UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA DE TELEINFORMÁTICA

CURSO DE GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

THYAGO FREITAS DA SILVA

ESTUDO COMPARATIVO DE MODELOS DE MACHINE LEARNING APLICADOS AO CONTEXTO DE DETECÇÃO DE FRAUDES EM CARTÕES DE CRÉDITOS.

FORTALEZA

2023

THYAGO FREITAS DA SILVA

ESTUDO COMPARATIVO DE MODELOS DE MACHINE LEARNING APLICADOS AO CONTEXTO DE DETECÇÃO DE FRAUDES EM CARTÕES DE CRÉDITOS.

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia De Computação do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia De Computação.

Orientador: Prof. Dr. Victor Hugo Costa de Albuquerque

FORTALEZA

2023

THYAGO FREITAS DA SILVA

ESTUDO COMPARATIVO DE MODELOS DE MACHINE LEARNING APLICADOS AO CONTEXTO DE DETECÇÃO DE FRAUDES EM CARTÕES DE CRÉDITOS.

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia De Computação do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia De Computação.

Aprovada em:

BANCA EXAMINADORA

Prof. Dr. Victor Hugo Costa de Albuquerque (Orientador)
Universidade Federal do Ceará (UFC)

MSc Jefferson Silva Almeida
Universidade Federal do Ceará (UFC)

MSc Renê Ripardo Calixto
Universidade Federal do Ceará (UFC)

AGRADECIMENTOS

Ao Prof. Dr. Prof. Victor Hugo C. de Albuquerque por me aceitar e me acompanhar como orientador neste trabalho de conclusão de curso. Ao aluno de pós-graduação Jefferson pelo tempo gasto com revisões, questionamentos e conselhos que contribuiram para que este trabalho pudesser ser concluído.

Aos meus pais por me apoiarem durante a graduação de inúmeras formas, deixando na maioria das vezes, a jornada mais leve.

Aos amigos que fiz dentro e fora da graduação e que mantenho contato até hoje, seja dentro do contexto acadêmico, profissional ou pessoal.

Por último, a mim mesmo por não ter desistido, dado que muitas vezes desistir é o caminho mais rápido e fácil.

"O ato ético é um ato de religação: com o outro, com os seus, com a comunidade, e uma inserção na religação cósmica."

(Edgar Morin)

RESUMO

De acordo com dados disponizabilizados pelo Banco Central do Brasil no decorrer dos anos, o uso de cartões de crédito no país vem se popularizando fazendo com que o mesmo se torne alvo de uma grande quantidade de fraudes. Pode-se definir fraude em cartão de créditos como o uso não autorizado do meio de pagamento, uso esse que pode ocorrer com ou sem a utilização da versão física do cartão, dada a possibilidade de roubo dos dados do mesmo em ataques cibernéticos. Neste contexto, o objetivo principal deste trabalho é, além de apresentar a problemática, realizar um estudo comparativo da aplicação de diferentes técnicas de aprendizado de máquina com o objetivo de automatizar o processo de detecção de fraudes em compras feitas com o uso de cartões de crétido. Dentre as técnicas analisadas estão : *Random Forest*, *Logistic Regression*, *Decision Tree*, *Naive Bayes* e *Multi-Layer Perceptron (MLP)*. Já com relação a análise do desempenho dos modelos foram utilizados os indicadores : acurácia, coeficiente de correlação de Mathews (MCC), $F1_{Score}$, a área sob a curva de Precisão-Recall (AUC-PR), tempo de treino e o tempo utilizado pelos modelos para classificar novas amostras. Como resultado, foi obtido que o modelo *Multi-Layer Perceptron (MLP)* teve o melhor desempenho, tanto em relação a capacidade de detectar transações fraudulentas e genuínas quanto em relação ao tempo gasto durante os processos de treinamento e classificação de novas amostras, métrica essa muito importante devido a existência de cenários onde são feitas milhares de transações de forma simultânea. Por fim, como conclusão tem-se que a viablidade e relevância do uso de diferentes técnicas de machine learning e processamento de dados no combate às fraudes em cartões de crédito são pontos interessantes de se analisar, indicando caminhos promissores para futuros avanços dentro da área de segurança financeira.

Palavras-chave: Cartão de crédito. Fraude. Aprendizado de máquina. Estudo comparativo

ABSTRACT

According to data made available by the Central Bank of Brazil over the years, the use of credit cards in the country has been becoming more popular, making it a target for a significant number of frauds. Credit card fraud can be defined as the unauthorized use of the payment method, which can occur with or without the physical card being used, given the possibility of data theft through cyber attacks. In this context, the main objective of this work is not only to present the problem but also to conduct a comparative study of different machine learning techniques to automate the process of fraud detection in credit card transactions. Among the analyzed techniques are *Random Forest*, *Logistic Regression*, *Decision Tree*, *Naive Bayes*, and *Multi-Layer Perceptron (MLP)*. Regarding the performance analysis of the models, the following indicators were used: accuracy, Mathews correlation coefficient (MCC), $F1_{Score}$, the area under the Precision-Recall curve (AUC-PR), training time, and the time used by the models to classify new samples. As a result, it was found that the *Multi-Layer Perceptron (MLP)* model had the best performance, both in terms of detecting fraudulent and genuine transactions and in terms of the time spent during the training and classification processes of new samples. This metric is crucial due to the existence of scenarios where thousands of transactions are made simultaneously. In conclusion, the feasibility and relevance of using different machine learning and data processing techniques in combating credit card fraud are exciting points to analyze, indicating promising paths for future advancements in financial security.

Keywords: Credit card. Fraud. Machine learning. Comparative study

LISTA DE FIGURAS

Figura 1 – Percentuais de uso por meio de pagamento. 12
Figura 2 – Perdas financeiras causadas por atividades fraudulentas no decorrer dos anos. 16
Figura 3 – Tipos de aprendizado e exemplos de aplicação. 18
Figura 4 – Exemplos de problemas de classificação e regressão. 19
Figura 5 – Árvore de decisão representando a decisão de qual atividade o usuário irá executar. 19
Figura 6 – Fluxo de funcionamento do algoritmo *random forest*. 21
Figura 7 – Exemplo de arquitetura de uma Rede Neural Artificial *Multi-Layer Perceptron* (MLP). 23
Figura 8 – Exemplo de aumento de amostras aplicada a processamento de imagens. . . 25
Figura 9 – Exemplo de execução de validação cruzada com 3 partições (*folds).* 26
Figura 10 – Exemplo de curva precisão-recall (AUC-PR). 30
Figura 11 – Framework utilizando no desenvolvimento do trabalho. 33
Figura 12 – Percentual de transações no conjunto de dados por tipo. 34
Figura 13 – Resultado da busca por valores nulos nas colunas *Amount* e *Time*. 35
Figura 14 – Comparação entre os tempos de treino dos diferentes modelos com e sem o uso de data augmentation. 40
Figura 15 – Matriz de confusão do modelo Random Forest gerada durante o experimento B. 43
Figura 16 – Matriz de confusão do modelo Naive Bayes Forest gerada durante o experimento B. 43
Figura 17 – Comparação entre as métricas obtidas após a execução de todos os modelos sem o uso de data augmentation. 44
Figura 18 – Comparação entre os tempos de classificação dos diferentes modelos com e sem o uso de data augmentation. 45

LISTA DE TABELAS

Tabela 1 – Estado atual do conjunto de dados após a filtragem de colunas. 36
Tabela 2 – Colunas obtidas como resultado do processo de normalização. 36
Tabela 3 – Lista de hiperparamêtros por modelo. 37
Tabela 4 – Métricas utilizadas para avaliar o desempenho dos modelos. 38
Tabela 5 – Resultados obtidos sem aplicação de data augmentation. 39
Tabela 6 – Resultados obtidos com aplicação de data augmentation. 41

LISTA DE ABREVIATURAS E SIGLAS

AVS	*Address Verification Service*
BCB	Banco Central do Brasil
CART	*Classification and Regression Trees*
MCC	Coeficiente de correlação de Matthews
MLP	*Multi-Layer Perceptron*
SMOTE	*Synthetic Minority Over-sampling Technique*
SVM	*Support Vector Machine*

SUMÁRIO

1 INTRODUÇÃO 12
1.1 Objetivos 13
1.2 Organização do trabalho 13
2 FUNDAMENTAÇÃO TEÓRICA 15
2.1 Detecção de fraudes 15
2.2 Machine Learning 17
2.2.1 Tipos de aprendizado 17
2.3 Modelos de classificação 19
2.3.1 Decision Tree 19
2.3.2 Random Forest 20
2.3.3 Logistic Regression 20
2.3.4 Naive Bayes 22
2.3.5 Multi-Layer Perceptron 23
2.4 Data augmentation 24
2.5 Validação cruzada 25
2.6 Métricas de avaliação 27
2.6.1 Acurácia 27
2.6.2 F1-Score 28
2.6.3 Coeficiente de correlação de Matthews 29
2.6.4 Área abaixo da curva precisão-sensibilidade 29
2.7 Trabalhos Relacionados 30
3 METODOLOGIA 32
3.1 Ferramentas computacionais 32
3.2 Análise do conjunto de dados 33
3.3 Pré-processamento 34
3.4 Execução dos modelos 35
3.5 Avaliação dos resultados 38
4 RESULTADOS 39
4.1 Resultados do Experimento A : Sem data augmentation 39
4.2 Resultados do Experimento B : Com data augmentation 41

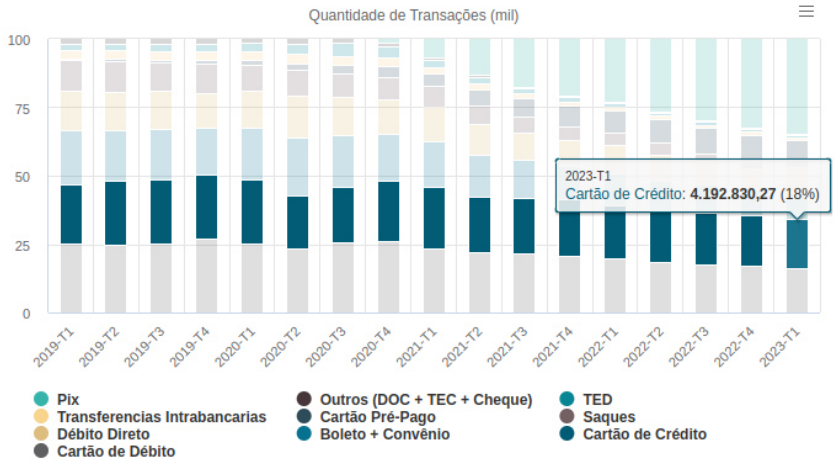
4.3 **Análise dos resultados** 41
5 **CONCLUSÕES E TRABALHOS FUTUROS** 46
REFERÊNCIAS 49

1 INTRODUÇÃO

De acordo as estatísticas de meios de pagamentos disponibilizadas pelo Banco Central do Brasil (BCB) que podem ser visualizadas na Figura 1, durante o primeiro trimestre de 2023 foram efetuadas aproximadamente 4 milhões de transações utilizando cartões de crédito, o que representa cerca de 18% das movimentações financeiras feitas em todo o país (Banco Central do Brasil, 2023). Tal popularidade do meio de pagamento faz com que ele seja alvo de uma grande quantidade de tentativas de fraude.

Segundo o relatório disponibilizado em 2021 pela empresa de cibersegurança Axur desenvolvido com dados de 2020, o Brasil foi o país líder em vazamento de dados de cartões, sendo responsável por cerca de de 45,5% dos casos reportados em todo o mundo (Moura, Hugo, 2021).

Figura 1 – Percentuais de uso por meio de pagamento.



Fonte: Autor: Banco Central do Brasil. Fonte: https://www.bcb.gov.br/estatisticas/spbadendos

.

Como forma de amenizar o impacto financeiro e operacional causadas por atividades fraudulentas utilizando cartões de crédito, instituições financeiras vem investindo cada vez mais em aplicações baseadas em técnicas de aprendizado de máquina, *deep learning* e *big data* capazes de detectar e bloquear de forma automática tais ações não autorizadas. No mercado é possível encontrar uma lista de softwares e empresas especializadas na área que se utilizam desde validações feitas com base no endereço do portador do cartão e o endereço de cobrança fornecido, *Address Verification Service* (AVS), (Chen, James and J.Catalano, Thomas, 2023) à aplicações mais robustas capazes de processar e detectar padrões e anomalias em transações (FICO, 2023).

Já na literatura atual, é possível notar a presença de vários artigos dedicados ao tema de detecção de fraudes em cartões de crédito utlizando técnicas de inteligência artificial. Os métodos empregados neste contexto incluem algoritmos de classificação supervisionada como *Random Forest, Logistic Regression* e *Support Vector Machine* (SVM) (XUAN *et al.*, 2018; RANDHAWA *et al.*, 2018), modelos de clusterização e lógica fuzzy (BEHERA; PANIGRAHI, 2015; MITTAL; TYAGI, 2019) entre outros.

Tomando como base essas referências e outras, este trabalho foi feito como intuito de desenvolver um estudo comparativo entre diversas técnicas encontradas na literatura dentro do contexto de detecção de fraude em cartões de crédito levando em consideração pecualiaridades da problemática.

1.1 Objetivos

O objetivo principal deste trabalho é o desenvolvimento de um estudo comparativo feito com base no uso de diferentes modelos de classificação e de métricas de desempenho a fim de definir qual ou quais dos modelos se adequam melhor ao contexto de detecção de fraudes. Como objetivos específicos é possível listar:

- Avaliação de métricas de desempenho além da acurácia;
- Investigação do uso de *data augmentation* em problemas de desbalanceamento de classes;
- Estudo de diferentes modelos de classificação e suas características.

1.2 Organização do trabalho

Este trabalho foi dividido em 5 capítulos sendo o primeiro deles o de introdução onde foi apresentado a popularidade do uso de cartões de crédito como meio de pagamento e o mesmo como alvo crescente de fraudes, a utilização de técnicas de *machine learning* como meio de detectar tais fraudes de forma automática assim como a motivação e os objetivos que levaram ao desenvolvimento deste trabalho.

Já no capítulo 2, intitulado como fundamentação teórica , são apresentados os conceitos necessários para o entendimento do *framework* utilizado durante o desenvolvimento deste trabalho assim como as técnicas de *machine learning* usadas e os indicadores de desempenho escolhidos para avaliar e comparar tais modelos.

