Avaliação de Algoritmos de Aprendizado de Máquina na Classificação de Lançamentos Bancários: Um Estudo Aplicado ao Ministério Público do Estado da Bahia

Jucimar Cerqueira dos Santos
Centro Universitário SENAI CIMATEC
Salvador, Brasil
jucimarcerqsan@gmail.com

Marcelo da Silva Lima
Centro Universitário SENAI CIMATEC
Salvador, Brasil
mdslima@hotmail.com

Marcus Vinicius Caetano Pinto
Centro Universitário SENAI CIMATEC
Salvador, Brasil
marcuspinto70@gmail.com

Oritentador: Tacito Henrique da Silva Graça
Centro Universitário SENAI CIMATEC
Salvador, Brasil
tacito.silva@fieb.org.br

Resumo **-** Este trabalho investigou a eficácia de diversos algoritmos de aprendizado de máquina na tarefa de classificação de lançamentos bancários, utilizando dados recebidos de instituições financeiras pelo Ministério Público do Estado da Bahia (MPBA). O objetivo geral do estudo foi fortalecer a capacidade de investigação do MPBA através da automação e aperfeiçoamento do processo de classificação e análise de transações financeiras. A metodologia empregada focou em uma abordagem comparativa do desempenho, na execução da tarefa proposta, de técnicas tradicionais de aprendizado de máquina: Regressão Logística, *Support Vector Machine* (SVM), Árvore de Decisão e K-*Nearest Neighbors* (KNN) e, também, de variados ajustes do modelo pré-treinado BERT. Os dados utilizados neste estudo foram anonimizados em adequação às exigências legais e rigorosamente explorados e dimensionados visando o atendimento das demandas técnicas e à capacidade computacional disponível. Os principais resultados revelaram que, de forma geral, os modelos estudados apresentaram ótimo desempenho na classificação dos lançamentos bancários. O modelo SVM, com a configuração apropriada, atingiu o F1-*score* de 88,54% no conjunto de dados de teste, superando os demais modelos testados. Este estudo contribui significativamente para a adoção de soluções de inteligência artificial no setor público, propondo os fundamentos de um modelo escalável que pode ser aplicado em outras instituições públicas para a análise de dados financeiros. A aplicação dessas técnicas não apenas aprimora as operações existentes, mas também abre novas avenidas para a inovação no tratamento e análise de grandes volumes de dados em órgãos governamentais.

Palavras-chave **-** *Lançamentos bancários; Aprendizado de máquina; SVM; Análise de dados; Setor público; BERT.*

1. INTRODUÇÃO

A Coordenadoria de Segurança Institucional e Inteligência (CSI) do MPBA desempenha um papel crucial no apoio à atividade finalística do Órgão através da operacionalização da coleta de dados de afastamento de sigilo e da produção de relatórios de análise técnica a respeito destes [1]. Tais tarefas são essenciais para a execução das atividades investigativas promovidas por Membros do *Parquet*. Diante do grande volume de dados resultante do afastamento de sigilo bancário, os analistas da CSI enfrentam o desafio significativo de analisar milhares de lançamentos bancários. Este processo é dificultado pelo eventual uso, por parte das instituições financeiras, de códigos de categorias genéricas (códigos “104” ou “205” = “Lançamento Avisado” e “999”) para classificação de parte dos lançamentos [2]. O uso de categorias de classificação genéricas pode induzir a distorções nas análises, destacando-se assim a necessidade urgente de abordagens mais precisas e eficientes na classificação desses dados.

Além do desafio técnico mencionado, observa-se que o contexto nacional em ciência de dados na administração pública reflete tanto oportunidades quanto lacunas significativas. Conforme o Índice de Prontidão em Inteligência Artificial (IA) dos Governos da Oxford Insights, o Brasil, em 2020, ocupava o 63º lugar mundialmente e o 6º na região da América Latina e Caribe em termos de capacidade governamental de implementação de soluções de IA. Um estudo do Tribunal de Contas da União em 2021 revelou que apenas 28% dos órgãos públicos brasileiros utilizam alguma solução de IA e que 38% não têm intenção de adotar tais tecnologias em seus processos [3]. Estes dados indicam um vasto campo de ação ainda inexplorado, reforçando a necessidade de projetos como o proposto neste trabalho, que busca explorar e expandir o uso de IA no apoio à análise de dados para fortalecimento da atuação Ministerial.

Sabe-se que a simples utilização de ferramentas tecnológicas não se traduz automaticamente em avanços nos processos de apoio à investigação. Também é fundamental a adequação às necessidades específicas, ao contexto legal, e à cultura da organização em questão [4]. Além disso, devem ser considerados vários aspectos inerentes à inovação no setor público: restrições orçamentárias, dificuldades na operacionalização de aquisições, qualificação profissional, confidencialidade de dados, falta de padrões técnicos, entre

outros [5]. Nesse cenário, o projeto ora exposto ganha importância crucial. Ele não apenas responde a uma necessidade premente de aprimoramento nos processos investigativos, como também representa uma oportunidade significativa de inovação no setor público através da utilização de tecnologias de inteligência artificial adaptadas ao contexto específico do MPBA.

Neste estudo, os dados dos lançamentos bancários são compostos de informações categóricas, como o 'tipo de pessoa', o 'tipo de conta', o 'código do banco', entre outros, e pequenos textos, como o 'histórico do lançamento' e a 'observação', que são escritos em jargão bancário ou, eventualmente, em linguagem natural. A experiência demonstra que, dada sua facilidade em analisar tanto texto quanto dados, técnicas de Aprendizado de Máquina supervisionado seriam muito adequadas para solucionar nosso problema principal [5] [6]. Nessas técnicas, para cada exemplo de lançamento bancário apresentado ao algoritmo é necessário apresentar, também, a resposta desejada (ou seja, um rótulo informando a que categoria o lançamento pertence) [7].

O objetivo deste estudo se concentrará na comparação entre modelos de Aprendizado de Máquina na tarefa de classificar lançamentos bancários, devidamente anonimizados, obtidos do banco de dados da CSI/MPBA. O uso dos próprios lançamentos com classificação genérica, que exigiria um extenso período de classificação manual para o treinamento adequado dos modelos, não será abordado diretamente neste projeto. Contudo, os resultados obtidos servirão como um valioso ponto de partida para futuras ações no âmbito do Ministério Público, onde projetos mais abrangentes, com maior alocação de tempo, equipe e recursos técnicos, poderão utilizar as descobertas deste estudo comparativo para selecionar os modelos mais eficazes para utilização em escala de produção.

2. Referencial Teórico

2.1. Inteligência artificial

A inteligência artificial tem provocado mudanças digitais significativas nas empresas e nos governos, trazendo benefícios e desafios [8]. Mas o que é inteligência artificial? É o estudo das computações que tornam possível perceber, raciocinar e agir [9] ou conjunto de técnicas para a construção de máquinas inteligentes, capazes de resolver problemas que requerem inteligência humana [10]. Uma área de pesquisa que investiga formas de habilitar o computador a realizar tarefas nas quais, até o momento, o ser humano tem um melhor desempenho [11]. É o ramo da ciência da computação dedicado à automação do comportamento inteligente [12].

Todos esses conceitos trabalham com a ideia de que a inteligência artificial envolve a construção de sistemas computacionais capazes de realizar e resolver tarefas que normalmente exigiriam a inteligência humana. Ela pode resolver problemas complexos com mais precisão. Conforme enumerados na Figura 1, alguns dos usos de IA hoje são assistentes virtuais, robôs automatizados, carros autônomos, visão computacional, aprendizado de máquina, processamento de linguagem natural e outros [12].

Fig. 1: Aplicações e campos da IA. Fonte: Adaptado de [13].

Há uma relação entre Inteligência Artificial (IA), *Machine Learning* (ML) e *Deep Learning* (DL), conforme Figura 2. A IA é o campo que se dedica a criar sistemas que aprendem com a experiência, processam linguagem natural, raciocinam e tomam decisões de forma semelhante aos humanos. Já no ML os sistemas não são programados para realizar uma tarefa, mas sim para aprender com dados históricos ou ambiente. O DL, uma área emergente do aprendizado de máquina, envolve treinar uma rede neural usando grandes quantidades de dados históricos para executar uma tarefa específica [13].

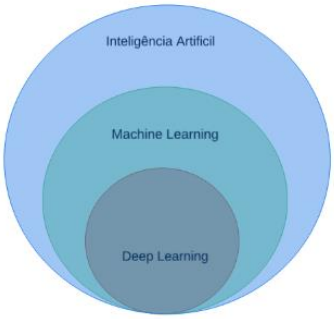

Fig. 2: Relação entre IA, ML e DL. Fonte: Adaptado de [13].

2.2. Mineração de dados

A Mineração de dados pode ser entendida como o processo de descobrir padrões interessantes e conhecimento a partir de grandes quantidades de dados [14]. A Descoberta de Conhecimento em Bancos de Dados (KDD) é o termo usado para descrever a mineração de dados, que consiste na extração de informações implícitas, previamente desconhecidas e potencialmente úteis de bancos de dados. A

mineração de dados e a descoberta de conhecimento em bancos de dados (ou KDD) são geralmente pensadas como sinônimos, mas na verdade a mineração de dados é um componente do processo de descoberta de conhecimento [15].

A mineração apresenta alguns métodos: previsão, descrição, classificação, agrupamento, sumarização, detecção de anomalias, descritos na Figura 3. A previsão ou regressão estima os valores futuros de outras variáveis usando dados conhecidos. Já o objetivo da descrição é encontrar padrões compreensíveis pelos humanos para descrever os dados [16]. A classificação, também conhecida como classificação supervisionada, usa rótulos de classe específicos para organizar os objetos na coleta de dados. As técnicas de classificação geralmente empregam um conjunto de instrução em que todos os objetos já estão conectados a rótulos de classe conhecidos [14].

No agrupamento, semelhante à classificação e conhecida por classificação não supervisionada, os rótulos das classes são desconhecidos, o que significa que o algoritmo de cluster deve descobrir quais são as classes. A sumarização é um resumo das características gerais dos objetos em uma classe alvo e produz o que é chamado de regras características. Na associação a descoberta usa regras de associação e é normalmente usada para análise de cesta de compras. A deteção de anomalias ou análise de outliers são elementos de dados que não podem ser agrupados em uma determinada classe ou cluster, ou seja, exceções, surpresas ou valores discrepantes que podem ocasionar ruídos [14].

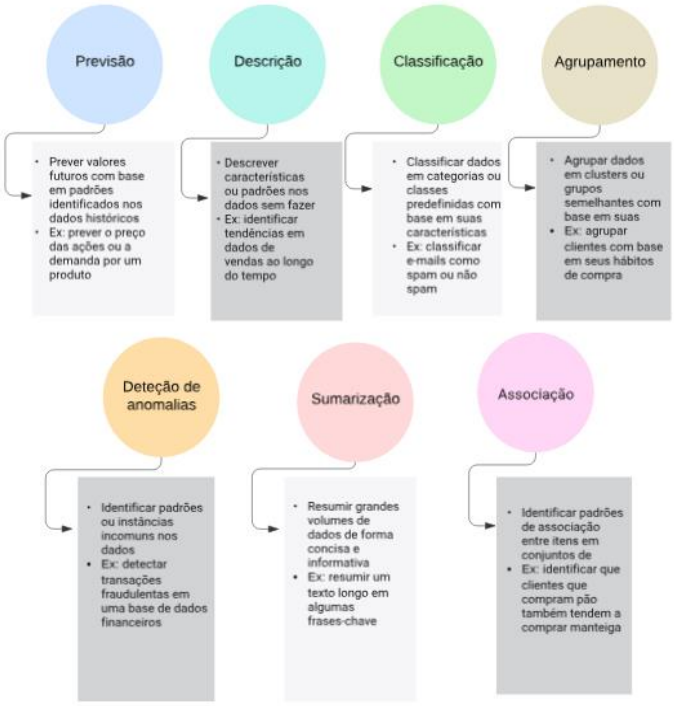

Fig. 3: Técnicas de mineração. Fonte: Autores.

2.3. Análise exploratória de dados

A análise exploratória de dados (AED) é um conjunto de técnicas de análise que visam resumir as principais características de um conjunto de dados, frequentemente com a ajuda de métodos visuais. Originalmente desenvolvida pelo matemático americano John Tukey na década de 1970, as técnicas de AED continuam sendo um método amplamente usado no processo de descoberta de dados hoje. É uma etapa inicial crucial no *pipeline de machine learning* que envolve a investigação dos dados para descobrir padrões, detectar anomalias, testar hipóteses e verificar suposições [17].

A AED é essencial no entendimento dos dados porque permite aos analistas e cientistas de dados entenderem a estrutura, as principais características e as relações presentes nos dados. Isto ajuda a identificar problemas de qualidade dos dados, como valores faltantes, outliers e inconsistências prevenindo erros que se não forem tratados podem levar a modelos de machine learning sem robustez ou incorretos [17].

A AED é dividida em quatro tipos principais: univariada não gráfica, univariada gráfica, multivariada não gráfica e multivariada gráfica. A análise univariada não gráfica é a forma mais simples, lidando com uma única variável para descrever dados e identificar padrões. A análise univariada gráfica utiliza gráficos como gráficos de caule e folhas, histogramas e *box plots* para fornecer uma visualização completa dos dados. A análise multivariada não gráfica envolve a relação entre duas ou mais variáveis através de cruzamento de tabelas ou estatísticas. A análise multivariada gráfica utiliza gráficos, como gráficos de barras agrupadas, para exibir relações entre múltiplas variáveis, representando diferentes níveis de cada variável em barras separadas [17].

2.4. NLP (*Natural Language Processing*)

A pesquisa sobre a tradução automática na década de 1950 deu origem ao campo. A NLP, baseada em regras, teve pouco sucesso nos primeiros dias. O potencial real da NLP só começou a ser visto na década de 90, quando os métodos estatísticos foram apresentados e o aumento no poder do computador ao longo dos anos contribuiu para este avanço. Isso fez com que o computador pudesse trabalhar com esses tipos de dados estatísticos que antes não eram possíveis. A introdução de modelos de redes neurais para NLP foi um avanço que ocorreu rapidamente e esses modelos de última geração foram demonstrando uma variedade de tarefas [19].

A área de estudo e aplicação conhecida como *Natural Language Processing* (NLP) estuda como os computadores podem compreender e manipular texto ou fala em linguagem natural para realizar tarefas. Para que os sistemas de computador possam compreender e manipular a linguagem natural para realizar as tarefas pretendidas, os pesquisadores da NLP buscam adquirir conhecimento sobre a forma como os seres humanos entendem e usam a linguagem [18]. É uma abordagem para fazer os computadores aprenderem, compreender e processar texto de maneira semelhante à que os humanos fazem [19].

Uma variedade de disciplinas abrange os fundamentos da NLP, como ciências da computação e informação, linguística, matemática, engenharia elétrica e eletrônica,

inteligência artificial e robótica, psicologia, entre outras [18]. A introdução de *chatbots* como o ChatGPT tem atraído atenção recentemente, mas o campo vai além disso. Ele inclui tarefas como tradução de texto entre idiomas, condensação de grandes quantidades de texto em poucas linhas e conversão de informações de bancos de dados para linguagem humana [19].

2.5. Pré-processamento de dados textuais

A extração e o pré-processamento de recursos são etapas cruciais para aplicações de classificação de texto. A maioria dos conjuntos de dados de texto e documentos contém muitas palavras desnecessárias, como palavras irrelevantes, erros ortográficos, gírias etc. Em muitos algoritmos, especialmente algoritmos de aprendizagem estatística e probabilística, ruído e recursos desnecessários podem ter efeitos adversos no desempenho do sistema [20].

Os dados de entrada para tarefas de linguagem natural são textos não estruturados e brutos. A informação textual não tem representação numérica intrínseca, ao contrário de outros tipos de dados, como imagens ou séries temporais. Antes de poder ser usado por qualquer classificador, a informação textual deve ser projetada em um espaço de recursos apropriado. Portanto, os procedimentos de pré-processamento são cruciais, pois limpam e normalizam os dados de entrada para melhorar os resultados da etapa de extração de recursos [20] [21].

A maioria dos documentos e conjuntos de dados de texto contém uma quantidade excessiva de caracteres desnecessários, como pontuações e caracteres especiais. A pontuação crítica e os caracteres especiais são essenciais para a compreensão humana dos documentos, mas podem ser inconvenientes para os algoritmos de classificação, sendo necessário sua limpeza. Outra abordagem é quando ao formar uma frase, os pontos de dados de texto e documento usam uma variedade de letras maiúsculas, podendo gerar dificuldade na classificação de documentos grandes e para corrigir isso, e o método mais comum é reduzir todas as letras para minúsculas [20].

Outro processo utilizado no pré-processamento é a tokenização, que consiste numa técnica que divide o texto em palavras, frases, símbolos ou outros componentes importantes e tem como objetivo descobrir quais palavras estão presentes em uma frase [20]. Assim, a tokenização é a operação mais simples que deve ser usada para texto e é comumente descrito como o processo de quebrar um fluxo de texto em pedaços menores, que são chamados de tokens [21]. Já *stopwords* são palavras não informativas que aparecem em grande número, mas não têm importância semântica [22], ou seja, no processo de classificar textos e documentos, muitas palavras, como "a", "sobre", "acima", "de", "depois", "novamente", etc., não têm relevância para os algoritmos de classificação, sendo necessária sua remoção [20].

Ainda temos a stemização e lematização. A primeira é um processo de remoção de quaisquer sufixos e prefixos (afixos) para reduzir suas palavras as formas radicais [22]. Já a segunda é um processo que remove ou substitui o sufixo de uma palavra para obter a forma básica da palavra, fazendo reduzir a dimensionalidade dos dados e ajudar no agrupamento de palavras semelhantes. [20]. A finalidade principal de stemização e lematização é semelhante. Ambos reduzem uma variante de palavra ao seu "radical" na pronúncia e ao seu "lema" na lematização. A pequena diferença entre os dois conceitos está no fato que na stemização, o "radical" é obtido aplicando uma série de regras, mas sem levar em consideração a classe gramatical ou o contexto em que a palavra ocorre. Por outro lado, a lematização é o processo de obter o "lema" de uma palavra, o que envolve reduzir as formas da palavra à sua forma raiz após a compreensão de sua classe gramatical e seu contexto na frase [22].

2.6. Codificação de variáveis categóricas

Variáveis categóricas frequentemente aparecem em conjuntos de dados para tarefas de classificação e regressão e precisam ser codificadas em valores numéricos antes do treinamento. Como muitos codificadores foram criados e podem influenciar significativamente o desempenho, escolher o codificador certo para uma tarefa torna-se uma questão prática demorada [23]. As quatro principais variáveis categóricas de dados estão as escalas nominal, ordinal, intervalo e proporcional. Classificações sem valor quantitativo, como gênero ou estado civil, são representadas por variáveis em escala nominal. Embora a escala ordinal demonstre uma ordem, as distâncias entre as categorias, como os níveis de satisfação, não podem ser quantificadas. As escalas de proporção têm um zero absoluto e permitem comparar diferentes valores, como altura ou peso. Por outro lado, as escalas de intervalo fornecem ordem e intervalos iguais, como temperatura em Fahrenheit ou Celsius [24].

Há várias técnicas de codificação, entre elas: *one-hot*, *ordinal*, *target*, CatBoost, soma, contagem. No *one hot*, uma variável categórica é transformada em várias variáveis binárias e cada categoria da variável inicial é transformada em uma nova variável binária. Ela recebe um valor de 1 se a categoria estiver presente, e um valor de 0 se não estiver presente [24]. Assim, na codificação *one-hot*, cada nível de uma variável é comparado com um nível de referência. Cada nível da variável é transformado em uma variável binária separada. [25].

No *ordinal encoding* (codificação ordinal), cada categoria é atribuída a um número inteiro até que o número total de categorias seja conhecido. Embora não adicione colunas adicionais aos dados, fornece uma ordem para a variável que pode não existir de fato [24]. A *label encoder* (codificador de rótulos) é uma das aplicações na qual transforma níveis de recursos (categorias) em números inteiros de 1 a N, onde N é a cardinalidade de recurso. Este tipo de codificação cria um relacionamento ordinal entre níveis de recursos. Tal esquema pode ser vantajoso se uma ordem verdadeira estiver presente nos dados, pois preserva mais informações para ajustar um modelo [26].

A *target encoding* (codificação alvo), também conhecida como codificação média, consiste em substituir cada categoria pela média da variável alvo para aquela categoria,

ou alguma outra combinação. A variável prevista no modelo de aprendizado de máquina é a variável alvo. A principal vantagem da codificação de destino é que pode fornecer uma representação de dados mais informativa do que a codificação ordinal ou *one-hot*. Ao codificar as categorias com base em como elas se relacionam com a variável alvo, pode descobrir mais sobre a estrutura subjacente dos dados [27].

A codificação CatBoost é como a codificação alvo, mas inclui um princípio de ordenação para resolver o problema de vazamento alvo. Ele usa um princípio parecido com o da validação de dados de série temporal. Os valores da estatística alvo são baseados no histórico observado e com outras palavras, a probabilidade alvo para o recurso atual é encontrada apenas a partir das linhas (observações) que existem antes disso. Esta abordagem garante que as estatísticas alvo das observações iniciais no conjunto de dados tenham uma variância significativamente maior do que as observações subsequentes [25].

A codificação de soma compara a média da variável dependente para um nível específico com a média da variável dependente em todo o nível. Em outras palavras, o nível 1 está em desacordo com todo o resto, o nível 2 está em desacordo com todo o resto e o nível 3 está em desacordo com todo o resto [25]. A codificação de contagem, também conhecida como codificação de frequência, muda a frequência ou contagem de cada variável categórica em um conjunto de dados. Esta abordagem pode ser vantajosa para variáveis com várias categorias, fornecendo uma representação mais informativa do que a codificação única quente. Mas se as frequências não forem representativas das distribuições subjacentes das categorias, pode causar viés.

2.7. Representação de textos

Antes de qualquer processo de classificação, a representação do texto é uma das tarefas mais importantes que deve ser concluída. Além disso, como quase todos os algoritmos recebem entrada em números como vetores de recursos com tamanho predeterminado em vez de dados textuais com comprimento variável, os textos não podem ser fornecidos como entrada para modelos de aprendizado de máquina. Para resolver este problema, é necessário transformar os dados textuais para documentar vetores. Em geral, há duas maneiras de fazer isso. A primeira abordagem independente do contexto apresenta um documento como um conjunto de termos com suas frequências correspondentes. A segunda técnica aumenta a eficiência da classificação encontrando e usando informações de associação de termos, dependendo do contexto. Eles permitem que a forma como um termo contribui para um resultado de classificação seja influenciada por sua presença ou ausência [28].

O método *bag-of-words* (BoW) é uma abordagem de representação de texto que conta a frequência de palavras para tornar documentos mais simples. Este método, também conhecido como "saco de palavras", atribui a cada palavra um espaço de características baseado em sua contagem em cada documento. É uma técnica básica, mas eficaz para extrair recursos de texto [20]. É uma maneira de representar todas as palavras em um texto sem levar em consideração seu contexto, gramática ou relações semânticas, como sinônimo [19]. Um outro método chamado Frequência Inversa do Documento (IDF) pode ser usado em conjunto com a frequência do termo para reduzir o impacto das palavras que são implicitamente comuns no conjunto de dados de treinamento dos modelos de linguagem (*corpus*). O IDF dá um peso maior às palavras que aparecem com frequência baixa ou alta no documento. O termo “frequência de documento inversa de frequência de termo” refere-se combinação TF-IDF [20].

Outra maneira de representar palavras numericamente é a incorporação de palavras (*word embeddings*). Ela captura o significado e o contexto de cada termo usando vetores densos de baixa dimensão. Esses vetores preservam relações semânticas e contexto, tornando palavras com significados semelhantes próximas umas das outras. Essa técnica é útil em tarefas de Processamento de Linguagem Natural, como classificação de texto, análise de sentimentos e tradução automática. Os *word embeddings* mais populares hoje são Word2vec, GloVe e FastText [59].

O Word2vec é um modelo baseado na previsão que obtém *embeddings* usando uma rede neural. Para aprender a incorporação de palavras, *Word2vec* usa duas arquiteturas: *continuous-bag-of-words* (CBOW) e *continuous-skip-gram* (SKIP-GRAM). O modelo CBOW utiliza a informação contextual das palavras de entrada para prever a palavra-alvo, ajustando suas configurações durante o treinamento para melhorar sua interferência. O modelo *Skip-gram* opera de forma oposta ao CBOW. Em vez de prever uma palavra-alvo com base no contexto, dada uma única palavra, o *Skip-gram* tenta prever as palavras de contexto dentro de um intervalo específico [19].

