UNIVERSIDADE DE CAXIAS DO SUL
ÁREA DO CONHECIMENTO DE CIÊNCIAS EXATAS E ENGENHARIAS

CARLOS EDUARDO FERREIRA PINTO

UM ESTUDO SOBRE A APLICAÇÃO DO ALGORITMO *RANDOM FOREST* EM PROBLEMAS DE CLASSIFICAÇÃO E REGRESSÃO

CAXIAS DO SUL

2024

CARLOS EDUARDO FERREIRA PINTO

UM ESTUDO SOBRE A APLICAÇÃO DO ALGORITMO *RANDOM FOREST* EM PROBLEMAS DE CLASSIFICAÇÃO E REGRESSÃO

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel em Ciência da Computação na Área do Conhecimento de Ciências Exatas e Engenharias da Universidade de Caxias do Sul.

Orientador: Profa. Dra. Elisa Boff

CAXIAS DO SUL

2024

CARLOS EDUARDO FERREIRA PINTO

UM ESTUDO SOBRE A APLICAÇÃO DO ALGORITMO *RANDOM FOREST* EM PROBLEMAS DE CLASSIFICAÇÃO E REGRESSÃO

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel em Ciência da Computação na Área do Conhecimento de Ciências Exatas e Engenharias da Universidade de Caxias do Sul.

Aprovado em 27/11/2024

BANCA EXAMINADORA

_____

Profa. Dra. Elisa Boff
Universidade de Caxias do Sul - UCS

_____

Prof. Dra. Carine Geltrudes Webber
Universidade de Caxias do Sul - UCS

_____

Prof. Dra. Helena Graziottin Ribeiro
Universidade de Caxias do Sul - UCS

AGRADECIMENTOS

Agradeço primeiramente a Deus, por me conceder saúde, força e determinação ao longo desta jornada acadêmica e aos meus pais, Antônio Carlos Dias Pinto e Iêda Ferreira Pinto, por todo o amor, incentivo e suporte emocional incondicional, fundamentais para a realização deste sonho. Sua confiança em meu potencial foi essencial para que eu pudesse superar os desafios.

Aos meus amigos da "Equipe Vida Rasa", Alisson Bertotti, Ana Luísa, Gabriela Ribeiro, Gustavo Vacari e Pamela Jantsch, assim como aos meus irmãos de outra mãe Jardel Athayde, Patrick Kugert e João Antônio e minhas amigas Larissa Camelo e Daniela Cioato. Obrigado por toda a parceria e companheirismo, porque sempre estiveram do meu lado em todos momentos.

Agradeço à pessoa que entrou na minha vida recentemente, minha namorada Maria Eduarda, que se fez presente nessa reta final da minha graduação, me dando todo o suporte e apoio possível e acreditando que eu era sim capaz de realizar esse sonho e nunca me deixou desistir.

Um agradecimento especial também para meus amigos e atuais colegas de trabalho Guilherme Mena, Niala Manfro e Ana Ciotta, vocês são incríveis. Também aos meus colegas de curso Henrique Ferreto e Gustavo Susin que foram meus parceiros em praticamente todas as disciplinas, sempre apoiando e incentivando.

Agradeço a Professora Elisa Boff, responsável pela orientação deste trabalho. Também sou grato aos Professores Carine Webber, Helena Graziottin, Daniel Notari, Andre Martinotto e Ricardo Dorneles, que me apoiaram durante toda a graduação e auxiliaram de alguma forma para a conclusão deste trabalho.

Aos demais familiares agradeço por todo apoio e carinho e por sempre acreditarem em mim. Já aos demais amigos e afins, agradeço a cada um que se esteve e está presente na minha vida. Todos que passaram e ainda continuam tem um lugar importante dentro do meu coração.

A todos vocês, o meu mais sincero e profundo agradecimento.

"Só se pode alcançar um grande êxito quando nos mantemos fiéis a nós mesmos."

Friedrich Nietzsche

RESUMO

O Aprendizado de Máquina (AM) é uma subárea da Inteligência Artificial (IA) que possui duas abordagens, sendo elas o aprendizado supervisionado e não supervisionado. Dentro desses, ainda possuem mais ramos, como por exemplo a divisão do aprendizado supervisionado em problemas de classificação e regressão, os quais possuem algoritmos próprios para aplicação e solução dos problemas. Para que o AM ocorra, podemos citar diversos algoritmos de AM como as *Artificial Neural Networks* (ANN), o *Random Forest* (RF), os *Support Vector Machine* (SVM), o *K-Nearest Neighbor* (kNN), o *Clustering*, e diversos outros, onde cada um deles possui suas particularidades, modelo de construção e aplicação e são adequados para certos tipos de problemas, em função da proposta de aprendizado de cada um deles. Portanto, este trabalho propõe a realização de uma pesquisa sobre AM e alguns de seus algoritmos, dando ênfase no algoritmo de RF e realizar a aplicação dele em três *datasets* e analisar os resultados. O desenvolvimento foi feito na linguagem R e foi utilizada a biblioteca do *Random Forest* (RF) disponível. Três *datasets* foram previamente selecionados, sendo eles de mesmas área de aplicação, porém cada um com particularidades diferentes, visando a melhor adequação dos parâmetros das funções da biblioteca para cada um dos conjuntos de dados. Por fim, os resultados foram avaliados com base nas métricas de cálculo definidas previamente, baseando-se na parametrização feita para cada umas das execuções e que, ao fim da análise, obtiveram-se resultados eficazes e sastisfatórios.

Palavras-chave: Aprendizado de Máquina. Random Forest. Classificação. Regressão.

ABSTRACT

Machine Learning (ML) is a subfield of Artificial Intelligence (AI) that encompasses two main approaches: supervised learning and unsupervised learning. Within these, there are further subdivisions, such as the division of supervised learning into classification and regression problems, each of which has its own specific algorithms for application and problem-solving. To facilitate ML, various algorithms can be cited, such as ANNs, RF, SVM, kNN, clustering, and many others. Each algorithm has unique characteristics, construction models, and applications, making them suitable for specific types of problems based on their learning framework. This study aims to conduct research on ML and some of its algorithms, with a particular focus on the RF algorithm, applying it to three datasets and analyzing the results. The development was carried out in R, using the *Random Forest* library (RF) available. Three datasets were pre-selected, all belonging to the same application domain but with distinct particularities, aiming to better tailor the function parameters of the library to each dataset. Finally, the results were evaluated based on predefined calculation metrics, considering the parameterization used for each execution. The analysis yielded effective and satisfactory results.

Keywords: Machine Learning. Random Forest. Classification. Regression.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ALGORITMOS

LISTA DE ABREVIATURAS E SIGLAS

AI Artificial Intelligence

AM Aprendizado de Máquina

ANN *Artificial Neural Networks*

AUC *Area Under the Curve*

FN Falsos Negativos

FP Falsos Positivos

IA Inteligência Artificial

IDE Ambiente de Desenvolvimento Integrado

IMC Índice de Massa Corporal

INEP Instituto Nacional de Estudos e Pesquisas Educacionais

kNN *K-Nearest Neighbor*

ML Machine Learning

MAE Erro Médio Absoluto

MSE Erro Médio Quadrático

PLN Processamento de Linguagem Natural

R^2 Coeficiente de Determinação

RMSE Raiz do Erro Médio Quadrático

RF *Random Forest*

ROC *Receiver Operating Characteristics*

SVM *Support Vector Machine*

TCC Trabalho de Conclusão de Curso

VN Verdadeiros Negativos

VP Verdadeiros Positivos

LISTA DE SÍMBOLOS

E	Conjunto de dados
E_n	Enésimo exemplo do conjunto de dados
E_i	Exemplo do conjunto de dados
$\vec{\mathbf{x}}$	Vetor de valores (atributo)
y	Classe do exemplo
$f(\vec{\mathbf{x}})$	Função conceito
$h(\vec{\mathbf{x}})$	Função hipótese
C	Conjunto discreto de classes
$C_{N_{cl}}$	Exemplo do conjunto de classes
$\mathbb{R}$	Conjunto dos números Reais
$g(\mathbf{x})$	Função
Err	Taxa de Erro
Υ	Variável
x_d	Exemplo do vetor x
$r(\mathbf{x})$	Função de regressão
mad	Diferença Média Absoluta
mse	Erro Médio Quadrático
T	Conjunto de dados de treino
a	Quantidade de atributos
n	Quantidade de exemplos
T_k	Exemplo do conjunto de dados de treino
m	Quantidade de atributos randômicos
Gini(S)	Coeficiente de *Gini*
Entropia(S)	*Entropia*

T'	Conjunto de dados de treinamento
w_i	Dado de um conjunto **x**
y_{wi}	Classe do dado *w*
k	Variável
β	Parâmetro
j	Neurônio da camada oculta
w	Vetor
θ	Ângulo formado entre os vetores
C_k	Partição do conjunto
acc	Taxa de acerto
mae	Erro Médio Absoluto
RMSE	Raiz do Erro Médio Quadrático
R^2	Coeficiente de Determinação

SUMÁRIO

1 INTRODUÇÃO

A Inteligência Artificial (IA) é uma ramificação da Ciência da Computação, que foi considerada pela primeira vez na década de 50, especificamente no ano de 1956, pelo cientista da computação norte-americano John McCarthy (1927–2011) (SILVA, 2013).

Além disso, ela é uma área que possui diversas atribuições e responsabilidades, sendo a principal delas a pesquisa e proposição da "elaboração de dispositivos computacionais capazes de simular aspectos do intelecto humano, ao modo da capacidade de raciocinar, perceber, tomar decisões e resolver problemas" (SILVA, 2013).

A partir disso, podemos perceber que a IA possui uma aplicação muito vasta, visto que, alguns dos seus exemplos de utilização estão sendo observados cada vez mais nos dias atuais, como por exemplo: no setor médico, com os diagnósticos médicos automáticos; na indústria automotiva, com máquinas dirigindo automóveis de forma totalmente autônoma; no setor do entretenimento e *e-commerce*, visto nas grandes plataformas por meio dos sistemas recomendação de séries, filmes, produtos de maneira geral, etc (LUDERMIR, 2021). E esses foram somente alguns dos setores e exemplos que poderiam ter sido citados, o que mostra a gama enorme que possui essa área da Computação.

Dentro da IA ainda há diversas subáreas, sendo uma delas o AM. Essa, tendo surgido um pouco depois, na década de 60, que possui como principal objetivo "o desenvolvimento de técnicas computacionais sobre o aprendizado, bem como a construção de sistemas capazes de adquirir conhecimento de forma automática" (MONARD; BARANAUSKAS, 2003). Além disso, Monard e Baranauskas (2003) também citam que os sistemas de aprendizado são softwares que realizam escolhas baseado no conhecimento adquirido após resoluções corretas de problemas já solucionados no passado.

Ademais, existem ainda algumas subdivisões do AM que são estreitamente relacionadas a parte de mineração de dados e estatística. Também há uma subdivisão relacionada a parte de pesquisa, a qual tem o foco na identificação das características dos métodos e da exigência computacional do AM. Sua utilização direta se dá desde o "Processamento de Linguagem Natural (PLN), motores de busca, diagnósticos médicos, bioinformática, reconhecimento de fala, reconhecimento de escrita, visão computacional, locomoção de robôs e sistemas de previsão" (AMORIM; BARONE; MANSUR, 2008).

Como um exemplo básico da AM, é possível citar um programa desenvolvido para executar uma tarefa simples: identificar variedades de uma flor de uma mesma espécie. Basicamente, assim como nós humanos aprendemos a diferenciar com base nas características, o algoritmo de AM seria produzido da mesma forma, onde por meio de um processo de aprendizagem, consegue realizar a categorização da flor com base nas suas propriedades (CERRI;

FERREIRA *et al.*, 2019).

Por conseguinte, é possível citar os algoritmos atualmente mais conhecidos de Aprendizado de Máquina (AM): *Support Vector Machine* (SVM); *Random Forest* (RF); *K-Nearest Neighbor* (kNN); *Artificial Neural Networks* (ANN); entre diversos outros; os quais são capazes de adquirir conhecimento de maneira automática a partir de conjuntos de dados ofertados (ALBA *et al.*, 2022). Complementando, segundo Teodoro e Kappel (2020), Redes Neurais e ANN são técnicas que podem ser melhor aproveitadas para grandes processamentos de dados enquanto Árvores de Decisão e RF são mais adequadas para identificação de atributos determinantes dentro dos conjuntos de dados. Assim, nesta pesquisa será realizada uma análise detalhada sobre um dos algoritmos de AM, o RF, visando melhor aplicação do mesmo nos diferentes conjuntos de dados selecionados para exemplo.

1.1 OBJETIVOS

O principal objetivo deste trabalho consiste em estudo sobre AM, focando no algoritmo Random Forest, e aplicando-o em diferentes datasets, a fim de adequar os parâmetros do método para melhor aplicação e aproveitamento do mesmo em cada caso. Com base no objetivo geral, foram definidos os seguintes objetivos específicos:

- Apresentar os fundamentos dos métodos de Aprendizado de Máquina (AM);
- Apresentar os algoritmos de AM mais conhecidos, com ênfase no RF a fim de entender os cenários e problemas de aplicação do algoritmo RF;
- Desenvolver uma aplicação para comparar o comportamento do RF em diferentes datasets;
- Verificar e comparar valores atingidos pelas métricas após a aplicação do algoritmo nos *datasets*.

1.2 ESTRUTURA DO TRABALHO

O presente trabalho está organizado da seguinte forma:

- O Capítulo 2 apresenta a pesquisa do referencial teórico sobre aprendizado de máquina, evidenciando os algoritmos mais conhecidos, juntamente com seus fundamentos e características, e quais problemas podem solucionar.
- O Capítulo 3 apresenta trabalhos relacionados ao assunto desta monografia e que serão auxiliadores na construção da proposta de solução.

- O Capítulo 4 apresenta a proposta de solução sobre parametrização do método com base em cada um dos Datasets selecionados.
- O Capítulo 5 apresneta os resultados obtidos com a aplicação do algoritmo e quais foram as conclusões chegadas.
- O Capítulo 6 apresenta as considerações finais e as ideias para possíveis trabalhos, pesquisas e projetos.

2 APRENDIZADO DE MÁQUINA

Aprendizado de Máquina AM, popularmente conhecido como *machine learning*, é uma subdivisão da IA que, em sua essência, surgiu com uma finalidade exclusivamente voltada para a área de Computação. Esse conceito acabou mudando um pouco antes da virada do século, cuja área acabou por estabelecer conceitos próprios e passou a ter diversas interações com outras áreas do conhecimento e não somente voltada para a área da informática (IZBICKI; SANTOS, 2020).

Além disso, para Mitchell (1997) o principal objetivo da AM "é a construção de programas que melhorem seu desempenho por meio de exemplos". Para que isso se concretize, é necessária uma gama grande de dados para que a máquina realize o aprendizado, que nada mais é do que suposições que são geradas a partir dos dados.

Desta forma, seguindo pela ordem de classificação do AM, na Seção 2.1 serão apresentados os tipos de aprendizado. Posteriormente, na Seção 2.2 serão apresentadas as classificações dos algoritmos de AM. Por fim, na Seção 2.3 serão listados e explicados os algoritmos mais conhecidos, que são mais utilizados atualmente.

2.1 TIPOS DE APRENDIZADO

Assim como nas diversas outras áreas, o AM possui também suas ramificações (Figura 1) que, segundo Monard e Baranauskas (2003), são: Supervisionado e Não Supervisionado.

Figura 1 – Hierarquia do Aprendizado de Máquina

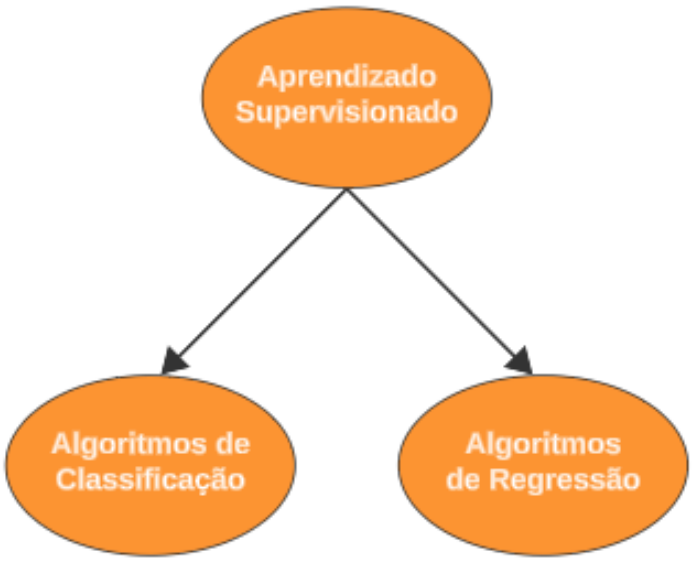
Fonte: O Autor (2024).

2.1.1 Aprendizado Supervisionado

Os algoritmos de aprendizagem supervisionada, de maneira geral, funcionam associando uma saída com uma entrada, baseados em dados previamente rotulados, cujos atributos podem ser valores numéricos ou até mesmo classes pré-definidas. Para que esses realizem o aprendizado de forma correta, "o usuário alimenta o algoritmo pares de entradas e saídas conhecidos, normalmente na forma de vetores" (FONTANA, 2020). Dessa forma, é possível entender-se que, os algoritmos aprendem com base em dados já conhecidos para assim determinar padrões existentes entre os mesmos, assim conseguindo realizar a classificação correta dos próximos casos que serão analisados por ele.

Além disso, Fontana (2020) define os dois tipos de algoritmos de aprendizado supervisionado (Figura 2), Classificação e de Regressão. No algoritmo de Classificação o resultado pode ser somente uma resposta pertencente a um conjunto de rótulos definidos previamente, não sendo um valor aleatório. O algoritmo de Regressão é praticamente o oposto do de Classificação, no qual as saídas podem ser valores reais, visto que o algoritmo atua na previsão dos valores com base nas condições dos dados de entrada. O primeiro tipo será explicado mais detalhadamente na Seção 2.2.1, assim como o segundo na Seção 2.2.2.

Figura 2 – Hierarquia do Aprendizado Supervisionado



Fonte: O Autor (2024).

É imprescindível, para que algoritmos de aprendizagem supervisionada sejam utilizados de maneira eficaz, que os dados estejam formatados de uma maneira que o algoritmo possa realizar a interpretação de maneira correta. Nesse sentido, ocorre de que em diversas vezes é necessário realizar a transformação dos dados, como, por exemplo, por realizar a alteração das informações qualitativas em informações quantitativas, assim permitindo a realização de operações numéricas. Ademais, quando os atributos possuem escalas muito diferentes, normalizar os dados, ou seja, ajustá-los para a mesma escala, pode garantir uma melhoria significativa no desempenho do algoritmo (FONTANA, 2020).

Para demonstrar da melhor maneira o aprendizado supervisionado, Batista *et al.* (2003) utiliza fórmulas para representar e exemplificar.É fornecido ao algoritmo o conjunto de dados $E = \{E_1, E_2, ...E_N\}$, onde cada exemplo $E_i \in E$ possui um rótulo, o qual define qual classe o dado pertence. O exemplo $E_i \in E$ é definido como:

$$E_i = (\vec{x_i}, y_i). \tag{2.1}$$

Na fórmula acima, $\vec{x_i}$ representa um vetor de valores que apresentam os atributos do exemplo E_i, enquanto y_i é a variável que representa qual a classe do exemplo. Dessa forma, o objetivo do aprendizado supervisionado é construir um modelo $y = f(\vec{x})$, onde a função f é chamada de função conceito, que possa ser possível prever as classes y para exemplos que não foram visualizados anteriormente (BATISTA *et al.*, 2003). Além disso, segundo Batista *et al.* (2003), normalmente a quantidade de dados fornecidos para que esse modelo seja construído não é suficiente para realizar uma categorização completa da função *f*. Nesse caso, é criada uma

função *h*, denominada *hipótese*, sobre a função *f*, que é uma fórmula próxima da original, sendo representada pela expressão $h(\vec{x}) \approx f(\vec{x})$.

O atributo *Y*, que se refere a classe pode ser um valor qualitativo, assumindo um valor de um conjunto discreto $C = \{C_1, C_2, ...C_{N_{cl}}\}$ ou um valor dentro de um conjunto de valores reais ($\mathbb{R}$) (BATISTA *et al.*, 2003).

Adicionalmente, um conjunto de exemplos normalmente é separado em subconjuntos, um denominado de conjunto de treinamento e o outro chamado de conjunto de teste. O primeiro é empregado para que o algoritmo realize a parte do aprendizado, enquanto o segundo é aplicado para realizar a medição da efetividade do conhecimento adquirido pelo algoritmo. A separação dos subconjuntos dá-se pelo fato de que é necessário confirmar que os resultados obtidos pelo conjunto de teste sejam estatisticamente autênticos, que só é possível se os dados de teste utilizados sejam diferentes dos dados de treino (MONARD; BARANAUSKAS, 2003).

2.1.2 Aprendizado Não Supervisionado

Enquanto no aprendizado supervisionado o algoritmo já recebe exemplos com os rótulos pré-estabelecidos, no aprendizado não supervisionado os dados são fornecidos sem as classes. Dessa forma, o algoritmo se responsabiliza por realizar a junção dos dados com base nas características em comum entre os atributos fornecidos. Ademais, também segundo Ludermir (2021), o algoritmo ainda realiza a análise do que lhe foi entregue, tentando determinar se alguns deles podem ser agrupados de alguma forma, criando dessa maneira os chamados de "agrupamentos ou *clusters*". Logo, também se faz necessário um estudo para determinar se os agrupamentos se encaixam dentro do problema que foi proposto (LUDERMIR, 2021).

Os algoritmos do método de aprendizado não supervisionado precisam definir por si mesmos a quantidade de classes e os atributos fora do comum. Esse processo é chamado de *clustering*, cujos métodos tendem a apresentar resultados favoráveis quando as classes estão bem separadas no espaço de atributos e, se as mesmas estiverem sobrepostas, ou seja, quando os objetos possuem semelhanças, possui mais dificuldades (DOMINGUES, 2003; COSTA, 2001).

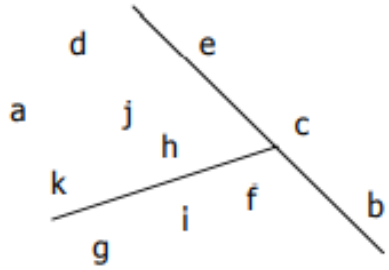
Também no artigo de Domingues (2003) há o complemento de que, normalmente, os agrupamentos são realizados em bancos de dados que possuem grande quantidade de dados, nos quais são possíveis verificar diversos e "diferentes tipos de dados em classes ou grupo de objetos", que são agrupados por algum nível de semelhança, sempre baseando nos atributos dos mesmos. Além disso, após a realização desses agrupamentos, será possível utilizá-los como referência para a realização de classificações novas.

A excelência do resultado depende, além das medidas de semelhança utilizadas pelo algoritmo, de sua parametrização e maneira de codificação, mas também da capacidade que ele possuirá de reconhecer os padrões mascarados entre as classes. A semelhança intraclasse sendo alta e a interclasse sendo baixa também são requisitos utilizados para definir que o agrupamento

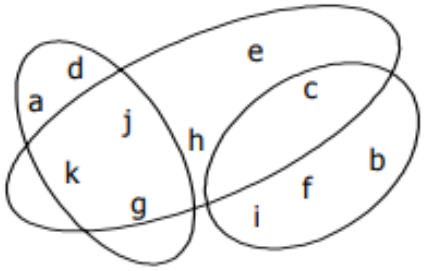
foi realizado com alta eficiência (DOMINGUES, 2003; LOPES, 1999).

