

UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

KLEBERSON JOHN SANTOS DE MARIA

CRIPTOGRAFIA COMPLETAMENTE HOMOMÓRFICA COM CONCRETE ML: ANÁLISE DE DESEMPENHO COM O SCIKIT-LEARN

CAMPINA GRANDE - PB

2025

KLEBERSON JOHN SANTOS DE MARIA

CRIPTOGRAFIA COMPLETAMENTE HOMOMÓRFICA COM CONCRETE ML: ANÁLISE DE DESEMPENHO COM O SCIKIT-LEARN

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador : Eanes Torres Pereira

CAMPINA GRANDE - PB

2025

M332c Maria, Kleberson John Santos de.
Criptografia completamente homomórfica com Concrete ML: análise de desempenho com o Scikit-learn / Kleberson John Santos de Maria. – Campina Grande, 2025.
07 f.

Artigo (Bacharelado em Ciência da Computação) – Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática, 2025.
"Orientação: Prof. Dr. Eanes Torres Pereira".
Referências.

1. Criptografia Completamente Homomórfica (FHE). 2. Machine Learning. 3. Concrete ML. 4. Scikit-learn. 5. Análise Comparativa. 6. Avaliação de Desempenho. I. Pereira, Eanes Torres. II. Título.

CDU 004.056.55(04)

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECÁRIA SEVERINA SUELI DA SILVA OLIVEIRA CRB-15/225

KLEBERSON JOHN SANTOS DE MARIA

CRIPTOGRAFIA COMPLETAMENTE HOMOMÓRFICA COM CONCRETE ML: ANÁLISE DE DESEMPENHO COM O SCIKIT-LEARN

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA:

Eanes Torres Pereira

Orientador – UASC/CEEI/UFCG

Leandro Balby Marinho

Examinador – UASC/CEEI/UFCG

Rohit Gheyi

Examinador – UASC/CEEI/UFCG

Trabalho aprovado em: 09 de MAIO de 2025.

CAMPINA GRANDE - PB

RESUMO

Com a crescente adoção da computação em nuvem para processamento de dados, incluindo informações sensíveis, a segurança da informação em pipelines de aprendizado de máquina tornou-se crucial. A criptografia tradicional protege dados em repouso e em trânsito, mas a necessidade de processamento em texto simples durante a execução de modelos representa uma vulnerabilidade de privacidade. A Criptografia Completamente Homomórfica (FHE) surge como uma solução promissora, permitindo computações diretamente em dados criptografados para mitigar esse risco. No entanto, sua aplicação em algoritmos de ML enfrenta desafios práticos, especialmente em relação ao potencial impacto na acurácia e eficácia dos modelos. Para investigar a viabilidade de manter o desempenho sob FHE, este estudo realiza uma análise comparativa entre modelos tradicionais do Scikit-Learn e suas implementações criptografadas no Concrete ML. Utilizando dados de saúde pública relacionados à COVID-19, avaliou-se o desempenho de Regressão Logística e Árvores de Decisão em termos de acurácia, precisão, recall e AUC-ROC. Os resultados demonstram que enquanto as Árvores de Decisão mantiveram desempenho estatisticamente equivalente em sua versão criptografada, a Regressão Logística apresentou redução de 8% na acurácia, porém com recall excepcionalmente alto (>99%). Testes estatísticos realizados confirmaram estas diferenças, sugerindo que a FHE atualmente oferece melhor custo-benefício para modelos não-lineares. O estudo conclui que, embora existam trade-offs entre privacidade e desempenho, a FHE já se mostra viável para casos de uso específicos onde a confidencialidade dos dados é prioritária, especialmente quando combinada com otimizações de hardware e técnicas de quantização adaptativa.

FULLY HOMOMORPHIC ENCRYPTION WITH CONCRETE ML: PERFORMANCE ANALYSIS WITH SCIKIT-LEARN

ABSTRACT

With the growing adoption of cloud computing for data processing, including sensitive information, information security in machine learning pipelines has become crucial. Traditional cryptography protects data at rest and in transit, but the need for plaintext processing during model execution represents a privacy vulnerability. Fully Homomorphic Encryption (FHE) has emerged as a promising solution, allowing computations directly on encrypted data to mitigate this risk. However, its application in ML algorithms faces practical challenges, especially in relation to the potential impact on the accuracy and effectiveness of the models. To investigate the feasibility of maintaining performance under FHE, this study performs a comparative analysis between traditional Scikit-Learn models and their encrypted implementations in Concrete ML. Using public health data related to COVID-19, the performance of Logistic Regression and Decision Trees was evaluated in terms of accuracy, precision, recall and AUC-ROC. The results show that while Decision Trees maintained statistically equivalent performance in their encrypted version, Logistic Regression showed an 8% reduction in accuracy, but with exceptionally high recall (>99%). Statistical tests carried out confirmed these differences, suggesting that FHE currently offers better value for money for non-linear models. The study concludes that, although there are trade-offs between privacy and performance, FHE is already proving viable for specific use cases where data confidentiality is a priority, especially when combined with hardware optimizations and adaptive quantization techniques.