No terceito capítulo é comentada a metodologia utilizada e as etapas que a compõem. Inicialmente é apresentada a configuração do ambiente utilizado assim como as tecnologias

e bibliotecas. Logo após é descrito o conjunto de dados utilizado, suas colunas e todo o processamento realizado a fim de padronizar alguns dos valores contidos no *dataset*. Por último é apresentado como aconteceu o processo de treinamento e alguns dos hiperparametros usados, o uso da técnica *Synthetic Minority Over-sampling Technique* (SMOTE) e de validação cruzada estratificada, ambas usadas na parte experimental, e a forma como os modelos foram avaliados e comparados à luz das métricas escolhidas e suas importâncias.

Os resultados obtidos durante os experimentos são apresentados e discutidos no capítulo 4. Por fim, no último capítulo é feito um resumo do trabalho e dos resultados obtidos elucidando qual dos modelos foi escolhido como o melhor e as razões que levaram a tal decisão. Também são sugeridas melhorias futuras como base para novos projetos dentro do contexto de detecção de fraudes explicitado.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são introduzidos alguns dos conceitos básicos importantes para o desenvolvimento deste trabalho como: problemática de detecções de fraude, a área de aprendizado de máquina, breve descrição dos modelos de classificação utilizados, técnicas para amenizar o problema de desbalanceamento de dados, validação cruzada e métricas de avaliação de desempenho.

2.1 Detecção de fraudes

A área de detecção de fraudes em geral é crucial dentro de diversos setores indo desde transações financeiras realizadas através dos mais diferentes meios de pagamento como cartões de crédito, débito e PIX até a segurança de sistemas de comércio eletrônico, *e-commerce*, e aplicações governamentais.

De acordo com o código penal brasileiro, o conceito de fraude está diretamente ligado ao crime de "estelionato" tratado no artigo 171. A definição de tal artigo é dada como: "Obter, para si ou para outrem, vantagem ilícita, em prejuízo alheio, induzindo ou mantendo alguém em erro, mediante artifício, ardil ou qualquer outro meio fraudulento.". Uma ação fraudulenta pode causar diferentes tipos de danos, sejam eles financeiros, de imagem ou psicológicos, à vitima e a instituições relacionadas ao ato, como bancos e emissores de cartões por exemplo.

Dentro do contexto de fraudes em cartões de crédito é possível citar diferentes tipos de fraude, como:

- Roubo ou perda de cartão: Quando o cartão físico é roubado ou perdido, o fraudador pode usá-lo para fazer compras em lojas físicas ou sistemas de *e-commerce*, tudo isso antes que o titular do cartão perceba e o bloqueie;
- *Skimming*: Neste tipo de fraude dados de cartões são capturados através de dispositivos chamados de "*skimmer*" e que podem ser instalados em terminais de pagamento (caixas eletrônicos, máquinas de cartão em lojas, etc.) (Chauncey Crail, Dia Adams, 2023). Os dados obitidos são, então, usados para criar cartões falsos que podem ser utilizados pelos fraudadores;
- Invasão de sistemas de pagamento: Nesses casos, criminosos invadem sistemas de pagamentos, acessando e extraindo assim dados de cartões de crédito e realizando transações fraudulentas;

- *Phishing*: Fraudes deste tipo involvem o uso de técnicas baseadas em engenharia social de forma a obter dados sensíveis dos cartões de crédito através de e-mails, mensagens, sites falsos e ligações se passando por instituições financeiras legítimas como bancos e empresas de comércio eletrônico. Os dados obtidos são utilizados posteriormente sem o consentimento dos usuários. (Chauncey Crail, Dia Adams, 2023)

De acordo com dados disponibilizados no ano de 2021 pela empresa de tecnologia Axur, durante o ano de 2020 cerca de quase 3 milhões de cartões foram expostos na *surface, deep* e *dark web*. Dentre esses resultados, o volume de dados de cartões vulneráveis pertencentes a brasileiros representa cerca de 45,4% dos dados expostos (Moura, Hugo, 2021). Já no que tange a valores monetários, de acordo com um relatório desenvolvido pela empresa *Nilson Report*, no ano de 2021 cerca de 32 bilhões de dólares foram perdidos em todo mundo devido a ações fraudulentas que se utilizavam de cartões de crédito. Perda essa que afeta tanto emissores quanto comerciantes e adquirentes (Nilson Report, 2022).

Figura 2 – Perdas financeiras causadas por atividades fraudulentas no decorrer dos anos.

Fonte: Autor: Nilson Report, Kamil. Fonte: https://nilsonreport.com/newsletters/1232/

.

Na Figura 2 é possível visualizar o crescimento dos prejuízos financeiros gerados por atividades financeiras ilegítimas utilizando cartões de crédido ao redor do mundo. Tal crescimento no número de fraudes e o aumento na popularidade do uso de cartões de crédito

como meio de pagamento fizeram com que aumentasse o interesse em plataformas capazes de detectar de forma automática transações ilegais através de diferentes técnicas de processamento de dados e detecção de padrões como aprendizado de máquina, diminuindo assim o número de recursos financeiros e operacionais necessários para que o processo de detecção e tratamento legal da transação seja feito.

No mercado hoje é possível encontrar inúmeras plataformas e serviços capazes de detectar fraudes em transações feitas por cartões de crétido como é o exemplo do sistema *Address Verification Service (AVS)* que valida a transação com base no endereço do portador do cartão e o endereço de cobrança fornecido durante o cadastro do cartão ou da compra (Chen, James and J.Catalano, Thomas, 2023). Outros sistemas como o *SAS Fraud Detection and Investigation* e *FICO Falcon Fraud Manager* se utilizam de técnicas mais robustas como *machine learning* e inteligência artificial a fim de detectar e prever fraudes, incluindo em cartões de crédito (SAS, 2023; FICO, 2023).

2.2 Machine Learning

Machine learning, ou aprendizado de máquina, é um campo da inteligência artificial em que se concentram a pesquisa e desenvolvimento de algoritmos e modelos capazes de aprender à partir de conjuntos de dados , fazendo assim com tais técnicas sejam capazes de fazer previsões númericas ou categóricas sem explicitamente serem codificadas para tal. Nesta subseção serão apresentados brevemente os conceitos principais da área assim como exemplos de uso.

2.2.1 Tipos de aprendizado

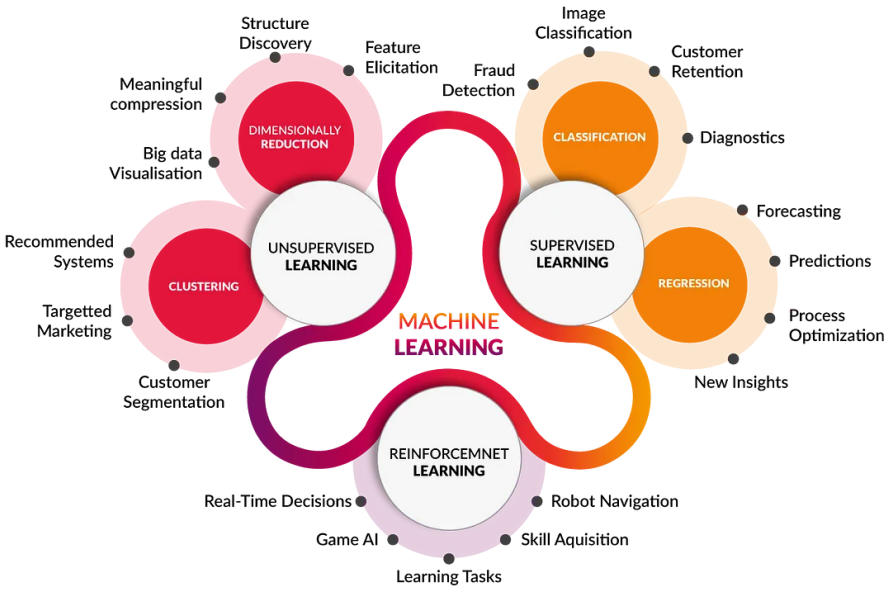
O aprendizado de máquina pode ser classificado em diferentes tipos, dependendo da natureza do problema e do objetivo do modelo. Os principais tipos de aprendizado são:

- **Aprendizado supervisionado**: Neste tipo de aprendizado, o modelo é treinado usando um conjunto de dados previamente rotulados, ou seja, o usuário fornece ao algoritmo uma lista de características e qual deve ser a saída para aquela amostra de forma que o modelo deve determinar uma forma de relacionar as características apresentadas à saída esperada. Casos onde a saída deve assumir rótulos pré-definidos (como uma cor, uma doença, uma espécie de planta e etc), são conhecidos como problemas de classificação. De forma similar, caracterizamos como problemas de regressão casos onde a saída do modelo

deve ser um valor real, como por exemplo o valor de um sensor de temperatura utilizado em processos da indústria química (BISHOP, 2006).

- **Aprendizado não supervisionado**: Diferente do aprendizado supervisionado, neste tipo o algoritmo de *machine learning* é treinado com dados que não possuem rótulos, de forma que não existem informações prévias como classe ou valor numérico esperado. Este tipo de abordagem tem como resultado esperado *insights* sobre os dados que podem ser utilizados em conjunto com técnicas de análise de dados. Em alguns casos é possível utilizar durante o treinamento do modelo uma fração de dados rótulados, o que pode aumentar o desempenho dos algoritmos. Essa abordagem híbrida é conhecida na literatura como aprendizado semi-supervisionado (ENGELEN JESPER E.AND HOOS, 2020). Seu uso vai desde mineração de dados à bioinformática, visão computacional e robótica.
- **Aprendizado por reforço**: Por último temos o aprendizado por reforço como uma abordagem onde um agente autônomo interage com um ambiente e aprende a tomar ações que maximizam um valor denominado como "recompensa"que se acumula ao longo do tempo. Ao contrário do outros tipos de aprendizado citados anteriormente, este tipo é baseado em um sistema de recompensa e punição (KAELBLING *et al.*, 1996).

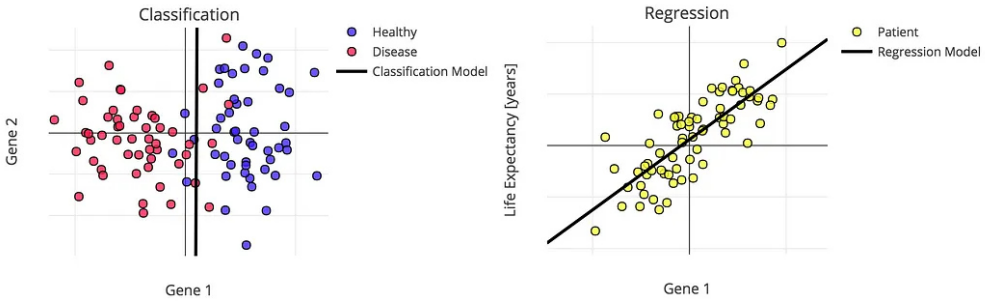
Figura 3 – Tipos de aprendizado e exemplos de aplicação.



Fonte: Autor: Cognub. Fonte: http://www.cognub.com/index.php/cognitive-platform/

.

Figura 4 – Exemplos de problemas de classificação e regressão.



Fonte: Autor: Krzyk, Kamil. Fonte: https://towardsdatascience.com/coding-deep-learning-for-beginners-types-of-machine-learning-b9e651e1ed9d
.

2.3 Modelos de classificação

Os fundamentos teóricos basicos dos modelos de aprendizado de máquina utilizados neste trabalho serão descritos nas subseções sequentes assim como seus usos na literatura em diferentes áreas.

2.3.1 Decision Tree

As árvores de decisão são algoritmos de aprendizado de máquina que permitem a tomada de decisões com base em uma sequência de perguntas ou condições sobre as características dos dados. Na literatura é possível encontrar o uso desse tipo de modelo em diferentes aplicações como detecção de fraudes em cartões de crédito (DILEEP *et al.*, 2021) e de falhas de alta impedância em redes distribuídas (SHAHRTASH; SARLAK, 2006). Sua estrutura de decisões é representada como uma árvore, na qual cada nó interno corresponde a uma pergunta e cada folha corresponde a uma decisão ou classificação feita com base no conjunto de perguntas.

Figura 5 – Árvore de decisão representando a decisão de qual atividade o usuário irá executar.

Fonte: Elaborada pelo autor (2023).

Na Figura 5 temos um exemplo de árvore de decisão sendo utilizada para definir se uma pessoa irá pedir pizza ou não. No caso do exemplo, as folhas são representadas pelas atividades “Ir dormir” e “Pedir uma pizza!” e os questionamentos como “Estou de dieta?” representam os nós internos de decisão. Ainda sobre a Figura 5, é possível verificar que dentre as inúmeras vantagens do uso de *decisions trees*, a interpretabilidade é uma delas dado que é muito fácil compreender e explicar como uma decisão é tomada em cada etapa da árvore pois sua estrutura permite visualizar de forma objetiva o fluxo de decisões tomadas até chegar em um ou mais folhas.

