UNIVERSIDADE ESTADUAL DA PARAÍBA
CAMPUS I
CENTRO DE CIÊNCIAS E TECNOLOGIA
DEPARTAMENTO DE ESTATÍSTICA
CURSO DE BACHARELADO EM ESTATÍSTICA

VIVIANE COSTA SILVA

USO DE MACHINE LEARNING PARA PREDIÇÃO NA AGRICULTURA: UM TUTORIAL EM R

CAMPINA GRANDE - PB
2021

VIVIANE COSTA SILVA

USO DE MACHINE LEARNING PARA PREDIÇÃO NA AGRICULTURA: UM TUTORIAL EM R

Trabalho de Conclusão de Curso (Artigo) apresentado ao curso de Bacharelado em Estatística do Departamento de Estatística do Centro de Ciências e Tecnologia da Universidade Estadual da Paraíba, como requisito parcial à obtenção do título de bacharel em Estatística.

Orientador: Prof. Drº Tiago Almeida de Oliveira

CAMPINA GRANDE - PB
2021

S586u Silva, Viviane Costa.

Uso de *machine learning* para predição na agricultura [manuscrito] : um tutorial em R / Viviane Costa Silva. - 2021.

29 p. : il. colorido.

Digitado.

Trabalho de Conclusão de Curso (Graduação em Estatística) - Universidade Estadual da Paraíba, Centro de Ciências e Tecnologia , 2021.

"Orientação : Prof. Dr. Tiago Almeida de Oliveira , Coordenação do Curso de Estatística - CCT."

1. Inteligência artificial. 2. Machine learning. 3. Boosting. 4. Estatística. I. Título

21. ed. CDD 006.31

Elaborada por Lucas H. A. da Silva - CRB - 15/898 BC/UEPB

VIVIANE COSTA SILVA

USO DE MACHINE LEARNING PARA PREDIÇÃO NA AGRICULTURA: UM TUTORIAL EM R

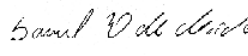
Trabalho de Conclusão de Curso (Artigo) apresentado ao curso de Bacharelado em Estatística do Departamento de Estatística do Centro de Ciências e Tecnologia da Universidade Estadual da Paraíba, como requisito parcial à obtenção do título de bacharel em Estatística.

Trabalho aprovado em 02 de dezembro de 2021.

BANCA EXAMINADORA

Prof. Drº Tiago Almeida de Oliveira
(Orientador)
Universidade Estadual da Paraíba (UEPB)

Prof. Drº Ricardo Alves de Olinda
Universidade Estadual da Paraíba (UEPB)



Prof. Drº Samuel Victor Medeiros Macedo
Instituto Federal de Educação, Ciência e
Tecnologia de Pernambuco (IFPE)

Este trabalho de pesquisa é inteiramente dedicado aos meus pais Lurdinez e Evangelista e as minhas irmãs Vanessa, Vaniara e Vitória (*In Memorian*). Os maiores incentivadores das realizações dos meus sonhos. Muito obrigada!

“Omnia tempus habent.”
Eclesiastes 3,1

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

USO DE MACHINE LEARNING PARA PREDIÇÃO NA AGRICULTURA: UM TUTORIAL EM R

MACHINE USE LEARNING FOR PREDICTION IN AGRICULTURE: A TUTORIAL IN R

Viviane Costa Silva *

RESUMO

O setor da Agricultura tem criado e armazenado grandes quantidade de dados, esses dados podem ser reunidos, armazenados e analisados para auxiliar na tomada de decisão gerando valor competitivo e o uso de técnicas de *Machine Learning* tem sido de grande utilidade para esse mercado. Neste trabalho, foi realizado um estudo de *Machine Learning* usando modelos de classificação supervisionada para predizer doença em uma safra, podendo com isso, identificar o modelo que melhor se adequada aos dados observados. Foram utilizados os modelos *Decision Treen*, *Random Florest* e *Xgboost* para quantificar a safra como sádia ou doente Pode-se observar, que o modelo *Xgboost* forneceu melhores ajuste aos dados com uma acurácia de 0,85 e a área sobre a curva de 0,80.

Palavras-chaves: Boosting; Inteligência Artificial; Estatística.

ABSTRACT

The Agriculture sector has created and stored large amounts of data this data can be gathered, stored and analyzed to assist in decision making, generating competitive value and the use of Machine Learning techniques has been very useful for this market. In this work, a Machine Learning study was carried out using supervised classification models to predict disease in a crop, thus identifying the model that best fits the observed data. Decision Treen, Random Forest and Xgboost models were used to quantify the crop as healthy or sick. It can be observed that the Xgboost model provided better fit to the data accuracy of 0.85 and the area under the curve of 0.80.

Keywords: Boosting; Artificial intelligence; Statistic.

1 INTRODUÇÃO

A agricultura é a principal fonte de subsistência que forma a espinha dorsal do Brasil. Os desafios atuais de escassez de água, custo descontrolado devido ao suprimento de demanda e incerteza climática exigem que os agricultores sejam equipados com agricultura inteligente. Em particular, o baixo rendimento das culturas devido a mudanças climáticas, instalações de irrigação ruins, redução da fertilidade do solo e técnicas agrícolas tradicionais precisam ser contornadas. A agricultura corresponde por 12% do PIB (Produto Interno Brasileiro) e parte da economia brasileira depende da agricultura, pois este é um setor que gera empregos para 22% da população ativa, 20% das exportações são de produtos agrícolas (RONCON, 2011). A produção agrícola depende da estação meteorológica, causas biológicas e econômicas. O prognóstico de saúde da planta agrícola é uma tarefa desafiadora e desejável para todos os campos de produção.

Na agricultura a produção de cultivo pode ser por meio de culturas alimentares ou comerciais. As culturas alimentares incluem arroz, trigo, milho, etc., enquanto as culturas

* Aluna do curso Estatística, Depto Estatística, UEPB, Campina Grande, PB, viviane.silva@aluno.uepb.edu.br

comerciais são algodão, amendoim, caju etc. A produtividade das culturas é significativamente influenciada pelas condições climáticas, portanto, a previsão da saúde da safra que afeta o rendimento é um grande problema que deve ser tratado.

O setor produtivo tem criado e armazenado grandes quantidade de dados. Esses dados podem ser reunidos, armazenados e analisados para auxiliar na tomada de decisão gerando valor competitivo. Novos métodos avançados e abordagens como *machine learning* podem ser utilizados reunindo informações de resultados anteriores de produção de um sistema agrícola para realizar estimativas da produção no tempo futuro (predição). Condições meteorológicas, tais como precipitação, temperatura, condições do solo, topografia e fatores socioeconômicos são responsáveis por 30% do crescimento de uma produção agrícola. Vários trabalhos foram propostos na literatura para predizer a produção usando ferramentas de aprendizado de máquina, tais como regressão, redes neurais artificiais, suporte vector machine entre outros. O aprendizado de máquina (*machine learning*) é uma técnica que pode ser empregada para prever o rendimento das culturas na agricultura, e assim auxiliar o agricultor na tomada de previsão. Várias técnicas de *machine learning* para previsão como: classificação, regressão e agrupamento, são utilizadas para prever o rendimento de uma colheita. Os classificadores ou regressores de Redes neurais artificiais (NN), máquinas de vetores de suporte (SVM), regressão linear e logística, árvores de decisão, Naive Bayes são alguns dos algoritmos usados para implementar a previsão.

Um alto nível de pesticida pode levar a safra a ser considerada como morta/inadequada para consumo entre muitos resultados negativos. Os defensivos agrícolas também são especiais, pois ao mesmo tempo protegem a lavoura com a dosagem certa. Mas, se você adicionar além do necessário, eles podem estragar toda a colheita. Neste sentido, o objetivo deste trabalho é apresentar tutorial utilizando os algoritmos de classificação supervisionada baseadas em *boosting* para predizer doença em uma safra.

2 METODOLOGIA

2.1 *Machine Learning* - Aprendizado de Máquina

O aprendizado de máquina é uma área de IA (Intelegência Artificial) que tem por objetivo desenvolver técnicas computacionais sobre o aprendizado bem como a construção de sistemas capazes de adquirir conhecimento de forma automática sem interferência humana (MONARD; BARANAUSKAS, 2003).

