UNIVERSIDADE DE SÃO PAULO

ESCOLA DE ENGENHARIA DE LORENA

MURILO AFONSO ROBIATI BIGOTO

Avaliação de modelos de *machine learning* para predição da temperatura crítica de supercondutores

Lorena
2020

MURILO AFONSO ROBIATI BIGOTO

Avaliação de modelos de *machine learning* para predição da temperatura crítica de supercondutores

Trabalho de Graduação apresentado à Escola de Engenharia de Lorena da Universidade de São Paulo como requisito parcial para conclusão da Graduação do curso de Engenharia Física.

Orientador: Prof. Dr. Luiz Tadeu Fernandes Eleno

Lorena
2020

Ficha catalográfica elaborada pelo Sistema Automatizado
da Escola de Engenharia de Lorena,
com os dados fornecidos pelo(a) autor(a)

Bigoto, Murilo Afonso Robiati
Avaliação de modelos de machine learning para predição da temperatura crítica de supercondutores / Murilo Afonso Robiati Bigoto; orientador Luiz Tadeu Fernandes Eleno. - Lorena, 2020.
78 p.

Monografia apresentada como requisito parcial para a conclusão de Graduação do Curso de Engenharia Física - Escola de Engenharia de Lorena da Universidade de São Paulo. 2020

1. Machine learning. 2. Supercondutividade. 3. Temperatura crítica. 4. Inteligência artificial. I. Título. II. Eleno, Luiz Tadeu Fernandes, orient.

Dedico este trabalho aos meus pais, que sempre me transmitiram amor e apoio.

Agradecimentos

Agradeço inestimavelmente aos meus pais, Radamez e Sandra, por todos os esforços e incentivos direcionados à minha formação acadêmica e profissional. Agradeço também a eles por propiciarem a minha experiência de existir e a formação do meu caráter. Sou muito grato à minha mãe por me ensinar o que é o verdadeiro amor e por ser um exemplo de força e bravura. E ao meu pai, sou extremamente grato por ser meu exemplo de virtudes e honestidade.

Sem o apoio e incentivo dos meus pais e avós, eu certamente não teria saído atrás dos meus sonhos e abandonado a pacata Santana da Ponte Pensa, que com seus mil e quatrocentos habitantes, vive em tempos arcaicos. Assim, agradeço a toda minha família, pois sem eles eu não teria forças para seguir em frente e batalhar pelo que eu acredito.

À minha avó Sebastiana Diniz, sou eternamente grato pela companhia, suporte, demonstrações de amor e amizade. Ela me ensinou o verdadeiro significado de amor ao próximo e companheirismo. Ao meu avô José Bigoto, tenho uma dívida eterna por ter me mostrado que as coisas simples são as mais importantes, que a vida é uma oportunidade única, sendo indispensável ser quem realmente somos.

Ao meu avô Alvides Robiati, agradeço grandemente por ter me mostrado que envelhecer não passa de uma perspectiva física e que vitalidade é algo intrínseco ao ser que habita uma carcaça de carne. À minha avó Anita Neves, deixo o agradecimento por me mostrar um amor puro e belo. É dela que eu tenho o maior exemplo de sensatez, compaixão e verdade.

Agradeço ao meu pequeno irmão Miguel, por me possibilitar experimentar o sentimento de ser uma pessoa relevante e me fazer querer que o mundo seja um lugar melhor. Sou muito grato por ele se mostrar ser uma criança cética e admirar com muita pureza a natureza.

Não poderia deixar de agradecer a todos os professores e cientistas comprometidos com a verdadeira ciência, pois eles são quase sempre os responsáveis por mostrar que a ciência é a ferramenta mais disruptiva para a evolução da nossa espécie. Além disso, eles são um dos principais mediadores do conhecimento científico – o agente mais importante no combate ao fanatismo, extremismo, incoerência e ignorância.

Nessa perspectiva, agradeço ao Prof. Dr. Luiz Tadeu Fernandes Eleno, por ter aceito meu pedido de orientação e ser um profissional admirável e compromissado com a transmissão do conhecimento científico.

Agradeço muito aos meus amigos, que mesmo antes do meu ingresso à universidade, estiveram ao meu lado e me incentivaram a buscar meus sonhos. Neste contexto, agradeço o companheirismo e as noites regadas a Raul Seixas, que meu grande amigo João Pedro me proporcionou.

Ao decorrer da minha graduação, conheci muitas pessoas importantes, que com certeza marcaram para sempre minha vida. Dentre elas, devo mencionar e agradecer ao meu grande amigo Carlos A. Cortez Jr., que em todos os momentos da minha trajetória universitária esteve ao meu lado. Ele foi meu amigo, minha dupla para trabalhos e estudos, meu companheiro de praticamente todas as disciplinas e momentos.

Devo indispensavelmente agradecer à minha namorada, por todo amor e afeto transmitido a mim, durante minha trajetória universitária. Ela foi minha companheira constante, a qual sempre esteve ao meu lado, seja em momentos delicados ou aventureiros. Além disso, é com ela e meu cachorro Ozzy, que divido de maneira mais leve este infortúnio momento de quarentena, em decorrência à Covid-19.

Resumo

O fenômeno da supercondutividade foi descoberto em 1911 por Heike K. Onnes. Desde então, não foi possível consolidar um desenvolvimento teórico capaz de explicar o comportamento de materiais supercondutores em diferentes faixas de temperatura. Desse modo, descrever propriedades supercondutoras, como a temperatura crítica (T_c), ainda é um grande desafio. Nessa perspectiva, usar ferramentas de Inteligência Artificial se torna uma alternativa para a predição de propriedades dos supercondutores. Assim, este trabalho busca avaliar a capacidade de alguns modelos de *machine learning* na predição de T_c. Para treinar os modelos de *machine learning*, foram utilizados dados de supercondutores extraídos do banco de dados do Instituto Nacional de Ciência dos Materiais do Japão (NIMS). Assim, a partir da fórmula química dos supercondutores extraídos, foram calculadas medidas estatísticas relacionadas à massa atômica, primeira energia de ionização, raio atômico, densidade, afinidade eletrônica, calor de fusão, condutividade térmica e valência. Com base nas medidas calculadas e na temperatura crítica obtida do NIMS, os modelos de *machine learning* foram submetidos ao processo de aprendizado supervisionado. Para melhor adequar estes modelos ao problema proposto, seus hiperparâmetros foram obtidos pelo processo de validação cruzada. Os modelos de *machine learning* avaliados nesta monografia são: Regressão Linear Múltipla; Elastic-Net; Máquinas de Vetores de Suporte; Árvore de Decisão; Floresta Aleatória; Árvores Extremamente Aleatórias; Gradient Boosting; Rede Neural Profunda (Multicamadas de Perceptrons). Como melhor resultado, o modelo Árvores Extremamente Aleatórias alcançou um R^2 de 0,94 e um $RMSE$ de 8,69 K para um conjunto de dados de teste, contendo amostras de supercondutores não empregadas no processo de treinamento.

Palavras-chave: Aprendizado de Máquina. Supercondutividade. Temperatura Crítica.

Abstract

The phenomenon of superconductivity was discovered in 1911 by Heike K. Onnes. Since then, it has not been possible to consolidate a theoretical development capable of explaining the behavior of superconducting materials in different temperature ranges. Thus, describing superconducting properties, such as critical temperature (T_c), is still a major challenge. In this perspective, using Artificial Intelligence tools becomes an alternative for the prediction of superconducting properties. Thus, this work seeks to evaluate the capacity of some machine learning models in the prediction of T_c. To train the machine learning models, data from superconductors extracted from the database of the National Institute of Materials Science in Japan (NIMS) were used. Thus, from the chemical formula of the superconductors extracted, statistical measures related to atomic mass, first ionization energy, atomic ray, density, electronic affinity, heat of fusion, thermal conductivity and valence were calculated. Based on the calculated measures and the critical temperature obtained from the NIMS, the machine learning models were submitted to the supervised learning process. To better adapt these models to the proposed problem, their hyperparameters were obtained by the cross-validation process. The machine learning models evaluated in this monograph are: Multiple Linear Regression; Elastic-Net; Support Vector Machines, Decision Tree; Random Forest; Extremely Random Trees; Gradient Boosting; Deep Neural Network (Perceptrons multilayer). As a best result, the Extremely Random Trees model reached a R^2 of 0,94 and a $RMSE$ of 8,69 K for a set of test data, containing samples of superconductors not used in the training process.

Keywords: Machine Learning. Superconductivity. Critical Temperature.

Lista de Ilustrações

Lista de Tabelas

Lista de Abreviaturas e Siglas

BCS	Bardeen-Cooper-Schrieffer
C.T.	Condutividade Térmica
CPU	*Central Process Unit* / Unidade Central de Processamento
DFT	*Density Functional Theory* / Teoria do Funcional da Densidade
EEL - USP	Escola de Engenharia de Lorena da Universidade de São Paulo
GB	Gigabyte
GPU	*Graphics Processing Unit* / Unidade de Processamento Gráfico
IA	Inteligência Artificial
ME	Migdal-Eliashberg
ML	*Machine Learning* / Aprendizado de Máquina
MLP	*Multi Layer Perceptron* / Perceptron Multicamada
NIMS	*National Institute for Materials Science* / Instituto Nacional de Ciência dos Materiais do Japão
RAM	*Random Access Memory* / Memória de Acesso Aleatório
RBF	*Radial Basis Function* / Função base radial
SQUID	*Superconducting Quantum Interference Device*
SVM	*Support Vector Machines* / Máquinas de Vetores de Suporte
YBCO	*Yttrium Barium Copper Oxide* / Óxido Ítrio Bário Cobre

Lista de Símbolos

K	Kelvin
T_c	Temperatura crítica
Δ	*gap*
E_F	Energia de Fermi
$\varepsilon_{\mathbf{k}}$	Espectro Eletrônico
V	Potencial
$\mathbf{k}$	Vetor de onda do elétron
T	Temperatura
ω_D	Frequência de Debye
N	Densidade de Estados
κ_B	Constante de Boltzmann
λ	Acoplamento
$\omega_{\log}$	Frequência Média Logarítmica dos Fônons
μ^*	Pseudopotencial de Morel-Anderson
f	Função Desconhecida
$\widehat{f}$	Aproximação para f
$\mathbf{x}$	Vetor de Características
$\mathbf{x}_i$	Vetor de Características para uma amostra i de $\mathbf{X}$
y_i	Valor de um rótulo para uma amostra i
$\Re$	Conjunto dos Números Reais
MSE	*Mean Square Error* / Erro Quadrático Médio
$RMSE$	*Root Mean Square Error* / Raiz Quadrada do Erro Quadrático Médio
R^2	Coeficiente De Determinação

$\mathbf{w}$	Vetor com pesos das características
w_0	Termo de Polarização
$\mathbf{X}$	Matriz com amostras $\mathbf{x}_i$
$\mathbf{Y}$	Matriz com amostras y_i
$\|\cdot\|_k$	Norma k
ρ	Taxa de Mistura
J	Função de Custo
α	Hiperparâmetro de Regularização
h	Função Hipótese (como $\widehat{f}$)
ϵ	Hiperparâmetro SVM: Precisão do Modelo
ξ	Variáveis de Folga
C	Hiperâmetro SVM: Controlador de Regularização
α_i	Multiplicador de Lagrange
K	Função Kernel
F	Estimador
L	Perda
l	Gradient Boosting: Função de Perda / Rede Neural: Número do Neurônio
g_i	Derivadas Parciais para o modelo Gradient Boosting
υ	Taxa de Aprendizado
u	Soma da entrada de um neurônio
f_a	Função de Ativação
w_j	Peso da j-ésima conexão de um neurônio
t	Tempo
c	Camada
e_l	Erro do Neurônio
$u.m.a$	Unidade de Massa Atômica

KJ	Kilojoules
mol	Constante de Avogadro
pm	Picômetro
Kg	Quilograma
m^3	Metro Cúbico
W	Watt
mK	Millikelvin
Δ	Intervalo da Propriedade
σ	Desvio Padrão
r	Coeficientede Correlação Pearson
μ	Média das Amostras

Sumário

1 Introdução

A supercondutividade já foi tema de 5 Prêmios Nobel em Física e por mais que tenha sido descoberta há aproximadamente 109 anos, ainda é uma área de estudos atual. De uma maneira ampla, um material é dito supercondutor quando conduz corrente elétrica a uma resistência praticamente nula, abaixo de determinada temperatura (HAMIDIEH, 2018). Esta temperatura, a qual abaixo dela o sistema entra no estado supercondutor, é denominada temperatura crítica (T_c).

Segundo Costa e Pavão (2012), a descoberta da supercondutividade foi extremamente disruptiva, pois contrariou a descrição das teorias de condutividade vigentes na época. Dessa forma, por mais que a supercondutividade tenha sido observada primeiramente por Heike K. Onnes em 1911, ela só foi satisfatoriamente teorizada pela teoria BCS, desenvolvida em 1957.

Apesar da teoria BCS ter revolucionado a história da supercondutividade, ela não foi capaz de explicar os adventos dos supercondutores de altas temperaturas e dos supercondutores com acoplamentos fortes (BOERI, 2020). Ainda assim, obter um modelo científico ou teoria capaz de predizer satisfatoriamente a temperatura crítica dos supercondutores, ainda é um problema em aberto (HAMIDIEH, 2018).

Nessa perspectiva, este trabalho se propõe a avaliar modelos de aprendizado de máquina, no processo de predição da temperatura crítica de supercondutores, a partir de suas fórmulas químicas. Os dados usados para treinar os modelos foram inspirados na abordagem usada por Hamidieh (2018). Nesta abordagem, medidas estatísticas de propriedades físicas foram calculadas com base na fórmula química dos supercondutores. As fórmulas químicas e as temperaturas críticas dos supercondutores foram obtidas do Instituto Nacional de Ciência dos Materiais do Japão.

Com o cálculo das características - medidas estatísticas das propriedades físicas -, os modelos de aprendizado de máquina foram submetidos ao treinamento pelo processo de aprendizado supervisionado. Além disso, buscando generalizar melhor os modelos, o processo de validação cruzada foi explorado para encontrar os melhores hiperparâmetros em cada caso.

No desenvolvimento da metodologia deste trabalho, os seguintes modelos de aprendizado de máquina foram abordados: Regressão Linear Múltipla; Elastic-Net; Máquinas de Vetores de Suporte; Árvore de Decisão; Floresta Aleatória; Árvores Extremamente Aleatórias; Gradient Boosting; Rede Neural Profunda. Estes modelos foram avaliados e comparados através das métricas R^2 e $RMSE$, que são, respectivamente, o coeficiente de determinação e a raiz quadrada do erro quadrático médio.

Por fim, espera-se com este trabalho avaliar a capacidade dos modelos apresentados acima, na tarefa de predição da temperatura crítica de novos supercondutores. Além disso, objetiva-se

encontrar o melhor modelo entre os avaliados e compará-lo com resultados encontrados em literaturas semelhantes.

1.1 Organização do Trabalho

No Capítulo 2 são apresentados conceitos básicos, que elucidam brevemente o panorama da supercondutividade e do aprendizado de máquina. É neste capítulo, que são abordados conceitos fundamentais sobre cada modelo de aprendizado de máquina e seus hiperparâmetros. No Capítulo 3 é apresentada a metodologia empregada no desenvolvimento dessa monografia. Além disso, neste capítulo são apresentados os procedimentos usados para: obtenção dos dados; busca dos melhores hiperparâmetros; treinamento e avaliação dos modelos. O Capítulo 4 é responsável por apresentar superficialmente os dados, mostrar os hiperparâmetros obtidos pelo processo de validação cruzada e discutir os resultados da predição de cada modelo, com base nas métricas R^2 e $RMSE$. Ainda, o Capítulo 4 é responsável por discutir de forma geral os resultados dos modelos e compará-los com a literatura. O Capítulo 5 é usado para sintetizar e concluir as discussões gerais sobre a avaliação dos modelos. Ademais, este último capítulo é usado para apresentar futuras pretensões sobre a continuação desta monografia.

2 Fundamentação Teórica

2.1 Supercondutividade

A supercondutividade foi descoberta em 1911 por Heike K. Onnes, ao estudar o comportamento da resistência elétrica dos materiais a baixas temperaturas (COSTA; PAVÃO, 2012). Onnes observou que próximo a 4,2 K a resistência do mercúrio caia abruptamente, comportando-se de forma inesperada perante as teorias de condutividade vigentes para época (COSTA; PAVÃO, 2012). A esta temperatura foi atribuído o nome de temperatura crítica (T_c), que representa a temperatura abaixo da qual determinado sistema entra no estado supercondutor, como brevemente mencionado na Capítulo 1 (COSTA; PAVÃO, 2012).

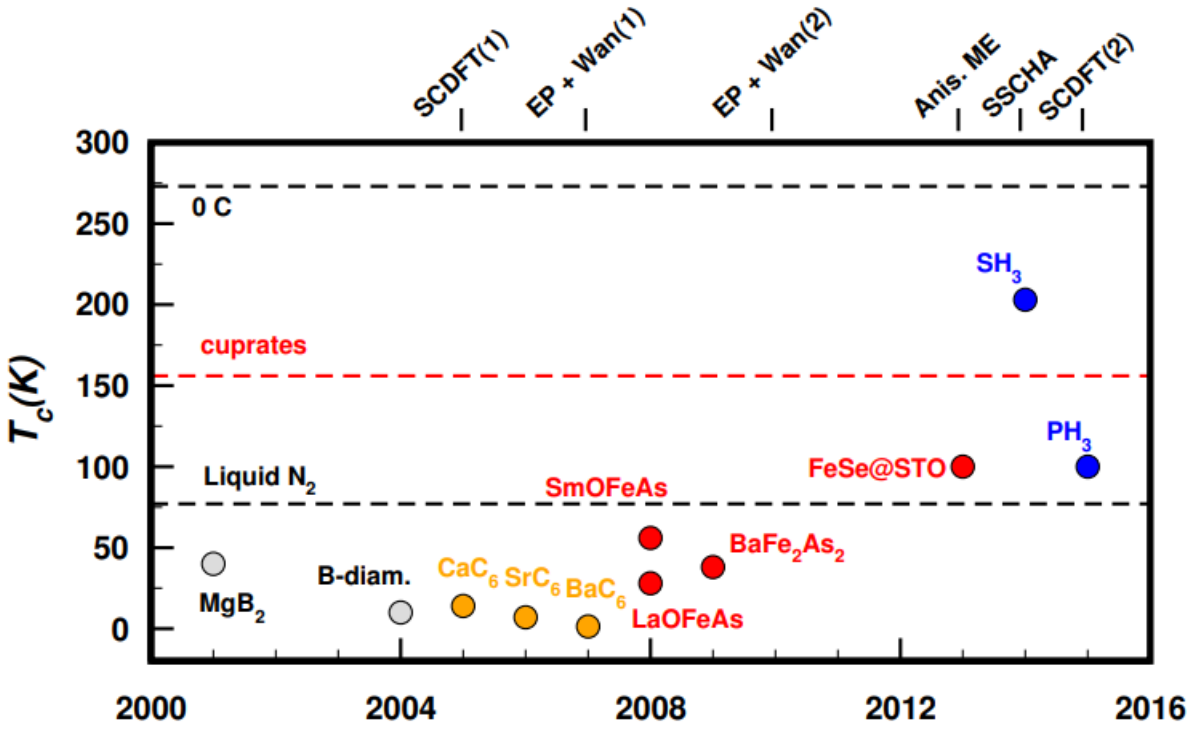
Basicamente, materiais supercondutores são aqueles capazes de conduzir corrente com resistência elétrica praticamente zero (HAMIDIEH, 2018). Dentre suas aplicações, pode-se destacar a atuação de supercondutores em aparelhos de ressonância magnética, dispositivos extremamente sensíveis a campos magnéticos (SQUIDs) e bobinas de altos campos, que são empregadas em aceleradores de partículas (HAMIDIEH, 2018).

Por mais tempo que a supercondutividade tenha sido descoberta, não há uma teoria consolidada capaz de descrever a ocorrência deste fenômeno em diferentes faixas de temperatura crítica (COSTA; PAVÃO, 2012). Além disso, a obtenção de supercondutores que manifestem o estado supercondutor em temperaturas próximas ou maiores que a ambiente, ainda é um grande desafio (COSTA; PAVÃO, 2012). Nessa perspectiva, predizer a temperatura crítica de um supercondutor através de um modelo ou teoria científica, ainda é um problema em aberto, como mencionado na Introdução (HAMIDIEH, 2018).

Após a descoberta da supercondutividade, a teoria BCS foi a primeira teoria microscópica a tentar descrevê-la, entretanto, ela foi incapaz de explicar o comportamento de supercondutores de altas temperaturas críticas (COSTA; PAVÃO, 2012). Apesar disso, a supercondutividade conseguiu evoluir com as predições de T_c e outras propriedades, através da teoria anisotrópica de Midgal-Eliashberg e por métodos baseados em Teoria do Funcional da Densidade (DFT), a partir de abordagens de primeiros princípios (*ab initio*) (BOERI, 2020).

Na Figura 2.1 são apresentadas temperaturas críticas em função do ano em que foram descobertas, para os principais supercondutores deste século. No eixo horizontal superior do gráfico da Figura 2.1, são elucidadas as principais metodologias desenvolvidas (BOERI, 2020): Teoria Funcional da Densidade Supercondutora (*Superconducting Density Functional Theory*) - SDFT(1); interação elétron-fônon com Funções Wannier - EP+WAN; teoria anisotrópica *ab initio* Migdal-Eliashberg - Anis. ME; aproximação harmônica autoconsistente - SSCHA; *ab initio* flutuações de *spin* - SCDFT(2).

Figura 2.1 – Temperaturas críticas e metodologias *ab initio* mais importantes do século 21 para a supercondutividade



Fonte: Boeri (2020)

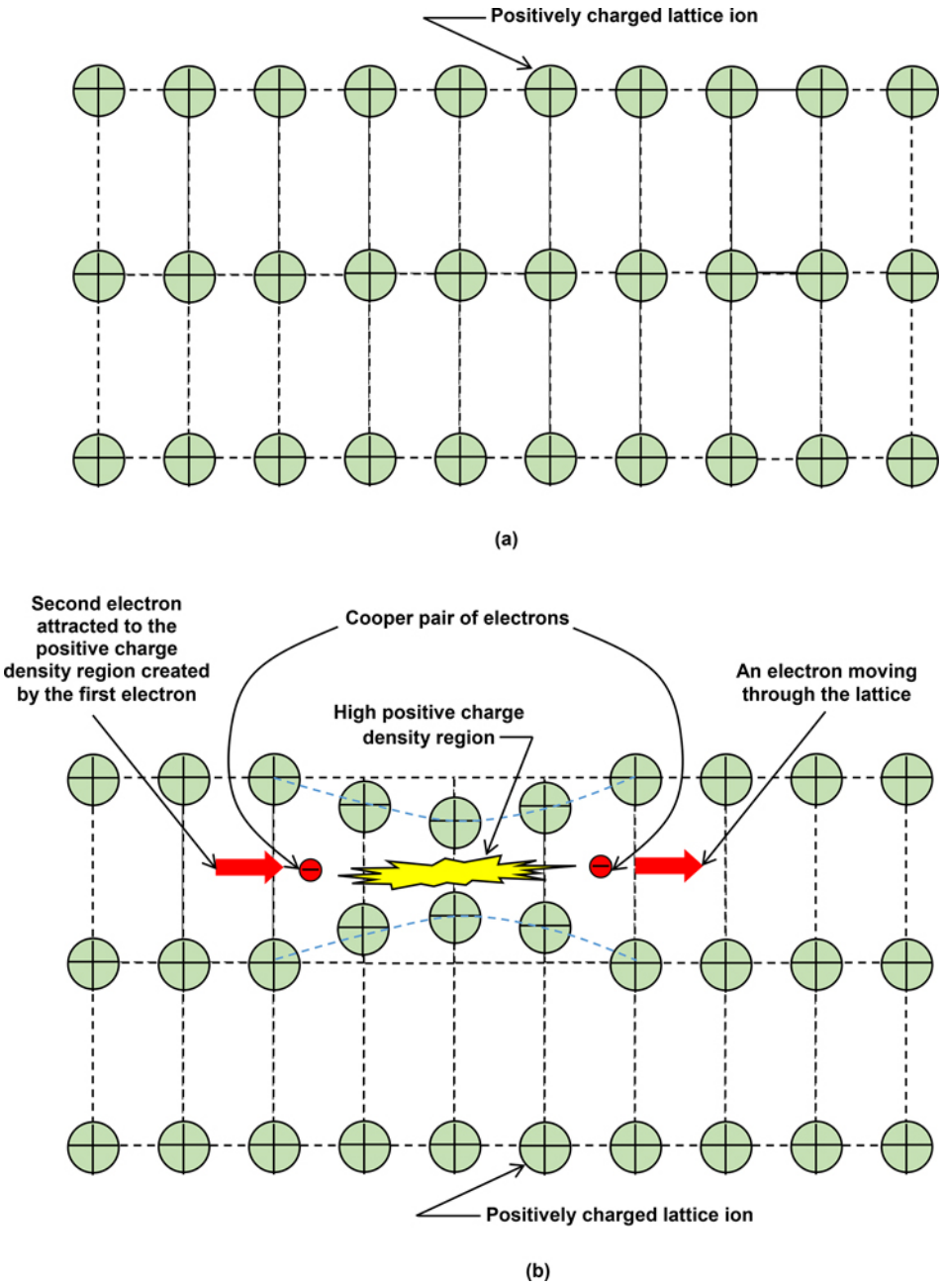
A fundamentação teórica por trás da supercondutividade e seus métodos computacionais *ab initio*, mostrados na Figura 2.1, é pautada nas teorias microscópicas de Bardeen-Cooper-Schrieffer (BCS) e de acoplamento forte de Migdal-Eliashberg (ME), e na capacidade da DFT em fornecer espectros eletrônicos e bosônicos precisos para a maioria dos materiais (BOERI, 2020).

Como mencionado anteriormente, a teoria BCS foi a primeira teoria microscópica usada para descrever a supercondutividade. Ela foi elaborada por Bardeen, Cooper e Schrieffer em 1957 (BOERI, 2020). O fator central desta teoria é a interação entre elétrons mediada pela energia quantizada de vibração da rede cristalina (fônons) (COSTA; PAVÃO, 2012).

Desse modo, um elétron interage com a estrutura da rede cristalina e a deforma. Após a deformação, um outro elétron a usa para minimizar sua energia, constituindo um sistema elétron-fônon-elétron (COSTA; PAVÃO, 2012). A Figura 2.2 elucida a deformação da rede cristalina para a formação do par de Cooper - como são chamados os dois elétrons que compõem o sistema elétron-fônon-elétron descrito nesse parágrafo (KHANNA, 2017).

Herbert Fröhlich foi o primeiro a propor a interação eletrônica mediada por fônons e formular o aparecimento de um *gap* proveniente dela (COSTA; PAVÃO, 2012). O *gap* mencionado refere-se ao aparecimento de uma banda proibida entre o estado fundamental e o primeiro estado

Figura 2.2 – A imagem (a) representa a rede cristalina para um metal acima da T_c. A imagem (b) apresenta a formação do par de Cooper a partir da interação elétron-fônon-elétron, para temperaturas abaixo da T_c



Fonte: Khanna (2017)

excitado do sistema elétron-fônon-elétron (COSTA; PAVÃO, 2012). Segundo Costa e Pavão (2012), o *gap* em questão emerge no estado supercondutor, pois a interação elétron-fônon gera uma atratividade entre os elétrons envolvidos, a qual é maior que a repulsão coulombiana.

Dessa forma, o *gap* (Δ) no estado supercondutor é desenvolvido ao redor do nível de Fermi (E_F), no qual Δ é máximo a 0 K e inexistente quando a temperatura (T) é igual a T_c

(BOERI, 2020). A razão de elétrons que formam pares de Cooper é dada por $\Delta/E_F \approx 10^{-3}$, que muitas vezes é denominada de razão de condensado de um supercondutor (BOERI, 2020).

Com base no que foi discutido até o momento para a BCS, a Equação 2.1 elucida um variacional para uma função de onda de muitos corpos, que representa a superposição de elétrons e pares de Cooper (BOERI, 2020). Dessa forma, a presença da fração de condensado faz com que o *gap* apareça no espectro eletrônico $\varepsilon_{\mathbf{k}}$.

$$\Delta_k = \frac{1}{2}\sum_{\mathbf{k},\mathbf{k}'} \frac{V_{\mathbf{k},\mathbf{k}'}\Delta_{\mathbf{k},\mathbf{k}'}}{\sqrt{\varepsilon_{\mathbf{k}}^2 + \Delta_{\mathbf{k}}^2}} \tanh\left(\frac{\sqrt{\varepsilon_{\mathbf{k}}^2 + \Delta_{\mathbf{k}}^2}}{2T}\right) \tag{2.1}$$

Na Equação 2.1, o potencial $V_{\mathbf{k},\mathbf{k}'}$ da teoria BCS representa a interação elétron-elétron, somente se os dois elétrons com vetores de onda $\mathbf{k}$ e $\mathbf{k}'$ estiverem em uma região de energia ω_D ao redor da energia de Fermi (E_F). Resolvendo a Equação 2.1 analiticamente obtêm-se (BOERI, 2020):

$$\Delta\left(T=0\right) \simeq 2\omega_D \exp\left(-\frac{1}{N\left(E_F\right)V}\right), \quad \kappa_B T_c = 1,13\omega_D \exp\left(-\frac{1}{N\left(E_F\right)V}\right), \tag{2.2}$$

no qual V representa a interação potencial entre os elétrons, ω_D representa a escala energética dos fônons (como a frequência de Debye), $N(E_F)$ é a densidade de estados no nível de Fermi e κ_B é a constante de Boltzmann. Nessa perspectiva, a 0 K a seguinte relação envolvendo T_c pode ser obtida (COSTA; PAVÃO, 2012):

$$2\Delta(0) = 3,52\kappa_B T_c. \tag{2.3}$$

Apesar de grandes contribuições para o campo da supercondutividade, a teoria BCS foi capaz de explicar somente supercondutores com acoplamento fraco ($\lambda = N(E_F)V << 1$) (BOERI, 2020). Além disso, ela foi inapta a explicar o comportamento de supercondutores com temperaturas críticas mais elevadas (COSTA; PAVÃO, 2012).

A descrição do acoplamento forte para a supercondutividade é feita quantitativamente pela teoria de muitos corpos de Migdal-Eliashberg (ME), que é baseada em um conjunto de equações diagramáticas autoconsistentes (BOERI, 2020). Neste contexto, os bósons que mediam o emparelhamento supercondutor podem ser fônons, plasmons ou flutuações de spin (BOERI, 2020).

Ademais, uma abordagem detalhada sobre a teoria de Midgal-Eliashberg usufrui de conceitos e formulações muito complexas e diverge do objetivo desta seção, que é o de elucidar teorias usadas na predição da temperatura crítica.

Apesar disso, deve-se mencionar a Equação 2.4, conhecida como expressão de Mc-Millan-Allen-Dynes, obtida para supercondutores que apresentam mediação por fônons, como mostrado

por Boeri (2020):

$$T_c = \frac{\omega_{log}}{1,2\kappa_B} \exp\left[-\frac{1,04(1+\lambda)}{\lambda - \mu^*(1+0,62\lambda)}\right], \tag{2.4}$$

no qual a interação elétron-fônon é dada por λ, ω_{log} representa a frequência média logarítmica dos fônons e μ^* é o pseudopotencial de Morel-Anderson. A equação em questão, concorda com conceitos provenientes da conhecida teoria ME para acoplamentos fortes.

