MAURICIO HIROKI TOMIDA & MAYKON SOUZA CRUZ

MACHINE LEARNING PARA RATING DE CRÉDITO DE COMPANHIAS BRASILEIRAS NÃO FINANCEIRAS

São Paulo
2022

,

MAURICIO HIROKI TOMIDA & MAYKON SOUZA CRUZ

MACHINE LEARNING PARA RATING DE CRÉDITO DE COMPANHIAS BRASILEIRAS NÃO FINANCEIRAS

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Engenheiro Mecatrônico.

São Paulo
2022

MAURICIO HIROKI TOMIDA & MAYKON SOUZA CRUZ

MACHINE LEARNING PARA RATING DE CRÉDITO DE COMPANHIAS BRASILEIRAS NÃO FINANCEIRAS

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Engenheiro Mecatrônico.

Orientador:

Marcos de Sales Guerra Tsuzuki

São Paulo
2022

AGRADECIMENTOS

Gostaríamos de agradecer a todos que contribuíram direta e indiretamente para tornar este projeto possível, especialmente:

Aos nossos familiares que sempre nos apoiaram.

Ao nosso orientador Prof. Dr. Marcos de Sales Guerra Tsuzuki por ter nos norteado ao longo de toda jornada.

Aos nossos colegas pela disponibilidade e companheirismo.

RESUMO

Este trabalho propõe um modelo de machine learning como ferramenta de apoio para análise do risco de crédito, detectando a tendência do rating de crédito de companhias brasileiras não financeiras com base, exclusivamente, nos seus demonstrativos financeiros. Foi proposto um modelo de machine learning utilizando os demonstrativos financeiros da Economatica e o histórico dos ratings de crédito da Bloomberg. Como as bases não são totalmente compatíveis, foi construída uma aplicação em R para compatibilizá-las. Em seguida, a base de dados foi importada no ambiente Python e, utilizando bibliotecas de machine learning, principalmente o Sklearn, foram executadas as seguintes etapas: pré-processamento da base de dados, divisão da base em treinamento e teste, treinamento dos modelos utilizando validação cruzada com a base de treinamento. Foi feita a seleção dentre os seguintes modelos: árvorte de decisão, KNN, Regressão Logística, SVM, Rede Neural (MLP), Random Forest, XGBoost e LightGBM. O melhor modelo foi o Random Forest, que passou por um tuning e avaliação utilizando a base de teste com diferentes métricas, obtendo uma acurácia de 90.12% e F1-Score de 90.04%, demonstrando o bom desempenho do modelo em classificar tanto as classes majoritárias quanto as classes minoritárias. Os resultados mostram que o modelo é sensível às alterações dos indicadores dos demonstrativos financeiros, capturando a tendência da saúde financeira real da companhia, traduzida em forma de rating de crédito.

Palavras-Chave – Machine Learning; Rating de Crédito; Random Florest; Pré-processamento de dados; Análise de dados

ABSTRACT

This paper proposes a machine learning model as a support tool for credit risk analysis, detecting the credit rating trend of Brazilian non-financial companies based exclusively on their financial statements. A machine learning model was proposed using Economatica's financial statements and Bloomberg's credit rating history. As the bases are not fully compatible, an application was built in R to make them compatible. Then, the database was imported into the Python environment and using machine learning libraries, mainly Sklearn, the following steps were performed: pre-processing the database, split the base into training and testing, model training using cross-validation with the training base. A selection was made among the following models: decision tree, KNN, Logistic Regression, SVM, Neural Network (MLP), Random Forest, XGBoost and LightGBM. The best model was the Random Forest, which underwent tuning and evaluation using the test base with different metrics, obtaining an accuracy of 90.12% and F1-Score of 90.04%, demonstrating the good performance of the model in classifying both the majority classes and minority classes. The results show that the model is sensitive to changes in financial statement indicators, capturing the trend of the company's real financial health, translated into the form of a credit rating.

Keywords – Machine Learning; Credit Rating; Random Forest; Data pre-processing; Data Analysis

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Segundo dados da Associação Brasileira das Entidades dos Mercados Financeiro e de Capitais, o mercado de crédito privado brasileiro cresceu em quatro vezes nos últimos seis anos. No entanto, existe uma disfunção na distribuição dos recursos de acordo com os relatórios do Banco Central sobre empréstimos para Pessoa Jurídica no Brasil, que mostram que a parcela de crédito destinada às grandes empresas é maior que o volume de todas as demais somado. Para a McKinsey, parte dessa assimetria é explicada pela falta de informações ou por informações não confiáveis sobre a saúde financeira da companhia que procura por empréstimos.

O modelo para rating de crédito de empresas não financeiras é uma tentativa de se antecipar às alterações no rating de crédito das companhias tomadoras de crédito no Brasil, assim, logo que alguma nova informação relativa aos demonstrativos financeiros de determinada companhia é divulgada, o modelo é capaz de dizer se houve ou não alteração no rating, bem como se contrapor às classificações divulgadas pelas agências, mostrando, assim, a real tendência da saúde financeira das companhias em forma de rating de crédito.

Quando uma instituição financeira vai conceder crédito a uma companhia, o credor avalia diversos fatores para concessão ou não do crédito, entre esses, um dos mais importantes é o risco crédito do tomador, que é obtido por meio do rating de crédito. O rating de crédito de uma companhia é divulgado pelas agências classificatórias de riscos, como, Standard & Poor's (S&P), Moody's e Fitch, que a partir das análises dos demonstrativos financeiros da empresa, análise macroeconômica e setorial, obtêm o score. No entanto, a divulgação do rating é feita em intervalos esparsos, o que protege a empresa de oscilações momentâneas de mercado. Todavia, essa estabilidade pode mascarar uma alteração real do rating, já que uma deterioração em algum dos indicadores financeiros da companhia poderia demorar para ser refletida pelas agências. Assim, surge a necessidade de um modelo que possa detectar com antecedência as alterações no rating da companhia baseado nos dados dos demonstrativos financeiros e que tenha boa acurácia considerado a até três notche de distância, que é o degrau entre cada nota na régua de rating.

Como os modelos das agências são fechados, o modelo foi desenvolvido com base nos dados históricos de rating divulgados pelas agências em conjunto com o histórico dos demonstrativos financeiros. Assim, grande parte do trabalho ficou concentrada no tratamento e análise dos dados, bem como, na escolha da metodologia mais adequada.

Desse modo, utilizou-se a seguinte estratégia: obter os dados históricos de rating e demonstrativos financeiros das companhias de capital aberto listadas na Bolsa de Valores de São Paulo (B3), tratar e analisar os dados para assim encontrar quais são as variáveis relevantes, dentre todos os indicadores financeiros, como, liquidez, alavancagem, setor, margem líquida, operacional etc., que influenciam no rating.

Na preparação de dados, realizou-se a coleta do histórico de rating das companhias na Bloomberg e o histórico dos demonstrativos financeiros na Economatica utilizando a linguagem R para coleta e manipulação de um grande volume de dados. Em seguida, importou-se a base no Python. Utilizando bibliotecas de machine learning, principalmente o Sklearn, foram executadas as seguintes etapas: pré-processamento da base de dados, divisão da base em treinamento e teste, treinamento do modelo utilizando validação cruzada com a base de treinamento, seleção e tuning do melhor modelo e avaliação do modelo final utilizando a base de teste com diferentes métricas.

2 OBJETIVOS GERAIS

Assim, o objetivo do trabalho é não só obter um modelo para o rating de crédito de uma companhia a partir dos indicadores dos demonstrativos financeiros unicamente, como também, chegar à metodologia que faz com que os resultados do modelo mais se aproximem da realidade.

O modelo será capaz de determinar o rating de uma companhia, exclusivamente, a partir dos demonstrativos financeiros, quando o rating ainda não foi divulgado pelas agências, com 80% de acurácia, considerando até três notches de distância do rating encontrado para o que será divulgado pelas agências de classificação. O modelo, também, será capaz de fazer contrapontos aos rating divulgados pelas agências utilizando apenas os demonstrativos financeiros.

Nos casos em que há discordância entre o modelo e as agências, isso pode indicar que as agências não estão refletindo a saúde financeira da companhia baseada nos demonstrativos financeiros, uma vez que o modelo de machine learning aprende o padrão de classificação das agências baseado nos indicadores financeiros e, com um modelo com capacidade de generalização e sem sobreajuste à base de dados, quando houver uma discrepância entre o rating predito pelo modelo e o rating divulgado pelas agências, o modelo deverá detectar. Visto que o modelo aprende o padrão de classificação das agências, que normalmente reflete com fidelidade a saúde financeira das companhias, é possível utilizar o modelo proposto para prever os ratings nos casos em que as agências atrasam a divulgação das classificações.

2.1 Interesse econômico/social

Saber se vai haver ou não alteração no rating de crédito de uma companhia com antecedência é fundamental tanto para a companhia que poderá atuar rapidamente na correção do parâmetro deteriorado, quanto para as instituições financeiras concessoras de crédito, já que esta poderá se precaver de um possível default, isso pensado em operações

já em andamento. Todavia, deve ser considerada a possibilidade do uso de má-fé por determinada empresa que, de posse da informação da existência de deterioração de algum dos seus indicadores financeiros, poderia postergar um empréstimo, ou ainda tomar novas dívidas, mesmo sabendo que não terá como honrá-las. Portanto, fica claro a necessidade de se antecipar às alterações divulgadas pelas agências, assim como se contrapor às classificações quando divulgadas, mostrando a real tendência da saúde financeira das companhias em forma de rating de crédito.

2.2 Mérito técnico/científico

Além do desenvolvimento do modelo, será feito um extenso estudo voltado ao comparativo de técnicas para solução de problemas de mesma natureza e que necessitam de resultados robustos e consistentes. Abordando diferentes técnicas de machine learning, ao final, foi possível determinar, não só as vantagens e desvantagens do uso de uma ou outra técnica, bem como o caminho que entrega os resultados mais consistentes e representativos da realidade.

3 ESTADO DA ARTE

Depois de uma busca por trabalhos relacionados com o tema, foram encontrados alguns textos que, embora de forma segmentada, abordam o tema. Assim, foi necessário dividir a pesquisa em dois grandes grupos que serão aqui chamados de Modelo Econômico e Modelo de Machine Learning, o que possibilita uma abordagem detalhada de cada um dos temas.

3.1 Modelo econômico

O objetivo central dos modelos de credit scoring é obter a probabilidade de default (PD) de uma dívida ou ainda uma proxy para o rating de crédito de uma companhia. De modo geral, a probabilidade de default é a possibilidade de um cedente não honrar alguma de suas obrigações financeiras.

No modelo econômico, o foco é encontrar, principalmente, quais serão os parâmetros candidatos a variáveis explicativas, bem como a determinação da composição desses parâmetros a partir dos indicadores financeiros. Nesse ponto, fica claro que uma segmentação por setores é necessária, considerando que cada setor possui características intrínsecas, todavia há parâmetros que são essenciais para a saúde financeira de qualquer companhia. Dentre esses, os mais comuns encontrados nos trabalhos utilizados como base são:

$\frac{\textbf{Capital Próprio}}{\textbf{Ativo total}}$: Esse indicador mostra qual a porção dos ativos da companhia é proveniente de recursos próprios, assim, quanto mais próximo de zero for essa razão, mais frágil é a companhia, pois indica que seus ativos estão sendo financiados por terceiros, logo, grande parte das receitas geradas serão destinadas ao pagamento de juros e amortizações das dívidas, além de deixar a companhia exposta aos movimentos das curvas de juros.

$\frac{\textbf{Dívida Circulante}}{\textbf{EBITDA}}$: Esse índice traduz quanto do EBITDA (Earnings Before Interest, Taxes, Depreciation and Amortization), proxy para o lucro operacional, estará comprometido com a dívida de curto prazo da companhia, quanto menor esse indicador, mais robusta é a geração de caixa da companhia frente às suas dívidas. No entanto, se o resultado for

maior que 1, significa que a empresa não consegue gerar receita suficiente para cobrir as dívidas de curto prazo, situação que pode evoluir para um default caso o caixa da cia não seja suficiente para cobrir o déficit do EBITDA.

$\frac{\textbf{Receita Líquida}}{\textbf{Ativos Totais}}$: Esse indicador mede quanto a companhia consegue gerar de receita com os ativos que possui, uma tentativa de mensurar a eficiência da operação, quanto maior esse indicador, mais eficiente é a empresa.

Valor de Mercado: Indicador que reflete como o mercado "vê" a companhia, ou seja, o valor de consenso (justo) no mercado, logo, é um consolidado da opinião de todos os agentes do mercado, todavia, para evitar distorções com o valor de empresas de grande porte, será utilizado o log do valor de mercado.

$\frac{\textbf{Ativo Circulante}}{\textbf{Passivo Circulante}}$: Conhecido como liquidez corrente, esse indicador mensura a capacidade da companhia de cobrir suas dívidas de curto prazo (até 1 ano), ou seja, se essa razão for menor que 1, indica que a empresa terá dificuldades de honrar suas dívidas de curto prazo, o que pode ser o prenúncio de um default, o inverso não necessariamente é um bom sinal, se o resultado da fração for muito maior que 1, pode significar um mal gerenciamento dos recursos da companhia, pois há um grande provisionamento de recursos para o curto prazo que não serão utilizados.

$\frac{\textbf{Lucro líquido}}{\textbf{Receita líquida}}$: Chamado de margem líquida, esse indicador reproduz a eficiência da companhia em transformar receita líquida em lucro líquido, logo, quanto maior essa razão, mais lucrativa a empresa, no entanto, quando o resultado é negativo indica que, apesar de gerar receita, a empresa não consegue convertê-la em lucro, tal cenário expõe uma companhia com saúde financeira fragilizada, o que representa um elevado risco de default.

Para o modelo econômico, foram escolhidos dois trabalhos como referência:

O primeiro artigo intitulado "Assessing Corporate Risk: a PD model based on credit Ratings" (CARDOSO et al., 2013) [1], discute de forma precisa, ainda que resumida, todas as fases da construção do modelo econômico que aqui foi abordada.

Para o estudo, os pesquisadores utilizaram uma amostra com dados de mais de 2 mil companhias não financeiras globais que possuem rating de crédito emitido por agências internacionais e mapearam os ratings para a média da probabilidade de default de 5 anos, com base no Acordo de Basiléia II (BCBS, 2004). Esse intervalo de 5 anos é uma tentativa de capturar eventos de crédito, já que no curto prazo tais eventos são raros. Para as a variáveis, os pesquisadores dividiram os indicadores dos demonstrativos financeiros em 5 categorias: Rentabilidade, alavancagem, liquidez, tamanho, setor e cobertura de dívida, que estão relacionados com probabilidade de default, segundo a literatura tradicional (desde Fitzpatrick 1928, Beaver 1966, Altman, 1968). Para estimar o modelo, os pesquisadores utilizaram regressão logística, que será tratado no próximo tópico.

O segundo trabalho com título "Likelihood of nonpayment of large enterprises in National Financial System"(AKIAMA, 2008) [2], foi incluído como referência porque assumi uma abordagem diferente da conduzida pelo paper anterior. Nesse, a pesquisadora explorou profundamente a revisão bibliográfica, o que torna o trabalho uma poderosa fonte de consulta.

