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

Estudo comparativo de modelos de Machine Learning na predição de um Behaviour Score

Fernanda Waltrs Freitas

Trabalho de Conclusão de Curso

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

Estudo comparativo de modelos de *Machine Learning* na predição de um Behaviour Score

Fernanda Waltrs Freitas
Orientador: Prof. Dr. Rafael Izbicki

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

São Carlos
Fevereiro de 2025

Federal University of São Carlos
Exact and Technology Sciences Center
Department of Statistics

Comparative Study of Machine Learning Models in Predicting a Behaviour Score

Fernanda Waltrs Freitas
Advisor: Prof. Dr. Rafael Izbicki

Bachelors dissertation submitted to the Department of Statistics, Federal University of São Carlos - DEs-UFSCar, in partial fulfillment of the requirements for the degree of Bachelor in Statistics.

São Carlos
February 2025

Fernanda Waltrs Freitas

Estudo comparativo de modelos de *Machine Learning* na predição de um Behaviour Score

Este exemplar corresponde à redação final do trabalho de conclusão de curso devidamente corrigido e defendido por Fernanda Waltrs Freitas e aprovado pela banca examinadora.

Aprovado em 05 de Fevereiro de 2025.

Banca Examinadora:

- Prof. Dr. Rafael Izbicki
- Prof. Dr. Thiago Rodrigo Ramos
- Prof. Dr. Danilo Lourenço Lopes

Resumo

A modelagem de risco de crédito é uma técnica importante para instituições financeiras na tomada de decisões analíticas inteligentes, como na concessão de créditos, financiamentos ou empréstimos. Dessa forma, essas entidades buscam identificar com maior precisão os indivíduos propensos a cumprir suas obrigações dentro do prazo, por meio da estimação da probabilidade dos clientes serem adimplentes ou inadimplentes. Sendo assim, este trabalho tem como objetivo desenvolver a predição de um *Behaviour Score* utilizado por empresas para atribuir pontuações aos seus clientes com base em seu histórico comportamental de transações usando um banco de dados com mais de 400 mil clientes e mais de 200 covariáveis.

Para isso, analisamos o conjunto de dados e ajustamos quatro modelos de aprendizado de máquina, abrangendo desde o modelo mais tradicional, a Regressão Logística com Penalização (Oliveira *et al.* (2012)) e Floresta Aleatória (Breiman (2001)) a modelos mais recentes como eXtreme Gradient Boosting (XGBoost) (Friedman (2001); Chen e Guestrin (2016)) e Light Gradient Boosting Machine (LightGBM) (Ke *et al.* (2017)).

Como resultado final, ao comparar os modelos na amostra de teste, concluímos que no geral todos discriminam bem as classes da variável resposta. No entanto, o método que obteve os melhores resultados preditivos foi o *XGBoost*, com área sob a curva ROC de $82,03\%$, coeficiente de Gini de $64,06\%$, KS de $47,97\%$, entre outras métricas de desempenho. Por outro lado, o método que apresentou os menores resultados preditivos foi a *Regressão Logística com Penalização*, que obteve área sob a curva ROC de $78,46\%$, coeficiente de Gini de $56,92\%$, KS de $42,73\%$, entre outras medidas de desempenho, além de apresentar o maior tempo de treinamento em comparação aos demais modelos.

Palavras-chave: *Behaviour Score, Credit Scoring, LightGBM, Modelagem, Random Forest, Regressão Logística, Risco de Crédito, XGBoost.*

Abstract

Credit Risk Modeling is an important technique for financial institutions in making intelligent analytical decisions, such as granting credit, financing, or loans. In this way, these entities seek to more accurately identify individuals likely to meet their obligations on time by estimating the probability of customers being compliant or delinquent.

Thus, this study aims to develop the prediction of a Behaviour Score used by companies to assign scores to their customers based on their transactional behavior history, using a database with more than 400.000 customers and over 200 covariates.

To achieve this, we analyzed the dataset and adjusted four machine learning models, ranging from the most traditional model, Penalized Logistic Regression (Oliveira *et al.* (2012)) and Random Forest (Breiman (2001)), to more recent models such as eXtreme Gradient Boosting (XGBoost) (Friedman (2001); Chen e Guestrin (2016)) and Light Gradient Boosting Machine (LightGBM) (Ke *et al.* (2017)).

As a final result, when comparing the models in the test sample, we concluded that, in general, all models discriminate well between the response variable classes. However, the method that achieved the best predictive results was XGBoost, with an area under the ROC curve of 82.03%, a Gini coefficient of 64.06%, KS of 47.97%, among other performance metrics. On the other hand, the method that presented the lowest predictive results was Penalized Logistic Regression, which obtained an area under the ROC curve of 78.46%, a Gini coefficient of 56.92%, KS of 42.73%, among other performance measures, in addition to having the longest training time compared to the other models.

Keywords: Score, Credit Scoring, LightGBM, Modeling, Random Forest, Logistic Regression, Credit Risk, XGBoost.

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

A avaliação da probabilidade dos clientes serem bons pagadores ao longo de qualquer processo financeiro desempenha um papel importante na tomada de decisões por parte de instituições financeiras, estabelecimentos comerciais, empresas, seguradoras, entre outros. Tal importância decorre do risco substancial de perdas que essas instituições podem enfrentar caso os tomadores de crédito não cumpram suas obrigações de pagamento, resultando em inadimplência.

Neste contexto, existem diferentes pontuações para os clientes, sendo as mais conhecidas o *Credit Score* e o *Behaviour Score*. O *Credit Score* é uma pontuação atribuída para um possível cliente, por exemplo, na primeira concessão de um cartão ou na realização da primeira compra, quando não há informações internas sobre o histórico de transações. Neste caso, apenas dados cadastrais e demográficos são utilizados para o modelo. Por outro lado, o *Behaviour Score* é uma pontuação atribuída a pessoas que já são clientes há um tempo, considerando também dados comportamentais desse cliente, como o histórico de compras, transações, pagamentos atrasados ou antecipados, entre outros fatores (Luchetta, 2019). Segundo Ribeiro (2011) uma das vantagens desta última pontuação é seu uso na retenção de clientes, especialmente os melhores. Neste estudo, iremos predizer um *Behaviour Score* utilizando variáveis de transações de um cartão exclusivo de determinada loja, juntamente com variáveis cadastrais e demográficas.

As técnicas de aprendizado de máquina estão sendo amplamente empregadas no setor financeiro, especialmente na classificação de clientes como bons ou maus pagadores. Entre os motivos para seu uso, destaca-se a capacidade de lidar com grandes conjuntos de dados com alta velocidade computacional. Segundo a pesquisa realizada por Jadhav *et al.* (2018), técnicas de *machine learning* são utilizadas para construir modelos de avaliação de crédito,

auxiliando instituições financeiras na tomada de decisões relacionadas à concessão ou aumento de crédito.

Neste trabalho, serão estudados e aplicados modelos de *machine learning* para prever a predição da probabilidade do cliente ser bom pagador utilizando variáveis de relacionamento, ou seja, gerar o *Behaviour Score* nos seguintes métodos: Regressão Logística, Random Forest, XGBoost e LightGBM. Esses modelos serão comparados e analisados, considerando suas vantagens, desvantagens e desempenho preditivo. Este problema é de grande relevância, pois a modelagem de risco de crédito é amplamente utilizada por diversas empresas e a construção de modelos mais assertivos possui uma importância significativa para a minimização de riscos e a maximização de retornos.

Trabalhos relevantes para a área incluem Jadhav *et al.* (2018); Qin *et al.* (2021); Santos (2021); Oliveira *et al.* (2012); Mayank Anand (2022); Ribeiro (2011); Forti (2018) na qual comparam metodologias e classificadores, integrando modelos matemáticos para desenvolver novas abordagens e soluções para o problema em análise.

1.0.1 Objetivos

O objetivo do trabalho é comparar quatro modelos, a Regressão Logística com Penalização, Random Forest, XGBoost e LightGBM na predição da probabilidade do cliente ser bom pagador, ou seja, prever um **Behaviour Score**. A comparação foi feita por meio de medidas de desempenho para problemas de classificação e analisamos suas principais vantagens e desvantagens.

Capítulo 2

Metodologia

2.1 Aprendizado de Máquina

Os algoritmos de aprendizado de máquina aprendem com os dados a criar modelos capazes de fazer previsões ou identificar padrões. As principais divisões desses algoritmos são: aprendizado supervisionado, aprendizado não supervisionado e aprendizado por reforço. Neste trabalho, focaremos no aprendizado supervisionado, que visa gerar predições a partir de uma função de mapeamento baseada em um banco de dados com rótulos observados. Dessa maneira, essa função busca representar com um bom poder preditivo e menor erro, a relação entre um conjunto de variáveis de entrada (X) e uma variável de saída (Y). A variável Y é frequentemente chamada de variável resposta, variável dependente ou rótulo (*label*, em inglês), já as observações contidas em $x = (x_1, ..., x_p)$ são geralmente chamadas de variáveis independentes, variáveis explicativas, características, atributos (*features*, em inglês), covariáveis ou preditores (Hastie *et al.*, 2009).

A relação entre X e Y por meio da função f é representada a seguir:

$$Y_i = f(\mathbf{X}_i),$$

em que y_i é a variável resposta para a i-ésima observação e $\mathbf{X}_i$ é o vetor de covariáveis usadas para prever a i-ésima observação.

O objetivo desse algoritmo é aprender a função de tal modo que ao receber novos dados de entrada (X) seja possível fazer previsões mais precisas e com menor erro de predição possível. Ademais, existem duas divisões desse aprendizado, denominadas Regressão e Classificação, que variam conforme a natureza da variável resposta.

2.2 Regressão

Para os problemas de regressão, Y é quantitativa e a estrutura da função de predição é do tipo $\mathbb{R}^p \to \mathbb{R}$, ou seja, $Y \in \mathbb{R}$ para um vetor $\mathbf{x} = (x_1, x_2, ..., x_p) \in \mathbb{R}^p$. Podemos medir sua eficiência quantificando o desvio entre a predição e o valor observado, por exemplo, usando o erro quadrático médio:

$$R(g) = \mathbb{E}[(Y - g(\mathbf{X}))^2],$$

em que $g(\mathbf{X})$ é a função de predição, $\mathbf{X}$ o vetor de covariáveis e Y é a variável resposta.

2.3 Classificação

Para problemas de classificação, Y é qualitativa e assume valores em um conjunto de classes C (para o exemplo principal deste trabalho, C é o conjunto de {bons pagadores, maus pagadores}). Para medir sua eficiência, podemos nos basear na função de risco dada pela probabilidade de erro de uma nova observação:

$$R(g) = \mathbb{E}[\mathbb{I}(Y \neq g(\mathbf{X}))] = \mathbb{P}(Y \neq g(\mathbf{X})),$$

em que $g(\mathbf{X})$ é a função de predição, $\mathbf{X}$ o vetor de covariáveis e Y é a variável resposta.

2.4 Particionamento dos Dados

Após determinar o tipo de aprendizado adequado para nosso problema, devemos escolher a forma de obter os resultados e os *inputs*. Uma maneira usual e eficiente é utilizar o *Data Splitting*, que tem como uma de suas principais vantagens evitar o superajuste do modelo (conhecido como *overfitting*). Basicamente, o *Data Splitting* divide a base de dados em duas amostras aleatórias: **treinamento** e **teste**. O exemplo a seguir divide aleatoriamente o banco de dados com n observações, em 80% para a amostra de treinamento e 20% para a amostra de teste.

$$\text{Treinamento (80\%): } (\mathbf{X}_1, Y_1), (\mathbf{X}_2, Y_2), \ldots, (\mathbf{X}_s, Y_s)$$
$$\text{Teste (20\%): } (\mathbf{X}_{s+1}, Y_{s+1}), (\mathbf{X}_{s+2}, Y_{s+2}), \ldots, (\mathbf{X}_n, Y_n)$$

O conjunto de treinamento é usado exclusivamente para estimar a função de predição,

por exemplo, estimar os coeficientes da Regressão Logística. Já o conjunto de teste é usado para verificar o desempenho da função estimada no treinamento usando métricas de avaliação como Acurária, Gini, Curva ROC, Valor Preditivo Positivo, Sensibilidade, Especificidade, entre outras.

Outra maneira de encontrarmos os *inputs* é através da validação cruzada que utiliza toda a amostra. Primeiramente, os dados são separados aleatoriamente em k-folds de tamanhos próximos. Um desses lotes é usado para teste e os demais serão usados para o treinamento. Esse procedimento é repetido de modo que todos os k-folds sejam usados como amostra de teste e no final podemos calcular uma medida da função de erros desses lotes de teste (Izbicki e dos Santos, 2020).

Após entender qual tipo de problema temos e como particionaremos os dados, chegamos à etapa da modelagem. Neste trabalho, aplicaremos quatro modelos de classificação: Regressão Logística, Random Forest, XGBoost e LightGBM.

Capítulo 3

Modelos

Primeiramente, definiremos a variável resposta utilizada neste trabalho:

$$Y = \begin{cases} 1, & \text{se o cliente é classificado como mau pagador} \\ 0, & \text{se o cliente é classificado como bom pagador} \end{cases}$$

Neste trabalho, aplicaremos técnicas de classificação e na aplicação nos aprofundaremos mais sobre o banco de dados.

3.1 Regressão Logística

Neste contexto, é relevante notar que visamos estimar um score, uma pontuação que varia de 0 a 1000, onde valores mais altos indicam um cliente com melhor perfil. Para isso, estimamos a probabilidade de um cliente ser bom pagador e multiplicaremos por 1000 para obter o score. Seguiremos a explicação da regressão logística apresentada na literatura, utilizando a estimação da probabilidade de classificar $Y = 1$ (mau pagador). De maneira análoga, o processo é aplicado à estimação da probabilidade de pertencimento à classe 0 e que será utilizada na aplicação.

A regressão logística é um método de classificação cujo intuito é modelar as probabilidades das classes de saída dada uma função linear no espaço de covariáveis, e o resultado permanece entre valores de 0 e 1. Além disso, ela é um método paramétrico, ou seja, a estimativa da função de regressão pertence necessariamente a um espaço de funções que pode ser parametrizado por um número finito de parâmetros (Diniz e Louzada, 2013).

No modelo de regressão logística binária, Y possui 2 classes, nesse caso $Y \in \{0, 1\}$,

e sua média é função de uma ou mais covariáveis, denotadas por $\mathbf{x} = (x_1, ..., x_p)$. Dessa maneira, o intuito é estimar a probabilidade de $Y = 1$ segundo informações de um conjunto de *features* que podem ser numéricas ou categóricas. Podemos representar essa definição pela expressão:

$$\mathbb{E}[Y|\mathbf{x}] = \mathbb{P}(Y = 1|\mathbf{x}).$$

Para modelar $\mathbb{P}(Y = 1|\mathbf{x})$, assumimos que $Y|\mathbf{x} \sim$ Bernoulli$(\pi(\mathbf{x}))$, em que $\pi(\mathbf{x}) = \mathbb{P}(Y = 1|\mathbf{x})$ representa a probabilidade de sucesso. Para modelar essa probabilidade, utiliza-se a função de ligação logito, dada por (Hastie *et al.*, 2009) :

$$logito(\pi(\mathbf{x})) = \log\left(\frac{\pi(\mathbf{x})}{1-\pi(\mathbf{x})}\right) = \boldsymbol{\beta}^T\mathbf{x} = \beta_0 + \beta_1 x_1 + ... + \beta_p x_p, \tag{3.1}$$

considerando $\boldsymbol{\beta}$ o vetor de parâmetros desconhecidos do modelo estimado referente ao vetor de covariáveis $\mathbf{x} = (x_1, x_2, ..., x_p)$. O uso da função logito garante que $\pi(\mathbf{x}) \in [0, 1]$ para quaisquer valores de $x_1, ..., x_p$. De 3.1 a probabilidade de sucesso $\pi(\mathbf{x})$, é dada por:

$$\pi(\mathbf{x}) = \frac{e^{\beta_0+\boldsymbol{\beta}^T\mathbf{x}}}{1+e^{\beta_0+\boldsymbol{\beta}^T\mathbf{x}}}.$$

Para estimar os coeficientes $\boldsymbol{\beta}$, podemos usar o método de máxima verossimilhança, necessitando de técnicas numéricas para maximizar a verossimilhança induzida pela regressão logística (Izbicki e dos Santos, 2020). Portanto, dado uma amostra independente $(X_1, Y_1), (X_2, Y_2), ..., (X_n, Y_n)$ a função verossimilhança pode ser escrita como:

$$L(y; (\mathbf{x}, \boldsymbol{\beta})) = \prod_{i=1}^{n} \pi(\mathbf{x})^{y_i}(1-\pi(\mathbf{x}))^{(1-y_i)}.$$

Aplicando o logaritmo da função de verossimilhança tem-se a equação da log-verossimilhança:

$$\begin{aligned} l(y; (\mathbf{x}, \boldsymbol{\beta})) &= \sum_{i=1}^{n}[y_i \log(\pi(\mathbf{x})) + (1-y_i)\log(1-\pi(\mathbf{x}))] \\ &= \sum_{i=1}^{n}\left[y_i \log\left(\frac{e^{\beta_0+\boldsymbol{\beta}^T\mathbf{x}}}{1+e^{\beta_0+\boldsymbol{\beta}^T\mathbf{x}}}\right) + (1-y_i)\log\left(1-\frac{e^{\beta_0+\boldsymbol{\beta}^T\mathbf{x}}}{1+e^{\beta_0+\boldsymbol{\beta}^T\mathbf{x}}}\right)\right], \end{aligned} \tag{3.2}$$

em que a estimativa do vetor $\boldsymbol{\beta}$ é aquela que maximiza $l(y; (\mathbf{x}, \boldsymbol{\beta}))$.

Com objetivo de maximizar a log-verossimilhança devemos igualar a sua derivada em relação a cada coeficiente de $\boldsymbol{\beta}$ a zero, ou seja, $l'(y; (\mathbf{x}, \boldsymbol{\beta})) = 0$ e em seguida para confirmar que os pontos encontrados são de máximo devemos verificar se a segunda derivada

em relação a cada coeficiente de $\boldsymbol{\beta}$ é negativa. Para resolver as equações referente a maximização podemos usar métodos numéricos como o algoritmo de *Newton–Raphson* (Casella e Berger, 2024). Neste estudo, utilizaremos a biblioteca `glmnet` do R para realizar essas estimações.

Uma das principais vantagens deste método é o seu poder interpretativo de variação na log-razão de chances utilizando os coeficientes da regressão logística, assim um valor positivo no modelo se refere a uma maior chance de ocorrência do sucesso da variável resposta quando se aumenta a respectiva variável explicativa, de forma análoga, os valores de parâmetros negativos representam uma probabilidade menor de ocorrência de sucesso da variável resposta quando se aumenta a respectiva variável explicativa. Além de ser um método computacionalmente eficiente.

3.1.1 Seleção de variáveis: Lasso

No modelo de Regressão Logística, o ajuste pode ter muitos parâmetros estimados, e nem todos são relevantes para o estudo. Um modelo com muitos parâmetros tende a apresentar um erro de predição mais baixo nos dados de treinamento, mas pode se tornar excessivamente complexo e ter dificuldade para generalizar para novas observações. Por outro lado, um modelo com poucos parâmetros pode ter um erro de predição mais alto, pois não consegue capturar padrões importantes, resultando em previsões imprecisas. Portanto, é importante encontrar um equilíbrio, evitando tanto um número excessivo quanto um número insuficiente de parâmetros.

A mesma conclusão pode ser obtida ao considerar o balanço entre viés e variância do estimador de predição $\hat{g}$, já que ambos podem variar conforme a sua escolha. De modo geral, um modelo com muitos parâmetros tende a apresentar um viés baixo e uma variância alta, uma vez que é necessário estimar todos os parâmetros. Em contraste, modelos com poucos parâmetros geralmente possuem uma variância baixa e um viés alto, pois são mais simples para descrever os dados. Portanto, também é importante encontrar um equilíbrio entre viés e variância.

Dessa forma, um método útil e eficiente para minimizar o risco e equilibrar o viés e a variância é a seleção de variáveis. Neste trabalho, utilizamos o método Lasso, que apresenta vantagens práticas e teóricas em comparação com outros métodos de seleção, como o Stepwise e o AIC. Uma das vantagens práticas do Lasso é a rapidez na execução do algoritmo e teoricamente, ele tem a vantagem de proporcionar a convergência do estimador

mesmo quando a função não é linear, conforme explorado em Izbicki e dos Santos (2020).

De forma prática, o método lasso quando aplicado a problemas de classificação onde a estimação dos parâmetros é feita via máxima verossimilhança adiciona uma penalização ao oposto do logaritmo da função de verossimilhança. Para a regressão logística, a log-verossimilhança é dada em 3.2. A equação 3.3 abaixo transcreve em termos matemáticos a função objetivo desse método.

$$-l(y;(\mathbf{x},\boldsymbol{\beta})) + \lambda \sum_{i=1}^{p} |\beta_i|, \tag{3.3}$$

sujeito a $\sum_{i=1}^{p} |\beta_i| \leq B$ e λ é um hiperparâmetro.

A medida de complexidade $\sum_{i=1}^{p} |\beta_i|$ é conhecida como Norma L1 e utilizada no Lasso. Por outro lado, também existe a Norma L2, representada por $\sum_{i=1}^{p} \beta_i^2$ e empregada no método Ridge.

Dessa maneira, desejamos minimizar a expressão 3.3 com objetivo de encontrar uma solução em que os coeficientes β se aproximem de zero e sendo possível realizar a seleção de variáveis para cada λ. A Figura 3.1 retirada do livro Hastie *et al.* (2009) mostra como a norma L1 se compara com a L2 graficamente; os gráficos representam um conjunto de soluções para uma regressão com duas covariáveis, sendo o primeiro gráfico em relação ao funcionamento do Lasso e o segundo do Ridge. As elipses nas figuras são os contornos que se referem ao mesmo valor da soma de quadrados dos resíduos, observe que a região da norma L1 é em forma de losango e a L2 de círculo. A solução dos métodos é quando as elipses atingem o primeiro ponto no conjunto de soluções. Observe que pela norma L1 ser losango e possuir vértices em sua solução temos que a elipse tangencia o losango atingindo a estimação e um dos parâmetros é zero. Isso não ocorre geralmente com o Ridge, pois não há vértices em sua solução, sendo difícil sua solução atingir zero em um dos parâmetros (Hastie *et al.*, 2009). Há uma outra variação de penalização denominada Elastic Net que é uma junção do Lasso e Ridge, controlando tanto seleção de modelos como o nível de importância das covaríaveis, mas não será nosso foco no momento.

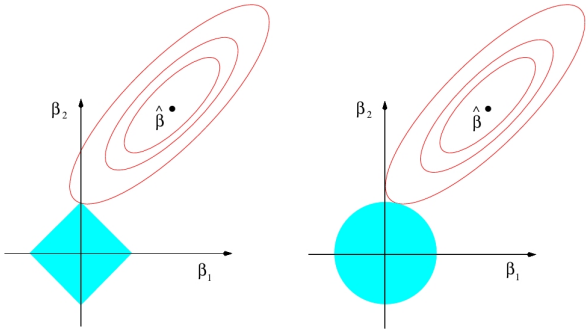


Figura 3.1: Representação da estimação de β usando Lasso e Ridge respectivamente, com os contornos e conjunto solução para duas variáveis. Fonte: Hastie *et al.* (2009)

No Lasso, para realizar a seleção de variáveis com a estimação dos parâmetros, devemos padronizar as covariáveis contínuas de maneira que elas sejam centradas em zero e possuam variância unitária, para que o resultado da estimação não tenha influência da ordem de grandeza das variáveis.

