UNIVERSIDADE DE SÃO PAULO

ESCOLA POLITÉCNICA

YURI MISSIMA DIAS

MÉTODOS DE MACHINE LEARNING APLICADOS A RISCO E RECUPERAÇÃO DE CRÉDITO

SÃO PAULO

2022

YURI MISSIMA DIAS (10432522)

MÉTODOS DE MACHINE LEARNING APLICADOS A RISCO E RECUPERAÇÃO DE CRÉDITO

Relatório final da disciplina PTC 3531 - Laboratório de Projeto de Automação e Controle II, da Escola Politécnica da Universidade de São Paulo

Orientador: Prof. Dr. Felipe Miguel Pait

SÃO PAULO

2022

SUMÁRIO

1 INTRODUÇÃO

O ciclo de crédito é composto pelas fases de relacionamento entre uma instituição financeira que concede empréstimos e financiamentos, e seus clientes. São elas: prospecção de novos clientes, análise e concessão de crédito, gerenciamento da carteira e por último, cobrança de clientes em atraso.

Figura 1 – O ciclo de crédito.

Fonte: Forti, 2018.

Em um primeiro momento, após a prospecção de novos clientes adequados a um determinado produto, são aplicados modelos de *credit score*, que avaliam o risco de inadimplência no momento da concessão do crédito. Modelos chamados de *behavior score* estimam o risco para os já clientes e ajudam a fazer a manutenção da carteira de crédito.

Para os clientes que por acaso se encontrem em atraso, são aplicados os chamados modelos de *collection score***,** que os classificam segundo sua propensão de pagamentos, ou seja, atribuem uma probabilidade à recuperação dos valores dos contratos em atraso.

As instituições financeiras, como os bancos, têm como foco os modelos de *credit score*, dado que espera-se que bons modelos de concessão de crédito levem a uma baixa inadimplência. Por outro lado, dados recentes mostram que 65 milhões de brasileiros têm algum tipo de dívida em atraso (Serasa, 2022), o maior nível registrado em 12 anos, o que explica a relevância dos modelos pós-concessão de crédito.

Além disso, o volume e a complexidade crescentes de informações disponíveis nos dias atuais exige a aplicação de técnicas mais sofisticadas como *machine learning* para a tomada de decisões mais assertivas. Conhecer melhor o comportamento dos consumidores é um diferencial competitivo para as empresas, porque permite uma melhor alocação de recursos, tem o potencial de redução de custos entre 20%-25%, e ajuda a melhorar a aquisição e retenção de novos consumidores (Lee e Shin, 2020).

Estimativas mais precisas de risco de crédito também são necessárias para o cumprimento de regulações internacionais como os Acordos de Basileia, que dispõem sobre regras de capital econômico, capital regulatório e provisionamento. Os Acordos de Basileia III, versão mais recente das regulações, surgiram como resposta à crise financeira de 2007-2008 para reforçar a capacidade do setor financeiro em absorver choques, sendo essenciais para o tratamento e redução do risco sistêmico (Schwerter, 2011).

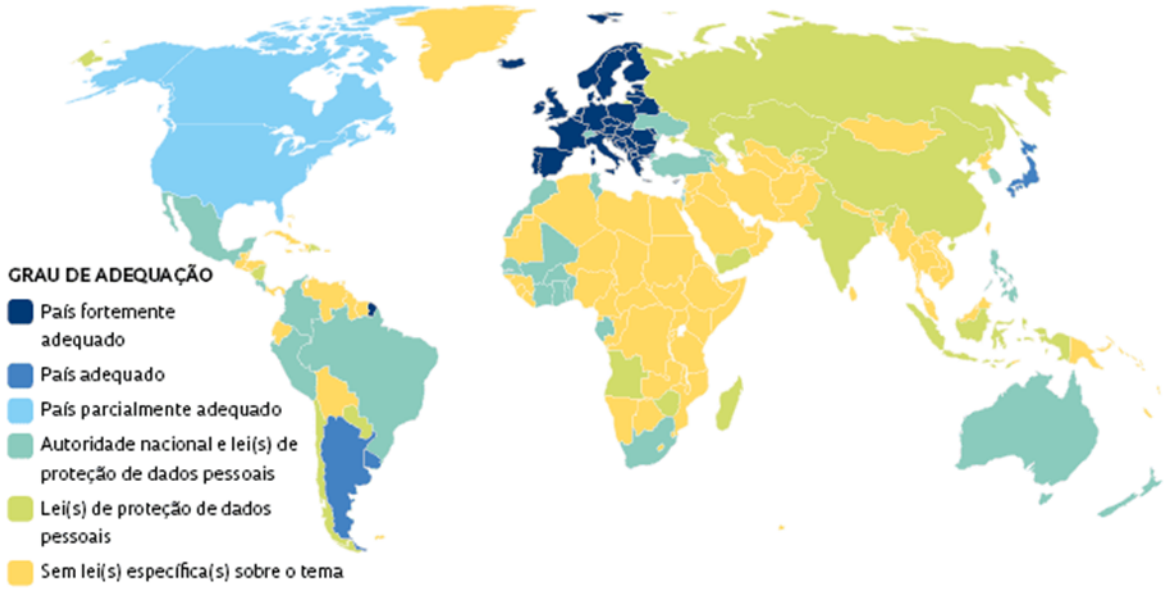
Mais especificamente, o objetivo deste trabalho é fazer um estudo comparativo das capacidades preditiva e interpretativa de técnicas de *machine learning* aplicadas ao risco e à recuperação de crédito, utilizando dados reais para construir modelos de PD (*probability of default*) e LGD (*loss given default).*

2 BREVE DISCUSSÃO SOBRE LGPD E TRATAMENTO DE DADOS NO BRASIL

Para treinamento de modelos de *machine learning* de crédito são necessárias informações sobre o comportamento passado de relacionamento entre consumidores e empresas que são por sua natureza dados pessoais. Por este motivo, no Brasil, seu tratamento está sob o escopo da LGPD (Lei Geral de Proteção de Dados Pessoais) e é feito pelos birôs de crédito (Serasa, SPC, Quod e Boa Vista).

Em um contexto de aumento exponencial do volume e complexidade dos dados disponíveis, e evolução tecnológica de *hardware* e *software,* que permite que empresas e governos extraiam informações sobre o comportamento de indivíduos, a lei surgiu “com o objetivo de proteger e garantir os direitos fundamentais de liberdade e de privacidade e o livre desenvolvimento da personalidade da pessoa natural" (Lei nº 13.709). É fundamentada em regulamentos internacionais como o RGPD (Regulamento Geral sobre a Proteção de Dados), da União Europeia, que possui os países mais adequados neste sentido.

Figura 2 – Proteção de dados pessoais no mundo.



Fonte: Comissão Nacional de Informática e Liberdade (CNIL/França).

Para garantir a proteção dos dados, a lei se apoia em diversos princípios. Em geral, esses princípios exigem que os dados sejam tratados com uma finalidade específica e legítima e que o seu uso seja limitado de acordo com a necessidade. Além disso, o titular deve ter o direito de consulta sobre a forma e a duração do tratamento dos seus dados. Estabelece também medidas de segurança para evitar acessos não autorizados, medidas de prevenção para evitar danos em virtude de tratamento não adequado e proíbe o uso dos dados para fins discriminatórios, sob pena de multa e até mesmo suspensão das atividades de quem faz o tratamento dos dados, caso haja descumprimento.

A LGPD estabeleceu dez bases legais, ou hipóteses, para tratamento dos dados (Lei nº 13.709, Art. 7º). Os birôs de crédito, empresas autorizadas no Brasil a tratar o tipo de dado que é objeto de estudo deste trabalho, são parceiras e recebem dados, por exemplo, de bancos ou lojas que vendem a prazo (chamados controladores dos dados) e podem utilizar os dados pelas hipóteses da lei de: proteção ao crédito; execução de contratos de clientes e parceiros; e cumprimento de obrigação regulatória/legal (que não precisam do consentimento do titular).

A lei prevê também a possibilidade de tratamento “para a realização de estudos por órgão de pesquisa, garantida, sempre que possível, a anonimização dos dados pessoais”, hipótese que também não precisa do consentimento do titular. Instituições de ensino superior e pesquisa como a USP se enquadram na hipótese. Mesmo assim, o compartilhamento dos dados depende do consentimento do controlador dos dados. Foram solicitadas amostras de dados anonimizados aos quatro birôs. Um deles respondeu que não compartilha dados em “conformidade com as finalidades previstas na legislação”, mesmo anonimizados, outro respondeu que ainda estavam revisando processos internos devido à LGPD, e por hora o compartilhamento de bases para estudos estava pausado sem previsão de retomada. Um deles ofereceu serviços apenas para empresas, e o último não retornou o contato.

É importante ressaltar que o órgão regulatório responsável pela fiscalização da lei, a ANPD (Autoridade Nacional de Proteção de Dados), era de criação recente no momento da escrita deste trabalho e os procedimentos e formas de atuação da

mesma ainda estavam sendo definidos, e as empresas e instituições de ensino e pesquisa estavam adequando seus procedimentos internos à nova lei. Assim, não existia nitidez sobre a interpretação da lei no contexto de pesquisa e isso impossibilitou o acesso aos dados e o desenvolvimento do trabalho no contexto que foi inicialmente idealizado do mercado de crédito brasileiro. Os dados que foram utilizados no trabalho são dados do mercado de crédito norte-americano e foram obtidos mais facilmente.

Ainda há muitas questões a serem discutidas. Existem estudos que mostram que é possível reidentificar indivíduos usando atributos de *datasets,* como mostrou Roscer (2019) e escândalos de vazamento de dados são frequentes. Isso gera questionamentos sobre privacidade e dificulta a pesquisa científica. A própria ANPD divulgou um estudo técnico em abril de 2022 com o intuito de promover um debate público sobre essas questões (ANPD, 2022).

3 REVISÃO TEÓRICA

A PD (*probability of default)* é uma estimativa da probabilidade de um tomador de crédito descumprir as obrigações legais do seu contrato de empréstimo (o que é chamado de *default*), como por exemplo não fazer um pagamento da sua hipoteca. Dado que houve um *default,* a LGD (*loss given default*) é uma estimativa do percentual de perda do valor total em exposição no momento do *default*. Ambas são parâmetros essenciais para o cálculo de perdas esperadas, capital econômico e capital regulatório sob as regras dos Acordos de Basileia.

O objetivo do trabalho é ajustar modelos de PD e LGD a uma base real de contratos de hipotecas americanas (*mortgages*) utilizando as técnicas:

- Regressão Logística e Regressão Linear
- K-Nearest Neighbors
- Árvores de Decisão
- Random Forest
- Adaboost
- Gradient Boosting
- XGBoost

A principal diferença entre os modelos de PD e LGD é que os primeiros são modelos de classificação binária (variável categórica) e os últimos são modelos de regressão (variável contínua).

O *dataset* é dividido em subconjuntos de treino e de teste para averiguação da eficácia dos modelos.

Abaixo se encontra uma revisão dos principais conceitos de cada uma das técnicas. Um detalhamento maior da teoria por trás das técnicas e algoritmos pode ser encontrado em Hastie et al. (2017) e/ou publicações originais que os descrevem.

3.1 Regressão

3.1.1 Regressão logística

A regressão logística é a técnica utilizada como referência para comparar resultados em relação às outras técnicas de machine learning de classificação, por ser a técnica mais tradicional e de fácil interpretação de parâmetros. Também é a mais aceita por órgãos regulatórios. O objetivo é predizer resultados em que a variável dependente é categórica e para modelos de PD, existem apenas duas categorias: *default* e não *default*. Além do mais, é assumido que os contratos seguem uma distribuição binomial, ou seja, cada resultado é um experimento ou variável independente de Bernoulli, com mesma probabilidade de *default* π a priori.

Dado uma amostra de *d* contratos com resultados conhecidos (*default* ou *não default*), a PD é estimada através da maximização da função de verossimilhança (minimização da função de custo) a seguir, onde $d_i = 1$ indica ocorrência de *default* no contrato i :

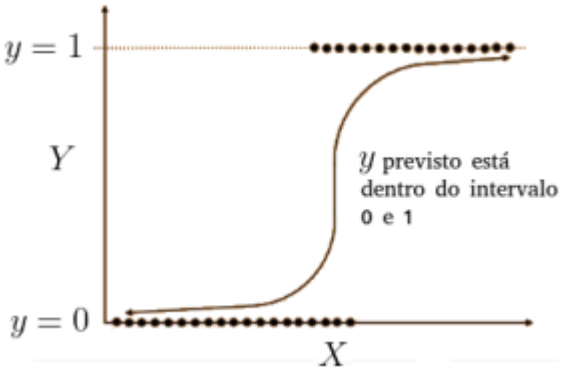
$$L(\pi \mid d) = \prod_{i}^{n} f(d_i \mid \pi) = \prod_{i}^{n} \pi^{d_i} (1 - \pi)^{1-d_i}$$

Como o objetivo é estimar a probabilidade π como uma combinação linear das features *x* e essa combinação linear produz valores fora do intervalo de probabilidade [0,1], é aplicada a Transformação Logit:

$$\pi = f(\beta, x) = \frac{exp(\beta x)}{1+exp(\beta x)}$$

Logo, as probabilidades estimadas ficam dentro do intervalo [0,1].

Figura 3 – Regressão logística.



Fonte: Oliveira, 2021.

Os modelos podem apresentar um ajuste muito bom para o conjunto de dados de treino mas apresentar baixa acurácia para novos dados, ou seja, desempenho ruim para o conjunto de testes. Esse comportamento é referido como *overfitting* (ou sobreajuste, em português), e pode ser atenuado em modelos de regressão através da regularização.

A regularização é uma forma de reduzir os coeficientes β em direção a zero através de uma penalização adicional ao minimizar a função de custo. Na regularização chamada de L_1, essa penalização é aplicada ao valor absoluto dos coeficientes e na regularização chamada de L_2, ela é aplicada ao quadrado dos coeficientes.

3.1.2 Regressão linear

Para modelos de LGD, a variável dependente é contínua (uma taxa de perda contínua). Nesse caso, o modelo utilizado como referência é a regressão linear. O ajuste dos dados é feito através do método dos mínimos quadrados, no qual o melhor ajuste é encontrado minimizando-se a soma dos quadrados dos resíduos entre os valores observados e previstos.

Para a regressão linear, o método de regularização chamado de LASSO é análogo ao método L_1 e o método Ridge é análogo ao método L_2.

3.2 KNN (K-Nearest Neighbors)

A técnica K-Nearest Neighbors (ou K-Vizinhos Mais Próximos, em português) parte da ideia de classificar uma nova observação de acordo com a proximidade dos seus vizinhos. Essa proximidade é medida mais comumente em relação à distância euclidiana no espaço p-dimensional de um *dataset* com p *features.* Por exemplo, para modelos de PD, se k é escolhido como 4, e três dos vizinhos mais próximos apresentarem como resultado *default*, o modelo classifica esse novo contrato com 75% de probabilidade de *default.* Matematicamente, dado o conjunto de treino x com respostas y:

$$G(x) = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i$$

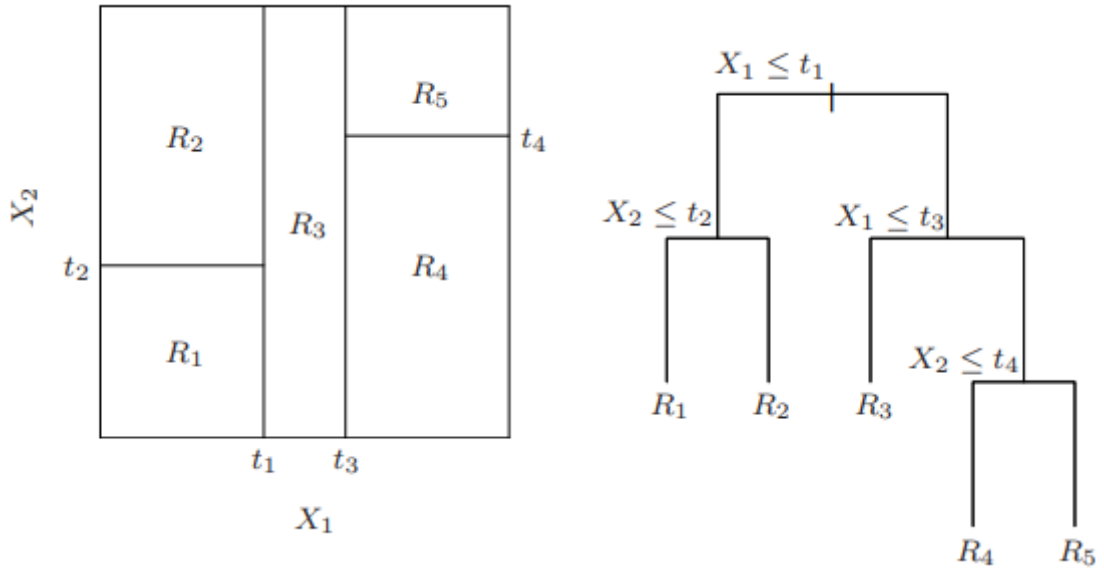
Onde $N_K(x)$ é a vizinhança de x definida pelas k observações mais próximas x_i no subconjunto de treino.

Modelos com valores de k pequenos são muito sensíveis a variações, ou seja, apesar de apresentarem pouco viés (ou *bias,* em inglês), têm muita variância e não são flexíveis para novos dados. O ideal, não só para esse tipo de modelo, mas para todos os outros, é encontrar um meio termo entre viés e variância, o que é chamado na literatura de *bias-variance tradeoff.*

3.3 Árvores de decisão

Em uma árvore de decisão (*decision tree*, em inglês), divide-se recursivamente o espaço de variáveis (*features)* em partições, selecionando uma variável e um ponto de divisão. Essas divisões são feitas minimizando-se uma função de custo para que sejam ótimas, como por exemplo a Impureza de Gini ou entropia para modelos de classificação e erro quadrático médio para modelos de regressão.

Figura 4 – Árvores de decisão. O painel à esquerda mostra uma partição de um espaço de variáveis bidimensional e o painel à direita mostra a árvore de decisão correspondente.

