UNIVERSIDADE FEDERAL DE GOIÁS (UFG)
UNIDADE ACADÊMICA ESPECIAL DE MATEMÁTICA E TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM MODELAGEM E OTIMIZAÇÃO
(PPGMO)

Bruno Pereira Barella

MODELAGEM COM APRENDIZADO DE MÁQUINA APLICADA AOS SISTEMAS DE MONITORAMENTO DE INTEGRIDADE ESTRUTURAL

DISSERTAÇÃO DE MESTRADO

CATALÃO – GO, 2021

UNIVERSIDADE FEDERAL DE GOIÁS
UNIDADE ACADÊMICA ESPECIAL DE MATEMÁTICA E TECNOLOGIA

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO (TECA) PARA DISPONIBILIZAR VERSÕES ELETRÔNICAS DE TESES

E DISSERTAÇÕES NA BIBLIOTECA DIGITAL DA UFG

Na qualidade de titular dos direitos de autor, autorizo a Universidade Federal de Goiás (UFG) a disponibilizar, gratuitamente, por meio da Biblioteca Digital de Teses e Dissertações (BDTD/UFG), regulamentada pela Resolução CEPEC nº 832/2007, sem ressarcimento dos direitos autorais, de acordo com a Lei 9.610/98, o documento conforme permissões assinaladas abaixo, para fins de leitura, impressão e/ou download, a título de divulgação da produção científica brasileira, a partir desta data.

O conteúdo das Teses e Dissertações disponibilizado na BDTD/UFG é de responsabilidade exclusiva do autor. Ao encaminhar o produto final, o autor(a) e o(a) orientador(a) firmam o compromisso de que o trabalho não contém nenhuma violação de quaisquer direitos autorais ou outro direito de terceiros.

1. Identificação do material bibliográfico

[x] Dissertação [ ] Tese

2. Nome completo do autor

Bruno Pereira Barella

3. Título do trabalho

"MODELAGEM COM APRENDIZADO DE MÁQUINA APLICADA AOS SISTEMAS DE MONITORAMENTO DE INTEGRIDADE ESTRUTURAL"

4. Informações de acesso ao documento (este campo deve ser preenchido pelo orientador)

Concorda com a liberação total do documento [X] SIM [ ] NÃO[1]

[1] Neste caso o documento será embargado por até um ano a partir da data de defesa. Após esse período, a possível disponibilização ocorrerá apenas mediante:
a) consulta ao(à) autor(a) e ao(à) orientador(a);
b) novo Termo de Ciência e de Autorização (TECA) assinado e inserido no arquivo da tese ou dissertação.
O documento não será disponibilizado durante o período de embargo.
Casos de embargo:
- Solicitação de registro de patente;
- Submissão de artigo em revista científica;
- Publicação como capítulo de livro;
- Publicação da dissertação/tese em livro.

Obs. Este termo deverá ser assinado no SEI pelo orientador e pelo autor.

Documento assinado eletronicamente por **Jose Dos Reis Vieira De Moura Junior**, **Orientador**, em 29/04/2021, às 06:11, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **BRUNO PEREIRA BARELLA**, **Discente**, em 29/04/2021, às 09:50, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de

8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://sei.ufg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **2028795** e o código CRC **8B2C33AC**.

Referência: Processo nº 23070.015110/2021-81 SEI nº 2028795

BRUNO PEREIRA BARELLA

MODELAGEM COM APRENDIZADO DE MÁQUINA APLICADA AOS SISTEMAS DE MONITORAMENTO DE INTEGRIDADE ESTRUTURAL

Dissertação apresentada ao Programa de Pós-Graduação em Modelagem e Otimização, da Unidade Acadêmica Especial de Matemática e Tecnologia da Universidade Federal de Goiás (UFG), como requisito para a obtenção do título de Mestre em Modelagem e Otimização.
Área de concentração: Modelagem e Otimzação

Orientador:
José dos Reis Vieira de Moura Júnior

Coorientador:
Márcio José da Cunha

CATALÃO – GO

2021

Ficha de identificação da obra elaborada pelo autor, através do
Programa de Geração Automática do Sistema de Bibliotecas da UFG.

Barella, Bruno Pereira
Modelagem com Aprendizado de Máquina Aplicada aos Sistemas de Monitoramento de Integridade Estrutural [manuscrito] / Bruno Pereira Barella. - 2021.
CLVI, 156 f.

Orientador: Prof. Dr. José dos Reis Vieira de Moura Júnior; co orientador Dr. Márcio José da Cunha.
Dissertação (Mestrado) - Universidade Federal de Goiás, Unidade Acadêmica Especial de Matemática e Tecnologia, Catalão, Programa de Pós-Graduação em Modelagem e Otimização, Catalão, 2021.
Bibliografia.
Inclui siglas, abreviaturas, tabelas, lista de figuras, lista de tabelas.

1. Impedância eletromecânica. 2. Monitoramento da integridade estrutural. 3. Sistemas de monitoramento de integridade estrutural. 4. Aprendizado de máquina. 5. Inteligência artificial. I. Moura Júnior, José dos Reis Vieira de, orient. II. Título.

CDU 519.25

UNIVERSIDADE FEDERAL DE GOIÁS

UNIDADE ACADÊMICA ESPECIAL DE MATEMÁTICA E TECNOLOGIA

ATA DE DEFESA DE DISSERTAÇÃO

Ata nº 08 da sessão de Defesa de Dissertação de **Bruno Pereira Barella**, que confere o título de **Mestre(a) em Modelagem e Otimização**, na área de concentração em **Modelagem e Otimização**.

Aos vinte e cinco de março de 2021, às 16h30min, por Webconfência, via sistema Sala google meeting, reuniram-se os componentes da banca examinadora, professores(as) **Dr. José dos Reis Vieira de Moura Júnior (orientador)** (PPGMO - "RC/UFG - UFCAT em transição”)**, Dr. Márcio José da Cunha (coorientador) (**FEELT/UFU), **Dr. Vaston Gonçalves da Costa** (PPGMO - "RC/UFG - UFCAT em transição”) e **Dr. Carlos Alberto Gallo (**FEMEC/UFU) para, em sessão pública, procederem a avaliação da Dissertação intitulado(a) “*MODELAGEM COM APRENDIZADO DE MÁQUINA APLICADA AOS SISTEMAS DE MONITORAMENTO DE INTEGRIDADE ESTRUTURAL*”, de autoria de **Bruno Pereira Barella**, discente do Programa de Pós-graduação em Modelagem e Otimização – PPGMO, da "RC/UFG-UFCAT em transição". A sessão foi aberta pelo(a) presidente, que fez a apresentação formal dos membros da banca. Em seguida, a palavra foi concedida ao discente que, em 54 min procedeu a apresentação. Terminada a apresentação, cada membro da banca arguiu o examinando. Terminada a fase de arguição, procedeu-se a avaliação da Dissertação, que foi considerado(a): **(X) Aprovado ou ( ) Reprovado(a)**. Cumpridas as formalidades de pauta, a presidência da mesa encerrou a sessão e, para constar, lavrou-se a presente ata que, depois de lida e aprovada, segue assinada pelos membros da banca examinadora e pelo discente.

Obs: "*Banca Examinadora de Qualificação/Defesa Pública de Dissertação/Tese realizada em conformidade com a Portaria da CAPES n. 36, de 19 de março de 2020, de acordo com seu segundo artigo:*

Art. 2o A suspensão de que trata esta Portaria não afasta a possibilidade de defesas de tese utilizando tecnologias de comunicação à distância, quando admissíveis pelo programa de pós-graduação stricto sensu, nos termos da regulamentação do Ministério da Educação."

TÍTULO SUGERIDO PELA BANCA

Documento assinado eletronicamente por **Jose Dos Reis Vieira De Moura Junior**, **Orientador**, em 06/04/2021, às 14:00, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **BRUNO PEREIRA BARELLA**, **Discente**, em 06/04/2021, às 15:56, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Márcio José da Cunha**, **Usuário Externo**, em 06/04/2021, às 16:46, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Carlos Alberto Gallo**, **Usuário Externo**, em 07/04/2021, às 08:59, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Vaston Gonçalves Da Costa**, **Professor do Magistério Superior**, em 19/04/2021, às 15:21, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://sei.ufg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **1980617** e o código CRC **373C441E**.

Referência: Processo nº 23070.015110/2021-81 SEI nº 1980617

Agradecimentos

Inicialmente gostaria de agradecer imensamente meu orientador, José dos Reis, pela amizade e carinho. Sem dúvidas, seu apoio foi, e sempre será fundamental para o meu desenvolvimento. Saiba que és mais que um orientador para mim, é um querido paizão que admiro e me espelho. Deixo aqui meu agradecimento por todos ensinamentos, conversas e momentos felizes e tristes que passamos e ainda vamos passar.

Ao meu querido coorientador Márcio, deixo meu imenso agradecimento por todo apoio e direcionamento. Seus apontamentos foram muito importantes para o meu desenvolvimento e deste trabalho. Que possamos continuar a trabalhar juntos, produzindo bastante e fortalecendo nossa amizade.

Ao grande amigo Stanley Washington, deixo aqui meu agradecimento pelo companheirismo, lealdade e cumplicidade. Todo apoio foi fundamental para o meu desenvolvimento e deste trabalho. Que nossa amizade perdure o resto de nossas vidas.

À minha mãe Leandra, deixo um agradecimento especial, por todas as lições de amor, companheirismo, amizade, caridade, dedicação, abnegação, compreensão e perdão que você me dá a cada novo dia. Em minha vida a considero meu pilar e sua grande força foi a mola propulsora que permitiu o meu avanço, mesmo durante os momentos mais difíceis. Sinto-me orgulhoso e privilegiado por tê-la.

E ao meu pai Reinaldo deixo um carinhoso agradecimento, frente a todos os momentos de alegria, amor e amizade que passamos juntos. Vejo muitas características suas em mim e acredito que grande parte da minha capacidade é herança da sua pessoa. Tenho orgulho e me considero privilegiado por tê-lo.

E aos meus irmãos Camila, Giovana e Vinícius, sempre prontos a me apoiar em tudo nesta vida, deixo meu agradecimento. Apesar da distância acredito que todos vocês sempre se fizeram presentes de alguma forma. Sou grato pela vida de vocês e espero estarmos juntos para sempre.

Por fim agradeço aos meus avós Vó Nalí, Vô Zuis e Vó Fátima, por terem cuidado de mim quando precisei, por serem tão presentes em minha vida e por todo o amor e carinho.

RESUMO

BARELLA, BRUNO PEREIRA. *Modelagem com Aprendizado de Máquina Aplicada aos Sistemas de Monitoramento de Integridade Estrutural.* 2021. 153 f. Dissertação (Mestrado em Modelagem e Otimização) – Unidade Acadêmica Especial de Matemática e Tecnologia, Universidade Federal de Goiás, Catalão – GO.

O monitoramento da integridade estrutural (SHM – *Structural Health Monitoring*) utilizando o método de impedância eletromecânica tem como foco o desenvolvimento de sistemas responsáveis por monitorar a integridade de estruturas, tais como, fuselagem aeroespacial e estruturas metálicas. Para isso, o emprego de ferramentas e recursos computacionais são essenciais. A identificação, localização, quantificação de danos ou até a previsão de vida útil de sistemas mecânicos tem papel fundamental para garantia da segurança das pessoas e financeira. O emprego de ferramentas de aprendizado de máquina possibilita desempenhar tais funções devido a sua alta capacidade de identificar padrões não necessitando compensar efeitos naturais de carregamento e temperatura que comumente atuam sobre as estruturas mecânicas. Portanto, foram aplicadas 3 abordagens possíveis para o desenvolvimento de modelos a serem empregados nos sistemas de monitoramento de integridade estrutural, são elas: detecção de anomalias, multi classificação de danos e regressão sobre a variação de massa. Para o desenvolvimento das abordagens, foi realizado um experimento simulando danos estruturais através do processo de usinagem superficial numa viga de alumínio exposta a variações térmicas de 10 °C a 40 °C. Sistemas que utilizam aprendizado de máquina possuem modelos desenvolvidos com especificações únicas intrínsecas do problema e, para o desenvolvimento dessas, é necessário o emprego de ferramentas e métodos capazes de desempenhar suas funções com alta confiabilidade e precisão. Nesse sentido foram apresentados os principais conceitos das técnicas de monitoramento de integridade estrutural e as características dos materiais inteligentes que são utilizados como sensores. Posteriormente, o método de impedância eletromecânica foi abordado com os conceitos essenciais para seu entendimento, bem como os principais sistemas disponíveis para coleta das assinaturas de impedância e os modelos estatísticos comumente utilizados. Em seguida, foram apresentados os conceitos e recursos comuns da área de inteligência artificial, bem como ferramentas de análise, interpretação e avaliação de modelos. Por fim, um sistema foi implantado utilizando a abordagem de detecção de anomalia por meio de um aplicativo de monitoramento, através dos recursos de conteinerização e computação em nuvem.

Palavras-chaves: Impedância eletromecânica, Monitoramento da integridade estrutural, Sistemas de monitoramento de integridade estrutural, Computação em nuvem, Inteligência artificial, Aprendizado de máquina.

ABSTRACT

BARELLA, BRUNO PEREIRA. *Modeling with Machine Learning Applied to Structural Health Monitoring*. 2021. 153 f. Master Thesis in Modelling and Optimization – Unidade Acadêmica Especial de Matemática e Tecnologia, Universidade Federal de Goiás, Catalão – GO.

Structural Health Monitoring (SHM) using the electromechanical impedance method focuses on the development of systems responsible for monitoring the integrity of structures, such as, aerospace fuselage and metallic structures. Therefore, the use of computational tools and resources are essential. The identification, location, quantification of damage or even the prediction of the useful life of mechanical systems plays a fundamental role in ensuring the financial and public safety. That way, the use of machine learning tools is possible to perform such functions due to their high ability to identify patterns without needing to compensate for natural effects such as loading and temperature, which often act on mechanical structures. Thus, three possible approaches were applied for the development of models to be used in structural health monitoring systems, which are: anomalies detection, multi classification of damage and mass variation regression. To develop the approaches, an experiment was carried out simulating the structural damage produced by the surface machining process in a aluminum beam exposed to thermal variations between 10 °C and 40 °C. Systems that use machine learning have models developed with unique specifications intrinsic to the problem and, for their development, it is necessary to use tools and methods capable of performing their functions with high reliability and precision. In this sense, the main concepts of structural integrity monitoring techniques and the characteristics of intelligent materials used as sensors were presented. Subsequently, the electromechanical impedance method was approached with the essential concepts for its understanding, as well as the main systems available for collecting impedance signatures and the commonly used statistical models. Then, the common concepts and resources in the artificial intelligence field were presented, as well as tools for the analysis, interpretation and evaluation of models. Finally, a system was deployed containing the anomalie detection approach by means of a monitoring application, using the resources of containerization and cloud computing.

Keywords: Electromechanical impedance, Structural Health Monitoring, Structural integrity monitoring systems, cloud computing, Artificial Inteligence, Machine learning.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

AI — Artificial Inteligence

CC — Correlation Coefficient

CDA — Conversor Digital-Analógico

CI — Circuito Integrado

Cov — Covariance

DDS — Direct Digital Synthesizer

DL — Deep Learning

DSP — Digital Signal Processors

EFS — Effective Frequency Shift

END — Ensaios Não Destrutivos

FFT — Fast Fourier Transform

HP — Hewlett-Packard Company

IE — Impendância Eletromecânica

IS — Impedance Signature

ISHM — Impedance Based Structural Health Monitoring

KPCA — Kernel Principal Component Analysis

MAPD — Mean Absolute Percentage Deviation

MFC — Macro Fiber Composite

ML — Machine Learning

PVDF — Fluórido de Polivinilideno

PZT — Titanato-Zirconato de Chumbo

RMSD — Root Mean Square Deviation

RNA — Redes Neurais Artificiais

SHM — Structural Health Monitoring

SUMÁRIO

Capítulo 1

INTRODUÇÃO

A manutenção de sistemas denominados críticos é de vital importância, uma vez que alguns desempenham funções que podem impactar tanto na segurança de pessoas quanto em prejuízos financeiros. O monitoramento de integridade estrutural é utilizado para avaliar dinamicamente a integridade dos sistemas mecânicos ao longo de sua vida útil, permitindo a mínima intervenção humana e reduzindo custos (INMAN *et al.*, 2005). Fatores de extrema importância são a localização e a quantificação dos danos, bem como a forma que os mesmos afetam a estrutura.

Quando o processo de degradação de um componente/estrutura é monitorado, a manutenção pode ser planejada dinamicamente (manutenção baseada em condições) em vez de periodicamente (manutenção programada) com base em dados de monitoramento da integridade (WILLIAMS; DAVIES; DRAKE, 1994; ELEFTHEROGLOU *et al.*, 2018). Isso requer capacidade prognóstica para prever a evolução dos danos do estado de degradação do componente/estrutura no futuro. A palavra prognóstico é originalmente uma palavra grega que significa conhecer antecipadamente, prever (ELATTAR; ELMINIR; RIAD, 2016). Em engenharia, prognósticos são definidos como as estimativas da vida útil de um componente que está se degradando durante sua operação (AVEN, 2011).

Um método capaz de avaliar dinamicamente a degradação de um componente/estrutura é o Monitoramento da Integridade Estrutural (SHM - do inglês *Structural Health Monitoring*) que, segundo Farrar, Lieven e Bement (2005), é um processo de detecção de danos em estruturas que envolvem a observação do sistema ao logo do tempo, utilizando diversas métricas para avaliar as respostas dadas a partir de um conjunto de sensores, onde essas respostas abstraem as principais características sensíveis a algum dano.

Entre os diferentes tipos de SHM, existem dezenas de metodologias que lidam com a avaliação da integridade de estruturas, desde processos baseados em conceitos de vibrações clássicas no domínio do tempo e frequência até processos mais sofisticados (THOMSON, 2018; CAWLEY, 1984). Os mais recentes métodos baseados em bandas de alta frequência em

sua grande maioria usam sistemas de estruturas inteligentes cujos sensores/atuadores são incorporados à estrutura.

Dentre os diversos métodos que realizam o SHM podem ser destacados: impedância eletromecânica, ondas de Lamb, testes radiográficos, ensaios por ultrassom, líquidos penetrantes, medições de propriedades dinâmicas e partículas magnéticas (BRAY; MCBRIDE, 1992; PALOMINO, 2008; TSURUTA, 2008; MOURA JR; STEFFEN JR, 2008; LEUCAS, 2009). Dentre todos os métodos, neste trabalho é utilizado o de impedância eletromecânica. Este método utiliza as assinaturas de impedância (IS - do inglês *Impedance Signature*), onde essas são comparadas entre si a fim de identificar a presença de um dano.

Diante deste cenário, para a condução de um monitoramento de integridade estrutural baseado em impedância eletromecânica são utilizados conjuntos de elementos essenciais, são eles: rede de transdutores piezoelétricos, sistema para aquisição das IS, modelos para tomada de decisão e software para a visualização/acompanhamento do monitoramento. É inevitável o uso de um grande volume de dados para o emprego do monitoramento contínuo, devido à coleta periódica de IS. Também é um fator importante a utilização de modelos que monitorem as variações nas IS, uma vez que é necessária uma atuação ágil sobre danos estruturais. Portanto a utilização de recursos computacionais é vital para o desempenho do sistema.

Os modelos de monitoramento devem possuir alta detecção e precisão, além de possibilitar o emprego de interpretações e justificativas das decisões performadas, visando proporcionar uma melhoria continua dos modelos. Os modelos podem ser aplicados nas mais diversas tarefas, como regressão, avaliando a progressão de dano ou vida útil, classificação de danos e detecção de anomalias (MONAVARI, 2019; REZENDE *et al.*, 2020; BAO *et al.*, 2019).

Para garantir a performance dos modelos voltados para o monitoramento, é crucial realizar modelagem dos mesmos, considerando aspectos intrínsecos das tarefas a serem executadas. A utilização de ferramentas automatizadas para a modelagem proporciona um aumento de produtividade, atingindo resultados adequados em um intervalo de tempo mínimo, aumento da confiabilidade dos resultados devido a possibilidade de interpretar a forma como o modelo tomou determinada decisão e melhorando ou atualizando os modelos conforme a necessidade. A alta capacidade em identificar padrões dos modelos utilizados possibilitou atingir ótimos desempenhos no monitoramento de integridade, considerando dano e variação de temperatura sem a necessidade de nenhum tratamento de compensação, redução de ruídos ou extração de métricas de dano, ou seja, utilizando apenas faixas frequenciais.

Em seguida, o modelo é então colocado em produção, utilizando novas ferramentas que garantam e simplifiquem o processo de análise, implantação e monitoramento. Dessa forma, para se provisionar os modelos desenvolvidos são empregados os processos de virtu-

alização e conteinerização. A conteinerização se dá pelo uso da ferramenta Docker, a qual possibilita a utilização mais eficiente dos recursos do sistema, facilitando ciclos de entrega e portabilidade de aplicativos, unificando todas as versões e ferramentas em um ambiente que facilita o seu desenvolvimento e implantação (CELESTI *et al.*, 2016; UPHILL *et al.*, 2017; SHAH; DUBARIA, 2019; ISLAM *et al.*, 2019; DOCKER, 2020b).

Pra a disponibilização dos serviços utiliza-se os recursos da computação em nuvem proporcionando inúmeros benefícios computacionais. A computação em nuvem (*cloud computing*), é um modelo que permite o acesso conveniente e sob demanda da rede de recursos de computação (por exemplo, redes, servidores, armazenamento, aplicativos e serviços) que podem ser rapidamente provisionados e lançados com o mínimo de esforço de gerenciamento e interação com o provedor do serviço (MELL; GRANCE, 2009; DILLON; WU; CHANG, 2010; XU, 2020).

Este trabalho apresenta uma revisão bibliográfica acerca das ferramentas para desenvolvimento, análise, monitoramento e provisionamento de modelos voltados para o monitoramento de integridade estrutural. São aplicadas ferramentas de desenvolvimento automático de inteligência computacional e suas respectivas análises e interpretações das decisões dos modelos. Também apresenta o provisionamento de uma aplicação web de monitoramento estrutural, utilizando as tecnologias de conteinerização e computação em nuvem, assim como as ferramentas e teorias necessárias para realização do provisionamento.

1.1 Estrutura do Trabalho

Este trabalho foi dividido em nove capítulos para uma melhor compreensão do leitor a respeito dos temas abordados.

No capítulo 1 é realizada uma introdução a respeito do monitoramento da integridade estrutural buscando relacionar a necessidade do desenvolvimento de modelos aliado ao provisionamento dos mesmos. Também são apresentados os principais objetivos desta dissertação.

No capítulo 2 é feita uma revisão bibliográfica para um melhor desenvolvimento do trabalho, onde serão apresentados os principais conceitos relacionados aos materiais inteligentes e processo de monitoramento da integridade estrutural. Os conceitos e ferramentas de aquisição de dados também são apresentados nesse capítulo.

No capítulo 3 são apresentados os conceitos teóricos e práticos relacionados a inteligência artificial. Também são destacados os tipos de aprendizagem, tipos de tarefas, ciclo de vida dos modelos e o ferramental e técnicas para o desenvolvimento, análise e implantação de modelos.

No capítulo 4 é apresentado os elementos e configurações do experimento utilizado

para aplicação e desenvolvimento dos modelos.

No capítulo 5, 6 e 7 são apresentadas as possíveis tarefas e alternativas para o emprego de modelos voltados ao monitoramento de integridade estrutural, bem como as avaliações e interpretações dos mesmos.

No capítulo 8 é apresentado o desenvolvimento da aplicação web utilizando a abordagem baseada na classificação binária supervisionada, com a finalidade de colocar em produção o aplicativo utilizando os recursos apropriados para um sistema de monitoramento.

No capítulo 9 é então apresentada uma conclusão da contribuição, explanando os conceitos aplicados e desenvolvidos, buscando apresentar a relevância e o impacto das ferramentas e procedimentos aqui apresentados. Também é apresentada uma perspectiva para trabalhos futuros, visando uma continuação do desenvolvimento desta contribuição e das ferramentas e conceitos aplicados ao monitoramento de integridade estrutural.

Capítulo 2

MONITORAMENTO DE INTEGRIDADE ESTRUTURAL

Neste Capítulo é abordado inicialmente os conceitos acerca dos materiais inteligentes. Dado suas características são apresentados os processos para o emprego dos monitoramentos de integridade estrutural. Em seguida, os conceitos de impedância são introduzidos. Então o método de SHM baseado em impedância eletromecânica é abordado, apresentando seu funcionamento e características. Também são apresentados os principais efeitos nas IS, os modelos estatísticos comumente utilizados e os analisadores de impedância responsáveis por coletar as IS.

2.1 Materiais Inteligentes

Materiais inteligentes são conhecidos pela sua capacidade de adaptação a estímulos externos, como cargas ou ambiente, com inteligência inerente (KAMILA, 2013). Eles são empregados nas mais diversas indústrias, como aeroespacial, mecânicas e civis, e outras áreas como ciência dos materiais, inteligência artificial, nanotecnologia e biotecnologia.

Segundo Rogers (1989) (Workshop do Escritório de Pesquisa do Exército dos EUA), os materiais inteligentes desempenham alterações em seu comportamento físico de maneira precisa em resposta à contribuição de estímulos. Os estímulos podem ser acústicos, tensão mecânica, mudança de temperatura, campos elétricos e magnéticos, fotoquímicos e químicos ou radiação nuclear. A variável relacionada ao comportamento físico pode ser frequência, forma, flexibilidade, rigidez, energia potencial, viscosidade ou amortecimento (MAURYA; RAWAT; JHA, 2020; XU; LIAO; LIU, 2019; SHANKER *et al.*, 2011).

Os materiais inteligentes podem ser divididos como ativos ou passivos. O material é considerado um material inteligente ativo se, sob a aplicação de estímulos, alterar suas propriedades geométricas ou do material, por exemplo, adesivos PZT, ligas com memória de forma, sensores magnetorrestritivos, fibras de dois componentes e fluído eletro-reológico.

Os materiais passivos são aqueles que possuem a capacidade inerente de transduzir energia, como as fibras ópticas (FAIRWEATHER, 1999; NEWNHAM; RUSCHAU, 1993).

Os sistemas que possuem esses materiais são classificados como estruturas inteligentes. Essas detêm cadeias de sensores e atuadores, mecanismos de controle e capacidade de detecção, controle e autorreparo. O sistema de materiais inteligentes consiste em vários processos e componentes, como obtenção de dados, comunicação de dados, unidade de comando e controle, aprendizado de máquinas e dentre outras (PARIHAR; KAJAL; PALLAVI, 2016).

O monitoramento de integridade estrutural SHM é uma das abordagens usadas para a implementação de métodos de detecção, localização e quantificação de danos para estruturas de engenharia. O princípio básico do método consiste em determinar a presença de falhas/danos nos materiais ou estruturas. Vale ressaltar as vantagens do método baseado em Impedância Eletromecânica (IE ou ISHM - do inglês *Impedance-based Structural Health Monitoring*) : um alto nível de confiança, devido à baixa intervenção humana; sensibilidade a danos incipientes e monitoramento contínuo da integridade estrutural de forma não invasiva.

Nessa técnica são utilizados elementos piezoelétricos feitos de Titanato-Zirconato de Chumbo (PZT). Estes dispositivos são fabricados de diversos tamanhos, possibilitando a utilização nas mais diversas estruturas (BANKS; SMITH; WANG, 1996a; PEAIRS, 2002a).

A produção desses elementos consiste na polarização de um material cerâmico, ou seja, elevar a temperatura do mesmo acima da temperatura de Curie (a qual os dipolos podem mudar de orientação na fase sólida), ao passo em que esses elementos são submetidos a elevados campos elétricos na direção da polarização desejada. Após essa etapa, o material é resfriado mantendo o campo elétrico na direção da polarização desejada e assim fixando o alinhamento dos dipolos (MOHEIMANI, 2003).

A piezoeletricidade dos tradutores é fator crucial para o método de ISHM, uma vez que esta possibilita a transição entre os domínios elétrico e mecânico. Essa transição entre domínios é caracterizada por dois efeitos. São eles:

- **Efeito Piezoelétrico Direto**: quando aplicada uma deformação mecânica sobre o transdutor, este gera um deslocamento de carga em seus terminais;
- **Efeito Piezoelétrico Inverso**: ao aplicar uma diferença de potencial sobre os terminais do transdutor, o mesmo gera uma deformação mecânica.

No método de ISHM ambos os efeitos são empregados para excitar dinamicamente a estrutura e então obter a resposta a esta excitação em termos da impedância eletromecânica. Vale ressaltar que para o método ISHM os transdutores comumente operam no espectro ultrassônico, evitando baixas frequências e ruídos comuns (BORGES, 2017).

As equações 2.1 e 2.2 formuladas por Voigt (1910), representam os efeitos piezoelétricos direto e inverso (BANKS; SMITH; WANG, 1996b). Equações 2.1 e 2.2 representam o efeito direto e inverso, respectivamente.

$$D = d\sigma + \varepsilon E_1; \tag{2.1}$$

$$S = s\sigma + dE_1, \tag{2.2}$$

onde D é o vetor das deformações $[N/m]$, $E1$ é o vetor do campo elétrico $[V/m]$, S corresponde ao tensor das deformações $[N/m^2]$, σ é o vetor das tensões $[N/m^2]$, $[d]$ é o tensor de tensão piezoeléctrica $[m/V]$, $[\varepsilon]$ é o tensor dielétrico do material e $[s]$ é a propriedade elástica do material piezoelétrico.

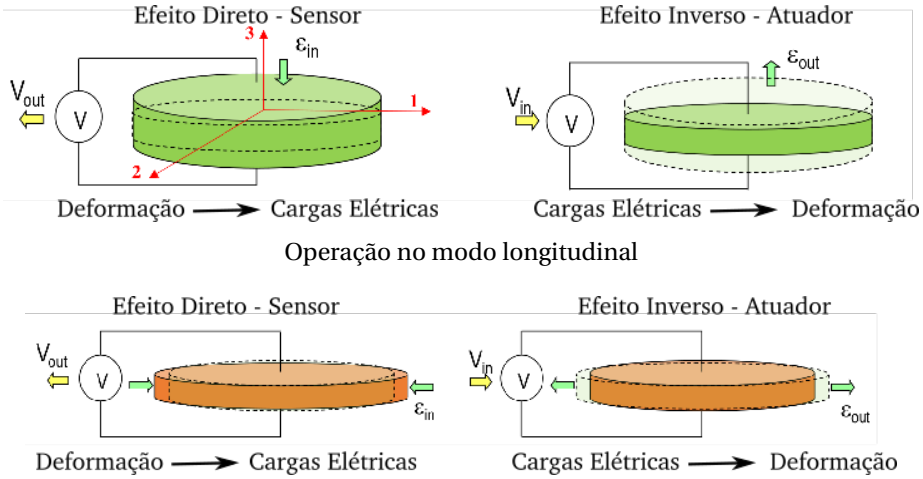
Dentre os materiais piezoelétricos mais utilizados, destaca-se um piezocerâmico, conhecido como Titanato Zirconato de Chumbo (PZT), um piezopolímero, conhecido como Fluórido de Polivinilideno (PVDF) e um compósito, chamado de MFC (*Macro Fiber Composite*). (BENTO *et al.*, 2018). A Figura 2.1 apresenta algumas pastilhas PZTs produzidas pela fabricante *Ultra-piezo*.

Figura 2.1 – Exemplos de pastilhas PZT fabricadas pela fabricante *Ultra-piezo*

Fonte: Retirado de http://www.ultra-piezo.com

A Figura 2.2 ilustra um material piezoelétrico de camada única usado para detectar e atuar de acordo com a direção da deformação. O material piezoelétrico pode ser generalizado para dois modos que são particularmente importantes na definição do coeficiente de acoplamento eletromecânico. O primeiro opera no modo longitudinal e o segundo opera no modo transversal. Como atuador piezoelétrico, no modo longitudinal, o campo elétrico é aplicado na direção '3' e o material é tensionado na direção de '3'. No modo transversal, o campo elétrico é aplicado na direção '3' e o material é tensionado na direção '1' ou perpendicular à direção do campo elétrico. Como sensor piezoelétrico, quando uma deformação é aplicada ao material na direção '3' (modo longitudinal) ou na direção '1' (modo transversal), é gerado o campo elétrico (HUYNH; DANG; KIM, 2017)

Figura 2.2 – Sensor e atuador PZT de uma única camada



Operação no modo longitudinal

Operação no modo transversal

Fonte – Adaptado de: (HUYNH; DANG; KIM, 2017)

Devido a capacidade de conversão de energia dos transdutores PZT, apenas um elemento é utilizado tanto como sensor, utilizando o efeito piezoelétrico direto, quanto como atuador, utilizando o efeito piezoelétrico inverso. Deve-se avaliar o ambiente que o PZT estará exposto, pois caso submetido a elevadas temperaturas (acima da temperatura de Curie) o mesmo pode perder as propriedades piezoelétricas (BANKS; SMITH; WANG, 1996a).

O método de monitoramento da integridade estrutural baseado em impedância eletromecânica (ISHM) foi desenvolvido baseado na capacidade de transição entre os domínios elétrico e mecânico, amparado pelos transdutores piezoelétricos ou pastilhas PZT, acoplados ou embebidos à uma estrutura a ser monitorada (SUN *et al.*, 1995a; PARK *et al.*, 2003).

2.2 Processo do Monitoramento da Integridade Estrutural

Os danos em um sistema estrutural são definidos como alterações intencionais ou não intencionais nas propriedades dos materiais, incluindo alterações nas condições de contorno e na conectividade do sistema, que afetam o desempenho atual ou futuro do sistema (INMAN *et al.*, 2005). Farrar e Worden (2007) definem dano como alterações de massa, rigidez e/ou amortecimento na estrutura. Para diferentes tamanhos e localizações dos danos e das cargas aplicadas ao sistema, os efeitos adversos desses, podem ter impactos imediatos ou levar algum tempo para alterar o desempenho do sistema. Em termos de escala de comprimento, todos os danos ocorrem no nível do material e, em condições adequadas de carregamento, progridem para danos no componente e subsequentemente ao sistema. Em termos de escalas de tempo, os danos podem acumular-se incrementalmente por longos períodos de tempo, como danos associados à fadiga ou corrosão (INMAN *et al.*, 2005).

Uma vantagem relevante para o monitoramento de integridade estrutural é a capacidade de identificação prematura de danos, possibilitando uma atuação contundente para contenção e prevenção dos possíveis problemas advindos de um dano estrutural.

Existem duas abordagens para análise de danos em estruturas: direta e indireta. Na abordagem direta o sistema em análise é simples, seu modelo matemático é conhecido, e uma amostra real de sinal do sistema é coletado e confrontado com o modelo. Na abordagem para problemas inversos, normalmente aplicado sobre sistemas complexos, não é conhecido o modelo matemático, dessa forma são comparados sinais do sistema ao longo do tempo, sendo alterações nesses sinais indicativos da presença de danos (MAIO, 2011).

Existem várias maneiras de descrever um processo de SHM. A descrição apresentada aqui acompanha a literatura sobre o tema observada em Farrar, Doebling e Nix (2001) e Farrar e Worden (2007). Segundo os autores citados, um processo de reconhecimento de padrões para monitoramento de integridade estrutural consiste em quatro passos sequenciais.

1. **Avaliação operacional:** Inicialmente é avaliado a capacidade de implantação do sistema para monitoramento e identificação de falhas. Esse passo é desenvolvido por questionários e inspeções;

2. **Aquisição, normalização e seleção de dados:** Seleciona-se tanto os sistemas para medições e extração de características, quanto o software. Também são selecionados métodos para eliminação de pertubações nos sinais devido as condições ambientais. Por fim são definidos como serão selecionados os dados para o processo de SHM.

3. **Reconhecimento de padrões e condensação das informações:** Nesta etapa são selecionados quais características ou padrões abstraem os principais estados da estrutura, como por exemplo, com dano e sem dano.

4. **Desenvolvimento de modelo estatístico:** E, finalmente, são selecionados métodos estatísticos para transformar características extraídas em decisões de nível de desempenho atualizadas.

Todas as etapas possuem uma dependência funcional onde caso haja alguma não conformidade em alguma das etapas, podem levar a interpretações errôneas dos danos, falsos positivos e não identificação dos danos.

Conforme Rytter (1993), Doebling, Farrar e Prime (1998), Maio (2011) a caracterização de falhas para um melhor entendimento, pode ser classificada nos seguintes níveis:

- Nível 1 – Detecta a existência da falha;
- Nível 2 – Detecta e localiza a falha;

- Nível 3 – Detecta, localiza e quantifica a falha;
- Nível 4 – Detecta, localiza, quantifica a falha e então estima a vida útil restante do equipamento.

E conforme proposto por Inman (2001), quando incorporado os materiais inteligentes, foram propostos três níveis adicionais:

- Nível 5 – Combina o nível 4 com estruturas inteligentes para auto-diagnóstico de falhas estruturais;
- Nível 6 – Combina o nível 4 com estruturas inteligentes e controle para formar um sistema de auto-reparo estrutural;
- Nível 7 – Combina o nível 1 com controle ativo e estruturas inteligentes para obtenção de um sistema simultâneo de controle e monitoramento.

2.2.1 Conceitos de Impedância Elétrica e Impedância Mecânica

A impedância elétrica é definida pela oposição ao fluxo de cargas elétricas em um determinado componente ou circuito elétrico, dos quais estão submetidos a uma tensão alternada, e pode ser calculada pela razão entre a tensão fornecida ao circuito ou componente e a corrente que passa por ele, a equação 2.3 apresenta essa relação. Esta pode ser divida em impedância capacitiva, referente aos capacitores, e impedância indutiva, referente aos indutores.

$$Z_e = \frac{V_e}{I_e}, \tag{2.3}$$

visto que Z_e é a impedância elétrica, V_e é a tensão elétrica alternada e I_e é a corrente.

Quando uma corrente alternada atravessa dispositivos com características indutivas, como, por exemplo, motores, transformadores e reatores, a obtenção do valor de impedância se dá pela equação 2.3 (JUNIOR, 2013). Dessa forma a magnitude da reatância indutiva varia conforme a frequência e quanto maior essa for, maior essa será. Já a equação 2.4 apresenta a relação entre essas grandezas (GIBILISCO, 2002).

$$X_L = 2 \cdot \pi \cdot \omega_e \cdot L, \tag{2.4}$$

onde L é a indutância expressa em Henries (H) e ω_e é a frequência de excitação em Hertz (Hz) do circuito de corrente alternada considerado.

A reatância capacitiva é uma imagem espelhada da reatância indutiva, que irá variar com a frequência (MOURA JR; STEFFEN JR, 2008). Ou seja, quanto maior for a frequência

menor será a reatância capacitiva. A equação 2.5 apresenta a relação entre essas grandezas. (GIBILISCO, 2002).

$$X_C = \frac{1}{2 \cdot \pi \cdot \omega_e \cdot C}, \tag{2.5}$$

onde C é a capacitância expressa em Farad (F) e ω_e é a frequência de excitação em (Hz) do circuito de corrente alternada considerado (IRWIN, 2005). A reatância capacitiva é inversamente proporcional a frequência, ou seja, quanto maior for a frequência, menor será a reatância capacitivas.

A impedância mecânica representa o quanto uma estrutura pode resistir ao movimento quando lhe é aplicada uma força. Desta forma, para se obter a impedância mecânica deve-se dividir a a força aplicada em um ponto do sistema pela velocidade com que este ponto se desloca (MASSOUD, 1985). A equação 2.6 expressa essa relação.

$$Z_m = \frac{F_m}{v_m}, \tag{2.6}$$

visto que Z_m é a impedância mecânica, F_m é a força aplicada e v_m é a velocidade. Esta é uma grandeza complexa, pois a força e a velocidade são grandezas vetoriais (com módulo e ângulo de fase).

Ao passar a impedância mecânica para o domínio da frequência (Fourier), nas frequências de ressonância há uma baixa resistência ao movimento, ou seja, menos força é necessária para movimentar a estrutura em uma dada velocidade (PALOMINO, 2008).

A impedância mecânica se relaciona com três grandezas básicas, são elas: o amortecimento mecânico, massa mecânica e a flexibilidade mecânica. O amortecimento está associado à parte real da impedância, e é responsável por dissipar a energia mecânica entregue ao sistema, a equação 2.7 expressa essa relação.

$$Z_m = \frac{R_m}{v_m}, \tag{2.7}$$

onde Z_m é a impedância mecânica, R_m é a força aplicada e v_m é a velocidade. Esta é uma grandeza complexa, pois a força e a velocidade são grandezas vetoriais (com módulo e ângulo de fase).

A massa mecânica é associada à parte imaginária positiva da impedância complexa. A característica principal dessa propriedade é acelerar um dispositivo, proporcionalmente a uma força aplicada sobre o mesmo. A equação 2.8 expressa essa relação.

$$f(t) = M_m \, a_m \tag{2.8}$$

onde M_m é a massa mecânica e a_m é a aceleração.

A flexibilidade mecânica está associada com a parte imaginária negativa da impedância. Um dispositivo se comporta como uma flexibilidade mecânica quando, se desloca devido a uma força aplicada. A equação 2.9 apresenta essa relação.

$$x(t) = C_m f(t), \tag{2.9}$$

onde $x(t)$ é o deslocamento e C_m é a flexibilidade mecânica. Normalmente é utilizado o inverso dessa grandeza, conhecida como rigidez (RABELO *et al.*, 2014). A equação 2.10 apresenta essa relação.

$$K = \frac{1}{C_m} \tag{2.10}$$

2.2.2 Técnica de SHM Baseado em Impedância Eletromecânica

O método de monitoramento da integridade estrutural baseado em impedância eletromecânica (ISHM) utiliza dos efeitos piezoelétricos dos elementos PZT para empregar o monitoramento. Estas pastilhas PZTs são acopladas na estrutura de modo a não atrapalhar seu comportamento dinâmico (MOURA JR; STEFFEN JR, 2008).

O principio básico desse método consiste em monitorar as variações das Assinaturas de Impedância (IS - do inglês *Impedance Signature*), das quais são sensíveis a alterações na impedância mecânica causados pela presença de dano. A impedância eletromecânica é utilizada uma vez que a medição da impedância mecânica das estruturas é difícil de ser obtida (PARK; INMAN, 2005). Essa IS é constante enquanto as características estruturais não se alterarem. O processo de monitoramento decorre da comparação da primeira IS obtida, chamada de *baseline*, com futuras IS obtidas ao longo do tempo de monitoramento da estrutura.

Os transdutores atuam no sistema tanto como sensor quanto como atuador, simultaneamente, onde, através do efeito inverso (efeito atuador), recebem uma diferença de potencial elétrico e deformam-se, gerando uma força de excitação sobre a estrutura ou sistema monitorado. Esta força de excitação gerar uma deformação sobre a pastilha, fazendo com que o efeito direto (efeito sensor) gere uma corrente elétrica. Dada a tensão aplicada sobre a pastilha PZT e a corrente gerada pela deformação da estrutura, pode-se obter a impedância elétrica do transdutor. A detecção de falhas esta associada com as alterações nessa impedância elétrica, pois alterações na impedância mecânica da estrutura monitorada resultam em uma mudança na impedância elétrica do transdutor que esta acoplado a estrutura monitorada (RABELO *et al.*, 2017).

A resposta dinâmica da estrutura representa somente a área local do sensor. A resposta mecânica da vibração da área onde se encontra a pastilha de PZT é transmitida ao sensor na forma de uma resposta elétrica. Então quando ocorre uma falha mecânica na es-

trutura, há mudanças na resposta dinâmica (dada pelo sinal da impedância), e consequentemente na resposta elétrica do PZT (PALOMINO, 2008). Um simples PZT, pode ser capaz de identificar uma falha até uma distância (radial) de 0,4 metros em estruturas compósitas e até 2 metros em estruturas de barra de um único metal (PARK *et al.*, 2003).

Para um correto desempenho do método, o sistema de monitoramento deve ser capaz de avaliar e identificar ruídos ambientais que possam gerar alterações na estrutura que não representem um dano (MOURA JR; STEFFEN JR, 2006). O efeito da temperatura é o que mais gera alterações nas IS (PALOMINO L *et al.*, 2011). Para compensar esse efeito, diversos estudos foram desenvolvidos, com a intenção de minimizar a taxa de falso dano (falso positivo) (BORGES, 2017; RABELO; NETO; STEFFEN JR, 2015; BAPTISTA *et al.*, 2014; KOO *et al.*, 2009; SUN *et al.*, 1995c).

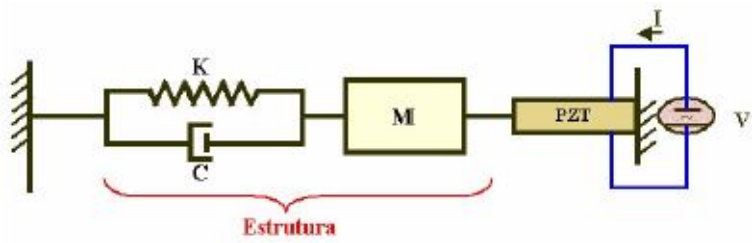
O estudo que sugere o emprego da técnica de impedância eletromecânica no monitoramento da integridade estrutural foi inicialmente abordado por Liang, Sun e Rogers (1994). Nos anos seguintes, diversos trabalhos nesta área foram escritos e as principais contribuições que se destacam são: Chaudhry *et al.* (1995a), Chaudhry *et al.* (1995b), Sun *et al.* (1995b), Park *et al.* (1999), Park, Cudney e Inman (1999), Park, Cudney e Inman (2000a), Park, Cudney e Inman (2000b), Park, Cudney e Inman (2001), Soh *et al.* (2000), Giurgiutiu e Zagrai (2000), Giurgiutiu e Zagrai (2002), Giurgiutiu, Zagrai e Bao (2002), Bhalla *et al.* (2002), Bhalla, Naidu e Soh (2003), Giurgiutiu e Zagrai (2005), Moura Jr e Steffen Jr (2008), Liu e Jiang (2009), Neto *et al.* (2011), Palomino L *et al.* (2011), Maio (2011), Oliveira (2013), Tahmasebpour (2014), Rabelo, Neto e Jr (2015), Rabelo *et al.* (2017), Tsuruta *et al.* (2017), Na e Baek (2018), Budoya e Baptista (2018), Antunes *et al.* (2019), Castro, Baptista e Ciampa (2019) e Castro, Baptista e Ciampa (2020).

O método de SHM baseado em impedância eletromecânica pode ser aplicado a qualquer estrutura complexa, onde os dados podem ser interpretados e analisados. O primeiro trabalho aplicado a detecção de danos foi o proposto por Annamdas e Yang (2012), que apresentou resultados de monitoramento da escavação do solo realizada para construção de um nova estação de transporte rodoviário de massa, utilizando a técnica ISHM. Os transdutores PZT foram instalados em uma estrutura de suporte temporária (para evitar o colapso do solo durante a construção), onde as assinaturas de impedância eram monitoradas por até um ano. Embora nenhum dano significativo tenha sido relatado na estrutura, verificou-se que a técnica EMI foi eficaz na detecção de alterações no carregamento causadas pelo solo circundante.

Um modelo que identifica e descreve o processo de medição da impedância eletromecânica para um grau de liberdade foi desenvolvido por Liang, Sun e Rogers (1994). Este modelo com o inverso da impedância eletromecânica, a admitância eletromecânica. A Figura 2.3 apresenta um exemplo para visualização do modelo de SHM baseado em impedância eletromecânica. Este pode ser simplificado em um sistema massa-mola com um grau de

liberdade.

Figura 2.3 – Modelo unidimensional do acoplamento eletromecânico utilizado para realizar o SHM baseado em impedância eletromecânica.



