Sabrina Alencar Ferreira

APLICAÇÃO DE TÉCNICAS DE *MACHINE LEARNING* NA ANÁLISE DE CONTRATOS DE EMPRÉSTIMOS COMERCIAIS PARA EMPRESAS DO SEGMENTO DE CRÉDITO COMERCIAL ATACADO

Brasil

2022

Sabrina Alencar Ferreira

APLICAÇÃO DE TÉCNICAS DE *MACHINE LEARNING* NA ANÁLISE DE CONTRATOS DE EMPRÉSTIMOS COMERCIAIS PARA EMPRESAS DO SEGMENTO DE CRÉDITO COMERCIAL ATACADO

Dissertação apresentada ao Curso de Mestrado Acadêmico em Economia, Universidade de Brasília, como requisito parcial para a obtenção do título de Mestre em Economia

Universidade de Brasília - UnB

Faculdade de Administração Contabilidade e Economia - FACE

Departamento de Economia - ECO

Programa de Pós-Graduação

Orientador: Daniel Oliveira Cajueiro

Brasil

2022

Sabrina Alencar Ferreira
APLICAÇÃO DE TÉCNICAS DE *MACHINE LEARNING* NA ANÁLISE DE CONTRATOS DE EMPRÉSTIMOS COMERCIAIS PARA EMPRESAS DO SEGMENTO DE CRÉDITO COMERCIAL ATACADO/ Sabrina Alencar Ferreira. – Brasil, 2022-
127p. : il. (algumas color.) ; 30 cm.

Orientador: Daniel Oliveira Cajueiro

Dissertação (Mestrado) – Universidade de Brasília - UnB
Faculdade de Administração Contabilidade e Economia - FACE
Departamento de Economia - ECO
Programa de Pós-Graduação, 2022.

1. Palavra-chave1. 2. Palavra-chave2. 3. Palavra-chave3. II. Universidade de Brasília. III. Faculdade de Administração, Contabilidade e Economia - FACE. IV. Departamento de Economia IV. APLICAÇÃO DE TÉCNICAS DE *MACHINE LEARNING* NA ANÁLISE DE CONTRATOS DE EMPRÉSTIMOS COMERCIAIS PARA EMPRESAS DO SEGMENTO DE CRÉDITO COMERCIAL ATACADO

Sabrina Alencar Ferreira

APLICAÇÃO DE TÉCNICAS DE *MACHINE LEARNING* NA ANÁLISE DE CONTRATOS DE EMPRÉSTIMOS COMERCIAIS PARA EMPRESAS DO SEGMENTO DE CRÉDITO COMERCIAL ATACADO

Dissertação apresentada ao Curso de Mestrado Acadêmico em Economia, Universidade de Brasília, como requisito parcial para a obtenção do título de Mestre em Economia

Trabalho aprovado. Brasil, 28 de abril de 2022:

Daniel Oliveira Cajueiro
Orientador

Marina Delmondes de Carvalho Rossi
Convidado 1

Herbert Kimura
Convidado 2

Brasil
2022

Este trabalho é o resultado de um sonho que não foi sonhado, vivido e construído sozinho. Uma vida dedicada aos estudos e à certeza de grandes descobertas e infinitas inspirações, pois parar de estudar é parar de viver. O estudo sempre pode salvar vidas e renovar nosso espírito.

Agradecimentos

Aqui não poderei deixar de lembrar da presença de Deus junto às minhas escolhas, abençoando cada passo dentro e fora do mundo acadêmico, desde a primeira letra pronunciada ainda no antigo maternal até esse sonho atual realizado.

Aos meus familiares pela compreensão do tempo que tivemos que ficar longe uns dos outros, pela ausência em momentos comemorativos e pela escuta ativa sempre que o assunto mestrado era abordado nas conversas virtuais. Aqui eu conheci o poder da educação, seus recursos escassos, mas sua abundância de aplicação e é por eles que caminho saudosamente em busca de compromissos firmes com a pesquisa científica.

Aos colegas de trabalho que seguraram a barra quando necessário e entenderam a grandiosidade dessa imersão que começou aos finais de semanais e depois foram transformadas em longas férias e noites à disposição desse estudo.

Aos amigos pelo acolhimento quando estive sozinha, pela paciência quando precisei estar ausente e pelos convites nos momentos essenciais para continuar aguentando tanta pressão. Aqui eu aprendi a real necessidade de buscar o tão prezado equilíbrio nos pilares da vida. Alguns amigos estão intimamente ligados pelo coração, outros pela alma, outros nem sabemos explicar, esses eu deixo um cheiro tão grande que consigo adormecer somente em aproximar.

Ao amigo e professor Igor pela excelência em ensinar e inspirar um raciocínio lógico perspicaz, sem você eu jamais teria amado o *Python* como o amo hoje, obrigada pela compartilhamento incrível.

Aos colegas de mestrado pela companhia, entrega, estudos aos finais de semana, compartilhamento de material de estudo e angustias, em especial à Camila e Janaína pela inspiração e incentivo de entregar o certo e com qualidade em todos os semestres do curso.

Aos professores Marina, Kimura e Cajueiro pela excelência e compromisso dedicados ao ensino, vocês foram inspiração, ou seria melhor dizer "puro desafio". Algumas vezes trouxeram tantos questionamentos disruptivos naquele momento que muitas ideias precisaram ser desconstruídas para nascer um pensamento mais lógico e racional. Aos monitores Cayan Atreio Portela e Pedro Watuha pela disponibilidade, atenção e comprometimento em compartilhar conhecimento.

À motivação que sempre existiu aqui dentro de mim, seja pela paixão pelo aprendizado, pelo compartilhamento ou pelo novo. Que esse poder sempre se renove.

Amém.

"Nenhum projeto é viável se não começa a construir-se desde já: o futuro será o que começamos a fazer dele no presente."
(Içami Tiba)

Resumo

O presente estudo aplica técnicas supervisionadas de *Machine Learning* (ML) e compara sua performance em problemas de classificação para predizer respostas de interesse para o processo de monitoramento de contratos de crédito comercial no segmento atacado, modalidade de capital de giro. Verificou-se a necessidade de resolver a problemática das classes desbalanceadas, portanto Subamostras aleatórios simples foram geradas de modo que os grupos de análises estivessem balanceados em torno da variável resposta. Além de utilizar técnicas tradicionais de classificação, tais como *Logistic Regression*, este estudo explorou as técnicas de *Decision Tree, Bagging Classifiser, Random Forest, AdaBoost* e *Gradiente Boosting* para a previsão do atraso, ou seja, inadimplência pelo não pagamento das prestações devidas em prazo superior ou igual a 30, 60 e 90 dias. Além de *features*/variáveis relacionadas aos contratos e aos tomadores de crédito, foram inseridas variáveis macroeconômicas no aprendizado nos modelos, pois modelos que observam estas variáveis têm produzido melhores preditores para o risco de crédito. Para o subgrupo "Atraso $\geq$ 30 dias", o melhor desempenho foi atribuído ao algoritmo *Random Forest* que demonstrou uma predição aceitável, acima do nível de inadimplência da carteira analisada. Já para subgrupo "Atraso $\geq$ 60 dias"não foi possível identificar qual o melhor algoritmo, pois cada Subamostra gerada diversificou o desempenho dos modelos, já o subgrupo "Atraso $\geq$ 90 dias"dias apresentou algoritmos com os melhores desempenho dentro as simulações realizadas com destaque para o algoritmo *Adaboost*. A aplicação de técnicas para seleção de variáveis permitiu reduzir o *dataset* utilizado em cada simulação, o que implicou em melhor desempenho mais os modelos.

Palavras-chave: *Machine Learning*. Crédito comercial. Capital de Giro. Bases desbalanceadas. Poder preditivo.

Abstract

The present study applies supervised Machine Learning (ML) techniques and compares their performance in classification problems to predict responses of interest for the process of monitoring commercial credit contracts in the wholesale segment, working capital modality. The need to solve the problem of unbalanced classes was verified, therefore Simple random subsamples were generated so that the analysis groups were balanced around the response variable. In addition to using traditional classification techniques, such as Logistic Regression, this study explored Decision Tree, Bagging Classifiser, Random Forest, AdaBoost and Gradient Boosting techniques for the prediction of arrears, i.e., default by non-payment of installments due in 30, 60 and 90 days or more. In addition to eatures/variables related to contracts and borrowers, macroeconomic variables were inserted into the learning in the models, as models that observe these variables have produced better predictors for credit risk. For the subgroup "Delay $\geq$ 30 days", the best performance was attributed to the Random Forest algorithm, which demonstrated an acceptable prediction, above the default level of the analyzed portfolio. For the subgroup "Delay $\geq$ 60 days" it was not possible to identify the best algorithm, since each subsample generated diversified the models' performance, but the subgroup "Delay $\geq$ 90 days" presented algorithms with the best performances in the simulations performed, especially the Adaboost algorithm. The application of techniques for variable selection allowed the reduction of the dataset used in each simulation, which implied a better performance for the models.

Keywords: Machine Learning. Trade credit. Working Capital. Unbalanced bases. Predictive power.

Lista de ilustrações

Lista de quadros

Lista de abreviaturas e siglas

AUC	*Area Under the Curve*
Bacen	Banco Central do Brasil
BC	*Bagging Classifier*
B3	Brasil, Bolsa, Balcão
CNPJ	Cadastro Nacional da Pessoa Jurídica
CNAE	Classificação Nacional de Atividades Econômicas
Comef	Comitê de Estabilidade Financeira
CMN	Conselho Monetário Nacional
DT	*Decision Trees*
GB	*Gradiente Boosting*
IA	Inteligência Artificial
IBGE	Instituto Brasileiro de Geografia e Estatística
IPCA	Índice Nacional de Preços ao Consumidor Amplo
LR	*Logistic Regression*
MCC	Coeficiente de correlação de Matthews
MIC	*Multi Information Classification*
ML	*Machine Learning*
PD	Probabilidade de inadimplência
PIB	Produto Interno Bruto
PFI	*Permutation Feature Importance*
PJ	Pessoa Jurídica
RF	*Random Forest*
REF	Relatório de Estabilidade Financeira

ROC	*Receiver operating characteristic*
SCR	Sistema de Informações de Crédito [1]
SELIC	Sistema Especial de Liquidação e de Custódia
SFN	Sistema Financeiro Nacional
SGS	Sistema Gerenciador de Séries Temporais
SQR	*Soma de quadrados dos resíduos*
STV	Sem todas as Variáveis
SVM	*Support Vector Machiness*
TV	Todas as Variáveis
VBA	*Visual Basic for Application*

[1] Instrumento de registro gerido pelo Bacen e alimentado mensalmente pelas instituições financeiras.

Lista de símbolos

α	Letra grega minúscula Alfa
$\approx$	Relações Binárias
$<$	Relações Binárias
$\leq$	Relações Binárias
$>$	Relações Binárias
$\geq$	Relações Binárias
$\setminus$	Operadores Binários
β	Letra grega minúscula Beta
$\equiv$	Relações Binárias
λ	Letra grega minúscula lambda
$\prod$	Operador matemático
Σ	Letra grega maiúscula Sigma

Sumário

1 Introdução

Com a crise financeira global, acadêmicos e profissionais destacaram a gestão do risco de crédito em instituições financeiras como uma das mais importantes no sistema financeiro e o Acordo de Basileia ainda incluiu vários parâmetros de risco de crédito para quantificá-lo (XIA et al., 2021a), com isso as instituições financeiras começaram a empregar metodologias baseadas em *ratings* internos e assim construir seus próprios modelos quantitativos para estimar esses parâmetros de risco, dentro eles a PD que tem recebido destaque dos bancos e pesquisadores, uma vez que subsidia as tomadas de decisões no momento da concessão de empréstimos, no seu monitoramento e no cálculo da exigência de capital regulamentar (CROOK; EDELMAN; THOMAS, 2007).

Nosso trabalho compara abordagens populares de aprendizagem de máquina comumente utilizadas nos estudos sobre pontuação de crédito e seu monitoramento com modelos tradicionais, como por exemplo a *Logist Regression*, (XIA et al., 2021a) e (LESSMANN et al., 2015), e objetiva prevê a probabilidade de um contrato de empréstimo dentro do segmento de crédito comercial atacado, na modalidade capital de giro [1], inadimplir[2] durante a fase de amortização da dívida contratada, ou seja, após a concessão (fase de pontuação do crédito e atribuição do *rating*[3] para tomador e operação). Tal comparação utiliza bases balanceadas entre contratos considerados bons e ruins, apesar de ser bastante presente no segmento de grandes empresas a presença de desbalanceamento de classes, pois a quantidade de contratos ruins (que atrasaram) é bem menor quando comparado a quantidade de contratos bons (que não atrasaram), seleção de hiper-parâmetros dos modelos, avaliação das métricas de desempenho e inclusão de variáveis macroeconômicas além daquelas relacionadas ao tomador do crédito e à operação contratada, em qualquer período de tempo após a data de concessão do crédito, o que demonstrou que o desempenho dos algoritmos de ML [4], superou os modelos tradicionais.

O presente trabalho está relacionado com a literatura que aplica as técnicas ML às variáveis de predição de inadimplência e pontuação de crédito (VIEIRA et al., 2019), (PANDIMURUGAN et al., 2021), (XIA et al., 2021a) e (XIA et al., 2021b), cujos modelos utilizados contemplavam as técnicas de *Logist Regression* (VIEIRA et al., 2019), *Decision*

[1] Recursos que as empresas (independente do porte ou ramo de atuação) necessitam para arcar com os custos operacionais, obrigações financeiras cotidianas da administração de um negócio, para mantê-lo operando de maneira saudável.

[2] Descumprimento de uma obrigação previamente acordada, especialmente relacionada com a falta de pagamento de uma dívida seja num contrato de crédito comercial ou financiamento em geral.

[3] Conceito ou nível de classificação de risco de crédito de um tomador ou de uma operação que reflete a probabilidade de não cumprimento das obrigações de crédito

[4] Ramo da inteligência artificial baseado na ideia de que os sistemas podem aprender com os dados, identificar padrões e tomar decisões com o mínimo de intervenção humana. Estudo de algoritmos de computador que se aprimoram automaticamente por meio da experiência e do uso de dados.

Tree (XIA et al., 2021a), *Random Forest* (GOLBAYANI; FLORESCU; CHATTERJEE, 2020), *Bagging Classifier* (BREIMAN, 1996), (WANG et al., 2011), *Gradient Boosting* (FRIEDMAN, 2002), (LU; MAZUMDER, 2020) e *Adaboost* (HUANG; LING, 2005). Além disso, também se liga à literatura recente que utiliza variáveis macroeconômicas na modelagem com algoritmos de classificação e aspectos relacionados ao crédito, buscando capturar a influência do comportamento dessas variáveis na dinâmica do círculo empresarial no pagamento de empréstimos assim como realizado por Xia et al. (2021a) e Zhang e Thomas (2012), incorporando a volatilidade do índice IBOVESPA e da inflação - IPCA e o crescimento médio da taxa de juros SELIC e do PIB, pois modelos que observam estas variáveis têm produzido melhores preditores para o risco de crédito.

A Resolução do CMN n° 4.557/2017 define risco de crédito como a possibilidade de ocorrência de perdas associadas a: VI - custos de recuperação de exposições caracterizadas como ativos problemáticos nos termos do artigo 24 (BRASIL, 2017b).

Conforme o referido artigo, para fins do gerenciamento do risco de crédito, a exposição deve ser caracterizada como ativo problemático quando verificado pelo menos um dos seguintes eventos:

I - a respectiva obrigação está em atraso há mais de 90 dias; e

II - há indicativos de que a respectiva obrigação não será integralmente honrada sem que seja necessário recurso a garantias ou a colaterais.

Diante da necessidade de detectar previamente a falta de pagamento de prestações de contratos de empréstimos e consequentemente seu posterior inadimplemento, torna-se necessário adiantar-se quanto às análises que devem ser realizadas pelas unidades de gerenciamento de risco, controle interno e auditoria interna das instituições financeiras de modo que seja possível predizer comportamentos, desenvolver controles adequados ou ajustar os já existentes minimizando a possibilidade de perdas financeiras (APOSTOLIK; CHRISTOPHER; PETER, 2009).

Com um gerenciamento de risco eficaz e a introdução de novas técnicas de análise é possível mitigar o risco de crédito das operações de crédito comercial, evitar prováveis judicializações, melhorar acurácia da régua de cobrança, mitigar o risco operacional (erros normais em contratos) e subsidiar o controle da suficiência e constituição de garantias pactuadas, além de contribuir com a disseminação de novas técnicas de tratamento de dados entre os empregados das instituições financeiras (APOSTOLIK; CHRISTOPHER; PETER, 2009).

Tal avanço com a utilização dessas técnicas de ML oferece aos órgãos de controle interno e aos gestores de produtos um padrão consolidado de referência, proporcionando, assim, maior compreensão e controle das características dos clientes e dos controles mitigadores de risco de crédito (garantias) prevendo um possível inadimplemento, antes do

período já contemplado na legislação , Resolução CMN n° 4.557/2017, 90 dias de atraso (BRASIL, 2017b), pois a inadimplência reduz os resultados das instituições financeiras, o capital de referência e o potencial para ampliar negócios, assim como observado por Vieira et al. (2019).

Diante disso, construímos uma base de dados com mais de 1400 contratos de empréstimos firmados/concedidos entre janeiro de 2016 e dezembro de 2019, contemplando 12 variáveis explicativas relacionadas aos tomadores de crédito, às operações e ao contexto macroeconômico de exposição. O montante de crédito concedido nesse período é aproximadamente de R$ 4.9 bilhões de reais. Estimamos os nossos modelos para 3 subgrupos, atraso maior ou igual a 30, 60 e 90 dias, contemplando amostras aleatórios de classes balanceadas entre contratos ruins e bons. Os modelos foram comparados observando seus respectivos desempenhos, com ou sem algumas variáveis, conforme aplicação de técnicas de seleção de *features*, além da aplicação da validação cruzada no processo de seleção dos hiper-parâmetros.

O presente estudo foi distribuído da seguinte forma. No capítulo 2, descrevemos a ferramenta utilizada para aplicação dos modelos, o conjunto de dados utilizados, as variáveis disponíveis e respectivas análises realizadas, além das técnicas para seleção de *features*. No capítulo 3, apresentamos os modelos, as especificações utilizadas para as previsões, o procedimento adotado para escolher os hiper-parâmetros de cada modelo, as métricas para avaliar o desempenho e o processo de permuta das *features*/variáveis mais importantes. Mostramos então os nossos resultados gerais e examinamos os melhores modelos no Capitulo 4, considerando a utilização de todas variáveis disponíveis e a exclusão de algumas. Finalmente, no Capítulo 5, apresentamos uma breve descrição sobre as nossas conclusões.

3 Referencial Teórico

Neste capítulo, apresentamos os algoritmos testados (Quadro 4) para predizer a inadimplência dos contratos de empréstimo comercial. Este capítulo está dividido em três partes. Primeiro apresentamos breve descrição sobre os modelos selecionados, em seguida explicamos o processo de seleção de seus hiper-parâmetros por meio de validação cruzada, na sequência introduzimos a avaliação do desempenho dos modelos em cada subgrupo (atraso maior ou igual a 30, 60 e 90 dias).

Quadro 4 – Tipos de Classificadores

Tipo de classificador/Método	Nome do classificador
Linear	*Logist Regression*
Baseado em árvores	*Decision Tree*
Ensemble	*Random Forest*
Ensemble	*Bagging Classifier*
Ensemble	*Gradiente Boosting*
Ensemble	*AdaBoost*

Fonte: Elaborada pela autora (2022).

Em todos os métodos supervisionados utilizados foi aplicado o método de *holdout* para separação aleatória dos dados de treino (70%) e teste (30%), conforme Python... (2017)(Figura 16), aplicando validação cruzada na base de treino e avaliando a estimativa do erro de predição associado a aplicação em novas observações, na base de teste.

Figura 16 – Roteiro para aplicação de ML em modelos preditivos

Fonte: (PYTHON..., 2017)

3.1 Modelos

Na literatura e em estudos acadêmicos, diversos algoritmos têm sido desenvolvidos para solucionar problemas diversificados, sendo essencial realizar uma comparação entre eles pelo menos para selecionar aquele que resulta em um modelo com melhor performance preditiva, pois nenhum algoritmo domina todos os outros e em todos os conjuntos de dados disponíveis; ou seja, em um conjunto de dados particular, um algoritmo específico pode funcionar melhor, mas o mesmo pode não ser o que apresenta desempenho mais desejável em outro conjunto de dados (JAMES et al., 2013). Diante dessa situação, após a escolha dos hiper-parâmetros dos algoritmos que resultarão na construção de modelos preditivos candidatos ao modelo final, restou uma questão: qual seria o modelo com melhor desempenho dentro do objetivo do presente estudo?

Alguns modelos possuem performance equivalentes, portanto é possível ponderar os benefícios de cada aplicação, seja pela complexidade computacional, facilidade de implementar a função de predição estimada ou sua interpretabilidade (KUHN; JOHNSON et al., 2013). As próximas seções apresentam as principais características de algoritmos utilizados na etapa de aprendizado de modelos preditivos ajustado para o problema de classificação do presente estudo.

3.1.1 *Logistic Regression*

Dentre os modelos probabilísticos, optamos pela utilização do modelo discriminativo *logit*, que tenta aprender a fronteira entre as classes, ou seja, predizer a probabilidade de cada observação pertencer a uma das classes da resposta de interesse. A variável resposta da base de treino é modelada por meio de uma distribuição binomial, com parâmetro, p, para a probabilidade de ocorrência de uma classe específica dentro de um intervalo [0,1] e com relação direta com os preditores mensurados para cada observação do conjunto de dados (KUHN; JOHNSON et al., 2013).

Tal modelo utiliza a seguinte função logística, responsável pela modelagem da relação entre a probabilidade de determinada resposta, $p(X) = Pr(Y = k \mid X = x)$, e o conjunto de preditores, X:

$$p(X) = Pr(Y = k \mid X = x) = \frac{exp(\beta_0 + \sum_{j=1}^{p} \beta_j X_i)}{1 + exp(\beta_0 + \sum_{j=1}^{p} \beta_j X_i)} \tag{3.1}$$

A partir da aplicação do método da máxima verossimilhança, que estima valores para o vetor de parâmetros, β, o modelo logístico pode ser ajustado. Tais estimativas corresponderão àquelas que resultam em uma probabilidade predita para cada observação da base de treino, $\hat{p}(x_i)$, mais próxima da verdadeira classe a qual a observação pertence (JAMES et al., 2013).

A fórmula matemática da função de verossimilhança formaliza o método, de modo que as estimativas de β serão escolhidas a fim de maximizá-la:

$$L(\beta) = \prod_{i=1}^{n}(p(X)^{y_i}(1-p(X))^{1-y_i}) \tag{3.2}$$

O vetor de parâmetros estimados, $\hat{\beta}$, será utilizado para predizer a resposta de interesse em novas observações (JAMES et al., 2013).

Em problemas de classificação com respostas binárias, 0 e 1, se $p(X)$ é a probabilidade associada à presença de determinada resposta, a medida que descreve a relação entre a resposta de interesse e os preditores mensurados, $(p/1-p)$, desse evento ocorrer será:

$$\frac{Pr(Y=1 \mid X=x)}{Pr(Y=0 \mid X=x)} = \frac{p(X)}{1-p(X)} = exp(\beta_0 + \sum_{j=1}^{p}\beta_j X_i) \tag{3.3}$$

Quando é aplicada a transformação *logit* à essa medida, o modelo torna-se linear em X e a fronteira de decisão linear do modelo de regressão logística ficará relacionada à escolha de um ponto de corte para $p(X)$:

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \sum_{j=1}^{p}\beta_j X_i \tag{3.4}$$

3.1.2 *Decision Tree*

É um método de aprendizagem supervisionado não paramétrico utilizado tanto para classificação quanto para regressão. Através da aprendizagem de regras de decisão simples e uma aproximação constante por partes, ou seja, inferidas a partir das características dos dados disponíveis, o algoritmo busca criar um modelo que preveja o valor de uma variável alvo. Quanto mais profunda for a árvore, mais complexas serão as regras de decisão e mais adequado poderá ser o modelo (JAMES et al., 2013).

A aplicação do algoritmo *Decision Tree*, traduzido para o português como árvore de decisão, pode ser resumida em duas partes: 1) partição do espaço dos preditores, ou seja, do conjunto de valores possíveis para $X_1, X_2, ..., X_p$, em J regiões disjuntas e distintas $R_1, R_2, ..., R_j$; 2) realização de um ajuste em um modelo para predição da resposta de interesse em cada região. O conjunto de regras utilizado para particionar o espaço dos preditores pode ser descrito como uma árvore, por isso restou conhecido como árvore de decisão (FRIEDMAN, 2002; HASTIE et al., 2009).

A construção de uma árvore de classificação é baseada na minimização da SQR onde $\hat{y}_{R_j}$ é a resposta média das observações de treinamento alocadas na j-ésima região:

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2 \tag{3.5}$$

A abordagem divisão binária recursiva é frequentemente utilizada para a construção da árvore, iniciando com todas as observações de treino que pertencem a uma única região, posteriormente, são considerados o conjunto de valores possíveis para $X_1, X_2, ..., X_p$ e os pontos de corte, s, com a finalidade de escolher a combinação preditor-ponto de corte que implica em um menor SQR. Assim, para uma região "j"e um ponto de corte "s"quaisquer, define-se um par de regiões (JAMES et al., 2013):

$$R_1(j, s) = \{X \setminus X_j < s\} \tag{3.6}$$

e

$$R_2(j, s) = \{X \setminus X_j \geq s\} \tag{3.7}$$

Adicionalmente, buscam-se os valores de j e s que minimizem:

$$\sum_{i:x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2 \tag{3.8}$$

Onde $\hat{y}_R1$ e $\hat{y}_R2$ representam a resposta média das observações de treino na $R_1(j, s)$ e na $R_2(j, s)$, respectivamente. As regiões resultantes são divididas em mais duas regiões e esse roteiro prossegue até que um critério de parada seja atingido (HASTIE et al., 2009).

Um vez criadas as regiões $R_1, R_2, ..., R_j$ e estando diante de um problema de aprendizado por classificação, a predição da resposta para uma nova observação será a classe mais frequente entre as observações de treino da região à qual a nova observação pertence, e para atribuir uma nova observação em uma determinada região, serão utilizados os valores mensurados para seus preditores(JAMES et al., 2013).

Uma árvore completa, A_0, com J regiões, pode ser bastante complexa e ter predisposição para o sobreajuste da base de treino e ao aumento do erro quando generalizar para novas observações. Tal problema pode ser solucionado controlando o tamanho de A_0, processo comumente denominado de poda, o que resulta em árvores menos complexas T e com performance preditiva melhor que a de A_0 em novas observações(HASTIE et al., 2009).

Um hiper-parâmetro λ é responsável pelo controle do limiar entre o tamanho da árvore e a qualidade de ajuste do modelo, cuja otimização pode ser feita por meio da validação cruzada *kfold* (KUHN; JOHNSON et al., 2013) e assim ajusta-se o critério da poda.

Cada valor de λ implica em uma nova árvore correspondente, T_λ onde a função perda é mínima:

$$\sum_{n=1}^{T} \sum_{i:x_i \in R_n} (Y_i - \hat{y}_{R_n})^2 + \lambda |T| \tag{3.9}$$

Onde $|T|$ significa o número de nós terminais, R_n é a região relacionada ao n-ésimo nó terminal e $\hat{y}_{R_n}$ é a resposta predita para as observações de R_n. De modo inversamente proporcional, à medida que λ aumenta, menor será a árvore T_λ que minimiza a função perda e, assim, tora mais fácil interpretar o modelo ajustado (FRIEDMAN, 2002; HASTIE et al., 2009).

Três medidas de impureza ou critérios de divisão normalmente são utilizados na aplicação de árvores de decisão binárias (PYTHON..., 2017):

I - Erro de classificação (I_E), fração de observações de treino de uma determinada região que não pertence à classe mais frequente desta região, critério útil para a poda, mas não recomendado para o cultivo de uma árvore de decisão, uma vez que é menos sensível a mudanças nas probabilidades de classe dos nós:

$$I_E(k) = 1 - max\{p(n/k)\} \tag{3.10}$$

Onde $p(n/k)$ é a proporção de observações da n-ésima região que são da k-ésima classe.

II - Índice de Gini (I_G), que minimiza a probabilidade de um erro de classificação e representa uma medida da variância total ao longo das k classes da resposta de interesse, aqui pequenos valores significam que um determinado nó terminal contém observações de uma única classe:

$$I_G = \sum_{k=1}^{K} p(n/k)(1 - p(n/k)) = 1 - \sum_{k=1}^{k} p(n/k)^2 \tag{3.11}$$

III - Entropia (I_H), apresentando valores pequenos se o n-ésimo nó for puro, ou seja, se apresentar frequência elevada de observações pertencentes a mesma classe. A entropia é portanto 0 se todas as amostras de um nó pertencerem à mesma classe, e a entropia é máxima se tivermos uma distribuição de classe uniforme:

$$I_H(k) = -\sum_{k=1}^{k} p(n/k) \log p(n/k) \tag{3.12}$$

A aplicação dessa técnica implica em vantagens e desvantagens na modelagem (Quadro 5).

Quadro 5 – Vantagens e Desvantagens na utilização do algoritmo DT

Vantagens
Simplicidade de compreensão e interpretação, pois as árvores podem ser visualizadas. Requer pouca preparação de dados. O custo da utilização da árvore (ou seja, previsão de dados) é logarítmico no número de pontos de dados utilizados para treinar a árvore. Capaz de tratar tanto os dados numéricos como os categóricos. Capaz de lidar com problemas de multi-saídas. O processo de validação pode ocorrer por meio de testes estatísticos.
Desvantagens
O aprendizado das árvores de decisão podem criar sobreajustamento, ou seja, árvores demasiadamente complexas e que não generalizam bem os dados. Para isto existem mecanismos que podem evitar tais problemas, como por exemplo: a poda, fixação do número mínimo de amostras necessárias num nó de folha ou a fixação da profundidade máxima da árvore. Pequenas variações nos dados podem resultar na geração de uma árvore completamente diferente o que traz instabilidade ao processo de criação das árvores de decisão. As previsões das árvores de decisão não são boas em extrapolação, pois não são suaves nem contínuas, mas sim aproximações constantes por partes. Existem conceitos difíceis de aprender porque as árvores de decisão não os expressam facilmente. O aprendizado das árvores de decisão criam árvores tendenciosas se algumas classes dominarem, portanto, recomenda-se que se equilibre o conjunto de dados antes de se adaptar à árvore de decisão.

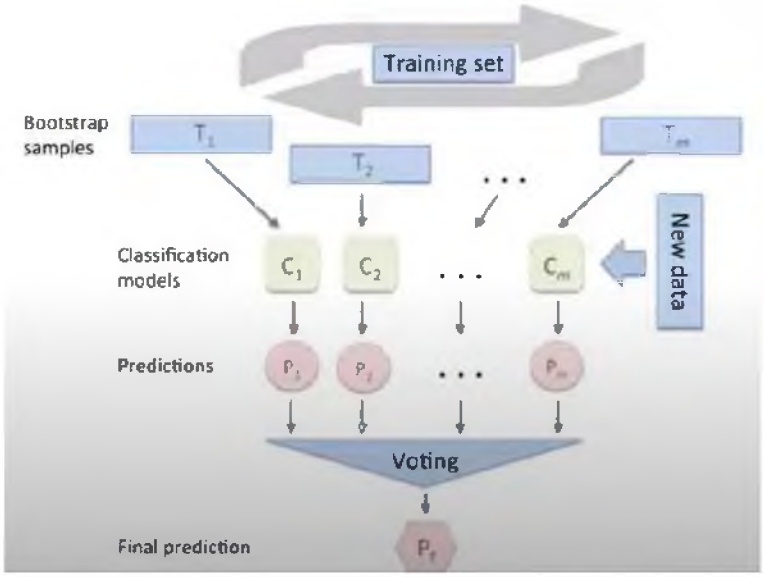
Fonte: Elaborada pela autora (2022).

3.1.3 *Bagging Classifier*

Também conhecido como um método de aprendizado em conjunto baseado em ensacamento, foi proposto por Breiman (1996) e gera um conjunto de dados por amostragem *bootstrap* dos dados originais, ou seja, os classificadores são treinados de forma independente por diferentes conjuntos de treinamento através do método de inicialização, cujo uso é particularmente atraente quando a informação disponível é de tamanho limitado (WANG et al., 2011). O método objetiva evitar o *overfitting*, através da criação de diferentes modelos de ML, utilizando como resultado final a média das respostas encontradas ou a mais votada.

Para construí-los é necessário montar m conjuntos de treinamento idênticos e replicar esses dados de treinamento de forma aleatória para construir m classificadores independentes por *bootstrap*. Em seguida, deve-se agregá-los por meio de um método de combinação apropriada, tal como a maioria de votos (WANG et al., 2011), cuja Figura (17) ilustra a execução do algoritmo. O *bagging* nos permite escolher qualquer algoritmo de ML, já nosso estudo utilizou o parâmetro padrão, *Decision Tree*.

Figura 17 – Algoritmo - *Bagging Classifier*



Fonte: (PYTHON..., 2017)

Para garantir que há amostras de treinamento suficientes em cada subconjunto, grandes porções de amostras (75 - 100%) são colocadas em cada subconjunto, o que acarreta na formação de subconjuntos individuais que se sobrepõem de forma significativa, porém, para assegurar a diversidade de situações são aplicadas pequenas perturbações em diferentes amostras de treinamento (WANG et al., 2011).

A Figura 18 traz um resumo do código do algoritmo proposto por Breiman (1996).