Por fim, a escolha do parâmetro de regularização λ é importante, pois se λ for suficientemente grande, muitos coeficientes se tornam zero melhorando a capacidade preditiva do modelo ao reduzir a variância dos estimadores, no entanto, um valor muito alto pode excluir variáveis importantes. É comum a escolha de λ ser feita por validação cruzada ou simulação e na prática encontramos dois valores importantes: o lambda.min, que corresponde ao menor erro durante a validação cruzada, e o lambda.1se, que é o maior valor de λ para o qual o erro está dentro de uma unidade de desvio padrão do mínimo.

Neste estudo, utilizamos a penalização ao estimar os coeficientes da Regressão Logística, visando obter um melhor poder preditivo ao reduzir a variância do estimador (Izbicki e dos Santos, 2020; James *et al.*, 2013).

3.2 Random Forests

As árvores de decisão são consideradas métodos não paramétricos, pois não exigem que a distribuição da população seja definida por parâmetros específicos. Esse método consiste em particionar recursivamente o espaço das covariáveis independentes em subdivisões $R_1, R_2, ...R_p$ para retornar a variável resposta. Em uma árvore, cada particionamento é denominado nó e o resultado é chamado de folha ou nó terminal. Observe a Figura 3.2 que se refere a uma árvore genérica.

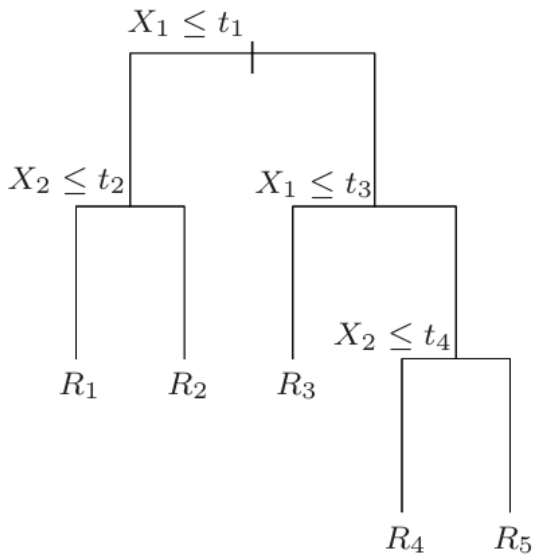


Figura 3.2: Exemplo de árvore genérica. Fonte: James *et al.* (2013).

Fazemos previsões usando árvores observando as condições descritas no topo e seguindo essas condições até encontrar uma folha; se a condição for verdadeira seguimos para a esquerda, caso contrário, seguimos para a direita. Por exemplo, na árvore da Figura 3.2, se a condição $X_1 \leq t_1$ e $X_2 \leq t_2$ for satisfeita, a variável resposta pertence ao grupo R_1. Quando a variável resposta é qualitativa, temos uma árvore de classificação e caso contrário, uma árvore de regressão.

Definiremos formalmente uma árvore de classificação:

Definição 3.4 (Árvore de Classificação)

Uma árvore é um método que gera partições distintas e disjuntas em um espaço de covariáveis: $R_1, R_2, ..., R_j$. *Quando* Y *é dicotômica, denominamos árvore de classificação e a predição para a resposta* Y *usando as covariáveis* $\boldsymbol{x}$ *e que pertencem à região* R_j *é dada pela moda das observações do conjunto de treinamento que pertencem a essa região, ou seja:*

$$h(\boldsymbol{x}) = moda\{y_i : x_i \in R_j\},$$

para i = 1,...,n.

De maneira sucinta, a construção de uma árvore pode ser resumida em duas etapas:

1. Construção de uma árvore complexa,
2. Poda da árvore.

Na 1ª etapa gera-se uma árvore complexa e para encontrar as melhores partições utilizaremos o índice de Gini, dado por:

$$G = \sum_{R} \sum_{j \in C} p_{\hat{R},j}(1 - p_{\hat{R},j}),$$

em que $p_{\hat{R},j}$ corresponde a proporção de observações classificadas como j entre as que estão na região R da árvore (Hastie *et al.*, 2009).

Na prática, queremos minimizar o índice de Gini, pois quando cada folha contém apenas observações de uma única classe encontramos todas as proporções $p_{\hat{R},j}$ próximas de 0 ou 1, resultando em um valor pequeno para G. Nessa situação, dizemos que as árvores estão mais próximas de serem "puras".

Portanto, construímos uma árvore "grande" com muitas divisões e selecionando as variáveis que melhor separam os dados em relação às classes, utilizando para isso as divisões que possuem o menor índice de Gini. Já na 2ª etapa, removemos nós um por vez e analisamos como varia o erro estimado no conjunto de validação. Essa etapa tem o objetivo de evitar o *overfitting* (super ajuste) e melhorar o desempenho preditivo do modelo.

Uma das principais vantagens das árvores de decisão é sua interpretação, pois as predições podem ser explicadas pelos caminhos que levam da raiz às folhas. No entanto, elas costumam apresentar um baixo poder preditivo. Para superar essa limitação, pode-se utilizar *Random Forest*, que é uma extensão do *Bagging* dentro da classe dos métodos *Ensembles*. Esse método consiste em gerar B árvores distintas e combinar seus resultados para melhorar o desempenho preditivo (Izbicki e dos Santos, 2020).

Dessa maneira, o *Bagging* gera B árvores distintas utilizando uma amostra bootstrap da amostra original e que não serão podadas, visando gerar árvores próximas de serem não-viesadas. Contudo, as funções de predição tendem a ter alta correlação, mesmo utilizando amostras bootstrap e assim resultar em predições parecidas.

Logo, *Random Forests* tentam diminuir essa correlação (Breiman, 2001). Além de usar amostragem bootstrap como no *Bagging*, elas limitam a escolha das variáveis em cada nó, permitindo que seja escolhida uma covariável dentre as $m < d$ disponíveis, em vez de escolher entre as d covariáveis originais. O valor m é escolhido aleatoriamente e geralmente usamos validação cruzada ou o valor padrão de muitos softwares para problemas de classificação é a raiz quadrada do número de covariáveis.

Portanto, considere $h_b(\mathbf{x})$ a função de predição obtida pela *b-ésima* árvore de classificação, logo a agregação das diferentes árvores de classificação é feita através da função:

$$h(\mathbf{x}) = moda\{h_b(\mathbf{x}), b = 1, 2, ..., B\}.$$

Dessa maneira, uma observação com covariáveis x é classificada por cada árvore construída e, posteriormente, a predição é dada pela categoria predita com maior frequência (Izbicki e dos Santos (2020)). Portanto, teoricamente a classe de estimadores *Bagging* e *Random Forest* combinados são baseados em gerar funções de predições que não são correlacionadas, não viesadas e que têm a mesma variância.

A seguir, apresentamos alguns hiperparâmetros mais estudados na *Random Forests*:

- **Número de árvores:** Segundo Izbicki e dos Santos (2020) escolher o número de árvores usado em Random Forests não traz grandes ganhos em comparação com outros métodos, pois o risco estimado se estabiliza para um valor consideravelmente alto, como 500 árvores;

- **O valor m de covariáveis aleatórias a serem consideradas em cada divisão:** a quantidade m de covariáveis que serão analisadas em cada divisão quando construímos uma árvore, em problemas de classificação o padrão é considerar a raiz quadrada do número de *features* mas podemos encontrar esse valor por validação cruzada ou simulação;

- **Profundidade máxima:** o caminho máximo entre a raiz da árvore e alguma folha;

- **Tamanho mínimo de um nó:** o número mínimo de obervações a serem incluídas em um nó, geralmente quando esse valor é alto a árvore tende a ser menor, pois reduzimos o número de divisões que acontecem.

Por fim, é possível destacar algumas vantagens desse método, como a seleção automática de covariáveis, uma vez que as árvores realizam a seleção das variáveis mais relevantes e descartam aquelas irrelevantes para a predição, além da facilidade na alocação de variáveis categóricas no modelo.

Já uma das desvantagens é a dificuldade de interpretar a influência de cada covariável sobre a variável resposta. Por outro lado, existe uma medida de importância para cada covariável, que auxilia na identificação das variáveis mais relevantes na predição de Y.

Essa medida é baseada na contribuição de cada variável na predição de Y, podendo ser baseada no índice de Gini.

A *Random Forest* geralmente não requer padronização dos dados, pois é baseada em árvores de decisão, que não são afetadas pela escala dos recursos.

3.3 Otimização Hiperparamétrica

Os hiperparâmetros são valores externos definidos antes do treinamento do modelo e geralmente possuem um papel importante na otimização do seu desempenho. Portanto, a otimização de hiperparâmetros, ou *tuning parameters*, é uma abordagem que testa diferentes valores dos hiperparâmetros de um modelo e seleciona o subconjunto que apresenta melhores resultados preditivos. Existem diferentes métodos para fazer a otimização, como o *Random Search*, *Grid Search* e *Otimização Bayesiana*.

O método *Random Search* seleciona aleatoriamente conjuntos específicos dentro do espaço de possibilidade para avaliação. Essas amostras são utilizadas para treinar o modelo e permitem explorar um espaço de busca maior do que o *Grid Search*, já que não está restrito a um grid fixo. Além disso, o *Random Search* é facilmente paralelizável, pois cada avaliação de amostra é independente das demais.

Já o método *Grid Search* avalia todas as combinações possíveis de valores dentro de uma grade pré-definida para encontrar a melhor combinação, esta técnica pode ser custosa, mas interessante para pesquisar todo o espaço de possibilidades e é fácil de implementar. A sua principal desvantagem é a ineficiência em espaços de alta dimensionalidade, pois à medida que o número de hiperparâmetros aumenta, o número de interações cresce exponencialmente tornando o processo custoso.

A principal desvantagem desses dois métodos é que cada avaliação é realizada sem considerar os resultados anteriores, resultando em um desperdício de tempo ao testar várias combinações de hiperparâmetros que são independentes.

Dessa maneira, o método da *Otimização Bayesiana* não possui essa limitação já que essa técnica tem algoritmos interativos que determinam o ponto de avaliação futura com base no resultado obtido na interação passada, sendo uma abordagem probabilística eficaz para otimizações de funções objetivas do tipo "caixa preta", ou seja, funções nas quais não temos conhecimento sobre suas características (Brochu *et al.*, 2010).

Neste estudo, usamos a técnica de *Grid Search* no método de *Random Forest* pois ele

possui menos hiperpararâmetros a serem otimizados e queremos testar um grid pequeno de valores pré-definidos, já nos métodos de *Boosting*, como o *XGBoost* e *LightGBM* utilizamos a Otimização Bayesiana e a abordagem do Estimador de Parzen com Estrutura em Árvore (TPE).

A Otimização Bayesiana (OB) ou Otimização Sequencial Baseada em Modelo é um método vantajoso quando a avaliação da função é custosa, quando as derivadas não estão disponíveis, ou quando o problema não apresenta características de convexidade ou linearidade (Brochu *et al.*, 2010). Basicamente, é uma técnica usada para localizar o ponto máximo ou mínimo de uma função que pode não ter uma formulação explícita, mas da qual é possível obter amostras de entradas e saídas.

Segundo Pinheiro (2023) a Otimização Bayesiana é composta basicamente por 3 componentes. O primeiro é o modelo substituto que prevê a performance de cada possível combinação de hiperparâmetros, ou seja, visa atender a todos os pontos observados na função objetivo. Já o segundo é uma função de aquisição (ou seleção), que após obter a distribuição posterior das funções que descrevem a função a ser otimizada, retorna o ponto que equilibra o trade-off entre exploration e exploitation (**exploration** envolve amostrar em áreas ainda não exploradas, enquanto a **exploitation** foca em amostrar nas regiões que já parecem promissoras, onde o ótimo global é mais provável, com base nas informações da distribuição posterior). Por fim, o terceiro, mas opcional, é a utilização de uma técnica de inicialização com parâmetros previamente vencedores em problemas anteriores. Resumidamente, esse algoritmo atua seguindo os passos:

1. Construa um modelo substituto probabilístico para a função objetivo.
2. Encontre os valores ideais dos hiperparâmetros no modelo substituto.
3. Aplique esses valores de hiperparâmetros à função objetivo para avaliação.
4. Atualize o modelo substituto com os novos resultados.
5. Repita as etapas 2 a 4 até atingir o número máximo de iterações.

Ela possui duas vertentes principais, sendo estas:

- A técnica de Processo Gaussiano (Snoek *et al.*, 2012), que se baseia em um *kernel* para problemas de regressão não linear. Este método assume uma distribuição *a priori* do tipo gaussiana para modelar a relação entre os dados de treinamento,

restringindo a forma dessa distribuição de maneira inicial. À medida que novos dados de treinamento são apresentados, essa distribuição é atualizada para refletir a incerteza reduzida, ajustando sua forma com base nos dados observados. O kernel atua como uma função de similaridade entre os pontos de dados, permitindo capturar relações complexas, mesmo em problemas não lineares.

- A técnica do Estimador de Parzen com Estrutura em Árvore (TPE) (Bergstra *et al.*, 2011), na qual é denominado por este nome pelo fato de que podemos lidar com um espaço de busca estruturado em árvore e utilizar estimadores de Parzen, também conhecidos como estimadores de densidade por kernel (KDEs) (Watanabe, 2023), que é um método não paramétrico para estimar as curvas de densidade de uma variável aleatória. A TPE constrói modelos sequenciais iniciais sobre quais possuem os melhores hiperparâmetros do nosso modelo e atualiza o modelo fundamentado à medida que aprendemos como diferentes hiperparâmetros afetam o desempenho do modelo. Isso já representa uma melhoria significativa em relação ao *Grid Search* e ao *Random Search*.

A Figura 3.3 traz uma representação do conceito do TPE retirada do artigo de Watanabe (2023) conjuntamente com a legenda abaixo.

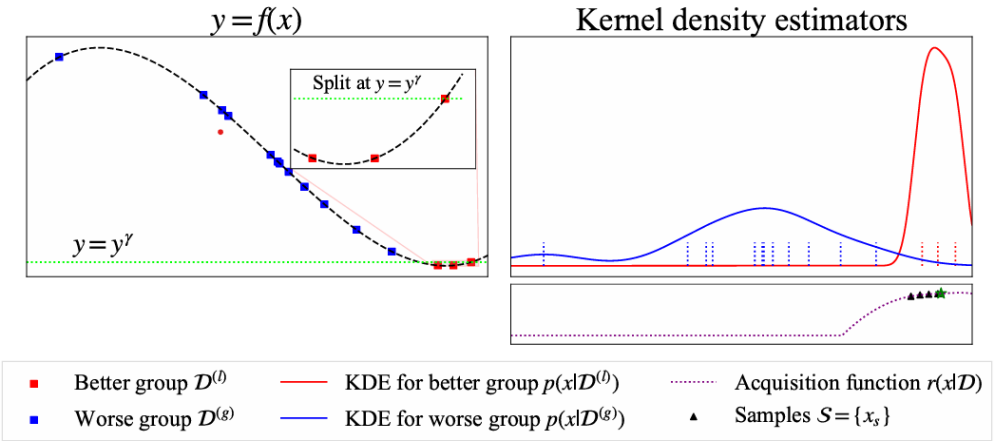


Figura 3.3: Visualização Conceitual do TPE. Fonte: Watanabe (2023).

O Artigo Watanabe (2023) traz a seguinte explicação: À esquerda, a função objetivo $y = f(x)$ (linha tracejada preta) e suas observações D para dois diferentes grupos ($D^{(l)}$: vermelhos e $D^{(g)}$: azuis) e a figura ampliada mostra a fronteira $y = y^{\gamma}$ (linha pontilhada verde) de D. Na Figura a direita, são apresentadas as estimativas de densidade de probabilidade (KDEs) de $D^{(l)}$ (linha sólida vermelha) e $D^{(g)}$ (linha sólida azul). No canto

inferior direito, a razão de densidade $p(x|D^{(l)})/p(x|D^{(g)})$ (linha pontilhada roxa) é utilizada na função de aquisição. A configuração com o melhor valor da função de aquisição é indicada pela estrela verde nas amostras (triângulos pretos) de $p(x|D^{(l)})$.

Aplicaremos a Otimização Bayesiana com a técnica do Estimador de Parzen com Estrutura em Árvore (TPE) (Bergstra *et al.*, 2011) nos modelos de XGBoost e LightGBM. Para tal, utilizamos a biblioteca **Optuna** da linguagem Python, explorada a seguir.

Optuna

A biblioteca utilizada para a Otimização Bayesiana no Python foi a **Optuna**. Esta biblioteca utiliza Tree-Structured Parzen Estimator como *default* e a função objetivo utilizada neste estudo será ajustada no treinamento e avaliada na amostra de validação por meio da métrica da área sob a Curva ROC (AUC). Essa biblioteca usa uma classe Trial para armazenar informações sobre uma combinação específica de hiperparâmetros, que serão aplicados posteriormente no modelo de aprendizado de máquina.

Um objeto de estudo pode ser utilizado para otimizar a função objetivo, identificando a melhor combinação de hiperparâmetros. Ele realizará experimentos iterativamente até que seja atingido o número máximo de tentativas ou o tempo limite especificado pelo usuário. O teste com a melhor configuração de hiperparâmetros será salvo em `study.best_trial` (Optunagoogl, 2021). Resumidamente, os principais passos são:

1. **Definição da função objetivo:** função para treinar o modelo e retornar a métrica de desempenho que se deseja otimizar. No nosso caso, iremos ajustar um *XGBoost* e *LightGBM* e maximizar a área sob a curva ROC (AUC);

2. **Criação de um estudo:** objeto que gerencia o processo de otimização, armazenando os resultados de cada teste (no nosso caso, trial = 30) e monitorando qual combinação de hiperparâmetros oferece o melhor desempenho;

3. **Sugerir hiperparâmetros:** utiliza como default a Otimização Bayesiana com abordagem TPE (Tree-structured Parzen Estimator), que modela a função de probabilidade dos resultados anteriores para sugerir melhores combinações no futuro;

4. **Execução dos trials:** executa a função objetivo repetidamente, cada vez sugerindo uma nova combinação de hiperparâmetros e ajusta os valores com base nos resultados dos testes anteriores para encontrar a melhor combinação possível;

5. **Armazena os melhores hiperparâmetros:** após a otimização, armazena o melhor trial com seus hiperparâmetros e o resultado da métrica avaliada.

Dessa forma, agora iremos explorar os algoritmos baseados em *Boosting*.

Boosting

O *Boosting* opera de forma semelhante ao *Bagging*, com a diferença de que, nele os classificadores fracos (geralmente árvores de decisão) são gerados de maneira sequencial, ou seja, cada árvore é construída incrementalmente, utilizando as informações das árvores anteriores e o processo não envolve amostragem por *bootstrap*. O fato de cada árvore ser ajustada em uma versão modificada do conjunto de dados original pretende que cada nova árvore se concentre em corrigir os erros das árvores anteriores sem muitas restrições na escolha da função de perda (James *et al.*, 2013), de modo que diminua o viés e aumente a variância da nova função de predição (Rocca, 2019). Além disso, a função é multiplicada por λ (*learning rate*) assumindo um valor entre 0 e 1 e tem como finalidade evitar o super-ajuste. Os passos a seguir sumarizam o algoritmo do *Boosting*:

1. Definimos uma função de predição inicial $g(\mathbf{x})$ e os resíduos $r_i = y_i$ para $i = 1, \ldots, n$.

2. Para cada árvore $b = 1, ..., B$, ajustamos uma nova árvore com p folhas para $(x_1, r_1), \ldots, (x_n, r_n)$ sendo $g_b(x)$ sua respectiva função de predição e atualizamos $g(x)$ e os resíduos:

$$g(\mathbf{x}) \leftarrow g(\mathbf{x}) + \lambda g_b(\mathbf{x})$$

$$r_i \leftarrow Y_i - g(\mathbf{x})$$

3. Retornamos o modelo final $g(\mathbf{x})$.

Gradient Boosting Machine

O *Gradient Boosting Machine* (GBM) foi desenvolvida por Jerome H. Friedman (Friedman, 2001) com objetivo de solucionar uma otimização numérica para minimizar a perda de um modelo. Ele é uma forma específica de *Boosting* e também constrói sequencialmente vários modelos "fracos" (geralmente árvores de decisão), mas utiliza uma métrica com erro diferenciável para aplicar o gradiente descendente, onde buscamos um valor de previsão que minimize a função de perda, contribuindo diretamente para o processo de

otimização dos resíduos e tornando-o mais eficiente e flexível (Anshul, 2024). Segundo Saini (2021) o GBM utiliza três componentes:

1. **Função de Perda:** sua principal característica é ser diferenciável. As funções mais utilizadas incluem o Erro Quadrático Médio, em problemas de regressão, e a Verossimilhança Logarítmica, em problemas de classificação.

2. **Modelo Fraco:** geralmente utilizamos árvores de decisão, que escolhem os melhores pontos de divisão com base em medidas de pureza como o coeficiente de Gini.

3. **Modelo Aditivo:** construímos árvores sequencialmente, adicionando uma por vez, sem alterar os casos já pertencentes ao modelo e usando o gradiente descendente com objetivo de minimizar a função de perda.

Gradient Boosting Decision Tree (GBDT)

O *Gradient Boosting Decision Tree* (GBDT) é um tipo de modelo de *ensemble* e *Gradient Boosting Machine* composto por árvores de decisão, que são treinadas de forma sequencial. A cada iteração, o modelo ajusta as previsões corrigindo as árvores por meio do uso de gradientes negativos ou erros residuais, o que permite melhorar o desempenho.

Segundo Ke *et al.* (2017) para o GBDT aprender uma função com base no conjunto de dados de entrada X e no espaço gradiente G, ele utiliza árvores de decisão como componentes de sua estrutura. Dado um conjunto de treinamento com n observações i.i.d., a cada iteração o gradiente negativo da função de perda em relação à saída do modelo é calculado, sendo representado como $\{g_1, g_2, ..., g_n\}$. Assim, a árvore de decisão divide cada nó no ponto que apresenta o maior ganho de informação, o qual pode ser frequentemente medido pela variância após a divisão.

A Figura 3.4 se refere a um fluxograma dos modelos não-lineares estudados neste trabalho desde o Random Forest até XGBoost e LightGBM sendo um resumo dos métodos estudados.