O GloVe é outra técnica de representação de palavras, semelhante ao Word2Vec, porém com uma abordagem baseada em contagem. Também, o Word2Vec é um modelo preditivo e o GloVe aprende a similaridade semântica entre palavras ao explorar as estatísticas do corpus, como a coocorrência de palavras. Além disso, os modelos GloVe reduzem a dimensionalidade para lidar com a alta dimensionalidade da matriz de coocorrência das palavras. Já o FastText é uma das maneiras de incorporar palavras estáticas. Este método aborda principalmente a questão de que seus antecessores ignoraram, a morfologia das palavras, atribuindo um vetor único a cada palavra. Cada palavra é representada por um "saco de caracteres n-grama" no FastText [21].

2.8. Balanceamento

O desequilíbrio de classes é o problema que ocorre com frequência no que diz respeito ao campo da mineração de dados e da ciência de dados. A distribuição das classes é distorcida no conjunto de dados desequilibrados, e as instâncias de uma classe são muito maiores do que as de outras classes. Este problema de desequilíbrio de classe faz com que a classificação seja menos precisa porque prevê incorretamente as instâncias da classe minoritária [29].

Algoritmos de aprendizado de máquina padrão falham em classificar dados desbalanceados porque o erro de classificação na classe majoritária domina o erro de classificação na classe minoritária. Essa dominação resulta em afastar a função de separação da classe majoritária para diminuir o erro de classificação durante o processo de ajuste de peso. Consequentemente, os dados de teste na classe minoritária são frequentemente classificados erroneamente com mais frequência do que aqueles na classe majoritária [30].

Parar resolver problemas de dados desequilibrados podem ser utilizadas duas maneiras: solução no nível dos dados ou solução no nível do algoritmo. A solução ao nível dos dados é obtida equilibrando a distribuição da classe maioritária e minoritária usando subamostragem, sobreamostragem ou uma combinação de ambas. A aplicação da solução no nível do algoritmo é realizada alterando os métodos classificados ou otimizando o desempenho do algoritmo de aprendizagem. O benefício do nível de dados é que ele pode ser usado independentemente do classificador escolhido [31].

No nível de dados, os métodos para balanceamento das classes mais conhecidos são *undersampling* (subamostragem) que tenta equilibrar os dados reduzindo as amostras da classe majoritária e o *oversampling* (sobreamostragem), que remove a amostras da classe minoritária. Assim, na sobreamostragem as instâncias de classe minoritária são escalonadas, a fim de igualar as classes. Já o método de subamostragem, exclui a maioria das instâncias de classe para equilibrar o conjunto de dados [29]. A abordagem de sobreamostragem é usada com mais frequência do que a subamostragem, uma vez que o método de subamostragem eliminará os dados na classe majoritária [31].

Existem alguns algoritmos utilizados na sobreamostragem e na subamostragem. Na sobreamostragem o SMOTE é a mais popular, o qual substitui as instâncias da classe minoritária, criando exemplos sintéticos. O ADASYN é outro método alternativo que usa instâncias entre as duas classes dentro da classe minoritária em vez de exemplos sintéticos e usa a distribuição ponderada para as duas classes na classe minoritária. Na subamostragem existem os Tomek Links que são usados para identificar a linha de fronteira e dados ruidosos. Também usados para limpar dados que tiveram a sobreposição criada pelos métodos de amostragem, ou seja, são uma coleção de vizinhos mais próximos de classes opostas que estão a uma distância mínima uns dos outros [29].

2.9. Algoritmos de classificação

2.9.1. SVM

O SVM (*Support Vector Machine*) é um modelo muito robusto e versátil de aprendizado de máquina capaz de fazer classificações lineares e não lineares. É adaptada para classificações de conjuntos de dados complexos pequenos e de médio porte [7]. O SVM foi introduzido como uma nova máquina de aprendizado para problemas de classificação de dois grupos [57]. A principal ideia por trás dos SVMs é mapear os vetores de entrada de forma não linear para um espaço de características de altíssima dimensão. Nesse espaço de características, é construída uma superfície de decisão linear que separa as duas classes. As propriedades especiais dessa superfície de decisão garantem uma alta capacidade de generalização da máquina de aprendizado, o que significa que ela pode efetivamente classificar novos dados que não foram vistos durante o treinamento [57].

2.9.2. Árvores de Decisão

As árvores de decisão são uma técnica popular de aprendizado de máquina que são utilizadas para tarefas de classificação e regressão. Elas são importantes por sua capacidade de compreensão e interpretabilidade, além de serem úteis em uma variedade de domínios. Uma árvore de decisão é uma estrutura de árvore onde cada nó interno representa uma decisão baseada em um atributo (ou característica) específico, e cada ramo representa o resultado dessa decisão. Os nós da folha representam as classes ou valores de regressão que serão atribuídos aos exemplos de dados que alcançam esse ponto [32].

Elas são altamente interpretáveis e facilmente visualizáveis, o que significa que os resultados de um modelo de árvore de decisão podem ser facilmente compreendidos e explicados. Além disso, as árvores de decisão são versáteis e podem lidar com uma variedade de tipos de dados, incluindo dados categóricos e numéricos. Elas também são eficientes em termos computacionais e podem lidar com conjuntos de dados grandes. Existem várias técnicas associadas às árvores de decisão. Uma delas é a escolha do melhor atributo para dividir em cada nó da árvore, o que pode ser feito usando diferentes critérios de divisão, como o índice de Gini ou a entropia. Além disso, é comum utilizar técnicas de poda para evitar o *overfitting*, onde partes da árvore são removidas para melhorar a generalização do modelo [32].

2.9.3. Regressão Logística

A regressão logística é uma técnica estatística utilizada para modelar a relação entre uma variável dependente binária (ou categórica ordinal) e um conjunto de variáveis independentes, podendo ser tanto categóricas quanto contínuas. Ela é especialmente útil em problemas de classificação binária, onde o objetivo é prever a probabilidade de ocorrência de um evento. Ela é uma extensão da regressão linear, onde a relação entre as variáveis independentes e a variável dependente é modelada usando a função logística. Essa função logística transforma a soma ponderada das variáveis independentes em uma probabilidade que varia entre 0 e 1, representando a probabilidade de sucesso (ou pertencer à classe positiva) em um determinado evento [33].

Também é altamente interpretável e fácil de entender, o que permite que os resultados do modelo sejam explicados de forma simples. Além disso, é uma técnica robusta e eficiente que pode lidar com uma variedade de tipos de dados e distribuições. A regressão logística é amplamente utilizada em diversas áreas, incluindo medicina, ciências sociais, marketing e finanças, para prever eventos binários, como

diagnósticos médicos, resposta de *marketing*, inadimplência financeira, entre outros [33].

Na regressão logística, as estimativas dos parâmetros do modelo são obtidas através do método da máxima verossimilhança. O modelo resultante é avaliado usando diversas técnicas, como o teste de Wald, o teste de razão de verossimilhança e o teste de Hosmer-Lemeshow, entre outros. Além disso, técnicas de regularização, como a penalização L1 (*Lasso*) ou L2 (*Ridge*), podem ser aplicadas para evitar o *overfitting* e melhorar a generalização do modelo [33].

2.9.4. KNN

A classificação de texto usando o algoritmo k-vizinhos mais próximos (KNN) aborda o problema de encontrar as k instâncias rotuladas mais semelhantes a uma instância não rotulada para atribuir a categoria mais comum. Este método não paramétrico é rápido, calculando apenas as distâncias entre os pontos de dados. No entanto, seu desempenho é sensível à função de distância escolhida, sendo necessário ajustá-la para lidar eficazmente com grandes conjuntos de dados. Em ambientes de alta dimensionalidade, como textos com muitas características, a eficácia do KNN pode diminuir devido à "maldição da dimensionalidade", onde a noção de proximidade entre instâncias semelhantes se torna menos clara conforme o número de dimensões aumenta [21].

2.9.5. Métodos de ensemble

Os métodos de aprendizagem de conjunto (*Ensemble learning*) são amplamente categorizados em *boosting*, *bagging* e *stacking*. *Boosting* é uma técnica de aprendizado de máquina capaz de converter classificadores fracos em um classificador forte. A ideia principal por trás do *boosting* envolve a aplicação iterativa do algoritmo de aprendizagem base a versões ajustadas dos dados de entrada. Gradient boosting, XGBoost, CatBoost e AdaBoost são exemplos de algoritmos de *boosting*. Já o *Bagging* ajuda melhorar o desempenho de classificação de modelos de ML, combinando as previsões de conjuntos de treinamento gerados aleatoriamente. O Random Forest é o mais conhecido [34].

Por fim, o *Stacking* é uma estrutura de aprendizagem de conjunto que treina um algoritmo de ML separado para combinar as previsões de dois ou mais membros do conjunto. Seu principal benefício é utilizar a capacidade de vários algoritmos de bom desempenho para fazer classificações melhores do que qualquer um dos algoritmos individuais usados para construir o conjunto [34].
Gradient Boosting é um algoritmo que utiliza a técnica de *boosting* para criar modelos robustos, geralmente usando árvores de decisão como base. Introduzido por Breiman, ele representa o *boosting* como uma técnica de otimização em uma função de perda. A principal vantagem do Gradient Boosting é a capacidade de aprender padrões complexos corrigindo erros de modelos anteriores. No entanto, como outros algoritmos de *boosting*, ele pode sofrer de *overfitting* e modelar ruídos se os dados de entrada forem ruidosos. Já o AdaBoost é um algoritmo de *boosting* que combina múltiplos classificadores fracos para criar um classificador forte. Introduzido por Yoav Freund e Robert Schapire em 1995, o AdaBoost treina classificadores base, geralmente árvores de decisão, ajustando os pesos das amostras conforme as previsões dos classificadores. Apesar de ser influente na história do machine learning, o AdaBoost é sensível a dados ruidosos e outliers, o que pode levar ao *overfitting* devido à sua abordagem iterativa de aprendizagem [34].

XGBoost é um algoritmo baseado em árvore de decisão que emprega a estrutura de reforço de gradiente. É escalável e altamente preciso usado para aplicações de classificação e regressão. Desenvolvido em 2016 por Chen e Guestrin, tendo vários avanços em comparação com o algoritmo de reforço de gradiente convencional. O algoritmo XGBoost oferece várias vantagens, como a necessidade mínima de engenharia de recursos, já que pode lidar com normalização de dados e dimensionamento de recursos. Ele também lida bem com valores ausentes e pode gerar a importância dos recursos, útil para entender melhor os dados e realizar a seleção de recursos. XGBoost é mais rápido que a maioria dos algoritmos de machine learning, pode lidar com grandes conjuntos de dados e é menos propenso a *overfitting*. No entanto, possui a limitação de ter muitos hiperparâmetros, o que torna seu ajuste mais difícil [34].

O algoritmo CatBoost é uma implementação de aumento de gradiente e lida efetivamente com recursos categóricos durante a fase de treinamento. Sua melhoria é a capacidade de realizar estimativas de gradiente imparciais que reduzem o *overfitting* e transformação automaticamente de recursos categóricos em numéricos [34].

O *Random Forest* é um algoritmo de aprendizado de máquina que pertence à classe dos chamados "*ensemble methods*", ou métodos de conjunto. Ele é construído sobre o conceito de árvores de decisão, mas em vez de utilizar uma única árvore de decisão, o Random Forest cria uma coleção (ou "floresta") de árvores de decisão onde cada árvore é construída independentemente, utilizando uma amostra aleatória dos dados de treinamento e uma seleção aleatória de características em cada divisão. Em seguida, as previsões de cada árvore são combinadas para produzir uma previsão final [36].

O *Random Forest* é importante por várias razões. Primeiro, ele é altamente eficaz em uma variedade de problemas de classificação e regressão, e geralmente produz resultados robustos e precisos. Além disso, o *Random Forest* é resistente ao *overfitting*, devido à sua capacidade de generalização a partir das previsões de múltiplas árvores de decisão. Além disso, é um algoritmo versátil que pode lidar com conjuntos de dados grandes e de alta dimensionalidade.

Uma das principais técnicas utilizadas no *Random Forest* é o *bootstrap sampling*, onde cada árvore é construída a partir de uma amostra aleatória (com substituição) dos dados de treinamento. Além disso, a seleção aleatória de características em cada divisão ajuda a aumentar a diversidade entre as árvores na floresta, o que contribui para a precisão e robustez do modelo [36].

2.10. Redes Neurais

Redes neurais são apresentadas como modelos computacionais inspirados no cérebro humano, compostos por neurônios interconectados que processam informações. Elas desempenham um papel fundamental no campo do aprendizado de máquina devido à sua capacidade de aprender a partir de dados e realizar tarefas complexas de classificação, regressão, reconhecimento de padrões, entre outras [7].

Uma rede neural é composta por camadas de neurônios, incluindo uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída. Cada neurônio em uma camada está conectado a todos os neurônios da camada anterior e da camada seguinte, e cada conexão possui um peso associado que determina a força da influência do neurônio de entrada sobre o neurônio de saída. Durante o treinamento, os pesos das conexões são ajustados iterativamente para minimizar uma função de custo, com o objetivo de melhorar o desempenho do modelo [7].

As redes neurais são importantes porque são capazes de aprender representações complexas e não lineares dos dados, o que as torna muito eficazes em uma ampla gama de problemas de aprendizado de máquina. Elas têm sido especialmente bem-sucedidas em problemas de visão computacional, processamento de linguagem natural, reconhecimento de padrões, entre outros. Além disso, o desenvolvimento de técnicas de treinamento mais avançadas, como o *backpropagation* e o uso de algoritmos de otimização, tem contribuído significativamente para o sucesso das redes neurais em diversas aplicações [7].

São diversas técnicas associadas às redes neurais, incluindo diferentes arquiteturas de redes (como redes neurais *feedforward*, redes neurais convolucionais, redes neurais recorrentes), algoritmos de treinamento (como *backpropagation*, *gradient descent*, e técnicas de regularização), e funções de ativação (como ‘sigmoid’, ‘tanh, relu’ e ‘softmax’). Além disso, existem técnicas avançadas como *dropout*, *batch normalization*, e *transfer learning* para melhorar o desempenho e a robustez dos modelos de redes neurais [7].

2.11. BERT

O BERT (*Bidirectional Encoder Representations from Transformers*) é um modelo de linguagem pré-treinado que se destaca por sua capacidade de capturar o contexto bidirecional das palavras em uma sentença. Ele é pré-treinado em grandes quantidades de texto não rotulado usando a arquitetura de *transformers*. Durante o pré-treinamento, o BERT é treinado em duas tarefas principais: previsão de palavras mascaradas (*Masked Language Model* - MLM) e previsão de sentenças seguintes (*Next Sentence Prediction* - NSP) [37].

A técnica de MLM envolve mascarar uma certa porcentagem de palavras em uma frase de entrada e treinar o modelo para prever as palavras mascaradas com base no contexto restante. Isso permite que o BERT capture o contexto bidirecional das palavras em uma sentença. A técnica de NSP treina o modelo para prever se uma frase é a próxima frase em um par de frases, o que ajuda o BERT a entender melhor a relação entre as frases em um texto [37].

O BERT é importante porque alcançou resultados de estado-da-arte em uma variedade de tarefas de processamento de linguagem natural, incluindo classificação de texto, tradução automática, geração de respostas, entre outras. Sua capacidade de capturar o contexto bidirecional das palavras em uma sentença permite que ele produza representações de texto mais ricas e informativas, levando a um melhor desempenho em uma variedade de aplicações [37].

2.12. Ajuste de hiperparâmetros

Hiperparâmetros em aprendizado de máquina são variáveis que são definidas antes do início do processo de treinamento e regulam diversos aspectos do comportamento do algoritmo de aprendizado. Diferente dos parâmetros do modelo, que são determinados pelos dados durante o treinamento, os hiperparâmetros são fatores externos que afetam a forma como o modelo descobre e generaliza padrões a partir dos dados. A seleção de hiperparâmetros pode ter um impacto considerável no desempenho de um modelo, na taxa de convergência e na capacidade de evitar *overfitting*. Assim, a otimização de hiperparâmetros pode melhorar o desempenho de um modelo de aprendizado de máquina. Há algumas técnicas de otimização de hiperparâmetros para diferentes modelos de aprendizado de máquina entre elas a busca manual, a busca em grade (*Grid Search*), a busca aleatória (*Random Search*) e a otimização bayesiana [38].

A busca manual é um método básico de ajuste de hiperparâmetros e implementado por ajuste 100% manual, muito utilizado por estudantes e pesquisadores. O processo consiste em testar muitos valores de hiperparâmetros possíveis com base na experiência, em suposições ou na análise de resultados previamente avaliados, sendo repetido até que este resultado seja satisfeito. Essa abordagem requer uma quantidade suficiente de conhecimento prévio para identificar hiperparâmetros ideais, o que torna inviável para muitos problemas devido a vários fatores, como um grande número de hiperparâmetros, modelos complexos, e avaliações de modelos demoradas [39].

A busca em grade é uma pesquisa exaustiva ou um método de força bruta que avalia todas as combinações de hiperparâmetros dadas à grade de configurações. Funciona avaliando o produto cartesiano a partir de um conjunto finito de valores que o usuário pode definir. Ele tem uma desvantagem, que é sua ineficiência no espaço de configuração de hiperparâmetros de alta dimensionalidade, pois o número de receitas aumenta exponencialmente com o número de hiperparâmetros [39].

Conforme mostra a Figura 4, a busca aleatória é semelhante à busca em grade, mas em vez de testar todos os valores no espaço de busca, seleciona aleatoriamente um número predefinido de amostras entre os limites superior e inferior como candidatas valores de hiperparâmetros e, em seguida, treina esses os candidatos até que o orçamento definido esteja esgotado [39].

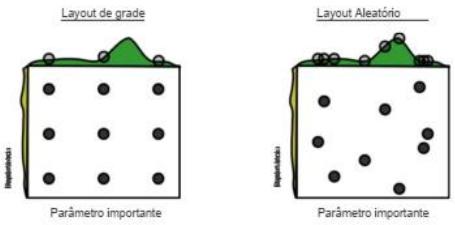

Fig. 4: Busca e grade aleatórias. Fonte: Adaptado de [48].

A otimização bayesiana é um algoritmo iterativo com dois ingredientes principais: um modelo substituto probabilístico e uma função de aquisição para decidir qual ponto avaliar em seguida. Em cada iteração, o modelo substituto é ajustado a todas as observações da função alvo feitas até o momento. Em seguida, a função de aquisição, que utiliza a distribuição preditiva do modelo probabilístico, determina a utilidade de diferentes pontos candidatos, negociando exploração e aproveitamento [40].

2.13. Validação cruzada

A validação cruzada é uma técnica fundamental em aprendizado de máquina para avaliar a capacidade de generalização de um modelo. Ela é importante porque fornece uma estimativa mais confiável do desempenho de um modelo em dados não vistos, o que é crucial para garantir que o modelo seja capaz de generalizar bem para novos casos [35].

O processo de validação cruzada envolve dividir o conjunto de dados disponível em várias partes, ou dobras. Em seguida, o modelo é treinado em uma combinação dessas dobras e avaliado em outra dobra que não foi usada durante o treinamento. Esse processo é repetido várias vezes, de modo que cada dobra seja usada como conjunto de teste em pelo menos uma iteração. Algumas técnicas comuns de validação cruzada estão descritas a seguir.

a) Validação Cruzada K-Fold: Divide o conjunto de dados em k dobras de tamanho igual. O modelo é treinado em k-1 dobras e avaliado na dobra restante. Esse processo é repetido k vezes, com cada dobra sendo usada como conjunto de teste uma vez.

b) *Leave-One-Out Cross-Validation* (LOOCV): É uma forma extrema de validação cruzada K-Fold, onde k é igual ao número de instâncias no conjunto de dados. Nesse caso, o modelo é treinado em todas as instâncias, exceto uma, que é usada como conjunto de teste.

c) Validação Cruzada Estratificada: Garante que a distribuição das classes em cada dobra seja semelhante à distribuição original do conjunto de dados. Isso é particularmente útil em problemas de classificação com classes desbalanceadas.

Essas técnicas de validação cruzada ajudam a mitigar o viés na estimativa do desempenho do modelo, fornecendo uma avaliação mais confiável do quão bem o modelo generaliza para novos dados [35].

2.14. Avaliação

2.14.1. Métricas

Métricas de avaliação são medidas quantitativas que nos ajudam a entender o desempenho de modelos de aprendizado de máquina em tarefas específicas. Essas métricas são essenciais para avaliar o quão bem um modelo está se saindo e para comparar diferentes modelos entre si [7].

A **acurácia** é uma medida da proporção de previsões corretas feitas pelo modelo em relação ao total de previsões feitas. Em termos mais simples, é a porcentagem de exemplos classificados corretamente pelo modelo em relação ao total de exemplos. Ela é uma das métricas mais básicas e intuitivas para avaliar o desempenho de um modelo de classificação fornecendo uma medida geral da qualidade do modelo em fazer previsões corretas e é amplamente utilizada em uma variedade de aplicações de machine learning. A acurácia é calculada simplesmente dividindo o número de previsões corretas pelo total de previsões feitas pelo modelo [17].

$$\text{Acurácia} = \frac{\text{Número de previsões corretas}}{\text{Total de previsões}}$$

No entanto, é importante estar ciente de suas limitações, especialmente em conjuntos de dados desbalanceados, onde a acurácia pode não fornecer uma imagem precisa do desempenho do modelo. Em tais casos, outras métricas, como precisão, *recall* e F1-*score*, podem ser mais informativas [58].

A **precisão** é uma métrica básica que mede a proporção de exemplos classificados corretamente pelo modelo em relação ao total de exemplos. A precisão varia de 0 a 1, onde 1 representa uma precisão perfeita (todos os exemplos positivos são corretamente classificados) e 0 representa uma precisão mínima (nenhum exemplo positivo é corretamente classificado). É uma medida simples e intuitiva, mas pode ser enganosa em problemas com classes desbalanceadas [7].
A fórmula para calcular a precisão é:

$$\text{Precisão} = \frac{\text{Verdadeiros Positivos (TP)}}{\text{Verdadeiros Positivos (TP)}+\text{Falsos Positivos (FP)}}$$

Onde:
Verdadeiros Positivos (TP): Número de exemplos que foram corretamente classificados como positivos. Falsos Positivos (FP): Número de exemplos que foram erroneamente classificados como positivos (quando deveriam ser negativos) [7].

O ***recall***, também conhecido como sensibilidade, mede a proporção de exemplos positivos que foram corretamente identificados pelo modelo em relação ao total de exemplos positivos reais. É uma métrica importante em problemas onde a identificação correta dos positivos é fundamental. Assim como a precisão, o *recall* também varia de 0 a 1, onde 1 representa um *recall* perfeito (todos os exemplos positivos são corretamente identificados) e 0 representa um *recall*

mínimo (nenhum exemplo positivo é corretamente identificado) [7].

A fórmula para calcular o recall é:

$$\text{Recall} = \frac{\text{Verdadeiros Positivos (TP)}}{\text{Verdadeiros Positivos (TP)}+\text{Falsos Negativos (FN)}}$$

Onde:
Verdadeiros Positivos (TP): Número de exemplos que foram corretamente classificados como positivos. Falsos Negativos (FN): Número de exemplos que foram erroneamente classificados como negativos (quando deveriam ser positivos) [7].

O **F1-*score*** é uma medida que combina precisão e *recall* em uma única métrica, sendo útil em problemas com classes desbalanceadas. Ele é calculado como a média harmônica entre precisão e *recall* [7].

A fórmula para calcular o F1-*score* é:

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

Onde:
Precisão é a métrica de precisão, calculada como o número de verdadeiros positivos (TP) dividido pelo total de positivos previstos (TP + FP). *Recall* é a métrica de *recall*, calculada como o número de verdadeiros positivos (TP) dividido pelo total de positivos reais (TP + FN) [7].

A **curva ROC** (*Receiver Operating Characteristic*) e a área sob a curva (ROC-AUC) são métricas amplamente utilizadas em problemas de classificação binária. A curva ROC mostra a taxa de verdadeiros positivos em função da taxa de falsos positivos para diferentes pontos de corte, enquanto a área sob a curva fornece uma medida geral da capacidade discriminativa do modelo [7].