Figura 18 – Pseudo Código *Bagging Classifier*

Entrada/INPUT: *Dataset* $D = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$;
Algoritmo de aprendizado de base L;
Número de rounds de aprendizagem T.

Processo/PROCESS: *For* $t = 1,2, ..., T$;
$D_t = Bootstrap(D)$; #I
$h_t = L(D_t)$ #II
end.

#I - Geração de um conjunto *boostrap* de dados D_t selecionando m exemplos aleatórios do conjunto de treinamento com substituição.
#II - Resultado da base de treinamento do algoritmo baseado em D_t.

Saida/OUTPUT: Classificador $H(x) = maioria\ [(h_1 x), ... (h_T x)]$

Fonte: Elaborado pela autora (2002).

Considere uma base de treino $Z = \{(x_1, y_1), (x_2, y_2), ..., (x_M, y_M)\}$ com $w_i = (X_i, y_i)$, onde x_i representa um vetor de variáveis dependentes (valores para os preditores) e y_i a variável independente, nossa resposta de interesse, contínua ou categórica para a í-ésima observação. Com a aplicação do método *bootstrap*, B conjunto de dados de tamanho m são amostrados com reposição na base de treino, cuja amostra é representada pelo par $w_i = (X_i, y_i)$ (FRIEDMAN, 2002).

Em cada conjunto *bootstrap*, W_b, $b = 1, 2, ..., B$, é aplicada uma árvore de decisão, implicando na geração de B modelos preditivos, sem poda, e por conseguinte, $\hat{y}_b$ predições para uma mesma observação (JAMES et al., 2013). Aqui o viés é baixo e a variância é alta em cada árvore.

Para a análise preditiva por classificação, o algoritmo será baseado no voto majoritário:

$$\hat{P}_{bag}(x) = votomajoritário\{G_b(x)\}_1^B \tag{3.13}$$

Conforme Kuhn, Johnson et al. (2013), Hastie et al. (2009), tal algoritmo apresenta como desvantagem o fato de as B árvores agregadas apresentarem correlação alta devido à utilização de todas as variáveis dependentes/preditoras como candidatas em todas as etapas de divisão das B árvores. Com o propósito de reduzir a correlação entre essas árvores que foram agregadas, pode-se utilizar o método *Random Forest* descrito a seguir.

3.1.4 *Random Forest*

Random Forest é um método de aprendizagem de máquinas que combina árvores de decisão preditoras de tal forma que cada árvore depende dos valores de um vetor aleatório amostral e independente com a mesma distribuição para todas as árvores (BREIMAN, 2001).

Em resumo, este modelo cultiva uma floresta de árvores que permite votar na classe mais popular (se num problema de classificação, objetivo do presente estudo) ou fazer a média das previsões (se num problema de regressão), ou seja, algoritmos criados por várias árvores de decisão, geralmente treinados com o método de *bagging*, porém agora com a realização de um sorteio aleatório das variáveis dependentes/preditoras dentro da base de treino, de modo que a ideia principal é a combinação de modelos e a redução da correlação entre as árvores agregadas, assim melhorando o desempenho final (JAMES et al., 2013; HASTIE et al., 2009; FRIEDMAN, 2002).

Embora os modelos baseados em árvores de decisão convencionais reduzam a possibilidade de *overfitting*, escolhendo parâmetros que controlam a sua profundidade e o número de folhas, o algoritmo *Random Forest* também procura reduzir o *overfitting* de árvores tradicionais por meio da combinação de diferentes árvores.

Pandimurugan et al. (2021) definem *Random Forest* como um grupo de árvores de classificação ou regressão não podadas, baseado no conceito de aprendizagem em conjunto, onde múltiplos classificadores são combinados para resolver um problema complexo e melhorar ainda mais o desempenho do modelo.

Já Golbayani, Florescu e Chatterjee (2020) o descrevem como outro algoritmo

baseado no conjunto de árvores de decisão, porém com uma etapa extra, além de tomar o subconjunto aleatório de dados, também escolhe aleatoriamente um subconjunto de X em cada nó e calcula a melhor divisão nesse nó apenas dentro do subconjunto dado de X. Ele cria uma floresta aleatória com muitas árvores de decisão sem utilizar a totalidade dos dados quando cada árvore é criada, pois o método *bootstrap* seleciona algumas amostras dos dados de maneira aleatória, o que também ocorre na criação dos nós das árvores por meio da seleção de variáveis de forma randômica.

Quanto maior for o número de árvores, maior é a precisão do seu resultado previsto, pois a média de todas estas árvores de decisão é utilizada para aumentar a eficiência e a precisão do poder preditivo do algoritmo. Trata-se de não linearidade explorando a correlação entre as características dos dados/experimento, onde o conjunto de dados permanece o mesmo, porém, cada vez que se treina o modelo, é retirado um subconjunto de todo o conjunto de dados (PANDIMURUGAN et al., 2021).

Tal estrutura fornece previsões não correlacionadas ou fracamente correlacionadas e todas as árvores da floresta são cultivadas em profundidade utilizando apenas dois hiper-parâmetros: o número de variáveis no subconjunto aleatório em cada nó e o número de árvores na floresta (GOLBAYANI; FLORESCU; CHATTERJEE, 2020). Além disso, o algoritmo classifica as variáveis pela importância de uma variável com base na precisão da classificação, enquanto considera a interação entre as variáveis.

Depois de um grande número de árvores ter sido gerado, cada árvore vota para a classe mais popular, cujos procedimentos de votação por árvore são definidos coletivamente como florestas aleatórias. Aqui dois parâmetros têm de ser definidos para a aplicação da técnica: o número de árvores e o número de atributos usado para cultivar cada árvore, ambos gerados aleatoriamente (BROWN; MUES, 2012).

Assim como no caso do algoritmo *Bagging Classifier*, para a análise preditiva por classificação, o algoritmo será baseado também no voto majoritário (JAMES et al., 2013):

$$\hat{P}_{rf}(x) = votomajoritário\{G_b(x)\}_1^B \tag{3.14}$$

A aplicação dessa técnica implica em vantagens e desvantagens na modelagem conforme Quadro 6.

3.1.5 *AdaBoost*

Ao contrário do *Bagging*, no algoritmo *Boosting* os conjuntos de dados re-amostrados são construídos especificamente para gerar aprendizados complementares e a importância do voto é ponderado com base no desempenho de cada modelo, em vez da atribuição de mesmo peso para todos os votos. Essencialmente, esse procedimento permite aumentar

Quadro 6 – Vantagens e Desvantagens na utilização do algoritmo RF

Vantagens
Melhora a precisão ao reduzir a variação dos dados.
Lida com variáveis categóricas e contínuas.
Capaz de manusear grandes conjuntos de dados com alta dimensionalidade.
Pode trabalhar muito eficientemente com parâmetros não lineares.
Melhora a precisão ao reduzir a variação dos dados.
Desvantagens
Complexidade.
Requer alta potência computacional.
Período de formação mais longo.

Fonte: Elaborada pela autora (2022).

o desempenho de um limiar arbitrário simplesmente adicionando *learners*[1] mais fracos, classificadores fracos.

A ideia básica do *Boosting* é aplicar repetidamente um *learner* a versões modificadas do conjunto de dados de formação, produzindo assim uma sequência de *learners* para um número pré-definido de iterações.

Existem várias versões do algoritmo *Boosting*, o *AdaBoost*, proposto por Schapire (1990), "*Adaptive Boosting*", é considerado uma combinação de *Bagging* e *Boosting* e não é uma exigência ter um grande conjunto de treinamento como o *Boosting*. De início, cada exemplo de formação de um determinado conjunto de treinamento tem o mesmo peso (WANG et al., 2011).

Cada iteração de reforço encaixa um *learner* nos dados iniciais ponderados. Aqui o erro é calculado e o peso das instâncias corretamente classificadas é reduzido, enquanto as instâncias classificadas incorretamente receberão pesos mais elevados, de maneira que o modelo final obtido é uma combinação linear de vários *learners* ponderados pelo seu próprio desempenho (WANG et al., 2011).

Em outras palavras, o algoritmo inicia o treino de uma árvore de decisão na qual a cada observação é atribuído um peso igual. Na sequência, a primeira árvore é avaliada e observam-se os pesos das observações que são difíceis de classificar, aumentando-os, já os pesos das que são fáceis de classificar são reduzidos. O próximo modelo (árvore) é elaborado com base nesse resultado, nesses dados ponderados. Aqui, a ideia é melhorar as previsões da primeira árvore (WANG et al., 2011). O novo modelo é, portanto, a árvore 1 + árvore 2.

Calcula-se então o erro de classificação a partir deste novo modelo (contemplando duas árvores) e um terceiro modelo(árvore) é construído para prever os resíduos revisados.

[1] Mistura de especialistas ou sistema de classificadores múltiplo.

Repete-se este processo para um número especificado de iterações. As árvores subsequentes ajudam a classificar as observações que não são bem classificadas pelas árvores anteriores. As previsões do modelo de conjunto final é, portanto, a soma ponderada das previsões feitas pelos modelos de árvores anteriores (WANG et al., 2011).

3.1.6 *Gradient Boosting*

Um dos mais poderosos algoritmos de *boosting*[2], também conhecido como algoritmo de reforço de gradiente (GBM), cria árvores de decisões com o objetivo de prever o valor dos erros da árvore de decisão anterior e utilizá-los na definição de seu resultado final, aqui são incorporados os conceitos de função perda e modelo aditivo (PYTHON. . . , 2017).

Apesar de poder ser aplicado a diversos algoritmos, a árvore de decisão é usualmente escolhida como algoritmo base, pois pode ser considerada um *learn* quando é construída com poucas divisões (KUHN; JOHNSON et al., 2013).

Poderoso algoritmo de aprendizagem supervisionada que combina vários modelos simples com excelente desempenho preditivo e funciona muito bem em várias tarefas de previsão, como por exemplo, na filtragem de spam, publicidade online, detecção de fraudes, detecção de anomalias, física computacional, dentre outros, além lidar com conjuntos de dados heterogêneos (dados altamente correlacionados, dados em falta, dados categóricos, etc.), conduzindo as análise para modelos interpretáveis através da construção de um modelo aditivo (LU; MAZUMDER, 2020).

Observe que o modelo inicial possui viés alto, porém variância baixa e, a cada passo, o valor predito é atualizado de modo a diminuir o viés e aumentar a variância do modelo atualizado (IZBICKI; SANTOS, 2018).

Conforme Breiman (2001), *Gradient Boosting* melhora a precisão de uma função preditiva através da minimização incremental do termo de erro. Após a base inicial, o *learn* (mais comumente uma árvore de decisão) é cultivado, cada árvore da série é adequada aos chamados "pseudo-resíduos"da predição de árvores anteriores, com o objetivo de reduzir o erro. Isto conduz para o seguinte modelo:

$$F(X) = G_0 + \beta_1 T_1(X) + \beta_2 T_2(X) + ... + \beta_n T_n(X), \tag{3.15}$$

onde G_0 é igual ao primeiro valor da série, $T_1, ..., T_n$ são as árvores ajustadas aos pseudo-residuais e β_i são coeficientes para os respectivos nós de cada árvore computados pelo algoritmo de aumento de gradiente. Uma explicação mais detalhada do reforço de gradiente pode ser encontrada em (FRIEDMAN, 2002).

[2] Técnica de aprendizado em sequência.

O classificador de gradiente exige afinação de dois hiper-parâmetros: a profundidade da árvore, relacionada ao número de divisões em cada árvore, e o número de iterações, m, relacionados ao número de árvores do modelo *boosting* final, os quais podem ser otimizados por validação cruzada *kfolds* (JAMES et al., 2013; KUHN; JOHNSON et al., 2013).

Em resumo, o algoritmo treina muitos modelos de uma maneira gradual, aditiva e sequencial e a maior diferença entre este e o *Adaboost* é como os dois algoritmos identificam as deficiências dos *learns*. Enquanto o modelo *AdaBoost* identifica as deficiências usando pontos de dados de peso elevado, o *Gradient Boosting* executa a mesma tarefa, mas usando gradientes em uma função de perda ($y = a \times +b + e$, onde e precisa de um destaque especial, pois é o termo de erro).

Tal função é uma medida que indica quão bons são os coeficientes do modelo na adequação dos dados subjacentes. Para a aplicação do presente estudo, a função de perda seria uma medida de quão bom é nosso modelo de previsão na classificação de contratos ruins.

O algoritmo *Gradient Boosting* envolve três elementos: I - uma função de perda a ser otimizada que depende do tipo de problema que está sendo resolvido; II - um *learn* para fazer previsões, comumente árvores de decisão, escolhendo os melhores pontos de divisão com base em pontuações de pureza como Gini ou para minimizar a perda, restringindo o *learn* de maneiras específicas, tais como um número máximo de camadas, nós, fendas ou nós foliares, e III - um modelo aditivo que adiciona *learns* para minimizar a função de perda. Neste último caso, as árvores são adicionadas uma de cada vez e as árvores existentes no modelo não são alteradas, alocando um procedimento de descida por gradiente para minimizar a perda ao adicionar árvores (WANG et al., 2011; HASTIE et al., 2009; FRIEDMAN, 2002).

O algoritmo *Gradient Boosting* pode se sobrepor rapidamente a um conjunto de dados de treinamento, ou seja, ele pode se beneficiar de métodos de regularização que penalizam várias partes do algoritmo e geralmente melhoram o desempenho do algoritmo ao reduzir o superajuste. Algumas melhorias podem adicionar valor à aplicação do algoritmo: restrições do número de árvores, encolhimento, amostragem aleatória e aprendizado penalizado (HASTIE et al., 2009).

As previsões de cada árvore são somadas sequencialmente e a contribuição de cada árvore para esta soma pode ser ponderada para retardar o aprendizado pelo algoritmo, chamada de contração ou taxa de aprendizagem (*learning rate*) (PYTHON. . . , 2017).

O resultado é que o aprendizado é desacelerado e para isso requer que mais árvores sejam adicionadas ao modelo, que por sua vez demora mais para treinar, proporcionando um *trade-off* de configuração entre o número de árvores (número de estimadores) e a taxa de aprendizado. É comum aplicar *learning rates* de pequenos valores na faixa de 0,1 a 0,3,

assim como valores inferiores a 0,1 (PYTHON..., 2017).

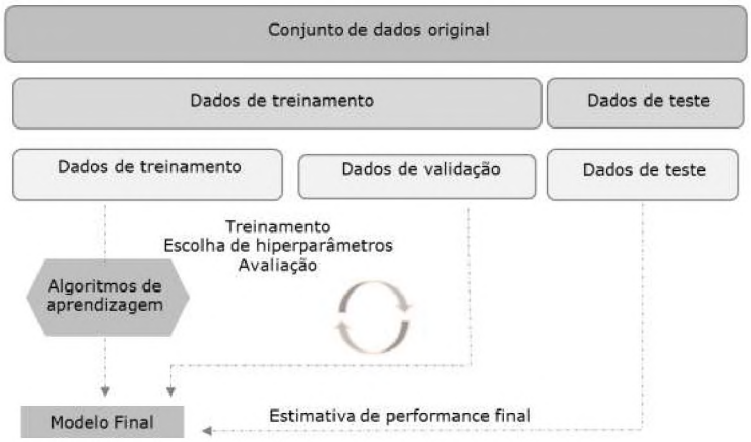
Apesar de aumentar a precisão de um *learn*, como uma árvore de decisão ou regressão linear, o *Gradient Boosting* sacrifica a inteligibilidade e a interpretabilidade. Observar o caminho da tomada de decisão de uma árvore de decisão parece ser trivial e auto-explicativo, mas seguir os caminhos de centenas ou milhares de árvores torna a avaliação de seu desempenho quanto a interpretabilidade muito difícil além de exigir uma maior demanda computacional (WANG et al., 2011).

3.2 Validação Cruzada

Em geral, o aprendizado de modelos preditivos contempla dois objetivos principais: selecionar e avaliar modelos. No primeiro caso, estima-se a performance dos modelos a fim de escolher qual possui o melhor desempenho, já o segundo caso, após o modelo ser escolhido, pretende-se estimar o seu erro de predição, ou seja, o quanto ele pode errar quando generalizar em novas observações (HASTIE et al., 2009).

A melhor abordagem para a seleção de modelos, quando o estudo dispõe de um conjunto grande de dados originais, consiste em dividí-la em três partes: treinamento, validação e teste. Sendo os dados de treino usados para ajustar os modelos, os de validação para selecionar um modelo com base em sua performance preditiva e os de teste para avaliar o erro de generalização do modelo selecionado (HASTIE et al., 2009). A Figura 19 descreve tal divisão.

Figura 19 – Divisão do conjunto de dados original em treino, validação e teste para aplicações de ML



Fonte: (PYTHON..., 2017).

Nosso *dataset* não permitiu realizar esse tipo de divisão nos dados devido a problemática do desbalanceamento das classes que mesmo com seleção aleatória de subamostras tornou-se inócuo pela quantidade de observações disponíveis, ou seja, o conjunto de dados não é suficiente para executar essas três divisões, e técnicas de re-amostragem

mostram-se efetivas para aproximar o conjunto de validação por meio da reutilização de observações/registros da base de treino (HASTIE et al., 2009).

Dentre as técnicas de re-amostragem mais utilizadas em problemas de ML, usamos a validação cruzada, termo em inglês *cross validation*, avalia a capacidade de generalização de um modelo por meio da divisão aleatória da base de treino em k partes de tamanhos aproximadamente iguais, em que $k-1$ irão representar dados de treino para ajuste do modelo preditivo e a outra parte ficará reservada para a estimativa de sua performance. Tal processo se repetiu até que todas as partes tenham participado tanto do treino como da validação do modelo, resultando em k estimativas de performance que serão resumidas, geralmente, pelo cálculo da média e do erro padrão (KUHN; JOHNSON et al., 2013).

Tal técnica auxiliou a não cometer o erro de usar os dados de teste para avaliar e tentar melhorar o modelo avaliado, além de estabelecer um intervalo de quão bom o modelo pode ser (HASTIE et al., 2009; JAMES et al., 2013).

Em ML, os algoritmos podem apresentar um ou mais parâmetros para controlar o equilíbrio entre o viés e a variância, ou seja, a complexidade do modelo ajustado, cujos parâmetros de sintonização ou hiper-parâmetros podem ser escolhidos e especificados antes do ajuste do modelo preditivo. Não existe uma fórmula disponível para o cálculo do seu valor apropriado, seja pela estimação na base de treino ou na otimização por validação cruzada (KUHN; JOHNSON et al., 2013). Exemplos de hiper-parâmetros, o número de m árvores de decisão presente no algoritmo *Boosting*, a profundidade das árvores no algoritmo Random Forest e o tipo de penalidade presente na *Logist Regression* foram observados em nossa aplicação para o presente estudo.

O presente estudo aplicou o processo de validação cruzada chamada *leave-one-out* para estimar os melhores parâmetros de cada modelo testado, antes de avaliar o desempenho, e para isso utilizou a ferramenta *GridSearchCV* da biblioteca "*sklearn.model_selection*", conforme roteiro descrito na Figura 20.

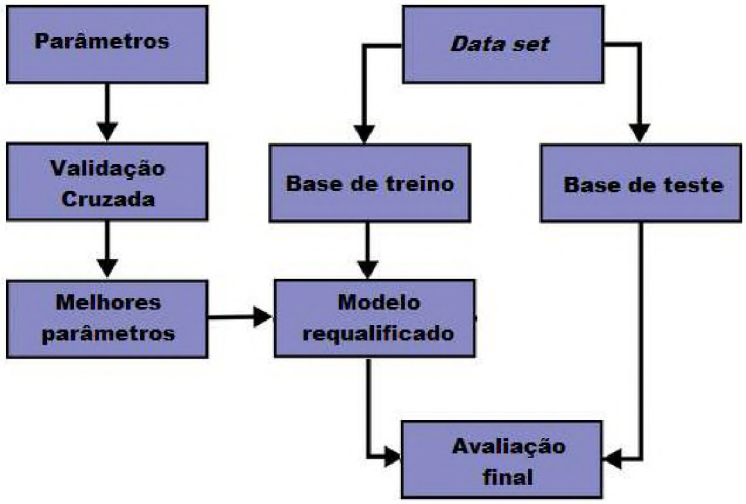
Tal ferramenta implementou um método de "ajuste"e de "pontuação", ou seja, por meio de uma maneira sistemática realizou diversas combinações de parâmetros e, depois de avaliá-los, os armazenou num único objeto.

Dentre as métricas utilizadas, para o objeto "param_ gride"foi definido o parâmetro cv [3] $= 5$ e $cv = 10$, com $shuffle = True$, ou seja, misturando as informações para eliminar qualquer influência da disposição dos dados dentro da função *StratifiedKFolds*, que em bases desbalanceadas garante que em todos as partes/*folds* a proporção das classes sejam a mesma, auxiliando um melhor treinamento e teste em cada simulação.

Os parâmetros de cada modelo foram listados e os valores desta "lista" foram arbitrariamente escolhidos e combinados com o objetivo de encontrar valores otimizados.

[3] Determina a estratégia de divisão de partes/*folds* executado pela validação cruzada.

Figura 20 – Processo de validação cruzada *leave-one-out*



Fonte: Elaborado pela autora (2002).

Os parâmetros utilizados na técnica de validação cruzada "*GridSearchCV*"tiveram como objetivo usar métricas para a escolha de hiper-parâmetros de modo que os modelos não os usassem de modo super ou subestimados, evitando assim o problema de *overfitting*.

Como os hiper-parâmetros apresentam relação com a flexibilidade (complexidade) de cada modelo preditivo, escolher inadequadamente seus valores podem resultar em *overfitting*, sobreajuste, performance ruim do modelo quando aplicado em novas observações/registros. Geralmente, escolhe-se uma métrica, por exemplo, AUC em problemas de classificação, e para cada algoritmo tal métrica é avaliada em uma lista de candidatos aos hiper-parâmetros por validação cruzada, cujo objetivo é selecionar aquele que resulte em um modelo que minimiza o erro de predição (KUHN; JOHNSON et al., 2013).

A validação cruzada aplicada no presente estudo contemplou a avaliação não só de uma métrica, mas de várias (precisão, sensibilidade, acurácia balanceada e *f-beta score*[4], esta com beta = 1). O parâmetro beta determina o peso da sensibilidade na pontuação combinada, onde beta < 1 implica em mais peso para a precisão, enquanto beta > 1 implica mais peso para a sensibilidade. Dentro do processo de validação cruzada, optamos pelo parâmetro *refit* igual ao "F1"(*f-score ou f-mensure*) dentro da função *GridSearchCV*, ou seja, ele refaz um estimador usando o melhor parâmetro encontrado em todo o conjunto de dados.

A aplicação de alguns hiper-parâmetros implicarão em modelos mais simples e outros podem resultar em modelos mais complexos. Em geral, modelos simples apresentam baixa variância e viés alto, ou seja, sua função estimada não apresentará modificações substanciais entre diferentes bases de treino, aqui existem poucos parâmetros estimados

[4] média harmônica ponderada entre a precisão e a sensibilidade, atingindo o seu valor ótimo em 1 e o seu pior valor em 0.

→ variância baixa, porém pode não ser uma boa aproximação para o padrão presente nos dados (alto viés) Nosso processo de seleção de hiper-parâmetros contempla um equilíbrio entre o viés e a variância.

Já modelos mais complexos apresentam variância alta e viés baixo, pois sua função estimada pode conter vários parâmetros, assim pequenas modificações na base de treino podem implicar funções estimadas bastante diferentes (alta variância). Para uma base de treino em particular, tal função se aproxima muito bem do padrão presente nos dados e, assim, apresenta viés reduzido. Tais características estão presentes em modelos que apresentam baixa capacidade de generalização (*overfitting*, ou seja, que se ajustam bem aos dados de treino, mas apesentam desempenho ruim quando aplicados a novas observações/registros (IZBICKI; SANTOS, 2018). Foi possível observar esses conceitos em cada modelo avaliado, o que contribuiu para um melhor desempenho de alguns modelos.

Com relação ao equilíbrio entre viés e variância, existem modelos menos complexos que representarão a melhor alternativa, pois estão menos sujeitos ao sobreajuste/*overfitting*, resultando em predições mais acuradas para Y quando inseridas novas observações. Portanto, não existe um único algoritmo capaz de apresentar boa performance em todas as aplicações, sendo estritamente necessário realizar comparação entre eles e escolher o mais adequado ao problema que está sendo estudado (JAMES et al., 2013), portanto, o presente estudo escolheu um rol de parâmetros para serem testados e, posteriormente, selecionados em cada modelo.

Para aplicação do processo de validação cruzada realizada, o Anexo D detalha os hiper-parâmetros testados e já aqueles efetivamente utilizados em cada algoritmo estão descritos no código disponível no repositório do GitHub:

(*https* : *//github.com/sabrinaalencar*2611/*Inadimplencia_de_contratos.git*).

3.3 Desempenho dos Modelos

Para se ter uma avaliação justa das metodologias desenvolvidas que realizarão a previsão de possíveis atrasos nos contratos, precisamos ter uma base de avaliação igual para os métodos utilizados, além de utilizar o mesmo conjunto de dados com a mesma estrutura de formação, validação e testes e, em segundo lugar, precisamos ter uma medida justa para avaliação de desempenho (GOLBAYANI; FLORESCU; CHATTERJEE, 2020).

A avaliação da performance dos algoritmos de ML em nosso conjunto de dados é realizada por meio da mensuração do quão bem as predições decorrentes do modelo ajustado reproduzem o valor observado para a resposta de interesse (o atraso). Logo, quantificamos o quanto o valor predito ($\hat{Y}_i$) para a resposta de uma observação/registro se aproxima de um valor observado, Y_i (JAMES et al., 2013).

Após a aplicação dos algoritmos à base de treino para a seleção de um modelo que se ajustasse satisfatoriamente aos dados, a performance deles foi avaliada em dados novos (base de teste), que não participaram do ajuste do modelo (PYTHON..., 2017).

Para os modelos de classificação, usualmente, a avaliação da performance resultam em dois tipos de predição: uma contínua ($\dot{p_k}$), correspondente à probabilidade de cada uma das classes, k, $k = 1, 2, ..., K$, da resposta de interesse, e outra categórica (podendo ser, classe 0: apresentou determinado comportamento e classe 1: não apresentou), referente à classe predita para uma dada observação/registro (KUHN; JOHNSON et al., 2013).

As predições contínuas possibilitam a utilização do classificador (modelo ajustado) em vários cenários, considerando pontos de corte conforme o interesse do estudo. Com a aplicação dela, uma nova observação é atribuída à classe onde ($\dot{p_k}$) é máxima. Exemplificando no caso de respostas dicotômicas, a nova observação/registro é atribuída à classe 1 se $(\dot{p_k}) > 0,5$, caso este seja o ponto de corte escolhido no estudo. Tal ponto de corte pode ser alterado conforme objetivo do problema de predição (JAMES et al., 2013). Na prática, esses classificadores, em respostas dicotômicas, podem implicar em dois erros: classificar indevidamente o indivíduo que apresentou determinado comportamento à classe correspondente ao indivíduo que não apresentou e vice versa.

Em cada modelo ajustado, uma matriz de confusão é gerada indicando os erros e acertos em cada execução, comparando a situação real e a detectada, ou seja, descrevendo o desempenho de um modelo de classificação no conjunto de dados de teste, cujos valores verdadeiros são conhecidos (a Figura 21 demonstrada a nomenclatura utilizada no presente estudo). A partir da definição do ponto de corte ($\dot{p_k}$), a matriz de confusão foi comumente utilizada para visualizar esses erros. As caselas da diagonal principal (VP e VN) denotam casos em que as classes são corretamente preditas, enquanto que as caselas fora da diagonal (FP e FN) representam os erros de classificação (KUHN; JOHNSON et al., 2013).

Figura 21 – Matriz de Confusão

		Detectada	
		Sim	**Não**
Real	**Sim**	Verdadeiro Positivo (VP)	Falso Negativo (FN)
	Não	Falso Positivo (FP)	Verdadeiro Negativo (VN)

Fonte: Elaborado pela Autora (2022)

Em termos de medidas de validação, aplicamos acurácia, precisão/valor preditivo positivo, recall/sensibilidade/revocação, especificidade, valor preditivo positivo, valor preditivo negativo, coeficiente gini, f1-score, curva ROC e AUC/eficiência, que foram amplamente utilizados em estudos semelhantes (LESSMANN et al., 2015; VIEIRA et al., 2019; SANTOS et al., 2002; LIU; FAN; XIA, 2022; DUMITRESCU et al., 2022).

A acurácia ou taxa de erro geral é definida como a proporção de predições corretas, sem levar em consideração o que é positivo e o que é negativo, representa a métrica mais simples derivada da matriz de confusão. Esta medida é altamente suscetível a desbalanceamentos do conjunto de dados e pode facilmente induzir a uma conclusão errada sobre o desempenho do modelo, pois não faz distinção entre o tipo de erro cometido (classificação indevida) e não considera a frequência natural (prevalência) de cada classe (JAMES et al., 2013; KUHN; JOHNSON et al., 2013):

$$Acurácia = \frac{VP + VN}{VP + VN + FP + FN} \tag{3.16}$$

A *recall*/sensibilidade/revocação é definida como a proporção de verdadeiros positivos, ou seja, a capacidade do modelo em predizer corretamente a condição para casos que realmente se deseja prever, ou seja, onde a resposta de interesse foi, de fato, observada (JAMES et al., 2013; KUHN; JOHNSON et al., 2013):

$$Sensibilidade = \frac{VP}{VP + FN} \tag{3.17}$$

A especificidade é definida como a proporção de verdadeiros negativos, ou seja, a capacidade do modelo em predizer corretamente a ausência da condição para casos que realmente não a tem, ou seja, entre aqueles onde a resposta de interesse (observada) está ausente (JAMES et al., 2013; KUHN; JOHNSON et al., 2013):

$$Especificidade = \frac{VN}{VN + FP} \tag{3.18}$$

Por meio da sensibilidade e da especificidade, é possível determinar qual o erro derivado de um modelo preditivo (classificador), ou seja, conhecendo o desempenho específico, de acordo com as classes da resposta de interesse, no nosso caso o atraso (PYTHON. . . , 2017).

A precisão é definida como a proporção de verdadeiros positivos em relação a todas as predições positivas. Tal medida é altamente suscetível a desbalanceamentos do conjunto de dados e pode facilmente induzir a uma conclusão errada sobre o desempenho do modelo (HASTIE et al., 2009):

$$Precisão = \frac{VP}{VP + FP} \tag{3.19}$$

O valor preditivo negativo (VPN) é definido como a proporção de verdadeiros negativos em relação a todas as predições negativas, aqui a problemática da presença de desbalanceamento entre classes pode também prejudicar a análise sobre o desempenho do modelo(JAMES et al., 2013):

$$VPN = \frac{FP}{VN + FP} \tag{3.20}$$

O MCC ou coeficiente phi retorna um valor entre (-1) e $(+1)$, em que um coeficiente de $(+1)$ representa uma predição perfeita, (0) representa uma predição aleatória média, e (-1) uma predição inversa(KUHN; JOHNSON et al., 2013):

$$MCC = \frac{(VP \times VN) - (FP \times FN)}{\sqrt{(VP + FP) \times (VP + FN) \times (VN + FP) \times (TN + FN)}} \tag{3.21}$$

O F-*measure*, F-*score* ou *score* F1 é definido como a média harmônica entre a precisão e o *recall*/sensibilidade. Ela é muito boa quando você possui um *dataset* com classes desproporcionais, e o seu modelo não emite probabilidades. Isso não significa que não possa ser usada com modelos que emitem probabilidades, tudo depende do objetivo de sua tarefa de ML. Em geral, quanto maior, melhor (KUHN; JOHNSON et al., 2013; PYTHON. . . , 2017) e tal métrica foi utilizada em nosso processo de seleção dos hiper-parâmetros dos modelos:

$$F1 - score = 2(\frac{Precisão \times Sensibilidade}{Precisão + Sensibilidade}) \tag{3.22}$$

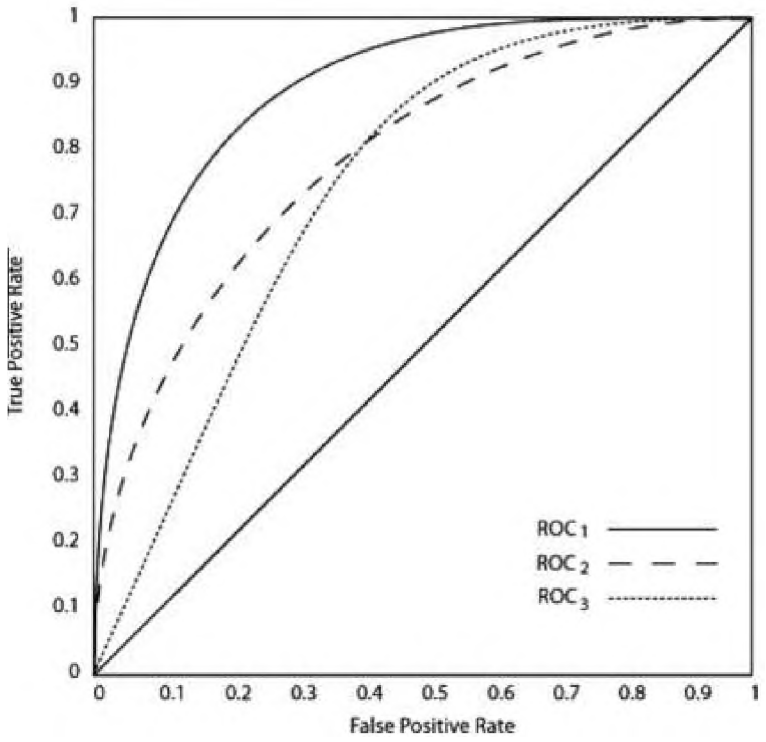
A curva ROC é considerada uma ferramenta comum para avaliar modelos de classificação, pois ilustra graficamente o *trade-off* entre a taxa de verdadeiros positivos (sensibilidade), ou seja, taxa de casos em que a categoria alvo (1) foi classificada corretamente, e a taxa falsos positivos (1-especificidade), taxa de casos em que a categoria 0 foi incorretamente classificada como sendo da categoria alvo (1) (BROWN; MUES, 2012).

