UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Abordagens de aprendizado de máquina na seleção de genótipos de maracujazeiro amarelo resistentes ao CABMV

Deurimar Herênio Gonçalves Júnior

Monografia - MBA em Inteligência Artificial e Big Data

ICMC USP SÃO CARLOS

Deurimar Herênio Gonçalves Júnior

Abordagens de aprendizado de máquina na seleção de genótipos de maracujazeiro amarelo resistentes ao CABMV

Monografia apresentada ao Departamento de Ciências de Computação do Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo - ICMC/USP, como parte dos requisitos para obtenção do título de Especialista em Inteligência Artificial e Big Data.

Área de concentração: Inteligência Artificial

Orientador: Prof. Dr. Ricardo Cerri

Versão original

São Carlos

2025

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi, ICMC/USP, com os dados fornecidos pelo(a) autor(a)

Gonçalves Júnior, Deurimar Herênio
S856m Abordagens de aprendizado de máquina na seleção de genótipos de maracujazeiro amarelo resistentes ao CABMV / Deurimar Herênio Gonçalves Júnior ; orientador Ricardo Cerri. – São Carlos, 2025.
103 p. : il. (algumas color.) ; 30 cm.

Monografia (MBA em Inteligência Artificial e Big Data) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2025.

1. LaTeX. 2. abnTeX. 3. Classe USPSC. 4. Editoração de texto. 5. Normalização da documentação. 6. Tese. 7. Dissertação. 8. Documentos (elaboração). 9. Documentos eletrônicos. I. Cerri, Ricardo, orient. II. Título.

Deurimar Herênio Gonçalves Júnior

Abordagens de aprendizado de máquina na seleção de genótipos de maracujazeiro amarelo resistentes ao CABMV

Monograph presented to the Departamento de Ciências de Computação do Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo - ICMC/USP, as part of the requirements for obtaining the title of Specialist in Artificial Intelligence and Big Data.

Concentration area: Artificial Intelligence

Advisor: Prof. Dr. Ricardo Cerri

Original version

São Carlos

2025

RESUMO

Gonçalves Júnior, D. H. **Abordagens de aprendizado de máquina na seleção de genótipos de maracujazeiro amarelo resistentes ao CABMV**. 2025. 103p. Monografia (MBA em Inteligência Artificial e Big Data) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2025.

A investigação conduzida neste estudo focou na aplicação de algoritmos de aprendizado de máquina para a seleção de genótipos de maracujazeiro amarelo com resistência ao cowpea aphid-borne mosaic virus (CABMV), um dos principais entraves à produtividade dessa cultura. O objetivo central consistiu em desenvolver uma metodologia eficaz para classificar genótipos quanto à resistência e produtividade, empregando um conjunto de dados composto por 87 genótipos avaliados em condições de campo. A metodologia adotada integrou etapas de aprendizado não supervisionado, utilizando o algoritmo K-Means para agrupamento dos genótipos, seguidas de aprendizado supervisionado com a aplicação de diversos algoritmos, incluindo Regressão Logística, Árvore de Decisão, Random Forest, Gradient Boosting, AdaBoost, SVM, KNN, Redes Neurais Artificiais (RNA) e Naive Bayes. A avaliação dos modelos foi realizada com base em métricas consolidadas, tais como acurácia, precisão, recall e F1-Score. Os resultados obtidos evidenciaram um desempenho notável de múltiplos algoritmos, com ênfase no Naive Bayes, que atingiu 100% em todas as métricas analisadas. Algoritmos como Regressão Logística, Árvore de Decisão, Random Forest, Gradient Boosting, SVM e RNA apresentaram desempenho consistente, alcançando acurácia de 94%. Por sua vez, AdaBoost e KNN registraram acurácias de 88% e 89%, respectivamente. Entre as variáveis analisadas, a área abaixo da curva de progresso da doença (AACPD), o número de frutos e o rendimento de polpa emergiram como fatores determinantes para a classificação. Conclui-se, portanto, que os algoritmos de aprendizado de máquina constituem uma ferramenta robusta e precisa para otimizar programas de melhoramento genético, viabilizando a seleção eficiente de genótipos resistentes e produtivos.

Palavras-chave: Algoritmo supervisionado; Inteligência artificial; Melhoramento

genético; Naive Bayes.

ABSTRACT

Gonçalves Júnior, D. H. **Machine learning approaches in the selection of yellow passion fruit genotypes resistant to CABMV**. 2025. 103p. Monograph (MBA in Artificial Intelligence and Big Data) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2025.

The investigation conducted in this study focused on the application of machine learning algorithms for the selection of yellow passion fruit genotypes resistant to cowpea aphid-borne mosaic virus (CABMV), one of the main obstacles to the productivity of this crop. The central objective was to develop an effective methodology for classifying genotypes in terms of resistance and productivity, using a dataset composed of 87 genotypes evaluated under field conditions. The adopted methodology integrated unsupervised learning steps, employing the K-Means algorithm for genotype clustering, followed by supervised learning with the application of various algorithms, including Logistic Regression, Decision Tree, Random Forest, Gradient Boosting, AdaBoost, SVM, KNN, Artificial Neural Networks (ANN), and Naive Bayes. Model evaluation was conducted based on established metrics, such as accuracy, precision, recall, and F1-Score. The results demonstrated remarkable performance across multiple algorithms, with Naive Bayes standing out by achieving 100% in all analyzed metrics. Algorithms such as Logistic Regression, Decision Tree, Random Forest, Gradient Boosting, SVM, and ANN showed consistent performance, reaching an accuracy of 94%. Meanwhile, AdaBoost and KNN achieved accuracies of 88% and 89%, respectively. Among the analyzed variables, the area under the disease progress curve (AUDPC), the number of fruits, and pulp yield emerged as determining factors for classification. It is concluded, therefore, that machine learning algorithms constitute a robust and precise tool for optimizing genetic improvement programs, enabling the efficient selection of resistant and productive genotypes.

Keywords: Supervised algorithms; Artificial Intelligence; Genetic breeding; Naive Bayes.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A inteligência artificial (IA) é um campo em rápida evolução que envolve o desenvolvimento de algoritmos e programas de computador que podem aprender e tomar decisões com base em dados. Os sistemas de IA são projetados para imitar funções cognitivas humanas, como percepção, raciocínio, aprendizado e resolução de problemas, a fim de executar tarefas que, de outra forma, exigiriam intervenção humana (JACKSON, 2019).

Existem muitas abordagens diferentes para IA, incluindo sistemas baseados em regras, árvores de decisão e redes neurais. Esses sistemas são usados em uma ampla gama de aplicações, desde assistentes virtuais e *chatbots* até carros autônomos e ferramentas de diagnóstico médico. A IA também é usada em pesquisa científica, análise de negócios e segurança cibernética (LEE; PAN; HSIEH, 2022; HORVITZ; BREESE; HENRION, 1988; DING *et al.*, 2013; DAS *et al.*, 2015).

Apesar do tremendo potencial da IA, também há preocupações sobre seu impacto na sociedade, como deslocamento de empregos, viés na tomada de decisões e violações de privacidade. À medida que a tecnologia de IA continua a evoluir, é importante considerar essas implicações éticas e sociais e desenvolver estratégias para o desenvolvimento e implantação responsáveis de IA (MAKRIDAKIS, 2017).

O aprendizado de máquina é um subcampo da inteligência artificial que envolve o treinamento de máquinas para aprender com dados e melhorar seu desempenho em uma tarefa específica sem ser explicitamente programado. É um campo em rápido crescimento que está mudando a forma como interagimos com a tecnologia e levou a grandes avanços em áreas como visão computacional, processamento de linguagem natural e sistemas autônomos. Os algoritmos de aprendizado de máquina são usados em uma ampla variedade de aplicações, desde recomendações personalizadas em plataformas de mídia social até detecção de fraude em bancos e finanças. Com a crescente disponibilidade de dados e avanços no poder computacional, o aprendizado de máquina está pronto para continuar impulsionando a inovação e transformando as indústrias nos próximos anos (DAS

et al., 2015).

O aprendizado de máquina vem sendo usado na agricultura para melhorar o rendimento das colheitas, reduzir o desperdício e aumentar a eficiência nas operações agrícolas. Ao analisar grandes quantidades de dados de padrões climáticos, condições do solo e crescimento das culturas, os algoritmos de aprendizado de máquina podem fornecer informações e previsões valiosas para ajudar os agricultores a tomar melhores decisões (JHA *et al.*, 2019; SOOD; SHARMA; BHARDWAJ, 2022).

Uma das principais aplicações do aprendizado de máquina na agricultura é a agricultura de precisão. Isso envolve o uso de sensores e outras ferramentas de coleta de dados para monitorar culturas e condições do solo em tempo real e, em seguida, usar algoritmos de aprendizado de máquina para analisar os dados e fazer recomendações para irrigação, aplicação de fertilizantes e manejo de pragas. Isso pode levar a uma economia significativa de custos para os agricultores, bem como ao aumento do rendimento das colheitas e à redução do impacto ambiental (MEGETO *et al.*, 2020).

O aprendizado de máquina também vem sendo amplamente utilizado no campo da genética e o melhoramento de plantas. Ao analisar os dados genéticos das culturas, os algoritmos de aprendizado de máquina podem identificar características associadas a características desejáveis, como resistência a doenças ou alto rendimento. Isso pode ajudar os melhoristas a desenvolver novas cultivares mais adequadas a diferentes condições de cultivo e ambientes. No geral, o uso de aprendizado de máquina na agricultura tem o potencial de transformar o setor, fornecendo soluções mais precisas e baseadas em dados para alguns dos maiores desafios enfrentados pelos agricultores atualmente (DIJK, 2021).

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Análise de dados no melhoramento vegetal

O uso de estatísticas é útil em todos os campos de estudo, incluindo agricultura e ciências biológicas. As estatísticas são um instrumento crucial para todas as formas de pesquisa. A validade do resultado experimental depende do processo preciso de coleta, análise e interpretação dos dados. Portanto, os métodos estatísticos têm uma utilidade significativa para os melhoristas de plantas principalmente para obter um resumo descritivo da amostra, fornecer um meio de inferência e realizar comparações. (OTT; LONGNECKER, 2015; RESENDE MARCOS DEON VILELA, 2007; ACQUAAH, 2012).

2.2 Aprendizado de máquina

O aprendizado de máquina é um campo da inteligência artificial que se concentra no desenvolvimento de algoritmos e modelos que permitem que os computadores aprendam com os dados sem serem explicitamente programados. Tornou-se uma das áreas de crescimento mais rápido da ciência da computação, com aplicações em uma ampla gama de setores, incluindo saúde, finanças, transporte, entretenimento e agricultura (DIJK, 2021). Existem três tipos principais de aprendizado de máquina: aprendizado supervisionado, aprendizado não supervisionado, aprendizado semi-supervisionado, aprendizado por reforço, transdução e *Learning to learn* (DIJK, 2021).

O aprendizado de máquina tem uma ampla gama de aplicações em vários setores. Aqui estão alguns dos mais comuns. Na saúde, os algoritmos de aprendizado de máquina podem ser usados para analisar imagens médicas, como raios-X e exames de ressonância magnética, para identificar padrões e anomalias que podem ser difíceis de serem detectados por um radiologista humano. Eles também podem ser usados para desenvolver modelos preditivos que podem ajudar os médicos a diagnosticar doenças mais cedo e fornecer opções de tratamento mais personalizadas (NAYYAR; GADHAVI; ZAMAN, 2021).

Na esfera das finanças, o aprendizado de máquina pode ser usado para identificar padrões em dados financeiros, como preços de ações e tendências de mercado, para tomar melhores decisões de investimento. Também pode ser usado para detectar fraudes e identificar riscos em solicitações de empréstimos (AHMED *et al.*, 2022). Ou exemplo de aplicação do AM é na área de transporte no geral, carros autônomos dependem fortemente de algoritmos de aprendizado de máquina para interpretar dados de sensores e tomar decisões sobre direção. O aprendizado de máquina também pode ser usado para otimizar rotas e horários de transporte, reduzindo o congestionamento e melhorando a eficiência (MOUJAHID *et al.*, 2018).

Os algoritmos de aprendizado de máquina são também amplamente usados âmbito do entretenimento para desenvolver recomendações personalizadas para filmes, programas de TV e música com base nas preferências e comportamentos anteriores de um usuário. Eles também podem ser usados para criar experiências de jogo mais realistas e imersivas (LYTVYN *et al.*, 2019). No contexto da agricultura, o aprendizado de máquina tem uma diversidade de aplicações,tais como o manejo de culturas, previsão de produtividade, detecção de doenças, detecção de ervas daninhas, qualidade da cultura, reconhecimento de espécies, gestão da água e manejo do solo, dentre outras várias (LIAKOS *et al.*, 2018).

Embora o aprendizado de máquina tenha um enorme potencial, também há desafios significativos a serem superados. Um dos maiores desafios é a necessidade de dados de alta qualidade. Os algoritmos de aprendizado de máquina são tão bons quanto os dados nos quais são treinados, por isso é crucial garantir que os dados sejam precisos, imparciais e representativos. Outro desafio é a complexidade de alguns modelos de aprendizado de máquina. O aprendizado profundo, um subcampo do aprendizado de máquina que envolve o treinamento de redes neurais artificiais, pode ser particularmente desafiador de implementar e interpretar (DAS *et al.*, 2015). Por fim, há considerações éticas e legais a serem consideradas ao implementar o aprendizado de máquina. É essencial garantir que os algoritmos sejam justos, transparentes e não discriminatórios de forma alguma (NASTESKI, 2017).

2.2.1 Aprendizado não supervisionado

No aprendizado não supervisionado o algoritmo é treinado em dados não rotulados, o que significa que a saída desejada não é fornecida para cada entrada. O algoritmo aprende a identificar padrões e relacionamentos dentro dos dados, sem nenhum conhecimento prévio de qual deve ser a saída. Existem dois tipos principais de aprendizado não supervisionado: agrupamento ou do inglês *clustering* e redução de dimensionalidade (DIJK, 2021).

O *clustering* é usado para agrupar pontos de dados semelhantes com base em suas características. O objetivo do agrupamento é encontrar padrões e estruturas nos dados que não são imediatamente aparentes. A redução de dimensionalidade é usada para simplificar dados complexos, reduzindo o número de variáveis. O objetivo da redução de dimensionalidade é reter as informações mais importantes enquanto descarta os dados menos relevantes (DIJK, 2021).

Dentre as numerosas aplicações do aprendizado não supervisionado, pode-se citar: detecção de anomalias, segmentação de mercado, análise de imagem e vídeo, sistemas de Recomendação, entre outras. Todavia, a falta de dados rotulados é um desafio presente a ser superado. Sem uma ideia clara de qual deve ser a saída, pode ser difícil avaliar o desempenho do algoritmo. Outro obstáculo é a complexidade dos algoritmos que exigem mais recursos computacionais e conhecimento especializado (QI; LUO, 2022).

2.2.2 Aprendizado supervisionado

O aprendizado supervisionado é um ramo do aprendizado de máquina no qual um algoritmo é treinado em um conjunto rotulado de dados, o que significa que a saída desejada é fornecida para cada entrada. O algoritmo aprende a fazer previsões ou decisões com base nesses dados de treinamento, e o objetivo é prever com precisão a saída de novas entradas invisíveis (DIJK, 2021). Existem dois tipos principais de aprendizado supervisionado: regressão e classificação (DIJK, 2021).

A regressão é usada quando a variável de saída é contínua, como prever o preço de uma casa com base em seu tamanho e localização. O objetivo da regressão é aprender uma função que mapeia as variáveis de entrada para um valor de saída

contínuo (NASTESKI, 2017). Já os algoritmos de classificação mapeiam o espaço de entrada em classes predefinidas. Existem muitas alternativas para representar classificadores, por exemplo, Máquina de Vetores de Suporte (*Support Vectors Machine* — SVM, do inglês), árvores de decisão, funções algébricas, etc., e podem ser aplicados, por exemplo, na previsão se um e-mail é spam ou não. Ao lado da regressão e estimativa de probabilidade, a classificação é um dos modelos mais estudados, possivelmente de maior relevância prática. Os benefícios potenciais do progresso na classificação são imensos, pois a técnica tem grande impacto em outras áreas, tanto dentro da Mineração de Dados quanto em suas aplicações (NASTESKI, 2017).

O aprendizado supervisionado tem uma ampla gama de aplicações em vários setores. Algoritmos de aprendizado supervisionado podem ser usados para tarefas como análise de sentimento, classificação de texto e tradução automática, área conhecida como processamento de linguagem natural (JURAFSKY; MARTIN, 2023). Na análise de imagem e vídeo pode ser usado para identificar objetos, pessoas e atividades em imagens e vídeos (JIANG; GRADUS; ROSELLINI, 2020).

O aprendizado supervisionado pode ser usado também para detectar atividades fraudulentas, como fraude de cartão de crédito ou fraude de seguro; em sistemas de recomendação personalizadas para produtos, serviços e conteúdo com base nas preferências e comportamentos anteriores do usuário (JIANG; GRADUS; ROSELLINI, 2020). Ainda que o aprendizado supervisionado tenha muitos benefícios, também existem alguns desafios a serem superados. Um dos maiores desafios é a necessidade de dados rotulados de alta qualidade. Criar um conjunto de dados grande, diverso e representativo pode ser demorado e caro (QI; LUO, 2022).

2.2.2.1 Regressão Logística

A Regressão Logística é uma técnica de classificação que tem como objetivo modelar a probabilidade de uma amostra ser associada a uma determinada classe. Para isso, utiliza uma função logística que transforma as variáveis preditoras em probabilidades. O algoritmo calcula a probabilidade $P(y = 1|x)$, ou seja, a probabilidade de uma amostra x pertencer à classe 1 (genótipos resistentes ao CABMV e produtivos), utilizando a seguinte equação logística:

$$P(y = 1|x) = \frac{1}{1 + e^{-(\beta_0+\beta_1 x_1+\beta_2 x_2+\cdots+\beta_n x_n)}}, \tag{2.1}$$

onde: β_0 é o termo de interceptação (bias), $\beta_1, \beta_2, \ldots, \beta_n$ são os coeficientes associados às variáveis preditoras $x_1, x_2, \ldots, x_n$, e é a base do logaritmo natural.

A função logística converte uma combinação linear das variáveis preditoras em um valor que varia entre 0 e 1, interpretado como a probabilidade de a amostra pertencer à classe 1. Para realizar a classificação, utiliza-se um limiar de 0,5: se $P(y = 1|x) \geq 0,5$, a amostra é considerada como resistente e produtiva; caso contrário, é classificada como suscetível e não produtiva.

2.2.2.2 Árvore de Decisão

A Árvore de Decisão é um algoritmo de classificação que segue uma estrutura hierárquica de decisões baseada em regras do tipo "se-então", onde cada nó interno representa um teste sobre um atributo, cada ramo representa o resultado desse teste, e cada nó folha representa um rótulo de classe (BREIMAN *et al.*, 1984). O algoritmo constrói a árvore de forma recursiva, selecionando em cada nó o atributo que melhor divide os dados em subconjuntos homogêneos em relação à classe.

Para determinar a melhor divisão em cada nó, o algoritmo utiliza medidas de impureza, sendo as mais comuns o índice de Gini e a Entropia. O índice de Gini é calculado como:

$$Gini(t) = 1 - \sum_{j=1}^{c} p(j|t)^2 \tag{2.2}$$

onde $p(j|t)$ é a proporção de amostras que pertencem à classe j no nó t, e c é o número total de classes.

A Entropia, por sua vez, é calculada como:

$$Entropia(t) = -\sum_{j=1}^{c} p(j|t) \log_2 p(j|t) \tag{2.3}$$

Para cada possível divisão, calcula-se o ganho de informação, que é a redução na impureza resultante da divisão:

$$Ganho(t, a) = I(t) - \sum_{v \in values(a)} \frac{|t_v|}{|t|} I(t_v) \tag{2.4}$$

onde $I(t)$ é a medida de impureza (Gini ou Entropia) no nó t, $values(a)$ são os possíveis valores do atributo a, $|t_v|$ é o número de amostras no subconjunto t_v, e $|t|$ é o número total de amostras no nó t.

O crescimento da árvore é controlado por parâmetros como profundidade máxima (*max_depth*), número mínimo de amostras para divisão (*min_samples_split*) e número mínimo de amostras em nós folha (*min_samples_leaf*). No presente estudo, esses parâmetros foram otimizados usando o GridSearchCV. A classificação de uma amostra x é feita percorrendo a árvore desde o nó raiz até um nó folha, seguindo as regras de decisão associadas a cada nó. A classe predita é a classe majoritária no nó folha correspondente.

2.2.2.3 *Random Forest*

O algoritmo *Random Forest* (Floresta Aleatória) é um método de aprendizado de máquina que utiliza um conjunto de árvores de decisão para melhorar a precisão e a robustez do modelo (BREIMAN, 2001a). Cada árvore na floresta é treinada em um subconjunto aleatório dos dados de treinamento, com amostragem com reposição (técnica chamada *bootstrap*), e utiliza um subconjunto aleatório de características para fazer as divisões nos nós. Esse processo reduz o risco de *overfitting* e aumenta a capacidade de generalização do modelo (LIAW; WIENER, 2002). A construção de uma Floresta Aleatória pode ser dividida em três etapas principais:

1. **Amostragem com Bootstrap**: Para cada árvore, um subconjunto dos dados de treinamento é amostrado com reposição. Isso implica que algumas amostras podem ser selecionadas mais de uma vez, enquanto outras podem ser ignoradas. Esse subconjunto é utilizado para treinar a árvore, o que ajuda a criar diversidade dentro da floresta.

2. **Seleção Aleatória de Features**: Durante a construção de cada árvore, em cada nó de divisão, apenas um subconjunto aleatório das características é

considerado para encontrar a melhor divisão. O tamanho desse subconjunto é controlado pelo hiperparâmetro *max_features*. Essa seleção aleatória ajuda a reduzir a correlação entre as árvores, aumentando a robustez e a capacidade de generalização do modelo.

3. **Agregação de Resultados**: Após o treinamento das árvores, a previsão final é obtida pela agregação dos resultados individuais. No caso de classificação, a classe predita é determinada pela votação majoritária das árvores. Essa agregação ajuda a diminuir o viés e a variância do modelo, tornando a previsão mais precisa e estável.

A qualidade de um nó $I(t)$ em uma árvore de decisão é avaliada utilizando métricas como o índice de Gini (Equação 2.2) ou a Entropia (Equação 2.3), que ajudam a mensurar a "impureza"dos dados naquele ponto. A divisão ideal para cada nó é determinada pelo ganho de informação (Equação 2.4), que refere-se à diminuição da impureza após a divisão dos dados. Quanto maior o ganho de informação, melhor será a separação dos dados e a qualidade da árvore gerada.