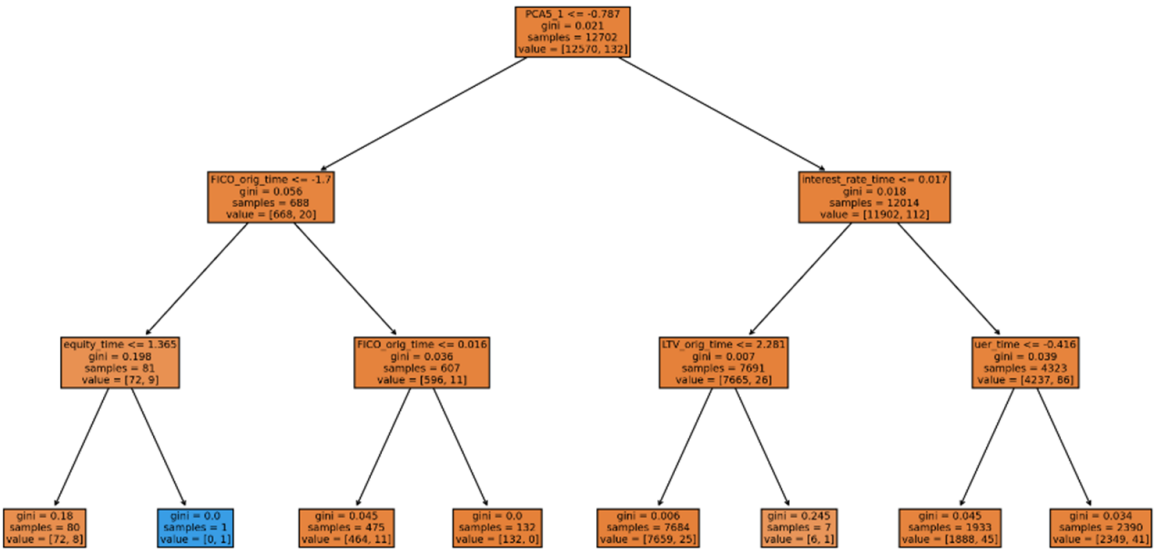


Fonte: Hastie et al., 2017.

Por exemplo, na figura acima, a primeira divisão é feita em $X_1 = t_1$. A região $X_1 \leq t_1$ é então dividida em $X_2 = t_2$ e a região $X_1 > t_1$ é dividida em $X_1 = t_3$, e assim sucessivamente até algum critério de parada, resultando em regiões (ou nós) terminais $R_1, R_2, \ldots, R_5$. Cada divisão gera o que são chamados de ramos (*branches*) e os nós terminais são chamados de folhas (*leaves*). O algoritmo faz, portanto, uma divisão binária recursiva do espaço de variáveis.

No caso de PD, novas observações são classificadas de acordo com a proporção de *defaults* e não *defaults* das observações de treino das folhas correspondentes. No caso de LGD, os valores estimados para novas observações são dados pela média dos valores das observações de treino da respectiva folha resultante.

Figura 5 – Exemplo de árvore de decisão gerada com a biblioteca *scikit-learn.*



Fonte: de autoria própria.

As árvores de decisão tem a vantagem de serem fáceis de interpretar, mas possuem baixa acurácia (Hastie et al., 2017).

3.4 Métodos Ensemble

Os métodos anteriores são métodos independentes. Isso significa que após inicializado o modelo, o ajuste (*fit*) dos dados é feito uma única vez. Uma extensão disso é o ajuste com *tuning* de hiperparâmetros e validação cruzada, já que envolve várias estimativas. No entanto, para isso é utilizada uma classe fixa de modelo. Após o ajuste dos hiperparâmetros, o *fit* é feito uma única vez e os resultados são computados.

Uma alternativa são os métodos chamados de *ensemble* (ou conjunto, em português). A ideia por trás desses métodos é combinar vários estimadores mais simples e com menor poder preditivo para gerar modelos mais complexos e com maior acurácia. Dentre as classes de modelos *ensemble*, as mais utilizadas e que são abordadas neste trabalho são: *Bagging* e *Boosting*.

Na metodologia *Bagging* (junção dos termos ***b**ootstrapping* e ***agg**regation*), os estimadores base são treinados separadamente a partir de amostras com reposição (*bootstrapping*) do conjunto de dados de treino. Os resultados desses estimadores então são agregados de alguma forma, como por exemplo a maioria dos votos para modelos de classificação ou a média dos resultados para modelos de regressão.

A ideia é combinar estimadores com alta variância e baixo viés, e assim reduzir a variância do estimador final. A técnica funciona especialmente bem para procedimentos como as árvores de decisão, como destaca Hastie et al. (2017). Matematicamente, a variância da média de *B* estimadores, considerando que cada árvore de decisão é identicamente distribuída e não independente (e logo tem uma correlação ρ com as demais), é dada por:

$$\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$$

Se os estimadores forem relativamente descorrelacionados, à medida que *B* aumenta, o segundo termo tende a zero e a variância do estimador final é tão mais baixa quanto mais baixo for a correlação ρ.

Já na metodologia *Boosting*, os preditores são treinados sequencialmente, com os sucessores tentando melhorar o resultado dos antecessores. O algoritmo é aplicado até que se atinja algum critério de parada, sempre observando o princípio de *bias-variance tradeoff* para que não ocorra o *overfitting* dos dados de treino.

Os estimadores base são estimadores “fracos”, e são ligeiramente melhores que um palpite aleatório. Esses estimadores são então combinados através de uma votação ponderada para gerar o estimador final, na qual cada estimador tem um peso α que é tão maior quanto maior for sua acurácia. Para classificação, por exemplo, o estimador final é dado por:

$$G(x) = sign\left(\sum_{m=1}^{M} \alpha_m G_m(x)\right)$$

3.4.1 Random Forest

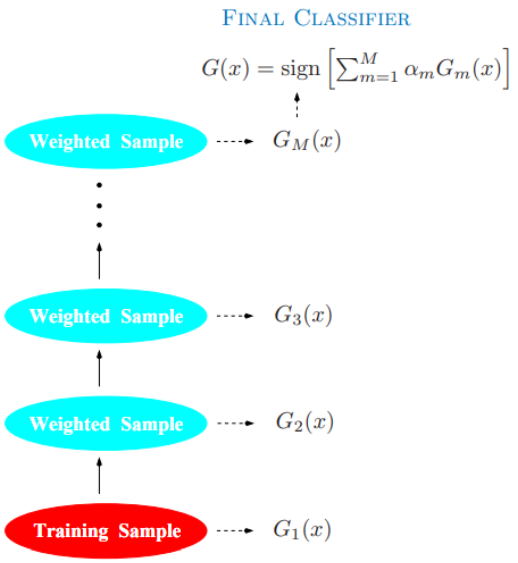
O método *Random Forest* consiste na aplicação dos conceitos de *bagging* a árvores de decisão. A variância do conjunto é melhorada reduzindo a correlação entre as árvores. Isso é feito no processo de crescimento das árvores através de seleção aleatória das variáveis de entrada, escolhendo a cada divisão um número de variáveis $m <= p$. Tipicamente m é menor ou igual a $\sqrt{p}$ (Hastie et al., 2017).

3.4.2 Adaboost

O algoritmo Adaboost, introduzido por Freund e Schapire (1997), foi o primeiro do tipo *boosting*.

A principal ideia do algoritmo é que os estimadores base treinados sequencialmente compensam os erros dos estimadores anteriores. Isso é feito aumentando o peso das amostras de treino com maior erro e diminuindo caso contrário, o que faz com que os próximos estimadores dêem um foco maior para as amostras com maior erro. A importância (ou peso) de um estimador base no estimador final depende do seu desempenho.

Figura 6 – Esquemático do algoritmo AdaBoost.



Fonte: Hastie et al., 2017.

Na prática, os estimadores base utilizados são árvores de decisão, geralmente árvores com somente uma variável, ou seja, que possuem somente um nó e duas folhas. São também chamadas de “toco” (ou *stump*, em inglês).

O algoritmo usado para problemas de classificação, descrito por Freund e Schapire (1997), é chamado de “AdaBoost.M1” e é dado pelo seguinte pseudocódigo:

Algoritmo *AdaBoost.M1.*

---

1. Dadas: *N* observações de treino (x_i, y_i),..., (x_N, y_N)
2. Inicialize os pesos $w_i = 1/N,\ i = 1, 2,..., N$
3. Para $m = 1$ até M faça:
   (a) Treine um classificador $G_m(x)$ usando os pesos w_i
   (b) Calcule o erro do classificador:
   $$err_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}$$
   (c) Calcule o peso do classificador:
   $$\alpha_m = ln(\frac{1-err_m}{err_m})$$
   (d) Atualize os pesos das observações:
   $$w_i \leftarrow w_i e^{\alpha_m},\ \ se\ y_i \neq G_m(x_i)$$
   $$w_i \leftarrow w_i e^{-\alpha_m},\ se\ y_i = G_m(x_i)$$
4. *Output*:
   $$G(x) = sign\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$$

O algoritmo adaptado para problemas de regressão, chamado de “AdaBoost.R2”, é análogo ao algoritmo de classificação, e é descrito com detalhes em Drucker (1997).

3.4.3 Gradient Boosting

O algoritmo Gradient Boosting, introduzido por Friedman (2001), é uma generalização dos algoritmos de boosting. O nome “gradiente” vem do fato da técnica calcular o gradiente descendente da função de perda a cada iteração, com o objetivo de minimizá-la.

Em contraste com o algoritmo de Adaboost, que implicitamente minimiza uma função de perda exponencial, o Gradient Boosting permite qualquer função de perda L diferenciável. Ambos os algoritmos dão um foco maior aos maiores erros a cada iteração, mas no algoritmo Adaboost isso é feito através do rebalanceamento dos pesos das amostras e no algoritmo Gradient Boosting as amostras com maior erro são identificadas por meio dos maiores pseudo-resíduos calculados. Outra diferença é que o primeiro utiliza *stumps* como estimadores base e o segundo utiliza árvores de decisão com profundidade um pouco maior.

O algoritmo generalizado é dado pelo seguinte pseudocódigo:

Algoritmo *Gradient Tree Boosting*

---

1. Dadas: *N* observações de treino $(x_i,\ y_i)$,..., $(x_N,\ y_N)$
2. Inicialize o modelo com um valor constante: $F_0(x) = argmin_\gamma \sum_{i=1}^{N} L(y_i, \gamma)$
3. Para $m = 1$ até M faça:
   (a) Para i = 1,...,N calcule:
   $$r_{im} = - \left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} \right]_{F = F_{m-1}}$$
   (b) Ajuste uma árvore de decisão os pseudo-resíduos r_{im} e crie regiões terminais R_{jm}, $j = 1,...,J_m$
   (c) Para $j = 1$ até J_m calcule:
   $$\gamma_{jm} = argmin_\gamma \sum_{x_i \in R_{jm}} L(y_i, F_{m-1}(x_i) + \gamma)$$
   (d) Atualize $F_m(x) = F_{m-1}(x) + \alpha \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$
4. *Output*:
   $F_M(x)$

Como destaca Forti (2018), um fator importante para a aplicação é a escolha dos hiperparâmetros, em especial: o número de iterações M, a taxa de aprendizagem α, a profundidade das árvores e o número de indivíduos por nó.

3.4.4 XGBoost

Recentemente, um algoritmo que vem ganhando muita popularidade por ser a escolha de muitas equipes vencedoras em competições de *machine learning* (Github, 2022) é o algoritmo XGBoost introduzido por Chen e Guestrin (2016). É uma implementação de Gradient Boosting projetada para velocidade e desempenho.

A biblioteca de software por trás do método traz várias características como processamento em paralelo, otimização de memória e *cache,* formas de tratar valores faltantes (*missing values*), regularizações, entre outras, que fazem o algoritmo seja considerado do estado da arte.

3.5 Tuning de hiperparâmetros

Hiperparâmetros são parâmetros de um modelo de *machine learning* que não podem ser estimados a partir dos dados. Eles controlam o aprendizado dos modelos. *Tuning* refere-se ao conjunto de procedimentos realizados para encontrar os hiperparâmetros que produzem os melhores desempenhos nos modelos.

Uma das abordagens para realizar o *tuning* é a busca exaustiva, chamada de *Grid Search*. Nesse tipo de busca testa-se exaustivamente combinações de parâmetros para se encontrar a combinação com a melhor pontuação segundo uma métrica de avaliação. Apesar do conceito ser simples, cada novo parâmetro inserido na busca causa um aumento exponencial no número de combinações a serem testadas, como destaca Pellicer et al (2019).

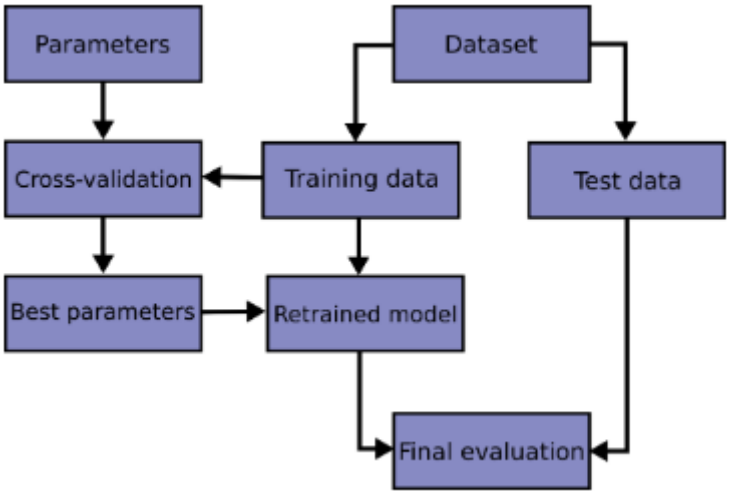
Outra abordagem é a busca aleatória, chamada de *Random Search*. Nesse tipo de busca são testadas combinações de parâmetros selecionadas de alguma forma de distribuição pré-definida.

Essas duas abordagens são utilizadas no trabalho. Existem diversos outros tipos de abordagem, como o algoritmo desenvolvido por Pellicer et al. (2019) que utiliza as propriedades da equação do baricentro para otimizar os hiperparâmetros.

3.6 Validação cruzada

Para treinamento de um modelo a base de dados deve ser sempre dividida em subconjuntos de treino e teste. Assim, o modelo é treinado utilizando os dados de treino e seu desempenho é avaliado utilizando os dados de teste, que são dados não vistos anteriormente e que simulam o poder de predição em um ambiente de produção com novos dados. Caso contrário aconteceria *overfitting* já que o modelo conseguiria prever apenas resultados já vistos.

Figura 7 – Fluxograma de treinamento de modelos utilizando validação cruzada.



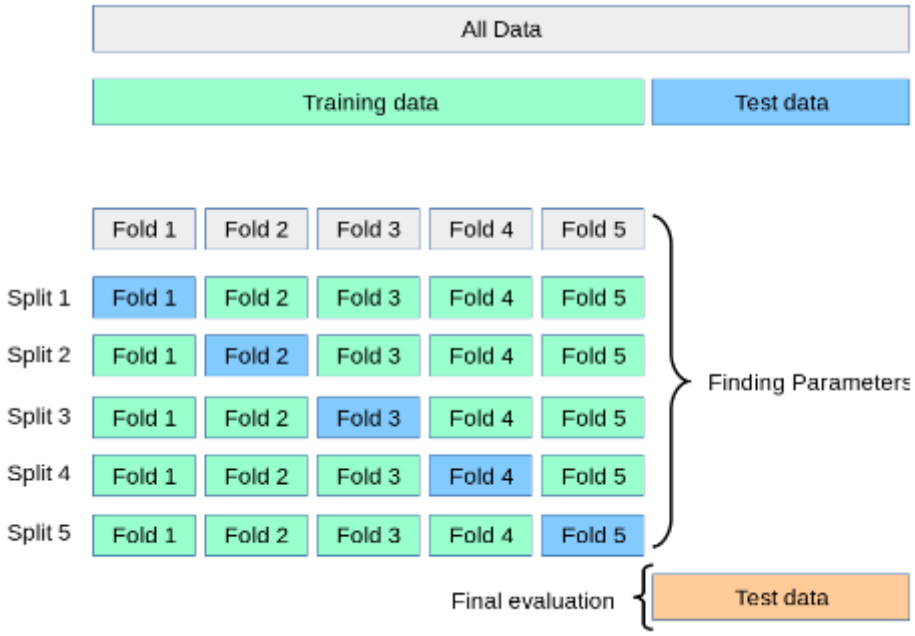
Fonte: *Scikit Learn.*

Além disso, quando avalia-se diferentes conjuntos de hiperparâmetros através do *tuning,* ainda existe o risco de *overfitting* se forem utilizados apenas os subconjuntos de treino e teste, pois os hiperparâmetros podem ser ajustados até que o modelo tenha um bom desempenho para treino e teste, mas não necessariamente terá um bom desempenho para novos dados.

Para solucionar esse problema, pode-se utilizar o procedimento de validação cruzada (*cross-validation*). Nessa abordagem, o subconjunto de treino é dividido aleatoriamente em *k* partes iguais. Utiliza-se $k - 1$ partes para treinamento e a

k-ésima parte restante, chamada de parte de validação, é utilizada para avaliação do desempenho. Isso é feito iterativamente *k* vezes selecionando a cada iteração uma nova parte para validação. O desempenho de um conjunto de hiperparâmetros é então calculado como a média dos valores calculados nas iterações.

Figura 8 – Divisão dos subconjuntos de treino e teste na modelagem com validação cruzada.



