INSTITUTO FEDERAL DE EDUCAÇÃO CIÊNCIA E TECNOLOGIA DE MINAS GERAIS - *CAMPUS* IBIRITÉ
ENGENHARIA DE CONTROLE E AUTOMAÇÃO

André Nery Cruz Leite

USO DE ENSEMBLE LEARNING PARA CONTROLE DE QUALIDADE DE HORTALIÇAS

Ibirité - MG
2024

ANDRÉ NERY CRUZ LEITE

USO DE ENSEMBLE LEARNING PARA CONTROLE DE QUALIDADE DE HORTALIÇAS

Trabalho de conclusão de curso apresentado ao Curso de Engenharia de Controle e Automação do Instituto Federal de Educação Ciência e Tecnologia de Minas Gerais - *Campus* Ibirité para a obtenção do título de Engenheiro de Controle e Automação.

Orientador: Prof. Carlos Dias da Silva Junior

Ibirité - MG
2024

L533u

Leite, André Nery Cruz.
Uso de ensemble learning para controle de qualidade de hortaliças. [recurso eletrônico] / André Nery Cruz Leite. – Ibirité, MG, 2024.
53 p. : il. color.

Orientador: Prof. Carlos Dias da Silva Junior.
Trabalho de Conclusão de Curso (Graduação) – Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais, *Campus* Ibirité, Bacharelado em Engenharia de Controle e Automação, 2024.

1. Aprendizado do computador 2. Agricultura. 3. Fitopatologia. I. Silva Junior, Carlos Dias da. II. Instituto Federal de Minas Gerais. *Campus* Ibirité. III. Título.

CDD 629.895

Catalogação: Viviane Barbosa Andrade - Bibliotecária - CRB-6/2819

André Nery Cruz Leite

USO DE ENSEMBLE LEARNING PARA CONTROLE DE QUALIDADE DE HORTALIÇAS

Trabalho de conclusão de curso apresentado ao Curso de Engenharia de Controle e Automação do Instituto Federal de Educação Ciência e Tecnologia de Minas Gerais - *Campus* Ibirité para a obtenção do título de Engenheiro de Controle e Automação.

Aprovado em: 24/ 09/ 2024 pela banca examinadora:

gov.br Documento assinado digitalmente
CARLOS DIAS DA SILVA JUNIOR
Data: 27/11/2024 08:43:36-0300
Verifique em https://validar.iti.gov.br

Prof. Carlos Dias da Silva Junior - IFMG (Orientador)

gov.br Documento assinado digitalmente
THIAGO HENRIQUE BARBOSA DE CARVALHO TA
Data: 27/11/2024 07:48:43-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Thiago Henrique Barbosa de Carvalho Tavares - IFMG

gov.br Documento assinado digitalmente
IVAN REINALDO MENEGHINI
Data: 22/11/2024 15:22:53-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Ivan Reinaldo Meneghini - IFMG

Dedico este trabalho à minha família, pelo apoio incondicional, e aos amigos que estiveram ao meu lado, me incentivando a nunca desistir dos meus objetivos.

AGRADECIMENTOS

Gostaria de expressar minha profunda gratidão a todos que, de alguma forma, contribuíram para a realização deste trabalho. Agradeço primeiramente à minha família, por todo o apoio, paciência e incentivo ao longo desta jornada acadêmica. Aos meus amigos, pela companhia e motivação nos momentos mais desafiadores.

Aos meus orientadores e professores, deixo minha sincera gratidão pelos ensinamentos, pela orientação e pelo conhecimento transmitido, que foram essenciais para o desenvolvimento deste projeto. Agradeço também à instituição de ensino por fornecer o ambiente e os recursos necessários para minha formação.

“Nossas virtudes e nossos defeitos são inseparáveis, assim como a força e a matéria. Quando se separam, o homem deixa de existir."

Nikola Tesla

RESUMO

O desenvolvimento da tecnologia para a agricultura tem se mostrado essencial para aumentar a eficiência na detecção de doenças em plantas e, consequentemente, melhorar a produção agrícola. Nesse contexto, este trabalho apresenta uma proposta voltada para a implementação de um sistema de classificação de doenças em plantas, utilizando técnicas de aprendizado de máquina. O objetivo principal foi construir um modelo capaz de identificar doenças específicas em cultivos de batata, milho e tomate, auxiliando os agricultores.

Foram implementados e testados diferentes modelos de classificação, como Decision Tree, Random Forest, Support Vector Classification (SVC), K-Nearest Neighbors (KNN), Gradient Boosting e Stacking Learning, a fim de selecionar o mais preciso para essa aplicação. O sistema foi desenvolvido em Python, utilizando a biblioteca Scikit-Learn, e os modelos foram avaliados com base em métricas como Erro Médio Absoluto (MAE), Erro Quadrático Médio (MSE), Raiz do Erro Quadrático Médio (RMSE) e acurácia.

Os resultados indicaram que o modelo de Stacking Learning apresentou o melhor desempenho na classificação das doenças, destacando-se pela precisão superior. Adicionalmente, foi criada uma interface em React Native, facilitando o uso do sistema em dispositivos móveis.

A partir da conclusão deste trabalho, foi possível propor uma ferramenta para o diagnóstico automatizado de doenças em plantas, oferecendo um apoio relevante para a agricultura e o manejo mais eficiente das culturas.

Palavras-chave: Ensemble Learning. Machine Learning. Classificação.

ABSTRACT

The development of technology for agriculture has proven essential for increasing efficiency in plant disease detection and, consequently, improving agricultural production. In this context, this work presents a proposal focused on implementing a plant disease classification system using machine learning techniques. The main objective was to build a model capable of identifying specific diseases in potato, corn, and tomato crops, assisting farmers in managing their crops.

Different classification models, such as Decision Tree, Random Forest, SVC, KNN, Gradient Boosting, and Stacking Learning, were implemented and tested to select the most accurate model for this application. The system was developed in Python, using the Scikit-Learn library, and the models were evaluated based on metrics like Mean Absolut Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and accuracy.

The results indicated that the Stacking Learning model achieved the best performance in disease classification, standing out for its superior accuracy. Additionally, a React Native interface was created to facilitate the system's use on mobile devices.

Upon the conclusion of this work, an automated plant disease diagnosis tool was proposed, offering valuable support for agriculture and more efficient crop management.

Keywords: Ensemble Learning. Machine Learning. Classification.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

GCP	Google Cloud Platform
API	Application Programming Interface
IA	Inteligência Artificial
AM	Aprendizado de Máquina
KNN	K-Nearest Neighbors
SVC	Support Vector Classifier
SVM	Máquinas de Vetores de Suporte
RF	Random Forest
EL	Ensemble Learning
VANT	Veículos automotores não tripulados
NB	Naive Bayes
GCS	Google Cloud Storage
ToMV	Vírus do Mosaico do Tomateiro
ToBRFV	Vírus do Rugoso Castanho do Tomate
MAE	Mean Absolut Error
MSE	Mean Squared Error

SUMÁRIO

1 INTRODUÇÃO

Nas primeiras civilizações, as pessoas utilizavam de uma dieta simples, que incluía a caça de animais e coleta de frutas, grãos e raízes por onde passavam, em um estilo de vida nômade. Com o passar do tempo, o ser humano se tornou capaz de domesticar plantas, desenvolvendo a agricultura, produzindo mais alimentos e possibilitando o crescimento populacional (SILVA; FLORêNCIO; SANTOS, 2024).

As hortaliças são plantas que possuem um ciclo de vida curto em relação à plantas mais resistentes. Seu consumo elevado desde os primórdios, levou a uma intensa comercialização pelo homem do campo, aumentando sua diversificação, podendo até serem produzidas em baixa escala, em culturas familiares (GOMES, 2023).

Um dos grandes problemas mundiais, é o desperdício de alimentos. Estudos apontam que mais de 1.3 bilhão de toneladas de alimentos são perdidos no mundo, sendo que no Brasil, aproximadamente 45% da produção de hortaliças é perdida (SILVA; FLORêNCIO; SANTOS, 2024).

A agricultura é um setor fundamental para a segurança alimentar e o desenvolvimento econômico de diversas nações. No entanto, as doenças em plantas representam um grande desafio para a produtividade agrícola, causando perdas significativas na colheita e na qualidade dos produtos, logo, o diagnóstico precoce e preciso de doenças é crucial para o controle eficaz e a minimização de impactos negativos na colheita.

O aparecimento e desenvolvimento de doenças afeta não só a quantidade da colheita como também a qualidade dos produtos colhidos, gerando um deficit no lucro final (BIONDO, 2021). A maior causa dessas doenças são micro-organismos, doenças genéticas e agentes infecciosos como bactérias, fungos e vírus (BIONDO, 2021).

A utilização inadequada de defensivos agrícolas, tais como venenos e remédios, podem agravar a proliferação das doenças, visto que aspersões recorrentes podem aumentar a resistência destes agentes maléficos à plantação, causando um descontrole na contenção da doença, gerando mais trabalho e custo (CALDEIRA, 2020).

Para realizar a análise da saúde das folhas e poder identificar qual a doença pode estar danificando a colheita, é necessário realizar um processo muito demorado e dispendioso, no qual é necessário colher uma amostragem da plantação e passá-la por uma análise criteriosa em laboratório, sendo que o tempo pode variar de acordo com a doença presente naquela plantação (CARDOSO, 2022). Este é um processo que também necessita de uma mão de obra qualificada, que nem sempre é um fator disponível, aumentando mais o custo de produção. Portanto, torna-se necessária a busca de formas mais eficientes e mais baratas de realizar esta curadoria de hortaliças.

Com o avanço da tecnologia, o uso dos computadores para a realização de tarefas repetitivas e maçantes. Na 4ª Revolução Industrial, ocorrida nos séculos XX e XXI, houve um grande avanço na tecnologia, levando à uma sociedade completamente interligada e digital. O uso de

elementos virtuais mais velozes e adaptáveis as necessidades dos usuários, deixam a realização de tarefas morosas, simples (FONTANA, 2021).

Com a evolução tecnológica, fazer as máquinas trabalharem pelos seres humanos se tornou uma realidade. Fazendo-se uso de linguagens de programação, como o Python, tornou-se possível fazer o computador entender comandos passados pelos usuários. Tarefas simples se tornaram banais e tarefas mais complexas simplificaram drasticamente.

A evolução tecnológica não só facilitou a automação de tarefas manuais, onde não é necessária uma análise para realização, como também favoreceu a automação de tarefas racionais, onde a máquina raciocina para a realização de uma ação (LUDERMIR, 2021).

Por meio de exemplos, a máquina é capaz de aprender a realizar uma ação com base nesse aprendizado, um processo conhecido como Aprendizado de Máquina (LUDERMIR, 2021). A partir da análise desses exemplos, a máquina consegue identificar padrões e, por exemplo, classificar dados com características semelhantes, como imagens ou informações de usuários.

Assim como no Aprendizado de Máquina, onde a máquina aprende a realizar ações com base em exemplos, outra abordagem importante é o *Ensemble Learning*, que visa melhorar a precisão das classificações ao combinar múltiplos modelos preditivos e, ao agregar seus resultados, consegue reduzir a variabilidade e os erros (VERMA; MEHTA, 2017).

A classificação de imagens pela máquina é realizada através de um processamento destas imagens, onde através de diversos exemplos ela consegue reconhecer elementos comuns entre elas, como objetos, paisagens e padrões (ROLA, 2022). A partir disso, é possível a criação de um programa que analise uma hortaliça e consiga identificar, de forma mais precisa que o olho humano, se a mesma está doente ou saudável.

Dado este cenário, torna-se necessária a utilização de meios tecnológicos no setor agrícola. É uma grande ajuda aos agricultores para não obterem um déficit de lucro ao final da colheita, visto que o Aprendizado de Máquina poderia diminuir o tempo de espera de um diagnóstico da plantação, reduzir a subjetividade deste diagnóstico e, consequentemente, aumentar o lucro.

1.1 Objetivos

Nesta seção, serão explicados os objetivos deste projeto, e o que pretende-se entender e buscar.

1.1.1 *Objetivo geral*

O objetivo geral deste projeto é desenvolver um sistema de controle de qualidade de hortaliças utilizando técnicas de *Ensemble Learning*, através de um aprendizado supervisionado. Este sistema visa automatizar o diagnóstico de doenças em hortaliças, proporcionando uma análise precisa e rápida das condições das plantas. A implementação deste sistema busca otimizar

o processo de identificação de patologias, reduzir o tempo e os custos associados aos métodos tradicionais de inspeção laboratorial, e, assim, melhorar a produtividade e a qualidade dos produtos agrícolas, contribuindo para a sustentabilidade e a eficiência do setor agrícola.

Isto será reunido em um aplicativo onde o usuário poderá tirar uma foto da hortaliça e se informará se está saudável ou possui alguma doença.