Existem 4 grupos, para Witten e Frank (2005), para representar os resultados dos agrupamentos (Figura 3): exclusivos (Figura 3a), sobrepostos (Figura 3b), probabilísticos (Figura 3c) e hierárquicos (Figura 3d). Para o grupo dos exclusivos, as ocorrências pertencem exclusivamente a um único grupo; nos sobrepostos, uma ocorrência pode pertencer a mais de um grupo; para o grupo dos probabilísticos, cada ocorrência pertence a cada grupo possuindo certa probabilidade e, por fim, o grupo dos hierárquicos, onde é realizada uma partição das ocorrências em grupos maiores, os quais são tratados, dividindo-se em grupos menores.

Figura 3 – Diferentes formas de representação de agrupamentos



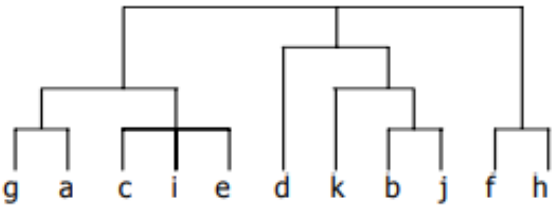
(a) Exclusivos



(b) Sobrepostos

	1	2	3
a	0,4	0,1	0,5
b	0,1	0,8	0,1
c	0,3	0,3	0,4
d	0,1	0,1	0,8
e	0,4	0,2	0,4
f	0,1	0,4	0,5
g	0,7	0,2	0,1
h	0,5	0,4	0,1
...			

(c) Probabilísticos



(d) Hierárquicos

Fonte: Witten e Frank (2005).

O aprendizado não supervisionado é alimentado apenas de um conjunto de exemplos *E*, que possui somente os vetores $\vec{x}$, não recebendo previamente qual o valor da classe *y*. Dessa forma, Batista *et al.* (2003) resume que o foco principal do aprendizado não supervisionado é formar agrupamentos ou *clusters* com os exemplos que possuam propriedades parecidas, construindo um modelo que realiza a pesquisa das similaridades entre os exemplos.

Assim como no aprendizado supervisionado, os algoritmos de aprendizado não supervisionado recebem um conjunto de dados $E = \{E_1, E_2, ...E_N\}$, porém, conforme também já citado, sem o atributo classe *y* associado ao dado (BATISTA *et al.*, 2003).

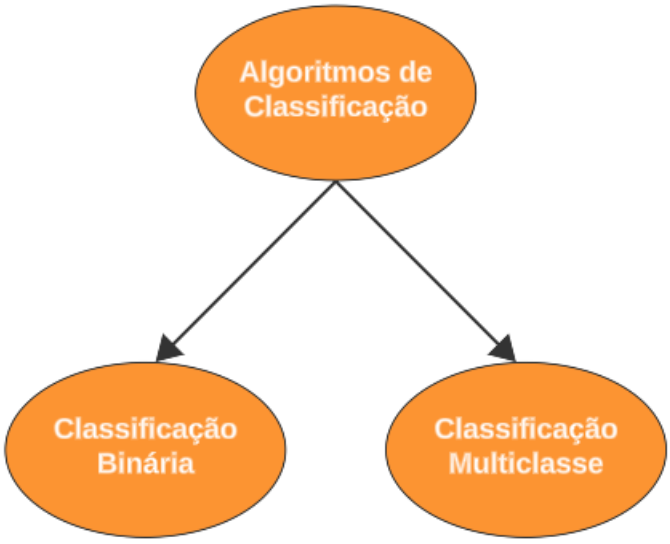
2.2 TIPOS DE PROBLEMAS

Existem, dentro do aprendizado supervisionado, dois tipos de problemas, os de Classificação e Regressão e, na Seção 2.2.1 e Seção 2.2.2 serão detalhados e explicados.

2.2.1 Classificação

Os métodos de classificação são algoritmos de aprendizado supervisionado onde, segundo Fontana (2020), o objetivo "é prever uma classe ou rótulo associado com uma variável de entrada contendo determinados atributos". O autor também cita as etapas para o melhor funcionamento do algoritmo, onde o mesmo passa pelo processo de treinamento, utilizando de exemplos onde as classes já são conhecidas, as quais possuem duas subclassificações. A Classificação Binária, onde há somente duas classes entre todo o conjunto de dados, e a Classificação Multiclasse, possuindo várias classes dentro do conjunto (Figura 4).

Figura 4 – Hierarquia Algoritmos de Classificação



Fonte: O Autor (2024).

Para exemplificar de maneira simplificada uma atividade de classificação, Cerri, Ferreira *et al.* (2019) utilizam o exemplo de um sistema de recomendação de crédito, onde basicamente um banco desenvolve sistema que realize a divisão dos seus clientes em duas categorias, visando a verificação se o cliente pode ou não fazer um empréstimo, por meio de uma classificador SIM e NÃO. Esse sistema utiliza métricas como o histórico de crédito, salário e tempo de emprego, que seriam, nesse caso, os dados de entrada do modelo, e o algoritmo realizaria o aprendizado para diferenciar quais são os clientes que o banco pode ofertar um empréstimo.

Nos problemas de classificação, para que se conclua o objetivo de se construir uma função $g(\boldsymbol{x})$ que consiga realizar a rotulação de observações novas, ou seja, $(\mathbf{X_{n+1}}, \Upsilon_{n+1}), ..., (\mathbf{X_{n+m}}, \Upsilon_{n+m})$,

são considerados exemplos que possuem observações autônomas $(\mathbf{X_1}, \Upsilon_1), ..., (\mathbf{X_n}, \Upsilon_n) \sim (\mathbf{X}, \Upsilon)$. De maneira geral, a fórmula que busca-se chegar é a seguinte:

$$g(\mathbf{x_{n+1}}) \approx y_{n+1}, ..., g(\mathbf{x_{n+m}}) \approx y_{n+m}. \tag{2.2}$$

Complementando, a diferença entre os problemas de classificação e regressão é que nos algoritmos de classificação a variável responsável pela rotulação (Υ) é uma variável qualitativa (IZBICKI; SANTOS, 2020).

Para que sejam realizadas as medidas de erro entre os valores verdadeiros e os valores cujo algoritmo realizou a previsão, utiliza-se, segundo Weiss e Indurkhya (1998), a fórmula da taxa de erro, Err, definida por:

$$Err = \frac{\sum_{i=1}^{N} \varepsilon(\mathbf{h}(E_i), f(E_i))}{N} \tag{2.3}$$

onde $\varepsilon(a, b)$ = 1 se a $\neq$ b. Caso a = b, então $\varepsilon(a, b)$ = 0 (BATISTA *et al.*, 2003).

Por fim, podemos citar alguns exemplos de algoritmos de classificação, como o RF (Seção 2.3.1), kNN (Seção 2.3.2), SVM (Seção 2.3.4), Árvores de Decisão e *Naive Bayes*, os quais, os 3 primeiros serão aprofundados nas seções indicadas, respectivamente.

2.2.2 Regressão

A história dos métodos de regressão começa há mais de 200 anos, com os matemáticos Legendre (1806) e Gauss (1877), os quais realizaram estudos utilizando o método dos mínimos quadrados, visando realizar a previsão de órbitas ao redor do Sol (IZBICKI; SANTOS, 2020).

Segundo Izbicki e Santos (2020), a principal função de um algoritmo de regressão é determinar a conexão entre uma variável aleatória $\Upsilon \in \mathbb{R}$ e um vetor $\mathbf{x} = (x_1, ..., x_d) \in \mathbb{R}$. Em resumo, a função de regressão é definida como:

$$r(\mathbf{x}) := \mathbb{E}[\Upsilon | \mathbf{X = x}] \tag{2.4}$$

onde, se o problema for considerado de regressão, o valor de Υ será quantitativo.

Assim como para os algoritmos de classificação, Weiss e Indurkhya (1998) também evidenciaram medidas para que se possa calcular o erro entre valores reais e valores preditos. Nesse caso, para os algoritmos de regressão, as medidas mais utilizadas são: Diferença Média Absoluta (mad), definida pela equação:

$$mad = \frac{1}{N} \sum_{i=1}^{N} |(E_i) - f(E_i)| \tag{2.5}$$

e Erro Médio Quadrático (mse), definido pela equação:

$$mse = \frac{1}{N}\sum_{i=1}^{N}((E_i) - f(E_i))^2. \tag{2.6}$$

Dentre ambas as equações, o valor do erro, se utilizarmos a Diferença Média Absoluta (mad) é menor do que a raiz quadrada do resultante do erro Erro Médio Quadrático (mse) (BATISTA *et al.*, 2003).

Ademais, Izbicki e Santos (2020) citam que existem dois motivos para que se faça uma análise de regressão: *motivo inferencial* e o *motivo preditivista*. Esse último, que é utilizado quando o objetivo é realizar boas predições nos dados e aquele que é recomendado de se utilizar quando se faz necessário chegar em resultados sobre as populações que os dados se encontram.

São chamados de algoritmos de regressão, segundo Fontana (2020), "os algoritmos de aprendizagem que relacionam um conjunto de atributos de entrada com uma ou mais saídas contínuas", sendo que essas saídas tem a possibilidade de assumir um valor aleatório dentro de um espaço de valores.

Para exemplificar esse tipo de modelo, podemos utilizar um ajuste de pontos a uma curva *y = f(x)*, onde é possível relacionar somente um atributo *x* com um resultado *y* ou, assim como nos algoritmos de classificação, há a possibilidade de relacionar a saída com vários atributos dentro de um vetor $\vec{\mathbf{x}}$ (FONTANA, 2020).

Nos algoritmos de regressão, ao contrário dos de classificação, o objetivo é prever qual será o atributo de saída, com base em outros atributos de entrada. Portanto, ao invés de rotular o dado e atribuir uma classe a ele, o algoritmo busca "encontrar uma função que mapeie um exemplo para um número" (CERRI; FERREIRA *et al.*, 2019).

Finalmente, é possível elencar alguns algoritmos de regressão, como o kNN (Seção 2.3.2), SVM (Seção 2.3.4).

2.3 ALGORITMOS DE APRENDIZADO DE MÁQUINA

Nesta seção é apresentado de maneira detalhada alguns algoritmos de AM sendo eles: RF, kNN, ANN, SVM e *Clustering*.

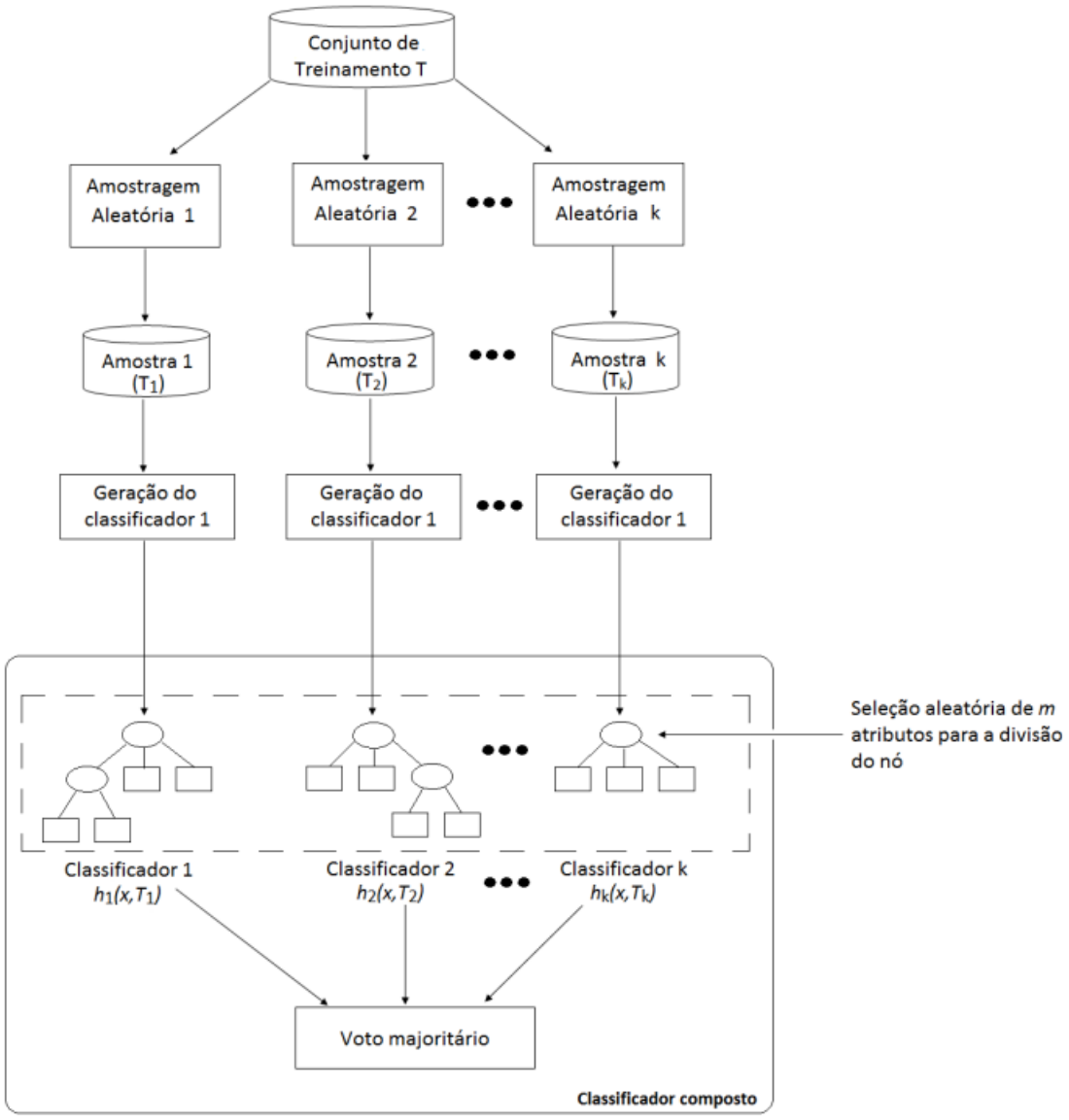
2.3.1 *Random Forest*

Basicamente, o algoritmo *Random Forest* (RF) é um dos algoritmos de aprendizado supervisionado, que pode ser utilizado para resolução de problemas tanto de classificação quanto de regressão. Em sua metodologia referente a um classificador, realiza a geração de agrupamentos de diversas Árvores de Decisão, onde cada uma dessas contém características individuais e diferentes umas das outras (LEITE; MORAES; LOPES, 2020; TEODORO; KAPPEL, 2020; BREIMAN, 2001).

RF é considerado um método *ensemble*, que como já foi dito, realiza a construção de várias árvores de decisão. Essas serão utilizadas para realizar, para um novo dado, a classificação do mesmo através do chamado "voto majoritário"e, cada uma delas, utiliza de um subconjunto de atributos que são escolhidos de maneira randômica baseados nos atributos que estão no conjunto original de dados (OSHIRO, 2013).

Para exemplificar a metodologia de utilização do RF, podemos considerar um conjunto de dados para treino *T*, que possui *a* atributos e *n* exemplos, onde T_k é uma amostra do conjunto, o qual possui *n* exemplos e utiliza *m* atributos randômicos, tal qual $m \leq a$ em cada um dos subconjuntos das árvores. Essa condição pode ser vista melhor na Figura 5 (OSHIRO, 2013).

Figura 5 – Funcionamento do Método *Random Forest*



Fonte: Adaptado de Oshiro (2013).

Oshiro (2013) explica a Figura 5 afirmando que, na RF, o conjunto de dados de trei-

namento é montado referindo-se no conjunto de treinamento original. Assim sendo, o restante da árvore é criada utilizando o próprio novo subconjunto e com uma junção de características aleatórias. Dessa forma, para cada nó da árvore, é atribuído um subconjunto de *m* características (sendo esse valor fixo para todos os nós) para que seja possível realizar a avaliação. Após isso, a melhor característica é selecionada para que o nó seja subdividido novamente.

Lima *et al.* (2021) citam em seu artigo que o algoritmo de RF provê a diversidade, baseado na distribuição de maneira randômica e aleatória dos dados. Dessa forma, cada árvore criada pelo algoritmo é treinada com atributos provindos de uma distribuição aleatória. Completa também que uma das maiores qualidades do algoritmo "é a facilidade para se medir a importância relativa de cada atributo para a predição", ou seja, o algoritmo é capaz de listar quais características são mais expressivas no momento da tomada de decisão (BREIMAN, 2001).

Como o algoritmo utiliza as folhas de cada nó para chegar a uma definição final, a precisão do modelo é aumentada, visto que são utilizados os resultados de muitas árvores de decisão distintas entre si, sendo necessário encontrar uma média. Essa grande grande quantidade de árvores distintas que são geradas acaba por fazer com que o algoritmo possua mais eficiência para conjuntos com um grande número de dados (LEITE; MORAES; LOPES, 2020; OLSON; DELEN, 2008).

Nos problemas de classificação, o algoritmo realiza a aplicação da fórmula do coeficiente de *Gini* com a finalidade de identificar qual das ramificações tem maior probabilidade de acontecer, analisando a variação dos dados, dessa forma sendo possível calcular a impureza em um nó. A fórmula que define esse coeficiente dá-se:

$$Gini(S) = 1 - \sum_{i=1}^{c} (p_i)^2. \tag{2.7}$$

Nesse caso, o p_i é a frequência relativa da classe do referido conjunto de dados *S* e *c* representa o número de classes (LEITE; MORAES; LOPES, 2020).

Outra fórmula muito utilizada dentro dos algoritmos de RF é a da *Entropia*. Com ela, é possível analisarmos a impureza ou não homogeneidade do conjunto de dados, analisando-se da seguinte forma: se ela for zero significa que o conjunto de dados é totalmente homogêneo; se ela for um, significa que ele é igualmente dividido. A fórmula para se calcular a *entropia* é:

$$Entropia(S) = \sum_{i=1}^{c} p_i log_2(p_i). \tag{2.8}$$

Além disso, a *entropia* está fortemente relacionada ao ganho de informação, visto que o ganho de informação está diretamente ligado com a redução desse coeficiente. Sendo assim, pode-se definir que montar uma árvore de decisão implica em descobrir o atributo que traz o maior ganho de informação (LEITE; MORAES; LOPES, 2020).

2.3.2 *K-Nearest Neighbor*

O kNN, cuja tradução k-Vizinhos mais Próximos, é um dos algoritmos mais comuns para se utilizar em problemas de classificação. Basicamente esses algoritmos funcionam memorizando os dados que foram utilizados para treinamento e prevendo qual será a classe de um novo dado, baseando-se na classe dos vizinhos mais próximos desse novo dado, visto que, o algoritmo assume que os vizinhos próximos tem uma chance maior de possuírem a mesma classificação. Dessa forma, durante a parte de treinamento do algoritmo não é necessário alterar a parametrização do algoritmo para o modelo (FONTANA, 2020).

Para exemplificar a utilização do algoritmo, Fontana (2020) define o conjunto de treinamento *T*, o qual possui *m* vetores **x**, os quais possuem a sua respectiva classificação *y*:

$$T = (\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), ..., (\mathbf{x_m}, y_m). \tag{2.9}$$

Para que seja determinado qual será a classe do de um novo dado **x'**, é necessário descobrir qual a distância do novo dado em relação aos dados do grupo de treinamento. Logo em seguida, é feita a ordenação dos dados, baseado na distância e definido por um novo grupo *T'*:

$$T = (\mathbf{w_1}, y_{w1}), (\mathbf{w_2}, y_{w2}), ..., (\mathbf{w_m}, y_{wm}). \tag{2.10}$$

Nesse caso, a variável $\mathbf{w}_i$ refere-se a um dos dados $\mathbf{x}_i$ e y_{wi} corresponde a classe desse dado. A fórmula que define qual será a ordenação dos dados é:

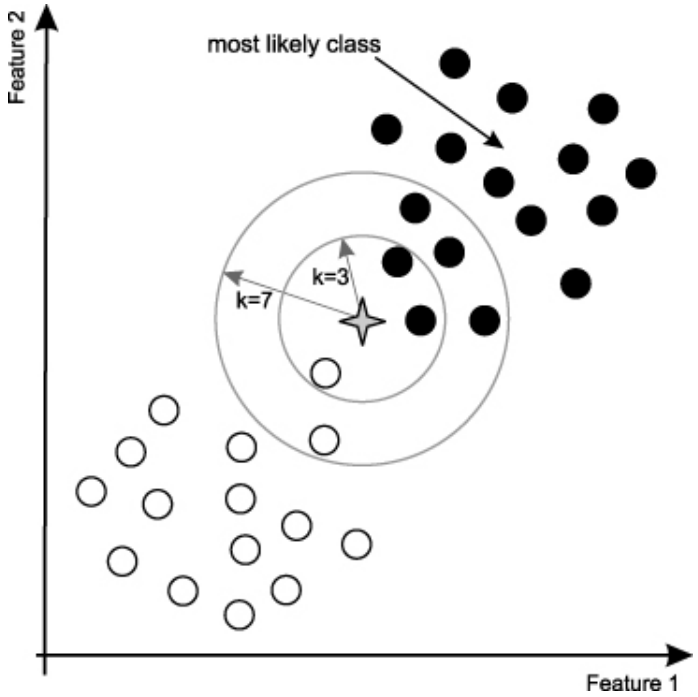
$$f(\mathbf{w_i}, \mathbf{x'}) \leq f((\mathbf{w_{i+1}}, \mathbf{x'}). \tag{2.11}$$

Essa fórmula é lida da seguinte maneira: "o elemento $\mathbf{w_i}$ está mais próximo de **x'** do que o elemento $\mathbf{w_{i+1}}$ (FONTANA, 2020).

Também, é importante citar um dos pré-requisitos para que seja estabelecida a classificação pelo método kNN, os classificadores por distância. Dentro desse conceito, existem algumas métricas de distância: Métrica de Distância Euclidiana, Métrica de Distância Euclidiana Quadrada, Distância Manhattan, Distância Chebyshev (FUJIKAWA, 2016; RADY, 2011).

Segundo FUJIKAWA (2016), o único parâmetro que é livre para ser alterado é o parâmetro que define o número de vizinhos que será feita a verificação. Esse parâmetro é alterado pelo próprio usuário, visando melhorar a eficiência do resultado da classificação. Para ilustrar isso, é possível verificar a Figura 6, onde a variável k foi escolhida para ser a variável do número de vizinhos e, para $k = 3$, há 2 dados da classe em preto, enquanto há 1 dado da classe em branco. Se aumentado e setado o $k = 7$, podemos perceber que há 5 dados da classe preta para 2 da classe branca. Nesse caso, é possível inferir que o novo dado classificado será enquadrado dentro da classe preta, devido a essa classe possuir mais dados vizinhos com o novo elemento.

Figura 6 – Imagem ilustrando um Classificador *k-Nearest Neighbor*



Fonte: FUJIKAWA (2016), Momin e Pardeshi (2013).

2.3.3 *Artificial Neural Networks*

As ANN's são um conceito bem antigo, que foi falado pela primeira vez no trabalho de McCulloch e Pitts (1943) e que foi continuado e completado quando foi apresentado o *Perceptron*, de Rosenblatt (1958), o qual realizou a comprovação do teorema de convergência. Basicamente as ANN's "são construções matemáticas relativamente simples que foram inspiradas no modelo biológico do sistema nervoso". O processamento das informações dentro das ANN's é realizado nos chamados neurônios artificiais, popularmente conhecidos por Neurônio McCulloch e Pits ou modelo MCP (BATISTA *et al.*, 2003; MCCULLOCH; PITTS, 1943).

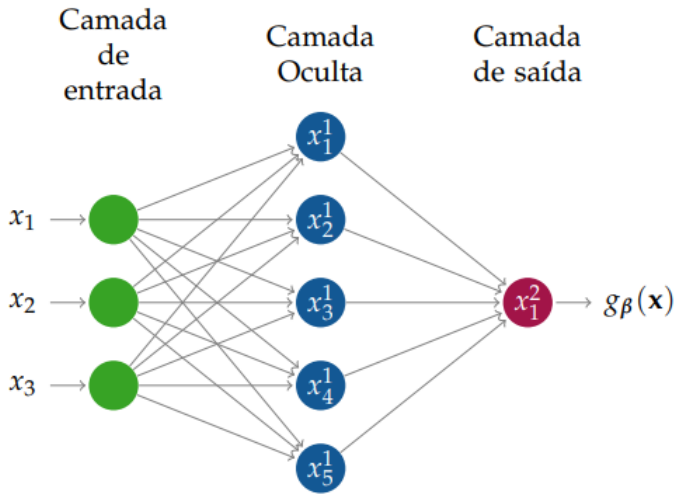
Segundo Cerri, Ferreira *et al.* (2019), as ANN's, assim como o cérebro humano, possuem "neurônios artificias organizados em camadas". Funcionam basicamente aplicando uma expressão matemática a partir do que recebe como sinal de entrada, os quais podem ser tanto os próprios dados ou resultados vindos de outros neurônios. O aprendizado fica guardado nos chamados pesos das conexões que existem entre os neurônios, que, com o decorrer do aprendizado da rede, vão sendo alterados e ajustados.