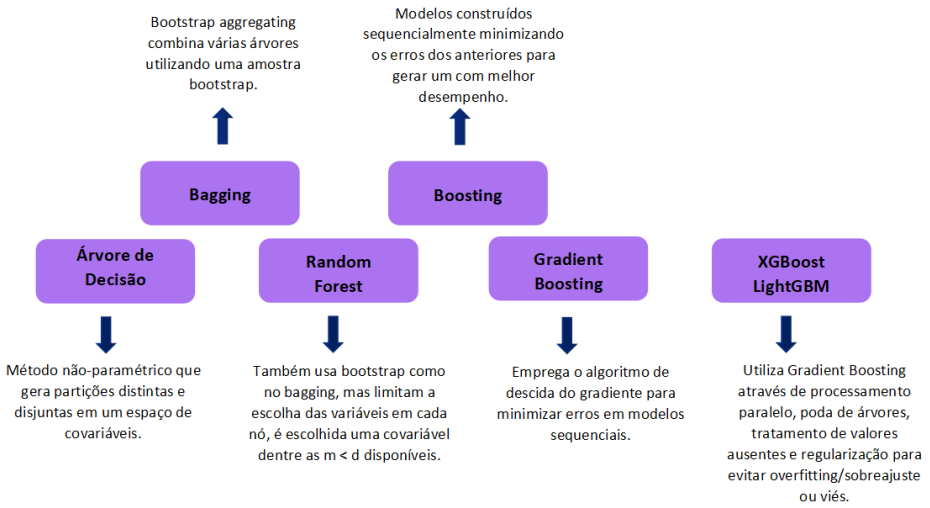
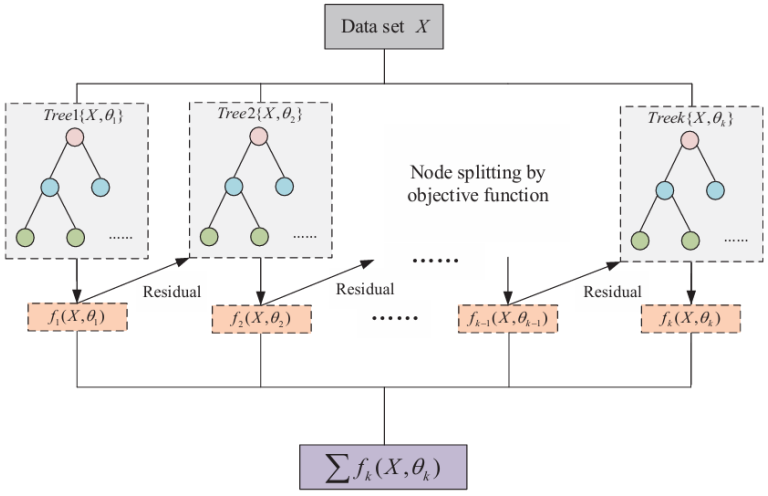


Figura 3.4: Representação: Fluxograma dos métodos não-lineares aplicados neste trabalho. Fonte: Autora.

3.4 XGBoost

O *eXtreme Gradient Boosting* (XGBoost) foi desenvolvido por Tianqi Chen como parte de sua pesquisa de doutorado e pode ser encontrado no artigo Chen e Guestrin (2016), desde então, este método se tornou popular na comunidade de dados, amplamente utilizado no mercado e em projetos de *Machine Learning*. Ele pertence à categoria de *Gradient Boosting* (Gradient Boosted Decision Trees, GBDT), ou seja, constrói sequencialmente várias árvores de decisão e utiliza uma métrica com erro diferenciável para aplicar o gradiente descendente. Nesse processo, busca-se um valor de previsão que minimize a função de perda, o que contribui de maneira precisa para a otimização do resíduo. A Figura 3.5 traz um fluxograma referente ao método *XGBoost*.



S

Figura 3.5: Fluxograma do XGBoost. Fonte: Guo *et al.* (2020).

Além disso, o *XGBoost* geralmente não requer padronização dos dados, ao ser baseado em árvores de decisão, que não são afetadas pela escala dos recursos. A seguir, exploraremos um exemplo simples que ilustra como o XGBoost faz uma predição para problemas de classificação. Este estudo visa aprofundar nossa compreensão sobre o processo de construção das árvores neste método, antes da discussão teórica. O exemplo foi retirado dos vídeos do canal meanxai (2024b).

1. Os dados para o exemplo estão representados na Tabela 3.1 e na Figura 3.6, se referem ao efeito (1 ou 0) para cada dosagem de um certo medicamento.

Tabela 3.1: Tabela de dosagem do remédio e seus efeitos.

Dosagem	**Efeito**	$\hat{y}_i^{(0)}$	$r_i^{(0)}$
3	0	0,5	-0,5
8	1	0,5	0,5
12	1	0,5	0,5
17	0	0,5	-0,5

Figura 3.6: Exemplo XGBoost. Fonte: Autora.

Eles contém as informações da dosagem, do efeito, da predição inicial ($y_i^{(0)}$) e os resíduos ($r_i^{(0)}$) referente a predição inicial, sendo i o número de observações, ou seja, $i = 1, 2, 3, 4$.

O primeiro passo é construir uma árvore, para tal, os resíduos iniciais estão contidos na raiz e a árvore é criada pela divisão dela.

2. Na segunda etapa, calcula-se a pontuação de similaridade para a raiz utilizando a fórmula a seguir:

$$score = \frac{\left(\sum_{i \in I_L} resíduos\right)^2}{\sum_{i \in I_L} h_i + \lambda}, \tag{3.5}$$

sendo $h_i = \hat{y}_i^{(0)}(1 - \hat{y}_i^{(0)})$, λ um parâmetro de regularização e L o total de covariáveis.

3. Na terceira etapa, dividimos a raiz usando todos os candidatos ao ponto de divisão da covariável x e em seguida, a pontuação de cada nó filho é calculada. Como exemplo, veja alguns dos possíveis candidatos a árvores na Figura 3.7.

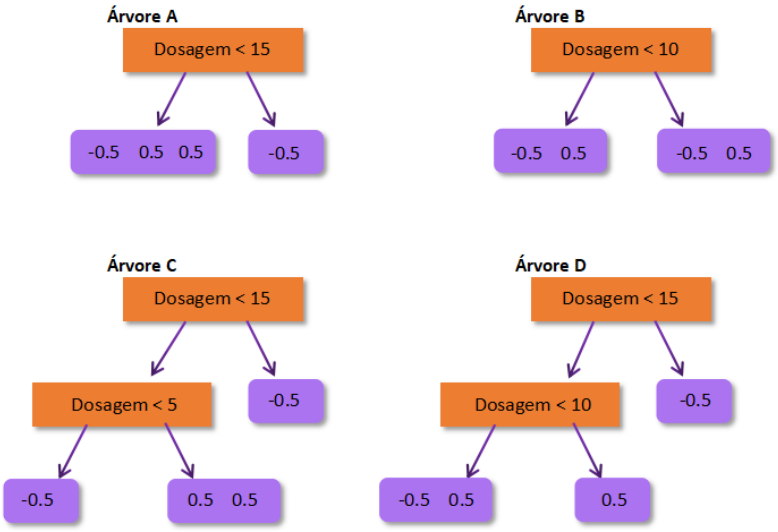


Figura 3.7: Exemplo XGBoost, sendo que os nós em laranja representam a regra da partição segundo a dosagem e os em roxo são os resíduos. Fonte: Autora.

Agora, calcularemos o score de similaridade para cada nó de cada árvore considerando $\lambda = 1$ e usando a Expressão 3.5. Por exemplo, para a árvore A, o $Score_{root}$ é a pontuação referente a todas as dosagens da raiz (usando os resíduos $-1/2, 1/2, 1/2, -1/2$) assim $Score_{root} = 0$, já o $Score_{left}$ é calculado com as dosagens menores do que 15 (dosagem $3, 8$ e 12) e respectivos resíduos $-1/2, 1/2, 1/2$, assim $Score_{left} = 0,14$ e de maneira análoga, o $Score_{right} = 0,2$ pois é referente a dosagem 17 e resíduo $-1/2$.

Lembrando que na primeira predição, assumimos todos os valores preditos como 1/2. As Tabelas 3.2 e 3.3 contém os cálculos das similaridade para as árvores da Figura 3.7.

Tabela 3.2: $Score_{left}$.

Árvore	Cálculo	Valor
A	$\frac{(-0,5+0,5+0,5)^2}{3\times0.5(1-0,5)+1}$	0,14
B	$\frac{(-0,5+0,5)^2}{2\times0,5(1-0.5)+1}$	0
C	$\frac{(-0,5)^2}{0,5(1-0,5)+1}$	0,2
D	$\frac{(-0,5+0,5)^2}{2\times0,5(1-0,5)+1}$	0

Tabela 3.3: $Score_{right}$.

Árvore	Cálculo	Valor
A	$\frac{(-0,5)^2}{1\times0,5(1-0,5)+1}$	0,2
B	$\frac{(-0,5+0,5)^2}{2\times0,5(1-0,5)+1}$	0
C	$\frac{(0,5+0,5)^2}{1\times2\times0,5(1-0,5)+1}$	0,67
D	$\frac{(0,5)^2}{1\times0,5(1-0,5)+1}$	0,2

4. Na quarta etapa, calculamos o *Gain* (Ganho) dado por:

$$Gain = score_{left} + score_{right} - score_{root} - \gamma.$$

O γ é um hiperparâmetro relacionado com a poda da árvore, apresentando a seguinte relação:

$$Gain = \begin{cases} \text{se for } \textbf{positivo}, \text{ então não pode.} \\ \text{se for } \textbf{negativo}, \text{ então pode.} \end{cases}$$

Dessa maneira, se consideramos $\gamma = 0,1$; o *Gain* das árvores da Figura 3.7 é dado na Tabela 3.4.

Tabela 3.4: Ganho das árvores.

Árvore	**Ganho**	**Valor**
A	0,14+0,2-0-0,1	0,24
B	0+0-0-0,1	-0,1
C	0,2+0,67-0,14-0,1	0,63
D	0+0,2–0,14-0,1	-0,04

Portanto, consideramos o hiperparâmetro de profundidade da árvore sendo 2 e ficamos com a árvore C já que seu ganho é maior.

5. Na quinta etapa, calculamos os valores de saída das folhas quando a árvore estiver finalizada, dado por:

$$\text{output value } (w_j) = \frac{\sum_{i \in I_j} resíduos_i^{(t)}}{\sum_{i \in I_j} \hat{y}_i^{(t)}(1 - \hat{y}_i^{(t)}) + \lambda}, \tag{3.6}$$

onde j é o número da folha e t referente a quantidade de árvores construídas.. Os valores de saída da árvore C são:

Tabela 3.5: Output da árvore C.

Output	**Cálculo**	**Valor**
$w_{j=1}$	$\frac{-0,5}{1 \times 0,5(1-0,5)+1}$	-0,4
$w_{j=2}$	$\frac{0,5+0,5}{2 \times 0,5(1-0,5)+1}$	0,67
$w_{j=3}$	$\frac{-0,5}{1 \times 0,5(1-0,5)+1}$	-0,4

6. Na sexta etapa, calculamos as previsões dadas pelas expressões:

$$F_i^{(1)} = F_i^{(0)} + \eta\, w_{i\in I_i} \quad (\eta\text{: taxa de aprendizado}).$$

Assumindo a taxa de aprendizado como $0,8$:

$$F_1^{(1)} = 0 + 0,8 \times (-0,4) = 0 - 0,32 = -0,32$$

$$\hat{y}_1^{(1)} = \frac{1}{1+\exp(0,32)} = 0,42$$

As seguintes observações se aplicam:

- $F_1^{(0)} = \log\left(\frac{y_1^{(0)}}{1-y_1^{(0)}}\right) = 0$
- $\hat{y}_i^{(1)} = \frac{1}{1+\exp(-F_i^{(1)})}$
- A probabilidade $y^{(0)}$ e log(odds) $F^{(0)}$ são intercambiáveis, pois é possível converter de um para o outro.
- $y^{(0)} \in [0,\, 1]$, $F^{(0)} \in (-\infty, +\infty)$

De maneira análoga, $\hat{y}_2^{(1)} = 0,63$,$\hat{y}_3^{(1)} = 0,63$ e $\hat{y}_4^{(1)} = 0,42$.

7. Calcule os novos resíduos usando as novas previsões calculadas no passo anterior e a Tabela 3.6 contém os resultados das novas predições.

$$r_1^{(1)} = y_1 - \hat{y}_1^{(1)} = 0 - 0,42 = -0,42$$
$$r_2^{(1)} = y_2 - \hat{y}_2^{(1)} = 1 - 0,63 = 0,37$$
$$r_3^{(1)} = y_3 - \hat{y}_3^{(1)} = 1 - 0,63 = 0,37$$
$$r_4^{(1)} = y_4 - \hat{y}_4^{(1)} = 0 - 0,42 = -0,42$$

Tabela 3.6: Tabela resumida com informações de dosagem, efeito do medicamento, previsões iniciais e resíduos.

i	**Dosagem**	**Efeito**	$\hat{y}_i^{(0)}$	$r_i^{(0)}$	$\hat{y}_i^{(1)}$	$r_i^{(1)}$
1	3	0	0,5	-0,5	0,42	-0,42
2	8	1	0,5	0,5	0,63	0,37
3	12	1	0,5	0,5	0,63	0,37
4	17	0	0,5	-0,5	0,42	-0,42

8. Repetir os passos agora com os resíduos $r_i^{(1)}$, por exemplo, considere a segunda árvore final como sendo:

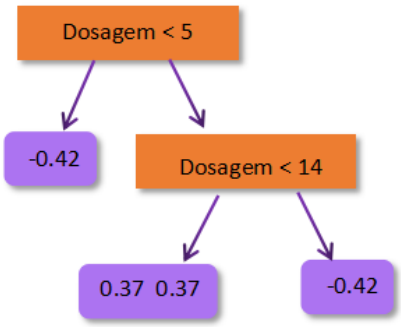


Figura 3.8: Segunda árvore do exemplo. Fonte: Autora.

Os valores de saída da árvore são:

Tabela 3.7: Output da árvore.

Output	**Cálculo**	**Valor**
$w_{j=1}$	$\frac{-0,42}{1 \times 0,42(1-0,42)+1}$	-0,338
$w_{j=2}$	$\frac{0,37+0,37}{2 \times 0,37(1-0,37)+1}$	0,505
$w_{j=3}$	$\frac{-0,42}{1 \times 0,42(1-0,42)+1}$	-0,338

Agora basta calcular as novas previsões usando os outputs e assim calcular os novos resíduos. Repetimos este passo até os resíduos serem bem pequenos.

9. Suponha agora que desejamos calcular, no conjunto de teste, a previsão para uma dosagem de 10 utilizando ambas as árvores apresentadas.

$$F^* = F_0 + \eta \cdot predict_1(x_{\text{test}}) + \eta \cdot predict_2(x_{\text{test}}) = 0 + 0,8 \cdot 0,67 + 0,8 \cdot 0,505 = 0,94$$

e então $y_{prob} = \dfrac{1}{1+exp(-0.94)} = 0,72$

Portanto, para a dosagem 10 estimamos $P(Y=0) = 0,72$.

Detalhes matemáticos para classificação

Nesta subseção, abordaremos alguns detalhes matemáticos deste método se baseando na documentação Chen e Guestrin (2016) e Guo *et al.* (2020).

1. Função de perda para classificação binária é dada por

$$L(y_i, \hat{y}_i) = -[y_i log(p_i) + (1 - y_i) log(1 - p_i)].$$

2. A função objetivo do XGBoost inclui duas partes: erro de treinamento e regularização, que é

$$L(\phi)_{obj} = \sum_{i=1}^{n} L(y_i, \hat{y}_i) + \sum_{b=1}^{B} \Omega(f_b), \tag{3.7}$$

na qual

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \|w\|^2,$$

em que $L(\hat{y}_i, y_i)$ representa a função de perda convexa e diferenciável entre o valor real y_i e previsto $\hat{y}_i$, B o número de árvores de decisão, $\Omega(f_b)$ penaliza a complexidade do modelo (ou seja, as funções da árvore de decisão) onde T é o número de nós terminais (folhas) em uma árvore, γ é uma penalidade definida pelo usuário destinada à poda e w é os valores preditivos atribuídos a cada folha. Dessa forma, o termo adicional de regularização ajuda a suavizar os pesos finais aprendidos para evitar o sobreajuste (Chen e Guestrin, 2016).

3. Utilizando a ideia de boosting por gradiente, temos que

$$\begin{cases} \hat{y}_i^{(0)} = 0,5 \\ \hat{y}_i^{(1)} = f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i) \\ \hat{y}_i^{(2)} = f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i) \\ \quad \vdots \\ y_i^{(B)} = \sum_{b=1}^{B} f_b(x_i) = \hat{y}_i^{(B-1)} + f_B(x_i) \end{cases} \tag{3.8}$$

4. Combinando a equação 3.7 e 3.8 podemos reescrever a função objetivo para a $B-$ésima por

$$L^B = \sum_{i=1}^{n} L(y_i, \hat{y}_i^{B-1} + f_B(X_i)) + \Omega(f_b), \tag{3.9}$$

sendo $f_B(X_i)$ função de entrada da $B-$ésima árvore de decisão.

5. Para encontrar a função objetivo que pode ser minimizada, fazemos a expansão de Taylor da função de perda até a segunda ordem. Assim, a função objetivo é

aproximadamente dada por:

$$L^B = \sum_{i=1}^{n} \left[L(y_i, \hat{y}_i^{(B-1)} + f_B(X_i)) + \frac{1}{2} h_i f_B^2(X_i) \right] + \gamma T$$

No lugar de olhar para todas as folhas da árvore, iremos observar apenas a uma, a folha t, cuja função de perda a ser minizada é:

$$L^B = \sum_{j=1}^{n} \left[\sum_{i \in I_j} (g_i) w_j + \frac{1}{2} \sum_{i \in I_j} (h_i + \lambda) w_j^2 \right] + \gamma T$$

$$L_j^B = \sum_{i \in I_j} (g_i) w_j + \frac{1}{2} \sum_{i \in I_j} (h_i + \lambda) w_j^2 + \lambda T$$

onde g_i é a primeira derivada da função de perda em relação ao preditos, h_i é a segunda derivada.

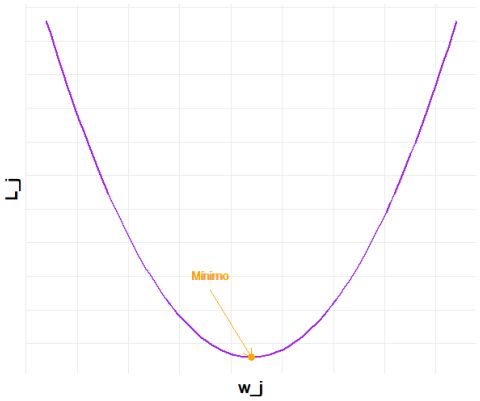


Figura 3.9: Representação da função de erro para uma folha, na qual é quadrática. Fonte: Autora.

Como o objetivo é encontrar o conjunto de pesos w que minimiza L e vemos que a função de erro para uma folha é quadrática, então os valores ótimos de w é a primeira derivada da função de perda igual a zero. Fazendo as manipulações, os valores da função objetivo obtidos pela solução são:

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

$$L_j^B = -\frac{1}{2} \sum_{j=1}^{T} \frac{G_j^2}{H_j + \lambda} + \gamma T,$$

onde $G_j = \sum_{i \in I_j} g_i$ e $H_j = \sum_{i \in I_j} h_i$. Na prática, essa equação será utilizada para analisar cada divisão em uma nova árvore, da mesma maneira que a entropia ou o coeficiente de Gini são comumente empregados na construção de árvores de decisão. A cada divisão, são criados dois nós: o esquerdo (left) e o direito (right). O ganho da divisão é calculado somando-se L_{left} (esquerdo) e L_{right} (direito), que correspondem às novas folhas, e subtraindo o erro anterior.

As equações abaixo sumarizando o calculo do ganho.

$$Gain = Left_{Similaridade} + Right_{Similaridade} - Root_{Similaridade} - \gamma$$

$$Gain = \frac{1}{2}\left[\frac{\left(\sum_{i \in I_{Left}} g_i\right)^2}{\sum_{i \in I_{Left}} h_i + \lambda} + \frac{\left(\sum_{i \in I_{Right}} g_i\right)^2}{\sum_{i \in I_{Right}} h_i + \lambda} - \frac{\left(\sum_{i \in I_j} g_i\right)^2}{\sum_{i \in I_j} h_i + \lambda}\right] - \gamma$$

Desse ponto, basta calcular todos os splits possíveis e escolher aquele que gera maior ganho.

Crescimento das árvores

O *XGBoost* divide as árvores se baseando no hiperparâmetro `max_depth` especificado e, em seguida, começa a podar a árvore para trás, removendo as divisões nas quais não há ganho positivo.

Ele utiliza como método padrão de crescimento das folhas a técnica de crescimento "level-wise" (nível a nível) para construir suas árvores, ou seja, o seu crescimento ocorre por linha em mais de uma folha. A Figura 3.10 traz uma representação do crescimento em uma árvore no método *XGBoost*.

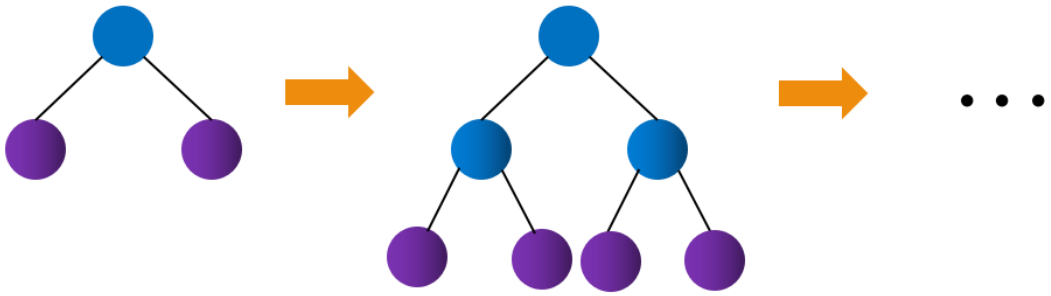


Figura 3.10: Representação: Level-wise tree growth. Fonte: Autora.

Divisões dos nós

A maioria das linguagens de programação pode adotar a busca exaustiva por todas as possibilidades (mais precisa, mas menos eficiente) ou optar pela busca baseada em histogramas (menos precisa, mas mais eficiente). Neste trabalho, abordaremos a última técnica.

Alguns hiperparâmetros

A seguir, apresentamos alguns hiperparâmetros do *XGBoost* que serão otimizados usando a técnica de Otimização Bayesiana.

- **learning_rate:** conhecida como taxa de aprendizado, controla o quanto o modelo ajusta os pesos de uma árvore em cada iteração;
- **n_estimators:** é o número de árvores de decisão que serão construídas no modelo;
- **max_depth:** é a profundidade máxima das árvores de decisão. Árvores mais profundas podem capturar mais complexidade dos dados, mas também aumentam a chance de *overfitting*;
- **colsample_bytree:** é a fração de covariáveis a serem usadas na construção de cada árvore;
- **subsample:** é a proporção de amostras a serem usadas para construir cada árvore. Valores muitos baixos podem provocar um desempenho baixo, mas reduzem a chance de *overfitting*, pois cada árvore verá apenas uma parte dos dados;
- **min_child_weight:** é peso mínimo necessário para subdividir uma folha;
- **gamma:** Controla o mínimo de ganho de perda necessário para que uma subdivisão em um nó ocorra. Se a subdivisão não reduzir a perda pelo menos esse valor, ela não será feita.

3.5 LightGBM

O *Light Gradient Boosting Machine* (LightGBM) foi desenvolvido pela Microsoft em 2016 com objetivo de lidar com bases de dados complexas e grandes, melhorando a eficiência no uso de memória e reduzindo o tempo de processamento. Esse algoritmo de aprendizado de máquina como o XGBoost também pertence à categoria *Ensemble* e

Gradient Boosted Decision Trees (GBDT), ou seja, os ajustes simples são realizados com árvores de decisão que utilizam gradiente para corrigir os resíduos. Essa abordagem é amplamente utilizada devido à sua alta eficiência, precisão e interpretabilidade.

O *LightGBM* geralmente não requer padronização dos dados, pois é baseado em árvores de decisão, que não são afetadas pela escala dos recursos. A seguir, apresentamos uma ideia de como as árvores do LightGBM são geradas baseada em um histograma altamente otimizado.

Para entender melhor como funciona construir uma árvore por meio do histograma altamente otimizado considere o seguinte exemplo retirado dos vídeos de meanxai (2024a). A Tabela 3.8 contém a covariável, variável resposta, predição inicial e o resíduo $g = (y - \hat{y_0})$.