3.2 Rápida Revisão de Machine Learning

Aprendizado de máquina (Machine Learning) é o processo de automatizar a aprendizagem das relações significativas e padrões por meio de exemplos e observações, ao invés de codificar o conhecimento nos computadores [3]. Um dos grandes avanços nos algoritmos de aprendizagem foi a evolução das redes neurais artificiais para redes neurais profundas, que trouxeram um novo panorama na capacidade de aprendizagem [4]. Tanto o Machine Learning quanto o Deep Learning possuem suas particularidades, assim como vantagens e desvantagens [5].

Existem diferentes tipos de modelos preditivos e suas respectivas aplicações [6]. Para o estudo dos artigos e para o desenvolvimento do presente projeto são utilizados modelos de aprendizagem supervisionada [7]. Para esse tipo de modelo, os dados de entrada possuem rótulos (labels), ou seja, cada entrada de dado possui um rótulo com a saída correta, permitindo o modelo aprender sobre cada tipo de dado. Na fase de treinamento, a máquina

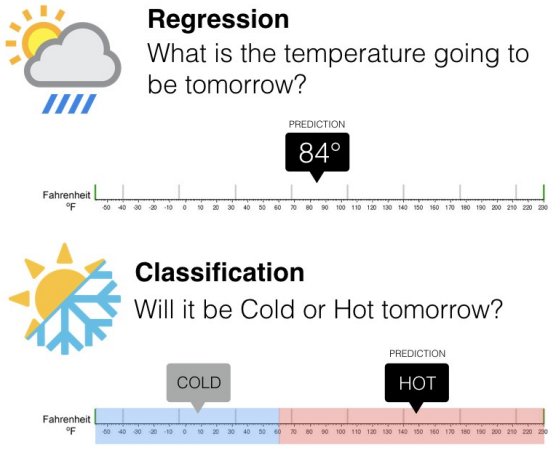
de aprendizagem é provida com os dados de entrada e de saída (rótulo) para o algoritmo encontrar as relações e padrões entre a entrada e a saída, criando assim um modelo. Em seguida, na fase de teste a máquina recebe apenas as entradas e a saída é prevista pelo modelo e comparada com a saída correta.

A aprendizagem supervisionada pode ainda ser dividida em duas categorias: Classificação e Regressão [8].

- Classificação: algoritmo utilizado quando a variável de saída é categórica, ou seja, temos um número discreto de categorias possíveis para a saída.
- Regressão: a saída é contínua, ou seja, um número real.

A figura 1 apresenta um exemplo de classificação e regressão para um mesmo contexto de previsão da temperatura.

Figura 1: Classificação vs Regressão



Fonte: Towards Data Science, Taylor Fogarty,2019. [9]

Nas seções a seguir, serão abordados os conceitos e técnicas, além de 4 artigos relacionados com o tema deste trabalho com enfoque no pré-processamento de dados, criação de modelos, seleção de algoritmos, seleção de atributos e ajuste de hiperparâmetros [10] para melhor performance dos modelos. Além disso, serão comparadas as informações que permitem encontrar as semelhanças e divergências entre as técnicas e resultados deste trabalho e dos artigos estudados.

3.3 Conceitos e Técnicas Importantes sobre Machine Learning

Nessa seção, serão abordados conceitos e técnicas importantes para compreensão das técnicas utilizadas para a criação de um modelo preditivo.

Serão abordados os seguintes tópicos: pré-processamento de dados, modelos de Machine Learning, treinamento e teste de modelos e métricas para avaliação de modelos.

3.3.1 Pré-processamento de dados

Ao utilizar uma base de dados, devemos prepará-la para torná-la adequada aos modelos de machine learning [11]. A seguir, serão descritos alguns procedimentos importantes para preparar os dados não-tratados (raw data) para torná-los adequados aos algoritmos de Machine Learning.

a) Limpeza de Dados

Os modelos de machine learning não funcionam com dados faltantes (missing values), assim como dados duplicados, inconsistentes e outliers podem afetar o desempenho do algoritmo. Portanto, deve-se utilizar técnicas para remover/tratar dados indesejáveis [12].

b) Data Enconding

Outro requisito dos modelos de machine learning é a utilização apenas de dados com atributos numéricos. Logo, utiliza-se técnicas para transformar dados categóricos em dados numéricos: Label Encoding, One-Hot Encoding, Effect Encoding, entre outros [13].

c) Escalonamento de Dados

Alguns modelos de machine learning são sensíveis a escala dos dados, afetando a sua performance. Existem diferentes técnicas para resolver o problema da escala dos dados, tais como: MinMaxScaler, RobustScaler, StandardScaler e Normalizer [14].

d) Dados desbalanceados

Dados desbalanceados é quando a base de dados não possui as classes igualmente representadas, pois há pequena incidência de uma ou mais classes (classes minoritárias) em comparação com as demais categorias (classes majoritárias). Se o modelo for treinado sem considerar essa desproporcionalidade, os resultados podem não ser representativos para as classes minoritárias, devido ao Paradoxo da Acurácia [15]. Portanto, devem ser aplicadas algumas técnicas para lidar com esse problema [16].

3.3.2 Modelos de Machine Learning

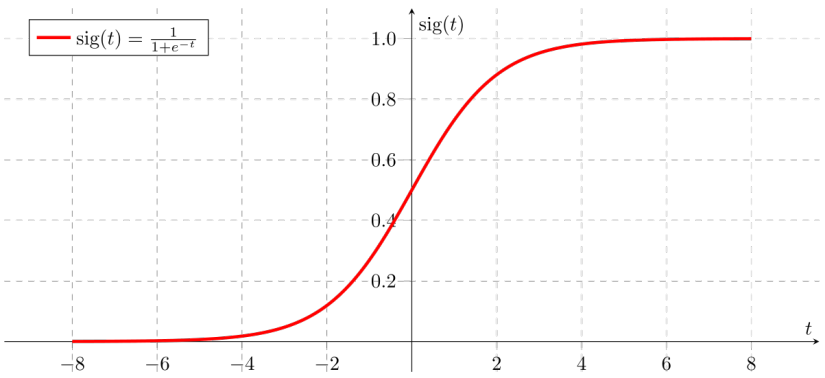
Para este trabalho serão estudados os seguintes modelos: Regressão Logística, KNN (K-Nearest Neighbors), SVM (upport Vector Machine), árvores de decisão, ensembles (bagging, random florest e boosting) e redes neurais.

Como os modelos possuem uma teoria matemática e estatística extensa, serão abordados resumidamente os modelos mencionados e serão fornecidas fontes para maior aprofundamento, pois são temas já abordados exaustivamente por artigos, blogs, revistas, cursos, livros, etc.

I) Regressão Logística

A regressão logística é utilizada para algoritmos de classificação e regressão. Utiliza uma função sigmóide (formato em "S") para calcular a probabilidade de determinada instância pertencer a uma determinada classe. 2.

Figura 2: Regressão Logística



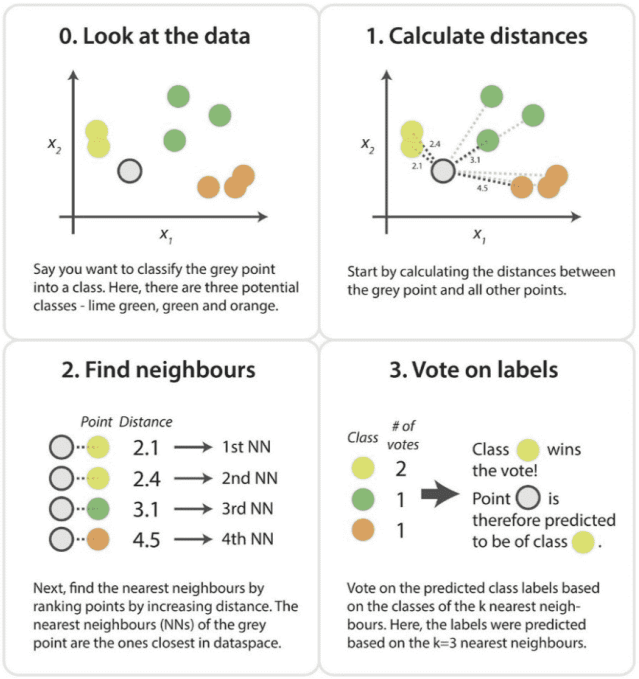
Fonte: Towards Data Science, Saishruthi Swaminathan, 2018. [17]

Esse modelo apresenta diferentes tipos: binário, multinomial, ordinal. Para seu funcionamento, o algoritmo precisa: estimar probabilidades, treinar, encontrar parâmetros que minimize a função de custo e definir as fronteiras de decisão [17]

II) KNN (K-Nearest Neighbors)

O KNN é um algoritmo não-paramétrico que pode ser utilizado para regressão e classificação [18]. O método se baseia em calcular a distância de uma nova instância com as K instâncias mais próximas na base de dados e definir a qual classe ela irá pertencer 3. Ou seja, não é criado um modelo para esse método, sendo necessário ter a base de dados disponível para poder calcular a distância entre novas instâncias com os dados na base de dados.

Figura 3: KNN



Fonte: Medium, Antony Christopher, 2021. [19]

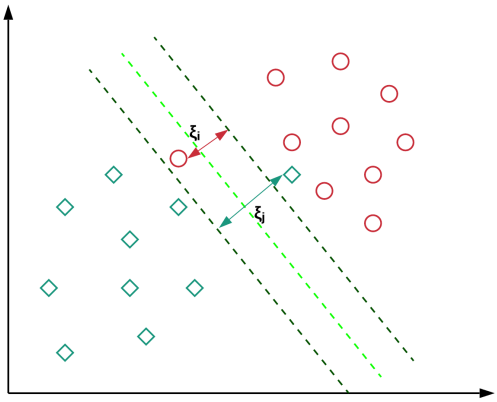
Para o cálculo da distância, podem ser utilizados diferentes métodos [20].

III) SVM (Support Vector Machine)

O SVM é um algoritmo para regressão e classificação, que busca o hiperplano que melhor separa as classes com margem máxima 4, ou seja, o hiperplano que melhor se

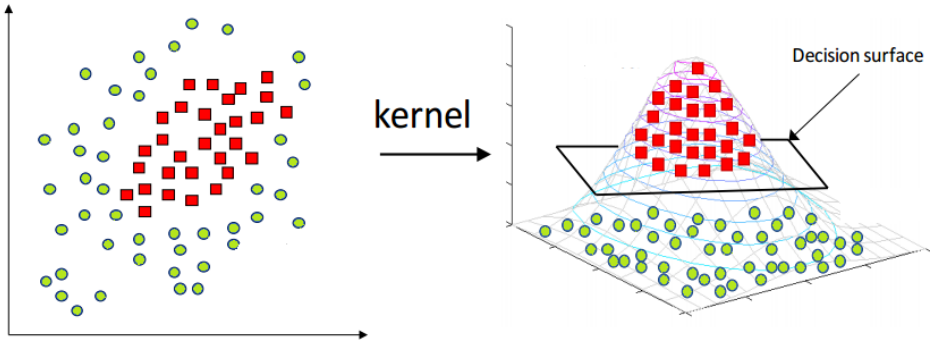
distancia dos pontos mais próximos de cada classe [21]. A separação de dados não-lineares é feita através de kernels 5 [22].

Figura 4: Funcionamento do SVM: margem máxima



Fonte: Towards Data Science, Rishabh Misra, 2019. [23]

Figura 5: Funcionamento do SVM: Kernel



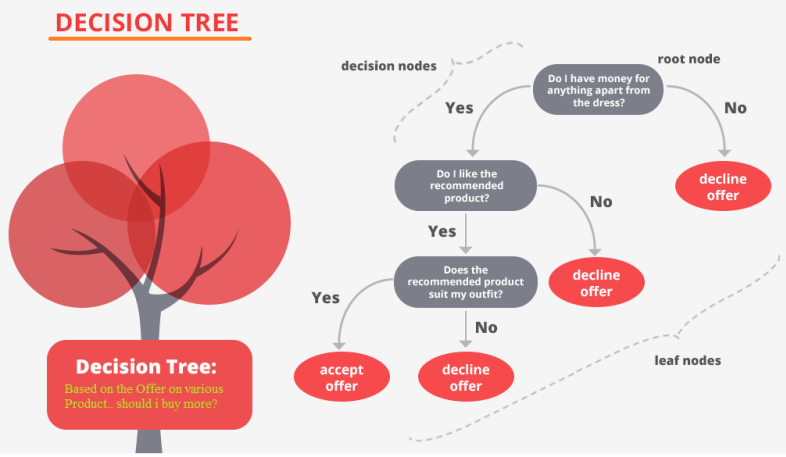
Fonte: Medium, Siddhartha Sharma, 2019. [22]

IV) Árvores de decisão

As árvores de decisão são algoritmos para regressão e classificação. As árvores de decisão estabelecem nós que estão ligados de forma hierárquica. Existe o nó-raiz, que é o mais importante, e os nós-folha, que apresentam a saída do modelo. A ligação entre os nós é feita pelas regras de "if-else", ou seja, ao chegar em um determinado nó, o algoritmo verifica se determinada condição é satisfeita e, então, decide para qual nó prosseguir. Esse processo é repetido até chegar no nó-folha [24].

A seguir, um exemplo de árvore de decisão de classificação 6.

Figura 6: Exemplo de Árvore de Decisão de Classificação



Fonte: Data Science Foundation, Mayank Tripathi, 2020. [25]

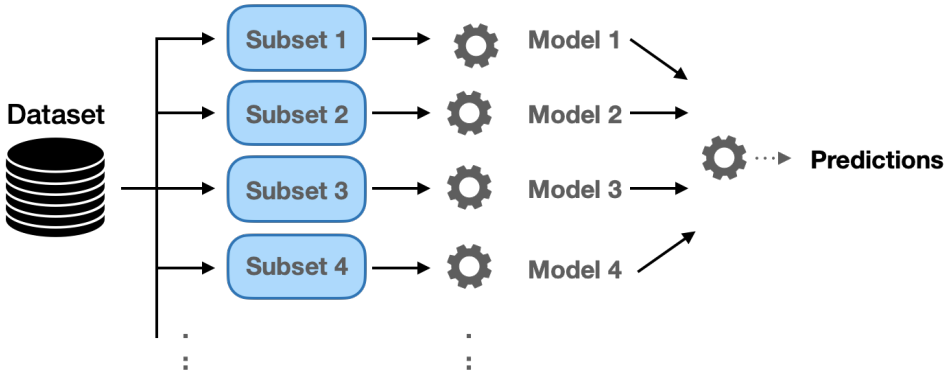
V) Ensembles

Modelos de Ensemble é uma técnica de aprendizagem onde múltiplos modelos mais fracos são treinados com a mesma base de dados e seus resultados são combinados para obter modelos mais robustos e com melhor performance [26]. Neste trabalho, serão explorados ensembles de árvores de decisão.

a) Bagging .

Nesse modelo, n árvores de decisão são treinadas de forma independente por diferentes amostras escolhidas aleatoriamente da base de dados. O resultado do modelo é a combinação das n árvores treinadas 7.

Figura 7: Modelos utilizando Bagging.



Fonte: Towards Data Science, Mohammed Alhamid, 2021. [26]

b) Random Florest .

Random Forest é uma extensão do Bagging, onde além de n amostra aleatória da base de dados, também são selecionadas aleatoriamente m características (features) para cada árvore.

c) Boosting .

Nesse modelo também há n árvores de decisões com amostras aleatórias, porém elas não são independentes, pois a aprendizagem é sequencial. Cada árvore é treinada de forma a minimizar o erro cometido pelas árvores anteriores 8.