Fonte: *Scikit Learn.*

3.7 Métricas de validação

Existem diversas métricas que podem ser utilizadas para avaliar o desempenho dos modelos e compará-los entre si. Neste trabalho foram utilizadas:

- AUROC para modelos de PD
- RMSE para modelos de LGD
- Análise de ordenação de score para modelos de PD e LGD

3.7.1 AUROC

Uma matriz de confusão é uma tabela que mostra as frequências de classificação para cada classe do modelo:

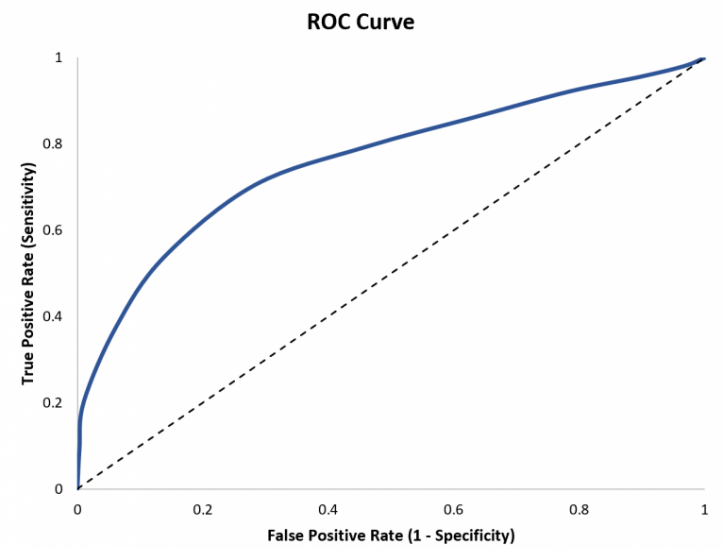
Figura 9 – Matriz de confusão.

		Valor Observado	
		Positivos	Negativos
Valor Previsto	Positivos	VP - Verdadeiro Positivo	FP - Falso Positivo
	Negativos	FN - Falso Negativo	VN - Verdadeiro Negativo

Fonte: Forti, 2018.

A área sob a curva ROC (AUROC ou AUC) conceitualmente é a área sobre o gráfico da taxa de verdadeiros positivos (também conhecida como sensibilidade) contra a taxa de falsos positivos (também chamada de 1-especificidade) plotados variando-se o *threshold* (PD) de 0 a 1. Quanto mais esse número for próximo de 1, melhor é o poder de discriminação do modelo.

Figura 10 – Curva ROC.



Fonte: Statology.org.

3.7.2 RMSE

O raiz quadrada do erro quadrático médio é uma medida da diferença entre o valor previsto pelo modelo e o valor observado:

$$RMSE = \sqrt{\sum \frac{(y_{pred} - y_{obs})^2}{N}}$$

3.7.3 Análise de ordenação de score

Além das duas métricas citadas anteriormente, uma outra análise foi desenvolvida para comparação entre os modelos: a análise de ordenação de score e avaliação de pontos de corte.

Para modelos de PD, a construção segue os seguintes passos:

1. Ordenação da base de dados pela probabilidade (score), considerando a ordenação do pior para o melhor, ou seja, da maior probabilidade de *default* para a menor.
2. Cálculo das seguintes métricas para cada ponto de corte: % não *default*, % não *default* acumulado e % *default* acumulado.

A tabela abaixo apresenta um exemplo de análise. Para os 40% classificados com a maior probabilidade de *default,* temos uma porcentagem de não *default* de 94,2% para o modelo 1 e 93,6% para o modelo 2, uma porcentagem acumulada de não *default* de 39,2% para o modelo 1 e 39% para o modelo 2 e uma porcentagem acumulada de default de 62,1% para o modelo 1 e 68,4% para o modelo 2. Os números são indicativos que a ordenação do modelo 2 é superior à do modelo 1, ou seja, o modelo 2 é superior na identificação de contratos que têm maior probabilidade de inadimplência.

Tabela 1 – Exemplo de análise de ordenação de *score* para modelos de PD.

Corte	% Não default LR		% Não default acum		% Default acum	
	M1	M2	M1	M2	M1	M2
1,0%	92,9%	92,1%	1,0%	1,0%	1,9%	2,1%
5,0%	93,7%	91,5%	4,9%	4,8%	8,5%	11,1%
10,0%	93,6%	91,7%	9,7%	9,6%	17,1%	21,8%
20,0%	93,7%	92,5%	19,5%	19,3%	33,8%	39,8%
30,0%	94,0%	93,0%	29,3%	29,0%	48,1%	56,2%
40,0%	94,2%	93,6%	39,2%	39,0%	62,1%	68,4%
50,0%	94,5%	94,1%	49,1%	48,9%	73,7%	79,9%
60,0%	94,9%	94,6%	59,2%	59,0%	82,9%	87,5%
70,0%	95,3%	95,1%	69,3%	69,1%	90,3%	93,8%
80,0%	95,6%	95,5%	79,4%	79,4%	95,2%	97,8%
90,0%	96,0%	96,0%	89,7%	89,7%	98,6%	99,5%
95,0%	96,2%	96,2%	94,8%	94,8%	99,6%	99,9%
99,0%	96,3%	96,3%	99,0%	99,0%	100,0%	100,0%
100,0%	96,4%	96,4%	100,0%	100,0%	100,0%	100,0%

Fonte: de autoria própria.

Já para os modelos de LGD, a construção segue os seguintes passos:

1. Ordenação da base de dados pela perda estimada, considerando a ordenação da menor perda estimada para a maior perda estimada.
2. Cálculo das seguintes métricas para cada ponto de corte: recuperação (em $ MM, valores reais), e % de recuperação do total

Nesse tipo de avaliação, calcula-se como uma aproximação para valores recuperados o saldo do contrato no momento do *default* multiplicado por $1 - LGD_{realizada}$.

A tabela abaixo mostra um exemplo de análise. Para os 20% com menor perda estimada, temos uma recuperação de $80,2 MM ou 23,4% do risco total da base para o modelo 1 e uma recuperação de $88 MM ou 25,7% do risco total da base para o modelo 2. Pode-se pensar na ordenação como uma ação de cobrança para os 20% da base em *default* que têm a menor perda estimada, que se traduz em uma maior chance de recuperação. Nesse caso, o modelo 2 proporcionaria um ganho financeiro de $7,8 MM em relação ao modelo 1.

Tabela 2 – Exemplo de análise de ordenação de *score* para modelos de LGD.

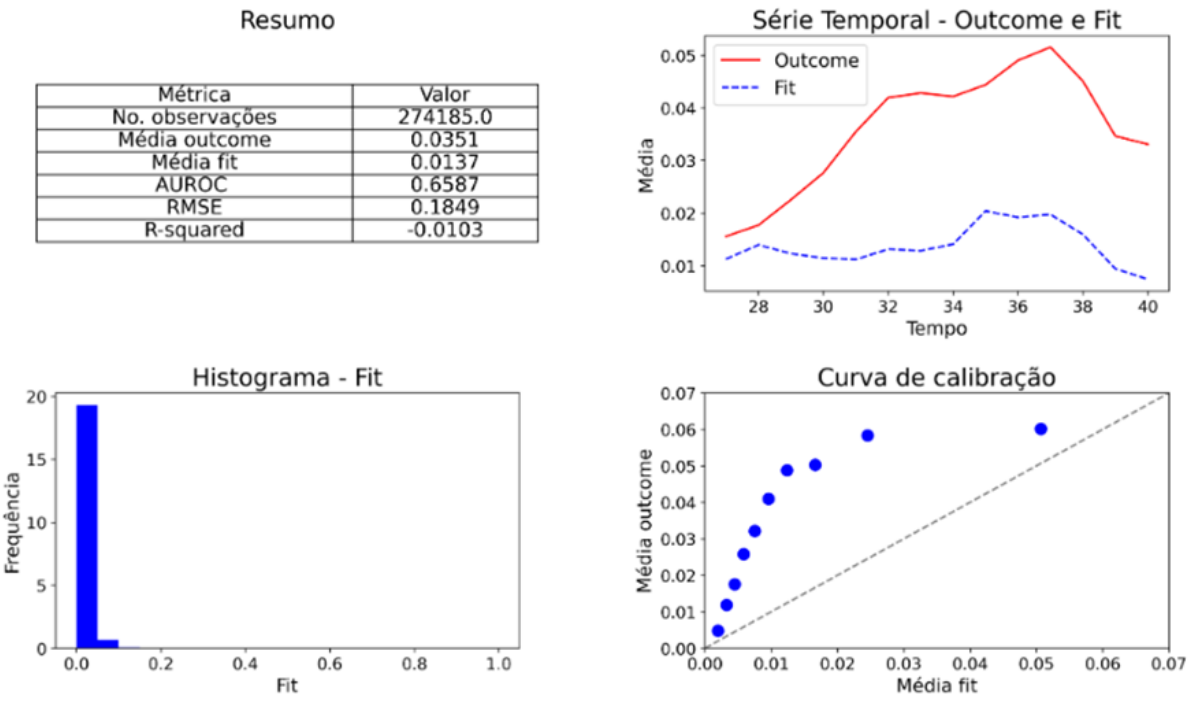
	$ recuperação		% recuperação	
corte	M1	M2	M1	M2
1%	2,0	3,4	0,6%	1,0%
5%	18,0	19,6	5,3%	5,7%
10%	35,5	41,7	10,4%	12,2%
20%	80,2	88,0	23,4%	25,7%
30%	124,8	127,7	36,5%	37,3%
40%	167,9	168,0	49,1%	49,1%
50%	205,0	203,5	59,9%	59,5%
60%	240,9	237,2	70,4%	69,3%
70%	271,7	271,0	79,4%	79,2%
80%	299,5	298,6	87,5%	87,3%
90%	323,9	321,6	94,7%	94,0%
95%	334,5	332,7	97,8%	97,2%
99%	341,2	341,1	99,7%	99,7%
100%	342,2	342,2	100,0%	100,0%

Fonte: de autoria própria.

3.8 Painéis de validação

Com o intuito de facilitar a visualização e comparação dos resultados, criou-se para cada modelo painéis com as principais métricas de validação e gráficos, como sugerido por Rösch e Scheule (2020). Como exemplo, para um determinado modelo são apresentados os seguintes painéis:

Figura 11 – Exemplo de painéis de validação



Fonte: de autoria própria.

- painel superior esquerdo: tabela com resumo das principais métricas de validação;
- painel superior direito: média dos valores observados (*outcome*) e valores ajustados (*fit*) da variável alvo ao longo do tempo;
- painel inferior esquerdo: histograma dos valores ajustados (*fit*);
- painel inferior direito: curva de calibração do modelo.

4 METODOLOGIA

4.1 Base de dados

No trabalho foi utilizado um *dataset* contendo 50.000 hipotecas (*mortgages)* americanas, observadas trimestralmente por 60 períodos, desde o começo dos anos 2000 e englobando a crise financeira de 2008. O *dataset* é fornecido pelo órgão *International Financial Research,* e contém mais de 15.000 inadimplentes (*default*), o que é adequado para ajuste de modelos de risco e recuperação de crédito.

O *dataset* possui 28 atributos (*features*). São eles:

- *id*: identificação do contrato
- *time*: período de observação
- *orig_time*: período de origem do contrato
- *first_time*: primeira observação do contrato
- *mat_time*: período de vencimento do contrato
- *res_time*: período de resolução do contrato
- *balance_time*: saldo devedor no período
- *LTV_time*: *loan-to-value ratio* no período
- *interest_rate_time*: taxa de juros do contrato
- *rate_time*: taxa de juros livre de risco no período
- *hpi_time*: *house price index* no período
- *gdp_time*: crescimento do PIB no período
- *uer_time*: taxa de desemprego no período
- *REtype_CO_orig_time*: tipo de imóvel - *condominium*: 1
- *REtype_PU_orig_time*: tipo de imóvel - *planned urban developments*: 1
- *REtype_SF_orig_time*: tipo de imóvel - *single family home*: 1
- *investor_orig_time*: *investor borrower*: 1
- *balance_orig_time*: saldo devedor no momento da originação
- *FICO_orig_time*: FICO *score* no momento da originação
- *LTV_orig_time*: *loan-to-value ratio* no momento da originação
- *interest_rate_orig_time*: taxa de juros no momento da originação
- *state_orig_time*: estado no qual o imóvel está localizado

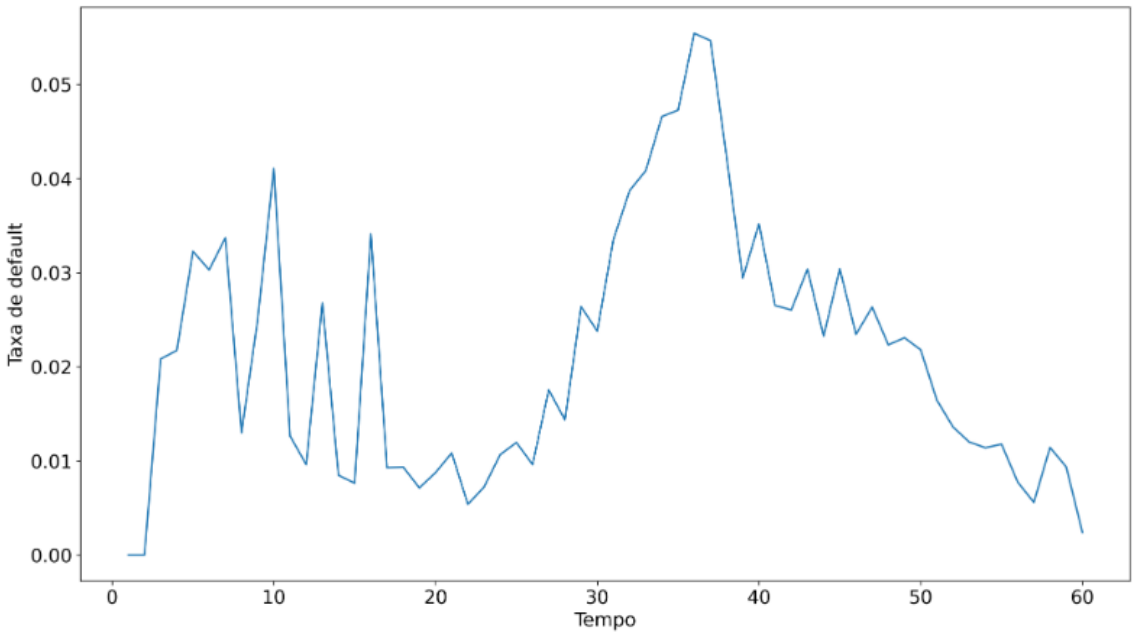
- *hpi_orig_time*: *house price index* no momento da originação
- *default_time*: indicador de *default* no período
- *payoff_time*: indica se o contrato foi quitado no período
- *status_time*: 0:não *default*/não *payoff;* 1: *default*, 2: *payoff*
- *ldg_time*: *loss given default*
- *recovery_res*: soma de todos os fluxos de caixas recebidos nos períodos de resolução (após *default*)

O *dataset* é organizado da seguinte forma. Cada contrato tem um resultado, que é observado 1 período após as *features*: *survival* (sobrevivência), *default* (inadimplência), *payoff* (pagamento), *maturity* (vencimento). Os três últimos são eventos de encerramento, ou seja, o contrato deixa de ser observado caso ocorra um deles. Variáveis como *lgd_time* e *recovery_res* são observadas na mesma linha do evento de encerramento.

Foram selecionadas 12 *features* relevantes para os modelos de PD e 8 *features* mais variáveis indicadoras do estado de origem para os modelos de LGD para ajuste dos modelos, incluindo algumas *features* calculadas a partir das originais.

Para os modelos de PD, o *dataset* foi dividido em subconjuntos até o período 27 para treino e do período 27 ao 40 para teste, observando que a crise financeira de 2007-2008 começou aproximadamente do período 27 com término aproximadamente no período 40. Essa divisão é indicada pelos próprios pesquisadores que disponibilizaram o *dataset*, porque assim o desempenho dos modelos pode ser testado no cenário mais adverso possível. Nota-se no gráfico abaixo, que a partir do período 27 há um aumento rápido da taxa de *default*. Até o período 27 a média era de 1,1% e entre o período 27 e o período 40 a média era de 3,5%.

Figura 12 – Taxa de *default* do *dataset* por períodos



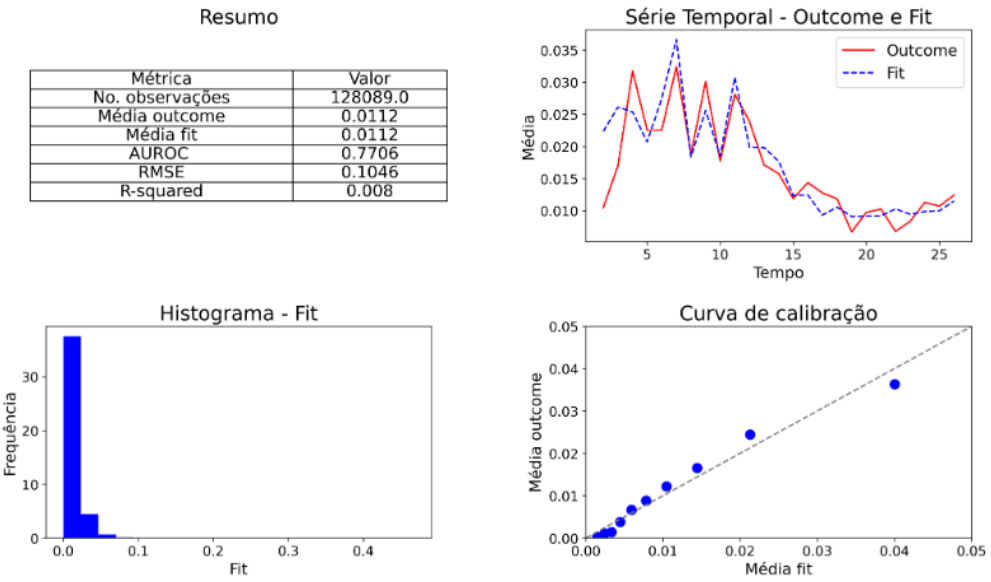
Fonte: de autoria própria.

Para modelos de LGD, o *dataset* foi dividido em subconjuntos até o período 40 para treino e até o período 60 (último período) para testes. A diferença da divisão da base para PD e LGD tem o intuito de garantir um número suficiente de *defaults* para ajuste dos modelos de LGD (aproximadamente 11.000 para treino e 4000 para testes).

Em um primeiro momento foi utilizado uma amostra de 5.000 contratos com a mesma representatividade de defaults do *dataset* completo para estudo das técnicas, por questões de tempo de processamento para gerar os modelos. Posteriormente foi utilizada a base completa com os 50.000 contratos. Para o *tuning* de hiperparâmetros o *dataset* de treino foi dividido recursivamente usando validação cruzada em 5 partes para treino dos modelos.