Pode-se perceber que os esforços para teorizar a predição da temperatura crítica foram múltiplos, como as Equações 2.3 e 2.4 elucidam. Entretanto, como já anteriormente discutido, nenhuma formulação por si foi capaz de descrever com satisfatoriedade a temperatura crítica em diferentes faixas de temperatura.

2.2 Aprendizado de Máquina: Uma visão geral

A inteligência artificial (IA) foi considerada uma área teórica por muito tempo, sendo empregada em alguns poucos problemas (FACELI et al., 2011). Segundo Faceli et al. (2011), foi a partir da década de 1970 que a IA começou a ser mais empregada na resolução de problemas, através de técnicas computacionais.

Foi o aumento da complexidade dos problemas e o acesso aos grandes números de dados gerados nas últimas décadas, que propiciaram a necessidade da utilização de ferramentas e técnicas computacionais mais aprimoradas (FACELI et al., 2011). Nesse contexto, as técnicas de IA foram oportunas para desenvolver, a partir de uma experiência, hipóteses e funções capazes de descrever determinado problema (FACELI et al., 2011).

Faceli et al. (2011) define então que o desenvolvimento de uma hipótese ou função, a qual descreve determinado problema a partir de uma experiência passada, é chamado de aprendizado de máquina - ou em inglês *machine learning* (ML).

Em suma, o aprendizado de máquina busca otimizar parâmetros que compõem funções e hipóteses genéricas, a partir de um conjunto de dados (ALPAYDIN, 2020). Assim, o aprendizado de máquina consiste em executar um programa de computador, que ajusta os parâmetros de determinado modelo matemático, a partir dos dados de treinamento – dados usados no processo de ajuste dos modelos matemáticos. Posteriormente, o modelo ajustado é capaz de descrever o problema proposto pelos dados de treinamento.

O aprendizado de máquina usa teorias estatísticas na construção de modelos matemáticos, pois sua tarefa principal é a inferência amostral (ALPAYDIN, 2020). O processo de ajuste de parâmetros perante os dados, é chamado de treinamento. No treinamento, é necessário fornecer algoritmos capazes de resolver o problema de otimização e de processamento de um grande volume de dados (ALPAYDIN, 2020).

Durante o processo de treinamento, o ajuste de parâmetros das funções matemáticas pode não generalizar bem o problema proposto, podendo ocorrer duas situações: o sobreajuste (*overfitting*) ou o subajuste (*underfitting*) nos dados de treino.

No caso de sobreajuste, o modelo matemático praticamente memoriza os dados, fazendo com que uma função ajustada passe por praticamente quase todos os pontos do conjunto de dados de treino (GÉRON, 2019). Já o subajuste ocorre quando o modelo é incapaz de ajustar seus parâmetros, não encontrando padrões nos dados de treino (FACELI et al., 2011). O subajuste é o caso inverso do sobreajuste.

Como os modelos de ML se otimizam através de padrões encontrados nos dados, seus desempenhos dependem da qualidade dos mesmos (BISHOP, 2006). Nessa perspectiva, no contexto do aprendizado de máquina, o pré-processamento e a análise dos dados são uma das etapas mais importantes (GÉRON, 2019).

Na etapa de análise de dados, busca-se caracterizar cada característica do conjunto de dados, levantando qual o seu tipo e sua escala. Uma característica representa uma coluna (campo) na base de dados - ou até mesmo uma coluna em uma matriz de dados $\mathbf{X}$, na qual cada linha representa uma amostra $\mathbf{x}_i$ qualquer. Além disso, nesta etapa os dados devem ser explorados através de diversas medidas e gráficos, para o seu entendimento (ALPAYDIN, 2020).

O pré-processamento busca adequar os dados às tarefas as quais eles serão expostos. Nesta etapa é comum avaliar a mudança do formato de dado em cada característica ou fazer a limpeza de dados discrepantes ou incompletos (ALPAYDIN, 2020).

Para este trabalho, os dados foram explorados graficamente e estatisticamente, como exibido na Seção 4.1. Além disso, como a base de dados usada foi criada pela metodologia deste trabalho, os cuidados com a estrutura e representatividade foram garantidos no desenvolvimento apresentado na Seção 3.2. Dessa forma, a etapa de pré-processamento neste trabalho se baseou em avaliar a redução da dimensionalidade dos dados.

Como o presente trabalho busca predizer a temperatura crítica dos supercondutores, a partir de medidas estatísticas (características) extraídas de suas fórmulas químicas, utilizaram-se algoritmos de regressão baseados em modelos preditivos.

Os modelos preditivos buscam desenvolver um estimador para predizer um rótulo (T_c neste caso) com base em um domínio conhecido (FACELI et al., 2011). Como estamos diante de um problema de regressão, busca-se desenvolver uma função que associe um conjunto de características a um rótulo.

Para elucidar melhor o que foi discutido até aqui, este parágrafo se baseia na enunciação de Faceli et al. (2011). Para um conjunto de observações conhecidas $\{(\mathbf{x}_i, f(\mathbf{x}_i)), i = 1, ..., n\}$, f simboliza uma função não conhecida, que associa um vetor de características $\mathbf{x}_i$ a um rótulo y_i (T_c), que neste caso é o domínio de f. De acordo com Faceli et al. (2011), um modelo de ML aprende uma aproximação $\widehat{f}$ para a função desconhecida f. A aproximação $\widehat{f}$ é utilizada para

prever o valor de f para um novo vetor de características $\mathbf{x}$ (FACELI et al., 2011). A essência de f pode ser descrita pela equação abaixo:

$$y_i = f(\mathbf{x}_i) \in \Re, \tag{2.5}$$

no qual y_i representa o valor para um rótulo conhecido.

A qualidade de um estimador em fazer previsões é mediada por uma função de custo, que avalia o quão bem uma estimativa se aproxima do esperado (ALPAYDIN, 2020). A função de custo é usada para aprimorar o modelo, que tende a minimizá-la. Por mais que cada algoritmo possa possuir uma função de custo, em problemas de regressão geralmente ela está associada ao erro quadrático médio, representado abaixo (FACELI et al., 2011):

$$MSE\left(\widehat{f}\right) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \widehat{f}(\mathbf{x}_i)\right)^2, \tag{2.6}$$

no qual n é o número de instâncias do conjunto avaliado, y_i é o valor conhecido e $\widehat{f}(\mathbf{x}_i)$ o predito pelo modelo.

A Equação 2.6 representa o erro da hipótese $\widehat{f}$ em relação à diferença entre o valor estimado $\widehat{f}(\mathbf{x}_i)$ e o valor esperado y_i (FACELI et al., 2011). O MSE está vinculado com o $RMSE$, que é a medida de desempenho típica em problemas de regressão. O $RMSE$ é usado para quantificar os erros de um modelo de ML em suas previsões, dando um peso maior aos grandes erros (GÉRON, 2019). A equação para o $RMSE$, deriva da Equação 2.6 da seguinte forma:

$$RMSE\left(\widehat{f}\right) = \sqrt{MSE\left(\widehat{f}\right)}. \tag{2.7}$$

Outra medida de desempenho empregada nesta monografia é o coeficiente de determinação R^2, que representa a proporção de variância explicada pelas variáveis independentes (características) do modelo (SCIKIT-LEARN, 2020d). Assim, o R^2 explica também o quão bem as amostras nunca vistas pelos modelos serão previstas por ele (SCIKIT-LEARN, 2020d). A melhor pontuação para R^2 é 1 e a pior 0. A expressão abaixo representa como o R^2 foi calculado neste trabalho (SCIKIT-LEARN, 2020d):

$$R^2\left(y, \widehat{f}\right) = 1 - \frac{\sum_{i=1}^{n}\left(y_i - \widehat{f}\left(\mathbf{x}_i\right)\right)^2}{\sum_{i=1}^{n}\left(y_i - \overline{y}\right)^2}, \tag{2.8}$$

no qual $\overline{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$.

Nesta monografia o aprendizado supervisionado foi empregado. Basicamente, para este tipo de aprendizado os dados usados para treinar o modelo incluem as soluções desejadas, chamadas também de rótulos (GÉRON, 2019). Exemplificando com a problemática deste trabalho, ao entregar os dados calculados para cada supercondutor, apresenta-se ao modelo a sua temperatura crítica.

Desse modo, suponha-se que os dados estatísticos obtidos através da fórmula química de determinado supercondutor, sejam representados pelo vetor de atributos $\mathbf{x}_i$. Para o algoritmo em questão, são apresentados o conjunto $\mathbf{x}_i$ e a temperatura crítica (y_i), associada a $\mathbf{x}_i$. Assim, ao apresentar $\mathbf{x}_i$ e o seu devido valor y_i, o modelo aprimora os parâmetros de uma $\widehat{f}$ genérica, de forma que ela se aproxime da função desconhecida (Equação 2.5), que representa a dependência ideal da temperatura crítica aos atributos $\mathbf{x}_i$.

Uma forma de saber se o modelo está generalizando bem o problema analisado, é testá-lo na predição de novos rótulos, a partir de novos valores de características ($\mathbf{x}$) (GÉRON, 2019). Uma opção para avaliar a predição de modelos sobre novos dados, pode ser a divisão do conjunto de dados em conjunto de treino e conjunto de teste (GÉRON, 2019). Assim, é possível treinar o modelo utilizando os dados do conjunto de treino, e avaliá-lo utilizando o conjunto de teste.

Em literaturas como a de Bishop (2006) e Géron (2019), é comum utilizar 80% dos dados para treinar o modelo e o restante para testá-lo. Ao avaliar o modelo sobre a perspectiva dos dados de teste, obtêm-se o erro de generalização (GÉRON, 2019). Ainda segundo Géron (2019), se este erro for alto pode indicar que o modelo está sobreajustado aos dados usados no treinamento.

Como será discutido adiante neste trabalho, cada modelo (algoritmo) de ML possui hiperparâmetros que regulam os parâmetros relacionados à sua formulação matemática. O processo de controle dos hiperparâmetros é chamado de regularização, e busca restringir o ajuste de parâmetros dos modelos aos dados (GÉRON, 2019). Dessa forma, o processo de regularização busca evitar sobreajustes nos dados de treinamento, fazendo com que o modelo consiga descrever o problema proposto da maneira mais genérica possível (SCIKIT-LEARN, 2020a).

Então, uma maneira de diminuir o erro de generalização é ajustar os hiperparâmetros, de forma que eles possam generalizar melhor o modelo de ML empregado (GÉRON, 2019). Entretanto, usar somente o conjunto de teste para avaliar determinado arranjo de hipeparâmetros, pode conduzir o modelo a uma sobreposição sobre os dados de teste (SCIKIT-LEARN, 2020a). Nessa situação, os hiperparâmetros são ajustados para conduzir melhores resultados no conjunto de teste, ajustando o modelo a este conjunto.

Para contornar este problema, usa-se um procedimento denominado validação cruzada. O processo de validação cruzada busca criar um conjunto de validação nos dados de treinamento, de modo a deixar o conjunto de teste somente para avaliar o desempenho final do modelo treinado (FACELI et al., 2011). Assim, a maior parte do conjunto de treino é usada para treinar o modelo e uma pequena parte deste conjunto é usada para avaliá-lo.

O processo de treinamento é repetido algumas vezes, alternando os dados do conjunto de validação. Desse modo, é possível testar o melhor arranjo de hiperparâmetros em cada ciclo de treinamento (SCIKIT-LEARN, 2020a). Ademais, uma melhor explicação do processo de validação cruzada, direcionada aos propósitos deste trabalho, é feita no Capítulo 3 por ser uma das principais etapas da metodologia.

Por fim, os modelos usados nesta monografia, bem como seus fundamentos, serão explicados nas próximas subseções. Todos os modelos nesta monografia foram treinados sobre a perspectiva do aprendizado supervisionado. Além disso, todos os modelos foram submetidos ao processo de validação cruzada para determinação dos melhores hiperparâmetros. A avaliação dos resultados de cada modelo foi feita pelas medidas de desempenho $RMSE$ e R^2, sobre os dados de teste.

2.2.1 Modelos lineares

Os modelos apresentados nesta seção têm como objetivo a tarefa de regressão, na qual $\widehat{f}$ é formulada a partir da combinação linear das características da base de dados. Assim, $\widehat{f}$ pode ser descrita da seguinte forma (SCIKIT-LEARN, 2020c):

$$\widehat{f}(\mathbf{w}, \mathbf{x}) = w_0 + w_1 x_1 + ... + w_p x_p, \tag{2.9}$$

no qual $\widehat{f}$ é a função que estima valores para os rótulos, p o número de características (ou atributos), $\mathbf{w} = (w_1, ..., w_p)$ o vetor com os coeficientes (ou pesos dos atributos) e w_0 é o termo de polarização (ou interceptação). As colocações entre parênteses possuem o mesmo significado dos termos que as precedem. Por exemplo, na literatura os nomes das colunas das bases de dados podem ser chamados de atributos ou características.

2.2.1.1 Regressão Linear Múltipla - Método dos Mínimos Quadrados

Para este modelo, ocorre um ajuste dos coeficientes $\mathbf{w} = (w_1, ..., w_p)$ para minimizar a soma residual dos quadrados entre o rótulo observado (T_c observada) e o previsto (T_c prevista). Assim, para treinar este modelo deve-se resolver matematicamente a seguinte equação (SCIKIT-LEARN, 2020c):

$$\min_{w} ||\mathbf{X}\mathbf{w} - \mathbf{Y}||_2^2, \tag{2.10}$$

sendo $\mathbf{X}$ um conjunto de dados em que cada linha representa uma amostra qualquer $\mathbf{x}_i$. E $\mathbf{Y}$ representa a matriz de uma coluna, que contém todos os rótulos (medidas de T_c). Em suma, uma amostra $\mathbf{x}_i$ de $\mathbf{X}$, tem sua y_i correspondente, sendo que y_i está em $\mathbf{Y}$.

A Equação 2.10 representa a função de custo para a Equação 2.9. Assim, ajustando os coeficientes - parâmetros deste modelo - com dados conhecidos, é possível consolidar uma função que generalize o problema de predição e possa predizer valores para novas amostras.

O $||.||_2$ na Equação 2.10 representa a norma euclidiana. Segundo Géron (2019), a norma $||.||_k$ pode ser generalizada da seguinte forma:

$$||\mathbf{v}||_k = \left(|v_0|^k + |v_1|^k + ... + |v_n|^k \right)^{\frac{1}{k}}, \tag{2.11}$$

no qual $\mathbf{v}$ é um vetor como n elementos. A norma apresentada na Equação 2.10 corresponde ao $RMSE$, apresentado na Equação 2.7 (GÉRON, 2019). Nessa perspectiva, é possível explicar

porquê o $RMSE$ é mais sensível a dados discrepantes. Como pode ser visto na Equação 2.11, quanto maior o k mais a norma se concentra em grandes valores (GÉRON, 2019). É por esse fato que a raiz do erro quadrático médio é mais usada do que o erro médio absoluto, no contexto de ML.

Perante o parágrafo anterior, é possível inferir que a Equação 2.10 representa o MSE. Dessa forma, a função de custo, neste caso, é simplesmente igual ao erro quadrático médio.

2.2.1.2 Elastic Net

O modelo Elastic Net foi empregado para tentar avaliar o efeito da regularização sobre o método de regressão linear múltipla. O modelo Elastic Net é uma mistura dos modelos de Ridge e Lasso. Para este modelo, pode-se controlar o hiperparâmetro ρ, que representa a taxa de mistura entre os dois modelos mencionados (GÉRON, 2019).

A função de custo Elastic Net apresentada por Géron (2019), pode ser escrita da seguinte forma:

$$J(\mathbf{w}) = MSE(\mathbf{w}) + \rho\alpha \left\|\mathbf{w}\right\|_1 + \frac{\alpha(1-\rho)}{2} \left\|\mathbf{w}\right\|_2^2, \tag{2.12}$$

o termo $MSE(\mathbf{w})$ apresenta basicamente a função de custo mostrada na Equação 2.10 e α é um hiperparâmetro, o qual seu valor indica o quão regularizado o modelo deve estar.

O termo com norma $k = 2$ (norma euclidiana) é proveniente da regressão de Ridge. Este termo busca manter os parâmetros do modelo (neste caso o $\mathbf{w}$) o mais reduzido possível (GÉRON, 2019). O hiperparâmetro α, que acompanha este termo, busca traduzir o quanto o modelo deve ser regularizado. Segundo Géron (2019), se α for muito pequeno o modelo tende a uma regressão linear múltipla, se for muito grande ele tende a fazer os pesos ($\mathbf{w}$) ficarem próximos de zero.

O termo com norma $k = 1$ representa o modelo de regressão de Lasso. Devido à norma, este modelo procura extinguir os pesos relacionados às características menos relevantes (GÉRON, 2019). Para este modelo, o hiperparâmetro α tem a mesma representatividade do modelo de Ridge.

Quando o hiperparâmetro ρ é zero, a Equação 2.12 é igual a função de custo para o modelo Ridge. Quando ρ é um, a Equação 2.12 é igual a função de custo do modelo Lasso.

Autores como Hamidieh (2018) e Bishop (2006) apresentam a função de custo como a apresentada na Equação 2.12. Assim, o ajuste do modelo é proveniente da minimização de uma função J. Por outro lado, a documentação da biblioteca Scikit-learn apresenta a função de custo como a apresentada pela Equação 2.10.

É importante mencionar que os hiperparâmetros α e ρ são determinados pelo processo de validação cruzada. Neste processo, várias combinações de α e ρ são testadas no conjunto de treino. A combinação que apresenta os melhores valores para a medida de desempenho $RMSE$, é a escolhida para ser incorporada ao modelo.

2.2.2 Máquinas de Vetores de Suporte - *Support Vector Machine* (SVM)

Seguindo a explicação teórica de Faceli et al. (2011), o modelo de SVM para a tarefa de regressão visa encontrar uma função $h(\mathbf{x})$ que gere saídas contínuas perante os dados de treino, desviando um valor máximo de ϵ do rótulo esperado. Além disso, requer-se para $h(\mathbf{x})$ uniformidade e regularidade (FACELI et al., 2011).

Considerando inicialmente um comportamento linear para $h(\mathbf{x})$, tem-se o hiperplano (FACELI et al., 2011):

$$h(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b, \tag{2.13}$$

no qual $\mathbf{w} \cdot \mathbf{x}$ representa o produto escalar entre os vetores $\mathbf{w}$ e $\mathbf{x}$, $\mathbf{w}$ neste caso representa um vetor normal ao hiperplano e $\frac{b}{||\mathbf{w}||}$ com $b \in \Re$ é a distância do hiperplano em relação à origem.

O modelo então é ajustado para encontrar um pequeno $\mathbf{w}$ para a Equação 2.13, isto é feito através da minimização da norma $||\mathbf{w}||$ como segue (GÉRON, 2019):

$$\min_{\mathbf{w},b} \frac{1}{2}\mathbf{w}^2. \tag{2.14}$$

Além disso, a otimização mostrada na Equação 2.14 é sujeita às seguintes restrições (FACELI et al., 2011):

$$\begin{cases} y_i - \mathbf{w} \cdot \mathbf{x_i} - b \leq \epsilon_i \\ \mathbf{w} \cdot \mathbf{x_i} + b - y_i \leq \epsilon_i \end{cases} \tag{2.15}$$

Nessa perspectiva, Faceli et al. (2011) resume que a função linear busca associar os pares $(\mathbf{x}_i, y_i)$ do conjunto de treino com uma precisão de ϵ. Ademais, este processo pode ser elucidado na Figura 2.3.

Dessa forma, como sugere a Figura 2.3, espera-se obter uma h na qual os dados usados no treinamento fiquem na região sombreada ao redor dela.

Em problemas reais dificilmente é possível enquadrar os dados nas margens delimitadas por ϵ e $-\epsilon$. Nessa perspectiva, para lidar com ruídos e *outliers*, deve-se introduzir variáveis de folga, que possibilitam que algumas amostras predigam fora da margem ϵ e $-\epsilon$, elucidada na Figura 2.3 (FACELI et al., 2011).

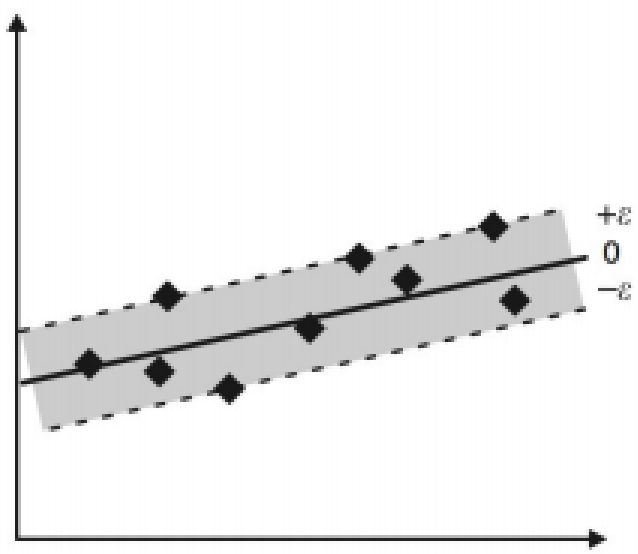
Desse modo, Faceli et al. (2011) propõe o seguinte problema de otimização:

$$\min_{\mathbf{w},b,\xi,\overline{\xi}} \frac{1}{2} \left\|\mathbf{w}^2\right\| + C \left(\sum_{i=1}^{n} \xi_i + \overline{\xi}_i \right), \tag{2.16}$$

no qual ξ e $\overline{\xi}$ representam as variáveis de folga e C uma variável que equilibra a regularização de h com a quantidade de desvios permitidos. Assim, obtêm-se as seguintes restrições:

$$\begin{cases} y_i - \mathbf{w} \cdot \mathbf{x_i} - b \leq \epsilon_i + \xi \\ \mathbf{w} \cdot \mathbf{x_i} + b - y_i \leq \epsilon_i + \overline{\xi} \\ \xi_i, \overline{\xi}_i \geq 0 \end{cases} \tag{2.17}$$

Figura 2.3 – Ilustração do procedimento realizado por um SVM para uma tarefa de regressão



Fonte: Faceli et al. (2011)

É muito comum no âmbito do aprendizado de máquina, resolver os problemas de otimização empregando a formulação lagrangiana. Por mais que explicar a empregabilidade da formulação lagrangiana esteja fora dos objetivos deste trabalho, ela se faz necessária para elucidar a utilização dos SVMs na metodologia apresentada.

Resumidamente, para resolver problemas de otimização, como apresentado na Equação 2.16, modela-se uma expressão lagrangiana que abranja o problema de otimização em questão, e suas restrições associadas a um multiplicador de lagrande. Dessa forma, a partir de uma expressão lagrangiana, calculam-se suas derivadas parciais para o valor nulo. Os resultados das derivações parciais são usados para substituir termos na equação lagrangiana inicial.

Usando o procedimento apresentado no parágrafo anterior para as Equações 2.16 e 2.17 é possível obter a seguinte expressão (FACELI et al., 2011):

$$\max_{\alpha,\overline{\alpha}} -\frac{1}{2}\sum_{i,j=1}^{n}(\alpha_i-\overline{\alpha}_i)(\alpha_j-\overline{\alpha}_j)K(\mathbf{x}_i,\mathbf{x}_j)-\epsilon\sum_{i=1}^{n}(\alpha_i-\overline{\alpha}_i)+\sum_{i=1}^{n}y_i(\alpha_i-\overline{\alpha}_i), \tag{2.18}$$

no qual α_i e $\overline{\alpha}_i$ são variáveis (multiplicadores) de Lagrange e K é a função kernel. K emerge do produto interno gerado na manipulação da Equação 2.17 na formulação lagrangiana. As restrições para a Equação 2.18 ficam (FACELI et al., 2011):

$$\begin{cases} \sum_{i=1}^{n}(\alpha_i-\overline{\alpha}_i)=0 \\ \alpha_i,\overline{\alpha_i}\in[0,C] \end{cases} \tag{2.19}$$

Para finalizar a explicação do uso da formulação lagrangiana, deve-se colocar que dentro da margem, entre ϵ e $-\epsilon$, as variáveis de Langrange são zero.

Dessa forma, o problema de otimização para as SVM é de maximizar a Equação 2.18. No âmbito desta monografia, a Equação 2.18 foi otimizada para três funções de kernel K. Estas funções podem ser extraídas da Tabela 2.1. Os parâmetros σ, κ e d são determinados para cada problema.

Tabela 2.1 – Tipos de Kernel avaliados na otimização das SVM

Tipo de kernel	**Função $K(\mathbf{x}_i, \mathbf{x}_j)$**	**Parâmetros**
Polinomial	$(\delta\,(\mathbf{x}_i \cdot \mathbf{x}_j) + \kappa)^d$	δ ,κ e d
RBF	$\exp\left(-\sigma\,\|\mathbf{x}_i - \mathbf{x}_j\|^2\right)$	σ

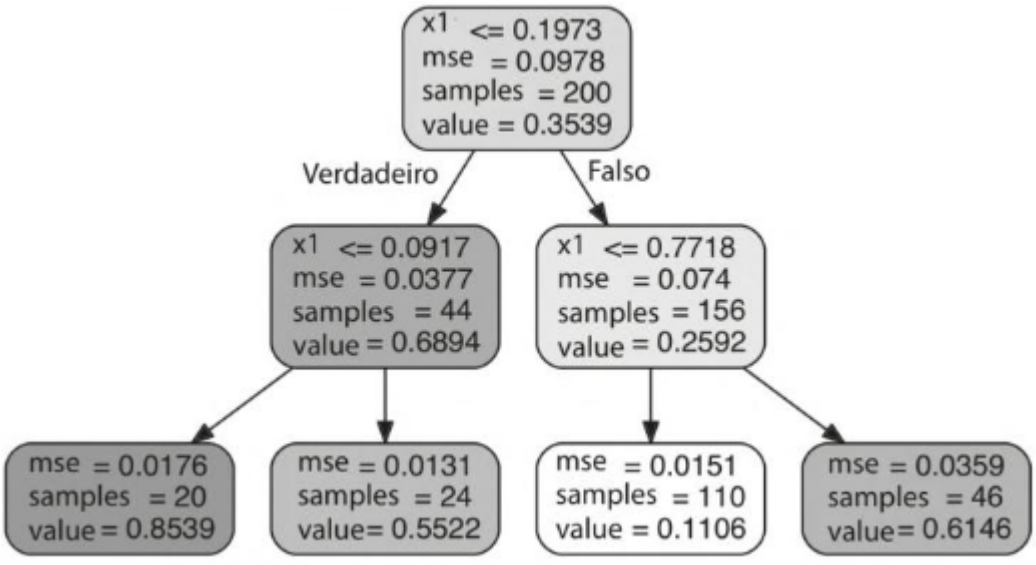
Fonte: Faceli et al. (2011)

Além dos K apresentados na Tabela 2.1, a condição linear para as SVM foi avaliada para a seguinte configuração do Kernel Polinomial: $\delta = 1$, $d = 1$ e $\kappa = 0$. Quando não calculados pelo próprio algoritmo, os parâmetros σ, κ e d foram obtidos através do processo de validação cruzada.

2.2.3 Árvores de Decisão

As árvores de decisão são comumente empregadas para as tarefas de classificação. Entretanto, elas podem ser usadas para tarefas de regressão, conseguindo descrever problemas bastantes complexos (GÉRON, 2019). Baseando-se na fundamentação de Géron (2019), o funcionamento de uma árvore de decisão pode ser descrito a partir do exemplo ilustrado pela Figura 2.4.

Figura 2.4 – Exemplo de uma árvore de decisão direcionada à tarefa de regressão



Fonte: Géron (2019)

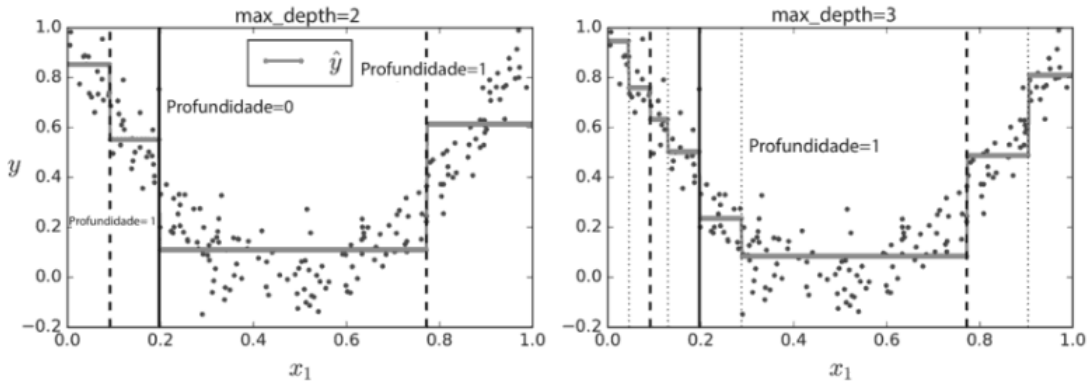
A Figura 2.4 representa o desenvolvimento de uma árvore de decisão para uma característica, ou seja, representa apenas um atributo do conjunto de treino. Em problemas reais, a árvore de decisão engloba todas as características do conjunto de treino em sua construção.

Supõe-se que determinada característica x_1 apresente o valor 0,8 e supõe-se também que a árvore da Figura 2.4 prediga T_c a partir de x_1. Assim, uma amostra $\mathbf{x}_i$ com característica $x_1 = 0,8$ entra na raiz da árvore (primeiro quadrado de cima para baixo na Figura 2.4). Na raiz, ela encontra a condição $x_1 \leq 0,1973$. Para esta condição, a amostra apresenta o resultado Falso, assim, ela caminha para a direita na Figura 2.4. E dessa forma, a amostra vai passando por cada nível (profundidade) da árvore, até atingir o último nó da folha construído.

Para a mostra em questão, o nó da folha prediz uma temperatura crítica de 0,6146 K. A previsão neste nó é simplesmente a média dos rótulos das 46 amostras de treinamento, que foram empregadas em sua construção (GÉRON, 2019). Além do valor previsto, é possível recolher informações sobre o MSE da predição, que deriva das 46 amostras que compõem o nó.

A Figura 2.5 apresenta a $\widehat{f}$, que emerge da árvore da Figura 2.4, para uma profundidade igual a 2 e a 3. O valor previsto por $\widehat{f}$ em cada intervalo da Figura 2.5, refere-se à média dos rótulos das amostras de treinamento no intervalo (GÉRON, 2019).

Figura 2.5 – Previsão do modelo Árvore de Decisão com profundidade 2 (imagem do lado esquerdo) e profundidade 3 (imagem do lado direito)



Fonte: Géron (2019)

Exemplificado pela Figura 2.5, o algoritmo busca dividir as regiões de diferentes maneiras, de modo a fazer $\widehat{f}$ se aproximar ao máximo das amostras de treino (GÉRON, 2019).