Para calcular a curva ROC, precisamos primeiro calcular a TPR (Taxa de Verdadeiros Positivos) e a FPR (Taxa de Falsos Positivos) em vários limiares de classificação [7].

A Taxa de Verdadeiros Positivos (TPR) é calculada pela fórmula:

$$TPR = \frac{TP}{TP+FN}$$

Onde:
TP é o número de verdadeiros positivos (amostras positivas corretamente classificadas). FN é o número de falsos negativos (amostras positivas erroneamente classificadas como negativas) [7].

A Taxa de Falsos Positivos (FPR) é calculada pela fórmula:

$$FPR = \frac{FP}{FP+TN}$$

Onde:
FP é o número de falsos positivos (amostras negativas erroneamente classificadas como positivas). TN é o número de verdadeiros negativos (amostras negativas corretamente classificadas) [7].

Depois de calcular a TPR e a FPR em diferentes limiares de classificação, a curva ROC é plotada com TPR no eixo y e FPR no eixo x. Cada ponto na curva ROC representa o desempenho do modelo em um determinado limiar de classificação [7].

2.14.2. Matriz de confusão

A matriz de confusão é uma importante ferramenta na avaliação do desempenho de modelos de aprendizado de máquina. Consiste em uma tabela que mostra o desempenho do modelo ao prever as classes verdadeiras em relação às classes previstas possuindo quatro células principais:

	Classe Positiva	Classe Negativa
Previsto Positivo	TP	FP
Previsto Negativo	FN	TN

Verdadeiro Positivo (*True Positive* - TP): Instâncias que foram corretamente classificadas como positivas.
Falso Positivo (*False Positive* - FP): Instâncias que foram erroneamente classificadas como positivas (quando deveriam ser negativas).
Falso Negativo (*False Negative* - FN): Instâncias que foram erroneamente classificadas como negativas (quando deveriam ser positivas).
Verdadeiro Negativo (*True Negative* - TN): Instâncias que foram corretamente classificadas como negativas.

A matriz de confusão fornece uma visão detalhada do desempenho do modelo, permitindo calcular diversas métricas de avaliação, tais como precisão, *recall* e F1-*score* sendo útil para entender onde o modelo está acertando e onde está errando em suas previsões, ajudando os desenvolvedores a identificarem áreas de melhoria e a ajustar o modelo conforme o necessário tendo por base o comportamento em relação aos dados de teste e validação [7].

3. METODOLOGIA

Para atingir o objetivo proposto, a comparação do desempenho entre modelos de aprendizado de máquina na tarefa de classificação de lançamentos bancários, adotou-se uma abordagem metodológica estruturada em etapas clássicas de projetos de ciência de dados: obter os dados, explorar os dados, preparar os dados, selecionar modelos promissores e aperfeiçoá-los [7].

Respeitando as particularidades da legislação, os conjuntos de dados utilizados passaram por um meticuloso processo de seleção de atributos e anonimização, visando garantir a conformidade com as normas de proteção de dados e privacidade dos envolvidos.

Dada a reduzida capacidade computacional dos equipamentos disponíveis para a realização dos estudos (processador Intel Core i5-9500T CPU @ 2.20GHz, 8,00 GB de RAM instalada e sistema operacional Windows 10 Pro), optou-se pela redução do número de registros de duas formas: apenas os registros das operações de crédito foram tratados e foram removidos os registros em duplicidade.

Uma vez que o pré-processamento dos dados foi executado e utilizando-se os mesmos conjuntos de dados, a metodologia tomou duas rotas distintas, descritas abaixo.

a) O treinamento/validação e teste de modelos baseados em **algoritmos tradicionais** (SVM, *Decision Tree*, Regressão Logística e KNN). Tais modelos foram escolhidos por serem os mais utilizados em problemas de classificação, por sua simplicidade de entendimento e interpretação, por exigirem menos recursos computacionais e por sua facilidade de implementação [7] [21] [32] [33]. Já os **algoritmos de *ensemble*** utilizados neste trabalho foram escolhidos por sua capacidade de melhorar os resultados das métricas de avaliação, detectar padrões complexos e mitigar vieses individuais [34] [36].

b) O ajuste/validação e teste de modelos pré-treinados **BERT** (Multilingual e Português) na tarefa de classificação de lançamentos. O modelo foi escolhido por ser um dos mais populares e por sua alta precisão e eficácia em tarefas como processamento de linguagem natural e classificação de textos [56].

Utilizando-se do mesmo conjunto de métricas (acurácia, precisão, *recall* e F1-*score*), os melhores resultados de cada rota foram comparados para tomada da decisão final.

A figura 5 ilustra as etapas da metodologia adotada: obtenção, exploração e preparação dos dados, treinamento, validação, ajuste e teste dos modelos e comparação dos resultados.

3.1. Obtenção dos dados

Os dados analisados neste estudo referem-se às movimentações bancárias em contas de pessoas físicas e jurídicas investigadas pelo MPBA. Esses dados são coletados através do Sistema de Investigação de Movimentações Bancárias (SIMBA), desenvolvido pela Secretaria de Perícia, Pesquisa e Análise (SPPEA), um órgão da Procuradoria-Geral da República (PGR). O SIMBA foi desenvolvido para o recebimento e processamento, de forma padronizada e segura, de dados decorrentes do afastamento judicial do sigilo financeiro [41].

O MPBA é conveniado do SIMBA desde dezembro de 2009 e, a partir de 2010, os dados referentes às movimentações bancárias de investigados são transmitidos pelas instituições financeiras via SIMBA e armazenados nos bancos de dados locais no MPBA.

Foram feitas duas extrações de dados em momentos distintos. A primeira, realizada em 12 de dezembro de 2023, foi referente aos dados recebidos entre outubro de 2010 e dezembro de 2023. A partir desta extração foi gerado um arquivo CSV com 21.608.406 registros de movimentações bancárias que serão utilizados para **treinamento/validação** dos modelos. A segunda extração foi realizada em 19 de abril de 2024 e refere-se aos dados recebidos entre janeiro de 2024 e abril de 2024. A partir desta segunda extração foi gerado um arquivo com 1.014.432 registros que serão utilizados para **teste** dos modelos.

Os dois arquivos CSV extraídos são compostos por 18 campos presentes em três arquivos transmitidos pelas instituições financeiras: 'TITULARES' (identifica as pessoas, naturais ou jurídicas, titulares, representantes legais e procuradores das contas bancárias que tiveram o sigilo bancário afastado), 'EXTRATO' (identifica os lançamentos relacionados às contas investigadas) e 'ORIGEM_DESTINO' (identifica a origem ou o destino de recursos relacionados a um lançamento informado no arquivo 'EXTRATO' [42].

Após a obtenção dos arquivos CSV iniciais com os dados das movimentações bancárias, foram necessárias algumas etapas de processamento para assegurar a anonimicidade e conformidade com as normas de proteção de dados [43] [44]. Os dados contidos nesses arquivos foram submetidos a uma rigorosa limpeza e transformação com o intuito de garantir a privacidade dos indivíduos envolvidos.

Nos processos de limpeza e transformação, os dados presentes nos dois arquivos CSV iniciais foram submetidos ao mesmo tratamento. Os dados sensíveis foram substituídos por categorias ou simplesmente eliminados. Outros dados foram transformados para adequação ao processo de treinamento.

3.2. Exploração e preparação dos dados

No primeiro momento, foi realizada a análise do percentual de valores preenchidos e ausentes nos 18 campos, conforme ilustrado na figura 6.

A partir deste ponto, iniciou-se o processo de preparação dos arquivos que serviriam de base para o treinamento/validação e teste de nossos modelos. Utilizando-se de técnicas de programação em linguagem *python*, foram executados tratamentos para a criação de arquivos intermediários (figura 7) e dos arquivos definitivos (figura 8). O objetivo dessas etapas foi a limpeza de inconsistências eventualmente encontradas dos arquivos, a redução do número de registros e a garantia de anonimização dos dados.

Uma limpeza inicial resultou no descarte de 613 registros do arquivo de treino/validação e 20 registros do arquivo de teste. Tais registros foram descartados devido à presença do caractere ponto e vírgula (;) em alguns dos campos, algo que desalinharia as colunas do registro. Dado o reduzido número de ocorrências, optou-se por, simplesmente, desprezar esses registros.

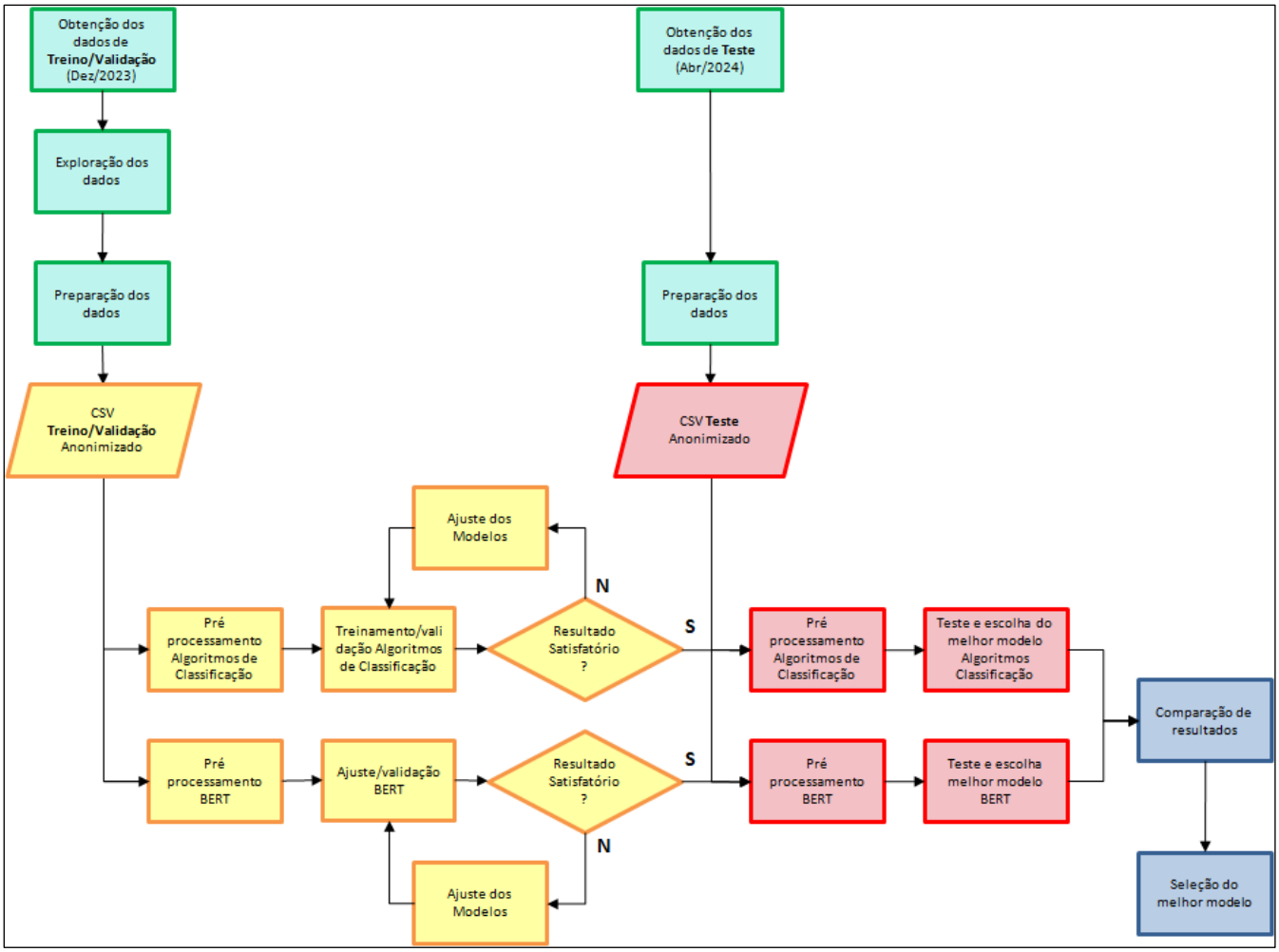

Fig. 5: Diagrama geral da metodologia. Fonte: Autores.

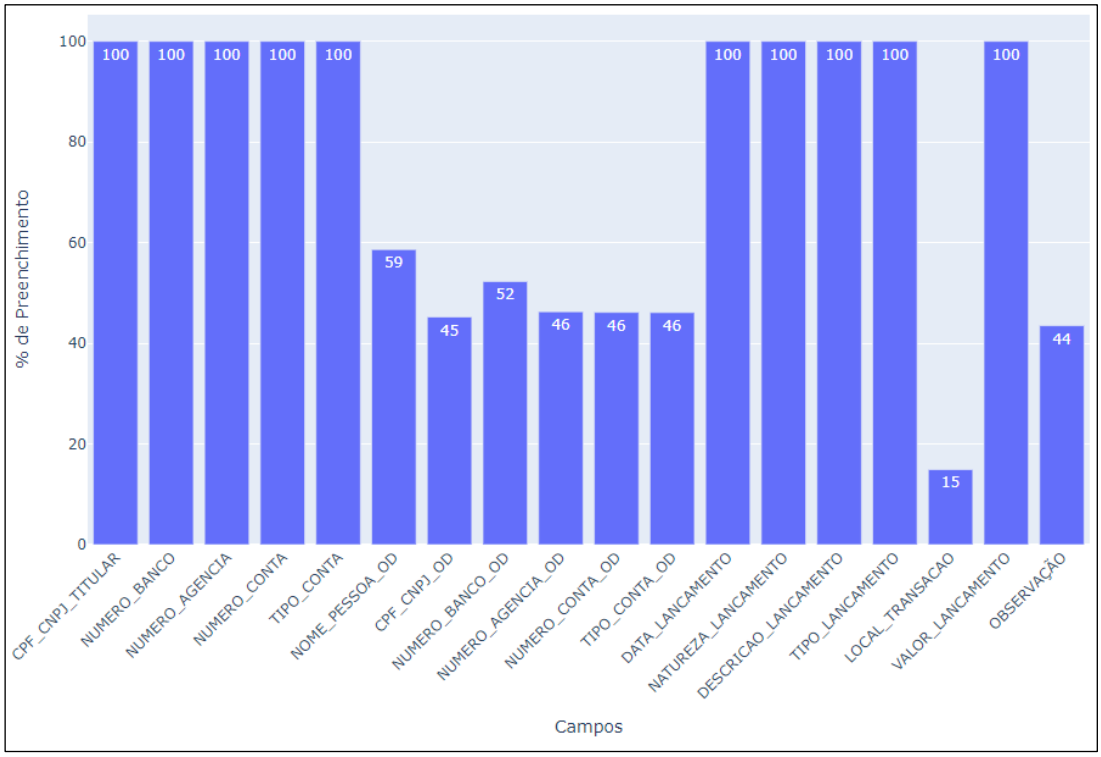

Fig. 6: Percentual de preenchimento por campo. Fonte: Autores.

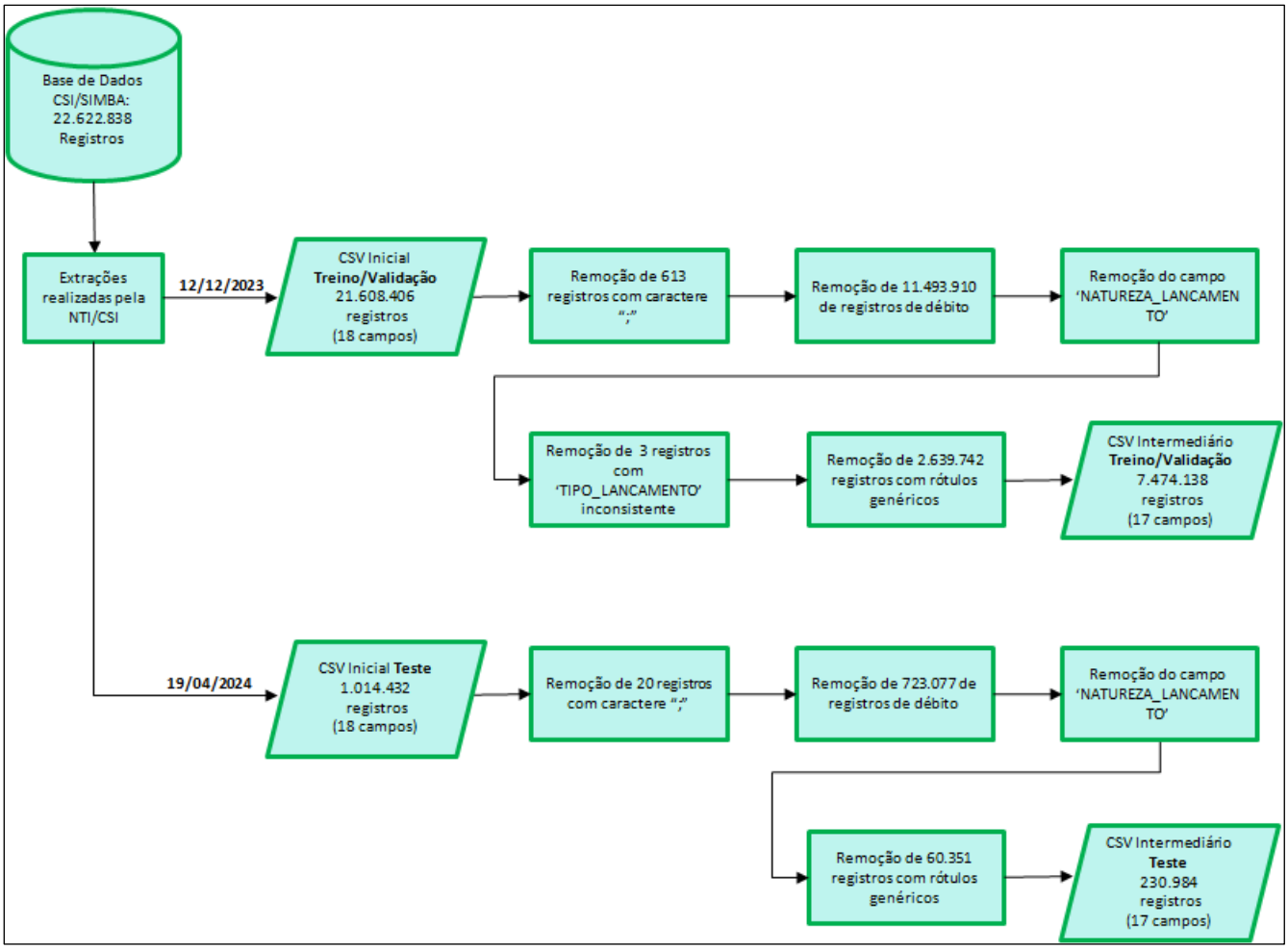

Fig. 7: Etapas da geração dos arquivos intermediários de treino/validação e teste. Fonte: Autores.

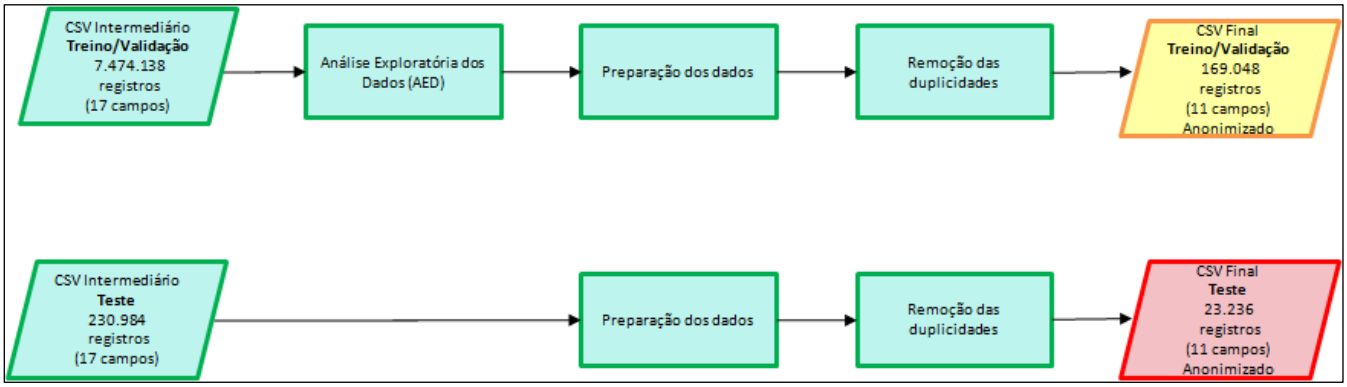

Fig. 8: Etapas da geração dos arquivos finais (anonimizados) de treino/validação e teste. Fonte: Autores.

Em seguida, pensou-se em encontrar um campo que poderia separar o conjunto de dados ao meio. Foi escolhido o campo 'NATUREZA_LANCAMENTO' pelos motivos expostos a seguir.

a) A partir da documentação do sistema SIMBA, constata-se que este campo está diretamente relacionado à variável 'TIPO_LANCAMENTO'. Ou seja, quando 'NATUREZA_LANCAMENTO' = "D" (débito), os códigos de 'TIPO_LANCAMENTO' (variável alvo) devem ser sempre menores que "200".

b) A 'NATUREZA_LANCAMENTO' está preenchida em 100% dos casos.

c) A 'NATUREZA_LANCAMENTO' divide o conjunto de dados praticamente ao meio (figura 9).

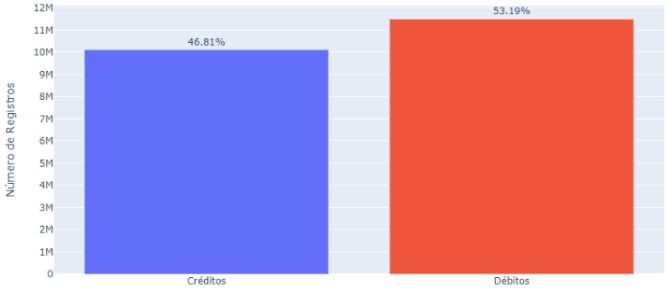

Fig. 9: Distribuição dos dados do campo 'NATUREZA_LANCAMENTO'. Fonte: Autores.

Logo, foi executada a exclusão dos lançamentos por natureza (campo 'NATUREZA_LANCAMENTO' = "D" para débitos, qualquer outro conteúdo foi considerado crédito). Dessa forma, o arquivo de treino/validação resultou em 10.113.883 registros referentes a créditos e o arquivo de teste resultou em 291.335 registros referentes a créditos. O atributo 'NATUREZA_LANCAMENTO' foi removido dos arquivos resultantes.

Numa varredura efetuada nos arquivos, foi detectada a presença de três registros, somente no arquivo de treino/validação, cujo 'TIPO_LANCAMENTO' seria menor que "200". Este tipo de inconsistência tem origem na própria instituição financeira. Versões mais recentes do sistema SIMBA passaram a inibir a possibilidade desse erro, efetuando a validação antes da transmissão dos arquivos para os órgãos investigadores. Esses três registros foram removidos do arquivo de treino/validação.

Na próxima etapa, foram descartados os registros com rótulos genéricos ('CD LANC'= "205" ou "999"), uma vez que para o treinamento do modelo são necessários registros devidamente rotulados pelas instituições financeiras.

Neste ponto, o arquivo de treino/validação passou a contar com 7.474.138 de registros referentes a créditos e o arquivo de teste passou a contar com 230.984 de registros referentes a créditos.