Tabela 3.8: Dados do exemplo com predição inicial e diferenças $y - \hat{y_0}$.

ID	**x**	**y**	$\hat{y_0}$	$y - \hat{y_0}$
0	0,41	0	0,5	-0,5
1	0,22	0	0,5	-0,5
2	0,70	1	0,5	0,5
3	0,06	0	0,5	-0,5
4	0,48	0	0,5	-0,5
5	0,80	1	0,5	0,5
6	0,05	0	0,5	-0,5
7	0,72	1	0,5	0,5
8	0,34	0	0,5	-0,5
9	0,15	0	0,5	-0,5
10	0,60	1	0,5	0,5
11	0,31	1	0,5	0,5
12	0,01	0	0,5	-0,5
13	0,02	0	0,5	-0,5
14	0,29	1	0,5	0,5
15	0,55	1	0,5	0,5
16	0,53	1	0,5	0,5
17	0,22	0	0,5	-0,5
18	0,40	0	0,5	-0,5
19	0,98	1	0,5	0,5

Por meio dos dados podemos construir um histograma considerando o número de categorias como 5, por exemplo. A Tabela 3.9 contém a distribuição dos dados segundo esses intervalos e com base nela construímos o histograma da Figura 3.11.

Tabela 3.9: Tabela de frequência conforme os intervalos de x.

	Bin	**ID**	Σg
$0 \leq x < 0,2$	0	3, 6, 9, 12, 13	$-0,5-0,5-0,5-0,5-0,5=-2,5$
$0,2 \leq x < 0,4$	1	1, 8, 11, 14, 17	$-0,5-0,5+0,5+0,5-0,5=-0,5$
$0,4 \leq x < 0,6$	2	0, 4, 15, 16, 18	$-0,5-0,5+0,5+0,5-0,5=-0,5$
$0,6 \leq x < 0,8$	3	2, 7, 10	$0,5+0,5+0,5=1,5$
$0,8 \leq x < 1,0$	4	5, 19	$0,5+0,5=1,0$

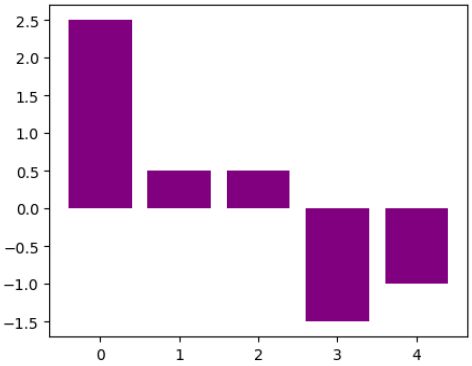


Figura 3.11: Histograma da distribuição de frequência de x.

Da mesma forma que calculamos o ganho de cada divisão no XGBoost faremos aqui usando a expressão 3.5 considerando $\lambda = 0$ mas analisaremos os intervalos segundo a Tabela 3.9 e não todas as possibilidades de divisões. A Tabela 3.10 contém os devidos cálculos para cada intervalo, concluímos que o intervalo com maior ganho é $x < 0,6$, portanto, a árvore gerada será com a raiz sendo $x < 0,6$.

Tabela 3.10: Ganho para cada possível divisão considerando $\lambda = 0$.

Raiz	Σg **(left)**	Σg **(right)**	**Score (left)**	**Score (right)**	**Ganho**
0,2	-2,5	1.5	$\frac{(-2,5)^2}{(0,25*5)} = 5,0$	$\frac{1,5^2}{(0,25*15)} = 0,6$	5,6
0,4	-3,0	2.0	$\frac{(-3,0)^2}{(0,25*10)} = 3,6$	$\frac{2,0^2}{(0,25*10)} = 1,6$	5,2
0,6	-3,5	2,5	$\frac{(-3,5)^2}{(0,25*15)} = 3,3$	$\frac{2,5^2}{(0,25*5)} = 5,0$	8,3
0,8	-2,0	1,0	$\frac{(-2,0)^2}{(0,25*18)} = 0,9$	$\frac{1,0^2}{(0,25*2)} = 2,0$	2,9

O método continua com a construção das árvores, escolhendo as divisões com maiores ganhos como no *XGBoost*.

Neste trabalho, utilizaremos o método de histogramas altamente otimizados no *XGBoost* e no *LightGBM* devido à sua eficiência. Embora seja possível enumerar todos os pontos de divisão, mas essa abordagem é computacionalmente muito custosa e preferível para banco de dados menores.

GOSS

A *Gradient-Based One-Side Sampling* (GOSS) ou *Amostragem de um Lado Baseada em Gradiente* foi proposto como um método de amostragem para GBDT que pode alcançar um bom equilíbrio entre a redução do número de instâncias de dados e a manutenção da precisão das árvores de decisão geradas. Segundo Ke *et al.* (2017) o método GOSS no *LightGBM* prioriza a retenção de instâncias de dados com gradientes maiores, pois essas contribuem mais para o cálculo do ganho de informação. Em vez de amostrar dados de forma aleatória, o GOSS mantém as instâncias com gradientes grandes (mais relevantes) e faz uma amostragem aleatória com as de gradientes pequenos, o que melhora a precisão da estimativa do ganho de informação em comparação com a amostragem aleatória uniforme, especialmente quando o ganho de informação varia muito. Portanto, podemos utilizar o GOSS como método de amostragem eficiente no *LightGBM* para reduzir a magnitude dos dados sem perder informações significativas. Em resumo, o algoritmo GOSS consiste em:

1. É realizada a ordenação decrescente das instâncias utilizadas no treinamento, conforme os valores absolutos de seus gradientes;

2. Com base em um valor pré-estabelecido, α, são mantidas na amostra, seguindo a ordem previamente determinada, as $\alpha \times 100\%$ observações com os maiores gradientes, formando o subconjunto A;

3. Para as observações com menores gradientes (o complementar do subconjunto A) com representatividade de $(1 - \alpha) \times 100\%$ retiramos uma amostra aleatória denominada de subconjunto B de tamanho $b \cdot |A^c|$;

4. Por último, é feita a divisão das observações com base no ganho de informação (variância) sob o conjunto $A \cup B$.

Observe que quando $\alpha = 0$ temos uma amostragem aleatória simples (Ke *et al.*, 2017).

EFB

Em aplicações reais, é comum encontrar banco de dados com um grande número de características e com espaço esparso, geralmente busca-se reduzir o número de características, filtrando as não relevantes sem grandes perdas. Um exemplo de método

de redução de dimensionalidade após a filtragem de covariáveis é Componentes Principais (PCA), mas dependendo do tipo de problema as suas suposições são inválidas. No LightGBM, foi proposto o método Exclusive Feature Bundling (EFB) ou *Método de Agrupamento de Recursos Exclusivos* como meio de melhorar a eficiência através da redução de recursos.

De acordo com Ke *et al.* (2017), muitas características são mutuamente exclusivas, ou seja, nunca assumem valores não nulos ao mesmo tempo. Essas características podem ser agrupadas em um único "pacote de características exclusivas" utilizando um algoritmo de varredura cuidadosamente projetado, é possível construir histogramas a partir desses pacotes de forma idêntica aos histogramas que seriam gerados pelas características individuais. Isso reduz a complexidade de $O(nd)$ para $O(nq)$, onde n é o número de informações na base de dados, d é o número de características e q é a quantidade de características após a aplicação do método de empacotamento e desde que $q < d$. É importante determinar quais recursos devem ser agrupados e como construir o empacotamento.

Crescimento das árvores

O crescimento das árvores no *LightGBM* ocorre por nível, ou seja, em cada nível é escolhida apenas uma folha para crescer e geralmente se escolhe aquela que reduz a perda, por isso, denominamos de crescimento em *Leaf-wise tree growth*. Diferentemente do *XGBoost* onde o seu crescimento ocorre por linha em mais de uma folha. A Figura 3.12 traz uma representação do crescimento em uma árvore no método *LightGBM*.

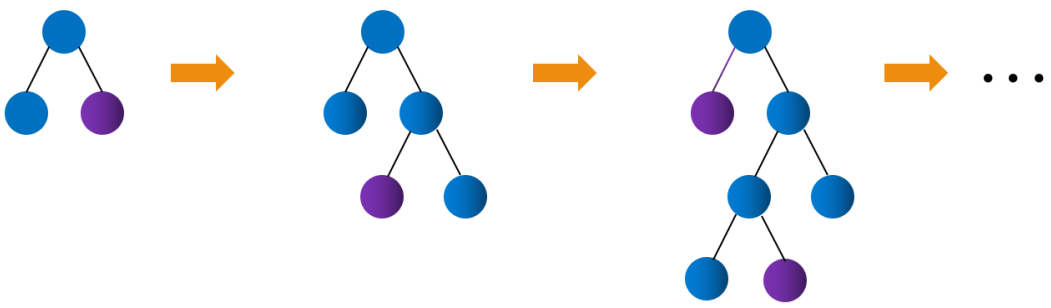


Figura 3.12: Representação: Leaf-wise tree growth. Fonte: Autora.

Alguns hiperparâmetros

Alguns hiperparâmetros mencionados no *XGBoost* também foram analisados no *LightGBM* como learning_rate, n_estimators, max_depth, subsample, colsample_bytree. Já outros hiperparâmetros que foram otimizados:

- **num_leaves:** número máximo de folhas de uma árvore única,
- **min_child_samples:** é o número mínimo de amostras que uma folha deve conter após uma divisão,
- **min_split_gain:** ganho mínimo necessário para uma divisão ser feita em um nó da árvore.

3.6 Métricas de desempenho

Para avaliar o desempenho do modelo de classificação dada uma função de predição $g(\mathbf{x})$ podemos utilizar algumas métricas de desempenho na amostra de teste.

3.6.1 Função de Risco

A função de risco é a proporção de erros estimados em novas observações. Portanto, a função de perda considerada é a indicadora da diferença entre valores preditos e verdadeiros usando a amostra de teste. A equação a seguir representa essa relação:

$$R(g) := E[\mathbb{I}(Y \neq g(\mathbf{X}))] = P(Y \neq g(\mathbf{X})). \tag{3.10}$$

Em certas ocasiões de desbalanceamento das classes de Y a função de risco utilizada para estimar o poder preditivo de um método não fornece todas as informações necessárias sobre o classificador pois ela tende a estimar a classe majoritária com alta precisão. Esse é o caso presente, pois há poucos clientes classificados como 1 (observamos que essa taxa é de 5.37%). Nesse contexto, é esperado que $g(\mathbf{x})$ apresente um baixo risco, porém seu desempenho pode não ser eficiente. A seguir, apresentaremos outras medidas empregadas para avaliar o desempenho de modelos de classificação utilizando a amostra de teste.

Utilizando o conjunto de teste, podemos também fazer um intervalo de confiança para o risco. Denotamos por $(\boldsymbol{X_1}, Y_1), ..., (\boldsymbol{X_n}, Y_n)$ os elementos do conjunto de teste, um estimador usual para o risco da função de predição (estimada no treinamento) é:

$$\hat{R}(g) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}\left(Y_i \neq g(\boldsymbol{X_i})\right).$$

Como $\hat{R}(g)$ é uma média de variáveis independentes e identicamente distribuídas, pelo Teorema do Limite Central e considerando n suficientemente grande, temos que:

$$\hat{R}(g) \approx Normal(R(g), \frac{1}{n}\mathbb{V}(g)),$$

de forma que $\mathbb{V}(g)$ corresponde a variância da função de predição, estimada por $\hat{\sigma}^2$. Dessa maneira, o intervalo de confiança aproximado de 95% é dado por:

$$\hat{R}(g) \pm 1,96 \cdot \sqrt{\frac{1}{n}\hat{\sigma}^2}.$$

3.6.2 Matriz de Confusão

Como já mencionado, nesse trabalho usaremos *Data Splitting* e o conjunto de dados é dividido em duas amostras: treinamento e teste. Com a primeira amostra iremos encontrar a função de predição e com a segunda amostra avaliaremos o modelo por meio de métricas de desempenho.

Uma das formas de realizar a avaliação do desempenho do modelo é usar a matriz de confusão representada na Tabela 3.11, na qual apresenta a relação entre valores preditos e verdadeiros usando a amostra de **teste** e considerando a classe positiva sendo a minoritária ($Y = 1$).

Tabela 3.11: Matriz de Confusão.

	Classe Real	
Classe Predita	Y = 0	Y = 1
Y = 0	Verdadeiro Negativo (VN)	Falso Negativo (FN)
Y = 1	Falso Positivo (FP)	Verdadeiro Positivo (VP)

De maneira sucinta, explicaremos as entradas da matriz de confusão:

- **VN** se refere aos casos classificados corretamente como pertencente à classe negativa. No nosso caso, representam o número de clientes bons que foram corretamente classificados como bons;

- **VP** são os casos classificados corretamente como pertencentes à classe positiva.

No nosso caso, representam o número de clientes maus que foram corretamente classificados como maus;

- **FN** são os casos classificados incorretamente como pertencentes à classe negativa. No nosso caso, representam o número de clientes maus tendo sido classificados como bons;

- **FP** se refere aos casos classificados incorretamente como pertencentes à classe positiva. No nosso caso, representam o número de clientes bons tendo sido classificados como maus.

A partir da Tabela 3.11 podemos definir algumas medidas de desempenho, descritas a seguir (Izbicki e dos Santos, 2020):

- **Sensibilidade/Recall:** é a estimativa da probabilidade de classificar indivíduos na classe positiva corretamente entre todas as reais observações positivas. No nosso caso, é a proporção de clientes classificados como maus dentre aqueles que de fato são maus. É expressa por $S = \frac{VP}{(VP + FN)}$;

- **Especificidade:** é a estimativa da probabilidade de classificar indivíduos na classe negativa corretamente entre todas as reais observações negativas. No nosso caso, é a proporção de clientes classificados como bons dentre aqueles que de fato são bons. É expressa por $E = \frac{VN}{(VN + FP)}$;

- **Valor Preditivo Positivo/Precisão:** é a estimativa da probabilidade dos clientes classificados corretamente como positivos em relação ao total de positivos preditos. No nosso caso, é a proporção de clientes classificados corretamente como maus dentre todos os preditos como maus. É expressa por $VPP = \frac{VP}{(VP + FP)}$;

- **Valor Preditivo Negativo:** é a estimativa da probabilidade dos clientes classificados corretamente como negativos em relação ao total de negativos preditos. No nosso caso, é a proporção de clientes classificados corretamente como bons dentre todos os preditos como bons. É expressa por $VPN = \frac{VN}{(VN + FN)}$;

- **F1 Score:** se refere a média harmônica entre Recall e Precisão, é matematicamente a expressão $F1 = 2 \cdot \frac{\mathrm{VPP} \cdot \mathrm{S}}{\mathrm{VPP} + \mathrm{S}}$;

- **Acurácia:** é a estimativa da probabilidade das classificações corretas em relação ao total de indivíduos. No nosso caso, é a proporção de clientes bons classificados como bons e os clientes maus classificados como maus com relação ao total de clientes na amostra. É expressa por $Acurácia = \frac{(VN + VP)}{(VN + FN + FP + VP)}$.

3.6.3 Desbalanceamento das Classes

Muitas vezes, temos situações de desbalanceamentos de classes de Y, isto é, existe um grande desequilíbrio entre as observações das classes da variável resposta. No nosso caso, a proporção de maus pagadores é de aproximadamente $5,37\%$ refletindo esse desequilíbrio. Desse modo, queremos um classificador que não identifique futuras observações de forma tão errada, pois nesse contexto ele pode apresentar uma tendência a favorecer a classe majoritária. Nesta situação, uma abordagem é usar o classificador de Bayes.

Para isso, a função de perda mais adequada para o contexto de classificação é $L(g(\mathbf{x}), Y) = \mathbb{I}(Y \neq g(\mathbf{X}))$, denominada de função de perda 0-1. A melhor função de classificação g de acordo com Equação 3.10, é dada por:

$$g(\mathbf{x}) = \arg\max_{d \in C} P(Y = d \mid \mathbf{x}),$$

ou seja, classificamos a observação como sendo da classe com maior probabilidade a posteriori. Esse classificador é conhecido como classificador de Bayes e para o caso binário pode ser reescrito como:

$$g(\mathbf{x}) = 1 \Leftrightarrow \hat{\mathbb{P}}(Y = 1|\mathbf{x}) \geq \frac{1}{2},$$

isto é, dado um conjunto de covariáveis $\mathbf{x}$ classificamos $Y = 1$ se a probabilidade estimada for maior ou igual a $0,5$.

Quando se lida com o desbalanceamento entre classes em problemas de classificação, uma estratégia comum é ajustar o corte K para valores diferentes de $0,5$ nos classificadores probabilísticos e definimos a função de classificação como:

$$g(\mathbf{x}) = \mathbb{I}(\hat{\mathbb{P}}(Y = 1|\mathbf{x}) \geq K),$$

para diferentes cortes de valor K.

Existem diversas abordagens para selecionar o valor do ponto de corte K. Neste estudo, usamos a Curva ROC (*Receiver Operating Characteristic Curve*), este gráfico de

diagnóstico é construído com o conjunto de teste e ilustra a relação entre a sensibilidade e a especificidade do modelo para diferentes valores de corte. Na Curva ROC, uma linha diagonal serve como referência para avaliar o desempenho dos modelos; um desempenho acima dessa linha indica que o modelo é superior ao acaso, enquanto um desempenho abaixo da linha sugere que o modelo é inferior ao acaso. Assim, é possível escolher o ponto de corte desejado observando a variação entre a sensibilidade e a especificidade.

A partir da Curva ROC, definimos um ponto de corte maximizando a estatística de Youden. Essa métrica leva em consideração tanto a sensibilidade (Recall) quanto a especificidade do modelo, dada por:

$$\text{J} = \text{Sensibilidade} + \text{Especificidade} - 1.$$

Também iremos utilizar a Curva de Precisão-Recall para calcular o ponto de corte K, que se concentra especificamente em avaliar o desempenho de um classificador em relação à classe minoritária (classe positiva). Essa curva mostra a variação entre Precisão e Recall para diferentes valores de corte, e podemos definir K maximizando a média harmônica entre essas duas métricas, conhecida como *F1*, definida anteriormente pela seguinte expressão:

$$\text{F1} = 2 \cdot \frac{\text{Precisão} \cdot \text{Recall}}{\text{Precisão} + \text{Recall}}.$$

Além disso, também faremos a média das predições estimadas usando o conjunto de teste para escolher o valor de K, ou seja:

$$K = mean(previsões). \tag{3.11}$$

Portanto, para encontrar o valor do corte K analisaremos essas 3 abordagens: otimização da estatística de Youden, otimização da $F1$ e média das predições estimadas.

3.6.4 Outras Métricas de Avaliação

A partir da Curva ROC, é possível calcular a AUC (*Area Under the Curve*), que avalia o desempenho preditivo do modelo, sendo que, quanto maior essa métrica, melhor é a capacidade preditiva do modelo. Além disso, o coeficiente de Gini também pode ser calculado a partir da Curva ROC, sendo determinado pela expressão:

$$Gini : 2 \cdot AUC - 1,$$

demonstrando a mesma relação da AUC (Assunção, 2012).

A última métrica de desempenho analisada neste estudo será a estatística do **Teste de Kolmogorov-Smirnov**, um teste não paramétrico utilizado para verificar se as funções de distribuição de duas populações independentes são iguais. Embora não tenha sido proposta com esse objetivo, é muito utilizada na área de *credit scoring* para avaliar o poder preditivo de um modelo, usado em trabalhos como de Luchetta (2019) e Forti (2018). Neste caso, desejamos comparar a distribuição dos escores (probabilidade multiplicada por 1000) entre clientes bons (B) e maus (M) pagadores. Para isso, consideramos F_B como a função de distribuição da probabilidade prevista para os bons pagadores e F_M como a função de distribuição da probabilidade prevista para os maus pagadores. Assim, as hipóteses do teste são:

$$\begin{aligned} H_0 : &\quad F_B(x) = F_M(x) \quad \text{para todo } x \in \mathbb{R}, \\ H_1 : &\quad F_B(x) \neq F_M(x) \quad \text{para pelo menos um valor de } x. \end{aligned}$$

A estatística é em homenagem aos matemáticos russos Andrei Kolmogorov e Nikolai Smirnov (Barakat *et al.*, 2019), dada por:

$$KS = \max |F_B(x) - F_M(x)|,$$

em que $F_B(x)$ e $F_M(x)$ correspondem a proporção de clientes bons e maus com escore menor ou igual a x, respectivamente.

Desse modo, quanto maior for a estatística de KS maior será a distância entre as proporções acumuladas e maior a discriminação entre os clientes adimplentes e inadimplentes. Segundo Corrêa e Vellasco (2008) uma boa discriminação é alcançada quando a estatística KS está entre 30% e 40%. Valores acima desse intervalo indicam um nível

excelente de discriminação. A Figura 3.13 representa a curva de distribuição acumuladas para as duas populações em estudo com a nossa base de dados desbalanceada, contendo bons e maus pagadores na variável resposta. Ao identificar o ponto de maior distância vertical entre as duas curvas, o valor do KS será a diferença percentual entre as duas curvas nesse ponto (representado pela linha tracejada).

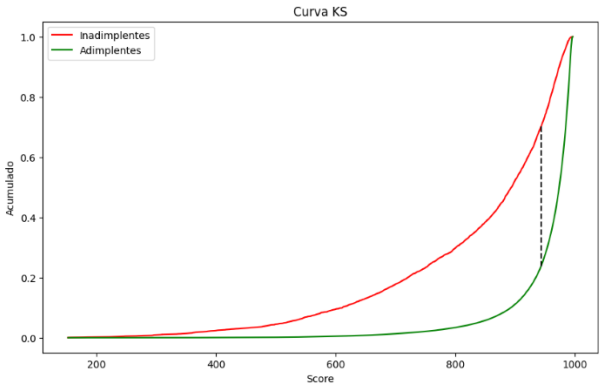


Figura 3.13: Representação: Funções distribuições para os bons e maus clientes e a estatística KS. Fonte: Autora.

Capítulo 4

Aplicação

4.1 Banco de Dados

A base de dados descaracterizada utilizada neste trabalho foi fornecida pela empresa Stepwise e contém informações sobre 411.469 clientes com um cartão de varejo, abrangendo o período de dezembro de 2021 a setembro de 2022. Ela possui 213 *features* sendo numéricas ou categóricas, e algumas dessas características têm dados faltantes (NA's). A variável resposta é binária e se refere ao tipo de cliente da loja, definida como:

$$Y = \begin{cases} 1, & \text{se o cliente é classificado como mau pagador.} \\ 0, & \text{se o cliente é classificado como bom pagador.} \end{cases}$$

Com a variável resposta podemos calcular a Taxa de Inadimplência, uma métrica amplamente conhecida no mercado, que se refere a proporção de clientes que são maus pagadores ($Y = 1$) em relação ao total.

Dentre as *features* totais, se encontram as variáveis relacionadas com o comportamento do cliente que possui o cartão, como o tempo em meses até a primeira compra pelo cartão, tempo em meses até fazer o 1º cartão da loja, histórico de transações, vencimentos, atrasos e as variáveis cadastrais do cliente como idade, gênero, características do CEP de residência, renda, entre outras. As variáveis estão nomeadas de maneira descaracterizada e a seguir apresentaremos a Tabela 4.1 com algumas *features* e sua descrição. O dicionário de todas as covariáveis pode ser encontrado no Apêndice A deste trabalho.

Tabela 4.1: Algumas covariáveis que compõe o banco de dados.