O sistema de aprendizado é um programa computacional que toma decisões apartir de experiências passadas. Existem dois tipos de aprendizado indutivo, o supervisionado que é a tentativa de prever uma variável dependente a partir de uma lista de variáveis independentes, e os não supervisionados que analisam os exemplos fornecidos e tentam determinar se alguns deles podem ser agrupados de alguma maneira, formando agrupamentos ou clusters (CHEESEMAN; STUTZ; HANSON, 1990).

Nenhum algortimo é predominante em todas as situações e problemas que encontramos, portanto é necessário realizar experimentos para verificar qual algoritmo se ajusta melhor no conjunto de dados de estudo (WOLPERT, 1996).

2.2 Pacote Tidymodels

O Pacote Tidymodels (KUHN; WICKHAM, 2020) foi criado por Max Kuhn, Hadley Wickham e RStudio, é uma coleção de pacotes para modelagem e aprendizado de máquina usando os princípios do *tidyverse* que é uma coleção opinativa de pacotes R projetados para ciência de dados, logo na sua inicialização são carregados os pacotes,

- *broom* que leva a saída de funções internas em R, tais como lm, nls, or t.test, e os transforma em quadros de dados arrumados;
- *dials* cria e gerencia valores de parâmetros de ajuste;
- *dplyr* contém uma gramática para manipulação de dados;
- *ggplot2* implementa uma gramática de gráficos;
- *infer* é realizar inferência estatística usando uma gramática estatística;
- *parsnip* generaliza interfaces de modelos entre pacotes;
- *purrr* é um kit de ferramentas de programação funcional;
- *recipes* é um pré-processador de dados geral com uma interface moderna;
- *rsample* possui infraestrutura para reamostragem de dados para que os modelos possam ser avaliados e validados empiricamente.
- *tibble* tem uma reformulação moderna do quadro de dados;
- *tune* contém as funções para otimizar os hiperparâmetros do modelo;
- *workflows* possui métodos para combinar etapas de pré-processamento e modelos em um único objeto;
- *yardstick* contém ferramentas para avaliar modelos.

Os pacotes em *tidymodels* são projetados para trabalhar juntos em um ecossistema unificado, mas são flexíveis e modulares; você pode usar pacotes *tidymodels* para certas partes de uma análise de modelagem sem se comprometer com todo o ecossistema, quando preferir.

2.3 Base de dados

O conjunto de dados utilizado foi obtido da competição hacktoon (VIDHYA, 2020) os dados são baseados em safras colhidas por vários agricultores no final da temporada de colheita e adaptado para esta análise, em que a variável resposta/target (dano na colheita - crop damage) foi convertida em dois valores (0 - sadia, 1- doente, no conjunto de dados original era 0, 1 e 2, sadio, dano por pesticidas e danos por outras causas). O banco de dados original tem 148.168 sendo 88.858 observações não nulas para a variável target. As features consideradas no conjunto de dados são: Quantidade de Insetos estimados (Estimated Insects Count - contínua; Tipo de produção (Crop Type - Contínua); Tipo de Solo (Soil type - Contínua); Categoria de uso de pesticida (Pesticide Use Category - Contínua); Número de doses na semana (Number Doses Week - Contínua); Número de semanas (Number Weeks Used - discreta); Number Weeks Quit (Contínua); Período (Season - Contínua).

2.4 Métodos estatísticos

2.4.1 Técnicas de pré-processamento de dados

A fase de pré-processamento inicia tão longo os dados serem coletados e organizados. Algumas técnicas utilizadas são: 1. Feature selection utilizada para selecionar os atributos mais

relevantes que serão utilizados para treinar o modelo. 2. Feature engineering, arte de criar variáveis a partir de um conjunto de dados visando melhorar a performance de um modelo, técnica não utilizada neste trabalho. 3. Normalização, dataset pode conter variáveis em diferentes escalas e, assim sendo, recomenda-se padronizar esses dados para uma mesma escala. Para isso, usamos a técnica da normalização do conjunto de treino, que trata-se de subtrair a média e dividir pelo desvio padrão dos dados (Técnica de Z-score). 4. Redução de dimensionalidade utilizada quando se tem conjuntos de dados com muitas dimensões (colunas), o que acaba prejudicando a capacidade de generalização de alguns algoritmos, podendo ser utilizado uma técnica de PCA, técnica não utilizada neste trabalho. 5. Divisão dos dados em treino e teste. Em aprendizagem supervisionada, deve-se dividir os dados em dois datasets: treino e teste. Os dados de treino são usados para criar o modelo e os dados de teste são usados para verificar a performance do modelo. Além disso, essa divisão deve ser feita de forma aleatória.

2.4.2 Aprendizagem – Construção do Modelo

Nessa fase, o modelo é construído a partir dos dados que são apresentados ao algoritmo. Algumas técnicas utilizadas são: Cross-validation utilizada para treinar e validar um modelo com o mesmo conjunto de dados, dividindo-os em partições (Figura1).

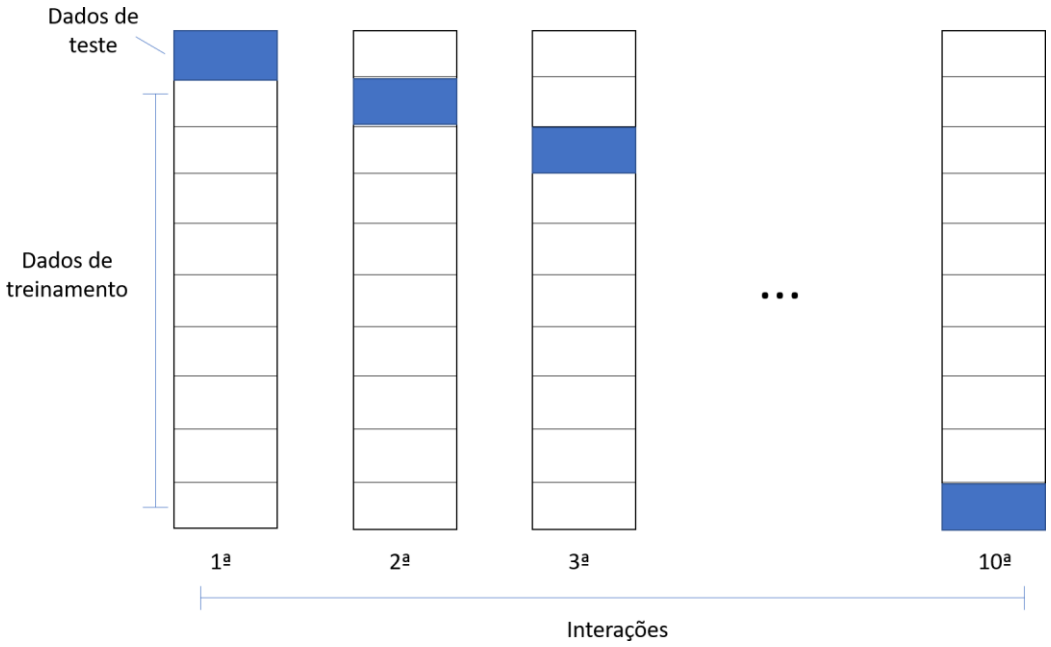


Figura 1 – Representação da validação cruzada (*cross-validation*).

Segundo (HAYKIN, 2007) o *cross validation* é uma técnica que trata da 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.

O *cross validation* (validação cruzada) tem como função separar a base de teste original em duas partes, a primeira sendo a base treinamento dos modelos, que damos o nome de *training data* e a segunda a base de teste onde será testado a validação do modelo que obteve melhor ajuste, damos o nome de *testing data*.

2.4.3 Métricas de desempenho

Existem diversas métricas para medir o desempenho de um modelo. Apenas para exemplificar, podemos medir a acurácia, que é o percentual de previsões corretas em problemas de classificação. Um resumo delas pode ser visto na Tabela 1.

Tabela 1 – Fórmulas de cálculo e explicações de métricas de classes binárias.