Figura 8: Modelos utilizando Boosting

Fonte: Towards Data Science, Mohammed Alhamid, 2021. [26]

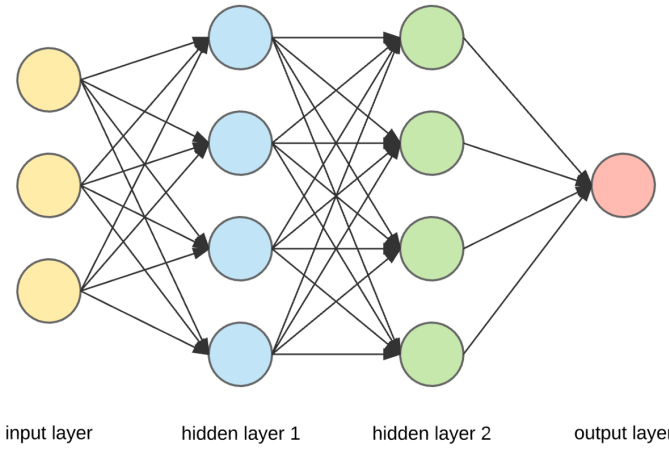
Uma extensão dos algoritmos de boosting é o Gradient Boosting, porém o erro é minimizado utilizando os erros residuais das árvores anteriores. Ou seja, é a combinação do algoritmo gradient descent [27] com o método de boosting.

VI) Redes Neurais

As redes neurais artificiais são estruturas inspiradas no cérebro humano, imitando o funcionamento e estrutura dos neurônios biológicos [28].

As redes neurais artificiais são compostas por camadas de nós (neurônios artificiais). Essas camadas podem ser divididas em: camada de entrada, camadas intermediárias ou ocultas e camada de saída 9.

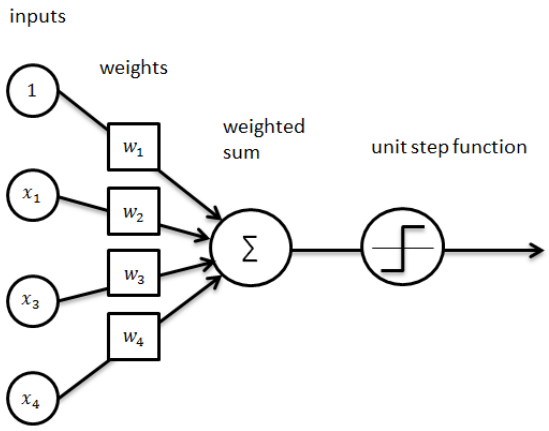
Figura 9: Camadas das Redes Neurais



Fonte: Towards Data Science, Gavril Ognjanovski, 2019. [29]

O funcionamento das redes neurais é composto por dados de entrada, um viés (ou limite), pesos e uma saída. É feita uma soma ponderada utilizando as entradas e os pesos e o resultado passa por uma função de ativação para gerar a saída 10.

Figura 10: Funcionamento das Redes Neurais



Fonte: Towards Data Science, Gavril Ognjanovski, 2019. [29]

3.3.3 Treinamento e Teste do Modelo

Escolhido o modelo, deve-se treinar e testá-lo utilizando a base de dados. Esses processos podem ser feitos de diferentes maneiras:

1. A forma mais clássica e simples é a divisão da base de dados em base de treinamento e base de teste [30]. A maior parte dos dados devem ser usados para treinamento (75% - 85% dos dados) e o restante para teste (15% - 25% dos dados). O modelo é treinado com o conjunto de treinamento por meio de aprendizagem supervisionada. Em seguida, utiliza-se as entradas do conjunto de teste para o modelo prever as saídas, que é então comparada com as saídas corretas.

2. Alternativamente, é possível dividir a base de dados em três partições: treinamento, validação e teste [31]. Dessa forma, o modelo utiliza o conjunto de validação para avaliar os resultados do conjunto de treinamento. Então, utiliza o conjunto de teste para verificar o melhor modelo testado no conjunto de validação.

3. Separar os dados em somente dois conjuntos (treinamento e teste) pode trazer resultados divergentes, dependendo da informação contida em cada conjunto [32]. Por isso, a validação cruzada por k-fold minimiza esse problema. O método consiste em dividir os dados em K partes iguais, ajusta-se o modelo utilizando K-1 partes, e a parcela restante fica destinada à validação. Esse processo é repetido K vezes (em

cada momento uma partição diferente será a validação). O resultado é a média dos K resultados obtidos. [33].

3.3.4 Métricas para avaliação de modelos

Ao criar diferentes modelos é necessário avaliá-los utilizando alguma métrica. Existem diversas métricas diferentes [34], entretanto, nesse documento, serão exploradas as métricas relevantes para a análise dos artigos e desenvolvimento do projeto: matriz de confusão, precisão, revocação, acurácia e F1 Score.

a) Matriz de confusão

Matriz NxN, onde N é o número de classes sendo previstas e é utilizada para aprendizagem supervisionada de classificação [35]. Para N=2, temos uma matriz de confusão para um modelo de saída binária, onde:

- Verdadeiro positivo (True Positive - TP): ocorre quando a classe positiva é prevista corretamente.
- Verdadeiro negativo (True Negative - TN): ocorre quando a classe negativa é prevista corretamente.
- Falso positivo (False Positive - FP): ocorre quando a classe positiva é prevista incorretamente.
- Falso negativo (False Negative - FN): ocorre quando a classe negativa é prevista incorretamente.

A matriz de confusão binária pode ser visualizada na figura 11

Figura 11: Matriz de Confusão Binária

		Valor Predito	
		Sim	**Não**
Real	**Sim**	Verdadeiro Positivo (TP)	Falso Negativo (FN)
	Não	Falso Positivo (FP)	Verdadeiro Negativo (TN)

Fonte: Diego Nogare, 2020. [36]

b) Precisão e Revocação

A precisão é uma métrica que indica, dentre as classificações positivas do modelo, quantas foram corretamente previstas (3.1).

$$Precisao = \frac{TP}{TP + FP} \tag{3.1}$$

Já a revocação é uma métrica que indica, dentre as amostras positivas existentes, quantas o modelo previu corretamente (3.2).

$$Revocacao = \frac{TP}{TP + FN} \tag{3.2}$$

Infelizmente, não é possível ter precisão e revocação altos, o que é chamado de compensação da Precisão/Revocação [37].

c) Acurácia

A acurácia mensura a proporção do número de classes previstas corretamente em relação a totalidade das classes 3.3.

$$Acuracia = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.3}$$

d) F1 score

Para tentar encontrar a melhor combinação de precisão (3.1) e revocação (3.2), utiliza-se a métrica F1 Score(3.4), que calcula a média harmônica entre precisão e revocação.

$$F_1 = 2 \times \frac{Precisao \times Revocacao}{Precisao + Revocacao} \tag{3.4}$$

e) Avaliação multi-classes

Para base de dados com múltiplas classes de saída (múltiplos ratings), a precisão, a revocação e o F1-score podem ser calculadas de diferentes formas:

- Macro: todas classes contribuem igualmente para a métrica final. É calculada pela média da métrica de todas as classes.
- Micro: todas as amostras contribuem igualmente para a métrica final. É calculada pela métrica do total de valores.
- Weighted: cada classe contribui para a métrica final proporcionalmente ao seu tamanho. É calculada baseada na métrica por classe ponderada pelo número de amostras por classe na base de dados.

3.4 Estado da Arte: Machine Learning

Os algoritmos de regressão logística ainda são amplamente utilizados para analisar crédito, apesar de vários artigos demonstrarem uma maior eficiência de outros algoritmos mais sofisticados, tais como Random Forest e XGBoost.

A seguir, serão apresentados 4 artigos relacionados ao tema deste trabalho e, em seguida, serão feitas comparações com a metodologia e resultados deste projeto.

3.4.1 Artigo 1 - Análise de Risco de Crédito usando algoritmos de Machine Learning

No trabalho de Pereira [38], foi utilizada a base de dados da CRSP (Center for Research in Security Prices, LLC), coletando uma amostra inicial composta por 3320 observações das demonstrações financeiras anuais de diversas empresas que constituem

o índice bolsista S&P500, no intervalo temporal de 2010 à 2018. Inicialmente, foram utilizadas 72 características (features) para criação do modelo.

Uma das limitações do estudo se deu pelo fato da pouca diversidade dos ratings, pois a base de dados continha poucas empresas com ratings baixos, próximos do nível default e, portanto, criou-se um modelo de classificação binário: Investment grade (de AAA até BBB-) e Non-Investment grade (de BB+ até CC).

Os Modelos testados foram: Regressão Logística com regularização L1 e L2 (ou algoritmo de Lasso e Ridge, respectivamente), Linear Discriminant Analysis (LDA), Gaussian Naives Bayes (GNB), Decision Tree, Random Forest, Support Vector Machine e K-Nearest Neighbours.

O trabalho de Pereira seguiu as seguintes etapas sequencialmente: preparação dos dados, criação do modelo, seleção dos algoritmos, seleção das variáveis e melhoramento do algoritmo escolhido com a respectiva análise dos resultados.

a) Preparação de dados

Inicialmente 3320 de demonstrações financeiras relativas a empresas do S&P500 de 2010 a 2018, e por mais de 500 variáveis para cada observação. Dentro deste conjunto, foram escolhidas pré-selecionadas 20 variáveis. Destas, foram criadas mais 5 novas variáveis e 46 rácios financeiros, além do acréscimo do Credit Rating do S&P500 (feature que deseja prever, ou seja, a saída do modelo), resultando em 72 variáveis (69 numéricas, 1 discreta, 1 categórica e 1 temporal).

Foram removidas as observações que apresentavam variáveis sem valores (missing values), resultando em uma amostra de 1755 observações.

Foram observados muitos outliers das variáveis numéricas e para solucionar este problema, foi calculado o skewness (distorção) [39] para saber qual algoritmo de escalonamento seria utilizado:

- Robust scaling(skewness > 2)
- MinMax Scalling ($1 <$ skewness < 2)
- Normal Standadization (skewness < 1)

Por último, a base de dados foi dividida em treino (85%, ou seja, 1491 observações) e de teste (15%, ou seja, 264 observações).

b) Criação do modelo e seleção do algoritmo

Usou a técnica SMOTE (Synthetic Minority Over-sampling Technique) [40] para solucionar o problema de dados desbalanceados. Com isso, foram obtidas 2246 observações, 1123 classificadas como Investment grade e 1123 como Non-Investment grade.

Após testar os modelos, Random Forest apresenta a maior acurácia (95.08%) e maior pontuação na validação cruzada (93.77%).

c) Seleção de variáveis e melhoramento do Algoritmo

Por último, foram feitos ajustes dos hiperparâmetros e seleção de variáveis do melhor modelo (Random Forest), além da análise de performance através da curva ROC, AUC e matriz de confusão.

Para selecionar as variáveis relevantes, foi testada uma variável de cada vez no novo modelo e caso o acréscimo da variável não resultar em um aumento superior a 0.1%, a feature é descartada. Dessa forma, foram selecionadas 20 features 12. Com isso, foram obtidos os seguintes resultados:

- Sem seleção de variáveis: AUC = 97.4835%
- Com seleção de variáveis: AUC = 97.5698%

Figura 12: Análise do artigo 1: Features Selecionadas

Features	Metric
CURRENT ASSETS	act
TOTAL ASSETS	at
CASH-FLOW/SALES	cfsale
DIVIDENDS	dvt
EBIT	ebit
EBIT/EBITDA	ebitebitda
EBITDA/TOTAL ASSETS	ebitdaat
EBT/EBIT	ebtebit
EMPLOYEES/SALES	empsale
INVENTORIES	invt
INVENTORIES/SALES	invtsale
GROSS PROFIT/LIABILITIES	gplt
CURRENT LIABILITIES	lct
LONG-TERM LIABILITIES/TOTAL ASSETS	ltlat
LN (NET INCOME)	lnni
NET INCOME/EMPLOYEES	niemp
NET INCOME/EQUITY	niteq
SALES/INVENTORIES	saleinvt
SALES/EQUITY	saleteq
INCOME TAXES	txt

Fonte: Print Retirado do Artigo de Pereira, 2020. [38]

Para o ajuste dos hiperparâmetros, foi utilizado Random Search [41], melhorando o AUC do modelo para 97.5748%.

Portanto, o modelo final é um Random Forest para classificação binária com um AUC de 97.5748%.

3.4.2 Artigo 2 - Analisando Métodos de Machine Learning e Avaliação de Risco de Crédito

No artigo de Coelho et al. [42] foram comparados diferentes métodos estatísticos na predição de inadimplência. Foram comparados os modelos: regressão logística, Random Forest e XGBoost.

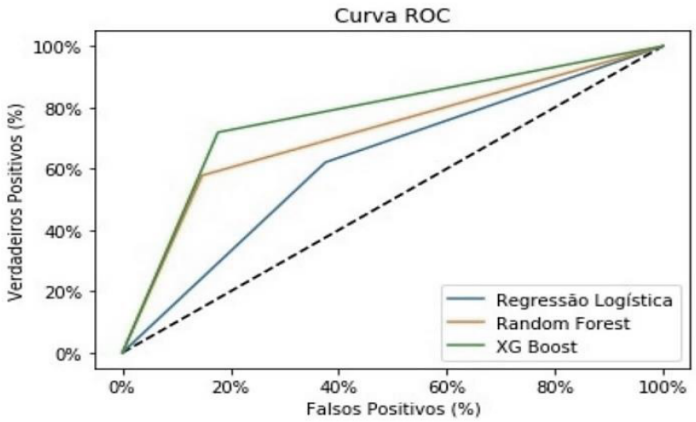
Foi utilizada uma base de dados exclusiva com informações de 3944 e 17800 observações de pequenas e médias empresas, clientes de uma locadora de automóveis com atuação em todo Brasil (de janeiro de 2018 a abril de 2019). Apenas dados de cliente que tiveram solicitação de crédito aprovado, como uma limitação, não foi possível analisar os clientes que foram reprovados.

Classificação binária em clientes Bons e Ruins. Amostra final com 3844 clientes, dos

quais 1492 eram classificados como Bons e 1492 eram classificados como Ruins. Foram considerados inadimplentes (Ruins) os clientes que ficaram com títulos em aberto em um prazo maior que 90 dias. Foram Utilizadas tanto variáveis financeiras como não financeiras, pois ambas são importantes para prever a inadimplência, particularmente, considerando uma amostra de pequenas e médias empresas. Para os testes, foi utilizada uma Base de treino com 2718 observações e base de teste de 1166 observações. Para medir o desempenho, foi utilizada acurácia, a curva ROC e a estatística Kolmorogov-Smirnov (KS) [43].

Os resultados podem ser vistos nas figuras 13 e 14.

Figura 13: Análise do artigo 2: Curva ROC dos modelos



Fonte: Print Retirado do Artigo de Coelho et al., 2021. [42]

Figura 14: Análise do artigo 2: Desempenho dos modelos (acurácia e KS)

Métodos	**KS**	**ACC**
Regressão logística	30,36%	63,02%
Random forest	47,00%	66,81%
XGBoost	57,12%	72,04%

Fonte: Print Retirado do Artigo de Coelho et al., 2021. [42]

Portanto, o modelo XGBoost possui a melhor performance, com uma capacidade de previsão de inadimplência maior em relação aos outros modelos de Regressão Logística e Random Forest.