2.3.2 Random Forest

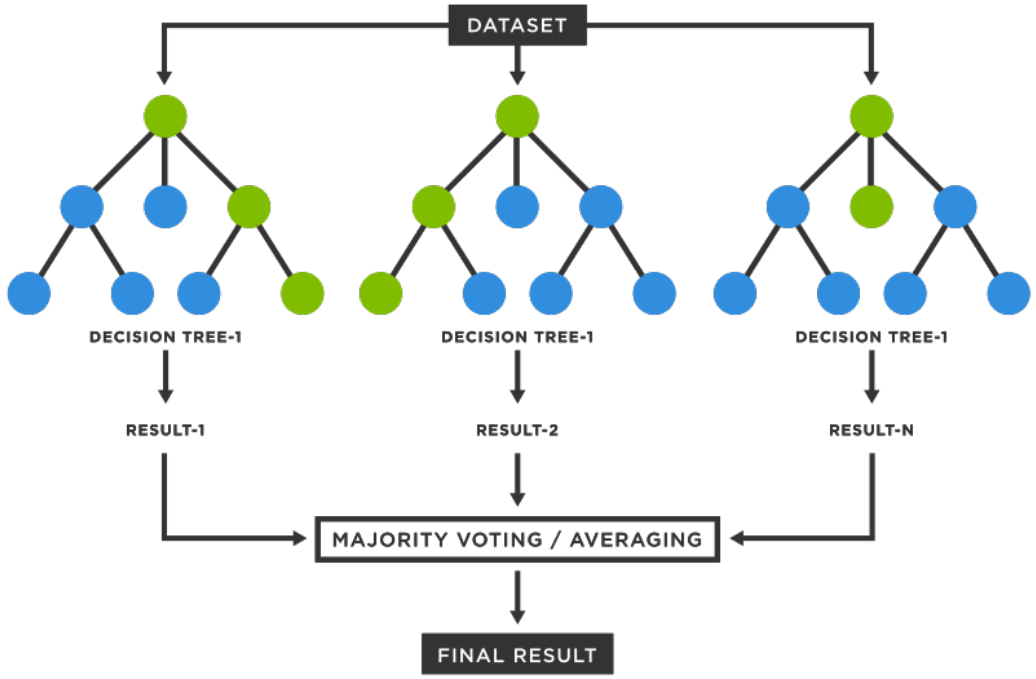
A *random forest*, ou floresta aleatória, é uma técnica de aprendizado de máquina amplamente utilizada em problemas de classificação como detecção de fraudes em cartões de crédito (DILEEP *et al.*, 2021) e regressão criada por Leo Breiman (2001) e proposta em seu artigo intitulado “*Random Forests*” (BREIMAN, 2001). Sua execução se dá a partir da criação de múltiplas arvores de decisão nas quais cada árvore é construída a partir de uma amostra aleatória com reposição do conjunto de treinamento. Quando aplicada em problemas de classificação, o resultado é obtido através de votação dentre os resultados obtidos pelas árvores de decisão, sendo escolhida a classe que apresentar a maior frequência. No caso de problemas de regressão, o resultado final é calculado através da média das previsões de todas as árvores.

O modelo *random forest* possui uma lista de hiperparamêtros, sendo alguns deles herdados da técnica *decision tree* que compõe a floresta aleatória, e sua implementação pode ser encontrada em bibliotecas como *scikit-learn* desenvolvido em python assim como todos os outros modelos utilizados neste trabalho.

2.3.3 Logistic Regression

Logistic Rregression, ou regressão logística, é uma famosa técnica de*machine learning* muito utilizada em problemas de classificação binária onde o objetivo é prever a probabilidade da amostra testada pertencer a uma das classes possíveis. Seu uso pode ser notado em diferentes áreas do conhecimento humano como na detecção de doenças do coração (LI *et al.*, 2020), agricultura inteligente (RAJESHWARI *et al.*, 2021), detecção de falhas em sistemas de distribuição de energia (XU; CHOW, 2006) e identificação de fraudes em transações usando cartões de crédito (AWOYEMI *et al.*, 2017).

Figura 6 – Fluxo de funcionamento do algoritmo *random forest*.



Fonte: Autor: TIBCO. Disponível em: https://www.tibco.com/reference-center/what-is-a-random-forest

Tal modelo pode ser definido através da equação logo abaixo onde o resultado é a probabilidade da amostra X pertencer a classe positiva, e é a base do logaritmo natural e z é a combinação linear das variáveis de entrada X ponderadas pelos coeficientes β_0, β_1, β_2, β_3, β_4 ...β_n do modelo.

$$p(X) = \frac{1}{1+e^{-z}} \tag{2.1}$$

$$z = \beta_0 + x_1\beta_1 + x_2\beta_2 + x_3\beta_3 + x_4\beta_4 + ... + x_n\beta_n \tag{2.2}$$

Tais coeficientes determinam a importância de cada variável independente na previsão da probabilidade da amostra pertencer à classe positiva e podem ser encontrados através do uso da função de máxima verossimilhança (JR *et al.*, 2013).Com relação a análise do resultado da função, caso a probabilidade resultante esteja abaixo de um determinado limiar, *threshold*, especificado durante a execução do modelo a amostra será classificado como pertencente a classe negativa 0, seu funcionamento é análogo para a classe positiva 1.

2.3.4 Naive Bayes

O algoritmo Naive Bayes é um método de classificação supervisionada baseada em medidas probabilisticas e tem seu uso associado a diferentes problemas das mais diferentes áreas como: detecção de *fake news* (DESHPANDE; RAO, 2017), diagnóstico de falhas na fabricação de semicondutores (FAN *et al.*, 2020), identificação de depressão (DESHPANDE; RAO, 2017) e de fraudes em cartões de crédito (AWOYEMI *et al.*, 2017). Seu funcionamento está diretamente associado ao Teorema de Bayes expresso na equação logo abaixo.

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)} \tag{2.3}$$

Sendo:

- $P(A|B)$ representa a probabilidade de ocorrência do evento A, dado que o evento B já ocorreu;
- $P(B|A)$ representa a probabilidade de ocorrência do evento B, dado que o evento A já ocorreu;
- P(A) é a probabilidade de ocorrência do evento A, independentemente de qualquer outra informação;
- P(B) é a probabilidade de ocorrência do evento B, independentemente de qualquer outra informação.

Dentro do contexto de um problema de classificação existe a necessidade de encontrar a probabilidade de uma amostra X pertencer a uma classe W com base em suas características. Logo, temos que o teorema pode ser escrito como:

$$P(W|X) = \frac{P(X|W) \times P(W)}{P(X)} \tag{2.4}$$

O uso do termo *naive*, que pode ser traduzido para o português como ingênuo(a), na nomenclatura do modelo está relacionado ao fato de que o algoritmo faz uma suposição ingênua de independência condicional entre os atributos, independência essa nem sempre encontrada ao analisar dados reais. É importante citar que existem variações do algoritmo *Naive Bayes* e que o uso de uma versão específica está diretamente associada ao tipo de dado tratado. Dentre as variações é possível citar:

- Naive Bayes Gaussiano: Utilizado para dados contínuos, assume que os atributos seguem uma distribuição Gaussiana (PEDREGOSA *et al.*, 2011);
- Naive Bayes Multinomial: Adequado para dados discretos, como contagens de palavras para classificação de texto (PEDREGOSA *et al.*, 2011);

- Naive Bayes Bernoulli: Usado para dados binários, onde cada atributo é binário (0 ou 1) (PEDREGOSA *et al.*, 2011).

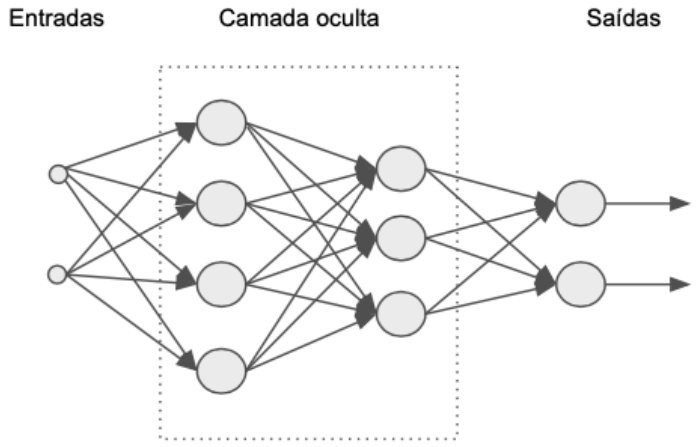
Neste trabalho foi usado a versão intitulada como *Naive Bayes Gaussiano* devido a natureza dos dados presentes no conjunto de dados utilizado para treinamento, avaliação e teste dos modelos.

2.3.5 Multi-Layer Perceptron

Muito utilizado em problemas de classificação tanto binárias quanto multiclasse onde o resultado do modelo é uma classe dentre uma lista de outras, o modelo MLP, pode ser encontrado na literatura sendo utilizado em diferentes tipos de aplicação como classificação de images (FOODY; MATHUR, 2004) e detecção de doenças cardiovasculares (LI *et al.*, 2020).

A base teórica por trás desta técnica, intitulada de Rede Neural Multilayer Perceptron (MLP), é fundamentada na tentativa de modelar de forma matemática o funcionamento do cerébro humano e seus componentes através de redes neurais artificais (NWANKPA *et al.*, 2018). Uma MLP normalmente é constituída de múltiplas camadas formadas por vários neurônios, também intitulados de *perceptrons*, onde cada instância dessa entidade é uma unidade de processamento capaz de computar as informações recebidas das camadas anteriores e produzir resultados que serão repassados para as camadas subsequentes (WYTHOFF, 1993).

Figura 7 – Exemplo de arquitetura de uma Rede Neural Artificial MLP.



Fonte: Autor: WEISS, V. A. Fonte: https://ateliware.com/blog/redes-neurais-artificiais

Na Figura 7 é possível visualizar um exemplo de arquitetura que compõe o modelo MLP. A camada de entrada representa a etapa inicial onde o modelo recebe os dados do problema. As camadas seguinte intituladas como *hidden layers*, ou camada ocultas, são compostas por

múltiplos neurônios e tem como finalidade processar e extrair caraterísticas e padrões dos dados resultantes da camada de entrada. Cada neurônio presente nas camadas ocultas calcula uma combinação linear ponderada usando os dados que recebeu das camadas anteriores e aplica uma função chamada de função de ativação (NWANKPA *et al.*, 2018). O uso de funções não-lineares nessa etapa do processamento permite que o MLP seja capaz de aprender relações mais complexas entre os dados de entrada e os de saída, tornando assim o modelo mais robusto e aumentando a possibilidade de uso em diferentes problemas.

Neste trabalho a função de ativação utilizada foi a conhecida como *ReLU* e que tem sua equação definida logo abaixo. Tal função retorna o próprio valor de entrada se for positivo e zero caso contrário. Algumas das razões que levam a popularidade dessa função são sua velocidade, desempenho e capacidade de generalização (ZEILER *et al.*, 2013; LECUN *et al.*, 2015). O resultado do processamento feito pelas camadas ocultas é utilizado como entrada na cama de saída que é responsável por computar e apresentar ao usuário o(s) resultado(s) final(is) do modelo (NWANKPA *et al.*, 2018).

$$\mathrm{ReLU}(x) = \max(0, x) \tag{2.5}$$

O processo de treinamento do modelo MLP pode ser descrito basicamente em 2 etapas, sendo elas:

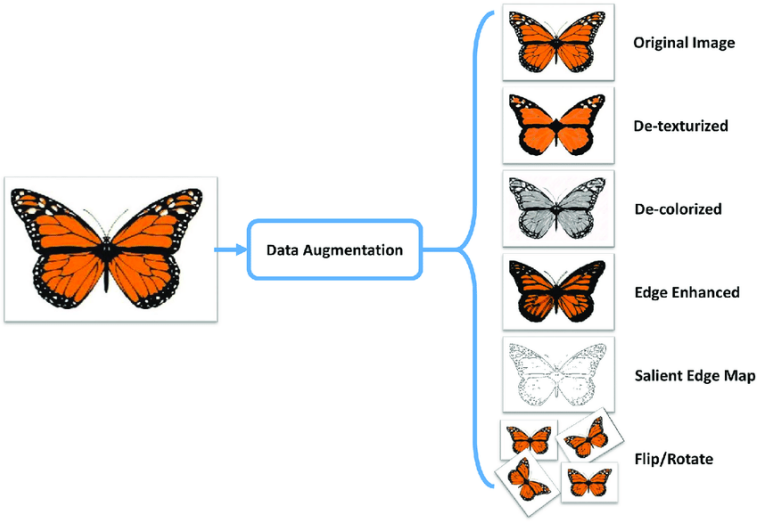
- *Forward propagation* ou propagação direta: Nesta etapa já descrita anteriormente, os dados e o resultados do neurônios que compõem as camadas fluem dentro da rede iniciando na camada de entrada, passando pelas *n* camadas ocultas e tendo sua saída exposta pela camada de saída (WYTHOFF, 1993);
- *Backpropagation* ou retro propagação do erro: Em tal etapa os resultados da camada de saída são comparados aos resultados reais do dados gerando assim um erro. Esse erro se propaga dentro da rede fazendo com que os pesos sejam recalculados e otimizados levando em consideração uma função de custo (WYTHOFF, 1993). Essa otimização muitas vezes é feita utilizando um algoritmo conhecido como Gradiente Descendente Estocástico.

2.4 Data augmentation

O termo *data augmentation*, aumento de amostras, refere-se a um conjunto de técnicas capazes de gerar novos exemplos de treinamento por meio de transformações nos dados pré-existentes, com o objetivo de aumentar a diversidade e a quantidade dos dados

de treinamento melhorando a capacidade do modelo de generalizar para dados não vistos (SHORTEN; KHOSHGOFTAAR, 2019). Na Figura 8 é possível ver a aplicação desse tipo de metologia ao gerar novas imagens com base em uma imagem já existente aplicando diversos tipos de técnicas de processameto de imagem como binarização, rotação e segmentação.

Figura 8 – Exemplo de aumento de amostras aplicada a processamento de imagens.



Fonte: https://medium.com/secure-and-private-ai-writing-challenge/data-augmentation-increases-accuracy-of-your-model-but-how-aa1913468722

Uma das técnicas de *data augmentation* é conhecida como SMOTE e tem seu uso principal atrelado à problemas onde existe um desbalanceamento na quantidade de amostras de uma ou mais classes no conjunto de dados. Tal técnica foi proposta por Nitesh V. Chawla, et al., em 2002 e tem como principal ideia a criação de amostras sintéticas para a classe (ou classes) minoritárias, de forma aumentar a representativade das amostras. Seu funcionamento se da através do uso do algoritmo de vizinho mais próximo onde a nova amostra da classe minoritária é o resultado de combinação entre a amostra analisada e um de seus vizinhos (CHAWLA *et al.*, 2002).

