UNIVERSIDADE FEDERAL DE SÃO CARLOS
CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA
BACHARELADO EM ENGENHARIA FÍSICA

GABRIEL DE ALBUQUERQUE

ANÁLISE PREDITIVA DOS SPREADS DE CRÉDITO NO MERCADO DE DEBÊNTURES DO IDEX-CDI JGP POR MEIO DE MACHINE LEARNING

SÃO CARLOS

2025

UNIVERSIDADE FEDERAL DE SÃO CARLOS
CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA
BACHARELADO EM ENGENHARIA FÍSICA

GABRIEL DE ALBUQUERQUE

ANÁLISE PREDITIVA DOS SPREADS DE CRÉDITO NO MERCADO DE DEBÊNTURES DO IDEX-CDI JGP POR MEIO DE MACHINE LEARNING

Trabalho Final de Curso apresentado ao Centro de Ciências Exatas e de Tecnologia da Universidade Federal de São Carlos, para obtenção do título/grau de bacharel em Engenharia Física.
Orientação: Prof. Dr. Claudio Antonio Cardoso

SÃO CARLOS
2025

de Albuquerque, Gabriel

Análise preditiva dos spreads de crédito no mercado de debêntures do Idex-CDI JGP por meio de machine learning / Gabriel de Albuquerque -- 2025.
131f.

TCC (Graduação) - Universidade Federal de São Carlos, campus São Carlos, São Carlos
Orientador (a): Claudio Antonio Cardoso
Banca Examinadora: Fabio Luis Zabotto, Leonardo José Dalla Costa
Bibliografia

1. Machine learning. 2. Análise preditiva. 3. Spread de crédito. I. de Albuquerque, Gabriel. II. Título.

Ficha catalográfica desenvolvida pela Secretaria Geral de Informática (SIn)

DADOS FORNECIDOS PELO AUTOR

Bibliotecário responsável: Arildo Martins - CRB/8 7180

UNIVERSIDADE FEDERAL DE SÃO CARLOS
CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA
BACHARELADO EM ENGENHARIA FÍSICA

GABRIEL DE ALBUQUERQUE

ANÁLISE PREDITIVA DOS SPREADS DE CRÉDITO NO MERCADO DE DEBÊNTURES DO IDEX-CDI JGP POR MEIO DE MACHINE LEARNING

Trabalho Final de Curso apresentado ao Centro de Ciências Exatas e de Tecnologia da Universidade Federal de São Carlos, para obtenção do título/grau de bacharel em Engenharia Física.
Orientação: Prof. Dr. Claudio Antonio Cardoso

SÃO CARLOS, 01 de dezembro de 2025:

Prof. Dr. Claudio Antonio Cardoso

Prof. Dr. Fabio Luis Zabotto

Prof. Dr. Leonardo José Dalla Costa

SÃO CARLOS
2025

Dedico este trabalho à minha família, fonte de força e inspiração;
à verdade, que guia o pensamento;
e à vida, que traz boas novas a cada passo.

"Non quia difficilia sunt non audemus, sed quia non audemus difficilia sunt."
Não é porque as coisas são difíceis que não ousamos,
é porque não ousamos que elas são difíceis.
— Sêneca

Resumo

O mercado de crédito privado brasileiro tem ganhado crescente relevância como fonte alternativa de financiamento corporativo, impulsionado pelo aumento das emissões de debêntures e pela atuação dos fundos de investimento em renda fixa. Nesse contexto, compreender a dinâmica dos spreads de crédito e sua relação com os fluxos de captação líquida torna-se fundamental para a gestão de portfólios. Este trabalho analisa essa relação utilizando o Idex–CDI JGP, que é um índice representativo de debêntures líquidas e indexadas ao CDI como referência para mensurar o comportamento dos spreads. A metodologia combina técnicas econométricas e de machine learning, iniciando pela análise de correlação entre spreads e captação líquida com diferentes defasagens e séries dessazonalizadas. Observou-se uma correlação negativa moderada no curto prazo, coerente com maior aversão ao risco, mas que se fortalece expressivamente após a dessazonalização, atingindo Pearson acima de 0,82 em torno de dez meses de defasagem. O modelo de Mínimos Quadrados Ponderados (WLS) apresentou um ajuste ótimo com R^2 acima de 0,90 e mostrou que o spread defasado é um forte determinante da captação líquida. Para prever os spreads, foram testados modelos univariados (apenas spread) e multivariados (spread e captação líquida) em um pipeline de três etapas: um modelo autorregressivo com Elastic Net, um modelo de correção e uma etapa final de previsão por horizonte com HistGradientBoostingRegressor, seleção por Spearman, clipping e calibração. O modelo univariado teve desempenho alto no curtíssimo prazo, mas perdeu precisão em prazos maiores. Já a inclusão das variáveis de fluxo trouxe poucos ganhos no curto prazo, mas melhorou a previsão para horizontes maiores indicando que a captação líquida ajuda a identificar fases do ciclo de crédito.

Palavras-chave: Aprendizado de máquina. Análise preditiva. Séries temporais. Elastic Net. Spread de crédito. Histogram-based Gradient Boosting Regression Tree.

Abstract

The Brazilian private credit market has gained increasing relevance as an alternative source of corporate financing, driven by the rise in debenture issuances and the growing role of fixed-income investment funds. In this context, understanding the dynamics of credit spreads and their relationship with net fund flows becomes essential for portfolio management. This study analyzes this relationship using the Idex–CDI JGP, a representative index of liquid debentures indexed to the CDI, as a benchmark for measuring spread behavior. The methodology combines econometric and machine learning techniques, beginning with a correlation analysis between spreads and net fund flows using different lags and deseasonalized series. A moderate negative short-term correlation was observed, consistent with higher risk aversion, but it strengthens significantly after deseasonalization, reaching a Pearson coefficient above 0.82 at around a ten-month lag. The Weighted Least Squares (WLS) model delivered an excellent fit, with an R^2 above 0.90, and showed that the lagged spread is a strong determinant of net fund flows. To forecast the spreads, both univariate models (spread only) and multivariate models (spread and net fund flows) were tested within a three-stage pipeline: an autoregressive model with Elastic Net, a correction step, and a final horizon-based forecasting stage using the HistGradientBoostingRegressor, along with Spearman selection, clipping, and calibration. The univariate model performed strongly in the very short term but lost accuracy over longer horizons. The incorporation of flow-related variables produced limited gains at short horizons, but it improved predictive accuracy over longer forecasting windows, indicating that net fund flows contribute to identifying different phases of the credit cycle.

Keywords: Machine learning. Predictive analysis. Time series. Elastic Net. Credit spread.

Lista de ilustrações

Lista de quadros

Lista de abreviaturas e siglas

ACF *Autocorrelation Function* – Função de Autocorrelação.

ADF *Augmented Dickey–Fuller* – Teste de Dickey–Fuller Aumentado.

ANBIMA Associação Brasileira das Entidades dos Mercados Financeiro e de Capitais.

AR *Autoregressive Model* – Modelo Autorregressivo.

ARIMA *Autoregressive Integrated Moving Average* – Modelo Autorregressivo Integrado de Médias Móveis.

B3 Brasil, Bolsa, Balcão – Bolsa de valores brasileira.

BCB Banco Central do Brasil.

BPS *Basis Points* – Pontos Base.

CDB Certificado de Depósito Bancário.

CDI Certificado de Depósito Interbancário.

CDI+ Certificado de Depósito Interbancário acrescido de spread.

CRA Certificado de Recebíveis do Agronegócio.

CRI Certificado de Recebíveis Imobiliários.

CV *Cross-Validation* – Validação Cruzada.

CVM Comissão de Valores Mobiliários.

EN *Elastic Net* – Regressão Elastic Net.

FCC *Cross-Correlation Function* – Função de Correlação Cruzada.

FIF Fundo de Investimento Financeiro.

FII Fundo de Investimento Imobiliário.

FIP Fundo de Investimento em Participações.

GLS *Generalized Least Squares* – Mínimos Quadrados Generalizados.

HAC *Heteroskedasticity and Autocorrelation Consistent* – Erros robustos à heterocedasticidade e autocorrelação.

HGB *HistGradientBoostingRegressor* – Regressor baseado em *gradient boosting* com histogramas.

Idex-CDI JGP Índice de Debêntures Indexadas ao CDI desenvolvido pela gestora JGP.

JGP JGP Asset Management – Gestora de recursos.

L1 Penalização L1 (norma-1).

L2 Penalização L2 (norma-2).

LassoCV Lasso com Validação Cruzada.

LF Letra Financeira.

MAE *Mean Absolute Error* – Erro Absoluto Médio.

ML *Machine Learning* – Aprendizado de Máquina.

MQO Mínimos Quadrados Ordinários.

MSE *Mean Squared Error* – Erro Quadrático Médio.

OLS *Ordinary Least Squares* – Mínimos Quadrados Ordinários.

PACF *Partial Autocorrelation Function* – Função de Autocorrelação Parcial.

PIB Produto Interno Bruto.

PL Patrimônio Líquido.

PU Preço Unitário.

R$ Real – Moeda brasileira.

RidgeCV Ridge com Validação Cruzada.

RMSE *Root Mean Squared Error* – Raiz Quadrada do Erro Médio.

VAR *Vector Autoregression* – Modelo Vetorial Autorregressivo.

VECM *Vector Error Correction Model* – Modelo Vetorial de Correção de Erros.

WLS *Weighted Least Squares* – Mínimos Quadrados Ponderados.

Lista de símbolos

α	Parâmetro de regularização global no modelo Elastic Net, controlando a intensidade total da penalização.
$\hat{\alpha}$	Valor ótimo do hiperparâmetro de regularização obtido por validação cruzada.
β	Vetor de coeficientes estimados em regressões (Lasso, Ridge, Elastic Net).
β_0	Intercepto da equação de tendência linear.
β_1	Coeficiente angular da tendência linear.
$\hat{\beta}_{OLS}$	Estimador dos parâmetros sob Mínimos Quadrados Ordinários.
$\hat{\beta}_{WLS}$	Estimador dos parâmetros sob Mínimos Quadrados Ponderados.
$\hat{\beta}_{\text{EN}}$	Vetor de coeficientes estimado pelo modelo Elastic Net.
$\hat{\beta}_{\text{Lasso}}$	Vetor de coeficientes estimado pelo modelo Lasso.
$\hat{\beta}_{\text{Ridge}}$	Vetor de coeficientes estimado pelo modelo Ridge.
$\gamma(h)$	Função de autocovariância em defasagem h.
γ_1, γ_2	Coeficientes harmônicos na modelagem de sazonalidade.
ε	Vetor de erros aleatórios no modelo de regressão.
ε_t	Termo de erro aleatório (ruído branco) com média zero e variância constante.
$\hat{\sigma}_i^2$	Variância condicional estimada da i-ésima observação.
σ^2	Variância do erro aleatório.
θ	Vetor genérico de parâmetros a serem estimados (em contexto geral de modelos).
λ	Parâmetro de regularização (penalização L_1 ou L_2).
μ	Termo constante na regressão do teste ADF.
ϕ	Parâmetro de autorregressão no teste Dickey–Fuller Aumentado (ADF).
ψ_i	Coeficiente associado às defasagens Δz_{t-i} no teste ADF.
ρ	Parâmetro de mistura entre penalizações L_1 e L_2 no Elastic Net.

$\rho(h)$	Coeficiente de autocorrelação em defasagem h.
$\rho_S(k)$	Correlação de Spearman entre $\text{rank}(x_{t-k})$ e $\text{rank}(y_t)$.
$\bar{x}_k$, $\bar{y}_k$	Médias amostrais das séries x_t e y_t nos pares válidos.
$\bar{y}_W$	Média ponderada da variável dependente no modelo WLS.
$\mathbf{I}_p$	Matriz identidade de ordem p.
$\mathbf{X}$	Matriz de variáveis explicativas (dimensão $n \times p$).
$\mathscr{F}$	Conjunto de fundos considerados na amostra.
C_t	Componente cíclico da série temporal.
$CV(\alpha)$	Função de erro médio da validação cruzada para o parâmetro α.
d_t	Diferença entre os postos pareados de x_{t-k} e y_t na correlação de Spearman.
d_t^2	Quadrado da diferença entre postos pareados em Spearman.
$E_{i,t}$	Eventos de fluxo de caixa da debênture i (juros, amortizações, resgates).
$f(t, X_t)$	Função determinística ou estocástica que descreve a dependência temporal da série.
$f(\cdot)$	Função estimada por aprendizado de máquina que mapeia preditores em valores previstos.
$\hat{f}$	Função preditiva ajustada pelo modelo supervisionado.
H_0	Hipótese nula (existência de raiz unitária).
H_1	Hipótese alternativa (série estacionária).
$\hat{u}_t$	Série dessazonalizada (resíduos da regressão com dummies mensais).
$\hat{w}_i$	Peso estimado com base na variância condicional estimada $\hat{\sigma}_i^2$.
$\hat{y}_i$	Valor previsto pelo modelo na i-ésima observação.
$I_{T,0}$	Nível acumulado do índice entre a data inicial e o período T.
$I_{t,t-1}$	Variação do índice entre os períodos $t-1$ e t.
k	Defasagem temporal usada na análise de correlação cruzada.
K	Número de partições (folds) na validação cruzada.
$\ell(\cdot)$	Função de perda (por exemplo, erro absoluto ou quadrático).

n	Número total de observações do conjunto de avaliação.
n_k	Número de observações válidas após considerar uma defasagem k.
n_t	Número de títulos que compõem o índice no tempo t.
p	Número de variáveis preditoras em um modelo.
p-valor	Probabilidade de observar estatística igual ou mais extrema sob hipótese nula.
$PU_{i,t}$	Preço unitário de mercado da debênture i na data t.
$\Delta PU_{i,t}$	Variação do preço unitário do título i na data t.
$Q(\beta)$	Soma ponderada dos quadrados dos resíduos no modelo WLS.
$q_{i,t}$	Quantidade de títulos disponíveis do ativo i na data t.
$r^{(-j)}$	Resíduo parcial na regressão, excluindo o efeito da variável j.
$r_{xy}(k)$	Correlação cruzada entre as séries x_t e y_t na defasagem k.
R^2_W	Coeficiente de determinação ponderado no modelo WLS.
s	Período da sazonalidade (ex.: $s = 12$ para dados mensais).
S_t	Componente de sazonalidade da série temporal.
$S(z, \gamma)$	Operador de *soft-thresholding* usado na atualização dos coeficientes Lasso/Elastic Net.
t	Índice temporal ou instante de tempo.
T	Horizonte temporal total (número de períodos).
T_t	Componente de tendência da série temporal.
$\tilde{\mathbf{X}}$, $\tilde{\mathbf{y}}$	Matriz e vetor aumentados na formulação do Elastic Net.
u_t	Termo de erro ou componente não sistemático.
w_i	Peso inversamente proporcional à variância condicional do erro ε_i.
w_t	Peso atribuído à observação t no modelo WLS.
$w_{i,t}$	Peso do ativo i no índice na data t.
W	Matriz diagonal de pesos inversamente proporcionais à variância condicional.

x_t	Vetor de variáveis preditoras observadas no tempo t.
$x_{\cdot j}$	Coluna j da matriz de preditores $\mathbf{X}$.
X_t	Vetor de variáveis exógenas no tempo t.
y	Vetor $(n \times 1)$ de variáveis dependentes.
y_i	Valor observado na i-ésima observação.
y_t	Valor observado da variável no tempo t.
Δy_t	Primeira diferença da série temporal y_t.
z_t	Série em nível usada no teste ADF.
Δz_t	Primeira diferença da série z_t, usada para eliminar tendências estocásticas.

Sumário

1 Introdução

O mercado de crédito privado brasileiro tem se tornado cada vez mais relevante no sistema financeiro nacional, consolidando-se como uma fonte alternativa de financiamento às tradicionais operações bancárias (SILVA; OLIVEIRA, 2021). Essa evolução decorre da redução do crédito direcionado e pela maior participação dos fundos de investimento em renda fixa, que passaram a desempenhar papel central no financiamento corporativo. Os fundos de investimento reúnem alocações de diversos investidores sob gestão profissional e regras de diversificação, permitindo ganhos de escala, acesso a classes de ativos e serviços especializados. No contexto de crédito privado, as empresas vêm buscando diversificar suas fontes de captação, recorrendo a instrumentos de dívida emitidos diretamente no mercado de capitais (COSTA, 2022).

O crédito privado compreende os instrumentos de dívida emitidos por empresas e companhias securitizadoras, sendo que entre os principais instrumentos destacam-se as debêntures, as notas promissórias, os Certificados de Recebíveis Imobiliários (CRI) e os Certificados de Recebíveis do Agronegócio (CRA). Entre esses instrumentos, as debêntures correspondem a títulos de dívida de médio e longo prazo emitidos por sociedades anônimas, permitindo que às empresas captem recursos diretamente dos investidores, oferecendo, em contrapartida, o pagamento periódico de juros e o reembolso do principal no vencimento (MARQUES; SANTOS, 2020). Sua expansão torna-se bastante relevante, com emissões anuais ultrapassando R$ 100 bilhões a partir de 2023, evidenciando sua importância no crédito não direcionado (ANBIMA, 2024).

Por se tratarem de obrigações financeiras, as debêntures enquadram-se na categoria de instrumentos de renda fixa, cujo retorno é determinado em relação a um indexador, podendo ser prefixado, pós-fixado ou híbrido. No caso das debêntures pós-fixadas atreladas ao CDI (Certificado de Depósito Interbancário), a remuneração combina a variação da taxa de juros interbancária com um spread que reflete o risco de crédito da empresa emissora (ALMEIDA, 2021). Esse spread, por sua vez, incorpora diversos fatores, como o risco do emissor, as condições macroeconômicas e o prêmios exigidos pelos investidores para compensar a liquidez do ativo (NETO, 2000; SINATORA, 2016).

Dentre os fatores que constituem a estrutura de remuneração das debêntures, destaca-se o CDI (Certificado de Depósito Interbancário), taxa média das operações interbancárias de um dia, calculada e divulgada diariamente pela B3. O CDI tornou-se referência essencial para o mercado de renda fixa brasileiro, servindo como benchmark tanto para títulos pós-fixados quanto para a formação da taxa básica de juros da economia (ANBIMA,). Dessa forma, a indexação das debêntures ao CDI conecta o desempenho desses papéis às dinâmicas do sistema financeiro nacional.

Com o amadurecimento do mercado e o aumento da complexidade dos produtos financeiros, tornou-se imperativa a criação de índices de referência (benchmarks) capazes de mensurar o desempenho e gerir o risco dos ativos corporativos. Do ponto de vista de eficiência alocativa, a existência de um benchmark líquido e replicável facilita a precificação primária, alinhando novas ofertas ao valor justo observado no mercado secundário e a transparência para cotistas, que podem comparar retorno e risco de fundos com métricas padronizadas (SINATORA, 2016). Nesse contexto, a gestora de recursos JGP desenvolveu o Idex-CDI JGP, índice formado por debêntures líquidas e indexadas ao CDI, com a finalidade de servir como referência para mensurar o desempenho dos ativos de crédito privado no Brasil.

O índice incorpora critérios de elegibilidade que visam garantir transparência, liquidez e relevância econômica. Tais filtros asseguram que o índice mantenha uma composição representativa dos títulos corporativos mais líquidos, atuando como parâmetro robusto para a mensuração do desempenho e da dinâmica do crédito privado no Brasil (JGP, 2023).

Os spreads funcionam, nesse contexto, como um sinalizador da aversão ao risco dos agentes de mercado, no qual sua diminuição reflete confiança e estabilidade macroeconômica, enquanto seu aumento traduz elevação da incerteza, levando à desalocação e ao encolhimento dos fluxos de investimento (HAFFKI; HENNIG, 2025). Assim, o Idex–CDI JGP oferece não apenas um instrumento de acompanhamento de performance, mas também uma medida sintética da percepção agregada de risco e retorno no mercado de crédito corporativo brasileiro. Ao quantificar essa correlação, busca-se oferecer elementos que contribuam para o aperfeiçoamento da gestão de liquidez, fortaleçam a utilização de benchmarks de crédito, como o Idex–CDI JGP, e orientem políticas de investimento.

O presente estudo analisa a correlação entre a captação líquida dos fundos de crédito e o comportamento dos spreads do Idex-CDI JGP, investigando em que medida o desempenho do mercado influencia o fluxo de captação de recursos. Busca-se compreender em que medida a performance do mercado, expressa pela variação dos spreads, influencia o comportamento dos investidores institucionais e individuais. A hipótese central sustenta que períodos de valorização do índice associados à compressão dos spreads e à consequente valorização das cotas tendem a atrair aportes nos fundos, enquanto fases de deterioração do desempenho induzem resgates e retração das captações líquidas. Após, o estudo incorpora a utilização de técnicas de machine learning para a previsão dos spreads de crédito com o objetivo de aprimorar a compreensão das dinâmicas de mercado e antecipar movimentos de risco e retorno no segmento de crédito privado.

2 Fundamentação Teórica

2.1 Índice Idex-CDI JGP

O Idex–CDI JGP é construído como um índice de retorno total encadeado, metodologia tradicionalmente utilizada em índices financeiros e de preços para medir a evolução de um portfólio representativo ao longo do tempo. Essa estrutura tem como objetivo refletir o retorno acumulado de um conjunto de debêntures ponderadas por valor de mercado, incorporando não apenas as variações de preço, mas também os fluxos de caixa gerados pelos títulos (JGP Asset Management, 2023). Matematicamente, o índice é definido pelas Equações 2.3 e 2.4:

A amostra do Idex–CDI JGP é revisada no início de cada mês, com o objetivo de preservar a representatividade e a liquidez do universo de debêntures que compõem o índice. São incluídas apenas os títulos que, simultaneamente, atendem a critérios voltados à consistência de preços e à profundidade de mercado. De maneira que, incluem-se: (i) debêntures indexadas ao CDI, seja na forma percentual (%CDI) ou acrescida de spread (CDI+); (ii) presença de spreads de compra e venda em, no mínimo, 70% dos dias dos dois meses anteriores nas corretoras de referência; (iii) prazo remanescente superior a dois meses; (iv) a não conversibilidade em ações; (v) inexistência de benefícios fiscais específicos, como isenção de imposto de renda; e (vi) volume nominal mínimo de emissão de R$ 100 milhões na amostra geral do índice (JGP Asset Management, 2023).

A ponderação e o limite por ativo seguem a Equação 2.1, em que o peso $w_{i,t}$ de cada debênture i na data t é proporcional ao seu valor de mercado:

$$w_{i,t} = \frac{q_{i,t}\, PU_{i,t}}{\sum_{j=1}^{n_t} q_{j,t}\, PU_{j,t}} \tag{2.1}$$

com $\sum_{i=1}^{n_t} w_{i,t} = 1$ e limite máximo de 10% por emissão. Sempre que esse limite é excedido, os demais pesos são recalculados e redistribuídos, de modo a preservar a diversificação do índice. Nessa formulação, $q_{i,t}$ representa a quantidade de títulos em circulação e $PU_{i,t}$ o preço unitário de mercado (JGP Asset Management, 2023).

No cálculo diário, o preço unitário $PU_{i,t}$ é obtido a partir da menor taxa de compra observada nas corretoras de referência. A variação diária de preço $\Delta PU_{i,t}$ é decomposta em carrego, que representa o acumulo do CDI e dos fluxos contratuais; e marcação a mercado, associada às oscilações de taxa e de spread. A variação total é dada pela Equação 2.2:

$$\Delta PU_{i,t} = PU_{i,t} + E_{i,t} - PU_{i,t-1}, \tag{2.2}$$

em que $E_{i,t}$ agrega os eventos de fluxo de caixa, como pagamento de cupons, amortizações e resgates (JGP Asset Management, 2023).

Assim, o índice busca capturar a evolução, ao longo do tempo, de um portfólio hipotético de debêntures ponderadas por valor de mercado por meio de uma estrutura de retorno encadeado. Essa estrutura incorpora simultaneamente as variações de preço e os fluxos de caixa associados aos títulos que compõem a carteira de referência (JGP Asset Management, 2023). O nível acumulado do índice entre a data-base inicial e o período T é dado por:

$$I_{T,0} = \prod_{t=1}^{T} I_{t,t-1}, \tag{2.3}$$

em que $I_{t,t-1}$ representa o fator de retorno entre $t-1$ e t, definido por:

$$I_{t,t-1} = \sum_{i=1}^{n_t} w_{i,t} \left(\frac{\Delta PU_{i,t}}{PU_{i,t-1}} + 1 \right), \tag{2.4}$$

sendo n_t o número de debêntures elegíveis na data t e $w_{i,t}$ o peso de mercado da debênture i, calculado conforme a Equação 2.1. O termo $\frac{\Delta PU_{i,t}}{PU_{i,t-1}} + 1$ corresponde ao retorno bruto do título i entre $t-1$ e t, em que $\Delta PU_{i,t}$ é dado pela Equação 2.2 (JGP Asset Management, 2023).

Adota-se $I_{0,0} = 1$ em 1º de agosto de 2017, valor que estabelece o nível inicial da série histórica e serve como referência para o cálculo das variações subsequentes. Essa convenção facilita a comparação temporal do desempenho do índice, que passa a ser expresso em termos relativos ao valor inicial (JGP Asset Management, 2023).

2.1.1 Captação Líquida de Fundos

A captação líquida de fundos de investimento mede a variação do passivo dos fundos decorrente de subscrições e resgates de cotas. De maneira geral, isto representa a pressão de demanda de investidores sobre os ativos-alvo dos fundos e, portanto, constitui um vínculo entre o comportamento do investidor e a formação de preços no mercado de crédito privado (PRADO, 2025). As entradas e saídas de recursos nos fundos refletem tanto a conjuntura macroeconômica quanto os incentivos microeconômicos proporcionados por custos, liquidez e sinalização de desempenho relativos a alternativas de investimento (ANBIMA, 2023). Em particular, em fundos de crédito, a dinâmica de captação e resgates retroalimenta o ciclo de originação e negociação de títulos corporativos, condicionando prêmios de risco e disponibilidade de financiamento (PRADO, 2025).

Em sua formulação, parte-se da noção de que, para um universo $\mathscr{F}$ de fundos e um mês civil m, a captação líquida mensal agrega entradas e saídas diárias em uma grandeza única. Seja $\mathrm{App}_{f,d}$ o montante aplicado e $\mathrm{Red}_{f,d}$ o montante resgatado no fundo $f \in \mathscr{F}$ no dia d. Define-se

$$\mathrm{NF}_m = \sum_{d \in m} \sum_{f \in \mathscr{F}} \mathrm{App}_{f,d} - \sum_{d \in m} \sum_{f \in \mathscr{F}} \mathrm{Red}_{f,d}. \tag{2.5}$$

Para fins de interpretação macro e comparabilidade intertemporal, é conveniente a reescala em bilhões de reais,

$$\mathrm{NF}_m^{(\mathrm{bi})} = \mathrm{NF}_m \,/\, 10^9, \tag{2.6}$$

reduzindo a sensibilidade a variações residuais decorrentes de arredondamentos e atualizações retroativas. A agregação mensal, ao substituir a granularidade diária por períodos econômicos, atua como um filtro de ruído microestrutural e enfatiza a tendência do ciclo de captação, respeitando a estrutura de reconhecimento de eventos contábeis que incidem sobre o passivo dos fundos (SINATORA, 2016).

A interpretação econômica de NF_m está relacionado ao conceito em que entradas líquidas persistentes em classes de fundos de crédito operam como choques de demanda o lado comprador, pressionando as ofertas no mercado primário e secundário e, por consequência, os spreads de crédito. Em contraponto, ondas de resgates forçam vendas ou postergações de alocação, elevando prêmios requeridos em debêntures e encurtando o prazo efetivo de investimento (SINATORA, 2016).

2.2 Séries Temporais

Uma série temporal pode ser definida como uma sequência ordenada de observações de uma variável aleatória ao longo do tempo. Cada observação é representada por y_t, com $t = 1, 2, \ldots, T$ representando o índice temporal, busca-se estudar as dependências entre valores passados e futuros a fim de compreender o processo gerador dos dados e, a partir disso, produzir previsões consistentes (BOX et al., 2016). Em termos gerais, uma série temporal pode ser representada pela Equação 2.7:

$$y_t = f(t, X_t) + \varepsilon_t, \tag{2.7}$$

em que $f(t, X_t)$ é uma função determinística ou estocástica que descreve a estrutura sistemática da série, podendo incorporar componentes de tendência, sazonalidade e o efeito de variáveis exógenas X_t. O termo ε_t representa o erro aleatório, usualmente modelado como ruído branco com média zero e variância constante, isto é, $\varepsilon_t \sim WN(0, \sigma^2)$ (SHUMWAY; STOFFER, 2017).

Além diisso, uma série temporal pode ser entendida como uma realização finita de um processo estocástico, isto é, uma coleção de variáveis aleatórias indexadas no tempo, $\{Y_t, t \in T\}$. Cada Y_t é uma variável aleatória que descreve o valor potencial que a série pode assumir no instante t, enquanto a série observada $\{y_1, y_2, \ldots, y_T\}$ representa uma única amostra desse processo. O objetivo da análise é inferir as propriedades do processo subjacente com base nessa realização observada (BOX et al., 2016). Por conseguinte, a série temporal dada pela Equação 2.7 pode ser detalhada em quatro componentes fundamentais: tendência (T_t), sazonalidade (S_t), componente cíclico (C_t) e componente aleatório (ε_t).

A tendência corresponde ao comportamento de longo prazo da série, refletindo o movimento geral de crescimento, declínio ou estabilidade ao longo do tempo. Ela está associada a variações sistemáticas que não são periódicas e que, em geral, resultam de fatores estruturais, como progresso tecnológico, inflação, políticas econômicas ou mudanças demográficas (SHUMWAY; STOFFER, 2017). No caso de uma tendência linear, tem-se a Equação 2.8:

$$T_t = \beta_0 + \beta_1 t, \tag{2.8}$$

em que β_0 é o intercepto e β_1 o coeficiente angular. Quando há aceleração ou desaceleração, podem ser incluídos termos de ordem superior (quadrática, cúbica etc.), ampliando a flexibilidade do modelo. A extração da tendência pode ser realizada por técnicas de suavização, como médias móveis ou regressão (SHUMWAY; STOFFER, 2017).

A sazonalidade representa as flutuações regulares e previsíveis que ocorrem em intervalos de tempo fixos, geralmente dentro de um ano, mês, semana ou dia. Essas variações se repetem periodicamente e estão associadas a padrões climáticos, hábitos de consumo, calendário ou atividades produtivas. É caracterizada por um período s. Por exemplo, $s = 12$ em dados mensais (um ciclo anual) ou $s = 4$ em dados trimestrais (SHUMWAY; STOFFER, 2017). A sazonalidade pode ser modelada por funções harmônicas, conforme a Equação 2.9.

$$S_t = \gamma_1 \sin\left(\frac{2\pi t}{s}\right) + \gamma_2 \cos\left(\frac{2\pi t}{s}\right), \tag{2.9}$$

O componente cíclico corresponde a flutuações de longo prazo não periódicas, típicas de séries macroeconômicas, cujos intervalos variam conforme condições econômicas, tecnológicas ou sociais (SHUMWAY; STOFFER, 2017). Uma forma geral de representá-lo é:

$$C_t = g(t), \tag{2.10}$$

em que $g(t)$ é uma função suavizada de baixa frequência. Esses ciclos podem ser extraídos por métodos espectrais ou filtros passa-baixa, como o filtro de Hodrick–Prescott.

O componente aleatório, também denominado irregular ou resíduo, representa as variações imprevisíveis da série, atribuídas a fatores aleatórios ou eventos isolados que não seguem nenhum padrão sistemático. Esse termo é usualmente modelado como ruído branco, denotado por $\varepsilon_t \sim WN(0, \sigma^2)$, isto é, uma sequência de variáveis aleatórias independentes, com média zero e variância constante (BROWNLEE, 2018).

O ruído branco constitui a parte não explicada pelo modelo. Quando os resíduos de um modelo ajustado apresentam comportamento compatível com um ruído branco, entende-se que os componentes determinísticos e dependentes foram adequadamente capturados, restando apenas o ruído inerente ao processo (BROWNLEE, 2018).

Os modelos de decomposição de séries temporais são ferramentas fundamentais para compreender e representar a estrutura interna dos dados ao longo do tempo. Seu objetivo é separar a série observada em componentes distintos permitindo uma análise mais clara dos padrões determinísticos e das flutuações aleatórias presentes. Entre as abordagens mais utilizadas, destacam-se os modelos aditivos e multiplicativos, que diferem na forma como os componentes se combinam para gerar o comportamento total da série (SHUMWAY; STOFFER, 2017).

No modelo aditivo, supõe-se que os efeitos de tendência, sazonalidade e ciclo atuam de forma independente e linear, somando-se ao componente aleatório. Assim, a série pode ser representada pela Equação 2.11:

$$y_t = T_t + S_t + C_t + \varepsilon_t, \tag{2.11}$$

Quando a magnitude das flutuações depende do nível da série, utiliza-se o modelo multiplicativo, expresso pela Equação 2.12. Essa abordagem é indicada para séries em que a amplitude das flutuações cresce ou diminui com o nível da tendência, característica comum em séries econômicas e financeiras (BROWNLEE, 2018).

$$y_t = T_t \times S_t \times C_t \times \varepsilon_t. \tag{2.12}$$

Outro conceito fundamental é o de estacionaridade, condição necessária para a aplicação de diversos modelos lineares de séries temporais. Um processo é dito estritamente estacionário se suas propriedades estatísticas, como a distribuição de probabilidade conjunta, forem invariantes no tempo. Na prática, trabalha-se com a estacionaridade fraca (ou de segunda ordem), que exige apenas que a média $E(Y_t) = \mu$, a variância $\text{Var}(Y_t) = \sigma^2$ e a covariância $\text{Cov}(Y_t, Y_{t+h}) = \gamma(h)$ não dependam do instante t, mas apenas do atraso h (SHUMWAY; STOFFER, 2017).

