UNIVERSIDADE FEDERAL DE VIÇOSA

VICTOR AGSTER BONACHE DE LIMA

UTILIZANDO TÉCNICAS DE APRENDIZADO DE MÁQUINA PARA IDENTIFICAR NÚMEROS PRIMOS

VIÇOSA - MINAS GERAIS
2026

VICTOR AGSTER BONACHE DE LIMA

UTILIZANDO TÉCNICAS DE APRENDIZADO DE MÁQUINA PARA IDENTIFICAR NÚMEROS PRIMOS

Monografia, apresentada ao Curso de Matemática da Universidade Federal de Viçosa, como requisito para obtenção do título de Licenciatura em Matemática.

Orientadora: Rosane Soares Moreira Viana

VIÇOSA - MINAS GERAIS
2026

Documento assinado digitalmente
gov.br
VICTOR AGSTER BONACHE DE LIMA
Data: 07/07/2026 14:01:59-0300
Verifique em https://validar.iti.gov.br

VICTOR AGSTER BONACHE DE LIMA

UTILIZANDO TÉCNICAS DE APRENDIZADO DE MÁQUINA PARA IDENTIFICAR NÚMEROS PRIMOS

Monografia apresentada ao Curso de Matemática da Universidade Federal de Viçosa, como requisito para obtenção do título de Licenciatura em Matemática.

APROVADO: 30 de junho de 2026.

Documento assinado digitalmente
gov.br
ARIANE PIOVEZAN ENTRINGER
Data: 07/07/2026 12:03:32-0300
Verifique em https://validar.iti.gov.br

Dra. Ariane Piovezan Entringer
(UFV)

Documento assinado digitalmente
gov.br
POUYA MEHDIPOUR
Data: 07/07/2026 13:11:01-0300
Verifique em https://validar.iti.gov.br

Dra. Pouya Mehdipour
(UFV)

Documento assinado digitalmente
gov.br
ROSANE SOARES MOREIRA VIANA
Data: 07/07/2026 12:36:45-0300
Verifique em https://validar.iti.gov.br

Dra. Rosane Soares Moreira Viana
(Orientadora)
(UFV)

AGRADECIMENTOS

Agradeço, primeiramente, aos meus pais, por todo o amor, apoio e incentivo que me ofereceram ao longo de minha trajetória acadêmica e pessoal. O apoio incondicional foi fundamental para que eu pudesse superar desafios e alcançar este objetivo. Sem vocês, esta conquista não seria possível. Este trabalho é também fruto de todos os esforços, ensinamentos e valores que me transmitiram ao longo da vida.

À minha orientadora, expresso minha profunda gratidão pela dedicação, paciência e comprometimento demonstrados durante todo o processo de desenvolvimento desta monografia. Sua orientação segura, suas contribuições valiosas e sua compreensão nos momentos de dificuldade foram essenciais para a construção deste trabalho. Agradeço pela disponibilidade, pelo incentivo e pela confiança depositada em mim ao longo desta caminhada.

Por fim, agradeço à vida, que me permitiu trilhar este caminho e me tornar quem sou. Sou grato pelas experiências, pelos desafios, pelos aprendizados e pelas oportunidades que contribuíram para o meu crescimento. Agradeço pela possibilidade de expressar minha singularidade, desenvolvendo minhas potencialidades e construindo minha própria identidade. Que eu continue valorizando aquilo que me torna único e buscando evoluir em todas as dimensões da existência.

A todos que, de alguma forma, contribuíram para esta conquista, deixo meu sincero agradecimento.

RESUMO

LIMA, Victor Agster Bonache de. Universidade Federal de Viçosa, junho de 2026. **Utilizando técnicas de aprendizado de máquina para identificar números primos**. Orientadora: Rosane Soares Moreira Viana.

Este trabalho investiga a aplicabilidade de técnicas de aprendizado de máquina na identificação e modelagem de números primos. Foram empregados os algoritmos KNN, Random Forest, XGBoost e MLP em duas tarefas supervisionadas: regressão, para prever o valor de um primo a partir de sua posição, intervalo e classe módulo 4; e classificação, para distinguir números primos de compostos utilizando o valor numérico e a função aritmética $\Omega(n)$. Os modelos foram avaliados sobre uma base inicial de 999.999 primos. Nos experimentos de regressão, apenas o MLP obteve desempenho satisfatório ($R^2 = 0,96$; MAPE $= 1,58\%$), enquanto os demais algoritmos apresentaram R^2 negativos. Na classificação, o uso de $\Omega(n)$ como variável preditora resultou em acurácia perfeita para a maioria dos algoritmos, tornando o problema trivial. Sem $\Omega(n)$, todos os modelos limitaram-se a prever a classe majoritária (composto), obtendo altas acurácias devido ao forte desbalanceamento entre as classes, mas sem conseguir identificar nenhum número primo (sensibilidade zero). Os resultados indicam que o valor numérico isolado não contém informação suficiente para a classificação, e que a engenharia de atributos é determinante para o sucesso da abordagem.

Palavras-chave: Aprendizado de Máquina; Números Primos; Classificação; Regressão; Função Ômega.

ABSTRACT

LIMA, Victor Agster Bonache de. Universidade Federal de Viçosa, June, 2026. **Using machine learning techniques to identify prime numbers**. Adviser: Rosane Soares Moreira Viana.

This work investigates the applicability of machine learning techniques for identifying and modeling prime numbers. The KNN, Random Forest, XGBoost, and MLP algorithms were employed in two supervised tasks: regression, to predict the value of a prime based on its rank, interval, and equivalence class modulo 4; and classification, to distinguish prime numbers from composite numbers using the numerical value and the arithmetic function $\Omega(n)$. The models were evaluated on a dataset of 999,999 primes initially. In regression experiments, only the MLP achieved satisfactory performance ($R^2 = 0.96$; MAPE = 1.58%), while the other algorithms presented negative R^2 values. In classification, the use of $\Omega(n)$ as a predictor variable resulted in perfect accuracy for most algorithms, making the problem trivial. Without $\Omega(n)$, all models were limited to predicting the majority class (composite), achieving high accuracies due to the strong class imbalance, but failing to identify any prime number (zero recall). The results indicate that the numerical value alone does not contain enough information for classification, and that feature engineering is crucial for the success of the approach.

Keywords: Machine Learning; Prime Numbers; Classification; Regression; Omega Function.

Sumário

1 INTRODUÇÃO

Os números primos constituem um dos objetos de estudo mais fundamentais da Teoria dos Números (HARDY e WRIGHT, 2008), pois exercem um papel central tanto no desenvolvimento histórico da disciplina quanto em aplicações modernas. Sua distribuição, entretanto, apresenta caráter irregular e de difícil previsão, característica que tem instigado matemáticos e pesquisadores ao longo dos séculos.

Do ponto de vista prático, os números primos assumem grande relevância em áreas como a criptografia e a segurança da informação, nas quais a dificuldade de fatoração de números inteiros grandes constitui a base de protocolos de segurança digital (NENE e ULUDAG, 2022). Esse aspecto reforça não apenas o interesse teórico, mas também a necessidade de abordagens que ampliem a compreensão de suas propriedades.

Embora resultados expressivos tenham sido alcançados, como os avanços decorrentes da demonstração do Teorema dos Números Primos por De La Vallée-Poussin e Hadamard em 1896, que descreveram a distribuição assintótica dos números primos entre os naturais, a busca por uma expressão determinística capaz de indicar a posição exata desses números na reta numérica permanece um desafio em aberto. Ainda que Honsberger (1976) apresente uma fórmula que, em princípio, permite gerar números primos, sua elevada complexidade a torna inviável para aplicações práticas. Essa limitação evidencia a dificuldade de caracterizar a distribuição dos primos por métodos puramente analíticos, abrindo espaço para abordagens alternativas. Neste contexto, destacam-se as técnicas da ciência de dados e, em particular, do aprendizado de máquina, cujo objetivo é identificar padrões e realizar inferências em cenários de alta complexidade.

Assim, este trabalho tem como objetivo investigar a aplicabilidade de técnicas de aprendizado de máquina na identificação de números primos por meio da comparação do desempenho de diferentes algoritmos e da exploração de abordagens variadas. Busca-se avaliar em que medida tais ferramentas podem contribuir para a análise empírica da distribuição dos primos. Para tanto, serão empregados os algoritmos: KNN, *Random Forest*, *XGBoost* e MLP. O desempenho desses algoritmos será mensurado por métricas estatísticas amplamente utilizadas na literatura (MONARD e BARANAUSKAS, 2003), incluindo o Erro Percentual Absoluto Médio (MAPE) e o Coeficiente de Determinação (R^2) no caso de métodos de regressão, a acurácia e a matriz de confusão, para métodos de classificação.

A relevância deste estudo reside na possibilidade de integrar áreas tradicionalmente distintas, como a matemática pura, a ciência de dados e a inteligência artificial, com o intuito de explorar novas perspectivas sobre um problema clássico (DAVIES, 2021). Ao adotar uma abordagem computacional e preditiva, pretende-se não apenas avaliar a eficácia de diferentes algoritmos, mas também fornecer subsídios para o desenvolvimento de pesquisas futuras.

Esta monografia está estruturada da seguinte forma: a primeira parte corresponde a esta Introdução. Na segunda parte, apresenta-se a Fundamentação Teórica na qual são abordados os temas: Aritmética e Teoria dos Números, Criptografia, Aprendizado de Máquina, métricas estatísticas para avaliação de modelos e uma revisão bibliográfica de estudos semelhantes. Na terceira parte, descreve-se a metodologia adotada no estudo. Na quarta parte, são expostos e analisados os resultados obtidos. Por fim, na quinta e última parte, apresentam-se as considerações finais e as sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta as principais definições, conceitos e fundamentos teóricos que servirão de base para a compreensão e o desenvolvimento deste estudo. De acordo com Gil (2002), "o referencial teórico, mais do que um simples ponto de partida, deve constituir um processo contínuo que acompanha o desenvolvimento da pesquisa, auxiliando o pesquisador a refinar e ajustar suas hipóteses e abordagen".

Dessa forma, o presente capítulo não se limita à exposição estática de conceitos, mas busca estabelecer um diálogo constante entre a teoria e os objetivos do trabalho, de modo a fundamentar as escolhas metodológicas e analíticas adotadas nos capítulos subsequentes.

Para tanto, esta fundamentação teórica está organizada em cinco tópicos principais. O primeiro, intitulado "Aritmética e Teoria dos Números", aborda os fundamentos matemáticos relacionados aos números primos, suas propriedades e relevância histórica. O segundo trata da Criptografia, com destaque para o algoritmo RSA. O terceiro apresenta os conceitos fundamentais de aprendizado de máquina, bem como a descrição dos algoritmos utilizados. O quarto tópico discute as métricas estatísticas empregadas na avaliação do desempenho dos modelos desenvolvidos. Por fim, o quinto tópico contempla uma revisão bibliográfica de trabalhos recentes com temática semelhante à deste estudo.

2.1 Aritmética e Teoria dos Números

A Aritmética constitui um dos ramos mais antigos e fundamentais da Matemática, sendo dedicado ao estudo das propriedades e das operações envolvendo números inteiros, racionais e reais, bem como das relações entre eles. Historicamente, sua origem está associada à necessidade humana de contar, medir e comparar quantidades, tendo sido progressivamente sistematizada por civilizações como a babilônica, a egípcia e a grega (BOYER e MERZBACH, 2011). No contexto matemático contemporâneo, a aritmética abrange tanto aspectos elementares das operações numéricas quanto conceitos teóricos que servem de base para áreas mais avançadas, como a Teoria dos Números.

A Teoria dos Números, por sua vez, é o ramo da Matemática voltado ao estudo das propriedades e das relações entre os números inteiros. De acordo com Burton (2010), essa área busca compreender os padrões e as estruturas subjacentes aos números, investigando desde conceitos básicos, como a divisibilidade, até resultados mais avançados relacionados à distribuição dos números primos. Diferentemente da aritmética elementar, a Teoria dos Números possui um caráter mais abstrato e investigativo, priorizando a compreensão conceitual das propriedades numéricas em detrimento do simples cálculo.

2.1.1 Importância histórica dos números primos

O estudo dos números primos remonta à Grécia Antiga, especialmente à obra *Elementos* de Euclides, na qual é demonstrada a infinitude dos números primos – um dos primeiros resultados de caráter verdadeiramente teórico da Matemática (BURTON, 2010). Esse marco histórico contribuiu para o desenvolvimento inicial da Teoria dos Números, estabelecendo fundamentos importantes para a decomposição dos inteiros e para o raciocínio lógico-dedutivo que caracteriza a matemática moderna.

Séculos mais tarde, Eratóstenes desenvolveu o célebre Crivo de Eratóstenes, um método sistemático para identificar números primos menores que um determinado limite, ainda hoje considerado uma das construções mais elegantes da aritmética (PADILHA, 2013).

Durante o Renascimento e o período moderno, o interesse pelos números primos intensificou-se com o surgimento de novas abordagens e conjecturas. Pierre de Fermat (1601–1665) formulou importantes hipóteses sobre a forma dos números primos, além de contribuir significativamente para o desenvolvimento da aritmética modular. Posteriormente, Leonhard Euler (1707–1783) estabeleceu relações profundas entre os números primos e as funções zeta, sendo o primeiro a evidenciar a conexão entre a série harmônica dos primos e o logaritmo natural, antecipando resultados que seriam formalizados apenas no século XIX (PADILHA, 2013).

Com Carl Friedrich Gauss (1777–1855), o estudo da distribuição dos números primos atingiu um novo patamar. Em seus trabalhos juvenis, Gauss propôs uma estimativa para a quantidade de números primos menores que um dado número n, lançando as bases do Teorema dos Números Primos, posteriormente demonstrado de forma rigorosa por Jacques Hadamard e Charles de la Vallée-Poussin, em 1896 (HARDY e WRIGHT, 2008).

Por fim, na era moderna, os números primos passaram a desempenhar um papel fundamental também fora da matemática pura, especialmente na área da criptografia, onde se tornaram elementos essenciais para a segurança dos sistemas de comunicação.

2.1.2 Conceitos básicos

Para a compreensão deste estudo, apresenta-se uma breve introdução a alguns conceitos básicos de Aritmética e Teoria dos Números.

O primeiro conceito central é o de **divisibilidade**. Diz-se que um número inteiro a é divisível por outro inteiro $b \neq 0$ se existir um número inteiro k tal que

$$a = b \cdot k.$$

Nesse caso, afirma-se que b divide a, denotando-se por $b \mid a$. Caso contrário, escreve-se $b \nmid a$ (PADILHA, 2013).

Por outro lado, temos o conceito de **congruência**, que é fundamental na Teoria dos Números e desempenha papel essencial em diversas aplicações modernas, especialmente na criptografia de chave pública, como o algoritmo RSA, o qual será abordado posteriormente neste trabalho.

Diz-se que dois inteiros a e b são congruentes módulo n, onde $n \in \mathbb{N}^* = \mathbb{N} - \{0\}$, se n divide a diferença $a - b$ (ou seja, $n \mid (a - b)$). Escreve-se

$$a \equiv b \pmod{n}.$$

Em seguida, define-se como **número primo** todo número inteiro $p > 1$ que possui exatamente dois divisores positivos distintos: 1 e ele mesmo (RIZEL, 2014). Essa definição é essêncial para o nosso estudo.

Os números que não satisfazem essa condição são denominados compostos, por poderem ser expressos como o produto de dois ou mais fatores inteiros positivos menores que eles.

Assim, a importância dos números primos também é evidenciada no **Teorema Fundamental da Aritmética**, o qual estabelece que todo número inteiro positivo maior que 1 pode ser representado de maneira única como o produto de números primos, independentemente da ordem dos fatores (RIZEL, 2014). Em termos formais, se $n > 1$, então existem primos p_1, p_2, ..., p_k e inteiros positivos α_1, α_2, ..., α_k tais que

$$n = p_1^{\alpha_1} \cdot p_2^{\alpha_2} \cdot \ldots \cdot p_k^{\alpha_k} \ .$$

A unicidade da fatoração em primos constitui um dos pilares da Teoria dos Números.