Fonte: Retirado de (MOURA JR; STEFFEN JR, 2008)

Segundo Liang, Sun e Rogers (1994), a admitância $Y(\omega)$ do PZT utilizado como atuador, é uma função combinada da impedância mecânica do PZT $Z_a(\omega)$ e da estrutura $Z(\omega)$. Ela é apresentada na equação 2.11:

$$Y(\omega) = \frac{I}{V} = i \cdot \omega \cdot a \cdot \left(\bar{\varepsilon}_{33}^{T}(1 - i\delta) - \frac{Z(\omega)}{Z(\omega) + Z_a(\omega)} d_{3x}^{2} \cdot \hat{Y}_{xx}^{E} \right) \tag{2.11}$$

onde: Y é a admitância elétrica; V é a voltagem de entrada no atuador PZT; I é a corrente de saída do PZT; a é a constante geométrica do PZT; d_{3x} é a constante de acoplamento piezoelétrico em uma direção x com deformação nula; $\hat{Y}_{xx}^{E}$ é o módulo complexo de Young do PZT com campo elétrico nulo; $\bar{\varepsilon}_{33}^{T}$ é a constante dielétrica complexa do PZT com tensão zero; ω é a frequência angular; Z é a impedância complexa da estrutura; Z_a é a impedância complexa do PZT e δ é o fator de perda dielétrica tangencial do PZT.

Segundo Moura Jr e Steffen Jr (2008) e Bitencourt e Steffen Júnior (2009), as propriedades mecânicas do PZT são constantes em toda vida útil do sistema de monitoramento, então, a partir da equação 2.11, a impedância elétrica do PZT é diretamente relacionada à impedância mecânica da estrutura. Desta forma, as variações na impedância elétrica são consideradas variações na impedância mecânica da estrutura e, assim uma presença de falha na estrutura. A parte real da impedância complexa medida é mais sensível às características de dano dentro da estrutura enquanto a parte imaginária é mais sensível à temperatura (SUN *et al.*, 1995b; BHALLA; NAIDU; SOH, 2003; PARK *et al.*, 2003; AFSHARI, 2012).

Segundo Park *et al.* (2003) a faixa de frequência normalmente utilizada para o emprego do monitoramento fica entre 30kHz e 250kHz. Dentro desse intervalo são utilizados métodos para seleção da melhor sub faixa de frequência, tentativa e erro ou procedimentos de otimização (BENTO *et al.*, 2018; MOURA JR; STEFFEN JR, 2008). Para Sun *et al.* (1995b), a identificação da melhor banda de frequência a ser monitorada sugere que as faixas acima de 200KHz são favoráveis para obter falhas localizadas, enquanto bandas menores que 70KHz são mais indicadas para áreas maiores de falhas. Baseado nos estudos de Borges (2017), Maio (2011), Park *et al.* (2003) a seleção do *range* de atuação baseado no material dependerá

do tipo de material, variações de temperatura, tipo de ensaio, pois para cada combinação destes haverá uma variação da faixa de frequência adequada.

2.2.3 Efeitos nas Assinaturas de Impedância

Existem muitas variáveis que podem interferir nos resultados de um sistema que realiza o monitoramento de integridade estrutural. Identificar, separar ou mesmo eliminar esses erros é muito importante, assegurando a confiabilidade das análises e respostas.

Um dos efeitos mais impactantes nas assinaturas de impedância é o da temperatura como citado anteriormente na secção 2.2.2. Este afeta tanto as propriedades da pastilha PZT (constante dielétrica complexa) quanto as propriedades da estrutura mecânica (módulo de *Young*), essas alterações são descritas pela equação 2.11. Normalmente a temperatura afeta a forma dos picos de ressonância, vales das antirressonâncias e desloca horizontalmente e verticalmente a IS. Dessa forma, o efeito da temperatura pode ser confundido com um dano, se o mesmo não for isolado da verdadeira assinatura de impedância do sistema.

Para compensar os efeitos da temperatura, métodos são empregados. Tsuruta *et al.* (2017) e Rabelo *et al.* (2017) utilizam técnicas de normalização e algoritmo de otimização híbrido para compensar o efeito da temperatura. Lim *et al.* (2011) desenvolveram uma nova técnica de detecção de danos usando a normalização de dados baseada na Análise de Componentes Principais do Kernel (KPCA - do inglês *Kernel Principal Component Analysis*) com o objetivo de melhorar a detectabilidade e minimizar diagnósticos de falsos positivos, ocasionados por variações de temperatura. Também relatam o emprego das Redes Neurais Artificiais (RNA) , e destacam que essas requerem grandes quantidades de dados de treinamento para uma operação adequada. Caso contrário, poderão ocorrer diagnósticos falsos se os dados de treinamento não incluírem uma condição ou faixa de temperatura específica.

Koo *et al.* (2009) utilizam uma técnica de compensação de temperatura proposta anteriormente com base na correlação cruzada entre os dados de impedância de referência e os dados de impedância subsequentes. Neste estudo, é utilizado o Coeficiente de Correlação Cruzada (CC - do inglês *Correlation Coefficien*) após uma mudança de Frequência Efetiva (EFS - do inglês Effective Frequency Shift), que é definida como a mudança de frequência que faz com que dois dados de impedância tenham a correlação máxima.

Outros efeitos que afetam as IS são os carregamentos estático e dinâmico. Estes causam variações na magnitude dos picos e vales das IS. A identificação destes efeitos é importante pois o funcionamento da estrutura sob estes efeitos não é comprometido.

Uma carga estática é adicionada gradualmente sobre à estrutura sem causar mudanças no tempo. Os efeitos do carregamento estático normalmente são utilizados para simular danos à estrutura de modo a não danifica-la, ou seja, por meio de um Ensaio Não Destrutivo (END). Para a condução desses ensaios, uma massa é integrada à estrutura por meio

de adesivos, acoplamentos magnéticos e ou acoplamentos mecânicos, permitindo assim a realização de testes de integridade estrutural sem danificar a estrutura em análise (PEAIRS, 2006). Segundo Maio (2011) e Cavalcante, SCHONELL e NETO (2016), as métricas de danos possuem a capacidade para compensar as interferências do carregamento estático.

O carregamento dinâmico é definido como variações ao longo do tempo da aplicação de cargas cuja a magnitude, direção e posição alterem o deslocamento, velocidade e aceleração de resposta da estrutura (BORGES, 2017; SILVA *et al.*, 2011). Normalmente estes são provocados por impacto de objetos, dinâmica do vento, vibrações do solo e dentre outros. Essas fontes geradoras de carregamento dinâmico impactam em alterações na estrutura ou sistema mecânico em análise o que implica em erros na assinatura de impedância obtida a partir da estrutura.

Por último, um efeito a ser considerado segundo Borges (2017) é o tipo de sensor utilizado. Existem variações consideráveis na frequência de operação, nas propriedades mecânicas, térmicas e eletromagnéticas, dependendo do tipo de PZT usados. Cada tipo de sensor/atuador PZT possui sensibilidade e deformação diferentes, dependentes das dimensões e composições do material. Assim, implicando em respostas e resultados diferentes para cada aplicação. Modelos matemáticos são empregados para compensação desses efeitos, a fim de identificar, separar e/ou eliminar as interferências nas assinaturas de impedância.

2.2.4 Modelos Estatísticos para SHM

As ISs representam o estado de integridade da estrutura. Entretanto, não fornecem informações quantificáveis sobre a presença e o estado do dano na estrutura. Dessa forma é necessário empregar um índice que tenha a capacidade e sensibilidade na identificação de um dano. Tem-se na literatura diversos índices, também denominados métricas de dano, são eles (AFSHARI, 2012; PALOMINO *et al.*, 2008; TSENG; NAIDU, 2002):

- RMSD: desvio médio da raiz quadrada;
- ASD: diferença média quadrada;
- CC: coeficiente de correlação;
- MAPD: desvio percentual absoluto médio;
- M: soma das diferenças quadráticas.

Sun *et al.* (1995b), determinam que os gráficos das assinaturas de impedância não possuem a capacidade de expressar a presença de um dano quantitativamente, porém, com

a avaliação por meio quantitativo, a partir da aplicação de uma métrica, é possível identificar um dano. Uma delas é o RMSD, descrita pela equação 2.12.

$$M_{RMSD} = \sum_{i=1}^{n} \sqrt{\frac{[Re(Z_{i,1}) - Re(Z_{i,2})]^2}{[Re(Z_{i,1}]^2}}, \tag{2.12}$$

onde: M representa a medida da falha; $Z_{i,1}$ representa sinal da impedância medido pelo PZT quando a estrutura está sob condições saudáveis (*baseline*); $Z_{i,2}$ representa o sinal da impedância medido pelo PZT quando a estrutura está sob condição de falha ou em estados posteriores de análise; i representa o *range* de frequência e n representa a quantidade de pontos frequenciais a serem analisados. Tais variações irão representar uma maior aproximação da falha quanto maior seja o valor da medida da falha (MOURA JR; STEFFEN JR, 2008).

Segundo Raju (1997), a métrica da diferença média quadrada é empregada pelo método da impedância eletromecânica com a finalidade de quantificar um dano. Sua formulação matemática é apresentada na equação 2.13. Com o uso desta métrica de dano, também busca-se tirar o efeito das variações da amplitude devidas a mudanças no meio ambiente.

$$ASD = \sum_{i=1}^{n} \left[Re(Z_{1,i}) - (Re(Z_{2,i}) - \delta)\right]^2, \tag{2.13}$$

onde: δ é a diferença das médias de cada um dos sinais, como pode ser visto a partir da equação 2.14.

$$\delta = Re(\bar{Z_1}) - Re(\bar{Z_2}) \tag{2.14}$$

O desvio do coeficiente de correlação é usado para quantificar a diferença entre dois conjuntos de dados (GIURGIUTIU; ZAGRAI, 2005). A formulação matemática é dada pela diferença apresentada pela equação 2.15.

$$CCD = 1 - CC \tag{2.15}$$

onde: CCD é o desvio do coeficiente de correlação e CC é o coeficiente de correlação dado pela equação 2.16.

$$CC = \frac{1}{n} \sum_{i=1}^{n} \frac{(Re(Z_{1,i}) - Re(\bar{Z_1}))(Re(Z_{2,i}) - Re(\bar{Z_2}))}{S_{Z_1} S_{Z_2}} \tag{2.16}$$

em que S_{Z_1} é o desvio padrão do sinal de impedância da referência e S_{Z_2} é o desvio padrão do sinal de impedância a ser comparado. Nos casos em que o coeficiente de correlação é igual a 1 os sinais possuem correlação máxima.

Já a métrica de desvio percentual da média absoluta utilizada por Tseng e Naidu (2002), avalia a diferença dos sinais em cada ponto dos dados de medição. Sua formulação é apresentada pela equação 2.17 .

$$MAPD = \sum_{i=1}^{n} \left| \frac{(Re(Z_{1,i}) - Re(Z_{2,i}))}{Re(Z_{1,i})} \right| \tag{2.17}$$

Observa-se que o $MAPD$ é similar ao $RMSD$ definido na equação 2.12, pois ambos avaliam as diferenças dos sinais em cada ponto dos dados da medição (PALOMINO, 2008).

A métrica de dano é definida como a soma das diferenças quadráticas das mudanças de impedância para cada frequência. Este índice simplifica a interpretação das variações de impedância e fornece informações do sinal de resposta em impedância para a região de influência do atuador. A falha métrica M é dada pela equação 2.18.

$$M = \sum_{i=1}^{n} \left[(Z_{i,1}) - (Z_{i,2}) \right]^2 \tag{2.18}$$

em que $Z_{i,1}$ é a impedância medida na estrutura intacta, $Z_{i,2}$ é a impedância medida na estrutura em condições normais de operação e i é o número de pontos utilizados na aquisição do sinal (PARK; CUDNEY; INMAN, 2000a).

2.2.5 Analisadores de Impedância

As assinaturas de impedância são obtidas através do analisador de impedância. Os dados adquiridos são transferidos para o computador para uma posterior análise e avaliação. No princípio do desenvolvimento do método de monitoramento existiam alguns problemas para a sua aplicação, como por exemplo, que os equipamentos eram grandes (cerca de 25kg) e caros (cerca de US$ 40 mil), como os HP4194A e HP4294A (BAPTISTA; FILHO, 2009). Esses equipamentos possuíam uma alta precisão e ainda são utilizados (RUGINA *et al.*, ; ROTH; GIURGIUTIU, 2017; WANG *et al.*, 2018).

Atualmente, como alternativa a estes analisadores de impedância que não são mais comercializados, existe o *E4990A* da *Keysight Technologies*. Seu preço está na faixa de US$ 30 mil. A Figura 2.4 é apresentado o modelo de analisador de impedância *HP 4194 A*.

Figura 2.4 – Analisador de impedância *HP 4194A*

Fonte: Retirado de (AFSHARI, 2012)

Como alternativas, pode-se citar os analisadores de impedância de baixo custo *PmodIA* e *Eval - AD5933*, eles serão mais detalhados nos tópicos posteriores.

Com o intuito de tornar o monitoramento baseado em impedância eletromecânica mais prático, portátil, economicamente viável e adequado a aplicações práticas, foram desenvolvidos sistemas alternativos buscando superar as desvantagens encontradas nos sistemas convencionais de aquisição de assinaturas de impedância. Um dos primeiros sistemas desenvolvidos foi o proposto por Peairs (2002b), Peairs, Park e Inman (2004) seguido por Xu e Giurgiutiu (2005), Baptista e Filho (2009), Bhalla *et al.* (2009), Panigrahi, Bhalla e Gupta (2010), Bitencourt *et al.* (2010), Ledesma *et al.* (2012) e Piasecki, Chabowski e Nitsch (2016). O Quadro 2.1 apresenta as principais características dos sistemas citados acima.

Autor	**Descrição**
(PEAIRS, 2002b; PEAIRS; PARK; INMAN, 2004)	Circuito baseado em amplificadores operacionais, juntamente com um analisador de espectro para calcular a transformada discreta de Fourier DFT.
(XU; GIURGIUTIU, 2005)	Circuito baseado em resistores, onde é necessário um gerador de funções para excitar o PZT e um Dispositivo de Aquisição DAQ que realiza a aquisição da assinatura.
(BAPTISTA; FILHO, 2009)	Circuito utiliza apenas um dispositivo DAQ para excitar o PZT em série com um resistor e um Conversor Analógico Digital ADC para adquirir a resposta do PZT. O processamento como DFT é realizado pelo computador.
(BHALLA *et al.*, 2009)	Circuito utiliza um gerador de função para excitar o PZT e um multímetro digital para avaliação da impedância.
(PANIGRAHI; BHALLA; GUPTA, 2010)	Circuito utiliza um gerador de função para excitar o PZT e um osciloscópio digital para avaliação e coleta das assinaturas de impedância.
(BITENCOURT *et al.*, 2010)	Circuito utiliza uma placa de aquisição de dados USB-6259 que realiza a excitação do PZT e colhe a resposta do mesmo. O princípio de funcionamento consistem em medir a impedância de um resistor em série com o PZT.
(LEDESMA *et al.*, 2012)	Circuito utiliza um microcontrolador e um Sinal Senoidal Sintetizado DDS para a excitação do PZT, por meio de um ADC do próprio microcontrolador a resposta do PZT é armazenada.
(PIASECKI; CHABOWSKI; NITSCH, 2016)	Circuito utiliza um microcontrolador que realiza a partir dos DAC e ADCs integrados para excitar e coletar as respostas do PZT. Este não utiliza um resistor para avaliação da variação da impedância.

Fonte – o autor.

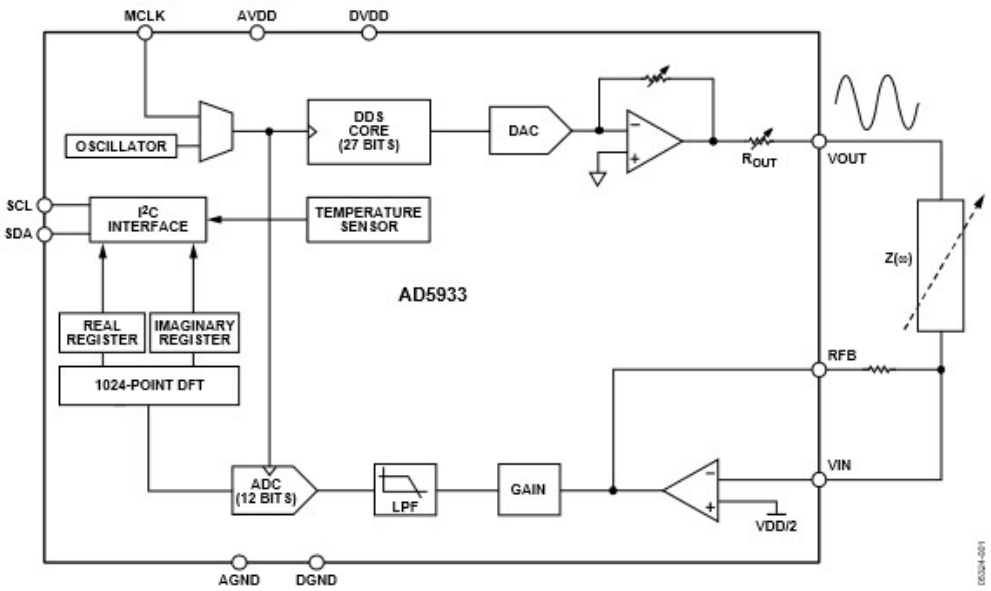
Quadro 2.1 – Autores e descrição dos circuitos utilizados para aquisição de sinais

Um dos conversores de impedância mais utilizados como alternativa aos analisadores de impedância da *HP*, são os baseados no circuito integrado AD5933, da empresa Analog Devices, que possui um preço estimado em cerca de US$ 39. Um dispositivo que

utiliza esse circuito integrado é o *PmodIA*, da empresa *Digilent*, usado para medições de impedância elétrica e eletromecânica, uma vez que possui um Sintetizador Digital Direto (DDS - do inglês *Direct Digital Synthesizer*) e um Conversor Digital-Analógico (CDA) integrados.

O princípio de seu funcionamento consiste em gerar um sinal senoidal a partir de um cristal interno, para a excitação da pastilha PZT e o conversor CDA amostra o sinal de resposta. O próprio Circuito Integrado (CI) possui um Processador Direto de Sinais (DSP - do inglês *Digital Signal Processors*), o qual realiza uma Transformada Rápida de Fourier (FFT - do inglês *Fast Fourier Transform*), que retorna as componentes da impedância mensurada para cada banda de frequência. A Figura 2.5 apresenta o diagrama simplificado do CI AD5933.

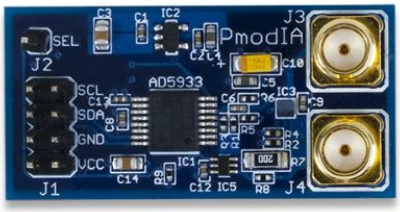
Figura 2.5 – Diagrama AD5933



Fonte – (DEVICES., 2008)

A placa PmodIA utiliza o CI AD5933 para desempenhar a conversão de impedância, embarcando algum componentes importantes para a realização de coletas e disponibilizando a comunicação protocolo I^2C para programação dos registros. A Figura 2.6 apresenta o conversor de impedância em questão.

Figura 2.6 – Conversor de impedância *PmodIA*

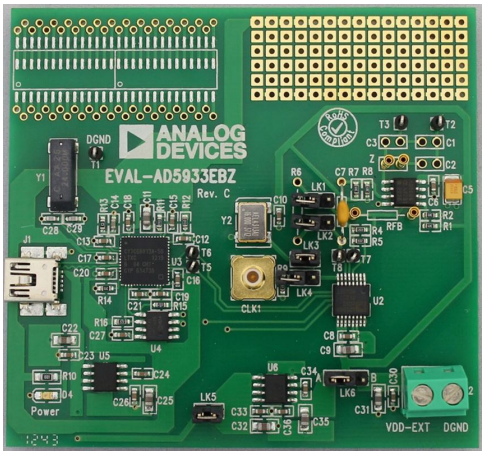


Fonte: Retirado de (DIGILENT, 2016)

A empresa do CI também desenvolveu uma alternativa barata para realizar a medição de impedâcia eletromecânica que é a placa de modelo $EvalAD5933EBZ$. Como a PmodIA, essa placa também possui o *CI AD5933*. Tem como diferencial permitir a utilização de uma fonte de excitação externa, assim aumentando o *range* de frequência limitado de até 100kHz para o *range* da fonte externa.

Para realizar aquisições de impedância, a placa deve operar com tensão de 2,7 V até 5,5 V. A qual normalmente utiliza a alimentação da porta USB do computador. Este registra a magnitude, parte real e imaginária da impedância (SEPEHRY; SHAMSHIRSAZ; BASTANI, 2011; DEVICES, 2017). A Figura 2.7 apresenta a placa de aquisição de dados em questão.

Figura 2.7 – Analisador de impedância *Eval - AD5933EBZ*



Fonte: (DEVICES., 2008)

Capítulo 3

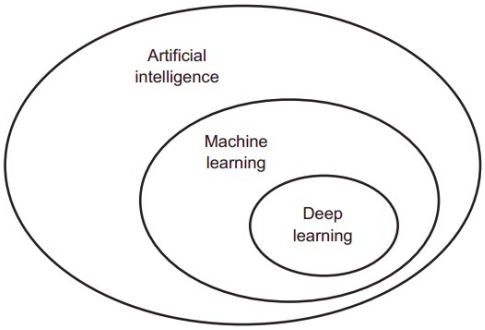
FUNDAMENTAÇÃO TEÓRICA SOBRE INTELIGENCIA ARTIFICIAL

Neste Capítulo é contextualizado a área de inteligência artificial, aprendizado de máquina e aprendizado profundo. É destacado os conceitos fundamentais como tipo de tarefa, tipo de aprendizado e avaliação dos modelos, todos esses relacionados ao aprendizado de máquina. Também são abordados os principais desafios para o aprendizado de máquina, assim como o seu ciclo de vida. Em seguida, são apresentadas as ferramentas para o desenvolvimento, análise, interpretação e implantação de modelos de aprendizado de máquina. Por fim, o estado da arte é destacado através das contribuições que relacionam as áreas de monitoramento de integridade estrutural com o aprendizado de máquina.

3.1 Inteligencia Artificial (*Artificial Inteligence*)

Inteligência Artificial (AI - do inglês *Artificial Inteligence*) nasceu em 1950 com a ideia de que os computadores fossem capazes de pensar, a qual trouxe uma definição para esse campo de estudo como: “o esforço para automatizar tarefas intelectuais normalmente executadas por seres humanos”. AI abrange os campos de Aprendizado de Maquina (ML - do inglês *Machine Learning*) e Aprendizado Profundo (DL - do inglês *Deep Learning*) (FRANCOIS, 2018). A Figura 3.1 apresenta essas relações.

Figura 3.1 – Inteligencia artificial, aprendizado de maquina e aprendizado profundo



Fonte – o autor.

No final dos anos 80 um sistema de AI era desenvolvido por um conjunto de regras explícitas para manipulação do conhecimento, bem como os sistemas especialistas. Esses processos eram comummente denominados de Inteligência Artificial Simbólica (FRANCOIS, 2018).

Com o aumento da complexidade das tarefas como reconhecimento de imagens, classificação de sinais e tradução de idiomas, ficou inviável a utilização da AI simbólica, devido a necessidade da definição de grandes e complexas regras. Desta forma, fez-se necessário o desenvolvimento de outras técnicas de aprendizado de máquina.

3.2 Aprendizado de Máquina (*Machine Learning*)

O surgimento do ML teve sua motivação inicial nas perguntas feitas por seu criador Alan Turing em TURING (1950): Se os computadores eram capazes de pensar ou até criar regras apenas olhando para os conjuntos de dados.

Com essas perguntas um novo paradigma foi criado, onde o conceito anterior de inserir dados, definir base de regras e então obter os resultados é substituído. Os dados e as respostas desejadas são avaliados e com isso uma base de regras é criada, a qual poderá ser utilizada sobre outro conjunto de dados. Ou seja, um sistema de aprendizado de máquina é treinado em vez de programado explicitamente.

Segundo Chollet (2018), para construir um aprendizado de máquina são necessárias 3 grupos de informações: dados de entrada, exemplos de saídas esperadas e um modo de avaliar se o procedimento está acertando os alvos de modo que está avaliação realimente o sistema, induzindo o aprendizado.

Todos os algoritmos de aprendizado de máquina consistem em encontrar automaticamente transformações que mapeiam dados em representações mais úteis para uma determinada tarefa. Normalmente os algoritmos buscam mapear os dados em um espaço de busca predefinido chamado de espaço de hipótese.

Uma definição utilizada na literatura para aprendizado de maquina é o seguinte: um programa de computador aprende com a experiência ou aprendizagem E, em relação a alguma classe de tarefas T, medida por um desempenho P, se seu desempenho nas tarefas T, conforme medido por P, melhora com a experiência ou aprendizagem E (BENGIO; GOODFELLOW; COURVILLE, 2017; MITCHELL, 1997)

3.3 Aprendizado Profundo (*Deep Learning*)

O DL é um subcampo específico do ML. Outros nomes apropriados para o campo poderiam ser aprendizagem de representações em camadas e aprendizagem de representações hierárquicas. O aprendizado profundo moderno frequentemente envolve dezenas ou mesmo centenas de camadas sucessivas de representações e todas são aprendidas automaticamente com a exposição aos dados de treinamento. Enquanto isso, outras abordagens de aprendizado de máquina tendem a se concentrar no aprendizado de apenas uma ou duas camadas de representações dos dados; portanto, às vezes são chamados de Aprendizado Superficial (*shallow learning*). O nome aprendizado profundo não é uma referência a nenhum tipo de compreensão mais profunda alcançada pela abordagem; ao contrário, representa simplesmente a ideia de sucessivas camadas de representações. (CHOLLET, 2018).

A aprendizagem dos modelos é aplicada em redes neurais, que é derivada da neurobiologia, inspirada no funcionamento do cérebro. Entretanto, o aprendizado profundo não é considerado um modelo cerebral.

A ideia principal da utilização de múltiplas camadas nos modelos é o aumento da abstração na observação dos dados, onde estes são avaliados de diferentes perspectivas, afim de filtrar as informações e obter um maior grau de associação entre as entradas e as saídas.

3.4 Tarefas no Aprendizado de Máquina

Conforme a definição dada por Bengio, Goodfellow e Courville (2017), Mitchell (1997) na Secção 3.2, as tarefas determinam o tipo do problema em análise. Nesta definição relativamente formal da palavra “tarefa”, o processo de aprendizagem em si não é a tarefa. Aprender é o meio de atingir a capacidade de realizar uma dada tarefa.

As tarefas de aprendizado de máquina geralmente são descritas em termos de como o sistema de aprendizado de máquina deve processar as instâncias ou exemplos. Suponha uma coleção de recursos ou características (*features*) que foram medidos quantitativamente a partir de algum objeto ou evento que o sistema de aprendizado de máquina processe. Normalmente representa-se uma instância como um vetor $x \in \mathbb{R}^n$, onde cada entrada x_i do vetor é outra característica. Por exemplo, os recursos de uma imagem geralmente são os valores dos pixels da imagem.

Muitos tipos de tarefas podem ser resolvidos com o aprendizado de máquina. Algumas das tarefas de aprendizado de máquina mais comuns incluem:

- **Classificação**: neste tipo de tarefa, o programa de computador é solicitado a especificar a qual das k categorias algumas entradas pertencem. Para resolver esta tarefa, o algoritmo de aprendizagem é geralmente solicitado a produzir uma função $f : \mathbb{R}^n \to \{1, ..., k\}$. Quando $y = f(x)$, o modelo atribui uma entrada descrita pelo vetor x a uma categoria identificada pelo código numérico y. Existem outras variantes da tarefa de classificação, por exemplo, onde f gera uma distribuição de probabilidade sobre as classes. Um exemplo de tarefa de classificação é o reconhecimento de objetos, em que a entrada é uma imagem (geralmente descrita como um conjunto de valores de brilho de pixel) e a saída é um código numérico que identifica o objeto na imagem.

- **Classificação com entradas ausentes**: A classificação se torna mais desafiadora se o programa de computador não tiver garantia de que todas as medições em seu vetor de entrada sempre serão fornecidas. Para resolver a tarefa de classificação, o algoritmo de aprendizagem só precisa definir um mapeamento de função única de uma entrada vetorial para uma saída categórica. Quando algumas das entradas podem estar faltando, em vez de fornecer uma única função de classificação, o algoritmo de aprendizagem deve aprender um conjunto de funções. Cada função corresponde a classificar x com um subconjunto diferente de suas entradas ausentes. Esse tipo de situação surge frequentemente no diagnóstico médico, porque muitos tipos de exames médicos são caros ou invasivos. Uma maneira de definir com eficiência um conjunto tão grande de funções é aprender uma distribuição de probabilidade sobre todas as variáveis relevantes e, em seguida, resolver a tarefa de classificação marginalizando as variáveis ausentes. Com n variáveis de entrada, pode-se obter todas as 2^n funções de classificação diferentes necessárias para cada conjunto possível de entradas ausentes, sendo preciso aprender uma única função que descreve a distribuição de probabilidade conjunta.

- **Regressão**: neste tipo de tarefa, o programa de computador é solicitado a prever um valor numérico dado alguma entrada. Para resolver esta tarefa, o algoritmo de aprendizagem é solicitado a produzir uma função $f : \mathbb{R}^n \to \mathbb{R}$. Este tipo de tarefa é semelhante à classificação, exceto que o formato de saída é diferente. Um exemplo de uma tarefa de regressão é a previsão do valor de sinistro esperado que uma pessoa segurada fará (usado para definir prêmios de seguro) ou a previsão de preços futuros de títulos. Esses tipos de previsões também são usados para negociação algorítmica.

- **Transcrição**: neste tipo de tarefa, o sistema de aprendizado de máquina é solicitado a observar uma representação relativamente não estruturada de algum tipo de dado

e transcrevê-la em uma forma textual discreta. Por exemplo, no reconhecimento óptico de caracteres, o programa de computador é mostrado uma fotografia contendo uma imagem de texto e é solicitado a retornar esse texto na forma de uma sequência de caracteres (por exemplo, em formato ASCII ou Unicode). O Google Street View usa aprendizado profundo para processar números de endereço dessa forma (GOODFELLOW *et al.*, 2013). Outro exemplo é o reconhecimento de voz, em que o programa de computador recebe uma forma de onda de áudio e emite uma sequência de caracteres ou códigos de identificação de palavras que descrevem as palavras que foram faladas na gravação de áudio. O aprendizado profundo é um componente crucial dos sistemas modernos de reconhecimento de voz usados em grandes empresas, incluindo Microsoft, IBM e Google (HINTON *et al.*, 2012).

- **Tradução automática**: em uma tarefa de tradução automática, a entrada já consiste em uma sequência de símbolos em algum idioma, e o programa de computador deve convertê-la em uma sequência de símbolos em outro idioma. Isso é comumente aplicado a línguas naturais, como a tradução do inglês para o francês. A aprendizagem profunda tem um impacto importante neste tipo de tarefa (SUTSKEVER; VINYALS; LE, 2014; BAHDANAU; CHO; BENGIO, 2014).

- **Saída estruturada**: as tarefas de saída estruturada envolvem qualquer tarefa em que a saída seja um vetor (ou outra estrutura de dados contendo vários valores) com relacionamentos importantes entre os diferentes elementos. Esta é uma categoria ampla e inclui as tarefas de transcrição e tradução, mas também muitas outras tarefas. Um exemplo é o mapeamento de uma frase de linguagem natural em uma árvore que descreve sua estrutura gramatical e marcando os nós das árvores como sendo verbos, substantivos ou advérbios e assim por diante. Outro exemplo é a segmentação de imagens em termos de pixel, em que o programa de computador atribui cada pixel de uma imagem a uma categoria específica (COLLOBERT, 2011).

- **Detecção de anomalias**: neste tipo de tarefa, o programa de computador examina um conjunto de eventos ou objetos e sinaliza alguns deles como incomuns ou atípicos. Um exemplo de tarefa de detecção de anomalias é a detecção de fraude de cartão de crédito. Ao modelar seus hábitos de compra, uma administradora de cartão de crédito pode detectar o uso indevido de seus cartões. Se um ladrão roubar seu cartão de crédito ou informações de cartão de crédito, as compras do ladrão geralmente virão de uma distribuição de probabilidade diferente da sua própria. A administradora do cartão de crédito pode evitar fraudes, retendo uma conta assim que o cartão for usado para uma compra incomum (CHANDOLA; BANERJEE; KUMAR, 2009).

- **Síntese e amostragem**: neste tipo de tarefa, o algoritmo de aprendizado de máquina é solicitado a gerar novos exemplos semelhantes aos dos dados de treinamento. Sín-

tese e amostragem via aprendizado de máquina podem ser úteis para aplicativos de mídia em que pode ser caro ou complicados para a geração em grandes volumes. Por exemplo, videogames podem gerar texturas automaticamente para grandes objetos ou paisagens, em vez de exigir que alguém rotule manualmente cada pixel (CHANDOLA; BANERJEE; KUMAR, 2009). Em alguns casos, é requerido que o procedimento de amostragem ou síntese gere algum tipo específico de saída dada a entrada. Por exemplo, em uma tarefa de síntese de fala, é fornecido uma frase escrita e então é solicitado ao programa para emitir uma forma de onda de áudio contendo uma versão falada dessa frase. Este é um tipo de tarefa de saída estruturada, mas com a qualificação adicional de que não existe uma única saída correta para cada entrada e é necessário explicitamente uma grande variação na saída, para que a saída pareça mais natural e realista.

- **Imputação de valores ausentes**: neste tipo de tarefa, o algoritmo de aprendizado de máquina recebe um novo exemplo $x \in \mathbb{R}^n$, mas com algumas entradas x_i de x ausentes. O algoritmo deve fornecer uma previsão dos valores das entradas ausentes.

- **Remoção de ruídos (*Denoising*)**: Neste tipo de tarefa, o algoritmo de aprendizado de máquina recebe na entrada um exemplo corrompido $\tilde{x} \in \mathbb{R}^n$ obtido por um processo de corrupção desconhecido de um exemplo limpo $x \in \mathbb{R}^n$. O preditor ou algoritmo de aprendizado de máquina deve predizer o exemplo limpo x a partir de sua versão corrompida $\tilde{x}$, ou mais comumente predizer a distribuição de probabilidade condicional $p(x \mid \tilde{x})$.

- **Estimativa de densidade ou estimativa de função de massa de probabilidade**: No problema de estimativa de densidade, o algoritmo de aprendizado de máquina é solicitado a aprender uma função $p_{modelo} : \mathbb{R}^n \rightarrow \mathbb{R}$, onde $p_{modelo}(x)$ pode ser interpretado como uma função de densidade de probabilidade (se x for contínuo) ou uma função de massa de probabilidade (se x for discreto) no espaço de onde os exemplos foram retirados. Para executar bem essa tarefa, o algoritmo precisa aprender a estrutura dos dados que recebeu. Deve saber onde os exemplos se agrupam fortemente e onde é improvável que ocorram. A maioria das tarefas requerem que o algoritmo de aprendizagem capture, pelo menos implicitamente, a estrutura da distribuição de probabilidade. A estimativa da densidade nos permite capturar explicitamente essa distribuição. Em princípio, pode-se realizar cálculos nessa distribuição para resolver as outras tarefas também. Por exemplo, se tivesse que realizar uma estimativa de densidade para obter uma distribuição de probabilidade $p(x)$, pode-se usar essa distribuição para resolver a tarefa de imputação de valor ausente. Se um valor x_i está faltando e todos os outros valores, denotados x_{-i}, são dados, então se sabe que a distribuição sobre este é dada por $p(x_i \mid x_{-i})$. Na prática, a estimativa de densidade nem sempre

permite resolver todas essas tarefas relacionadas, porque em muitos casos as operações necessárias em $p(x)$ são intratáveis computacionalmente.

3.5 Conceitos de Aprendizagem

Segundo Géron (2019) existem inúmeros sistemas de aprendizado de maquinas, os quais podem ser classificados em categorias amplas baseadas em:

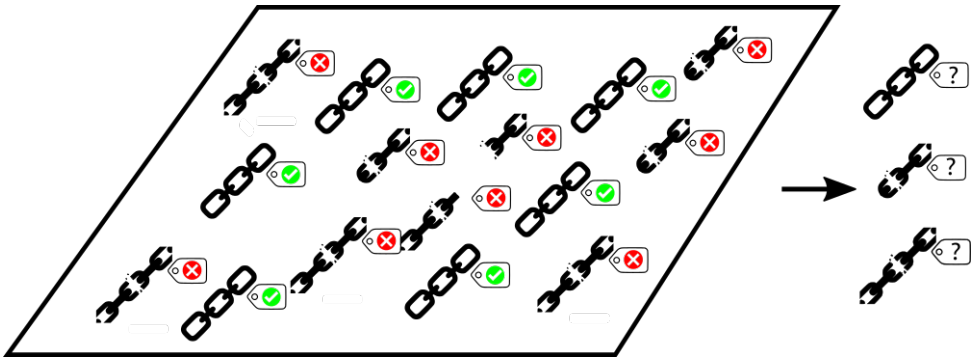
- Se os modelos são ou não treinados com supervisão humana (supervisionado, não supervisionado, semi-supervisionado e aprendizado por reforço);
- Se os modelos podem ou não aprender de forma incremental em tempo real (aprendizagem online *versus* aprendizagem em lote (*batch*));
- Os modelos que trabalham comparando novos pontos de dados com pontos de dados conhecidos ou, em vez disso, detectam padrões nos dados de treinamento e constroem modelos preditivos (aprendizagem baseado em instância *versus* aprendizagem baseada em modelo).

Esses critérios não são exclusivos, ou seja, os aprendizados podem ser combinados de modo a produzir os resultados desejados.

3.5.1 Aprendizado Supervisionado

Na aprendizagem supervisionada, os dados de treinamento são rotulados, ou seja, suas soluções são fornecidas ao algoritmo. Estas soluções/rótulos também são chamadas de *target*. A Figura 3.2 apresenta um exemplo de aprendizado supervisionado para tarefa de classificação de danos.

Figura 3.2 – Classificação de estruturas com danos ou sem danos



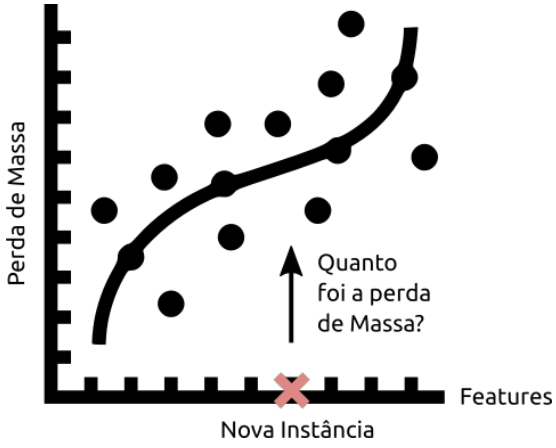
Fonte – o autor.

Um exemplos de problema que utiliza o aprendizado supervisionado é o filtro de spam: que consiste em treinar o modelo com muitos *e-mails* de exemplo junto com sua

classe (spam ou não spam), e aprender a classificar novos *e-mails*, uma típica tarefa de classificação.

Outra tarefa típica é prever um valor numérico alvo (tarefa de regressão), como o preço de um carro, dado um conjunto de características (quilometragem, idade, marca, entre outros) chamado de preditores. A Figura 3.3 apresenta um exemplo de tarefa de regressão voltado a identificar a perda de massa dado uma característica (*feature*) do sistema em análise.

Figura 3.3 – Regressão para avaliar a perda de massa dado alguma característica (*feature*)



Fonte – o autor.

Aqui estão alguns dos algoritmos/modelos de aprendizagem supervisionada mais importantes:

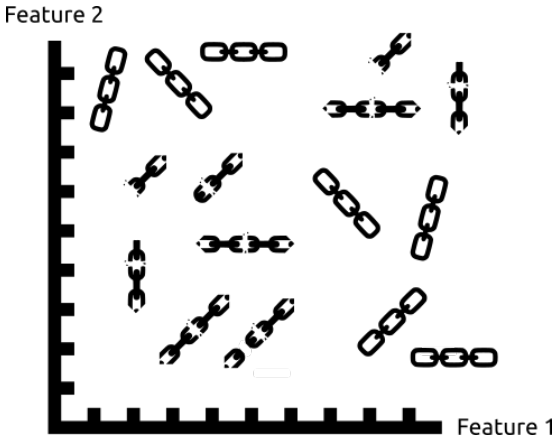
- *k-Nearest Neighbors*;
- Regressão Linear;
- Regressão Logística;
- Maquina de Vetores de Suporte (*Support Vector Machines (SVMs)*);
- Árvores de Decisão (*Decision Trees*);
- Florestas Aleatórias (*Random Forests*);
- Redes Neurais.

3.5.2 Aprendizado não Supervisionado

No aprendizado não supervisionado, os dados de treinamento não são rotulados. O sistema tenta aprender sem uma solução prévia. A Figura 3.4 apresenta um exemplo de

aplicação em uma tarefa de agrupamento (*clustering*) para um problema de identificação de danos em uma estrutura.

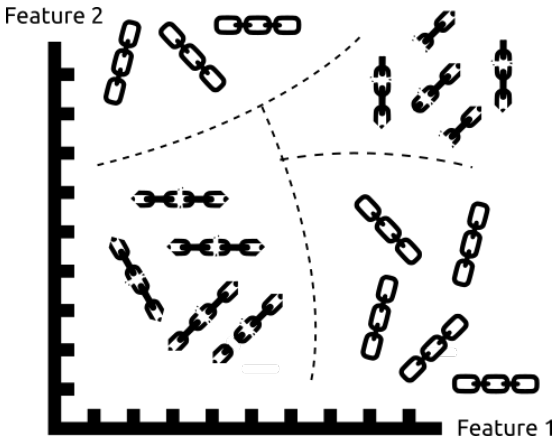
Figura 3.4 – Categorias de estruturas com danos e sem danos dado 2 características



Fonte – o autor.

Como a intenção do problema apresentado pela Figura 3.4 é o agrupamento de estruturas com características semelhantes, de modo a identificar dano, os métodos não supervisionados aprendem com as características presentes nas informações passadas para os métodos, sem a necessidade de indicar a qual classe dada instância pertence. Por exemplo, as *features 1 e 2* podem ser massa e rigidez, onde alguma alteração nessas propriedades podem representar a existência de um dano.

Figura 3.5 – *Clusters* de estruturas com e sem dano relativo a 2 característica



Fonte – o autor.

As tarefas de aprendizado não supervisionado não se restringem a clusterização, tem-se os algoritmos de visualização de dados utilizados para extrair ou identificar padrões complexos. A redução de dimensionalidade é outra tarefa não supervisionada utilizada para reduzir a quantidade de dimensões (colunas) dos dados, minimizando a perda de informações. Já para controle e prevenção/detecção de erros tem-se as tarefas de detecção de

anomalias ou detecção de novidades, possibilitando a identificação automática de valores discrepantes, evitando possíveis resultados inconsistentes apresentados por algum modelo. Enquanto que, para buscar profundamente nos dados de modo a revelar relações entre eles, tem-se o aprendizado não supervisionado por regra de associação.

Os algoritmos de visualização geram uma representação 2D ou 3D dos dados que podem ser facilmente visualizados. Esses algoritmos tentam preservar o máximo de estrutura possível (por exemplo, tentando evitar que *clusters* separados no espaço de entrada se sobreponham na visualização), para que seja possível o entendimento de como os dados são organizados e talvez identificar padrões de difícil visualização.

A tarefa de redução da dimensionalidade tem como objetivo simplificar os dados sem perder muitas informações. Uma maneira de fazer isso é mesclar vários recursos correlacionados em um. Por exemplo, a quilometragem de um carro pode estar muito correlacionada com sua idade, então o algoritmo de redução de dimensionalidade irá mesclá-los em um recurso que representa o desgaste do carro. Isso é chamado de extração de recursos.

A tarefa não supervisionada de detecção de anomalias pode ser aplicada para detectar transações incomuns de cartão de crédito para evitar fraude, detectar defeitos de fabricação ou remover automaticamente valores discrepantes de um conjunto de dados antes de alimentá-lo para outro algoritmo de aprendizagem. Ao sistema é mostrado principalmente instâncias normais durante o treinamento, de forma que o mesmo aprenda a reconhecê-las e quando analisar uma nova instância, puder dizer se essa parece normal ou se é provavelmente uma anomalia. Uma tarefa muito semelhante é a detecção de novidades: a diferença é que os algoritmos de detecção de novidades esperam ver apenas dados normais durante o treinamento, enquanto os algoritmos de detecção de anomalias são geralmente mais tolerantes. Os algoritmos de detecção de novidade tem um bom desempenho, mesmo com uma pequena porcentagem de valores discrepantes no conjunto de treinamento.

Finalmente, outra tarefa comum não supervisionada é o aprendizado por regras de associação, em que o objetivo é buscar profundamente em grandes quantidades de dados e descobrir relações interessantes entre atributos. Por exemplo, suponha que um supermercado deseja executar uma regra de associação em seus registros de vendas de modo a revelar que as pessoas que compram molho de churrasco e batatas fritas também tendem a comprar bife. Portanto, pode-se querer colocar esses itens próximos uns dos outros.

Alguns dos algoritmos mais utilizados em tarefas de aprendizado não supervisionado são:

- *Clustering* ou Agrupamento:
  - K-Means;
  - DBSCAN;

  - Hierarchical Cluster Analysis (HCA);

- Detecção de anomalias e detecção de novidades:
  - One-class SVM
  - Isolation Forest

- Visualização e redução de dimensionalidade:
  - Principal Component Analysis (PCA);
  - Kernel PCA;
  - Locally-Linear Embedding (LLE);
  - t-distributed Stochastic Neighbor Embedding (t-SNE);

- Aprendizagem por regras de associação:
  - Apriori;
  - Eclat

3.5.3 Aprendizado Semi-supervisionado

A aprendizagem semi-supervisionada é uma situação em que, os dados de treinamento de algumas das amostras não estão rotulados. Os estimadores semi-supervisionados são capazes de fazer uso desses dados não rotulados adicionais para capturar melhor a forma da distribuição dos dados subjacentes e generalizar melhor para novas amostras. Esses algoritmos podem funcionar bem quando tem-se uma quantidade muito pequena de pontos rotulados e uma grande quantidade de pontos não rotulados (PEDREGOSA *et al.*, 2011; GÉRON, 2019).

Alguns serviços de hospedagem de fotos, como o Google Fotos, são bons exemplos para aplicação do aprendizado semi-supervisionado. Depois de fazer upload das fotos para o serviço, o sistema reconhece automaticamente que a mesma pessoa aparece nas fotos 1, 5 e 11, enquanto outra pessoa B aparece nas fotos 2, 5 e 7, por exemplo. Esta é a parte não supervisionada do algoritmo (*clustering*). Agora, tudo o que o sistema precisa é que diga quem são essas pessoas. Apenas um rótulo por pessoa, é o suficiente para nomear todos em todas as fotos, o que é útil para pesquisar fotos.

A maioria dos algoritmos de aprendizado semi-supervisionado são combinações de algoritmos supervisionados e não supervisionados. Por exemplo, Redes de Crença Profunda (DBNs) são baseadas em componentes não supervisionados chamados Máquinas de Boltzmann Restritas (RBMs) empilhadas umas sobre as outras. Os RBMs são treinados sequencialmente de maneira não supervisionada e, em seguida, todo o sistema é ajustado usando técnicas de aprendizado supervisionado (PEDREGOSA *et al.*, 2011; GÉRON, 2019).

3.5.4 Aprendizado por Reforço

Aprendizagem por Reforço é um sistema de aprendizagem que possui estruturas conhecidas como agentes, responsáveis por observar o ambiente, selecionar e executar ações e obter recompensas em troca (ou penalidades na forma de recompensas negativas). O sistema deve então aprender por si mesmo qual é a melhor estratégia, chamada de política, para obter a maior recompensa ao longo do tempo. Uma política define qual ação o agente deve escolher quando estiver em uma determinada situação (GÉRON, 2019)

Por exemplo, muitos robôs implementam algoritmos de Aprendizado por Reforço para aprender a andar, jogar, negociar ações no mercado financeiro, entre outras aplicações.