SUMÁRIO

1. INTRODUÇÃO.....1
2. FUNDAMENTAÇÃO TEÓRICA.....2
2.1 Criptografia completamente homomórfica (FHE).....2
2.2 Concrete ML.....2
2.3 Trabalhos Relacionados.....3
3. METODOLOGIA.....4
3.1 Ambiente Experimental.....4
3.2 Base De Dados.....4
3.3 Modelos Selecionados.....4
3.4 Métricas De Avaliação.....5
4. RESULTADOS E DISCUSSÃO.....5
4.1 Desempenhos dos modelos.....5
4.2 Estatísticas de Resultados.....5
5. CONCLUSÃO.....6
6. AGRADECIMENTOS.....6
7. REFERÊNCIAS.....7

Criptografia Completamente Homomórfica com Concrete ML: Análise de desempenho com o Scikit-learn

Kleberson John Santos de Maria
Universidade Federal de Campina Grande, Campina Grande, Paraíba
kleberson.maria@ccc.ufcg.edu.br

Eanes Torres Pereira
Universidade Federal de Campina Grande, Campina Grande, Paraíba
eanes@computacao.ufcg.edu.br

RESUMO

Com a crescente adoção da computação em nuvem para processamento de dados, incluindo informações sensíveis, a segurança da informação em pipelines de aprendizado de máquina tornou-se crucial. A criptografia tradicional protege dados em repouso e em trânsito, mas a necessidade de processamento em texto simples durante a execução de modelos representa uma vulnerabilidade de privacidade. A Criptografia Completamente Homomórfica (FHE) surge como uma solução promissora, permitindo computações diretamente em dados criptografados para mitigar esse risco. No entanto, sua aplicação em algoritmos de ML enfrenta desafios práticos, especialmente em relação ao potencial impacto na acurácia e eficácia dos modelos. Para investigar a viabilidade de manter o desempenho sob FHE, este estudo realiza uma análise comparativa entre modelos tradicionais do Scikit-Learn e suas implementações criptografadas no Concrete ML. Utilizando dados de saúde pública relacionados à COVID-19, avaliou-se o desempenho de Regressão Logística e Árvores de Decisão em termos de acurácia, precisão, recall e AUC-ROC. Os resultados demonstram que enquanto as Árvores de Decisão mantiveram desempenho estatisticamente equivalente em sua versão criptografada, a Regressão Logística apresentou redução de 8% na acurácia, porém com recall excepcionalmente alto (>99%). Testes estatísticos realizados confirmaram estas diferenças, sugerindo que a FHE atualmente oferece melhor custo-benefício para modelos não-lineares. O estudo conclui que, embora existam trade-offs entre privacidade e desempenho, a FHE já se mostra viável para casos de uso específicos onde a confidencialidade dos dados é prioritária, especialmente quando combinada com otimizações de hardware e técnicas de quantização adaptativa.

PALAVRAS-CHAVE:

Criptografia Completamente Homomórfica, FHE, Machine Learning, Concrete ML, Scikit-learn, Análise Comparativa, Avaliação de Desempenho

1. INTRODUÇÃO

A crescente adoção de computação em nuvem para processamento de dados sensíveis [1,2] tem exposto vulnerabilidades críticas nos pipelines de machine learning. Embora a criptografia tradicional proteja dados em repouso e em trânsito, as etapas de processamento em texto simples durante ETL e inferência via APIs representam riscos significativos à privacidade [3]. Neste contexto, a Criptografia Completamente Homomórfica (FHE) emerge como solução técnica promissora ao permitir operações computacionais diretamente sobre dados criptografados [4], preservando a confidencialidade durante todo o ciclo de vida do dado.

A aplicação prática da FHE em ambientes de produção de machine learning enfrenta dois desafios principais inter relacionados: o alto custo computacional inerente às operações criptográficas e os potenciais impactos na acurácia e utilidade dos modelos. Esses desafios tornam-se particularmente críticos em domínios sensíveis como saúde e finanças, onde requisitos rigorosos de privacidade devem coexistir com a necessidade de modelos precisos [3]. Ferramentas como o Concrete ML [5] buscam mitigar essas barreiras ao oferecer uma interface familiar baseada no Scikit-learn, mas a literatura carece de análises rigorosas que quantifiquem estatisticamente as compensações entre privacidade e desempenho em cenários reais.

Este trabalho aborda essa lacuna através de uma análise comparativa abrangente entre implementações tradicionais no Scikit-learn e suas contrapartes criptografadas no Concrete ML. Utilizando dados reais de saúde pública relacionados à COVID-19, avaliamos sistematicamente as diferenças em métricas críticas como acurácia, precisão, F1-score e AUC-ROC, complementando a análise com testes estatísticos robustos (Shapiro-Wilk e Wilcoxon) para determinar a significância prática dessas variações. Nossos resultados demonstram que, embora existam trade-offs mensuráveis, a FHE já apresenta viabilidade operacional para certas classes de modelos, particularmente em aplicações onde a privacidade é prioritária.

REPOSITÓRIOS

Segue abaixo o Notebook com os códigos relativos ao experimento:

https://colab.research.google.com/drive/1eBPJbLt4pO4-zrNtR6zWRnOECpJoa8or?usp=sharing

E a base de dados usada [7]:

https://dados.gov.br/dados/conjuntos-dados/srag-2021-e-2022.

