UNIVERSIDADE TECNÓLOGICA FEDERAL DO PARANÁ
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA AMBIENTAL
MULTICAMPI APUCARANA E LONDRINA

LUCAS AUGUSTO VIEIRA

REDUÇÃO DE USO DE AGROTÓXICOS POR MEIO DE INTELIGÊNCIA ARTIFICIAL

APUCARANA
2021

LUCAS AUGUSTO VIEIRA

REDUÇÃO DE USO DE AGROTÓXICOS POR MEIO DE INTELIGÊNCIA ARTIFICIAL

REDUCTION OF PESTICIDES USE BY MEANS OF ARTIFICIAL INTELLIGENCE

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Ambiental / multicampi Apucarana e Londrina da Universidade Tecnológica Federal do Paraná, como requisito parcial à obtenção do título de Mestre em Engenharia Ambiental.

Área de Concentração: Engenharia Ambiental

Linha de Pesquisa: Tecnologias Ambientais

Orientador: Prof. Dr. Thiago Gentil Ramires

APUCARANA

2021

-

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Campus Londrina

LUCAS AUGUSTO VIEIRA

REDUÇÃO DE USO DE AGROTÓXICOS POR MEIO DE INTELIGÊNCIA ARTIFICIAL

Trabalho de pesquisa de mestrado apresentado como requisito para obtenção do título de Mestre Em Engenharia Ambiental da Universidade Tecnológica Federal do Paraná (UTFPR). Área de concentração: Engenharia Ambiental.

Data de aprovação: 26 de Agosto de 2021

Prof Thiago Gentil Ramires, - Universidade Tecnológica Federal do Paraná

Prof Alesandro Bail, Doutorado - Universidade Tecnológica Federal do Paraná

Prof Jorge Alberto Martins, Doutorado - Universidade Tecnológica Federal do Paraná

Prof Luiz Ricardo Nakamura, Doutorado - Universidade Federal de Santa Catarina (Ufsc)

Documento gerado pelo Sistema Acadêmico da UTFPR a partir dos dados da Ata de Defesa em 26/08/2021.

Dedico este trabalho aos meus pais, minha esposa, minha filha e aos meus amigos.

AGRADECIMENTOS

Ao orientador, amigo e “pai científico”, Prof. Dr. Thiago Gentil Ramires, pela competência, respeito, atenção, estímulo e dedicação com que conduziu este processo, do início da ideia até a sua concretização.

Agradeço aos meus familiares, que são minha base e sustentação durante todo o caminho percorrido, em especial à minha esposa, Danielle Gonçalves de Oliveira Prado, e à minha filha, Giovana Prado Vieira, que me estimulam cada dia mais galgar novos patamares na carreira cientifica.

Externo meus agradecimentos também ao PPGEA – Campus Apucarana e aos membros da banca examinadora por aceitarem prontamente o nosso convite.

“Pesquisar é acordar para o mundo.”

(LAMY, 2011)

VIEIRA, Lucas Augusto. **REDUÇÃO DE USO DE AGROTÓXICOS POR MEIO DE INTELIGÊNCIA ARTIFICIAL**. 2021. 95 p. Dissertação (Mestrado em Engenharia Ambiental) - Universidade Tecnológica Federal do Paraná. Apucarana, 2021.

RESUMO

O cultivo de cana-de-açúcar vem ganhando grande destaque em vários países devido à diversidade de usos. A modernização da agricultura permitiu alta produtividade em diversos cultivares, os quais são afetados em grande parte pela invasão de ervas daninhas. Com a agricultura sustentável, o uso de herbicidas tem sido cada vez mais evitado na sociedade, exigindo métodos mais eficazes de controle de pragas. Nessa pesquisa foram propostos diversos modelos de inteligência artificial capazes de identificar a invasão de plantas daninhas em um cultivar de cana-de-açúcar, utilizando como variáveis de input quatro espectros de cores, obtidas por uma câmera multiespectral montada em um veículo aéreo não tripulado. Os modelos estudados no presente trabalho foram: GAMLSS, Redes Neurais, Árvores de Decisão, Florestas Aleatórias, Bagging Cart, Gradient Boosting Machine, K-Nearest Neighbors e C4.5. O modelo com melhor desempenho preditivo dentre os apresentados foi o modelo Florestas Aleatórias. Para cada modelo, toda a parte computacional foi disponibilizada para futuros estudos. Com a identificação exata da infestação de ervas daninhas, é possível realizar o manejo em campo com aplicações de herbicidas nos locais exatos, evitando assim o aumento do custo de produção e, principalmente, evitando a contaminação desnecessária de solos e efluentes.

Palavras-chave: Agricultura de Precisão; Inteligência artificial; Sensoriamento Remoto.

VIEIRA, Lucas Augusto. **REDUCTION OF PESTICIDES USE THROUGH ARTIFICIAL INTELLIGENCE**. 2021. 95 p. Dissertation (Master in Environmental Engineering) – Federal University of Technology – Paraná, Apucarana, 2021.

ABSTRACT

The cultivation of sugarcane has gained great prominence in several countries due to its large diversity of application. The modernization of agriculture has allowed high productivity in several cultivars, which are largely affected by the invasion of weeds. With sustainable agriculture, the use of herbicides has been strongly avoided, and more effective pest control methods have been required. In this survey, several artificial intelligence models capable of identifying the invasion of weeds in a sugarcane cultivar were proposed. In addition, four colors spectra based on a multispectral camera coupled to an unmanned aerial vehicle were employed as input variables. The models studied in this work were: GAMLSS, Neural Networks, Decision Trees, Random Forests, Bagging Cart, Gradient Boosting Machine, K-Nearest Neighbors and C4.5. Among the studied models, the Random Forests presented the best performance. For each model, all computational codes was presented for future new surveys. The precise identification of the weed infestation place allowed field management to be carried out only by the herbicide application in the damaged points. This procedure avoids an increase in the production costs, and mainly prevents the soil contamination and generation of hazardous effluents. Thus, avoiding an increase in production costs and, above all, it is avoiding unnecessary contamination of soils and effluents.

Keywords: Precision Agriculture; Artificial Intelligence; Remote Sensing.

.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CART	Classification Regression Trees
CSS3	Cascading Style Sheets
DDR	Double Data Rate
G	Nível de faixa verde
GAMLSS	Generalized Additive Models for Location, Scale and Shape
GB	Gigabytes
GBM	Gradient Boosting Machine
GHz	Gigahertz
GNUGPL	General Public License
GPS	Sistema de Posicionamento Global
HTML5	Hypertext Markup Language, versão 5
Hz	Hertz
IA	Inteligência artificial
IBGE	Instituto Brasileiro de Geografia e Estatística
ILPF	Integração Lavoura-Pecuária-Floresta
IoT	*Internet of thinks*
Lat	Latitude
Long	Longitude
NIR	Nível de banda de infravermelho próximo
pH	Potencial Hidrogeniônico
R	Nível de banda vermelha
RE	Nível de faixa de borda vermelha
RF	Random Forest
RN	Redes Neurais
SQL	Structured Query Language
SSD	Solid State Drives
UTM	Universal transversa de mercator
VANT´S	Veículos aéreos não tripulados

SUMÁRIO

1 INTRODUÇÃO

O interesse na produção de cana-de-açúcar está aumentando no mundo devido a sua ampla gama de aplicações, inclusive na natureza, na forma de forragem, alimento animal, assim como matéria-prima para a fabricação de álcool e açúcar. Além disso, seus resíduos têm alto valor econômico, uma vez que podem ser transformados em fertilizantes ou combustíveis de base renováveis (etanol). Entre os maiores produtores de cana-de-açúcar no mundo, o Brasil é o país de maior produção, representando 40% da produção mundial, com expectativa de alta de 3% para a safra 2020/2021 em relação à safra anterior, segundo a União da Indústria de Cana-de-Açúcar (UNICA), totalizando uma produção de mais de 590 milhões de toneladas de commodity agrícola. Segundo o Ministério da Agricultura, entre janeiro e novembro de 2016, a exportação de açúcar chegou a 9,3 bilhões de dólares. Ainda, segundo a UNICA, o Brasil é responsável por 20% da produção global de cana-de-açúcar, sendo que desse total, 45% é a referente destinada ao mercado externo.

A introdução da informática como ferramenta para o gerenciamento e simulação de operações agrícolas, principal característica da agricultura 4.0, foi um dos fatores que mais impactaram na redução dos custos de produção da cana-de-açúcar no Brasil. Programas e sistemas desenvolvidos para este fim permitiram a redução da frota de caminhões, tratores, colheitadeiras e implementos, maximização da quantidade de cana-de-açúcar por hectare, otimização da operação das frentes de colheita, avaliação do desempenho online e controle de todas as atividades agrícolas operacionais. Mesmo com o desenvolvimento e a disponibilidade comercial do Sistema de Posicionamento Global (GPS), os sensores necessários para quantificar os fatores de produção, como produtividade, fertilização aplicada, infestação de plantas invasoras e propriedades do solo, ainda estão em estágios de constante desenvolvimento, devido ao surgimento contínuo de novas tecnologias. Embora o Brasil possua um vasto território agricultável, o qual possibilita a ampliação da produção de cana-de-açúcar, isso não faz com que o custo de produção diminua, ainda mais quando comparado com a concorrência mundial. Isso faz com que o uso de novas tecnologias para redução de custos, visando o aumento da produtividade sem a contrapartida do aumento de área agricultável, torna-se de extrema importância atualmente.

Em outra via, a pressão mundial pela diminuição do desmatamento (o qual possibilitaria a expansão de áreas agricultáveis), assim como o apelo global para a redução de uso de herbicidas e agrotóxicos, faz com que os produtores de cana-de-açúcar tenham que buscar e introduzir novas técnicas de manejo em suas plantações, fazendo com que o produto final seja, além de competitivo, sustentável. Segundo o Instituto Brasileiro de Geografia e Estatística (IBGE), de 2000 a 2010, o maior responsável pelo desmatamento de áreas florestais foi a expansão das atividades agrícolas, totalizando uma área de 236.600 km^2. Por outro lado, como já mencionado anteriormente, existe uma tendência mundial que valoriza o consumo de produtos sustentáveis, fazendo com que os produtos que não se enquadrem em tais políticas sejam boicotados, como por exemplo, em 2006, um movimento liderado pelo Greenpeace, fez com que houvesse boicote à compra da soja produzida em áreas desmatadas (NEPSTAD, et al., 2014). Por fim, também podemos destacar a correlação entre o desmatamento e o surgimento de doenças, o que é intensificado em meio a uma pandemia, período contemporâneo a esta dissertação, como por exemplo, em um estudo apresentado por Santos e Almeida (2018), os autores apresentam uma análise espacial, a qual relaciona o desmatamento, assim como outros fatores, com a elevação de infecções de malária na região amazônica.

1.1 JUSTIFICATIVA

Diante dos fatos expostos acima, essa dissertação elenca o uso de tecnologias de inteligência artificial em cultivares de cana-de-açúcar, visando a redução do uso de agrotóxicos, fazendo com que o custo de produção diminua e, consequentemente, aumente a concorrência, assim como torne as áreas de cultivo mais produtivas, visando a diminuição da expansão de áreas agricultáveis, via desmatamento.

1.2 OBJETIVOS

O presente estudo está dividido em objetivo geral e objetivos específicos.

1.2.1 Objetivo geral

O principal objetivo da pesquisa é o desenvolvimento de modelos de inteligência artificial que possibilitam o mapeamento de cultivares de cana-de-açúcar, a partir de diferentes propostas disponíveis na literatura. Como resultado, espera-se que essa pesquisa ajude os produtores a reduzirem o uso de herbicidas, técnica essa conhecida como agricultura de precisão.

1.2.2 Objetivos específicos

Com o auxílio de ferramentas computacionais e imagens multiespectrais, os resultados dessa pesquisa devem:

-Identificar os modelos com melhores taxas de acurácia para classificar os pixels das imagens como ervas-daninhas, solo (não germinação das sementes de cana-de-açúcar), e cana-de-açúcar, reduzindo então o uso de agrotóxicos, bem como aumentando a produtividade por área.

- Identificar os modelos com menores custos computacionais, possibilitando que os mesmos possam ser implementados em tempo real e hábil no manejo da cultura de cana-de-açúcar;

- Disponibilizar toda a rotina computacional para que futuros usuários (pesquisadores e agricultores) possam se beneficiar dos resultados obtidos dessa pesquisa.

2 REVISÃO DE LITERATURA

As revisões realizadas pelo autor, por meio de levantamentos bibliográficos, estão dispostas nas próximas subseções, as quais estão organizadas por seus respectivos temas.

2.1 AGROTÓXICOS E A POLUIÇÃO AMBIENTAL

Em um cenário onde é notório um aumento significativo na produção agrícola, voltadas tanto para o consumo interno quanto para a exportação, é perceptível a utilização de agrotóxicos com ação biocida, das quais cita-se os herbicidas, fungicidas, inseticidas, dentre outros, que visam um controle sistemático de parasitas que ocasionam prejuízo à lavoura (FILIZOLA et al., 2002). Costa et al. (2004) relatam que os princípios ativos destes produtos apresentam em sua composição poluentes como metais pesados, surfactantes dentre outros, os quais causam de maneira direta danos irreversíveis ao meio ambiente. A necessidade da utilização de tais produtos se torna viável pela facilidade e rendimento serem totalmente altos, deixando desperceber os danos ambientais e de saúde pública.

A ascensão da cadeia produtiva do agronegócio, trouxe consigo poluições, enfermidades e patologias relacionados ao uso indiscriminado de agrotóxicos, ocasionando danos ao meio ambiente e à saúde humana. Esse uso desenfreado com o objetivo de atingir as pragas, acaba por si contaminando as lavouras, o meio ambiente e consequentemente a vida humana (CARNEIRO, 2015).

A contaminação do solo e, consequentemente dos recursos hídricos, se dá principalmente em área de manejo tradicional, onde o solo não é preparado de forma correta para a aplicação, e seu manejo se dá de maneira intensiva, com pouca presença de resíduos orgânicos em sua superfície, deixando-o mais favorável à degradação, ou seja, um aumento em seu processo erosivo, dificultando o escoamento superficial.

2.2 PRODUTIVIDADE E USO DE HERBICIDAS

Relacionar eficiência na produção à baixos custos dos alimentos e implementos é um dos fatores de maior relevância no que tange ao conceito de produtividade. No entanto algumas ervas-daninhas se destacam negativamente, de forma a influenciar a produção final da safra, independente da cultura trabalhada. Isso se torna preocupante e demanda um cuidado especial. Estudos mostram que a redução de 50% da produtividade pode estar relacionada à infestação das ervas daninhas capim-braquiária e capim-colonião (Panicum maximum Jacq.), muito frequentes em pastagens degradadas (KUVA, 1999).

Atualmente procura-se agregar novas tecnologias ao uso de herbicidas, com a finalidade de tornar o manejo mais eficiente. É possível encontrar no mercado várias opções para seu uso, porém é importante levar em consideração o conhecimento do agricultor. De nada adianta ter uma boa ferramenta, se ela não é utilizada de forma correta. O maior desafio será capacitar sua inserção no conhecimento das ferramentas tecnológicas de forma coerente todos os agentes envolvidos nesse processo.

De acordo com Richard Jr. (1997), o herbicida imazapyr não prejudica a produtividade, em contrapartida, em Richard Jr. (1998), verifica-se que a utilização do mesmo herbicida modifica as propriedades físicas e a produtividade final quando aplicado mais de uma vez. Ainda, o herbicida atrazina, que também é muito utilizado na cultura da cana-de-açúcar, possui adsorção diretamente relacionada à proporcionalidade da quantidade de matéria orgânica, bem como o teor de umidade (WALKER, 1972). Por outro lado, ainda sobre o herbicida atrazina, Baoshan et al. (1996) apresenta um estudo mostrando que a matéria orgânica age com adsorvente do herbicida. Senesi (1992) relata que a adsorção, retenção e degradação de herbicidas está diretamente relacionada ao pH, umidade, capacidade de trocas catiônicas e temperatura.

O uso de estratégias na aplicação dos herbicidas é fundamental para o bom desempenho do plantio. É importante que o produtor veja sua lavoura com um olhar diferenciado e combine os produtos já utilizados e que tenham resultados eficazes, com a implementação de novas tecnologias. O uso indiscriminado de agrotóxicos em qualquer cultura leva à vários danos ambientais e, consequentemente, à sociedade (SPADOTTO, 2006). O uso frequente e inadequado de agrotóxicos leva à

contaminação de solos, atmosfera, lençóis freáticos, trazendo danos à saúde humana. Torna-se então de suma importância que os produtores observem e analisem o comportamento de cada agrotóxico, de acordo com a cultura, levando em conta a periculosidade relacionada aos danos ambientais.

Fatores como assistência técnica, infraestrutura e uso de novas tecnologias relacionadas à produtividade agrícola são temas cruciais e muito abordados na literatura. No Brasil, diversos programas, como ‘’Plano ABC’’, ‘’Plano Agrícola e Pecuário’’, ‘’Prona Solos’’, dentre outros, vêm sendo desenvolvidos, possibilitando a capacitação adequada do agricultor. Além disso, existem políticas que fomentam e incentivam o desenvolvimento do cultivo no campo, como o Programa ‘’Política Nacional ILPF’’.

Sem investimentos, como o crédito rural, que auxilia na pesquisa e na modernização dos implementos do setor agrícola, a produção fica prejudicada, visto que seu êxito está diretamente vinculado ao apoio recebido tanto pelo governo quanto por demais investidores. Gasques et al. (2012) relatam que a queda da produtividade das principais lavouras está relacionada principalmente à redução e falta de investimentos em pesquisa, que culmina em uma menor quantidade de inovações tecnológicas no setor.

É imprescindível frisar que, produtividade e rentabilidade andam lado a lado, uma vez que se leva em consideração não apenas o quanto está produzindo, mas também o montante gasto. Por isso, espera-se que todo o capital aplicado seja utilizado de forma a ser convertido em lucro para o gestor.

2.3 CANA-DE-AÇÚCAR

A cana-de-açúcar, gramínea do gênero Saccharum, possui uma característica semi-perene. Para acumular sacarose nos colmos, apresenta uma queda hídrica ou térmica em sua produção na fase vegetativa (JOAQUIM, 1998). Com sua grande importância econômica para o país, o cultivo da cultura exerce um papel de suma relevância para o mercado interno e para o mercado externo. O estado de São Paulo se destaca entre os maiores produtores, com dados estatísticos que mostram a produção de 60% de açúcar e 62% de álcool da produção nacional (UNICA, 2012).

O cultivo dessa cultura afeta diretamente a produção de etanol e de energia elétrica através da queima dos resíduos e/ou bagaço nas usinas sucroalcooleiras. O Brasil lidera o ranking da produção mundial de cana-de-açúcar, com o estado de São Paulo, que é responsável por 53,6% da produção nacional (CONAB, 2019).

Ainda que o país possua uma elevada produção, algumas regiões ainda apresentam técnicas tradicionais e de eficiência baixa comparada às técnincas utilizadas atualmente. Uma delas é a coleta de imagens multiespectrais oriundas de sensoriamento remoto e agricultura de precisão, realizado em terrenos com plantios. Nelas são extraídas informações relevantes, que fornecem resultados quantitativos de suma importância e contribuem para potencializar o aumento produtivo do setor sucroalcooleiro (RUDORFF e MOREIRA, 2002).

Em complemento, necessita-se de tecnologias que auxiliem no controle de pragas e ao mesmo tempo diminuam o lançamento de contaminantes ao solo, água, ar e lençóis freáticos. O controle químico, um dos métodos mais utilizados, devido à sua economia e fácil aplicabilidade, vem sendo retirado aos poucos. Segundo Oliveira Jr. (2011), os herbicidas que são utilizados selecionam as plantas corretas para o controle, porém há a necessidade de pulverização em toda a lavoura.

Necessita-se então utilizar recursos computacionais para que se possa mapear culturas com a finalidade de auxiliar na tomada de decisões. A possibilidade de avaliação de dados, predispõe um melhor planejamento de estratégias com a finalidade de um aumento na produtividade. Todo esse processo recebe o nome de sensoriamento remoto. Em outras palavras, o sensoriamento remoto é um instrumento que auxilia na coleta e no processamento de dados de uma determinada área agrícola (MOTOMIYA et al., 2012).

2.4 AGRICULTURA DE PRECISÃO

Em tempos em que a agricultura enfrenta variados desafios, devido aos elevados custos de produção, nota-se um denominador comum entre a rentabilidade da atividade e a produtividade. Para transpor esses desafios impostos, surgiram no decorrer dos anos, tecnologias que auxiliam na elevação do potencial produtivo, no que tange ao uso de sensoriamento remoto, ferramenta esta que possibilita de forma rápida e precisa, informações do desenvolvimento das culturas agrícolas, norteando o produtor na tomada de decisões. A utilização de tecnologias no campo traz

informações cada vez mais precisas, as quais poderiam facilmente passar despercebidas com o uso de técnicas já consideradas retrógradas.

A agricultura de precisão, também conhecida como *precision farming*, ou também agricultura 4.0 (esse conceito engloba uma área maior, a qual contém agricultura de precisão), consiste em um sistema de manejo integrado de informações e tecnologias, os quais visam maximizar o rendimento dos cultivos, por consequência, o lucro (COELHO, 2005). Atualmente existem inúmeras técnicas adotadas na agricultura de precisão, sendo que algumas dessas já vêm sendo utilizadas desde algumas décadas. Pires et al. (2004) citam algumas com maior visibilidade no mercado, como por exemplo sistemas de informações geográficas, monitoramento das áreas "*CropScouting*", monitores de colheita, sistema de posicionamento global, sensoriamento remoto de vegetal, amostradores de solo, colheitadeiras automáticas, sensores de matéria orgânica, plantas daninhas, umidade de solo, potencial Hidrogeniônico (pH), nitrato no solo, compactação (penetrômetros), internet das coisas "*internet of thinks - IoT",* sensores de condutividade elétrica do solo, sensores de umidade e de proteína de grãos, clorofilômetros, sensores de dinâmica da fertilidade, pulverizadores de precisão, fotografias aéreas, dentre outras.

Shockley et al. (2011), ao adotarem um sistema de plantio direto, utilizando um sistema de inovação de processo produtivo, demonstraram uma redução de custos em torno de 2,4%, 2,2% e 10,4% para sementes, fertilizantes e combustíveis, respectivamente, em uma plantação de soja e milho. Artuzo et al. (2017) apresentaram um estudo em que ao aplicar técnicas de agricultura de precisão em uma produção de soja e milho, foi reduzido o impacto ambiental, devido ao menor uso de fertilizante agrícola, em função da estimativa para a necessidade real da área a ser aplicada. Além disso, com a redução nos custos de aquisição de fertilizantes agrícolas e aumento na produção, o processo produtivo se tornou eficiente economicamente, ou seja, reduziu os custos econômicos.

Em uma pesquisa publicada por Bassoi et al. (2019), é possível encontrar detalhes de como se realiza todo o processo da agricultura de precisão, além de apresentar como se dá o uso de hardwares e softwares, mostrando que eles podem ser realizados com diferentes graus de automação. No Brasil, essa discussão tem tomado uma proporção cada vez maior e sua adaptação e validação tem crescido bastante nos últimos anos.

De acordo com Molin et al. (2015), práticas agrícolas, sejam elas em grande ou pequena escala, são tratadas de maneira uniforme erroneamente, visto que, quando existem diferenças, seja no solo ou no relevo, exige-se um cuidado no tratamento de cada uma das regiões, mesmo considerando uma mesma cultura. A variabilidade entre as culturas é diretamente proporcional à área de plantio, tornando cada vez mais difícil o controle do manejo. A agricultura de precisão necessita de ferramentas de tecnologias como Sensoriamento Remoto e o Sistema de Navegação Global.

Insere-se nesse contexto a aplicação de drones para coleta de dados na identificação de zona de manejos. Obtém-se também uma série de imagens de satélites e dados de produtividade, que permitem identificar e manejar de forma adequada cada região. A aplicabilidade dos veículos aéreos não tripulados (VANT´s), mais conhecidos no meio como ‘’drones’’, tem de certa maneira, suas vantagens e desvantagens. Dentre as principais vantagens está a facilidade de aquisição, o custo e a qualidade dos produtos gerados (mapeamento detalhado em alta resolução) e, dentre as desvantagens, cita-se a pequena cobertura de abrangência espacial, elevado custo computacional, especialização dos operadores através de cursos específicos e problemas relacionados à calibração das câmeras a bordo. Para isso, requer um entendimento prévio de conceitos de sensoriamento remoto (MORIYA et al., 2017).