A dependência entre as observações é quantificada por meio da função de autocorrelação (ACF), definida pela Equação 2.13

$$\rho(h) = \frac{\gamma(h)}{\gamma(0)}, \tag{2.13}$$

em que $\rho(h)$ mede o grau de correlação linear entre observações separadas por h períodos. Complementarmente, a função de autocorrelação parcial (PACF) mede a correlação entre y_t e y_{t-h} após eliminar o efeito intermediário das defasagens anteriores (BOX et al., 2016).

2.2.1 Dessazonalização de Séries Temporais

A presença de sazonalidade mensal em séries financeiras e de captação de recursos é um fenômeno amplamente documentado na literatura de econometria e análise de séries temporais, decorrente de fatores como calendários de pagamentos, recolhimentos tributários, janelas de mercado e efeitos institucionais recorrentes (SHAO et al., 2024). Em termos estatísticos, quando essa sazonalidade é de natureza determinística, sua remoção por meio de uma regressão linear com variáveis dummies mensais constitui uma técnica apropriada, pois permite isolar as variações sistemáticas associadas ao calendário e obter uma série residual centrada, que preserva apenas a dinâmica estocástica de curto prazo relevante para a inferência subsequente (BROCKWELL; DAVIS, 2009).

Considerando uma série temporal mensal y_t, por exemplo representando a captação líquida de recursos ou o spread bancário médio, a modelagem dessa estrutura sazonal determinística pode ser realizada pela Equação 2.14:

$$y_t = \alpha + \sum_{m=2}^{12} \delta_m D_{m,t} + u_t, \tag{2.14}$$

em que $D_{m,t}$ é a variável dummy que assume valor 1 quando a observação pertence ao mês m e 0 caso contrário, sendo janeiro a categoria de referência. Essa especificação segue o formato clássico de uma regressão com variáveis indicadoras, estimada por Mínimos Quadrados Ordinários (MQO) (BROCKWELL; DAVIS, 2009).

A série dessazonalizada é então definida pelos resíduos estimados da regressão:

$$\hat{u}_t \equiv y_t - \hat{\alpha} - \sum_{m=2}^{12} \hat{\delta}_m D_{m,t}. \tag{2.15}$$

onde $\hat{u}_t$ é ortogonal ao espaço gerado pelas dummies mensais, ou seja, não contém componente média mensal fixa, o que implica que quaisquer padrões sazonais determinísticos foram removidos. Assim, os resíduos $\hat{u}_t$ podem ser utilizados em testes de autocorrelação, heterocedasticidade ou na estimação de modelos dinâmicos, como ARIMA, VAR ou regressões com defasagens, sem viés mecânico de calendário (BROCKWELL; DAVIS, 2009).

Na aplicação ao crédito corporativo, a dessazonalização ajusta as séries de spreads e fluxos de fundos, tornando-as comparáveis entre meses com diferentes regimes de emissão e resgate. Ao remover os efeitos de calendário, a análise passa a refletir de forma mais acertiva as mudanças nas condições de mercado, eliminando distorções criadas por padrões operacionais previsíveis. Esse enquadramento está intimamente ligado à estrutura institucional do mercado de capitais e ao seu calendário operacional, que influenciam o comportamento dos agentes financeiros e as dinâmicas de captação e alocação de recursos ao longo do ano (BROCKWELL; DAVIS, 2009).

2.2.2 Estacionariedade de Séries Temporais

A estacionariedade é uma propriedade que implica a constância das características estatísticas da série ao longo do tempo. Em um processo estacionário, a média, a variância e a covariância entre observações dependem apenas da defasagem temporal e não do instante em que são observadas. Essa condição assegura que o comportamento da série permaneça estável e previsível, permitindo a aplicação de modelos lineares (NEUSSER et al., 2016).

A ausência de estacionariedade, frequentemente associada à presença de tendências estocásticas ou raízes unitárias, pode gerar inferências incorretas. Nesses casos, torna-se necessária a diferenciação ou transformação da série antes da modelagem, a fim de remover componentes não

estacionários e garantir a validade estatística das estimativas (BROCKWELL; DAVIS, 2016). O teste de Dickey–Fuller Aumentado (ADF) se baseia na regressão conforme a Equação 2.16:

$$\Delta z_t = \mu + \phi z_{t-1} + \sum_{i=1}^{p} \psi_i \Delta z_{t-i} + \varepsilon_t \tag{2.16}$$

em que z_t representa a série em nível (por exemplo, $z_t = \hat{u}_t$, isto é, os resíduos dessazonalizados), $\Delta z_t = z_t - z_{t-1}$ indica a primeira diferença, e ε_t é o termo de erro não correlacionado. Para o parâmetro ϕ sob a hipótese nula $H_0 : \phi = 0$, a série possui uma raiz unitária, ou seja, é não estacionária e contém uma tendência estocástica. Sob a hipótese alternativa $H_1 : \phi < 0$, a série é estacionária, significando que oscila em torno de uma média constante e apresenta variância finita. A inclusão das defasagens Δz_{t-i} (com $i = 1, 2, \ldots, p$) visa eliminar a autocorrelação serial dos resíduos, garantindo a validade assintótica dos testes de significância (BROCKWELL; DAVIS, 2016).

O teste ADF avalia se a série tende a retornar ao seu valor médio após choques temporários. Se o parâmetro estimado $\hat{\phi}$ for significativamente negativo, rejeita-se H_0, concluindo-se que a série é estacionária. Caso contrário, a série é considerada integrada de ordem um, denotada $I(1)$, o que implica que suas diferenças de primeira ordem são estacionárias (NEUSSER et al., 2016).

A estacionariedade é uma condição central para evitar correlações não genuínas e para garantir a validade de procedimentos de inferência, como estimações por Mínimos Quadrados Ordinários e testes de causalidade de Granger. Em séries financeiras e macroeconômicas, a presença de uma raiz unitária pode indicar elevada persistência de choques ou tendências de longo prazo associadas ao nível da atividade econômica, à taxa de juros ou aos spreads de crédito (WEI, 2019).

2.2.3 Correlação de Séries Temporais com Defasagens

A correlação linear de Pearson constitui a medida clássica de associação entre duas variáveis e pode ser estendida ao contexto de séries temporais por meio da correlação com defasagens. Nesse caso, investiga-se se variações em uma série x_t tendem a anteceder ou seguir alterações em outra série y_t, permitindo explorar relações dinâmicas ao longo do tempo. Essa generalização é formalizada pela Função de Correlação Cruzada (FCC), definida como o conjunto dos coeficientes de correlação de Pearson calculados para diferentes defasagens k (DERRICK; THOMAS, 2004).

Ademais, a FCC mede o grau de associação linear entre x_t e y_t para cada defasagem k. Quando $k > 0$, avalia-se se valores passados de x_t estão associados a valores atuais de y_t, sugerindo que x antecede y. Por outro lado, quando $k < 0$, examina-se se y tende a preceder x, o que indicaria uma relação temporal inversa (DERRICK; THOMAS, 2004). Para uma defasagem

k, o coeficiente de correlação de Pearson é dado pela Equação 2.17:

$$r_{xy}(k) \;=\; \frac{\sum_{t=k+1}^{n}\left(x_{t-k}-\bar{x}_k\right)\left(y_t-\bar{y}_k\right)}{\sqrt{\sum_{t=k+1}^{n}\left(x_{t-k}-\bar{x}_k\right)^2}\sqrt{\sum_{t=k+1}^{n}\left(y_t-\bar{y}_k\right)^2}}, \tag{2.17}$$

em que $\bar{x}_k$ e $\bar{y}_k$ são as médias amostrais nos $n_k = n-k$ pares válidos.

A significância aproximada da correlação cruzada $r_{xy}(k)$ pode ser avaliada por meio do teste t apresentado na Equação 2.18:

$$t\left[r_{xy}(k)\right] \;=\; r_{xy}(k)\sqrt{\frac{n_k-2}{1-r_{xy}(k)^2}}, \qquad t \sim t_{n_k-2} \tag{2.18}$$

Essa estatística de teste segue, sob certas condições, uma distribuição t de Student com n_k-2 graus de liberdade, em que $n_k = n-k$ representa o número de observações válidas após considerar a defasagem k. O valor calculado de $t[r_{xy}(k)]$ é então comparado com os valores críticos da distribuição t, permitindo avaliar se a correlação na defasagem k é estatisticamente significativa, isto é, se difere de zero em um determinado nível de confiança (TRIOLA, 2008).

Contudo, essa formulação parte da suposição de que as observações de x_t e y_t são independentes entre si no tempo. Em séries temporais reais, essa condição raramente se verifica, pois ambas as variáveis frequentemente exibem autocorrelação serial. Quando há autocorrelação, os valores de $r_{xy}(k)$ tendem a ser inflados, o que aumenta a probabilidade de rejeitar incorretamente a hipótese nula de ausência de correlação (TRIOLA, 2008).

Para contornar esse problema e obter uma inferência mais confiável, recomenda-se o procedimento conhecido como prewhitening. Essa técnica consiste em ajustar um modelo autorregressivo (AR) a uma das séries (geralmente x_t) com o objetivo de eliminar sua autocorrelação, aplicando em seguida o mesmo filtro à outra série y_t. A Função de Correlação Cruzada (FCC) é então calculada entre os resíduos filtrados dessas regressões, refletindo apenas as correlações contemporâneas e defasadas genuínas entre os componentes inovadores de x_t e y_t (WEI, 2019).

Além disso, quando ambas as séries apresentam memória temporal de longo prazo, o cálculo da FCC deve ser realizado sobre os resíduos de modelos univariados previamente ajustados. Essa prática remove efeitos autorregressivos e sazonais internos, garantindo que a correlação observada decorra da interdependência real entre as séries, e não de estruturas dinâmicas comuns subjacentes (WEI, 2019).

A correlação de Spearman é uma alternativa não paramétrica e robusta à correlação de Pearson, sendo especialmente útil quando as séries apresentam não normalidade, presença de *outliers* ou relações monotônicas não lineares. Diferentemente da medida de Pearson, que utiliza os valores observados, a correlação de Spearman baseia-se nas posições relativas (*ranks*) dos dados. Formalmente, ela é definida como a correlação de Pearson calculada entre os postos $\text{rank}(x_{t-k})$ e $\text{rank}(y_t)$, isto é, entre as ordens dos valores dentro de cada amostra (WOOLDRIDGE, 2025).

Em situações sem empates, a correlação de Spearman pode ser expressa de forma compacta, conforme a Equação 2.19:

$$\rho_S(k) \;=\; \mathrm{Corr}\big(\,\mathrm{rank}_{n_k}(x_{t-k}),\,\mathrm{rank}_{n_k}(y_t)\big) \;=\; 1 \;-\; \frac{6\sum_{t=k+1}^{n} d_t^2}{n_k\,(n_k^2-1)}, \tag{2.19}$$

em que $d_t = \mathrm{rank}_{n_k}(x_{t-k}) - \mathrm{rank}_{n_k}(y_t)$ representa a diferença entre os *ranks* pareados e $n_k = n-k$ é o número de pares válidos considerados no cálculo (WOOLDRIDGE, 2025).

Essa formulação possui uma interpretação intuitiva: quanto mais próximas forem as classificações relativas de x_{t-k} e y_t, menor será $\sum d_t^2$ e, portanto, maior será o coeficiente de correlação de Spearman. Assim, $\rho_S(k) = 1$ indica uma relação perfeitamente crescente, $\rho_S(k) = -1$ representa uma relação perfeitamente decrescente e $\rho_S(k) = 0$ sugere ausência de associação monotônica (WOOLDRIDGE, 2025).

A busca pela defasagem ideal que maximiza $|r_{xy}(k)|$ em uma grade $k \in \{0,\ldots,K\}$ introduz viés de seleção, na medida em que múltiplas comparações ampliam a probabilidade de se obter correlações elevadas apenas por acaso. Para contornar esse problema, busca-se restringir a grade de k com base em hipóteses econômicas *ex ante* sobre o mecanismo de transmissão (por exemplo, janelas de captação e prazos de *mark-to-market* de debêntures), de modo a limitar o espaço de busca e reduzir o risco de erro por seleção. Além disso, a defasagem escolhida deve ser submetida a validação fora da amostra ou em subperíodos, de forma a verificar a estabilidade temporal da relação e evitar sobreajuste. Por fim, é recomendado estimar um modelo dinâmico com controles para persistência, heterocedasticidade e autocorrelação (por exemplo, MQO com erros HAC ou WLS/GLS), registrando a incerteza de previsão e a sensibilidade a a diferentes janelas (FÁVERO; BELFIORE, 2017).

Quando as duas séries são originalmente não estacionárias, recomenda-se trabalhar com as versões dessazonalizadas e, se necessário, diferenciadas, de modo a aplicar a FCC em processos sem tendência estocástica. Alternativamente, em presença de cointegração, relações de longo prazo devem ser modeladas em um VECM, deixando a correlação defasada como ferramenta exploratória de curto prazo (WOOLDRIDGE, 2025).

2.2.4 Modelagem por Mínimos Quadrados Ponderados

A técnica dos Mínimos Quadrados Ponderados (Weighted Least Squares ou WLS) constitui uma extensão do método clássico dos Mínimos Quadrados Ordinários (OLS) e foi desenvolvida para lidar com situações em que a variância dos erros não é constante ao longo das observações, fenômeno conhecido como heterocedasticidade (MONTGOMERY; PECK; VINING, 2015). Em modelos de séries financeiras e macroeconômicas, a heterocedasticidade é frequente, uma vez que a volatilidade dos choques econômicos e as mudanças de regime de mercado fazem com que a variabilidade dos resíduos varie no tempo. O método de WLS ajusta o

modelo por meio da introdução de pesos inversamente proporcionais à variância condicional estimada de cada observação, buscando restaurar a eficiência dos estimadores (NORIEGA; RIVERA; HERRERA, 2023).

Nos modelos lineares clássicos, parte-se da formulação geral apresentada na Equação 2.20:

$$y = X\beta + \varepsilon, \tag{2.20}$$

em que y é o vetor $(n \times 1)$ de variáveis dependentes, X é a matriz $(n \times k)$ de regressores (ou variáveis explicativas), β é o vetor $(k \times 1)$ de parâmetros desconhecidos e ε é o vetor de erros aleatórios (MONTGOMERY; PECK; VINING, 2015). O estimador de OLS é obtido minimizando a soma dos quadrados dos resíduos, conforme a Equação 2.21:

$$\hat{\beta}_{OLS} = \arg\min_{\beta} (\varepsilon'\varepsilon) = (X'X)^{-1}X'y, \tag{2.21}$$

Para que esse estimador seja válido e eficiente, assumem-se as condições clássicas do modelo linear: $\mathbb{E}[\varepsilon|X] = 0$ e $\mathrm{Var}(\varepsilon|X) = \sigma^2 I_n$, ou seja, os erros possuem média condicional nula e variância constante, característica conhecida como homocedasticidade (MONTGOMERY; PECK; VINING, 2015).

Entretanto, quando essa hipótese é violada e a variância dos erros passa a depender das observações, tem-se a Equação 2.22:

$$\mathrm{Var}(\varepsilon|X) = \sigma^2 W^{-1}, \tag{2.22}$$

em que W é uma matriz diagonal $(n \times n)$ cujos elementos w_i representam pesos inversamente proporcionais à variância condicional de cada erro, isto é, $w_i \propto 1/\mathrm{Var}(\varepsilon_i)$ (MONTGOMERY; PECK; VINING, 2015). Nesse contexto, o estimador eficiente é obtido pela minimização da soma ponderada dos quadrados dos resíduos, conforme a Equação 2.23:

$$Q(\beta) = \sum_{i=1}^{n} w_i (y_i - x_i'\beta)^2, \tag{2.23}$$

Em notação matricial, tem-se a Equação 2.24:

$$Q(\beta) = (y - X\beta)'W(y - X\beta). \tag{2.24}$$

Derivando $Q(\beta)$ em relação a β e igualando a zero, obtém-se o estimador de WLS:

$$\hat{\beta}_{WLS} = (X'WX)^{-1}X'Wy. \tag{2.25}$$

De acordo com o teorema de Gauss–Markov generalizado, $\hat{\beta}_{WLS}$ é o melhor estimador linear não viesado sob heterocedasticidade, desde que os pesos w_i sejam conhecidos ou consistentemente estimados (GREENE, 2018). Em prática, os pesos são frequentemente obtidos de uma etapa preliminar de estimação, por exemplo, a partir dos resíduos de um modelo OLS inicial, em que se ajusta uma função para a variância condicional:

$$\hat{w}_i = \frac{1}{\hat{\sigma}_i^2}. \tag{2.26}$$

Essa reponderação permite restaurar a eficiência dos estimadores, corrigindo o problema de variância não constante e evitando que observações de alta variabilidade distorçam o ajuste global do modelo (MONTGOMERY; PECK; VINING, 2015).

A qualidade estatística do ajuste ponderado pode ser avaliada por meio de uma medida análoga ao coeficiente de determinação tradicional, o R^2 ponderado, definido na Equação 2.27:

$$R_W^2 = 1 - \frac{\sum_t w_t \, \hat{u}_t^2}{\sum_t w_t \, (y_t - \bar{y}_W)^2}, \qquad \bar{y}_W = \frac{\sum_t w_t \, y_t}{\sum_t w_t}, \tag{2.27}$$

em que w_t representa os pesos atribuídos a cada observação, $\hat{u}_t$ são os resíduos ponderados e $\bar{y}_W$ é a média ponderada da variável dependente. Essa medida expressa a proporção da variabilidade ponderada de y_t explicada pelo modelo, ajustando o critério de qualidade de ajuste à estrutura de heterocedasticidade tratada pelo método WLS. Assim, valores mais elevados de R_W^2 indicam que o modelo ponderado explica melhor a variação condicionalmente ponderada da amostra (MONTGOMERY; PECK; VINING, 2015).

2.3 Machine Learning

O aprendizado de máquina (Machine Learning ou ML) insere-se no campo da inteligência artificial e da estatística computacional, sendo voltado ao desenvolvimento de algoritmos capazes de identificar padrões em dados e realizar previsões ou inferências automáticas a partir da experiência empírica. Diferentemente dos modelos econométricos tradicionais, que são guiados por hipóteses teóricas sobre a estrutura de geração dos dados, os métodos de ML enfatizam abordagens orientadas a dados, capazes a capturar relações complexas e possivelmente não lineares entre variáveis explicativas e a variável de interesse (MURPHY, 2012).

Os fundamentos de Machine Learning aplicados à previsão de spreads de crédito em debêntures indexadas ao CDI possibilitam uma forma de representar o comportamento do mercado de crédito corporativo. Em finanças, essas técnicas são majoritariamente empregadas no regime supervisionado, em que se dispõe de uma variável-alvo y como por exemplo, o spread de compra convertido para CDI+, e de um conjunto de preditores X, o qual agrega informações de mercado, indicadores microeconômicos e variáveis macroeconômicas (DIXON; HALPERIN; BILOKON, 2020).

O objetivo central consiste em estimar uma função que relacione o espaço de preditores ao valor esperado da variável-alvo, conforme a Equação 2.28:

$$f : \mathbb{R}^p \to \mathbb{R}, \tag{2.28}$$

de modo que essa função produza previsões acuradas fora da amostra, isto é, maximize a capacidade de generalização do modelo (DIXON; HALPERIN; BILOKON, 2020).

Em problemas de previsão com séries temporais financeiras, é fundamental preservar a ordem cronológica dos dados no processo de particionamento da amostra, uma vez que o objetivo é simular o fluxo real de informações no tempo e evitar o chamado data leakage, que é a situação em que o modelo, inadvertidamente, utiliza informações futuras para prever o passado ou o presente. Assim, o conjunto de dados é tipicamente dividido em três partes: uma janela inicial de treino, utilizada para estimar os parâmetros do modelo; um conjunto de validação, empregado para a seleção de hiperparâmetros e comparação entre algoritmos; e um conjunto de teste, reservado para a avaliação final e imparcial do desempenho preditivo (NORIEGA; RIVERA; HERRERA, 2023).

Formalmente, considera-se um conjunto de pares ordenados $\{(y_t, x_t)\}_{t=1}^{T}$, em que y_t representa o alvo (por exemplo, o spread de crédito) e x_t o vetor de preditores observados no tempo t. O modelo de aprendizado supervisionado busca aprender uma função $f(\cdot)$ tal que, para instantes futuros $t > T_{\text{treino}}$, escrevemos $\hat{y}_t = f(x_t)$, deixando explícito que as previsões se baseiam apenas nas informações disponíveis até o fim da janela de treino e são avaliadas fora dessa janela (NORIEGA; RIVERA; HERRERA, 2023; WRIGHT; MA, 2022).

2.3.1 Regressão Lasso

A Regressão Lasso (Least Absolute Shrinkage and Selection Operator) é um método de regularização e seleção de variáveis que impõe uma penalização L_1 sobre os coeficientes do modelo linear. Essa técnica busca simultaneamente melhorar a capacidade de generalização das previsões e reduzir a complexidade do modelo, tornando-se uma abordagem relevante para a estatística moderna e do machine learning aplicado a dados econômicos e financeiros (FAHRMEIR et al., 2013).

A formulação matemática da Regressão Lasso é dada pela Equação 2.29:

$$\hat{\beta}^{\text{lasso}} = \arg\min_{\beta \in \mathbb{R}^p} \left\{ \frac{1}{2n} \sum_{i=1}^{n} \left(y_i - \beta_0 - \mathbf{x}_i^\top \beta \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\}. \tag{2.29}$$

em que $\lambda \geq 0$ é o parâmetro de regularização que controla o grau de penalização. O primeiro termo mede o erro de ajuste (soma dos quadrados dos resíduos), enquanto o segundo termo introduz uma penalização linear sobre os coeficientes, incentivando que alguns sejam exatamente iguais a zero (HASTIE; TIBSHIRANI; WAINWRIGHT, 2015).

O parâmetro de regularização λ exerce um papel fundamental no equilíbrio entre viés e variância do modelo. Quando $\lambda = 0$, a penalização é inexistente e o estimador coincide com o dos Mínimos Quadrados Ordinários (OLS), ou seja, o modelo busca ajustar perfeitamente os dados sem qualquer regularização. Para valores intermediários de λ, o Lasso realiza seleção automática de variáveis, mantendo apenas aquelas que contribuem significativamente para a explicação da variável dependente e eliminando as menos informativas, o que resulta em um modelo mais interpretável. Já para valores muito elevados de λ, a penalização domina completamente o termo de ajuste, levando ao encolhimento total dos coeficientes em direção a zero e, consequentemente, a um modelo degenerado, incapaz de capturar a variabilidade dos dados (FAHRMEIR et al., 2013).

A escolha ótima de λ é tipicamente feita por validação cruzada, minimizando o erro médio de previsão fora da amostra. Em implementações modernas, como a função `LassoCV` do pacote `scikit-learn`, esse processo é automatizado, testando diversos valores de λ e selecionando aquele que fornece o melhor desempenho preditivo (HASTIE; TIBSHIRANI; WAINWRIGHT, 2015). Do ponto de vista computacional, a solução do Lasso é obtida por *coordinate descent*, em que os coeficientes são atualizados iterativamente por meio do operador de *soft-thresholding*:

$$S(z, \gamma) = \text{sgn}(z) \max\{|z| - \gamma, 0\}. \tag{2.30}$$

Esse operador é o responsável pela esparsidade: quando o termo $|z|$ é menor que o limiar γ, o coeficiente correspondente é definido como zero; caso contrário, ele é reduzido de forma contínua.

Em aplicações financeiras e macroeconômicas, o Lasso é particularmente útil quando o número de preditores p é grande, ou quando existe multicolinearidade entre variáveis explicativas, por exemplo, múltiplos indicadores de risco de crédito, liquidez e condições de mercado. Nesses casos, o Lasso seleciona automaticamente os indicadores mais informativos, simplificando o modelo e melhorando sua interpretabilidade. No entanto, quando as variáveis são altamente correlacionadas, o Lasso tende a selecionar apenas uma entre várias variáveis similares, o que pode gerar instabilidade na seleção (ANDERSSON, 2020).

2.3.2 Regressão Ridge (RidgeCV)

A Regressão Ridge, também conhecida como regressão linear com regularização L2, é uma extensão do modelo de Mínimos Quadrados Ordinários desenvolvida para lidar com problemas de multicolinearidade e sobreajuste (overfitting) em conjuntos de dados com elevado número de preditores ou forte correlação entre variáveis explicativas. Em sua formulação clássica, a Regressão Ridge busca estimar o vetor de coeficientes β que minimiza a função de custo penalizada, conforme a Equação 2.31:

$$\hat{\beta}^{\text{ridge}} = \arg\min_{\beta \in \mathbb{R}^p} \left\{ \frac{1}{2n} \sum_{i=1}^{n} \left(y_i - \beta_0 - \mathbf{x}_i^\top \beta \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}. \tag{2.31}$$

onde y_i é a variável dependente, $\mathbf{x}_i$ o vetor de preditores, p o número de variáveis e $\lambda \geq 0$ o parâmetro de regularização, que controla a penalização sobre a magnitude dos coeficientes. Valores maiores de λ impõem uma restrição mais forte, reduzindo a variância do modelo e mitigando problemas de multicolinearidade, ao custo de um pequeno aumento no viés (GéRON, 2022).

O contraste entre as Equações 2.29 e 2.31 reside no termo de penalização adicionado ao erro de ajuste. Em ambos os casos, o primeiro termo das funções de custo mede a qualidade do ajuste do modelo aos dados, por meio da soma dos quadrados dos resíduos normalizada por $2n$, isto é, a discrepância entre os valores observados y_i e as predições lineares $\beta_0 + \mathbf{x}_i^\top \beta$. Esse componente é idêntico nas duas formulações e corresponde ao critério de Mínimos Quadrados Ordinários, de modo que, na ausência de penalização ($\lambda = 0$), tanto a Regressão Ridge quanto a Regressão Lasso colapsam para o estimador clássico de MQO (MURPHY, 2012).

A diferença torna-se evidente no segundo termo de cada equação, que introduz a regularização. Para a Regressão Ridge, a penalização é dada pela soma dos quadrados dos coeficientes, $\lambda \sum_{j=1}^{p} \beta_j^2$, correspondente à norma L_2. Já na Regressão Lasso, a penalização assume a forma $\lambda \sum_{j=1}^{p} |\beta_j|$, baseada na norma L_1. Enquanto o termo L_2 tende a encolher de maneira contínua todos os coeficientes em direção a zero, preservando em geral todas as variáveis no modelo, o termo L_1 induz esparsidade; ou seja, para valores adequados de λ, alguns coeficientes são forçados exatamente a zero, o que implica seleção automática de variáveis relevantes e exclusão das menos informativas ou redundantes (WU et al., 2025).

O parâmetro de regularização λ exerce, em ambos os modelos, o papel de controlar o equilíbrio entre viés e variância, mas seus efeitos práticos diferem em função da norma utilizada. Em valores muito reduzidos, tanto no Lasso quanto no Ridge, a penalização se torna desprezível, aproximando as soluções do estimador de MQO, com maior risco de sobreajuste e elevada variância nas predições. À medida que λ aumenta, a Regressão Ridge produz coeficientes progressivamente suavizados, reduzindo a variância e tornando o modelo mais estável em contextos de multicolinearidade ou alto número de preditores, porém sem promover seleção estrita de variáveis. No caso do Lasso, o aumento de λ não só encolhe os coeficientes como também zera alguns deles, simplificando a estrutura do modelo e as interpretações das variáveis, ao custo de um viés adicional (GéRON, 2022; HASTIE; TIBSHIRANI; WAINWRIGHT, 2015).

A solução analítica da Regressão Ridge é expressa na forma matricial pela Equação 2.32:

$$\hat{\beta}_{\text{Ridge}} = \left(\mathbf{X}'\mathbf{X} + \lambda \mathbf{I} \right)^{-1} \mathbf{X}'\mathbf{y}, \tag{2.32}$$

em que $\mathbf{X}$ é a matriz de preditores de dimensão $n \times p$, $\mathbf{y}$ o vetor de observações da variável

dependente, $\mathbf{I}$ a matriz identidade de ordem p, e λ o parâmetro de penalização previamente definido.

Essa formulação evidencia que o termo $\lambda\mathbf{I}$ atua como um fator de regularização que estabiliza a inversão da matriz $\mathbf{X}'\mathbf{X}$, especialmente em situações de multicolinearidade elevada ou quando p se aproxima de n. Ao inflar os autovalores da matriz de covariância dos preditores, a Regressão Ridge reduz a variância dos coeficientes estimados e evita soluções instáveis que surgem quando as variáveis explicativas são fortemente correlacionadas (GéRON, 2022).

A Equação 2.33 representa formalmente o cálculo da função de erro médio de validação cruzada utilizada na escolha do hiperparâmetro de regularização α na Regressão Ridge. Esse procedimento tem como objetivo avaliar a capacidade preditiva fora da amostra do modelo para diferentes valores de α, permitindo selecionar aquele que melhor equilibra viés e variância.

$$CV(\alpha) = \frac{1}{K}\sum_{k=1}^{K}\frac{1}{n_k}\sum_{i\in F_k}\ell\left(y_i, \hat{y}_i^{(-k)}(\alpha)\right), \tag{2.33}$$

onde K representa o número de partições (folds) da validação cruzada, nas quais o conjunto de dados é dividido de modo que cada observação sirva uma vez como teste e $K-1$ vezes como treino. O termo F_k indica o subconjunto de teste na iteração k, contendo n_k observações utilizadas para o cálculo do erro médio dessa partição. O valor observado da variável dependente é denotado por y_i, enquanto $\hat{y}_i^{(-k)}(\alpha)$ representa a previsão obtida com o modelo treinado fora do subconjunto F_k, isto é, com as observações restantes empregadas como amostra de treino. A função $\ell(\cdot)$ mede o erro de previsão, podendo corresponder ao erro absoluto médio (MAE), ao erro quadrático médio (MSE) ou a outra métrica apropriada ao contexto do problema (GéRON, 2022).

A escolha final do hiperparâmetro é feita conforme a Equação 2.34:

$$\hat{\alpha} = \arg\min_{\alpha} CV(\alpha), \tag{2.34}$$

O hiperparâmetro de regularização α controla a intensidade da penalização L_2, isto é, o grau de contração imposto aos coeficientes do modelo e portanto, valores grandes de α aumentam a penalização, promovendo maior suavização dos coeficientes e redução da variância das previsões (GéRON, 2022).

2.3.3 Regressão Elastic Net

A Regressão Elastic Net é uma extensão dos métodos de regularização Ridge (penalização L_2) e Lasso (penalização L_1), que combina as vantagens de ambos para produzir estimadores mais estáveis em contextos com alta correlação entre preditores e muitos parâmetros em relação ao número de observações (HASTIE; TIBSHIRANI; FRIEDMAN, 2009). O estimador Elastic Net é definido pela Equação 2.35:

$$\hat{\beta}_{\text{EN}} = \arg\min_{\beta} \left\{ \sum_{i=1}^{n} (y_i - \mathbf{x}_i'\beta)^2 + \lambda \left[(1-\alpha) \sum_{j=1}^{p} \beta_j^2 + \alpha \sum_{j=1}^{p} |\beta_j| \right] \right\}, \tag{2.35}$$

em que $\lambda \geq 0$ controla a intensidade global da regularização e $\alpha \in [0,1]$ define o grau de mistura entre as penalizações L_1 (Lasso) e L_2 (Ridge). Quando $\alpha = 0$, o modelo reduz-se à Regressão Ridge, enquanto para $\alpha = 1$ coincide com a Regressão Lasso. Valores intermediários de α produzem uma combinação ponderada entre ambas, equilibrando a seleção de variáveis e a estabilidade dos coeficientes (HASTIE; TIBSHIRANI; FRIEDMAN, 2009).

Essa combinação permite ao Elastic Net herdar o poder de seleção de variáveis do Lasso e a robustez à multicolinearidade da Ridge, evitando que variáveis altamente correlacionadas sejam arbitrariamente excluídas do modelo. Em vez disso, o método tende a selecioná-las em conjunto, com coeficientes suavizados em magnitude, o que é desejável em cenários de regressão com forte correlação entre indicadores de mercado, spreads e variáveis macroeconômicas (KUHN; JOHNSON, 2013).

Em aplicações financeiras, essa combinação é especialmente valiosa. Muitos indicadores como ratings de crédito, duration, momentum de spreads e métricas setoriais, capturam dimensões de risco semelhantes e são fortemente correlacionados. O Elastic Net trata essa redundância de forma equilibrada, evita excluir variáveis informativas correlacionadas, mas também impede que todas recebam coeficientes excessivos. Assim, o método produz um modelo estável e coerente com a estrutura de correlação dos dados financeiros, o que o torna particularmente adequado para previsão de spreads de crédito e análise de fatores de risco (KUHN; JOHNSON, 2013).