1.1.2 *Objetivos específicos*

I. Desenvolver um sistema de classificação de doenças em plantas utilizando técnicas de aprendizado de máquina para culturas de batata, milho e tomate;

II. Implementar diferentes técnicas de *Ensemble Learning*;

III. Implementar e comparar diferentes algoritmos de classificação, como Decision Tree, Random Forest, SVC, KNN, Gradient Boosting e Stacking Learning, para identificar qual apresenta o melhor desempenho na detecção de doenças;

IV. Avaliar o desempenho dos modelos utilizando métricas como MAE, MSE e acurácia, a fim de identificar o modelo mais preciso para a classificação das doenças nas imagens de plantas;

V. Construir uma interface móvel em React Native que permita aos usuários capturar e carregar imagens de plantas para diagnóstico de doenças, facilitando a aplicação do modelo em campo;

VI. Aplicar técnicas de processamento de imagem com OpenCV para melhorar a qualidade das imagens das folhas e tentar aumentar a acurácia dos modelos preditivos;

VII. Testar o desempenho do sistema com diferentes datasets, incluindo um conjunto de imagens de menor qualidade, para verificar a qualidade dos modelos de classificação em condições adversas;

VIII. Explorar o uso de servidores em nuvem para melhorar o processamento de grandes volumes de imagens, otimizando a eficiência e escalabilidade do sistema.

1.2 Justificativa

A crescente demanda por alimentos saudáveis e de alta qualidade impulsionou a necessidade de avanços tecnológicos no setor agrícola. As hortaliças, essenciais na alimentação humana devido ao seu alto valor nutritivo, são suscetíveis a uma variedade de doenças que podem comprometer tanto a quantidade quanto a qualidade da produção. O diagnóstico precoce e preciso destas doenças é crucial para garantir a segurança alimentar e a sustentabilidade econômica dos produtores. No entanto, os métodos tradicionais de inspeção são caros, demorados e muitas

vezes inacessíveis aos pequenos agricultores, criando um cenário onde a inovação tecnológica é essencial.

O desenvolvimento de um sistema de controle de qualidade de vegetais baseado no *Ensemble Learning*, oferece uma solução promissora para estes desafios. As técnicas de *Ensemble Learning*, que combinam vários modelos de aprendizado de máquina para melhorar a precisão e a robustez das previsões, podem fornecer diagnósticos rápidos e confiáveis das condições da planta. Esta abordagem não só reduz o tempo e os custos associados ao diagnóstico manual, mas também mitiga a subjetividade do diagnóstico humano. Ao automatizar a detecção de doenças, os agricultores podem responder mais rapidamente aos problemas, preservando a saúde das suas colheitas e aumentando a produtividade.

Além dos benefícios econômicos, a adoção de um sistema automatizado de controle de qualidade tem significância para a sustentabilidade ambiental. O uso racional de defensivos agrícolas, baseado em diagnósticos precisos, pode reduzir o impacto ambiental negativo associado ao uso excessivo de agrotóxicos e herbicidas. Ao promover práticas agrícolas mais eficientes e sustentáveis, este trabalho contribui para a proteção ambiental e a saúde pública. Portanto, este projeto não só aumenta a eficiência operacional dos produtores, mas também apoia o desenvolvimento de uma agricultura mais sustentável e responsável, alinhada com os objetivos globais de desenvolvimento sustentável e segurança alimentar.

1.3 Organização do Texto

Este trabalho está estruturado em cinco capítulos:

- Capítulo 1: Introdução: Apresenta o contexto e a motivação do estudo, os objetivos da pesquisa, a relevância e a contribuição do trabalho.
- Capítulo 2: Revisão de Literatura: Apresenta um panorama das principais doenças em plantas, dos métodos tradicionais de diagnóstico, das técnicas de aprendizado de máquina e das aplicações de modelos preditivos.
- Capítulo 3: Metodologia: Descreve a metodologia utilizada para o desenvolvimento do modelo preditivo e do aplicativo, incluindo o pré-processamento de dados, o treinamento do modelo, o desenvolvimento do aplicativo e a avaliação do modelo e do aplicativo.
- Capítulo 4: Resultados e Discussão: Apresenta os resultados obtidos na avaliação do modelo preditivo e do aplicativo, discute os resultados e as implicações para a prática agrícola.
- Capítulo 5: Conclusões: Apresenta as principais conclusões do estudo, as sugestões para trabalhos futuros e as perspectivas para o uso da tecnologia de modelos preditivos e aplicativos móveis no diagnóstico de doenças em plantas.

- Referências Bibliográficas: Foram incluídas citações que buscam contribuir para este trabalho, conceituando e referenciando.

2 REVISÃO BIBLIOGRÁFICA

Este capítulo está dividido em duas partes: a primeira aborda os principais conceitos relacionados ao tema; e a segunda descreve os trabalhos correlatos já desenvolvidos na área.

A agricultura é uma das bases fundamentais para o avanço da civilização humana, sendo um dos principais responsáveis para o desenvolvimento de sociedades mais complexas e tecnológicas. Produzir alimentos pode ser um trabalho oneroso, seja de tempo e/ou dinheiro, que inclui várias variáveis para apresentar qualidade satisfatória, como qualidade do alimento e segurança alimentar. Hortaliças são extremamente vulneráveis a doenças, pragas e as condições climáticas, por isso o controle de qualidade se torna fundamental neste setor.

O uso da tecnologia na agricultura é muito importante, seja produção em massa ou agricultura familiar. Segundo Souza *et al.* (2019), a agricultura familiar revela um deficit quanto ao uso das principais tecnologias em práticas e insumos, quando comparada ao agronegócio de produção em massa. Os autores explicam que ter conhecimento sobre a tecnologia das principais agriculturas do país é necessário, pois assim a criação de políticas para mitigar a disparidade tecnológica entre agricultores, se torna mais fácil.

A concepção do controle de qualidade do cultivo dos alimentos não é acabar com todas as pragas e doenças existentes na área cultivada. Segundo Nascimento (2020), o uso de agrotóxicos e venenos visa diminuir ao máximo pragas e doenças, mas pretendendo interferir o mínimo no ecossistema, buscando um equilíbrio entre os dois.

Para Oliveira *et al.* (2006), grandes desequilíbrios ecológicos nos cultivos deve-se ao fato do uso incorreto e indiscriminado de agrotóxicos no plantio. Tal fato gera uma seleção natural nas plantações e pragas nela existentes, podendo provocar superpopulação de pragas e alta resistência de parasitas, além de danificar o solo e aquíferos ali presentes.

A quarta revolução industrial trouxe consigo uma era de interconectividade e digitalização, onde tecnologias avançadas podem ser aplicadas para resolver problemas tradicionais de maneira mais eficiente e, com o avanço da tecnologia, especialmente no campo da inteligência artificial (IA) e aprendizado de máquina, novas oportunidades surgiram para a automação de tarefas complexas na agricultura.

2.1 Conceitos Gerais

Segundo Giraffa e Khols-Santos (2023), Inteligência Artificial (IA), se define como um sistema que, através de reconhecimento de padrões, aprendizado de máquina, raciocínio e tomada de decisões, é capaz de realizar tarefas pré-definidas.

Para Sichman (2021), inteligência artificial não se trata apenas de uma tecnologia, mas sim um conceito que utiliza conjuntos de técnicas que são usadas para resolver problemas de diferentes maneiras. É apresentado o conceito de agentes inteligentes, que são sistemas computacionais que

podem agir de forma autônoma em um ambiente, de forma a cumprir seus objetivos. O autor finaliza o artigo explorando riscos que a IA pode trazer, se usada de forma errônea, reforçando que para minimizar esses riscos, é necessário desenvolver sistemas de IA que sejam transparentes, responsáveis e que possam colaborar com os humanos.

O Aprendizado de Máquina (AM) é um ramo da Inteligência Artificial, que torna possível que máquinas aprendam a partir de dados, sem a necessidade de uma pré-programação para a realização de tarefas. Analisando grandes volumes de dados, é possível identificar padrões e características dos objetos estudados, tornando o AM uma ferramenta poderosa para resolução de inúmeros problemas, fáceis e/ou complexos. De acordo com Sousa (2023), a capacidade de gerar resultados confiáveis e consistentes torna o AM uma ferramenta capaz de reduzir a necessidade de intervenção humana, sendo capazes de realizar tarefas complexas de forma autônoma e eficiente.

Aprendizado de Máquina pode ser utilizado para várias tarefas distintas, como classificação de um grupo de dados, ou também consegue realizar a predição de algum tipo de dado, como por exemplo as probabilidades de um cliente de banco se tornar inadimplente. No estudo de Teodoro e Kappel (2020) buscou-se predizer o risco de evasão escolar em instituições públicas de ensino superior no Brasil. O estudo busca alcançar uma predição mais confiável e, para isso, os autores utilizam de diversas técnicas de AM para elencar qual possuía maior acurácia, e assim nela poder confiar. Foram utilizadas as técnicas: Naive Bayes, K-Nearest Neighbors, Árvores de Decisão, Random Forest e Redes Neurais, sendo que Random Forest alcançou um maior resultado.

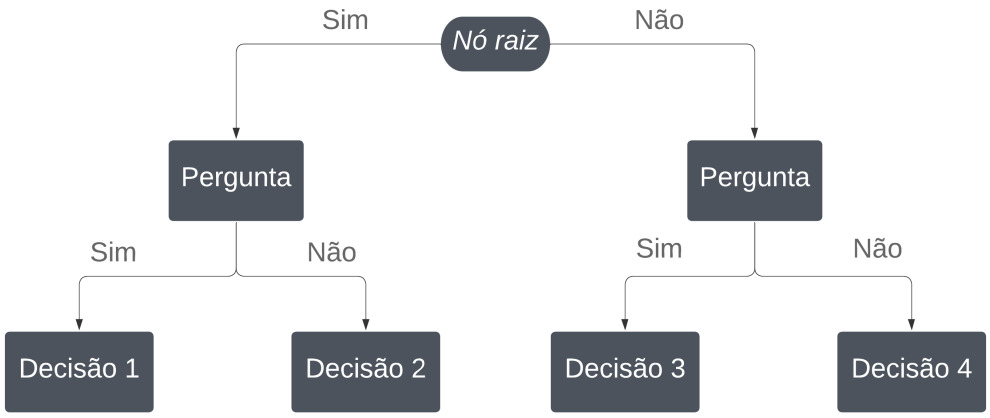
Segundo Carvalho (2014), árvore de decisão é um dos algoritmos de aprendizado supervisionado mais populares e intuitivos no campo da inteligência artificial e aprendizado de máquina. Seu funcionamento é baseado na construção de um modelo em forma de árvore, que toma decisões com base em regras condicionais, semelhantes ao processo de tomada de decisão humano.

É composta por nós e, cada nó interno da árvore representa uma pergunta ou condição sobre os dados de entrada, enquanto cada ramo representa o resultado dessa pergunta. As folhas da árvore correspondem às decisões ou classificações finais (CARVALHO, 2014), conforme Figura 1. A vantagem de uma árvore de decisão é a sua simplicidade de interpretação, pois é possível entender facilmente as regras que levam a uma determinada decisão ou classificação.

Porém, árvores de decisão possuem um risco muito grande de sobreajuste (*overfitting*), onde o modelo se torna muito específico aos dados de treinamento e perde a capacidade de generalizar para novos dados. Para evitar isso, técnicas como a poda ou o ajuste da profundidade máxima da árvore podem ser aplicadas (CARVALHO, 2014).

No geral, a árvore de decisão é uma ferramenta poderosa e versátil para tarefas de classificação e regressão, proporcionando uma abordagem simples e eficiente para a tomada de decisões com base em dados.

Figura 1 – Árvore de decisão.



Fonte: Elaborado pelo autor, 2024.

Para Carvalho (2014), o algoritmo K-Nearest Neighbors (KNN) é um dos métodos mais simples e eficazes de classificação e regressão, baseado na ideia de que amostras semelhantes devem estar próximas no espaço de características, e toma decisões baseadas nas classes ou valores das amostras mais próximas (vizinhas) da amostra a ser classificada ou estimada, atribuindo a classe que é mais frequente entre esses vizinhos no caso de classificação, ou a média dos valores no caso de regressão.

O Support Vector Classifier (SVC) é um algoritmo de aprendizado supervisionado baseado no conceito de Máquinas de Vetores de Suporte (SVM), projetado para resolver problemas de classificação. Ele funciona criando um hiperplano que separa os dados em diferentes classes, maximizando a distância entre os pontos de cada classe e o hiperplano de separação. O objetivo principal é encontrar o hiperplano que melhor separa as classes de dados. O critério para escolher o melhor hiperplano é a maximização da margem, que é a distância entre o hiperplano e os pontos mais próximos de cada classe. Esses pontos mais próximos são chamados de vetores de suporte (GHOSH; DASGUPTA; SWETAPADMA, 2019).

Random Forest (RF) é um dos modelos mais utilizados atualmente e, segundo Amaro (2023), isso se deve ao fato desta técnica conseguir trabalhar com alta dimensionalidade de dados, sendo capaz de se aprofundar em cada variável e verificar a importância de cada uma. A autora também explica que o RF é um algoritmo de aprendizado de máquina que baseia-se no aprendizado de máquina por *ensemble*, ou seja, o trabalho é feito por um grande número de árvores de decisão, onde a saída do sistema é uma média das previsões de todas estas árvores utilizadas.

Para Leite, Moraes e Lopes (2021), o modelo constrói várias árvores de decisão independentes, cada uma treinada em subconjuntos diferentes dos dados de entrada. Esses subconjuntos

são gerados através de uma técnica chamada *bootstrap*, onde uma amostra aleatória com reposição é retirada do conjunto de dados original. Além disso, ao construir as árvores, o algoritmo seleciona aleatoriamente um subconjunto de características para cada nó de decisão. Esse processo de amostragem de dados e características contribui para a diversidade das árvores no conjunto.