A necessidade da utilização da adesão da inteligência artificial vem a cada dia mais se reformulando na agricultura, com a aplicação e desenvolvimento de modelos cada vez mais precisos, com foco na área de qualidade e diminuição de custos (KUMAR, 2018). É notório que a inteligência artificial está sendo aplicada em variados campos do setor agrícola, principalmente no setor de maquinários e implementos agrícolas, os quais terão a autonomia de tomar decisões no lugar do homem, possibilitando que o produtor, de maneira remota, acompanhe seu rendimento, desenvolvimento, entre outros, por meio de relatórios, resultando em uma redução de desperdício e uma simplificação do sistema agrícola (SOAM e RAGHUPHATI, 2018).

2.5 INTELIGÊNCIA ARTIFICIAL

A inteligência artificial (IA) é um ramo da ciência que abrange a capacidade de processamentos computacionais com finalidade de resolução de problemas e, para isso, utiliza métodos investigativos fundamentais, acompanhados de experimentação com apoio tecnológico para a resolução. Advém da vontade humana de emular em computadores as capacidades cognitivas à nossa inteligência. Se compõem em vários subcampos de tecnologia como o aprendizado de máquina (*Machine Learning*), e outros subcampos, que nos últimos campos teve seu ápice que são as Redes Neurais ou *Deep Learning*, que se cunhou a terminologia desde meados do pós Segunda Guerra Mundial, onde devido ao declínio da indústria e da escassez de mão de obra, as memas se viram obrigadas a aumentar a mecanização da produção, utilizando máquinas com maior complexidade e com uma gama maior de funções (FREITAS, 2016).

Nas últimas décadas houve uma explosão de dados e informações devido à transformação digital e uma redução significativa no custo da capacidade computacional, além da evolução dos algoritmos produzidos a partir destas tecnologias.

Segundo Pozzebon et al. (2004), uma visão histórica sobre a IA mostra que em meados do século XX seu avanço se deu em consonância com a evolução computacional, onde tornou-se cabível a simulação de aspectos da inteligência humana, o que fez com que o homem indagasse se as evoluções das máquinas seriam capazes de ter a capacidade cognitiva do ser humano.

Na área computacional, tem-se que a IA está voltada ao desenvolvimento de sistemas inteligentes de informática, que mostrem características, as quais vinculam inteligência do comportamento humano com artefatos de computação (FERNANDES, 2003). Atualmente, há uma forte relacão entre os pesquisadores e IA. Segundo fontes, até meados do ano de 2050, a produção mundial alimentícia necessariamente terá que ter um aumento de 50% em sua produção, relacioando aos tempos atuais, com um menor aparato de recursos e uma maior eficiência no processo produtivo, através da utilização das ferramentaas de inteligência artificial (EMBRAPA, 2019).

Em via de regra, a IA sistematiza, normatiza e automatiza determinadas tarefas, tendo relevância nas variadas áreas da atividade intelectual humana. Em

síntese, seu principal objetivo é fazer com que computadores operem de maneira inteligente com a finalidade de obtenção na melhoria de resultados, em tempos que o acúmulo de informações e de dados são elevados. São necessárias ferramentas para rápida leitura e interpretação.

2.6 LINGUAGENS COMPUTACIONAIS

A atual era digital e o avanço tecnológico contribuíram com o crescimento da ciência. Ferramentas de informática auxiliam na abordagem de problemas que antes não podiam ser solucionados.

Para Vieira e Lima (2001), a linguística computacional teve seu impulso nas décadas de 1950 e 1960 por meio do desenvolvimento de programas na área da Inteligência Artificial. Na literatura existem diversas opções de linguagens de programação, e cada uma apresenta uma vantagem específica.

A linguagem *Python*, por exemplo, é eficaz no desenvolvimento de scripts com a finalidade de realizar análises espaciais, gerenciamento e conversão de dados, automação e produção de mapas (ESRI, 2018). De acordo com De Paiva (2018), esse software permitiu ao poder público desenvolver planos de monitoramento, conservação e tomada de decisões com relação aos recursos hídricos do bioma Pantanal. A metodologia utilizada é considerada promissora e econômica, visto que o script gerado pode ser replicado inúmeras vezes para diferentes dados.

Campione e Walrath (1996) apresentam a linguagem computacional Java, que é adequada para aplicações que se baseiam em redes de Internet, redes fechadas ou ainda, programas stand-alone. Essa linguagem possui uma poderosa versatilidade, oferecendo recursos suficientes para construir inúmeros aplicativos que podem ser executados de forma online ou não (WUTKA, 1997). Da Silva et al. (2009) utiliza o software com a finalidade de georreferenciar dados obtidos para obtenção de cálculos de tempo de irrigação necessário para que uma determinada área da propriedade rural atinja uma umidade específica.

O uso da linguagem SQL (Structured Query Language) também tem se destacado muito nos últimos tempos, devido sua presença constante em sistemas de big data. Através do uso de uma biblioteca de códigos web foi possível a estruturação de páginas para a internet através das tecnologias HTML5, CSS3 e JavaScript. Mueller et al. (2016) usaram SQL para analisar imagens e dados espaciais de culturas agrícolas.

O R (Team, 2020), software gratuito e *open source,* muito requisitado por estatísticos, foi a ferramenta escolhida para esse trabalho. A vantagem desse programa é o fácil acesso aos pacotes disponíveis no mesmo, além de que o software pode ser utilizado em variados sistemas operacionais, como Windows, Linux e Mac. Por se tratar de um software com de código aberto, assim como o *Phyton*, o R permite que novas funções sejam implementadas facilmente por diferentes usuários do mundo, tornando o mesmo uma ferramenta poderosa em quaisquer áreas de pesquisa. Na área de cultivares agrícolas, Peternelli et al. (2009) utilizou esse software com a finalidade de efetuar uma análise de genealogias e coeficiente de graus de parentesco. Para tanto, este trabalho utilizará o R como ferramenta para a pesquisa.

3 MATERIAL E MÉTODOS

São apresentadas nas próximas subseções os dados que foram utilizados nessa pesquisa, o detalhamento do software utilizado (R e RStudio), oito modelos de classificação utilizados, assim como técnicas de predição e seleção de modelos.

3.1 DADOS

O conjunto de dados utilizado foi coletado em uma fazenda experimental situada em Piracicaba (SP) por uma equipe de pesquisadores da empresa Raízen, na qual a lavoura foi dividida em dois campos experimentais, denominados campo 1 e campo 2, separados por uma estrada e vegetação natural. Para ambos os campos, o plantio foi na terceira soqueira, com 120 dias depois da sua última colheita. Um veículo aéreo não tripulado foi utado para coletar os dados, nos quais uma câmera multiespectral foi acoplada, tirando fotos de ambos os campos e retornando as seguintes informações:

- NIR: nível de banda de infravermelho próximo;
- RE: nível de banda de borda vermelha;
- R: nível de banda vermelha;
- G: nível de banda verde;
- Lat: coordenada de latitude em universal transversa de mercator (UTM);
- Long: coordenadas de longitude em UTM.

Para ilustrar o processo de coleta de dados, é apresentado na Figura 1 (esquerda) o drone acoplado com a câmera multiespectral, assim como a imagem obtida pelo mesmo (direita) com as respectivas informações, obtidas em um único pixel, de cada nível de espectro, juntamente com as coordenadas de latitude e longitude.

FIGURA 1 - Drone acoplado com a câmera multiespectral (esquerda) e mapeamento da plantação experimental com as informações das cores e coordenadas obtidas para um pixel (direita).

FONTE: Autoria própria (2021).

Depois de obter as informações das cores para todos os pixels dos dois campos experimentais, com as respectivas coordenadas, os pesquisadores da empresa Raízen foram para os campos e registraram, manualmente, para algumas coordenadas selecionadas aleatoriamente, qual era a variável resposta Y presente no lugar, sendo classificada em três níveis:

$$Y = \begin{cases} 1, & solo \\ 2, & cana - de - açucar \\ 3, & erva\ danimnha \end{cases}$$

Vale ressaltar que o nível 3 agrupa diferentes espécies de plantas daninhas: *Brachiaria decumbens, Cynodon dactylon* e *Amaranthus viridis.* O total de pontos observados pelo drone para ambos os campos foi N = 127.853, dos quais apenas n = 8801 (6,88%) foram classificados manualmente, ou seja, apenas 8801 observações do conjunto de dados contêm informações sobre a resposta Y. A área

total dos dois campos é de 4,47 ha, dos quais 0,3 ha foram observados (0,06 ha de solo; 0,13 ha de cana-de-açúcar; e 0,11 ha de plantas daninhas).

A principal proposta neste trabalho é construir modelos que classifiquem as respostas Y, ou seja, modelos que classifiquem a observação encontrada em cada pixel, com base nas informações de cores NIR, RE, R e G, usando um modelo de classificação. Um primeiro estudo publicado com esses dados é apresentado em Righetto et al. (2019), em que os autores propuseram um modelo probabilistico para realizar tais classificações, o qual iremos descutir detalhadamente nas próximas seções.

Posteriormente, após determinado o melhor modelo de classificação, utilizaremos o mesmo para prever todos os 119.052 valores não observados de Y, possibilitando melhor caracterizar a invasão de ervas daninhas e fornecer uma ferramenta poderosa para selecionar em quais locais a aplicação de herbicida ou remoção mecânica é necessária.

Além disso, o mesmo modelo pode ser usado para fazer previsões em novas plantações de cana-de-açúcar, fazendo com que a nova coleta de dados no campo não seja mais necessária. Note que para aplicar tal técnica em outra cultura, será necessário reajustar o modelo de classificação.

Na literatura existem diversos modelos estatísticos capazes de serem utilizados para descrever o comportamento de dados categóricos (Y nesse caso, com três níveis), sendo eles: árvores de decisão (SAFAVIAN e LANDGREBE, 1991), redes neurais artificiais (ZURADA, 1992), máquinas de vetores de suporte (HSU e LIN, 2002), modelos aditivos generalizados para locação, forma e escala (GAMLSS) (STASINOPOULOS e RIGBY, 2007), entre outros. Tais modelos serão detalhados com maiores detalhes nas próximas seções.

Observa-se que ambos os modelos mencionados acima pertencem a uma gama de modelos contidos na área denominada inteligência artificial. Para propor o melhor modelo, os mesmos serão construídos, com ajuda do software R (TEAM, 2020) e, com base em algumas técnicas de seleção de modelos, as quais serão descritas nas seções posteriormente, o melhor modelo será selecionado

3.2 LINGUAGEM R

Grande parte das empresas no cenário mundial, produzem materiais com um enorme volume de dados. Em tempos que as atividades matemáticas, utilizam um elevado número de dados para a obtenção de resultados mais expressivos, conhecimentos estatísticos e práticas computacionais são requisitados com a finalidade de facilitar a compreensão das informações recebidas (MONTGOMERY e RUNGER, 2003).

A linguagem de programação R, nos fornece cálculos estatísticos e gráficos, os quais estão pré-programados em funções disponibilizadas em bibliotecas. Sua obtenção é de forma gratuita e sua utilização é versátil, de fácil entendimento e que auxilia a resolução de problemas de investigação. Está disponível sob a licença GNU GPL. Teve suas origens em meados de 1996, desenvolvido pelos professores de estatística Ross Ihaka e Robert Gentleman, oriundos da Universidade de Auckland, Nova Zelândia. Foram 5 anos de pesquisas para o desenvolvimento da linguagem, visto que os mesmos não tinham muita experiência na área computacional (MARTINS, 2010).

O software R é dotado de uma gama de técnicas estatísticas que auxiliam na leitura e análise dos dados, sendo elas a modelagem linear e não-linear, testes estatísticos clássicos, análises de séries temporais, dentre outros. Suas versões binárias estão disponíveis para os sistemas operacionais Windows, Macintosh, Unix/Linux. Similar ao pacote S-Plus, o mesmo possui expansões, com o uso de pacotes, que são expansões que possuem funções específicas para cada área de estudo, proporcionando ao usuário, caso necessite, a criação de funcionalidades ao programa caso não existam. Também faz comunicação com outros softwares, como o (Excel, OpenBugs e Statistical Analysis System). Atualmente, grandes empresas utilizam o R, como por exemplo, a Google, a Oracle, dentre outros.

Para a reprodução dessa pesquisa, o primeiro passo é realizar o *download* do software R, o qual encontra-se disponível no endereço https://www.r-project.org/. Após, para facilitar a escrita dos códigos, utilizaremos o editor de texto do R, chamado de R-Studio, o qual também é de acesso livre e pode ser baixado no endereço https://www.rstudio.com/.

3.3 MODELOS DE CLASSIFICAÇÃO

Nesta subseção são apresentados os oito modelos de classificação utilizados na pesquisa.

3.3.1 GAMLSS

Os modelos aditivos generalizados para locação, forma e escala, do inglês *Generalized Additive Models for Location, Scale and Shape (GAMLSS),* são modelos de regressão os quais permitem modelar todos os parâmetros do modelo de forma linear e não linear (com o uso de *splines*, por exemplo). Inicialmente esses modelos foram propostos por RIGBY and STASINOPOULOS (2005), os quais foram implementados no pacote *GAMLSS* no software R.

O princípio dos GAMLSS consiste em utilizar funções de ligação para cada um dos parâmetros do modelo probabilístico, garantindo que todos os resultados estão contidos no espaço paramétrico dos mesmos. Assim, dada uma função de probabilidade qualquer com quatro parâmetros, a qual iremos denotar por os GAMLSS são expressos da forma apresentada na Equação (1),

$$\theta = \begin{bmatrix} \boldsymbol{\mu} \\ \boldsymbol{\sigma} \\ \boldsymbol{\nu} \\ \boldsymbol{\tau} \end{bmatrix} = \begin{bmatrix} g_1\left(\boldsymbol{X_1\beta_1} + \sum_{j=1}^{J_1} h_{j1}\left(\boldsymbol{x_{j1}}\right)\right) \\ g_2\left(\boldsymbol{X_2\beta_2} + \sum_{j=1}^{J_2} h_{j1}\left(\boldsymbol{x_{j2}}\right)\right) \\ g_3\left(\boldsymbol{X_3\beta_3} + \sum_{j=1}^{J_3} h_{j1}\left(\boldsymbol{x_{j3}}\right)\right) \\ g_4\left(\boldsymbol{X_4\beta_4} + \sum_{j=1}^{J_4} h_{j1}\left(\boldsymbol{x_{j4}}\right)\right) \end{bmatrix}, \tag{1}$$

em que g_k para k=1,2,3,4, são funções de ligações apropriadas para cada parâmetro do modelo, $\boldsymbol{\beta_k}$ denota o vetor de parâmetros que deverão ser estimados e X_k representa a matriz do modelo. Todos os componentes apresentados até agora são relacionados a parte linear do modelo, os quais são somas, por exemplo.

A segunda parte dos termos aditivos, representa as funções não lineares não paramétricas, são funções suavizadoras em termos de variáveis explanatórias contínuas. Existem inúmeras funções suavizadoras que podem ser consideradas aqui, como por exemplo, *splines* cúbicos, lasso, entre outras. Nessa pesquisa iremos considerar como funções de suavização apenas P-*spines,* função essa introduzida por Eilers and Marx (1996). Os P-*splines*, também conhecido como *splines* penalizados, são funções polinomiais definidas com base de funções B-*splines,* as quais estão implementadas no pacote GAMLSS do software R, em que, por *default,* possuem 20 nós igualmente espaçados e possuem grau de polinômio igual a três, função essa baseada nas parametrizações definidas por Eilers and Marx (1996) e Eilers, Marx and Durbán (2015).

Como mencionado no início da seção, os GAMLSS são baseados em funções de probabilidade. No caso deste estudo, a variável resposta é uma variável categórica com 3 níveis de resposta, sendo então necessário utilizar um modelo probabilístico discreto. Para isso, o modelo que será utilizado é o multinomial com 3 níveis, o qual generaliza o modelo de regressão logístico. Basicamente, o modelo multinomial ajusta a razão entre as probabilidades dos níveis da variável resposta, escolhendo uns dos 3 fatores como fator base (por *default* o último nível, y=3=erva daninha). O modelo é expresso pela Equação 2

$$\frac{P(Y=1)}{P(Y=3)} = \mu \;\; e \quad \frac{P(Y=2)}{P(Y=3)} = \sigma, \tag{2}$$

em que μ representa a razão entre probabilidade da resposta ser igual a solo sobre a probabilidade de ser erva daninha e σ representa a razão entre as probabilidades de ser cana e erva daninha.

Dessa forma, o principal objetivo do modelo apresentado na Equação (2) é identificar e quantificar os fatores que estão associados as razões de probabilidade e tendo que quanto maior o valor de maior é a probabilidade do resultado ser solo e quando maior o valor de maior é a probabilidade do resultado ser cana-de-açúcar.

Note que ao relacionar o modelo probabilístico (2) com os modelos GAMLSS (1), o mesmo possui apenas 2 parâmetros, ou seja, a matriz apresentada em (1) será composta apenas pelas 2 primeiras linhas. Além disso, como os parâmetros do modelo são razões de probabilidade, isso significa que o espaço paramétrico (ou

seja, possíveis valores que os parâmetros podem assumir) será sempre positivo. Dessa forma, as funções de ligação devem garantir que os valores resultantes dos termos aditivos sejam sempre positivos. Nesse caso, as funções de ligação que utilizaremos serão as logarítmicas, resumindo o modelo na conforme Equação (3).

$$\theta = \begin{bmatrix} \boldsymbol{\mu} \\ \boldsymbol{\sigma} \end{bmatrix} = \begin{bmatrix} exp\left(\boldsymbol{X_1 \beta_1} + \sum_{j=1}^{J_1} h_{j1}(\boldsymbol{x_{j1}}) \right) \\ exp\left(\boldsymbol{X_2 \beta_2} + \sum_{j=1}^{J_2} h_{j1}(\boldsymbol{x_{j2}}) \right) \end{bmatrix} \quad (3)$$

Sabendo que a soma de todas as probabilidades é 1, é possível resolver o sistema de equações apresentado em (2), tendo como resultado

$$P(Y = 1) = \frac{\mu}{\mu+\sigma+1}\ ,\ P(Y = 2) = \frac{\sigma}{\mu+\sigma+1}\ e\ P(Y = 3) = \frac{1}{\mu+\sigma+1}. \quad (4)$$

Por fim, após ajustar o modelo, ou seja, obter os valores dos parâmetros em (3), ao combinar os resultados com (4), são obtidas as probabilidades de cada um dos possíveis níveis da variável resposta. O ajuste do modelo probabilístico é realizado internamente da função GAMLSS do *software* R, o qual utiliza o método de estimação Rigby and Stasinopoulos (RS), que também é considerado o default. Como exemplo, é apresentado abaixo os códigos em R exemplificando como instalar e carregar o pacote gamlss.

```
1. # GAMLSS
2. install.packages('gamlss')
3. library(gamlss)
```

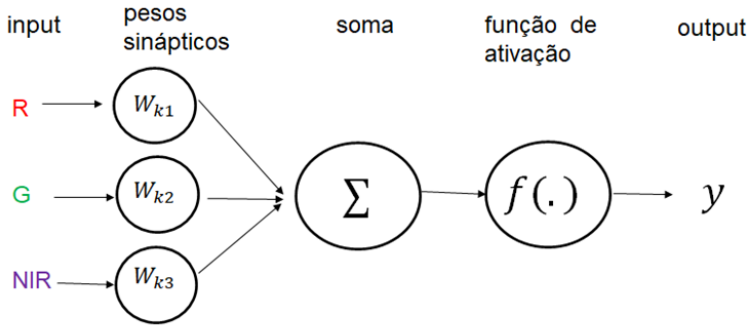
3.3.2 Redes Neurais

Algoritmos de Redes Neurais Artificiais são classificados como *Deep Learning* (aprendizado profundo), um ramo de aprendizado da máquina (machine learning) com elevadas capacidades de processamento (FRIEDMAN, 2001). Da mesma maneira em que um neurônio se desenvolve, as redes também se

manifestam, e é esse processo que se dá o nome de capacidade de reconhecimento de padrões.

Uma Rede Neural realiza o reconhecimento de uma sequência de esquemas por meio de uma sessão de treinamento. Sua arquitetura divide-se em três partes: camadas de entradas, camadas ocultas (hiddens) e camadas de saídas. Na primeira camada encontram-se os dados de entrada, nas quais estão dispostas as informações das variáveis preditoras. Nas camadas ocultas se processam as conexões, conhecidas por sinapse , e é o local onde estão as funções sigmóide e funções tangente hiperbólica, funções essas que captam as informações das variáveis preditoras, relacionando-as com a variável resposta. O resultado (variável resposta) fica na camada de saída. Para demonstrar como se comporta o modelo, a Figura 2 ilustra de maneira sucinta seu funcionamento.

FIGURA 2 - Desenho esquemático do funcionamento de uma Rede Neural.



FONTE: Autoria própria (2021).

Para uma sistemática resumida de cada neurônio *j* da camada oculta, temos a entrada equivalente a uma soma ponderada dada pela Equação (5),

$$\sum_{i=0}^{r} \gamma_{ji} x_i = \gamma_{j0} + \gamma_{j1} x_1 + \gamma_{j2} x_2 + \cdots + \gamma_{jr} x_r \tag{5}$$

em que são as informações das variáveis preditoras e os pesos relacionados à *i*-ésima variável preditora para a *j*-ésima observação, os quais serão estimados pelos modelos. O primeiro elemento da soma é considerado o "viés", que relaciona a existência de um neurônio aos demais, onde a rede com sinal de saída é igual a unidade . Assim sendo, logo depois da aplicação da função de ativação *f(.)*, obtém-se a saída de cada neurônio *j* da camada oculta, expressa pela Equação (6).

$$h_j = f\left(\sum_{i=1}^{r} \gamma_{ji} x_i\right) \tag{6}$$

Ao admitir a existência de q neurônios na camada oculta, obtem-se a entrada do neurônio da camada de saída expressa pela Equação (7).

$$\sum_{j=0}^{q} \beta_j h_j = \beta_0 + \beta_1 h_1 + \beta_2 h_2 + \cdots + \beta_q h_q \tag{7}$$

Por fim, o sinal de saída da Rede Neural é expressa pela Equação (8).

$$f\left[\sum_{j=0}^{q} \beta_j \cdot f\left(\sum_{j=0}^{r} \gamma_{ji} x_i\right)\right] \tag{8}$$

O treinamento de um modelo de Redes Neurais, utilizando o software R, pode ser feito utilizando o pacote *nnet* (RIPLEY, VENABLES e RIPLEY, 2016). Para a construção do modelo utilizando tal pacote, primeiramente deve-se dividir a variável resposta em 3 colunas, seguido da normalização das informações das variáveis preditoras (cores), passo esse importante para diminuir o custo computacional (tempo). Dessa forma, a seguinte linguagem de programação foi utilizada para instalar e carregar o pacote.

```
1. #Redes neurais
2. install.packages('nnet')
3. library(nnet)
```

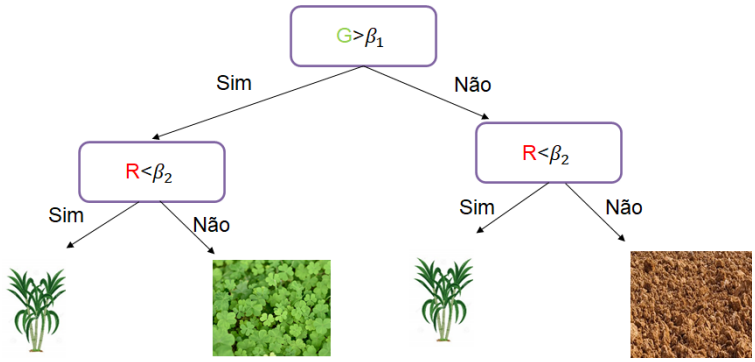
3.3.3 Árvores de Decisão

Decision Tree, ou árvore de decisão, também denominado atualmente como Classification Regression Trees (CART) (ressalta-se que CART também pode ser entendido como uma métodologia de Árvores de decisão), trata-se de uma ferramenta de classificação de ocorrências ou não de determinada irregularidade visando identificar as variáveis, a partir de um banco de dados de monitoramento. Esta técnica destaca-se na seleção de imagens, pois determina um aumento da acurácia e categoriza conjuntos de dados distintos (HANSEN et al., 1996; IM e JENSEN, 2005; HANSEN et al., 2008; GONG et al., 2011). Nesse contexto, a

utilização desse mecanismo possibilita melhores resultados entre variados produtos. Apresentam como dados de entrada algo semelhante ao binário e funciona particionando a informação recebida em subconjuntos.

