Mateus Prevelato Athayde

Sistema de recomendação de plantio utilizando Aprendizado de Máquina

UBERLÂNDIA

2023

Mateus Prevelato Athayde

Sistema de recomendação de plantio utilizando Aprendizado de Máquina

Projeto de Trabalho de Conclusão de Curso da Engenharia de Controle e Automação da Universidade Federal de Uberlândia - UFU - Campus Santa Mônica, como requisito para a obtenção do título de Graduação em Engenharia de Controle e Automação

Universidade Federal de Uberlândia – UFU

Faculdade de Engenharia Elétrica

Orientador: Prof. Dr. Renato Ferreira Fernandes Júnior

UBERLÂNDIA

2023

Ficha Catalográfica Online do Sistema de Bibliotecas da UFU
com dados informados pelo(a) próprio(a) autor(a).

A865 Athayde, Mateus Prevelato, 1996-
2023 Sistema de recomendação de plantio utilizando Aprendizado de Máquina [recurso eletrônico] / Mateus Prevelato Athayde. - 2023.

Orientador: Renato Ferreira Fernandes Júnior.
Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Uberlândia, Graduação em Engenharia de Controle e Automação.
Modo de acesso: Internet.
Inclui bibliografia.
Inclui ilustrações.

1. Processos de fabricação - Automação. I. Fernandes Júnior, Renato Ferreira ,1971-, (Orient.). II. Universidade Federal de Uberlândia. Graduação em Engenharia de Controle e Automação. III. Título.

CDU: 67.02:681.3

Bibliotecários responsáveis pela estrutura de acordo com o AACR2:
Gizele Cristine Nunes do Couto - CRB6/2091
Nelson Marcos Ferreira - CRB6/3074

Mateus Prevelato Athayde

Sistema de recomendação de plantio utilizando Aprendizado de Máquina

Projeto de Trabalho de Conclusão de Curso da Engenharia de Controle e Automação da Universidade Federal de Uberlândia - UFU - Campus Santa Mônica, como requisito para a obtenção do título de Graduação em Engenharia de Controle e Automação.

Uberlândia (MG), 20 de janeiro de 2023.

Banca de Avaliação:

_____

Prof. Dr. Renato Ferreira Fernandes Júnior
Orientador

_____

Prof. Dr. Josué Silva de Morais
Membro

_____

Prof. Dr. Márcio José da Cunha
Membro

Agradecimentos

Agradeço primeiramente a Deus e aos meus pais, Regina e Eduardo, por todo o apoio, amor e investimento na minha educação.

À toda a minha família, em especial ao meu irmão Eduardo.

Agradecimentos também a minha namorada Mariana, pelo companheirismo e amor.

Aos amigos, em especial aos colegas de curso Arthur, Gustavo, Henrique, Patrick, Vitor, Kenji e ao amigo de infância Daniel pela amizade e apoio.

Aos meus professores, em especial ao Renato, pela orientação durante a realização deste trabalho.

Por fim, a Universidade Federal de Uberlândia, por ter me proporcionado um enriquecimento acadêmico, pessoal e profissional durante a minha formação.

"A ciência é mais que um corpo de conhecimento, é uma forma de pensar, uma forma cética de interrogar o universo, com pleno conhecimento da falibilidade humana.",
(Carl Sagan)

Resumo

Através de sistemas computacionais de gestão de fazendas, drones, veículos autônomos, sensores de umidade e temperatura, houve um enorme crescimento na quantidade de dados provenientes de atividades agronômicas. Esses dados passaram então a ser utilizados em algoritmos de Aprendizado de Máquinas de forma a buscar maior rendimento de plantio, previsão de colheitas, reconhecimento de doenças em plantas, recomendação de plantio entre outras aplicações. Este trabalho visa o desenvolvimento de um sistema de recomendação de plantio utilizando algoritmos de Aprendizado de Máquinas, de modo a facilitar a escolha da melhor cultura de acordo com a base de dados utilizada, que possui as características do local, como: umidade, temperatura, pH e nutrientes presentes no solo.

Para a criação e compartilhamento desse sistema de recomendação de plantio, foram utilizados algoritmos voltados para a classificação de dados como Árvore de Decisão, Floresta Aleatória, *Naive Bayes*, entre outros. Os modelos foram criados utilizando a linguagem de programação Python e compartilhados através da biblioteca Streamlit. Assim, após a criação e comparação dos modelos, foi incorporado dentro do aplicativo *Web*, criado através do Streamlit, o algoritmo de Floresta Aleatória. Ao final, foi possível a criação de um aplicativo com ótima performance de recomendação, intuitivo e de fácil utilização para o usuário.

Palavras-chave: Aprendizado de Máquinas, Recomendação de Plantio, Agricultura de Precisão.

Abstract

Through computer systems for managing farms, drones, autonomous vehicles, humidity and temperature sensors, there has been a huge growth in the amount of data coming from agronomic activities. This data then began to be used in Machine Learning algorithms in order to seek greater planting yield, harvest prediction, recognition of plant diseases, planting recommendation among other applications. This work aims at the development of creating and sharing a planting recommendation system using Machine Learning algorithms, in order to facilitate the choice of the best crop according to the database used, which has the characteristics of the place, such as: humidity, temperature, pH and nutrients present in the soil.

For the creation and sharing of this planting recommendation system, algorithms aimed at data classification were used, such as Decision Tree, Random Forest, Naive Bayes, among others. The models were created using the Python programming language and shared through the Streamlit library. Thus, after creating and comparing the models, it was incorporated into the web application, created through Streamlit, the Random Forest algorithm. In the end, it was possible to create an application with excellent recommendation performance, intuitive and easy to use for the user.

Keywords: Machine Learning, Crop Recommendation, Cutting-edge Agriculture.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

AD	*Árvore de Decisão*
AM	*Aprendizado de Máquina*
ATP	*Trifosfato de adenosina*
CSV	*Comma-separated values*
FA	*Floresta Aleatória*
IA	*Inteligência Artificial*
KNN	*K-Nearest Neighbors*
MVS	*Máquina de Vetores de Suporte*
NB	*Naive Bayes*
NPK	*Nitrogênio, Fósforo e Potássio*
PCA	*Principal Component Analysis*
SQL	*Structured Query Language*
SVM	*Support Vector Machine*
TIC	*Tecnologias de informação e comunicação*

Sumário

1 Introdução

O avanço tecnológico, principalmente nas últimas décadas, possibilitou a obtenção de uma enorme quantidade de dados, seja através de sites, smartphones, aplicativos de streaming, dados financeiros, operacionais, médicos entre outros. Essa expansão causou um considerável aumento na utilização de técnicas de Aprendizado de Máquina (AM) em diferentes aplicações e setores da sociedade. Não somente vinculados à robótica e ao setor industrial, mas à criação de veículos autônomos, reconhecimento de voz, reconhecimento de vídeos e imagens, previsão de rendimento de plantações e detecção de fraudes em sistemas financeiros (QIN; CHIANG, 2019).

Em relação a agricultura, ela tem enfrentado diversos desafios, seja pelas mudanças climáticas, degradação do meio ambiente, crescimento populacional, entre outros fatores. Algumas maneiras de reduzir o impacto dessas questões na agricultura é através da modernização e avanços tecnológicos no setor, de modo a suprir essa necessidade urgente de se obter maior rendimento de plantio e reduzir os efeitos causados no meio ambiente, gerando assim a transformação em agricultura de precisão (BENOS *et al.*, 2021).

Alguns pré-requisitos necessários para a agricultura moderna seriam: a adoção de Tecnologias de Informação e Comunicação (TIC) que incluem sensores de umidade e solo, sistemas para gestão de fazendas, redes de internet sem fio, câmeras, drones, satélites, veículos autônomos (SORENSEN *et al.*, 2019). Com o grande volume de dados gerados pelas diversas tecnologias digitais utilizadas, usualmente conhecido como *Big Data*, cria-se a necessidade de armazenar, analisar e interpretar esse grande volume de informação, já que as técnicas convencionais não incapazes de interpretar e tratar essa grande quantidade de dados. Dessa forma, o Aprendizado de Máquina surge como uma solução para extrair informações valiosas através dos dados (SORENSEN *et al.*, 2019).

Existem diversas aplicações de Aprendizado de Máquina dentro da agricultura, sendo divididas em quatro categorias genéricas de estudos: plantio, solo, água e manejo de gado (LIAKOS *et al.*, 2018).

É possível encontrar diversos estudos voltados para a utilização de AM dentro da agricultura e também estudos voltados para a criação de modelos de classificação utilizando AM. Dentre os diversos estudos, podemos destacar os autores como: Alvim (2019), Souza (2021), Moraes (2022), Benos *et al.* (2021), Souza (2021) e Thilakarathne *et al.* (2022).

Apesar de vários estudos já presentes referentes a parte de colheita e plantio, existem questões que ainda podem ser exploradas e desenvolvidas, principalmente no que se trata em relação a sistemas de recomendação de plantio. Dentro desse contexto, este trabalho busca contribuir com este tema nas questões de criação de modelos de AM, métricas avaliativas de modelos, implantação e distribuição do modelo através de aplicações *web* e agricultura de precisão.

1.1 Justificativa

Este trabalho tem por motivação acompanhar os avanços tecnológicos presentes na agricultura atualmente, e conectar o ambiente acadêmico com as necessidades atuais do mercado agro através da criação de um sistema de recomendação de plantio.

Atualmente, a agronomia e a agroindústria se beneficiam muito da tecnologia, não somente em equipamentos, mas nos dados obtidos com sementes, plantios, defensivos agrícolas, localização e mapeamento geográfico para obter maior rendimento, segurança e benefício ao meio ambiente. Apesar do foco ser em um sistema de recomendação do plantio, este estudo se concentra nos algoritmos que possibilitam a criação desse sistema.

No âmbito acadêmico, o estudo realizado nesse projeto incrementa discussões relacionadas a este tipo de modelo. Por se tratar não somente de programação, mas de estatística, análise de dados e modelagem de sistemas, se torna um estudo rico no âmbito multidisciplinar no curso de Engenharia de Controle e Automação.

1.2 Objetivo

O objetivo deste trabalho é elaborar um sistema de recomendação de plantio, de acordo com dados acadêmicos que possuem as características do local, criando assim algoritmos de classificação utilizando a linguagem de programação Python. Os modelos de AM a serem criados serão: Árvore de Decisão, Floresta Aleatória, Máquina de Vetores de Suporte, k-Vizinhos Mais Próximos e *Naive Bayes*.

A base de dados a ser utilizada no sistema tanto de treinamento quanto de teste e validação será baseada em dados acadêmicos abertos de domínio público, de modo a não comprometer dados coletados por empresas.

1.2.1 Objetivos específicos

- Análise exploratória dos dados;
- Criação dos modelos de classificação utilizando a linguagem Python;
- Análise da performance de cada modelo (escalabilidade, tempo de complicação);
- Comparação estatística entre os modelos utilizando métricas como: acurácia, precisão, *recall*, entre outros;
- Desenvolvimento de uma interface *web* para interação do usuário, publicação e compartilhamento do melhor modelo.

2 Referencial teórico

Neste capítulo serão apresentados alguns fundamentos teóricos cujos conhecimentos são necessários para uma melhor compreensão necessária para o desenvolvimento a ser realizado. Dessa forma, esse capítulo se concentrará nos fundamentos de aprendizado de máquinas, aprendizado supervisionado e não supervisionado e métricas estatísticas para medir o desempenho dos algoritmos.

2.1 Planejamento do plantio

O planejamento é essencial em qualquer trabalho, e na agricultura também se mostra dessa forma. Um dos principais fatores a se levar em conta logo no início do planejamento é observar o que acontece em sua volta, para considerar questões como: necessidade da população, capacidade de atender a demanda, tamanho da área de cultivo, distância do produto aos centros comerciais, armazenamento (JACTO, 2018).

O conhecimento da natureza influencia diretamente na questão da escolha da cultura a ser plantada e como será seu rendimento. Com isso, um estudo prévio de zoneamento climático se torna essencial. Assim, é possível conhecer as condições climáticas da região e decidir qual cultura melhor se adapta a ela e como será a produtividade nesse local (CANAL AGRO, 2020).

A adubação também está entre os principais fatores para se obter um bom rendimento na colheita, já que é fundamental para o desenvolvimento correto das plantas (CANAL AGRO, 2020). Quando ocorre a decomposição de uma planta, seus nutrientes retornam ao solo, gerando o desenvolvimento das outras plantas a sua volta e assim repetindo um ciclo natural. Com a colheita, ocorre a interrupção desse ciclo, criando a necessidade de adubação.

2.1.1 Aspectos climáticos

Aspectos como intensidade luminosa, umidade do ar, umidade do solo, temperatura e quantidade de chuvas são fundamentais para determinar a cultura a ser plantada, já que estão diretamente relacionadas ao sucesso da colheita. Lembrando ainda que esses aspectos devem ser atendidos de forma dinâmica, porque cada etapa de plantio, desenvolvimento da planta e de colheita possui necessidades diferentes de

temperatura, umidade do ar/solo, chuva, entre outros (CANAL AGRO, 2020).

Focando na temperatura e na chuva, pode-se citar seus impactos no plantio:

- Temperaturas altas: podem queimar as folhas, criando a necessidade de maior irrigação nas plantas. Nas etapas de florescimento e frutificação, temperaturas acima do recomendado podem ocasionar a queda desses itens;
- Temperaturas baixas: temperaturas abaixo do ideal podem ocasionar a pausa na floração, impactando negativamente seu ciclo reprodutivo;
- Índice pluviométrico: as chuvas são essenciais durante os ciclos de uma planta. Esta necessidade se altera dependendo do ciclo em que ela se encontra. Assim, o excesso ou a falta de chuvas podem impactar diretamente a produtividade.

2.1.2 Nutrição e adubação

Para que aconteça o crescimento e desenvolvimento adequado das plantas, elas precisam dos nutrientes corretos e também um nível de pH (Potencial Hidrogeniônico) adequado, criando assim a necessidade de se utilizar fertilizantes. Um dos adubos mais utilizados, é conhecido como Fertilizantes NPK, justamente por possuir os nutrientes mais necessários para as plantas, sendo eles o nitrogênio, o fósforo e o potássio (DUARTE, 2020).

2.1.2.1 Nitrogênio

O nitrogênio, possui o efeito mais rápido no que se diz sobre o crescimento vegetal. Na planta, possui como função básica o crescimento, e também é responsável pela coloração verde escura delas, desenvolvendo o sistema radicular e melhorando a absorção de outros nutrientes do solo. O nitrogênio faz parte da composição proteica de todas as plantas e animais. No solo, o nitrogênio se comporta como cátion (NH_4^+) e também como o ânion (NO_3^-). A maior parte se encontrar na forma de NO_3^-, que costuma ser lixiviada para fora da zona de absorção das raízes (SENGIK, 2003).

Os principais sinais de deficiência do nitrogênio podem ser observados em gramíneas, quando a coloração de suas folhas mais velhas se torna um amarelo-esverdeada. As plantas que possuem essa deficiência apresentam folhas pequenas, caules finos e não muitas ramificações. No milho, por exemplo, pode-se citar a redução no tamanho das espigas. Um exemplo de excesso de nitrogênio são os cabelos das espigas de milho que permanecem verdes.

2.1.2.2 Fósforo

O fósforo é absorvido pelas raízes na forma de íon ortofosfato ($H_2PO_4^-$). O fósforo tem uma grande relevância na formação do ATP (trifosfato de adenosina) que é a principal fonte de energia da planta. Sendo essa energia utilizada no transporte de assimilados, no armazenamento e transferência de energia, na divisão celular, no crescimento de células e na transferência de informações de sequências genéticas. No solo, os que possuem acidez elevada tende a predominar a forma de ortofosfato primário ($H_2PO_4^-$), já em solos alcalinos predomina o íon ortofosfato secundário (HPO_4^{--}). Na maioria dos casos, o pH que proporciona maior disponibilidade de fósforo está entre 6,0 e 6,5, já que nessa faixa a reação ou fixação do fósforo é mínima. O valor de pH ideal é considerado 6,3 no quesito disponibilidade de fósforo as plantas (SENGIK, 2003).

A deficiência de fósforo pode ser observada através da coloração púrpura aparecendo nas folhas mais velhas, ocasionando também um menor crescimento da planta. No caso do milho, por exemplo, a falta de fósforo ocasiona redução no tamanho das espigas, torção na ponta e com grãos pequenos.

2.1.2.3 Potássio

O potássio, absorvido como um íon cátion (K^+), regula e participa de muitos processos essenciais nas plantas, sendo eles: fotossíntese, absorção de água do solo, síntese proteica, entre outros. Alguns produtos agrícolas possuem sua qualidade dependente da disponibilidade de potássio, como o teor de açúcar na cana-de-açúcar, tamanho de frutos cítricos e a durabilidade de hortaliças. No solo, o potássio se comporta como íon cátion monovalente e dessa forma pode ser facilmente absorvido, fixado, adsorvido as argilas ou permanecer na solução do solo (SENGIK, 2003).

O sintoma típico de deficiência de potássio é a clorose das margens das folhas mais velhas, no milho por exemplo se dá com espigas com poucos grãos na extremidade e com sementes se soltando no sabugo. Em contrapartida, caso ocorra o consumo em excesso pela planta, é um desperdício de recursos e desnecessária.

2.1.3 A Influência do pH

O conhecimento dos processos químicos relacionados a agricultura é fundamental para os produtos garantirem uma melhor fertilidade de seu plantio. Dessa forma, é essencial a análise e correção, quando necessária, do pH do solo (BRFÉRTIL,

2019).

O pH indica a quantidade de íons de hidrogênio presentes em um líquido. Com a concentração maior, essa substância é considerada ácida. A variação do valor se dá de 0 a 14, com a seguinte classificação:

- Abaixo de 7 – pH ácido
- Igual a 7 – pH Neutro
- Acima de 7 – pH básico

O pH do solo conduz diretamente as reações do solo. Na maioria dos casos, os valores de pH entre 5,5 e 6,5 são considerados favoráveis ao desenvolvimento da maioria das culturas. Os macronutrientes nitrogênio, fósforo, potássio, cálcio, magnésio e enxofre, direta ou indiretamente, tem sua disponibilidade aumentado quando os valores de pH estão próximos a neutralidade (normalmente entre 6,0 e 6,5). Quando o pH tende a acidez, abaixo de 5,5, tem a possibilidade de ocorrer danos ao desenvolvimento das plantas devido à elevada atividade de elementos eventualmente fitotóxicos, como por exemplo o Alumínio e o Manganês (BRFÉRTIL, 2019).

2.2 Conceitos básicos de Aprendizado de Máquina

2.2.1 O que é Aprendizado de Máquina

Aprendizado de Máquina, ou em inglês, *Machine Learning* é um ramo da inteligência artificial (IA) que tem seu foco em utilizar dados e algoritmos para imitar o pensamento humano e gradualmente melhorar sua precisão (IBM, 2020).

O Aprendizado de Máquina é uma das maneiras de se utilizar IA. O aprendizado de máquinas foi definido na década de 1950 pelo pioneiro em IA Arthur Samuel como "A área de estudos que possibilitam os computadores a habilidade de aprenderem sem serem explicitamente programados para isso" (MIT, 2021).

Um exemplo que se pode dar é o filtro de *spam* que possui nos *e-mails*. Já que através de um algoritmo de AM, consegue-se fornecer dados, em que alguns exemplos de *e-mails* estão classificados como *spam* e outros classificados como autênticos. Dessa forma esse modelo criado através desse tipo de tecnologia consegue aprender com os exemplos e, posteriormente, classificar novos *e-mails* de acordo com as características apresentadas.

Assim, a área de Aprendizado de Máquina se trata sobre extrair conhecimento de dados, é um campo de pesquisa na intersecção entre estatística, inteligência

artificial, e ciências da computação, mas também é conhecida como análise preditiva ou aprendizado estatístico.

A aplicação dos métodos de AM tem se tornados onipresentes no cotidiano. Englobando desde recomendações de filmes para serem assistidos, qual refeição pedir ou produtos para comprar, reconhecimento de conhecidos em fotos (GUIDO, 2016).

Com isso se pode citar as principais vantagens do Aprendizado de Máquina (MORAIS, 2020):

- Capacidade de maior aprendizado e melhorar através dos erros;
- Velocidade com que os dados podem ser analisados;
- Automatização de processos;
- Cria soluções para o mundo real;
- Redução de custos.

Apesar dos inúmeros exemplos de aplicações e vantagens, ainda existem desvantagem e dificuldades dentro das aplicações de Aprendizado de Máquina, sendo essas:

- Disseminação de informações enviesadas sobre as técnicas de AM;
- Grande volume de dados disponíveis em baixa qualidade, criando a necessidade de realizar tratamento nos dados para evitar ruídos e erros;
- Carência de profissionais qualificados;
- Falta de planejamento e organização na implantação de projetos;
- Rejeição com novas tecnologias nas empresas.

Assim, com os conceitos descritos, o Aprendizado de Máquina pode ser descrito como um conjunto de técnicas e abordagens, com as quais é possível aprender padrões e criar modelos a partir de dados. Com esses modelos é possível realizar predições de valores futuros ou inexplorados.

2.2.2 Histórico do Aprendizado de Máquina

A origem do Aprendizado de Máquina começa em 1943, com o primeiro modelo matemático de redes neurais, que foi apresentado no artigo científico "*A logical calculus of the ideas imanente in nervous activity*" por Walter Pitts e Warren McCulloch. Já em 1949, o livro "*The Organization of Behavior*" por Donald Hebb é publicado e conta com teorias de que relacionam o comportamento humano com redes

neurais e atividades cerebrais, se tornando um dos principais pilares do desenvolvimento do Aprendizado de Máquina. Alan Turing, em 1950, criou o Teste de Turing para determinar se um computador possui inteligência de verdade. Para que se passasse no teste, o computador deveria enganar um humano, fazendo-o acreditar que ele também é um humano, sendo apresentado o seu princípio de funcionamento enquanto trabalha da Universidade de Manchester. O artigo "*Computing Machinery and Intelligence*" inicia com as seguintes palavras: "Eu proponho a considerar a questão, 'Máquinas podem pensar?'" (FIRICAN, 2019).