Com as ANN's é possível realizar o processamento de uma quantidade grande de dados e também encontrar padrões que não são visualizados facilmente por humanos. Estes padrões permitem gerar modelos computacionais para realizar classificações mais precisas e identificar atributos mais determinantes no momento da classificação (TEODORO; KAPPEL, 2020; GISLASON; BENEDIKTSSON; SVEINSSON, 2006; VLAHOU *et al.*, 2003).

Tratando-se, dentro de um contexto de regressão, as ANN's é um estimador não linear

$r(\mathbf{x})$, o qual podemos ver a estrutura conforme a Figura 7.

Figura 7 – Exemplo de rede neural com uma camada oculta



Fonte: Izbicki e Santos (2020).

Nesse caso, os nós que estão a esquerda da figura são as entradas da rede, as quais representam as variáveis do banco de dados. Na segunda coluna estão os nós que são conhecidos como nós da camada oculta e, cada seta é um parâmetro β. Dentro dessa camada, cada nó é nada mais nada menos que uma alteração dos nós que estavam nas camadas anteriores (IZBICKI; SANTOS, 2020).

No exemplo da rede da Figura 7, podemos definir um neurônio *j* da camada oculta pela função:

$$x_j^1 := f\left(\beta_{0,j}^0 + \sum_{i=1}^{3} \beta_{i,j}^0 x_i^0\right) \tag{2.12}$$

onde $x_i^0 = x_i$ se a variável *i* = 1,2,3. Ademais, f é a chamada função de ativação, sendo essa definida pelo próprio usuário. Após definidos os valores de x_j^1 em todos os neurônios da camada oculta, é possível então definir a saída do modelo, a qual, para a Figura 7, é dado pela equação:

$$x_1^2 := f\left(\beta_{0,j}^1 + \sum_{i=1}^{5} \beta_{i,1}^1 x_i^1\right) =: g_\beta(\mathbf{x}). \tag{2.13}$$

2.3.4 *Support Vector Machine*

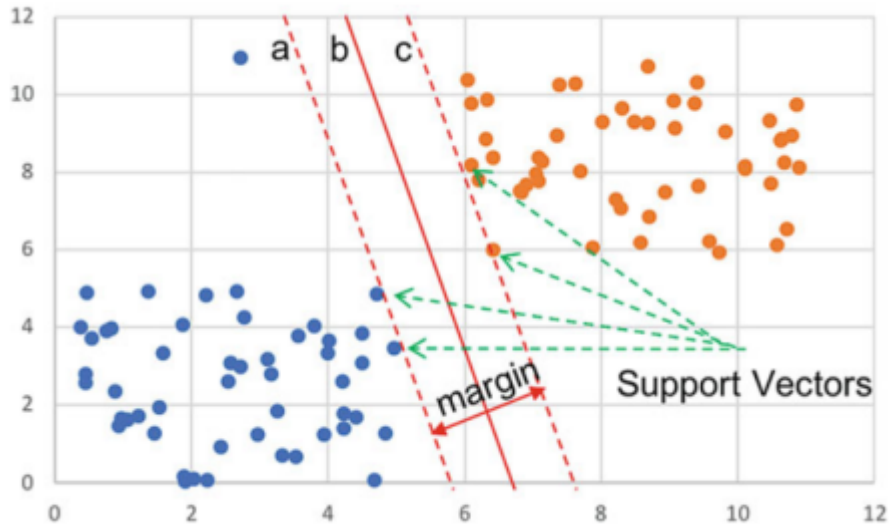
Segundo Cerri, Ferreira *et al.* (2019), SVM's são utilizados tanto para problemas de classificação quanto para problemas de regressão. Funcionam basicamente elevando a separação entre exemplos nos problemas binários. Esse algoritmo é capaz de realizar o mapeamento de "exemplos para um espaço de dimensão superior", cuja classificação se torna facilitada.

Ao utilizar os algoritmos baseados em SVM's, tem-se o objetivo de encontrar, segundo Fontana (2020), "qual é o plano de separação entre os grupos que esteja mais distante possível dos elementos de cada grupo", onde seja possível registrar uma separação que fica exatamente entre as duas classes.

Ademais, os algoritmos são considerados classificadores binários, realizando a separação dos dados em um espaço vetorial que possui dimensão *N*, utilizando uma reta ou hiperplanos que possuem tamanho $N - 1$. No entanto, por meio de intersecções entre os classificadores binários, problemas com fronteiras e por meio de expressões núcleo específicas, é possível realizar a aplicação dos algoritmos baseados em SVM's para problemas em que seja necessária a classificação multiclasse (FONTANA, 2020).

Para exemplificar esse algoritmo, podemos utilizar o exemplo da Figura 8 onde é possível perceber que a a linha *b* possui a mesma distância entre os dados mais próximos de cada uma das classes. Além disso, podemos ver que as linhas *a* e *c*, que por si só são paralelas a *b*, representam as linhas onde são encontrados os dados mais próximos de cada classe, sendo a linha *a* mais próxima dos dados com classe azul e a linha *c* mais próxima dos dados com classe laranja. Também, podemos ver que há uma *margem de separação*, a qual mede a distância entre as linhas e, por si só, a distância entre os dados mais próximos de cada classe (FONTANA, 2020).

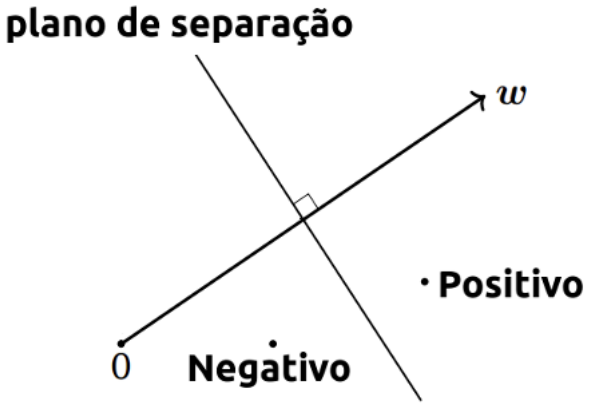
Figura 8 – Representação do plano de separação usando *Support Vector Machine*



Fonte: Fontana (2020).

De uma maneira geral, o plano de separação realiza a divisão dos dados colocando-os em espaços que possuam somente pontos que tem a mesma classe, onde, se tratando de classificações binárias, esses espaços normalmente são chamados de regiões *positivas* e *negativas*. Na Figura 9 é possível verificar graficamente essa teoria dos pontos em partes negativas e positivas, utilizando um vetor **w**, no sentido normal ao plano (FONTANA, 2020).

Figura 9 – Projeção do vetor normal ao plano de separação



Fonte: Fontana (2020).

Uma maneira de definir se um elemento está dentro do plano negativo ou positivo é verificar qual o produto escalar, definido pela fórmula:

$$w \cdot x = \| w \| \| x \| cos(\theta) \tag{2.14}$$

entre o vetor do dado com o vetor **w**, onde θ representa o ângulo que é formado entre os vetores. Essa função retorna um escalar, cujo é o produto da norma do vetor **w** com a projeção do vetor **x** sobre **w**, onde caso essa distância escalar seja grande, significa que o ponto está na parte positiva e, caso contrário, estará na parte negativa. Dessa forma, podemos definir uma função que determine em qual região está o ponto:

$$w \cdot x \geq C \tag{2.15}$$

onde *C* representa uma constante ainda não conhecida. Nesse caso, está validando se o resultado do produto escalar é maior que a constante, ou seja, caso verdadeiro, estará na parte positiva. Para o contrário, ou seja, o ponto ficar na parte negativa, a função seria o inverso (FONTANA, 2020).

2.3.5 *Clustering*

Clustering possui diversas explicações e é julgado como o principal e mais importante método dentro do aprendizado não-supervisionado. Inclusive, os métodos de análise *Clustering* ou Agrupamento possui o objetivo de dividir os dados em grupos, de maneira que, cada grupo seja único e diferentes entre si, porém com os dados que estão contidos nesse grupo precisam ter características semelhantes (FILHO, 2017; IZBICKI; SANTOS, 2020).

Oposto aos outros métodos, que realizam a análise dos dados já com classes definidas, os algoritmos de *Clustering* ou Agrupamento, realizam a análise dos dados que ainda não possuem

classificação, ou seja, a mesma é desconhecida. Esse método pode ser utilizado para gerar os rótulos dos dados de treinamento, onde as classes ainda não estão definidas (DOMINGUES, 2003; JIAWEI; MICHELINE, 2006).

Segundo Jiawei e Micheline (2006), "os objetos são agrupados sob o princípio da maximização da similaridade intraclasse e minimização da similaridade interclasse". Sendo assim, as junções entre os dados e elementos são realizadas com base no quão parecidos são os objetos que estão dentro dos agrupamentos. Dentro destes, cada agrupamento pode ser uma classificação onde determinados padrões podem ser extraídos (DOMINGUES, 2003).

Os algoritmos de agrupamento foram, segundo Xu e Tian (2015), divididos em 2: nos algoritmos tradicionais e nos algoritmos modernos, os quais foram subdivididos em categorias. Os algoritmos tradicionais foram fracionados em 9 categorias, que são: baseado em partições; baseado em hierarquia; baseado em lógica fuzzy; baseado na distribuição dos dados; baseado em densidade; baseado em teoria dos grafos; baseado em grid; baseado em teoria dos fractais; baseado em modelo. Enquanto isso, os algoritmos modernos foram segmentados em 10 categorias: baseado em *Kernel*; baseado em um conjunto de técnicas (*ensemble*); baseado em inteligência de enxames (*Swarm Intelligence*); baseado em mecânica quântica; baseado em teoria espectral de grafos; baseado em propagação por afinidade; baseado em densidade e distância; Algoritmos para dados espaciais; Algoritmos para *stream* de dados e finalmente algoritmos para dados de grande escala (FILHO, 2017).

Já Hennig *et al.* (2015) não classificam em tantas subdivisões. Os algoritmos foram separados em somente quatro grandes abordagens: a primeira que engloba métodos de otimização, cujos procuram uma função que mostre o quanto os dados se juntam; a segunda que contempla os métodos baseados em dissimilaridade, que basicamente defendem o tema que dados com carecterísticas parecidas devem estar no mesmo agrupamento, e vice-versa; a terceira que abrange os métodos baseados em probabilidade e, os quais seriam modelos de mistura e particionamento e; a última que abrange os métodos não paramétricos, que ao contrário de definir modelos para os agrupamentos, identificam-nos como "ilhas de alta densidade".

É possível detalhar o passo a passo para utilização dos métodos de *clustering*, segundo Milligan (1996), sendo eles:

1. definir quais são os dados que serão agrupados;
2. definir quais serão os valores que serão agrupados;
3. padronizar os valores;
4. definir qual será a medida de similaridade;
5. definir qual será o método de *clustering*;

6. definir o número de agrupamentos a serem realizados, sempre lembrando que existem métodos cujo número é um parâmetro inicial;

7. testar, interpretar e validar os agrupamentos.

Basicamente, Izbicki e Santos (2020) definem através de fórmulas o método de *clustering*, com o objetivo de ter partições $C_1, ..., C_k$ dos elementos dos conjuntos de dados 1,...,*n*. Convertendo, fica dessa forma:

$$C_1 \cup C_2 \cup ... \cup C_K = \{1, 2, ..., n\} \tag{2.16}$$

e

$$C_i \cap C_j = \varnothing \forall i \neq j. \tag{2.17}$$

Além disso, um fator muito importante para os métodos de agrupamento é a realização da medição da similaridade ou dissimilaridade entre os elementos com covariáveis $\mathbf{x_i}$ e $\mathbf{x_j}$. Visto isso, Izbicki e Santos (2020) citam algumas medidas possíveis para esses casos:

- **distância Euclidiana:** $d^2(\mathbf{x_i}, \mathbf{x_j}) = \sum_{k=1}^{d}(x_{i,k} - x_{j,k})^2$;
- **distância de Mahalanobis:** $d(\mathbf{x_i}, \mathbf{x_j}) = \sum_{k=1}^{d}|x_{i,k} - x_{j,k}|$;
- **distância cosseno:** $d(\mathbf{x_i}, \mathbf{x_j}) = 1 - \frac{x_i \cdot x_j}{||x_i||||x_j||}$;
- **distância de Jaccard:** $d(\mathbf{x_i}, \mathbf{x_j}) = 1 - \frac{\sum_{k=1}^{d} x_{i,k}x_{j,k}}{\sum_{k=1}^{d} x_{i,k} + \sum_{k=1}^{d} x_{j,k} - \sum_{k=1}^{d} x_{i,k}x_{j,k}}$, em que **x** assume valores de 0 e 1.

Dentre as fórmulas apresentadas, cada uma possui uma melhor aplicação dependendo do tipo de contexto (IZBICKI; SANTOS, 2020).

2.4 CONSIDERAÇÕES SOBRE O CAPÍTULO

Com base nas pesquisas e estudo realizado nesse capítulo, foi possível concluir que, dentro de todo o contexto de AM e suas subdivisões há uma variedade de técnicas e algoritmos os quais tornam essa área de pesquisa bastante ampla.

Inicialmente começa-se com as divisões entre os dois tipos de aprendizado, supervisionado e não supervisionado, cujo primeiro é fortemente utilizado para dados que já estão previamente rotulados, enquanto o segundo é aplicado em dados que não possuem classificação e busca-se encontrar padrões para realizar a mesma.

Além disso, o aprendizado supervisionado conta ainda com mais subdivisões, possuindo algoritmos e métodos de classificação e de regressão. De classificação são aqueles utilizados para rotular e prever futuros dados a partir de dados pré estabelecidos. Há a classificação binária

e classificação multiclasse. Basicamente ambas são autoexplicativas, porém resumidamente a classificação binária limita a rotulação dos dados em somente 2 classes, enquanto a multiclasse permite uma gama maior de classificações, sendo mais liberal nesse quesito. Ao contrário disso, há os algoritmos de regressão, que são mais utilizados para a verificação de quais atributos e variáveis dentro do conjunto de dados que são capazes de estar mais relacionadas com os vetores, ou seja, com os dados em si.

Aprofundando ainda mais dentro do mundo de AM e suas camadas foi possível chegar finalmente nos exemplos de algoritmos mais utilizados, dentre eles o *Random Forest* (RF), as *Artificial Neural Networks* (ANN), os *K-Nearest Neighbor* (kNN), *Support Vector Machine* (SVM), *Clustering* e diversos outros também existentes mas que não tiveram um estudo aprofundado sobre. Cada um desses algoritmos possui sua maneira de aplicação dentro dos conjuntos de dados, onde cada um se aplica a diferentes tipos de problemas. SVM e kNN são métodos que podem ser utilizados tanto para problemas de classificação quanto de regressão, onde é realizada uma análise prévia de cada necessidade para que seja aplicado o melhor algoritmo. Já RF, ANN e afins possuem uma grande vantagem em relação aos outros algoritmos quando trata-se de problemas de classificação, sendo muitas das vezes escolhidos para aplicação em diversos tipos de estudos. Por fim, o *Clustering*, que é o principal algoritmo de aprendizado não supervisionado, utilizado principalmente em conjuntos de dados que não possui pré classificações e é necessário estabelecê-las, aplicando o método para estabelecer padrões entre os dados e chegar a um rótulo final.

Afinal, vale-se destacar que a utilização e combinação dos algoritmos pode ser de extrema relevância, visto que conforme já supracitado, cada um possui uma melhor maneira de aplicação e um algoritmo pode completar e auxiliar o outro dentro de problemas para que se chegue numa melhor precisão e acerto dos resultados.

3 REVISÃO SISTEMÁTICA DA LITERATURA

Para construção dessa monografia, foram feitas pesquisas dentro de alguns dos principais motores de busca de estudos científicos, sendo eles *IEEE Explore*, *SciELO Brasil* e *Google Scholar*. A partir das buscas, foram selecionados 12 artigos através das palavras-chave aprendizado de máquina, *Random Forest* (RF) e processamento de dados e dentro do período de 20 Desses, apenas avaliando o resumo dos mesmos, foram reduzidos para 8 artigos que estavam mais alinhados com o problema desta pesquisa. Além disso, foi feito o descarte de alguns artigos pois não apresentavam uma análise própria dos resultados embasados no algoritmo RF ou pelo fato de estarem explicando outros algoritmos.

Após a realização do descarte, restaram 3 artigos que estavam de acordo com o tema desta monografia e que serão detalhados na Seção 3.1, Seção 3.2 e Seção 3.3.

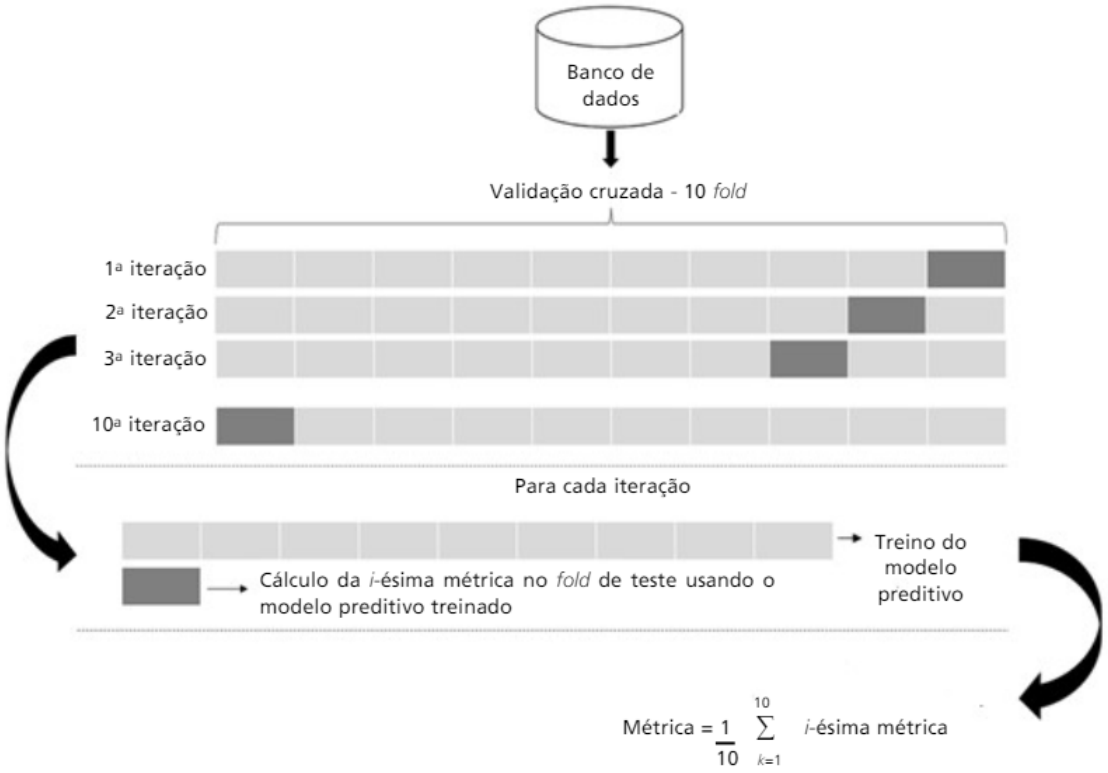
3.1 PREDIÇÃO DE MORTALIDADE E CARACTERÍSTICAS CLÍNICAS EM IDOSOS COM COVID-19 USANDO *RANDOM FOREST*

O primeiro artigo utilizado como referência para esta monografia foi o escrito por Lima *et al.* (2021), onde foi realizado um estudo com a finalidade de "treinar um classificador do tipo *Random Forest* (RF) para estimar o risco de óbito em idosos acima de 60 anos diagnosticados com *COVID-19*".

O trabalho começou com toda a contextualização sobre a pandemia do COVID-19 (doença causada pelo coronavírus e diagnosticada pela primeira vez em 2019) e foi feita uma explicação sobre as vantagens de se identificar os fatores de risco que poderiam ser agravantes para levar pessoas a óbito devido a doença, como foi feito o levantamento dos dados e a explicação sobre o algoritmo selecionado, que nesse caso foi o *Random Forest* (RF).

Em conjunto com o algoritmo RF, foi utilizado o método de validação cruzada, denominado *K-fold cross validation*, que realiza a análise de desempenho do algoritmo. Esse método consiste em dividir randomicamente os dados em K partes (*folds*) exclusivas e com mesmo tamanho. Para esse trabalho, conforme Figura 10, o valor utilizado para K foi 10.

Figura 10 – Fluxograma da metodologia de Validação Cruzada utilizando 10 *folds*



Fonte: Lima *et al.* (2021).

Além disso também foi realizado o cálculo de duas taxas para desempenho métrico do algoritmo classificador. A primeira delas, a taxa de erro (err), que compara a classe atribuída pelo classificador (h) com a sua classe verdadeira, pode ser obtida de duas maneiras. A primeira maneira é dada pela fórmula abaixo:

$$err(h) = \frac{1}{\pi} - \sum_{i=1}^{n} ||h(x_i) \neq f(x_i)|. \tag{3.1}$$

Nesse caso, se as duas classes forem iguais, ou seja, $h(x_i) = f(x_i)$, então $|[h(x_i) \neq f(x_i)]| = 1$. Se for o oposto, o valor para o resultado é 0. Além disso, a taxa de erro (err) também pode dar-se através de uma matriz de confusão, cuja representa uma matriz a qual a dimensão é o numero de classes que possui o conjunto de dados e a fórmula da-se conforme abaixo:

$$err(h) = \frac{FN + FP}{VP + FN + FP + VN}. \tag{3.2}$$

A segunda taxa é a de acerto (acc), que nada mais é que o complemento da taxa de erro e seu valor é definido através das seguintes fórmulas:

$$acc(h) = 1 - err(h). \tag{3.3}$$

A outra fórmula para se calcular o valor da taxa de acerto (acc) também é dada da mesma forma que da taxa de erro, através de uma matriz de confusão, e é definida pela fórmula:

$$acc(h) = \frac{VP + VN}{VP + FN + FP + VN}. \tag{3.4}$$

Em complemento a isso, foram realizadas outros cálculos de medida de desempenho baseados nas matrizes de confusão, que são o calculo da sensibilidade, que é representada pela função abaixo:

$$Sensibilidade = \frac{VP}{(VP + FN)} \tag{3.5}$$

e pelo cálculo da especificidade, dado pela fórmula:

$$Especificidade = \frac{VN}{(FP + VN)}. \tag{3.6}$$

Ambos os resultados dessas equações foram utilizados para geração da *Area Under the Curve* (AUC), que dá-se através da geração de um gráfico de sensibilidade *versus* (1 - especificidade), que é conhecido como curva *Receiver Operating Characteristics* (ROC).

Adicionalmente, todos os cálculos e aplicações foram realizados com base em dados de mais de 11.000 pacientes que tinham idade acima de 60 anos, alocados todos em somente um conjunto de dados, captados no período de fevereiro a junho de 2020, em Pernambuco, Brasil. Depois do trabalho realizado previamente nos dados, restaram aproximadamente 7500 idosos.

Por fim, foi identificado que o classificador RF teve um acerto de 78,39% dos pacientes do conjunto de dados. O desempenho do algoritmo foi possível se dar através da matriz de confusão e métricas, conforme é possível confirmar na Tabela 1.