Variável	Descrição
var_1	Gênero do cliente
var_3	Tempo até a primeira compra com o cartão
var_14	Máximo de dias de vencimentos atrasados nos últimos 3 meses
var_21	Média de dias de vencimentos atrasados nos últimos 3 meses
var_39	Média de dias de vencimentos antecipados nos últimos 3 meses
var_55	Quantidade de contratos nos últimos 6 meses
var_63	Máxima quantidade de parcelas de contratos dos últimos 12 meses
var_69	Quantidade de vencimentos antecipados no último mês
var_75	Soma dos valores dos vencimentos antecipados nos últimos 3 meses
var_100	Quantidade de vencimentos pontualmente nos últimos 3 meses
var_240	Percentual de vencimentos quitados no último ano em relação renda
var_251	Renda do cliente
var_339	Indicador de endereço desfavorecido socialmente
var_275	Indicador de auxílios governamentais
var_421	Percentual de esgoto na região do CEP do titular

Como os valores ausentes se referem as situações que os clientes não têm algum vencimento para quitar no período ou que já foi quitado, podemos imputar pelo valor 0 sem nenhuma perda de informação e também consideramos como filtro somente pessoas com idade maior do que 16 anos para modelar o score. No final, a base possui 5 variáveis categóricas e 208 variáveis contínuas. As variáveis categóricas incluem o indicador de auxílio emergencial, gênero, indicador de endereço desfavorecido, indicador de UPA e classe econômica.

O script do tratamento de dados, descrição e modelagem está disponível no GitHub da autora no seguinte link: Modelos para prever um Behaviour Score.

4.2 Análise Descritiva

Inicialmente, realizamos uma análise descritiva envolvendo a variável resposta. Podemos calcular a proporção de maus clientes (quando $Y = 1$) em relação ao total, neste trabalho, ou seja a "Taxa de Inadimplência"para nos referirmos a essa proporção. Dessa maneira, podem ser observadas informações sobre essa medida por período na Tabela 4.2 e Figura 4.1.

Observe que a variável resposta é pouco frequente já que $P(Y = 1) = 0,0537$. Desse modo, temos indicativos de um desbalanceamento das classes da variável resposta e possivelmente teremos que alterar o ponto de corte no momento da classificação.

Tabela 4.2: Dados sobre a variável resposta e taxa de inadimplência.

Período	Total de Clientes	Y = 1	P(Y=1)
Dez/2021	82.131	4.296	5,23%
Fev/2022	82.592	5.170	6,26%
Mai/2022	82.076	4.594	5,60%
Jul/2022	81.762	4.308	5,27%
Set/2022	82.908	3.752	4,52%
Total	411.469	22.120	5,37%

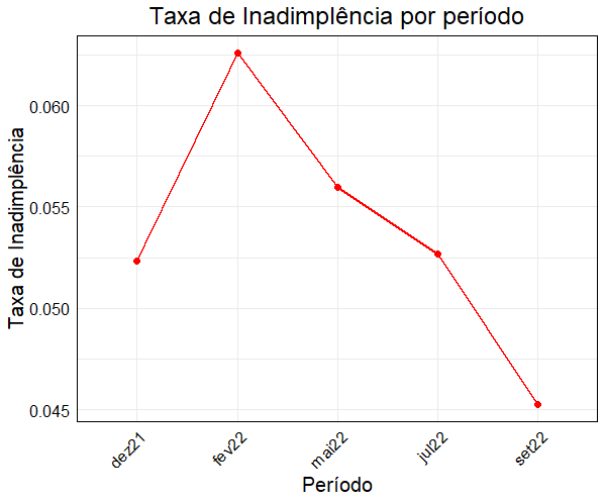


Figura 4.1: Gráfico referente a taxa de inadimplência em cada período.

Considerando que o banco de dados possui uma grande quantidade de variáveis selecionamos algumas covariáveis para uma análise descritiva de acordo com o objetivo do trabalho. É a partir da análise descritiva que iremos ter uma primeira visão sobre o público da base de dados, detectar valores discrepantes e variáveis mais significativas na discriminação das classes Y.

Análise de variáveis de cadastro

Inicialmente, analisamos as variáveis de cadastros do banco de dados. A Tabela 4.3 se refere a análise descritiva de todas as variáveis de cadastro dos clientes que são categóricas contendo informações das categoria, frequência e proporção de maus pagadores por categoria.

Tabela 4.3: Resumo das covariáveis categóricas.

Covariável	Categoria	Total	Frequência	Y=1	P(Y=1)
Auxílio Emergencial	NA	361651	87,90%	18311	5,06%
	Não	5080	1,23%	628	12,36%
	Sim	44738	10,87%	3181	7,11%
EDS	Não	404822	98,38%	21556	5,32%
	Sim	6647	1,62%	564	8,49%
Gênero	Feminino	247458	60,14%	11575	4,68%
	Masculino	164011	39,86%	10545	6,43%
Classe Econômica	A	761	0,19%	54	7,09%
	B	44031	10,70%	2328	5,29%
	C	241031	58,58%	11832	4,91%
	D	110616	26,88%	6839	6,18%
	E	15030	3,65%	1067	7,10%
UPA	Não	198241	48,18%	9941	5,01%
	Sim	213228	51,82%	12179	5,71%

Com a análise da Tabela 4.3 podemos ter alguns indicativos, ao analisar a variável *Auxílio Emergencial*, vemos que a maioria do público não possui informações sobre esse auxílio $(87,90\%)$, e aqueles que não receberam esse benefício apresentam a maior proporção de inadimplência $(12,36\%)$. Além disso, ao examinar a variável categórica de *EDS (Endereço Desfavorecido Socialmente)*, observou-se que a maioria do público não reside em áreas socialmente desfavorecidas $(98,38\%)$, e esse grupo apresenta a menor proporção de inadimplentes. No que se refere à variável de *Gênero*, verificou-se que aproximadamente $60,14\%$ dos clientes são mulheres, sendo que elas apresentam uma taxa de inadimplência menor do que a dos homens.

Além disso, ao analisar a variável *Classe Econômica* dos clientes, observa-se que aproximadamente $58,58\%$ dos clientes pertencem à classe econômica C, a qual apresenta a menor proporção de inadimplentes, correspondendo a $4,91\%$. Ademais, verifica-se que uma parcela reduzida dos clientes pertence à classe econômica A $(0,19\%)$, cuja taxa de inadimplência é próxima à verificada na classe econômica E. Por fim, ao analisar a presença de *UPA* na região do CEP dos clientes, constatou-se que a maioria reside próxima a uma UPA $(51,82\%)$, contudo, a taxa de inadimplência nesse grupo é semelhante à daqueles que não residem em áreas próximas a uma UPA, sendo a única variável categórica que não possui uma discriminação forte aparente das classes de Y.

Agora, analisamos algumas variáveis contínuas de cadastro, descritas abaixo:

- **Idade**(var_2): Idade do cliente em anos,
- **Renda** (var_251): Renda do cliente em reais,

- **Ensino Médio** (var_403): Porcentagem de pessoas que cursam o ensino médio no CEP do cliente,
- **Benefícios sociais** (var_410): Porcentagem de pessoas recebem benefício social no CEP do cliente,
- **Aluguel** (var_394): Aluguel estimado do CEP do cliente em salários mínimos,
- **Única pessoa** (var_393): Porcentagem de pessoas que moram sozinhas no CEP do cliente.

A Tabela 4.4 apresenta as medidas resumo dessas variáveis de cadastro e a Figura 4.2 ilustra a relação das variáveis com a variável resposta por meio de *Boxplots*, fornecendo uma visão mais clara das possíveis associações entre elas.

Tabela 4.4: Resumo Estatístico de algumas variáveis contínuas.

Covariável	Mínimo	1º Quartil	2º Quartil	Média	3º Quartil	Máximo
Idade	17	31	42	43,23	54	111
Renda	0	3.150	3.900	4.093	4.700	50.800
Aluguel	0,10	0,57	0,68	0,72	0,84	6,55
Ensino médio	5,53%	13,51%	14,91%	14,98%	16,43%	33,83%
Benefícios sociais	0%	2,10%	2,60%	2,68 %	3,19%	18,16%
Única pessoa	2,61%	8,91%	10,48%	10,96%	12,32%	43,61%

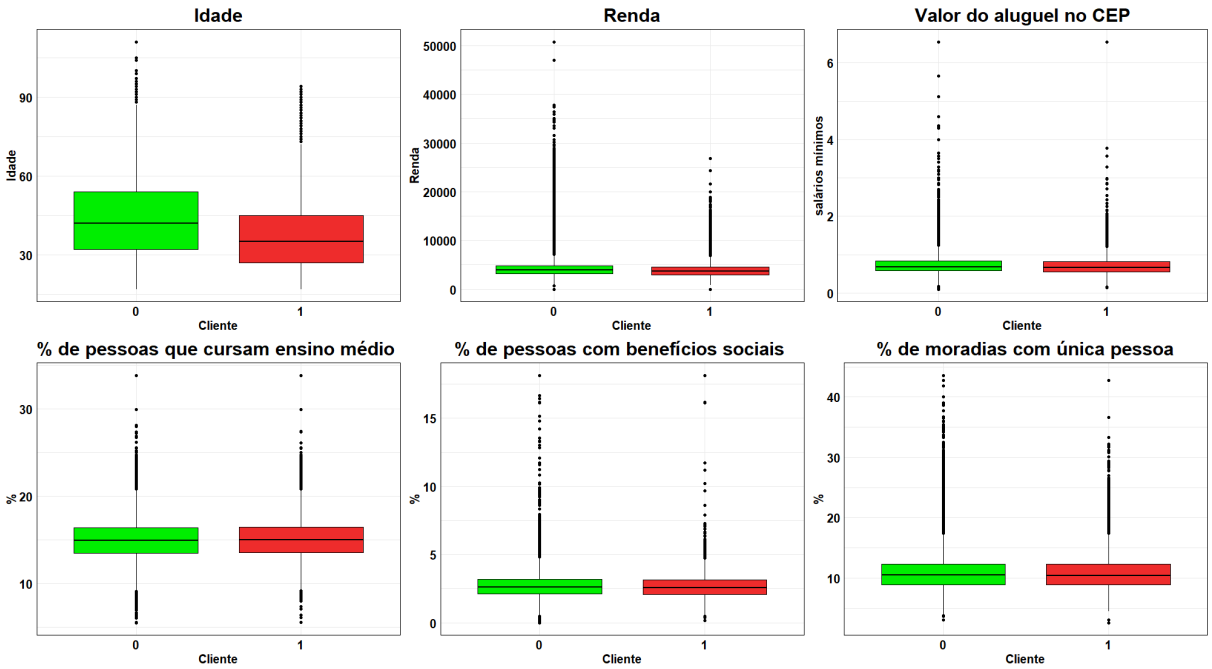


Figura 4.2: Boxplot das variáveis de cadastro.

Com a análise dos *Boxplots* presentes na Figura 4.2 temos um indicativo que somente a variável *Idade*, por si só, consegue discriminar de maneira satisfatória os clientes em suas

verdadeiras classes, mas que pode encontrar dificuldades na classificação de novos clientes, uma vez que se observa uma discrepância moderada no padrão de comportamento. Além disso, podemos observar que os clientes adimplentes (bons pagadores) geralmente possuem idade superior comparados a clientes inadimplentes. Já as demais variáveis, por si só, podem enfrentar dificuldades ao classificar corretamente um novo cliente em sua classe real, pois apresentam padrões de comportamento semelhantes em ambas as classes de Y.

Análise de variáveis da loja

No banco de dados temos 115 covariáveis de comportamento do cliente, sendo sumarizadas na Tabela 4.5 por meio do seu "tipo"e um exemplo de variável combinada é a *Soma de Vencimentos nos Últimos 6 Meses em Relação a Renda do Titular*.

Tabela 4.5: Covariáveis de comportamento do cliente.

Tipo	**Quantidade**
Atraso	22
Antecipados	25
Pontual	15
Quitados	25
Contratos	11
Tempo	2
Combinadas	15

Selecionamos algumas covariáveis de cada "tipo"de comportamento para fazer uma análise descritiva, apresentadas a seguir:

- **var_199:** Quantidade de vencimentos quitados no último mês,
- **var_204:** Soma dos valores de vencimentos quitados no último mês,
- **var_68:** Média da quantidade de parcelas dos contratos nos últimos 2 anos,
- **var_19:** Média de dias de vencimentos quitados atrasados nos últimos 3 meses,
- **var_22:** Média de dias de vencimentos quitados atrasados nos últimos 2 anos,
- **var_70:** Quantidade de vencimentos quitados antecipadamente nos últimos 3 meses,
- **var_55:** Quantidade de contratos nos últimos 6 meses,
- **var_110:** Valor médio de vencimentos pontuais nos últimos 3 meses,
- **var_239:** Soma de vencimentos quitados nos últimos 6 meses em relação a renda do titular,
- **var_228:** % da soma de vencimentos atrasados nos últimos 6 meses em relação a 24 meses,
- **var_3:** Tempo até a primeira compra (em dias),
- **var_4:** Tempo até o primeiro cartão (em dias).

A Figura 4.3 contém uma análise por meio de *boxplots* da variável resposta em relação as covariáveis selecionadas.

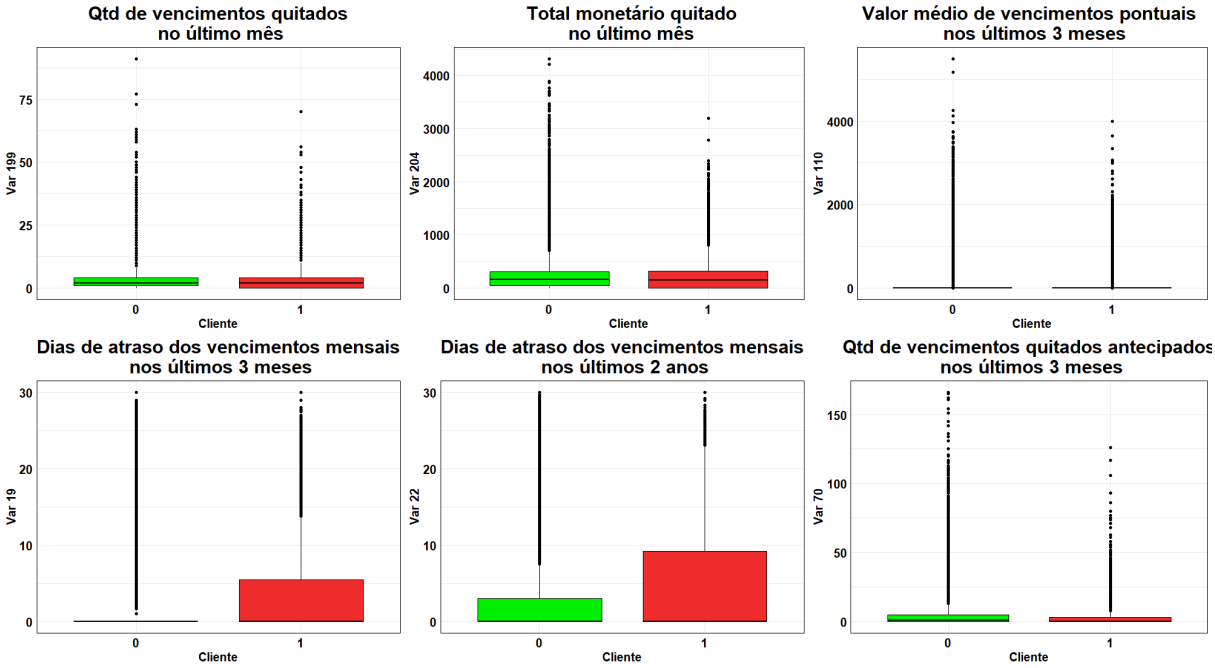


Figura 4.3: Boxplot da variável resposta em relação a variáveis comportamentais.

Com a análise da Figura 4.3 notamos que as variáveis *Dias de atraso nos vencimentos mensais nos últimos 3 meses* e *Dias de atraso nos vencimentos mensais nos últimos 2 anos* (var_19 e var_22,respectivamente) parecem distinguir muito bem os clientes em suas classes e vemos que a maior parte dos clientes adimplentes possuem poucos dias de atrasos. Além disso, as demais variáveis podem encontrar dificuldades na classificação de novos clientes, pois, por si só, não aparentam ter uma discrepância no padrão de comportamento das classes de Y. Agora, plotamos mais *Boxplots* de variáveis representados na Figura 4.4.

Ao analisar a Figura 4.4 notamos que a variável *Média de parcelas dos contratos nos últimos 2 anos* possui uma leve discrepância no comportamento das classes de Y, já a variável *Quantidade de contratos nos últimos 6 meses* não aparenta ter essa pouca discrepância. Já em relação às variáveis de tempo (var_3 e var_4), observe que ambas as variáveis exibem um comportamento similar em relação à variável resposta e notamos que, quando o tempo até a primeira compra ou até a obtenção do primeiro cartão é menor, há mais clientes inadimplentes, logo, essas variáveis conseguem discriminar as classes de Y, por si só, de maneira satisfatória, além disso, ambas as variáveis apresentam valores outliers. Já a variável combinada *% de vencimentos atrasados nos últimos 6 meses em*

relação ao últimos 2 anos consegue discriminar as classes de Y e percebemos que há mais inadimplentes quanto maior essa proporção.

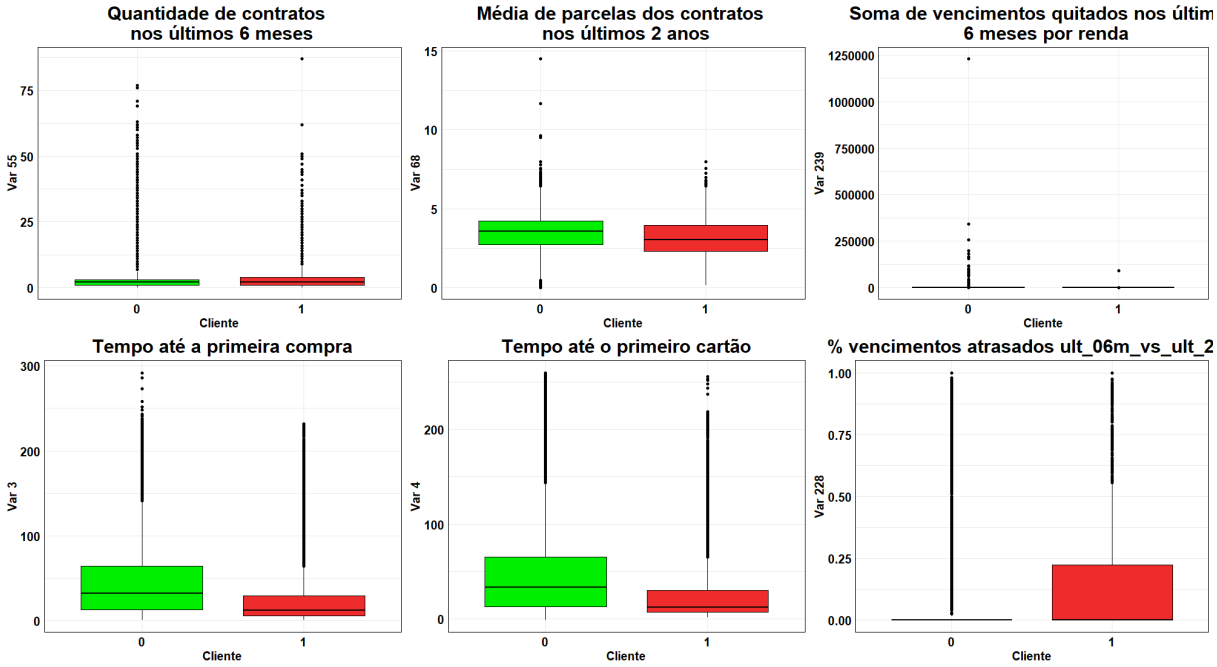


Figura 4.4: Boxplot da variável resposta em relação a variáveis comportamentais.

De modo geral, há indícios de que algumas das covariáveis selecionadas, por si só, conseguem discriminar os clientes em suas classes verdadeiras de forma satisfatória, como a variável var_228 que se refere a *% de vencimentos atrasados nos últimos 6 meses vs 2 anos* evidência que a maioria dos clientes inadimplentes estão concentrados nos últimos 6 meses, já que quanto maior essa proporção geralmente os clientes são inadimplentes. No entanto, é esperado que haja dificuldades ao classificar observações da classe minoritária, devido ao nível severo de desbalanceamento e à presença de variáveis que exibem padrões semelhantes em seus *boxplots*.

Após realizar a análise descritiva e exploratória dos dados para identificar seus comportamentos, a disposição das informações, bem como possíveis erros ou ausências, avançamos para a etapa de modelagem. O primeiro passo foi realizar *Data Splitting* e dividir a base em amostras. A base inicial possui 411.469 clientes e utilizando a semente 290 realizamos uma amostragem aleatória considerando aproximadamente 80% dos dados para o conjunto de treino e 20% para o conjunto de teste. Quando necessário uma amostra de validação para estimar hiperparâmetros dividiremos a amostra de treino.

Vale ressaltar que a Taxa de Inadimplência (proporção de clientes maus pagadores) no conjunto de treinamento é de $5,36\%$ e no conjunto de teste é de $5,43\%$, essa proporção

próxima nas duas amostras ajuda a garantir a consistência das generalizações dos modelos e a uniformidade na representação dos dados.

4.3 Regressão Logística com Penalização

Ajustamos o modelo de Regressão Logística com Penalização (lasso) aos dados de treinamento, com o objetivo de selecionar covariáveis para o modelo. Foi necessário usar **variáveis dummies** nas variáveis categóricas, é por meio das variáveis dummies que fazemos um dos fatores de cada variável categórica ser igual a zero (denominada de referência) e mede-se os demais a partir dele.

O modelo de Regressão Logística não possui a necessidade de estimar hiperparâmetros, então não foi necessária uma amostra de validação ou otimização de hiperparâmetros. Como já mencionado, queremos estimar a probabilidade do *já cliente* ser classificado como 0, ou seja, com base em seu histórico de transações e dados cadastrais queremos estimar a chance do cliente ser "bom" pagador.

Para ajustar o modelo usamos a biblioteca `glmnet` da linguagem R, essa função automaticamente já realiza padronização das covariáveis contínuas e usamos penalização lasso para o ajuste final. Para tal, utilizamos validação cruzada considerando 10 folds e o critério de "deviance" para estimar o λ, ele mede o desvio da log-verossimilhança. Vale ressaltar, que também testamos o ajuste usando o critério da área sob a curva ROC, mas como este selecionou mais covariáveis, preferimos o critério usando o deviance. O tempo gasto para a seleção de variáveis foi de $20,09$ minutos, já o tempo do treinamento do modelo após o lasso foi de 40 segundos.

Com o ajuste realizado encontramos diferentes valores de λ, a Tabela 4.6 e a Figura 4.5 sumarizam as informações sobre os diferentes valores. Note que a quantidade de variáveis selecionadas não zeradas também inclui as variáveis dummies. Optamos por usar o valor do lambda.1se, já que seleciona menos *features* e ao realizar uma manipulação de dados observamos que foram selecionadas 50 covariáveis dentre as 213 originais. Apresentaremos as variáveis descaracterizadas, mas a Tabela A.1 que está no apêndice apresenta todas variáveis do banco de dados conjuntamente com a sua descrição.