Na década de 50, Arthur Samuel, na época um cientista da IBM, lançou o primeiro programa de aprendizado voltado para o jogo de damas, dessa forma, a máquina aprimorava suas habilidades no jogo à medida em que jogava. Assim, o computador começa a propor melhores estratégias e sempre incorporando novos aprendizados. Nesta mesma época, teve o surgimento do Perceptron, que foi criado por Frank Rossenblatt, que foi a primeira rede neural para computadores. Um outro passo importante no aprendizado de máquinas foi na década de 1960, em que foi criado o algoritmo conhecimento como "vizinhos próximos" (*nearest neighbor*), que possibilitou computadores a usar reconhecimento básico de padrões.

Pode-se considerar a década de 1990 onde o Aprendizado de Máquina realmente despontou, principalmente por ser orientada a dados. Dessa forma os cientistas começaram a criar programas para computadores que analisavam grandes quantidades de dados e obtinham conclusões, ou aprendiam com esses resultados. Um marco dessa época podemos considerar em 1997, em que o supercomputador IBM Deep Blue venceu o campeão mundial de xadrez. Nos anos 2000, o termo *Deep Learning* (Aprendizagem Profunda) foi inventado por Geoffrey Hinton para explicar novos algoritmos capazes de dar a habilidade de "ver" e diferenciar objetos aos computadores. Outros grandes marcos foram o Kinect, desenvolvido pela Microsoft que detectava até 20 movimentos humanos distintos em uma frequência de 30 vezes por segundo, possibilitando as pessoas interagirem com os computadores através de gestos e movimentos. É possível mencionar também o Facebook, que em 2014 desenvolveu o DeepFace, um programa capaz de reconhecer e verificar humanos em fotografias, da mesma forma que uma pessoa faria.

É cabível observar, após esse breve histórico, que apesar de ser uma tecnologia que surgiu há várias décadas, o Aprendizado de Máquina conseguiu sua maior evolução e destaque a partir da década de 1990, sem sombra de dúvidas ao fato da evolução tecnológica vertiginosa nessa década e nos anos posteriores, justificada devido a maior de processamento dos computadores, a internet e a maior disponibilidade de dados. Hoje consegue-se perceber em todos os campos a utilização dessa tecnologia e a necessidade de profissionais especializados na área.

2.2.3 Tipos de Aprendizado de Máquina

Existem vários tipos de sistemas de Aprendizado de Máquina, assim, facilita classificá-los eles em duas grandes categorias, sendo (GÉRON, 2017):

- Se são ou não treinados através de intervenção humana (supervisionado, não supervisionado, semisupervisionado e aprendizado por reforço);
- Se podem ou não aprender incrementalmente em tempo real (*online* ou em lote);
- A forma de trabalho do sistema é baseada em comparar novos dados com dados conhecidos, ou procurar padrões nos dados de treinamento e constrói um modelo preditivo, assim como cientistas fazem (aprendizado baseado em instância ou em modelo).

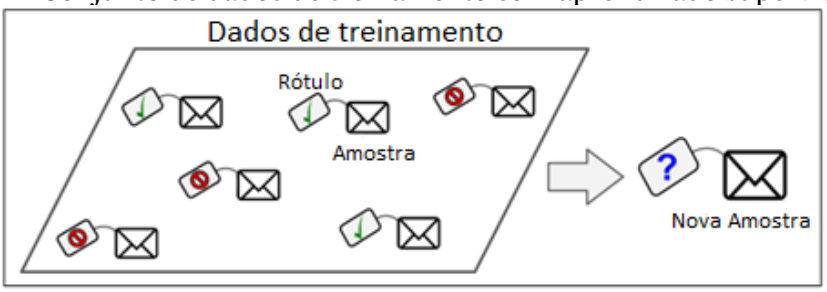
Como essas categorias não são exclusivas, é admissível ter por exemplo um sistema supervisionado, que aprende em tempo real baseado em um modelo. Para melhor compreensão, será discutida cada uma dessas categorias.

2.2.3.1 Aprendizado supervisionado e não supervisionado

Os sistemas de Aprendizado de Máquina podem ser classificados de acordo com a quantidade e tipo de supervisão que eles recebem durante seu treinamento. São categorizados majoritariamente em quatro categorias, sendo elas: supervisionado, não supervisionado, semisupervisionado e aprendizado por reforço (GÉRON, 2017).

No aprendizado supervisionado, exemplificado pela Figura 1, os dados de treinamento que são inseridos no algoritmo incluem o valor desejado, chamado de rótulo (*label*) (GÉRON, 2017).

Figura 1 – Conjunto de dados de treinamento com aprendizado supervisionado



Fonte: Adaptado de (GÉRON, 2017)

Um exemplo típico de aprendizado supervisionado são os sistemas de classificação. Na prática, um filtro de *spam* de *e-mail* é um bom exemplo, devido ao sistema ser treinado com vários exemplos de *e-mail*, juntos de seu rótulo (*spam* ou

legítimo), podendo assim classificar novos *e-mails*.

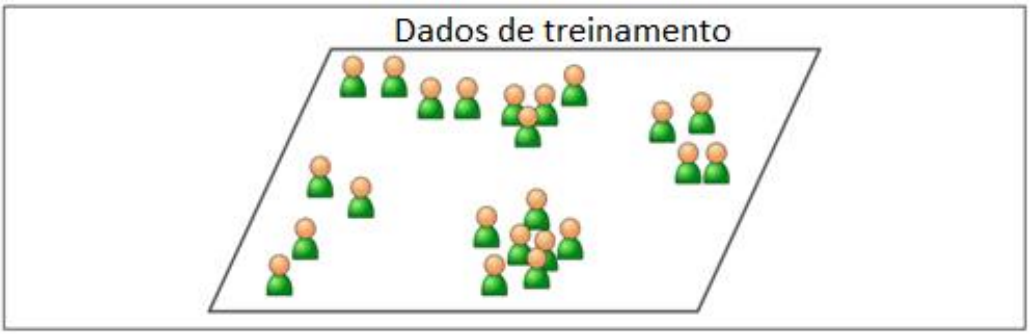
Outra aplicação, seria por exemplo, prever um valor numérico, como o preço de um carro dependendo de seus atributos (ano de fabricação, quilometragem, marca, entre outros). Esse tipo de trabalho é denominado regressão. Alguns algoritmos podem realizar os dois tipos de função.

Alguns dos principais algoritmos de aprendizado supervisionado são (GÉRON , 2017):

- K-Vizinhos Mais Próximos (*K-Nearest Neighbors*);
- Regressão Linear;
- Regressão Logística;
- Máquina de Vetores de Suporte (SVM);
- Arvore de Decisão (*Decision Tree*);
- Floresta Aleatória (*Random Forest*);
- Reforço de Gradiente (*Gradient Boosting*);
- *Naive Bayes*;
- Redes Neurais.

No caso do aprendizado não supervisionado, exemplificado pela Figura 2, o conjunto de dados de treinamento não é rotulado, dessa forma o sistema tenta aprender de forma independente.

Figura 2 – Conjunto de dados de treinamento com aprendizado não supervisionado



Fonte: Adaptado de (GÉRON, 2017)

Alguns dos mais importantes algoritmos de aprendizado não supervisionado são:

- Agrupamento (*Clustering*):
  - *K-Means*;

- Agrupamento Hierárquico (*Hierarchical Clustering*).

- Visualização e Redução de Dimensionalidade:
- Componente Principal de Análise (PCA);
- *Kernel* PCA.

Um exemplo de aplicação, seria o usuário possuir dados sobre os visitantes de seu blog. Com um algoritmo de Agrupamento é possível detectar grupos de visitantes que possuem similaridades entre si, e em nenhum momento é dito ao computador a qual grupo os visitantes pertencem. Detectar anomalias também é uma aplicação muito utilizando nesse tipo de supervisão, em que o computador detecta o comportamento normal presente na maioria dos dados, assim, ele consegue dizer se um novo dado possui o mesmo comportamento ou não, sendo muito presente em algoritmos de prevenção de fraudes bancárias.

Alguns algoritmos podem compartilhar tanto dados de treinamento supervisionados quanto não supervisionados, sendo categorizados como aprendizado semisupervisionado. O Google Fotos é um exemplo desse tipo de algoritmo, pois além de detectar automaticamente a mesma pessoa em diferentes fotos, ele pergunta para o usuário quem ela é, rotulando-a.

Aprendizado por reforço se trata de uma outra abordagem. O sistema de aprendizado, denominado agente, observa o ambiente, seleciona e executa ações, recebendo recompensas em retorno (ou penalidades). Dessa forma ele aprende, de forma autônoma, qual a melhor estratégia ou política, para receber a maior quantidade de recompensas. Um exemplo seria os robôs que implementam esse tipo de algoritmo para aprender a andar.

2.2.3.2 Treinamento em lote ou *online*

Outra maneira de se classificar os algoritmos de Aprendizado de Máquina é se o sistema é capaz de aprender incrementalmente através de um fluxo de dados.

O aprendizado por lotes é incapaz de aprender de maneira incremental ou *online*. Neste caso, é necessário que ele seja treinado com todos os dados disponíveis (GÉRON, 2017). Normalmente leva bastante tempo e é necessário muito processamento computacional, por isso é feito de forma *offline*. O sistema é treinado, lançado em produção e funciona sem aprender mais. Esse processo é denominado como aprendizado offline. Caso necessite inserir novos dados, é necessário treinar uma nova versão do modelo, com os novos dados e os dados antigos juntos, e substituir o antigo modelo por esse novo.

Esse tipo de aprendizado é fácil de ser automatizado e implementado, as desvantagens são o tempo e a necessidade de um equipamento condizente com o volume de dados para processas e treinar o novo modelo em um tempo adequado.

Em um sistema com aprendizado *online*, o treinamento desse sistema é realizado de maneira incremental, em que é alimentado por dados de forma sequencial, seja individual ou em grupos de dados (mini lotes). Cada etapa é rápida e precisa de pouco processamento, dessa forma o sistema aprende sobre novos dados de forma rápida, em tempo real (GÉRON, 2017). Aprendizado online é ótimo para sistema que recebem dados novos em um fluxo contínuo e necessitam se adaptar de forma rápida e autônoma. Também é uma ótima opção quando se possui recursos computacionais limitados. Esse tipo de aprendizado também economiza bastante espaço, já que após o sistema aprender com os novos dados inseridos, esses dados podem ser descartados.

Um parâmetro muito importante do aprendizado *online* é a taxa de aprendizado, já que caso se adaptarem rapidamente com os novos dados, o sistema acaba por esquecer os dados antigos, e caso isso aconteça de forma lenta, os novos dados não terão muito impacto no sistema. Assim, é importante estar monitorando de perto o sistema, para que não ocorram perdas na performance, caso os novos dados sejam de baixa qualidade por exemplo.

2.2.3.3 Aprendizado baseado em instância ou em modelo

Uma outra forma de se categorizar sistemas de Aprendizado de Máquina é de acordo com a forma que eles "generalizam". A maioria dos sistemas trabalham fazendo predições, ou seja, dando a eles vários exemplos, o sistema se torna capaz de inferir respostas a exemplos que não haviam sido vistos anteriormente. Ter uma boa performance nos dados de treinamento é algo bom, mas insuficiente, o objetivo é ter uma boa performance em novos dados (GÉRON, 2017).

No aprendizado baseado em instância, o algoritmo realiza a medição das similaridades dos novos dados com os dados de treinamento. No caso do exemplo dado anteriormente sobre o filtro de *spam*, o sistema identifica similaridades em *e-mails*, como por exemplo, o número de palavras em comum com *e-mails* classificados como suspeitos ou fraudulentos. Assim, quanto mais próximo desse valor, maior a probabilidade desse novo *e-mail* a ser classificado ser, de verdade, um *spam*. A Figura 3 exemplifica este tipo de aprendizado.

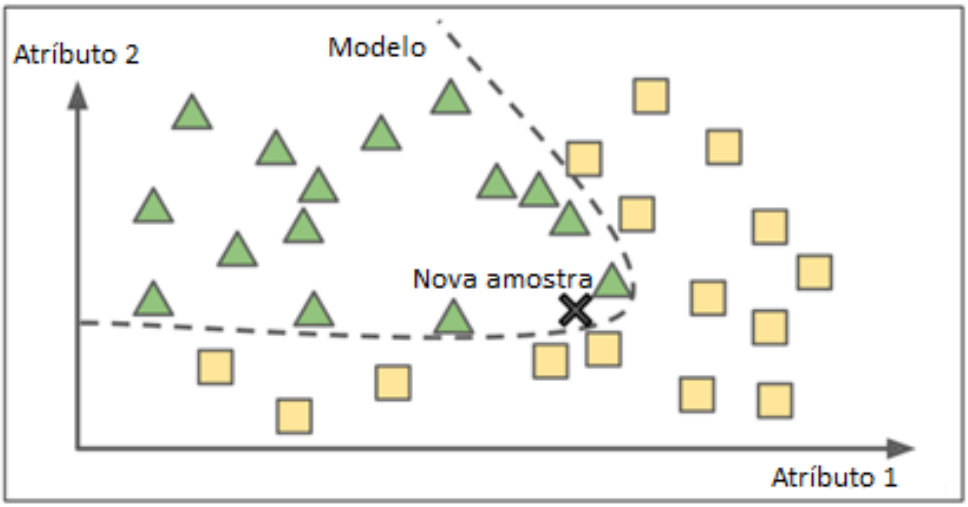
Com o aprendizado baseado em modelo, a diferença está em que as suposições sobre o problema podem ser explicitas através de uma equação matemática, de forma a representar o comportamento do sistema, como as regressões, como mostrado na

Figura 4.

Figura 3 – Aprendizado baseado em instância

Fonte: Adaptado de (GÉRON, 2017)

Figura 4 – Aprendizado baseado em modelo



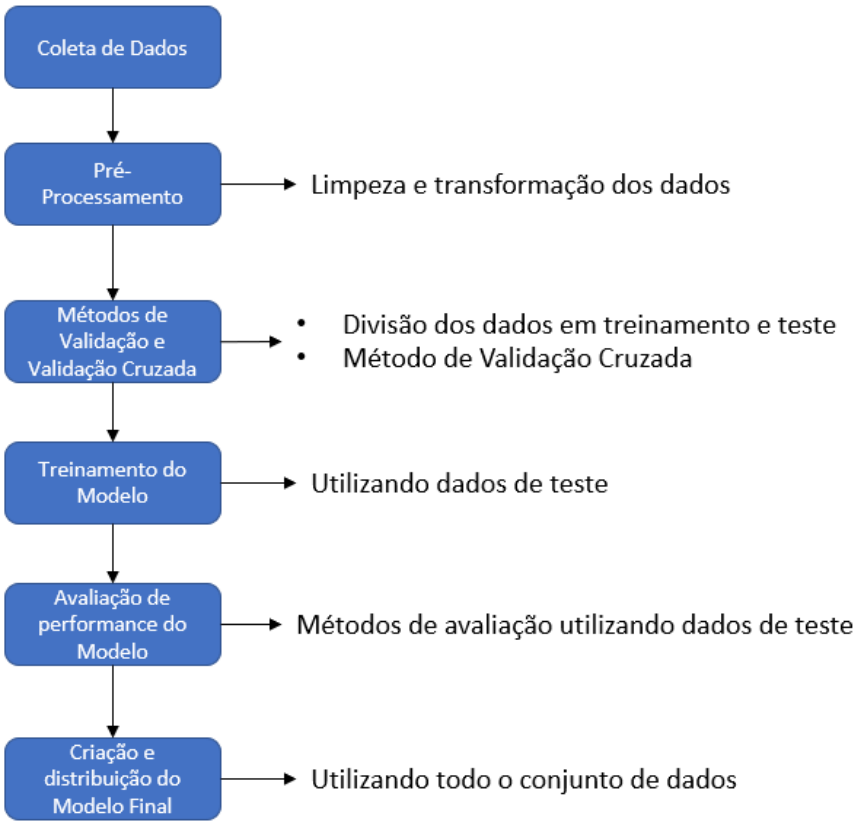
Fonte: Adaptado de (GÉRON, 2017)

2.3 Etapas de desenvolvimento de um sistema de Aprendizado de Máquina

O processo para se elaborar um modelo de Aprendizado de Máquina é composto por várias etapas como coleta de dados, pré-processamento, métodos de validação cruzada, treinamento e avaliação do modelo. Estas diferentes etapas são mostradas no fluxograma da Figura 5 (ALVIM, 2019).

Todas as etapas mostradas na Figura 5 são relevantes e importantes para o modelo atingir uma performance satisfatória. Nas próximas seções serão detalhadas cada uma das etapas do processo para desenvolvimento de um modelo de Aprendizado de Máquina.

Figura 5 – Fluxograma representando as etapas de desenvolvimento de um sistema de aprendizado supervisionado



Fonte: Autor

2.3.1 Coleta de dados

O primeiro passo no desenvolvimento de um sistema de AM é a coleta de dados. Mesmo parecendo ser uma etapa simples, ela impacta drasticamente todos as futuras etapas de desenvolvimento, e, obviamente, sua performance final.

Como os algoritmos de Aprendizado de Máquina são muito dependentes de dados, é necessário um grande volume de observações para que se atinjam uma performance aceitável. Também deve-se considerar os dados enviesados, pois podem alterar drasticamente a performance e a capacidade de generalização do sistema (ALVIM,2019).

Os dados podem ser de diferentes formas, estruturados ou não, numéricos (preço, distância, temperatura, etc.), categóricos (cor, gênero, marca, etc.) e até mesmo textos (comentários em um site, receitas médicas). Eles podem ser coletados de diferentes formas, como através de sensores em uma indústria, *websites*, dados de plantio, dados inseridos em sistemas por usuários, entre outros (DATAROBOT, s.d.).

2.3.2 Pré-processamento de dados

Como dito anteriormente, os algoritmos de Aprendizado de Máquina aprendem através dos dados, por causa disso é essencial que os dados fornecidos sejam de qualidade. Muitos esforços foram feitos para analisar dados, criar ferramentas de análises, mas ignoraram o fato dos problemas que existem com dados reais. Por esse motivo, ferramentas de análise comerciais ou de pesquisa devem prover ferramentas de pré-processamento de dados para serem utilizadas antes e durante o processo de análise, justamente para facilitar esse processo devido ao grande volume de dados necessário para a criação de um modelo (FAMILI, 1997).

Os algoritmos possuem diferentes particularidades, assim o pré-processamento trata esses dados de forma a atender os requisitos mínimos para o bom funcionamento de cada tipo de algoritmo, resultando em um sistema com maior robustez e performance.

2.3.2.1 Limpeza de dados

Conjuntos de dados contendo valores nulos podem ocorrer com certa frequência no mundo real, seja por defeitos nos equipamentos, erros humanos em inserção de dados, dados corrompidos, problemas de armazenamento, etc. Porém, é de grande importância que sejam tratados esses dados, já que alguns algoritmos não suportam valores nulos de entrada, mas existem algumas técnicas para lidar com esse tipo de problema.

Uma dessas técnicas se trata de remover os valores nulos, que, como o próprio nome diz, remover as amostras ou atributos que possuem valores faltantes. Pode ser vista como uma técnica simples, mas contém o risco de eliminar informações valiosas para o algoritmo. Recomenda-se utilizá-la quando possui um grande volume de dados ou quando os valores ausentes forem em pequena quantidade (ALVIM, 2019).

Existe também a técnica de imputação de valores, considerada mais segura que a primeira, que consiste em substituir os valores ausentes de um atributo (coluna) pela média, moda ou mediana dos valores que existem nesta mesma coluna. No caso de colunas categóricas, utiliza-se a moda da classe que mais aparece (ALVIM ,2019).

2.3.2.2 Técnicas e métodos de transformação dos dados

Grande parte das técnicas de Aprendizado de Máquina são sensíveis à escala de

seus dados (AKSOY; HARALICK, 2001). Por esse motivo é essencial executar o escalonamento de atributos. Esse método funciona limitando o intervalo dos atributos (colunas), dessa forma eles podem ser comparados na mesma base, evitando que o atributo com o maior intervalo domine os resultados e ocasione previsões imprecisas. Método utilizado em variáveis contínuas.

Existem várias maneiras para se fazer o escalonamento de atributos, algumas delas são (RASCHKA, 2014):

- Padronização ou Normalização por escore-Z.

  - Essa técnica resulta em todos os atributos possuírem as propriedades de uma distribuição normal padrão, com média $\mu = 0$ e desvio padrão $\sigma = 1$. O escore-z das amostras é calculado de acordo com a Equação 1:

$$z = \frac{x - \mu}{\delta} \tag{1}$$

Sendo:

x = valor antigo;

μ = média de todos os valores da coluna;

δ = desvio padrão da coluna.

- Escalonamento Mínimo – Máximo.

  - Esse escalonamento consiste em converter os valores para um intervalo definido, normalmente entre [0,1]. Este tipo de intervalo limitado, ao contrário da padronização, é que se obtém valores menores de desvios-padrão, suprimindo o efeito de outliers. Sendo aplicado seguindo a Equação 2:

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{2}$$

Sendo:

x = valor antigo;

x_{min} = valor mínimo da coluna;

x_{max} = valor máximo da coluna.