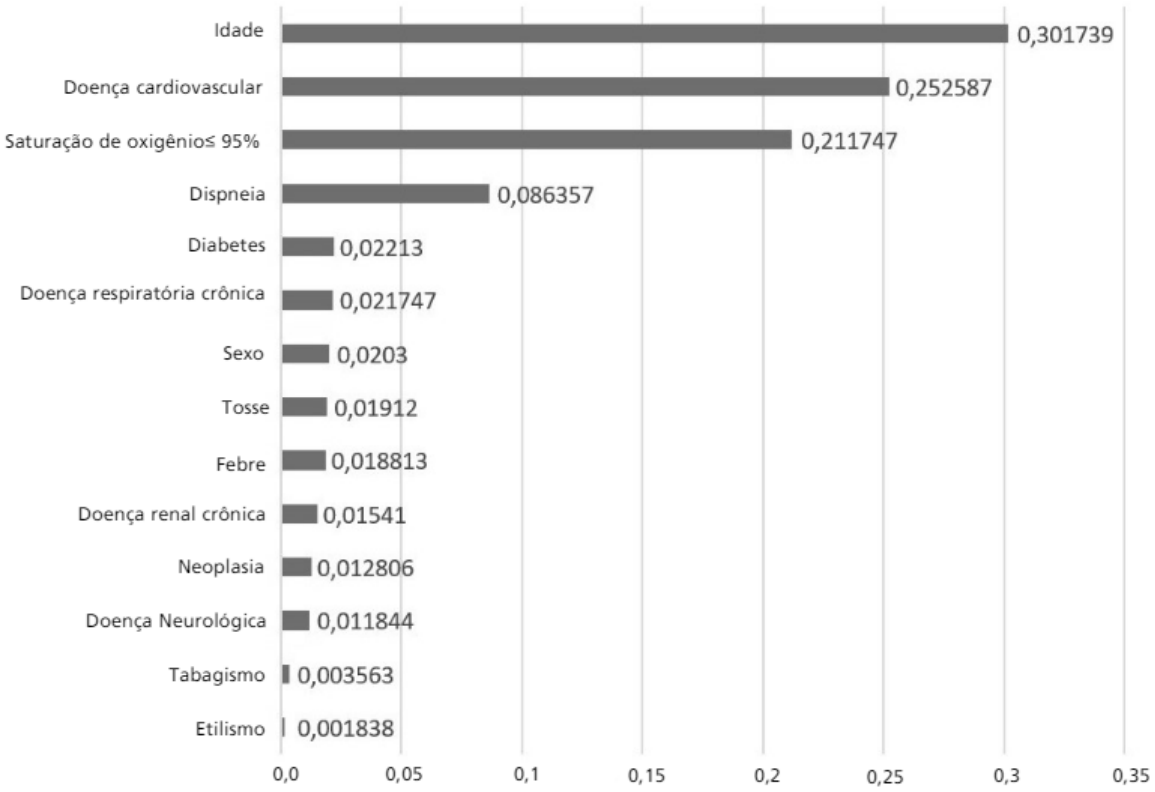
Tabela 1 – Métricas para avaliação do desempenho do classificador *Random Forest*

Taxa de Verdadeiro positivo	**Taxa de Falso positivo**	**Precisão**	**Sesibilidade**	**AUC ROC**	**Desfecho**
0,848	0,306	0,794	0,848	0,839	Recuperado
0,694	0,152	0,767	0,694	0,839	Óbito
0,784	0,241	0,783	0,784	0,839	Média Ponderada

Fonte: Lima *et al.* (2021).

Também foi gerada uma imagem, conforme ilustra a Figura 11. Nela é mostrada a importância de cada um dos atributos, em ordem decrescente de importância. Pode-se concluir que a idade, com 0,302, a presença de doença cardiovascular, com 0,252 e a saturação de oxigênio menor ou igual a 95% (0,212) foram os três principais fatores importantes para levar o paciente idoso a óbito pela doença.

Figura 11 – Classificação de importância dos atributos



Fonte: Lima *et al.* (2021).

Demais comparações históricas foram feitas em relação aos dados conseguidos através dessa pesquisa, porém não se referem a utilização do algoritmo de RF.

3.2 USO DE APRENDIZADO DE MÁQUINA EM DADOS DE SENSORIAMENTO REMOTO PARA MAPEAR FLORESTAS URBANAS

O segundo trabalho, realizado por Cano e Junior (2021), trouxe a pesquisa sobre a utilização e aplicação do aprendizado de máquina em dados de sensoriamento remoto com a finalidade de realizar o mapeamento de florestas urbanas. Segundo os autores, florestas urbanas são capazes de fornecer diversos benefícios para as cidades, dentre eles: redução das temperaturas; melhor qualidade do ar; saúde e lazer dos moradores e; proteção de bacias hidrográficas. Ademais, Campo Grande - MT foi a cidade selecionada, visto que a mesma já é reconhecida como uma cidade comprometida com a preservação das florestas e o desenvolvimento sustentável.

Para que o estudo tenha sido realizado, Cano e Junior (2021) definiram um total de quatro passos a serem seguidos, os quais são: escolher dados para treinamento; divisão da imagem do satélite; classificação supervisionada e; por fim a avaliação dos resultados da classificação.

Para o primeiro passo, foram selecionadas um total de 415 amostras para o treinamento

de classificação, divididas entre 218 árvores individuais, blocos de árvores e florestas de diferentes tons, enquanto as 197 que sobraram incluíam todo o restante que não é considerado árvore. De uma maneira geral a classificação foi dada em duas classes: as árvores e outras. A partir disso, foi realizado o segundo passo: a segmentação da imagem de satélite, onde foi realizada a junção dos pixeis adjacentes com características espectrais semelhantes em objetos das imagens. Após isso foi realizada a classificação supervisionada que, para isso, foi escolhido o algoritmo de RF, visto que esse método já havia apresentado resultados positivos em diversos testes e estudos com vegetação.

Com a finalidade de conferir a precisão, foram criados 500 pontos colocados de maneira aleatória dentro do plano estudado, diferentes dos utilizados nos dados de treinamento, permitindo que o algoritmo realizasse a classificação e fosse feita a validação manual dos resultados. Após isso, foi gerada uma matriz de confusão com as informações dos dados classificados de maneira correta e incorreta, comparando sempre com os dados reais do estudo, o que permitiu que fossem geradas métricas como sensibilidade (Equação 3.7):

$$Sensibilidade = \frac{VP}{(VP + FN)}; \tag{3.7}$$

especificidade (Equação 3.8):

$$Especificidade = \frac{VN}{(FP + VN)}; \tag{3.8}$$

precisão (Equação 3.9):

$$Precisao = \frac{VP}{(VP + FP)}; \tag{3.9}$$

acurácia (Equação 3.10):

$$Acuracia = \frac{(VP + VN)}{(P + N)} \tag{3.10}$$

e; pontuação F1 (Equação 3.11):

$$F1 = \frac{2 \cdot VP}{[(2 \cdot VP) + FP + FN]}. \tag{3.11}$$

Os resultados obtidos da matriz de confusão podem ser vistos na Tabela 2 e os resultados das métricas podem ser vistas na Tabela 3.

Tabela 2 – Matriz de Confusão

		Valor Verdadeiro	
		Positivo	**Negativo**
Valor Previsto	**Positivo**	83 (VP)	9 (FP)
	Negativo	20 (FN)	388 (VN)

Fonte: Adaptado de Cano e Junior (2021).

Tabela 3 – Métricas avaliadas e reulstados obtidos

MÉTRICAS	VALOR OBTIDO %
Sensibilidade	80.58
Especificidade	97.73
Precisão	90.22
Acurácia	94.20
Pontuação F1	85.13

Fonte: Adaptado de Cano e Junior (2021).

Ao realizar a avaliação dos resultados, perceberam que, as áreas em que o algoritmo atribuiu como não arbóreas corretamente acabou por alcançar um percentual quase de 100,00%, chegando ao número de 97,73%. Ademais, a precisão geral ficou na casa acima dos 94%, enquanto que a medida de F1 ultrapassou os 85%, assegurando que, a aplicação do algoritmo de RF para realizar esse tipo de trabalho possui um bom desempenho no mapeamento e pode sim ser utilizada (CANO; JUNIOR, 2021).

3.3 PREDIÇÃO DE RISCO DE EVASÃO NO ENSINO SUPERIOR PÚBLICO BRASILEIRO USANDO APRENDIZADO DE MÁQUINA

O terceiro trabalho utilizado como base para desenvolvimento desta monografia foi o escrito por Teodoro e Kappel (2020), cujo objetivo foi aplicar algoritmos de aprendizado de máquina em uma base de dados de estudantes visando a identificação de principais padrões de características dos alunos que possuem maior chance de desistir do ensino público superior, descobrindo também os principais atributos nestes padrões de estudantes.

Começa com uma introdução sobre todo o cenário de evasão dos alunos de ensino superior nas instituições públicas brasileiras, juntamente com a explicação sobre aprendizado de máquina e os algoritmos que serão aplicados no trabalho, visando a busca do melhor que se aplicasse nessa situação. Ademais, seguiu-se apresentando os trabalhos relacionados com o tema. Em seguida quais foram os métodos e processos adotados para obtenção e preparação dos dados, explicando etapas do pré-processamento e as técnicas de AM utilizadas. Por fim, apresentação dos resultados obtidos, assim como as conclusões.

Basicamente a metodologia e preparação dos dados foi realizada tendo sido pega a base de dados disponibilizada pelo Instituto Nacional de Estudos e Pesquisas Educacionais (INEP) e foi trabalhado nas relações entre as tabelas para que fosse criada a estrutura de dados propriamente dita. Nesse caso, foram separados 75% dos dados para utilizar na base de treinamento e o restante ficou para a base de testes.

Feito isso, aplicados os métodos de aprendizado de máquina e definidos os melhores parâmetros para cada um, obtivemos o resultado conforme é possível verificar na Tabela 4, a

qual mostra qual foi a técnica, a parametrização utilizada e a matriz de confusão gerada. Em conjunto com essa matriz de confusão, também foram calculadas as seguintes métricas:

- Acurácia: $Acuracia = \frac{VP+VN}{VP+VN+FP+FN}$;
- Precisão: $Precisao = \frac{VP}{VP+FP}$;
- Recall: $Recall = \frac{VP}{VP+FN}$;
- F1-Score: $F1 = 2 \cdot \frac{Precisao \cdot Recall}{Precisao+Recall}$;
- Curva ROC e AUC:
  - Taxa de Verdadeiros Positivos (TVP): $TVP = \frac{VP}{VP+FN}$;
  - Taxa de Falsos Positivos (TFP): $TFP = \frac{FP}{FP+VN}$.

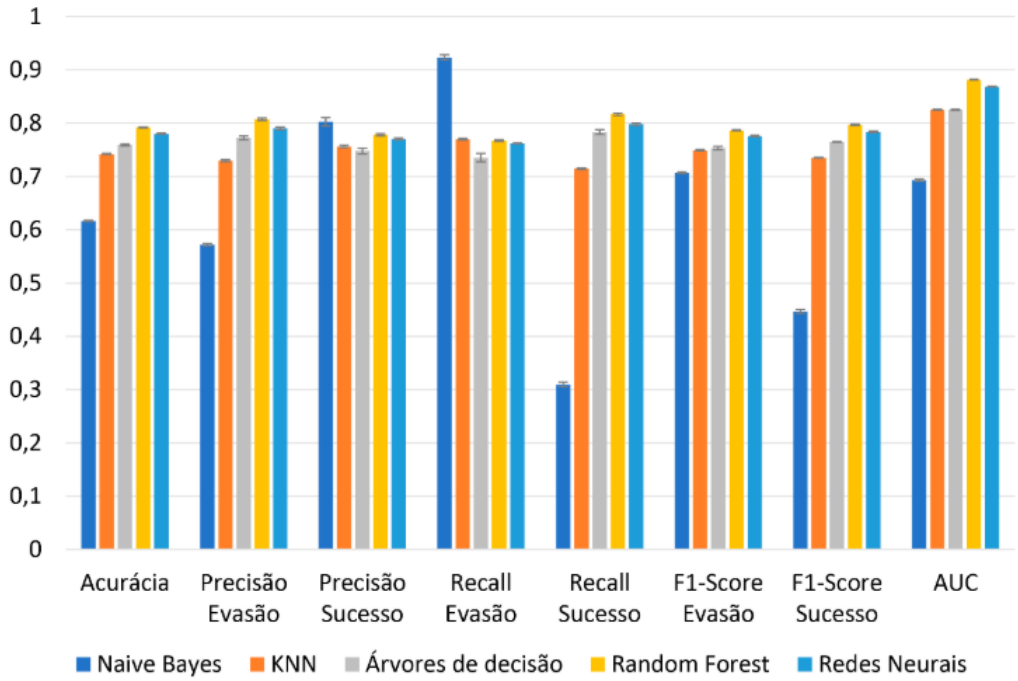
Tabela 4 – Parâmetros e Matriz de Confusão das técnicas aplicadas

Tecnica	**Configuração**	**Matriz de Confusão**
Naive Bayes	-	VP = 34806,8 ± 224,1 FN = 2867,8 ± 178,0 FP = 26017,2 ± 200,5 VN = 11657,4 ± 177,1
kNN	K = 10	VP = 29005,6 ± 112,0 FN = 8669,0 ± 83,2 FP = 10746,2 ± 77,0 VN = 26928,4 ± 102,1
Árvores de Decisão	Profundidade máxima = 18	VP = 27703,0 ± 258,5 FN = 9971,6± 321,4 FP = 8165,0 ± 203,1 VN = 29509,6 ± 130,8
Random Forest	Profundidade máxima = 22 Número de características usadas = 70 Número de árvores = 70	VP = 28910,0 ± 83,4 FN = 8764,6 ± 89,2 FP = 6898,4 ± 103,2 VN = 30776,2 ± 95,6
Redes Neurais	Neurônios na camada de entrada: 100 Neurônios na camada escondida: 40 Neurônios na camada de saída: 20	VP = 28713,4 ± 121,8 FN = 8961,2 ± 47,1 FP = 7606,0 ± 88,4 VN = 30068,6 ± 128,2

Fonte: Adaptado de Teodoro e Kappel (2020).

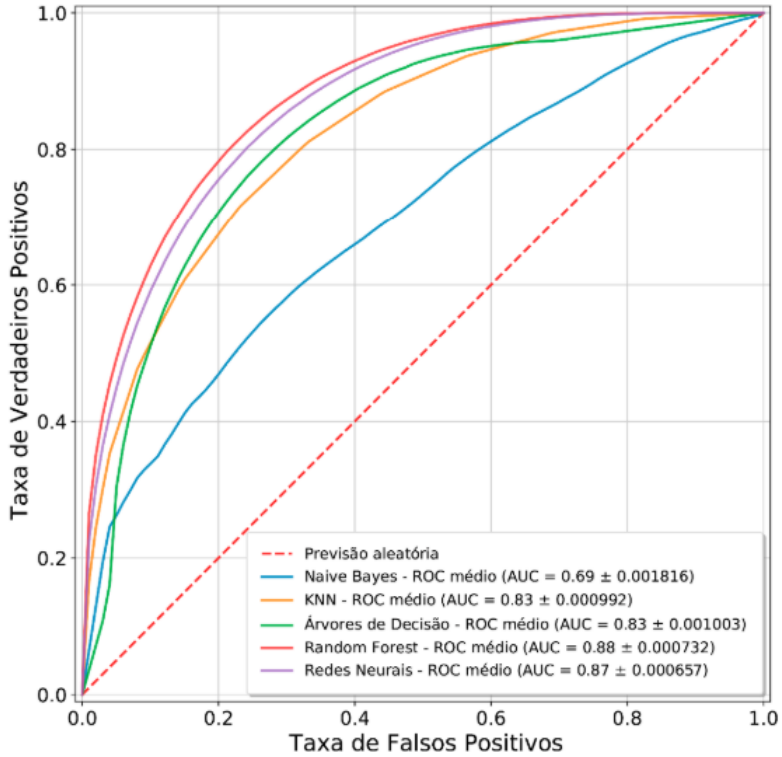
Realizadas todas as aplicações dos algoritmos, feitos todos os cálculos e formações dos resultados, começou-se a parte de análise dos dados, onde os resultados da validação cruzada podem ser vistos na Figura 12 e as curvas ROC podem ser vistas na Figura 13.

Figura 12 – Resultados obtidos pela validação cruzada de cada técnica



Fonte: Teodoro e Kappel (2020).

Figura 13 – Curvas ROC médias para cada classificador

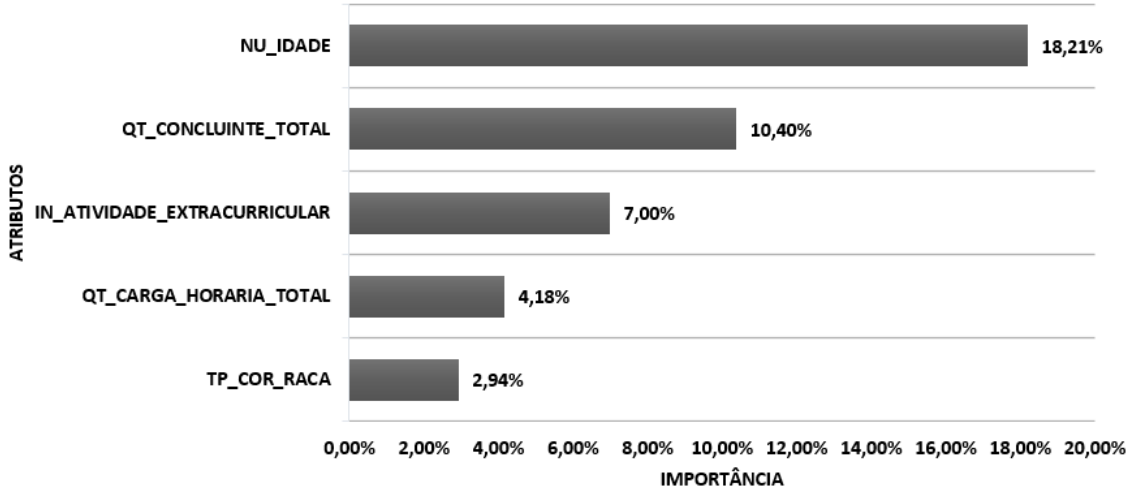


Fonte: Teodoro e Kappel (2020).

De modo geral, analisando as duas figuras, Figura 10 e Figura 13, conclui-se que o método de RF foi o que teve o melhor desempenho de maneira geral entre todos os algoritmos aplicados, visto que, principalmente pela sua curva ROC, que possui o maior valor entre todos. Adicionalmente, mesmo que tenha tido resultados mais baixos em alguma outra métrica de avaliação, mostrou-se mais confiável devido aos seus resultados nas demais métricas, comprovando eficácia alta tanto nas previsões de evasão quanto de sucesso.

Em paralelo a isso, o método de RF ainda consegue fornecer dados que auxiliam no momento de avaliação de quais atributos são os mais impactantes no momento de realizar as classificações e previsões. A Figura 14 mostra isso e a Tabela 5 dá a descrição desses atributos de forma detalhada de cinco dos trinta atributos. A título de curiosidade, a soma de todos os 30 atributos correspondem a mais de 78% de importância, um AUC de 0,88 e uma acurácia próxima de 0,80, o que quer dizer que quando o modelo treinado for classificar um aluno novo, ou seja, um novo dado, ele terá 80% de chance de acertar se é um aluno provável de evasão ou sucesso.

Figura 14 – Lista ordenada dos valores médios das importâncias de cinco atributos considerados mais determinantes para a previsão de evasão pelo classificador *Random Forest*



Fonte: Adaptado de Teodoro e Kappel (2020).

Tabela 5 – Descrição dos cinco atributos da Figura 14

Identificador	Descrição
NU_IDADE	Idade que o aluno completou no ano de referência do Censo.
QT_CONCLUINTE_TOTAL	Número total de concluintes no curso.
IN_ATIVIDADE_EXTRACURRICULAR	Participação em algum tipo de atividade extracurricular. (0 - não, 1 - sim)
QT_CARGA_HORARIA_TOTAL	Total da carga horária dos componentes curriculares do curso.
TP_COR_RACA	Código que identifica a cor/raça do aluno.

Fonte: Adaptado de Teodoro e Kappel (2020).

Concluiu-se, através da análise da Figura 14 que, mesmo com atributos pessoais sendo impactantes, os atributos relacionados a instituição que o estudante está matriculado também possuem grande interferência na evasão e sucesso dos alunos. Ademais, o decorrer da pesquisa e trabalho foram feitas mais análises e inferências referentes aos demais atributos que foram elencados pelo método RF.

Por fim, a conclusão do trabalho deu-se com a explicação dos pesquisadores em relação a contribuição do trabalho, apresentando quais seriam as expectativas criadas, onde basicamente espera-se que os resultados "possibilitem o desenvolvimento de estratégias de redução de evasão focadas no suporte a estudantes que se encontrem nos padrões característicos identificados".

3.4 CONSIDERAÇÕES SOBRE O CAPÍTULO

Foi possível perceber e identificar com os trabalhos selecionados que ambos apresentam abordagens diferentes sobre a utilização de aprendizado de máquina, sendo elas em áreas diferentes de atuação mas que é identificável que a aplicação funciona da mesma forma. A utilização do método *Random Forest* foi padrão para todos os trabalhos e, no que utilizou também outros algoritmos, o *Random Forest* mostrou-se mais eficiente, visto que seus resultados nas variáveis de medição de desempenho se mostraram melhores.

Dessa forma, foi perceptível que o RF se mostra aplicável nos mais diversos cenários e áreas do conhecimento, sendo ele um dos melhores algoritmos quando trata-se do tema de aprendizado de máquina. Ademais, a partir desse embasamento teórico e dos métodos e experimentos realizados através nesses estudos e de outros que foram utilizados para investigação da aplicação do algoritmo nesta monografia, fica mais claro e evidente como deve ser desenvolvida a proposta de solução, onde poderá ser melhor explorado e aproveitado o algoritmo.

4 PROPOSTA DE SOLUÇÃO

Como esse trabalho visa realizar um estudo sobre Aprendizado de Máquina (AM), tendo ênfase no algoritmo *Random Forest* (RF), a proposta de solução baseia-se na aplicação do algoritmo em três *datasets* de diferentes áreas de aplicação e para problemas de classificação e regressão.

A aplicação do algoritmo foi desenvolvida na linguagem R e foi utilizada a plataforma de desenvolvimento do *RStudio*. Por conseguinte, foi utilizada a biblioteca do RF que a linguagem R possui, a qual será detalhada mais profundamente na Seção 4.3. A linguagem e a plataforma serão brevemente explicadas na Seção 4.1 e Seção 4.2, respectivamente, assim como as bibliotecas complementares que foram utilizadas (Seção 4.4).

A solução desenvolvida neste trabalho, seguiu as seguintes etapas:

- Análise explanatória dos dados, explicitada na Seção 4.5;
- Preparação dos datasets, pré-processamento e transformação dos dados, explicitada na Seção 4.6;
- Serão realizados testes aplicando o algoritmo de RF, com alterações diversas de valores dos parâmetros, explicitada na Seção 4.7;
- Análise dos resultados obtidos, baseando-se nos critérios de desempenho adotados conforme abaixo e explicitado no Capítulo 5:
  - Para classificação: Acurácia, Sensibilidade, Especificidade, Precisão, onde para todas essas medidas, quanto mais próximo do 100%, melhor.
  - Para regressão:
    - Erro Médio Absoluto (MAE): onde quanto mais próximo de 0, melhor.
    - Erro Médio Quadrático (MSE): onde quanto mais próximo de 0, melhor.
    - Raiz do Erro Médio Quadrático (RMSE): onde quanto mais próximo de 0, melhor.
    - Coeficiente de Determinação (R^2): onde quanto mais próximo de 1, melhor.

Vale ressaltar que, a máquina cujos algoritmos foram executados possuía a seguinte configuração de hardware:

- Processador Intel Core i7 - 3ª Geração;
- SSD de 256GB;

- 16GB de memória ram;
- Placa de vídeo integrada ao processador.