Durante o processo de previsão, cada árvore da floresta faz uma previsão individual, e a decisão final é tomada com base em um processo de votação (no caso de classificação) ou pela média das previsões (no caso de regressão). Esse mecanismo de votação ou média ajuda a reduzir o viés e a variância do modelo, tornando o Random Forest menos propenso ao sobreajuste (*overfitting*) em comparação com uma única árvore de decisão (LEITE; MORAES; LOPES, 2021).

Para dados de classificação, quando se utiliza o método Random Forest, o índice de Gini é utilizado para decidir como os nós em uma árvore de decisão se ramificam. Este índice é dado pela equação 2.1 (LEITE; MORAES; LOPES, 2021).

$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2, \tag{2.1}$$

Pela probabilidade, esta equação consegue determinar o Gini de cada ramo em um nó, determinando qual dos ramos é mais propenso a ocorrer. Aqui, *pi* representa a frequência relativa da classe que você está observando no conjunto de dados e c representa o número de classes.

Entropia também pode ser usada para esta função (LEITE; MORAES; LOPES, 2021).

$$Entropia = \sum_{i=1}^{C} -p_i * log_2(p_i), \tag{2.2}$$

O trabalho de Amaro (2023) traz uma análise da técnica Random Forest a partir da sua utilização para predizer a produtividade da cana-de-açúcar antes de ser feita a colheita. Para a realização do estudo foram utilizados três conjuntos de dados distintos, onde o primeiro seria um conjunto geral, enquanto nos outros dois ocorreu uma curadoria das variáveis correlacionando-as. Tal curadoria gerou uma melhora na performance do modelo, visto que reduziu dimensionalidade para o algoritmo. Foram criados três modelos distintos, onde seus parâmetros são diferentes para encontrar o modelo que mais entrega performance. A autora chegou a conclusão de que esta é uma boa técnica, mas alerta que para o resultado ser mais promissor, a calibragem dos parâmetros é necessária.

O Ensemble Learning (EL) é uma técnica de aprendizado de máquina que combina múltiplos modelos para melhorar a performance preditiva (FIGUEIREDO *et al.*, 2024). Métodos como bagging, boosting e stacking são frequentemente utilizados para aumentar a precisão e a robustez das previsões, mitigando erros que poderiam ocorrer se apenas um modelo fosse utilizado. Segundo Figueiredo *et al.* (2024), a utilização de apenas um modelo de aprendizado de

máquina pode trazer uma visão falsa do que realmente acontece (trazer apenas resultados ótimos para um conjunto de dados, analisando apenas uma amostragem, ignorando o total). Os autores também citam a técnica de Votação que se combina as previsões de todos os modelos, elegendo aquela que apresenta melhor resultado, decidindo pela maioria dos votos (Majority Voting) ou por pesos atribuídos (Soft Voting).

O estudo de Figueiredo *et al.* (2024) criou um modelo de AutoEncoder para detecção de intrusão em redes. Utilizando Ensemble Learning para melhorar o modelo, assim como clusterização, método de votação e algoritmo MINAS os resultados foram bastante satisfatórios, alcançando uma acurácia de 83%, aproximadamente.

Para Felix e Sasipraba (2019), o algoritmo Gradient Boosting é uma técnica de aprendizado de máquina amplamente utilizada para resolver problemas de classificação e regressão, criando uma sequência de modelos fracos, geralmente árvores de decisão, com o objetivo de melhorar o desempenho do modelo a cada etapa. O processo de construção acontece de forma consecutiva, onde cada novo modelo tenta corrigir os erros cometidos pelos modelos anteriores.

A grande força do Gradient Boosting vem da forma como ele minimiza o erro em cada passo. Para minimizar o erro em cada passo, o algoritmo usa uma função de perda que mede a diferença entre as previsões do modelo atual e os valores reais e, a cada nova iteração, o modelo tenta reduzir esse erro com a ajuda dos novos modelos que são adicionados (FELIX; SASIPRABA, 2019)

Stacking Learning é uma técnica de aprendizado de máquina que combina múltiplos modelos de diferentes tipos para criar um modelo final mais robusto e preciso. Ao contrário de métodos como bagging ou boosting, onde vários modelos do mesmo tipo são combinados, neste modelo utiliza-se diferentes algoritmos, como árvores de decisão, redes neurais, e máquinas de vetores de suporte (SVM), para tirar proveito dos pontos fortes de cada um, que são chamados de modelos base (WU *et al.*, 2021).

Essa abordagem permite que o modelo final se beneficie das vantagens individuais de cada algoritmo, compensando possíveis fraquezas, por exemplo, se um modelo tem boa capacidade de capturar relações complexas nos dados, enquanto outro modelo é mais robusto contra ruído, o stacking pode unir essas forças em um único modelo mais preciso..

A utilização do Aprendizado de Máquina é muito ampla, podendo estar presente em diversas áreas, desde a indústria e mercado financeiro, até a medicina e educação. Os autores Silva *et al.* (2021) desenvolveram uma ferramenta que busca melhorar a performance e acurácia de um jogo para ensino de estudantes de medicina.

Durante o processo de formação, o estudante necessita treinar e aprender antes de realizar qualquer ação na prática. Para isso, para estudantes de medicina, foram desenvolvidos jogos que buscam simular situações reais, proporcionando experiências próximas da realidade. Este é um jogo complexo, visto que demanda cautela, pois são muitas doenças e diagnósticos, grande

volume de dados para uma saída que precisa apresentar grande acurácia. Os autores Silva *et al.* (2021) utilizaram Ensenble Learning para facilitar este processo, onde construiu-se um módulo inteligente com alta confiabilidade.

Para a criação deste módulo inteligente, foram utilizados ferramentas da biblioteca *Scikit-learn*, presente na linguagem Python. Tal biblioteca possui várias ferramentas atuais e confiáveis para se utilizar na área do AM (SILVA *et al.*, 2021). Como no EL é feita a criação de variados modelos de aprendizado de máquina, é necessária a escolha do melhor resultado, portanto, foi utilizada a técnica de Validação. Foram construídos oito classificadores, e a saída do sistema foi a média destes classificadores, mitigando a chance de ocorrer um *ovefitting*. Ao final do projeto, os autores afirmam que o sistema criado é capaz de gerenciar o jogo, por conta da sua estrutura mais robusta.

Segundo Jiang *et al.* (2021), VGG16 é uma rede neural convolucional (CNN) pré-treinada, amplamente utilizado em tarefas de visão computacional devido à sua arquitetura profunda e eficiente, que consiste em 13 camadas de convolução, cinco de pooling e três camadas densas totalmente conectadas, como descrito na Figura 2. Ele foi treinado em milhões de imagens e é capaz de capturar padrões visuais complexos, tornando-o ideal para a tarefa de classificação de imagens diversas.

Camadas convolucionais são capazes de extrair vários recursos, como bordas, texturas e padrões de imagens. O pooling consegue reduzir a dimensão espacial e preserva informações importantes. Logo, combinar tais camadas permite que as CNNs aprendam sobre os dados analisados, o que é essencial para tarefas de visão computacional (PACHECO, 2019).

Há diversas formas de construir um algoritmo para aprendizado de máquina, e um deles é a Lógica Fuzzy. Se trata de uma lógica que busca tratar a incerteza e a imprecisão, permitindo que variáveis assumam valores contínuos entre 0 e 1, sendo possível determinar quando algo está parcialmente relacionada a duas categorias (MOREIRA *et al.*, 2020).

O SVM funciona encontrando o hiperplano que melhor separa os dados em diferentes classes. Em um problema de classificação binária, o objetivo do SVM é encontrar o hiperplano que maximiza a margem entre as duas classes (CARMO, 2021). O Naive Bayes classifica dados baseando-se na fórmula do Teorema de Bayes e na suposição de independência condicional. Isso significa que, dado o valor da classe, as características são independentes entre si (CARMO, 2021).

A criação de modelos de aprendizado de máquina para classificação de um determinado objeto de estudo, é atrativa para a grande maioria das pessoas, porém, sem uma interface para usabilidade pode ser bem pouco atrativa. Partindo deste ponto, o desenvolvimento de um aplicativo para dispositivos móveis revela-se uma opção vantajosa, pois permite fornecer ferramentas essenciais, juntamente com a facilidade de uma interação intuitiva, típica dos smartphones (HARAKAWA *et al.*, 2023).

No trabalho de Harakawa *et al.* (2023), os autores criaram um aplicativo, em React Native, para interagir com um blog e publicar conteúdo. React Native é um framework da linguagem Javascript que visa criar interfaces de usuário para dispositivos móveis, podendo ser disponível para Android e iOS, visto que, a estrutura usa APIs de renderização nativa em Objective-C (para iOS) ou Java (para Android). O aplicativo criado possui sete telas e utilizou o Firebase para armazenagem de dados em um banco de dados não relacional.

Uma Application Programming Interface (API) é descrita como um contrato entre partes interessadas, ou seja, se ocorrer o envio de uma requisição para a API, sua resposta determinará como o software irá agir a partir daquele momento (BASTOS, 2020). Segundo BASTOS (2020), para que uma aplicação funcione corretamente e com uma performance elevada, se torna necessário o uso de APIs externas ao código, seja para processamento, análise, coleta de dados, tudo que facilite o seu funcionamento.

Flask se trata de um framework para linguagem Python e, para Mufid *et al.* (2019), ele oferece maneiras simplificadas de construir websites, pois é mais leve e independe de bibliotecas pesadas e/ou conflituosas. Em resumo, este framework facilita a comunicação via API, entre diferentes pontos do site, como backend e frontend.

Nos tempos atuais, o uso de ferramentas de Cloud Computing tem se tornado cada vez mais prevalente no desenvolvimento de aplicações devido à necessidade de agilidade, escalabilidade e eficiência. O uso de ferramentas de cloud pode significativamente auxiliar no desenvolvimento de um aplicativo React Native, proporcionando uma série de benefícios que otimizam o processo de criação e manutenção.

Para Grosso (2021), a computação em nuvem tornou possível a comunicação eficiente na internet e o envio de recursos de um ponto a outro. Isso se deve aos três pilares fundamentais da computação em nuvem: software, plataforma e infraestrutura. No contexto da computação em nuvem, o software oferece serviços diretos ao usuário por meio de aplicações; a plataforma fornece componentes e serviços, funcionando de maneira semelhante a um hardware virtual; e a infraestrutura integra os dois primeiros pilares, proporcionando uma solução completa que combina software e plataforma.

O Google Cloud Storage (GCS) é uma solução robusta e escalável para armazenamento de dados em nuvem, oferecendo uma ampla gama de recursos e benefícios para empresas e desenvolvedores. É possível armazenar e acessar os dados de qualquer lugar do mundo, com a flexibilidade de ajustar o armazenamento de acordo com as necessidades (CHANDRA; HARTONO, 2018).

Na dissertação (GROSSO, 2021) o autor desenvolveu um sistema de regras para análise de grandes volumes de dados, utilizando computação em nuvem. Segundo o autor, a computação em nuvem foi utilizada, pois, como se trata de um grande volume de dados, é preciso tirar proveito da capacidade de processamento de tais infraestruturas. Os dados são coletados a partir de sensores

de motores e trens, e armazenados no Google Big Query, que se trata de um serviço oferecido pela Google Cloud Platform, que trabalha com SQL e é capaz de analisar grandes volumes de dados facilmente. O dado também é tratado no BigQuery. Pelo Cloud Logging, é possível verificar toda e qualquer atividade que ocorre na plataforma, como não é possível enviar *triggers* pelo BigQuery, utilizou-se o Cloud Logging para realizar este envio ao Cloud Pub/Sub, que se trata de um serviço de menssagens entre outros serviços. Com o Cloud Pub/Sub foi possível enviar um trigger para uma Cloud Function, que é um serviço que permite a criação de códigos e algoritmos em nuvem. Na Cloud Function ocorre a aplicação de regras nos dados tratados e, a partir dela, é possível armazenagem dos dados com as regras aplicadas em outra tabela do BigQuery. Segundo o autor, os resultados atingiram as expectativas iniciais. Foi capaz de desenvolver uma arquitetura para realização de análises em grandes volumes de dados.

No trabalho de Xie e Lu (2013), os autores construíram método para detectar bordas de imagem e assim determinar o número exato do núcleo de cobre no fio fino. Para a realização, utilizaram a biblioteca OpenCV, que dispõe de algoritmos para processamento de imagens e visão computacional (XIE; LU, 2013). Ao final do trabalho, os autores conseguiram alcançar o objetivo de identificar a quantidade do núcleo de cobre no fio fino.

2.2 Trabalhos relacionados

Em seu trabalho, os autores Moreira *et al.* (2020) buscaram detectar uma doença fúngica, a septoriose, de um tomateiro utilizando lógica Fuzzy. Através do Matlab (programa computacional utilizado para problemas matemáticos e científicos) construíram um algoritmo que é capaz de reconhecer padrões e, baseando nisso, aplicar uma série de regras para a realização da classificação do objeto.

As entradas do sistema são os sintomas que as plantas estudadas apresentaram. A partir da combinação destas entradas, foram estabelecidas regras que buscavam inferir a severidade da doença que a planta apresentava. A severidade pode assumir três valores, leve, moderado e grave. Utilizou-se dados fictícios para validar o sistema, que apresentou resultado satisfatório.

É comum em grandes áreas de agricultura a utilização de veículos automotores não tripulados (VANT), popularmente chamados de *drones*, tanto para obtenção de fotos das hortaliças, para monitoramento, como também para mapeamento da região. A autora Carmo (2021) utilizou técnicas de aprendizado de máquina para detecção de doenças em plantações de alface, através de processamento de imagens.