2. FUNDAMENTAÇÃO TEÓRICA

2.1 Criptografia completamente homomórfica (FHE)

A Criptografia Completamente Homomórfica (FHE) é um esquema criptográfico baseado na teoria de grupos que permite a execução de operações sobre dados criptografados, gerando resultados que, quando descriptografados, equivalem aos obtidos em processamento de texto simples [4]

Seja S o conjunto dos textos/valores, a, b c $\in$ S elementos de S que queremos que sejam criptografados, e (+) uma operação sobre esse conjunto, de tal forma que define um grupo, ou seja, valem as seguintes propriedades:

1. Fechamento:

$$a + b \in S, \forall\ a,b$$

2. Associatividade:

$$a + (b + c) = (a + b) + c, \forall\ a,b,c \in S$$

3. Elemento neutro:

$$\exists\ e \in S \mid a + e = e + a = a, \forall\ a \in S$$

4. Inverso:

$$\exists!\ u \in S \mid a + u = u + a = e, \forall\ a \in S$$

No caso de S ser um anel, ele deve ter uma segunda operação (x) e ter mais estas propriedades:

5. Comutatividade da soma (+):

$$a + b = b + a, \forall\ a,b \in S.$$

6. Fechamento do produto (x):

$$a \times b \in S, \forall\ a,b$$

7. Associatividade do produto (x):

$$a \times (b \times c) = (a \times b) \times c, \forall\ a,b,c \in S$$

8. Distributividade do produto (x) com relação a soma (+):

   8.1. à direita:

$$a \times (b + c) = a \times b + a \times c, \forall\ a,b,c \in S$$

   8.2. à esquerda:

$$(a + b) \times c = a \times c + b \times c, \forall\ a,b,c \in S$$

Essas definições vêm do conceito de estruturas algébricas. Para uma melhor compreensão recomenda-se [14] e [15], mas podemos resumir que queremos que os elementos do conjunto S sejam bem-comportados em relação a estas operações, tal qual os números naturais com a soma e os números racionais com o produto.

Com o conjunto S satisfazendo essas propriedades, podemos falar de homomorfismo, uma propriedade de estruturas algébricas que permite fazer associação de operações de diferentes conjuntos.

Voltando ao nosso problema, Seja C o conjunto dos textos cifrados, esse conjunto é dotado de uma operação $\oplus$ e que também seja um grupo (ou um anel, com uma segunda operação $\odot$). Se K é o conjunto das chaves de segurança, E um algoritmo/função de encriptação e D o algoritmo/função de decriptação correspondente, o sistema (S, C, K, E, D) é dito uma criptografia homomórfica se, para todos os elementos k em K e para todos os elementos a, b em S vale que:

$$E_k(a + b) = E_k(a) \oplus E_k(b)$$

Se essa propriedade é válida para uma única operação (+ e $\oplus$) ela é dita parcialmente homomórfica, se vale tanto para "soma" e produto" é dita uma criptografia completamente homomórfica (*fully homomorphic encryption*)

Exemplificando: se a e b são números inteiros e se quer saber da soma deles em texto cifrado, basta que some suas versões cifradas e o resultado pode ser mapeado para um valor correspondente no contradomínio, preservando as propriedades do grupo. Isso significa que mesmo sem sabermos descriptografar à priori, poderíamos fazer operações com os textos cifrados que levem a resultados úteis.

Um exemplo de sistema de criptografia parcialmente homomórfica é o esquema RSA onde a partir do produto dos textos cifrados, podemos obter o texto cifrado do produto dos textos originais. É importante notar que o RSA, em sua forma padrão, não oferece essa propriedade para a operação de soma, por isso é apenas parcialmente homomórfica.

O primeiro modelo funcional de criptografia completamente homomórfica foi desenvolvido em 2009 com o trabalho de Gentry a partir de teoria dos grupos com reticulados ideais [4,16,17]. Outros esquemas foram desenvolvidos, a saber os mais famosos BFV [18], GFV [19], CKKS [20], FHEW [21] e TFHE [22]. A explicação detalhada destes foge do escopo deste texto. A fhe.org [23] é uma comunidade pública com ênfase no estudo do FHE, possui diferentes materiais e recursos dos artigos mais influentes, onde se pode obter informações acerca dos formalismos de cada esquema.

Com esta definição em mãos, nota-se que a criptografia homomórfica é uma solução elegante e promissora para o problema da segurança dos dados, pois seria possível fazer operações com dados já criptografados, garantindo assim a comunicação segura. O ponto é que por ser uma tecnologia moderna e computacionalmente custosa, a maioria dos cientistas de dados ainda usam modelos clássicos, como os do Scikit-Learn e afins, pois refazer e aprender como converter seu projeto para uma versão com criptografia homomórfica também acarreta custo e esforço por parte do desenvolvedor e da empresa. Desde então, várias bibliotecas foram desenvolvidas por várias empresas para se ter uma implementação desse sistema. Nesse contexto, a biblioteca Concrete ML se apresenta como uma tentativa de facilitar a aplicação da FHE em projetos de aprendizado de máquina, buscando otimizar o desempenho e reduzir a complexidade de implementação.

2.2 Concrete ML

A Concrete ML é uma biblioteca em python de código aberto com o objetivo de ser um *framework* de aprendizagem de máquina que preserva a privacidade e é baseada em Criptografia Totalmente Homomórfica (FHE) [5], permitindo que desenvolvedores sem conhecimento dos conceitos de FHE possam utilizar suas vantagens. Na Concrete ML podemos fazer a conversão de

modelos já treinados para suas versões em FHE, além de pré-processar e treinar modelos com dados já criptografados.

Seu funcionamento se dá a partir de dois pontos importantes: a quantização de modelos e a compilação dos modelos para suas versões FHE. A quantização se deve ao fato dos esquemas de FHE aceitarem apenas inteiros, então é necessária uma conversão dos tipos de dados para tal, naturalmente implicando numa perda de precisão [6]. Além disso, se torna necessário fornecer ao modelo uma quantidade de bits (*n_bits* para os modelos equivalentes do Scikit-Learn*)* para aferir esse nível de precisão, se tornando um *trade-off* a ser analisado, apesar de um aumento exagerado não implicar em aumento de métricas. A compilação dos modelos é a conversão deste modelo para um modelo FHE compatível. o nome compilação se vem do fato de o Concrete ML servir de *wrapper* para outro programa da Zama, o Concrete, sendo este um compilador para modelos FHE feito em Rust. Isso permite que operações das funções sejam feitas de forma eficiente por conta da linguagem base. É necessário ter feito o ajuste do modelo com o *fit* e só então chamar a função *compile* para realizar a conversão. O processo de compilação é computacionalmente custoso e leva tempo para ser realizado, mas após isso pode ser utilizado normalmente.