Para variáveis centradas e padronizadas, o problema de otimização do Elastic Net conforme definido na Equação 2.35 pode ser resolvido de forma eficiente por meio do algoritmo de *coordinate descent* (descida coordenada). Esse método é amplamente empregado em regressões regularizadas por sua simplicidade computacional e convergência garantida em funções convexas separáveis, como é o caso do Elastic Net (HASTIE; TIBSHIRANI; WAINWRIGHT, 2015). A ideia central consiste em atualizar iterativamente cada coeficiente β_j enquanto mantém os demais fixos. Em cada iteração, o modelo recalcula o impacto da variável $x_{\cdot j}$ sobre o vetor de resíduos parciais, definido pela Equação 2.36:

$$r^{(-j)} = y - \sum_{k \neq j} x_{\cdot k} \beta_k, \tag{2.36}$$

que representa o componente do erro explicável apenas pela j-ésima variável, considerando que as demais já foram ajustadas. A atualização de β_j é então realizada aplicando-se o operador de *soft-thresholding*, anteriormente apresentado pela Equação 2.30.

A regra de atualização coordenada do Elastic Net é então expressa pela Equação 2.37:

$$\hat{\beta}_j \leftarrow \frac{S\left(\frac{1}{n} x_{\cdot j}^{\top} r^{(-j)},\, \alpha\rho\right)}{1+\alpha(1-\rho)}, \tag{2.37}$$

em que $\frac{1}{n} x_{\cdot j}^{\top} r^{(-j)}$ mede a correlação parcial entre a variável x_j e o resíduo atual, α é o parâmetro global de regularização, que define a intensidade total da penalização e $\rho \in [0,1]$ é o parâmetro de mistura entre os termos L_1 e L_2, controlando o equilíbrio entre esparsidade e suavização.

Quando o termo $|\frac{1}{n} x_{\cdot j}^{\top} r^{(-j)}| \leq \alpha\rho$, o numerador da fração em 2.37 é zerado, resultando em $\hat{\beta}_j = 0$. Essa condição representa o mecanismo de seleção de variáveis, característica herdada da penalização L_1. Por outro lado, quando o valor absoluto da correlação excede o limiar $\alpha\rho$, ocorre o encolhimento contínuo do coeficiente, controlado pelo denominador $(1+\alpha(1-\rho))$, que modera a magnitude de $\hat{\beta}_j$ com base na penalização L_2 (MONTGOMERY; PECK; VINING, 2015).

Esse procedimento iterativo é repetido ciclicamente sobre todos os coeficientes até a convergência. Sua implementação é computacionalmente eficiente mesmo em conjuntos de dados de alta dimensão, e constitui o núcleo dos algoritmos modernos empregados em pacotes como `glmnet` e `scikit-learn` (GéRON, 2022).

O Elastic Net pode ser interpretado de forma equivalente a um modelo Lasso aplicado a um conjunto de dados "aumentado", o que fornece uma perspectiva geométrica e algébrica útil para compreender suas propriedades. Nesse enquadramento, o problema do Elastic Net é reescrito sobre um espaço expandido das variáveis explicativas e da resposta, definidos pela Equação 2.38:

$$\tilde{\mathbf{X}} = \begin{bmatrix} \mathbf{X} \\ \alpha(1-\rho)\,\mathbf{I}_p \end{bmatrix}, \quad \tilde{\mathbf{y}} = \begin{bmatrix} \mathbf{y} \\ \mathbf{0} \end{bmatrix}, \tag{2.38}$$

com penalização $\alpha\rho\,\|\beta\|_1$. Essa representação mostra que o termo de penalização L_2, correspondente à parte Ridge, pode ser visto como a adição de observações artificiais $\left(\alpha(1-\rho)\,\mathbf{I}_p\right)$ que "puxam" os coeficientes β em direção a zero, promovendo estabilidade numérica. Ao mesmo tempo, a penalização L_1 atua como no Lasso, forçando esparsidade e eliminando variáveis com baixa contribuição preditiva. Essa combinação explica o efeito de agrupamento. Variáveis altamente correlacionadas tendem a entrar ou sair do modelo em conjunto, com coeficientes de magnitudes similares, em vez de uma seleção arbitrária de apenas uma delas (MONTGOMERY; PECK; VINING, 2015).

O parâmetro α define o grau global de encolhimento, onde quanto maior seu valor, maior o viés e menor a variância do estimador, pois o modelo se torna mais regularizado. Já o `l1_ratio` (ρ) controla o equilíbrio entre esparsidade e estabilidade sob colinearidade, onde valores próximos de 1 favorecem a seleção agressiva de variáveis, aproximando o modelo do Lasso

e valores intermediários (geralmente entre 0,1 e 0,9) promovem um equilíbrio entre parcimônia e agrupamento, tornando o modelo mais robusto à multicolinearidade (MONTGOMERY; PECK; VINING, 2015).

A escolha conjunta dos hiperparâmetros (α, ρ) deve ser realizada por validação cruzada, selecionando a combinação que minimize o erro médio de previsão fora da amostra. Em dados temporais ou de painel, é imprescindível utilizar esquemas de validação que preservem a ordem temporal, como *blocked* ou *expanding window cross-validation*, para evitar o viés de antecipação, que é um erro comum em previsões financeiras, quando o modelo inadvertidamente utiliza informações futuras para ajustar o passado (GéRON, 2022).

Como a penalização depende diretamente da escala das variáveis, é necessário que as colunas de $\mathbf{X}$ sejam padronizadas, isto é, transformadas para média zero e variância unitária, de modo que todas as variáveis recebam penalizações comparáveis. Essa padronização assegura que o peso da regularização não seja distorcido por diferenças de unidade ou magnitude entre os preditores. Por convenção, o intercepto β_0 não é penalizado, o que evita deslocamentos indevidos da média das previsões, sendo essa a configuração padrão em implementações modernas, como a da função `ElasticNetCV` do pacote `scikit-learn` (GéRON, 2022).

2.3.4 Regressão Hist Gradient Boosting Regressor (HGBR)

O Gradient Boosting é uma técnica de aprendizado supervisionado que combina múltiplos modelos fracos, geralmente árvores de decisão, para formar um modelo com alta capacidade preditiva. O método baseia-se na minimização sequencial dos erros residuais, em que cada novo modelo é ajustado para corrigir as falhas do anterior. De forma geral, busca-se estimar uma função $F(x)$ capaz de aproximar uma variável resposta y, minimizando uma função de perda $L(y, F(x))$ (FRIEDMAN, 2001). A equação básica do modelo é expressa pela Equação 2.39:

$$F_M(x) = \sum_{m=1}^{M} \gamma_m h_m(x), \tag{2.39}$$

em que $F_M(x)$ corresponde à predição final após M iterações, $h_m(x)$ representa o m-ésimo estimador base, e γ_m denota o peso de contribuição desse estimador, aprendido de forma iterativa por meio de gradiente descendente funcional (FRIEDMAN, 2001).

A cada iteração, o algoritmo ajusta uma nova árvore $h_m(x)$ para aproximar o gradiente negativo da função de perda em relação às previsões anteriores, conforme a Equação 2.40:

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}, \tag{2.40}$$

onde r_{im} representa o pseudo-resíduo da observação i na iteração m (FRIEDMAN, 2001). Em seguida, o modelo é atualizado de acordo com a Equação 2.41:

$$F_m(x) = F_{m-1}(x) + \nu\,\gamma_m\,h_m(x), \tag{2.41}$$

sendo $\nu \in (0,1]$ o fator de aprendizado, responsável por controlar a contribuição de cada árvore no ajuste final e, portanto, o grau de regularização por *shrinkage* do modelo (FRIEDMAN, 2001).

O Hist Gradient Boosting Regressor (HGBR) é uma implementação moderna e otimizada do Gradient Boosting, que utiliza histogramas de frequência para discretizar variáveis contínuas, reduzindo o custo computacional e permitindo lidar com grandes volumes de dados. Enquanto o modelo tradicional examina todos os pontos possíveis de divisão nas variáveis preditoras, o HGBR agrupa os valores contínuos em intervalos discretos, chamados *bins*. Cada *bin* representa uma faixa de valores, e as divisões das árvores de decisão são determinadas com base em estatísticas agregadas desses intervalos, como médias, somas de erros ou gradientes locais. Esse procedimento reduz substancialmente o número de cálculos necessários para determinar os pontos de corte ideais, diminuindo o tempo de treinamento e o consumo de memória sem comprometer, em geral, a precisão preditiva (FABIAN, 2011).

Visando complementar os modelos lineares penalizados como Ridge, Lasso e Elastic Net, o HGBR atua diretamente sobre os coeficientes por meio da regularização, controlando a complexidade do modelo, mitigando multicolinearidade e, no caso do Lasso e do Elastic Net, promovendo as soluções esparsas, com um subconjunto reduzido de variáveis ativas (KAZYMYR et al., 2025).

Esta regularização não se dá por penalização explícita de coeficientes em um espaço linear de regressão, mas pela imposição de restrições estruturais ao conjunto de árvores adicionadas de forma sequencial o que implica na mensuração da profundidade máxima das árvores, número mínimo de observações em cada folha, coeficiente de regularização L_2 nos pesos das folhas, tamanho do *learning rate* e, eventualmente, critérios de parada antecipada. Esses hiperparâmetros desempenham papel do parâmetro de penalização λ nos modelos Ridge, Lasso e Elastic Net, pois governam o equilíbrio entre viés e variância, evitando que o ensemble de árvores se ajuste em excesso ao ruído das séries históricas (KAZYMYR et al., 2025).

Nesse contexto, os modelos penalizados capturam a estrutura global linear das relações entre o spread e dessazonalização, enquanto o HGBR atua sobre os resíduos dinâmicos dessa estrutura, aprendendo padrões não lineares e interações de ordem superior entre as variáveis. Além disso, como o HGBR é invariável a transformações monotônicas e não exige padronização das variáveis, ele consegue explorar de forma eficiente heterogeneidades em escala e distribuição, o que é particularmente relevante em séries financeiras sujeitas a períodos de estresse, mudanças de regime e caudas pesadas (PANDA et al., 2024).

2.4 Métricas de Avaliação

As métricas de avaliação de desempenho são fundamentais na análise de modelos preditivos, pois quantificam a discrepância entre os valores observados e os estimados. No contexto de previsão de séries temporais financeiras, como spreads de crédito ou fluxos de captação, essas métricas permitem mensurar de forma objetiva a qualidade das previsões e comparar diferentes modelos sob critérios consistentes de erro.

2.4.1 Erro Absoluto Médio (MAE)

O Erro Absoluto Médio (MAE) quantifica a magnitude média dos erros de previsão, independentemente de seu sinal, sendo definido pela Equação 2.42:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|, \tag{2.42}$$

onde y_i representa o valor observado da variável dependente na i-ésima observação, enquanto $\hat{y}_i$ corresponde ao valor previsto pelo modelo para essa mesma observação. O termo n indica o número total de observações incluídas no conjunto de avaliação, ou seja, o tamanho da amostra utilizada para calcular o erro médio (JAPKOWICZ; BOUKOUVALAS, 2024).

Essa métrica expressa o desvio médio absoluto entre os valores previstos e observados, mantendo as mesmas unidades da variável analisada, o que facilita sua interpretação direta. Por atribuir peso igual a todos os erros, o MAE não amplifica discrepâncias ocasionais, sendo robusto a valores extremos e representativo da precisão média típica do modelo (GÜN; KARTAL, 2025).

2.4.2 Raiz do Erro Quadrático Médio (RMSE)

A Raiz do Erro Quadrático Médio (RMSE) é especialmente utilizada em séries temporais e aplicações financeiras por sua capacidade de capturar a magnitude média dos erros com ênfase em desvios grandes. Sua formulação é dada pela Equação 2.43:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}. \tag{2.43}$$

onde y_i representa o valor observado da variável dependente na i-ésima observação, $\hat{y}_i$ é o valor previsto pelo modelo para essa mesma observação, e n corresponde ao número total de observações do conjunto de avaliação (GÜN; KARTAL, 2025).

Diferentemente do MAE, o RMSE eleva os erros ao quadrado antes de calculá-los, de modo que erros maiores recebem peso desproporcionalmente maior na média. Em seguida, a raiz quadrada é aplicada para que o resultado retorne à mesma unidade da variável analisada, o que permite interpretações diretas, por exemplo, em pontos-base ou percentuais. Essa característica

faz com que o RMSE seja mais sensível a valores extremos e, portanto, uma métrica mais adequada quando se deseja penalizar fortemente previsões que apresentem grandes desvios em relação aos valores reais (JAPKOWICZ; BOUKOUVALAS, 2024).

3 Metodologia

A metodologia adotada neste trabalho busca integrar técnicas econométricas tradicionais e métodos modernos de aprendizado de máquina para analisar a relação entre spreads de crédito corporativo e fluxos de captação líquida em fundos de crédito no mercado brasileiro. O estudo combina etapas de pré-processamento de dados, modelagem estatística e avaliação de desempenho preditivo. Realiza-se uma análise estatística da correlação entre as variações do Idex-CDI JGP e as captações líquidas de fundos de crédito privado, com base em dados disponibilizados pela JGP (JGP, 2023) e pela ANBIMA (ANBIMA, 2024).

O universo analítico considerado nesta pesquisa é composto exclusivamente por fundos de investimento autorizados a aplicar em debêntures e demais ativos de crédito privado, conforme estabelecido em suas diretrizes de Informações Básicas, em conformidade com o Suplemento B do Anexo Normativo I da Resolução CVM nº 175, referente aos Fundos de Investimento Financeiros (FIF). Dessa forma, foram incluídos apenas os fundos cuja política de investimento explicita de forma clara a possibilidade de alocação em crédito privado. Fundos que não admitem essa classe de ativo em sua política ou que não informam o limite de alocação, foram excluídos da amostra garantindo a coerência regulatória (CVM, 2022).

A Resolução CVM nº 175 estabelece salvaguardas específicas para fundos cuja exposição em ativos de crédito privado ultrapassa 50% do patrimônio líquido (PL) (Subseção VII, art. 70), determinando, entre outras exigências, o uso do sufixo “Crédito Privado” e a inclusão de alertas adicionais de risco. Com base nesse arcabouço, adotou-se, nesta pesquisa, um critério mais restritivo, considerando como efetivamente expostos ao segmento aqueles fundos com alocação mínima de 67% do PL em ativos de crédito privado. Esse corte mais estrito tem por objetivo assegurar a materialidade da exposição, evitando a inclusão de fundos cuja presença em crédito é marginal ou residual (CVM, 2022).

De modo simétrico, foram excluídos os fundos com exposição inferior a 5% em debêntures, uma vez que seus fluxos são predominantemente determinados por outras classes de ativos como CDBs, Letras Financeiras (LFs) e títulos públicos, e não refletem adequadamente as dinâmicas do mercado de crédito corporativo.

Por fim, para manter a consistência regulatória e setorial, a amostra exclui Fundos Imobiliários (FII) e Fundos de Participação (FIP), que possuem regimes jurídicos e universos de ativos distintos, regidos pelos Anexos III e IV da Resolução CVM nº 175. Essas categorias são voltadas, respectivamente, a investimentos em ativos imobiliários e participações societárias, não sendo comparáveis ao escopo deste estudo, que se concentra na dinâmica dos fundos de crédito corporativo listável e negociável.

Do ponto de vista operacional, a aplicação dos filtros metodológicos ocorre antes de

qualquer etapa de agregação de fluxos, garantindo que apenas fundos elegíveis ao universo de crédito privado componham a base analítica.

3.1 Base de Dados e Tratamentos

A escolha do recorte temporal iniciado em abril de 2022 fundamenta-se na necessidade de garantir a consistência estatística e a representatividade estrutural da amostra. O período precedente foi marcado por uma ruptura na dinâmica do crédito privado brasileiro, precipitada pela pandemia de COVID-19. Em março de 2020, o mercado enfrentou uma crise de liquidez sistêmica que interrompeu praticamente todas as negociações no mercado secundário de debêntures, resultando em uma abertura abrupta dos spreads de crédito e em resgates massivos na indústria de fundos. Dados de gestoras como a JGP indicam que os prêmios de risco médios sobre o CDI saltaram de um patamar de aproximadamente 2% para cerca de 5% no auge da crise, enquanto a ANBIMA reportou que o spread médio ponderado do índice IDA–DI atingiu 4,31% em março de 2020, o ponto mais alto da série histórica (JGP, 2023; ANBIMA, 2023).

Concentrar a análise a partir de 2022, portanto, permite capturar um ambiente mais homogêneo, livre das distorções provocadas pelos choques pandêmicos e mais representativo das condições contemporâneas de precificação, liquidez e governança do crédito privado. O mercado pós-pandemia deixoa de ser apenas uma alternativa de *yield*,rendimento ou lucro gerado por um investimento e passou a constituir um pilar estrutural de financiamento corporativo, com spreads mais racionais, gestão de risco aprimorada e maior capacidade de absorver choques, como os casos da Americanas e da Light em 2023 (ANBIMA, 2023). Assim, a base de dados empregada neste estudo reflete uma fase de maturidade e estabilidade relativa, essencial para a análise dos determinantes contemporâneos dos prêmios de risco no mercado de debêntures.

Conforme o Apêndice A, o script inicia pela importação das bibliotecas destinadas à manipulação e à limpeza dos dados. As bibliotecas `pandas` e `numpy` fornecem estruturas de dados e operações vetorizadas para séries e tabelas. Em seguida, definem-se dois caminhos absolutos para as planilhas-fonte, uma com os fluxos de aplicações e resgate de fundos advinda da base disponibilizada pela CVM (CVM, 2025) e outra com a série histórica do Idex-CDI (JGP, 2025). Por fim, duas asserções (`assert`) verificam a existência desses arquivos antes de qualquer processamento; na ausência de algum deles, a execução é interrompida com mensagem explícita, resguardando a integridade dos resultados.

3.1.1 Captação Líquida dos Fundos de Investimento

Essa etapa transforma uma base operacional e detalhada em uma série temporal consolidada e padronizada, que reflete a evolução mensal dos fluxos líquidos de recursos no mercado de fundos de crédito. A variável resultante, *Captação Líquida (bi)*, constitui um indicador da

direção e intensidade dos fluxos de investimento, servindo de contraparte à série de spreads ponderados do Idex–CDI nas análises de correlação e nos modelos preditivos subsequentes.

Após eliminar linhas sem data válida, o código soma os valores de aplicações e resgates, gerando as colunas "Captação" e "Resgate". O resultado é uma tabela (`agg-fundos`) ordenada cronologicamente, que representa o volume total de recursos movimentados a cada mês. Por fim, calcula-se a *Captação Líquida (bi)*, a diferença entre o total de aplicações e o total de resgates, dividida por 10^9, e portanto expressa o resultado em bilhões de reais, unidade que facilita a interpretação e a comparação com as demais variáveis macroeconômicas analisadas.

3.1.2 Spread de Crédito

A segunda etapa do código apresentado no Apêndice A é responsável por transformar a base do Idex–CDI JGP em uma série mensal de spreads médios ponderados, devidamente ajustada e limpa de ruídos causados por emissores problemáticos. Os dados fornecido por "Data", "Spread de compra (%)", "Peso no índice (%)" e "Emissor" fornecem as informações necessárias para calcular o spread agregado, pois cada emissão contribui proporcionalmente ao seu peso de mercado. Após a leitura, a coluna de datas é convertida para o tipo `datetime`, e as observações sem data válida são removidas, garantindo a consistência temporal antes da agregação.

Na sequência, o código aplica um filtro que exclui emissões associadas a empresas em situação de *distress*, notadamente "Lame", "Light", "Aeris", "Viveo" e "Americanas S.A.". Esses emissores estiveram envolvidos em episódios de inadimplência, recuperação judicial ou reestruturação de dívida, o que produz cotações distorcidas, não representativas das condições médias do crédito corporativo. A exclusão é realizada por meio do método `str.contains()` combinado com uma expressão regular formada por `"|".join(distressed)`, de modo a eliminar todas as linhas cujo campo "Emissor" contenha qualquer uma dessas denominações, sem distinção entre maiúsculas e minúsculas. Para isso, utilizou-se o código presente no Quadro 1:

Quadro 1 – Remoção de emissores em *distress* do painel do Idex–CDI

```
distress = ["Americanas", "Light", "LAME", "Aeris", "Viveo"]
pat = "|".join(distress)
df_idex = df_idex[~df_idex["Emissor"].str.contains(pat, case=
    False, na=False)]
```

Fonte: Elaborado pelo autor.

Após o saneamento, as variáveis "Spread de compra (%)" e "Peso no índice (%)" são convertidas para escala fracionária com a função `tofraction()`, que transforma percentuais como 2,5 em 0,025. Essas colunas passam a ser denominadas `Spreadfrac` e `Pesofrac`, servindo como base para o cálculo da média ponderada.

O cálculo da média ponderada é realizado pela função `wavg(g)`, que recebe um subconjunto mensal do `DataFrame` e utiliza `numpy.average()` para determinar a média dos spreads ponderada pelos pesos relativos. Verificações condicionais asseguram que os pesos não sejam nulos e que a soma total não seja zero, prevenindo divisões indefinidas em períodos de menor liquidez.

O resultado é um novo conjunto de dados denominado `spreadmensal`, com as colunas "Mês" e "Spread ponderado (fração)". Para facilitar a interpretação, o valor é reconvertido para percentual ao multiplicar-se por 100, gerando a coluna final "Spread ponderado".

3.1.3 Fusão das bases: Captação Líquida e Spread Ponderado

A última parte do código apresentado no Apêndice A realiza a integração entre as bases de captação líquida dos fundos e o índice Idex–CDI, consolidando em uma única tabela todas as variáveis necessárias para as análises econométricas e preditivas subsequentes. Essa etapa realiza o cruzamento temporal das séries, o refinamento visual dos dados e a exportação do resultado final em formato adequado para modelagem.

O processo inicia-se com a junção (`merge`) entre o conjunto agregado de fluxos de fundos (`aggfundos`) e a série mensal de spreads ponderados (`spreadmensal`), conforme apresenta-se no Quadro 2. O parâmetro `on="Mes"` garante que o cruzamento ocorra pela variável temporal comum, que é o mês de referência, enquanto o método `how="inner"` restringe o resultado às observações que aparecem simultaneamente em ambas as bases, assegurando consistência amostral. Assim, apenas os meses com registros simultâneos de captação e spreads são mantidos. Em seguida, a coluna "Mes" é renomeada para "Data", adequando-se ao padrão utilizado no restante do trabalho, e o conjunto resultante é ordenado cronologicamente com `sortvalues("Data")`, produzindo uma sequência temporal contínua e coerente.

Quadro 2 – Fusão das bases e ordenação temporal.

```
1 tabela = (agg_fundos.merge(spread_mensal[['Mes', '
    spread_ponderado']],
2                            on='Mes', how='inner')
3                    .rename(columns={'Mes': 'Data'})
4                    .sort_values('Data'))
```

Fonte: Elaborado pelo autor.

Após o `merge`, o código seleciona explicitamente as colunas de interesse para as análises ("Data", "Captação", "Resgate", "Captação Líquida (bi)" e "Spread ponderado"), eliminando variáveis intermediárias utilizadas apenas em cálculos auxiliares. Essa estrutura representa a base consolidada final, contendo, para cada mês, o volume total aplicado e resgatado pelos fundos, o saldo líquido de captação (em bilhões de reais) e o spread médio ponderado das debêntures

corporativas do Idex–CDI. Em seguida, o `DataFrame` resultante é exportado em formato Excel por meio do comando `toexcel()`, sendo salvo no mesmo diretório do arquivo-base com o nome `tabelamensal fluxospread.xlsx`. O caminho de saída (`outpath`) é construído a partir de `pathbase.parent`, permitindo uma verificação preliminar da consistência e da formatação do resultado.

Por fim, com o propósito de complementar as análises e oferecer uma leitura conjunta dos principais sinais do mercado de crédito privado, implementou-se um módulo de visualização (descrito no Apêndice B) que representa, no mesmo horizonte temporal, a Captação Líquida mensal dos fundos de crédito (em bilhões de reais) e o Spread Ponderado (%) das debêntures que compõem o índice Idex–CDI. Essa visualização tem como objetivo identificar os regimes de mercado, correlações e pontos de inflexão na relação entre liquidez e prêmios de risco, permitindo observar, por exemplo, se entradas líquidas de recursos antecedem a compressão de spreads ou se ondas de resgates se associam à sua abertura.

3.2 Análise da Captação Líquida vs Spread Ponderado

O código apresentado no Apêndice C constitui a base da etapa econométrica do estudo, implementando a regressão por Mínimos Quadrados Ponderados (WLS) para examinar a relação entre o spread e a captação líquida. O código python inicia com a importação da biblioteca `statsmodels.api` que fornece os métodos para estimação do modelo WLS e funções da biblioteca `scipy.stats`, como `pearsonr` e `spearmanr`, empregadas na medição da correlação linear e não linear entre as variáveis, e o teste de raiz unitária adfuller, proveniente de `statsmodels.tsa.stattools`, utilizado para avaliar a estacionariedade das séries. A função `uniform_filter1d` é incluída para permitir a suavização de dados em análises complementares.

A função principal, denominada `analisedefinitiva()`, constitui o núcleo metodológico do modelo *Weighted Least Squares* (WLS) e consolida, em uma única estrutura, as etapas de preparação dos dados, cálculo estatístico e ajuste econométrico. Ela é projetada para processar a base consolidada que contém as variáveis *Captação Líquida (bi)* e *Spread ponderado (%)*, aplicando filtros, transformações e testes estatísticos que antecedem a regressão ponderada. A execução da função se inicia com uma série de instruções de abertura que imprimem no console o título e a formatação da análise, demarcando o início do módulo.

A etapa de estatísticas descritivas tem como objetivo caracterizar o comportamento empírico das séries antes da modelagem econométrica. Para isso, é definida uma função interna denominada `calcularestatisticas()`, que recebe uma série numérica e retorna um conjunto de medidas resumo: média, mediana, desvio padrão, valores mínimo e máximo, amplitude, coeficiente de variação e tamanho da amostra. Os resultados são organizados em uma estrutura do tipo `Series` e posteriormente combinados em um `DataFrame` comparativo denominado `statscomparativo`, o qual permite visualizar, lado a lado, as propriedades descritivas das duas

variáveis de interesse.

3.2.1 Correlação com Defasagens

A Análise de Correlação com Defasagens, marca o ponto em que o modelo passa a investigar a dimensão temporal da relação entre o spread ponderado e a captação líquida, tendo como propósito identificar se existe um comportamento sistemático em que variações no spread antecedem, de forma estatisticamente significativa, os movimentos de entrada ou saída de recursos dos fundos. Em termos práticos, busca-se mensurar a sensibilidade dos fluxos de captação a mudanças no prêmio de risco do mercado de crédito, considerando defasagens de até doze períodos mensais.

O procedimento é implementado por meio da função interna `analisecorrelacao defasagens()`, que recebe como parâmetros o `DataFrame` principal, as variáveis de interesse (`spreadpct` e `fluxobi`) e o número máximo de defasagens (`maxlag`). Dentro da função, um laço iterativo percorre sucessivamente valores de defasagem entre zero e doze. Em cada iteração, a variável explicativa, no caso o spread, é deslocada no tempo, permitindo que o valor corrente da captação líquida seja comparado com o spread observado em meses anteriores. Essa estrutura de análise temporal torna possível avaliar se o comportamento do mercado de crédito, refletido nos spreads, exerce efeito antecedente sobre os fluxos de investimento.

Para cada defasagem testada, o código calcula dois tipos de correlação: o coeficiente de *Pearson*, que mede a intensidade da relação linear entre as variáveis, e o coeficiente de *Spearman*. Ambos os coeficientes são acompanhados de seus respectivos valores-p, que indicam o nível de significância estatística. Cada resultado é armazenado em uma lista contendo o número da defasagem, os coeficientes, os valores-p, o tamanho amostral (n) e um marcador que sinaliza se a correlação de Pearson é estatisticamente significativa ao nível de 5%.

Após a execução da função, os resultados são reunidos em um `DataFrame` denominado `corrspreadfluxo`, e uma cópia formatada (`corrspreadfluxodisplay`) é gerada para exibição. Nessa versão de saída, os valores-p são padronizados pela função `formatarpvalor()` e a tabela resultante é então impressa no console. O código também identifica automaticamente o *melhor lag*, ou seja, o número de períodos em que a correlação entre o spread e a captação líquida é mais intensa em termos absolutos. Essa identificação é realizada pela função `idxmax()`, que localiza o valor máximo do coeficiente de correlação de Pearson em módulo.

3.2.2 Dessazonalização das Séries Temporais

A dessazonalização das séries temporais tem como objetivo eliminar padrões sazonais recorrentes das variáveis analisadas de modo a preservar apenas suas componentes estruturais e cíclicas. A execução do processo inicia-se com a criação de uma variável auxiliar denominada `mes`, obtida a partir da extração do mês de referência de cada observação da base de dados.

Essa variável é utilizada como componente explicativa no modelo de regressão que realizará a dessazonalização. Em seguida, o código define a função `dessazonalizarserie()`, responsável por aplicar uma regressão de Mínimos Quadrados Ordinários (OLS), modelando a influência dos efeitos mensais sobre a variável em análise. Para isso, os meses são transformados em variáveis *dummies* (colunas binárias que indicam a presença ou ausência de cada mês), com a exclusão da primeira categoria (`dropfirst=True`) para evitar o problema de colinearidade perfeita.

A função concatena a série original com a matriz de dummies, elimina eventuais valores ausentes e ajusta o modelo OLS por meio do módulo `statsmodels`. Os resíduos obtidos dessa regressão, que representam a porção da série não explicada pelos efeitos mensais, são extraídos e armazenados como a versão dessazonalizada da variável. Dessa forma, o procedimento remove sistematicamente as flutuações previsíveis associadas ao calendário, preservando apenas as variações genuínas relacionadas a fatores econômicos, financeiros ou estruturais.

Após, a função de dessazonalização é aplicada de maneira independente às duas variáveis principais do estudo: `fluxobi` (captação líquida) e `spreadpct` (spread ponderado). Os resíduos resultantes são armazenados nas colunas `fluxores` e `spreadres`, respectivamente. Com isso, cada uma das séries passa a dispor de uma versão ajustada para sazonalidade, isenta de oscilações repetitivas e não informativas. Essa etapa é essencial para garantir que as análises de correlação e o modelo WLS operem sobre séries temporalmente comparáveis e estatisticamente limpas, permitindo inferências mais precisas.

É realizada uma comparação pré e pós dessazonalização, que tem como objetivo avaliar quantitativamente o impacto do processo de dessazonalização sobre a relação estatística. O código realiza duas mensurações paralelas, sendo que a primeira calcula a correlação contemporânea entre o *spread ponderado (%)* e a *captação líquida (bi)* em suas formas originais, isto é, sem qualquer ajuste sazonal. Para isso, o script cria um subconjunto de dados (`temporiginal`) e aplica o coeficiente de correlação de *Pearson*, retornando tanto o valor da correlação (`rantes`) quanto o respectivo valor-*p* (`pantes`), que indica o nível de significância estatística dessa relação.

Na sequência, é conduzida a segunda mensuração, desta vez utilizando as versões dessazonalizadas das variáveis, ou seja, os resíduos obtidos na etapa anterior, armazenados nas colunas `spreadres` e `fluxores`. O cálculo da correlação é realizado pelo mesmo método, produzindo os valores `rdepois` e `pdepois`. Essa comparação direta permite observar se a remoção dos efeitos mensais gerou uma relação mais consistente e estatisticamente robusta entre as variáveis centrais do modelo.

Os valores-*p* obtidos em ambas as mensurações são padronizados pela função `formatar pvalor()` e em seguida, o código calcula uma métrica adicional chamada *melhoria de magnitude*, que quantifica a variação percentual na força da correlação após o ajuste sazonal. Essa métrica é obtida pela comparação entre os valores absolutos de `rantes` e `rdepois`, fornecendo uma medida objetiva da intensificação (ou redução) da associação entre as séries após a dessazonalização.

Após, realizou-se a análise de correlação das séries dessazonalidas, para isso o código reaplica a rotina de correlação com defasagens à dupla de resíduos obtidos na etapa anterior. Esse reaproveitamento do procedimento garante comparabilidade metodológica em relação ao exercício realizado com as séries brutas, ao mesmo tempo em que concentra a inferência na componente econômica propriamente dita. A função `analise_correlacao_defasagens()` é chamada agora tendo `spread_res` como variável antecedente e `fluxo_res` como variável resposta, varrendo lags mensais de zero até doze. Em cada defasagem, são calculados os coeficientes de Pearson e Spearman e seus respectivos valores-p, os quais são padronizados por meio de `formatar_p_valor()` para exibição. Na sequência, o código identifica o *melhor lag* como aquele que maximiza, em termos absolutos, o coeficiente de Pearson; o índice correspondente é recuperado, os principais números são impressos e o p-valor do lag ótimo é exibido com a formatação científica adequada quando necessário. O inteiro `lag_otimo` é então registrado para uso posterior na modelagem econométrica.