A seguir, detalha-se o tratamento dispensado a cada um dos demais campos, presentes nos arquivos intermediários.

a) 'CPF_CNPJ_TITULAR': CPF ou CNPJ do titular da conta investigada. Este campo foi removido dos arquivos finais para atender aos critérios de anonimização. Para que se preservasse o sigilo e, ainda assim, fosse extraída alguma informação que poderia agregar valor ao treinamento dos modelos, o número informado nesse campo foi submetido à função *python* 'brazilnum' [45] para que se determinasse o tipo da pessoa investigada (0=Pessoa Física e 1=Pessoa Jurídica). Apenas a resposta dessa função foi incluída no campo categórico 'TIPO PESSOA TITULAR' nos arquivos finais.

b) 'NUMERO_BANCO': Código de três dígitos, conforme o Sistema de Compensação de Cheques e Outros Papéis (COMPE), da instituição financeira na qual o investigado possui conta. Informação que isoladamente não identifica o titular da conta. Foi mantido como campo categórico 'BCO DO TITULAR' nos arquivos finais.

c) 'NUMERO_AGENCIA': Número da agência, sem dígito verificador, na qual o investigado possui conta, não foi incluído nos arquivos finais.

d) 'NUMERO CONTA': Número da conta investigada com o dígito verificador, não foi incluído nos arquivos finais.

e) 'TIPO_CONTA': Utiliza os seguintes códigos: "1" para conta corrente, "2" para conta de poupança, "3" para conta investimento e "4" para outros casos. Informação que isoladamente não identifica o titular da conta, foi mantido como campo categórico 'TIPO CTA DO TITULAR' nos arquivos finais.

f) 'NOME_PESSOA_OD' e 'CPF_CNPJ_OD': Nome e número do CPF ou CNPJ da pessoa que efetuou a transação com o investigado (origem ou destino da transação, conforme a natureza da operação). Estes campos não foram incluídos nos arquivos finais, mas, a partir dessas informações combinadas foi criado o campo categórico 'TIPO INFO TERC' (tipo de informação sobre o terceiro) nos arquivos anonimizados, com o seguinte domínio: "0" para Pessoa Física, "1" para Pessoa Jurídica, "2" caso o CPF ou CNPJ informado seja inválido, "3" indicando que só foi informado o campo referente ao nome, "4" indicando que nada informado, "5" indicando que foi informado um CNPJ igual ao CNPJ do banco do investigado, "6" indicando que foi informado um CNPJ igual ao CNPJ de outro BANCO e "7" indicando que foi informado um CPF ou CNPJ igual ao do próprio investigado.

g) ‘NUMERO_BANCO_OD’: Código COMPE da instituição financeira que enviou ou recebeu recursos da conta investigada. Informação que isoladamente não identifica o terceiro titular da conta. Foi mantido como campo categórico ‘BCO DO TERC’ nos arquivos finais. Os registros não preenchidos foram considerados como uma categoria: “não informado”.

h) ‘NUMERO_AGENCIA_OD’, ‘NUMERO_CONTA_OD' e ‘TIPO_CONTA_OD': Número da agência (sem dígito verificador), número da conta (com o dígito verificador) e tipo da conta que enviou ou recebeu recursos da conta investigada. Estes campos não foram incluídos nos arquivos finais, mas, a partir dessas informações combinadas foi criado o campo categórico ‘TIPO INFO CTA TERC’ (tipo de informação sobre a conta do terceiro) nos arquivos finais, com o seguinte domínio: “0” indicando que a conta é igual à conta do titular, “1” indicando que foi informada outra conta e “2” indicando que a conta do terceiro não foi informada (faltou banco, agência ou conta).

i) ‘DATA_LANCAMENTO’: Data em que foi verificada a realização do lançamento na conta investigada. Foi extraída a porção do ano como campo categórico ‘ANO’ para inclusão nos arquivos finais. A figura 10 facilita a compreensão da distribuição de lançamentos investigados por ano.

j) ‘TIPO_LANCAMENTO’: Código do tipo da transação (em 03 caracteres), preenchido conforme Anexo da CARTA-CIRCULAR Nº 3.454. **É a variável alvo**. Foi mantido como campo ‘CD LANC’ nos arquivos finais. Na figura 11 observa-se a proporção do desbalanceamento das classes.

k) ‘DESCRICAO_LANCAMENTO’: Histórico da transação, descrição (em até 50 caracteres) do tipo de lançamento realizado. Foram removidos os caracteres numéricos do campo. Foi mantido como campo ‘HISTORICO’ nos arquivos finais. São 14.215 descrições (históricos) diferentes.

l) ‘LOCAL_TRANSACAO’: Local de realização da transação. O atributo foi adicionado recentemente ao layout de transmissão SIMBA e ainda não é muito utilizado pelas instituições financeiras, estando preenchido em apenas 24,41% dos casos. Dessa forma, não foi incluído nos arquivos finais.

m) ‘VALOR_LANCAMENTO’: Valor do lançamento em reais (R$). Ao observar-se o *boxplot* (figura 12) do campo, verificou-se disparidade na distribuição dos valores, com um máximo elevado e uma concentração significativa de dados em faixas inferiores (a linha vermelha indica a mediana, ou seja, o ponto em que o conjunto de dados se divide ao meio).

Para melhor observação da distribuição do campo ‘VALOR_LANCAMENTO’, foi utilizada a distribuição por quartis. Valores entre 0,00 e 200,00 (Q1), valores entre 200,01 e 747,88 (Q2-mediana), valores entre 747,89 e 2.300,00 (Q3) e valores superiores a 2.300,00 (Q4).

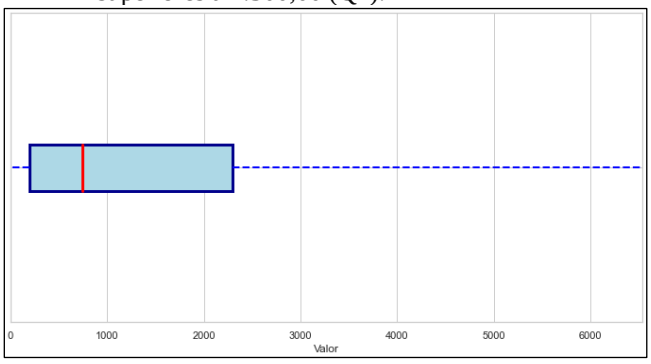

Fig. 12: Distribuição em quartis dos valores no campo ‘VALOR_LANCAMENTO’. Fonte: Autores.

Dessa forma, nos arquivos finais, o campo ‘VALOR_LANCAMENTO’ foi convertido no campo ‘FAIXA DE VALOR’ a partir da divisão do campo em quartis (“0” para valores no Q1, “1” para valores no Q2, “2” para valores no Q3 e “3” para valores no Q4).

n) ‘OBSERVAÇÃO’: Outras informações importantes (em até 250 caracteres). Foram removidos os caracteres numéricos do campo. Foi mantido como campo ‘OBSERV’ nos arquivos finais.

Finalmente, foram removidos os registros idênticos (duplicidades), resultando num arquivo de **treino/validação** com **169.048 registros** e num arquivo de **teste** com **23.236** registros. Cada arquivo contando com 11 atributos: ‘TIPO PESSOA TITULAR’, ‘BCO DO TITULAR’, ‘TIPO CTA DO TITULAR’, ‘TIPO INFO TERC’, ‘TIPO INFO CTA TERC’, ‘BCO DO TERC’, ‘ANO’, ‘HISTORICO, ‘OBSERV’, ‘FAIXA DE VALOR’ e, o alvo, ‘CD LANC’.

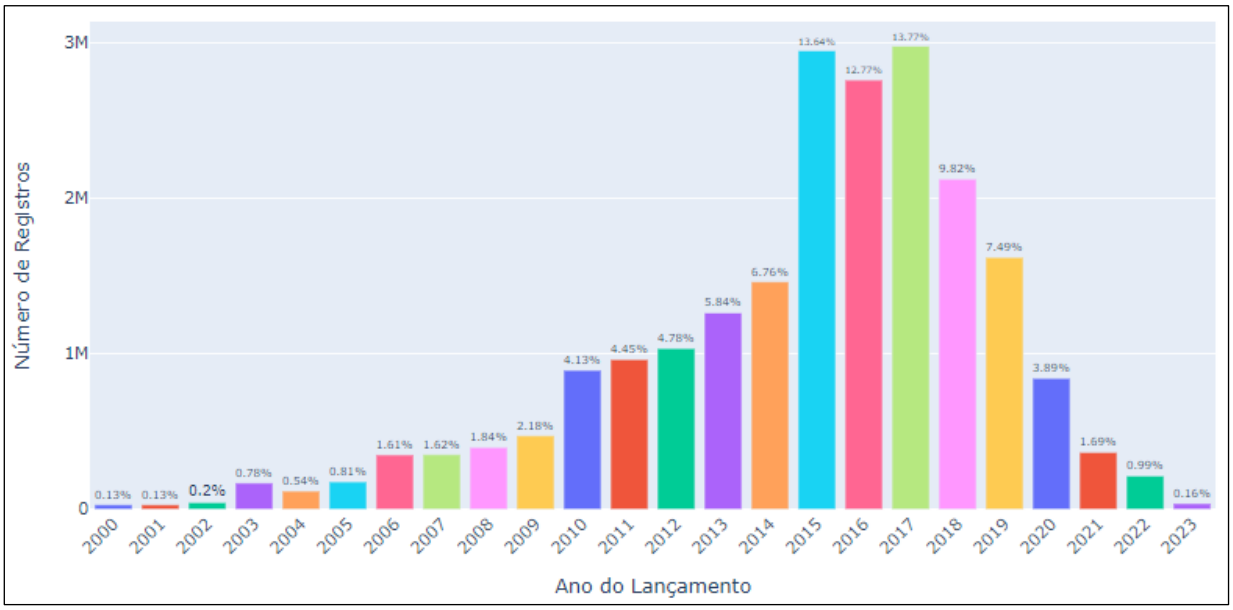

Fig. 10: Distribuição anual dos lançamentos investigados. Fonte: Autores.

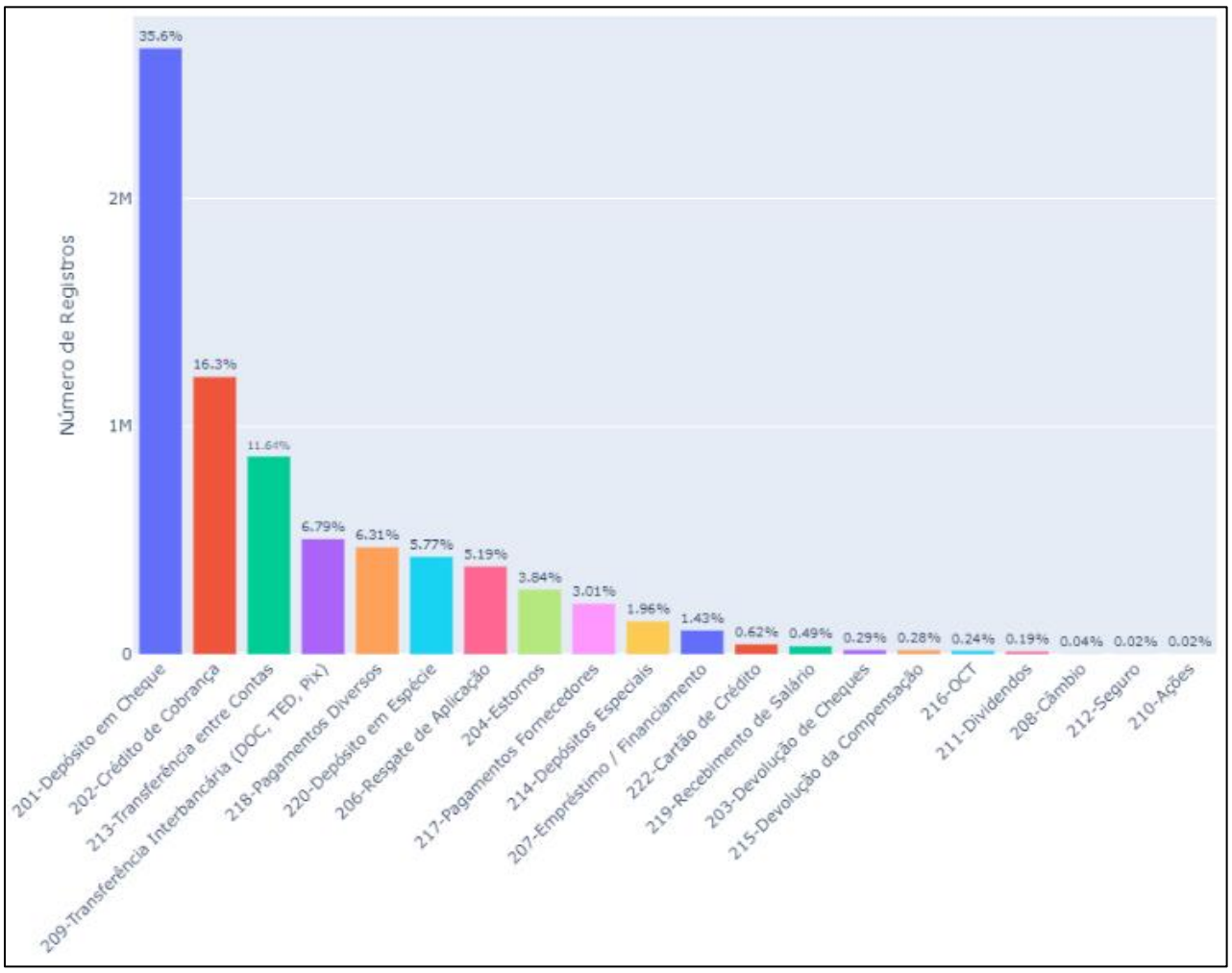

Fig. 11: Distribuição dos dados do campo ‘CD LANC’. Fonte: Autores.

A tabela 13 resume o tratamento dado aos campos obtidos do sistema SIMBA no processo de anonimização.

CAMPO NO ARQUIVO INICIAL (SIMBA)	AÇÃO	CAMPO NO ARQUIVO FINAL (ANONIMIZADO)	TIPO DA VARIÁVEL FINAL	DOMÍNIO
NATUREZA_LANCAMENTO	REMOVIDO			
CPF_CNPJ_TITULAR	SUBSTITUÍDO	TIPO PESSOA TITULAR	CATEGÓRICA	0=PF 1=PJ
NUMERO_BANCO	MANTIDO	BCO DO TITULAR	CATEGÓRICA	CÓDIGO DO BANCO COM 3 POSIÇÕES
NUMERO_AGENCIA	REMOVIDO			
NUMERO_CONTA	REMOVIDO			
TIPO_CONTA	MANTIDO	TIPO CTA DO TITULAR	CATEGÓRICA	1=CONTA CORRENTE 2=CONTA POUPANÇA 3=INVESTIMENTO 4=OUTRAS
NOME_PESSOA_OD	SUBSTITUÍDOS	TIPO INFO TERC	CATEGÓRICA	0=PF 1=PJ 2=NÃO IDENTIFICADO (CPF/CNPJ INVÁLIDO) 3=INCOMPLETO (SÓ NOME INFORMADO) 4=NADA INFORMADO 5=CNPJ DO BANCO DO INVESTIGADO 6=CNPJ DE OUTRO BANCO 7=CPF/CNPJ DO PRÓPRIO INVESTIGADO
CPF_CNPJ_OD				
NUMERO_BANCO_OD	MANTIDO	BCO DO TERC	CATEGÓRICA	CÓDIGO DO BANCO COM 3 POSIÇÕES
NUMERO_AGENCIA_OD	SUBSTITUÍDOS	TIPO INFO CTA TERC	CATEGÓRICA	0=IGUAL À CONTA DO TITULAR (BCO AG CTA) 1=OUTRA CONTA (BCO AG CTA) 2=NÃO INFORMADA (falta BCO, AG OU CTA)
NUMERO_CONTA_OD				
TIPO_CONTA_OD	REMOVIDO			
DATA_LANCAMENTO	SUBDIVIDIDO	ANO	CATEGÓRICA	ANO DO LANÇAMENTO
TIPO_LANCAMENTO	MANTIDO	CD LANC	CATEGÓRICA	
DESCRICAO_LANCAMENTO	MANTIDO	HISTORICO	STRING	TEXTO LIVRE
LOCAL_TRANSACAO	REMOVIDO			
VALOR_LANCAMENTO	SUBSTITUÍDO	FAIXA VALOR	CATEGÓRICA	0=MUITO BAIXO (ENTRE 0,00 E 199,99) 1=BAIXO (ENTRE 200,00 E 747,87) 2=MÉDIO (ENTRE 747,88 E 2.299,99) 3=ALTO (>= 2.300,00)
OBSERVAÇÃO	MANTIDO	OBSERV	STRING	TEXTO LIVRE

Tabela 13: Representação do "DE/PARA" dos campos. Fonte: Autores.

3.3. Seleção de modelos

3.3.1. Algoritmos de classificação

Para a seleção do melhor algoritmo de classificação foram idealizadas e executadas as seguintes etapas descritas a seguir.

a) Estabelecimento de **70 ciclos** de execução combinando **04 algoritmos de classificação** (SVM, Árvore de Decisão, Regressão Logística e KNN), **05 algoritmos de ensamble** (XgBoost, CatBoost, AdaBoost, Gradient Boosting e *Random Forest*), **03 métodos de pré-processamento** (limpeza dos dados, lematização e stematização), **05 métodos de vetorização/codificação** (BOW, TF-IDF, Word2Vec, FastText e GloVe) e **04 métodos de balanceamento de dados** (sem balanceamento, SMOTE, SMOTE_Tomek Links e atribuição de pesos às classes).

b) Execução dos 70 ciclos avaliando a performance dos modelos com **validação cruzada** e posterior **treinamento/validação** dos modelos, conforme figura 14. Seleção das 12 melhores combinações baseada nos maiores percentuais de F1-*score* nos resultados da validação (buscando obter o equilíbrio entre *precision* e *recall*, medidas indicadas para tarefas mutilasses) [7].

c) Execução do **ajuste dos hiperparâmetros** dos modelos selecionados nos 12 melhores ciclos de execução anteriores, conforme figura 15.

d) Execução de **24 ciclos de teste** com os dados nunca vistos (12 para os melhores modelos treinados SEM ajuste de hiperparâmetros e 12 para os modelos treinados COM ajuste de hiperparâmetros) do arquivo final **Teste** (anonimizado), conforme figura 16.

A execução de todos os ciclos de teste, envolvendo algoritmos de classificação, algoritmos de ensamble, métodos de pré-processamento, métodos de vetorização/codificação e métodos de balanceamento propostos, resultaria em 540 combinações possíveis, o que tornaria o estudo inviável. Assim, chegou-se a um subconjunto de 70 ciclos, à medida em que se consideravam as combinações que apresentavam melhor desempenho no conjunto de validação e menor tempo de processamento. Por exemplo, quando utilizado como método de vetorização o *Word Embeddings* foi utilizado apenas a atribuição de pesos como método de balanceamento, devido ao menor tempo processamento. O mesmo critério foi aplicado nos testes com os algoritmos de *ensemble*.

Nos próximos parágrafos, apresenta-se o detalhamento de alguns pontos da metodologia utilizada.

Três técnicas de **pré-processamento** foram aplicadas em 169.048 registros para testar e identificar qual delas apresentaria melhor desempenho com os dados, conforme relacionadas.

a) **Limpeza dos atributos** textuais 'HISTORICO' e 'OBSERV'. Utilizou-se expressões regulares para substituir caracteres especiais e pontuação por espaços em branco e remover espaços múltiplos. A biblioteca unidecode[1] foi empregada para eliminar acentos e converter os textos para minúsculas. O pacote nltk[2] foi utilizado para remover *stopwords* específicas do português, otimizando ainda mais o processamento dos dados.

b) A combinação dos procedimentos descritos no primeiro item com a **lematização**, buscando refinar o tratamento do texto ao reduzir as palavras a suas formas base.

c) A combinação dos procedimentos descritos no primeiro item com a **stematização**, técnica que reduz as palavras a seus radicais, simplificando a análise textual e potencialmente melhorando a comparação entre diferentes registros.

[1] *https://pypi.org/project/Unidecode/*

[2] *https://www.nltk.org/*

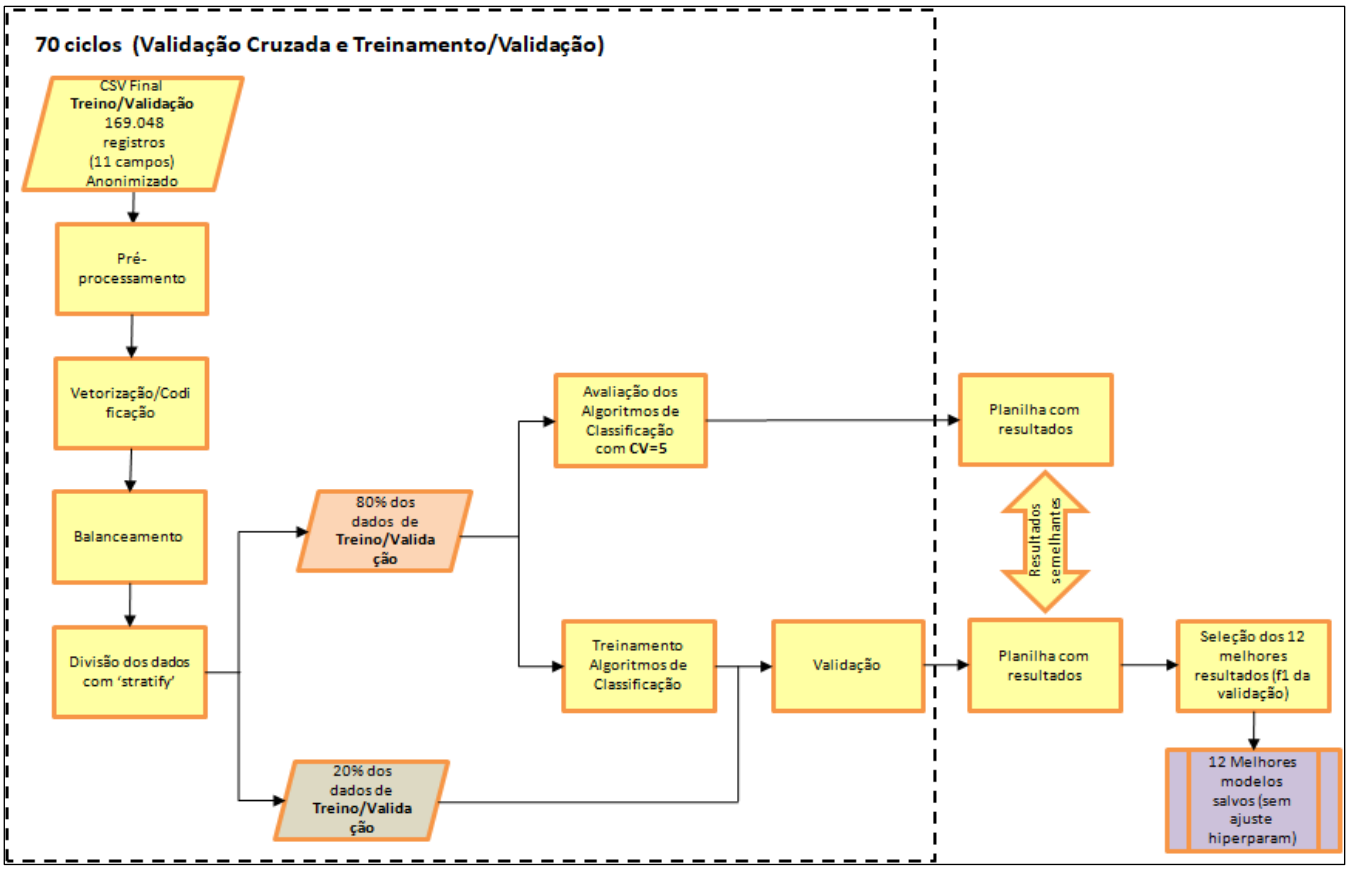

Fig. 14: Fluxo dos 70 ciclos de execução de validação cruzada e treinamento/validação. Fonte: Autores.

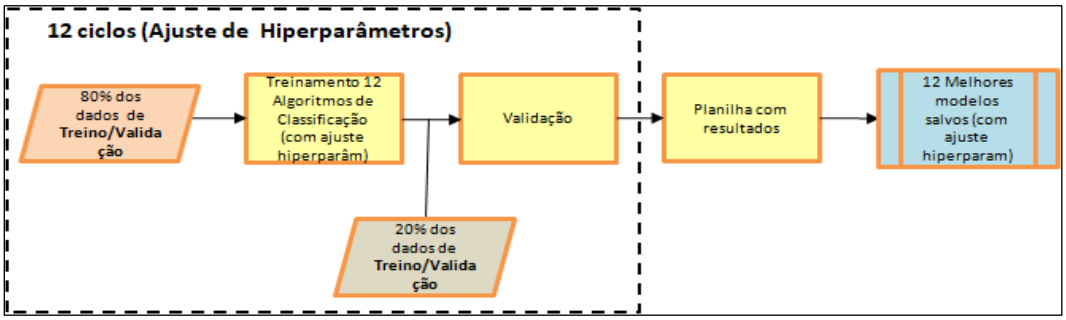

Fig. 15: Fluxo dos 12 ciclos de ajuste de hiperparâmetros. Fonte: Autores.