A formação de uma árvore de decisão dá-se por nós, ramos e folhas. Os nós elencam regiões onde se localiza os testes lógicos de separação de dados e são divididos em nó raiz e nó principal. Os ramos são conexões entre os nós raiz e os nós filhos. As folhas são regiões em que se localizam os rótulos ou valores. É uma técnica que facilita a interpretação de resultados, pois de forma coesa, simplifica sua interpretação. A Figura 3 demonstra o comportamento de uma árvore de decisão.

FIGURA 3 - Desenho esquemático de uma Árvore de Decisão.



FONTE: Autoria própria (2021).

A principal vantagem das árvores de decisão é a rápida tomada de decisão (classificação), pois de uma forma ágil, identifica variáveis e as relações entre si. Por se tratar de uma metodologia não paramétrica, não há pressuposto à distribuição espacial, assim como à estrutura do classificador. Algumas das desvantagens de utilização seria o sobreajuste (*overfitting*), que possui solução através de definição de restrições sobre os dados do modelo.

As Árvores de Decisões têm sua especificidade por se tratar de árvores binárias. Uma das vantagens dessa metodologia é que as variáveis explicativas são uma combinação de variáveis nominais, contínuas e ordinais, muito utilizadas em estudos multidimensionais. Além disso, sua aplicação em adaptação aos dados faltosos sem a necessidade de acatar condições de aplicabilidade dos modelos paramétricos e a logaritmização facilitam seu uso.

Determina-se então o hiperparâmetro, que neste caso é denominado parâmetro de complexidade (c_p), o qual tem por finalidade evitar a divisão da partição, caso não se obtenha melhoras significativas da qualidade do modelo. O

parâmetro de complexidade determina uma correção à árvore, devido ao grande número de divisões. Assim sendo, tem-se um valor padrão de 0,01, ou seja, quanto maior, menor a árvore. Quando se tem um valor extremamente pequeno, tem-se que há uma superequiparação e, para um valor elevado, resultando em uma árvore pequeno, sendo que em ambos os casos há uma diminuição do desempenho preditivo do modelo c_p.

Para a construção de um modelo Decision Tree no software R, utilizaremos o pacote *rpart* (THERNEAU, ATKINSON e RIPLEY, 2015). O código utilizado para a instalação do pacote, bem como carregar o mesmo, é apresentado abaixo.

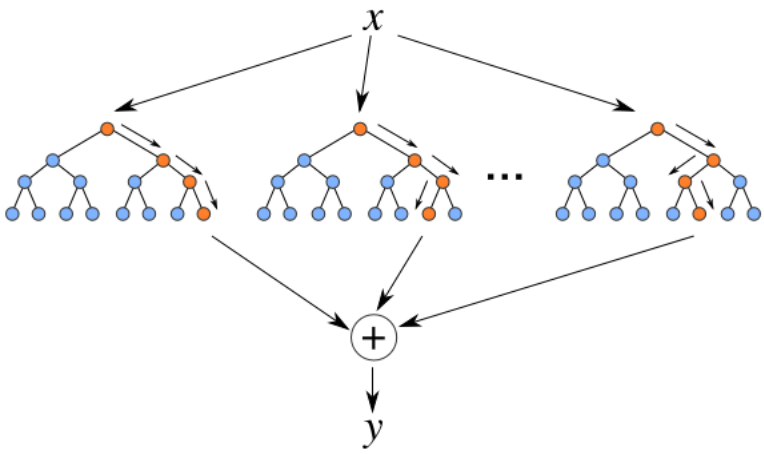
```
1. #Árvores de Decisão
2. install.packages('rpart')
3. library(rpart)
```

3.3.4 Florestas Aleatórias

A partir de um vetor de entrada, o modelo de classificação Random Forest (florestas aleatórias), disponível no pacote *randomForest* (LIAW et al., 2002) no software R, realiza classificações, onde se analisa cada árvore (tree) da floresta. Um dos benefícios no uso desse pacote é poder trabalhar com um grande conjunto de dados com maior dimensionalidade, assim como identificar as variáveis e sua importância.

O funcionamento de um Random Forest consiste no crescimento de variadas árvores, ao invés de apenas uma, como apresentado na Figura 4. Cada árvore dá uma classificação de votos para a classe, efetuando ao fim, uma média das saídas.

FIGURA 4 - Desenho esquemático de uma Floresta Aleatória.

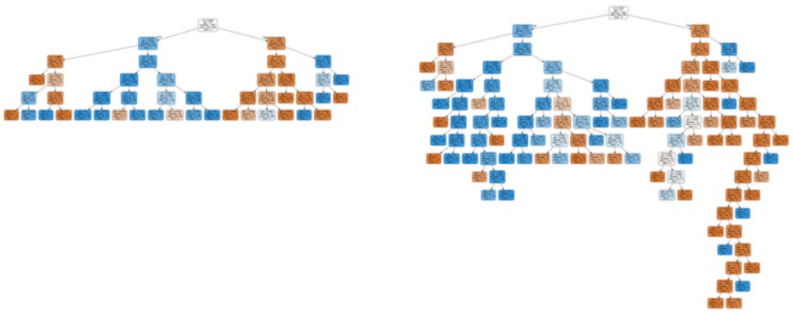


FONTE: Autoria Própria (2021).

Algumas das vantagens da utilização dessa técnica estão ligadas diretamente à resolução de problemas de classificação e regressão, realizando uma estimativa coesa nos dois casos, além de poder trabalhar com dados em volumes grandes, produzindo um grau de importância das variáveis. Além disso, a técnica é dotada de uma eficaz metodologia de estimação de dados faltantes com precisão e possui um equilíbrio de erros em conjuntos de dados.

Um hiperparâmetro encontrado nos modelos de florestas aleatórias é o número de árvores (B), o qual está relacionado diretamente ao valor da acurácia, ou seja, quanto maior B, maior a acurácia. Este aumento limita-se à uma quantidade finita, devido ao fato do quantitativo de árvores interferirem diretamente nos recursos computacionais, conforme Figura 5.

FIGURA 5 - Modelo esquemático da utilização do hiperparâmetro: (a) hiperparâmetro menor (b) hiperparâmetro maior.

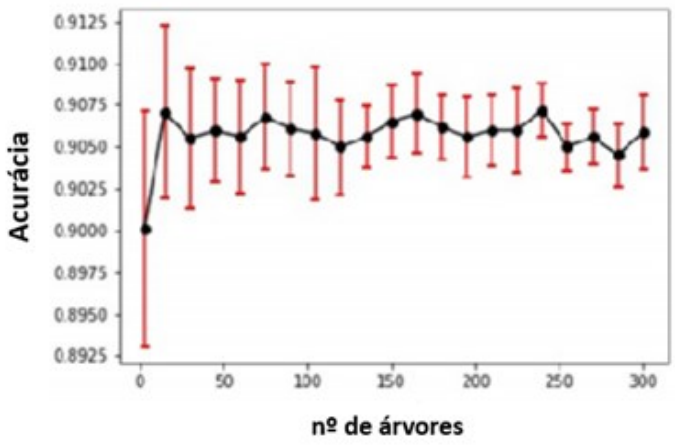


(a) (b)

FONTE: Autoria Própria (2021).

O hiperparâmetro visa controlar o grau de não relação entre as árvores da floresta aleatória, promovendo de certa maneira, melhorias na redução da variância através do método bagging, que ajusta a correlação entre as árvores, ocorrendo uma melhora na robustez (CRIMINISI e SHOTTON, 2013). Assim sendo, para um elevado número de árvores, onde algumas de certa forma demonstram pouca importância para a predição, há um pobre desempenho na floresta aleatória (Friedman et al., 2001), devido a probabilidade baixa de ocorrer um split (bom atributo a ser elencado). A Figura 6 ilusta a relação entre o número de árvores e a acurácia de uma floresta aleatória.

FIGURA 6 - Determinação da melhor acurácia relacionada ao número de árvores.



FONTE: Autoria Própria (2021).

A amostragem *bootstrap* é inserida nesse contexto, sendo que um terço dos dados não é utilizado para treinamento e sim para testes. Assim, ao utilizar esse procedimento há como diminuir o fator devido a alta sensibilidade das árvores de decisão, ou seja, alterações ínfimas no *dataset* geram mudanças no modelo. Metodologias de seleção de amostras, como por exemplo a mencionada *bootstrap,* serão descritas nas seções futuras.

Uma das desvantagens dos *Random Forest*, no que tange ao excelente trabalho de classificação, é que há um déficit de sua excelência aos problemas de regressão. Isso se dá pelo fato do não fornecimento de previsões exatas para variáveis contínuas. Para o problema em questão, essa desvantagem não se aplica, pois trata-se de uma classificação de uma variável resposta com 3 níveis. O código

utilizado para a instalação do pacote, bem como carregar o mesmo, é apresentado abaixo.

```
1. #Random Forest
2. install.packages('randomForest')
3. library(randomForest)
```

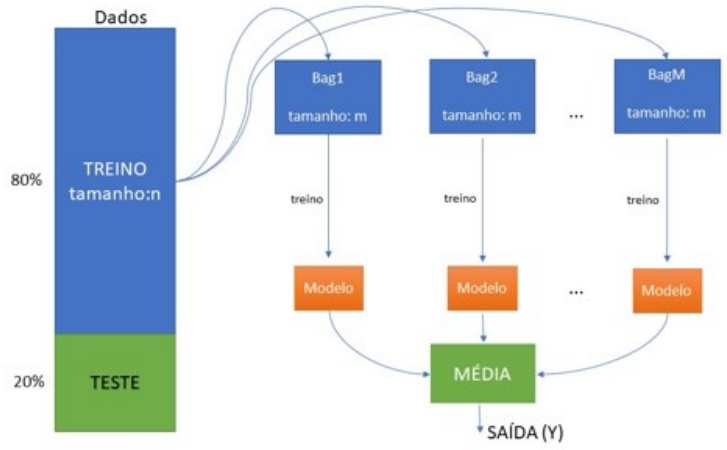
3.3.5 Bagging Cart

Bagging Cart é uma metodologia que visa a obtenção de novas amostras, sendo utilizada apenas em um conjunto de dados/amostras de treinamento que tem por finalidade reduzir a variância do modelo e evitar a sobreparametrização. Assim sendo, consiste na combinação de duas técnicas, o *bootstrap* e a agregação, partindo do pressuposto de combinação de inúmeros modelos preditivos em apenas um. Nota-se ainda que sua utilização se dá em variados algoritmos além do CART, sendo para classificação e para regressão, porém devido a sua acurácia, sua significância se dará apenas em modelos instáveis (BREIMAN, 1996).

Trata-se um método controlado, onde os dados de treinamento são subordinados ao algoritmo com as classes verdadeiras de cada amostra. Durante a etapa de teste, nota-se o fornecimento de novos dados com comparativo dos resultados às classes do novo conjunto.

O funcionamento de um Bagging Cart, ilustrado na Figura 7, mostra uma combinação entre Bags para formação de uma média das predições entre os modelos. Para isso, cria-se *m* amostras *bootstrap* dos dados de treinamento, onde tem-se que as mesmas são distintas, porém possuem a mesma distruibuição. Em seguida, é dado que para cada amostra obtida, um modelo de árvore sem poda é construído e, por fim, efetua-se a média das predições entre os modelos construídos. Esta média de variadas árvores faz com que se tenha uma redução da variabilidade, melhorando o desempenho preditivo.

FIGURA 7 - Desenho esquemático do sistema Bagging Cart.



FONTE: Autoria Própria (2021).

O processo ilustrado anteriormente na Figura 6, é também utilizado no modelo Bagging Cart, porém agora para selecionar o número de baggs. Assim sendo, uma amostra *bootstrap* trabalha com 2/3 dos dados para treinamento e 1/3 dos dados, a qual denota-se por *out-of-bag*, utiliza-se para estimação da precisão do modelo, ou seja um processo de validação cruzada.

Partindo de uma visão computacional, o algorítimo utiliza de comandos recursivos (*loop*), através de critérios de pausas, o número de amostras, bootstraps e iterações pré determinada e reduzidno a variância de acordo com o aumento de número de árvores.

O hiperparâmetro utilizado no modelo é o nbagg. Este hiperparâmetro refere-se a quantidade de bagging utilizado, através de validação cruzada. Conforme ilustra a Figura 6, do método Florestas aleatórias, no modelo Bagging Cart há um default padrão de 25 bootstrap, além disso é notório que para a determinação de um valor com melhor acurácia, o número de bagging deve ser o menor valor com maior acurácia. O código utilizado para a instalação do pacote, bem como carregar o mesmo, é apresentado abaixo.

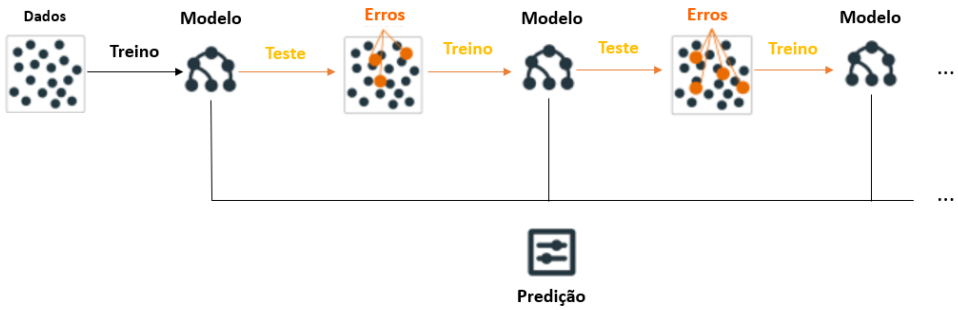
```
1. #BAGGING CART
2. install.packages('ipred')
3. library(ipred)
```

3.3.6 Gradient Boosting Machine (GBM)

GBM é um método baseado em combinações de conjuntos com a finalidade de oferecer um desempenho preditivo superior e preciso. Gong et al. (2017) dizem que se trata de uma técnica muito utilizada para construção de modelos preditivos. O procedimento se adapta sequencialmente aos novos modelos e aos dados inseridos, com a finalidade de aumentar a ênfase nas observações modeladas de maneira errônea através das “Decision Tree”, ou seja, é um modelo baseado em árvores de decisões. O melhoramento pode ser obtido através da construção de novos modelos de base. Assim como alguns algorítimos baseados em árvores de decisão, o Gradient Boosting Machine tem como principal característica a sua facilidade em lidar com variáveis heterogêneas.

Durante a etapa de treimaneto do modelo, utiliza-se o pacote Caret, que tem como objetivo trabalhar com uma função de treinamento de maneira que selcione o conjunto de parâmetros, através da validação cruzada, apresentando um melhor desempenho preditivo e um menor erro na validação, sendo este realizado para cada variável do conjunto de saída (RITT, 2017). O processo esquemático do funcionamento do Gradient Boosting Machine é ilustrado na Figura 8.

FIGURA 8 - Desenho esquemático de um Gradient Boosting Machine.



FONTE: Autoria Própria (2021).

Um dos mais importantes hiperparâmetros para o GBM é o número de árvores de decisão, utilizadas no presente trabalho. As árvores de decisão são adicionadas ao modelo de maneira sequencial com o intuito da correção e mellhora das previsões efetuadas por árvores antecessoras. Assim sendo, quanto maior o número de árvores, melhor é a predição. Porém, deve-se ter um equilíbrio com a taxa de aprendizagem, ou seja, quanto maior o número de árvores, menor a taxa de aprendizagem. Alguns outros hiperparâmetros também desempenham efeitos no

desempenho do modelo, dos quais cita-se a taxa de aprendizagem, a variância do modelo controlado através do tamanho da amostra utilizada para treinamento, dentre outros apresentados abaixo.

- *n.trees* - número total de árvores para ajuste, ou seja, o número de iterações e funções.
- *Skrinkage* – também conhecido como taxa de aprendizagem, trata-se de um parâmetro de redução aplicado a cada árvore.
- *interaction.depth* – profundidade máxima das iterações.
- *n.minobsinnode* – número real de observações.
- *bag fraction* – trata-se da fração das observações do conjunto de treinamento selecionadas aleatoriamente com a finalidade de propor uma nova árvore, introduzindo aleatoriedade no ajuste do modelo.

Em síntese, trata-se de um algoritmo de conversão de aprendizagem fraca em aprendizagem forte, ou seja, seleciona a função objetivo e utiliza um modelo de aprendizagem fraca para que ocorra a minimização da perda. O *Boosting* combina a aprendizagem fraca (ou base de aprendizagem) em uma regra forte, efetuando um melhoramento no poder de predição do modelo. O código utilizado para a instalação do pacote, bem como carregar o mesmo, é apresentado abaixo.

```
4. #GRADIENT BOOSTING MACHINE
5. install.packages('gbm')
6. library(gbm)
```

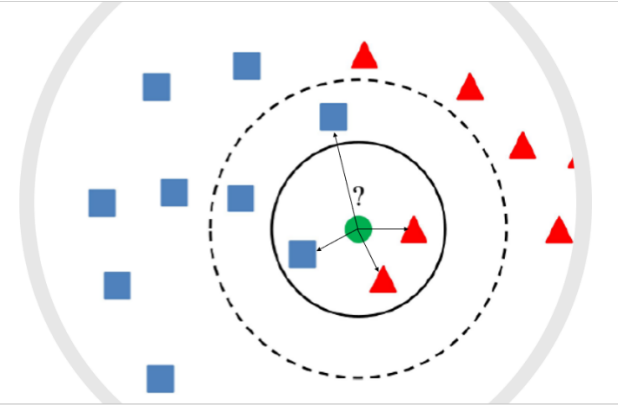
3.3.7 k-Nearest Neighhbors

K-Nearest Neighbors (K vizinhos próximos) é um método que permite a obtenção de valores considerados importantes para todas as coordenadas da matriz do georreferenciamento. A metodologia se baseia na área analisada, onde há uma imagem multiespectral com resolução semelhante à matriz utilizada nos dados amostrais. Cita-se como uma das vantagens, a fácil implementação e a natureza não-paramétrica (BISHOP, 2006).

Esse modelo consiste na classificação de exemplos, onde para cada observação a ser classificada, escolhe-se no conjunto de treino os k mais próximos,

conforme ilustrado na Figura 9. Por se tratar de um modelo não-paramétrico, o método é totalmente dependente do número de exemplos comparativos disponíveis.

FIGURA 9 - Desenho esquemático do modelo K-Nearest Neighbors.



FONTE: Autoria Própria (2021).

Para a determinação do K, ou seja, a quantidade de vizinhos que será utilizado, o processo é similar ao apresentando na Figura 6, ou seja, varia-se o tamanho de k, e determina a acurácia do modelo, determinando assim, a quantidade de vizinhos mais próximos que resulta numa maior acurácia. O código utilizado para a instalação do pacote, bem como carregar o mesmo, é apresentado abaixo.

```
1. # k-Nearest Neighbors
2. install.packages('caret')
3. library(caret)
```

3.3.8 C4.5

O modelo C4.5 (QUINLAN, 1993) apresenta métodos fundamentados em idéias de informações que são pautadas em hipóteses (AMARO, 2021), ou seja, o modelo adiquiri aprendizados relevantes para elaboração de uma árvore de decisão, oriundo de uma base de dados, apresentando aprendizados para a tomada de decisão (LORENZETT e TELÖCKEN, 2016). Esse método realiza a analise do melhor atributo em cada nó da arvore, determinando o nó que mais particiona as amostras em subconjuntos, os quais tendem para uma ou outra categoria. Para a classificação das testagens nos subconjuntos, faz-se necessário o cálculo do critério de seleção de dados (CARVALHO, 2014).

Em resumo, o método C4.5 é bem semelhante ao método CART, porém a diferença mais notável entre os dois é que o método CART constrói ávores baseado em um critério numérico, enquanto o método C4.5 inclui uma etapa intermediária para tal construçõa, baseando-se no conjunto de regras, as quais seguem alguns conceitos:

- Todas as amostras pertencem a uma determinada categoria, ou seja, quando isto ocorre, o algorítimo cria um nó folha para a árvore de decisão, escolhendo assim a categoria.
- Quando as características não fornecem informação, o modelo cria um nó de decisão utilizando o valor esperado.

Assim sendo, pode-se citar algumas vantagens do modelo C4.5 sobre outros sistemas de Árvore de decisão, os quais citam-se:

- Emprego específico do porcesso de poda para que se tenha a redução do sobreajuste;
- Consegue lidar facilmente com problemas de dados incompletos.

O código utilizado para a instalação do pacote, bem como carregar o mesmo, é apresentado abaixo.

```
1. # C 4.5
2. install.packages('RWeka')
3. library(RWeka)
```

3.4 PREDIÇÕES

Após a construção dos modelos de classificação, deseja-se utilizar os modelos para realizar previsões/predições. Para isso, os modelos agora dependem apenas das informações contidas nas variáveis preditoras, ou seja, no nosso caso as informações dos níveis de cores.

Uma das utilidades de realizar predições, é a construção de estatísticas que forneçam o nível de adequabilidade do modelo ajustado, ou seja, confrontar os valores observados com os valores preditos pelo modelo. Caso ambos valores sejam iguais, dizemos que o modelo fez a predição correta, possibilitando então determinar o nível de acurácia e especificidade dos modelos ajustados.

Também é possível utilizar os valores preditos pelo modelo para verificar se algum dos níveis da variável resposta não está sendo bem explicado pelas variáveis explicativas, ou seja, os níveis de cores para o caso em estudo. Isso possibilita que novas variáveis sejam levadas em consideração no estudo, como por exemplo, novos níveis de cores, combinações entre tais níveis, etc.

Por fim, como objetivo principal deste estudo, podemos realizar previsões, ou seja, determinar os valores da variável resposta (terra, cana-de-açúcar ou erva daninha), baseando-se apenas das informações dos níveis de cores obtidos com as imagens. Tal previsão é de suma importância, pois permitirá o uso de herbicidas nos locais exatos infestados por ervas daninhas, fazendo com que o uso do mesmo seja apenas o necessário, assim como aumentando a produtividade no campo e, consequentemente, diminuindo o custo do plantio.

No software R, após construídos os modelos de classificação, utiliza-se as funções *predic* ou *comput,* dependendo de qual modelo está sendo utilizado. Tais funções serão apresentadas na seção de Resultados, os quais serão detalhados todos os algoritmos.

3.5 SELEÇÃO DE MODELOS

3.5.1 Validação Cruzada

Trata-se de uma técnica que tem por finalidade a validação de modelos a partir de um conjunto de dados utilizados para estimar variados parâmetros que acompanham a evolução em curvas que correspondem aos dados inseridos (HAYKIN, 2007).

A principal idéia do método de validação cruzada é separar a base de dados completos em duas partes, sendo que uma será utilizada para o treinamento dos modelos, sendo então esses dados conhecidos como *training data,* e o restante dos dados sendo utilizados para realizar a verificação da adequação do modelo proposto, base de dados conhecida como *testing data,* pois dessa forma as informações fornecidas para a construção do modelo não estarão contidas na base de dados de testes, ou seja, o teste não será viciado.

Existem diversas formas de particionar a base de dados original nas bases de dados *training data* e *testing data* (BROWNE, 2000), sendo que a mais conhecida

é utilizar 2/3 dos dados para “treinar”/ajustar os modelos, e o restante, 1/3, utilizado para testar a adequação dos modelos ajustados. Esse método também é conhecido na literatura como *data splitting,* o qual tem como objetivo não apresentar erros muito otimistas obtidos do modelo, caso esse que se observa quando não utilizado dados de teste e treino. Ressalta-se aqui que a escolha das observações que irão comportar as duas bases de dados para a validação cruzada são selecionadas de forma aleatória.

Para essa pesquisa observa-se que a variável resposta possui 3 níveis de respostas, terra, cana-de-açúcar e erva daninha. Dessa forma, os dados de treinamento (2/3) foram selecionados aleatoriamente de tal forma que a composição dos dados de treinamento seja composta por 2/3 de cada um dos diferentes níveis.

3.5.2 K- Folds Cross Validation

O método de validação cruzada (*data splitting*) mencionado na seção anterior possui algumas falhas, como por exemplo, pode ocorrer que informações importantes para a construção do modelo sejam alocadas apenas na base de dados de teste. Dessa forma, uma evolução desta metodologia seria utilizar o método de K-folds.