Dentre os principais hiperparâmetros ajustados, um dos mais relevantes é o **n_estimators**, que define quantas árvores compõem a floresta. Embora o aumento no número de árvores possa melhorar a precisão do modelo, ele também eleva o custo computacional, pois mais árvores exigem maior tempo de processamento e consumo de memória. Outro fator importante é o **max_depth**, que limita a profundidade de cada árvore. Essa restrição é fundamental para evitar que o modelo se torne excessivamente complexo, o que poderia resultar em overfitting, além de contribuir para uma maior capacidade de generalização do modelo.

Além disso, os hiperparâmetros **min_samples_split** e **min_samples_leaf** têm um impacto direto na estrutura das árvores. O primeiro determina o número mínimo de amostras necessário para que um nó interno seja dividido, enquanto o segundo especifica a quantidade mínima de amostras exigidas para um nó folha. Ambos os parâmetros são essenciais para controlar o crescimento das árvores, evitando que divisões excessivas levem a modelos excessivamente específicos, o que prejudica a generalização. Por fim, o parâmetro **bootstrap** define se a amostragem com reposição será utilizada durante o treinamento de cada árvore.

Esse método é fundamental para garantir a diversidade das árvores, uma vez que permite que cada uma seja treinada em um subconjunto distinto dos dados.

A predição final do modelo *Random Forest* para uma amostra x é determinada pela combinação das previsões feitas por cada árvore do conjunto. Em problemas de classificação, a classe atribuída à amostra é definida pela votação majoritária das árvores, conforme ilustrado na Equação (2.5):

$$\hat{y}(x) = \text{majority vote}\left(\{h_t(x)\}_{t=1}^{T}\right), \tag{2.5}$$

onde $h_t(x)$ representa a predição da t-ésima árvore, e T é o número total de árvores no ensemble. Para problemas de regressão, a predição final é calculada como a média das predições das árvores, conforme mostrado na Equação (2.6):

$$\hat{y}(x) = \frac{1}{T}\sum_{t=1}^{T} h_t(x). \tag{2.6}$$

A robustez do *Random Forest* decorre da diversidade entre as árvores, que é garantida pela amostragem aleatória dos dados e das *features* durante o treinamento. Essa diversidade reduz a correlação entre as árvores, o que melhora significativamente a capacidade de generalização do modelo (BREIMAN, 2001a).desempenho possível para a tarefa em questão.

2.2.2.4 *Gradient Boosting*

O algoritmo *Gradient Boosting* é uma técnica de aprendizado de máquina baseada em ensemble, que combina múltiplos modelos fracos (geralmente árvores de decisão) de forma sequencial para criar um modelo forte. Diferentemente do *Random Forest*, que treina árvores de forma independente, o *Gradient Boosting* treina cada árvore para corrigir os erros das árvores anteriores, utilizando o gradiente de uma função de perda para guiar o processo de aprendizado (FRIEDMAN, 2001). Essa abordagem iterativa permite que o modelo capture padrões complexos nos dados, melhorando sua precisão e capacidade de generalização.

O *Gradient Boosting* funciona minimizando uma função de perda $L(y, F(x))$, onde y são os valores reais e $F(x)$ são as predições do modelo. Em cada iteração

m, o algoritmo ajusta uma nova árvore $h_m(x)$ para aproximar o gradiente negativo da função de perda em relação às predições atuais $F_{m-1}(x)$. A predição do modelo é então atualizada conforme a Equação (2.7):

$$F_m(x) = F_{m-1}(x) + \nu \cdot h_m(x), \tag{2.7}$$

onde ν é a taxa de aprendizado (*learning rate*), um hiperparâmetro que controla a contribuição de cada árvore ao modelo final. Um valor menor de ν geralmente resulta em um modelo mais preciso, mas requer mais iterações para convergir.

A função de perda mais comum para problemas de regressão é o erro quadrático médio (MSE), dado por:

$$L(y, F(x)) = \frac{1}{2}(y - F(x))^2. \tag{2.8}$$

Para problemas de classificação binária, a função de perda logística é frequentemente utilizada:

$$L(y, F(x)) = -\left[y \log(p) + (1 - y) \log(1 - p)\right], \tag{2.9}$$

onde $p = \sigma(F(x))$ é a probabilidade predita, e σ é a função sigmoide.

O crescimento das árvores no *Gradient Boosting* é controlado por hiperparâmetros como a profundidade máxima (*max_depth*), o número mínimo de amostras para divisão (*min_samples_split*) e o número mínimo de amostras em nós folha (*min_samples_leaf*). Os principais hiperparâmetros testados incluem:

- **n_estimators**: Número de árvores no ensemble. Um número maior de árvores geralmente melhora a precisão, mas aumenta o custo computacional.
- **learning_rate**: Taxa de aprendizado, que controla a contribuição de cada árvore ao modelo final.
- **max_depth**: Profundidade máxima de cada árvore. Controla a complexidade do modelo, evitando *overfitting*.

- **min_samples_split**: Número mínimo de amostras necessárias para dividir um nó interno.
- **min_samples_leaf**: Número mínimo de amostras necessárias em um nó folha.

A predição final do *Gradient Boosting* para uma amostra x é dada pela soma ponderada das predições de todas as árvores, conforme a Equação (2.10):

$$F_M(x) = F_0(x) + \nu \sum_{m=1}^{M} h_m(x), \tag{2.10}$$

onde $F_0(x)$ é a predição inicial (geralmente a média dos valores de y para regressão ou o log-odds para classificação), M é o número total de árvores, e $h_m(x)$ é a predição da m-ésima árvore.

A robustez do *Gradient Boosting* vem de sua capacidade de ajustar iterativamente os erros residuais do modelo, o que permite capturar relações complexas nos dados.

2.2.2.5 AdaBoost (*Adaptive Boosting*)

O método *AdaBoost* (do inglês *Adaptive Boosting*) é uma abordagem de aprendizado de máquina que utiliza a técnica de *ensemble* para aprimorar a precisão das previsões. Ele funciona combinando diversos modelos simples, frequentemente chamados de "classificadores fracos", para formar um modelo final mais robusto. Ao contrário de outras estratégias de *ensemble*, como *Random Forest* ou *Gradient Boosting*, o *AdaBoost* ajusta os pesos das observações durante o processo de treinamento, priorizando aquelas que foram mal classificadas nas etapas anteriores. Dessa forma, o algoritmo direciona sua atenção para os casos mais desafiadores, refinando sua capacidade de generalização a cada iteração (FREUND; SCHAPIRE, 1997).

No início do processo, o *AdaBoost* atribui pesos idênticos a todas as observações do conjunto de treinamento. A cada iteração t, um classificador fraco $h_t(x)$ é ajustado com o objetivo de reduzir ao máximo o erro ponderado ϵ_t, que é calculado da seguinte forma:

$$\epsilon_t = \sum_{i=1}^{N} w_i^{(t)} \cdot I(y_i \neq h_t(x_i)), \tag{2.11}$$

onde $w_i^{(t)}$ é o peso da i-ésima amostra na iteração t, y_i é o rótulo verdadeiro da amostra, $h_t(x_i)$ é a predição do aprendiz fraco, e $I(\cdot)$ é a função indicadora que retorna 1 se a condição for verdadeira e 0 caso contrário.

Após o ajuste do classificador fraco, o peso α_t correspondente a ele é determinado com base no erro ponderado ϵ_t, conforme a Equação (2.12):

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right). \tag{2.12}$$

O valor (ou peso) de α_t define o quanto o classificador fraco $h_t(x)$ influencia o modelo final. Classificadores que cometem menos erros recebem um peso maior, enquanto aqueles com desempenho inferior têm sua contribuição reduzida.

Posteriormente, os pesos das observações são recalculados para a iteração seguinte $t + 1$, priorizando as amostras que foram classificadas de forma incorreta. A fórmula para a atualização dos pesos é dada pela Equação (2.13):

$$w_i^{(t+1)} = w_i^{(t)} \cdot \exp\left(-\alpha_t \cdot y_i \cdot h_t(x_i)\right), \tag{2.13}$$

onde $y_i \cdot h_t(x_i)$ é positivo se a classificação estiver correta e negativo caso contrário. Após a atualização, os pesos são normalizados para garantir que somem 1.

A previsão final do *AdaBoost* para uma observação x é calculada por meio da soma ponderada das previsões individuais de todos os classificadores fracos, conforme expresso na Equação (2.14):

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t \cdot h_t(x) \right), \tag{2.14}$$

onde T é o número total de aprendizes fracos, α_t é o peso do t-ésimo aprendiz, e $h_t(x)$ é a predição do t-ésimo aprendiz. A função sign$(\cdot)$ retorna a classe predita com base no sinal da soma ponderada.

O desempenho do *AdaBoost* depende de vários hiperparâmetros, como a quantidade de classificadores fracos (*n_estimators*), a taxa de aprendizado (*learning_rate*) e o método escolhido para atualizar os pesos (*algorithm*). Os principais hiperparâmetros incluem o número de classificadores fracos no ensemble (*n_estimators*), para o qual foram testados os valores 50, 100, 200 e 300. Um número maior de classificadores tende a melhorar a acurácia, mas também eleva o custo computacional.

A taxa de aprendizado (*learning_rate*), que define o impacto de cada classificador no modelo final, foi testada com os valores 0.01, 0.1, 0.5 e 1. Taxas menores demandam mais classificadores para alcançar a convergência, mas podem levar a modelos mais refinados. Por fim, o método utilizado para ajustar os pesos (*algorithm*) foi o *SAMME* (*Stagewise Additive Modeling using a Multi-class Exponential loss function*), que é apropriado para tarefas de classificação envolvendo múltiplas classes.

A robustez do *AdaBoost* está diretamente relacionada à sua habilidade de concentrar-se nas observações mais desafiadoras, ajustando de forma iterativa os pesos das amostras e integrando as previsões de diversos classificadores fracos.

2.2.2.6 *Support Vector Machine* (SVM)

O algoritmo *Support Vector Machine* (SVM) tem como objetivo identificar um hiperplano ideal que divide as classes no espaço de características, maximizando a distância entre os pontos mais próximos de cada classe, denominados vetores de suporte (CORTES; VAPNIK, 1995). Essa característica torna o SVM especialmente eficiente em problemas de alta dimensionalidade e em situações onde a separação entre as classes não é linear. No contexto de classificação binária, o SVM procura determinar um hiperplano descrito por:

$$w \cdot x + b = 0, \tag{2.15}$$

onde w é o vetor de pesos, x é o vetor de características da amostra, e b é o termo de viés. O objetivo é maximizar a margem M, que é a distância entre o

hiperplano e os vetores de suporte mais próximos de cada classe. A margem é dada por:

$$M = \frac{2}{\|w\|}, \tag{2.16}$$

onde $\|w\|$ é a norma do vetor de pesos. Para maximizar a margem, o SVM resolve o seguinte problema de otimização quadrática:

$$\min_{w,b} \frac{1}{2}\|w\|^2 \quad \text{sujeito a} \quad y_i(w \cdot x_i + b) \geq 1 \quad \forall i, \tag{2.17}$$

onde y_i é o rótulo da classe ($+1$ ou -1) da i-ésima amostra, e x_i é o vetor de características correspondente.

Em casos onde os dados não são linearmente separáveis, o SVM utiliza uma técnica chamada *kernel trick*, que mapeia os dados para um espaço de maior dimensionalidade onde a separação linear é possível (CORTES; VAPNIK, 1995). A função de kernel $K(x_i, x_j)$ calcula o produto interno entre os vetores de características no espaço transformado. As funções de kernel mais comuns incluem o kernel linear, definido por:

$$K(x_i, x_j) = x_i \cdot x_j, \tag{2.18}$$

que é adequado para dados linearmente separáveis e não realiza nenhuma transformação não linear (SCHÖLKOPF; SMOLA, 2002). O kernel polinomial, dado por:

$$K(x_i, x_j) = (\gamma \cdot x_i \cdot x_j + r)^d, \tag{2.19}$$

que mapeia os dados para um espaço de características polinomiais, onde γ controla a influência de cada amostra, r é um termo de interceptação, e d é o grau do polinômio. Esse kernel é útil para capturar relações não lineares de grau d (VAPNIK, 1998). O kernel radial basis function (RBF), definido por:

$$K(x_i, x_j) = \exp\left(-\gamma \|x_i - x_j\|^2\right), \tag{2.20}$$

é uma das funções de kernel mais utilizadas e é adequado para dados com estruturas complexas e não lineares, onde γ controla a influência de cada amostra no espaço transformado (SCHÖLKOPF; SMOLA, 2002). Por fim, o kernel sigmoide, dado por:

$$K(x_i, x_j) = \tanh(\gamma \cdot x_i \cdot x_j + r), \tag{2.21}$$

é semelhante à função de ativação sigmoide usada em redes neurais, onde γ controla a influência de cada amostra, e r é um termo de interceptação. Esse kernel pode ser útil para capturar relações não lineares em certos casos (VAPNIK, 1998).

O desempenho do SVM é diretamente impactado por hiperparâmetros como o parâmetro de regularização C, que define o equilíbrio entre a maximização da margem e a redução do erro de classificação, e o parâmetro γ, utilizado em kernels como RBF, polinomial e sigmoide (CORTES; VAPNIK, 1995).

2.2.2.7 *K-Nearest Neighbors* (KNN)

O algoritmo *K-Nearest Neighbors* (KNN) é uma técnica simples e eficiente que armazena todos os exemplos conhecidos e classifica novos dados com base em sua similaridade com os registros existentes (COVER; HART, 1967). A seleção adequada dos hiperparâmetros desse algoritmo é fundamental para garantir seu bom desempenho. O primeiro hiperparâmetro, **n_neighbors**, define o número de vizinhos (k) que serão utilizados para classificar um novo ponto. Valores muito pequenos de k podem tornar o modelo suscetível a ruídos, enquanto valores muito altos podem resultar em fronteiras de decisão excessivamente suavizadas (ZHANG, 2016a).

O segundo hiperparâmetro, **weights**, controla a contribuição dos vizinhos na classificação: com a opção *uniform*, todos os vizinhos têm o mesmo peso, enquanto com *distance*, os vizinhos mais próximos exercem maior influência na decisão (ALTMAN, 1992). Por último, o hiperparâmetro **metric** define a métrica

de distância usada para calcular a proximidade entre os pontos. Entre as métricas testadas estão a distância euclidiana, que mede a distância "em linha reta"entre dois pontos, a distância de Manhattan, que considera a soma das diferenças absolutas entre as coordenadas, e a distância de Minkowski, uma generalização das duas anteriores (DEZA; DEZA, 2012).

O KNN classifica os dados de entrada com base nos rótulos dos K vizinhos mais próximos, a partir do conjunto de treinamento $\{x_i, y_i\}$. Para isso, é calculada a distância entre a amostra de treinamento e a amostra de teste, ambas representadas como vetores binários de dimensão $(L + R)$, sendo x_u o ponto de entrada e x_i cada ponto do conjunto de treinamento. A métrica de distância utilizada (euclidiana, Manhattan ou Minkowski) define como essa proximidade é medida. A seguir, são apresentadas as fórmulas para cada uma dessas métricas:

- **Distância Euclidiana**:

$$d(x_u, x_i) = \sqrt{\sum_{j=1}^{n} (x_{u,j} - x_{i,j})^2} \tag{2.22}$$

  onde n é o número de dimensões dos vetores x_u e x_i (DEZA; DEZA, 2012).

- **Distância de Manhattan**:

$$d(x_u, x_i) = \sum_{j=1}^{n} |x_{u,j} - x_{i,j}| \tag{2.23}$$

  que corresponde à soma das diferenças absolutas entre as coordenadas dos pontos (DEZA; DEZA, 2012).

- **Distância de Minkowski**:

$$d(x_u, x_i) = \left(\sum_{j=1}^{n} |x_{u,j} - x_{i,j}|^p \right)^{1/p} \tag{2.24}$$

  onde p é um parâmetro que define a ordem da distância. Para $p = 1$, a distância de Minkowski equivale à distância de Manhattan, e para $p = 2$, equivale à distância euclidiana (DEZA; DEZA, 2012).

O rótulo de categoria de x_u foi atribuído com base na maioria dos votos dos rótulos de categoria de seus k-vizinhos mais próximos (KNN) (SUN; ZHAO, 2015):

$$y_u = \arg\max_{y_j} \sum_{i=1}^{k} \delta(x_i, y_j), \tag{2.25}$$

onde $\delta(x_i, y_j) \in \{0, 1\}$ indica se x_i pertence a y_j.

2.2.2.8 Redes Neurais Artificiais (MLP)

As Redes Neurais Artificiais (RNA), especialmente as *Multilayer Perceptrons* (MLP), são modelos de aprendizado de máquina que empregam múltiplas camadas de neurônios artificiais para capturar relações complexas e não lineares nos dados (HAYKIN, 1999). As funções de ativação testadas foram *logistic*, *tanh* e *ReLU*, definidas respectivamente por:

$$f(z) = \frac{1}{1 + e^{-z}} \quad \text{(logística)}, \tag{2.26}$$

$$f(z) = \tanh(z) \quad \text{(tangente hiperbólica)}, \tag{2.27}$$

$$f(z) = \max(0, z) \quad \text{(ReLU)}. \tag{2.28}$$

2.2.2.9 *Naive Bayes* (GaussianNB)

O algoritmo *Naive Bayes* (GaussianNB) é um método de classificação probabilístico baseado no teorema de Bayes, que assume independência condicional entre as características dadas as classes (MITCHELL, 1997). Apesar dessa suposição simplificadora, o GaussianNB é eficaz em muitos problemas de classificação, especialmente quando o número de características é grande ou quando os dados são esparsos.

2.2.2.9.1 Teorema de Bayes e Suposição de Independência

O teorema de Bayes é a base do algoritmo Naive Bayes e é dado por:

$$P(y|X) = \frac{P(X|y) \cdot P(y)}{P(X)}, \tag{2.29}$$

onde: - $P(y|X)$ é a probabilidade posterior da classe y dado o vetor de características X, - $P(X|y)$ é a verossimilhança das características X dado a classe y, - $P(y)$ é a probabilidade a priori da classe y, - $P(X)$ é a probabilidade marginal das características X.

A suposição de independência condicional do Naive Bayes assume que as características $X_1, X_2, \ldots, X_n$ são independentes entre si dado a classe y. Isso permite que a verossimilhança $P(X|y)$ seja fatorada como:

$$P(X|y) = \prod_{i=1}^{n} P(X_i|y). \tag{2.30}$$

Essa suposição simplifica o cálculo da probabilidade posterior, tornando o algoritmo computacionalmente eficiente.

No caso do GaussianNB, assume-se que as características contínuas seguem uma distribuição normal (Gaussiana) para cada classe. A probabilidade $P(X_i|y)$ é calculada utilizando a função de densidade de probabilidade da distribuição normal:

$$P(X_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(X_i - \mu_y)^2}{2\sigma_y^2}\right), \tag{2.31}$$

onde: - μ_y é a média da característica X_i para a classe y, - σ_y^2 é a variância da característica X_i para a classe y.

Para evitar problemas numéricos quando uma característica tem variância zero em uma classe, o GaussianNB utiliza um parâmetro de suavização chamado *var_smoothing*. Esse parâmetro adiciona uma pequena constante à variância, garantindo que ela nunca seja zero. A variância suavizada é calculada como:

$$\sigma_y^2 \leftarrow \sigma_y^2 + \text{var\_smoothing}. \tag{2.32}$$

2.3 Aprendizado de máquina no melhoramento de plantas

A população mundial está projetada para exceder nove bilhões de pessoas até 2050, o que exigirá melhorias significativas na produção das principais culturas que contribuem para a segurança alimentar global, portanto aumentar a produtividade é o principal objetivo da maioria dos programas de melhoramento de plantas (DUBEY *et al.*, 2019). No melhoramento vegetal, no entanto, aferir características como o rendimento, que é influenciado por uma combinação de características quantitativas e qualitativas, em grandes populações, é oneroso, demorado e trabalhoso (XIONG *et al.*, 2018; CAI *et al.*, 2016).

Os métodos clássicos de melhoramento de plantas incluem principalmente avaliação e classificação da diversidade genética, análise de componentes de rendimento (seleção indireta de genótipos superiores), análise de estabilidade da produtividade (interação genótipo × ambiente), tolerância a estresses bióticos e abióticos e programas de melhoramento de híbrido (NIAZIAN; NIEDBAŁA, 2020). Aliado ao melhoramento clássico, as ferramentas de biotecnologia visam o desenvolvimento da área, tornando o processo mais rápido, preciso e eficiente (ADLAK *et al.*, 2019). Em geral, as abordagens relacionadas à biotecnologia amplamente adotadas no processo de melhoramento de plantas podem ser divididas em cultura de tecidos, engenharia genética, marcadores moleculares e análises de DNA (FALEIRO; ANDRADE SOLANGE ROCHA MONTEIRO; Fábio Bueno dos Reis Junior, 2011).

No processo de melhoramento são gerados os chamados "*big datas*", provenientes de dados fenotípicos dos mais diverso, sequência de moléculas, dados de pedigree, análise de imagens etc (NIAZIAN; NIEDBAŁA, 2020). Técnicas estatísticas clássicas têm sido aplicadas para analisar e interpretar os resultados oriundos desses dados, todavia, tais técnicas, incluindo as baseadas em regressão, geralmente são limitadas em sua capacidade de analisar dados de alta dimensão e não conseguem capturar relações complexas e multivariadas entre as variáveis, que frequentemente apresentam propriedades não lineares e não determinísticas e estão inextricavelmente ligadas aos sistemas biológicos das plantas e fontes externas (DIJK, 2021; NIAZIAN; NIEDBAŁA, 2020; WEI *et al.*, 2020; HESAMI *et al.*, 2019).

Os algoritmos de aprendizado de máquina vem chamando a atenção de pesquisadores na otimização de métodos de melhoramento baseados em modelos que podem melhorar a eficiência do processo (HESAMI *et al.*, 2020). Uma das redes neurais artificiais (RNAs) mais comuns, o *multilayer perceptron* (MLP) (PAL; MITRA, 1992) tem sido amplamente utilizado para modelar e prever características complexas, como rendimento, em diferentes programas de melhoramento (INOCENTE; GARBUGLIO; RUAS, 2022; SANDHU *et al.*, 2021b). As chamadas máquinas de vetores de suporte (SVMs) são conhecidas como um dos algoritmos de aprendizado de máquina poderosos e fáceis de usar que podem reconhecer padrões e comportamento de relacionamentos não lineares (AURIA; MORO, 2008; SU *et al.*, 2017). Além de MLP e SVM, a *Random Forest* (RF) (BREIMAN, 2001a) é outro método de modelagem de dados com uma fase de treinamento computacional eficiente e precisão de generalização muito alta que tem sido amplamente utilizada no melhoramento de plantas (ANSARIFAR; AKHAVIZADEGAN; WANG, 2020; ACHARJEE *et al.*, 2016; SARKAR *et al.*, 2015). Abaixo são descritos com detalhes alguns dos algoritmos de aprendizado de máquina utilizados no melhoramento vegetal.