A transformação de atributos categóricos também se faz parte no projeto de criação de um sistema de AM (BISHOP *et al.*, 1977). Grande parte dos algoritmos de Aprendizado de Máquina não manipulam bem atributos categóricos, assim é necessário realizar as conversões para atributos numéricos.

Os dados categóricos podem ser classificados de duas maneiras, sendo elas (VERMA, 2021):

- Dados nominais os dados não têm nenhum tipo de valor numérico. Um exemplo seria os departamentos de uma empresa (como recursos humanos, financeiro, jurídico);
- Dados ordinais: as variáveis possuem um valor que se refere a uma escala ou ordem, como por exemplo o nível de satisfação de cliente após uma compra pode ser classificado como: muito satisfeito, satisfeito, pouco satisfeito, insatisfeito.

Devido aos diferentes tipos de dados categóricos, é utilizado, geralmente, duas abordagens, a Codificação por Rótulo (*Label Encoding*) e a Codificação *One-Hot* (*One-Hot Encoding*).

O método Codificação por Rótulo é utilizado quando os dados são ordinais, convertendo cada rótulo dessa categoria em um valor inteiro, de forma a representar a sequência dos rótulos. A Tabela 1 demonstra como fica a conversão em uma tabela contendo dados sobre o nível de satisfação dos clientes, após realizarem compras em uma loja.

Tabela 1 – Dados sobre vendas com uma coluna categórica ordinal (Nível de Satisfação)

ID_Compra	Cliente	Nível de Satisfação
1	Bruno	Muito satisfeito
2	Carla	Insatisfeito
3	Marcelo	Satisfeito
4	Juliana	Pouco satisfeito
5	Henrique	Satisfeito
6	Breno	Insatisfeito
7	Mateus	Muito satisfeito

Fonte: Autor

Após o método de conversão ser realizado, a coluna Nível de Satisfação é convertida para a forma abaixo, exemplificado na Tabela 2, indicando uma escala presente neste atributo.

O método Codificação *One-Hot* converte cada rótulo presente em um atributo em um novo atributo e atribui a ele um valor binário (0 ou 1). Este tipo de conversão é utilizado quando os dados categóricos são do tipo nominal. Estes novos atributos são conhecidos por rótulos fictícios (*dummy variables*) e sua quantidade depende da quantidade de rótulos presentes no atributo original. A Tabela 3 demonstra como ficaria o atributo de Departamento em uma base de dados referente a uma empresa.

Tabela 2 – Dados sobre vendas com o atributo Nível de Satisfação convertido através do método Codificação por Rótulo

ID_Compra	Cliente	Nível de Satisfação
1	Bruno	3
2	Carla	0
3	Marcelo	2
4	Juliana	1
5	Henrique	2
6	Breno	0
7	Mateus	3

Fonte: Autor

Tabela 3 – Dados sobre quantidade de lucro por departamento de uma empresa

Ano	Lucro	Departamento
2020	198999	Carros de Passeio
2020	78909	Motocicletas
2020	267890	Carros Esportivos
2020	34510	Caminhonetes

Fonte: Autor

Aplicando o Codificador *One-Hot* a Tabela 4 representa como os dados desta empresa seriam convertidos e ficariam da seguinte forma.

Tabela 4 – Dados sobre quantidade de lucro da empresa após aplicação de Codificação *One-Hot* no atributo Departamento

Ano	Lucro	Dep_carros_de_passeio	Dep_motocicletas	Dep_carros_esportivos	Dep_caminhonetes
2020	198999	1	0	0	0
2020	78909	0	1	0	0
2020	267890	0	0	1	0
2020	34510	0	0	0	1

Fonte: Autor

2.3.3 Conjuntos de dados e métodos de validação

Os conceitos de divisão dos dados em treino e teste e a validação do modelo são importantíssimos no conceito de ciência de dados, principalmente no que tange a minimizar os efeitos de *overfitting*. Normalmente, é treinado o modelo com dados de treino para fazer predições em dados que não foram usados no treinamento (dados de validação e de teste) (BRONSHTEIN, 2017).

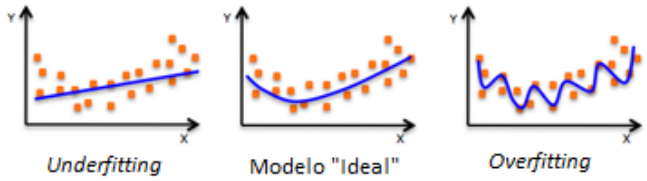
Quando é realizado o treinamento e a predição com o conjunto de dados, pode ocorrer dois tipos de problemas: *overfitting* e *underfitting*. Não é desejável que isso aconteça pois afeta a previsibilidade do modelo criado, assim se tornaria um modelo com baixa acurácia e/ou não generalizado.

2.3.3.1 *Overfitting* e *underfitting*

Overfitting significa quando o modelo foi treinado "bem demais", e agora ele se ajusta muito bem aos dados de treino. Acontece normalmente quando os dados são muito complexos (quando muitos atributos influenciam em uma previsão que se é desejável obter). O modelo terá alta acurácia nos dados de treino, mas não terá uma boa acurácia nos dados de teste ou em dados novos. Isso se dá ao fato de o modelo não estar generalizado, significa que é possível generalizar os resultados, mas não consegue fazer inferências em novos dados. Basicamente, o modelo aprende os ruídos presentes nos dados de treino ao invés do relacionamento entre os atributos (BRONSHTEIN, 2017).

Em contraste ao que ocorre no *overfitting*, quando um modelo possui problemas de *underfitting*, significa que o modelo não se ajusta aos dados de treinamento, e com isso não aprende as tendências e relacionamentos presentes nos dados de treino. O modelo também não consegue ser generalista o suficiente com novos dados, resultado assim em um modelo muito simples. Pode-se observar os 3 casos na Figura 6.

Figura 6 – Exemplo de modelo com *underfitting*, um modelo "ideal" e um modelo com *overfitting*



Fonte: Adaptado de (BRONSHTEIN, 2017)

Importante ressaltar que o *underffiting* não é tão prevalente quanto o *overfitting*. De qualquer forma, é importante evitar este tipo de problema nos modelos criados. O ideal é sempre buscar o meio termo entre o *underfitting* e o *overfitting*. A divisão dos dados entre treino/teste e validação cruzada (*cross validation*) ajudam a evitar estes problemas (BRONSHTEIN, 2017).

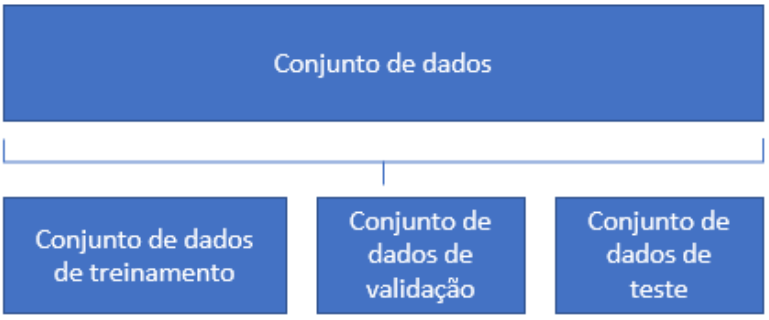
2.3.3.2 Conjuntos de treino, validação e teste

Na construção de modelos de Aprendizado de Máquina é essencial avaliar sua performance, baseado em uma ou mais métricas estatísticas. Além disso, importante ressaltar que os testes para se obterem as métricas devem ser feitos em dados novos, que não foram utilizados no processo de treinamento do modelo, podendo assim observar e mensurar com maior confiança o comportamento desse modelo com dados

novos (KOHAVI, 2001). Os métodos de reamostragem e de validação cruzada servem para cumprir essa função.

Obtendo-se dados suficientes para a obtenção de um modelo, é necessário dividir os dados obtidos em três diferentes partes, sendo elas: dados de treino, dados de validação e dados de teste. A Figura 7 representa essa divisão.

Figura 7 – Divisão dos dados em conjuntos



Fonte: Autor

O conjunto de dados de treino é utilizado para o treinamento do modelo. Aqui é onde acontece os ajustes dos parâmetros do modelo. No caso do algoritmo de Floresta Aleatória, utiliza-se os dados de treino para obter o melhor valor de profundidade máxima (*maximum depth*) do algoritmo.

Após ajustar o modelo, é possível calcular o erro de treinamento através da Equação 3.

$$Erro\ de\ Treinamento = Valor\ Verdadeiro - Valor\ Ajustado \tag{3}$$

Todavia, não se deve utilizar essa métrica para avaliar a performance do modelo. Ao testar um modelo com dados de treino, não se pode obter sua performance em um ambiente real, já que não se sabe sua performance com dados nunca vistos anteriormente. Por esse motivo, utiliza-se os conjuntos de dados de validação e de teste.

O conjunto de dados de validação possui o objetivo de ajustar os parâmetros que foram obtidos pelo conjunto de dados de treino. Normalmente, realiza-se o ajuste em vários modelos e depois verifica-se o erro de predição no conjunto de validação. Após o final desse processo, é escolhido o modelo com o menor erro obtido. Deve se atentar que é comum encontrar modelos com *overfitting* para o conjunto de validação. Assim, o conjunto de dados de teste é utilizado para estimar o erro de predição do modelo que foi selecionado para se ter certeza de que o modelo não possui *overfitting*.

Assim, o conjunto de dados de teste tem por função avaliar o melhor modelo que foi selecionado nas etapas anteriores. Com a métrica calculada, em caso de ser

suficiente para a aplicação a ser utilizado, o modelo é enviado para ser utilizado em produção.

Regularmente a proporção em que se divide os conjuntos de dados segue da seguinte maneira:

- 70% dos dados são designados ao conjunto de treino;
- 20% designados para o conjunto de validação;
- 10% dos dados designados ao conjunto de teste.

Essa divisão pode variar, como por exemplo: 60/20/20, 70/15/15 etc.

2.3.3.3 Validação Cruzada

A Validação Cruzada (VC) é um método utilizado para avaliar e comparar algoritmos de aprendizado de máquinas em que divide o conjunto de dados em duas partes: uma utilizada para aprender e treinar um modelo e a outra utilizada para validar o modelo (REFAEILZADEH; TANG; LIU, 2009). Ou seja, são diferentes métodos para segmentar dados em conjuntos de treino, validação e teste, assim como foi demonstrado na seção anterior. Como são diferentes técnicas, algumas das mais utilizadas serão descritas abaixo.

O método *Hold-Out* é uma técnica simples de VC, nele o conjunto de dados é separado em dois conjuntos, o de treinamento e o de teste. O modelo é treinado no conjunto de treino e sua performance avaliada no conjunto de teste, de forma a medir sua performance em dados distintos aos que foram utilizados em seu treinamento.

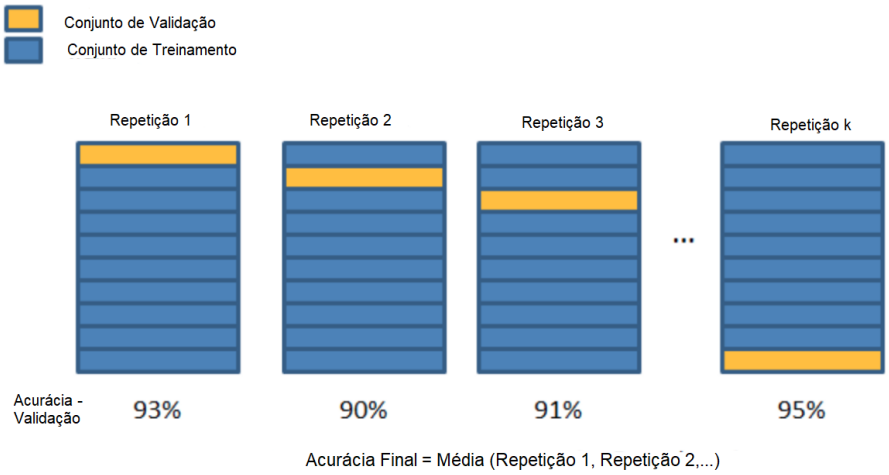
A desvantagem desse método seria que ele não utilizada todos os dados disponíveis, e como os resultados são dependentes dos dados escolhidos na divisão entre treino e teste, os resultados podem variar significativamente dependendo de como a divisão foi realizada.

O método de VC *k-fold* divide o conjunto de dados em *k* subconjuntos, após isso o método de validação é repetido *k* vezes. A cada repetição, um dos *k* subconjuntos é utilizada como o conjunto de teste e os outros *k*-1 são utilizados no treinamento do modelo. Após isso, o erro médio em todas as repetições é calculado. A Figura 8 exemplifica esse método.

Uma outra técnica, similar a anterior é a Validação Cruzada *k-fold* com Repetição, em que basicamente é repetida várias vezes a técnica *k-fold*, e em cada repetição os subconjuntos são divididos de forma diferente. Após cada repetição a métrica é calculada, e posteriormente sumarizada, geralmente utilizando a média, a

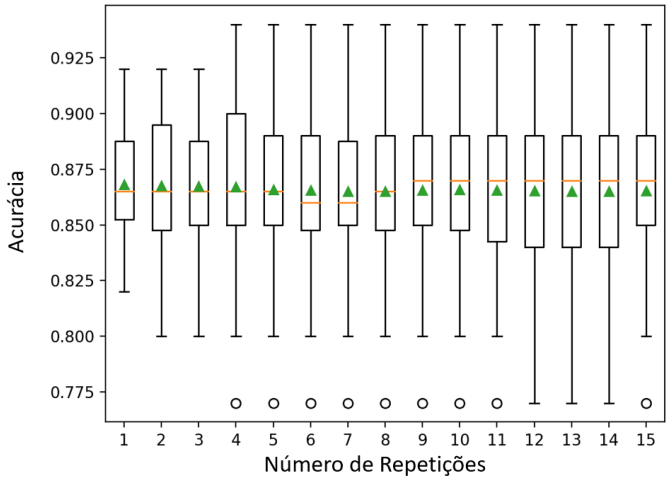
performance de todas as repetições para obter-se uma avaliação final do modelo. Essa técnica reduz o ruído e o viés presente na VC *k-fold*, já que a cada vez que a *k-fold* é realizada, a performance é diferente dependendo de como foi feita a divisão dos subconjuntos. Na Figura 9 tem-se um exemplo de valores de acurácia dependendo da quantidade de repetições que foram utilizadas, o triângulo verde representa a média aritmética da acurácia, a linha laranja a mediana e o eixo horizontal a quantidade de repetições realizas de uma VC *k-fold*.

Figura 8 – Representação do método de Validação Cruzada *k-fold*



Fonte: Adaptado de (BRONSHTEIN, 2017)

Figura 9 – Box Plot representando a acurácia de classificação de um modelo x repetições de Validação Cruzada *k-fold*



Fonte: Adaptado de (BROWNLEE, 2020)

Escolher a quantidade de repetições também é importante. Neste exemplo, escolher 5 repetições para medir a performance do modelo aparenta ser uma boa escolha, já que como a média e a mediana estão coincidindo, isso sugere uma distribuição simétrica razoável e a média pode capturar de forma satisfatória a tendência central (BROWNLEE, 2020).

2.4 Algoritmos de classificação

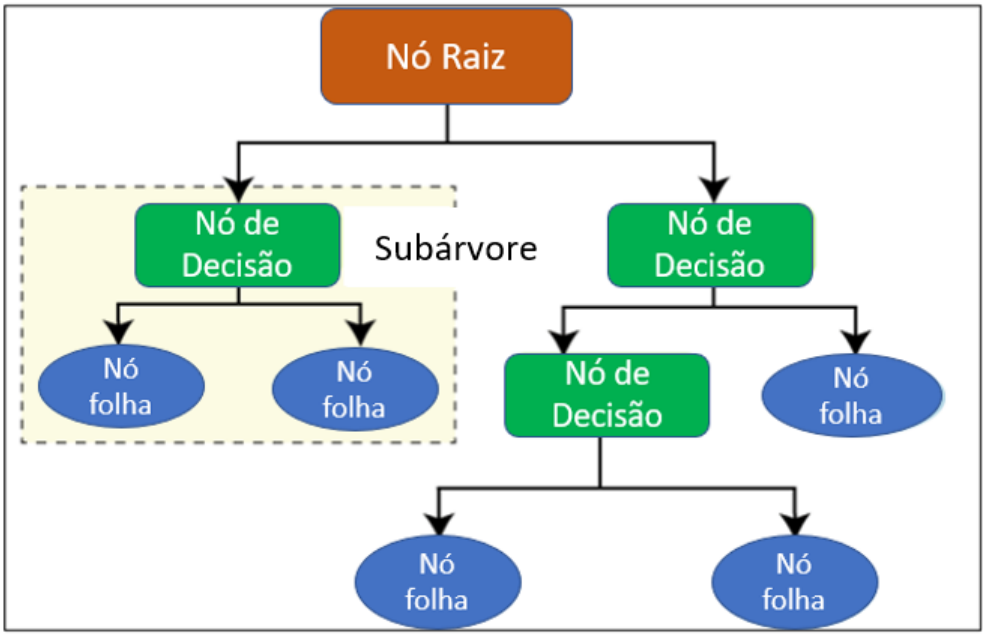
Vários tipos de algoritmos são utilizados na área de Aprendizado de Máquina, como citamos nas seções anteriores, nesta seção serão abordados os principais algoritmos de classificação, que são os que estão presentes no escopo deste trabalho.

2.4.1 Árvore de Decisão

A Árvore de Decisão (*Decision Tree*) é uma das mais poderosas técnica de AM supervisionado, utilizada comumente como um modelo de classificação (JIJO; ABDULAZEEZ, 2021).

Este algoritmo utiliza de uma estrutura de árvores, representando diferentes caminhos, através de ramos, partindo da primeira decisão (nó raiz), em diferentes caminhos subjacentes, chegando em novas decisões (nós de decisão), até chegar na classificação/resposta prevista pelo algoritmo (nós folha). Cada subseção da árvore principal, denominamos de subárvore. A Figura 10 exemplifica essa estrutura.

Figura 10 – Estrutura de uma Árvore de Decisão



Fonte: Adaptado de (JIJO; ABDULAZEEZ, 2021)

Cada folha é atribuída a uma classe representando o valor mais apropriado, assim como, alternativamente, a folha pode representar um vetor probabilidade, que indica a probabilidade de um atributo possuir um certo valor (ROKACH; MAIMOM, 2005). A Figura 11 descreve uma Árvore de Decisão que classifica animais de acordo com suas características.

Figura 11 – Exemplo de uma árvore de decisões representando a classificação de animais

Fonte: Adaptado de (GUIDO; MULLER, 2016)

No exemplo acima, o nó raiz representa uma pergunta, questionando se o animal apresenta penas, nas ramificações estão as possíveis respostas, sendo verdadeiro ou falso, criando assim subárvores com os nós decisão adjacentes (se podem voar ou se possuem barbatanas). No fim observa-se os nós folha, em que é possível ver a resposta final do algoritmo.

2.4.1.1 Algoritmo CART

O Algoritmo CART pode ser utilizado tanto em regressão quando em classificação. Ele basicamente divide o conjunto de treino em dois subconjuntos utilizando um atributo k e um limitante t_k . O algoritmo procura o par (k, t_k)que produzem os subconjuntos mais puros (GÉRON, 2017). A Equação 4 representa o custo que o algoritmo tenta minimizar.

$$J(k, t_k) = \frac{m_{left}}{m} G_{left} + \frac{m_{right}}{m} G_{right} \quad (4)$$

Sendo:

G_{left}/G_{right} mede a impureza do lado esquerdo/direito do subconjunto;

m_{left}/m_{right} é o número de instancia do lado esquerdo/direito do subconjunto.

Após dividir o Nó Raiz em dois, o algoritmo segue dividindo os nós subjacentes usando a mesma lógica e assim por diante, recursivamente. O algoritmo para de dividir quando alcança a profundidade máxima (hiper parâmetro), ou caso não se encontre uma divisão que reduzirá a impureza.

O Coeficiente de Gini se trata de uma medida de não homogeneidade. É uma das medidas mais utilizadas em Árvores de Decisão em classificação. A Equação 5 demonstra o Coeficiente de Gini.

$$GiniIndex = \sum_{ipi}(1 - p_i) \tag{5}$$

A variável p_i é a probabilidade do atributo i e o intervalo de Gini é $[0, 0.5]$. Para um problema de duas classes, a impureza de Gini é dada através da Equação 6.

$$p_1(1 - p_1) + p_2(1 - p_2) \tag{6}$$

Nota-se facilmente que quando a divisão gera um conjunto puro, uma das probabilidades é 0 e o Coeficiente de Gini se torna o menor possível. Do outro lado, quando $p_1 = p_1 = 0.5$, o Coeficiente de Gini se torna o maior possível, com isso, a pureza do Nó é a menor possível (LIN, M.; MING, L, 2021).

O uso do algoritmo Árvore de Decisão possui diversas vantagens, entre elas se destacam (ROKACH; MAIMOM, 2005):

- Elas são autoexplicativas, fáceis de acompanhar e compreender, mesmo para pessoas não técnicas na área;
- Suportam tanto atributos numéricos quanto categóricos;
- A representação da Árvore de Decisão é rica suficiente para representar qualquer classificador de valor discreto;
- Capaz de manipular conjunto de dados que possuem erros e valores ausentes;
- É considera um método não paramétrico, não assumindo suposições sobre distribuição espacial e a estrutura do classificador.

Por outro lado, o algoritmo também apresenta algumas desvantagens:

- Devido ao processo de divisões, o algoritmo apresenta melhor performance se poucos atributos foram relevantes para a previsão. Não tendo uma boa performance caso os atributos possuem iterações complexas entre si;
- Sensível a ruídos e a atributos irrelevantes;
- Com o aumento de dados, a complexidade da Árvore de Decisão tende a aumentar.