Por sua vez, o método K-Folds de validação cruzada consiste na divisão da base de dados em k partes iguais, utilizando k-1 partes para treinamento e a parte restante para teste, realizando o procedimento por k vezes e produzindo k resultados experimentais independentes para avaliações estatísticas. Para cada vez, realiza-se uma testagem do modelo com um fold (conjunto de treino) distinto, sendo calculado a métrica para avaliação do modelo, e a partir daí o cálculo da média e desvio padrão (WITTEN e FRANK, 2002). Resumindo, efetua-se uma divisão dos dados em treinamento e teste, k vezes, utilizando como medidas finais, a média das medidas (acuracia, por exemplo) modelos obtidos nas *k*-ésimas etapas.

3.5.3 Medidas de qualidade de ajuste

Por meio da elaboração de uma matriz de confundimento, de acordo com Tabela 1, tem-se que as linhas são os valores classificados, as colunas são os valores observados, a diagonal são os acertos e: A_E : Quantitativo de acertos de

erva daninha; A_S : Quantitativo de acertos de solo; A_C : Quantitativo de acertos de cana; E_{E-S}: Quantitativo de erros classificados como erva e observados como solo; E_{E-C}: Quantitativo de erros classificados como erva e observados como cana; E_{S-E}: Quantitativo de erros classificados como solo e obesrvados como erva; E_{S-C}: Quantitativo de erros classificados como solo e observados como cana; E_{C-E}: Quantitativo de erros classificados como cana e observados como erva; E_{C-S}: Quantitativo de erros classificados como cana e observados como solo.

TABELA 1 - Matriz confundimento para as medidas de ajuste

CLASSIFICAÇÃO\OBSERVADO	**ERVA**	**SOLO**	**CANA**
ERVA	A_E	E_{E-S}	E_{E-C}
SOLO	E_{S-E}	A_S	E_{S-C}
CANA	E_{C-E}	E_{C-S}	A_C

FONTE: Autoria Própria (2021).

Existem diferentes técnicas para determinar a qualidade de um modelo de classificação, as quais são baseadas nas predições feitas pelos modelos utilizando os dados de teste. Uma delas consiste em determinar a acurácia do modelo, a qual é dada pela Equação 9.

$$Acurácia = \frac{ACERTO}{TOTAL} \tag{9}$$

Outra medida que pode ser utilizada, a qual é adaptada para o caso de uma variável resposta com 3 níveis, é precisão do modelo. A precisão é interpretada como uma qualidade do estimador para cada nível das classes. Essa medida pode ser construída para cada um dos níveis da variável resposta, sendo expressa pela Equação 10.

$$Precisão = \frac{ACERTO}{TOTAL\ CLASSIFICADO} \tag{10}$$

Além da medidade de precisão, pode-se também calcular a medida *recall.* Essa medida fornece com que frequência o classificador está classificando

exemplos de uma classe. Enquanto a precisão pode ser interpretada como uma medida de qualidade/exatidão, a medida *recall* é uma medida de integridade/quantidade. Essa medida é dada pela Equação 11.

$$Recall = \frac{ACERTO}{TOTAL\ OBSERVADO} \tag{11}$$

Considerando os diferentes métodos de validação cruzada apresentados nas subseções anteriores, tais medidas de qualidade de ajuste precisam levar em consideração o número de predições (validações) realizadas. Assim sendo, o presente trabalho elenca as medidas de acurácia, precisão e recall, conforme apresentado na Tabela 2, relacionado aos dados da Tabela 1 da matriz de confundimento.

TABELA 2 - Fórmulas de acurácia, precisão e recall das medidas de ajustes

Classes\Medidas	Acurácia	Precisão	*Recall*
Classe 1 (solo)	$\boldsymbol{Acurácia = \frac{ACERTO}{TOTAL}}$	$\boldsymbol{Precisão_S = \frac{A_S}{A_S + E_{S-E} + E_{S-C}}}$;	$\boldsymbol{Recall_S = \frac{A_S}{A_E + E_{E-S} + E_{C-S}}}$
Classe 2 (cana)		$\boldsymbol{Precisão_C = \frac{A_C}{A_C + E_{C-E} + E_{C-S}}}$	$\boldsymbol{Recall_C = \frac{A_C}{A_E + E_{E-C} + E_{S-C}}}$
Classe 3 (erva)		$\boldsymbol{Precisão_E = \frac{A_E}{A_E + E_{E-S} + E_{E-C}}}$	$\boldsymbol{Recall_E = \frac{A_E}{A_E + E_{S-E} + E_{C-E}}}$

FONTE: Autoria Própria (2021).

Levando em consideração as Equações (10) e (11), referentes à precisão e recall, respectivamente, observa-se que: Uma precisão baixa para erva daninha, significa que ervas daninhas foram classificadas como solo e cana, implicando em uma possível alastração no plantio devido à não aplicação do herbicida; Um recall baixo para erva daninha indica que tanto solo quanto cana foram classificados como erva daninha, fazendo com que o herbicida seja aplicado em local errado, tendo como consequência o aumento do custo. Na prática, como não é possível determinar o prejuízo causado pela erva daninha no campo nessa situação, tão pouco determinar o custo do uso errôneo de herbicida, tais fatores não serão utilizados para determinar se precisão ou recall é a melhor medida. Ressalta-se que em algumas situações tais valores podem ser obtidos, por isso iremos considerar nesse trabalho ambas medidas. Por outro lado, levando em consideração aspectos

ambientais, ou seja, uma redução da aplicação de herbicidas no campo, a precisão se torna a medida ideal para a escolha dos melhores modelos.

Além das medidas especificadas acima, outro fator que deve-se levar em consideração são os custos (tempos) computacionais. No presente trabalho, efetua-se a contabilização do tempo computacional para estimação dos hiperparâmetros, bem como o tempo para a construção do modelo, sendo a soma destes o custo computacional total de cada modelo.

3.5.4 Seleção de Hiperparâmetros

Hiperparâmetros são variáveis que controlam diversas funções dos modelos, os quais necessitam de uma definição prévia desses valor antes da realização do treinamento e, de certa maneira, definem características relacionadas a complexidade do modelo, evitando problemas como super ajustes (*orverfitting)*. Sendo assim, a estimação do valor otimizado para os mesmos torna-se de suma importância para que se obtenha um melhor desempenho de predição e menor custo computacional (BISHOP, 2006).

No presente trabalho, para selecionar os hiperparâmetros, utilizou-se de validação cruzada, via k-fold, utilizando k=5 como o número de folds. Tais parâmetros foram selecionados de forma a maximizar a acurácia dos modelos.

4 RESULTADOS E DISCUSSÃO

Todas as análises que serão apresentadas nas próximas subseções foram conduzidas utilizando as seguintes configurações de softwares e máquina: R Version 4.0.3 (2020-10-10) – ‘’Bunny-wunnies Freak Out’’ Plataform: x86_64-w64-mingw32/x64 (64-bit), computador Dell Inspiron 15 5000, Intel Core i7 – 10510U (4.9 GHz, cache 8 MB, quad-core, 10ª geração), Windows 10 Home Single Language, 64 bits, memória 16 GB, DDR4, 2666 MHz, SSD de 256 GB.

4.1 DADOS

Primeiramente, para organizar a base de dados, o primeiro passo foi transformar a unidade de medida das coordenadas, que originalmente estão em UTM, para latitude e longitude (lat, long). Para isso, utilizou-se o pacote *rgdal (*BIVAND *et al.*, 2020) do software R. Os códigos utilizados foram:

```
####Dados totais####
dados <- read.csv("SQ_Extract2.csv", sep=";")
#Convertento em long. e lat.
library(rgdal)
utm1 <- data.frame(x=dados$Coord_X,y=dados$Coord_Y)
coordinates(utm1) <- ~x+y
class(utm1)
#zona = 23  local da amostra
proj4string(utm1) <-
 CRS("+proj=utm +zone=23 +datum=WGS84 +units=m +ellps=WGS84 +south")
utm2<-spTransform(utm1,CRS("+proj=longlat +datum=WGS84")) #long lati
head(utm2)
dados$long=utm2$x
dados$lat=utm2$y
```

Antes de iniciar as contruções dos modelos de classificação, iremos dividir a base de dados completa em 10 pedaços (folds) aleatórios, para conduzirmos as técnicas de validação cruzada via método k-fold (com k=10). Para isso, o pacote auxiliar utilizado nessa seção foi o cvTools (ALFONS, 2012), o qual divide a base de dados aleatóriamente em k grupos determinados. Para evitar grupos com poucas respostas de uma determinada classe (erva, cana ou solo), a base de dados foi dividia em 3 novos dados, uma contendo apenas respostas de erva-daninha, outra apenas cana e a última apenas com dados de solo. Após essa divisão, o método k-fold foi aplicado a cada uma dessas bases de dados, garantindo a proporcionalidade das classes no banco de dados final para treino e teste, os quais são formados agrupando os folds obtidos de cada um dos 3 conjuntos de dados. Os códigos utilizandos nessa seção estão dispóstos abaixo. Note que esses códigos serão utilizados para todos os modelos que serão apresentados nas próximas subseções.

```
####Dados com resposta####
dados1<-dados[1:8801,]
head(dados1)
names(dados1)<-c("NIR","RE",'R','G',"Class",'Coord_X',"Coord_Y")

dadossolo=subset(dados1,Class==1)#### Dados solo
dadoscana=subset(dados1,Class==2)#### Dados cana
dadoserva=subset(dados1,Class==3)#### Dados erva
```

```
9.
10.library(cvTools) #biblioteca para k-fold
11.k <- 10 #numero de folds
12.
13.folds1 <- cvFolds(NROW(dadossolo), K=k)#folds solo
14.folds2 <- cvFolds(NROW(dadoscana), K=k)#folds cana
15.folds3 <- cvFolds(NROW(dadoserva), K=k)#folds erva
```

Após a seleção dos 10 folds para construção dos modelos, também iremos realizar a construção dos 5 folds para os modelos que necessitem realizar a etapa da seleção dos hiperparâmetros, conforme os códigos apresentados abaixo. Ressalta-se então que os objetos *folds1, folds2* e *folds3* são os folds para construção do modelo e *foldscana, foldserva* e *foldssolo* são os folds (com k=5) utilizados para estimação dos hiperparâmetros.

```
1. #Hiperparameter selection
2. k=5 #the number of folds
3.
4. foldscana <- cvFolds(NROW(dadoscana), K=k) #folds para cada componente
5. foldscana
6.
7. foldserva <- cvFolds(NROW(dadoserva), K=k)
8. foldserva
9.
10.foldssolo <- cvFolds(NROW(dadossolo), K=k)
11.foldssolo
```

4.2 GAMLSS

Como já mencionado anteriormente, esses dados foram primeiramente analisados por Righetto *et al.* (2019). Ressalta-se que os autores utilizaram apenas o método de *data splitting* como validação cruzada, ou seja, foram selecionados 70% dos dados proporcionalmente por classes para montar o modelo, sendo os 30% restantes utilizados para verificar a acuracia do modelo. Observa-se que apenas de ser um método muito utilizado na literatura, esse possui alguns fatores negatios, como discutido na seção 3.5.2. Nota-se também que a medida de qualidade do modelo apresentado pelos autores foi a acuracia total, não levando em conta as medidas de precisa e *recall,* as quais podem ser extremamentes úteis ao analisar a resposta de ervas daninhas, tento em vista que essas podem se propagar, fazendo

com que o estudo das taxas de acerto dessa classe sejam mais importantes do que as demais classes.

Nesses sentido, com o objetivo de obter uma análise com resultados mais fidedignos, utilizaremos o modelo final apresentado pelos autores, porém toda a parte de estimação e validação cruzada será refeita utilizando o método k-fold (considerando k=10). Além das medidas de acuracia, precisão e *recall,* também foi considerado o tempo total de obtenção das tais estimativas. Por fim, como modelo final, foi utilizado a base de dados completa para realizar as predições das respostas incompletas da base de dados. O modelo final, disposto na Eq. (12) do trabalho apresentado por Righetto *et al.* (2019), é dado por

$$\mu = \exp(\beta_{01} + \beta_{11}NIR + \beta_{21}RE + \beta_{31}NIRxRE)$$
$$\sigma = \exp(\beta_{02} + \beta_{12}NIR + \beta_{22}RE + \beta_{32}G + \beta_{42}R + \beta_{52}RExG + \beta_{62}NIRxG) \quad (12)$$

Os resultados obtidos estão apresentados na Tabela 3. Observa-se que o tempo computacional gasto para realizar as estimativas dos 10 modelos (10-folds) foi de 2.64 minutos. Uma observação importante a ser mencionada aqui é que esse tipo de modelo não necessita do processo de seleção de hiperparâmetros. A média das acurácias obtidas para os 10 modelos foi de 0,969. Os valores das precisões médias, considerando as 3 classes (1=solo, 2=cana e 3=erva), foram de 0,999, 0,965 e 0,955, respectivamente. Já os valores médios da medida *recall*, também considerando as 3 classes, foram 0,999, 0,962 e 0,959, respectivamente.

TABELA 3 - Medidas de qualidade de ajuste obtidas nos 10-folds para o modelo GAMLSS

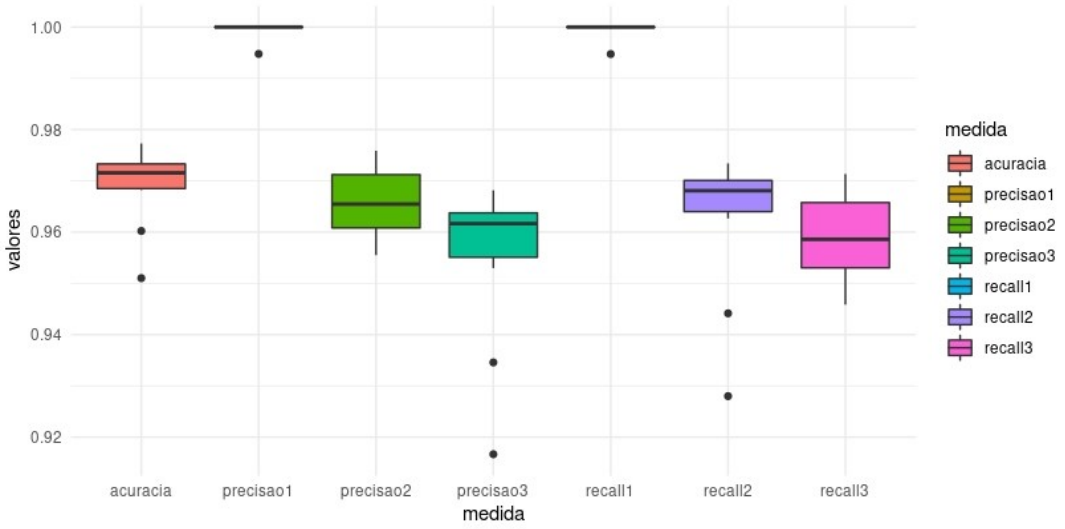
Classes\Medidas	Tempo (segundos)	Acurácia	Precisão	*Recall*
Classe 1 (solo)	158,400	0,969	0,999	0,999
Classe 2 (cana)			0,965	0,962
Classe 3 (erva)			0,955	0,959

FONTE: Autoria Própria (2021).

Na Figura 10 são apresentados os boxplots para tais resultados, nos quais podemos observar que: a medida de acuracia, como já esperado, possui uma

relação com a base de dados escolhida, justificando assim o método k-fold em relação ao método *data splitting;* a precisão para a erva-daninha foram maiores que 95% em quase 100% das vezes; o *recall* para a erva-daninha também foi maior que 95% em quase 100% das vezes. Em comparativo com os resultados apresentaods por Righetto *et al.* (2019), a acuracia total apresentada pelos autores também foi de 96,9%.

FIGURA 10 - Bloxplots das medidas de qualidade de ajuste obtidos nos 10-folds para o modelo GAMLSS.



FONTE: Autoria própria (2021).

Os códigos utlizados nessa seção estão dispóstos abaixo, ressaltando que esses sucedem os já apresentados na subseção anterior, utilizados para organizar as bases de dados em 10-folds.

```
1. library(gamlss);library(gamlss.dist)
2.
3. acuracia=precisao1=precisao2=precisao3=c()
4. recall1=recall2=recall3=c()  #objetos para os resultados finais
5.
6. tempoinicio=Sys.time() #tempo inicial
7. k=10
8. for(i in 1:k){ #cv k-fold
9. teste1=dadossolo[folds1$subsets[folds1$which==i],] #teste solo
10.treino1=dadossolo[folds1$subsets[folds1$which!=i],] #treino solo
11.
12.teste2=dadoscana[folds2$subsets[folds2$which==i],] #teste cana
13.treino2=dadoscana[folds2$subsets[folds2$which!=i],] #treino cana
14.
15.teste3=dadoserva[folds3$subsets[folds3$which==i],] #teste erva
```

```
treino3=dadoserva[folds3$subsets[folds3$which!=i],] #treino erva

dadosteste=rbind(teste1,teste2,teste3) #dados teste
dadostreino=rbind(treino1,treino2,treino3) #dados treino

#modelo gamlss
modelo<-gamlss(Class~NIR+RE+NIR*RE,sigma.fo=~NIR+RE+G+R+RE*G+NIR*G,
      data=dadostreino,family=MN3, n.cyc=1500)

#predicting
musigma<-predictAll(modelo,newdata=dadosteste, type='response')

py1=py2=py3=c()
for(j in 1:length(musigma$mu)){  #prob. de cada nivel
  py1<-c(py1,dMN3(1,musigma$mu[j],musigma$sigma[j]))
  py2<-c(py2,dMN3(2,musigma$mu[j],musigma$sigma[j]))
  py3<-c(py3,dMN3(3,musigma$mu[j],musigma$sigma[j]))
 }

mtx<-cbind(py1,py2,py3)  #juntando as probs em colunas
ynew<-c()
for(j in 1:length(py1)){
   m<-max(mtx[j,])  #maximo eh o valor predito
   ynew<-c(ynew,which(mtx[j,] %in% m))
   }

 confundimento=table(ynew,dadosteste$Class)#matriz confundimento
 acuracia=c(acuracia,sum(diag(confundimento))/sum(confundimento))

 precisao1=c(precisao1,confundimento[1,1]/sum(confundimento[1,]))
 precisao2=c(precisao2,confundimento[2,2]/sum(confundimento[2,]))
 precisao3=c(precisao3,confundimento[3,3]/sum(confundimento[3,]))

 recall1=c(recall1,confundimento[1,1]/sum(confundimento[,1]))
 recall2=c(recall2,confundimento[2,2]/sum(confundimento[,2]))
 recall3=c(recall3,confundimento[3,3]/sum(confundimento[,3]))
 } #fim k-fold
 tempofim=Sys.time() #tempo fim
 tempoinicio-tempofim #tempo total

#medidas
 medida=c(rep('acuracia',10),
          rep('precisao1',10),rep('precisao2',10),rep('precisao3',10),
          rep('recall1',10),rep('recall2',10),rep('recall3',10))
 valores=c(acuracia,precisao1,
           precisao2,precisao3,
           recall1,recall2,recall3)

 dadosmedidas=data.frame(medida,valores)
 library(ggplot2)
 ggplot(dadosmedidas,aes(medida,valores,fill=medida))+
```

```
          geom_boxplot()+theme_minimal()
tapply(dadosmedidas$valores, dadosmedidas$medida, mean)
```

4.3 REDES NEURAIS

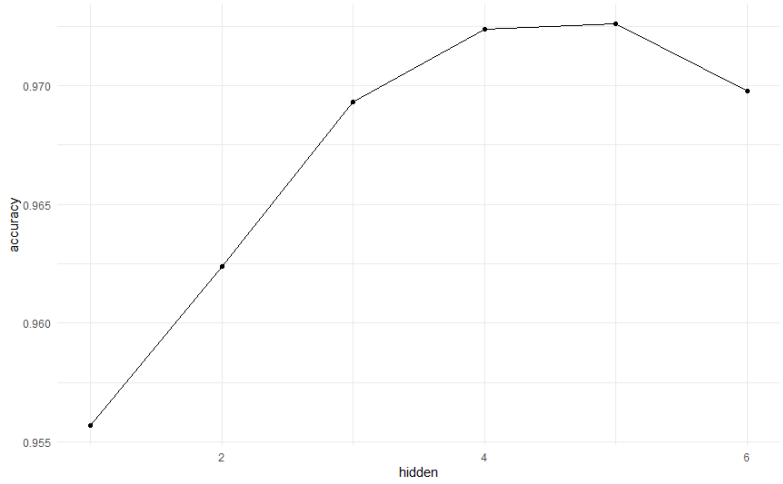
Para o modelo de redes reurais, inicialmente foi realizada a seleção do hiperparâmetro (número de hiddens ou número de neurônios). Utilizou-se então a quantidade mínima de 1 até 6 hiddens, para realizar a busca do melhor. Inicialmente foi considerado o máximo de 10 hiddens, porém o tempo computacional se deu muito elevado e, após 6 dias do algoritmo rodando, o mesmo foi pausado, diminuindo então o máximo para 6 hiddens. Os códigos utilizados para a obtenção do hiperparâmetro foram:

```
#Hiperparameter selection
h=6 #number of hidden maximum hiperparâmetro
accuracy_cv<-vector(length=k)  #acuracia nas k validacoes
accuracy<-vector(length=h) #armazenar acuracia

inicio=Sys.time()
k=5
for (j in 1:h) {print(c("hidden",j))
  for (i in 1:k) {print(c("fold",i))

  t1=dadoscana[foldscana$subsets[foldscana$which==i],]#teste
  tr1=dadoscana[foldscana$subsets[foldscana$which!=i],]#treino

  t2=dadoserva[foldserva$subsets[foldserva$which==i],]
  tr2=dadoserva[foldserva$subsets[foldserva$which!=i],]

  t3=dadossolo[foldssolo$subsets[foldssolo$which==i],]
  tr3=dadossolo[foldssolo$subsets[foldssolo$which!=i],]

  dadosteste=rbind(t1,t2,t3)
  dadostreino=rbind(tr1,tr2,tr3)
  dadostreino = cbind(dadostreino[, 1:4],
                      class.ind(as.factor(dadostreino$Class)))
  dadosteste = cbind(dadosteste[, 1:4],
                     class.ind(as.factor(dadosteste$Class)))

  #normalizando
  normalize <- function(x) {return ((x - min(x)) / (max(x) - min(x)))}
  dadostreino[,1:4]= as.data.frame(lapply(dadostreino[,1:4], normalize))
  dadosteste[,1:4] <- as.data.frame(lapply(dadosteste[,1:4], normalize))
  names(dadosteste) = c(names(dadosteste)[1:4],"l1","l2","l3")
  names(dadostreino) = c(names(dadostreino)[1:4],"l1","l2","l3")
```

```
  NN = neuralnet(l1+l2+l3~NIR+RE+R+G,dadostreino,
                 linear.output=F,hidden=j,stepmax = 1e+10)
prediction <-try(compute(NN, dadosteste[,1:4]),silent = F)

  acerto<-max.col(prediction$net.result[,1:3]) #maximo coluna (predicao)
  cm<-table(max.col(dadosteste[,5:7]),acerto) # matriz confundimento
  accuracy_cv[i]<-sum(diag(cm))/sum(cm) #acuracia
}
  accuracy[j]<-mean(accuracy_cv) #media das acuracias das 5 cv
}
fim=Sys.time()
fim-inicio #tempo final
```

Efetuado a seleção do hiperparâmetro a partir dos códigos apresentados acima, obteve-se as médias das acurácias para cada valor destes, conforme apresentado na Figura 11. Dessa forma, tem-se que o número otimizado de *hidden* foi igual a 4, ou seja, o valor que aumenta o poder de predição. Observa-se ainda nessa figura que para *hidden*=5, o aumento do poder de predição é insignificante quando comparado com *hidden*=4, porém o custo computacional aumenta expressivamente. O tempo computacional de realização do hiperparâmetro foi de 3.917,0 minutos, o que torna este modelo menos atrativo.

FIGURA 11 - Determinação do hiperparâmetro do modelo de Redes Neurais.



FONTE: Autoria própria (2021).