3 MATERIAIS E MÉTODOS

3.1 Conjuntos de Dados

O conjunto de dados é proveniente de um experimento de campo que incluiu 87 genótipos de três famílias de irmãos completos resultantes do cruzamento (recombinação) entre indivíduos da primeira geração de retrocruzamento do programa de melhoramento de maracujazeiro da Universidade Estadual do Norte Fluminense Darcy Ribeiro (UENF). O experimento foi realizado entre em março de 2018 e a junho de 2019. Os atributos avaliados foram: resistência ao CABMV, formato do fruto, cor da casca, cor da polpa, número total de frutos por plantas, peso total de frutos, diâmetro longitudinal médio do fruto, diâmetro transversal médio do fruto, índice de formato médio do fruto, massa média do fruto, massa média da polpa, rendimento médio de polpa, espessura média da casca e Teor de sólidos solúveis totais médio (BRIX).

3.2 Normalização dos dados

A normalização dos dados é o processo de ajustar os valores de cada atributo para uma faixa específica, como de -1 a 1 ou de 0 a 1. Esse procedimento é essencial para evitar que atributos com escalas de valores maiores influenciem de maneira desproporcional o desempenho de algoritmos de aprendizado de máquina (GOLDSCHMIDT, 2015).

Neste trabalho, utilizou-se a normalização por desvio padrão, também conhecida como Z-Score, uma técnica amplamente utilizada para padronizar dados. Essa abordagem ajusta os valores de cada atributo no conjunto de dados de forma que a média seja 0 e o desvio padrão seja igual a 1 (GOLDSCHMIDT, 2015).

A Equação seguinte descreve o processo de normalização:

$$Z = \frac{x - \mu}{\sigma} \tag{3.1}$$

onde x é o valor individual da variável, μ é a média da variável no conjunto de

dados e σ é o desvio padrão da variável no conjunto de dados.

3.3 Análise de *cluster*

3.3.1 Agrupamento com *K-means*

O algoritmo K-Means foi utilizado para agrupar os genótipos. O algoritmo K-means é uma técnica de aprendizado não supervisionado, baseado em partições, ele busca dividir os dados em k clusters, com o objetivo de minimizar a soma das distâncias quadráticas entre os pontos e os centros de cada cluster. O *K-means* é um dos algoritmos de particionamento de clustering mais amplamente usados (IKOTUN *et al.*, 2023).

O algoritmo K-means agrupa dados com características semelhantes em um mesmo *cluster*, enquanto dados com características distintas são alocados em *clusters* diferentes, garantindo que os elementos de um *cluster* apresentem pouca variação entre si. A proximidade entre dois objetos é determinada pela distância entre eles. Da mesma forma, a proximidade de um dado a um *cluster* específico é avaliada com base na distância entre esse dado e o centro do *cluster*. A menor distância entre um dado e o centro de um *cluster* define a qual *cluster* ele pertence (GOLDSCHMIDT, 2015). O cálculo da distância entre os dados e os centroides de cada *cluster* é realizado utilizando a fórmula da distância euclidiana, descrita a seguir:

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(x^i - y^i)^2} \tag{3.2}$$

onde x^i é o primeiro ponto e y^i é o segundo ponto. As etapas para executar as áreas do Algoritmo de Agrupamento K-Means são as seguintes:

1. Determinar o valor de K como o número de *clusters*.
2. Selecionar K do conjunto de dados X como o centroide.
3. Alocar todos os dados para o centroide com métrica de distância usando a Equação 3.2.

4. Recalcular o centroide C com base nos dados que seguem cada *cluster*. Repetir.

3.3.2 Determinação do número de ótimo *clusters*

Um dos desafios na utilização do do *K-means* é que ele requer que o número de *clusters* seja pré-especificado antes que o algoritmo seja aplicado (KUMARSAGAR; SHARMA, 2014) e que esse número escolhido seja validado de alguma forma. A validação de clusters por meio de medidas internas é uma abordagem utilizada para avaliar se os clusters foram formados corretamente. Dois fatores principais são considerados nessa análise: a coesão, que mede o quão compactos estão os dados dentro de um cluster, e a separação, que avalia o quão distintos os clusters estão uns dos outros (SAPUTRA; SAPUTRA; OSWARI, 2020). Para determinar o número ideal de clusters k, foram aplicados dois métodos, o Método do cotovelo ou método de (*Elbow*) e o Silhouette Score, e para todos eles foram avaliados de 1 a 10 *clusters*.

3.3.2.1 Método do cotovelo (*Elbow*)

O Método do Cotovelo (ou *Elbow Method*, em inglês) é uma das técnicas mais utilizadas para definir o número ideal de clusters k ao aplicar o algoritmo *K-Means*, ele é usado para medir a coesão de *clusters*, avaliando quão semelhantes os dados dentro de um mesmo *cluster* são. Esse método busca identificar o valor de k que oferece a melhor solução para o agrupamento, equilibrando a qualidade dos *clusters* com a complexidade do modelo.

O *K-Means* é um algoritmo de agrupamento baseado em partições que tem como objetivo dividir um conjunto de dados em k clusters, de modo que os dados dentro de cada cluster sejam o mais homogêneos possível. O algoritmo busca minimizar a soma das distâncias quadráticas entre os pontos de dados e os centros de seus respectivos clusters. O critério utilizado para avaliar o agrupamento é a inércia, que é calculada pela soma das distâncias quadráticas entre todos os pontos de dados e seus centros de cluster.

Matematicamente, a inércia W_k para um dado número de clusters k é dada

por:

$$W_k = \sum_{i=1}^{k} \sum_{x_j \in C_i} \|x_j - \mu_i\|^2 \tag{3.3}$$

Onde: k é o número de clusters, C_i é o conjunto de pontos atribuídos ao cluster i, x_j é o ponto de dados, μ_i é o centroide do cluster i, $\|x_j - \mu_i\|^2$ é a distância quadrática entre o ponto x_j e o centroide μ_i. No presente trabalho foram avaliados 10 possíveis clusters.

3.3.2.2 Método da silhueta (*Silhouette Score*)

O Método Silhouette usa um coeficiente de silhueta que combina separação e coesão. O coeficiente de silhueta é determinado pela divisão da medida de separação pela medida de coesão e subtraindo esse valor por 1 se a medida de separação for maior que a medida de coesão ou por 1 subtraído pelo valor da medida de coesão dividido pela medida de separação se a coesão for maior que a separação. Quanto maior o coeficiente de silhueta, melhor o cluster é (SAPUTRA; SAPUTRA; OSWARI, 2020).

$$s = \begin{cases} 1 - \frac{\text{medida de coesão}}{\text{medida de separação}}, & \text{se coesão} < \text{separação} \\ \frac{\text{medida de separação}}{\text{medida de coesão}} - 1, & \text{se coesão} > \text{separação} \end{cases} \tag{3.4}$$

O Método Silhouette propõe a construção de um gráfico onde o eixo Y representa o coeficiente Silhouette e o eixo X representa o valor de K. O valor de K ideal é aquele que corresponde ao maior valor do coeficiente Silhouette, que indica a melhor separação e coesão entre os clusters formados (SAPUTRA; SAPUTRA; OSWARI, 2020).

3.3.2.3 Índice Calinski-Harabasz

Foi também empregado para a validação quantitativa da capacidade do algoritmo de *K-means* em capturar os grupos inerentes aos dados o índice de Calinski-Harabasz, também conhecido como Critério da Razão da Variância. O índice de Calinski-Harabasz se baseia na razão entre a dispersão entre os *clusters* (o quão separados os *clusters* estão uns dos outros) e a dispersão dentro dos *clusters*

(o quão compactos os pontos estão dentro de cada *cluster*) (U, 2002). As etapas para calcular o índice são descritas a seguir:

1. Passo 1: Calcular a soma dos quadrados entre grupos (BGSS), a qual é uma medida que calcula a soma ponderada do quadrado das distâncias entre os centroides de cada *cluster* e o centroide geral do conjunto de dados. A fórmula para o cálculo é:

$$BGSS = \sum_{i=1}^{k} n_i \cdot \|c_i - c\|^2 \tag{3.5}$$

   onde k é o número de clusters, n_i é a quantidade de pontos no cluster i, c_i é o centroide do cluster i, c é o centroide geral do conjunto de dados, $\|c_i - c\|^2$ representa o quadrado da distância entre o centroide do cluster i e o centroide geral.

2. Passo 2: Calcular a soma dos quadrados dentro do grupo (WGSS), que é usada para medir a soma do quadrado das distâncias entre cada observação e o centroide do *cluster* ao qual pertence. Para cada *cluster* k, o $WGSS_k$ é calculado da seguinte forma:

$$WGSS_k = \sum_{x \in C_k} \|x - c_k\|^2 \tag{3.6}$$

   em que C_k representa o conjunto de observações no *cluster* k, x é uma observação pertencente ao *cluster* k, c_k é o centroide do *cluster* k, $\|x - c_k\|^2$ é o quadrado da distância entre a observação x e o centroide c_k.

3. Passo 3: Calcular o índice Calinski-Harabasz de acordo com a seguinte equação:

$$CH = \frac{\text{BGSS}/(k-1)}{\text{WGSS}/(n-k)} \tag{3.7}$$

   onde CH é o índice Calinski-Harabasz, $BGSS$ é a soma dos quadrados entre grupos, $WGSS$ é a soma dos quadrados dentro dos grupos, k é o número de clusters, n é o número total de observações. Esse índice tende a valores mais altos quando os *clusters* são bem separados e compactos, o que indica uma melhor qualidade do agrupamento.

3.3.3 Avaliação da Estabilidade dos *Clusters*

3.3.3.1 Índice de Rand Ajustado (ARI)

O Índice de Rand Ajustado (ARI) é uma métrica que mede a similaridade entre duas atribuições de clusters, comparando os pares de pontos que estão no mesmo cluster ou em clusters diferentes. Ele é uma versão ajustada do Índice de Rand, que corrige o efeito do acaso, garantindo que valores próximos de zero indiquem uma concordância aleatória entre os agrupamentos (HUBERT; ARABIE, 1985).

O ARI é calculado da seguinte forma:

$$ARI = \frac{\text{Índice de Rand} - \text{Índice de Rand Esperado}}{\text{Índice de Rand Máximo} - \text{Índice de Rand Esperado}} \tag{3.8}$$

onde Índice de Rand é a proporção de pares de pontos que estão no mesmo *cluster* em ambas as atribuições, Índice de Rand Esperado é o valor esperado do Índice de Rand sob uma distribuição aleatória e Índice de Rand Máximo é o valor máximo que o Índice de Rand pode assumir.

O ARI varia entre -1 e 1, onde 1 indica que as duas atribuições de *clusters* são idênticas, 0 indica que a concordância entre as atribuições é aleatória e -1 indica que as atribuições são completamente diferentes.

3.4 Avaliação de algoritmos de classificação

O fluxo de trabalho para o desenvolvimento dos modelos de classificação foi dividido em seis etapas sequenciais, conforme ilustrado de forma esquemática na Figura 1. Cada uma dessas etapas é descrita em detalhes nas seções a seguir.

3.4.1 Ler os dados

Nessa etapa, realizou-se a leitura dos dados armazenados em um arquivo com extensão CSV, que foram então convertidos para uma estrutura do tipo *dataframe*. Esse formato facilitou o processamento e a manipulação dos dados nas etapas seguintes. Os atributos presentes nesse conjunto de dados incluíam: ID,

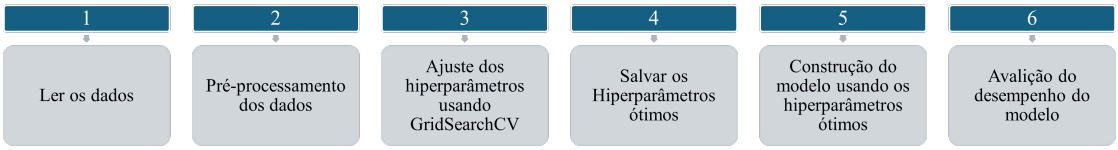

Figura 1 – Estágios de desenvolvimento do modelo.

Prog, Rep, Parc, Arv, AACPDM, formato, cor_casca, cor_polpa, NF, PT, Comp, Diam, ind_form, PF, PPB, RP, EC e BRIX.

3.4.2 Pré-processamento dos dados

Nessa fase, verificou-se se as linhas de dados possuíam duplicatas ou valores nulos em suas colunas. Em seguida, os dados foram normalizados para reduzir a redundância e garantir a consistência e a confiabilidade das informações. Foram utilizadas apenas as variáveis quantitativas no modelo, excluindo-se as variáveis categóricas, como formato, cor_casca e cor_polpa, bem como as colunas relacionadas às informações do experimento, como Prog, Rep, Parc e Arv. Além disso, para assegurar resultados de avaliação precisos, os dados foram divididos em três conjuntos: treino e teste, seguindo proporções previamente definidas e apresentadas na Tabela 1.

Tabela 1 – Divisão dos dados em conjuntos de treino e teste

Tipo de dados	**Percentagem**	**Quantidade de dados**
Treino	80%	69
Teste	20%	18
Total	100%	87

3.4.3 Ajuste dos hiperparâmetros usando GridSearchCV

Nessa etapa, o ajuste de hiperparâmetros teve como objetivo identificar os valores que maximizavam o desempenho do modelo. Para isso, utilizou-se a técnica de *Grid Search Cross-Validation* (GridSearchCV), conforme descrito por Siji George C. G. e B. Sumathi (2020). Essa ferramenta, disponível no módulo *scikit-learn*,

permitiu otimizar os hiperparâmetros de forma automática e sistemática, avaliando múltiplas combinações de parâmetros e selecionando aquelas que proporcionavam os melhores resultados em termos de acurácia, precisão, *recall* e *F1-score*.

Além disso, o GridSearchCV possibilitou a validação simultânea de diferentes modelos, garantindo que o ajuste fosse realizado de maneira robusta e confiável. Informações específicas sobre os hiperparâmetros testados para cada algoritmo de classificação podem ser encontradas na Tabela 2, enquanto detalhes adicionais sobre cada hiperparâmetro estão disponíveis na documentação oficial de cada algoritmo.

Tabela 2 – Hiperparâmetros testados para cada algoritmo de classificação[1].

Algoritmo	**Hiperparâmetros Testados**
Random Forest	• *n_estimators*: 50, 100, 200, 300, 500 • *max_depth*: None, 10, 20, 30 • *min_samples_split*: 2, 5, 10, 15 • *min_samples_leaf*: 1, 2, 4 • *bootstrap*: True, False
Gradient Boosting	• *n_estimators*: 50, 100, 200, 300 • *learning_rate*: 0.01, 0.05, 0.1, 0.2 • *max_depth*: 3, 5, 7, 10 • *min_samples_split*: 2, 5, 10 • *min_samples_leaf*: 1, 2, 4

Tabela 2 – Continuação

Algoritmo	**Hiperparâmetros Testados**
AdaBoost	• *n_estimators*: 50, 100, 200, 300 • *learning_rate*: 0.01, 0.1, 0.5, 1 • *algorithm*: SAMME
SVM	• *C*: 0.01, 0.1, 1, 10, 100 • *kernel*: linear, rbf, poly, sigmoid • *gamma*: scale, auto, 0.1, 1, 10
KNN	• *n_neighbors*: 3, 5, 7, 9, 11, 15 • *weights*: uniform, distance • *metric*: euclidean, manhattan, minkowski
Redes Neurais (MLP)	• *hidden_layer_sizes*: (50,), (100,), (50, 50), (100, 50), (100, 100) • *activation*: logistic, tanh, relu • *solver*: sgd, adam • *alpha*: 0.0001, 0.001, 0.01 • *learning_rate*: constant, adaptive • *learning_rate_init*: 0.001, 0.01, 0.1 • *batch_size*: 32, 64, 128

Tabela 2 – Continuação

Algoritmo	**Hiperparâmetros Testados**
GaussianNB	• *var_smoothing*: 1e-9, 1e-8, 1e-7, 1e-6, 1e-5

3.4.4 Salvar os Hiperparâmetros ótimos

Após o ajuste dos hiperparâmetros, identificou-se as configurações que proporcionavam o melhor desempenho para os modelos em desenvolvimento. Esses parâmetros foram armazenados, o que permitiu simplificar e reproduzir de forma eficiente o processo de avaliação dos modelos.

3.4.5 Construção dos modelos usando os hiperparâmetros ótimos

Nessa etapa, cada modelo foi construído utilizando os hiperparâmetros ótimos encontrados na etapa do GridSearchCV. Foram testados e avaliados oito algoritmos de classificação: Regressão Logística, Árvore de Decisão, Random Forest, Gradient Boosting, Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Redes Neurais Artificiais (MLP) e Naive Bayes (GaussianNB). Em seguida, cada modelo foi treinado para ajustar os dados, separando-os em características (features) e rótulos de classe (labels) identificadas na fase de clusterização do projeto. Esse processo foi executado por meio do comando model.fit(), que realizou o treinamento do modelo com os dados fornecidos. A metodologia de cada algoritmo de classificação é detalhada na Tabela 3

3.5 Avaliação do desempenho dos modelos

A avaliação do desempenho dos modelos foi conduzida utilizando métricas de classificação amplamente reconhecidas, como acurácia, precisão, recall, F1-score, matriz de confusão e curva ROC. Essas métricas proporcionam uma análise

[1] AdaBoost (Adaptive Boosting), SVM (Support Vector Machine), KNN (K-Nearest Neighbors), MLP (Multi-Layer Perceptron), GaussianNB (Gaussian Naive Bayes).

Tabela 3 – Resumo dos Algoritmos de Classificação em Aprendizado de Máquina e Suas Principais Funções

Algoritmo	**Função Principal**
Regressão Logística	Classificação binária através de modelagem probabilística usando função logística
Árvore de Decisão	Classificação baseada em regras hierárquicas usando medidas de impureza (Gini/Entropia)
Random Forest	Ensemble de árvores de decisão com bootstrap e seleção aleatória de features para maior robustez
Gradient Boosting	Ensemble sequencial que corrige erros residuais anteriores usando gradiente da função de perda
AdaBoost	Combinação adaptativa de classificadores fracos com ajuste iterativo de pesos das amostras
SVM	Classificação através de hiperplanos ótimos com maximização de margens e kernel trick
KNN	Classificação baseada na similaridade direta com os k vizinhos mais próximos
MLP (Redes Neurais)	Modelagem de relações não-lineares complexas através de múltiplas camadas de neurônios artificiais
Naive Bayes	Classificação probabilística baseada no teorema de Bayes com suposição de independência condicional

abrangente do desempenho dos algoritmos, considerando tanto a capacidade de previsão correta quanto a robustez em relação a falsos positivos e falsos negativos (HAND, 2006).

3.5.1 Acurácia

A acurácia é a métrica mais utilizada para avaliar modelos de classificação e representa a proporção de previsões corretas em relação ao total de previsões realizadas. A fórmula da acurácia é definida por:

$$\text{Acurácia} = \frac{\text{Número de previsões corretas}}{\text{Número total de previsões}}. \tag{3.9}$$

3.5.2 Precisão e Recall

A precisão mede a proporção de previsões positivas que são de fato corretas, enquanto o recall avalia a proporção de casos positivos reais que foram identificados corretamente pelo modelo. As fórmulas para precisão e recall são, respectivamente:

$$\text{Precisão} = \frac{\text{VP}}{\text{VP} + \text{FP}}, \tag{3.10}$$

$$\text{Recall} = \frac{\text{VP}}{\text{VP} + \text{FN}}, \tag{3.11}$$

onde VP são os verdadeiros positivos, FP são os falsos positivos e FN são os falsos negativos.

3.5.3 F1-Score

O F1-score é a média harmônica entre precisão e recall, sendo uma métrica útil quando há um desequilíbrio entre as classes. A fórmula do F1-score é dada por:

$$\text{F1-score} = 2 \cdot \frac{\text{Precisão} \cdot \text{Recall}}{\text{Precisão} + \text{Recall}}. \tag{3.12}$$

3.5.4 Matriz de Confusão

A matriz de confusão é uma tabela que compara as previsões do modelo com os valores reais, permitindo uma análise detalhada dos erros de classificação. Ela é composta por quatro elementos principais: verdadeiros positivos (VP), falsos positivos (FP), verdadeiros negativos (VN) e falsos negativos (FN). A matriz de confusão é visualizada utilizando a função *ConfusionMatrixDisplay* da biblioteca *scikit-learn*, com cores que facilitam a interpretação dos resultados.

3.5.5 Curva ROC e AUC

A curva ROC (*Receiver Operating Characteristic*) é uma representação gráfica que relaciona a taxa de verdadeiros positivos (TPR) com a taxa de falsos positivos (FPR) para diversos limiares de classificação (FAWCETT, 2006). A área sob a curva ROC (AUC) é uma métrica que resume o desempenho do modelo

em um único valor, onde um AUC igual a 1 indica um modelo perfeito, e um AUC de 0.5 sugere um modelo com desempenho equivalente a uma classificação aleatória (HANLEY; MCNEIL, 1982). A Curva ROC é construída com base nas probabilidades preditas pelo modelo, e a AUC é calculada por:

$$\text{AUC} = \int_0^1 \text{TPR}(\text{FPR})\, d\text{FPR}. \tag{3.13}$$

3.5.6 Curva de Aprendizado

A curva de aprendizado é uma ferramenta útil para diagnosticar problemas de overfitting ou underfitting. Ela mostra a evolução da acurácia no conjunto de treinamento e no conjunto de validação em função do tamanho do conjunto de treinamento. A curva de aprendizado é gerada utilizando a função *learning_curve* da biblioteca *scikit-learn*, que divide o conjunto de dados em diferentes tamanhos e calcula a acurácia média para cada tamanho.

A Curva ROC e AUC, bem como a Curva de Aprendizado, foram aplicados exclusivamente aos algoritmos de Redes Neurais Artificiais (RNA) e Naive Bayes.

3.6 Análises Posteriores à Seleção do Modelo

Após a seleção do modelo, foram conduzidas análises complementares para avaliar o desempenho e a importância das variáveis, além de compreender a distribuição dos dados e a relação entre as *features* e o *target*. Essas análises são fundamentais para validar a robustez do modelo, identificar as variáveis mais relevantes e garantir que o modelo generalize adequadamente para novos dados. As etapas realizadas incluíram a visualização da distribuição das *features*, o cálculo da importância das variáveis, a análise de correlação e a seleção de *features* com base no desempenho.