Q-learning e SARSA (*State Action Reward State Action*) são dois algoritmos de aprendizado por reforço comumente usados. Enquanto o *Q-learning* é um método fora da política em que o agente aprende o valor com base na ação a' (derivada de outra política), SARSA é um método dentro da política onde o mesmo aprende o valor com base em sua ação atual derivada de sua política atual. Esses dois métodos são simples de implementar, mas carecem de generalidade, pois não têm a capacidade de estimar valores para estados invisíveis. Isso pode ser superado por algoritmos mais avançados, como *Deep Q-Networks*, que usam redes neurais para estimar os valores de Q. Mas os DQNs só podem lidar com espaços de ação discretos e de baixa dimensão. DDPG (*Deep Deterministic Policy Gradient*) é um algoritmo ator-crítico livre de modelo, fora da política, que aborda esse problema por meio de políticas de aprendizagem em espaços de ação contínua e de alta dimensão (NANDY; BISWAS, 2017; LONZA, 2019).

3.5.5 Aprendizagem em Lote (*Batch*)

Outro critério usado para classificar sistemas de aprendizado de máquina é se o sistema pode ou não aprender incrementalmente a partir de um fluxo de dados de entrada.

Os sistemas que não possuem a capacidade de aprender de forma incremental são denominados sistemas de aprendizagem em lote, ou seja, no treinamento todos os dados disponíveis são utilizados. Geralmente esse método de aprendizagem requer muito tempo e recursos computacionais, portanto, todo o processo de treinamento é normalmente feito *offline*. Após o treinamento, o modelo é lançado em produção e funciona sem aprender mais; apenas aplica o que aprendeu. Isso é chamado de aprendizagem *offline*.

Quando são disponibilizados mais dados, o modelo deve ser treinado novamente com todos os dados disponíveis (não apenas os novos dados, mas também os antigos), para então substituir o sistema antigo pelo novo.

Felizmente, todo o processo de treinamento, avaliação e lançamento de um sistema de aprendizado de máquina pode ser automatizado com bastante facilidade, de modo que mesmo um sistema de aprendizado em lote pode se adaptar às mudanças. Basta atualizar

os dados e treinar uma nova versão do sistema a partir do zero com a frequência necessária (GÉRON, 2019).

A solução de treinar todos os dados com a chegada de novos dados geralmente funciona bem, mas o treinamento usando o conjunto completo de dados pode levar muitas horas. Logo, um novo sistema é normalmente treinado apenas a cada 24 horas ou até mesmo semanalmente. Existem problemas acerca da utilização dessa metodologia de retreinamento do sistema quando há a necessidade de se adaptar aos dados que mudam rapidamente (por exemplo, para prever os preços das ações). Neste caso é necessária uma solução mais reativa (por exemplo, aprendizagem online).

Existem alguns empecilhos, caso o sistema necessite aprender de forma autônoma e possui recursos limitados (por exemplo, um aplicativo de smartphone ou um rover em Marte). Neste caso, carregar grandes quantidades de dados de treinamento e ocupar muitos recursos para treinar por horas a cada dia é impraticável. Felizmente, a melhor opção em todos esses casos é usar algoritmos capazes de aprender de forma incremental.

3.5.6 Aprendizagem Online

No aprendizado online, o treinamento do sistema é feito de forma incremental, fornecendo instâncias de dados de forma sequencial, individualmente ou por pequenos grupos chamados mini lotes (*mini-batches*). Cada etapa de aprendizado é rápida e requer pouco dos recursos computacionais, então o sistema pode aprender com novos dados rapidamente, assim que eles chegam.

É recomendado a utilização do aprendizado online para sistemas que recebam dados de forma contínua, (por exemplo, preço de ações, jogos online) e precisam se adaptar rapidamente de forma autônoma.

Outro ponto a se considerar é que essa metodologia de aprendizado pode ser utilizado caso os recursos computacionais sejam limitados. Com isso também não é necessário armazenar os dados utilizados no treinamento pelo sistema, então pode-se descartá-los (a menos que queira reverter para um estado anterior). Isso pode economizar muito espaço.

Também pode-se empregar a metodologia afim de treinar sistemas com grandes quantidades de dados, que por sua vez, não cabem na memória principal de uma máquina (isso é chamado de aprendizagem fora do núcleo, (*out-of-core learning*). O algoritmo carrega parte dos dados, executa uma etapa de treinamento nesses dados e repete o processo até que seja executado em todos os dados (GÉRON, 2019).

Neste tipo de aprendizado é importante considerar a rapidez com que os sistemas devem se adaptar aos dados variáveis: isso é chamado de taxa de aprendizagem. Com uma alta taxa de aprendizado, o sistema se adaptará rapidamente aos novos dados, mas também tenderá a esquecer rapidamente os dados antigos. Por outro lado, definir uma taxa de

aprendizado baixa fará o sistema ter mais inércia, ou seja, aprenderá mais lentamente, mas também será menos sensível a ruídos nos novos dados ou a sequências de pontos de dados não representativos (*outliers*).

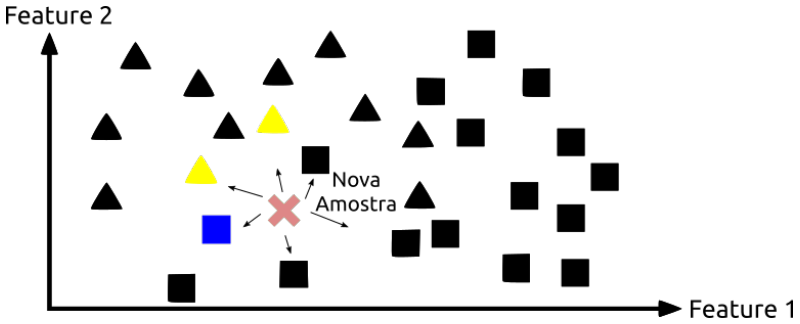
Um cuidado a ser tomado no emprego do aprendizado online é a qualidade dos dados fornecidos aos sistemas, pois se dados ruins forem fornecidos ao sistema, o desempenho do sistema diminuirá gradualmente. Esses dados de baixa qualidade podem advir de diversas fontes, deve-se monitorar o sistema de perto e desligar o aprendizado imediatamente (e possivelmente reverter para um estado de funcionamento anterior) se detectar uma queda no desempenho. Também pode-se monitorar os dados de entrada e reagir a dados anormais (por exemplo, usando um algoritmo de detecção de anomalias) (GÉRON, 2019).

3.5.7 Aprendizagem Baseada em Instância x Aprendizagem Baseada em Modelo

Outra maneira de categorizar os sistemas de aprendizado de máquina é a forma como eles generalizam, ou seja, capacidade de generalizar seus resultados para exemplos que nunca foram treinados. Ter uma boa medida de desempenho nos dados de treinamento é bom, mas insuficiente; o verdadeiro objetivo é ter um bom desempenho em novas instâncias. Existem duas abordagens principais para a generalização: aprendizagem baseada em instâncias e aprendizagem baseada em modelos.

O aprendizado baseado em instâncias se baseia em medidas de similaridade, ou seja, o quão próximo uma instância nova estaria daquelas treinadas pelo sistema. O sistema aprende os exemplos de cor e, em seguida, generaliza para novos casos comparando-os com os exemplos aprendidos (ou um subconjunto deles), usando uma medida de similaridade. A Figura 3.6 apresenta uma classificação de quadrados e triângulos baseado em instância.

Figura 3.6 – Classificação baseada em instância de quadrados e triângulos relativo a 2 características (*feature*)

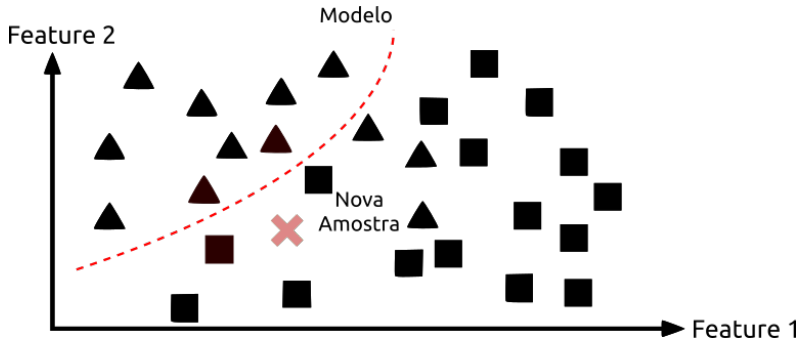


Fonte – o autor.

Outra maneira de generalizar é, a partir de um conjunto de exemplos, construir um modelo e, em seguida, usa-lo para fazer previsões. Isso é chamado de aprendizagem baseada

em modelo. A Figura 3.7 apresenta uma classificação de quadrados e triângulos baseado em modelo.

Figura 3.7 – Classificação baseado em modelo de quadrados e triângulos relativo a 2 características



Fonte – o autor.

3.5.8 Avaliando a Performance de Modelos

Para avaliar a performance de um algoritmo de aprendizado de máquina, deve-se utilizar uma medida quantitativa de seu desempenho. Normalmente, essa medida de desempenho P é específica para a tarefa T que está sendo executada pelo sistema.

Para tarefas de classificação geralmente é medida a precisão do modelo. A precisão é a proporção de exemplos para os quais o modelo produz a saída correta. Outra medida utilizada é a taxa de erro, que é a proporção de exemplos para os quais o modelo produz uma saída incorreta. Normalmente a taxa de erro varia entre 0 e 1, onde é 0 se estiver classificado corretamente e 1 caso contrário.

Para tarefas de regressão, deve-se usar métricas de desempenho que forneçam ao modelo uma pontuação de valor contínuo para cada exemplo. A abordagem mais comum é relatar a probabilidade média logarítmica que o modelo atribui a alguns exemplos (BENGIO; GOODFELLOW; COURVILLE, 2017).

A medida de desempenho é sempre avaliada sobre o conjunto de dados que o sistema nunca viu. Dessa forma, é possível avaliar a capacidade de generalização do modelo, o qual determina o quão bem o mesmo funcionará quando implantado no mundo real. Portanto, a avaliação dessas medidas de desempenho são aplicadas usando um conjunto de dados de teste separado dos dados usados para treinar o sistema de aprendizado de máquina.

A escolha da medida de desempenho pode parecer direta e objetiva, mas geralmente é difícil escolher uma medida de desempenho que corresponda bem ao comportamento desejado do sistema (BENGIO; GOODFELLOW; COURVILLE, 2017). Isso se dá porque é difícil decidir o que deve ser medido ou o que deve ser considerando pela métrica e, nos casos em que se sabe o que medir, pode ser impraticável medi-los.

De forma geral as funções de perda podem ser divididas em dois grupos, dependendo

da tarefa de aprendizado associado: perdas de classificação e perdas de regressão. Na classificação deseja-se associar a resposta a uma das classes conhecidas, enquanto a regressão associa o resultado da resposta a uma função contínua. Inicialmente, serão apresentadas as funções de perda mais comuns para algoritmos de regressão, que são: RMSE, RMSLE, MAPE, MSE, MAE e MBE.

A equação (3.1) apresenta o Erro Médio Quadrático, também chamado MSE (*Mean Square Error*) ou Perda L2.

$$MSE = \frac{\sum_{i=1}^{n}(y_i - \widehat{y}_i)^2}{n} \tag{3.1}$$

onde n é o número de casos a serem avaliados, i é o i-ésimo elemento avaliado, y_i e $\widehat{y}_i$ são, respectivamente, os valores reais e preditos pelo modelo. Como este erro considera a média do quadrado entre predições e valores reais, é focada apenas na magnitude do erro, sem se preocupar com a direção que ocorre. Entretanto, como o calculo é elevado ao quadrado, predições com muito desvio são penalizadas em relação as que possuem pouco desvio. Matematicamente o MSE é utilizado devido ao cálculo fácil de seus gradientes, uma vez que alguns algoritmos de treinamento necessitam dessa informação.

A equação (3.2) apresenta o Erro Absoluto Médio, também chamado MAE (*Mean Absolute Error*) ou Perda L1.

$$MAE = \frac{\sum_{i=1}^{n}|y_i - \widehat{y}_i|}{n} \tag{3.2}$$

Este tipo de erro também não considera a direção do erro por conta do módulo, mas é mais robusto para manipulação com valores discrepantes (*outliers*) uma vez que a diferença não é elevada ao quadrado. No entanto, este erro é mais complexo de se obter suas derivadas que podem ser obtidas através de técnicas de análise numérica.

A equação (3.3) apresenta o Erro Enviesado Médio, também chamado MBE (*Mean Bias Error*).

$$MBE = \frac{\sum_{i=1}^{n}(y_i - \widehat{y}_i)}{n} \tag{3.3}$$

Esse erro apresentado pela equação 3.3 é mais dificilmente aplicado porque não possui módulo ou quadrado e pode facilmente anular os erros positivos com os negativos. Entretanto, em condições específicas pode indicar o viés positivo ou negativo dos resultados que os anteriores não apresentam. Em algumas situações o MBE é utilizado em conjunto com algum dos anteriores.

A Raiz Quadrada Média do Erro, ou RMSE, mede a precisão geral do modelo e é uma base para compará-lo com outros modelos, incluindo modelos ajustados usando técnicas de aprendizagem de máquina (BRUCE; BRUCE; GEDECK, 2020). A equação 3.4 apresenta a formulação para a métrica RMSE.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \widehat{y}_i)^2}{n}} \tag{3.4}$$

Semelhante ao RMSE, tem-se o erro padrão residual, ou RSE. Neste caso, é utilizada a quantidade de variáveis que o modelo utiliza. A equação 3.5 apresenta sua formulação.

$$RSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \widehat{y}_i)^2}{(n-P-1)}} \tag{3.5}$$

A única diferença entre as duas métricas, é que o denominador são os graus de liberdade $n-P-1$, onde P é a quantidade de variáveis, em oposição ao número de casos n. Na prática, a diferença entre RMSE e RSE é muito pequena, especialmente para aplicações que utilizam grande volume de dados.

Vale considerar que é comum utilizar a métrica RMSE quando não se considera nenhum aspecto em relação às diferenças de magnitudes entre o que foi previsto e o que era esperado na base de avaliação. Porém quando não se quer penalizar erros que ocorram em magnitudes distintas é então utilizado o RMSLE. A equação 3.6 apresenta sua formulação.

$$RMSLE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(log(y_i+1) - log(\widehat{y}_i+1))^2} \tag{3.6}$$

O termo logaritmo do RMSLE realiza o ajuste entre $\widehat{y}_i$ e y_i e calcula a diferença dentro da mesma magnitude antes do cálculo da média quadrática. Isso significa que, mesmo com uma ordem de magnitude muito maior do que nos erros anteriores, o logaritmo fará a suavização desses erros, retirando a magnitude na média quadrática.

O erro percentual médio absoluto MAPE, também conhecido como desvio percentual absoluto médio (MAPD), é uma métrica de avaliação para problemas de regressão. A ideia dessa métrica é ser sensível a erros relativos. Por exemplo, não é alterado por uma escala global da variável de destino. A equação 3.7 apresenta a formulação para a métrica MAPE.

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{\left|y_i - \widehat{y}_i\right|}{max(\epsilon, \left|y_i\right|)} \tag{3.7}$$

ϵ é um número arbitrário pequeno, mas estritamente positivo, para evitar resultados indefinidos quando y_i é zero. O MAPE também é algumas vezes relatado como uma porcentagem, que é a equação acima multiplicada por 100. A diferença entre y_i e $\widehat{y}_i$ é dividida pelo valor real y_i novamente. O valor absoluto neste cálculo é somado para cada ponto previsto e dividido pelo número de pontos ajustados n.

O coeficiente de determinação, também chamado de estatística R ao quadrado ou R^2, pode variar de 0 a 1, medindo a proporção de variação nos dados que são contabilizados no modelo. É útil principalmente em usos explicativos de regressão, onde se deseja avaliar o quão bem o modelo se ajusta aos dados. A equação 3.8 apresenta a formulação para R^2.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \widehat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \overline{y})^2} \tag{3.8}$$

Partindo de que n é o número de casos a serem avaliados, i é o i-ésimo elemento avaliado, y_i e $\widehat{y}_i$ são, respectivamente, os valores reais e preditos pelo modelo. Como este erro considera a média do quadrado entre predições e valores reais, o denominador composto por y_i e $\overline{y}$ se refere a soma total dos quadrados, ou seja, a soma dos quadrados das diferenças entre a média e cada valor observado. Já o numerador se refere a soma dos quadrados dos resíduos, que calcula a parte que não é explicada pelo modelo (BRUCE; BRUCE; GEDECK, 2020; AKOSSOU; PALM, 2013).

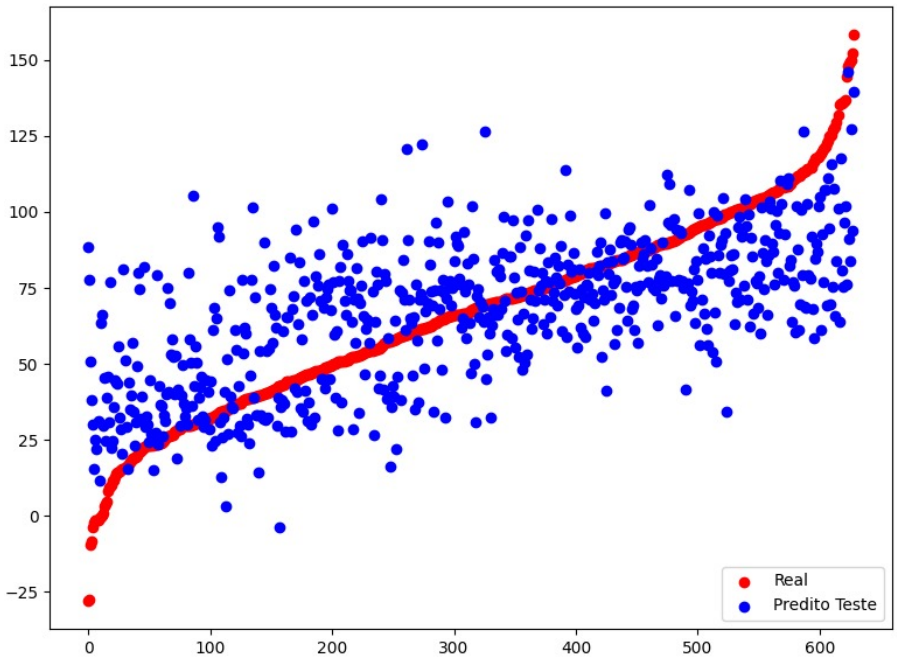
Adicionar mais variáveis, no entanto, não significa necessariamente um modelo melhor. Os estatísticos usam o princípio da navalha de Occam para orientar a escolha de um modelo: todas as coisas sendo iguais, um modelo mais simples deve ser usado em preferência a um modelo mais complicado. Incluir variáveis adicionais sempre reduz o RMSE e aumenta o R2 para os dados de treinamento. Portanto, eles não são apropriados para ajudar a orientar a escolha do modelo. Uma abordagem para incluir a complexidade do modelo é usar o R^2 ajustado. A equação 3.9 apresenta a formulação para o R^2 ajustado (BRUCE; BRUCE; GEDECK, 2020).

$$R^2_{ajustado} = 1 - (1 - R^2)\frac{n-1}{n-P-1}, \tag{3.9}$$

onde n é o número de casos a serem avaliados e P é o numero de variáveis do modelo.

Outras ferramentas importantes para a avaliação das tarefa de regressão são as baseadas em visualização. Consiste em plotar as curvas do modelo em análise de modo a ter uma comparação qualitativa da performance do modelo. O procedimento para realização dessa análise consiste em ordenar os valores da parte real e então plotar as respectivas previsões feita pelo modelos. A Figura 3.8 apresenta um exemplo dessa análise.

Figura 3.8 – Análise qualitativa da performance do modelo comparando as estimativas com o valor real

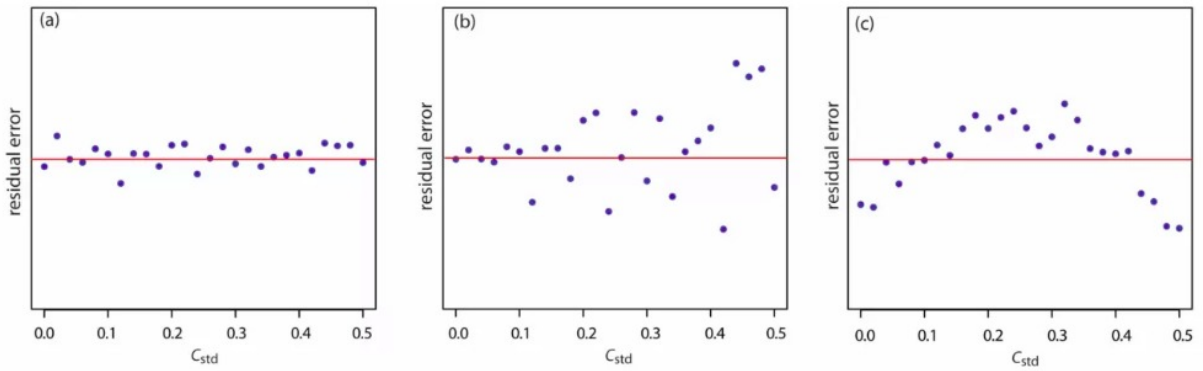


Fonte – o autor.

Como pode ser observado pela Figura 3.8 o modelo teve uma performance relativamente boa para o problema de regressão, apesar das previsões estarem com viés mais positivo, indicando uma possível falta de dados.

Outra ferramenta de análise muito importante é o gráfico de dispersão de erros. Este tem o papel de avaliar a variabilidade do modelo. Para isso, após a construção do modelo de regressão, deve-se fazer um gráfico de dispersão dos resíduos (valor ajustado menos valor real) e verificar se eles se concentram em torno do zero e tem uma variabilidade constante. A Figura 3.9 apresenta 3 exemplos de gráficos de dispersão apresentando as possíveis distribuições.

Figura 3.9 – Tipos de distribuições de erro para problemas de regressão



Fonte – o autor.

Na Figura 3.9 tem-se três gráficos de dispersão de resíduos para modelos diferentes. Em (a), os resíduos se concentram em torno do zero e possuem uma variabilidade aparentemente constante. Em (b), os dados também se concentram em torno de zero, mas a variabilidade aumenta no lado direito do gráfico. Por fim, em (c) tem-se dados não concentrados em torno de zero, mas variabilidade constante. Resumindo, apenas em (a) são satisfeitos os pressupostos do modelo de variância constante e independência.

Para os algoritmos de classificação, em que há um conjunto definido de respostas e a predição classifica em alguns dos casos conhecidos, são aplicadas as funções Perda de Dobradiça e Perda de Entropia Cruzada.

A perda de dobradiça (*hinge*) se baseia no fato de que a nota da classificação correta de uma resposta deve ser maior que a soma das notas das classificações erradas geralmente pelo valor de uma unidade. Assim, a perda de dobradiça é usada para a classificação de margem máxima, muito usado em Máquinas de Vetores de Suporte (SVM). Embora não seja uma função diferenciável, a mesma trabalha bem com os usuais otimizadores convexos contidos em bibliotecas de aprendizado de máquina. A equação (3.10) apresenta sua descrição.

$$SVMLoss = \sum_{j \neq y_i} max(0, s_j - s_{y_i} + 1) \tag{3.10}$$

Como pode ser observado pela expressão, um valor *SVMLoss* próximo de zero indica uma predição correta da classe enquanto valores muito elevados mostram um erro maior.

A equação (3.11) apresenta a Perda de Entropia Cruzada, ou também chamada Log Loss.

$$CEL = -(y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i)) \tag{3.11}$$

De acordo com a expressão, quando o valor de y_i se aproxima de 1, o segundo termo desaparece, quando o valor de y_i se aproxima de zero, o primeiro termo da expressão se anula. Assim, a função penaliza muito predições que tenham vários elementos similares, mas que um dos valores esteja totalmente fora, ou seja, parece que acertou, mas errou na classificação.

Os classificadores podem trabalhar em tarefas que realizam a classificação de 2 rótulos (classificação binária), que realizam a classificação de mais de 2 rótulos (multi-classificação), que classificam vários rótulos para uma instância (multilabel) e, por fim, a classificação multidimensional, que é uma generalização da classificação multilabel (também chamada de multioutput), na qual existem várias variáveis respostas, cada uma delas com mais de um valor possível (GÉRON, 2019; PEDREGOSA *et al.*, 2011).

Avaliar um classificador é significativamente mais complexo do que avaliar um regressor. Algumas das métricas mais relevantes são: matriz de confusão, curva ROC, AUC, acurácia, *precision*, *recall* e F_1-*score*.

A acurácia é a métrica mais simples e muito utilizada. A mesma expressa a proporção de acerto do classificador, ou seja, é a razão entre a soma de todas as predições corretas pela quantidade total de exemplos. Apesar de ser muito comum, deve-se tomar cuidados ao utiliza-la em problemas onde há desbalanceamento entre as classes, ou seja, mais exemplos de uma classe do que de outra. Esse cuidado se dá pois a acurácia pode estar mascarada, apresentando um valor alto, quando na verdade o classificador não está ajustado.

A matriz de confusão apresenta de forma tabelada as frequências de classificação para cada classe do modelo. Cada linha na matriz de confusão representa uma classe real, enquanto cada coluna representa uma classe predita. O Quadro 3.1 apresenta a matriz de confusão para um exemplo de classificação binaria.

		Classe Predita	
		Sim	Não
Classe Real	Sim	Verdadeiro Positivo (TP)	Verdadeiro Negativo (FN)
	Não	Falso Positivo (FP)	Verdadeiro Negativo (TN)

Fonte – o autor.

Quadro 3.1 – Matriz de confusão

No Quadro 3.1 pode-se observar os valores de Verdadeiro Positivo (TP), Falso Negativo (FN), Falso Positivo (FP) e Verdadeiro Negativo (TN) que tem como significado:

- **Verdadeiro Positivo** (*true positive* — TP): ocorre quando no conjunto real, a classe positiva que se está avaliando foi prevista corretamente como positiva.
- **Falso Positivo** (*false positive* — FP): ocorre quando no conjunto real, a classe negativa que se está avaliando foi prevista incorretamente como positiva.
- **Verdadeiro Negativo** (*true negative* — TN): ocorre quando no conjunto real, a classe negativa que se está avaliando foi prevista corretamente como negativa.
- **Falso Negativo** (*false negative* — FN): ocorre quando no conjunto real, a classe positiva que se está tentando prever foi prevista incorretamente como negativa.

A matriz de confusão proporciona uma visualização global da performance do modelo. Nela, está embutido um aspecto importante a ser avaliado, a precisão das previsões

positivas, ou seja, a precisão do classificador. A equação 3.12 apresenta a equação para determinar a precisão do classificador.

$$precision = \frac{TP}{TP + FP} \tag{3.12}$$

Uma maneira trivial de ter uma precisão perfeita é fazer uma única previsão positiva e garantir que essa seja correta (precisão = 1/1 = 100%). Isso não seria muito útil, pois o classificador ignoraria todas as instâncias, exceto uma. Portanto, a precisão é normalmente usada junto com outra métrica chamada *recall*, também chamada de sensibilidade ou taxa de detecção de verdadeiros positivos (TRP). A equação 3.13 apresenta a equação para se obter a sensibilidade do modelo.

$$recall = \frac{TP}{TP + FN} \tag{3.13}$$

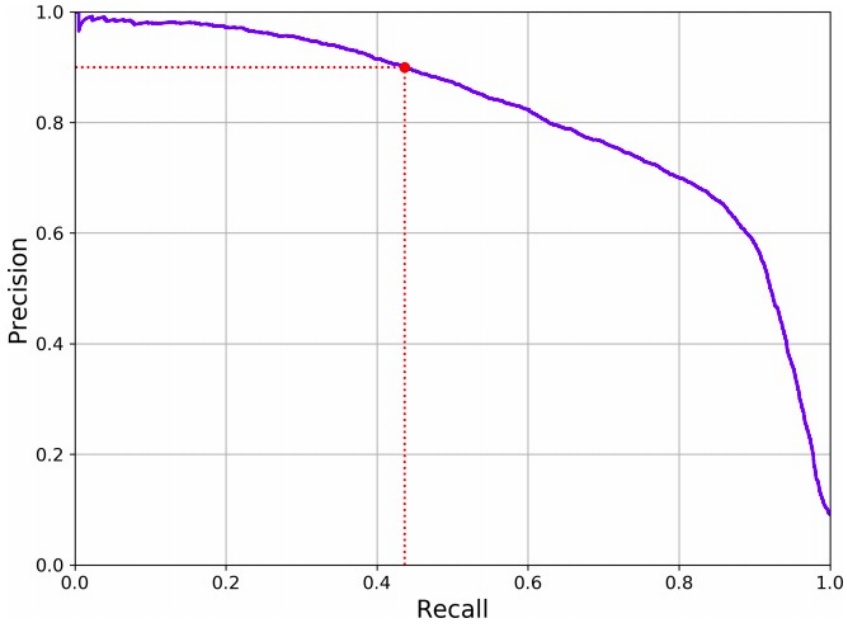
É conveniente combinar precisão e *recall* em uma única métrica chamada F_1 *-score*. A F_1 *-score* é a média harmônica da precisão e *recall*. Enquanto a média regular trata todos os valores igualmente, a média harmônica atribui muito mais peso aos valores baixos. Como resultado, o classificador só obterá uma pontuação alta de F_1 *-score* se o *recall* e a precisão forem altos. A equação 3.14 apresenta a equação para se obter a métrica F_1 *-score*.

$$F_1 = \frac{2}{\frac{1}{precision} + \frac{1}{recall}} = \frac{TP}{TP + \frac{FN+FP}{2}} \tag{3.14}$$

A F_1 *-score* favorece classificadores com precisão e *recall* similares e altos. Em alguns casos esse não é o objetivo, pois em determinados contextos a métrica a ser maximizada é a precisão. Essa abordagem é comum em problemas de recomendação onde o objetivo é indicar muitos itens certos e aceitar perder alguns possíveis itens corretos. Já em outros casos, por exemplo, problema de detecção, o foco pode ser na *recall* onde a não detecção pode acarretar em prejuízos, ou seja, é melhor receber alertas falsos do que perder a detecção.

Infelizmente, não se pode ter as duas coisas: aumentar a precisão e reduzir o *recall* ou vice-versa. Isso é chamado de *tradeoff* precisão/*recall* (GÉRON, 2019). Uma forma de se selecionar um bom *tradeoff* precisão/*recall* é através da plotagem dessas métricas. A Figura 3.10 apresenta a relação entre precisão/*recall*.

Figura 3.10 – Análise do *tradeoff* da precisão/*recall*



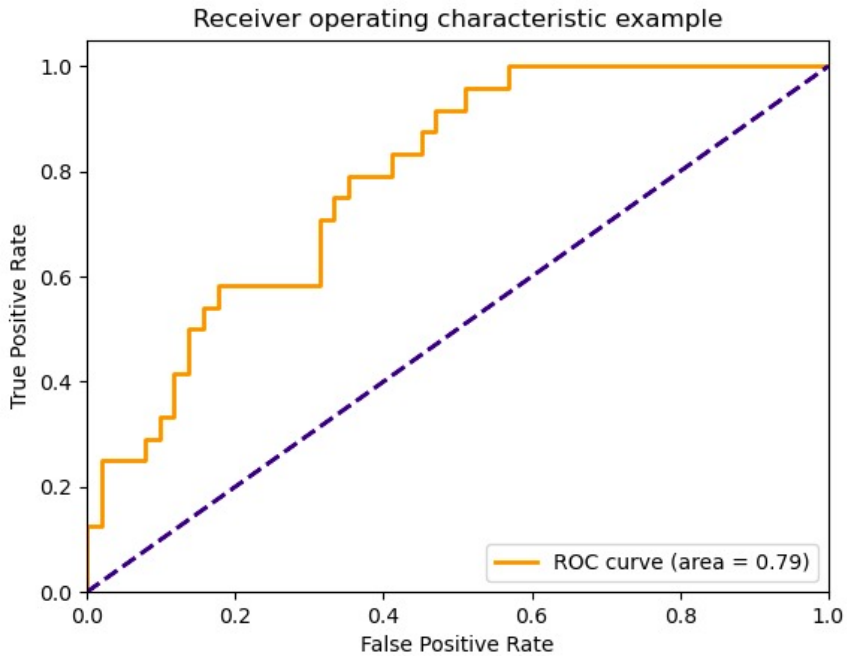
Fonte – o autor.

Na Figura 3.10 pode-se observar que a precisão começa a cair drasticamente em torno de 80% da *recall*. A depender do objetivo do projeto, provavelmente deseja-se selecionar uma compensação de precisão/*recall* pouco antes dessa queda, por exemplo, em cerca de 60% de *recall*.

A curva de característica de operação do receptor ROC (*Receiver Operating Characteristic*) é outra ferramenta comum usada com classificadores binários. É muito semelhante à curva de precisão/*recall*, mas em vez de traçar a precisão versus *recall*, a curva ROC traça a taxa de verdadeiro positivo (outro nome para *recall* - TP) contra a taxa de falso positivo (FP). Isso significa que o canto superior esquerdo do gráfico é o ponto "ideal", ou seja, taxa de falso positivo igual a zero e uma taxa de verdadeiro positivo igual a um. Isso não é muito realista, mas significa que uma área maior sob a curva AUC (*Area Under the Curve*) geralmente é melhor (GÉRON, 2019; PEDREGOSA *et al.*, 2011).

A "inclinação" das curvas ROC também é importante, pois é ideal para maximizar a taxa de verdadeiros positivos enquanto minimiza a taxa de falsos positivos. É comum plotar uma linha representando um classificador puramente aleatório afim de comparação. A Figura 3.11 apresenta um exemplo de curva ROC com uma AUC de 0.79 e uma linha representando um classificador puramente aleatório.

Figura 3.11 – Análise da curva ROC



Fonte – o autor.

Para estender a curva ROC e a área ROC para a classificação multi-rótulo, é necessário binarizar a saída. Uma curva ROC pode ser desenhada por rótulo, mas também se pode traçar uma curva ROC considerando cada elemento da matriz do indicador de rótulo como uma previsão binária (micro-média).

3.5.9 Principais Desafios no Aprendizado de Máquina

Segundo Géron (2019) existem duas coisas que podem dar errado na tarefa de selecionar um algoritmo de aprendizado de máquina e treina-lo em alguns dados, são elas "algoritmos ruins"e "dados ruins".

Para Bengio, Goodfellow e Courville (2017) o principal desafio do aprendizado de máquina é a capacidade em se ter bom desempenho em entradas novas e não vistas, não apenas naquelas em que o modelo foi treinado. A capacidade de um bom desempenho em entradas não observadas anteriormente é chamada de generalização.

Ao selecionar um algoritmo de aprendizado de máquina e treina-lo através de um conjunto de treinamento, pode-se calcular o erro nesse procedimento, chamado de erro de treinamento, tendo como objetivo minimiza-lo. Com isso é possível diminuir o erro de generalização, o qual é definido como o valor esperado do erro em uma nova entrada. Aqui, a expectativa é feita em diferentes entradas possíveis, extraídas da distribuição de entradas que se espera que o sistema encontre na prática. Normalmente o erro de generalização de um modelo de aprendizado de máquina é medido através do desempenho em um conjunto

de testes de exemplos que foram coletados separadamente do conjunto de treinamento.

Para que o modelo possa ter uma alta capacidade de generalização deve-se considerar fatores que impactam no alcance dessa capacidade. Abaixo são apresentados esses fatores:

- **Quantidade insuficiente de dados de treinamento**: Para tarefas que possuem uma quantidade insuficiente de dados pode-se destacar o trabalho de Halevy, Norvig e Pereira (2009), o qual relatam a importância de se ter mais dados do que algoritmos complexos. Outro trabalho realizado por Banko e Brill (2001) relaciona os desempenhos semelhantes de diversos algoritmos simples e complexos quando se tem uma quantidade insuficiente de dados. Para Zheng e Casari (2018), se não houver recursos informativos suficientes, o modelo será incapaz de realizar a tarefa final. Porém, se houver muitos recursos, ou se a maioria deles forem irrelevantes, o modelo será mais caro e complicado de treinar e algo pode dar errado no processo de treinamento, afetando o desempenho do modelo;

- **Dados de treinamento não representativos**: Para desenvolver modelos com alta capacidade de generalização, é crucial que os dados de treinamento representem bem os casos para os quais se deseja generalizar. Isso geralmente é mais difícil do que parece: se a amostra for muito pequena, é possível a existência de ruído de amostragem (ou seja, dados não representativos como resultado do acaso), mas mesmo amostras muito grandes podem ser não representativas se o método de amostragem for falho. Isso é chamado de viés de amostragem (*sampling bias*);

- **Dados de baixa qualidade**: É comum o empenho para melhorar a qualidade dos dados, visando remover ou tratar valores extremos denominados de *outliers*, tratar ou remover valores ausentes buscando evitar a perda de dados. Segundo Zheng e Casari (2018) os recursos que são relevantes para uma dada tarefa devem ser fáceis de serem ingeridos pelo modelo uma vez que alguns modelos são mais apropriados para alguns tipos de recursos e vice-versa.

- **Recursos (*features*) irrelevantes**: Uma parte crítica do aprendizado de máquina é apresentar um bom conjunto de recursos para treinar. Este processo, de melhoria de recursos é denominado engenharia de recursos e envolve a seleção de recursos mais úteis para treinar entre os recursos existentes: extração de recursos combinando recursos existentes para produzir um mais útil (algoritmos de redução de dimensionalidade) e criação de novos recursos por meio da coleta de novos dados (GÉRON, 2019). A engenharia de recursos é o processo de formular os recursos mais apropriados de acordo com os dados, o modelo e a tarefa (ZHENG; CASARI, 2018).

- ***Overfitting* no treinamento**: *Overfitting* é o uso de modelos ou procedimentos que violam a parcimônia, ou seja, que incluem mais termos do que o necessário ou usam

abordagens mais complicadas do que o necessário, fazendo com que os modelos fiquem superajustados aos dados de treinamento, com isso reduzindo drasticamente a capacidade de generalização (HAWKINS, 2004). Para solucionar esse problema é comum simplificar o modelo, selecionando um com menos parâmetros (por exemplo, um modelo linear em vez de um modelo polinomial de alto grau), reduzindo o número de atributos nos dados de treinamento ou restringindo o modelo; aplicar regularizadores em modelos que usam essa abordagem e reunir mais dados de treinamento para reduzir o ruído nos dados de treinamento (por exemplo, corrigir erros de dados e remover *outliers*).

- ***Underfitting* no treinamento**: O *underfitting* é o oposto de *overfitting*: ocorrendo quando o modelo é muito simples para aprender a estrutura subjacente dos dados. As principais opções para corrigir este problema são: selecionar um modelo mais poderoso, com mais parâmetros, alimentar melhores recursos para o algoritmo de aprendizagem (engenharia de recursos) e reduzir as restrições no modelo (por exemplo, reduzir o hiperparâmetro de regularização) (GÉRON, 2019).

Uma prática comum no desenvolvimento de modelos de aprendizado de máquina, visando atingir o objetivo de maximizar a capacidade de generalização, é dividir a base de dados em 3 conjuntos, treinamento, validação e teste, cada uma delas independente uma da outra. Essa separação visa encontrar os melhores hiperparâmetros do modelo em análise. O procedimento consiste em treinar o modelo com o conjunto de treinamento e varrer intervalos predefinidos para alguns parâmetros do modelo, no qual esse modelo deve ser avaliado no conjunto de validação. Isso é feito até encontrar os hiperparâmetros que possuem a melhor métrica sobre o conjunto de validação. A ultima etapa é agrupar o conjunto de treino e validação, realizar um novo treinamento para então avaliar o modelo no conjunto de teste afim de obter uma estimativa da capacidade de generalização do modelo.

Outra abordagem a ser considerada é utilizar uma validação cruzada para o conjunto de treino e validação. É indicada a utilização dessa abordagem quando se tem um conjunto de validação ou de treino pequenos.

É válido destacar que um modelo é uma versão simplificada das observações que são passadas para ele. Essas simplificações devem descartar os detalhes supérfluos que provavelmente não se generalizarão para novas instâncias. No entanto, para decidir quais dados descartar e quais dados manter, é necessário fazer suposições sobre esses.

O teorema "sem almoço grátis"(do inglês - *no free lunch theorem*) para aprendizado de máquina afirma que, calculada a média de todas as distribuições geradoras de dados possíveis, cada algoritmo de classificação tem a mesma taxa de erro ao classificar pontos anteriormente não observados. Em outras palavras, em certo sentido, nenhum algoritmo de

aprendizado de máquina é universalmente melhor do que qualquer outro (WOLPERT, 1996; BENGIO; GOODFELLOW; COURVILLE, 2017).

O algoritmo mais sofisticado conhecido tem o mesmo desempenho médio (sobre todas as tarefas possíveis) como meramente predizendo que cada ponto pertence à mesma classe. Felizmente, esses resultados são válidos apenas quando é feito a média de todas as distribuições possíveis de geração de dados.

Se forem feitas suposições sobre os tipos de distribuições de probabilidade que se encontram em aplicativos do mundo real, pode-se projetar algoritmos de aprendizagem que funcionam bem nessas distribuições. Isso significa que o objetivo da pesquisa em aprendizado de máquina não é buscar um algoritmo de aprendizado universal ou o melhor algoritmo de aprendizado absoluto. Em vez disso, o objetivo é entender quais tipos de distribuições são relevantes para o "mundo real"que um agente de AI experimenta e que tipos de algoritmos de aprendizado de máquina funcionam bem em dados extraídos dos tipos de distribuições geradoras de dados que estão sob estudo.

3.6 Ferramentas e Conceitos para Soluções de Aprendizado de Máquina

Nesta Seção será abordado inicialmente o ciclo de vida de uma aplicação de aprendizado de máquina, apresentando os principais fluxos de trabalho existentes. Em seguida, são apresentadas as ferramentas para a criação, desenvolvimento e análise de modelos (Sklearn e Pycaret). Posteriormente, é abordado a biblioteca SHAP responsável por interpretar os modelos desenvolvidos de modo a identificar possíveis viés, justificar predições e apresentar como os algoritmos abstraem os padrões nos dados. E por fim, são apresentadas ferramentas para entrega de uma solução de aprendizado de máquina, baseadas em implantação (Streamlit) e em conteinerização (Docker).

3.6.1 Ciclo de Vida de Modelos

O ciclo de desenvolvimento de modelo passa por vários estágios, desde a coleta de dados até a construção do modelo. A exploração dos dados para entender as relações (em variáveis) é sempre recomendada. O entendimento do problema em questão traz ideias ou *insights* para agregar no desenvolvimento dos modelos.

É comum seguir um fluxo de trabalho para desenvolvimento de uma aplicação de aprendizado de máquina. Um fluxo apresentado por Ashmore, Calinescu e Paterson (2019) destaca que o processo se dá em quatro fases: gerenciamento de dados, que se concentra na preparação dos dados necessários para construir um modelo de aprendizagem de máquina; aprendizagem do modelo, onde ocorre a seleção e o treinamento do modelo; verificação do

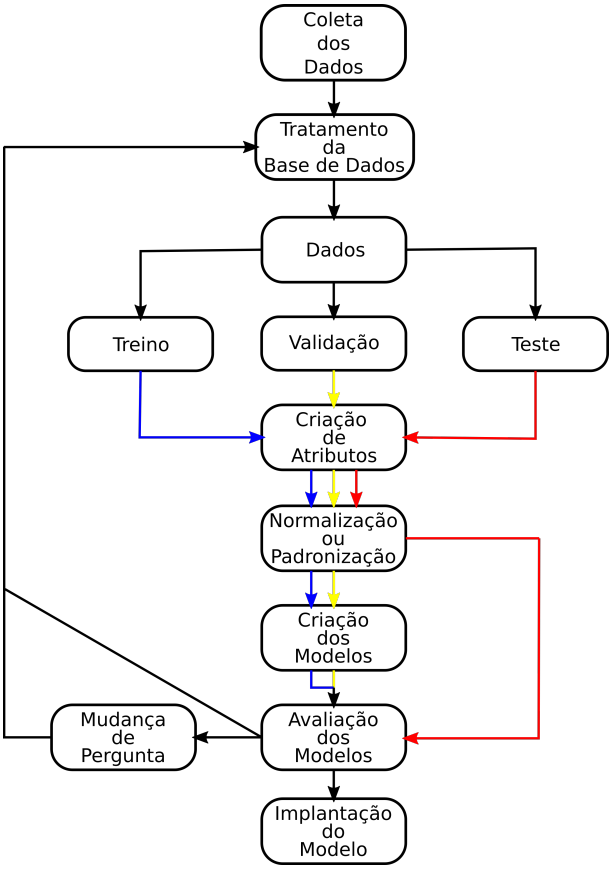
modelo, cujo objetivo principal é garantir que o modelo adira a determinados requisitos de desempenho; e implantação do modelo, que consiste na integração do modelo treinado na infraestrutura de software necessária para executá-lo.

Outro fluxo apresentado por Sato, Wider e Windheuser (2019) apresenta uma visão ampla do desenvolvimento de soluções de aprendizado de máquina, se preocupando com desenvolvimento de modelos, avaliação de modelos, teste dos modelos, implantação e monitoramento. A metologia de entrega contínua para aprendizado de máquina (CD4ML do inglês - *Continuous Delivery for Machine Learning*) possibilita a entrega de aplicativos de aprendizado de máquina baseados em código, dados e modelos em incrementos pequenos e seguros que podem ser reproduzidos e liberados de forma confiável a qualquer momento.

Outra abordagem de fluxo de trabalho apresentada por Harrison (2019) segue o Processo Padrão do Mercado para Mineração de Dados (CRISP-DM). São seguidos 5 passos para uma melhoria continua da aplicação: entendimento do negócio, entendimento dos dados, modelagem, avaliação e implantação.

Todos os fluxos apresentados possuem proximidade nas etapas de trabalho, diferenciando o foco principal para cada abordagem. O foco do processo CRISP-DM será seguido nessa contribuição. A Figura 3.12 apresenta o fluxo de trabalho a ser seguido.

Figura 3.12 – Fluxo de trabalho para criação de modelos utilizando metodologia CRISP-DM



Fonte – Adaptado de: (HARRISON, 2019)

O fluxo de trabalho apresentado pela Figura 3.12 inicia-se na coleta de dados e então segue para o seu tratamento. Essa etapa é crucial para o bom desempenho dos processos subsequentes. Nessa etapa são feitos ajustes para padronizar a estrutura e es valores dos dados. Por exemplo, conversão dos atributos para numéricos, remoção ou tratamento de valores ausentes (*outliers*), dentre outros.

Em seguida, é então feita e separação da base de dados em 3 conjuntos: treino, validação e teste. O conjunto de treino é utilizado nos modelos afim de que os mesmos possam aprender e identificar padrões, capacitando-os a generalizar para outro conjunto de valores. O conjunto de validação é utilizado na avaliação dos modelos e busca ou *tuning* dos hiperparâmetros. E o conjunto de teste simula a existência de novos dados nunca vistos pelo modelo, também utilizado para estimar a capacidade de generalização do modelo.

Após a separação, é então aplicada a engenharia de características (*feature engineering*) para criação de novos atributos, buscando obter bons desempenhos nos modelos avaliados.

Então é realizado o processamento dos valores de modo a normaliza-los ou padroniza-los. Vale destacar a separação do conjunto de teste do conjunto de treino/validação, esse é feito na intenção de evitar vazamento de informações para o conjunto de teste, assim enviesando as métricas de avaliação.

Para o desenvolvimento de modelos de aprendizado de maquina é comum empregar a avaliação de múltiplos modelos e então selecionar os melhores baseados nas métricas sobre o conjunto de validação.