A biblioteca se propõe a ser compatível o máximo possível com o Scikit-Learn, onde a nomenclatura das funções em nível de código é bastante semelhante (*fit*, *predict*, *predict_proba* etc.). Isso é uma de suas grandes vantagens pois permite que a curva de aprendizado seja menor para desenvolvedores iniciantes ou não familiarizados com o tema facilitando a adesão.

Vale salientar que o Concrete ML também suporta redes neurais, com um pouco mais de manejo manual. Todas as informações além de explicação para iniciantes podem ser encontradas na documentação da plataforma, fornecida pela Zama [7].

2.3 Trabalhos Relacionados

Nesta seção, serão abordados os trabalhos relacionados ao tema deste projeto, fornecendo um panorama das pesquisas e desenvolvimentos existentes na área de Aprendizado de Máquina com Preservação de Privacidade utilizando Criptografia Completamente Homomórfica, com foco na biblioteca Concrete ML.

O Zama Team [10] publicou um tutorial comparativo que demonstra a aplicação da biblioteca Concrete ML [5] para investigar o desempenho de diversos modelos de regressão, incluindo regressores lineares, baseados em árvores (como Decision Tree Regressor, RandomForestRegressor e XGBoost Regressor) e baseados em redes neurais, em relação aos seus equivalentes no Scikit-learn. O material apresenta resultados de métricas de performance (como R^2) e métricas de tempo obtidas, incluindo tempo de geração de chave e tempo de execução por amostra. As conclusões indicam a viabilidade da execução desses diferentes tipos de modelos de regressão em Criptografia Completamente Homomórfica, indicando que modelos de regressão linear possuem desempenho semelhantes ao do Scikit-Learn, exigindo baixa quantização (*n_bits*) e os modelos de Árvore de Decisão (a exceção do XGBoost) possuem precisão igual ou superior (para RandomForest) que seus equivalentes.

O trabalho de J. Frery et al. [11] apresenta a implementação de algoritmo e avaliação de modelos baseados em árvore, utilizando o esquema TFHE, que serve de base da implementação para o suporte a árvores na Concrete ML [5]. O artigo detalha a técnica, que supera a limitação de control-flow em FHE ao converter a travessia da árvore em operações tensoriais, alavancando representações de inteiros multi-bit e o mecanismo Programmable Bootstrapping (PBS) do TFHE. Os autores demonstram a aplicação do método em diversos datasets e apresentam resultados experimentais que indicam que a acurácia em dados criptografados é muito próxima à dos modelos em texto simples. As análises de desempenho também incluem latência de inferência, que se mostra competitiva com o estado da arte, e mostram que o impacto da precisão da quantização (*n_bits*) na acurácia a e no tempo de execução tem seu valor ótimo na faixa de 6 bits, onde um valor maior aumenta o tempo, mas não a acurácia. Na comparação com uma implementação anterior baseada no esquema BFV, nota-se menor latência com acurácia comparável. Os autores concluem que sua solução representa um avanço, oferecendo vantagens como fácil customização do trade-off entre quantização e velocidade, compatibilidade com métodos de ensemble e acurácia equivalente à execução em texto simples.

A dissertação de Trovero [12] realizou uma análise de performance de aplicações de Criptografia Completamente Homomórfica (FHE), com o objetivo de investigar o custo computacional e a praticidade de circuitos homomórficos. O estudo incluiu uma análise comparativa de bibliotecas FHE como OpenFHE e Concrete da Zama, optando por utilizar a biblioteca Concrete devido à sua maior facilidade de uso e funcionalidades relevantes para a análise. A metodologia dos testes de performance, conduzidos em um ambiente controlado, comparou o desempenho de operações matemáticas básicas em Python com operações equivalentes em FHE (com e sem paralelismo), e avaliou algoritmos de Machine Learning. Os resultados abrangeram testes sobre a dependência da compilação do circuito no conjunto de entrada, e análises de performance para operações matemáticas básicas. Trovero utilizou a biblioteca Concrete ML para implementar modelos (Regressão Logística, Linear SVM, Linear SVR) e comparou seu desempenho com os equivalentes no Scikit-learn em datasets sintéticos. Os achados para os modelos de ML, baseados na medição de diversas métricas de performance (como acurácia, precisão, R^2, MSE, entre outras) e tempos de execução demonstraram um aumento significativo no tempo de execução em FHE comparado ao Scikit-learn, embora a compilação do circuito e a geração da chave representem uma porção considerável desse custo. Em termos de acurácia e outras métricas de avaliação, os modelos implementados com Concrete ML apresentaram resultados bastante próximos e comparáveis aos seus equivalentes em Scikit-learn, com pequenas diferenças observadas para modelos de classificação (Regressão Logística, Linear SVM) e resultados muito próximos para regressão (Linear SVR). A conclusão reforça que a aplicação de FHE resulta em um aumento significativo no tempo computacional, destacando o trade-off entre segurança e performance, e sugere a necessidade de análises de performance mais abrangentes em futuros trabalhos com outros algoritmos e bibliotecas.