Após a seleção do hiperparâmetro como *hidden*=4, podemos agora iniciar a construção do modelo final, consideran fixando esse hiperparâmetro do algoritmo. Os códigos utlizados estão dispóstos abaixo, ressaltando que esses sucedem os já apresentados na subseção anterior, utilizados para organizar as bases de dados em 10-folds.

```
library(neuralnet) # pacote para construção do modelo
library(nnet) # pacote auxiliar para predição

acuracia=precisao1=precisao2=precisao3=c()
recall1=recall2=recall3=c()

tempoinicio=Sys.time() #tempo
k=10
for(i in 1:k){ #cv k-fold
  print(c("fold",i)) #saber qual fold
  teste1=dadossolo[folds1$subsets[folds1$which==i],] #teste solo
  treino1=dadossolo[folds1$subsets[folds1$which!=i],] #treino solo

  teste2=dadoscana[folds2$subsets[folds2$which==i],] #teste cana
  treino2=dadoscana[folds2$subsets[folds2$which!=i],] #treino cana

  teste3=dadoserva[folds3$subsets[folds3$which==i],] #teste erva
  treino3=dadoserva[folds3$subsets[folds3$which!=i],] #treino erva

  dadosteste=rbind(teste1,teste2,teste3)
  dadostreino=rbind(treino1,treino2,treino3)
  dadostreino = cbind(dadostreino[, 1:4],
                      class.ind(as.factor(dadostreino$Class)))
  dadosteste = cbind(dadosteste[, 1:4],
                      class.ind(as.factor(dadosteste$Class)))

  #normalizando
  normalize <- function(x) {return ((x - min(x)) / (max(x) - min(x)))}
 dadostreino[,1:4]=as.data.frame(lapply(dadostreino[,1:4], normalize))
  dadosteste[,1:4] <- as.data.frame(lapply(dadosteste[,1:4], normalize))
  names(dadosteste) = c(names(dadosteste)[1:4],"l1","l2","l3")
  names(dadostreino) = c(names(dadostreino)[1:4],"l1","l2","l3")
  NN = neuralnet(l1+l2+l3~NIR+RE+R+G,
                 dadostreino,linear.output=F,hidden=4,stepmax = 1e+10)

  #predicting
  ynew <-try(compute(NN, dadosteste[,1:4]),silent = F)

  acerto<-max.col(ynew$net.result[,1:3]) #maximo coluna (predicao)
 confundimento<-table(max.col(dadosteste[,5:7]),acerto)# confundimento

  acuracia=c(acuracia,sum(diag(confundimento))/sum(confundimento))
  precisao1=c(precisao1,confundimento[1,1]/sum(confundimento[1,]))
  precisao2=c(precisao2,confundimento[2,2]/sum(confundimento[2,]))
  precisao3=c(precisao3,confundimento[3,3]/sum(confundimento[3,]))

  recall1=c(recall1,confundimento[1,1]/sum(confundimento[,1]))
  recall2=c(recall2,confundimento[2,2]/sum(confundimento[,2]))
  recall3=c(recall3,confundimento[3,3]/sum(confundimento[,3]))
} #fim k-fold
tempofim=Sys.time() #tempo fim
```

```
tempofim-tempoinicio #tempo total

#medidas
medida=c(rep('acuracia',10),
         rep('precisao1',10),rep('precisao2',10),rep('precisao3',10),
         rep('recall1',10),rep('recall2',10),rep('recall3',10))
valores=c(acuracia,precisao1,precisao2,precisao3,
          recall1,recall2,recall3)

dadosmedidas=data.frame(medida,valores)
ggplot2::ggplot(dadosmedidas,aes(medida,valores,fill=medida))+geom_boxplot()+ theme_minimal()+ylim(0.92,1)
tapply(dadosmedidas$valores, dadosmedidas$medida, mean)
```

Os resultados obtidos estão apresentados na Tabela 4. Observa-se que o tempo computacional gasto para realizar as estimativas dos 10 modelos (10-folds) foi de 230 min. A média das acurácias obtidas para os 10 modelos foi de 0,966. Os valores das precisões médias, considerando as 3 classes (1=solo, 2=cana e 3=erva), foram de 0,999, 0,973 e 0,938, respectivamente. Já os valores médios da medida recall, também considerando as 3 classes, foram 0,998, 0,951 e 0,967, respectivamente.

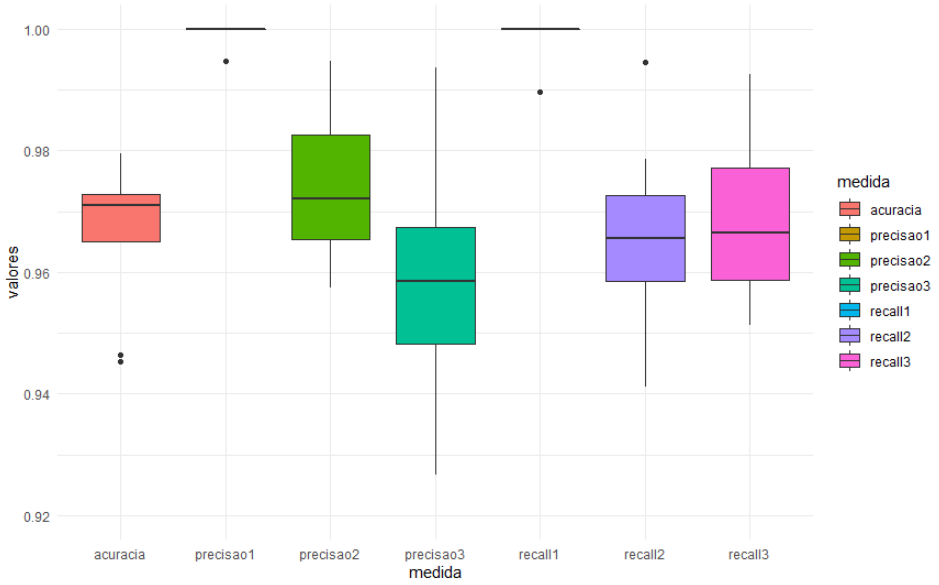
TABELA 4 - Medidas de qualidade de ajuste obtidas nos 10-folds para o modelo de Redes Neurais

Classes\Medidas	Tempo (segundos)	Acurácia	Precisão	*Recall*
Classe 1 (solo)	248.820,000	0,966	0,999	0,998
Classe 2 (cana)			0,973	0,951
Classe 3 (erva)			0,938	0,967

FONTE: Autoria Própria (2021).

Na Figura 12 são apresentados os boxplots para tais resultados, nos quais podemos observar que: há uma baixa variabilidade para predição de solo, a precisão de erva daninha variou de aproximadamente 92,5 a 99,5% e o recall de erva daninha variou de 95,2 a 99,2%.

FIGURA 12 - Bloxplots das medidas de qualidade de ajuste obtidos nos 10-folds para o modelo REDES NEURAIS.



FONTE: Autoria própria (2021).

4.4 ÁRVORES DE DECISÃO (CART)

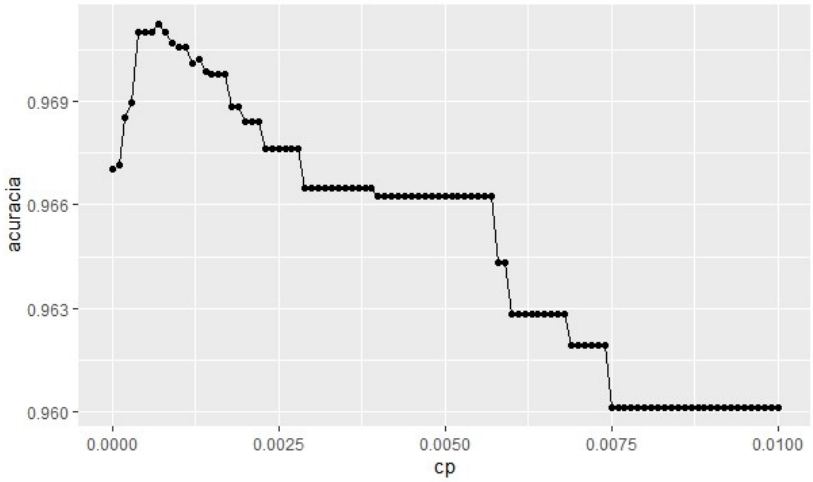
Para iniciar a seleção do modelo CART, inicialmente foi realizada a seleção do hiperparâmetro Cp, considerando o número de folds como k=5. Ressalta-se que é necessário utilizar os códigos apresentados na subseção 4.1 para proceder com essa analise. Os valores considerados para o parâmetro de complexidade foram de 0 a 0,01. Os códigos utilizados para esta seleção foram:

```
#Hiperparameter selection
library(rpart) #pacote modelo cart

acuracia=acuracia2=c()
tempoinicio=Sys.time() #tempo
k=5

for(c in seq(0,0.01,0.0001)){ #for para parametron de complexidade
for(i in 1:k){ #cv k-fold
  teste1=dadossolo[folds1$subsets[folds1$which==i],] #teste solo
  treino1=dadossolo[folds1$subsets[folds1$which!=i],] #treino solo

  teste2=dadoscana[folds2$subsets[folds2$which==i],] #teste cana
  treino2=dadoscana[folds2$subsets[folds2$which!=i],] #treino cana

  teste3=dadoserva[folds3$subsets[folds3$which==i],] #teste erva
  treino3=dadoserva[folds3$subsets[folds3$which!=i],] #treino erva
```

```
18.
19.   dadosteste=rbind(teste1,teste2,teste3) #dados teste
20.   dadostreino=rbind(treino1,treino2,treino3) #dados treino
21.
22.   #modelo cart
23.   modelo<-rpart(Class~NIR+RE+R+G,data=dadostreino,method = 'class',cp=c)
24.
25.   #predicting
26.   ynew<-predict(modelo,newdata=dadosteste, type='class')
27.
28.   confundimento=table(ynew,dadosteste$Class)#matriz confundimento
29.   acuracia=c(acuracia,sum(diag(confundimento))/sum(confundimento))
30. } #fim k-fold
31. acuracia2=c(acuracia2,mean(acuracia))
32. acuracia=c()
33. } #fim c
34.
35. tempofim=Sys.time() #tempo fim
36. ts=tempofim-tempoinicio #tempo total
37.
38. cp=seq(0,0.01,0.0001)
39. acuracia2
40. datacp=data.frame(acuracia=acuracia2,cp)
41. library(ggplot2)
42. ggplot(datacp,aes(cp,acuracia))+geom_point()+geom_line()
43.
```

A partir dos códigos apresentados acima, para a obtenção do hiperparâmetro, obteve-se as médias da acurácias para cada valor do hiperparâmetro Cp, apresentados na Figura 13. A figura abaixo ilustra que tem-se um número otimizado Cp de 0,0007, ou seja, o valor que onde ocorre um aumento no poder de predição. O tempo computacional de realização do hiperparâmetro foi de 21,728 segundos.

FIGURA 13 – Seleção de hiperparâmetro do modelo Árvores de Decisão.



FONTE: Autoria própria (2021).

Em seguida, após a seleção do hiperparâmetro, elaborou-se a construção do modelo final. Os códigos para a construção do modelo, considerando o método de validação cruzada k-fold, com k=10, são apresentados abaixo.

```
1. library(rpart) #pacote modelo cart
2.
3. acuracia=precisao1=precisao2=precisao3=c()
4. recall1=recall2=recall3=c()
5. k=10
6. tempoinicio=Sys.time() #tempo
7. for(i in 1:k){ #cv k-fold
8.   teste1=dadossolo[folds1$subsets[folds1$which==i],] #teste solo
9.   treino1=dadossolo[folds1$subsets[folds1$which!=i],] #treino solo
10.
11.  teste2=dadoscana[folds2$subsets[folds2$which==i],] #teste cana
12.  treino2=dadoscana[folds2$subsets[folds2$which!=i],] #treino cana
13.
14.  teste3=dadoserva[folds3$subsets[folds3$which==i],] #teste erva
15.  treino3=dadoserva[folds3$subsets[folds3$which!=i],] #treino erva
16.
17.  dadosteste=rbind(teste1,teste2,teste3) #dados teste
18.  dadostreino=rbind(treino1,treino2,treino3) #dados treino
19.
20.  #modelo cart
21.  Modelo=rpart(Class~NIR+RE+R+G,data=dadostreino,method='class',cp=0.0007)
22.
23.  #predicting
24.  ynew<-predict(modelo,newdata=dadosteste, type='class')
25.
26.  confundimento=table(ynew,dadosteste$Class)#matriz confundimento
27.  acuracia=c(acuracia,sum(diag(confundimento))/sum(confundimento))
```

```
28.
29.   precisao1=c(precisao1,confundimento[1,1]/sum(confundimento[1,]))
30.   precisao2=c(precisao2,confundimento[2,2]/sum(confundimento[2,]))
31.   precisao3=c(precisao3,confundimento[3,3]/sum(confundimento[3,]))
32.
33.   recall1=c(recall1,confundimento[1,1]/sum(confundimento[,1]))
34.   recall2=c(recall2,confundimento[2,2]/sum(confundimento[,2]))
35.   recall3=c(recall3,confundimento[3,3]/sum(confundimento[,3]))
36. } #fim k-fold
37. tempofim=Sys.time() #tempo fim
38. tempofim-tempoinicio #tempo total
39.
40. #medidas
41. medida=c(rep('acuracia',10),
42.          rep('precisao1',10),rep('precisao2',10),rep('precisao3',10),
43.          rep('recall1',10),rep('recall2',10),rep('recall3',10))
44. valores=c(acuracia,precisao1,precisao2,precisao3,recall1,recall2,recall3)
45.
46. dadosmedidas=data.frame(medida,valores)
47. library(ggplot2)
48. ggplot(dadosmedidas,aes(medida,valores,fill=medida))+geom_boxplot()+
49.   theme_minimal()+ylim(0.92,1)
50.
51. tapply(dadosmedidas$valores, dadosmedidas$medida, mean)
```

Os resultados obtidos estão apresentados na Tabela 5. Observa-se que o tempo computacional gasto para realizar as estimativas dos 10 modelos (10-folds) foi de 0,637 segundos. A média das acurácias obtidas para os 10 modelos foi de 0,970. Os valores das precisões médias, considerando as 3 classes (1=solo, 2=cana e 3=erva), foram de 0,998, 0,965 e 0,958, respectivamente. Já os valores médios da medida recall, também considerando as 3 classes, foram 0,999, 0,965 e 0,957, respectivamente.

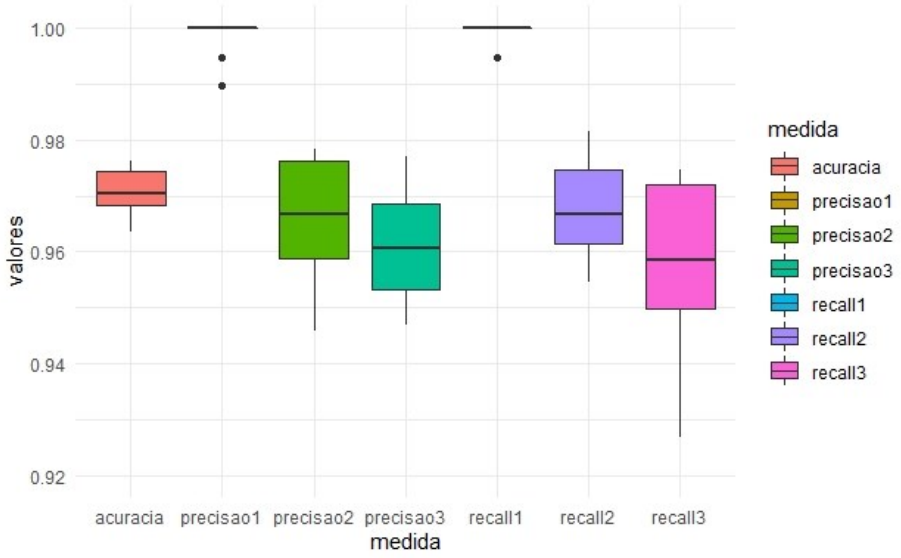
TABELA 5 - Medidas de qualidade de ajuste obtidas nos 10-folds para o modelo Árvores de Decisão

Classes\Medidas	Tempo (segundos)	Acurácia	Precisão	*Recall*
Classe 1 (solo)	22,365	0,970	0,998	0,999
Classe 2 (cana)			0,965	0,965
Classe 3 (erva)			0,958	0,957

FONTE: Autoria Própria (2021).

Na Figura 14 são apresentados os boxplots para tais resultados, nos quais podemos observar que: há uma baixa variabilidade para predição de solo ainda que a precisão variou entre 99,5% a 100% e, a precisão de erva daninha variou de aproximadamente 95,5 a 97% e o recall de erva daninha variou de 92,8 a 97,5%.

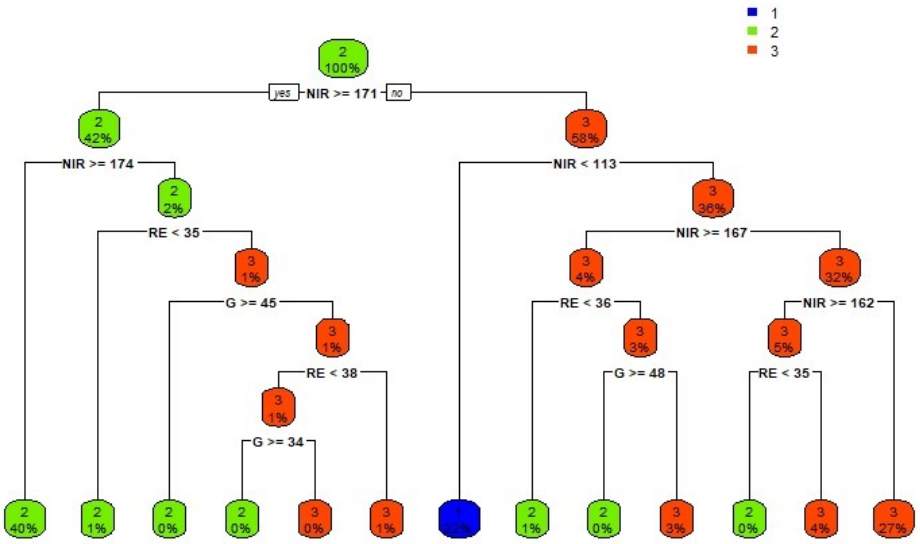
FIGURA 14 - Bloxplots das medidas de qualidade de ajuste obtidos nos 10-folds para o modelo ÁRVORES DE DECISÃO.



FONTE: Autoria própria (2021).

A Figura 15 ilustra a árvore gerada pelo modelo, onde o tamanho de cada ramo é dado proporcionalmente à diminuição do erro quadrático médio. Nota-se que a variável com menor entropia, ou seja, a que gera a primeira divisão nos dados, é a variável NIR >= 171.

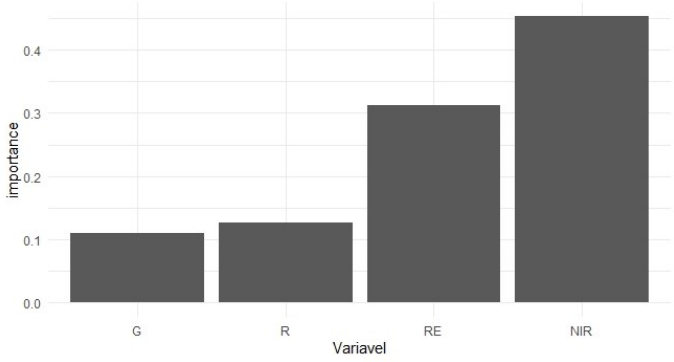
FIGURA 15 - Árvore gerada pelo método CART.



FONTE: Autoria própria (2021).

Baseado na importância dos níveis de cores, temos a variável com maior potencial o NIR (nível de infra-vermelho) em comparação ao nível de cor G (Verde), conforme Figura 16 fornecida pelo pacote.

FIGURA 16 - Importância dos níveis de cores do modelo CART.



FONTE: Autoria própria (2021).

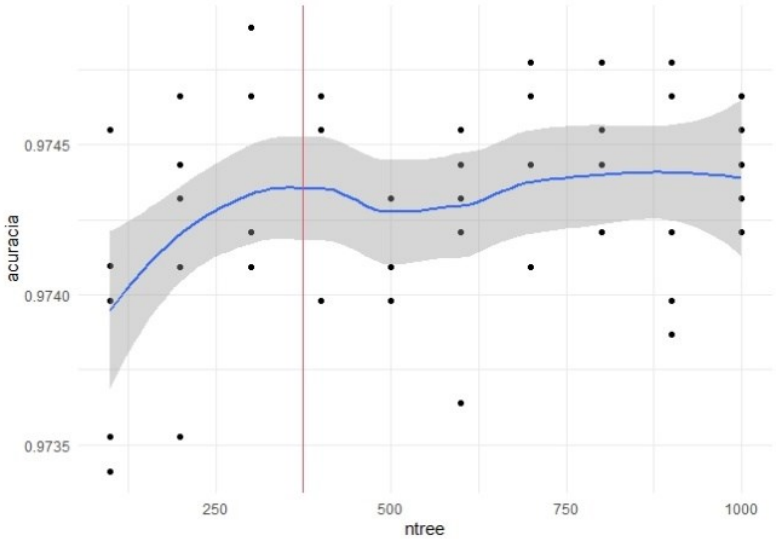
4.5 FLORESTAS ALEATÓRIAS

Para a construção do modelo Florestas Aleatórias, inicialmente deve-se determinar a seleção do hiperparâmetro, o qual refere-se à quantidade de *árvores* a ser utilizado. Para isso, nos códigos abaixo utilizamos o método de validação cruzada k-fold (com k=5), variando o número de *árvores* de 1 a 1000 para a determinação da melhor acurácia.

```
1. library(randomForest)
2. 
3. acuracia=acuracia2=c()
4. tempoinicio=Sys.time() #tempo
5. k=5
6. for(c in seq(100,1000,100)){
7. for(i in 1:k){ #cv k-fold
8.   teste1=dadossolo[folds1$subsets[folds1$which==i],] #teste solo
9.   treino1=dadossolo[folds1$subsets[folds1$which!=i],] #treino solo
10. 
11.  teste2=dadoscana[folds2$subsets[folds2$which==i],] #teste cana
12.  treino2=dadoscana[folds2$subsets[folds2$which!=i],] #treino cana
13. 
14.  teste3=dadoserva[folds3$subsets[folds3$which==i],] #teste erva
15.  treino3=dadoserva[folds3$subsets[folds3$which!=i],] #treino erva
16. 
17.  dadosteste=rbind(teste1,teste2,teste3) #dados teste
18.  dadostreino=rbind(treino1,treino2,treino3) #dados treino
19. 
20.  #modelo Random Forest
21.  Modelo=randomForest(as.factor(Class)~NIR+RE+R+G,
22.                        data=dadostreino,ntree=c)
23.  #predicting
24.  ynew<-predict(modelo,newdata=dadosteste, type='class')
25. 
26.  confundimento=table(ynew,dadosteste$Class)#confundimento
27.  acuracia=c(acuracia,sum(diag(confundimento))/sum(confundimento))
28.  } #fim k-fold
29.  acuracia2=c(acuracia2,mean(acuracia))
30.  acuracia=c()
31.} #fim ntree
32.tempofim=Sys.time() #tempo fim
33.tempofim-tempoinicio #tempo total 14.6min
34. 
35. ntree=rep(seq(100,1000,100),5)
36. datacp=data.frame(acuracia=acuracia2,ntree)
37.ggplot(datacp,aes(ntree,acuracia))+geom_point()+geom_smooth()+
38.     geom_vline(xintercept = 375,col=2)
```

A partir dos códigos apresentados acima, obteve-se as médias da acurácias para cada valor do hiperparâmetro apresentado na Figura 17. Conforme mostrado na figura, tem-se que o número otimizado de árvores é de aproximadamete 375, ou seja, o valor que aumenta o poder de predição. O tempo computacional de realização do hiperparâmetro foi de 88 segundos.

FIGURA 17 - Determinação do hiperparâmetro para o modelo Florestas aleatórias.



FONTE: Autoria própria (2021).

