan, divided by the borrower's self-reported monthly income.
dti_joint	A ratio calculated using the co-borrowers' total monthly payments on the total debt obligations, excluding mortgages and the requested LC loan, divided by the co-borrowers' combined self-reported monthly income
earliest_cr_line	The month the borrower's earliest reported credit line was opened
emp_length	Employment length in years. Possible values are between 0 and 10 where 0 means less than one year and 10 means ten or more years.
emp_title	The job title supplied by the Borrower when applying for the loan.*
fico_range_high	The upper boundary range the borrower's FICO at loan origination belongs to.
fico_range_low	The lower boundary range the borrower's FICO at loan origination belongs to.
funded_amnt	The total amount committed to that loan at that point in time.
funded_amnt_inv	The total amount committed by investors for that loan at that point in time.
grade	LC assigned loan grade
home_ownership	The home ownership status provided by the borrower during registration or obtained from the credit report. Our values are: RENT, OWN, MORTGAGE, OTHER
id	A unique LC assigned ID for the loan listing.
il_util	Ratio of total current balance to high credit/credit limit on all install acct
initial_list_status	The initial listing status of the loan. Possible values are – W, F
inq_fi	Number of personal finance inquiries
inq_last_12m	Number of credit inquiries in past 12 months
inq_last_6mths	The number of inquiries in past 6 months (excluding auto and mortgage inquiries)
installment	The monthly payment owed by the borrower if the loan originates.

int_rate	Interest Rate on the loan
issue_d	The month which the loan was funded
last_credit_pull_d	The most recent month LC pulled credit for this loan
last_fico_range_high	The upper boundary range the borrower's last FICO pulled belongs to.
last_fico_range_low	The lower boundary range the borrower's last FICO pulled belongs to.
last_pymnt_amnt	Last total payment amount received
last_pymnt_d	Last month payment was received
loan_amnt	The listed amount of the loan applied for by the borrower. If at some point in time, the credit department reduces the loan amount, then it will be reflected in this value.
loan_status	Current status of the loan
max_bal_bc	Maximum current balance owed on all revolving accounts
member_id	A unique LC assigned Id for the borrower member.
mo_sin_old_il_acct	Months since oldest bank installment account opened
mo_sin_old_rev_tl_op	Months since oldest revolving account opened
mo_sin_rcnt_rev_tl_op	Months since most recent revolving account opened
mo_sin_rcnt_tl	Months since most recent account opened
mort_acc	Number of mortgage accounts.
mths_since_last_delinq	The number of months since the borrower's last delinquency.
mths_since_last_major_derog	Months since most recent 90-day or worse rating
mths_since_last_record	The number of months since the last public record.
mths_since_rcnt_il	Months since most recent installment accounts opened
mths_since_recent_bc	Months since most recent bankcard account opened.
mths_since_recent_bc_dlq	Months since most recent bankcard delinquency
mths_since_recent_inq	Months since most recent inquiry.
mths_since_recent_revol_delinq	Months since most recent revolving delinquency.
next_pymnt_d	Next scheduled payment date
num_accts_ever_120_pd	Number of accounts ever 120 or more days past due
num_actv_bc_tl	Number of currently active bankcard accounts
num_actv_rev_tl	Number of currently active revolving trades
num_bc_sats	Number of satisfactory bankcard accounts
num_bc_tl	Number of bankcard accounts
num_il_tl	Number of installment accounts
num_op_rev_tl	Number of open revolving accounts

num_rev_accts	Number of revolving accounts
num_rev_tl_bal_gt_0	Number of revolving trades with balance >0
num_sats	Number of satisfactory accounts
num_tl_120dpd_2m	Number of accounts currently 120 days past due (updated in past 2 months)
num_tl_30dpd	Number of accounts currently 30 days past due (updated in past 2 months)
num_tl_90g_dpd_24m	Number of accounts 90 or more days past due in last 24 months
num_tl_op_past_12m	Number of accounts opened in past 12 months
open_acc	The number of open credit lines in the borrower's credit file.
open_acc_6m	Number of open trades in last 6 months
open_il_12m	Number of installment accounts opened in past 12 months
open_il_24m	Number of installment accounts opened in past 24 months
open_act_il	Number of currently active installment trades
open_rv_12m	Number of revolving trades opened in past 12 months
open_rv_24m	Number of revolving trades opened in past 24 months
out_prncp	Remaining outstanding principal for total amount funded
out_prncp_inv	Remaining outstanding principal for portion of total amount funded by investors
pct_tl_nvr_dlq	Percent of trades never delinquent
percent_bc_gt_75	Percentage of all bankcard accounts >75% of limit.
policy_code	publicly available policy_code=1new products not publicly available policy_code=2
pub_rec	Number of derogatory public records
pub_rec_bankruptcies	Number of public record bankruptcies
purpose	A category provided by the borrower for the loan request.
pymnt_plan	Indicates if a payment plan has been put in place for the loan
recoveries	post charge off gross recovery
revol_bal	Total credit revolving balance
revol_util	Revolving line utilization rate, or the amount of credit the borrower is using relative to all available revolving credit.
sub_grade	LC assigned loan subgrade
tax_liens	Number of tax liens
term	The number of payments on the loan. Values are in months and can be either 36 or 60.