3.2.3 Modelagem Mínimos Quadrados Ponderados (WLS)

A modelagem econométrica na qual é implementada a regressão por *Mínimos Quadrados Ponderados* (WLS), técnica escolhida para capturar a relação entre o *spread ponderado* e a *captação líquida* após o tratamento das defasagens e a dessazonalização das séries. O processo inicia-se com a preparação da base de dados para estimação. O código cria uma cópia do `DataFrame` principal e insere duas novas variáveis defasadas: a primeira, `spreadreslagX`, corresponde à série de spreads dessazonalizados deslocada no tempo de acordo com o *lag ótimo* identificado na etapa anterior; a segunda, `fluxoreslag1`, representa a defasagem de um período da própria captação líquida dessazonalizada, incorporando a dependência temporal da variável explicada. Essa estrutura permite que o modelo capture tanto os efeitos de inércia da captação quanto o impacto defasado do spread, refletindo de forma mais realista o comportamento dinâmico dos fluxos financeiros.

Em seguida, são definidas as variáveis do modelo: `X` contém as variáveis explicativas, acrescidas de uma constante com o uso da função `addconstant()` do módulo `statsmodels`, enquanto y representa a variável dependente, a captação líquida dessazonalizada. A estimação ponderada é realizada em duas etapas. Primeiramente, executa-se uma regressão preliminar por *Mínimos Quadrados Ordinários* (OLS), utilizada apenas para obter os resíduos absolutos e, a partir deles, derivar os pesos do modelo WLS. Essa abordagem é necessária porque o método WLS exige uma estrutura de ponderação que compense a heterocedasticidade, isto é, a variância não constante dos erros, típica de séries financeiras.

Os resíduos absolutos do modelo OLS são suavizados por meio de um filtro móvel (`uniformfilter1d`) com janela de três períodos, o que reduz oscilações abruptas e estabiliza a escala dos pesos. Em seguida, os pesos são calculados como o inverso do quadrado da variância suavizada, atribuindo menor peso às observações com maior volatilidade residual.

Essa ponderação corrige o viés causado pela heterocedasticidade e melhora a eficiência das estimativas.

Com os pesos definidos, o modelo WLS final é ajustado utilizando a função `sm.WLS()` da biblioteca `statsmodels`. O resultado é uma regressão ponderada robusta, em que os coeficientes estimados expressam o impacto do spread defasado e da captação passada sobre a variação contemporânea da captação líquida, considerando explicitamente a estrutura de volatilidade dos dados.

No segundo momento o código analisa a performance dos resultados com objetivo validar a consistência estatística dos parâmetros obtidos pelo modelo WLS. O código inicia com a estimação de três modelos complementares: o modelo *OLS* padrão, o modelo *OLS* com correção *HAC* (Heteroskedasticity and Autocorrelation Consistent) e o modelo *RLM* (Robust Linear Model). O primeiro segue a formulação tradicional dos Mínimos Quadrados Ordinários, funcionando como referência base. O segundo aplica a correção de Newey–West (*HAC*) com até três defasagens, ajustando a matriz de covariância para lidar simultaneamente com heterocedasticidade e autocorrelação nos resíduos. O terceiro modelo, baseado na estimação robusta por *M-estimation* com função de perda Huber–T, é menos sensível a valores extremos e outliers, sendo amplamente utilizado em regressões robustas aplicadas a dados de alta volatilidade. Após as estimações, os parâmetros e métricas de cada modelo são reunidos em uma tabela comparativa denominada `comparacaomodelos`, que contém o coeficiente de determinação (R^2), os coeficientes estimados das variáveis (*Constante*, *AR1* e *Spread defasado*) e seus respectivos valores-p.

3.3 Gráfico Evolução do Spread Ponderado

Realizou-se o código apresentado no Apêndice D para gerar o gráfico da evolução do spread ponderado. O script gera o gráfico principal da série temporal, onde o código ajusta os parâmetros visuais no `matplotlib`, configurando dimensões, tipografia e espessura das linhas. A série de spreads ponderados é plotada em um único eixo com marcação temporal dinâmica.

3.4 Machine Learning do Spread Puro

O módulo de machine learning do spread puro descreve o desenvolvimento de um pipeline voltado à modelagem preditiva da série de spreads ponderados do Idex–CDI, sem incluir variáveis macroeconômicas ou de captação como explicativas. O objetivo central é avaliar a capacidade de algoritmos de aprendizado supervisionado em capturar padrões temporais e estruturais intrínsecos ao comportamento do prêmio de risco do crédito corporativo, utilizando apenas as propriedades estatísticas da própria série e de suas transformações. O código apresentado no Apêndice E é estruturado em três etapas: pré-processamento, modelagem e avaliação preditiva.

O modelo de previsão é composto por três etapas principais. Na primeira, é treinado um modelo Elastic Net utilizando apenas defasagens do spread, constituindo o benchmark autorregressivo. Na segunda etapa, outro modelo Elastic Net é ajustado sobre variáveis micro derivadas da dinâmica local da série, com o objetivo de corrigir os resíduos do benchmark e gerar um nowcast mais refinado. Por fim, a terceira etapa realiza a previsão de múltiplos horizontes utilizando um modelo de gradient boosting (HistGradientBoostingRegressor), cuja saída é estabilizada por técnicas de clipping e calibrada linearmente. Todo o processo é conduzido de forma temporalmente consistente, evitando qualquer tipo de vazamento de informação (look-ahead bias).

O pré-processamento é realizado conforme os códigos anteriores. Na sequência, são construídas variáveis microestruturais por meio da função `create_micro_features`, que gera indicadores de volatilidade, médias móveis e tendências recentes do spread. Essas variáveis capturam características importantes da dinâmica temporal, como persistência e reversões de curto prazo. Complementarmente, a função `create_lagged_features` gera defasagens tanto do próprio spread quanto dessas variáveis micro, o que irá compor o conjunto de informações passadas, as quais servirão como base para os modelos subsequentes.

A primeira etapa do modelo corresponde ao componente benchmark autorregressivo, cujo objetivo é capturar a parte persistente e sistematicamente previsível da dinâmica do spread. Para isso, selecionam-se exclusivamente as variáveis de defasagem do próprio spread, isto é, colunas do tipo `spread_wavg_lagk`, que representam o valor do spread observado k dias antes. Essa seleção é feita a partir do conjunto total de *features* construídas, filtrando apenas aquelas cujo nome inicia com `spread_wavg_lag`. Dessa forma, o modelo utiliza unicamente a própria memória temporal da série, sem incorporar ainda informações microestruturais.

A estimação desse benchmark é realizada por meio de um pipeline composto por duas etapas sequenciais: um `StandardScaler` e um `ElasticNetCV`. O `StandardScaler` padroniza cada variável explicativa removendo a média e normalizando pelo desvio padrão calculado exclusivamente sobre o conjunto de treino, garantindo que todas as defasagens estejam na mesma escala. Esse procedimento é particularmente importante porque o `ElasticNetCV` utiliza penalizações nos coeficientes, e tais penalizações seriam distorcidas se as variáveis tivessem ordens de grandeza diferentes.

Em seguida, o `ElasticNetCV` realiza a regressão penalizada, combinando termos de regularização L1 (Lasso) e L2 (Ridge) e explorando diferentes valores de `l1_ratio` por meio de validação cruzada. A escolha automática do nível ótimo de penalização torna o modelo mais robusto, evitando sobreajuste e permitindo selecionar apenas as defasagens realmente relevantes para explicar o spread. O modelo resultante constitui, assim, uma forma regularizada de autorregressão, capaz de capturar estrutura temporal de baixa frequência, como persistência e inércia da série.

Após o treinamento, o pipeline é utilizado para gerar a previsão do spread no conjunto

de treino. Essa previsão representa a componente lenta ou estrutural da dinâmica diária. A diferença entre o valor observado e o valor previsto pelo benchmark resulta no vetor de resíduos, correspondendo exatamente à parcela do spread que o mecanismo autorregressivo não conseguiu explicar. Esses resíduos passam a ser o alvo da segunda etapa do modelo, na qual são exploradas variáveis microestruturais para capturar movimentos de curto prazo e flutuações mais rápidas que escapam ao componente autorregressivo.

A segunda etapa do modelo tem como objetivo refinar a previsão obtida pelo componente autorregressivo, concentrando-se agora na parcela do spread que não foi explicada pelas defasagens do próprio índice. Para isso, os resíduos gerados na etapa anterior, isto é, a diferença entre o spread observado e a previsão do benchmark, passam a constituir o novo alvo de modelagem. Esses resíduos representam movimentos de mais alta frequência, oscilações abruptas e comportamentos não lineares que a estrutura autorregressiva regularizada não consegue capturar. Assim, a segunda etapa atua como um mecanismo de correção, voltado exclusivamente para explicar essa fração remanescente da variabilidade do spread.

A modelagem dessa correção é realizada por meio de um novo pipeline composto por um `StandardScaler` e um `ElasticNetCV`. Contudo, diferentemente da primeira etapa, o conjunto de regressoras não inclui defasagens do próprio spread. Nessa etapa, utilizam-se apenas variáveis microestruturais derivadas da dinâmica local da série, como medidas de volatilidade em janelas curtas, médias móveis, indicadores de tendência e defasagens dessas variáveis. Em termos de código, essas *features* são identificadas como todas aquelas que não começam com `spread_wavg_lag`, o que garante a separação explícita entre a parte autorregressiva e a parte microestrutural. O `StandardScaler` padroniza essas variáveis para torná-las comparáveis entre si, enquanto o `ElasticNetCV` aplica a penalização combinada L1 e L2 para selecionar as microvariáveis mais relevantes e evitar sobreajuste, especialmente porque tais variáveis frequentemente apresentam alta correlação entre si.

Após o treinamento desse segundo modelo, gera-se a previsão dos resíduos em cada instante. Essa previsão representa exatamente a fração rápida ou local do movimento do spread. A etapa é então concluída somando-se as previsões da primeira e da segunda etapa: o valor previsto pelo componente autorregressivo mais a correção microestrutural estimada pelo Elastic Net. O resultado dessa soma consiste no *nowcast* final do spread, que sintetiza tanto a dinâmica de longo prazo quanto as flutuações de curto prazo. Esse *nowcast* cumpre dois papéis fundamentais no pipeline: primeiro, é usado como referência para avaliar o desempenho das etapas iniciais; segundo, funciona como meta-*feature* essencial na etapa seguinte, que prevê o spread a múltiplos horizontes à frente.

A Etapa 3 é responsável por transformar o *nowcast* diário em previsões de múltiplos horizontes à frente. Em termos de implementação, essa etapa está encapsulada na função `etapa3_simplificada`. Ela recebe como entrada os subconjuntos temporais `train`, `valid` e `test`, o DataFrame completo `data`, o alvo `target = "spread_wavg"`, os modelos previamente

ajustados `model_benchmark` e `model_micro`, e a lista de horizontes definida em `HORIZONTES = (5, 22, 66)`. Cada horizonte H representa um número de dias úteis à frente para os quais se deseja prever a variação do spread, formalmente dada pela Equação 3.1:

$$\Delta = y_{t+H} - y_t \tag{3.1}$$

É aplicado um filtro por `DATA_INICIAL` para garantir o número mínimo de observações e o script define três janelas cronológicas não sobrepostas: treino (aproximadamente 60% das observações, validação (cerca de 20%) e teste (os 20% finais). Dessa forma, toda estimação de parâmetros e seleção de modelos utiliza apenas informações anteriores às fases de validação e teste.

O primeiro passo consiste na construção de uma meta-feature que sintetiza o *nowcast* diário produzido nas etapas iniciais. Essa operação é realizada pela função interna `add_nowcast(df)` somando ambas as previsões para gerar a coluna `nowcast_final`. Esse procedimento é aplicado de forma coerente às janelas `train_e3`, `valid_e3` e `test_e3`. Em seguida, define-se um conjunto inicial de variáveis explicativas, denominado `base_pool`, composto por todas as colunas de `data`, exceto o alvo e quaisquer colunas cujo nome termine com `_futuro`. Esse conjunto constitui o *pool* a partir do qual a seleção de variáveis será realizada para cada horizonte.

Para cada horizonte $H \in \{5, 22, 66\}$, a função constrói a variável-alvo como a diferença futura do spread. Isso é feito pela função interna `build_delta(df, H, feat_cols)`, que calcula a matriz de regressoras `X` contendo apenas `feat_cols`, o vetor de deltas Δ, o vetor `y_t` e o vetor `y_true` correspondente ao nível futuro y_{t+H}. Esse procedimento é aplicado separadamente às janelas de treino (`train_e3`), validação (`valid_e3`) e teste (`test_e3`), de modo que somente instantes com informação completa de y_t e y_{t+H} são mantidos.

A seleção de variáveis para cada horizonte é realizada exclusivamente com base no conjunto de treino, via `select_by_spearman(X_tr, y_trΔ, k)`. Essa função computa a correlação de Spearman entre cada coluna de `X_tr` e o alvo `y_trΔ`, ordena as variáveis pelo valor absoluto dessa correlação e seleciona as `top-k`. Assim, cada horizonte possui um conjunto próprio de variáveis com maior associação ao delta futuro.

Uma vez selecionadas as variáveis relevantes, a função utiliza `build_delta` para gerar os conjuntos de treino, validação e teste específicos de cada horizonte. Sobre `X_tr` e $y_{\text{tr}\Delta}$, ajusta-se um modelo `HistGradientBoostingRegressor`, cuja configuração é escolhida dentre os conjuntos de hiperparâmetros definidos em `grid_by_h`, que incluem taxa de aprendizado, profundidade das árvores, tamanho mínimo de folha, regularização L2 e a função de perda apropriada ao horizonte considerado. Para cada combinação dessa grade e para cada quantil de *clipping* em `q_grid`, o modelo é treinado no conjunto de treino e subsequentemente avaliado no conjunto de validação.

A validação envolve um processo obtenção de previsão dos deltas em validação. Em

seguida, calcula-se o módulo dos deltas observados no treino (`absΔ`), e para cada quantil q determina-se um limite Δ_{cap} como o quantil q de `absΔ`. As previsões de delta em validação são então truncadas para o intervalo $[-\Delta_{\text{cap}}, \Delta_{\text{cap}}]$. Depois disso, reconstrói-se a previsão de nível futuro conforme a Equação 3.2

$$y\_va\_hat = y\_va\_t + d\_va \tag{3.2}$$

em que `y_va_t` é o spread observado no tempo t e `d_va` são os deltas previstos após *clipping*.

Para corrigir eventuais vieses sistemáticos entre `y_va_hat` e o valor observado `y_va_true`, aplica-se uma calibração linear por meio de `LinearRegression`. Quando há observações suficientes, estima-se uma regressão simples usando o conjunto de validação. A composição modelo de `HistGradientBoostingRegressor` + *clipping* + calibração linear é avaliada pelo erro absoluto médio (MAE). Para cada horizonte, a combinação que apresenta o menor MAE é selecionada como configuração final, seguindo integralmente o fluxo treino–validação, mantendo o conjunto de teste completamente intocado.

Com o modelo e os parâmetros selecionados, a função aplica a mesma estrutura ao conjunto de teste: calcula a previsão de deltas `d_te`, realiza o mesmo *clipping*, reconstrói o nível futuro e aplica, quando disponível, a calibração linear para obter `y_te_pred`. Em seguida, calcula-se o desempenho via `metrics(y, yhat)`, que devolve MAE, RMSE, R^2, correlação de Pearson e correlação de Spearman. Os erros são convertidos para pontos-base multiplicando-se por 10^4. Todos esses resultados, incluindo o número de observações efetivas em treino, validação e teste, são organizados em um `DataFrame` de resumo.

Por fim, são construídas séries temporais específicas para o conjunto de teste, armazenando, para cada horizonte, as trajetórias de valores realizados e previstos, posteriormente utilizadas na geração de gráficos comparativos entre nível futuro do spread e previsão para cada horizonte considerado.

3.5 Machine Learning do Spread e Captação Líquida

O machine learning do spread considerando a captação líquida tem como o objetivo de integrar o comportamento agregado de liquidez dos fundos de crédito privado à análise dos prêmios de risco corporativos e o código apresenta-se no Apêndice F. A estrutura geral do pipeline permanece a mesma do modelo desenvolvido inicialmente para previsão baseada apenas no spread: os dados passam pelas mesmas três etapas principais, construção do benchmark autorregressivo, modelagem da correção por regressão penalizada e previsão a múltiplos horizontes por meio de *gradient boosting* calibrado. O que se altera, contudo, é a incorporação sistemática de informações macroeconômicas de captação líquida da indústria, que enriquecem a segunda e

a terceira etapas do processo e ampliam o conjunto de determinantes considerados no modelo.

A lógica das três etapas de modelagem é preservada, mas com ajustes. Na Etapa 1, continuam sendo utilizadas apenas as defasagens do próprio spread dentro de um `Pipeline` composto por `StandardScaler` e `ElasticNetCV`, reproduzindo o benchmark autorregressivo do modelo anterior. A incorporação das variáveis macroeconômicas de captação ao modelo ocorre com a base diária unificada e a função `create_lagged_features` é responsável pela construção das defasagens relevantes. Além dos lags do alvo, `spread_wavg_lagk`, essa função identifica todas as colunas macro e gera no mínimo um lag diário. Para a variável de captação líquida, são ainda criados lags aproximados mensais constituídos pelos deslocamentos equivalentes a múltiplos de aproximadamente 22 dias úteis. A base final de modelagem passa, assim, a conter defasagens do spread, das variáveis microestruturais e das variáveis macroeconômicas de captação.

A mudança na Etapa 2 decorre da ampliação do conjunto de regressoras. No modelo de spread puro, a segunda etapa utilizava apenas variáveis microestruturais, definidas no código como todas aquelas que não eram lags de `spread_wavg`. Na versão estendida, o mesmo critério de exclusão é mantido, mas o conjunto `features_macro_model` passa a incluir simultaneamente as variáveis micro e todas as variáveis macro e seus lags. Em termos operacionais, isso significa que a Etapa 2 continua a ser treinada sobre os resíduos do benchmark autorregressivo, sendo a diferença entre o spread observado e a previsão da Etapa 1, mas agora utilizando como regressoras um bloco ampliado de informações contendo medidas de volatilidade, médias móveis e tendências do spread, somadas aos indicadores de captação líquida, suas tendências, seus regimes e seus lags diários e mensais.

Do ponto de vista computacional, a arquitetura permanece idêntica àquela empregada no modelo de spread puro. A Etapa 2 continua a utilizar um `Pipeline` composto por `StandardScaler` e `ElasticNetCV`, agora aplicado ao vetor de resíduos e ao conjunto ampliado `features_macro_model`. O `StandardScaler` normaliza todas as variáveis micro e macro para uma escala comparável, enquanto o `ElasticNetCV` realiza simultaneamente seleção e regularização dos coeficientes, determinando automaticamente o nível ótimo de penalização por meio de validação cruzada.

Na Etapa 3, o procedimento de previsão por horizonte também é mantido, mas passa a utilizar como entrada um *nowcast* que já incorpora informações macroeconômicas. O restante da etapa, com a estimação dos deltas com `HistGradientBoostingRegressor`, aplicação de *clipping* baseado em quantis, calibração linear em validação, avaliação em teste e cálculo de importâncias por permutação, seguem integralmente o método estabelecido no modelo para previsão baseada apenas no spread.

4 Resultados

4.1 Relação entre Spread Ponderado e Captação Líquida dos Fundos de Crédito

A análise da relação entre o spread ponderado do índice Idex–CDI e a captação líquida mensal dos fundos de crédito privado representa um o cerne deste estudo; e compreender essa dinâmica é fundamental para identificar como as oscilações nos prêmios de risco influenciam os fluxos de recursos e refletem a percepção de risco dos investidores institucionais.

A Figura 1 apresenta a evolução conjunta dessas duas variáveis entre abril de 2022 e agosto de 2025. Uma primeira análise visual revela um padrão no qual os movimentos das séries apresentam trajetórias opostas em diversos momentos do período analisado. Quando os spreads se elevam, a captação líquida tende a diminuir ou até mesmo tornar-se negativa, indicando resgates líquidos. Por outro lado, em períodos de compressão dos spreads, observam-se captações positivas mais robustas.

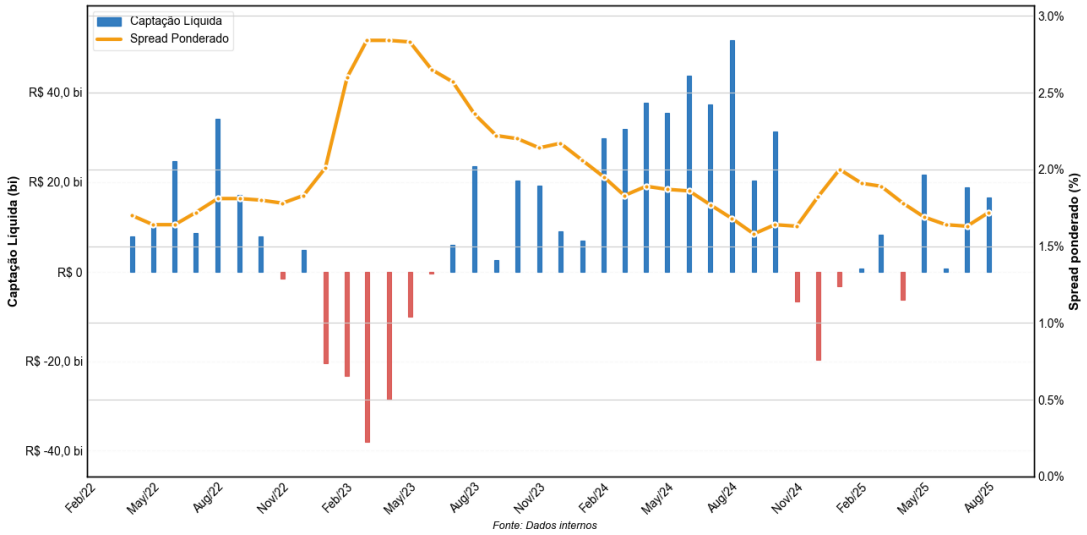

Figura 1 – Evolução conjunta do spread ponderado (%) e do fluxo de captação líquida (R$ bilhões) entre 2022 e 2025.

Fonte: Elaboração própria, a partir de dados da JGP e CVM.

Esse comportamento fica particularmente evidente em três momentos distintos. No início de 2023, o spread ponderado atingiu seu pico histórico, aproximando-se de 2,9%, justamente quando a captação líquida registrou os maiores volumes de resgates, chegando a valores negativos superiores a R$ 30 bilhões. Em contraste, durante o segundo semestre de 2024, quando os spreads recuaram para patamares próximos a 1,7%, a captação líquida alcançou máximos históricos,

superando R$ 40 bilhões em alguns meses. Mais recentemente, no início de 2025, observa-se uma nova elevação dos spreads acompanhada de redução nos fluxos de captação.

Essa relação inversa sugere que os investidores institucionais demonstram maior aversão ao risco em momentos de ampliação dos spreads. Spreads elevados podem sinalizar deterioração da qualidade do crédito ou aumento da incerteza macroeconômica, levando os gestores a adotarem postura mais defensiva e reduzirem suas exposições. Por outro lado, a compressão dos spreads parece estar associada a períodos de maior confiança no mercado de crédito privado, estimulando novas captações.

Para quantificar essa relação de forma mais rigorosa, foi implementado o Código C, que calcula medidas de correlação com diferentes defasagens temporais e estima a variância explicada (R^2) por meio de modelos lineares. Assim identificando não apenas a direção da relação entre as variáveis, mas também avalia se existe algum padrão de antecedência temporal; ou seja, se variações nos spreads precedem alterações nos fluxos de captação ou vice-versa.

Os resultados apresentados a seguir combinam estatísticas descritivas, correlações defasadas e estimativas obtidas por Mínimos Quadrados Ponderados (WLS). Essa estrutura de análise foi escolhida por permitir uma interpretação abrangente da dinâmica entre spreads e captação, contemplando tanto a intensidade quanto a temporalidade dos efeitos observados ao longo do período estudado.

4.1.1 Características das Séries e Procedimentos de Estimação

A Tabela 1 apresenta as estatísticas descritivas das séries utilizadas na análise. Os dados revelam comportamentos distintos entre as duas variáveis. A captação líquida apresentou média de R$ 10,476 bilhões no período, com desvio-padrão de 20,187 bilhões. Já o spread ponderado registrou média de 1,976%, com desvio-padrão de 0,364 pontos percentuais.

Tabela 1 – Estatísticas descritivas das séries originais

	Captação Líquida (R$ bi)	Spread Ponderado (%)
Média	10,476	1,976
Mediana	8,557	1,830
Desvio-padrão	20,187	0,364
Mínimo	-38,015	1,580
Máximo	51,643	2,840
Amplitude	89,658	1,260
Coef. de variação (%)	192,693	18,409
n	41	41

Fonte: Elaboração própria.

Um aspecto que chama atenção é a alta variabilidade da captação líquida, evidenciada por seu coeficiente de variação de 192,7%. Esse valor expressa uma dispersão relativa muito

superior à observada no spread ponderado, que apresentou coeficiente de variação de apenas 18,4%. Essa diferença é esperada e reflete a natureza mais volátil dos fluxos de investimento, que respondem não apenas aos prêmios de risco, mas também a fatores como episódios de estresse de liquidez, mudanças regulatórias, ciclos macroeconômicos e fatores não-sistemáticos de cada fundo.

A amplitude das séries também merece destaque. Enquanto a captação líquida variou de resgates de R$ 38 bilhões até captações de R$ 51,6 bilhões, o que representa uma amplitude de quase R$ 90 bilhões, o spread ponderado oscilou entre 1,58% e 2,84%, permanecendo dentro de um intervalo relativamente estreito de 1,26 pontos percentuais. Essa assimetria reforça a interpretação de que pequenas variações nos spreads podem estar associadas a grandes movimentos nos fluxos de capital.

Antes de prosseguir com a análise de correlação, foi necessário verificar as propriedades estatísticas das séries. Os testes de Dickey-Fuller Aumentado (ADF) indicaram que nenhuma das séries é estacionária aos níveis usuais de significância. Para a captação líquida, o teste resultou em $p = 0,179$, e para o spread ponderado, $p = 0,148$. Esses resultados apontam para a presença de raiz unitária em ambas as séries, o que poderia comprometer a validade das estimativas de correlação e dos modelos de regressão caso não fossem adotados procedimentos adequados.

Diante dessa constatação, optou-se por trabalhar com especificações que consideram defasagens temporais e, quando necessário, aplicar transformações para remover tendências e sazonalidades. Dessa maneira, com as características das séries devidamente mapeadas, a próxima etapa consiste em explorar a estrutura temporal das correlações.

O procedimento de correlação com defasagens foi operacionalizado de maneira sistemática no Código C, mediante o deslocamento sucessivo da série de spreads em $k \in \{0, \ldots, 12\}$ períodos. Nesse sentido, os resultados são apresentados pela Tabela 2 e revela um comportamento progressivo das correlações ao longo do tempo. Observa-se uma correlação negativa moderada ($r = -0,5277$; $p < 0,001$), indicando que elevações no spread estão associadas, de forma imediata, à redução dos fluxos líquidos de captação. Entretanto, à medida que o intervalo de defasagem aumenta, verifica-se uma inversão gradual do sinal e um fortalecimento das correlações positivas, atingindo seu valor máximo em 12 períodos ($r = 0,6406$; $p = 0,000182$).

Assim, o resultado sugere que a variação nos spreads antecede a resposta dos fluxos da captação de recursos em aproximadamente um ano. Nesse âmbito , a ampliação do spread está correlacionada com o aumento da aversão ao risco por parte dos investidores. Esse movimento em direção a uma projeção mais pessimista por parte do mercado se reflete, em um primeiro momento, em uma captação mais tímida da classe de ativos de crédito. Mas o ajuste da percepção de risco na economia parece, ainda assim, converter, em um segundo momento, os fundos de crédito privado como oportunidades do mercado.

Os resultados da dessazonalização das séries, apresentados na Tabela 3 mostram maior

Tabela 2 – Correlação entre Spread e Captação por Defasagem (séries originais)

Lag	Pearson	p (Pearson)	Spearman	p (Spearman)	n	Sig.
0	-0,5277	0,000392	-0,3609	0,020	41	Sim
1	-0,3635	0,021	-0,2312	0,151	40	Sim
2	-0,1499	0,362	-0,0113	0,945	39	Não
3	0,0475	0,777	0,1494	0,371	38	Não
4	0,1901	0,260	0,3388	0,040	37	Não
5	0,2949	0,081	0,4797	0,0031	36	Não
6	0,3623	0,032	0,5193	0,0014	35	Sim
7	0,4101	0,016	0,5552	0,000654	34	Sim
8	0,4697	0,0058	0,5530	0,000846	33	Sim
9	0,5226	0,0022	0,5773	0,000541	32	Sim
10	0,5531	0,0013	0,5861	0,000531	31	Sim
11	0,5887	0,000621	0,5936	0,000545	30	Sim
12	0,6406	0,000182	0,5784	0,0010	29	Sim

precisão estatística e coerência estrutural na relação entre as variáveis. Após a remoção dos componentes sazonais, a correlação contemporânea passou de $r = -0{,}5277$ ($p = 0{,}000392$) para $r = -0{,}5498$ ($p = 0{,}000197$), indicando que parte dos movimentos previamente capturados refletia padrões sazonais e não propriamente interações econômicas. No conjunto dessazonalizado, a matriz de defasagens apresentou melhoria substancial, com um coeficiente máximo em dez períodos nos valores de $r = 0{,}8286$ ($p = 8{,}61 \times 10^{-9}$) e 31 observações válidas. Reforçando a tese de que a remoção de efeitos sazonais amplia a capacidade explicativa do spread sobre a captação líquida.

Tabela 3 – Correlação entre Spread e Captação por Defasagem (séries dessazonalizadas)

Lag	Pearson	p (Pearson)	Spearman	p (Spearman)	n	Sig.
0	-0,5498	0,000197	-0,4255	0,0056	41	Sim
1	-0,4017	0,010	-0,2744	0,087	40	Sim
2	-0,2245	0,169	-0,1435	0,383	39	Não
3	-0,0457	0,785	-0,0189	0,910	38	Não
4	0,0946	0,578	0,0721	0,672	37	Não
5	0,2160	0,206	0,1315	0,444	36	Não
6	0,3713	0,028	0,2807	0,102	35	Sim
7	0,5012	0,0025	0,3986	0,020	34	Sim
8	0,6630	$2{,}62 \times 10^{-5}$	0,5819	0,000382	33	Sim
9	0,8177	$1{,}11 \times 10^{-8}$	0,8050	$2{,}79 \times 10^{-8}$	32	Sim
10	0,8286	$8{,}61 \times 10^{-9}$	0,8097	$3{,}46 \times 10^{-8}$	31	Sim
11	0,8182	$3{,}34 \times 10^{-8}$	0,7700	$6{,}53 \times 10^{-7}$	30	Sim
12	0,7569	$2{,}02 \times 10^{-6}$	0,7320	$6{,}39 \times 10^{-6}$	29	Sim

Com base nesses resultados, procedeu-se ao ajuste do modelo final utilizando o método dos Mínimos Quadrados Ponderados (WLS). Visando lidar com sua capacidade com possíveis

problemas de heterocedasticidade presentes nas séries financeiras, atribuindo pesos diferenciados às observações de acordo com sua variabilidade. A especificação incorpora dois elementos centrais: um termo autorregressivo de primeira ordem para capturar a persistência dos fluxos de captação, e o spread ponderado dessazonalizado defasado em 10 períodos, conforme identificado na análise de correlação como a defasagem que maximiza a relação entre as variáveis.

O modelo estimado é formalmente apresentado na Equação 4.1:

$$\text{Captação}_t = \alpha + \phi\,\text{Captação}_{t-1} + \beta\,\text{Spread}_{t-10} + \varepsilon_t \tag{4.1}$$

onde Captação_t representa a captação líquida no mês t; Captação_{t-1} é a captação líquida defasada em um período, responsável por capturar a persistência dos fluxos; Spread_{t-10} corresponde ao spread ponderado dessazonalizado defasado em 10 meses; α é o intercepto; ϕ e β são os coeficientes a serem estimados; e ε_t é o termo de erro aleatório.

Essa especificação permite avaliar simultaneamente tanto a inércia dos fluxos de captação, capturada por ϕ, qunato o impacto defasado dos spreads sobre as decisões de alocação dos investidores, mensurado por β. A defasagem de 10 períodos reflete o tempo médio de resposta dos investidores institucionais às mudanças nas condições de precificação do crédito privado.

O uso de ponderações proporcionais ao inverso da variância condicional das observações reduz a influência de valores com maior dispersão, garantindo maior eficiência e consistência aos estimadores. A Tabela 4 apresenta os coeficientes estimados e as principais medidas do ajuste do modelo proposto.

Tabela 4 – Resultados do modelo WLS para captação líquida (t)

Variável	Coef.	Erro-padrão	p
Constante	-1,4848	1,450	0,315
Captação_{t-1}	0,4769	0,106	$< 0{,}001$
Spread_{t-10}	24,8039	5,577	$< 0{,}001$
R^2			0,914
R^2 ajustado			0,908
DW			2,565
n			31

O modelo apresentou elevado poder explicativo, evidenciado pelo coeficiente de determinação $R^2 = 0{,}914$ e pelo R^2 ajustado de 0,908, o que indica que mais de 90% da variação observada na captação líquida é explicada pelas variáveis defasadas incorporadas à regressão. O coeficiente autorregressivo positivo e estatisticamente significativo ($\phi = 0{,}4769$, $p < 0{,}001$) revela a presença de persistência temporal nos fluxos de captação, sugerindo que períodos de maior entrada de recursos tendem a ser seguidos por novos aportes.