4.1 LINGUAGEM R

A linguagem R foi criada no inicio da década de 90, pelos estatísticos Ross Ihaka (1954) e Robert Gentleman (1959) dentro da própria Universidade de Auckland, na Nova Zelândia e seu nome, "R", provém das iniciais dos criadores ("R & R"). Ela foi construída totalmente a partir da Linguagem S, criada por John Chambers (1949), ex-presidente executivo e CEO da *Cisco Systems* (R, 2017).

Ela é uma linguagem de programação orientada a objetos, gratuita e *open-source* que, atualmente, há diversos materiais de suporte dentro da *web* que podem auxiliar no desenvolvimento do trabalho, além de poder ser utilizado nos sistemas operacionais mais conhecidos como Windows, Linux e Mac OS (FREIRE, 2021).

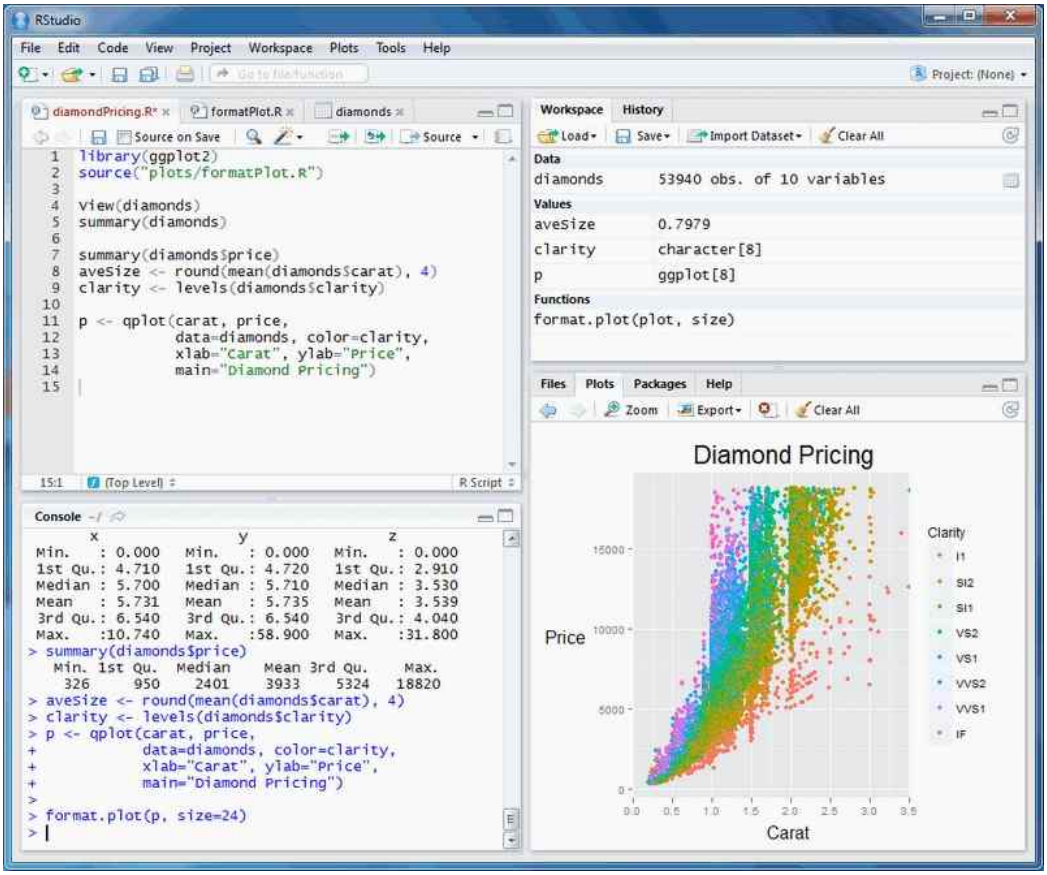
Ademais, possui os *data frames*, uma ferramenta poderosa para manipulação de dados em memória, cuja permite realizar a organização de múltiplos tipos de dados em um único elemento do conjunto de dados. Também, por ser um software livre, há diversos pacotes na comunidade que auxiliam na realização das análises e visualização dos dados, assim como no desenvolvimento de modelos para o aprendizado de máquina (FREIRE, 2021).

4.2 *RSTUDIO*

Segundo seu CEO, Joseph J. Allaire (1969), *RStudio* é um Ambiente de Desenvolvimento Integrado (IDE) próprio para se utilizar com a linguagem de programação R. ele é um ambiente de código aberto, que realiza agrupamento de diversos componentes da linguagem, como: console, edição de código, plotagem gráfica, históricos, ajudas, etc.

Ademais, ele foi totalmente projetado para ter um ambiente interativo e que facilite a utilização tanto para novos usuários quanto para os mais experientes em programação, pois também fornece ferramentas de alta produtividade. É possível ver a interface na Figura 15.

Figura 15 – Interface *RStudio* para Windows



Fonte: Allaire (2012)

Assim como a própria linguagem, o *RStudio* também é executável nos principais sistemas operacionais, que incluem o Windows, Linux e Mac OS. Outro destaque dessa IDE é que ela pode ser configurada e instalada dentro de um servidor, parametrizando-a para permitir acesso *web* as sessões R que são executadas remotamente (ALLAIRE, 2012).

4.3 BIBLIOTECA DO *RANDOM FOREST*

A biblioteca do *Random Forest* realiza a implementação do algoritmo de floresta aleatória de Breiman e é utilizado tanto para problemas de classificação quanto para problemas de regressão. Ademais, pode ser utilizado para problemas de aprendizado de máquina não supervisionado quando se faz necessário avaliar a distância entre dados.

Possui como principal função o código Algoritmo 1, que será o mais utilizado no desenvolvimento do Trabalho de Conclusão de Curso (TCC) II.

Algoritmo 1 – Exemplo função principal *randomForest*

```
1 randomForest(x, y = NULL, xtest = NULL, ytest = NULL, ntree = 500,
2 mtry = if (!is.null(y) && !is.factor(y))
3 max(floor(ncol(x)/3), 1) else floor(sqrt(ncol(x))),
```

```
4                 weights = NULL,
5                 replace = TRUE, classwt = NULL, cutoff, strata,
6                 sampsize = if (replace) nrow(x) else ceiling(.632*nrow(x)),
7                 nodesize = if (!is.null(y) && !is.factor(y)) 5 else 1,
8                 maxnodes = NULL,
9                 importance = FALSE, localImp = FALSE, nPerm = 1,
10                proximity, oob.prox = proximity,
11                norm.votes = TRUE, do.trace = FALSE,
12                keep.forest = !is.null(y) && is.null(xtest), corr.bias = FALSE,
13                keep.inbag = FALSE, ...)
```

Fonte: Disponível em: <https://www.rdocumentation.org/>. Acesso em 22 jun. 2024.

Cada um dos argumentos do Algoritmo 1 está detalhado na Tabela 6.

Tabela 6 – Detalhamento dos argumentos da função *randomForest*

Argumento	**Descrição**
data	Um *dataframe* opcional contendo as variáveis do modelo. Por padrão, as variáveis são obtidas do ambiente de onde *randomForest* é chamado.
subset	Um vetor de índice indicando quais linhas devem ser usadas. Se fornecido, este argumento deve ser nomeado.
na.action	Uma função para especificar a ação a ser tomada se NA's forem encontrados. Se fornecido, este argumento deve ser nomeado.
x, formula	Um *dataframe* ou uma matriz de preditores, ou uma fórmula que descreve o modelo a ser ajustado (para o método *print*, um objeto *randomForest*).
y	Representa um vetor de resposta. Se omitido, *randomForest* será executado em modo não supervisionado.
xtest	Um *dataframe* ou matriz (como x) contendo preditores para o conjunto de teste.
ytest	Resposta para o conjunto de teste.
ntree	Número de árvores a crescer. A sugestão é que não seja um número pequeno.
mtry	Número de variáveis amostradas aleatoriamente como candidatas em cada divisão. Os valores padrão são diferentes para classificação (sqrt(p)) e regressão (p/3) - p é o número de variáveis em x.
weights	Um vetor de comprimento igual a y que são pesos positivos usados apenas em dados de amostragem para fazer crescer cada árvore.
replace	A amostragem dos casos deve ser feita com ou sem reposição?
classwt	Priores das classes. Não é necessário somar um. Ignorado para regressão.
cutoff	Um vetor de comprimento igual ao número de classes. O padrão é 1/k, onde k é o número de classes. Válido somente para classificação.
strata	Uma variável fatorial usada para amostragem estratificada.
sampsize	Tamanho(s) da amostra a ser extraída.
nodesize	Tamanho mínimo dos nós terminais. Definir esse número para um valor maior faz com que árvores menores sejam cultivadas. Os valores padrão são diferentes para classificação (1) e regressão (5).
maxnodes	Número máximo de nós terminais que as árvores da floresta podem ter. Se não for fornecido, as árvores crescerão ao máximo possível. Se definido acima do máximo possível, um aviso será emitido.
importance	A importância dos preditores deve ser avaliada?
localImp	A medida de importância caso a caso deve ser calculada? Se definido como TRUE, substituirá o argumento importance).
nPerm	Número de vezes que os dados OOB são permutados por árvore para avaliar a importância da variável. Números maiores que 1 fornecem uma estimativa um pouco mais estável, mas não muito eficaz. Somente para regressão.
proximity	A medida de proximidade entre as linhas deve ser calculada?
oob.prox	A proximidade deveria ser calculada apenas com base em dados "prontos para uso"?
norm.votes	Se *TRUE* (padrão), o resultado final dos votos será expresso como frações. Se FALSE, as contagens brutas de votos serão retornadas. Ignorado para regressão.
do.trace	Se definido como *TRUE*, fornece uma saída mais detalhada à medida que *randomForest* é executado. Se definido como algum número inteiro, a saída em execução será impressa para cada árvore *do.trace*.
keep.forest	Se definido como *FALSE*, a floresta não será retida no objeto de saída. Se *xtest* for fornecido, o padrão é FALSE.
corr.bias	Argumento experimental. Realizar correção de viés para regressão?
keep.inbag	Deve ser retornada uma matriz n por *ntree* que monitore quais amostras estão "no saco" em quais árvores?
...	Parâmetros opcionais a serem passados para a função de baixo nível *randomForest.default*.

Fonte: Adaptado de *R Documentation*. Disponível em: <https://www.rdocumentation.org/>. Acesso em 22 jun. 2024.

Dentre os parâmetros supracitados, os selecionados para

Além disso, também existem diversas outras funções dentro da biblioteca do RF no R, como: *classCenter*, *combine*, *getTree*, *grow*, *importance*, *imports85*, *margin*, *MDSplot*, *na.roughfix*, *outlier*, *partialPlot*, *plot.randomForest*, *predict.randomForest*, *rfcv*, *rfImpute*, *rfNews*, *treesize*,

tuneRF, *varImpPlot*, *varUsed*.

Destas, as que foram utilizadas foram: *randomForest*, para aplicação do algoritmo propriamente dito, *importance* e *varImpPlot*, utilizadas para extrair o valor da importância das variáveis e plotagem da mesma.

4.4 BIBLIOTECAS COMPLEMENTARES UTILIZADAS

A seguir está um breve resumo das bibliotecas complementares que foram utilizadas no desenvolvimento, juntamente com uma breve explanação sobre as funções que foram utilizadas.

4.4.1 *Caret*

Essa biblioteca *Caret*, ou pacote como comumente é falado, é a abreviação vinda do inglês *Classification and Regression Training*. Possui diversas funções para auxiliar no treinamento de modelos para problemas de classificação e regressão. Ademais, nele estão acoplados diversos outras bibliotecas da linguagem R[1].

Dentre as diversas funções que esse pacote possuí, as que foram utilizadas dentro desse trabalho foram, juntamente com uma breve descrição da utilização:

- $confusionMatriz()$: realiza o calculo das Calcula uma tabulação cruzada de classes observadas e previstas com estatísticas associadas.
- $createDataPartition()$: função que realiza uma divisão/partição de dados, cujos serão separados em dados para treino e dados para teste.

4.4.2 *Base*, *Utils* e *Stats*

Tanto a biblioteca/pacote *Base*, *Utils* e *Stats* são pacotes básicos da linguagem R, que não necessitam de importação e instalação pois já estão inclusos na linguagem[2].

Dentre as funções de ambos, as que foram utilizadas dentro do desenvolvimento foram, juntamente com uma breve descrição da utilização:

- $install.packages()$: função do pacote *utils* para instalar pacotes/bibliotecas necessários;
- $library()$: função do pacote *base* para carregar os pacotes/bibliotecas utilizadas;
- $remove()$: função do pacote *base* para, no momento do início da execução, fazer a limpeza de toda e qualquer variável que possa vir a atrapalhar a execução;

[1] https://cran.r-project.org/web/packages/caret/caret.pdf
[2] https://cran.r-project.org/doc/manuals/r-release/R-exts.html

- $setwd()$: função do pacote *base* para setar o diretório de trabalho;
- $gc()$: função do pacote *base* para liberar memória não utilizada;
- $read.csv()$: função do pacote *utils* para realizar a leitura do arquivo do dataset;
- $factor()$: função do pacote *base* para converter os dados do atributo *target* para um fator;
- $c()$: - função do pacote *base* para criar um conjunto/lista de dados;
- $summary()$: função do pacote *base* para realizar um resumo sobre os dados contidos em cada um dos atributos;
- $na.omit()$: função do pacote *stats* para realizar a remoção dos dados nulos/inválidos.

4.5 ANÁLISE EXPLANATÓRIA DOS *DATASETS*

Nesta seção são apresentados os três *datasets* escolhidos para a aplicação do algoritmo *Random Forest* (RF) e é mostrada uma análise explanatória dos dados e sua estrutura que cada um deles contém. Todos os três *datasets* selecionados foram retirados do site *Kaggle* [3], ambos de uma mesma área de aplicação, que foi a área da saúde, porém com temas diferentes e cada um com suas particularidades, conforme serão descritos a seguir.

4.5.1 Dados de Previsão de Diabetes (*Diabetes Prediction Dataset*)

O primeiro *dataset* selecionado foi o Conjunto de Dados de Previsão de Diabetes (*Diabetes Prediction Dataset*). Esse é um *dataset* que possui exatamente 100.000 dados, sendo nenhum deles nulo e/ou inválido e estão armazenados em um arquivo .csv. Além disso, possui um total de 9 atributos, sendo 2 do tipo texto, 3 do tipo decimal e 4 do tipo inteiro. A Tabela 7 mostra o nome de cada um desses, juntamente com o tipo e uma breve explicação de cada variável.

Esse *dataset* foi selecionado devido a ser relacionado a área da saúde, tema principal dos datasets selecionados e por não possuir nenhuma anomalia em seus dados, visando a análise da aplicação do algoritmo de RF em um conjunto de dados totalmente balanceado e sem inconsistências. A ideia de aplicação foi a trativa de um problema de classificação, onde buscou-se classificar e prever se o paciente possui diabetes com base na combinação dos outros atributos.

[3] https://www.kaggle.com/

Tabela 7 – Atributos *Diabetes Prediction Dataset*

Atributo	Tipo	Descrição
gender	texto	Variável que refere-se ao sexo biológico do indivíduo. Existem três categorias: masculino, feminino e outros
age	decimal	Variável que refere-se a idade do indivíduo. É um fator importante pois diabetes é mais comum em adultos mais velho. A idade varia entre 0 e 80
hypertension	inteiro	Variável que indica se o indivíduo possui hipertensão ou não. Os valores são: 0 indica que não têm hipertensão e 1 significa que têm hipertensão
heart_disease	inteiro	Variável que indica se o indivíduo possuí doença cardíaca ou não. Os valores são: 0 indica que não têm doenças cardíacas e 1 significa que têm doenças cardíacas
smoking_history	texto	Variável que representa o histórico de tabagismo. As categorias são: não atual, antigo, sem informações, atual e nunca.
bmi	decimal	Variável que indica o valor do IMC, onde valores altos estão associados a um maior risco de diabetes. A faixa de IMC nesse *dataset* é de 10,16 a 71,55. IMC abaixo de 18,5 é considerado abaixo do peso, 18,5-24,9 é normal, 25-29,9 está acima do peso e 30 ou mais é obeso
HbA1c_level	decimal	Variável que indica o nível de hemoglobina A1C, que é a medida do nível médio de açúcar no sangue do indivíduo nos últimos 2-3 meses. Níveis acima de 6,5% indica diabetes
blood_glucose_level	inteiro	Variável que indica o nível de glicose no sangue
diabetes	inteiro	Variável alvo prevista. Os valores são: 1 indicando a presença de diabetes e 0 indicando a ausência de diabetes

Fonte: Adaptado de *Kaggle*. Disponível em: <https://www.kaggle.com/>. Acesso em 16 jun. 2024.

Para este *dataset*, o atributo *target* selecionado para a aplicação do algoritmo e classificação foi o atributo *diabetes*, cujo conforme explanado na Tabela 7, é o atributo que indica a presença de diabetes ou não.

Para complementar a explanação e análise inicial dos dados desse *dataset*, podemos indicar o resultado obtido pelo Algoritmo 2.

Algoritmo 2 – Código em R para análise inicial dos dados do *Diabetes Prediction Dataset*

```
1
2  ### LEITURA DOS DADOS
3  dados = read.csv("diabetes_prediction_dataset.csv", header = T, sep = ",")
4
5  ### MOSTRA RESUMO DOS DADOS
6  summary(dados)
7
8  ### MOSTRA UMA PEQUENA AMOSTRA DOS DADOS
9  str(dados)
```

Fonte: O Autor (2024).

O resultado da execução das funções $summary()$ e $str()$ pode ser visto na Figura 16 e Figura 17, respectivamente.

Figura 16 – Resultado execução $summary()$

```
   gender              age         hypertension     heart_disease     smoking_history         bmi
Length:100000      Min.   : 0.08   Min.   :0.00000   Min.   :0.00000   Length:100000      Min.   :10.01
Class :character   1st Qu.:24.00   1st Qu.:0.00000   1st Qu.:0.00000   Class :character   1st Qu.:23.63
Mode  :character   Median :43.00   Median :0.00000   Median :0.00000   Mode  :character   Median :27.32
                   Mean   :41.89   Mean   :0.07485   Mean   :0.03942                      Mean   :27.32
                   3rd Qu.:60.00   3rd Qu.:0.00000   3rd Qu.:0.00000                      3rd Qu.:29.58
                   Max.   :80.00   Max.   :1.00000   Max.   :1.00000                      Max.   :95.69
 HbA1c_level    blood_glucose_level diabetes
Min.   :3.500   Min.   : 80.0       0:91500
1st Qu.:4.800   1st Qu.:100.0       1: 8500
Median :5.800   Median :140.0
Mean   :5.528   Mean   :138.1
3rd Qu.:6.200   3rd Qu.:159.0
Max.   :9.000   Max.   :300.0
```

Fonte: O Autor (2024).

Figura 17 – Resultado execução $str()$

```
'data.frame':   100000 obs. of  9 variables:
 $ gender              : chr  "Female" "Female" "Male" "Female" ...
 $ age                 : num  80 54 28 36 76 20 44 79 42 32 ...
 $ hypertension        : int  0 0 0 0 1 0 0 0 0 0 ...
 $ heart_disease       : int  1 0 0 0 1 0 0 0 0 0 ...
 $ smoking_history     : chr  "never" "No Info" "never" "current" ...
 $ bmi                 : num  25.2 27.3 27.3 23.4 20.1 ...
 $ HbA1c_level         : num  6.6 6.6 5.7 5 4.8 6.6 6.5 5.7 4.8 5 ...
 $ blood_glucose_level: int  140 80 158 155 155 85 200 85 145 100 ...
 $ diabetes            : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 2 1 1 1 ...
```

Fonte: O Autor (2024).

Conforme pode-se analisar, a função utilizada foi a $read.csv$, cujos parâmetros passados foram o nome do arquivo, seguido da indicação de que haviam cabeçalhos no arquivo ($Header = T$) e qual foi o caracter separador entre as colunas que, nesse caso, foi a ",".

4.5.2 Indicadores de Doenças Cardíacas (*Indicators of Heart Disease Dataset*)

O segundo *dataset* selecionado foi o Conjunto de Dados de Indicadores de Doença Cardíaca (*Indicators of Heart Disease Dataset*). Esse é um *dataset* que possui aproximadamente 450.000 dados, cujos possuem dados nulos espalhados entre seus atributos e estão armazenados um arquivo .csv. Além disso, possui um total de 40 atributos, sendo 12 do tipo texto, 6 do tipo decimal e 22 do tipo booleano. A Tabela 8 mostra o nome de cada um desses, juntamente com o tipo e uma breve explicação de cada variável. Ademais, será aplicado o algoritmo de RF buscando verificar como se comporta quando possui dados nulos.

Esse *dataset* foi selecionado devido à ser relacionado a área da saúde, tema principal dos datasets selecionados e por possuir anomalias em seus dados como dados nulos e afins. Dessa forma, será analisado como o algoritmo de RF se comporta quando aplicado em datasets

não homogêneos e com dados faltantes em problema de classificação, onde busca-se classificar e prever se o paciente possui doença cardíaca ou não com base na combinação de atributos.

Tabela 8 – Atributos *Indicators of Heart Disease Dataset*

Atributo	Tipo	Descrição
State	texto	Indica o país de naturalidade do paciente
Sex	texto	Indica qual o sexo do paciente
GeneralHealth	texto	Indica qual o sexto do paciente
PhysicalHealthDays	decimal	Valor que indica quantos dias de problema na saúde física o paciente teve
MentalHealthDays	decimal	Valor que indica quantos dias de problema na saúde mental o paciente teve
LastCheckupTime	texto	Indica quando foi o último check-up
PhysicalActivities	booleano	Indica se o paciente pratica atividades físicas
SleepHours	decimal	Valor de quantas horas de sono o paciente tem por dia
RemovedTeeth	texto	Indica a quantidade de dentes removidos que o paciente já teve
HadHeartAttack	booleano	Indica se o paciente teve ataque cardíaco
HadAngina	booleano	Indica se o paciente já teve Angina
HadStroke	booleano	Indica se o paciente já teve um AVC
HadAsthma	booleano	Indica se o paciente já teve asma
HadSkinCancer	booleano	Indica se o paciente já teve câncer de pele
HadCOPD	booleano	Indica se o paciente possui doença pulmonar obstrutiva crônica
HadDepressiveDisorder	booleano	Indica se o paciente já teve transtornos depressivos
HadKidneyDisease	booleano	Indica se o paciente já teve doença renal
HadArthritis	booleano	Indica se o paciente já teve artrite
HadDiabetes	texto	Indica se o paciente tem diabetes
DeafOrHardOfHearing	booleano	Indica se o paciente é surdo ou tem deficiência na audição
BlindOrVisionDifficulty	booleano	Indica se o paciente é cego ou tem dificuldade na visão
DifficultyConcentrating	booleano	Indica se o paciente tem dificuldade em se concentrar
DifficultyWalking	booleano	Indica se o paciente tem dificuldade em caminhar
DifficultyDressingBathing	booleano	Indica se o paciente tem dificuldade em se vestir
DifficultyErrands	booleano	Indica se o paciente tem dificuldade em fazer tarefas sozinho
SmokerStatus	texto	Indica se o paciente é fumante
ECigaretteUsage	texto	Indica se o paciente já fumou cigarro eletrônico
ChestScan	booleano	Indica se o paciente fez tomografia de tórax
RaceEthnicityCategory	texto	Indica qual categoria de raça o paciente se enquadra
AgeCategory	texto	Indica em qual categoria de idade o paciente se encontra
HeightInMeters	decimal	Valor da altura em metros (m)
WeightInKilograms	decimal	Valor do peso em kilogramas (kg)
BMI	decimal	Valor do IMC - Índice de Massa Corporal
AlcoholDrinkers	booleano	Indica se o paciente é usuário de álcool
HIVTesting	booleano	Indica se o paciente fez teste de HIV
FluVaxLast12	booleano	Indica se o paciente recebeu vacina contra gripe nos últimos 12 anos
PneumoVaxEver	booleano	Indica se o paciente já recebeu vacina contra doença pneumocócica
TetanusLast10Tdap	texto	Indica se o paciente recebeu vacina contra tétano nos últimos 10 amos
HighRiskLastYear	booleano	Indica se é um paciente de alto risco
CovidPos	texto	Indica se o paciente teve testou positivo para COVID

Fonte: Adaptado de *Kaggle*. Disponível em: <https://www.kaggle.com/>. Acesso em 20 out. 2024.