Mensuração	Fórmula	Foco de Avaliação
Precisão	$\frac{tp+tn}{tp+fp+tn+fn}$	Usado para medir a proporção de amostras estimadas com precisão em relação ao número total de amostras. Pode-se considerar que o modelo é o melhor se houver alta precisão no modelo usado.
Taxa de erro	$\frac{fp+fn}{tp+fp+tn+fn}$	Usado para medir a proporção dos valores de amostras estimadas incorretamente para o número total de amostras.
Recall (r)	$\frac{tp}{tp+fn}$	Usado para medir a proporção de valores positivos classificados como verdadeiros.
Especificidade	$\frac{tn}{tn+fp}$	Usado para medir a proporção de valores negativos classificados como verdadeiros.
Precisão (p)	$\frac{tp}{tp+fp}$	A proporção de amostras positivas classificadas corretamente em relação ao total de amostras positivas estimadas. Isso também é chamado de valor preditivo positivo.
F1-Score	$\frac{2*p*r}{p+r} = \frac{2*\frac{tp}{tp+fp}*\frac{tp}{tp+tn}}{\frac{tp}{tp+fp}+\frac{tp}{tp+tn}}$	Média harmônica da sensibilidade. Portanto, ele leva em consideração os falsos positivos e os falsos negativos. Especialmente em casos de distribuição irregular de classes, olhar para a pontuação F1 pode ser mais útil do que olhar para a precisão.

Fonte: Produzido pelo o autor.

2.4.3.1 Curva ROC

Uma técnica bastante utilizada na avaliação de métricas em problemas de classificação é a Curva ROC, elas são criadas plotando-se a sensibilidade (verdadeiro-positivo) no eixo y contra 1 especificidade (verdadeiro-negativo) no eixo x para cada valor encontrado (GARCÍA; FERREIRA; PATINO, 2021).

Quanto mais a curva ROC se aproxima do canto superior esquerdo, melhor é a qualidade do teste quanto à capacidade para discriminar. Ainda, a linha de referência diagonal da curva ROC representa uma região de completa aleatoriedade do teste, incapaz de classificar tanto safras doentes como saudáveis (sensibilidade = especificidade). O valor apresentado pela curva ROC fica no intervalo entre 0 e 1,(JR; LEMESHOW; STURDIVANT, 2013) deve ser considerado aceitável acima de 0,7.

2.4.4 Otimização de hiperparâmetros

Cada algoritmo possui um conjunto de hiperparâmetros que podem ser alterados. Assim, essa técnica busca encontrar a combinação certa de valores com o objetivo de melhorar a performance do modelo. Existe diversas técnicas sendo as principais GridSearch, Random Search e HyperOpt.

Segundo (LUO, 2016) existem muitos algoritmos de aprendizado de máquina, a maioria dos quais são complexos. Cada algoritmo de aprendizado de máquina possui dois tipos de parâmetros de modelo: parâmetros comuns que são automaticamente otimizados ou aprendidos em uma fase de treinamento de modelo e hiperparâmetros que são tipicamente definidos pelo usuário manualmente antes de um modelo de aprendizado de máquina ser treinado. A otimização dos hiperparâmetros, pode ser considerado como o processo de encontrar boas configurações para os hiperparâmetros de um determinado algoritmo de *Machine Learning* para um conjunto de dados (PROBST; BISCHL; BOULESTEIX, 2018).

2.4.5 Avaliação do Modelo

Nesta fase, os dados de teste são apresentados ao modelo e, com isso, são geradas previsões. Essas previsões são comparadas com os resultados desejados para avaliar o desempenho do modelo.

2.4.6 Predição

Se o modelo avaliado apresentar um bom resultado, poderá ser utilizado para receber novos dados e realizar predições. Na Figura 2 é apresentado o Workflow da análise em Machine Learning,

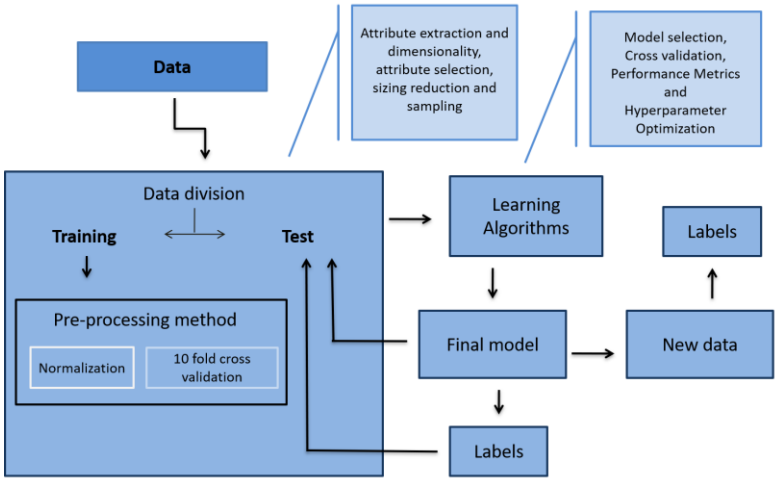


Figura 2 – Workflow da análise de *Machine Learning*

2.4.7 Classificadores utilizados

Para os testes realizados, foram selecionados algoritmos de *Machine Learning* com aprendizado supervisionado para executar a tarefa de classificação binária do problema (safra

sadia ou doente). Decidiu-se por avaliar três tipos de algoritmos de aprendizado de máquina, com o objetivo de verificar aquele que apresenta as melhores métricas e a melhor interpretação. Foram utilizados algoritmos de árvore de decisão, sendo eles o *Decision trees*, *Random Florest* que se utiliza do método *bagging* que é uma combinação de modelos de aprendizagem que aumenta o resultado geral. e *Xgboost* sendo o último baseado em boosting (boosting trees), este tipo de algoritmo apresenta a vantagem de para conjuntos de dados médio apresentar um tempo de treinamento rápido comparados a outros algoritmos além de não requerer muito tempo para ajustar os hiperparâmetros (*hyperparameters tuning*).

2.4.7.1 Decision trees

Decision trees (Árvores de decisão) são métodos de classificação de dados no contexto da chamada Mineração de Dados. A formação de uma árvore de decisão é composta por nós, ramos e folhas. Os nós são as regiões onde se localiza os testes lógicos de separação de dados e são divididos em nó raiz e nó principal. Os ramos fazem uma conexão entre os nós raiz e os nós filhos. Nas folhas se localizam os rótulos ou valores. Uma grande vantagem das árvores de decisão é a tomada de decisão apartir dos atributos mais relevantes, como mostra a Figura 3.

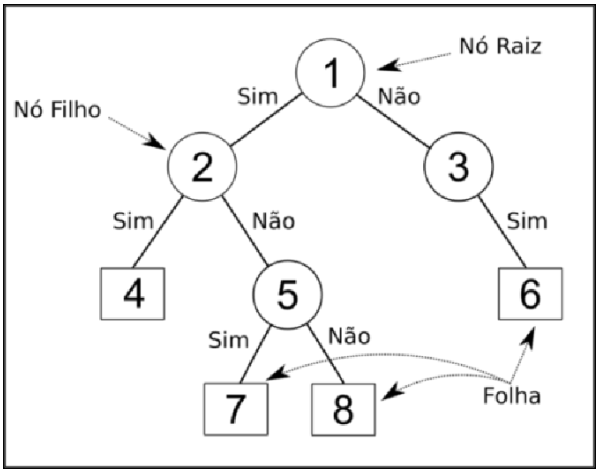


Figura 3 – Estrutura geral de uma árvore de decisão.

Fonte: https://www.researchgate.net/figure/Figura-1-Estrutura-geral-de-uma-arvore-de-decisao_fig1_261913754

As Árvores de Decisão são representações simples do conhecimento e um meio eficiente de construir classificadores que predizem classes baseadas nos valores de atributos de um conjunto de dados (GARCIA, 2003).

Após a construção da árvore, avalia-se os modelos por meio da utilização de dados que não tenham sido usados no treinamento, a base de teste dos dados. Essa estratégia permite estimar como a árvore generaliza os dados e adapta-se a novas situações, além de determinar a proporção de erros e acertos ocorridos na construção da árvore (BRAZDIL, 2002).

O hiperparâmetro complexidade de custo (cp) descrito pelo o comando `cost_complexity` tem a finalidade de evitar a divisão da partição, caso não se obtenha melhoras significativas na qualidade do modelo. Esse hiperparâmetro determina uma correção à árvore, devido ao grande número de divisões. É obtido um valor padrão de 0,01 que quanto maior, menor a árvore.

2.4.7.2 Random Florest

A partir de um vetor de entrada, o modelo de classificação *Random Forest* (florestas aleatórias), disponível no pacote *ranger* (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.

Existem três hiperparâmetros no algoritmo *Random Florest* do pacote *ranger*, mtry que é o número de preditores que serão amostrados aleatoriamente em cada divisão ao criar os modelos de árvore, o trees o número de árvores contidas no conjunto e o min_n sendo o número mínimo de pontos de dados em um nó que são necessários para que o nó seja dividido posteriormente.