Para finalizar o desenvolvimento do modelo, é então retreinado com os dados de treino e validação com os melhores hiperparâmetros encontrados. Então é analisada a estimativa da capacidade de generalização do modelo através das métricas específicas, sobre os dados de teste. Em seguida, o processo entra em um *loop*, onde é possível monitorar o objetivo ou pergunta que o modelo trabalha, ao mesmo tempo que o processo de treinamento/validação é repetido conforme a adição de novos dados. A cada iteração dessa, é então obtido um modelo que é utilizado na aplicação a partir de um processo de implantação.

3.6.2 Ferramentas para Criação de Modelos

3.6.2.1 Sklearn

A biblioteca Scikit-Learn (scikit-learn.org) fornece implementações sólidas de uma variedade de algoritmos de aprendizado de máquina e tem sido nos últimos anos a biblioteca mais utilizada para ciência de dados e aprendizado de máquina tanto pela comunidade acadêmica como INRIA (França) quanto empresas e indústrias como J.P.Morgan, Spotify, Evernote, Booking.com, entre outras.

A biblioteca Scikit-Learn surgiu de um projeto de verão no Google em 2007 e é baseada em outras também famosas bibliotecas para Python, como NumPy, utilizada para manipulação vetorial; Scipy, responsável por funções matemáticas, científicas e de engenharia e o Matplotlib, que é uma biblioteca de visualização e geração de gráficos. É importante relembrar que todas essas bibliotecas são *open-source* utilizáveis comercialmente (licença BSD), sendo desenvolvidas e aperfeiçoadas constantemente pela comunidade mundial da linguagem Python.

O Scikit-Learn teve seu lançamento em 2010 com o apoio do INRIA, instituto francês, e é responsável por implementar funções específicas para o uso do aprendizado de máquina como regressão, classificação, redução de dimensões e análise de conglomerado. Alguns algoritmos clássicos implementados nesta biblioteca são as Máquinas de Vetores de Suporte (SVM - do inglês *Support Vector Machine*), regressão logística, *gradient boosting*, DBSCAN, Naive Bayes, k-médias, florestas aleatórias e outras baseadas em árvores de decisão. A biblioteca também implementa redes neurais simples do tipo perceptron de camadas múltiplas.

O tratamento dos dados no Scikit-Learn se dá por meio de tabelas de dados, onde uma tabela básica é uma grade bidimensional de dados, em que as linhas representam os elementos individuais do conjunto de dados e as colunas representam as quantidades relacionadas a cada um desses elementos.

Esse layout de tabela deixa claro que as informações podem ser pensadas como uma matriz ou vetor numérico bidimensional, que são nomeadas de matriz de recursos. Por convenção, esta matriz de recursos é frequentemente armazenada em uma variável chamada X. A matriz de recursos é considerada bidimensional, com forma [n amostras, n *features*], e geralmente está contida em uma matriz NumPy ou em um Pandas DataFrame, embora alguns modelos Scikit-Learn também aceitem matrizes esparsas SciPy.

Além da matriz de recursos X, geralmente também se trabalha com um rótulo ou vetor de destino, que normalmente é chamado por convenção de y. A matriz de destino geralmente é unidimensional, com comprimento [n amostras] e comumente está contida em uma matriz NumPy ou Pandas Series. O vetor alvo pode ter valores numéricos contínuos ou classes/rótulos discretos, embora alguns estimadores Scikit-Learn lidem com vários valores de destino na forma de uma [n amostras, n *targets*] matriz de destino bidimensional.

Segundo o Buitinck *et al.* (2013) o desenvolvimento da API foi baseada no conceito de consistência, onde todos os objetos compartilham uma interface comum desenhada a partir de um conjunto limitado de métodos. A API possuí documentação consistente, inspeção onde todos os valores de parâmetro especificados são expostos como atributos públicos. Além disso, também se baseia na inspeção hierarquia limitada de objetos onde apenas algoritmos são representados por classes Python. Os conjuntos de dados são representados em formatos padrão (matrizes NumPy, Pandas DataFrames, matrizes esparsas SciPy) e nomes de parâmetros usam strings Python padrão, composição onde muitas tarefas de aprendi-

zado de máquina podem ser expressas como sequências de algoritmos mais fundamentais. O Scikit-Learn faz uso disso sempre que possível, por fim, padrões sensíveis em que os modelos requerem parâmetros especificados pelo usuário, a biblioteca define um valor padrão apropriado. Para a utilização da API é empregada 5 etapas:

- Escolha da classe de modelo importando a classe de estimador apropriada do Scikit-Learn.
- Escolha dos hiperparâmetros do modelo instanciando essa classe com os valores desejados.
- Organizando os dados em uma matriz de recursos e vetor de destino.
- Ajustando o modelo aos dados chamando o método fit() da instância do modelo.
- Aplicando o modelo a novos dados:
  - Para o aprendizado supervisionado, geralmente são previstos rótulos para dados desconhecidos usando o método predict().
  - Para aprendizagem não supervisionada, frequentemente são transformados ou inferidos as propriedades dos dados usando o método transform() ou predict().

No entanto, apesar de implementar um grande número e diferentes abordagens de algoritmos de aprendizado de máquina, cada modelo deve ser projetado, criado, treinado e testado independentemente. Dessa forma, a automação de etapas para seleção de abordagem de algoritmo é uma ótima alternativa na construção de um modelo razoável tanto para economia de tempo quanto para alcançar bons resultados.

Para aplicação dessa automação surge a biblioteca Pycaret. A intenção do Pycaret é que dado o conjunto de dados sob investigação, seja possível a criação de vários modelos utilizando diferentes algoritmos e abordagens. Sendo assim, uma vez que são criados, treinados e testados simultaneamente, é possível extrair um conjunto de respostas que podem ser utilizadas na decisão da escolha de qual abordagem final para representação dos dados será adotada. De uma forma simplificada, a biblioteca irá automatizar o processo de escolha de qual modelo de aprendizado de máquina seus dados estão com melhor aderência, economizando um tempo de analista razoável e reduzindo custos de implementação.

Entretanto, o que parece ser uma simplificação, pode não resolver tão facilmente o problema. Na área de aprendizado de máquina existem problemas típicos que podem surgir durante o processo de modelagem. O item mais importante que deve-se atentar é o ajuste ou otimização dos hiperparâmetros do algoritmo de aprendizagem de máquina. Uma vez que as técnicas utilizam heurísticas para solução do problema, cada algoritmo possui um

conjunto de restrições, pesos e taxas de aprendizado que são ajustadas caso a caso, não havendo um valor teórico, a priori, que garanta um melhor resultado. A complexidade desta tarefa varia de algoritmo adotado assim como pelos dados que estão sendo utilizados, tendo maior ou menor aderência ao tipo de convergência que o modelo possa utilizar no treinamento. Quando o conjunto de hiperparâmetros do algoritmo está ajustado, os dados irão ter grande aderência ao modelo trazendo um bom resultado do aprendizado de máquina.

3.6.2.2 Pycaret

De acordo com os desenvolvedores, o Pycaret é uma biblioteca de aprendizado de máquina de "baixa codificação"que permite desde a preparação dos dados à construção do modelo em poucos minutos e com poucas linhas de código. O Pycaret, na verdade, encapsula o conjunto de métodos desenvolvidos pela biblioteca Scikit-Learn e dentre outras afim de automatizar o processo de construção e análise de soluções de aprendizado de máquina.

Como ferramenta de aprendizado de máquina, Pycaret possui seis módulos para geração de modelos, descritos no Quadro 3.2.

Módulo	**Como Importar**
Classificação	from pycaret.classification import *
Regressão	from pycaret.regression import *
Análise de Conglomerado	from pycaret.clustering import *
Detecção de Anomalia	from pycaret.anomaly import *
Processamento de Linguagem Natural	from pycaret.nlp import *
Mineração de Regras de Associação	from pycaret.arules import *

Fonte – o autor.

Quadro 3.2 – Módulos da biblioteca Pycaret.

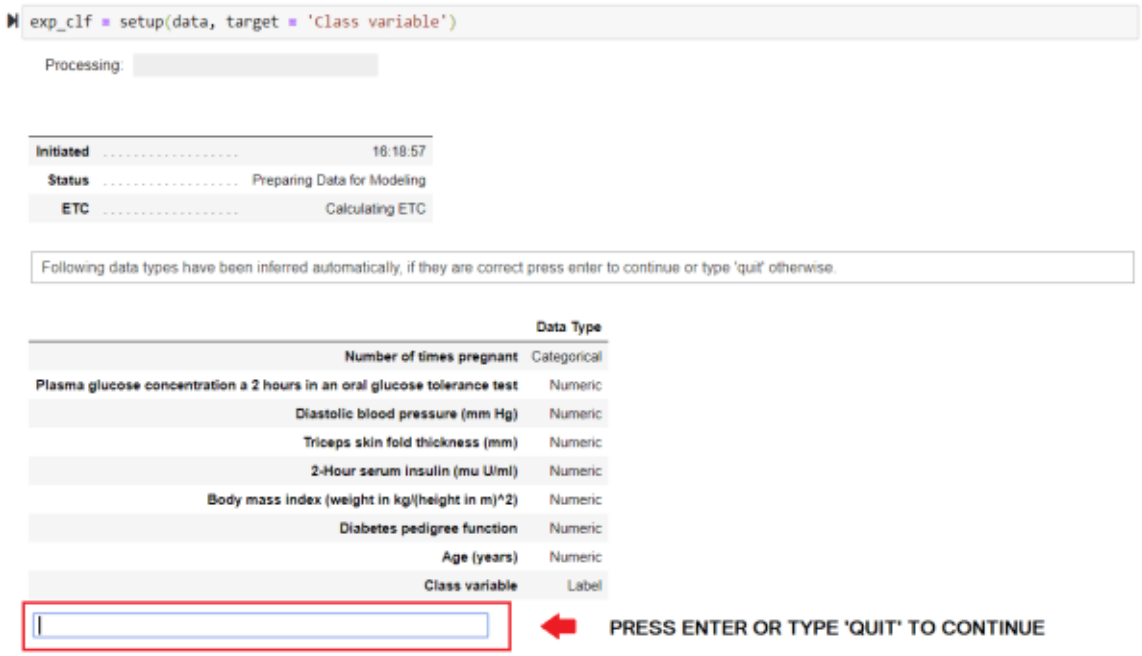
Durante o processo de modelagem com o Pycaret, algumas etapas devem ser seguidas: Inicialização, Treinamento do Modelo, Conjunto de Modelos, Análise do Modelo e Implementação do Modelo. Na etapa de inicialização são apenas realizadas a alocação dos dados para análise e a configuração (*setup*) dos parâmetros do Pycaret do projeto de ML.

Independente de qual tipo de aprendizado de máquina que será construída, o Pycaret exige que seja feito um *setup* que é o primeiro e único passo obrigatório. Na definição de um *setup* mínimo e básico, são necessários configurar: Inferência do Tipo de Dado; Limpeza dos Dados e Preparação; Amostragem dos Dados; Separação dos Conjuntos de Treino e Teste; Designação do ID da Sessão como Semente.

A primeira subetapa do *setup*, chamada Inferência do Tipo de Dado, inicia com a determinação do tipo de dado de cada característica ou variável. A função de *setup* percorre o conjunto de dados realizando inferência de qual é o tipo de cada dado. Depois que o

algoritmo do Pycaret infere sobre cada tipo de dado, é exibida uma tela com os possíveis resultados conforme a Figura 3.13.

Figura 3.13 – Inferência dos tipos de dados no *setup* do Pycaret.



Fonte – o autor.

Conforme é ilustrado na figura, se todos os tipos de dados forem inferidos corretamente, basta pressionar a tecla *ENTER* para continuar ou senão digitar *quit* para interromper o experimento. Caso haja interrupção porque um ou mais tipos não foram inferidos corretamente, é possível sobrescreve-los dentro do comando de *setup* modificando o parâmetro *catergorical_feature* para *force categorical type* e o parâmetro *numeric_feature* para *force numeric type*. Alem disso, colocar o parâmetro *ignore_features* para variáveis que você desprezará no modelo. Ainda é possível alterar *silent* para *True* para não exibir a tela de diálogo, o que pode gerar confusão caso os dados não sejam corretamente inferidos.

A segunda subetapa no *setup* do Pycaret é o preenchimento automático de valores ausentes e codificação de atributos categóricos, uma vez que é demandado para qualquer experimento em ML. O preenchimento padrão utiliza o valor médio da variável para tipos numéricos e o valor mais frequente (moda) para variáveis categóricas. Da mesma forma, aqui é possível alterar o método nos parâmetros *numeric_imputation* e *categorical_imputation*. Em casos de modelos de classificação, se o alvo não é um tipo numérico, é então realizado a codificação numérica.

Durante a terceira subetapa, na amostragem dos dados, se o espaço amostral é superior a 25 mil itens, o Pycaret constrói automaticamente de forma preliminar um modelo linear baseado em diferentes tamanhos de amostras e ilustra visualmente o desempenho do

modelo baseado em seu tamanho amostral. Se o espaço amostral é menor do que 25 mil é possível escolher o tamanho da amostra para que se tenha uma relação de eficiência versus desempenho adequado.

Na subetapa posterior na qual é realizada a divisão do conjunto de treinamento e de testes, a divisão padrão é de 70% dos dados amostrais para treinamento e 30% para testes, podendo ser alterado modificando o parâmetro *train_size* durante o setup. Nesta etapa de treinamento, o Pycaret realiza a otimização dos hiperparâmetros através da validação cruzada k-fold apenas sobre o conjunto de treinamento. A validação *k-fold* é muito utilizada em técnicas de ML. Este processo parte da divisão do conjunto de dados em *k* partes. Por exemplo, suponha que k=3: aplica-se então o primeiro conjunto como teste e os outros dois como conjuntos de treinamento, sendo avaliado o modelo ao final. Em seguida, aplica-se o segundo conjunto como teste e o primeiro e terceiro como treinamento, sendo avaliada sua desempenho ao final. Por fim, utiliza-se o terceiro grupo como teste e o primeiro e segundo grupos como treinamento com posterior avaliação de desempenho.

Durante a designação do ID da sessão, é gerado um número pseudo aleatório como semente se nenhum parâmetro *session_id* for informado. Este ID será utilizado como semente em todas funções de forma a se isolar o efeito de aleatoriedade, uma vez que determinados valores aleatórios poderiam beneficiar mais do que outros, criando um viés em algum modelo específico. Assim, é possível comparar algoritmos diferentes sob um mesmo pressuposto, garantindo reprodutibilidade de condições.

Na etapa de treinamento do modelo, existem as subetapas: Criar Modelo, Ajustar Hiperparâmetros e Comparar Modelos. Na criação do modelo é utilizada a função *create_model* que possui um argumento relativo ao algoritmo a ser implementado conforme os Quadros 3.3 e 3.4.

Classificação		**Regressão**	
ID	**Name**	**ID**	**Name**
'lr'	Logistic Regression	'lr'	Linear Regression
'knn'	K Nearest Neighbour	'lasso'	Lasso Regression
'nb'	Naives Bayes	'ridge'	Ridge Regression
'dt'	Decision Tree Classifier	'en'	Elastic Net
'svm'	SVM – Linear Kernel	'lar'	Least Angle Regression
'rbfsvm'	SVM – Radial Kernel	'llar'	Lasso Least Angle Regression
'gpc'	Gaussian Process Classifier	'omp'	Orthogonal Matching Pursuit
'mlp'	Multi Level Perceptron	'br'	Bayesian Ridge
'ridge'	Ridge Classifier	'ard'	Automatic Relevance Determination
'rf'	Random Forest Classifier	'par'	Passive Aggressive Regressor
'qda'	Quadratic Discriminant Analysis	'ransac'	Random Sample Consensus
'ada'	Ada Boost Classifier	'tr'	TheilSen Regressor
'gbc'	Gradient Boosting Classifier	'huber'	Huber Regressor
'lda'	Linear Discriminant Analysis	'kr'	Kernel Ridge
'et'	Extra Trees Classifier	'svm'	Support Vector Machine
'xgboost'	Extreme Gradient Boosting	'knn'	K Neighbors Regressor
'lightgbm'	Light Gradient Boosting	'dt'	Decision Tree
'catboost'	CatBoost Classifier	'rf'	Random Forest
		'et'	Extra Trees Regressor
		'ada'	AdaBoost Regressor
		'gbr'	Gradient Boosting Regressor
		'mlp'	Multi Level Perceptron
		'xgboost'	Extreme Gradient Boosting
		'lightgbm'	Light Gradient Boosting
		'catboost'	CatBoost Regressor

Fonte – o autor.

Quadro 3.3 – Algoritmos e módulos classificação e regressão da biblioteca Pycaret.

Análise de Conglomerados		**Detecção de Anomalia**	
ID	**Name**	**ID**	**Name**
'kmeans'	K-Means Clustering	'abod'	Angle-base Outlier Detection
'ap'	Affinity Propagation	'iforest'	Isolation Forest
'meanshift'	Mean shift Clustering	'cluster'	Clustering-Based Local Outlier
'sc'	Spectral Clustering	'cof'	Connectivity-Based Outlier Factor
'hclust'	Agglomerative Clustering	'histogram'	Retirado de: histogram-based Outlier Detection
'dbscan'	Density-Based Spatial Clustering	'knn'	k-Nearest Neighbors Detector
'optics'	OPTICS Clustering	'lof'	Local Outlier Factor
'birch'	Birch Clustering	'svm'	One-class SVM detector
'kmodes'	K-Modes Clustering	'pca'	Principal Component Analysis
		'mcd'	Minimum Covariance Determinant
		'sod'	Subspace Outlier Detection
		'sos	Stochastic Outlier Selection

Fonte – o autor.

Quadro 3.4 – Algoritmos e módulos de análise de conglomerado e detecção de anomalia da biblioteca Pycaret.

Nos casos dos módulos supervisionados, tanto para classificação quanto regressão, a função retorna uma tabela dos treinamentos com as métricas de desempenho com validação cruzada *k-fold*. Este número pode ser definido pelo parâmetro *fold* mas possui um valor padrão de 10, ou seja, dividido em 10 partes, tomando 9 para treinamento a cada ciclo.

Para o módulo de Análise de Conglomerados Não Supervisionados, é retornada a métrica de desempenho. Nestes casos citados anteriormente, a função também retorna o objeto instanciado do modelo treinado. Nos outros módulos do Pycaret: Detecção de Anomalia, Processamento de Linguagem Natural e Mineração de Regras de Associação, a função apenas retorna o objeto instanciado do modelo treinado.

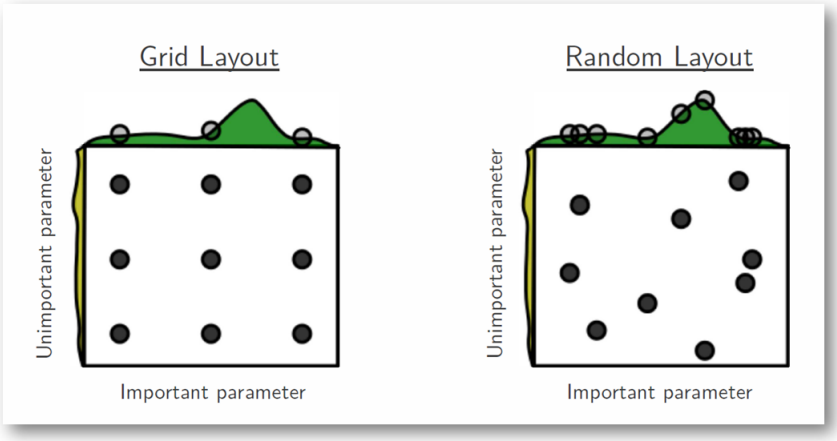
É importante fazer um esclarecimento sobre a diferença entre os conceitos de Parâmetro e Hiperparâmetros de um modelo de ML. Os parâmetros de um modelo são variáveis que são ajustadas durante o processo de treinamento enquanto os hiperparâmetros são outras variáveis que devem ser ajustadas antes do processo de treinamento. Pode-se estabelecer que os ajustes dos hiperparâmetros são realizados antes do treinamento para garantir como será a topologia do modelo, enquanto o posterior ajuste dos parâmetros irão atribuir o valor correto para estas variáveis. Exemplificando, o ajuste de hiperparâmetros de uma rede neural seria a definição do número de camadas e neurônios por camada, enquanto o posterior ajuste de parâmetros, que ocorre na etapa de treinamento, seria o ajuste correto dos pesos dos neurônios. Ambos ajustes de hiperparâmetros e parâmetros afetam significativamente na qualidade do modelo e ambos podem ser otimizados focando na minimização de alguma métrica de erro ou maximizando alguma métrica de desempenho.

O processo de otimização nos hiperparâmetros e parâmetros ocorre através de diversas execuções do modelo, alterando suas condições em cada execução. Para realizar este ajuste, é comum adotar uma das técnicas de otimização baseado em Busca em Grade (Grid) ou Busca Aleatória. Ambas as técnicas se resumem a explorar o domínio dos hiperparâmetros ou parâmetros de forma homogênea ou probabilística. A busca aleatória habitualmente é mais utilizada por utilizar um espaço de busca mais amplo e complexidade de tempo computacional inferior. A Figura 3.14 ilustra a diferença entre as técnicas de otimização.

Na busca em grade, os hiperparâmetros importantes são apresentados no eixo x e os de menor importância no eixo y, sendo criada uma grade na qual se realiza a amostragem dos hiperparâmetros de forma homogênea ao longo do domínio. No caso da busca aleatória, esta amostragem é feita de maneira casual.

Durante a subetapa de Ajuste de Hiperparâmetros do modelo de aprendizado de máquina, o Pycaret utiliza a função *tune_model*. Essa função ajusta os hiperparâmetros do modelo estimados através das técnicas de busca em grade ou aleatória, podendo ser configuradas. Nos casos de técnicas supervisionadas, tanto classificação quanto regressão, no momento de otimizar os hiperparâmetros é definida uma função objetivo associada ao alvo/resposta que se deseja que o modelo resolva. Nos casos dos módulos de Análise de Con-

Figura 3.14 – Tipos de técnicas para otimização dos hiperparâmetros.



Fonte – Retirado de: https://www.jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf

glomerados, Detecção de Anomalias e Processamento de Linguagem Natural, que são Não Supervisionadas, o Pycaret permite definir uma função objetivo especificando a variável alvo supervisionada através do parâmetro *supervised_target* que está no *tune_model*. Nos casos de aprendizado supervisionado, esta função retorna uma tabela com os valores das métricas sobre uma validação cruzada no formato *k-fold* já apresentado, que divide o grupo amostral em k partes, pegando uma de cada vez para testes e o restante para treinamento. A função também retorna o objeto instanciado do modelo treinado. Nos casos de aprendizado não supervisionado, a função apenas retorna o objeto do modelo treinado. As métricas de avaliação para Classificação na biblioteca são *Accuracy, AUC, Recall, Precision, F1, Kappa, MCC*. Já para Regressão, as métricas de avaliação são *MAE, MSE, RMSE, R2, RMSLE, MAPE*.

O número k das divisões da amostra pode ser configurado no parâmetro *fold* da função *tune_model*, sendo seu valor padrão 10. Todas as métricas são arredondadas para quatro dígitos decimais em seu valor padrão, mas podem ser alteradas no parâmetro *round*. Na condição padrão, a função irá executar 10 iterações aleatórias sobre o espaço de busca e pode ser alterada no parâmetro *n_iter* também em *tune_model*, que altera o tempo de treinamento e qualidade do modelo. Por definição padrão, modelos de Regressão utilizam *R2* como métrica de desempenho enquanto Classificação utiliza *Accuracy*, mas podem ser alterados como apresentado anteriormente.

Na subetapa posterior, Comparação de Modelos, o Pycaret treina todos os modelos da biblioteca utilizando os hiperparâmetros padrões e avalia as métricas de desempenho através da validação cruzada. Nesta função obtêm-se também os ou o melhor objeto do modelo treinado baseado nas métricas de desempenho apresentadas anteriormente. Como saída, exibe uma tabela com a nota média de todos os modelos ao longo das *k* partes da validação por *k-folds*. Para facilitar ao usuário, a tabela é ordenada na métrica de desempenho pelo maior para o menor valor, sendo possível selecionar o tipo de ordenação. Pelo valor

padrão, o Pycaret irá ordenar modelos de Classificação por *Accuracy* enquanto modelos de Regressão por *R2*. Se caso o modelo tenha uma longa duração de processamento em relação aos outros, o mesmo é penalizado e interrompido, o que pode ser alterado modificando o parâmetro *turbo* para *False*. Apenas como ilustração, a Figura 3.15 apresenta a tabela de comparação de um problema de Classificação.

Figura 3.15 – Tabela de comparação de modelos de Classificação.

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
0	Logistic Regression	0.8263	0.8959	0.7262	0.8139	0.7644	0.6280	0.6338	0.0513
1	Linear Discriminant Analysis	0.8263	0.8938	0.7536	0.7938	0.7713	0.6317	0.6342	0.0146
2	Ridge Classifier	0.8236	0.0000	0.7499	0.7920	0.7680	0.6262	0.6292	0.0182
3	Ada Boost Classifier	0.8075	0.8637	0.7053	0.7837	0.7398	0.5881	0.5924	0.0750
4	Gradient Boosting Classifier	0.8062	0.8869	0.7363	0.7651	0.7479	0.5909	0.5939	0.1283
5	CatBoost Classifier	0.8049	0.8932	0.7326	0.7629	0.7457	0.5878	0.5899	2.9858
6	Extreme Gradient Boosting	0.7914	0.8716	0.7294	0.7367	0.7309	0.5609	0.5633	0.0965
7	Light Gradient Boosting Machine	0.7861	0.8806	0.7053	0.7393	0.7195	0.5471	0.5497	0.1460
8	Quadratic Discriminant Analysis	0.7621	0.8240	0.6267	0.7397	0.6678	0.4863	0.5000	0.0142
9	Random Forest Classifier	0.7608	0.8397	0.6674	0.7124	0.6848	0.4928	0.4974	0.1119
10	Decision Tree Classifier	0.7594	0.7519	0.6911	0.6970	0.6907	0.4943	0.4975	0.0097
11	Extra Trees Classifier	0.7433	0.8205	0.6708	0.6758	0.6698	0.4605	0.4638	0.1581
12	K Neighbors Classifier	0.7231	0.7683	0.6062	0.6600	0.6287	0.4094	0.4129	0.0091
13	Naive Bayes	0.7140	0.7952	0.7466	0.6100	0.6708	0.4227	0.4301	0.0030
14	SVM - Linear Kernel	0.5267	0.0000	0.4200	0.2409	0.2561	0.0204	0.0299	0.0162

Fonte – Retirado de: https://pycaret.org/compare-models/

Na Figura 3.15 são grifados em amarelo os melhores campos entre os modelos. Assim, da tabela exemplificada, o modelo mais adequado selecionado seria a Análise de Discriminante Linear pelo seu maior número de "melhores resultados"em diferentes métricas entre todos os outros modelos.

Na etapa seguinte, Conjunto de Modelos, a primeira opção é o Modelo de Conjunto, que é a utilização da função *ensemble_model* que toma o objeto do modelo treinado da etapa anterior como argumento. Apenas é possível utilizar esta função para modelos de Classificação e Regressão. Nesta opção de Modelo de conjunto, o objeto do modelo treinado é melhorado através de duas tarefas: Ensacamento (*Bagging*) e Impulsionamento (*Boosting*).

O Ensacamento, ou agregação por *Bootstrap*, é um meta-algoritmo desenvolvido para melhorar a estabilidade e acurácia dos algoritmos tanto para classificação quanto regressão. Ele também reduz a variância e ajuda a evitar o sobreajuste (*overfitting*). Este método é mais utilizado em métodos de árvores de decisão mas pode ser utilizado em diversos outros tipos. O uso da função é bastante simples, como por exemplo, num modelo de árvore de decisão

(*Decision Tree = dt*), o modelo corrigido seria obtido através da adição da linha de código *bagged_dt = ensemble_model(dt, method = 'Bagging')*.

A técnica de Impulsionamento é um conjunto de meta-algoritmos que reduzem o viés e variância em aprendizado supervisionado. Essa técnica faz parte da família de algoritmos de aprendizado de máquina que convertem alunos fracos em fortes na aprendizagem de grupos. Esta definição de aluno fraco corresponde a quando um classificador é levemente correlacionado com a classificação correta, predizendo exemplos melhores do que apenas uma escolha aleatória da resposta. Em contrapartida, um aluno forte é um classificador que está fortemente correlacionado com a predição correta de classificação. Da mesma forma que o ensacamento, a correção do modelo é facilmente obtida usando Pycaret através da linha de código *boosted_dt = ensemble_model(dt, method = 'Boosting', n_estimators = 100)*. Em ambas funções de correção, é necessário reamostrar os dados e ajustar os estimadores. Estes estimadores podem ser definidos no parâmetro *n_estimators* cujo valor padrão é 10. Após a aplicação de ambas funções, o Pycaret exibe uma tabela com estes estimadores e o valor médio conforme a Figura 3.16 que ilustra um exemplo de Classificação.

Figura 3.16 – Tabela de estimadores após correção para um modelo de Classificação.

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.7467	0.8223	0.6552	0.6786	0.6667	0.4625	0.4627
1	0.8533	0.9040	0.6897	0.9091	0.7843	0.6763	0.6912
2	0.7067	0.7864	0.5862	0.6296	0.6071	0.3736	0.3742
3	0.7867	0.8606	0.7586	0.7097	0.7333	0.5559	0.5567
4	0.7200	0.8531	0.6897	0.6250	0.6557	0.4207	0.4222
5	0.7200	0.8396	0.5517	0.6667	0.6038	0.3902	0.3944
6	0.8133	0.8604	0.7667	0.7667	0.7667	0.6111	0.6111
7	0.8133	0.8811	0.7333	0.7857	0.7586	0.6067	0.6077
8	0.7838	0.8383	0.6207	0.7826	0.6923	0.5290	0.5375
9	0.8649	0.8977	0.7586	0.8800	0.8148	0.7093	0.7142
Mean	0.7809	0.8543	0.6810	0.7434	0.7083	0.5335	0.5372
SD	0.0534	0.0336	0.0723	0.0939	0.0704	0.1127	0.1147

Fonte – Retirado de: https://pycaret.org/ensemble-model/

Outra opção de correção de modelos dentro do Conjunto de Modelos do Pycaret é a Mistura de Modelos (*Blend Model*) e o Empilhamento (*Stacking*), embora este segundo esteja sendo descontinuado nas próximas versões. A Mistura de Modelos é uma abordagem que permite utilizar o conjunto de respostas preditas de diversos tipos de modelos, usualmente os com melhores resultados. Assim, é possível realizar uma correção ou ajuste no modelo final considerando diversas abordagens e vantagens. Esta mistura de modelos pode

ser construída, por exemplo, ponderando o resultado predito por três modelos diferentes. No Pycaret a Mistura de Modelos utiliza um consenso entre os estimadores para gerar predições finais através da função *blend_models*. A função pode utilizar o parâmetro *estimator_list* para definir especificamente os modelos treinados, ou caso omitido, utilizará todos os modelos da biblioteca. Nos casos de Classificação o parâmetro *method* pode ser definido como *soft* ou *hard* para usar probabilidades preditas para votação ou usar rótulos preditos. Da mesma forma que anteriormente, a função retorna uma tabela com as notas das métricas e o objeto do modelo treinado corrigido. Um exemplo do uso da função seria o código *blender_specific = blend_models(estimator_list = ['dt','rf','adaboost'], method = 'soft')* no qual compara os modelos de árvore de decisão (*dt*), floresta aleatória (*rf*) e adaboost.

Dando continuidade ao fluxo, a etapa de Análise do Modelo é composta por: Gráficos do Modelo, Interpretação do Modelo, Atribuir Modelo, Calibração do Modelo e o Limite de Otimização. Indiferente quanto ao modelo escolhido, a análise do desempenho do modelo construído é essencial. Uma das melhores formas de compreensão dos resultados do modelo é através de resultados visuais em gráficos que o Pycaret tem na função *plot_model*. Esta função recebe o objeto do modelo treinado e o tipo de gráfico e exibe. Os tipos de gráficos possíveis estão nos Quadros 3.5, 3.6 e 3.7.

Classificação		Regressão	
Name	**Plot**	**Name**	**Plot**
Area Under the Curve	'auc'	Residuals Plot	'residuals'
Discrimination Threshold	'threshold'	Prediction Error Plot	'error'
Precision Recall Curve	'pr'	Cooks Distance Plot	'cooks'
Confusion Matrix	'confusion_matrix'	Recursive Feature Selection	'rfe'
Class Prediction Error	'error'	Learning Curve	'learning'
Classification Report	'class_report'	Validation Curve	'vc'
Decision Boundary	'boundary'	Manifold Learning	'manifold'
Recursive Feature Selection	'rfe'	Feature Importance	'feature'
Learning Curve	'learning'	Model Hyperparameter	'parameter'
Manifold Learning	'manifold'		
Calibration Curve	'calibration'		
Validation Curve	'vc'		
Dimension Learning	'dimension'		
Feature Importance	'feature'		
Model Hyperparameter	'parameter'		

Fonte – o autor.

Quadro 3.5 – Tipos de gráficos por módulos de classificação e regressão da biblioteca Pycaret.

É possível perceber que existe uma grande quantidade de tipos de gráficos que podem ser utilizados para cada módulo durante a construção do modelo, lembrando que estas ferramentas gráficas são fatores chaves na escolha e determinação do algoritmo.

Dentro da etapa de Análise do Modelo existe também a Interpretação do Modelo que

Análise de Conglomerados		Detecção de Anomalia	
Name	**Plot**	**Name**	**Plot**
Cluster PCA Plot (2d)	'cluster'	t-SNE (3d) Dimension Plot	'tsne'
Cluster TSnE (3d)	'tsne'	UMAP Dimensionality Plot	'umap'
Elbow Plot	'elbow'		
Silhouette Plot	'silhouette'		
Distance Plot	'distance'		
Distribution Plot	'distribution'		

Fonte – o autor.

Quadro 3.6 – Tipos de gráficos por módulos de análise de conglomerados e detecção de anomalia da biblioteca Pycaret.

Processamento de Linguagem Natural		Mineração de Regras de Associação	
Name	**Plot**	**Plot**	**Abbrev. String**
Word Token Frequency	'frequency'	Support, Confidence and Lift (2d)	'frequency'
Word Distribution Plot	'distribution'	Support, Confidence and Lift (3d)	'distribution'
Bigram Frequency Plot	'bigram'		
Trigram Frequency Plot	'trigram'		
Sentiment Polarity Plot	'sentiment'		
Part of Speech Frequency	'pos'		
t-SNE (3d) Dimension Plot	'tsne'		
Topic Model (pyLDAvis)	'topic_model'		
Topic Infer Distribution	'topic_distribution'		
Word cloud	'wordcloud'		
UMAP Dimensionality Plot	'umap'		

Fonte – o autor.

Quadro 3.7 – Tipos de gráficos por módulos de processamento de linguagem natural e mineração de regras de associação da biblioteca Pycaret.

é um ponto importante nas técnicas de ML. Nesta etapa, podem ser compreendidos alguns aspectos qualitativos do modelo e o Pycaret a executa através da função *interpret_model*. A função recebe também o objeto do modelo treinado e um tipo de gráfico como parâmetros. Esta interpretação é implementada baseada no SHAP (*SHapley Additive exPlanations*) e está disponível para interpretar qualquer modelo de aprendizado de máquina ou aprendizado profundo.

3.6.2.3 SHAP

Não existe uma definição matemática de interpretabilidade. Uma definição (não matemática) dada por Miller (2018) é que a interpretabilidade é o grau em que um ser humano pode compreender a causa de uma decisão. Outra dada por Kim, Khanna e Koyejo (2016) é que a interpretabilidade é o grau em que um ser humano pode prever de forma consistente o resultado do modelo. Quanto maior a interpretabilidade de um modelo de aprendizado de máquina, mais fácil será para alguém compreender por que certas decisões ou previsões foram feitas. Um modelo é melhor interpretável do que outro modelo se suas decisões são

mais fáceis de serem compreendidas por um ser humano do que as decisões do outro modelo (MOLNAR, 2019).

O objetivo desta etapa de Interpretação do Modelo com o SHAP é de esclarecer e compreender os aspectos do modelo de aprendizado de máquina, uma vez que na grande maioria, o modelo acaba se tornando uma caixa preta que apenas apresenta resultados. O SHAP busca explicitar cada variável e seus impactos no modelo para decisão da resposta. Assim, o modelo de aprendizado de máquina não apenas pode ser utilizado para predição de resultados, mas também esclarecer a verdadeira relação entre as características de entrada e a resposta.

SHAP (SHapley Additive exPlanations) é uma abordagem da teoria de jogos para explicar a saída de qualquer modelo de aprendizado de máquina ou aprendizado profundo. Ele conecta a alocação de crédito ideal com explicações locais usando os valores clássicos de Shapley da teoria dos jogos e suas extensões relacionadas.

O emprego dessa interpretação pode ser feito utilizando os métodos da biblioteca Pycaret integrados da biblioteca SHAP. Os métodos disponíveis são *summary_plot, correlation_plot* e *force_plot*. Todos esses e outros métodos podem ser usados nativamente através da biblioteca SHAP, instanciando um objeto denominado *explainer* utilizando como parâmetro o modelo desenvolvido e então proceder com a análise.

Um dos gráficos que pode ser explorado no SHAP é ilustrado pela Figura 3.17. Este gráfico é obtido pela função *interpret_model* no Pycaret, passando apenas o modelo, sem especificação de gráfico, como gráfico padrão. O mesmo gráfico também pode ser obtido com a função nativa *summary_plot* da biblioteca SHAP.

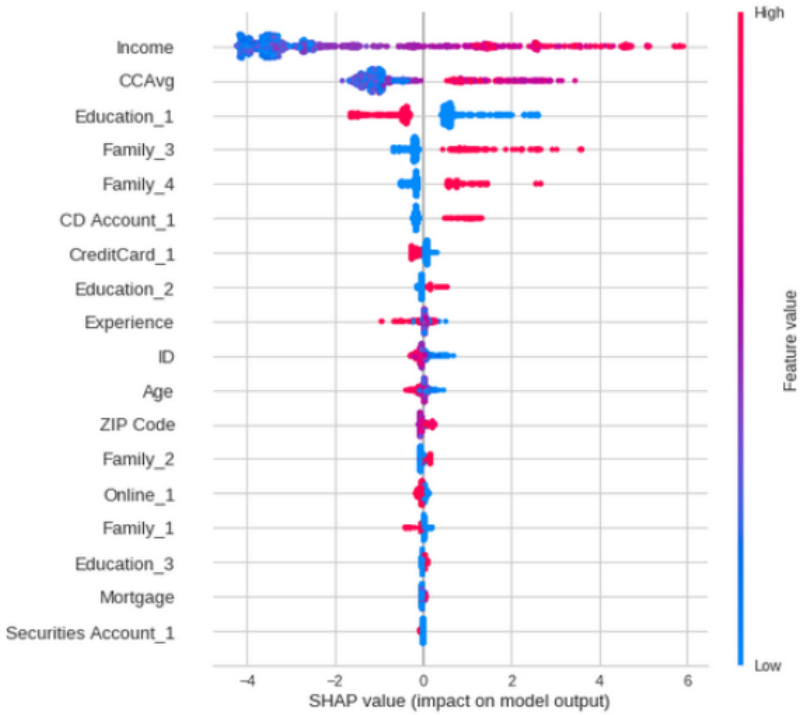
Perceba que pode-se compreender os valores de cada característica (parâmetro de entrada do modelo) de todas as observações através da Figura 3.17, de forma a se avaliar o nível com que cada recurso impacta na resposta do modelo. Além disso, é possível identificar viés dos recursos, descartar recursos com pouco impacto, identificar recursos com grande variabilidade de impacto, indicando assim a existência de ruídos.

É possível obter um gráfico de correlação ao se modificar o parâmetro *plot* como o exemplo *interpret_model(modelo, plot = 'correlation')*. Este tipo de gráfico é ilustrado na Figura 3.18 para um caso hipotético.

A análise sobre a Figura 3.18 não muda muito em relação a um gráfico comum de correlação. A maior mudança é o eixo y indicando o impacto dos recursos. Nele é possível extrair pontos onde o impacto muda de sentido, relações que podem ser combinadas de modo a produzir um terceiro recurso com um impacto maior e entre outras informações.

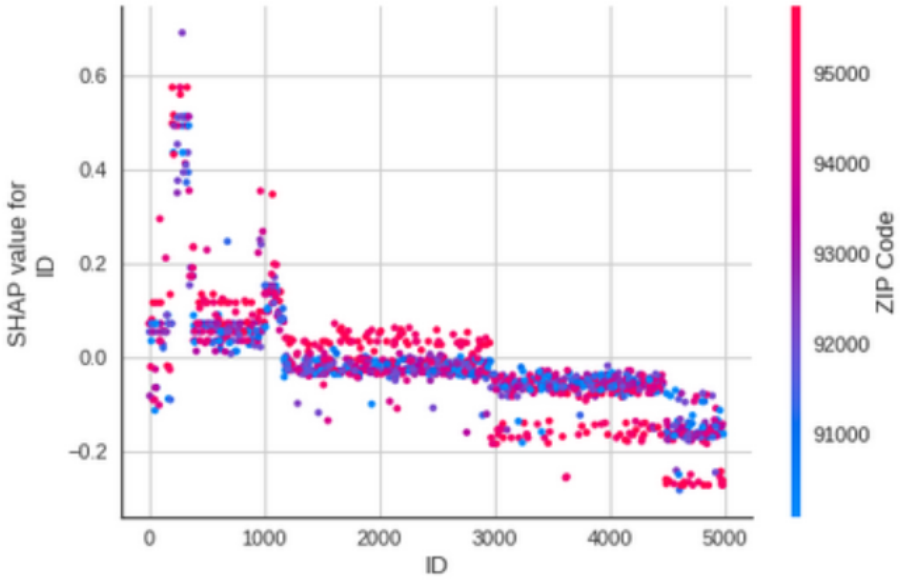
O último gráfico gerado pelo SHAP e Pycaret para interpretação do modelo é o Gráfico de Razão. Um exemplo é apresentado na Figura 3.19.

Figura 3.17 – Gráfico do impacto das características das observações na resposta do modelo.



Fonte – Retirado de: https://www.analyticsvidhya.com/blog/2020/05/pycaret-machine-learning-model-seconds/

Figura 3.18 – Gráfico da correlação entre duas características de entrada do modelo



Fonte – Retirado de: https://www.analyticsvidhya.com/blog/2020/05/pycaret-machine-learning-model-seconds/

A Figura 3.19 apresenta o gráfico com a capacidade de incorporar maior compreensão sobre os dados, uma vez que irá analisar uma observação específica do conjunto de treino (um elemento amostral, aqui denominado argumento razão). Nele nota-se dois conjuntos de setas azuis e vermelhas. As azuis representam as variáveis desta observação específica

Figura 3.19 – Gráfico de Razão para uma observação do modelo.

Fonte – Retirado de: https://towardsdatascience.com/announcing-pycaret-an-open-source-low-code-machine-learning-library-in-python-4a1f1aad8d46/

empurrando para a direção negativa da variável resposta. Em oposição, as setas vermelhas descrevem as variáveis e seus valores da observação que empurram a variável resposta para uma direção positiva. Este gráfico é de extrema validade uma vez que apresenta qualitativamente o grau de importância das variáveis de entrada para o resultado de saída. No entanto, é importante ressaltar que o gráfico é criado para cada observação, tornando-se às vezes improdutivo realizar a análise, porém muito útil em outros momentos.

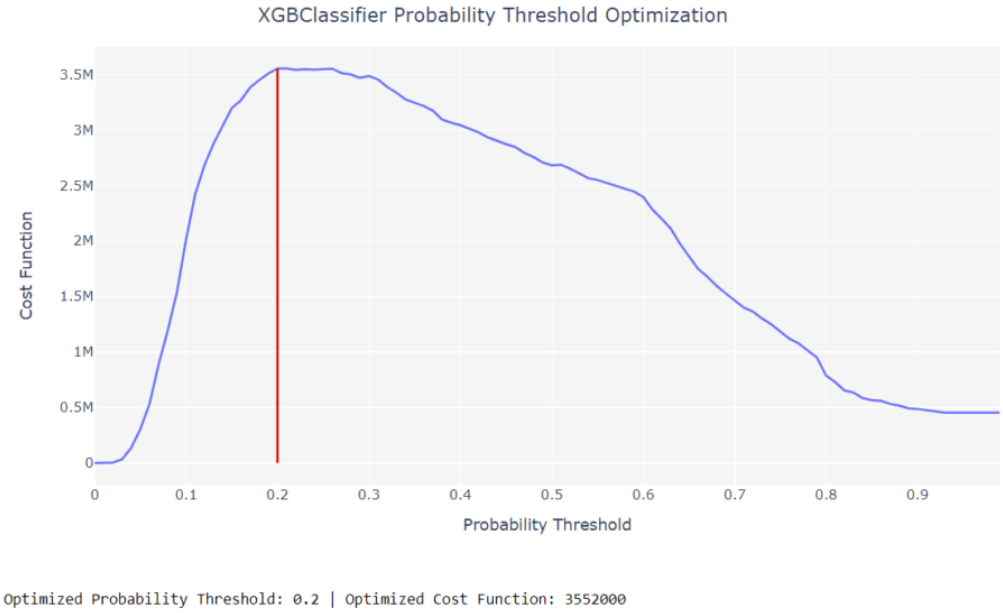
No Pycaret a funcionalidade Atribuir Modelo é utilizada para os módulos de Análise de Conglomerados, Detecção de Anomalias e Processamento de Linguagem Natural. Em um problema de agrupamento, o algoritmo irá identificar o grupo ao qual um elemento pertence a partir de rótulos gerados pelo modelo. Da mesma forma, num caso de detecção de anomalias um rótulo binário será gerado para os *outliers* ou, no caso de um modelo de processamento de linguagem natural, gerará rótulos ao qual um tópico de um documento pertence. A função no Pycaret para isto é a *assign_model* e apenas utiliza como parâmetro o objeto do modelo treinado.

Um outro recurso da Análise do Modelo no Pycaret é a Calibração do Modelo. Esse recurso é aplicado em modelos de Classificação, uma vez que em muitos casos não se deseja apenas obter a classe da resposta, mas também a probabilidade da predição, passando algum tipo de confiança sobre a correta classificação. É comum encontrar em alguns modelos, casos de baixa estimativa de probabilidade. Esta função no Pycaret se chama *calibrate_model* e toma como argumentos o objeto do modelo treinado e o método de calibração que pode ser ajustado no parâmetro *method*. Uma vez que pode levar a condições de overfitting, não é recomendável o uso da calibração isotônica com poucas amostras (muito menores que 1000). Da mesma forma como em outros casos, a função retorna uma tabela com as notas de validação cruzada das k-partes das métricas de avaliação da classificação e o objeto modelo treinado ajustado.

Por fim, a última funcionalidade da Análise do Modelo no Pycaret é o Limite de Otimização. Este limite está associado a modelos de Classificação e lida com o custo dos falsos positivos serem diferentes dos falsos negativos. Assim, se estiver modelando um problema em que o custo dos erros do Tipo 1 e 2 gerem diferentes impactos, é possível no Pycaret de-

finir o custo para verdadeiros positivos, verdadeiros negativos, falsos positivos e falsos negativos separadamente. A função se denomina *optimize_threshold* e utiliza como argumentos o objeto do modelo treinado e a função de perda representada pelos valores verdadeiros positivos, verdadeiros negativos, falsos positivos e falsos negativos. Um gráfico é gerado como resultado com a função de perda no eixo y em relação a diferentes valores de limites de probabilidade no eixo x. Além disto, uma linha vertical é apresentada para o melhor valor de limite (*threshold*) para aquele classificador. Em geral, os classificadores são treinados para predizer as classes positivas com 50%. Como exemplo hipotético da função, um modelo xgboost poderia apresentar a configuração *optimize_threshold(xgboost, true_negative = 1500, false_negative = -5000)* gerando a Figura 3.20.