Efetuada a seleção do hiperparâmetro, elaborou-se a construção do modelo final. Os códigos para a construção do modelo, considerando o método de validação cruzada k-fold, com k=10, é expresso por:

```
library(randomForest)

acuracia=precisao1=precisao2=precisao3=c()
recall1=recall2=recall3=c()
k=10
tempoinicio=Sys.time() #tempo
for(i in 1:k){ #cv k-fold
  teste1=dadossolo[folds1$subsets[folds1$which==i],] #teste solo
  treino1=dadossolo[folds1$subsets[folds1$which!=i],] #treino solo

  teste2=dadoscana[folds2$subsets[folds2$which==i],] #teste cana
  treino2=dadoscana[folds2$subsets[folds2$which!=i],] #treino cana

  teste3=dadoserva[folds3$subsets[folds3$which==i],] #teste erva
  treino3=dadoserva[folds3$subsets[folds3$which!=i],] #treino erva

  dadosteste=rbind(teste1,teste2,teste3) #dados teste
  dadostreino=rbind(treino1,treino2,treino3) #dados treino

  #modelo Random Forest
  Modelo=randomForest(as.factor(Class)~NIR+RE+R+G,
                      data=dadostreino,ntree=375)

  #predicting
  ynew<-predict(modelo,newdata=dadosteste, type='class')

```

```
  confundimento=table(ynew,dadosteste$Class)#matriz confundimento
  acuracia=c(acuracia,sum(diag(confundimento))/sum(confundimento))

  precisao1=c(precisao1,confundimento[1,1]/sum(confundimento[1,]))
  precisao2=c(precisao2,confundimento[2,2]/sum(confundimento[2,]))
  precisao3=c(precisao3,confundimento[3,3]/sum(confundimento[3,]))

  recall1=c(recall1,confundimento[1,1]/sum(confundimento[,1]))
  recall2=c(recall2,confundimento[2,2]/sum(confundimento[,2]))
  recall3=c(recall3,confundimento[3,3]/sum(confundimento[,3]))
 print(i)} #fim k-fold
tempofim=Sys.time() #tempo fim
tempofim-tempoinicio #tempo total

#medidas
medida=c(rep('acuracia',10),
         rep('precisao1',10),rep('precisao2',10),rep('precisao3',10),
         rep('recall1',10),rep('recall2',10),rep('recall3',10))
         valores=c(acuracia,precisao1,precisao2,precisao3,
                   recall1,recall2,recall3)

dadosmedidas=data.frame(medida,valores)
ggplot(dadosmedidas,aes(medida,valores,fill=medida))+geom_boxplot()

tapply(dadosmedidas$valores, dadosmedidas$medida, mean)


####Dados com resposta
dados1=dados[1:8801,]
####Dados sem resposta
dados2=dados[8802:127853,]

#modelo final
modelofinal=randomForest(as.factor(Class)~NIR+RE+R+G,
             data=dados1,ntree=375)

#predicting
ynew<-predict(modelofinal,newdata=dados2, type='class')
dados2$Class=ynew
dadostudo=rbind(dados1,dados2)
dadostudo$Class=as.numeric(dadostudo$Class)

#mapas
library(ggmap)

#campo 2
campo2=subset(dadostudo,long<= -47.815)

#solo, cana, erva
color2<-c('blue', 'chartreuse2', 'orangered')

```

```
qmplot(long, lat, data = campo2 ,zoom = 14)+ ggtitle("Field 2")+
  geom_point(aes(long,lat),color=color2[campo2$Class])

#campo 1
campo1=subset(dadostudo,long>= -47.815)

qmplot(long, lat, data = campo1 ,zoom = 14)+ ggtitle("Field 1")+
  geom_point(aes(long,lat),color=color2[campo1$Class])
```

Os resultados obtidos estão apresentados na Tabela 6. Observa-se que o tempo computacional gasto para realizar as estimativas dos 10 modelos (10-folds) foi de 12 segundos e, calculando o custo computacional como o tempo de seleção do hiperparâmetro com o custo da construção do modelo, o tempo total foi de 100s. A média das acurácias obtidas para os 10 modelos foi de 0,974. Os valores das precisões médias, considerando as 3 classes (1=solo, 2=cana e 3=erva), foram de 0,999, 0,969 e 0,965, respectivamente. Já os valores médios da medida recall, também considerando as 3 classes, foram 1,000, 0,971 e 0,962, respectivamente.

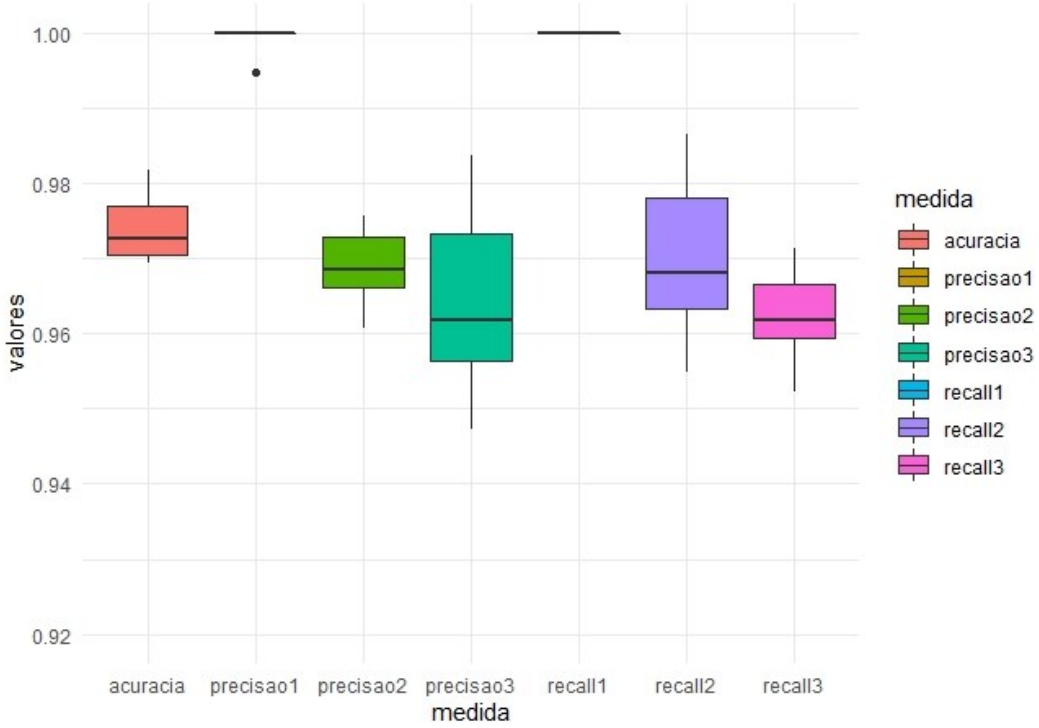
TABELA 6 - Medidas de qualidade de ajuste obtidas nos 10-folds para o modelo Florestas Aleatórias

Classes\Medidas	Tempo (segundos)	Acurácia	Precisão	*Recall*
Classe 1 (solo)	100,000	0,974	0,999	1,000
Classe 2 (cana)			0,969	0,971
Classe 3 (erva)			0,965	0,962

FONTE: Autoria Própria (2021).

A dispersão destas medidas podem ser observadas através dos Boxplots ilustados na Figura 18, na qual verificou-se uma baixa variabilidade para predição de solo, a precisão de erva daninha variou de aproximadamente 95 a 98,5% e o recall de erva daninha variou de 95 a 97%.

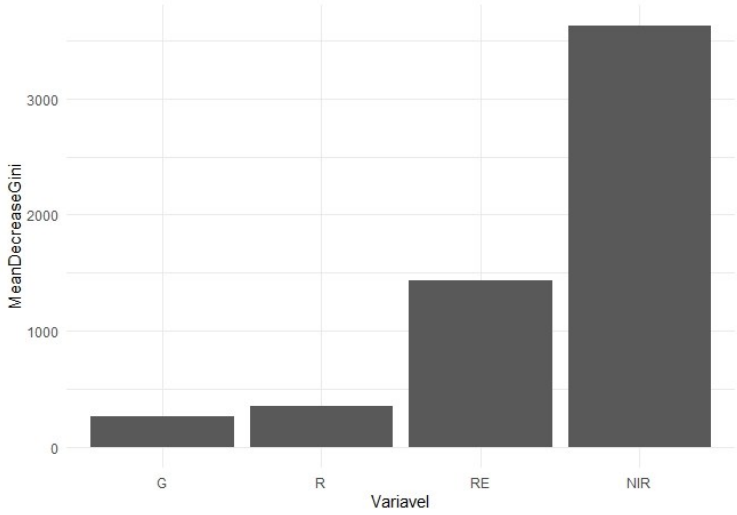
FIGURA 18 - Bloxplots das medidas de qualidade de ajuste obtidos nos 10-folds para o modelo FLORESTA ALEATÓRIA.



FONTE: Autoria própria (2021).

Ainda como informação, foram determinadas as importâncias de cada um dos níveis de cores, como apresentado na Figura 19. Constata-se a partir da figura que o NIR foi a informação com maior significância, seguido por RE, R e G, respectivamente.

FIGURA 19 – Importância dos níveis de cores para o modelo FLORESTA ALEATÓRIA.



FONTE: Autoria própria (2021).

4.6 BAGGING CART

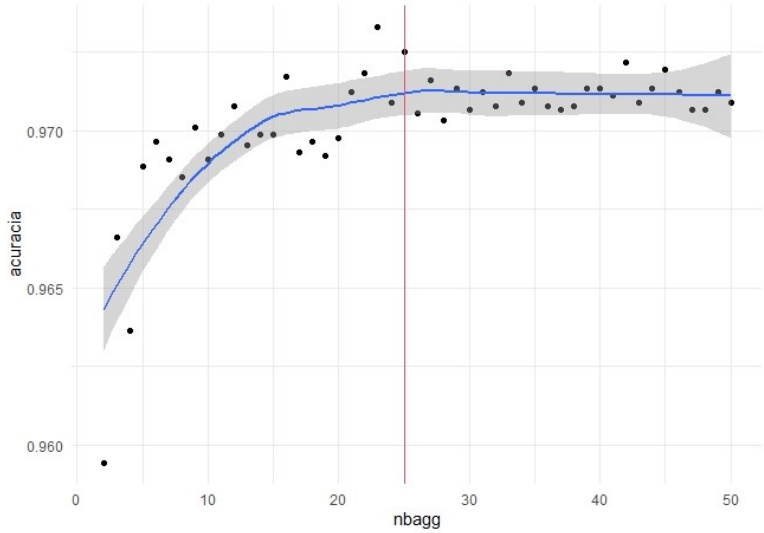
Para a construção do modelo Bagging Cart, o primeiro passo deve ser a seleção do hiperparâmetro, o qual refere-se a quantidade de *bagging* a ser utilizado, sendo que o *default* é de 25. Para isso, nos códigos abaixo estamos utilizando o método de validação cruzada k-fold (com k=5), variando o número de *bagging* de 2 a 50 e, com isso, determinando a melhor acurácia.

```
1. library(ipred) #pacote modelo bagging cart
2. acuracia=acuracia2=c()
3. k=5
4. tempoinicio=Sys.time() #tempo
5. for(o in 2:50){ #bagging
6. for(i in 1:k){ #cv k-fold
7.   teste1=dadossolo[folds1$subsets[folds1$which==i],] #teste solo
8.   treino1=dadossolo[folds1$subsets[folds1$which!=i],] #treino solo
9.
10.  teste2=dadoscana[folds2$subsets[folds2$which==i],] #teste cana
11.  treino2=dadoscana[folds2$subsets[folds2$which!=i],] #treino cana
12.
13.  teste3=dadoserva[folds3$subsets[folds3$which==i],] #teste erva
14.  treino3=dadoserva[folds3$subsets[folds3$which!=i],] #treino erva
15.
16.  dadosteste=rbind(teste1,teste2,teste3) #dados teste
17.  dadostreino=rbind(treino1,treino2,treino3) #dados treino
18.
19.  #modelo bagging cart
20.  modelo<-bagging(as.factor(Class)~NIR+RE+R+G,
21.           data=dadostreino,method = 'class',nbagg=o)
22.
23.  #predicting
24.  ynew<-predict(modelo,newdata=dadosteste, type='class')
25.
26.  confundimento=table(ynew,dadosteste$Class)#matriz confundimento
27.  acuracia=c(acuracia,sum(diag(confundimento))/sum(confundimento))}
28.#fim k-fold
29.
30.acuracia2=c(acuracia2,mean(acuracia))
31.acuracia=c()}
32.tempofim=Sys.time() #tempo fim
33.tempofim-tempoinicio
```

Com os códigos apresentados acima, foram obtidas as médias das acuracias para cada valor do hiperparâmetro, as quais são apresentadas na Figura 20. Com essa figura, observa-se que o número ótimo de bagging, ou seja, o valor que aumenta o poder de predição, é de aproximadamente 25, valor esse que

coincide com o default do pacote. Por fim, o tempo computacional para realizar a seleção do hiperparâmetro foi de 1 minuto e 57 segundos.

FIGURA 20 – Determinação do hiperparâmetro para o modelo Bagging Cart.



FONTE: Autoria própria (2021).

Após a seleção do hiperparâmetro, foi construido o modelo final, considerando o número de *bagging* fixo de 25. Os códigos para a construção do modelo, considerando o método de validação cruzada k-fold, com k=10, é dado é apresentado abaixo.

```
k <- 10 #numero de folds

library(ipred) #pacote modelo bagging cart
acuracia=precisao1=precisao2=precisao3=c()
recall1=recall2=recall3=c()

tempoinicio=Sys.time() #tempo
for(i in 1:k){ #cv k-fold
  teste1=dadossolo[folds1$subsets[folds1$which==i],] #teste solo
  treino1=dadossolo[folds1$subsets[folds1$which!=i],] #treino solo

  teste2=dadoscana[folds2$subsets[folds2$which==i],] #teste cana
  treino2=dadoscana[folds2$subsets[folds2$which!=i],] #treino cana

  teste3=dadoserva[folds3$subsets[folds3$which==i],] #teste erva
  treino3=dadoserva[folds3$subsets[folds3$which!=i],] #treino erva

  dadosteste=rbind(teste1,teste2,teste3) #dados teste
  dadostreino=rbind(treino1,treino2,treino3) #dados treino

  #modelo bagging cart
```

```
  modelo<-bagging(as.factor(Class)~NIR+RE+R+G,
            data=dadostreino,method = 'class')

  #predicting
  ynew<-predict(modelo,newdata=dadosteste, type='class')

  confundimento=table(ynew,dadosteste$Class)#matriz confundimento
  acuracia=c(acuracia,sum(diag(confundimento))/sum(confundimento))

  precisao1=c(precisao1,confundimento[1,1]/sum(confundimento[1,]))
  precisao2=c(precisao2,confundimento[2,2]/sum(confundimento[2,]))
  precisao3=c(precisao3,confundimento[3,3]/sum(confundimento[3,]))

  recall1=c(recall1,confundimento[1,1]/sum(confundimento[,1]))
  recall2=c(recall2,confundimento[2,2]/sum(confundimento[,2]))
  recall3=c(recall3,confundimento[3,3]/sum(confundimento[,3]))
} #fim k-fold
tempofim=Sys.time() #tempo fim
tempofim-tempoinicio #tempo total

#medidas
medida=c(rep('acuracia',10), rep('precisao1',10), rep('precisao2',10),
  rep('precisao3',10),rep('recall1',10),
  rep('recall2',10),rep('recall3',10))
valores=c(acuracia,precisao1,precisao2,
            precisao3,recall1,recall2,recall3)

dadosmedidas=data.frame(medida,valores)
library(ggplot2)
ggplot(dadosmedidas,aes(medida,valores,fill=medida))+geom_boxplot()
tapply(dadosmedidas$valores, dadosmedidas$medida, mean)
```

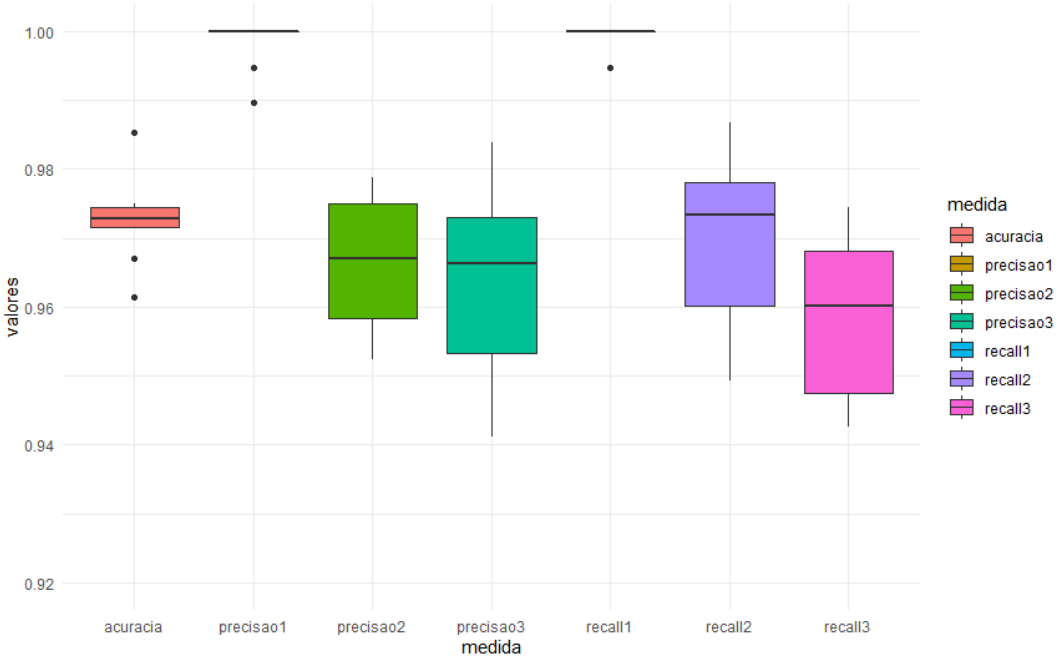
Os resultados obtidos estão apresentados na Tabela 7. Observa-se que o tempo computacional gasto para realizar as estimativas dos 10 modelos (10-folds) foi de 5 segundos, sendo que a soma do custo computacional para a seleção do hiperparâmetro com o custo da construção do modelo foi de 2min02s. A média das acurácias obtidas para os 10 modelos foi de 0,972. Os valores das precisões médias, considerando as 3 classes (1=solo, 2=cana e 3=erva), foram de 0,998, 0,966 e 0,963, respectivamente. Já os valores médios da medida recall, também considerando as 3 classes, foram 0,999, 0,969 e 0,959, respectivamente. As dispersões de tais medidas podem ser observadas na Figura 21, na qual observa-se uma baixa variabilidade para predição de solo, a precisão de erva daninha variou de aproximadamente 94 a 98,5% e o recall de erva daninha variou de 94 a 97%.

TABELA 7 - Medidas de qualidade de ajuste obtidas nos 10-folds para o modelo Bagging Cart

Classes\Medidas	Tempo (segundos)	Acurácia	Precisão	*Recall*
Classe 1 (solo)	122,000	0.972	0.998	0.999
Classe 2 (cana)			0.966	0.969
Classe 3 (erva)			0.963	0.959

FONTE: Autoria Própria (2021).

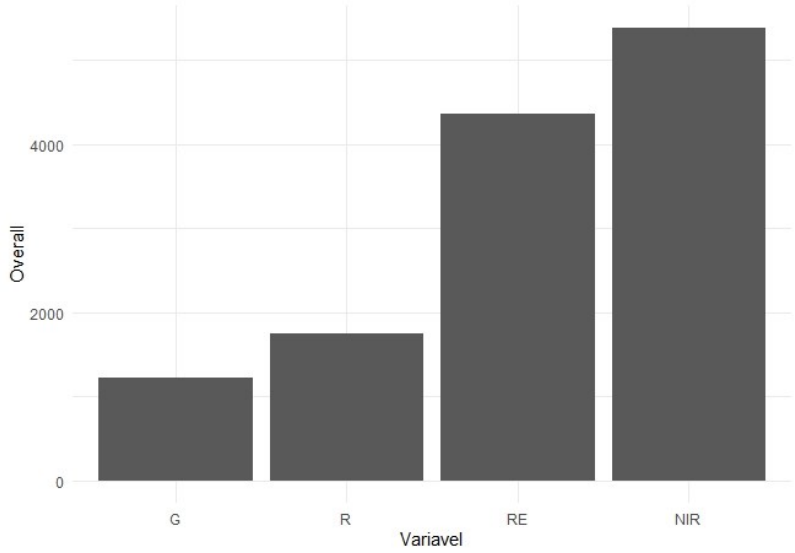
FIGURA 21 - Bloxplots das medidas de qualidade de ajuste obtidos nos 10-folds para o modelo BAGGING CART.



FONTE: Autoria própria (2021).

Como uma informação adicional, foram determinadas as importâncias de cada um dos níveis de cores, como apresentado na Figura 22. Tem-se com essa figura que o NIR foi a informação com maior significância, seguido por RE, R e G, respectivamente.

FIGURA 22 – Importância dos níveis de cores para o modelo Bagging Cart.



FONTE: Autoria própria (2021).

4.7 GRADIENT BOOSTING MACHINE

Para o modelo GBM, assim como os demais modelos, inicia-se realizando a determinação do hiperparâmetro. O primeiro hiperparâmetro a ser determinado é a estimativa do número de árvores n.trees. O número otimizado de árvores utilizada foi em torno de 60. Assim sendo, nos códigos abaixo utilizamos o método de validação cruzada k-fold (com k=5).

Em seguida, realizou-se a estimação dos outros hiperparâmetros, também contido nos códigos abaixo, que são eles: taxa de aprendizagem (shrinkage), profundidade das iterações (interaction.depth), número real de observações (n.minobsinnode) e a fração das observações do conjunto de treinamento (nbag.fraction). Os valores considerados para montagem do grid foram shrinkage = (0,01, 0,1, 0,3), interaction.depth = (1, 3, 5), n.minobsinnode = (5, 10, 15) e bag.fraction = (0,65, 0,8, 1), obtendo uma combinação total de 81 modelos. Após realizar todas as estimativas dos 81 modelos, os valores dos parâmetros que obtiveram a melhor acurácia foram n.trees=96, shrinkage =0,1, interaction.depth=5, n.minobsinnode=15 e bag.fraction = 1.

```
tempoinicio=Sys.time()
# train GBM model
gbm.fit2 <- gbm(Class~NIR+RE+R+G, distribution = "multinomial",
  data = dados1, n.trees = 100, interaction.depth = 3,
  shrinkage = 0.1, cv.folds = 5,
  n.cores = NULL, # will use all cores by default
  verbose = FALSE)
tempofim=Sys.time()
tempofim-tempoinicio #tempo total

# plot loss function as a result of n trees added to the ensemble
gbm.perf(gbm.fit2, method = "cv")

# create hyperparameter grid
hyper_grid <- expand.grid(
  shrinkage =         c(.01, .1, .3),
  interaction.depth = c(1, 3, 5),
  n.minobsinnode =    c(5, 10, 15),
  bag.fraction =      c(.65, .8, 1),
  optimal_trees = 0,               # a place to dump results
  min_RMSE = 0                     # a place to dump results
)# total number of combinations 81

#randomize data
random_index <- sample(1:nrow(dados1), nrow(dados1))
random_ames_train <- dados1[random_index, ]

tempoinicio=Sys.time()
# grid search
for(i in 1:nrow(hyper_grid)) {

  # train model
    gbm.tune <- gbm(Class~NIR+RE+R+G,distribution = "multinomial",
    data = random_ames_train,n.trees = 100,
    interaction.depth = hyper_grid$interaction.depth[i],
    shrinkage = hyper_grid$shrinkage[i],
    n.minobsinnode = hyper_grid$n.minobsinnode[i],
    bag.fraction = hyper_grid$bag.fraction[i],
    train.fraction = .75,
    n.cores = NULL, # will use all cores by default
    verbose = FALSE)

  # add min training error and trees to grid
  hyper_grid$optimal_trees[i] <- which.min(gbm.tune$valid.error)
  hyper_grid$min_RMSE[i] <- sqrt(min(gbm.tune$valid.error))
}
hyper_grid %>%   dplyr::arrange(min_RMSE) %>%  head(10)

tempofim=Sys.time()
tempofim-tempoinicio #tempo total
```

O tempo computacional para a determinação dos hiperparâmetros foi de 65s. Efetuada a seleção do hiperparâmetro, foi realizada enão a construção do modelo final. Os códigos para a construção do modelo, considerando o método de validação cruzada k-fold, com k=10, é dado é apresentado abaixo.

```
k <- 10 #numero de folds

#normalizando
normalize <- function(x) {return ((x - min(x)) / (max(x) - min(x)))}

library(gbm) #pacote modelo gradient boosted machine

acuracia=precisao1=precisao2=precisao3=c()
recall1=recall2=recall3=c()

tempoinicio=Sys.time() #tempo
for(i in 1:k){ #cv k-fold
  teste1=dadossolo[folds1$subsets[folds1$which==i],] #teste solo
  treino1=dadossolo[folds1$subsets[folds1$which!=i],] #treino solo

  teste2=dadoscana[folds2$subsets[folds2$which==i],] #teste cana
  treino2=dadoscana[folds2$subsets[folds2$which!=i],] #treino cana

  teste3=dadoserva[folds3$subsets[folds3$which==i],] #teste erva
  treino3=dadoserva[folds3$subsets[folds3$which!=i],] #treino erva

  dadosteste=rbind(teste1,teste2,teste3) #dados teste
  dadostreino=rbind(treino1,treino2,treino3) #dados treino

  dadostreino[,1:4]= as.data.frame(lapply(dadostreino[,1:4], normalize))
  dadosteste[,1:4]= as.data.frame(lapply(dadosteste[,1:4], normalize))

  #modelo GBM
  modelo=gbm(Class~NIR+RE+R+G,data=dadostreino,
      distribution  = 'multinomial',
      n.trees = 96,shrinkage = 0.1,interaction.depth = 5,
      n.minobsinnode = 15,bag.fraction =1,cv.folds = 1)

  #predicting
  prediction <-predict(modelo,newdata = dadosteste)
  ynew=max.col(prediction[,,])

  confundimento=table(ynew,dadosteste$Class)#matriz confundimento
  acuracia=c(acuracia,sum(diag(confundimento))/sum(confundimento))

  precisao1=c(precisao1,confundimento[1,1]/sum(confundimento[1,]))
  precisao2=c(precisao2,confundimento[2,2]/sum(confundimento[2,]))
  precisao3=c(precisao3,confundimento[3,3]/sum(confundimento[3,]))

```

```
  recall1=c(recall1,confundimento[1,1]/sum(confundimento[,1]))
  recall2=c(recall2,confundimento[2,2]/sum(confundimento[,2]))
  recall3=c(recall3,confundimento[3,3]/sum(confundimento[,3]))
} #fim k-fold
tempofim=Sys.time() #tempo fim
tempofim-tempoinicio #tempo total

#medidas
medida=c(rep('acuracia',10),
         rep('precisao1',10),rep('precisao2',10),rep('precisao3',10),
         rep('recall1',10),rep('recall2',10),rep('recall3',10))
valores=c(acuracia,precisao1,precisao2,precisao3,
          recall1,recall2,recall3)

dadosmedidas=data.frame(medida,valores)
library(ggplot2)
ggplot(dadosmedidas,aes(medida,valores,fill=medida))+geom_boxplot()
tapply(dadosmedidas$valores, dadosmedidas$medida, mean)
summary.gbm(modelo)
```

Os resultados obtidos estão apresentados na Tabela 8. Observa-se que o tempo computacional gasto para realizar as estimativas dos 10 modelos (10-folds) foi de 74,15 segundos. A média das acurácias obtidas para os 10 modelos foi de 0,960. Os valores das precisões médias, considerando as 3 classes (1=solo, 2=cana e 3=erva), foram de 0,997, 0,946 e 0,958, respectivamente. Já os valores médios da medida recall, também considerando as 3 classes, foram 0,998, 0,966 e 0,929, respectivamente.