Outra definição importante é a de **Máximo Divisor Comum (MDC)** de dois números inteiros a e b. Pode ser denotado por $mdc(a, b)$, ou somente (a, b), e é o maior número inteiro positivo que divide simultaneamente a e b, em notação matemática,

$$mdc(a, b) = \max \{d \in \mathbb{N} \ ; \ d \mid a \text{ e } d \mid b\}.$$

O conceito de coprimalidade deriva dessa definição e diz que, dois inteiros a e b são ditos **coprimos** se

$$mdc(a, b) = 1.$$

2.1.3 Funções aritméticas relevantes

Com isso, temos o conceito de função aritmética, definida como qualquer função cujo domínio é o conjunto dos naturais e cujo valor depende das propriedades aritméticas de seus argumentos (HARDY e WRIGHT, 2008). Em outras palavras, uma função aritmética é uma regra $f : \mathbb{N} \to \mathbb{C}$ que associa a cada número natural um valor, geralmente relacionado à sua estrutura aritmética. Exemplos clássicos incluem a função número de

divisores $d(n)$, a soma dos divisores $\sigma(n)$ e as funções relacionadas à contagem de fatores primos.

Entre as funções aritméticas mais relevantes para os treinamentos posteriormente abordados neste estudo, destacam-se a **função totiente de Euler**, denotada por ϕ, que associa cada inteiro positivo n a quantidade de inteiros positivos menores que n e coprimos a n, e a função ômega, denotada por Ω, associada ao número total de fatores primos de n, considerada as respectivas multiplicidades. Formalmente, a função totiente de Euler é definida por

$$\phi(n) = \#\{k \in N \ ; \ 1 \leq k < n \text{ e } mdc(k, n) = 1\}.$$

em que o símbolo # representa a cardinalidade do conjunto.

E se $n = p_1^{\alpha_1} \cdot p_2^{\alpha_2} \cdot ... \cdot p_k^{\alpha_k}$, com $p_1, p_2, \ldots, \ p_k$ primos, então

$$\Omega(n) = \alpha_1 + \alpha_2 + \cdots + \alpha_k \ .$$

Uma observação importante é que se p é **número primo**, então

$$\phi(p) = p - 1 \quad \text{e} \quad \Omega(p) = 1.$$

2.1.4 Teorema dos Números Primos

A distribuição dos números primos entre os naturais é descrita assintoticamente pelo Teorema dos Números Primos (PNT, do inglês *Prime Number Theorem*), demonstrado independentemente por Jacques Hadamard (1896) e Charles Jean de la Vallée-Poussin (1896). Esse teorema estabelece que, se $\pi(x)$ denota a quantidade total de números primos existentes até o valor de x, então

$$\pi(x) \sim \frac{x}{\ln(x)},$$

ou seja,

$$\lim_{x \to \infty} \frac{\pi(x) \ln(x)}{x} = 1.$$

Em outras palavras, a densidade (probabilidade) dos primos na vizinhança de x é aproximadamente $\frac{1}{\ln(x)}$. Uma consequência imediata é que os primos se tornam cada vez mais raros à medida que se avança na reta numérica, fato que já havia sido conjecturado por Gauss em sua juventude (BURTON, 2010). A função $\frac{x}{\ln(x)}$ fornece uma boa aproximação para a função contadora de números primos $\pi(x)$ embora, conforme Gomes (2020), existam estimativas mais precisas como o logaritmo integral, denotado por $\text{li}(x)$, definido por

$$\text{li}(x) = \int_2^x \frac{dt}{\ln(t)}.$$

2.1.5 Classificação dos números primos via congruência módulo 4

Com excessão do 2, todos os números primos são ímpares. Consequentemente, todo primo ímpar p pode ser expresso na forma $p = 2k+1$, para algum inteiro k. Ao analisarmos a divisão de p por 4, encontramos apenas duas possibilidades de resto: 1 ou 3. Em termos de congruência, temos

$$p \equiv 1 \pmod 4 \quad \text{ou} \quad p \equiv 3 \pmod 4.$$

Os primos congruentes à 1 módulo 4, como 5, 13, 17 e 29, e aqueles congruentes à 3 módulo 4, como 3, 7, 11 e 19 apresentam propriedades aritméticas distintas. Por exemplo, um resultado clássico devido à Fermat estabelece que um primo ímpar p pode ser expresso como soma de dois quadrados, $p = a^2 + b^2$, com $a, b \in \mathbb{Z}$ se e somente se $p \equiv 1 (\text{mod } 4)$ (HARDY e WRIGHT, 2008). Em contra partida, os primos da forma $4k+3$ não admitem esse tipo de representação.

O Teorema dos Números Primos pode ser estendido para progressões aritméticas. Um resultado fundamental nesse cenário é o Teorema de Dirichlet, que assegura a existência de infinitos primos em qualquer progressão aritmética $a + nq$, desde que $\text{mdc}(a, q) = 1$ (BURTON, 2010). Como consequência, existem infinitos primos na congruência 1 módulo 4 quanto na congruência 3 módulo 4.

Para quantificar essa distribuição, definem-se as funções de contagem:

$$\pi_1(x) = \#\{p \leq x : p \equiv 1 \pmod 4\}, \qquad \pi_3(x) = \#\{p \leq x : p \equiv 3 \pmod 4\}.$$

O Teorema dos Números Primos para progressões aritméticas afirma que,

$$\pi_1(x) \sim \pi_3(x) \sim \frac{1}{2}\pi(x) \sim \frac{x}{2 \ln x}.$$

Isto significa que assintoticamente, os números primos se distribuem de maneira equitativa, dividindo-se igualmente entre ambas as classes residuais.

Mas embora as duas contagens sejam assintoticamente iguais, a forma como se aproximam desse equilíbrio é repleta de nuances. Um fenômeno notável, observado por Chebyshev, é que $\pi_3(x)$ tende a ser ligeiramente maior que $\pi_1(x)$ para a maioria dos valores de x (GRANVILLE e MARTIN, 2006). Em outras palavras, a desigualdade $\pi_3(x) > \pi_1(x)$ ocorre com muito mais frequência do que a desigualdade contrária. Esse viés é conhecido como *viés de Chebyshev* (ou *Chebyshev's bias*) e está relacionado com propriedades profundas da função zeta de Dirichlet.

Apesar da aparente dominância de $\pi_3(x)$, a diferença $\pi_3(x) - \pi_1(x)$ muda de sinal infinitas vezes, como provado por Littlewood em 1914 (GRANVILLE e MARTIN, 2006). Contudo, as regiões onde $\pi_1(x) > \pi_3(x)$ são extremamente raras e ocorrem para valores

grandes de x. De acordo com Granville e Martin (2006), a primeira vez que os primos da forma $4n + 1$ assumem a liderança de uma suposta corrida entre $\pi_1(x)$ e $\pi_3(x)$ ocorre no primo 26.861. No entanto, como 26.863 também é primo, $\pi_3(x)$ empata e retoma a liderança até que $\pi_1(x)$ volte à frente em 616.841, mantendo-se à frente em diversos números até 633.798. Então $\pi_3(x)$ recupera a liderança até que $\pi_1(x)$ avança novamente em 12.306.137, permanecendo à frente até 12.382.326. Assim $\pi_3(x)$ retoma a liderança até que $\pi_1(x)$ assume novamente em 951.784.481, mantendo-se até 952.223.506. Em seguida, $\pi_3(x)$ reassume até que $\pi_1(x)$ lidera em 6.309.280.697, até 6.403.150.362. Posteriormente, a $\pi_3(x)$ recupera a liderança até que $\pi_1(x)$ assume em 18.465.126.217, mantendo-se até 19.033.524.538, subsequentemente $\pi_3(x)$ reassume a liderança e a mantém até pelo menos vinte bilhões.

A classificação dos primos segundo o resto módulo 4 é diretamente utilizada na metodologia deste estudo como uma das variáveis preditoras para a tarefa de regressão (variável Z_4). Além disso, o conhecimento da distribuição assintótica e das particularidades das funções $\pi_1(x)$ e $\pi_3(x)$ fornece um arcabouço teórico para interpretar os resultados empíricos obtidos pelos modelos de aprendizado de máquina, especialmente no que tange à possível identificação de padrões residuais na sequência dos primos.

2.2 Criptografia

A criptografia é o ramo da ciência que estuda métodos de proteger informações, tornando-as inteligíveis apenas para aqueles que possuem autorização para acessá-las. Do ponto de vista técnico, pode ser definida como o conjunto de técnicas matemáticas empregadas para transformar dados legíveis em dados cifrados, de forma que apenas indivíduos com a chave correta consigam reverter o processo (TEIXEIRA, 2020). Assim, a criptografia desempenha papel essencial na segurança da informação, garantindo confidencialidade, autenticidade e integridade dos dados.

Historicamente, a criptografia evoluiu de técnicas simples de substituição e transposição de letras, utilizadas desde a Antiguidade, até os sistemas modernos baseados em fundamentos matemáticos sólidos. Com o advento da computação, os métodos criptográficos passaram a explorar propriedades de funções unidirecionais e problemas matemáticos de difícil resolução computacional, como a fatoração de números primos grandes (TEIXEIRA, 2020).

A criptografia moderna baseia-se em princípios da Teoria dos Números, da Álgebra Modular e da Aritmética Computacional. O conceito de chave pública e chave privada foi introduzido por Diffie e Hellman em 1976, inaugurando o paradigma da criptografia assimétrica, em que a codificação e a decodificação utilizam chaves diferentes. Esse avanço permitiu a comunicação segura sem a necessidade de compartilhamento prévio de uma chave secreta.

Entre os sistemas assimétricos, o algoritmo RSA, proposto por Rivest, Shamir e Adleman em 1978, é um dos mais amplamente utilizados e baseia-se na dificuldade de fatorar grandes números compostos. Sua segurança depende do fato de que, embora seja fácil multiplicar dois números primos grandes, é computacionalmente inviável decompor o produto em seus fatores primos.

2.2.1 Algoritmo RSA

O algoritmo RSA utiliza propriedades da aritmética modular e do Teorema de Euler (RIVEST e SHAMIR e ADLEMAN, 1978), que afirma que, para dois números inteiros positivos a e n tais que $mdc(a, n) = 1$, tem-se

$$a^{\phi(n)} \equiv 1 \pmod{n},$$

onde $\phi(n)$ é a função aritmética definida anteriormente na seção 2.1.3.

O processo de geração das chaves e de cifragem/decifragem no RSA segue as seguintes etapas:

I Geração das chaves:

- Escolhem-se dois números primos grandes p e q;
- Calcula-se $n = p \cdot q$;
- Determina-se $\phi(n) = \phi(p \cdot q) = \phi(p) \cdot \phi(q) = (p-1) \cdot (q-1)$;
- Escolhe-se um número inteiro e tal que $1 < e < \phi(n)$ e $mdc(e, \phi(n)) = 1$;
- Calcula-se o número d, inverso multiplicativo de e módulo $\phi(n)$, isto é

$$d \cdot e \equiv 1 \pmod{\phi(n)}.$$

A chave pública é composta pelo par (n, e), e a chave privada, pelo par (n, d).

II Cifragem e Decifragem:

- Dada uma mensagem M (representada como um número $M < n$), o texto cifrado C é obtido por

$$C \equiv M^e \pmod{n}.$$

- Para recuperar a mensagem original, utiliza-se a chave privada

$$M \equiv C^d \pmod{n}.$$

  Essa relação é garantida pelo teorema de Euler, pois

$$M^{ed} \equiv M^{k\phi(n)+1} \equiv M \pmod{n}.$$

A segurança do RSA depende diretamente do tamanho dos primos escolhidos: quanto maiores os valores de p e q, maior o esforço computacional necessário para fatorar n e quebrar o sistema (NENE e ULUDAG, 2022).

2.3 Aprendizado de máquina

O Aprendizado de Máquina (*Machine Learning*) é um subcampo da Inteligência Artificial que tem como objetivo desenvolver algoritmos capazes de identificar padrões em dados e realizar previsões ou decisões com base em experiências anteriores, sem que sejam explicitamente programados para tal. Segundo Mitchell (1997), um programa de computador é dito aprender a partir de uma experiência E com respeito a uma classe de tarefas T e uma medida de desempenho P, se o seu desempenho em T, medido por P, melhora com a experiência E. Dessa forma, o aprendizado de máquina pode ser entendido como um processo de extração de conhecimento a partir de dados, sustentado por fundamentos matemáticos e estatísticos.

O funcionamento do aprendizado de máquina baseia-se na construção de modelos matemáticos e computacionais capazes de generalizar comportamentos observados em amostras de dados para novos casos não vistos. Tais modelos são obtidos por meio de algoritmos de treinamento, que ajustam parâmetros internos de acordo com um critério de erro ou desempenho.

2.3.1 Conceitos básicos

A literatura apresenta diferentes tipos de aprendizado de máquina, que variam conforme os dados fornecidos ao algoritmo durante o processo de treinamento. No **aprendizado não supervisionado**, o algoritmo recebe apenas os dados de entrada, sem rótulos ou saídas esperadas. Seu objetivo é identificar estruturas, padrões ou agrupamentos naturais dentro dos dados. Técnicas como agrupamento (clustering) e redução de dimensionalidade são exemplos desse tipo de abordagem (LURDERMIR, 2021).

Por outro lado, existem métodos como o **aprendizado por reforço**, no qual o agente interage com um ambiente dinâmico e aprende a partir de recompensas e penalidades associadas às suas ações (LURDERMIR, 2021). Esse tipo de aprendizado é amplamente utilizado em controle adaptativo e jogos, mas apresenta natureza distinta das abordagens puramente estatísticas.

Por fim, o **aprendizado supervisionado** que é o de maior relevância para o nosso estudo. Nesse tipo de aprendizado, o modelo é treinado a partir de pares entrada-saída conhecidos, ou seja, para cada exemplo de entrada x_i, o algoritmo conhece a saída esperada y_i. O objetivo é ajustar uma função f tal que $f(x_i) = y_i$ para novas amostras, permitindo inferências sobre dados ainda não observados (BISHOP, 2006).

Dentro do aprendizado supervisionado, os problemas se dividem principalmente em **classificação** e **regressão**.

Nos problemas de **classificação**, o modelo procura atribuir cada amostra a uma das categorias possíveis, que são representadas por rótulos discretos. Formalmente, define-

se uma função $f : \mathbb{R}^{\kappa} \rightarrow \{1, 2, ..., k\}$, em que k é o número de classes. Esse tipo de problema é comum em reconhecimento de padrões, diagnóstico médico e detecção de fraudes (BISHOP, 2006).

E nos problemas de **regressão**, por outro lado, o objetivo é predizer valores contínuos a partir de variáveis independentes. O modelo busca aproximar uma função $f : \mathbb{R}^{\kappa} \rightarrow \mathbb{R}$, de modo a minimizar o erro entre os valores previstos e observados. Essa abordagem é fundamental em contextos que envolvem previsões quantitativas, como séries temporais, economia e modelagem física (BISHOP, 2006).

Um dos desafios centrais do aprendizado de máquina está relacionado à capacidade de generalização de um modelo, isto é, sua habilidade de realizar previsões corretas não apenas sobre os dados de treinamento, mas também sobre novos dados que não foram utilizados durante o processo de aprendizagem. Essa capacidade está diretamente associada ao equilíbrio entre dois comportamentos indesejáveis: o *overfitting* e o *underfitting*.

De acordo com Monard e Baranauskas (2003), o *underfitting* ocorre quando o modelo é muito simples para capturar as relações subjacentes entre as variáveis de entrada e saída. Nesse caso, o algoritmo não consegue aprender de forma satisfatória a estrutura dos dados, apresentando desempenho insatisfatório tanto no conjunto de treinamento quanto no de teste. Em termos formais, pode-se dizer que há viés elevado e baixa variância, pois o modelo assume uma forma restrita (como uma hipótese muito limitada) que não representa adequadamente o fenômeno estudado.