O valor estatístico de Durbin–Watson (DW = 2,565) indica ausência de autocorrelação significativa nos resíduos, o que garante consistência interna do modelo e a validade das inferências realizadas. Esses resultados confirmam, portanto, a hipótese de que o spread ponderado do Idex–CDI exerce influência antecipada e economicamente relevante sobre a dinâmica dos fluxos líquidos dos fundos de crédito privado.

O parâmetro do spread ($\beta = 24{,}80$) permite uma interpretação econômica direta: um aumento de 1 ponto percentual no spread ponderado está associado, em média, a um acréscimo de aproximadamente R$ 24,8 bilhões na captação líquida após dez meses. A magnitude, a significância e a estabilidade desse coeficiente convergem com o mecanismo de transmissão teórico segundo o qual a elevação do prêmio de risco torna a classe de crédito privado relativamente mais atrativa, induzindo entradas líquidas de capital com certo intervalo temporal. Ademais, o coeficiente autorregressivo positivo ($\phi = 0{,}4769$) reforça a evidência de inércia nos fluxos, sugerindo que choques na captação não são totalmente dissipados, mas se propagam parcialmente ao longo do tempo.

Por conseguinte, comparando com outros estimadores, incluindo Mínimos Quadrados Ordinários (OLS) com correção de erros-padrão de Newey–West (HAC) e Regressão Robusta (RLM). Em todas as especificações testadas, o coeficiente do spread manteve sinal positivo, magnitude similar (variando entre 24,80 e 28,82) e níveis de significância compatíveis com a rejeição da hipótese nula ao nível de 1%.

Tabela 5 – Robustez: comparação entre WLS, OLS+HAC e RLM

Método	R^2	Constante	AR(1)	Coef. Spread (p)
WLS	0,914	-1,4848	0,4769	24,8039 (p=0,000125)
OLS+HAC	0,737	-1,5663	0,3864	26,6518 (p=0,000118)
RLM	—	-2,5256	0,3749	28,8151 (p=0,0018)

Os resultados representados na Tabela 5 evidenciam a estabilidade estrutural do modelo principal, caracterizada por variações marginais nos coeficientes e pela manutenção da significância estatística em todas as especificações testadas. O elevado valor de R^2 e a convergência entre diferentes estimadores evidenciam a capacidade explicativa substancial do modelo, ao passo que a interpretação econômica permanece coerente com o ciclo de realocação de portfólios e com o processo de precificação de risco característico do mercado de crédito.

4.2 Modelos de Previsão e Análise de Desempenho

Esta seção tem como objetivo apresentar os resultados da abordagem desenvolvida pelo Código E e avaliar em que medida o modelo é capaz de reproduzir a dinâmica recente dos spreads e de antecipar movimentos futuros em diferentes horizontes de tempo. Os resultados mostram que um modelo construído exclusivamente a partir da própria série histórica do spread

ponderado que por meio de defasagens, medidas de volatilidade e médias móveis da série é capaz de gerar previsões com bom nível de acurácia no curto prazo, ainda que seu desempenho se deteriore à medida que o horizonte de previsão se alonga. As principais métricas de desempenho em amostra de teste, expressas em pontos-base (bps), encontram-se sintetizadas na Tabela 6.

O parâmetro TopK $= 18$ indica que, em todos os horizontes, o modelo opera com um conjunto relativamente estável de variáveis explicativas selecionadas a partir de defasagens e *microfeatures* do spread. Já o parâmetro Δcap representa o limite máximo imposto para a variação prevista $\Delta_{t,H} = y_{t+H} - y_t$, e o aumento desse parâmetro de aproximadamente 8,4 para 79,1 pontos-base entre 5 e 66 dias reflete a maior amplitude típica dos movimentos de spread em horizontes mais longos e, ao mesmo tempo, atua como mecanismo de contenção de extrapolações nas previsões.

Tabela 6 – Desempenho preditivo do modelo de Spread por horizonte de previsão em dias úteis.

Hor. (dias)	n_{tr}	n_{val}	n_{te}	TopK	Δcap (bps)	MAE (bps)	RMSE (bps)	R^2	r	ρ
5	514	168	169	18	8,4461	2,4598	3,2441	0,9340	0,9744	0,9156
22	497	151	152	18	24,6773	6,1684	6,9892	0,5892	0,8720	0,8786
66	453	107	108	18	79,0972	9,0712	10,4645	-2,1655	-0,5029	-0,4937

Fonte: Elaboração própria.

No horizonte de 5 dias úteis, o modelo apresenta desempenho elevado e estatisticamente consistente com uma boa capacidade de antecipação de movimentos de curto prazo. O erro absoluto médio situa-se em aproximadamente 2,5 pontos-base e o RMSE em torno de 3,2 pontos-base, valores que representam uma fração reduzida do nível médio dos spreads. O coeficiente de determinação $R^2 \approx 0{,}93$ indica que mais de noventa por cento da variância fora da amostra é explicada pelas previsões. As correlações de Pearson e Spearman, respectivamente $r \approx 0{,}97$ e $\rho \approx 0{,}92$, mostram que o modelo acompanha de forma muito próxima tanto o nível quanto a ordenação dos períodos de maior e menor spread. Em termos operacionais, isso significa que, em janelas semanais, as projeções geradas são suficientemente precisas, com erros residuais concentrados em movimentos mais abruptos e menos frequentes. Visualmente, a Figura 2 evidencia essa aderência: a trajetória prevista (linha tracejada) praticamente se sobrepõe à curva do spread efetivamente observado no horizonte de 5 dias à frente, com desvios concentrados em movimentos mais abruptos.

O gráfico de horizonte de 5 dias não mostra apenas a projeção para cinco dias à frente, mas sim um *backtest* em horizonte fixo. Em outras palavras, para cada dia t dentro da janela de teste o modelo gera uma previsão do nível do spread em $t + H$ (no caso, $t + 5$ dias úteis). Isso produz uma série contínua de pontos previstos e realizados ao longo de todo o período de teste, de 2025-01 até 2025-09.

Para o horizonte de 22 dias úteis, observa-se um padrão coerente com o aumento da

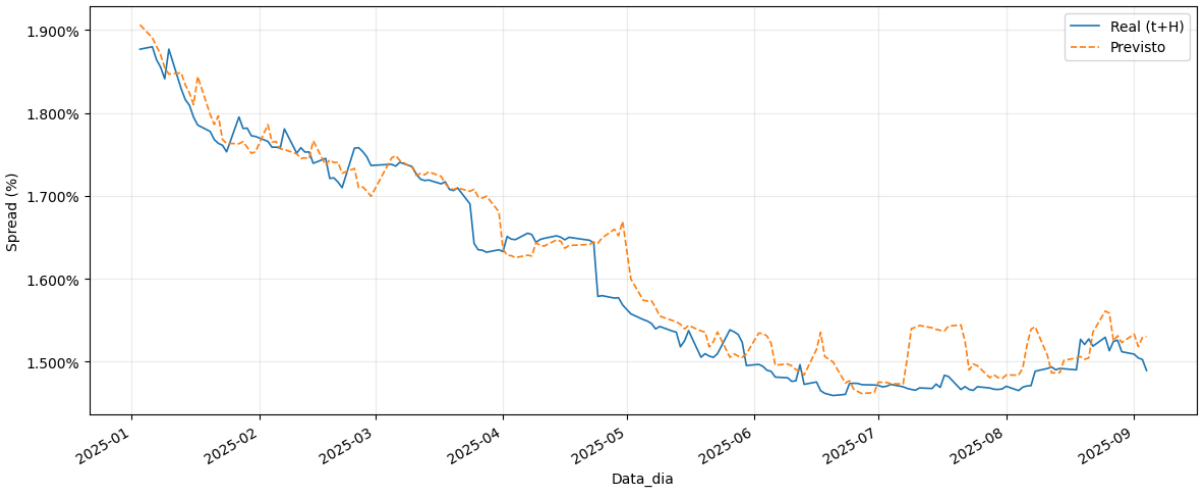

Figura 2 – Desempenho do modelo de previsão com horizonte de 5 dias.

Fonte: Elaborado pelo autor.

incerteza temporal. O MAE sobe para aproximadamente 6,2 bps e o RMSE para cerca de 7,0 bps, enquanto o R^2 se reduz para valores próximos de 0,59. Ainda assim, as correlações de Pearson e Spearman permanecem elevadas em torno de 0,87 e 0,88; sugerindo que o modelo continua capturando de forma razoável a direção dos movimentos, ainda que com erros maiores em nível. A Figura 3 ilustra a curva prevista do spread, mas com maior suavização e algum descolamento em momentos de inflexão mais rápida no mercado.

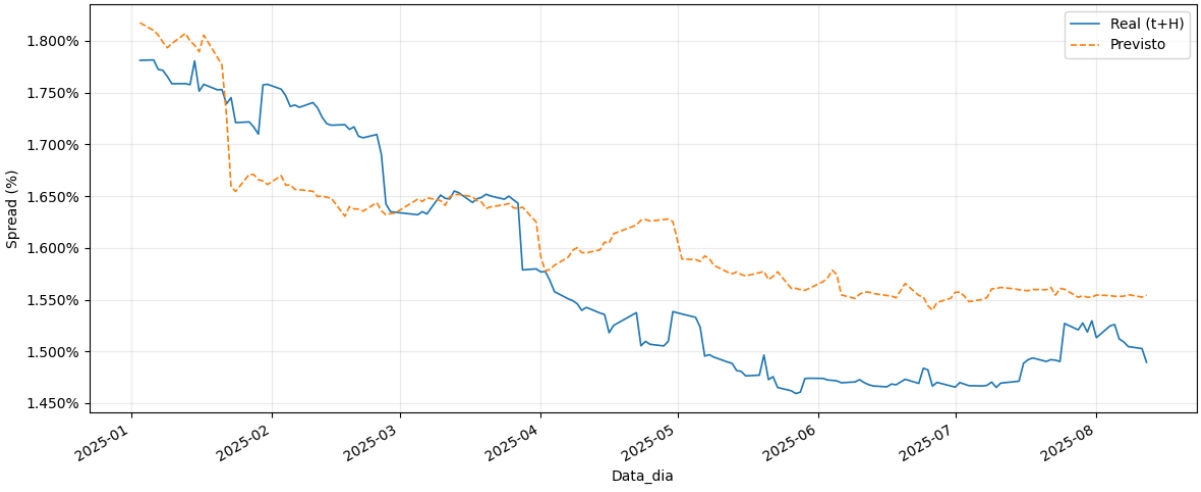

Figura 3 – Desempenho do modelo de previsão com horizonte de 22 dias.

Fonte: Elaborado pelo autor.

No horizonte mais longo, de 66 dias úteis, a capacidade preditiva do modelo se torna substancialmente mais limitada. O MAE se aproxima de 9,1 bps, o RMSE ultrapassa 10 bps e o R^2 torna-se negativo, sinalizando que, nesse prazo, a previsão baseada apenas na informação contida nos spreads passados passa a ser menos informativa do que um modelo de referência

simples. As correlações de Pearson e Spearman negativas sugerem que a perda de aderência na Figura 4, tenta capturar tendências mais suaves, mas não acompanha adequadamente pontos de inflexão mais fortes nem deslocamentos de nível ao longo do período, o que indica que o modelo univariado de spread ponderado produz previsões muito precisas em horizontes curtos, mantém desempenho razoável em torno de um mês e perde qualidade de forma acentuada em horizontes trimestrais.

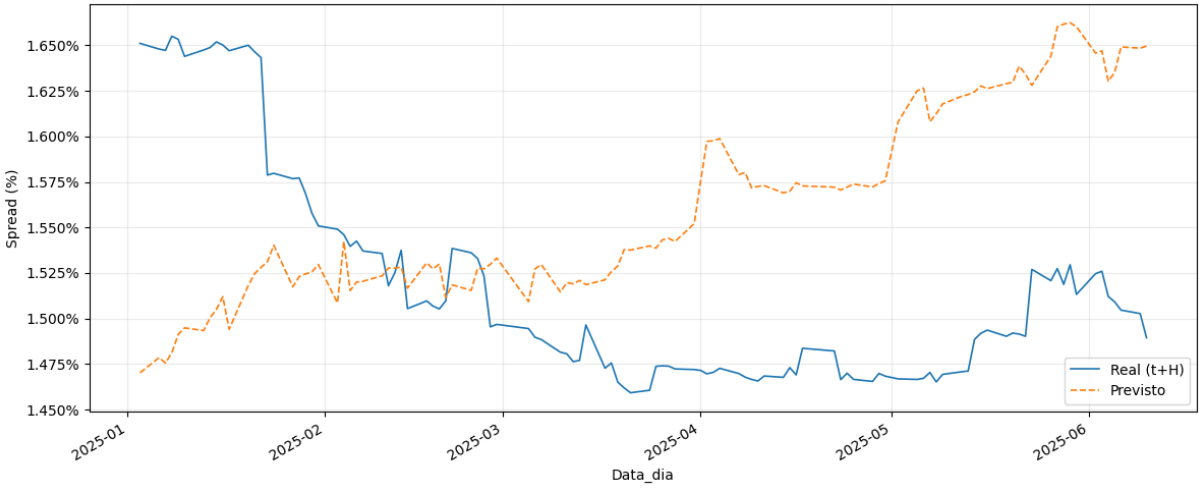

Figura 4 – Desempenho do modelo de previsão com horizonte de 66 dias.

Fonte: Elaborado pelo autor.

4.3 Modelo Multivariado de Previsão e Análise de Desempenho

Nesta seção são apresentados os resultados da abordagem multivariada implementada pelo Código F, na qual a trajetória do spread ponderado é modelada em conjunto com variáveis de captação líquida dos fundos de crédito privado. O objetivo é avaliar em que medida essa dimensão de fluxo melhora a capacidade preditiva em relação ao modelo univariado baseado apenas na própria série histórica do spread, para diferentes horizontes em dias úteis. O desempenho é expresso em pontos base e sintetizado na Tabela 7.

Tabela 7 – Desempenho preditivo do modelo multivariado (Spread + Captação) por horizonte de previsão em dias úteis.

Hor. (dias)	n_{tr}	n_{val}	n_{te}	TopK	Δcap (bps)	MAE (bps)	RMSE (bps)	R^2	r	ρ
5	514	168	169	40	8,4461	3,9366	4,7965	0,8557	0,9820	0,9322
22	497	151	152	30	24,6773	6,1086	6,9320	0,5959	0,9125	0,8572
66	453	107	108	25	79,0972	8,0152	8,9504	-1,3157	0,7786	0,5798

Fonte: Elaboração própria.

Os parâmetros TopK e Δcap auxiliam na leitura dos resultados. O TopK indica quantas variáveis foram selecionadas para cada horizonte como mais correlacionadas com a variação futura do spread. A redução de 40 para 25 variáveis, à medida que o horizonte de previsão se alonga, está associada à diminuição do número efetivo de observações e à menor previsibilidade da série, o que torna recomendável limitar a complexidade do modelo e operar com um conjunto mais controlado de variáveis explicativas. Já Δcap representa o limite máximo imposto para a variação prevista do spread, calibrado a partir da distribuição histórica de $|\Delta y|$. O crescimento de Δcap de aproximadamente 8,4 para 79,1 pontos base entre 5 e 66 dias traduz a expectativa de que movimentos de maior magnitude se concentrem em horizontes mais longos e indica também qual ordem de grandeza o modelo considera plausível ao projetar cenários de estresse.

No horizonte de 5 dias úteis, o modelo multivariado apresenta desempenho equiparável com o modelo univariado. O erro absoluto médio situa-se em torno de 3,9 bps, com RMSE próximo de 4,8 bps e coeficiente de determinação $R^2 \approx 0{,}86$. As correlações de Pearson ($r \approx 0{,}98$) e de Spearman ($\rho \approx 0{,}93$) permanecem muito elevadas, indicando que a trajetória prevista acompanha com fidelidade tanto o nível quanto as oscilações de curto prazo da série. A Figura 5 evidencia esse padrão: a curva prevista (linha tracejada) reproduz a tendência de queda dos spreads ao longo do período de teste, com um viés de nível moderado, sobretudo na parte inicial da amostra.

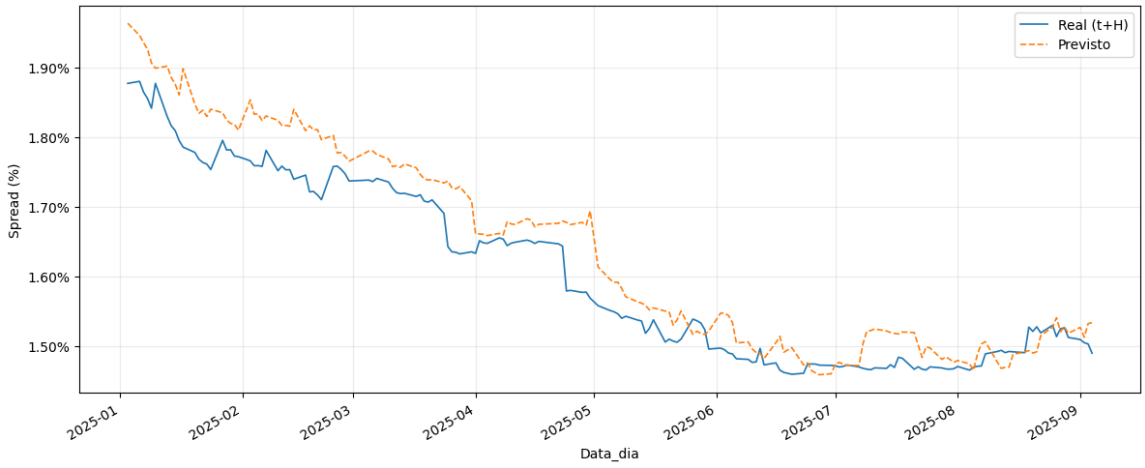

Figura 5 – Desempenho do modelo multivariado de previsão com horizonte de 5 dias.

Fonte: Elaborado pelo autor.

No horizonte de 22 dias úteis, observa-se um padrão coerente com o aumento da incerteza temporal. O MAE cresce para aproximadamente 6,1 bps e o RMSE para cerca de 6,9 bps, enquanto o R^2 se mantém em torno de 0,60. As correlações de Pearson e Spearman, da ordem de 0,91 e 0,86, sugerem que o modelo continua capturando de forma razoável a direção dos movimentos de médio prazo. A Figura 6 mostra, contudo, que a série prevista é mais suave que a observada: a linha tracejada acompanha a tendência descendente dos spreads, mas apresenta

deslocamento de nível e menor amplitude em relação aos degraus mais abruptos, especialmente nas transições entre regimes de maior e menor risco.

Figura 6 – Desempenho do modelo multivariado de previsão com horizonte de 22 dias.

Fonte: Elaborado pelo autor.

No horizonte mais longo, de 66 dias úteis, a capacidade preditiva torna-se substancialmente mais limitada em termos de erro de nível: o MAE atinge cerca de 8,0 bps, o RMSE aproxima-se de 9,0 bps e o coeficiente de determinação torna-se negativo ($R^2 \approx -1{,}32$), indicando que, nesse prazo, o modelo não é capaz de reduzir o erro quadrático em relação a um preditor de referência. Ainda assim, as correlações de Pearson e Spearman permanecem positivas e relativamente elevadas ($r \approx 0{,}78$ e $\rho \approx 0{,}58$). A Figura 7 ilustra a trajetória prevista e exibe um perfil suavizado em relação à série observada, capturando a direção geral de queda, mas não reproduzindo plenamente as mudanças de regime nem o movimento de recomposição parcial dos spreads ao final da amostra.

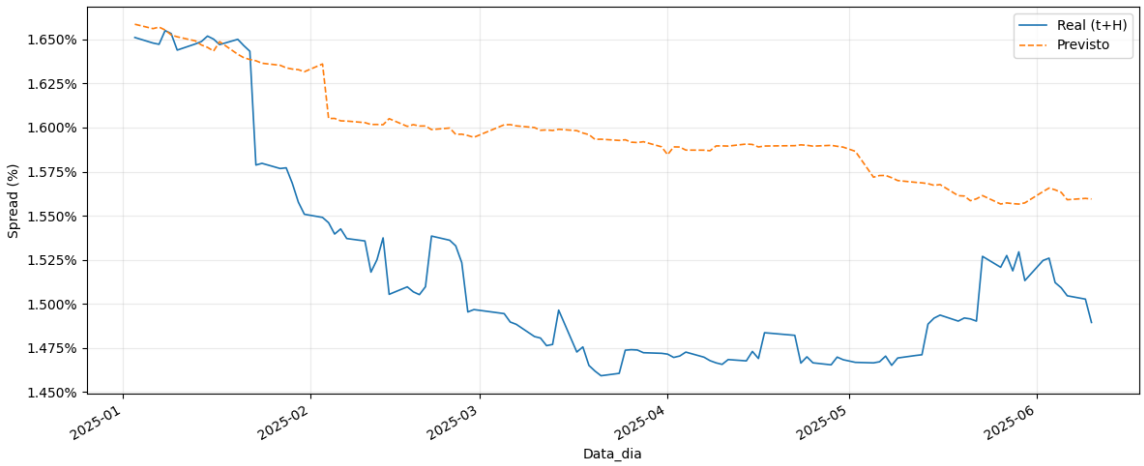

Figura 7 – Desempenho do modelo multivariado de previsão com horizonte de 66 dias.

Fonte: Elaborado pelo autor.

5 Discussão dos Resultados

A comparação entre o modelo univariado, baseado apenas na própria série de spread ponderado, e o modelo multivariado, que incorpora variáveis de captação líquida e indicadores macroeconômicos derivados, deve ser compreendida sobre o contexto amostral onde ambos utilizaram o período de abril de 2022 a setembro de 2025, já em um ambiente pós-pandemia de COVID-19, em que o mercado de crédito privado brasileiro passa por um processo de retomada e amadurecimento, com incremento gradual de liquidez e diversificação de emissores.

Em teste, o modelo apenas com defasagens do spread apresenta erro absoluto médio em torno de 0,86 bps, enquanto o modelo com captação obtém aproximadamente 0,84 bps de MAE, ambos com RMSE próximo de 1,33 bps. Trata-se, portanto, de uma melhoria incremental de cerca de 2% no erro médio, compatível com a ideia de que o comportamento de curto prazo do spread é dominado por sua própria inércia e microestrutura, enquanto as variáveis de captação macroeconômica e de fluxo agregado contribuem apenas para refinar o ajuste residual. As Tabelas 6 e 7 reforçam esse quadro: em todos os horizontes, o modelo parte de um patamar de erro já muito baixo, o que naturalmente reduz a margem para ganhos substanciais.

No horizonte de 5 dias úteis, o modelo univariado exibe desempenho superior: o MAE de aproximadamente 2,46 bps e o $R^2 \approx 0{,}93$ indicam que, em uma janela de uma semana, as defasagens do spread e as *microfeatures* de volatilidade e tendência já são suficientes para capturar com precisão os movimentos observados, como mostram as figuras de desempenho de curto prazo (Figuras 2 e 5). Em contraste, o modelo com captação apresenta MAE próximo de 3,94 bps e $R^2 \approx 0{,}86$, evidenciando perda de qualidade em relação à versão apenas com spread. Uma interpretação plausível é que, em horizontes tão curtos, a inclusão de variáveis mensais de fluxo e de regime macroeconômico introduz mais ruído do que sinal. Assim, para decisões táticas de curtíssimo prazo, o modelo univariado se mostra mais eficiente.

No horizonte intermediário de 22 dias úteis (aproximadamente um mês de pregões), a comparação se torna mais equilibrada. As métricas de ambos os modelos ficam bastante próximas: o MAE do modelo com captação situa-se em torno de 6,11 bps, ligeiramente melhor que os 6,17 bps do modelo apenas com spread, enquanto o R^2 permanece na faixa de 0,59 para os dois casos. As correlações de Pearson e Spearman, da ordem de 0,87 a 0,91, sugerem que ambos os modelos capturam bem a direção dos movimentos de médio prazo, embora o modelo multivariado tenda a produzir trajetórias mais suaves e com menor overshooting em pontos de inflexão, como se observa ao comparar as Figuras 3 e 6. Esse resultado é compatível com a evidência, discutida na seção de correlação entre spread e captação, de que os fluxos líquidos possuem algum poder explicativo em janelas mensais, mas ainda convivem com uma forte componente autorregressiva da própria série de spread. Do ponto de performance, o modelo com

captação em $H = 22$ dias não apresentou ganhos expressivos frente à complexidade ao modelo mais simples.

A diferença mais marcante surge no horizonte de 66 dias úteis, em que ambos os modelos passam a enfrentar o desafio de antecipar movimentos trimestrais em um ambiente sujeito a mudanças de regime, como a reprecificação de risco após choques corporativos e a reavaliação de expectativas frente à trajetória da política monetária e da taxa de câmbio. Nesse horizonte mais longo, as métricas absolutas se deterioram, com R^2 negativo tanto para o modelo sem captação quanto para o modelo com captação, sinalizando que nenhuma das abordagens é capaz de reduzir o erro quadrático. Ainda assim, o modelo multivariado apresenta desempenho relativamente melhor com o MAE de cerca de 9,07 bps (modelo apenas spread) que cai para 8,02 bps, e o RMSE de 10,46 bps para 8,95 bps, ao mesmo tempo em que as correlações voltam a ser positivas e relativamente elevadas ($r \approx 0{,}78$, $\rho \approx 0{,}58$), em contraste com as correlações negativas observadas no modelo univariado. Esses números sugerem que, apesar de não produzir previsões pontuais precisas em três meses, o modelo com captação consegue ao menos ordenar corretamente períodos de abertura e fechamento de spread, capturando ciclos mais persistentes de risco que se refletem na captação líquida agregada.

Em termos de interpretação econômica, essa evidência é coerente com a dinâmica observada no mercado brasileiro no período analisado. A valorização do dólar no final de 2024 e início de 2025, combinada com incertezas fiscais, impactou gradualmente os prêmios de risco e a disposição dos investidores em alocar recursos em crédito privado. Além disso, eventos como os casos de Lojas Americanas e Light, atuaram como gatilhos de reavaliação de risco, mas seus efeitos mais duradouros se manifestam justamente via fluxos de captação e resgate nos fundos. O fato de o modelo com captação apresentar desempenho relativamente melhor em horizontes mais longos, de cerca de 11% com relação ao MAE e de 14% para o RMSE, sugere que as variáveis macro e de fluxo capturam essas forças de fundo, ainda que de maneira imperfeita. Em contrapartida, a fraca performance global em $H = 66$ dias indica que o mercado ainda está sujeito a quebras estruturais e a episódios de estresse difíceis de antecipar com base apenas nos dados históricos disponíveis, o que limita o poder preditivo de qualquer modelo puramente estatístico.

Do ponto de vista de gestão de risco, os resultados apontam para um uso complementar dos dois modelos. Em horizontes de curtíssimo prazo, o modelo univariado se mostra mais adequado como ferramenta operacional, seja para monitorar desvios entre spread observado e previsto, seja para auxiliar em decisões táticas de hedge e marcação a mercado, dado seu erro menor e maior aderência gráfica nas Figuras 2 e 3. Já o modelo com captação parece mais apropriado para análises de cenário e planejamento em janelas mensais e trimestrais, quando o objetivo não é tanto acertar o ponto exato do spread, mas sim identificar tendências e fases do ciclo de crédito associadas a mudanças de fluxo e de condições macroeconômicas, como ilustram as Figuras 6 e 7.

6 Considerações Finais

Por meio da análise de correlação com defasagens e o modelo de Mínimos Quadrados Ponderados (WLS) existe uma relação robusta de entre os spreads e a captação líquida dos fundos. Verificou-se uma correlação negativa moderada, coerente com a ideia de que a abertura dos spreads coincide com saídas líquidas de recursos e maior aversão ao risco. Após a dessazonalização das séries, essa relação tornou-se ainda mais nítida, e a correlação máxima passou a ocorrer em torno de dez meses de defasagem, com coeficiente de Pearson superior a 0,82. O modelo WLS estimado incorpora essa defasagem ótima e um termo autorregressivo da captação, alcançando um R^2 superior a 0,90 e coeficiente positivo e estatisticamente significativo para o spread defasado. Em termos econômicos, os resultados sugerem que períodos de spread elevado, embora inicialmente associados a resgates, acabam por atrair fluxos líquidos relevantes de recursos alguns meses à frente, refletindo um comportamento de *yield chasing*: à medida que o prêmio de risco se torna mais atrativo, investidores institucionais passam a realocar capital para fundos de crédito privado.

No eixo preditivo, os modelos de *machine learning* aplicados à série de spreads ponderados evidenciaram tanto o potencial quanto os limites da previsibilidade no mercado de crédito privado. O primeiro Modelo, univariado, baseado apenas em defasagens e transformações do próprio spread, mostrou-se extremamente eficiente com MAE em torno de 0,86 bps e R^2 próximo de 0,99 no conjunto de teste. Na Etapa 3, que passa a prever variações futuras em horizontes fixos, o modelo manteve desempenho muito bom em 5 dias úteis e razoável em 22 dias, mas apresentou deterioração acentuada em 66 dias, com R^2 negativo e correlações inversas, indicando dificuldade em antecipar movimentos trimestrais em um ambiente sujeito a mudanças de regime.

O Modelo multivariado incorpora variáveis derivadas da captação líquida como informação macroeconômica de baixa frequência, trouxe nuances importantes para a interpretação da previsibilidade. Houve um ganho pequeno, com redução de aproximadamente 2% no MAE em relação ao modelo apenas com defasagens do spread, reforçando a ideia de que, no curtíssimo prazo, a própria série de preços contém quase toda a informação relevante. Em horizontes de 5 dias, a inclusão de variáveis de fluxo chegou a prejudicar o desempenho, sugerindo que, no curto prazo, os agregados mensais de captação introduzem mais ruído do que sinal. Entretanto, em 22 dias os resultados tornaram-se praticamente equivalentes entre os modelos, e em 66 dias o modelo com captação passou a apresentar ganhos relativos claros: redução de cerca de 11% no MAE e de 14% no RMSE em comparação ao modelo univariado, além de correlações positivas e significativas com os valores futuros, ainda que com R^2 globalmente negativo. Esses achados indicam que, embora o nível exato do spread a três meses seja difícil de prever, as variáveis de fluxo ajudam a ordenar melhor os períodos de abertura e fechamento, funcionando como indicador de tendência e de fase do ciclo de crédito.

Diante das limitações que o estudo apresenta, o horizonte temporal de pouco mais de três anos, ainda que posterior ao choque da COVID-19 e incluindo episódios relevantes como as crises de Lojas Americanas e Light e a recente volatilidade cambial, ainda é relativamente curto para capturar ciclos completos de crédito, especialmente em um mercado em expansão e em transformação regulatória. Em decorrência do objetivo desse trabalho, a análise concentra-se em um único índice de referência (Idex–CDI JGP) e em uma medida agregada de captação líquida, o que implica abstrair heterogeneidades setoriais, diferenças de mandatos entre fundos e eventuais mudanças na composição do próprio índice. Além disso, os modelos de machine learning e outras estruturas econométricas privilegiam estruturas lineares ou quase lineares e não tratam explicitamente mudanças de regime, o que pode limitar a capacidade de capturar comportamentos não lineares em períodos de estresse mais agudo.

As limitações apontam, por outro lado, para caminhos promissores de pesquisa futura. A direção consiste em estender a base histórica para além de 2022, combinando períodos pré e pós-pandemia com modelos de *switching* ou estruturas de regime, de forma a capturar explicitamente quebras estruturais na dinâmica de spreads e fluxos. Por fim, a comparação entre os modelos sugere que a inclusão de variáveis macro mais ricas, como medidas de volatilidade implícita, prêmios de risco de mercado (spread entre NTN-B e DI) e indicadores de atividade setorial, podem ajudar a capturar melhor os impactos de choques como a alta do dólar e mudanças na percepção de risco-país. Além disso, explorar modelos capazes de lidar explicitamente com mudanças de regime como por exemplo, *boosting*, redes neurais recorrentes ou LSTM são vias de possíveis análise. Em um mercado ainda em expansão e sujeito a episódios de estresse, ampliar o horizonte de dados e refinar a modelagem ao longo do tempo é condição necessária para que o poder preditivo desses modelos se aproxime do seu potencial pleno.

O trabalho evidencia que a relação entre spread de crédito e captação líquida é forte, estatistica e economicamente relevante no mercado brasileiro, e que a combinação entre técnicas econométricas tradicionais e ferramentas modernas de machine learning oferece um caminho para aprofundar o entendimento dessa dinâmica. Os modelos desenvolvidos não esgotam as possibilidades de previsão, mas fornecem um ponto de contribuição para um mercado de crédito privado mais transparente, eficiente e resiliente.

Referências

ALMEIDA, C. Precificação de debêntures e risco de crédito: Uma abordagem aplicada ao mercado brasileiro. *Revista Brasileira de Finanças*, v. 19, n. 3, p. 102–128, 2021.

ANBIMA. *Boletim de Fundos de Investimento – Anbima - anbima.com.br*. <https://www.anbima.com.br/pt_br/informar/relatorios/fundos-de-investimento/boletim-de-fundos-de-investimentos/boletim-de-fundos-de-investimentos.htm>. Acesso em: 5 out. 2025.

ANBIMA. *Debêntures incentivadas registram recorde de R$ 120,3 bilhões em emissões em 2024 – ANBIMA — anbima.com.br*. 2024. <https://www.anbima.com.br/pt_br/imprensa/debentures-incentivadas-registram-recorde-de-r-120-3-bilhoes-em-emissoes-em-2024.htm>. Acesso em: 5 out. 2025.