Ademais, o algoritmo do modelo Árvore de Decisão tende a minimizar a seguinte função de custo (GÉRON, 2019):

$$J(k, t_k) = \frac{m_{esquerda}}{m} MSE_{esquerda} + \frac{m_{direita}}{m} MSE_{direita}, \tag{2.20}$$

no qual k representa uma característica qualquer, t_k um limiar ($x_1 \leq 0,1973$, por exemplo), $m_{esquerda/direita}$ número de amostras em cada divisão (subconjunto),m o número de amostras

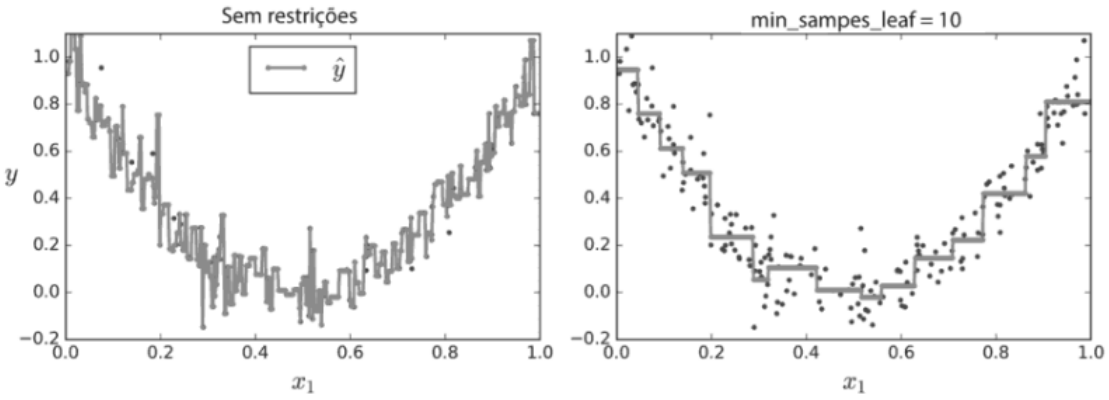
avaliadas pela função de custo e $MSE_{esquerda,direita}$ o erro quadrático médio dos subconjuntos a serem divididos. Assim, têm-se o cálculo em cada nó (GÉRON, 2019):

$$\begin{cases} MSE_{\text{nó}} = \sum_{i \in \text{nó}} (\widehat{f}_{\text{nó}} - y_i)^2 \\ \widehat{f}_{\text{nó}} = \frac{1}{m_{\text{nó}}} \sum_{i \in \text{nó}} y_i \end{cases} \tag{2.21}$$

Após o algoritmo do modelo Árvores de Decisão dividir o subconjunto dos dados de treino em dois, utilizando a Equação 2.20, ele utiliza os mesmos artifícios para os outros subconjuntos, até alcançar uma árvore como a da Figura 2.4 (GÉRON, 2019).

As árvores de decisão são bastantes propensas a sobreajustes, pois podem se desenvolver até suas estruturas chegarem bem próximas aos dados de treinamento (GÉRON, 2019). Assim, regularizar alguns hiperparâmetros das árvores de decisão é fundamental para generalizar bem uma hipótese $\widehat{f}$. A Figura 2.6 mostra o comportamento de uma $\widehat{f}$ não regularizada e outra $\widehat{f}$ com restrições no número mínimo de amostras que um nó da folha deve possuir.

Figura 2.6 – Efeito da regularização no modelo Árvore de Decisão



Fonte: Géron (2019)

O gráfico à esquerda na Figura 2.6 apresenta sobreajuste nos dados de treino e o gráfico à direita apresenta uma $\widehat{f}$ regularizada para o modelo Árvore de Decisão. As regularizações mais empregadas nas árvores de decisão são nos seguintes hiperparâmetro (GÉRON, 2019): profundidade da árvore; número mínimo de amostras que um nó deve ter antes de se dividir; número mínimo de amostras que um nó da folha deve possuir.

Ressalta-se aqui, que os melhores valores para os hiperparâmetros são levantados no processo de validação cruzada.

2.2.4 Floresta Aleatória e Árvores Extremamente Aleatórias

As Florestas Aleatórias e as Árvores Extremamente Aleatórias fazem parte de uma técnica conhecida como *Ensemble Learning*. Esta técnica se baseia em usar diversos estimadores para

melhorar os resultados dos estimadores individuais (GÉRON, 2019). Até o presente momento, apenas estimadores individuais foram abordados nesta monografia.

Ambos modelos tratados nesta seção são baseados na construção de diversas árvores de decisão. Assim, esses modelos se baseiam nessas árvores para fazerem suas previsões. Além de usarem diversas árvores de decisão, os modelos desta seção buscam criar árvores de decisão com critérios aleatórios. Dessa forma, eles tendem a criar diversas árvores de decisão diferentes.

Como foi mostrado na Seção 2.2.3, as árvores de decisão tendem a escolher a melhor característica para dividir um novo nó. Apesar disso, o modelo Floresta Aleatória busca escolher aleatoriamente uma característica para dividir um nó, no momento de construção de suas árvores de decisão (GÉRON, 2019). Assim, as Florestas Aleatórias tendem a gerar uma ampla diversidade de árvores, como já mencionado.

Ademais, a aleatoriedade injetada nas florestas, no momento de construção das árvores, faz com que as árvores de decisão possuam erros de previsão dissociados. Por outro lado, por possuir um grande número de árvores estes erros são corrigidos, uma vez que a predição dos modelos *ensemble* é feita pela média das predições dos estimadores (SCIKIT-LEARN, 2020b).

As Florestas Aleatórias atingem uma variação reduzida por combinarem diversas árvores, assim, elas se submetem a um aumento no viés (SCIKIT-LEARN, 2020b). A pequena variância atingida é usualmente associada a uma predição mais generalizada (SCIKIT-LEARN, 2020b).

Nas Árvores Extremamente Aleatórias, além de escolher uma característica aleatória para dividir um nó das árvores de decisão, o limiar desta característica também é escolhido aleatoriamente (GÉRON, 2019). Este fato torna o treinamento das Árvores Extremamente Aleatórias mais rápido do que o das Florestas Aleatórias, uma vez que encontrar o melhor limiar para determinada característica é uma das tarefas mais demoradas no processo de construção das árvores de decisão (GÉRON, 2019). Como afirma Géron (2019), troca-se novamente um maior viés por uma menor variância no modelo Árvores Extremamente Aleatórias.

Por fim, foi possível entender que ambos modelos usufruem da construção aleatória de diversas árvores de decisão. Além disso, apresentou-se que a utilização dessas árvores pode diminuir a variância das predições, a um custo no aumento do viés. Em questão da aleatoriedade nas árvores de decisão, pode-se afirmar que as Árvores Extremamente Aleatórias produzem árvores muito mais aleatórias que as Florestas Aleatórias.

Para estes modelos, os hiperparâmetros avaliados pelo processo de validação cruzada foram os mesmos das árvores de decisão. Entretanto, nos modelos desta seção foi avaliado também a quantidade de árvores de decisão construídas.

2.2.5 Gradient Boosting

O Gradient Boosting é um método *ensemble* que usufrui de diversos estimadores. Nesta monografia, os estimadores usados para o modelo Gradient Boosting também são árvores de decisão. Ao invés de construir diversos estimadores independentes, o algoritmo deste modelo busca adicioná-los sequencialmente, de modo que eles possam corrigir os erros residuais dos estimadores anteriores (GÉRON, 2019).

A seguinte formulação matemática pode ser feita para o Gradient Boosting (SCIKIT-LEARN, 2020b):

$$\widehat{f_i} = F_M(\mathbf{x}_i) = \sum_{m=1}^{M} h_m(\mathbf{x}_i), \tag{2.22}$$

no qual $\widehat{f_i}$ representa a estimação para um y_i, com base em uma amostra genérica $\mathbf{x}_i$. Além disso, h_m são os estimadores para as árvores de decisão e M é o número de estimadores (árvores).

Como mencionado, os estimadores são adicionados sequencialmente, assim, podem ser descritos da seguinte forma (SCIKIT-LEARN, 2020b):

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + h_m(\mathbf{x}), \tag{2.23}$$

no qual h_m (uma nova árvore) é introduzido ao estimador anterior (F_{m-1}) para minimizar a soma de suas perdas (L_m). Desse modo, a minimização da soma das perdas pode ser escrita da seguinte maneira (SCIKIT-LEARN, 2020b):

$$h_m = \min_h L_m = \min_h \sum_{i=1}^{n} l(y_i, F_{m-1}(\mathbf{x}_i) + h(\mathbf{x}_i)). \tag{2.24}$$

Para esta monografia, $l(y_i, F(\mathbf{x}_i))$ representa uma função de perda com base no método dos mínimos quadrados. Os mínimos quadrados são escolhidos por possuirem robustez em resolver problemas de regressão (SCIKIT-LEARN, 2020b).

O F_0 inicial, que emerge da Equação 2.24, representa a média dos rótulos quando l for referência ao método dos mínimos quadrados (SCIKIT-LEARN, 2020b). Expandindo $l(y_i, F_{m-1}(\mathbf{x}_i) + h_m(\mathbf{x}_i))$ por aproximação de Taylor, tem-se (SCIKIT-LEARN, 2020b):

$$l(y_i, F_{m-1}(\mathbf{x}_i) + h_m(\mathbf{x}_i)) \approx l(y_i, F_{m-1}(\mathbf{x}_i)) + h_m(\mathbf{x}_i) \left[\frac{\partial l(y_i, F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)}\right]_{F=F_{m-1}}. \tag{2.25}$$

Como l é diferenciável, o cálculo de $\left[\frac{\partial l(y_i, F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)}\right]_{F=F_{m-1}}$ é resolvido facilmente para uma forma fechada. Chamando o termo das derivadas parciais de g_i, o seguinte problema de otimização é alcançado:

$$h_m \approx \min_h \sum_{i=1}^{n} h(\mathbf{x}_i) g_i. \tag{2.26}$$

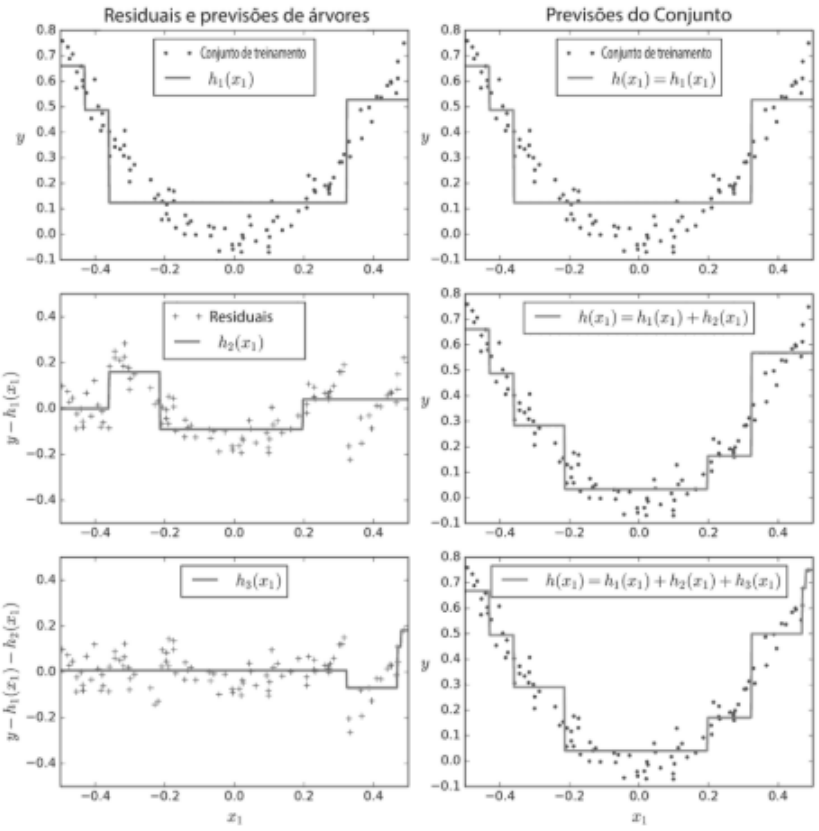
A Equação 2.26 sugere que o estimador h_m busca prever um gradiente negativo das amostras anteriores durante seu reajuste, configurando um gradiente descendente (SCIKIT-LEARN, 2020b).

Para regularizar as correções nos estimados, é introduzido um termo denominado taxa de aprendizagem. Assim, cada vez que uma árvore (h_m) é adicionada ao modelo, a taxa de aprendizado dita o quanto ela vai corrigir o g_i do estimador anterior a ela. Segundo Scikit-learn (2020b), a taxa de aprendizado ν regulariza o modelo da seguinte forma:

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \nu h_m(\mathbf{x}). \tag{2.27}$$

A Figura 2.7 ilustra a atuação de novos h_m (árvores) no modelo Gradient Boosting. Como pode ser visto na figura, novos estimadores tendem a se ajustar sobre os resíduos dos estimadores anteriores. Dessa maneira, a combinação dos estimadores tende a gerar uma $\widehat{f}$ que descreve melhor os dados de treino.

Figura 2.7 – Ajuste sobre os resíduos pelo método Gradient Boosting



Fonte: Géron (2019)

Os hiperparâmetros avaliados pelo processo de validação cruzada neste modelo foram: número de estimadores M; taxa de aprendizagem ν; profundidade das árvores; número mínimo de amostras para dividir um novo nó; número mínimo de amostras para que um nó da folha exista.

2.2.6 Rede Neural Profunda

A partir da década de 1940, despertou-se o interesse em determinar modelos matemáticos e computacionais inspirados na estrutura do sistema nervoso e na capacidade de aprendizagem humana. McCulloch e Pitts (1943) foram dois dos pioneiros na área, desenvolvendo estudos sobre descrição matemática dos neurônios artificias, de modo que pudessem realizar funções lógicas (FACELI et al., 2011). Foram eles também responsáveis por demonstrar que uma rede formada por esses neurônios possuíam grande capacidade de descrever funções complexas (FACELI et al., 2011).

Por mais que outros pesquisadores importantes como Hebb (1949) - pesquisador da capacidade de aprendizado dos neurônios - e Rosenblatt (1958) - criador da teoria dos perceptrons - tenham alcançado grandes feitos na área, as pesquisas em sistemas neurais artificiais foram interrompidas na década de 1970 (FACELI et al., 2011). Entretanto, o aumento da capacidade de processamento dos computadores, o interesse em processamento paralelo e novas propostas de configurações para as redes neurais, contribuíram para que as pesquisas na área retornassem na década de 1980 (GÉRON, 2019).

As Redes Neurais Artificiais se baseiam no funcionamento do sistema nervoso humano. Desse modo, elas são compostas por neurônicos artificias, que são densamente interconectados (FACELI et al., 2011). Estes neurônios são capazes de computar funções matemáticas.

Segundo Faceli et al. (2011), nas arquiteturas das Redes Neurais Artificiais, as conexões simulam as sinapses biológicas. Além disso, as conexões entre neurônios possuem pesos associados, que regulam a atividade do neurônio na rede. Os pesos das conexões são ajustados durante o processo de aprendizado, que nesta monografia se configura como supervisonado. Ademais, estes pesos podem indicar comportamento excitatório, quando forem positivos, e inibitório, quando forem negativos (FACELI et al., 2011).

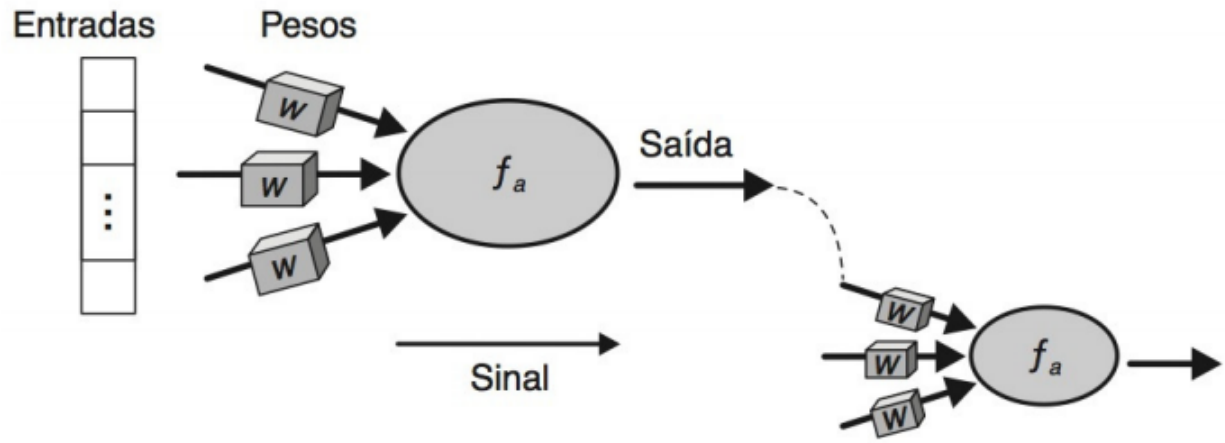
A Figura 2.8 apresenta uma estruturação simples para um neurônio artificial. Na figura, as Entradas representam a admissão dos dados pela unidade lógica apresentada. Após a entrada dos dados no neurônio, uma função matemática (representada por f_a na Figura 2.8) é responsável por ponderá-los e associá-los. A saída do neurônio representa a resposta dada por f_a (FACELI et al., 2011).

Para representar matematicamente as funções nos neurônios, Faceli et al. (2011) escreveu a seguinte expressão para a entrada de um neurônio u:

$$u(\mathbf{x}, \mathbf{w}) = \sum_{j=1}^{d} x_j w_j, \tag{2.28}$$

no qual $\mathbf{x}$ é um objeto com d características, podendo ser escrito como $\mathbf{x} = [x_1, x_2, ..., x_d]$, e $\mathbf{w}$ representa um vetor com os pesos de d terminais de entrada, podendo ser escrito como $\mathbf{w} = [w_1, w_2, ..., w_d]$. Ressaltando o que já foi mencionado, os pesos associados a um neurônio

Figura 2.8 – Estrutura simplificada de um neurônio artificial

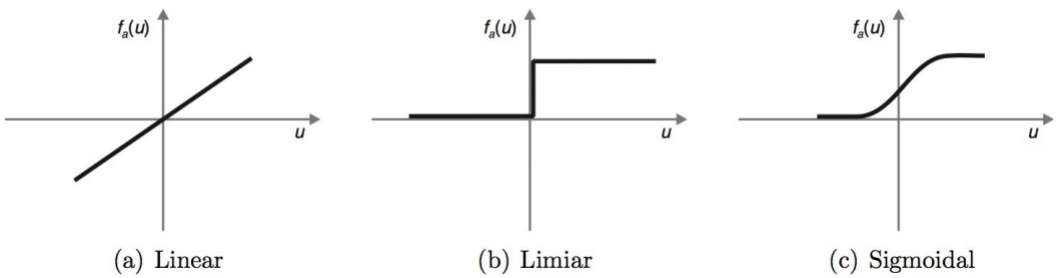


Fonte: Faceli et al. (2011)

podem apresentar sinal positivo, negativo ou nulo. Quando um peso for nulo, implica que a conexão associada a ele é inativa (FACELI et al., 2011).

Conforme apresentado por Faceli et al. (2011), a saída associada ao neurônio é regulada por uma função de ativação f_a. Exemplos dessas funções podem ser elucidados na Figura 2.9. A função Linear, apresentada na Figura 2.9, implica que a saída assume o valor de u. A função Limiar impõe uma saída igual 0 ou 1, é comum também usar uma função limiar que resulte em 1 ou -1. Para a função Limiar, ilustrada na Figura 2.9, o neurônio se torna ativo quando a soma u ultrapassa o limiar ($f_a(u) = 1$) (FACELI et al., 2011). Já a função Signomial, usufrui de diferentes inclinações e representa uma função de ativação contínua e diferenciável (FACELI et al., 2011).

Figura 2.9 – Exemplos de funções de ativação



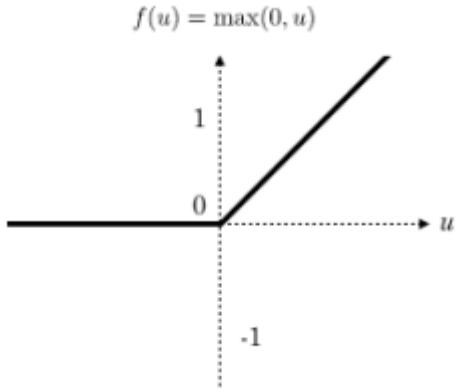
(a) Linear (b) Limiar (c) Sigmoidal

Fonte: Faceli et al. (2011)

Para este trabalho, utilizou-se uma função de ativação conhecida como ReLU, esta função pode ser ilustrada pela Figura 2.10. A função ReLU é frequentemente usada em problemas de

regressão (ALPAYDIN, 2020).

Figura 2.10 – Ilustração da função ReLu

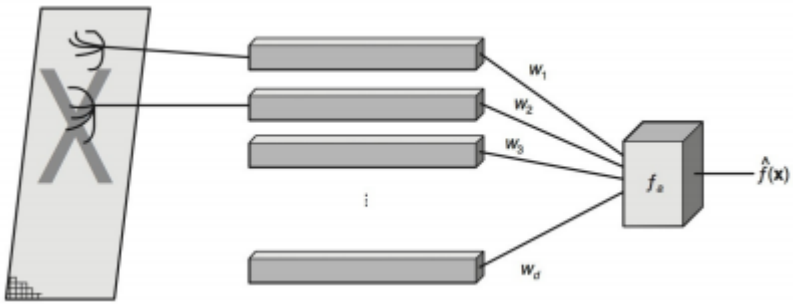


Fonte: Pauly et al. (2017)

A rede perceptron é considera a mais antiga Rede Neural Artificial a ser submetida ao processo de treinamento (GÉRON, 2019). Ela se estrutura em apenas uma camada de neurônio e apesar disso, consegue atingir boa acurácia para diversos problemas de classificação (FACELI et al., 2011).

A Figura 2.11 apresenta a estrutura da primeira rede perceptron, que era composta por apenas um neurônio (FACELI et al., 2011). Ainda, como pode ser visto na Figura 2.11, a rede perceptron possuía uma retina, que era responsável por pré-processar os objetos de entrada (FACELI et al., 2011).

Figura 2.11 – Primeira rede perceptron



Fonte: Faceli et al. (2011)

O treinamento da rede perceptron se baseia na correção do erro de predição, perante os pesos associados ao neurônio. Segundo Faceli et al. (2011), os ajustes nos pesos se dão pela seguinte maneira:

$$w_j(t+1) = w_j(t) + \nu x_i^j (y_i - \widehat{f}(\mathbf{x}_i)), \tag{2.29}$$

no qual $w_j(t)$ é o valor do peso da j-ésima conexão no tempo t, ν é a taxa de aprendizado para este caso, x_i^j representa o valo da j-ésima característica da amostra de entrada $\mathbf{x}_i$ e $\widehat{f}(\mathbf{x}_i)$ é saída dada pela rede, perante a saída esperada y_i no tempo t. Assim como no caso do Gradient Boosting, a taxa de aprendizado representa a expressividade do ajuste no peso $w_j(t+1)$. De acordo com Faceli et al. (2011), a magnitude da taxa de aprendizado infere na velocidade de convergência da rede.

Por mais que as redes perceptrons tenham sido propostas para tarefas de classificação, elas podem ser usadas também para regressão. A grande diferença entre a tarefa de classificação e regressão está na função de ativação. Para tarefas de classificação, a saída deve ser discreta, desse modo, uma função como a Limiar poderia ser empregada. Na regressão a saída $\widehat{f}(\mathbf{x}_i)$ deve ser contínua, assim, uma função de ativação como a Linear poderia ser incorporada. Ambas funções de ativação usadas neste exemplo podem ser encontradas na Figura 2.9.

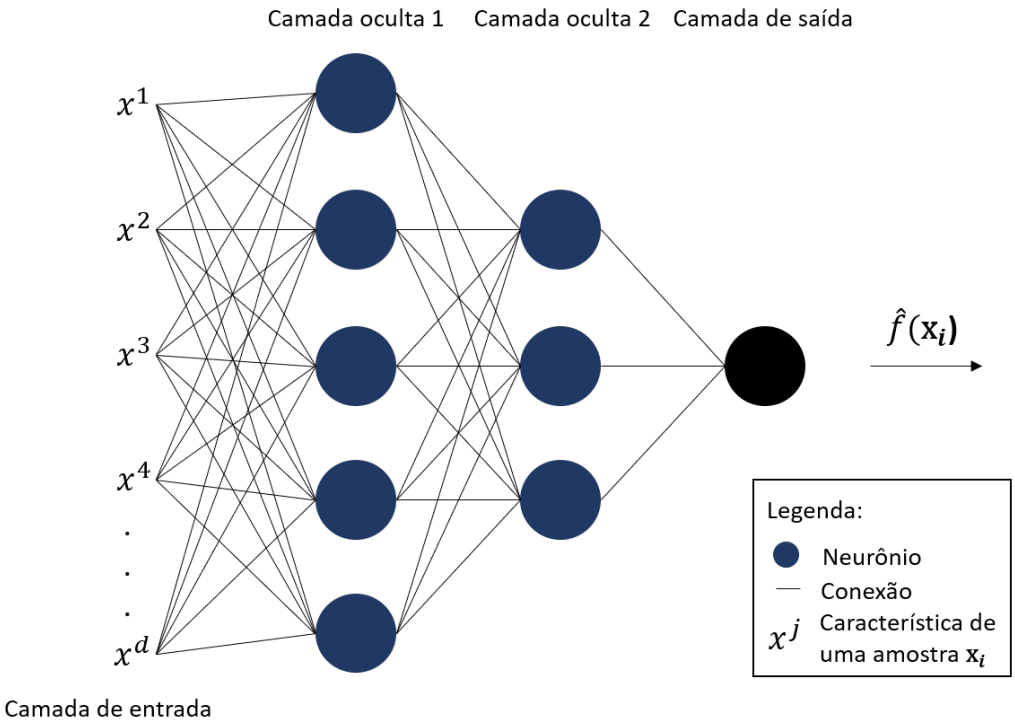
A descontinuidade histórica em pesquisas nesta área, mencionada no início desta seção, se deve à monografia elaborada por Marvin Minsky e Seymour Papert (GÉRON, 2019). Nela, eles destacaram diversas fraquezas nos perceptrons, relacionadas a problemas de modelagem de funções não lineares (FACELI et al., 2011). Entretanto, algumas das limitações impostas por Marvin e Seymour foram contornadas ao se implementar mais camadas de neurônios em redes perpectrons (GÉRON, 2019). O sistema com diversas camadas de perceptrons foi denominado Perceptron Multicamada (*Multi Layer Perceptron* (MLP) em inglês). Além disso, como retratado por Géron (2019), quando uma Rede Neural Artificial possui mais de uma camada oculta - camada entre as camadas de entrada e saída - ela é denominada Rede Neural Profunda.

A Figura 2.12 mostra a configuração mais comum para uma rede MLP. Nessa configuração, todos os neurônios de uma camada l estão conectados a todos os neurônios de uma camada $l+1$ (FACELI et al., 2011). A camada de entrada é composta por todas as características x^j ($j = 1, 2, ..., d$), que compõem determinada amostra $\mathbf{x}_i$, como pode ser observado na Figura 2.12. A camada de saída é arquitetada para cada tipo de tarefa, para problemas de regressão geralmente a camada de saída possui apenas um neurônio, que entrega $\widehat{f}(\mathbf{x}_i)$ como retratada a Figura 2.12.

O treinamento das MLP foi possível graças ao algoritmo *back-propagation*, baseado em gradiente descendente (FACELI et al., 2011). Para a utilização do gradiente, deve-se garantir que a função de ativação seja diferenciável e contínua perante as condições do problema.

O algoritmo *back-propagation* funciona da seguinte maneira: a camada de entrada transmite os dados para a primeira camada intermediária, nesta camada, ocorre a aplicação da função de ativação pelos neurônios, que geram valores de saída, os quais são utilizados pelas próximas camadas como valores de entrada (FACELI et al., 2011). O processo continua até os neurônios da camada de saída produzirem seus valores de saída $\widehat{f}(\mathbf{x}_i)$. Então, $\widehat{f}(\mathbf{x}_i)$ é comparado com o valor desejado y_i. Esta comparação representa o erro da rede perante o objeto $\mathbf{x}_i$ (FACELI et al., 2011).

Figura 2.12 – Rede Neural Profunda para problemas de regressão



Fonte: o Autor

O valor do erro da rede é utilizado para ajustar os pesos de entrada dos neurônios da camada de saída até a primeira camada intermediária (FACELI et al., 2011). A equação a seguir apresenta o ajuste dos pesos para o algoritmo *back-propagation* (FACELI et al., 2011):

$$w_{jl}(t+1) = w_{jl}(t) + \nu x^j \delta_l, \tag{2.30}$$

no qual w_{jl} é o valor do peso entre o neurônio l e a j-ésima característica de uma amostra $\mathbf{x}_i$ ou j-ésimo valor de saída para um neurônio de camadas internas. δ_l representa o erro associado ao neurônio l e x^j o valor da saída de um neurônio ou o valor da característica.

Segundo Faceli et al. (2011), os erros são conhecidos apenas nos neurônios da camada de saída. Desse modo, os demais precisam ser estimados. Nessa perspectiva, os erros dos neurônios das camadas intermediárias são estimados com base nos erros da próxima camada (FACELI et al., 2011). Assim, a estimativa começa do final da rede e termina na primeira camada intermediária. A formulação empregada nas estimativas dos erros, pode ser apresentada a seguir (FACELI et al., 2011):

$$\delta_l = \begin{cases} f'_a e_l & \text{se } n_l \in c_{sai} \\ f'_a \sum w_{lk}\delta_k & \text{se } n_l \in c_{int} \end{cases} \tag{2.31}$$

Na Equação 2.31, n_l representa o neurônio l, c_{sai} a camada de saída, c_{int} a camada intermediária, f'_a a derivada parcial da função de ativação e e_l o erro do neurônio da camada de

saída (FACELI et al., 2011). O e_l possui a seguinte formulação quadrática (FACELI et al., 2011):

$$e_l = \frac{1}{2}\sum_{q=1}^{k}(y_q - \widehat{f}_q)^2. \tag{2.32}$$

A derivada da função de ativação norteia o algoritmo sobre o ajuste dos pesos. Desse modo, se f'_a for positiva, um ajuste deve ser feito no sentido de minimizá-la, uma vez que ela representa o aumento da diferença entre y_i e $\widehat{f}_i$. Se negativa, f'_a representa contribuição para que a saída da rede $\widehat{f}_i$ seja próxima de y_i.

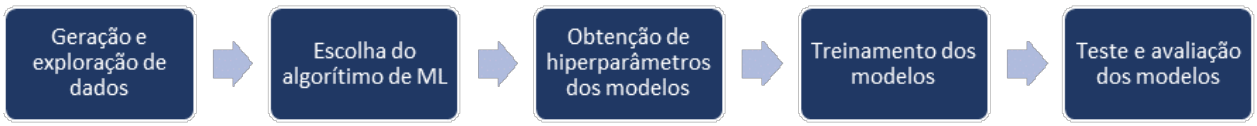
Por fim, com base na apresentação teórica mostrada nesta seção, os hiperparâmetros avaliados nesta monografia para a Rede Neural Profunda foram: número de camadas; número de neurônios por camada; números de dados que passam pela rede em cada ajuste da Equação 2.30; número de vezes que todos os dados passam pela rede, denominado também como número de época. Além disso, adotou-se a f_a mostrada na Figura 2.9 e um $\nu = 0,001$.

3 Metodologia

3.1 Fluxo de trabalho

A metodologia empregada nesta monografia pode ser resumida pelo fluxo de trabalho, representado na Figura 3.1.

Figura 3.1 – Fluxo de trabalho para treinamento e avaliação de um modelo de *machine learning*



Fonte: o autor

O fluxo apresentado na Figura 3.1, apesar de exclusivo para este trabalho, é encontrado de maneira semelhante em trabalhos como o de Mitchell, Michalski e Carbonell (2013). As seções deste capítulo abordam cada uma das etapas apresentadas na Figura 3.1.

A implementação do desenvolvimento metodológico desta monografia e todos os resultados provenientes dela, podem ser encontrados em Bigoto (2020).

3.2 Geração e exploração dos dados

Como mencionado por Géron (2019), a maior parte do tempo em trabalhos de *machine learning* é direcionada à preparação dos dados. A integridade e qualidade dos dados são cruciais para a obtenção de bons resultados em treinamentos de modelos de *machine learning* (MITCHELL; MICHALSKI; CARBONELL, 2013). Assim, esta etapa da metodologia foi executada de forma minuciosa e precisa, uma vez que a base de dados se mostra como elemento fundamental do aprendizado de máquina.