Por outro lado, o *overfitting* ocorre quando o modelo se ajusta excessivamente aos dados de treinamento, inclusive aos ruídos, flutuações aleatórias ou observações atípicas (*outliers*). Isso faz com que o modelo apresente desempenho excelente nos dados de treinamento, mas desempenho significativamente inferior quando aplicado a novos dados. Formalmente, trata-se de um caso de baixo viés e alta variância, em que o modelo é demasiadamente complexo em relação à quantidade ou à variabilidade dos dados disponíveis.

O equilíbrio entre viés e variância é essencial para alcançar um bom desempenho preditivo. Estratégias como regularização, validação cruzada e aumento do tamanho do conjunto de treinamento são comumente utilizadas para mitigar o *overfitting* e promover uma melhor generalização.

2.3.2 Algoritmo K-Nearest Neighbors (KNN)

O algoritmo *K-Nearest Neighbors* (KNN), como descrito por Mitchell (1997), é um método de aprendizado supervisionado que se baseia na suposição fundamental de que instâncias com características semelhantes tendem a apresentar comportamentos ou rótulos semelhantes. Trata-se de um algoritmo de natureza preguiçosa (*lazy learning*), pois não realiza uma fase de treinamento explícita. Em vez disso, o modelo retém todo o

conjunto de treinamento e realiza a generalização apenas no momento da predição.

Dado um conjunto de treinamento $\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_m, y_m)\}$, onde cada $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{in}) \in \mathbb{R}^n$ representa um vetor com n características (atributos) do i-ésimo exemplo, e y_i é o valor de saída correspondente (um rótulo de classe na classificação ou um valor numérico na regressão). Para um novo exemplo $\mathbf{x}$ (cuja saída se deseja prever), o algoritmo KNN realiza os seguintes passos:

1. Calcula a distância entre $\mathbf{x}$ e cada $\mathbf{x}_i$ do conjunto de treinamento.
2. Seleciona os k exemplos mais próximos (os k vizinhos) segundo a distância calculada.
3. Para classificação, o rótulo predito $\hat{y}$ é a classe que aparece com maior frequência entre esses k vizinhos. Para regressão, $\hat{y}$ é a média aritmética dos valores y_i dos k vizinhos.

A medida de distância mais comum é a **distância euclidiana**, definida como

$$d(\mathbf{x}, \mathbf{x}_i) = \sqrt{\sum_{j=1}^{n} (x_j - x_{ij})^2},$$

onde:

- x_j é o valor da j-ésima característica do novo exemplo $\mathbf{x}$;
- x_{ij} é o valor da j-ésima característica do exemplo de treinamento $\mathbf{x}_i$;
- n é o número total de características.

Essa fórmula calcula a raiz quadrada da soma dos quadrados das diferenças entre as coordenadas dos dois pontos, correspondendo à distância geométrica em um espaço de n dimensões.

Uma generalização é a **distância de Minkowski**

$$d(\mathbf{x}, \mathbf{x}_i) = \left(\sum_{j=1}^{n} |x_j - x_{ij}|^p \right)^{1/p},$$

onde p é um parâmetro que controla a forma da métrica. Para $p = 1$ obtém-se a distância Manhattan (soma dos valores absolutos), e para $p = 2$ retorna-se a distância euclidiana.

O KNN é útil para problemas com fronteiras de decisão complexas, porém seu custo computacional na predição pode ser alto, pois requer o cálculo de distâncias para todos os exemplos de treinamento (MITCHELL, 1997).

2.3.3 Algoritmo de Árvore de Decisão

Uma árvore de decisão é um modelo preditivo que particiona recursivamente o espaço de características em regiões disjuntas, atribuindo a cada região um valor constante (para regressão) ou um rótulo de classe (para classificação). Cada partição é definida por um teste em um único atributo, geralmente da forma $x_j \leq t$ (para variáveis contínuas) ou $x_j \in S$ (para variáveis categóricas). O processo de construção é realizado da seguinte forma: a cada nó, seleciona-se o atributo e o ponto de corte que maximizam a redução de uma medida de impureza (BREIMAN et al., 1984).

Durante o processo de construção da árvore, a escolha do atributo ideal e de seu respectivo ponto de corte em cada nó fundamenta-se na aplicação das chamadas medidas de impureza (BREIMAN et al., 1984). Essas medidas funcionam como métricas estatísticas para quantificar o nível de heterogeneidade ou mistura dos dados dentro de cada subconjunto gerado pelo particionamento. Do ponto de vista conceitual, um nó é considerado "puro" (impureza igual a zero) se todos os elementos contidos nele pertencerem a uma única classe, eliminando qualquer incerteza na predição. Por outro lado, a impureza atinge seu valor máximo quando as classes estão distribuídas de forma perfeitamente uniforme, representando o cenário de máxima desordem e imprevisibilidade dos dados. Para modelar matematicamente esse comportamento em um conjunto D composto por c classes, onde p_i representa a proporção de exemplos da classe i, a literatura adota predominantemente duas funções de impureza:

- **Índice de Gini**:
$$Gini(D) = 1 - \sum_{i=1}^{c} p_i^2.$$
Essa medida varia entre 0 (quando todos os exemplos pertencem a uma única classe) e $1 - 1/c$ (distribuição uniforme). Quanto menor o Gini, mais puro é o nó.

- **Entropia**:
$$Ent(D) = -\sum_{i=1}^{c} p_i \log_2 p_i.$$
A entropia mede a incerteza ou desordem. Seu valor máximo é $\log_2 c$ (todos os p_i iguais) e mínimo é 0 (pureza total).

Ao dividir D em dois subconjuntos D_{esq} e D_{dir} (por exemplo, valores $\leq t$ e $> t$), a redução da impureza é dada por

$$\Delta = Imp(D) - \frac{|D_{\text{esq}}|}{|D|} Imp(D_{\text{esq}}) - \frac{|D_{\text{dir}}|}{|D|} Imp(D_{\text{dir}}),$$

onde Imp pode ser o índice de Gini ou a entropia, e $|D|$ representa o número de exemplos no conjunto. O algoritmo escolhe o teste que maximiza Δ.

Para evitar *overfitting*, o crescimento da árvore é limitado por critérios como profundidade máxima, número mínimo de amostras em um nó para permitir uma divisão, ou número mínimo de amostras em uma folha. Além disso, técnicas de poda (remoção de nós que não trazem melhoria significativa) podem ser aplicadas após a construção, utilizando um conjunto de validação (BREIMAN, 2001).

2.3.4 Algoritmo Random Forest

O algoritmo *Random Forest* é um método de *ensemble* (combinação de múltiplos modelos) proposto por Breiman (2001). Ele constrói um grande número de árvores de decisão e combina suas predições: para classificação, por votação majoritária; para regressão, pela média dos valores previstos. A aleatoriedade introduzida no processo reduz a correlação entre as árvores e melhora a capacidade de generalização, evitando o *Overfit*.

Cada árvore na floresta é treinada a partir de duas fontes independentes de aleatoriedade:

- *Bagging (bootstrap aggregating)*: uma amostra com reposição do conjunto original, de mesmo tamanho, é retirada para cada árvore. Cerca de 2/3 dos exemplos são repetidos, e 1/3 (chamados *out-of-bag*) ficam de fora. Essa técnica reduz a variância do modelo final.
- **Seleção aleatória de atributos**: em cada nó da árvore, apenas um subconjunto de m atributos (escolhido aleatoriamente) é considerado para a melhor divisão, onde m é tipicamente muito menor que o número total de atributos M. Isso garante que as árvores sejam diversas, reduzindo a correlação entre elas.

Para cada árvore, o processo de construção é o seguinte:

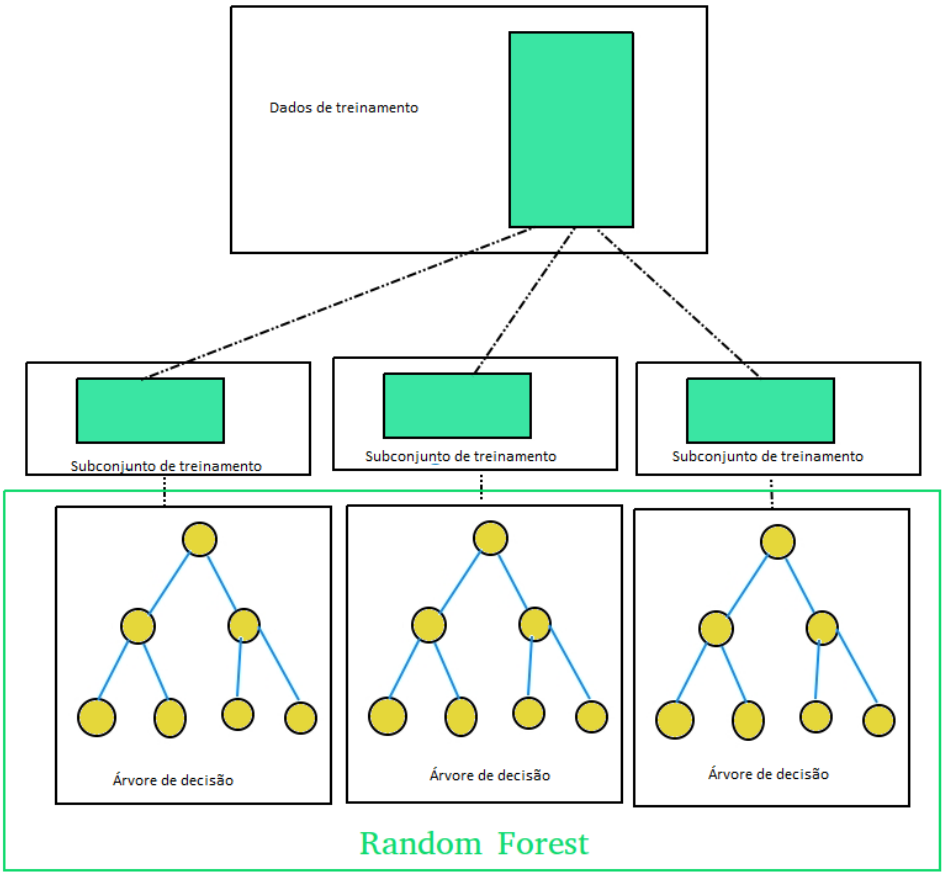
1. Gere uma amostra *bootstrap* a partir do conjunto de treinamento.
2. Inicie com um único nó contendo todos os exemplos da amostra.
3. Para cada nó não terminal:
    - Selecione aleatoriamente m atributos dentre os M disponíveis.
    - Para cada um desses m atributos, avalie todos os possíveis pontos de corte (valores que separam os exemplos em dois grupos) e calcule a redução de impureza (classificação) ou a redução do erro quadrático (regressão).
    - Escolha o atributo e o ponto de corte que maximizam a redução.
    - Particione o nó em dois filhos (esquerdo e direito) com base nesse teste.

4. Continue recursivamente até que um critério de parada seja atingido (profundidade máxima, número mínimo de exemplos por folha, ou nenhuma redução significativa).

5. Não realize poda; as árvores são mantidas profundas (alta variância individual) para que o conjunto tenha baixo viés.

A Figura 1 ilustra de forma esquemática esse processo de construção do *Random Forest*, evidenciando como múltiplas árvores de decisão são geradas a partir de diferentes subconjuntos de treinamento (obtidos via *bootstrap*) e como suas predições são posteriormente combinadas para formar o modelo final.

Figura 1 — Representação esquemática do algoritmo Random Forest, mostrando a geração de múltiplas árvores de decisão a partir de diferentes subconjuntos de treinamento (*bootstrap*) e a combinação de suas predições



Fonte: O autor

E assim, para um novo exemplo $\mathbf{x}$, temos:

- **Classificação**: cada árvore $h(\mathbf{x}, \Theta_k)$ produz um rótulo de classe. A predição final $\hat{y}$ é a classe que recebe a maioria dos votos

$$\hat{y} = \arg\max_{c} \sum_{k=1}^{K} \mathbf{1}\{h(\mathbf{x}, \Theta_k) = c\},$$

onde $\mathbf{1}\{\cdot\}$ é a função indicadora (vale 1 se a condição for verdadeira, 0 caso contrário) e K é o número total de árvores.

- **Regressão**: a predição final é a média das saídas das árvores

$$\hat{y} = \frac{1}{K} \sum_{k=1}^{K} h(\mathbf{x}, \Theta_k).$$

Agora, seja Θ_k o vetor aleatório independente que controla a amostragem bootstrap e a seleção de atributos para a k-ésima árvore. Para um número grande de árvores ($K \to \infty$), o erro de generalização do Random Forest (probabilidade de classificação incorreta) converge para

$$PE^* = P_{\mathbf{X},Y} \left(P_\Theta(h(\mathbf{X}, \Theta) = Y) - \max_{j \neq Y} P_\Theta(h(\mathbf{X}, \Theta) = j) < 0 \right),$$

onde:

- $\mathbf{X}, Y$ representam um par (entrada, rótulo verdadeiro) extraído da distribuição real dos dados.
- P_Θ denota a probabilidade sobre a geração de uma árvore (ou seja, sobre a aleatoriedade da amostra bootstrap e da seleção de atributos).
- A expressão dentro da desigualdade é chamada de **margem**: ela mede o quanto a probabilidade de acerto supera a probabilidade da classe concorrente mais forte. Se a margem é positiva, a floresta acerta; se negativa, erra.

Esse resultado, provado por Breiman (2001), garante que o Random Forest não sofre *overfitting* à medida que mais árvores são adicionadas, pois o erro converge para um valor limite.

O desempenho do Random Forest depende de dois fatores fundamentais:

- **Força (*strength*)** (s): é o valor esperado da margem sobre a distribuição dos dados:

$$s = \mathbb{E}_{\mathbf{X},Y} \left[P_\Theta(h(\mathbf{X}, \Theta) = Y) - \max_{j \neq Y} P_\Theta(h(\mathbf{X}, \Theta) = j) \right].$$

  Intuitivamente, mede o quão confiáveis são as predições individuais das árvores. Quanto maior s, melhor.

- **Correlação média** ($\bar{\rho}$): é a correlação entre as saídas de duas árvores diferentes (mais precisamente, entre suas funções de margem bruta). Valores altos de correlação significam que as árvores cometem erros semelhantes, reduzindo o ganho do conjunto (*ensemble*).

Breiman (2001) derivou uma cota superior para o erro de generalização

$$PE^* \leq \frac{\bar{\rho}\,(1 - s^2)}{s^2}.$$

Essa desigualdade mostra que, para obter alta acurácia, é necessário que:

- As árvores individuais sejam fortes (s próximo de 1);
- A correlação entre elas seja baixa ($\bar{\rho}$ pequeno).

O Random Forest equilibra esses dois objetivos por meio da seleção aleatória de atributos: m pequeno reduz a correlação, mas também pode reduzir a força. O valor ideal de m depende do problema e pode ser ajustado .

Uma propriedade importante do Random Forest é a possibilidade de estimar o erro de generalização, a força e a correlação sem necessidade de um conjunto de validação separado. Como cada árvore é treinada com cerca de 2/3 dos exemplos, os 1/3 restantes (chamados *out-of-bag* – OOB) formam um conjunto de teste natural para aquela árvore. Para cada exemplo do conjunto de treinamento, calcula-se a predição agregada apenas das árvores nas quais ele não apareceu (ou seja, que o consideram *out-of-bag*). A acurácia dessa predição agregada é a **estimativa *out-of-bag* do erro de generalização**, que tende a ser muito próxima daquela obtida com um conjunto de teste independente (BREIMAN, 2001).

Além disso, as estimativas *out-of-bag* podem ser usadas para medir a importância de cada variável: ao permutar aleatoriamente os valores de uma variável nos exemplos *out-of-bag* e medir o aumento no erro, obtém-se uma métrica de quão essencial aquela variável é para a predição.