3.6.1 Distribuição das Features por Classe

Para compreender como as *features* se distribuem em relação às classes, foram gerados gráficos de densidade (KDE - *Kernel Density Estimation*) para cada

feature, separando os dados por classe. A densidade de probabilidade é calculada pela função:

$$f(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right), \tag{3.14}$$

onde K é a função kernel (Gaussiana, por padrão), x_i são os pontos de dados, n é o número de pontos e h é a largura de banda. Essa análise permite verificar se as *features* apresentam distribuições distintas para cada classe, o que pode indicar sua capacidade de discriminar entre as diferentes categorias.

3.6.2 Importância das Variáveis

A importância das variáveis foi avaliada utilizando três métodos complementares, cada um com uma abordagem específica:

1. **Diferença das Médias**: Para cada feature, foi calculada a diferença absoluta entre as médias das classes:

$$\text{Diferença das Médias} = |\mu_1 - \mu_0|, \tag{3.15}$$

onde μ_1 e μ_0 são as médias das features para as classes 1 e 0, respectivamente. Essa métrica é útil para identificar features cujos valores médios diferem significativamente entre as classes, sugerindo que elas podem ser relevantes para a classificação.

2. **Importância por Permutação**: A importância por permutação foi calculada utilizando a função `permutation_importance` da biblioteca `scikit-learn` (PEDREGOSA *et al.*, 2011). Essa técnica mede a queda no desempenho do modelo quando os valores de uma feature são permutados, indicando sua relevância para a classificação:

$$\text{Importância por Permutação} = \frac{1}{n} \sum_{i=1}^{n} (\text{Acurácia Original} - \text{Acurácia com Permutação}). \tag{3.16}$$

Esse método é robusto, pois avalia a contribuição de cada *feature* diretamente no desempenho do modelo, sendo menos sensível a relações não lineares ou interações entre as *features*.

3. **Correlação com o Target**: A correlação entre cada feature e o target foi calculada utilizando o método de Pearson:

$$\text{Correlação} = \frac{\text{Cov}(X, y)}{\sigma_X \sigma_y}, \tag{3.17}$$

onde $\text{Cov}(X, y)$ é a covariância entre a feature e o target, e σ_X e σ_y são os desvios padrão de X e y, respectivamente. A correlação mede a relação linear entre as features e o target, sendo útil para identificar features que variam de forma consistente com a variável dependente.

3.6.3 Seleção de Features com Base no Desempenho

A seleção de *features* é uma etapa crucial no desenvolvimento de modelos de *machine learning*, podendo ser realizada tanto no pré-processamento quanto após a modelagem, dependendo do objetivo. No pré-processamento, a seleção de *features* é feita para reduzir a dimensionalidade dos dados, eliminar *features* irrelevantes ou redundantes, e melhorar a eficiência computacional.

A seleção de *features* também pode ser realizada **após a modelagem**, com o objetivo de avaliar a importância das *features* no contexto do modelo treinado. Métodos como a importância por permutação ou análise de coeficientes (em modelos lineares) permitem entender como o modelo utiliza as *features* para fazer previsões. Essa abordagem é particularmente útil para validar a seleção inicial de *features* e identificar aquelas que, apesar de parecerem importantes no pré-processamento, não contribuem significativamente para o modelo final. Além disso, a análise pós-modelagem pode revelar interações entre *features* que não são capturadas durante o pré-processamento.

Para selecionar as features mais relevantes, foi utilizada a técnica `SelectKBest` da biblioteca `scikit-learn`, que seleciona as k features com maior pontuação estatística. A pontuação foi calculada utilizando o teste F de ANOVA:

$$F = \frac{\text{Variância entre Grupos}}{\text{Variância dentro dos Grupos}} \tag{3.18}$$

O teste F de ANOVA avalia a capacidade de uma *feature* em discriminar entre as classes, comparando a variância entre as médias das classes com a variância dentro

de cada classe. *Features* com valores de F mais altos indicam maior capacidade de separar as classes e, portanto, são selecionadas para compor o conjunto final de *features* (PEDREGOSA *et al.*, 2011).

3.7 Ferramentas e Configuração do Ambiente

Os gráficos e imagens apresentados neste trabalho foram gerados utilizando as bibliotecas `matplotlib` e `seaborn` em Python.

Os experimentos foram realizados em um notebook ACER Nitro 5 AN515-57-520Y, equipado com um processador Intel Core i5-11400H (11ª geração), com 6 núcleos, 12 threads, frequência base de 2.7 GHz e turbo de até 4.5 GHz. O sistema conta com 16 GB de memória RAM DDR4, operando a 3200 MHz, e uma placa de vídeo NVIDIA GeForce GTX 3050, com 4 GB de memória dedicada GDDR6. O armazenamento é feito em um SSD de 512 GB NVMe PCIe, e o sistema operacional utilizado é o Windows 11 Home, 64 bits. O ambiente de desenvolvimento consistiu no Visual Studio Code (VSCode) com Python 3.13.1, permitindo a execução eficiente dos modelos e a geração de visualizações de alta qualidade para análise dos resultados.

4 RESULTADOS E DISCUSSÃO

4.1 Análise de *cluster*

4.1.1 Determinação do Número de Clusters Ótimo

4.1.1.1 Método do cotovelo (*Elbow*)

A seguir, são apresentados os resultados da inércia obtidos durante a aplicação do algoritmo *K-Means* para diferentes números de clusters, conforme a tabela 4.

Tabela 4 – Resultados da inércia para diferentes números de *clusters* no método do cotovelo.

Número de Clusters	**Inércia**
1	957,0
2	768,91
3	672,30
4	644,97
5	535,12
6	497,83
7	463,48
8	409,06
9	380,09
10	360,52

Esses valores mostram a diminuição da inércia conforme o número de clusters aumenta, o que é esperado, uma vez que a inércia é uma medida que reflete a dispersão das amostras dentro dos clusters. A inércia tende a diminuir à medida que mais clusters são adicionados, mas o método do cotovelo busca identificar o ponto onde essa diminuição começa a desacelerar, indicando o número ideal de clusters.

A Figura 2 mostra o comportamento da inércia em relação ao número de clusters. Pode-se observar que À medida que o número de *clusters* aumenta, a inércia (ou variância dentro dos *clusters*) diminui. Isso acontece porque, com mais

clusters, os pontos de dados tendem a ficar mais próximos dos seus respectivos centroides.

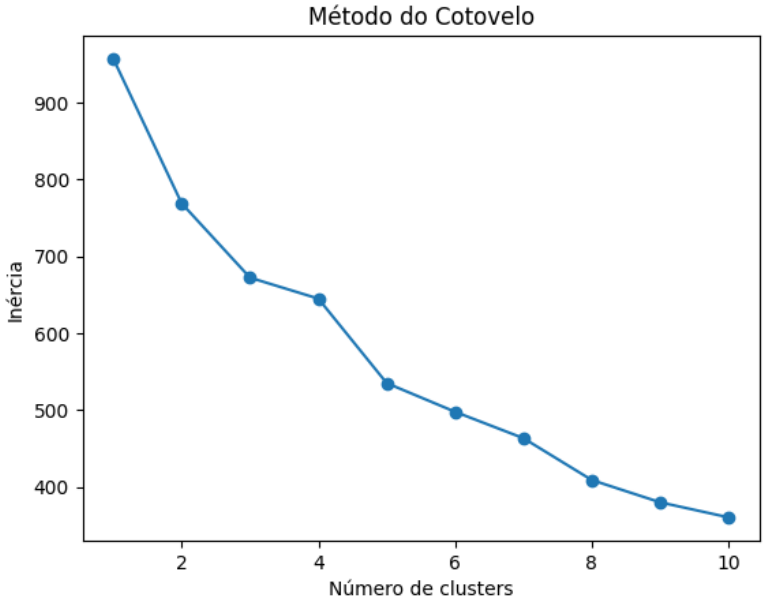

Figura 2 – Curva do cotovelo mostrando a inércia em função do número de *clusters* para o algoritmo *K-Means*.

A determinação do número ideal de *clusters* por meio do método do cotovelo é frequentemente considerada uma tarefa subjetiva, uma vez que se baseia em uma interpretação visual (NAINGGOLAN *et al.*, 2019). A ausência de uma métrica objetiva para identificar com precisão o ponto exato do cotovelo limita a confiabilidade desse método em algumas situaçes.

Embora o método do cotovelo seja útil quando o ponto de inflexão é evidente no gráfico, sua aplicação em casos onde a curva é menos pronunciada ou apresenta múltiplos "cotovelos"pode gerar ambiguidade na escolha do número de *clusters*. A figura 2, por exemplo, ilustra uma situação em que a identificação do ponto ideal torna-se desafiadora, demandando o emprego de outras técnicas complementares.

4.1.1.2 Método da silhueta (*Silhouette Score*)

O maior valor do Índice de Silhueta para $k = 2$, com um score de 0,2081, indicando que a melhor estrutura de agrupamento, entre os valores testados (de

2 a 10), é alcançada com dois *clusters* (Tabela 5). Para $k = 4$, há uma queda acentuada no índice (0,1016), sugerindo que a divisão dos dados em quatro clusters resulta em agrupamentos de qualidade inferior, possivelmente devido à sobreposição significativa entre os *clusters* ou à falta de coesão interna. Entre $k = 5$ e $k = 10$, o Índice de Silhueta oscila, com valores variando entre 0,1189 e 0,1654, sem se aproximar do pico observado em $k = 2$. O aumento discreto nos valores de $k = 9$ (0,1654) e $k = 10$ (0,1636) pode sugerir alguma estrutura adicional, mas a divisão em dois clusters continua sendo a mais indicada, com melhores resultados.

Tabela 5 – Índice de *Silhouette Score* para diferentes valores de k.

Número de Clusters (k)	***Silhouette Score***
2	0.2081
3	0.1978
4	0.1016
5	0.1451
6	0.1234
7	0.1189
8	0.1390
9	0.1654
10	0.1636

O Índice de Silhueta avalia a proximidade de um objeto com seu próprio cluster em comparação com outros clusters. Valores próximos de +1 indicam que os objetos estão bem agrupados, valores perto de 0 sugerem que estão próximos da fronteira entre clusters, e valores próximos de -1 indicam que o objeto pode ter sido atribuído ao cluster errado.

No seu caso, o valor de 0,2081 para $k = 2$ indica que, embora haja alguma sobreposição entre os clusters (já que não está próximo de 1), a divisão em dois grupos ainda representa melhor a estrutura dos dados dentro das opções testadas. A queda para 0,1016 em $k = 4$ sugere um agrupamento de baixa qualidade, com significativa sobreposição entre os clusters. Os valores variando entre 0,1189 e 0,1654 para $k = 5$ a $k = 10$ indicam agrupamentos intermediários, mas consistentemente inferiores à divisão em dois clusters.

Em conclusão, a análise das tabelas e figuras apresenta uma forte evidência de que a melhor escolha para o número de clusters é $k = 2$. O Índice de Silhueta

atinge seu valor máximo nesse ponto, indicando a melhor combinação de coesão interna e separação entre os clusters. Embora haja pequenas flutuações nos valores para $k > 2$, nenhum valor supera o resultado obtido para $k = 2$.

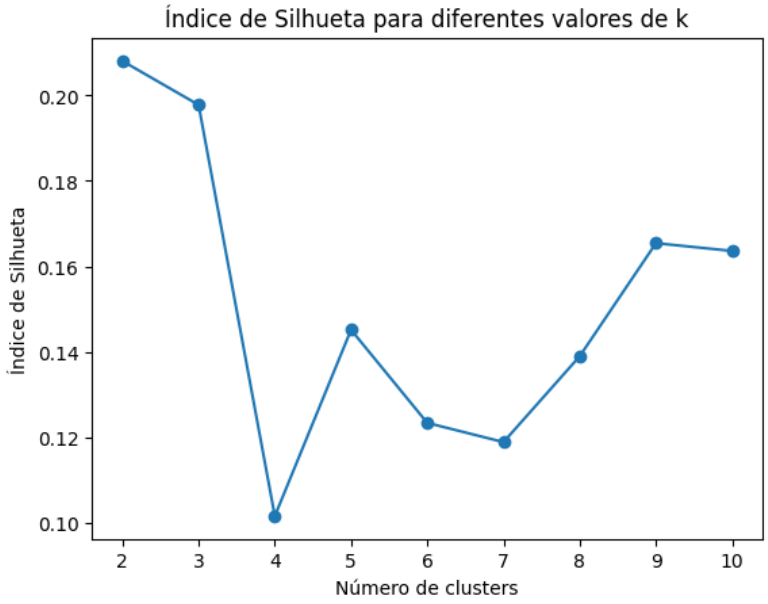

Figura 3 – Gráfico do Índice de *Silhouette Score* para diferentes números de clusters (k).

4.1.1.3 Índice Calinski-Harabasz

Os resultados apontam que o número ideal de *clusters*, segundo o índice de Calinski-Harabasz, é $k = 2$ (Tabela 6). Isso indica que os dados se organizam de maneira mais eficiente em dois grupos bem definidos. O valor elevado do índice para $k = 2$ (20,79) reflete uma boa separação entre os *clusters*, demonstrada por uma alta soma dos quadrados entre grupos (BGSS), e uma forte coesão dentro de cada cluster, evidenciada por uma baixa soma dos quadrados dentro dos grupos (WGSS).

Para valores maiores de k, como $k = 3$ ou $k = 4$, houve uma redução nos valores do índice, indicando que a introdução de mais clusters não melhora a qualidade do agrupamento (Figura4). Na verdade, isso pode levar a uma diminuição na separação entre os *clusters* ou a uma piora na coesão interna. Esses resultados

Tabela 6 – Índice de Calinski-Harabasz para diferentes valores de k.

Número de Clusters (k)	***Calinski-Harabasz Score***
2	20.7926
3	17.7861
4	13.3850
5	16.1621
6	14.9422
7	14.1977
8	15.1174
9	14.7990
10	14.1549

sugerem que forçar o agrupamento em mais *clusters* do que o necessário gera divisões artificiais nos dados ou *clusters* com sobreposição significativa, comprometendo a qualidade do modelo.

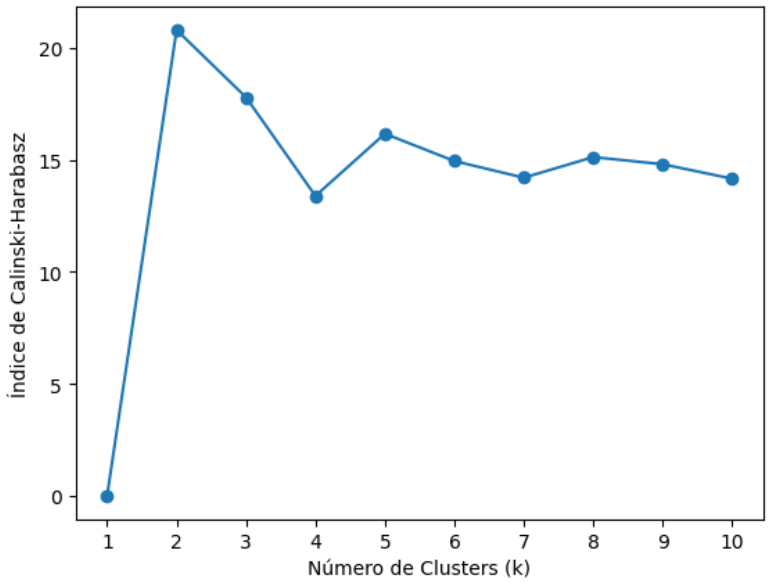

Figura 4 – Índice Calinski-Harabasz para diferentes números de clusters (k) (1 a 10).

A análise das métricas de avaliação de *clustering* indica uma forte concordância em torno de $k = 2$. Tanto o índice de Calinski-Harabasz quanto o *Silhouette Score* apontam para esse número como o ideal para a divisão dos dados. O Calinski-

Harabasz atinge seu valor máximo em $k = 2$, indicando uma boa separação entre os clusters e alta coesão interna. O *Silhouette Score* também é mais alto para $k = 2$ (0,208), sugerindo que os pontos estão bem agrupados dentro de seus respectivos *clusters* e que há uma distinção clara entre os grupos. Embora o valor de 0,208 não seja excepcionalmente alto, ele ainda é o melhor entre os valores analisados e está alinhado com o resultado do Calinski-Harabasz.

Por outro lado, para valores de $k > 2$, enquanto o índice de Calinski-Harabasz mostra uma queda consistente, o *Silhouette Score* apresenta algumas flutuações, com um pequeno aumento para $k = 9$. No entanto, esses valores permanecem abaixo do score obtido para $k = 2$. Essa pequena discrepância pode ocorrer porque as duas métricas avaliam aspectos diferentes do agrupamento. O Calinski-Harabasz se concentra na razão entre a variância entre e dentro dos *clusters* (CALINSKI; HARABASZ, 1974), enquanto o *Silhouette Score* mede a similaridade de um ponto com seu próprio *cluster* em comparação com o *cluster* mais próximo (ROUSSEEUW, 1987). Em conjunto, a forte concordância em torno de $k = 2$ oferece uma evidência robusta de que os dados são melhor agrupados em dois *clusters*, e as flutuações observadas em valores maiores de k sugerem que, embora haja algum ajuste em termos de clusters adicionais, a estrutura geral dos dados favorece claramente a divisão em dois grupos.

4.1.2 Agrupamento com *K-means*

A figura 5 mostra os resultados do algoritmo de agrupamento *K-Means*, aplicado com dois clusters a um conjunto de dados quantitativos sobre qualidade e produtividade dos frutos e resistência de genótipos de maracujazeiro ao CABMV. No gráfico, as cores indicam os dois grupos formados, permitindo observar tanto as distribuições individuais das variáveis quanto suas inter-relações. Essa visualização facilita a análise de como as características avaliadas se organizam nos clusters, oferecendo percepções sobre o desempenho produtivo e a resistência dos genótipos ao vírus, o que é crucial para inferências no contexto do melhoramento genético.

A distribuição das variáveis revela informações significativas sobre os genótipos em cada cluster. A área abaixo da curva de progresso da doença (AACPDM), que é fundamental para avaliar a resistência, mostra uma clara distinção entre os

clusters. Os genótipos do grupo resistente, localizados em um cluster, apresentam valores baixos de AACPDM, o que indica uma progressão mais lenta da doença ao longo do tempo. Por outro lado, o cluster com genótipos mais suscetíveis concentra valores mais altos de AACPDM, refletindo uma maior severidade da doença. Essa separação é crucial, pois genótipos mais resistentes são preferidos em programas de melhoramento genético, como evidenciado por estudos anteriores sobre resistência a viroses em *Passiflora edulis* (GONÇALVES *et al.*, 2021; VIDAL *et al.*, 2021).

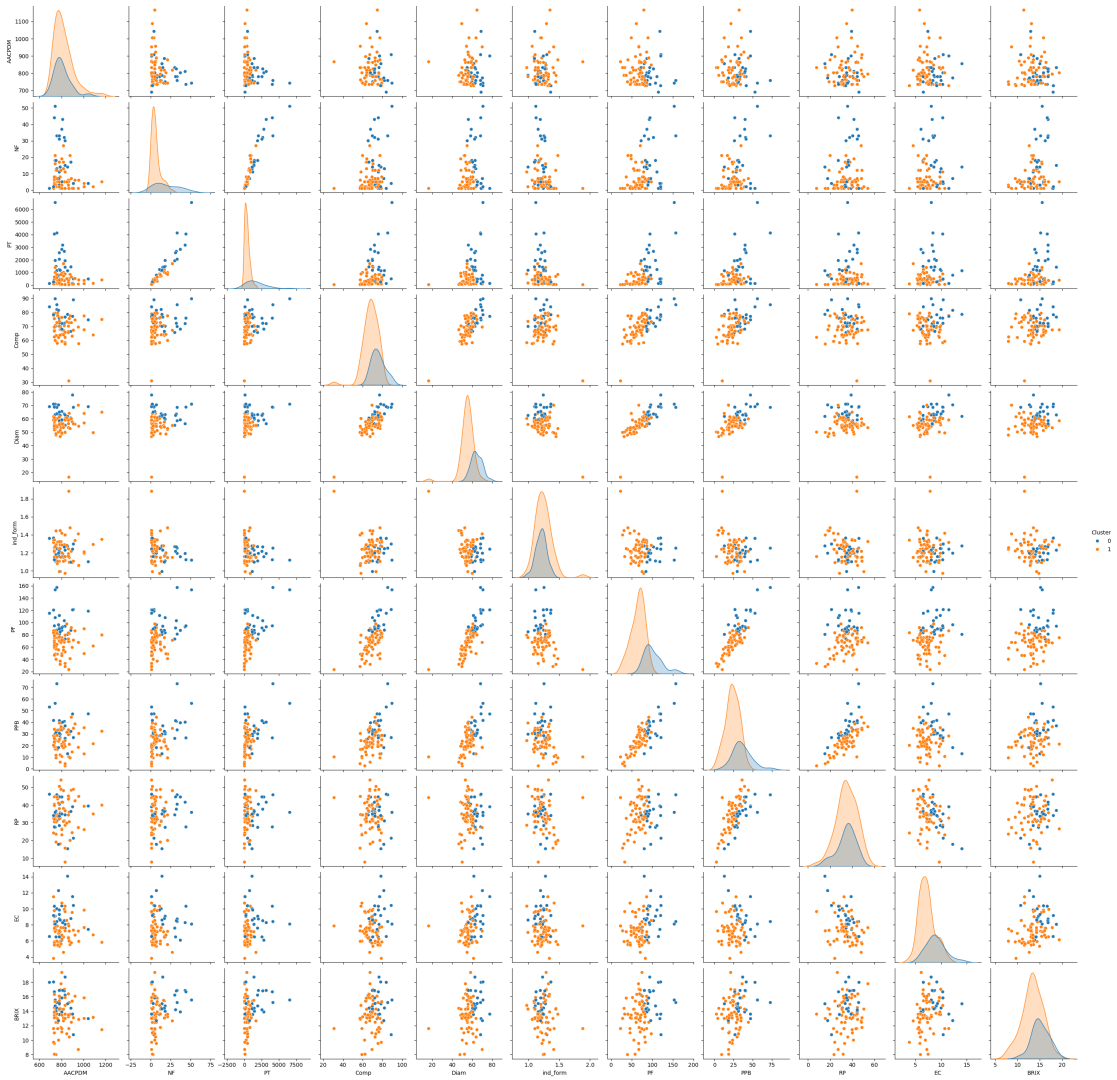

Figura 5 – Clusters gerados pelo algoritmo K-Means.