3.4.3 Demais artigos

Os artigos de Hamori et al. [44] e Addo et al. [45] apresentam trabalhos semelhantes ao feito por Coelho et al [42].

O trabalho de Hamori et al. comparou o desempenho dos modelos de ensemble (bagging, random florest e boosting) e várias configurações de redes neurais (1 ou 2 camadas) com diferentes funções de ativação (Tanh e ReLU), com a utilização ou não de Dropout. Os modelos foram avaliados com a métrica de acurácia, curva ROC e F1 score.

Já o trabalho de Addo et al. comparou Regressão Logística, Random Forest, Gradient Boosting e 4 versões de rede neural com diferentes configurações. Aqui, os modelos foram avaliados pela AUC e RMSE.

Em ambos os trabalhos, os modelos de boosting apresentaram melhores perfomances, demonstrando que esses algoritmos geram ótimas previsões para dados estruturados, superando até as redes neurais.

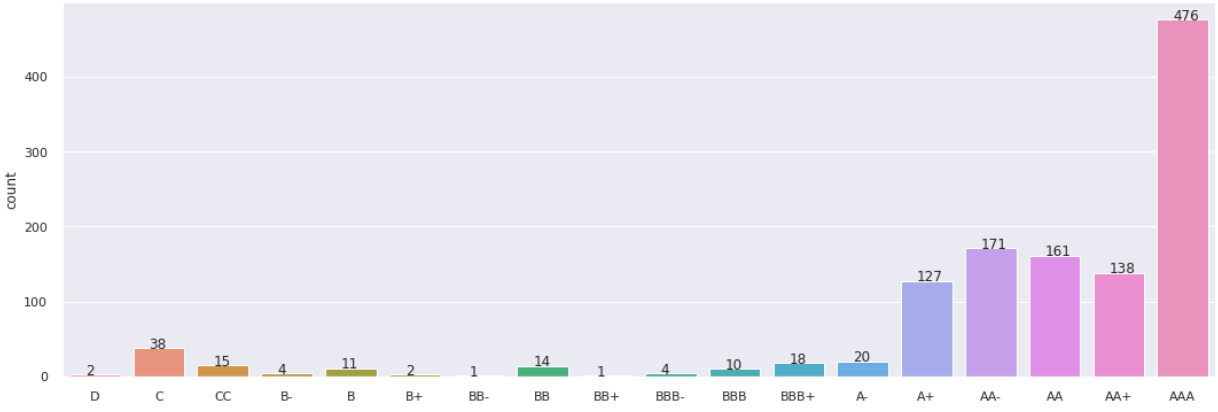
4 METODOLOGIA

4.1 Preparação de Dados

- **Coleta de dados:**
  - Bloomberg: Coleta de todo o histórico de ratings nacionais de companhias brasileiras, associados à emissão de debêntures, classificadas por *Standard & Poor's (S&P), Moody's e Fitch.*
  - Economatica: Coleta de todo o histórico dos demonstrativos financeiros de companhias brasileiras com alguma debênture emitida.
- **Join das tabelas:**

  Para fazer a junção das tabelas, foi feita a associação das companhias de cada dataset utilizando o ISIN(*International Securities Identification Number*) e a data da classificação e do demonstrativo financeiro, ou seja, é utilizado mais de uma observação da mesma companhia ao longo do tempo.
- **Observações**

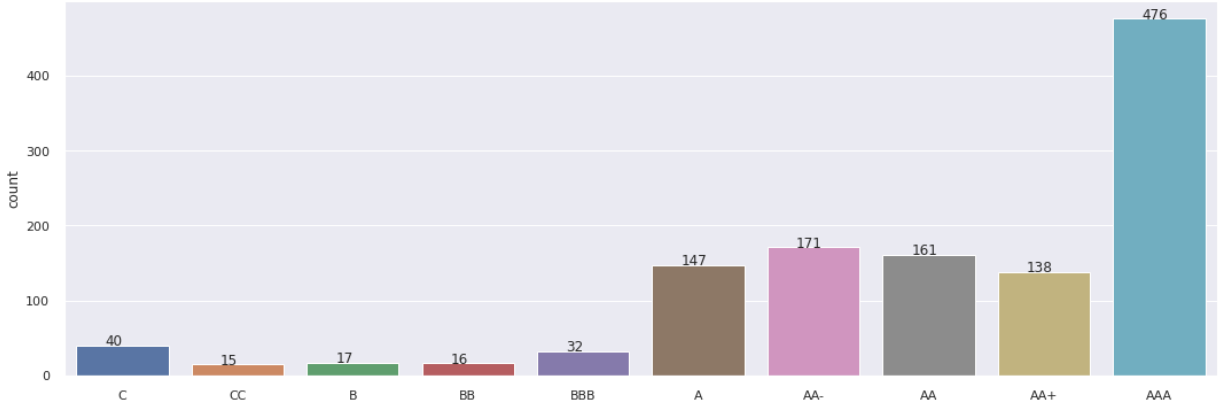
Figura 15: Distribuição dos Ratings



Fonte: Autoria própria. 2022

Como é visto na (Figura 15), há um grande desbalanceamento na distribuição das classes. Assim, para aumentar a quantidade de observações por classe, foram agrupados os ratings menos observados com níveis semelhantes de probabilidade de default, o que resulta em uma menor segmentação, sem perder, no entanto, a capacidade de classificação das companhias. O resultado pós agrupamento é visto na (Figura 16).

Figura 16: Agrupamento dos Ratings

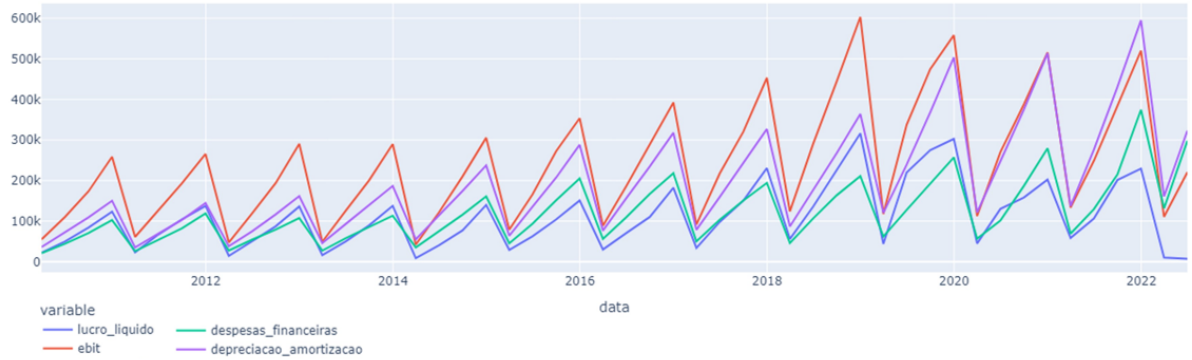


Fonte: Autoria própria. 2022

- **Remoção de sazonalidade:**

  Alguns dos demonstrativos financeiros, como, Lucro Líquido, EBIT, EBITDA, Despesas Financeiras, Depreciação e Amortização, são reportados de forma acumulada por trimestre, o que introduz uma variação artificial do indicador. Na imagem seguinte, é visto como essas variáveis se comportam:

Figura 17: Variáveis com efeitos de sazonalidade

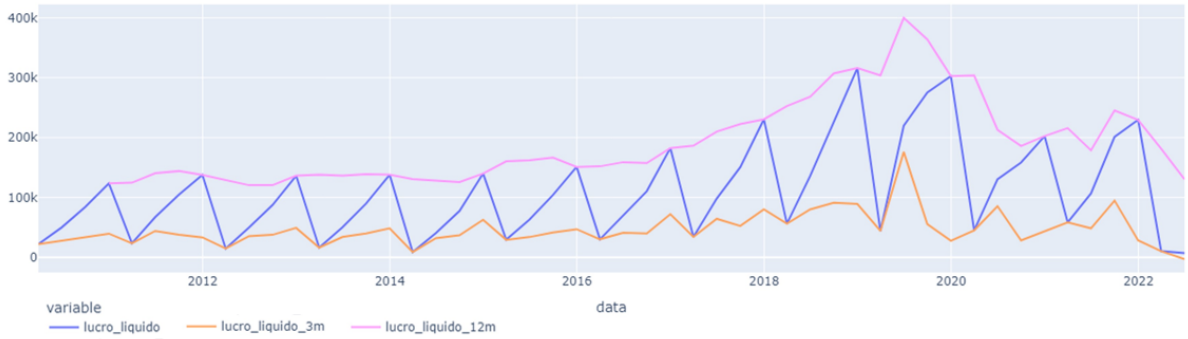


Fonte: Autoria própria. 2022

Para corrigir o efeito da sazonalidade, foi feita a discretização do indicador por trimestre e, posteriormente, a soma em janelas de quatro trimestres, gerando assim

uma série temporal com variação suave a cada trimestre. Na figura seguinte é visto os três gráficos, a curva azul é a evolução do lucro líquido no exercício, o gráfico laranja é indicador a cada trimestre, já a curva rosa representa a evolução da soma móvel dos últimos 4 trimestres.

Figura 18: Evolução do Lucro Líquido



Fonte: Autoria própria. 2022

- **Dados faltantes:**

  Ainda com os dados brutos, foi feito o primeiro tratamento dos dados faltantes esporádicos, ou seja, quando em uma coluna existe um dado faltante acompanhado dos dados dos demais trimestres. Nesses casos, é feita a média do trimestre imediatamente anterior e posterior ponderada pelo índice do trimestre.

- **Variável *Dummy*:**

  Foi observado que o setor de *utilities* possui um comportamento diferente dos demais setores, assim, para diferenciá-lo, foi criada uma variável *flag* chamada de *dummy_utilities*.

- ***Ratios*:**

Tabela 1: Descrição dos Ratios

Ratio	Interpretação
Passivo/Ativo	Proxy para a liquidez geral da companhia.
Despesas Financeiras/EBITDA	Proxy para a cobertura de juros.
Lucro Líquido/Ativo Total	Também chamado de ROA(Return on Asset), esse ratio representa o retorno líquido de uma companhia, em relação aos investimentos, atrelados aos seus ativos.
Lucro Líquido/Patrimônio Líquido	Também chamado de ROE(Return on Equity), esse ratio representa o retorno líquido de uma compahia em relação aos seus recursos próprios.
ROE/ROA	Também chamado de Alavancagem Financeira, esse ratio mede o quanto uma companhia utiliza do capital de terceiros para aumentar o seu retorno.
Lucro Líquido/Receita Líquida	Também chamado de Margem Líquida, esse ratio mede o quanto uma companhia obtém de lucro para cada unidade de receita.
Ativo Circulante/Passivo Circulante	Também chamado de Liquidez Corrente, esse ratio mede a capacidade da companhia de honrar suas dívidas de curtíssimo prazo.
EBIT/EBITDA	Esse ratio mede o impacto da Depreciação e Amortização de uma companhia sobre o seu lucro operacional.
Dívida Líquida/Patrimônio Líquido	Esse ratio mede o endividamento da companhia em relação aos seus recursos próprios.
EBITDA/Ativo Total	Esse ratio mede a capacidade de geração de lucro operacional da companhia com os ativos totais.
Lucro Líquido/EBITDA	Esse ratio mede a capacidade da companhia de converter lucro operacional em lucro líquido.

Fonte: Autoria própria. 2022

Após todo o tratamento, foram obtidos 1253 registros com 82 atributos mais a saída (rating) que se deseja prever.

4.2 Machine learning

A metodologia adotada baseia-se nos resultados empíricos de diversos testes, onde foram testados 8 modelos, desde os mais simples até os mais complexos. Também foram testadas diferentes técnicas de pré-processamento de dados para os seguintes propósitos: tratamento de valores faltantes, seleção de atributos, redução de dimensionalidade, escalonamento e tratamento de dados desbalanceados. Outro ponto explorado foi a utilização de Otimização Bayesiana para o tuning do modelo no lugar dos famosos e amplamente utilizados GridSearch e RandomizedSearch. Além disso, foram tomadas as devidas precauções para evitar o sobreajuste e superestimação do modelo, tais como a utilização de pipelines, validação cruzada e métricas de avaliação adequadas para garantir que o modelo final não sobreajuste e tenha boa capacidade de generalização.

Uma das precauções adotadas na metodologia aplicada é de considerar os dados desbalanceados, pois a acurácia pode ser uma estimativa superestimada de um modelo que não prevê adequadamente as classes minoritárias. A segunda precaução é em relação ao vazamento de dados de teste para o treinamento, gerando um modelo enviesado e superestimado ao avaliá-los por meio de validação cruzada. O primeiro problema é possível mitigar utilizando técnicas de amostragem estratificada, balanceamento de dados (sobreamostragem e subamostragem) e métricas que são sensíveis ao desbalanceamento dos dados, como o F1-score. O problema de vazamento de dados é resolvido utilizando pipelines, não permitindo o vazamento de dados do treinamento para o teste, gerando resultados com menor sobreajuste (overfitting).

4.3 Modelos

Os modelos e suas respectivas bibliotecas testadas podem ser vistos na tabela 2

Tabela 2: Modelos e bibliotecas

Modelo	Biblioteca
Árvore de decisão	sklearn.tree.DecisionTreeClassifier
KNN	sklearn.neighbors.KNeighborsClassifier
Regressão Logística	sklearn.linear_model.LogisticRegression
SVM	sklearn.svm.SVC
Rede Neural (MLP)	sklearn.neural_network.MLPClassifier
Random Forest	sklearn.ensemble.RandomForestClassifier
XGBoost	xgboost.XGBClassifier
LigthGBM	lightgbm.LGBMClassifier

Fonte: Autoria própria. 2022

4.4 Divisão base de treinamento e teste

Como a base de dados está desbalanceada, a divisão entre treinamento e teste foi realizada por meio de amostragem estratificada dos ratings, garantindo que tanto o treinamento como o teste tenham todas categorias e proporção da distribuição dos ratings da base de dados original. Foram separados 20% dos dados para teste. A biblioteca utilizada foi sklearn.model_selection.train_test_split, que possui o parâmetro "stratify"para realizar a divisão de forma estratificada.

4.5 Pipeline

Para evitar o vazamento dos dados de teste para o treinamento e manter os dados de teste o mais fiel possível aos dados reais, foram utilizados pipelines para a aplicação de todas as técnicas de aprendizagem de máquina para avaliação dos modelos por meio de validação cruzada. Utilizando pipeline, foram aplicados o preenchimento de valores faltantes, o escalonamento e o balanceamento de dados e, em seguida, os modelos são avaliados por validação cruzada. O pipeline faz os cálculos dos parâmetros (método fit) para o preenchimento de valor faltante (média, mediana, moda etc.) e para o escalonamento usando apenas os dados de treinamento e, em seguida, é aplicada a transformação em todos os dados (método transform), transformando toda base sem usar os dados de teste para o cálculo dos parâmetros, evitando vazamento de dados de teste para o treinamento. Além disso, o pipeline aplica as técnicas de balanceamento de dados apenas nos dados de treinamento, mantendo os dados de teste desbalanceados, permanecendo fiéis aos dados reais. A biblioteca utilizada foi a imblearn.pipeline.make_pipeline, pois permite a aplicação correta das técnicas de balanceamento de dados.