Um balanço entre sensibilidade e especificidade pode ser apropriado quando há diferentes penalidades associadas a cada tipo de erro. Aqui, a curva ROC representa uma importante ferramenta para avaliar a sensibilidade e a especificidade decorrentes de todos os pontos de corte possíveis p_k (Figura 22) . Um ponto de corte igual a $0,5$ minimiza a taxa de erro geral, entretanto, cada estudo pode ter interesse em predizer corretamente mais uma classe ou outra e assim ajustar tal ponto de corte (JAMES et al., 2013; KUHN; JOHNSON et al., 2013).

Para um classificador com bom desempenho, a curva ROC tem de estar tão longe quanto o canto superior esquerdo, tanto quanto possível, tal métrica foi observada em cada comparação e o resultado pode ser observado no Capítulo Resultados. Conforme Figura 22, o classificador com melhor desempenho está representado pela curva ROC1.

Com isso, se a curva ROC de um modelo se aproxima bastante do ponto na parte alta e esquerda do gráfico, esse modelo pode ser considerado bom. Outra forma de visualizar esse desempenho seria quanto mais longe a curva ROC estiver da linha diagonal, que é a

Figura 22 – Curva ROC



Fonte: (BROWN; MUES, 2012).

curva ROC de um modelo aleatório (BROWN; MUES, 2012), melhor consegue-se calcular a AUC e ressaltar o ótimo desempenho do modelo avaliado.

Todos os algoritmos foram avaliados também em termos da sua AUC (área abaixo da curva das características operacionais do receptor), cuja medida tem o poder de discriminação de um classificador em todos os pontos de corte possíveis para p_k sem considerar a distribuição de classes ou o custo de classificação errada (BAESENS et al., 2003). O presente estudo contempla uma classificação binária e a AUC foi calculada da seguinte forma, seguindo Huang e Ling (2005):

$$AUC = \frac{S_0 - n_0(n_0 + 1)/2}{n_0 n_1}, \tag{3.23}$$

Onde n_0 e n_1 denotam o número de empréstimos bons e ruins, respectivamente, no conjunto de testes e $S_0 = \sigma Classificação_j$ é o somatório da classificação das predições de probabilidade dos empréstimos $j - ésimo$ ruins.

A AUC avalia a curva ROC dos modelos e calcula a área da forma bidimensional formada abaixo dessa curva, indicando a probabilidade de duas previsões serem corretamente ranqueadas, logo a AUC será um valor entre 0 e 1 e quanto maior esse valor, melhor a capacidade do modelo em separar as classes. Ela também pode ser útil na comparação de dois ou mais modelos com diferentes preditores, hiper-parâmetros ou mesmo classificadores decorrentes de algoritmos completamente diferentes (JAMES et al.,

2013; KUHN; JOHNSON et al., 2013).

Portanto, a AUC pode ser entendida como a eficiência do modelo, que é definida como a média aritmética da sensibilidade e especificidade, pois na prática, estas variam em direções opostas, ou seja, geralmente, quando um método é muito sensível a positivos, tende a gerar muitos falso-positivos, e vice-versa. Assim, um modelo perfeito (100% de sensibilidade e 100% especificidade) raramente é alcançado, e um balanço entre ambos deve ser atingido (JAMES et al., 2013).

Para o presente estudo, avaliou-se o desempenho do modelo que melhor conseguiu predizer se o um contrato de empréstimo comercial vai atrasar com prazo maior ou igual a 30, 60 ou 90 dias dentro do prazo de observação dos contratos. Portanto, é essencial que o modelo acerte quando o contrato atrasar e com isso tente obter uma taxa de falso negativo próximo a zero.

Como geramos aleatoriamente subamostras balanceadas dentro de um universo que contempla classes desbalanceadas em cada subgrupo, se o modelo prever todos os contratos como bons por meio de uma acurácia de 90%, isso pode induzir o leitor a conclusões equivocadas, portanto o conjunto de métricas utilizadas para avaliar o desempenho dos modelos é avaliada de forma conjunta, conforme quadros de desempenho apresentados no capítulo Resultados.

Apesar de todas as métricas citadas, para o estudo proposto, observa-se que a sensibilidade traz um interessante ponto de atenção ao comparar os algoritmos, pois valores altos de sensibilidade indicam altos valores de verdadeiros positivos mesmo quando se leva em conta os falsos negativos, em outras palavras, o objetivo é identificar todos os contratos ruins quando realizado o monitoramento dos contratos de crédito, pois a possibilidade de perda financeira classificando indevidamente um contrato ruim como bom pode ser muito maior que o contrário, ou seja, uma situação em que os falsos negativos são considerados mais prejudiciais que os falsos positivos. Em resumo, o melhor modelo deve de qualquer maneira encontrar todos os contratos ruins, mesmo que classifique alguns contratos bons como ruins (situação de falso positivo) no processo, ou seja, o modelo deve ter alta sensibilidade, pois classificar contratos ruins como bons pode ser prejudicial para a execução de um efetivo monitoramento de contratos que impede uma ação proativa da instituição financeira frente a uma provável perda financeira futura.

Os contratos de empréstimos para grandes empresas são inevitavelmente concedidos em montantes elevados e quando estes entram em estado de inadimplência pelo não pagamento das prestações podem implicar em perda financeiras para as instituições financeiras decorrente de aumento no provisionamento para crédito de liquidação duvidosa, na medida do atraso conforme legislação vigente (BRASIL, 1999).

Considerando as amostras utilizadas em nosso estudo, também destacamos um

olhar preponderante para as métricas de AUC e *score* F1 que têm boa aplicação nessas situações (JAMES et al., 2013).

Outra métrica que foi avaliada em conjunto, com peso considerável na comparação dos modelos, foi a precisão, pois a quantidade que é classificada corretamente interessa ao processo de monitoramento efetivo de contratos, a medida que seu resultado é maior que a inadimplência dos subgrupos avaliados.

4 Resultados

Ao final das simulações realizadas, identificamos desempenho distintos entre os algoritmos em cada subgrupo avaliado, porém destacamos que ao aplicarmos as técnicas de seleção de variáveis, todos os modelos tiveram sua performance melhorada quando foram excluídas *features*. Adicionalmente, ressaltamos que a aplicação da técnica de validação cruzada para seleção dos hiper-parâmetros, também permitiu um incremento no desempenho dos modelos.

O resultado final será apresentado em dois tópicos: considerando todas as variáveis previamente selecionadas, e, posteriormente, excluindo as variáveis "*Rating* do Contrato"que mostrou-se não apresentar poder explicativo junto a variável independente (*target* ou variável resposta) na data de concessão dos contratos, "Valor do Contrato"e "Quantidade de Renegociações", conforme resultado das técnicas de seleção de variáveis em cada subgrupo.

4.1 Aplicando todas as variáveis disponíveis

Considerando o subgrupo *Atraso* ≥ 30, quando observamos o nível de inadimplência do conjunto de dados original de $8,59\%$, observamos que, ao equilibrarmos a sensibilidade e a precisão, o algoritmo *Random Forest* mostrou-se com melhor performance frente aos demais algoritmos em praticamente em todas as simulações conforme Quadros 7, 8, 9, 10 e 11 com destaque para as métricas F1 e AUC acima de $60,00\%$ e precisão acima de $70,00\%$, acima de inadimplência da carteira (Subgrupo).

Quadro 7 – Avaliação de desempenho - Subgrupo Atraso ≥ 30 - TV

Subamostra 01						
Métrica	LR	DT	RF	BC	GB	AD
Acurácia	52,00%	66,67%	**69,33%**	49,33%	64,00%	50,67%
Precisão	52,38%	76,00%	**75,86%**	0,00%	67,74%	50,67%
VPN	51,52%	62,00%	**65,22%**	48,65%	61,36%	0,00%
Sensibilidade	57,89%	50,00%	**57,89%**	0,00%	55,26%	100,00%
Especificidade	45,95%	83,78%	**81,08%**	97,30%	72,97%	0,00%
Eficiência	51,92%	66,89%	**69,48%**	48,65%	64,11%	50,00%
Coef. phi	0,04	0,36	**0,40**	-0,12	0,29	0,00
F1 Score	55,00	60,32	**65,67**	0,00	60,87	67,26
AUC	0,519	0,669	**0,695**	0,500	0,641	0,500

Fonte: Elaborada pela autora (2022)

Quadro 8 – Avaliação de desempenho - Subgrupo Atraso ≥ 30 - TV

Subamostra 02						
Métrica	LR	DT	RF	BC	GB	AD
Acurácia	50,67%	48,00%	**68,00%**	50,67%	44,00%	58,67%
Precisão	50,67%	48,89%	**71,88%**	50,67%	46,67%	59,46%
VPN	0,00%	46,67%	**65,12%**	0,00%	33,33%	57,89%
Sensibilidade	100,00%	57,89%	**60,53%**	100,00%	73,68%	57,89%
Especificidade	0,00%	37,84%	**75,68%**	0,00%	13,51%	59,46%
Eficiência	50,00%	47,87%	**68,11%**	50,00%	43,60%	58,67%
Coef. phi	0,00	-0,04	0,37	0,00	-0,16	0,17
F1 Score	67,26	53,01	**65,72**	67,26	57,14	58,66
AUC	0,500	0,479	**0,681**	0,500	0,436	0,587

Fonte: Elaborada pela autora (2022)

Quadro 9 – Avaliação de desempenho - Subgrupo Atraso ≥ 30 - TV

Subamostra 03						
Métrica	LR	DT	RF	BC	GB	AD
Acurácia	50,67%	60,00%	**66,67%**	45,33%	62,67%	58,67%
Precisão	50,67%	59,52%	**68,57%**	44,44%	64,71%	60,61%
VPN	0,00%	60,61%	**65,00%**	45,83%	60,98%	57,14%
Sensibilidade	100,00%	65,79%	**63,16%**	31,58%	57,89%	62,63%
Especificidade	0,00%	54,05%	**70,27%**	59,46%	67,57%	64,86%
Eficiência	50,00%	59,92%	**66,72%**	45,52%	62,73%	58,75%
Coef. phi	0,00	0,2	**0,33**	-0,09	0,26	0,18
F1 Score	67,26	62,50	**65,75**	36,92	61,11	56,34
AUC	0,500	0,599	**0,667**	0,455	0,627	0,587

Fonte: Elaborada pela autora (2022)

A matriz de confusão resultante da comparação entre os algoritmos dentro da Subamostra 02 pode ser consultada na Figura 23. As demais matrizes estão consignadas no Anexo E.

Considerando o subgrupo *Atraso* ≥ 60, quando observamos o nível de inadimplência do conjunto de dados utilizados, 2, 46%, observamos que, ao equilibrarmos a sensibilidade e a precisão, os algoritmos do tipo *Ensemble* performaram melhor que os tradicionais (*Logist Regression*) em grande parte das Subamostras avaliadas, conforme Quadros 12, 13, 14, 15 e 16, porém destacamos que os desempenhos aqui foram inferiores àqueles observados no subgrupo *Atraso* ≥ 30 apresentado anteriormente.

A matriz de confusão resultante da comparação entre os algoritmos dentro da Subamostra 01 pode ser consultada na Figura 24. As demais matrizes estão consignadas no Anexo E.

Quadro 10 – Avaliação de desempenho - Subgrupo Atraso ≥ 30 - TV

Subamostra 04						
Métrica	LR	DT	RF	BC	GB	AD
Acurácia	54,67%	66,67%	**61,63%**	49,33%	61,33%	53,33%
Precisão	53,45%	84,21%	**62,86%**	0,00%	60,98%	53,49%
VPN	58,82%	60,71%	**60,00%**	48,65%	61,76%	53,12%
Sensibilidade	81,58%	42,11%	**57,89%**	0,00%	65,79%	60,53%
Especificidade	27,03%	91,89%	**64,86%**	97,30%	56,76%	45,95%
Eficiência	54,30%	67,00%	**61,38%**	48,65%	61,28%	53,24%
Coef. phi	0,1	0,39	**0,23**	-0,12	0,23	0,07
F1 Score	64,58	56,14	**60,27**	0,00	63,29	56,79
AUC	0,543	0,670	**0,614**	0,500	0,613	0,532

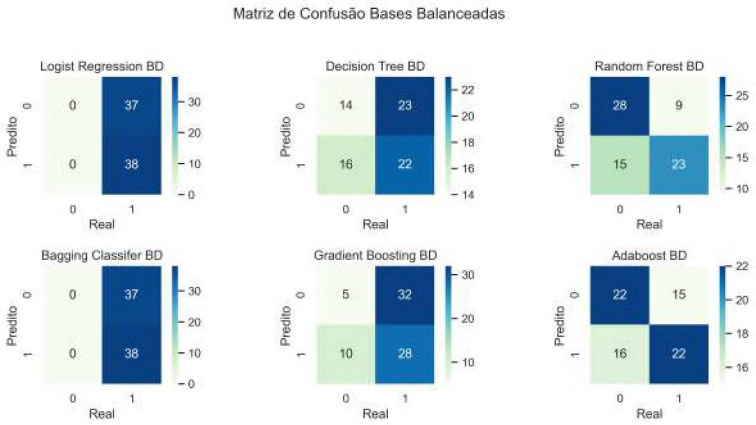
Fonte: Elaborada pela autora (2022)

Quadro 11 – Avaliação de desempenho - Subgrupo Atraso ≥ 30 - TV

Subamostra 05						
Métrica	LR	DT	RF	BC	GB	AD
Acurácia	56,00%	60,00%	52,00%	58,67%	48,00%	61,33%
Precisão	56,41%	58,33%	52,78%	62,07%	48,78%	62,16%
VPN	55,56%	62,96%	51,28%	56,52%	47,06%	60,53%
Sensibilidade	57,89%	73,68%	50,00%	47,37%	52,63%	60,53%
Especificidade	54,05%	45,95%	54,05%	70,27%	43,24%	62,16%
Eficiência	55,97%	59,82%	52,02%	58,82%	47,94%	61,34%
Coef. phi	0,12	0,20	0,04	0,18	-0,04	0,23
F1 Score	57,14	65,11	51,35	53,73	50,63	61,33
AUC	0,560	0,598	0,520	0,588	0,479	0,613

Fonte: Elaborada pela autora (2022)

Figura 23 – Matriz de Confusão - Subamostra 02 - Atraso ≥ 30 - TV



Fonte: (PYTHON..., 2017).

Quadro 12 – Avaliação de desempenho - Subgrupo Atraso ≥ 60 - TV

Subamostra 01						
Métrica	LR	DT	RF	BC	GB	AD
Acurácia	52,38%	61,90%	38,10%	61,90%	57,14%	52,38%
Precisão	52,38%	63,64%	40,00%	61,54%	57,14%	53,33%
VPN	0,00%	60,00%	36,36%	62,50%	57,14%	50,00%
Sensibilidade	100,00%	63,64%	36,36%	72,73%	72,73%	72,73%
Especificidade	0,00%	60,00%	40,00%	50,00%	40,00%	30,00%
Eficiência	50,00%	61,82%	38,18%	61,37%	56,37%	51,37%
Coef. phi	0,00	0,24	-0,24	0,23	0,13	0,03
F1 Score	68,75	63,64	38,09	66,67	64,00	61,54
AUC	0,500	0,618	0,382	0,614	0,564	0,514

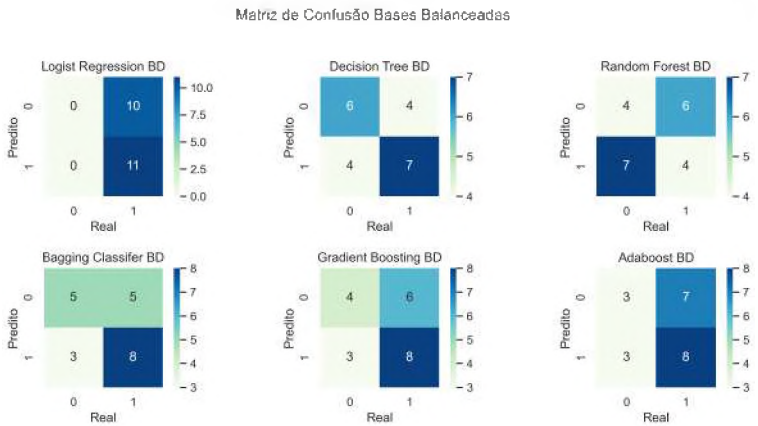
Fonte: Elaborada pela autora (2022)

Quadro 13 – Avaliação de desempenho - Subgrupo Atraso ≥ 60 - TV

Subamostra 02						
Métrica	LR	DT	RF	BC	GB	AD
Acurácia	47,62%	57,14%	61,90%	57,14%	52,38%	42,86%
Precisão	50,00%	58,33%	63,64%	60,00%	54,55%	45,45%
VPN	45,45%	55,56%	60,00%	54,55%	50,00%	40,00%
Sensibilidade	45,45%	63,64%	63,64%	54,55%	54,55%	45,45%
Especificidade	50,00%	50,00%	60,00%	60,00%	50,00%	40,00%
Eficiência	47,73%	56,82%	61,82%	57,27%	52,27%	42,73%
Coef. phi	-0,05	0,14	0,24	0,15	0,05	-0,15
F1 Score	47,62	60,87	63,64	57,15	54,55	45,45
AUC	0,477	0,568	0,618	0,573	0,523	0,427

Fonte: Elaborada pela autora (2022)

Figura 24 – Matriz de Confusão - Subamostra 01 - Atraso ≥ 60 - TV



Fonte: (PYTHON..., 2017).

Quadro 14 – Avaliação de desempenho - Subgrupo Atraso ≥ 60 - TV

Subamostra 03						
Métrica	LR	DT	RF	BC	GB	AD
Acurácia	52,38%	47,62%	52,38%	47,362%	38,10%	61,90%
Precisão	55,56%	50,00%	55,56%	50,00%	41,67%	61,54%
VPN	50,00%	44,44%	50,00%	40,00%	33,33%	62,50%
Sensibilidade	45,45%	54,55%	45,45%	72,73%	45,45%	72,73%
Especificidade	60,00%	40,00%	60,00%	20,00%	30,00%	50,00%
Eficiência	52,73%	47,27%	52,73%	46,37%	37,73%	61,37%
Coef. phi	0,06	-0,06	0,06	-0,09	-0,25	0,23
F1 Score	50,00	52,18	50,00	59,26	43,48	66,67
AUC	0,527	0,473	0,527	0,464	0,377	0,614

Fonte: Elaborada pela autora (2022)

Quadro 15 – Avaliação de desempenho - Subgrupo Atraso ≥ 60 - TV

Subamostra 04						
Métrica	LR	DT	RF	BC	GB	AD
Acurácia	52,38%	33,33%	52,38%	52,38%	57,14%	52,38%
Precisão	52,38%	40,00%	54,55%	52,38%	58,33%	52,94%
VPN	0,00%	16,67%	50,00%	0,00%	55,56%	50,00%
Sensibilidade	100,00%	54,55%	54,55%	100,00%	63,64%	81,82%
Especificidade	0,00%	10,00%	50,00%	00,00%	50,00%	20,00%
Eficiência	50,00%	32,27%	52,27%	50,00%	56,82%	50,91%
Coef. phi	0,00	-0,39	0,05	0,00	0,14	0,02
F1 Score	68,75	46,16	54,55	68,75	60,87	64,27
AUC	0,500	0,323	0,523	0,500	0,568	0,509

Fonte: Elaborada pela autora (2022)

Considerando o subgrupo *Atraso* ≥ 90, quando observamos o nível de inadimplência do conjunto de dados utilizados, 1,76%, observamos que, ao equilibrarmos a sensibilidade e a precisão, os algoritmos do tipo *Ensemble* performaram, mais uma vez, melhor que os tradicionais (*Logist Regression*) em grande parte das Subamostras avaliadas, conforme Quadros 17, 18, 19, 20 e 21, porém destacamos que os desempenhos aqui também foram inferiores àqueles observados no subgrupo *Atraso* ≥ 30 apresentados anteriormente, com exceção da Subamostra 01 e 05 que apresentaram resultados de F1 em torno de 70,00%, destaque para os algoritmos *AdaBoost* e *Gradiente Boosting*.

As matrizes de confusão e as métricas AUC resultantes da comparação entre os algoritmos das Subamostras 01 e 05, dentro do subgrupo Atraso ≥ 90, podem ser consultadas nas Figuras 25, 26, 27 e 28.

A métrica AUC apurada em cada algoritmo considerando todas as Subamostras

Quadro 16 – Avaliação de desempenho - Subgrupo Atraso ≥ 60 - TV

Subamostra 05						
Métrica	LR	DT	RF	BC	GB	AD
Acurácia	52,38%	47,62%	62,38%	33,33%	52,38%	47,62%
Precisão	52,38%	50,00%	54,55%	28,57%	53,85%	50,00%
VPN	0,00%	46,15%	50,00%	35,71%	50,00%	42,86%
Sensibilidade	100,00%	36,36%	54,55%	18,18%	63,64%	63,64%
Especificidade	0,00%	60,00%	50,00%	50,00%	40,00%	30,00%
Eficiência	50,00%	48,18%	52,27%	34,09%	51,82%	46,82%
Coef. phi	0,00	-0,04	0,05	-0,34	0,04	-0,07
F1 Score	68,75	42,10	54,55	22,22	58,34	56,00
AUC	0,500	0,482	0,523	0,341	0,518	0,468

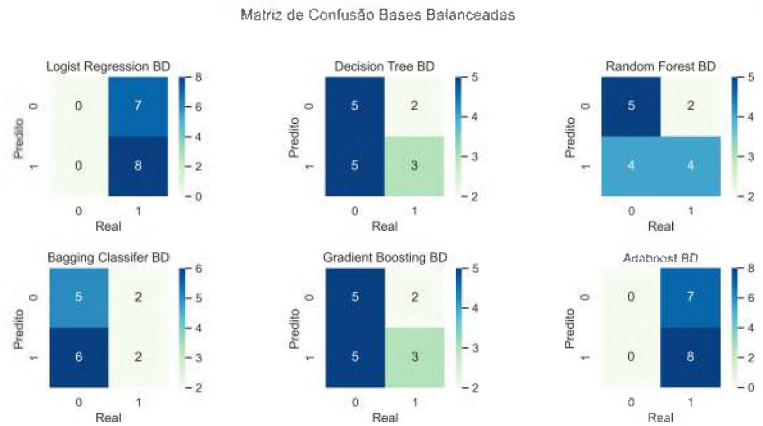
Fonte: Elaborada pela autora (2022)

Quadro 17 – Avaliação de desempenho - Subgrupo Atraso ≥ 90 - TV

Subamostra 01						
Métrica	LR	DT	RF	BC	GB	AD
Acurácia	53,33%	53,33%	60,00%	46,67%	53,33%	53,33%
Precisão	53,33%	60,00%	66,67%	50,00%	60,00%	53,33%
VPN	0,00%	50,00%	55,56%	45,45%	50,00%	0,00%
Sensibilidade	100,00%	37,50%	50,00%	25,00%	37,50%	100,00%
Especificidade	0,00%	71,43%	71,43%	71,43%	71,43%	0,00%
Eficiência	50,00%	54,47%	60,72%	48,22%	54,47%	50,00%
Coef. phi	0,00	0,09	0,22	-0,04	0,09	0,00
F1 Score	69,56	46,15	57,14	33,33	46,15	69,56
AUC	0,500	0,545	0,607	0,482	0,545	0,500

Fonte: Elaborada pela autora (2022)

Figura 25 – Matriz de Confusão - Subamostra 01 - Atraso ≥ 90 - TV



Fonte: (PYTHON..., 2017).

Quadro 18 – Avaliação de desempenho - Subgrupo Atraso ≥ 90 - TV

Subamostra 02						
Métrica	LR	DT	RF	BC	GB	AD
Acurácia	66,67%	66,67%	60,00%	40,00%	60,00%	40,00%
Precisão	80,00%	80,00%	75,00%	33,33%	62,50%	33,33%
VPN	60,00%	60,00%	54,55%	41,67%	57,14%	41,67%
Sensibilidade	50,00%	50,00%	37,50%	12,50%	62,50%	12,50%
Especificidade	85,71%	85,71%	85,71%	71,43%	57,14%	71,43%
Eficiência	67,85%	67,85%	61,60%	41,97%	59,82%	41,97%
Coef. phi	0,38	0,38	0,26	-0,20	0,20	-0,20
F1 Score	61,54	61,54	50,00	18,18	62,50	18,18
AUC	0,679	0,679	0,616	0,420	0,598	0,420

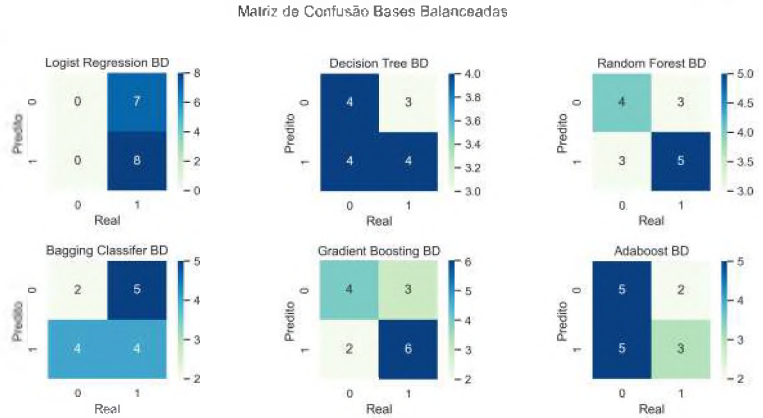
Fonte: Elaborada pela autora (2022)

Quadro 19 – Avaliação de desempenho - Subgrupo Atraso ≥ 90 - TV

Subamostra 03						
Métrica	LR	DT	RF	BC	GB	AD
Acurácia	66,67%	60,00%	60,00%	53,33%	60,00%	53,33%
Precisão	100,00%	62,50%	66,67%	100,00%	62,50%	54,55%
VPN	58,33%	57,14%	55,56%	50,00%	57,14%	50,00%
Sensibilidade	37,50%	62,50%	50,00%	12,50%	62,50%	75,00%
Especificidade	100,00%	57,14%	71,43%	100,00%	57,14%	28,57%
Eficiência	68,75%	59,82%	60,72%	56,25%	59,82%	51,78%
Coef. phi	0,47	0,20	0,22	0,25	0,20	0,04
F1 Score	54,55	62,50	57,14	22,22	62,50	63,16
AUC	0,688	0,598	0,607	0,562	0,598	0,518

Fonte: Elaborada pela autora (2022)

Figura 26 – Matriz de Confusão - Subamostra 05 - Atraso ≥ 90 - TV



Fonte: (PYTHON..., 2017).

Quadro 20 – Avaliação de desempenho - Subgrupo Atraso ≥ 90 - TV

Subamostra 04						
Métrica	LR	DT	RF	BC	GB	AD
Acurácia	53,33%	53,33%	40,00%	60,00%	26,67%	46,67%
Precisão	66,67%	57,14%	40,00%	75,00%	28,57%	50,00%
VPN	50,00%	50,00%	40,00%	54,55%	25,00%	44,44%
Sensibilidade	25,00%	50,00%	20,00%	37,50%	25,00%	37,50%
Especificidade	85,71%	57,14%	57,14%	85,71%	28,57%	57,14%
Eficiência	55,35%	53,57%	41,07%	61,60%	26,79%	47,32%
Coef. phi	0,13	0,07	-0,19	0,26	-0,46	-0,05
F1 Score	36,36	53,33	30,77	50,00	26,67	42,86
AUC	0,554	0,536	0,411	0,616	0,268	0,473

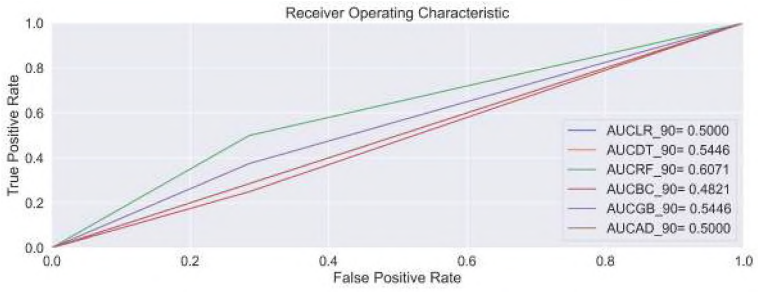
Fonte: Elaborada pela autora (2022)

Quadro 21 – Avaliação de desempenho - Subgrupo Atraso ≥ 90 - TV

Subamostra 05						
Métrica	LR	DT	RF	BC	GB	AD
Acurácia	53,33%	53,33%	60,00%	40,00%	66,67%	53,33%
Precisão	53,33%	57,14%	62,50%	44,44%	66,67%	60,00%
VPN	0,00%	50,00%	57,14%	33,33%	66,67%	50,00%
Sensibilidade	100,00%	50,00%	62,50%	50,00%	75,00%	37,50%
Especificidade	0,00%	57,14%	57,14%	28,57%	57,14%	71,43%
Eficiência	50,00%	53,57%	59,82%	39,28%	66,07%	54,47%
Coef. phi	0,00	0,07	0,20	-0,22	0,33	0,09
F1 Score	69,56	53,33	62,50	47,06	70,59	46,15
AUC	0,500	0,536	0,598	0,393	0,661	0,545

Fonte: Elaborada pela autora (2022)

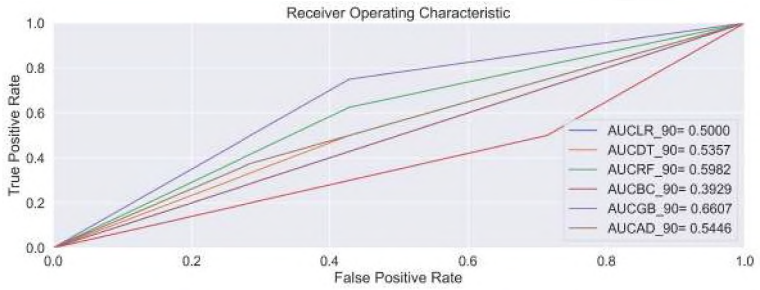
Figura 27 – AUC - Subamostra 01 - Atraso ≥ 90 - TV



Fonte: Elaborada pela autora(2022).

testadas foram disponibilizadas no Anexo E.

Figura 28 – AUC - Subamostra 05 - Atraso $\geq$ 90 - TV



Fonte: Elaborada pela autora(2022).

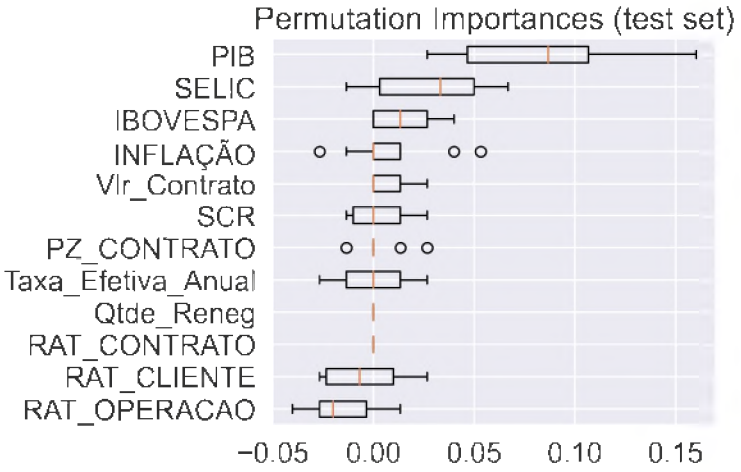
4.1.1 *Permutation feature importance*

Aplicamos PFI para o algoritmo *Random Forest* dentro do subgrupo *Atraso* $\geq$ 30 onde foi possível identificá-lo com o melhor desempenho em praticamente todas as subamostras geradas e determinamos as melhores *features*/variáveis usadas no modelo, ou seja, capturamos qual a influência de cada variável na previsões do modelo (BREIMAN, 2001), mesmo sabendo que a importância delas nesse modelo deriva de cálculos de estatísticas do conjunto de dados de treino e com isso podem ter uma alta importância mesmo para características que não são preditivas para a variável resposta, desde que o modelo tenha a capacidade de usá-las em excesso.

O presente estudo aplicou a técnica no modelo utilizando a base de treino e no modelo já treinado (em seu conjunto de teste), cujo desempenho foi o mais satisfatório e o resultado foi apresentado em uma lista decrescente de pontuações considerando a sensibilidade do desempenho.

O resultado teve como origem os dados da Subamostra 01 e as *features* mais importantes ratificam àquelas encontradas pelas técnicas de seleção de variáveis utilizadas no presente estudo, ver Figura 29 .

Figura 29 – PFI - Subamostra 01 - Atraso $\geq$ 30 - TV



Fonte: Elaborada pela autora(2022).

Não aplicamos PFI nos demais modelos observados nos demais subgrupos (Atraso $\geq$ 60 e Atraso $\geq$ 90), devido alternância de melhor desempenho entre os algoritmos comparados.

4.2 Aplicação das técnicas de seleção de variáveis

Aqui identificamos melhora do desempenho dos algoritmos quando foram retiradas variáveis das Subamostras em cada subgrupo *Atraso* $\geq$ 30, *Atraso* $\geq$ 60 e *Atraso* $\geq$ 90, conforme resultado das técnicas de seleção de *features* aplicadas no presente estudo.