2.5 Validação cruzada

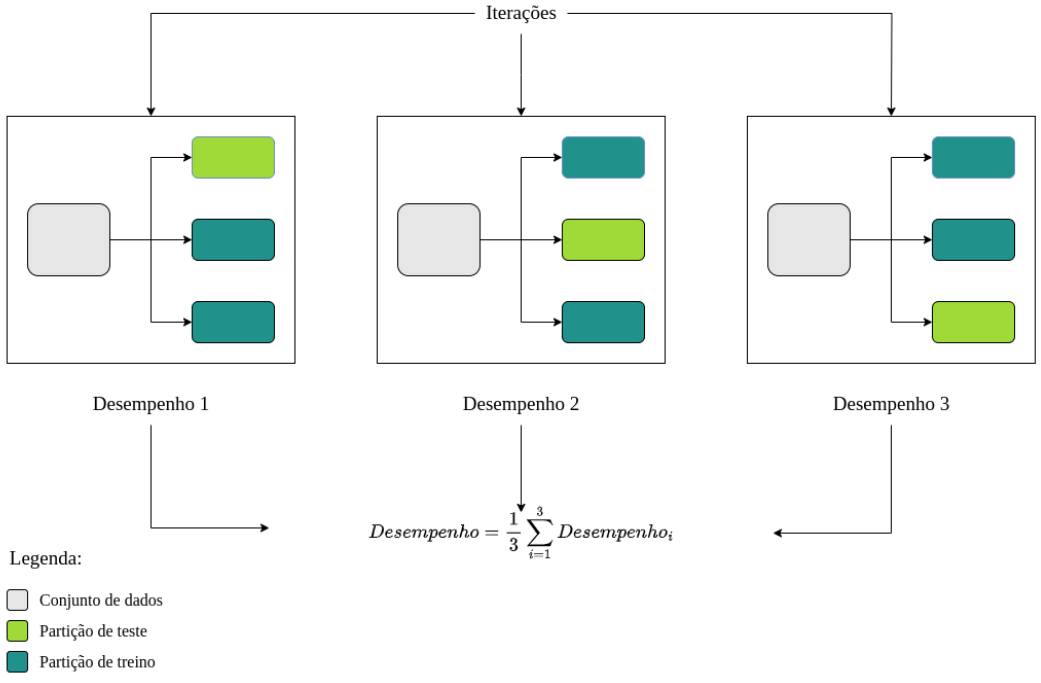
A validação cruzada, em inglês *cross validation*, é uma técnica amplamente utilizada na avaliação de modelos de aprendizado de máquina. Ela permite uma estimativa robusta do desempenho do modelo, levando em consideração a variação dos resultados devido à aleatoriedade da divisão dos dados em conjuntos de treinamento e teste. Através da *cross validation*, podemos obter uma medida mais confiável do desempenho do modelo em dados não vistos.

Existem várias abordagens de validação cruzada, incluindo a validação cruzada *k-fold*, *leave-one-out cross-validation (LOOCV)*, *hold-out* e a validação cruzada *k-fold* estratificada, utilizada neste projeto e introduzida por Thomas Dietterich em seu trabalho intitulado *"Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms"*.

O princípio básico por tras da validação cruzada consiste em separar de forma aleatório o conjunto de dados em 2 partes, sendo elas o conjunto de testes e o conjunto de treino. O conjunto de treinamento é usado para ajustar o modelo aos dados, enquanto o conjunto de teste é usado para avaliar o desempenho do modelo. Este processo é repetido um determinado número de vezes, à depender da técnica utilizada (DIETTERICH, 1998).

Na Figura 9 é possível visualizar um exemplo básico da validação cruzada k-fold, tendo o valor de k igual 3. Nesta variação da técnica, o conjunto de dados é divido em k partições (também chamados de *folds*), onde uma das partições é utilizada como conjunto de teste e as k-1 partições restantes são usadas como conjunto de treino do modelo. Logo após o treino e teste do modelo de *machine learning* escolhido, a etapa de separação do conjunto de dados em k partições é refeita e o modelo é treinado e executado novamente. Este processo é refeito até que o número de iterações seja igual ao valor de k. Uma vez concluídas as k iterações, é calculada a média, ou outra medida estatística como a mediana, das métricas obtidas nas k iterações para obter uma estimativa geral do desempenho do modelo.

Figura 9 – Exemplo de execução de validação cruzada com 3 partições (*folds*).



Fonte: Elaborada pelo autor (2023).

Uma variação da validação cruzada k-partições conhecida como *stratified k-fold cross-validation*, ou validação cruzada k-partições estratificada, possui ampla utilização em problemas onde existem desbalanceamento na quantidade de amostras por classes no conjunto de dados. Seu uso frequente nesses tipos de situações se dá pelo fato de que durante a criação das partições a proporção de cada classe no conjunto de dados original é levada em consideração de tal maneira que as partições resultantes do processo terão aproximadamente a mesma proporção.

Como exemplo temos um conjunto de dados com 2 classes, A e B, onde a proporção é de 33% e 66% respectivamente. As partições geradas através do uso dessa base de dados também irão possuir, aproximadamente, 33% de amostras da classe A e 66% de amostras da classe B. Desta forma garantimos que o modelo terá a oportunidade de aprender com exemplos de todas as classes durante o treinamento, garantia essa que não existe na versão convencional da validação cruzada k-partições.

2.6 Métricas de avaliação

Nesta seção, apresentamos as métricas de avaliação de desempenho utilizadas para mensurar o desempenho dos modelos de aprendizado de máquina. Tais indicadores fornecem medidas quantitativas que nos permitem mensurar e comparar a eficácia dos modelos quando aplicados em diferentes cenários como classificação. A escolha de uso de uma métrica pode variar de acordo com a natureza do problema dado que cada indicador fornece uma perspectiva diferente do desempenho do modelo e enfatiza aspectos diferentes da classificação.

2.6.1 Acurácia

Frequentemente utilizada como uma das primeiras métricas de avaliação de desempenho de um modelo de *machine learning* por ser um indicador fácil de calcular e de interpretar, a acurácia pode ser definida como a relação entre o número predições corretas e o total de predições realizadas pelo modelo. Matematicamente tal indicador pode ser expressa pela equação abaixo e seu valor varia entre 0 e 1. Quando seu valor máximo é alcançado, indica que o modelo classificou corretamente todas as amostras.

$$Acuracia = \frac{\text{Número de predicões corretas}}{\text{Número total de predições}} \tag{2.6}$$

Outra abordagem para descrever matemáticamente a acurácia é basear-se nos termos definidos na matriz de confusão, como verdadeiro positivo (VP), verdadeiro negativo (VN), falso

positivo (FP) e falso negativo (FN).

$$Acuracia = \frac{VP + VN}{VP + VN + FP + FN} \tag{2.7}$$

Apesar de ser uma métrica amplamente utilizada e considerada relevante, a acurácia pode não ser a escolha mais adequada em situações em que lidamos com conjuntos de dados desbalanceados, como casos de diagnóstico médico e detecção de fraudes, pois pode apresentar resultados enviesados devido à desproporção entre as classes. Um modelo que simplesmente prevê a classe majoritária pode obter uma alta acurácia, mesmo que seja incapaz de classificar corretamente as amostras da classe minoritária. Portanto, é importante que a acurácia não seja o único indicador analisado nesses tipos de situações e métricas adicionais como $F1_{score}$ e O coeficiente de correlação de Matthews (MCC) sejam consideradas.

2.6.2 F1-Score

$F1_{score}$ ou $F_{measure}$ é uma métrica amplamente utilizada como indicador de avaliação de desempenho de um modelo em problemas de classificação binária onde existe um desabalanceamento na quantidade de amostras. Seu valor pode variar entre 0 e 1 e pode ser obtido através da média harmônica entre precisão e sensibilidade.

$$F1_{score} = 2 \times \frac{Precisao \times Sensibilidade}{Precisao + Sensibilidade} \tag{2.8}$$

$$Precisão = \frac{VP}{VP + FP} \tag{2.9}$$

$$Sensibilidade = \frac{VP}{VP + FN} \tag{2.10}$$

Algumas interpretações para seu valor são:

- Valores próximos de 1 indicam um bom desempenho do modelo. Isso significa que o modelo é capaz de identificar corretamente a maioria das instâncias positivas (possui um valor alto de precisão) ao mesmo tempo em que minimiza os falsos positivos (sensibilidade);
- Valores próximos de 0 indicam um desempenho ruim do modelo. Isso pode ser devido a uma baixa precisão , ou seja, o modelo está classificando amostras como positivas quando na realidade são negativas, uma baixa sensibilidade, quando o algoritmo classifica erroneamente uma grande quantidade de amostras como negativa, ou ambos;
- Se o $F1_{score}$ estiver em algum lugar entre 0 e 1, isso indica um desempenho moderado do modelo o que siginifica que existe um equilibro entre os valores de precisão e sensibilidade.

2.6.3 Coeficiente de correlação de Matthews

O Coeficiente de correlação de Matthews (MCC), também conhecido como coeficiente ϕ, é uma métrica utilizada como um bom critério de desempenho em problemas de classificação binários onde existe um desbalanceamento na quantidade de amostras utilizada durante o processo de treino do modelo (CHICCO; JURMAN, 2020). Sua classificação como um indicador robusto para esse tipo de problema se da pelo fato de que para calculá-lo é qnecéssario levar em consideração todos os quatro elementos presentes na matriz de confusão de forma a englobar tanto os erros quanto os acertos do modelo.

Seu valor pode ser obtido através da equação abaixo e tem seu valor podendo variar entre -1 e 1, onde:

$$MCC = \frac{VP \times VN - FP \times FN}{(VP+FP)(VP+FN)(VN+FP)(VN+FN)} \tag{2.11}$$

- Valores próximos de 1 indicam um bom desempenho do modelo, ou seja, existe uma alta taxa de acertos do modelo, tanto dos positivos quanto dos negativos;
- Valores próximos de 0 indicam que não existe uma relação direta entre os resultados reais e os previstos, de forma que seus resultados não são melhores do que um modelo que fornece classificações aleatórias;
- Valores próximos de -1 indicam que o modelo está se comportando de modo inverso ao esperado. Em outras palavras, o algoritmo está classificando as amostras de forma invertida, de forma que, amostras classificadas como positivas na verdade são negativas e vice-versa.

2.6.4 Área abaixo da curva precisão-sensibilidade

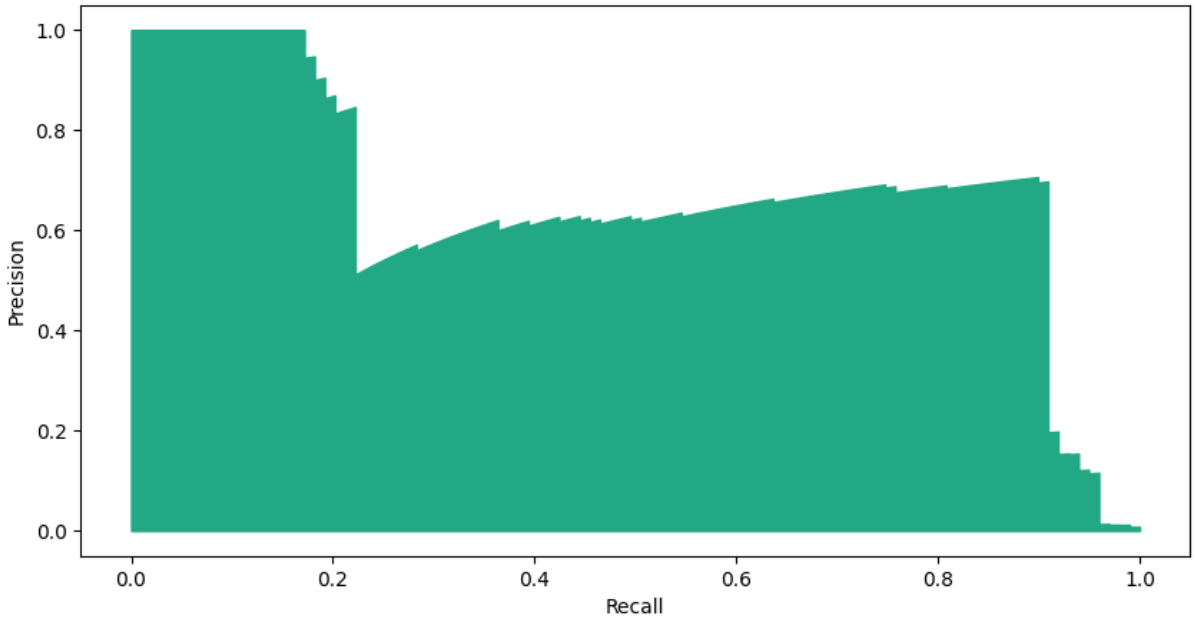
A área abaixo da curva precisão-sensibilidade, ou *precision-recall* (AUC-PR), é um indicador frequentemente utilizado para avaliar o desempenho de modelos de classificação em problemas onde o desbalanceamento de classes é uma preocupação, como diagnóstico médico, detecção de fraudes e eventos raros (DAVIS; GOADRICH, 2006; SOFAER *et al.*, 2019; KHAN; RANA, 2019). Tal métrica considera a relação entre a precisão e *recall* e pode ser obtida resolvendo a equação integral definida logo abaixo:

$$AUC\text{-}PR = \int_0^1 p(r)dr \tag{2.12}$$

Sua interpretação é semelhante à da curva *ROC (Receiver Operating Characteristic)*, definida como a relação entre a taxa de falsos positivos e a taxa de verdadeiros positivos conhecida

como sensibilidade. Quanto maior a área abaixo da curva, melhor o desempenho do modelo em termos da capacidade de classificar corretamente os exemplos positivos e minimizar os falsos positivos. Na Figura 10 é possível visualizar um exemplo de gráfico que mostra a curva precisão (*precision*) - sensibilidade (*recall*).

Figura 10 – Exemplo de curva precisão-recall (AUC-PR).



Fonte: Elaborada pelo autor (2023).

2.7 Trabalhos Relacionados

Nesta etapa serão descritos trabalhos relacionados encontrados na literatura dentro do contexto de detecção de fraudes que se utilizam de diferentes modelos de *machine learning*.

No trabalho desenvolvido em (XUAN *et al.*, 2018) foi proposto uma metodologia baseada no uso de 2 variações do algoritmo conhecido como *random forest*. Sua variação é dada pela composição das florestas aleatórias, sendo 1 composta por árvores de decisão e outra baseada em *Classification and Regression Trees* (CART). Como resultado os autores citam que, apesar do modelo *random forest* obter bons resultados dentro dos experimentos desenvolvidos, existem problemas que precisam ser tratados em trabalhos futuros como o desbalanceamento do conjunto de dados que exige que métricas diferentes da acurácia sejam usadas para avaliação do desempenho dos modelos.