Como exemplo de modelagem utilizando os *datasets* de treino e teste, o modelo de regressão logística, sem regularização e *tuning* de hiperparâmetros apresentou os seguintes resultados para o subconjunto de treino:

Figura 13 – Modelo Regressão logística de PD, sem regularização e sem *tuning* de hiperparâmetros - subconjunto de treino.

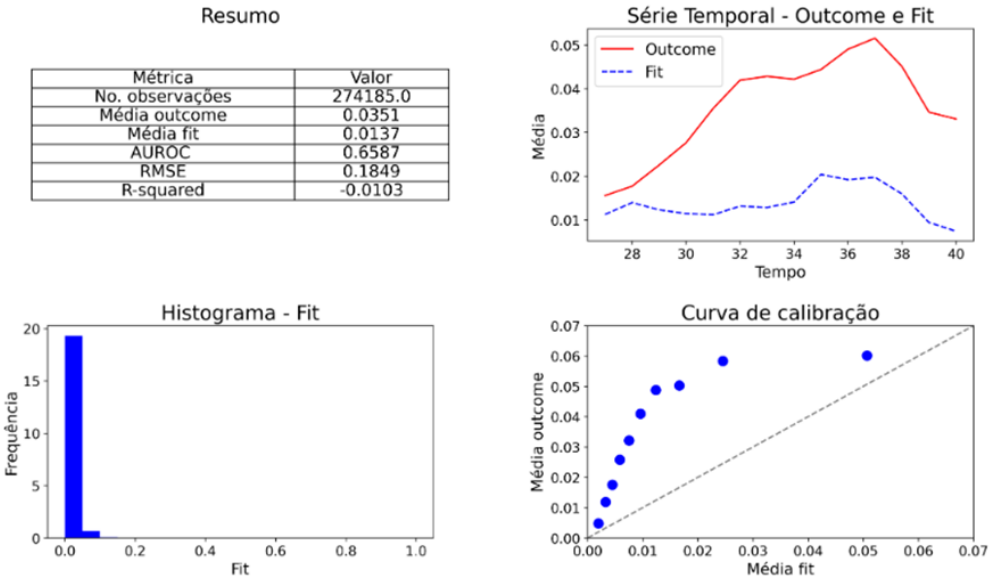


Fonte: de autoria própria.

No painel superior esquerdo observa-se um AUROC alto pois os dados são de treino. No painel superior direito percebe-se que o ajuste (*fit)* é próximo do observado (*outcome)* ao longo do tempo. No canto inferior esquerdo o histograma apresenta concentração próxima de 0%, pois no período pré-crise os contratos apresentaram probabilidade baixa de *default*. No canto inferior direito a curva de calibração mostra uma boa calibração.

Este primeiro modelo apresentou bom fit para o subconjunto de treino mas subestimou as probabilidades de *default* para o subconjunto de teste, como pode ser visto nos painéis abaixo, tanto na série temporal quanto na curva de calibração, apresentando um AUROC menor.

Figura 14 – Modelo Regressão logística de PD, sem regularização e sem *tuning* de hiperparâmetros - subconjunto de teste.



Fonte: de autoria própria.

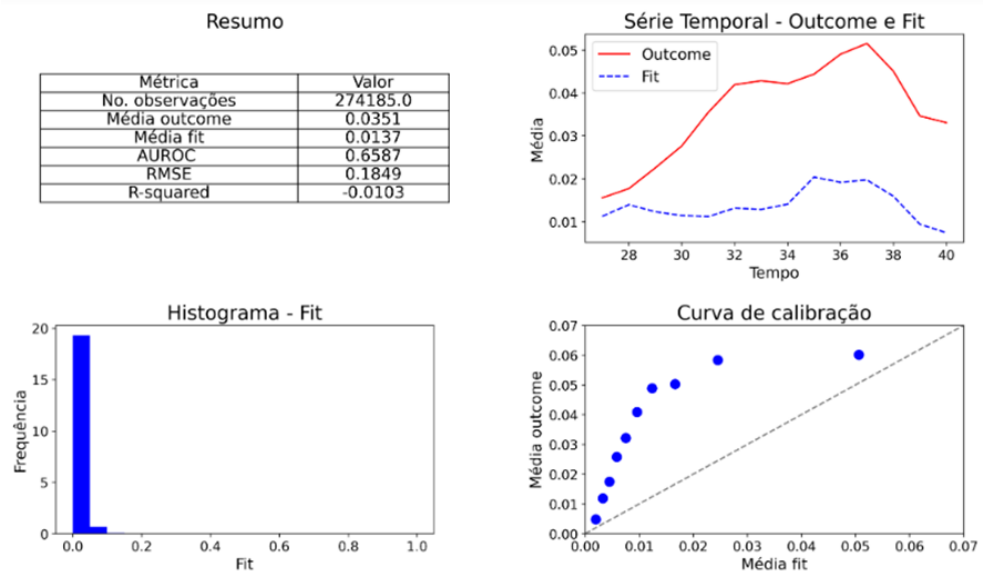
Para os próximos modelos serão apresentados apenas os painéis utilizando a base completa e subconjunto de teste. As bibliotecas *scikit-learn* (Pedregosa et. al, 2011) e *xgboost* (Chen e Guestrin, 2016) foram utilizadas para implementar os modelos. A lista de hiperparâmetros considerados nos modelos com *tuning* e suas respectivas descrições e valores podem ser consultados no Apêndice.

4.2 Modelos PD

4.2.1 Regressão logística para PD

A regressão logística é implementada através da classe *LogisticRegression* do pacote *scikit-learn*. O *tuning* de hiperparâmetros foi feito através de busca exaustiva com *GridSearchCV*, procurando o melhor desempenho para os hiperparâmetros: tipo de regularização, força da regularização e número máximo de iterações.

Figura 15 – Modelo Regressão logística de PD, com regularização e *tuning* de hiperparâmetros.

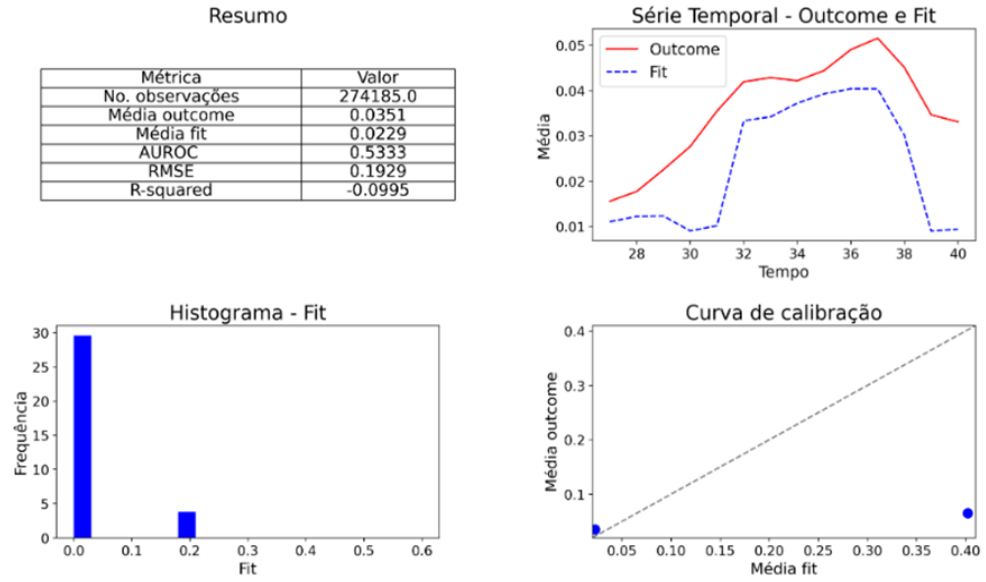


Fonte: de autoria própria.

4.2.2 KNN para PD

O algoritmo KNN é implementado pela classe *KNeighborsClassifier.* Para $k = 5$, o ajuste apresenta *overfitting* (sobreajuste) e não apresenta bons resultados com os dados de teste:

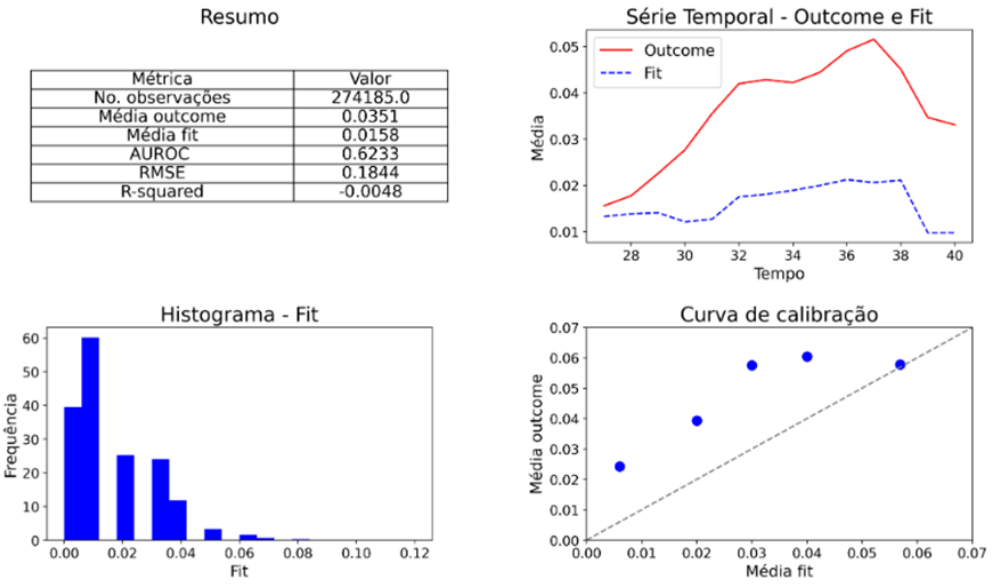
Figura 16 – Modelo KNN de PD, sem *tuning* de hiperparâmetros - *k = 5.*



Fonte: de autoria própria.

Modelo com $k = 100$:

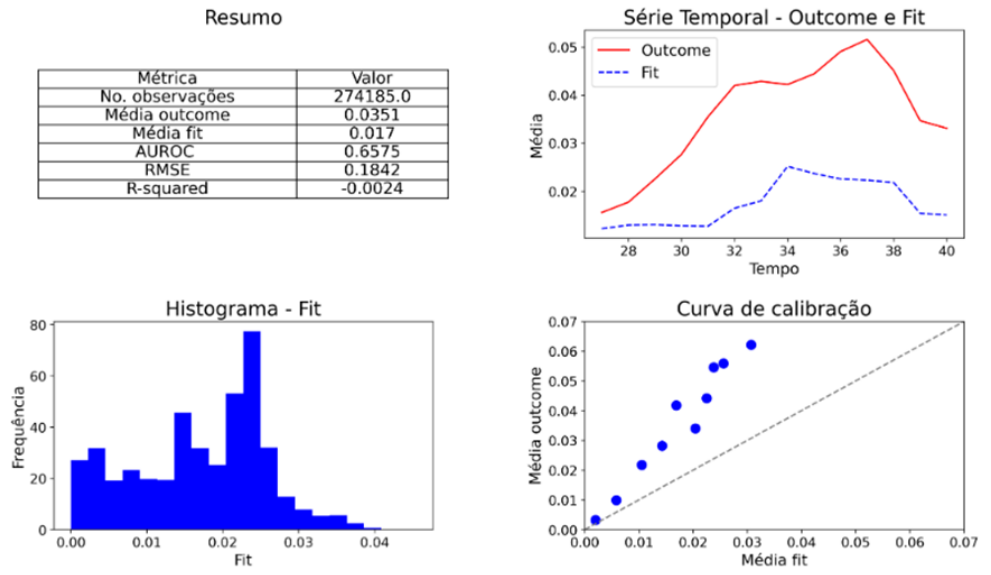
Figura 17 – Modelo KNN de PD, sem *tuning* de hiperparâmetros - *k = 100.*



Fonte: de autoria própria.

Modelo com $k = 2048$, hiperparâmetro otimizado encontrado utilizando *GridSearchCV*:

Figura 18 – Modelo KNN de PD, com *tuning* de hiperparâmetros - *k = 2048.*

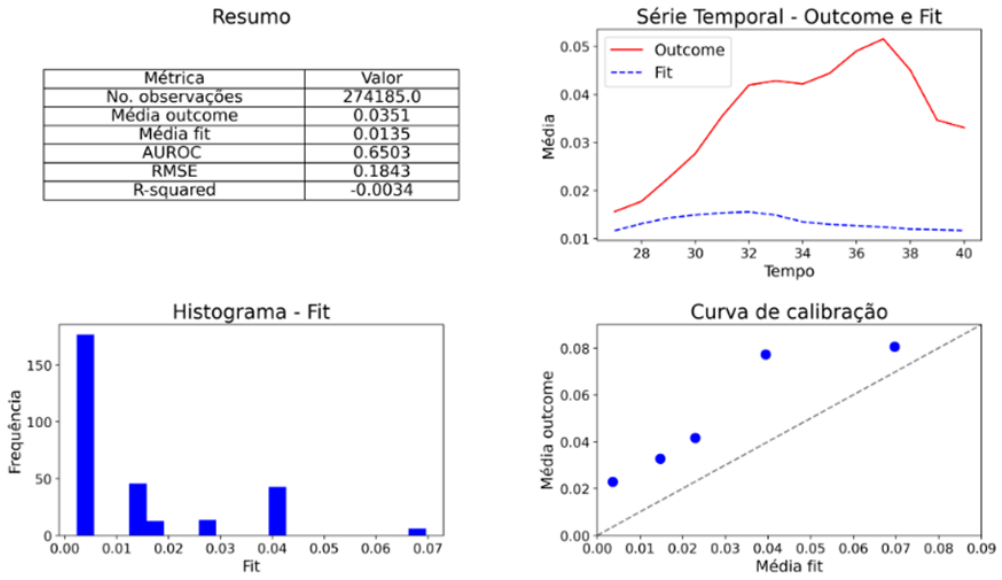


Fonte: de autoria própria.

4.2.3 Árvores de Decisão para PD

As árvores de classificação são implementadas através da classe *DecisionTreeClassifier.* Modelo sem *tuning*:

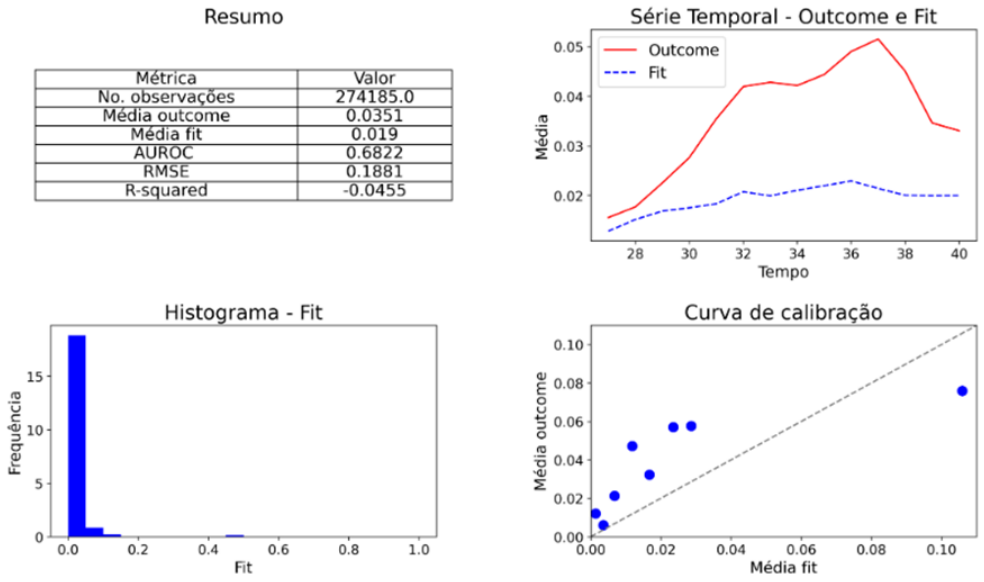
Figura 19 – Modelo Árvore de decisão de PD, sem *tuning* de hiperparâmetros.



Fonte: de autoria própria.

Modelo com *tuning*, utilizando *GridSearchCV* para encontrar os hiperparâmetros: profundidade máxima da árvore, número máximo de *features* consideradas a cada divisão e porcentagem de amostras do nó necessárias para que seja feita uma nova divisão:

Figura 20 – Modelo Árvore de decisão de PD, com *tuning* de hiperparâmetros.

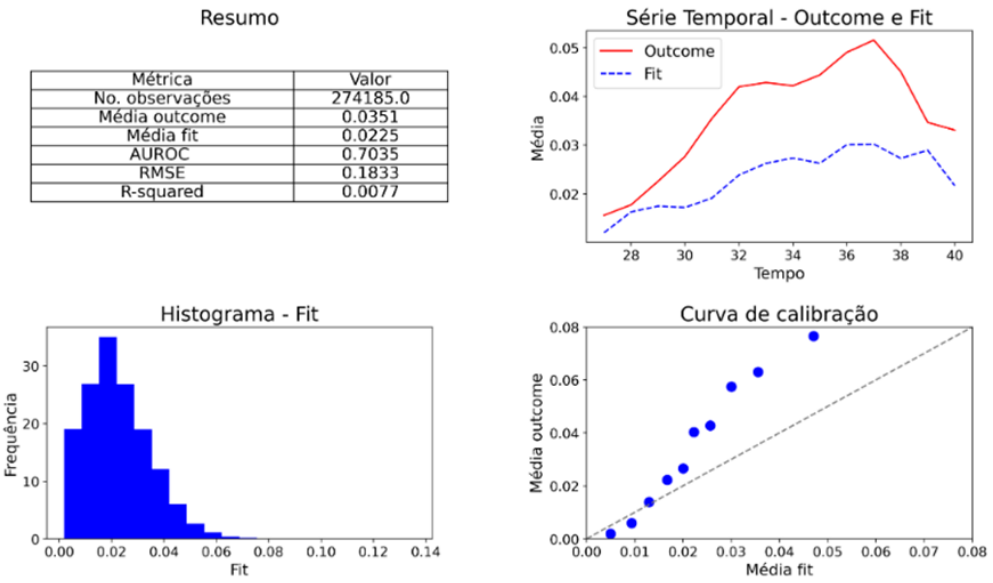


Fonte: de autoria própria.