A geração dos dados, empregados no desenvolvimento deste trabalho, foi inspirada na metodologia usada por Hamidieh (2018). A ideia central por trás do artigo de Hamidieh (2018) é a geração dos dados de supercondutores, a partir de suas fórmulas químicas. A associação destes dados com a temperatura crítica dos supercondutores, é usada para o treinamento de modelos de *machine learning*, com base no aprendizado supervisionado.

As fórmulas químicas e as temperaturas críticas dos supercondutores foram extraídas de um banco de dados do Instituto Nacional de Ciência dos Materiais do Japão (NIMS). Este banco

de dados pode ser facilmente acessado em National Institute for Materials Science (2020), após um simples cadastro.

Com as fórmulas químicas e as temperaturas extraídas de National Institute for Materials Science (2020), foram calculados 81 características. Essas características foram calculadas a partir das propriedades dos elementos, que compõem as fórmulas químicas dos supercondutores.

As propriedades elementares envolvidas no processo de construção das características são: Massa Atômica [$u.m.a$]; Primeira Energia de Ionização [KJ/mol]; Raio Atômico [pm]; Densidade [Kg/m^3]; Afinidade Eletrônica [KJ/mol]; Calor de Fusão [KJ/mol]; Condutividade Térmica (C.T.) [$W/(mK)$]; Valência [sem unidades]. Os valores dessas propriedades para cada elemento (átomo) foram obtidos de Mathematica (2020).

Para calcular as características, foram estimados p e w, como sugerido por Hamidieh (2018). O p representa a razão de determinado elemento, com relação a outros em uma fórmula química. Já o w, neste caso, representa a fração de determinada propriedade do elemento, em relação a outros na fórmula química. Além de p e w, estimou-se um coeficiente intermediário em função desses, representado por $f(p, w)$ neste trabalho. Para elucidar o procedimento, Hamidieh (2018) utilizou um exemplo que é descrito no próximo parágrafo.

Para o supercondutor Re_6Zr_1, t representa qualquer uma das 8 propriedades elementares apresentadas anteriormente. Para esse exemplo, supõe-se que t represente a condutividade térmica. Assim, para o Rênio $t_1 = 48W/(mK)$ e para o Zircônio $t_2 = 23W/(mK)$. Dessa forma:

$$p_1 = \frac{6}{6+1} \quad \text{e} \quad p_2 = \frac{1}{6+1}. \tag{3.1}$$

Para w obtêm-se:

$$w_1 = \frac{t_1}{t_1 + t_2} = \frac{48}{71} \quad \text{e} \quad w_2 = \frac{t_2}{t_1 + t_2} = \frac{23}{71}. \tag{3.2}$$

Em função das equações 3.1 e 3.2, têm-se:

$$A = \frac{p_1 w_1}{p_1 w_1 + p_2 w_2} = 0.926 \quad \text{e} \quad B = \frac{p_2 w_2}{p_1 w_1 + p_2 w_2} = 0.074. \tag{3.3}$$

Com base no que foi representado nas equações 3.1, 3.2 e 3.3, Hamidieh (2018) determinou as características baseando-se na Tabela 3.1.

Assim, como elucidado na Tabela 3.1, para cada uma das 8 propriedades elementares são estimadas: média; média ponderada; média geométrica; média geométrica ponderada; entropia; entropia ponderada; intervalo da propriedade (Δ); intervalo ponderado; desvio padrão (σ); desvio padrão ponderado.

Além das 80 características que emergem do parágrafo anterior, o número de elementos em uma fórmula química também é considerado uma característica. Por fim, as 81 características

Tabela 3.1 – Resumo do procedimento utilizado para extração de características a partir da fórmula química dos supercondutores. Apresentação de valores para o Re_6Zr_1

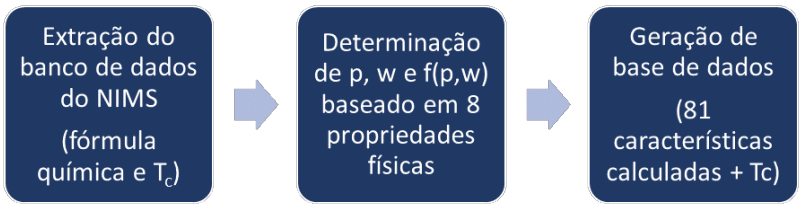
Feature & description	Formula	Sample value
Mean	$=\mu=(t_1+t_2)/2$	35.5
Weighted mean	$=\nu=(p_1t_1)+(p_2t_2)$	44.43
Geometric mean	$=(t_1t_2)^{1/2}$	33.23
Weighted geometric mean	$=(t_1)^{p_1}(t_2)^{p_2}$	43.21
Entropy	$=-w_1\ln(w_1)-w_2\ln(w_2)$	0.63
Weighted entropy	$=-A\ln(A)-B\ln(B)$	0.26
Range	$=t_1-t_2\ (t_1>t_2)$	25
Weighted range	$=p_1t_1-p_2t_2$	37.86
Standard deviation	$=[(1/2)((t_1-\mu)^2+(t_2-\mu)^2)]^{1/2}$	12.5
Weighted standard deviation	$=[p_1(t_1-\nu)^2+p_2(t_2-\nu)^2)]^{1/2}$	8.75

Fonte: Hamidieh (2018)

são reunidas com a temperatura crítica em uma base de dados, usada para treinar os modelos de ML.

O processo de geração de dados, até o presente momento, pode ser resumido pela Figura 3.2. No contexto deste trabalho, elaborou-se um módulo em Python direcionado a realizar de maneira robusta o processo mostrado na Figura 3.2. Assim, foi possível construir funções capazes de tratar e fornecer dados para todas as etapas de desenvolvimento do projeto.

Figura 3.2 – Processo de geração da base de dados usada para treinamento de modelos de *machine learning* neste trabalho



Fonte: o autor

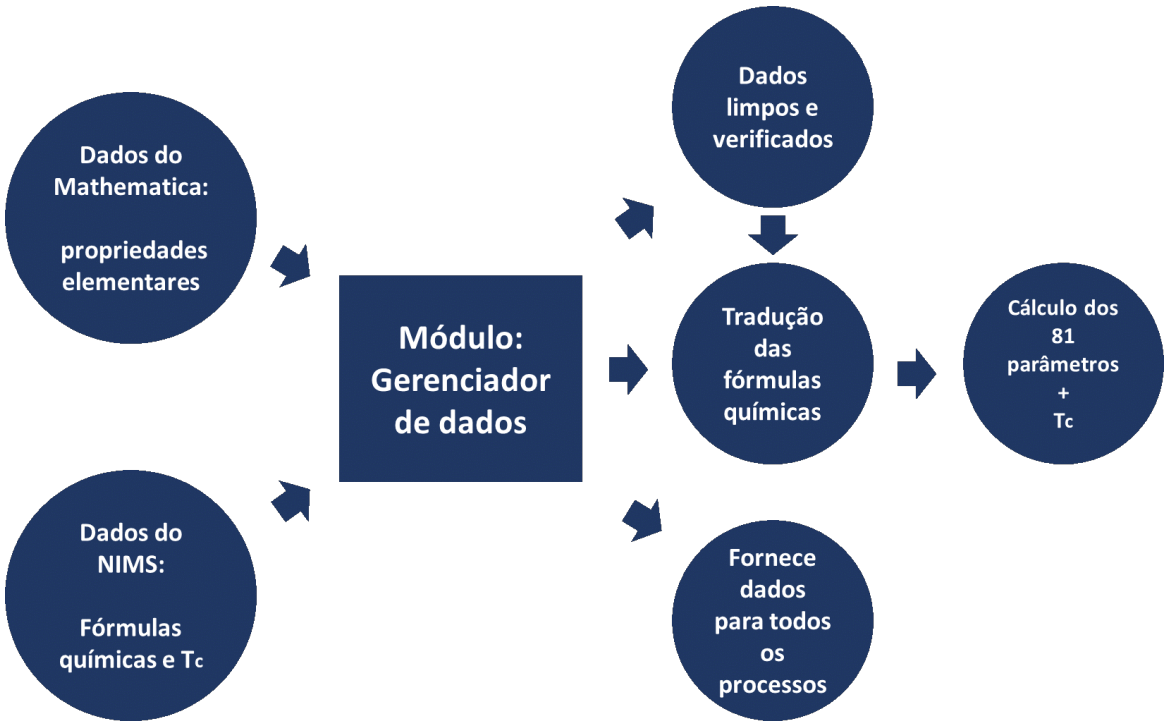
Na etapa de extração do banco de dados do NIMS, foi possível a extração de 33.244 amostras de medidas de supercondutores. Entretanto, elaborou-se no módulo mencionado uma função para validação dessas amostras. A função foi desenvolvida para encontrar erros nas fórmulas químicas dos supercondutores e possíveis incoerências físicas (como identificação de elementos inexistentes ou temperaturas críticas inviáveis). Assim, a função eliminou: campos do banco de dados não preenchidos; valores de temperaturas críticas absurdos (como $T_c >> 200K$); fórmulas químicas que possuíam números de elementos não exatos (como O_{5+X} ou O_{7-Z}); elementos não existentes (como Yo).

Após a execução da função que verifica os dados do banco de dados do NIMS, restaram 21.539 amostras válidas de supercondutores. Dessa forma, elas puderam ser empregadas no

processo de geração de características.

O módulo construído, responsável pelos dados, também possui uma função que é capaz de traduzir uma fórmula química de um supercondutor em algo interpretável. Basicamente, a função interpreta quais elementos estão presentes em uma fórmula química e quais suas proporções. Além disso, ela é responsável por deixar todas as fórmulas químicas em um único formato e verificar possíveis erros de preenchimento.

Figura 3.3 – Esquematização do módulo construído em Python para geração dos dados explicados nessa seção



Fonte: o autor

Com a limpeza dos dados provenientes do banco de dados do NIMS e a possibilidade de leitura das fórmulas químicas de maneira única, foi possível criar uma função que calcula p, w e $f(p, w)$ - representado por A e B na Equação 3.3 - para cada uma das 8 propriedades elementares. Além de calcular esses termos, esta função também calcula as características elucidados na Tabela 3.1 e as integra à característica que representa o número de elementos, constituindo as 81 características citadas. A saída desta função entrega os dados necessários para a construção da base de dados usada no treinamento dos modelos de ML.

O funcionamento deste módulo, feito para tratamento dos dados, pode ser esquematizado pela Figura 3.3. A partir da leitura da figura, é possível ver que o módulo em questão recebe os dados das propriedades elementares e dos supercondutores. Feito isso, o módulo é capaz de limpar e verificar os dados de entrada, padronizar e possibilitar cálculos com fórmulas químicas e fornecer dados à diversas instâncias deste trabalho. Além disso, através da tradução das fórmulas

químicas é possível calcular diretamente as características, que serão usadas como dados de treinamento para os modelos de ML.

Nesse contexto, é interessante citar o Pandas, uma biblioteca criada para a linguagem Python, que visa manipular e analisar dados (IDRIS, 2014). No presente trabalho, o Pandas foi amplamente usado para manipulação dos dados, seja na importação ou exportação em relação aos ambientes do Python. Além disso, ele foi muito empregado em funções e consultas às bases de dados.

Além do Pandas, o Numpy também foi largamente explorado neste trabalho. Ele é uma biblioteca do Python usada na computação científica, principalmente para cálculos matriciais multidimensionais (IDRIS, 2014). Ainda, o Numpy possui diversas funções matemáticas prontas, o que facilita a implementação de modelos matemáticos no Python.

Após a geração dos dados, realizou-se uma análise sobre eles. A análise teve o objetivo de visualizar a distribuição dos dados e levantar indícios de correlação entre eles. A análise usufruiu de recursos gráficos e estatísticos, e a correlação entre os dados foi estimada através do Coeficiente de Correlação Pearson, apresentado na equação abaixo (GÉRON, 2019):

$$r_{x,y} = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n} (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^{n} (y_i - \bar{y})^2}}, \tag{3.4}$$

no qual n é o número de amostras, x_i e y_i são características distintas e $\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$, seguindo o mesmo caminho para $\bar{y}$.

Portanto, a metodologia proposta por Hamidieh (2018), simplificada pela Figura 3.2, foi executada através da criação de um módulo em Python. A esquematização desse módulo é representada pela Figura 3.3. Ademais, a base de dados gerada a partir do módulo descrito nessa seção, alimentou todos os modelos de ML explorados neste trabalho.

3.3 Escolha do algoritmo de ML

Como mostrado no Capítulo 2, modelos de regressão no aprendizado de máquina buscam generalizar uma função capaz de descrever o problema exposto pelos dados. Assim, eles tendem a ajustar uma função genérica aos dados apresentados.

Para a tarefa de regressão, utilizou-se nesta metodologia o aprendizado de máquina fundamentado no aprendizado supervisionado. Salienta-se que o aprendizado supervisionado se baseia em dados conhecidos (rotulados).

Para o treinamento do modelo, determinado supercondutor é representado pelas 81 características, provenientes da sua fórmula química. As 81 características são rotuladas pela temperatura crítica do supercondutor. Assim, os modelos de ML buscam generalizar uma função que associa as 81 características à temperatura crítica.

Os algoritmos de *machine learning*, usados neste trabalho, foram: Regressão Linear Simples, Elastic Net, Máquinas de Vetores de Suporte (SVM), Árvore de Decisão, Floresta de Árvores Aleatórias, Árvores Extremamentes Aleatórias, Gradient Boosting e Rede Neural Perceptron Multicamadas (Rede Neural Profunda).

Os algoritmos do parágrafo anterior estão apresentados em ordem crescente de complexibilidade e robustez. Eles foram implementados nesta ordem, pois ao passar por cada um, foi possível perceber a complexidade do problema em questão e a necessidade da implementação de um algoritmo mais poderoso.

As implementações dos algoritmos Regressão Linear Simples, Elastic Net, Máquinas de Vetores de Suporte, Árvore de Decisão, Floresta de Árvores Aleatórias, Árvores Extremamentes Aleatórias e Gradient Boosting foram feitas pelo Scikit-learn. O Scikit-learn é uma biblioteca de código aberto desenvolvida para aprendizado de máquina no Python (PEDREGOSA et al., 2011). Ele é conhecido por sua fácil e flexível utilização, além de fornecer uma vasta gama de módulos úteis para utilização no âmbito do *machine learning*. Além disso, o Scikit-learn é bastante usado em empresas como Spotify, J.P.Morgan, Booking.com, MARS, Inria e muitas outras (SCIKIT-LEARN, 2020e).

Para implementar o algoritmo de Rede Neural Profunda, utilizou-se o TensorFlow - uma biblioteca de código aberto desenvolvida pelo Google para treinamento de modelos de *machine learning* (GOOGLE BRAIN TEAM, 2020). O TensorFlow é extremamente recomendado para treinamento de Redes Neurais, pois ele possibilita processamento paralelo em GPU. Ele é usado por empresas como Coca-Cola, Google, Intel, Twitter e muitas outras (GOOGLE BRAIN TEAM, 2020).

Os algoritmos implementados pelo Scikit-learn foram processados em uma máquina física - um notebook Dell Inspiron 3583 com 8 GB de memória RAM, processador i7 8ª geração de 2 Ghz e sistema operacional Windows 10 64 bits. Além disso, estes algoritmos foram executados no JupyterLab, uma plataforma que suporta a linguagem Python em uma interface *web* (JUPYTER, 2020).

A Rede Neural Profunda, sob o TensorFlow, foi implementada no Google Colab, um ambiente virtual desenvolvido pelo Google para processamento de dados e treinamento de modelos de ML em Python (GOOGLE COLAB, 2020). O ambiente é alocado em um servidor virtual do Google, onde permite acesso a uma GPU, uma CPU e uma memória RAM de 13 GB. A GPU e CPU possuem velocidades variadas relativas à disponibilidade do servidor em questão.

3.4 Obtenção de hiperparâmetros dos modelos

Como já mencionado, treinar determinado modelo de *machine learning* com todos os dados provenientes da Seção 3.2, poderia conduzir a um sobreajuste (*overfitting*). No qual o

modelo treinado poderia apenas repetir os rótulos dos dados de treinamento.

Para evitar esta situação, é comum segmentar a base de dados em um conjunto de treino e outro de teste. Geralmente, o conjunto de treino corresponde à faixa de 70% a 80% dos dados (GÉRON, 2019). Neste trabalho, os dados envolvidos no processo de treinamento também correspondem a 80% da base de dados, assim, a maior parte dos dados é empregada para treinamento do modelo. O conjunto de teste é usado para avaliar o resultado do treinamento.

Além de segmentar os dados em um conjunto de treino e outro de teste, é possível também controlar os hiperparâmetros dos modelos de ML. O processo de controle dos hiperparâmetros dos modelos é chamado de regularização pela literatura (GÉRON, 2019). O parágrafo a seguir apresenta um exemplo, que mostra a relevância do processo de regularização.

Uma árvore de decisão, se não regularizada, pode crescer indefinidamente. Se não controlada, ela pode se desenvolver ao ponto de se sobrepor aos dados de treino. Dessa maneira, a regularização se torna uma aliada para a generalização, pois ela impõe restrições na construção do modelo. Nesta situação, pode-se controlar a profundidade da árvore, o número de amostras em cada nó ou até mesmo o número mínimo de amostras necessárias para crescer um novo nó.

A Tabela 3.3 apresenta os hiperparâmetros empregados no processo de regularização em cada modelo. Vários hiperparâmetros foram avaliados em cada modelo, entretanto, os que aparecem na Tabela 3.3 mostraram-se mais relevantes no processo de regularização. A Tabela 3.3 também apresenta os valores dos hiperparâmetros avaliados no processo de regularização.

Como pode ser percebido pela Tabela 3.3, o processo de regularização até o presente momento, não mencionou o algoritmo usado no treinamento da Rede Neural Profunda; ele será mencionado mais adiante nesta seção.

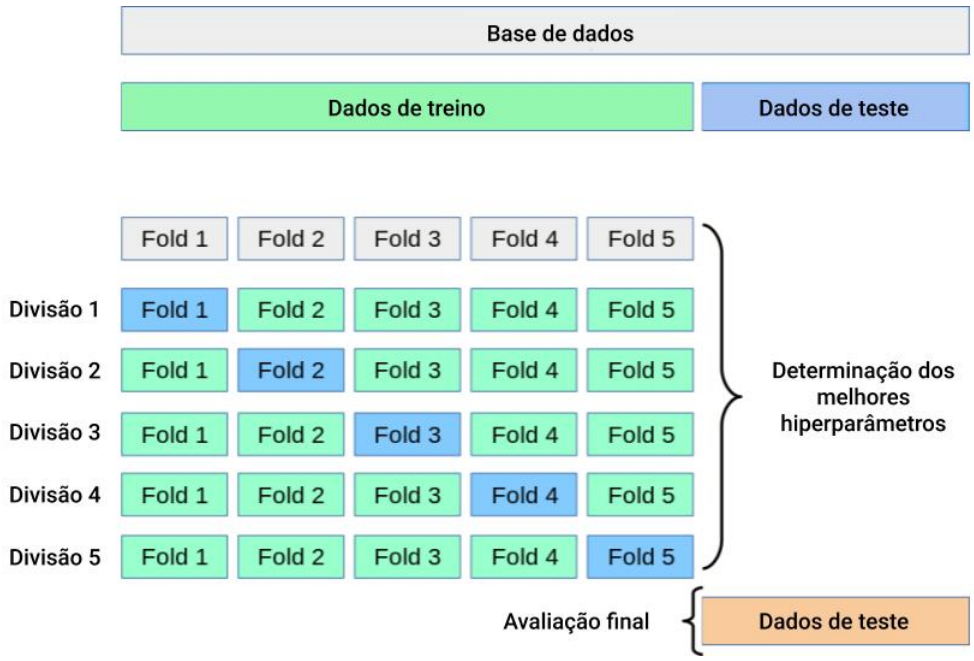
O processo de validação cruzada é usado para determinação dos melhores hiperparâmetros dos modelos apresentados na Tabela 3.3. A Figura 3.4 esquematiza o funcionamento do processo de validação cruzada empregado nesta metodologia, para determinados hiperparâmetros de um modelo de ML.

Analisando a Figura 3.4 é possível perceber que todos os dados gerados através da Seção 3.2 são divididos em conjunto de treino e teste, como mencionado no início dessa seção. O processo de validação cruzada busca manter integro o conjunto de teste, usando somente o conjunto de treino para obtenção dos melhores hiperparâmetros.

Além disso, o processo de validação cruzada divide igualmente o conjunto de treino. No caso representado pela Figura 3.4, o conjunto de treino é dividido em 5 partes iguais. Assim, no processo de treinamento 4 partes são usadas para treinar o modelo e 1 parte para avaliá-lo. Como o conjunto de treino foi dividido em 5 partes, o processo de validação cruzada busca repetir o procedimento de treinamento 5 vezes, alternando a parte dos dados usada para validação.

Para cada conjunto de dados treinado no processo de validação cruzada é gerada uma

Figura 3.4 – Esquematização do processo de validação cruzada usado neste trabalho



Fonte: Adaptado de Scikit-learn (2020a)

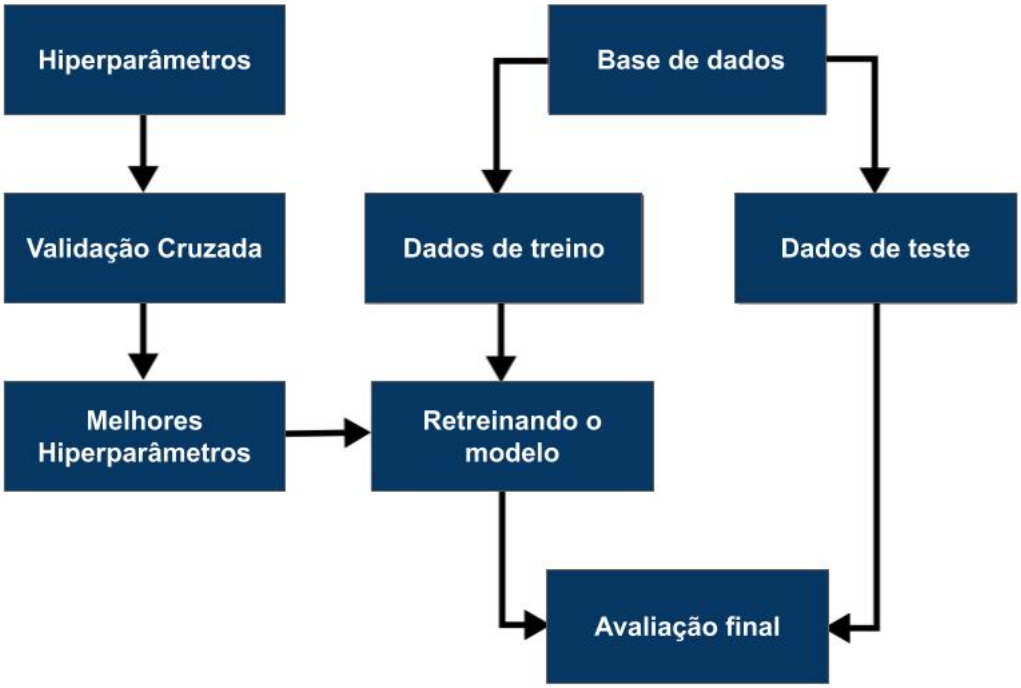
medida de desempenho. Assim, é possível determinar o desempenho médio do modelo em relação a um conjunto de hiperparâmetros. Após a medida de desempenho para determinados valores de hiperparâmetros, outra combinação de hiperparâmetros passa pelo processo de validação cruzada. A combinação com o melhor desempenho carrega os hiperparâmetros adequados para treinar determinado modelo de ML. Neste trabalho, a medida de desempenho envolvida no processo de validação cruzada é o R^2.

Por fim, o conjunto de teste pode ser usado para avaliar o resultado final do treinamento do modelo de ML, que incorpora os melhores hiperparâmetros. Para este trabalho, o conjunto de treinamento também foi dividido em 5 partes no processo de validação cruzada, ao avaliar cada combinação de hiperparâmetros dos modelos da Tabela 3.3.

A Figura 3.5 apresenta o procedimento total empregado na avaliação dos modelos de ML neste trabalho. No procedimento, a validação cruzada encontra os melhores hiperparâmetros e eles são usados para treinar o modelo de ML com todos os dados de treino. Feito isso, o conjunto de teste é empregado para avaliação final do modelo de *machine learning*.

Assim como na Seção 3.2, um módulo em Python foi criado para executar o procedimento apresentado na Figura 3.5. Basicamente, esse módulo é composto por duas funções, uma é responsável por receber os dados gerados pelo procedimento descrito na Seção 3.2 e distribuí-los aleatoriamente em um conjunto de treino e outro de teste. A distribuição aleatória foi empregada no intuito de homogeneizar a distribuição dos dados nos conjuntos, de forma que eles pudessem ter a mesma representatividade sobre o problema.

Figura 3.5 – Procedimento usado para treinamento e avaliação dos modelos de ML



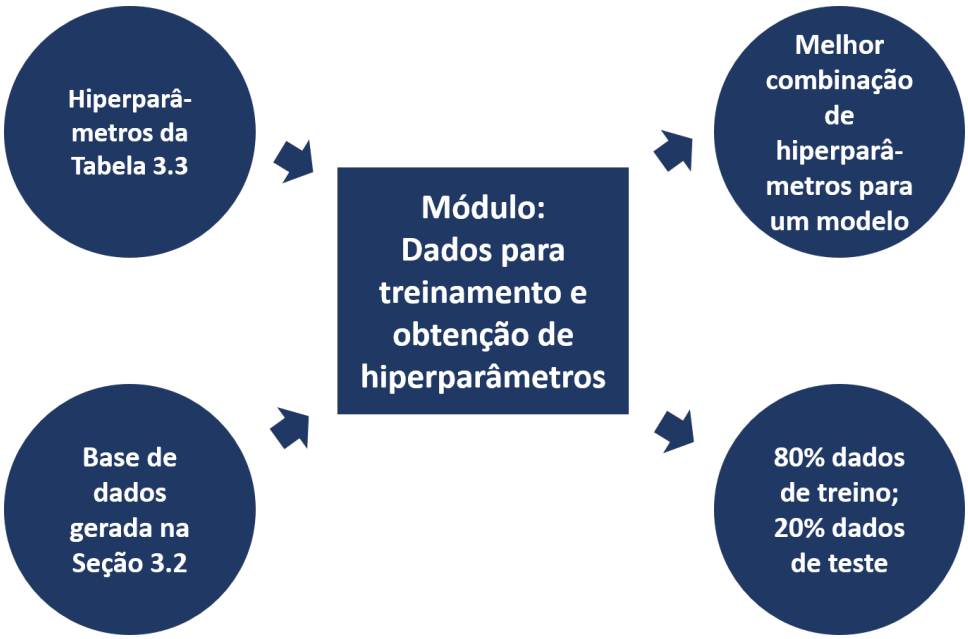
Fonte: Adaptado de Scikit-learn (2020a)

A outra função foi construída para testar todos os hiperparâmetros indicados na Tabela 3.3. Dessa forma, a função faz com que todos eles sejam submetidos ao processo de validação cruzada (Figura 3.4). Após testar todas as possibilidades de combinação dos hiperparâmetros perante determinado modelo, esta função indica os melhores hiperparâmetros com base no R^2. A melhor combinação dos hiperparâmetros é usada para treinar o modelo efetivamente, como já mencionado. A Figura 3.6 resume o funcionamento do módulo em questão.

Além de garantir a aleatoriedade na distribuição dos dados e arquitetar o processo de validação cruzada, o módulo ilustrado na Figura 3.6 permiti controlar o consumo de processamento da CPU. Para este trabalho, configurou-se que todos os núcleos do processador, exceto dois deles, fossem usados nas tarefas de obtenção de hiperparâmetos. Esta configuração foi necessária, pois ao permitir o processamento em todas unidade lógicas, a máquina física - mencionada na Seção 3.3 - se sobrecarregava.

A obtenção de hiperparâmetros para a Rede Neural Profunda exige uma abordagem diferente. Treinar algoritmos de Redes Neurais demanda muito processamento e tempo. Dessa forma, é inviável submeter este modelo de aprendizado de máquina à verificação de diversos valores de hiperparâmetros. Além disso, não há embasamento teórico estabelecido para arquitetar o sistema (hiperparâmetros) da Rede Neural (SILVA; SPATTI; FLAUZINO, 2010).

Figura 3.6 – Esquematização do módulo construído para dividir os dados e encontrar os melhores hiperparâmetros



Fonte: o autor

Apesar de não existir um procedimento consolidado para definição de hiperparâmetros, há preceitos para fazer com que o sistema da Rede Neural trabalhe melhor, evitando sobreajuste nos dados. Para este trabalho, os hiperparâmetros analisados são: número de camadas ocultas da rede; número de neurônios em cada camada; número de etapas de treinamento; quantidade de dados usada para ajuste da rede em cada etapa.

Sabe-se, por Géron (2019), que apenas uma camada oculta pode modelar funções complexas, se possuir um número adequado de neurônios. Por outro lado, Géron (2019) afirma que redes com mais de uma camada possuem eficiência de parâmetros - capacidade de modelar funções complexas com menos neurônios e mais velocidade. É comum o uso de duas camadas ocultas; muitas camadas podem conduzir ao sobreajuste nos dados (SILVA; SPATTI; FLAUZINO, 2010).

Tabela 3.2 – Hiperparâmetros e seus valores analisados para o modelo de Rede Neural Profunda

Modelo	Hiperparâmetros	Valores Analisados
Rede Neural Profunda (*Multi-Modal Perception*)	Número de camadas ocultas da rede	1, 2, 3
	Número de neurônios em cada camada	162, 81, 41, 21, 11
	Número de etapas de treinamento	Parada antecipada em cada caso
	Quantidade de dados usada para ajuste da rede em cada etapa	32, 64, 128

Fonte: o autor

O número de neurônios na entrada e saída da rede é determinado pela necessidade da tarefa

(MITCHELL; MICHALSKI; CARBONELL, 2013). Neste trabalho, a entrada é composta por 81 neurônios, número de características da base de dados. A saída é composta por um neurônio, que entrega o valor da temperatura crítica prevista pela rede. Para os neurônios das camadas ocultas, é comum dimensioná-los para a formação de um funil - cada vez menos neurônios (GÉRON, 2019).

A Tabela 3.2 apresenta os valores de hiperparâmetros testados neste trabalho para a Rede Neural. Para o hiperparâmetro que representa o número de etapas de treinamento, foi implementada uma parada antecipada. Essa parada representa o ponto, no qual o modelo para de reduzir significantemente o erro sobre as predições de T_c. Além disso, a parada evita o início do sobreajuste nos dados.

A quantidade de dados usada para ajuste da rede em cada etapa, foi testada para os valores 32, 64 e 128. Esses números representam a quantidade de dados que passam pela rede, antes do otimizador realizar ajustes nos pesos relacionados aos neurônios (Equação 2.30). Poderiam ser utilizados números maiores, entretanto, o aumento no valor desse hiperparâmetro não apresentou melhoria nas predições. Além disso, o menor valor nesse hiperparâmetro indica maior aproveitamento do conjunto de treino.

O otimizador Adam é utilizado neste contexto, para reduzir o MSE em cada etapa. Além disso, a função de ativação em cada neurônio é a ReLU, por apresentar rapidez no cálculo do gradiente do otimizador (GÉRON, 2019).

O conjunto de treino, que sai do módulo apresentado na Figura 3.6, é usado para o processo de obtenção dos melhores hiperparâmetros da Rede Neural. Dentro deste conjunto, 20% dos dados são usados para validar o processo de aprendizagem.