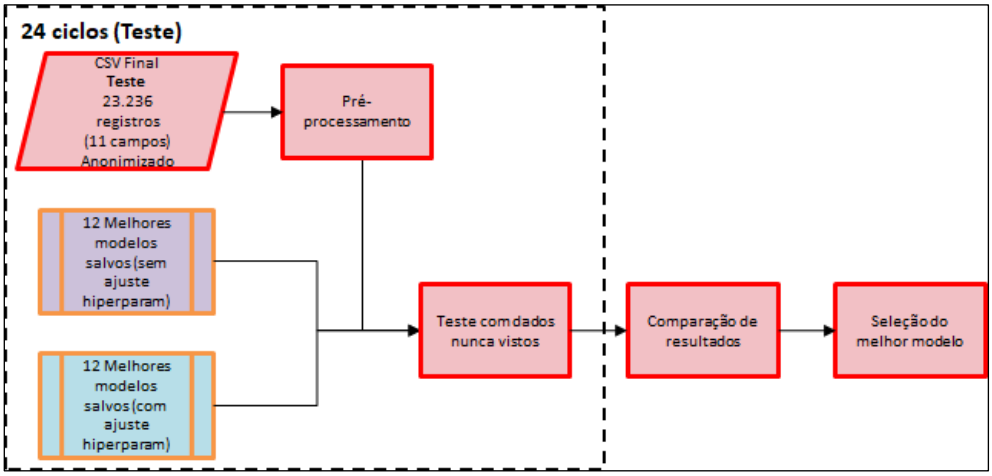

Fig. 16: Fluxo dos 24 ciclos de teste. Fonte: Autores.

Para reduzir o tamanho dos textos no atributo 'OBSERV', que tendem a ser longos e aumentam a carga de processamento, bem como para garantir a anonimicidade (considerando que algumas instituições financeiras podem incluir nomes de pessoas no campo 'OBSERVAÇÃO'), foram extraídas apenas as três primeiras palavras de cada *string*. Em seguida, criou-se o atributo 'HIST_OBSERV', combinando as colunas 'HISTORICO' e 'OBSERV'.

No processo de **codificação** e **vetorização** dos dados, as colunas categóricas e a coluna textual 'HIST_OBSERV' foram inicialmente identificadas para processamento. Definiu-se um transformador para aplicar a codificação *OneHotEncoder* nos atributos categóricos, sem a necessidade de parâmetros adicionais.

Optou-se por não usar o *LabelEncoder*, uma vez que os atributos categóricos não apresentam uma ordem implícita nos rótulos. Além disso, os resultados indicam que a codificação *one-hot* proporciona a maior precisão na maioria dos algoritmos de aprendizado de máquina, embora aumente o tempo de execução, especialmente em casos de alta cardinalidade dos atributos [27].

Nas técnicas de vetorização para os atributos textuais (por exemplo "CHEQUE DEVOLVIDO SEM FUNDO", "RECEBIMENTO FORNECEDOR", "DP BLQ BCOS"), foi utilizado o BOW (CountVectorizer) e o TF-IDF (TfidfVectorizer), com um número máximo de características variando entre 11.472 e 14.004, dependendo do tipo de pré-processamento, sem a especificação de outros parâmetros. Também foram exploradas outras técnicas de vetorização como *word embeddings* (Word2Vec, GloVe e FastText) utilizando modelos pré-treinados do Repositório de Word Embeddings do NILC [46]. Nos modelos pré-treinados, foram testadas as abordagens CBOW e Skip-gram para Word2Vec e CBOW para FastText, bem como o modelo GloVe, todos com uma dimensionalidade de 600, ou seja, cada palavra no vocabulário é representada por um vetor com 600 elementos. Em testes realizados com dimensionalidade aumentada (maior que 600), o tempo de processamento não compensou a melhora marginal nos resultados. Inicialmente, foram carregados os modelos pré-treinados e realizada a vetorização do texto, dividindo cada texto em palavras individuais ou tokens. Em seguida, procedeu-se à combinação dos vetores somados para testes. Por fim, esses vetores foram utilizados como entrada para os algoritmos de aprendizado de máquina.

Para lidar com conjuntos de **dados desbalanceados**, foram aplicadas técnicas de balanceamento, incluindo a sobreamostragem SMOTE, ajustada com parâmetros específicos: 'k_neighbors=5', 'sampling_strategy=auto', 'n_jobs=none' e 'random_state=42'. Além disso, utilizou-se uma combinação de técnicas de balanceamento, iniciando com o SMOTE combinado com aplicação do método Tomek Links, utilizando a parametrização padrão.

Por fim, foi aplicado o ajuste dos pesos das classes como método para balancear os dados, uma vez que isso reduziu significativamente o custo computacional dos treinamentos dos modelos. Modelos de Regressão Logística, SVM e Random Forest foram automaticamente configurados para ajustar os pesos das classes desbalanceadas, utilizando o parâmetro 'class_weight=balanced'. Para o XgBoost, foi usado o parâmetro 'scale_pos_weight'. Em contraste, para os modelos de Árvore de Decisão, KNN, Gradient Boosting, CatBoost e AdaBoost, os pesos foram ajustados manualmente, levando em consideração a frequência dos registros, usando 'sample_weight' e 'class_weights' [52].

Como exemplo da sobreamostragem SMOTE, a quantidade de registros subiu de 169.048 para 803.180. A figura 17 mostra as quantidades de registros por classe antes e depois do balanceamento com SMOTE.

Foi realizada a **divisão dos dados** em conjuntos de treinamento/validação. Definiu-se que 20% dos dados seriam reservados para o conjunto de validação, enquanto os 80% restantes seriam destinados ao treinamento. Foi utilizado para garantir a divisão estratificada, assegurando que a distribuição dos rótulos no conjunto dividido permanecesse semelhante à do conjunto de dados original, uma abordagem crucial especialmente em conjuntos de dados desbalanceados [51].

Para a realização da validação cruzada, quando os conjuntos de dados estavam balanceados, configurou-se 'K-Fold' e quando foram utilizados os conjuntos de dados desbalanceados, utilizou-se 'StratifiedKFold', todos com *n_splits* de 5 [47].

Dentre as 70 combinações executas, foram selecionadas 12 combinações (04 algoritmos) que apresentaram maiores percentuais de F1 nos resultados da validação. Para estes 12 casos foi realizado o **ajuste de hiperparâmetros** [60] [61] por meio do RandomizedSearchCV, conforme tabela 18.

Nesta abordagem metodológica, não foi utilizada o GridSearchCV devido à sua alta demanda computacional em comparação com o RandomizedSearchCV. Além disso, buscas aleatórias provaram ser mais eficientes do que as buscas em grade para a otimização de hiperparâmetros em vários algoritmos de aprendizagem. Isso ocorre porque as buscas em grade tendem a alocar muitos testes para explorar dimensões que são menos críticas. Portanto, a busca aleatória é mais adequada em espaços de alta dimensão e geralmente produz melhores modelos em menos tempo [48].

Também neste estudo foi adotada a abordagem *One-vs-Rest* (*OvR*) para calcular e plotar as curvas ROC, dado tratar-se de um problema de classificação multiclasse. Assim, foi feita a binarização dos rótulos de classe, ou seja, o cálculo da curva ROC considerou uma classe como positiva e todas as outras como negativas [7].

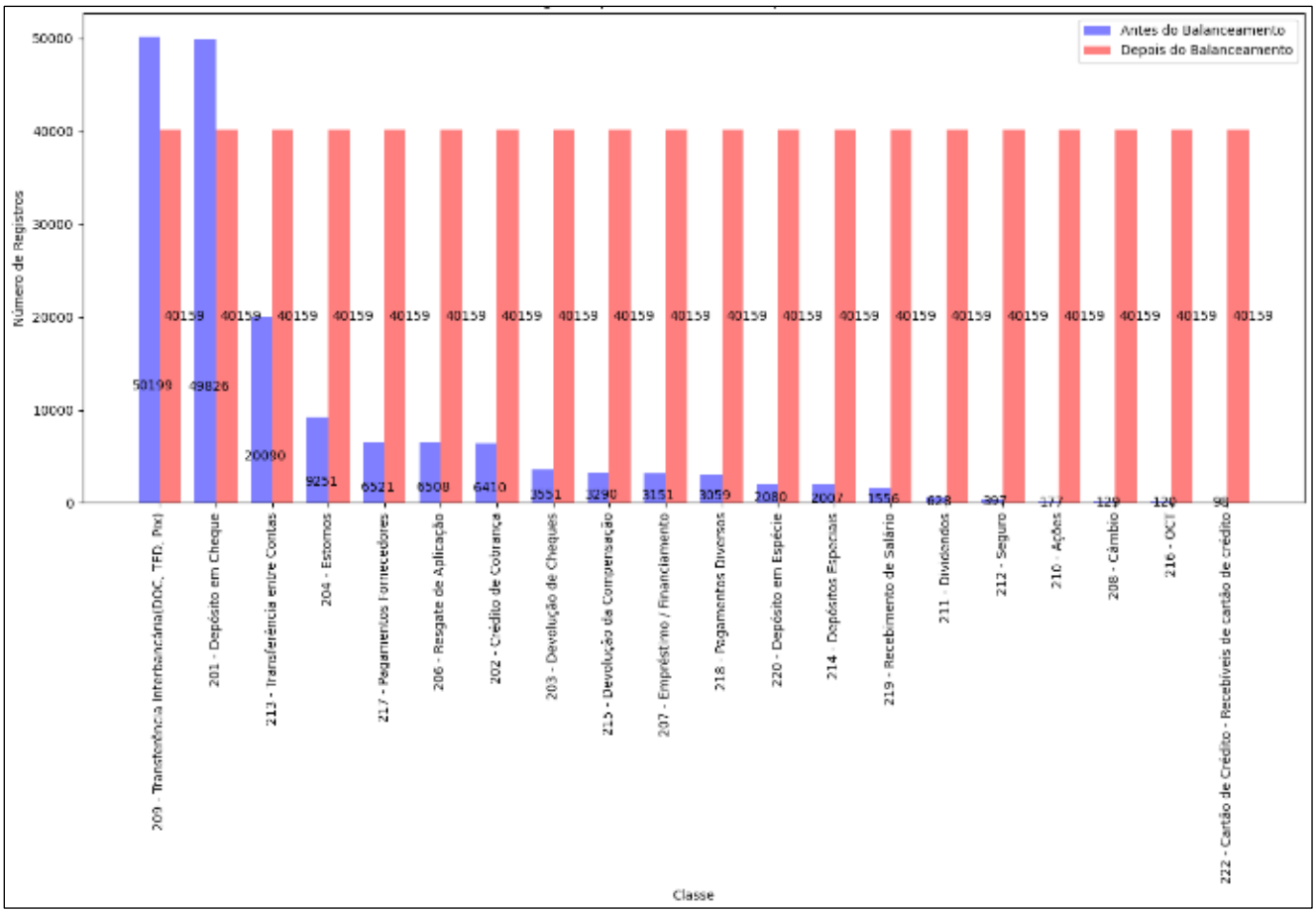

Fig. 17: Número de registros por classe antes e depois do balanceamento com SMOTE. Fonte: Autores.

Modelo	Hiperparâmetros
SVM	- C: [1, 10, 15, 20, 30]
	- *cache_size*: [50, 100, 200, 300]
	- *gamma*: [*scale*, auto, 0.01, 0.1, 1]
	- *kernel: [linear, rbf, poly]*
	- *probability: [True, False]*
Logistic Regression	- C: [0.001, 0.01, 0.1, 1, 10, 100]
	- *penalty: [l1, l2]*
	- *solver: [lbfgs, liblinear, saga]*
	- *max_iter*: [100, 200, 300]
	- *n_estimators*: [50, 100, 200]
Random Forest	- *max_features*: [auto, sqrt]
	- *max_depth*: [*None*, 10, 20, 50]
	- *min_samples_split*: [2, 5, 10]
	- *min_samples_leaf*: [1, 2, 4]
	- *max_depth*: [3, 4, 5, 6, 7]
XGBoost	- *learning_rate*: [0.01, 0.05, 0.1, 0.15, 0.2]
	- *n_estimators*: [100, 150, 200, 250, 300]
	- *min_child_weight*: [1, 3, 5]
	- *gamma*: [0.0, 0.1, 0.2, 0.3, 0.4]
	- *subsample*: [0.6, 0.7, 0.8, 0.9, 1.0]
	- *colsample_bytree*: [0.6, 0.7, 0.8, 0.9, 1.0]
	- *reg_alpha*: [0, 0.1, 0.5, 1.0]
	- *reg_lambda*: [0, 1.0, 10.0, 100.0]
	- *random_state*: [42]

Tabela 18: Relação de hiperparâmetros aplicados. Fonte: Autores.

3.3.2. BERT

Modelos pré-treinados, como o *BERT*, podem ser ajustados *(fine-tuning*) para a execução de tarefas específicas (*downstream tasks*) [49] como a proposta neste trabalho, que seria a classificação de lançamentos bancários em categorias.

Para a seleção da melhor configuração de ajustes do modelo pré-treinado foram idealizadas e executadas seguintes etapas (figura 19).

a) Estabelecimento de 04 ciclos de ajuste/validação combinando 02 modelos pré-treinados (‘*bert-base-uncased’* e ‘*neuralmind/bert-base-portuguese-cased’*) e 02 formas diferentes de ajuste de peso das classes (SEM ajuste e COM ajuste).

b) Execução de 04 ciclos de teste, submetendo os modelos previamente ajustados aos dados nunca vistos do arquivo final **Teste** (anonimizado).

Para o presente estudo, o ajuste dos modelos BERT exigiu etapas específicas de **pré-processamento** dos atributos. Visando diminuir a carga de processamento e, num esforço de garantia da anonimicidade (uma vez que algumas instituições financeiras, eventualmente, informam o nome da pessoa no campo ‘OBSERVAÇÃO’), optou-se por utilizar apenas as três primeiras palavras do atributo ‘OBSERV’ ao invés de utilizá-lo integralmente. Nos casos em que o atributo estivesse vazio, foi utilizada a literal “não informada” para composição da sentença.

Diferentemente dos algoritmos tradicionais, onde foi utilizado o *one-hot encoding* para os atributos categóricos e vetorização para os atributos textuais, os 10 atributos de cada linha de nosso arquivo de entrada foram convertidos em sentença específica para o modelo BERT. Utilizou-se um padrão onde cada peça de informação foi separada pelo token especial "[SEP]", e cada sentença iniciada com o token "[CLS]". Essa formatação é crucial porque o modelo BERT requer uma estrutura específica para processar eficientemente as informações [49]. A figura 20 exemplifica uma sentença formatada pelo processo descrito.

Na próxima etapa foi inicializado o tokenizador para o modelo BERT. Foram realizados testes com o ‘*bert-base-uncased’* (modelo fornecido pela Google que não diferencia maiúsculas de minúsculas) e o *'neuralmind/bert-base-portuguese-cased*' (versão do BERT adaptada e treinada para o português [50]. Nesse caso, foi utilizado o parâmetro ‘**do_lower_case**=True’ para que todas as entradas fossem convertidas em minúsculas).

Contou-se o número máximo de tokens em uma sentença (107) e, também, observou-se a distribuição do número de tokens por sentença de entrada. Com base nesses números, **o comprimento máximo das sequências de tokens** foi definido como 64. Eventualmente, algumas sentenças serão truncadas, mas a preocupação com o tempo de processamento foi decisiva.

Utilizando o tokenizador do BERT, os tokens de cada sentença foram convertidos para seus respectivos índices no vocabulário dos modelos BERT. Foram criadas máscaras de atenção para as sequências de índices de entrada. A máscara de atenção é uma lista onde cada posição contém o valor 1 ou 0, indicando se o token correspondente é um token válido (1) ou um token de preenchimento (0). Essas máscaras são importantes para que o modelo BERT possa diferenciar entre conteúdo útil e preenchimento (*filler*) durante o treinamento ou a inferência, otimizando, assim, o processamento e direcionando a atenção apenas aos tokens relevantes [49].

Em seguida, os conjuntos de dados (índices dos tokens e máscaras de atenção) foram divididos em conjuntos de treinamento e validação. A estratégia de divisão incluiu o uso do parâmetro ‘**stratify**’ [51], fundamental para garantir que a proporção das classes nos rótulos seja mantida igual tanto no conjunto de treinamento quanto no de validação. Foi feita a divisão de 20% dos dados que serão usados para validação, enquanto os restantes 80% serão utilizados para treinamento.

Como já se observou, as classes são desbalanceadas. Os arquivos que serão utilizados para treinamento não possuem registros duplicados, portanto, a remoção de registros (*undersampling*) removeria dados importantes para o treinamento. Por outro lado, a criação de registros adicionais (*oversampling* ou *data augmentation*) resultaria em um número excessivamente grande de registros, tornando o processo inviável com os recursos computacionais atualmente disponíveis. Dessa forma, optou-se por utilizar pesos para lidar com o desequilíbrio entre as classes [52]. Foi utilizada uma técnica que atribui maior importância às classes minoritárias. Um parâmetro ‘alpha’ foi utilizado para ajustar esse equilíbrio. O valor ‘alpha=1’ indica que todas as classes têm o mesmo peso e a diminuição do valor de ‘alpha’ aumenta a importância dada às classes minoritárias.

Conforme indicado pelos autores do BERT, para o ajuste fino (*fine-tuning*) do modelo, a maioria dos hiperparâmetros são herdados da etapa de pré-treinamento. No entanto, existem exceções: o tamanho do lote (*batch size*), a taxa de aprendizado (learning rate) e o número de épocas de treinamento, que podem variar de acordo com a tarefa específica em questão. Os autores sugerem uma gama de valores para esses três hiperparâmetros que geralmente apresentam bons resultados em várias tarefas. Eles também observam que conjuntos de dados grandes, com mais de 100 mil registros, como no caso deste estudo, tendem a ser menos sensíveis às variações na escolha desses hiperparâmetros [53]. Portanto, neste trabalho, adotamos os seguintes parâmetros: **‘Batch size**=32’, ‘**Learning rate**=2e-5’, e ‘**Número de épocas**=4’ para todos os ciclos de ajuste/validação.

Durante o treinamento, foi executado um loop de iteração sobre as épocas, calculando a perda e a acurácia para os conjuntos de treinamento e validação. Foram salvos os dados do modelo referentes à melhor época, bem como os dados do tokenizador, para posterior uso na etapa de teste. Os resultados do ajuste/validação e dos testes foram armazenados para análises posteriores [49].

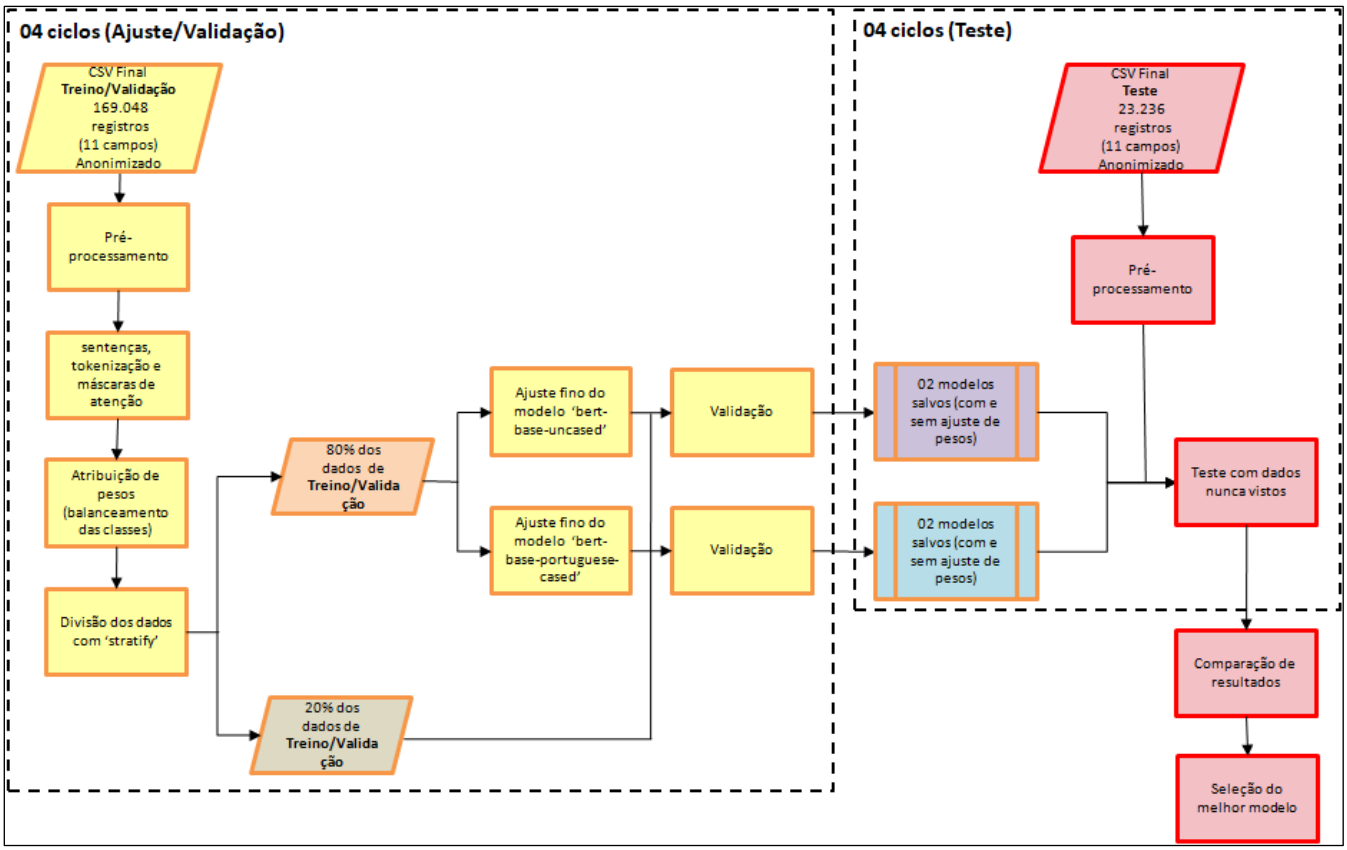

Fig. 19: Fluxo dos 04 ciclos de ajuste/validação e dos 04 ciclos de teste. Fonte: Autores.

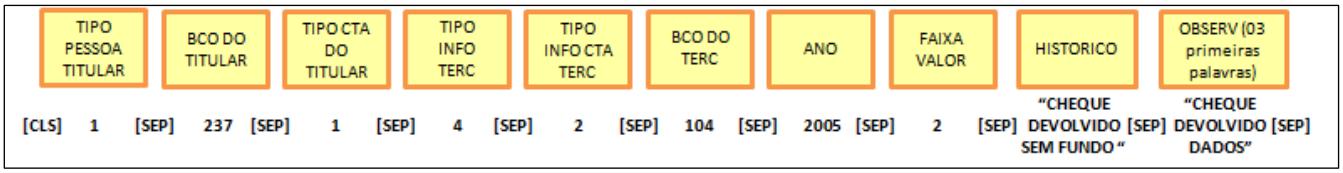

Fig. 20: Exemplo de sentença formatada para o modelo pré-treinado. Fonte: Autores.

4. RESULTADOS E DISCUSSÕES

4.1. Algoritmos de Classificação

De acordo com a metodologia, são apresentados na figura 21 os resultados dos 70 ciclos de execução de treinamento/ validação dos algoritmos de classificação.

Observou-se que os resultados das métricas da validação cruzada estão muito próximos dos resultados obtidos na validação tradicional, indicando que os modelos estão generalizando bem. Adicionalmente, o baixo desvio padrão (0.0011) da acurácia observado entres os *folds* de alguns testes de referência cruzada significam pontuações consistentes entre os diferentes *fold*s indicando que o modelo tem uma performance estável e não está altamente sensível a quais partes dos dados são usadas para treinamento ou validação.

De maneira geral, observa-se que os modelos XGBoost, Regressão Logística**,** SVM e Random Forest apresentaram os melhores resultados na métrica F1 da etapa de validação.
A combinação da remoção de *stopwords* com stemização ou lematização tende a melhorar ligeiramente os resultados, mas não de forma significativa para todos os modelos. Modelos usando BOW geralmente mostraram desempenho comparável ou ligeiramente superior em relação ao TF-IDF. Nos dados de validação o BOW, sem balanceamento de classes, geralmente apresenta os maiores percentuais nas métricas.

O uso do *Word embeddings*, especialista em análise contextual, na maioria dos casos não superou os métodos tradicionais de vetorização nos conjuntos de validação, uma vez que os textos apresentados aos modelos se trata de um vocabulário específico, com frases curtas, abreviações de palavras e contextualmente limitadas, como o jargão bancário [28].