4.2.4 Random forest para PD

O estimador *random forest* é implementado através da classe *RandomForestClassifier.* Modelo sem *tuning*:

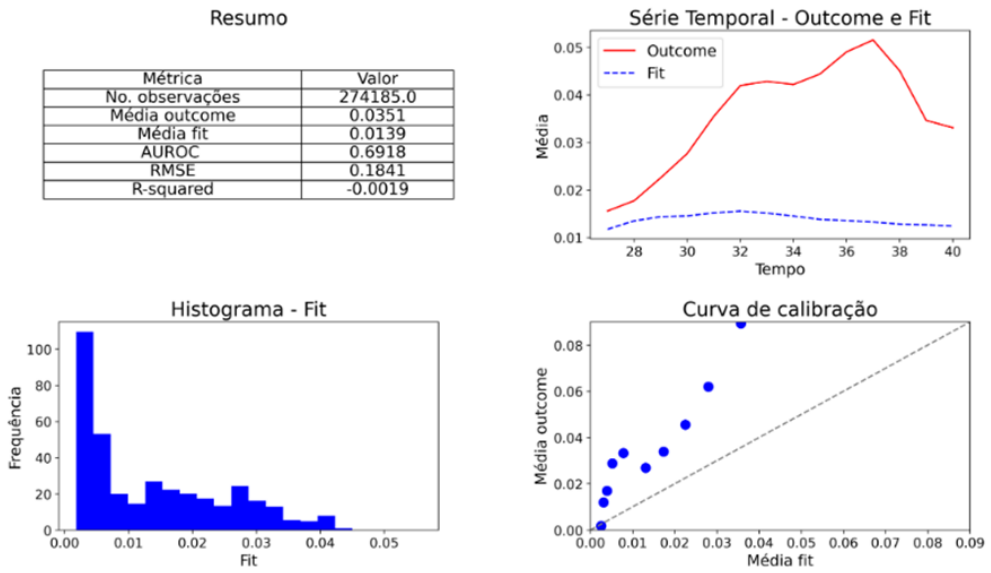
Figura 21 – Modelo Random forest de PD, sem *tuning* de hiperparâmetros.



Fonte: de autoria própria

Modelo com *tuning*, utilizando *RandomizedGridSearch* com 50 combinações para encontrar os hiperparâmetros otimizados de número de árvores, profundidade máxima das árvores, número máximo de *features* consideradas a cada divisão, porcentagem de amostras do nó necessárias para que seja feita uma nova divisão e se é utilizado o *bootstrapping* na construção das árvores:

Figura 22 – Modelo Random forest de PD, com *tuning* de hiperparâmetros.

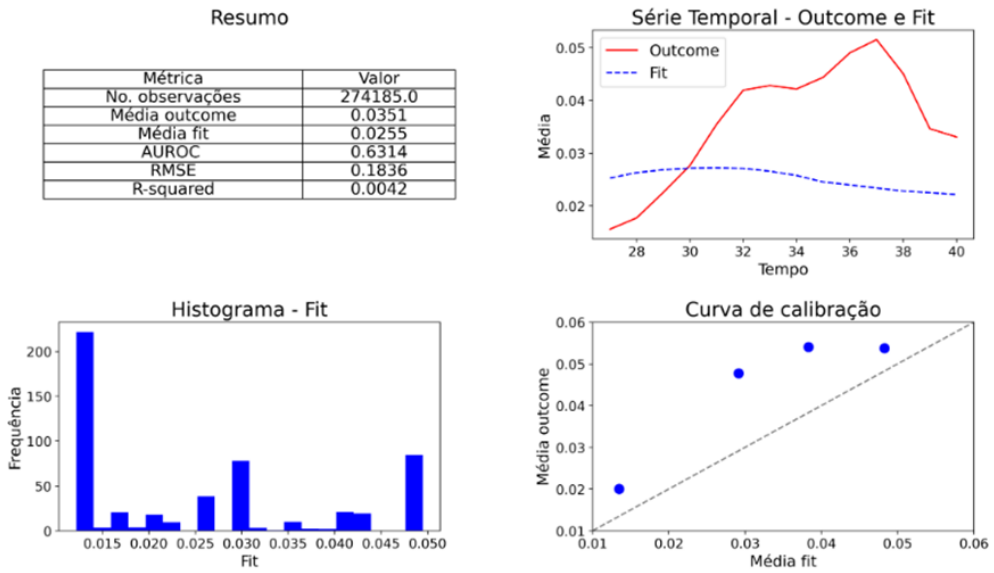


Fonte: de autoria própria.

4.2.5 Adaboost para PD

O algoritmo AdaBoost é implementado por meio da classe *AdaBoostClassifier.* Modelo sem *tuning* de hiperparâmetros e utilizando como estimadores base árvores de decisão com profundidade igual a 1 (*stump*):

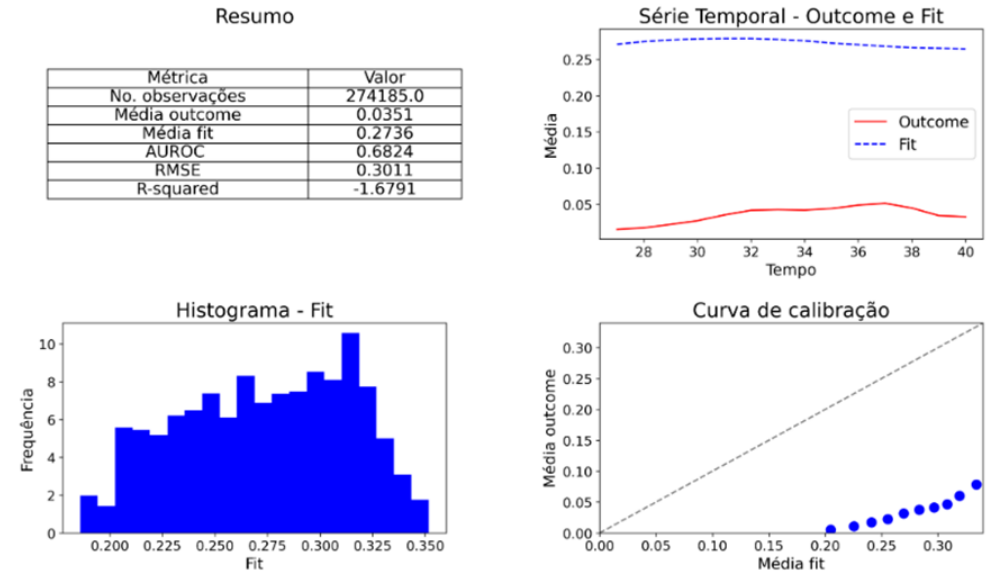
Figura 23 – Modelo AdaBoost de PD, sem *tuning* de hiperparâmetros.



Fonte: de autoria própria.

Modelo com *tuning,* com hiperparâmetros otimizados de taxa de aprendizagem e número de estimadores base encontrados com *GridSearchCV*:

Figura 24 – Modelo AdaBoost de PD, com *tuning* de hiperparâmetros.

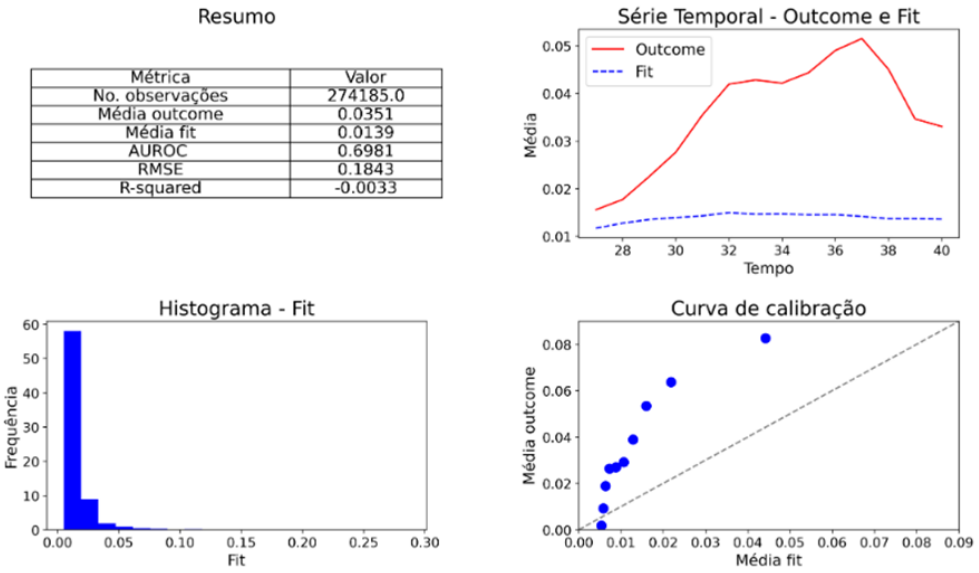


Fonte: de autoria própria.

4.2.6 Gradient Boosting para PD

O modelo é implementado através da classe *GradientBoostingClassifier*. Modelo sem *tuning*:

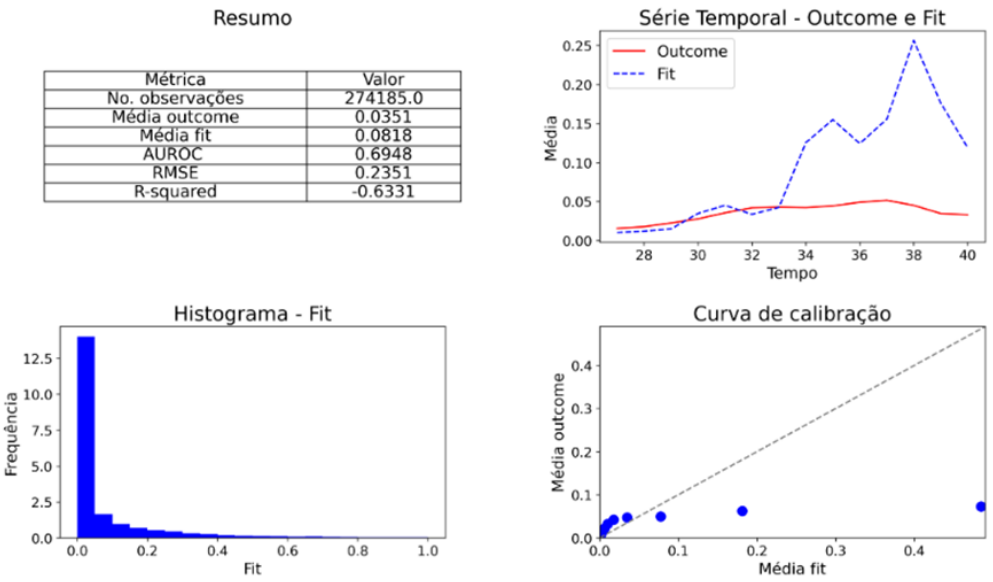
Figura 25 – Modelo Gradient Boosting de PD, sem *tuning* de hiperparâmetros.



Fonte: de autoria própria.

Modelo com *tuning,* com hiperparâmetros otimizados de taxa de aprendizagem, número de estimadores, profundidade máxima das árvores, número máximo de *features* consideradas a cada divisão, porcentagem de amostras para ajuste de cada estimador e critério de parada antecipada por não melhora da métrica de validação:

Figura 26 – Modelo Gradient Boosting de PD, com *tuning* de hiperparâmetros.

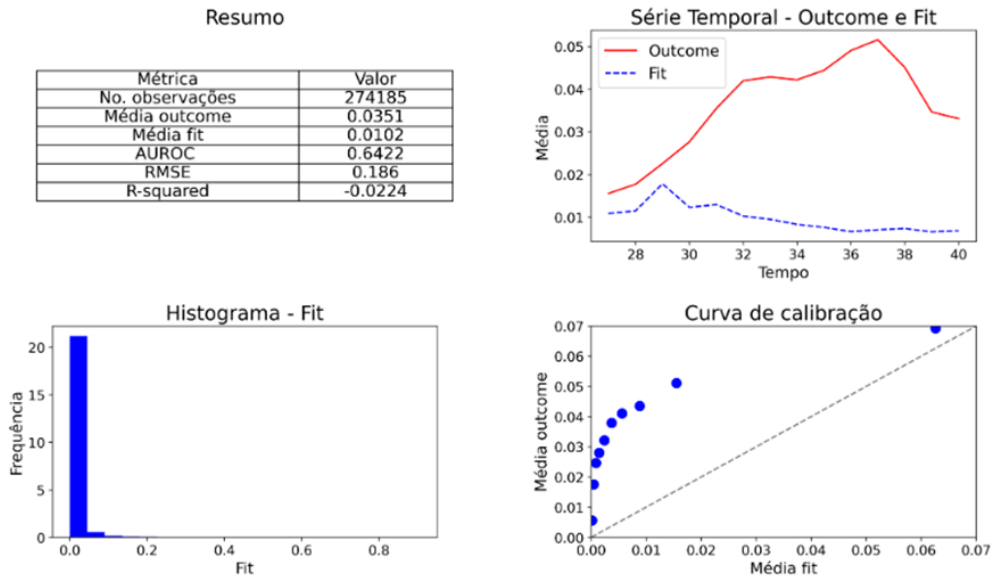


Fonte: de autoria própria.

4.2.7 XGBoost para PD

O modelo XGBoost é implementado através da classe *XGBClassifier* da biblioteca *xgboost.* Modelo sem *tuning* de hiperparâmetros:

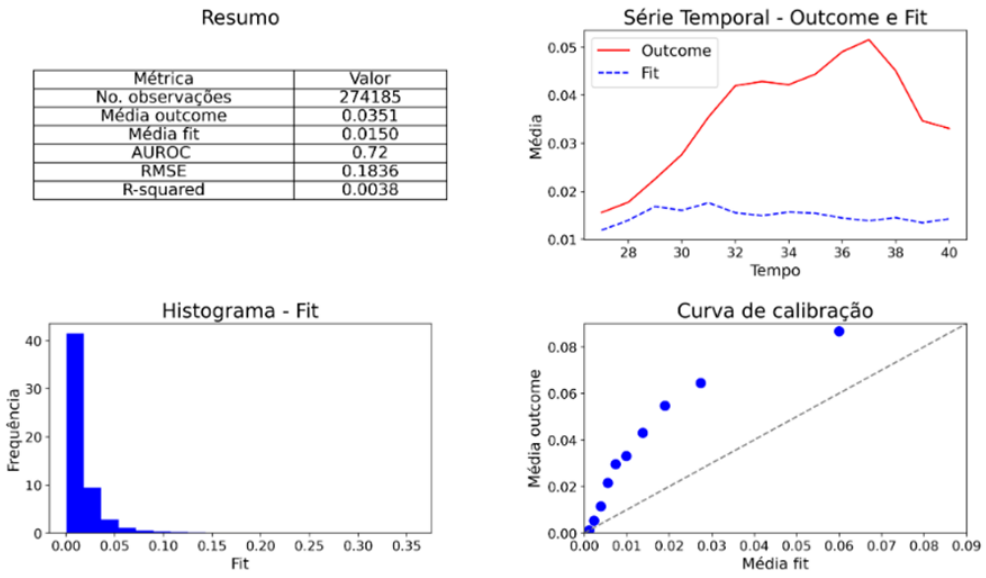
Figura 27 – Modelo XGBoost de PD, sem *tuning* de hiperparâmetros.



Fonte: de autoria própria.

Modelo com *tuning* de hiperparâmetros, considerando: profundidade máxima das árvores, taxa de aprendizagem, *gamma* (parâmetro relacionado à *pruning* das árvores), e regularização:

Figura 28 – Modelo XGBoost de PD, com *tuning* de hiperparâmetros.



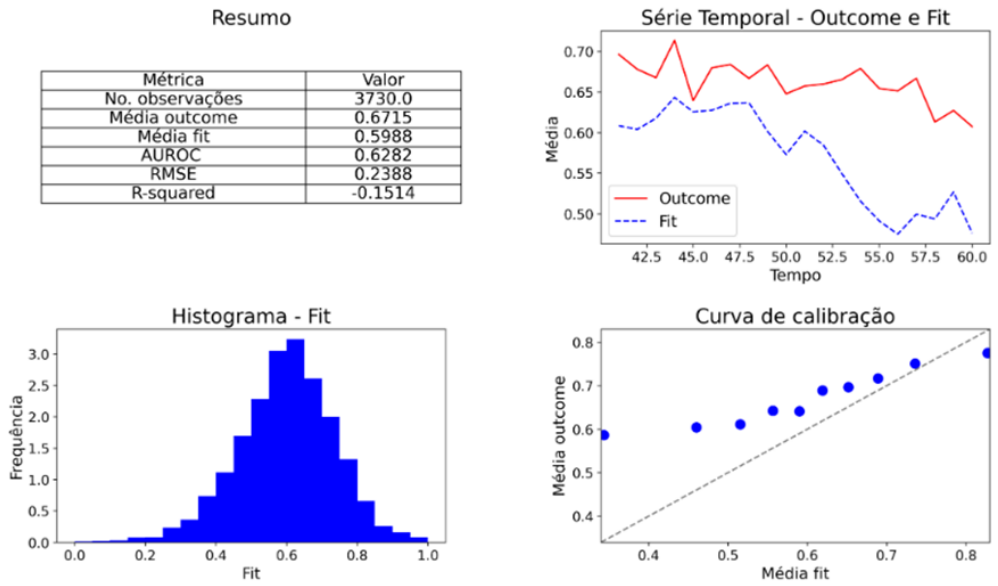
Fonte: de autoria própria.