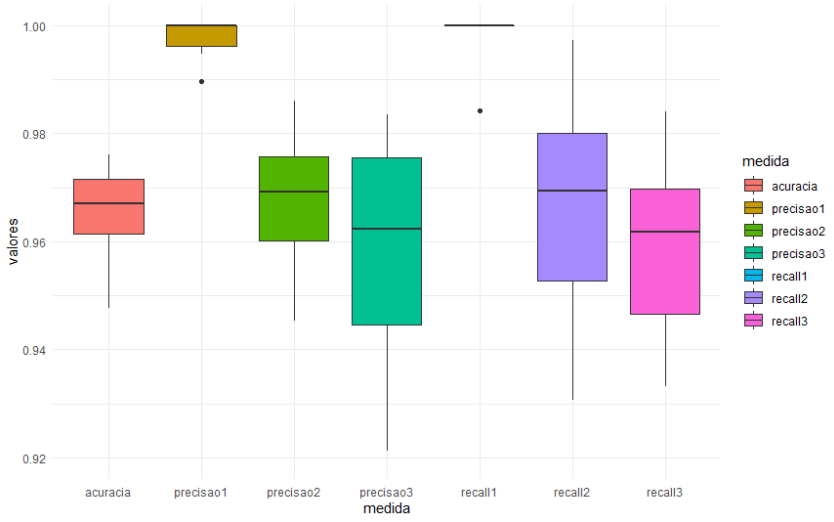
TABELA 8 - Medidas de qualidade de ajuste obtidas nos 10-folds para o modelo Gradient Boosting Machine

Classes\Medidas	Tempo (segundos)	Acurácia	Precisão	*Recall*
Classe 1 (solo)	74,150	0,960	0,997	0,998
Classe 2 (cana)			0,946	0,966
Classe 3 (erva)			0,958	0,929

FONTE: Autoria Própria (2021).

Na Figura 23 são apresentados os boxplots para tais resultados, nos quais podemos observar que também há uma baixa variabilidade para predição de solo, a precisão de erva daninha variou de aproximadamente 92,2 a 98,5% e o recall de erva daninha variou de 93,5 a 98,5%.

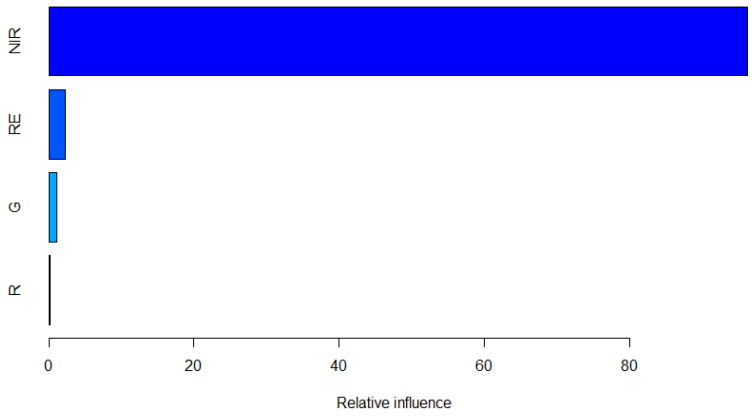
FIGURA 23 - Bloxplots das medidas de qualidade de ajuste obtidos nos 10-folds para o modelo GRADIENT BOOSTING MACHINE.



FONTE: Autoria própria (2021).

Ainda como informação, foram determinadas as importâncias de cores para cada nível, como apresentado na Figura 24, ressaltando também que o NIR possuiu maior significância, seguido por RE, G e R respectivamente.

FIGURA 24 – Importância dos níveis de cores para o pacote Gradient Boosting Machine.



FONTE: Autoria própria (2021).

4.8 K-NEAREST-NEIGHBORS

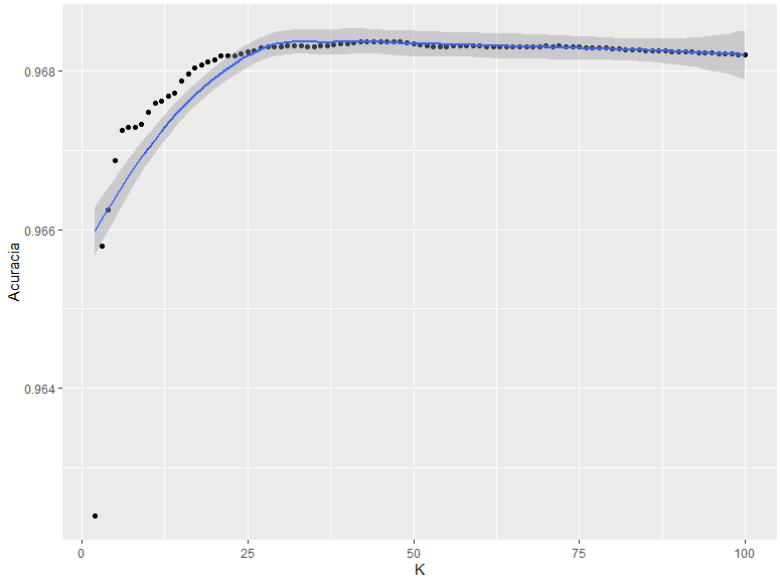
Para o modelo KNN, inicia-se efetuando a determinação da seleção do hiperparanetro (quantidade de vizinhos), considerando um minimo de k=2 e máximo de k=100, com o número de folds=5, e utilizado os códigos para esta seleção conforme abaixo.

```
1. library(cvTools) #biblioteca para k-fold
2. k <- 5 #numero de folds
3.
4. normalize <- function(x) {return ((x - min(x)) / (max(x) - min(x)))}
5. library(caret) #pacote modelo k-Nearest Neighbors
6. library(e1071) #transforma a variável deependente em um fator
7. acuracia=acuracia2=jotas=c()
8.
9. tempoinicio=Sys.time() #tempo
10.for(j in 2:100){ #vizinhos 2 a 100
11.
12.for(i in 1:k){ #cv k-fold
13.    teste1=dadossolo[folds1$subsets[folds1$which==i],] #teste solo
14.    treino1=dadossolo[folds1$subsets[folds1$which!=i],] #treino solo
15.
16.    teste2=dadoscana[folds2$subsets[folds2$which==i],] #teste cana
17.    treino2=dadoscana[folds2$subsets[folds2$which!=i],] #treino cana
18.
19.    teste3=dadoserva[folds3$subsets[folds3$which==i],] #teste erva
20.    treino3=dadoserva[folds3$subsets[folds3$which!=i],] #treino erva
21.
22.    dadosteste=rbind(teste1,teste2,teste3) #dados teste
23.    dadostreino=rbind(treino1,treino2,treino3) #dados treino
24.
25.dadostreino[,1:4]=as.data.frame(lapply(dadostreino[,1:4], normalize))
26.dadosteste[,1:4]= as.data.frame(lapply(dadosteste[,1:4], normalize))
27.
28.    #modelo k-Nearest Neighbors
29.    modelo<-knn3(as.factor(Class)~NIR+RE+R+G,data=dadostreino,k=j)
30.    prediction <-predict(modelo,newdata = dadosteste)
31.    ynew=max.col(prediction)
32.
33.    confundimento=table(ynew,dadosteste$Class)#matriz confundimento
34.    acuracia=c(acuracia,sum(diag(confundimento))/sum(confundimento))
35.  } #fim k-fold
36.acuracia2=c(acuracia2,mean(acuracia))
37.jotas=c(jotas,j)
38.}
```

```
39.tempofim=Sys.time() #tempo fim
40.tempofim-tempoinicio #tempo total
```

Com os códigos apresentados acima, foram obtidas as médias das acuracias para cada valor do hiperparâmetro, as quais são apresentadas na Figura 25. Os valores atribuidos à constante K (de 2 a 100), mostrou que a maior acuracia é obtida quando utilizado 25 vizinhos mais próximos. O tempo computacional para a seleção do hiperparâmetro foi de 61,2 s.

FIGURA 25 – Seleção do hiperparâmetro para o modelo K–Nearest-Neighbors.



FONTE: Autoria própria (2021).

Após efetuada a seleção do hiperparâmetro, elaborou-se a construção do modelo final, conforme o código abaixo, considerando o número de vizinhos mais próximos igual à 25.

```
1. library(cvTools) #biblioteca para k-fold
2. k <- 10 #numero de folds
3.
4. library(caret) #pacote modelo k-Nearest Neighbors
5. normalize <- function(x) {return ((x - min(x)) / (max(x) - min(x)))}
6.
7. acuracia=precisao1=precisao2=precisao3=c()
8. recall1=recall2=recall3=c()
9.
10.tempoinicio=Sys.time() #tempo
11.for(i in 1:k){ #cv k-fold
12.  teste1=dadossolo[folds1$subsets[folds1$which==i],] #teste solo
13.  treino1=dadossolo[folds1$subsets[folds1$which!=i],] #treino solo
14.
15.  teste2=dadoscana[folds2$subsets[folds2$which==i],] #teste cana
```

```
  treino2=dadoscana[folds2$subsets[folds2$which!=i],] #treino cana

  teste3=dadoserva[folds3$subsets[folds3$which==i],] #teste erva
  treino3=dadoserva[folds3$subsets[folds3$which!=i],] #treino erva

  dadosteste=rbind(teste1,teste2,teste3) #dados teste
  dadostreino=rbind(treino1,treino2,treino3) #dados treino

dadostreino[,1:4]=as.data.frame(lapply(dadostreino[,1:4], normalize))
dadosteste[,1:4]=as.data.frame(lapply(dadosteste[,1:4], normalize))

  #modelo k-Nearest Neighbors
  modelo<-knn3(as.factor(Class)~NIR+RE+R+G,data=dadostreino,k=25)

  #predicting
  prediction <-predict(modelo,newdata = dadosteste)
  ynew=max.col(prediction)

  confundimento=table(ynew,dadosteste$Class)#matriz confundimento
  acuracia=c(acuracia,sum(diag(confundimento))/sum(confundimento))

  precisao1=c(precisao1,confundimento[1,1]/sum(confundimento[1,]))
  precisao2=c(precisao2,confundimento[2,2]/sum(confundimento[2,]))
  precisao3=c(precisao3,confundimento[3,3]/sum(confundimento[3,]))

  recall1=c(recall1,confundimento[1,1]/sum(confundimento[,1]))
  recall2=c(recall2,confundimento[2,2]/sum(confundimento[,2]))
  recall3=c(recall3,confundimento[3,3]/sum(confundimento[,3]))
} #fim k-fold
tempofim=Sys.time() #tempo fim
tempofim-tempoinicio #tempo total

#medidas
medida=c(rep('acuracia',10),
         rep('precisao1',10),rep('precisao2',10),rep('precisao3',10),
         rep('recall1',10),rep('recall2',10),rep('recall3',10))
valores=c(acuracia,precisao1,precisao2,precisao3,
           recall1,recall2,recall3)

dadosmedidas=data.frame(medida,valores)
ggplot(dadosmedidas,aes(medida,valores,fill=medida))+geom_boxplot()

tapply(dadosmedidas$valores, dadosmedidas$medida, mean)
```

Os resultados obtidos com a metodologia não paramétrica KNN estão apresentados na Tabela 9, os quais mostram diferentes padrões de desempenhos . Observa-se que o tempo computacional gasto para realizar as estimativas dos 10 modelos (10-folds) foi de 0,5 segundos. A média das acurácias obtidas para os 10 modelos foi de 0,967. Os valores das precisões médias, considerando as 3 classes

(1=solo, 2=cana e 3=erva), foram de 0,999, 0,958 e 0,961, respectivamente. Já os valores médios da medida recall, também considerando as 3 classes, foram 0,998, 0,967 e 0,949, respectivamente.

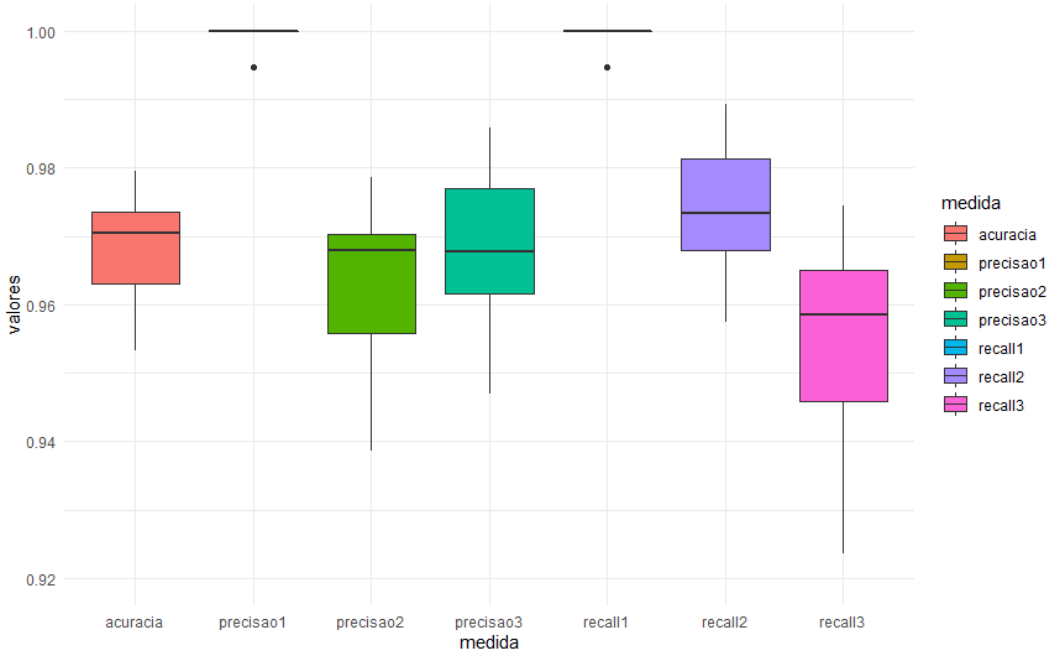
TABELA 9 - Medidas de qualidade de ajuste obtidas nos 10-folds para o modelo K-NEAREST-NEIGHBORS

Classes\Medidas	Tempo (segundos)	Acurácia	Precisão	*Recall*
Classe 1 (solo)	61,700	0,967	0,999	0,998
Classe 2 (cana)			0,958	0,967
Classe 3 (erva)			0,961	0,949

FONTE: Autoria Própria (2021).

Na Figura 26 são apresentados os boxplots para tais resultados, nos quais podemos observar que há uma baixa variabilidade para predição de solo assim como nos demais modelos, a precisão de erva daninha variou de aproximadamente 96 a 98% e o recall de erva daninha variou de 94,5 a 95%.

FIGURA 26 - Bloxplots das medidas de qualidade de ajuste obtidos nos 10-folds para o modelo K-NEAREST NEIGHBORS.



FONTE: Autoria própria (2021).

4.9 C 4.5

O pacote C 4.5 delinea a construção de modelos de classificação a partir de dados analisados. Os códigos utlizados nessa seção estão dispóstos abaixo, salientando que os mesmos, são utilizados para organizar as bases de dados em 10-folds.

```
k <- 10 #numero de folds
library(RWeka) #pacote modelo C4.5

acuracia=precisao1=precisao2=precisao3=c()
recall1=recall2=recall3=c()

tempoinicio=Sys.time() #tempo
for(i in 1:k){ #cv k-fold
  teste1=dadossolo[folds1$subsets[folds1$which==i],] #teste solo
  treino1=dadossolo[folds1$subsets[folds1$which!=i],] #treino solo

  teste2=dadoscana[folds2$subsets[folds2$which==i],] #teste cana
  treino2=dadoscana[folds2$subsets[folds2$which!=i],] #treino cana

  teste3=dadoserva[folds3$subsets[folds3$which==i],] #teste erva
  treino3=dadoserva[folds3$subsets[folds3$which!=i],] #treino erva

  dadosteste=rbind(teste1,teste2,teste3) #dados teste
  dadostreino=rbind(treino1,treino2,treino3) #dados treino

  #modelo C 4.5
  modelo<-J48(as.factor(Class)~NIR+RE+R+G,data=dadostreino)
  ynew<-predict(modelo,newdata=dadosteste, type='class')

  confundimento=table(ynew,dadosteste$Class)#matriz confundimento
  acuracia=c(acuracia,sum(diag(confundimento))/sum(confundimento))

  precisao1=c(precisao1,confundimento[1,1]/sum(confundimento[1,]))
  precisao2=c(precisao2,confundimento[2,2]/sum(confundimento[2,]))
  precisao3=c(precisao3,confundimento[3,3]/sum(confundimento[3,]))

  recall1=c(recall1,confundimento[1,1]/sum(confundimento[,1]))
  recall2=c(recall2,confundimento[2,2]/sum(confundimento[,2]))
recall3=c(recall3,confundimento[3,3]/sum(confundimento[,3]))}
tempofim=Sys.time() #tempo fim
tempofim-tempoinicio #tempo total
```

Os resultados obtidos estão apresentados na Tabela 10. Observa-se que o tempo computacional gasto para realizar as estimativas dos 10 modelos (10-folds) foi de 0.6 segundos. A média das acurácias obtidas para os 10 modelos foi de 0,969.

Os valores das precisões médias, considerando as 3 classes (1=solo, 2=cana e 3=erva), foram de 0,999, 0,965 e 0,955, respectivamente. Já os valores médios da medida *recall*, também considerando as 3 classes, foram 0,999, 0,963 e 0,958, respectivamente.

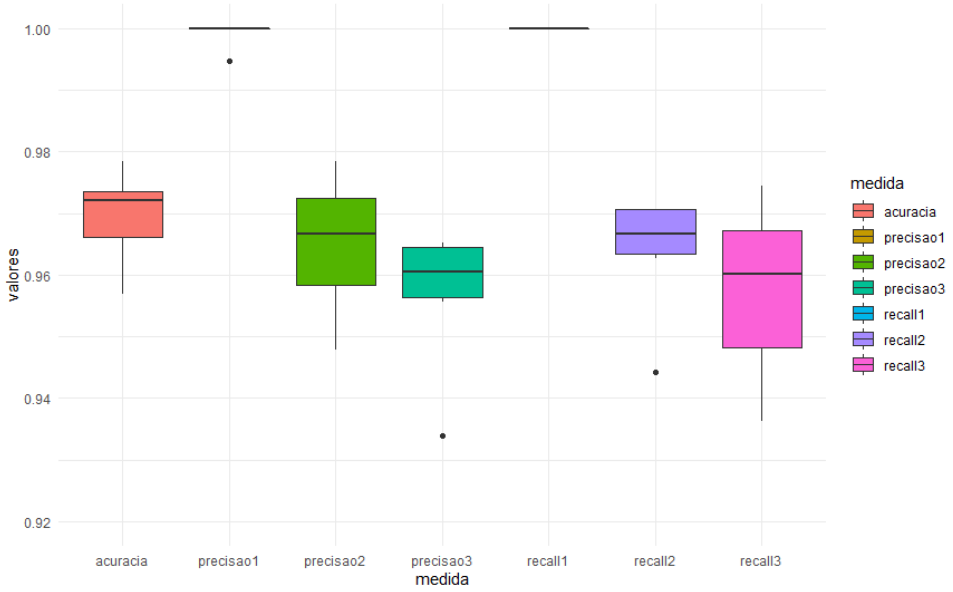
TABELA 10 - Medidas de qualidade de ajuste obtidas nos 10-folds para o modelo C 4.5

Classes\Medidas	Tempo (segundos)	Acurácia	Precisão	*Recall*
Classe 1 (solo)	0,600	0,969	0,999	0,999
Classe 2 (cana)			0,965	0,963
Classe 3 (erva)			0,955	0,958

FONTE: Autoria Própria (2021).

Na Figura 27 são apresentados os boxplots para tais resultados, nos quais podemos observar que há uma baixa variabilidade para predição de solo assim como os demais modelos, a precisão de erva daninha variou de aproximadamente 95,5 a 96,5% e o recall de erva daninha variou de 93,5 a 97,5%.

FIGURA 27 - Bloxplots das medidas de qualidade de ajuste obtidos nos 10-folds para o modelo C 4.5.

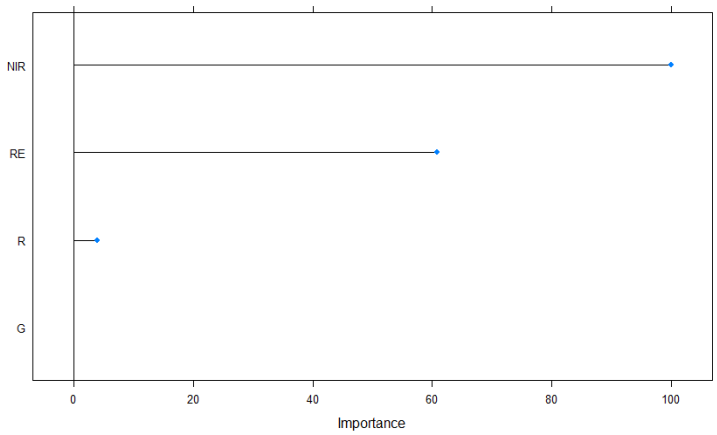


FONTE: Autoria própria (2021).

Como informação adicional, foram determinadas as importâncias de cores para cada nível do modelo C 4.5, como apresentado na Figura 28, ressaltando

também que o NIR possuiu maior significância, seguido por RE, R e G respectivamente.

FIGURA 28 – Importância dos níveis de cores para o modelo C 4.5.



FONTE: Autoria própria (2021).