Para este *dataset*, o atributo *target* selecionado para a aplicação do algoritmo e classificação foi o atributo *HadHeartAttack*, cujo conforme explanado na Tabela 7, é o atributo que indica se o paciente teve ou não ataque cardíaco.

Para complementar a explanação e análise inicial dos dados desse *dataset*, podemos

indicar o resultado obtido pelo Algoritmo 3.

Algoritmo 3 – Código em R para análise inicial dos dados do *Indicators of Heart Disease Dataset*

```
1
2  ### LEITURA DOS DADOS
3  dados = read.csv("heart_2022_with_nans_dataset.csv", header = T, sep = ";")
4
5  ### MOSTRA RESUMO DOS DADOS
6  summary(dados)
7
8  ### MOSTRA UMA PEQUENA AMOSTRA DOS DADOS
9  str(dados)
```

Fonte: O Autor (2024).

O resultado da execução das funções $summary()$ e $str()$ pode ser visto na Figura 18 e Figura 19, respectivamente.

Figura 18 – Resultado execução $summary()$

```
   State              Sex            GeneralHealth      PhysicalHealthDays MentalHealthDays
Length:445132      Length:445132      Length:445132      Min.   : 0.000     Min.   : 0.000
Class :character   Class :character   Class :character   1st Qu.: 0.000     1st Qu.: 0.000
Mode  :character   Mode  :character   Mode  :character   Median : 0.000     Median : 0.000
                                                         Mean   : 4.348     Mean   : 4.383
                                                         3rd Qu.: 3.000     3rd Qu.: 5.000
                                                         Max.   :30.000     Max.   :30.000
                                                         NA's   :10927      NA's   :9067
LastCheckupTime    PhysicalActivities   SleepHours     RemovedTeeth       HadHeartAttack
Length:445132      Length:445132      Min.   : 1.000   Length:445132      Yes : 25108
Class :character   Class :character   1st Qu.: 6.000   Class :character   No  :416959
Mode  :character   Mode  :character   Median : 7.000   Mode  :character   NA's:  3065
                                      Mean   : 7.023
                                      3rd Qu.: 8.000
                                      Max.   :24.000
                                      NA's   :5453
 HadAngina          HadStroke          HadAsthma          HadSkinCancer        HadCOPD
Length:445132      Length:445132      Length:445132      Length:445132      Length:445132
Class :character   Class :character   Class :character   Class :character   Class :character
Mode  :character   Mode  :character   Mode  :character   Mode  :character   Mode  :character

 HadDepressiveDisorder HadKidneyDisease   HadArthritis       HadDiabetes
 Length:445132         Length:445132      Length:445132      Length:445132
 Class :character      Class :character   Class :character   Class :character
 Mode  :character      Mode  :character   Mode  :character   Mode  :character
DeafOrHardOfHearing BlindOrVisionDifficulty DifficultyConcentrating DifficultyWalking
Length:445132       Length:445132           Length:445132           Length:445132
Class :character    Class :character        Class :character        Class :character
Mode  :character    Mode  :character        Mode  :character        Mode  :character

DeafOrHardOfHearing BlindOrVisionDifficulty DifficultyConcentrating DifficultyWalking
Length:445132       Length:445132           Length:445132           Length:445132
Class :character    Class :character        Class :character        Class :character
Mode  :character    Mode  :character        Mode  :character        Mode  :character



DifficultyDressingBathing DifficultyErrands  SmokerStatus       ECigaretteUsage
Length:445132             Length:445132      Length:445132      Length:445132
Class :character          Class :character   Class :character   Class :character
Mode  :character          Mode  :character   Mode  :character   Mode  :character



 ChestScan          RaceEthnicityCategory AgeCategory        HeightInMeters  WeightInKilograms
Length:445132      Length:445132         Length:445132      Min.   :0.910   Min.   : 22.68
Class :character   Class :character      Class :character   1st Qu.:1.630   1st Qu.: 68.04
Mode  :character   Mode  :character      Mode  :character   Median :1.700   Median : 80.74
                                                           Mean   :1.703   Mean   : 83.07
                                                           3rd Qu.:1.780   3rd Qu.: 95.25
                                                           Max.   :2.410   Max.   :292.57
                                                           NA's   :28652   NA's   :42078
     BMI         AlcoholDrinkers     HIVTesting        FluVaxLast12       PneumoVaxEver
Min.   :12.02   Length:445132      Length:445132      Length:445132      Length:445132
1st Qu.:24.13   Class :character   Class :character   Class :character   Class :character
Median :27.44   Mode  :character   Mode  :character   Mode  :character   Mode  :character
Mean   :28.53
3rd Qu.:31.75
Max.   :99.64
NA's   :48806
TetanusLast10Tdap  HighRiskLastYear     CovidPos
Length:445132      Length:445132      Length:445132
Class :character   Class :character   Class :character
Mode  :character   Mode  :character   Mode  :character
```

Fonte: O Autor (2024).

Figura 19 – Resultado execução $str()$

```
'data.frame':   445132 obs. of  40 variables:
 $ State                   : chr  "Alabama" "Alabama" "Alabama" "Alabama" ...
 $ Sex                     : chr  "Female" "Female" "Female" "Female" ...
 $ GeneralHealth           : chr  "very good" "Excellent" "Very good" "Excellent" ...
 $ PhysicalHealthDays      : num  0 0 2 0 2 1 0 0 0 1 ...
 $ MentalHealthDays        : num  0 0 3 0 0 0 0 0 0 0 ...
 $ LastCheckupTime         : chr  "Within past year (anytime less than 12 months ago)" "" "Within past year (anytime less than 12
months ago)" "Within past year (anytime less than 12 months ago)" ...
 $ PhysicalActivities      : chr  "No" "No" "Yes" "Yes" ...
 $ SleepHours              : num  8 6 5 7 9 7 7 8 6 7 ...
 $ RemovedTeeth            : chr  "" "" "" "" ...
 $ HadHeartAttack          : Factor w/ 2 levels "Yes","No": 2 2 2 2 2 1 2 2 2 2 ...
 $ HadAngina               : chr  "No" "No" "No" "No" ...
 $ HadStroke               : chr  "No" "No" "No" "No" ...
 $ HadAsthma               : chr  "No" "No" "No" "Yes" ...
 $ HadSkinCancer           : chr  "No" "Yes" "Yes" "No" ...
 $ HadCOPD                 : chr  "No" "No" "No" "No" ...
 $ HadDepressiveDisorder   : chr  "No" "No" "No" "No" ...
 $ HadKidneyDisease        : chr  "No" "No" "No" "No" ...
 $ HadArthritis            : chr  "No" "No" "No" "Yes" ...
 $ HadDiabetes             : chr  "Yes" "No" "No" "No" ...
 $ DeafOrHardOfHearing     : chr  "No" "No" "No" "No" ...
 $ BlindOrVisionDifficulty : chr  "No" "No" "No" "No" ...
 $ DifficultyConcentrating : chr  "No" "No" "No" "No" ...
 $ DifficultyWalking       : chr  "No" "No" "No" "No" ...
 $ DifficultyDressingBathing: chr  "No" "No" "No" "No" ...
 $ DifficultyErrands       : chr  "No" "No" "No" "No" ...
 $ SmokerStatus            : chr  "Never smoked" "Never smoked" "Never smoked" "Current smoker - now smokes some days" ...
 $ ECigaretteUsage         : chr  "Not at all (right now)" "Never used e-cigarettes in my entire life" "Never used e-cigarettes in
my entire life" "Never used e-cigarettes in my entire life" ...
 $ ChestScan               : chr  "No" "No" "No" "Yes" ...
 $ RaceEthnicityCategory   : chr  "White only, Non-Hispanic" "White only, Non-Hispanic" "White only, Non-Hispanic" "White only, No
n-Hispanic" ...
 $ AgeCategory             : chr  "Age 80 or older" "Age 80 or older" "Age 55 to 59" "" ...
 $ HeightInMeters          : num  NA 1.6 1.57 1.65 1.57 1.8 1.65 1.63 1.7 1.68 ...
 $ WeightInKilograms       : num  NA 68 63.5 63.5 54 ...
 $ BMI                     : num  NA 26.6 25.6 23.3 21.8 ...
 $ AlcoholDrinkers         : chr  "No" "No" "No" "No" ...
 $ HIVTesting              : chr  "No" "No" "No" "No" ...
 $ FluVaxLast12            : chr  "Yes" "No" "No" "Yes" ...
 $ PneumoVaxEver           : chr  "No" "No" "No" "Yes" ...
 $ TetanusLast10Tdap       : chr  "Yes, received tetanus shot but not sure what type" "No, did not receive any tetanus shot in the
past 10 years" "" "No, did not receive any tetanus shot in the past 10 years" ...
 $ HighRiskLastYear        : chr  "No" "No" "No" "No" ...
 $ CovidPos                : chr  "No" "No" "Yes" "No" ...
```

Fonte: O Autor (2024).

Conforme pode-se analisar, a função utilizada foi a $read.csv$, cujos parâmetros passados foram o nome do arquivo, seguido da indicação de que haviam cabeçalhos no arquivo ($Header = T$) e qual foi o caracter separador entre as colunas que, nesse caso, foi o ";".

4.5.3 Previsão de Gordura Corporal (*Body Fat Prediction Dataset*)

O terceiro *dataset* selecionado foi o Conjunto de Dados de Previsão de Gordura Corporal (*Body Fat Prediction Dataset*). Esse é um *dataset* relativamente pequeno, pois reuniu somente 252 dados, onde todas amostras são as medidas de homens. Em meio a esses dados, não possuem dados nulos espalhados entre seus atributos e estão armazenados um arquivo .csv. Além disso, possui um total de 15 atributos, sendo 14 do tipo decimal e 1 do tipo inteiro. A Tabela 9 mostra o nome de cada um desses, juntamente com o tipo e uma breve explicação de cada variável.

Esse *dataset* foi selecionado devido à ser relacionado a área da saúde, tema principal dos datasets selecionados e por possuir anomalias em seus dados como dados nulos e afins. Dessa forma, será analisado como o algoritmo de RF se comporta quando aplicado em datasets não homogêneos e com dados faltantes para um problema de regressão, onde busca-se predizer

qual será o valor da gordura corporal baseado nos outros atributos e cálculos.

Tabela 9 – Atributos *Body Fat Prediction Dataset*

Atributo	**Tipo**	**Descrição**
Density	decimal	Valor da densidade determinada por pesagem subaquática
BodyFat	decimal	Valor da Porcentagem de gordura corporal calculado pela equação de Siri (1956)
Age	inteiro	Valor da Idade (anos)
Weight	decimal	Valor do Peso (lbs)
Height	decimal	Valor da Altura (polegadas)
Neck	decimal	Valor da Circunferência do pescoço (cm)
Chest	decimal	Valor da Circunferência do peito (cm)
Abdomen	decimal	Valor da Circunferência do abdômen (cm)
Hip	decimal	Valor da Circunferência do quadril (cm)
Thigh	decimal	Valor da Circunferência da coxa (cm)
Knee	decimal	Valor da Circunferência do joelho (cm)
Ankle	decimal	Valor da Circunferência do tornozelo (cm)
Biceps	decimal	Valor da Circunferência do bíceps (estendido) (cm)
Forearm	decimal	Valor da Circunferência do antebraço (cm)
Wrist	decimal	Valor da Circunferência do pulso (cm)

Fonte: Adaptado de *Kaggle*. Disponível em: <https://www.kaggle.com/>. Acesso em 20 out. 2024.

Para este *dataset*, o atributo *target* selecionado para a aplicação do algoritmo realizando regressão foi o atributo *BodyFat*, cujo conforme explanado na Tabela 7, é o atributo representa o valor da porcentagem de gordura corporal que o paciente possui.

Para complementar a explanação e análise inicial dos dados desse *dataset*, podemos indicar o resultado obtido pelo Algoritmo 4.

Algoritmo 4 – Código em R para análise inicial dos dados do *Body Fat Prediction Dataset*

```
1
2  ### LEITURA DOS DADOS
3  dados = read.csv("bodyfat_dataset.csv", header = T, sep = ";")
4
5  ### MOSTRA RESUMO DOS DADOS
6  summary(dados)
7
8  ### MOSTRA UMA PEQUENA AMOSTRA DOS DADOS
9  str(dados)
```

Fonte: O Autor (2024).

O resultado da execução das funções $summary()$ e $str()$ pode ser visto na Figura 20 e Figura 21, respectivamente.

Figura 20 – Resultado execução $summary()$

```
    Density          BodyFat           Age            Weight          Height           Neck      
 Min.   : 0.995   Min.   : 0.00   Min.   :22.00   Min.   :118.5   Min.   :29.50   Min.   :31.10  
 1st Qu.:10.414   1st Qu.:12.47   1st Qu.:35.75   1st Qu.:159.0   1st Qu.:68.25   1st Qu.:36.40  
 Median :10.549   Median :19.20   Median :43.00   Median :176.5   Median :70.00   Median :38.00  
 Mean   :10.520   Mean   :19.15   Mean   :44.88   Mean   :178.9   Mean   :70.15   Mean   :37.99  
 3rd Qu.:10.704   3rd Qu.:25.30   3rd Qu.:54.00   3rd Qu.:197.0   3rd Qu.:72.25   3rd Qu.:39.42  
 Max.   :11.089   Max.   :47.50   Max.   :81.00   Max.   :363.1   Max.   :77.75   Max.   :51.20  
     Chest           Abdomen            Hip            Thigh           Knee           Ankle      
 Min.   : 79.30   Min.   : 69.40   Min.   : 85.0   Min.   :47.20   Min.   :33.00   Min.   :19.1  
 1st Qu.: 94.35   1st Qu.: 84.58   1st Qu.: 95.5   1st Qu.:56.00   1st Qu.:36.98   1st Qu.:22.0  
 Median : 99.65   Median : 90.95   Median : 99.3   Median :59.00   Median :38.50   Median :22.8  
 Mean   :100.82   Mean   : 92.56   Mean   : 99.9   Mean   :59.41   Mean   :38.59   Mean   :23.1  
 3rd Qu.:105.38   3rd Qu.: 99.33   3rd Qu.:103.5   3rd Qu.:62.35   3rd Qu.:39.92   3rd Qu.:24.0  
 Max.   :136.20   Max.   :148.10   Max.   :147.7   Max.   :87.30   Max.   :49.10   Max.   :33.9  
     Biceps          Forearm          Wrist      
 Min.   :24.80   Min.   :21.00   Min.   :15.80  
 1st Qu.:30.20   1st Qu.:27.30   1st Qu.:17.60  
 Median :32.05   Median :28.70   Median :18.30  
 Mean   :32.27   Mean   :28.66   Mean   :18.23  
 3rd Qu.:34.33   3rd Qu.:30.00   3rd Qu.:18.80  
 Max.   :45.00   Max.   :34.90   Max.   :21.40  
```

Fonte: O Autor (2024).

Figura 21 – Resultado execução $str()$

```
'data.frame':   252 obs. of  15 variables:
 $ Density: num  10.7 10.9 10.4 10.8 10.3 ...
 $ BodyFat: num  12.3 6.1 25.3 10.4 28.7 20.9 19.2 12.4 4.1 11.7 ...
 $ Age    : int  23 22 22 26 24 24 26 25 25 23 ...
 $ Weight : num  154 173 154 185 184 ...
 $ Height : num  67.8 72.2 66.2 72.2 71.2 ...
 $ Neck   : num  36.2 38.5 34 37.4 34.4 39 36.4 37.8 38.1 42.1 ...
 $ Chest  : num  93.1 93.6 95.8 101.8 97.3 ...
 $ Abdomen: num  85.2 83 87.9 86.4 100 94.4 90.7 88.5 82.5 88.6 ...
 $ Hip    : num  94.5 98.7 99.2 101.2 101.9 ...
 $ Thigh  : num  59 58.7 59.6 60.1 63.2 66 58.4 60 62.9 63.1 ...
 $ Knee   : num  37.3 37.3 38.9 37.3 42.2 42 38.3 39.4 38.3 41.7 ...
 $ Ankle  : num  21.9 23.4 24 22.8 24 25.6 22.9 23.2 23.8 25 ...
 $ Biceps : num  32 30.5 28.8 32.4 32.2 35.7 31.9 30.5 35.9 35.6 ...
 $ Forearm: num  27.4 28.9 25.2 29.4 27.7 30.6 27.8 29 31.1 30 ...
 $ Wrist  : num  17.1 18.2 16.6 18.2 17.7 18.8 17.7 18.8 18.2 19.2 ...
```

Fonte: O Autor (2024).

Conforme pode-se analisar, a função utilizada foi a $read.csv$, cujos parâmetros passados foram o nome do arquivo, seguido da indicação de que haviam cabeçalhos no arquivo ($Header = T$) e qual foi o caracter separador entre as colunas que, nesse caso, foi o ";".

4.6 PREPARAÇÃO, PRÉ-PROCESSAMENTO E TRANSFORMAÇÃO DOS DADOS DOS *DATASETS*

Todos três *datasets* passaram por uma preparação, pré-processamento e transformação inicial comum para todos. Toda essa sequência de funções executadas está descrita conforme abaixo:

1. Conversão do atributo *target* dos *datasets Diabetes Prediction Dataset* (Seção 4.5.1) e *Indicators of Heart Disease Dataset* (Seção 4.5.2) para *factor*. Como o *dataset Body Fat Prediction Dataset* (Seção 4.5.3) foi utilizado para regressão, não é necessária a conversão do atributo. Para essas conversões foi utilizada a função $factor()$, cujos parâmetros passados foram o atributo *target* e quais eram os *levels* do atributo em questão;

2. Foi realizada a remoção das linhas que possuíam dados nulos e/ou inválidos utilizando da função $na.omit()$, cujo parâmetro de entrada foi a variável que recebeu a leitura dos dados.

3. Foi utilizada da função $createDataPartition()$ para realizar a criação das partições dos dados, que posteriormente seria usado para separar em dados para treino e dados para teste. Os parâmetros utilizados para execução dessa função foram: a variável *target*, qual seria a porcentagem de divisão, que para todos os casos foi de 0.75 (75%); para o parâmetro *times*, foi utilizado o valor 1, representado que só seria realizada uma partição e *false* para o parâmetro *list*, cujo qual se fosse *true*, iria salvar os dados em uma lista ao invés de uma matriz;

4. Foi realizada a separação dos dados entre dados de treino e dados de teste. Para todos 3 *datasets* foram realizadas atribuições padrões de dados, partindo dos dados particionados que foram gerados anteriormente;

5. Por fim, para os *datasets* em que seria aplicado o algoritmo de RF visando classificação, foi realizada também a conversão dos atributos *target* para *factor*, utilizando da mesma estrutura que o passo 1.

O Algoritmo 5, Algoritmo 6 e Algoritmo 7 explicitam os códigos propriamente ditos que foram executados para cada um dos *datasets*.

Algoritmo 5 – Código em R de preparação dos dados do *Diabetes Prediction Dataset*

```
1
2 ### CONVERTER ATRIBUTO TARGET PARA FACTOR
3 dados$diabetes <- factor(dados$diabetes, levels = c('0','1'))
4
5 ### REMOVER AMOSTRAS COM DADOS INVALIDOS/NULOS
6 dados = na.omit(dados)
```

```
7
8  ### CRIAR PARTICAO DOS DADOS PARA DIVISAO ENTRE TREINO E TESTE
9  dadosParticioados <- createDataPartition(y = dados$diabetes,
10                                           p = 0.75,
11                                           times = 1,
12                                           list = FALSE)
13
14 ### SEPARAR OS DADOS DE TREINO E TESTE
15 dados.treino <- dados[dadosParticioados,]
16 dados.teste  <- dados[-dadosParticioados,]
17
18 ### CONVERTER ATRIBUTOS TARGETS PARA FACTOR
19 dados.treino$diabetes <- factor(dados.treino$diabetes, levels = c('0','1'))
20 dados.teste$diabetes  <- factor(dados.teste$diabetes, levels = c('0','1'))
```

Fonte: O Autor (2024).

Algoritmo 6 – Código em R de preparação dos dados do *Indicators of Heart Disease Dataset*

```
1
2  ### CONVERTER ATRIBUTO TARGET PARA FACTOR
3  dados$HadHeartAttack <- factor(dados$HadHeartAttack,
4                                 levels = c("Yes","No"))
5
6  ### REMOVER AMOSTRAS COM DADOS INVALIDOS/NULOS
7  dados = na.omit(dados)
8
9  ### CRIAR PARTICAO DOS DADOS PARA DIVISAO ENTRE TREINO E TESTE
10 dadosParticioados <- createDataPartition(y = dados$HadHeartAttack,
11                                          p = 0.75,
12                                          times = 1,
13                                          list = FALSE)
14
15 ### SEPARAR OS DADOS DE TREINO E TESTE
16 dados.treino <- dados[dadosParticioados,]
17 dados.teste  <- dados[-dadosParticioados,]
18
19 ### CONVERTER ATRIBUTOS TARGETS PARA FACTOR
20 dados.treino$HadHeartAttack <- factor(dados.treino$HadHeartAttack,
21                                        levels = c("Yes","No"))
22 dados.teste$HadHeartAttack  <- factor(dados.teste$HadHeartAttack,
23                                        levels = c("Yes","No"))
```

Fonte: O Autor (2024).

Algoritmo 7 – Código em R de preparação dos dados do *Body Fat Prediction Dataset*

```
1
2  ### REMOVER AMOSTRAS COM DADOS INVALIDOS/NULOS
```

```
3   dados = na.omit(dados)
4
5   ### INICIALIZAR VARIAVEIS AUXILIARES
6   dados.x <- dados[, -2]
7   dados.y <- dados$BodyFat
8
9   ### CRIAR PARTICAO DOS DADOS PARA DIVISAO ENTRE TREINO E TESTE
10  dadosParticioados <- createDataPartition(dados.y,
11                                           p = 0.75,
12                                           times = 1,
13                                           list = FALSE)
14
15  ### SEPARAR OS DADOS DE TREINO E TESTE
16  dados.treino.x <- dados.x[dadosParticioados,]
17  dados.treino.y <- dados.y[dadosParticioados]
18  dados.teste.x  <- dados.x[-dadosParticioados,]
19  dados.teste.y  <- dados.y[-dadosParticioados]
```

Fonte: O Autor (2024).

Para fins de análise, após execução da função $na.omit()$, o primeiro *dataset* e o terceiro *dataset* não tiveram nenhum de seus dados removidos, enquanto o segundo *dataset* teve uma redução de pouco mais de 66.000 amostras. Com certeza esses dados faltantes impactam muito na aplicação dos algoritmos.

4.7 TESTES DA APLICAÇÃO DO ALGORITMO DE *RANDOM FOREST*

Conforme já explanado no Capítulo 4, foram realizados testes e foi realizada aplicação do algoritmo de RF nos *datasets* selecionados e será explicitado conforme abaixo o código executado juntamente com uma explicação sobre o que foi feito.