4.3 Modelos LGD

4.3.1 Regressão linear para LGD

A regressão linear é implementada através da classe *LinearRegression.* Regressão linear sem regularização:

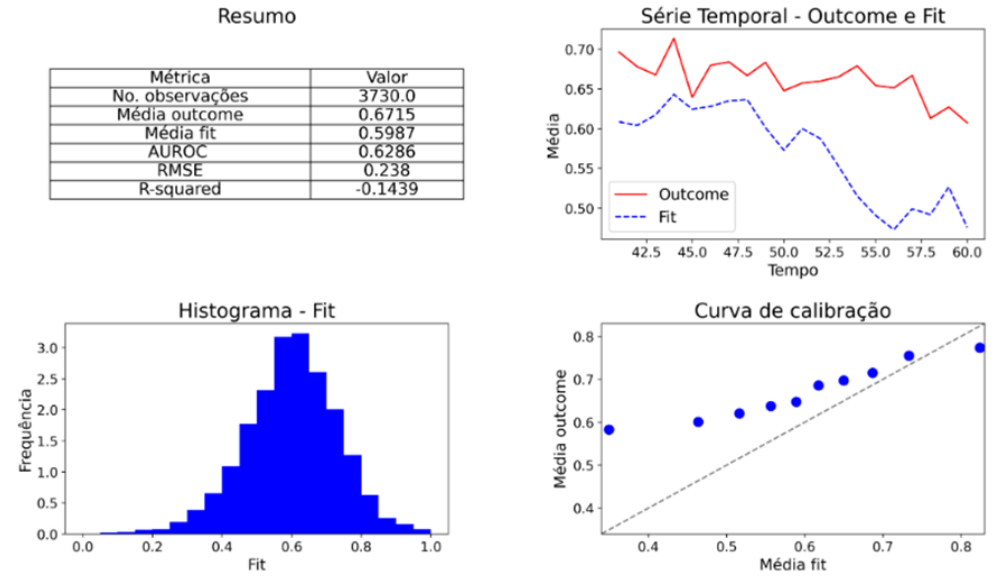
Figura 29 – Modelo Regressão linear de LGD, sem regularização.



Fonte: de autoria própria.

Regressão com regularização Ridge, com hiperparâmetro de regularização encontrado com *GridSearchCV:*

Figura 30 – Modelo Regressão linear de LGD, com regularização Ridge.



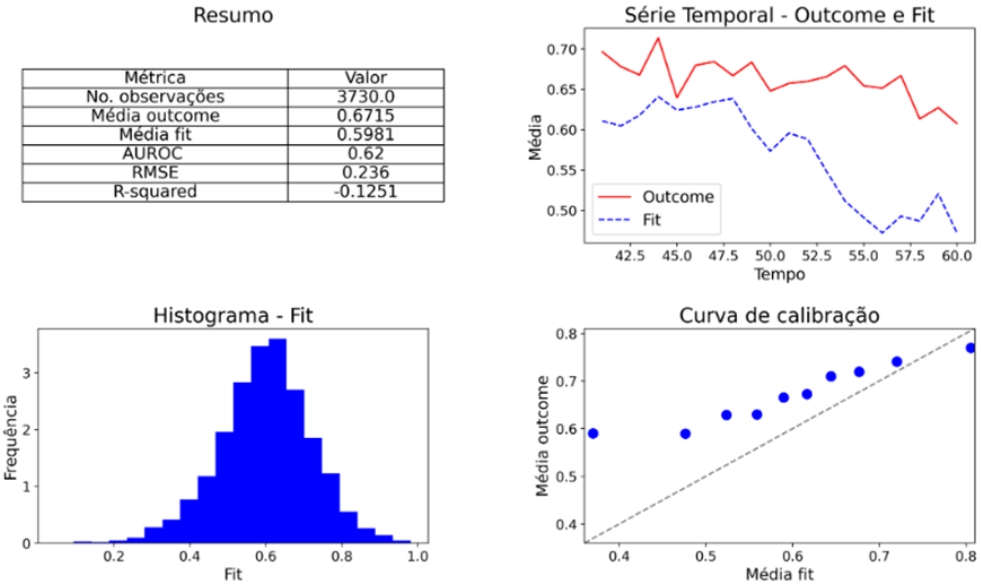
Fonte: de autoria própria.

Regressão LASSO, com hiperparâmetro de regularização encontrado com *GridSearchCV:*

Figura 31 – Modelo Regressão linear de LGD, com regularização LASSO.

Resumo

Métrica	Valor
No. observações	3730.0
Média outcome	0.6715
Média fit	0.5981
AUROC	0.62
RMSE	0.236
R-squared	-0.1251



Fonte: de autoria própria.

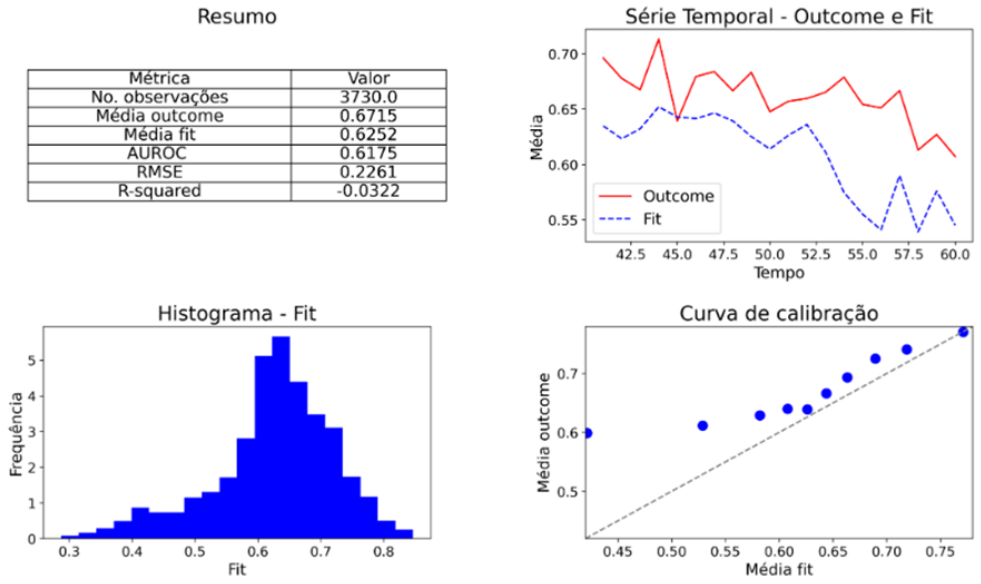
4.3.2 KNN para LGD

O algoritmo KNN é implementado pela classe *KNeighborsRegressor.* Modelo com $k = 100$:

Figura 32 – Modelo KNN de LGD, sem *tuning* de hiperparâmetros - *k = 100.*

Resumo

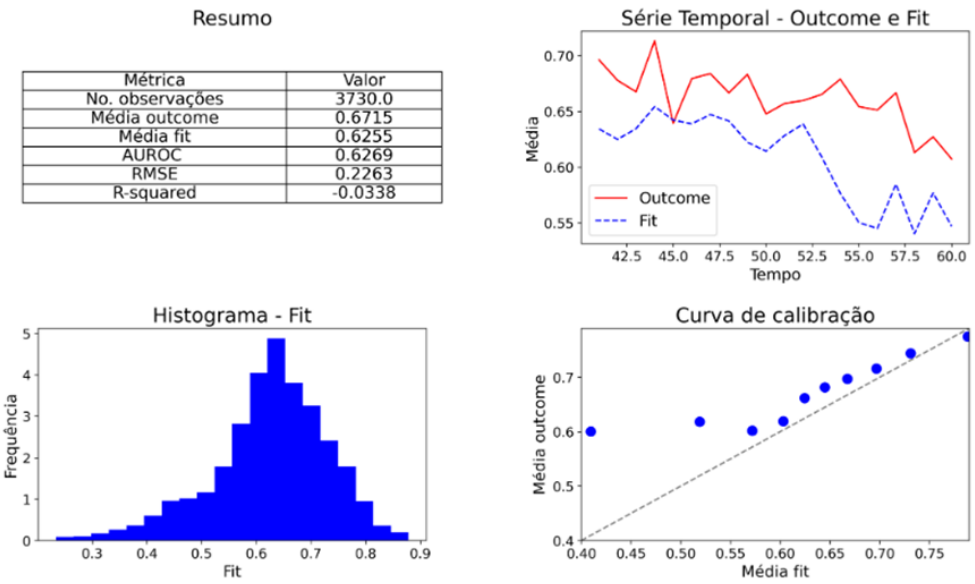
Métrica	Valor
No. observações	3730.0
Média outcome	0.6715
Média fit	0.6252
AUROC	0.6175
RMSE	0.2261
R-squared	-0.0322



Fonte: de autoria própria.

Modelo com $k = 48$, encontrado utilizando *GridSearchCV*:

Figura 33 – Modelo KNN de LGD, com *tuning* de hiperparâmetros - *k = 48.*

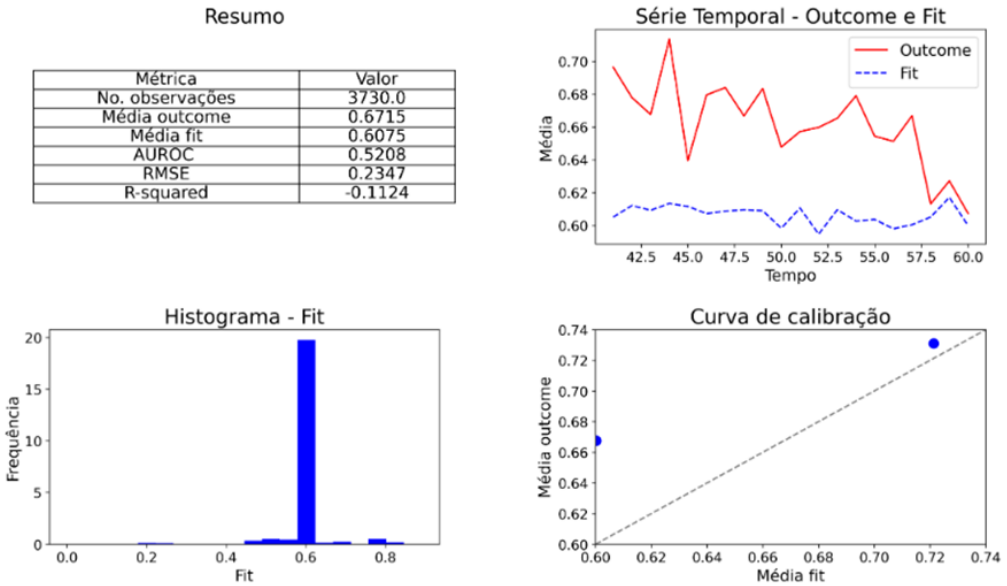


Fonte: de autoria própria.

4.3.3 Árvores de decisão para LGD

As árvores de classificação são implementadas através da classe *DecisionTreeRegressor.* Modelo sem tuning de hiperparâmetros:

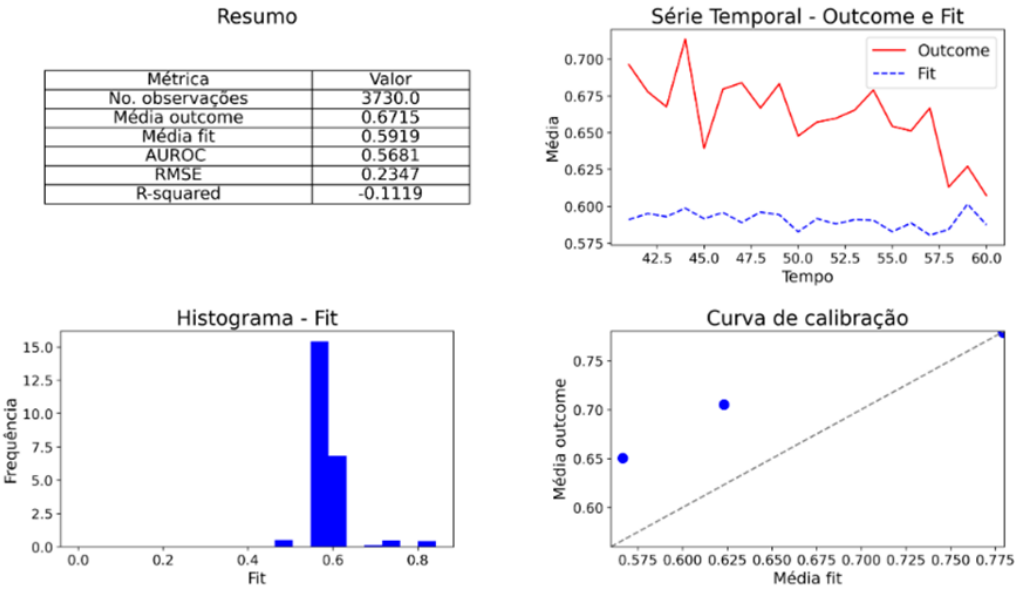
Figura 34 – Modelo Árvore de decisão de LGD, sem *tuning* de hiperparâmetros.



Fonte: de autoria própria.

Modelo com *tuning*, utilizando *RandomizedGridSearch* com 50 combinações para encontrar os hiperparâmetros otimizados de profundidade máxima da árvore, número máximo de *features* consideradas a cada divisão e porcentagem de amostras do nó necessárias para que seja feita uma nova divisão:

Figura 35 – Modelo Árvore de decisão de LGD, com *tuning* de hiperparâmetros.

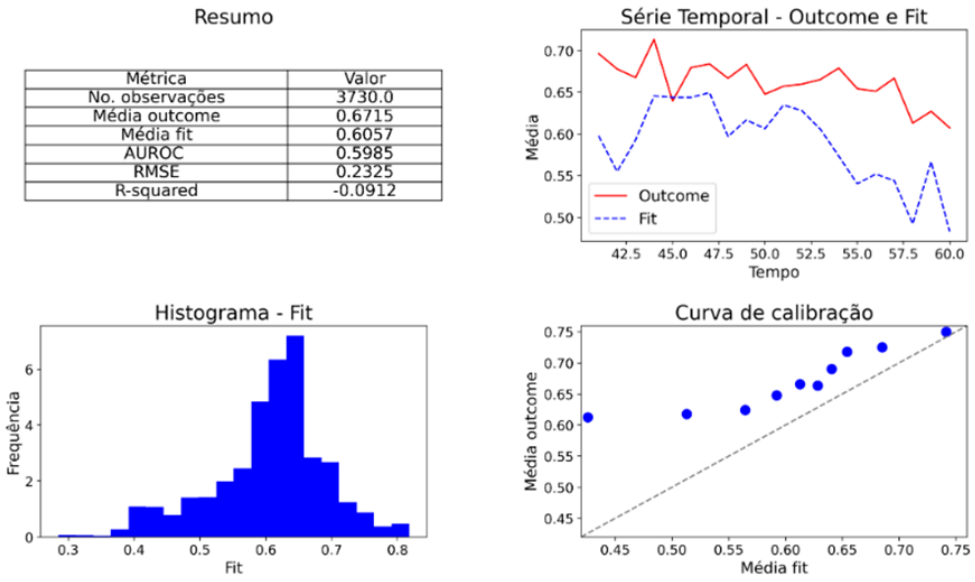


Fonte: de autoria própria.

4.3.4 Random forest para LGD

O estimador *random forest* é implementado através da classe *RandomForestRegressor*. Modelo sem *tuning*:

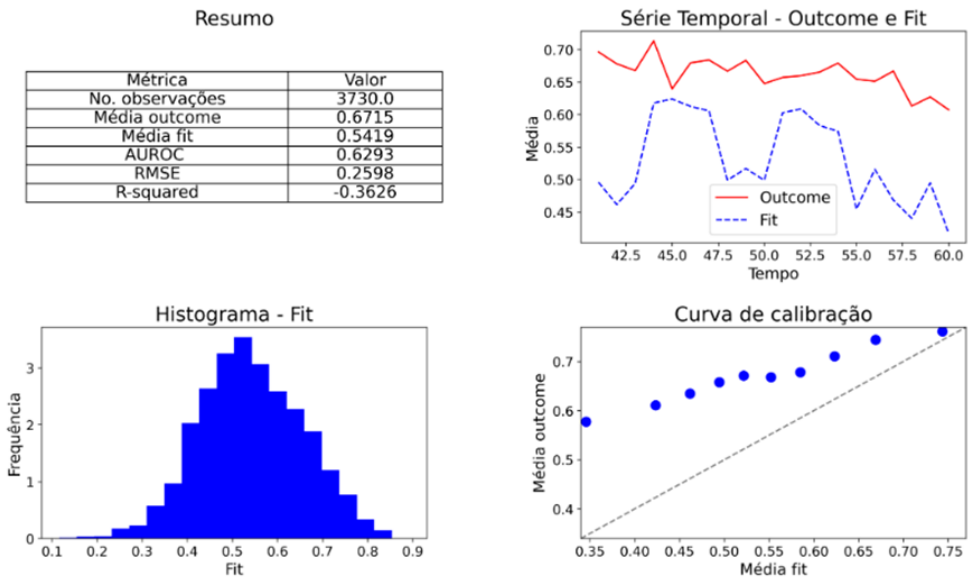
Figura 36 – Modelo Random forest de LGD, sem *tuning* de hiperparâmetros.



Fonte: de autoria própria.

Modelo com *tuning*, utilizando *RandomizedGridSearch* com 50 combinações para encontrar os hiperparâmetros otimizados de número de árvores, profundidade máxima das árvores, número máximo de *features* consideradas a cada divisão, porcentagem de amostras do nó necessárias para que seja feita uma nova divisão e se é utilizado o *bootstrapping* na construção das árvores:

Figura 37 – Modelo Random forest de LGD, com *tuning* de hiperparâmetros.