4.10 COMPARATIVO ENTRE MODELOS

Traçando um comparativo entre os modelos acima citados, é apresentado na Tabela 11 os resultados das acurácias e custos computacionais para cada modelo. Já com a Tabela 12, são apresentados os comparativos das precisões e recall para cada modelo considerado no estudo. Para facilitar uma tomada de decisão, a Tabela 12 é apresentada na Figura 29. De acordo com a Tabela 11, o menor tempo computacional foi apresentado pelo modelo C4.5, porém o melhor valor de acurácia é do modelo Florestas Aleatórias. Ressalta-se também que o tempo de 100 segundos, para o modelo de Florestas Aleatórias, não é um valor elevado, quando comparado com o modelo de Redes Neurais.

TABELA 11 – Valores de Acurácia e Tempo dos 8 modelos trabalhados

Modelo	*Acurácia*	*Tempo (s)*
Gamlss	0,969	158,400
Redes Neurais	0,966	248.820,000
Árvore de Decisâo	0,970	22,365
Florestas Aleatórias	**0,974**	100,000
Bagging Cart	0,972	122,000
Gradient Boosting Machine	0,960	74,150
K-Nearest- Neighbors	0,967	61,700
C 4.5	0,969	0,600

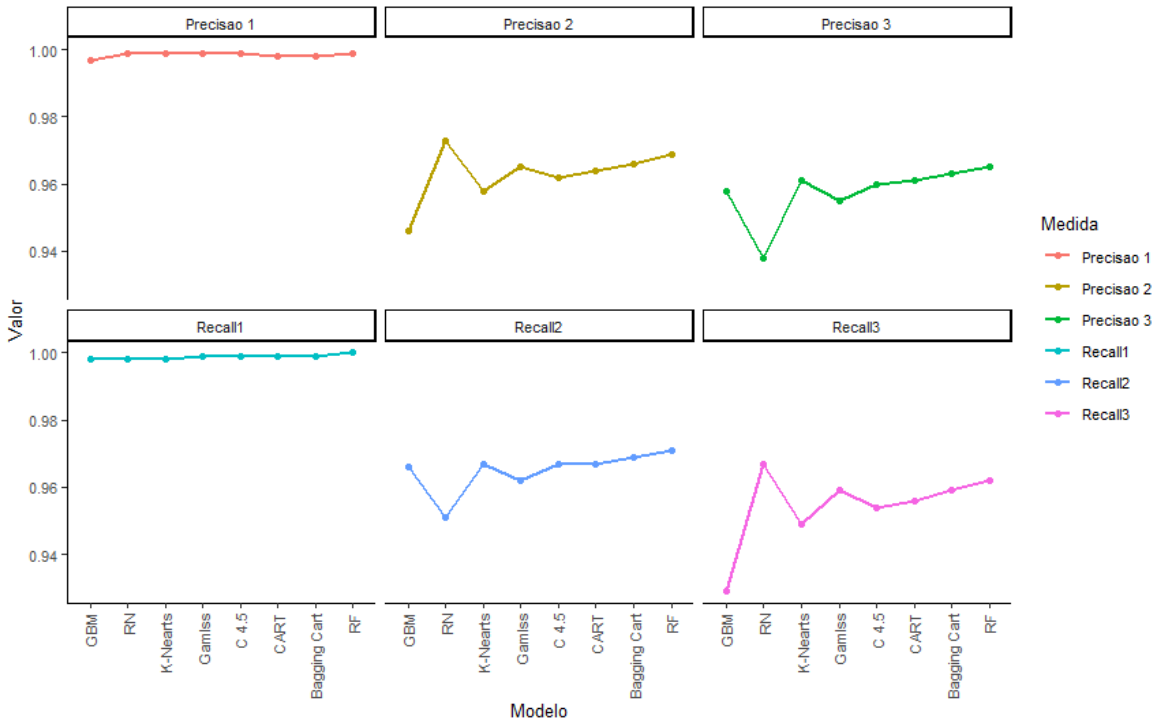
FONTE: Autoria própria (2021).

TABELA 12 - Valores de Precisão e Recall dos 8 modelos trabalhados

Modelo	*Precisão 1*	*Precisão 2*	*Precisão 3*
Gamlss	0,999	0,965	0,955
Redes Neurais	0,999	0,973	0,938
Árvore de Decisão	0,998	0,965	0,958
Florestas Aleatórias	0,999	0,969	**0,965**
Bagging Cart	0,998	0,966	0,963
Gradient Boosting Machine	0,997	0,946	0,958
K-Nearest- Neighbors	0,999	0,958	0,961
C 4.5	0,999	0,965	0,955
Modelo	*Recall1*	*Recall 2*	*Recall 3*
Gamlss	0,999	0,962	0,959
Redes Neurais	0,998	0,951	**0,967**
Árvore de Decisão	0,999	0,965	0,957
Florestas Aleatórias	1,000	0,971	0,962
Bagging Cart	0,999	0,969	0,959
Gradient Boosting Machine	0,998	0,966	0,929
K-Nearest- Neighbors	0,998	0,967	0,949
C 4.5	0,999	0,963	0,958

FONTE: Autoria própria (2021).

FIGURA 29 – Resumo das medidas apresentadas na Tabela 12



FONTE: Autoria própria (2021).

Já com a a Tabela 12 e Figura 29, observa-se que a precisão e o recall para o solo possuem resultados bem próximos de 1, o que significa um ótimo poder de

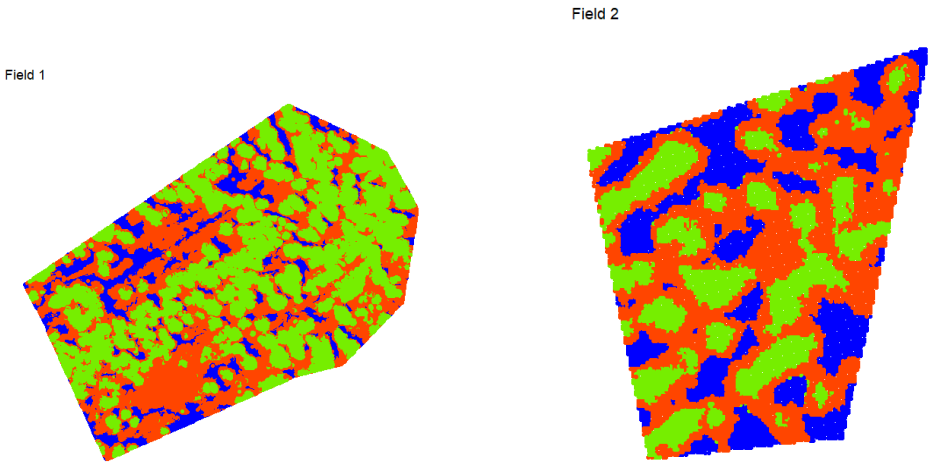
identificação dessa classe. No entanto, o melhor valor de precisão para erva daninha, ou seja, o melhor valor classificado quando comparado aos preditos, é o modelo Florestas Aleatórias.

Por fim, com relação à analise dos niveis de cores, por unanimidade o NIR (infra-vermelho) predominou sobre as outras cores, fato este importante pois considerando que o infra-vermelho possui uma frequência de onda menor que a luz visível (300.10^9 Hz a 430.10^{12} Hz), além de comprimento de onda em torno de 700 e 50.000 nanômetros, sendo a mesma não perceptível pelo olho humano.

Dessa forma, baseado nessas informações e levando em consideração que o principal objetivo do estudo, que é viabilizar a redução de contaminantes no meio ambiente, o modelo Florestas Aleatórias foi o escolhido.

Para realizar as predições das demais classes que não foram classificadas, ou seja, as 119.052 observações que não foram classificadas manualmente, utilizou-se então o modelo de Florestas Aleatórias. Para uma representação gráfica dos dois campos, utilizou-se o pacote ggmap, o qual facilita a compreensão através de mapas utilizando estutura do pacote ggplot2. Os códigos utlizados foram aparesentados na Seção 3.3.4. Conforme apresentado na Figura 30, temos para cada cor a seguinte especificação: azul para solo, verde para cana-de-açucar e vermelho para erva dadinha.

FIGURA 30 – Predição dos dois campos completos utilizando o modelo de Florestas Aleatórias, para cana-de-açúcar (verde), erva daninha (vermelho) e solo (azul).



FONTE: Autoria própria (2021).

Observa-se então que o total de infestação de erva daninha dos dois campos, utilizando o modelo de Florestas Aleatórias, foi de 39,67%. Ao confrontar os resultados apresentados em Righetto *et al.* (2019), os quais utilizaram o modelo GAMLSS para realizar as predições, o nível de infestação detectado foi de 27,02%. Ao comparar as acuracias de ambos modelos, tem-se 97,4% para o modelo de Florestas Aleatórias versus 96,9% para o modelo GAMLSS (RIGHETTO *et al.,* 2019). Dessa forma, tem-se que o modelo Florestas Aleatórias foi capaz de prever um percentual preditivo de 12,65% maior que o modelo proposto por Righetto et al. (2019).

5 CONSIDERAÇÕES FINAIS

Por fim, o presente estudo buscou por meio de um levantamento de modelos de classificação disponíveis no software R, o modelo com um maior poder de predição para o problema elencado. Diante disso, chegou-se à conclusão de que o modelo Florestas Aleatórias obteve o melhor desempenho preditivo, assim como um custo computacional aceitável, em comparação aos demais modelos selecionados para estudo, assim como o modelo já apresentado na literatura.

5.1 RECOMENDAÇÕES PARA TRABALHOS FUTUROS

Uma das principais recomendações para futuros trabalhos, seria a aplicação da metodologia em variados cultivares como por exemplo: soja, milho, trigo, dentre outros, que são plantados em latifúndios e demandam grande quantidade de agrotóxicos para otimização de sua produção, além de estudos de outros modelos com melhor desempenho preditivo para melhora na redução de lançamentos de agrotóxicos.

REFERÊNCIAS

ARTUZO, F. D.; SOARES, C.; WEISS, C. R. (2017). Inovação de processo: O impacto ambiental e econômico da adoção da agricultura de precisão. **Espacios,** 38(2), 1-6.

BASSOI, L. H., et al. Agricultura de precisão e agricultura digital. **Embrapa Pecuária Sudeste-Artigo em periódico indexado (ALICE)**, 2019.

BISHOP, C. M. **Pattern Recognition and Machine Learning.** 1th edição, Springer-Verlag, 2006.

BREIMAN, L. Bagging predictors. **Machine Learning**, v. 24, n. 2, p. 123–140, 1996.

BROWNE, M. W. Cross-validation methods. **Journal of mathematical psychology**, v. 44, p. 108-132, 2000.

CAMPIONE, M.; WALRATH, K. **The Java tutorial: object-oriented programming for the Internet**. Addison-Wesley Longman Publishing Co., Inc., 1996.

CARNEIRO, F. F., et al. **Dossiê ABRASCO: um alerta sobre os impactos dos agrotóxicos na saúde**. EPSJV/Expressão Popular, 2015.

CARVALHO, H. M. **Aprendizado de Máquina voltado para Mineração de Dados: Árvores de Decisão**. Monografia (Especialização em Engenharia de Software), Universidade de Brasília, Brasília, 2014.

COELHO, A. M. Agricultura de precisão: manejo da variabilidade espacial e temporal dos solos e culturas. **Embrapa Milho e Sorgo-Documentos (INFOTECA-E)**, 2005.

CONAB - Companhia Nacional de Abastecimento. Acompanhamento de safra brasileira: Cana-de-açúcar. Safra 2018/2019. Segundo levantamento, Brasília, p. 1-65, agosto 2019.

COSTA, C. N. et al. Contaminantes e poluentes do solo e do ambiente. **MEURER, EJ Fundamentos de química do solo**, v. 2, p. 207-237, 2004.

CRIMINISI, A.; SHOTTON, J. (Ed.). **Decision forests for computer vision and medical image analysis**. Springer Science & Business Media, 2013.

DA SILVA, A. B. R. et al. Introdução da agricultura de precisão no manejo de irrigação utilizando qualisolo mobile. In: **Embrapa Instrumentação-Artigo em anais de congresso (ALICE)**. In: CONGRESSO BRASILEIRO DE CIENCIA DO SOLO, 32., Fortaleza, CE. O solo e a produção de bioenergia: perspectivas e desafios.[S. l.]: SBCS, 2009. 1 CD-ROM., 2009.

DE PAIVA, M. M., et al. Índice de água por diferença normalizada com a linguagem de programação Python aplicado ao Pantanal de Cáceres–Mato Grosso. 7º SIMPÓSIO DE GEOTECNOLOGIAS NO PANTANAL, 2018, Jardim – MS. **Anais**. 2018. p. 799-807.

EILERS, P. H. C.; MARX, B. D.; DURBÁN, M. Twenty years of P-splines. **SORT: statistics and operations research transactions**, v. 39, n. 2, p. 0149-186, 2015.

EILERS, P. H.; MARX, B. D. Flexible smoothing with B-splines and penalties. **Statistical science,** v. 11, n. 2, p. 89-121, 1996.

EMBRAPA, **Agricultura e Inteligência Artificial serão inseparáveis**. Disponível em: https://www.embrapa.br/ . Acesso em: 20 dezembro 2019.

ESRI. **Python For ArcGIS. 2018**. Disponível em: < https://pro.arcgis.com/en/pro-app/arcpy/get-started/installing-python-for-arcgis-pro.htm> . Acesso em: 29 de julho de 2020.

FERNANDES, A. M. R. Inteligência artificial: noções gerais. **Florianópolis, SC: VisualBooks Editora**, 2003.

FILIZOLA, H. F.; FERRACINI, V. L.; SANS, L. M. A.; GOMES, M. A. F.; FERREIRA, C. J. A. (2002) Monitoramento e avaliação do risco de contaminação por pesticidas em água superficial e subterrânea na região de Guaíra. Pesquisa Agropecuária Brasileira, v. 37, n. 5, p. 659-667.

FREITAS, L. F. **Elaboração de um plano de manutenção em uma pequena empresa do setor metal-mecânica de Juiz de Fora com base nos conceitos da Manutenção Preventiva e Preditiva.** 2016. 96f. Dissertação (Trabalho de Conclusão de Curso) – Universidade Federal de Juiz de Fora, Juiz de Fora, 2016.

GASQUES, J. G., et al. Total factor productivity in Brazilian agriculture. **Productivity growth in agriculture: an international perspective**, 2012.

GONG, B.; IM, J.; MOUNTRAKIS, G. An artificial immune network approach to multi-sensor land use/land cover classification. **Remote Sensing of Environment**, v. 115, p. 600-614, 2011.

GONG, R., et al. Acoustic scene classification by fusing LightGBM and VGG-net multichannel predictions. In: **Proc. IEEE AASP Challenge Detection Classification Acoust. Scenes Events**. 2017. p. 1-4.

HANSEN, M.; Dubayah, R.; Defries, R. Classification trees: an alternative to traditional land cover classifiers. **International Journal of Remote Sensing**, v. 17, n. 5, p. 1075-1081, 1996.

HANSEN, M. C., et al. A method for integrating MODIS and Landsat data for systematic monitoring of forest cover and change in the Congo Basin. **Remote Sensing of Environment**, v. 112, n. 5, p. 2495-2513, 2008.

FRIEDMAN, J., et al. **The elements of statistical learning**. New York: Springer series in statistics, 2001.

HAYKIN, S. **Redes neurais: princípios e prática**. Bookman Editora, 2007.

HSU, C. W.; LIN, C. J. (2002). **A comparison of methods for multiclass support vector machines**. IEEE transactions on Neural Networks, 13(2), 415-425.

IM, J.; JENSEN, J. R. A change detection model based on neighborhood correlation image analysis and decision tree classification. **Remote Sensing of Environment**, v. 99, n. 3, p. 326-340, 2005.

JOAQUIM, A. C. **Identificação de variedades de cana-de-açúcar em três classes texturais de solos, na região de Araraquara - SP, através de análise de nível de cinza em imagens Landsat/TM.** Dissertação (Mestrado em Engenharia Agrícola) Faculdade de Engenharia Agrícola, Universidade Estadual de Campinas, Campinas, 1998.

KUMAR, S. **Reinventing Agri-produce quality testing using AI. National Workshop on Artificial Intelligence in Agriculture.** Indian Council of Agricultural Research. New Delhi. p.44-45. 2018.

KUVA, M. A. **Efeito de período de controle e de convivência das plantas daninhas na cultura da cana-de-açúcar (Saccharum sp) no Estado de São Paulo**. Dissertação (Mestrado em Agronomia) Escola Superior de Agricultura Luiz de Queiroz, Piracicaba SP, 1999.

LIAW, A., et al. Classification and regression by randomForest. **R news**, v. 2, n. 3, p. 18-22, 2002.

MARTINS, M. C. **Análise conjunta de regressões com desenvolvimentos computacionais em Linguagem R**. Tese (Doutorado em Ciências e Tecnologia) Universidade Aberta, Lisboa, 2010.

MOLIN, J. P.; DO AMARAL, L. R.; COLAÇO, A.. **Agricultura de precisão.** Oficina de Textos, 2015.

MONTGOMERY, D. C.; RUNGER, G. C. Estatística aplicada e probabilidade para engenheiros, 2ª. **Ed. Rio de Janeiro: Editora LTC**, 2003.

MORIYA, É. A. S.; IMAI, N. N.; TOMMASELLI, A. M. G. Avaliação do potencial de índices de vegetação para detecção de doença na cana-de-açúcar em imagens hiperespectrais adquiridas por VANT. In: SIMPÓSIO BRASILEIRO DE SENSORIAMENTO REMOTO-SBSR, XVIII., 2017, Santos-SP. **Anais**. 2017. p. 0886-0993.

MOTOMIYA, A. V. A., et al. Mapeamento do índice de vegetação da diferença normalizada em lavoura de algodão. **Pesquisa Agropecuária Tropical**, v. 42, p. 112-118, 2012.

MUELLER, W., et al. An IT system for the simultaneous management of vector and raster images. In: **Eighth International Conference on Digital Image Processing (ICDIP 2016)**. International Society for Optics and Photonics, 2016. p. 100332N.

NEPSTAD, D., et al. Slowing Amazon deforestation through public policy and interventions in beef and soy supply chains. **science**, v. 344, n. 6188, p. 1118-1123, 2014.

OLIVEIRA JR., R. S. Introdução ao Controle Químico. In: **BIOLOGIA e Manejo de Plantas Daninhas.** [S. l.: s. n.], 2011. Disponível em: http://omnipax.com.br/livros/2011/BMPD/BMPD-cap6.pdf. Acesso em: 13 maio 2019

PETERNELLI, L. A., et al. Análise dos coeficientes de endogamia e de parentesco para qualquer nível de ploidia usando o pacote estatístico R. **Bragantia**, v. 68, p. 849-855, 2009.

PIRES, J. L. F., et al. Discutindo agricultura de precisão-aspectos gerais. **Embrapa Trigo-Documentos (INFOTECA-E)**, 2004.

POZZEBON, E.; FRIGO, L. B.; BITTENCOURT, G. Inteligência artificial na educação universitária: quais as contribuições. **Revista do Centro de Ciências da Economia e Informática da Universidade da Região da Campanha Urcamp, Editora da URCAMP-EDIURCAMP**, v. 8, n. 13, p. 34-41, 2004.

QUINLAN, J. **C4. 5**: programs for machine learning. San Mateo: M. Kaufmann, 1993. 302 p.

RICHARD JR., E. P. Contral of perennated bermudagrass (Cynodon dactylon) and johnsongrass (Sorghum halepense) in sugarcane (Saccharum spp. hybrids). **Weed Technology**, v.12, p.128-133, 1998.

RICHARD JR., E. P. Effects offallow bermudagrass (Cynodon dactylon) control programs on newly planted sugarcane (Saccharum spp. hybrids). **Weed Technology**, v.ll, p.677-682, 1997.

RIGBY, R. A.; STASINOPOULOS, D. M. Generalized additive models for location, scale and shape. **Journal of the Royal Statistical Society: Series C (Applied Statistics)**, v. 54, n. 3, p. 507-554, 2005.

RIGHETTO, A. J., et al. Predicting weed invasion in a sugarcane cultivar using multispectral image. **Journal of Applied Statistics**, v. 46, n. 1, p. 1-12, 2019.

RIPLEY, B; VENABLES, W; RIPLEY, M.B. **Package 'nnet'**. R package version, v. 7, p. 3-12, 2016.

RITT, A. C. **Análise de Relacionamentos em Conjuntos de Dados de Múltiplas Fontes Utilizando Gradient Boosting**. Tese de Doutorado. Dissertação (Mestrado em Desenvolvimento de Tecnologia) Área de Concentração em Geração e Transferência de Tecnologia, do Instituto de Tecnologia para o Desenvolvimento, Curitiba PR, 2017.

BIVAND, R.; KEITT, T.; ROWLINGSON, B. rgdal: Bindings for the 'Geospatial' Data Abstraction Library. **R package version 1.5-18**. Disponível em: https://CRAN.R-project.org/package=rgdal, 2020. Acesso em: 27 outubro 2020.

RUDORFF, B. F. T.; MOREIRA, M. A.; ALVES, M. Sensoriamento remoto aplicado à agricultura. **São José dos Campos: INPE**, 2002.

SAFAVIAN, S. R.; LANDGREBE, D. A survey of decision tree classifier methodology. **IEEE transactions on systems, man, and cybernetics**, v. 21, n. 3, p. 660-674, 1991.

SANTOS, A. S.; ALMEIDA, A. N. The impact of deforestation on malaria infections in the Brazilian Amazon. **Ecological economics**, v. 154, p. 247-256, 2018.

SENESI, N. Binding mechanisms of pesticides to soil humic substances. **Science of the total Environment**, v. 123, p. 63-76, 1992.

SHOCKLEY, J. M.; DILLON, C. R.; STOMBAUGH, T. S. A whole farm analysis of the influence of auto-steer navigation on net returns, risk, and production practices. **Journal of Agricultural and Applied Economics**, v. 43, n. 1, p. 57-75, 2011.

SOAM, S. K.; RAGHUPHATI, B. **Artificial Intelligence in Agriculture:** Global Status. National Workshop on Artificial Intelligence in Agriculture. Indian Council of Agricultural Research. New Delhi. P.1-7. 2018.

SPADOTTO, C. A. Abordagem interdisciplinar na avaliação ambiental de agrotóxicos. In: **Embrapa Meio Ambiente-Artigo em anais de congresso (ALICE)**. In: JORNADA JURÍDICA DA FACULDADE MARECHAL RONDON, 4., 2006, São Manuel, SP. Artigos publicados... São Manuel, SP: FMR, 2006. p. 1-9. Revista do Núcleo de Pesquisa Interdisciplinar, São Manuel, p. 1-9, maio 2006., 2006.

STASINOPOULOS, D. M., et al. Generalized additive models for location scale and shape (GAMLSS) in R. **Journal of Statistical Software**, v. 23, n. 7, p. 1-46, 2007.

TEAM, R. Core. R: a language and environment for statistical computing. R Foundation for Statistical Computing website. 2020.

TELOKEN, A., et al. Estudo Comparativo entre os algoritmos de Mineração de Dados Random Forest e J48 na tomada de Decisão. **Simpósio de Pesquisa e Desenvolvimento em Computação**, v. 2, n. 1, 2016.

THERNEAU, T.; ATKINSON, B.; RIPLEY, B. rpart: Recursive partitioning and regression trees. **R package version**, v. 4, p. 1-9, 2015.

UNICA, União da indústria de cana-de-açucar. Moagem de cana-de-açucar e produção de açucar e etanol – safra 2011/2012. 2012. Disponível em:http://unica.com.br; Acesso em: 01 Jun. 2021.

VIEIRA, R.; LIMA, V. L. S. Lingüística computacional: princípios e aplicações. In: **Anais do XXI Congresso da SBC. I Jornada de Atualização em Inteligência Artificial**. sn, 2001. p. 47-86.

WALKER, A. Availability of atrazine to plants in different soils. **Pesticide Science**, v. 3, n. 2, p. 139-148, 1972.

WITTEN, I. H.; FRANK, E. Data mining: practical machine learning tools and techniques with Java implementations. **Acm Sigmod Record**, v. 31, n. 1, p. 76-77, 2002.

WUTKA, M., et al. Java: Técnicas Profissionais. **Berkeley, São Paulo**, 1997.

XING, B.; PIGNATELLO, J. J.; GIGLIOTTI, B. Competitive sorption between atrazine and other organic compounds in soils and model sorbents. **Environmental science & technology**, v. 30, n. 8, p. 2432-2440, 1996.

ZURADA, J. M. **Introduction to artificial neural systems**. St. Paul: West, 1992.