Já no trabalho (AWOYEMI *et al.*, 2017) é feito um estudo comparativo de diversas técnicas de aprendizado supervisionado dentro do contexto de detecção de fraudes em cartões de

crédito, onde o conjunto de dados utilizado pelos autores é o mesmo usado no desenvolvimento deste artigo. A lista de modelos de classificação utilizados é composta pelas técnicas *Naive Bayes*, *K-Nearest Neighbour* e *Logistic Regression*. Diferente do artigo desenvolvido por (XUAN *et al.*, 2018), neste trabalho foram utilizadas diferentes métricas de avaliação da qualidade dos modelos já levando em consideração o problema causado pela falta de amostras para a classe associada a fraude. Indicadores como MCC, precisão, sensitividade e especificidade foram utilizadas de forma a comparar os resultados obtidos pelos modelos usados no estudo. Como resultado tem-se que o modelo KNN foi o que obteve os melhores resultados , à excessão da métrica acurácia obtida em um dos experimentos.

Por último, tem-se o estudo comparativo feito no trabalho (ESENOGHO *et al.*, 2022) onde são analisados uma lista de modelos de classificação, dentre eles MLP e *Decision Tree*. Neste estudo também é possível notar o uso de uma variação da técnica SMOTE com o objetivo de analisar o ganho, ou perda, do uso *data augmentation* durante o processo de treino dos modelos.

Dessa forma, tem-se que a escolha dos modelos utilizados durante o desenvolvimendo deste trabalho foi baseada em suas presenças em diferentes artigos na literatura dentro do contexto de detecção de fraudes em cartões de crétido.

3 METODOLOGIA

Nesta seção será descrito o processo metodológico adotado durante o desenvolvimento deste projeto que tem como principal objetivo avaliar o desempenho de diferentes modelos de *machine learning* aplicados ao contexto de detecção de fraudes de forma a determinar a técnica mais eficaz à luz de métricas selecionadas. Tal metodologia pode ser visualizada na figura 11 e seus passos serão descritos de forma detalhada nas subseções subsequentes.

3.1 Ferramentas computacionais

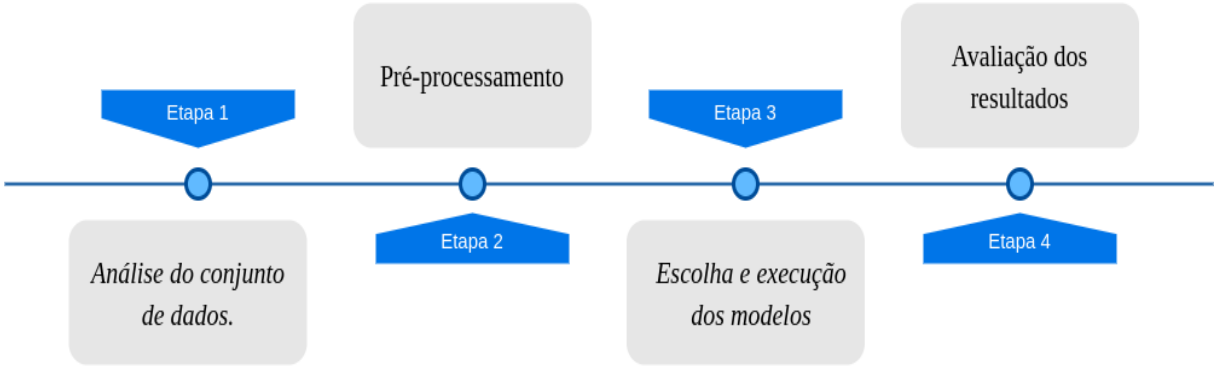
Todo o processo técnico de análise, processamento, desenvolvimento e validação dos modelos e do conjunto de dados utilizados neste projeto foram feitos utilizando o ambiente descrito abaixo junto de sua respectiva configuração. A fim de garantir a reprodutibilidade dos resultados, foi utilizado durante o desenvolvimento deste projeto um valor de estado fixo para a variável *random_state* presente nas bibliotecas de *machine learning* utilizadas.

- Processador Intel® Core™ i5-8250U CPU @ 1.60GHz × 8;
- Memória 12GB DDR4;
- SSD 240 GB WD Green PC SN350;
- Sistema operacional Ubuntu 20.04.1 LTS;
- Linguagem de programação: Python 3.8.10;
- Pacotes utilizados:
  - jupyter-notebook: Ferramenta que possibilita executar blocos de código *python*, dentre outras linguagens, de forma interativa através de uma interface baseada em tecnologias web (KLUYVER *et al.*, 2016);
  - scikit-learn: Módulo *python* com uma grande variedade de algoritmos de aprendizado de máquina, sejam eles supervisionado ou não (PEDREGOSA *et al.*, 2011);
  - imblearn: Biblioteca desenvolvida para lidar com problemas de conjunto de dados desbalanceados (LEMAîTRE *et al.*, 2017);
  - numpy: Biblioteca de código aberto que possui inúmeras funcionalidades que podem ser aplicadas a problemas de computação numérica (HARRIS *et al.*, 2020);
  - pandas: Ferramenta de manipulação e análise de dados escrita em *python*, muito utiliza em contextos de ciência de dados (TEAM, 2020; MCKINNEY, 2010);
  - matplotlib: Módulo *python* utilizado para gerar imagens e visualizações interati-

vas (HUNTER, 2007);

– seaborn: Biblioteca de visualização de dados baseada na *matplotlib* (WASKOM, 2021).

Figura 11 – Framework utilizando no desenvolvimento do trabalho.



Fonte: Elaborada pelo autor (2022).

3.2 Análise do conjunto de dados

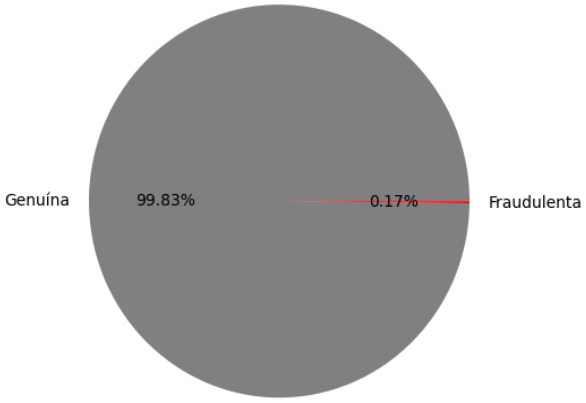
O conjunto de dados utilizado neste trabalho foi o *dataset* intitulado "Credit Card Fraud Detection" disponibilizado de forma pública pela ULB (Université Libre de Bruxelles) através do site *Kaggle* [1] (ULB, 2018). Tal conjunto de dados possui 284,807 transações reais feitas por portadores europeus dentro do período de dois dias em setembro de 2013, sendo apenas 492 consideradas fraude. Na figura 12 é possível visualizar de forma gráfica o percentual de transações fraudulentas e genuínas no conjunto de dados utilizado. Tal *dataset* possui 31 colunas, sendo 28 delas idenfiticadas como V_i, onde i varia de 1 à 28 e seus valores são resultado da execução de um método de redução de dimensionalidade conhecido como *Principal Component Analysis* (PCA) capaz de transformar as variáveis iniciais mantendo apenas as que possuem maior quantidade de informação.

Por questões de confidenciabilidade, os autores do conjunto de dados não foram capazes de revelar maiores informações sobre os valores iniciais dessas colunas antes do processo feito pelo *PCA*, assim como o real significado das mesmas, de forma que este projeto foi desenvolvido utilizando os valores já manipulados pela técnica de redução de dimensionalidade (ULB, 2018).

As outras colunas são nomeadas como *Class, Amount* e *Time*, sendo a coluna *Class* responsável por identificar se tal transação foi classificada como fraude ou não, tendo 1 associado

[1] https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud

Figura 12 – Percentual de transações no conjunto de dados por tipo.



Fonte: Elaborada pelo autor (2023).

a transações fraudulentas e 0 a transações válidas. A coluna *Amount* indica o valor gasto na transação e a coluna *Time* indica o tempo, em segundos, decorrido desde a primeira transação registrada.

3.3 Pré-processamento

Esta etapa foi desenvolvido através da utilização de funções implementadas em *python* presentes nas bibliotecas *pandas* e *numpy*, utilizadas para análise de dados e computação numérica.

Inicialmente foi verificada a possibilidade das colunas *Amount* e *Time* que não tiveram seus valores modificados pelo algoritmo de redução de dimensionalidade, PCA, possuirém valos nulos ou faltantes, pois esse tipo de incosistência pode atrapalhar diretamente a perfomance e a execução dos algoritmos de classificação. Após análise, foi constatado que todos os 284,807 registros contidos no conjunto de dados possuiam dados considerados válidos , não-nulos e não-vazios, de forma que não foi necessário utilizar técnicas de substituição ou remoção dos registros inválidos. O resultado dessa análise pode ser vista na Figura 13, onde é possível verificar que para essas colunas o total de linhas não-nulas é igual ao número total de amostras no conjunto de dados, 284,807.

A segunda e última etapa realizada foi padraonizar os valores das colunas *Amount* e *Time*, pois como é possível verificar na Tabela 1, tais colunas possuem valores muito destoantes dos valores das outras colunas. Essa discrepância nos intervalos de valores pode afetar o desempenho dos modelos, dado que as colunas com maiores valores podem acabar dominando as outras colunas gerando um viés na análise de importância das características (SINGH; SINGH,

Figura 13 – Resultado da busca por valores nulos nas colunas *Amount* e *Time*.

```
RangeIndex: 284807 entries, 0 to 284806
Data columns (total 2 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   Amount  284807 non-null  float64
 1   Time    284807 non-null  float64
dtypes: float64(2)
memory usage: 4.3 MB
```

Fonte: Elaborada pelo autor (2023).

2020). Tomando isso em consideração , a equação 3.3, utilizada pelo objeto *StandardScaler* presente na biblioteca *scikit-learn*, foi aplicada nos valores das colunas citadas fazendo com que os valores passassem a se enquadrar no range de -1 à 1. O valor Z_i representa o novo valor da i amostra, x_i o valor real da observação, μ representa a média dos valores da coluna tratada , σ o desvio padrão e n o número total de amostras.

$$Z_i = \frac{x_i - \mu}{\sigma} \tag{3.1}$$

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{3.2}$$

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2} \tag{3.3}$$

O resultado dessa normalização pode ser encontrado na Tabela 2. Nela é possível visualizar que os valores das novas colunas passaram a ser valores mais próximos aos valores das colunas que foram processadas previamente pelo PCA.

3.4 Execução dos modelos

Foram realizados dois experimêntos, ambos usando validação cruzada *k-fold* estratificada com k igual a 5, a fim de comparar tanto os desempenhos dos modelos entre si, quanto com e sem o uso de data augmentation. Como implementação da técnica de aumento de amostras *Synthetic Minority Over-sampling Technique (SMOTE)* (CHAWLA *et al.*, 2002) foi utilizado o método *fit_resample* presente na biblioteca *imblearn* escrita em python. Já a implementação da técnica de validação cruzada *k-fold* estratificada, assim como todos os modelos de classificação utilizados neste trabalho, está disponível na biblioteca *scikit-learn* também escrita e disponibilizada na linguagem *python*.

Todos os algoritmos de classificação utilizados nos testes experimentais estão listados na Tabela 3 assim como seus possíveis hiperparâmetros. Com o intuito de simplificar os testes, em ambos os expêrimentos os valores *default* dos hiperparâmetros foram utilizados, sendo assim

Tabela 1 – Estado atual do conjunto de dados após a filtragem de colunas.

	média	desvio padrão	min	max
V1	1.1683e-15	1.9586	-56.4075	2.4549
V2	3.4169e-16	1.6513	-72.7157	22.0577
V3	-1.3795e-15	1.5162	-48.3256	9.3825
V4	2.0741e-15	1.4158	-5.6832	16.8753
V5	9.6040e-16	1.3802	-113.7433	34.8016
V6	1.4873e-15	1.3322	-26.1605	73.3016
V7	-5.5564e-16	1.2370	-43.5572	120.5895
V8	1.2135e-16	1.1943	-73.2167	20.0072
V9	-2.4063e-15	1.0986	-13.4340	15.5950
V10	2.2390e-15	1.0888	-24.5882	23.7451
V11	1.6733e-15	1.0207	-4.7975	12.0189
V12	-1.2470e-15	0.9992	-18.6837	7.8484
V13	8.1900e-16	0.9952	-5.7919	7.1268
V14	1.2073e-15	0.9586	-19.2143	10.5267
V15	4.8874e-15	0.9153	-4.4989	8.8777
V16	1.4377e-15	0.8762	-14.1298	17.3151
V17	-3.7722e-16	0.8493	-25.1627	9.2535
V18	9.5641e-16	0.8381	-9.4987	5.0410
V19	1.0399e-15	0.8140	-7.2135	5.5919
V20	6.4062e-16	0.7709	-54.4977	39.4209
V21	1.6540e-16	0.7345	-34.8304	27.2028
V22	-3.5686e-16	0.7257	-10.9331	10.5030
V23	2.5786e-16	0.6244	-44.8077	22.5284
V24	4.4732e-15	0.6056	-2.8366	4.5845
V25	5.3409e-16	0.5212	-10.2954	7.5195
V26	1.6834e-15	0.4822	-2.6045	3.5173
V27	-3.6601e-16	0.4036	-22.5657	31.6121
V28	-1.2274e-16	0.3301	-15.4301	33.8478
Time	9.4814e+04	47488.14	0	172792
Amount	8.8345e+01	250.12	0	25691.16

Fonte: elaborado pelo autor (2023).