Além da resistência, as variáveis produtivas, como número de frutos (NF), produção total (PT), peso do fruto (PF) e rendimento de polpa (RP), também exibem padrões de distribuição que favorecem os genótipos resistentes. No *cluster* resistente, essas variáveis apresentam valores consistentemente mais altos, o que sugere que genótipos com menor severidade da doença foram agrupados com os genótipos de maior capacidade produtiva. A produção total, por exemplo, mostra uma correlação positiva com o número de frutos, indicando que o aumento da resistência não só reduz os danos causados pelo CABMV, mas também melhora o potencial produtivo. Esses resultados corroboram achados de estudos anteriores que indicam que a resistência a patógenos em maracujazeiros está frequentemente associada a uma maior produtividade e melhor qualidade dos frutos (GOMES *et al.*, 2022).

A relação entre as variáveis também revela padrões significativos. O peso do fruto (PF) está fortemente associado ao comprimento (Comp) e ao diâmetro do fruto (Diam), com genótipos que apresentam frutos maiores concentrados no cluster mais resistente. Essa associação indica que o tamanho do fruto, uma característica frequentemente valorizada no mercado, pode ser influenciado pela resistência ao CABMV. Além disso, o rendimento de polpa (RP), uma variável de grande importância econômica, é superior no grupo resistente, reforçando a conexão entre resistência e características de qualidade. Por outro lado, variáveis como o teor de sólidos solúveis totais (BRIX) apresentam maior sobreposição entre os clusters, sugerindo que, embora essas características sejam importantes para o mercado consumidor, elas podem não estar diretamente relacionadas à resistência ou à produtividade. Isso pode ser influenciado por fatores genéticos específicos ou condições ambientais, como observado por Gomes *et al.* 2022.

A distribuição das variáveis também revela padrões de variabilidade interna dentro dos *clusters*. A espessura da casca (EC), que pode impactar a durabilidade e o transporte dos frutos, apresenta uma maior variabilidade entre os genótipos do *cluster* resistente. Essa variação pode ser aproveitada no programa de melhoramento, permitindo a seleção de genótipos que atendam tanto às exigências de resistência quanto às preferências do mercado consumidor. Da mesma forma, o índice de formato (ind_form), que indica o quão arredondado é o fruto, mostra uma ampla

distribuição em ambos os *clusters*, sem uma distinção clara entre eles. Isso destaca a complexidade de características morfológicas como o formato do fruto, que podem ser influenciadas por uma combinação de fatores genéticos e ambientais.

A análise das relações entre as variáveis destaca o papel crucial da resistência ao CABMV na determinação do desempenho produtivo. Genótipos mais resistentes não só apresentam menor severidade da doença, mas também se destacam em termos de desempenho produtivo e qualidade comercial. Esse padrão é consistente com os observados de Gomes *et al.* 2022, que ressaltaram a importância da resistência em programas de melhoramento de maracujazeiros como um fator integrador entre produtividade e viabilidade. No entanto, algumas variáveis apresentam interações mais complexas. O teor de sólidos solúveis totais, por exemplo, é uma característica influenciada tanto por fatores genéticos quanto ambientais, como evidenciado por Chavarría-Perez *et al.* 2020, podendo exibir variabilidade significativa até mesmo entre genótipos altamente resistentes.

4.1.3 Avaliação da Estabilidade dos *Clusters*

4.1.3.0.1 Índice de Rand Ajustado (ARI)

O Índice de Rand Ajustado (ARI) foi utilizado para avaliar a consistência dos *clusters* formados. O valor médio obtido para o ARI foi de 0,42 $\pm$ 0,30 para dois clusters. O valor médio de 0,42 sugere uma moderada consistência entre os *clusters* formados em diferentes execuções. Isso indica que, embora haja uma certa concordância entre os agrupamentos, eles não são totalmente idênticos. A presença de um desvio padrão de 0,30 reforça essa interpretação, mostrando que há uma variação significativa nos resultados entre as execuções.

A moderada consistência observada pode ser atribuída a vários fatores como sensibilidade do *K-means* à inicialização, onde o algoritmo *K-means* é conhecido por ser sensível à escolha inicial dos centróides, o que pode levar a resultados diferentes em execuções distintas (KAUFMAN; ROUSSEEUW, 2009); estrutura dos dados, ou seja, se os dados não possuem uma estrutura clara de *clusters*, o algoritmo pode gerar agrupamentos inconsistentes; e número de *clusters*, em que o número de *clusters* escolhido pode não ser o ideal, levando a uma sobreposição ou

má definição dos grupos. Todavia, os resultados para o número ideal de *clusters* mostraram que dois é o mais adequado de *clusters* entre 2 e 10 avaliados. Além disso, a avaliação visual dos *clusters* formados evidencia na prática a divisão bem clara dos genótipos resistentes ao CABMV e com características produtivas ideais.

4.2 Avaliação de algoritmos de classificação

4.2.1 Ajuste dos hiperparâmetros usando GridSearchCV

A Tabela 7 apresenta os melhores hiperparâmetros encontrados para cada modelo, destacando as configurações que maximizaram o desempenho de cada algoritmo.

4.2.1.1 Regressão Logística

A Regressão Logística, um dos métodos mais tradicionais para problemas de classificação binária, foi configurada com os hiperparâmetros ótimos `C=1.0`, `penalty='l2'` e `solver='lbfgs'`, conforme selecionados pelo GridSearchCV e apresentados na Tabela 7. O parâmetro `C=1.0`, que controla a intensidade da regularização, desempenha um papel fundamental no equilíbrio entre a capacidade do modelo de se ajustar aos dados de treinamento e a necessidade de evitar *overfitting*. Valores menores de `C` aumentam a regularização, o que pode ser benéfico em conjuntos de dados com alto risco de *overfitting*, enquanto valores maiores permitem um ajuste mais flexível aos dados (PEDREGOSA *et al.*, 2011). Neste estudo, `C=1.0` demonstrou ser o valor ideal para manter um bom desempenho sem comprometer a generalização do modelo, conforme validado pelo GridSearchCV.

A escolha da penalidade `l2`, também conhecida como regularização Ridge, é responsável por assegurar a estabilidade do modelo. Essa penalidade adiciona uma restrição proporcional ao quadrado dos coeficientes do modelo ao custo de otimização, resultando em coeficientes menores e mais distribuídos. Isso não apenas ajuda a evitar *overfitting*, mas também melhora a generalização do modelo, especialmente em conjuntos de dados onde a multicolinearidade (alta correlação entre variáveis preditoras) pode ser um problema (HASTIE; TIBSHIRANI; FRIEDMAN, 2009). O GridSearchCV confirmou que a penalidade `l2` é a mais adequada para

Tabela 7 – Melhores Parâmetros de Cada Algoritmo. AdaBoost (*Adaptive Boosting*); SVM (*Support Vector Machine*); KNNK (*K-Nearest Neighbors*); RNA MLP (Redes Neurais Artificiais - *Multilayer Perceptron*); GaussianNB (*Gaussian Naive Bayes*).

Algoritmo	**Melhores Parâmetros**
Regressão Logística	{'C': 1.0, 'penalty': 'l2', 'solver': 'lbfgs'}
Árvore de Decisão	{'criterion': 'entropy', 'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'splitter': 'best'}
Random Forest	{'bootstrap': True, 'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 200}
Gradient Boosting	{'learning_rate': 0.05, 'max_depth': 3, 'min_samples_leaf': 4, 'min_samples_split': 2, 'n_estimators': 200}
AdaBoost	{'algorithm': 'SAMME', 'learning_rate': 0.5, 'n_estimators': 50}
SVM	{'C': 1, 'gamma': 'scale', 'kernel': 'sigmoid'}
KNN	{'metric': 'euclidean', 'n_neighbors': 5, 'weights': 'uniform'}
RNA (MLP)	{'activation': 'tanh', 'alpha': 0.0001, 'batch_size': 32, 'hidden_layer_sizes': (50,), 'learning_rate': 'constant', 'learning_rate_init': 0.1, 'solver': 'adam'}
Naive Bayes (GaussianNB)	{'var_smoothing': 1e-09}

o conjunto de dados em questão, garantindo um equilíbrio entre viés e variância (JAMES *et al.*, 2013).

Por fim, o solver `lbfgs` (*Limited-memory Broyden–Fletcher–Goldfarb–Shanno*) traz eficiência em problemas de classificação com conjuntos de dados de tamanho moderado, o que é importante no presente trabalho. Esse algoritmo é particularmente adequado para problemas com regularização `l2`, como no caso deste estudo, e é conhecido por sua convergência rápida e uso eficiente de memória (PEDREGOSA *et al.*, 2011).

4.2.1.2 Árvore de Decisão

Os hiperparâmetros ótimos selecionados pelo *GridSearchCV* para o algoritmo de Árvore de Decisão foram: `{'criterion': 'entropy', 'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'splitter': 'best'}`, conforme detalhado na Tabela 7. O critério `entropy` foi escolhido para avaliar a qualidade das divisões na árvore, sendo uma métrica que mede a impureza dos nós com base na distribuição das classes. Esse critério é especialmente útil em problemas de classificação onde a separação entre classes não é linear, como no caso da seleção de genótipos resistentes ao CABMV e produtivos (BREIMAN *et al.*, 1984).

O parâmetro `max_depth` foi definido como `None`, o que permite que a árvore cresça até que todas as folhas sejam puras ou até que outras condições de parada sejam atingidas. Isso pode resultar em uma árvore mais complexa, mas, combinado com `min_samples_leaf = 1` e `min_samples_split = 2`, garante que a árvore capture padrões detalhados nos dados sem restrições excessivas. O `splitter` definido como `best` assegura que, em cada divisão, o algoritmo escolha a melhor característica para maximizar a pureza dos nós resultantes (PEDREGOSA *et al.*, 2011).

Essa configuração de hiperparâmetros resultou em um modelo de Árvore de Decisão altamente adaptável, capaz de capturar relações complexas entre as variáveis preditoras e a resistência ao CABMV. No entanto, é fundamental monitorar o risco de *overfitting*, especialmente em conjuntos de dados menores, o que ocorre nesse caso, onde a árvore pode se ajustar excessivamente aos dados de treinamento. A combinação desses parâmetros reflete um equilíbrio entre a capacidade de modelagem e a generalização, essencial para aplicações práticas em melhoramento genético necessárias no presente estudo (GONçALVES *et al.*, 2021).

4.2.1.3 Random Forest

Os hiperparâmetros ótimos selecionados pelo *GridSearchCV* para o algoritmo de Random Forest foram: `{'bootstrap': True, 'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 200}`, conforme apresentado na Tabela 7. O parâmetro `n_estimators = 200` indica que o

modelo foi configurado com 200 árvores no *ensemble*, o que geralmente melhora a precisão e a robustez do modelo, reduzindo a variância e o risco de *overfitting* (BREIMAN, 2001b). A técnica de `bootstrap = True` garante que cada árvore seja treinada em um subconjunto aleatório dos dados, aumentando a diversidade do *ensemble* e, consequentemente, sua capacidade de generalização.

O parâmetro `max_depth = None` permite que as árvores individuais cresçam até que todas as folhas sejam puras ou até que outras condições de parada sejam atingidas. Isso, combinado com `min_samples_leaf = 1` e `min_samples_split = 2`, resulta em árvores profundas e complexas, capazes de capturar padrões detalhados nos dados. No entanto, a natureza do *Random Forest*, que combina múltiplas árvores, ajuda a reduzir o risco de *overfitting* que poderia surgir com árvores individuais muito complexas (PEDREGOSA *et al.*, 2011). Essa configuração de hiperparâmetros resultou em um modelo de Random Forest altamente eficaz, com alta acurácia e capacidade de generalização, conforme evidenciado pelos resultados apresentados na Tabela 7.

4.2.1.4 Gradient Boosting

Os hiperparâmetros ótimos selecionados pelo *GridSearchCV* para o algoritmo de Gradient Boosting foram: `{'learning_rate': 0.05, 'max_depth': 3, 'min_samples_leaf': 4, 'min_samples_split': 2, 'n_estimators': 200}`, conforme detalhado na Tabela 7. A taxa de aprendizado (`learning_rate = 0.05`) foi ajustada para um valor relativamente baixo, o que permite que o modelo aprenda de forma mais gradual e precisa, reduzindo o risco de sobreajuste enquanto mantém uma boa capacidade de generalização (FRIEDMAN, 2001). Esse valor, combinado com `n_estimators = 200`, garante que o modelo tenha um número suficiente de iterações para capturar padrões complexos nos dados sem comprometer a eficiência computacional.

O parâmetro `max_depth = 3` limita a profundidade das árvores individuais, criando modelos mais simples e interpretáveis, enquanto `min_samples_leaf = 4` e `min_samples_split = 2` controlam o crescimento das árvores, evitando divisões excessivas que poderiam levar a folhas com poucas amostras. Essa configuração equilibra a complexidade do modelo com a necessidade de evitar *overfitting*, especial-

mente em conjuntos de dados com características multivariadas, como os utilizados neste estudo (PEDREGOSA *et al.*, 2011). A escolha desses hiperparâmetros reflete uma estratégia cuidadosa para otimizar o desempenho do *Gradient Boosting* em tarefas de classificação. Além disso, a combinação de uma taxa de aprendizado moderada com um número elevado de estimadores permite que o modelo refine suas previsões de forma iterativa, resultando em uma performance superior em comparação com métodos mais simples.

4.2.1.5 AdaBoost (*Adaptive Boosting*)

Os hiperparâmetros ótimos selecionados pelo *GridSearchCV* para o algoritmo de AdaBoost foram: `{'algorithm': 'SAMME', 'learning_rate': 0.5, 'n_estimators': 50}`, conforme apresentado na Tabela 7. O uso do algoritmo `SAMME` (*Stagewise Additive Modeling using a Multi-class Exponential loss function*) é especialmente adequado para problemas de classificação multiclasse, como a seleção de genótipos resistentes ao CABMV e produtivos, pois permite que o modelo ajuste iterativamente os pesos das amostras mal classificadas, priorizando aquelas que são mais difíceis de prever (FREUND; SCHAPIRE, 1997). Essa abordagem iterativa é reforçada pelo `learning_rate = 0.5`, que controla a contribuição de cada classificador fraco ao modelo final. Um valor moderado como 0.5 garante que o modelo aprenda de forma eficiente, sem ser excessivamente conservador ou agressivo em suas atualizações.

O número de estimadores (`n_estimators = 50`) foi ajustado para um valor relativamente baixo, o que sugere que o AdaBoost conseguiu alcançar um bom desempenho com um número limitado de iterações. Isso pode ser atribuído à eficácia do `SAMME` em combinar classificadores fracos de forma a maximizar a precisão global do modelo. A interação entre o `learning_rate` e o `n_estimators` é crucial: um `learning_rate` mais alto permite que cada classificador contribua de forma mais significativa, reduzindo a necessidade de um grande número de estimadores para alcançar a convergência (ZHU *et al.*, 2009).

Essa configuração de hiperparâmetros resultou em um modelo de AdaBoost que equilibra eficiência computacional e capacidade de generalização. O uso de um `learning_rate` moderado e um número reduzido de estimadores indica que

o modelo é capaz de capturar padrões importantes nos dados sem se tornar excessivamente complexo.

4.2.1.6 Máquina de Vetores de Suporte (SVM)

Os hiperparâmetros ótimos selecionados pelo *GridSearchCV* para o algoritmo de Support Vector Machine (SVM) foram: `{'C': 1, 'gamma': 'scale', 'kernel': 'sigmoid'}`, conforme detalhado na Tabela 7. O parâmetro `C = 1` define um equilíbrio entre a maximização da margem de separação e a minimização do erro de classificação. Um valor intermediário como 1 sugere que o modelo não é excessivamente restritivo, permitindo uma margem de separação flexível que se adapta bem a dados com ruídos ou sobreposição entre classes (CORTES; VAPNIK, 1995).

O kernel `sigmoid` foi escolhido para mapear os dados para um espaço de maior dimensionalidade, onde a separação entre classes pode ser mais clara. Esse kernel é particularmente útil quando a relação entre as variáveis preditoras e a variável alvo não é linear, mas ainda pode ser capturada por uma função de ativação sigmoide. A escolha de `gamma = 'scale'` garante que o parâmetro de escala do kernel seja ajustado automaticamente com base na variância dos dados, o que ajuda a evitar problemas de *overfitting* ou *underfitting* (PEDREGOSA *et al.*, 2011).

A combinação desses hiperparâmetros resulta em um modelo de SVM que é ao mesmo tempo robusto e eficiente. O kernel sigmoide, aliado a um valor moderado de `C`, permite que o modelo capture relações complexas entre as variáveis preditoras e os genótipos resistentes ao CABMV e produtivos, enquanto o ajuste automático de `gamma` garante que o modelo generalize bem para novos dados.

4.2.1.7 K-Vizinhos Mais Próximos (KNN)

Os hiperparâmetros ótimos selecionados pelo *GridSearchCV* para o algoritmo de K-Nearest Neighbors (KNN) foram: `{'metric': 'euclidean', 'n_neighbors': 5, 'weights': 'uniform'}`, conforme apresentado na Tabela 7. A métrica `euclidean` calcula a distância entre os pontos de dados, sendo uma das medidas mais comuns e eficazes para problemas de classificação (COVER; HART, 1967).

O parâmetro `n_neighbors = 5` indica que o modelo considera os cinco vizinhos mais próximos para realizar a classificação. Geralmente esse valor é escolhido para um equilíbrio entre a sensibilidade a ruídos (que pode ocorrer com valores menores de `n_neighbors`) e a perda de detalhes locais (que pode ocorrer com valores maiores). Além disso, a configuração `weights = 'uniform'` atribui o mesmo peso a todos os vizinhos, o que simplifica o processo de classificação e evita que pontos muito próximos, mas potencialmente ruidosos, tenham uma influência desproporcional no resultado (ZHANG, 2016a).

A combinação desses hiperparâmetros resulta em um modelo de KNN que é simples, mas eficaz. A métrica euclidiana garante que as distâncias sejam calculadas de forma consistente, enquanto a escolha de `n_neighbors = 5` e `weights = 'uniform'` assegura que o modelo seja robusto a variações locais nos dados.

4.2.1.8 Redes Neurais Artificiais (RNA - *Multilayer Perceptron*)

Os hiperparâmetros ótimos selecionados pelo *GridSearchCV* para o algoritmo de Redes Neurais Artificiais (MLP) foram: `{'activation': 'tanh', 'alpha': 0.0001, 'batch_size': 32, 'hidden_layer_sizes': (50,), 'learning_rate': 'constant', 'learning_rate_init': 0.1, 'solver': 'adam'}`, conforme detalhado na Tabela 7. A função de ativação `tanh` (tangente hiperbólica) foi introduz uma não linearidade ao modelo, permitindo que a rede capture relações complexas entre as variáveis preditoras. Essa função é particularmente eficaz em problemas onde os dados apresentam padrões não lineares, pois mapeia as entradas para um intervalo entre -1 e 1, o que ajuda a evitar problemas de saturação que podem ocorrer com outras funções de ativação, como a sigmoide (HAYKIN, 1999).

O parâmetro `alpha = 0.0001` controla a regularização L2, adicionando uma penalidade aos pesos da rede para evitar *overfitting*. Um valor tão baixo indica que a regularização é suave, permitindo que o modelo mantenha uma alta capacidade de aprendizado sem se tornar excessivamente complexo. Essa escolha é especialmente importante em redes neurais, onde a flexibilidade do modelo pode levar a ajustes excessivos aos dados de treinamento. A combinação de `alpha` com o tamanho da camada oculta `hidden_layer_sizes = (50,)` resulta em uma arquitetura balanceada, com uma única camada de 50 neurônios, suficiente para

capturar padrões relevantes sem aumentar desnecessariamente a complexidade computacional (PEDREGOSA *et al.*, 2011).

O uso do solver `adam` (*Adaptive Moment Estimation*) pode ser considerada uma escolha estratégica, pois combina as vantagens do método de momentum e do RMSprop, adaptando a taxa de aprendizado para cada parâmetro da rede. Isso resulta em uma convergência mais rápida e estável, especialmente em conjuntos de dados de tamanho moderado. A taxa de aprendizado inicial `learning_rate_init = 0.1` é relativamente alta, o que permite que o modelo faça ajustes significativos nos pesos durante as primeiras iterações, enquanto o `learning_rate = 'constant'` garante que essa taxa permaneça fixa ao longo do treinamento, evitando flutuações que poderiam prejudicar a estabilidade do modelo (KINGMA; BA, 2014).

O tamanho do lote (`batch_size = 32`) foi ajustado para um valor intermediário, o que equilibra a eficiência computacional e a precisão do gradiente. Um lote menor permite atualizações mais frequentes dos pesos, mas pode aumentar a variância do gradiente, enquanto um lote maior reduz a variância, mas pode tornar o treinamento mais lento. A escolha de 32 é um compromisso que se mostrou eficaz para o conjunto de dados em questão, permitindo que o modelo generalize bem sem sacrificar o desempenho computacional (GOODFELLOW; BENGIO; COURVILLE, 2016).

Essa configuração de hiperparâmetros resulta em um modelo de MLP que é ao mesmo tempo poderoso e eficiente. A combinação da função de ativação `tanh`, a regularização suave com `alpha = 0.0001`, e o uso do solver `adam` com uma taxa de aprendizado constante cria uma rede neural capaz de aprender padrões complexos sem se tornar excessivamente sensível a ruídos ou sobreajuste. Essa abordagem é particularmente útil em problemas onde a relação entre as variáveis preditoras e a variável alvo é altamente não linear, mas ainda requer um modelo que generalize bem para novos dados.

4.2.1.9 Naive Bayes Gaussian (GaussianNB)

O hiperparâmetro ótimo selecionado pelo *GridSearchCV* para o algoritmo de Naive Bayes Gaussiano foi: `{'var_smoothing': 1e-09}`, conforme detalhado na Tabela 7. O parâmetro `var_smoothing` é uma técnica crucial no GaussianNB,

pois adiciona uma pequena constante às variâncias das características durante o cálculo das probabilidades condicionais. Isso evita problemas numéricos que podem surgir quando uma característica tem variância zero em uma classe, o que tornaria a probabilidade indefinida. Um valor tão baixo como `1e-09` indica que o modelo está priorizando a precisão das estimativas de probabilidade, sem adicionar ruído excessivo aos dados (MITCHELL, 1997).

A escolha de `var_smoothing = 1e-09` reflete um equilíbrio delicado entre a estabilidade numérica e a fidelidade aos dados originais. Valores muito altos de `var_smoothing` podem suavizar excessivamente as distribuições de probabilidade, resultando em um modelo menos sensível às nuances dos dados. Por outro lado, valores muito baixos podem levar a instabilidades numéricas, especialmente em características com variâncias próximas de zero. O valor selecionado pelo *GridSearchCV* sugere que o modelo consegue manter uma alta precisão nas estimativas de probabilidade, sem comprometer a robustez do algoritmo (PEDREGOSA *et al.*, 2011).