Considerando o subgrupo *Atraso* $\geq$ 30, quando observamos o nível de inadimplência do conjunto de dados original de 8,59%, observamos que, ao equilibrarmos a sensibilidade e a precisão, o algoritmo *Random Forest* mostrou-se com melhor performance frente aos demais algoritmos, atingindo acurácia, precisão e F1 acima de 70,00% em mais de uma Subamostra selecionada, conforme Quadros 22, 23, 24, 25 e 26.

No subgrupo *Atraso* $\geq$ 30, em todas as simulações houve uma melhora de desempenho dos melhores algoritmos considerando cada Subamostra avaliada, demonstrando que a retirada das variáveis "Rating do Contrato", "Valor do Contrato"e "Quantidade de renegociações"aumentaram a performance do algoritmo *Random Forest*, em quatro Subamostras e melhor desempenho na Subamostra 02, e *Decision Tree.*, em uma Subamostra.

Quadro 22 – Avaliação de desempenho - Subgrupo Atraso $\geq$ 30 - STV

Subamostra 01						
Métrica	LR	DT	RF	BC	GB	AD
Acurácia	49,33%	58,67%	**72,00%**	54,67%	66,67%	50,67%
Precisão	50,00%	60,61%	**79,31%**	83,33%	70,97%	50,67%
VPN	48,84%	57,14%	**67,39%**	52,17%	63,64%	0,00%
Sensibilidade	42,11%	52,63%	**60,53%**	13,16%	57,89%	100,00%
Especificidade	56,76%	64,86%	**83,78%**	97,30%	75,68%	0,00%
Eficiência	49,44%	58,75%	**72,16%**	55,23%	66,78%	50,00%
Coef. phi	-0,01	0,18	**0,45**	0,19	0,34	0,00
F1 Score	45,72	56,34	**68,66**	22,73	63,77	67,26
AUC	0,494	0,587	**0,722**	0,552	0,668	0,500

Fonte: Elaborada pela autora (2022)

A matriz de confusão e a AUC resultantes da comparação entre os algoritmos na Subamostra 02, onde verificamos o melhor desempenho, encontram-se nas Figuras 30 e 31, respectivamente.

As matrizes de confusão e as métricas AUC resultantes da comparação entre os demais algoritmos em cada Subamostra do subgrupo Atraso $\geq$ 30 encontram-se consignadas

Quadro 23 – Avaliação de desempenho - Subgrupo Atraso ≥ 30 - STV

Subamostra 02						
Métrica	LR	DT	RF	BC	GB	AD
Acurácia	50,67%	61,33%	**72,00%**	65,33%	66,67%	65,33%
Precisão	50,67%	62,86%	**74,29%**	83,33%	66,67%	65,00%
VPN	0,00%	60,00%	**70,00%**	59,65%	66,67%	65,71%
Sensibilidade	100,00%	57,89%	**68,42%**	39,47%	68,42%	68,42%
Especificidade	0,00%	64,86%	**75,68%**	91,89%	64,86%	62,16%
Eficiência	50,00%	61,38%	**72,05%**	65,68%	66,64%	65,29%
Coef. phi	0,00	0,23	**0,44**	0,37	0,33	0,31
F1 Score	67,26	60,27	**71,23**	53,57	67,53	66,67
AUC	0,500	0,614	**0,720**	0,657	0,666	0,653

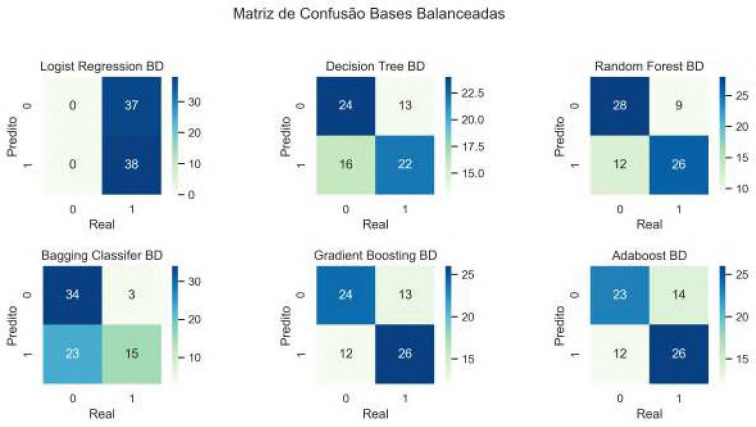
Fonte: Elaborada pela autora (2022)

Quadro 24 – Avaliação de desempenho - Subgrupo Atraso ≥ 30 - STV

Subamostra 03						
Métrica	LR	DT	RF	BC	GB	AD
Acurácia	40,00%	54,67%	**60,00%**	49,33%	60,00%	56,00%
Precisão	40,54%	59,09%	**60,53%**	0,00%	64,29%	57,14%
VPN	39,47%	52,83%	**59,46%**	48,65%	57,45%	55,00%
Sensibilidade	39,47%	34,21%	**60,53%**	0,00%	47,37%	52,63%
Especificidade	40,54%	75,68%	**59,46%**	97,30%	72,97%	59,46%
Eficiência	40,00%	54,95%	**60,00%**	48,65%	60,17%	56,05%
Coef. phi	-0,20	0,11	**0,20**	-0,12	0,21	0,12
F1 Score	40,00	43,33	**60,53**	0,00	54,55	54,79
AUC	0,400	0,549	**0,600**	0,500	0,602	0,560

Fonte: Elaborada pela autora (2022)

Figura 30 – Matriz de Confusão - Subamostra 02 - Atraso ≥ 30 - STV



Fonte: Elaborada pela autora(2022).

Quadro 25 – Avaliação de desempenho - Subgrupo Atraso ≥ 30 - STV

Subamostra 04						
Métrica	LR	DT	RF	BC	GB	AD
Acurácia	54,67%	58,67%	**61,33%**	46,67%	69,33%	54,67%
Precisão	54,76%	58,54%	**61,54%**	0,00%	69,23%	54,76%
VPN	54,55%	58,82%	**61,11%**	47,95%	69,44%	54,55%
Sensibilidade	60,53%	63,16%	**63,16%**	0,00%	71,05%	60,53%
Especificidade	48,65%	54,05%	**59,46%**	94,59%	67,57%	48,65%
Eficiência	54,59%	58,60%	**61,31%**	47,30%	69,31%	54,59%
Coef. phi	0,09	0,17	**0,23**	-0,17	0,39	0,09
F1 Score	57,50	60,76	**62,34**	0,00	70,13	57,50
AUC	0,546	0,586	**0,613**	0,473	0,693	0,546

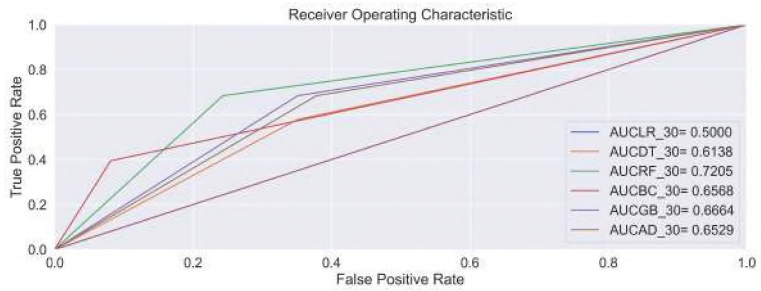
Fonte: Elaborada pela autora (2022)

Quadro 26 – Avaliação de desempenho - Subgrupo Atraso ≥ 30 - STV

Subamostra 05						
Métrica	LR	DT	RF	BC	GB	AD
Acurácia	53,33%	61,33%	58,67%	60,00%	62,67%	56,00%
Precisão	53,66%	59,57%	58,54%	65,38%	63,89%	56,41%
VPN	52,94%	64,29%	58,82%	57,14%	61,54%	55,56%
Sensibilidade	57,89%	73,68%	63,16%	44,74%	64,53%	57,89%
Especificidade	48,65%	48,65%	54,05%	75,68%	64,86%	54,05%
Eficiência	53,27%	61,17%	58,60%	60,21%	62,70%	55,97%
Coef. phi	0,07	0,23	0,17	0,21	0,25	0,12
F1 Score	55,69	65,88	60,76	53,13	62,16	57,14
AUC	0,533	0,512	0,586	0,602	0,627	0,560

Fonte: Elaborada pela autora (2022)

Figura 31 – AUC - Subamostra 02 - Atraso ≥ 30 - STV



Fonte: Elaborada pela autora(2022).

no Anexo F.

Considerando o subgrupo *Atraso* ≥ 60, quando observamos o nível de inadimplência do conjunto de dados utilizados, 2, 46%, observamos que, ao equilibrarmos a sensibilidade

e a precisão, os algoritmos do tipo *Ensemble*, em especial *Bagging Classifier* e *Gradient Boosting*, performaram melhor que os tradicionais (*Logist Regression*) em grande parte das Subamostras avaliadas, conforme Quadros 27, 28, 29, 30 e 31, porém destacamos que os desempenhos aqui foram inferiores àqueles observados no subgrupo *Atraso* ≥ 30 apresentado anteriormente.

Dentre os demais algoritmos comparados, as métricas sofreram bastante alteração quando excluíamos as variáveis "Rating do Contrato"e "Quantidade de renegociações"não sendo possível perceber um padrão de aumento de performance em todos os algoritmos.

Quadro 27 – Avaliação de desempenho - Subgrupo Atraso ≥ 60 - STV

Subamostra 01						
Métrica	LR	DT	RF	BC	GB	AD
Acurácia	52,38%	38,10%	33,33%	57,14%	52,38%	52,38%
Precisão	52,38%	40,00%	33,33%	57,14%	52,38%	52,38%
VPN	0,00%	36,36%	33,33%	57,14%	0,00%	0,00%
Sensibilidade	100,00%	36,36%	27,27%	72,73%	100,00%	100,00%
Especificidade	0,00%	40,00%	40,00%	40,00%	0,00%	0,00%
Eficiência	50,00%	38,18%	33,63%	56,37%	50,00%	50,00%
Coef. phi	0,00	-0,24	-0,33	0,13	0,00	0,00
F1 Score	68,75	38,09	30,00	64,00	68,75	68,75
AUC	0,500	0,382	0,336	0,564	0,500	0,500

Fonte: Elaborada pela autora (2022)

Quadro 28 – Avaliação de desempenho - Subgrupo Atraso ≥ 60 - STV

Subamostra 02						
Métrica	LR	DT	RF	BC	GB	AD
Acurácia	47,62%	52,38%	61,90%	61,90%	52,38%	52,38%
Precisão	50,00%	54,55%	63,64%	61,54%	54,55%	53,33%
VPN	45,45%	50,00%	60,00%	62,50%	50,00%	50,00%
Sensibilidade	45,45%	54,55%	63,64%	72,73%	54,55%	72,73%
Especificidade	50,00%	50,00%	60,00%	50,00%	50,00%	50,00%
Eficiência	47,73%	52,27%	61,82%	61,37%	52,27%	51,37%
Coef. phi	-0,05	0,05	0,24	0,23	0,05	0,03
F1 Score	47,62	54,55	63,64	66,67	54,55	61,54
AUC	0,477	0,523	0,618	0,614	0,523	0,514

Fonte: Elaborada pela autora (2022)

A matriz de confusão e a AUC resultantes da comparação entre os algoritmos na Subamostra 01 onde verificamos os algoritmos com melhores desempenhos, encontram-se nas Figuras 32 e 33, respectivamente.

Quadro 29 – Avaliação de desempenho - Subgrupo Atraso ≥ 60 - STV

Subamostra 03						
Métrica	LR	DT	RF	BC	GB	AD
Acurácia	52,38%	47,62%	42,86%	52,38%	52,38%	47,62%
Precisão	55,56%	50,00%	42,86%	60,00%	53,33%	50,00%
VPN	50,00%	33,33%	42,86%	50,00%	50,00%	42,86%
Sensibilidade	45,45%	81,82%	27,27%	27,27%	72,73%	63,64%
Especificidade	60,00%	10,00%	60,00%	80,00%	30,00%	30,00%
Eficiência	52,73%	45,91%	43,63%	53,63%	51,38%	46,82%
Coef. phi	0,06	-0,12	-0,13	0,09	0,03	-0,07
F1 Score	50,00	62,07	33,33	37,50	61,54	56,00
AUC	0,527	0,459	0,436	0,536	0,514	0,468

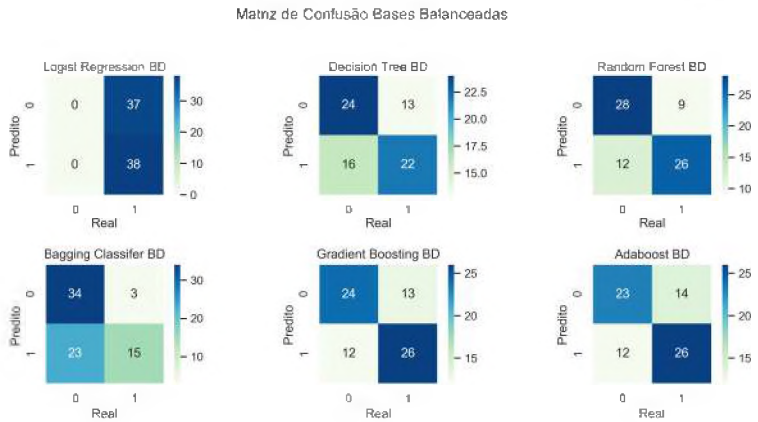
Fonte: Elaborada pela autora (2022)

Quadro 30 – Avaliação de desempenho - Subgrupo Atraso ≥ 60 - STV

Subamostra 04						
Métrica	LR	DT	RF	BC	GB	AD
Acurácia	52,38%	47,62%	47,62%	38,10%	61,90%	52,38%
Precisão	52,38%	50,00%	50,00%	40,00%	61,54%	53,85%
VPN	0,00%	40,00%	42,86%	36,36%	62,50%	50,00%
Sensibilidade	100,00%	72,73%	63,64%	36,36%	72,73%	63,64%
Especificidade	0,00%	20,00%	30,00%	40,00%	50,00%	40,00%
Eficiência	50,00%	46,37%	46,82%	38,18%	61,37%	51,82%
Coef. phi	0,00	-0,09	-0,07	-0,24	0,23	0,04
F1 Score	68,75	59,26	56,00	38,09	66,67	58,34
AUC	0,500	0,464	0,468	0,382	0,614	0,518

Fonte: Elaborada pela autora (2022)

Figura 32 – Matriz de Confusão - Subamostra 01 - Atraso ≥ 60 - STV



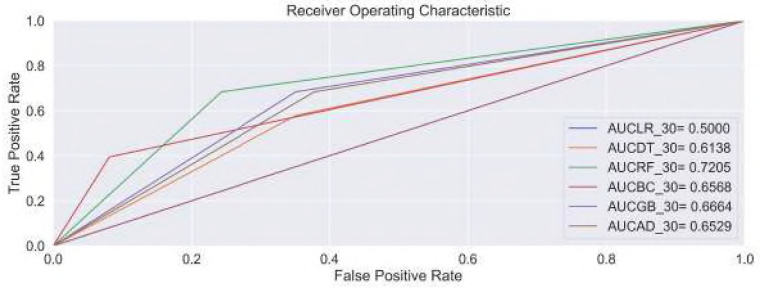
Fonte: Elaborada pela autora(2022).

Quadro 31 – Avaliação de desempenho - Subgrupo Atraso ≥ 60 - STV

Subamostra 05						
Métrica	LR	DT	RF	BC	GB	AD
Acurácia	52,38%	47,86%	33,33%	33,33%	52,38%	47,62%
Precisão	52,38%	44,44%	36,36%	40,00%	53,85%	50,00%
VPN	0,00%	41,67%	30,00%	16,67%	50,00%	42,86%
Sensibilidade	100,00%	36,36%	36,36%	54,55%	63,64%	63,64%
Especificidade	0,00%	50,00%	30,00%	10,00%	40,00%	30,00%
Eficiência	50,00%	43,18%	33,18%	32,27%	51,82%	46,82%
Coef. phi	0,00	-0,14	-0,34	-0,39	0,04	-0,07
F1 Score	68,75	40,00	36,36	46,16	58,34	56,00
AUC	0,500	0,432	0,332	0,323	0,518	0,468

Fonte: Elaborada pela autora (2022)

Figura 33 – AUC - Subamostra 01 - Atraso ≥ 60 - STV



Fonte: Elaborada pela autora(2022).

As matrizes de confusão e as métricas AUC resultantes da comparação entre os demais algoritmos em cada Subamostra do subgrupo Atraso ≥ 60 encontram-se consignadas no Anexo F.

Considerando o subgrupo *Atraso* ≥ 90, quando observamos o nível de inadimplência do conjunto de dados utilizados, 1, 76%, observamos que, ao equilibrarmos a sensibilidade e a precisão, os algoritmos do tipo *Ensemble*, em especial *Bagging Classifier* e *AdaBoost*, performaram, mais uma vez, melhor que os tradicionais (*Logist Regression*) em grande parte das Subamostras avaliadas, conforme Quadros 32, 33, 34, 35 e 36, chegando a apresentar um conjunto de métricas acima de 70, 00%, melhor resultado dentro as simulações realizadas no presente estudo.

As matrizes de confusão e as métricas AUC resultantes da comparação entre os algoritmos da Subamostra 02 e 05, dentro do subgrupo Atraso ≥ 90, podem ser consultadas nas Figuras 34, 35, 36 e 37.

As demais matrizes de confusão e AUC apuradas em cada algoritmo considerando todas as Subamostras testadas dentro do subgrupo Atraso ≥ 90 foram disponibilizadas no Anexo F.

Quadro 32 – Avaliação de desempenho - Subgrupo Atraso ≥ 90 - STV

Subamostra 01						
Métrica	LR	DT	RF	BC	GB	AD
Acurácia	53,33%	53,33%	46,67%	40,00%	40,00%	53,33%
Precisão	53,33%	60,00%	50,00%	44,44%	40,00%	53,33%
VPN	0,00%	50,00%	44,44%	33,33%	40,00%	0,00%
Sensibilidade	100,00%	37,50%	37,50%	50,00%	25,00%	100,00%
Especificidade	0,00%	71,43%	57,14%	28,57%	57,14%	0,00%
Eficiência	50,00%	54,47%	47,32%	39,28%	41,07%	50,00%
Coef. phi	0,00	0,09	0,-0,05	-0,22	-0,19	0,00
F1 Score	69,56	46,15	42,86	47,06	30,77	69,56
AUC	0,500	0,545	0,473	0,393	0,411	0,500

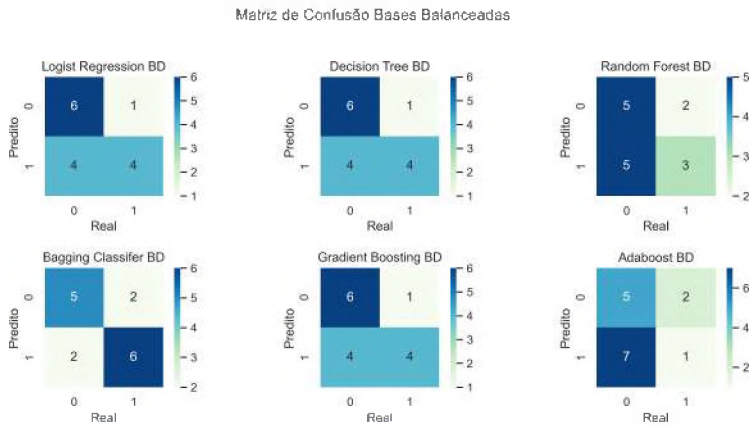
Fonte: Elaborada pela autora (2022)

Quadro 33 – Avaliação de desempenho - Subgrupo Atraso ≥ 90 - STV

Subamostra 02						
Métrica	LR	DT	RF	BC	GB	AD
Acurácia	66,67%	66,67%	53,33%	73,33%	66,67%	40,00%
Precisão	80,00%	80,00%	60,00%	75,00%	80,00%	33,33%
VPN	60,00%	60,00%	50,00%	71,43%	60,00%	41,67%
Sensibilidade	50,00%	50,00%	37,50%	75,00%	50,00%	12,50%
Especificidade	85,71%	85,71%	71,43%	71,43%	85,71%	71,43%
Eficiência	67,85%	67,85%	54,47%	73,22%	67,85%	41,97%
Coef. phi	0,38	0,38	0,09	0,46	0,38	-0,20
F1 Score	61,54	61,54	46,15	75,00	61,45	18,18
AUC	0,679	0,679	0,545	0,732	0,672	0,420

Fonte: Elaborada pela autora (2022)

Figura 34 – Matriz de Confusão - Subamostra 02 - Atraso ≥ 90 - STV



Fonte: (PYTHON..., 2017).

Quadro 34 – Avaliação de desempenho - Subgrupo Atraso ≥ 90 - STV

Subamostra 03						
Métrica	LR	DT	RF	BC	GB	AD
Acurácia	66,67%	60,00%	46,67%	53,33%	53,33%	53,33%
Precisão	100,00%	62,50%	50,00%	53,33%	54,55%	54,55%
VPN	58,33%	57,14%	44,44%	0,00%	50,00%	50,00%
Sensibilidade	37,50%	62,50%	37,50%	100,00%	75,00%	75,00%
Especificidade	100,00%	57,14%	57,14%	0,00%	28,57%	28,57%
Eficiência	68,75%	59,82%	47,32%	50,00%	51,78%	51,78%
Coef. phi	0,47	0,20	-0,05	0,00	0,04	-0,04
F1 Score	54,55	62,50	42,86	69,56	63,16	63,16
AUC	0,688	0,598	0,473	0,500	0,518	0,518

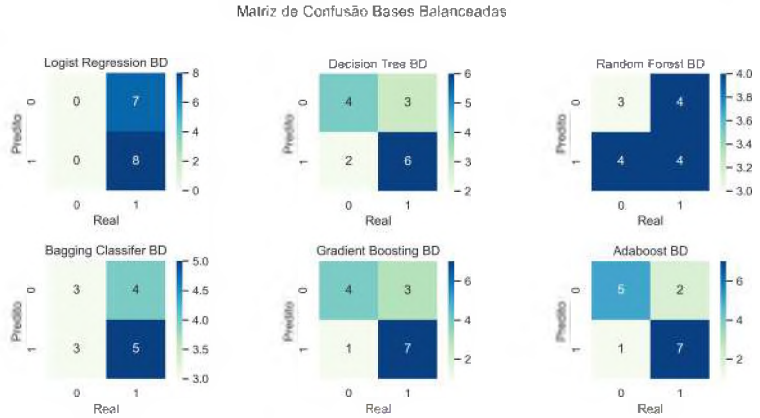
Fonte: Elaborada pela autora (2022)

Quadro 35 – Avaliação de desempenho - Subgrupo Atraso ≥ 90 - STV

Subamostra 04						
Métrica	LR	DT	RF	BC	GB	AD
Acurácia	53,33%	66,67%	46,67%	46,67%	26,67%	40,00%
Precisão	66,67%	80,00%	50,00%	50,00%	28,57%	42,86%
VPN	50,00%	60,00%	44,44%	40,00%	25,00%	37,50%
Sensibilidade	25,00%	50,00%	37,50%	62,50%	25,00%	37,50%
Especificidade	85,71%	85,71%	57,14%	28,57%	28,57%	42,86%
Eficiência	55,35%	67,85%	47,32%	45,53%	26,79%	40,18%
Coef. phi	0,13	0,38	-0,05	-0,09	-0,46	-0,20
F1 Score	36,36	61,54	42,86	55,56	26,67	40,00
AUC	0,554	0,679	0,473	0,455	0,268	0,402

Fonte: Elaborada pela autora (2022)

Figura 35 – Matriz de Confusão - Subamostra 05 - Atraso ≥ 90 - STV



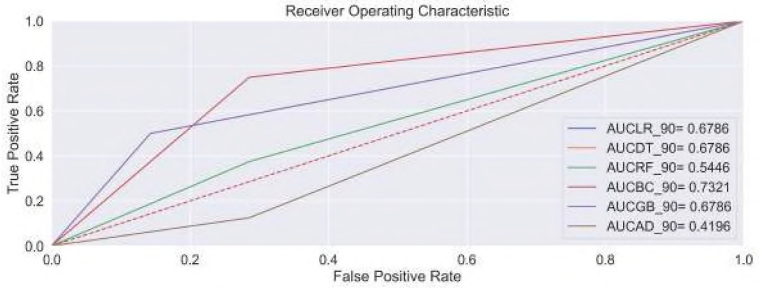
Fonte: (PYTHON..., 2017).

Quadro 36 – Avaliação de desempenho - Subgrupo Atraso ≥ 90 - STV

Subamostra 05						
Métrica	LR	DT	RF	BC	GB	AD
Acurácia	53,33%	66,67%	46,67%	53,33%	73,33%	**80,00%**
Precisão	53,33%	66,67%	50,00%	55,56%	70,00%	**77,78%**
VPN	0,00%	66,67%	42,86%	50,00%	80,00%	**83,33%**
Sensibilidade	100,00%	75,00%	50,00%	62,50%	87,50%	**87,50%**
Especificidade	0,00%	57,14%	42,86%	42,86%	57,14%	**71,43%**
Eficiência	50,00%	66,07%	46,43%	52,68%	72,32%	**79,47%**
Coef. phi	0,00	0,33	-0,07	0,05	0,47	**0,60**
F1 Score	69,56	70,59	50,00	58,83	77,78	**82,35**
AUC	0,500	0,661	0,464	0,527	0,723	**0,795**

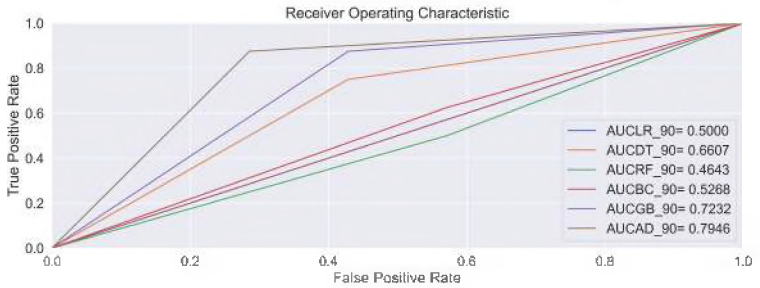
Fonte: Elaborada pela autora (2022)

Figura 36 – AUC - Subamostra 02 - Atraso ≥ 90 - STV



Fonte: Elaborada pela autora(2022).

Figura 37 – AUC - Subamostra 05 - Atraso ≥ 90 - STV



Fonte: Elaborada pela autora(2022).

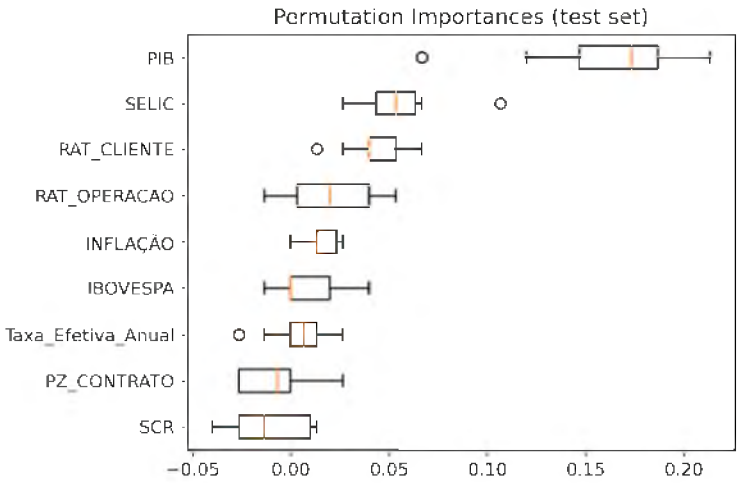
4.2.1 *Permutation feature importance*

Aplicamos PFI para o algoritmo *Random Forest* dentro do subgrupo *Atraso* ≥ 30 onde foi possível identificá-lo com o satisfatório desempenho dentro das simulações realizadas em cada Subamostra e determinamos as melhores *features*/variáveis usadas no modelo.

O resultado teve como origem os dados da Subamostra 02 e as *features* mais importantes ratificam àquelas encontradas pelas técnicas de seleção de variáveis utilizadas

no presente estudo, ver Figura 38 .

Figura 38 – PFI - Subamostra 02 - Atraso ≥ 30 - STV

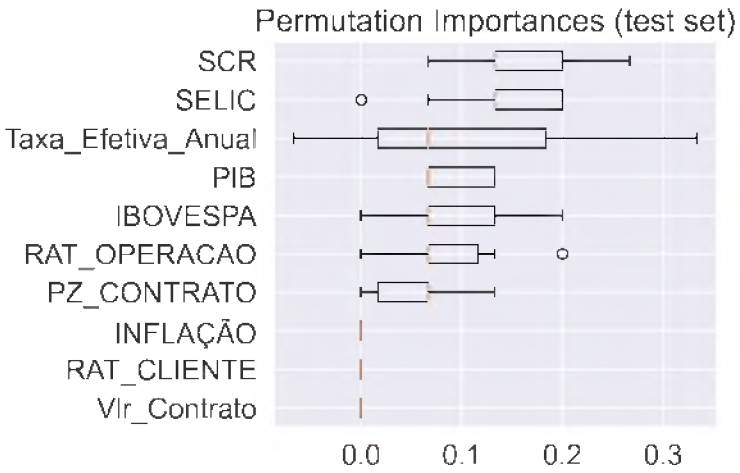


Fonte: Elaborada pela autora(2022).

Aplicamos PFI para o algoritmo *Adaboost* dentro do subgrupo *Atraso* ≥ 90 onde foi possível identificá-lo com o melhor desempenho dentro todos as simulações realizadas no presente estudo e determinamos as melhores *features*/variáveis usadas no modelo.

O resultado teve como origem os dados da Subamostra 05 e as *features* mais importantes ratificam àquelas encontradas pelas técnicas de seleção de variáveis utilizadas no presente estudo, ver Figura 39 .

Figura 39 – PFI - Subamostra 05 - Atraso ≥ 90 - STV



Fonte: Elaborada pela autora(2022).

Não aplicamos PFI nos demais modelos observados no subgrupo Atraso ≥ 60, devido alternância de melhor desempenho entre os algoritmos comparados.

5 Conclusões

Mesmo sem novas contribuições de programas emergenciais, o crédito bancário às empresas do segmento atacado continuou em alto crescimento no primeiro semestre de 2021 e com forte tendência de aumento no segundo semestre, apesar da materialização do risco de crédito bancário ter tido uma relativa estabilidade. Entretanto a possibilidade de recrudescimento da pandemia trouxe incertezas sobre as atividades das empresas, conforme BCB - Banco Central do Brasil (2021).

Métodos de avaliações e análises de concessão de crédito, além de seu respectivo monitoramento, por meio de critérios subjetivos foram amplamente aplicados no passado, porém com o uso de instrumentos mais sofisticados é possível mensurar melhor o risco de crédito, utilizando técnicas mais objetivas e com abordagens empíricas que enfatizam a previsão (ALTMAN; SAUNDERS, 1997), tornando possível evitar prováveis judicializações, melhorar acurácia da régua de cobrança, mitigar o risco operacional (erros normais em contratos) e subsidiar o controle da suficiência e constituição de garantias pactuadas (APOSTOLIK; CHRISTOPHER; PETER, 2009).

Os resultados aqui apresentados mostram que o aprimoramento no monitoramento de contratos de crédito já concedidos podem produzir resultados significativos em termos de redução da inadimplência, pois é possível predizer comportamentos, desenvolver controles adequados ou ajustar os já existentes minimizando a possibilidade de perdas financeiras para as instituições.

Os resultados mostram a superioridade dos algoritmos computacionais - dois melhores métodos (*Random Forest* e *AdaBoost*) - baseados em classificadores por *ensemble* e reforçam as contribuições apresentadas por Lessmann et al. (2015), Wang et al. (2011), Mason et al. (1999), Pandimurugan et al. (2021), Breiman (2001).

Na avaliação de desempenho do modelo dentro do subgrupo $Atraso \geq 90$, face aos diferentes critérios utilizados, a capacidade preditiva dos modelos melhorou à medida que aumentava o número de dias de atraso, apesar de poucas observações disponíveis nas Subamostras selecionadas, o que reforça a noção de que a utilização da variável *default* igual a 90 dias é adequada para obter bons resultados de um modelo em termos de classificação inadimplentes e não inadimplentes, aderente ao que preconiza a Brasil (2017b). Aqui destacamos que a performance dos algoritmos melhoraram sensivelmente quando as variáveis "*Rating* do Contrato"e "Quantidade de renegociações"foram retiradas dos exames.

Observando as métricas que se destacaram e o percentual de inadimplência do subgrupo $Atraso \geq 90$, especialmente, na Subamostra 05, onde o algoritmo *Adaboost* teve satisfatório resultado preditivo, alcançando uma sensibilidade de $87,50\%$, o montante

de empréstimos concedidos que poderia ser monitorado e selecionado para ação proativa pela instituição financeira remonta de, aproximadamente, R$ 71 milhões de reais, ou seja, dentro de apenas uma Subamostra seria possível implementar esforços para evitar tal valor de inadimplência pelos tomadores (aplicação da sensibilidade de 87,50% em R$ 80,7 milhões de reais, total de contratos ruins da Subamostra 05).

Já a capacidade preditiva dos modelos piorou à medida que os dias em atraso se concentravam no subgrupo $Atraso \geq 60$, não sendo conclusiva o desempenho dos modelos considerando todas as variáveis disponíveis ou excluindo algumas, porém observamos uma grande diversificação de performances entre possíveis algoritmos a serem escolhidos.