Embora os trabalhos analisados comprovem a viabilidade operacional da FHE em ML, todos compartilham uma limitação fundamental: a falta de análise estatística rigorosa sobre as diferenças de desempenho entre modelos criptografados e originais. Nosso estudo supre essa lacuna através de: (1) testes de hipótese formais (Shapiro-Wilk, Wilcoxon) com dados reais de saúde; (2) quantificação dos trade-offs entre métricas conflitantes; e (3) avaliação em cenários clinicamente relevantes onde falsos negativos são críticos. Essa abordagem fornece evidências estatísticas robustas para orientar a adoção prática da FHE.

3. METODOLOGIA

3.1 Ambiente Experimental

Para o ambiente foi usado a plataforma Google Colab, ferramenta de uso global gratuita para executar código python no formato de Jupyter Notebooks (ipynb). Além disso, como bibliotecas instaladas no ambiente temos apenas o Concrete Ml, versão 1.8.0, por meio do gerenciador de pacotes Pip. Vale salientar que outras bibliotecas já instaladas no sistema foram atualizadas e podem ser visualizadas no código. Todas as configurações do ambiente da VM foram os padrões sem alterações para GPU. A versão do Python no momento do experimento foi 3.11.11, lembrando que devido ao hardware ser totalmente controlado, variações nos pacotes podem ocorrer.

O ambiente se justifica pelo amplo uso para fins educacionais, além da viabilidade via *browser*, o que permite fácil acesso ao a recursos e ao código gerado.

3.2 Base De Dados

Para a escolha da base de dados, foram consideradas tanto a necessidade de privacidade dos dados enquanto temática quanto o tamanho para serem problemas representativos. Foi escolhida a base de dados de Síndrome Respiratória Aguda Grave (SRAG), do Portal de Dados Abertos do Governo Federal, do ano de 2021 [8]. Esse ano foi marcante pelo grande número de casos de COVID-19 sendo uma base de dados ampla para uma classificação. O objetivo principal do experimento de aprendizado de máquina foi construir modelos de classificação capazes de prever se um paciente, presente nesta base de dados, foi diagnosticado com COVID. A base original contém 1.731.292 registros, dos quais foram amostrados aleatoriamente 173.129 casos (≈10%) para os experimentos. A nomenclatura das variáveis pode ser vista no dicionário de dados [9].

Para definir a variável alvo (*target variable*), foi utilizada a coluna CLASSI_FIN da base de dados original. Os registros onde o valor da coluna CLASSI_FIN era igual a 5 foram considerados como casos positivos de COVID-19 e receberam o rótulo 1 na nova variável denominada TARGET. Os demais registros foram considerados como casos negativos ou outras condições respiratórias graves e receberam o rótulo 0,

A base de dados original passou por diversas etapas de pré-processamento para ser adequada à modelagem. Inicialmente, foi selecionada uma amostra aleatória de 10% dos registros da base de dados original para reduzir o custo computacional e o tempo de execução dos experimentos. As variáveis preditoras (*features*) utilizadas para treinar os modelos foram: NU_IDADE_N, CS_SEXO, FEBRE, TOSSE, DISPNEIA, GARGANTA, PERD_OLFT, PERD_PALA, CARDIOPATI, DIABETES, OBESIDADE, PNEUMOPATI, UTI, SUPORT_VEN, RAIOX_RES, TOMO_RES, RES_AN, PCR_RESUL e VACINA_COV. A coluna CLASSI_FIN foi removida após a criação da variável alvo TARGET. A escolha foi baseada na forte relação entre essa comorbidades/exames com o diagnóstico de COVID-19

Durante o pré-processamento, valores ausentes em colunas categóricas foram preenchidos com a string 'Desconhecido'. As colunas DOSE_1_COV e DOSE_2_COV (presumivelmente datas de vacinação) foram convertidas para o número de dias desde a data de referência de 26 de fevereiro de 2020 (data do primeiro caso de COVID-19), com valores inválidos ou datas fora de um intervalo razoável sendo substituídos por -1. Categorias menos frequentes nas colunas RAIOX_RES, TOMO_RES e RES_AN foram agrupadas sob a categoria 'Outros' para reduzir a dimensionalidade. As variáveis ordinais PCR_RESUL e VACINA_COV foram codificadas utilizando *Label Encoding*, enquanto as demais variáveis categóricas foram convertidas para um formato numérico através da técnica de *One-Hot Encoding*. A variável numérica NU_IDADE_N foi padronizada utilizando *StandardScaler*. Por fim, foi realizada uma análise de correlação para identificar e remover variáveis altamente correlacionadas (com um limiar de 0,7) com o objetivo de mitigar a multicolinearidade.

3.3 Modelos Selecionados

Foram escolhidos dois modelos do Scikit-Learn e seus equivalentes do Concrete ML: Regressão Logística e Árvore de Decisão. Os modelos específicos usados e parâmetros seguem abaixo.

Parâmetros Comuns: *tree_params* = {

'max_depth': 15*,*

'min_samples_split': 50*,*

'min_samples_leaf':20*,*

'max_leaf_nodes': 100*,*

'max_features': 'sqrt',

'ccp_alpha': 0,005

}

Do Scikit-Learn:

- *LogisticRegression*
  - Parametros: *n_jobs*= -1
- *DecisionTreeClassifier*
  - Parâmetros: *tree_params*

Do Concrete ML:

- *ConcreteLogisticRegression*
  - Parametros: *n_jobs*= -1, *n_bits* = 8
- *ConcreteDecisionTreeClassifier*
  - Parâmetros: *tree_params*, *n_bits*=6

Vale salientar que esses parâmetros foram definidos manualmente, usando tantos padrões indicados pela Zama em relação ao *n_bits*, quanto a valores que permitissem árvores profundas.