Para os *datasets Diabetes Prediction Dataset* e *Indicators of Heart Disease Dataset*, o que basicamente foi executado está dentro do Algoritmo 8 e Algoritmo 9, cuja explicação sobre os parâmetros passados nas funções estão descritos abaixo:

1. Foi aplicada a função $tuneRF()$ para descobrir qual seria o valor ideal para o parâmetro *mtry*. Nela foram passados os parâmetros conforme abaixo:
    - $dados.treino[, -NUMERO\_COLUNA\_TARGET]$: conjunto de dados de treino, retirando a coluna do atributo *target*;
    - $dados.treino[, NUMERO\_COLUNA\_TARGET]$: conjunto de dados de treino, passando somente a coluna do atributo *target*
    - $stepFactor = 1.5$: valor que refere-se ao quando varia o valor de *mtry* para cada um dos testes;

- $improve = 0.01$: valor que refere-se ao número mínimo de melhora necessário para considerar o valor de *mtry*. Por exemplo, setado como 0.01, é necessário que a cada teste do parâmetro, tenha pelo menos 0.01 de melhora na execução;
- $ntreeTry = 50$: valor padrão do parâmetro. Refere-se ao número de estimadores realizados;
- $trace = TRUE$: indicado como *TRUE* para que fosse possível verificar o status da execução e quais os valores tentados;

2. Foi aplicado o algoritmo do *Random Forest* (nomeado pela função $randomForest()$) propriamente dito. Nele foram passados os parâmetros essenciais para sua execução, variando apenas os parâmetros conforme supracitado. Abaixo segue, de maneira mais genérica, a descrição dos parâmetros passados para a função:

   - $x = dados.treino[, -COLUNA\_ATRIBUTO\_TARGET]$: representa a atribuição de todos os dados e atributos excluindo somente o atributo *target* na variável x;
   - $y = dados.treino[, COLUNA\_ATRIBUTO\_TARGET]$: representa a atribuição de todas as linhas do atributo *target* na variável y;
   - $importance = TRUE$: definido como *TRUE* para que seja possível verificar a importância de cada um dos atributos do *dataset*, baseado em dois critérios principais, segundo a documentação oficial *randomForest* da linguagem R [4].
     - *MeanDecreaseGini*: é responsável por indicar o quanto a acurácia é reduzida quando essa variável é reduzida. Valores maiores indicam que a variável é mais importante para a precisão do modelo.
     - *MeanDecreaseAccuracy*: é responsável por indicar a importância da variável na pureza de cada um dos nós de cada árvore. Assim como o *MeanDecreaseGini*, quanto maior o valor, significa que a variável contribui mais na divisão das árvores.
   - $ntree = ntree$: representa a atribuição do número máximo de árvores de decisão que seriam geradas;
   - $mtry = mtry$: representa a atribuição do número de atributos selecionados de maneira aleatória para realizar a criação de cada árvore de decisão;
   - $nodesize = 1$: representa o número máximo que os nodos de cada árvore pode ter. Foi setado como 1 pois é um problema de classificação;
   - $keep.forest = TRUE$: definido como *TRUE* para que a floresta seja retida no objeto de saída.

[4] https://cran.r-project.org/

3. Foi feita a execução da função $predict()$ e atribuído o resultado na variável *predicao*, ficando da seguinte forma: $predicao = predict(rf, newdata = dados.teste)$. Os parâmetros passados foram o resultado gerado pela execução do algoritmo de RF;

4. Foi feita a execução da função $confusionMatriz()$ e atribuído o resultado na variável *matriz_confusao*, ficando da seguinte forma: $matriz\_confusao <- confusionMatrix(predicao, dados.teste\$NM\_ATRIB\_TARGET)$. Os parâmetros passados foram o resultado gerado pela execução da predição no passo anterior e somente o atributo *target* do conjunto de dados;

5. Foram executados, conforme apresentado abaixo, as atribuições e cálculos das variáveis que foram utilizadas para medição da análise de desempenho da aplicação do algoritmo:

   - $acuracia <- matriz\_confusao\$overall["Accuray"]$: representa a atribuição do valor da acurácia gerado após execução da função $confusionMatriz()$ no passo anterior;
   - $precisao <- matriz\_confusao\$overall["Pos\ Pred\ Value"]$: representa a atribuição do valor da precisão gerado após execução da função $confusionMatriz()$ no passo anterior;
   - $sensibilidade <- matriz\_confusao\$overall["Sensitivity"]$: representa a atribuição do valor da sensibilidade gerado após execução da função $confusionMatriz()$ no passo anterior;
   - $especificidade <- matriz\_confusao\$overall["Specificity"]$: representa a atribuição do valor da especificidade gerado após execução da função $confusionMatriz()$ no passo anterior;
   - $F1\_score <- 2 * (precisao * sensibilidade)/(precisao + sensibilidade)$: representa o cálculo do valor de *F1_score* com base nos valores das variáveis de precisão e sensibilidade geradas nos passos anteriores.

6. Foram executadas as funções, conforme apresentado abaixo, para verificar a importância dos atributos:

   - $importance(rf)$ e $varImp(rf)$: funções executadas para verificar a importância dos atributos;
   - $varImpPlot(rf, main = 'Importancia\ Atributos')$: função executada para realizar a plotagem das variáveis em relação a importância;

Algoritmo 8 – Código da Aplicação do RF no *Diabetes Prediction Dataset*

```
1
2   ### FUNCAO PARA VERIFICAR O NUMERO IDEAL DE MTRY
3   melhor_mtry <- tuneRF(dados.treino[, -9],
```

```
                            dados.treino[, 9],
                            stepFactor = 1.5,
                            improve = 0.01,
                            ntreeTry = 500,
                            trace = TRUE)

    ### ATRIBUICAO DO MELHOR VALOR DE MTRY PARA TESTE
    mtry <- melhor_mtry[melhor_mtry[, 2] == min(melhor_mtry[, 2]), 1]

    ### APLICACAO DO RANDOMFOREST
    rf<- randomForest(x = dados.treino[,-9],
                      y = dados.treino[,9],
                      importance = TRUE,
                      ntree = ntree,
                      mtry = mtry,
                      nodesize = 1,
                      keep.forest = TRUE)

    ### FAZER PREVISOES E GERAR A MATRIZ DE CONFUSAO
    predicao <- predict(rf, newdata = dados.teste)
    matriz_confusao <- confusionMatrix(predicao, dados.teste$diabetes)

    ### REALIZAR OS CALCULOS DAS METRICAS DE DESEMPENHO
    acuracia        <- matriz_confusao$overall["Accuracy"]
    precisao        <- matriz_confusao$byClass["Pos␣Pred␣Value"]
    sensibilidade   <- matriz_confusao$byClass["Sensitivity"]
    especificidade  <- matriz_confusao$byClass["Specificity"]
    F1_score        <- 2 * (precisao * sensibilidade) /
                           (precisao + sensibilidade)

    ### VERIFICAR IMPORTANCIA DOS ATRIBUTOS E PLOTAGEM DO RESULTADO
    importance(rf)
    varImpPlot(rf, main = 'Importancia␣Atributos')
```

Fonte: O Autor (2024).

Algoritmo 9 – Código da Aplicação do RF no *Indicators of Heart Disease Dataset*

```
### FUNCAO PARA VERIFICAR O NUMERO IDEAL DE MTRY
melhor_mtry <- tuneRF(dados.treino[, -10],
                      dados.treino[, 10],
                      stepFactor = 1.5,
                      improve = 0.01,
                      ntreeTry = 50,
                      trace = TRUE)

### ATRIBUICAO DO MELHOR VALOR DE MTRY PARA TESTE
mtry <- melhor_mtry[melhor_mtry[, 2] == min(melhor_mtry[, 2]), 1]
```

```
11
12 ### APLICACAO DO RANDOM FOREST
13 rf <- randomForest(x = dados.treino[,-10],
14                     y = dados.treino[,10],
15                     importance = TRUE,
16                     ntree = ntree,
17                     mtry = valor_mtry,
18                     nodesize = 1,
19                     keep.forest = TRUE)
20
21 ### FAZER PREVISOES E GERAR A MATRIZ DE CONFUSAO
22 predicao <- predict(rf, newdata = dados.teste)
23 matriz_confusao <-confusionMatrix(predicao, dados.teste$HadHeartAttack)
24
25 ### REALIZAR OS CALCULOS DAS METRICAS DE DESEMPENHO
26 acuracia        <- matriz_confusao$overall["Accuracy"]
27 precisao        <- matriz_confusao$byClass["Pos␣Pred␣Value"]
28 sensibilidade   <- matriz_confusao$byClass["Sensitivity"]
29 especificidade  <- matriz_confusao$byClass["Specificity"]
30 F1_score        <- 2 * (precisao * sensibilidade) /
31                        (precisao + sensibilidade)
32
33 ### VERIFICAR IMPORTANCIA DOS ATRIBUTOS E PLOTAGEM DO RESULTADO
34 importance(rf)
35 varImpPlot(rf, main = 'Importancia␣Atributos')
```

Fonte: O Autor (2024).

Para o terceiro *dataset*, *Body Fat Prediction Dataset*, tiveram algumas pequenas alterações na parte de parametrização, que valem um detalhamento:

1. Foi aplicada a função $tuneRF()$ para descobrir qual seria o valor ideal para o parâmetro *mtry*. Nela foram passados os parâmetros conforme abaixo:
    - $dados.treino.x$: conjunto de dados de treino, passados sem os dados referentes ao atributo *target*;
    - $dados.treino.y$: conjunto de dados de treino, passados somente os dados referente ao atributo *target*;
    - $stepFactor = 1.5$: valor que refere-se ao quando varia o valor de *mtry* para cada um dos testes;
    - $improve = 0.01$: valor que refere-se ao número mínimo de melhora necessário para considerar o valor de *mtry*. Por exemplo, setado como 0.01, é necessário que a cada teste do parâmetro, tenha pelo menos 0.01 de melhora na execução;
    - $ntreeTry = 50$: valor padrão do parâmetro. Refere-se ao número de árvores que foram tentadas;

- $trace = TRUE$: indicado como ***TRUE*** para que fosse possível verificar o status da execução e quais os valores tentados;

2. Foi aplicado o algoritmo do ***Random Forest*** (nomeado pela função $randomForest()$) propriamente dito. Nele foram passados os parâmetros essenciais para sua execução, variando apenas os parâmetros conforme supracitado. Abaixo segue, de maneira mais genérica, a descrição dos parâmetros passados para a função:

   - $x = dados.treino.x$: representa a atribuição de todos os dados e atributos utilizados para treino na variável x;
   - $y = dados.treino.y$: representa a atribuição de todas as linhas do atributo *target* na variável y;
   - $importance = TRUE$: definido como ***TRUE*** para que seja possível verificar a importância de cada um dos atributos do *dataset*, baseado em dois critérios principais, segundo a documentação oficial *randomForest* da linguagem R [5].
     - *MeanDecreaseGini*: é responsável por indicar o quanto a acurácia é reduzida quando essa variável é reduzida. Valores maiores indicam que a variável é mais importante para a precisão do modelo.
     - *MeanDecreaseAccuracy*: é responsável por indicar a importância da variável na pureza de cada um dos nós de cada árvore. Assim como o *MeanDecreaseGini*, quanto maior o valor, significa que a variável contribui mais na divisão das árvores.
   - $ntree = ntree$: representa a atribuição do número máximo de árvores de decisão que seriam geradas;
   - $mtry = mtry$: representa a atribuição do número de atributos selecionados de maneira aleatória para realizar a criação de cada árvore de decisão;
   - $nodesize = 5$: representa o número máximo que os nodos de cada árvore pode ter. Foi setado como 1 pois é um problema de classificação;
   - $keep.forest = TRUE$: definido como ***TRUE*** para que a floresta seja retida no objeto de saída.

3. Foi feita a execução da função $predict()$ e atribuído o resultado na variável *predicao*, ficando da seguinte forma: $predicao = predict(rf, newdata = dados.teste.x)$. Os parâmetros passados foram o resultado gerado pela execução do algoritmo de RF e o conjunto de dados de teste.

4. Foram executados, conforme apresentado abaixo, as atribuições e cálculos das variáveis que foram utilizadas para medição da análise de desempenho da aplicação do algoritmo:

[5] https://cran.r-project.org/

- $mae < -mean$: representa a atribuição do valor da precisão gerado após execução da função $confusionMatriz()$ no passo anterior;
- $mse < -mean((predicao - dados.teste.y)\hat{}2)$: representa a atribuição do valor da sensibilidade gerado após execução da função $confusionMatriz()$ no passo anterior;
- $rmse < -sqrt(mse)$: representa a aplicação da função $sqrt()$ e atribuição do resultado da mesma na variável *rmse*. Como parâmetros passados, foi o valor gerado na variável anterior *mse*;
- $r2 < -cor(predicao, dados.teste.y)^2$: representa a aplicação da função $cor()^2$ e atribuição do resultado da mesma na variável *r2*. Como parâmetros passados, foi a predição gerada anteriormente e o conjunto de dados de teste, passando somente o atributo *target*.

5. Foram executadas as funções conforme abaixo para verificar a importância dos atributos:
   - $importance(rf)$ e $varImp(rf)$: funções executadas para verificar a importância dos atributos;
   - $varImpPlot(rf, main = 'Importancia\ Atributos')$: função executada para realizar a plotagem das variáveis em relação a importância;

Ademais, o Algoritmo 10 demonstra como foi estruturado tudo o que foi previamente apresentado.

Algoritmo 10 – Código da Aplicação do RF no *Body Fat Prediction Dataset*

```
1  ### FUNCAO PARA VERIFICAR O NUMERO IDEAL DE MTRY
2  melhor_mtry <- tuneRF(dados.treino.x,
3                        dados.treino.y,
4                        stepFactor = 1.5,
5                        improve = 0.01,
6                        ntreeTry = 50,
7                        trace = TRUE)
8
9  ### ATRIBUICAO DO MELHOR VALOR DE MTRY PARA TESTE
10 mtry <- melhor_mtry[melhor_mtry[, 2] == min(melhor_mtry[, 2]), 1]
11
12 ### APLICACAO DO RANDOMFOREST
13 rf <- randomForest(x = dados.treino.x,
14                     y = dados.treino.y,
15                     importance = TRUE,
16                     ntree = ntree,
17                     mtry = mtry,
18                     nodesize = 5,
19                     keep.forest = TRUE)
```

```
20
21 ### REALIZAR A PREDICAO
22 predicao <- predict(rf, newdata = dados.teste.x)
23
24 ### REALIZAR OS CALCULOS DAS METRICAS DE DESEMPENHO
25 mae  <- mean(abs(predicao - dados.teste.y))
26 mse  <- mean((predicao - dados.teste.y)^2)
27 rmse <- sqrt(mse)
28 r2   <- cor(predicao, dados.teste.y)^2
29
30 ### VERIFICAR IMPORTANCIA DOS ATRIBUTOS E PLOTAGEM DO RESULTADO
31 importance(rf)
32 varImp(rf)
33 varImpPlot(rf, main = 'Importancia␣Variaveis')
```

Fonte: O Autor (2024).

5 RESULTADOS E DISCUSSÕES

Após aplicação do algoritmo de *Random Forest* nos três *datasets* selecionados, conforme toda codificação que foi explanada dentro do Capítulo 4, foram obtidos resultados para cada um dos conjuntos de dados, que serão explanados no decorrer deste Capítulo 5. Além disso, por mais que os algoritmos executados tivessem funções iguais, os resultados apresentados são exclusivos para cada *dataset*, portanto a análise será feita um a um.

Para o *Diabetes Prediction Dataset*, que foi selecionado e utilizado para avaliar o RF para um problema de classificação, cujo objetivo foi a previsão de diabetes a partir dos dados do *dataset*, teve-se que, após a execução da função $tuneRF()$, a qual retornou o valor ideal para o parâmetro *mtry* como 2. A partir disso, foram realizados diversos testes com alteração do outro principal parâmetro da função $randomForest()$, o parâmetro *ntree*. A Tabela 14 mostra os resultados obtidos para as medidas de desempenho da acurácia, especificidade, precisão, sensibilidade e *F1_score* ao alterar o parâmetro *ntree*.

Tabela 10 – Resultados *Diabetes Prediction Dataset*

mtry	***ntree***	**acuracia**	**especificidade**	*F1_score*	**precisao**	**sensibilidade**
2	100	0.971	0.665	0.984	0.97	1
2	150	0.971	0.665	0.984	0.97	1
2	190	0.971	0.665	0.985	0.97	1
2	200	0.971	0.666	0.985	0.97	1
2	250	0.971	0.666	0.985	0.97	1
2	400	0.971	0.666	0.985	0.97	1
2	500	0.971	0.666	0.985	0.97	1
2	800	0.971	0.666	0.985	0.97	1
2	1000	0.971	0.665	0.985	0.97	1
2	2000	0.971	0.666	0.985	0.97	1
2	3000	0.971	0.666	0.985	0.97	1

Fonte: O Autor (2024).

Como foi possível perceber, a partir das atribuições de $ntree = 200$, os resultados das métricas de desempenho começaram a estabilizar-se, o que mostrou que não importava se fossem alterados os parâmetros, o resultado já teria atingido o seu máximo. Ao total, foi alcançada uma acurácia total de pouco mais de 97%.

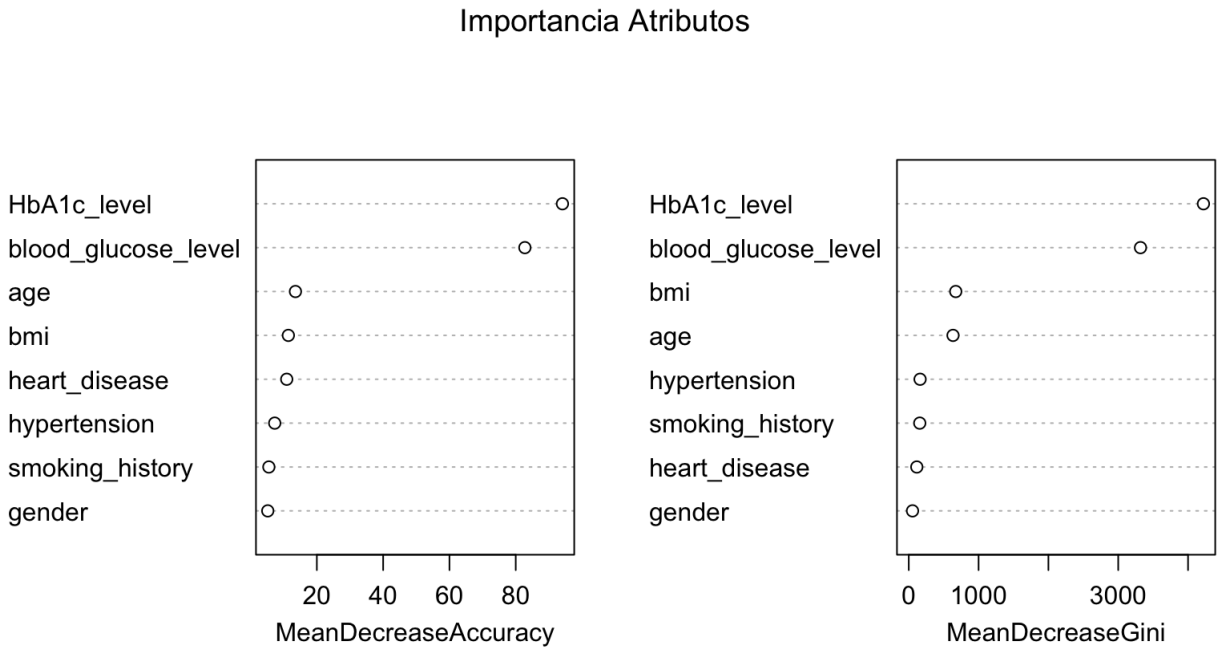
Também, Figura 22 e Figura 23 nos mostram a relação das importâncias dos atributos dentro da realização da classificação, onde o atributo *HbA1c_level* mostrou-se o que mais influência na classificação, seguido pelo atributo *blood_glucose_level* e *age* como os 3 principais atributos para o resultado "1"de variável, ou seja, a presença de diabetes. Ademais, para o resultado "0", ou seja, ausência de diabetes, temos além dos atributos *HbA1c_level* e *blood_glucose_level*, o atributo *heart_desease* visto que, quanto maior o valor que aparecem, significa que mais importam dentro do conjunto de dados.

Figura 22 – Importância Atributos pela função $importance()$

	0	1	MeanDecreaseAccuracy	MeanDecreaseGini
gender	5.817139	-0.2904721	5.191863	53.9396
age	5.133897	13.2035070	13.543285	633.9038
hypertension	5.264902	4.7991788	7.291230	162.2884
heart_disease	7.819848	8.6883015	10.912019	115.1012
smoking_history	2.988989	5.1729514	5.523577	157.4380
bmi	7.154324	8.6629505	11.404266	672.8659
HbA1c_level	92.055739	72.3342988	94.085328	4220.5343
blood_glucose_level	72.586312	72.8776978	82.751511	3318.8642

Fonte: O Autor (2024).

Figura 23 – Plot Importância Atributos pela função $varImpPlot()$



Fonte: O Autor (2024).

A Tabela 11 mostra a matriz de confusão do dataset, conforme resultado da execução da função $confusionMatriz()$.

Tabela 11 – Matriz de Confusão *Dataset* 1

	Reference	
Prediction	**0**	**1**
0	22866	692
1	9	1433

Fonte: O Autor (2024).

Conforme foi possível perceber visualizando a matriz de confusão, dentre todos os dados do conjunto de testes, o algoritmo previu de maneira correta 24.299 amostras, sendo 22.866 da classificação 0, ou seja, sem diabetes, e 1.433 da classificação 1, ou seja, com diabetes. As 701 amostras restantes foram previstas de maneira incorreta. A razão entre essas duas quantidades comprova a acurácia de pouco mais de 97%.

Para o *Indicators of Heart Disease Dataset*, que foi selecionado e utilizado para avaliar o algoritmo de RF para um problema de classificação que, neste caso, foi a previsão do paciente ter uma doença cardíaca a partir de um conjunto de atributos e evidências (conjunto de dados), teve-se que após a execução da função $tuneRF()$, a qual retornou o valor ideal para o parâmetro *mtry* como 4. A partir disso, foram realizados diversos testes com alteração do outro principal parâmetro da função $randomForest()$, o parâmetro *ntree*. A Tabela 12 mostra os resultados obtidos para as medidas de desempenho da acurácia, especificidade, precisão, sensibilidade e *F1_score* ao alterar o parâmetro *ntree*.

Tabela 12 – Resultados *Indicators of Heart Disease Dataset*

mtry	***ntree***	***acuracia***	***especificidade***	***F1_score***	***precisao***	***sensibilidade***
4	100	0.945	0.994	0.226	0.579	0.14
4	200	0.946	0.991	0.307	0.573	0.21
4	300	0.946	0.991	0.307	0.569	0.21
4	400	0.946	0.991	0.308	0.576	0.21
4	500	0.946	0.991	0.308	0.574	0.21
4	600	0.947	0.991	0.309	0.583	0.21
4	800	0.946	0.991	0.308	0.577	0.21
4	900	0.945	0.994	0.226	0.579	0.14

Fonte: O Autor (2024).

A partir da análise dos resultados obtidos pelas métricas de desempenho, obteu-se que, pelo conjunto dos valores, o melhor valor de *ntree* encontrado foi $ntree = 600$. Ao total, foi alcançada uma acurácia total de quase 95%.