4.6 Validação Cruzada

Para a validação cruzada, utilizou-se 10 Folds e 3 repetições com diferentes aleatoriedades e com divisão estratificada. Dessa forma, foram gerados 30 modelos e a métrica

de avalição é calculada pela média dos 30 modelos, gerando um resultado com maior confiabilidade e efetividade para avaliar as classes desbalanceadas. Para aplicar esse método, foi utilizada a biblioteca sklearn.model_selection.RepeatedStratifiedKFold.

4.7 Técnicas de pré-processamento

Para conseguir a melhor performance, foram exploradas diferentes técnicas de pré-processamento da base de dados, essas técnicas podem ser divididas nas seguintes categorias:

- Tratamento de dados faltantes;
- Seleção de atributos;
- Redução de dimensionalidade;
- Escalonamento;
- Balanceamento de dados (Sobreamostragem e Subamostragem).

Cada categoria de técnicas é avaliada individualmente utilizando validação cruzada para cada modelo e a técnica que gerar a melhor performance foi selecionada, considerando acurácia e F1-Score macro.

4.7.1 Tratamento de valores faltantes

Para tratar os dados faltantes, foram exploradas as seguintes técnicas:

- Remover registros (linhas) com valores faltantes em qualquer um dos atributos (colunas);
- Preenchimento dos valores faltantes com a média (SimpleImputer());
- Preenchimento dos valores faltantes com a mediana (SimpleImputer(strategy='median'));
- Preenchimento dos valores faltantes com a moda (SimpleImputer(strategy='most_frequent'));
- Preenchimento dos valores faltantes com KNN (KNNImputer()).

As técnicas de preenchimento de valores faltantes foram feitas utilizando a biblioteca sklearn.impute.SimpleImputer e sklearn.impute.KNNImputer e a remoção dos registros com valores faltantes foi utilizando o método drop_na da biblioteca Pandas.

4.7.2 Seleção de atributos

Foram utilizadas duas técnicas para seleção de atributos: ExtraTree e Correlação.

O ExtraTrees é um modelo parecido com RandomForest que escolhe as variáveis e amostras randomicamente, e o threshold para spilt dos dados de maneira randômica. Após o treinamento do modelo, é possível ver a importância de cada atributo. A seleção de atributos por correlação é feita pela correlação das entradas (atributos) com a saída (rating). A importância de cada atributo pela ExtraTree e por correlação variam entre 0 e 1, sendo os atributos mais importantes os que apresentam os maiores valores. Para importar o ExtraTree, foi utilizado a biblioteca sklearn.ensemble.ExtraTreesClassifier e a correlação é calculada usando o método corr() da biblioteca Pandas.

4.7.3 Redução de Dimensionalidade

Para redução de dimensionalidade, foi explorada apenas a técnica PCA, pois esta é uma técnica eficiente e comumente utilizada para reduzir a dimensionalidade da base de dados. Utilizou-se a biblioteca sklearn.decomposition.PCA.

4.7.4 Escalonamento

Como alguns modelos são sensíveis à escala dos dados, foram testados as seguintes técnicas de escalonamento:

- MinMaxScaler;
- MaxAbsScaler;
- StandartScaler;
- RobustScaler;
- Normalizer;
- Quantile Transformer;
- Power Transformer.

Para aplicar as técnicas de escalonamento listadas, foi importada a biblioteca sklearn.preprocessing.

4.7.5 Balanceamento de dados

Foram testadas diferentes técnicas de para tratar o desbalanceamento dos dados. Os balanceamentos dos dados podem ser divididos em sobreamostragem, subamostragem e combinação, esse último se refere a aplicação de uma técnica de sobreamostragem seguido de uma subamostragem.

a) Sobreamostragem

- SMOTE
- ADASYN

b) Subamostragem

- Tomek Link
- AllKNN

c) Combinação

- SMOTETOMEK
- SMOTEENN

As técnicas de subamostragem, sobreamostragem e combinação foram aplicadas usando as bibliotecas imblearn.over_sampling, imblearn.under_sampling e imblearn.combine, respectivamente.

4.8 Combinação das melhores técnicas e modelos

As técnicas exploradas foram testadas separadamente por categoria, pois testar todas as combinações de técnicas exigiria um alto custo computacional. Logo, foram testadas as combinações das melhores técnicas, baseado nos testes individuais, para encontrar a combinação ótima.

4.9 Ajuste de parâmetros de técnicas de pré-processamento

Algumas técnicas de pré-processamento possuem parâmetros que podem ser ajustados e, portanto, após a seleção da melhor combinação de técnicas, é feito o ajuste, se possível, dos parâmetros de cada técnica selecionada. Por exemplo, a técnica SMOTE é uma técnica

de sobreamostragem que possui um parâmetro de número de vizinhos (k_neighbors) que pode ser ajustada.

4.10 Validação Cruzada dos melhores modelos e técnicas de pré-processamento

Em seguida, foi feito uma validação cruzada com as melhores técnicas de pré-processamento com seus parâmetros ajustados e apenas para os modelos que apresentaram as melhores performances.

4.11 Tuning do modelo

Logo depois, selecionou-se o melhor modelo, que tem seus hiperparâmetros otimizados por meio de Otimização Bayesiana, gerando o modelo final. A biblioteca importada foi a skopt.BayesSearchCV.

4.12 Avaliação Final do melhor modelo

Por fim, foi feita a avaliação final com a base de teste para validar a performance das melhores técnicas de pré-processamento e do melhor modelo. As métricas de avalição utilizadas foram a acurácia, precisão macro, revocação macro, F1-Score macro e matriz de confusão. As bibliotecas utilizadas foram yellowbrick.classifier.ConfusionMatrix para gerar a matriz de confusão e o sklearn.metrics para as demais métricas.

5 RESULTADOS E DISCUSSÕES

5.1 Tratamento de valores faltantes

Os resultados das diferentes técnicas de tratamento de valores faltantes podem ser vistos na tabela 3.

Tabela 3: Tratamento de valores faltantes

Imputer	Modelo	Acurácia	F1-Score
SimpleImputer(strategy='median')	RandomForestClassifier(random_state=42)	87.66%	81.01%
drop_na	RandomForestClassifier(random_state=42)	87.44%	82.74%
SimpleImputer()	RandomForestClassifier(random_state=42)	87.22%	80.05%
SimpleImputer(strategy='most_frequent')	RandomForestClassifier(random_state=42)	86.98%	80.24%
KNNImputer()	RandomForestClassifier(random_state=42)	86.46%	80.53%
SimpleImputer(strategy='median')	LGBMClassifier(random_state=42)	85.88%	77.01%
SimpleImputer(strategy='most_frequent')	LGBMClassifier(random_state=42)	85.67%	77.44%
SimpleImputer()	LGBMClassifier(random_state=42)	85.57%	76.59%
drop_na	LGBMClassifier(random_state=42)	85.25%	74.69%
KNNImputer()	LGBMClassifier(random_state=42)	85.09%	75.64%
drop_na	XGBClassifier(random_state=42)	80.85%	70.51%
SimpleImputer(strategy='most_frequent')	XGBClassifier(random_state=42)	80.34%	72.32%
KNNImputer()	XGBClassifier(random_state=42)	80.24%	70.74%
SimpleImputer()	XGBClassifier(random_state=42)	79.83%	70.07%
SimpleImputer(strategy='median')	XGBClassifier(random_state=42)	79.73%	69.84%
KNNImputer()	DecisionTreeClassifier(random_state=42)	76.53%	69.58%
drop_na	DecisionTreeClassifier(random_state=42)	75.59%	66.90%
SimpleImputer(strategy='most_frequent')	DecisionTreeClassifier(random_state=42)	75.53%	66.75%
SimpleImputer()	DecisionTreeClassifier(random_state=42)	75.40%	68.63%
SimpleImputer(strategy='median')	DecisionTreeClassifier(random_state=42)	75.33%	67.86%
drop_na	KNeighborsClassifier()	73.85%	63.60%
SimpleImputer(strategy='most_frequent')	KNeighborsClassifier()	72.78%	61.01%
SimpleImputer(strategy='median')	KNeighborsClassifier()	72.68%	60.59%
SimpleImputer()	KNeighborsClassifier()	72.51%	60.61%
KNNImputer()	KNeighborsClassifier()	72.47%	61.22%
drop_na	MLPClassifier(random_state=42)	46.46%	25.30%
SimpleImputer(strategy='median')	MLPClassifier(random_state=42)	45.95%	27.48%
SimpleImputer(strategy='most_frequent')	MLPClassifier(random_state=42)	45.02%	24.93%
drop_na	LogisticRegression(random_state=42)	44.54%	17.19%
SimpleImputer()	MLPClassifier(random_state=42)	44.02%	23.89%
KNNImputer()	MLPClassifier(random_state=42)	43.75%	24.83%
KNNImputer()	LogisticRegression(random_state=42)	42.92%	18.42%

SimpleImputer(strategy='most_frequent')	LogisticRegression(random_state=42)	42.68%	18.18%
drop_na	SVC(random_state=42)	41.96%	7.65%
SimpleImputer(strategy='median')	LogisticRegression(random_state=42)	41.51%	17.02%
SimpleImputer()	LogisticRegression(random_state=42)	41.03%	17.18%
SimpleImputer(strategy='most_frequent')	SVC(random_state=42)	39.69%	7.09%
SimpleImputer()	SVC(random_state=42)	39.42%	6.28%
SimpleImputer(strategy='median')	SVC(random_state=42)	39.38%	6.11%
KNNImputer()	SVC(random_state=42)	39.38%	6.11%

Fonte: Autoria própria. 2022

Analisando a tabela 3, é possível verificar que os modelos com melhor performance são: RandomForest e LGBM. Para o RandomForest, a técnica de remover os registros com valores faltantes tem o melhor equilíbrio entre acurácia e F1-Score, enquanto para o LGBM as melhores técnicas são mediana e moda.

Outro ponto importante a ser considerado é o fato de que remover os valores faltantes ocorre em maior parte para os ratings de classe minoritária, não sendo o ideal para previsão destas classes.

Para o teste das outras categorias de técnicas a seguir, será mantida a aplicação da técnica que apresentou a melhor acurácia, ou seja, preenchimento de valores faltantes com a mediana (SimpleImputer(strategy='median')).

5.2 Seleção de atributos

Como a base de dados tem alta dimensionalidade (82 atributos), é necessário reduzir o número de atributos para evitar o sobreajuste do modelo. Logo, foram utilizadas as duas técnicas para seleção de atributos: ExtraTree e Correlação. Portanto, são selecionados os atributos com importância ExtraTree maior ou igual a 0.012 e correlação maior ou igual a 0.1. Com isso, o número de atributos reduziu de 82 para 22 atributos.

5.3 Redução de Dimensionalidade

Os resultados da aplicação do PCA com diferentes parâmetros de número de componentes podem ser vistos na tabela 4.

Tabela 4: Redução de Dimensionalidade

Redução	Modelo	Acurácia	F1-Score
Nenhum	RandomForestClassifier(random_state=42)	87.66%	81.01%
PCA(n_components=20)	RandomForestClassifier(random_state=42)	86.12%	80.12%
Nenhum	LGBMClassifier(random_state=42)	85.88%	77.01%
PCA(n_components=20)	LGBMClassifier(random_state=42)	84.91%	76.09%
PCA(n_components=10)	RandomForestClassifier(random_state=42)	83.95%	79.71%
PCA(n_components=8)	RandomForestClassifier(random_state=42)	83.85%	79.02%
PCA(n_components=8)	LGBMClassifier(random_state=42)	82.23%	73.42%
PCA(n_components=10)	LGBMClassifier(random_state=42)	81.51%	72.69%
Nenhum	XGBClassifier(random_state=42)	79.73%	69.84%
PCA(n_components=20)	XGBClassifier(random_state=42)	78.38%	68.76%
Nenhum	DecisionTreeClassifier(random_state=42)	75.33%	67.86%
PCA(n_components=10)	XGBClassifier(random_state=42)	73.33%	67.17%
PCA(n_components=8)	KNeighborsClassifier()	72.82%	60.68%
Nenhum	KNeighborsClassifier()	72.68%	60.59%
PCA(n_components=20)	KNeighborsClassifier()	72.68%	60.59%
PCA(n_components=10)	KNeighborsClassifier()	72.65%	60.57%
PCA(n_components=8)	XGBClassifier(random_state=42)	72.27%	66.70%
PCA(n_components=20)	DecisionTreeClassifier(random_state=42)	71.51%	63.01%
PCA(n_components=10)	DecisionTreeClassifier(random_state=42)	70.07%	60.12%
PCA(n_components=8)	DecisionTreeClassifier(random_state=42)	69.48%	63.91%
Nenhum	MLPClassifier(random_state=42)	45.95%	27.48%
PCA(n_components=20)	MLPClassifier(random_state=42)	44.67%	25.67%
PCA(n_components=10)	MLPClassifier(random_state=42)	43.20%	25.19%
PCA(n_components=8)	MLPClassifier(random_state=42)	42.78%	24.22%
Nenhum	LogisticRegression(random_state=42)	41.51%	17.02%
PCA(n_components=8)	SVC(random_state=42)	40.03%	9.17%
PCA(n_components=10)	SVC(random_state=42)	40.03%	9.17%
PCA(n_components=20)	SVC(random_state=42)	40.03%	9.17%
Nenhum	SVC(random_state=42)	39.38%	6.11%
PCA(n_components=20)	LogisticRegression(random_state=42)	33.57%	19.45%
PCA(n_components=10)	LogisticRegression(random_state=42)	33.54%	19.41%
PCA(n_components=8)	LogisticRegression(random_state=42)	33.51%	19.45%

Fonte: Autoria própria. 2022

Observando os resultados da tabela 4, não foi aplicada redução de dimensionalidade PCA.

5.4 Escalonamento

Os testes das diferentes técnicas de escalonamento podem ser vistos na tabela 5.