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

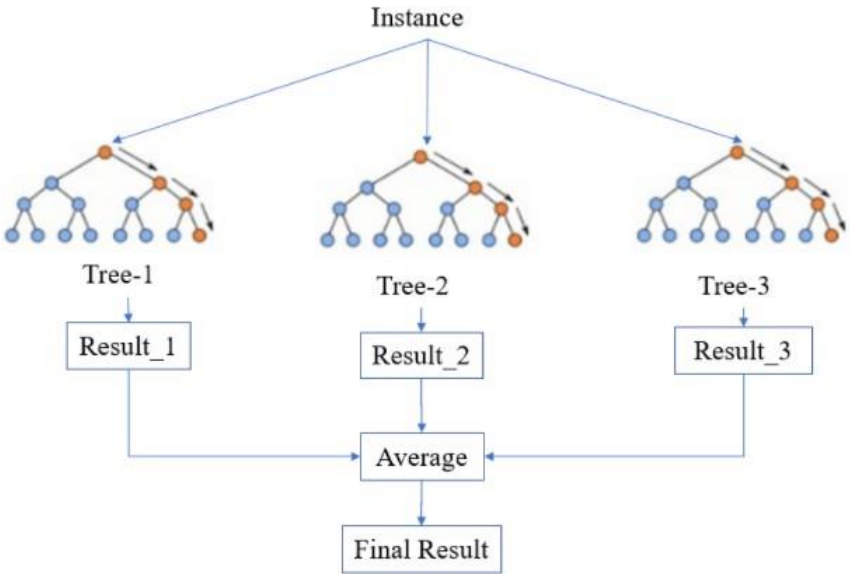


Figura 4 – Estrutura geral do *Random Florest*.

Fonte: https://www.researchgate.net/figure/Simplified-structure-of-random-forest$_{f}$ig1$_{3}$48025909

2.4.7.3 XGBoost

O *eXtreme Gradient Boosting* (*XGBoost*) é um algoritmo altamente escalável, versátil e flexível. Ele foi projetado para explorar os recursos corretamente e para superar as limitações do *gradient boosting*.

O *XGBoost* é um algoritmo de aprendizado supervisionado que implementa um processo chamado boosting (impulsionar) para produzir modelos precisos. É um algoritmo de aumento da árvore de decisão. *Boosting* se refere à técnica de aprendizado de conjunto de construir muitos modelos sequencialmente, com cada novo modelo tentando corrigir as deficiências do modelo anterior. Na árvore de reforço, cada novo modelo adicionado ao conjunto é uma árvore de decisão. A principal diferença entre *XGBoost* e os gradiente *boosting* é que o *XGBoost* utiliza uma nova

forma de regularização para controlar o *overfitting*. Ele é mais rápido e mais robusto durante a tunagem do modelo.

O modelo *Boosted trees* via *xgboost* usado nesse trabalho, utiliza de 6 hiperparâmetros:

- `tree_depth` a profundidade da árvore;
- `learn_rate` Taxa de aprendizagem;
- `mtry` Preditores que são selecionados aleatoriamente;
- `min_n` Tamanho mínimo do nó;
- `loss_reduction` Redução de perda mínima;
- `sample_size` Proporção de observações amostradas;
- `trees` Número de árvores.

A estrutura básica de um modelo *XGBoost* é mostrada na Figura 5.

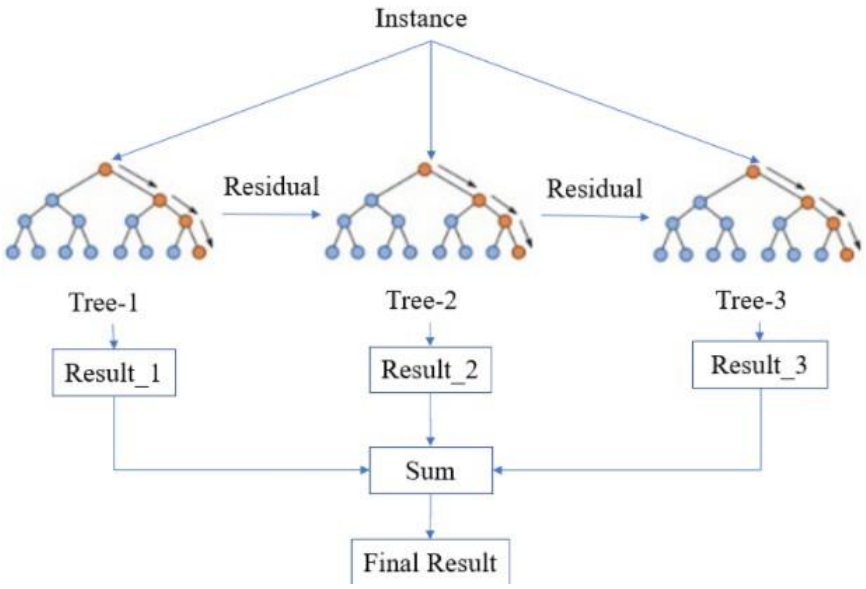


Figura 5 – Estrutura simplificada do *XGBoost*.

Fonte: https://www.researchgate.net/figure/Simplified-structure-of-XGBoost$_f$ig2$_3$48025909

3 APLICAÇÃO

3.1 Intalação do pacote

Primeiro, precisamos instalar a biblioteca *tidymodels*, esse procedimento só precisará ser feito uma única vez, após a instalação é necessário "carregar" a biblioteca para que os pacotes juntos com suas funções possam ser usadas no R. As funções `install.packages` e `library` são usadas para instalação e carregamento da biblioteca respectivamente.

```
install.packages(tidymodels)
library(tidymodels)
```

Atribui-se ao objeto `dadosagro` o banco de dados, o comando <- (sinal de menor e sinal de menos) significa assinalar (*assign*). Indica que tudo que vem após este comando será salvo com o nome que vem antes. É o mesmo que dizer "salve os dados a seguir com o nome de dados". Como os dados estão numa planilha Excel e em formato *csv* é atribuído o comando `read.csv`, com separação em “,” e `header = TRUE` indicando que temos um título.

```
dadosagro <- read.csv("Dados.csv", sep = ";", header = TRUE)
```

Como a análise dos dados é uma classificação para sabermos se a colheita é sadia ou doente, precisamos transformar a variável *target*(resposta) em um fator, o comando `as.factor` faz essa função.

```
dados$Crop_Damage = as.factor(dados$Crop_Damage)
```

3.2 Análise descritiva

A Tabela 2 apresenta as estatísticas descritivas das variáveis de colheita. O banco de dados é formado por 4 variáveis quantitativas, o número de semanas utilizadas e o número de semanas para sair não são apresentados os valores de média, mediana e desvio padrão, por não apresentarem resultados práticos. A variável de desfecho Dano a colheita possui dois valores 0 (sadio) e 1(doente).

É necessário “chamar” os pacotes da biblioteca *fBasics* e em seguida usa a função `basicStats` junto com o conjunto de dados.

```
library(fBasics)
basicStats(dados)
```

Tabela 2 – Distribuição amostral das variáveis

Variáveis	Min	Max	Média	MD	Variância	DP
Contagem estimada de insetos	150	4097	1398,21	1212	721.136,26	849.197,42
Categoria de uso de pesticidas	1	3	2,263802	2	0,213217	0,461755
Número Doses - Semana	0	95	25,850724	20	241,038263	15,525407
Número de semanas usadas	0	67	28,656485	28	153,850842	12,403662
Número de semanas para sair	0	50	9,565324	7	97,664593	9,88254

Nota: MD = Mediana; DP = Desvio Padrão

As frequecências absolutas e relativas para o desfecho de dano na colheita é apresentada na Tabela 3, juntamente das variáveis tipo de solo, tipo de colheita e temporada. A proporção de danos na colheita seja por pesticida ou por outras causas é de 16%, sendo um desfecho desbalanceado, ou seja, distante de uma equidade proporcional na variável resposta.

Usando o pacote *ggplot2* (WICKHAM et al., 2016) que implementa uma gramática de gráficos podemos vizualizar na Figura 6 as estatísticas descritivas em forma gráfica.