2.4.2 Floresta Aleatória

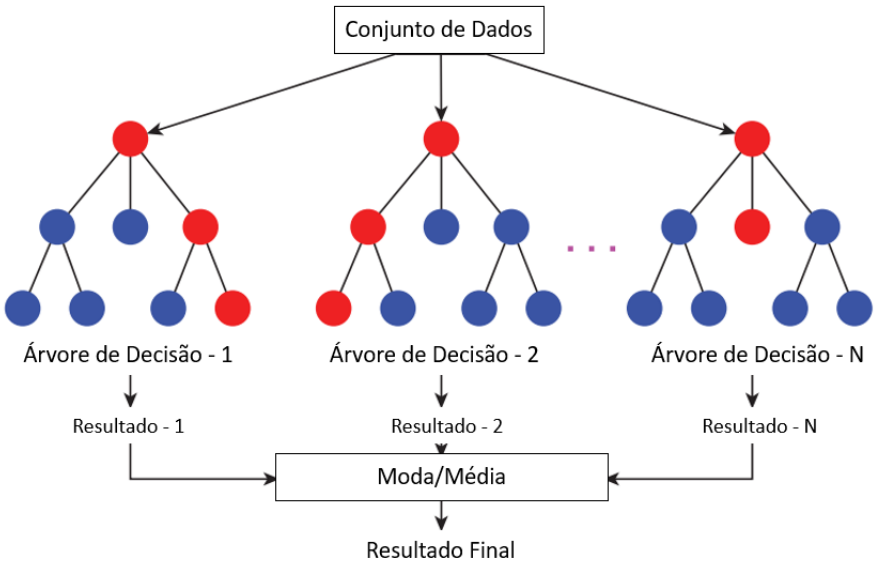
Floresta Aleatória (*Random Forest*) é um algoritmo derivado da Árvore de Decisão, sendo um algoritmo do time ensemble (algoritmos que combinam várias técnicas de AM para obter um melhor desempenho) podendo sendo ser usado tantos em tarefas de regressão quanto de classificação (GÉRON, 2017).

O Algoritmo Árvore de Decisão possui diversas vantagens, como fácil compreensão mesmo para não técnicos da área, suportam atributos numéricos e categóricos, previsões e classificações rápidas (HO, 1995). Entretanto, podem ocorrer *overfitting* dos dados de treino, diminuindo sua performance em dados desconhecidos. De forma a reduzir esse tipo de problema, a Floresta Aleatória é utilizada.

O algoritmo Floresta Aleatória funciona criando diversas Árvores de Decisão, sendo essas treinadas em amostras distintas do conjunto de dados. A classificação final é dada pela moda de todas as classes obtidas (no caso de classificação), ou utiliza a média das previsões calculadas (no caso de regressão).

A Figura 12 exemplifica como é feita a classificação, ou previsão final.

Figura 12 –Representação de uma Floresta Aleatória



Fonte: Adaptado de (KHAN *et al.*, 2021)

Os principais benefícios presentes no algoritmo de Floresta Aleatória são (GÉRON, 2017):

- Redução do risco de *overfitting*;
- Produz flexibilidade por lidar com tarefas de classificação e regressão;
- Facilidade em determinar a importância de um atributo.

Também possui algumas desvantagens listadas a seguir:

- Necessita de um maior tempo de processamento, devido a grande quantidade de dados;
- A complexidade do modelo aumenta, já que se trata de diversas Árvores de Decisão.

2.4.3 K-Vizinhos mais próximos

O algoritmo K-Vizinhos mais próximos (*K-Nearest Neighbors*, ou KNN) é uma técnica de AM aplicada tanto para problemas de regressão quanto de classificação. Este algoritmo é considerado bem simples comparado aos outros de Aprendizado de Máquina, pois não possui premissas matemáticas (COOMANS; MASSART, 1982). Como o foco deste trabalho é em algoritmos de classificação, os exemplos e explicações abaixo serão focadas neste tipo de algoritmo.

O algoritmo KNN funciona memorizando o conjunto de dados de treino. Após isso, ele prevê a classe de qualquer novo dado inserido baseado na classe de seus vizinhos mais próximos, de acordo com o que foi aprendido no treinamento. O método se baseia na suposição de que os atributos utilizados para descrever os dados são relevantes para sua classificação, assim, os novos dados que são semelhantes a esses dados possuem a mesma classe (ALTMAN, 1982).

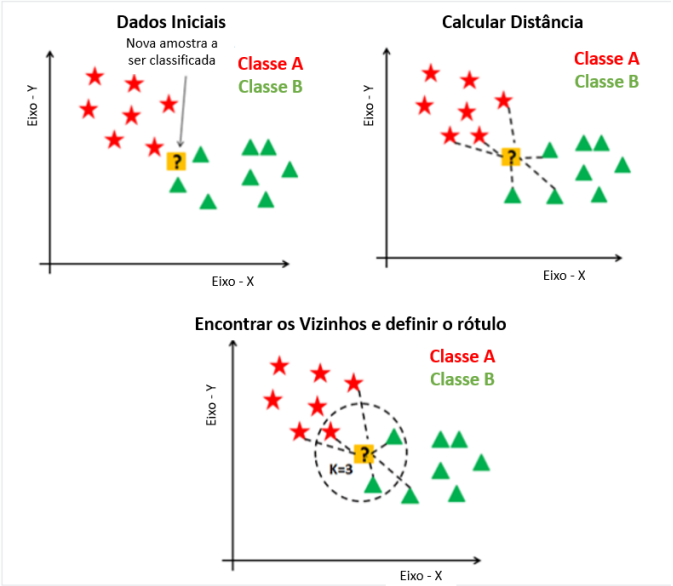
Quando os novos dados são carregados no modelo, é feito o cálculo das distâncias entre os dados, de forma a se determinar a classe desses novos dados, baseados nos dados que possuem as menores distâncias entre ele. Através da Figura 13, podemos observar o funcionamento, em que um novo dado é inserido no modelo para ser classificado, então é calculada as distâncias das observações, baseada no tipo de distância e na quantidade (k) de vizinhos, e por fim, sua classe é determinada pela média em caso de regressão, e pela moda em caso de classificação.

Para que se chegue aos vizinhos mais próximos, podemos utilizar vários cálculos distância, por exemplo: Hamming, Manhattan, Minkowski e Euclidiana. A Euclidiana acaba sendo a mais utilizada, e ela é definida pela Equação 7.

$$Dist(X_1 X_2) = \sqrt{\sum_{i=1}^{k} (X_{1i} - X_{2i})^2} \tag{7}$$

Onde X_1 e X_2 são vetores de atributos.

Figura 13 – Exemplo do funcionamento de um algoritmo KNN



Fonte: Adaptado de (NALANI, 2018)

Um parâmetro importante para o modelo é justamente o valor de k, que deve ser escolhido no momento de construção do modelo. Pesquisas demonstram que não existe um valor otimizado que gera uma boa performance em todo tipo de conjunto de dados, assim cada conjunto de dados possui seu valor otimizado (NALANI, 2018).

Em caso de se utilizar um valor pequeno, o ruído presente nos dados pode influenciar na classificação do modelo, no caso de um valor grande pode necessitar de um grande processamento computacional. O que também pode vir a ocorrer com um valor pequeno de vizinhos é que o modelo terá um ajuste flexível, com pouco viés e grande variação, ocasionando *overfitting*. Em contraponto, com um grande valor, obtém-se um declínio na performance do modelo, e podendo ocasionar *underfitting* (Codeacademy, s.d.).

Geralmente, se escolhe um valor ímpar caso a quantidade de atributos for par. Uma prática bastante comum é gerar diversos modelos com valores distintos de k e comparar as performances obtidas.

As principais vantagens de se utilizar o KNN são:

- Algoritmo intuitivo, simples e fácil de se implementar;
- Pode ser utilizado em dados não-lineares;
- Um parâmetro principal.

As principais desvantagens são:

- Em grande volume de dados necessita de maior processamento computacional, se tornando um algoritmo lento;

- Necessário normalizar os dados, já que dados com magnitude maior influenciam no cálculo da distância dos vizinhos.

2.4.4 *Naive Bayes*

O *Naive Bayes* (NB) é um algoritmo de classificação de que baseia no Teorema de Bayes, desta forma, um teorema probabilístico. O motivo do seu nome possuir *Naive* (inocente) é por causa de ele funcionar supondo que seus atributos são independentes entre si (ZHANG, 2004). Apesar de que na realidade os atributos podem ser dependentes entre si para a classificação.

Pode-se dar como exemplo a classificação de um animal como um peixe ou não. Um animal pode ser considerado um peixe, caso possua nadadeiras, escamas, opérculo e a linha lateral. O classificador NB considera que cada uma dessas características (atributos) contribui para a probabilidade do animal ser um peixe, de forma independente e sem nenhum tipo de correlação entre os atributos.

O Teorema de Bayes demonstra como é possível prever a classe de um dado desconhecido, utilizando como exemplo um treinamento de um dado conhecido (DUDA; HART, 1973). Melhor dizendo, ele auxilia na descoberta de uma probabilidade futura (posterior), dada uma certa condição. O Teorema de Bayes é dado através da Equação 8.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{8}$$

Sendo:

A: Classe original;

B: Atributo (Preditor);

P(A|B): Probabilidade posterior da classe (A) dada preditor (B);

P(B|A): Probabilidade do preditor dada a classe;

P(A): Probabilidade original da classe;

P(B): Probabilidade original do preditor.

O algoritmo *Naive Bayes* possui 3 tipos, de acordo com a distribuição de probabilidade especificada na escolha do algoritmo (TAMAIS, 2019). Os principais são: *Gaussian Naive Bayes*, *Multinominal Naive Bayes* e *Bernoulli Naive Bayes*.

A Tabela 5 exemplifica o funcionamento do modelo.

Tabela 5 – Dados históricos sobre empréstimo

	Income	***Age***	***Loan***	***Default***
0	alta	>18 e <30	alto	1
1	média	>60	alto	0
2	média	>30 e <60	médio	0
3	baixa	>30 e <60	médio	0
4	baixa	>30 e <60	médio	0
5	baixa	>60	baixo	1
6	baixa	>18 e <30	alto	0
7	alta	>18 e <30	médio	1
8	baixa	>18 e <30	baixo	1
9	média	>18 e <30	baixo	1

Fonte: Adaptado de (*TAMAIS, 2019*)

A tabela 6 são os valores de probabilidade de cada atributo ser de um certo valor.

Tabela 6 – Probabilidades dos atributos

Risco de crédito		***Income***			***Age***			***Loan***		
		baixa 5	**média** 2	**alta** 2	**>18 e <30** 5	**>30 e <60** 3	**>60** 2	**baixo** 3	**médio** 4	**alto** 3
0	5/10	3/5	2/5	0	1/5	3/5	1/5	0	1/5	2/5
1	5/10	2/5	1/5	2/5	4/5	0	1/5	3/5	3/5	1/5

Fonte: Adaptado de (*TAMAIS, 2019*)

Assim, é possível testar no caso de um indivíduo tiver baixa renda (*income*), idade (*age*) entre 18 e 30 anos e solicitar um empréstimo (*loan*) alto. Deseja-se classificar seu risco de crédito (*default*), sendo 1 para verdadeiro e 0 para falso). Com isso temos:

$$\text{Probabilidade Parcial } (Risco\ de\ Crédito\ =\ 0\ ) = \left(\frac{5}{10}\right)\cdot\left(\frac{1}{2}\right)\cdot\left(\frac{1}{2}\right)\cdot 1 = \frac{1}{8}$$

$$\text{Probabilidade Parcial } (Risco\ de\ Crédito\ =\ 1\ ) = \left(\frac{5}{10}\right)\cdot\left(\frac{4}{8}\right)\cdot\left(\frac{4}{8}\right)\cdot\frac{1}{8} = \frac{1}{32}$$

$$P(Risco\ de\ Crédito\ = 0)\ =\ \frac{\left(\frac{1}{8}\right)}{\left(\left(\frac{1}{8}\right)+\left(\frac{1}{32}\right)\right)} = 80\%$$

$$P(Risco\ de\ Crédito\ = 1)\ =\ \frac{\left(\frac{1}{32}\right)}{\left(\left(\frac{1}{8}\right)+\left(\frac{1}{32}\right)\right)} = 20\%$$

Os algoritmos do tipo *Naive Bayes* possuem vantagens:

- Por assumir que os atributos são independentes, torna a criação do modelo muito rápida quando comparada a algoritmos mais complexos;
- Funciona muito bem com dados com grandes dimensões, como textos,

mensagens, comentários de sites.

Como principais desvantagens que o algoritmo pode possuir são:

- Ao assumir que os atributos são independentes pode diminuir a acurácia do modelo, mas, nem sempre na realidade isso acontece;
- Necessário aplicar técnicas de suavização, quando acontecer de os dados de teste possuir classificações que não estavam presentes nos dados de treino.

2.4.5 Máquina de Vetores de Suporte

Máquina de Vetores de Suporte (MVS), ou *Support Vector Machine* (SVM), se trata de um dos mais populares modelos de Aprendizado de Máquina supervisionado, podendo ser utilizado para tarefas de classificação lineares e não-lineares, regressão e detecção de outliers nos dados (MITCHELL, 1997). Se trata de um modelo utilizado principalmente em conjuntos de dados de tamanho pequeno ou médio, mas que possuem alto nível de complexidade (CORTES; VAPNIK, 1995).

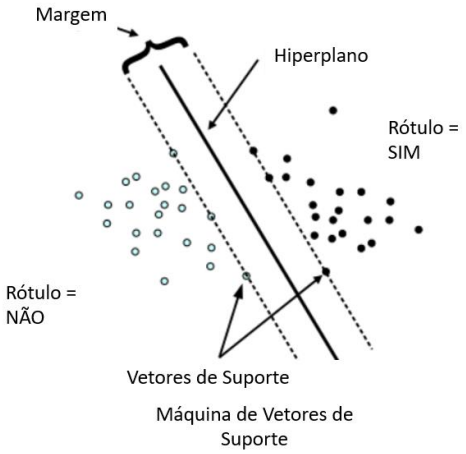
O algoritmo se trata basicamente em distribuir cada dado em um espaço *n*-dimensional, sendo *n* a quantidade de atributos presentes no conjunto de dados a ser utilizado no modelo. Assim, no caso de uma aplicação de classificação, que é o objetivo deste trabalho, é realizada encontrando a linha ou hiperplano que separa da melhor forma as classes no espaço n-dimensional (BOSER; GUYON; VAPNIK, 1992).

Os principais termos, quando se fala de MVS são (BANERJEE, 2020):

- Hiperplano: é um limite de decisão que separa um determinado conjunto de pontos de dados com diferentes rótulos de classe. O classificador separa os pontos de dados usando um hiperplano com a quantidade máxima de margem. Este hiperplano é conhecido como hiperplano de margem máxima e o classificador linear que ele define é conhecido como classificador de margem máxima;
- Vetor de Suporte: são os pontos das amostras do conjunto de dados que estão mais próximos do hiperplano. Eles são responsáveis por definir a melhor linha de separação ou hiperplano através do cálculo das margens;
- Margem: é uma lacuna de separação entre as duas linhas nos pontos de dados mais próximos. É calculada como a distância perpendicular da linha aos vetores de suporte. Nos algoritmos de MVS tentamos maximizar a margem.

A Figura 14 abaixo ilustra os conceitos descritos acima.

Figura 14 – Diagrama exemplificando os conceitos de um modelo SVM



Fonte: Adaptado de (BANERJEE, 2020)

Quando se utiliza MVS, o principal objetivo é buscar o hiperplano com a maior margem possível entre os vetores de suporte, dado um conjunto de dados. Essa busca se dá através de 2 etapas:

- Gerar hiperplanos que segregam as classes na melhor maneira possível. Existem diversos hiperplanos que podem classificar os dados, deve-se atentar ao que representa a maior lacuna, ou margem, entre as duas classes (no caso de uma classificação binária);

- Então, é escolhido este hiperplano com a máxima margem entre os vetores de suporte. Caso esse hiperplano exista, ele é conhecido como hiperplano de margem máxima e o classificador é definido como classificador de margem máxima.

Na prática, os algoritmos de MVS são implementados utilizando um *Kernel*. O *Kernel* se trata de uma função que mapeia os dados para uma dimensão maior, onde os dados são separados. O *Kernel* transforma os dados imputados de baixa dimensão em dados de maior dimensão, dessa forma, é possível converter problemas de separação não-linear em problemas de separação linear ao adicionar maiores dimensões. Além disso, ele auxilia na construção de um modelo de classificação com maior acurácia. Os *Kernels* mais famosos são: Linear, Polinomial, Base Radial e Sigmoide.

É possível utilizar o algoritmo de Máquina de Vetores de Suporte em classificações com mais de duas classes, utilizando o mesmo princípio descrito anteriormente. O problema de classificação com diversas classes é dividido em múltiplos casos de classificação binária, conhecido como abordagem *One-One*. Nesta abordagem, cada classificador separa pontos de duas diferentes classes, e compreendendo todos os classificadores um contra um, resultado em um classificador de múltiplas classes (MARIUS, 2020).

Os algoritmos MVS possuem diversas vantagens, se destacando:

- Mais eficiente em altas dimensões;
- Boa eficiência de memória;
- Boa performance quando tem uma margem de dissociação compreensível entre as classes.

Entre as desvantagens destacam-se:

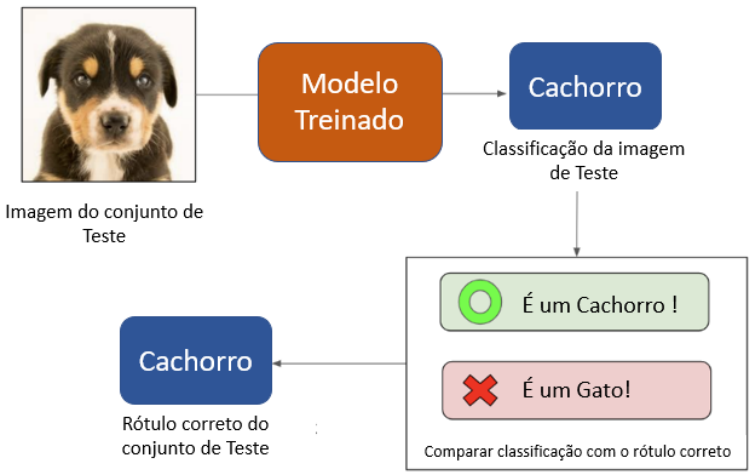
- Não apropriada para grandes conjuntos de dados;
- Sensível a ruídos;
- Não possui uma explicação probabilística para a classificação, já que o algoritmo funciona selecionando pontos acima e abaixo do hiperplano classificador.

2.5 Métricas de performance de modelos de Aprendizado de Máquina

Logo após a criação de um modelo de Aprendizado de Máquina, é essencial mensurar sua performance. Desta forma, é possível utilizar de diferentes métricas avaliativas nos modelos (STEHMAN, 1997). Como este trabalho se trata da criação de modelos de classificação, serão ilustradas as métricas somente relacionadas a este tipo de modelo.

De modo a facilitar a compreensão das métricas, será dado como exemplo um classificador, que classifica se o animal presente na imagem é um cachorro ou um gato. A Figura 15 ilustra este exemplo.

Figura 15 – Exemplo de funcionamento de um classificador de imagens de cachorros



Fonte: Adaptado de (LIU, 2022)

Neste exemplo, o modelo foi treinado através de imagens de cachorros e de gatos com o rótulo presente. Após isso, o modelo faz sua predição e confere se ela está correta ou não. Esse processo ocorre novamente com todas as imagens que se deseja classificar e ao fim é contada a quantidade de predições corretas e incorretas. O problema é que nem todas essas predições corretas e incorretas possuem o mesmo peso na realidade. Dessa forma é definido:

- Verdadeiro Positivo (VP): Significa uma classificação correta, por exemplo uma imagem de cachorro ser classificada como um cachorro;
- Verdadeiro Negativo (VN): Significa uma classificação correta da classe negativa, como uma imagem de gato ser classificada como um gato;
- Falso Positivo (FP): Significa uma classificação errônea da classe positiva (cachorro). Neste caso uma imagem de gato foi classificada como cachorro;
- Falso Negativo (FN): Significa uma classificação errônea da classe negativa (gato). Neste caso uma imagem de cachorro foi classificada como gato;

Através dessas distinções entre as possíveis predições, é calculada a acurácia, sendo essa a medida da frequência com que o classificador realiza uma classificação correta. A acurácia, através da Equação 9, é dada pela razão entre a quantidade de classificações corretas e o número total de classificações realizadas.

$$Acurácia = \frac{Classificações\ Corretas}{Total\ de\ Classificações} \tag{9}$$

Deste modo, é possível chegar na Equação 10.

$$Acurácia = \frac{(VP + VN)}{(VP + VN + FP + FN)} \tag{10}$$

A Acurácia é uma das mais comuns métricas de avaliação de performance de tarefas de classificação, sendo bem utilizada quando a os rótulos das classes estão balanceados, não sendo uma boa escolha quando não possui um balanceamento (LIU, 2022). No caso de possuir 99 imagens de cachorros e somente 1 de gato nos dados de treino, o modelo sempre irá classificar como cachorro e possuir uma Acurácia de 99%. Desta forma, para conseguir avaliar de forma mais completa e abrangente o modelo, é extremamente recomendável utilizar de outras métricas.

É importante levar em conta o peso que tem a quantidade de FP e FN dentro do modelo, já que dependendo do objetivo da classificação, esses dois fatores podem ser de muita importância, como um exame médico classificar um paciente portador de

uma doença como não portador.