Imagens eram capturadas por um VANT, processadas e tratadas através de algoritmos. Utilizou-se modelos de classificação para definir alfaces saudáveis e infectadas. Foram utilizados os modelos Support Vector Machine (SVM) e o Naive Bayes (NB). Os modelos criados foram muito performáticos e alcançaram alta acurácia, apontando plantas na fase da doença em que ainda não é possível perceber os sintomas a olho nu. Isto ocorreu, pois uma bactéria pode alterar

as características de um vegetal, imperceptível a olho nu, mas perceptível ao processamento de imagens.

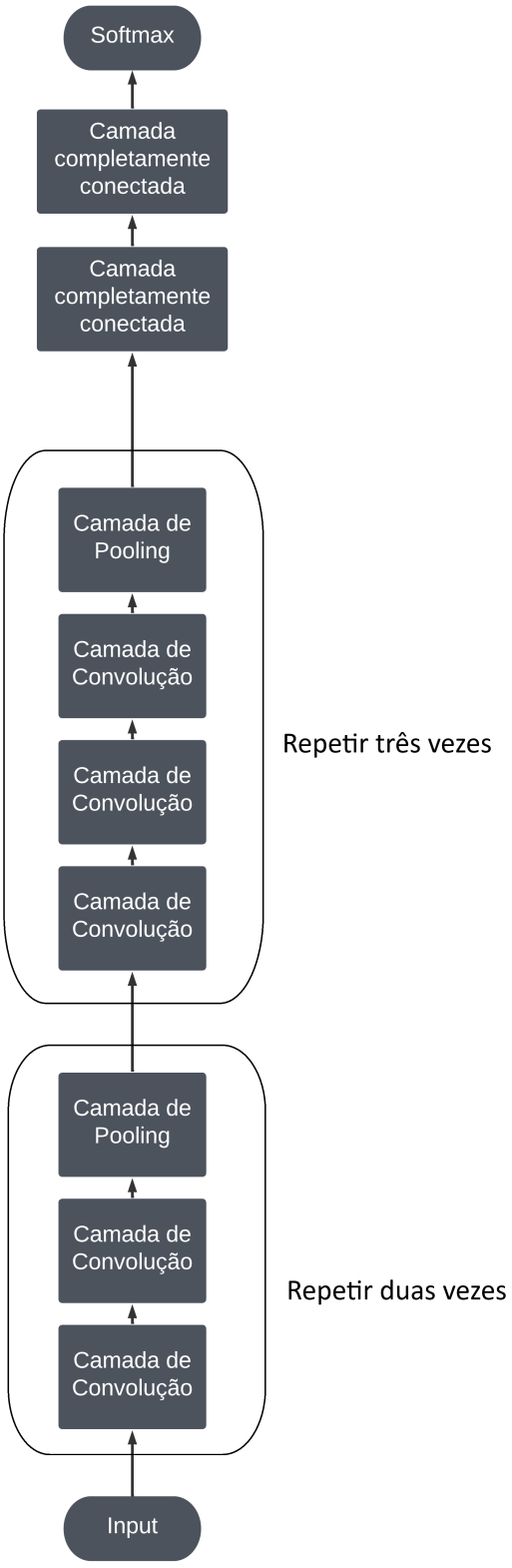
Em seu estudo, Ahmed e Reddy (2021) buscaram demonstrar o uso de redes neurais convolucionais (CNNs) para detecção de doenças em plantas, com o objetivo de otimizar a análise de imagens no campo, diretamente pelos agricultores. Foram testados três modelos de CNN: MobileNet, MNasNet e InceptionV3. Embora o InceptionV3 tenha apresentado a maior precisão, com 95,79%, também foi o modelo com maior latência em dispositivos móveis, enquanto o MobileNet, com uma precisão de 92,83%, destacou-se pelo menor tempo de resposta. A pesquisa possui o objetivo de demonstrar que o uso de modelos de CNN em dispositivos móveis para análise de doenças em plantas é possível, proporcionando uma ferramenta prática e eficiente para agricultores, com grande potencial para aumentar a precisão e rapidez na identificação de sintomas diretamente no campo. Em seu trabalho, os autores não utilizaram processamento de imagens com visão computacional, como o OpenCV.

Seguindo a mesma linha, Costa *et al.* (2024) visaram reconhecer doenças nas folhas de banana, utilizando *Deep Learning*. Construíram a base de dados a partir de imagens fotografadas por dispositivos móveis, em uma plantação de bananas real, e produziram um modelo que possui uma arquitetura que começa com um neurônio na camada de entrada, responsável por receber uma imagem de 300x300 pixels, composta por três canais RGB. Em seguida, a arquitetura inclui três camadas convolutivas sequenciais, intercaladas com camadas de pooling. Ao final, há duas camadas totalmente conectadas, com a última representando a camada de saída do modelo. O modelo alcançou uma acurácia de 86% que, segundo os autores, não é um número ótimo, porém é um início para trabalhos futuros, onde poderá ter uma base de dados mais rica para treino dos modelos.

Com o objetivo de desenvolver uma aplicação mais acessível, Silva (2023) criou um aplicativo para dispositivos móveis que permite ao usuário identificar doenças em folhas de manga de forma prática e eficiente. Utilizando o *TinyML*, um conceito que visa reduzir o tamanho de modelos treinados, possibilitando o uso local em hardwares de baixo desempenho.

O autor empregou redes neurais para a construção do modelo, utilizando a técnica de aprendizado por transferência. Foi aproveitada uma rede neural convolucional pré-treinada, e, na sua saída, foram aplicadas técnicas para mitigar o *overfitting*. Em seguida, duas camadas densas foram adicionadas, resultando na previsão final do modelo. Em seguida, o TensorFlow Lite foi utilizado para implementar o modelo em dispositivos móveis, com o aplicativo sendo desenvolvido em *Kotlin*. Um dos principais objetivos do trabalho era comparar o desempenho dos modelos executados localmente, utilizando *TinyML*, com a execução em ambientes na nuvem. Os resultados obtidos foram satisfatórios, indicando que o processamento local pode oferecer um desempenho superior em comparação ao uso de modelos baseados em armazenamento na nuvem.

Figura 2 – Estrutura VGG16.



Fonte: Elaborado pelo autor, 2024.

3 METODOLOGIA

A metodologia adotada neste trabalho é fundamental para garantir o desenvolvimento de um sistema de classificação de doenças em plantas, portanto, foi realizado um conjunto de etapas que envolvem desde a coleta e preparação dos dados até a implementação e teste dos modelos de machine learning. A seleção criteriosa dos algoritmos, o pré-processamento das imagens e o ajuste dos parâmetros dos modelos foram aspectos cruciais para atingir os resultados. A seguir, são descritas as principais etapas do processo, detalhando a base de dados utilizada e os métodos aplicados.

3.1 Base de Dados

Com o avanço das tecnologias de machine learning e inteligência artificial, a capacidade de utilizar grandes volumes de dados tornou-se essencial para obter resultados preditivos confiáveis. Logo, a análise de dados é um fator determinante no desenvolvimento de soluções tecnológicas, especialmente quando o objetivo é identificar padrões e tomar decisões baseadas em evidências. Esse cenário é particularmente relevante no setor agrícola, onde o uso de dados para prever doenças em plantas pode otimizar a produção e reduzir perdas.

Neste contexto, a coleta e organização de uma base de dados adequada para o treinamento de modelos preditivos foi um passo crucial para o desenvolvimento deste trabalho. Para alcançar os resultados desejados, a base de dados utilizada neste projeto foi obtida a partir do Kaggle, uma plataforma conhecida por fornecer conjuntos de dados variados e de alta qualidade. Essa escolha foi motivada pela diversidade e organização das informações presentes na plataforma, que permitem treinar modelos confiáveis, sejam para regressão ou classificação.

Em aplicações voltadas à detecção de doenças em plantas, a disponibilidade de um conjunto de dados representativo, contendo imagens tanto de folhas saudáveis quanto de plantas afetadas por diferentes patologias, é essencial para garantir a precisão do sistema de classificação.

Para este projeto, o conjunto de dados inclui imagens de diversas culturas, como milho, batata e tomate, abrangendo tanto folhas saudáveis quanto afetadas por diferentes doenças, com cada condição especificada de forma clara. Para a análise, foram escolhidas essas três culturas, considerando suas respectivas enfermidades.

O conjunto de dados referente à batata continha duas categorias de doenças: Requeima e Pinta Preta, além de uma pasta dedicada a plantas saudáveis. No caso do tomate, havia uma maior diversidade, com imagens distribuídas em pastas correspondentes a nove doenças, como Requeima, Pinta Preta, Mancha Bacteriana, Mancha-fuliginosa, Septoriose, Ácaro Rajado, Mancha Alvo e Enrolamento amarelo das folhas do tomateiro (TYLCD), além da pasta de plantas saudáveis. Por fim, o conjunto de dados do milho apresentava imagens divididas em três tipos de doenças: Cercosporiose, Ferrugem Comum e Helmintosporiose, juntamente com uma pasta contendo imagens de plantas saudáveis.

Essa organização clara dos dados facilita o treinamento de modelos de machine learning, permitindo que o sistema reconheça e diferencie com precisão as condições analisadas.

Figura 3 – Folha de batata com requeima.

Fonte: (ALI, 2019)

3.2 Tratamento dos dados

3.2.1 Escolha da linguagem

Para manipular os dados de forma mais facilitada, visando a agilidade e simplicidade, foi utilizada a linguagem Python, no ambiente Jupyter, da empresa Anaconda. Este software tem o intuito de facilitar a criação de algoritmos, onde, através de blocos de código, o usuário consegue testar pequenos trechos do código, ao invés de precisar executar um script inteiro.

A escolha da linguagem Python para o desenvolvimento deste trabalho foi baseada em diversas vantagens que ela oferece, especialmente em projetos de machine learning e visão computacional. Python permite maior foco na lógica da aplicação em vez de detalhes técnicos da linguagem, onde a clareza e a eficiência são essenciais, através da sua simplicidade.

Outra razão para utilizar Python neste trabalho é a sua vasta coleção de bibliotecas especializadas para aprendizado de máquina, visão computacional e manipulação de dados. Bibliotecas como TensorFlow, utilizada para implementar o modelo VGG16, e NumPy, que facilita a manipulação de arrays e dados numéricos, proporcionam ferramentas poderosas que aceleram o desenvolvimento e a experimentação. Além disso, o Keras, que serve como uma API de alto nível dentro do TensorFlow, torna o processo de definição, treinamento e validação de redes neurais mais acessível e intuitivo.

No contexto deste trabalho, Python permitiu o desenvolvimento eficiente de um pipeline de processamento de imagens, extração de características, e integração com o modelo VGG16 pré-treinado, utilizando técnicas de transferência de aprendizado. Além disso, a facilidade de manipulação de grandes volumes de dados e a interoperabilidade com plataformas de cloud computing tornaram Python a escolha ideal para alcançar os objetivos propostos, garantindo uma implementação eficaz e escalável.

3.2.2 Processamento dos dados

O processamento dos dados começou com a função *extract_features*, que redimensiona as imagens para 224x224 pixels, o tamanho esperado pela entrada do VGG16. Após o redimensionamento, as imagens são convertidas em arrays numéricos e passam por uma etapa de pré-processamento específica do modelo, que ajusta os valores de pixel para o intervalo esperado pela rede. O VGG16, sem suas camadas superiores (a parte de classificação final), é então usado para gerar um vetor de características que representa cada imagem. Esse vetor é uma abstração das informações visuais presentes na folha, condensadas em um formato que pode ser usado para a classificação.

Além disso, a função *define_targets* foi utilizada para percorrer os diretórios de imagens de cada planta (milho, batata e tomate), organizando as imagens de acordo com suas respectivas categorias (plantas saudáveis ou doentes). Cada imagem foi processada para extrair seu vetor de características utilizando o VGG16, que então foi armazenado na lista X. Ao mesmo tempo, os rótulos correspondentes (indicando a saúde ou a doença da planta) foram armazenados na lista y.

Essa abordagem, conhecida como *transfer learning*, aproveita a capacidade do VGG16 de generalizar padrões visuais, evitando a necessidade de coletar um volume imenso de dados para treinar um modelo do zero. Neste trabalho, essa técnica foi importante para garantir que as características visuais extraídas das folhas fossem suficientemente representativas, facilitando a identificação e classificação das diferentes doenças nas plantas selecionadas (milho, batata e tomate).

3.2.3 Separação entre treino e teste

Para garantir a validação adequada do modelo desenvolvido neste trabalho, foi realizada a separação dos dados em conjuntos de treino e teste. Prática essencial no aprendizado de máquina, pois permite avaliar o desempenho do modelo em dados que ele não viu durante o treinamento, evitando o risco de sobreajuste (overfitting) e proporcionando uma medida mais confiável.

A separação foi realizada utilizando a função *train_test_split* da biblioteca *scikit-learn*, que é amplamente utilizada para particionar conjuntos de dados. Nesse projeto, 80% dos dados foram destinados para o treinamento do modelo e 20% foram reservados para o teste. Esse particionamento balanceado, com a variável *random_state* fixada em 42 para garantir a reproduti-

bilidade, assegura que a avaliação do modelo seja justa e que ele seja capaz de aprender padrões úteis a partir dos dados de treino.

Além disso, antes de realizar essa separação, as classes presentes no conjunto de dados foram codificadas numericamente utilizando a função LabelEncoder. Isso transformou os rótulos categóricos das classes em valores inteiros, facilitando o processamento pelo modelo.