A escolha se justifica por serem exemplos de base para os diferentes tipos de grupos de modelos que ambos possuem, regressões e modelos baseados em árvore. Com isso consegue-se ter uma noção de desempenho sem ter um custo computacional /tempo muito alto. Deve-se considerar as limitações do Google Colab de computação de até 12 horas contínuas [9].

3.4 Métricas De Avaliação

Foram escolhidas as métricas mais comuns em relatórios de classificação, acurácia, precisão, f1-score e AUC-ROC. As métricas foram obtidas tanto por validação cruzada quanto para desempenho médio, e em experimentos independentes com diferentes *random states* em cada split. O tempo não será considerado pela diferença de tempo grande demais para necessitar testes de hipóteses.

4. RESULTADOS E DISCUSSÃO

4.1 Desempenhos dos modelos

Para medir a avaliação média de desempenho foi realizada validação cruzada, com 10 *folds* para cada um dos modelos, todos com o mesmo *random state*. Os resultados se seguem:

Modelo \ Métrica	**acurácia**	**Precisão**	**recall**	**f1-score**	**AUC-ROC**
LR	0,78	0,81	0,89	0,85	0,83
DT	0,76	0,77	0,94	0,85	0,75
CR_LR	0,70	0,70	**0,99**	0,82	0,60
CR_DT	0,76	0,77	0,94	0,85	0,75

Tabela 1: Comparação de métricas (média em 10-fold CV). LR: Regressão Logística; DT: Árvore de Decisão; CR_: Versões Concrete ML. Destaque para o recall >99% no CR_LR

Analisando as tabelas vemos que na média os modelos com métricas mais próximas são os de árvores de decisão. Deve-se observar também um aumento significativo no valor de *recall*, de mais de 10%, provavelmente devido à quantização dos modelos ou a conversão de modelos para tensores pelo *hummingbird*, biblioteca da qual o Concrete ML depende [13]. Curiosamente, para o experimento e os dados, ter um *recall* maior é uma boa opção já que minimiza as chances de falsos negativos. Isso pode ser útil em medicina, pois falsos negativos são mais graves e podem acarretar mortes. Mesmo assim, à exceção do *recall*, todas as métricas dos modelos do Concrete ML foram inferiores aos seus equivalentes

Também vale salientar a diferença entre o valor de AUC-ROC do modelo de regressão do Concrete ML que não aparenta ser compatível com os de *precision* e *recall.* Isso pode indicar uma diferença nos valores devido a forma como a Concrete ML funciona internamente com os valores encriptados, possivelmente no limiar de classificação/calibração, já que na chamada as funções de *predict*, todo o *pipeline* de encriptação, operações FHE e decriptação são realizados.

4.2 Estatísticas de Resultados

Foi realizado teste de hipótese para avaliar se as diferenças entre as métricas são estatisticamente significativas. Foram realizados 15 experimentos para avaliação. Na divisão entre treino e teste foi usado *holdout* 80/20 e os seguintes random states:

random_states: [42, 123, 456, 789, 101, 222, 333, 444, 555, 666,777, 888, 999, 112, 234]

Para cada *random state* as mesmas métricas da validação foram obtidas e serviram para o teste de hipóteses.

Para averiguar a normalidade das amostras foi realizado o Teste de Shapiro-Wilk, onde a hipótese nula (H_0) é de normalidade dos dados e obtemos os seguintes resultados:

Modelos	**Métrica**	**Estatística W**	**Valor-p**	**Normalidade (α=0,05)**
LR → CR_LR	Acurácia	0,921	0,197	Não rejeita H_0
DT → CR_DT	Acurácia	0,614	< 0,001	Rejeita H_0
LR → CR_LR	Precisão	0,951	0,537	Não rejeita H_0
DT → CR_DT	Precisão	0,603	< 0,001	Rejeita H_0
LR → CR_LR	Recall	0,830	0,009	Rejeita H_0
DT → CR_DT	Recall	0,572	< 0,001	Rejeita H_0
LR → CR_LR	F1-score	0,844	0,015	Rejeita H_0
DT → CR_DT	F1-score	0,620	< 0,001	Rejeita H_0
LR → CR_LR	AUC-ROC	0,895	0,079	Não rejeita H_0
DT → CR_DT	AUC-ROC	0,659	< 0,001	Rejeita H_0

Tabela 2 - Teste de normalidade de Shapiro-Wilk para as diferenças entre modelos Scikit-learn e Concrete ML

Os resultados do teste de Shapiro-Wilk (Tabela 2) revelam padrões distintos entre modelos lineares e não-lineares:

Para CR_LR, apenas Recall (p=0,009) e F1-score (p=0,015) violaram a normalidade. Em CR_DT, todas as métricas apresentaram distribuição não-normal ($p < 0,001$).

Essa divergência justifica o uso de testes não-paramétricos. Optou-se pelo teste de Wilcoxon signed-rank para todas as comparações entre Scikit-learn e Concrete ML, dada a violação de normalidade na maioria das métricas (Tabela 2) e a robustez do método a distribuições não-normais. Essa abordagem garante consistência na análise, mesmo para os casos em que o teste de Shapiro-Wilk não rejeitou a normalidade ($p > 0,05$).

Com este teste em mente partimos para a hipóteses:

Hipótese Nula (H_0): *"Não existe diferença*

estatisticamente significativa nas métricas de desempenho (acurácia, precisão, recall, F1-score e AUC-ROC) entre os modelos implementados no Scikit-learn e suas versões equivalentes no Concrete ML."