ANBIMA, A. B. das Entidades dos Mercados Financeiro e de C. *Captação líquida de fundos de investimento*. São Paulo: [s.n.], 2023. <https://www.anbima.com.br/pt_br/noticias/fundos-resgates-no-fim-de-janeiro-anulam-captacao-liquida-no-mes.htm>. Acesso em: 26 out. 2025.

ANDERSSON, O. *Forecasting Corporate Credit Spreads : A Deep Learning Approach*. Dissertação (Mestrado) — KTH, School of Architecture and the Built Environment (ABE), Real Estate and Construction Management, 2020.

BOX, G. E. P. et al. *Time Series Analysis: Forecasting and Control*. [S.l.]: John Wiley & Sons, 2016.

BROCKWELL, P. J.; DAVIS, R. A. *Time series: theory and methods*. [S.l.]: Springer science & business media, 2009.

BROCKWELL, P. J.; DAVIS, R. A. *Introduction to Time Series and Forecasting*. 3rd. ed. [S.l.]: Springer, 2016. (Springer Texts in Statistics). ISBN 9783319298542.

BROWNLEE, J. *Deep Learning for Time Series Forecasting: Predict the Future with MLPs, CNNs and LSTMs in Python*. [S.l.]: Machine Learning Mastery, 2018.

COSTA, R. Expansão do mercado de crédito privado e seus determinantes recentes. *Revista de Economia e Finanças*, v. 28, n. 3, p. 123–140, 2022.

CVM, C. de V. M. *Resolução CVM nº 175, de 23 de dezembro de 2022*. Brasília: [s.n.], 2022. Dispõe sobre a constituição, o funcionamento e a divulgação de informações dos fundos de investimento. Anexo Normativo I – Fundos de Investimento Financeiros (FIF), Art. 70, Subseção VII. Disponível em: <https://conteudo.cvm.gov.br/export/sites/cvm/legislacao/resolucoes/anexos/100/resol175consolid.pdf>.

CVM, C. de V. M. *Base de dados – Informações diárias dos fundos de investimento (INF_DIARIO)*. 2025. <https://dados.cvm.gov.br/dados/FI/DOC/INF_DIARIO/DADOS/>. Série histórica de informações diárias de fundos de investimento. Acesso em: 11 set. 2025.

DERRICK, T. R.; THOMAS, J. M. Time series analysis: The cross-correlation function. In: . [s.n.], 2004. Disponível em: <https://api.semanticscholar.org/CorpusID:265951764>.

DIXON, M.; HALPERIN, I.; BILOKON, P. *Machine Learning in Finance: From Theory to Practice*. [S.l.]: Springer International Publishing, 2020. ISBN 9783030410681.

FABIAN, P. Scikit-learn: Machine learning in python. *Journal of machine learning research 12*, p. 2825, 2011.

FAHRMEIR, L. et al. *Regression: Models, Methods and Applications*. [S.l.]: Springer Berlin Heidelberg, 2013. ISBN 9783642343339.

FÁVERO, L. P.; BELFIORE, P. *Manual de análise de dados: estatística e modelagem multivariada com Excel®, SPSS® e Stata®*. [S.l.]: Elsevier Brasil, 2017.

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

GREENE, W. H. *Econometric Analysis*. 8th. ed. [S.l.]: Prentice Hall, 2018. ISBN 9780134461366.

GÜN, M.; KARTAL, B. *Machine Learning in Finance: Trends, Developments and Business Practices in the Financial Sector*. Springer Nature Switzerland, 2025. (Contributions to Finance and Accounting). ISBN 9783031832666. Disponível em: <https://books.google.com.br/books?id=soZSEQAAQBAJ>.

GéRON, A. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. 3rd. ed. [S.l.]: O'Reilly Media, 2022. ISBN 978-1098125974.

HAFFKI, R.; HENNIG, K. Determinants of credit spreads and cash flow-related lending in commercial real estate. *Research in International Business and Finance*, v. 74, p. 102669, 2025. ISSN 0275-5319. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0275531924004628>.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd. ed. [S.l.]: Springer, 2009. ISBN 978-0387848570.

HASTIE, T.; TIBSHIRANI, R.; WAINWRIGHT, M. *Statistical Learning with Sparsity: The Lasso and Generalizations*. [S.l.]: Chapman and Hall/CRC, 2015. ISBN 978-1498712163.

JAPKOWICZ, N.; BOUKOUVALAS, Z. *Machine learning evaluation: towards reliable and responsible AI*. [S.l.]: Cambridge University Press, 2024.

JGP. *Metodologia do Índice Idex-CDI JGP*. 2023. Disponível em: <https://www.jgp.com.br>. Acesso em: 5 out. 2025.

JGP. *Base histórica do Índice Idex-CDI (arquivo .xlsx)*. 2025. <https://jgp-credito-public-s3.s3.us-east-1.amazonaws.com/idex/idex_cdi_geral_datafile.xlsx>. Arquivo de dados em formato Excel. Acesso em: 11 set. 2025.

JGP Asset Management. *Metodologia Idex-CDI JGP: Índice de Crédito Privado Doméstico*. Rio de Janeiro, 2023. Acesso em: 26 out. 2025. Disponível em: <https://idex.jgp.com.br/>.

KAZYMYR, V. et al. *Mathematical Modeling and Simulation of Systems: Selected Papers of 19th International Conference, MODS, November 11–13, 2024, Chernihiv, Ukraine*. [S.l.]: Springer Nature Switzerland, 2025. (Lecture Notes in Networks and Systems). ISBN 9783031907357.

KUHN, M.; JOHNSON, K. *Applied Predictive Modeling*. [S.l.]: Springer, 2013. ISBN 9781461468493.

MARQUES, A. L.; SANTOS, B. H. *Mercado de Capitais e Financiamento Corporativo no Brasil*. São Paulo: Atlas, 2020.

MONTGOMERY, D.; PECK, E.; VINING, G. *Introduction to Linear Regression Analysis*. [S.l.]: Wiley, 2015. (Wiley Series in Probability and Statistics). ISBN 9781119180173.

MURPHY, K. P. *Machine Learning: A Probabilistic Perspective*. [S.l.]: The MIT Press, 2012. ISBN 0262018020.

NETO, A. A. Mercado financeiro. 2000.

NEUSSER, K. et al. *Time series econometrics*. [S.l.]: Springer, 2016. v. 1.

NORIEGA, J.; RIVERA, L.; HERRERA, J. Machine learning for credit risk prediction: A systematic literature review. *Data*, MDPI, v. 8, n. 11, p. 169, 2023.

PANDA, S. et al. *Computing, Communication and Learning: Second International Conference, CoCoLe 2023, Warangal, India, August 29–31, 2023, Proceedings*. [S.l.]: Springer Nature Switzerland, 2024. (Communications in Computer and Information Science). ISBN 9783031569982.

PRADO, M. *Fundo de crédito privado com resgate alongado é opção para driblar retorno apertado dessa aplicação, diz XP*. 2025. Disponível online. Acesso em: 15 set. 2025.

SHAO, Y. et al. *A Novel Methodology in Credit Spread Prediction Based on Ensemble Learning and Feature Selection*. 2024.

SHUMWAY, R. H.; STOFFER, D. S. *Time Series Analysis and Its Applications: With R Examples*. 4th. ed. [S.l.]: Springer, 2017. (Springer Texts in Statistics). ISBN 9783319524528.

SILVA, J. P.; OLIVEIRA, M. O papel do crédito privado no sistema financeiro brasileiro. *Revista Brasileira de Finanças*, v. 19, n. 2, p. 45–68, 2021.

SINATORA, J. R. P. Mercado de capitais. *Londrina: Editora e Distribuidora Educacional SA*, 2016.

TRIOLA, M. F. Introdução à estatística. In: *Introdução à estatística*. [S.l.: s.n.], 2008. p. xxvi–310.

WEI, W. *Multivariate Time Series Analysis and Applications*. [S.l.]: Wiley, 2019. (Wiley Series in Probability and Statistics). ISBN 9781119502852.

WOOLDRIDGE, J. M. *Introductory Econometrics: A Modern Approach*. 8th. ed. [S.l.]: Cengage Learning, 2025. ISBN 9780357900161.

WRIGHT, J.; MA, Y. *High-Dimensional Data Analysis with Low-Dimensional Models: Principles, Computation, and Applications*. [S.l.]: Cambridge University Press, 2022.

WU, Y. et al. *Predicting Credit Spreads and Ratings with Machine Learning: The Role of Non-Financial Data*. 2025.

APÊNDICE A – Código Python para a Base de Dados

```
1
2  import pandas as pd
3  import numpy as np
4  import re
5  from pathlib import Path
6
7  # ===== Caminhos (Windows) =====
8  path_base = Path(r"\estudo_jgp.xlsx")
9  path_idex = Path(r"\idex_cdi_geral_datafile.xlsx")
10
11 assert path_base.exists(), f"Arquivo nao encontrado: {path_base
       }"
12 assert path_idex.exists(),  f"Arquivo nao encontrado: {
       path_idex}"
13
14 # ===== Helpers =====
15 def parse_brl(x):
16     """Converte texto BRL em float (R$, pontos de milhar,
          virgula decimal, parenteses e hifen unicode)."""
17     if pd.isna(x):
18         return np.nan
19     s = str(x).strip().replace("-", "-")
20     neg = False
21     if s.startswith("(") and s.endswith(")"):
22         neg = True
23         s = s[1:-1]
24     s = re.sub(r"[^\d,.\-]", "", s)
25     if "," in s and s.count(",") == 1:
26         s = s.replace(".", "").replace(",", ".")
27     try:
28         v = float(s)
29     except:
30         v = np.nan
31     return -v if neg else v
32
```

```
33 def pick_col(df, guesses, fallback_idx):
34     cols = df.columns.tolist()
35     for c in cols:
36         if any(g.lower() in str(c).lower() for g in guesses):
37             return c
38     return cols[fallback_idx]
39 
40 def to_fraction(series):
41     s = pd.to_numeric(series, errors="coerce")
42     return s/100.0 if s.dropna().abs().median() > 1 else s
43 
44 # ===== 1) Fundos: somar Aplicacoes (D) e Resgates (E) por mes
       =====
45 df_f = pd.read_excel(path_base, sheet_name="base", header=0)
46 
47 col_data = pick_col(df_f, ["data"], 0)              # A
48 col_cnpj = pick_col(df_f, ["cnpj", "fundo"], 1)     # B
49 col_D    = pick_col(df_f, ["aplic", "entrada", "subscri", "col
       d", " d "], 3)  # D
50 col_E    = pick_col(df_f, ["resg", "saida", "saida", "col e", "
       e "], 4)        # E
51 
52 df_f = df_f[[col_data, col_cnpj, col_D, col_E]].copy()
53 df_f.columns = ["Data", "CNPJ_Fundo", "Aplicacoes_raw", "
       Resgates_raw"]
54 
55 df_f["Data"] = pd.to_datetime(df_f["Data"], errors="coerce")
56 df_f["Aplicacoes"] = df_f["Aplicacoes_raw"].apply(parse_brl)
57 df_f["Resgates"]   = df_f["Resgates_raw"].apply(parse_brl)
58 
59 # Normaliza sinais (esperado >= 0)
60 if df_f["Aplicacoes"].dropna().median() < 0:
61     df_f["Aplicacoes"] = -df_f["Aplicacoes"]
62 if df_f["Resgates"].dropna().median() < 0:
63     df_f["Resgates"] = -df_f["Resgates"]
64 
65 df_f = df_f.dropna(subset=["Data"])
66 df_f["Mes"] = df_f["Data"].dt.to_period("M").dt.to_timestamp()
       # primeiro dia do mes
67 
68 agg_fundos = (df_f.groupby("Mes")
69                 .agg(Captacao=("Aplicacoes", "sum"),
```

```
                        Resgate=("Resgates", "sum"))
                .reset_index()
                .sort_values("Mes"))

agg_fundos["Captacao Liquida (bi)"] = (agg_fundos["Captacao"] -
    agg_fundos["Resgate"]) / 1e9

# ===== 2) IDEX: media ponderada mensal do spread =====
df_idex = pd.read_excel(
    path_idex, sheet_name="Detalhado",
    usecols=["Data", "Spread de compra (%)", "Peso no indice
       (%)", "Emissor"]
)
df_idex["Data"] = pd.to_datetime(df_idex["Data"], errors="
   coerce")
df_idex = df_idex.dropna(subset=["Data"])

distressed = ["Lame", "Light", "Aeris", "Viveo", "AMERICANAS SA
   "]
df_idex = df_idex[~df_idex["Emissor"].str.contains("|".join(
   distressed), case=False, na=False)]

df_idex["Spread_frac"] = to_fraction(df_idex["Spread de compra
   (%)"])
df_idex["Peso_frac"]   = to_fraction(df_idex["Peso no indice
   (%)"])
df_idex["Mes"] = df_idex["Data"].dt.to_period("M").dt.
   to_timestamp()

def wavg(g):
    w, x = g["Peso_frac"], g["Spread_frac"]
    return np.average(x, weights=w) if w.notna().any() and w.
       sum() != 0 else np.nan

spread_mensal = (df_idex.groupby("Mes").apply(wavg)
                 .rename("Spread_ponderado_frac")
                 .reset_index()
                 .sort_values("Mes"))
spread_mensal["Spread ponderado"] = spread_mensal["
   Spread_ponderado_frac"] * 100  # em %

# ===== 3) Merge e selecao de colunas =====
```

```
tabela = (agg_fundos
          .merge(spread_mensal[["Mes", "Spread ponderado"]], on="Mes", how="inner")
          .rename(columns={"Mes": "Data"})
          .sort_values("Data"))

tabela = tabela[["Data", "Captacao", "Resgate", "Captacao Liquida (bi)", "Spread ponderado"]]

# Arredondamentos para visualizacao
tabela["Captacao"] = tabela["Captacao"].round(2)
tabela["Resgate"]  = tabela["Resgate"].round(2)
tabela["Captacao Liquida (bi)"] = tabela["Captacao Liquida (bi)"].round(3)
tabela["Spread ponderado"] = tabela["Spread ponderado"].round(2)

# ===== 4) Salvar e exibir =====
out_path = path_base.parent / "tabela_mensal_fluxo_spread.xlsx"
tabela.to_excel(out_path, index=False)
print(f"Arquivo salvo em: {out_path}")

# Mostra as primeiras linhas
tabela.head(12)
```

APÊNDICE B – Código Python do Gráfico Captação Líquida vs Spread Ponderado

```
1
2  # === Grafico: Captacao Liquida vs Spread Ponderado ===
3  import matplotlib.pyplot as plt
4  import matplotlib.ticker as mtick
5  import matplotlib.dates as mdates
6  import numpy as np
7
8  def plot_fluxo_vs_spread_p(tabela, salvar_png=None, estilo='
       seaborn-v0_8-whitegrid'):
9      """
10     Plota barras da 'Captacao Liquida (bi)' e linha do 'Spread
           ponderado (%)'
11
12     Parametros:
13     -----------
14     tabela : DataFrame
15         DataFrame com colunas: Data, Captacao Liquida (bi),
               Spread ponderado
16     salvar_png : str, optional
17         Caminho para salvar a figura
18     estilo : str
19         Estilo matplotlib a ser aplicado
20     """
21
22     # Verificar colunas necessarias
23     req = {"Data", "Captacao Liquida (bi)", "Spread ponderado"}
24     if not req.issubset(tabela.columns):
25         raise ValueError(f"Faltam colunas: {sorted(req - set(
               tabela.columns))}")
26
27     # Aplicar estilo clean
28     plt.style.use(estilo)
29
30     # Preparar dados
```

```
    plot_df = tabela.sort_values("Data").dropna(subset=["Data"]).copy()

    # Criar figura com proporcao
    fig, ax1 = plt.subplots(figsize=(13, 6.5))

    # --- PALETA DE CORES ---
    cores = {
        'captacao_positiva': '#1E6FBA',     # Azul
        'captacao_negativa': '#D9534F',     # Vermelho
        'spread': '#F39C12',                # Laranja
        'grid': '#E5E5E5',
        'fundo': '#FFFFFF',
        'texto': '#000000',                 # PRETO para texto e eixos
        'eixos': '#000000'                  # PRETO para eixos
    }

    # Fundo branco puro
    fig.patch.set_facecolor(cores['fundo'])
    ax1.set_facecolor(cores['fundo'])

    # --- BARRAS: Captacao Liquida (bi) ---
    # Separar positivos e negativos para cores distintas
    positivos = plot_df[plot_df["Captacao Liquida (bi)"] >= 0]
    negativos = plot_df[plot_df["Captacao Liquida (bi)"] < 0]

    if not positivos.empty:
        ax1.bar(positivos["Data"], positivos["Captacao Liquida (bi)"],
                color=cores['captacao_positiva'], alpha=0.9, width=6,
                edgecolor=cores['captacao_positiva'], linewidth=0.8,
                label='Captacao Liquida (bi)')

    if not negativos.empty:
        ax1.bar(negativos["Data"], negativos["Captacao Liquida (bi)"],
                color=cores['captacao_negativa'], alpha=0.9, width=6,
```

```
65                 edgecolor=cores['captacao_negativa'], linewidth
                       =0.8)
66
67     ax1.set_ylabel("Captacao Liquida (bi)", fontsize=11,
           fontweight='bold',
68                 color=cores['texto'])  # Texto em preto
69
70     # Formatacao BR para eixo Y1
71     def reais_bi(x, pos):
72         if x == 0:
73             return "R$ 0"
74         return f"R$ {x:,.1f} bi".replace(",", "X").replace(".",
               ",").replace("X", ".")
75
76     ax1.yaxis.set_major_formatter(mtick.FuncFormatter(reais_bi)
           )
77
78     # --- LINHA: Spread ponderado (%) ---
79     ax2 = ax1.twinx()
80     linha_spread, = ax2.plot(plot_df["Data"], plot_df["Spread
           ponderado"],
81                              color=cores['spread'], linewidth=3,
                                   marker='o',
82                              markersize=4.5, markeredgecolor='
                                   white', markeredgewidth=1.5,
83                              label='Spread ponderado (%)',
                                   zorder=5)
84
85     ax2.set_ylabel("Spread ponderado (%)", fontsize=11,
           fontweight='bold',
86                 color=cores['texto'])  # Texto em preto
87     ax2.yaxis.set_major_formatter(mtick.PercentFormatter(
           decimals=1))
88
89     # --- MELHORIAS NO EIXO X ---
90     dias_span = (plot_df["Data"].max() - plot_df["Data"].min())
           .days
91
92     if dias_span <= 90:
93         locator = mdates.WeekdayLocator(interval=2)
94         formato = mdates.DateFormatter('%d/%b')
95     elif dias_span <= 365:
```

```
96          locator = mdates.MonthLocator()
97          formato = mdates.DateFormatter('%b/%y')
98      else:
99          locator = mdates.MonthLocator(interval=3)
100         formato = mdates.DateFormatter('%b/%y')
101
102     ax1.xaxis.set_major_locator(locator)
103     ax1.xaxis.set_major_formatter(formato)
104
105     # Rotacao suave das labels - todas em preto
106     plt.setp(ax1.get_xticklabels(), rotation=45, ha='right',
            rotation_mode='anchor',
107                color=cores['texto'])
108
109     # --- GRADE ---
110     ax1.grid(True, axis='y', linestyle='--', alpha=0.25, color=
            cores['grid'])
111     ax1.grid(False, axis='x')  # Remove grid horizontal para
            mais clean
112
113     # --- EIXOS E TICKS EM PRETO ---
114     # Configurar todas as cores dos eixos para preto
115     ax1.tick_params(axis='both', colors=cores['texto'])
116     ax2.tick_params(axis='y', colors=cores['texto'])
117
118     # Configurar as bordas dos eixos (spines) para preto
119     for spine in ax1.spines.values():
120         spine.set_color(cores['eixos'])
121         spine.set_linewidth(1)
122
123     for spine in ax2.spines.values():
124         spine.set_color(cores['eixos'])
125         spine.set_linewidth(1)
126
127     # --- LIMITES E ESCALA ---
128     # Ajuste automatico dos limites com margem inteligente
129     margem_captacao = plot_df["Captacao Liquida (bi)"].abs().
            max() * 0.15
130     ax1.set_ylim(plot_df["Captacao Liquida (bi)"].min() -
            margem_captacao,
131                  plot_df["Captacao Liquida (bi)"].max() +
                        margem_captacao)
```

```
132
133     # Spread comeca em 0 para melhor referencia
134     margem_spread = plot_df["Spread ponderado"].max() * 0.08
135     ax2.set_ylim(0, plot_df["Spread ponderado"].max() +
            margem_spread)
136
137     # --- TITULO ---
138     periodo = f"{plot_df['Data'].min():%b/%y}-{plot_df['Data'].
            max():%b/%y}"
139     ax1.set_title(f"Captacao Liquida vs. Spread Ponderado - {
            periodo}",
140                   pad=15, fontsize=12, fontweight='bold', color
                          =cores['texto'])
141
142     # --- LEGENDA ---
143     from matplotlib.patches import Patch
144
145     # Elementos da legenda
146     elementos_legenda = [
147         Patch(color=cores['captacao_positiva'], alpha=0.9,
                label='Captacao Liquida'),
148         plt.Line2D([0], [0], color=cores['spread'], linewidth
                =3,
149                    label='Spread Ponderado')
150     ]
151
152     # Legenda
153     legenda = ax1.legend(handles=elementos_legenda,
154                          loc='upper left', frameon=True, framealpha=0.95,
155                          edgecolor=cores['grid'], fontsize=10,
156                          facecolor=cores['fundo'])
157
158     for text in legenda.get_texts():
159         text.set_color(cores['texto'])
160
161     # --- RODAPE ---
162     fonte_texto = "Fonte: Dados internos"
163     fig.text(0.5, 0.02, fonte_texto, ha='center', va='bottom',
164              fontsize=9, style='italic', color=cores['texto'])
165
166     # --- AJUSTES FINAIS DE LAYOUT ---
167     plt.tight_layout()
```

```
168     plt.subplots_adjust(bottom=0.12)  # Espaco para fonte
169
170     # Salvar figura
171     if salvar_png:
172         fig.savefig(salvar_png, dpi=300, bbox_inches='tight',
173                     facecolor=cores['fundo'], edgecolor='none',
174                     transparent=False)
175
176     plt.show()
177
178     # --- ESTATISTICAS ---
179     stats = {
180         'periodo': periodo,
181         'captacao_media': plot_df["Captacao Liquida (bi)"].mean
                ()
182         'spread_medio': plot_df["Spread ponderado"].mean(),
183         'correlacao': np.corrcoef(plot_df["Captacao Liquida (bi
                )"],
184                                   plot_df["Spread ponderado"])
                                      [0,1]
185     }
186
187     print(f" Estatisticas {periodo}:")
188     print(f" Captacaoo media: R$ {stats['captacao_media']:+.2f}
            bi")
189     print(f" Spread medio: {stats['spread_medio']:.2f}%")
190     print(f" Correlacao: {stats['correlacao']:.2%}")
191
192     return fig, (ax1, ax2), stats
193
194 # Grafico com eixos em preto
195 fig, axes, stats = plot_fluxo_vs_spread_p(
196     tabela,
197     salvar_png="grafico_fluxo_vs_spread.png",
198     estilo='seaborn-v0_8-whitegrid'
199 )
```

APÊNDICE C – Código Python para Análise da Captação Líquida vs Spread Ponderado

```
1
2  # ANALISE: SPREAD VS CAPTACAO - MODELO WLS
3
4
5  import numpy as np
6  import pandas as pd
7  import matplotlib.pyplot as plt
8  import statsmodels.api as sm
9  from scipy.stats import pearsonr, spearmanr
10 from statsmodels.tsa.stattools import adfuller
11 from scipy.ndimage import uniform_filter1d
12 import warnings
13 warnings.filterwarnings('ignore')
14
15
16 # FUNCAO AUXILIAR PARA FORMATAR P-VALORES
17
18
19 def formatar_p_valor(p_valor, limite=0.0001):
20     """
21     Formata p-valor para exibicao adequada
22
23     Parametros:
24     -----------
25     p_valor : float
26         Valor p a ser formatado
27     limite : float
28         Limite para usar notacao cientifica (padrao: 0.0001)
29
30     Retorna:
31     --------
32     str : p-valor formatado
33     """
```

```
    if pd.isnull(p_valor):
        return "N/A"

    if p_valor < limite:
        # Notacao cientifica para valores muito pequenos
        return f"{p_valor:.2e}"
    elif p_valor < 0.001:
        # 6 casas decimais para valores pequenos
        return f"{p_valor:.6f}"
    elif p_valor < 0.01:
        # 4 casas decimais
        return f"{p_valor:.4f}"
    else:
        # 3 casas decimais para valores maiores
        return f"{p_valor:.3f}"


# FUNCAO PRINCIPAL DE ANALISE


def analise_definitiva(tabela):
    """
    Analise completa da relacao entre Spread e Captacao Liquida
    Retorna modelo WLS final e todas as estatisticas
    """

    print("="*80)
    print("ANALISE: SPREAD VS CAPTACAO - MODELO WLS")
    print("="*80)

    # Preparacao inicial dos dados
    df = tabela.copy()
    df = df.rename(columns={
        "Captacao Liquida (bi)": "fluxo_bi",
        "Spread ponderado": "spread_pct"
    })
    df = df.sort_values("Data").dropna(subset=["fluxo_bi", "
        spread_pct"]).reset_index(drop=True)


    # 1.1 ESTATISTICAS DESCRITIVAS