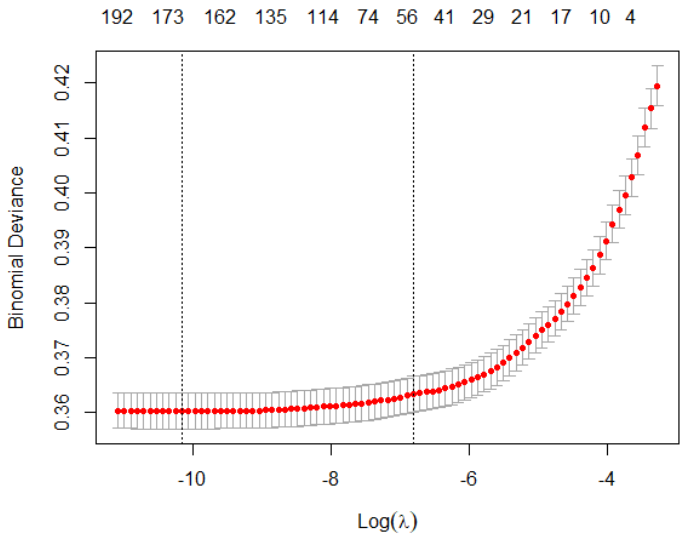


Figura 4.5: Quantidade de variáveis versus valor do λ na validação cruzada.

Tabela 4.6: Informações de λ usando validação cruzada.

λ	Valor	Variáveis selecionadas
Lambda.min	0.0000389	171
Lambda.1se	0.0011079	51

Features mais importantes

Como mencionado, escolhemos o valor do lambda.1se e dentre as 213 covariáveis iniciais do banco de dados o modelo de regressão logística com lasso selecionou 50 covariáveis. Para fins interpretativos construímos a Figura 4.6 que representa o nível de importância das covariáveis em relação aos coeficientes positivos e negativos. O intercepto é positivo de valor $2,32$.

Conforme o dicionário de covariáveis da Tabela A.1, observa-se:

- Algumas das *features* com coeficientes negativos estão relacionadas a vencimentos atrasados (var_13, var_15, var_14, var_17, var_19,var_20,var_21), à presença de auxílio emergencial (var_275) e ao gênero do cliente (var_1),

- Algumas das *features* com coeficientes positivos estão associadas a vencimentos quitados antecipadamente ou pontualmente (como var_233, var_228, var_199 e var_214), tempo até fazer a primeira compra (var_3), tempo até fazer o primeiro cartão (var_4), idade (var_2), renda do cliente (var_251).

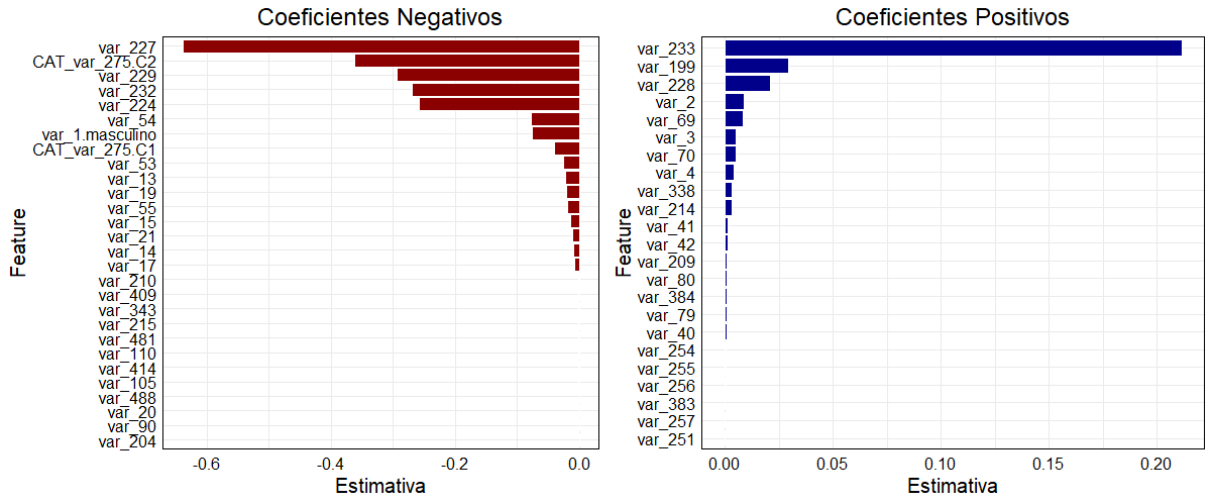


Figura 4.6: Representação dos coeficientes da Regressão Logística em ordem decrescente.

Medidas de desempenho

Para calcular as métricas de avaliação utilizamos a amostra de teste, e o tempo gasto para gerar as previsões neste método foi de $0,63$ segundos. Além disso, a área sob a Curva ROC foi de $78,46\%$ e o índice de Gini de $56,92\%$, esses resultados indicam que o modelo possui um bom desempenho na discriminação das classes da variável resposta. Na Tabela 4.7, apresentamos algumas medidas de sensibilidade, especificidade e ponto de corte conforme a curva ROC.

Tabela 4.7: Algumas medidas de sensibilidade, especificidade e corte.

Sensibilidade	Especificidade	$\mathbb{P}(Y = 0\|\mathbf{x})$
0,522	0,852	0,914
0,400	0,911	0,888
0,250	0,952	0,838
0,174	0,978	0,800

Uma interpretação da Tabela 4.7 é a seguinte: se a probabilidade de classificar um cliente como "bom"for maior que $0,800$ ($\mathbb{P}(Y = 0|\mathbf{x}) \geq 0,800$), a estimativa da probabilidade de classificar corretamente os clientes maus é de $0,174$ (Sensibilidade). Em contrapartida, a estimativa da probabilidade de classificar corretamente os clientes bons é de $0,978$ (Especificidade).

Devido ao desbalanceamento das classes da variável resposta vamos analisar alguns pontos de corte para encontrarmos as medidas de desempenho, os pontos de corte analisados foram:

- Otimização a estatística Youden: 0,943
- Otimização da F1: 0,870

- Média das probabilidades: 0,946

A Tabela 4.8 e 4.9 apresentam a matriz de confusão e as métricas de desempenho para os diferentes pontos de corte, respectivamente.

Tabela 4.8: Matriz de Confusão para diferentes pontos de corte.

(a) Corte 0,870

	Verdadeiro	
Predito	Y = 0	Y = 1
Y = 0	73107	2904
Y = 1	4969	1490

(b) Corte 0,943

	Verdadeiro	
Predito	Y = 0	Y = 1
Y = 0	56905	1328
Y = 1	21171	3066

(c) Corte 0,946

	Verdadeiro	
Predito	Y = 0	Y = 1
Y = 0	55312	1250
Y = 1	22764	3144

Tabela 4.9: Métricas de Desempenho para diferentes pontos de corte.

Métrica	K = 0,870	K = 0,943	K = 0,946
Sensibilidade	33,91%	69,78%	71,55%
Especificidade	93,63%	72,88%	70,84%
Precisão	23,07%	12,65%	12,13%
VPN	96,18%	97,72%	97,79%
F1	27,45%	21,41%	20,75%
Acurácia	90,45%	72,72%	70,88%

Podemos observar que as medidas de desempenho variam significativamente conforme o ponto de corte. Neste trabalho, nosso objetivo é encontrar um equilíbrio entre sensibilidade e especificidade para tentar classificar corretamente ambos clientes, sendo conversadores. Portanto, o ponto de corte selecionado que melhor equilibra essas duas métricas é a média das probabilidades estimadas $K = 0,946$.

No capítulo de comparação dos modelos, apresentaremos as medidas de desempenho para o modelo de Regressão Logística com Lasso utilizando o ponto de corte $0,946$.

4.4 Random Forest

Ajustamos o método *Random Forest* e *Bagging* na base de dados por meio da biblioteca `ranger()` da linguagem R. Conforme discutido na parte teórica, *Random Forests* possuem alguns hiperparâmetros, por isso utilizaremos como método de otimização a *Grid Search* através de várias combinações pré-definidas e usaremos uma amostra de validação

para analisar a performance de cada combinação, selecionando os melhores valores de hiperparâmetros. Para esse procedimento, dividimos uma parte da amostra de treinamento na amostra de validação.

A Tabela 4.10 apresenta os hiperparâmetros escolhidos para serem otimizados e seus respectivos espaços de possibilidades discretos.

- **mtry**: número de features que escolhidas aleatoriamente que serão analisadas para escolher cada divisão das árvores,
- **max.depth**: caminho máximo entre raiz e alguma folha,
- **num.trees**: número de árvores.

Tabela 4.10: Valores dos hiperparâmetros para Otimização da *Random Forest* usando *Grid Search*.

Hiperparâmetro	**Valores**
mtry	[5,10,15]
max.depth	[3,6,10]
num.trees	[250,500,600]

A Tabela 4.11 traz alguns resultados da otimização, como o valor da estatística KS, AUC e Gini para algumas combinações de hiperparâmetros na amostra de validação.

Tabela 4.11: Medidas de desempenho usando a amostra de validação para algumas combinações.

mtry	**num.trees**	**max.depth**	**KS**	**AUC**	**Gini**
15	250	10	0,4529	0,8001	0,6001
15	500	10	0,4530	0,8001	0,6001
15	600	10	0,4530	0,8001	0,6000
10	250	10	0,4498	0,7977	0,5955
5	250	10	0,4427	0,7920	0,5841
15	250	6	0,4338	0,7855	0,5711
20	250	3	0,3724	0,7559	0,5119

Observe na Tabela 4.11 que o número de árvore já se estabiliza em $B = 250$, já que as métricas de desempenho são bem próximas fixando os demais hiperparâmetros e variando o *num.trees*. Portanto, com o *tuning parameters* o melhor subconjunto de hiperparâmetros foi a combinação $mtry = 15$, $num.trees = 250$ e $max.depth = 10$. O tempo de treinamento foi de 2 minutos para este método.

Features mais importantes

Como visto na parte teórica, uma das desvantagens de *Random Forests* é seu baixo poder interpretativo, pois não conseguimos interpretar os coeficientes das covariáveis como na Regressão Logística. Contudo, há uma medida de importância calculada para cada *feature* do modelo baseada no índice de Gini. Durante a construção de cada árvore, a impureza do nó é reduzida quando uma variável significativa é usada para fazer uma divisão. A importância de uma variável é a soma dessa redução em todos os nós onde foi utilizada, e a média dessa redução é calculada considerando todas as árvores da floresta.

A Figura 4.7 apresenta as 25 covariáveis mais importantes do modelo se baseando neste índice.

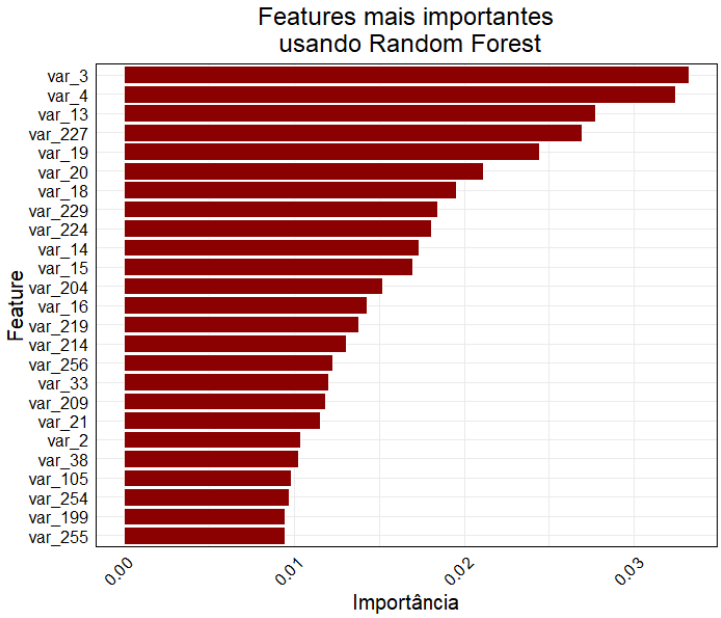


Figura 4.7: Gráfico que representa as 25 covariáveis mais importantes de Random Forest.

Utilizando o dicionário da Tabela A.1, observe que entre as covariáveis mais importantes para este método são:

- As variáveis de tempo até o cliente fazer o primeiro cartão (var_3) e o tempo até a primeira compra (var_4),
- Variáveis relacionadas ao atraso no pagamento (var_13, var_14,var_15, var_19, var_18, var_20),
- Idade (var_2),
- Soma de valores de vencimentos quitados no último mês (var_204),
- Quantidade de vencimentos quitados no último mês ou pontualmente (var_199 e var_33).

Notamos que algumas dessas covariáveis também estão presentes no modelo de Regressão Logística, principalmente as covariáveis mais importantes em *Random Forest*. Uma observação a se destacar é que a renda foi selecionada no modelo de regressão logística com penalização, mas com coeficiente positivo pequeno e em Random Forest com importância menor dos que as 25 citadas acima.

Medidas de desempenho

Para calcular as métricas de avaliação usamos a amostra de teste e o tempo gasto para gerar as previsões deste método foi de $5,36$ segundos. A medida da área sob a Curva ROC (AUC) foi de $80,56\%$ e o Índice de Gini foi de $61,12\%$, observamos que o modelo apresenta um bom desempenho na discriminação das classes da variável resposta.

Devido ao desbalanceamento das classes da variável resposta, os pontos de corte analisados foram:

- Otimização a estatística Youden: 0,942
- Otimização da F1: 0,855
- Média das probabilidades: 0,946

A Tabela 4.12 e 4.13 apresentam a matriz de confusão e as medidas de desempenho para os 3 diferentes cortes, respectivamente.

Tabela 4.12: Matriz de Confusão para os diferentes pontos de corte.

(a) K = 0,855

	Verdadeiro	
Predito	Y = 0	Y = 1
Y = 0	74297	2858
Y = 1	3779	1536

(b) K = 0,942

	Verdadeiro	
Predito	Y = 0	Y = 1
Y = 0	57949	1260
Y = 1	20127	3134

(c) K = 0,946

	Verdadeiro	
Predito	Y = 0	Y = 1
Y = 0	55970	1151
Y = 1	22106	3243

Podemos observar que as medidas de desempenho variam conforme o ponto de corte, como nosso objetivo é encontrar um equilíbrio entre sensibilidade e especificidade para tentar classificar corretamente ambos clientes, escolhemos o ponto de corte da média das probabilidades estimadas. No capítulo de comparação dos modelos apresentaremos as medidas de desempenho para o modelo de *Random Forest* usando o corte $K = 0,946$.

Tabela 4.13: Métricas de Desempenho para os diferentes pontos de corte.

Métrica	K = 0,855	K = 0,942	K = 0,946
Sensibilidade	34,96%	71,32%	73,80%
Especificidade	95,16%	74,22%	71,69%
Precisão	28,90%	13,47%	12,79%
VPN	96,30%	97,87%	97,98%
F1	31,64%	22,66%	21,80%
Acurácia	91,95%	74,07%	71,80%

4.5 XGBoost

Para o ajuste do *XGBoost* vamos usar a Otimização Bayesiana e abordagem **TPE (Tree-structured Parzen Estimator)** com a biblioteca `Optuna` para escolher a melhor combinação de hiperparâmetros e para a modelagem usamos a biblioteca `xgboost`, ambas da linguagem Python. Para tal, a biblioteca Optuna executa 30 ensaios, onde em cada um testa uma combinação diferente de hiperparâmetros se baseando na abordagem TPE e usando a área sob a Curva ROC como critério de maximização.

A Tabela 4.14 apresenta os hiperparâmetros escolhidos para serem otimizados e seus respectivos espaços de possibilidades.

Tabela 4.14: Intervalos dos hiperparâmetros para Otimização Bayesiana no XGBoost.

Hiperparâmetro	**Intervalo**
learning_rate	[0,0001 ; 0,4]
n_estimators	[100 ; 500]
max_depth	[3 ; 10]
subsample	[0,05 ; 1]
colsample_bytree	[0,05 ; 1]
gamma	[0 ; 5]
min_child_weight	[1 ; 30]

Na Tabela 4.15, apresentamos os valores dos hiperparâmetros considerados no modelo final após a Otimização Bayesiana. O tempo de ajuste foi de $2,30$ minutos.

Features mais importantes

A Figura 4.8 apresenta as 25 covariáveis mais importantes do modelo XGBoost. A importância de cada covariável é calculada com base em como cada uma contribui para a construção do modelo de árvores de decisão. Neste caso, o valor de importância de uma feature é determinado pelo número de vezes que ela é usada para dividir os nós nas árvores de decisão do modelo. Por exemplo, se uma feature foi usada com mais frequência,

Tabela 4.15: Valor dos hiperparâmetros selecionadas após a otimização do XGBoost.

Hiperparâmetro	**Valor**
objective	"binary:logistic"
tree_method	"hist"
importance_type	"gain"
learning_rate	0,0333
n_estimators	244
max_depth	9
subsample	0,8676
colsample_bytree	0,8
gamma	1,7240
min_child_weight	17
reg_lambda	1
alpha	0
random_state	29

o XGBoost atribui uma maior importância a ela, assumindo que ela contribui mais para as decisões do modelo.

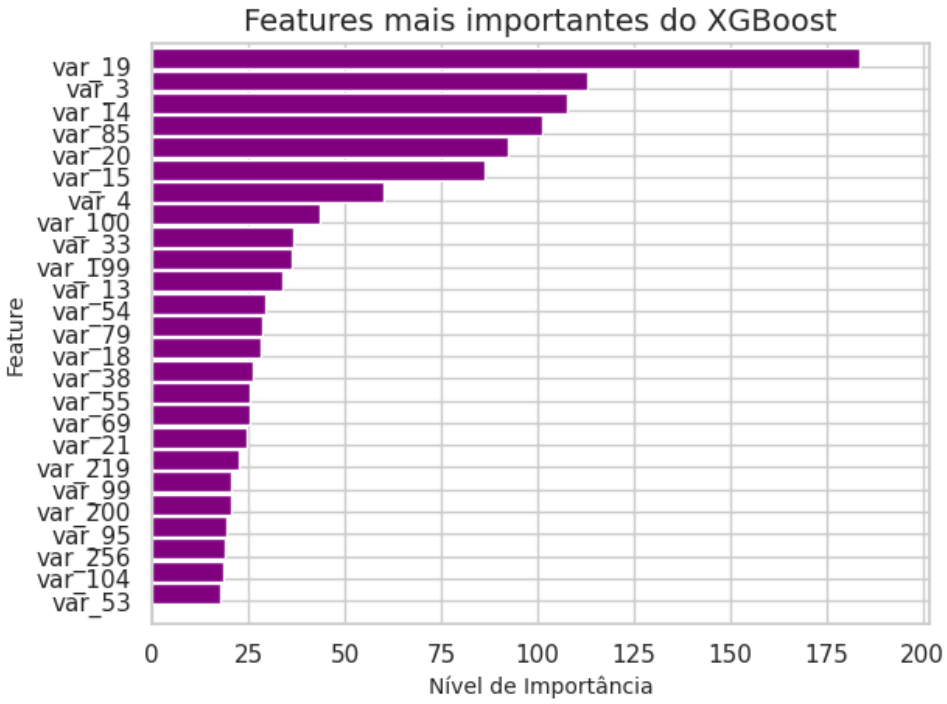


Figura 4.8: Gráfico que representa as 25 covariáveis mais importantes do XGBoost.

As variáveis mais importantes do *XGBoost* segundo a Tabela A.1, são:

- Variáveis do tipo "atraso no pagamento" (var_19, var_14, var_20,var_15, var_13),
- Variáveis do tipo "vencimentos quitados" (var_219),
- Variáveis de tempo até fazer a primeira compra ou primeiro cartão (var_3 e var_4),
- Quantidade de contratos ou parcelas (var_54, var_55,var_69),
- Vencimentos quitados pontualmente (var_104),

- Vencimentos quitados antecipados ou pontualmente (var_33 e var_199).

Medidas de desempenho

Para calcular as métricas de avaliação usamos a amostra de teste e o tempo gasto para gerar as previsões foi de $1,22$ segundos. A medida da área sob a Curva ROC (AUC) foi de $82,03\%$ e o Índice de Gini foi de $64,06\%$, observamos que o modelo apresenta um bom desempenho na discriminação das classes da variável resposta.

Devido ao desbalanceamento das classes da variável resposta, os pontos de corte analisados foram:

- Otimização a estatística Youden: 0,947
- Otimização da F1: 0,855
- Média das probabilidades: 0,946

A Tabela 4.16 e 4.17 apresentam a matriz de confusão e as medidas de desempenho para os 3 diferentes cortes, respectivamente.

Tabela 4.16: Matriz de Confusão para os diferentes pontos de corte.

(a) K = 0,855

	Verdadeiro	
Predito	Y = 0	Y = 1
Y = 0	73307	2572
Y = 1	4769	1822

(b) K = 0,946

	Verdadeiro	
Predito	Y = 0	Y = 1
Y = 0	58579	1196
Y = 1	19497	3198

(c) K = 0,947

	Verdadeiro	
Predito	Y = 0	Y = 1
Y = 0	58128	1166
Y = 1	19948	3228

Tabela 4.17: Métricas de Desempenho para os diferentes pontos de corte.

Métrica	K = 0,855	K = 0,946	K = 0,947
Sensibilidade	41,47%	72,78%	73,46%
Especificidade	93,89%	75,03%	74,45%
Precisão	27,64%	14,09%	13,93%
VPN	96,61%	97,99%	98,03%
F1	33,17%	26,61%	23,41%
Acurácia	91,10%	74,91%	74,40%

Podemos observar que as medidas de desempenho variam conforme o ponto de corte, como nosso objetivo é encontrar um equilíbrio entre sensibilidade e especificidade para tentar classificar corretamente ambos clientes, escolhemos o ponto de corte da Estatística

de Youden. No capítulo de comparação dos modelos apresentaremos as medidas de desempenho para o modelo de *XGBoost* usando o corte $K = 0,947$.

4.6 LightGBM

Para otimizar os hiperparâmetros do LightGBM também iremos usar Otimização Bayesiana e a abordagem **TPE (Tree-structured Parzen Estimator)** com a biblioteca `Optuna` e `lightgbm`, ambas da linguagem Python.

A Tabela 4.18 apresenta os hiperparâmetros a serem otimizados usando Otimização Bayesiana e seus respectivos espaços de possibilidades.

Tabela 4.18: Intervalos dos hiperparâmetros para Otimização Bayesiana do LightGBM.

Hiperparâmetro	**Intervalo**
learning_rate	[0,0001; 0,4]
num_leaves	[50 ; 150]
colsample_bytree	[0,05 ; 1]
min_child_samples	[5 ; 50]
max_depth	[5 ; 15]
n_estimators	[50 ; 200]
subsample	[0,05; 1]
min_split_gain	[0 ; 0,1]

Na Tabela 4.19, após a Otimização Bayesiana apresentamos os principais parâmetros considerados no modelo final. O tempo de treinamento do modelo foi de 54 segundos.

Tabela 4.19: Valor dos hiperparâmetros selecionadas após a otimização do LightGBM.

Hiperparâmetro	**Valor**
objective	"binary:logistic"
metric	"goss"
importance_type	"gain"
learning_rate	0,0391
num_leaves	69
colsample_bytree	0,6826
min_child_samples	45
max_depth	10
n_estimators	150
subsample	0,8242
min_split_gain	0,0175
random_state	29

Features mais importantes

A Figura 4.9 apresenta as 25 covariáveis mais importantes do modelo LightGBM.