O uso de SMOTE, Smote_tomek e pesos para lidar com desequilíbrio de classes melhorou o recall e o F1-*score* em muitos casos no conjunto de validação, especialmente para modelos como SVM e Random Forest.

O SVM e o Gradient Boosting exigem mais tempo de treinamento, especialmente com conjuntos de dados balanceados. *Logistic Regression*, *Decision Tree*, e KNN são geralmente mais rápidos.

Após os 70 ciclos de execução, foram selecionados os 12 modelos referentes às combinações que apresentaram os maiores percentuais de F1-*score* nos resultados da validação. A partir dessas 12 combinações foram gerados mais 12 modelos com ajuste de hiperparâmetros (RandomizedSearchCV). Esses 24 modelos foram submetidos ao conjunto de dados de **Teste** (dados nunca vistos), conforme exposto na figura 22.

O melhor resultado obtido foi com o **modelo SVM**, no **ciclo Nº 17**, **com ajuste de hiperparâmetros**. O modelo selecionado apresentou o maior F1 no conjunto de teste (88,54%).

A curva de aprendizado (figura 23) ilustra como a acurácia do modelo selecionado varia em relação ao número de exemplos de treinamento (642.544 registros). Inicialmente, observa-se que a acurácia nos dados de treinamento é significativamente mais alta do que nos dados de validação, sugerindo a possibilidade de *overfitting*. Contudo, conforme aumenta o número de exemplos de treinamento, a acurácia nos dados de validação melhora substancialmente e começa a convergir com a acurácia de treinamento. Isso indica que o modelo está generalizando melhor e aprendendo de maneira mais eficaz à medida que é exposto a mais dados.

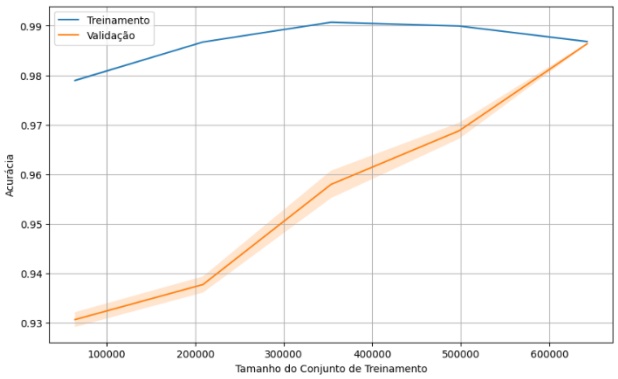

Fig. 23: Gráfico de Curva de Treinamento vs. Validação do modelo SVM (Nº 17). Fonte: Autores.

4.2. BERT

Conforme estabelecido na metodologia, a figura 24 apresenta os resultados dos 04 ciclos de ajuste/validação e teste utilizando os modelos pré-treinados.

Os resultados foram muito próximos. Foi selecionada a combinação apresentada no ciclo 01 que obteve o melhor F1-*score* no conjunto de dados de **Teste** (dados nunca vistos).

A curva de perda mostrada na figura 25 demonstra que o modelo rapidamente se ajusta aos dados e a partir da terceira época a perda no conjunto de treinamento tende a se estabilizar, reforçando a indicação dos autores quanto ao número de épocas recomendadas para ajuste do modelo.

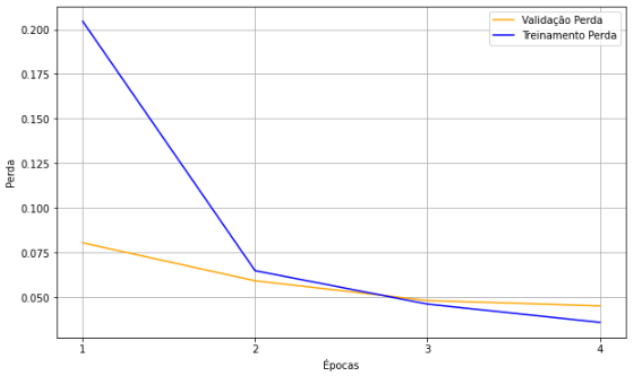

Fig. 25: Gráfico de Perda de Treinamento vs. Validação do modelo BERT. Fonte: Autores.

Nº	Pré-process	Vetorização/W Embeddings	Balance	Modelos	Acuracy CV (%)	Precision CV (%)	Recall CV (%)	F1 CV (%)	Tempo (s) CV	Acuracy Valid (%)	Precision Valid (%)	Recall Valid (%)	F1 Valid (%)	Tempo (s) Treino Total (80%)
1	Limp+Stop	BOW	N/A	R Logistic	97,85	96,13	93,54	94,77	00:00:42	97,95	96,49	94,01	95,17	00:00:27
2	Limp+Stop	BOW	N/A	SVM	98,08	96,46	93,17	94,65	00:46:41	98,24	96,69	93,79	95,08	00:17:03
3	Limp+Stop	BOW	N/A	R Forest	97,81	95,70	93,08	94,32	00:28:21	97,78	94,80	93,68	94,18	00:13:09
4	Limp+Stop	BOW	N/A	D Tree	97,40	95,14	90,78	92,71	00:13:39	97,63	94,09	93,69	93,84	00:00:27
5	Limp+Stop	BOW	N/A	KNN	97,65	93,43	93,29	93,34	00:04:26	97,67	95,76	91,35	93,24	00:00:52
6	Limp+Stop	BOW	Pesos	R Forest	97,81	95,70	93,81	94,71	00:19:18	97,77	94,88	94,08	94,44	00:11:38
7	Limp+Stop	BOW	Pesos	SVM	97,21	91,59	96,96	93,88	01:01:04	97,36	91,17	96,71	93,54	00:21:10
8	Limp+Stop	BOW	Pesos	KNN	97,20	94,10	91,30	92,57	00:09:18	97,36	93,98	91,78	92,74	00:01:01
9	Limp+Stop	BOW	Pesos	R Logistic	96,63	88,67	96,71	92,06	00:00:35	96,68	89,00	96,43	92,11	00:00:28
10	Limp+Stop	BOW	Pesos	D Tree	97,42	93,07	93,48	93,26	00:01:23	97,57	92,53	92,05	92,07	00:00:30
11	Limp+Stop	TF-IDF	N/A	R Forest	97,87	95,80	93,36	94,50	00:06:42	97,83	94,86	93,76	94,26	00:03:03
12	Limp+Stop	TF-IDF	N/A	SVM	97,72	96,02	91,19	93,33	00:51:09	98,04	96,33	92,27	94,03	00:19:22
13	Limp+Stop	TF-IDF	N/A	D Tree	97,58	93,37	92,79	93,06	00:00:12	97,65	93,79	93,82	93,74	00:00:05
14	Limp+Stop	TF-IDF	N/A	R Logistic	97,57	94,95	91,88	93,31	00:00:22	97,68	95,36	92,12	93,54	00:00:09
15	Limp+Stop	TF-IDF	N/A	KNN	87,85	81,61	59,82	66,02	00:06:04	88,38	83,52	60,94	66,40	00:00:01
16	Limp+Stop	TF-IDF	Smote	R Forest	97,85	95,75	93,80	94,77	01:05:19	97,84	94,91	94,20	94,52	00:23:51
17	Limp+Stop	TF-IDF	Smote	SVM	97,69	92,95	95,02	93,97	02:51:41	98,00	93,26	96,10	94,51	00:52:33
18	Limp+Stop	TF-IDF	Smote	D Tree	97,32	93,06	92,72	92,89	00:02:08	97,39	93,48	93,76	93,59	00:00:58
19	Limp+Stop	TF-IDF	Smote	R Logistic	96,45	88,16	96,10	91,96	00:02:10	96,55	88,58	96,33	91,89	00:00:44
20	Limp+Stop	TF-IDF	Smote	KNN	83,99	63,53	80,27	70,93	00:02:28	84,53	65,44	81,40	71,31	00:00:31
21	Limp+Stop	TF-IDF	Pesos	R Forest	97,81	95,42	93,98	94,68	00:07:29	97,77	95,02	94,10	94,51	00:05:24
22	Limp+Stop	TF-IDF	Pesos	D Tree	97,39	93,11	93,58	93,33	00:00:23	97,61	94,36	92,70	93,44	00:00:10
23	Limp+Stop	TF-IDF	Pesos	SVM	96,73	90,35	96,13	92,82	00:27:10	97,03	90,29	96,32	92,88	00:19:22
24	Limp+Stop	TF-IDF	Pesos	R Logistic	96,04	86,92	96,52	90,90	00:00:35	96,13	86,93	96,32	90,83	00:00:14
25	Limp+Stop	TF-IDF	Pesos	KNN	88,51	79,04	63,01	68,17	00:06:32	88,93	79,23	64,87	69,10	00:00:52
26	Limp+Stop+Lema	BOW	N/A	R Logistic	97,88	96,14	93,79	94,90	00:00:39	98,02	96,26	94,44	95,28	00:00:29
27	Limp+Stop+Lema	BOW	N/A	R Forest	97,82	95,81	93,30	94,49	00:43:01	97,79	95,07	93,31	94,06	00:12:58
28	Limp+Stop+Lema	BOW	N/A	SVM	98,07	96,39	93,23	94,64	00:30:04	98,22	96,71	94,02	95,22	00:16:17
29	Limp+Stop+Lema	BOW	N/A	D Tree	97,60	93,42	93,17	93,27	00:03:25	97,58	94,50	93,75	94,04	00:00:28
30	Limp+Stop+Lema	BOW	N/A	KNN	97,39	95,14	90,97	92,84	00:08:48	97,59	94,96	91,79	93,16	00:00:53
31	Limp+Stop+Stem	BOW	N/A	SVM	98,09	96,31	93,41	94,73	00:31:23	98,25	96,69	94,01	95,20	00:14:29
32	Limp+Stop+Stem	BOW	N/A	R Logistic	97,80	96,01	93,37	94,62	00:02:37	97,88	96,48	94,03	95,16	00:00:25
33	Limp+Stop+Stem	BOW	N/A	D Tree	97,64	93,76	93,44	93,59	00:02:23	97,62	94,25	93,56	93,84	00:00:25
34	Limp+Stop+Stem	BOW	N/A	R Forest	97,81	95,65	93,24	94,38	00:22:41	97,80	94,86	93,51	94,08	00:11:49
35	Limp+Stop+Stem	BOW	N/A	KNN	97,34	95,06	90,49	92,52	00:09:12	97,64	95,61	91,70	93,42	00:00:51
36	Limp+Stop+Stem	BOW	Smote	SVM	97,96	93,04	95,74	94,37	01:02:45	98,12	93,42	96,34	94,72	00:34:59
37	Limp+Stop+Stem	BOW	Smote	R Forest	97,82	95,68	93,84	94,75	00:45:21	97,81	94,89	94,11	94,46	00:33:52
38	Limp+Stop+Stem	BOW	Smote	D Tree	97,56	93,48	93,62	93,55	00:00:36	97,54	93,97	93,74	93,78	00:00:29
39	Limp+Stop+Stem	BOW	Smote	R Logistic	96,82	90,05	95,89	92,88	00:01:13	96,90	90,47	96,55	93,06	00:00:26
40	Limp+Stop+Stem	BOW	Smote	KNN	96,41	88,13	93,39	90,68	00:05:00	96,71	88,68	94,60	91,15	00:01:12
41	Limp+Stop+Stem	BOW	Pesos	XGBoost	98,19	95,63	94,35	94,99	00:02:53	98,32	95,66	95,18	95,40	00:00:45
42	Limp+Stop+Stem	TF-IDF	Pesos	XGBoost	98,19	95,63	94,34	94,98	00:02:37	98,32	95,66	94,83	95,22	00:01:03
43	Limp+Stop	TF-IDF	Smote_tomek	R Forest	98,02	95,88	94,04	94,95	00:55:02	97,73	94,27	93,76	93,97	00:19:26
44	Limp+Stop	TF-IDF	Smote_tomek	SVM	97,81	92,89	95,59	94,22	01:22:45	97,81	92,34	95,29	93,64	00:42:39
45	Limp+Stop	TF-IDF	Smote_tomek	D Tree	96,89	92,89	92,89	92,89	00:01:59	97,20	92,28	93,16	92,65	00:00:56
46	Limp+Stop	TF-IDF	Smote_tomek	R Logistic	96,02	87,40	95,10	91,09	00:01:32	96,29	86,42	95,41	90,22	00:00:42
47	Limp+Stop	TF-IDF	Smote_tomek	KNN	84,12	62,02	79,87	68,20	00:30:02	84,39	62,48	80,24	68,76	00:02:01
48	Limp+Stop+Stem	BOW	Pesos	Grad Boost	96,02	93,98	88,12	90,96	03:45:04	96,16	94,10	88,23	90,05	01:05:01
49	Limp+Stop+Stem	BOW	Pesos	AdaBoost	55,4	38,98	29,74	33,74	00:04:54	55,65	39,35	29,81	29,78	00:01:35
50	Limp+Stop+Stem	BOW	Pesos	CatBoost	97,51	95,21	93,55	94,37	00:35:41	97,65	95,37	93,72	94,47	00:11:19
51	Limp+Stop	Word2 CBOW	Pesos	R Forest	96,05	93,83	86,06	89,77	00:12:41	96,20	94,13	86,61	89,85	00:04:31
52	Limp+Stop	Word2 CBOW	Pesos	SVM	95,57	93,27	83,76	88,26	00:51:02	95,72	93,57	84,31	88,12	00:23:27
53	Limp+Stop	Word2 CBOW	Pesos	R Logistic	95,05	91,89	84,10	87,83	00:03:21	95,19	92,19	84,65	87,80	00:00:45
54	Limp+Stop	Word2 CBOW	Pesos	D Tree	94,68	90,02	85,37	87,63	00:04:01	94,82	90,32	85,92	87,75	00:01:59
55	Limp+Stop	Word2 CBOW	Pesos	KNN	95,14	90,06	84,32	87,10	00:00:04	95,28	90,46	84,87	87,14	00:00:01
56	Limp+Stop	Word2 Gram	Pesos	SVM	93,24	81,99	91,45	86,46	00:00:08	93,38	82,39	92,00	84,10	00:00:02
57	Limp+Stop	Word2 Gram	Pesos	R Forest	91,27	73,63	91,98	81,79	00:00:04	91,41	74,03	92,43	78,30	00:00:01
58	Limp+Stop	Word2 Gram	Pesos	D Tree	95,80	91,76	86,29	88,94	00:03:25	95,94	92,16	86,74	88,99	00:01:46
59	Limp+Stop	Word2 Gram	Pesos	R Logistic	91,69	78,74	91,30	84,55	00:00:04	91,85	79,14	91,75	82,27	00:00:01
60	Limp+Stop	Word2 Gram	Pesos	KNN	92,78	79,58	92,51	85,56	00:50:45	92,94	79,88	92,96	83,16	00:15:37
61	Limp+Stop	Fast CBOW	Pesos	R Forest	96,01	93,99	86,32	89,99	00:15:09	96,17	94,29	86,87	90,09	00:04:07
62	Limp+Stop	Fast CBOW	Pesos	D Tree	94,81	89,64	85,55	87,55	00:04:03	94,97	89,94	86,10	87,72	00:01:41
63	Limp+Stop	Fast CBOW	Pesos	KNN	94,92	91,19	83,73	87,30	00:00:04	95,08	91,49	84,28	87,29	00:00:01
64	Limp+Stop	Fast CBOW	Pesos	SVM	95,14	92,73	80,84	86,37	02:52:13	95,29	92,98	81,22	84,89	00:39:34
65	Limp+Stop	Fast CBOW	Pesos	R Logistic	94,80	88,20	81,65	84,80	00:03:25	94,95	88,45	82,03	84,80	00:00:42
66	Limp+Stop	Glove	Pesos	R Forest	96,15	94,23	86,53	90,21	00:10:41	96,30	94,48	86,91	90,17	00:04:16
67	Limp+Stop	Glove	Pesos	SVM	95,92	93,81	85,45	89,44	00:58:35	96,05	94,06	86,00	89,36	00:19:02
68	Limp+Stop	Glove	Pesos	R Logistic	95,30	92,23	85,37	88,67	00:03:12	95,43	92,48	85,92	88,74	00:00:46
69	Limp+Stop	Glove	Pesos	KNN	94,82	90,05	85,03	87,47	00:00:04	94,95	90,30	85,58	87,61	00:00:01
70	Limp+Stop	Glove	Pesos	D Tree	95,26	90,80	84,59	87,59	00:07:22	95,39	91,05	85,14	87,56	00:02:05

Fig. 21: Quadro de resultados de 70 ciclos de execução. Fonte: Autores.

Nº	Ajuste Hiper	Pré-process	Vetorização/ W Embeddings	Balance	Modelos	Acuracy Valid (%)	Precision Valid (%)	Recall Valid (%)	F1 Valid (%)	Tempo (s) Treino Total (80%)	Acuracy Teste (%)	Precision Teste (%)	Recall Teste (%)	F1 Teste (%)
1	N	Limp+Stop	BOW	N/A	R Logistic	97,95	96,49	94,01	95,17	00:00:27	93,62	89,95	87,16	88,18
	S	Limp+Stop	BOW	N/A	R Logistic	98,16	96,75	94,75	95,70	01:01:45	93,63	87,91	87,99	87,09
2	N	Limp+Stop	BOW	N/A	SVM	98,24	96,69	93,79	95,08	00:17:03	93,72	89,86	85,16	86,62
	S	Limp+Stop	BOW	N/A	SVM	98,47	97,05	95,38	96,17	02:26:34	94,01	89,91	87,64	88,37
16	N	Limp+Stop	TF-IDF	Smote	R Forest	97,84	94,91	94,20	94,52	00:23:51	93,69	89,47	87,61	88,12
	S	Limp+Stop	TF-IDF	Smote	R Forest	97,93	92,71	96,35	94,34	14:42:33	93,63	86,54	87,96	86,75
17	N	Limp+Stop	TF-IDF	Smote	SVM	98,00	93,26	96,10	94,51	00:52:33	93,47	89,77	87,60	88,33
	S	Limp+Stop	TF-IDF	Smote	SVM	98,37	97,06	94,41	95,60	17:44:29	93,63	90,34	87,47	88,54
21	N	Limp+Stop	TF-IDF	Pesos	R Forest	97,77	95,02	94,10	94,51	00:05:24	93,69	89,50	87,49	88,02
	S	Limp+Stop	TF-IDF	Pesos	R Forest	97,96	92,64	96,16	94,22	00:47:10	93,64	87,50	88,06	87,27
26	N	Limp+Stop+Lema	BOW	N/A	R Logistic	98,02	96,26	94,44	95,28	00:00:29	93,58	89,59	86,06	87,32
	S	Limp+Stop+Lema	BOW	N/A	R Logistic	98,12	96,70	94,98	95,79	00:56:39	93,75	89,26	86,75	87,51
28	N	Limp+Stop+Lema	BOW	N/A	SVM	98,22	96,71	94,02	95,22	00:16:17	93,72	89,36	85,01	86,28
	S	Limp+Stop+Lema	BOW	N/A	SVM	98,47	96,59	95,41	96,14	01:40:57	93,98	89,40	86,92	87,76
31	N	Limp+Stop+Stem	BOW	N/A	SVM	98,25	96,69	94,01	95,20	00:14:29	93,79	89,91	85,20	86,64
	S	Limp+Stop+Stem	BOW	N/A	SVM	98,47	96,95	95,23	96,03	05:53:27	96,69	89,78	87,83	88,43
32	N	Limp+Stop+Stem	BOW	N/A	R Logistic	97,88	96,48	94,03	95,16	00:00:25	93,39	89,36	86,70	87,56
	S	Limp+Stop+Stem	BOW	N/A	R Logistic	98,06	96,8	94,58	95,62	00:14:23	93,51	88,63	87,79	87,83
36	N	Limp+Stop+Stem	BOW	Smote	SVM	98,12	93,42	96,34	94,72	00:34:59	94,01	89,62	88,16	88,50
	S	Limp+Stop+Stem	BOW	Smote	SVM	98,13	94,72	95,07	94,78	12:16:44	93,60	89,29	87,39	87,93
41	N	Limp+Stop+Stem	BOW	Pesos	XGBoost	98,32	95,66	95,18	95,40	00:00:45	86,81	66,83	65,99	66,00
	S	Limp+Stop+Stem	BOW	Pesos	XGBoost	97,54	94,56	93,59	94,20	00:05:04	85,44	65,41	64,52	65,10
42	N	Limp+Stop+Stem	TF-IDF	Pesos	XGBoost	98,32	95,66	94,83	95,22	00:01:03	86,70	66,63	65,84	65,79
	S	Limp+Stop+Stem	TF-IDF	Pesos	XGBoost	98,10	95,10	94,72	94,90	00:06:54	86,40	66,10	65,32	65,95

Fig. 22: Quadro de resultados de 24 ciclos de testes. Fonte: Autores.

Nº	Modelo Pré-treinado	Tamanho máximo da sentença	Tamanho do Batch	Atribuição de pesos ('alpha')	nº Épocas	Otimizador	Taxa de aprendizado	weight_decay	Acuracia na validação	Precisão na validação	Recall na validação	F1 na validação	tempo total	Precisão no teste	Recall no teste	F1 no teste
1	bert-base-uncased	64	32	1.0	4	AdamW	2e-5	0,1	98,39	96,95	95,05	96,00	29:01:44	89,20	85,45	87,33
2	bert-base-uncased	64	32	0.5	4	AdamW	2e-5	0,1	98,35	96,45	95,55	96,00	29:40:14	86,75	86,60	86,68
3	bert-base-portuguese-cased	64	32	1.0	4	AdamW	2e-5	0,1	98,37	96,50	95,50	96,00	29:15:00	87,45	87,10	87,28
4	bert-base-portuguese-cased	64	32	0.5	4	AdamW	2e-5	0,1	98,35	96,65	95,40	96,03	29:05:16	88,75	85,25	87,00

Fig. 24: Quadro de resultados dos 04 ciclos de ajuste/validação e teste dos modelos BERT. Fonte: Autores.

Da mesma forma, na figura 26, pode-se constatar que a acurácia geral do modelo tende a se estabilizar, num excelente patamar (98%), a partir da terceira época, indicando o ponto de parada do treinamento, num momento ótimo, antes da possibilidade de ocorrência de sobreajuste (*overfitting*).

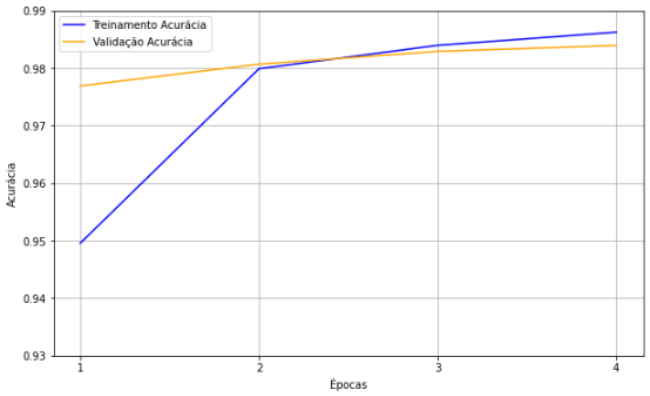

Fig. 26: Gráfico de Acurácia de Treinamento vs. Validação do modelo BERT. Fonte: Autores.

4.3. Comparação de Resultados

A comparação é iniciada com a apresentação dos **Relatórios de Classificação** e **Matriz de Confusão** relativos ao desempenho dos modelos escolhidos no conjunto de **Teste**. As figuras 27 e 28 apresentam os resultados do algoritmo de classificação **SVM** e as figuras 29 e 30 apresentam os resultados do modelo pré-treinado **BERT**.

A partir da análise dos referidos resultados, destacam-se os pontos listados a seguir.

a) O SVM atingiu 87% de *recall* para a classe "220-Depósito em Espécie" deixando de classificar lançamentos dessa classe (e classificando-os como "201-Depósito em Cheque"). O BERT atingiu 84%, falhando da mesma forma. Tais erros podem ser justificados ao observar-se que a categoria "220-Depósito em Espécie" foi adicionada recentemente ao layout 'Padrão FEBRABAN 240 Posições' [54]. Ao se consultar versões anteriores do mesmo documento [55], constata-se que antes existia apenas a categoria "201-Depósitos", englobando as duas formas de depósito em conta (cheque e espécie). Dessa forma, como a maioria dos exemplos de treinamento foram da classe "201", a classe "220" introduzida recentemente não é muito "lembrada".

b) O SVM atingiu 71% de *precision* para a classe "202-Crédito de Cobrança", utilizando erroneamente essa classe para classificar lançamentos da classe "217-Pagamentos Fornecedores". O BERT atingiu 59%, falhando ao utilizar a classe "202" para classificar as classes "217" e "222-Recebíveis de cartão de crédito". As classes "202" e "217" são tecnicamente semelhantes em sua natureza e as próprias instituições financeiras, eventualmente, as tratam de forma indistinta. A figura 31 ilustra como, para dada instituição financeira, o mesmo conteúdo de 'HISTORICO' pode atender indistintamente as classes "202" e "217".