A combinação desses elementos resulta em um modelo de Naive Bayes Gaussiano que é ao mesmo tempo simples e eficaz. A suavização de variância com `var_smoothing = 1e-09` garante que o modelo seja numericamente estável, enquanto a suposição de independência condicional permite que ele faça previsões rápidas e eficientes. Essa configuração é especialmente adequada para problemas de classificação onde a simplicidade e a velocidade do modelo são tão importantes quanto sua precisão, como em análise de dados biológicos (ZHANG, 2004a).

4.2.2 Desempenho dos algoritmos de classificação

4.2.2.1 Acurácia, precisão, recall e F1-score

Os algoritmos de aprendizado de máquina utilizados neste estudo apresentaram desempenho variado nas métricas de avaliação, incluindo acurácia, precisão, recall e F1-score. Esses resultados são apresentados na Figura 6.

O *Naive Bayes* (GaussianNB) destacou-se como o algoritmo de melhor desempenho, atingindo acurácia, precisão, recall e F1-score perfeitos, todos com valor 1.0. Esses resultados excepcionais podem ser atribuídos à suposição de independên-

cia condicional entre as variáveis, que mostrou-se adequada para este conjunto de dados (ZHANG, 2004b). Além disso, a utilização do parâmetro de suavização de variância (*var_smoothing*) garantiu a estabilidade numérica do modelo, mesmo em cenários com variâncias muito pequenas (MURPHY, 2012).

Os algoritmos de Regressão Logística, Árvore de Decisão, *Random Forest*, *Gradient Boosting* e Redes Neurais Artificiais (RNA) apresentaram desempenho semelhante, com acurácia e F1-scores de 0.94. Essa uniformidade sugere que o conjunto de dados possui uma estrutura bem definida e características adequadas para classificação, permitindo que modelos diversos atinjam alta precisão (BREIMAN, 2001b; FRIEDMAN; HASTIE; TIBSHIRANI, 2001). Modelos baseados em *ensemble*, como *Random Forest* e *Gradient Boosting*, são reconhecidos por sua robustez e capacidade de lidar com dados não-lineares, enquanto Redes Neurais Artificiais destacam-se pela flexibilidade em capturar padrões complexos (LECUN; BENGIO; HINTON, 2015).

O *Support Vector Machine* (SVM) apresentou resultados competitivos, com acurácia de 0.94, precisão de 0.93, recall de 1.0 e F1-score de 0.96. Esses valores refletem a capacidade do SVM de identificar classes positivas com alta sensibilidade, mesmo em conjuntos de dados de alta dimensionalidade, devido à sua abordagem de maximização de margens (VAPNIK, 1998). Esse desempenho confirma a eficácia do SVM em cenários onde os dados são bem separados, mas com algumas variações na precisão.

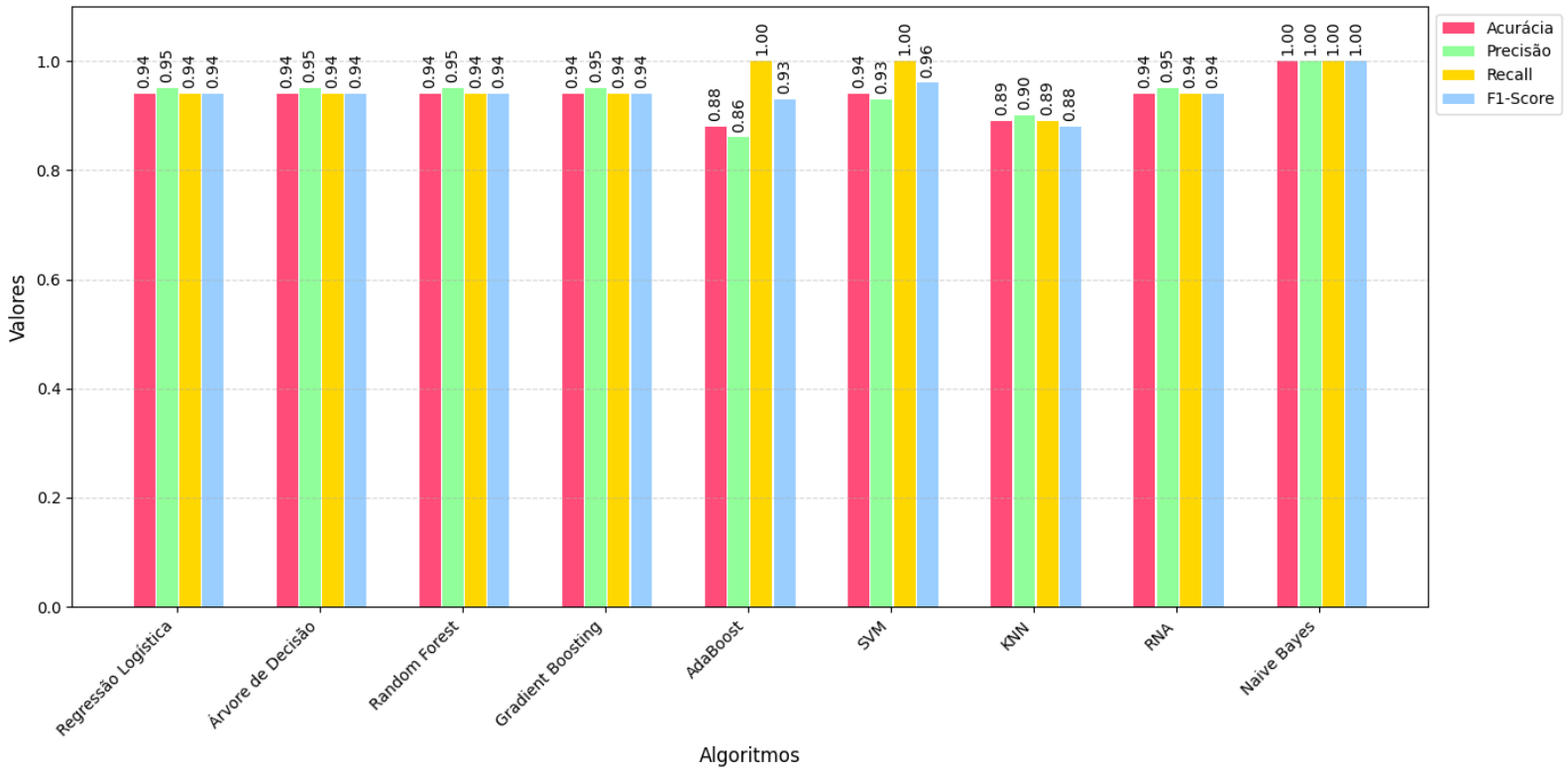

Figura 6 – Desempenho dos Algoritmos de Classificação. AdaBoost (*Adaptive Boosting*); SVM (*Support Vector Machine*); KNN (*K-Nearest Neighbors*); RNA MLP (Redes Neurais Artificiais - *Multilayer Perceptron*); GaussianNB (*Gaussian Naive Bayes*).

Por outro lado, o AdaBoost e o *K-Nearest Neighbors* (KNN) apresentaram desempenhos ligeiramente inferiores. O AdaBoost alcançou acurácia de 0.88 e recall perfeito de 1.0, mas precisou sacrificar a precisão (0.86), resultando em um F1-score de 0.93. Esse padrão reflete a característica do AdaBoost de priorizar a correção de erros em classificações anteriores, o que pode levar ao aumento de falsos positivos em alguns cenários (FREUND; SCHAPIRE, 1997). Já o KNN demonstrou acurácia de 0.89 e F1-score de 0.88. Apesar de apresentar um desempenho equilibrado entre precisão e recall, o KNN mostrou-se limitado pela sensibilidade à escolha do número de vizinhos (k=5) e à métrica de distância utilizada (ALTMAN, 1992).

A análise das métricas de avaliação revelou que, com exceção do AdaBoost e do KNN, todos os algoritmos alcançaram acurácia elevada, variando entre 0.94 e 1.0. Isso reflete tanto a qualidade do conjunto de dados quanto a eficácia das técnicas de pré-processamento, como padronização e seleção de variáveis. O *Naive Bayes*, com precisão perfeita, destacou-se pela capacidade de minimizar falsos positivos, enquanto o recall foi perfeito para o *Naive Bayes*, SVM e AdaBoost, evidenciando sua eficácia em capturar todos os casos positivos. Por fim, o F1-score foi consistentemente elevado, com destaque para o *Naive Bayes* (1.0) e o SVM (0.96), enquanto o KNN apresentou o menor F1-score (0.88), refletindo desafios relacionados à sua sensibilidade e precisão.

Os resultados gerais indicam que o *Naive Bayes* foi o mais eficiente no conjunto de dados avaliado, enquanto *Random Forest* e *Gradient Boosting* se mostraram alternativas robustas com excelente equilíbrio entre simplicidade e desempenho. Além disso, um estudo recente teve como objetivo comparar a eficácia de diferentes algoritmos de aprendizado de máquina na discriminação de variedades e linhas de batata com base em dados espectroscópicos de fluorescência. Nesse contexto, o algoritmo *Naive Bayes* destacou-se ao alcançar uma acurácia média de 95%, evidenciando sua eficácia em aplicações agrícolas. A análise da matriz de confusão revelou que todas as amostras da variedade Sante e da linha S 716 foram corretamente classificadas, ressaltando a capacidade do algoritmo em identificar características distintivas entre as amostras. Esses resultados corroboram a ideia de que o *Naive Bayes* é uma ferramenta poderosa para a classificação de cultivares, especialmente em cenários onde a complexidade dos dados pode ser um desafio.

A combinação de sua simplicidade e robustez torna-o uma escolha valiosa para aplicações práticas na agricultura, onde a precisão na identificação de variedades é crucial para o melhoramento genético e a produção eficiente. (SLAVOVA *et al.*, 2022b).

No geral, os resultadosindicam que o *Naive Bayes* foi o algoritmo mais eficiente no conjunto avaliado, enquanto *Random Forest* e *Gradient Boosting* apresentaram bom equilíbrio entre simplicidade e desempenho. Em um estudo recente, o *Naive Bayes* obteve 95% de acurácia na classificação de variedades de batata com dados espectroscópicos, destacando-se por identificar corretamente todas as amostras das variedades analisadas (SLAVOVA *et al.*, 2022b).

4.2.2.2 Análise das Matrizes de Confusão

As matrizes de confusão para os diferentes algoritmos de classificação são apresentadas na Figura 7. Essas matrizes fornecem uma visão detalhada do desempenho de cada modelo, permitindo a análise de verdadeiros positivos (VP), falsos positivos (FP), verdadeiros negativos (VN) e falsos negativos (FN). A Regressão Logística, Árvore de Decisão, Random Forest, Gradient Boosting, SVM e Redes Neurais Artificiais (RNA) apresentaram matrizes de confusão idênticas, com 4 verdadeiros negativos, 1 falso positivo, 0 falsos negativos e 13 verdadeiros positivos. Esse padrão indica que esses algoritmos foram altamente eficazes na classificação dos genótipos resistentes ao CABMV e produtivos, com uma taxa de acerto de 94% e uma taxa de falsos positivos de aproximadamente 6%. A consistência desses resultados reforça a robustez desses métodos para a tarefa em questão, especialmente quando combinados com técnicas de pré-processamento e ajuste de hiperparâmetros (PEDREGOSA *et al.*, 2011).

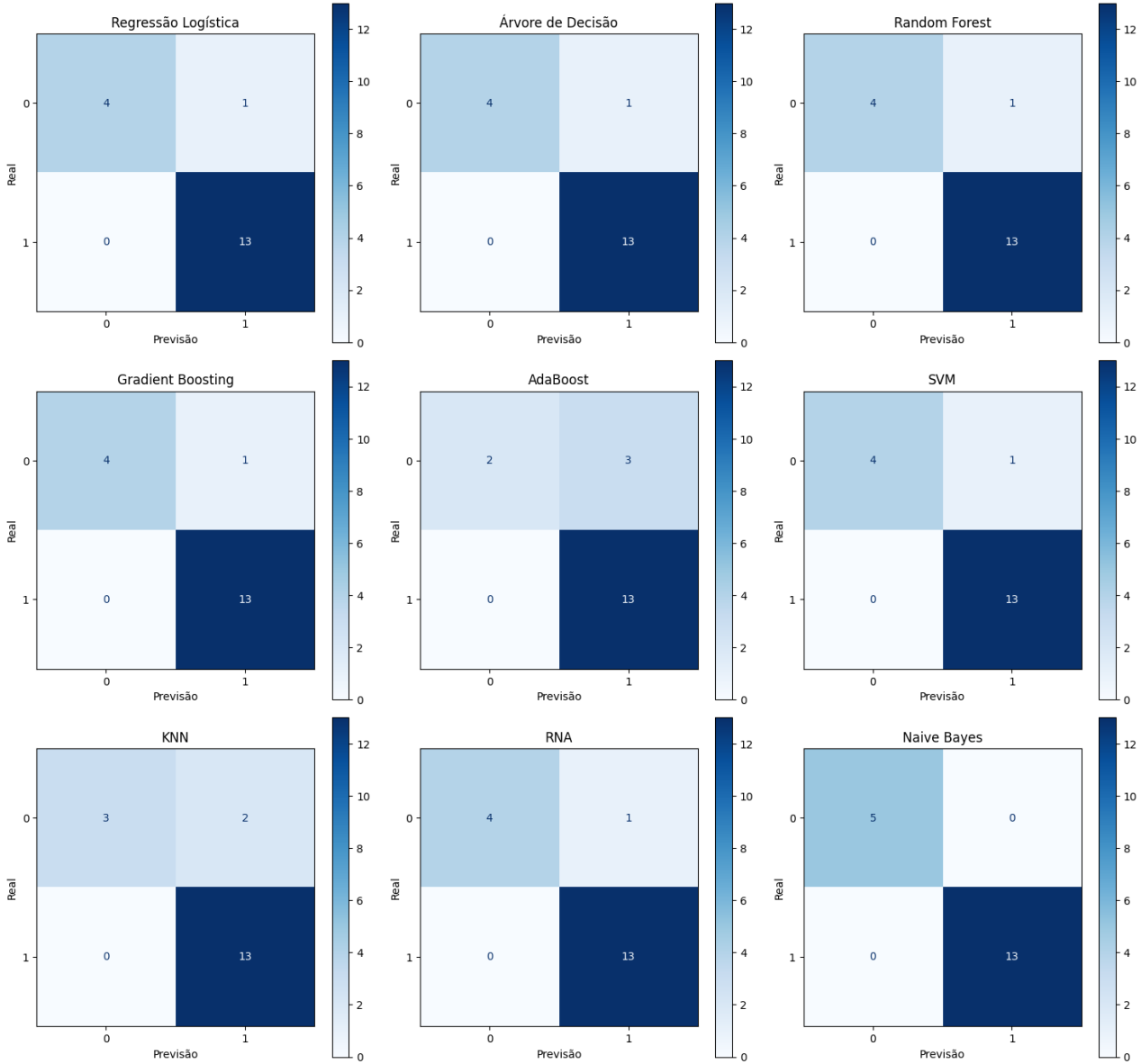

Figura 7 – Matrizes de Confusão dos Algoritmos de classificação. AdaBoost (*Adaptive Boosting*); SVM (*Support Vector Machine*); KNN (*K-Nearest Neighbors*); RNA MLP (Redes Neurais Artificiais - *Multilayer Perceptron*); GaussianNB (*Gaussian Naive Bayes*).

O algoritmo AdaBoost apresentou uma matriz de confusão com 2 verdadeiros negativos, 3 falsos positivos, 0 falsos negativos e 13 verdadeiros positivos. Esse resultado reflete um desempenho ligeiramente inferior em comparação com os outros algoritmos, com uma taxa de falsos positivos de 60%. Esse comportamento pode ser atribuído à natureza iterativa do AdaBoost, que, ao priorizar a correção de erros em classificações anteriores, pode levar a um ajuste excessivo (*overfitting*) em certos cenários (FREUND; SCHAPIRE, 1997). Apesar disso, a ausência de falsos negativos indica que o modelo foi capaz de identificar todos os casos positivos, o que pode ser crucial em aplicações onde a detecção de todos os genótipos resistentes é prioritária.

O *K-Nearest Neighbors* (KNN) apresentou uma matriz de confusão com 3 verdadeiros negativos, 2 falsos positivos, 0 falsos negativos e 13 verdadeiros positivos. Esse resultado reflete uma taxa de falsos positivos de 40%, o que é superior à maioria dos outros algoritmos. A sensibilidade do KNN à escolha do número de vizinhos (k) e à métrica de distância utilizada pode explicar esse desempenho. No presente estudo, o valor ótimo de k foi determinado como 5, o que pode não ser suficiente para capturar completamente a complexidade dos dados (ZHANG, 2016b). Apesar disso, a ausência de falsos negativos e a alta taxa de verdadeiros positivos indicam que o KNN ainda é uma técnica viável para a classificação de genótipos resistentes.

O algoritmo *Naive Bayes* (GaussianNB) destacou-se com uma matriz de confusão perfeita, com 5 verdadeiros negativos, 0 falsos positivos, 0 falsos negativos e 13 verdadeiros positivos. Esse resultado excepcional confirma o desempenho superior do *Naive Bayes* em comparação com os outros algoritmos, alcançando uma taxa de acerto de 100%. A suposição de independência condicional entre as características, combinada com a aplicação de suavização de variância (*var_smoothing*), mostrou-se altamente eficaz para o conjunto de dados em questão (MITCHELL, 1997). Esse desempenho reforça a utilidade do *Naive Bayes* em problemas de classificação onde as suposições do modelo são válidas e os dados seguem uma distribuição Gaussiana.

Em síntese, a análise das matrizes de confusão confirma que a maioria dos algoritmos testados é capaz de classificar eficientemente os genótipos de maracujazeiro resistentes ao CABMV, com destaque para o *Naive Bayes*, que apresentou

um desempenho perfeito. A escolha do algoritmo ideal deve considerar não apenas as métricas de desempenho, mas também a interpretabilidade do modelo e a capacidade de generalização para novos dados. A análise desses resultados reforça a importância da aplicação de técnicas de aprendizado de máquina no melhoramento genético de plantas, oferecendo ferramentas poderosas para a seleção de genótipos com características desejáveis (DIJK, 2021).

4.2.2.3 Análise das Curvas ROC

As curvas ROC (*Receiver Operating Characteristic*) para o RNA MLP (Figura 8A) e o GaussianNB (Figura 8B) são apresentadas na Figura 8. Ambas as curvas demonstram um desempenho excepcional, com uma área sob a curva (AUC) de 1.00, indicando uma capacidade perfeita de discriminação entre as classes positiva (genótipos resistentes ao CABMV) e negativa (genótipos suscetíveis). Esse resultado confirma a eficácia desses algoritmos na tarefa de classificação, destacando-se como ferramentas poderosas para o melhoramento genético de plantas.

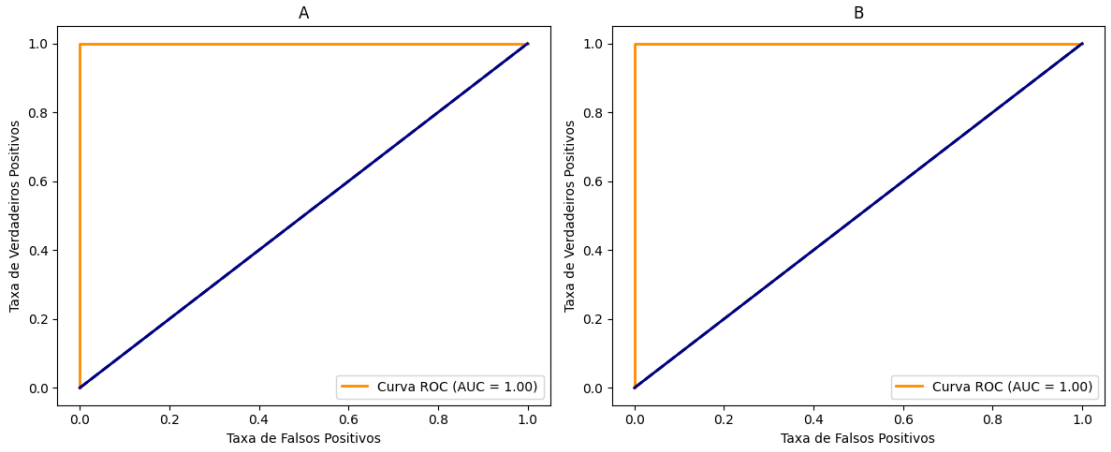

Figura 8 – Curvas ROC para o RNA MLP (A) e o GaussianNB (B).

Na Figura 8A, a curva ROC para o RNA MLP mostra uma taxa de verdadeiros positivos (TPR) de 1.0 e uma taxa de falsos positivos (FPR) de 0.0 no ponto ideal, indicando que o modelo foi capaz de classificar corretamente todos os casos positivos sem gerar falsos positivos. Esse desempenho reflete a

capacidade das redes neurais de capturar relações complexas e não lineares nos dados, especialmente quando combinadas com técnicas de pré-processamento e ajuste de hiperparâmetros (HAYKIN, 1999). A arquitetura do MLP, com múltiplas camadas ocultas e funções de ativação não lineares, mostrou-se altamente eficaz para a tarefa em questão, reforçando a utilidade desse método em problemas de classificação de alta dimensionalidade.

A curva ROC ilustrada na Figura 8B, referente ao classificador *GaussianNB*, exibe um desempenho ideal, com taxa de verdadeiros positivos (TPR) igual a 1.0 e taxa de falsos positivos (FPR) nula. Tal excelência pode ser explicada pela validade das premissas subjacentes ao modelo, particularmente a independência condicional entre as variáveis preditoras e a adequação da distribuição Gaussiana aos dados analisados (MITCHELL, 1997). A implementação de suavização de variância (*var_-smoothing*) desempenhou um papel crucial na estabilização numérica do modelo, assegurando sua robustez e confiabilidade. A superioridade do *GaussianNB* em relação a outros métodos de classificação sublinha a necessidade de alinhar as características intrínsecas dos dados com as premissas teóricas dos algoritmos, destacando a relevância de uma seleção criteriosa de técnicas de aprendizado de máquina.

Em síntese, as curvas ROC confirmam que tanto o RNA MLP quanto o GaussianNB são altamente eficazes na classificação de genótipos de maracujazeiro resistentes ao CABMV. A escolha entre esses algoritmos deve considerar não apenas o desempenho, mas também a complexidade do modelo, a interpretabilidade e a capacidade de generalização para novos dados. A análise desses resultados reforça a importância da aplicação de técnicas de aprendizado de máquina no melhoramento genético de plantas, oferecendo ferramentas poderosas para a seleção de genótipos com características desejáveis (DIJK, 2021).