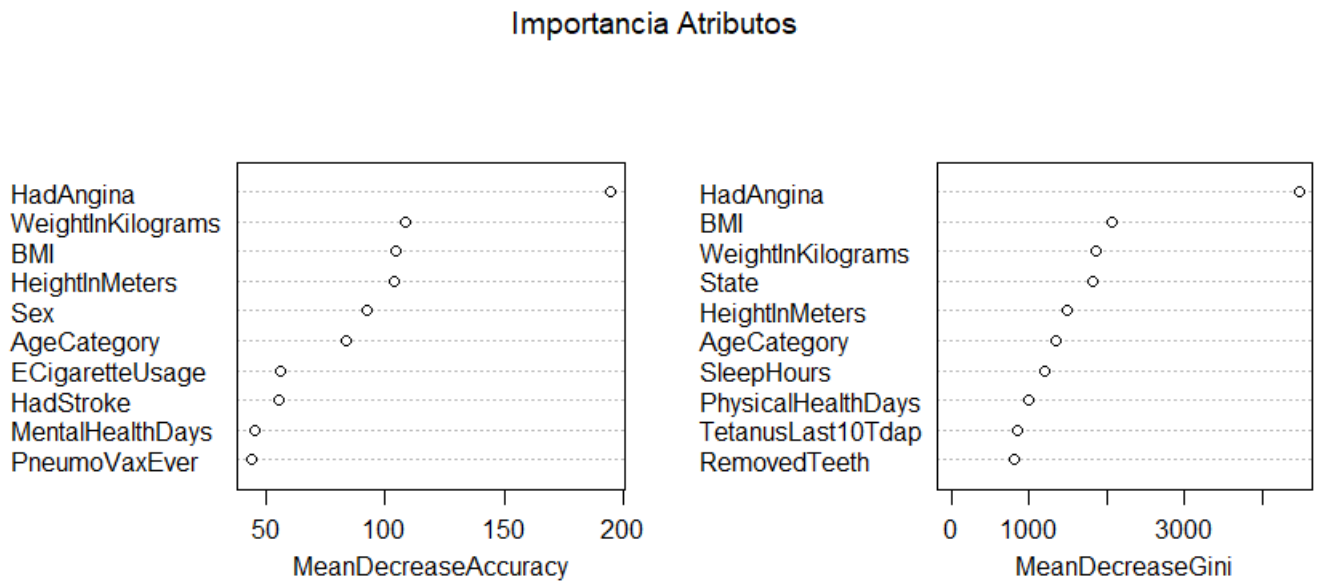
Também, Figura 24 e Figura 25 nos mostram a relação das importâncias dos atributos dentro da realização da classificação, onde o atributo *HadAngina* mostrou-se o que mais influência na classificação, seguido pelo atributo *WeightInKilograms*, *BMI* e *HeightInMeters* como os 4 principais atributos, visto que, quanto maior o valor que aparecem, significa que mais importam dentro do conjunto de dados.

Figura 24 – Importância Atributos

	Yes	No	MeanDecreaseAccuracy	MeanDecreaseGini
State	2.804511	0.2395972	1.4115000	1816.9697
Sex	44.466828	80.0506926	92.2418325	346.2961
GeneralHealth	33.588696	12.3863269	29.9745643	788.6831
PhysicalHealthDays	17.408967	31.9846041	38.5101715	993.1262
MentalHealthDays	0.851062	47.0539173	45.1592219	751.4952
LastCheckupTime	5.561579	23.3082582	25.3386331	270.1155
PhysicalActivities	7.860392	13.2281640	15.5776374	347.0999
SleepHours	12.202416	2.3387024	7.2409858	1189.8731
RemovedTeeth	35.222289	-7.0656622	9.7889211	816.1907
HadAngina	233.940911	117.1542144	194.4568224	4465.5039
HadStroke	88.303761	8.5714251	55.4522638	629.2520
HadAsthma	-5.226631	14.3090765	10.9326626	298.5868
HadSkinCancer	-6.493113	22.2461037	18.6410045	308.7293
HadCOPD	20.571616	-2.8929504	6.6052345	348.9597
HadDepressiveDisorder	-3.103927	28.6849593	26.1220008	315.1046
HadKidneyDisease	14.031392	-4.1954879	2.8886230	287.3645
HadArthritis	-6.711346	35.5930013	30.6553648	380.5571
HadDiabetes	42.528294	3.2386133	22.6362654	488.3691
DeafOrHardOfHearing	7.979942	2.1047018	5.6093636	317.8913
BlindOrVisionDifficulty	13.329586	-2.8250207	3.2484616	243.7336
DifficultyConcentrating	-7.441766	44.2136993	40.6297412	287.2351
DifficultyWalking	15.154596	31.6191766	35.3921470	404.5058
DifficultyDressingBathing	-11.063345	33.6723955	31.4668711	177.5992
DifficultyErrands	-0.263958	29.9247061	29.6347321	236.0600
SmokerStatus	32.100019	21.9485011	34.0392144	623.3924
ECigaretteUsage	-13.966563	59.9063143	56.1621483	427.7428
ChestScan	46.105774	-24.3324765	-0.5786919	500.4227
RaceEthnicityCategory	-2.618908	26.9557999	24.2673509	507.9192
AgeCategory	40.021504	65.8714084	83.7880664	1340.4071
HeightInMeters	-11.391122	103.0056817	103.8668910	1494.9470
WeightInKilograms	-36.508920	110.6161897	108.5963296	1849.5368
BMI	-39.517926	107.8577093	104.7545331	2057.8297
AlcoholDrinkers	7.710438	24.0619685	25.8349480	403.4138
HIVTesting	-10.319391	40.4586314	36.5240808	474.9367
FluVaxLast12	-19.239106	45.9022778	41.1969064	408.9682
PneumoVaxEver	-5.541187	45.0130179	43.7053585	434.5452
TetanusLast10Tdap	2.257407	21.0245143	21.0738268	857.2060
HighRiskLastYear	-14.012353	41.0287478	39.7080173	154.0466
CovidPos	-7.842467	31.9085792	30.3779737	444.4691

Fonte: O Autor (2024).

Figura 25 – Plot Importância Atributos



Fonte: O Autor (2024).

A Tabela 13 mostra a matriz de confusão do dataset, conforme resultado da execução da função $confusionMatriz()$.

Tabela 13 – Matriz de Confusão *Dataset* 2

	Reference	
Prediction	**Yes**	**No**
Yes	692	422
No	4658	88471

Fonte: O Autor (2024).

Conforme foi possível perceber visualizando a matriz de confusão, dentre todos os dados do conjunto de testes, o algoritmo previu de maneira correta 89.163 amostras, sendo 692 da classificação *Yes*, ou seja, tiveram ataques cardíacos, e 88.471 da classificação *No*, ou seja, não tiveram ataques cardíacos. As 5.080 amostras restantes foram previstas de maneira incorreta. A razão entre essas duas quantidades comprova a acurácia de mais de 94%.

Para o *Body Fat Prediction Dataset*, que foi selecionado e utilizado para avaliar o algoritmo de RF para um problema de regressão, no caso a previsão do valor de gordura corporal, teve-se que após a execução da função $tuneRF()$, a qual retornou o valor ideal para o parâmetro *mtry* como 13. A partir disso, foram realizados diversos testes com alteração do outro principal parâmetro da função $randomForest()$, o parâmetro *ntree*. A Tabela 14 mostra os resultados obtidos para as medidas de desempenho da acurácia, especificidade, precisão, sensibilidade e *F1_score* ao alterar o parâmetro *ntree*.

Tabela 14 – Resultados *Body Fat Prediction Dataset*

mtry	***ntree***	***mae***	***mse***	***rmse***	***r2***
13	100	0.2033	0.096	0.3098	0.9983
13	200	0.1695	0.0521	0.2283	0.9991
13	300	0.1837	0.0635	0.2521	0.9989
13	400	0.18	0.065	0.255	0.9989
13	500	0.1859	0.0652	0.2554	0.9989
13	600	0.1863	0.0634	0.2518	0.999
13	700	0.1708	0.0554	0.2355	0.9991
13	800	0.1773	0.062	0.249	0.9989
13	900	0.1811	0.0634	0.2152	0.999
13	1000	0.1785	0.0607	0.2465	0.999
13	1500	0.18	0.0595	0.2439	0.999
13	2000	0.1808	0.058	0.2408	0.9991
13	3000	0.1819	0.0604	0.2457	0.999
13	4000	0.1774	0.0576	0.2401	0.9991
13	5000	0.175	0.056	0.2367	0.9991
13	6000	0.176	0.0552	0.235	0.9991
13	7000	0.1771	0.0564	0.2375	0.9991
13	8000	0.1738	0.0556	0.2358	0.9991
13	9000	0.1755	0.0564	0.2376	0.9991
13	10000	0.1799	0.059	0.2428	0.999
13	15000	0.1755	0.0559	0.2364	0.9991
13	20000	0.1759	0.0557	0.236	0.9991

Fonte: O Autor (2024).

Baseado nas métricas adotas para medir o desempenho dessa aplicação, foi feita uma análise do conjunto de todas as respostas, ordenando-as conforme melhor opção para cada uma das variáveis, chega-se a conclusão que para $ntree = 200$, teve-se o melhor conjunto de valores. No entanto se tivesse sido adotada somente uma medida de desempenho, como o *RMSE*, por exemplo, teríamos que o melhor valor para *ntree* seria de 900. Isso comprova que a deve-se sempre estabelecer mais de uma métrica, para que o teste não seja viciado e limitado.

Também, a Figura 26, Figura 27 e Figura 28 nos mostram a relação das importâncias dos atributos dentro da realização da regressão, onde o atributo *Density* se mostrou de maior relevância em relação a todos os outros, tendo uma importância de mais de 70 de *Overall*, enquanto o segundo atributo mais importante foi o *Knee*, com pouco mais de 4.5 de *Overall*.

Figura 26 – Importância Atributos pela função $importance()$

	%IncMSE	IncNodePurity
Density	71.9954446	13190.771014
Age	0.6380342	18.576628
Weight	0.4859614	14.469511
Height	-0.3613954	49.332541
Neck	2.5899422	15.683488
Chest	-0.7363051	38.463665
Abdomen	4.5990059	614.297184
Hip	2.4794559	16.680528
Thigh	0.8646997	8.792436
Knee	4.5993416	43.207159
Ankle	0.6928525	5.124222
Biceps	0.2397151	10.617225
Forearm	0.9904876	8.551410
Wrist	2.8665184	19.191811

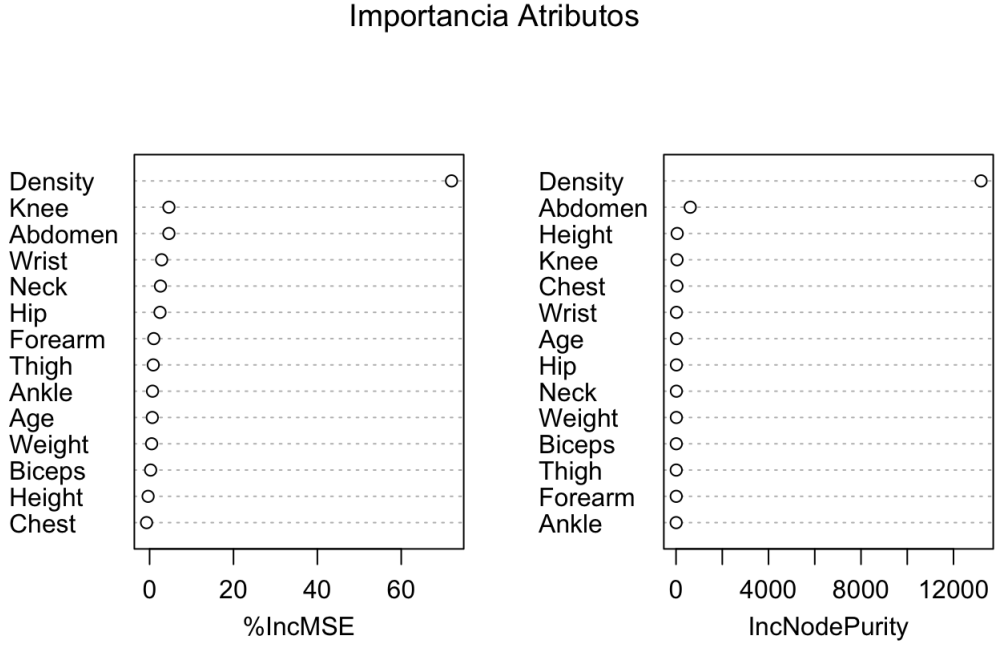
Fonte: O Autor (2024).

Figura 27 – Importância Atributos pela função $varImp()$

	Overall
Density	71.9954446
Age	0.6380342
Weight	0.4859614
Height	-0.3613954
Neck	2.5899422
Chest	-0.7363051
Abdomen	4.5990059
Hip	2.4794559
Thigh	0.8646997
Knee	4.5993416
Ankle	0.6928525
Biceps	0.2397151
Forearm	0.9904876
Wrist	2.8665184

Fonte: O Autor (2024).

Figura 28 – Plot Importância Atributos pela função $varImpPlot()$



Fonte: O Autor (2024).

5.1 COMPARAÇÃO DOS RESULTADOS E CONCLUSÕES

Realizando a comparação entre os resultados dos três *datasets*, é possível perceber que para todas as aplicações, o algoritmo se mostra eficaz. Dentre a comparação entre todos os três, percebe-se que, pelo *dataset 1* estar mais adequado e sem inconsistências, confirma o motivo de ter tido uma acurácia melhor e um melhor resultado na análise.

No entanto, isso não descarta a comprovação de que o algoritmo de RF funciona muito bem para conjuntos de dados que apresentem inconsistências, tratando-se de classificação, visto que o *dataset 2* teve uma acurácia de mais de 94%, comprovando a eficiência do algoritmo. No entanto vale ressaltar que, tirando a acurácia, o segundo conjunto de dados teve um resultado não tão positivo para as métricas de *F1_score*, onde o resultado foi abaixo dos 35%, sensibilidade com menos de 22% e precisão com aproximadamente 60%.

Por fim, tratando-se do *dataset 3*, cujo foi utilizado para regressão, também teve um resultado considerado aceitável, porém diversos fatores influenciam nesse resultado, dentre eles a quantidade pequena de dados. Para conjuntos de dados com mais amostras, o algoritmo tende a ter um melhor resultado.

De modo geral, os resultados obtidos foram satisfatórios, visto que foi possível perceber que a aplicação do algortimo se faz eficiente para os mais diversos tipos de conjuntos de dados e tipos de problemas, seja classificação ou regressão, e tendo inconsitências nos dados ou não.

Uma observação importante a ser tratada para encerramento, a limitação de hardware

de onde foram executados os algoritmos também se fez muito importante, visto que, poderiam ter sido realizados mais testes alterando outros parâmetros e valores, que, por limitações de hardware e também de tempo de execução, não foram suportados. O que mostra que para grandes *datasets*, não importa somente a codificação e parametrização do algoritmo, também faz-se necessário que tenha um recurso operacional eficiente e que suporte uma grande quantidade de transações e dados.

6 CONSIDERAÇÕES FINAIS

Neste trabalho, foi realizada uma pesquisa a qual explorou a fundamentação teórica sobre aprendizado de máquina, suas subdivisões, alguns de seus principais algoritmos e aplicações de cada um deles, havendo uma ênfase no algoritmo *Random Forest* (RF), o qual foi aplicado nos conjuntos de dados citados na Seção 4.5, que são a base do TCC II.

Com a pesquisa, ficou evidente que a escolha certa do algoritmo para cada tipo de situação e problema é uma das partes mais importantes no aprendizado de máquina, juntamente com a parametrização do algoritmo. No entanto, vale também ressaltar que a combinação entre algoritmos e a utilização de métricas para avaliação dos mesmos também é algo de extrema importância e pode ser muito útil para melhor desempenho e solução dos problemas.

Ademais, conforme já citado no Capítulo 4, foi realizada a aplicação do algoritmo de RF nos *datasets* previamente selecionados, onde foram feitos diversos testes e avaliações, visando a melhor parametrização do algoritmo em cada um dos conjuntos de dados escolhidos. Todo o desenvolvimento do TCC II foi feito na linguagem R, com a utilização da interface do ambiente de desenvolvimento do *RStudio*.

Por resultados deste trabalho, foi possível primeiramente perceber que o AM é algo que se faz cada vez mais necessário em meio ao mundo técnologico em que estamos vivendo e, devido a quantidade massiva de dados que existem, pode-se fazer análises muito mais rapidamente aplicando um bom algoritmo, tendo realizado uma boa parametrização. Ademais, também foi possível confirmar a eficiência do algortimo de RF tanto para problemas de classificação quanto para problemas de regressão, visto que em ambos os problemas os resultados foram válidos e satisfatórios. Também vale ressaltar que, devido a limitações de hardware, possivelmente com mais testes poderia ter sido obtidos resultados melhores, que vale para possíveis futuros trabalhos.

6.1 IDEIAS PARA FUTUROS TRABALHOS, PESQUISAS E PROJETOS

Inicialmente, recomenda-se que haja uma continuação deste trabalho, porém com uma limitação computacional menor, para que possam ser executados mais testes e mais possíbilidades de parametrizações para o algoritmo, buscando assim atingir melhores métricas de desempenho.

Também, vale ressaltar que, nesta monografia foram utilizados dados relacionados a área da saúde, porém a aplicação do algoritmo é multi-área, portanto, realizar uma pesquisa e aplicação do mesmo para outra área do conhecimento, com diferentes valores, atributos e resultados é algo interessante para comprovação da eficiência do algoritmo.

Por fim, um adendo a ser feito também dentro da área da saúde é uma demanda alta que tenho percebido durante meu tempo de trabalho na maior cooperativa médica da serra, Unimed Serra Gaúcha, é uma demanda de que atualmente possui um grande banco de dados que armazena resultados de exames, sejam eles de imagem ou exames mais simples, resultados de diagnósticos e diversos laudos médicos que, hoje em dia, não estão sendo utilizados e que poderiam, se trabalhados adequadamente, ser analisados, identificar padrões e auxiliar os pacientes a evitarem doenças e conseguir um diagnóstico mais rápido sobre possíveis problemas. Dentro disso, seria algo que o algoritmo de RF teria uma ótima aplicação, visto que lida muito bem com conjuntos de dados grandes e as predições e classificações se mostraram muito eficazes.

REFERÊNCIAS

ALBA, E. *et al.* Comparação entre algoritmos de aprendizado de máquina para a identificação de floresta tropical sazonalmente seca. **Anuário do Instituto de Geociências**, Universidade Federal do Rio de Janeiro, v. 45, p. 1–10, 2022.

ALLAIRE, J. Rstudio: integrated development environment for r. **Boston, MA**, v. 770, n. 394, p. 165–171, 2012.

AMORIM, M. J.; BARONE, D.; MANSUR, A. U. Técnicas de aprendizado de máquina aplicadas na previsao de evasao acadêmica. In: **Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)**. [S.l.: s.n.], 2008. v. 1, n. 1, p. 666–674.

BATISTA, G. E. d. A. P. *et al.* **Pré-processamento de dados em aprendizado de máquina supervisionado**. Tese (Doutorado) — Universidade de São Paulo, 2003.

BREIMAN, L. Random forests. **Machine learning**, Springer, v. 45, p. 5–32, 2001.

CANO, P. L. G.; JUNIOR, J. M. Aplicação de aprendizado de máquina com dados de sensoriamento remoto para o mapeamento de florestas urbanas. **Revista Geociências-UNG-Ser**, v. 20, n. 2, p. 16–27, 2021.

CERRI, R.; FERREIRA, A. C. P. de L. *et al.* Aprendizado de máquina: breve introdução e aplicações. **Cadernos de Ciência & Tecnologia**, v. 34, n. 3, p. 297–313, 2019.

COSTA, L. d. F. Máquinas tomam decisões: Reconhecimento de padrões e mineração de dados. **Ciência Hoje**, v. 30, n. 176, p. 22–29, 2001.

DOMINGUES, M. L. C. S. Mineração de dados utilizando aprendizado não-supervisionado: um estudo de caso para bancos de dados da saúde. 2003.

FILHO, C. H. P. **Técnicas de aprendizado não supervisionado baseadas no algoritmo da caminhada do turista**. Tese (Doutorado) — Universidade de São Paulo, 2017.

FONTANA, E. Introdução aos algoritmos de aprendizagem supervisionada. **Departamento de Engenharia Química, Universidade Federal do Paraná**, 2020.

FREIRE, S. M. **Introdução ao R**. [S.l.: s.n.], 2021. v. 1.

FUJIKAWA, C. S. Reconhecimento facial utilizando descritores de textura e aprendizado não supervisionado. **Monografia (Bacharel em Ciências da Computação), UNESP (Universidade Estadual Paulista" Júlio de Mesquita Filho"), Rio Claro, Brazil**, 2016.

GAUSS, C. F. **Theoria motus corporum coelestium in sectionibus conicis solem ambientium**. [S.l.]: FA Perthes, 1877. v. 7.

GISLASON, P. O.; BENEDIKTSSON, J. A.; SVEINSSON, J. R. Random forests for land cover classification. **Pattern recognition letters**, Elsevier, v. 27, n. 4, p. 294–300, 2006.

HENNIG, C. *et al.* **Handbook of cluster analysis**. [S.l.]: CRC press, 2015.

IZBICKI, R.; SANTOS, T. M. dos. **Aprendizado de máquina: uma abordagem estatística**. [S.l.]: Rafael Izbicki, 2020.

JIAWEI, H.; MICHELINE, K. **Data mining: concepts and techniques**. [S.l.]: Morgan kaufmann, 2006.

LEGENDRE, A. M. **Nouvelles méthodes pour la détermination des orbites des comètes: avec un supplément contenant divers perfectionnemens de ces méthodes et leur application aux deux comètes de 1805**. [S.l.]: Courcier, 1806.

LEITE, D. R. A.; MORAES, R. M. de; LOPES, L. W. Método de aprendizagem de máquina para classificação da intensidade do desvio vocal utilizando random forest. **Journal of Health Informatics**, v. 12, 2020.

LIMA, T. P. F. *et al.* Previsão de óbito e importância de características clínicas em idosos com covid-19 utilizando o algoritmo random forest. **Revista Brasileira de Saúde Materno Infantil**, SciELO Brasil, v. 21, p. 445–451, 2021.

LOPES, C. H. P. Classificação de registros em banco de dados por evolução de regras de associação utilizando algoritmos genéticos. **Dissertacão de Mestrado, Departamento de Engenharia Eletrica, PUC-Rio**, 1999.

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

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. **The bulletin of mathematical biophysics**, Springer, v. 5, p. 115–133, 1943.

MILLIGAN, G. W. Clustering validation: results and implications for applied analyses. In: **Clustering and classification**. [S.l.]: World Scientific, 1996. p. 341–375.

MITCHELL, T. Machine learning. **McGraw Hill**, 1997.

MOMIN, B. F.; PARDESHI, M. S. High dimension cnf to dnf conversion using grid computing. In: IEEE. **2013 15th International Conference on Advanced Computing Technologies (ICACT)**. [S.l.], 2013.

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

OLSON, D. L.; DELEN, D. **Advanced data mining techniques**. [S.l.]: Springer Science & Business Media, 2008.

OSHIRO, T. M. **Uma abordagem para a construção de uma única árvore a partir de uma Random Forest para classificação de bases de expressão gênica**. Tese (Doutorado) — Universidade de São Paulo, 2013.

R, W. **História do R — Wiki R,**. 2017. Acesso em 22-junho-2024. Disponível em: <https://www.ufrgs.br/wiki-r/index.php?title=Hist%C3%B3ria_do_R&oldid=184>.

RADY, H. Face recognition using principle component analysis with different distance classifiers. **IJCSNS International Journal of Computer Science and Network Security**, v. 11, n. 10, p. 134–144, 2011.

ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. **Psychological review**, American Psychological Association, v. 65, n. 6, p. 386, 1958.

SILVA, R. Inteligência artificial. Amigos da Enciclopédia, 2013.

TEODORO, L. de A.; KAPPEL, M. A. A. Aplicação de técnicas de aprendizado de máquina para predição de risco de evasão escolar em instituições públicas de ensino superior no brasil. **Revista Brasileira de Informática na Educação**, v. 28, p. 838–863, 2020.

VLAHOU, A. *et al.* Diagnosis of ovarian cancer using decision tree classification of mass spectral data. **Journal of Biomedicine and Biotechnology**, Hindawi Publishing Corporation PO BOX 3079, CAYAHOGA FALLS, OH 44233 USA, v. 2003, n. 5, p. 308–314, 2003.

WEISS, S. M.; INDURKHYA, N. **Predictive data mining: a practical guide**. [S.l.]: Morgan Kaufmann, 1998.

WITTEN, I. H.; FRANK, E. **Data Mining: Practical machine learning tools and techniques**. [S.l.]: Diane Cerra, 2005. v. 2.

XU, D.; TIAN, Y. A comprehensive survey of clustering algorithms. **Annals of data science**, Springer, v. 2, p. 165–193, 2015.