Apesar da validação cruzada não ser utilizada para obter os melhores hiperparâmetros da Rede Neural, o processo apresentado na Figura 3.5 também pode ser empregado para explicar o ciclo de treinamento da Rede Neural. Entretanto, o que indica o melhor conjunto de hiperparâmetros é o R^2 e o $RMSE$ de cada situação apresentada na Tabela 3.2.

Tabela 3.3 – Hiperparâmetros e seus valores analisados em cada modelo de ML

Modelo	Hiperparâmetros	Valores Analisados
Regressão Simples	Normalização dos dados	*True* ou *False*
Elastic-Net	Coefiente Ridge	0.00001, 0.00001, 0.001, 0.01, 0.1, 1, 10
	Taxa de mistura	0, 0.01, 0.1, 0.5, 1
SVM Polinomial	Grau	2, 3, 4, 5
	C	1, 10, 100, 250, 500
	ϵ	0.01, 0.1, 1, 10
SVM RBF	C	1, 10, 100, 250, 500, 1000, 1500
	ϵ	0.01, 0.1, 1, 10
SVM Linear	C	0.01, 0.1, 1, 10, 50
	ϵ	0.01, 0.1, 1, 10, 50
Árvore de Decisão	Profundidade da árvore	10, 20, 30, 40, 50, 60, 64, 65, 66, 67, 68, 70
	nº mínimo de amostras em um nó	1, 2, 3, 4, 5, 6, 7
	nº mínimo de amostras para dividir um nó	1,3,5,7,9,10,11,12,13,14
Floresta Aleatória	nº de árvores	100, 200, 300, 400, 500, 600, 700, 800, 850
	Profundidade das árvores	10, 20, 30, 40, 50, 60
	nº mínimo de amostras em um nó	1, 2, 3, 4, 5, 6, 7
	nº mínimo de amostras para dividir um nó	1,3,5,7,9,10,11,12,13,14
Árvores Extremamente Aleatórias	nº de árvores	100, 200, 300, 400, 450, 475, 500, 600
	Profundidade das árvores	10, 20, 30, 40, 50, 60
	nº mínimo de amostras em um nó	1, 2, 3, 4, 5, 6, 7
	nº mínimo de amostras para dividir um nó	1,3,5,7,9,10,11, 12,13,14
Gradient Boosting	nº de árvores	100, 150, 200, 300, 400, 500
	Profundidade das árvores	10, 20, 30, 40, 50, 60
	nº mínimo de amostras em um nó	1, 2, 3, 4, 5, 6, 7
	nº mínimo de amostras para dividir um nó	1, 2, 3, 4, 5, 6, 7
	Taxa de aprendizado	0.001, 0.01, 0.1, 0.5

Fonte: o autor

3.5 Treinamento dos modelos

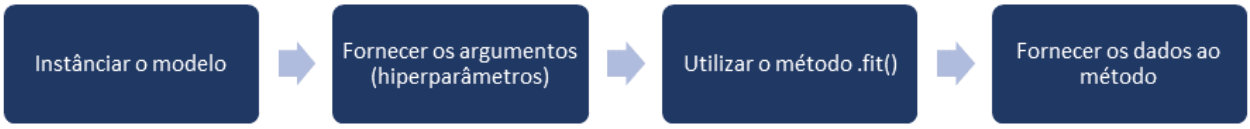
Nesta metodologia, o conceito de treinamento já foi apresentado na seção anterior. Para obtenção dos melhores hiperparâmetros, é necessário treinar os modelos com todas as combinações de hiperparâmetros. A melhor combinação, a que apresenta o melhor desempenho, é usada para treinar de fato o modelo de ML. Como já discutido, a Figura 3.5 apresenta o fluxo total de treinamento de determinado modelo.

Com os dados que emergem do módulo representado pela Figura 3.6, o modelo é treinado utilizando o conjunto de treino (80 % dos dados da base). Após o treinamento com os melhores hiperparâmetros, o conjunto de teste é usado para validar o treinamento e determinar a eficiência do modelo.

É nesta etapa da metodologia, que o modelo de ML escolhido busca reconhecer determinados tipos de padrões nos dados de treino (MICROSOFT, 2020). Assim, esta seção se dedica a descrever o processo de fornecer determinado algoritmo, que com base nos dados vai ajustar o modelo de *machine learning*.

O processo utilizado neste trabalho para implementar (treinar) um modelo de ML em Python, seja usando o Scikit-learn ou o Tensorflow, é meramente ilustrado pela Figura 3.7. Analisando a figura, pode-se perceber que a primeira etapa consiste no instânciamento do modelo. O modelo aqui é evocado como uma classe da biblioteca (Scikit-learn ou Tensorflow).

Figura 3.7 – Etapas para implementar o processo de treinamendo dos modelos de ML em Python



Fonte: o autor

No Tensorflow, antes de chamar o algoritmo do modelo (instânciar a classe), foi necessário criar a arquitetura do sistema da Rede Neural Profunda a ser avaliado. Dessa maneira, foi preciso criar uma classe especificando as camadas e os neorônios por camada. A criação foi intermediada pelo Keras, uma API de alto nível feita para redes neurais, que é executada como *front-end* no TensorFlow (KERAS SPECIAL INTEREST GROUP, 2020).

O segundo passo elucidado na Figura 3.7, representa o momento no qual os melhores hiperparâmetros são inseridos no modelo instânciado. Os hiperparâmetros alimentam os módulos, que constituem o algoritmo do modelo de ML. Após alimentar os algoritmos, o método *fit* é usado para treinar o modelo de *machine learning*. Esse método apresenta os dados de treinamento ao algoritmo.

Para o modelo Máquinas de Vetores de Suporte, as 81 características foram normalizadas como sugerido em Scikit-learn (2020f). A normalização se deve ao fato dos algoritmos das Máquinas de Vetores de Suporte não serem invariáveis à escala. A normalização foi feita a partir da equação a serguir:

$$z = (x - \mu)/\sigma \tag{3.5}$$

no qual x representa o valor de uma característica, μ a média das amostras de treinamento e σ o desvio padrão das amostras de treinamento.

O modelo de Rede Neural Profunda foi normalizado pelo mesmo motivo das Máquinas de Vetores de Suporte, usando a Equação 3.5.

3.6 Teste e avaliação dos modelos

Esta seção aborda a metodologia utilizada para avaliar o processo de treinamento descrito até a seção anterior. Com os modelos treinados, foi possível utilizá-los para realizar previsões. Estas previsões foram feitas utilizando o método *predict*, que funciona de maneira semelhante ao método *fit*. Ao passar dados de determinado supercondutor ao método *predict*, ele foi capaz de prever a temperatura crítica do mesmo, com base no modelo treinado.

Para avaliar os modelos treinados, foram estimados o R^2 e o $RMSE$, como mencionado no Capítulo 2. Para estimá-los, comparam-se os valores das temperaturas críticas provenientes do NIMS, com as temperaturas críticas estimadas pelos modelos de ML.

Nesta monografia, o R^2 e o $RMSE$ foram estimados para o conjunto de treino e para o conjunto de teste, que emergiram do módulo representado pela Figura 3.6. A estimativa sobre o conjunto de treino, revela a eficiência dos modelos na previsão de temperaturas críticas, usando dados já apresentados a eles.

No conjunto de teste, o R^2 e o $RMSE$ são obtidos sobre dados inéditos aos modelos de *machine learning*. Nessa perspectiva, a apuração da eficiência dos modelos sobre os dados de teste foi a mais importante, pois representa a capacidade do modelo em prever a temperatura crítica de supercondutores nunca treinados.

Para cada modelo de ML treinado foi construído um gráfico, que apresenta em um eixo a temperatura crítica proveniente do NIMS e em outro a temperatura crítica prevista pelo modelo. Além da plotagem das temperaturas previstas pelos modelos no gráfico, uma reta representando $R^2 = 1$ foi incorporada a ele. Deve-se mencioanar que os gráficos são construídos a partir das previsões feitas com o conjunto de teste.

Para os melhores modelos - aqueles que apresentaram menor erro (menor $RMSE$) e maior assertividade (R^2 mais próximo de 1) - realizou-se uma análise sobre as previsões em diferentes faixas de temperaturas. Além disso, eles foram utilizados para estimar a temperatura crítica de alguns supercondutores, que aparecem em artigos do Departamento de Engenharia de Materiais da EEL - USP. Esta estimativa permite comparar as temperaturas críticas estimadas pelos modelos, com as temperaturas críticas que aparecem nos artigos.

Como resultado desta metodologia, para cada modelo de *machine learning* são apresentados os melhores hiperparâmetros obtidos, o R^2 e o $RMSE$ para os conjuntos de treino e teste, e o gráfico de temperatura crítica proveniente do NIMS *versus* a temperatura crítica prevista pelo modelo. Ainda, para os melhores modelos são apresentados os resultados decorrentes ao parágrafo anterior.

4 Resultados

4.1 Apresentação dos dados

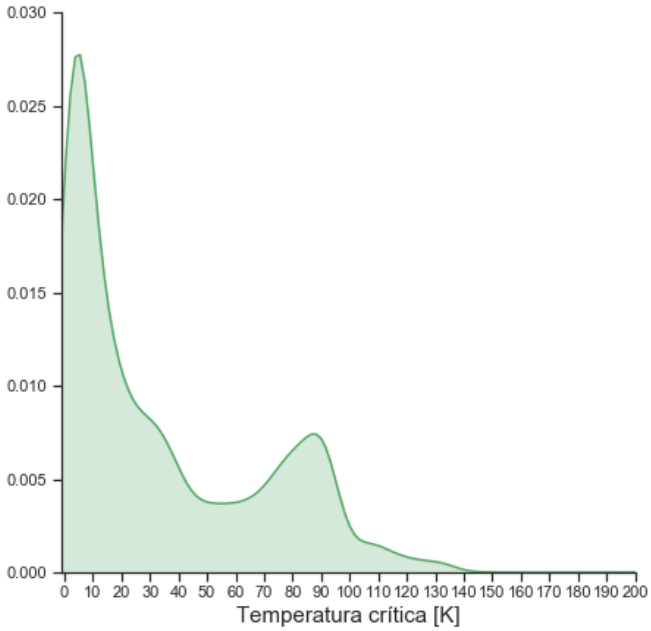
A Tabela 4.1 apresenta um resumo estatístico das temperaturas críticas usadas neste trabalho. Os resultados apresentados na tabela em questão, são provenientes da base de dados descrita na Seção 3.2.

Tabela 4.1 – Resumo estatístico dos dados de temperatura crítica

Contagem	Média	σ	Valor Mín.	1° Quartil	2° Quartil	3° Quartil	Valor Máx.
21539	34,12 K	34,19 K	0,00021 K	5,24 K	19,7 K	62 K	185 K

Fonte: o autor

Figura 4.1 – Distribuição das temperaturas críticas representadas pela Tabela 4.1



Fonte: o autor

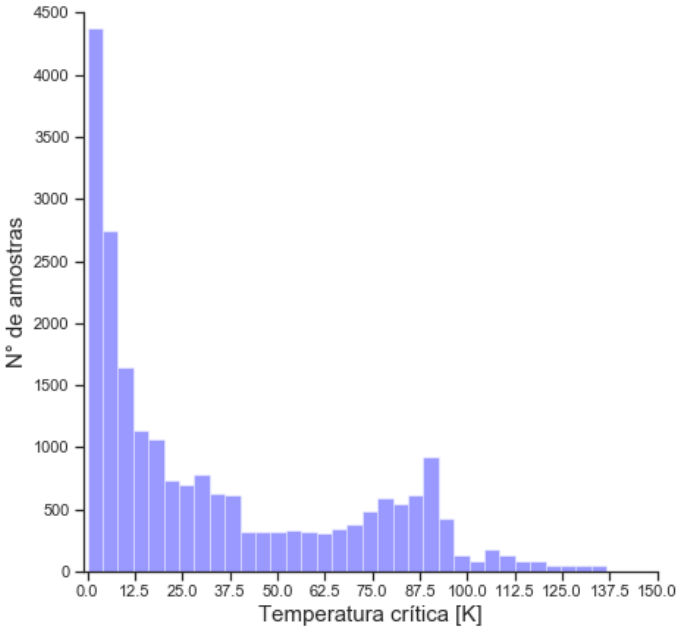
Através da Tabela 4.1, é possível ver que neste trabalho foram utilizadas 21539 medidas de temperaturas críticas de supercondutores. A média dessas temperaturas é de 34,12 K e desvio padrão de 34,19 K. A menor temperatura crítica avaliada neste trabalho foi de $2,1.10^{-4}$ K e a

maior foi de 185 K. Os valores do primeiro, segundo e terceiro quartil são, respectivamente, 5,24 K, 19,7 K e 62 K.

A Figura 4.1 apresenta o resultado de uma função de distribuição para os dados de temperatura crítica. Na figura, é possível ver a concentração dos dados de temperatura crítica em diferentes faixas de temperaturas. Na Figura 4.2, pode-se observar a quantidade de amostras de supercondutores ao longo do eixo da temperatura crítica. Constata-se uma maior concentração de supercondutores em baixas temperaturas, como previa o primeiro quartil da Tabela 4.1.

A Figura 4.3 representa a função de distribuição para o conjunto de teste, que emerge do módulo descrito pela Figura 3.6. Espera-se que o conjunto de teste represente uma curva de distribuição de dados correlata à apresentada na Figura 4.1. Salienta-se que o conjunto de teste é usado para avaliar o resultado final dos modelos, dessa forma, ele deve conter dados representando todas as faixas de temperaturas.

Figura 4.2 – Número de supercondutores em diferentes faixas de temperaturas críticas

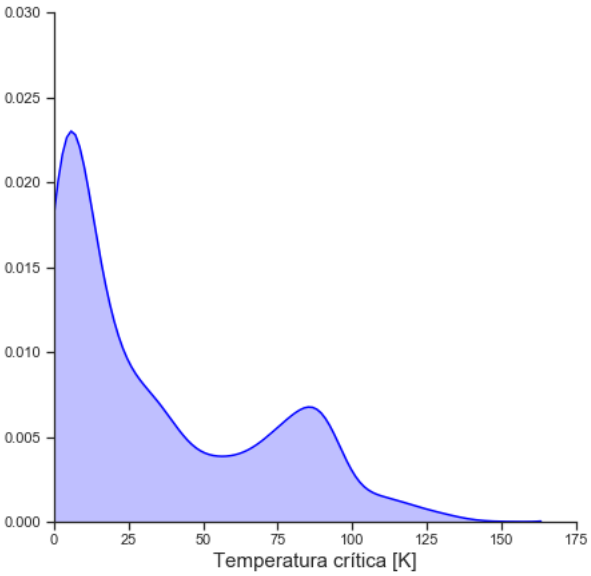


Fonte: o autor

O Coeficientede de Correlação Pearson, elucidado na Equação 3.4, foi calculado para todos as características em relação à temperatura crítica. Os coeficientes mais expressivos são apresentados na Figura 4.4.

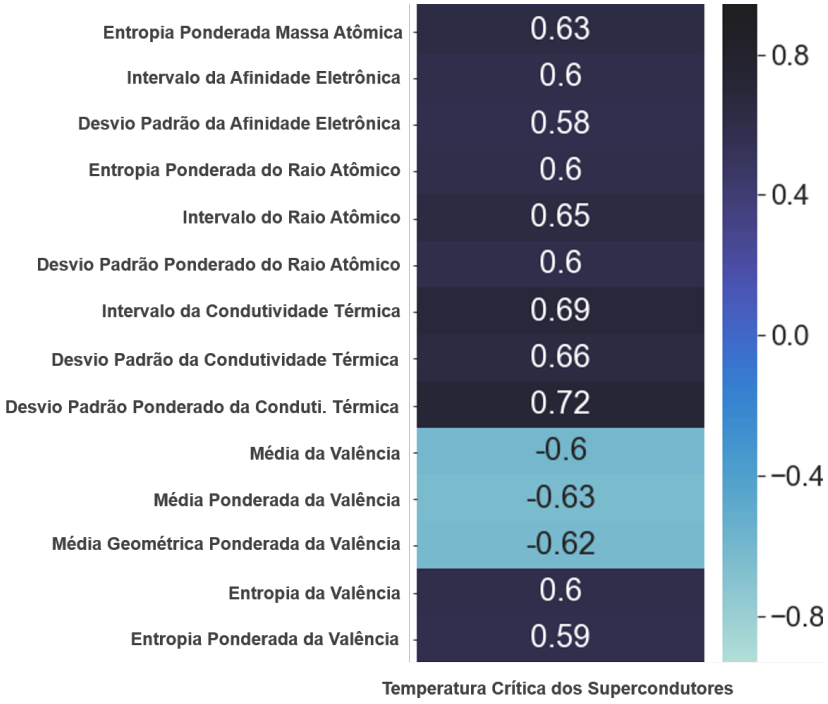
Como pode ser observado na Figura 4.4, as características com maior correlação estão relacionadas à massa atômica, afinidade eletrônica, raio atômico, condutividade térmica e valência. Literaturas como as de Géron (2019) e Bishop (2006) levantam que características fortemente

Figura 4.3 – Distribuição dos dados do conjunto de teste



Fonte: o autor

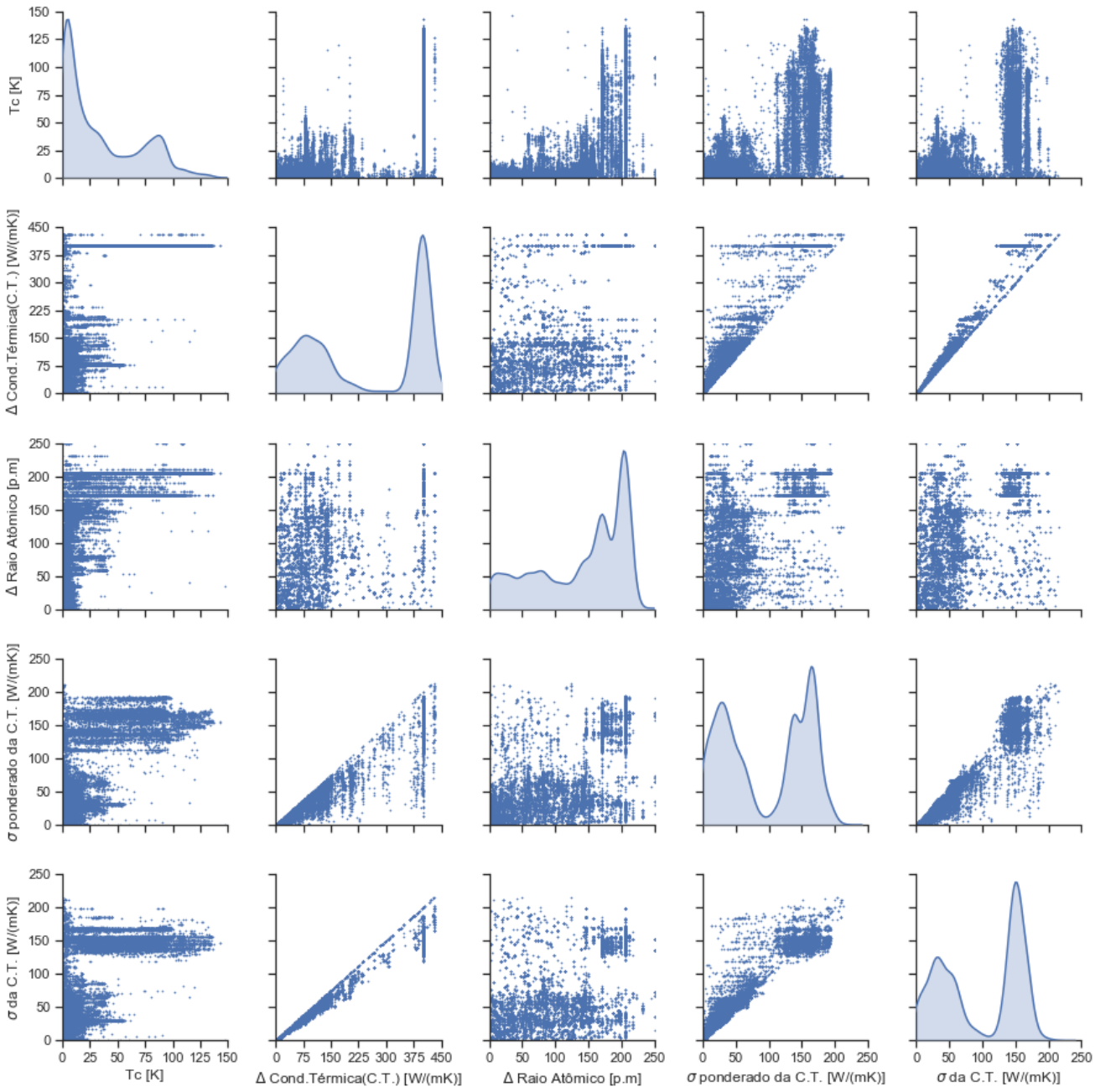
Figura 4.4 – Principais coeficientes de correlação das características em relação à temperatura crítica



Fonte: o autor

correlacionadas costumam contribuir mais com o ganho de informações, no processo de treinamento dos modelos de *machine learning*. A Figura 4.5 exibe a distribuição e o relacionamento das características mostradas na Figura 4.4 e da temperatura crítica.

Figura 4.5 – Gráficos das características com maiores coeficientes de correlação da Figura 4.4



Fonte: o autor

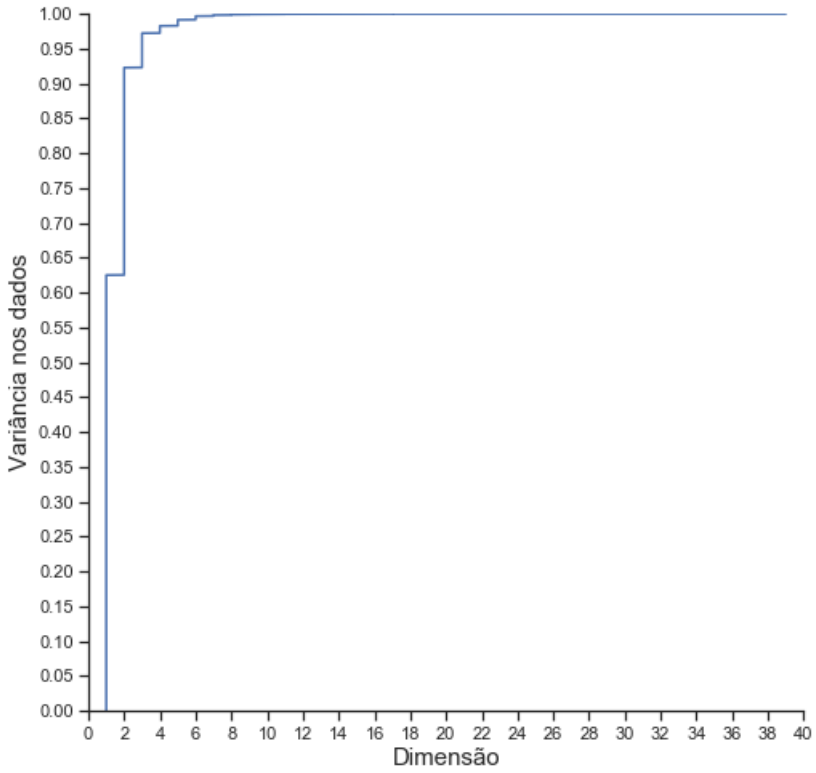
Como muitas características foram calculadas na Seção 3.2, o sistema avaliado pelos modelos de ML possuía 81 dimensões (81 características calculadas). É uma prática comum avaliar

os resultados da redução de dimensionalidade nos dados. Dessa forma, as 81 características usadas para estimar a temperatura crítica foram sujeitas ao processo de redução de dimensionalidade.

O processo da redução se baseou na projeção das 81 características em dimensões inferiores, de forma a garantir a variância nos dados (GÉRON, 2019). O algoritmo PCA (Análise dos Componentes Principais) da biblioteca Scikit-learn foi empregado no processo de redução de dimensionalidade, por possuir eficiência na identificação do hiperplano - usado na projeção - mais próximo aos dados (GÉRON, 2019).

A Figura 4.6 apresenta a variância dos dados em relação à dimensão da projeção. Como pode ser visto, a variância dos dados é preservada mesmo em baixas dimensões. Uma projeção em 6 dimensões preserva uma variância de 99,9%. Os dados com reduções de dimensionalidade foram submetidos aos processos de treinamento. Nenhuma dimensão reduzida apresentou bom resultado no treinamento dos modelos. No modelo de regressão linear múltipla, por exemplo, o R^2 chega a cair 50%, quando a variância é preservada em 99,9%.

Figura 4.6 – Preservação da variância dos dados com mudança na dimensão



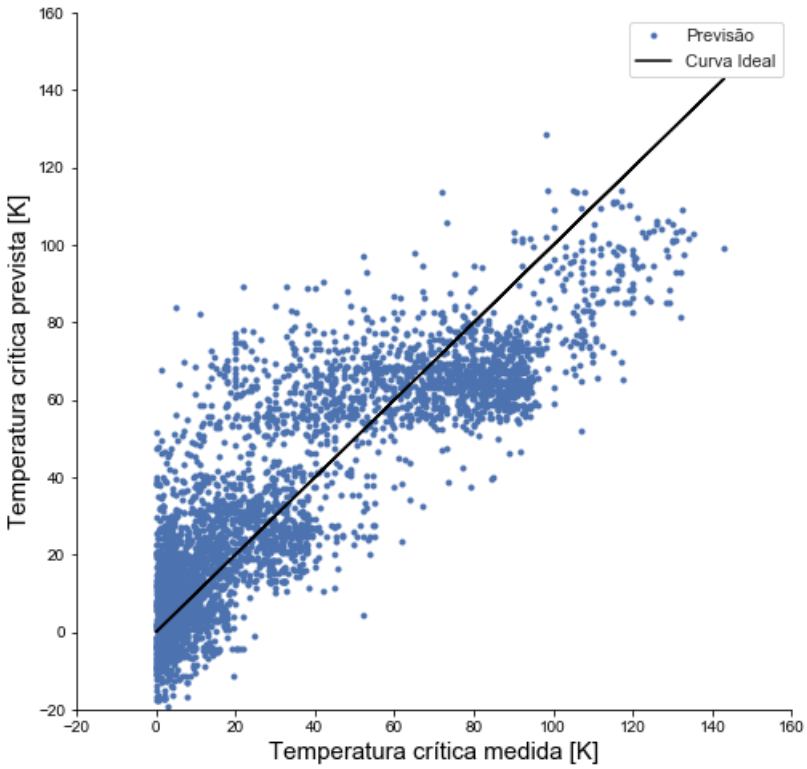
Fonte: o autor

4.2 Regressão linear múltipla

Para a regressão linear múltipla avaliou-se a normalização dos dados. Entretanto, a normalização não mostrou melhoria nos resultados do modelo. Este modelo de regressão linear mostrou R^2 de 0,73 e $RMSE$ de 17,66 K para os dados usados no treinamento. Para o conjunto de teste, o modelo apresentou R^2 de 0,75 e $RMSE$ de 17,03 K. Para os dados normalizados, o modelo obteve exatamente o mesmo resultado.

A Figura 4.7 mostra a temperatura crítica prevista pelo modelo *versus* a temperatura crítica obtida do NIMS. Os dados usados na construção da Figura 4.7 são provenientes do conjunto de teste, com dados nunca vistos pelo modelo. A Curva Ideal, representada no gráfico, refere-se à situação na qual o R^2 seria 1, ou seja, quando o modelo prediz T_c com assertividade de 100%.

Figura 4.7 – T_c prevista *versus* T_c observada para o modelo de regressão linear múltipla, com R^2 de 0,75 e $RMSE$ de 17,03 K



Fonte: o autor

Como é possível ver na Figura 4.7, o modelo em questão prediz para alguns supercondutores, temperaturas críticas menores que 0 K. Esta predição é extremamente inapropriada, pois configura uma incoerência física. O modelo de regressão linear múltipla é utilizado geralmente

como referência para outros modelos de regressão (HAMIDIEH, 2018).

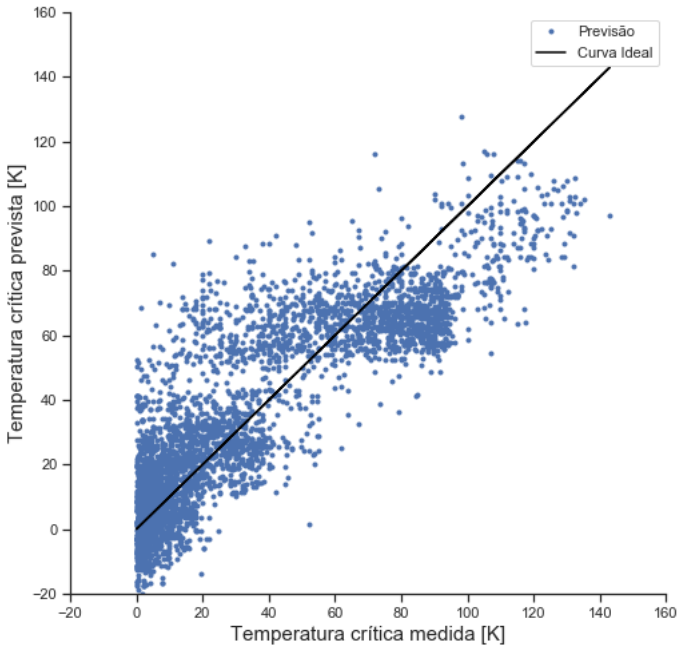
4.3 Regressão Elastic-Net

Os melhores hiperparâmetros encontrados pelo processo de validação cruzada para a regressão Elastic-Net foram $\alpha = 0.0001$ e ρ igual a zero. O hiperparâmetro ρ sendo zero, indica que o modelo Elastic-Net se equivalha ao modelo de regressão de Ridge. Com o hiperparâmetro α sendo pequeno, implica que o modelo tendeu a regularizar pouco os coeficientes da regressão linear.

O R^2 e o $RMSE$ para o conjunto de dados de treino foram de 0,73 e 17,74 K. Para o conjunto de teste, essas medidas de desempenho foram de 0,75 e 17,15 K, respectivamente. Como pode ser percebido, a tentativa de regularizar melhor o modelo de regressão linear múltipla com o modelo Elastic-Net não foi satisfatória, uma vez que não houve melhora nos indicadores de desempenho.

A Figura 4.8 apresenta o gráfico de T_c prevista *versus* T_c observada para o modelo de regressão Elastic-Net, perante o conjunto de teste.

Figura 4.8 – T_c prevista *versus* T_c observada para o modelo de regressão Elastic-Net, com R^2 de 0,75 e $RMSE$ de 17,15 K



Fonte: o autor

Como pode ser visto na Figura 4.8, o problema de predição abaixo de 0 K permanece.