No caso do subgrupo $Atraso \geq 30$, o melhor desempenho foi também observado quando excluímos as variáveis "Valor do Contrato", "Rating do Contrato"e "Quantidade de renegociações", com destaque para o algoritmo *Random Forest* que confirmou o desempenho superior aos demais modelos testados, apresentando uma previsibilidade adequada em termos de desempenho conjunto.

Os resultados aqui apresentados também destacam que as variáveis macroeconômicas se mostram bem aderentes ao processo de modelagem com algoritmos de classificação para aspectos relacionados ao crédito no segmento atacado, ou seja, assim como realizado por Xia et al. (2021a) e Zhang e Thomas (2012), pois o resultado da PFI aplicado aos melhores algoritmos aqui evidenciados mostrou que essas variáveis têm produzido bons preditores para o risco de crédito presente no processo de predição avaliado em nosso estudo.

Nosso estudo apresenta algumas limitações. Primeiro, a problemática do desbalanceamento de classes agregada ao curto período de análises (horizonte de três anos) pela grande dificuldade de extração dos dados junto a instituição financeira o que não permitiu incluir mais variáveis em um horizonte de tempo maior e direcionou o estudo para uma extração aleatória de Subamostras em cada subgrupo, mesmo utilizando algoritmos que já tenha esse parâmetro em dua essência, como *Random Forest* e *Bagging Classifier* por meio do processo de geração de subamostras. Aqui também reforçamos a possibilidade de incluir outras operações de crédito comercial dentro do mesmo segmento. Em segundo lugar, conduzimos a técnica de seleção de hiper-parâmetros sempre equilibrando o poder computacional e a quantidade de parâmetros testados em cada modelo de modo que fosse possível gerar resultados tempestivamente (BROWN; MUES, 2012; CHAWLA et al., 2002; JAPKOWICZ et al., 2000).

De posse da matriz de correlação das variáveis disponíveis e do resultado das técnicas de seleção de variáveis aplicadas, possíveis melhorias em estudos futuros podem contemplar exclusão ou inclusão de diferentes *features* agregadas as variáveis macroeconômicas aqui selecionadas, ou adicionado outras como por exemplo váriáveis relacionadas ao mercado de trabalho, com o objeto de encontrar melhores desempenhos entre os algoritmos de

classificação.

Acrescentamos ainda a possibilidade de melhorar a etapa de pré-processamento de dados, observando a problemática de escalas quando utilizarmos modelos lineares, incluindo assim novas simulações contemplando técnicas de seleção de *features* e inclusão de novos modelos ou combinação entre eles.

Nesse sentido, essas limitações citadas podem ser objeto de melhorias para cada modelo ou mesmo uma combinação deles e podem ser examinadas detalhadamente em estudos futuros, expandido a análise para outros tipos de operações de crédito, onde a disponibilidade de informação é mais robusta.

Referências

ALTMAN, E. I.; SAUNDERS, A. Credit risk measurement: Developments over the last 20 years. *Journal of banking and finance*, Elsevier, v. 21, n. 11-12, p. 1721–1742, 1997. Citado na página 61.

APOSTOLIK, R.; CHRISTOPHER, D.; PETER, W. Foundations of banking risk: An overview of banking, banking risks, and risk-based banking regulation. John Wiley & Sons, p. 79, 2009. Citado 2 vezes nas páginas 2 e 61.

BAESENS, B. et al. Benchmarking state-of-the-art classification algorithms for credit scoring. *Journal of the Operational Research Society*, Taylor & Francis, v. 54, n. 6, p. 627–635, 2003. Disponível em: <https://doi.org/10.1057/palgrave.jors.2601545>. Citado na página 38.

BCB - BANCO CENTRAL DO BRASIL. *Relatório de Estabilidade Financeira*. [S.l.], 2021. v. 20, n. 1, 6 p. Disponível em: <https://www.bcb.gov.br/content/publicacoes/ref/202110/RELESTAB202110-refPub.pdf>. Citado na página 61.

BRASIL. Resolução do conselho monetário nacional nº 2.682. *Diário Oficial da União*, p. 19–20, 1999. Disponível em: <https://www.bcb.gov.br/pre/normativos/res/1999/pdf/res_2682_v2_L.pdf>. Citado 3 vezes nas páginas 10, 11 e 39.

BRASIL. Resolução do conselho monetário nacional nº 4.553. *Diário Oficial da União*, 2017. Disponível em: <https://www.bcb.gov.br/estabilidadefinanceira/exibenormativo?tipo=Resolu%C3%A7%C3%A3o&numero=4553>. Citado na página 5.

BRASIL. Resolução do conselho monetário nacional nº 4.557. *Diário Oficial da União*, p. 41–46, 2017. Disponível em: <https://www.bcb.gov.br/pre/normativos/busca/downloadNormativo.asp?arquivo=/Lists/Normativos/Attachments/50344/Res_4557_v1_O.pdf>. Citado 3 vezes nas páginas 2, 3 e 61.

BREIMAN, L. Bagging predictors. *Machine learning*, Springer, v. 24, n. 2, p. 123–140, 1996. Citado 3 vezes nas páginas 2, 24 e 25.

BREIMAN, L. Random forests. *Machine learning*, Springer, v. 45, n. 1, p. 5–32, 2001. Citado 4 vezes nas páginas 26, 29, 49 e 61.

BROWN, I.; MUES, C. An experimental comparison of classification algorithms for imbalanced credit scoring data sets. *Expert Systems with Applications*, Elsevier, v. 39, n. 3, p. 3446–3453, 2012. Citado 4 vezes nas páginas 27, 37, 38 e 62.

CHAWLA, N. V. et al. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, v. 16, p. 321–357, 2002. Citado na página 62.

CROOK, J. N.; EDELMAN, D. B.; THOMAS, L. C. Recent developments in consumer credit risk assessment. *European Journal of Operational Research*, v. 183, n. 3, p. 1447–1465, 2007. ISSN 0377-2217. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0377221706011866>. Citado na página 1.

DUMITRESCU, E. et al. Machine learning for credit scoring: Improving logistic regression with non-linear decision-tree effects. *European Journal of Operational Research*, Elsevier, v. 297, n. 3, p. 1178–1192, 2022. Citado na página 35.

FAWCETT, T. An introduction to roc analysis. *Pattern Recognition Letters*, v. 27, n. 8, p. 861–874, 2006. ISSN 0167-8655. ROC Analysis in Pattern Recognition. Disponível em: <https://www.sciencedirect.com/science/article/pii/S016786550500303X>. Citado na página 15.

FRIEDMAN, J. H. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, v. 38, p. 367–378, 2002. Citado 7 vezes nas páginas 2, 21, 23, 25, 26, 29 e 30.

GOLBAYANI, P.; FLORESCU, I.; CHATTERJEE, R. A comparative study of forecasting corporate credit ratings using neural networks, support vector machines, and decision trees. *The North American Journal of Economics and Finance*, v. 54, p. 101251, 2020. ISSN 1062-9408. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1062940820301480>. Citado 4 vezes nas páginas 2, 26, 27 e 34.

HASTIE, T. et al. *The elements of statistical learning: data mining, inference, and prediction*. [S.l.]: Springer, 2009. v. 2. Citado 8 vezes nas páginas 21, 22, 23, 26, 30, 31, 32 e 36.

HUANG, J.; LING, C. Using auc and accuracy in evaluating learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, v. 17, n. 3, p. 299–310, 2005. Citado 2 vezes nas páginas 2 e 38.

IMF - INTERNATIONAL MONETARY FUND. *Regional Economic Outlook. Western Hemisphere : pandemic persistence clouds the recovery*. [S.l.], 2020. 1-42 p. Disponível em: <https://www.elibrary.imf.org/view/books/086/29365-9781513558370-pt/29365-9781513558370-pt-book.xml>. Citado na página 5.

IZBICKI, R.; SANTOS, T. Machine learning sob a ótica estatística: Uma abordagem preditivista para estatística com exemplos em r. *Notes*, 2018. Citado 2 vezes nas páginas 29 e 34.

JAMES, G. et al. *An introduction to statistical learning*. [S.l.]: Springer, 2013. v. 112. Citado 13 vezes nas páginas 20, 21, 22, 26, 27, 30, 32, 34, 35, 36, 37, 39 e 40.

JAPKOWICZ, N. et al. Learning from imbalanced data sets: a comparison of various strategies. In: AAAI PRESS MENLO PARK, CA. *AAAI workshop on learning from imbalanced data sets*. [S.l.], 2000. v. 68, p. 10–15. Citado na página 62.

KRASKOV, A.; STÖGBAUER, H.; GRASSBERGER, P. Erratum: estimating mutual information [phys. rev. e 69, 066138 (2004)]. *Physical Review E*, APS, v. 83, n. 1, p. 019903, 2011. Citado na página 15.

KUHN, M.; JOHNSON, K. et al. *Applied predictive modeling*. [S.l.]: Springer, 2013. v. 26. Citado 12 vezes nas páginas 14, 20, 22, 26, 29, 30, 32, 33, 35, 36, 37 e 39.

LESSMANN, S. et al. Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research. *European Journal of Operational Research*, Elsevier, v. 247, n. 1, p. 124–136, 2015. Citado 3 vezes nas páginas 1, 35 e 61.

LIU, W.; FAN, H.; XIA, M. Credit scoring based on tree-enhanced gradient boosting decision trees. *Expert Systems with Applications*, Elsevier, v. 189, p. 116034, 2022. Citado na página 35.

LU, H.; MAZUMDER, R. Randomized gradient boosting machine. *SIAM Journal on Optimization*, SIAM, v. 30, n. 4, p. 2780–2808, 2020. Citado 2 vezes nas páginas 2 e 29.

MASON, L. et al. Boosting algorithms as gradient descent. *Advances in neural information processing systems*, v. 12, 1999. Citado na página 61.

OZON, M. S.; CHELA, J. L.; BERGMANN, D. R. Mensuração de probabilidade default e qualidade de crédito: uma aplicação no mercado brasileiro. *Caderno de Administração. Revista da Faculdade de Administração da FEA*, v. 12, n. 1, 2018. Citado 2 vezes nas páginas 11 e 12.

PANDIMURUGAN, V. et al. Random forest tree classification algorithm for predicating loan. *Materials Today: Proceedings*, Elsevier, 2021. Citado 5 vezes nas páginas 1, 6, 26, 27 e 61.

PYTHON machine learning: Machine learning and deep learning with python. [S.l.]: Packt Publishing, 2017. v. 2. Citado 20 vezes nas páginas 7, 14, 19, 23, 25, 29, 30, 31, 35, 36, 37, 43, 44, 46, 47, 56, 57, 115, 116 e 117.

ROSS, B. C. Mutual information between discrete and continuous data sets. *PloS one*, Public Library of Science San Francisco, USA, v. 9, n. 2, p. e87357, 2014. Citado na página 15.

SANTOS, E. M. d. et al. Teoria e aplicação de support vector machines à aprendizagem e reconhecimento de objetos baseado na aparênica. Universidade Federal de Campina Grande, 2002. Citado na página 35.

SCHAPIRE, R. E. The strength of weak learnability. *Machine learning*, Springer, v. 5, n. 2, p. 197–227, 1990. Citado na página 28.

VIEIRA, J. R. de C. et al. Machine learning models for credit analysis improvements: predicting low-income families' default. *Applied Soft Computing*, Elsevier, v. 83, p. 105640, 2019. Citado 3 vezes nas páginas 1, 3 e 35.

WANG, G. et al. A comparative assessment of ensemble learning for credit scoring. *Expert systems with applications*, Elsevier, v. 38, n. 1, p. 223–230, 2011. Citado 8 vezes nas páginas 2, 24, 25, 28, 29, 30, 31 e 61.

XIA, Y. et al. A dynamic credit scoring model based on survival gradient boosting decision tree approach. *Technological and Economic Development of Economy*, v. 27, n. 1, p. 96–119, 2021. Citado 4 vezes nas páginas 1, 2, 7 e 62.

XIA, Y. et al. Incorporating multilevel macroeconomic variables into credit scoring for online consumer lending. *Electronic Commerce Research and Applications*, v. 49, p. 101095, 2021. ISSN 1567-4223. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1567422321000673>. Citado na página 1.

ZHANG, J.; THOMAS, L. C. Comparisons of linear regression and survival analysis using single and mixture distributions approaches in modelling lgd. *International Journal of Forecasting*, v. 28, n. 1, p. 204–215, 2012. ISSN 0169-2070. Special Section 1: The Predictability of Financial Markets Special Section 2: Credit Risk Modelling and Forecasting. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0169207010001160>. Citado 2 vezes nas páginas 2 e 62.

Anexos

ANEXO A – Séries Macroeconômicas utilizadas

Mensal	IPCA	SELIC	IBOVESPA	PIB
Jan/15	1,24	0,94	26,96	474.568,80
fev/15	1,22	0,82	19,4	466.530,30
Mar/15	1,32	1,04	23,26	515.567,30
abr/15	0,71	0,95	18,53	496.922,80
mai/15	0,74	0,99	18,77	492.304,40
Jun/15	0,79	1,07	18,17	490.749,80
Jul/15	0,62	1,18	18,33	507.302,10
ago/15	0,22	1,11	26,88	501.180,70
set/15	0,54	1,11	25,12	499.745,40
out/15	0,82	1,11	25,39	521.183,00
Nov/15	1,01	1,06	28,11	513.506,10
dez/15	0,96	1,16	25,37	516.226,30
Jan/16	1,27	1,06	28,93	481.956,80
fev/16	0,9	1	33,97	490.146,50
Mar/16	0,43	1,16	37,9	528.164,90
abr/16	0,61	1,06	30,65	520.527,50
mai/16	0,78	1,11	21,81	516.542,20
Jun/16	0,35	1,16	26,48	521.949,20
Jul/16	0,52	1,11	13,07	521.908,60
ago/16	0,44	1,22	16,87	530.685,60
set/16	0,08	1,11	23,94	524.575,90
out/16	0,26	1,05	12,24	539.540,10
Nov/16	0,18	1,04	32,28	545.957,70
dez/16	0,3	1,12	23,71	547.373,20

Jan/17	0,38	1,09	20,36	513.463,20
fev/17	0,33	0,87	16,53	511.071,60
Mar/17	0,25	1,05	20,76	561.271,50
abr/17	0,14	0,79	16,73	537.487,90
mai/17	0,31	0,93	37,32	550.717,70
Jun/17	-0,23	0,81	13,33	542.647,00
Jul/17	0,24	0,8	10,99	548.229,10
ago/17	0,19	0,8	12,31	555.895,70
set/17	0,16	0,64	13,21	544.510,10
out/17	0,42	0,64	16,58	568.349,50
Nov/17	0,28	0,57	23,17	573.886,10
dez/17	0,44	0,54	14,15	577.949,60
Jan/18	0,29	0,58	18,36	552.168,60
fev/18	0,32	0,47	23,78	539.714,30
Mar/18	0,09	0,53	13,5	589.695,90
abr/18	0,22	0,52	16,47	587.351,20
mai/18	0,4	0,52	26,29	561.866,90
Jun/18	1,26	0,52	23,95	584.444,90
Jul/18	0,33	0,54	15,67	591.895,90
ago/18	-0,09	0,57	23,5	599.045,80
set/18	0,48	0,47	20,86	576.932,00
out/18	0,45	0,54	32,67	612.249,20
Nov/18	-0,21	0,49	21,92	607.926,30
dez/18	0,15	0,49	18,05	600.850,20
Jan/19	0,32	0,54	17,69	578.912,10
fev/19	0,43	0,49	21,26	577.023,40
Mar/19	0,75	0,47	26,4	603.377,10
abr/19	0,57	0,52	16,98	614.167,20
mai/19	0,13	0,54	19,89	616.580,80
Jun/19	0,01	0,47	15,77	598.016,20

Jul/19	0,19	0,57	13,35	632.182,90
ago/19	0,11	0,5	24,18	629.413,40
set/19	-0,04	0,46	10,44	619.035,60
out/19	0,1	0,48	17,6	649.556,90
Nov/19	0,51	0,38	14,44	637.034,30
dez/19	1,15	0,37	10,2	633.831,20
Jan/20	0,21	0,38	20,85	606.762,00
fev/20	0,25	0,29	33,32	610.938,80
Mar/20	0,07	0,34	123,94	627.860,70
abr/20	-0,31	0,28	47,59	559.922,20
mai/20	-0,38	0,24	31,41	566.831,80
Jun/20	0,26	0,21	26,3	595.694,40
Jul/20	0,36	0,19	21,5	629.678,80
ago/20	0,24	0,16	21,33	623.299,40
set/20	0,64	0,16	22,24	635.263,90
out/20	0,86	0,16	25,65	660.759,80
Nov/20	0,89	0,15	22,3	665.124,80
dez/20	1,35	0,16	15,4	685.479,80
Jan/21	0,25	0,15	25,37	654.087,80
fev/21	0,86	0,13	27,7	677.190,50
Mar/21	0,93	0,2	23,27	734.676,40
abr/21	0,31	0,21	13,52	715.039,00
mai/21	0,83	0,27	15,74	710.892,60
Jun/21	0,53	0,31	11,83	714.671,30
Jul/21	0,96	0,36	19,46	744.744,50
ago/21	0,87	0,43	16,95	739.331,00
set/21	1,16	0,44	26,43	731.109,10
out/21	1,25	0,49	25,02	748.125,20
Nov/21	0,95	0,59	22,34	747.697,30

Fonte: Elaborado pela Autora (2022)

ANEXO B – Cálculo das variáveis macroeconômicas

O código abaixo, escrito em VBA, demonstra o tratamento realizado no conjunto de dados para cálculo das variáveis macroeconômicas utilizados no processo de modelagem:

```
Sub ibovespa30()

    Dim lin_inicial As Integer
    Dim lin_fim As Integer
    Dim i As Integer

    i = 2

'caminhar na coluna

    While Sheets("BASE >=30").Range("A" & i) <> ""

        lin_inicial = Sheets("BASE >=30").Range("S" & i).Value
        lin_fim = Sheets("BASE >=30").Range("T" & i).Value
        Sheets("BASE >=30").Range("U" & i) =
            Application.WorksheetFunction.StDev(Sheets("VARIA_MACRO").Range("D"
            & lin_inicial & ":" & "D" & lin_fim))
        i = i + 1

    Wend

End Sub


Sub ibovespa60()

    Dim lin_inicial As Integer
    Dim lin_fim As Integer
    Dim i As Integer

    i = 2
```

```
'caminhar na coluna

    While Sheets("BASE>=60").Range("A" & i) <> ""

        lin_inicial = Sheets("BASE>=60").Range("S" & i).Value
        lin_fim = Sheets("BASE>=60").Range("T" & i).Value
        Sheets("BASE>=60").Range("U" & i) =
            Application.WorksheetFunction.StDev(Sheets("VARIA_MACRO").Range("D"
            & lin_inicial & ":" & "D" & lin_fim))
        i = i + 1

    Wend

End Sub

Sub ibovespa90()

    Dim lin_inicial As Integer
    Dim lin_fim As Integer
    Dim i As Integer

    i = 2

'caminhar na coluna

    While Sheets("BASE>=90").Range("A" & i) <> ""

        lin_inicial = Sheets("BASE>=90").Range("S" & i).Value
        lin_fim = Sheets("BASE>=90").Range("T" & i).Value
        Sheets("BASE>=90").Range("U" & i) =
            Application.WorksheetFunction.StDev(Sheets("VARIA_MACRO").Range("D"
            & lin_inicial & ":" & "D" & lin_fim))
        i = i + 1

    Wend

End Sub

Sub inflacao30()

    Dim lin_inicial As Integer
    Dim lin_fim As Integer
```

```
    Dim i As Integer

    i = 2

'caminhar na coluna

    While Sheets("BASE >=30").Range("A" & i) <> ""

        lin_inicial = Sheets("BASE >=30").Range("S" & i).Value
        lin_fim = Sheets("BASE >=30").Range("T" & i).Value
        Sheets("BASE >=30").Range("V" & i) =
            Application.WorksheetFunction.StDev(Sheets("VARIA_MACRO").Range("B"
            & lin_inicial & ":" & "B" & lin_fim))
        i = i + 1

    Wend

End Sub

Sub inflacao60()

    Dim lin_inicial As Integer
    Dim lin_fim As Integer
    Dim i As Integer

    i = 2

'caminhar na coluna

    While Sheets("BASE>=60").Range("A" & i) <> ""

        lin_inicial = Sheets("BASE>=60").Range("S" & i).Value
        lin_fim = Sheets("BASE>=60").Range("T" & i).Value
        Sheets("BASE>=60").Range("V" & i) =
            Application.WorksheetFunction.StDev(Sheets("VARIA_MACRO").Range("B"
            & lin_inicial & ":" & "B" & lin_fim))
        i = i + 1

    Wend

End Sub
```

```
Sub inflacao90()

    Dim lin_inicial As Integer
    Dim lin_fim As Integer
    Dim i As Integer

    i = 2

'caminhar na coluna

    While Sheets("BASE>=90").Range("A" & i) <> ""

        lin_inicial = Sheets("BASE>=90").Range("S" & i).Value
        lin_fim = Sheets("BASE>=90").Range("T" & i).Value
        Sheets("BASE>=90").Range("V" & i) =
            Application.WorksheetFunction.StDev(Sheets("VARIA_MACRO").Range("B"
            & lin_inicial & ":" & "B" & lin_fim))
        i = i + 1

    Wend

End Sub

Sub selic30()

    Dim lin_inicial As Integer
    Dim lin_fim As Integer
    Dim i As Integer

    i = 2

'caminhar na coluna

    While Sheets("BASE >=30").Range("A" & i) <> ""

        lin_inicial = Sheets("BASE >=30").Range("S" & i).Value
        lin_fim = Sheets("BASE >=30").Range("T" & i).Value
        Sheets("BASE >=30").Range("W" & i) =
            (((Sheets("VARIA_MACRO").Range("C" & lin_fim)) /
            (Sheets("VARIA_MACRO").Range("C" & lin_inicial))) ^ (1 / (lin_fim -
            lin_inicial))) - 1
        i = i + 1
```

```
    Wend

End Sub

Sub selic60()

    Dim lin_inicial As Integer
    Dim lin_fim As Integer
    Dim i As Integer

    i = 2

'caminhar na coluna

    While Sheets("BASE>=60").Range("A" & i) <> ""

        lin_inicial = Sheets("BASE>=60").Range("S" & i).Value
        lin_fim = Sheets("BASE>=60").Range("T" & i).Value
        Sheets("BASE>=60").Range("W" & i) = (((Sheets("VARIA_MACRO").Range("C"
            & lin_fim)) / (Sheets("VARIA_MACRO").Range("C" & lin_inicial))) ^
            (1 / (lin_fim - lin_inicial))) - 1
        i = i + 1

    Wend

End Sub

Sub selic90()

    Dim lin_inicial As Integer
    Dim lin_fim As Integer
    Dim i As Integer

    i = 2

'caminhar na coluna

    While Sheets("BASE>=90").Range("A" & i) <> ""

        lin_inicial = Sheets("BASE>=90").Range("S" & i).Value
        lin_fim = Sheets("BASE>=90").Range("T" & i).Value
```

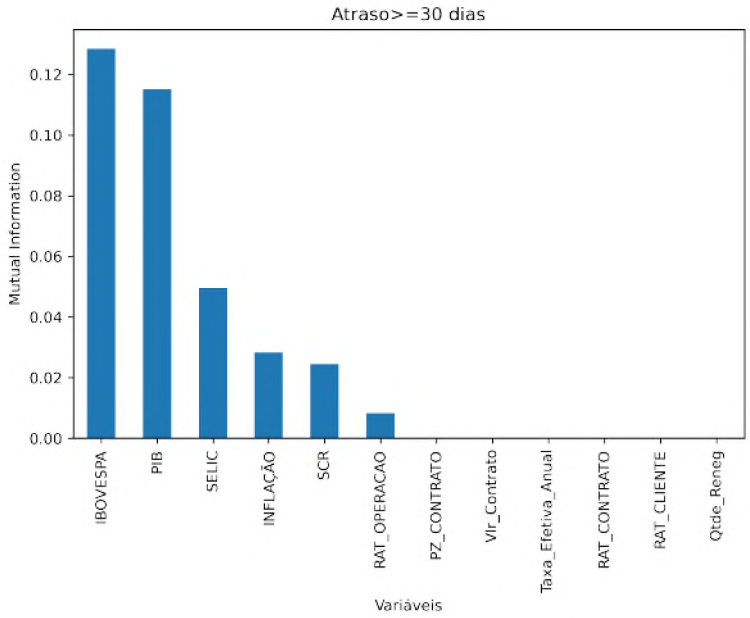
```
        Sheets("BASE>=90").Range("W" & i) = (((Sheets("VARIA_MACRO").Range("C"
            & lin_fim)) / (Sheets("VARIA_MACRO").Range("C" & lin_inicial))) ^
            (1 / (lin_fim - lin_inicial))) - 1
        i = i + 1

    Wend

End Sub


Sub pib30()

    Dim lin_inicial As Integer
    Dim lin_fim As Integer
    Dim i As Integer

    i = 2

'caminhar na coluna

    While Sheets("BASE >=30").Range("A" & i) <> ""

        lin_inicial = Sheets("BASE >=30").Range("S" & i).Value
        lin_fim = Sheets("BASE >=30").Range("T" & i).Value
        Sheets("BASE >=30").Range("X" & i) =
            (((Sheets("VARIA_MACRO").Range("E" & lin_fim)) /
            (Sheets("VARIA_MACRO").Range("E" & lin_inicial))) ^ (1 / (lin_fim -
            lin_inicial))) - 1
        i = i + 1

    Wend

End Sub

Sub pib60()

    Dim lin_inicial As Integer
    Dim lin_fim As Integer
    Dim i As Integer

    i = 2
```

```
'caminhar na coluna

    While Sheets("BASE>=60").Range("A" & i) <> ""

        lin_inicial = Sheets("BASE>=60").Range("S" & i).Value
        lin_fim = Sheets("BASE>=60").Range("T" & i).Value
        Sheets("BASE>=60").Range("X" & i) = (((Sheets("VARIA_MACRO").Range("E"
            & lin_fim)) / (Sheets("VARIA_MACRO").Range("E" & lin_inicial))) ^
            (1 / (lin_fim - lin_inicial))) - 1
        i = i + 1

    Wend

End Sub

Sub pib90()

    Dim lin_inicial As Integer
    Dim lin_fim As Integer
    Dim i As Integer

    i = 2

'caminhar na coluna

    While Sheets("BASE>=90").Range("A" & i) <> ""

        lin_inicial = Sheets("BASE>=90").Range("S" & i).Value
        lin_fim = Sheets("BASE>=90").Range("T" & i).Value
        Sheets("BASE>=90").Range("X" & i) = (((Sheets("VARIA_MACRO").Range("E"
            & lin_fim)) / (Sheets("VARIA_MACRO").Range("E" & lin_inicial))) ^
            (1 / (lin_fim - lin_inicial))) - 1
        i = i + 1

    Wend

End Sub
```

ANEXO C – Técnicas para seleção de variáveis

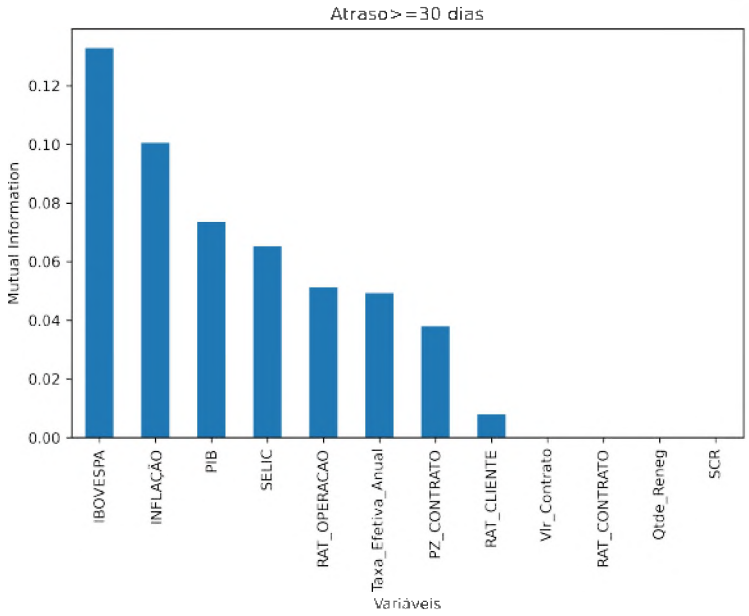
A aplicação das técnicas de seleção de variáveis foi realizada em cada subgrupo considerando a situação do *dataset* gerado aleatoriamente. Seguem exemplos de *outputs* da aplicação realizada:

Figura 40 – *MIC e SelectKBest - Subamostra 01 - Atraso* ≥ 30



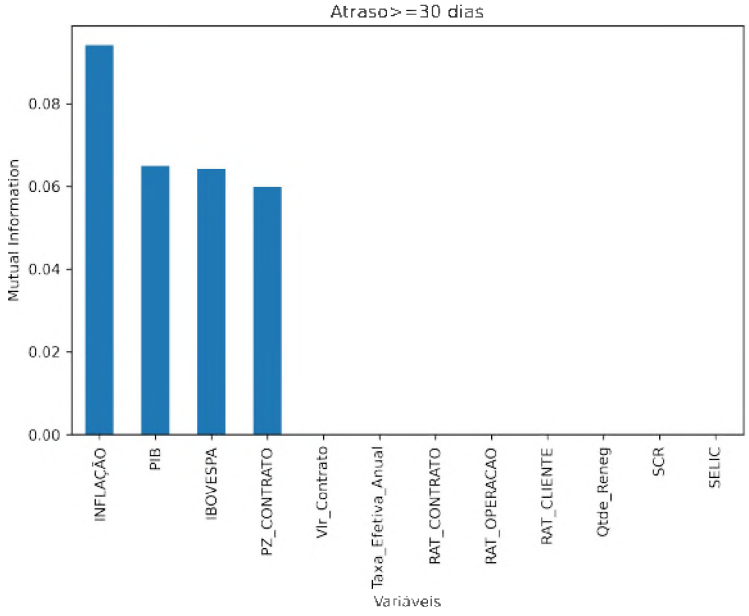
Fonte: Elaborado pela autora (2022).

Figura 41 – *MIC e SelectKBest - Subamostra 02 - Atraso* ≥ 30



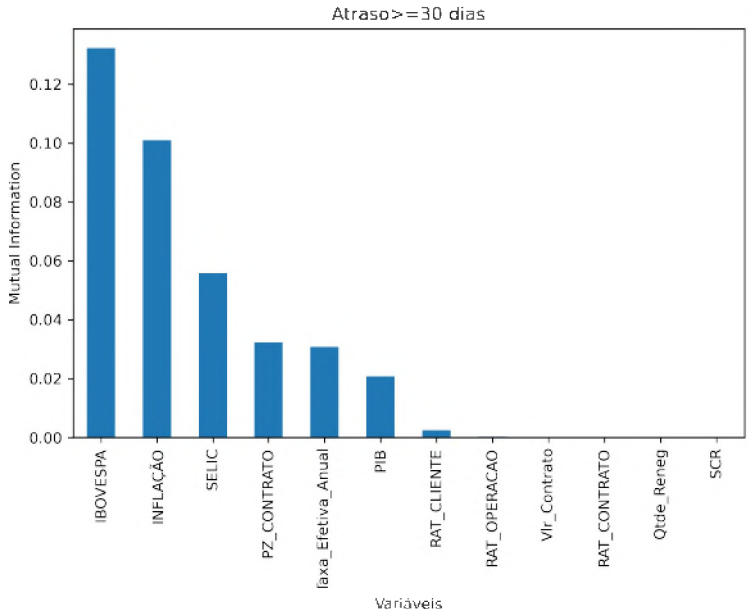
Fonte: Elaborado pela autora (2022).

Figura 42 – *MIC e SelectKBest - Subamostra 03 - Atraso* ≥ 30



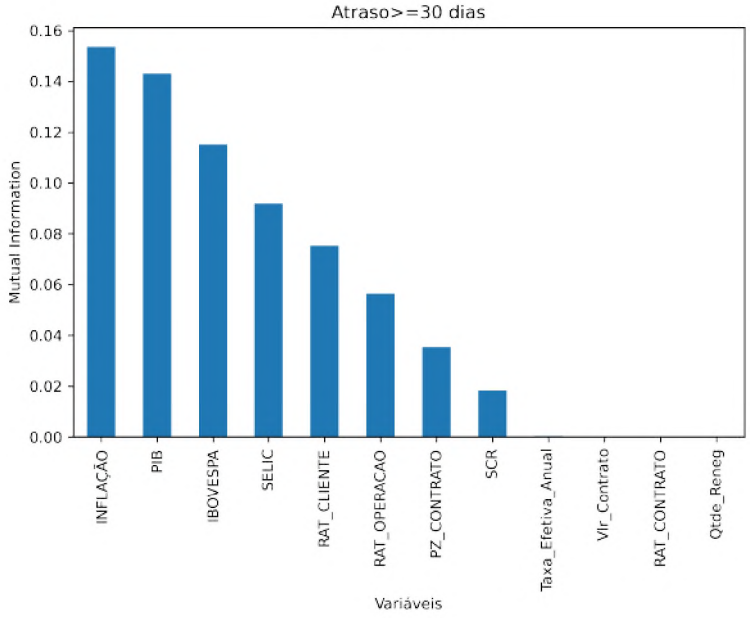
Fonte: Elaborado pela autora (2022).