title	The loan title provided by the borrower
tot_coll_amt	Total collection amounts ever owed
tot_cur_bal	Total current balance of all accounts
tot_hi_cred_lim	Total high credit/credit limit
total_acc	The total number of credit lines currently in the borrower's credit file
total_bal_ex_mort	Total credit balance excluding mortgage
total_bal_il	Total current balance of all installment accounts
total_bc_limit	Total bankcard high credit/credit limit
total_cu_tl	Number of finance trades
total_il_high_credit_limit	Total installment high credit/credit limit
total_pymnt	Payments received to date for total amount funded
total_pymnt_inv	Payments received to date for portion of total amount funded by investors
total_rec_int	Interest received to date
total_rec_late_fee	Late fees received to date
total_rec_prncp	Principal received to date
total_rev_hi_lim	Total revolving high credit/credit limit
url	URL for the LC page with listing data.
verification_status	Indicates if income was verified by LC, not verified, or if the income source was verified
verified_status_joint	Indicates if the co-borrowers' joint income was verified by LC, not verified, or if the income source was verified
zip_code	The first 3 numbers of the zip code provided by the borrower in the loan application.
revol_bal_joint	Sum of revolving credit balance of the co-borrowers, net of duplicate balances
sec_app_fico_range_low	FICO range (high) for the secondary applicant
sec_app_fico_range_high	FICO range (low) for the secondary applicant
sec_app_earliest_cr_line	Earliest credit line at time of application for the secondary applicant
sec_app_inq_last_6mths	Credit inquiries in the last 6 months at time of application for the secondary applicant
sec_app_mort_acc	Number of mortgage accounts at time of application for the secondary applicant
sec_app_open_acc	Number of open trades at time of application for the secondary applicant

sec_app_revol_util	Ratio of total current balance to high credit/credit limit for all revolving accounts
sec_app_open_act_il	Number of currently active installment trades at time of application for the secondary applicant
sec_app_num_rev_accts	Number of revolving accounts at time of application for the secondary applicant
sec_app_chargeoff_within_12_mths	Number of charge-offs within last 12 months at time of application for the secondary applicant
sec_app_collections_12_mths_ex_-med	Number of collections within last 12 months excluding medical collections at time of application for the secondary applicant
sec_app_mths_since_last_major_derog	Months since most recent 90-day or worse rating at time of application for the secondary applicant
hardship_flag	Flags whether or not the borrower is on a hardship plan
hardship_type	Describes the hardship plan offering
hardship_reason	Describes the reason the hardship plan was offered
hardship_status	Describes if the hardship plan is active, pending, canceled, completed, or broken
deferral_term	Amount of months that the borrower is expected to pay less than the contractual monthly payment amount due to a hardship plan
hardship_amount	The interest payment that the borrower has committed to make each month while they are on a hardship plan
hardship_start_date	The start date of the hardship plan period
hardship_end_date	The end date of the hardship plan period
payment_plan_start_date	The day the first hardship plan payment is due. For example, if a borrower has a hardship plan period of 3 months, the start date is the start of the three-month period in which the borrower is allowed to make interest-only payments.
hardship_length	The number of months the borrower will make smaller payments than normally obligated due to a hardship plan
hardship_dpd	Account days past due as of the hardship plan start date
hardship_loan_status	Loan Status as of the hardship plan start date
orig_projected_additional_accrued_-interest	The original projected additional interest amount that will accrue for the given hardship payment plan as of the Hardship Start Date. This field will be null if the borrower has broken their hardship payment plan.

hardship_payoff_balance_amount	The payoff balance amount as of the hardship plan start date
hardship_last_payment_amount	The last payment amount as of the hardship plan start date
disbursement_method	The method by which the borrower receives their loan. Possible values are: CASH, DIRECT_PAY
debt_settlement_flag	Flags whether or not the borrower, who has charged-off, is working with a debt-settlement company.
debt_settlement_flag_date	The most recent date that the Debt_Settlement_Flag has been set
settlement_status	The status of the borrower's settlement plan. Possible values are: COMPLETE, ACTIVE, BROKEN, CANCELLED, DENIED, DRAFT
settlement_date	The date that the borrower agrees to the settlement plan
settlement_amount	The loan amount that the borrower has agreed to settle for
settlement_percentage	The settlement amount as a percentage of the payoff balance amount on the loan
settlement_term	The number of months that the borrower will be on the settlement plan


ICMC
USP
SÃO CARLOS