A Matriz de Confusão possibilita uma análise com maior riqueza em detalhes a respeito das classificações para cada classe (STEHMAN, 1997). Com essa matriz, observa-se de maneira mais clara quais classes estão sendo classificadas corretamente e incorretamente. A Figura 16 exemplifica a Matriz de Confusão em um caso de classificação binária.

Figura 16 – Exemplo de Matriz de Confusão

		Valor Real	
		Positivo	Negativo
Valor Predito	Positivo	VP	FP
	Negativo	FN	VN

Fonte: Adaptado de (LIU, 2022)

Dessa forma, é possível calcular a Acurácia para cada classe, além da Acurácia global de modo a possuir mais métricas para garantir uma boa performance do modelo.

A Precisão também é uma outra métrica importante, ela corresponde a razão entre os VP e todos os resultados positivos, é possível calcular seu valor através da Equação 11.

$$Precisão = \frac{VP}{(VP + FP)} \tag{11}$$

Utiliza-se a Precisão para medirmos a relação entre as instâncias classificadas como positivas e aquelas que realmente são positivas. Um baixo valor de Precisão indica maior quantidade de predições FP.

A medida de Revocação expressa a proporção de instância positivas que foram classificadas de maneira correta, sua expressão é dada pela Equação 12.

$$Revocação = \frac{VP}{(VP + FN)} \tag{12}$$

Caso o modelo possua um baixo valor de Revocação, maiores as chances de o modelo realizar classificações FN.

Dependendo da aplicação a ser utilizada com o modelo, é desejável maximizar a Precisão ou a Revocação, sempre ao custo da outra métrica. Entretanto, em casos que se deseja maximizar essas duas medidas, utiliza-se a métrica denominada *F1 Score*, demonstrada pela Equação 13, sendo a média harmônica entre a Precisão e a Revocação.

$$F1\ Score = 2\ \times \frac{Precisão\ \times Revocação}{Precisão + Revocação} \tag{13}$$

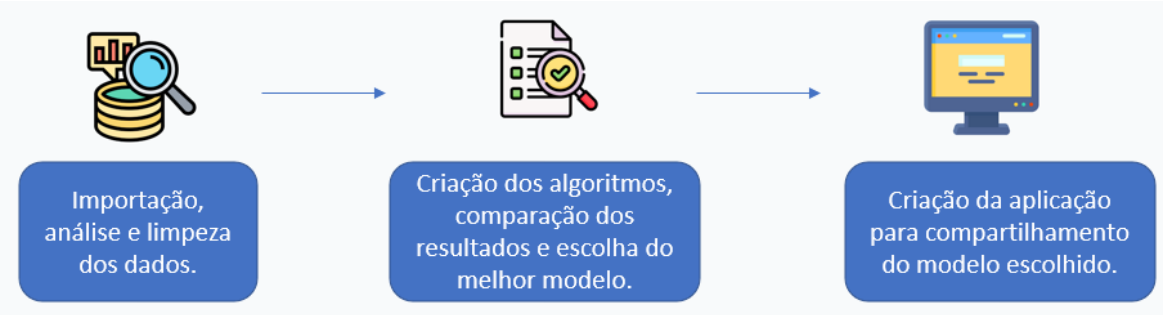
O valor do *F1 Score* varia entre 0 e 1, sendo 1 um modelo de classificação perfeito, e 0 um modelo inoperante. Um bom valor de *F1 Score* indica uma baixa presença de valores FP e FN.

3 Metodologia

Este trabalho visa elaborar um sistema de recomendação de plantio em solo utilizando algoritmos de Aprendizado de Máquina, através de uma base de dados pública com informações sobre plantio. Para isso será desenvolvido um software em linguagem Python, utilizando algoritmos de AM.

Para o desenvolvimento desse projeto, foi realizado sua divisão em etapas, como: o entendimento da base de dados; análise, limpeza e exploração estatística dos dados; seleção dos algoritmos de aprendizado de máquinas; aplicação dos algoritmos e seleção do que possuir o melhor resultado; publicação e compartilhamento do projeto em uma aplicação *web*. Todas essas etapas foram utilizadas bibliotecas do Python, majoritariamente as bibliotecas: *Pandas,* Matplotlib, Seaborn, Plotly, Scikit-Learn e Streamlit. A Figura 17 mostra a representação de todo o esquemático do projeto, contendo suas etapas e ferramentas utilizadas.

Figura 17 – Esquemático do projeto



Fonte: Autor

Como o propósito deste trabalho era de estudo e desenvolvimento de uma ferramenta de Aprendizado de Máquina, optou-se por obter uma base de dados publica com dados agrícolas como estudo de caso.

O conjunto de dados utilizada se trata de uma base de dados acadêmico da Índia, presente na plataforma do Kaggle que coletou dados referentes a plantio neste País. O objetivo é indicar a melhor cultura a ser plantada dentro das características de solo e ambientais. As seguintes culturas (rótulos) são presentes no conjunto de dados: arroz, milho, grão de bico, feijão, feijão fava, feijão Mungo, feijão preto, lentilha, romã, banana, manga, uvas, melancia, melão, maçã, laranja, papaia, coco, algodão, juta e café. Os atributos abaixo se referem as características coletadas de plantio (INGLE, 2020):

- *N* – Proporção do teor de nitrogênio no solo - *kg/ha*;

- *P* – Proporção do teor de fósforo no solo - *kg/ha*;
- *K* – Proporção do teor de potássio no solo - *kg/ha*;
- *temperature* – Temperatura em graus ^{o}C;
- *humidity* – Umidade relativa em %;
- *pH* – pH do solo;
- *rainfall* – pluviosidade em *mm*.

A Tabela 7 demonstra algumas linhas presentes no conjunto de dados.

Tabela 7 – Parte do conjunto de dados de plantio

	N	P	K	temperature	humidity	ph	rainfall	label
0	90	42	43	20.879744	82.002744	6.502985	202.935536	rice
1	85	58	41	21.770462	80.319644	7.038096	226.655537	rice
2	60	55	44	23.004459	82.320763	7.840207	263.964248	rice
3	74	35	40	26.491096	80.158363	6.980401	242.864034	rice
4	78	42	42	20.130175	81.604873	7.628473	262.717340	rice

Fonte: Autor

3.1 Importação e análise exploratória dos dados

Ao iniciar qualquer estudo ou aplicação em Aprendizado de Máquinas é essencial possuir um conjunto de dados e fazer uma análise exploratória dele. A análise é importante para que se observe a qualidade e integridade dos dados, de forma a ver se são suficientes para a criação de um algoritmo. Neste trabalho a linguagem de programação utilizada foi Python, sendo seu código disponível no GitHub (ATHAYDE, 2022).

A escolha pela linguagem Python é devido ao fato de a linguagem ter se tornado uma referência para a maioria das aplicações em ciência de dados, por combinar o poder das linguagens de programação destinadas para propósitos gerais quanto linguagens que são utilizadas em propósitos específicos como R e MATLAB. O Python possui bibliotecas para carregamento de dados, visualização, estatística, processamento de linguagem natural, processamento de imagens e muito mais (GUIDO; MULLER, 2016).

Uma das bibliotecas mais famosas se trata da Pandas, sendo utilizada principalmente para tratamento e análise de dados, sendo construída em volta de uma estrutura de dados denominada DataFrame, uma estrutura tabular semelhante a uma tabela do Microsoft Excel. Entrando em contraste com o NumPy, uma biblioteca em que é necessário que todos os dados sejam do mesmo tipo em uma lista, o Pandas

permite que cada coluna do DataFrame possa ser de diferentes tipos de dados. Além disso é possível com o Pandas importar dados para serem analisados e tratados de diferentes formatos e fontes, seja através de arquivos CSV, consultas SQL, arquivos do Excel, entre outros (GUIDO, S.; MULLER, A., 2016).

A biblioteca Scikit-Learn é um projeto de uso e distribuição gratuita, possuindo diversos algoritmos de Aprendizado de Máquina, assim como uma extensa documentação sobre eles. O projeto Scikit-Learn possui uma comunidade de usuários ativas pois trata de um projeto em constante evolução, sendo utilizado amplamente no ambiente acadêmico e industrial. A estrutura de dados principal do Scikit-Learn se trata de lista de objetos da biblioteca NumPy, sendo este um pacote fundamental para computação cientifica em Python. Com o NumPy é possível criar listas e matrizes de dados e realizar operações matemáticas com eles com os módulos presentes na biblioteca. Quando se trata de visualização de dados e criação de gráficos, a biblioteca Matplotlib é considerada a principal, sendo possível criar histogramas, diagramas de caixa, gráficos de linha, de forma a gerar conclusões e análises através das visualizações criadas (GUIDO, S.; MULLER, A., 2016).

Além da Matplotlib outras duas bibliotecas se destacam na criação de visualizações e gráficos. A Seaborn é uma biblioteca criada baseada na Matplotlib e possui grande interação com ela e as tabelas do Pandas, possuindo diversas aplicações voltadas para gráficos estatísticos (SEABORN, 2022). Outra de grande destaque é a biblioteca Plotly, em que é possível a criação de gráficos e visualizações interativos (PLOTLY, 2022).

A importação é feita utilizando a biblioteca Pandas e a criação dos gráficos utilizando principalmente as bibliotecas Seaborn, Plotly e Matplotlib. Na Tabela 7 foi apresentado um exemplo de como são os dados utilizados neste trabalho, além disso, outros passos essenciais são ver a quantidade de dados nulos presentes, para que sejam realizados remoções ou substituição de valores. A Figura 18 mostra a distribuição de dados não-nulos presentes neste conjunto:

Figura 18 – Dados nulos no conjunto de dados

```
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   N            2200 non-null   int64
 1   P            2200 non-null   int64
 2   K            2200 non-null   int64
 3   temperature  2200 non-null   float64
 4   humidity     2200 non-null   float64
 5   ph           2200 non-null   float64
 6   rainfall     2200 non-null   float64
 7   label        2200 non-null   object
```

Fonte: Autor

Foi possível observar nenhuma presença de dados nulos, posteriormente foi

realizada uma contagem dos rótulos (culturas a serem plantadas) para ver se a distribuição está igualitária, sendo este o cenário ideal. A Tabela 8 representa uma distribuição ideal no conjunto de dados e o índice (*index*) referente a cada rótulo.

Tabela 8 – Distribuição dos rótulos (*labels*)

index	label	count
0	apple	100
1	banana	100
2	blackgram	100
3	chickpea	100
4	coconut	100
5	coffee	100
6	cotton	100
7	grapes	100
8	jute	100
9	kidneybeans	100
10	lentil	100
11	maize	100
12	mango	100
13	mothbeans	100
14	mungbean	100
15	muskmelon	100
16	orange	100
17	papaya	100
18	pigeonpeas	100
19	pomegranate	100
20	rice	100
21	watermelon	100

Fonte: Autor

É importante ver as estatísticas descritivas dos atributos presentes nos conjuntos de dados, principalmente por todos serem numéricos. A Tabela 9 descreve estatisticamente os dados dos atributos do conjunto de dados utilizado.

Tabela 9 – Estatísticas descritivas dos atributos (*features*)

	N	P	K	temperature	humidity	ph	rainfall
count	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000
mean	50.551818	53.362727	48.149091	25.616244	71.481779	6.469480	103.463655
std	36.917334	32.985883	50.647931	5.063749	22.263812	0.773938	54.958389
min	0.000000	5.000000	5.000000	8.825675	14.258040	3.504752	20.211267
25%	21.000000	28.000000	20.000000	22.769375	60.261953	5.971693	64.551686
50%	37.000000	51.000000	32.000000	25.598693	80.473146	6.425045	94.867624
75%	84.250000	68.000000	49.000000	28.561654	89.948771	6.923643	124.267508
max	140.000000	145.000000	205.000000	43.675493	99.981876	9.935091	298.560117

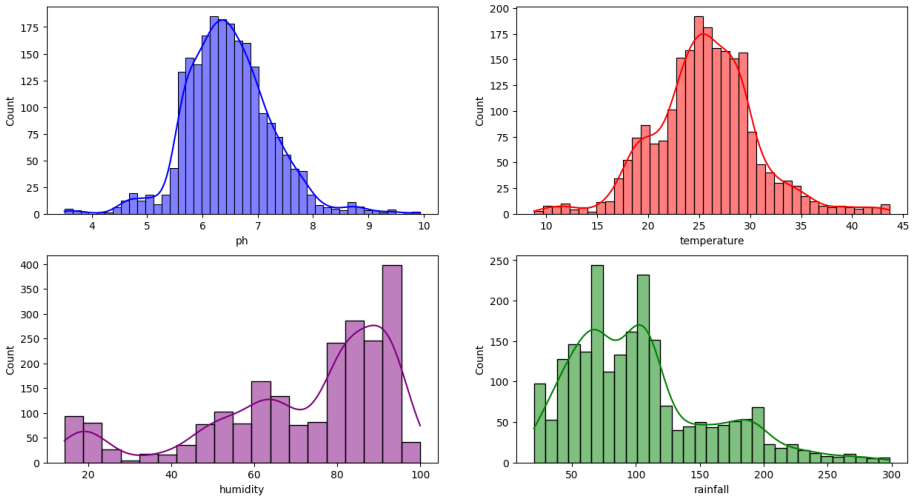
Fonte: Autor

Assim, com essa análise estatística foi possível identificar que os atributos estão em escalas bem distintas entre si, sendo essa questão relevante para algoritmos em que a escala interfere na modelagem.

Foram criados alguns histogramas para que sejam observadas as frequências dos valores dos atributos presentes no conjunto de dados, com foco em ver onde estão

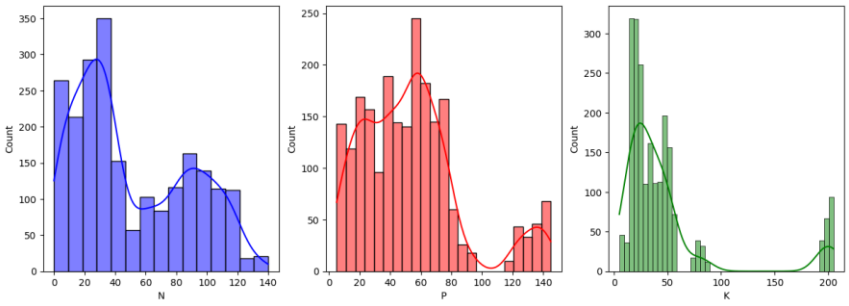
as maiores concentrações dos valores e a simetria deles. A Figura 19 e a Figura 20 representam, respectivamente, a distribuição dos atributos climáticos e o *pH*, e atributos relacionados ao solo e adubagem.

Figura 19 – Histograma dos valores dos atributos climáticos e o *pH*



Fonte: Autor

Figura 20 – Histograma dos valores dos atributos *N*, *P* e *K*



Fonte: Autor

Com os histogramas foi possível ver uma certa uniformidade na distribuição dos valores de *pH* e temperatura, em contrapartida, o restante dos atributos se comporta de maneira não uniforme, se destacando o atributo *K* pela não uniformidade da distribuição dos valores.

Buscando uma abordagem mais especifica em relação a cada cultura, a Tabela 10 representa os valores médios de umidade, temperatura, pluviosidade, *pH*, *N*, *P* e *K* de cada cultura.

Um Diagrama de Caixa também foi criado para demonstrar sua influência na tomada de decisão de cada uma das culturas a serem recomendadas para plantio. A Figura 21 demonstra como os valores de pH estão distribuídos em cada um dos rótulos presentes.

Com os valores do *pH*, distribuídos entre média, mediana e os quartis, foi

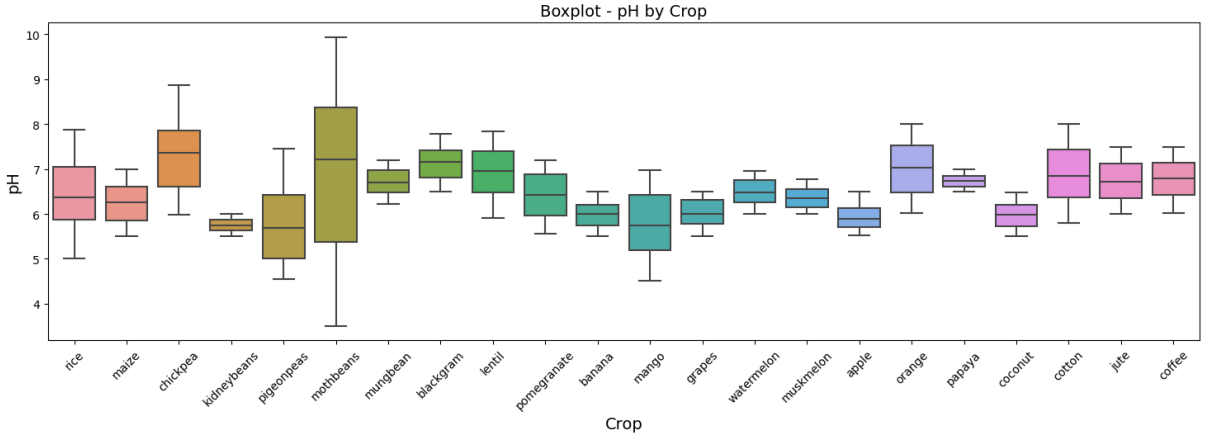
possível ver algumas culturas que precisam estar entre uma pequena faixa de valores para que tenha um bom plantio, enquanto algumas poucas não são tão dependentes deste atributo.

Tabela 10 – Média dos valores dos atributos para cada cultura

	label	K	N	P	humidity	ph	rainfall	temperature
0	apple	199.89	20.80	134.22	92.333383	5.929663	112.654779	22.630942
1	banana	50.05	100.23	82.01	80.358123	5.983893	104.626980	27.376798
2	blackgram	19.24	40.02	67.47	65.118426	7.133952	67.884151	29.973340
3	chickpea	79.92	40.09	67.79	16.860439	7.336957	80.058977	18.872847
4	coconut	30.59	21.98	16.93	94.844272	5.976562	175.686646	27.409892
5	coffee	29.94	101.20	28.74	58.869846	6.790308	158.066295	25.540477
6	cotton	19.56	117.77	46.24	79.843474	6.912675	80.398043	23.988958
7	grapes	200.11	23.18	132.53	81.875228	6.025937	69.611829	23.849575
8	jute	39.99	78.40	46.86	79.639864	6.732778	174.792798	24.958376
9	kidneybeans	20.05	20.75	67.54	21.605357	5.749411	105.919778	20.115085
10	lentil	19.41	18.77	68.36	64.804785	6.927932	45.680454	24.509052
11	maize	19.79	77.76	48.44	65.092249	6.245190	84.766988	22.389204
12	mango	29.92	20.07	27.18	50.156573	5.766373	94.704515	31.208770
13	mothbeans	20.23	21.44	48.01	53.160418	6.831174	51.198487	28.194920
14	mungbean	19.87	20.99	47.28	85.499975	6.723957	48.403601	28.525775
15	muskmelon	50.08	100.32	17.72	92.342802	6.358805	24.689952	28.663066
16	orange	10.01	19.58	16.55	92.170209	7.016957	110.474969	22.765725
17	papaya	50.04	49.88	59.05	92.403388	6.741442	142.627839	33.723859
18	pigeonpeas	20.29	20.73	67.73	48.061633	5.794175	149.457564	27.741762
19	pomegranate	40.21	18.87	18.75	90.125504	6.429172	107.528442	21.837842
20	rice	39.87	79.89	47.58	82.272822	6.425471	236.181114	23.689332
21	watermelon	50.22	99.42	17.00	85.160375	6.495778	50.786219	25.591767

Fonte: Autor

Figura 21 – Diagrama de Caixas com os valores de pH para cada cultura



Fonte: Autor

3.2 Criação e avaliação dos algoritmos de Aprendizado de Máquina

Após a análise exploratória dos dados, antes de se criar qualquer modelo de AM, é necessário identificar a existência de correlação entre os atributos. O principal método se dá através da correlação de Pearson, sendo está uma medida que

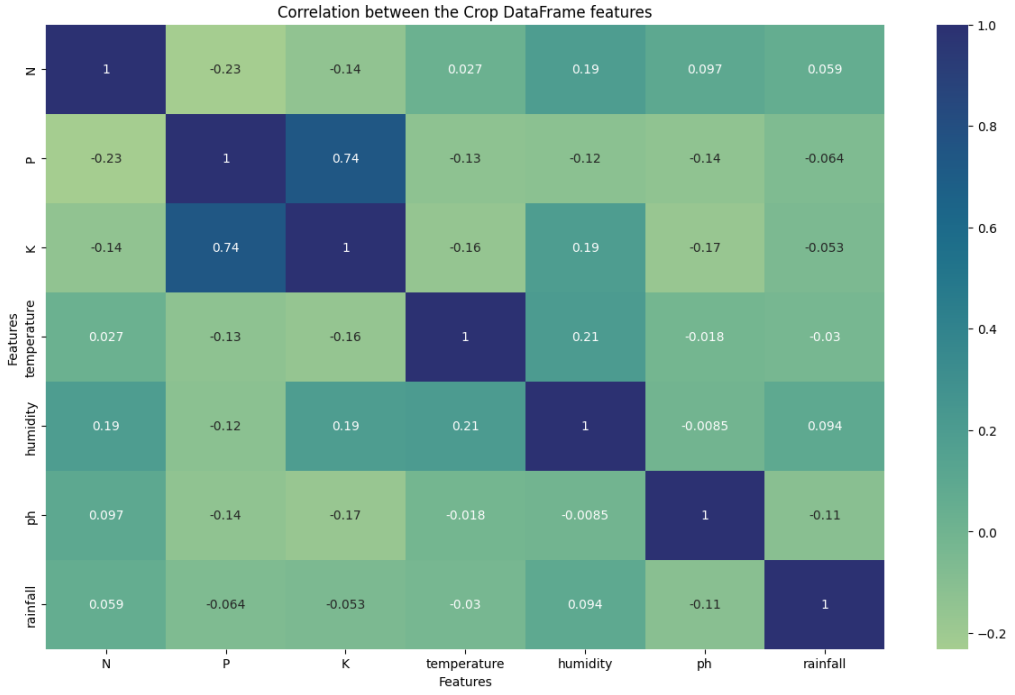
representação um relacionamento linear entre duas variáveis (SAEED, 2022). Considerando duas variáveis, sendo essas X e Y. Matematicamente considerando σXY a covariância entre X e Y, e σX o desvio padrão de X, então obtém-se o coeficiente da correlação de Pearson ρ , que é dado pela Equação 14.

$$\rho X, Y = \frac{\sigma XY}{\sigma X \sigma Y} \tag{14}$$

A correlação é de grande interessante de ser observada entre os atributos de um conjunto de dados, por auxiliar no entendimento de como eles se comportam, se é possível reduzir a quantidade de atributos ao possuir dois com alta correlação, de forma a diminuir o conjunto de dados, reduzindo também erros e tempo de processamento. Com a biblioteca Pandas é possível criar uma matriz de correlação, e interpretar essa matriz através dos valores de seus coeficientes. O coeficiente de correlação varia entre -1 e 1, quando o valor é próximo de 1, significa que as duas variáveis possuem uma forte correlação linear positiva. Em caso do coeficiente se aproximar de -1, significa que as duas variáveis possuem uma forte correlação linear negativa. No caso de se aproximarem de nulo significa que não possuem correlação ou possuem uma correlação extremamente fraca (GÉRON, 2017).