Tabela 3 – Frequências absolutas e relativas das variáveis

Variáveis	Frequência	Prop. %
Danos na colheita		
0	66743	84%
1	13115	16%
Tipo de Solo		
0	80441	54%
1	67727	46%
Tipo de Colheita		
0	105873	71%
1	42295	29%
Temporada		
0	7495	5%
1	81484	0,57%
2	26535	19%
3	26761	19 %

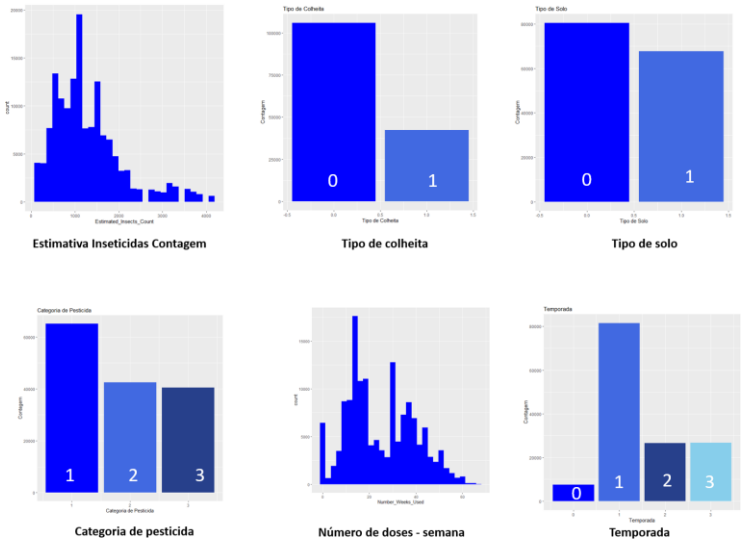


Figura 6 – Painel de gráficos descritivos para *features.*

3.3 Splits – Divisão dos dados

Primeiro, vamos dividir nosso conjunto de dados em dados de treinamento e teste. Os dados de treinamento serão usados para ajustar nosso modelo e seus parâmetros, e os dados de teste serão usados para avaliar o desempenho de nosso modelo final. Essa divisão pode ser feita usando `inital_split()`, função do pacote *rsample* que cria um objeto de divisão. O argumento *strata* usa amostragem estratificada para criar a reamostragem, isso fará que seja aproximadamente as mesmas proporções de Crop_Damage (Dano a colheita) no conjuntos de treinamento e teste.

```
splits <- initial_split(dadosagro, strata = Crop_Damage)
splits
```

```
<Analysis/Assess/Total>
<59893 / 19965 / 79858>
```

A saída do objeto *splits*, diz quantas observações temos no conjunto de treinamento, teste e total. Usando as funções *training* e *testing* o conjunto de treinamento e teste são extraídos do conjunto de dados. Automaticamente o R divide o treinamento com 75% de seus dados e 25% para o teste.

```
data_train <- training(splits)
data_test <- testing(splits)
```

3.4 Engine – Especificação dos modelos

Na especificação dos modelos usaremos o pacote *parsnip* esse pacote oferece uma plataforma unificada de modelos que já existem no R. Existem alguns componentes que fornecem a especificação tal como o tipo do modelo que será modelado. No trabalho são gerados três modelos classificadores de arvores de decisão, *decission tree, random florest e xgboosting*. Deve-se definir o pacote subjacente de onde virá o modelo usado, definido pela função `set_engine` e também a modo do modelo `set_mode`, pois muitos modelos podem ser tanto para classificação quanto para regressão. A função `tune()` significa que os hiperparâmetros serão ajustados pelo pipeline, ou seja o valor do parâmetro que produzir o melhor desempenho será escolhido. O pipe %>% serve para separar e organizar funções dentro de funções. Para a construção de um modelo *Decision Tree* no software R, utilizaremos o pacote *rpart* (THERNEAU et al., 2015) para o *Random Florest* o pacote *ranger* (BREIMAN, 2001) e para o modelo *Xgboost* o pacote *xgboost* (CHEN et al., 2015).

```
dt_model <- decision_tree(
cost_complexity = tune(), min_n = tune()
) %>%
set_engine("rpart") %>%
set_mode("classification")
```

```
rf_model <- rand_forest(
mtry = tune(), min_n = tune(), trees = 1000
) %>%
set_engine("ranger") %>%
set_mode("classification")
```

```
xgb_model <- boost_tree(
tree_depth = tune(), learn_rate = tune(),
loss_reduction = tune(), min_n = tune(),
sample_size = tune(), trees = tune()
) %>%
set_engine("xgboost") %>%
set_mode("classification")
```

3.5 Recipes - Pré processamento dos dados

Nesta etapa é realizado uma série de processos que envolvem a preparação, organização e estrutura dos dados utilizados, a função `recipe` recebe a variável *target* que é Crop_Damge (Dano à colheita) e a fórmula abreviada onde (.) representa todas as variáveis explicativas e ( ) ajustará um modelo que prevê o resultado usando todas as outras colunas que são as *features* variáveis explicativas do banco de dados de treinamento. As funções `step_impute_knn(all_predictors()` retira os dados *missing* esses dados são prejudiciais para tomada de decisão (SHENG; VENDRUSCULO, 2017) e `step_normalize(all_numeric_predictors()` normaliza as variáveis para que todas fiquem em uma escala comum (SIVAKUMAR; GUNASUNDARI, 2017).

```
recipe <- recipe(Crop_Damage ~. ., data = data_train) %>%
step_impute_knn(all_predictors()) %>%
step_normalize(all_numeric_predictors())
```

3.6 Workflow

Após esplecificar os modelos e realizar os pre-processamentos colocaremos tudo junto em um *workflow*. Para cada modelo foi feito um workflow, que inicia-se com a função `workflow()` e em seguida adiciona o modelo `add_model()` e o *recipe* `add_recipe()`. Todos os modelos podem ser colocados juntos em um único *workflow*, usando a função `workflow_set`.

```
dt_workflow <- workflow()
%>%
add_model(dt_model) %>%
add_recipe(recipe)
```

```
rf_workflow <- workflow()
%>%
add_model(rf_model) %>%
add_recipe(recipe)
```

```
xgb_workflow <- workflow()
%>%
add_model(xgb_model) %>%
add_recipe(recipe)
```

3.7 Cross Validation

Existem vários métodos de se estimar a habilidade dos modelos, um bastante utilizado é o *cross validation* (validação cruzada), uma técnica de reamostragem usada para comparar e selecionar os melhores modelos. Para fazermos o ajuste utlizaremos a função `vflod_cv` nele acrescentaremos o nosso banco de dados de treinamento, colocaremos em quantas partes dividiremos os dados `v = 4`, essa divisão é de escolha do pesquisador e especificamos a variável strata = Crop_Damage que é a variável *target* do nosso banco de dados.

```
cv_splits <- vfold_cv( data_train, v = 4, strata = Crop_Damage) #4 folds
```

3.8 Treinamento

Nessa etapa, faremos o treinamento dos modelos, o objeto *workflow* do modelo selecionado é adicionado , dentro da função `tune_grid`, inclui-se a validação cruzada e a grade de parâmetros que deve ser um quadro de dados de combinações de parâmetros. no presente trabalho, foram feitas 10 combinações de parâmetros, a função `control_grid` armazena as informações do treinamento para serem avaliadas posteriomente e em `metric_set` se adiciona as métricas utilizadas para comparar o desempenho dos modelos. As métricas utilizadas neste trabalho foram acurácia, a área sobre a curva, sensibilidade, recall, especificidade e precisão.

```
dt_trained <- dt_workflow %>%
tune_grid(
cv_splits,
grid = 10,
control = control_grid(save_pred = TRUE),
metrics = metric_set(accuracy, roc_auc, sensitivity, recall,
specificity,precision))
```

```
rf_trained <- rf_workflow %>%
tune_grid(
cv_splits,
grid = 10,
control = control_grid(save_pred = TRUE),
metrics = metric_set(accuracy, roc_auc, sensitivity, recall,
specificity,precision))
```

```
xgb_trained <- xgb_workflow %>%
tune_grid(
cv_splits,
grid = 10,
control = control_grid(save_pred = TRUE),
metrics = metric_set(accuracy, roc_auc, sensitivity, recall,
specificity,precision))
```