3.3 Aplicação dos modelos de classificação

No desenvolvimento deste trabalho, foram implementados diversos modelos de aprendizado de máquina para realização da classificação das doenças nas folhas das plantas. Foi utilizada uma abordagem variada de algoritmos com o objetivo de comparar suas performances e selecionar o mais eficiente para a tarefa. Dentre os modelos implementados, utilizou-se a árvore de decisão, Random Forest, SVC, K-Nearest Neighbors (KNN), Gradient Boosting e Stacking. Para a aplicação dos modelos, foram utilizadas as bibliotecas Python: *sklearn.tree*, *sklearn.neighbors*, *sklearn.ensemble* e *sklearn.svm*.

A implementação da árvore de decisão foi realizada utilizando o DecisionTreeClassifier, sendo ajustado aos dados de treino, permitindo classificar novas amostras com base em uma série de decisões binárias.

O modelo Random Forest foi configurado com 100 árvores, utilizando o RandomForestClassifier da biblioteca Scikit-learn. O uso de 100 estimadores e a definição de um estado aleatório garantiram a estabilidade e reprodutibilidade dos resultados, reduzindo o risco de overfitting.

Para o modelo SVC, foi utilizado um kernel linear, que é adequado para separar classes em problemas de classificação com margens bem definidas. A configuração *probability=True* permitiu a obtenção de probabilidades das predições, fornecendo uma maior flexibilidade na análise dos resultados.

O modelo K-Nearest Neighbors (KNN) foi configurado para considerar 10 vizinhos. Este algoritmo classifica uma nova amostra com base na proximidade em relação às amostras já classificadas, analisando as 10 mais próximas para determinar a classe. Embora o KNN seja simples e intuitivo, sua eficiência depende da distribuição das classes no espaço de características, sendo computacionalmente mais pesado para grandes volumes de dados.

Também foi utilizado o Gradient Boosting, que se mostrou uma técnica poderosa para otimização. Foi configurado utilizando o GradientBoostingClassifier com 100 estimadores, um método que melhora a precisão ao adicionar sequencialmente modelos fracos, ajustando-os a cada iteração para corrigir os erros do modelo anterior.

Por fim, aplicou-se a técnica de Stacking, que combina diferentes algoritmos de base, incluindo Random Forest, SVC e KNN, para melhorar a performance geral. No Stacking, cada modelo base realiza suas predições independentemente, e as saídas desses modelos são utilizadas

como entrada para um estimador final. O estimador final escolhido foi o Gradient Boosting, que combinou as predições dos modelos de base para realizar a predição final. Essa abordagem de ensemble foi essencial para aproveitar as forças de cada modelo individual, garantindo uma maior robustez e precisão no sistema de classificação, ou seja, o sistema torna-se capaz de lidar eficientemente com dados variáveis ou condições adversas sem perder desempenho.

Figura 4 – Aplicação do modelo Stacking Learning.

```python
# Definir os estimadores base
estimators = [
    ('rf', RandomForestClassifier(n_estimators=100, random_state=42)),
    ('svc', SVC(kernel='linear', probability=True)),
    ('knn', KNeighborsClassifier(n_neighbors=10))
]

# Definir o estimador final
final_estimator = GradientBoostingClassifier(n_estimators=100, random_state=42, n_iter_no_change=10)
```

```python
def stacking(X_train, X_test, y_train, y_test, plant):
    # Configurar o StackingClassifier
    clf = StackingClassifier(
        estimators=estimators,
        final_estimator=final_estimator
    )

    # Treinar o modelo
    clf.fit(X_train, y_train)

    # Fazer previsões
    y_pred = clf.predict(X_test)

    # Calcular a acurácia
    accuracy = accuracy_score(y_test, y_pred)
    print(f"Accuracy: {accuracy * 100:.2f}%")


    model_saved = joblib.dump(clf, f"clf_{plant}.joblib")

    return y_pred, model_saved
```

Fonte: Fonte: Elaborado pelo autor, 2024.

Além dos modelos mais complexos utilizados, como Random Forest, SVC e Gradient Boosting, também foi implementado o modelo Naive Bayes para fins de comparação. O Naive Bayes é um modelo mais simples e baseado em probabilidade, que assume independência entre as características de entrada. Sua simplicidade permite uma execução mais rápida e eficiente em termos de computação, mas ele pode ser limitado quando as variáveis não são realmente independentes, como frequentemente ocorre em imagens complexas. No entanto, sua inclusão foi importante para avaliar como um modelo mais básico se comportaria em relação a algoritmos mais avançados, proporcionando um ponto de referência no desenvolvimento da solução.

Com essa diversidade de modelos, foi possível realizar uma análise comparativa de suas performances e selecionar a solução mais adequada para a tarefa de classificação de doenças nas folhas das plantas, garantindo uma abordagem eficiente e precisa para o problema.

3.4 Salvando os modelos

Para garantir a agilidade e eficácia do aplicativo final, optou-se por salvar os modelos utilizando a biblioteca *joblib*, que é amplamente utilizada para serialização de objetos no Python, especialmente em projetos de machine learning. A escolha da *joblib* foi motivada pela sua eficiência ao lidar com grandes arrays NumPy, tornando-o ideal para salvar e carregar modelos complexos de forma rápida e eficaz.

Após a escolha do modelo final, utilizou-se esta biblioteca para salvar o estado do modelo em arquivos binários, que foram armazenados no Google Cloud Storage (GSC). Este último se trata de uma solução de armazenamento em nuvem que permite acesso rápido e seguro aos arquivos, através de uma API.

Essa estratégia permitiu a escalabilidade do projeto, facilitando o compartilhamento e o acesso remoto aos modelos por diferentes sistemas e plataformas. Além disso, garante que o trabalho de treinamento não precise ser repetido a cada execução, otimizando o tempo de processamento e a eficiência do sistema.

3.5 Desenvolvimento do aplicativo

Um aplicativo é normalmente dividido em duas partes principais: o frontend e o backend. Este último é a parte responsável por gerenciar a lógica de negócios, manipular dados e realizar operações no servidor, funciona como um intermediário entre o frontend (que lida com a interface do usuário) e o banco de dados, garantindo que a aplicação funcione de maneira correta e eficiente.

3.5.1 Backend

O backend da aplicação foi desenvolvido utilizando o framework Flask, escolhido por ser uma ferramenta leve e flexível para a criação de APIs em Python. É responsável por receber as imagens enviadas pelo usuário, processá-las, realizar predições utilizando modelos de aprendizado de máquina previamente treinados e retornar os resultados ao frontend.

Inicialmente, os modelos de machine learning, previamente salvos em arquivos *.joblib*, foram armazenados no Google Cloud Storage. Para carregá-los no backend, foi utilizada a biblioteca *joblib*, que permite a serialização dos modelos, e a biblioteca *google.cloud.storage* para realizar o download dos arquivos diretamente do *bucket* (contêiner usado para armazenar objetos, como arquivos e dados, no Google Cloud). Assim, ao iniciar o servidor, os modelos de classificação para milho, tomate e batata são baixados e carregados na memória, prontos para serem usados durante as predições.

Quando o usuário envia uma imagem para a API, o backend primeiro processa essa imagem utilizando o modelo VGG16, que foi pré-treinado em uma grande base de dados de imagens e tem um excelente desempenho na extração de características visuais relevantes. A

imagem recebida é convertida para o formato adequado (224x224 pixels) e passa por um pré-processamento antes de ser inserida no modelo VGG16, que gera um vetor de características (features) da imagem.

Com o vetor de características gerado, o backend seleciona o classificador apropriado de acordo com o tipo de planta informado pelo usuário (milho, tomate ou batata) e realiza a predição. O classificador, treinado especificamente para identificar doenças nas folhas de cada planta, utiliza o vetor de características da imagem processada para determinar a provável doença presente na folha.

Por fim, o backend retorna o resultado da predição em formato JSON (formato de intercâmbio de dados baseado em texto, derivado da linguagem de programação JavaScript, que utiliza uma estrutura de chave-valor, permitindo a representação de dados em forma de objetos, arrays, strings, números e valores booleanos) que é então exibido ao usuário na interface da aplicação. O fluxo de comunicação entre frontend e backend é simplificado, tornando o sistema eficiente e permitindo que novos modelos sejam facilmente integrados à medida que a aplicação evolui.

3.5.2 Servidor

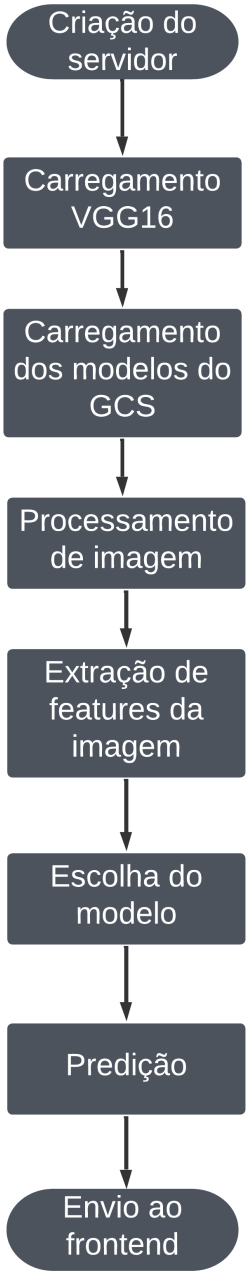
Para viabilizar o uso local do backend durante o desenvolvimento e testes da aplicação, utilizou-se o Ngrok para expor o servidor Flask de maneira acessível pela internet, que, por padrão, roda localmente no endereço localhost, o que limita o acesso apenas à máquina onde o servidor está sendo executado. Para permitir que dispositivos externos, como smartphones, pudessem enviar requisições ao servidor durante o desenvolvimento, o Ngrok foi uma solução ideal.

O Ngrok funciona criando uma conexão entre a máquina local e a internet, fornecendo um endereço de URL público que redireciona as requisições para o servidor Flask local. Esse procedimento foi essencial para testar a integração entre o frontend, rodando em dispositivos móveis, e o backend.

O processo envolveu os seguintes passos: primeiro, o servidor Flask foi iniciado localmente, escutando na porta 8080. Em seguida, o comando do Ngrok foi executado para conectar essa porta à internet, gerando uma URL pública, sendo possível configurar o frontend da aplicação React Native para enviar as imagens e dados diretamente ao servidor, permitindo a realização de predições em tempo real.

Essa abordagem facilitou o desenvolvimento, ao permitir a simulação de um ambiente de produção.

Figura 5 – Fluxo de trabalho do backend.



Fonte: Elaborado pelo autor, 2024.

3.5.3 Frontend

A criação da interface frontend da aplicação utiliza o React Native, uma biblioteca popular para o desenvolvimento de aplicativos móveis que permite a criação de interfaces para Android e iOS com uma base de código única. O objetivo principal do aplicativo é permitir que os usuários carreguem ou capturem imagens de plantas e escolham o tipo de planta que desejam avaliar. Com base na imagem e na seleção, a aplicação envia os dados para o backend para realizar a predição,

e os resultados são apresentados ao usuário de forma clara e intuitiva.

A interface é desenvolvida com foco na simplicidade e usabilidade, em que o usuário pode escolher entre duas opções para fornecer a imagem: selecionar uma foto da galeria do dispositivo ou capturar uma nova foto com a câmera. Para isso, a aplicação utiliza o pacote expo-image-picker, que solicita as permissões necessárias e permite o acesso à galeria e à câmera. Quando o usuário faz uma seleção ou captura uma imagem, o URI da foto é armazenado na aplicação para posterior processamento.

Após o carregamento da imagem, o usuário deve selecionar o tipo de planta que deseja avaliar, feita através de um componente Picker, que apresenta opções como milho, batata e tomate. A escolha do usuário é armazenada e utilizada para direcionar a imagem para o modelo de predição apropriado no backend.

O envio da imagem para o backend é realizado por meio de uma requisição HTTP POST, utilizando a biblioteca Axios, uma biblioteca JavaScript popular que simplifica a realização de requisições HTTP e o gerenciamento de respostas em aplicativos web e móveis, que permite enviar e receber dados de forma eficiente e oferece uma API para lidar com as operações de rede.

A imagem é enviada para o backend dentro de um objeto FormData, que é uma forma de representar os dados de um formulário para serem enviados via HTTP. Além da imagem, a FormData também inclui a informação sobre o tipo de planta selecionado pelo usuário. Uma vez que o backend recebe e processa esses dados, ele realiza a predição e retorna os resultados, que são então exibidos na interface do usuário, proporcionando feedback imediato sobre a análise da imagem.

Durante o processamento da imagem, um indicador de carregamento (ActivityIndicator) é mostrado para informar ao usuário que a operação está em andamento. Quando a predição é recebida, os resultados são apresentados em um componente de texto de forma clara.

O uso do React Native e das bibliotecas auxiliares permitiu a criação de uma aplicação que se comunica de maneira eficaz com o backend, garantindo a entrega de resultados precisos e úteis ao usuário final.

3.5.4 Teste da aplicação

Para testar a aplicação em um dispositivo móvel, foi utilizado o Expo, uma ferramenta e framework que facilita o desenvolvimento e a execução de aplicativos React Native. O Expo oferece um ambiente de desenvolvimento que permite compilar e rodar aplicativos em dispositivos físicos e emuladores com rapidez e eficiência.