A matriz de correlação do conjunto de dados deste trabalho está mostrada na Figura 22.

Figura 22 – Matriz de correlação dos atributos



Fonte: Autor

Através da Figura 22 foi possível identificar que somente dois atributos

possuem uma correlação linear positiva, sendo esta moderada. Que são os valores de Potássio (*K*) e Fósforo (*P*). Reforçando que correlação não significa causa e consequência, e por não se tratar de uma correlação extremamente forte, foi mantido os dois atributos para a construção dos modelos.

De forma a conseguir avaliar os modelos criados da melhor maneira possível, e sem viés, é necessário que os dados sejam divididos entre dados de treino e de teste. Para isso a biblioteca Scikit-Learn possui um módulo que faz essa divisão de maneira randômica do nosso conjunto de dados. Assim ficou dividido da seguinte forma:

- 70% dos dados destinados para treinamento, totalizando 1540 amostras;
- 30% dos dados destinados para teste, totalizando 660 amostras.

Todas os algoritmos utilizados para a criação dos modelos, neste projeto, possuem suporte inerente para casos classificação multiclasse, exceto pelo algoritmo de Máquina de Vetores de Suporte. No caso do MVS, é utilizada uma técnica, pelo próprio módulo do Scikit-Learn em que utiliza de diversas classificações binárias para representar uma classificação multiclasse. Essa estratégia é denominada *one-versus-one* (*OvO*). Assim são criados diferente modelos de classificação, cada um representando uma das possíveis classificações binárias, englobando todos os possíveis pares existentes (GÉRON, 2017).

Para a obtenção dos melhores parâmetros e hiper parâmetros de cada algoritmo para se obter os modelos, a biblioteca Scikit-Learn também possui um módulo denominado GridSearchCV. Através dele é possível testar a criação de um modelo com uma lista de diferentes valores de parâmetros e hiper parâmetros e o módulo cria vários modelos com todas as possíveis combinações de valores e calcula sua acurácia. A acurácia é calculada utilizando a separação dos dados de treinamento novamente em treinamento e teste através da técnica Validação Cruzada *k-fold*. Assim, o GridSearchCV retorna qual combinação de parâmetros e hiper parâmetros possui a melhor performance e quais são esses valores para que se crie o melhor modelo possível para cada algoritmo.

Para o algoritmo de Árvore de Decisão foi utilizado os seguintes valores no GridSearchCV:

- Critério: Gini, Entropia;
- Profundidade Máxima: 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22.

Nesse caso, os parâmetros com a melhor performance foram Gini e a Profundidade Máxima em 12.

Para o algoritmo de Floresta Aleatória foi utilizado os seguintes valores no

GridSearchCV:

- Critério: Gini, Entropia;
- Número de Estimadores: 50, 100, 150, 200;
- Profundidade Máxima: 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22.

Nesse caso, os parâmetros com a melhor performance foram Gini, Profundidade Máxima em 12 e o Número de Estimadores em 50.

Para o algoritmo de K-Vizinhos Mais Próximos foi utilizado diversos valores de *k*, sendo *k* o parâmetro de número de vizinhos. Assim a Tabela 11 mostra os valores de *k* e os valores de acurácia dos modelos.

Tabela 11 – Valores do parâmetro *k* e performance de cada modelo

	k	Train Score	Test Score
0	4	0.979221	0.974242
1	5	0.982468	0.975758
2	6	0.976623	0.974242
3	7	0.981818	0.971212
4	8	0.976623	0.972727
5	9	0.975974	0.975758
6	10	0.970779	0.972727
7	11	0.973377	0.969697
8	12	0.968831	0.965152
9	13	0.967532	0.965152
10	14	0.966234	0.956061
11	15	0.965584	0.960606
12	16	0.962987	0.956061
13	17	0.964286	0.954545
14	18	0.962338	0.948485
15	19	0.959091	0.948485

Fonte: Autor

No caso do algoritmo K-Vizinhos Mais Próximos o valor escolhido foi $k = 9$. Por possuir a melhor acurácia, apesar de $k = 5$ possuir a mesma acurácia, é preferível escolher um valor maior para evitar casos de *overfitting*.

Para o algoritmo Máquina de Vetores de Suporte foi utilizado os seguintes valores no GridSearchCV:

- C: 0.1, 1, 100, 1000;
- *Kernel*: rbf*, poly, sigmoid, linear*;

- Grau: 1, 2, 3, 4, 5, 6;
- Gamma: 0.0001, 0.001, 0.01, 0.1, 1.

Para o algoritmo de MVS, o GridSearchCV retornou como os valores de parâmetros C em 100, o *Kernel* como rbf (Função de Base Radial), o Grau em 1 e o Gamma em 0.01.

No caso do modelo *Naive-Bayes* já foi utilizado o algoritmo referente a distribuição Gaussiana, chamado de *Gaussian Naive-Bayes*, que é utilizado quando os atributos são valores contínuos e presume que eles seguem uma distribuição Gaussiana (normal). Não possuindo assim parâmetros para serem utilizados.

Após o GridSearchCV nos retornar os melhores parâmetros para serem utilizados, os modelos de cada tipo foram criados utilizando esses valores, conseguindo assim comparar suas performances através das métricas disponíveis utilizando os dados de teste para essas avaliações.

Na seção seguinte, Resultados e Discussões, serão demonstradas as métricas de performance de cada modelo, e o porquê da escolha do modelo de Floresta Aleatória foi utilizado na aplicação *web* utilizando a biblioteca Streamlit.

3.3 Criação e distribuição da aplicação *web*

Com o modelo selecionado, no Python é possível transformá-lo em um arquivo, através do módulo Pickle, de forma a poder utilizá-lo em outros arquivos, aplicativos e ser compartilhado de uma maneira simples, rápida e fácil. Dessa forma não possui a necessidade de treinar o modelo novamente.

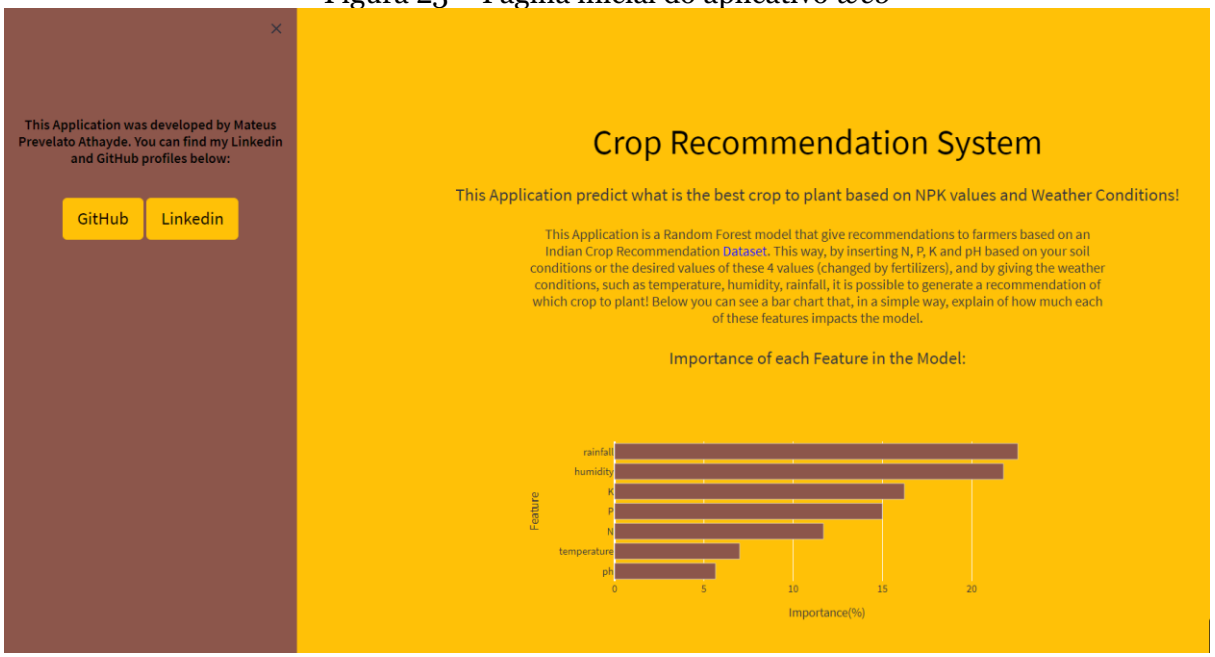
O Streamlit é uma biblioteca de código aberto que possibilita criar e compartilhar de maneira fácil e simples aplicações *web* customizáveis focadas principalmente em ciência de dados e Aprendizado de Máquina, podendo ser feito puramente em Python sem necessidade de outras linguagens de programação (STREAMLIT, 2022). Com o Streamlit também é possível compartilhar o aplicativo na nuvem de forma grátis. O Streamlit Cloud possibilita conectar sua conta do GitHub, onde possui o código fonte do projeto, e compartilhá-lo na internet. Outro benefício é que o aplicativo é atualizado caso for realizada qualquer alteração no código.

O intuito de utilizar o Streamlit neste trabalho foi a possibilidade de se poder criar o sistema de recomendação de plantio em um aplicativo *web*. Assim, qualquer pessoa, mesmo que não possua conhecimentos técnicos em programação, poderia utilizar o modelo de aprendizado de máquinas e gerar sua recomendação quando e como quiser com uma interface de fácil entendimento e intuitiva.

Com o modelo em mãos, a próxima etapa, sendo está a final, é de criação do aplicativo *web*, para que se possa ser feito o compartilhamento e uso do sistema de recomendação de plantio.

Utilizando a biblioteca Streamlit, é possível criar uma janela lateral retrátil, que foi utilizada nesse projeto para ter os botões que redirecionam para o contato do Linkedin do autor deste projeto e o repositório no GitHub para acessar o código fonte do aplicativo e da criação dos modelos. A Figura 23 mostra o design da janela lateral do aplicativo e como ele é apresentado ao ser aberto.

Figura 23 – Página inicial do aplicativo *web*



Fonte: Autor

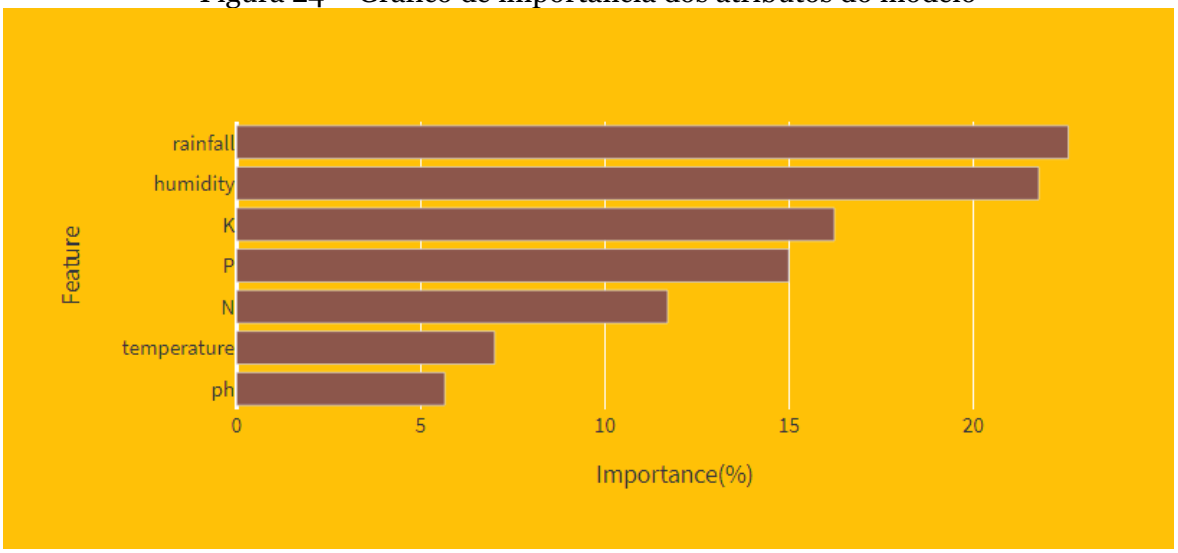
Pela Figura 23, é possível ver, além da janela lateral, um pequeno texto que descreve a funcionalidade do aplicativo e o *link* com os dados utilizados para a criação do modelo. Logo de cara também é possível observar um gráfico referente a importância de cada um dos atributos na escolha da melhor cultura a ser plantada.

O cálculo realizado para se calcular a importância de cada atributo no modelo é calculando a média da profundidade em que cada atributo aparece nas Árvores de Decisão presentes dentro da Floresta Aleatória. Assim, um atributo importante irá aparecer logo nos primeiros nós das Árvores de Decisão, enquanto os menos importantes irão, normalmente, ao fim, próximo as folhas das Árvores (GÉRON, 2017). A Figura 24 representa a importância de cada um dos atributos presentes no modelo.

Um pouco mais abaixo, através da Figura 25, podemos observar as caixas de entrada de texto. Cada uma das entradas do algoritmo possui, no canto superior direito, um campo de ajuda, que ao passar o cursor do *mouse* por cima, aparece um

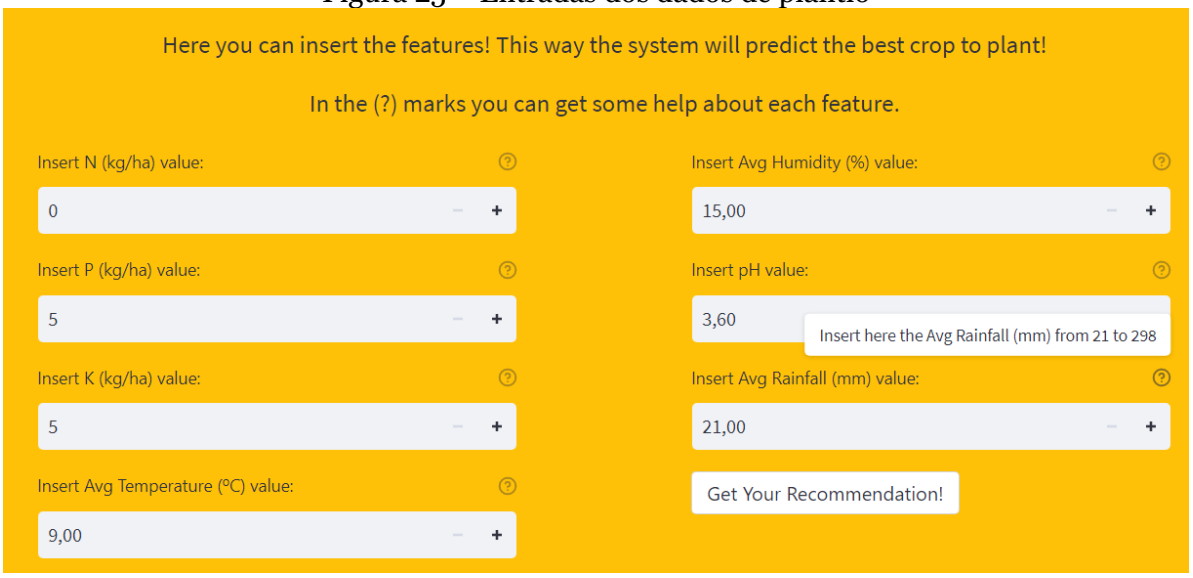
pequeno texto explicando os valores possíveis a serem inseridos naquele atributo em específico. No canto inferior direito, tem um botão que ao ser pressionado gera a recomendação de plantio.

Figura 24 – Gráfico de importância dos atributos do modelo



Fonte: Autor

Figura 25 – Entradas dos dados de plantio

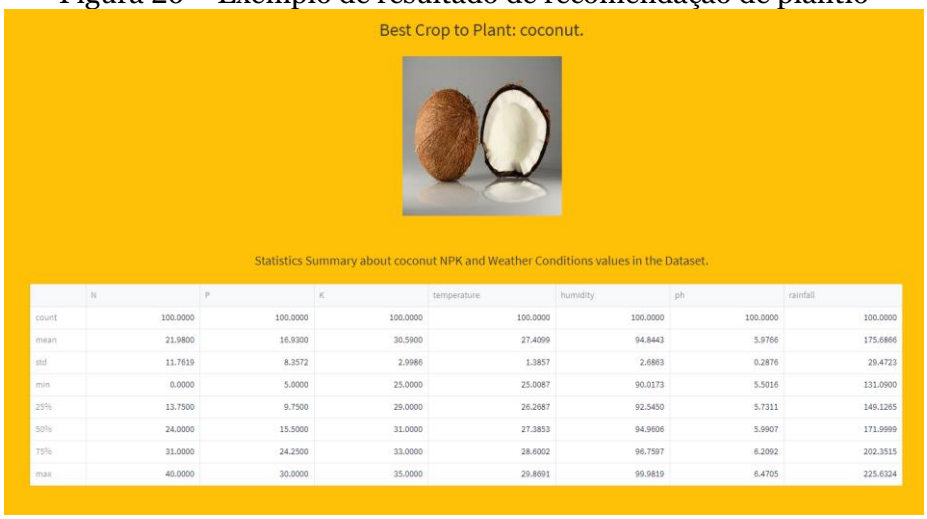


Fonte: Autor

Após pressionar o botão, a recomendação é gerada. Assim, é mostrado na tela o nome da cultura a ser plantada, uma imagem dela e um quadro com uma descrição estatística de cada atributo relacionado a aquela cultura, de acordo com os dados presentes no conjunto de dados.

No exemplo da Figura 26, foi recomendado o plantio de coco, e logo abaixo os valores médios, desvios padrão, mínimos, quantis e máximos para cada atributo em que foi plantado coco.

Figura 26 – Exemplo de resultado de recomendação de plantio



	N	P	K	temperature	humidity	ph	rainfall
count	100.0000	100.0000	100.0000	100.0000	100.0000	100.0000	100.0000
mean	21.9800	16.9300	30.5900	27.4099	94.8443	5.9766	175.6866
std	11.7619	8.3572	2.9986	1.3857	2.6863	0.2876	29.4723
min	0.0000	5.0000	25.0000	25.0087	90.0173	5.5016	131.0900
25%	13.7500	9.7500	29.0000	26.2687	92.5450	5.7311	149.1265
50%	24.0000	15.5000	31.0000	27.3853	94.9606	5.9907	171.9999
75%	31.0000	24.2500	33.0000	28.6002	96.7597	6.2092	202.3515
max	40.0000	30.0000	35.0000	29.8691	99.9819	6.4705	225.6324

Fonte: Autor

4 Resultados e discussões

Neste Capítulo serão apresentados os resultados obtidos nos diferentes algoritmos de AM estudados neste trabalho. Vale ressaltar que foi utilizado uma base de dados pública com informações sobre plantio onde 70% da base foi usada para treinamento dos modelos e 30% para testes de validação. A base de dados possui 2200 amostras de dados de plantio da Índia.

Através da obtenção dos melhores parâmetros e hiper parâmetros para cada algoritmo, os modelos foram criados de modo a atingir a melhor performance possível (acurácia) e comparados entre si. Por fim, é compartilhado para o usuário final, através da aplicação *web*, o melhor modelo que possui as melhores métricas e robustez.

4.1 Performance dos modelos de Aprendizado de Máquina

Após a criação dos modelos, foram utilizados os dados de teste para comparar suas performances. Assim, foi utilizado o modelo para classificar os resultados de acordo com os atributos e foi comparado se são iguais ao rótulo das amostras de teste. As métricas de Precisão, Revocação e *F1 Score* gerais são calculadas através da média ponderada de cada uma dessas métricas a cada classificação/predição feita pelo modelo e compara com o verdadeiro valor do rótulo presente nos dados de teste. Assim obtém-se as seguintes métricas calculadas:

- Acurácia;
- Precisão;
- Revocação;
- *F1 Score*.

Alguns outros valores que são interessantes de serem mencionados são:

- Tempo de compilação do GridSearchCV (quando utilizado);
- Tempo de compilação da criação do modelo.