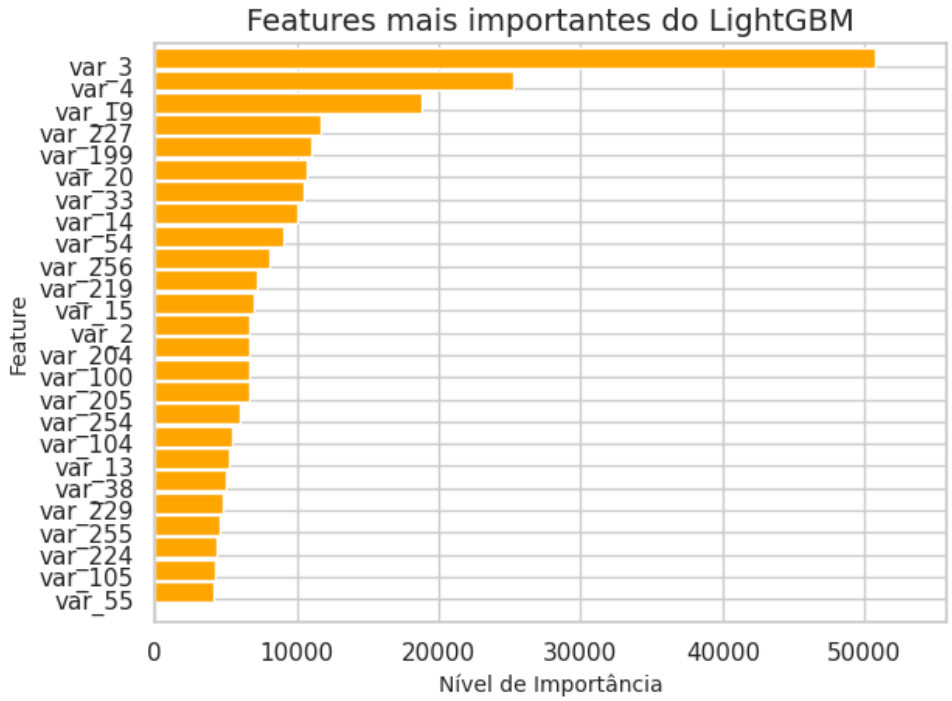


Figura 4.9: Gráfico que representa as 25 covariáveis mais importantes do LightGBM.

As variáveis mais importantes do *LightGBM* segundo a Tabela A.1, são:

- Variáveis do tipo "atraso no pagamento" (var_13, var_14, var_15,var_19, var_20),
- Variáveis do tipo "vencimentos quitados" (var_219),
- Variáveis de tempo até fazer a primeira compra ou primeiro cartão (var_3 e var_4),
- Quantidade de contratos ou parcelas (var_54, var_55),
- Vencimentos quitados pontualmente (var_104),
- Vencimentos quitados antecipados ou pontualmente (var_33 e var_199),
- Idade do cliente (var_2).

Medidas de desempenho

Para calcular as métricas de avaliação usamos a amostra de teste e o tempo gasto para gerar as previsões foi de 2 segundos. A medida da área sob a Curva ROC (AUC) foi de $81,97\%$ e o Índice de Gini foi de $63,93\%$, observamos que o modelo apresenta um bom desempenho na discriminação das classes da variável resposta.

Devido ao desbalanceamento das classes da variável resposta, os pontos de corte analisados foram:

- Otimização a estatística Youden: 0,948
- Otimização da F1: 0,840
- Média das probabilidades: 0,946

A Tabela 4.20 e 4.21 apresentam a matriz de confusão e as medidas de desempenho para os 3 diferentes cortes, respectivamente.

Tabela 4.20: Matriz de Confusão para os diferentes pontos de corte.

(a) K = 0,840

	Verdadeiro	
Predito	Y = 0	Y = 1
Y = 0	73965	2707
Y = 1	4111	1687

(b) K = 0,946

	Verdadeiro	
Predito	Y = 0	Y = 1
Y = 0	57754	1173
Y = 1	20322	3221

(c) K = 0,948

	Verdadeiro	
Predito	Y = 0	Y = 1
Y = 0	56931	1122
Y = 1	21145	3272

Tabela 4.21: Métricas de Desempenho para os diferentes pontos de corte.

Métrica	K = 0,840	K = 0,946	K = 0,948
Sensibilidade	38,39%	73,30%	74,46%
Especificidade	94,73%	73,97%	72,91%
Precisão	29,09%	13,68%	13,40%
VPN	96,46%	98,00%	98,07%
F1	33,10%	23,06%	22,71%
Acurácia	91,73%	73,94%	73,00%

Podemos observar que as medidas de desempenho variam conforme o ponto de corte, como nosso objetivo é encontrar um equilíbrio entre sensibilidade e especificidade para tentar classificar corretamente ambos clientes, escolheremos o ponto de corte da média das probabilidades estimadas. No capítulo de comparação dos modelos apresentaremos as medidas de desempenho para o modelo de *LightGBM* usando o corte $K = 0,946$.

Capítulo 5

Comparação dos modelos

Nesta seção, realizaremos uma comparação entre os quatro modelos quanto à sua capacidade de discriminar as categorias da variável resposta, em ser bom ou mau pagador. A Figura 5.1 apresenta a Curva ROC correspondente a cada modelo, enquanto a Tabela 5.1 contém as métricas de desempenho obtidas. As duas primeiras linhas da tabela indicam, respectivamente, o tempo gasto no treinamento dos modelos e o tempo necessário para realizar previsões utilizando a amostra de teste.

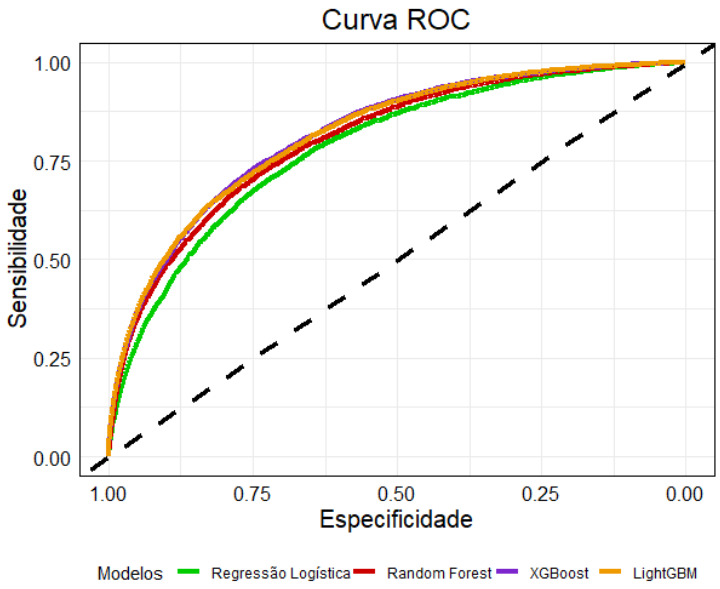


Figura 5.1: Curva ROC dos quatro dos modelos ajustados neste trabalho.

Tabela 5.1: Métricas de Desempenho usando o conjunto de teste para os quatro modelos ajustados.

Métrica	Regressão Logística	Random Forest	XGBoost	LightGBM
Ajuste	58 seg	2 min	2,30 min	54 seg
Previsão	0,63 seg	5,36 seg	1,22 seg	2 seg
Recall	71,55%	73,80%	73,46%	73,30%
Especificidade	70,84%	71,69%	74,45%	73,97%
Precisão	12,13%	12,79%	13,93%	13,68%
VPN	97,79%	97,98%	98,03%	98,00%
F1-Score	20,75%	21,80%	23,41%	23,06%
Acurácia	70,88%	71,80%	74,40%	73,94%
AUC	78,46%	80,56%	82,03%	81,97%
Gini	56,92%	61,12%	64,06%	63,93%
KS	42,73%	45,65%	47,97%	47,50%
Risco Preditivo	29,12%	28,20%	25,60%	26,06%

Ressaltamos que os procedimentos, as análises e métricas foram realizados utilizando os softwares R e Python. Em primeiro lugar, observa-se que em geral todos os quatro modelos apresentam uma boa capacidade de discriminação entre as classes da variável resposta. Isso é evidenciado pelas métricas como AUC, *Gini* e KS que apresentam valores elevados em todos os modelos evidenciando a boa capacidade preditiva, por exemplo, como o KS acima de 40% e AUC próximo de 80% para a maioria dos modelos. Além disso, os modelos demonstram valores de desempenho bastante próximos entre si, reforçando sua eficácia na classificação. Mas ao compará-los, podemos observar que o modelo *XGBoost* se destaca com as maiores métricas de AUC ($\approx$ 82%), Gini ($\approx$ 64%), KS ($\approx$ 48%), Especificidade ($\approx$ 74%), Precisão ($\approx$ 14%), VPN ($\approx$ 98%), Acurácia ($\approx$ 74%) e F1-Score ($\approx$ 24%) em relação aos demais modelos. Em contrapartida, o modelo de *Regressão Logística com Penalização (Lasso)* se destaca ao possuir os menores valores de medidas de performance entre os demais modelos, com destaque na AUC ($\approx$ 78%), Gini ($\approx$ 57%), KS ($\approx$ 43%), Especificidade ($\approx$ 71%), Precisão ($\approx$ 12%), VPN ($\approx$ 97%), Acurácia ($\approx$ 71%) e F1-Score ($\approx$ 21%).

É importante destacar que os pontos de corte utilizados para o cálculo das métricas de *Recall*, *Especificidade*, *Precisão*, *Valor Preditivo Negativo (VPN)* e *Acurácia* foram definidos com base na estatística da média das probabilidades ou de Youden estimadas no conjunto de teste, cujos valores foram 0.946 ou 0.947, todos bem próximos.

O *LightGBM* foi o modelo que apresentou o menor tempo de treinamento e modelo *XGBoost* apresentou o maior tempo de treinamento. Ademais, todos os quatro mode-

los apresentaram tempos pequenos de previsão no conjunto de teste, com destaque da *Regressão Logística com Lasso*, com o menor valor (0, 63 segundos).

Agora, a Tabela 5.2 contém o intervalo de confiança para o risco estimado dos quatros modelos.

Tabela 5.2: Intervalo de confiança para o risco estimado dos modelos.

Método	**IC($\hat{R}$(x);95%)**
Regressão Logística	[28,81%, 29,43%]
Random Forest	[27,89%, 28,51%]
XGBoost	[25,30%, 25,90%]
LightGBM	[25,76%, 26,36%]

Ao analisar a Tabela 5.2 podemos observar que no geral ambos os intervalos de confiança possuem tamanhos próximos, sendo que o intervalo que mais distante entre os modelos é o da regressão logística com lasso, com limite inferior e superior maiores em relação aos intervalos dos demais modelos. Vale destacar a proximidade dos intervalos dos métodos *boosting*.

É importante comparar os modelos também por suas vantagens ou desvantagens além das medidas de performance, podemos destacar que a Regressão Logística com Penalização selecionou 50 covariáveis do banco de dados, reduzindo a variância do estimador, além disso, sua função de predição possui um alto poder interpretativo, como podemos ver com algumas variáveis selecionadas com coeficientes negativos se referem a atrasos nos pagamentos das faturas, fazendo sentido com a análise descritiva. Enquanto os modelos baseados em árvores não possuem esse poder interpretativo.

Dessa maneira, também é interessante destacar que algumas covariáveis aparecem em todos os modelos, tanto por serem selecionadas pelo método *Lasso* quanto por estarem entre as variáveis mais importantes nos modelos de *Random Forest* e nos métodos de *Boosting*. Entre essas covariáveis, destacam-se o *tempo até o primeiro cartão*, o *tempo até a primeira compra*, a *idade* e *variáveis de atraso no pagamento*.

5.0.1 Análise descritiva das predições

A Tabela 5.3 traz algumas informações descritivas das predições do score (probabilidade de ser classificado como "bom" pagador vezes 1000) nos modelos e a Figura 5.2 contém seus *boxplots*, ambos com a amostra teste.

Conforme observado na Tabela 5.3, o modelo com a menor probabilidade de um cliente

Tabela 5.3: Análise descritivas dos scores dos modelos na amostra de teste.

Modelo	Mínimo	1º Quartil	2º Quartil	Média	3º Quartil	Máximo
Regressão Logística	8,70	935,27	964,83	946,04	980,50	1000
Random Forest	428,9	936	966	945,6	979,8	993,5
XGBoost	125,5	940,7	973,3	946,8	987,8	998,9
LightGBM	95,7	938,8	972,1	945,60	986,4	997,3

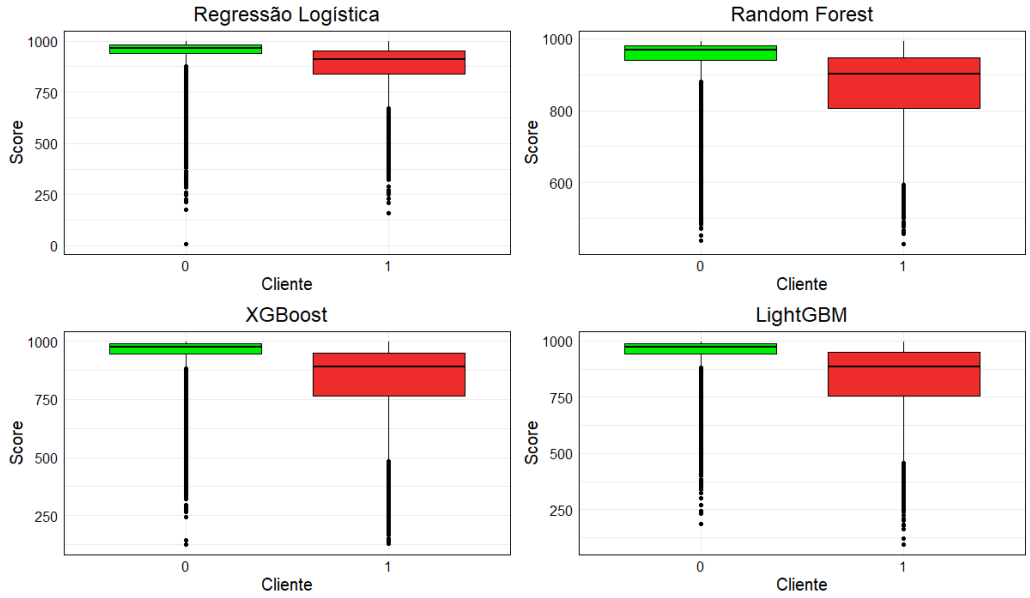


Figura 5.2: *Boxplots* do Score por cliente nos modelos.

ser adimplente é a Regressão Logística com Penalização com score de $8,70$. No entanto, todos os modelos, a partir do segundo quartil, apresentam pontuações já próximas a 900. Esse resultado pode ser explicado pela natureza da base de dados utilizada neste estudo, que abrange clientes da loja que possuem um cartão e um tempo de relacionamento estabelecido, ou seja, em sua maioria, são indivíduos que realizam compras com certa frequência, tendendo a indicar um perfil de bons pagadores. Além disso, conforme mostrado na Figura 5.2, é possível observar que a maior parte dos clientes com menores scores são classificados como *maus pagadores*, o que indica que nossas predições estão alinhadas com a realidade. É importante destacar também que, como já mencionado, as distribuições das probabilidades estimadas pelos quatro modelos ajustados são semelhantes. No entanto, os *boxplots* revelam que o modelo de *Regressão Logística com Lasso* apresenta uma distribuição ligeiramente distinta em comparação aos demais.

De maneira análoga, na Figura 5.3 foram construídas as densidades dessas predições em relação as classes verdadeiras de maus e bons pagadores, levando à mesma conclusão obtida com os *boxplots*, a densidade vermelha se refere aos maus (1) pagadores e a verde

aos bons pagadores (0).

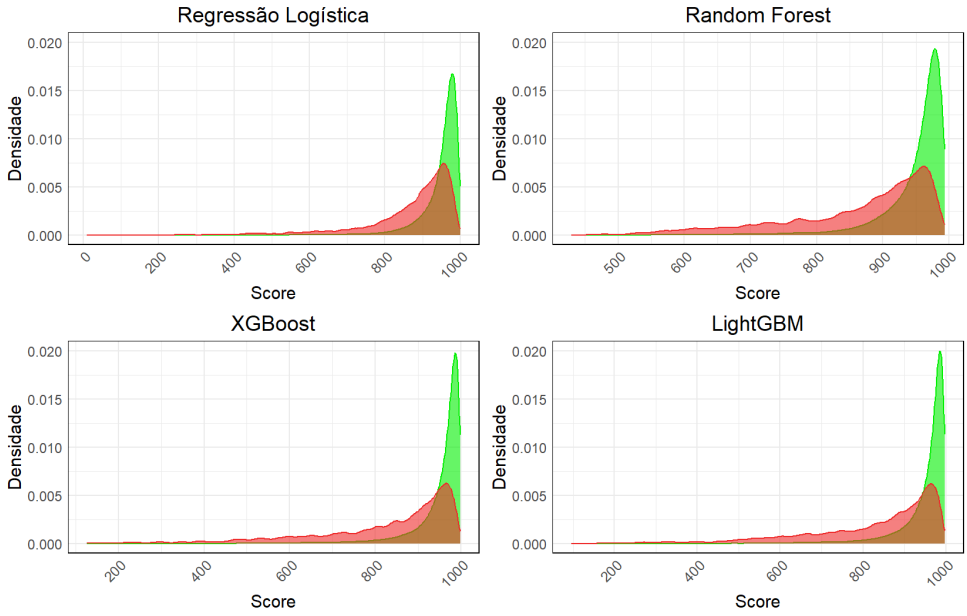


Figura 5.3: Densidade do Score por cliente nos modelos.

Com a análise da Figura 5.3 verificamos novamente que a Regressão Logística parece se diferenciar quanto a distribuição dos dados em comparação aos demais modelos. Para finalizar a análise descritiva das predições no conjunto de teste, construímos os gráficos da Figura 5.4.

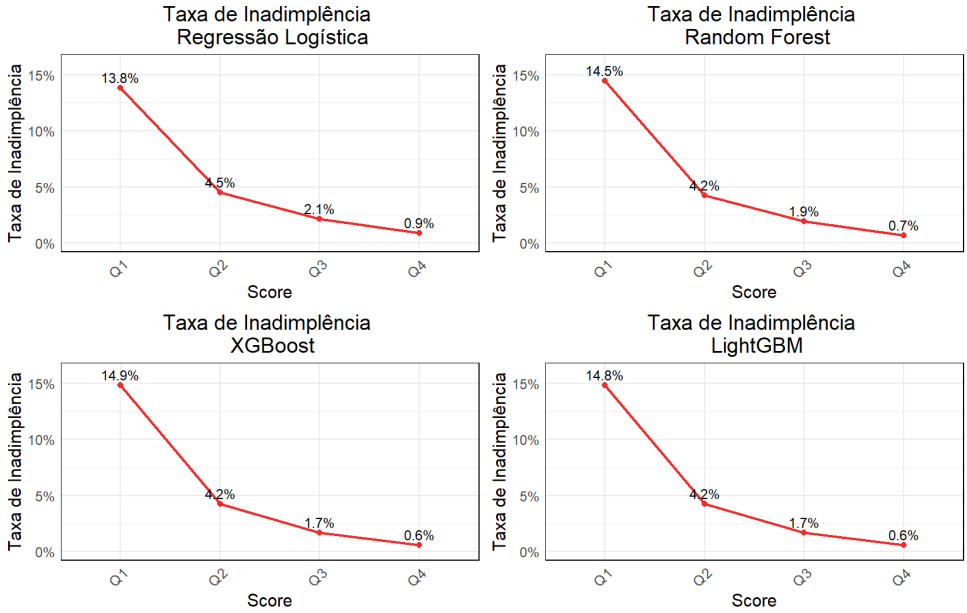


Figura 5.4: Gráficos de taxa de inadimplência por cliente nos modelos em relação aos quartis de cada modelo.

Ao analisar a Figura 5.4, observamos os gráficos para cada um dos modelos em relação à taxa de inadimplência (proporção de maus pagadores) na amostra de teste em relação

aos quartis com valores representados na Tabela 5.3.

A distribuição do score mostrou-se semelhante entre os modelos, ambos evidenciando uma concentração de pontuações acima de 800. Além disso, a taxa de inadimplência apresentou padrões de decaimento próximos entre os métodos, com uma sutil exceção na Regressão Logística com Lasso, que exibiu proporções de maus pagadores distintas em comparação aos demais modelos. Por outro lado, os modelos *XGBoost* e *LightGBM* revelaram comportamentos mais lineares, com curvas bastante próximas entre si.

Capítulo 6

Considerações Finais

Neste estudo, analisamos quatro modelos de classificação de aprendizado de máquina aplicados à avaliação de crédito, com foco específico na previsão da probabilidade de um cliente ser um bom ou mau pagador. Os modelos examinados incluem desde abordagens clássicas, como a Regressão Logística com Penalização (ver Seção 3.1), até técnicas mais avançadas baseadas em árvores, como Random Forest (ver Seção 3.2), e métodos de *boosting*, como XGBoost (ver Seção 3.4) e LightGBM (ver Seção 3.5), discutindo também a teoria por trás de cada um. Além disso, exploramos diferentes métricas de desempenho para problemas de classificação, como a Área sob a Curva ROC e o Coeficiente de Gini (ver Seção 3.6). Observamos que, em cenários de análise de crédito, é comum lidar com dados desbalanceados, onde uma classe da variável resposta predomina sobre a outra e neste trabalho apenas $5,37\%$ dos clientes são maus pagadores, sendo uma evidência de um problema de desbalanceamento nas classes da variável resposta. Nesses casos, para avaliar o desempenho dos modelos, foi necessário adotar um ponto de corte nas predições diferente do padrão de $0,5$, neste trabalho avaliamos algumas estatísticas e no final optamos por utilizar a médias das predições ou a estatística de Youden para definir um ponto de corte.

Utilizando um conjunto de dados de crédito de uma carteira de clientes de uma loja de varejo que possuem o cartão da própria loja, composto por 441.469 clientes e abrangendo o período de dezembro de 2021 a setembro de 2022, aplicamos os quatro métodos estudados e realizamos os ajustes necessários para cada um. Antes da modelagem, fizemos uma análise descritiva e concluimos que há indícios de que algumas das covariáveis selecionadas para este estudo conseguem discriminar, por si só, os clientes em suas classes verdadeiras de forma satisfatória, como a variável var_19 que se refere a *Dias de atraso dos vencimentos mensais os últimos 3 meses* e demonstra que a maioria dos clientes inadimplentes

possuem muitos dias em atraso. No entanto, é esperado que haja dificuldades ao classificar observações da classe minoritária, devido ao nível severo de desbalanceamento e à presença de variáveis que exibem padrões semelhantes em seus *boxplots*. Também analisamos as variáveis categóricas, como *EDS*, sendo que a taxa de inadimplência é maior quando o cliente reside em um endereço desfavorecido socialmente. Dessa maneira, ajustamos os modelos com a amostra de treino.

Como conclusão, comparamos os métodos (ver Seção 5) usando a amostra de teste, avaliamos os quatro modelos ajustados por meio das métricas apresentadas e concluímos que todos discriminam bem as classes da variável resposta, apresentando métricas de desempenho semelhantes. Entre os métodos, destacam-se os de *Boosting*, especialmente o *XGBoost*, que alcançou medidas de desempenho maiores em relação aos demais modelos, como área sob a curva ROC ($\approx 82\%$) e KS ($\approx 48\%$). Por outro lado, a *Regressão Logística com Penalização* apresentou as menores métricas de desempenho, com destaque para a área sob a curva ROC ($\approx 78\%$) e KS ($\approx 43\%$). Ademais, o *LightGBM* apresentou o menor tempo de treinamento e o *XGBoost* apresentou o maior tempo.