```

```
75
76      print("\n" + "ESTATISTICAS DESCRITIVAS DAS SERIES ORIGINAIS
            ")
77      print("-" * 50)
78
79      def calcular_estatisticas(serie, nome):
80          stats = {
81              "Media": serie.mean(),
82              "Mediana": serie.median(),
83              "Desvio Padrao": serie.std(),
84              "Minimo": serie.min(),
85              "Maximo": serie.max(),
86              "Amplitude": serie.max() - serie.min(),
87              "Coef. Variacao": (serie.std() / serie.mean()) *
                    100,
88              "n": len(serie.dropna())
89          }
90          return pd.Series(stats, name=nome)
91
92      stats_fluxo = calcular_estatisticas(df["fluxo_bi"], "
            Captacao Liquida (R$ bi)")
93      stats_spread = calcular_estatisticas(df["spread_pct"], "
            Spread Ponderado (%)")
94
95      stats_comparativo = pd.concat([stats_fluxo, stats_spread],
            axis=1)
96      print(stats_comparativo.round(3))
97
98      # 2. ANALISE DE ESTACIONARIEDADE (ADF)
99
100
101     print("\n" + "TESTE DE ESTACIONARIEDADE - AUGMENTED DICKEY-
            FULLER (ADF)")
102     print("-" * 60)
103
104     def teste_adf_completo(serie, nome):
105         resultado = adfuller(serie.dropna(), autolag="AIC")
106         return {
107             "Serie": nome,
108             "Estatistica ADF": resultado[0],
109             "p-valor": resultado[1],
```

```
110             "Estacionaria": "SIM" if resultado[1] < 0.05 else "
                    NAO",
111             "Numero Observacoes": resultado[3]
112         }
113
114     adf_fluxo = teste_adf_completo(df["fluxo_bi"], "Captacao
            Liquida")
115     adf_spread = teste_adf_completo(df["spread_pct"], "Spread
            Ponderado")
116
117     df_adf = pd.DataFrame([adf_fluxo, adf_spread])
118
119     # Formatar p-valores no DataFrame ADF
120     df_adf_display = df_adf.copy()
121     df_adf_display['p-valor'] = df_adf_display['p-valor'].apply
            (formatar_p_valor)
122     print(df_adf_display.to_string(index=False))
123
124
125     # 3. ANALISE DE CORRELACAO COM DEFASAGENS
126
127
128     print("\n" + "ANALISE DE CORRELACAO COM DEFASAGENS")
129     print("-" * 40)
130
131     def analise_correlacao_defasagens(df, x_col, y_col, max_lag
            =12):
132         resultados = []
133         for lag in range(0, max_lag + 1):
134             temp_df = df[[x_col, y_col]].copy()
135             temp_df[x_col] = temp_df[x_col].shift(lag)
136             temp_df = temp_df.dropna()
137
138             if len(temp_df) > 5:
139                 r_pearson, p_pearson = pearsonr(temp_df[x_col],
                        temp_df[y_col])
140                 r_spearman, p_spearman = spearmanr(temp_df[
                        x_col], temp_df[y_col])
141
142                 resultados.append({
143                     'Lag': lag,
144                     'Pearson': r_pearson,
```

```
145                     'p_Pearson': p_pearson,
146                     'Spearman': r_spearman,
147                     'p_Spearman': p_spearman,
148                     'n': len(temp_df),
149                     'Significativo': 'SIM' if p_pearson < 0.05
                            else 'NAO'
150                 })
151
152         return pd.DataFrame(resultados)
153
154     # Correlacao Spread - Fluxo
155     corr_spread_fluxo = analise_correlacao_defasagens(df, "
        spread_pct", "fluxo_bi")
156
157     # Formatar p-valores para exibicao
158     corr_spread_fluxo_display = corr_spread_fluxo.copy()
159     corr_spread_fluxo_display['p_Pearson'] =
        corr_spread_fluxo_display['p_Pearson'].apply(
        formatar_p_valor)
160     corr_spread_fluxo_display['p_Spearman'] =
        corr_spread_fluxo_display['p_Spearman'].apply(
        formatar_p_valor)
161
162     print("\n CORRELACAO: Spread - Captacao (Spread antecede
        Captacao)")
163     display(corr_spread_fluxo_display.round(4))
164
165     # Encontrar melhor lag
166     melhor_lag_idx = corr_spread_fluxo['Pearson'].abs().idxmax
        ()
167     melhor_lag = corr_spread_fluxo.loc[melhor_lag_idx]
168
169     # Formatar p-valor do melhor lag
170     p_melhor_lag_formatado = formatar_p_valor(melhor_lag['
        p_Pearson'])
171
172     print(f"\n MELHOR LAG ENCONTRADO: {int(melhor_lag['Lag'])}
        periodos")
173     print(f"   Correlacao Pearson: {melhor_lag['Pearson']:.4f}
        (p = {p_melhor_lag_formatado})")
174     print(f"  Correlacao Spearman: {melhor_lag['Spearman']:.4f}
        ")
```

```
175     print(f"   Observacoes: {int(melhor_lag['n'])}")
176
177
178     # 4. DESSAZONALIZACAO DAS SERIES
179
180
181     print("\n" + " DESSAZONALIZACAO DAS SERIES")
182     print("-" * 30)
183
184     # Adicionar componente mensal
185     df["mes"] = df["Data"].dt.month
186
187     def dessazonalizar_serie(y, meses):
188         """Remove sazonalidade mensal via regressao OLS"""
189         X_m = pd.get_dummies(meses.astype(int), prefix="m",
                drop_first=True, dtype=float)
190         X = sm.add_constant(X_m, has_constant="add")
191
192         reg_df = pd.concat([y.astype(float), X], axis=1).dropna
                ()
193         yv = reg_df.iloc[:, 0].to_numpy(dtype=float)
194         Xv = reg_df.iloc[:, 1:].to_numpy(dtype=float)
195
196         modelo = sm.OLS(yv, Xv).fit()
197         residuos = pd.Series(modelo.resid, index=reg_df.index,
                name=f"{y.name}_res")
198
199         # Reconstruir serie no indice original
200         serie_dessazonalizada = pd.Series(np.nan, index=y.index
                , name=f"{y.name}_res")
201         serie_dessazonalizada.loc[residuos.index] = residuos
202
203         return serie_dessazonalizada
204
205     # Aplicar dessazonalizacao
206     df["fluxo_res"] = dessazonalizar_serie(df["fluxo_bi"], df["
            mes"])
207     df["spread_res"] = dessazonalizar_serie(df["spread_pct"],
            df["mes"])
208
209
210     # 5. COMPARACAO PRE/POS-DESSAZONALIZACAO (VERSAO CORRIGIDA)
```

```
211
212
213     print("\n" + "COMPARACAO: PRE vs POS-DESSAZONALIZACAO")
214     print("-" * 45)
215
216     # Correlacao antes da dessazonalizacao
217     temp_original = df[["fluxo_bi", "spread_pct"]].dropna()
218     r_antes, p_antes = pearsonr(temp_original["spread_pct"],
            temp_original["fluxo_bi"])
219
220     # Correlacao apos dessazonalizacao
221     temp_dessaz = df[["fluxo_res", "spread_res"]].dropna()
222     r_depois, p_depois = pearsonr(temp_dessaz["spread_res"],
            temp_dessaz["fluxo_res"])
223
224     # Formatar p-valores
225     p_antes_formatado = formatar_p_valor(p_antes)
226     p_depois_formatado = formatar_p_valor(p_depois)
227
228     # CORRECAO: Calcular melhoria baseada na forca da
            correlacao (valor absoluto)
229     melhoria_magnitude = ((abs(r_depois) - abs(r_antes)) / abs(
            r_antes)) * 100
230
231     print(f" CORRELACAO:")
232     print(f"    Antes da dessazonalizacao:  r = {r_antes:.4f} (p
             = {p_antes_formatado})")
233     print(f"    Apos a dessazonalizacao:    r = {r_depois:.4f} (
            p = {p_depois_formatado})")
234
235     if abs(melhoria_magnitude) > 1:  # Mostrar se houver
            mudanca relevante
236         if melhoria_magnitude > 0:
237             print(f" Fortalecimento da relacao:  {
                    melhoria_magnitude:+.1f}%")
238             print(f" A relacao spread-captacao ficou {
                    melhoria_magnitude:.1f}% mais forte apos
                    dessazonalizacao")
239         else:
240             print(f" Enfraquecimento da relacao:  {
                    melhoria_magnitude:+.1f}%")
241     else:
```

```
242         print(f"   Estabilidade: {melhoria_magnitude:+.1f}% (
                mudanca insignificante)")
243
244     # Mostrar a melhoria na significancia estatistica
245     melhoria_significancia = ((p_antes - p_depois) / p_antes) *
            100
246     print(f"  Melhoria na significancia: {
            melhoria_significancia:+.1f}% (p-valor reduziu)")
247
248
249
250     # 6. ANALISE DE CORRELACAO COM SERIES DESSAZONALIZADAS
251
252
253     print("\n" + "CORRELACOES COM SERIES DESSAZONALIZADAS")
254     print("-" * 45)
255
256     corr_dessaz = analise_correlacao_defasagens(df, "spread_res
            ", "fluxo_res")
257
258     # Formatar p-valores para exibicao
259     corr_dessaz_display = corr_dessaz.copy()
260     corr_dessaz_display['p_Pearson'] = corr_dessaz_display['
            p_Pearson'].apply(formatar_p_valor)
261     corr_dessaz_display['p_Spearman'] = corr_dessaz_display['
            p_Spearman'].apply(formatar_p_valor)
262
263     print("CORRELACAO: Spread Dessazonalizado - Captacao
            Dessazonalizada")
264     display(corr_dessaz_display.round(4))
265
266     # Encontrar melhor lag nas series dessazonalizadas
267     melhor_lag_dessaz_idx = corr_dessaz['Pearson'].abs().idxmax
            ()
268     melhor_lag_dessaz = corr_dessaz.loc[melhor_lag_dessaz_idx]
269     lag_otimo = int(melhor_lag_dessaz['Lag'])
270
271
272     p_valor_formatado = formatar_p_valor(melhor_lag_dessaz['
            p_Pearson'])
273
```

```
274     print(f"\n MELHOR LAG (Dessazonalizado): {lag_otimo}
            periodos")
275     print(f"   Correlacao Pearson: {melhor_lag_dessaz['Pearson
            ']:.4f} (p = {p_valor_formatado})")
276     print(f"   Observacoes: {int(melhor_lag_dessaz['n'])}")
277
278
279     # 7. MODELAGEM WLS (MINIMOS QUADRADOS PONDERADOS)
280
281
282     print("\n" + "MODELO WLS - MINIMOS QUADRADOS PONDERADOS")
283     print("-" * 45)
284
285     # Preparar dados para o modelo
286     df_modelo = df.copy()
287     df_modelo[f"spread_res_lag{lag_otimo}"] = df_modelo["
            spread_res"].shift(lag_otimo)
288     df_modelo["fluxo_res_lag1"] = df_modelo["fluxo_res"].shift
            (1)
289
290     # Dataset final para modelagem
291     dados_modelo = df_modelo[["fluxo_res", "fluxo_res_lag1", f"
            spread_res_lag{lag_otimo}"]].dropna()
292
293     # Variaveis do modelo
294     X = sm.add_constant(dados_modelo[["fluxo_res_lag1", f"
            spread_res_lag{lag_otimo}"]])
295     y = dados_modelo["fluxo_res"]
296
297
298     # 7.1 ESTIMACAO WLS COM PESOS OTIMIZADOS
299
300
301     # Estimativa preliminar OLS para calcular pesos
302     modelo_preliminar = sm.OLS(y, X).fit()
303     residuos_absolutos = pd.Series(np.abs(modelo_preliminar.
            resid), index=dados_modelo.index)
304
305     # Suavizar a escala dos residuos
306     escala_suavizada = pd.Series(
307         uniform_filter1d(residuos_absolutos.values, size=3,
                mode="nearest"),
```

```
        index=residuos_absolutos.index
    )

    # Calcular pesos (inverso da variancia)
    pesos = 1.0 / np.clip(escala_suavizada, 1e-8, None)**2
    pesos = pd.Series(pesos, index=dados_modelo.index, name="pesos")

    # Modelo WLS final
    modelo_wls = sm.WLS(y, X, weights=pesos).fit()

    print(f"\n MODELO WLS ESTIMADO COM SUCESSO")
    print(f"   Periodo analisado: {len(y)} observacoes")
    print(f"   Lag otimo utilizado: {lag_otimo} periodos")
    print(f"   Variaveis: Captacao(t-1) + Spread(t-{lag_otimo})")


    # 8. RESULTADOS E INTERPRETACAO


    print("\n" + " RESULTADOS DO MODELO WLS")
    print("-" * 30)
    print(modelo_wls.summary())


    # 9. METRICAS DE IMPACTO PARA O NEGOCIO


    print("\n" + " IMPACTO PARA O NEGOCIO")
    print("-" * 25)

    coeficiente_spread = modelo_wls.params[2]  # spread_res_lag{lag_otimo} na posicao 2
    r2_wls = modelo_wls.rsquared

    print(f" DESCOBERTA PRINCIPAL:")
    print(f"   'O spread dessazonalizado defasado em {lag_otimo} periodos e um")
    print(f"    preditor forte e robusto da captacao liquida,'")
```

```
344     print(f"    explicando {r2_wls*100:.1f}% de sua variacao.'"
            )
345     print(f"")
346     print(f" IMPACTO ECONOMICO:")
347     print(f"   Coeficiente Spread: {coeficiente_spread:.2f}")
348     print(f"   Interpretacao: Aumento de 1% no spread resulta
            em")
349     print(f"     R$ {abs(coeficiente_spread):.2f} bi de
            captacao apos {lag_otimo} periodos")
350     print(f"")
351     print(f" APLICACOES PRATICAS:")
352     print(f"   1. FERRAMENTA DE PREVISAO: Modelo operacional
            para planejamento")
353     print(f"   2. GESTAO DE SPREAD: Impacto quantificado no
            longo prazo")
354     print(f"   3. VANTAGEM COMPETITIVA: Insight sobre dinamica
            de mercado")
355     print(f"")
356     print(f" QUALIDADE DO MODELO:")
357     print(f"   R^2: {r2_wls:.3f} ({r2_wls*100:.1f}% de
            variancia explicada)")
358     print(f"   Poder Preditivo: {'EXCELENTE' if r2_wls > 0.8
            else 'BOM'}")
359     print(f"   Robustez: Validado por multiplos testes
            estatisticos")
360
361
362     # 10. ANALISE DE ROBUSTEZ
363
364
365     print("\n" + "ANALISE DE ROBUSTEZ - COMPARACAO ENTRE
            METODOS")
366     print("-" * 50)
367
368     # OLS padrao
369     modelo_ols = sm.OLS(y, X).fit()
370
371     # OLS com HAC: Usar indices numericos
372     modelo_ols_hac = modelo_ols.get_robustcov_results(cov_type=
            "HAC", maxlags=3)
373
374     # RLM (Robust Linear Model): Usar indices numericos
```

```
375     modelo_rlm = sm.RLM(y, X, M=sm.robust.norms.HuberT()).fit()
376
377     # Usar indices numericos para acessar os parametros
378     comparacao_modelos = pd.DataFrame({
379         'Metodo': ['WLS', 'OLS+HAC', 'RLM'],
380         'R^2': [modelo_wls.rsquared, modelo_ols.rsquared, np.
                nan],
381         'Coef_Const': [
382             modelo_wls.params[0],  # const
383             modelo_ols_hac.params[0],  # const
384             modelo_rlm.params[0]   # const
385         ],
386         'Coef_AR1': [
387             modelo_wls.params[1],  # fluxo_res_lag1
388             modelo_ols_hac.params[1],  # fluxo_res_lag1
389             modelo_rlm.params[1]   # fluxo_res_lag1
390         ],
391         'Coef_Spread': [
392             modelo_wls.params[2],  # spread_res_lag{lag_otimo}
393             modelo_ols_hac.params[2],  # spread_res_lag{
                    lag_otimo}
394             modelo_rlm.params[2]   # spread_res_lag{lag_otimo}
395         ],
396         'P_valor_Spread': [
397             modelo_wls.pvalues[2],  # p-value do spread
398             modelo_ols_hac.pvalues[2],  # p-value do spread
399             modelo_rlm.pvalues[2]   # p-value do spread
400         ],
401         'Observacoes': [len(y), len(y), len(y)]
402     })
403
404     # Formatar p-valores para exibicao na comparacao
405     comparacao_modelos_display = comparacao_modelos.copy()
406     comparacao_modelos_display['P_valor_Spread'] =
           comparacao_modelos_display['P_valor_Spread'].apply(
407         lambda x: formatar_p_valor(x) if pd.notnull(x) else 'N/
                A'
408     )
409
410     print(comparacao_modelos_display.round(4))
411
412     print(f"\n CONCLUSAO SOBRE ROBUSTEZ:")
```

```
413     print(f"   Todos os metodos confirmam a relacao
            significativa entre")
414     print(f"   spread e captacao com defasagem de {lag_otimo}
            periodos")
415     print(f"   Coeficiente do spread varia entre {
            comparacao_modelos['Coef_Spread'].min():.2f} e {
            comparacao_modelos['Coef_Spread'].max():.2f}")
416
417
418     # 11. RETORNO DOS RESULTADOS
419
420
421     resultados = {
422         'dataframe': df,
423         'modelo_wls': modelo_wls,
424         'modelo_ols': modelo_ols,
425         'modelo_ols_hac': modelo_ols_hac,
426         'modelo_rlm': modelo_rlm,
427         'lag_otimo': lag_otimo,
428         'correlacao_melhor_lag': melhor_lag_dessaz['Pearson'],
429         'r2_wls': r2_wls,
430         'coeficiente_spread': coeficiente_spread,
431         'estatisticas_descritivas': stats_comparativo,
432         'resultados_adf': df_adf,
433         'comparacao_modelos': comparacao_modelos,
434         'dados_modelo': dados_modelo,
435         'X': X,
436         'y': y
437     }
438
439     print("\n" + "="*80)
440     print(" ANALISE CONCLUIDA COM SUCESSO")
441     print("="*80)
442     print(f" RESUMO FINAL:")
443     print(f" Lag otimo: {lag_otimo} periodos")
444     print(f" R^2 WLS: {r2_wls:.3f} ({r2_wls*100:.1f}% de
            variancia explicada)")
445     print(f" Coeficiente Spread: {coeficiente_spread:.2f}")
446     print(f" Impacto: 1% spread - R$ {abs(coeficiente_spread)
            :.2f} bi captacao em {lag_otimo} periodos")
447     print(f" Significancia: p = {formatar_p_valor(modelo_wls.
            pvalues[2])}")
```

```
448
449     return resultados
450
451
452 # EXECUCAO DA ANALISE
453
454
455 if __name__ == "__main__":
456     print(" COMO USAR:")
457     print("1. Certifique-se de que seu DataFrame tem as colunas
            :")
458     print("   - 'Data' (datetime)")
459     print("   - 'Captacao Liquida (bi)' (float)")
460     print("   - 'Spread ponderado' (float)")
461     print("2. Execute: resultados = analise_definitiva(
            seu_dataframe)")
462     print("3. Acesse os resultados: resultados['modelo_wls'],
            resultados['r2_wls'], etc.")
463     print("\n" + "="*80)
464
465
466 # RESULTADOS CONSOLIDADOS
467
468 # 1. Execute a analise completa
469 resultados = analise_definitiva(tabela)
470
471 # 2. Acesse os principais resultados
472 print(f" Lag otimo: {resultados['lag_otimo']}")
473 print(f" R^2 WLS: {resultados['r2_wls']:.3f}")
474 print(f" Coeficiente Spread: {resultados['coeficiente_spread
        ']:.2f}")
475
476 # 3. Use o modelo para previsoes
477 modelo_wls = resultados['modelo_wls']
478 X_novo = resultados['X']  # Estrutura para novas previsoes
479 # previsoes = modelo_wls.predict(X_novo)
480
481 # 4. Acesse os dados processados
482 df_processado = resultados['dataframe']
483 dados_modelo = resultados['dados_modelo']
```

APÊNDICE D – Código Python Gráfico Evolução do Spread Ponderado

```
1
2  import pandas as pd
3  import numpy as np
4  import matplotlib.pyplot as plt
5  import matplotlib.dates as mdates
6  from matplotlib.ticker import PercentFormatter
7  from dateutil import parser
8  from pandas.tseries.offsets import MonthEnd
9  from pathlib import Path
10
11
12 # Parametros
13
14 ARQUIVO_XLSX = Path(r"\idex_cdi_geral_datafile.xlsx")
15 ABA          = "Detalhado"
16 COL_DATA     = "Data"
17 COL_SPREAD   = "Spread de compra (%)"
18 COL_PESO     = "Peso no indice (%)"
19 COL_EMISSOR  = "Emissor"
20 EMISSORES_DISTRESS = ["Lame", "Light", "Aeris", "Viveo", "
       AMERICANAS SA"]
21
22 # Escolha UMA das opcoes abaixo
23 DATA_INICIAL   = "2022-04-01"          # ex.: "2020-01-01" ou
       "01/01/2020" ou None
24 DATA_FINAL     = None            # ex.: "2025-09-30" ou
       "30/09/2025" ou None
25 ULTIMOS_N_MESES = None          # ex.: 24  (se definido, ignora
        as datas acima)
26
27 # Aparencia
28 cor_spread = "#1f77b4"
29 arq_pdf    = Path(r"\spread_ponderado_mensal.pdf")
30
31
```

```python
# Helpers

def to_fraction(val):
    """Converte 12.3 ou '12,3%' -> 0.123; 0.123 fica 0.123."""
    if pd.isna(val):
        return np.nan
    if isinstance(val, str):
        s = val.strip().replace("%", "").replace(" ", "")
        s = s.replace(".", "").replace(",", ".") if s.count(",")==1 and s.count(".")>1 else s.replace(",", ".")
        try:
            num = float(s)
        except Exception:
            return np.nan
        return num / 100.0
    try:
        num = float(val)
    except Exception:
        return np.nan
    return num / 100.0 if num > 1 else num

def wavg(g):
    w = g["Peso_frac"].astype(float)
    x = g["Spread_frac"].astype(float)
    m = (~w.isna()) & (~x.isna())
    w, x = w[m], x[m]
    if len(w) == 0 or np.isclose(w.sum(), 0.0):
        return np.nan
    return np.average(x, weights=w)

def parse_date_safe(s):
    if s is None:
        return None
    if isinstance(s, (pd.Timestamp, pd.DatetimeIndex)):
        return pd.to_datetime(s)
    try:
        return pd.to_datetime(parser.parse(str(s), dayfirst=("/" in str(s))))
    except Exception:
        return None

def aplicar_filtro_periodo(df_mes):
```

```
72      """Aplica filtro por DATA_INICIAL/DATA_FINAL ou
            ULTIMOS_N_MESES."""
73      if df_mes.empty:
74          return df_mes
75
76      # limites totais do dataset
77      min_all = df_mes["Mes"].min()
78      max_all = df_mes["Mes"].max()
79
80      if ULTIMOS_N_MESES and ULTIMOS_N_MESES > 0:
81          # pega ate o fim do ultimo mes disponivel
82          end = (max_all + MonthEnd(0))
83          start = end - pd.DateOffset(months=ULTIMOS_N_MESES-1)
84      else:
85          start = parse_date_safe(DATA_INICIAL)
86          end   = parse_date_safe(DATA_FINAL)
87          # normaliza: inicio do mes para start, fim do mes para
                end
88          start = pd.Timestamp(start).to_period("M").to_timestamp
                () if start is not None else min_all
89          end   = (pd.Timestamp(end).to_period("M").to_timestamp
                () + MonthEnd(0)) if end is not None else (max_all +
                 MonthEnd(0))
90
91      mask = (df_mes["Mes"] >= start) & (df_mes["Mes"] <= end)
92      return df_mes.loc[mask].copy()
93
94  def escolher_intervalo_xtick(n_meses):
95      """Retorna um step de meses para os ticks do eixo X,
            conforme o tamanho do periodo."""
96      if n_meses <= 12:   return 1    # mensal
97      if n_meses <= 36:   return 3    # trimestral
98      if n_meses <= 84:   return 6    # semestral
99      return 12                       # anual
100
101
102 # 1) Ler planilha e calcular media ponderada mensal
103
104 usecols = [COL_DATA, COL_SPREAD, COL_PESO, COL_EMISSOR]
105 df = pd.read_excel(ARQUIVO_XLSX, sheet_name=ABA, usecols=
        usecols)
106
```

```
107 df[COL_DATA] = pd.to_datetime(df[COL_DATA], errors="coerce")
108 df = df.dropna(subset=[COL_DATA])
109
110 # Exclui emissores em distress
111 if EMISSORES_DISTRESS:
112     df = df[~df[COL_EMISSOR].astype(str).str.contains("|".join(
        EMISSORES_DISTRESS), case=False, na=False)]
113
114 # Converte para fracao
115 df["Spread_frac"] = df[COL_SPREAD].apply(to_fraction)
116 df["Peso_frac"]   = df[COL_PESO].apply(to_fraction)
117
118 # Agrega por mes (inicio do mes)
119 df["Mes"] = df[COL_DATA].dt.to_period("M").dt.to_timestamp()
120
121 spread_mensal = (
122     df.groupby("Mes").apply(wavg)
123       .rename("Spread_ponderado_frac")
124       .reset_index()
125       .sort_values("Mes")
126 )
127 spread_mensal["Spread_ponderado_%"] = 100 * spread_mensal["
    Spread_ponderado_frac"]
128
129
130 # 2) Aplicar filtro de periodo
131
132 spread_filtrado = aplicar_filtro_periodo(spread_mensal)
133
134
135 # 3) Plot - um unico eixo
136
137 plt.rcParams.update({
138     "figure.figsize": (12, 6),
139     "axes.titlesize": 14,
140     "axes.labelsize": 12,
141     "xtick.labelsize": 10,
142     "ytick.labelsize": 10,
143     "lines.linewidth": 2.2,
144     "lines.markersize": 5.5,
145 })
146
```

```
147 fig, ax = plt.subplots()
148
149 linha, = ax.plot(
150     spread_filtrado["Mes"],
151     spread_filtrado["Spread_ponderado_%"],
152     label="Spread de Compra (Ponderado)",
153     color=cor_spread
154 )
155
156 ax.set_title("Evolucao do Spread Ponderado (DI+) - Media Mensal
        ")
157 ax.set_xlabel("Data")
158 ax.set_ylabel("Spread Ponderado (%)", color=cor_spread)
159 ax.tick_params(axis="y", labelcolor=cor_spread)
160 ax.yaxis.set_major_formatter(PercentFormatter(xmax=100))
161
162 # ticks X dinamicos conforme numero de meses
163 n_meses = max(1, len(spread_filtrado))
164 step = escolher_intervalo_xtick(n_meses)
165 ax.xaxis.set_major_locator(mdates.MonthLocator(interval=step))
166 ax.xaxis.set_major_formatter(mdates.DateFormatter("%b/%Y"))
167 plt.setp(ax.get_xticklabels(), rotation=45, ha="right")
168
169 # grade e anotacao do ultimo ponto
170 ax.grid(True, which="major", linestyle="--", alpha=0.35)
171 if not spread_filtrado.empty:
172     x_last = spread_filtrado["Mes"].iloc[-1]
173     y_last = spread_filtrado["Spread_ponderado_%"].iloc[-1]
174     ax.scatter([x_last], [y_last], color=cor_spread)
175     ax.annotate(f"{y_last:,.2f}%", xy=(x_last, y_last),
176                 xytext=(10, 8), textcoords="offset points",
                        fontsize=10)
177
178 ax.legend(loc="upper left")
179 fig.tight_layout()
180 plt.savefig(arq_pdf, format="pdf")
181 plt.show()
182 print(f"Figura salva em: {arq_pdf}")
```

APÊNDICE E – Código Python Machine Learning Spread Puro

```python
import os, sys
if sys.platform.startswith("win"):
    os.environ.setdefault("JOBLIB_MULTIPROCESSING", "0")  # desativa loky/multiprocessing
    os.environ.setdefault("LOKY_MAX_CPU_COUNT", "1")      # impede chamada 'wmic'
    # (opcional) evita over-subscription de BLAS
    os.environ.setdefault("OPENBLAS_NUM_THREADS", "1")
    os.environ.setdefault("MKL_NUM_THREADS", "1")
    os.environ.setdefault("NUMEXPR_NUM_THREADS", "1")
    os.environ.setdefault("OMP_NUM_THREADS", "1")

# IMPORTS

import warnings; warnings.filterwarnings("ignore")

from pathlib import Path
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.ticker import PercentFormatter
from dateutil import parser
from pandas.tseries.offsets import MonthEnd

from sklearn.decomposition import PCA
from sklearn.linear_model import ElasticNetCV
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.metrics import mean_absolute_error, mean_squared_error


# CONFIGURACOES

ARQUIVO_XLSX = Path(r"\idex_cdi_geral_datafile.xlsx")
```

```
ABA           = "Detalhado"
COL_DATA      = "Data"
COL_SPREAD    = "Spread de compra (%)"
COL_PESO      = "Peso no indice (%)"
COL_EMISSOR   = "Emissor"
EMISSORES_DISTRESS = ["Lame", "Light", "Aeris", "Viveo", "
    AMERICANAS SA"]

# Filtro temporal
DATA_INICIAL     = "2022-04-01"      # "YYYY-MM-DD" ou None
SPLIT_PCTS = (0.6, 0.2, 0.2) # treino, validacao, teste
MAX_LAG_FEATURES= 15
MIN_DIAS_TREINO = 100
HORIZONTES = (5, 22, 66)


# HELPERS

def to_fraction(val):
    if pd.isna(val): return np.nan
    if isinstance(val, str):
        s = val.strip().replace("%","").replace(" ","").replace
            ("-","-")
        # Trata '1.234,56' e '1234,56'
        if s.count(",")==1 and s.count(".")>1:
            s = s.replace(".", "").replace(",", ".")
        else:
            s = s.replace(",", ".")
        try: x = float(s)
        except: return np.nan
        return x/100.0
    try: x = float(val)
    except: return np.nan
    return x/100.0 if abs(x)>1 else x

def create_micro_features(df, target="spread_wavg"):
    if df.empty: return df
    d = df.copy()
    d["vol_5d"]  = d[target].diff().rolling(5,  min_periods=2).
       std()
    d["vol_22d"] = d[target].diff().rolling(22, min_periods=5).
       std()
```

```
71      d["spread_rollmean_5"]  = d[target].rolling(5).mean()
72      d["spread_rollmean_22"] = d[target].rolling(22).mean()
73      d["spread_trend_5"]  = d[target].diff(5)
74      d["spread_trend_22"] = d[target].diff(22)
75      d = d.fillna(method="ffill").fillna(0.0)
76      return d
77
78  def create_lagged_features(df, target="spread_wavg", max_lag
        =15):
79      """Gera lags do alvo e lags das micro-features (lag1)."""
80      if df.empty: return df
81      d = df.copy()
82      n = len(d)
83      max_lag_calc = max(3, min(max_lag, int(0.15*n)))
84      # lags do alvo (benchmark)
85      for k in range(1, min(6, max_lag_calc+1)):
86          d[f"{target}_lag{k}"] = d[target].shift(k)
87      # lags de micro-features (lag1)
88      micro_cols = [c for c in d.columns if c.startswith("vol_")
            or "rollmean" in c or "trend" in c]
89      for c in micro_cols:
90          d[f"{c}_lag1"] = d[c].shift(1)
91      return d
92
93
94  def etapa3_simplificada(
95      train, valid, test, data, target,
96      model_benchmark, model_correcao_micro,
97      features_benchmark, features_micro_model,
98      horizontes=(5, 22, 66),
99      topk_by_h={5: 50, 22: 30, 66: 25},
100     q_grid=(0.90, 0.95, 0.98, 0.995),
101     relatorio_xlsx='relatorio_etapa3_simplificada.xlsx',
102     gerar_graficos=True,
103     random_state=42
104 ):
105     """
106     ETAPA 3 compacta (apenas Spread):
107
108     """
109
```

```
110     assert train.index.max() < valid.index.min() <= test.index.
            min(), \
111         "Ordem temporal esperada: TREINO - VALIDACAO - TESTE"
112 
113     def add_nowcast(df):
114         out = df.copy()
115         try:
116             nb = model_benchmark.predict(out[features_benchmark
                    ])
117             nc = model_correcao_micro.predict(out[
                    features_micro_model])
118             out['nowcast_final'] = nb + nc
119         except Exception:
120             out['nowcast_final'] = 0.0
121         return out
122 
123     train_e3 = add_nowcast(train)
124     valid_e3 = add_nowcast(valid)
125     test_e3  = add_nowcast(test)
126 
127     base_pool = [c for c in data.columns if c != target and not
            str(c).endswith('_futuro')]
128     if 'nowcast_final' not in base_pool:
129         base_pool.append('nowcast_final')
130 
131     # grades pequenas de HGB
132     grid_by_h = {
133         5:  [
134             dict(learning_rate=0.04, max_depth=3,
                    min_samples_leaf=20, l2_regularization=1e-3,
                    loss='squared_error'),
135             dict(learning_rate=0.03, max_depth=3,
                    min_samples_leaf=30, l2_regularization=3e-3,
                    loss='squared_error'),
136         ],
137         22: [
138             dict(learning_rate=0.03, max_depth=3,
                    min_samples_leaf=40, l2_regularization=1e-2,
                    loss='absolute_error'),
139             dict(learning_rate=0.02, max_depth=3,
                    min_samples_leaf=60, l2_regularization=3e-2,
                    loss='absolute_error'),
```

```
        ],
        66: [
            dict(learning_rate=0.02, max_depth=3,
                min_samples_leaf=60, l2_regularization=3e-2,
                loss='absolute_error'),
            dict(learning_rate=0.015,max_depth=3,
                min_samples_leaf=80, l2_regularization=5e-2,
                loss='absolute_error'),
        ]
    }

    def build_delta(df, H, feat_cols):
        y_future = df[target].shift(-H)
        y_delta  = (y_future - df[target]).rename(f'delta_h{H}'
            )
        m = y_delta.notna()
        X = df.loc[m, feat_cols]
        y   = y_delta.loc[m]
        y_t = df.loc[m, target].rename('y_t')
        y_true = y_future.loc[m].rename('y_true')
        return X, y  , y_t, y_true

    def select_by_spearman(X_tr, y_tr  , k):
        scores = []
        for f in X_tr.columns:
            try:
                rho = spearmanr(X_tr[f].values, y_tr .values,
                    nan_policy='omit').correlation
            except Exception:
                rho = 0.0
            if not np.isfinite(rho): rho = 0.0
            scores.append((f, abs(rho)))
        scores.sort(key=lambda z: z[1], reverse=True)
        return [f for f,_ in scores[:min(k, len(scores))]]

    def metrics(y, yhat):
        y, yhat = np.asarray(y), np.asarray(yhat)
        mae  = mean_absolute_error(y, yhat) * 1e4    # bps
        rmse = np.sqrt(mean_squared_error(y, yhat)) * 1e4
        r2   = r2_score(y, yhat)
        pear = pearsonr(y, yhat)[0] if len(y) > 2 else np.nan
```

```
175         spear = spearmanr(y, yhat, nan_policy='omit').
               correlation if len(y) > 2 else np.nan
176         return mae, rmse, r2, pear, spear
177
178     resultados, importancias, series_plot = [], {}, {}
179
180     for H in horizontes:
181         Xtr_pool = train_e3[base_pool]
182         _, y_full, _, _ = build_delta(train_e3, H, base_pool)
183         feats = select_by_spearman(Xtr_pool.loc[y_full.index
               ], y_full, topk_by_h.get(H, 30))
184
185         # datasets
186         X_tr, y_tr , y_tr_t, _   = build_delta(train_e3, H,
               feats)
187         X_va, y_va , y_va_t, y_va_true = build_delta(valid_e3,
                H, feats)
188         X_te, y_te , y_te_t, y_te_true = build_delta(test_e3,
                H, feats)
189         if len(X_tr) < 50 or len(X_te) < 10:
190             print(f"[H={H}] Dados insuficientes; pulando ")
191             continue
192
193         best = (np.inf, None, None, None)  # (MAE_val, modelo,
                _cap , calibrador)
194         for cfg in grid_by_h.get(H, grid_by_h[66]):
195             hgb = HistGradientBoostingRegressor(random_state=
                   random_state, **cfg)
196             hgb.fit(X_tr, y_tr )
197
198             abs   = np.abs(y_tr .values)  # quantis extraidos
                   do TREINO
199             for q in q_grid:
200                  _cap   = float(np.quantile(abs , q))
201                 d_va = np.clip(hgb.predict(X_va), - _cap ,
                        _cap )
202                 y_va_hat = (y_va_t.values + d_va)
203
204                 m = np.isfinite(y_va_true.values) & np.isfinite
                       (y_va_hat)
205                 if m.sum() >= 10:
```

```
206                     cal = LinearRegression().fit(y_va_hat[m].
                            reshape(-1,1), y_va_true.values[m])
207                 else:
208                     cal = None
209
210                 y_va_pred = (cal.predict(y_va_hat.reshape(-1,1)
                        ).reshape(-1)
211                              if cal is not None else y_va_hat)
212
213                 mae_val = mean_absolute_error(y_va_true.values[
                        m], y_va_pred[m]) if m.sum()>=10 \
214                             else mean_absolute_error(y_va_true.
                                values, y_va_pred)
215
216                 if mae_val < best[0]:
217                     best = (mae_val, hgb,  _cap , cal)
218
219         _, hgb_best,  _cap_best , cal_best = best
220
221         # TESTE
222         d_te = np.clip(hgb_best.predict(X_te), - _cap_best ,
                _cap_best )
223         y_te_hat = (y_te_t.values + d_te)
224         y_te_pred = (cal_best.predict(y_te_hat.reshape(-1,1)).
                reshape(-1)
225                      if cal_best is not None else y_te_hat)
226
227         mae, rmse, r2, pear, spear = metrics(y_te_true.values,
                y_te_pred)
228         resultados.append({
229             'Horizonte_dias': H,
230             'n_treino': len(X_tr), 'n_valid': len(X_va), '
                    n_teste': len(X_te),
231             'TopK': len(feats), ' _cap (bps)':  _cap_best *1e4,
232             'MAE_bps': mae, 'RMSE_bps': rmse, 'R2': r2,
233             'Pearson_r': pear, 'Spearman_ ': spear
234         })
235
236             with parallel_backend("threading", n_jobs=1):
237                 imp = permutation_importance(
238                     hgb_best, X_va, y_va ,
```

```
                    n_repeats=5, scoring='neg_mean_absolute_error',
                    n_jobs=1, random_state=random_state
                )
            importancias[H] = (pd.DataFrame({'Feature': X_va.columns,
                                              'Importance_Mean': imp.importances_mean,
                                              'Importance_Std': imp.importances_std})
                               .sort_values('Importance_Mean', ascending=False).head(15))
        except Exception:
            importancias[H] = pd.DataFrame({'Feature': [], 'Importance_Mean': [], 'Importance_Std': []})

        # GRAFICO
        idx = y_te_true.index
        series_plot[H] = pd.DataFrame({
            'Real (t+H)': pd.Series(y_te_true.values, index=idx),
            'Previsto'  : pd.Series(y_te_pred,        index=idx)
        })

    # RESULTADOS
    resumo = pd.DataFrame(resultados).sort_values('Horizonte_dias')
    print("\n RESUMO - Etapa 3")
    if not resumo.empty:
        print(resumo.to_string(index=False, float_format=lambda v: f"{v:,.4f}"))
    else:
        print("Nenhum resultado (verifique dados/intervalos).")

    try:
        with pd.ExcelWriter(relatorio_xlsx) as w:
            if not resumo.empty:
                resumo.to_excel(w, "Resumo", index=False)
```

```
268             for H, dfimp in importancias.items():
269                 if dfimp is not None and not dfimp.empty:
270                     dfimp.to_excel(w, f"Importancia_H{H}",
                            index=False)
271         print(f"Relatorio salvo em: {relatorio_xlsx}")
272     except Exception as e:
273         print(f"Aviso: falha ao salvar Excel ({e}).")
274 
275     # GRAFICO REAL VS PREVISTO
276         try: plt.style.use('default')
277         except: pass
278         for H, dfp in series_plot.items():
279             if dfp is None or dfp.empty: continue
280             plt.figure(figsize=(12,5))
281             (100*dfp['Real (t+H)']).plot(linewidth=1.2, label='
                Real (t+H)')
282             (100*dfp['Previsto']).plot(linewidth=1.2, linestyle
                ='--', label='Previsto')
283             ax = plt.gca(); ax.yaxis.set_major_formatter(
                PercentFormatter(xmax=100))
284             plt.title(f'Previsao {H} dias a frente - Teste (
                apenas Spread)')
285             plt.ylabel('Spread (%)'); plt.legend(); plt.grid(
                True, alpha=0.25)
286             plt.tight_layout(); plt.show()
287 
288     return resumo, importancias, series_plot
289 
290 # PIPELINE
291 
292     # 1) Carrega planilha e calcula Spread Ponderado diario
293     try:
294         use_cols = [COL_DATA_SPREAD, COL_SPREAD, COL_PESO,
                COL_EMISSOR]
295         raw = pd.read_excel(ARQUIVO_XLSX, sheet_name=ABA_SPREAD
                , usecols=use_cols)
296         raw[COL_DATA_SPREAD] = pd.to_datetime(raw[
                COL_DATA_SPREAD], errors="coerce")
297         raw[COL_EMISSOR]     = raw[COL_EMISSOR].astype("string"
                )
298         raw = raw.dropna(subset=[COL_DATA_SPREAD])
299 
```

```
300         # remove distress
301         if EMISSORES_DISTRESS:
302             pat = "|".join(EMISSORES_DISTRESS)
303             mask_distress = raw[COL_EMISSOR].astype(str).str.
                    contains(pat, case=False, na=False)
304             print(f"   Removendo {mask_distress.sum()} entradas
                    distress.")
305             raw = raw[~mask_distress]
306 
307         raw["Spread_frac"] = raw[COL_SPREAD].apply(to_fraction)
                .astype(float)
308         raw["Peso_frac"]   = raw[COL_PESO].apply(to_fraction).
                astype(float)
309 
310         raw = raw[(raw["Peso_frac"] > 0) & (raw["Spread_frac"].
                notna())].copy()
311         if len(raw) > 0:
312             q_low, q_high = raw["Spread_frac"].quantile([0.02,
                    0.98])
313             raw["Spread_frac"] = raw["Spread_frac"].clip(q_low,
                    q_high)
314 
315         raw["Data_dia"] = raw[COL_DATA_SPREAD].dt.floor("D")
316         conta_dia = raw.groupby("Data_dia")[COL_EMISSOR].
                transform("nunique")
317         raw = raw[conta_dia >= K_MIN_EMISSORES]
318 
319         raw["Spread_x_Peso"] = raw["Spread_frac"] * raw["
                Peso_frac"]
320         g = raw.groupby("Data_dia")
321         agg = g.agg(Spread_x_Peso_sum=('Spread_x_Peso', 'sum'),
322                     Peso_frac_sum=('Peso_frac', 'sum'))
323         y_daily = (agg['Spread_x_Peso_sum']/agg['Peso_frac_sum'
                ]).rename("spread_wavg").sort_index().to_frame()
324         print(f"   Serie agregada: {len(y_daily)} dias")
325     except Exception as e:
326         print(f"   ERRO ao processar spread: {e}")
327         y_daily = pd.DataFrame()
328 
329     # 2) Features micro e lags
330     if not y_daily.empty:
```

```
331         micro = create_micro_features(y_daily, target="
               spread_wavg")
332         data  = create_lagged_features(micro, target="
               spread_wavg", max_lag=MAX_LAG_FEATURES)
333         data  = data.fillna(method="bfill").fillna(method="
               ffill").fillna(0)
334         print(f"   Dataset: {len(data)} dias, {len(data.columns
               )} colunas")
335     else:
336         data = pd.DataFrame()
337         print("   ERRO: Sem dados para modelagem.")
338 
339     # 3) Filtro temporal
340     if not data.empty and DATA_INICIAL:
341         try:
342             data = data[data.index >= pd.to_datetime(
                   DATA_INICIAL)].copy()
343             print(f"   Mantendo dados a partir de {DATA_INICIAL
                   }: {len(data)} dias")
344         except Exception as e:
345             print(f"   Aviso: Falha ao aplicar filtro de data:
                   {e}")
346 
347     # 4) Split temporal
348     if data.empty or len(data) < MIN_DIAS_TREINO:
349         print(f"\nERRO: Dados insuficientes ({len(data)}),
               minimo {MIN_DIAS_TREINO}.")
350         raise SystemExit
351 
352     n = len(data)
353     tr_size = int(n * SPLIT_PCTS[0]); va_size = int(n *
           SPLIT_PCTS[1])
354     train = data.iloc[:tr_size].copy()
355     valid = data.iloc[tr_size:tr_size+va_size].copy()
356     test  = data.iloc[tr_size+va_size:].copy()
357     print(f"   Treino:     {train.index[0].date()} - {train.
           index[-1].date()} ({len(train)})")
358     print(f"   Validacao:  {valid.index[0].date()} - {valid.
           index[-1].date()} ({len(valid)})")
359     print(f"   Teste:      {test.index[0].date()}  - {test.
           index[-1].date()}  ({len(test)})")
360 
```

```
361     target = "spread_wavg"
362
363     # 5) Definicao de features por etapa (apenas spread)
364     allowed_suffixes = [f"_lag{k}" for k in range(1,
            MAX_LAG_FEATURES+1)] + ["_lag1"]  # micro-lag1
365     all_features = [c for c in data.columns if c != target and
            any(c.endswith(s) for s in allowed_suffixes)]
366     features_benchmark   = [f for f in all_features if f.
            startswith("spread_wavg_lag")]    # E1
367     features_micro_model = [f for f in all_features if not f.
            startswith("spread_wavg_lag")] # E2
368
369     print(f"   Benchmark (lags spread): {len(features_benchmark
            )}")
370     print(f"   Correcao micro:          {len(
            features_micro_model)}")
371
372     # 6) Treinamento Etapas 1 e 2
373     # E1: benchmark
374     model_benchmark = Pipeline([
375         ("scaler", StandardScaler()),
376         ("elasticnet", ElasticNetCV(cv=5, n_jobs=1, max_iter
                =2000,
377                                     l1_ratio
                                        =[0.05,0.1,0.3,0.5,0.7,0.9,0.95])
                                        )
378     ])
379     model_benchmark.fit(train[features_benchmark], train[target
            ])
380     # residuos no treino
381     train_pred_bench = model_benchmark.predict(train[
            features_benchmark])
382     train_residuos   = train[target] - train_pred_bench
383
384     # E2: correcao (apenas micro)
385     model_micro = Pipeline([
386         ("scaler", StandardScaler()),
387         ("elasticnet", ElasticNetCV(cv=5, n_jobs=1, max_iter
                =2000,
388                                     l1_ratio
                                        =[0.05,0.1,0.3,0.5,0.7,0.9,0.95])
                                        )
```

```
389     ])
390     model_micro.fit(train[features_micro_model], train_residuos
            )
391
392     # 7) Resumo E1/E2 em teste (coerencia com Modelo Final)
393     test_pred_bench = model_benchmark.predict(test[
            features_benchmark])
394     test_pred_corr  = model_micro.predict(test[
            features_micro_model])
395     test_pred_final = test_pred_bench + test_pred_corr
396
397     mae_bps  = mean_absolute_error(test[target],
            test_pred_final) * 1e4
398     rmse_bps = np.sqrt(mean_squared_error(test[target],
            test_pred_final)) * 1e4
399     r2_t     = r2_score(test[target], test_pred_final)
400     print(f" MAE Teste:  {mae_bps:.2f} bps  |  RMSE: {rmse_bps
            :.2f} bps  |  $R^2$: {r2_t:.3f}")
401
402     # 8) Etapa 3: graficos por horizonte
403     resumo_e3, imp_e3, series_e3 = etapa3_simplificada(
404         train, valid, test, data, target,
405         model_benchmark, model_micro,
406         features_benchmark, features_micro_model,
407         horizontes=HORIZONTES,
408         relatorio_xlsx='relatorio_etapa3_spread_only.xlsx',
409         gerar_graficos=True
410     )
```