Tabela 5: Escalonamento

Escalonamento	Modelo	Acurácia	F1-Score
QuantileTransformer()	RandomForestClassifier(random_state=42)	87.73%	81.80%
MinMaxScaler()	RandomForestClassifier(random_state=42)	87.70%	81.36%
Nenhum	RandomForestClassifier(random_state=42)	87.66%	81.01%
MaxAbsScaler()	RandomForestClassifier(random_state=42)	87.66%	81.01%
StandardScaler()	RandomForestClassifier(random_state=42)	87.66%	81.01%
RobustScaler()	RandomForestClassifier(random_state=42)	87.66%	81.01%
PowerTransformer()	RandomForestClassifier(random_state=42)	87.59%	81.19%
PowerTransformer()	LGBMClassifier(random_state=42)	85.91%	76.81%
Nenhum	LGBMClassifier(random_state=42)	85.88%	77.01%
MaxAbsScaler()	LGBMClassifier(random_state=42)	85.88%	77.01%
MinMaxScaler()	LGBMClassifier(random_state=42)	85.77%	76.53%
QuantileTransformer()	LGBMClassifier(random_state=42)	85.77%	76.53%
StandardScaler()	LGBMClassifier(random_state=42)	85.57%	76.39%
Normalizer()	RandomForestClassifier(random_state=42)	85.43%	80.68%
RobustScaler()	LGBMClassifier(random_state=42)	85.29%	76.87%
Normalizer()	LGBMClassifier(random_state=42)	84.36%	74.78%
MinMaxScaler()	XGBClassifier(random_state=42)	79.73%	69.84%
Nenhum	XGBClassifier(random_state=42)	79.73%	69.84%
MaxAbsScaler()	XGBClassifier(random_state=42)	79.73%	69.84%
StandardScaler()	XGBClassifier(random_state=42)	79.73%	69.84%
RobustScaler()	XGBClassifier(random_state=42)	79.73%	69.84%
PowerTransformer()	XGBClassifier(random_state=42)	79.69%	69.82%
QuantileTransformer()	XGBClassifier(random_state=42)	79.69%	69.99%
Normalizer()	XGBClassifier(random_state=42)	79.24%	72.90%
MaxAbsScaler()	KNeighborsClassifier()	76.67%	62.93%
QuantileTransformer()	KNeighborsClassifier()	76.19%	63.75%
StandardScaler()	KNeighborsClassifier()	75.88%	61.23%
PowerTransformer()	KNeighborsClassifier()	75.84%	60.83%
Nenhum	DecisionTreeClassifier(random_state=42)	75.33%	67.86%
MinMaxScaler()	DecisionTreeClassifier(random_state=42)	75.33%	67.86%
MaxAbsScaler()	DecisionTreeClassifier(random_state=42)	75.33%	67.86%
StandardScaler()	DecisionTreeClassifier(random_state=42)	75.33%	67.86%
RobustScaler()	DecisionTreeClassifier(random_state=42)	75.33%	67.86%
QuantileTransformer()	DecisionTreeClassifier(random_state=42)	75.29%	69.09%
PowerTransformer()	DecisionTreeClassifier(random_state=42)	75.29%	67.76%
MinMaxScaler()	KNeighborsClassifier()	74.98%	63.34%
Nenhum	KNeighborsClassifier()	72.68%	60.59%
RobustScaler()	KNeighborsClassifier()	72.41%	56.38%
Normalizer()	KNeighborsClassifier()	71.48%	60.49%
PowerTransformer()	MLPClassifier(random_state=42)	70.41%	59.19%
Normalizer()	DecisionTreeClassifier(random_state=42)	69.97%	63.22%
RobustScaler()	MLPClassifier(random_state=42)	68.83%	51.19%
QuantileTransformer()	SVC(random_state=42)	68.04%	46.95%
StandardScaler()	MLPClassifier(random_state=42)	66.98%	51.12%
QuantileTransformer()	MLPClassifier(random_state=42)	62.68%	44.67%
PowerTransformer()	SVC(random_state=42)	54.98%	33.04%
StandardScaler()	SVC(random_state=42)	53.37%	31.04%
PowerTransformer()	LogisticRegression(random_state=42)	52.23%	36.13%
StandardScaler()	LogisticRegression(random_state=42)	51.48%	35.07%
MaxAbsScaler()	MLPClassifier(random_state=42)	50.48%	27.27%

RobustScaler()	LogisticRegression(random_state=42)	49.18%	33.11%
QuantileTransformer()	LogisticRegression(random_state=42)	48.01%	24.04%
Nenhum	MLPClassifier(random_state=42)	45.95%	27.48%
MinMaxScaler()	LogisticRegression(random_state=42)	43.99%	11.03%
MinMaxScaler()	MLPClassifier(random_state=42)	43.99%	15.98%
MinMaxScaler()	SVC(random_state=42)	42.85%	10.52%
MaxAbsScaler()	LogisticRegression(random_state=42)	42.44%	11.21%
MaxAbsScaler()	SVC(random_state=42)	42.16%	12.10%
RobustScaler()	SVC(random_state=42)	41.62%	12.88%
Normalizer()	MLPClassifier(random_state=42)	41.58%	18.52%
Nenhum	LogisticRegression(random_state=42)	41.51%	17.02%
Normalizer()	SVC(random_state=42)	40.03%	9.82%
Nenhum	SVC(random_state=42)	39.38%	6.11%
Normalizer()	LogisticRegression(random_state=42)	39.38%	6.77%

Fonte: Autoria própria. 2022

Analisando a tabela 5, para os algoritmos baseados em árvore, não há uma diferença significativa de performance, pois esses modelos não são sensíveis à escala. Entretanto, como o QuantileTransformer e PowerTransformer apresentou uma pequena melhora de performance, serão testados os escalonamentos "QuantileTransformer", "PowerTransformer"e "Nenhum"(sem escalonamento).

Para o teste das técnicas de balanceamento a seguir, foi mantida a aplicação da técnica que apresentou a melhor acurácia, ou seja, QuantileTransformer.

5.5 Balanceamento de dados

A tabela 6 apresenta os resultados das técnicas para tratamento de dados desbalanceados.

Tabela 6: Balanceamento

Balanceamento	Modelo	Acurácia	F1-Score
Nenhum	RandomForestClassifier()	87.73%	81.80%
SMOTE(random_state=42, sampling_strategy='not majority')	RandomForestClassifier()	87.32%	80.70%
ADASYN(random_state=42, sampling_strategy='not majority')	RandomForestClassifier()	87.32%	78.28%
SMOTETomek(random_state=42, sampling_strategy='not majority')	RandomForestClassifier()	86.98%	79.76%
SMOTE(random_state=42, sampling_strategy='not majority')	LGBMClassifier()	86.15%	77.71%
SMOTETomek(random_state=42, sampling_strategy='not majority')	LGBMClassifier()	85.95%	77.39%
TomekLinks(sampling_strategy='all')	RandomForestClassifier()	85.88%	79.96%
ADASYN(random_state=42, sampling_strategy='not majority')	LGBMClassifier()	85.77%	77.39%
Nenhum	LGBMClassifier()	85.77%	76.53%
TomekLinks(sampling_strategy='all')	LGBMClassifier()	83.92%	73.89%
AllKNN(sampling_strategy='all')	RandomForestClassifier()	81.48%	69.36%
ADASYN(random_state=42, sampling_strategy='not majority')	KNeighborsClassifier()	81.00%	75.14%

SMOTEENN(random_state=42, sampling_strategy='not majority')	RandomForestClassifier()	80.76%	76.53%
AllKNN(sampling_strategy='all')	LGBMClassifier()	80.07%	66.60%
SMOTE(random_state=42, sampling_strategy='not majority')	KNeighborsClassifier()	79.93%	74.74%
SMOTETomek(random_state=42, sampling_strategy='not majority')	KNeighborsClassifier()	79.90%	74.81%
Nenhum	XGBClassifier()	79.69%	69.99%
SMOTEENN(random_state=42, sampling_strategy='not majority')	LGBMClassifier()	79.11%	72.21%
TomekLinks(sampling_strategy='all')	XGBClassifier()	79.00%	69.05%
SMOTETomek(random_state=42, sampling_strategy='not majority')	XGBClassifier()	77.70%	73.35%
SMOTE(random_state=42, sampling_strategy='not majority')	XGBClassifier()	77.53%	73.51%
ADASYN(random_state=42, sampling_strategy='not majority')	XGBClassifier()	76.43%	69.33%
Nenhum	KNeighborsClassifier()	76.19%	63.75%
AllKNN(sampling_strategy='all')	XGBClassifier()	76.01%	63.85%
TomekLinks(sampling_strategy='all')	DecisionTreeClassifier()	75.64%	66.91%
SMOTEENN(random_state=42, sampling_strategy='not majority')	KNeighborsClassifier()	75.40%	72.22%
TomekLinks(sampling_strategy='all')	KNeighborsClassifier()	75.36%	62.60%
Nenhum	DecisionTreeClassifier()	75.29%	69.09%
ADASYN(random_state=42, sampling_strategy='not majority')	DecisionTreeClassifier()	74.23%	64.63%
SMOTEENN(random_state=42, sampling_strategy='not majority')	XGBClassifier()	74.16%	70.74%
SMOTETomek(random_state=42, sampling_strategy='not majority')	DecisionTreeClassifier()	73.99%	66.07%
SMOTE(random_state=42, sampling_strategy='not majority')	DecisionTreeClassifier()	73.88%	66.60%
SMOTE(random_state=42, sampling_strategy='not majority')	SVC()	72.37%	67.18%
SMOTETomek(random_state=42, sampling_strategy='not majority')	SVC()	72.20%	67.04%
AllKNN(sampling_strategy='all')	KNeighborsClassifier()	72.20%	55.33%
AllKNN(sampling_strategy='all')	DecisionTreeClassifier()	71.24%	60.91%
ADASYN(random_state=42, sampling_strategy='not majority')	SVC()	70.27%	63.52%
SMOTEENN(random_state=42, sampling_strategy='not majority')	DecisionTreeClassifier()	68.59%	61.97%
TomekLinks(sampling_strategy='all')	SVC()	68.04%	47.08%
Nenhum	SVC(random_state=42)	68.04%	46.95%
SMOTETomek(random_state=42, sampling_strategy='not majority')	MLPClassifier()	67.94%	64.19%
SMOTEENN(random_state=42, sampling_strategy='not majority')	SVC()	67.42%	64.03%
AllKNN(sampling_strategy='all')	SVC()	66.70%	45.89%
SMOTE(random_state=42, sampling_strategy='not majority')	MLPClassifier()	66.39%	63.35%
SMOTEENN(random_state=42, sampling_strategy='not majority')	MLPClassifier()	65.40%	62.42%
ADASYN(random_state=42, sampling_strategy='not majority')	MLPClassifier()	64.26%	60.80%
Nenhum	MLPClassifier()	62.68%	44.67%
AllKNN(sampling_strategy='all')	MLPClassifier()	62.16%	43.58%
TomekLinks(sampling_strategy='all')	MLPClassifier()	61.27%	43.00%
AllKNN(sampling_strategy='all')	LogisticRegression()	48.42%	23.73%
Nenhum	LogisticRegression()	48.01%	24.04%
TomekLinks(sampling_strategy='all')	LogisticRegression()	47.97%	23.92%
SMOTETomek(random_state=42, sampling_strategy='not majority')	LogisticRegression()	42.65%	39.67%
SMOTE(random_state=42, sampling_strategy='not majority')	LogisticRegression()	42.41%	39.60%
ADASYN(random_state=42, sampling_strategy='not majority')	LogisticRegression()	42.16%	35.50%
SMOTEENN(random_state=42, sampling_strategy='not majority')	LogisticRegression()	41.27%	39.09%

Fonte: Autoria própria. 2022

Baseado na tabela 6, as técnicas de sobreamostragem aumentam o F1-Score de alguns modelos, entretanto, reduz levemente a acurácia.

5.6 Combinação das melhores técnicas e modelos

As técnicas por categoria a serem combinadas podem ser vistas na lista abaixo:

- Preenchimento de valores faltantes: SimpleImputer(strategy='median'), SimpleImputer(strategy='most_frequent');

- Redução: Nenhum;
- Escalonamento: Nenhum, QuantileTransformer, PowerTransformer;
- Balanceamento: Nenhum, SMOTE.

Os modelos Random Forest e LGBM apresentaram performance significativamente melhor que os demais modelos e, portanto, foram testados apenas esses dois modelos. Os resultados da melhor combinação de técnicas podem ser vistos nas tabelas 7 e 8.

Tabela 7: Combinação ótima de técnicas de pré-processamento - Random Forest

Imputer	Escalonamento	Balanceamento	Acurácia	F1-Score
SimpleImputer(strategy='median')	QuantileTransformer()	Nenhum	87.73%	81.80%
SimpleImputer(strategy='median')	Nenhum	Nenhum	87.66%	81.01%
SimpleImputer(strategy='median')	Nenhum	SMOTE()	87.63%	81.42%
SimpleImputer(strategy='median')	PowerTransformer()	Nenhum	87.59%	81.19%
SimpleImputer(strategy='median')	QuantileTransformer()	SMOTE()	87.32%	80.70%
SimpleImputer(strategy='most_frequent')	Nenhum	SMOTE()	87.18%	81.07%
SimpleImputer(strategy='most_frequent')	PowerTransformer()	SMOTE()	87.08%	80.30%
SimpleImputer(strategy='most_frequent')	QuantileTransformer()	Nenhum	86.98%	80.09%
SimpleImputer(strategy='most_frequent')	Nenhum	Nenhum	86.98%	80.24%
SimpleImputer(strategy='most_frequent')	PowerTransformer()	Nenhum	86.94%	80.22%
SimpleImputer(strategy='median')	PowerTransformer()	SMOTE()	86.80%	80.25%
SimpleImputer(strategy='most_frequent')	QuantileTransformer()	SMOTE()	86.70%	80.09%

Fonte: Autoria própria. 2022

Tabela 8: Combinação ótima de técnicas de pré-processamento - LightGBM

Imputer	Escalonamento	Balanceamento	Acurácia	F1-Score
SimpleImputer(strategy='median')	QuantileTransformer()	SMOTE()	86.15%	77.71%
SimpleImputer(strategy='median')	PowerTransformer()	Nenhum	85.91%	76.81%
SimpleImputer(strategy='most_frequent')	PowerTransformer()	SMOTE()	85.88%	78.06%
SimpleImputer(strategy='median')	Nenhum	Nenhum	85.88%	77.01%
SimpleImputer(strategy='median')	QuantileTransformer()	Nenhum	85.77%	76.53%
SimpleImputer(strategy='most_frequent')	Nenhum	Nenhum	85.67%	77.44%
SimpleImputer(strategy='median')	PowerTransformer()	SMOTE()	85.64%	77.63%
SimpleImputer(strategy='most_frequent')	PowerTransformer()	Nenhum	85.53%	76.68%
SimpleImputer(strategy='median')	Nenhum	SMOTE()	85.29%	76.42%
SimpleImputer(strategy='most_frequent')	QuantileTransformer()	Nenhum	85.19%	75.39%
SimpleImputer(strategy='most_frequent')	Nenhum	SMOTE()	85.09%	76.68%
SimpleImputer(strategy='most_frequent')	QuantileTransformer()	SMOTE()	84.57%	75.46%

Fonte: Autoria própria. 2022

5.7 Tuning do Balanceador (SMOTE)

A técnica de SMOTE apresentou uma melhora do F1-Score em alguns casos e, portanto, foram ajustados os parâmetros do SMOTE, testando a combinação dos valores da lista abaixo:

- sampling_strategy = ['minority', 'not minority', 'not majority', 'all']
- k_neighbors = [1,2,3,4,5,6,7,8]
- random_state = 42

Como o Random Forest possui uma ótima performance e um tempo de treinamento relativamente baixo, foi utilizado apenas esse modelo para o tuning do SMOTE. A tabela 9 apresenta os resultados do tuning do SMOTE.