4.4 Máquinas de Vetores de Suporte (SVM)

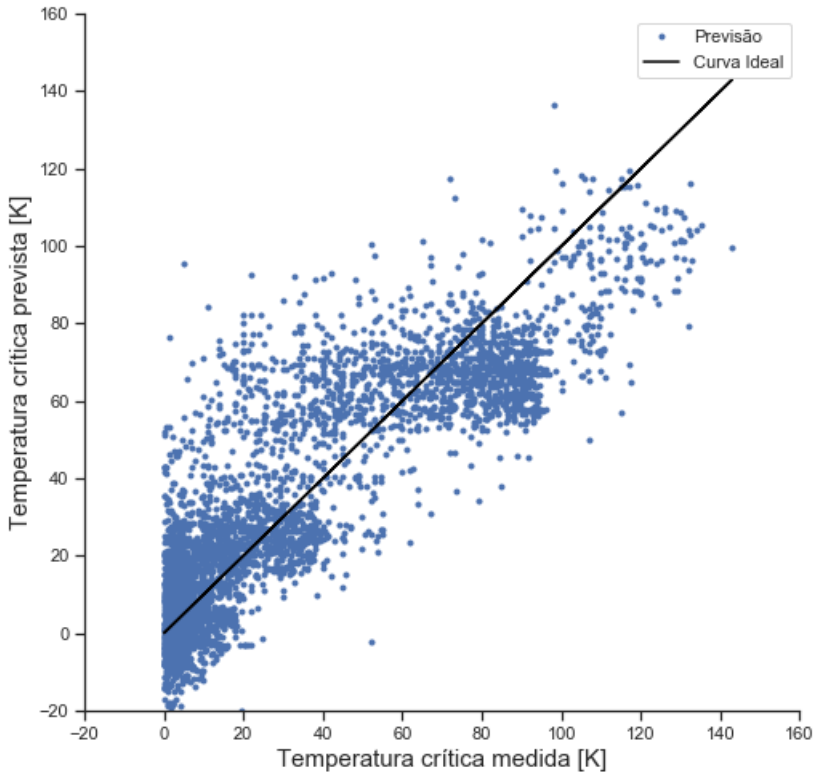
Para as SVM foram utilizados três funções de Kernel voltadas à tarefa de regressão. Foram utilizados o Kernel Linear, Polinomial e o RBF Gaussiano.

4.4.1 Kernel Linear

Os melhores hiperparâmetros para o Kernel Linear foram $C = 100$ e $\epsilon = 10$. O valor de C e ϵ para este caso, se mostra um pouco elevado perante aos indicados por Scikit-learn (2020f) e Géron (2019). Este fato conduz à conclusão que para o modelo obter melhores pontuações para R^2 e $RMSE$, ele se manteve menos regularizado.

O R^2 e o $RMSE$ do conjunto de treino para o Kernel Linear foram 0,73 e 17,84 K, respectivamente. No conjunto de teste, o modelo obteve 0,75 para o R^2 e 17,21 K para o $RMSE$. A Figura 4.9, mostra o gráfico das predições *versus* a temperatura crítica observada.

Figura 4.9 – T_c prevista *versus* T_c observada para o modelo SVM Linear, com R^2 de 0,75 e $RMSE$ de 17,21 K



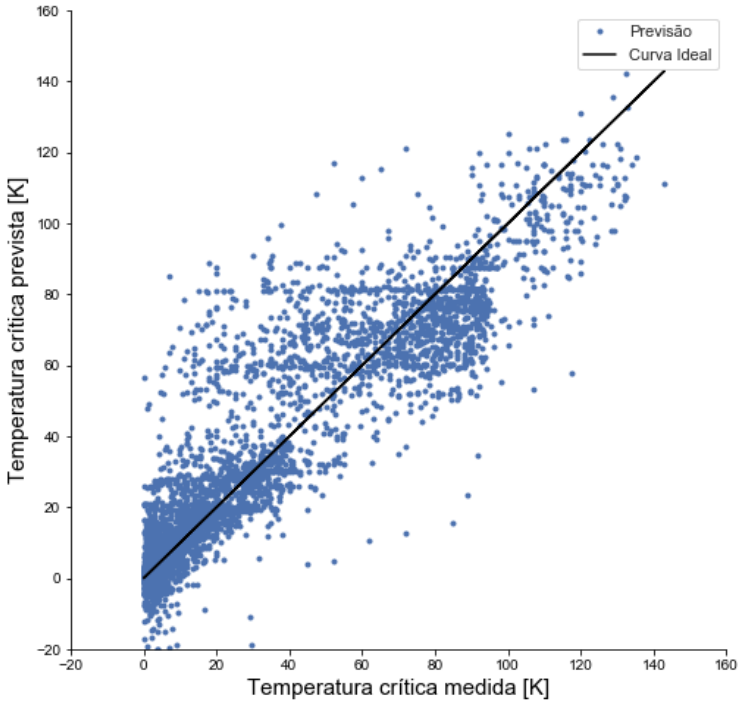
Fonte: o autor

4.4.2 Kernel Polinomial

Os resultados apresentados acima não mostraram ganhos nas medidas de desempenho. Assim, uma abordagem polinomial foi elaborada e implementada através das SVM. Como a dimensionalidade do problema envolvido neste trabalho é grande, uma abordagem polinomial demandaria muito processamento e memória RAM, o que a tornaria inviável. As SVM permitem implementar uma abordagem polinomial sem calcular a combinação de todas as características envolvidas no processo. Elas usam um artifício matemático conhecido como truque de Kernel (GÉRON, 2019).

Os melhores hiperparâmetros do modelo, encontrados pelo método da validação cruzada, foram $C = 500$, $\epsilon = 1$ e grau do polinômio igual a 2. Por mais que o hiperparâmetro C pareça elevado, o modelo não mostrou sobreajuste nos dados, pois o R^2 foi igual para o modelo de treino e teste. O modelo apresentou R^2 igual a 0,82 e $RMSE$ igual a 14,52 K para o conjunto de treino e 14,46 K para o conjunto de teste.

Figura 4.10 – T_c prevista *versus* T_c observada para o modelo SVM Polinomial, com R^2 de 0,82 e $RMSE$ de 14,46 K



Fonte: o autor

A Figura 4.10 mostra o gráfico T_c previsto *versus* T_c esperado para o conjunto de teste. Como pode ser observado, a abordagem polinomial melhorou as medidas de desempenho de

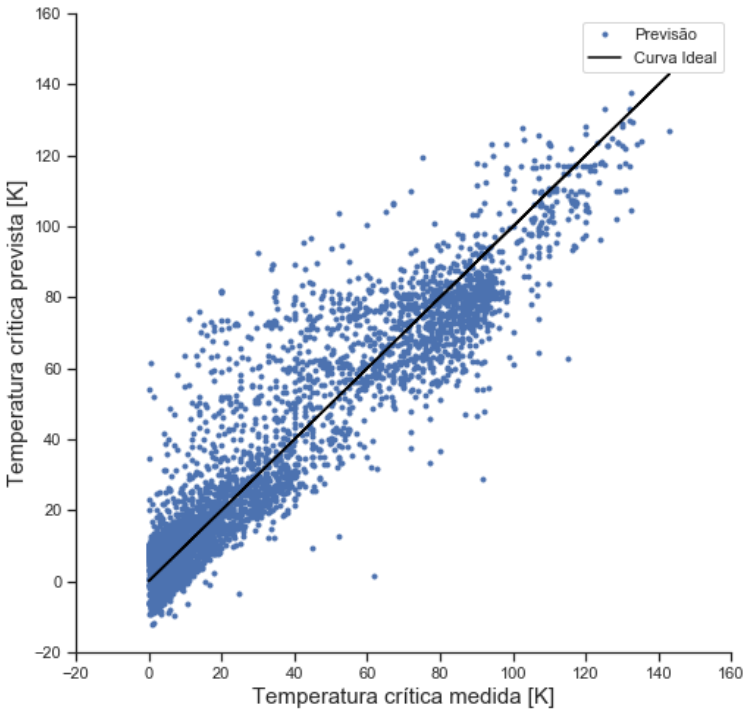
R^2 e $RMSE$. Além disso, na Figura 4.10 é possível encontrar uma melhor distribuição das temperaturas críticas ao longo da Curval Ideal.

Até o presente momento, nenhum modelo conseguiu generalizar bem a predição de T_c em baixas temperaturas. É possível ver nos gráficos de T_c prevista *versus* T_c observada, que os modelos predizem algumas temperaturas abaixo de 0 K, o que é uma incoerência física, como já mencionado. Apesar disso, o ganho no desempenho apresentado nesta subseção conduziu à investigação do modelo de SVM para o Kernel RBF Gaussiano.

4.4.3 Kernel RBF Gaussiano

Para o Kernel RBF Gaussiano, os melhores valores encontrados para os hiperparâmetros foram $C = 1000$ e $\epsilon = 10$. Com base nos hiperparâmetros C e ϵ, espera-se que este modelo esteja menos regularizado do que os demais apresentados. Entretanto, ao observar as medidas de desempenho para este modelo, contata-se que não houve sobreajuste significativo nos dados treino.

Figura 4.11 – T_c prevista *versus* T_c observada para o modelo SVM RBF, com R^2 de 0,87 e $RMSE$ de 12,35 K



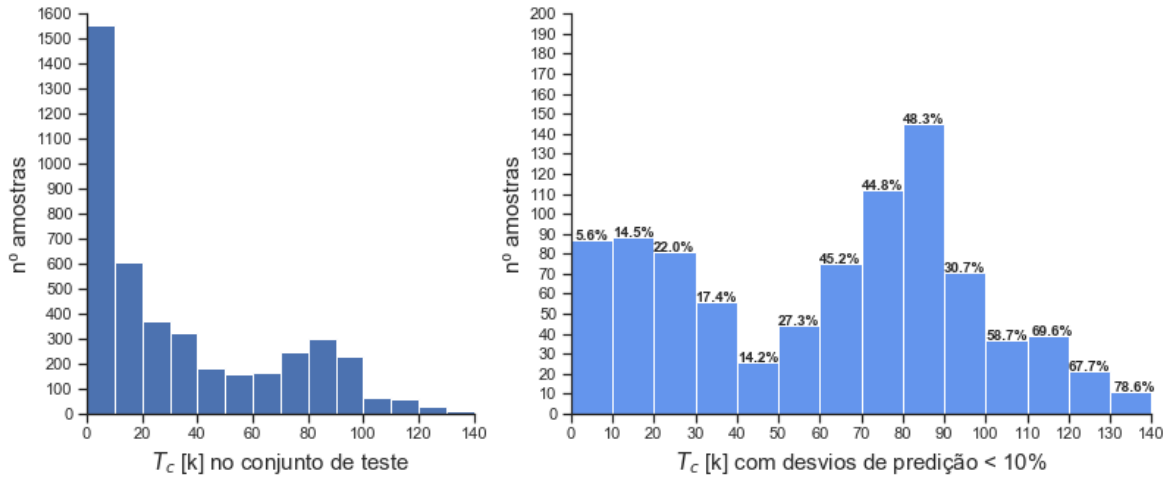
Fonte: o autor

O R^2 e o $RMSE$ para os dados de treino foram respectivamente, 0,88 e 11,81 K. Para os

dados de teste, o R^2 ficou em 0,87 e o $RMSE$ em 12,35 K. A Figura 4.11 apresenta o resultado de Tc prevista *versus* T_c observada.

É possível ver na Figura 4.11, que mesmo melhorando bastante as medidas de desempenho R^2 e $RMSE$, o problema da temperatura ser estimada abaixo de zero Kelvin não foi contornado. Por apresentar bons resultados para R^2 e $RMSE$, é apresentada na Figura 4.12 uma análise sobre a predição de temperaturas críticas em diferentes faixas de temperaturas. A análise destaca nas faixas de temperaturas, as amostras do conjunto de teste que tiveram desvio ($|T_cprevista - T_creal| / T_creal$) menor que 10% no processo de predição.

Figura 4.12 – O gráfico à esquerda mostra a quantidade de amostras no conjunto de teste. O gráfico à direita apresenta o percentual e o número de amostras do conjunto de teste, que apresentaram desvio menor que 10% no processo de predição da temperatura crítica pelo Kernel RBF



Fonte: o autor

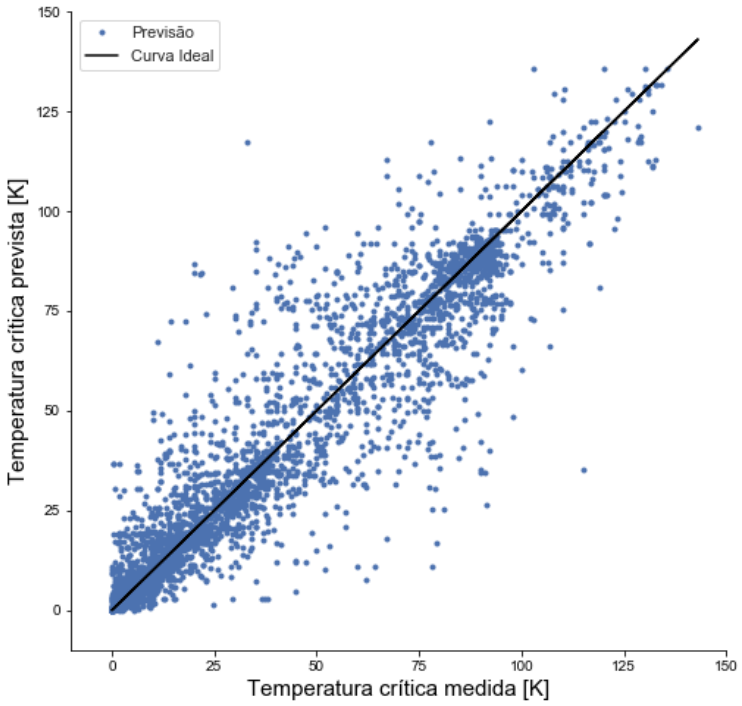
A partir da Figura 4.12 fica evidenciado que o modelo obtém seu pior resultado para as temperaturas críticas entre 0 e 10 K. Nessa faixa de temperatura está a maior parte das amostras do conjunto de teste e apenas 5,6% delas obtiveram desvios inferiores a 10%. Nessa perspectiva, o modelo descreve melhor as temperaturas que estão entre 130 e 140 K. Para esta última faixa de temperatura, o modelo previu 78,6% das temperaturas críticas com desvio menor que 10%.

4.5 Árvore de Decisão

Para o modelo Árvore de Decisão os melhores hiperparâmetros foram: profundidade máxima da árvore igual a 67; mínimo de 13 amostras para que um nó possa se dividir; mínimo de 6 amostras que um nó da folha deve possuir.

Com os hiperparâmetros apresentados, o modelo obteve para o conjunto de treino um R^2 de 0,96 e um $RMSE$ de 7,15 K. Para o conjunto de teste, o R^2 ficou em 0,90 e o $RMSE$ em 11,04 K. A Figura 4.13 apresenta o gráfico de T_c prevista *versus* T_c observada.

Figura 4.13 – T_c prevista *versus* T_c observada para o modelo Árvore de Decisão, com R^2 de 0,90 e $RMSE$ de 11,04 K



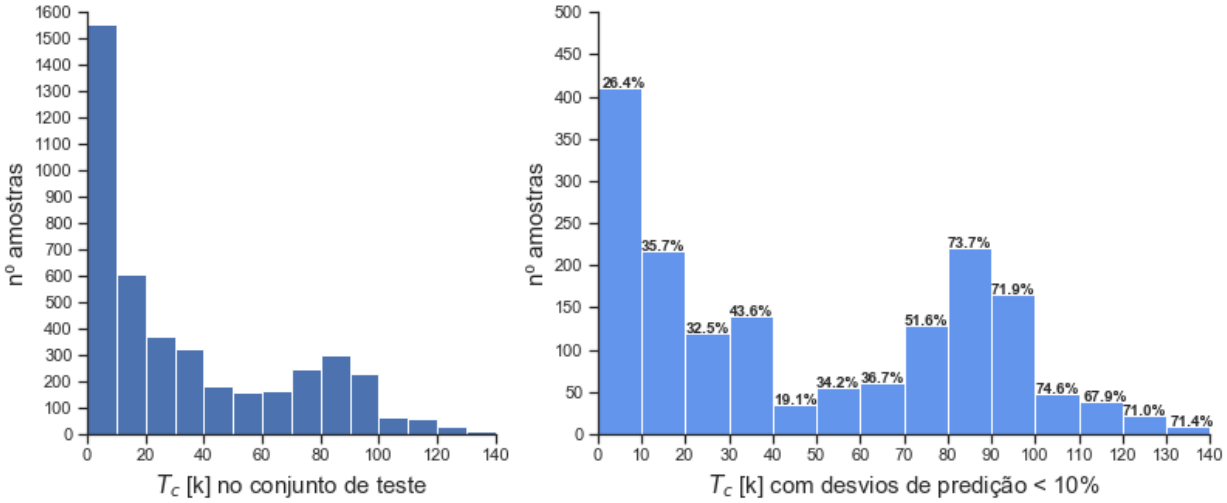
Fonte: o autor

Observa-se que a árvore de decisão conseguiu aproximar as predições à Curva Ideal. Além disso, este modelo de ML conseguiu eliminar o problema das temperaturas críticas negativas. Ademais, o modelo Árvore de Decisão apresentou um ganho significativo nas medidas de desempenho, em relação aos modelos apresentados até o momento.

O modelo Árvore de Decisão descreveu significantemente bem os dados de treino, como pode ser visto nas estimativas de R^2 e $RMSE$. Apesar disso, o modelo conseguiu atingir resultados satisfatórios com os dados de teste, dados nunca vistos por ele. Assim, pode-se concluir que o modelo generalizou bem o problema apresentado.

Assim como apresentado na Figura 4.12 para o Kernel RBF, a Figura 4.14 apresenta uma análise sobre a predição de T_c pelo modelo Árvore de Decisão, em diferentes faixas de temperaturas.

Figura 4.14 – O gráfico à esquerda mostra a quantidade de amostras no conjunto de teste. O gráfico à direita apresenta o percentual e o número de amostras do conjunto de teste, que apresentaram desvio menor que 10% no processo de predição de temperatura crítica pelo modelo Árvore de Decisão



Fonte: o autor

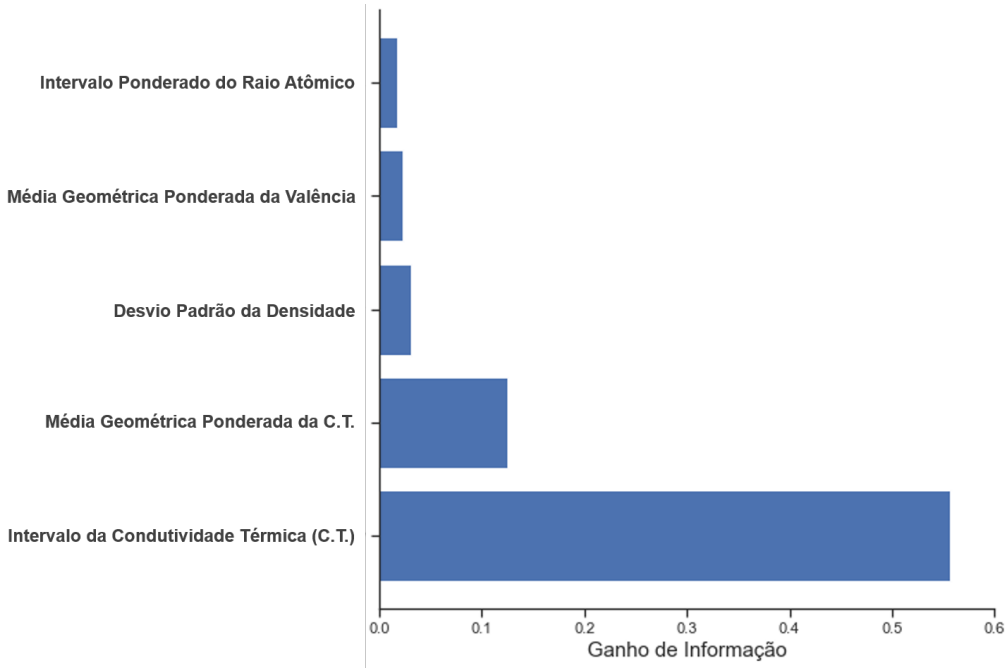
Como pode ser observado na Figura 4.14, a predição na faixa de temperatura de 0 a 10 K melhorou em relação à ilustrada na Figura 4.12. Nesta faixa, o modelo conseguiu prever 26,4% das temperaturas críticas com um desvio menor que 10%. Usando este desvio de 10% como parâmetro, o modelo melhor prediz temperaturas na faixa de 100 a 110 K e pior prediz temperaturas na faixa de 40 a 50 K.

Deve-se salientar, que a análise de desvio de 10% na predição de T_c em diferentes faixas de temperatura é referente aos dados de teste e que o espaço amostral empregado aqui, possui o intuito de comparação entre os modelos adotados nesta monografia.

O modelo Árvore de Decisão, bem como modelos que o utilizam, permite calcular o ganho de informação que cada característica possui. Assim, cada característica usada no treinamento recebe uma pontuação, que representa o quanto ela é importante na obtenção dos resultados. A Figura 4.15 mostra as cinco características que obtiveram maior ganho de informação. O cálculo de ganho de informação por cada característica foi obtido pelo atributo *feature_importances_* das classes dos modelos.

As duas características estatísticas que mais contribuíram para o ganho de informação estão relacionadas à condutividade térmica, como mostrado na Figura 4.15. Apesar da Figura 4.15 apresentar somente as cinco principais características, o modelo calcula o ganho de informação para as 81 características envolvidas.

Figura 4.15 – Características que mais contribuem com o ganho de informação no modelo Árvore de Decisão



Fonte: o autor

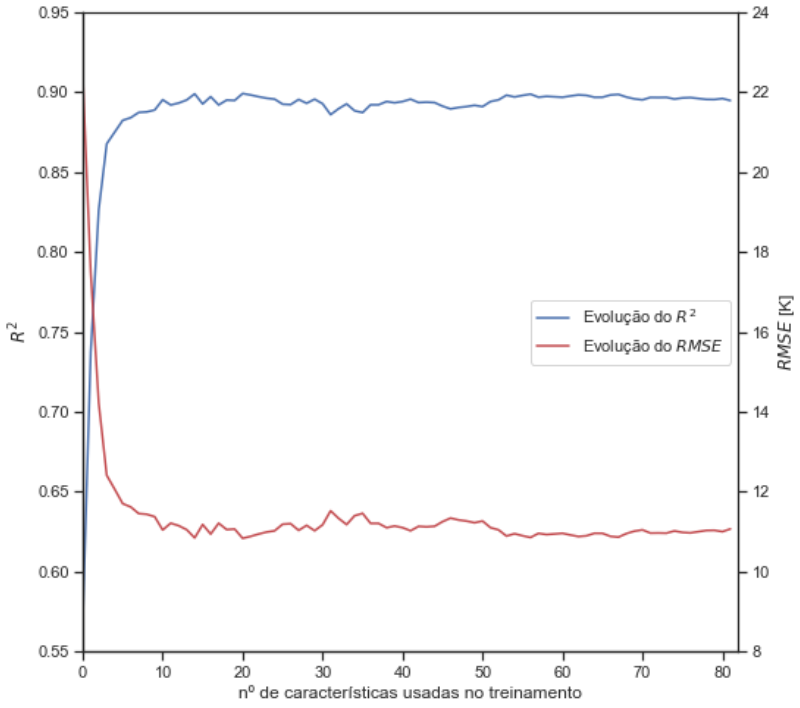
Ordenando as características em ordem decrescente de ganho de informação, é possível plotar a evolução das métricas R^2 e $RMSE$ perante o número de características usadas para treinar o modelo. A Figura 4.16 apresenta no eixo vertical direito valores para R^2 e no eixo vertical esquerdo valores para $RMSE$, perante o número de características utilizados para treinar o modelo. Os valores plotados na Figura 4.16 são referentes aos dados do conjunto de teste.

A Figura 4.16 mostra que as melhores características - as que obtiveram maiores pontuações no ganho de informação - contribuíram mais para os resultados de R^2 e $RMSE$. Assim, utilizar as principais características ou todas as 81 características, conduz a resultados similares em relação ao R^2 e $RMSE$.

Como o modelo apresentou uma maior coerência perante o problema proposto, ele foi utilizado para predizer algumas temperaturas críticas, que aparecem em artigos publicados pelo Departamento de Engenharia de Materiais, da Escola de Engenharia de Lorena (EEL - USP). Para esta tarefa, os seguintes supercondutores foram utilizados: Ti_2GeC; HfV_2Ga_4; $NiTe_2$; Ti_2InC; Nb_2SnC; $Zr_{0.96}V_{0.04}B_2$; Nb_5Ge_3; $Zr_5Pt_3C_{0.3}$.

O Ti_2GeC apresentou supercondutividade a 9,5 K por Bortolozo et al. (2012a). Segundo Ferreira et al. (2018), HfV_2Ga_4 apresenta comportamento supercondutor a 4,1 K. A T_c de 4,0 K para o $NiTe_2$ aparece em Lima et al. (2018), quando dopado com Ti. Bortolozo et al. (2007) apresenta a temperatura crítica do Ti_2InC em 3,1 K. Para o Nb_2SnC, Bortolozo et al. (2006) indica uma T_c de 7,8 K. Renosto et al. (2013) apresenta uma temperatura crítica de 8,7 K para

Figura 4.16 – Evolução de R^2 e $RMSE$ diante do número de características utilizadas para treinar o modelo Árvore de Decisão



Fonte: o autor

o $Zr_{0.96}V_{0.04}B_2$. Para o Nb_5Ge_3, a temperatura crítica é 15,3 K de acordo com Bortolozo et al. (2012b). Por fim, Renosto et al. (2018) apresenta uma T_c de 7 K para o $Zr_5Pt_3C_{0.3}$.

Tabela 4.2 – Comparação das temperaturas críticas estimadas pelo modelo Árvore de Decisão com as encontradas na literatura

Supercondutor	T_c Estimada	T_c Literatura
Ti_2GeC (BORTOLOZO et al., 2012a)	13,2 K	9,5 K
HfV_2Ga_4 (FERREIRA et al., 2018)	10,8 K	4,1 K
$Ti-NiTe_2$ (LIMA et al., 2018)	1,7 K	4,0 K
Ti_2InC (BORTOLOZO et al., 2007)	3,0 K	3,1 K
Nb_2SnC (BORTOLOZO et al., 2006)	6,3 K	7,8 K
$Zr_{0.96}V_{0.04}B_2$ (RENOSTO et al., 2013)	8,2 K	8,7 K
Nb_5Ge_3 (BORTOLOZO et al., 2012b)	2,3 K	15,3K
$Zr_5Pt_3C_{0.3}$ (RENOSTO et al., 2018)	7,0 K	7,0 K

Fonte: o autor

A Tabela 4.2 apresenta os valores das temperaturas críticas estimadas pelo modelo e as

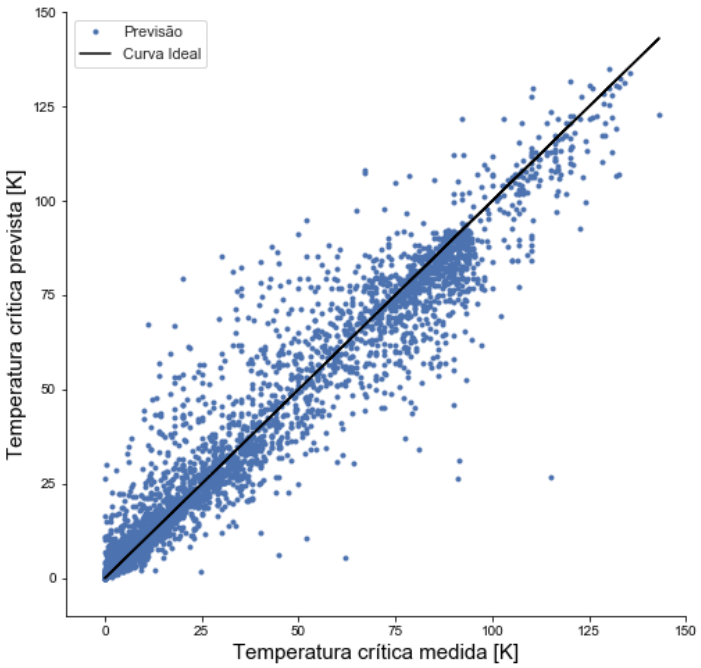
temperaturas críticas encontradas nos artigos citados acima. Analisando a tabela, é possível ver que o modelo pode predizer bem as temperaturas críticas do Ti_2InC, $Zr_{0.96}V_{0.04}B_2$ e $Zr_5Pt_3C_{0.3}$. As temperaturas críticas estimadas para esses supercondutores tiveram um desvio menor que 10%, em comparação às temperaturas críticas encontrada na literatura. Para os demais supercondutores, o modelo não pode predizer T_c de maneira satisfatória.

4.6 Floresta Aleatória

Com relação aos melhores hiperparâmetros, o modelo Floresta Aleatória apresentou 800 árvores de decisão com profundidade máxima de 20. Perante às árvores, elas deveriam possuir ao menos 3 amostras para dividir um novo nó e no mínimo 6 amostras para formar uma folha.

Este modelo apresentou R^2 e $RMSE$ de 0,97 e 5,53 K, respectivamente, para o conjunto de treino. No conjunto de teste, o R^2 ficou em 0,93 e o $RMSE$ em 8,97 K. Com relação ao modelo Árvore de Decisão, que havia apresentado o melhor resultado até o momento, a Floresta Aleatória apresentou aumento do R^2 e diminuição do $RMSE$. A Figura 4.17 apresenta o gráfico de T_c prevista *versus* T_c observada.

Figura 4.17 – T_c prevista *versus* T_c observada para o modelo Floresta Aleatória, com R^2 de 0,93 e $RMSE$ de 8,97 K

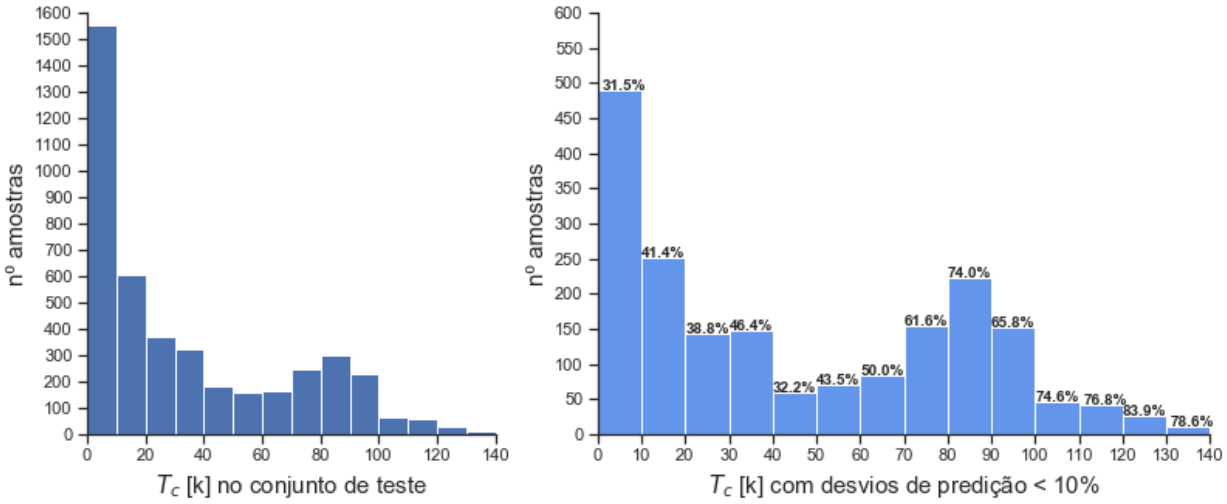


Fonte: o autor

Assim como mostrado na Figura 4.13, a Figura 4.17 revela que o modelo Floresta Aleatória também contornou o então problema de predição em baixas temperaturas. Além disso, a Figura 4.17 mostra uma concentração maior de amostras próximas a Curval Ideal.

A Figura 4.18 mostra uma análise sobre a predição de T_c em diferentes faixas de temperaturas para este modelo. A análise é elaborada com o mesmo intuito da apresentada pela Figura 4.14.