4.2.2.4 Análise das Curvas de Aprendizado

As curvas de aprendizado ilustradas na Figura 9, correspondentes ao modelo de Rede Neural Artificial *Multilayer Perceptron* (MLP) (Figura 9A) e ao classificador *GaussianNB* (Figura 9B), oferecem uma análise detalhada do comportamento dos modelos em relação ao volume de dados de treinamento. Tais curvas permitem

diagnosticar potenciais problemas, como sobreajuste (*overfitting*) ou subajuste (*underfitting*), ao avaliar a evolução da acurácia em função do aumento do conjunto de treinamento. Em ambos os casos, observa-se uma progressão positiva na precisão dos modelos à medida que mais dados são incorporados, evidenciando uma capacidade consistente de aprendizado e adaptação às características do conjunto analisado.

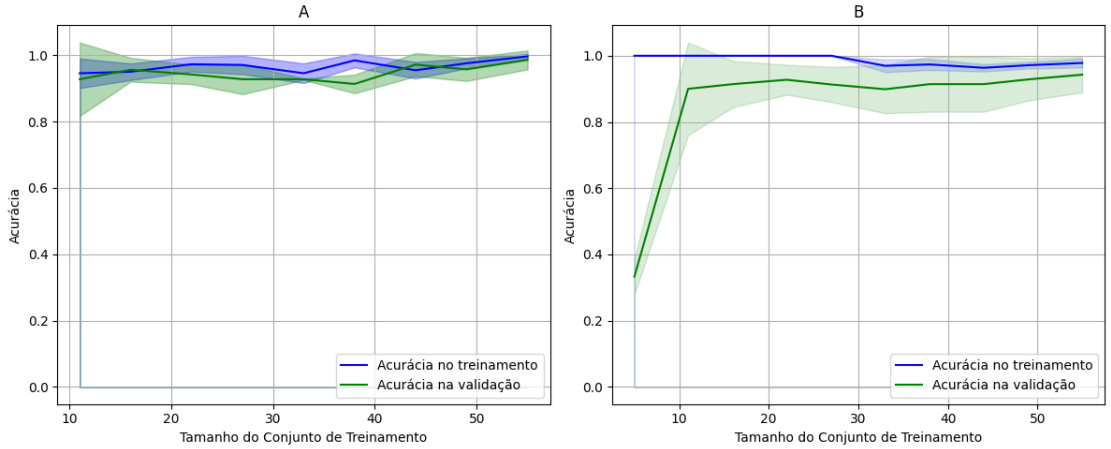

Figura 9 – Curvas de aprendizado para o RNA MLP (A) e o GaussianNB (B).

Na Figura 9A, a curva de aprendizado referente ao modelo de Rede Neural Artificial *Multilayer Perceptron* (RNA MLP) demonstra que as métricas de acurácia, tanto no treinamento quanto na validação, aproximam-se de 1.0 à medida que o volume de dados de treinamento aumenta. Esse padrão sugere uma generalização eficiente do modelo, sem indícios claros de sobreajuste (*overfitting*). A convergência das curvas de treinamento e validação evidencia a capacidade do RNA MLP de identificar padrões intrincados nos dados, mantendo ao mesmo tempo uma performance consistente em conjuntos não vistos (HAYKIN, 1999). Esse desempenho ressalta a adequação das redes neurais para problemas de classificação, especialmente quando associadas a estratégias de regularização e otimização de hiperparâmetros.

Por sua vez, na Figura 9B, a curva de aprendizado do classificador *GaussianNB* também exibe uma convergência entre as acurácias de treinamento e validação, ambas alcançando valores próximos a 1.0. Esse resultado destaca a

robustez do modelo, que, apesar de basear-se em premissas simplificadoras, como a independência condicional entre variáveis, mostrou-se altamente eficaz para o conjunto de dados analisado (MITCHELL, 1997). A ausência de sobreajuste ou subajuste (*underfitting*) indica que o *GaussianNB* está bem calibrado e apresenta uma capacidade satisfatória de generalização. Além disso, a rápida convergência das curvas sugere que esse modelo demanda uma quantidade menor de dados para atingir desempenho ótimo, contrastando com abordagens mais complexas, como o RNA MLP.

Em resumo, as curvas de aprendizado corroboram a eficácia tanto do RNA MLP quanto do *GaussianNB* na tarefa de aprendizado a partir dos dados, sem evidências de sobreajuste ou subajuste. A seleção entre esses métodos deve levar em conta não apenas a performance alcançada, mas também fatores como a complexidade do modelo, sua interpretabilidade e o volume de dados disponível. Esses achados reforçam a relevância da aplicação de técnicas de aprendizado de máquina no contexto do melhoramento genético vegetal, proporcionando ferramentas robustas para a identificação de genótipos com características agronomicamente vantajosas (DIJK, 2021).

4.2.3 Análises pós seleção do algoritmo ótimo

4.2.3.1 Distribuição das *features* por classe

Os gráficos de densidade (*Kernel Density Estimation* - KDE) das diversas características do conjunto de dados estão representados na Figura 10. Esses gráficos possibilitam uma avaliação detalhada da distribuição de cada variável, segmentando os dados conforme as classes de interesse (genótipos resistentes e produtivos, e suscetíveis ao CABMV e não produtivos). A análise dessas distribuições oferece informações cruciais sobre o potencial discriminativo de cada *feature*, permitindo identificar quais atributos contribuem de forma mais significativa para a distinção entre as classes. Essa abordagem é fundamental para compreender a relevância das variáveis no processo de classificação e para orientar a seleção de características mais informativas.

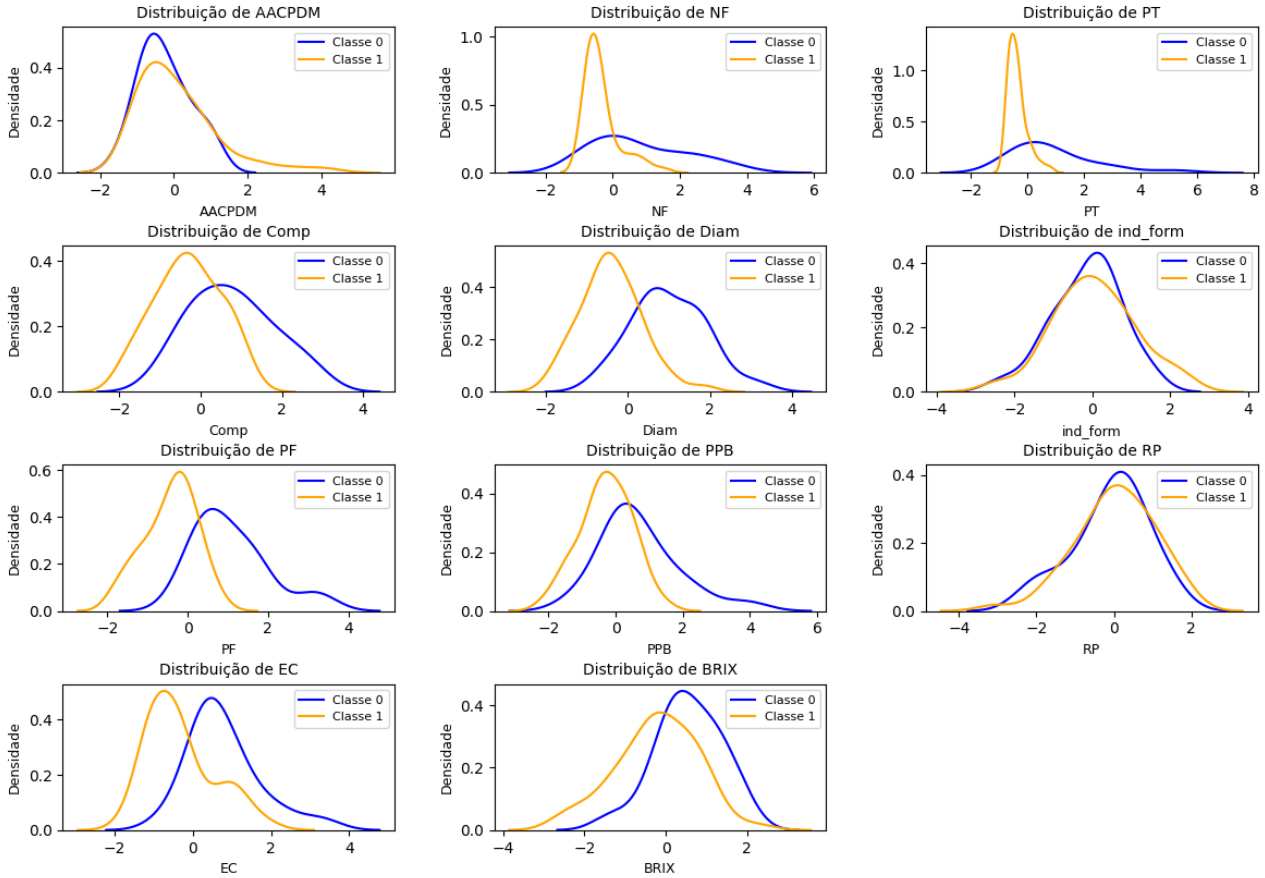

Figura 10 – Gráficos de distribuição (KDE - *Kernel Density Estimation*) para diferentes características do conjunto de dados. AACPDM (Área Abaixo da Curva de Progresso da Doença), NF (Número de Frutos), PT (Peso Total dos Frutos), Comp (Comprimento Médio do Fruto), Diam (Diâmetro do Fruto), Ind_form (Índice de Formato do Fruto), PF (Peso de Fruto), PPB (Peso da Polpa por Fruto), RP (Rendimento de Polpa), EC (Espessura da Casca) e BRIX (Teor de Sólidos Solúveis Totais).

A variável AACPDM exibe uma sobreposição quase completa entre as distribuições das duas classes, com padrões de densidade bastante semelhantes. Esse comportamento sugere que, isoladamente, a AACPDM possui um poder discriminativo limitado para distinguir entre os genótipos. Sua utilidade na classificação provavelmente depende da combinação com outras variáveis mais informativas (GONçALVES *et al.*, 2021).

O Número de Frutos (NF) e o Peso Total dos Frutos (PT) revelam padrões distintos entre as classes (Classe 0 e Classe 1). Ambas as características apresentam distribuições bem separadas, com sobreposição insignificante, indicando um poder discriminativo individual moderado a alto. Essa separação sugere que NF e PT são metricamente relevantes para diferenciar genótipos resistentes (Classe 0) e suscetíveis (Classe 1). Apesar da eficácia individual, a combinação dessas variáveis com outras características produtivas pode aprimorar ainda mais a robustez de modelos de classificação, especialmente em abordagens multivariadas.

O Comprimento do Fruto (Comp) apresenta distribuições claramente separadas, com sobreposição mínima entre as classes, indicando alto poder discriminativo para diferenciar os grupos. Essa separação sugere que o Comp é uma característica morfológica crítica na identificação de genótipos. Por outro lado, o Diâmetro (Diam) exibe uma separação moderada, com sobreposição limitada, apontando para um poder discriminativo individual relevante, porém menos pronunciado em comparação ao Comp. Apesar das diferenças na robustez individual, ambas as variáveis demonstram potencial para contribuir em modelos de classificação. Essas características podem ser particularmente úteis quando associadas a outras variáveis relacionadas ao tamanho e formato do fruto, ampliando sua eficácia na classificação (VIDAL *et al.*, 2021; GOMES *et al.*, 2022).

O Índice de Formato do Fruto (Ind_form) apresenta distribuições com sobreposição entre as classes (Classe 0 e Classe 1), indicando um baixo poder discriminativo individual. As curvas de densidade mostram padrões quase superpostos na maior parte do intervalo de valores (-4 a 4), com diferenças sutis nas extremidades. Apesar da limitação individual, essa variável pode adquirir relevância quando combinada com atributos morfológicos complementares, como comprimento ou diâmetro do fruto.

O Peso do Fruto (PF) e o Peso da Polpa por Fruto (PPB) mostram comportamentos distintos: enquanto o PF apresenta diferenças perceptíveis entre as classes, sugerindo um poder discriminativo moderado, o PPB exibe uma sobreposição significativa, indicando uma capacidade limitada de distinção. No entanto, ambas as características podem ser úteis quando integradas a outras variáveis relacionadas ao peso e à qualidade do fruto.

O Rendimento de Polpa (RP) apresenta distribuições com sobreposição quase completa entre as classes (Classe 0 e Classe 1), indicando um baixo poder discriminativo individual. Essa característica, embora pouco útil isoladamente, pode contribuir em análises multivariadas que integram variáveis produtivas (como Peso Total de Frutos ou Número de Frutos), identificando padrões indiretos.

Em contraste, a Espessura da Casca (EC) exibe separação clara entre as classes, com sobreposição mínima nas curvas de densidade, revelando um alto poder discriminativo. Essa distinção sugere que a EC está diretamente associada a características físicas dos frutos que influenciam na resistência ao CABMV e afetam a produtividade, como a capacidade de proteção contra danos mecânicos ou infecções causadas pelo vetor do vírus. Por fim, o Teor de Sólidos Solúveis Totais (BRIX) exibe clara separação entre as classes (Classe 0 e Classe 1), com sobreposição mínima nas curvas de densidade, indicando alto poder discriminativo.

Em síntese, a análise das distribuições das variáveis revela que características como Comprimento do Fruto (Comp), Espessura da Casca (EC) e Teor de Sólidos Solúveis Totais (BRIX) apresentam alto poder discriminativo, com separação clara entre as classes e sobreposição mínima. Essas variáveis são particularmente relevantes para a classificação de genótipos resistentes e suscetíveis, além de produtivos e não produtivos, destacando-se como atributos críticos em modelos preditivos.

Por outro lado, Número de Frutos (NF) e Peso Total dos Frutos (PT) também exibem poder discriminativo moderado a alto, com distribuições bem separadas e sobreposição limitada, reforçando sua utilidade na distinção entre classes. Já o Diâmetro do Fruto (Diam) apresenta separação moderada, com sobreposição parcial, indicando um desempenho individual relevante, porém menos robusto que Comp, EC e BRIX.

Em contraste, características como AACPDM, Índice de Formato do Fruto (Ind_form), Peso da Polpa por Fruto (PPB) e Rendimento de Polpa (RP) demonstram baixo poder discriminativo individual, com sobreposição significativa ou quase completa entre as classes. No entanto, essas variáveis podem ganhar relevância quando combinadas com outras características complementares.

Por exemplo, AACPDM e Ind_form podem ser integradas a atributos morfológicos (como Comp e Diam) para capturar padrões indiretos de resistência. PPB e RP podem ser associadas a variáveis produtivas (como NF e PT) para aprimorar análises de qualidade e produtividade. A integração estratégica dessas variáveis, aliada à seleção criteriosa das mais informativas, é fundamental para desenvolver modelos de classificação robustos, capazes de capturar padrões complexos associados à resistência ao vírus e à produtividade. Essa abordagem multivariada maximiza a precisão das predições, tornando-se essencial para avanços no melhoramento genético de plantas (DIJK, 2021).

4.2.3.2 Importância das variáveis

A avaliação da relevância das variáveis foi realizada através de duas metodologias integradas: análise comparativa das médias entre classes e mensuração da importância por meio de permutação. Tais estratégias possibilitaram a detecção dos atributos mais significativos para a discriminação de genótipos com resistência ao CABMV e produtivos, conforme demonstrado na Figura 11. A primeira abordagem baseou-se na comparação estatística das médias, enquanto a segunda empregou a permutação para quantificar a contribuição relativa de cada variável, garantindo uma análise robusta e multidimensional.

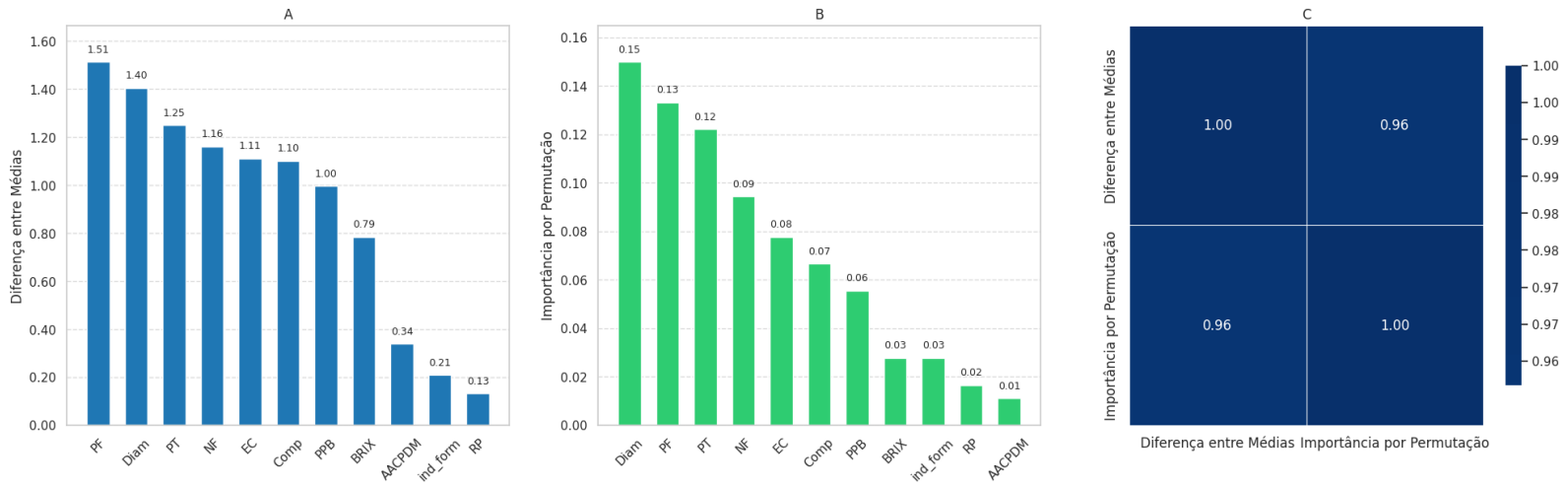

Figura 11 – Análise da importância das variáveis para a classificação de genótipos. (A) Diferença entre as médias das classes para cada variável, ordenadas em ordem decrescente de importância. (B) Importância das variáveis calculada por permutação, ordenadas em ordem decrescente de importância. (C) Correlação de *Spearman* entre os postos das variáveis, representada em um heatmap. AACPDM (Área Abaixo da Curva de Progresso da Doença), NF (Número de Frutos), PT (Peso Total dos Frutos), Comp (Comprimento Médio do Fruto), Diam (Diâmetro do Fruto), Ind_form (Índice de Formato do Fruto), PF (Peso de Fruto), PPB (Peso da Polpa por Fruto), RP (Rendimento de Polpa), EC (Espessura da Casca) e BRIX (Teor de Sólidos Solúveis Totais).

- **Diferença entre médias**: A magnitude das diferenças observadas entre as médias das classes destacou o peso do fruto (PF) como a variável mais discriminatória (1.513), seguida pelo diâmetro (Diam, 1.405) e pelo peso total (PT, 1.250). Tais achados indicam que atributos morfométricos e produtivos apresentam uma correlação significativa com a resistência ao vírus, alinhando-se com evidências documentadas em investigações prévias sobre fenotipagem em *Passiflora* (SANDHU *et al.*, 2021a).

- **Importância por permutação**: A abordagem baseada em permutações reafirmou a importância do PF (0.150) e do Diam (0.133), mas atribuiu maior ênfase ao número de frutos (NF, 0.122) em relação ao método de diferenças médias. Essa inconsistência pode ser explicada pela capacidade do modelo em capturar interações não-lineares, as quais não são explicitamente identificadas em análises univariadas (SLAVOVA *et al.*, 2022a).

A alta correlação de postos de Spearman ($\rho = 0.96$) entre as metodologias empregadas (Figura 11C) demonstra uma consistência notável na identificação das características consideradas críticas, especialmente para o peso do fruto (PF), diâmetro (Diam) e peso total (PT). Essa convergência entre abordagens reforça a confiabilidade das variáveis selecionadas, estando em sintonia com avanços recentes em seleção assistida por técnicas de aprendizado de máquina (DIJK, 2021). No entanto, a divergência observada para o número de frutos (NF), que ocupou o 4º posto em um método e o 3º em outro, evidencia a necessidade de integrar múltiplas estratégias analíticas para abranger distintas dimensões da relevância das variáveis.

4.2.3.3 Correlação com o *target*

A análise de correlação entre as variáveis preditoras e a classe alvo revelou padrões distintos conforme apresentado na Seção 12. Observou-se que o índice de formato do fruto (ind_form) e a área abaixo da curva de progresso da doença (AACPDM) apresentaram correlações positivas fracas (0,132 e 0,080 respectivamente), enquanto características morfométricas como peso do fruto (PF), peso total (PT) e diâmetro (Diam) demonstraram correlações negativas moderadas a fortes (-0,680 a -0,566). Esses resultados sugerem que genótipos classificados como

"Resistente e Produtivo"tendem a apresentar frutos com menores dimensões físicas, padrão consistente com observações prévias em estudos de seleção indireta para resistência a viroses (SANDHU *et al.*, 2021b).

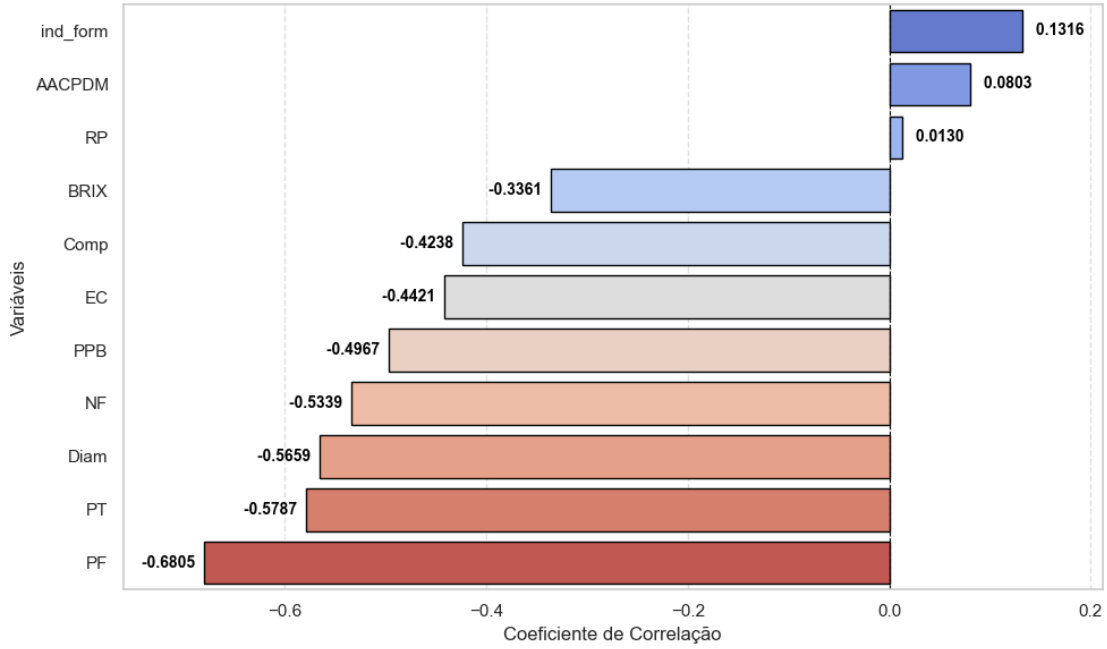

Figura 12 – Heatmap de correlação de Spearman entre variáveis preditoras e classes alvo. Valores positivos (azul) indicam associação com a classe "Resistente e Produtivo", enquanto correlações negativas (vermelho) relacionam-se com "Suscetível e não Produtivo". AACPDM (Área Abaixo da Curva de Progresso da Doença), NF (Número de Frutos), PT (Peso Total dos Frutos), Comp (Comprimento Médio do Fruto), Diam (Diâmetro do Fruto), Ind_form (Índice de Formato do Fruto), PF (Peso de Fruto), PPB (Peso da Polpa por Fruto), RP (Rendimento de Polpa), EC (Espessura da Casca) e BRIX (Teor de Sólidos Solúveis Totais).