Figura 3.20 – Gráfico do Limite de Otimização.



Fonte – Retirado de: https://pycaret.org/optimize-threshold/

Concluindo todas as funcionalidades da Análise do Modelo no Pycaret, existe a Implementação do Modelo, que possui as funcionalidades: Implantar o Modelo, Salvar o Modelo, Modelo de Previsão e Finalização do Modelo.

A primeira funcionalidade, Implantação do Modelo corresponde a possibilidade de se implantar o modelo localmente ou na nuvem (servidores na internet) através do Pycaret. Para implantar o modelo de forma local, a função Salvar o Modelo é utilizada através do *save_model* que pega como argumento o objeto do modelo treinado e salva o pipeline de transformação completo e o objeto do modelo treinado como um arquivo binário transferível para uso posterior.

Para implantar na nuvem utiliza-se a função *deploy_model*. Para usuários AWS, antes de implantar o modelo para um AWS S3, as variáveis de ambiente devem estar configuradas através da interface de linhas de comando. Esta tarefa pode ser realizada através das linhas

de comando prompt do Python através da função *aws configure*. Para realizar a implantação serão solicitados: AWS Access Key ID, AWS Secret Key Access, Nome da Região Padrão, formato de saída padrão (deixado em branco).

Uma vez que o modelo foi construído com sucesso e salvo na nuvem ou localmente, ele pode ser utilizado para realizar predições de dados desconhecidos através da função *predict_model*. É utilizado como argumentos de entrada o objeto do modelo treinado e o conjunto de dados a ser previsto. Assim, o método irá aplicar automaticamente o pipeline de transformação inteira criada durante a modelagem. Para problemas de Classificação, os rótulos de predição são criados com 50% de probabilidade, mas pode ser alterado através do Limite de Otimização explicado anteriormente.

Finalização do Modelo é a última etapa da modelagem de um problema supervisionado. Na criação de um modelo em Pycaret, na etapa de *setup* é criado um conjunto de validação que não é utilizada no modelo de treinamento. Por padrão, se o parâmetro *train_size* não for definido, o Pycaret irá separar 30% dos dados para este conjunto de validação e todas as outras funções do Pycaret irão utilizar os 70% restantes do conjunto de treinamento para criar, ajustar e agrupar os modelos. Assim, este conjunto de validação é uma garantia final e utilizado para testes de sobreajuste e subajuste. Todavia, uma vez que as predições através do *predict_model* são aplicadas no conjunto de validação, e foi escolhido um modelo específico para implementar, é treinado o modelo uma última vez com todos os dados, inclusive os do conjunto de validação. A função que finaliza este modelo é a *finalize_model* e utiliza como argumentos o objeto do modelo treinado e retorna um modelo treinado em todo o conjunto de dados.

3.6.3 Implantação de Modelos *Deploy*

Após as etapas de tratamento de dados e desenvolvimento dos modelos, a última etapa a ser realizada em um projeto de aprendizado de máquina é a implantação da aplicação. Existem diversas ferramentas voltadas para a implantação de aplicações de aprendizado de máquina, as principais são: Streamlit, Plotly, Shiny, Flask, Django e dentre outras. Destas ferramentas todas possuem seus requisitos como tempo para implementação da versão final e recursos computacionais (servidores, processamento). Baseado nesses aspectos, a biblioteca *open-source* Streamlit possui como vantagem a fácil e rápida implementação além de possibilitar colocar em produção sem a necessidade de um servidor ou recurso computacional expressivo (a depender da aplicação).

A arquitetura do Streamlit permite implementar aplicativos da mesma maneira que são implementados *widgets* em Python. Os aplicativos Streamlit têm um fluxo de dados exclusivo: sempre que algo precisa ser atualizado na tela, o Streamlit executa novamente todo o *widgets* Python de cima para baixo. Isso pode acontecer em duas situações:

- Sempre que houver modificações no código-fonte do aplicativo.
- Sempre que um usuário interagir com *widgets* no aplicativo. Por exemplo, ao arrastar um controle deslizante, inserir texto em uma caixa de entrada ou clicar em um botão.

O Streamlit faz um trabalho pesado nos bastidores utilizando o *@st.cachedecorador*, que permite aos desenvolvedores pular certos cálculos caros quando os aplicativos são executados novamente.

Existem algumas maneiras de exibir dados (tabelas, matrizes, quadros de dados) em aplicativos Streamlit. Para se escrever qualquer coisa na tela usa-se a função *st.write()*, desde texto a tabelas. Essa função inspeciona o tipo de dados que foi passado, e então decide como melhor renderizá-los no aplicativo. Além disso, a função *st.write()* possuí métodos que retornam um objeto que pode ser usado e modificado, adicionando ou substituindo dados a ele. Além disso, é possível modificar o comportamento do mesmo passando atributos adicionais.

No Streamlit os elementos como botões, campo de texto, *sliders*, caixas de seleção, entre outros, são tratados como *widgets*. Esses elementos são como variáveis, por exemplo: *x = streamlit.slider('x')* e para realizar alguma tarefa com o valor desse campo basta utilizar a função *st.write(x, 'x ao quadrado é: ', x * x)*.

O layout das aplicações do Streamlit possuem duas regiões gerais: a barra lateral esquerda *st.sidebar* e o corpo principal *st*. Cada elemento que é passado para a barra lateral à esquerda, permiti que os usuários se concentrem no conteúdo do aplicativo enquanto ainda têm acesso aos controles da UI (*User Interface*).

Alem das regiões do layout principal, o Streamlit disponibilizou mais 4 recursos de layout, os *st.beta_columns*: colunas lado a lado onde pode inserir elementos Streamlit; *st.beta_expander*: um widget de expandir / recolher para mostrar elementos seletivamente; *st.beta_container*: o bloco de construção fundamental do layout e *with column1: st.write("hi!")*: melhoria na sintaxe para especificar qual *container* usar.

Outra ferramenta muito importante para o desenvolvimento de aplicações é o gerenciador de *cache*. O *widgets* Streamlit permite que o aplicativo seja executado rapidamente, mesmo ao carregar dados da web, manipular grandes conjuntos de dados ou realizar cálculos caros. Para se utilizar essa ferramenta deve-se declarar a função com o *@st.cachedecorador*, que informa ao Streamlit sempre que a função é chamada, é necessário verificar:

- Os parâmetros de entrada com os quais executou a função;
- O valor de qualquer variável externa usada na função;
- O corpo da função;

- O corpo de qualquer função usada dentro da função em *cache*.

Se esta for a primeira vez que o Streamlit viu esses quatro componentes com esses valores exatos e nesta combinação e ordem exatas, a aplicação executa a função e armazena o resultado em um *widgets* local. Então, na próxima vez que a função em *widgets* for chamada, se nenhum desses componentes for alterado, o Streamlit simplesmente ignorará a execução da função e, em vez disso, retornará a saída armazenada anteriormente no cache. A função tem a capacidade de receber atributos para comportamentos mais amplos como, por exemplo, compartilhamento de *widgets* entre usuários.

Dessa forma as aplicações que utilizam a biblioteca Streamlit realizam as seguintes tarefas em sequência, ou seja, são executados de cima para baixo:

- Cada vez que um usuário abre uma guia do navegador apontando para seu aplicativo, o *widgets* é executado novamente;
- Conforme o *widgets* é executado, o Streamlit desenha sua saída ao vivo em um navegador;
- Os *widgets* usam o Streamlit para evitar a re-computação de funções caras, de modo que as atualizações acontecem rapidamente;
- Cada vez que um usuário interage com um widget, o *widgets* é executado novamente e o valor de saída desse widget é configurado para o novo valor durante a execução.

3.6.4 Docker

Durante os últimos anos, os serviços em nuvem foram se transformando de arquiteturas monolíticas para arquiteturas baseadas em microsserviços, em que cada serviço cuidava de um conjunto limitado de funções (ESPOSITO; CASTIGLIONE; CHOO, 2016; VILLAMIZAR *et al.*, 2015). Os benefícios do uso de arquiteturas de microsserviço incluem: flexibilidade, escalabilidade, eficiência, complexidade reduzida e gerenciabilidade aprimorada. Microsserviço pode ser implementado usando tecnologia de contêiner, onde apenas um ou poucos processos rodam dentro de um único contêiner (AMARAL *et al.*, 2015). Os contêineres Docker fornecem uma tecnologia leve, de baixa sobrecarga e rápida, permitindo o uso de arquiteturas de microsserviço (CELESTI *et al.*, 2016; ISLAM *et al.*, 2019).

Um contêiner é uma unidade padrão de software que empacota o código e todas as suas dependências para que o aplicativo seja executado de forma rápida e confiável de um ambiente de computação para outro (DOCKER, 2020b; UPHILL *et al.*, 2017). Uma imagem de contêiner do Docker é um pacote de software leve, autônomo e executável que inclui tudo o que é necessário para executar um aplicativo: código, tempo de execução, ferramentas do sistema, bibliotecas do sistema e configurações.

Docker é uma ferramenta que ajuda a explorar a virtualização no nível do sistema operacional para desenvolver e entregar software em contêineres. O Docker é um ambiente completo onde se constrói e implanta softwares. É como uma máquina Linux, exceto que é muito leve, rápida e não tem nada exceto o que é necessário para que o projeto ou software possa rodar sem falhas. Com o uso da conteinerização com o Docker é possível mover aplicações entre plataformas e ainda executá-lo sem instalar uma única dependência (DOCKER, 2020b; SHAH; DUBARIA, 2019).

O Docker pode atuar como um ambiente de desenvolvimento completo e sólido, na fase de implantação não exigindo uma grande configuração do ambiente de produção. Afim de corresponder aos ambientes de teste e desenvolvimento, na computação distribuída permite que aplicativos possam ser divididos em módulos e implantado em diferentes máquinas, todas comunicando-se entre si e gerenciadas por uma única interface. Na utilização das GPUs (Unidade de Processamento Gráfico), permite uma fácil utilização da GPU pelos contêineres Docker, possibilitando que algoritmos de aprendizado profundo ávidos por recursos sejam executados em grandes conjuntos de dados, sem desperdiçar quaisquer recursos fornecidos pela GPU.

O DockerFile é um arquivo de instruções usado para construir contêineres. Construir uma imagem é um processo incremental. É possível baixar a imagem do aplicativo e usar um DockerFile para instalar componentes em cima dele como uma versão específica, por exemplo, do Python. Pode-se adicionar novos pacotes, alterar arquivos de configuração, criar novos usuários e grupos e até mesmo copiar arquivos e diretórios para a imagem.

Uma ferramenta útil para gerenciar os microserviços é o *docker-compose*, a qual é uma ferramenta para definir e executar aplicativos Docker de vários contêineres. Com o Compose, utiliza-se um arquivo YAaprendizado de máquina para configurar os serviços do aplicativo. Então, com um único comando, é possível criar e iniciar todos os serviços de sua configuração (DOCKER, 2020a).

O Compose funciona em todos os ambientes: produção, teste, desenvolvimento, bem como fluxos de trabalho. Para usar o Compose é necessário executar um processo de três etapas:

- Define o ambiente do aplicativo com um Dockerfile para que possa ser reproduzido em qualquer lugar;
- Define os serviços que compõem o aplicativo com um arquivo *docker-compose.yml* para que possam ser executados juntos em um ambiente isolado;
- Com o comando *docker-compose up*, inicia e executa todo o aplicativo (todos os microsserviços).

A disponibilização de aplicativos de inteligência artificial necessitam de uma alta confiabilidade, ou seja, alta tolerância a falhas e alta consistência nos resultados. Essas necessidades se dão pela iteração com sistemas críticos, dos quais lidam diretamente com segurança de pessoas ou que geram grandes perdas financeiras caso falhem. A disponibilidade das informações em diversas plataformas e em qualquer momento são importantes, tanto para análise dos sistemas monitorados, como também para manutenções ou melhorias nos métodos (XU, 2020).

O emprego da computação em nuvem garante esses pré-requisitos sem contar que entrega outros benefícios como: pagar conforme o uso (*pay as you go*), alta disponibilidade de recursos de processamento para os modelos e rápida implantação.

Uma ferramenta útil para criação de ambientes Docker na nuvem é o Docker Machine a qual é uma ferramenta que permite instalar o Docker Engine em hosts virtuais e gerenciar os hosts com o comando *docker-machine*. É possível usar o Docker Machine para criar hosts Docker para os sistemas operacionais Windows, Mac, linux, na rede da empresa, em datacenters ou em provedores de nuvem como Azure, AWS ou DigitalOcean.

3.7 Aprendizado de Máquina Aplicado ao SHM

Giurgiutiu e Kropas-Hughes (2003) realizaram um experimento afim de determinar danos em um disco de alumínio. Os intervalos de frequência foram coletados em 10-40 kHz, 10-150 kHz e 300-450 kHz para 5 situações de dano, cada situação contendo 5 amostras. Um algoritmo de extração de características foi usado para determinar as frequências de ressonância e amplitudes contidas nesses espectros de alta frequência. Os vetores de características foram usados como entrada para uma Rede Neural Probabilística (PNN - do inglês *Probabilistic Neural Network*). O treinamento foi realizado usando um membro selecionado aleatoriamente de cada uma das 5 classes de dano, enquanto a validação foi realizada em todos os membros restantes. Quando o vetor de características tinha um tamanho pequeno, algumas classificações erradas foram observadas. Ao aumentar o tamanho do vetor de características, uma classificação excelente foi obtida em todos os casos.

Park *et al.* (2008b) realizaram um experimento afim de avaliar a integridade estrutural em perda de parafusos, onde o experimento apresenta duas vigas de alumínio acopladas por 4 parafusos. Para avaliação da integridade, inicialmente foi aplicada Análise de Componente Principal (PCA - do inglês *Principal Component Analysis*) como um módulo de pré-processamento para reduzir a dimensionalidade dos dados e eliminar os ruídos indesejados. Aplicou-se *K-means* para identificar os padrões das estruturas com perda de parafuso.

Park *et al.* (2008a) fizeram um experimento com a intenção de detectar danos em um trilho de ferro utilizando um classificador baseado em SVM. Os danos propostos foram aplicados tanto na alma do perfil quanto na cabeça do perfil. O dano na alma foi um buraco de 5

mm situado a 400 mm da extremidade e o dano na cabeça situado a 500 mm da extremidade. Utilizaram duas vezes as SVMs, inicialmente para detectar as assinaturas com dano e posteriormente categorizar essas assinaturas. O trabalho apresenta alta capacidade de detecção e classificação de danos.

Min, Park e Yun (2010) investigaram a seleção autônoma de uma faixa de frequência adequada para a técnica EMI usando uma técnica de Rede Neural Artificial (RNA). Primeiro, os sinais de impedância foram obtidos em uma ampla banda de frequência e os sinais foram divididos em várias subfaixas dessa banda ampla. Em seguida, o índice de dano predefinido foi avaliado para cada subfaixa, comparando os sinais de impedância entre os casos intactos e danosos. Posteriormente, os Coeficientes de Correlação Cruzada (CCs) são usados como índice de dano predefinido. A RNA é construída e treinada usando todos os valores de CC em várias faixas de frequência como múltiplas entradas e a gravidade do dano real como a única saída para vários cenários de danos pré-selecionados, de modo que estimativas de danos subsequentes possam ser realizadas selecionando as faixas de frequência governantes de forma autônoma. O desempenho da abordagem proposta foi examinado por meio de uma série de estudos experimentais para detectar parafusos soltos e rachaduras induzidas em pontes de aço reais e estruturas de edifícios. Verificou-se que a abordagem proposta determina autonomamente as faixas de frequência sensíveis a danos e pode ser usada para avaliação eficaz da gravidade dos danos em uma ampla variedade de casos de danos em estruturas reais.

Lim *et al.* (2011) utilizou uma técnica de normalização de dados usando a Análise de Componentes Principais do Kernel (KPCA - do inglês *Kernel Principal Component Analysis*). Essa técnica é desenvolvida para melhorar a detecção de danos sob temperatura variável e condições de carregamento externo, afim de minimizar falsos positivos devido a essas variações. A técnica proposta é usada para detectar o afrouxamento do parafuso dentro de um terminal de encaixe de metal, que conecta uma asa de uma aeronave composta a uma fuselagem. Testes de modelo e em escala real são realizados sob variações realistas de temperatura e carga para validar a técnica proposta. A singularidade deste artigo reside em: (1) uma técnica de normalização de dados adaptada para detecção de danos baseada em impedância, (2) vários parâmetros ambientais, como temperatura e carregamento estático / dinâmico são considerados simultaneamente para a normalização de dados e (3) a eficácia da técnica proposta é examinada usando dados coletados de um espécime de asa composta em escala real com uma geometria complexa.

Min *et al.* (2012) propuseram uma ferramenta de análise de padrões baseada em RNA para identificar faixas de frequência sensíveis a danos de forma autônoma e para fornecer informações detalhadas, como o tipo e a gravidade do dano. A importância de selecionar a faixa de frequência ideal foi investigada experimentalmente usando uma viga de alumínio simplesmente suportada. O desempenho da abordagem baseada em RNA proposta foi vali-

dado por meio de identificações de danos de parafusos soltos e entalhes em uma viga de alumínio com junta de parafuso e uma estrutura de tubo em escala de laboratório. Finalmente, o algoritmo proposto com base em RNA foi incorporado a um nó de sensor de impedância sem fio para detectar danos reais em uma ponte em escala real. No geral, a abordagem proposta incorporando um nó de sensor de impedância sem fio foi usada para avaliar o tipo e a gravidade do dano em casos de danos múltiplos e de múltiplos tipos estruturais.

Palomino, Steffen e Finzi (2012) com a intenção de identificação, localização e classificação de dois tipos de danos, a saber, trinca e perda de rebite, utilizaram métodos de análise de agrupamento fuzzy. Os resultados mostram que os métodos de análise de agrupamento fuzzy são úteis para identificação, localização e classificação desses tipos de danos.

Na e Lee (2013) usaram uma placa de vidro-epóxi com tamanho de 200 mm × 200 m × 3,5 mm com 3 transdutores PZT fixados em cada um dos 3 cantos da placa, com o objetivo de prever a localização do dano utilizando o algoritmo PNN. Neste estudo, 6 áreas diferentes foram classificadas para teste e 24 furos danificados em 30 foram previstos corretamente, uma taxa de precisão de 80%. O estudo conclui com a afirmação de que aumentar o número de dados de treinamento pode melhorar os resultados.

Jun, Juan e Tang (2013) monitoraram o crescimento de uma trinca em uma viga de alumínio livre-livre, coletando assinaturas de 4 PZTs. O dano proposto no experimento situou-se a 80 mm da extremidade da viga e foi aplicado em 3 profundidades de trinca, 5 mm, 10 mm e 15 mm. Utilizaram uma arquitetura multicamadas de redes neurais artificiais (RNA) para classificação do dano.

Selva *et al.* (2013) realizaram experimentos sobre Placas Reforçadas com Fibra de Carbono (CFRPs) e utilizam a técnica de impedância eletromecânica para detecção de danos. Empregaram simulações numéricas baseadas no método dos elementos finitos realizadas de forma a simular mais de uma centena de cenários de danos. Afim de quantificar e detectar os danos gerados, utilizaram as métricas de dano. E para localização dos danos, utilizou-se das RNAs, cujas entradas são derivadas de uma PCA das métricas de dano. Obtiveram uma RNA que pode ser usada como uma ferramenta para prever a posição no plano de um único dano em uma placa laminada composta.

Sun, Sevillano e Perera (2015) empregaram experimentos em um corpo de prova com 31.3 cm de comprimento, 9.5 cm de largura e 7 cm de profundidade onde as dimensões da armadura externa são de 29 cm de comprimento, 5 cm de largura e 0,12 cm de espessura. Três atuadores-sensores PZTs foram colados na face externa do corpo sobre uma placa metálica usando um adesivo epóxi. O dano proposto foi um furo no concreto abaixo da placa metálica. Foram coletadas assinaturas da estrutura integra inicialmente e posteriormente sob a condição de dano e repetiu-se as coletas em dias e temperaturas diferentes, afim de verificar a interferência dessas condições de contorno. Foi utilizado *ensemble*, com algoritmo de enxame de partícula (PSO) e *bagging*, para classificar o dano proposto.

Nick *et al.* (2015) apresentam uma abordagem de monitoramento de integridade estrutural baseado em agentes, dos quais tem o papel de determinar quais técnicas de classificação e em quais combinações de avaliação serão avaliados os dados. Quanto as técnicas, foram empregados aprendizado não supervisionado para identificar a existência e a localização do dano, além do aprendizado supervisionado para identificar o tipo e a gravidade do dano. As técnicas de aprendizagem supervisionada investigadas são SVM, classificadores naive Bayes e Redes Neurais *feed-forward* (FNN). As técnicas de aprendizagem não supervisionadas investigadas são k-means (com k igual a 3, 4, 5 e 6) e mapas auto-organizáveis (SOM, com 3, 4, 5 e 6 neurônios de saída). Para cada técnica, exceto SOM, testaram versões com e sem PCA para reduzir a dimensionalidade dos dados. Constataram diferenças significativas nas características dessas técnicas de aprendizado de máquina, com relação entre precisão e tempo de execução, que podem ser exploradas pelos agentes para encontrar combinações adequadas de técnicas de classificação.

Djemana, Hrairi e Jeroudi (2017) utilizaram o algoritmo Máquina de Aprendizagem Extrema (ELM - do inglês *Extreme Learning Machine*) para estimar o local do dano usando dados de sensores piezoelétricos. O modelo é treinado em dados gerados por simulação e testado em experimentos para estimar a localização do dano usando dados de sensores piezoelétricos. Resultados experimentais mostram que o ELM pode ser usado como ferramenta para prever um único dano em estruturas. Uma precisão geral de 84,5% é alcançada com a melhor precisão de 95%.

Oh *et al.* (2017) utilizaram RNAs combinada com a técnica ISHM para prever a resistência do concreto pelo monitoramento das assinaturas de impedância por 28 dias durante a cura do concreto. Para o treinamento do algoritmo RNA, foram usados a razão água e cimento, tempo de cura, temperatura de cura, maturidade e valor de CC calculado a partir das assinaturas de impedância. O erro da previsão da cura do concreto foi desprezível.

Rautela, Vashisht e Bijudas (2018) realizaram experimentos baseados em impedância eletromecânica aplicando dois tipos de danos, ou seja, descolamento do transdutor e descolamento estrutural (um tipo de dano estrutural) em vigas. Três casos diferentes de desconexão do transdutor e desconexão estrutural junto com sua combinação produziram dezesseis casos, incluindo casos não danificados. O espectro de admitância (real e imaginário) é coletado para todos os casos usando experimentos em analisador de impedância e simulações em pacotes FEM comercialmente disponíveis (ANSYS APDL-16.5). RNAs com retro-propagação probabilística são usadas para detectar os descolamentos. Tanto a parte real quanto a imaginária da admitância EMI para todos os 16 casos de simulações e 10 casos de experimentos são usados como matriz de entrada, enquanto a quantidade de desconexões é usada como o vetor de resposta de saída. A RNA também é projetada para identificar a gravidade de ambos os descolamentos e, em seguida, o caso em que cai é previsto manualmente. Conclui-se que a RNA é capaz de detectar danos em problemas SHM baseados em

impedância eletromecânica.

Oliveira *et al.* (2018) propuseram uma estratégia para melhorar a detecção de danos estruturais identificados em assinaturas de impedância eletromecânica via: (1) filtro Savitzky – Golay (SG), usando tanto a primeira quanto a segunda derivadas; (2) Rede Neural Probabilística (PNN); e (3) Rede Neural ARTMAP Fuzzy Simplificada (SFAN). Esses três métodos foram empregados para analisar os dados EMI obtidos experimentalmente em uma placa de alumínio contendo três adesivos de PZT (Titanato de Zirconato de Chumbo). Como experimento, foram considerados os cenários de danos simulados pela fixação de uma pequena porca metálica em três posições diferentes na placa de alumínio. Descobrimos que o método proposto atinge uma taxa de acerto de mais de 83%, que é significativamente maior do que as abordagens atuais de última geração. Além disso, essa abordagem resulta em uma melhoria de 93% quando considerado o melhor cenário.

Rezende, Barella e Jr (2020) contribuíram realizando um experimento em um disco de freio, afim de identificar danos por adição de massa. Utilizaram a técnica de aprendizado de máquina não supervisionada agrupamento *K-Means* foi aplicada ao conjunto de dados e um modelo de regressão quadrática foi usado também com base na métrica de dano RMSD dos casos.

Junior (2020) empregou um Monitoramento da Condição de Ferramentas (TCM – *Tool Condition Monitoring*) no processo de retificação. Desenvolveu uma nova abordagem para o TCM na operação de dressagem por meio do método EMI-PZT em conjunto com RNA. As variáveis dos modelos usadas foram métricas RMSD e CCDM em subfaixas de frequências. Também foi utilizado um modelo de lógica fuzzy para localização do dano, o qual recebeu como entrada as informações de danos da ferramenta de dressagem extraídas dos índices de danos representativos RMSD e CCDM em várias subfaixas de frequência.

Castro, Baptista e Ciampa (2020) aplicaram metodologias para prever o dano estrutural em estruturas de concreto combinando efetivamente ciência de dados e estratégias de ML. Os resultados de testes experimentais disponíveis publicamente são usados, onde os testes foram realizados variando a rigidez e condições de massa, com a suposição de que essas fontes de variabilidade são representativas das mudanças nas condições operacionais e ambientais, além das mudanças causadas por danos. Para aumentar a precisão da detecção de danos, em vez da tradicional análise de série temporal, o aprendizado de máquina é usado para aprender com a experiência anterior. Para detectar a existência e localização do dano na estrutura, usaram o aprendizado supervisionado e, para medir a gravidade do dano, o aprendizado não supervisionado. Os resultados de precisão são obtidos com três algoritmos de aprendizado de máquina bem conhecidos: KNN-k (*Nearest Neighbor*), SVM e RFC *Random Forest Classifier*. Neste estudo, o algoritmo RFC gerou boas previsões em condições danificadas e não danificadas com boa precisão, quando comparado com o algoritmo KNN e o SVM no modo supervisionado de ML.

O resumo das contribuições encontradas relacionando aprendizado de máquina e monitoramento de integridade estrutural são apresentados pelo Quadro 3.8.

Autor	Estrutura	Foco	Pré-processamento	Modelo
Giurgiutiu e Kropas-Hughes (2003)	Disco de alumínio	Detectar 5 estados de dano	-	Rede neural probabilística (PNN)
Park *et al.* (2008b)	Duas vigas de alumínio	Detectar a perda ou afrouxamento de parafusos	Redução de dimensionalidade PCA	Clusterização K-means
Park *et al.* (2008a)	Trilho de ferro	Detectar 2 tipos de dano (Alma e cabeça do perfil)	-	Dois modelos de SVM, inicialmente para categorizar em dano ou sem dano as assinaturas e posteriormente classificar os danos
Min, Park e Yun (2010)	Pontes de aço e estruturas de edifícios	Seleção de faixa de frequência	Coeficiente de correlação entre assinaturas para cada subfaixa de frequência	Rede neural artificial (ANN)
Lim *et al.* (2011)	Terminal de encaixe de metal, que conecta uma asa de uma aeronave	Detectar danos sob temperatura variável e condições de carregamento externo	Técnica de normalização de dados usando a análise de componentes principais do Kernel (KPCA)	-
Min *et al.* (2012)	Viga de alumínio com junta de parafuso e uma estrutura de tubo	Identificar faixas de frequência sensíveis a danos, o tipo e a gravidade do dano	-	Rede neural artificial (ANN)
Palomino, Steffen e Finzi (2012)		Identificação, localização e classificação de danos	-	Análise de agrupamento fuzzy

Continua na próxima página

Autor	Estrutura	Foco	Pré-processamento	Modelo
Na e Lee (2013)	Placa de vidro-epóxi	Localizar dano	-	Rede neural probabilística (PNN)
Jun, Juan e Tang (2013)	Viga de alumínio	Monitoramento do crescimento de uma trinca		Rede neural artificial (ANN)
Selva *et al.* (2013)	Placas reforçadas com fibra de carbono	Quantificar e detectar danos	Redução de dimensionalidade PCA	Rede neural artificial (ANN)
Sun, Sevillano e Perera (2015)	Placa metálica sobre corpo de prova de concreto	Avaliar iterações de dano sob temperaturas diferentes	-	*ensemble* algoritmo de enxame de partícula (PSO) com *bagging* para classificar o dano proposto
Nick *et al.* (2015)		Localizar, tipo e a gravidade do dano	Redução de dimensionalidade PCA	Maquina de vetores de suporte (SVM), Classificador naive bayes e redes neurais feed-forward (FNN). K- mean e mapas auto-organizáveis (SOM)
Djemana, Hrairi e Jeroudi (2017)		Localizar danos	-	Aprendizagem extrema (ELM)
Oh *et al.* (2017)		Prever a resistência do concreto	Coeficiente de correlação entre assinaturas	Rede neural artificial (ANN)
Rautela, Vashisht e Bijudas (2018)	Vigas de alumínio	Detectar danos estruturais e descolamento do transdutor	-	Rede neural probabilística (PNN)

Continua na próxima página

Autor	Estrutura	Foco	Pré-processamento	Modelo
Oliveira *et al.* (2018)	Placa de alumínio	Detectar danos	Suavização das assinaturas com Savitzky – Golay	Rede neural probabilística (PNN) e Rede neurais ART-MAP Fuzzy Simplificada (SFAN)
Rezende, Barella e Jr (2020)	Disco de freio	Detectar danos	-	Clusterização K-means
Junior (2020)	Ferramenta de dressagem	Monitorar desgaste de ferramentas e localizar danos	-	Rede neural artificial (ANN)
Castro, Baptista e Ciampa (2020)	Estruturas de concreto	Localizar e detectar danos	-	k Nearest Neighbor (KNN), maquina de vetores de suporte (SVM) e floresta aleatória (RF).

Fonte – o autor.

Quadro 3.8 – Resumo das contribuições na área de aprendizado de máquina

Capítulo 4

EXPERIMENTO PARA DESENVOLVIMENTO DE MODELOS DE SHM

Neste Capítulo é apresentado o experimento utilizado para o desenvolvimento, análise e implantação do modelo. São apresentados as metodologias empregadas para a geração de danos e variações de temperatura. As alterações nas IS são destacadas, apresentando as implicações devido às variações de temperatura e massa.

4.1 Experimento com Vigas de Alumínio Usinadas

Para a condução do experimento foram utilizadas quatro estruturas formadas por vigas de alumínio como corpos de prova e uma câmara climática de controle e temperatura e umidade EPL-4H da série Platinous, conforme ilustra a Figura 4.1. Esta câmara está instalada no Laboratório de Mecânica de Estruturas (LMEst) da Faculdade de Engenharia Mecânica (FEMEC) na Universidade Federal de Uberlândia (UFU).

Neste novo experimento, foram utilizadas vigas de alumínio de 500 mm de comprimento, 38 mm de largura e 3,2 mm de espessura. Em cada uma delas foi colada uma pastilha PZT de 1 mm de espessura e 20 mm de diâmetro a 100 mm da borda da estrutura.

O processo de usinagem superficial foi utilizado para danificar uma das faces das vigas com 30 mm de largura, a 70 mm da extremidade oposta, na mesma face, que estava colada a pastilha PZT. Foram realizados nove níveis de dano no total, sendo dois de referência (*baselines*, teoricamente iguais) e sete níveis de remoção de espessura por usinagem. O corpo de prova 4 foi utilizado como mecanismo de controle, sendo removido da câmara junto aos outros, mas nenhum mecanismo de falha foi introduzido nele.

A Figura 4.2 apresenta os quatro corpos de prova ensaiados na condição de contorno

Figura 4.1 – Câmara de temperatura EPL-4H.

Fonte – o autor

bi-apoiados. Nota-se que as vigas usinadas perderiam espessura ao longo dos processos de medição, o que poderia contribuir flexionando as vigas em seus dois pontos extremos de apoio. Ao colocá-las apoiadas sob a espessura, e não sobre a largura, seu maior momento de inércia nesta direção impede ou minimiza tais impactos no resultado.

Figura 4.2 – 4 corpos de prova utilizados no ensaio.

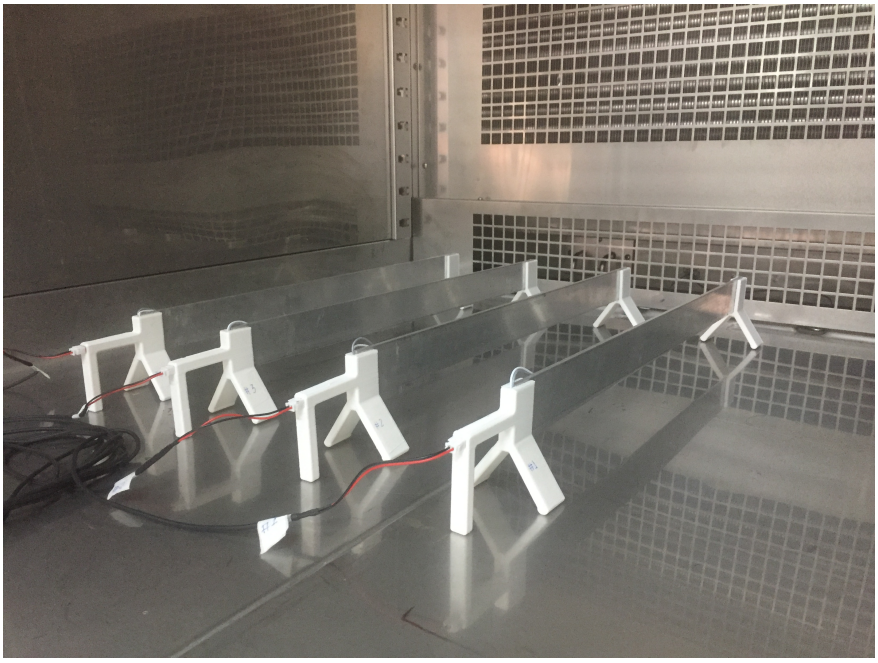
Fonte – o autor

É importante notar que para este ensaio foram projetados em impressão em plástico ABS os cavaletes para posicionamento das estruturas dentro da câmara. Os pés dos cavaletes são arredondados para diminuir a interferência do piso da câmara nas estruturas, o que

possibilitou o embutimento de conectores nos próprios corpos de prova. Assim, as estruturas eram removidas do interior da câmara para usinagem e retornavam para a câmara para continuidade das medições com menor interferência possível dos cabos, uma vez que eles mantinham a mesma posição dentro da câmara.

Este experimento também teve como objetivo provocar variações nas IS através da variação de temperatura, na qual a Figura 4.3 apresenta o posicionamento das estruturas na câmara. A temperatura foi reduzida em um passo de 3 °C e com 11 ciclos. O analisador de impedância esteve conectado ao servidor posicionado ao lado externo da câmara, adquirindo e armazenando os dados.

Figura 4.3 – Posicionamento dos corpos de prova no interior da câmara.



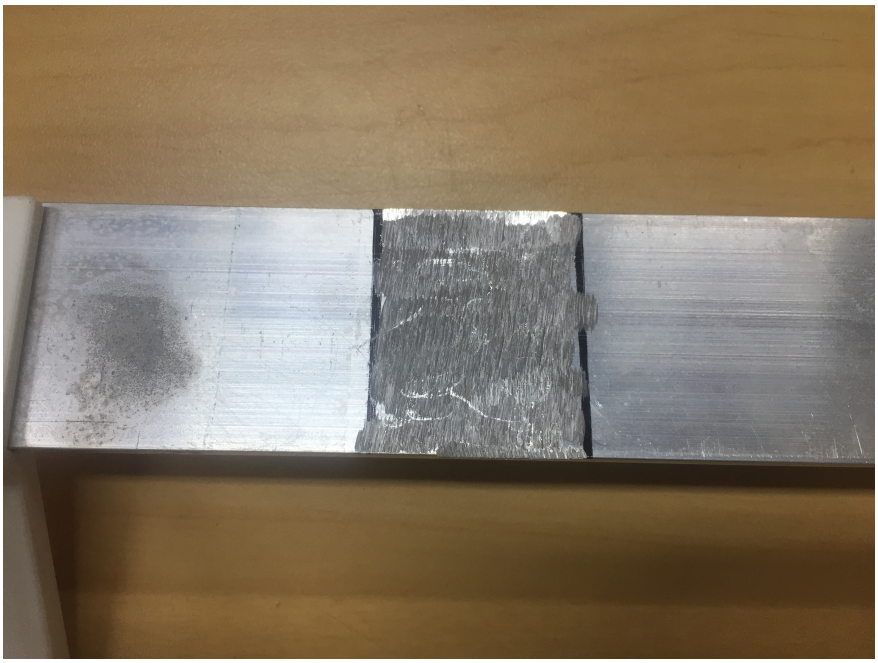
Fonte – o autor

A faixa de temperaturas considerada variou de forma crescente de 3 °C em 3 °C, abrangendo de 10 °C a 40 °C em 11 ciclos. Por ser uma variação de temperatura pequena, após cada término de coletas das IS a câmara elevava sua temperatura e a mantinha por 30 minutos para estabilização da nova temperatura (equilíbrio térmico dos corpos de prova e do ambiente) antes do início do novo ciclo de coletas. Para cada configuração: corpo de prova, temperatura e condição de dano foram realizadas 30 amostragens de assinaturas. Assim, no total de 11 ciclos de temperatura, 9 níveis de dano, 4 corpos de prova e 30 repetições, para cada corpo de prova foram coletadas 2970 assinaturas, totalizando 11880.

Considerando que o processo de inserção de falhas foi realizado manualmente através de uma esmeriladeira, optou-se por adotar duas variáveis de resposta de cada corpo de prova após a inserção do dano. A primeira resposta é a massa, considerando a perda de massa do estado anterior. Para esta resposta foi utilizada uma balança de precisão de duas

casas decimais de grama e foram realizadas oito medições para obtenção da média. A segunda resposta é a espessura na região delimitada para usinagem, considerando a perda de espessura em relação ao estado anterior. Para esta segunda resposta foi utilizado um micrômetro com resolução de 0.01mm e foram registradas a média de 10 medições aleatórias da espessura da área delimitada pela usinagem. Uma imagem de um dos estados usinados é ilustrada pela Figura 4.4, que apresenta a distância do ponto do cavalete e a região usinada.

Figura 4.4 – Dano 2 no corpo de prova 1.



Fonte – o autor

Os resultados das medições de massa e espessuras para cada corpo de prova são apresentados, respectivamente nas Tabelas 4.1 e 4.2.

Tabela 4.1 – Medições das massas (gramas) dos corpos de prova.

Condição	**Corpo de Prova 1**	**Corpo de Prova 2**	**Corpo de Prova 3**	**Corpo de Prova 4**
Baseline	191,36	192,42	192,86	195,89
Dano 1	191,20	192,27	192,69	-
Dano 2	191,16	192,02	192,58	-
Dano 3	190,95	191,82	192,39	-
Dano 4	190,41	191,48	192,02	-
Dano 5	189,96	190,54	191,33	-
Dano 6	189,16	189,42	189,78	-
Dano 7	187,44	188,57	188,37	195,83

Fonte – o autor.

Tabela 4.2 – Medições das espessuras (mm) dos corpos de prova.

Condição	**Corpo de Prova 1**	**Corpo de Prova 2**	**Corpo de Prova 3**	**Corpo de Prova 4**
Baseline	3,17	3,17	3,19	3,18
Dano 1	3,09	3,11	3,09	-
Dano 2	3,01	2,96	2,99	-
Dano 3	2,97	2,93	2,95	-
Dano 4	2,85	2,85	2,89	3,17
Dano 5	2,72	2,49	2,63	-
Dano 6	2,44	2,23	2,09	-
Dano 7	1,93	1,92	1,83	3,18

Fonte – o autor

Conforme pode ser observado em ambas as medições, os danos inseridos nos três corpos de prova jamais excederam 50% da espessura e os dois últimos níveis de usinagem aplicados foram mais severos para observar esse maior crescimento (progressões menores e maiores) na perda de massa através dos modelos que foram desenvolvidos e analisados.

O intervalo de frequência empregado na aquisição das assinaturas de impedância foi de 30k Hz a 70k Hz, com passo de 10 Hz, gerando um total de 4000 pontos frequenciais. Vale ressaltar ainda que não foi empregada a seleção de faixa de frequência devido a alta capacidade dos métodos utilizados em identificar padrões. As IS do corpo de prova 1 para cada nível de integridade aplicado são apresentadas pela Figura 4.5, onde todas as assinaturas pertencem ao grupo de temperatura de 30°C.

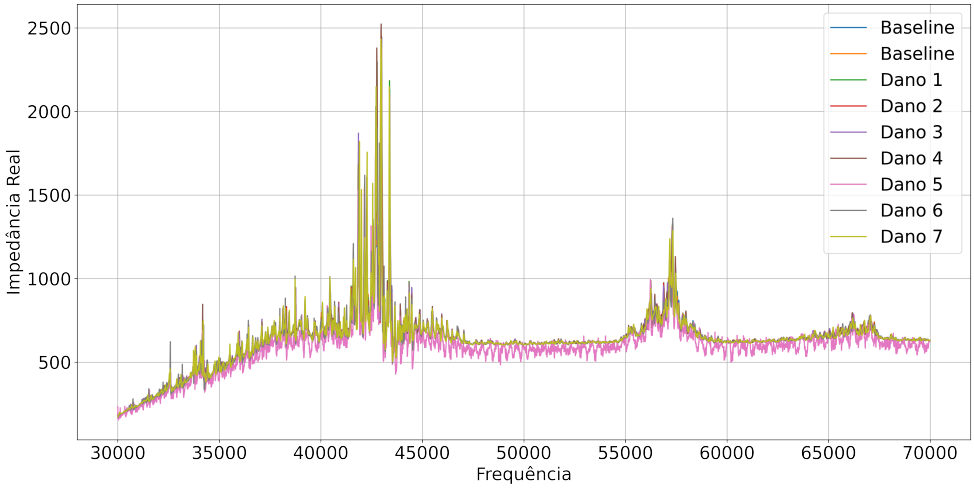
Na Figura 4.5 é possível observar a proximidade entre as assinaturas de níveis de integridade diferentes, destacando assim o grau de complexidade da tarefa de classificação de danos.

Para o nível de integridade *baseline*, a variação de temperatura imposta provocou deslocamentos verticais e horizontais já esperados, seguindo os conceitos apresentados na Seção 2.2.3. Essas variações são apresentadas pela Figura 4.6.

Para a compensação dessa variação apresentada pela Figura 4.6 é comum aplicar métodos de compensação de temperatura, dos quais transladam horizontalmente e verticalmente as assinaturas buscando minimizar a variação imposta pela oscilação da temperatura. Os métodos aqui utilizados não necessitam dessa compensação uma vez que possuem uma alta capacidade de identificar padrões.

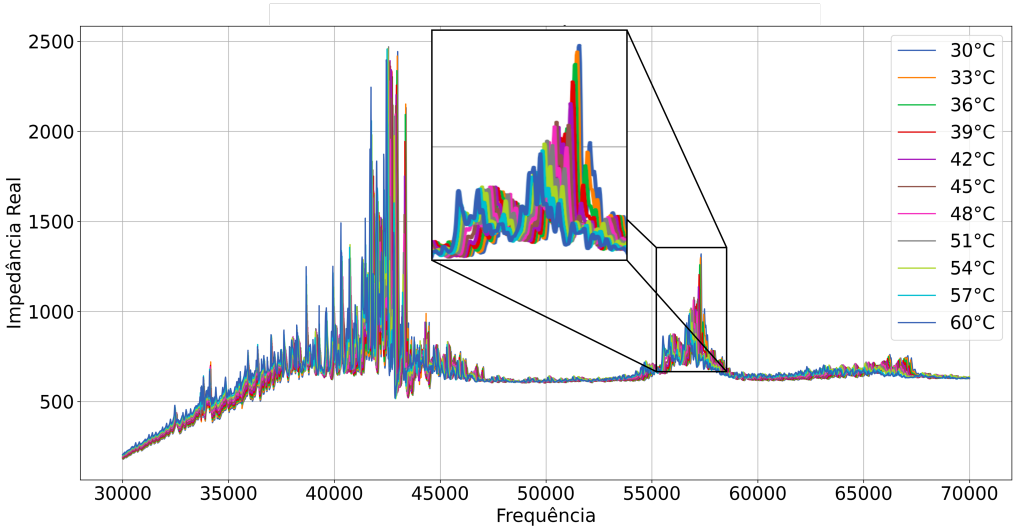
Os experimentos foram conduzidos sobre os dados dos corpos de prova 1, 2 e 3. Porém, os resultados aqui apresentados são referentes ao corpo de prova 1.

Figura 4.5 – Assinaturas de impedância do corpo de prova 1 para cada nível de dano proposto.



Fonte – o autor

Figura 4.6 – Variação da temperatura nas assinaturas de impedância para o baseline do corpo de prova 1.



Fonte – o autor

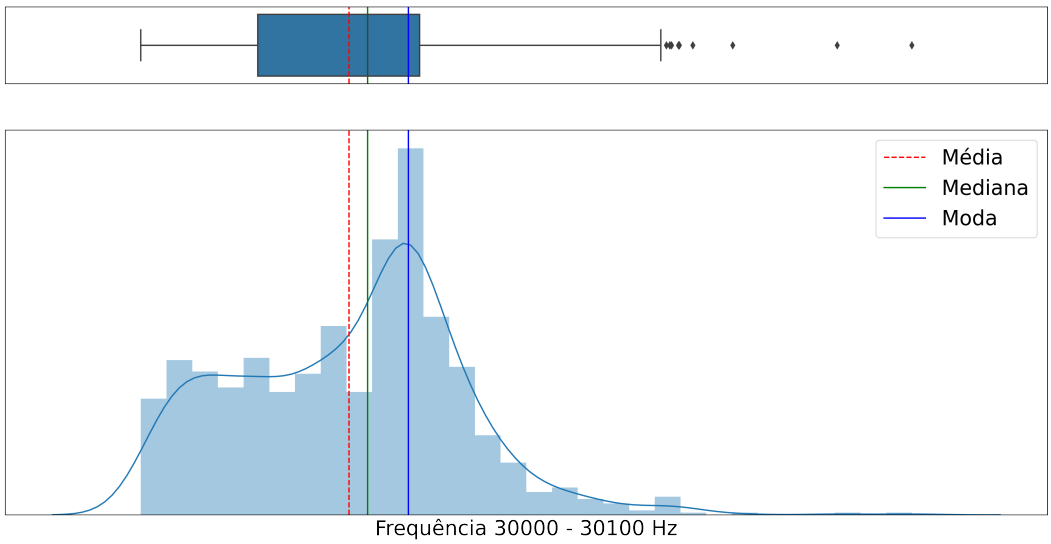
No intuito de delimitar o melhor modelo para identificar os níveis de severidade dos danos impostos nos sistemas mecânicos, inicialmente deve-se identificar o tipo e as dimensões dos dados de entrada que serão utilizados nos modelos a serem avaliados. Assinaturas de impedância eletromecânica são vetores bidimensionais, onde esse estão no domínio da frequência. Tal fenômeno pode ser modelado por meio de dois parâmetros: frequência e parte real da impedância, que podem ser passados como recursos para os modelos. No entanto, se considerarmos um *range* de frequência padronizado à todas as amostras avaliadas, uma outra abordagem unidimensional poderá ainda ser empregada, utilizando-se para isso

apenas os valores da impedância medida no transdutor.