3.9 Resultados da validação cruzada

3.9.1 Principais submodelos e suas estimativas de desempenho

A função `show_best()` extrai os modelos que melhor se ajustaram, segundo a métrica escolhida, como estamos trabalhando com classificação, uma das métricas mais recomendadas e usadas é a área sobre a curva que está associada ao poder discriminante de um teste de diagnóstico (BRAGA, 2001), para essa métrica utilizaremos o comando `roc_auc`, cada modelo contém sua média e desvio padrão a partir da métrica escolhida. Também nos são apresentadas as estimativas dos hiperparâmetros de cada algoritimo, configurações com as quais pode-se usar para controlar o comportamento do algoritmo (GOODFELLOW; BENGIO; COURVILLE, 2016). Podemos notar que o modelo 06 do algoritmo *xgboost* obteve um melhor desempenho, seu erro padrão é de 0,00319, enquanto o *Random Florest* obteve 0,00339 e o *Decision Treen* 0,00345 pela métrica da área sobre a curva, podemos observar essas métricas na Figura 7.

```
dt_trained %>% show_best("roc_auc")
rf_trained %>% show_best("roc_auc")
xgb_trained %>% show_best("roc_auc")
```

```
> dt_trained  %>% show_best("roc_auc")
# A tibble: 5 x 8
  cost_complexity min_n .metric .estimator  mean     n std_err .config
            <dbl> <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
1       1.21e-10    29 roc_auc binary     0.776     4 0.00345 Preprocessor1_Model02
2       4.51e- 6    20 roc_auc binary     0.775     4 0.00292 Preprocessor1_Model03
3       6.32e- 5    22 roc_auc binary     0.773     4 0.00344 Preprocessor1_Model06
4       1.98e- 8    14 roc_auc binary     0.769     4 0.00213 Preprocessor1_Model04
5       1.11e- 7    13 roc_auc binary     0.760     4 0.00319 Preprocessor1_Model09
> rf_trained  %>% show_best("roc_auc")
# A tibble: 5 x 8
   mtry min_n .metric .estimator  mean     n std_err .config
  <int> <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
1     2    38 roc_auc binary     0.798     4 0.00339 Preprocessor1_Model01
2     2    30 roc_auc binary     0.798     4 0.00331 Preprocessor1_Model09
3     3    25 roc_auc binary     0.795     4 0.00308 Preprocessor1_Model05
4     3     9 roc_auc binary     0.788     4 0.00301 Preprocessor1_Model02
5     6    33 roc_auc binary     0.787     4 0.00299 Preprocessor1_Model06
> xgb_trained  %>% show_best("roc_auc")
# A tibble: 5 x 12
  trees min_n tree_depth  learn_rate loss_reduction sample_size .metric .estimator  mean     n std_err .config
  <int> <int>      <int>       <dbl>          <dbl>       <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
1   520    22         12 0.000000933        4.89e- 9       0.862 roc_auc binary     0.799     4 0.00319 Preprocessor1_Model06
2  1859    14          8 0.000142           2.29e- 5       0.696 roc_auc binary     0.799     4 0.00306 Preprocessor1_Model04
3  1286    12          6 0.00115            2.09e- 4       0.919 roc_auc binary     0.798     4 0.00304 Preprocessor1_Model03
4  1608    32         12 0.000000272        1.66e+ 0       0.278 roc_auc binary     0.787     4 0.00357 Preprocessor1_Model08
5  1056    35          5 0.00000304         3.24e-10       0.594 roc_auc binary     0.784     4 0.00269 Preprocessor1_Model09
>
```

Figura 7 – Principais submodelos do algoritmos *Decision Treen*, *Random Florest* e *Xgboost* e suas estimativas de desempenho.

3.9.2 Métricas

Para observar os resultados das métricas utilizadas usa-se a função `collect_metrics()`. Nesse caso, as métricas vêm do desempenho da validação cruzada entre os diferentes valores dos parâmetros, a função `print(n = Inf)` indica que todas as métricas de todos os modelos treinados serão apresentadas.

```
dt_trained %>%
collect_metrics() %>%
print(n = Inf)
```

```
rf_trained %>%
collect_metrics() %>%
print(n = Inf)
```

```
xgb_trained %>%
collect_metrics() %>%
print(n = Inf)
```

3.9.3 Valores finais dos parâmetros

A função `select_best` extrai o melhor valor para a métrica de precisão, aplicamos a métrica `roc_auc` curva ROC, caso não se especifique qual métrica a ser utilizada o R automaticamente utilizará a primeira métrica atribuida no treinamento. São apresentadas além do melhor modelo, as respectivas estimativas dos hiperparâmetros, o melhor modelo para o *Decision Trees* foi `Preprocessor1_Model02`, para o *Random Florest* foi o `Preprocessor1_Model01` e por fim para o *Xgboost* o modelo `Preprocessor1_Model06`, todos esses selecionados na base de treinamento.

```
dt_trained %>% select_best("roc_auc")
rf_trained %>% select_best("roc_auc")
xgb_trained %>% select_best("roc_auc")
```

```
> dt_trained %>%
+   select_best("roc_auc")
# A tibble: 1 x 3
  cost_complexity min_n .config
            <dbl> <int> <chr>
1        1.21e-10    29 Preprocessor1_Model02
> rf_trained %>%
+   select_best("roc_auc")
# A tibble: 1 x 3
   mtry min_n .config
  <int> <int> <chr>
1     2    38 Preprocessor1_Model01
> xgb_trained %>%
+   select_best("roc_auc")
# A tibble: 1 x 7
  trees min_n tree_depth learn_rate loss_reduction sample_size .config
  <int> <int>      <int>      <dbl>          <dbl>       <dbl> <chr>
1   520    22         12 0.000000933  0.000000000489       0.862 Preprocessor1_Model06
> |
```

Figura 8 – Valores finais dos parâmetros

3.9.4 Vizualização gráfica dos modelos

Usando o pacote *ggplot2* podemos vizualizar gráficamente todos os hiperparâmetros dos 10 modelos ajustados segundo os três algoritmos *Decission Trees, Random Florest* e *Xgboost*.

```
ggplot2::autoplot(dt_trained)
ggplot2::autoplot(rf_trained)
ggplot2::autoplot(xgb_trained)
```

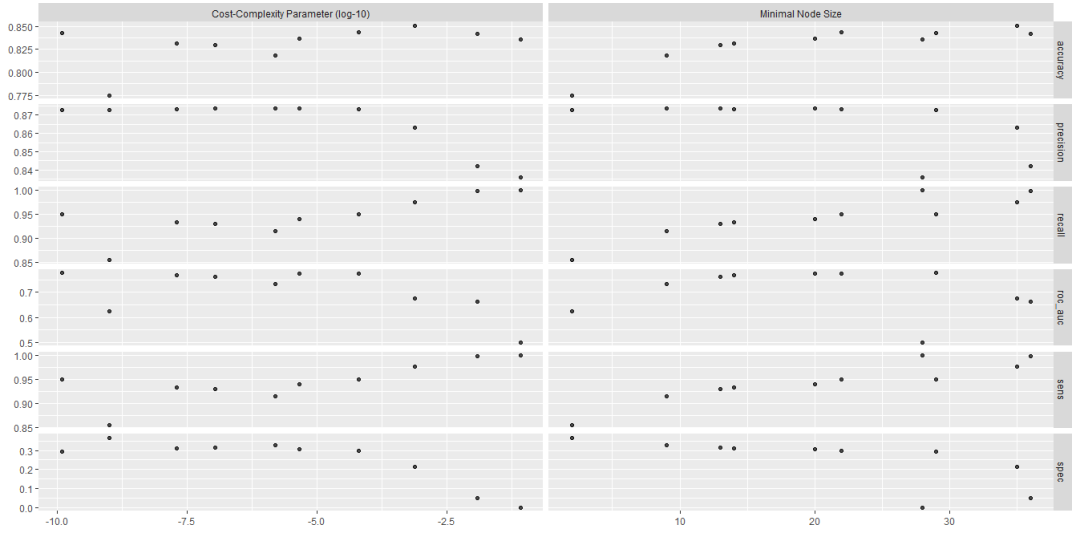


Figura 9 – Valores finais dos hiperparâmetros para o algoritmo *Decision Trees*.