BCO DO TITULAR	HISTORICO	TIPO PESSOA TITULAR	CD LANC	QTDE DE REGISTROS
237	'RECEBIMENTO FORNECEDOR'	0	202-Crédito de Cobrança	345
			217-Pagamentos Fornecedores	76
		1	202-Crédito de Cobrança	389
			217-Pagamentos Fornecedores	67

Fig. 31: Registros recebidos do BANCO BRADESCO (237) com o histórico "RECEBIMENTO FORNECEDOR". Fonte: Autores.

c) Semelhante ao ocorrido com a classe "220", recentemente foi criada pela FEBRABAN a classe "222-Recebíveis de cartão de crédito". O SVM reconheceu 82% dos registros dessa classe, enquanto o BERT reconheceu apenas 41%.

d) Por se tratar de categoria com razoável número de ocorrências (3481), merece especial atenção a classe "213-Transferência entre Contas" que registrou índices de recall reduzidos (73%) para ambos os modelos. Muitos registros da categoria "213" foram classificados como "209-Transferência Interbancária (DOC, TED, Pix)". A análise detalhada desses registros aponta que sua quase totalidade contém a literal "PIX TRANSF" no atributo 'HISTORICO'. Fica evidente que, dada a novidade do PIX, as próprias instituições financeiras estão classificando registros que seriam "209" como sendo "213".

e) O *recall* da classe "210-Ações" foi de 73% para o SVM e 70% para o BERT. Parte dos registros dessa classe foram classificados como "206-Resgate de Aplicação". O baixo número de exemplos de treinamento (35) e a similitude das operações (valores oriundos de investimentos do correntista) poderiam explicar a confusão.

f) Finalmente, é importante registrar que em sua última atualização, o leiaute FEBRABAN passou a contemplar uma nova categoria: "223-Crédito Pix via QrCode". No conjunto de treino/validação não foram encontrados registros dessa categoria. Já no conjunto de testes, verificou-se a ocorrência de 05 registros "223". Os 05 registros "223-Crédito Pix via QrCode" foram classificados como "209-Transferência Interbancária (DOC, TED, Pix)", o que parece razoável. Em etapas futuras, a nova classe "223" deverá fazer parte da etapa de treinamento.

Encerrando a comparação de métricas, foi realizada a comparação dos Gráficos da **Curva ROC** relativos ao desempenho dos modelos escolhidos no conjunto de **Teste** (figuras 32 e 33).

	precision	recall	f1-score	support
201-Depósito em Cheque	0.98	0.98	0.98	2905
202-Crédito de Cobrança	0.71	0.96	0.82	107
203-Devolução de Cheques	0.99	0.99	0.99	183
204-Estornos	0.96	0.96	0.96	963
206-Resgate de Aplicação	0.96	0.97	0.97	1156
207-Empréstimo /Financiamento	0.90	0.97	0.93	175
208-Câmbio	1.00	0.89	0.94	9
209-Transferência Interbancária (DOC, TED, Pix)	0.91	0.99	0.95	11330
210-Ações	0.96	0.73	0.83	33
211-Dividendos	0.94	1.00	0.97	78
212-Seguro	1.00	0.97	0.98	30
213-Transferência entre Contas	0.96	0.73	0.83	3841
214-Depósitos Especiais	1.00	0.95	0.97	92
215-Devolução da Compensação	1.00	0.99	0.99	488
217-Pagamentos Fornecedores	1.00	0.93	0.96	854
218-Pagamentos Diversos	0.81	0.83	0.82	312
219-Recebimento de Salário	0.99	0.97	0.98	197
220-Depósito em Espécie	0.99	0.87	0.93	429
222-Recebíveis de cartão de crédito	1.00	0.82	0.90	49
223-Crédito Pix via QrCode	0.00	0.00	0.00	5
accuracy			0.94	23236
macro avg	0.90	0.87	0.89	23236
weighted avg	0.94	0.94	0.93	23236

Fig. 27: Relatório de classificação no conjunto de **Teste** do **algoritmo de classificação** SVM. Fonte: Autores.

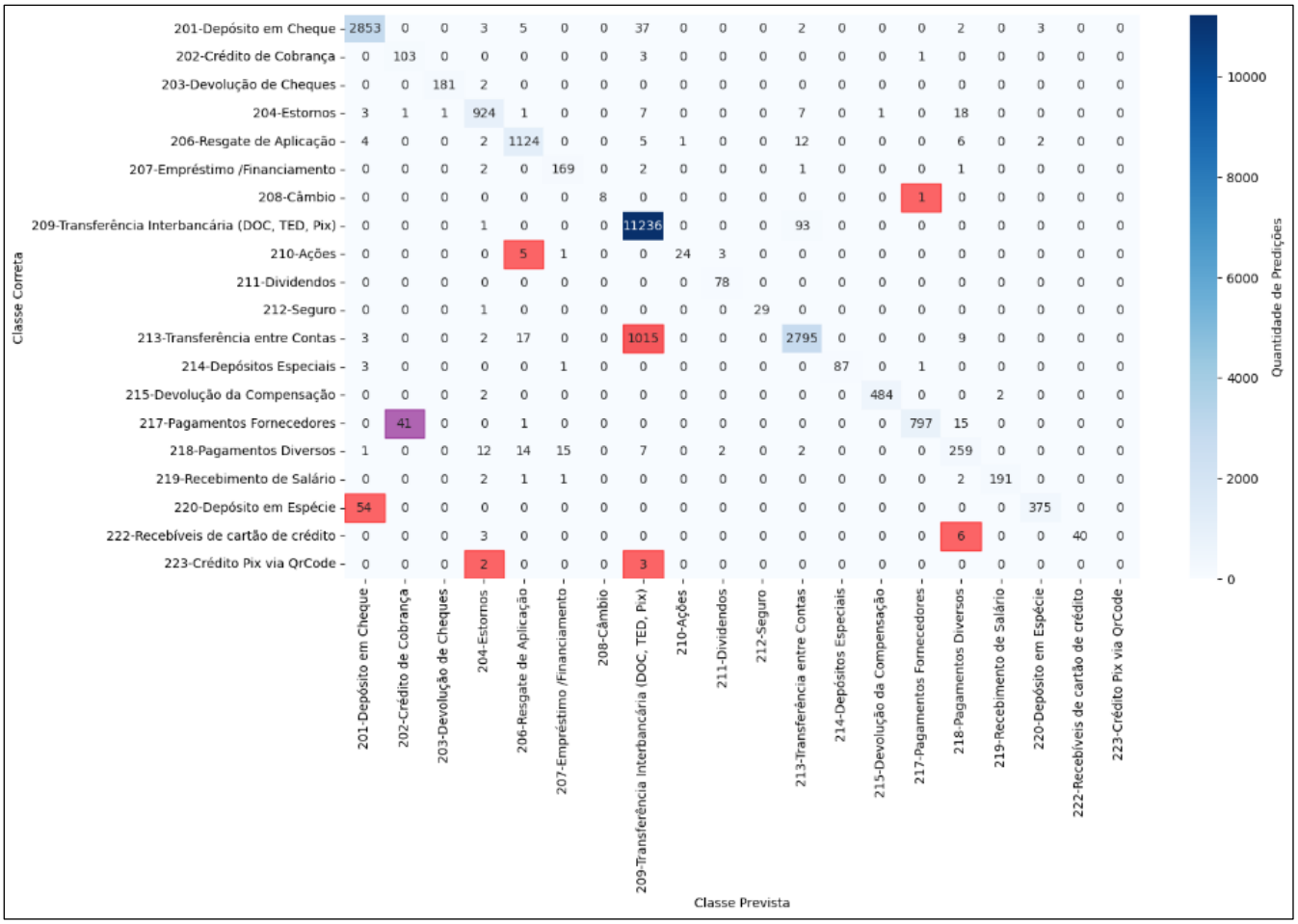

Fig. 28: Matriz de confusão no conjunto de **Teste** do **algoritmo de classificação** SVM. Fonte: Autores.

	precision	recall	f1-score	support
201-Depósito em Cheque	0.97	0.98	0.98	2905
202-Crédito de Cobrança	0.59	1.00	0.74	107
203-Devolução de Cheques	1.00	0.95	0.97	183
204-Estornos	0.99	0.96	0.97	963
206-Resgate de Aplicação	0.96	0.98	0.97	1156
207-Empréstimo / Financiamento	0.88	0.95	0.91	175
208-Câmbio	1.00	1.00	1.00	9
209-Transferência Interbancária (DOC, TED, Pix)	0.92	0.99	0.95	11330
210-Ações	0.96	0.70	0.81	33
211-Dividendos	1.00	0.97	0.99	78
212-Seguro	1.00	1.00	1.00	30
213-Transferência entre Contas	0.92	0.73	0.82	3841
214-Depósitos Especiais	0.93	0.97	0.95	92
215-Devolução da Compensação	1.00	1.00	1.00	488
217-Pagamentos Fornecedores	0.99	0.90	0.94	854
218-Pagamentos Diversos	0.78	0.79	0.78	312
219-Recebimento de Salário	0.98	0.97	0.97	197
220-Depósito em Espécie	0.97	0.84	0.90	429
222-Recebíveis de cartão de crédito	1.00	0.41	0.58	49
223-Crédito Pix via QrCode	0.00	0.00	0.00	5
accuracy			0.93	23236
macro avg	0.89	0.85	0.86	23236
weighted avg	0.93	0.93	0.93	23236

Fig. 29: Relatório de classificação no conjunto de **Teste** do **modelo pré-treinado** BERT. Fonte: Autores.

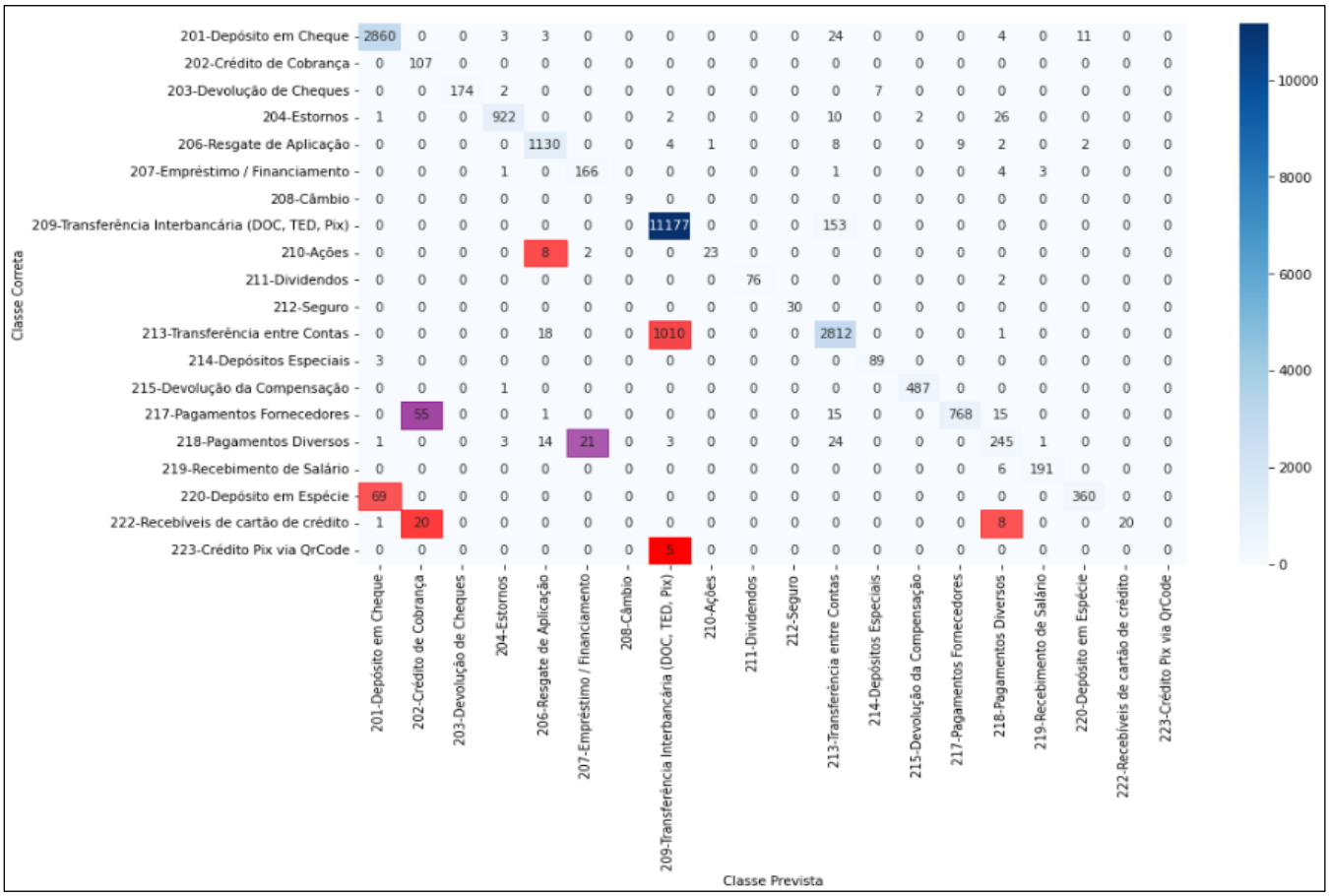

Fig. 30: Matriz de Confusão no conjunto de **Teste** do **modelo pré-treinado** BERT. Fonte: Autores.

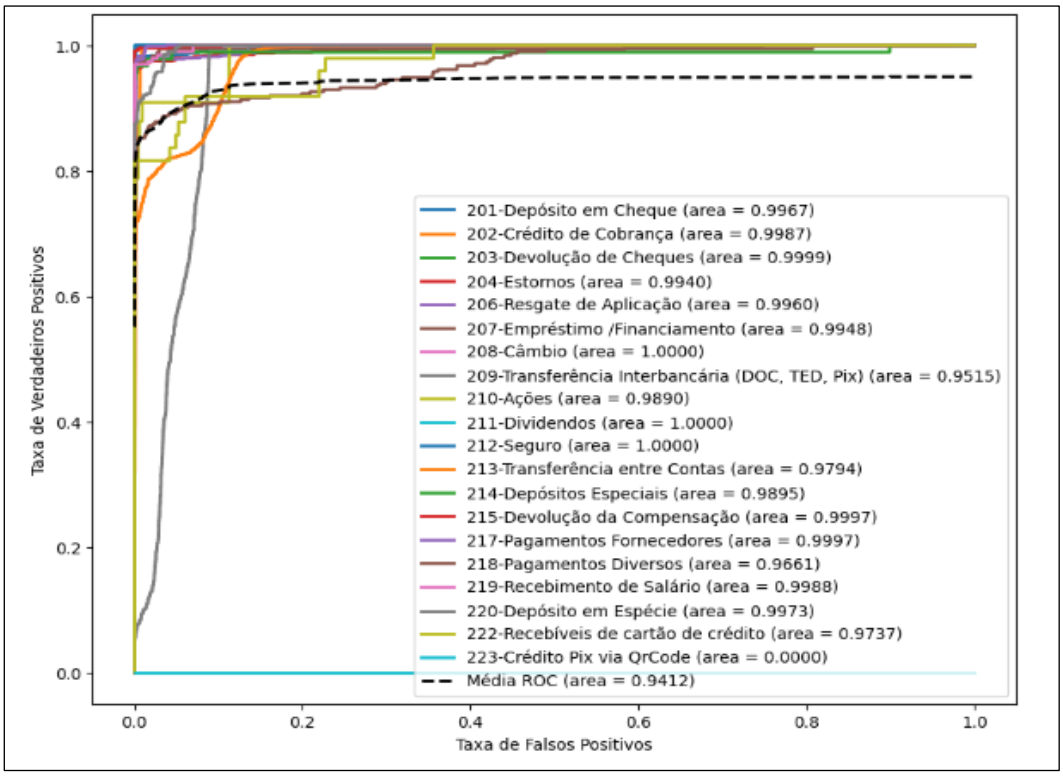

Fig. 32: Gráfico da Curva ROC no conjunto de **Teste** do **algoritmo de classificação** SVM. Fonte: Autores.

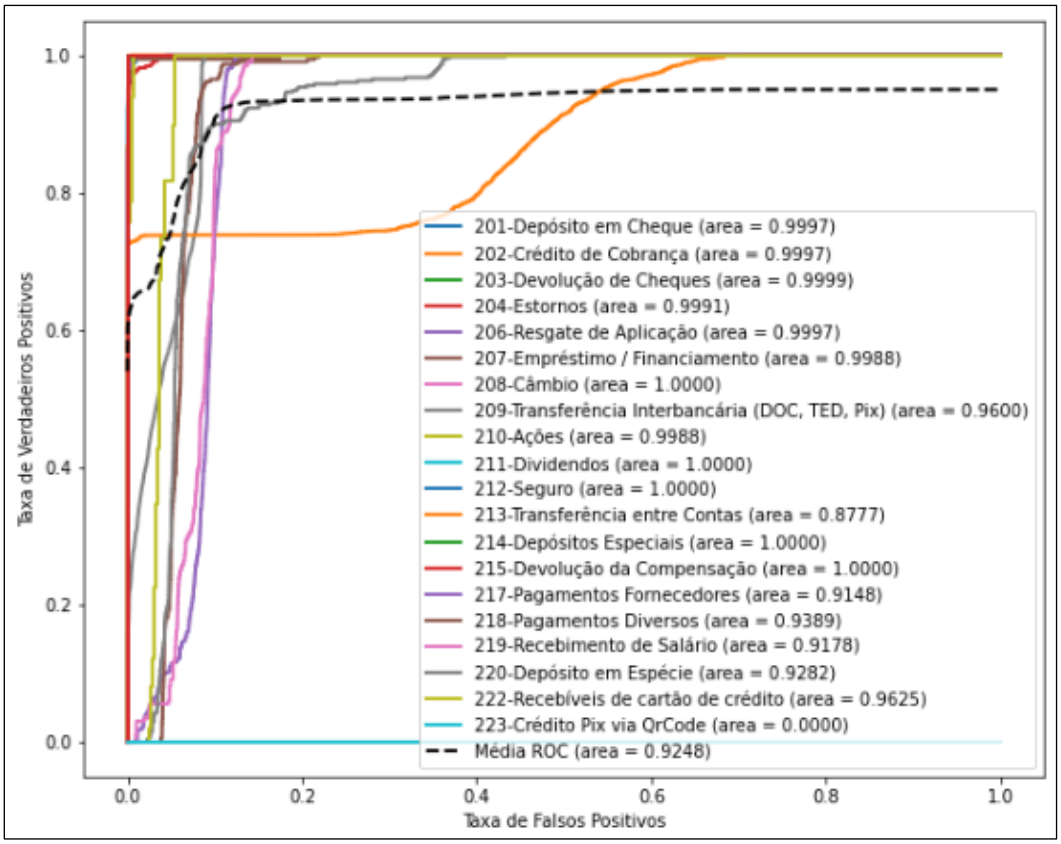

Fig. 33: Gráfico da Curva ROC no conjunto de **Teste** do **modelo pré-treinado** BERT. Fonte: Autores.

As curvas ROC referentes ao conjunto de **Teste** mostram uma excelente separação das classes, com Áreas Sob a Curva (AUC) próximas ou iguais a 1.0. As áreas menores refletem as discrepâncias previamente explicadas, principalmente devido à forma como a novidade das transações PIX vem sendo tratada pelas instituições financeiras. Os modelos apresentam alta taxa de verdadeiros positivos e baixa taxa de falsos positivos, ou seja, quase todas as instâncias de dada classe são corretamente identificadas como pertencentes a essa classe (alta sensibilidade) e poucas instâncias das outras classes são incorretamente identificadas como pertencentes à classe em questão (alta especificidade).

4.4. Discussão

Embora com resultados muito parecidos, o desempenho do modelo **SVM** selecionado foi 0,5% **superior** ao modelo **BERT** ajustado, no conjunto de dados de **Teste**. Traduzido em números, dentre os 23.236 registros do referido conjunto, o modelo **SVM** errou 1479 (6,3%) classificações e o modelo **BERT** ajustado errou 1590 (6,8%) classificações, um saldo de 111 acertos em favor do modelo **SVM**.

Outro ponto que pesa a favor do modelo **SVM** é o tempo de treinamento. O modelo **SVM** levou **17:44:29** para ser treinado/ajustado, enquanto o modelo **BERT** levou **29:01:44** para ser ajustado. Em relação ao tempo de treinamento, acrescenta-se que modelos mais simples, como a Regressão Logística, atingiram índices muito próximos em todas as métricas e, eventualmente, foram treinados em segundos.

Após a análise dos resultados, fica evidente que o modelo está bem ajustado e oferece uma alta capacidade de generalização, o que é essencial para sua aplicação prática. Contudo, alguns desafios específicos, como a confusão entre categorias similares e a adaptação a novas categorias, ainda persistem. Isso sugere a necessidade de ajustes contínuos no modelo, especialmente à medida que novos dados e categorias são incorporados. Como exemplo, a incorporação da categoria "223-Crédito Pix via QrCode" no treinamento e ajustes futuros poderão melhorar ainda mais a precisão e robustez do modelo.

5. Conclusão

O objetivo deste estudo foi comparar o desempenho de diversos modelos de aprendizado de máquina na classificação de lançamentos bancários. O resultado apontou o desempenho do modelo SVM (88,54% no F1-*score* com o conjunto de dados de teste) como superior aos demais na execução da tarefa proposta.

Além da comparação do desempenho de algoritmos de aprendizado de máquina em dados reais, este estudo oferece várias contribuições teóricas importantes. Primeiramente, o impacto das técnicas de pré-processamento no desempenho dos modelos, evidenciando a importância de escolher métodos adequados. Também pode ser destacada a observação de como a utilização de técnicas de balanceamento de dados influenciam os resultados em cenários de classes desbalanceadas.

Em termos de contribuição prática, o estudo apresenta melhorias significativas nos processos de investigação do Ministério Público, propondo um modelo escalável que pode ser adotado por outras instituições públicas. A classificação eficaz dos registros bancários permite uma compreensão completa das transações financeiras pelas partes envolvidas. Ao categorizar e analisar os dados, é possível identificar padrões e tendências nas movimentações de recursos, fornecendo uma visão geral das atividades financeiras dos indivíduos e entidades sob investigação. Essa compreensão ampla é essencial para entender o contexto em que os ilícitos financeiros ocorrem e identificar possíveis pontos de vulnerabilidade no sistema financeiro.

Os resultados dessas iniciativas poderão ser aplicados imediatamente na produção dos relatórios de análise técnica, proporcionando um impacto significativo na eficiência e eficácia dos analistas da CSI. A segurança e precisão na classificação dos registros resultarão em um ganho de tempo considerável na elaboração dos relatórios, que atualmente dependem de uma revisão manual extensiva e demorada das classificações, realizada em Excel. Essa transformação não apenas acelera o processo de produção de relatórios, mas também aprimora a capacidade do MPBA de conduzir investigações de forma mais rápida e precisa, fortalecendo assim a sua missão institucional.

Os resultados promissores deste estudo apontam para uma base sólida que justificaria futuras expansões e investimentos por parte do MPBA, especialmente no aprimoramento do parque tecnológico e humano da CSI. A eficácia demonstrada pelos modelos na classificação de movimentações financeiras justifica um investimento em recursos computacionais mais robustos e a criação de uma unidade específica para desenvolvimento de projetos na área de ciência de dados. Tais avanços permitirão a elaboração de projetos mais abrangentes na análise de movimentações financeiras, incluindo a automação de parte da elaboração dos relatórios de análise técnica. Este progresso poderia, eventualmente, levar ao desenvolvimento de ferramentas avançadas para a análise de padrões de comportamento financeiro dos investigados, utilizando a temporalidade das movimentações.