Figura 43 – *MIC e SelectKBest - Subamostra 04 - Atraso* ≥ 30



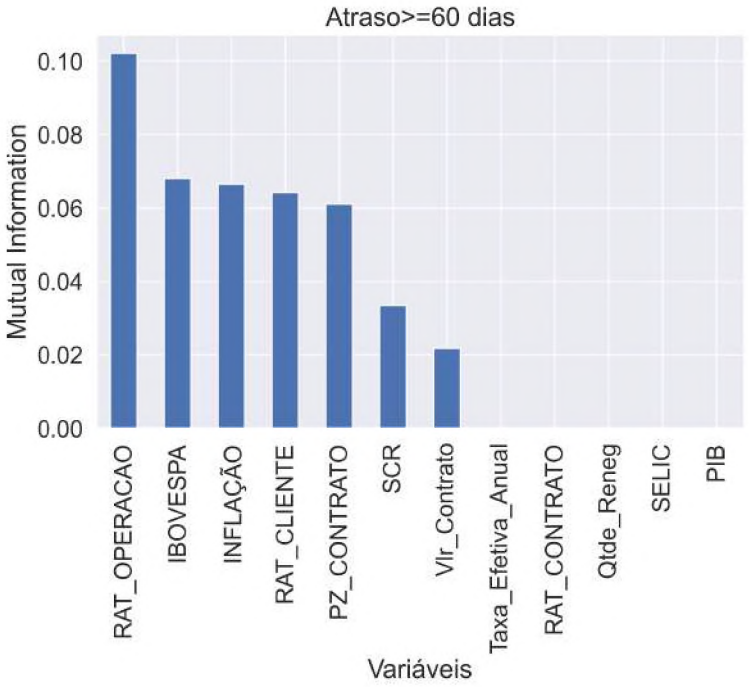
Fonte: Elaborado pela autora (2022).

Figura 44 – *MIC e SelectKBest - Subamostra 05 - Atraso* ≥ 30



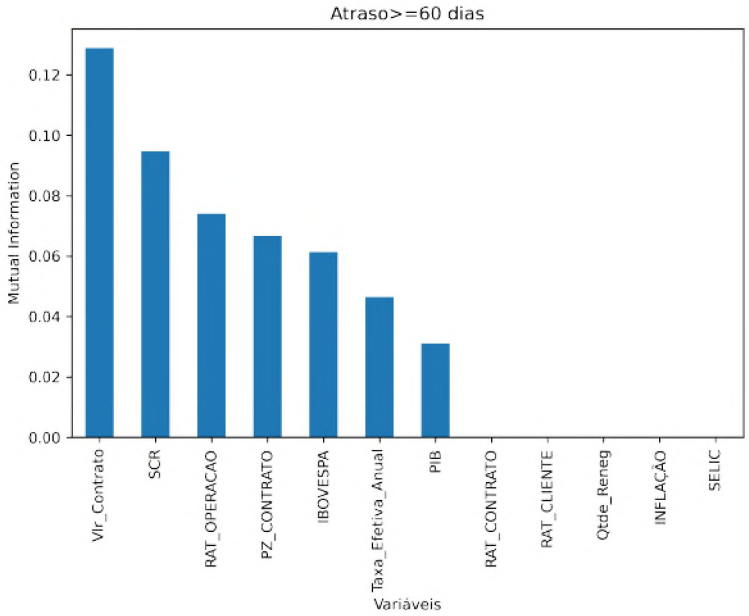
Fonte: Elaborado pela autora (2022).

Figura 45 – *MIC e SelectKBest - Subamostra 01 - Atraso* ≥ 60



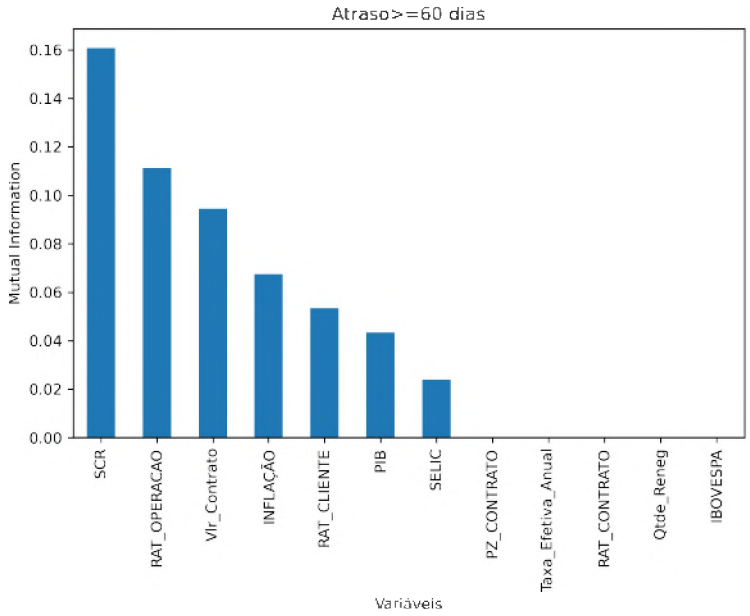
Fonte: Elaborado pela autora (2022).

Figura 46 – *MIC e SelectKBest - Subamostra 02 - Atraso* ≥ 60



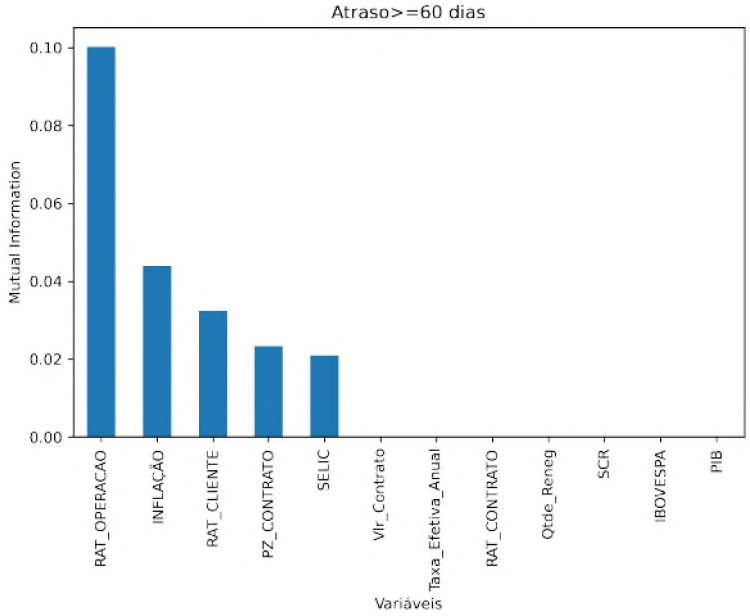
Fonte: Elaborado pela autora (2022).

Figura 47 – *MIC e SelectKBest - Subamostra 03 - Atraso* ≥ 60



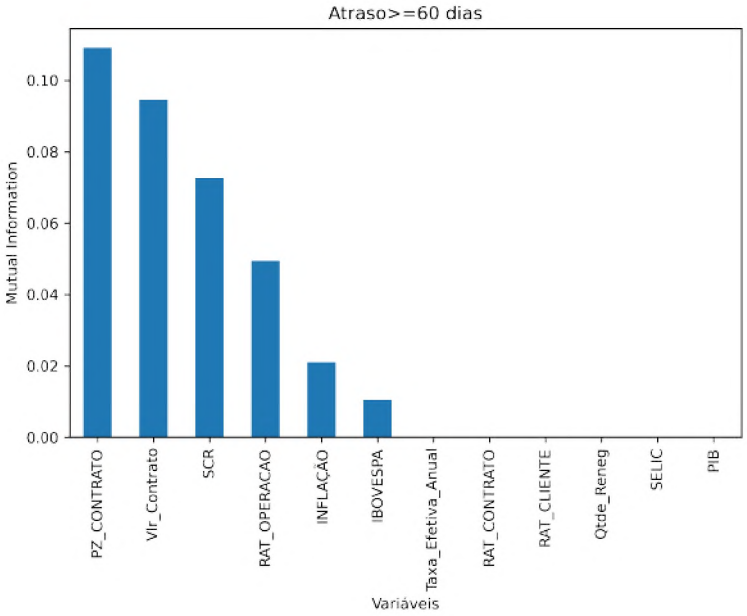
Fonte: Elaborado pela autora (2022).

Figura 48 – *MIC e SelectKBest - Subamostra 04 - Atraso* ≥ 60



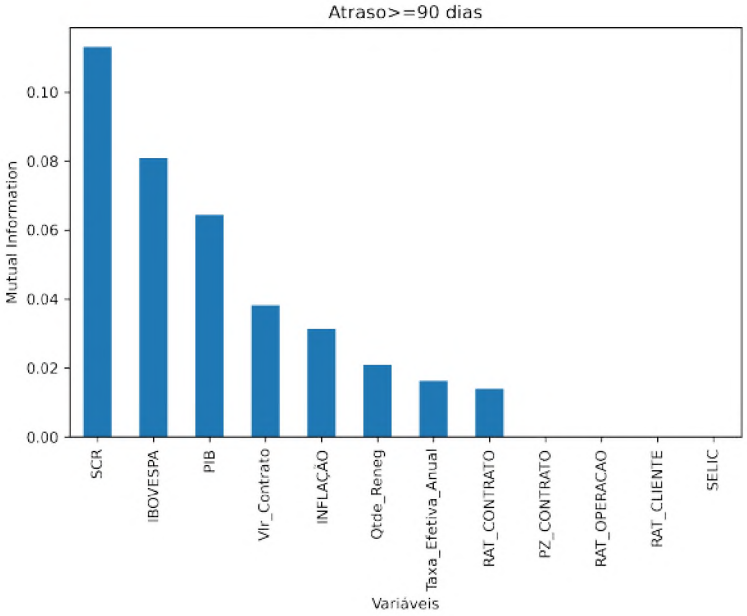
Fonte: Elaborado pela autora (2022).

Figura 49 – *MIC e SelectKBest - Subamostra 05 - Atraso* ≥ 60



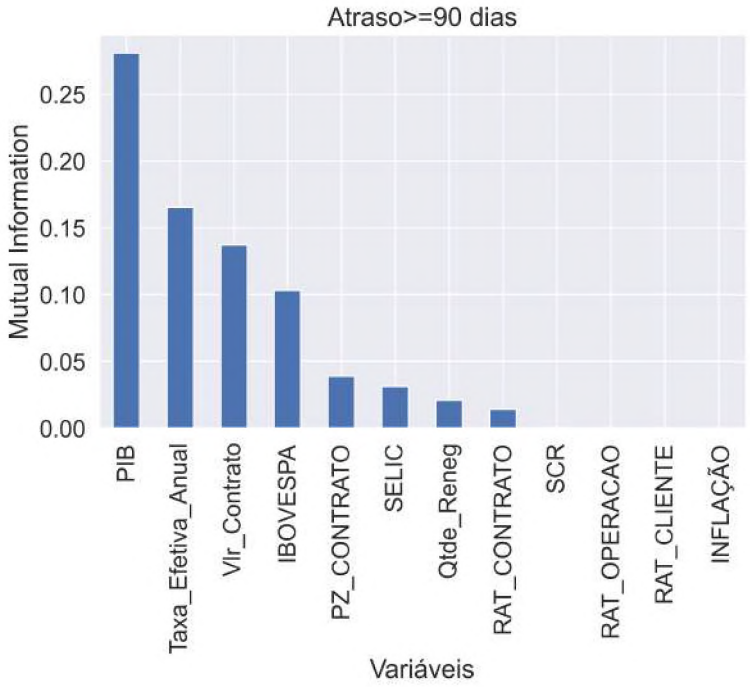
Fonte: Elaborado pela autora (2022).

Figura 50 – *MIC e SelectKBest - Subamostra 01 - Atraso* ≥ 90



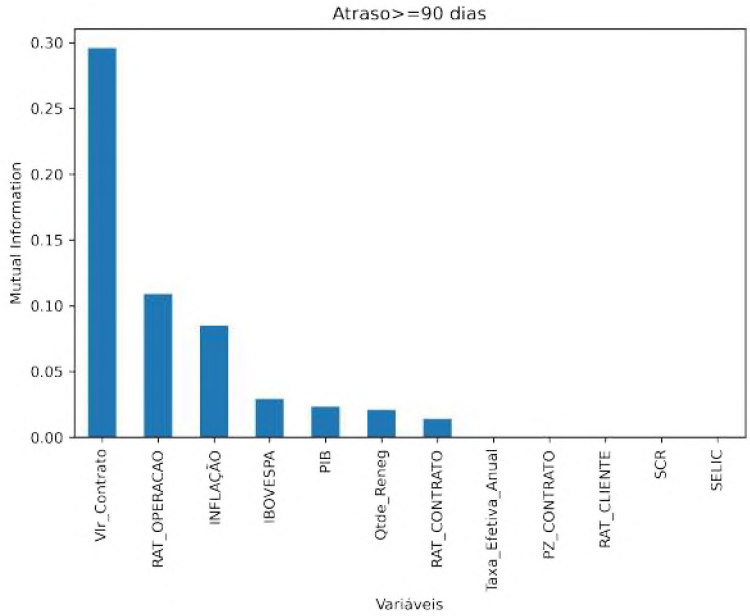
Fonte: Elaborado pela autora (2022).

Figura 51 – *MIC e SelectKBest - Subamostra 02 - Atraso* ≥ 90



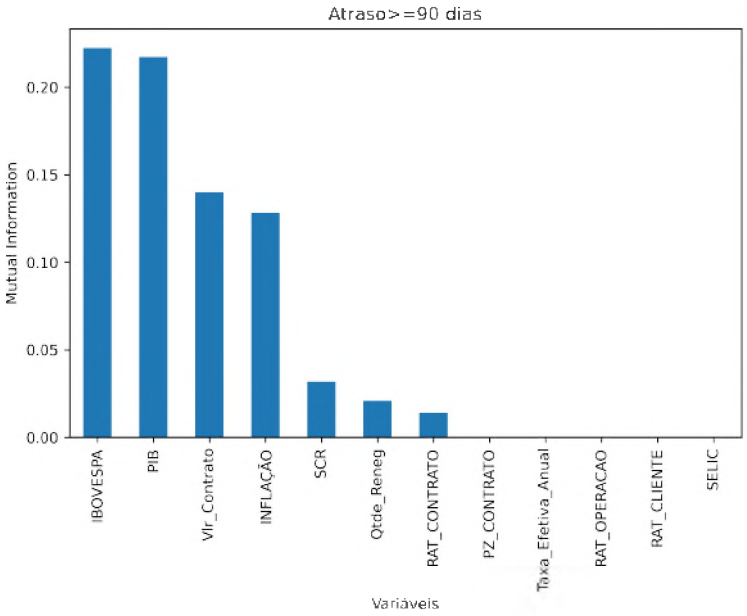
Fonte: Elaborado pela autora (2022).

Figura 52 – *MIC e SelectKBest - Subamostra 03 - Atraso* ≥ 90



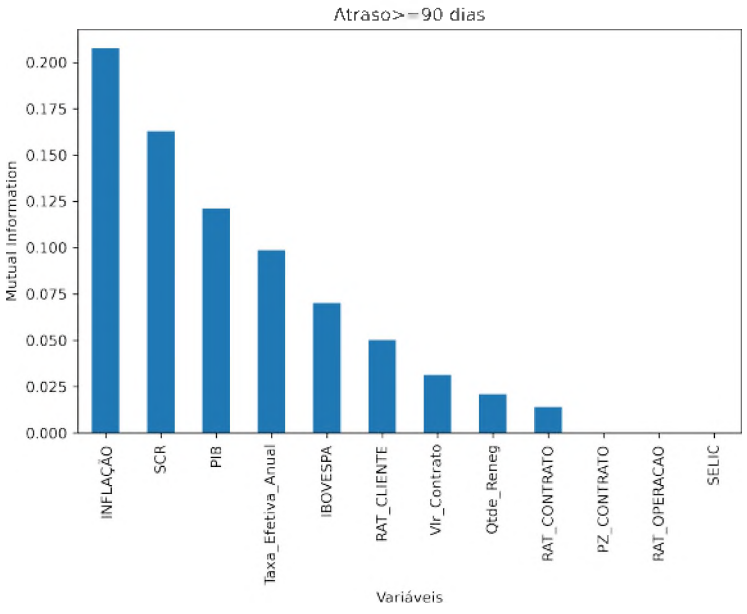
Fonte: Elaborado pela autora (2022).

Figura 53 – *MIC e SelectKBest - Subamostra 04 - Atraso* ≥ 90



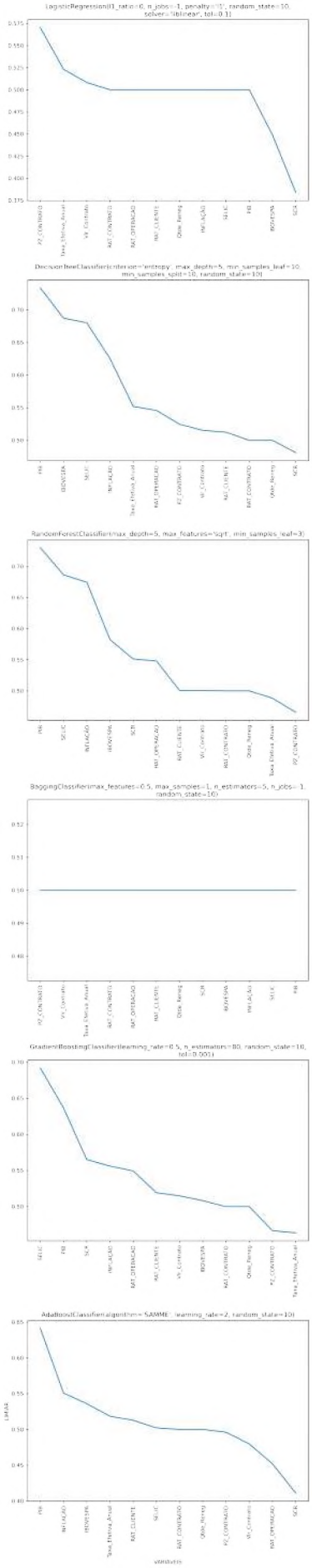
Fonte: Elaborado pela autora (2022).

Figura 54 – *MIC e SelectKBest - Subamostra 05 - Atraso* ≥ 90



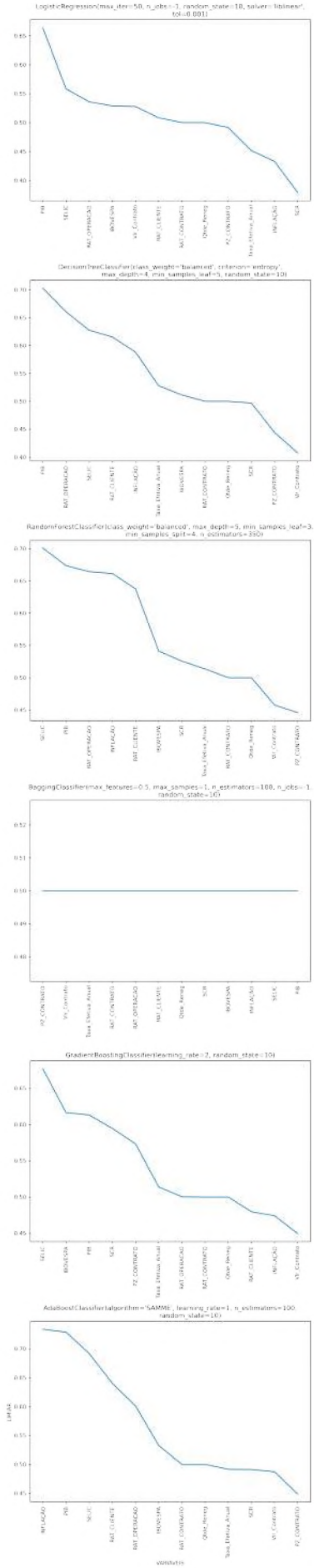
Fonte: Elaborado pela autora (2022).

Figura 55 – Usando ROC - AUC - Subamostra 01 - Atraso $\geq$ 30



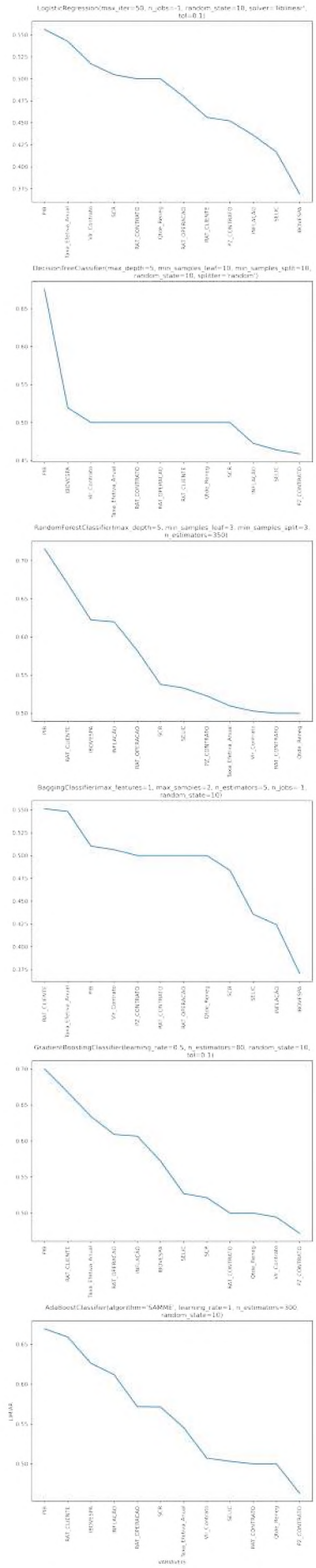
Fonte: Elaborado pela autora (2022).

Figura 56 – Usando ROC - AUC - Subamostra 02 - Atraso ≥ 30



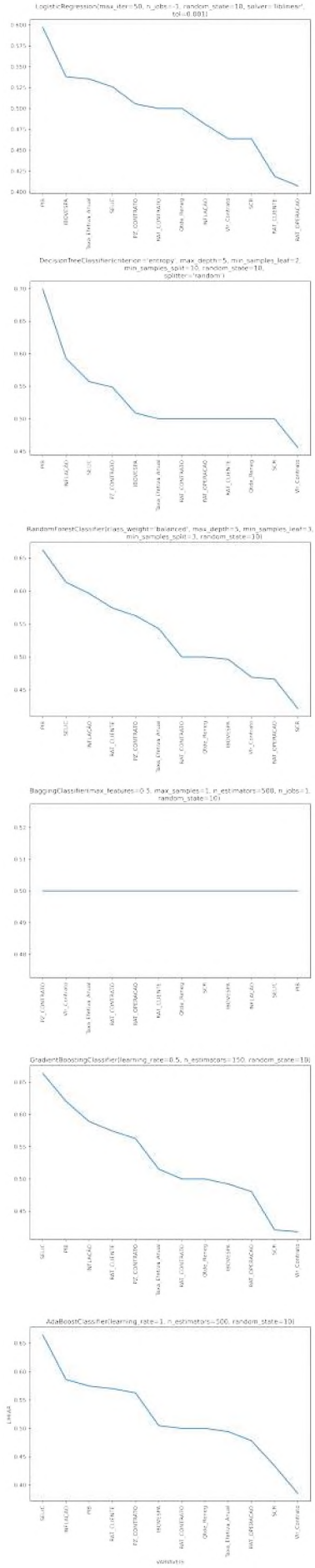
Fonte: Elaborado pela autora (2022).

Figura 57 – Usando ROC - AUC - Subamostra 03 - Atraso $\geq$ 30



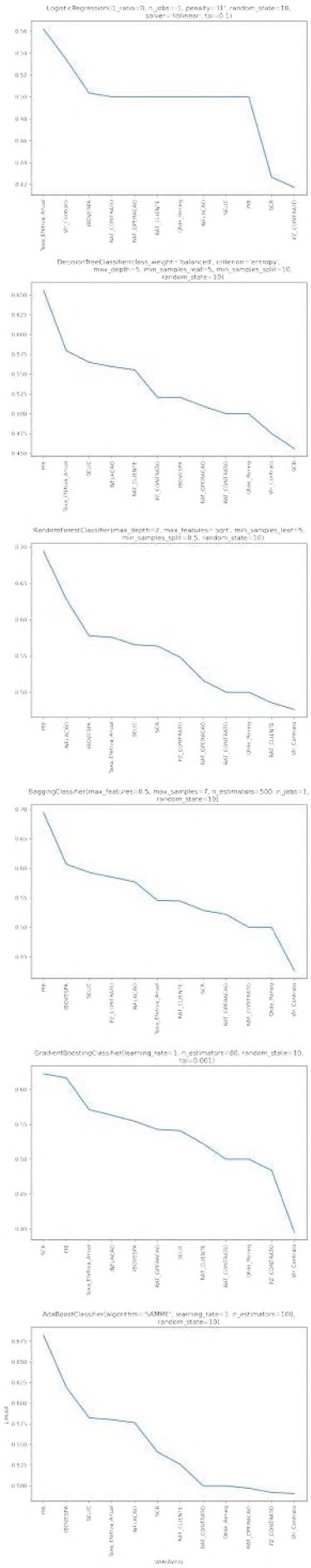
Fonte: Elaborado pela autora (2022).

Figura 58 – Usando ROC - AUC - Subamostra 04 - Atraso ≥ 30



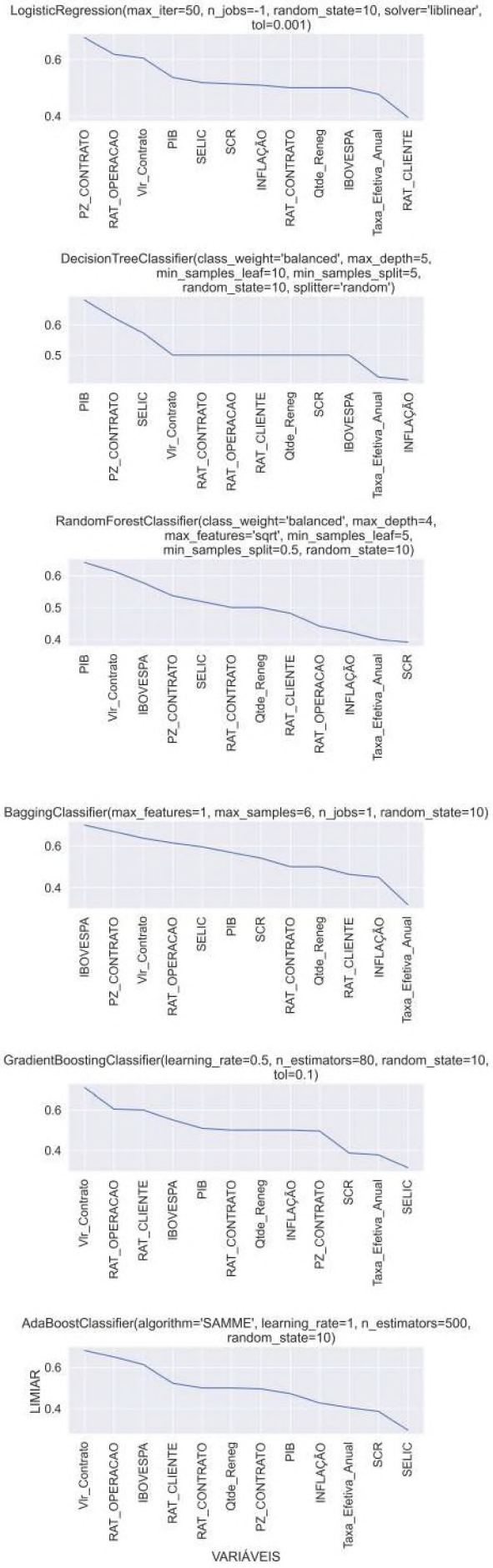
Fonte: Elaborado pela autora (2022).

Figura 59 – Usando ROC - AUC - Subamostra 05 - Atraso $\geq$ 30



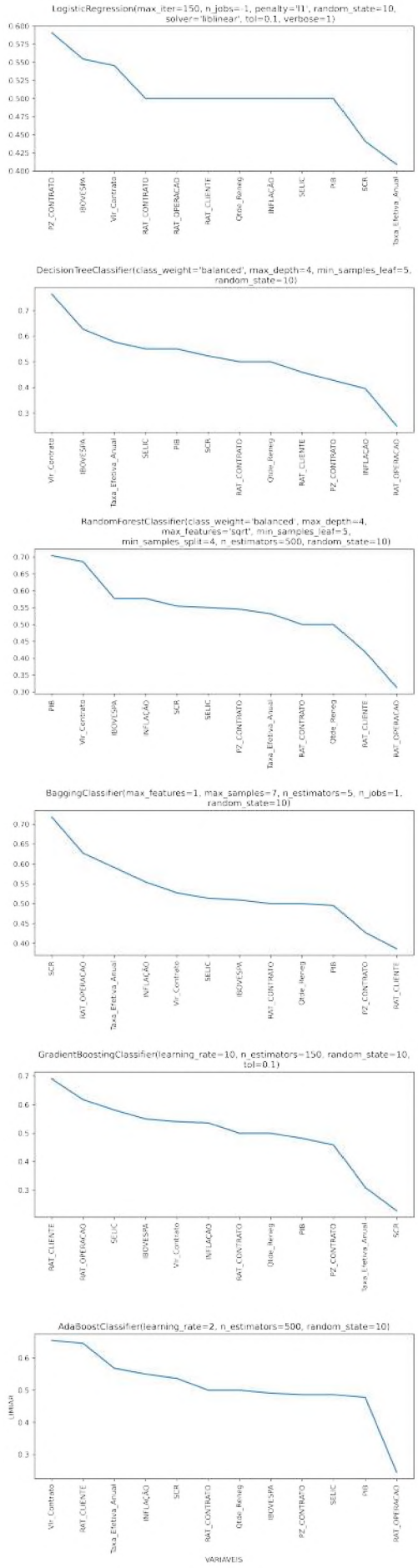
Fonte: Elaborado pela autora (2022).

Figura 60 – Usando ROC - AUC - Subamostra 01 - Atraso ≥ 60



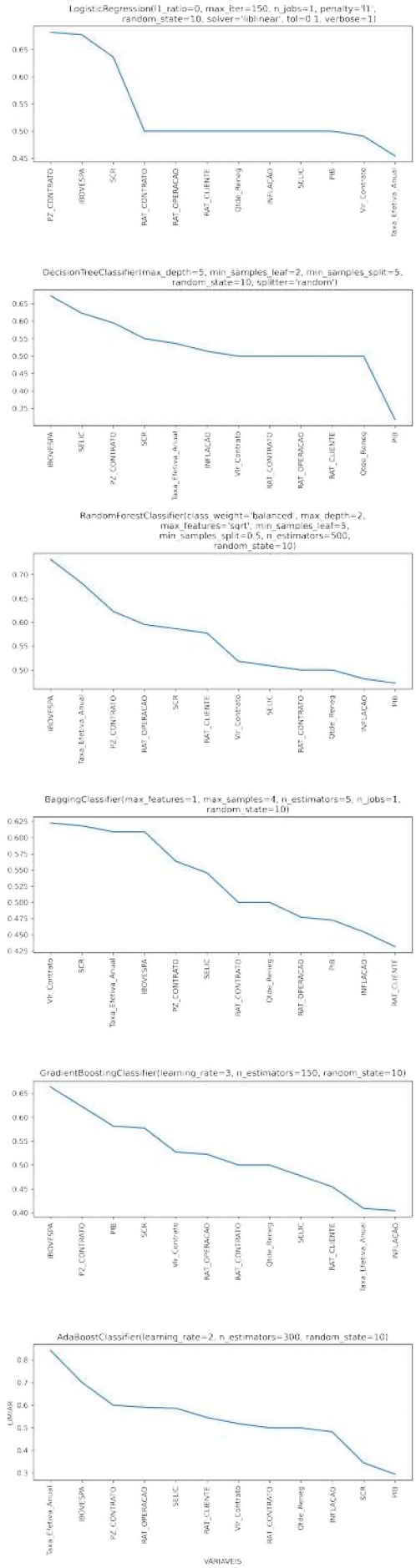
Fonte: Elaborado pela autora (2022).

Figura 61 – Usando ROC - AUC - Subamostra 02 - Atraso ≥ 60



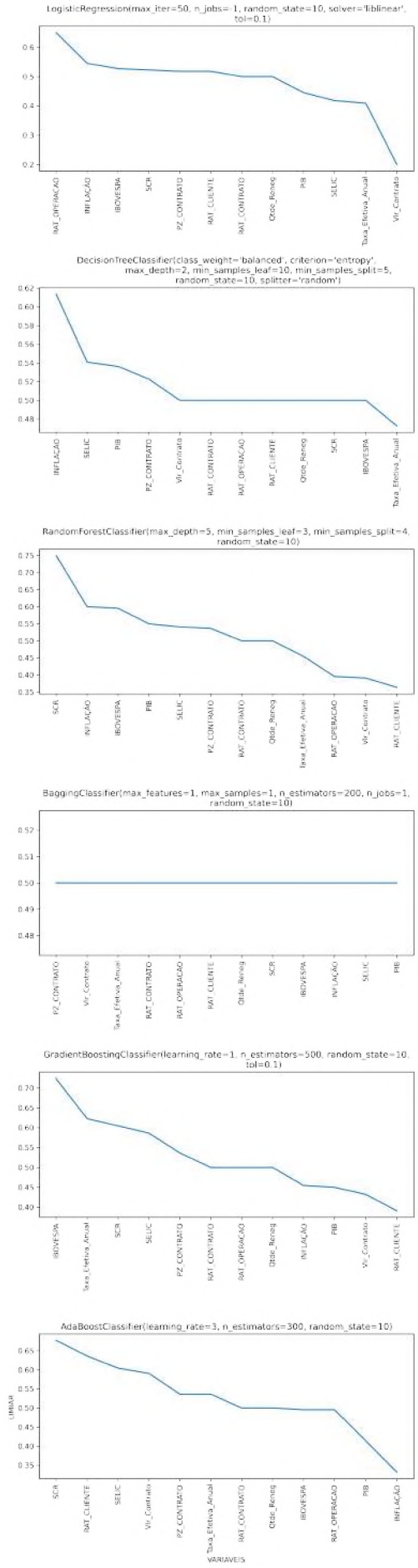
Fonte: Elaborado pela autora (2022).