O procedimento para a criação do modelo consiste na preparação dos dados para alimentar os modelos, modelagem (avaliação de diversos modelos e busca pelos melhores hiperparâmetros) e finalização do modelo (junção de toda a base de dados para treinamento do modelo final). Para a etapa de modelagem foi usada a biblioteca do Pycaret que possibilita automatizar o processo de seleção de modelos, busca de hiperparâmetros análise de desempenho do modelo e finalização do modelo. Além dessa etapa, também foi analisado como e quanto cada faixa de frequência impacta nas classificações do modelo, ou seja, como o modelo interpreta os dados de entrada. Afim de exemplificar a entrega de um modelo de identificação de falhas, é implantado um sistema web com a biblioteca Streamlit. Essa aplicação é disponibilizada utilizando os recursos de conteinerização, e computação em nuvem.

Inicialmente foi feita uma redução na complexidade dos dados a serem analisados, de modo a reduzir a quantidade de pontos frequenciais de 4000 para 400. A aplicação desse pre-processamento teve a intenção de reduzir a complexidade dos modelos a serem analisados. Para aplicar essa redução utilizou-se a mediana dos 100 pontos frequenciais. A escolha dessa medida de tendência central se deu pois, a distribuição não é simétrica, ou seja, os valores discrepantes na cauda puxam a média do centro para a cauda mais longa. A Figura 4.7 apresenta essa assimetria para a faixa de frequência de 31000 a 31100 Hz.

Figura 4.7 – Distribuição de frequência para a faixa de 31000 a 31100 Hz.



Fonte – o autor

A Figura 4.7 apresenta a distribuição assimétrica para a faixa de frequência de 31000 a 31100 Hz o qual foi reduzido para 1 ponto utilizando a mediana desses valores da parte real da impedância.

Capítulo 5

DETECÇÃO DE ANOMALIAS NO SHM

Neste Capítulo são apresentados os desenvolvimentos dos modelos de classificação, focados na detecção de anomalias supervisionados, não supervisionado e semi-supervisionado, onde para cada abordagem é avaliado e analisado os resultados. Na abordagem supervisionada, a qual atingiu os melhores resultados, foi realizada a interpretação das variáveis do modelo de modo a apresentar seus impactos, direções e associações nas respostas.

5.1 Construção dos Modelos de Detecção de Anomalias

A detecção de anomalias pode ser empregada de forma não supervisionada, semi-supervisionada e supervisionada. Para a forma não supervisionada é possível identificar padrões não conhecidos e utilizar estruturas de dados desbalanceados, ou identificar que a maioria das instâncias no conjunto de dados são normais, procurando instâncias que pareçam se ajustar menos ao restante do conjunto de dados. Para o emprego de métodos semi-supervisionados é utilizada a abordagem de detecção de novidade onde se conhece os dados sem anomalia, os quais são usados para criar modelos que detectam qualquer alteração que desvie desse padrão. E para a forma supervisionada é comum se conhecer os dados normais e anômalos, os quais são usados para criar modelos de classificação, onde é aplicado um problema de classificação binária separando dados normais dos anormais. A principal diferença do modelo de detecção de anomalias para muitos outros problemas de classificação estatística, é a natureza desequilibrada, inerente da detecção de valores discrepantes (DUNNING; FRIEDMAN, 2014; CHANDOLA; BANERJEE; KUMAR, 2009).

Foram avaliadas as 3 abordagens de detecção de anomalia para o corpo de prova 1. A abordagem que obteve os melhores resultados foi utilizada para uma interpretação das classificações realizadas através da biblioteca SHAP.

Os dados para o desenvolvimento das abordagens foram separados em dois conjuntos, um de treinamento/validação e outro de teste, simulando a entrada de novas assinaturas de impedância (IS) nunca vistas pelo modelo. Dessa forma, para o conjunto de testes foi selecionado aleatoriamente 10% da quantidade total dos dados.

Para o treinamento semi-supervisionado são passados apenas os dados referentes a classe de dano 0 (*baseline*), resultando num total de 601 IS. O restante das IS foram inseridas no conjunto de teste.

Para o treinamento não supervisionado foram utilizados os dados referentes a classe de dano 0, (601 IS), além de 30 IS aleatórias de cada uma das seguintes classes de dano: 1, 3, 5, e 7 (120 IS). Dessa forma, totalizando 721 IS de treinamento. Essa separação foi necessária pois os métodos não supervisionados trabalham com dados desbalanceados, com isso o treinamento foi realizado em um total de 17% de anomalias.

Para o treinamento supervisionado, os conjuntos de treino/validação e teste foram classificados em duas classes: 0 se dano (*baseline*) e 1 com dano.

5.1.1 Modelagem Semi-supervisionada para Detecção de Anomalias

Inicialmente foi desenvolvido um modelo semi-supervisionado utilizando a biblioteca Sklearn, no qual optou-se por utilizar o método não supervisionado *Local Outlier Factor* (LOF). A escolha deste método para a abordagem semi supervisionada ocorreu por ele, por padrão, possuir o parâmetro *novelty* que permite trabalhar como um método semi-supervisionado voltado a detecção de novidade.

Para a modelagem os dados não recebem rótulos, porém na etapa de treino é conhecida a porção que se deseja que o modelo abstraia. Essa porção é então submetida ao treinamento que, para o problema considerado, a porção utilizada são os *baselines*, os quais foram utilizados na intenção de identificar danos a partir das IS do corpo de prova 1.

Para se inicializar a modelagem do problema para a abordagem semi-supervisionada, deve-se inicialmente importar a biblioteca Sklearn, que possui o método LOF, e então instanciar um objeto dessa classe com os parâmetros numero de vizinhos, (*n_neighbors*), considerados no agrupamento. No problema abordado foram utilizados 10 vizinhos, e foi utilizado 15% de taxa de contaminação dos dados com o intuito de deixar o classificador bem criterioso na detecção de anomalias. Por fim, o parâmetro *novelty* foi definido como verdadeiro, indicando que se está trabalhando com detecção de novidade.

Com o modelo treinado pode-se então avaliar a capacidade de detecção de anomalias. A documentação do Sklearn não recomenda avaliar os dados de treinamento no modelo, pois isso levaria a resultados errados. As pontuações de anormalidade das amostras de treinamento estão sempre acessíveis por meio do atributo *negative_outlier_factor*. De forma geral o modelo obteve bons resultados, que podem ser evidenciados pela Figura 5.1.

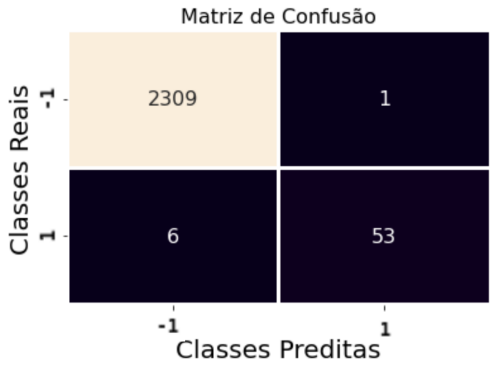
Figura 5.1 – Resultado do modelo LOF para tarefa semi-supervisionada de detecção de anomalia.

	precision	recall	f1-score	support
-1	0.997408	0.999567	0.998486	2310
1	0.981481	0.898305	0.938053	59
accuracy	0.997045	0.997045	0.997045	0
macro avg	0.989445	0.948936	0.968270	2369
weighted avg	0.997012	0.997045	0.996981	2369

Fonte – o autor

Através da Figura 5.1 pode-se verificar a precisão do modelo para a classe anômala -1 (corpo de prova com dano), a qual obteve uma alta precisão, indicando uma boa acertabilidade. Para os resultados da classe normal 1 (*baseline*) obteve uma precisão menor, indicando a existência de falsos positivos. Foi identificado uma alta taxa de detecção da classe anômala -1 indicando que, de todas IS referentes a um estado danoso, o método teve uma alta eficiência em identificá-los. Já para para a classe normal 1, a taxa de detecção foi menor, indicando que nem todos os *baselines* foram identificados. De forma geral, o modelo apresentou uma acurácia elevada. Os resultado da avaliação do modelo sobre os dados não vistos são apresentados pela Figura 5.1.

Figura 5.2 – Matriz de confusão do modelo LOF para tarefa semi-supervisionada de detecção de anomalia.



Fonte – o autor

A Figura 5.2 apresenta os acertos e erros do modelo desenvolvido. Em um total de 2369 IS de teste, 59 são normais 1 (*baseline*) e 2310 são classe anômala (corpo de prova com dano). Para a classe das IS normais 1 o modelo errou apenas 6 IS, classificando-as como anômalas (danosas), representando 6 falsos positivos. Já para a classe das IS anômalas (danosas) o modelo errou apenas 1 IS, ou seja, classificou como uma IS normal (*baseline*), porém a mesma era um dano. Ao verificar o erro cometido pelo modelo, identificou-se que a IS pertence ao nível de dano 1, o qual possui IS semelhantes as do *baseline*.

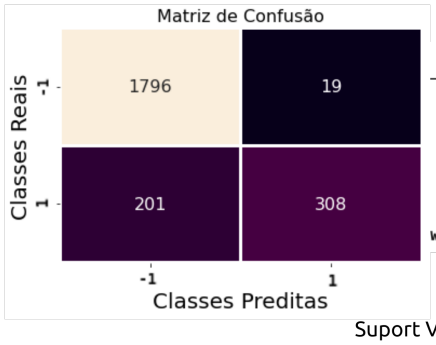
5.1.2 Modelagem Não Supervisionada para Detecção de Anomalias

Para o desenvolvimento da modelagem não supervisionada, utilizou-se a biblioteca Pycaret com os métodos e classes para detecção de anomalias com o objetivo de avaliar 3 modelos: *Isolation forest* (IF), *Support Vector Machine* (SVM) e *Local Outlier Factor* (LOF). Para se treinar os modelos foi importada a biblioteca Pycaret com os pacotes e classes de detecção de anomalia. Em seguida, foram instanciados e configurados os parâmetros: base de dados, tipo de dados, limpeza e preparação de dados, amostragem de dados, proporção de separação dos dados de treino e validação e a semente para os modelos (garantindo a reprodutividade dos resultados). Dentre esses parâmetros optou-se por utilizar os valores padrões da biblioteca, indicando apenas a base de dados, proporção de separação dos dados de treino/validação e a semente para os modelos.

Para o treinamento foram criados 3 objetos para cada tipo de modelo e então selecionada uma taxa de contaminação de 50% para os modelos IF e SVM, enquanto que para o LOF, foi selecionada uma taxa de 30%. Os valores dos parâmetros restantes foram configurados com os valores padrões para cada método. A taxa de contaminação foi selecionada na intenção de elevar o critério de detecção de anomalia para os modelos, visando obter os melhores resultados. Os resultados para cada modelo são apresentados na Figura 5.3.

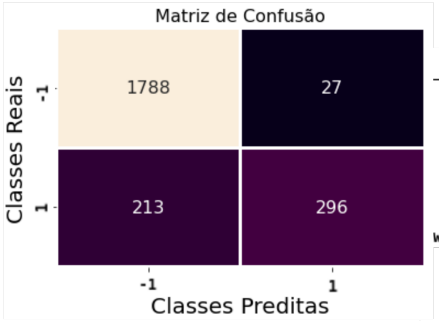
Figura 5.3 – Resultados da classificação dos modelos para tarefa não supervisionada de detecção de anomalia.

Isolation Forest



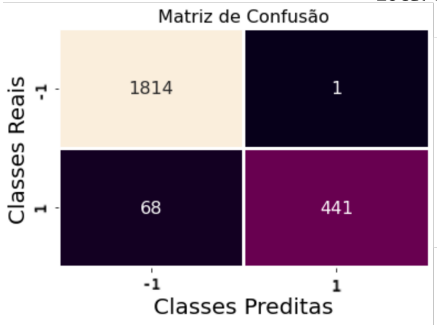
	precision	recall	f1-score	support
-1	0.899349	0.989532	0.942288	1815
1	0.941896	0.605108	0.736842	509
accuracy	0.905336	0.905336	0.905336	0
macro avg	0.920623	0.797320	0.839565	2324
weighted avg	0.908668	0.905336	0.897291	2324

Suport Vector Machine



	precision	recall	f1-score	support
-1	0.893553	0.985124	0.937107	1815
1	0.916409	0.581532	0.711538	509
accuracy	0.896730	0.896730	0.896730	0
macro avg	0.904981	0.783328	0.824323	2324
weighted avg	0.898559	0.896730	0.887703	2324

Local Outlier Factor



	precision	recall	f1-score	support
-1	0.963868	0.999449	0.981336	1815
1	0.997738	0.866405	0.927445	509
accuracy	0.970310	0.970310	0.970310	0
macro avg	0.980803	0.932927	0.954391	2324
weighted avg	0.971286	0.970310	0.969533	2324

Fonte – o autor

Pode-se observar que o modelo *Local Outlier Factor* (LOF) obteve os melhores resultados para todas as métricas. Vale ressaltar que foi classificado apenas 1 IS como não anômala quando na verdade não era. Essa IS classificada errada pertence a classe de dano nível 1, a qual possui uma alta proximidade entre os grupos *baseline* (não anômalo) e dano nível 1. Outra característica importante é o *recall* o qual indica a taxa de detecção. Para o grupo das IS não anômalas (*baselines*) obteve 86% indicando um aumento na quantidade de *baselines* não detectados, os quais são evidenciados pela matriz de confusão (68 IS não anômalas classificadas como anômalas).

5.1.3 Modelagem Supervisionada para Detecção de Anomalias

Na classificação supervisionada foi utilizada a biblioteca Pycaret com os métodos e classes para classificação convencional. Para esse tipo de classificação, as classes são definidas em 0 para ausência de dano (*baseline*) e 1 para existência de dano. Em seguida, foi

instanciado um objeto para configurar os parâmetros: base de dados, nome da coluna alvo, tipo de dados, limpeza e preparação de dados, amostragem de dados, proporção de separação dos dados de treino e validação e a semente para os modelos. Dentre esses parâmetros pode-se optar por utilizar os valores padrões da biblioteca, indicando apenas a base de dados, nome da coluna alvo, proporção de separação dos dados de treino e validação e a semente para os modelos.

Em seguida é executado o método para o treinamento de diversos modelos de classificação. Apesar de ser possível eliminar ou selecionar alguns modelos nesta iteração, todos foram utilizados. Também pode-se ordenar os resultados por alguma métrica em específico, selecionar a quantidade dos melhores modelos a serem retornados pelo método e a quantidade de grupos a serem avaliados na validação cruzada. Para o corpo de prova 1, o melhor modelo encontrado foi o *Logistic Regression*, pois atingiu o maior nível de *recall* e em seguida de precisão. A Figura 5.4 apresenta os resultados para todos os modelos avaliados.

Figura 5.4 – Comparação dos modelos para tarefa de detecção de anomalia supervisionado.

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
lr	Logistic Regression	0.9977	0.9979	0.9970	1.0000	0.9985	0.9936	0.9937	1.4820
et	Extra Trees Classifier	0.9971	0.9999	0.9985	0.9978	0.9981	0.9919	0.9920	0.2180
ridge	Ridge Classifier	0.9966	0.0000	0.9955	1.0000	0.9977	0.9905	0.9906	0.1360
lda	Linear Discriminant Analysis	0.9966	0.9978	0.9955	1.0000	0.9977	0.9905	0.9906	0.2580
rf	Random Forest Classifier	0.9965	1.0000	0.9992	0.9963	0.9978	0.9902	0.9903	0.4090
catboost	CatBoost Classifier	0.9965	0.9999	0.9978	0.9978	0.9978	0.9902	0.9903	39.9190
ada	Ada Boost Classifier	0.9960	0.9975	0.9993	0.9956	0.9974	0.9886	0.9887	1.1890
lightgbm	Light Gradient Boosting Machine	0.9954	0.9998	0.9963	0.9978	0.9970	0.9871	0.9871	2.4810
gbc	Gradient Boosting Classifier	0.9942	0.9999	0.9985	0.9941	0.9963	0.9836	0.9838	5.6670
xgboost	Extreme Gradient Boosting	0.9937	0.9999	0.9963	0.9956	0.9959	0.9820	0.9822	1.4440
dt	Decision Tree Classifier	0.9868	0.9818	0.9910	0.9918	0.9914	0.9627	0.9630	0.1260
knn	K Neighbors Classifier	0.9816	0.9894	0.9865	0.9895	0.9880	0.9483	0.9485	0.1020
qda	Quadratic Discriminant Analysis	0.9344	0.8579	1.0000	0.9217	0.9592	0.7932	0.8116	0.0970
svm	SVM - Linear Kernel	0.7946	0.0000	0.9006	0.8670	0.8637	0.3155	0.3748	0.0670
nb	Naive Bayes	0.2579	0.6873	0.0494	0.7776	0.0909	0.0011	0.0024	0.0440

Fonte – o autor

O critério utilizado para selecionar o melhor modelo se baseia tanto na precisão do modelo (capacidade de detectar as classes de dano), como no *recall* (taxa de detecção de nível de dano). Para um modelo de classificação de danos é importante possuir uma alta taxa de detecção, uma vez que alguns sistemas são críticos, ou seja, caso falhem, geram grandes perdas financeiras ou lidam diretamente com vida das pessoas. Dessa forma, é de suma importância os modelos detectarem todas as possíveis alterações estruturais, assim como ter uma boa precisão.

Após a seleção do melhor modelo é importante encontrar os seus melhores hiperparâmetros. Para isso é executada a função que realiza essa busca. Na busca dos hiperparâ-

metros foi utilizada uma validação cruzada com 10 grupos de validação. Os resultados são apresentados pela Figura 5.5.

Figura 5.5 – Resultados para cada um dos 10 grupos da validação cruzada da busca dos hiperparâmetros na detecção de danos.

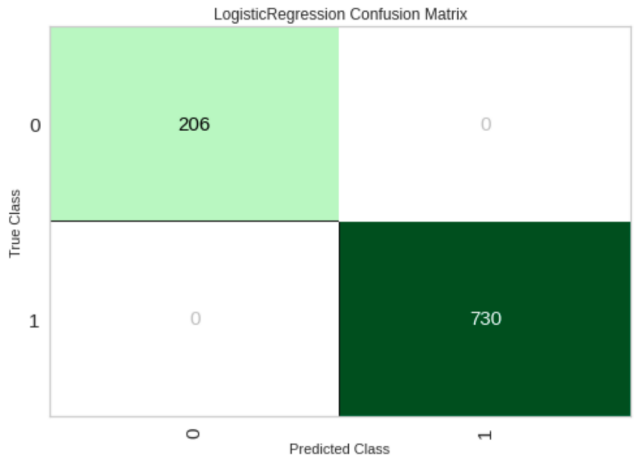
	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
1	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
2	0.9885	0.9925	0.9851	1.0000	0.9925	0.9681	0.9686
3	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
4	0.9943	0.9925	0.9925	1.0000	0.9963	0.9839	0.9840
5	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
6	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
7	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
8	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
9	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Mean	0.9983	0.9985	0.9978	1.0000	0.9989	0.9952	0.9953
SD	0.0037	0.0030	0.0048	0.0000	0.0024	0.0102	0.0101

Fonte – o autor

Através da Figura 5.5 pode-se verificar que não houve uma melhoria expressiva na performance do modelo com a busca dos hiperparâmetros. Isso ocorreu pois o modelo obteve um ótimo resultado com os hipérmetros padrões, que são evidenciados pela Figura 5.4.

Explorando os resultados dos modelos através da matriz de confusão apresentada pela Figura 5.6, pode-se observar a alta capacidade de detecção de danos para o corpo de prova 1. Para os dados de validação, o modelo não obteve nenhum erro de classificação.

Figura 5.6 – Matriz de confusão para o modelo de detecção de danos.



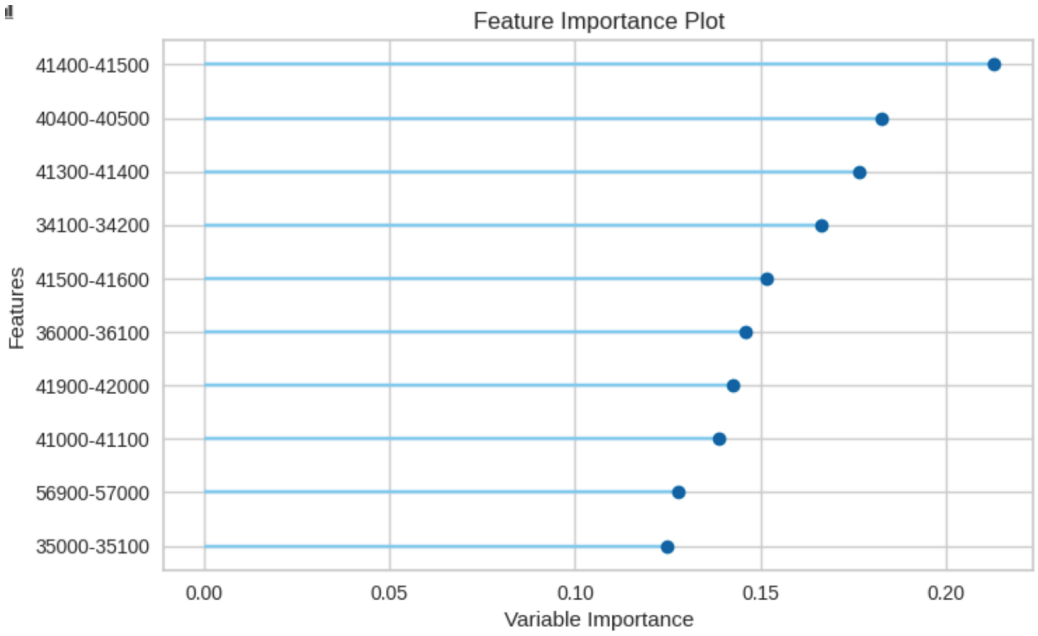
Fonte – o autor

Os resultados apresentados pela matriz de confusão da Figura 5.6 destacam a capacidade do modelo em identificar danos. Vale ressaltar que o modelo detectou todas as assinaturas para todas as classes, evidenciando assim sua alta taxa de detecção (*recall*), a qual

representa a consistência do modelo, uma vez que o mesmo pode ser aplicado a sistemas críticos.

Outro fator a considerar durante a modelagem é a importância das variáveis que o modelo utiliza. Essa análise é útil tanto para eliminar variáveis que não impactam na predição, como também caso seja necessário criar novas variáveis de modo a melhorar a capacidade de detecção do modelo. A Figura 5.7 apresenta as faixas de frequência mais importantes para o modelo.

Figura 5.7 – Variáveis mais importantes para o modelo de classificação supervisionada de detecção de danos.



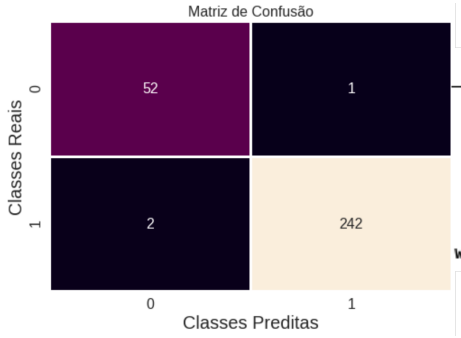
Fonte – o autor

Por fim, para simular a entrada de dados novos para testar a detecção do modelo, são então utilizados os 10% reservados para essa simulação. Para o emprego da simulação de novos dados, afim de estimar a capacidade de generalização do modelo o mesmo é finalizado utilizando o método *finalise_model*, o qual retreina o modelo com os dados de treino e validação. Os resultados dessa avaliação são apresentados pela Figura 5.8.

Pode-se verificar a alta capacidade de detecção de danos apresentados pelos resultados da Figura 5.8, que apresentou alta capacidade de detecção. Errou apenas 1 IS *baseline* classificada como um dano (falso positivo) e apenas 2 classificada como íntegro quando na verdade era um dano de nível 1.

Para se entender e justificar as tomadas de decisão realizadas pelo modelo ou na intenção de eliminar variáveis para melhorar a capacidade de detecção do modelo, é importante avaliar e interpretar quais as variáveis são consideras para as detecções, bem como as formas com que elas se relacionam umas com as outras. Para essa interpretação, foi selecionado o modelo que obteve os melhores resultados sob o conjunto de teste (dados não

Figura 5.8 – Resultados da classificação supervisionada para detecção de danos sobre dados de simulação.



	precision	recall	f1-score	support
0	0.962963	0.981132	0.971963	53
1	0.995885	0.991803	0.993840	244
accuracy	0.989899	0.989899	0.989899	0
macro avg	0.979424	0.986468	0.982901	297
weighted avg	0.990010	0.989899	0.989936	297

Fonte – o autor

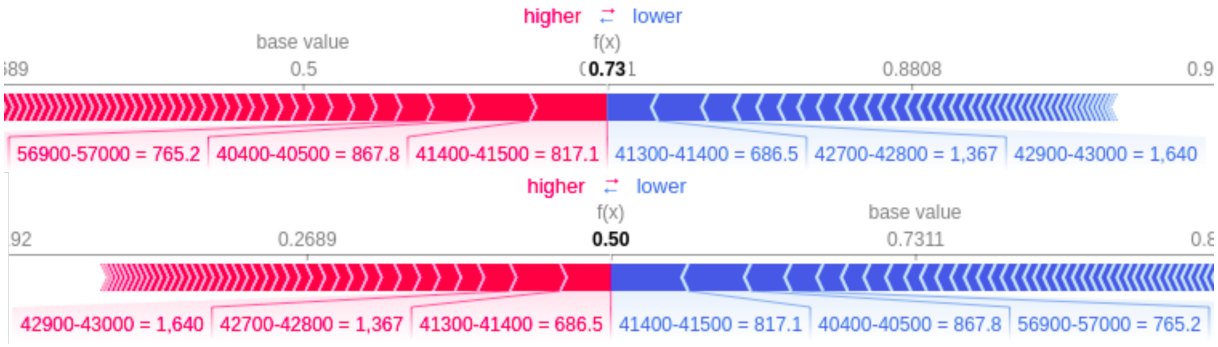
vistos pelo modelo). Dessa forma a abordagem supervisionada para o desenvolvimento do modelo de classificação binária foi a que obteve os melhores resultados.

5.1.4 Interpretando o Modelo Supervisionado

Com o modelo já definido e avaliado é importante interpretar como são feitas as classificações na intenção de justificar decisões, monitorar a qualidade do sinal, identificar algum viés nos dados ou no modelo, selecionar faixas de frequência mais apropriadas e obter mais informações para melhorar a capacidade de classificação de danos. Para isso foi empregada a biblioteca SHAP apresentada na Seção 3.6.2.3.

Afim de identificar as variáveis mais importantes para a classificação de uma IS pode-se utilizar o método *force_plot*, o qual apresenta as "forças"que guiam a classificação. Foi selecionado aleatoriamente uma IS que pertence a classe de *baseline* e então aplicou-se o método para inspecionar o impacto das variáveis nessa classificação. Como o problema de classificação de dano é uma tarefa de classificação binária, são apresentadas 2 figuras contendo as "forças"de predição para cada classe. A Figura 5.9 representa o impacto das faixas de frequência mais importantes da IS de impedância para a detecção de dano.

Figura 5.9 – Influência das faixas de frequência na detecção de dano.



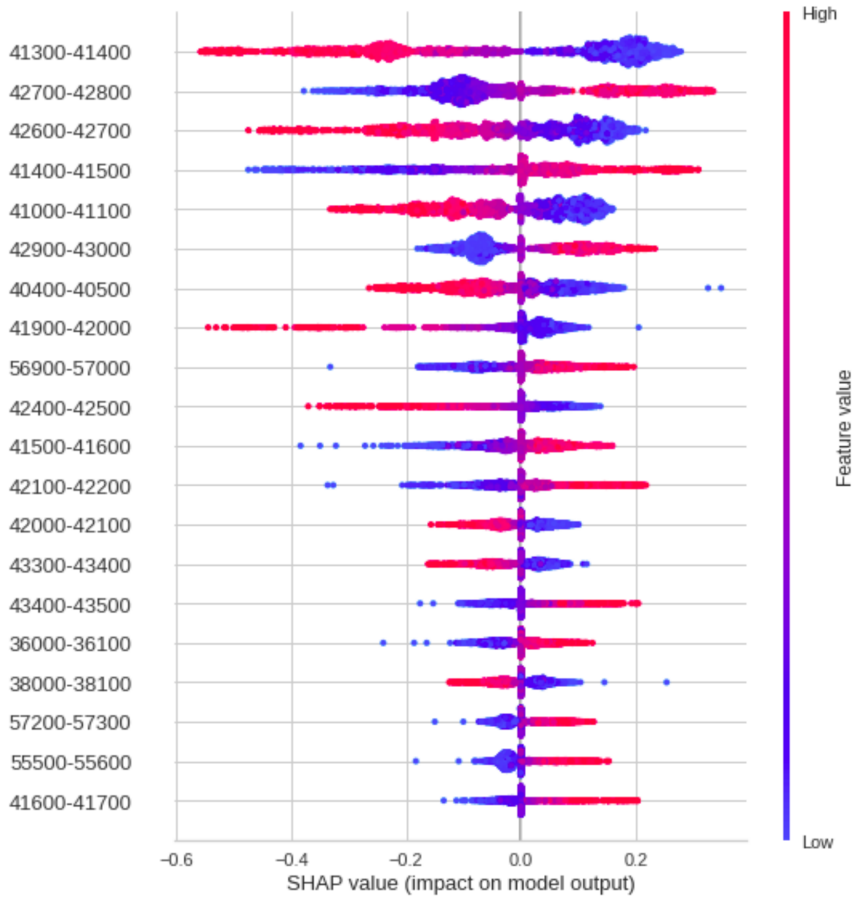
Fonte – o autor

De acordo com a Figura 5.9, cada gráfico de força apresenta uma probabilidade mé-

dia denominada valor base, a qual pode ser interpretada como uma previsão caso nenhum recurso fosse passado para o modelo. As variáveis tem o papel de "empurrar"esse valor para a probabilidade predita. De forma macro, para essa IS selecionada, as faixas de frequência de 40400 Hz a 57000 Hz tiveram um impacto significativo, no deslocamento dos valores base de probabilidade.

Uma análise macro das variáveis do modelo é dada pelo método *summary_plot*, o qual apresenta uma comparação dos valores das variáveis indicadas pela cor, bem como a forma (diminuindo ou aumentando a probabilidade de pertencer a classe em análise) como essas variáveis impactam no modelo. A Figura 5.10, apresenta o gráfico de resumo dos impactos das faixas de frequência considerando a magnitude desses valores para a classe 1 (com dano).

Figura 5.10 – Gráfico de resumo de impacto para as faixas de frequência na detecção de danos da classe 1.



Fonte – o autor

De acordo com a Figura 5.10 é possível identificar no eixo x o quanto cada variável de entrada apresentada no eixo y, juntamente com a magnitude do seu valor apresentado pela coloração, é responsável por uma variação. É importante notar o tamanho e a separabilidade para cada faixa de frequência, indicando uma boa interpretação por parte do modelo. Essa separabilidade observada nos valores SHAP no gráfico indicam que dado um parâmetro de

entrada, este desloca o valor base em uma direção baseado na sua magnitude.

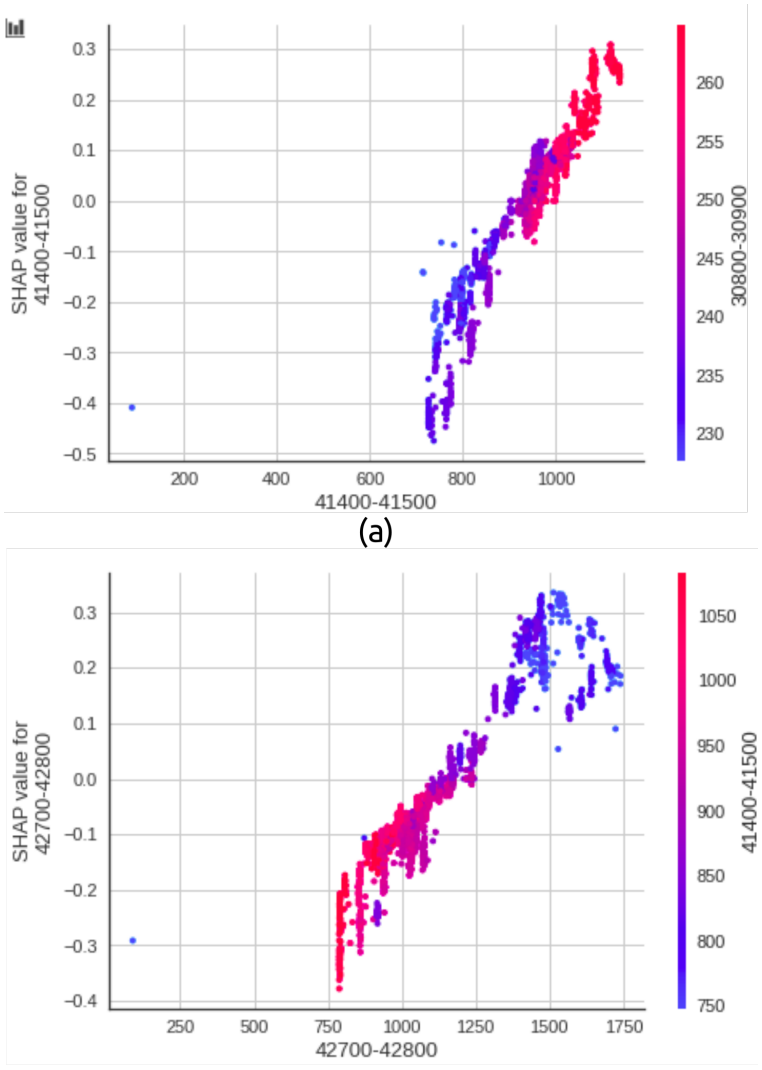
Uma forma de interpretar a Figura 5.10 seria para a faixa de frequência de 41300 Hz a 41400 Hz, onde valores altos impactam empurrando a estimativa para ser classificada como não pertencente a essa classe (danosa). Já para valores menores, o inverso é verdadeiro, ou seja, impactam aumentando a estimativa de pertencer a essa classe, ou seja, aumenta a estimativa de pertencer a classe 0 (*baseline*).

Para se avaliar as relações dos impactos no modelo entre as variáveis, utiliza-se o gráfico de dependência através do método *dependence_plot*, o qual recebe como parâmetro uma ou 2 faixas de frequência para apresentar as iterações. A análise consiste em avaliar o impacto de uma faixa de frequência em relação a outra, identificando correlações úteis tanto para melhoria do modelo, quanto para justificar resultados das classificações. A Figura 5.11 apresenta 2 comparações: (a) 41400 Hz a 41500 Hz com 30800 Hz a 30900 Hz e (b) 42700 Hz a 42800 com 41400 Hz a 41500 Hz.

Conforme pode ser observado pela Figura 5.11, existe uma correlação positiva em (a) em que ao aumentar os valores de impedância da faixa de frequência 41400 Hz a 41500 Hz, e também aumentar os valores de impedância da faixa de 30800 IS a 30900 Hz, tende a aumentar a probabilidade das instância serem da classe 1 (IS danosa). Já para (b), a medida que os valores de impedância da faixa de frequência de 42700 Hz a 42800 Hz aumentam e os valores de frequência da faixa de 41400 a 41500 Hz diminuem, aumenta a probabilidade das instância pertencerem a classe 1. Diversos tipos de análises análogas podem ser realizadas com o uso destes gráficos interativos no ambiente Python para compreensão da relação entre as variáveis.

Concluindo, nesta seção foram apresentadas 3 abordagens para detecção de anomalias voltadas para a detecção de dano, utilizando as bibliotecas Pycaret e Sklearn. Dessas, foi selecionada a abordagem supervisionada, a qual obteve os melhores resultados. Posteriormente, na intenção de interpretar o modelo treinado, foi utilizada a biblioteca SHAP. Foram apresentados os critérios de escolha e decisão do modelo baseado em diversas métricas. Por fim, foram apresentados alguns gráficos específicos do SHAP que são muito informativos para cada condição de resposta no cenário de um modelo de detecção de danos e os impactos e direções na resposta, além de suas associações/correlações. Com isto, acredita-se que o ferramental apresentado seja suficiente na seleção e construção de excelentes modelos de ML para representação de dados e sistemas, assim como compreensão dos principais parâmetros de entrada do modelo e suas combinações em relação a cada classe de resposta.

Figura 5.11 – Gráfico de dependência entre faixas de frequência para a detecção de dano da classe 1.



Fonte – o autor

Capítulo 6

MULTI CLASSIFICAÇÃO DE DANOS NO SHM

Neste Capítulo é apresentado a construção de um modelo de multi-classificação supervisionado, o qual tem como objetivo prever a qual classe uma dada IS pertence, dentre oito classes de dados. Nessa abordagem é avaliado e analisado os resultados para o posterior emprego da interpretação das variáveis do modelo. As interpretações se referem aos impactos, direções e associações/correlações nas respostas do modelo.

6.1 Construção do Modelo Multi classificação

Para se inicializar a modelagem do problema, deve-se inicialmente importar a biblioteca Pycaret considerando o tipo de tarefa a ser realizada, nesse caso, classificação. Subsequentemente, deve-se instanciar e configurar os parâmetros base de dados, nome da coluna alvo, tipo de dados, limpeza e preparação de dados, amostragem de dados, proporção de separação dos dados de treino e validação e a semente para os modelos. Dentre esses parâmetros, pode-se optar por utilizar os valores padrões da biblioteca indicando apenas a base de dados, nome da coluna alvo, proporção de separação dos dados de treino e validação e a semente para os modelos. A separação da base de dados seguiu o padrão de 90% para treino/validação e 10% aleatórios para teste.

Em seguida, é executado o método para o treinamento de diversos modelos de classificação. Para isso, pode-se excluir algum modelo da iteração, ordenar os resultados por alguma métrica em específico, selecionar a quantidade dos melhores modelos a serem retornados pelo método e a quantidade de grupos a serem avaliados na validação cruzada. Para o corpo de prova 1, o melhor modelo encontrado foi o *Linear Discriminant Analysis*, pois atingiu o maior nível de *recall* e em seguida de precisão, sem contar que é um dos modelos mais ágeis na etapa de treinamento. A Figura 6.1 apresenta os resultados para todos os modelos avaliados (linhas), as respectivas métricas utilizadas (demais colunas) e o tempo

de treino em segundos (TT).

Figura 6.1 – Comparação dos modelos para tarefa de multi classificação.

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
lda	Linear Discriminant Analysis	0.9971	0.9986	0.9968	0.9973	0.9971	0.9967	0.9967	0.1450
ridge	Ridge Classifier	0.9942	0.0000	0.9940	0.9945	0.9942	0.9933	0.9934	0.0420
rf	Random Forest Classifier	0.9942	0.9997	0.9935	0.9945	0.9942	0.9933	0.9934	0.4420
et	Extra Trees Classifier	0.9942	0.9999	0.9935	0.9945	0.9942	0.9933	0.9934	0.2230
lightgbm	Light Gradient Boosting Machine	0.9931	0.9999	0.9926	0.9935	0.9930	0.9920	0.9921	12.9000
xgboost	Extreme Gradient Boosting	0.9931	0.9999	0.9922	0.9934	0.9930	0.9920	0.9921	5.2830
catboost	CatBoost Classifier	0.9931	0.9997	0.9922	0.9934	0.9930	0.9920	0.9921	435.3900
lr	Logistic Regression	0.9890	0.9968	0.9885	0.9896	0.9890	0.9873	0.9874	4.1450
gbc	Gradient Boosting Classifier	0.9844	0.9998	0.9825	0.9853	0.9844	0.9819	0.9821	29.4410
knn	K Neighbors Classifier	0.9695	0.9902	0.9706	0.9705	0.9695	0.9646	0.9648	0.0690
qda	Quadratic Discriminant Analysis	0.9551	0.9748	0.9629	0.9626	0.9562	0.9481	0.9491	0.0900
dt	Decision Tree Classifier	0.9626	0.9784	0.9623	0.9642	0.9625	0.9566	0.9569	0.1210
nb	Naive Bayes	0.4156	0.8103	0.4631	0.5100	0.3726	0.3405	0.3659	0.0340
svm	SVM - Linear Kernel	0.4617	0.0000	0.4238	0.4154	0.3621	0.3635	0.4088	0.2610
ada	Ada Boost Classifier	0.3788	0.7486	0.2975	0.2568	0.2725	0.2446	0.2815	0.8400

Fonte – o autor

O critério utilizado para selecionar o melhor modelo se baseia tanto na precisão do modelo (capacidade de detectar as classes de dano), como no *recall* (taxa de detecção de nível de dano). É importante que um modelo para classificação de danos tenha uma alta taxa de detecção, pois alguns sistemas são críticos, ou seja, caso falhem, geram grandes perdas financeiras ou lidam diretamente com vida de pessoas. Dessa forma, é de suma importância os modelos detectarem todas as possíveis alterações estruturais, assim como terem uma boa precisão.

Após a seleção do melhor modelo, é importante encontrar seus melhores hiperparâmetros. Para isso, é executada a função para realizar essa busca. Para a busca dos hiperparâmetros foi utilizada uma validação cruzada com 10 grupos de validação. Os resultados são apresentados pela Figura 6.2.

Na Figura 6.2 pode-se verificar que não houve uma melhoria na performance do modelo com a busca dos hiperparâmetros. Isso ocorreu porque o modelo obteve um ótimo resultado com os hiperparâmetros padrões e a quantidade de iterações para cada grupo foi limitada a 10.

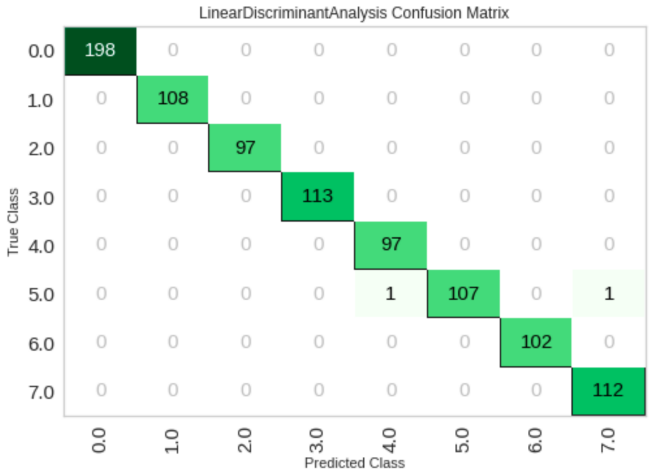
Explorando os resultados dos modelos através da matriz de confusão apresentada pela Figura 6.3, pode-se observar a alta capacidade de classificar os danos para o corpo de prova 1. Embora para a classe de dano 5 ocorreram dois erros de classificação (classificando a integridade da IS de dano nível 5 como sendo das classes 4 e 7), esse erro de classificação aconteceu pela proximidade entre as IS.

Figura 6.2 – Resultados para cada um dos 10 grupos da validação cruzada da busca dos hiperparâmetros para tarefa de multi classificação.

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
1	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
2	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
3	0.9943	0.9968	0.9934	0.9946	0.9943	0.9933	0.9934
4	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
5	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
6	0.9943	0.9968	0.9934	0.9946	0.9943	0.9933	0.9934
7	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
8	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
9	0.9827	0.9927	0.9812	0.9834	0.9825	0.9799	0.9801
Mean	0.9971	0.9986	0.9968	0.9973	0.9971	0.9967	0.9967
SD	0.0053	0.0023	0.0058	0.0051	0.0054	0.0062	0.0061

Fonte – o autor

Figura 6.3 – Matriz de confusão para a porção de treino do modelo de multi classificação de danos.



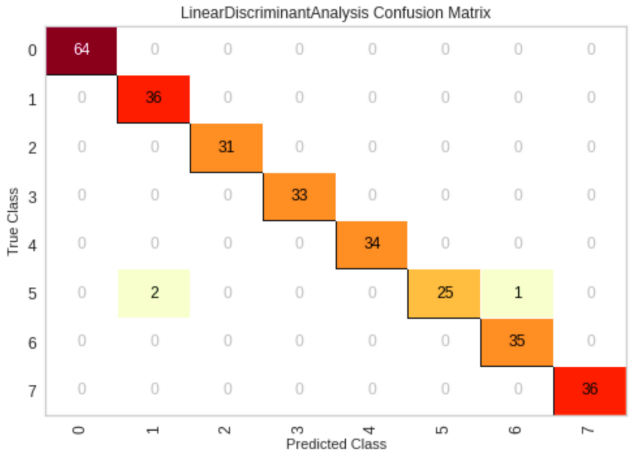
Fonte – o autor

Os resultados apresentados pela matriz de confusão da Figura 6.3 destacam a capacidade do modelo em identificar danos. Vale ressaltar que, para as classes que representam um dano estrutural, o modelo não foi capaz de detectar apenas 2 IS, evidenciando assim sua alta taxa de detecção (*recall*). Essa alta taxa representa a consistência do modelo, uma vez que o mesmo pode ser aplicado a sistemas críticos.

Finalmente, para simular a predição do modelo sobre dados novos, o mesmo é avaliado sobre a porção de teste. Antes de proceder essa avaliação o modelo é então finalizado com através do método *finalize_model*, o qual realiza o treinamento do modelo sobre as porções de treino e validação. A Figura 6.4 apresenta a matriz de confusão com os resultados da simulação.

Os resultados pela matriz de confusão da Figura 6.4 evidencia a alta capacidade do

Figura 6.4 – Matriz de confusão para a porção de teste do modelo de multi classificação de danos.

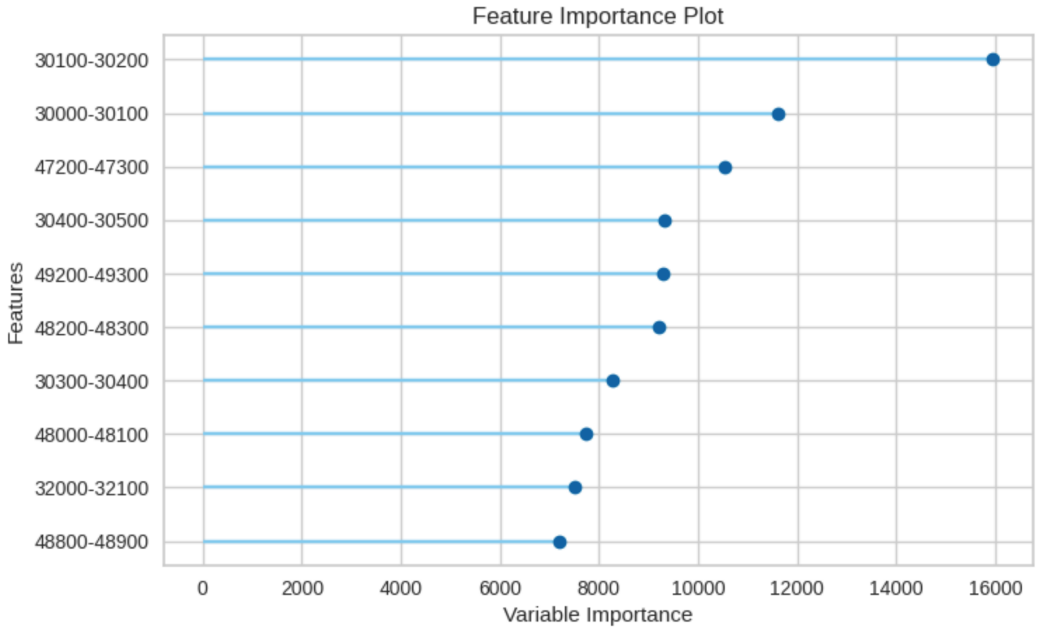


Fonte – o autor

modelo em identificar danos, embora para a classe de dano 5 o modelo, assim como na porção de treinamento, realizou 3 classificações erradas (2 para a classe de dano 1 e 1 para a classe de dano 6). Para contornar esse problema uma possível solução seria a coleta de mais assinaturas para a classe de dano 5.

Outro fator a considerar durante a modelagem é a importância das variáveis que o modelo utiliza. Essa análise é útil tanto para eliminar variáveis que não impactam na predição, como também caso seja necessário criar novas variáveis de modo a melhorar a capacidade de classificação do modelo. A Figura 6.5 apresenta as faixas de frequência mais importantes para o modelo.