Tabela 9: Tuning do SMOTE com RandomForest

Balanceador	Acurácia	F1-Score
SMOTE(k_neighbors=1, random_state=42, sampling_strategy='not majority')	87.42%	81.47%
SMOTE(k_neighbors=1, random_state=42, sampling_strategy='all')	87.42%	81.47%
SMOTE(k_neighbors=2, random_state=42, sampling_strategy='not minority')	87.35%	81.06%
SMOTE(k_neighbors=4, random_state=42, sampling_strategy='not majority')	87.35%	80.49%
SMOTE(k_neighbors=4, random_state=42, sampling_strategy='all')	87.35%	80.49%
SMOTE(random_state=42, sampling_strategy='not majority')	87.32%	80.70%
SMOTE(random_state=42, sampling_strategy='all')	87.32%	80.70%
SMOTE(k_neighbors=1, random_state=42, sampling_strategy='not minority')	87.29%	81.07%
SMOTE(k_neighbors=1, random_state=42, sampling_strategy='minority')	87.25%	80.17%
SMOTE(k_neighbors=8, random_state=42, sampling_strategy='not majority')	87.25%	79.63%
SMOTE(k_neighbors=8, random_state=42, sampling_strategy='all')	87.25%	79.63%
SMOTE(k_neighbors=8, random_state=42, sampling_strategy='not minority')	87.22%	77.62%
SMOTE(k_neighbors=2, random_state=42, sampling_strategy='not majority')	87.15%	80.86%
SMOTE(k_neighbors=2, random_state=42, sampling_strategy='all')	87.15%	80.86%
SMOTE(k_neighbors=3, random_state=42, sampling_strategy='not majority')	87.15%	78.53%
SMOTE(k_neighbors=3, random_state=42, sampling_strategy='all')	87.15%	78.53%
SMOTE(k_neighbors=3, random_state=42, sampling_strategy='minority')	87.11%	79.56%
SMOTE(k_neighbors=6, random_state=42, sampling_strategy='not majority')	87.08%	79.82%
SMOTE(k_neighbors=6, random_state=42, sampling_strategy='all')	87.08%	79.82%
SMOTE(random_state=42, sampling_strategy='not minority')	87.01%	79.14%
SMOTE(k_neighbors=4, random_state=42, sampling_strategy='minority')	86.94%	79.85%
SMOTE(random_state=42, sampling_strategy='minority')	86.94%	79.59%
SMOTE(k_neighbors=2, random_state=42, sampling_strategy='minority')	86.87%	79.46%
SMOTE(k_neighbors=4, random_state=42, sampling_strategy='not minority')	86.84%	79.05%
SMOTE(k_neighbors=6, random_state=42, sampling_strategy='minority')	86.84%	79.57%
SMOTE(k_neighbors=3, random_state=42, sampling_strategy='not minority')	86.80%	78.18%
SMOTE(k_neighbors=6, random_state=42, sampling_strategy='not minority')	86.67%	78.80%
SMOTE(k_neighbors=7, random_state=42, sampling_strategy='not majority')	86.63%	79.26%

SMOTE(k_neighbors=7, random_state=42, sampling_strategy='all')	86.63%	79.26%
SMOTE(k_neighbors=7, random_state=42, sampling_strategy='not minority')	86.63%	77.65%
SMOTE(k_neighbors=7, random_state=42, sampling_strategy='minority')	86.43%	79.65%
SMOTE(k_neighbors=8, random_state=42, sampling_strategy='minority')	86.43%	79.34%

Fonte: Autoria própria. 2022

As linhas com destaque em amarelo na tabela 9 apresentou o melhor resultado, foi utilizado a primeira, pois realiza menos resample, alterando menos a base.

5.8 Validação Cruzada dos melhores modelos e técnicas de pré-processamento

A validação cruzada dos melhores modelos com a melhor combinação de técnicas de pré-processamento ajustadas pode ser visto na tabela 10. Foram testados apenas os modelos LigthGBM e RandomForest, pois apresentaram os melhores resultados. As técnicas de pré-processamento utilizadas podem ser vistas na lista a seguir:

- Tratamento de valores faltantes: SimpleImputer(strategy='median');
- Redução: Nenhum;
- Escalonamento: QuantileTransformer();
- Balanceamento: SMOTE(k_neighbors=1, random_state=42, sampling_strategy='not majority').

Tabela 10: Validação Cruzada dos melhores modelos com pré-processamento tunado

Modelo	Acurácia	F1-Score
RandomForestClassifier(random_state=42)	87.42%	81.47%
LGBMClassifier(random_state=42)	86.56%	78.45%

Fonte: Autoria própria. 2022

5.9 Tuning dos modelos

O modelo Random Forest obteve a melhor performance. Logo, este modelo foi tunado utilizando Otimização Bayesiana para encontrar a melhor combinação de hiperparâmetros. Os valores de hiperparâmetros avaliados podem ser vistos na lista abaixo:

- 'bootstrap': [True]
- 'n_estimators': Integer(40,1000)
- 'max_samples': [None, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]
- random_state: 42

Além disso, são utilizadas as melhores técnicas de pré-processamento da base encontradas anteriormente:

- Tratamento de valores faltantes: SimpleImputer(strategy='median');
- Redução de Dimensionalidade: Nenhum;
- Escalonamento: QuantileTransformer;
- Balanceamento: SMOTE(k_neighbors=1, random_state=42, sampling_strategy='not majority').

O resultado do tuning em comparação ao modelo sem ajuste de hiperparâmetros pode ser visto na tabela 11.

Tabela 11: Resultado do tuning

Modelo	Acurácia	F1-Score
RandomForestClassifier(random_state=42)	87.42%	81.47%
RandomForestClassifier(max_samples=0.8, n_estimators=147, random_state=42)	87.53%	81.86%

Fonte: Autoria própria. 2022

O modelo apresentou um leve aumento de performance após o tuning, tanto em acurácia quanto em F1-Score.

5.10 Avaliação Final do melhor modelo

Por fim, foi feito o teste com a base de teste para validar a performance do melhor modelo. As técnicas de pré-processamento e o modelo final podem ser vistos em 12. Os resultados da avaliação com a base de teste podem ser vistos na tabela 13 e na matriz de confusão 19.

Tabela 12: Modelo final e Melhores técnicas de pré-processamento

Modelo
RandomForestClassifier(max_samples=0.8, n_estimators=147, random_state=42)
Seleção de Atributos
Extratree e Correlação
Preenchimento de valores faltantes
Mediana (SimpleImputer(strategy='median'))
Escalonador
QuantileTransformer()
Sobreamostragem
SMOTE(k_neighbors=1, random_state=42, sampling_strategy='not majority')

Fonte: Autoria própria. 2022

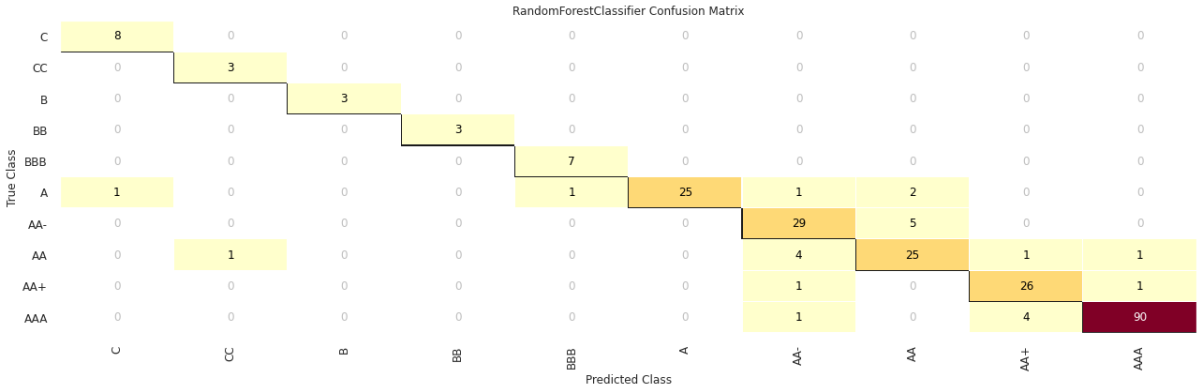
Tabela 13: Resultados da avaliação com a base de teste

Acurácia	**Precisão**	**Revocação**	**F1-Score**
90.12%	89.18%	93.43%	90.94%

Fonte: Autoria própria. 2022

Como pode ser visto na tabela 13, o modelo final apresentou um bom desempenho na classificação tanto das classes majoritárias, quanto das minoritárias.

Figura 19: Matriz de Confusão da avaliação com a base de teste

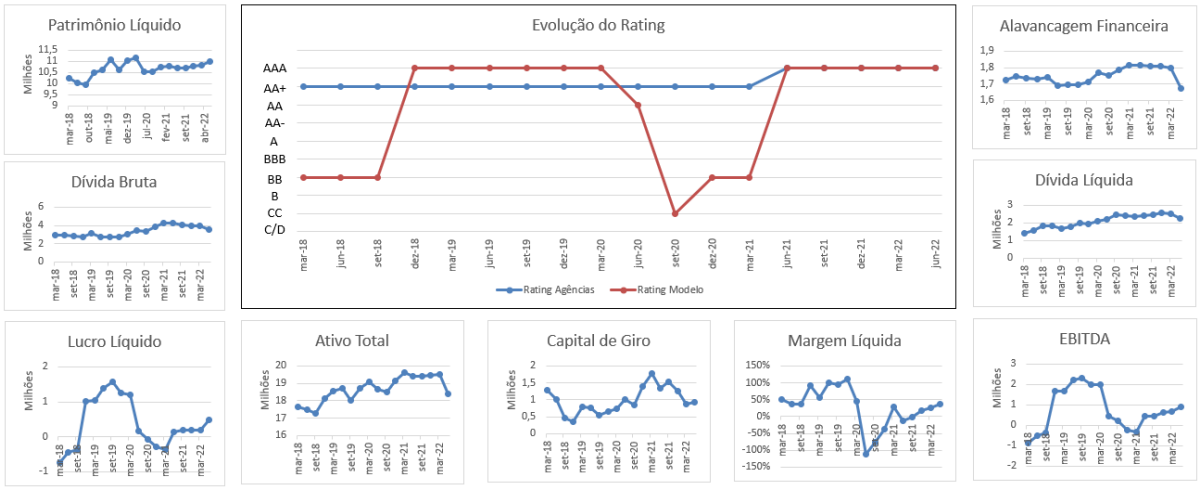


Fonte: Autoria Própria, 2022.

Na figura 19, é visto a matriz de confusão final, que mostra que o modelo tem uma boa performance na classificação. Além disso, pode-se observar que a maior parte dos erros são inferiores a três notches de distância. Os casos em que a distância foi superior a três notches são apresentados nas figuras 20 e 21, bem como a justificativa para a discrepância entre a classificação das agências e o modelo.

Nas figuras seguintes, são apresentados os gráficos da evolução do rating de algumas companhias obtidos tanto pelas agências de classificação, quanto pelo modelo final, juntamente com a evolução de alguns dos demonstrativos financeiros.

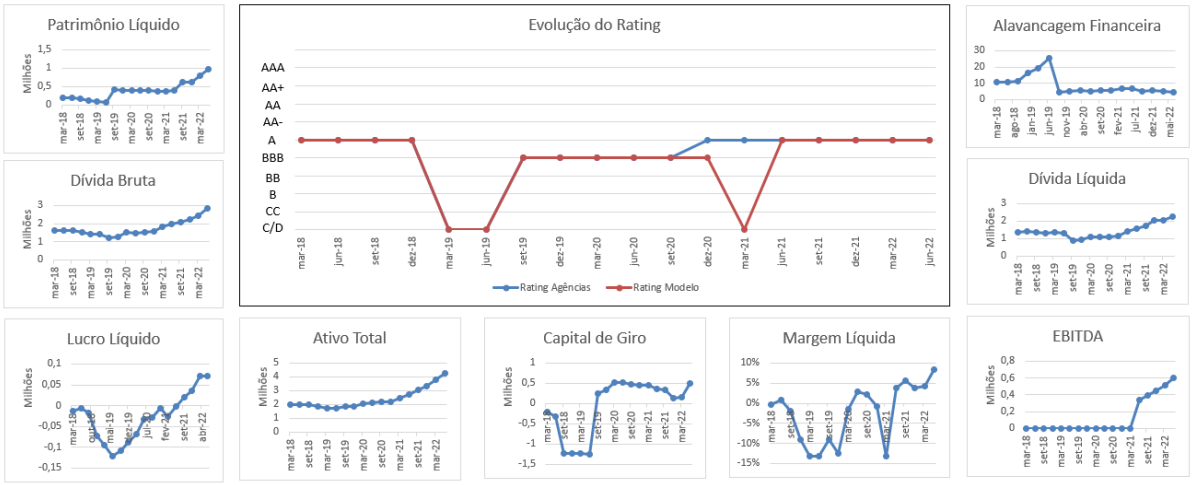
Figura 20: Evolução do rating de uma companhia do setor de shopping centers



Fonte: Autoria Própria, 2022.

Na figura 20, é visto como o modelo é sensível às alterações dos demonstrativos financeiros, enquanto as agências mantiveram a classificação estável mesmo depois da deterioração de alguns dos índices. Pode-se observar que o modelo aumenta o score da companhia de BB para AAA em dez/2018, mesmo período em que há um grande aumento do Lucro Líquido, EBITDA, Margem Líquida e redução do Alavancagem Financeira. Do mesmo modo, nota-se uma grande redução no rating, que vai de AAA em mar/2020 para CC em set/2020, em resposta à grave deterioração da Margem Líquida, Lucro Líquido e EBITDA, que pode explicada pelo fechamento dos shoppings para conter o avanço do coronavírus em 2020, apesar disso, as agências mantiveram a classificação estável.

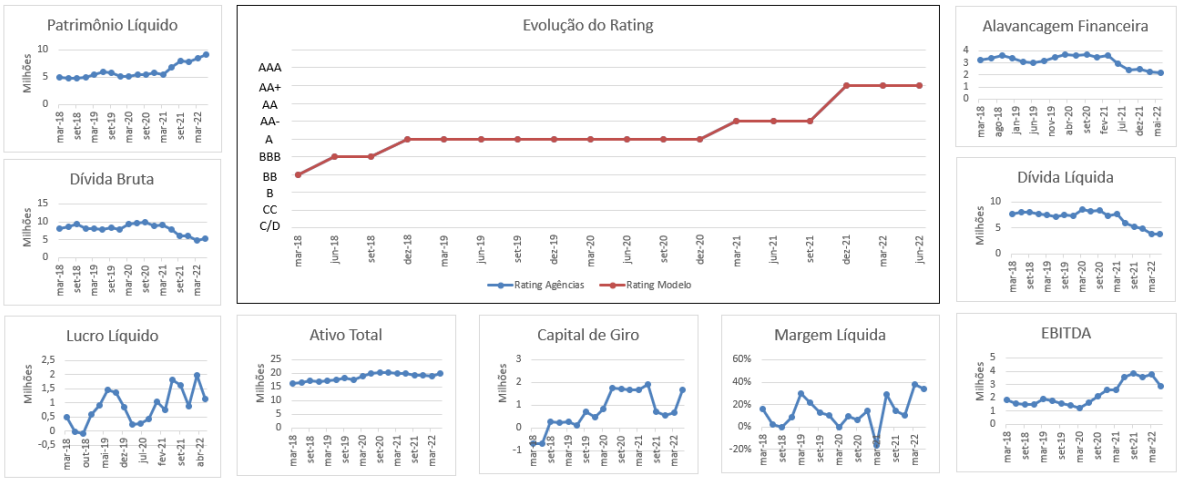
Figura 21: Evolução do rating de uma companhia do setor de locação de veículos



Fonte: Autoria Própria, 2022.

Na figura 21, pode ser observado como as agências e modelo caminharam juntos até set/2020, no entanto, em set/2021 as agências aumentam a classificação, enquanto o modelo reduz o rating de BBB para C, acompanhando a Margem Líquida que ficou negativa no período.