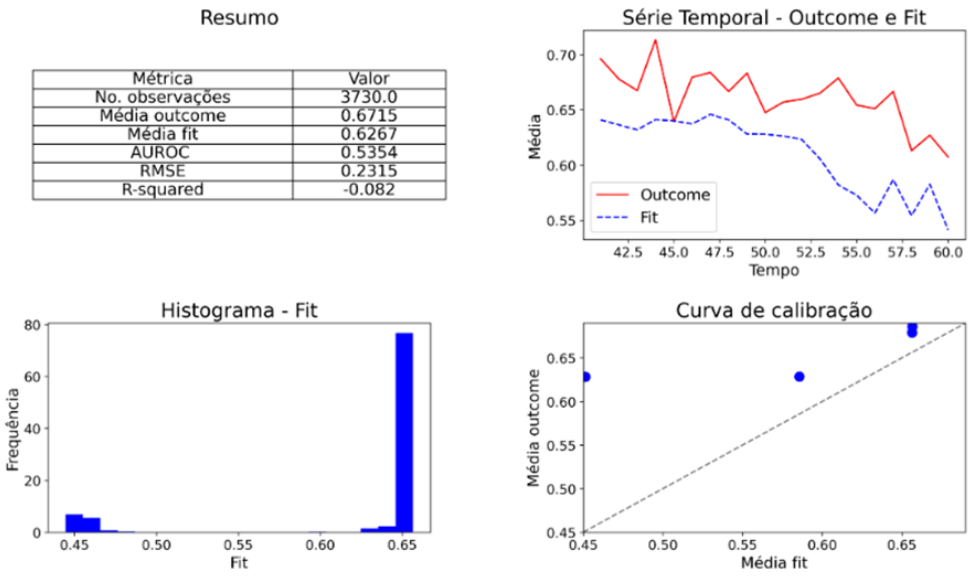


Fonte: de autoria própria.

4.3.5 Adaboost para LGD

O algoritmo AdaBoost é implementado por meio da classe *AdaBoostRegressor.* Modelo sem *tuning*:

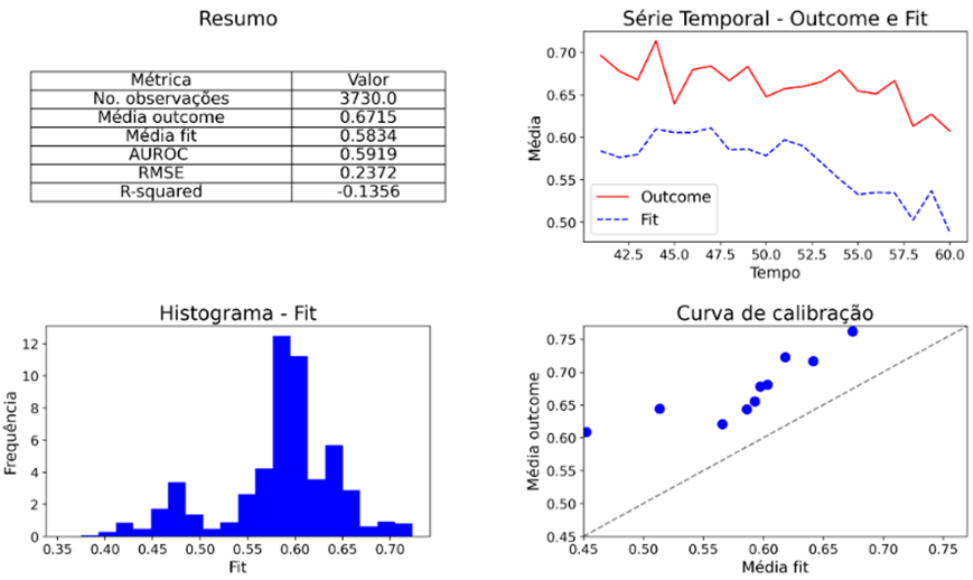
Figura 38 – Modelo Adaboost de LGD, sem *tuning* de hiperparâmetros.



Fonte: de autoria própria.

Modelo com *tuning,* com hiperparâmetros otimizados de taxa de aprendizagem e número de estimadores base encontrados com *GridSearchCV*:

Figura 39 – Modelo Adaboost de LGD, com *tuning* de hiperparâmetros.

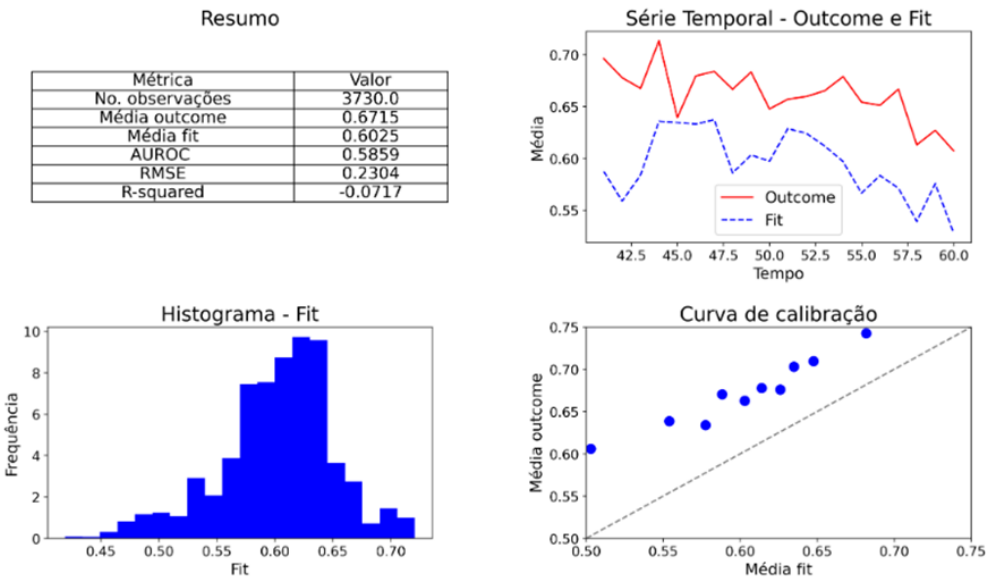


Fonte: de autoria própria.

4.3.6 Gradient Boosting para LGD

O modelo é implementado através da classe *GradientBoostingRegressor*. Modelo sem *tuning*:

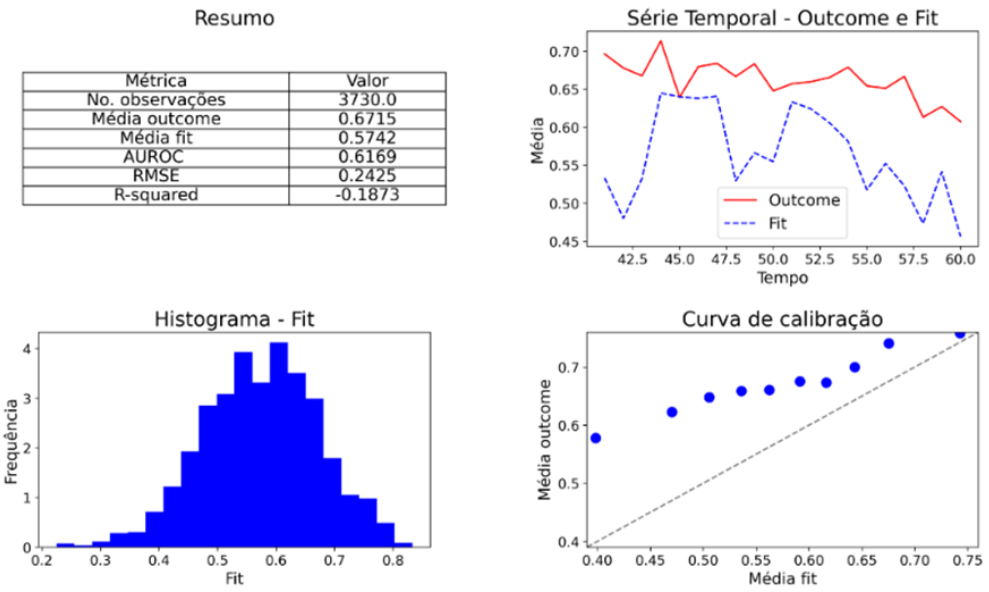
Figura 40 – Modelo Gradient Boosting de LGD, sem *tuning* de hiperparâmetros.



Fonte: de autoria própria.

Modelo com *tuning* dos hiperparâmetros: taxa de aprendizagem, número de estimadores, profundidade máxima das árvores, número máximo de *features* consideradas a cada divisão, porcentagem de amostras para ajuste de cada estimador e critério de parada antecipada por não melhora da métrica de validação:

Figura 41 – Modelo Gradient Boosting de LGD, com *tuning* de hiperparâmetros.

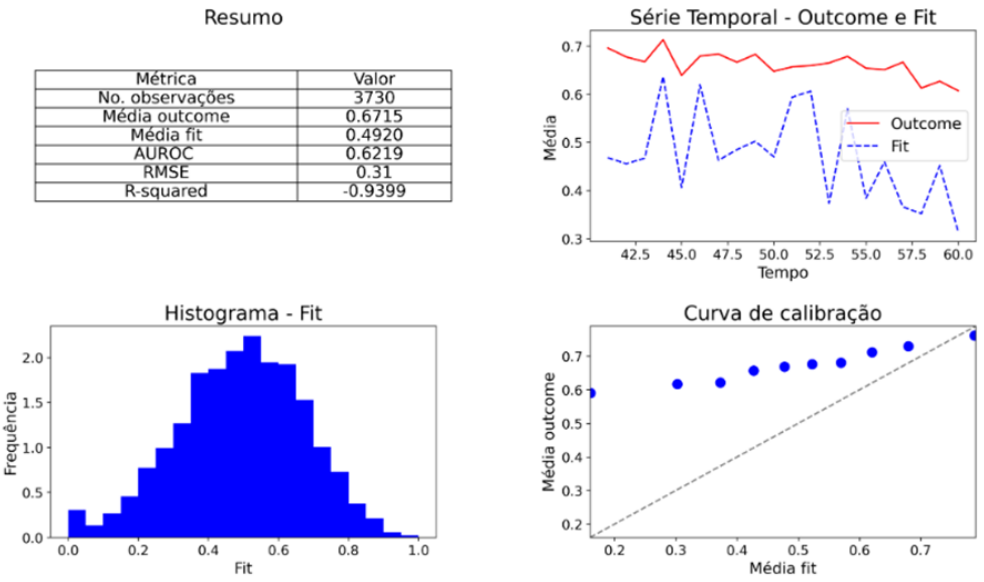


Fonte: de autoria própria.

4.3.7 XGBoost para LGD

O modelo XGBoost é implementado através da classe *XGBClassifier* da biblioteca *xgboost.* Modelo sem *tuning* de hiperparâmetros:

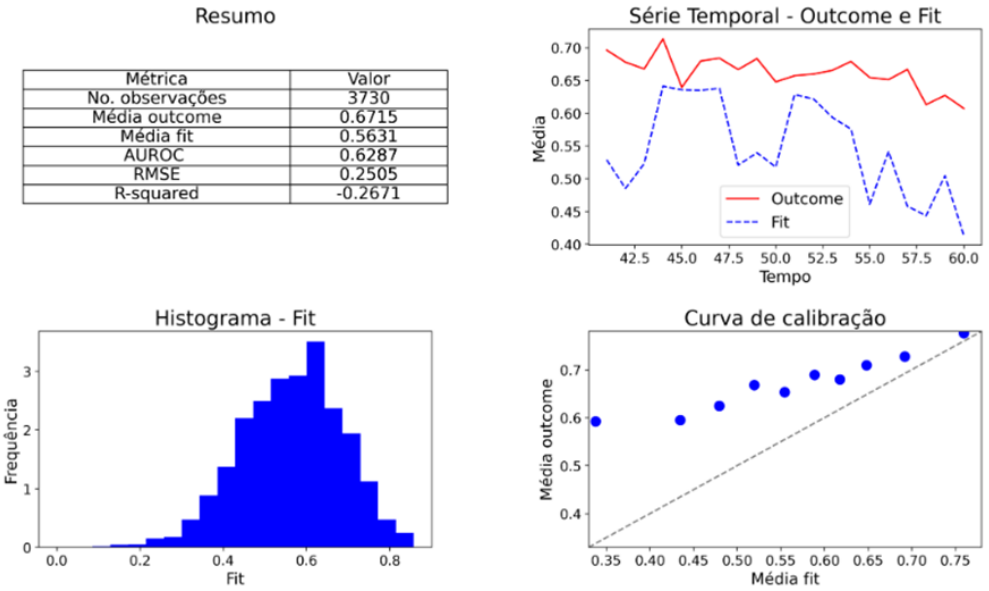
Figura 42 – Modelo XGBoost de LGD, sem *tuning* de hiperparâmetros.



Fonte: de autoria própria.

Modelo com *tuning* de hiperparâmetros, considerando: profundidade máxima das árvores, taxa de aprendizagem, *gamma* (parâmetro relacionado à *pruning* das árvores), e regularização:

Figura 43 – Modelo XGBoost de LGD, com *tuning* de hiperparâmetros.



Fonte: de autoria própria.

5 RESULTADOS

Sumarizando os resultados e analisando o AUROC dos modelos de PD, considerando a base completa (50.000 contratos), o melhor modelo foi o XGBoost com *tuning* de hiperparâmetros e o pior foi o modelo KNN com $k = 5$.

Tabela 3 – Resultados dos modelos de PD: AUROC

	5000 contratos	50000 contratos
Modelo	AUROC	AUROC
XGBoost com tuning	0,7224	0,7200
Random forest sem tuning	0,6921	0,7035
Gradient boosting sem tuning	0,6955	0,6981
Gradient boosting com tuning	0,6894	0,6948
Random forest com tuning	0,7000	0,6918
Adaboost com tuning	0,6820	0,6824
Decision tree com tuning	0,6552	0,6822
Regressão logística sem reg e tuning	0,6049	0,6587
Regressão logística com reg e tuning	0,6501	0,6587
Knn com tuning	0,6593	0,6575
Decision tree sem tuning	0,6392	0,6503
XGBoost sem tuning	0,6743	0,6422
Adaboost sem tuning	0,6435	0,6314
Knn (k=100)	0,6142	0,6233
Knn (k=5)	0,5623	0,5333

Fonte: de autoria própria.

Em relação aos modelos de LGD, o que apresentou melhor desempenho ao analisar a métrica RMSE foi o modelo KNN com $k = 100$ e o pior foi o modelo XGBoost.

Tabela 4 – Resultados dos modelos de LGD: RMSE

	5000 contratos	50000 contratos
Modelo	RMSE	RMSE
Knn (k=100)	0,227	0,2261
Knn com hiper. Tuning	0,224	0,2263
Gradient boosting	0,2282	0,2304
Adaboost	0,2363	0,2315
Random forest	0,2307	0,2325
Decision tree sem hiper. Tuning	0,2459	0,2347
Decision tree com hiper. Tuning	0,227	0,2347
Regressão linear Lasso	0,2352	0,236
Adaboost com hiper. Tuning	0,2313	0,2372
Regressão linear Ridge	0,2344	0,238
Regressão linear	0,2458	0,2388
Gradient boosting com hiper. Tuning	0,259	0,2425
XGBoost com hiper. Tuning	0,2364	0,2505
Random forest com hiper. Tuning	0,2382	0,2598
XGBoost	0,287	0,31

Fonte: de autoria própria.

Percebe-se que nem sempre utilizar a base completa proporcionou um resultado melhor (resultados piores que os da base parcial são destacados em vermelho nas tabelas), o que pode ser explicado em parte por um fator aleatório na seleção dos contratos para a base parcial (5.000 contratos), mesmo que ambas as bases tenham a mesma representatividade em termos de percentual de *default*.

O *tuning* de hiperparâmetros também não garante um AUROC melhor: os modelos Random Forest e Gradient Boosting sem *tuning* apresentaram melhores resultados. Uma explicação para isso é que a busca exaustiva pelos melhores hiperparâmetros utilizando validação cruzada considera apenas o conjunto de dados de treino, sendo o conjunto de dados de teste reservado para avaliar o desempenho do modelo.

Apesar de essa ser a metodologia correta, os dados de teste, da forma como foi feita a divisão, são dados relativos ao período da crise de 2007-2008 e apresentam características que não estão contidas no conjunto de dados de treino. Previsões para risco de crédito em geral apresentam desafios adicionais em relação às outras aplicações “tradicionais” de *machine learning* como reconhecimento de

imagens, em que queremos reconhecer coisas já vistas pelo modelo. Em risco de crédito, tipicamente mudanças estruturais e crises não estão contidas no conjunto de dados de treino, como destaca Rösch e Scheule (2020), o que torna mais difícil fazer previsões. Ainda assim, vários modelos apresentaram resultados superiores quando comparados às técnicas tradicionais como regressão, o que mostra que as técnicas de *machine learning* são muito úteis para esse tipo de previsão.

A análise de ordenação é uma alternativa para avaliação dos modelos. Nesse caso optou-se por comparar os modelos com *tuning* de hiperparâmetros e ajustados com a base completa. Para os modelos de PD a base é ordenada da maior probabilidade de *default* para a menor e são calculadas as porcentagens para cada ponto de corte: % não *default,* % não *default* acumulado, % *default* acumulado.

Tabela 5 – Ordenação de *score* para modelos de PD: % não default por faixa de corte.

Corte	% Não Default						
	Regressão Logística	KNN	Decision Tree	Random Forest	AdaBoost	Gradient Boost	XGBoost
1%	92,9%	92,8%	90,0%	92,1%	90,3%	90,1%	87,5%
5%	93,7%	93,4%	89,1%	91,5%	90,0%	91,3%	88,8%
10%	93,6%	93,5%	90,7%	91,7%	91,4%	92,1%	90,5%
20%	93,7%	93,9%	93,2%	92,5%	92,6%	92,7%	91,8%
30%	94,0%	94,0%	94,2%	93,0%	93,5%	93,4%	92,6%
40%	94,2%	94,3%	94,4%	93,6%	94,0%	93,8%	93,4%
50%	94,5%	94,8%	95,0%	94,1%	94,4%	94,2%	94,0%
60%	94,9%	94,9%	95,0%	94,6%	94,8%	94,6%	94,5%
70%	95,3%	95,2%	95,3%	95,1%	95,2%	95,1%	95,0%
80%	95,6%	95,6%	95,6%	95,5%	95,6%	95,5%	95,5%
90%	96,0%	96,0%	96,0%	96,0%	96,0%	96,0%	96,0%
95%	96,2%	96,2%	96,2%	96,2%	96,2%	96,2%	96,2%
99%	96,3%	96,3%	96,3%	96,3%	96,3%	96,3%	96,3%
100%	96,4%	96,4%	96,4%	96,4%	96,4%	96,4%	96,4%

Fonte: de autoria própria.