Figura 4.18 – O gráfico à esquerda mostra a quantidade de amostras no conjunto de teste. O gráfico à direita apresenta o percentual e o número de amostras do conjunto de teste, que apresentaram desvio menor que 10% no processo de predição de temperatura crítica pelo modelo Floresta Aleatória



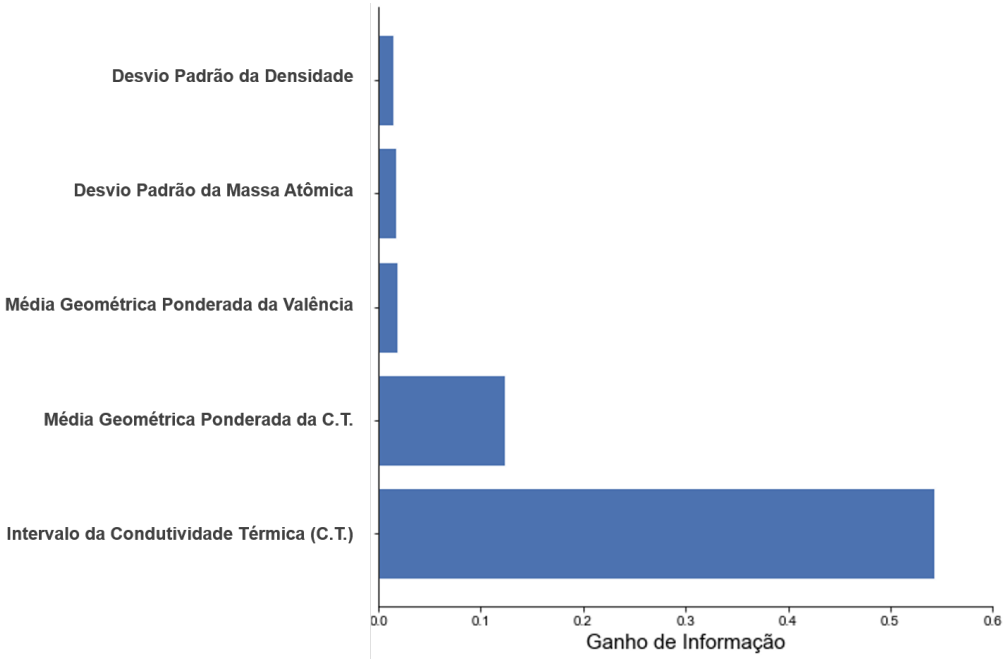
Fonte: o autor

Em comparação à análise feita para o modelo Árvore de Decisão, a Figura 4.18 mostra uma melhoria geral (aumento) nos percentuais, como é ilustrado no gráfico à direita. O aumento significa que o modelo conseguiu prever um maior número de amostras com um desvio menor que 10%.

Além disso, pode-se destacar que o modelo conseguiu predizer 83,9% das temperaturas na faixa de 120 a 130 K, com desvio menor que 10% dos valores medidos em laboratório. Por mais que apenas 32,2% das amostras na faixa de 40 a 50 K tenham desvios menores que 10%, houve uma melhora significativa (aumento de 13,1%) em relação ao modelo Árvore de Decisão.

Como o modelo Floresta Aleatória é constituído de várias árvores de decisão, é possível determinar o ganho de informação para cada característica usada no treinamento. A Figura 4.19 mostra as características de maior relevância, usadas pelo modelo para descrever a temperatura crítica.

Figura 4.19 – Características que mais contribuem com o ganho de informação no modelo Floresta Aleatória



Fonte: o autor

Assim como na Figura 4.15, a Figura 4.19 mostra que as duas características com maiores ganhos de informação estão relacionadas à condutividade térmica. Nos mesmos moldes da Seção 4.5, as características foram ordenadas de maneira decrescente em relação ao ganho de informação. Desse modo, a Figura 4.20 mostra a relação de R^2 e $RMSE$ com a utilização gradativa de características mais importantes.

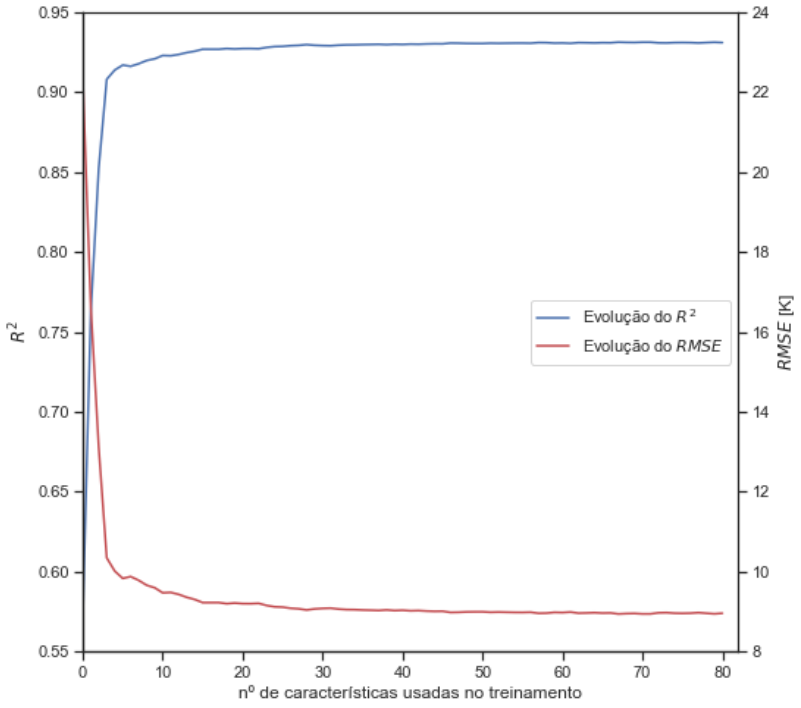
A Figura 4.20 também apresentou que a utilização de poucas características mais relevantes, conduz a resultados similares a utilização de todas as 81 características. Além disso, em comparação à Figura 4.16, a Figura 4.20 apresentou uma evolução menos ruidosa nas curvas de R^2 e $RMSE$. A presença de menos ruídos representa uma maior consistência nas predições, que se baseiam nas características mais importantes.

Como este modelo também se mostrou consistente perante o problema, e apresentou melhoria nas medidas de desempenho R^2 e $RMSE$, ele foi utilizado para predizer as temperaturas críticas dos supercondutores pesquisados no Departamento de Engenharia de Materiais (EEL - USP). Os supercondutores, suas temperaturas críticas observadas e suas temperaturas críticas estimadas pelo modelo em questão, estão representados na Tabela 4.3, assim como na Seção 4.5.

O modelo pode prever a temperatura crítica do Ti_2GeC, $Zr_{0.96}V_{0.04}B_2$, $Ti - NiTe_2$ e $Zr_5Pt_3C_{0.3}$ com desvios menores que 10%, em relação à temperatura crítica encontrada na literatura em questão. Apesar do modelo ter melhorado seu resultado na faixa de temperatura de 0 a 10 K, como mostra a Figura 4.18, as predições dos supercondutores na Tabela 4.3 não

melhoraram significantemente.

Figura 4.20 – Evolução de R^2 e $RMSE$ diante do número de características utilizadas para treinar o modelo Floresta Aleatória



Fonte: o autor

Tabela 4.3 – Comparação das temperaturas críticas estimadas pelo modelo de Floresta Aleatória com as encontradas na literatura

Supercondutor	T_c Estimada	T_c Literatura
Ti_2GeC (BORTOLOZO et al., 2012a)	9,4 K	9,5 K
HfV_2Ga_4 (FERREIRA et al., 2018)	7,3 K	4,1 K
$Ti-NiTe_2$ (LIMA et al., 2018)	4,4 K	4,0 K
Ti_2InC (BORTOLOZO et al., 2007)	4,8 K	3,1 K
Nb_2SnC (BORTOLOZO et al., 2006)	6,5 K	7,8 K
$Zr_{0.96}V_{0.04}B_2$ (RENOSTO et al., 2013)	7,8 K	8,7 K
Nb_5Ge_3 (BORTOLOZO et al., 2012b)	2,0 K	15,3 K
$Zr_5Pt_3C_{0.3}$ (RENOSTO et al., 2018)	6,4 K	7,0 K

Fonte: o autor

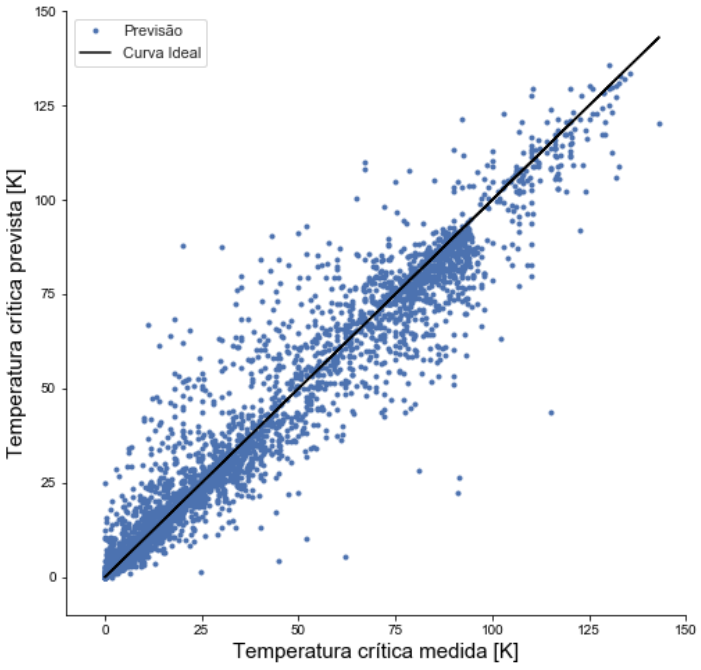
4.7 Árvores Extremamente Aleatórias

Para este modelo, foram utilizadas 475 árvores de decisão. Cada árvore possuía profundidade máxima de 20, requeria no mínimo 2 amostras de dados em uma folha e exigia no mínimo 4 amostras para dividir um novo nó.

Sobre o conjunto de treino, o modelo alcançou um R^2 de 0,98 e um $RMSE$ de 4,90 K. No conjunto de teste, o R^2 foi de 0,94 e o $RMSE$ de 8,72 K. Como pode ser percebido, os valores das medidas de desempenho para o modelo de Árvores Extremamente Aleatórias apresentaram o melhor resultado, até o presente momento.

A Figura 4.21 apresenta o gráfico de T_c prevista *versus* T_c observada para o modelo de Árvores Extremamente Aleatórias.

Figura 4.21 – T_c prevista *versus* T_c observada para o modelo Árvores Extremamente Aleatórias, com R^2 de 0,94 e $RMSE$ de 8,72 K



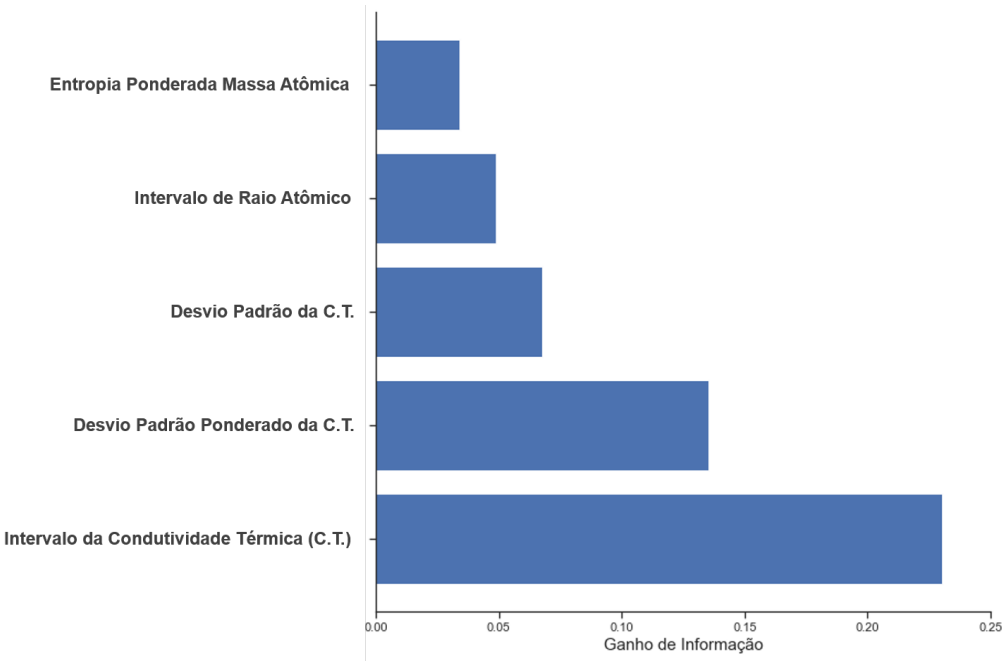
Fonte: o autor

Ao comparar a Figura 4.17 com a Figura 4.21, é evidente que ambas possuem grande semelhança. Isto se deve ao fato destes métodos serem arquitetados de maneira semelhante. A diferença entre eles é que o modelo Árvores Extremamente Aleatórias tende a gerar suas árvores de modo ainda mais aleatório. Assim, esse modelo tente a construir árvores mais diversificadas em relação ao modelo Floresta Aleatória. Além das semelhanças entre a Figura 4.17 e a Figura

4.21, os ganhos perante R^2 e $RMSE$ não são muito expressivos entre esses dois modelos.

Com base no que foi apresentado no parágrafo anterior, a Figura 4.22 ajuda elucidar a diferença entre o modelo Árvores Extremamente Aleatórias e o Floresta Aleatória. Como pode ser visto nela, o modelo atual consegue atribuir mais relevância a outras características. Nessa perspectiva, a pontuação sobre o ganho de informação é melhor distribuída entre as características.

Figura 4.22 – Características que mais contribuem com o ganho de informação no modelo Árvores Extremamente Aleatórias



Fonte: o autor

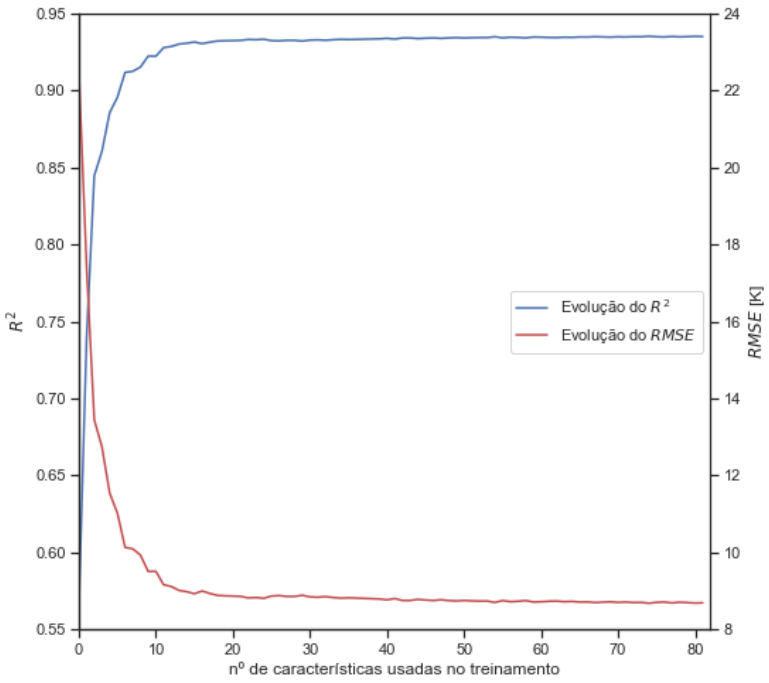
A partir da Figura 4.22 é possível perceber que as três características que mais contribuem com o ganho de informação no modelo, estão relacionadas à condutividade térmica. Além disso, todas as características que aparecem na Figura 4.22 apresentaram os maiores coeficientes de correlação Pearson (Equação 3.4), como pode ser visto na Figura 4.4.

A Figura 4.23 apresenta o gráfico da evolução de R^2 e $RMSE$ em relação ao número de características, ordenadas de forma decrescente em relação ao ganho de informação. A Figura 4.23 também mostra a convergência rápida para os valores de R^2 e $RMSE$. Entretanto, ao comparar a Figura 4.23 com a Figura 4.20, é possível perceber que o modelo Árvores Extremamente Aleatórias converge de maneira menos abrupta, do que o modelo Floresta Aleatória.

Na Figura 4.24 está representada uma análise sobre a predição de temperatura crítica em diferentes faixas de temperaturas. Em todas as faixas de temperaturas, exceto na faixa entre 40 e 50 K, o modelo Árvores Extremamente Aleatórias apresentou um maior percentual de amostras com desvio de predição menor que 10%. Na faixa entre 40 e 50 K, o modelo Floresta Aleatória conseguiu predizer 0,5% a mais de amostras com desvio menor que 10%. Além disso, o modelo

Árvores Extremamente Aleatórias conseguiu predizer 90,3% das amostras com desvio menor que 10%, na faixa de 120 a 130 K.

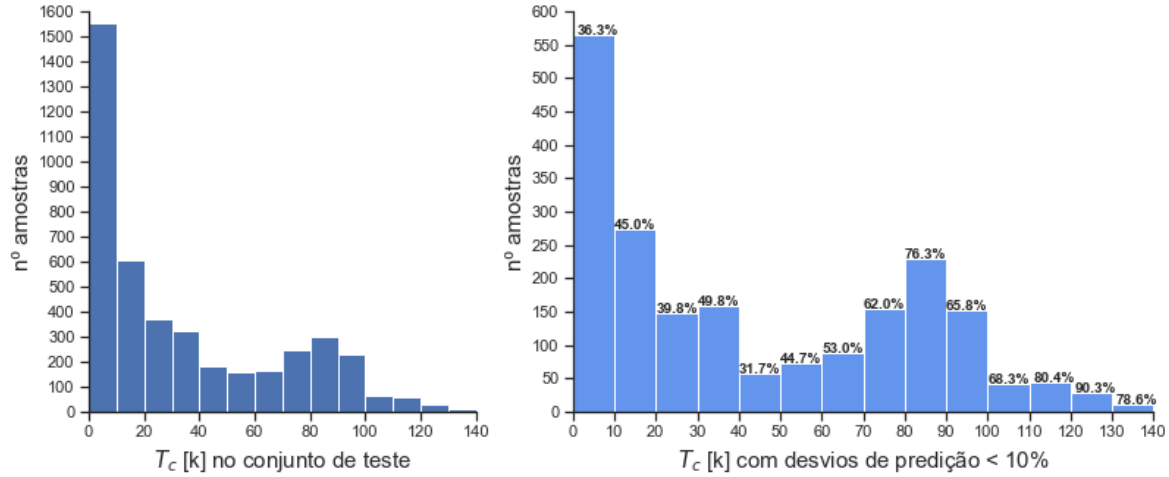
Figura 4.23 – Evolução de R^2 e $RMSE$ diante do número de características utilizadas para treinar o modelo Árvores Extremamente Aleatórias



Fonte: o autor

A Tabela 4.4 mostra os resultados de predição de T_c para os supercondutores introduzidos na Seção 4.5. Ao analisar a tabela, é possível identificar que o modelo conseguiu predizer a T_c do Ti_2GeC e do $Zr_{0.96}V_{0.04}B_2$ com desvio menor que 10%. Apesar do modelo apresentar melhores resultados para o R^2 e o $RMSE$, ele não melhorou significantemente a predição de T_c para os supercondutores apresentados na Tabela 4.4.

Figura 4.24 – O gráfico à esquerda mostra a quantidade de amostras no conjunto de teste. O gráfico à direita apresenta o percentual e o número de amostras do conjunto de teste, que apresentaram desvio menor que 10% no processo de predição de temperatura crítica pelo modelo Árvores Extremamente Aleatórias



Fonte: o autor

Tabela 4.4 – Comparação das temperaturas críticas estimadas pelo modelo de Árvores Extremamente Aleatórias com as encontradas na literatura

Supercondutor	T_c Estimada	T_c Literatura
Ti_2GeC (BORTOLOZO et al., 2012a)	7,7 K	9,5 K
HfV_2Ga_4 (FERREIRA et al., 2018)	6,9 K	4,1 K
$Ti-NiTe_2$ (LIMA et al., 2018)	7,7 K	4,0 K
Ti_2InC (BORTOLOZO et al., 2007)	3,9 K	3,1 K
Nb_2SnC (BORTOLOZO et al., 2006)	6,0 K	7,8 K
$Zr_{0.96}V_{0.04}B_2$ (RENOSTO et al., 2013)	7,9 K	8,7 K
Nb_5Ge_3 (BORTOLOZO et al., 2012b)	1,4 K	15,3 K
$Zr_5Pt_3C_{0.3}$ (RENOSTO et al., 2018)	5,8 K	7,0 K

Fonte: o autor

4.8 Gradient Boosting

Para o modelo Gradient Boosting os melhores hiperparâmetros foram: 150 árvores de decisão; profundidade das árvores igual a 10; mínimo de 6 amostras em cada folha; mínimo de 5 amostras para um nó se dividir; taxa de aprendizado de 0,1.

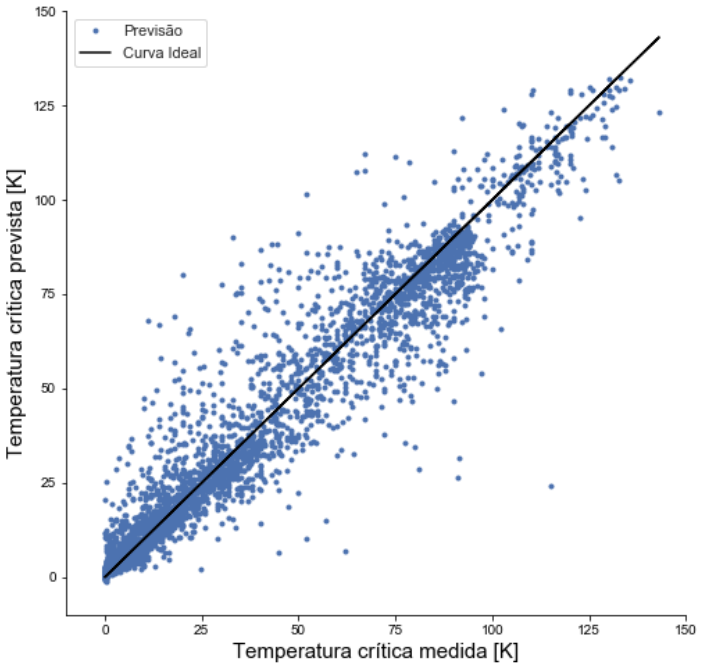
Com relação aos modelos anteriores, que envolviam a construção de árvores de decisão, o modelo Gradient Boosting se mostrou mais regularizado. Este modelo utilizou menos árvores para descrever o problema de regressão proposto. Além disso, as árvores envolvidas tiveram

profundida reduzida e exigiram mais das amostras para seu desenvolvimento, em relação aos modelos Árvores Extremamente Aleatórias e Floresta Aleatória.

Para o conjunto de treino, o modelo Gradient Boosting atingiu um R^2 de 0,98 e um $RMSE$ de 5,02 K. Perante os dados de treino, o modelo alcançou um R^2 de 0,93 e um $RMSE$ de 8,97 K. Em relação ao modelo Árvores Extremamente Aleatórias, o qual apresentou melhores R^2 e $RMSE$ até o momento, o modelo Gradient Boosting apresentou resultados inferiores. Entretanto, por mais que os resultados sejam inferiores ao modelo Árvores Extremamente Aleatórias, as diferenças nas medidas de desempenho não foram significativas.

A Figura 4.25 mostra o gráfico de T_c prevista *versus* T_c observada para o modelo Gradient Boosting. Ao observar a Figura 4.7, é possível encontrar semelhanças entre ela e as Figuras 4.17 e 4.21. Assim, pode-se mencionar que o modelo Gradient Boosting aproxima suas predições à Curva Ideal de forma correlata aos modelos Árvores Extremamente Aleatória e Floresta Aleatória.

Figura 4.25 – T_c prevista *versus* T_c observada para o modelo Gradient Boosting, com R^2 de 0,93 e $RMSE$ de 8,97 K

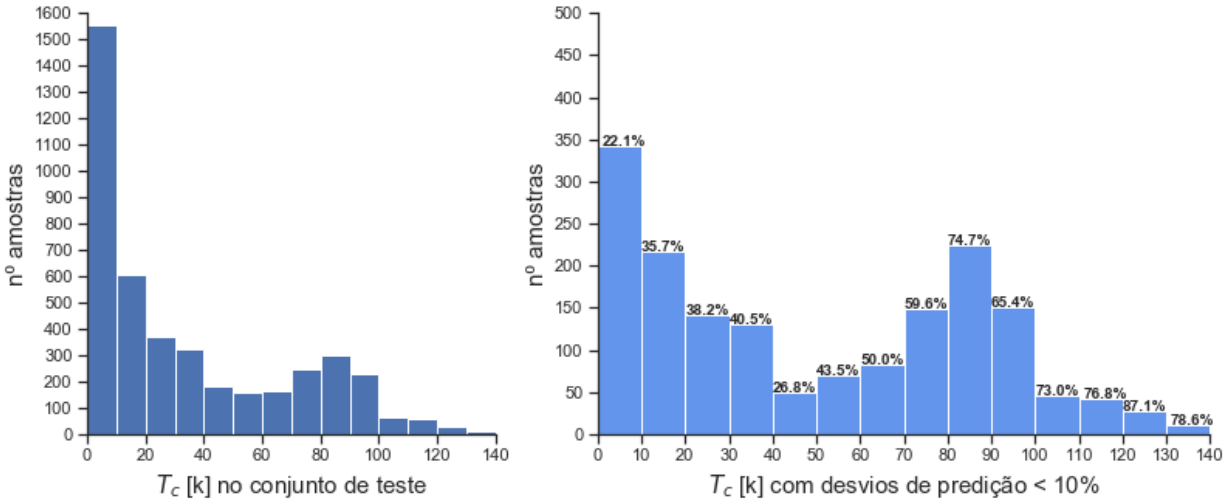


Fonte: o autor

A Figura 4.26 apresenta a análise sobre a predição de temperatura crítica em diferentes faixas de temperaturas, empregada neste trabalho. Com relação aos modelos que empregam árvores de decisão, a análise da Figura 4.26 permite afirmar que o modelo alcançou a pior predição para as temperaturas entre 0 e 10 K.

Comparando com os modelos anteriores, o modelo Gradient Boosting obteve resultados de predição semelhantes ao modelo Floresta Aleatória. Entretanto, o modelo Árvores Extremamente Aleatória apresentou um melhor desempenho em todas as faixas de temperaturas. Por fim, pode-se destacar para o modelo Gradient Boosting a predição de 87,1% das temperaturas críticas na faixa de 120 a 130 K, com desvio menor que 10% em relação às temperaturas observadas.

Figura 4.26 – O gráfico à esquerda mostra a quantidade de amostras no conjunto de teste. O gráfico à direita apresenta o percentual e o número de amostras do conjunto de teste, que apresentaram desvio menor que 10% no processo de predição de temperatura crítica pelo modelo Gradient Boosting



Fonte: o autor

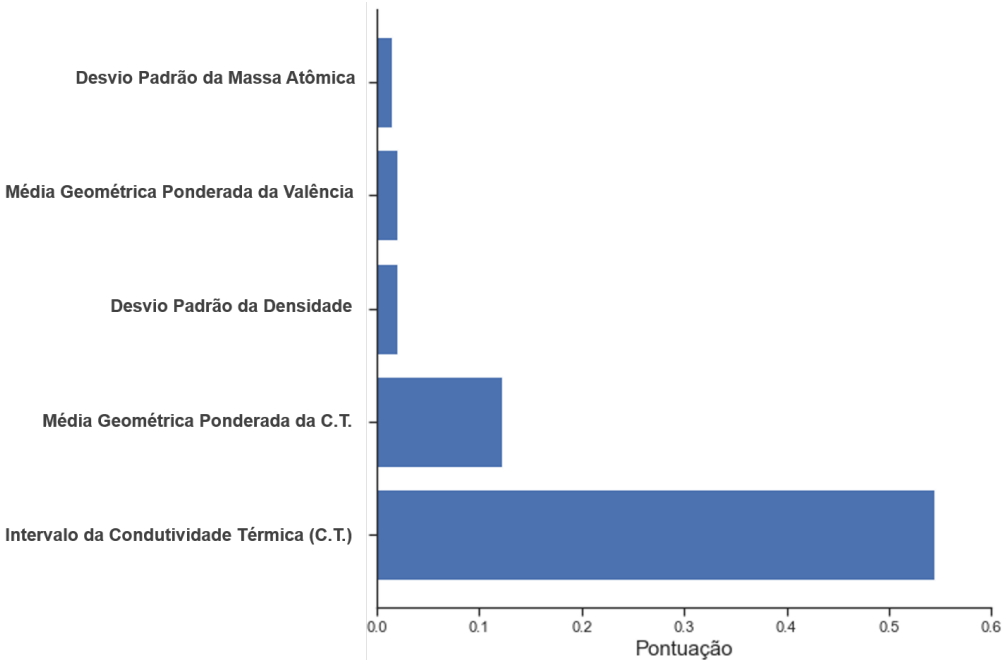
As melhores características, perante o ganho de informação no presente modelo, estão representadas na Figura 4.27. As duas características de maior ganho de informação estão relacionadas à condutividade térmica do supercondutor. Ao analisar a Figura 4.27, é possível perceber que as características mais importantes deste modelo estão presentes em outros, que também se basearam nas árvores de decisão.

Assim como para os outros modelos de árvores de decisão, a evolução das métricas R^2 e $RMSE$ em decorrência da utilização das características mais importantes está mostrada na Figura 4.28. Pode-se perceber que o gráfico elucidado na Figura 4.28 para o modelo Gradient Boosting, se assemelha muito ao gráfico para o modelo Árvores Extremamente Aleatórias (Figura 4.23). Este fato permite dizer que ambos modelos se apoiam de modo semelhante em suas principais características, para predizer a T_c dos supercondutores.

A Tabela 4.5 compara os valores das predições do modelo com os valores encontrados na literatura, para as temperaturas críticas dos supercondutores estudados na EEL-USP. Seguindo a linha de discussão das seções anteriores deste capítulo, pode-se afirmar que o presente modelo

pode predizer somente a temperatura crítica do supercondutor $Zr_5Pt_3C_{0.3}$, com um desvio menor que 10%.