Como o GridSearchCV faz uma validação cruzada nos dados de treinamento, ele divide estes dados em 5 conjuntos. Dessa forma, ele treina o modelo com 4 conjuntos e usa o quinto para testes. Após isso, ele refaz essa operação alterando qual conjunto será utilizado para teste, e ao fim, faz a média da acurácia de cada modelo obtido com

um conjunto de parâmetros específico. Como foi passada uma lista de parâmetros, esse processo descrito anteriormente é repetido para cada combinação possível de parâmetros. Assim, por exemplo, podemos treinar um modelo de Árvore de Decisão com os possíveis parâmetros de:

- Critério = Gini ou Entropia;
- Profundidade Máxima = 4 ou 10.

Neste exemplo, serão realizados a criação de 20 modelos, sendo 5 modelos para cada combinação possível de parâmetro. No fim, a combinação que obter a melhor média de acurácia é retornada pelo GridSearchCV para ser utilizada na criação do melhor modelo possível.

4.1.1 Performance do modelo Árvore de Decisão

Começando com o modelo Árvore de Decisão, utilizando os parâmetros:

- Critério = Gini;
- Profundidade Máxima = 12.

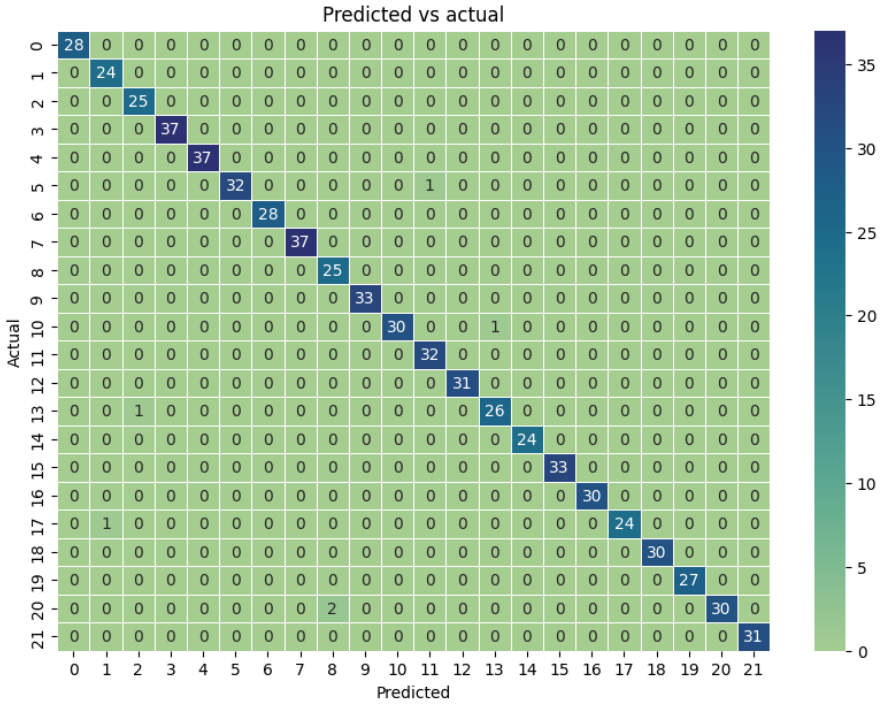
As métricas obtidas foram:

- Acurácia = 99,09%;
- Precisão = 99,13%;
- Revocação = 99,09%;
- *F1 Score* = 99,09%;
- Tempo de compilação do GridSearchCV = 11,8 segundos;
- Tempo de compilação da criação do modelo = 0,8 segundos.

Para facilitar a visualização, as culturas foram substituídas por números (a ordens numérica dos números dos rótulos e a ordem alfabética dos rótulos coincidem com os da Tabela 8) e o interessante é notar os valores que aparecem na diagonal principal, indicando classificações corretas.

Através da Figura 27, que representa a Matriz de Confusão do modelo Árvore de Decisão, nota-se que predominou as classificações corretas, já que a contagem dessas classificações está na diagonal principal, onde a previsão e o rótulo verdadeiro coincidem. Observa-se pouquíssimos casos de classificação incorreta distribuídos na matriz.

Figura 27 – Matriz de confusão do modelo Árvore de Decisão



Fonte: Autor

4.1.2 Performance do modelo Floresta Aleatória

Foi realizada a construção do modelo Floresta Aleatória, utilizando os parâmetros:

- Critério = Gini;
- Profundidade Máxima = 12;
- Número de Estimadores = 50.

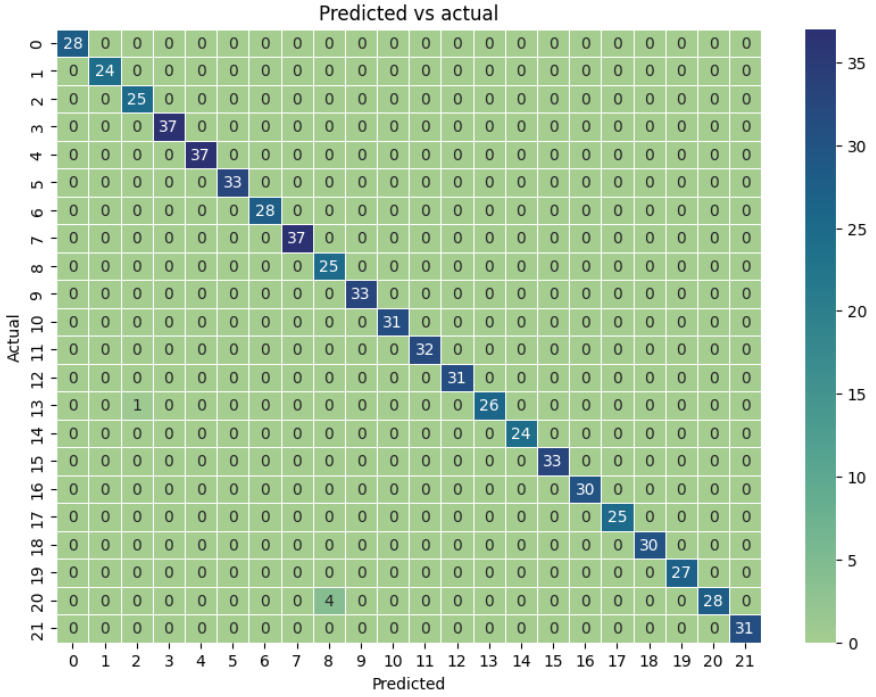
As métricas obtidas foram:

- Acurácia = 99,24%;
- Precisão = 99,33%;
- Revocação = 99,24%;
- *F1 Score* = 99,24%;
- Tempo de compilação do GridSearchCV = 35,9 segundos;
- Tempo de compilação da criação do modelo = 0,2 segundos.

Através da Figura 28 observa-se a Matriz de Confusão do modelo Floresta

Aleatória.

Figura 28 – Matriz de confusão do modelo Floresta Aleatória



Fonte: Autor

Através da Figura 28 observa-se que predominou as classificações corretas, já que a contagem dessas classificações está na diagonal principal, onde a previsão e o rótulo verdadeiro coincidem. Nota-se pouquíssimos casos de classificação incorreta distribuídos na matriz, dessa vez mais concentrado entre os rótulos 8 (juta) e 20 (arroz), em que por 4 vezes foi recomendada a previsão de juta, onde o mais apropriado seria arroz.

4.1.3 Performance do modelo K-Vizinhos mais próximos

Foi realizada a construção do modelo de K-Vizinhos Mais Próximos, utilizando os parâmetros:

- $k = 9$.

As métricas obtidas foram:

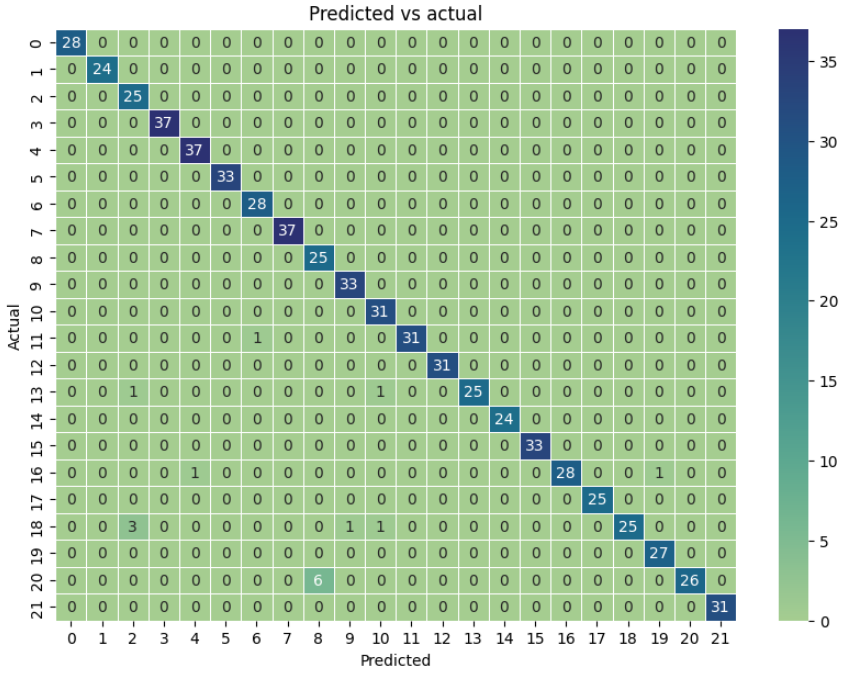
- Acurácia = 97,58%;
- Precisão = 97,87%;
- Revocação = 97,58%;
- *F1 Score* = 97,56%;

- Tempo de compilação do laço para obtenção do melhor valor de k = 1,6 segundos;

- Tempo de compilação da criação do modelo = 0,1 segundos.

Através da Figura 29 pode-se observar a Matriz de Confusão do modelo K-Vizinhos Mais Próximos.

Figura 29 – Matriz de confusão do modelo K-Vizinhos Mais Próximos



Fonte: Autor

Através da Figura 29 observa-se que predominou as classificações corretas, já que a contagem dessas classificações está na diagonal principal, onde a previsão e o rótulo verdadeiro coincidem. Nota-se pouquíssimos casos de classificação incorreta distribuídos na matriz, dessa vez mais concentrado entre os rótulos 8 (juta) e 20 (arroz), em que por 6 vezes foi recomendada a previsão de juta, onde o mais apropriado seria arroz.

4.1.4 Performance do modelo Máquina de Vetores de Suporte

Foi realizada a construção do modelo de Máquina de Vetores de Suporte, utilizando os parâmetros:

- C = 100;

- *Kernel* = rbf (Função de Base Radial);

- Grau = 1;

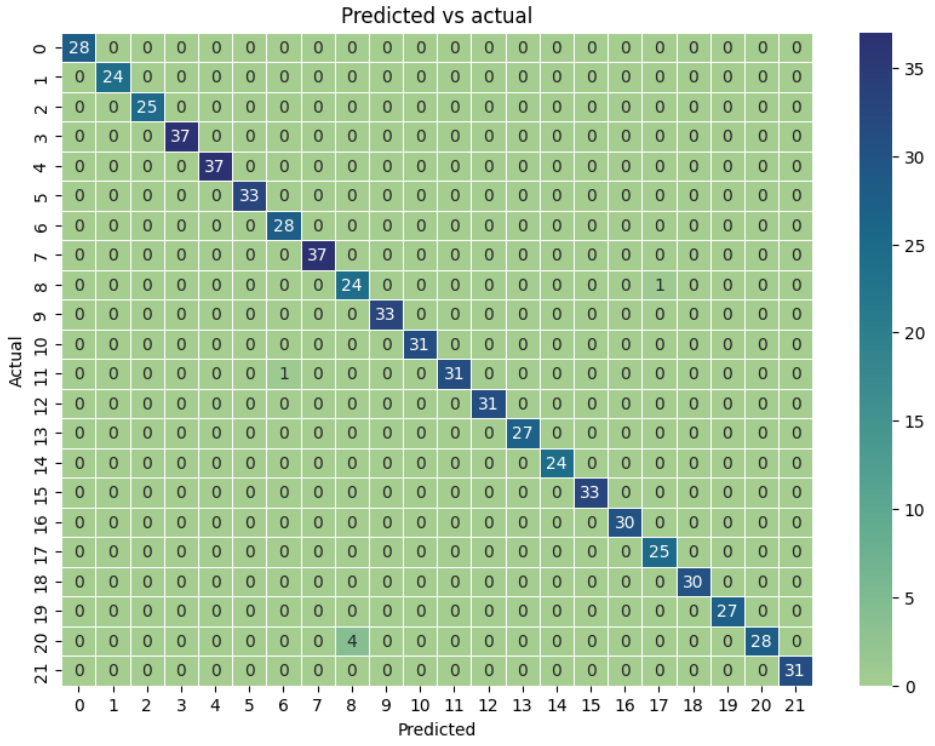
- *Gamma* = 0.01.

As métricas obtidas foram:

- Acurácia = 99,09%;
- Precisão = 99,17%;
- Revocação = 99,09%;
- *F1 Score* = 99,09%;
- Tempo de compilação do GridSearchCV = 30,7 segundos;
- Tempo de compilação da criação do modelo = 0,2 segundos.

Através da Figura 30 pode-se observar a Matriz de Confusão do modelo Máquina de Vetores de Suporte.

Figura 30 – Matriz de confusão do modelo Máquina de Vetores de Suporte



Fonte: Autor

Através da Figura 30, observa-se que predominou as classificações corretas, já que a contagem dessas classificações está na diagonal principal, onde a previsão e o rótulo verdadeiro coincidem. Nota-se pouquíssimos casos de classificação incorreta distribuídos na matriz, dessa vez mais concentrado entre os rótulos 8 (juta) e 20 (arroz), em que por 4 vezes foi recomendada a previsão de juta, onde o mais apropriado seria arroz.

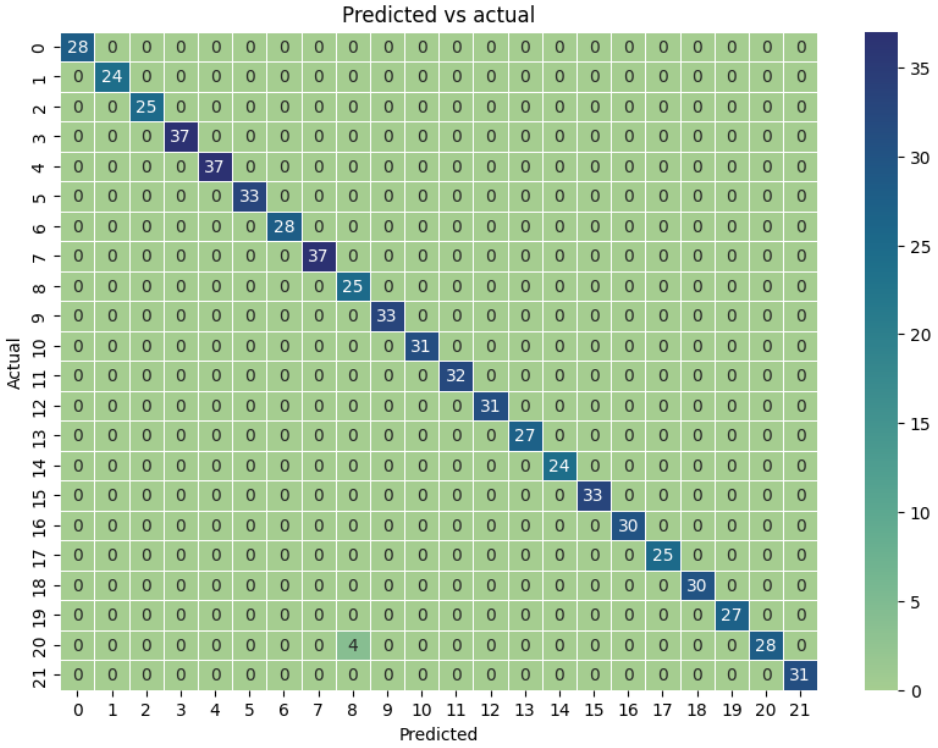
4.1.5 Performance do modelo *Gaussian Naive-Bayes*

Por ser um modelo que não possui parâmetros ou hiper parâmetros, o modelo foi construído e as métricas obtidas foram:

- Acurácia = 99,39%;
- Precisão = 99,48%
- Revocação = 99,39%
- *F1 Score* = 99,40%
- Tempo de compilação da criação do modelo = 0,9 segundos.

Através da Figura 31 pode-se observar a Matriz de Confusão do modelo *Gaussian Naive-Bayes*.

Figura 31 – Matriz de confusão do modelo *Gaussian Naive-Bayes*



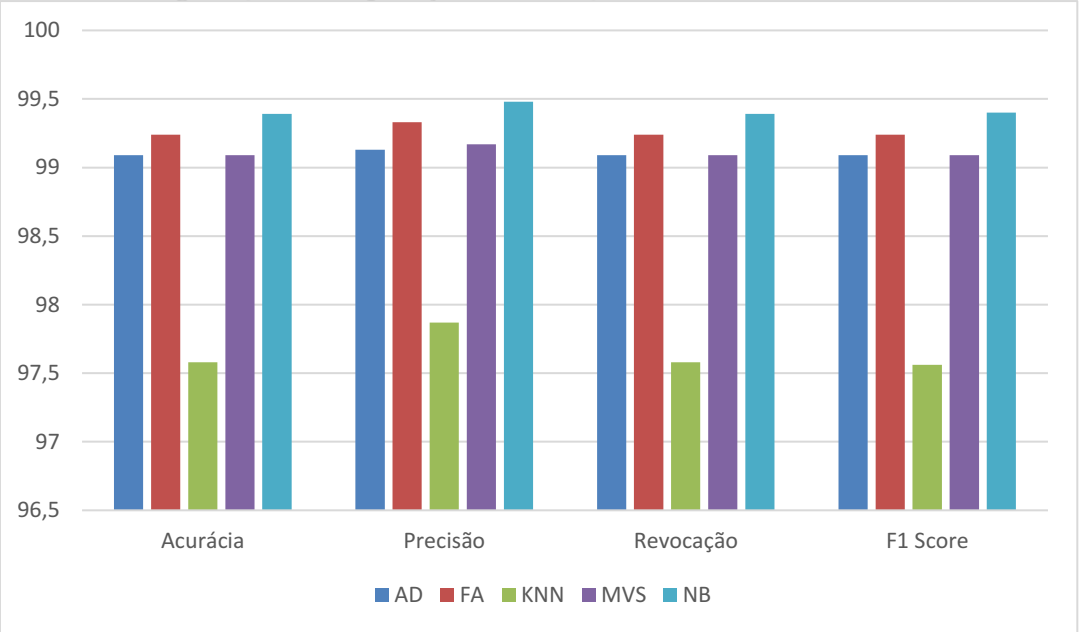
Fonte: Autor

Através da Figura 31 observa-se que predominou as classificações corretas, já que a contagem dessas classificações está na diagonal principal, onde a previsão e o rótulo verdadeiro coincidem. Nota-se pouquíssimos casos de classificação incorreta distribuídos na matriz, dessa vez mais concentrado entre os rótulos 8 (juta) e 20 (arroz), em que por 4 vezes foi recomendada a previsão de juta, onde o mais apropriado seria arroz.

4.1.6 Escolha e exportação do melhor modelo

Após a criação dos modelos foi realizada a comparação entre suas métricas, para escolha do melhor modelo. A Figure 32 mostra o resultado da comparação, onde AD é o modelo Arvore de Decisão, FA é o Floresta Aleatória, KNN é o K-Vizinhos mais Próximos, MVS é o Máquina de Vetores de Suporte e o NB é o *Naive Bayes*.

Figura 32 – Comparação entre os 5 modelos de AM utilizado



Fonte: Autor

De acordo com a Figura 32, dois modelos obtiveram um destaque em relação aos outros, sendo eles: Floresta Aleatória (FA) e *Gaussian Naive-Bayes* (NB).

Apesar do modelo baseado em *Naive-Bayes* possuir as métricas, principalmente o *F1 Score*, ligeiramente maior que o apresentada pelo modelo de Floresta Aleatória, ambos tiveram uma ótima performance em todas as métricas, superando os 99%. A escolha do modelo de Floresta Aleatória se dá pela robustez em relação a *outliers*, baixo risco de *overfitting*, apesar de necessitar de um tempo maior para a criação do modelo. O ponto fraco principal do modelo *Gaussian Naive-Bayes* é ele assumir que os atributos são independentes entre si, dessa forma, em caso de inserção de novas amostras, pode ser que o relacionamento entre os atributos se intensifique e o modelo não interpretaria essas questões.

Assim, com o módulo Pickle do Python foi possível exportar o modelo em formato de arquivo para que possa ser inserido e utilizado dentro do Streamlit sem a necessidade de todas as etapas descrita nesse trabalho para criação do mesmo. É possível também que se altere, com facilidade, o modelo a ser utilizado na aplicação *web*.

4.2 Utilização da aplicação *web*

Com a aplicação criada, tendo o modelo de Floresta Aleatória, uma breve descrição de como funciona e a interface de usuário construída, foi realizado alguns testes de modo a conferir se a aplicação atende ao esperado.