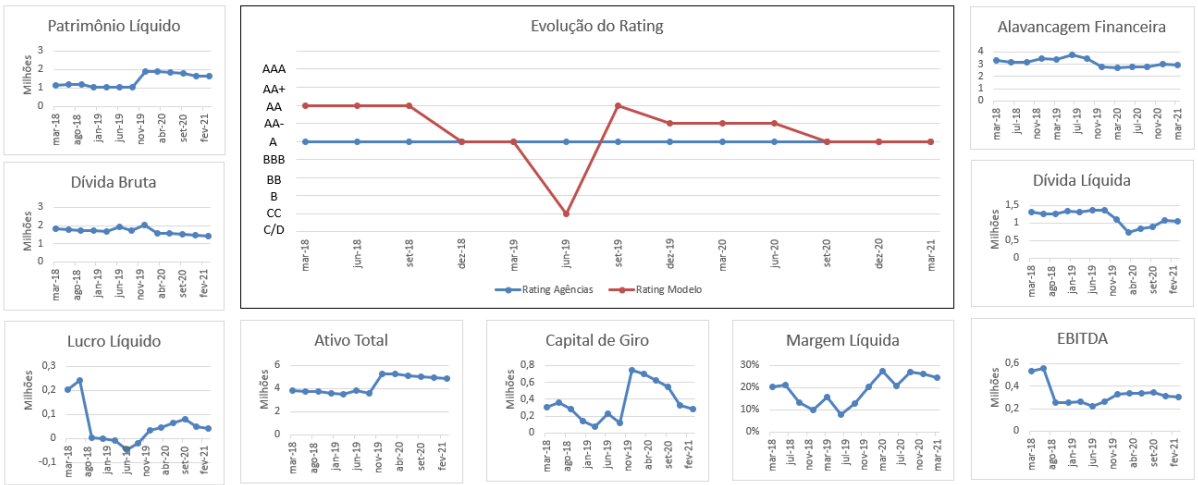
Figura 22: Evolução do rating de uma companhia do setor de energia elétrica



Fonte: Autoria Própria, 2022.

Na imagem 22, pode ser visto que tanto as agências, quanto o modelo tiveram o mesmo comportamento, partindo de BB para AA+, refletindo a melhora na saúde financeira da empresa, que aumenta o Lucro Líquido, Capital de Giro, EBITDA e Patrimônio Líquido e reduz a Dívida Líquida e a Alavancagem Financeira.

Figura 23: Evolução do rating de uma companhia do setor imobiliário



Fonte: Autoria Própria, 2022.

Na imagem 23, é mostrado um caso em que as agências mantiveram a mesma classificação ao logo de todo o período analisado, apesar da movimentação dos indicadores, como, o Lucro Líquido que chega a ficar negativo. No entanto, o modelo consegue capturar a

movimentação, reduzindo a classificação quando o Lucro Líquido, Capital de Giro, Margem Líquida e EBIDTA decrescem e aumentado quando os mesmos indicadores voltaram a subir.

6 CONCLUSÕES E TRABALHOS FUTUROS

Com o crescente aumento no volume das operações de crédito privado no mercado financeiro brasileiro e expansão das instituições provedoras de crédito, faz-se necessário o desenvolvimento de ferramentas que possibilitem uma análise objetiva do risco de crédito dos tomadores de empréstimo, isso porque uma grande parcela das companhias brasileiras de médio porte necessita do uso intensivo de capital nas suas operações, logo, precisam recorrer a instituições financeiras para obter crédito. No entanto, grande parte dessas companhias não possuem score de crédito; desse modo, o modelo apresentado neste trabalho pode ser utilizado como uma ferramenta na obtenção do rating, indicador fundamental na mensuração do risco de crédito da companhia.

Ao longo do desenvolvimento do projeto, foram testadas as técnicas mais utilizadas em machine learning, permitindo assim que um comparativo exaustivo do desempenho de cada técnica em aplicações dessa natureza pudesse ser feito.

Como apresentado na seção de resultados, o modelo com melhor desempenho foi utilizando Random Forest e as seguintes técnicas de pré-processamento: seleção de atributos utilizando ExtraTree e correlação, preenchimento de valor faltante com a mediana, escalonamento com QuantileTransformer e sobreamostragem das classes não majoritárias usando SMOTE. Por fim, foi realizado o tuning do modelo utilizando Otimização Bayesiana. Apesar da alta performance do modelo, com uma acurácia de 90.12% e F1-Score macro de 90.94%, o número de registros da base de dados ainda não é o ideal (1253 registros), principalmente para os ratings mais baixos, que são as classes minoritárias. Ainda assim, apesar das limitações da base de dados, o modelo final cumpre os objetivos propostos, permitindo análises do rating de crédito e confrontamentos com a classificação das agências.

6.1 Trabalhos Futuros

Próximos passos para o projeto seria incorporar ao modelo empresas financeiras, isso porque as companhias desse setor possuem uma estrutura de demonstrativo financeiro

diferente dos demais, assim, algumas das variáveis que são fundamentais para explicar o score do modelo atual, não estão presentes nos demonstrativos das companhias desse setor. Além disso, poderia ser introduzidas variáveis macro, como, PIB, IPCA, IGP-M, Selic, USDBRL, o que deixaria o modelo exposto não só às variáveis micro, mas também, às macroeconômicas. Por fim, para uma maior confiabilidade e performance do modelo, é necessário aumentar o número de registros da base de dados, principalmente para os ratings mais baixos, pois são as classes minoritárias. Dessa forma, será possível construir modelos mais detalhados, sem a necessidade de agrupar os ratings, com um ótimo desempenho e generalização dos resultados.

REFERÊNCIAS

[1] CARDOSO, V. et al. Assessing corporate risk: a pd model based on credit ratings. *FRAP - Finance, Risk and Accounting Perspectives Conference*, FRAP - Finance, Risk and Accounting Perspectives Conference, 2013.

[2] AKIAMA, S. Likelihood of nonpayment of large enterprises in national financial system. *Faculdade de Economia, Administração e Contabilidade, Universidade de São Paulo*, Faculdade de Economia, Administração e Contabilidade, Universidade de São Paulo, 2008.

[3] EDUCATION, I. C. *What is machine learning?* 2020. Disponível em: <https://www.ibm.com/cloud/learn/machine-learning#toc-what-is-ma-qhM6PX35>. Acesso em: 17/05/2022.

[4] PURAKKATT, A. *Deep Learning — Artificial Neural Network(ANN).* 2020. Disponível em: <https://medium.com/analytics-vidhya/deep-learning-artificial-neural-network-ann-13b54c3f370f>. Acesso em: 17/05/2022.

[5] JANIESCH, C.; ZSCHECH, P.; HEINRICH, K. Machine learning and deep learning. *Electronic Markets*, v. 31, p. 685–695, 2021.

[6] JAVATPOINT. *Types of Machine Learning.* Disponível em: <https://www.javatpoint.com/types-of-machine-learning>. Acesso em: 17/05/2022.

[7] JAVATPOINT. *Supervised Machine Learning.* Disponível em: <https://www.javatpoint.com/supervised-machine-learning>. Acesso em: 17/05/2022.

[8] BROWNLEE, J. *Difference Between Classification and Regression in Machine Learning.* 2017. Disponível em: <https://machinelearningmastery.com/classification-versus-regression-in-machine-learning/#:~:text=Classification%20vs%20Regression,-Classification%20predictive%20modeling&text=Classification%20is%20the%20task%20of,of%20predicting%20a%20continuous%20quantity.>. Acesso em: 17/05/2022.

[9] Taylor Fogarty. *Regression or Classification? Linear or Logistic?* 2019. Disponível em: <https://towardsdatascience.com/regression-or-classification-linear-or-logistic-f093e8757b9c>. Acesso em: 17/05/2022.

[10] CLOUD, G. *Visão geral do ajuste de hiperparâmetros.* Disponível em: <https://cloud.google.com/ai-platform/training/docs/hyperparameter-tuning-overview>. Acesso em: 17/05/2022.

[11] JAVATPOINT. *Data Preprocessing in Machine learning.* Disponível em: <https://www.javatpoint.com/data-preprocessing-machine-learning>. Acesso em: 17/05/2022.

[12] BARKVED, K. *Data Cleaning: The Most Important Step in Machine Learning*. 2022. Disponível em: <https://www.obviously.ai/post/data-cleaning-in-machine-learning>. Acesso em: 17/05/2022.

[13] VERMA, Y. *A Complete Guide to Categorical Data Encoding*. 2021. Disponível em: <https://analyticsindiamag.com/a-complete-guide-to-categorical-data-encoding/>. Acesso em: 17/05/2022.

[14] HALE, J. *Scale, Standardize, or Normalize with Scikit-Learn*. 2019. Disponível em: <https://towardsdatascience.com/scale-standardize-or-normalize-with-scikit-learn-6ccc7d176a02>. Acesso em: 17/05/2022.

[15] AFONJA, T. *Accuracy Paradox*. 2017. Disponível em: <https://towardsdatascience.com/accuracy-paradox-897a69e2dd9b>. Acesso em: 17/05/2022.

[16] BROWNLEE, J. *8 Tactics to Combat Imbalanced Classes in Your Machine Learning Dataset*. 2015. Disponível em: <https://machinelearningmastery.com/tactics-to-combat-imbalanced-classes-in-your-machine-learning-dataset/>. Acesso em: 17/05/2022.

[17] Saishruthi Swaminathan. *Logistic Regression — Detailed Overview*. 2018. Disponível em: <https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc>. Acesso em: 17/05/2022.

[18] YıLDıRıM, S. *K-Nearest Neighbors (kNN) — Explained*. 2020. Disponível em: <https://towardsdatascience.com/k-nearest-neighbors-knn-explained-cbc31849a7e3>. Acesso em: 17/05/2022.

[19] Antony Christopher. *K-Nearest Neighbor*. 2021. Disponível em: <https://medium.com/swlh/k-nearest-neighbor-ca2593d7a3c4>. Acesso em: 17/05/2022.

[20] IBM. *Compute KNN: distance metrics*. Disponível em: <https://www.ibm.com/topics/knn#:~:text=The%20k%2Dnearest%20neighbors%20algorithm%2C%20also%20known%20as%20KNN%20or,of%20an%20individual%20data%20point.>.

[21] COUTINHO, B. *Modelos de Predição / SVM*. 2019. Disponível em: <https://medium.com/turing-talks/turing-talks-12-classifica%C3%A7%C3%A3o-por-svm-f4598094a3f1>. Acesso em: 17/05/2022.

[22] SHARMA, S. *Kernel Trick in SVM*. 2019. Disponível em: <https://medium.com/analytics-vidhya/how-to-classify-non-linear-data-to-linear-data-bb2df1a6b781>. Acesso em: 17/05/2022.

[23] MISRA, R. *Support Vector Machines — Soft Margin Formulation and Kernel Trick*. 2019. Disponível em: <https://towardsdatascience.com/support-vector-machines-soft-margin-formulation-and-kernel-trick-4c9729dc8efe>. Acesso em: 17/05/2022.

[24] GUPTA, P. *Decision Trees in Machine Learning*. 2017. Disponível em: <https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052>. Acesso em: 17/05/2022.

[25] TRIPATHI, M. *Understanding Decision Trees with Python*. 2020. Disponível em: <https://datascience.foundation/sciencewhitepaper/understanding-decision-trees-with-python>. Acesso em: 17/05/2022.

[26] ALHAMID, M. *Ensemble Models*. 2021. Disponível em: <https://towardsdatascience.com/ensemble-models-5a62d4f4cb0c>. Acesso em: 17/05/2022.

[27] KESHAV. *Introduction To Gradient descent algorithm (With Formula)*. Disponível em: <https://vidyasheela.com/post/introduction-to-gradient-descent-algorithm-with-formula>. Acesso em: 17/05/2022.

[28] IBM. *Redes neurais)*. 2020. Disponível em: <https://www.ibm.com/br-pt/cloud/learn/neural-networks>. Acesso em: 17/05/2022.

[29] OGNJANOVSKI, G. *Everything you need to know about Neural Networks and Backpropagation — Machine Learning Easy and Fun)*. 2019. Disponível em: <https://towardsdatascience.com/everything-you-need-to-know-about-neural-networks-and-backpropagation-machine-le
Acesso em: 17/05/2022.

[30] DEVELOPERS, G. *Training and Test Sets: Splitting Data*. 2021. Disponível em: <https://developers.google.com/machine-learning/crash-course/training-and-test-sets/splitting-data>. Acesso em: 17/05/2022.

[31] DEVELOPERS, G. *Validation Set: Another Partition*. 2021. Disponível em: <https://developers.google.com/machine-learning/crash-course/validation/another-partition>. Acesso em: 17/05/2022.

[32] BOSE, A. *Cross Validation — Why & How*. 2019. Disponível em: <https://towardsdatascience.com/cross-validation-430d9a5fee22>. Acesso em: 17/05/2022.

[33] LEG/UFPR, L. de Estatística e G. *Métodos de reamostragem*. Disponível em: <http://cursos.leg.ufpr.br/ML4all/apoio/reamostragem.html#qual_valor_de_(k)_escolher>. Acesso em: 17/05/2022.

[34] SRIVASTAVA, T. *11 Important Model Evaluation Metrics for Machine Learning Everyone should know*. 2019. Disponível em: <https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/>. Acesso em: 17/05/2022.

[35] BHARATHI. *Confusion Matrix for Multi-Class Classification*. 2021. Disponível em: <https://www.analyticsvidhya.com/blog/2021/06/confusion-matrix-for-multi-class-classification/>. Acesso em: 17/05/2022.

[36] NOGARE, D. *Performance de Machine Learning – Matriz de Con-*

fusão. 2020. Disponível em: <https://diegonogare.net/2020/04/performance-de-machine-learning-matriz-de-confusao/>. Acesso em: 17/05/2022.

[37] UPADHYAY, A. *Precision/Recall Tradeoff*. 2020. Disponível em: <https://medium.com/analytics-vidhya/precision-recall-tradeoff-79e892d43134>. Acesso em: 17/05/2022.

[38] PEREIRA, P. M. P. *Análise de Risco de Crédito usando algoritmos de Machine Learning*. Dissertação (Dissertação de Mestrado) — INSTITUTO UNIVERSITÁRIO DE LISBOA DEPARTAMENTO DE FINANÇAS, 2020.

[39] SHARMA, A. *Statistics for Data Science: What is Skewness and Why is it Important?* 2020. Disponível em: <https://www.analyticsvidhya.com/blog/2020/07/what-is-skewness-statistics/>. Acesso em: 17/05/2022.

[40] SATPATHY, S. *Overcoming Class Imbalance using SMOTE Techniques*. 2020. Disponível em: <https://www.analyticsvidhya.com/blog/2020/10/overcoming-class-imbalance-using-smote-techniques/#:~:text=SMOTE%3A%20Synthetic%20Minority%20Oversampling%20Technique,problem%20posed%20by%20random%20oversampling.>. Acesso em: 17/05/2022.

[41] BENGIO, J. B. . Y. *Random Search for Hyper-Parameter Optimization*. 2012. Disponível em: <https://www.jmlr.org/papers/v13/bergstra12a.html>. Acesso em: 17/05/2022.

[42] COELHO, F.; AMORIM, D.; CAMARGOS, M. Analisando mÉtodos de machine learning e avaliaÇÃo do risco de crÉdito. *Revista Gestão & Tecnologia*, Elsevier, v. 29, n. 9, p. 1223–1230, 1996.

[43] WIKIPéDIA. *Teste Kolmogorov-Smirnov*. Disponível em: <https://pt.wikipedia.org/wiki/Teste_Kolmogorov-Smirnov>. Acesso em: 17/05/2022.

[44] HAMORI, S. et al. Ensemble learning or deep learning? application to default risk analysis. *Risk and Financial Management*, Risk and Financial Management, 2018.

[45] ADDO, P.; GUEGAN, D.; HASSANI, B. Credit risk analysis using machine and deep learning models. *Risks*, Risks, 2018.