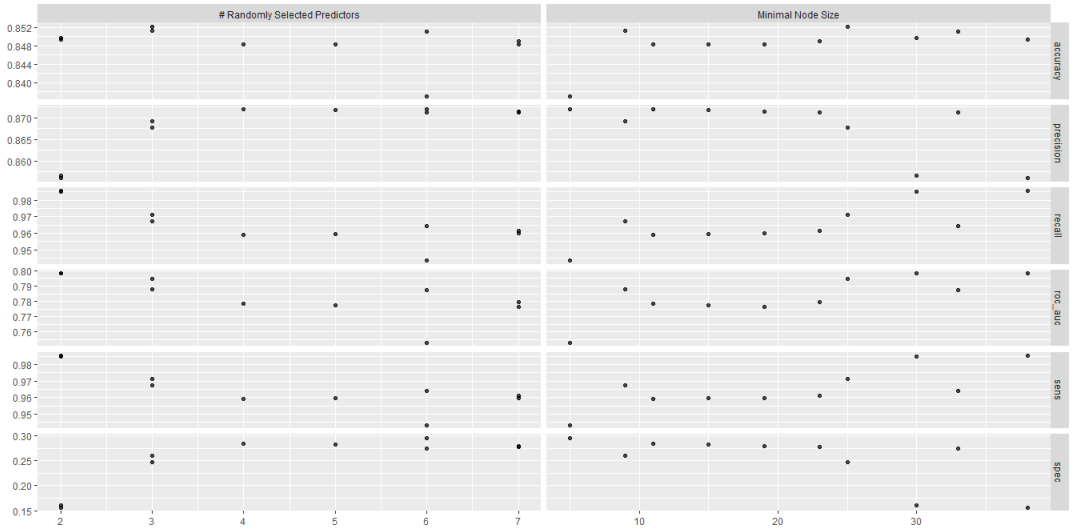


Figura 10 – Valores finais dos hiperparâmetros para o algoritmo *Random Florest*.

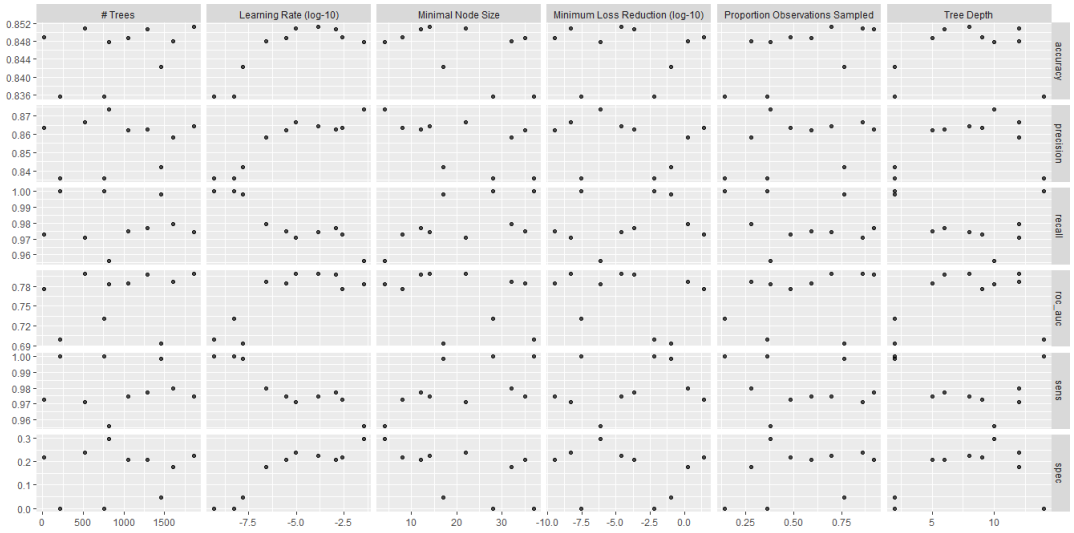


Figura 11 – Valores finais dos hiperparâmetros para o algoritmo *Xgboost*.

3.10 Finalização do workflow

Nessa etapa do processo, iremos selecionar o melhor parâmetro pela função `select_best`, no objeto `final_dt_model` é adicionado o modelo usado na análise e a função `finalize_model()` adiciona os melhores hiperparâmetros a este modelo. Quando se seleciona a função `final_dt_model` é possível observar os valores dos hiperparâmetros para cada algoritmo.

```
dt_best_tune <- select_best(dt_trained ,"roc_auc")
final_dt_model <- dt_model%>%
finalize_model(dt_best_tune)

Decision Tree Model Specification (classification)
Main Arguments:
cost_complexity = 1.20998224324819e-10
min_n = 29

Computational engine: rpart
```

```
rf_best_tune <- select_best(rf_trained , "roc_auc")
final_rf_model <- rf_model %>%
finalize_model(rf_best_tune)
Random Forest Model Specification (classification)
Main Arguments:
mtry = 2
trees = 1000
min_n = 38

Computational engine: ranger
```

```
xgb_best_tune <- select_best(xgb_trained , "roc_auc")
final_xgb_model <- xgb_model %>%
finalize_model(xgb_best_tune)
Boosted Tree Model Specification (classification)
Main Arguments:
trees = 520
min_n = 22
tree_depth = 12
learn_rate = 9.33154389598811e-06
loss_reduction = 4.88928250923649e-09
sample_size = 0.861715071944054

Computational engine: xgboost
```

3.11 Criando um novo workflow agora aplicado aos dados de teste

Após selecionar o modelo que obteve o melhor ajuste na base de treinamento, criaremos um novo `workflow()`, será adicionado pela função `add_recipe(recipe)` o recipe (pre-processamneto de dados) criado anteriormente, o modelo final selecionado no treinamento `add_model(final_xgb_model)`, no trabalho em questão o algortimo *Xgboost* obteve o melhor ajuste, depois a função `last_fit()` indica que é ultimo ajuste a ser feito, ao colocar *splits* o R já entende que é pra ser aplicado no banco de dados de teste, o comando `collect_predictions()` serve para que possamos vizualizar o valor real e as previsões, o `roc_curve(Crop_Damage, .pred_0)`

dará a area sobre a curva graficamente, como mostra a Figura 13.

```
workflow() %>%
add_recipe(recipe) %>%
add_model(final_xgb_model) %>%
last_fit(splits) %>%
collect_predictions() %>%
roc_curve(Crop_Damage, .pred_0) %>%
autoplot()
```

```
# A tibble: 19,965 x 7
   id               .pred_0 .pred_1  .row .pred_class Crop_Damage .config
   <chr>              <dbl>   <dbl> <int> <fct>       <fct>       <chr>
 1 train/test split   0.499   0.501     3 1           1           Preprocessor1_Model1
 2 train/test split   0.499   0.501     5 1           1           Preprocessor1_Model1
 3 train/test split   0.499   0.501     6 1           1           Preprocessor1_Model1
 4 train/test split   0.498   0.502    10 1           1           Preprocessor1_Model1
 5 train/test split   0.498   0.502    12 1           1           Preprocessor1_Model1
 6 train/test split   0.498   0.502    18 1           0           Preprocessor1_Model1
 7 train/test split   0.502   0.498    21 0           0           Preprocessor1_Model1
 8 train/test split   0.502   0.498    27 0           0           Preprocessor1_Model1
 9 train/test split   0.502   0.498    37 0           0           Preprocessor1_Model1
10 train/test split   0.502   0.498    45 0           0           Preprocessor1_Model1
# ... with 19,955 more rows
```

Figura 12 – Valores reais e suas predições para o modelo final *xgboost*

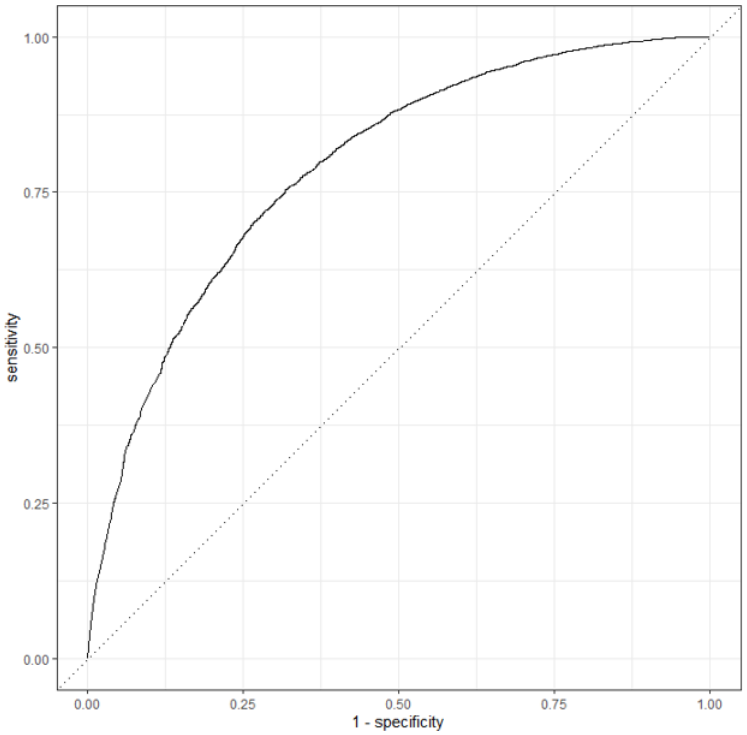


Figura 13 – Curva ROC para para o modelo final *xgboost*.