Figura 6.5 – Variáveis mais importantes para o modelo.



Fonte – o autor

Os algoritmos de aprendizado de máquina linear se ajustam a um modelo em que a previsão é a soma ponderada dos valores de entrada. Todos esses algoritmos encontram um conjunto de coeficientes a serem usados na soma ponderada para fazer uma previsão. Esses coeficientes podem ser usados diretamente como um tipo bruto de pontuação de importância das variáveis. Porém, essas pontuações não consideram a magnitude dos valores das variáveis, gerando assim uma abordagem enviesada dos recursos mais importantes. Uma abordagem para superar esse viés é a utilização de métodos baseados em permutação de recursos, sendo ELI5 e SHAP as bibliotecas mais utilizadas para isso.

6.2 Interpretação do Modelo Multi Classificação

Com o modelo já definido e avaliado é importante interpretar como são feitas as classificações na intenção de justificar decisões, monitorar a qualidade do sinal, identificar algum viés nos dados ou no modelo, selecionar faixar de frequência mais apropriadas e obter mais informações para melhorar a capacidade de classificação de danos. Para isso foi empregada a biblioteca SHAP apresentada na Seção 3.6.2.3.

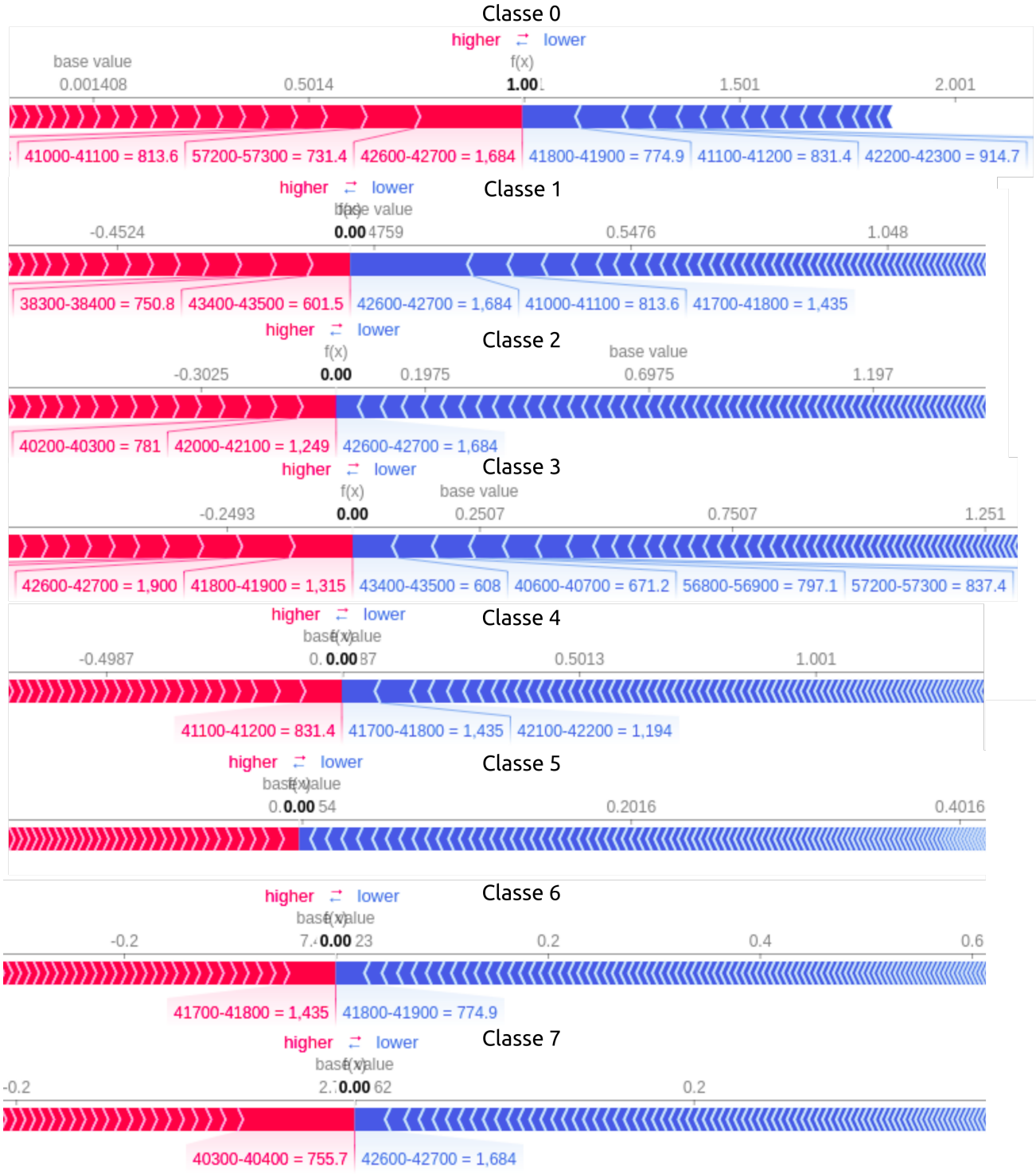
Afim de identificar as variáveis mais importantes para a classificação de uma IS, pode-se utilizar o método *force_plot*, que apresenta as "forças"que guiam a classificação. Foi selecionada aleatoriamente uma IS que pertence a classe de dano 0 (*baseline*) e então aplicou-se o método para inspecionar o impacto das variáveis nessa classificação. Como o problema de classificação de dano é uma tarefa multi classificação, existem 8 figuras apresentando as "forças"de predição para cada classe. A Figura 6.6 representa o impacto dos termos mais importantes da IS para a classificação de cada nível de dano.

De acordo com a Figura 6.6, cada gráfico de força apresenta uma probabilidade média denominada valor base, no qual as variáveis "empurram"esse valor para a probabilidade predita. De forma macro, para a IS selecionada as faixas de frequência até 42700 Hz tiveram impacto negativo significativo, diminuindo os valores base de probabilidade. Analisando os resultados para a classe 0 pode-se identificar uma alta probabilidade, que pode ser traduzida numa alta confiança do modelo de que a IS em análise pertence a essa classe. Além disso, as faixas importantes para essa classificação foram de 41000 a 57300 Hz das quais se interceptam com as faixas das outras classes, o que explica a alta confiabilidade do modelo.

Uma análise macro das variáveis do modelo é dada pelo método *summary_plot*, que apresenta uma comparação dos valores das variáveis indicados pela cor bem como a forma (diminuindo ou aumentando a probabilidade de pertencer a classe em análise) e como essas impactam no modelo. A Figura 6.7, apresenta o gráfico de resumo dos impactos das faixas de frequência considerando a magnitude desses valores para a classe 0.

De acordo com a Figura 6.7, é possível identificar no eixo x o quanto cada variável de entrada apresentada no eixo y juntamente com a magnitude do seu valor apresentado pela

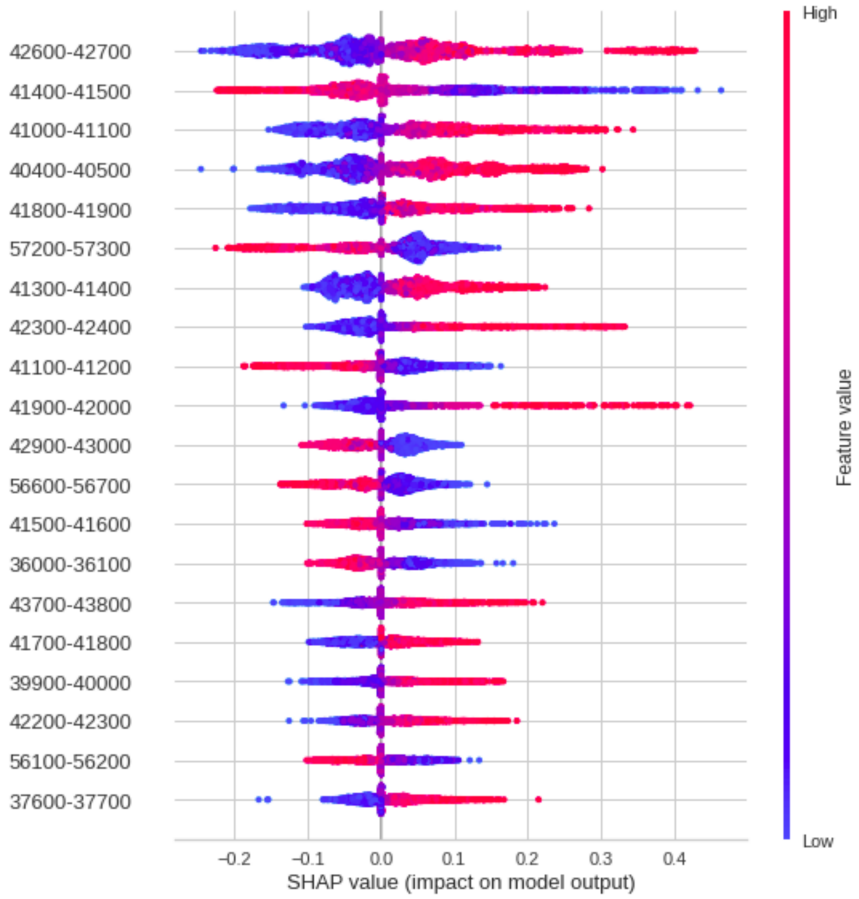
Figura 6.6 – Influência dos parâmetros de entrada na classificação de dano.



Fonte – o autor

coloração é responsável por uma variação. É importante notar o quão longa cada faixa de frequência é referente ao valor SHAP neste gráfico, indicando que muitas vezes um parâmetro de entrada afeta praticamente em uma direção o valor de resposta. Essa assimetria também indica o fato da alta probabilidade apresentada na Figura 6.6, pois as faixas de frequên-

Figura 6.7 – Gráfico de resumo para a classificação de dano da classe 0.



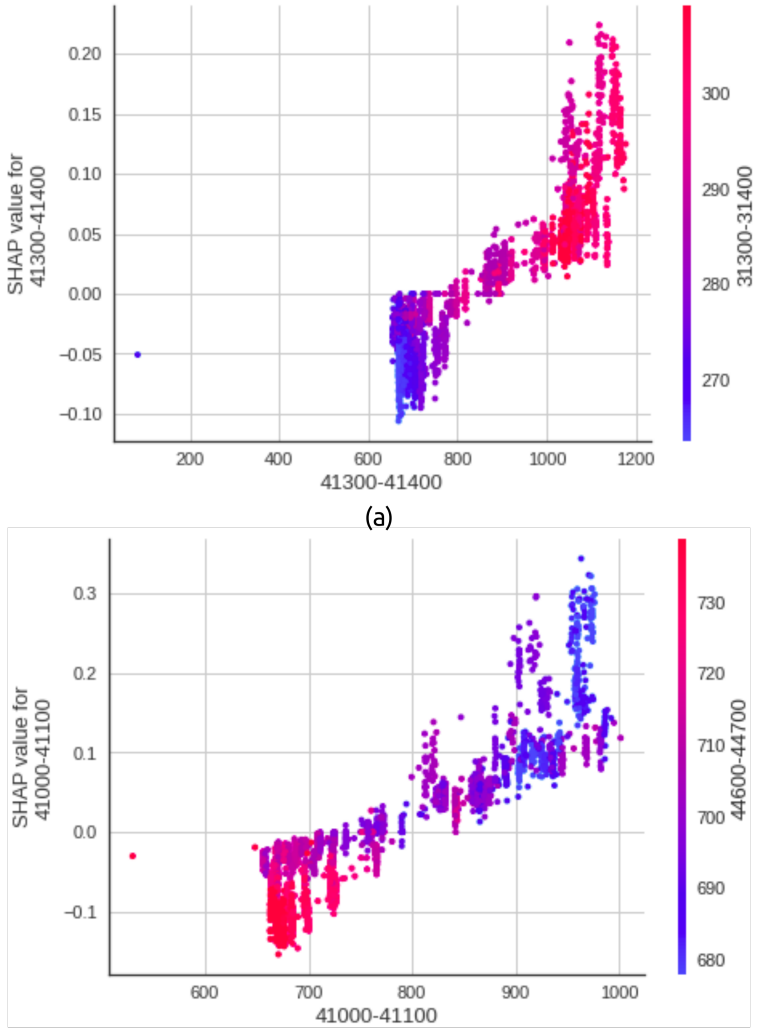
Fonte – o autor

cia mais importantes na classificação estão bem deslocadas, impactando em um drástico aumento na probabilidade. Outra análise poderia ser algum viés por parte do modelo se esse padrão não fosse replicado nas outras classes.

Para se avaliar as relações dos impactos no modelo entre as variáveis, utiliza-se o gráfico de dependência através do método *dependence_plot*, o qual recebe como parâmetro uma ou 2 faixas de frequência para apresentar as iterações. A análise consiste em avaliar o impacto de uma faixa de frequência em relação a outra, identificando correlações úteis tanto para melhoria do modelo, quanto para justificar resultados das classificações. A Figura 6.8 apresenta 2 comparações: (a) 41300 Hz a 41400 Hz com 31300 Hz a 31400 Hz e (b) 41000 Hz a 41100 Hz com 44600 Hz a 44700 Hz.

Conforme pode ser observado pela Figura 6.8, existem alguns pontos fora do padrão que prejudicam a análise do todo. No entanto, é possível verificar uma correlação positiva para o caso (a), em que ao aumentar os valores de impedância da faixa de frequência 41300 Hz a 41400 Hz e também aumentar os valores de impedância da faixa de 31300 Hz a 31400 Hz, tende a aumentar a probabilidade das instância pertencerem a classe 0. Já para (b), a medida que os valores de impedância da faixa de frequência de 41000 Hz a 41100 Hz au-

Figura 6.8 – Gráfico de dependência entre faixas de frequência para a classificação de dano da classe 0.



Fonte – o autor

mentam e os valores de frequência da faixa de 44600 Hz a 44700 Hz diminuem, impactam aumentando a probabilidade das instância pertencerem a classe 0. Diversos tipos de análises análogas podem ser realizadas com o uso destes gráficos interativos no ambiente Python para compreensão da relação entre as variáveis.

Concluindo, nesta seção foi apresentado o uso da biblioteca Pycaret e SHAP para identificação do melhor modelo de aprendizado de máquina a ser utilizado. Foram apresentados os critérios de escolha e decisão do modelo baseado em diversas métricas além do tempo de execução do treinamento. Por fim, foram apresentados alguns gráficos específicos do SHAP que são muito informativos para cada condição de resposta no cenário de um modelo de multi classificação e os impactos e direções na resposta, além de suas associações/correlações. Com isto, acredita-se que o ferramental apresentado seja suficiente na seleção e construção de excelentes modelos de aprendizado de máquina para representação de dados e sistemas, assim como compreensão dos principais parâmetros de entrada do

modelo e suas combinações em relação a cada classe de resposta.

Capítulo 7

MONITORAMENTO DE FALHAS VIA REGRESSÃO NO SHM

Neste Capítulo é apresentado a construção de um modelo de regressão para prever a massa do corpo de prova 1, sujeito a variações nas IS, causadas por danos e variações térmicas. Nessa abordagem é avaliado e analisado os resultados. Por fim, é realizada a interpretação das variáveis do modelo, identificando como cada uma delas impactam nas predições.

7.1 Construção do Modelo de Regressão

Uma perspectiva para avaliar alguma degradação do sistema em análise é relacionar as alterações das propriedades físicas com o as IS de impedância eletromecânicas, de modo a identificar e extrapolar alterações para estados futuros. Para o desenvolvimento dessa abordagem, foi desenvolvido um modelo de regressão visando prever a massa do sistema devido a uma progressão de dano.

A tarefa de regressão é supervisionada, na qual cada IS é referente a um valor de massa utilizado para treinar e avaliar os modelos. O desenvolvimento da modelagem é realizado sobre o corpo de prova 1 do qual possui variações de massa para cada um dos 8 níveis de dano aplicados. A divisão dos dados segue o padrão de treino/validação e teste, sendo que a divisão foi feita buscando identificar a capacidade de extrapolação de variação de massa do modelo para casos não treinados.

Para os dados de treino/validação foram utilizados 5 estados de dano. Já para os dados de teste que simulam a existência de dados novos nunca vistos pelo modelo, foram selecionados 10% aleatoriamente da base de dados do corpo de prova 1 e os outros 3 estados de danos não passados para o modelo.

Para a construção do modelo foi utilizada a biblioteca Pycaret, que possibilita avaliar diversos modelos de forma automática. Para interpretar e avaliar o melhor modelo foi uti-

lizada a biblioteca SHAP. Inicialmente é importada a biblioteca Pycaret com os métodos de regressão, em seguida é instanciado um objeto que realiza a configuração da base de dados treino/validação, se necessário, os parâmetros para executar preprocessamentos.

Para selecionar o melhor modelo é então executada a função de comparação de modelos. Dessa forma pode-se optar por excluir algum modelo da iteração, ordenar os resultados por alguma métrica em especifico, selecionar a quantidade dos melhores modelos a serem retornados pelo método e a quantidade de grupos a serem avaliados na validação cruzada. Para o corpo de prova 1, o melhor modelo encontrado para a tarefa de regressão foi o *Extra Trees Regressor*. Esse modelo atingiu os menores níveis de erro para as métricas MAE, MSE, RMSE e RMSLE, sem contar que é um dos modelos mais ágeis na etapa de treinamento. A Figura 7.1 apresenta os resultados para todos os modelos avaliados.

Figura 7.1 – Comparação dos modelos para tarefa de regressão.

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
et	Extra Trees Regressor	0.0093	0.0020	0.0412	0.9989	0.0002	0.0000	0.3920
catboost	CatBoost Regressor	0.0268	0.0037	0.0580	0.9980	0.0003	0.0001	30.0980
gbr	Gradient Boosting Regressor	0.0228	0.0077	0.0769	0.9960	0.0004	0.0001	2.3330
rf	Random Forest Regressor	0.0172	0.0078	0.0837	0.9958	0.0004	0.0001	1.9980
xgboost	Extreme Gradient Boosting	0.0287	0.0118	0.1010	0.9937	0.0005	0.0002	1.3440
knn	K Neighbors Regressor	0.0098	0.0149	0.0894	0.9927	0.0005	0.0001	0.0340
lightgbm	Light Gradient Boosting Machine	0.0331	0.0175	0.1169	0.9904	0.0006	0.0002	1.4910
dt	Decision Tree Regressor	0.0127	0.0293	0.1195	0.9842	0.0006	0.0001	0.0670
ada	AdaBoost Regressor	0.2659	0.1245	0.3344	0.9291	0.0017	0.0014	0.4860
en	Elastic Net	0.1553	0.2003	0.3381	0.9024	0.0018	0.0008	0.0590
lasso	Lasso Regression	0.2025	0.2330	0.3929	0.8888	0.0021	0.0011	0.0560
br	Bayesian Ridge	0.0331	0.2595	0.2590	0.8600	0.0013	0.0002	0.0920
ridge	Ridge Regression	0.0527	0.4442	0.4582	0.7670	0.0024	0.0003	0.0260
lr	Linear Regression	0.0559	0.6537	0.4940	0.6560	0.0025	0.0003	0.3000
llar	Lasso Least Angle Regression	1.0595	1.8923	1.3724	-0.0117	0.0072	0.0056	0.0100
omp	Orthogonal Matching Pursuit	0.1462	1.9828	0.9172	-0.0158	0.0047	0.0008	0.0200
par	Passive Aggressive Regressor	0.8778	11.5925	2.7562	-4.7918	0.0164	0.0046	0.0440
huber	Huber Regressor	0.6891	12.6339	2.7855	-5.2934	0.0168	0.0036	0.3380
lar	Least Angle Regression	1273780.6347	1484725286365969.7500	13208822.9986	-842536182986036.7500	3.7640	6704.7715	0.0780

Fonte – o autor

O procedimento comum realizado na intenção de obter os melhores resultados possíveis para o modelo selecionado é a busca por hiperparâmetros. Essa busca é realizada executando o método *tune_model*, que trabalha considerando uma validação cruzada de 10 grupos em um conjunto de 10 iterações por grupo. Os resultados das métricas de avaliação para a execução da busca dos hiperparâmetros são apresentados pela Figura 7.2.

Através da Figura 7.2 pode-se verificar que não houve uma melhoria na performance do modelo com a busca dos hiperparâmetros (limitada a 100 rodadas com 10 grupos). Isso ocorreu pois o modelo obteve um ótimo resultado com os hiperparâmetros padrões. Esses resultados são evidenciados pela Figura 7.1.

Para avaliar as predições realizadas pelo modelo no conjunto de validação, deve-se utilizar o método para plotar a distribuição dos resíduos num gráfico. No contexto dos mo-

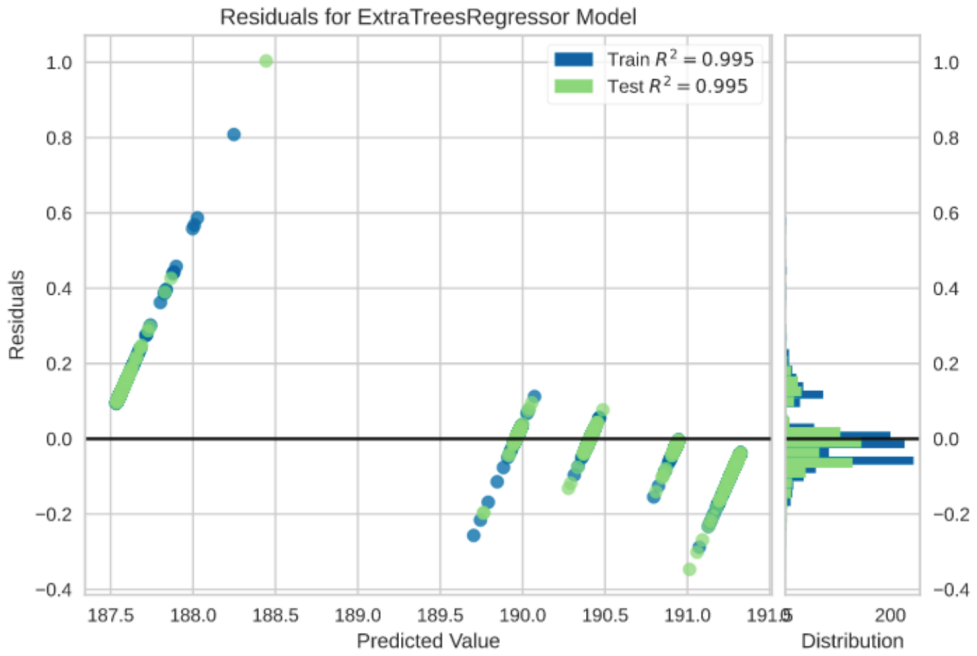
Figura 7.2 – Resultados para da validação cruzada de 10 grupos realizado na busca pelos hiperparâmetros.

	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	0.0699	0.0125	0.1118	0.9947	0.0006	0.0004
1	0.0735	0.0151	0.1228	0.9910	0.0006	0.0004
2	0.0713	0.0113	0.1061	0.9941	0.0006	0.0004
3	0.0744	0.0129	0.1134	0.9943	0.0006	0.0004
4	0.0569	0.0078	0.0884	0.9959	0.0005	0.0003
5	0.0458	0.0049	0.0699	0.9965	0.0004	0.0002
6	0.0724	0.0191	0.1382	0.9895	0.0007	0.0004
7	0.0541	0.0075	0.0867	0.9959	0.0005	0.0003
8	0.0479	0.0045	0.0671	0.9974	0.0004	0.0003
9	0.0642	0.0113	0.1064	0.9939	0.0006	0.0003
Mean	0.0630	0.0107	0.1011	0.9943	0.0005	0.0003
SD	0.0104	0.0043	0.0216	0.0023	0.0001	0.0001

Fonte – o autor

delos de regressão, esse gráfico apresenta a diferença entre o valor observado da variável alvo (y) e o valor previsto (ŷ), ou seja, o erro da previsão. O gráfico de resíduos apresenta a diferença entre os resíduos no eixo vertical e a variável dependente no eixo horizontal, permitindo detectar regiões dentro do alvo que possam ser suscetíveis a mais ou menos erros. A Fig.7.3 apresenta os resíduos e sua distribuição para a previsão dos dados de treino e validação.

Figura 7.3 – Resultados para da validação cruzada de 10 grupos realizado na busca pelos hiperparâmetros.



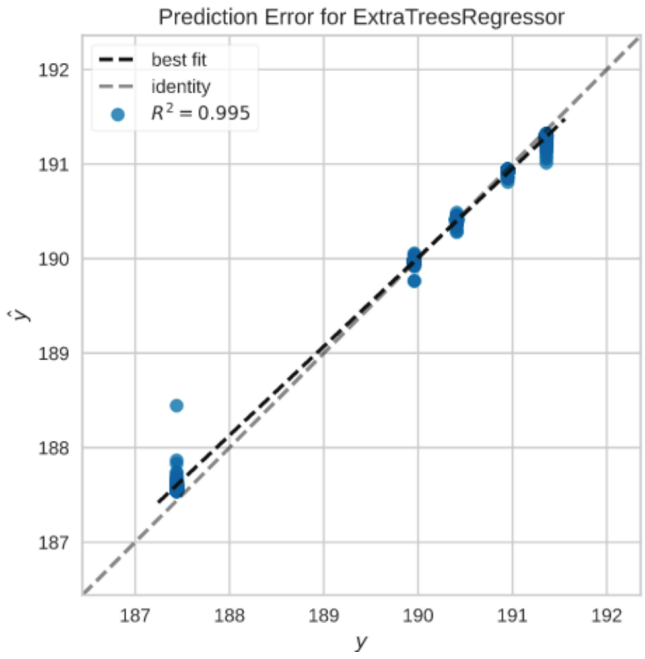
Fonte – o autor

A distribuição dos erros do modelo se aproxima de uma distribuição normal, centralizada em 0, a qual pode ser observada pelo gráfico de distribuição da Figura 7.3. No gráfico

é possível concluir que o modelo possui uma alta capacidade em estimar a massa conforme a IS.

Para analisar em quais regiões o modelo errou as previsões, é analisado o gráfico de erro de previsão, o qual mostra os alvos reais do conjunto de dados em relação aos valores previstos gerados pelo modelo. Isso permite ver quanta variação existe no modelo. É possível diagnosticar modelos de regressão usando este gráfico comparando com uma linha de 45 graus, a qual representa uma previsão perfeita do modelo. A Figura 7.4 apresenta os erros e acertos do modelo para os dados de validação.

Figura 7.4 – Erros da predição do modelo de regressão para os dados de validação.



Fonte – o autor

É possível identificar na Figura 7.4 a linha de 45 graus cinza tracejada, representando uma predição perfeita e a linha preta tracejada, representando a performance do modelo. Dessa forma, fica evidente a alta capacidade de predição de massa para o corpo de prova 1 adquirida pelo modelo.

Para simular a existência de mais IS e verificar a capacidade de extrapolar para condições não treinadas, deve-se avaliar o modelo sob os dados de teste separados da construção do modelo. Os resultados das métricas sob os dados de teste são apresentados pela Figura 7.5.

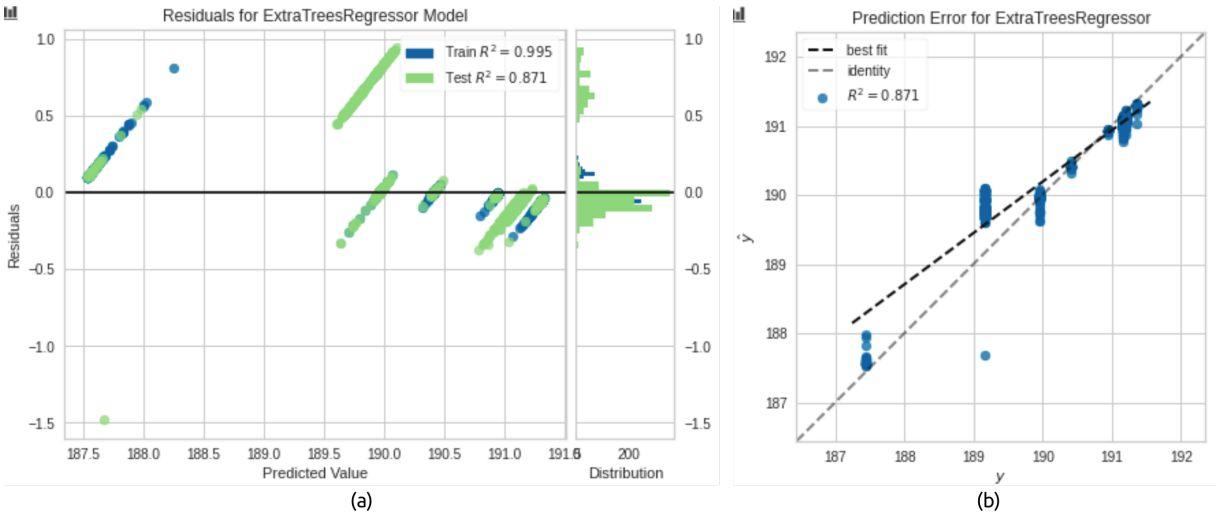
Figura 7.5 – Resultados das métricas das predições do modelo de regressão para os dados de teste.

Model	MAE	MSE	RMSE	R2	RMSLE	MAPE
Extra Tress Regressor	0.2086	0.1143	0.3381	0.8709	0.0018	0.0011

Fonte – o autor

Conforme a Figura 7.5, as predições apresentam níveis de erros baixos, porém existe uma diferença na magnitude, em comparação com as métricas sobre os dados de treino/-validação, indicado a existência de possíveis erros na predição de massa do modelo. Para verificar a existência desses erros bem como as regiões que o modelo errou, se faz necessário utilizar os gráficos de distribuição de resíduos e erros de predição. A Figura 7.6 apresenta os erros e acertos do modelo, alem da distribuição dos erros das predições para os dados de teste.

Figura 7.6 – Distribuição dos resíduos e erros das predições do modelo de regressão para os dados de teste.



Fonte – o autor

O desempenho do modelo para predizer a massa com relação a sua IS sobre dados desconhecidos pelo modelo obteve bons resultados, errando apenas na região de 189.16 gramas que foi predita como tendo uma massa de 190.41. Vale ressaltar que esses erros deixaram a distribuição dos resíduos assimétrica indicando que o modelo errou mais para o sentido positivo, ou seja, prevendo alguma IS com mais massa do que realmente possui.

7.1.1 Interpretando o Modelo de Regressão

Para compreender como o modelo interage com as faixas de frequência afim de interpretar e justificar as predições do modelo foi empregada a biblioteca SHAP, apresentada na Seção 3.6.2.3. Dessa forma, com o objetivo de identificar as variáveis que mais impactam na predição de massa a partir de uma IS, utilizou-se o método *force_plot* para apresentar as "forças"que guiam a predição. Foi selecionado aleatoriamente uma IS que possuía uma massa de 190.41 gramas e então aplicou-se o método para inspecionar o impacto das variáveis nessa predição. A Figura 7.7 representa o impacto das faixas de frequência mais importantes da IS de impedância para a predição dessa massa.

De acordo com a Figura 7.7, o gráfico de força apresenta uma massa média, também denominada de valor base, a qual pode ser interpretada como uma predição caso nenhum

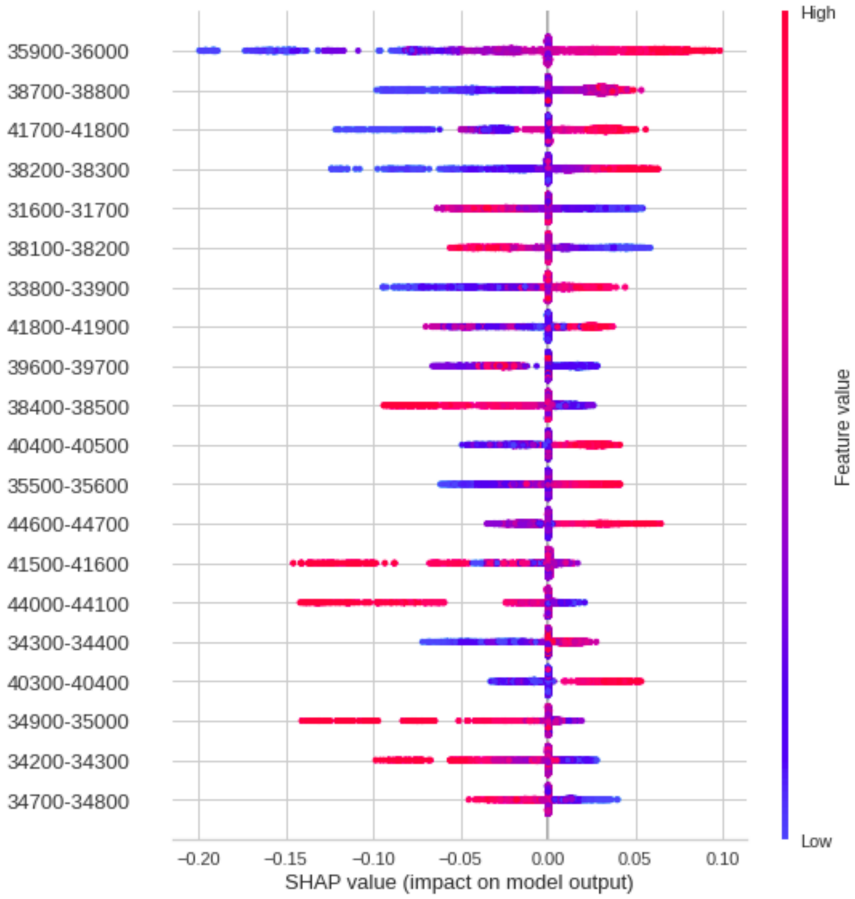
Figura 7.7 – Influência das faixas de frequência na predição de massa.

Fonte – o autor

recurso fosse passado para o modelo. As variáveis tem o papel de "empurrar"esse valor para a massa predita. De forma macro, para a IS selecionada, as faixas de frequência de 31600 Hz a 41800 Hz tiveram um impacto significativo no deslocamento do valor base de massa.

Uma análise macro das variáveis do modelo é dada pelo método *summary_plot*, o qual apresenta uma comparação dos valores das variáveis indicados pela cor, bem como a forma (diminuindo ou aumentando a massa em análise) como esses valores impactam no modelo. A Figura 7.8, apresenta o gráfico de resumo dos impactos das faixas de frequência considerando a magnitude desses valores para as predições de massa.

Figura 7.8 – Gráfico de resumo de impacto para as faixas de frequência na predição de massa.



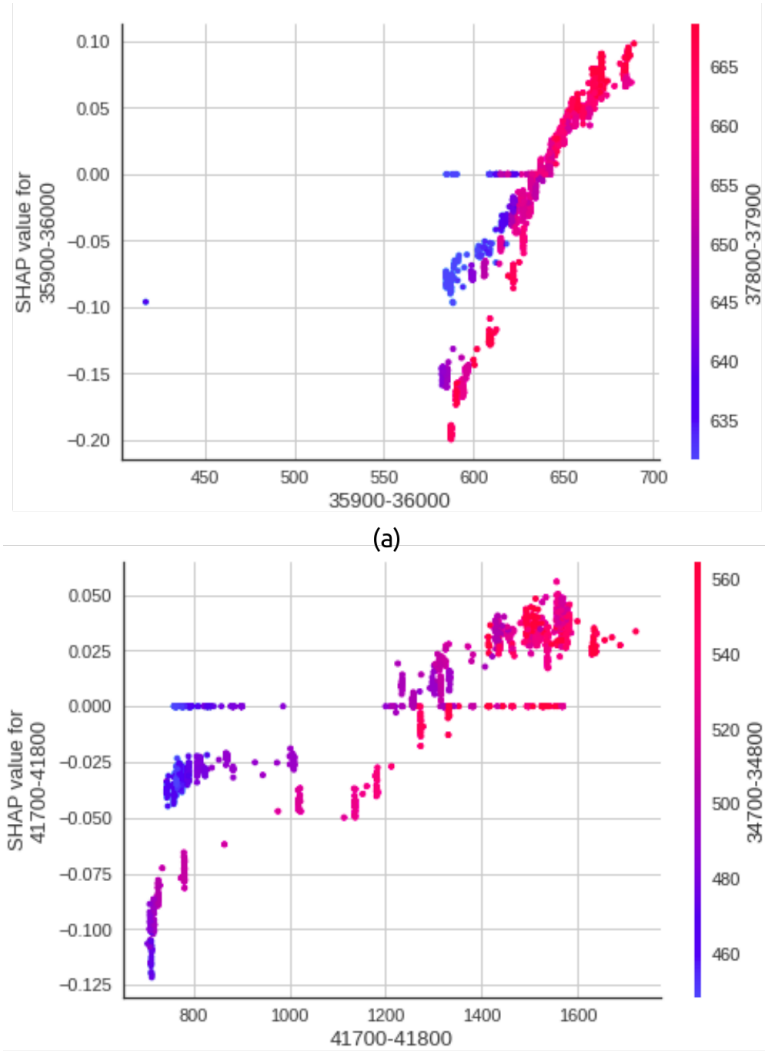
Fonte – o autor

De acordo com a Figura 7.8, é possível identificar o tamanho e a separabilidade para

cada faixa de frequência (conforme ocorre nas modelagens de multi-classificação e detecção de anomalia), indicando uma boa interpretação por parte do modelo. Essa separabilidade observada nos valores SHAP neste gráfico indicam que dado um parâmetro de entrada, este desloca o valor base de massa em uma direção baseado na sua magnitude.

Para se avaliar as relações dos impactos entre as variáveis no modelo, utiliza-se o gráfico de dependência através do método *dependence_plot*, o qual recebe como parâmetro uma ou duas faixas de frequência para apresentar as iterações. A análise consiste em avaliar o impacto de uma faixa de frequência em relação a outra, identificando correlações úteis tanto para melhoria do modelo, quanto para justificar resultados das classificações. A Figura 7.9 apresenta duas comparações: (a) 35900 Hz a 36000 Hz com 37800 Hz a 37900 Hz e (b) 41700 Hz a 41800 Hz com 34700 Hz a 34800 Hz.

Figura 7.9 – Gráfico de dependência entre faixas de frequência para a predição de massa.



Fonte – o autor

Conforme pode ser observado pela Figura 7.9, existe uma correlação positiva em (a)

e (b) na qual, ao aumentar os valores de impedância da faixa de frequência, aumenta o deslocamento do valor base, ou seja, aumentar a massa. Diversos tipos de análises análogas podem ser realizadas com o uso destes gráficos interativos no ambiente Python, para compreensão da relação entre as variáveis.

Concluindo, nesta seção foi apresentada uma abordagem para predição de massa utilizando as bibliotecas Pycaret. Foi selecionado o modelo que obteve os melhores resultados e, posteriormente, na intenção de interpretar o modelo treinado, foi utilizada a biblioteca SHAP. Foram apresentados os critérios de escolha e decisão do modelo baseado em diversas métricas. Por fim, foram apresentados alguns gráficos específicos do SHAP que são muito informativos para cada condição de resposta. Com isto, acredita-se que o ferramental apresentado seja suficiente na seleção e construção de excelentes modelos de aprendizado de máquina para representação de dados e sistemas, assim como a compreensão dos principais parâmetros de entrada do modelo e suas combinações em relação a cada classe de resposta.

Capítulo 8

APLICATIVO PARA DETECÇÃO DE FALHAS

Neste Capítulo é apresentada a etapa de entrega de uma solução de aprendizado de máquina. Para que um modelo de aprendizado de máquina possa resolver um problema, é necessário que o mesmo seja disponibilizado para uso. Isso é feito através de aplicações, responsáveis por criar uma interface entre as pessoas e a inteligência desenvolvida. Para o problema de detecção de dano em uma viga de alumínio, foi desenvolvido um aplicativo web capaz de identificar assinaturas de impedância com e sem dano. Portanto, foi desenvolvida uma interface utilizando a biblioteca Streamlit. Para implantação e disponibilização da aplicação, é utilizada a ferramenta Docker de conteinerização em conjunto com os recursos de computação em nuvem do fornecedor AWS.

8.1 Construção do Aplicativo

Para o desenvolvimento de um sistema para detecção de falhas, deve-se considerar a infraestrutura do sistema e tecnologias para implantação: sensoriamento do sistema a ser monitorado, servidor e protocolo de comunicação baseado em IOT (do inglês - *Internet of Things*), modelos e *engines* para avaliação dos dados e ambiente de implantação (servidores convencionais ou nuvem).

O servidor, em conjunto com o protocolo de comunicação utilizado, tem o papel de disponibilizar e gerenciar a coleta, armazenamento e disponibilização dos dados adquiridos pelos sensores/*hardware*. Os protocolos comumente utilizados para sistemas de IOT são: Protocolo de Aplicativo Restrito (CoAP), Transporte de Telemetria de Enfileiramento de Mensagens (MQTT), Protocolo Extensível de Mensagens e Presença (XMPP), Serviço de Distribuição de Dados (DDS), Protocolo de Enfileiramento de Mensagens Avançado (AMQP) e M2M leve (LwM2M).

Com a seleção do protocolo a ser empregado, é então integrado o modelo para avaliação dos dados em um ambiente afim de disponibilizar o monitoramento. A implementação do protocolo foi desconsiderada nessa contribuição e o mesmo foi simulado alimentando o aplicativo disponibilizado com um arquivo no formato .csv que continha os dados a serem avaliados.

Para o desenvolvimento do aplicativo, foi utilizado a biblioteca Streamlit, em conjunto com as ferramentas Docker Compose e Docker Machine, afim de criar um microsserviço de avaliação de integridade e então disponibilizar o mesmo no ambiente de computação em nuvem do provedor AWS.

Para isso, foi implementado um aplicativo para avaliar a integridade do corpo de prova 1 utilizando o modelo baseado na classificação binária supervisionada, apresentado na Seção 5.1.3. A Figura 8.1, apresenta o aplicativo desenvolvido.

Para executar localmente o aplicativo apresentado pela Figura 8.1, é executado o comando *streamlit run app.py*, e por padrão o mesmo é hospedado na porta 8501. Após validar as funcionalidades do aplicativo localmente, o mesmo é então convertido para o ambiente Docker no formato de microsserviço, através do Dockerfile e Docker Compose. A Fig 8.2 apresenta o diagrama do microsserviço desenvolvido para o aplicativo.

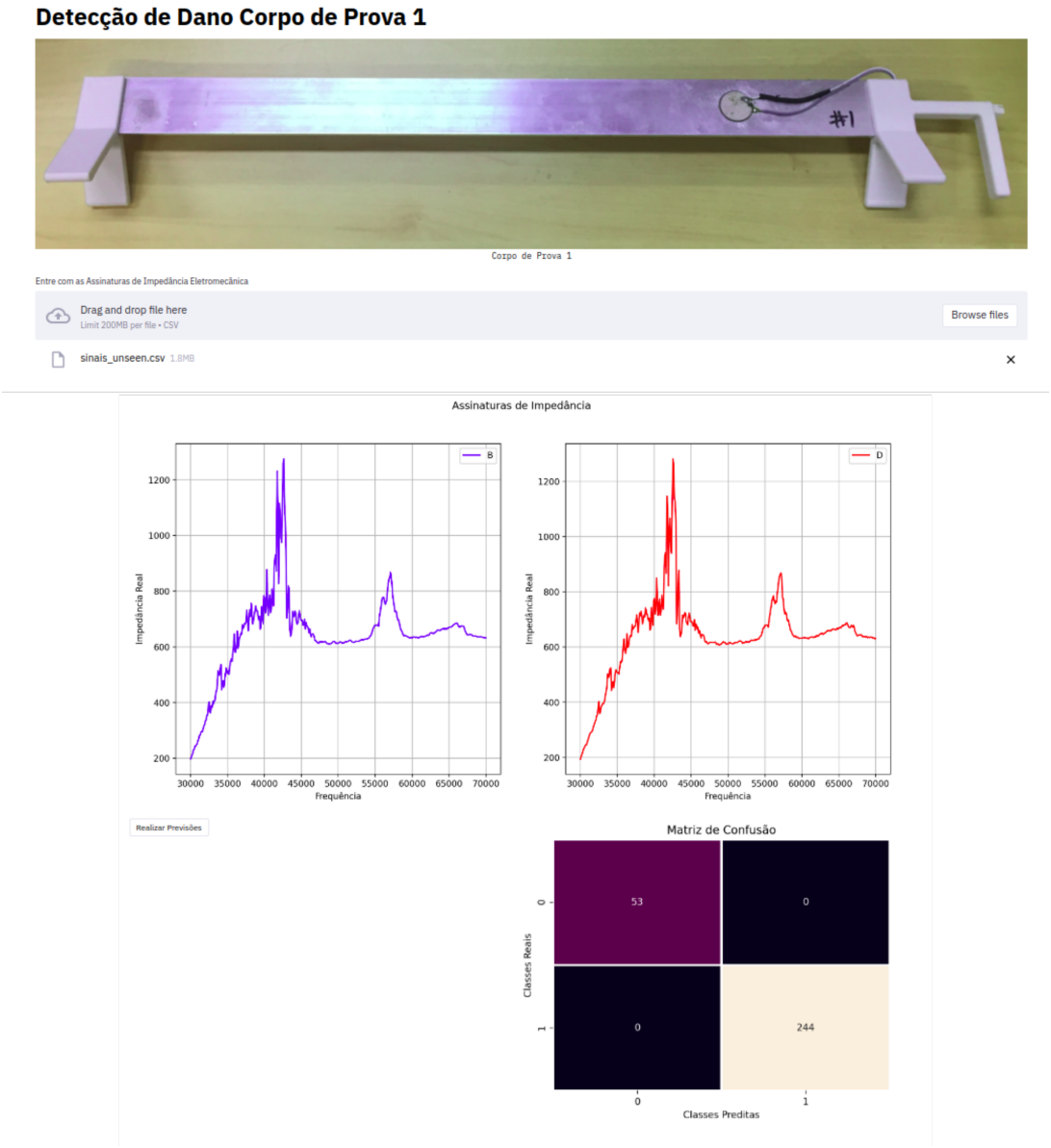
A Figura 8.2 apresenta o diagrama do aplicativo desenvolvido juntamente com as possibilidades de microsserviços, como banco de dados e servidor MQTT. É possível identificar as características principais para gerenciar diversos contêineres a partir de um único comando. Também é possível criar e iniciar todos os serviços especificados nas configurações. Além disso, vale destacar o papel dos Dockerfiles, responsáveis por construir os contêineres, criando imagens com rotinas específicas para o funcionamento do aplicativo.

Para avaliar a construção dos contêineres a partir do Docker Compose, pode-se construir com o comando via terminal *docker-compose up* no diretório dos arquivos. Em seguida, a aplicação estará disponível localmente na porta 8501 conforme especificado pelo arquivo YAML.

Como última etapa, é realizada a disponibilização ou implantação do aplicativo utilizando os recursos da computação em nuvem, fornecidos pelo provedor da Amazon AWS. Para realizar essa implantação, foram utilizadas as ferramentas de Interface da Linha de Comando (CLI) da AWS, que é uma ferramenta unificada para o gerenciamento de serviços da AWS, e a ferramenta Docker Machine, utilizada para criar um ambiente adequado para a execução do aplicativo desenvolvido. A Figura 8.3 apresenta o diagrama da implantação do aplicativo para detecção de danos no ambiente de computação em nuvem.

O processo apresentado pela Figura 8.3 se inicia com a parametrização das credenciais via linha de comando através da CLI da AWS. Isso é feito através do comando *aws configure*. Ao executar esse comando, são solicitadas 4 informações de credencial disponíveis

Figura 8.1 – Aplicativo para detecção de dano para o corpo de prova 1.