A forte correlação negativa do PF (-0,680) com o *target* indica que aumentos no peso médio do fruto estão associados à suscetibilidade ao CABMV e menor produtividade. Esse fenômeno pode ser explicado pelo compromisso energético entre alocação de recursos para defesa fitossanitária e desenvolvimento de estruturas reprodutivas, conforme documentado em estudos de trade-off fisiológico em *Passiflora*

spp. (GONçALVES *et al.*, 2021). Contudo, ressalta-se que correlações univariadas não capturam interações sinérgicas entre características, limitação amplamente discutida em abordagens de seleção assistida por machine learning (SLAVOVA *et al.*, 2022b).

A Figura 12C evidencia alta consistência ($\rho = 0,96$) entre os postos de importância das variáveis obtidos por diferentes metodologias, corroborando a robustez das associações identificadas. Nota-se que o teor de sólidos solúveis (BRIX) apresentou correlação negativa moderada (-0,336), sugerindo que genótipos resistentes possuem menor concentração de açúcares. Esse achado contraria expectativas convencionais sobre qualidade comercial, exigindo análise crítica entre objetivos de melhoramento para resistência e preferências de mercado (LIAKOS *et al.*, 2018).

Apesar do valor descritivo das correlações lineares, destaca-se que métodos multivariados e análises de importância baseadas em permutação são necessários para capturar relações não-lineares entre características (DIJK, 2021). A seleção exclusiva com base em correlações unidimensionais pode negligenciar sinergias entre atributos morfoagronômicos, conforme demonstrado em estudos comparativos com ensembles de aprendizado profundo (SANDHU *et al.*, 2021b).

4.2.3.4 Seleção de *features* com base no desempenho

A análise univariada por ANOVA F-test ($\alpha = 0,05$) identificou como preditores mais relevantes, em ordem decrescente de importância: Número de Fruto, Peso Total de Frutos, Diâmetro do Fruto, Peso do Fruto e Espessura da Casca. Essa hierarquia revela predominância de características morfométricas associadas à produtividade, padrão consistente com estudos em culturas perenes submetidas a estresse biótico (WANG; MOGHIMI; ZHANG, 2023). A exclusão da AACPDM reforça sua baixa discriminância no modelo selecionado. A seleção arbitrária de 5 variáveis, correspondendo a 45% do conjunto original.

A exclusão da Área Abaixo da Curva de Progresso da Doença (AACPDM) durante a seleção de variáveis, apesar de a única variável responsáel pela avaliação de resistência, revela limitações inerentes à classificação exata dos genótipos resistentes. Propõe-se a conversão da AACPDM em variáveis binárias (resistente/suscetível) por meio de limiarização pela mediana populacional, estratégia que reduziria

a sensibilidade a flutuações experimentais e permitiria adaptação contextual a diferentes populações (RIBEIRO; AMORIM, 2023).

5 CONCLUSÃO

A análise de *clustering* com o algoritmo *K-means* permitiu identificar dois grupos principais de genótipos de maracujazeiro: um associado à resistência ao CABMV e alta produtividade, e outro vinculado à suscetibilidade ao vírus e menor desempenho agronômico. A convergência entre os métodos do cotovelo, silhueta e Calinski-Harabasz na definição de $k = 2$ como número ótimo de *clusters* reforçou a consistência da segmentação, apesar da subjetividade inerente ao método do cotovelo. A estabilidade moderada dos *clusters* (ARI $= 0,42 \pm 0,30$) refletiu a sensibilidade do algoritmo à inicialização, porém a clara separação visual entre grupos validou sua relevância biológica.

Na avaliação de algoritmos de classificação, o *GaussianNB* destacou-se com acurácia, precisão, *recall* e F1-score perfeitos $(1,0)$, evidenciando sua adequação às premissas de independência condicional e distribuição Gaussiana dos dados. Modelos *ensemble* como *Random Forest* e *Gradient Boosting*, além de redes neurais (RNA MLP), também apresentaram alto desempenho (acurácia $0,94$), demonstrando robustez na captura de relações não lineares. O SVM mostrou-se ideal para cenários onde a identificação de todos os genótipos resistentes é prioritária, mesmo com custo de falsos positivos. A análise de importância das variáveis destacou o peso do fruto (PF), diâmetro (Diam) e número de frutos (NF) como atributos-chave para a classificação, enquanto a AACPDM, embora crítica para avaliação de resistência in vivo, mostrou baixo poder discriminativo isoladamente, reforçando a necessidade de abordagens multivariadas.

As curvas ROC (AUC $= 1,0$) e de aprendizado (acurácia convergente para $1,0$) confirmaram a capacidade dos modelos em generalizar padrões complexos, sem indícios de *overfitting*. A correlação negativa entre características morfométricas (ex.: PF, Diam) e a classe resistente ($\rho \approx -0,68$) sugeriu um dilema fisiológico entre alocação de recursos para defesa e desenvolvimento de frutos, implicando em desafios para programas de melhoramento que busquem conciliar produtividade e resistência. A integração de técnicas de aprendizado de máquina, como a seleção de *features* por permutação e a análise de distribuições multivariadas, provou-se

essencial para desvendar interações entre fenótipos, oferecendo ferramentas precisas para a seleção assistida de genótipos.

O presente estudo demonstrou que a combinação de *clustering* e classificação supervisionada pode otimizar a seleção de genótipos superiores, reduzindo custos e tempo em programas de melhoramento. A identificação de genótipos resistentes ao CABMV com alta produtividade (ex.: grupo com NF ≈ 25, PT $\approx 4,5$ kg) oferece material genético promissor para cultivos comerciais ou otimização do programa de melhoramento. A superioridade do GaussianNB, aliada à eficiência de modelos interpretáveis como Árvores de Decisão, posiciona essas técnicas como ferramentas acessíveis para laboratórios com recursos computacionais limitados, democratizando o uso de inteligência artificial na agricultura. Por fim, o estudo reforça a importância da fenotipagem precisa e da integração de dados multidisciplinares para avanços no melhoramento de plantas frente a desafios fitossanitários emergentes e graves.

REFERÊNCIAS

ACHARJEE, A. *et al.* Integration of multi-omics data for prediction of phenotypic traits using random forest. v. 17, n. S5, p. 180, 2016.

ACQUAAH, G. **Principles of Plant Genetics and Breeding**. 1. ed. [*S.l.: s.n.*]: Wiley, 2012.

ADLAK, T. *et al.* Biotechnology: An Advanced Tool for Crop Improvement. p. 1–11, 2019.

AHMED, S. *et al.* Artificial intelligence and machine learning in finance: A bibliometric review. v. 61, p. 101646, 2022.

ALTMAN, N. S. An introduction to kernel and nearest-neighbor nonparametric regression. **The American Statistician**, v. 46, n. 3, p. 175–185, 1992.

ANSARIFAR, J.; AKHAVIZADEGAN, F.; WANG, L. Performance prediction of crosses in plant breeding through genotype by environment interactions. v. 10, n. 1, p. 11533, 2020.

AURIA, L.; MORO, R. A. Support Vector Machines (SVM) as a Technique for Solvency Analysis. p. 1–16, 2008.

BREIMAN, L. Random Forests. v. 45, n. 1, p. 5–32, 2001.

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

BREIMAN, L. *et al.* **Classification and Regression Trees**. New York: CRC Press, 1984. ISBN 9780412048418.

C, S. G.; B Sumathi. Grid search tuning of hyperparameters in random forest classifier for customer feedback sentiment prediction. **Int. J. Adv. Comput. Sci. Appl.**, v. 11, n. 9, 2020.

CAI, G. *et al.* Genetic dissection of plant architecture and yield-related traits in Brassica napus. v. 6, n. 1, p. 21625, 2016.

CALINSKI, T.; HARABASZ, J. A dendrite method for cluster analysis. **Commun. Stat. Theory Methods**, Informa UK Limited, v. 3, n. 1, p. 1–27, 1974.

CHAVARRÍA-PEREZ, L. M. *et al.* Improving yield and fruit quality traits in sweet passion fruit: Evidence for genotype by environment interaction and selection of promising genotypes. **PLoS One**, v. 15, n. 5, p. e0232818, maio 2020.

CORTES, C.; VAPNIK, V. Support-vector networks. **Machine Learning**, v. 20, n. 3, p. 273–297, 1995.

COVER, T.; HART, P. Nearest neighbor pattern classification. **IEEE Trans. Inf. Theory**, v. 13, n. 1, p. 21–27, 1967.

DAS, S. *et al.* Applications of Artificial Intelligence in Machine Learning: Review and Prospect. v. 115, n. 9, p. 31–41, 2015.

DEZA, M.; DEZA, E. **Encyclopedia of Distances**. 2. ed. Berlin, Germany: Springer, 2012.

DIJK, A. D. J. van. Machine learning in plant science and plant breeding. v. 24, n. 1, p. 101890, 2021.

DING, S. *et al.* Evolutionary artificial neural networks: A review. v. 39, n. 3, p. 251–260, 2013.

DUBEY, A. *et al.* Growing more with less: Breeding and developing drought resilient soybean to improve food security. v. 105, p. 425–437, 2019.

FALEIRO, F. G.; ANDRADE SOLANGE ROCHA MONTEIRO, p. u.; Fábio Bueno dos Reis Junior. **Biotecnologia: Estado Da Arte e Aplicações Na Agropecuária**. 1. ed. [*S.l.: s.n.*]: Editora Embrapa, 2011. ISBN 978-85-7075-059-4.

FAWCETT, T. An introduction to roc analysis. **Pattern Recognition Letters**, v. 27, n. 8, p. 861–874, 2006.

FREUND, Y.; SCHAPIRE, R. E. A decision-theoretic generalization of on-line learning and an application to boosting. **Journal of Computer and System Sciences**, v. 55, n. 1, p. 119–139, 1997.

FRIEDMAN, J.; HASTIE, T.; TIBSHIRANI, R. The elements of statistical learning. **Springer series in statistics**, v. 1, n. 10, p. 337–387, 2001.

FRIEDMAN, J. H. Greedy function approximation: A gradient boosting machine. **Annals of Statistics**, v. 29, n. 5, p. 1189–1232, 2001.

GOLDSCHMIDT, R. **Data Mining**. second. [*S.l.: s.n.*]: Grupo GEN„ 2015. (E-Book).

GOMES, F. R. *et al.* Evaluation of production and fruit quality of a yellow passion fruit cultivar infected with the cowpea aphid-borne mosaic virus. **Rev. Bras. Frutic.**, FapUNIFESP (SciELO), v. 44, n. 3, 2022.

GONÇALVES, D. H. *et al.* Prospecting on passiflora backcross families: implications for breeding aiming at CABMV resistance. **Euphytica**, Springer Science and Business Media LLC, v. 217, n. 4, abr. 2021.

GONçALVES, D. H. *et al.* Prospecting on passiflora backcross families: implications for breeding aiming at cabmv resistance. **Euphytica**, v. 217, n. 4, p. 1–12, 2021.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [*S.l.: s.n.*]: MIT Press, 2016.

HAND, D. J. Classifier technology and the illusion of progress. **Statistical Science**, v. 21, n. 1, p. 1–14, 2006.

HANLEY, J. A.; MCNEIL, B. J. The meaning and use of the area under a receiver operating characteristic (roc) curve. **Radiology**, v. 143, n. 1, p. 29–36, 1982.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. **The Elements of Statistical Learning: Data Mining, Inference, and Prediction**. [*S.l.: s.n.*]: Springer, 2009.

HAYKIN, S. **Neural Networks: A Comprehensive Foundation**. [*S.l.: s.n.*]: Prentice Hall, 1999.

HESAMI, M. *et al.* Application of Adaptive Neuro-Fuzzy Inference System-Non-dominated Sorting Genetic Algorithm-II (ANFIS-NSGAII) for Modeling and Optimizing Somatic Embryogenesis of Chrysanthemum. v. 10, p. 869, 2019.

HESAMI, M. *et al.* Development of support vector machine-based model and comparative analysis with artificial neural network for modeling the plant tissue culture procedures: Effect of plant growth regulators on somatic embryogenesis of chrysanthemum, as a case study. v. 16, n. 1, p. 112, 2020.

HORVITZ, E. J.; BREESE, J. S.; HENRION, M. Decision theory in expert systems and artificial intelligence. v. 2, n. 3, p. 247–302, 1988.

HUBERT, L.; ARABIE, P. Comparing partitions. **Journal of Classification**, v. 2, n. 1, p. 193–218, 1985.

IKOTUN, A. M. *et al.* K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data. **Inf. Sci. (Ny)**, Elsevier BV, v. 622, p. 178–210, 2023.

INOCENTE, G.; GARBUGLIO, D. D.; RUAS, P. M. Multilayer perceptron applied to genotypes classification in diallel studies. v. 79, n. 3, p. e20200365, 2022.

JACKSON, P. C. **Introduction to Artificial Intelligence: Third Edition**. [*S.l.: s.n.*]: Courier Dover Publications, 2019. ISBN 978-0-486-84307-0.

JAMES, G. *et al.* **An Introduction to Statistical Learning**. [*S.l.: s.n.*]: Springer, 2013. v. 112.

JHA, K. *et al.* A comprehensive review on automation in agriculture using artificial intelligence. v. 2, p. 1–12, 2019.

JIANG, T.; GRADUS, J. L.; ROSELLINI, A. J. Supervised Machine Learning: A Brief Primer. v. 51, n. 5, p. 675–687, 2020.

JURAFSKY, D.; MARTIN, J. H. **Speech and Language Processing**. Third edition draft. [*S.l.: s.n.*]: Stanford University, 2023.

KAUFMAN, L.; ROUSSEEUW, P. J. **Finding groups in data: An introduction to cluster analysis**. [*S.l.: s.n.*]: John Wiley & Sons, 2009.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. **arXiv preprint arXiv:1412.6980**, 2014.

KUMARSAGAR, H.; SHARMA, V. Error evaluation on K- means and hierarchical clustering with effect of distance functions for iris dataset. **Int. J. Comput. Appl.**, Foundation of Computer Science, v. 86, n. 16, p. 1–5, 2014.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **Nature**, v. 521, n. 7553, p. 436–444, 2015.

LEE, C. T.; PAN, L.-Y.; HSIEH, S. H. Artificial intelligent chatbots as brand promoters: A two-stage structural equation modeling-artificial neural network approach. v. 32, n. 4, p. 1329–1356, 2022.

LIAKOS, K. *et al.* Machine Learning in Agriculture: A Review. v. 18, n. 8, p. 2674, 2018.

LIAW, A.; WIENER, M. Classification and regression by randomforest. **R News**, v. 2, n. 3, p. 18–22, 2002.

LYTVYN, V. *et al.* Design of a recommendation system based on collaborative filtering and machine learning considering personal needs of the user. v. 4, p. 6–28, 2019.

MAKRIDAKIS, S. The forthcoming Artificial Intelligence (AI) revolution: Its impact on society and firms. v. 90, p. 46–60, 2017.

MEGETO, G. A. S. *et al.* Artificial intelligence applications in the agriculture 4.0. v. 51, n. 5, p. 1–8, 2020.

MITCHELL, T. M. **Machine Learning**. [*S.l.: s.n.*]: McGraw-Hill, 1997.

MOUJAHID, A. *et al.* Machine Learning Techniques in ADAS: A Review. *In*: **2018 International Conference on Advances in Computing and Communication Engineering (ICACCE)**. [*S.l.: s.n.*]: IEEE, 2018. p. 235–242.

MURPHY, K. P. **Machine learning: a probabilistic perspective**. [*S.l.: s.n.*]: MIT press, 2012.

NAINGGOLAN, R. *et al.* Improved the performance of the k-means cluster using the sum of squared error (SSE) optimized by using the elbow method. **J. Phys. Conf. Ser.**, v. 1361, n. 1, p. 012015, 2019.

NASTESKI, V. An overview of the supervised machine learning methods. v. 4, p. 51–62, 2017.

NAYYAR, A.; GADHAVI, L.; ZAMAN, N. Machine learning in healthcare: Review, opportunities and challenges. *In*: **Machine Learning and the Internet of Medical Things in Healthcare**. [*S.l.: s.n.*]: Elsevier, 2021. p. 23–45.

NIAZIAN, M.; NIEDBAŁA, G. Machine Learning for Plant Breeding and Biotechnology. v. 10, n. 10, p. 436, 2020.

OTT, R. L.; LONGNECKER, M. T. **An Introduction to Statistical Methods and Data Analysis**. 7. ed. [*S.l.: s.n.*]: Cengage Learning, 2015.

PAL, S.; MITRA, S. Multilayer perceptron, fuzzy sets, and classification. v. 3, n. 5, p. 683–697, 1992.

PEDREGOSA, F. *et al.* Scikit-learn: Machine learning in Python. **Journal of Machine Learning Research**, v. 12, p. 2825–2830, 2011.

QI, G.-J.; LUO, J. Small Data Challenges in Big Data Era: A Survey of Recent Progress on Unsupervised and Semi-Supervised Methods. v. 44, n. 4, p. 2168–2187, 2022.

RESENDE MARCOS DEON VILELA, p. u. **Matemática e Estatística Na Análise de Experimentos e No Melhoramento Genético**. 1. ed. [*S.l.: s.n.*]: Editora UFV, 2007.

RIBEIRO, L. F.; AMORIM, E. P. Adaptive thresholding for disease resistance phenotyping. **Plant Methods**, v. 19, n. 1, p. 1–15, 2023.

ROUSSEEUW, P. J. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. **J. Comput. Appl. Math.**, Elsevier BV, v. 20, p. 53–65, 1987.

SANDHU, K. *et al.* Deep learning for predicting complex traits in spring wheat breeding program. **Frontiers in Plant Science**, v. 11, 2021.

SANDHU, K. S. *et al.* Deep Learning for Predicting Complex Traits in Spring Wheat Breeding Program. v. 11, 2021.

SAPUTRA, D. M.; SAPUTRA, D.; OSWARI, L. D. Effect of distance metrics in determining k-value in k-means clustering using elbow and silhouette method. *In*: **Proceedings of the Sriwijaya International Conference on Information Technology and Its Applications (SICONIAN 2019)**. Paris, France: Atlantis Press, 2020.

SARKAR, R. K. *et al.* Evaluation of random forest regression for prediction of breeding value from genomewide SNPs. v. 94, n. 2, p. 187–192, 2015.

SCHÖLKOPF, B.; SMOLA, A. J. **Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond**. [*S.l.: s.n.*]: MIT Press, 2002.

SLAVOVA, V. *et al.* A comparative evaluation of machine learning algorithms for potato variety discrimination. **European Food Research and Technology**, v. 248, p. 1765–1775, 2022.

SLAVOVA, V. *et al.* A comparative evaluation of bayes, functions, trees, meta, rules and lazy machine learning algorithms for the discrimination of different breeding lines and varieties of potato based on spectroscopic data. **Eur. Food Res. Technol.**, v. 248, n. 7, p. 1765–1775, 2022.

SOOD, A.; SHARMA, R. K.; BHARDWAJ, A. K. Artificial intelligence research in agriculture: A review. v. 46, n. 6, p. 1054–1075, 2022.

SU, Q. *et al.* Prediction of the aquatic toxicity of aromatic compounds to tetrahymena pyriformis through support vector regression. v. 8, n. 30, p. 49359–49369, 2017.

SUN, J.; ZHAO, H. The application of sparse estimation of covariance matrix to quadratic discriminant analysis. **BMC Bioinformatics**, v. 16, n. 1, p. 48, 2015.

U, S. B. M. Performance evaluation of some clustering algorithms and validity indices. **IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE**, v. 24, n. 12, p. 1650–1654, 2002.

VAPNIK, V. **Statistical Learning Theory**. [*S.l.: s.n.*]: Wiley-Interscience, 1998.

VIDAL, R. F. *et al.* Research article evaluation of resistance to *Cowpea aphid-borne mosaic virus* in passion fruit backcrosses for recurrent selection and development of resistant cultivars. **Genet. Mol. Res.**, Genetics and Molecular Research, v. 20, n. 1, 2021.

WANG, Q.; MOGHIMI, A.; ZHANG, C. Morphometric predictors of biotic stress resistance in perennial crops. **Frontiers in Plant Science**, v. 14, p. 1120456, 2023.

WEI, M. C. F. *et al.* Carrot Yield Mapping: A Precision Agriculture Approach Based on Machine Learning. v. 1, n. 2, p. 229–241, 2020.

XIONG, Q. *et al.* Response to Nitrogen Deficiency and Compensation on Physiological Characteristics, Yield Formation, and Nitrogen Utilization of Rice. v. 9, p. 1075, 2018.

ZHANG, H. The optimality of naive bayes. **American Association for Artificial Intelligence**, v. 1, n. 2, p. 3, 2004.

ZHANG, H. Theoretical analysis of Naive Bayes. **Proceedings of the ICML**, 2004.

ZHANG, Z. Introduction to machine learning: k-nearest neighbors. **Annals of Translational Medicine: Home**, v. 4, n. 11, p. 218, 2016.

ZHANG, Z. Introduction to machine learning: k-nearest neighbors. **Annals of Translational Medicine**, v. 4, n. 11, 2016.

ZHU, J. *et al.* Multi-class adaboost. **Statistics and Its Interface**, v. 2, n. 3, p. 349–360, 2009.