Figura 4.27 – Características que mais contribuem com o ganho de informação no modelo Gradient Boosting



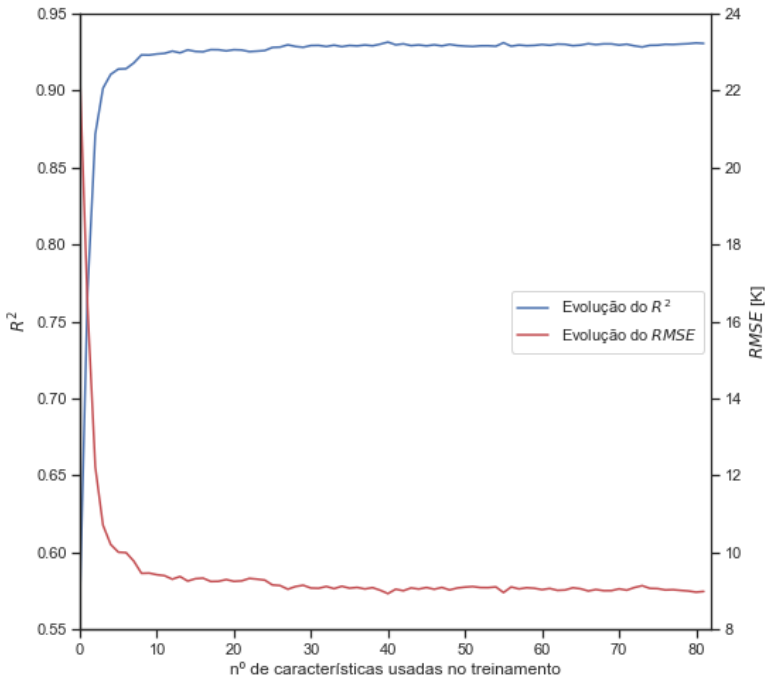
Fonte: o autor

Tabela 4.5 – Comparação das temperaturas críticas estimadas pelo modelo Gradient Boosting com as encontradas na literatura

Supercondutor	T_c Estimada	T_c Literatura
Ti_2GeC (BORTOLOZO et al., 2012a)	13,2 K	9,5 K
HfV_2Ga_4 (FERREIRA et al., 2018)	6,6 K	4,1 K
$Ti-NiTe_2$ (LIMA et al., 2018)	4,5 K	4,0 K
Ti_2InC (BORTOLOZO et al., 2007)	5,4 K	3,1 K
Nb_2SnC (BORTOLOZO et al., 2006)	6,6 K	7,8 K
$Zr_{0.96}V_{0.04}B_2$ (RENOSTO et al., 2013)	6,7 K	8,7 K
Nb_5Ge_3 (BORTOLOZO et al., 2012b)	3,8 K	15,3 K
$Zr_5Pt_3C_{0.3}$ (RENOSTO et al., 2018)	6,8 K	7,0 K

Fonte: o autor

Figura 4.28 – Evolução de R^2 e $RMSE$ diante do número de características utilizadas para treinar o modelo Gradient Boosting



Fonte: o autor

4.9 Rede Neural Profunda

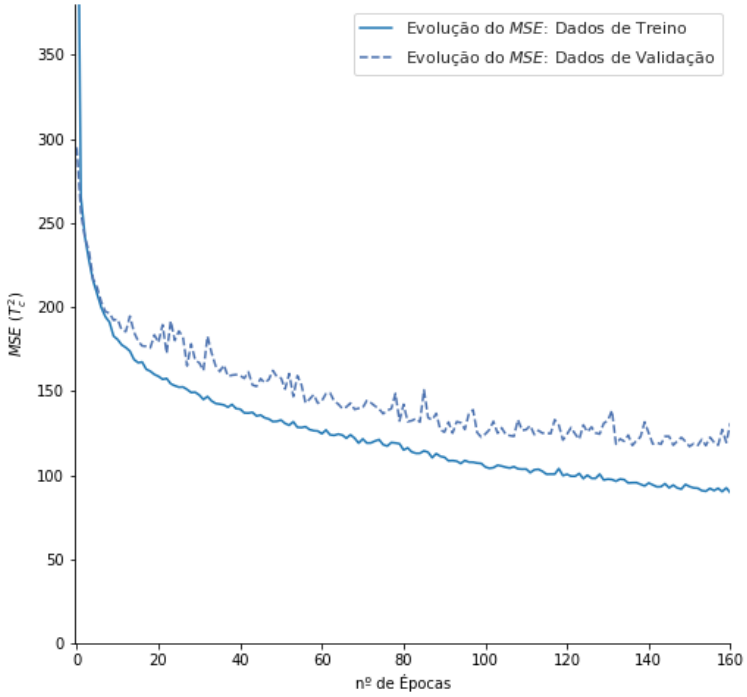
Para a Rede Neural Profunda, encontraram-se melhores valores para R^2 e $RMSE$ quando o sistema consistia em 3 camadas ocultas. Nas camadas ocultas, a primeira possuía 61 neurônios, a segunda 41 neurônios e a terceira 21 neurônios. Além disso, determinou-se que 32 amostras passariam pela rede, em cada reajuste dos pesos feito pelo otimizador.

O número de vezes em que todos os dados passam pela rede é denominado época (GOOGLE BRAIN TEAM, 2020). O valor da época foi determinado através de uma função de parada antecipada. Assim, este hiperparâmetro foi definido a partir do ponto em que o modelo para de ser melhorado significativamente, perante a redução do MSE. A Figura 4.29 apresenta a evolução do MSE a partir do número de épocas.

A partir da análise gráfica e de um atributo da biblioteca TensorFlow, o número de épocas foi definido em 153. O gráfico da Figura 4.29 apresenta a minimização do MSE com os dados do conjunto de treino. 20% deste conjunto foi usado para validar a minimização do MSE durante o treinamento da rede. A curva contínua na Figura 4.29 representa a redução do MSE, perante os dados que estão continuamente melhorando a predição do modelo. A curva tracejada na Figura

4.29, apresenta o desempenho da redução do MSE pelos 20% dos dados de treino - dados que não foram usados no treinamento. A partir da curva tracejada é que se defini o número de épocas.

Figura 4.29 – Evolução do MSE diante do número de épocas



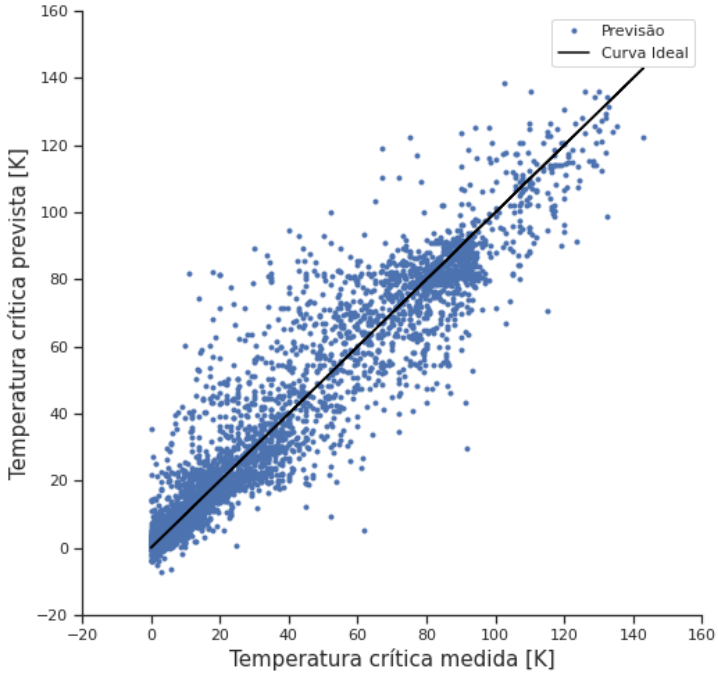
Fonte: o autor

A partir do treinamento da rede com os hiperparâmetros acima, pode-se obter um R^2 de 0,92 e um $RMSE$ de 9,83 K, para os dados de treino. Nos dados de teste, o R^2 se consolidou em 0,90 e o $RMSE$ em 11,19 K. A Figura 4.30 apresenta o gráfico de T_c prevista *versus* T_c observada para a Rede Neural Profunda.

Por mais que o modelo tenha atingido bons resultados para R^2 e $RMSE$, em comparação aos apresentados neste capítulo, ele não foi o melhor. Além disso, o modelo de Rede Neural Profunda trouxe novamente o problema de predição em baixas temperaturas. Na Figura 4.30, por mais que o modelo distribua bem as temperaturas ao redor da Curva Ideal, é possível ver que ele prediz algumas temperaturas com valores abaixo de 0 K, configurando uma incoerência física.

Considerando os bons valores de R^2 e $RMSE$, foi elaborada uma análise das predições de T_c em diferentes faixas de temperaturas, nos mesmos moldes dos modelos anteriores apresentados neste capítulo. A Figura 4.31 apresenta a análise feita para a Rede Neural Profunda, com base nas predições de T_c com desvios menores que 10%.

Figura 4.30 – T_c prevista *versus* T_c observada para a Rede Neural Profunda, com R^2 de 0,90 e $RMSE$ de 11,19 K

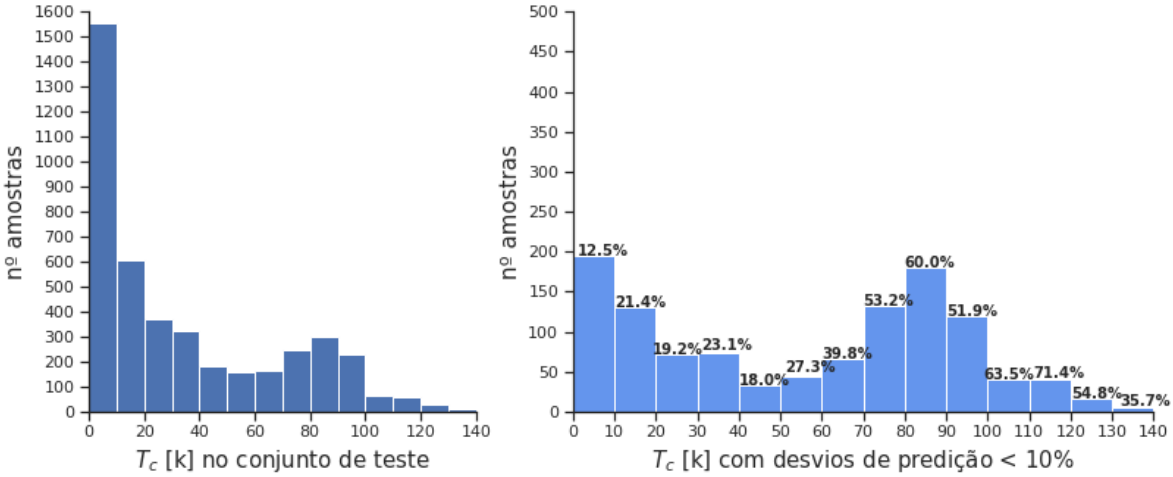


Fonte: o autor

A Figura 4.31 revela que na Rede Neural Profunda, os percentuais de predição de T_c com desvio menor que 10% foram bem inferiores aos apresentados por outros modelos, que se basearam em árvores de decisão. Apesar disso, a Rede Neural Profunda conseguiu prever 78,6% das temperaturas críticas na faixa de 130 a 140 K, com desvio menor que 10% do valor medido em laboratório.

A Tabela 4.6 apresenta os valores da predição de T_c feita pelo modelo Rede Neural Profunda, para os supercondutores apresentados na Seção 4.5. O modelo conseguiu predizer bem as temperaturas críticas dos supercondutores $NiTe_2$ e $Zr_{0.96}V_{0.04}B_2$, desviando a predição de ambos em 0,1 K.

Figura 4.31 – O gráfico à esquerda mostra a quantidade de amostras no conjunto de teste. O gráfico à direita apresenta o percentual e o número de amostras do conjunto de teste, que apresentaram desvio menor que 10% no processo de predição de temperatura crítica pela Rede Neural Profunda



Fonte: o autor

Tabela 4.6 – Comparação das temperaturas críticas estimadas pelo modelo Rede Neural Profunda com as encontradas na literatura

Supercondutor	T_c Estimada	T_c Literatura
Ti_2GeC (BORTOLOZO et al., 2012a)	15,2 K	9,5 K
HfV_2Ga_4 (FERREIRA et al., 2018)	9,3 K	4,1 K
$Ti-NiTe_2$ (LIMA et al., 2018)	9,3 K	4,0 K
Ti_2InC (BORTOLOZO et al., 2007)	2,3 K	3,1 K
Nb_2SnC (BORTOLOZO et al., 2006)	5,0 K	7,8 K
$Zr_{0.96}V_{0.04}B_2$ (RENOSTO et al., 2013)	8,8 K	8,7 K
Nb_5Ge_3 (BORTOLOZO et al., 2012b)	7,2 K	15,3 K
$Zr_5Pt_3C_{0.3}$ (RENOSTO et al., 2018)	4,8 K	7,0 K

Fonte: o autor

4.10 Discussão geral dos resultados

Com base nos resultados apresentados, o modelo Árvores Extremamente Aleatórias mostrou-se como o melhor. Além dos melhores valores para as medidas de desempenho R^2 e $RMSE$, o modelo Árvores Extremamente Aleatórias obteve os melhores resultados de predições em diferentes faixas de temperaturas, como pode ser observado na Figura 4.24.

Com base no parágrafo anterior, pode-se afirmar que este trabalho conseguiu melhor descrever o problema proposto, com um R^2 igual a 0,94 e $RMSE$ igual a 8,72 K, para os dados

de teste. Perante o trabalho de Hamidieh (2018), o mais parecido com esta monografia, o R^2 e o $RMSE$ obtidos para o mesmo problema foram de 0,92 e 9,5 K, respectivamente.

Hamidieh (2018) resolveu a mesma problemática apresentada neste trabalho, utilizando a mesma base de dados, porém com um número menor de amostras de supercondutores. Hamidieh (2018) utilizou o modelo Gradient Boosting, com os seguintes hiperparâmetros: 374 árvores de decisão; profundidade das árvores igual a 16; mínimo de 1 amostra em cada folha; taxa de aprendizado de 0,02.

Em comparação com os resultados do Gradient Boosting desta monografia ($R^2 = 0,93$ e $RMSE = 8,97$ K), é possível observar que ambos trabalhos atingiram resultados similares. Entretanto, os resultados apresentados na Seção 4.8 mostra um modelo Gradient Boosting mais regularizado para esta monografia.

O modelo Gradient Boosting deste trabalho utilizou menos árvores de decisão, com profundidade de árvores menores e com mais restrições para o crescimento das árvores. A explicação para este fato, é a utilização da metodologia empregada na Capítulo 3, que deriva da estruturação teórica proveniente de Scikit-learn (2020e) e Géron (2019).

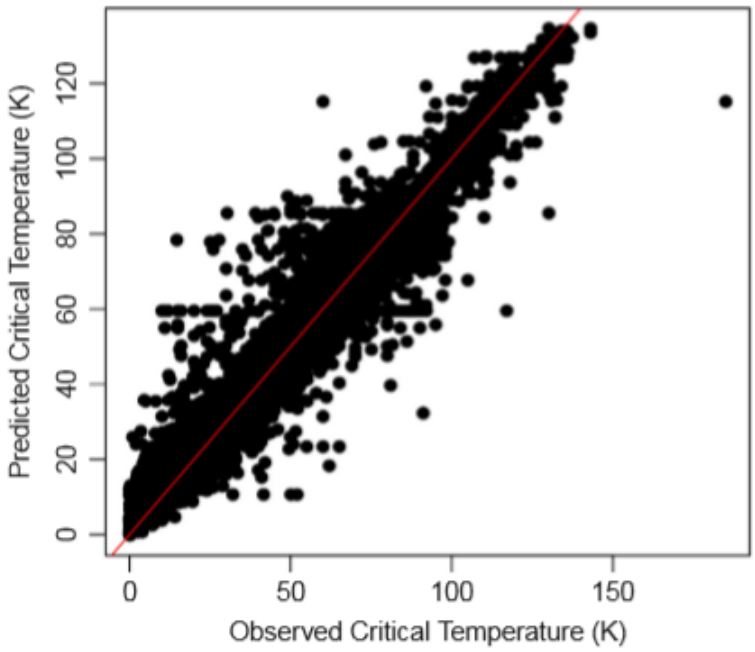
A Figura 4.32 mostra o resultado do gráfico de T_c prevista *versus* T_c observada para o trabalho de Hamidieh (2018). Para discutir a eficiência dos resultados na Figura 4.32, Hamidieh (2018) usou o modelo de Regressão Linear Múltiplica. Os resultados da Regressão Linear Múltiplica, apresentados em Hamidieh (2018), são idênticos aos elucidados neste trabalho na Seção 4.2.

Além de Hamidieh (2018), outros autores produziram trabalhos visando contornar o problema de predição da temperatura crítica de supercondutores, usando modelos de ML. Le et al. (2020), por exemplo, focou em explorar a capacidade da predição de T_c em altas temperaturas. Em seu trabalho, ele conseguiu atingir um R^2 de 0,94 e um $RMSE$ de 3,83 K, através de uma Rede Neural baseada no teorema de Bayes.

Além do resultado de Le et al. (2020), Stanev et al. (2018) revela um R^2 de 0,85 para a predição da T_c de supercondutores de altas temperaturas, usando um modelo de Floresta Aleatória. Também, Owolabi, Akande e Olatunji (2016) treinam um modelo de ML usando Máquinas de Vetores de Suporte, para predizer T_c de supercondutores YBCO. Para este último, um resultado de R^2 igual a 0,96 é alcançado.

Durante o desenvolvimento desta monografia, outros trabalhos foram publicados com a mesma temática. Roter e Dordevic (2020) publicaram na revista *Physica C: Superconductivity and its Applications*, um artigo sobre a predição de T_c usando um modelo com várias árvores de decisão. No trabalho de Roter e Dordevic (2020), a mesma base de dados explorada nesta monografia foi empregada. Assim, Roter e Dordevic (2020) conseguiram atingir um R^2 de 0,93 e um $RMSE$ de 8,91 K. A Figura 4.33 mostra o gráfico de T_c prevista *versus* T_c observada para o trabalho de Roter e Dordevic (2020).

Figura 4.32 – T_c prevista *versus* T_c observada para o Gradient Boosting, com R^2 de 0,92 e $RMSE$ de 9,5 K, segundo trabalho de Hamidieh (2018)



Fonte: Hamidieh (2018)

Perante os trabalhos de outros autores, pode-se afirmar que os modelos desenvolvidos nesta monografia atingiram bons resultados. Para o modelo de Floresta Aleatória, por exemplo, este trabalho alcançou um R^2 de 0,93. Para um mesmo modelo de Floresta Aleatória, limitado a altos valores de T_c, Stanev et al. (2018) atingiu um R^2 de 0,84.

Por mais que neste trabalho os modelos de SVM tenham sido pouco explorados, por não estimarem bem as baixas temperaturas, Owolabi, Akande e Olatunji (2016) apresentaram um R^2 de 0,96 para supercondutores YBCO, usando um modelo de SVM.

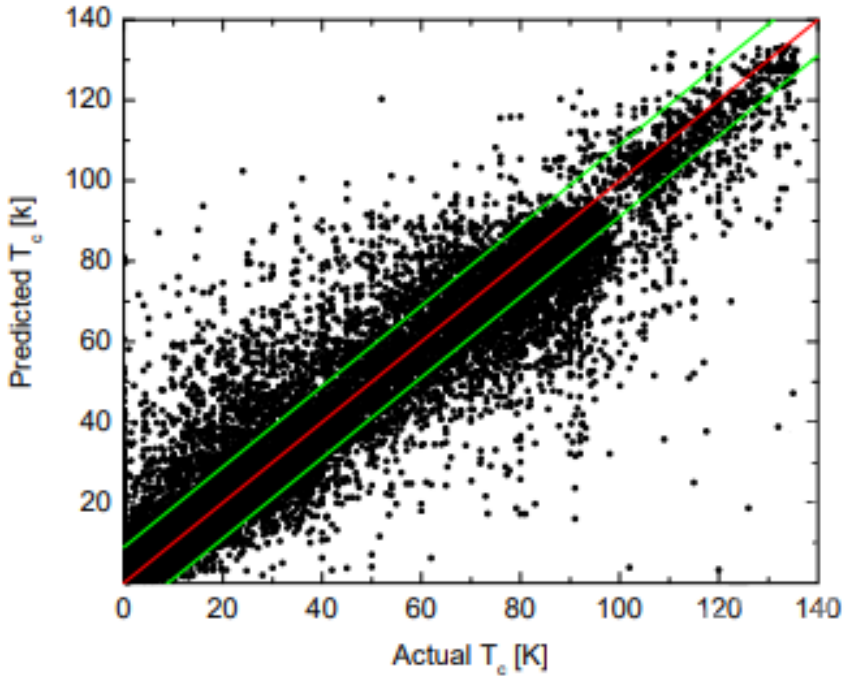
Destacou-se também o $RMSE$ de 3,83 K apresentado no trabalho de Le et al. (2020). Apesar do erro ser extremamente significativo perantes os apresentados neste trabalho, Le et al. (2020) também direciona suas análises à temperaturas críticas altas. Deve-se salientar, que nesta monografia a T_c foi explorada sem distinção de faixas de temperaturas.

Contudo, os resultados de Roter e Dordevic (2020) ficaram os mais próximos aos discutidos nesta monografia, para o modelo de Árvores Extremamente Aleatórias. Aliás, pode-se encontrar semelhanças entre as Figuras 4.21 e 4.33, que representam os resultados das predições desses modelos.

Na literatura, não foi possível encontrar resultados que apresentassem os mesmos hiperparâmetros atingidos nesta monografia. Além disso, não foi encontrado nenhum trabalho que explorasse a aplicação do modelo Árvores Extremamente Aleatórias para predizer a T_c de

supercondutores.

Figura 4.33 – T_c prevista *versus* T_c observada, com R^2 de 0,93 e $RMSE$ de 8,91 K segundo trabalho de Roter e Dordevic (2020)



Fonte: Roter e Dordevic (2020)

5 Considerações Finais

5.1 Conclusão

Pode-se concluir que o presente trabalho conseguiu melhor predizer a temperatura crítica dos supercondutores a um R^2 de 0,94 e a um erro de $\pm 8,72$ K. Estes resultados são provenientes do modelo Árvores Extremamente Aleatórias, que se mostrou o melhor modelo dentre os avaliados. Além de obter os melhores valores para R^2 e $RMSE$, este modelo se mostrou mais preciso nas predições de supercondutores em diferentes faixas de temperatura como pode ser visto na Figura 4.24. Como um diferencial para este modelo, pode-se destacar que ele conseguiu utilizar melhor as características calculadas na Seção 3.2 para predizer T_c, como pode ser mostrado na Figura 4.23.

Outra conclusão importante para este trabalho, é fato dos modelos que usaram árvores de decisão para realizar predições terem conseguido melhor generalizar o problema proposto. Como pode ser visto nos resultados mostrados no Capítulo 4, os modelos que usaram árvores de decisão, além de obterem os melhores resultados para R^2 e $RMSE$, conseguiram de maneira generalizada aproximar suas predições aos valores observados em laboratório. Este resultado é reafirmado pelas análises de predições em diferentes faixas de temperatura e pelos gráficos de T_c prevista *versus* T_c observada.

Comparando esta monografia com outros trabalhos, como foi feito na Seção 4.10, é possível concluir que os resultados dos melhores modelos apresentados no Capítulo 4 são próximos aos apresentados por Roter e Dordevic (2020) e Hamidieh (2018). Comparado com o trabalho de Hamidieh (2018), o qual possui a mesma estrutura de base de dados, este trabalho aumentou o R^2 em 0,02 e diminuiu o $RMSE$ em 0,8 K ao utilizar o modelo de Árvores Extremamente Aleatórias, ao invés do modelo Gradient Boosting. Além disso, ao comparar o modelo Gradient Boosting desenvolvido nesta monografia, com o mesmo modelo usado por Hamidieh (2018), é possível perceber que neste trabalho o modelo em questão encontra-se mais regularizado. O resultado de uma melhor regularização pode ser atribuído ao processo de validação cruzada nos moldes da Seção 3.4.

Na tentativa de predizer a T_c dos supercondutores abordados em artigos do Departamento de Engenharia de Materias da EEL-USP, pode-se concluir que nenhum modelo conseguiu prever bem os resultados dos supercondutores em questão. Um motivo possível para estas predições não serem próximas ao esperado para estes supercondutores, é a não representatividade das características levantas na Seção 3.2. Apesar disso, um R^2 de 0,94 conduz a relevância da busca por características mais representativas para a problemática apresentada, ou até mesmo novas abordagens de aprendizado de máquina.

Por fim, por mais que a IA tenha sido teorizada a muito tempo, como pode ser visto na Seção 2.2.6, ela vem sendo utilizada cada vez mais por conta da capacidade de processamento dos computadores e a disponibilidade de dados. Por esse motivo, é possível observar o aumento da utilização de modelos de ML associado à descrição de problemas físicos, como o apresentado aqui e nas publicações referenciadas no corpo desta monografia. Nessa perspectiva, por mais que a utilização de modelos de ML não substitua a necessidade de descrição teoria de fenômenos como o da supercondutividade, eles podem ser usados como ferramenta em apoio a resultados científicos.

5.2 Trabalhos Futuros

Este trabalho usou somente características estatísticas levantadas pela fórmula química de supercondutores. Desse modo, a exploração de características mais relacionadas à problemática da supercondutividade poderiam ser incorporadas no treinamento dos modelos de ML, como pressão, estrutura cristalina e melhor descrição teórica ou numérica para o supercondutor.

Ademais, para trabalhos futuros, diferentes modelos de *machine learning* poderiam ser explorados. Além disso, caberia a avaliação do treinamento de modelos restringindo-os a uma faixa de temperatura crítica, como feito nos trabalhos de Stanev et al. (2018) e Le et al. (2020), referenciados nesta monografia.

Outra alternativa à sequenciação deste trabalho seria a exploração de outras propriedades supercondutoras, como campo crítico ou densidade decorrente crítica. Ainda para a supercondutividade, além de implementar problemas de regressão, seria possível avaliar o desempenho dos modelos de ML em classificar se determinado material apresentaria ou não indícios de supercondutividade.

Também, uma outra possibilidade de continuação seria a utilização da metodologia desenvolvida por este trabalho, para avaliar o desempenho de modelos de ML na predição de outras propriedades físicas, não necessariamente ligadas à supercondutividade.

Referências

ALPAYDIN, E. *Introduction to machine learning*. [S.l.]: MIT press, 2020.

BIGOTO, M. A. R. *Implementação de modelos de machine learning para predição de temperaturas críticas de supercondutores*. 2020. Disponível em: <https://github.com/muriloafonso/TG_ENG_FISICA>. Acesso em: 14 jul. 2020.

BISHOP, C. M. *Pattern recognition and machine learning*. [S.l.]: springer, 2006.

BOERI, L. Understanding novel superconductors with ab initio calculations. *Handbook of Materials Modeling: Applications: Current and Emerging Materials*, Springer, p. 73–112, 2020.

BORTOLOZO, A. et al. Superconductivity in the nb_2snc compound. *Solid state communications*, Elsevier, v. 139, n. 2, p. 57–59, 2006.

BORTOLOZO, A. et al. Superconductivity in the hexagonal-layered nanolaminates ti_2inc compound. *Solid State Communications*, Elsevier, v. 144, n. 10-11, p. 419–421, 2007.

BORTOLOZO, A. et al. Superconductivity at 9.5 k in the ti_2gec compound. *Materials Science-Poland*, Springer, v. 30, n. 2, p. 92–97, 2012.

BORTOLOZO, A. D. et al. Interstitial doping induced superconductivity at 15.3 k in nb_5ge_3 compound. *Journal of Applied Physics*, American Institute of Physics, v. 111, n. 12, p. 123912, 2012.

COSTA, M.; PAVÃO, A. C. Supercondutividade: um século de desafios e superação. *Revista Brasileira de Ensino de Física*, SciELO Brasil, v. 34, n. 2, p. 2602–2615, 2012.

FACELI, K. et al. Inteligência artificial: Uma abordagem de aprendizado de máquina. 2011.

FERREIRA, P. et al. Insights into the unconventional superconductivity in hfv_2ga_4 and scv_2ga_4 from first-principles electronic-structure calculations. *Physical Review B*, APS, v. 98, n. 4, p. 045126, 2018.

GÉRON, A. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. [S.l.]: O'Reilly Media, 2019.

GOOGLE BRAIN TEAM. *An end-to-end open source machine learning platform*. 2020. Disponível em: <https://www.tensorflow.org/>. Acesso em: 13 mai. 2020.

GOOGLE COLAB. *Welcome to colaboratory*. 2020. Disponível em: <https://colab.research.google.com/notebooks/intro.ipynb>. Acesso em: 16 mai. 2020.

HAMIDIEH, K. A data-driven statistical model for predicting the critical temperature of a superconductor. *Computational Materials Science*, Elsevier, v. 154, p. 346–354, 2018.

IDRIS, I. *Python data analysis*. [S.l.]: Packt Publishing Ltd, 2014.

JUPYTER. *JupyterLab Overview*. 2020. Disponível em: <https://jupyterlab.readthedocs.io/en/stable/getting_started/overview.html#jupyterlab-releases>. Acesso em: 16 mai. 2020.

KERAS SPECIAL INTEREST GROUP. *Keras TensorFlow 2.0*. 2020. Disponível em: <https://keras.io/>. Acesso em: 02 jun. 2020.

KHANNA, V. K. Superconductive electronics for ultra-cool environments. In: *Extreme-Temperature and Harsh-Environment Electronics*. [S.l.]: IOP Publishing, 2017, (2053-2563). p. 12–1 to 12–47.

LE, T. D. et al. Critical temperature prediction for a superconductor: A variational bayesian neural network approach. *IEEE Transactions on Applied Superconductivity*, IEEE, v. 30, n. 4, p. 1–5, 2020.

LIMA, B. et al. Properties and superconductivity in ti-doped nite$_2$ single crystals. *Solid State Communications*, Elsevier, v. 283, p. 27–31, 2018.

MATHEMATICA. 11.2. wolfram research. *Inc. Champaign*, 2020.

MICROSOFT. *O que é um modelo de machine learning?* 2020. Disponível em: <https://docs.microsoft.com/pt-br/windows/ai/windows-ml/what-is-a-machine-learning-model>. Acesso em: 02 jun. 2020.

MITCHELL, R.; MICHALSKI, J.; CARBONELL, T. *An artificial intelligence approach*. [S.l.]: Springer, 2013.

NATIONAL INSTITUTE FOR MATERIALS SCIENCE. *Superconducting Material Database (SuperCon)*. 2020. Disponível em: <https://supercon.nims.go.jp/index_en.html>. Acesso em: 03 mar. 2020.

OWOLABI, T. O.; AKANDE, K. O.; OLATUNJI, S. O. Application of computational intelligence technique for estimating superconducting transition temperature of ybco superconductors. *Applied Soft Computing*, Elsevier, v. 43, p. 143–149, 2016.

PAULY, L. et al. Deeper networks for pavement crack detection. In: IAARC. *Proceedings of the 34th ISARC*. [S.l.], 2017. p. 479–485.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. v. 12, p. 2825–2830, 2011.

RENOSTO, S. et al. Evidence of multiband behavior in the superconducting alloy zr$_{0.96}$v$_{0.04}$b$_2$. *Physical Review B*, APS, v. 87, n. 17, p. 174502, 2013.

RENOSTO, S. T. et al. *Strong Electronic Interaction and Signatures of Nodal Superconductivity in $Zr_5Pt_3C_x$*. 2018.

ROTER, B.; DORDEVIC, S. Predicting new superconductors and their critical temperatures using machine learning. *Physica C: Superconductivity and its Applications*, Elsevier, p. 1353689, 2020.

SCIKIT-LEARN. *Cross-validation: evaluating estimator performance*. 2020. Disponível em: <https://scikit-learn.org/stable/modules/cross_validation.html>. Acesso em: 13 mai. 2020.

SCIKIT-LEARN. *Ensemble methods*. 2020. Disponível em: <https://scikit-learn.org/stable/modules/ensemble.html>. Acesso em: 07 jun. 2020.

SCIKIT-LEARN. *Linear Models*. 2020. Disponível em: <https://scikit-learn.org/stable/modules/linear_model.html>. Acesso em: 05 jun. 2020.

SCIKIT-LEARN. *Metrics and scoring: quantifying the quality of predictions*. 2020. Disponível em: <https://scikit-learn.org/stable/modules/model_evaluation.html>. Acesso em: 04 jun. 2020.

SCIKIT-LEARN. *Scikit-learn: Machine Learning in Python*. 2020. Disponível em: <https://scikit-learn.org/stable/index.html>. Acesso em: 15 mai. 2020.

SCIKIT-LEARN. *Support Vector Machines*. 2020. Disponível em: <https://scikit-learn.org/stable/modules/svm.html#svm-regression>. Acesso em: 01 jun. 2020.

SILVA, I.; SPATTI, D.; FLAUZINO, R. Redes neurais artificiais: para engenharia e ciências aplicadas. [sl]: Artliber. *Artliber; Edição: 2ª (1 de janeiro de 2016)*, v. 5, 2010.

STANEV, V. et al. Machine learning modeling of superconducting critical temperature. *npj Computational Materials*, Nature Publishing Group, v. 4, n. 1, p. 1–14, 2018.