Hipótese Alternativa (H_1): "*Existe diferença estatisticamente significativa no desempenho entre as implementações originais e as versões criptografadas.*"

Obtivemos os seguintes resultados:

Comparação	Métrica	Estatística W	Valor-p	Interpretação
LR → CR_LR	Precisão	0,000	< 0,001	Diferença significativa
DT → CR_DT	Precisão	10,000	0,917	Diferença não significativa
LR → CR_LR	Recall	0,000	< 0,001	Diferença significativa
DT → CR_DT	Recall	10,000	0,917	Diferença não significativa
LR → CR_LR	F1-score	0,000	< 0,001	Diferença significativa
DT → CR_DT	F1-score	10,000	0,917	Diferença não significativa
LR → CR_LR	AUC-ROC	0,000	< 0,001	Diferença significativa
DT → CR_DT	AUC-ROC	15.000	0,203	Diferença não significativa

Tabela 3 - Resultados do Teste de Wilcoxon Signed-Rank para Comparação de Modelos

Para os modelos lineares de Regressão Logística, observou-se uma queda expressiva e estatisticamente significativa em todas as métricas de desempenho ($p < 0,001$), com a acurácia apresentando redução de 8 pontos percentuais (de 78% para 70%) após a implementação da criptografia homomórfica. Esse padrão foi consistente em todas as métricas avaliadas, indicando que a transformação para FHE compromete substancialmente o desempenho de modelos lineares.

Em contraste, os modelos baseados em Árvores de Decisão mostraram resiliência à implementação FHE, mantendo desempenho estatisticamente equivalente ($p > 0,05$) em todas as métricas. Essa estabilidade sugere que a natureza discreta e não-linear das árvores de decisão as torna mais adequadas para implementações com criptografia homomórfica.

5. CONCLUSÃO

Este estudo demonstrou que a Criptografia Totalmente Homomórfica (FHE), implementada com a biblioteca Concrete ML, é uma solução viável e eficiente para tarefas de aprendizado de máquina que exigem privacidade de dados rigorosa. Embora os resultados tenham expressado diferenças significativas no desempenho de modelos padrão (Scikit-learn) e modelos criptografados, a evidência corrobora o valor da FHE na preservação da utilidade dos modelos, apesar das restrições de criptografia.

Experimentos mostraram que, embora modelos lineares (como regressão logística) sofram mais em termos de reduções em métricas como precisão e AUC-ROC da quantização necessária para operações homomórficas, eles obtém *recall* muito alto (>99%) – uma troca desejável para aplicações onde a redução de falsos negativos é de extrema importância, por exemplo, diagnóstico médico ou detecção de fraude. Por outro lado, modelos baseados em árvores (Árvores de Decisão) mostraram desempenho estatisticamente equivalente aos modelos de texto simples, provando que o FHE pode ser aplicado sem sofrer perdas significativas em algoritmos não lineares.

Essas descobertas reafirmam que, embora haja perda de desempenho computacional inerente ao FHE, a tecnologia já é viável para certas aplicações, particularmente quando a privacidade não é negociável. O Concrete ML, por meio de sua API baseada em Scikit-Learn, à qual os desenvolvedores estão acostumados, reduz muito a barreira de entrada para adoção do FHE sem exigir conhecimento especializado em criptografia.

Para que o FHE seja mais amplamente adotado na produção, são necessários os seguintes itens:

- Aceleração de hardware (GPU/TPU) para inferência criptografada em velocidades mais altas.
- Pesquisa sobre modelos híbridos, com apenas as partes privadas do pipeline sob FHE, equilibrando segurança e desempenho.
- Pesquisa sobre técnicas de quantização adaptativa para minimizar perda de precisão em modelos lineares.

Em resumo, este artigo reafirma o FHE como uma tecnologia promissora para *machine learning* com preservação de privacidade, enfatizando que, mesmo com suas limitações atuais, os benefícios de privacidade fazem com que valha a pena implantá-lo em aplicações de alto risco. Como os avanços de hardware e algoritmos estão no horizonte, a lacuna de desempenho provavelmente diminuirá, consolidando a criptografia homomórfica como uma pedra angular da IA responsável.

6. AGRADECIMENTOS

Primeiramente, gostaria de agradecer a Deus e depois a minha avó, Maria Lucia, e meu avô Antônio, em memória, por me permitirem que chegasse até aqui, cuidando de mim quando mais novo e me apoiando a seguir nos estudos. Tudo o que consegui na minha vida foi graças a eles.

Também agradeço ao resto da minha família por serem a segunda camada de apoio nas dificuldades da vida e ajudando nos estudos quando mais novo. Cito minha Mãe Cleuda, meu irmão Kedson, meu pai Maurício, meu sobrinho Pietro e os meus tios maternos, os quais tive mais proximidade na infância: Sandra, Kelma, Cledna, Selma, Cleodomar, Antônio Filho e Josimar. Que cada um de seus familiares sintam-se representados e saibam que direta ou indiretamente fizeram parte do que sou hoje.

Agradeço aos colegas de faculdade que consegui durante essa jornada. cada um deles foi importante a esse sucesso, nos ajudando nas dificuldades e fazendo a jornada mais agradável. Em especial à Ricardo, Forlán, Iele e Thiago, amigos que levo da

graduação para a vida toda. Também agradeço a Edvânia por ser um apoio grande na minha vida desde antes de entrar neste curso.

Em especial, agradeço também aos professores que me apoiaram e mostraram o poder transformador da educação. Luciano Gama, no ensino fundamental, Marcos Mesquita, do IFPB, e em especial ao meu professor orientador, Eanes Pereira, por me auxiliar durante a graduação, com dicas valiosas, e me permitir escrever este trabalho, apesar das dificuldades da vida. Também agradeço à banca pelas colocações pertinentes ao trabalho

No mais agradeço à Universidade Federal de Campina Grande e a sociedade por permitirem realizar meu sonho.