2.3.5 Algoritmo XGBoost

O *XGBoost* é uma implementação de alto desempenho do algoritmo de *gradient boosting*, proposta por Chen e Guestrin (2016). Diferentemente do Random Forest, que constrói árvores independentemente, o *boosting* adiciona árvores sequencialmente, onde cada nova árvore tenta corrigir os erros residuais (diferenças entre a predição atual e o valor real) do conjunto de árvores anteriores. O XGBoost se destaca por sua escalabilidade, eficiência computacional e uso de regularização para evitar *overfit*.

Dado um conjunto de treinamento $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}$ com n exemplos, o XGBoost constrói a predição como a soma de K árvores

$$\hat{y}_i = \sum_{k=1}^{K} f_k(\mathbf{x}_i), \quad f_k \in \mathcal{F},$$

onde $\mathcal{F}$ é o espaço das árvores de regressão . Cada f_k é uma árvore com estrutura própria, que associa uma pontuação a cada folha.

Para aprender o conjunto de funções $\{f_1, \ldots, f_K\}$, minimiza-se a seguinte função objetivo regularizada

$$\mathcal{L} = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k),$$

onde:

- $l(y_i, \hat{y}_i)$ é uma função de perda diferenciável que mede o erro entre o valor real y_i e a predição $\hat{y}_i$. Para regressão, pode ser o erro quadrático $l = (y_i - \hat{y}_i)^2$; para classificação binária, a entropia cruzada logística.
- $\Omega(f) = \gamma T + \frac{1}{2}\lambda \|\mathbf{w}\|^2$ é o termo de regularização para uma árvore f.
  - T: número de folhas na árvore.
  - $\mathbf{w} = (w_1, \ldots, w_T)$: vetor de pesos, ou seja, a pontuação associada a cada folha.
  - γ: parâmetro que penaliza o número de folhas (quanto maior γ, mais simples é a árvore, pois adicionar uma folha custa γ).
  - λ: parâmetro de regularização sobre os pesos das folhas, que evita que os pesos cresçam excessivamente.

A inclusão da regularização é uma das principais inovações do *XGBoost* em relação aos métodos de *boosting* anteriores, que geralmente não penalizavam a complexidade das árvores.

Conforme dito anteriormente, o modelo é treinado de forma aditiva. Portanto no passo t (adicionando a t-ésima árvore), a predição atual é $\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(\mathbf{x}_i)$. A função objetivo para esse passo é

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(\mathbf{x}_i)\right) + \Omega(f_t) + \text{constante}.$$

Conforme Chen e Guestrin (2016), para otimizar rapidamente, expande-se a perda em série de Taylor de segunda ordem em torno de $\hat{y}_i^{(t-1)}$:

$$l(y_i, \hat{y}_i^{(t-1)} + f_t) \approx l(y_i, \hat{y}_i^{(t-1)}) + g_i f_t + \frac{1}{2} h_i f_t^2,$$

onde:

$$g_i = \frac{\partial l(y_i, \hat{y}_i^{(t-1)})}{\partial \hat{y}_i^{(t-1)}}, \qquad h_i = \frac{\partial^2 l(y_i, \hat{y}_i^{(t-1)})}{\partial (\hat{y}_i^{(t-1)})^2}$$

são a primeira e a segunda derivadas da perda em relação à predição anterior. Essas derivadas dependem da escolha da função de perda; por exemplo, para o erro quadrático, $g_i = 2(\hat{y}_i^{(t-1)} - y_i)$ e $h_i = 2$ (constante).

Substituindo a expansão e removendo os termos constantes, a função objetivo a ser minimizada no passo t torna-se

$$\tilde{\mathcal{L}}^{(t)} = \sum_{i=1}^{n} \left[g_i f_t(\mathbf{x}_i) + \frac{1}{2} h_i f_t^2(\mathbf{x}_i) \right] + \Omega(f_t).$$

Para uma dada estrutura de árvore q (que mapeia cada exemplo $\mathbf{x}_i$ a uma folha), seja $I_j = \{i \mid q(\mathbf{x}_i) = j\}$ o conjunto de exemplos que caem na folha j. A função $f_t(\mathbf{x}_i)$ assume o valor w_j (peso da folha) para todos os $i \in I_j$. Então

$$\tilde{\mathcal{L}}^{(t)} = \sum_{j=1}^{T} \left[\left(\sum_{i \in I_j} g_i \right) w_j + \frac{1}{2} \left(\sum_{i \in I_j} h_i + \lambda \right) w_j^2 \right] + \gamma T.$$

Para uma estrutura fixa, essa expressão é uma função quadrática em w_j. Derivando em relação a w_j e igualando a zero, obtém-se o peso ótimo para cada folha

$$w_j^* = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda}.$$

Substituindo w_j^* de volta na expressão de $\tilde{\mathcal{L}}^{(t)}$, obtém-se o valor ótimo da função objetivo para aquela estrutura

$$\tilde{\mathcal{L}}^{(t)}(q) = -\frac{1}{2} \sum_{j=1}^{T} \frac{\left(\sum_{i \in I_j} g_i \right)^2}{\sum_{i \in I_j} h_i + \lambda} + \gamma T.$$

Essa equação é usada como **pontuação de qualidade** da estrutura da árvore: quanto menor o valor, melhor (CHEN e GUESTRIN, 2016).

Para decidir se um nó deve ser dividido em dois filhos, calcula-se a redução na função objetivo proporcionada pela divisão. Sejam $I = I_L \cup I_R$ o conjunto de exemplos do nó original, e I_L e I_R os conjuntos após a divisão. O ganho é

$$\mathcal{G} = \frac{1}{2} \left[\frac{(\sum_{i \in I_L} g_i)^2}{\sum_{i \in I_L} h_i + \lambda} + \frac{(\sum_{i \in I_R} g_i)^2}{\sum_{i \in I_R} h_i + \lambda} - \frac{(\sum_{i \in I} g_i)^2}{\sum_{i \in I} h_i + \lambda} \right] - \gamma.$$

O termo γ atua como uma penalidade por adicionar dois novos nós (um para cada filho) em substituição a um nó folha. Apenas divisões com $\mathcal{G} > 0$ são aceitas; caso contrário, o nó se torna uma folha. Esse mecanismo de poda prévia controla o crescimento excessivo da árvore (CHEN e GUESTRIN, 2016).

A eficácia do XGBoost decorre da combinação do uso de regularização (controle de *overfitting*) e aproximação de segunda ordem (convergência mais rápida e uso de funções de perda arbitrárias).

2.3.6 Redes Neurais Artificiais

As redes neurais artificiais (RNAs) constituem uma das principais áreas de estudo dentro do campo do aprendizado de máquina e da inteligência artificial. Elas são

sistemas computacionais inspirados na estrutura e no funcionamento do cérebro humano, compostos por unidades de processamento simples, denominadas neurônios artificiais, organizadas em camadas interconectadas. O objetivo central das RNAs é modelar relações complexas entre variáveis de entrada e saída, por meio de um processo de aprendizado baseado em dados (LUDERMIR, 2021).

Historicamente, o conceito de neurônio artificial surgiu com o trabalho de Warren McCulloch e Walter Pitts (1943), que propuseram um modelo matemático simplificado capaz de representar o comportamento lógico de um neurônio biológico. O modelo, conhecido como neurônio de McCulloch-Pitts, utilizava uma combinação linear de entradas ponderadas e uma função limiar para determinar a ativação da unidade (MCCULLOCH e PITTS, 1943). Essa formulação constituiu a base teórica para o desenvolvimento posterior das redes neurais.

Em 1958, Frank Rosenblatt introduziu o Perceptron, um modelo de rede neural capaz de aprender por meio de um algoritmo de ajuste de pesos baseado em erro (ROSENBLATT, 1958). O Perceptron marcou o início da aplicação prática das redes neurais em problemas de reconhecimento de padrões. Contudo, sua limitação em resolver problemas não linearmente separáveis foi evidenciada por Minsky e Papert (1969), o que resultou em um período de estagnação na pesquisa de redes neurais (MITCHELL, 1997).

A retomada desse campo ocorreu na década de 1980, com o desenvolvimento do algoritmo de retropropagação do erro (*backpropagation*) por Rumelhart, Hinton e Williams (1986), que permitiu o treinamento eficiente de redes com múltiplas camadas, as chamadas redes neurais multicamadas (MLP - *Multi-Layer Perceptron*). Esse avanço possibilitou às RNAs resolverem problemas não lineares e impulsionou sua aplicação em diversas áreas, como processamento de sinais, visão computacional e previsão de séries temporais (BISHOP, 2006).

No final da mesma década, um marco teórico consolidou a relevância matemática das redes neurais: o Teorema de Aproximação Universal, proposto por George Cybenko (1989). Esse teorema demonstrou que uma rede neural com apenas uma camada oculta, contendo um número finito de neurônios e uma função de ativação não linear, é capaz de aproximar qualquer função contínua definida em um conjunto compacto de $\mathbb{R}^n$ com precisão arbitrária.

Assim, o Teorema de Aproximação Universal forneceu a fundamentação teórica necessária para compreender por que redes neurais possuem alta capacidade de representação e generalização, explicando seu sucesso em tarefas complexas e pavimentando o caminho para o desenvolvimento das arquiteturas modernas de aprendizado profundo (*deep learning*).

O neurônio artificial (*Perceptron*) é um classificador linear que busca encontrar um hiperplano que separe as amostras de diferentes classes em um espaço de características.

Segundo Bishop (2006), para um vetor de entrada $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ e um vetor de

pesos $\mathbf{w} = (w_1, w_2, \ldots, w_n)$, ele calcula uma combinação linear acrescida de um viés b

$$z = \mathbf{w}^T\mathbf{x} + b = \sum_{j=1}^{n} w_j x_j + b.$$

O resultado z é então passado por uma função de ativação Φ para produzir a saída

$$\hat{y} = \Phi(z).$$

Para classificação binária, é comum usar a função degrau:

$$\Phi(z) = \begin{cases} 1, & \text{se } z \geq 0, \\ 0, & \text{caso contrário.} \end{cases}$$

O treinamento ajusta os pesos e o viés usando a regra de aprendizado

$$w_j^{(t+1)} = w_j^{(t)} + \eta(y_i - \hat{y}_i)x_{ij}, \quad b^{(t+1)} = b^{(t)} + \eta(y_i - \hat{y}_i),$$

onde η é a taxa de aprendizado (controla o tamanho do passo de ajuste) e $(y_i - \hat{y}_i)$ é o erro de classificação.

Embora eficiente em problemas linearmente separáveis, o *Perceptron* simples é incapaz de modelar relações não lineares. Como visto anteriormente essa limitação levou ao desenvolvimento do *Perceptron* Multicamadas (*Multilayer Perceptron* - MLP), introduzido por Rumelhart, Hinton e Williams (1986).

Com isso, o *Perceptron* Multicamadas consiste em uma rede de neurônios artificiais dispostos em camadas: uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída. Cada neurônio de uma camada l calcula uma transformação do tipo

$$a_j^{(l)} = \Phi\left(\sum_i w_{ji}^{(l)} a_i^{(l-1)} + b_j^{(l)}\right),$$

onde:

- $a_i^{(l-1)}$ é a saída do neurônio i da camada anterior,
- $w_{ji}^{(l)}$ é o peso da conexão entre o neurônio i (camada $l-1$) e o neurônio j (camada l),
- $b_j^{(l)}$ é o viés do neurônio j,
- Φ é uma função de ativação não linear.

A presença dessas funções não lineares permite que o MLP modele funções complexas, pois conforme já visto, o teorema da aproximação universal (CYBENKO, 1989) garante

que uma rede com uma única camada oculta e neurônios suficientes pode aproximar qualquer função contínua em um domínio compacto com precisão arbitrária.

O algoritmo de retropropagação do erro (Rumelhart, Hinton e Williams, 1986) é o método padrão para treinar MLPs. Seu objetivo é minimizar uma função de custo (ou perda) que mede a diferença entre as saídas preditas $\hat{y}_i$ e os valores reais y_i. Para regressão, uma função comum é o erro quadrático médio (MSE – *Mean Squared Error*)

$$J(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2,$$

onde n é o número de exemplos no conjunto de treinamento.

O algoritmo funciona em duas fases: *forward pass* (propagação para frente) e *backward pass* (propagação reversa). Conforme Bishop (2006), os dados de entrada são inseridos na rede e propagados camada por camada usando as equações do neurônio até que a saída $\hat{y}_i$ seja calculada. Em seguida, calcula-se o valor da função de custo J.

O objetivo é calcular o gradiente de J em relação a cada peso $w_{ji}^{(l)}$ e cada viés $b_j^{(l)}$, ou seja, $\dfrac{\partial J}{\partial w_{ji}^{(l)}}$ e $\dfrac{\partial J}{\partial b_j^{(l)}}$. Essas derivadas indicam a direção e a magnitude do ajuste necessário para reduzir o erro. O cálculo utiliza a regra da cadeia do cálculo diferencial.

Definindo o **erro local** (ou sensibilidade) do neurônio j na camada l como $\delta_j^{(l)} = \dfrac{\partial J}{\partial z_j^{(l)}}$, onde $z_j^{(l)} = \sum_i w_{ji}^{(l)} a_i^{(l-1)} + b_j^{(l)}$ é a entrada do neurônio antes da ativação. Para a camada de saída (L), o erro depende da derivada da função de custo e da derivada da função de ativação

$$\delta_j^{(L)} = \frac{\partial J}{\partial a_j^{(L)}} \cdot \Phi'(z_j^{(L)}).$$

Por exemplo, se a função de custo for o MSE e a ativação de saída for linear ($\Phi(z) = z$), então $\delta_j^{(L)} = (a_j^{(L)} - y_j)$.

Para as camadas ocultas (da camada $L-1$ até a primeira), o erro é propagado reversamente

$$\delta_j^{(l)} = \Phi'(z_j^{(l)}) \cdot \sum_k \delta_k^{(l+1)} w_{kj}^{(l+1)}.$$

Aqui, $\sum_k \delta_k^{(l+1)} w_{kj}^{(l+1)}$ representa a soma dos erros da camada seguinte $(l+1)$ ponderados pelos respectivos pesos das conexões que chegam ao neurônio j.

Com os erros $\delta_j^{(l)}$ calculados, os gradientes são obtidos por

$$\frac{\partial J}{\partial w_{ji}^{(l)}} = \delta_j^{(l)} a_i^{(l-1)}, \qquad \frac{\partial J}{\partial b_j^{(l)}} = \delta_j^{(l)}.$$

Os pesos e vieses são então ajustados na direção oposta ao gradiente (para diminuir o

custo) usando uma taxa de aprendizado η (hiperparâmetro)

$$w_{ji}^{(l)} \leftarrow w_{ji}^{(l)} - \eta \frac{\partial J}{\partial w_{ji}^{(l)}}, \qquad b_j^{(l)} \leftarrow b_j^{(l)} - \eta \frac{\partial J}{\partial b_j^{(l)}}.$$

Esse procedimento é repetido por várias **épocas** (passagens completas por todo o conjunto de treinamento) até que a função de custo atinja um valor aceitável ou não melhore mais.

A retropropagação pode ser entendida como um mecanismo de *"feedback"* que informa a cada neurônio o quanto ele contribuiu para o erro total da rede. Neurônios cujas saídas levaram a um erro grande têm seus pesos reduzidos; aqueles que contribuíram para acertos têm seus pesos reforçados. O processo é análogo ao ajuste fino de um sistema complexo onde cada componente ajusta sua influência com base no resultado final.

O MLP é capaz de aprender padrões complexos, mas requer cuidados para evitar *overfitting* (LUDERMIR, 2021).

2.4 Métricas estatísticas para avaliação dos modelos

A avaliação de desempenho é uma etapa fundamental no desenvolvimento de modelos de aprendizado de máquina, pois permite quantificar o grau de adequação entre as previsões do modelo e os valores reais observados. Conforme destacam Strauss, Villas Bôas Júnior e Ferreira (2022), a escolha adequada das métricas de avaliação é essencial, uma vez que cada métrica expressa um aspecto distinto do comportamento preditivo, podendo influenciar diretamente na interpretação da performance e na comparação entre diferentes algoritmos.