É fundamental considerar certas limitações que podem ter influenciado a interpretação dos resultados. A impossibilidade de utilização de estruturas de plataformas de computação em nuvem (como o Google Colab PRO e a Amazon Web Services-AWS) devido à obrigatoriedade de proteção dos dados sigilosos implicou na necessidade de redução do volume de dados tratados dada a limitada capacidade computacional disponível no MPBA. Também, a necessidade de anonimização para garantir a conformidade legal e o descarte de dados com rótulos genéricos, embora essenciais para este trabalho, podem ter reduzido a diversidade dos dados disponíveis para treinamento e teste dos modelos, uma vez que características importantes podem ter sido sub-representadas ou mesmo omitidas do estudo.
Outro fator limitante seria a dependência da qualidade dos dados oriundos das instituições financeiras. Conforme demonstrado neste estudo, as alterações promovidas no Sistema Financeiro Nacional, como, por exemplo, a recente introdução do PIX, prejudicam o padrão dos dados recebidos

das instituições financeiras (envio de registros com rótulos de classificação de lançamentos desatualizados). A precisão e eficácia dos modelos de aprendizado de máquina estão diretamente ligadas à qualidade e integridade dos dados utilizados. Dados incompletos, desatualizados ou incorretos podem comprometer os resultados e a confiabilidade das conclusões.

Essas limitações poderiam ter implicações diretas nos resultados e conclusões do estudo. A redução do volume de dados tratados e a necessidade de anonimização podem ter comprometido a representatividade e a variabilidade dos dados e podem ter levado a um aumento de erros de classificação, especialmente em categorias menos frequentes ou mais complexas. Além disso, a qualidade variável dos dados recebidos das instituições financeiras afetou negativamente a precisão dos modelos. A desatualização das classificações das transações PIX diminuiu a acurácia dos modelos no conjunto de Teste em torno de 4%.

Uma vez que, entre alguns modelos, os resultados foram muito próximos, a supressão de algumas das limitações apontadas poderia levar à escolha de outros modelos.

As próximas etapas do projeto incluirão a replicação do método utilizado neste estudo para treinar um modelo especialista em movimentações de débito. Na sequência, seriam utilizados os modelos de crédito e débito para a reclassificação dos registros com classificação genérica. Seguir-se-ia uma revisão e correção manual dos registros reclassificados, e um novo ciclo de treinamento dos modelos, agora ajustados para atender às especificidades e necessidades do MPBA como, por exemplo, a unificação de categorias semelhantes.

Para trabalhos futuros, sugere-se o desenvolvimento e treinamento de um modelo especializado em linguagem natural, por exemplo, derivado do BERT, que seja adaptado especificamente para o jargão financeiro aqui encontrado. Este modelo especializado poderia ser treinado usando um corpus extenso baseado nos textos dos históricos dos lançamentos bancários para capturar melhor a semântica e as nuances particulares desse vocabulário.

Outra proposta seria a exploração e/ou combinação de outros modelos de *deep learning*, como o GPT-3 e o Transformer-XL, na realização da tarefa ora proposta, para comparação dos resultados.

Também seria interessante a aplicação de técnicas de aprendizado semi-supervisionado, como *Self-Training* e *Co-Training*, para lidar eficientemente com grandes volumes de dados não rotulados. Essas abordagens têm o potencial de melhorar a generalização dos modelos ao aproveitar tanto dados rotulados quanto não rotulados durante o treinamento.

Por fim, explorar modelos de ensemble mais sofisticados, como *stacking* e *blending*, pode ser crucial para integrar de maneira mais eficaz as previsões de diferentes modelos individuais. Isso pode resultar em melhorias significativas na precisão das classificações, especialmente em problemas complexos e multifacetados como a classificação de lançamentos bancários.

REFERÊNCIAS

[1] Ministério Público da Bahia. **ATO Nº 181/2021**. Biblioteca. Disponível em <https://biblioteca.sistemas.mpba.mp.br/biblioteca/index.asp?codigo_sophia=26746>. Acesso em: 18 jun. 2024.

[2] Enclla. **Ação 10/2019: Realizar diagnóstico sobre a qualidade, abrangência e tempestividade das informações prestadas pelas instituições financeiras às autoridades judiciárias, policiais e ministériais via Sistema de Investigação de Movimentações Bancárias (SIMBA) e sugerir melhorias**. Disponível em <https://enccla.camara.leg.br/acoes/arquivos/resultados-enccla-2019/e2019a10-produto-ii-sugestoes-de-melhorias>. Acesso em: 18 jun. 2024.

[3] CALDEIRA, Danile Matos et al. **Ciência de Dados na Administração Pública: Desafios e Oportunidades**. Revista da CGU, [S. l.], volume 15, nº 27, 2023. Disponível em: <https://revista.cgu.gov.br/Revista_da_CGU/article/view/665>. Acesso em: 18 jun. 2024.

[4] ADAM, Isabelle; FAZEKAS, Mihály. **Are emerging technologies helping win the fight against corruption? A review of the state of evidence**. ScienceDirect. Volume 57, Dec. 2021. Disponível em: <https://www.sciencedirect.com/science/article/pii/S016762452100038X>. Acesso em: 18 jun. 2024.

[5] WEST, Darrell. **Using AI and machine learning to reduce government fraud**. Brookings. Sep. 2021. Disponível em: <https://www.brookings.edu/articles/using-ai-and-machine-learning-to-reduce-government-fraud/>. Acesso em: 18 jun. 2024.

[6] LUDEMIR, Teresa Bernarda. **Inteligência Artificial e Aprendizado de Máquina: estado atual e tendências**. Scielo Brasil. Abr. 2021. Disponível em: <https://doi.org/10.1590/s0103-4014.2021.35101.007>. Acesso em: 18 jun. 2024.

[7] GÉRON, Aurélien. **Mãos à Obra: Aprendizado de Máquina com Scikit-Learn, Keras&TensorFlow**: **Conceitos, ferramentas e técnicas para a construção de sistemas inteligentes.** 2ª edição. Rio de Janeiro: Alta Books, 2021.

[8] BUGHIN, Jacques et al. **Artificial Intelligence: The Next Digital Frontier?** McKinsey Global Institute. Discussion Paper. Jun. 2017. Disponível em: <https://www.mckinsey.com/ru/~/media/mckinsey/industries/advanced%20electronics/our%20insights/how%20artificial%20intelligence%20can%20deliver%20real%20value%20to%20companies/mgi-artificial-intelligence-discussion-paper.pdf>. Acesso em: 18 jun. 2024.

[9] WINSTON, Patrick Henry. **Artificial Intelligence**. 3ª edição. Estados Unidos da América: Addison-Wesley, 1992.

[10] NILSSON, Nils J. **Principles of Artificial Intelligence**. 1ª edição. Stanford: Tioga Publishing Company, 1982.

[11] RICH, Elaine; KNIGHT, Kevin. **Artificial Intelligence**. 2ª edição. Rio de Janeiro: McGraw-Hill, 1993.

[12] LUGER, George F. **Artificial Intelligence: Structures and Strategies for Complex Problem Solving**. 6ª edição. Estados Unidos da América: Addison-Wesley, 2009.Disponível em: <https://www.uoitc.edu.iq/images/documents/informatics-institute/exam_materials/artificial%20intelligence%20structures%20and%20strategies%20for%20%20complex%20problem%20solving.pdf>Acesso em: 18 jun. 2024.

[13] MONDAL, Bhaskar. **Artificial Intelligence**: **State of the Art.** 1ª edição. Cham: Springer International Publishing, 2020.Disponível em: <https://www.researchgate.net/publication/337400888_Artificial_Intelligence_State_of_the_Art>Acesso em: 18 jun. 2024.

[14] HAN, Jiawei; KAMBER, Micheline; PEI, Jian. **Data Mining: Concepts and Techniques**. 3ª edição. San Francisco: Morgan Kaufmann, 2011. Disponível em: <https://myweb.sabanciuniv.edu/rdehkharghani/files/2016/02/The-Morgan-Kaufmann-Series-in-Data-Management-Systems-Jiawei-Han-Micheline-Kamber-Jian-Pei-Data-Mining.-Concepts-and-Techniques-3rd-Edition-Morgan-Kaufmann-2011.pdf> Acesso em: 18 jun. 2024.

[15] ZAÏANE, Osmar R. **Principles of Knowledge Discovery in Databases.** Chapter I: Introduction do Data Mining. 1999. Disponível em: < https://www.yumpu.com/en/document/read/30663470/chapter-i-introduction-to-data-mining>. Acesso em: 18 jun. 2024.

[16] SILVA, Leandro August; PERES, Sarajane Marques; BOSCARIOLI, Clodis. **Introdução à Mineração de Dados com aplicações em R**. 1ª edição. São Paulo: Elsevier Brasil, 2016.

[17] IBM. **O que é análise exploratória de dados (EDA)?** Disponível em: <https://www.ibm.com/br-pt/topics/exploratory-data-analysis#:~:text=A%20an%C3%A1lise%20explorat%C3%B3ria%20de%20dados,m%C3%A9todos%20de%20visualiza%C3%A7%C3%A3o%20de%20dados>. Acesso em: 25 jun. 2024.

[18] CHOWDHURY, Gobinda. **Natural language processing**. Annual Review of Information Science and Technology, Volume 37. 2003. Disponível em:< https://pure.strath.ac.uk/ws/portalfiles/portal/131112/strathprints002611.pdf>. Acesso em: 25 jun. 2024.

[19] EMIL Winder. **Natural Language Processing Algorithms**. Uppsala University. Jun. 2023. Disponível em:< https://uu.diva-portal.org/smash/get/diva2:1775578/FULLTEXT01.pdf>. Acesso em: 25 jun. 2024.

[20] KAMRAN, Kowsari et al. **Text Classification Algorithms: A Survey**. MDPI. Charlottesville, Apr. 2019. Disponível em: < https://www.mdpi.com/2078-2489/10/4/150> Acesso em: 25 jun. 2024.

[21] GASPARETTO, Andrea et al**, A Survey on Text Classification Algorithms: From Text to Predictions**. MDPI. Venice, Feb. 2022. Disponível em: < https://www.mdpi.com/2078-2489/13/2/83> Acesso em: 25 jun. 2024.

[22] JIVANI, Anjali Ganesh. **A Comparative Study of Stemming Algorithms**. ResearchGate. Vadodara, Nov. 2011. Disponível em: < https://www.researchgate.net/publication/284038938_A_Comparative_Study_of_Stemming_Algorithms>. Acesso em: 25 jun. 2024.

[23] ZHU, Wenbin; QIU, Runwen; FU, Ying. **Comparative Study on the Performance of Categorical Variable Encoders in Classification and Regression Tasks**. Computer Science > Machine Learning. Disponível em: < https://arxiv.org/abs/2401.09682>. Acesso em: 25 jun. 2024.

[24] POTDAR, Kedar; PARDAWALA, Taher; PAI, Chinmay. **A Comparative Study of Categorical Variable Encoding Techniques for Neural Network Classifiers**. International Journal of Computer Applications. Volume 175 – No. 4, Oct. 2017. Disponível em: < https://www.researchgate.net/profile/Kedar-Potdar-2/publication/320465713_A_Comparative_Study_of_Categorical_Variable_Encoding_Techniques_for_Neural_Network_Classifiers/links/59e6f9554585151e5465859c/A-Comparative-Study-of-Categorical-Variable-Encoding-Techniques-for-Neural-Network-Classifiers.pdf>.Acesso em: 25 jun. 2024.

[25] OUAHI, Mariame; KHOULJI, Samira; KERKEB, Mohammed Laarbi. **Advancing Sustainable Learning Environments: A Literature Review on Data Encoding Techniques for Student Performance Prediction using Deep Learning Models in Education.** E3 Web of Conferences. Volume 477, jan. 2024. Disponível em: < https://www.e3s-conferences.org/articles/e3sconf/pdf/2024/07/e3sconf_star2024_00074.pdf>. Acesso em: 25 jun. 2024.

[26] EKATERINA, Poslavskaya; KOROLEV, Alexey; **Encoding categorical data: Is there yet anything 'hotter' than one-hot encoding?** Huawei Novosibirsk Research Center, Novosibirsk, Dec. 2023. Disponível em: < https://arxiv.org/pdf/2312.16930v1>. Acesso em: 25 jun. 2024.

[27] UDILÃ, Andrei-Iustin. **Encoding Methods for Categorical Data: A Comparative Analysis for Linear Models, Decision Trees, and Support Vector Machines.** EEMCS, Delft University of Technology. Delft, Jun. 2023. Disponível em: <

https://repository.tudelft.nl/islandora/object/uuid%3A10b91b99-2685-4a45-b44e-48fbbf808ce2>. Acesso em: 25 jun. 2024.

[28] DOGRA, Varun et al. **A Complete Process of Text Classification System Using State-of-the-Art NLP Models**. Computational Intelligence and Neuroscience. Jun. 2022. Disponível em: <https://onlinelibrary.wiley.com/doi/10.1155/2022/1883698 > Acesso em: 25 jun. 2024.

[29] VIMALRAJ, Spelmen; PORKODI, R. **A Review on Handling Imbalanced Dat**. Department of Computer Science Bharathiar University Coimbatore, India. Dec. 2018. Disponível em: < https://www.researchgate.net/publication/329584489_A_Review_on_Handling_Imbalanced_Data>. Acesso em: 25 jun. 2024.

[30] RAMYACHITRA, D.; MANIKANDAN, P. **Imbalanced dataset classification and solutions: A review**. International Journal of Computing and Business Research. Coimbatore, Vol. 5, No. 4. Jul. 2014. Disponível em: < https://www.researchmanuscripts.com/July2014/2.pdf>. Acesso em: 25 jun. 2024.

[31] SANTOSO, Budi et al. **Synthetic Over Sampling Methods for Handling Class Imbalanced Problems: A Review**. IOP Conference Series Earth and Environmental Science. March 2017. Disponível em: < https://www.researchgate.net/publication/315976373_Synthetic_Over_Sampling_Methods_for_Handling_Class_Imbalanced_Problems_A_Review>. Acesso em: 25 jun. 2024.

[32] BREIMAN, Leo et al. **Classification and Regression Trees**. 1ª edição. New York: Chapman and Hall/CRC, 1984. Disponível em: <https://www.academia.edu/5867603/Classification_and_Regression_Trees> Acesso em: 25 jun. 2024.

[33] HOSMER, David; LEMESHOW, Stanley; STURDIVANT, Rodney. **Applied Logistic Regression.** 3ª edição. Hoboken: Wiley, 2013. Disponível em: <https://dl.icdst.org/pdfs/files4/7751d268eb7358d3ca5bd88968d9227a.pdf> Acesso em: 25 jun. 2024.

[34] MIENYE, Domor; SUN, Yanxia. **A Survey of Ensemble Learning: Concepts, Algorithms, Applications, and Prospects**. IEEE Access. Johannesburg, Sep. 2022. Disponível em: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9893798>. Acesso em: 25 jun. 2024.

[35] JAMES, Gareth et al. **An Introduction to Statistical Learning: with Applications in R**. 1ª edição. New York: Springer Science & Business Media, 2013. Disponível em: <https://edisciplinas.usp.br/pluginfile.php/8159698/mod_resource/content/1/Statistic_Learning.pdf> Acesso em: 25 jun. 2024.

[36] BREIMAN, Leo. **Random Forests**. Machine Learning, Springer Link. Volume 45. Oct. 2001. Disponível em: <https://link.springer.com/article/10.1023/A:1010933404324>. Acesso em: 26 jun. 2024.

[37] LIN, Jimmy; NOGUEIRA, Rodrigo; YATES, Andrew. **Pretrained Transformers for Text Ranking: BERT and Beyond**. ResearchGate. Jan. 2021. Disponível em: < https://www.researchgate.net/publication/352365063_Pretrained_Transformers_for_Text_Ranking_BERT_and_Beyond >. Acesso em: 26 jun. 2024.

[38] MONICA; AGRAWA, Parul. **A Survey on Hyperparameter Optimization of Machine Learning Models**. IEEE Xplore. Mar. 2024. Disponível em: < https://ieeexplore.ieee.org/document/10489732/metrics#metrics>. Acesso em: 26 jun. 2024.

[39] YANG, Li; SHAMI, Abdallah. **On Hyperparameter Optimization of Machine Learning Algorithms: Theory and Practice**. Elsevier Neurocomputing. Volume 415. November 2020. Disponível em: < https://www.sciencedirect.com/science/article/abs/pii/S0925231220311693>. Acesso em: 26 jun. 2024.

[40] HUTTER, Frank; KOTTHOFF, Lars; FEURER, Matthias. **Automated Machine Learning Methods, Systems, Challenge.** 1ª edição. Switzerland: Springer, 2019. Disponível em: <https://link.springer.com/book/10.1007/978-3-030-05318-5>. Acesso em: 26 jun. 2024.

[41] Ministério Público Federal. **Secretaria de Perícia, Pesquisa e Análise – SPPEA**. Disponível em: < https://www.mpf.mp.br/atuacao-tematica/sppea>. Acesso em: 26 jun. 2024.

[42] Banco Central do Brasil. **Carta Circular n° 3.454 de 14/6/2010.** Disponível em: < https://www.bcb.gov.br/estabilidadefinanceira/exibenormativo?tipo=Carta%20Circular&numero=3454>. Acesso em: 26 jun. 2024.

[43] Presidência da República Casa Civil. **Lei |Complementar nº 105, de 10 de janeiro de 2001**. Disponível em: <https://www.planalto.gov.br/ccivil_03/leis/lcp/lcp105.htm >. Acesso em: 26 jun. 2024.

[44] Presidência da República Casa Civil. **Lei nº 13.709, de 14 de agosto de 2018**. Disponível em: <https://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/l13709.htm>. Acesso em: 26 jun. 2024.

[45] Github. **Validate Brazilian Identification Numbers.** Disponível em: <https://github.com/poliquin/brazilnum>. Acesso em: 26 jun. 2024.

[46] NILC-Embeddings. **Repositório de Word Embeddings do NILC**. Disponível em: < http://nilc.icmc.usp.br/embeddings>. Acesso em: 26 jun. 2024.

[47] BREIMAN, Leo; SPECTOR, Philip. **Submodel Selection and Evaluation in Regression. The X-Random Case**. International Statistical Review. Volume 60, n° 3. Dec. 1992. Disponível em: < https://sites.stat.washington.edu/courses/stat527/s14/readings/BreimanSpector_1992.pdf>. Acesso em: 26 jun. 2024.

[48] BERGSTRA, James; BENGIO, Yoshua. **Random Search for Hyper-Parameter Optimization**. Journal of Machine Learning Research. Volume 13. 2012. Disponível em: < https://www.jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf>. Acesso em: 26 jun. 2024.

[49] ROTHMAN, Denis. **Transformers for Natural Language Processing and Computer Vision: Explore Generative AI and Large Language Models with Hugging Face, ChatGPT, GPT-4V, and DALL-E 3**. 3ª edição. New York: Packt Publishing, 2024.

[50] Hugging Face. **BERTimbau Base (aka "bert-base-portuguese-cased")**. Disponível em: < https://huggingface.co/neuralmind/bert-base-portuguese-cased>. Acesso em: 26 jun. 2024.

[51] Scikit-learn.org. **Train_test_split.** Disponível em: < https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html>. Acesso em: 26 jun. 2024.

[52] ABHINAV, Ravi. **Improving Class Imbalance with Class Weights in Machine Learning.** Aug. 2023**.** Disponível em: <https://medium.com/@ravi.abhinav4/improving-class-imbalance-with-class-weights-in-machine-learning-af072fdd4aa4#:~:text=Using%20Class%20Weights%20to%.20Address%20Class%20Imbalance,-Class%20weights%20offer&text=The%20idea%20is%20to%20assign,make%20better%20predictions%20for%20it>. Acesso em: 26 jun. 2024.

[53] DEVLIN, Jacob et al. **BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.** arxiv.org. May 2019. Disponível em: < https://arxiv.org/pdf/1810.04805>. Acesso em: 26 jun. 2024.

[54] FEBRABAN. **Intercâmbio de Informações entre Bancos e Empresas**. Padrão FEBRABAN 240 Posições. Disponível em:<https://cmsarquivos.febraban.org.br/Arquivos/documentos/PDF/Layout%20padrao%20CNAB240%20V%2010%2011%20-%2021_08_2023.pdf>. Acesso em: 26 jun. 2024.

[55] BANCO BRADESCO. **Layout Conciliação Bancária 240 Posições - Versão 5.0**. Disponível em: <https://banco.bradesco/assets/pessoajuridica/pdf/solucoes-integradas/outros/layout-dearquivo/conciliacao_bancaria_240_posicoes_v_5.pdf>. Acesso em: 26 jun. 2024.

[56] LAMBERT, Nzungize.**The most popular HuggingFace models**. May, 2023. Disponível em: <https://medium.com/@nzungize.lambert/the-most-popular-huggingface-models-d67eaaea392c> Acesso em: 05 jul. 2024.

[57] C. CORTES and V. VAPNIK. **Support-vector networks, Machine Learning**, **vol. 20, no. 3**, pp. 273-297, Sept. 1995. Disponível em: <https://link.springer.com/article/10.1007/BF00994018> Acesso em: 05 jul. 2024.

[58] HOSSIN, M.; SULAIMAN, M.N. **A Review on Evaluation Metrics for Data Classification Evaluations. International Journal of Data Mining & Knowledge Management Process (IJDKP) Vol.5, No.2**, March 2015. Disponível em: <https://aircconline.com/ijdkp/V5N2/5215ijdkp01.pdf>. Acesso em: 25 jun. 2024.

[59] KHAN, Wahab et al. **Exploring the frontiers of deep learning and natural language processing: A comprehensive overview of key challenges and emerging trends.** Natural Language Processing Journal. Volume 4, Sep.2023. Disponível em: < https://www.sciencedirect.com/science/article/pii/S2949719123000237>. Acesso em: 20 ago.2024

[60] Scikit-learn.org**.** Disponível em: <https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html, https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html, https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html.>. Acesso em: 26 jun. 2024.

[61] XGBoost. Disponível em:< https://xgboost.readthedocs.io/en/stable/parameter.html>. Acesso em: 26 jun. 2024.

Sistema FIEB | SENAI CIMATEC
PELO FUTURO DA INOVAÇÃO

CENTRO UNIVERSITÁRIO SENAI CIMATEC
ESPECIALIZAÇÃO EM DATA SCIENCE & ANALYTICS

ATA DE APRESENTAÇÃO DE PROJETO FINAL DE CURSO

Ata de apresentação do Projeto Final de Curso, "**Avaliação de Algoritmos de Aprendizado de Máquina na Classificação de Lançamentos Bancários: Um Estudo Aplicado ao Ministério Público do Estado da Bahia"**, submetido pelo aluno **Marcelo da Silva Lima**, como parte dos requisitos para obtenção do Certificado de **Especialista em Data Science & Analytics** pelo Centro Universitário SENAI CIMATEC, às 18h00 do dia 31 de Julho de 2024. Reuniu-se remotamente pela plataforma Teams, a Banca Examinadora designada pelo Prof Esp Tácito Henrique da Silva Graça – Orientador, constituída pelo Prof Dr Márcio Freire Cruz e pelo Prof Dr Taniel Silva Franklin. A coordenação de Pós-Graduação *Lato Sensu*, deu início aos trabalhos com as devidas orientações, e a exposição foi realizada pelo estudante dentro do prazo de tempo estabelecido. Ao final da apresentação a banca reuniu-se atribuindo a seguinte nota: **8.8** (Oito Pontos e Oito Décimos).

A banca de avaliadores decidiu pela:

(X) Aprovação do trabalho
Caberá ao aluno apresentar em no máximo em 30 (trinta) dias a contar da data de assinatura desta Ata, uma cópia do trabalho em PDF, constando as considerações pontuadas pela banca. A Ata de Apresentação do Projeto Final de Curso deve ser digitalizada e inserida na terceira página do TCC ou como anexo do artigo.

( ) Reprovação do trabalho
O aluno terá que se matricular novamente no TCC – Trabalho de Conclusão de Curso e ser submetido a uma banca avaliadora no semestre seguinte**.**

As ações consequentes ao status de Aprovação deverão obedecer ao prazo proposto acima sob pena do parecer final ser modificado para o status de Reprovado automaticamente e sem possibilidade de recurso.

Para constar, lavrou-se a presente ata que vai assinada por todos os membros da Banca. Por estarem cientes de suas obrigações estão de acordo com os termos desse documento:

Salvador, 31 de Julho de 2024.

Tácito Henrique da Silva Graça Professor Orientador	**Márcio Freire Cruz** Membro da banca
Taniel Silva Franklin Membro da banca	**Patricia Freitas Tourinho** Coordenadora de Pós-Graduação *Lato Sensu*