Tabela 2 – Colunas obtidas como resultado do processo de normalização.

	média	desvio padrão	min	max
ScaledAmount	2.91e-17	1.00	-0.35	102.36
ScaledTime	-3.06e-16	1.00	-1.99	1.64

Fonte: elaborado pelo autor (2023).

nenhuma técnica de otimização de hiperparamêtros foi utilizada, à exemplo da busca em grade que também tem sua implementação disponível no pacote *scikit-learn*. Alguns dos valores *default* utilizados durante a execução dos modelos foram:

- *n_estimators* igual a 100. Representa o número de arvores de decisão utilizadas para compor uma *Random Forest* (PEDREGOSA *et al.*, 2011);
- *criterion* igual a *gini*. Refere-se a uma medida usada para avaliar a qualidade de uma divisão em um determinado nó da árvore de decisão (PEDREGOSA *et al.*, 2011);
- *max_depth* igual a *none*. Equivale a profundidade máxima da árvore de decisão, limitando

o número de decisões tomadas antes de se chegar às folhas. Quando o valor é *none* a árvore continuará se expandido ate que todas as amostras em suas folhas pertençam a mesma classe ou que todas folhas contenham menos amostras do que o valor mínimo definido por *min_samples_split* (PEDREGOSA *et al.*, 2011);

- *penalty* igual a *l2*. Refere-se a uma penalidade adicionada à função de custo usada durante o treinamento do modelo *Logistic Regression*. Essa penalidade é usada para evitar com que o modelo se ajuste demais as amostras de treino (*overfitting*) (PEDREGOSA *et al.*, 2011);
- *activation* igual a *relu*. Representa a função de ativação utilizada na saída da camada de uma rede neural (PEDREGOSA *et al.*, 2011);
- *learning_rate* igual a *constant*. Representa um valor de constante usado no algoritmo gradiente descendente estocástico. Tal valor afeta a forma em que o algoritmo converge (PEDREGOSA *et al.*, 2011);
- *max_iter* igual a 200. Caso o valor do hiperparametro *solver* esteja associado a alguma otimizador baseado em gradiente estocástico, esse valor representa o número de épocas, ou seja, o número de vezes que os dados serão apresentados à rede neural (PEDREGOSA *et al.*, 2011);
- *solver* igual a *adam*. Para o modelo MLP, esse hiperparametro refere-se ao método de otimização de pesos da rede neural baseado em gradiente estocástico (PEDREGOSA *et al.*, 2011);
- *tol* igual a $1e-4$. Determina a tolerância ou a precisão desejada para a solução encontrada pelo algoritmo de otimização usado na regressão logística (PEDREGOSA *et al.*, 2011).

Tabela 3 – Lista de hiperparamêtros por modelo.

Modelo	Hiperparamêtro
Random Forest	n_estimators, criterion, max_depth, min_samples_split, min_samples_leaf, max_features, etc.
Logistic Regression	penalty, C, tol, max_iter, solver, etc.
Naive Bayes	Não possui hiperparâmetros significativos na implementação do scikit-learn para o Naive Bayes Gaussiano.
Decision Tree	criterion, max_depth, min_samples_split, min_samples_leaf, etc.
Multi Layer Perceptron	hidden_layer_sizes, activation, learning_rate, alpha, tol, max_iter, solver, etc.

Fonte: elaborado pelo autor (2023).

3.5 Avaliação dos resultados

Dada a existência do problema de desbalanceamento de classes presente no conjunto de dados, além do uso da acurácia como métrica de avaliação de desempenho de um modelo foram utilizadas também as métricas presentes na Tabela 4 que podem ser obtidas através da análise da matriz de confusão. Sabendo disso, durante o processo de análise de desempenho foi dada uma menor ênfase à acurácia em comparação com outras métricas.

Tabela 4 – Métricas utilizadas para avaliar o desempenho dos modelos.

Métrica	Valor mínimo	Valor máximo	Importância
Acurácia	0	1	Baixa
$F1_{score}$	0	1	Alta
Coeficiente de correlação de Matthews (MCC)	-1	1	Alta
Área sob a curva de Precisão-Recall (AUC-PR)	0	1	Alta

Fonte: elaborado pelo autor (2023).

Devido a utilização de validação cruzada na execução dos modelos, foram usadas as medidas estatísticas média e desvio padrão para analisar os resultados obtidos em cada iteração do processo. A média foi calculada para obter a medida central dos resultados, fornecendo assim uma estimativa geral do desempenho do modelo. O desvio padrão, por sua vez, foi calculado para avaliar a dispersão dos resultados em relação à média, ajudando a entender a variabilidade dos resultados obtidos em cada iteração.

Medidas como o tempo de treino e o tempo gasto para classificar um novo conjunto de amostras também foram levadas em consideração pois podem infuenciar diretamente na escolha de um modelo dado que tais indicadores servem como paramêtro para análise da escalabilidade do modelo em ambientes produtivos que apresentam restrições temporais como é o caso de serviços de detecção de fraude em tempo real.

Por último também foram analisadas algumas das matrizes de confusão geradas durante o processo de validação cruzada, de forma a avaliar dentro do contexto de detecção de fraudes o impacto de falsos positivos, quando a aplicaçao responsável identifica erroneamente uma transação ou atividade como sendo fraudulenta, quando na verdade é legítima, e falsos negativos, quando o sistema falha em detectar a ocorrência de uma fraude real.

4 RESULTADOS

Este trabalho foi desenvolvido com o intuito de comparar diferentes modelos de aprendizado de máquina aplicados ao problema de detectar transações fraudulentas de forma automatizada. Essa problemática é um exemplo de classificação binária onde os dados são em sua maioria desbalanceados, dado que a quantidade de transações verdadeiras é muito maior do que as consideradas fraudulentas, fazendo com que diferentes métricas alem da acurácia fossem utilizadas como critério de avaliação da qualidade do modelo.

Como forma de amenizar também o problema de desbalanceamento, foram realizados 2 experimentos diferentes onde o primeiro se utilizou de uma técnica conhecida como data augmentation, método esse utilizado para equilibrar a quantidade de dados das diferentes classes, e outro experimento usando o conjunto de dados sem uso da técnica de aumento de amostras a fim de comparar os resultados obtidos pelos modelos desenvolvidos. Por último, também foi utilizado o método de validação cruzada com k-partições em sua versão onde as partições são estratificadas (*Stratified K-Fold*), assim cada conjunto possui aproximadamente a mesma quantidade de amostras de cada classe.

De acordo com as informações previamente debatidas na seção metodológica, os modelos empregados neste estudo compreenderam *Random Forest*, *Logistic Regression*, *Naive Bayes*, *Decision Tree* e *Multi Layer Perceptron (MLP)*. Quanto às métricas utilizadas para avaliar o desempenho, incluíram acurácia, coeficiente de correlação de Mathews (MCC), $F1_{Score}$ e a área sob a curva de Precisão-Sensibilidade (AUC-PR).

4.1 Resultados do Experimento A : Sem data augmentation

Tabela 5 – Resultados obtidos sem aplicação de data augmentation.

Métrica	Random Forest	Logistic Regression	Naive Bayes	Decision Tree	MLP
Acurácia (μ)	0,9995	0,9992	0,9779	0,9992	0,9994
Acurácia (σ)	0,0000	0,0001	0,0008	0,0000	0,0000
F1-Score (μ)	0,8537	0,7320	0,1150	0,7652	0,8408
F1-Score (σ)	0,0157	0,0299	0,0058	0,0138	0,0223
MCC (μ)	0,8578	0,7430	0,2228	0,7648	0,8425
MCC (σ)	0,0165	0,0282	0,0083	0,0138	0,0215
AUC-PR (μ)	0,8467	0,7599	0,0854	0,5862	0,8393
AUC-PR (σ)	0,0269	0,0204	0,0057	0,0211	0,0263

Fonte: elaborado pelo autor (2023).

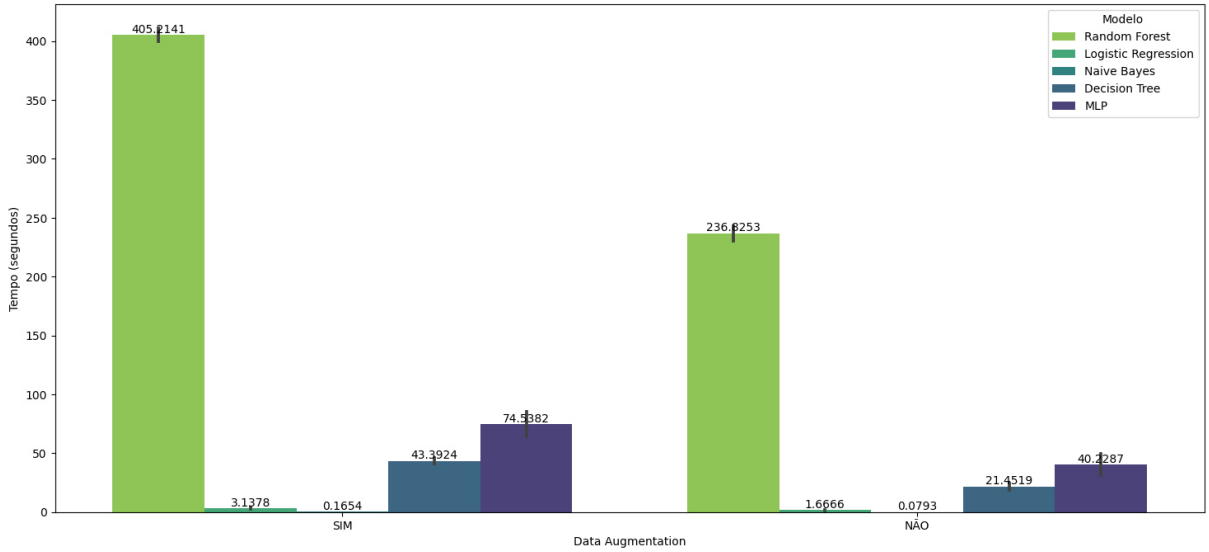
É possível verificar na Tabela 5 que o algoritmo *Random Forest* foi o modelo que

obteve melhor desempenho pois possui 3 das 4 melhores médias dentre as métricas utilizadas para avaliação de qualidade do modelo. Em contra partida, é possível constatar que o modelo *Naive Bayes* foi o modelo que registrou os piores resultados médios dentre os outros algoritmos, possuindo quase todo os seus valores abaixo de 0,5.

Já com relação a velocidade de classificação e treino, é possível visualizar na Figura 14 tanto o tempo médio gasto durante os processos de treino dos modelos quanto o tempo médio que cada técnica levou para classificar 10 novas amostras escolhidas de forma aleatória dentre o conjunto de teste, ou seja, amostras que não foram utilizadas durante o processo de treino. Temos então que apesar de ter tido o melhor desempenho dadas as métricas utilizadas para avaliação, o algoritmo de *Random Forest* foi o modelo que apresentou os piores resultados temporais, levando aproximadamente 28 vezes mais tempo para classificar as novas amostras quando comparado ao modelo mais rápido, *logistic regression*.

Por outro lado, temos os modelos MLP e *Logistic Regression* que apresentaram tanto bons valores para as métricas análisadas, com enfãse no modelo MLP que possui os valores muito próximos dos resultados do *Random Forest*, quanto o tempo médio durante o processo de treino e classificação.

Figura 14 – Comparação entre os tempos de treino dos diferentes modelos com e sem o uso de data augmentation.



Fonte: Elaborada pelo autor (2023).

Tabela 6 – Resultados obtidos com aplicação de data augmentation.

Métrica	Random Forest	Logistic Regression	Naive Bayes	Decision Tree	MLP
Acurácia (μ)	0,9995	0,9747	0,9757	0,9975	0,9992
Acurácia (σ)	0,0001	0,0014	0,0013	0,0001	0,0001
F1-Score (μ)	0,8565	0,1111	0,1093	0,5233	0,7833
F1-Score (σ)	0,0276	0,0063	0,0047	0,0186	0,0312
MCC (μ)	0,8571	0,2284	0,2203	0,5554	0,7831
MCC (σ)	0,0276	0,0083	0,0057	0,0182	0,0312
AUC-PR (μ)	0,8472	0,728677	0,0879	0,3103	0,8090
AUC-PR (σ)	0,0272	0,0293	0,0057	0,0206	0,0298

Fonte: elaborado pelo autor (2023).

4.2 Resultados do Experimento B : Com data augmentation

Com base nos valores apresentados na Tabela 6, é possível observar que a técnica denominada “Random Forest” foi aquela que demonstrou os melhores resultados médios em todas as métricas, possuindo todos os valores acima de 0,7. Todos os outros modelos avaliados no experimento obtiveram valores considerados ruins, à exceção do modelo MLP, pois apresentaram problemas abaixo de 0,7 para todas as métricas analisadas.

Com relação ao tempo gasto no treino dos modelos, é possível verificar através do gráfico presente na Figura 14 que o uso de data augmentation teve um impacto significativo no tempo de treinamento dos modelos *Random Forest* e *Multi Layer Perceptron (MLP)*. Ambos os algoritmos experimentaram um aumento substancial no tempo necessário para concluir o processo de treinamento. Essa mudança indica que a inserção de data augmentation trouxe um ônus adicional em termos de eficiência computacional para esses modelos, ônus esse não justificado dado que não houve aumento significativo no desempenho dos modelos como é possível verificar na Tabela 6.

4.3 Análise dos resultados

Ao comparar os resultados obtidos nos dois experimentos apresentados nas seções anteriores, é possível definir que o modelo *Random Forest* pode ser considerado o melhor modelo dentre os avaliados pois os resultados alcançados por tal modelo, representados pelas métricas de desempenho acurácia, coeficiente de correlação de Mathews (MCC), $F1_{Score}$ e a área sob a curva de Precisão-Sensibilidade (AUC-PR) demonstram um desempenho consistente e superior em comparação com os outros modelos avaliados. Também é possível visualizar que o modelo *Naive Bayes* foi o algoritmo que obteve os piores valores para algumas das médias apresentando

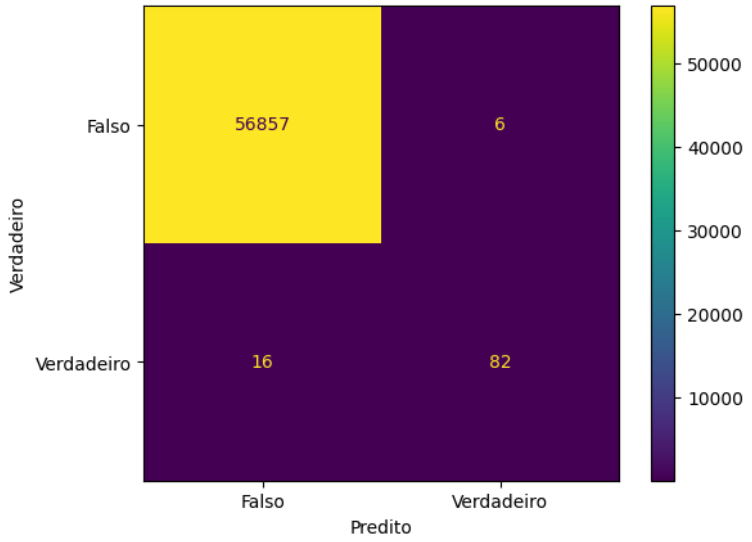
muitas vezes, com ou sem o uso de geração de amostras sintéticas, valores abaixo de 0,3.