De modo geral, as métricas são definidas em função da diferença entre as previsões realizadas pelo modelo e os valores reais dos dados, podendo assumir diferentes formas conforme a natureza da tarefa, sendo elas classificação ou regressão.

2.4.1 Métricas para Regressão

Nos problemas de regressão, as saídas são valores contínuos, e o objetivo é minimizar o erro entre as previsões $\hat{y}_i$ e os valores reais y_i. Segundo Junior (2021) as principais métricas para regressão são:

- **Erro Médio Absoluto** (MAE): mede o erro médio em termos absolutos entre os valores observados e previstos,

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i|.$$

- **Erro Quadrático Médio** (MSE): representa a média dos quadrados das diferenças entre os valores reais e os previstos, penalizando mais fortemente grandes

desvios,

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2.$$

- **Raiz do Erro Quadrático Médio ($RMSE$)**: é a raiz quadrada do MSE e fornece uma estimativa na mesma unidade da variável dependente, facilitando a interpretação do erro,

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}.$$

- **Erro Percentual Absoluto Médio ($MAPE$)**: mede o erro médio percentual em relação ao valor real, permitindo comparar o desempenho de modelos em diferentes escalas,

$$MAPE = \frac{100}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|.$$

  É importante ressaltar que o $MAPE$ é sensível a valores de y_i próximos de zero, podendo gerar distorções em tais casos.

- **Coeficiente de Determinação (R^2)**: indica o quanto da variabilidade dos dados é explicada pelo modelo

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2},$$

  onde $\bar{y}$ representa a média dos valores observados. Um valor de R^2 próximo de 1 indica que o modelo explica bem a variabilidade dos dados, enquanto valores próximos de 0 indicam baixo poder explicativo.

Essas métricas, quando interpretadas em conjunto, permitem uma avaliação abrangente da performance dos modelos. Segundo Strauss, Villas Bôas Júnior e Ferreira (2022), a seleção apropriada das métricas deve estar alinhada ao objetivo do estudo e à natureza dos dados, evitando interpretações equivocadas e garantindo maior robustez às conclusões.

2.4.2 Métricas para Classificação

Em problemas de classificação, o objetivo é atribuir corretamente uma classe $y_i \in \{1, 2, ..., k\}$ a cada amostra de entrada x_i. As métricas de avaliação baseiam-se na matriz de confusão, que contabiliza as previsões corretas e incorretas para cada classe.

Para avaliar o desempenho de um modelo de classificação, utiliza-se uma estrutura denominada matriz de confusão, que organiza os resultados das previsões em categorias de acertos e erros.

- **Matriz de Confusão**: é uma tabela que permite comparar as previsões do modelo com os valores verdadeiros das classes. No caso binário (duas classes: positiva e negativa), ela é expressa da seguinte forma:

Tabela 1 — Matriz de Confusão Binária

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

Fonte: O autor

Cada célula representa uma contagem de ocorrências:

- TP (*True Positives*) - número de amostras corretamente classificadas como positivas;
- TN (*True Negative*) - número de amostras corretamente classificadas como negativas;
- FP (*False Positives*) - amostras negativas incorretamente classificadas como positivas;
- FN (*False Negatives*) - amostras positivas incorretamente classificadas como negativas.

Matematicamente, se y_i é o rótulo verdadeiro e $\hat{y}_i$ é a previsão do modelo, então

$$TP = \sum_{i=1}^{N} [y_i = 1 \wedge \hat{y}_i = 1], \quad TN = \sum_{i=1}^{N} [y_i = 0 \wedge \hat{y}_i = 0],$$

$$FP = \sum_{i=1}^{N} [y_i = 0 \wedge \hat{y}_i = 1], \quad FN = \sum_{i=1}^{N} [y_i = 1 \wedge \hat{y}_i = 0],$$

onde o resultado vale 1 se a condição é verdadeira e 0 caso contrário.

A matriz de confusão é a base para o cálculo de diversas métricas de avaliação, pois descreve quantitativamente a natureza dos erros cometidos pelo modelo.

A seguir, apresentam-se as principais métricas derivadas da matriz de confusão, segundo Pádua (2020):

- **Acurácia** (Acc): mede a proporção de previsões corretas entre todas as amostras avaliadas, e é definida por

$$Acc = \frac{TP + TN}{TP + TN + FP + FN}$$

- **Precisão** (P): representa a fração das amostras classificadas como positivas que realmente pertencem à classe positiva, fornecida por

$$P = \frac{TP}{TP + FP}$$

- **Recall ou Sensibilidade** (R): mede a capacidade do modelo de identificar corretamente todos os exemplos positivos, de acordo com a equação

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

- **Medida F1** (F_1): combina precisão e recall por meio de sua média harmônica, e é denominada por

$$F_1 = 2 \cdot \frac{P \cdot R}{P + R}.$$

Conforme ressaltam Strauss, Villas Bôas Júnior e Ferreira (2022), a utilização de múltiplas métricas é essencial, uma vez que nenhuma métrica isolada é capaz de capturar todos os aspectos do desempenho de um modelo de classificação. A combinação da matriz de confusão com medidas derivadas, como precisão, recall e F1, permite uma avaliação mais abrangente e informada da performance preditiva.

2.5 Revisão Bibliográfica

Com o objetivo de contextualizar o presente estudo no panorama atual da aplicação de técnicas de aprendizado de máquina à teoria dos números, esta subseção apresenta uma revisão integrada de dois trabalhos representativos que abordam a identificação de números primos e a fatoração de inteiros por meio de modelos computacionais. A seleção abrange diversos métodos, permitindo uma análise comparativa das abordagens e de seus resultados.

2.5.1 Singh (2025): análise comparativa de modelos para classificação de primos

Singh (2025) realizou um estudo abrangente comparando sete algoritmos de aprendizado supervisionado, sendo eles: Regressão Logística, Árvore de Decisão, *Random Forest*, SVM (*Support Vector Machine*), KNN, *XGBoost* e *LightGBM*. Na tarefa de classificação binária de números primos em oito intervalos numéricos, variando de $2 - 97$ até $2 - 10^8$. Os resultados reportados indicam que o KNN obteve a maior acurácia média $(87,96\%)$, enquanto *LightGBM* apresentou o melhor equilíbrio entre acurácia e tempo de treinamento, especialmente nos maiores intervalos. Para o intervalo $2 - 10^8$, *LightGBM* atingiu acurácia de $96,69\%$ com apenas $13,2$ segundos de treinamento.

No entanto, é importante ressaltar que o estudo não apresenta matrizes de confusão para os modelos avaliados. A ausência dessa ferramenta dificulta a análise detalhada do comportamento dos classificadores, particularmente em relação ao desbalanceamento inerente entre as classes (números compostos são muito mais frequentes que primos). Sem a matriz de confusão, não é possível inferir com exatidão se as altas acurácias reportadas refletem uma real capacidade de identificação dos primos ou se são influenciadas pela predição majoritária da classe composta. Assim, embora os números de acurácia sejam expressivos, a falta de transparência sobre os erros cometidos limita a interpretação dos resultados.

2.5.2 Blake (2023): evidências de padrões ocultos na estrutura de semiprimos

Blake (2023) propôs uma abordagem para fatoração de semiprimos combinando aprendizado profundo com algumas técnicas oriundas da Teoria dos Números.

A motivação do estudo partia da intuição de que os semiprimos, por serem produtos de dois primos aleatórios, apresentariam uma estrutura tão imprevisível que nenhum classificador seria capaz de extrair informação útil, esperava-se um desempenho equivalente a um lançamento de moeda (50% de acurácia). Contrariando essa expectativa, o modelo alcançou 72% de acurácia fora da amostra, revelando que mesmo em dados aparentemente aleatórios há correlações que podem ser capturadas por redes neurais. A matriz de confusão mostrou que o modelo raramente cometia falsos negativos (apenas $5,6\%$ dos casos), mas apresentava uma taxa de falsos positivos de $22,7\%$. O resultado surpreendente sugere que, mesmo quando a intuição aponta para completa aleatoriedade, técnicas de aprendizado de máquina podem identificar padrões sutis, incentivando a exploração de problemas matemáticos sob uma perspectiva computacional.

2.5.3 Síntese e contribuições para este trabalho

Os dois estudos analisados oferecem perspectivas complementares sobre o uso de aprendizado de máquina na identificação de números primos. Singh (2025) fornece

uma visão ampla do desempenho de diferentes algoritmos em variados intervalos, mas deixa lacunas na análise de erros. Blake (2023), por sua vez, demonstra que mesmo modelos simples podem capturar informações não triviais sobre a estrutura de semiprimos, desafiando a intuição de que os dados são completamente aleatórios.

Para o presente trabalho, essas referências reforçam a importância de:

- Incluir matrizes de confusão para uma avaliação completa dos modelos.
- Considerar a possibilidade de que estudos relacionados aos números primos possam encontrar informações relevantes, mesmo com eles sendo aparentemente aleatórios.

Dessa forma, a presente monografia busca incorporar esses aprendizados, adotando uma metodologia que privilegia tanto a abrangência da avaliação quanto a profundidade da análise dos resultados.

3 METODOLOGIA

Nesta seção são descritos os procedimentos metodológicos adotados para a construção das bases de dados, seleção de atributos, definição dos modelos de aprendizado de máquina e avaliação de desempenho.

3.1 Base de dados

Os dados utilizados neste trabalho foram obtidos a partir do conjunto de dados disponível na plataforma Kaggle, sob a autoria de Charles Averill[1]. Esse conjunto contém os primeiros 999.999 números primos, estruturados nas seguintes colunas:

- **Rank**: posição do número primo na sequência (ex.: o primeiro primo é o 2, o segundo é o 3, e assim sucessivamente);
- **Num**: valor do número primo;
- **Interval**: diferença entre o primo atual e o primo anterior.

A partir dessa base, foram construídos dois conjuntos distintos, destinados às tarefas de regressão e classificação.

3.1.1 Conjunto para regressão

Para a tarefa de regressão, objetivou-se prever o valor do número primo com base em características posicionais e aritméticas. Utilizaram-se as seguintes variáveis preditoras:

- **Rank**: posição do primo na sequência;
- **Interval**: diferença entre o primo atual e o anterior;
- **Z4**: classe de equivalência do número primo módulo 4, definida como 1 se o primo é congruente a 1 módulo 4 e 0 se é congruente a 3 módulo 4 (primos maiores que 2 são ímpares, portanto só podem assumir esses dois valores).

A variável alvo (*label*) é o próprio número primo, representado pela coluna **Num**.

3.1.2 Conjunto para classificação

Para a tarefa de classificação, buscou-se distinguir números primos de números compostos. A partir da lista de primos, foi gerado um conjunto contendo números naturais até 999.999. Para cada número, foram computadas as seguintes variáveis:

[1]Disponível em: ⟨https://www.kaggle.com/datasets/charlesaverill/prime-numbers⟩. Acesso em: 16 mar. 2026.

- **Número**: o valor inteiro considerado;
- $\Omega(n)$: função aritmética que representa o número total de fatores primos de n, contados com multiplicidade. Conforme já definido, se $n = p_1^{\alpha_1} p_2^{\alpha_2} \dots p_k^{\alpha_k}$, então $\Omega(n) = \alpha_1 + \alpha_2 + \dots + \alpha_k$;
- **Primo**: variável alvo (*label*) binária indicadora, assumindo valor 1 se o número for primo e 0 caso contrário.

Para ambas as tarefas, regressão e classificação, os conjuntos de dados foram divididos em subconjuntos de treino e teste. O conjunto de treino foi utilizado para o ajuste dos modelos, enquanto o conjunto de teste foi reservado para a avaliação de desempenho, permitindo verificar a capacidade de generalização dos algoritmos em dados não vistos durante o treinamento.

Conforme estabelecido pelo Teorema dos Números Primos, a função contagem de primos $\pi(x)$, que denota o número de primos menores ou iguais a x, é assintoticamente equivalente a $\frac{x}{\ln x}$. Esse resultado clássico da Teoria dos Números fornece uma descrição aproximada da distribuição dos primos, mas não permite prever exatamente a posição de cada primo individualmente. Nesse contexto, a tarefa de regressão proposta neste trabalho busca justamente modelar o crescimento sequencial dos números primos, utilizando como variáveis preditoras a posição, o intervalo entre primos consecutivos e a classe módulo 4 ($\mathbb{Z}_4$). Por meio dessa abordagem, pretende-se investigar se é possível estimar o valor de um primo com base em suas características posicionais e aritméticas.

Paralelamente, a tarefa de classificação tem por objetivo discriminar, entre os números naturais, aqueles que são primos daqueles que são compostos, ou seja, separar o conjunto em dois subgrupos mutuamente exclusivos. Para tanto, emprega-se a função aritmética $\Omega(n)$ (número total de fatores primos com multiplicidade) como principal variável explicativa, juntamente com o próprio valor do número. Dessa forma, busca-se avaliar em que medida um modelo de aprendizado supervisionado é capaz de aprender a distinguir as duas classes a partir de informações sobre a composição multiplicativa dos inteiros.

Essas duas perspectivas: a modelagem da distribuição (regressão) e a distinção categórica (classificação), são complementares e exploram diferentes facetas dos números primos, permitindo uma análise mais abrangente da aplicabilidade de técnicas de aprendizado de máquina.

3.2 Modelos de Aprendizado de Máquina

Os modelos utilizados neste trabalho foram selecionados com o objetivo de comparar diferentes abordagens de aprendizado supervisionado. Os algoritmos empregados foram: *K-Nearest Neighbors* (KNN), *Random Forest*, *XGBoost* e *Perceptron* Multicamadas (MLP).

A fase inicial de experimentação e os primeiros treinamentos foram desenvolvidos como parte de atividades avaliativas realizadas durante as disciplinas de Inteligência Artificial cursadas ao longo da graduação.

Para o algoritmo KNN, foi realizada a análise do número ideal de vizinhos (k), buscando o melhor desempenho preditivo.

Nos modelos baseados em árvores, *Random Forest* e *XGBoost*, foram avaliados diferentes valores para o número de árvores e a profundidade máxima, com o objetivo de identificar a configuração mais adequada.

Para o modelo MLP, foram testadas diferentes arquiteturas, variando-se o número de camadas ocultas e a quantidade de neurônios em cada camada, visando encontrar a estrutura que melhor representasse os padrões presentes nos dados.

3.3 Métricas de Avaliação

Para avaliar o desempenho dos modelos nas tarefas de regressão e classificação, foram adotadas métricas estatísticas consolidadas na literatura, conforme recomendado por Strauss, Villas Bôas Júnior e Ferreira (2022). A escolha adequada das métricas é essencial para capturar diferentes aspectos do comportamento preditivo, especialmente em problemas com desbalanceamento de classes, como é o caso da classificação de números primos.

3.3.1 Métricas para Regressão

Na tarefa de regressão, objetivou-se prever o valor de um número primo com base em características posicionais e modulares. Foram utilizadas as seguintes métricas:

- **Erro Percentual Absoluto Médio (MAPE – *Mean Absolute Percentage Error*)**: mede o erro médio percentual entre os valores previstos $\hat{y}_i$ e os valores reais y_i. Conforme já visto, é definido por

$$\text{MAPE} = \frac{100}{n} \sum_{i=1}^{n} \left| \frac{y_i - \hat{y}_i}{y_i} \right|,$$

  onde n é o número de amostras no conjunto de teste. O MAPE é adimensional e permite comparar o desempenho em diferentes escalas. Valores menores indicam maior precisão. Ressalta-se novamente que o MAPE é sensível a valores de y_i muito próximos de zero, o que não ocorre neste estudo, pois os números primos são sempre maiores que 1.

- **Coeficiente de Determinação (R^2)**: indica a proporção da variabilidade dos

dados que é explicada pelo modelo. Conforme anteriormente definido, é dado por

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2},$$

onde $\bar{y}$ é a média dos valores reais. O R^2 varia de $-\infty$ a 1. Valores próximos de 1 indicam que o modelo se ajusta bem aos dados, e valores negativos indicam que o modelo é pior do que uma simples predição pela média.