Ao utilizar o Expo, foi possível carregar e testar a aplicação no dispositivo móvel. Ele simplifica o processo de testes ao permitir que o aplicativo seja atualizado em tempo real, diretamente no dispositivo, sem a necessidade de compilar e instalar manualmente a cada alteração

no código. Isso proporcionou um ciclo de desenvolvimento mais ágil e eficiente, garantindo que as funcionalidades fossem testadas em um ambiente real.

Foi construído um segundo dataset, de menor tamanho, com o objetivo de testar os modelos em condições mais desafiadoras. Esse novo conjunto de dados foi composto por imagens de qualidade inferior, apresentando maior quantidade de ruídos, falta de foco em uma única folha e, em alguns casos, sobreposição de partes de outras plantas. Essas características tornam a tarefa de classificação mais complexa, exigindo uma maior capacidade de generalização dos modelos.

Na tentativa de mitigar esses problemas e melhorar a acurácia, foi implementado um processo de pré-processamento de imagem utilizando a biblioteca OpenCV, com o objetivo de delimitar melhor a área da folha e, assim, fornecer imagens mais focadas e representativas para os modelos. Esta biblioteca foi utilizada, pois oferece um conjunto abrangente de ferramentas para realizar tarefas como detecção de objetos, reconhecimento facial, segmentação de imagens, filtragem e transformação geométrica, entre outras. Funciona através da manipulação direta de pixels de uma imagem, convertendo-os em uma matriz de valores que podem ser processados matematicamente.

Todo o código da aplicação, juntamente com os códigos de cada modelo, podem ser encontrados em (LEITE, 2024).

4 RESULTADOS

4.1 Separação dos datasets

A base escolhida é composta apenas por imagens e suas respectivas labels, logo, não foi necessário o estudo aprofundado do tipo de dado e sua correlação, analisando um mapa de calor, por exemplo. É evidente que as imagens serão as variáveis de entrada, e as labels serão as variáveis de saída, aquilo a ser predito.

A quantidade de imagens coletadas é relativa, ou seja, cada doença, de cada planta, tem uma quantidade diferente de imagens, por isso, o número de imagens para cada classe foi equilibrado para evitar o enviesamento. A quantidade respectiva de cada doença da batata está presente na Tabela 1. Assim como a quantidade do milho está expressa na Tabela 2, e a do tomate na Tabela 3.

Condição	Quantidade
Saudável	150
Requeima	150
Pinta Preta	150

Tabela 1 – Quantidade de imagens para a batata.

Condição	Quantidade
Saudável	500
Ferrugem Comum	500
Helmintosporiose	500
Cercosporiose	500

Tabela 2 – Quantidade de imagens para o milho.

Condição	Quantidade
Saudável	900
Requeima	900
Pinta Preta	900
Mancha Bacteriana	900
Mancha-fuliginosa	900
Septoriose	900
Ácaro Rajado	900
Mancha Alvo	900
Enrolamento amarelo das folhas do tomateiro (TYLCD)	900

Tabela 3 – Quantidade de imagens para o tomate.

Após processar as imagens e tratá-las para ficarem no formato em que os modelos foram treinados, conforme explicado na subseção 3.2.2, foi feita a separação entre teste e treino, uma separação em 80% para treino e 20% para teste, com o estado randômico de 42.

4.2 Validação dos modelos

Após a separação dos dados entre treino e teste, os modelos classificadores foram treinados e testados, utilizando os métodos presentes na biblioteca *Scikit-Learn*.

Para avaliação dos modelos, foram utilizados os métodos *Mean Absolut Error* (MAE), *Mean Squared Error* (MSE), além da acurácia que a própria biblioteca oferece.

O MAE calcula a média da diferença entre o valor predito e o valor real. É dado pela equação 4.1.

$$MAE_{(y_{real}, y_{previsto})} = \frac{1}{n}\sum_{i=1}^{n} \left| y_{real_i} - y_{previsto_i} \right| \tag{4.1}$$

O MSE calcula a diferença média entre o valore real e o valor predito. Porém, diferentemente do MAE, este consegue identificar outliers. É dado pela equação 4.2.

$$MSE_{(y_{real}, y_{previsto})} = \frac{1}{n}\sum_{i=1}^{n} \left(y_{real_i} - y_{previsto_i} \right)^2 \tag{4.2}$$

4.2.1 Avaliação dos modelos

Após o treinamento dos modelos implementados, foi possível realizar uma avaliação detalhada de seu desempenho utilizando: o erro médio absoluto (MAE), o erro quadrático médio (MSE) e a acurácia.

O MAE foi utilizado para medir a média das diferenças absolutas entre os valores reais e preditos, representando a magnitude do erro sem considerar a sua direção; o MSE, por sua vez, mede a média dos erros quadráticos, penalizando erros maiores de forma mais acentuada. Já a acurácia foi utilizada para avaliar o desempenho dos modelos em tarefas de classificação, representando a proporção de previsões corretas entre todas as previsões realizadas.

Essas métricas permitiram não apenas comparar o desempenho dos diferentes modelos, mas também identificar aqueles que apresentaram maior precisão e menor erro na previsão dos dados de teste, contribuindo para uma análise mais aprofundada dos resultados obtidos no projeto.

Os resultados para cada métrica e modelo podem ser encontrados nas Tabelas 4, 5 e 6.

De acordo com os dados retirados da utilização dos métodos de avaliação, é possível perceber que o modelo escolhido (Stacking Learning) nem sempre se manteve com uma precisão melhor que o dos outros modelos. Porém, foi utilizado, pois ele oferece uma abordagem mais robusta ao combinar o poder preditivo de vários modelos. Enquanto cada algoritmo de aprendizado de máquina, como Decision Tree, Random Forest, SVC, KNN e Gradient Boosting, tem suas próprias vantagens e limitações, o Stacking Learning permite que essas forças sejam agregadas. Ao usar múltiplos modelos base como Random Forest, SVC e KNN, e um estimador final (neste caso, Gradient Boosting), o algoritmo consegue explorar os pontos fortes de cada um.

Modelo	Batata	Milho	Tomate
Naive Bayes	0.31	0.51	1.23
Árvore de Decisão	0.23	0.39	1.34
Random Forest	0.04	0.12	0.45
SVC	0.06	0.10	0.15
KNN	0.72	0.81	2.32
Gradient Boosting	0.08	0.11	0.31
Stacking Learning	0.04	0.10	0.14

Tabela 4 – Cálculo do MAE.

Modelo	Batata	Milho	Tomate
Naive Bayes	0.38	1.05	7.86
Árvore de Decisão	0.3	0.76	5.39
Random Forest	0.04	0.24	1.53
SVC	0.06	0.21	0.47
KNN	1.21	1.66	12.99
Gradient Boosting	0.08	0.21	1.03
Stacking Learning	0.04	0.21	0.43

Tabela 5 – Cálculo do MSE.

Modelo	Batata	Milho	Tomate
Naive Bayes	72,22%	71,75%	38,70%
Árvore de Decisão	80%	78,5%	53,14%
Random Forest	95,55%	94%	79,87%
SVC	94,44%	94,75%	93,02%
KNN	52,22%	52,25%	45%
Gradient Boosting	92,22%	93,5%	85,67%
Stacking Learning	95,56%	95%	93,4%

Tabela 6 – Cálculo da Acurácia.

4.2.2 Teste dos modelos

Os resultados obtidos com o segundo dataset indicaram que os modelos não conseguiram se sair tão bem quanto no conjunto de dados original. As precisões observadas foram significativamente mais baixas, evidenciando que a presença de ruídos e a menor qualidade das imagens comprometeram o desempenho dos classificadores, mostrando sua limitação em cenários menos controlados.

Além disso, foi possível perceber um certo enviesamento dos modelos para a classificação de Requeima, especialmente em casos onde a imagem se tratava de um dado correspondente a Pinta Preta. Uma hipótese para esse comportamento é que as doenças se assemelham bastante em termos de sintomas, o que pode confundir o algoritmo, pois quando Pinta Preta já está em um estado mais avançado, como retratada na Figura 6, a semelhança com Requeima, retratada na Figura 7, aumenta consideravelmente, dificultando ainda mais a distinção entre as duas condições,

sugerindo que o modelo tem dificuldade em lidar com nuances sutis entre estágios avançados das doenças, levando a um viés na classificação.

Figura 6 – Folha do tomate diagnosticada com Pinta Preta.

Fonte: (ALI, 2019).

Figura 7 – Folha do tomate diagnosticada com Requeima.

Fonte: (ALI, 2019).

Apesar dos esforços em tentar melhorar a acurácia dos modelos com o OpenCV, os resultados obtidos com essa abordagem foram inferiores. O pré-processamento das imagens não contribuiu significativamente para o aumento da acurácia, e os modelos continuaram a apresentar dificuldades em classificar corretamente as imagens de menor qualidade e com ruídos visuais.

As imagens 8, 9, 10, mostram como o modelo se saiu na classificação do segundo dataset, sem o uso da ferramenta OpenCV para pré-processamento das imagens. Os valores estão retratados como porcentagem.

As imagens 11, 12 e 13 apresentam o comportamento do modelo após as imagens passarem pelo pré-processamento com o OpenCV, também retratado como porcentagem.

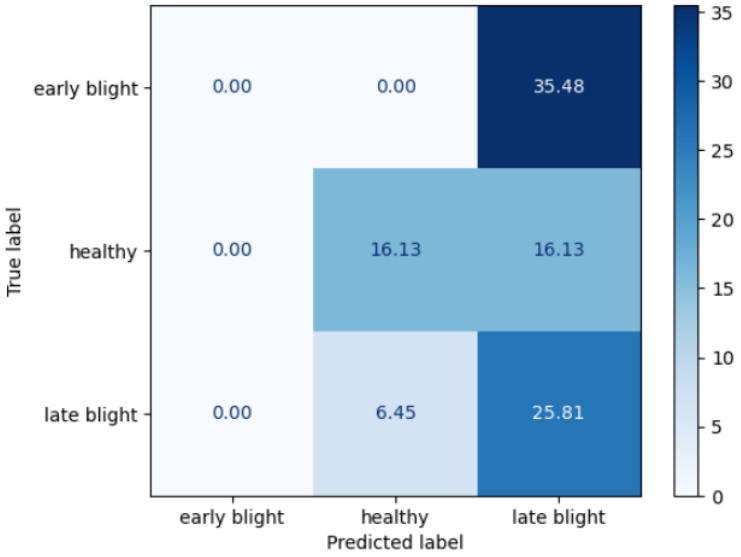
Nota-se que o modelo obteve mais acurácia sem utilizar o OpenCV para processar as imagens. A tabela 7 indica os resultados obtidos pelo modelo.

Hortaliça	Sem OpenCV	Com OpenCV
Batata	43,33%	43,33%
Milho	57,5%	40%
Tomate	14,44%	12,22%

Tabela 7 – Desempenho do modelo com o dataset secundário.

É possível notar nos gráficos obtidos nos testes, apesar dos diversos esforços aplicados, não foram tão satisfatórios quanto o esperado. Mesmo com o equilíbrio na quantidade de imagens entre as classes, a fim de evitar vieses nos modelos, e a utilização da técnica de early stopping

Figura 8 – Comportamento do modelo para batata, sem utilizar OpenCV.



Fonte: Elaborado pelo autor, 2024.

para mitigar o overfitting durante o treinamento, os modelos não atingiram a precisão desejada. Além disso, foi implementado um processo de segmentação de imagens usando OpenCV para identificar as bordas das folhas, na tentativa de melhorar o foco da análise nos elementos mais relevantes. Contudo, essas abordagens não resultaram em uma melhora significativa na acurácia dos modelos, indicando que outros fatores, como a qualidade das imagens e a complexidade dos dados, podem ter impactado o desempenho final, pois os modelos foram treinados apenas com imagens de qualidade ótima e foco total na folha, conforme Figura 3.

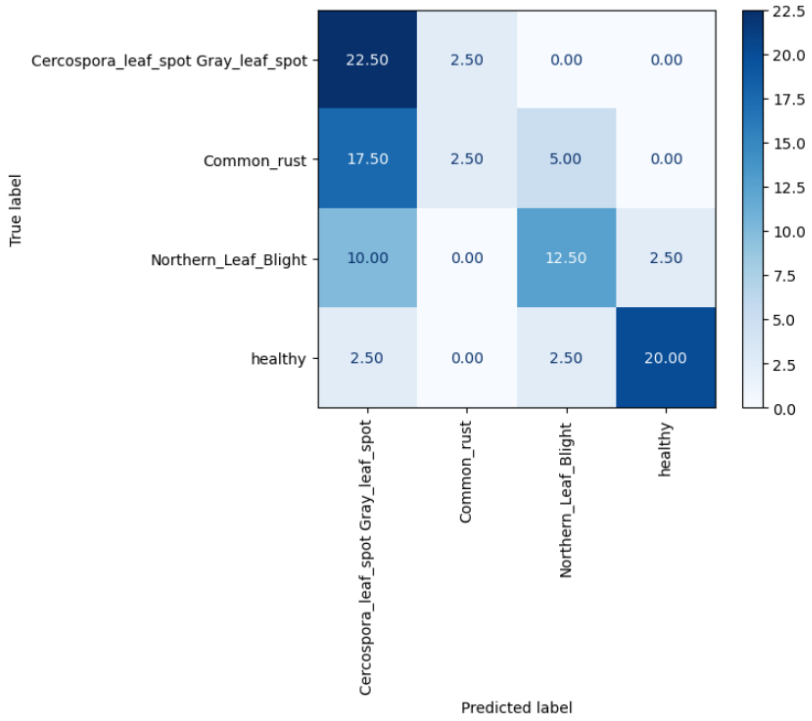
4.3 Construção da interface

A construção da interface para o aplicativo de classificação de doenças em plantas foi pensada para ser intuitiva e funcional, visando facilitar a experiência do usuário, onde o mesmo pudesse carregar uma imagem da planta que deseja analisar e, com poucos cliques, obter o diagnóstico da condição da folha.