Além disso, é relevante observar que a Regressão Logística com Penalização selecionou 50 covariáveis, contribuindo para a redução da variância do estimador e possui como principal vantagem seu poder interpretativo, evidenciado pelas covariáveis de atraso com coeficientes negativos neste modelo. Já os demais modelos não possuem esse poder interpreta

Pela análise descritiva dos valores preditos (em Score) utilizando a amostra de teste, é possível observar que, em todos os modelos, há um indicativo de que os clientes considerados maus pagadores geralmente apresentam scores mais baixos, como ilustrado na Figura 5.2 e Figura 5.3. Além disso, pela Figura 5.4 observamos que, à medida que o score aumenta, a taxa de maus pagadores diminui.

Referências Bibliográficas

Anshul (2024). Gradient boosting algorithm: A complete guide for beginners. https://www.analyticsvidhya.com/blog/2021/09/gradient-boosting-algorithm-a-complete-guide-for-beginners/#:~:text=Gradient%20boosting%20Algorithm%20in%20machine%20learning%20is%20a%20method%20standing#:~:text=Gradient%20boosting%20Algorithm%20in%20machine%20learning%20is%20a%20method%20standing. Acessado em: 7 out. 2024.

Assunção, F. (2012). *Estratégias para tratamento de variáveis com dados faltantes durante o desenvolvimento de modelos preditivos*. Dissertação de mestrado, Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo. Acesso em: 2024-06-16.

Barakat, H. M., Khaled, O. M. e Nigm, E.-S. M. (2019). *Statistical techniques for modelling extreme value data and related applications*. Cambridge Scholars Publishing.

Bergstra, J., Bardenet, R., Bengio, Y. e Kégl, B. (2011). Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, **24**.

Breiman, L. (2001). Random forests. *Machine learning*, **45**, 5–32.

Brochu, E., Cora, V. M. e De Freitas, N. (2010). A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*.

Casella, G. e Berger, R. (2024). *Statistical inference*. CRC Press.

Chen, T. e Guestrin, C. (2016). Xgboost: A scalable tree boosting system. Em *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, páginas 785–794.

Corrêa, M. F. e Vellasco, M. (2008). Análise de risco de crédito em correspondentes bancários através de redes neurais. *Revista Inteligência Computacional Aplicada*, **1**(1), 23–37.

Diniz, C. e Louzada, F. (2013). Métodos estatısticos para análise de dados de crédito. Em *6th Brazilian Conference on Statistical Modeling in Insurance and Finance, Maresias-SP*, volume 9, páginas 11–13.

Forti, M. (2018). *Técnicas de machine learning aplicadas na recuperação de crédito do mercado brasileiro*. Tese de doutorado.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, páginas 1189–1232.

Guo, R., Zhao, Z., Wang, T., Liu, G., Zhao, J. e Gao, D. (2020). Degradation state recognition of piston pump based on iceemdan and xgboost. *Applied Sciences*, **10**(18), 6593.

Hastie, T., Tibshirani, R., Friedman, J. H. e Friedman, J. H. (2009). *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer.

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

Jadhav, S., He, H. e Jenkins, K. (2018). Information gain directed genetic algorithm wrapper feature selection for credit rating. *Applied Soft Computing*, **69**, 541–553.

James, G., Witten, D., Hastie, T., Tibshirani, R. *et al.* (2013). *An introduction to statistical learning*, volume 112. Springer.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q. e Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, **30**.

Luchetta, F. (2019). Uma análise sobre o crédito: metodologia de credit scoring como ferramenta de mitigação de risco de crédito para instituições financeiras.

Mayank Anand, Arun Velu, P. W. (2022). Prediction of loan behaviour with machine learning models for secure banking. *Journal of Computer Science an Engineering (JCSE)*, **3**(1), 01–13.

meanxai (2024a). Lightgbm. Disponível em: https://www.youtube.com/watch?v=N39NE4Nj6vc&t=1143s.

meanxai (2024b). Xgboost. Disponível em: https://www.youtube.com/watch?v=mA3uJyMB4XU&t=4s.

Oliveira, M. S., Lopes, N. S. e Figueiroa, M. L. (2012). Classificação do rating de risco através de um behavior score gerado pela regressão logística binária.

Optunagoogl (2021). Optuna. Disponível em: https://optuna.readthedocs.io/en/stable/index.html.

Pinheiro, J. M. H. (2023). Um estudo sobre algoritmos de boosting e a otimização de hiperparâmetros utilizando optuna. *Monografia apresentada ao Curso de Engenharia Mecatrônica da Escola de Engenharia de São Carlos, Universidade de São Paulo.*.

Qin, C., Zhang, Y., Bao, F., Zhang, C., Liu, P. e Liu, P. (2021). Xgboost optimized by adaptive particle swarm optimization for credit scoring. *Mathematical Problems in Engineering*, **2021**, 1–18.

Ribeiro, R. d. S. (2011). Concessão de crédito para pessoa física: uma análise do mercado atual.

Rocca, J. (2019). Ensemble methods: bagging, boosting and stacking. medium-towards data science.

Saini, A. (2021). Gradient boosting algorithm: A complete guide for beginners. Disponível em: https://www.analyticsvidhya.com/blog/2021/09/gradient-boosting-algorithm-a-complete-guide-for-beginners/. Acesso em: 17-08-2023.

Santos, J. A. V. (2021). Inteligência artificial aplicada à avaliação de crédito bancário.

Snoek, J., Larochelle, H. e Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, **25**.

Watanabe, S. (2023). Tree-structured parzen estimator: Understanding its algorithm components and their roles for better empirical performance. *arXiv preprint arXiv:2304.11127*.

Apêndice A

Dicionário do banco de dados

Apresentamos o dicionário das variáveis que compõe o banco de dados:

Tabela A.1: Variáveis do banco de dados com descrição.

Variável	**Descrição**
Data	Mês observado
var_resposta	Variável dicotômica que indica mau (1) ou bom (0) pagador
var_1	Gênero do cliente
var_2	Idade em anos
var_3	Tempo do cartão até a primeira compra
var_4	Tempo que o cliente tem o cartão
var_13	Máximo de dias de vencimentos atrasados no último mês
var_14	Máximo de dias de vencimentos atrasados nos últimos 3 meses
var_15	Máximo de dias de vencimentos atrasados nos últimos 6 meses
var_16	Máximo de dias de vencimentos atrasados nos últimos 12 meses
var_17	Máximo de dias de vencimentos atrasados nos últimos 24 meses
var_18	Média de dias de vencimentos atrasados no último mes
var_19	Média de dias de vencimentos atrasados nos últimos 3 meses
var_20	Média de dias de vencimentos atrasados nos últimos 6 meses
var_21	Média de dias de vencimentos atrasados nos últimos 12 meses
var_22	Média de dias de vencimentos atrasados nos últimos 24 meses
var_33	Mínimo de dias de vencimentos quitados antecipados no último mês
var_34	Mínimo de dias de vencimentos quitados antecipados nos últimos 3 meses
var_35	Mínimo de dias de vencimentos quitados antecipados nos últimos 6 meses
var_36	Mínimo de dias de vencimentos quitados antecipados nos últimos 12 meses
var_37	Mínimo de dias de vencimentos quitados antecipados nos últimos 24 meses
var_38	Média de dias de vencimentos quitados antecipados no último mês
var_39	Média de dias de vencimentos quitados antecipados nos últimos 3 meses
var_40	Média de dias de vencimentos quitados antecipados nos últimos 6 meses
var_41	Média de dias de vencimentos quitados antecipados nos últimos 12 meses
var_42	Média de dias de vencimentos quitados antecipados nos últimos 24 meses
var_53	Quantidade de contratos no último mês
var_54	Quantidade de contratos nos últimos 3 meses

var_55	Quantidade de contratos nos últimos 6 meses
var_56	Quantidade de contratos nos últimos 12 meses
var_57	Quantidade de contratos nos últimos 24 meses
var_63	Valor máximo de parcelas dos contratos nos últimos 12 meses
var_64	Valor máximo de parcelas dos contratos nos últimos 24 meses
var_65	Valor mínimo de parcelas dos contratos nos últimos 12 meses
var_66	Valor mínimo de parcelas dos contratos nos últimos 24 meses
var_67	Valor médio de parcelas dos contratos nos últimos 12 meses
var_68	Valor médio de parcelas dos contratos nos últimos 24 meses
var_69	Quantidade de vencimentos antecipados no último mês
var_70	Quantidade de vencimentos antecipados nos últimos 3 meses
var_71	Quantidade de vencimentos antecipados nos últimos 6 meses
var_72	Quantidade de vencimentos antecipados nos últimos 12 meses
var_73	Quantidade de vencimentos antecipados nos últimos 24 meses
var_74	Soma de vencimentos antecipados no último mês
var_75	Soma de vencimentos antecipados nos últimos 3 meses
var_76	Soma de vencimentos antecipados nos últimos 6 meses
var_77	Soma de vencimentos antecipados nos últimos 12 meses
var_78	Soma de vencimentos antecipados nos últimos 24 meses
var_79	Valor médio dos vencimentos antecipados no último mês
var_80	Valor médio dos vencimentos antecipados nos últimos 3 meses
var_81	Valor médio dos vencimentos antecipados nos últimos 6 meses
var_82	Valor médio dos vencimentos antecipados nos últimos 12 meses
var_83	Valor médio dos vencimentos antecipados nos últimos 24 meses
var_85	Quantidade de vencimentos atrasados nos últimos 3 meses
var_86	Quantidade de vencimentos atrasados nos últimos 6 meses
var_87	Quantidade de vencimentos atrasados nos últimos 12 meses
var_88	Quantidade de vencimentos atrasados nos últimos 24 meses
var_90	Soma de vencimentos antecipados nos últimos 3 meses
var_91	Soma de vencimentos antecipados nos últimos 6 meses
var_92	Soma de vencimentos antecipados nos últimos 12 meses
var_93	Soma de vencimentos antecipados nos últimos 24 meses
var_95	Média de vencimentos atrasados nos últimos 3 meses
var_96	Média de vencimentos atrasados nos últimos 6 meses
var_97	Média de vencimentos atrasados nos últimos 12 meses
var_98	Média de vencimentos atrasados nos últimos 24 meses
var_99	Qnt de vencimentos quitados pontualmente no último mês
var_100	Qnt de vencimentos quitados pontualmente nos últimos 3 meses
var_101	Qnt de vencimentos quitados pontualmente nos últimos 6 meses
var_102	Qnt de vencimentos quitados pontualmente nos últimos 12 meses
var_103	Qnt de vencimentos quitados pontualmente nos últimos 24 meses
var_104	Soma de vencimentos quitados pontualmente no último mês
var_105	Soma de vencimentos quitados pontualmente nos últimos 3 meses
var_106	Soma de vencimentos quitados pontualmente nos últimos 6 meses
var_107	Soma de vencimentos quitados pontualmente nos últimos 12 meses
var_108	Soma de vencimentos quitados pontualmente nos últimos 24 meses
var_109	Média de vencimentos quitados pontualmente no último mês
var_110	Soma de vencimentos quitados pontualmente nos últimos 3 meses
var_111	Soma de vencimentos quitados pontualmente nos últimos 6 meses
var_112	Soma de vencimentos quitados pontualmente nos últimos 12 meses
var_113	Soma de vencimentos quitados pontualmente nos últimos 24 meses

var_199	Qtd de vencimentos quitados no último mês
var_200	Qtd de vencimentos quitados nos últimos 3 meses
var_201	Qtd de vencimentos quitados nos últimos 6 meses
var_202	Qtd de vencimentos quitados nos últimos 12 meses
var_203	Qtd de vencimentos quitados nos últimos 24 meses
var_204	Soma de vencimentos quitados no último mês
var_205	Soma de vencimentos quitados nos últimos 3 meses
var_206	Soma de vencimentos quitados nos últimos 6 meses
var_207	Soma de vencimentos quitados nos últimos 12 meses
var_208	Soma de vencimentos quitados nos últimos 24 meses
var_209	Máximo vencimentos quitados no último mês
var_210	Soma de vencimentos quitados nos últimos 3 meses
var_211	Soma de vencimentos quitados nos últimos 6 meses
var_212	Soma de vencimentos quitados nos últimos 12 meses
var_213	Soma de vencimentos quitados nos últimos 24 meses
var_214	Média de vencimentos quitados no último mês
var_215	Média de vencimentos quitados nos últimos 3 meses
var_216	Média de vencimentos quitados nos últimos 6 meses
var_217	Média de vencimentos quitados nos últimos 12 meses
var_218	Média de vencimentos quitados nos últimos 24 meses
var_219	Mínimo vencimentos quitados no último mês
var_220	Mínimo de vencimentos quitados nos últimos 3 meses
var_221	Mínimo de vencimentos quitados nos últimos 6meses
var_222	Mínimo de vencimentos quitados nos últimos 12 meses
var_223	Mínimo de vencimentos quitados nos últimos 24 meses
var_224	% de qtd de contratos nos últimos 6 meses vs últimos 24 meses
var_227	% de qtd de vencimentos nos últimos 6 meses vs últimos 24 meses
var_228	% de qtd de vencimentos atrasados nos últimos 6 meses vs últimos 24 meses
var_229	% de qtd de vencimentos pontual nos últimos 6 meses vs últimos 24 meses
var_230	% de qtd de vencimentos antecipados nos últimos 6 meses vs últimos 24 meses
var_231	% de vencimentos atrasados vs todos os vencimentos nos últimos 2 anos
var_232	% de vencimentos pontuais vs todos os vencimentos nos últimos 2 anos
var_233	% de vencimentos antecipados vs todos os vencimentos nos últimos 2 anos
var_236	% da média de vencimentos nos últimos 6 meses vs últimos 24 meses
var_239	% da soma vencimentos nos últimos 6 meses sob a renda do titular
var_240	% da soma vencimentos nos últimos 12 meses sob a renda do titular
var_241	% da soma vencimentos nos últimos 24 meses sob a renda do titular
var_242	% da média de vencimentos os últimos 6 meses sob a renda do titular
var_243	% da média de vencimentos os últimos 12 meses sob a renda do titular
var_244	% da média de vencimentos os últimos 24 meses sob a renda do titular
var_249	Score de risco de crédito
var_250	Limite Sugerido para o cliente
var_251	Renda Pressumida do cliente
var_252	Score de propensão de pagamento
var_253	Índice de propensão a fragilidade
var_254	Índice de risco de crédito pessoa física-varejo eventual
var_255	Índice de risco de crédito pessoa física-varejo
var_256	Índice de risco de crédito pessoa física-varejo recorrente
var_257	Índice de risco de fraude pessoa física-varejo
var_264	Quantidade total de empresas vinculadas ao cliente
var_272	Quantidade de empresas vinculadas ao cliente que são ativas

var_275	Indicadora de recebimento de auxílio emergencial (C0= NA, C1 = Sim, C2 = Não)
var_306	Grau de urbanização do endereço do cliente
var_309	Tipologia do município rural-urbano do cliente
var_310	Menor distância do endereço do cliente e o hospital mais próximo
var_311	Quantidade de hospitais em um raio de 3km do endereço do cliente
var_312	Quantidade de hospitais em um raio de 5km do endereço do cliente
var_313	Menor distância entre o endereço do cliente e o município mais próximo
var_320	Quantidade de postos de saúde em um raio de 1km do endereço do cliente
var_321	Quantidade de postos de saúde em um raio de 5km do endereço do cliente
var_324	Score da classe do endereço do cliente, quanto maior melhor é o endereço
var_325	Indica se há pelo menos uma UPA perto do endereço do cliente: TRUE ou FALSE
var_327	Quantidade de EFS que o município mais próximo ao endereço do cliente possui
var_328	Score do município mais próximo ao endereço do cliente
var_329	Quantidade de farmácias em um raio de 1km do endereço do cliente
var_330	Quantidade de farmácias em um raio de 5km do endereço do cliente
var_338	Menor distância entre o endereço e o EDS mais próximo
var_339	Indica se o endereço é desfavorecido socialmente (EDS): TRUE ou FALSE
var_340	Quantidade de EDS em um raio de 1km do endereço do cliente
var_341	Quantidade de EDS em um raio de 2km do endereço do cliente
var_342	Quantidade de EDS em um raio de 3km do endereço do cliente
var_343	Quantidade de EDS em um raio de 5km do endereço do cliente
var_346	Distância entre o endereço e o EDS de alta severidade mais próximo
var_351	Quantidade de EDS severos em um raio de 1km do endereço do cliente
var_352	Quantidade de EDS severos em um raio de 3km do endereço do cliente
var_353	Quantidade de EDS severos em um raio de 5km do endereço do cliente
var_354	Quantidade de EDS com altíssima severidade em um raio de 3km do endereço do cliente
var_355	Distância entre o endereço e o EFS mais próximo
var_356	Quantidade de EFS em um raio de 1km do endereço do cliente
var_357	Quantidade de EFS em um raio de 2km do endereço do cliente
var_358	Quantidade de EFS em um raio de 5km do endereço do cliente
var_359	Distância entre o endereço do cliente e o EAP mais próximo
var_360	Quantidade de moradias de alto padrão em um raio de 5km do endereço do cliente
var_365	Classe do endereço do cliente: A,B,C,D,E
var_369	% de moradias com máquina de lavar no CEP do cliente
var_370	Número de cômodos médio no CEP do cliente
var_371	% de moradia alugada no CEP do cliente
var_377	% de domicílios particulares com acesso à rede de esgoto no CEP do cliente
var_378	% de domicílios particulares com acesso à rede elétrica no CEP do cliente
var_382	% de moradias com a presença de automóvel no CEP do cliente
var_383	% de moradias sem a presença de máquina de lavar roupa no CEP do cliente
var_384	% de moradias sem a presença de rádio no endereço no CEP do cliente
var_385	% de moradias com a presença de rádio no endereço no CEP do cliente
var_386	% de moradias de alvenaria e sem revestimento no CEP do cliente
var_387	% de apartamentos no endereço do cliente no CEP do cliente
var_388	% condomínios no endereço do cliente no CEP do cliente
var_389	% de moradias que apresentaram apenas um responsável pela casa no CEP do cliente
var_390	% de moradias que têm mais de uma pessoa residindo, com ou sem, parentesco no CEP
var_391	% de moradias que têm mais de uma pessoa residindo, com grau de parentesco no CEP
var_392	% de moradias que têm mais de uma pessoa residindo e é um núcleo familiar do CEP
var_393	% de moradias que têm uma única pessoa no CEP
var_394	Valor do aluguel no CEP em salários mínimos (R$1100,00 em 2021 e R$1212,00 em 2022)

var_395	% de pessoas residentes no CEP que frequentam a creche
var_396	% de pessoas residentes no CEP que são desempregados e em busca de trabalho
var_397	% de pessoas residentes no CEP que são economicamente ativos
var_398	% de pessoas residentes no CEP que frequentaram o EJA
var_399	% de pessoas residentes no CEP que apresentam estado civil casado
var_400	% de pessoas residentes no CEP que apresentam estado civil separado
var_401	% de pessoas residentes no CEP que apresentam estado civil solteiro
var_402	% de pessoas residentes no CEP que apresentam estado civil viúvo
var_403	% de pessoas residentes no CEP que cursam o ensino médio
var_404	% de pessoas residentes no CEP que nasceram e sempre moraram no município
var_405	% de pessoas residentes no CEP que não possuem união
var_406	% de pessoas que possuem união de casamento civil e religioso residentes no CEP
var_407	% de pessoas que possuem união de natureza consensual residentes no CEP
var_408	% de pessoas que declararam ser empregadores residentes no CEP
var_409	% de pessoas que declararam seu trabalho como sendo militar residentes no CEP
var_410	% de pessoas que declararam ter rendimentos de programas sociais residentes no CEP
var_411	% de pessoas que declararam ser aposentados residentes no CEP
var_412	% de pessoas que declararam ter rendimentos de aluguel ou investimentos
var_413	% de pessoas que têm o tempo gasto de deslocamento para o trabalho < 5 min
var_414	% de pessoas que têm o tempo gasto de deslocamento para o trabalho de 31 a 60 min
var_415	% de pessoas que têm o tempo gasto de deslocamento para o trabalho $>$ 120min
var_416	% de pessoas que declararam que trabalharam e estão afastadas no CEP
var_417	% de pessoas que declararam ter trabalho não remunerado no CEP
var_418	% de domicílios contemplados com arborização em seu entorno no CEP
var_421	% de domicílios que apresentam exposição a esgoto a céu aberto no CEP
var_423	% de domicílios que apresentam iluminação pública em seu entorno no CEP
var_424	% de domicílios que apresentam pavimentação em seu entorno no CEP
var_481	Qtd de centros de atendimento a mulher no município do CEP
var_482	É a distância em linha reta (em km) até a unidade de INSS mais próxima
var_488	Distância da fronteira mais próxima ao CEP
var_490	Total de indenizações tipo 9 para carros premium, nos últimos 6 meses
var_491	Prob de risco de ocorrência de sinistro de furto com motos, nos últimos 24 meses
var_492	Prob de risco de sinistros, nos últimos 24 meses
var_493	% de indenização de sinistros de furto para carros premium, nos ult 18 meses
var_494	% de indenizações de sinistros de furto para outros tipos de veículos, nos ult 24 meses

Apêndice B

Códigos

```
library(dplyr)
library(glmnet)
library(ggpubr)
library(ggplot2)
require(caret)
library(pROC)

# Regress o Logistica
base = base %>% filter(var_2 > 15)
base = base[,-c(1,2,3)]
dmy = dummyVars(" ~ .", data = base[], fullRank=T)
base = data.frame(predict(dmy, newdata = base))
set.seed(290)
split <- sample(c("Treinamento","Teste"),prob=c(0.8,0.2),
size= nrow(base),replace=TRUE)
base$split = split

X_treino  = base[base$split =="Treinamento",-c(1,219)]
Y_treino = base[base$split=="Treinamento",c(1)]
Y_treino = 1-Y_treino
Y_treino = as.factor(Y_treino)
X_treino.matrix = X_treino %>% as.matrix()

set.seed(50)
vc_lasso = glmnet::cv.glmnet(X_treino.matrix,
                             Y_treino,
                             alpha=1,
```

```
                              family="binomial",
                              type.measure = "deviance")

# Random Forest (depois otimizar os hiperparametros)
floresta <- ranger(
  formula= var_resposta ~ .,
  data= treino,
  mtry= 15,
  max.depth= 10,
  num.trees = 250,
  importance = "impurity",
  replace=T,
  splitrule = "gini",
  probability = TRUE,
  verbose   = TRUE,
  seed    = 100,
  classification=T
)

#XGBoost (Python depois de otimizar os hiperparametros)
from xgboost import XGBClassifier
class_xgb = XGBClassifier(learning_rate= 0.03332747285295623,
                          tree_method = 'hist',
                          importance_type = 'gain',
                          n_estimators= 244,
                          max_depth= 9,
                          subsample= 0.8675817069630809,
                          colsample_bytree= 0.8,
                          gamma= 1.7240110839139073,
                          min_child_weight= 17,
                          alpha = 0,
                          reg_lambda = 1,
                          random_state=29,
                          enable_categorical=True)
class_xgb.fit(X_treino,Y_treino)

#LightGBM (Python depois de otimizar os hiperparametros)
import lightgbm as lgb
class_lgbm = lgb.LGBMClassifier(learning_rate= 0.03906262325063714,
                                feature_importances = "gain",
```

```
                                    boosting_type='goss',
                                    num_leaves= 69,
                                    colsample_bytree= 0.6825559136050567,
                                    min_child_samples= 45,
                                    subsample= 0.8242272537263762,
                                    max_depth= 10,
                                    n_estimators= 150,
                                    min_split_gain= 0.017499438251586272,
                                    lambda_l1 = 0,
                                    random_state=29)
class_lgbm.fit(X_treino,Y_treino)
```