Figura 62 – Usando ROC - AUC - Subamostra 03 - Atraso ≥ 60



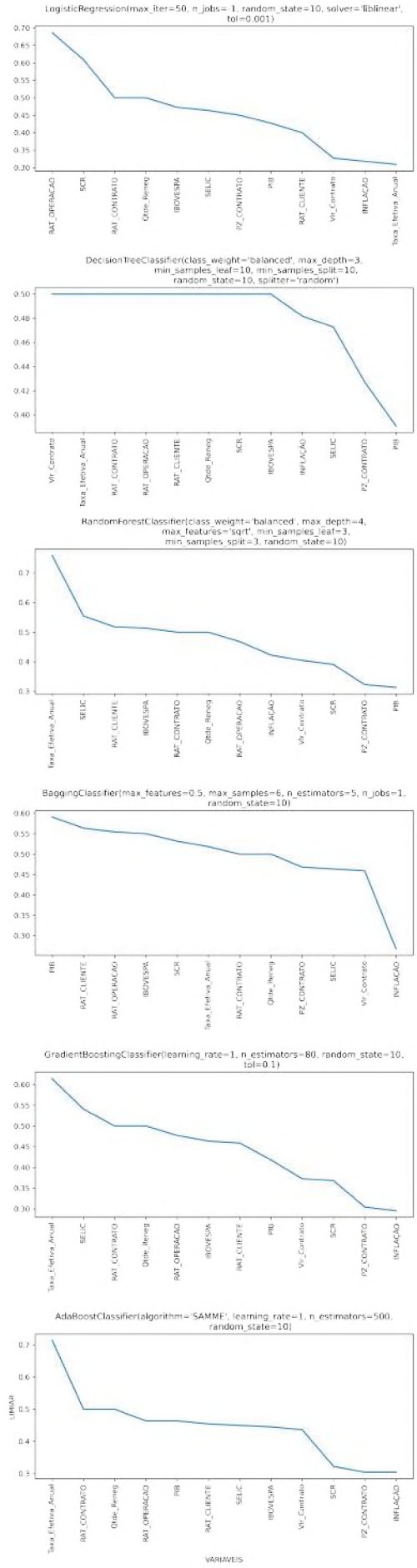
Fonte: Elaborado pela autora (2022).

Figura 63 – Usando ROC - AUC - Subamostra 04 - Atraso ≥ 60



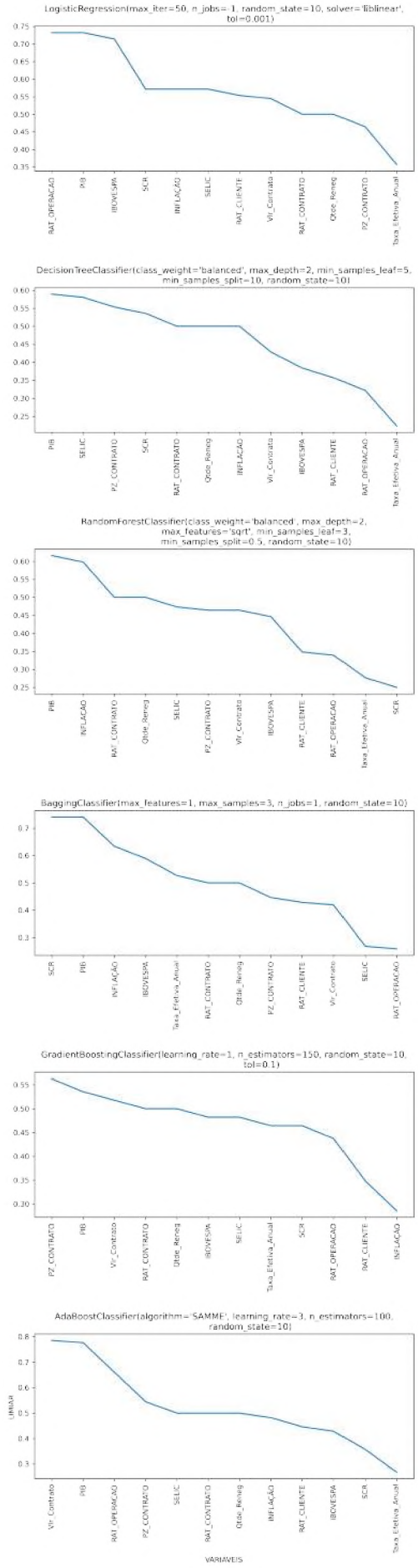
Fonte: Elaborado pela autora (2022).

Figura 64 – Usando ROC - AUC - Subamostra 05 - Atraso ≥ 60



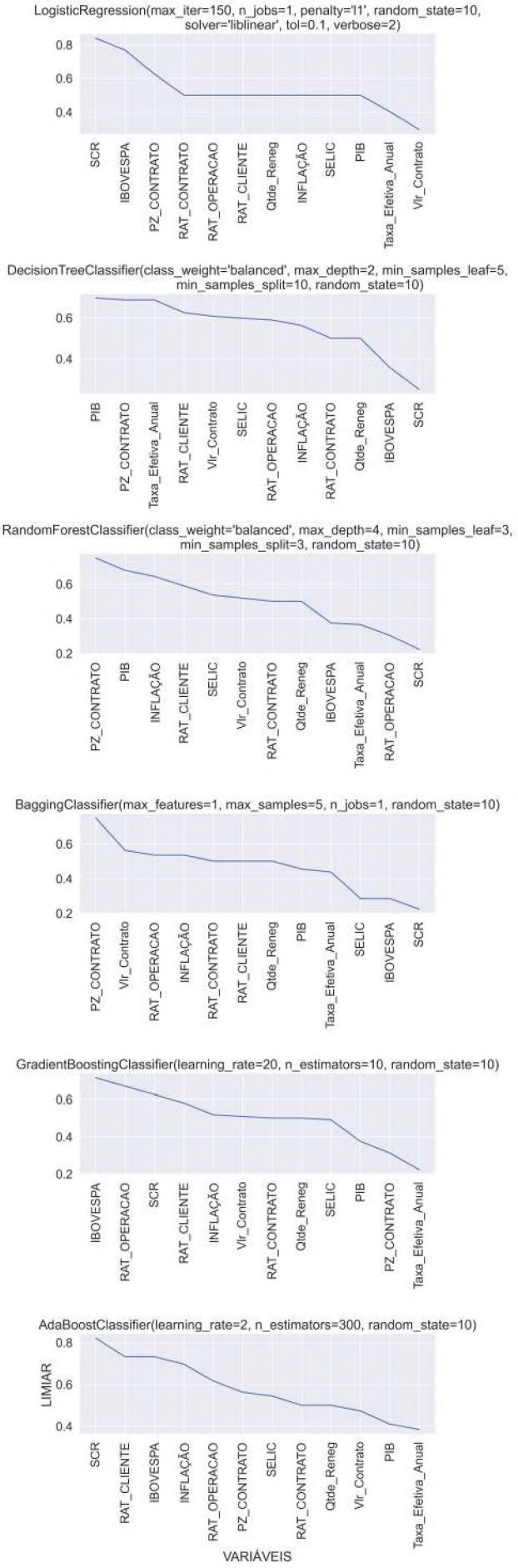
Fonte: Elaborado pela autora (2022).

Figura 65 – Usando ROC - AUC - Subamostra 01 - Atraso ≥ 90



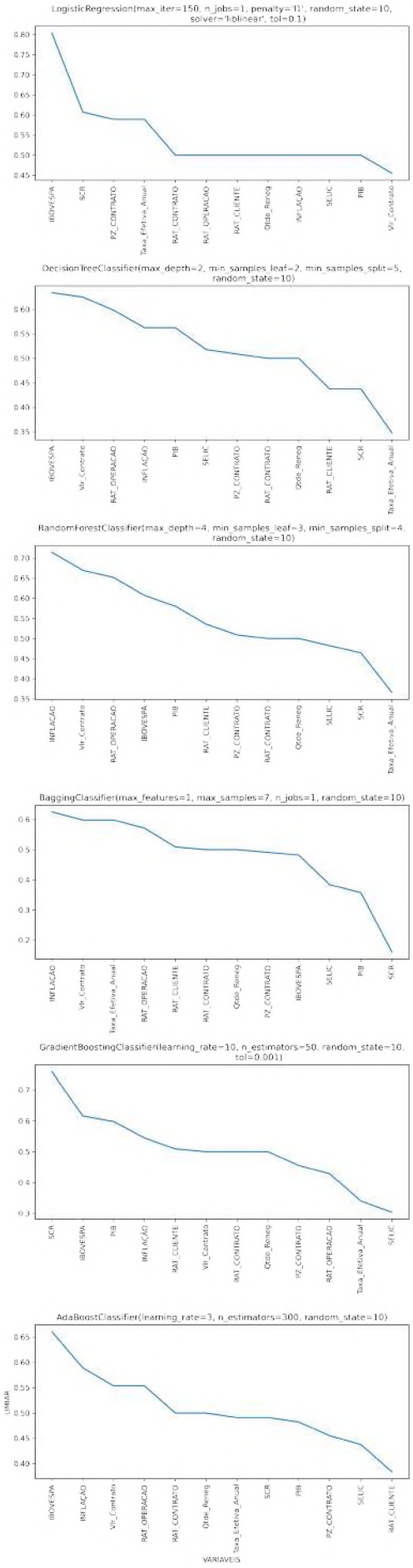
Fonte: Elaborado pela autora (2022).

Figura 66 – Usando ROC - AUC - Subamostra 02 - Atraso ≥ 90



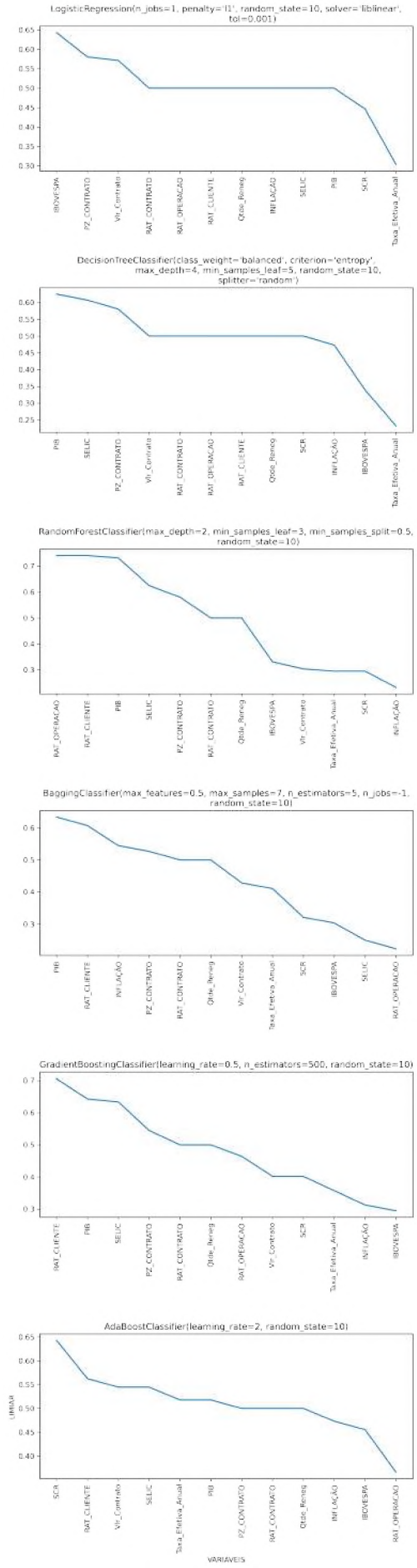
Fonte: Elaborado pela autora (2022).

Figura 67 – Usando ROC - AUC - Subamostra 03 - Atraso ≥ 90



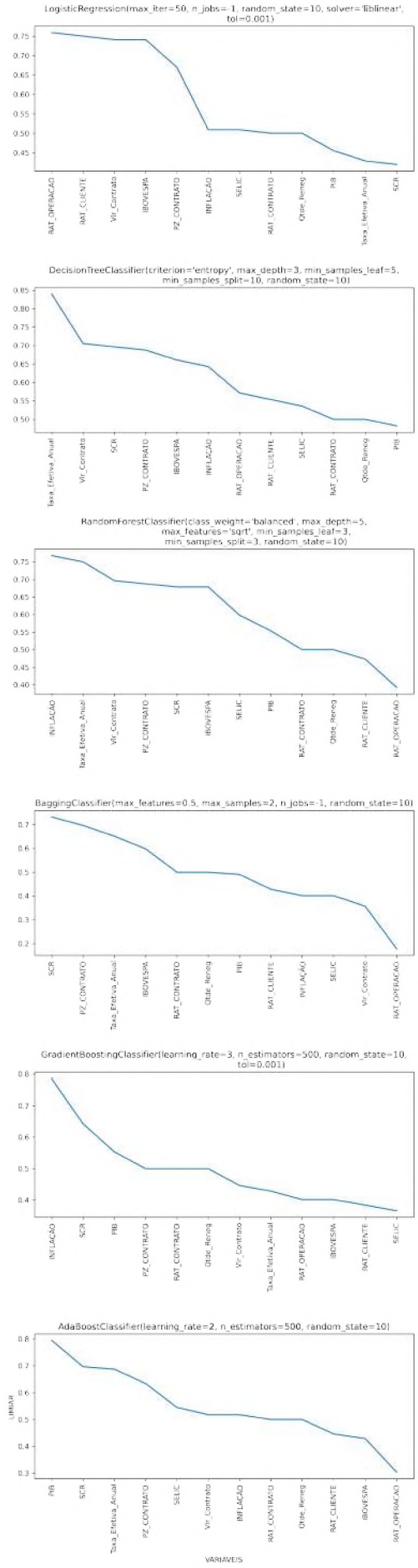
Fonte: Elaborado pela autora (2022).

Figura 68 – Usando ROC - AUC - Subamostra 04 - Atraso ≥ 90



Fonte: Elaborado pela autora (2022).

Figura 69 – Usando ROC - AUC - Subamostra 05 - Atraso ≥ 90



Fonte: Elaborado pela autora (2022).

ANEXO D – Hiper-parâmetros aplicados aos algoritmos de ML

Seguem as funções criadas para realizar validação cruzada *kflods* em cada algorítimo treinado e em cada subgrupo:

```
def SelectParlr(X_train,y_train,k):

    # k define a quantidade de kfolds na validao cruzada
    np.random.seed(10)
    cv = StratifiedKFold(n_splits = k, shuffle = True)
    #LOGIST REGRESSION

    # Criando um dicionrio com as mtricas que desejo calcular.
    meus_scores = {'balanced_accuracy' :make_scorer(balanced_accuracy_score),
                   'recall'  :make_scorer(recall_score),
                   'precision':make_scorer(precision_score),
                   'f1'      :make_scorer(fbeta_score, beta = 1)} #quanto mais
                      beta proximo de 1, melhor recall

    param_grid = {'fit_intercept': [True],
         'l1_ratio': [0,0.5,1],
         'max_iter': [50,100,150],
         'n_jobs': [-1,1],
         'tol': [0.0001,0.001,0.1],
         'verbose': [0,1,2],
         'penalty': ['l1', 'l2','none'], #panlidades suportadas pela lbfgs e
            liblinear abaixo
         'solver': ['lbfgs','liblinear'] #usados em pequenos datasets
        }

    # Models instances.
    logisticRegression = LogisticRegression()

    LogisticRegression_grid_search = fun(logisticRegression,
                                param_grid,cv=cv,
                                scoring = meus_scores,
                                return_train_score=True,
```

```
                                verbose=True, #enquanto treina escreve na tela
                                    o que esta fazendo, quanto maior o numero
                                    mais informacao aparece na tela
                                n_jobs=-1,
                                refit = 'f1')
    LogisticRegression_grid_search.fit(X_train, y_train)
    print(LogisticRegression_grid_search.best_params_)
    print(LogisticRegression_grid_search.best_estimator_)
    print(LogisticRegression_grid_search.best_score_)

def SelectPardt(X_train,y_train,k):

    # k define a quantidade de kfolds na validao cruzada
    np.random.seed(10)
    cv = StratifiedKFold(n_splits = k, shuffle = True)
    #DECISION TREE
    #Para  classificao  com poucas classes, min_samples_leaf=1  muitas vezes a
        melhor escolha.
    #Equilibre o seu conjunto de dados antes do treino para evitar que a rvore
        seja enviesada para as classes que so dominantes.

    # Criando um dicionrio com as mtricas que desejo calcular.
    meus_scores = {'balanced_accuracy' :make_scorer(balanced_accuracy_score),
                   'recall'  :make_scorer(recall_score),
                   'precision':make_scorer(precision_score),
                   'f1'      :make_scorer(fbeta_score, beta = 1)} #quanto mais
                       beta proximo de 1, melhor recall

    param_grid = {'splitter': ['best', 'random'],
        'criterion': ['gini', 'entropy'], #formula matematica que melhor
            separa as duas classes 0 e 1, cada uma com uma curva diferente
        'min_samples_leaf': [2,5,10],
        'max_depth': [2,3,4,5],
        'min_samples_split': [2,5,10]
        #'class_weight': [None, 'balanced'] #traz mais peso para a base menos
            balanceada, tentando balancear o peso das arvores
       }

    # Models instances.
    decisionTreeClassifier = DecisionTreeClassifier()

    DecisionTreeClassifier_grid_search = GridSearchCV(decisionTreeClassifier,
```

```
                                param_grid,cv=cv,
                                scoring = meus_scores,
                                return_train_score=True,
                                verbose=True, #enquanto treina escreve na tela
                                    o que esta fazendo, quanto maior o numero
                                    mais informacao aparece na tela
                                n_jobs=-1,
                                refit = 'f1')
    print (DecisionTreeClassifier_grid_search.fit(X_train, y_train))
    print (DecisionTreeClassifier_grid_search.best_params_)
    print (DecisionTreeClassifier_grid_search.best_estimator_)
    print (DecisionTreeClassifier_grid_search.best_score_)

def SelectParrf(X_train,y_train,k):

    # k define a quantidade de kfolds na validao cruzada
    np.random.seed(10)
    cv = StratifiedKFold(n_splits = k, shuffle = True)
    #RANDOM FOREST


    # Criando um dicionrio com as mtricas que desejo calcular.
    meus_scores = {'balanced_accuracy' :make_scorer(balanced_accuracy_score),
                   'recall'  :make_scorer(recall_score),
                   'precision':make_scorer(precision_score),
                   'f1'      :make_scorer(fbeta_score, beta = 1)} #quanto mais
                       beta proximo de 1, melhor recall

    param_grid ={'bootstrap': [True],
        'max_features': ['auto', 'sqrt'],
        'min_samples_leaf': [3,5],
        'n_estimators': [100,350,500],
        'max_depth': [2,3,4,5],
        'min_samples_split': [0.5,2,3,4]
        #'class_weight': [None, 'balanced'] #traz mais peso para a base menos
            balanceada, tentando balancear o peso das arvores
      }

    # Models instances.
    randomForestClassifier = RandomForestClassifier()

    RandomForestClassifier_grid_search = GridSearchCV(randomForestClassifier,
```

```
                                param_grid,cv=cv,
                                scoring = meus_scores,
                                return_train_score=True,
                                verbose=True, #enquanto treina escreve na tela
                                    o que esta fazendo, quanto maior o numero
                                    mais informacao aparece na tela
                                n_jobs=-1,
                                refit = 'f1')

    print (RandomForestClassifier_grid_search.fit(X_train, y_train))
    print (RandomForestClassifier_grid_search.best_params_)
    print (RandomForestClassifier_grid_search.best_estimator_)
    print (RandomForestClassifier_grid_search.best_score_)

def SelectParbc(X_train,y_train,k):

    # k define a quantidade de kfolds na validao cruzada
    np.random.seed(10)
    cv = StratifiedKFold(n_splits = k, shuffle = True)
    #BAGGING CLASSIFIER
    #https://scikit-learn.org/stable/modules/ensemble.html#bagging
    #https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingClassifier.html
    #Quando as amostras so desenhadas com substituio, ento o mtodo
        conhecido como Bagging
    #m particular, max_samples e max_features controlam o tamanho dos
        subconjuntos
    #em termos de amostras e caractersticas), enquanto que bootstrap e
        bootstrap_features
    #controlam se as amostras e caractersticas so desenhadas com ou sem
        substituio.

    # Criando um dicionrio com as mtricas que desejo calcular.
    meus_scores = {'balanced_accuracy' :make_scorer(balanced_accuracy_score),
                   'recall'  :make_scorer(recall_score),
                   'precision':make_scorer(precision_score),
                   'f1'      :make_scorer(fbeta_score, beta = 1)} #quanto mais
                       beta proximo de 1, melhor recall

    param_grid = {'bootstrap': [True],
        'bootstrap_features': [False],
        'n_jobs': [-1,1],
        'max_samples': [0,1,2,3,4,5,6,7],
```

```
        'max_features': [0.5,1,2],
        'n_estimators': [5,10,50,100,200,500]
       }

   # Models instances.
    baggingClassifier = BaggingClassifier()

    BaggingClassifier_grid_search = GridSearchCV(baggingClassifier,
                                param_grid,cv=cv,
                                scoring = meus_scores,
                                return_train_score=True,
                                verbose=True, #enquanto treina escreve na tela
                                    o que esta fazendo, quanto maior o numero
                                    mais informacao aparece na tela
                                n_jobs=-1,
                                refit = 'f1')

    print (BaggingClassifier_grid_search.fit(X_train, y_train))
    print (BaggingClassifier_grid_search.best_params_)
    print (BaggingClassifier_grid_search.best_estimator_)
    print (BaggingClassifier_grid_search.best_score_)

def SelectPargb(X_train,y_train,k):

    # k define a quantidade de kfolds na validao cruzada
    np.random.seed(10)
    cv = StratifiedKFold(n_splits = k, shuffle = True)
    #GRADIENT BOOSTING
    #Aspectos importantes para considerar:
    #  nmero  de alunos fracos (numero de arvos)  controlado pelo parmetro
        n_estimadores.
    #The 2 most important parameters of these estimators are n_estimators and
        learning_rate.

    # Criando um dicionrio com as mtricas que desejo calcular.
    meus_scores = {'balanced_accuracy' :make_scorer(balanced_accuracy_score),
                   'recall'  :make_scorer(recall_score),
                   'precision':make_scorer(precision_score),
                   'f1'      :make_scorer(fbeta_score, beta = 1)} #quanto mais
                       beta proximo de 1, melhor recall

    param_grid = {'tol': [0.0001,0.001,0.1],
```

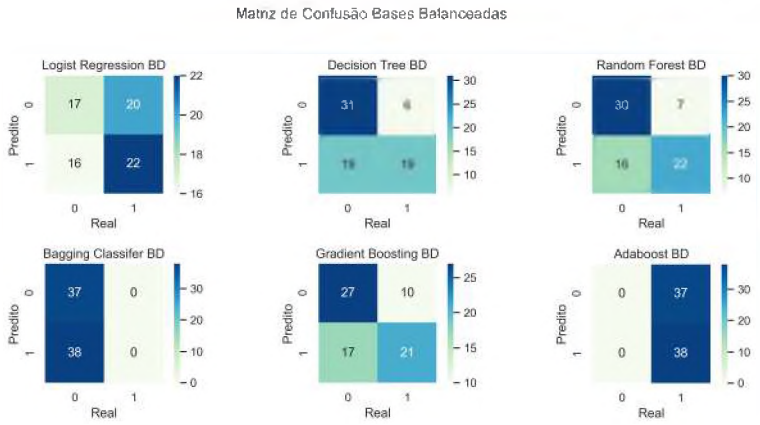
```
        #'max_depth': [1,2,3,4,5,6,7,8,9,10], #relacionados a decision tree
        #'min_samples_split':[1,2,3,4,5,6],  #relacionados a decision tree
        'learning_rate': [0.5,1,2,3,10,20], #controla o overfitting entre [0 e
            1]
        'n_estimators': [80,100,150,300,500]
       }

   # Models instances.
    gradientBoostingClassifier = GradientBoostingClassifier()

    GradientBoostingClassifier_grid_search =
        GridSearchCV(gradientBoostingClassifier,
                                param_grid,cv=cv,
                                scoring = meus_scores,
                                return_train_score=True,
                                verbose=True, #enquanto treina escreve na tela
                                    o que esta fazendo, quanto maior o numero
                                    mais informacao aparece na tela
                                n_jobs=-1,
                                refit = 'f1')

    print (GradientBoostingClassifier_grid_search.fit(X_train, y_train))
    print (GradientBoostingClassifier_grid_search.best_params_)
    print (GradientBoostingClassifier_grid_search.best_estimator_)
    print (GradientBoostingClassifier_grid_search.best_score_)

def SelectParab(X_train,y_train,k):

    # k define a quantidade de kfolds na validao cruzada
    np.random.seed(10)
    cv = StratifiedKFold(n_splits = k, shuffle = True)
    #ADABOOST
    #Aspectos que mais podem impactar:
    #  nmero  de alunos fracos (numero de arvos)  controlado pelo parmetro
        n_estimadores.
    # O parmetro learning_rate controla a contribuio dos aprendentes fracos
        na combinao final.
    #Os principais parmetros a afinar para obter bons resultados so os
        n_estimadores
    #e a complexidade dos estimadores de base (por exemplo, a sua profundidade
        mxima_profundidade
    #ou o nmero  mnimo  exigido de amostras para considerar uma diviso de
```

```
    min_amostragens).


# Criando um dicionrio com as mtricas que desejo calcular.
meus_scores = {'balanced_accuracy' :make_scorer(balanced_accuracy_score),
               'recall'  :make_scorer(recall_score),
               'precision':make_scorer(precision_score),
               'f1'      :make_scorer(fbeta_score, beta = 1)} #quanto mais
                  beta proximo de 1, melhor recall

param_grid = {'algorithm': ['SAMME','SAMME.R'],
    #'max_depth': [1,2,3,4,5,6,7,8,9,10],    #relacionados a decision tree
    #'min_samples_split':[1,2,3,4,5,6],     #relacionados a decision tree
    'learning_rate': [1,2,3], #controla o overfitting entre [0 e 1]
    'n_estimators': [50,100,300,500]
   }

# Models instances.
adaBoostClassifier = AdaBoostClassifier()

AdaBoostClassifier_grid_search = GridSearchCV(adaBoostClassifier,
                            param_grid,cv=cv,
                            scoring = meus_scores,
                            return_train_score=True,
                            verbose=True, #enquanto treina escreve na tela
                                o que esta fazendo, quanto maior o numero
                                mais informacao aparece na tela
                            n_jobs=-1,
                            refit = 'f1')

print (AdaBoostClassifier_grid_search.fit(X_train, y_train))
print (AdaBoostClassifier_grid_search.best_params_)
print (AdaBoostClassifier_grid_search.best_estimator_)
print (AdaBoostClassifier_grid_search.best_score_)
```

ANEXO E – Gráficos adicionais - Aplicando todas as variáveis disponíveis

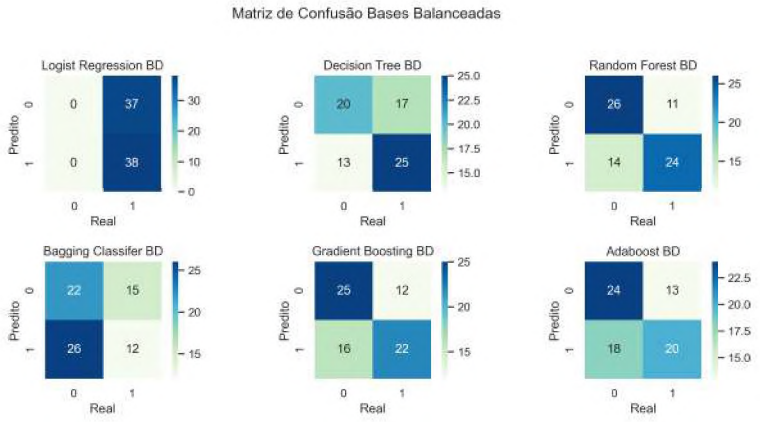
As matrizes de confusão resultantes da comparação entre os algoritmos dentro da Subamostra 01, 03, 04 e 05 no Subgrupo Atraso $\geq$ 30 - TV podem ser consultadas nas Figuras 70, 71, 72 e 73.

Figura 70 – Matriz de Confusão - Subamostra 01 - Atraso $\geq$ 30 - TV



Fonte: (PYTHON..., 2017).

Figura 71 – Matriz de Confusão - Subamostra 03 - Atraso $\geq$ 30 - TV

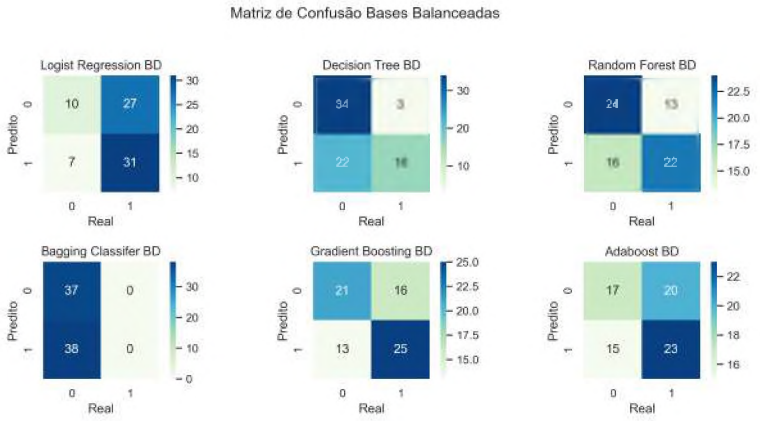


Fonte: (PYTHON..., 2017).

As matrizes de confusão resultantes da comparação entre os algoritmos dentro da Subamostra 02, 03, 04 e 05 no Subgrupo Atraso $\geq$ 60 - TV podem ser consultadas nas Figuras 74, 75, 76 e 77.

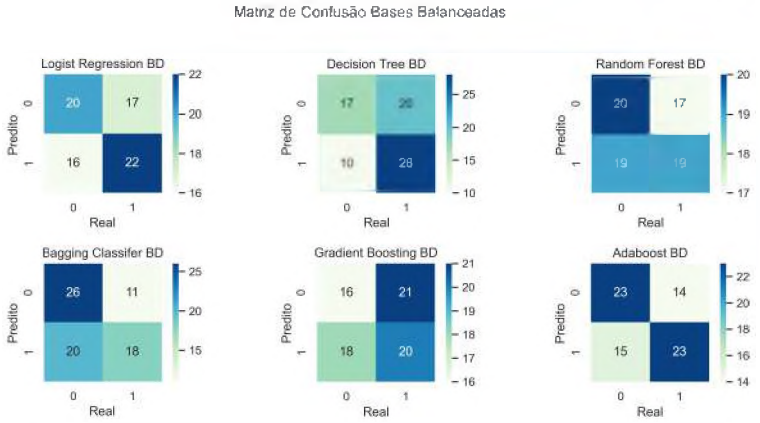
A decisão sobre a escolha do algoritmo com melhor desempenho também considerou a métrica AUC em cada Subamostra, conforme subgrupo, ver Figuras 78, 79, 80, 81, 82.

Figura 72 – Matriz de Confusão - Subamostra 04 - Atraso ≥ 30 - TV



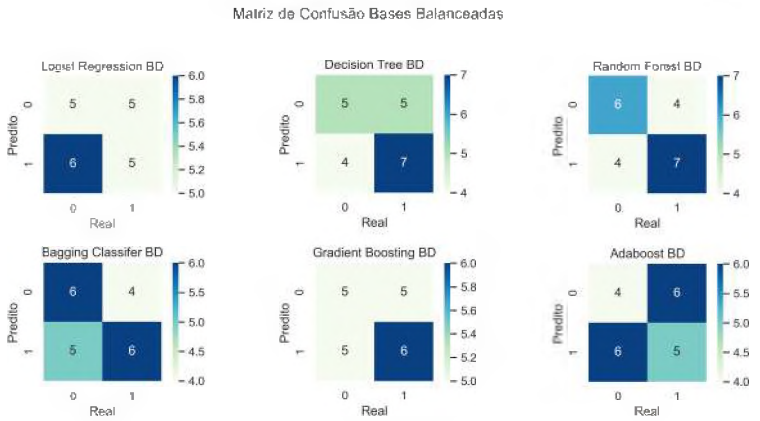
Fonte: (PYTHON..., 2017).

Figura 73 – Matriz de Confusão - Subamostra 05 - Atraso ≥ 30 - TV



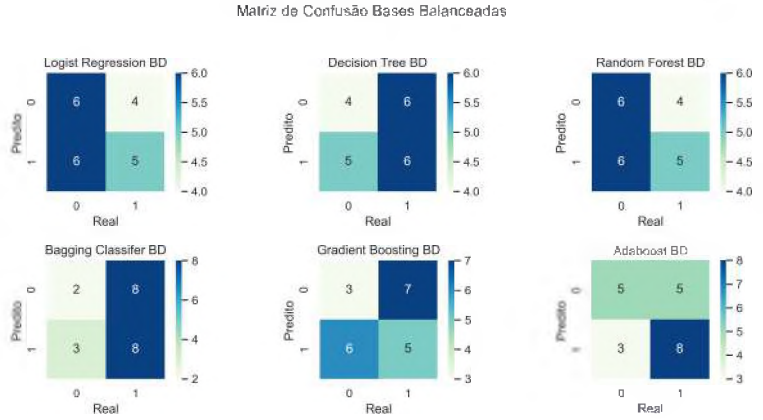
Fonte: (PYTHON..., 2017).