APÊNDICE F – Código Python Machine Learning Spread + Captação Líquida

```
if sys.platform.startswith("win"):
    os.environ.setdefault("JOBLIB_MULTIPROCESSING", "0")  # evita loky -> wmic
    os.environ.setdefault("LOKY_MAX_CPU_COUNT", "1")
    os.environ.setdefault("OPENBLAS_NUM_THREADS", "1")
    os.environ.setdefault("MKL_NUM_THREADS", "1")
    os.environ.setdefault("NUMEXPR_NUM_THREADS", "1")
    os.environ.setdefault("OMP_NUM_THREADS", "1")

import warnings
warnings.filterwarnings("ignore")

from pathlib import Path
import re
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.ticker import PercentFormatter
from scipy.stats import spearmanr, pearsonr

from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import ElasticNetCV, LinearRegression
from sklearn.ensemble import HistGradientBoostingRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.inspection import permutation_importance
from joblib import parallel_backend

# CONFIGURACOES
ARQUIVO_XLSX     = Path(r"C:\Users\gabri\OneDrive\area de Trabalho\TCC\Gabriel_v0.3\idex_cdi_geral_datafile.xlsx")
ARQUIVO_CAPTACAO = Path(r"C:\Users\gabri\OneDrive\Area de Trabalho\TCC\Gabriel_v0.3\estudo_jgp.xlsx")

```

```
ABA_SPREAD = "Detalhado"
COL_DATA_SPREAD = "Data"
COL_SPREAD      = "Spread de compra (%)"
COL_PESO        = "Peso no indice (%)"
COL_EMISSOR     = "Emissor"

EMISSORES_DISTRESS = ["Lame", "Light", "Aeris", "Viveo", "
    AMERICANAS SA"]
K_MIN_EMISSORES    = 10

ABA_CAPTACAO = "base"

DATA_INICIAL       = "2022-04-01"
SPLIT_PCTS         = (0.6, 0.2, 0.2)    # treino, validacao,
    teste
MAX_LAG_FEATURES   = 15
MIN_DIAS_TREINO    = 100
HORIZONTES         = (5, 22, 66)
RELATORIO_E3_XLSX  = "relatorio_etapa3_unificado.xlsx"
RANDOM_STATE       = 42

# HELPERS
def to_fraction(val):
    if pd.isna(val): return np.nan
    if isinstance(val, str):
        s = val.strip().replace("%","").replace("-","-").
            replace(" ","")
        if s.count(",")==1 and s.count(".")>1: s = s.replace(".
            ","").replace(",",".")
        else: s = s.replace(",",".")
        try: x = float(s)
        except: return np.nan
        return x/100.0
    try: x = float(val)
    except: return np.nan
    return x/100.0 if abs(x)>1 else x

def parse_brl(x):
    if pd.isna(x): return np.nan
    s = str(x).strip().replace("   ","-")
    neg = s.startswith("(") and s.endswith(")")
    if neg: s = s[1:-1]
```

```
70      s = re.sub(r"[^\d,.\-]", "", s)
71      if "," in s and s.count(",")==1:
72          s = s.replace(".","").replace(",",".")
73      try: v = float(s)
74      except: v = np.nan
75      return -v if neg else v
76
77  def carregar_e_processar_captacao(arquivo_path, sheet_name):
78      """Processa captacao macro mensal (agregados) e deriva
            variaveis macro."""
79      try:
80          df = pd.read_excel(arquivo_path, sheet_name=sheet_name,
                header=0)
81      except Exception as e:
82          print(f"   Aviso: falha ao ler captacao ({e}); seguindo
                sem macro.")
83          return pd.DataFrame()
84
85      def pick_col(df, guesses, fallback_idx):
86          cols = list(df.columns)
87          for c in cols:
88              if any(g.lower() in str(c).lower() for g in guesses
                    ):
89                  return c
90          return cols[fallback_idx]
91
92      col_data = pick_col(df, ["data"], 0)
93      col_ap   = pick_col(df, ["aplic", "entrada", "subs"], 3)
94      col_rg   = pick_col(df, ["resg", "saida", "saida"], 4)
95
96      df = df[[col_data, col_ap, col_rg]].copy()
97      df.columns = ["Data","Aplic","Resg"]
98      df["Data"] = pd.to_datetime(df["Data"], errors="coerce")
99      df["Aplic"] = df["Aplic"].apply(parse_brl)
100     df["Resg"]  = df["Resg"].apply(parse_brl)
101
102     # Normaliza sinais
103     if df["Aplic"].dropna().median() < 0: df["Aplic"] = -df["
            Aplic"]
104     if df["Resg"].dropna().median()  < 0: df["Resg"]  = -df["
            Resg"]
105
```

```
106     df = df.dropna(subset=["Data"])
107     df["Mes"] = df["Data"].dt.to_period("M").dt.to_timestamp()
108 
109     agg = (df.groupby("Mes")
110               .agg(Captacao=("Aplic","sum"), Resgate=("Resg","
                     sum"), Dias_Captacao=("Data","nunique"))
111               .reset_index().sort_values("Mes"))
112     agg["Captacao_Liquida_bi"] = (agg["Captacao"] - agg["
          Resgate"]) / 1e9
113     # Derivadas macro (regimes e tendencias)
114     agg["CL_rolling_3m"] = agg["Captacao_Liquida_bi"].rolling
          (3, min_periods=1).mean()
115     agg["CL_rolling_6m"] = agg["Captacao_Liquida_bi"].rolling
          (6, min_periods=1).mean()
116     agg["CL_tendencia"]  = agg["Captacao_Liquida_bi"].diff(3)
117     if agg["Captacao_Liquida_bi"].std() and agg["
          Captacao_Liquida_bi"].std() > 0:
118         agg["CL_zscore"] = (agg["Captacao_Liquida_bi"] - agg["
              Captacao_Liquida_bi"].mean())/agg["
              Captacao_Liquida_bi"].std()
119     else:
120         agg["CL_zscore"] = 0.0
121     q33, q66 = agg["Captacao_Liquida_bi"].quantile([0.33,
          0.66])
122     agg["CL_regime"] = np.select(
123         [agg["Captacao_Liquida_bi"] < q33, agg["
              Captacao_Liquida_bi"] > q66],
124         [-1, 1], default=0
125     )
126     print(f"   Captacao processada: {len(agg)} meses | {agg['
          Mes'].min().date()} - {agg['Mes'].max().date()}")
127     return agg.rename(columns={"Mes":"Data_Mes"})
128 
129 def incorporar_captacao_macro_mensal(Xy_diario,
      df_captacao_mensal):
130     """merge_asof (mensal-diario) + ffill de macro, com prefixo
          'macro_'"""
131     if df_captacao_mensal.empty: return Xy_diario
132     out = pd.merge_asof(
133         Xy_diario.sort_index(),
134         df_captacao_mensal.sort_values("Data_Mes"),
135         left_index=True,
```

```
136         right_on="Data_Mes",
137         direction="backward"
138     )
139     macro_cols = [c for c in df_captacao_mensal.columns if c !=
            "Data_Mes"]
140     ren = {c: f"macro_{c}" for c in macro_cols}
141     out = out.rename(columns=ren).drop(columns="Data_Mes")
142     macro_cols_renamed = list(ren.values())
143     out[macro_cols_renamed] = out[macro_cols_renamed].ffill()
144     return out
145
146 def create_micro_features(df, target="spread_wavg"):
147     d = df.copy()
148     d["vol_5d"]  = d[target].diff().rolling(5,  min_periods=2).
            std()
149     d["vol_22d"] = d[target].diff().rolling(22, min_periods=5).
            std()
150     d["spread_rollmean_5"]  = d[target].rolling(5).mean()
151     d["spread_rollmean_22"] = d[target].rolling(22).mean()
152     d["spread_trend_5"]  = d[target].diff(5)
153     d["spread_trend_22"] = d[target].diff(22)
154     d = d.fillna(method="ffill").fillna(method="bfill")
155     return d
156
157 def create_lagged_features(df, target="spread_wavg", max_lag
        =15):
158     d = df.copy()
159     n = len(d)
160     max_lag_calc = max(3, min(max_lag, int(0.15*n)))
161     # lags do alvo (benchmark)
162     for k in range(1, min(6, max_lag_calc+1)):
163         d[f"{target}_lag{k}"] = d[target].shift(k)
164     # lags macro (lag1 + lags mensais nos casos relevantes)
165     macro_cols = [c for c in d.columns if c.startswith("macro_"
            )]
166     for c in macro_cols:
167         d[f"{c}_lag1"] = d[c].shift(1)
168         if "Captacao_Liquida_bi" in c:
169             for lag_m in [1,2,3,6]:
170                 lag_dias = lag_m*22
171                 if lag_dias <= int(0.25*n):
```

```
                    d[f"{c}_lag{lag_m}m"] = d[c].shift(lag_dias)
    # lags das micro-features (lag1)
    micro_cols = [c for c in d.columns if c.startswith("vol_") or "rollmean" in c or "trend" in c]
    for c in micro_cols:
        d[f"{c}_lag1"] = d[c].shift(1)
    return d

# ETAPA 3
def etapa3_simplificada(
    train, valid, test, data, target,
    model_benchmark, model_macro,
    features_benchmark, features_macro_model,
    horizontes=(5, 22, 66),
    topk_by_h={5: 50, 22: 30, 66: 25},
    q_grid=(0.90, 0.95, 0.98, 0.995),
    relatorio_xlsx='relatorio_etapa3_unificado.xlsx',
    gerar_graficos=True,
    random_state=42
):
    # 0) garantia temporal (anti-leakage)
    assert train.index.max() < valid.index.min() <= test.index.min(), \
        "Esperado: TREINO - VALIDACAO - TESTE"

    # 1) meta-feature (nowcast) das Etapas 1/2
    def add_nowcast(df):
        out = df.copy()
        try:
            nb = model_benchmark.predict(out[features_benchmark])
            nm = model_macro.predict(out[features_macro_model])
            out['nowcast_final'] = nb + nm
        except Exception:
            out['nowcast_final'] = 0.0
        return out
    train_e3 = add_nowcast(train)
    valid_e3 = add_nowcast(valid)
    test_e3  = add_nowcast(test)

    # pool de variaveis, garante nowcast
```

```
    base_pool = [c for c in data.columns if c != target and not
        str(c).endswith('_futuro')]
    if 'nowcast_final' not in base_pool:
        base_pool.append('nowcast_final')

    # pequenas grades por H
    grid_by_h = {
        5:  [dict(learning_rate=0.04, max_depth=3,
           min_samples_leaf=20, l2_regularization=1e-3,  loss='
           squared_error'),
             dict(learning_rate=0.03, max_depth=3,
                min_samples_leaf=30, l2_regularization=3e-3,
                loss='squared_error')],
        22: [dict(learning_rate=0.03, max_depth=3,
           min_samples_leaf=40, l2_regularization=1e-2,  loss='
           absolute_error'),
             dict(learning_rate=0.02, max_depth=3,
                min_samples_leaf=60, l2_regularization=3e-2,
                loss='absolute_error')],
        66: [dict(learning_rate=0.02, max_depth=3,
           min_samples_leaf=60, l2_regularization=3e-2,  loss='
           absolute_error'),
             dict(learning_rate=0.015,max_depth=3,
                min_samples_leaf=80, l2_regularization=5e-2,
                loss='absolute_error')]
    }

    def build_delta(df, H, feat_cols):
        y_future = df[target].shift(-H)
        y_delta  = (y_future - df[target]).rename(f'delta_h{H}'
           )
        m = y_delta.notna()
        X = df.loc[m, feat_cols]
        y   = y_delta.loc[m]
        y_t = df.loc[m, target].rename('y_t')
        y_true = y_future.loc[m].rename('y_true')
        return X, y  , y_t, y_true

    def select_by_spearman(X_tr, y_tr  , k):
        scores = []
        for f in X_tr.columns:
```

```
237             try: rho = spearmanr(X_tr[f].values, y_tr .values,
                    nan_policy='omit').correlation
238             except Exception: rho = 0.0
239             if not np.isfinite(rho): rho = 0.0
240             scores.append((f, abs(rho)))
241         scores.sort(key=lambda z: z[1], reverse=True)
242         return [f for f,_ in scores[:min(k, len(scores))]]
243
244     def metrics(y, yhat):
245         y, yhat = np.asarray(y), np.asarray(yhat)
246         mae  = mean_absolute_error(y, yhat) * 1e4
247         rmse = np.sqrt(mean_squared_error(y, yhat)) * 1e4
248         r2   = r2_score(y, yhat)
249         pear = pearsonr(y, yhat)[0] if len(y) > 2 else np.nan
250         spear = spearmanr(y, yhat, nan_policy='omit').
               correlation if len(y) > 2 else np.nan
251         return mae, rmse, r2, pear, spear
252
253     resultados, importancias, series_plot = [], {}, {}
254
255     for H in horizontes:
256         # 2) selecao por |  | no TREINO
257         Xtr_pool = train_e3[base_pool]
258         _, y_full, _, _ = build_delta(train_e3, H, base_pool)
259         feats = select_by_spearman(Xtr_pool.loc[y_full.index
               ], y_full, topk_by_h.get(H, 30))
260
261         # 3) datasets alinhados
262         X_tr, y_tr , y_tr_t, _   = build_delta(train_e3, H,
               feats)
263         X_va, y_va , y_va_t, y_va_true = build_delta(valid_e3,
                H, feats)
264         X_te, y_te , y_te_t, y_te_true = build_delta(test_e3,
                H, feats)
265         if len(X_tr) < 50 or len(X_te) < 10:
266             print(f"[H={H}] Dados insuficientes; pulando.");
                   continue
267
268         # 4) escolher (HGB +   -cap + calibracao) na VALIDACAO
269         best = (np.inf, None, None, None)  # (MAE_val, modelo,
                _cap , calibrador)
270         for cfg in grid_by_h.get(H, grid_by_h[66]):
```

```
271             hgb = HistGradientBoostingRegressor(random_state=
                    random_state, **cfg)
272             hgb.fit(X_tr, y_tr )
273
274             abs  = np.abs(y_tr .values)  #  -cap calculado
                    no TREINO
275             for q in q_grid:
276                  _cap  = float(np.quantile(abs , q))
277                 d_va = np.clip(hgb.predict(X_va), - _cap ,
                        _cap )
278                 y_va_hat = (y_va_t.values + d_va)
279
280                 # calibracao linear (NaN-safe) na validacao
281                 m = np.isfinite(y_va_true.values) & np.isfinite
                        (y_va_hat)
282                 if m.sum() >= 10:
283                     cal = LinearRegression().fit(y_va_hat[m].
                            reshape(-1,1), y_va_true.values[m])
284                 else:
285                     cal = None
286
287                 y_va_pred = (cal.predict(y_va_hat.reshape(-1,1)
                        ).reshape(-1)
288                              if cal is not None else y_va_hat)
289
290                 mae_val = mean_absolute_error(y_va_true.values[
                        m], y_va_pred[m]) if m.sum()>=10 \
291                          else mean_absolute_error(y_va_true.
                                values, y_va_pred)
292
293                 if mae_val < best[0]:
294                     best = (mae_val, hgb,  _cap , cal)
295
296         _, hgb_best,  _cap_best , cal_best = best
297
298         # 5) TESTE
299         d_te = np.clip(hgb_best.predict(X_te), - _cap_best ,
                    _cap_best )
300         y_te_hat = (y_te_t.values + d_te)
301         y_te_pred = (cal_best.predict(y_te_hat.reshape(-1,1)).
                reshape(-1)
302                      if cal_best is not None else y_te_hat)
```

```
303
304         mae, rmse, r2, pear, spear = metrics(y_te_true.values,
                y_te_pred)
305         resultados.append({
306             'Horizonte_dias': H,
307             'n_treino': len(X_tr), 'n_valid': len(X_va), '
                    n_teste': len(X_te),
308             'TopK': len(feats), ' _cap (bps)':   _cap_best  *1e4,
309             'MAE_bps': mae, 'RMSE_bps': rmse, 'R2': r2,
310             'Pearson_r': pear, 'Spearman_ ': spear
311         })
312
313         # 6) VALIDACAO
314         try:
315             with parallel_backend("threading", n_jobs=1):
316                 imp = permutation_importance(
317                     hgb_best, X_va, y_va  ,
318                     n_repeats=5, scoring='
                            neg_mean_absolute_error',
319                     n_jobs=1, random_state=random_state
320                 )
321             importancias[H] = (pd.DataFrame({'Feature': X_va.
                    columns,
322                                              'Importance_Mean':
                                                     imp.
                                                    importances_mean
                                                    ,
323                                              'Importance_Std':
                                                     imp.
                                                    importances_std
                                                    })
324                                .sort_values('Importance_Mean',
                                       ascending=False).head(15))
325         except Exception:
326             importancias[H] = pd.DataFrame({'Feature': [], '
                    Importance_Mean': [], 'Importance_Std': []})
327
328         # 7) TESTE
329         idx = y_te_true.index
330         series_plot[H] = pd.DataFrame({
331             'Real (t+H)': pd.Series(y_te_true.values, index=idx
                    ),
```

```
            'Previsto'  : pd.Series(y_te_pred,       index=idx)
        })

    # 8) EXPORT
    resumo = pd.DataFrame(resultados).sort_values('
       Horizonte_dias')
    print("\nRESUMO - Etapa 3 - SPREAD + CAPTACAO MACRO")
    if not resumo.empty:
        print(resumo.to_string(index=False, float_format=lambda
            v: f"{v:,.4f}"))
    else:
        print("Nenhum resultado (verifique dados/intervalos).")

    try:
        with pd.ExcelWriter(relatorio_xlsx) as w:
            if not resumo.empty: resumo.to_excel(w, "Resumo",
               index=False)
            for H, dfimp in importancias.items():
                if dfimp is not None and not dfimp.empty:
                    dfimp.to_excel(w, f"Importancia_H{H}",
                       index=False)
        print(f"Relatorio salvo em: {relatorio_xlsx}")
    except Exception as e:
        print(f"Aviso: falha ao salvar Excel ({e}).")

    # 9) GRAFICOS
    if gerar_graficos:
        try: plt.style.use('default')
        except: pass
        for H, dfp in series_plot.items():
            if dfp is None or dfp.empty: continue
            plt.figure(figsize=(12,5))
            (100*dfp['Real (t+H)']).plot(linewidth=1.2, label='
               Real (t+H)')
            (100*dfp['Previsto']).plot(linewidth=1.2, linestyle
               ='--', label='Previsto')
            ax = plt.gca(); ax.yaxis.set_major_formatter(
               PercentFormatter(xmax=100))
            plt.title(f'Previsao {H} dias a frente     Teste (
               SPREAD + MACRO)')
            plt.ylabel('Spread (%)'); plt.legend(); plt.grid(
               True, alpha=0.25)
```

```
365             plt.tight_layout(); plt.show()
366
367     return resumo, importancias, series_plot
368
369 # PIPELINE
370 print("INICIANDO PIPELINE UNIFICADO: SPREAD + CAPTACAO MACRO")
371
372 # 1) macro (mensal)
373 df_captacao_macro = carregar_e_processar_captacao(
    ARQUIVO_CAPTACAO, ABA_CAPTACAO)
374
375 # 2) spread detalhado
376 try:
377     use_cols = [COL_DATA_SPREAD, COL_SPREAD, COL_PESO,
        COL_EMISSOR]
378     raw = pd.read_excel(ARQUIVO_XLSX, sheet_name=ABA_SPREAD,
        usecols=use_cols)
379     raw[COL_DATA_SPREAD] = pd.to_datetime(raw[COL_DATA_SPREAD],
         errors="coerce")
380     raw[COL_EMISSOR]     = raw[COL_EMISSOR].astype("string")
381     raw = raw.dropna(subset=[COL_DATA_SPREAD])
382
383     if EMISSORES_DISTRESS:
384         pat = "|".join(EMISSORES_DISTRESS)
385         mask = raw[COL_EMISSOR].astype(str).str.contains(pat,
            case=False, na=False)
386         print(f"   Removendo {mask.sum()} entradas distress.")
387         raw = raw[~mask]
388
389     raw["Spread_frac"] = raw[COL_SPREAD].apply(to_fraction).
        astype(float)
390     raw["Peso_frac"]   = raw[COL_PESO].apply(to_fraction).
        astype(float)
391     raw = raw[(raw["Peso_frac"] > 0) & (raw["Spread_frac"].
        notna())].copy()
392
393     if len(raw) > 0:
394         q_low, q_high = raw["Spread_frac"].quantile([0.02,
            0.98])
395         raw["Spread_frac"] = raw["Spread_frac"].clip(q_low,
            q_high)
396
```

```
397     raw["Data_dia"] = raw[COL_DATA_SPREAD].dt.floor("D")
398     conta_dia = raw.groupby("Data_dia")[COL_EMISSOR].transform(
            "nunique")
399     raw = raw[conta_dia >= K_MIN_EMISSORES]
400
401     raw["Spread_x_Peso"] = raw["Spread_frac"] * raw["Peso_frac"
            ]
402     g = raw.groupby("Data_dia")
403     agg = g.agg(Spread_x_Peso_sum=('Spread_x_Peso', 'sum'),
404                 Peso_frac_sum=('Peso_frac', 'sum'))
405     y_daily = (agg['Spread_x_Peso_sum'] / agg['Peso_frac_sum'])
            .rename("spread_wavg").sort_index().to_frame()
406     print(f"   Serie agregada: {len(y_daily)} dias")
407 except Exception as e:
408     print(f"   ERRO ao processar spread: {e}")
409     y_daily = pd.DataFrame()
410
411 # 3) micro-features + macro merge_asof + lags
412 if not y_daily.empty:
413     micro = create_micro_features(y_daily, target="spread_wavg"
            )
414     Xy      = incorporar_captacao_macro_mensal(micro,
            df_captacao_macro) if not df_captacao_macro.empty else
            micro
415     data    = create_lagged_features(Xy, target="spread_wavg",
            max_lag=MAX_LAG_FEATURES)
416     data    = data.fillna(method="bfill").fillna(method="ffill")
            .fillna(0)
417     print(f"   Dataset com features: {len(data)} dias, {len(
            data.columns)} colunas")
418 else:
419     data = pd.DataFrame()
420     print("   ERRO: Sem dados para modelagem.")
421
422 # 4) filtro temporal e split
423 print("\n4. Filtro temporal e split...")
424 if not data.empty and DATA_INICIAL:
425     data = data[data.index >= pd.to_datetime(DATA_INICIAL)].
            copy()
426     print(f"   Mantendo dados a partir de {DATA_INICIAL}: {len(
            data)} dias")
427 if data.empty or len(data) < MIN_DIAS_TREINO:
```

```
428     raise SystemExit(f"Dados insuficientes ({len(data)}),
            minimo {MIN_DIAS_TREINO} dias.")
429
430 n = len(data)
431 tr_size = int(n*SPLIT_PCTS[0]); va_size = int(n*SPLIT_PCTS[1])
432 train = data.iloc[:tr_size].copy()
433 valid = data.iloc[tr_size:tr_size+va_size].copy()
434 test  = data.iloc[tr_size+va_size:].copy()
435 print(f"   Treino:    {train.index[0].date()} - {train.index
        [-1].date()} ({len(train)})")
436 print(f"   Validacao: {valid.index[0].date()} - {valid.index
        [-1].date()} ({len(valid)})")
437 print(f"   Teste:     {test.index[0].date()}  - {test.index
        [-1].date()}  ({len(test)})")
438
439 target = "spread_wavg"
440
441 # 5) listas de features
442 allowed_suffixes = [f"_lag{k}" for k in range(1,
        MAX_LAG_FEATURES+1)] + [f"_lag{k}m" for k in [1,2,3,6,12]] +
         ["_lag1"]
443 all_features = [c for c in data.columns if c != target and any(
        c.endswith(s) for s in allowed_suffixes)]
444
445 features_benchmark   = [f for f in all_features if f.startswith
        ("spread_wavg_lag")]
446 features_macro_model = [f for f in all_features if f not in
        features_benchmark]
447
448 print(f"   Benchmark (lags spread): {len(features_benchmark)}")
449 print(f"   Correcao macro/micro:    {len(features_macro_model)}
        ")
450
451 # 6) Treinamento Etapas 1 e 2 (ElasticNetCV, E1 nivel; E2
        residuos)
452 print("\n6. Treinando Etapas 1 e 2 (ElasticNetCV; n_jobs=1)..."
        )
453 model_benchmark = Pipeline([
454     ("scaler", StandardScaler()),
455     ("elasticnet", ElasticNetCV(cv=5, n_jobs=1, max_iter=2000,
456                                 l1_ratio
                                       =[0.05,0.1,0.3,0.5,0.7,0.9,0.95])
```

```
                                        )
])
model_benchmark.fit(train[features_benchmark], train[target])

train_pred_bench = model_benchmark.predict(train[
    features_benchmark])
train_residuos   = train[target] - train_pred_bench

model_macro = Pipeline([
    ("scaler", StandardScaler()),
    ("elasticnet", ElasticNetCV(cv=5, n_jobs=1, max_iter=2000,
                                l1_ratio
                                    =[0.05,0.1,0.3,0.5,0.7,0.9,0.95])
                                    )
])
model_macro.fit(train[features_macro_model], train_residuos)

# 7) Metricas de referencia (Etapas 1/2) no TESTE
test_pred_bench = model_benchmark.predict(test[
    features_benchmark])
test_pred_corr  = model_macro.predict(test[features_macro_model
    ])
test_pred_final = test_pred_bench + test_pred_corr

mae_bps  = mean_absolute_error(test[target], test_pred_final) *
     1e4
rmse_bps = np.sqrt(mean_squared_error(test[target],
    test_pred_final)) * 1e4
r2_t     = r2_score(test[target], test_pred_final)
print(f" MAE Teste: {mae_bps:.2f} bps  |  RMSE: {rmse_bps:.2f}
    bps  |  $R^2$: {r2_t:.3f}")

resumo_e3, imp_e3, series_e3 = etapa3_simplificada(
    train, valid, test, data, target,
    model_benchmark, model_macro,
    features_benchmark, features_macro_model,
    horizontes=HORIZONTES,
    relatorio_xlsx=RELATORIO_E3_XLSX,
    gerar_graficos=True,
    random_state=RANDOM_STATE
)
```