Caso se queira ver as métricas do modelo final no banco de dados de teste, basta substtuir no objeto *workflow* a função collect_predictions() por collect_metrics(), no modelo proposto o *xgboost* obteve uma acurácia de 0,85 e a área sobre a curva de 0,79.

4 CONCLUSÃO

Vários tipos de pesquisa agrícola usam técnicas de *Machine Learning* (ML) para detectar, identificar e prever doenças e danos às plantações. Atualmente, as técnicas de ML têm utilizado o conceito de árvores de decisão, *Random Forest* e classificadores baseados em *boosting* como é o caso *XGBoost*, aplicados para detectar doenças em plantas, auxiliando o agricultor na detecção automática de tipos de doenças de cultivo e danos às colheitas. O algoritmo *XGBoost* apresentou melhor desempenho na previsão de danos às culturas com uma acurácia de 0,85 e a área sobre a curva de 0,79.

REFERÊNCIAS

BRAGA, A. Curvas roc: aspectos funcionais e aplicações. 2001. Citado na página 21.

BRAZDIL, P. Construção de modelos de decisão a partir de dados. 1999. *SHIBA, MH; SANTOS, RL; QUINTANILHA, JA. Classificação de imagens de sensoriamento remoto pela aprendizagem por árvore de decisão: uma avaliação de desempenho. Disponível em:< http://marte. dpi. inpe. br/col/ltid. inpe. br/sbsr/2004/11.23*, v. 11, 2002. Citado na página 14.

BREIMAN, L. Random forests. *Machine learning*, Springer, v. 45, n. 1, p. 5–32, 2001. Citado na página 19.

CHEESEMAN, P.; STUTZ, J.; HANSON, R. *Bayesian classification theory*. 1990. Citado na página 9.

CHEN, T. et al. Xgboost: extreme gradient boosting. *R package version 0.4-2*, v. 1, n. 4, p. 1–4, 2015. Citado na página 19.

GARCÍA, J. P.; FERREIRA, J. C.; PATINO, C. M. Análise roc: uma aliada na pandemia. *Jornal Brasileiro de Pneumologia*, SciELO Brasil, v. 47, 2021. Citado na página 12.

GARCIA, S. C. O uso de árvores de decisão na descoberta de conhecimento na área da saúde. 2003. Citado na página 14.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. Deep learning. book in preparation for mit press. *URL¡ http://www. deeplearningbook. org*, v. 1, 2016. Citado na página 21.

HAYKIN, S. *Redes neurais: princípios e prática*. [S.l.]: Bookman Editora, 2007. Citado na página 11.

JR, D. W. H.; LEMESHOW, S.; STURDIVANT, R. X. *Applied logistic regression*. [S.l.]: John Wiley & Sons, 2013. v. 398. Citado na página 12.

KUHN, M.; WICKHAM, H. Tidymodels: Easily install and load the'tidymodels' packages. *R package version 0.1. 0*, 2020. Citado na página 9.

LUO, G. A review of automatic selection methods for machine learning algorithms and hyper-parameter values. *Network Modeling Analysis in Health Informatics and Bioinformatics*, Springer, v. 5, n. 1, p. 1–16, 2016. Citado na página 13.

MONARD, M. C.; BARANAUSKAS, J. A. Conceitos sobre aprendizado de máquina. *Sistemas inteligentes-Fundamentos e aplicações*, Manole Ltda, v. 1, n. 1, p. 32, 2003. Citado na página 9.

PROBST, P.; BISCHL, B.; BOULESTEIX, A.-L. Tunability: Importance of hyperparameters of machine learning algorithms. *arXiv preprint arXiv:1802.09596*, 2018. Citado na página 13.

RONCON, N. A importância do setor agrícola para a economia brasileira. *Fundação Educacional do Município de Assis–FEMA/IMESA. Assis*, p. 69, 2011. Citado na página 8.

SHENG, L. Y.; VENDRUSCULO, L. G. Um procedimento de tratamento de missing data em dados agrometeorologicos. *Proceeding Series of the Brazilian Society of Computational and Applied Mathematics*, v. 5, n. 1, 2017. Citado na página 20.

SIVAKUMAR, A.; GUNASUNDARI, R. A survey on data preprocessing techniques for bioinformatics and web usage mining. *International Journal of Pure and Applied Mathematics*, v. 117, n. 20, p. 785–794, 2017. Citado na página 20.

THERNEAU, T. et al. Package 'rpart'. *Available online: cran. ma. ic. ac. uk/web/packages/rpart/rpart. pdf (accessed on 20 April 2016)*, 2015. Citado na página 19.

VIDHYA, A. Janatahack: Machine learning in agriculture. 2020. Citado na página 10.

WICKHAM, H. et al. ggplot2: Create elegant data visualisations using the grammar of graphics. *R package version*, v. 2, n. 1, 2016. Citado na página 17.

WOLPERT, D. H. The lack of a priori distinctions between learning algorithms. *Neural computation*, MIT Press One Rogers Street, Cambridge, MA 02142-1209, USA journals-info . . . , v. 8, n. 7, p. 1341–1390, 1996. Citado na página 9.

AGRADECIMENTOS

Primeiramente agradeço ao Bom Deus, que por sua infinita bondade tem me sustentado ao longo desses anos de graduação, sem sua presença seria impossível o meu crescimento.

Agradeço aos meus pais, Lurdinez Costa e Evangelista Silva, pelo amor, incentivo e apoio, durante toda vida dedicaram a me dar o melhor que podiam. Sou grata também as minhas duas irmãs Vanessa Costa e Vaniara Costa, duas grandes incentivadoras dos meus estudos, todo meu esforço é dedicado a essas 4 pessoas.

Ao meu orientador Profº Drº Tiago Almeida de Oliveira que desde 2018 vem passando seus conhecimentos para mim de forma tão generosa, além de ser o professor que mais me incentivou durante toda graduação.

A minha co-orientadora Profª Drª Ana Patricia Bastos Peixoto de Oliveira, com ela pude desenvolver 3 PIBICs (Projeto de Iniciação Cientifica) ao longo desses anos, uma profissional excelente em quem me espelho diariamente e hoje mais do que nunca posso chamar de minha amiga.

Aos meus colegas de curso Alvaro Nascimento, Eduardo Gomes e Rafaella Santos que inciaram comigo em 2016 essa estrada, a amizade e a ajuda de vocês me fizeram vencer a cada dia. Agradeço também à Beatriz Silveira e Eva Pontes, ambas não são de minha turma de origem mas se tornaram ao longo desses anos minhas companheiras de alegria e dor. Que nossa amizade continue a crescer pelo o resto de nossas vidas.

Agradeço de forma especial a tantos amigos que me incentivaram nesse processo, destaco Pe. Ivemar Pontes, Pe. Sergio Fernando, Pe. Sandro Sebastião e Regimário Moura, os quatro sabem o quanto me ajudaram, obrigada por sempre me ouvirem e rezarem por mim.

Aos meus patrões, Francisco Frutuoso, José Robelio, João Marcolino, Josenilton de Lima e Paulo de Pierre, minha eterna gratidão por terem me possibilitado sair antes do horário deixando minhas funções no escritório para que eu pudesse assim ir a Universidade, a vossa generosidade retribuo com orações.

Agradeço à universidade Estadual da Paraíba que me permitiu trilhar o árduo, mas satisfatório caminho acadêmico e assim, realizar o sonho da graduação.

A todos os professores do Departamento de Estatística por me proporcionarem excelentes aulas e a todos que direta ou indiretamente fizeram parte da minha formação acadêmica.