Durante o desenvolvimento da interface, foram adotados elementos que garantissem uma navegação clara e objetiva. Ela permite que o usuário escolha entre diferentes tipos de plantas (como batata, milho e tomate), faça o upload de uma imagem ou utilize a câmera do dispositivo para capturar a foto da folha. Após isso, a imagem é processada e os resultados são apresentados de forma visual e descritiva, informando a condição de saúde da planta ou se ela apresenta sinais de alguma doença, conforme Figura 14.

Foram adotadas tecnologias que permitem a utilização da aplicação em múltiplos dispositivos. A escolha do React Native foi fundamental para simplificar esse processo, pois a biblioteca oferece ferramentas que tornam a implementação do aplicativo mais eficiente e acessível. Além

Figura 9 – Comportamento do modelo para milho, sem utilizar OpenCV.



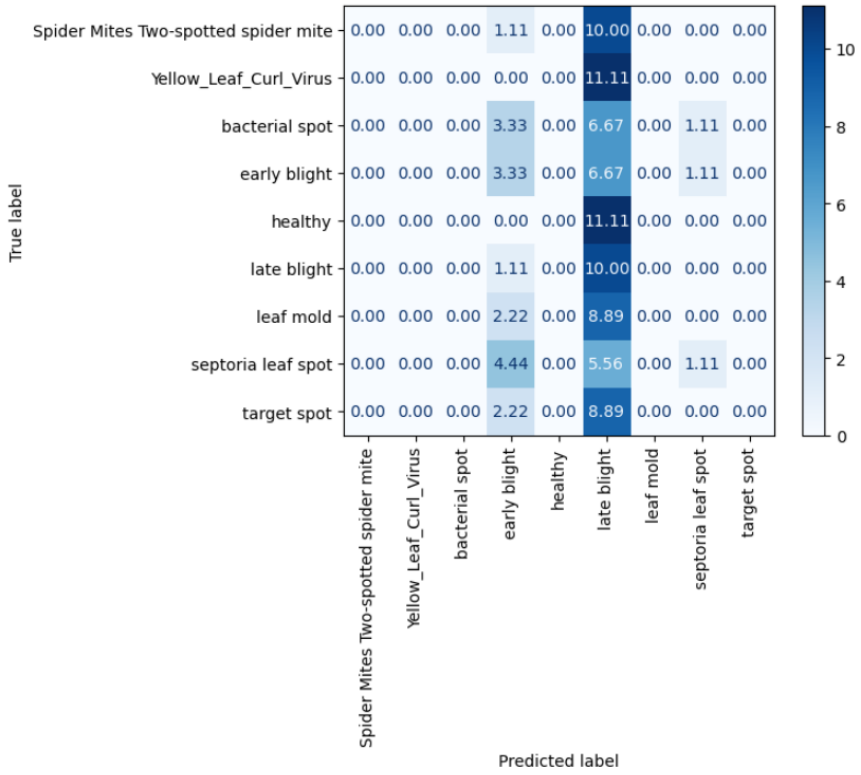
Fonte: Elaborado pelo autor, 2024.

disso, sua estrutura intuitiva facilita tanto o desenvolvimento quanto a construção da aplicação, proporcionando um ambiente de programação mais ágil e produtivo.

Durante os testes na interface, observou-se que o modelo é capaz de avaliar qualquer tipo de imagem, não se restringindo apenas a folhas. Esse comportamento abre possibilidades interessantes para futuros aprimoramentos, onde o algoritmo poderia ser ajustado para identificar primeiro se a imagem corresponde a uma folha. Apenas após essa verificação o modelo seguiria com a avaliação da doença, garantindo assim maior precisão e eficiência no diagnóstico.

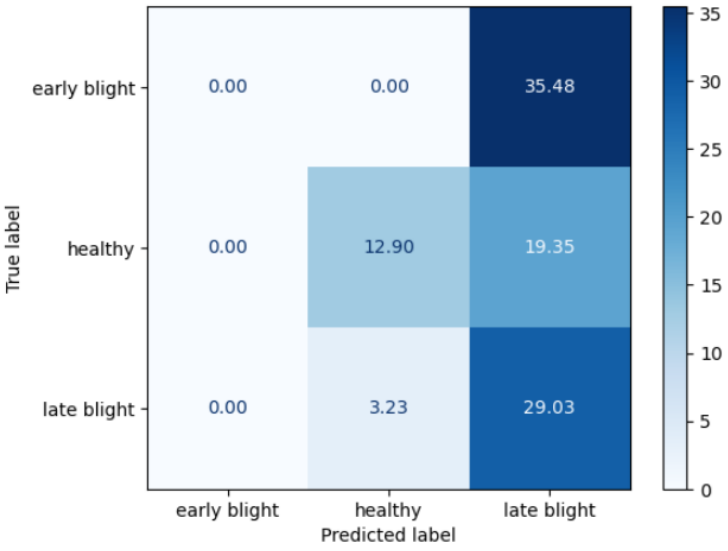
Também foi observado um certo delay ao clicar para realizar a predição. No entanto, esse tempo está relacionado ao processamento no servidor. Como o servidor local precisa buscar os modelos na nuvem, esse tempo de resposta é esperado. Ainda assim, o atraso não é significativo a ponto de comprometer a experiência do usuário.

Figura 10 – Comportamento do modelo para tomate, sem utilizar OpenCV.



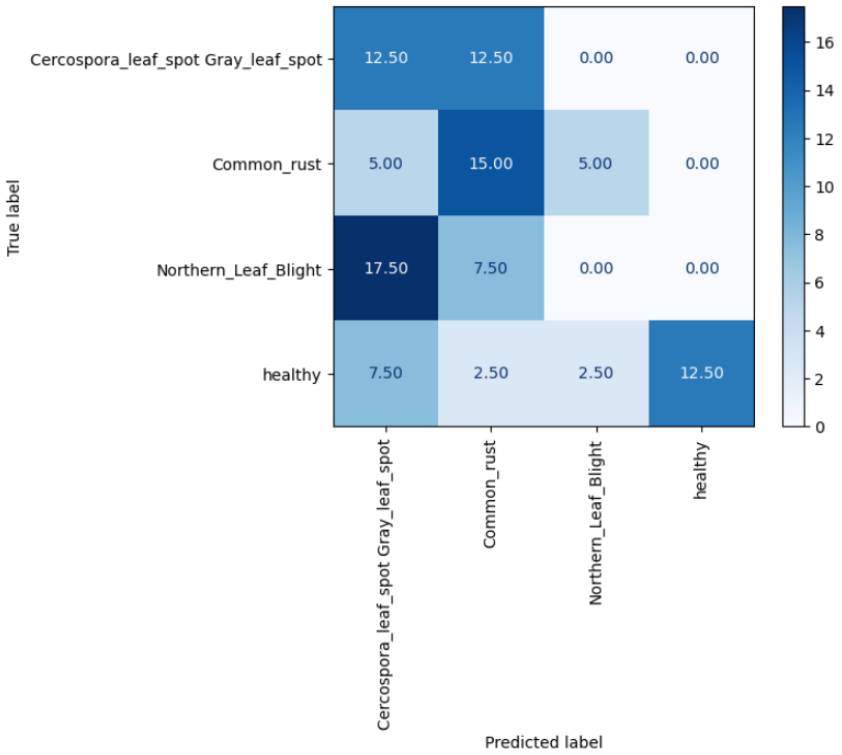
Fonte: Elaborado pelo autor, 2024.

Figura 11 – Comportamento do modelo para batata, utilizando OpenCV.



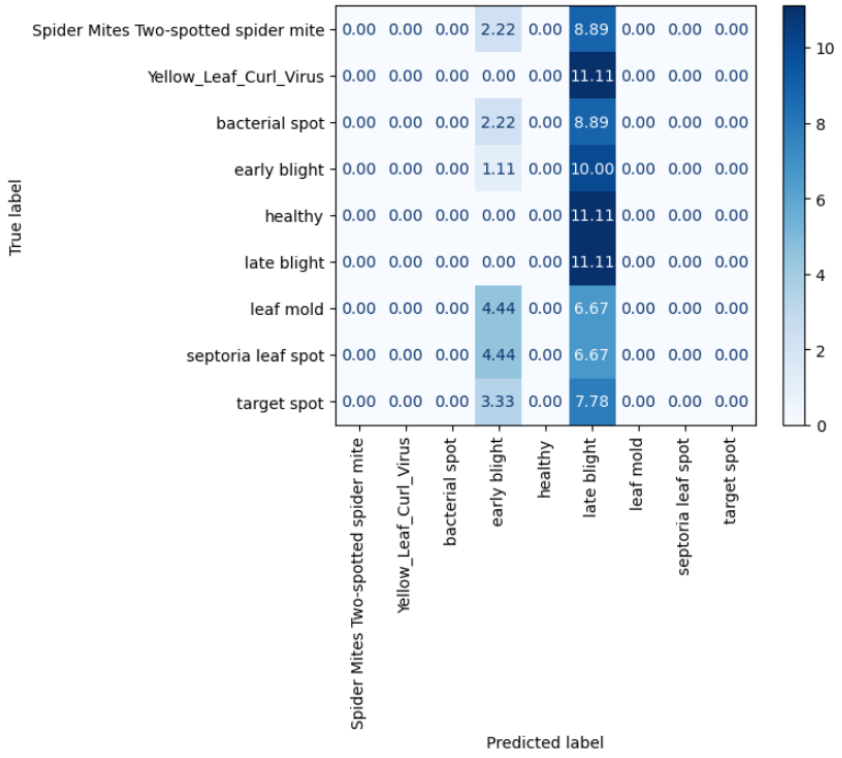
Fonte: Elaborado pelo autor, 2024.

Figura 12 – Comportamento do modelo para milho, utilizando OpenCV.



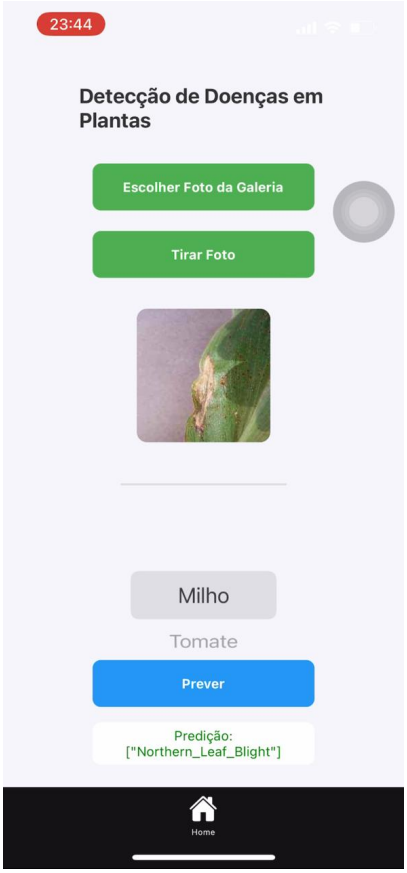
Fonte: Elaborado pelo autor, 2024.

Figura 13 – Comportamento do modelo para tomate, utilizando OpenCV.



Fonte: Elaborado pelo autor, 2024.

Figura 14 – Tela da interface criada, após predição.



Fonte: Elaborado pelo autor, 2024.

5 CONCLUSÃO E TRABALHOS FUTUROS

Ao longo do projeto, foi possível treinar e avaliar diversos modelos de classificação, como Decision Tree, Random Forest, SVC, KNN, Gradient Boosting e Stacking, utilizando métricas como MAE, MSE, RMSE e acurácia para medir seu desempenho. Os resultados obtidos demonstraram que, embora os modelos tenham se mostrado eficazes no conjunto de dados principal, desafios surgiram com a introdução de um segundo dataset, de menor qualidade, que revelou limitações na robustez dos modelos.

Foi observado um certo enviesamento na classificação de doenças semelhantes, especialmente entre Requeima e Pinta Preta, indicando que o modelo tende a confundir essas condições, principalmente em estágios mais avançados. Isso sugere que sintomas visuais semelhantes entre doenças podem afetar negativamente a precisão da classificação.

A tentativa de melhorar a acurácia utilizando técnicas de processamento de imagem com OpenCV para isolar a área da folha não trouxe os resultados esperados, evidenciando a complexidade de se trabalhar com imagens de baixa qualidade ou com ruídos.

A construção da interface do aplicativo, por sua vez, consegue permitir que usuários sem conhecimento técnico possam carregar ou capturar imagens e obter diagnósticos de forma simples e eficiente. A integração dos modelos treinados com a interface demonstrou a viabilidade de um sistema embarcado para análise local, apesar de desafios relacionados ao desempenho em dispositivos com menor capacidade.