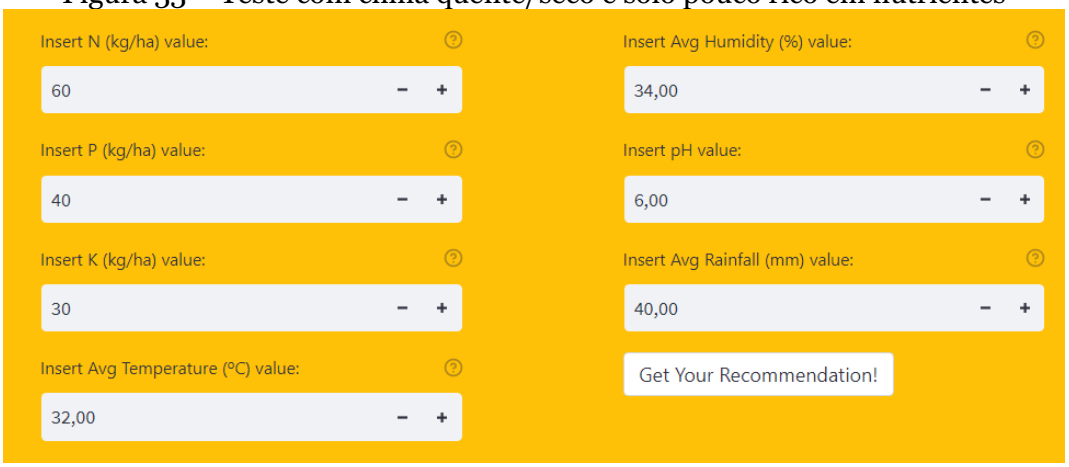
Foram realizados dois testes, cada um considerando características especificas de clima e condições de solo ou adubamento, e após isso a conferência da recomendação: se ela é coerente, se o nome da planta a ser cultivada e sua imagem está clara e se o resumo estatístico dela está de acordo com o que foi inserido pelo usuário.

4.2.1 Teste com clima quente/seco e solo pobre em nutrientes.

Através da Figura 33, observa-se que no primeiro teste foram utilizadas características de clima quente e seco, e características de solo com baixa concentração de nutrientes. Os valores utilizados foram:

- N = 60 kg/ha;
- P = 40 kg/ha;
- K = 30 kg/ha;
- Temperatura = 32°C;
- Umidade = 34%;
- pH = 6;
- Pluviosidade = 40 mm.

Figura 33 – Teste com clima quente/seco e solo pouco rico em nutrientes



Fonte: Autor

Após a inserção dos dados e o acionamento do botão *"Get Your Recommendation!"*, surgem na tela a recomendação, a foto do cultivo recomendado e a tabela com o resumo estatístico dos valores do conjunto de dados referente a aquele cultivo em específico. A Figura 34 representa o resultado deste teste, sendo o resultado o cultivo de manga (*mango*).

Figura 34 – Resultado do primeiro teste

Fonte: Autor

A Figura 35 apresenta o quadro com o resumo estatístico.

Figura 35 – Resumo estatístico do primeiro teste

Statistics Summary about mango NPK and Weather Conditions values in the Dataset.

	N	P	K	temperature	humidity	ph	rainfall
count	100.0000	100.0000	100.0000	100.0000	100.0000	100.0000	100.0000
mean	20.0700	27.1800	29.9200	31.2088	50.1566	5.7664	94.7045
std	12.3290	7.6639	3.0967	2.6539	2.7563	0.7037	3.3386
min	0.0000	15.0000	25.0000	27.0032	45.0224	4.5075	89.2915
25%	9.0000	19.7500	27.0000	28.9125	47.9306	5.1837	91.6127
50%	21.0000	27.5000	30.0000	31.3002	50.2816	5.7434	94.9060
75%	30.2500	35.0000	32.0000	33.3826	52.4828	6.4166	97.4758
max	40.0000	40.0000	35.0000	35.9901	54.9641	6.9674	100.8125

Fonte: Autor

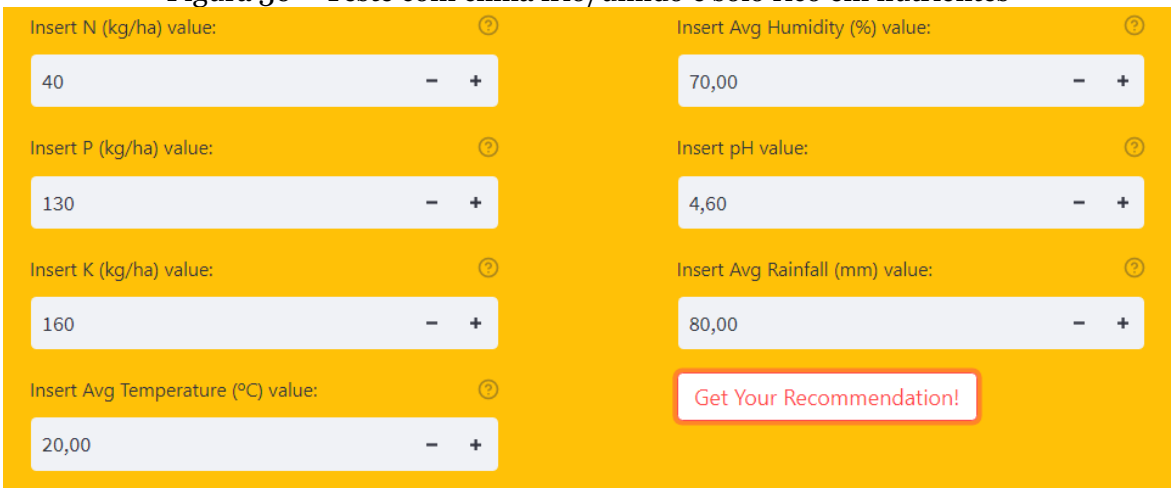
Os valores inseridos para teste, acabam que estão próximas as médias dos atributos do cultivo de manga. Assim, pode-se considerar coerente a recomendação de manga para as características climáticas e de solo inseridas.

4.2.2 Teste com clima frio/úmido e solo rico em nutrientes.

Através da Figura 36, observa-se que neste segundo teste foram utilizadas características de clima frio e úmido, e características de solo com alta concentração de nutrientes (*P* e *K*). Os valores utilizados foram:

- N = 40 kg/ha;
- P = 130 kg/ha;
- K = 160 kg/ha;
- Temperatura = 20°C;
- Umidade = 70%;
- pH = 4,6;
- Pluviosidade = 80 mm.

Figura 36 – Teste com clima frio/úmido e solo rico em nutrientes



Fonte: Autor

Após a inserção dos dados e o acionamento do botão *"Get Your Recommendation!"*, surgem na tela a recomendação, a foto do cultivo recomendado e a tabela com o resumo estatístico dos valores do conjunto de dados referente a aquele cultivo em específico. A Figura 37 representa o resultado deste segundo teste, sendo o resultado o cultivo de uvas (*grapes*).

Figura 37 – Resultado do segundo teste

Fonte: Autor

A Figura 38 apresenta o quadro com o resumo estatístico.

Figura 38 – Resumo estatístico do segundo teste

Statistics Summary about grapes NPK and Weather Conditions values in the Dataset.

	N	P	K	temperature	humidity	ph	rainfall
count	100.0000	100.0000	100.0000	100.0000	100.0000	100.0000	100.0000
mean	23.1800	132.5300	200.1100	23.8496	81.8752	6.0259	69.6118
std	12.4668	7.6190	3.2657	9.7386	1.1771	0.2983	2.9518
min	0.0000	120.0000	195.0000	8.8257	80.0164	5.5109	65.0110
25%	11.7500	125.7500	197.0000	16.2065	80.8595	5.7769	66.8368
50%	24.0000	133.0000	201.0000	23.0185	81.7246	6.0018	69.5362
75%	35.0000	139.0000	203.0000	30.8236	82.8992	6.3137	71.6094
max	40.0000	145.0000	205.0000	41.9487	83.9835	6.4996	74.9151

Fonte: Autor

O segundo teste também se mostrou satisfatória, devido aos valores inseridos pelo usuário estarem bem próximos as médias das características climáticas e de solo referentes ao cultivo de uvas.

5 Conclusão

Este trabalho teve como objetivo realizar um sistema de recomendação de plantio, dessa forma, através de características climáticas e do solo, recomendar a melhor cultura a ser plantada dentro dessas condições. Para isso foi proposto a criação de 5 modelos de classificação de AM, analisar a performance de cada modelo e desenvolvimento de uma interface *web* como interface do usuário utilizando o melhor modelo. Todos estes itens propostos conseguiram ser realizados com sucesso.

Com o objetivo de se aprofundar mais em Ciência de Dados e nas técnicas de Aprendizado de Máquina foram feitos estudos nestes diferentes modelos de classificação e optou-se pela linguagem Python pelo grande mercado que existe hoje nesta linguagem na área de AM. A utilização da linguagem de programação Python, assim como suas bibliotecas de análise, visualização e tratamento de dados, foi mais do que suficiente para entender a qualidade e distribuição dos dados obtidos, e por fim, a biblioteca Scikit-Learn se mostrou muito eficiente na criação de modelos robustos, todos obtendo uma ótima performance e com otimização dos parâmetros. Através do Streamlit foi possível a criação do aplicativo *web*, possibilitando uma interface simples e intuitiva para o usuário.

A respeito dos modelos de AM criados, todos apresentaram uma ótima performance nos testes, sendo os principais critérios utilizados para a escolha do modelo de Floresta Aleatória a sua performance e robustez comparado aos outros modelos desenvolvidos. Com relação ao aplicativo *web* para interação com o usuário, o aplicativo cumpre com os objetivos propostos em termos de facilidade de iteração e manutenção. Foi criado uma forma simples e rápida de escolha e troca de modelo, onde basta a troca de alguns parâmetros e então um novo modelo de AM já está sendo utilizado com os dados treinados.

As maiores dificuldades encontradas neste trabalho foram de entender como testar e treinar os modelos da melhor maneira possível, sendo possível principalmente conhecendo e aprofundando os estudos sobre as particularidades de cada tipo de modelo e métodos de validação. Outra dificuldade foi criar a interface do usuário dentro do aplicativo *web,* já que além do conhecimento técnico em programação, foi se necessário criatividade e se colocar no lugar do usuário, a fim de criar uma interface simples, para que qualquer pessoa tenha capacidade de compreender e utilizar sem dificuldades.

Como trabalhos futuros, uma melhoria ao aplicativo *web* seria em utilizar

outros atributos, para uma recomendação envolvendo outros fatores para um plantio de qualidade. Outra melhoria seria em realizar pesquisas envolvendo outros algoritmos de aprendizado de máquinas ou rede neurais. Também poderia obter dados coletados especificando a região, dados do Brasil ou da região de Uberlândia para um estudo de caso mais específico. Uma outra alternativa seria mudar a abordagem, para que se realize predições de produtividade de plantio ou reconhecimento de doenças através de imagens de plantações.

Referências bibliográficas

A importância do clima para o agronegócio. Canal Agro, 19 mai. 2020. Disponível em <https://summitagro.estadao.com.br/noticias-do-campo/importancia-clima-agronegocio/> Acesso em 25 ago. 2022.

A influência do pH do solo sobre a agricultura. BRFértil, 03 nov. 2019. Disponível em <https://brfertil.com.br/a-influencia-do-ph-do-solo-sobre-a-agricultura/> Acesso em 26 ago. 2022.

AKSOY, S.; HARALICK, R. M. Feature normalization and likelihood-based similarity measures for image retrieval. *Pattern Recognition Letters*, v. 22, n. 5, p. 563-582, 2001. https://doi.org/10.1016/S0167-8655(00)00112-4

ALTMAN, N. An introduction to kernel and nearest-neighbor nonparametric regression. *American Statistician - AMER STATIST*, v. 46, p. 175-185, 1992. https://doi.org/10.2307/2685209

ALVIM, João Paulo Nóbrega. Aplicação Web para geração de modelos automatizados de Aprendizado de Máquina. 2019. 108 f. Trabalho de Conclusão de Curso (Graduação em Engenharia da Computação) – Universidade Federal de Uberlândia, Uberlândia, 2019.

ATHAYDE, Mateus Prevelato. Crop_Recommendation_System. Disponível em: <https://github.com/MPrevelato/Crop_Recommendation_System> Acesso em 22 dez. 2022.

BANERJEE, P. SVM Classifier Tutorial. 2020. Disponível em: <https://www.kaggle.com/code/prashant111/svm-classifier-tutorial/notebook> Acesso em 21 set. 2022.

BENOS, L.; TAGARAKIS, A.C.; DOLIAS, G.; BERRUTO, R.; KATERIS, D.; BOCHTIS, D. Machine Learning in Agriculture: A Comprehensive Updated Review. *Sensors*, v. 21, pp. 3758, 2021. https://doi.org/10.3390/s21113758

BISHOP, Y. M.; Fienberg, S. E.; Holland, P. W. Discrete multivariate analysis: Theory and practice. *Applied Psychological Measurement*, v. 1, 1977.

BOSER, B. E.; GUYON, I. M.; VAPNIK, V. N. A training algorithm for optimal margin classifiers. *5th Annual ACM Workshop on Computational Learning Theory*, pp. 144-152, 1992. https://doi.org/10.1145/130385.130401

BRONSHTEIN, A. Train/Test Split and Cross Validation in Python. 2017. Disponível em: <https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61beca4b6/>. Acesso em 06 set. 2022.

BROWNLEE, J. Repeated k-Fold Cross-Validation for Model Evaluation in Python.

2020. Disponível em: <https://machinelearningmastery.com/repeated-k-fold-cross-validation-with-python/> Acesso em 07 set. 2022.

Caléndário agrícola: conheça as melhores épocas para plantar!. Jacto, 2018. Disponível em <https://blog.jacto.com.br/calendario-agricola-conheca-as-melhores-epocas-para-plantar/> Acesso em 25 ago. 2022.

COOMANS, D.; MASSART, D. Alternative k-nearest neighbour rules in supervisedpattern recognition: Part 1. k-nearest neighbour classification by using alternative voting rules. *Analytica Chimica Acta*, v. 136, p. 15-27, 1982. https://doi.org/10.1016/S0003-2670(01)95359-0

CORTES, C.; VAPNIK, V. Support-vector networks. *Machine Learning*, v. 20, p. 273-297, 1995. https://doi.org/10.1007/BF00994018

Data Collection | Data Robot ArtiĄcial Intelligence Wiki. [s.d.] *DATAROBOT*. Disponível em: <https://www.datarobot.com/wiki/data-collection/> Acesso em 02 set. 2022.

DUARTE, G. R. B. O que são fertilizantes NPK? Entenda todas as suas vantagens. CHBAGRO, 2020. Disponível em <https://blog.chbagro.com.br/o-que-sao-fertilizantes-npk-entenda-todas-as-suas-vantagens> Acesso em 26 ago. 2022.

DUDA, R. O.; HART, P. E. Pattern Classification and Scene Analysis. *New York: John Willey & Sons*, 1973.

FAMILI, A.; SHEN, W.; WEBER, R.; SIMOUDIS, E. Data preprocessing and intelligent data analysis. *Intelligent Data Analysis*, v. 1, p. 3-23, 1997. https://doi.org/10.1016/S1088-467X(98)00007-9

FIRICAN, G. The History of Machine Learning. *LightsOnData*, 2019. Disponível em <https://www.lightsondata.com/the-history-of-machine-learning/> Acesso em 29 ago. 2022.

GÉRON, A. Hands-On Machine Learning with Scikit-Learn and TensorFlow. United States of America: O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, 2017.

GUIDO, S.; MULLER, A. Introduction to Machine Learning with Python. United States of America.: O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, 2016.

HO, T. K. Random decision forests. *3rd International Conference on Document Analysis and Recognition*, v. 1, pp. 278-282, 1995. https://doi.org/10.1109/ICDAR.1995.598994

INGLE, A. Crop Recommendation Dataset. *Kaggle*. 2020. Disponível em: <https://www.kaggle.com/datasets/atharvaingle/crop-recommendation-dataset/metadata> Acesso em 27 set. 2022.

JIJO, B.; ABDULAZEEZ, A. Classification Based on Decision Tree Algorithm for Machine Learning. *Journal of Applied Science and Technology Trends*, v. 2, pp. 20-

28, 2021. https://doi.org/10.38094/jastt20165

KHAN, M.; QAYOOM, A.; NIZAMI, M.; SIDDIQUI, M.; WASI, S.; SYED, K. Automated Prediction of Good Dictionary EXamples (GDEX): A Comprehensive Experiment with Distant Supervision, Machine Learning, and Word Embedding-Based Deep Learning Techniques. *Complexity*, v. 2021, pp. 18, 2021. https://doi.org/10.1155/2021/2553199

KOHAVI, R. A study of cross-validation and bootstrap for accuracy estimation and model selection. *14th International Joint Conference on Artificial Intelligence*, v.2, pp. 1137-1143, 1995.

LIAKOS, K.G.; BUSATO, P.; MOSHOU, D.; PEARSON, S.; BOCHTIS, D. Machine Learning in Agriculture: A Review. *Sensors*, v. 18, pp. 2674, 2018. https://doi.org/10.3390/s18082674

LIN, M.; MING, L. Introduction to Data Science. 2021. Disponível em: <https://scientistcafe.com/ids/> Acesso em 11 set. 2022.

LIU, C. More Performance Evaluation Metrics for Classification Problems You Should Know. 2022. Disponível em: <https://www.kdnuggets.com/2020/04/performance-evaluation-metrics-classification.html> Acesso em 25 set. 2022.

Machine Learning Explained. *MIT Sloan*, 21 abr. 2021. Disponível em <https://mitsloan.mit.edu/ideas-made-to-matter/machine-learning-explained> Acesso em 27 ago. 2022.

Machine Learning. IBM, 15 jul. 2020. Disponível em <https://www.ibm.com/br-pt/cloud/learn/machine-learning> Acesso em 27 ago. 2022.

MARIUS, H. Multiclass Classification with Support Vector Machines (SVM), Dual Problem and Kernel Functions. 2020. Disponível em: <https://towardsdatascience.com/multiclass-classification-with-support-vector-machines-svm-kernel-trick-kernel-functions-f9d5377d6f02> Acesso em 21 set. 2022.

MITCHELL, T. M. Machine Learning. New York, NY, USA: McGraw-Hill, Inc., 1997.

MORAIS, Bruna Corrêa. Sistema integrado de análise e predição de indicadores de desempenho de um processo industrial. 2019. 90 f. Trabalho de Conclusão de Curso (Graduação em Engenharia de Controle e Automação) - Universidade Federal de Uberlândia, Uberlândia, 2020.

NALANI, A. NKK Classification using Scikit-learn. 2018. Disponível em <https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn> Acesso em: 12 set. 2022.

O que é NPK e como o adubo deve ser utilizado. *Canal Agro*, 17 fev. 2020. Disponível em: <https://summitagro.estadao.com.br/noticias-do-campo/o-que-e-npk-e-como-o-adubo-deve-ser-utilizado/ > Acesso em 25 ago. 2022.

Plotly. 2022. Disponível em: <https://plotly.com/python/> Acesso em 10 dez. 2022.

QIN, S. J.; CHIANG, L. H. Advances and opportunities in machine learning for process data analytics. *Computers Chemical Engineering*, v. 126, p. 465-473, 2019. https://doi.org/10.1016/j.compchemeng.2019.04.003

RASCHKA, S. About Feature Scaling and Normalization. 2014. Disponível em: <https://sebastianraschka.com/Articles/2014_about_feature_scaling.html>. Acesso em 05 set. 2022.

REFAEILZADEH, P.; TANG, L.; LIU, H. Cross-validation. Encyclopedia of Database Systems, *Springer*, 2009.

ROKACH, L.; MAIMOM, O. Decision Trees. Data Mining and Knowledge Discovery Handbook. *Springer*, Boston, 2022.

SAEED, M. Calculation Pearson Correlation Coefficiente in Python with Numpy. 2022. Disponível em: <https://stackabuse.com/calculating-pearson-correlation-coefficient-in-python-with-numpy/> Acesso em 10 dez. 2022.

Seaborn. 2022. Disponível em: <https://seaborn.pydata.org/> Acesso em 10 dez. 2022.

SENGIK, E. S. Os Macronutrientes e os Micronutrientes das Plantas. 2003. Universidade Estadual de Maringá. 22 p. Notas de Aula

SORENSEN, C.A.G.; KATERIS, D.; BOCHTIS, D. ICT Innovations and Smart Farming. In Communications in Computer and Information Science, *Springer,* Germany, 2019.

SOUZA, P. V. D. Rede neural artificial para predição da produtividade da cultura do milho. 2021. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) - Universidade Tecnológica Federal do Paraná, Santa Helena, 2021.

STEHMAN, S. Selecting and interpreting measures of thematic classification accuracy. *Remote Sensing of Environment*, v. 62, pp. 77-89, 1997. https://doi.org/10.1016/S0034-4257(97)00083-7

Streamlit. 2022. Disponível em: <https://streamlit.io/> Acesso em 10 dez. 2022.

Supervised Learning: Introduction to Classification: K-Nearest Neighbor Cheatsheet | Codeacademy. [s.d.]. Codeacademy. Disponível em: <https://www.codecademy.com/learn/introduction-to-supervised-learning-skill-path/modules/k-nearest-neighbors-skill-path/cheatsheet.> Acesso em 12 set. 2022.

TAMAIS, A. Modelos de Predição | Naive Bayes. 2019. Disponível em: <https://medium.com/turing-talks/turing-talks-16-modelo-de-predi%C3%A7%C3%A3o-naive-bayes-6a3e744e7986> Acesso em 13 set. 2022.

THILAKARATHNE, N. N., BAKAR, M., ABAS, P. E., & YASSIN, H. A Cloud Enabled Crop Recommendation Platform for Machine Learning-Driven Precision Farming. *Sensors*, v. 22, pp. 6299, 2022. https://doi.org/10.3390/s22166299

VERMA, Yugesh. A Complete Guide to Categorical Data Encoding. 2021. Disponível

em: <https://analyticsindiamag.com/a-complete-guide-to-categorical-data-encoding/>. Acesso em 06 set. 2022.

ZHANG, H. The optimality of naive bayes. *17th International Florida Artificial Intelligence Research Society Conference*, Estados Unidos, 2004.