3.3.2 Métricas para Classificação

Na tarefa de classificação, o objetivo foi distinguir números primos (classe positiva) de compostos (classe negativa). As métricas baseiam-se na matriz de confusão, que contabiliza os acertos e erros do modelo. Para um problema binário, definem-se:

- **Verdadeiros Positivos (TP)**: números primos corretamente classificados como primos.
- **Verdadeiros Negativos (TN)**: números compostos corretamente classificados como compostos.
- **Falsos Positivos (FP)**: números compostos incorretamente classificados como primos.
- **Falsos Negativos (FN)**: números primos incorretamente classificados como compostos.

A partir desses valores, calcula-se a **Acurácia (Acc)** que é a proporção de acertos em relação ao total de amostras

$$Acc = \frac{TP + TN}{TP + TN + FP + FN}.$$

Embora a métrica F_1 (média harmônica entre precisão e sensibilidade) seja comumente utilizada em problemas desbalanceados, os resultados experimentais mostraram que, na ausência da função $\Omega(n)$, a revocação para a classe primo foi sempre zero, tornando a métrica F_1 também zero. Por essa razão, optou-se por não incluí-lo nas tabelas de resultados, mantendo a acurácia e a matriz de confusão como principais ferramentas de análise.

4 ANÁLISE DOS RESULTADOS

Neste capítulo, são apresentados e discutidos os resultados obtidos na aplicação dos modelos de aprendizado de máquina para as tarefas de regressão e classificação, conforme a metodologia definida anteriormente.

4.1 Resultados da Tarefa de Regressão

Na tarefa de regressão, objetivou-se prever o valor de um número primo a partir de seu *rank*, do intervalo em relação ao primo anterior e da classe módulo 4 (Z_4). Os modelos foram treinados com os primeiros 749.999 primos e testados nos 250.000 primos restantes. A Tabela 2 apresenta os valores de Erro Percentual Absoluto Médio (MAPE) e Coeficiente de Determinação (R^2) para cada algoritmo.

Tabela 2 — Desempenho dos modelos de regressão no conjunto de teste

Algoritmo	**MAPE (%)**	R^2
KNN	14,56	-2,98
Random Forest	15,94	-3,54
XGBoost	14,74	-3,05
MLP	1,58	0,96

Fonte: O autor

Os algoritmos KNN, *Random Forest* e *XGBoost* apresentaram desempenho insatisfatório, com valores de R^2 negativos, indicando que os modelos são piores do que uma simples predição pela média dos valores observados. O alto MAPE (acima de 14%) reforça a incapacidade desses modelos em capturar o crescimento sequencial dos primos.

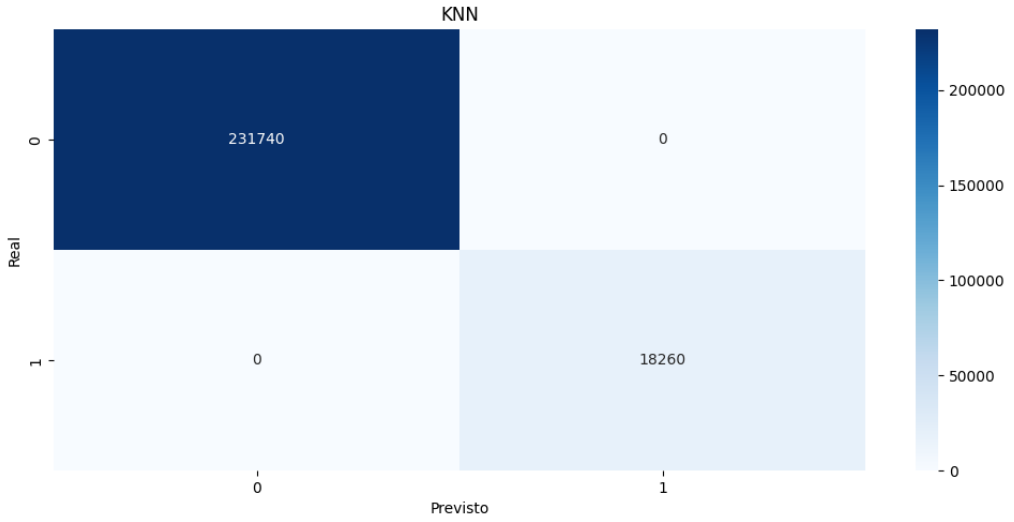
Em contraste, o MLP (*Perceptron* Multicamadas) obteve excelente desempenho, com $R^2 = 96\%$ e MAPE de apenas $1,58\%$. Esse resultado foi alcançado com apenas 53 épocas de treinamento, evidenciando rápida convergência. A arquitetura utilizada mostrou-se capaz de aprender a relação subjacente entre as variáveis preditoras e o valor do primo, superando significativamente os demais métodos. Esse desempenho superior pode ser atribuído ao Teorema de Aproximação Universal (CYBENKO, 1989), que confere às redes neurais a capacidade de aproximar funções contínuas complexas e não lineares de forma eficaz.

4.2 Resultados da Tarefa de Classificação

Na tarefa de classificação, o objetivo foi distinguir números primos de compostos. Inicialmente, utilizou-se o conjunto de números naturais de 2 a 999.999, com as variáveis *Número* e $\Omega(n)$ (função que conta o total de fatores primos com multiplicidade). A variável alvo era binária (1 para primo, 0 para composto). Os resultados são apresentados a seguir.

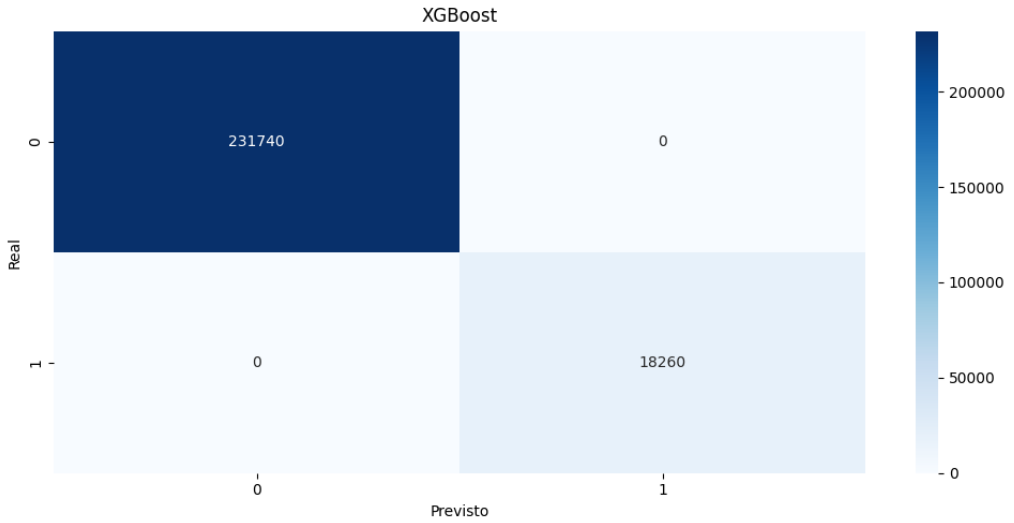
Quando $\Omega(n)$ foi incluída como *feature*, todos os algoritmos alcançaram 100% de acurácia no conjunto de teste (Figuras 2, 3 e 4), exceto o MLP, que obteve $92,7\%$. A matriz de confusão (Figura 5) revelou que o modelo classificou todas as amostras como *não primo* (classe 0), resultando em 100% de acertos para compostos, mas 0% de acertos para primos.

Figura 2 — Matriz de confusão do KNN, com a Feature Omega



Fonte: O autor

Figura 3 — Matriz de confusão do *XGBoost*, com a Feature Omega

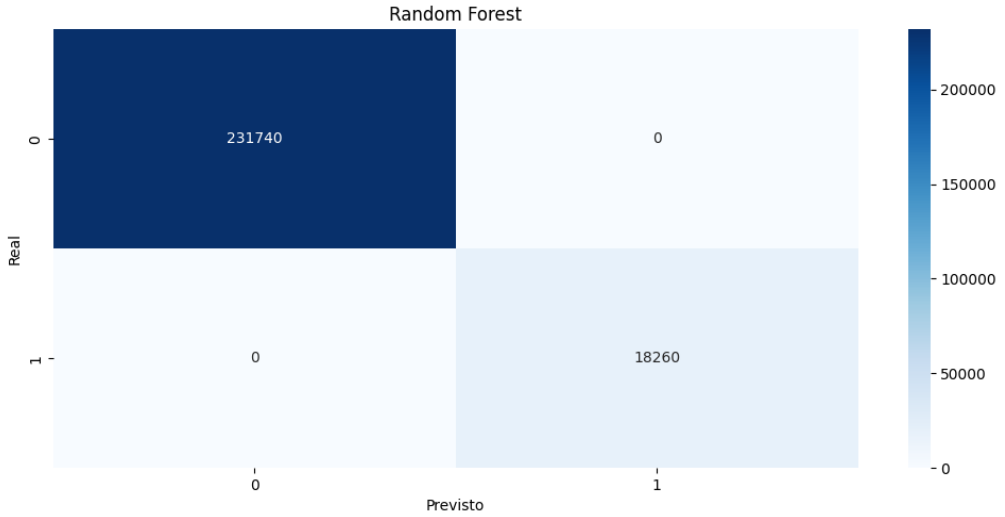


Fonte: O autor

A alta acurácia global ($92,7\%$) deve-se ao forte desbalanceamento entre as classes: aproximadamente $92,7\%$ dos números no intervalo considerado são compostos.

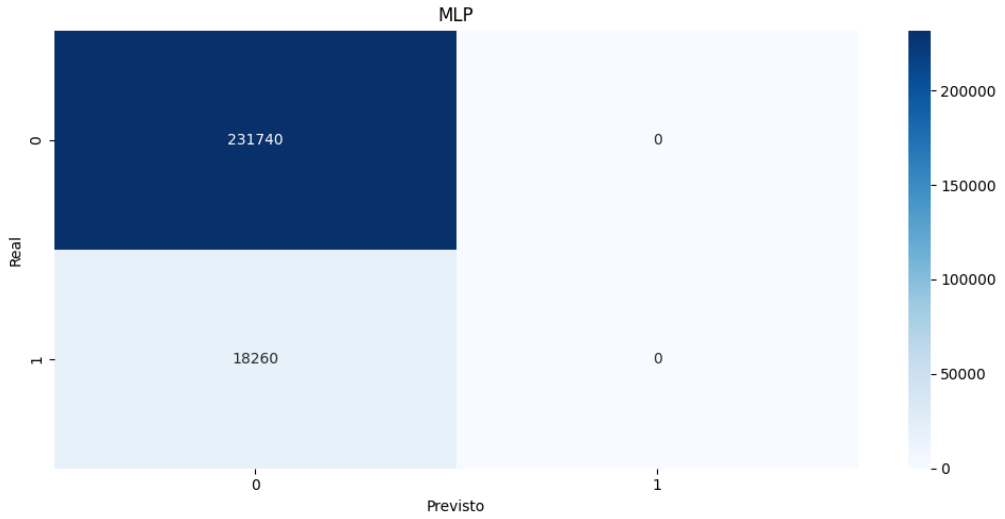
Assim, um modelo que simplesmente prevê sempre *não primo* atinge essa acurácia, mas é inútil para a identificação de primos.

Figura 4 — Matriz de confusão do *Random Forest*, com a Feature Omega



Fonte: O autor

Figura 5 — Matriz de confusão do MLP, com a Feature Omega



Fonte: O autor

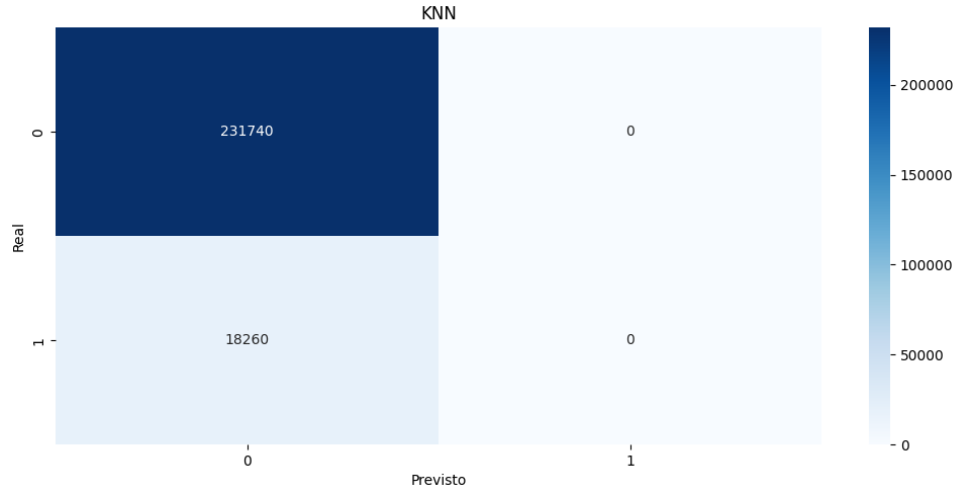
Para os demais algoritmos, a acurácia de 100% com a presença de $\Omega(n)$ ocorre porque a própria função $\Omega(n)$ resolve o problema: um número natural n é primo se e somente se $\Omega(n) = 1$.

Portanto, qualquer modelo que aprenda essa relação direta consegue classificar perfeitamente. Esse resultado, embora tecnicamente correto, não representa uma descoberta relevante, pois apenas reproduz uma propriedade aritmética conhecida.

Ao remover $\Omega(n)$ e manter apenas o valor do *Número* como variável preditora, todos os algoritmos (incluindo KNN, *Random Forest*, *XGBoost* e MLP) apresentaram acurácia de aproximadamente $92,7\%$, novamente devido à predição majoritária da classe composta. As matrizes de confusão (Figuras 6, 7, 8 e 9) ilustram esse comportamento.

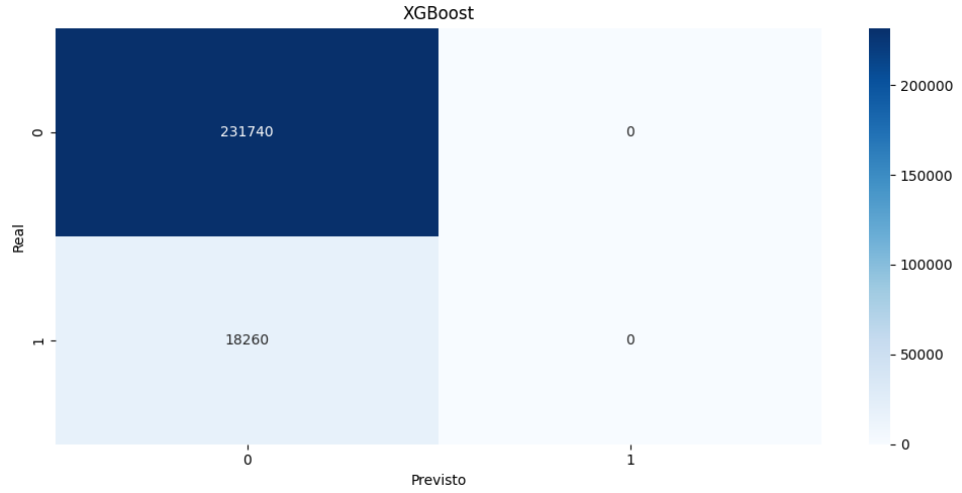
As matrizes mostram que nenhum primo foi corretamente classificado. A alta acurácia

Figura 6 — Matriz de confusão do KNN, sem a Feature Omega



Fonte: O autor

Figura 7 — Matriz de confusão *XGBoost*, sem a Feature Omega



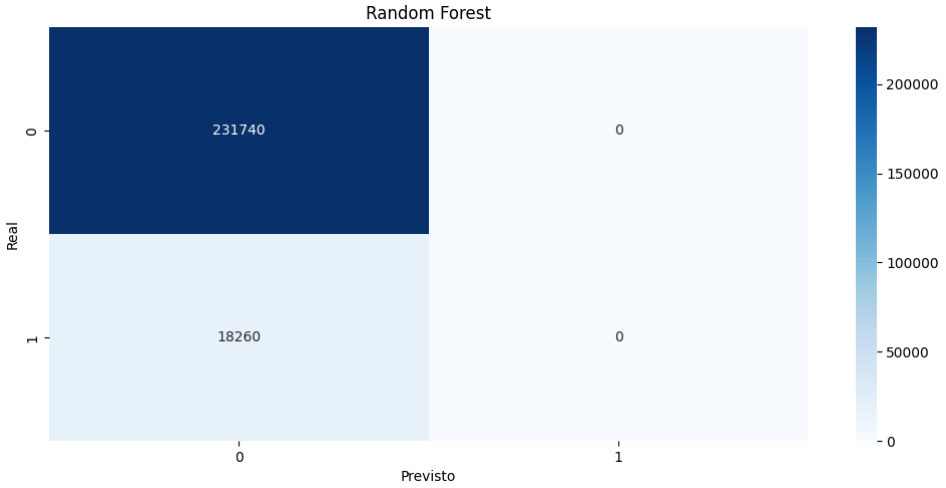
Fonte: O autor

global é enganosa e decorre apenas do desbalanceamento das classes. Esse resultado evidencia a necessidade de estratégias para lidar com conjuntos desbalanceados.