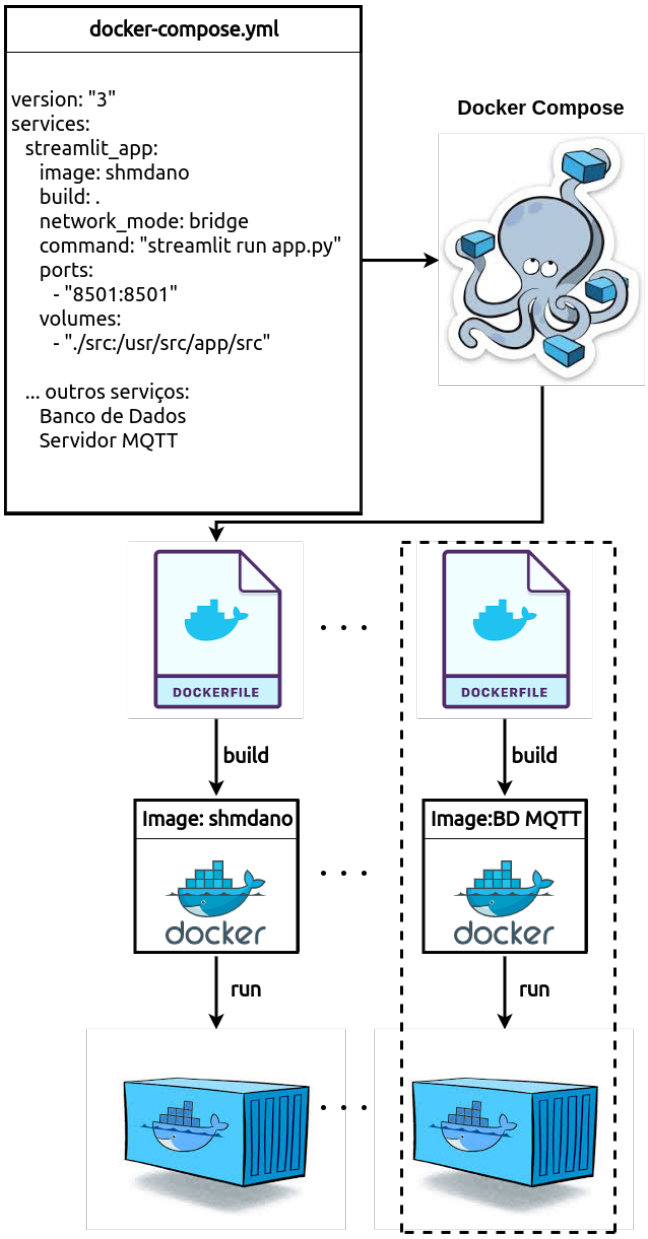
Fonte – o autor

no console da AWS na aba "Suas credenciais de segurança", em seguida, no item "Chaves de acesso (ID da chave de acesso e a chave de acesso secreta)". Caso não exista nenhuma chave deve-se criar uma nova.

Em seguida, é executado o comando para criação do ambiente preparado para executar os contêineres na nuvem. Para isso executa-se o comando *docker-machine create –driver amazonec2 aws01*. Nesse comando é especificado o drive a ser criado, (nesse caso o *amazonec2*) e em seguida o nome da máquina (*aws01*).

Após a criação da máquina no servidor da AWS é então realizado o procedimento para enviar os contêineres para a máquina. Para isso é então executado o comando *docker-*

Figura 8.2 – Diagrama do Docker Compose desenvolvido para o microsserviço do aplicativo.



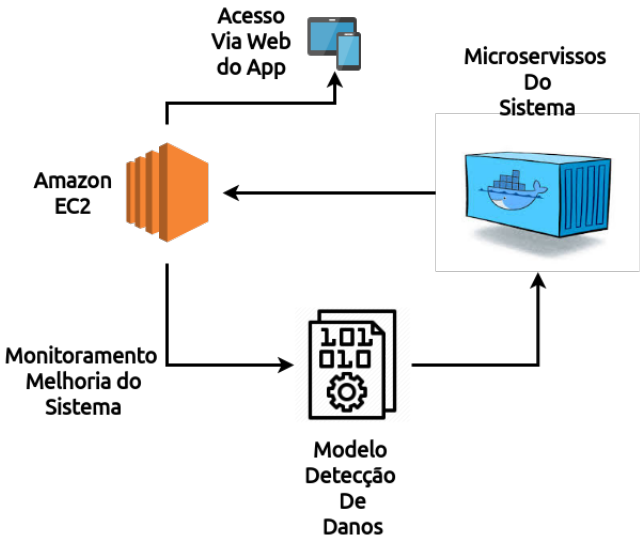
Fonte – o autor

machine env aws01 e em seguida o *eval $(docker-machine env aws01)*, dessa forma é realizada a conexão com a máquina provisionada na AWS e todos os comandos executados serão executados na nuvem. Com isso, pode-se executar o comando para construir a aplicação dentro da AWS através do comando *docker-compose up*.

Ao finalizar a construção do aplicativo pode-se verificar se existe algum contêiner em execução na máquina com o comando *docker ps* e então, para acessar o aplicativo provisionado, deve-se abrir a porta 8501 no console da AWS para acesso de qualquer pessoa. Por fim, a aplicação estará disponível através do IPV4 público na porta 8501.

O aplicativo é então monitorado e constantemente atualizado com a intenção de

Figura 8.3 – Diagrama da implantação do aplicativo em nuvem do aplicativo.



Fonte – o autor

manter a consistência dos resultados e garantir resultados confiáveis para a detecção de danos.

Capítulo 9

CONCLUSÕES E TRABALHOS FUTUROS

O monitoramento da integridade estrutural (SHM) é um campo de pesquisa que vem ganhando cada vez mais relevância nos últimos anos. Um dos principais fatores que têm contribuído para a evolução dos estudos nesta área foi o desenvolvimento da tecnologia e aplicabilidade dos sensores, além dos avanços recentes nas técnicas de aprendizado de máquina. Dessa forma, considerando as diversas oportunidades de pesquisa para o SHM, esta dissertação foca na utilização da técnica não-destrutiva de impedância eletromagnética junto com métodos de inteligência artificial para o desenvolvimento de sistemas de monitoramento de dano numa viga metálica. A escolha desse estudo se deu pela importância e frequência desse tipo de dano aliado com a inexistência de pesquisas semelhantes para essa vertente do problema.

Para garantir que a pesquisa seja bem elaborada e que seus resultados sejam atuais e de qualidade, é necessário o estudo dos principais conceitos do problema, além de uma revisão bibliográfica de pesquisas relevantes. Portanto, começando pelos principais conceitos referentes à SHM (Capítulo 2), também foram descritos os elementos essenciais para o desenvolvimento de modelos de inteligência artificial, bem como os tipos de aprendizagem, ferramentas para analise de resultados, interpretação e implantação de modelos. Além disso, também foram apresentadas as principais contribuições aplicadas ao monitoramento de integridade estrutural (Capítulo 3).

Estudando os aspectos e funcionalidades de um sistema SHM fica clara a relevância das abordagens envolvendo desempenho e segurança, uma vez que, caso essas estruturas falhem, existe não só a perda econômica, como também o risco de vida para todas as pessoas que estejam em contato com as mesmas. Portanto, é de suma importância utilizar modelos capazes de desempenhar com excelência a função de monitorar a integridade das estruturas. Dessa forma, no Capítulo 4 foi empregado um experimento com variações de temperatura

e integridade, visando estudar as principais tarefas e modelos de aprendizado de máquina, aliados a ferramentas importantes de análise dos resultados e interpretação das decisões.

Para a modelagem do sistema, foram avaliados três tipos de tarefas: detecção de anomalias, multi classificação e regressão (Capítulos 5, 6 e 7). Nessas tarefas foram empregadas ferramentas automáticas de aprendizado de máquina, tanto para avaliação de muitos modelos, quanto para análise dos resultados. A alta capacidade em identificar padrões dos modelos utilizados possibilitou atingir ótimos desempenhos no monitoramento de integridade, considerando dano e variação de temperatura sem a necessidade de nenhum tratamento de compensação, redução de ruídos ou extração de métricas de dano, ou seja, utilizando apenas faixas frequenciais. Esses resultados mostraram a relevância e utilidade das tarefas avaliadas ao serem aplicadas no SHM.

A justificação e interpretação dos resultados é outra ferramenta de grande valia para modelagem do sistema, uma vez que essa possibilita melhoria dos modelos, diminuição de complexidade, redução da quantidade de faixas de frequências avaliadas, interpretação profunda e com sensibilidade, identificando iteração entre variáveis, permitindo monitorar variações pontuais nas faixas de frequência com os gráficos de dependência e justificando e interpretando os resultados obtidos para cada assinatura de impedância.

Com o modelo definido e avaliado, foi apresentado o processo para disponibilizá-lo utilizando recursos de conteinerização (Capítulo 8). Esse processo, possibilita uma fácil, padronizada, rápida implantação extensão dos microsserviços como: banco de dados, servidor com protocolo de comunicação para IOT e *engines* de inteligência artificial através de microsserviços.

O provisionamento dos contêineres no ambiente de computação em nuvem reforça e garante o pleno funcionamento, ou seja, usufrui da infraestrutura de computação global escalável, confiável e segura. Proporciona mais flexibilidade para manipular e utilizar as máquinas criadas. Por fim, entrega uma alta capacidade de escalar a utilização dos recursos da forma mais econômica possível, pois utiliza o serviço de *pay to use* empregando o *Auto Scaling* e *Elastic Load Balancing*, para expandir ou reduzir a infraestrutura e capacidade de processamento de acordo com a demanda.

A integração dessas etapas abordadas consistem no desenvolvimento de um sistema para monitoramento de integridade estrutural onde foi apresentada uma gama de possibilidades para a modelagem, análise e implantação desse sistema. Foi possível demonstrar a praticidade e eficiência de todas as ferramentas e métodos utilizados, além da sua adaptabilidade para qualquer outro problema do tipo SHM.

Como trabalhos futuros, pretende-se desenvolver um sistema de monitoramento de integridade web, considerando o ciclo de vida de um sistema de aprendizado de máquina apresentado no Capítulo 3.6 além de utilizar os demais recursos tecnológicos citados ante-

riormente, aplicando-os em uma estrutura real que esteja em operação, afim de obter um produto validado, replicável e utilizável.

REFERÊNCIAS

AFSHARI, M. *Vibration- and Impedance-based Structural Health Monitoring Applications and Thermal Effects*. 144 f. Tese (Doctor of Philosophy In Mechanical Engineering) — Virginia Polytechnic Institute and State University, Blacksburg, VA, 2012. Citado 3 vezes nas páginas 30, 32 e 35.

AKOSSOU, A.; PALM, R. Impact of data structure on the estimators r-square and adjusted r-square in linear regression. *Int. J. Math. Comput*, v. 20, p. 84–93, 2013. Citado na página 56.

AMARAL, M. *et al.* Performance evaluation of microservices architectures using containers. In: IEEE. *2015 IEEE 14th International Symposium on Network Computing and Applications*. [S.l.], 2015. p. 27–34. Citado na página 86.

ANNAMDAS, V. G. M.; YANG, Y. Practical implementation of piezo-impedance sensors in monitoring of excavation support structures. *Structural Control and Health Monitoring*, Wiley Online Library, v. 19, n. 2, p. 231–245, 2012. Citado na página 29.

ANTUNES, R. A. *et al.* Modeling, simulation, experimentation, and compensation of temperature effect in impedance-based shm systems applied to steel pipes. *Sensors*, Multidisciplinary Digital Publishing Institute, v. 19, n. 12, p. 2802, 2019. Citado na página 29.

ASHMORE, R.; CALINESCU, R.; PATERSON, C. Assuring the machine learning lifecycle: Desiderata, methods, and challenges. *arXiv preprint arXiv:1905.04223*, 2019. Citado na página 65.

AVEN, T. Interpretations of alternative uncertainty representations in a reliability and risk analysis context. *Reliability Engineering & System Safety*, Elsevier, v. 96, n. 3, p. 353–360, 2011. Citado na página 17.

BAHDANAU, D.; CHO, K.; BENGIO, Y. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014. Citado na página 43.

BANKO, M.; BRILL, E. Scaling to very very large corpora for natural language disambiguation. In: *Proceedings of the 39th annual meeting of the Association for Computational Linguistics*. [S.l.: s.n.], 2001. p. 26–33. Citado na página 63.

BANKS, H.; SMITH, R.; WANG, Y. *Smart material structures: modeling, estimation, and control*. [S.l.]: John Wiley & Sons, 1996. ISSN 0298-3168. ISBN 9780471970248. Citado 2 vezes nas páginas 22 e 24.

BANKS, H. T.; SMITH, R. C.; WANG, Y. Smart material structures: modeling, estimation, and control. John Wiley & Son Ltd, 1996. Citado na página 23.

BAO, Y. *et al.* Computer vision and deep learning–based data anomaly detection method for structural health monitoring. *Structural Health Monitoring*, SAGE Publications Sage UK: London, England, v. 18, n. 2, p. 401–421, 2019. Citado na página 18.

BAPTISTA, F. G. *et al.* An experimental study on the effect of temperature on piezoelectric sensors for impedance-based structural health monitoring. *Sensors*, Multidisciplinary Digital Publishing Institute, v. 14, n. 1, p. 1208–1227, 2014. Citado na página 29.

BAPTISTA, F. G.; FILHO, J. V. A new impedance measurement system for pzt-based structural health monitoring. *IEEE Transactions on Instrumentation and Measurement*, IEEE, v. 58, n. 10, p. 3602–3608, 2009. Citado 3 vezes nas páginas 34, 35 e 36.

BENGIO, Y.; GOODFELLOW, I.; COURVILLE, A. *Deep learning*. [S.l.]: MIT press Massachusetts, USA:, 2017. v. 1. Citado 4 vezes nas páginas 41, 53, 62 e 65.

BENTO, J. P. M. *et al.* Uso das cadeias de markov associado ao monitoramento da integridade estrutural baseado em impedância eletromecânica. Universidade Federal de Goiás, 2018. Citado 2 vezes nas páginas 23 e 30.

BHALLA, S. *et al.* Ultra low-cost adaptations of electro-mechanical impedance technique for structural health monitoring. *Journal of Intelligent Material Systems and Structures*, Sage Publications Sage UK: London, England, v. 20, n. 8, p. 991–999, 2009. Citado 2 vezes nas páginas 35 e 36.

____. Practical issues in the implementation of electro-mechanical impedance technique for nde. In: INTERNATIONAL SOCIETY FOR OPTICS AND PHOTONICS. *SPIE's International Symposium on Smart Materials, Nano-, and Micro-Smart Systems*. [S.l.], 2002. p. 484–494. Citado na página 29.

BHALLA, S.; NAIDU, A. S. K.; SOH, C. K. *Influence of structure-actuator interactions and temperature on piezoelectric mechatronic signatures for NDE*. 2003. 263-269 p. Citado 2 vezes nas páginas 29 e 30.

BITENCOURT, T. F. *et al.* Desenvolvimento de um sistema de medição de sinais de impedância para monitoramento de integridade estrutural baseado em impedância eletromecânica. In: . [S.l.]: VI Congresso Nacional De Engenharia Mecânica, 2010. Citado 2 vezes nas páginas 35 e 36.

BITENCOURT, T. F.; STEFFEN JÚNIOR, V. Monitoramento da integridade estrutural de aeronaves. *Horizonte Científico*, v. 3, n. 2, p. 18, 2009. Citado na página 30.

BORGES, J. A. Modelagem e desenvolvimento de algoritmos para pré-processamento de sinais em sistemas mecânicos cuja integridade estrutural é monitorada pelo método da impedância eletromecânica. Universidade Federal de Uberlândia, 2017. Citado 4 vezes nas páginas 22, 29, 30 e 32.

BRAY, D. E.; MCBRIDE, D. *Nondestructive testing techniques*. [S.l.]: New York : Wiley, 1992. ISBN 0471525138 (alk. paper). Citado na página 18.

BRUCE, P.; BRUCE, A.; GEDECK, P. *Practical Statistics for Data Scientists: 50+ Essential Concepts Using R and Python*. [S.l.]: O'Reilly Media, 2020. Citado 2 vezes nas páginas 55 e 56.

BUDOYA, D. E.; BAPTISTA, F. G. A comparative study of impedance measurement techniques for structural health monitoring applications. *IEEE Transactions on Instrumentation and Measurement*, IEEE, v. 67, n. 4, p. 912–924, 2018. Citado na página 29.

BUITINCK, L. *et al.* Api design for machine learning software: experiences from the scikit-learn project. *arXiv preprint arXiv:1309.0238*, 2013. Citado na página 68.

CASTRO, B. A. de; BAPTISTA, F. G.; CIAMPA, F. Comparative analysis of signal processing techniques for impedance-based shm applications in noisy environments. *Mechanical Systems and Signal Processing*, Elsevier, v. 126, p. 326–340, 2019. Citado na página 29.

____. A comparison of signal processing techniques for impedance-based damage characterization in carbon fibers under noisy inspections. *Materials Today: Proceedings*, Elsevier, 2020. Citado 3 vezes nas páginas 29, 92 e 96.

CAVALCANTE, A. M.; SCHONELL, J.; NETO, R. F. Modelos estatísticos aplicados ao monitoramento de integridade estrutural baseado na técnica da impedância eletromecânica. *REVISTA INTERDISCIPLINAR DE PESQUISA EM ENGENHARIA*, v. 2, p. 97, 2016. Citado na página 32.

CAWLEY, P. The impedance method of non-destructive inspection. *NDT international*, Elsevier, v. 17, n. 2, p. 59–65, 1984. Citado na página 17.

CELESTI, A. *et al.* Exploring container virtualization in iot clouds. In: IEEE. *2016 IEEE International Conference on Smart Computing (SMARTCOMP)*. [S.l.], 2016. p. 1–6. Citado 2 vezes nas páginas 19 e 86.

CHANDOLA, V.; BANERJEE, A.; KUMAR, V. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, ACM New York, NY, USA, v. 41, n. 3, p. 1–58, 2009. Citado 3 vezes nas páginas 43, 44 e 104.

CHAUDHRY, Z. *et al.* Monitoring the integrity of composite patch structural repair via piezoelectric actuators/sensors. In: *36th Structures, Structural Dynamics and Materials Conference*. [S.l.: s.n.], 1995. p. 1074. Citado na página 29.

CHAUDHRY, Z. A. *et al. Local-area health monitoring of aircraft via piezoelectric actuator/-sensor patches*. 1995. 268-276 p. Citado na página 29.

CHOLLET, F. *Deep Learning with Python*. [S.l.]: Manning Publications Co., 2018. Citado 2 vezes nas páginas 40 e 41.

COLLOBERT, R. Deep learning for efficient discriminative parsing. In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. [S.l.: s.n.], 2011. p. 224–232. Citado na página 43.

DEVICES., A. *(2013) AD5933 Datasheet*. [S.l.], 2008. Disponível em: <http://www.analog.corn/static/imported-files/data/sheets/AD5933.pdf>. Citado 2 vezes nas páginas 37 e 38.

DEVICES, A. *Data Sheet - AD5933*. 2017. Technical Support Analog Devices. Disponível em: <http://www.mouser.com/ds/2/609/AD5933-877234.pdf>. Acesso em: 22/07/2017. Citado na página 38.

DIGILENT. *PmodIA Reference Manual*. 2016. Digilent - A National Instruments Company. Disponível em: <https://reference.digilentinc.com/_media/pmod:pmod:pmodia_rm.pdf>. Acesso em: 22/07/2017. Citado na página 37.

DILLON, T.; WU, C.; CHANG, E. Cloud computing: issues and challenges. In: IEEE. *2010 24th IEEE international conference on advanced information networking and applications*. [S.l.], 2010. p. 27–33. Citado na página 19.

DJEMANA, M.; HRAIRI, M.; JEROUDI, Y. A. Using electromechanical impedance and extreme learning machine to detect and locate damage in structures. *Journal of Nondestructive Evaluation*, Springer, v. 36, n. 2, p. 39, 2017. Citado 2 vezes nas páginas 91 e 95.

DOCKER. *Docker Compose*. 2020. Disponível em: <https://docs.docker.com/compose/>. Citado na página 87.

____. *Docker Overview*. 2020. Disponível em: <https://docs.docker.com/get-started/overview/>. Citado 3 vezes nas páginas 19, 86 e 87.

DOEBLING, S. W.; FARRAR, C. R.; PRIME, M. B. A summary review of vibration-based damage identification methods. *The Shock and Vibration Digest*, v. 30, n. 2, p. 91–105, 1998. Citado na página 25.

DUNNING, T.; FRIEDMAN, E. *Practical machine learning: a new look at anomaly detection*. [S.l.]: "O'Reilly Media, Inc.", 2014. Citado na página 104.

ELATTAR, H. M.; ELMINIR, H. K.; RIAD, A. Prognostics: a literature review. *Complex & Intelligent Systems*, Springer, v. 2, n. 2, p. 125–154, 2016. Citado na página 17.

ELEFTHEROGLOU, N. *et al.* Structural health monitoring data fusion for in-situ life prognosis of composite structures. *Reliability Engineering & System Safety*, Elsevier, v. 178, p. 40–54, 2018. Citado na página 17.

ESPOSITO, C.; CASTIGLIONE, A.; CHOO, K.-K. R. Challenges in delivering software in the cloud as microservices. *IEEE Cloud Computing*, IEEE, v. 3, n. 5, p. 10–14, 2016. Citado na página 86.

FAIRWEATHER, J. A. Designing with active materials: An impedance based approach. 1999. Citado na página 22.

FARRAR, C. R.; DOEBLING, S. W.; NIX, D. A. Vibration–based structural damage identification. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, The Royal Society, v. 359, n. 1778, p. 131–149, 2001. ISSN 1364-503X. Citado na página 25.

FARRAR, C. R.; LIEVEN, N. A.; BEMENT, M. T. An introduction to damage prognosis. *Damage prognosis for aerospace, civil and mechanical systems*, Wiley Online Library, 2005. Citado na página 17.

FARRAR, C. R.; WORDEN, K. An introduction to structural health monitoring. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, The Royal Society, v. 365, n. 1851, p. 303–315, 2007. ISSN 1364-503X. Citado 2 vezes nas páginas 24 e 25.

FRANCOIS, C. *Deep learning with Python*. [S.l.]: Manning Publications Company, 2018. Citado 2 vezes nas páginas 39 e 40.

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. Citado 11 vezes nas páginas 45, 49, 50, 51, 52, 58, 60, 61, 62, 63 e 64.

GIBILISCO, S. *Manual de eletrônica e des telecomunicações*. [S.l.]: Reichmann & Affonso, 2002. Citado 2 vezes nas páginas 26 e 27.

GIURGIUTIU, V.; KROPAS-HUGHES, C. V. Comparative study of neural network damage detection from a statistical set of electro-mechanical impedance spectra. In: INTERNATIONAL SOCIETY FOR OPTICS AND PHOTONICS. *Smart Nondestructive Evaluation and Health Monitoring of Structural and Biological Systems II*. [S.l.], 2003. v. 5047, p. 108–119. Citado 2 vezes nas páginas 88 e 94.

GIURGIUTIU, V.; ZAGRAI, A. Damage detection in thin plates and aerospace structures with the electro-mechanical impedance method. *Structural Health Monitoring*, v. 4, n. 2, p. 99–118, 2005. Citado 2 vezes nas páginas 29 e 33.

GIURGIUTIU, V.; ZAGRAI, A. N. Characterization of piezoelectric wafer active sensors. *Journal of Intelligent Material Systems and Structures*, v. 11, n. 12, p. 959–976, 2000. Citado na página 29.

____. Embedded self-sensing piezoelectric active sensors for on-line structural identification. *International Journal of the Condition Monitoring and Diagnostic Engineering Management*, v. 124, p. 116–125, 2002. Citado na página 29.

GIURGIUTIU, V.; ZAGRAI, A. N.; BAO, J. J. Piezoelectric wafer embedded active sensors for aging aircraft structural health monitoring. *Structural Health Monitoring*, v. 1, n. 1, p. 41–61, 2002. Citado na página 29.

GOODFELLOW, I. J. *et al.* Multi-digit number recognition from street view imagery using deep convolutional neural networks. *arXiv preprint arXiv:1312.6082*, 2013. Citado na página 43.

HALEVY, A.; NORVIG, P.; PEREIRA, F. The unreasonable effectiveness of data. *IEEE Intelligent Systems*, IEEE, v. 24, n. 2, p. 8–12, 2009. Citado na página 63.

HARRISON, M. *Machine Learning Pocket Reference: Working with Structured Data in Python*. O'Reilly Media, 2019. ISBN 9781492047513. Disponível em: <https://books.google.com.br/books?id=RoirDwAAQBAJ>. Citado na página 66.

HAWKINS, D. M. The problem of overfitting. *Journal of chemical information and computer sciences*, ACS Publications, v. 44, n. 1, p. 1–12, 2004. Citado na página 64.

HINTON, G. *et al.* Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, IEEE, v. 29, n. 6, p. 82–97, 2012. Citado na página 43.

HUYNH, T.-C.; DANG, N.-L.; KIM, J.-T. Advances and challenges in impedance-based structural health monitoring. *Struct. Monit. Maint*, v. 4, n. 4, p. 301–329, 2017. Citado 2 vezes nas páginas 23 e 24.

INMAN, D. J. Smart structures: examples and new problems. In: XVI CONGRESSO BRASILEIRO DE ENGENHARIA MECÂNICA. *Anais*. Uberlândia: ABCM, 2001. p. 4–13. Citado na página 26.

INMAN, D. J. *et al. Damage prognosis: for aerospace, civil and mechanical systems*. [S.l.]: John Wiley & Sons, 2005. Citado 2 vezes nas páginas 17 e 24.

IRWIN, J. D. Introdução à análise de circuitos elétricos. *Rio de Janeiro: LTC,[19–]*, 2005. Citado na página 27.

ISLAM, J. *et al.* Docker enabled virtualized nanoservices for local iot edge networks. In: IEEE. *2019 IEEE Conference on Standards for Communications and Networking (CSCN)*. [S.l.], 2019. p. 1–7. Citado 2 vezes nas páginas 19 e 86.

JUN, Z.; JUAN, T.; TANG, H. M. Study of crack identifications in coupling beam based on the emi and ann techniques. In: TRANS TECH PUBL. *Applied Mechanics and Materials*. [S.l.], 2013. v. 333, p. 1625–1628. Citado 2 vezes nas páginas 90 e 95.

JUNIOR, A. W. L. *Eletricidade e Eletrônica Básica - 4ª Edição Revisada*. ALTA BOOKS, 2013. ISBN 9788576087779. Disponível em: <https://books.google.com.br/books?id=BlC2BAAAQBAJ>. Citado na página 26.

JUNIOR, P. d. O. C. Tool condition monitoring in the dressing process through electromechanical impedance and machine learning. Universidade Estadual Paulista (UNESP), 2020. Citado 2 vezes nas páginas 92 e 96.

KAMILA, S. Introduction, classification and applications of smart materials: an overview. *American Journal of Applied Sciences*, Science Publications, v. 10, n. 8, p. 876, 2013. Citado na página 21.

KIM, B.; KHANNA, R.; KOYEJO, O. O. Examples are not enough, learn to criticize! criticism for interpretability. In: LEE, D. *et al.* (Ed.). *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2016. v. 29, p. 2280–2288. Disponível em: <https://proceedings.neurips.cc/paper/2016/file/5680522b8e2bb01943234bce7bf84534-Paper.pdf>. Citado na página 79.

KOO, K.-Y. *et al.* Automated impedance-based structural health monitoring incorporating effective frequency shift for compensating temperature effects. *Journal of intelligent material systems and structures*, Sage Publications Sage UK: London, England, v. 20, n. 4, p. 367–377, 2009. Citado 2 vezes nas páginas 29 e 31.

LEDESMA, N. E. C. *et al. Desenvolvimento e Implementação de um Sistema para Detecção de Falhas em Estruturas usando Microcontrolador*. Dissertação (Mestrado) — Universidade Estadual Paulista (UNESP), 2012. Citado 2 vezes nas páginas 35 e 36.

LEUCAS, L. de F. *Utilização das técnicas de impedância eletromecânica e ondas lamb para identificação de dano em estruturas com rebites*. 80 f. Dissertação (Mestrado em Engenharia Mecânica) — Faculdade de Engenharia de Mecânica, Universidade Federal de Uberlândia, Uberlândia, 2009. Citado na página 18.

LIANG, C.; SUN, F.; ROGERS, C. Coupled electro-mechanical analysis of adaptive material systems — determination of the actuator power consumption and system energy transfer. *Journal of Intelligent Material Systems and Structures*, v. 5, n. 1, p. 12–20, 1994. Citado 2 vezes nas páginas 29 e 30.

LIM, H. J. *et al.* Impedance based damage detection under varying temperature and loading conditions. *Ndt & E International*, Elsevier, v. 44, n. 8, p. 740–750, 2011. Citado 3 vezes nas páginas 31, 89 e 94.

LIU, X.; JIANG, Z. Design of a pzt patch for measuring longitudinal mode impedance in the assessment of truss structure damage. *Smart Materials and Structures*, IOP Publishing, v. 18, n. 12, p. 125017, 2009. Citado na página 29.

LONZA, A. *Reinforcement Learning Algorithms with Python: Learn, understand, and develop smart algorithms for addressing AI challenges*. [S.l.]: Packt Publishing Ltd, 2019. Citado na página 50.

MAIO, C. E. B. *Técnicas para monitoramento de integridade estrutural usando sensores e atuadores piezoelétricos*. Tese (Doutorado) — Universidade de São Paulo, 2011. Citado 4 vezes nas páginas 25, 29, 30 e 32.

MASSOUD, M. *Impedance methods for machine analysis: modal parameters extraction techniques*. [S.l.]: Université de Sherbrooke, 1985. 4-14 p. Citado na página 27.

MAURYA, K. K.; RAWAT, A.; JHA, G. Smart materials and electro-mechanical impedance technique: A review. *Materials Today: Proceedings*, Elsevier, 2020. Citado na página 21.

MELL, P.; GRANCE, T. Draft nist working definition of cloud computing-v15. *21. Aug 2009*, v. 2, p. 123–135, 2009. Citado na página 19.

MILLER, T. *Explanation in Artificial Intelligence: Insights from the Social Sciences*. 2018. Citado na página 79.

MIN, J.; PARK, S.; YUN, C.-B. Impedance-based structural health monitoring using neural networks for autonomous frequency range selection. *Smart Materials and Structures*, IOP Publishing, v. 19, n. 12, p. 125011, 2010. Citado 2 vezes nas páginas 89 e 94.

MIN, J. *et al.* Impedance-based structural health monitoring incorporating neural network technique for identification of damage type and severity. *Engineering Structures*, Elsevier, v. 39, p. 210–220, 2012. Citado 2 vezes nas páginas 89 e 94.

MITCHELL, T. M. *Machine Learning, volume 1 of 1*. [S.l.]: McGraw-Hill Science/Engineering/-Math, 1997. Citado na página 41.

MOHEIMANI, S. R. A survey of recent innovations in vibration damping and control using shunted piezoelectric transducers. *IEEE transactions on control systems technology*, IEEE, v. 11, n. 4, p. 482–494, 2003. Citado na página 22.

MOLNAR, C. *Interpretable Machine Learning*: A guide for making black box models explainable. [S.l.: s.n.], 2019. <https://christophm.github.io/interpretable-ml-book/>. Citado na página 80.

MONAVARI, B. *SHM-based structural deterioration assessment*. Tese (Doutorado) — Queensland University of Technology, 2019. Citado na página 18.

MOURA JR, J. R. V.; STEFFEN JR, V. Impedance-based health monitoring for aeronautic structures using statistical meta-modeling. *Journal of intelligent material systems and structures*, Sage Publications Sage CA: Thousand Oaks, CA, v. 17, n. 11, p. 1023–1036, 2006. Citado na página 29.

MOURA JR, J. R. V.; STEFFEN JR, V. *Uma contribuição aos sistemas de monitoramento de integridade estrutural aplicada a estruturas aeronáuticas e espaciais*. Tese (Doutorado), 2008. Citado 6 vezes nas páginas 18, 26, 28, 29, 30 e 33.

NA, S.; LEE, H.-K. Neural network approach for damaged area location prediction of a composite plate using electromechanical impedance technique. *Composites science and technology*, Elsevier, v. 88, p. 62–68, 2013. Citado 2 vezes nas páginas 90 e 95.

NA, W. S.; BAEK, J. A review of the piezoelectric electromechanical impedance based structural health monitoring technique for engineering structures. *Sensors*, Multidisciplinary Digital Publishing Institute, v. 18, n. 5, p. 1307, 2018. Citado na página 29.

NANDY, A.; BISWAS, M. *Reinforcement Learning: With Open AI, TensorFlow and Keras Using Python*. [S.l.]: Apress, 2017. Citado na página 50.

NETO, R. M. F. *et al.* A low-cost electromechanical impedance-based shm architecture for multiplexed piezoceramic actuators. *Structural Health Monitoring*, SAGE Publications Sage UK: London, England, v. 10, n. 4, p. 391–402, 2011. Citado na página 29.

NEWNHAM, R.; RUSCHAU, G. R. Electromechanical properties of smart materials. *Journal of Intelligent Material Systems and Structures*, Sage Publications Sage CA: Thousand Oaks, CA, v. 4, n. 3, p. 289–294, 1993. Citado na página 22.

NICK, W. *et al.* A study of machine learning techniques for detecting and classifying structural damage. *International Journal of Machine Learning and Computing*, IACSIT Press, v. 5, n. 4, p. 313, 2015. Citado 2 vezes nas páginas 91 e 95.

OH, T.-K. *et al.* Nondestructive concrete strength estimation based on electro-mechanical impedance with artificial neural network. *Journal of advanced concrete technology*, Japan Concrete Institute, v. 15, n. 3, p. 94–102, 2017. Citado 2 vezes nas páginas 91 e 95.

OLIVEIRA, M. A. d. Monitoramento de integridade estrutural baseada em sensores piezelétricos e análise de sinais no domínio do tempo. Universidade Estadual Paulista (UNESP), 2013. Citado na página 29.

OLIVEIRA, M. A. D. *et al.* Use of savitzky–golay filter for performances improvement of shm systems based on neural networks and distributed pzt sensors. *Sensors*, Multidisciplinary Digital Publishing Institute, v. 18, n. 1, p. 152, 2018. Citado 2 vezes nas páginas 92 e 96.

PALOMINO, L. V. *Análise das métricas de dano associadas à técnica da impedância eletromecânica para o monitoramento de integridade estrutural*. 117 f. Dissertação (Mestrado em Engenharia Mecânica) — Faculdade de Engenharia de Mecânica, Universidade Federal de Uberlândia, Uberlândia, 2008. Citado 4 vezes nas páginas 18, 27, 29 e 34.

PALOMINO L, V. *et al.* Impedance-based health monitoring and mechanical testing of structures. *Smart Structures and Systems*, Techno-Press, v. 7, n. 1, p. 15–25, 2011. Citado na página 29.

PALOMINO, L. V. *et al.* Análise das métricas de dano associadas à técnica da impedância eletromecânica para o monitoramento de integridade estrutural. Universidade Federal de Uberlândia, 2008. Citado na página 32.

PALOMINO, L. V.; STEFFEN, V.; FINZI, R. M. Fuzzy cluster analysis methods applied to impedance based structural health monitoring for damage classification. In: *Topics in Modal Analysis II, Volume 6*. [S.l.]: Springer, 2012. p. 205–212. Citado 2 vezes nas páginas 90 e 94.

PANIGRAHI, R.; BHALLA, S.; GUPTA, A. A low-cost variant of electro-mechanical impedance (emi) technique for structural health monitoring. *Experimental Techniques*, Wiley Online Library, v. 34, n. 2, p. 25–29, 2010. Citado 2 vezes nas páginas 35 e 36.

PARIHAR, A.; KAJAL, D. k.; PALLAVI, P. J. Smart materials. *Journal of Mechanical and Civil Engineering*, 2016. Citado na página 22.

PARK, G.; CUDNEY, H. H.; INMAN, D. J. *Impedance-based health monitoring technique for massive structures and high-temperature structures*. 1999. 461-469 p. Citado na página 29.

____. Impedance-based health monitoring of civil structural components. *Journal of Infrastructure Systems*, v. 6, n. 4, p. 153–160, 2000. Citado 2 vezes nas páginas 29 e 34.

____. An integrated health monitoring technique using structural impedance sensors. *Journal of Intelligent Material Systems and Structures*, v. 11, n. 6, p. 448–455, 2000. Citado na página 29.

____. Feasibility of using impedance-based damage assessment for pipeline structures. *Earthquake Engineering & Structural Dynamics*, John Wiley & Sons, Ltd., v. 30, n. 10, p. 1463–1474, 2001. ISSN 1096-9845. Citado na página 29.

PARK, G.; INMAN, D. J. Impedance-based structural health monitoring. *Damage Prognosis for Aerospace, Civil and Mechanical System*, Wiley Online Library, 2005. Citado na página 28.

PARK, G. *et al.* Impedance-based structural health monitoring for temperature varying applications. *JSME International Journal Series A*, v. 42, n. 2, p. 249–258, 1999. Citado na página 29.

____. Overview of piezoelectric impedance-based health monitoring and path forward. *Shock and Vibration Digest*, v. 35, n. 6, p. 451–463, 2003. Citado 3 vezes nas páginas 24, 29 e 30.

PARK, S. *et al.* Piezoelectric sensor-based health monitoring of railroad tracks using a two-step support vector machine classifier. *Journal of Infrastructure Systems*, American Society of Civil Engineers, v. 14, n. 1, p. 80–88, 2008. Citado 2 vezes nas páginas 88 e 94.

____. Electro-mechanical impedance-based wireless structural health monitoring using pca-data compression and k-means clustering algorithms. *Journal of intelligent material systems and structures*, Sage Publications Sage UK: London, England, v. 19, n. 4, p. 509–520, 2008. Citado 2 vezes nas páginas 88 e 94.

PEAIRS, D. M. *Development of a self-sensing and self-healing bolted joint*. 94 f. Dissertação (Master of Science) — Faculty of Virginia Polytechnic Institute and State University, Blacksburg, VA, 2002. Citado na página 22.

PEAIRS, D. M. ____. Tese (Doutorado) — Virginia Tech, 2002. Citado 2 vezes nas páginas 35 e 36.

PEAIRS, D. M. *High frequency modeling and experimental analysis for implementation of impedance-based structural health monitoring*. Tese (Doutorado) — Virginia Tech, 2006. Citado na página 32.

PEAIRS, D. M.; PARK, G.; INMAN, D. J. Improving accessibility of the impedance-based structural health monitoring method. *Journal of Intelligent Material Systems and Structures*, Sage Publications, v. 15, n. 2, p. 129–139, 2004. Citado 2 vezes nas páginas 35 e 36.

PEDREGOSA, F. *et al.* Scikit-learn: Machine learning in python. *Journal of machine learning research*, v. 12, n. Oct, p. 2825–2830, 2011. Citado 3 vezes nas páginas 49, 58 e 61.

PIASECKI, T.; CHABOWSKI, K.; NITSCH, K. Design, calibration and tests of versatile low frequency impedance analyser based on arm microcontroller. *Measurement*, Elsevier, v. 91, p. 155–161, 2016. Citado 2 vezes nas páginas 35 e 36.

RABELO, D. d. S. *et al.* Monitoramento de integridade estrutural baseado na técnica da impedância eletromecânica incorporando compensação do efeito da variação da temperatura. Universidade Federal de Uberlândia, 2014. Citado na página 28.

____. Técnicas avançadas de normalização de dados aplicadas ao método de monitoramento de integridade estrutural baseado em impedância eletromecânica. Universidade Federal de Uberlândia, 2017. Citado 3 vezes nas páginas 28, 29 e 31.

RABELO, D. de S.; NETO, R. M. F.; JR, V. S. . Impedance-based structural health monitoring incorporating compensation of temperature variation effects. *Mechanical Engineering*, p. 1–8, 2015. Citado na página 29.

RABELO, D. S.; NETO, R. M. F.; STEFFEN JR, V. Impedance-based structural health monitoring incorporating compensation of temperature variation effects. In: *Proceedings of the 23rd ABCM International Congress of Mechanical Engineering. Rio de Janeiro.* [S.l.: s.n.], 2015. Citado na página 29.

RAJU, V. *Implementing impedance - based health monitoring*. 225 f. Dissertação (Master of science in mechanical engineering) — Faculty of Virginia Polytechnic Institute and State University, Blacksburg, VA, 1997. Citado na página 33.

RAUTELA, M. S.; VASHISHT, R.; BIJUDAS, C. Electromechanical impedance based shm and artificial neural networks for disbond type and severity. In: . [S.l.]: ASET-2018 conference, 2018. Citado 2 vezes nas páginas 91 e 95.

REZENDE, S. W. F.; BARELLA, B. P.; JR, R. V. M. Damage identification of vehicle brake disks by the use of impedance-based shm and unsupervised machine learning method. International Journal of Advanced Engineering Research and Science, 2020. Citado 2 vezes nas páginas 92 e 96.

REZENDE, S. W. F. de *et al.* Convolutional neural network and impedance-based shm applied to damage detection. *Engineering Research Express*, IOP Publishing, v. 2, n. 3, p. 035031, 2020. Citado na página 18.

ROGERS, C. A. *US Army Research Office Workshop on Smart Materials, Structures and Mathematical Issues Held in Blacksburg, Virginia on 15-16 September 1988*. [S.l.], 1989. Citado na página 21.

ROTH, W.; GIURGIUTIU, V. Structural health monitoring of an adhesive disbond through electromechanical impedance spectroscopy. *International Journal of Adhesion and Adhesives*, Elsevier, v. 73, p. 109–117, 2017. Citado na página 34.

RUGINA, C. *et al.* The electromechanical impedance method for structural health monitoring of thin circular plates. Citado na página 34.

RYTTER, A. Vibration based inspection of civil engineering structures [ph. d. thesis]. *Department of Building Technology and Structural Engineering, Aalborg University, Denmark*, 1993. Citado na página 25.

SATO, D.; WIDER, A.; WINDHEUSER, C. *Continuous Delivery for Machine Learning*. 2019. Disponível em: <https://martinfowler.com/articles/cd4ml.html>. Citado na página 66.

SELVA, P. *et al.* Smart monitoring of aeronautical composites plates based on electromechanical impedance measurements and artificial neural networks. *Engineering Structures*, Elsevier, v. 56, p. 794–804, 2013. Citado 2 vezes nas páginas 90 e 95.

SEPEHRY, N.; SHAMSHIRSAZ, M.; BASTANI, A. Experimental and theoretical analysis in impedance-based structural health monitoring with varying temperature. *Structural Health Monitoring*, v. 10, n. 6, p. 573–585, 2011. Citado na página 38.

SHAH, J.; DUBARIA, D. Building modern clouds: using docker, kubernetes & google cloud platform. In: IEEE. *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*. [S.l.], 2019. p. 0184–0189. Citado 2 vezes nas páginas 19 e 87.

SHANKER, R. *et al.* Dual use of pzt patches as sensors in global dynamic and local electromechanical impedance techniques for structural health monitoring. *Journal of Intelligent Material Systems and Structures*, Sage Publications Sage UK: London, England, v. 22, n. 16, p. 1841–1856, 2011. Citado na página 21.

SILVA, T. d. M. *et al.* Ensaios de carregamento dinâmico em estacas no complexo de suape. Universidade Católica de Pernambuco, 2011. Citado na página 32.

SOH, C. K. *et al.* Performance of smart piezoceramic patches in health monitoring of a rc bridge. *Smart Materials and Structures*, v. 9, n. 4, p. 533, 2000. Citado na página 29.

SUN, F. P. *et al.* Truss structure integrity identification using pzt sensor-actuator. *Journal of Intelligent material systems and structures*, TECHNOMIC PUBLISHING CO., INC. 851 New Holland Ave., Box 3535, Lancaster, PA . . . , v. 6, n. 1, p. 134–139, 1995. Citado na página 24.

____. ____. *Journal of Intelligent Material Systems and Structures*, v. 6, n. 1, p. 134–139, 1995. Citado 3 vezes nas páginas 29, 30 e 32.

____. Automated real-time structure health monitoring via signature pattern recognition. In: INTERNATIONAL SOCIETY FOR OPTICS AND PHOTONICS. *Smart Structures and Materials 1995: Smart Structures and Integrated Systems*. [S.l.], 1995. v. 2443, p. 236–247. Citado na página 29.

SUN, R.; SEVILLANO, E.; PERERA, R. Debonding detection of frp strengthened concrete beams by using impedance measurements and an ensemble pso adaptive spectral model. *Composite Structures*, Elsevier, v. 125, p. 374–387, 2015. Citado 2 vezes nas páginas 90 e 95.

SUTSKEVER, I.; VINYALS, O.; LE, Q. V. Sequence to sequence learning with neural networks. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2014. p. 3104–3112. Citado na página 43.

TAHMASEBPOUR, K. *Finite Element Modeling of Electro-mechanical Impedance Technique for Structural Health Monitoring*. Tese (Doutorado) — Kulliyyah of Engineering, International Islamic University Malaysia, 2014. Citado na página 29.

THOMSON, W. *Theory of vibration with applications*. [S.l.]: CrC Press, 2018. Citado na página 17.

TSENG, K. K.-H.; NAIDU, A. S. K. Non-parametric damage detection and characterization using smart piezoceramic material. *Smart Materials and Structures*, v. 11, n. 3, p. 317, 2002. Citado 2 vezes nas páginas 32 e 34.

TSURUTA, K. *et al.* Electromechanical impedance-based fault detection in a rotating machine by using an operating condition compensation approach. In: INTERNATIONAL SOCIETY FOR OPTICS AND PHOTONICS. *A Tribute Conference Honoring Daniel Inman*. [S.l.], 2017. v. 10172, p. 1017206. Citado 2 vezes nas páginas 29 e 31.

TSURUTA, K. M. *Monitoramento de integridade estrutural de materiais compostos sujeitos a impactos empregando a técnica da impedância eletromecânica*. 138 f. Dissertação (Mestrado em Engenharia Mecânica) — Faculdade de Engenharia de Mecânica, Universidade Federal de Uberlândia, Uberlândia, 2008. Citado na página 18.

TURING, I. B. A. Computing machinery and intelligence-am turing. *Mind*, v. 59, n. 236, p. 433, 1950. Citado na página 40.

UPHILL, T. *et al. DevOps: Puppet, Docker, and Kubernetes*. [S.l.]: Packt Publishing Ltd, 2017. Citado 2 vezes nas páginas 19 e 86.

VILLAMIZAR, M. *et al.* Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud. In: IEEE. *2015 10th Computing Colombian Conference (10CCC)*. [S.l.], 2015. p. 583–590. Citado na página 86.

VOIGT, W. *Lehrbuch der kristallphysik:(mit ausschluss der kristalloptik)*. [S.l.]: BG Teubner, 1910. v. 34. Citado na página 23.

WANG, Z. *et al.* Influence of axial load on electromechanical impedance (emi) of embedded piezoceramic transducers in steel fiber concrete. *Sensors (Basel, Switzerland)*, v. 18, n. 6, 2018. Citado na página 34.

WILLIAMS, J. H.; DAVIES, A.; DRAKE, P. R. *Condition-based maintenance and machine diagnostics*. [S.l.]: Springer Science & Business Media, 1994. Citado na página 17.

WOLPERT, D. H. The lack of a priori distinctions between learning algorithms. *Neural computation*, MIT Press, v. 8, n. 7, p. 1341–1390, 1996. Citado na página 65.

XU, B.; GIURGIUTIU, V. *A low-cost and field portable electromechanical (E/M) impedance analyzer for active structural health monitoring*. [S.l.], 2005. Citado 2 vezes nas páginas 35 e 36.

XU, R. A design pattern for deploying machine learning models to production. 2020. Citado 2 vezes nas páginas 19 e 88.

XU, Y.; LIAO, G.; LIU, T. Magneto-sensitive smart materials and magnetorheological mechanism. In: *Ferrohydrodynamics and Magnetohydrodynamics*. [S.l.]: IntechOpen, 2019. Citado na página 21.

ZHENG, A.; CASARI, A. *Feature engineering for machine learning: principles and techniques for data scientists*. [S.l.]: "O'Reilly Media, Inc.", 2018. Citado na página 63.