Na Figura 15 é possível visualizar uma das matrizes de confusão geradas durante o processo de validação cruzada para o modelo *Random Forest*. Através dela é possível calcular as métricas precisão, sensibilidade e $F1_{Score}$ e chegar a algumas conclusões:

- O modelo apresentou o valor 0,8367 para a métrica sensibilidade, que indica, dentro do contexo apresentado de detecção de fraudes, a capacidade do modelo em identificar corretamente as instâncias de fraude. Uma alta sensibilidade significa que o modelo consegue detectar a maioria das transações fraudulentas, reduzindo o número de falsos negativos (casos em que uma transação fraudulenta é considerada uma transição legítima);
- Para a métrica precisão o modelo apresentou o valor 0,9318, métrica essa que indica a proporção de transações que foram classificadas como fraude corretamente, ou seja, que são realmente fraudes. Uma alta precisão é importante para evitar falsos positivos (casos em que uma transação é classificada como fraude erroneamente), evitando o transtorno causado aos clientes legítimos, reduzindo o tempo e os recursos gastos na investigação de falsos positivos e mantendo a confiabilidade no sistema de detecção;
- Para a métrica $F1_{Score}$, que combina através da média harmonica o valor obtido para a sensibilidade e o valor da precisão do modelo, o modelo apresentou o valor 0,8817, valor esse que tem seu valor máximo 1, e que representa o equilibrio entre a capacidade do modelo de identificar corretamente as instâncias de fraude (sensibilidade) e sua precisão em classificar corretamente as transações fraudulentas;
- Ja para o indicador MCC temos o valor 0,8828, valor esse muito próximo do ideal, 1, o que indica que o modelo é capaz de diferenciar muito bem tanto uma transação fraudulenta quanto uma genuína;

Já na Figura 16 é possível verificar uma das matrizes de confusão do modelo *Naive Bayes* considerado como o pior modelo dentre os analisados. Apesar de apresentar um valor valor alto, 0,8367, para a métrica sensibilidae que é a responsável por indicar a capacidade do modelo de classificar corretamente as transações fraudulentas, tal modelo apresenta um valor baixíssimo, 0,0576, de precisão, ou seja, o algoritmo classifica incorretamente como fraude um número elevado de transações legítimas que em um cenário real de uso causaria desgaste para os clientes portadores dos cartões que teriam suas compras negadas, redução na confiabilidade do sistema dada sua imprecisão e aumento do número de recursos utilizados para lidar com a falsa fraude.

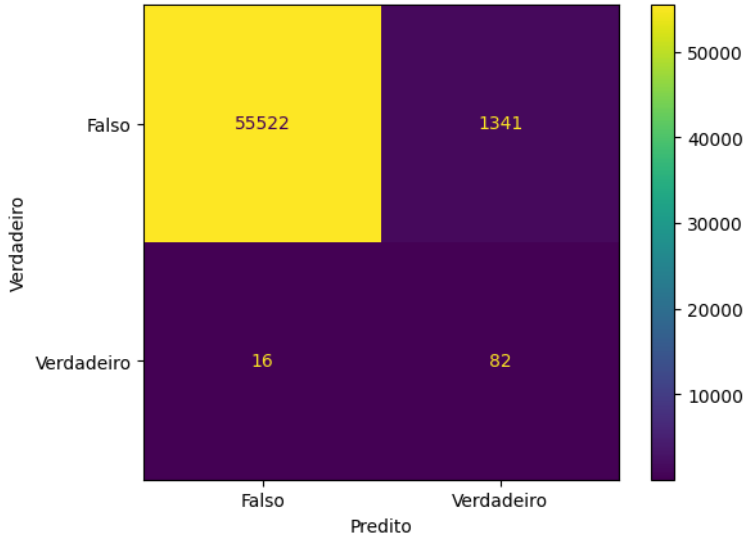
Figura 15 – Matriz de confusão do modelo Random Forest gerada durante o experimento B.



Fonte: Elaborada pelo autor (2023).

O mesmo modelo também apresentou valores médios considerados ruins para as métricas MCC, 0,2159 com o uso de data augmentation, e AUC-PR, 0,0879 também com o uso de data augmentation, pois são muito próximos de 0. Para o coeficiente de correlação de Matthews (MCC) isso siginifica que o modelo possui um comportamento similar ao de um algoritmo de comportamento aleatório.

Figura 16 – Matriz de confusão do modelo Naive Bayes Forest gerada durante o experimento B.

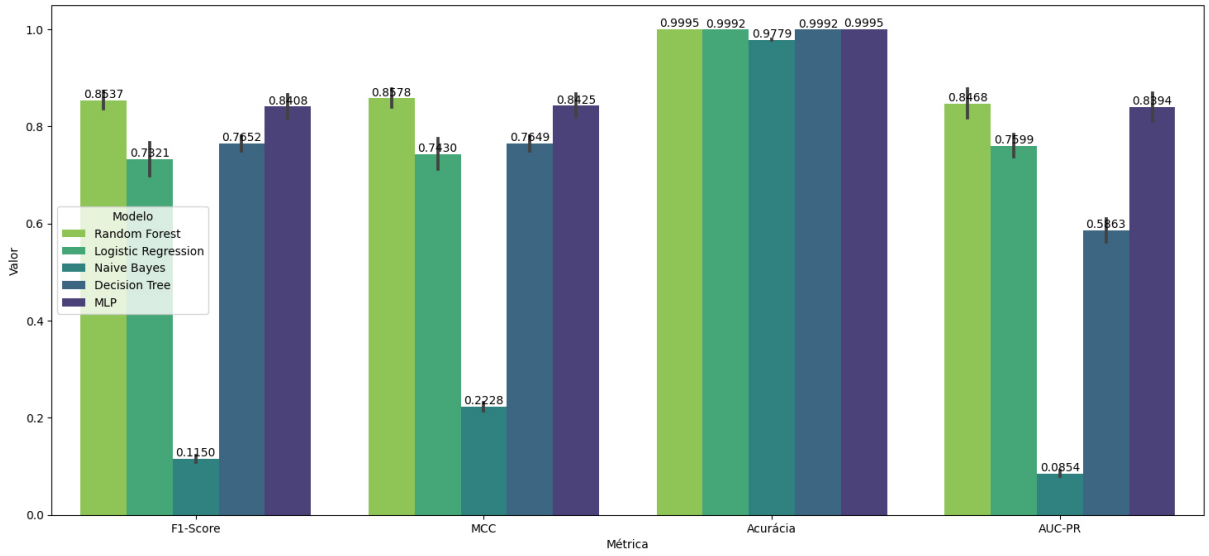


Fonte: Elaborada pelo autor (2023).

Em resumo, ao comparar os resultados obtidos durante os experimentos realizados com diferentes algoritmos foi possível classificar o modelo *Random Forest* como a técnica que obteve o melhor desempenho, com ou sem o uso da técnica de data augmentation, como pode

ser visto na Figura 17. Na realidade, a utilização da técnica resultou em uma deterioração do tempo de treino dos modelos, fazendo com que, por exemplo, o tempo médio gasto para treinar o modelo *Random Forest* fosse quase duplicado. Também não houve um ganho substancial no desempenho dos modelos, pelo contrário, houve redução nos valores obtidos por todos os modelos como é possível visualizar na figura A que mostra os resultados dos modelos por métrica com e seu o uso de data augmentation, assim como o percentual de diferença entre tais valores.

Figura 17 – Comparação entre as métricas obtidas após a execução de todos os modelos sem o uso de data augmentation.



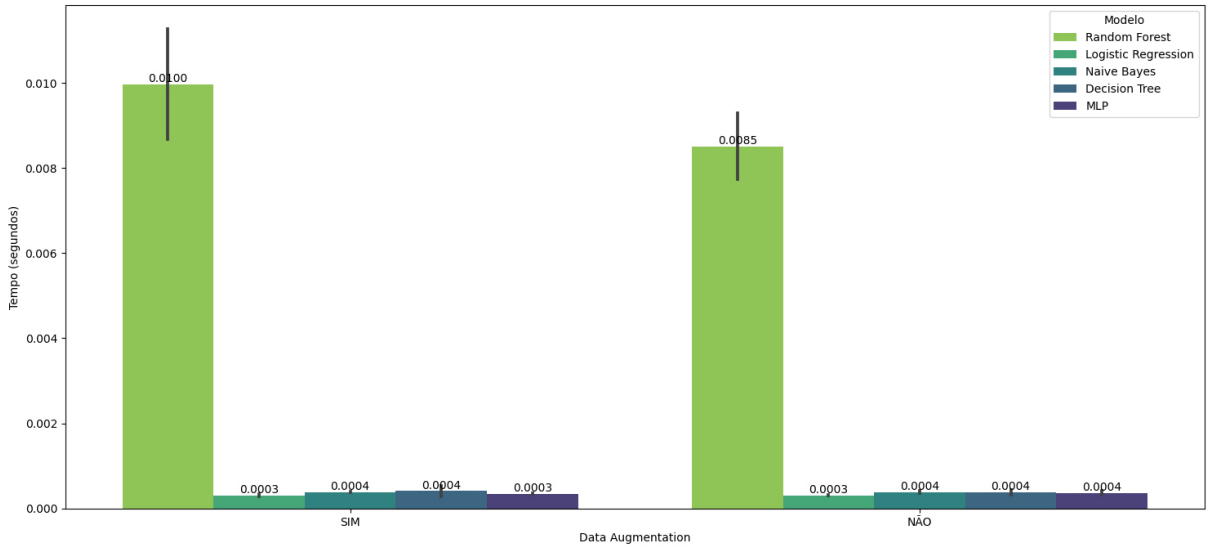
Fonte: Elaborada pelo autor (2023).

Em última análise, embora a técnica *Random Forest* tenha se destacado em termos de qualidade de previsão, a mesma foi considerada como a mais lenta para classificar uma nova amostra, levando quase meio segundo para finalizar o processo de acordo com o tempo médio registrado durante os experimentos. Esse é um ponto muito importante e que deve ser levado em consideração em cenários reais onde milhares de clientes realizam transações de forma simultânea.

Levando a problemática de tempo em consideração, embora o modelo *Random Forest* tenha obtido os melhores valores para as métricas de desempenho utilizadas, o modelo *MLP* foi considerado como o melhor algoritmo em termos gerais por ter obtido valores muito próximos dos resultados do Random Forest, com ou sem uso de data augmentation, com a vantagem de também ter sido o mais rápido tanto durante o processo de treino, importante caso essa etapa precise ser refeita devido ao acréscimo de novas caracerísticas, por exemplo, quanto durante o procedimento de classificação de novas amostras, sendo aproximadamente 21 vezes

mais rapido como é possível visualizar na Figura 18.

Figura 18 – Comparação entre os tempos de classificação dos diferentes modelos com e sem o uso de data augmentation.



Fonte: Elaborada pelo autor (2023).

5 CONCLUSÕES E TRABALHOS FUTUROS

A detecção de fraudes em cartões de crédito é um desafio crítico na indústria de meios de pagamento, pois as perdas decorrentes de transações fraudulentas podem ser significativas tanto para o emissor do cartão que precisa arcar com as despesas operacionais e financeiras quanto para o portador do cartão. Neste trabalho, o objetivo foi investigar, treinar e avaliar diferentes modelos de aprendizado de máquina através de diferentes métricas quando aplicados a problemática de detecção de fraudes em questão.

Inicialmente, foi realizada uma análise do problema levando em consideração seus impactos tanto do ponto de vista do emissor, aquele que emite e gerencia o cartão, quanto do portador. Para investigação e desenvolvimento do projeto foi escolhido um conjunto de dados na plataforma de ciência de dados *Kaggle* que agrupa dados de transações reais feitas durante 2 dias em setembro de 2013 por portadores europeus. O *dataset* possui 284.807 transações, sendo apenas 0,172% classificadas como fraude, o que torna a base de dados em questão altamente desbalanceada. Outra característica da base de dados é que por questões de confidenciabilidade, o conjunto de dados possui apenas valores numéricos que foram transformados utilizado o algoritmo *Principal Component Analsys (PCA)* e são nomeados como V_1, V_2 ... V_{28}, exceto pelas *features Time* e *Amount* que representam o tempo corrido em segundos desde a primeira transação registrada e o valor gasto em cada transação, respectivamente.

Durante a etapa de pré-processamento foi realizado um processo de padronização dos dados das colunas que não foram modificadas pelo algoritmo *Principal Component Analsys (PCA)* com intuito de reduzir a discrepância nos intervos dos valores quando comparado as outras colunas, discrepância essa que poderia gerar viés na análise da importância das *features* durante a execução dos modelos.

Em seguida, foram exploradas várias técnicas de aprendizado de máquina, incluindo algoritmos de classificação como *Random Forest*, *Logistic Regression*, *Naive Bayes*, *Decision Tree* e *Multi Layer Perceptron (MLP)*. Afim de investigar também o impacto do desbalanceamento do conjunto de dados foi utilizado uma técnica conhecida como *Synthetic Minority Over-sampling Technique (SMOTE)* capaz de gerar dados sintéticos para a classe minoritária, nesse caso a classe que representa fraude, fazendo assim com que os modelos tenham contato com um maior número de dados das duas classes.

Por fim, os desempenhos dos modelos foram avaliados utilizando as métricas: acurácia, coeficiente de correlação de Mathews (MCC), $F1_{Score}$, área sob a curva de Precisão-

Sensibilidade (AUC-PR), tempo de treino e tempo de classificação. Exceto para as métricas de tempo citadas, a técnica conhecida como *Random Forest* foi a que obteve melhor desempenho possuindo os melhores valores médios, seguido pelo modelo *MLP*. O modelo obteve um valor médio de 0,846 para a área abaixo da curva precisão-sensibilidade (AUC-PR), 0,857 para o coeficiente de correlação de Matthews (MCC) e 0,854 aproximadamente para o F1-score. Apesar dos bons resultados de desempenho, o modelo também foi classificado como o algoritmo mais lento tanto para treinar quanto para avaliar novas transações, levando em média mais de 0,01 segundo para executar a classificação de 10 novas amostras, fazendo assim como que fosse cerca de 21 vezes mais lento que o *MLP*.