Figura 74 – Matriz de Confusão - Subamostra 02 - Atraso ≥ 60 - TV



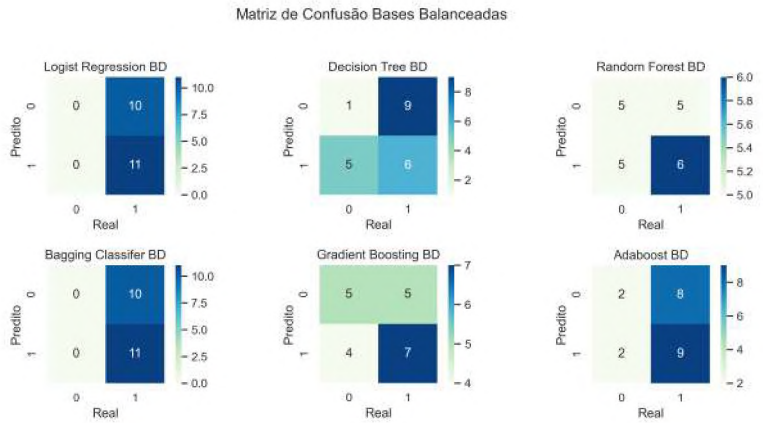
Fonte: (PYTHON..., 2017).

Figura 75 – Matriz de Confusão - Subamostra 03 - Atraso ≥ 60 - TV



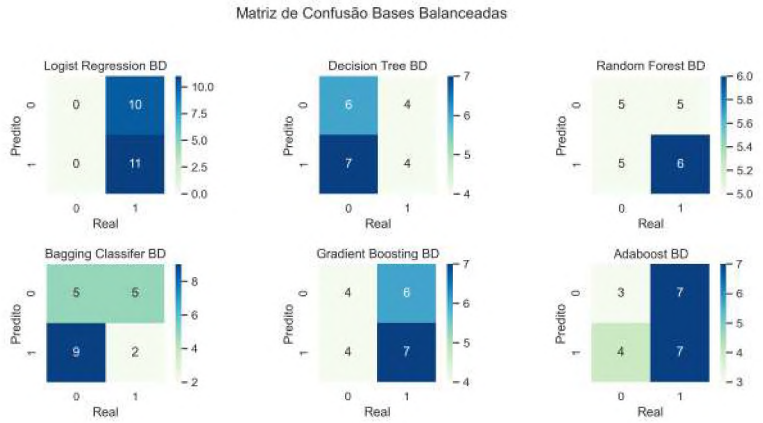
Fonte: (PYTHON..., 2017).

Figura 76 – Matriz de Confusão - Subamostra 04 - Atraso ≥ 60 - TV



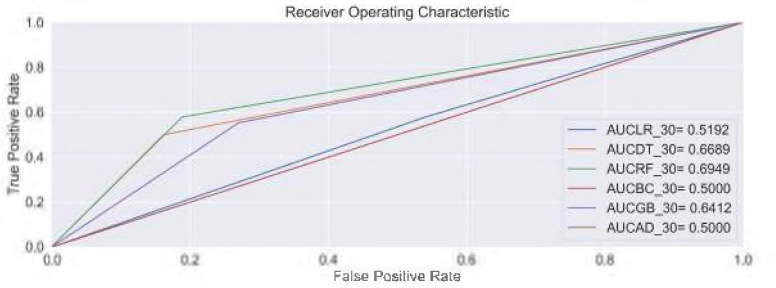
Fonte: (PYTHON..., 2017).

Figura 77 – Matriz de Confusão - Subamostra 05 - Atraso ≥ 60 - TV



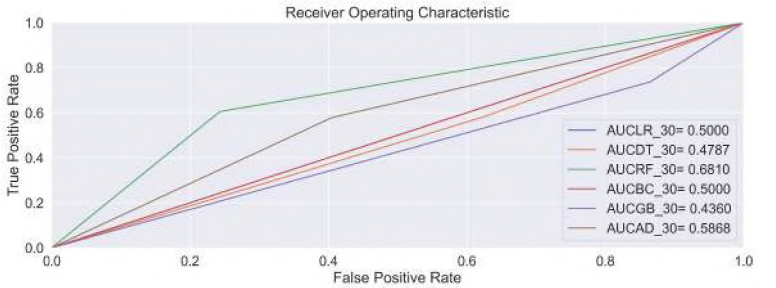
Fonte: (PYTHON..., 2017).

Figura 78 – AUC - Subamostra 01 - Atraso ≥ 30 - TV



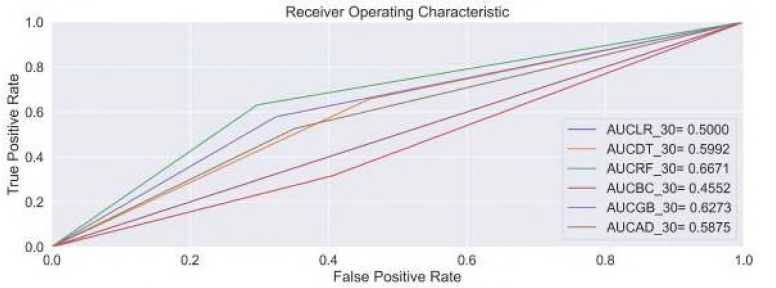
Fonte: Elaborada pela autora(2022).

Figura 79 – AUC - Subamostra 02 - Atraso ≥ 30 - TV



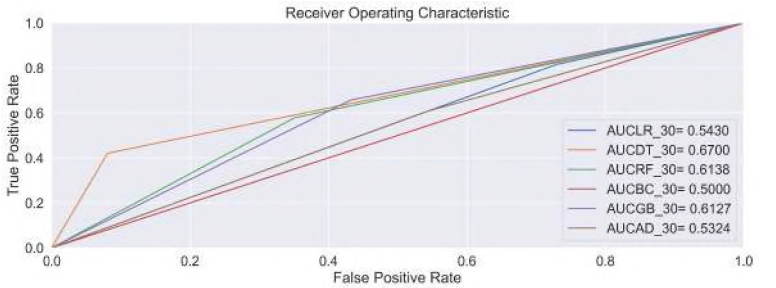
Fonte: Elaborada pela autora(2022).

Figura 80 – AUC - Subamostra 03 - Atraso ≥ 30 - TV



Fonte: Elaborada pela autora(2022).

Figura 81 – AUC - Subamostra 04 - Atraso ≥ 30 - TV

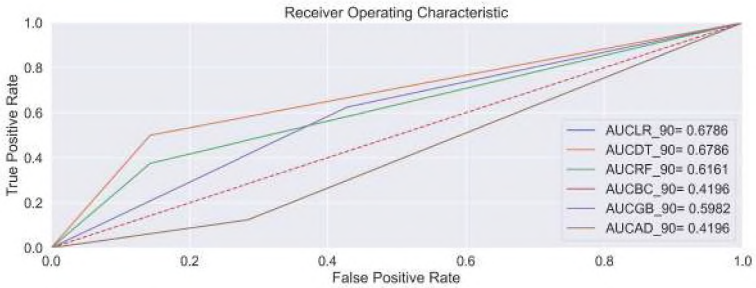


Fonte: Elaborada pela autora(2022).

Figura 82 – AUC - Subamostra 05 - Atraso ≥ 30 - TV

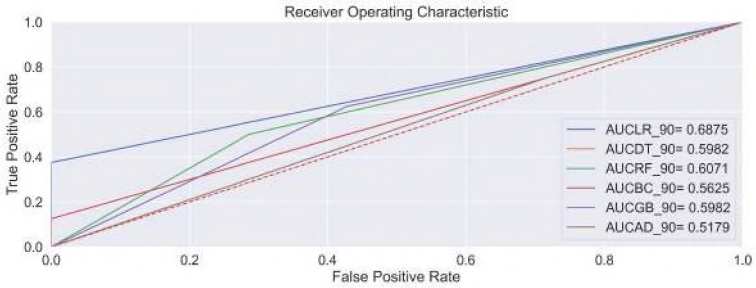
Fonte: Elaborada pela autora(2022).

Figura 83 – AUC - Subamostra 02 - Atraso ≥ 90 - TV



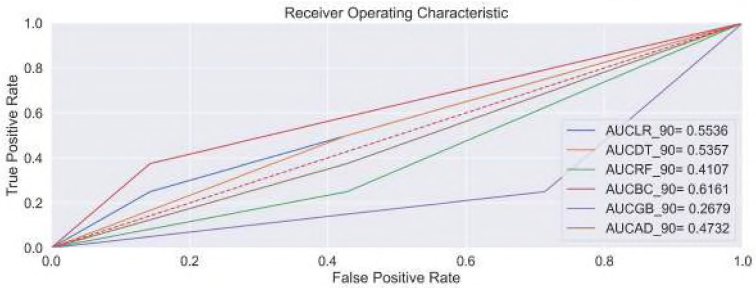
Fonte: Elaborada pela autora(2022).

Figura 84 – AUC - Subamostra 03 - Atraso ≥ 90 - TV



Fonte: Elaborada pela autora(2022).

Figura 85 – AUC - Subamostra 04 - Atraso ≥ 90 - TV

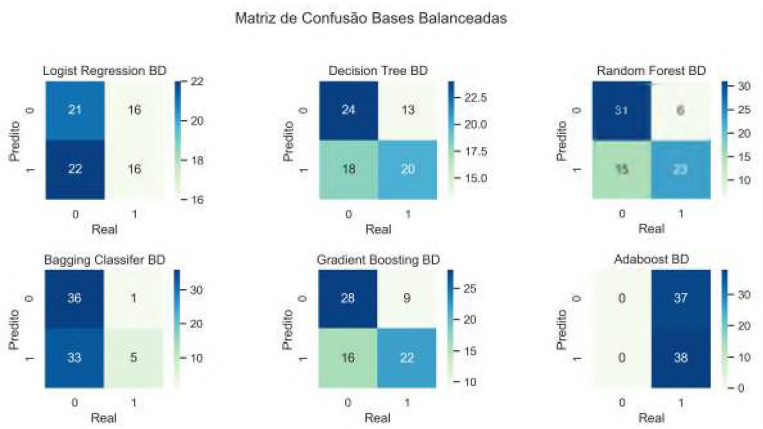


Fonte: Elaborada pela autora(2022).

ANEXO F – Gráficos adicionais - Aplicação das técnicas de seleção de variáveis

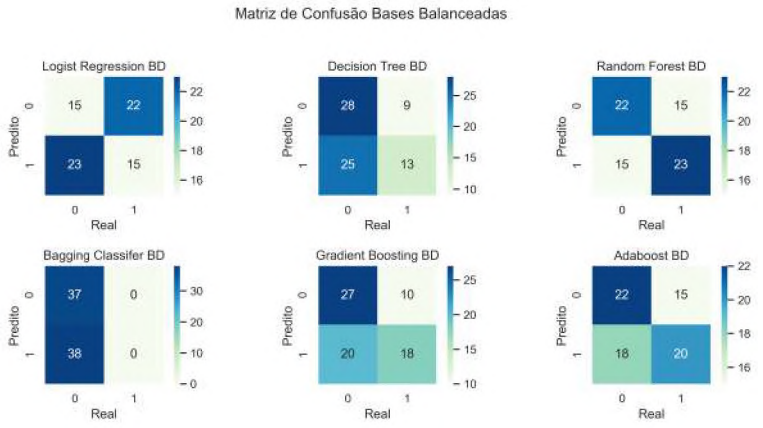
As matrizes de confusão resultantes da comparação entre os algoritmos dentro da Subamostra 01, 03, 04 e 05 no Subgrupo Atraso $\geq$ 30 - TV podem ser consultadas nas Figuras 86, 87, 88 e 89.

Figura 86 – Matriz de Confusão - Subamostra 01 - Atraso $\geq$ 30 - STV



Fonte: Elaborada pela autora(2022).

Figura 87 – Matriz de Confusão - Subamostra 03 - Atraso $\geq$ 30 - STV

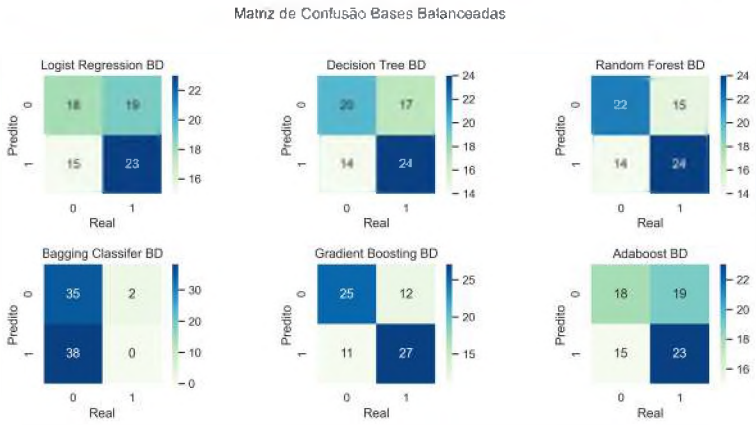


Fonte: Elaborada pela autora(2022).

A decisão sobre a escolha do algoritmo com melhor desempenho também considerou a métrica AUC em cada Subamostra, no subgrupo Atraso $\geq$ 30, ver Figuras 90, 91, 92 e 93.

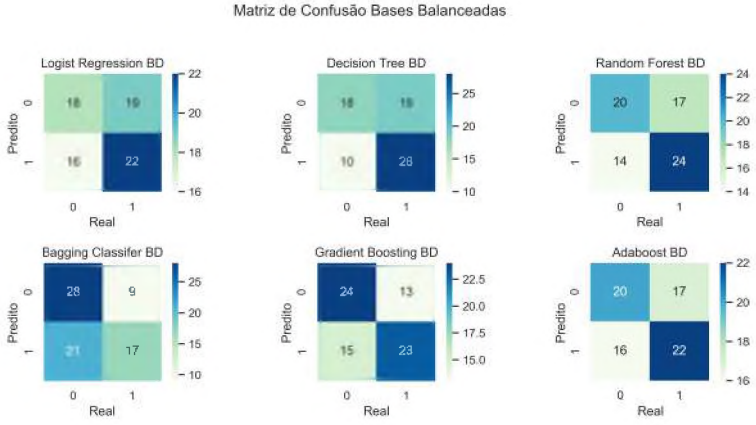
As matrizes de confusão resultantes da comparação entre os algoritmos dentro da Subamostra 02, 03, 04 e 05 no Subgrupo Atraso $\geq$ 60 - STV podem ser consultadas nas Figuras 94, 95, 96 e 97.

Figura 88 – Matriz de Confusão - Subamostra 04 - Atraso ≥ 30 - STV



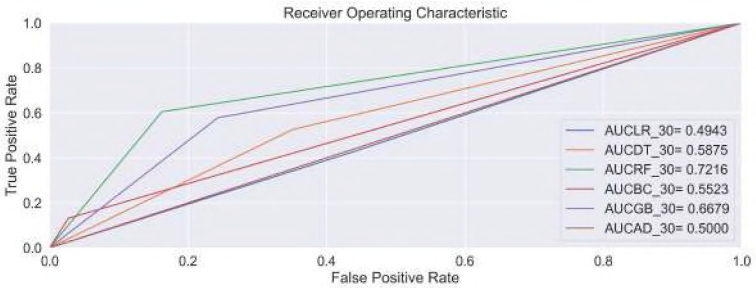
Fonte: Elaborada pela autora(2022).

Figura 89 – Matriz de Confusão - Subamostra 05 - Atraso ≥ 30 - STV



Fonte: Elaborada pela autora(2022).

Figura 90 – AUC - Subamostra 01 - Atraso ≥ 30 - STV

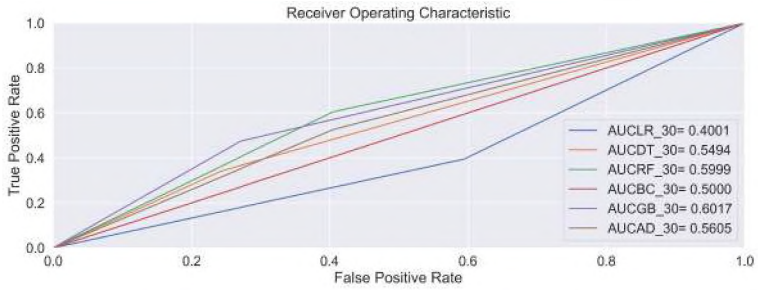


Fonte: Elaborada pela autora(2022).

A decisão sobre a escolha do algoritmo com melhor desempenho também considerou a métrica AUC em cada Subamostra, dentro do subgrupo Atraso ≥ 60, ver Figuras 98, 99, 100 e 101.

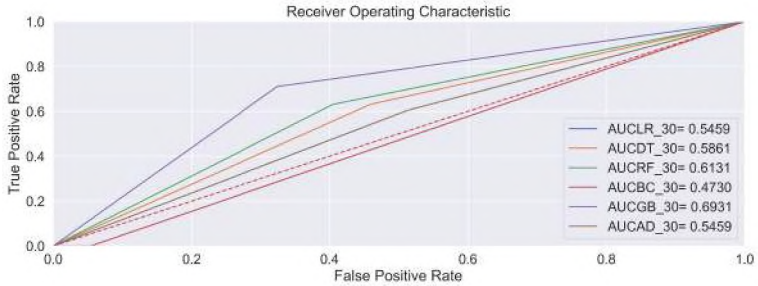
As matrizes de confusão resultantes da comparação entre os algoritmos dentro da Subamostra 01, 03 e 04, no Subgrupo Atraso ≥ 90 - STV, podem ser consultadas nas Figuras 102, 103 e 104.

Figura 91 – AUC - Subamostra 03 - Atraso ≥ 30 - STV



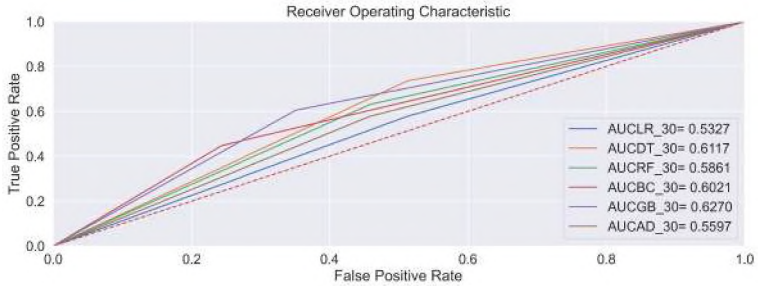
Fonte: Elaborada pela autora(2022).

Figura 92 – AUC - Subamostra 04 - Atraso ≥ 30 - STV



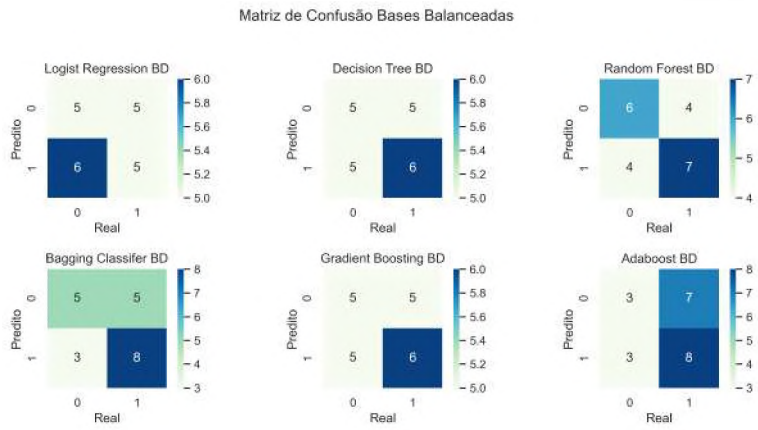
Fonte: Elaborada pela autora(2022).

Figura 93 – AUC - Subamostra 05 - Atraso ≥ 30 - STV



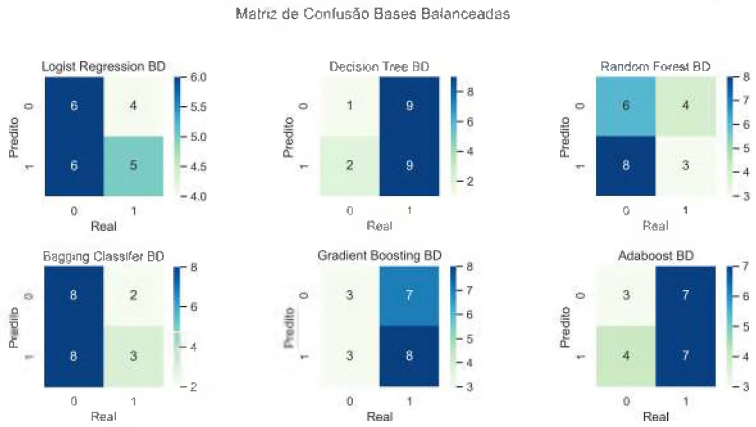
Fonte: Elaborada pela autora(2022).

Figura 94 – Matriz de Confusão - Subamostra 02 - Atraso ≥ 60 - STV



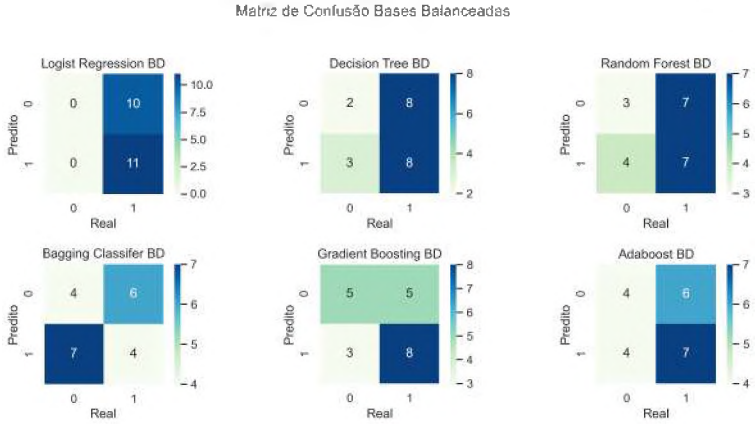
Fonte: Elaborada pela autora(2022).

Figura 95 – Matriz de Confusão - Subamostra 03 - Atraso $\geq$ 60 - STV



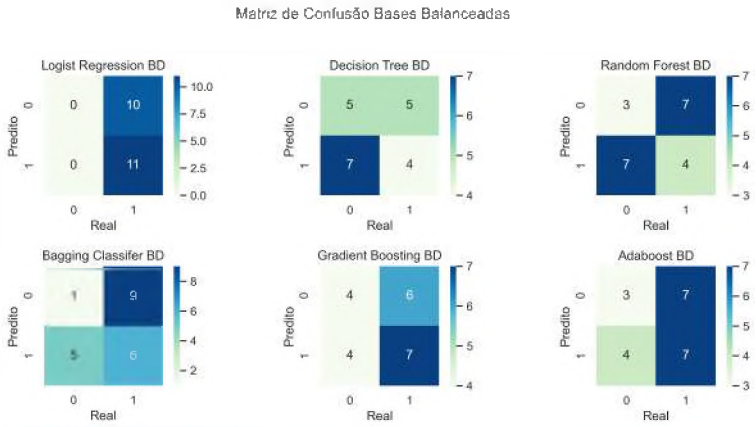
Fonte: Elaborada pela autora(2022).

Figura 96 – Matriz de Confusão - Subamostra 04 - Atraso $\geq$ 60 - STV



Fonte: Elaborada pela autora(2022).

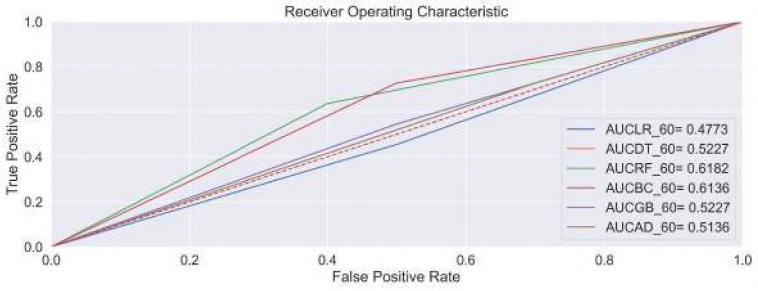
Figura 97 – Matriz de Confusão - Subamostra 05 - Atraso $\geq$ 60 - STV



Fonte: Elaborada pela autora(2022).

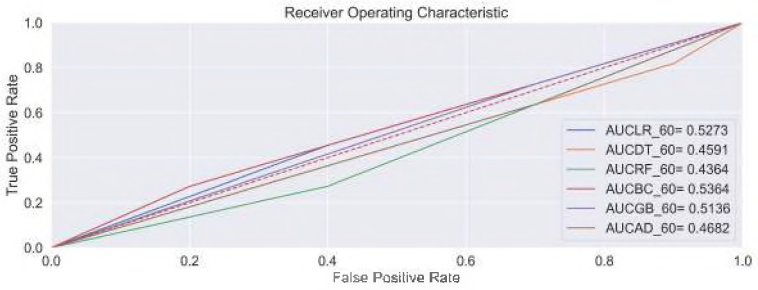
A decisão sobre a escolha do algoritmo com melhor desempenho também considerou a métrica AUC em cada Subamostra, dentro do subgrupo Atraso $\geq$ 90, ver Figuras 105, 106 e 107.

Figura 98 – AUC - Subamostra 02 - Atraso ≥ 60 - STV



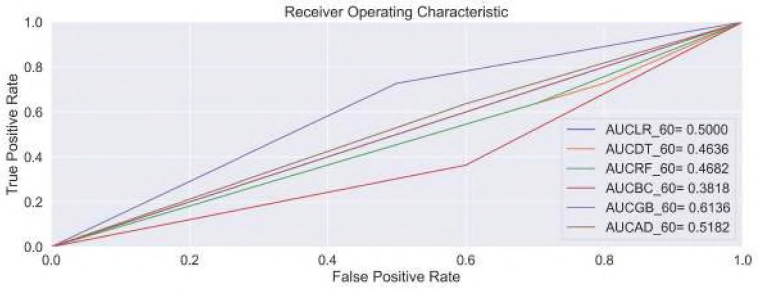
Fonte: Elaborada pela autora(2022).

Figura 99 – AUC - Subamostra 03 - Atraso ≥ 60 - STV



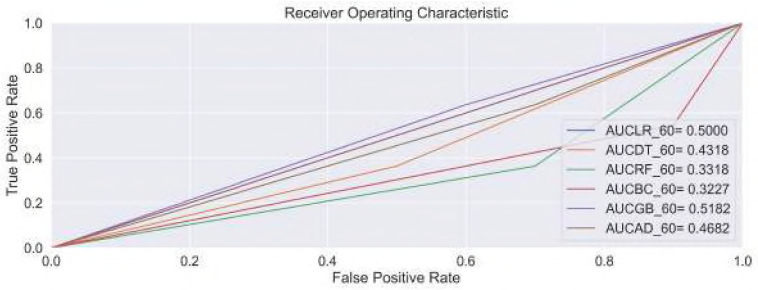
Fonte: Elaborada pela autora(2022).

Figura 100 – AUC - Subamostra 04 - Atraso ≥ 60 - STV



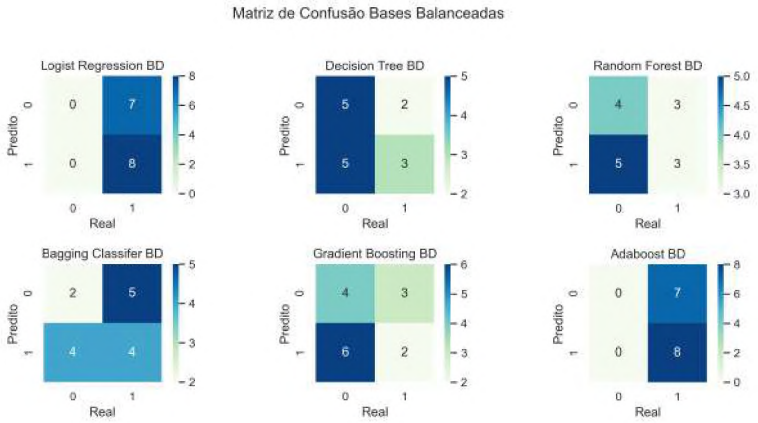
Fonte: Elaborada pela autora(2022).

Figura 101 – AUC - Subamostra 05 - Atraso ≥ 60 - STV



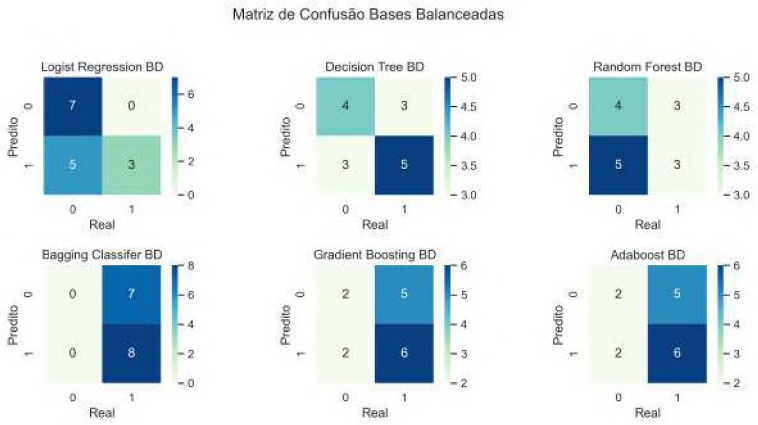
Fonte: Elaborada pela autora(2022).

Figura 102 – Matriz de Confusão - Subamostra 01 - Atraso ≥ 90 - STV



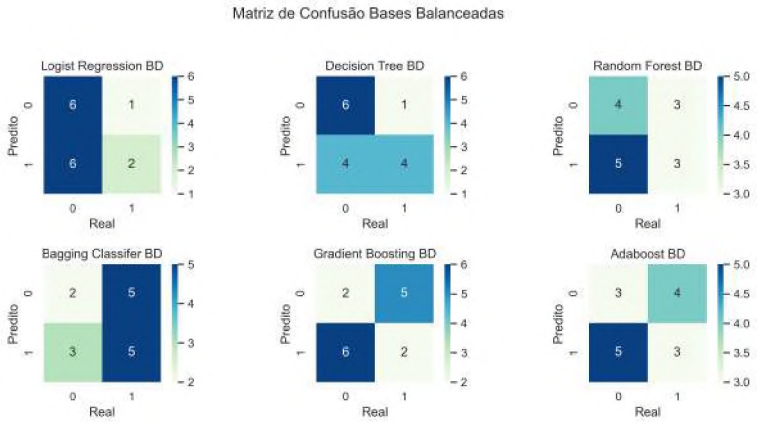
Fonte: Elaborada pela autora(2022).

Figura 103 – Matriz de Confusão - Subamostra 03 - Atraso ≥ 90 - STV



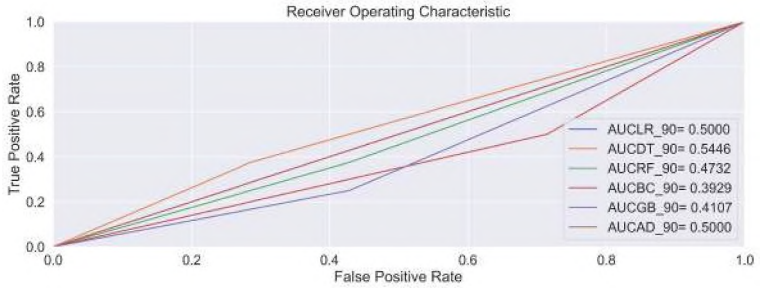
Fonte: Elaborada pela autora(2022).

Figura 104 – Matriz de Confusão - Subamostra 04 - Atraso ≥ 90 - STV



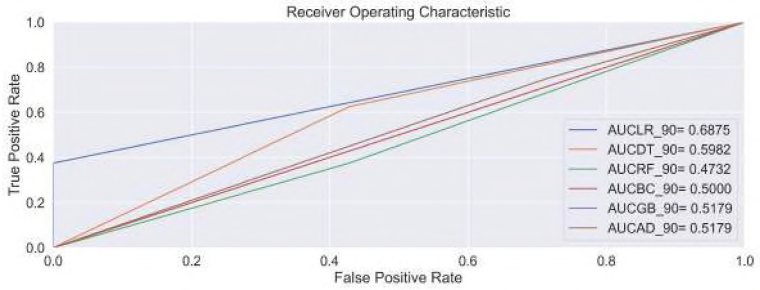
Fonte: Elaborada pela autora(2022).

Figura 105 – AUC - Subamostra 01 - Atraso ≥ 90 - STV



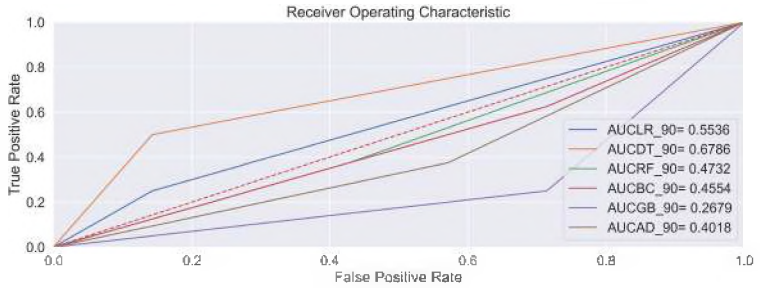
Fonte: Elaborada pela autora(2022).

Figura 106 – AUC - Subamostra 03 - Atraso ≥ 90 - STV



Fonte: Elaborada pela autora(2022).

Figura 107 – AUC - Subamostra 04 - Atraso ≥ 90 - STV



Fonte: Elaborada pela autora(2022).