Como tentativa de reduzir o desbalanceamento, foram removidos do conjunto de dados todos os números múltiplos de 2, 3 e 5. Essa filtragem reduziu o total de amostras de 999.998 para 266.667 e alterou a proporção entre as classes para aproximadamente 72% de compostos e 28% de primos. Os experimentos foram repetidos sob as mesmas condições.

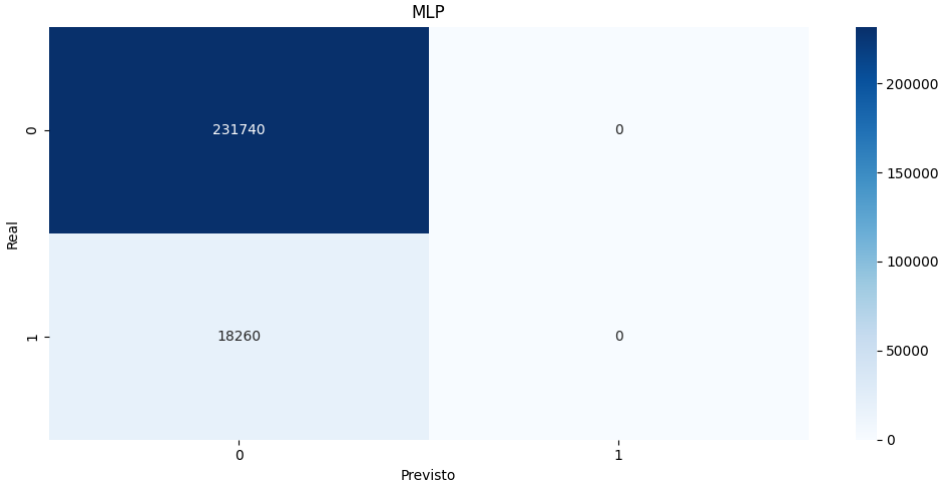
- **Com** $\Omega(n)$: conforme as matrizes de confusão (Figuras 10, 11, 12 e 13), os resultados se mantiveram inalterados: todos os algoritmos, exceto o MLP, alcançaram 100% de acurácia; o MLP obteve 72% de acurácia, novamente classificando todos os números como compostos.

Figura 8 — Matriz de confusão do *Random Forest*, sem a Feature Omega



Fonte: O autor

Figura 9 — Matriz de confusão do MLP, sem a Feature Omega



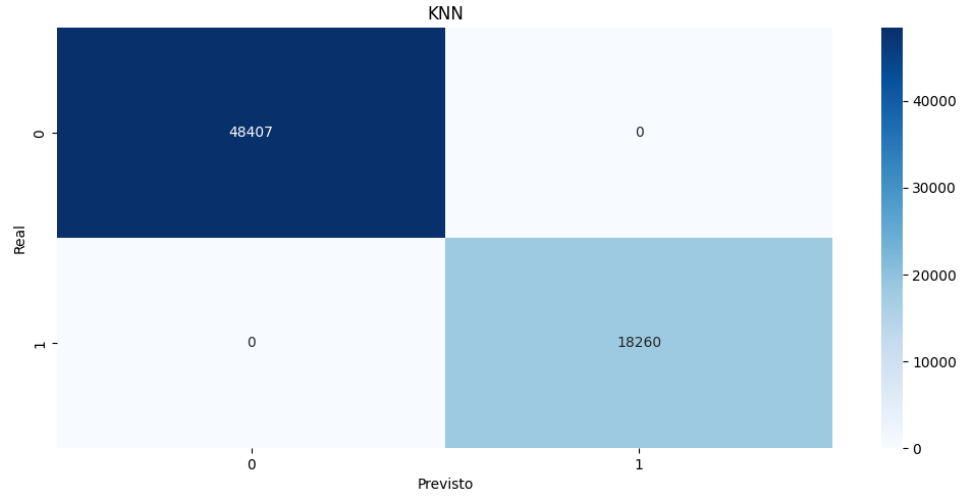
Fonte: O autor

- **Sem** $\Omega(n)$: todos os algoritmos apresentaram acurácia de 72%, correspondendo exatamente à proporção de compostos no novo conjunto. As matrizes de confusão (Figuras 14, 15, 16 e 17) confirmaram que nenhum primo foi identificado.

Esses resultados mostram que mesmo com um desbalanceamento reduzido, os modelos baseados apenas no valor numérico do número não conseguem aprender o conceito de primalidade. A acurácia continua sendo simplesmente a proporção da classe majoritária, e a sensibilidade (recall) para a classe "primo"permanece zero.

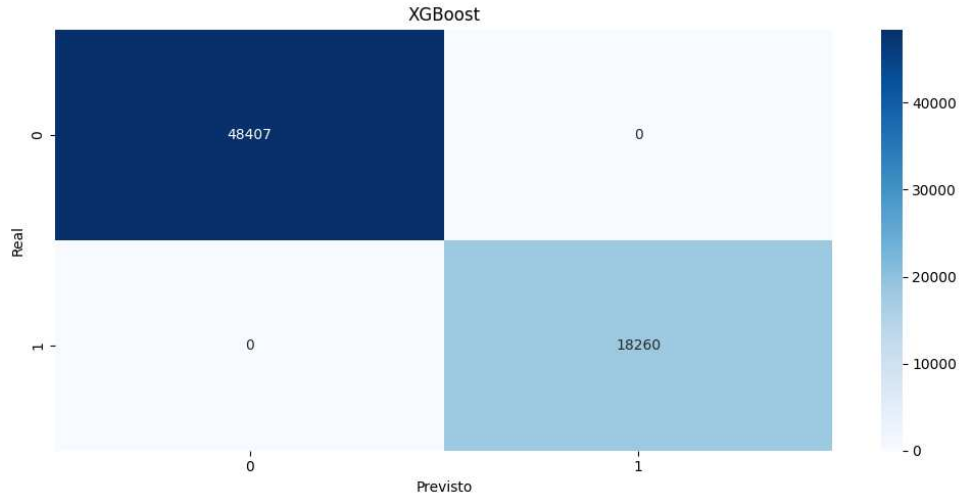
O fato de quase todos os algoritmos terem alcançado 100% de acurácia quando $\Omega(n)$ foi fornecida como *feature* confirma a validade teórica: $\Omega(n) = 1$ é condição necessária e suficiente para que n seja primo. Portanto, para a maioria dos modelos de aprendizado supervisionado que receba $\Omega(n)$ consegue aprender essa regra determinística. Contudo, para os propósitos deste trabalho, o uso de $\Omega(n)$ como variável preditora torna o problema

Figura 10 — Matriz de confusão do KNN, com a Feature Omega e filtrado



Fonte: O autor

Figura 11 — Matriz de confusão do *XGBoost*, com a Feature Omega e filtrado

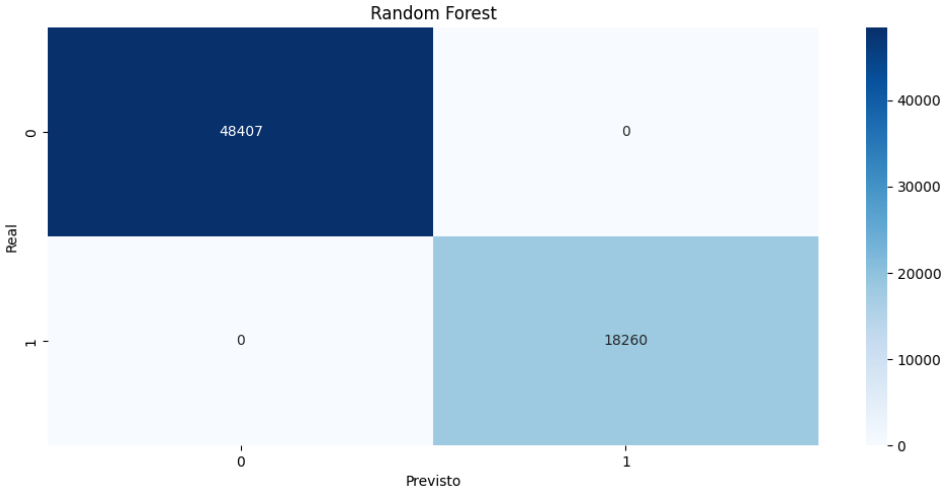


Fonte: O autor

trivial, pois a própria *feature* já contém a resposta.

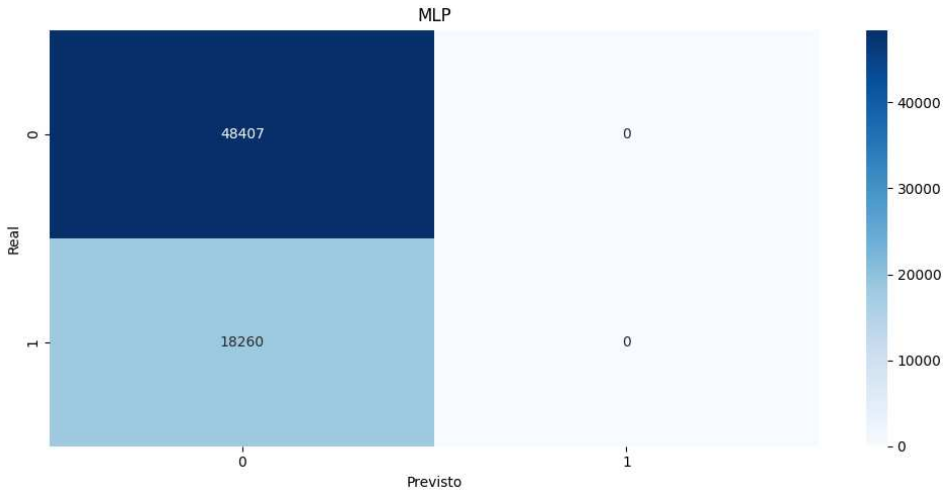
Consequentemente, conclui-se que o treinamento mais válido para futuras abordagens seria o desenvolvimento de um modelo capaz de identificar o valor de $\Omega(n)$ a partir de outras propriedades do número. Tal abordagem permitiria não apenas distinguir primos de compostos, mas também fornecer informações sobre a composição multiplicativa e o número de fatores de qualquer inteiro, o que teria maior relevância para a teoria dos números e criptografia. A criação de um modelo para prever o $\Omega(n)$ do número é um caminho promissor, pois a dificuldade de fatoração de grandes números é o que sustenta protocolos de segurança modernos, como o RSA (NENE e ULUDAG, 2022).

Figura 12 — Matriz de confusão do *Random Forest*, com a Feature Omega e filtrado



Fonte: O autor

Figura 13 — Matriz de confusão do MLP, com a Feature Omega e filtrado

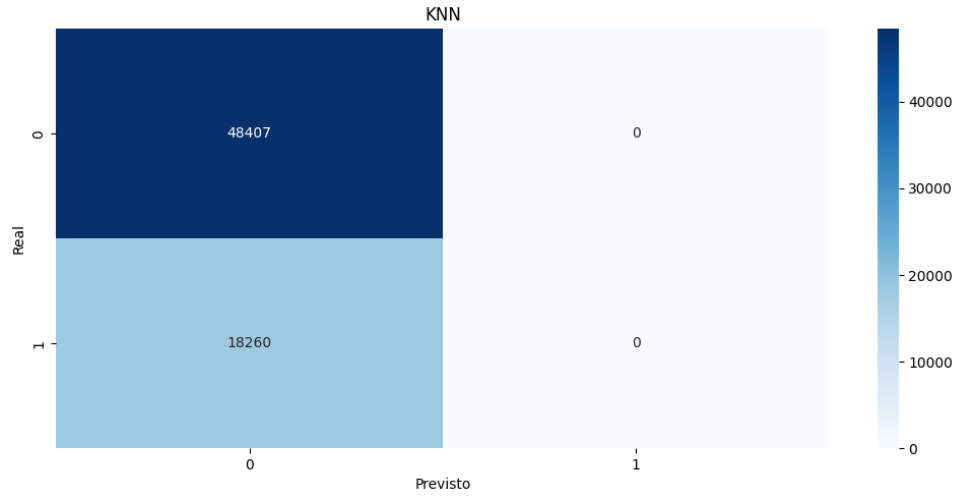


Fonte: O autor

4.3 Síntese dos Resultados

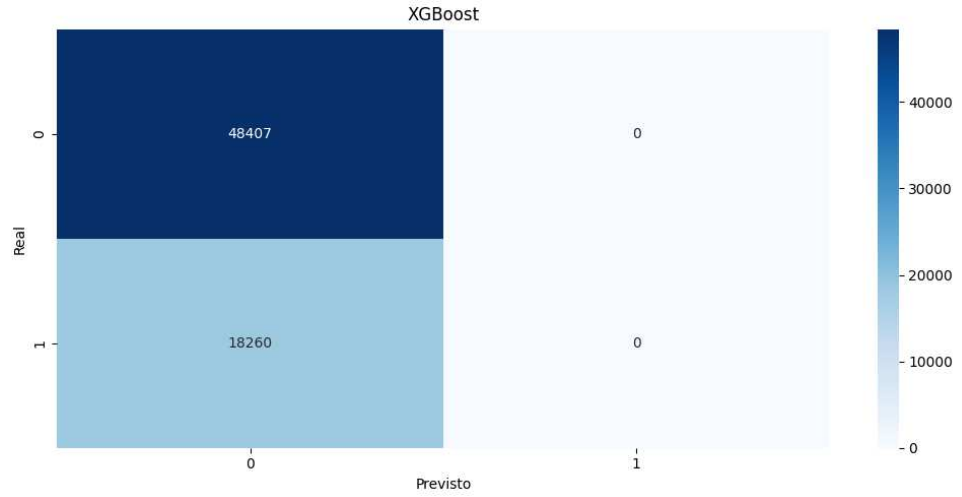
- Na **regressão**, apenas o MLP obteve desempenho satisfatório ($R^2 = 0,96$; MAPE = $1,58\%$). KNN, *Random Forest* e *XGBoost* falharam completamente (R^2 negativo).
- Na **classificação**, a inclusão de $\Omega(n)$ permitiu acurácia perfeita para a maioria dos algoritmos, mas isso equivale a fornecer a solução pronta. Sem $\Omega(n)$, nenhum modelo conseguiu identificar primos, limitando-se a prever a classe majoritária.
- Medidas de mitigação do desbalanceamento (filtragem de múltiplos de 2, 3 e 5) não alteraram esse quadro, indicando que o valor numérico do número, isoladamente, não contém informação suficiente para a distinção entre primos e compostos.