Ao levar em consideração essa diferença no tempo levado para classificar novas amostras, decidiu-se que o modelo *MLP* foi o que se saiu melhor pois , além de apresentar valores muito próximos dos resultados obtidos pelo *Random Forest*, diferença de 1,54% no valor do $F1_{Score}$ média, 1,82% para o MCC médio e 0,88% de diferença para o valor médio da AUC-PR também foi um dos algoritmos mais rápidos, característica essa muito importante quando se pensa em escalabilidade de um sistema que deve atender milhares de requisições de forma simultânea.

Por fim, com o desenvolvimento do projeto também foi possível observar que a detecção de fraudes em cartões de crédito é um desafio em constante evolução, uma vez que os responsáveis pelas fraudes estão sempre buscando novas maneiras de contornar os sistemas de segurança. Portanto, é importante que os esforços de pesquisa e desenvolvimento nessa área continuem, a fim de aprimorar os modelos existentes e explorar novas técnicas e abordagens.

Em resumo, este estudo contribuiu para o avanço da detecção de fraudes em cartões de crédito e foi capaz de demonstrar a eficácia das técnicas de aprendizado de máquina na identificação de transações maliciosas. Espera-se que os resultados e as metodologias apresentados neste trabalho possam ser aplicados na indústria financeira, auxiliando na proteção dos clientes e no combate às atividades fraudulentas em transações de cartões de crédito, reduzindo custos financeiros e operacionais, assim como aumentando a segurança em transações que se utilizam do meio de pagamento tratado.

Como sugestão de trabalhos futuros tem-se :

- Treinamento e avaliação de novos modelos de aprendizado de máquina;
- Utilização de busca em grade com intuito de encontrar a combinação ideal de hiperparâmetros para os modelos de aprendizado de máquina escolhidos, pois durante o desenvol-

vimento deste trabalho foram utilizados apenas os valores *default* disponibilizados pela biblioteca *sckit-learn*;

- Utilização de novas métricas de qualidade do modelo e possibilidade do uso do teste de Wilcoxon como parâmetro na comparação entre os resultados dos modelos utilizados;
- Desenvolvimento de um serviço web capaz de receber requisições contento dados de uma nova transação e classificá-las como fraude ou não utilizado o modelo de *machine learning* considerado como o melhor.

REFERÊNCIAS

AWOYEMI, J. O.; ADETUNMBI, A. O.; OLUWADARE, S. A. Credit card fraud detection using machine learning techniques: A comparative analysis. In: **2017 International Conference on Computing Networking and Informatics (ICCNI)**. [S.l.: s.n.], 2017. p. 1–9.

Banco Central do Brasil. **"Estatísticas de Meios de Pagamentos"**. 2023. Disponível em: <https://www.bcb.gov.br/estatisticas/spbadendos>. Acesso em: 18 Maio 2023.

BEHERA, T. K.; PANIGRAHI, S. **Credit Card Fraud Detection: A Hybrid Approach Using Fuzzy Clustering Neural Network**. [S.l.: s.n.], 2015. 494-499 p.

BISHOP, C. M. New York : Springer, [2006] ©2006, 2006. Disponível em: <https://search.library.wisc.edu/catalog/9910032530902121>.

BREIMAN, L. Random forests. **Machine Learning**, Springer, v. 45, n. 1, p. 5–32, 2001.

Chauncey Crail, Dia Adams. **"6 Credit Card Scams And How To Avoid Them."**. 2023. Disponível em: <https://www.forbes.com/advisor/credit-cards/credit-card-scams-and-how-to-avoid-them/>. Acesso em: 18 Julho 2023.

CHAWLA, N. V.; BOWYER, K. W.; HALL, L. O.; KEGELMEYER, W. P. Smote: Synthetic minority over-sampling technique. **Journal of Artificial Intelligence Research**, v. 16, p. 321–357, 2002.

Chen, James and J.Catalano, Thomas. **"Address Verification Service (AVS): Definition, Uses, and Example"**. 2023. Disponível em: <https://www.investopedia.com/terms/a/address-verification-system.asp>. Acesso em: 20 Junho 2023.

CHICCO, D.; JURMAN, G. The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation. **BMC Genomics**, v. 21, n. 1, p. 6, Jan 2020. ISSN 1471-2164. Disponível em: <https://doi.org/10.1186/s12864-019-6413-7>.

DAVIS, J.; GOADRICH, M. The relationship between precision-recall and roc curves. Association for Computing Machinery, New York, NY, USA, p. 233–240, 2006. Disponível em: <https://doi.org/10.1145/1143844.1143874>.

DESHPANDE, M.; RAO, V. Depression detection using emotion artificial intelligence. In: **2017 International Conference on Intelligent Sustainable Systems (ICISS)**. [S.l.: s.n.], 2017. p. 858–862.

DIETTERICH, T. G. Approximate statistical tests for comparing supervised classification learning algorithms. **Neural Comput.**, MIT Press, Cambridge, MA, USA, v. 10, n. 7, p. 1895–1923, oct 1998. ISSN 0899-7667. Disponível em: <https://doi.org/10.1162/089976698300017197>.

DILEEP, M. R.; NAVANEETH, A. V.; ABHISHEK, M. A novel approach for credit card fraud detection using decision tree and random forest algorithms. In: **2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)**. [S.l.: s.n.], 2021. p. 1025–1028.

ENGELEN JESPER E.AND HOOS, H. H. van. A survey on semi-supervised learning. **Machine Learning**, v. 109, n. 2, p. 373–440, Feb 2020. ISSN 1573-0565. Disponível em: <https://doi.org/10.1007/s10994-019-05855-6>.

ESENOGHO, E.; MIENYE, I. D.; SWART, T. G.; ARULEBA, K.; OBAIDO, G. A neural network ensemble with feature engineering for improved credit card fraud detection. **IEEE Access**, v. 10, p. 16400–16407, 2022.

FAN, S.-K. S.; HSU, C.-Y.; TSAI, D.-M.; HE, F.; CHENG, C.-C. Data-driven approach for fault detection and diagnostic in semiconductor manufacturing. **IEEE Transactions on Automation Science and Engineering**, v. 17, n. 4, p. 1925–1936, 2020.

FICO. **"Card Fraud"**. 2023. Disponível em: <https://www.fico.com/en/solutions/card-fraud>. Acesso em: 20 Junho 2023.

FOODY, G.; MATHUR, A. A relative evaluation of multiclass image classification by support vector machines. **IEEE Transactions on Geoscience and Remote Sensing**, v. 42, n. 6, p. 1335–1343, 2004.

HARRIS, C. R.; MILLMAN, K. J.; WALT, S. J. van der; GOMMERS, R.; VIRTANEN, P.; COURNAPEAU, D.; WIESER, E.; TAYLOR, J.; BERG, S.; SMITH, N. J.; KERN, R.; PICUS, M.; HOYER, S.; KERKWIJK, M. H. van; BRETT, M.; HALDANE, A.; RÍO, J. F. del; WIEBE, M.; PETERSON, P.; GÉRARD-MARCHANT, P.; SHEPPARD, K.; REDDY, T.; WECKESSER, W.; ABBASI, H.; GOHLKE, C.; OLIPHANT, T. E. Array programming with NumPy. **Nature**, Springer Science and Business Media LLC, v. 585, n. 7825, p. 357–362, set. 2020. Disponível em: <https://doi.org/10.1038/s41586-020-2649-2>.

HUNTER, J. D. Matplotlib: A 2d graphics environment. **Computing in Science & Engineering**, IEEE COMPUTER SOC, v. 9, n. 3, p. 90–95, 2007.

JR, D. W. H.; LEMESHOW, S.; STURDIVANT, R. X. **Applied Logistic Regression**. [S.l.]: John Wiley & Sons, 2013.

KAELBLING, L. P.; LITTMAN, M. L.; MOORE, A. W. Reinforcement learning: A survey. **Journal of Artificial Intelligence Research**, AI Access Foundation, v. 4, p. 237–285, maio 1996. Disponível em: <https://doi.org/10.1613/jair.301>.

KHAN, S. A.; RANA, Z. A. **Evaluating Performance of Software Defect Prediction Models Using Area Under Precision-Recall Curve (AUC-PR)**. [S.l.: s.n.], 2019. 1-6 p.

KLUYVER, T.; RAGAN-KELLEY, B.; PÉREZ, F.; GRANGER, B.; BUSSONNIER, M.; FREDERIC, J.; KELLEY, K.; HAMRICK, J.; GROUT, J.; CORLAY, S.; IVANOV, P.; AVILA, D.; ABDALLA, S.; WILLING, C.; TEAM, J. development. Jupyter notebooks ? a publishing format for reproducible computational workflows. In: LOIZIDES, F.; SCMIDT, B. (Ed.). **Positioning and Power in Academic Publishing: Players, Agents and Agendas**. IOS Press, 2016. p. 87–90. Disponível em: <https://eprints.soton.ac.uk/403913/>.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **Nature**, v. 521, n. 7553, p. 436–444, May 2015. ISSN 1476-4687. Disponível em: <https://doi.org/10.1038/nature14539>.

LEMAîTRE, G.; NOGUEIRA, F.; ARIDAS, C. K. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. **Journal of Machine Learning Research**, v. 18, n. 17, p. 1–5, 2017. Disponível em: <http://jmlr.org/papers/v18/16-365.html>.

LI, J. P.; HAQ, A. U.; DIN, S. U.; KHAN, J.; KHAN, A.; SABOOR, A. Heart disease identification method using machine learning classification in e-healthcare. **IEEE Access**, v. 8, p. 107562–107582, 2020.

MCKINNEY Wes. Data Structures for Statistical Computing in Python. In: WALT Stéfan van der; MILLMAN Jarrod (Ed.). **Proceedings of the 9th Python in Science Conference**. [S.l.: s.n.], 2010. p. 56 – 61.

MITTAL, S.; TYAGI, S. **Performance Evaluation of Machine Learning Algorithms for Credit Card Fraud Detection**. [S.l.: s.n.], 2019. 320-324 p.

Moura, Hugo. **"Relatório Axur: phishing cresce 99,23% em um ano"**. 2021. Disponível em: <https://blog.axur.com/pt/relat\%C3\%B3rio-da-atividade-criminosa-online-no-brasil-q4-2020>. Acesso em: 20 Junho 2023.

Nilson Report. **"Card Fraud Losses Worldwide"**. 2022. Disponível em: <https://nilsonreport.com/newsletters/1232/>. Acesso em: 18 Junho 2023.

NWANKPA, C.; IJOMAH, W.; GACHAGAN, A.; MARSHALL, S. **Activation Functions: Comparison of trends in Practice and Research for Deep Learning**. arXiv, 2018. Disponível em: <https://arxiv.org/abs/1811.03378>.

PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O.; BLONDEL, M.; PRETTENHOFER, P.; WEISS, R.; DUBOURG, V.; VANDERPLAS, J.; PASSOS, A.; COURNAPEAU, D.; BRUCHER, M.; PERROT, M.; DUCHESNAY, E. Scikit-learn: Machine learning in Python. **Journal of Machine Learning Research**, v. 12, p. 2825–2830, 2011.

RAJESHWARI, T.; VARDHINI, P. A. H.; REDDY, K. M. K.; PRIYA, K. K.; SREEJA, K. Smart agriculture implementation using iot and leaf disease detection using logistic regression. In: **2021 4th International Conference on Recent Developments in Control, Automation Power Engineering (RDCAPE)**. [S.l.: s.n.], 2021. p. 619–623.

RANDHAWA, K.; LOO, C. K.; SEERA, M.; LIM, C. P.; NANDI, A. K. Credit card fraud detection using adaboost and majority voting. **IEEE Access**, v. 6, p. 14277–14284, 2018.

SAS. **"Fraud, Anti-Money Laundering Security Intelligence Prevent fraud. Achieve compliance. Preserve security."**. 2023. Disponível em: <https://www.sas.com/pt_br/solutions/fraud-security-intelligence.html>. Acesso em: 18 Junho 2023.

SHAHRTASH, S. M.; SARLAK, M. High impedance fault detection using harmonics energy decision tree algorithm. In: **2006 International Conference on Power System Technology**. [S.l.: s.n.], 2006. p. 1–5.

SHORTEN, C.; KHOSHGOFTAAR, T. M. A survey on image data augmentation for deep learning. **Journal of Big Data**, Springer, v. 6, n. 1, p. 60, 2019.

SINGH, D.; SINGH, B. Investigating the impact of data normalization on classification performance. **Applied Soft Computing**, v. 97, p. 105524, 2020. ISSN 1568-4946. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1568494619302947>.

SOFAER, H. R.; HOETING, J. A.; JARNEVICH, C. S. The area under the precision-recall curve as a performance metric for rare binary events. **Methods in Ecology and Evolution**, v. 10, n. 4, p. 565–577, 2019. Disponível em: <https://besjournals.onlinelibrary.wiley.com/doi/abs/10.1111/2041-210X.13140>.

TEAM, T. pandas development. **pandas-dev/pandas: Pandas**. Zenodo, 2020. Disponível em: <https://doi.org/10.5281/zenodo.3509134>.

ULB. **"Credit Card Fraud Detection. Anonymized credit card transactions labeled as fraudulent or genuine."**. 2018. Disponível em: <https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud>. Acesso em: 18 Junho 2023.

WASKOM, M. L. seaborn: statistical data visualization. **Journal of Open Source Software**, The Open Journal, v. 6, n. 60, p. 3021, 2021. Disponível em: <https://doi.org/10.21105/joss.03021>.

WYTHOFF, B. J. Backpropagation neural networks: A tutorial. **Chemometrics and Intelligent Laboratory Systems**, v. 18, n. 2, p. 115–155, 1993. ISSN 0169-7439. Disponível em: <https://www.sciencedirect.com/science/article/pii/016974399380052J>.

XU, L.; CHOW, M.-Y. A classification approach for power distribution systems fault cause identification. **IEEE Transactions on Power Systems**, v. 21, n. 1, p. 53–60, 2006.

XUAN, S.; LIU, G.; LI, Z.; ZHENG, L.; WANG, S.; JIANG, C. **Random forest for credit card fraud detection**. [S.l.: s.n.], 2018. 1-6 p.

ZEILER, M.; RANZATO, M.; MONGA, R.; MAO, M.; YANG, K.; LE, Q.; NGUYEN, P.; SENIOR, A.; VANHOUCKE, V.; DEAN, J.; HINTON, G. On rectified linear units for speech processing. In: **2013 IEEE International Conference on Acoustics, Speech and Signal Processing**. [S.l.: s.n.], 2013. p. 3517–3521.