7. REFERÊNCIAS

[1] CISCO. Global Cloud Index Projects Cloud Traffic to Represent 95 Percent of Total Data Center Traffic by 2021. Disponível em: <https://newsroom.cisco.com/c/r/newsroom/en/us/a/y2018/m02/global-cloud-index-projects-cloud-traffic-to-represent-95-percent-of-total-data-center-traffic-by-2021.html>. Acesso em: 5 mar. 2025.

[2] IBGE. Pesquisa Nacional por Amostra de Domicílios Contínua (PNAD Contínua). Disponível em: <https://agenciadenoticias.ibge.gov.br/media/com_mediaibge/arquivos/6d5a4d4e68d61dace3aa220ef4712218.pdf>. Acesso em: 12 mar. 2025.

[3] NCCOE. Privacy-Enhancing Federated Learning for Health Data (NIST Special Publication 1800-26). Disponível em: <https://www.nccoe.nist.gov/publication/1800-26/VolA/index.html>. Acesso em: 3 mar. 2025.

[4] YI, X.; PAULET, R.; BERTINO, E. Homomorphic Encryption and Applications. Springer, 2014.

[5] ZAMA. Getting Started with Concrete ML. Disponível em: <https://docs.zama.ai/concrete-ml/get-started/getting-started>. Acesso em: 8 mar. 2025.

[6] ZAMA. Quantization in Concrete ML. Disponível em: <https://docs.zama.ai/concrete-ml/explanations/quantization>. Acesso em: 14 mar. 2025.

[7] MINISTÉRIO DA SAÚDE (Brasil). SRAG 2021 e 2022. Disponível em: <https://dados.gov.br/dados/conjuntos-dados/srag-2021-e-2022>. Acesso em: 1 mar. 2025.

[8] GOOGLE RESEARCH. Google Colaboratory FAQ. Disponível em: <https://research.google.com/colaboratory/intl/pt-BR/faq.html>. Acesso em: 7 mar. 2025.

[9] MINISTÉRIO DA SAÚDE (Brasil). Dicionário de Dados SRAG Hospitalizado. Disponível em: <https://s3.sa-east-1.amazonaws.com/ckan.saude.gov.br/SRAG/pdfs/Dicionario_de_Dados_SRAG_Hospitalizado_19.09.2022.pdf>. Acesso em: 10 mar. 2025.

[10] ZAMA TEAM. Comparison of Concrete ML Regressors. Disponível em: <https://www.zama.ai/post/comparison-of-concrete-ml-regressors>. Acesso em: 15 mar. 2025.

[11] FRERY, J. et al. Privacy-Preserving Tree-Based Inference with Fully Homomorphic Encryption. arXiv:2303.01254. Disponível em: <https://arxiv.org/pdf/2303.01254>. Acesso em: 2 mar. 2025.

[12] TROVERO, M. Performance Analysis of Fully Homomorphic Encryption for Machine Learning Applications. Dissertação. Disponível em: <https://webthesis.biblio.polito.it/secure/31085/1/tesi.pdf>. Acesso em: 9 mar. 2025.

[13] ZAMA. Hummingbird Usage in Concrete ML. Disponível em: <https://docs.zama.ai/concrete-ml/0,3/advanced-topics/hummingbird_usage>. Acesso em: 6 mar. 2025.

[14] ROSEN, K. H. Handbook of Discrete and Combinatorial Mathematics. CRC Press, 2017.

[15] HUNGERFORD, T. W. Abstract Algebra: An Introduction. Brooks/Cole, 2014.

[16] GENTRY, C. Fully Homomorphic Encryption Using Ideal Lattices. Disponível em: <https://www.cs.cmu.edu/~odonnell/hits09/gentry-homomorphic-encryption.pdf>. Acesso em: 10 abr. 2025.

[17] GENTRY, C. Computing Arbitrary Functions of Encrypted Data. Communications of the ACM, 2010, Disponível em: <https://dl.acm.org/doi/10,1145/1666420,1666444>. Acesso em: 10 abr. 2025.

[18] BRAKERSKI, Z.; GENTRY, C.; VAIKUNTANATHAN, V. Fully Homomorphic Encryption without Bootstrapping. 2011. Disponível em: <https://eprint.iacr.org/2011/277>. Acesso em: 11 abr. 2025.

[19] BRAKERSKI, Z. Fully Homomorphic Encryption without Modulus Switching. 2012. Disponível em: <https://eprint.iacr.org/2012/144>. Acesso em: 11 abr. 2025.

[20] CHEON, J. H. et al. Homomorphic Encryption for Arithmetic of Approximate Numbers. 2017. Disponível em: <https://link.springer.com/chapter/10,1007/978-3-319-70694-8_15>. Acesso em: 11 abr. 2025.

[21] DUCAS, L.; MICCIANCIO, D. FHEW: Bootstrapping Homomorphic Encryption in Less Than a Second. 2014. Disponível em: <https://eprint.iacr.org/2014/816>. Acesso em: 11 abr. 2025.

[22] CHILLOTTI, I. et al. TFHE: Fast Fully Homomorphic Encryption Over the Torus. 2018. Disponível em: <https://eprint.iacr.org/2018/421>. Acesso em: 11 abr. 2025.

[23] FHE.ORG. Community Resources. Disponível em: <https://fhe.org/resources>. Acesso em: 11 abr. 2025.