Em resumo, este trabalho destacou o potencial do uso de aprendizado de máquina na agricultura, especialmente na identificação de doenças em plantas, mas também enfatizou a importância de comparar modelos complexos com modelos mais simples. Um dos pontos centrais do estudo foi justamente essa comparação, que revelou que, embora modelos mais avançados possam oferecer maior precisão, eles nem sempre são a melhor escolha para todos os cenários. A simplicidade de alguns algoritmos, como o SVC, mostrou que modelos menos complexos podem ser mais eficientes e viáveis em determinadas situações, especialmente em sistemas com recursos limitados, visto que demandam menor poder computacional, e gastam um menor tempo para treinar e classificar. Além disso, o trabalho evidenciou a necessidade de maior cuidado na coleta de dados de boa qualidade e uma análise mais detalhada das semelhanças entre diferentes classes de doenças, para evitar possíveis confusões no processo de classificação.

5.1 Trabalhos Futuros

Como trabalhos futuros, várias melhorias e expansões podem ser implementadas para aprimorar a solução desenvolvida neste projeto. Em primeiro lugar, um refinamento dos modelos de classificação pode ser realizado, utilizando técnicas avançadas de pré-processamento de imagens para lidar com doenças que apresentam sintomas visuais semelhantes. Outro fator poderia ser a implementação de um mecanismo de pré-verificação, onde o sistema seria capaz

de identificar se a imagem fornecida é realmente uma folha antes de realizar a classificação de doenças. Essa funcionalidade agregaria uma camada adicional de precisão ao sistema, evitando que imagens irrelevantes sejam analisadas.

Em relação à interface do usuário, melhorias podem ser feitas para deixar o aplicativo mais fácil de usar e mais interativo. Isso inclui sugerir tratamentos ou cuidados para as plantas.

Também é possível pensar em rodar o sistema em computadores mais potentes, como servidores na nuvem, para permitir que ele processe um grande número de imagens de forma mais rápida e eficiente.

REFERÊNCIAS

AHMED, A. A.; REDDY, G. H. A mobile-based system for detecting plant leaf diseases using deep learning. **AgriEngineering**, MDPI, v. 3, n. 3, p. 478–493, 2021. Citado na página 27.

ALI, A. **Base de dados Plant Village**. 2019. Acessado em 10/10/2024. Disponível em: <https://www.kaggle.com/datasets/abdallahalidev/plantvillage-dataset?resource=download>. Citado 2 vezes nas páginas 30 e 42.

AMARO, R. P. **Estimativa de produtividade da cana-de-açúcar a partir de imagens do satélite Sentinel-2A e o algoritmo de aprendizagem de máquina Random Forest**. Dissertação (Mestrado) — Universidade de São Paulo, 2023. Citado 2 vezes nas páginas 21 e 22.

BASTOS, J. A. D. M. **PROMOTING CONVERSATIONAL APIS: A CONCEPTUAL FRAMEWORK AND A METHOD FOR API DESIGN**. Tese (Doutorado) — PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO, 2020. Citado na página 25.

BIONDO, D. R. Classificação de doenças em batata baseado em imagens das folhas de batata utilizando deep learning. Universidade Federal de São Carlos, 2021. Citado na página 14.

CALDEIRA, R. F. **Detecção de doenças na cultura do algodoeiro através de processamento digital de imagens**. Tese (Doutorado) — [sn], 2020. Citado na página 14.

CARDOSO, L. J. d. A. **Ferramenta computacional para avaliação de pragas e doenças em cafezais por imagens**. Dissertação (Mestrado) — Universidade de Brasília, 2022. Citado na página 14.

CARMO, G. J. d. S. **Detecção de podridão mole em alface por Pectobacterium carotovorum subsp. carotovorum por algoritmos de aprendizado de máquina a partir de imagens multiespectrais**. Dissertação (Mestrado) — Universidade Federal de Uberlândia, 2021. Citado 2 vezes nas páginas 24 e 26.

CARVALHO, H. M. **Aprendizado de máquina voltado para mineração de dados: árvores de decisão**. Dissertação (Bacharelado) — Universidade de Brasília, 2014. Citado 2 vezes nas páginas 20 e 21.

CHANDRA, Y. U.; HARTONO, S. Analysis factors of technology acceptance of cloud storage: A case of higher education students use google drive. In: IEEE. **2018 International Conference on Information Technology Systems and Innovation (ICITSI)**. [S.l.], 2018. p. 188–192. Citado na página 25.

COSTA, E. M. da *et al.* Uso de deep learning para reconhecimento de doenças em bananicultura no município de paragominas/pa. **Revista de Ciências Agrárias Amazonian Journal of Agricultural and Environmental Sciences**, v. 67, p. 1–12, 2024. Citado na página 27.

FELIX, A. Y.; SASIPRABA, T. Flood detection using gradient boost machine learning approach. In: **2019 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE)**. [S.l.: s.n.], 2019. p. 779–783. Citado na página 23.

FIGUEIREDO, I. S. de; FREDIANI, J. O. R. F.; ARAUJO, M. de T.; PRADO, S. d. G. D.; COSTA, K. A. P. da. Modelo de autoencoder com ensemble learning e clusterização para detecção de intrusão em redes. **Revista Contemporânea**, v. 4, n. 6, p. e4910–e4910, 2024. Citado 2 vezes nas páginas 22 e 23.

FONTANA, C. P. A evoluÇÃo do trabalho: Da prÉ-histÓria atÉ ao teletrabalho. **Revista Ibero-Americana de Humanidades, Ciências e Educação**, v. 7, n. 7, p. 1155–1168, jul. 2021. Disponível em: <https://periodicorease.pro.br/rease/article/view/1759>. Citado na página 15.

GHOSH, S.; DASGUPTA, A.; SWETAPADMA, A. A study on support vector machine based linear and non-linear pattern classification. In: IEEE. **2019 International Conference on Intelligent Sustainable Systems (ICISS)**. [S.l.], 2019. p. 24–28. Citado na página 21.

GIRAFFA, L.; KHOLS-SANTOS, P. Inteligência artificial e educação: conceitos, aplicações e implicações no fazer docente. **Educação em Análise**, v. 8, n. 1, p. 116–134, jul. 2023. Disponível em: <https://ojs.uel.br/revistas/uel/index.php/educanalise/article/view/48127>. Citado na página 19.

GOMES, T. M. **A REUTILIZAÇÃO DA ÁGUA NA PRODUÇÃO DE HORTALIÇAS NO BRASIL**. Dissertação (Mestrado) — INSTITUTO FEDERAL DE EDUCAÇÃO CIÊNCIA E TECNOLOGIA DO TOCANTINS–CAMPUS PALMAS CURSO SUPERIOR DE TECNOLOGIA EM AGRONEGÓCIO, 2023. Citado na página 14.

GROSSO, L. R. G. V. **Cloud Rule-based System for Analysis of IoT Data in a Big Data Context**. Dissertação (Mestrado) — NOVA University of Lisbon, 2021. Citado na página 25.

HARAKAWA, F. S.; PEREIRA, A. S.; FANCISCATTO, R.; PERTILE, S. d. L.; PREUSS, E. Desenvolvimento e implementação de um aplicativo mobile para gerenciar um blog. **Caderno Pedagógico**, v. 20, n. 1, p. 305–327, Oct. 2023. Disponível em: <https://ojs.studiespublicacoes.com.br/ojs/index.php/cadped/article/view/1519>. Citado 2 vezes nas páginas 24 e 25.

JIANG, Z.-P.; LIU, Y.-Y.; SHAO, Z.-E.; HUANG, K.-W. An improved vgg16 model for pneumonia image classification. **Applied Sciences**, v. 11, n. 23, 2021. ISSN 2076-3417. Disponível em: <https://www.mdpi.com/2076-3417/11/23/11185>. Citado na página 24.

LEITE, A. **Código da Aplicação Construída**. 2024. Acessado em 10/10/2024. Disponível em: <https://github.com/ancleite/tcctests>. Citado na página 38.

LEITE, D. R. A.; MORAES, R. M. d.; LOPES, L. W. Método de aprendizagem de máquina para classificação da intensidade do desvio vocal utilizando random forest. **Journal of Health Informatics**, v. 12, mar. 2021. Disponível em: <https://www.jhi.sbis.org.br/index.php/jhi-sbis/article/view/814>. Citado 2 vezes nas páginas 21 e 22.

LUDERMIR, T. B. Inteligência artificial e aprendizado de máquina: estado atual e tendências. **Estudos Avançados**, SciELO Brasil, v. 35, p. 85–94, 2021. Citado na página 15.

MOREIRA, L.; FERREIRA, A.; ARRUDA, G. de; BRITO, R. de. A utilização da lógica fuzzy como ferramenta auxiliar na detecção de doenças fúngicas do tomateiro como a septoriose. In: **Anais da VIII Escola Regional de Computação do Ceará, Maranhão e Piauí**. Porto Alegre, RS, Brasil: SBC, 2020. p. 9–15. ISSN 0000-0000. Disponível em: <https://sol.sbc.org.br/index.php/ercemapi/article/view/11462>. Citado 2 vezes nas páginas 24 e 26.

MUFID, M. R.; BASOFI, A.; RASYID, M. U. H. A.; ROCHIMANSYAH, I. F. *et al.* Design an mvc model using python for flask framework development. In: IEEE. **2019 International Electronics Symposium (IES)**. [S.l.], 2019. p. 214–219. Citado na página 25.

NASCIMENTO, M. A. S. d. **Diagnóstico e alternativas fitossanitárias para a produção de hortaliças agroecológicas na APROFAM**. Dissertação (Bacharelado) — Universidade Federal Rural do Semi-Árido, 2020. Citado na página 19.

OLIVEIRA, A. M. de; MARACAJÁ, P. B.; FILHO, E. T. D.; LINHARES, P. C. F. Controle biológico de pragas em cultivos comerciais como alternativa ao uso de agrotóxicos. **REVISTA VERDE DE AGROECOLOGIA E DESENVOLVIMENTO SUSTENTÁVEL**, Grupo Verde de Agroecologia e Abelhas (GVAA), 2006. Citado na página 19.

PACHECO, A. G. C. Classificação de espécies de peixe utilizando redes neurais convolucional. **CoRR**, abs/1905.03642, 2019. Disponível em: <http://arxiv.org/abs/1905.03642>. Citado na página 24.

ROLA, E. d. C. d. S. **Os principais contributos da inteligência artificial para o processamento de imagens digitais a utilizar na segurança pública**. Dissertação (Mestrado) — Universidade Lusíada de Lisboa, 2022. Citado na página 15.

SICHMAN, J. S. Inteligência artificial e sociedade: avanços e riscos. **Estudos Avançados**, Instituto de Estudos Avançados da Universidade de São Paulo, v. 35, n. 101, p. 37–50, Jan 2021. ISSN 0103-4014. Disponível em: <https://doi.org/10.1590/s0103-4014.2021.35101.004>. Citado na página 19.

SILVA, I. P. C. A. d. **Um estudo sobre o TinyML a partir de uma aplicação mobile para detecção de doenças em folhas de manga**. Dissertação (Bacharelado) — Universidade Federal de Alagoas, 2023. Citado na página 27.

SILVA, J. C. P. *et al.* Ensemble classifiers in a serious game for medical students in clinical cases. In: **Proceedings of the 10th Euro-American Conference on Telematics and Information Systems**. New York, NY, USA: Association for Computing Machinery, 2021. (EATIS '20). ISBN 9781450377119. Disponível em: <https://doi.org/10.1145/3401895.3402068>. Citado 2 vezes nas páginas 23 e 24.

SILVA, T. P. d. P.; FLORêNCIO, R. R.; SANTOS, A. E. O. Qualidade pós-colheita de hortaliças biofortificadas: revisão bibliográfica. **Revista Ouricuri**, v. 14, n. 1, p. 03–20, mar. 2024. Disponível em: <https://www.revistas.uneb.br/index.php/ouricuri/article/view/19764>. Citado na página 14.

SOUSA, M. C. C. **Uma análise do algoritmo K-means como introdução ao Aprendizado de Máquinas.** Dissertação (Licenciatura) — Universidade Federal do Tocantins, 2023. Citado na página 20.

SOUZA, P. M. d.; FORNAZIER, A.; SOUZA, H. M. d.; PONCIANO, N. J. Diferenças regionais de tecnologia na agricultura familiar no brasil. **Revista de Economia e Sociologia Rural**, Sociedade Brasileira de Economia e Sociologia Rural, v. 57, n. 4, p. 594–617, Oct 2019. ISSN 0103-2003. Disponível em: <https://doi.org/10.1590/1806-9479.2019.169354>. Citado na página 19.

TEODORO, L.; KAPPEL, M. A. Aplicação de técnicas de aprendizado de máquina para predição de risco de evasão escolar em instituições públicas de ensino superior no brasil. **Revista Brasileira de Informática na Educação**, v. 28, n. 0, p. 838–863, 2020. ISSN 2317-6121. Disponível em: <http://milanesa.ime.usp.br/rbie/index.php/rbie/article/view/v28p838>. Citado na página 20.

VERMA, A.; MEHTA, S. A comparative study of ensemble learning methods for classification in bioinformatics. In: IEEE. **2017 7th International Conference on Cloud Computing, Data Science & Engineering-Confluence**. [S.l.], 2017. p. 155–158. Citado na página 15.

WU, T.; ZHANG, W.; JIAO, X.; GUO, W.; Alhaj Hamoud, Y. Evaluation of stacking and blending ensemble learning methods for estimating daily reference evapotranspiration. **Computers and Electronics in Agriculture**, v. 184, p. 106039, 2021. ISSN 0168-1699. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0168169921000570>. Citado na página 23.

XIE, G.; LU, W. Image edge detection based on opencv. **International Journal of Electronics and Electrical Engineering**, v. 1, n. 2, p. 104–106, 2013. Citado na página 26.