Figura 14 — Matriz de confusão do KNN, sem a Feature Omega e filtrado



Fonte: O autor

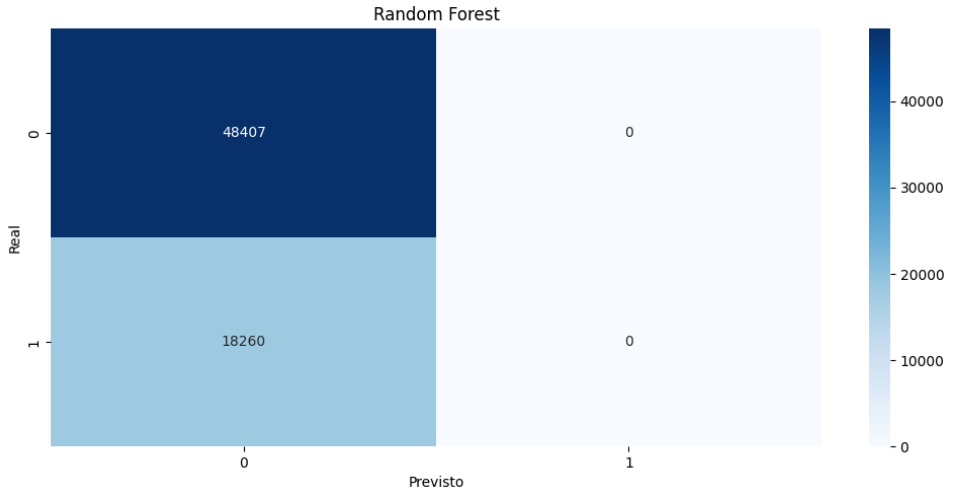
Figura 15 — Matriz de confusão do *XGBoost*, sem a Feature Omega e filtrado



Fonte: O autor

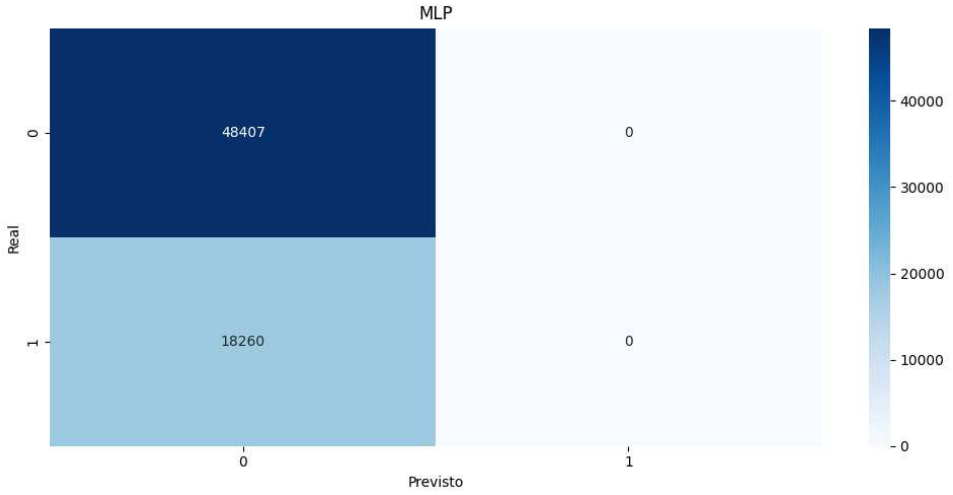
Os resultados indicam que, para a tarefa de regressão, a arquitetura MLP mostrou-se promissora na modelagem do crescimento dos primos. Para a classificação, é necessária uma engenharia de atributos mais sofisticada ou o uso de representações alternativas (como a binária) para capturar padrões não triviais, conforme sugerido nos trabalhos de Blake (2023) discutidos na Seção 2.5.

Figura 16 — Matriz de confusão do *Random Forest*, sem a Feature Omega e filtrado



Fonte: O autor

Figura 17 — Matriz de confusão do MLP, sem a Feature Omega e filtrado



Fonte: O autor

5 CONSIDERAÇÕES FINAIS

Este trabalho investigou a aplicação de técnicas de aprendizado de máquina na modelagem e identificação de números primos, por meio de regressão (previsão do valor do primo) e classificação (distinção entre primos e compostos). Os algoritmos KNN, *Random Forest*, *XGBoost* e MLP foram avaliados sobre uma base de 999.999 primos e conjuntos derivados já detalhados na Metodologia.

Na **regressão**, apenas o MLP apresentou desempenho satisfatório ($R^2 = 0,96$; MAPE $= 1,58\%$), convergindo em 53 épocas. KNN, *Random Forest* e *XGBoost* obtiveram R^2 negativos, indicando incapacidade de capturar o crescimento dos primos.

Na **classificação**, o uso da função $\Omega(n)$ (número total de fatores primos) como variável preditora levou a acurácia perfeita para KNN, *Random Forest* e *XGBoost*, e $92,7\%$ para

o MLP (que classificou todos como compostos). Esse resultado é trivial, pois $\Omega(n) = 1$ define primalidade. Sem $\Omega(n)$, todos os modelos limitaram-se a prever a classe majoritária (composto), com acurácia de $92,7\%$ (e 72% após filtragem de múltiplos de 2, 3 e 5), sem jamais identificar um primo (revocação zero). O desbalanceamento das classes mascara a verdadeira incapacidade dos modelos.

A escolha das variáveis é crucial: $\Omega(n)$ trivializa o problema, enquanto o valor numérico isolado não contém informação suficiente. A acurácia isolada é enganosa em conjuntos desbalanceados; a matriz de confusão revelou o real comportamento.

5.1 Sugestões para trabalhos futuros

- **Predizer** $\Omega(n)$ como preferência no lugar da primalidade direta, obtendo informação mais rica (grau de composição) e relevante para criptografia.
- **Explorar representações alternativas**: binária (conforme Blake, 2023) ou modulares.
- **Arquiteturas mais avançadas**: LSTM (*Long Short-Term Memory*) ou *transformers* para capturar padrões sequenciais.

5.2 Conclusão

Este trabalho demonstrou que, embora algoritmos de aprendizado de máquina possam alcançar excelente desempenho em tarefas de regressão sobre a sequência dos primos (especialmente com o MLP), a classificação de primalidade a partir apenas do valor numérico é inviável com os modelos testados. A aparente alta acurácia foi um artefato do desbalanceamento das classes. A principal contribuição é metodológica: a necessidade de validação criteriosa, uso de matrizes de confusão e escolha adequada de *features*. As sugestões de trabalhos futuros apontam para direções mais ambiciosas, como a predição de $\Omega(n)$ e o uso de representações aritméticas diversas, que podem efetivamente contribuir para a compreensão computacional da distribuição dos números primos.

Por fim, ressalta-se que a integração entre matemática pura e inteligência artificial, ainda que incipiente, tem potencial para gerar novas descobertas, conforme evidenciado por Davies et al. (2021). Este trabalho espera ter contribuído com um passo nessa direção.

Referências

BISHOP, Christopher M. **Pattern Recognition and Machine Learning**. New York: Springer, 2006.

BLAKE, Sam. **Integer factorisation, Fermat and machine learning on a classical computer**. 2023. Preprint. Disponível em: <⟨https://arxiv.org/pdf/2308.12290⟩>. Acesso em: 28 abr. 2026.

BOYER, C. B.; MERZBACH, U. C. **A History of Mathematics**. 3. ed. New York: Wiley, 2011.

BREIMAN, Leo; FRIEDMAN, Jerome H.; OLSHEN, Richard A.; STONE, Charles J. **Classification and regression trees**. Boca Raton: Chapman and Hall/CRC, 1984.

BREIMAN, Leo. **Random forests**. Machine Learning, v. 45, n. 1, p. 5—32, 2001. Disponível em: <⟨https://link.springer.com/article/10.1023/A:1010933404324⟩>. Acesso em: 16 jun. 2026.

BURTON, David M. **Elementary number theory**. 7. ed. New York: McGraw-Hill Education, 2010.

CHEN, Tianqi; GUESTRIN, Carlos. **XGBoost: a scalable tree boosting system**. In: PROCEEDINGS OF THE 22nd ACM SIGKDD INTERNATIONAL CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING, 2016, San Francisco. New York: ACM, 2016. p. 785-–794. Disponível em: <⟨https://dl.acm.org/doi/pdf/10.1145/2939672.2939785⟩>. Acesso em: 17 jun. 2026.

CYBENKO, G. **Approximation by Superpositions of a Sigmoidal Function**. Mathematics of Control, Signals and Systems, v. 2, n. 4, p. 303–314, 1989.

DAVIES, Alex et al. **Advancing mathematics by guiding human intuition with AI**. Nature, v. 600, p. 70–74, 2 dez. 2021. Disponível em: <⟨https://doi.org/10.1038/s41586-021-04086-x⟩>. Acesso em: 28 abr. 2026.

DE LA VALLÉE-POUSSIN, Charles Jean. **Recherches analytiques sur la théorie des nombres premiers**. Annales de la Société Scientifique de Bruxelles, Bruxelles, v. 20, p. 183–256, 1896.

DIFFIE, W.; HELLMAN, M. **New Directions in Cryptography**. IEEE Transactions on Information Theory, v. 22, n. 6, p. 644–654, 1976.

JUNIOR, C. **Métricas para Regressão:** Entendendo as métricas R^2, MAE, MAPE, MSE e RMSE, 2021. Disponivel em: <⟨https://medium.com/data-hackers/prevendo-n%C3%BAmeros-entendendom%C3%A9tricas-de-regress%C3%A3o-35545e011e70⟩>. Acesso em: 28 abr. 2026.

GIL, A. C. **Como elaborar projetos de pesquisa**. São Paulo, SP: Atlas, 1991. Disponível em: <⟨https://wwwp.fc.unesp.br/Home/helber-freitas/tcci/gil_como_elaborar_projetos_de_pesquisa_-anto.pdf⟩>. Acesso em: 28 abr. 2026.

GOMES, Lucas Vinnicius de Oliveira. **Sobre a distribuição dos números primos**. 2020. Trabalho de Conclusão de Curso (Bacharelado em Matemática) – Faculdade de Matemática, Universidade Federal de Uberlândia, Uberlândia, 2020. Disponível em: <⟨https://repositorio.ufu.br/bitstream/123456789/30796/1/SobreDistribuicaoNumeros.pdf⟩>. Acesso em: 28 abr. 2026.

GRANVILLE, Andrew; MARTIN, Greg. **Prime Number Races**. arXiv preprint arXiv:math/0408319, 2004. Disponível em: <⟨https://arxiv.org/pdf/math/0408319⟩>. Acesso em: 28 abr. 2026.

HADAMARD, Jacques. **Sur la distribution des zéros de la fonction $\zeta(s)$ et ses conséquences arithmétiques**. Bulletin de la Société Mathématique de France, Paris, v. 24, p. 199–220, 1896.

HARDY, Godfrey Harold; WRIGHT, Edward Maitland. **An introduction to the theory of numbers**. 6. ed. Oxford: Oxford University Press, 2008.

HONSBERGER, R. **Mathematical gems II**. Washington, D.C.: The Mathematical Association of America, 1976.

LUDERMIR, Teresa Bernarda. **Inteligência artificial e aprendizado de máquina: estado atual e tendências**. Estudos Avançados, v. 35, n. 101, p. 85–94, 2021. Disponível em: <⟨https://doi.org/10.1590/s0103-4014.2021.35101.007⟩>. Acesso em: 28 abr. 2026.

McCULLOCH, W. S.; PITTS, W. **A logical calculus of the ideas immanent in nervous activity**. Bulletin of Mathematical Biophysics, v. 5, p. 115–133, 1943. Disponível em: <⟨https://doi.org/10.1007/BF02478259⟩>. Acesso em: 28 abr. 2026.

MITCHELL, Tom M. **Machine Learning**. New York: McGraw-Hill, 1997. Disponível em: <⟨https://www.cs.cmu.edu/~tom/files/MachineLearningTomMitchell.pdf⟩>. Acesso em: 28 abr. 2026.

MINSKY, M.; PAPERT, S. **Perceptrons: An Introduction to Computational Geometry**. Cambridge: MIT Press, 1969.

MONARD, Maria Carolina; BARANAUSKAS, José Augusto. **Conceitos sobre aprendizado de máquina**. In: ENGEL, Paulo Morel (org.). Sistemas inteligentes: fundamentos e aplicações. Barueri: Manole, 2003. cap. 4, p. 39–56. Disponível em: <⟨https://dcm.ffclrp.usp.br/~augusto/publications/2003-sistemas-inteligentes-cap4.pdf⟩>. Acesso em: 28 abr. 2026.

NENE, Ryan; ULUDAG, Suleyman. **Machine learning approach to integer prime factorisation**. Preprint. setembro 2022. Disponível em: <⟨https://www.researchgate.net/publication/367073950⟩>. Acesso em: 28 abr. 2026.

PADILHA, José Cleiton Rodrigues. **Números primos**. 2013. 72 f. Dissertação (Mestrado Profissional em Matemática em Rede Nacional - PROFMAT) - Universidade Federal da Paraíba, Centro de Ciências Exatas e da Natureza, João Pessoa, 2013. Disponível em: <⟨https://repositorio.ufpb.br/jspui/handle/tede/7534⟩>. Acesso em: 28 abr. 2026.

PÁDUA, M. **Machine Learning -Métricas de avaliação:** Acurácia, Precisão e Recall, F1-score, 2020. Disponivel em: <⟨https://medium.com/@mateuspdua/machine-learning-m%C3%A9tricas-deavalia%C3%A7%C3%A3o-acur%C3%A1cia-precis%C3%A3o-e-recall-d44c72307959⟩>. Acesso em: 28 abr. 2026.

RIVEST, R.; SHAMIR, A.; ADLEMAN, L. **A Method for Obtaining Digital Signatures and Public-Key Cryptosystems**. Communications of the ACM, v. 21, n. 2, p. 120–126, 1978.

RIZEL, Ary Camargo. **Números primos**. 2014. Monografia (Especialização Lato Sensu para Professores com Ênfase em Cálculo) - Departamento de Matemática, Instituto de Ciências Exatas, Universidade Federal de Minas Gerais, Belo Horizonte, 2014. Disponível em: <⟨https://repositorio.ufmg.br/bitstream/1843/EABA-9REL7B/1/monografia_ary.pdf⟩>. Acesso em: 17 jun. 2026.

ROSENBLATT, Murray. **Some purely deterministic processes**. Journal of Mathematics and Mechanics, v. 6, n. 6, p. 801–810, 1957. Disponível em: <⟨http://www.jstor.org/stable/24900623⟩>. Acesso em: 28 abr. 2026.

RUMELHART, D.; HINTON, G.; WILLIAMS, R. **Learning representations by back-propagating errors**. Nature, v. 323, p. 533–536, 9 out. 1986. Disponível em: <⟨https://doi.org/10.1038/323533a0⟩>. Acesso em: 28 abr. 2026.

SINGH, D. P. **A comparative analysis of advanced machine learning techniques for prime number classification based on accuracy and computational efficiency**. IJCRT, v. 13, n. 8, p. 1–10, ago. 2025. ISSN 2320-2882. Disponível em: <⟨https://www.ijcrt.org/papers/IJCRT2508593.pdf⟩>. Acesso em: 28 abr. 2026.

STRAUSS, Edilberto; VILLAS BÔAS JÚNIOR, Manoel; FERREIRA, Wagner Luiz Lobo. **A importância de utilizar métricas adequadas de avaliação de performance em modelos preditivos de machine learning**. Projectus, Rio de Janeiro, v. 7, n. 2, p. 52–62, 2022. Disponível em: <⟨https://pdfs.semanticscholar.org/78fa/b9dcbbefa1361fef4a7eb0b849d523aae2aa.pdf⟩>. Acesso em: 28 abr. 2026.

TEIXEIRA, Marco Antonio Fávaro. **Números inteiros e criptografia RSA**. 2020. 72 f. Dissertação (Mestrado Profissional em Matemática) – Instituto de Geociências e Ciências Exatas, Universidade Estadual Paulista "Júlio de Mesquita Filho", Rio Claro, 2020. Disponível em: <⟨https://repositorio.unesp.br/bitstreams/105473c4-3eb0-4476-84f0-5aa76e829afe/download⟩>. Acesso em: 28 abr. 2026.