Tabela 6 – Ordenação de *score* para modelos de PD: % não default Acumulado por faixa de corte.

Corte	% Não Default Acumulado						
	Regressão Logística	KNN	Decision Tree	Random Forest	AdaBoost	Gradient Boost	XGBoost
1%	1,0%	1,0%	0,9%	1,0%	0,9%	0,9%	0,9%
5%	4,9%	4,9%	4,7%	4,8%	4,7%	4,8%	4,7%
10%	9,7%	9,7%	9,5%	9,6%	9,5%	9,6%	9,5%
20%	19,5%	19,5%	19,4%	19,3%	19,3%	19,3%	19,2%
30%	29,3%	29,3%	29,4%	29,0%	29,2%	29,2%	29,0%
40%	39,2%	39,2%	39,3%	39,0%	39,1%	39,0%	38,9%
50%	49,1%	49,2%	49,3%	48,9%	49,1%	49,0%	48,9%
60%	59,2%	59,2%	59,2%	59,0%	59,1%	59,0%	59,0%
70%	69,3%	69,2%	69,3%	69,1%	69,3%	69,1%	69,1%
80%	79,4%	79,4%	79,4%	79,4%	79,4%	79,4%	79,3%
90%	89,7%	89,7%	89,7%	89,7%	89,7%	89,7%	89,6%
95%	94,8%	94,8%	94,8%	94,8%	94,8%	94,8%	94,8%
99%	99,0%	99,0%	99,0%	99,0%	99,0%	99,0%	99,0%
100%	100,0%	100,0%	100,0%	100,0%	100,0%	100,0%	100,0%

Fonte: de autoria própria.

Tabela 7 – Ordenação de *score* para modelos de PD: % default acumulado por faixa de corte.

Corte	% Default Acumulado						
	Regressão Logística	KNN	Decision Tree	Random Forest	AdaBoost	Gradient Boost	XGBoost
1%	1,9%	1,9%	2,6%	2,1%	2,5%	2,6%	3,2%
5%	8,5%	8,8%	14,1%	11,1%	13,0%	11,4%	14,3%
10%	17,1%	17,5%	24,3%	21,8%	22,6%	20,8%	24,7%
20%	33,8%	32,9%	36,3%	39,8%	39,5%	38,6%	43,1%
30%	48,1%	48,6%	46,8%	56,2%	52,6%	52,7%	58,6%
40%	62,1%	61,2%	60,6%	68,4%	64,6%	66,3%	70,9%
50%	73,7%	70,7%	68,3%	79,9%	75,1%	78,4%	80,3%
60%	82,9%	82,7%	81,7%	87,5%	84,2%	87,6%	88,8%
70%	90,2%	90,7%	90,4%	93,8%	90,6%	93,8%	94,9%
80%	95,2%	96,6%	96,0%	97,8%	95,5%	97,8%	98,2%
90%	98,6%	99,2%	99,1%	99,5%	98,5%	99,5%	99,7%
95%	99,6%	99,8%	99,7%	99,9%	99,4%	99,8%	99,9%
99%	100,0%	100,0%	100,0%	100,0%	100,0%	100,0%	100,0%
100%	100,0%	100,0%	100,0%	100,0%	100,0%	100,0%	100,0%

Fonte: de autoria própria.

O melhor resultado foi o do modelo XGBoost, identificando 70,9% de todos os *defaults* da base para o ponto de corte 40%, 8,8% a mais que o modelo de regressão logística. Todos os modelos *ensemble* apresentaram resultado superior que o modelo de regressão logística segundo essas métricas, e os modelos KNN e Árvores de Decisão por sua vez apresentaram resultados piores.

Para os modelos de LGD a análise de ordenação de *score* também foi feita comparando-se os modelos com *tuning* de hiperparâmetros e ajustados com a base

completa. Nessa avaliação a base é ordenada da menor perda estimada para a maior perda estimada e são calculadas para cada ponto de corte a recuperação em valores monetários e porcentagem do total.

Tabela 8 – Ordenação de *score* para modelos de LGD: recuperação em $MM por faixa de corte.

Corte	Recuperação ($ MM)						
	Regressão Linear	KNN	Decision Tree	Random Forest	AdaBoost	Gradient Boost	XGBoost
1%	2,0	4,1	3,4	3,4	4,3	3,5	3,4
5%	18,0	17,2	22,5	19,6	17,2	18,7	18,5
10%	35,5	33,3	36,8	41,7	32,8	39,6	42,3
20%	80,2	73,4	76,6	88,0	67,9	81,7	93,4
30%	124,8	123,2	102,7	127,7	115,3	123,4	134,6
40%	167,9	167,6	138,3	168,0	157,5	159,8	170,2
50%	205,0	212,2	184,3	203,5	194,7	198,0	206,8
60%	240,9	248,6	239,3	237,2	235,9	231,5	242,1
70%	271,7	279,5	279,4	271,0	274,7	270,4	276,2
80%	299,5	304,5	300,2	298,6	300,3	299,9	301,7
90%	323,9	326,3	323,9	321,6	322,6	323,6	325,3
95%	334,5	334,9	334,1	332,7	331,1	333,8	335,4
99%	341,2	340,8	340,4	341,1	340,2	341,4	341,2
100%	342,2	342,2	342,2	342,2	342,2	342,2	342,2

Fonte: de autoria própria.

Tabela 9 – Ordenação de *score* para modelos de LGD: recuperação em % do total por faixa de corte.

Corte	Recuperação (%)						
	Regressão Linear	KNN	Decision Tree	Random Forest	AdaBoost	Gradient Boost	XGBoost
1%	0,6%	1,2%	1,0%	1,0%	1,3%	1,0%	1,0%
5%	5,3%	5,0%	6,6%	5,7%	5,0%	5,5%	5,4%
10%	10,4%	9,7%	10,8%	12,2%	9,6%	11,6%	12,4%
20%	23,4%	21,4%	22,4%	25,7%	19,8%	23,9%	27,3%
30%	36,5%	36,0%	30,0%	37,3%	33,7%	36,1%	39,3%
40%	49,1%	49,0%	40,4%	49,1%	46,0%	46,7%	49,7%
50%	59,9%	62,0%	53,9%	59,5%	56,9%	57,9%	60,4%
60%	70,4%	72,7%	69,9%	69,3%	68,9%	67,6%	70,7%
70%	79,4%	81,7%	81,7%	79,2%	80,3%	79,0%	80,7%
80%	87,5%	89,0%	87,7%	87,3%	87,8%	87,7%	88,2%
90%	94,7%	95,4%	94,7%	94,0%	94,3%	94,6%	95,1%
95%	97,8%	97,9%	97,7%	97,2%	96,8%	97,6%	98,0%
99%	99,7%	99,6%	99,5%	99,7%	99,4%	99,8%	99,7%
100%	100,0%	100,0%	100,0%	100,0%	100,0%	100,0%	100,0%

Fonte: de autoria própria.

O melhor resultado foi o do modelo XGBoost. com uma recuperação estimada de $93,4 MM para o ponto de corte 20%, um ganho de $13,2 MM em relação à

regressão linear. Todos os modelos *ensemble* com exceção do modelo AdaBoost apresentaram resultados melhores que os da regressão linear, e os modelos KNN e Árvores de Decisão por sua vez apresentaram resultados piores.

6 CONCLUSÃO

O objetivo principal do trabalho foi criar modelos de predição de *default* e perdas de crédito utilizando técnicas de *machine learning* aplicadas a uma base real de dados, pois além da importância da utilização de modelos mais precisos para controle de riscos sistêmicos no sistema financeiro, o uso dos mesmos pode ser um diferencial competitivo para as empresas, na medida em que permite conhecer melhor o comportamento dos seus clientes e reduzir custos.

Foram exploradas várias técnicas: Regressão, K-vizinhos mais próximos, Árvores de Decisão, Random Forest, Adaboost, Gradient Boosting e XGBoost. Modelos mais complexos como o XGBoost provaram-se úteis para fazer previsões mais precisas em relação a modelos mais tradicionais como os de Regressão, identificando com maior acurácia os contratos com *default* e estimando mais precisamente as perdas dos mesmos. Através da análise de ordenação de *score* dos modelos de LGD, propôs-se como aplicação prática uma possível ação de recuperação, na qual os modelos *ensemble* apresentaram um ganho financeiro em relação ao modelo de Regressão Linear.

Por fim, a melhora no poder preditivo ao utilizar modelos mais complexos vem ao custo da interpretabilidade dos mesmos. Essa relação inversamente proporcional entre acurácia e interpretabilidade pode ser um problema por causa da necessidade de cumprir regulações e controles internos, nos quais os tomadores de decisão que utilizam os modelos precisam interpretá-los em certo nível.

Em trabalhos futuros, seria interessante explorar mais a questão da LGPD e tratamento de dados no Brasil e criar modelos utilizando dados reais do mercado brasileiro, visto que existe uma pequena quantidade de trabalhos publicados sobre esses temas. Sugere-se também a utilização de outras técnicas de *machine learning*, a utilização de outras técnicas de *tuning* diferentes da busca exaustiva e explorar técnicas de seleção de *features* como o método *stepwise.*

REFERÊNCIAS

FORTI, M. Técnicas de machine learning aplicadas na recuperação de crédito do mercado brasileiro (Dissertação de Mestrado em Economia). Escola de Economia de São Paulo da Fundação Getúlio Vargas, 2018.

SERASA. Mapa da inadimplência e renegociação de dívidas no Brasil. Disponível em <https://www.serasa.com.br/limpa-nome-online/blog/mapa-da-inadimplencia-e-renogociacao-de-dividas-no-brasil/>. Acesso em: nov. de 2022.

LEE, I. e SHIN, Y. J. Machine learning for enterprises: Applications, algorithm selection, and challenges. Business Horizons, v. 63, n. 2, p. 157-170, 2020.

SCHWERTER, S. Basel III's ability to mitigate systemic risk. Journal of financial regulation and compliance, 2011.

BRASIL. Lei nº 13.709, de 14 de agosto de 2018. Lei Geral de Proteção de Dados Pessoais (LGPD). Disponível em: <https://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/l13709.htm>. Acesso em novembro de 2022.

ROCHER, L; HENDRICKX, J. M.; DE MONTJOYE, Y. Estimating the success of re-identifications in incomplete datasets using generative models. Nature communications, v. 10, n. 1, p. 1-9, 2019.

ANPD. A LGPD e o tratamento de dados pessoais para fins acadêmicos e para a realização de estudos por órgãos de pesquisa. Estudo técnico. Disponível em: <https://www.gov.br/anpd/pt-br/documentos-e-publicacoes/sei_00261-000810_2022_17.pdf>. Acesso em junho de 2022.

HASTIE T. et al. The Elements of Statistical Learning: Data Mining, Inference, and Prediction . 2. ed. New York City: Springer, 2017.

OLIVEIRA, E. L. Técnicas de aprendizado de máquina aplicadas na previsão de desempenho de operadores de centros de teleatendimento (Tese de Doutorado).. Universidade Fernando Pessoa, 2021.

FREUND, Y e SCHAPIRE, R. E. A decision-theoretic generalization of on-line learning and an application to boosting. Journal of computer and system sciences, v. 55, n. 1, p. 119-139, 1997.

DRUCKER, H. Improving regressors using boosting techniques. Proc. of the 14th Int. Conf. on Machine Learning, . p. 107–115, 1997.

FRIEDMAN, J. H. Greedy function approximation: a gradient boosting machine. Annals of statistics, p. 1189-1232, 2001.

CHEN, T. e GUESTRIN, C. Xgboost: A scalable tree boosting system. In: Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, p. 785-794, 2016.

GITHUB. Machine learning winning solutions. Disponível em <https://github.com/dmlc/xgboost/tree/master/demo#machine-learning-challenge-winning-solutions>. Acesso em: outubro de 2022.

PELLICER, L. F. e PAIT, F. M. BarySearch: Algoritmo de tuning de Modelos de Machine Learning com o Método do Baricentro. 8th Brazilian Conference on Intelligent Systems (BRACIS), p.99, 2019.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. the Journal of machine Learning research, v. 12, p. 2825-2830, 2011.

RÖSCH, D e SCHEULE, H. Deep credit risk: Machine learning with python. Fulda: Independently published, 2020

INTERNATIONAL FINANCIAL RESEARCH. International Financial Research. Página inicial. Disponível em: <http://www.internationalfinancialresearch.org/>. Acesso em 07 de nov. de 2022.

APÊNDICE - HIPERPARÂMETROS

Modelo	Hiperparâmetro	Descrição e valores otimizados
Regressão logística	Método de regularização	Adiciona um termo de regularização. Os termos de penalidade possíveis são L1, L2 ou ambos. O melhor hiperparâmetro encontrado foi L2.
	Inverso da força da regularização	Quanto menor for o valor, mais penalizados serão os coeficientes e menor a chance de *overfitting* dos dados. Os valores variam de 0 ao infinito. Foi encontrado um valor ótimo de 100.
	Número máximo de iterações	Número máximo de iterações realizadas para os *solvers* convergirem. O valor encontrado foi de 100.
Regressão linear Ridge/LASSO	Força da regularização	Quanto maior for o valor, mais penalizados serão os coeficientes e menor a chance de *overfitting* dos dados. Para o modelo Ridge foi encontrado um valor igual a 100 e para LASSO um valor de 50.
	Número máximo de iterações	Número máximo de iterações realizadas para os *solvers* convergirem. Para ambos o valor ótimo do hiperparâmetro foi de 100.
KNN	Número de vizinhos	O número de vizinhos mais próximos considerados pelo algoritmo. Para o modelo PD foi encontrado k=2048 e para LGD k=48.
DT	Profundidade máxima da árvore	Se o parâmetro é vazio, então os nós são expandidos até todas as folhas serem puras ou conterem menos amostras que o parâmetro "mínimo de amostras por divisão". Para o modelo PD foi encontrado o valor 5 e para LGD o valor 4.
	Número mínimo de amostras por divisão	Número mínimo de amostras do nó necessárias para que seja feita uma nova divisão. Pode ser um número inteiro ou uma porcentagem do total de amostras. Para ambos os modelos foi encontrado o valor 50.
	Número máximo de *features* a cada divisão	Se o parâmetro é vazio, então a cada nó todas as *features* são consideradas para fazer a melhor divisão. Para o modelo PD foi encontrado o valor 11 e para o modelo LGD foi encontrado o valor 6.
RF	Número de árvores	Número de árvores ajustadas pelo modelo. O valor padrão é 100. O valor ótimo para o modelo PD foi de 371 e para o modelo LGD foi de 326.
	Profundidade máxima das árvores	Se o parâmetro é vazio, então os nós são expandidos até todas as folhas serem puras ou conterem menos amostras que o parâmetro "mínimo de amostras por divisão". Para o modelo PD foi encontrado o valor 3 e para LGD o valor *None*.
	Número mínimo de amostras por divisão	Número mínimo de amostras do nó necessárias para que seja feita uma nova divisão. Pode ser um número inteiro ou uma porcentagem do total de amostras. Para ambos os modelos foi encontrado o valor 10%.
	Número máximo de *features* a cada divisão	Se o parâmetro é vazio, então a cada nó todas as *features* são consideradas para fazer a melhor divisão. Para o modelo PD foi encontrado o valor 8 e para o modelo LGD foi encontrado o valor 2.
	Bootstrapping	Utilizar ou não o método *bootstrapping* na construção das árvores. As buscas retornaram o valor *True* para ambos os modelos.

ADA	Número de estimadores	Número de estimadores para o *boosting*. O valor padrão é 100. Para PD foi encontrado o valor de 50 e para LGD o valor de 100.
	Taxa de aprendizagem	Peso aplicado a cada estimador a cada iteração. Um valor maior acarreta um aumento da contribuição de cada estimador ao estimador final. Os valores variam de 0 ao infinito. Para ambos os modelos foi encontrado o valor de 0,1.
GB	Número de estimadores	Número de estimadores para o *boosting*. O valor padrão é 100. Para ambos os modelos foi encontrado o valor 500.
	Taxa de aprendizagem	Uma taxa de aprendizagem maior aumenta a contribuição de cada árvore. Existe um *trade-off* entre a taxa de aprendizagem e o número de estimadores. Para o modelo PD foi encontrado o valor e 0,1 e para o modelo LGD foi encontrado o valor de 0,02.
	Profundidade máxima das árvores	O valor padrão é 3. Para ambos os modelos foi encontrado o valor de 5.
	Número máximo de *features* a cada divisão	Se o parâmetro é vazio, então a cada nó todas as *features* são consideradas para fazer a melhor divisão. Para o modelo PD foi encontrado o valor 4 e para o modelo LGD foi encontrado o valor 3.
	Subamostra	Porcentagem de amostras utilizadas para ajuste de cada estimador base. Valores menores do que 1.0 resultam em uma dimunição da variância do modelo e ele é chamado de *Stochastic Gradient Boosting*. Para o modelo PD o valor encontrado foi 1,0 e para o modelo LGD o valor encontrado foi 0,8.
	Critério de parada antecipada	Hiperparâmetro de critério de parada antecipada se a função ação não apresenta melhora após *n* iterações. Para o modelo PD o valor encontrado foi *n*=20 e para o modelo LGD o valor encontrado foi *n*=50.
XGBoost	Taxa de aprendizagem	Hiperparâmetro análogo à taxa de aprendizagem de GB. Faz com que o modelo seja mais robusto ao diminuir os pesos a cada iteração. Para ambos os modelos foi encontrado o valor de 0,1.
	Profundidade máxima das árvores	O valor padrão é 6. Para ambos os modelos foi encontrado o valor de 3.
	Mínima redução de perda para divisão	Hiperparâmetro de critério de mínima redução necessária da função de perda para nova divisão em um nó. O parâmetro é conhecido como *gamma* e quanto maior for seu valor, mais conservativo é o algoritmo. Para ambos os modelos foi encontrado o valor 0.
	Força da regularização	Adiciona um termo de regularização L2. O hiperparâmetro é conhecido como *lambda*. Para o modelo PD o valor encontrado foi 10 e para o modelo LGD o valor encontrado foi 0.