UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Câmpus São José do Rio Preto

Marcelo Velludo de Souza Prado

Tomografia de Hamiltoniano para Sistemas de Dois Qubits Utilizando Aprendizado de Máquina

Bauru
2023

Marcelo Velludo de Souza Prado

Tomografia de Hamiltoniano para Sistemas de Dois Qubits Utilizando Aprendizado de Máquina

Dissertação apresentada como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação, junto ao Programa de Pós-Graduação em Ciência da Computação, do Instituto de Biociências,Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", Câmpus de São José do Rio Preto.

Orientador: Prof. Dr. Felipe F. Fanchini

Faculdade de Ciências - UNESP

Bauru
2023

Prado, Marcelo Velludo de Souza.
Tomografia de Hamiltoniano para Sistemas de Dois Qubits Utilizando Aprendizado de Máquina / Marcelo Velludo de Souza Prado. – Bauru, 2023
66 f. : il., tabs.

Orientador: Prof. Dr. Felipe F. Fanchini
Dissertação (mestrado) - Universidade Estadual Paulista "Júlio de Mesquita Filho", Faculdade de Ciências

1. Tomografia de Hamiltoniano 2. Aprendizado de Máquina 3. Computação Quântica 4. Regressão I. Fanchini, Felipe F. II. Universidade Estadual Paulista "Júlio de Mesquita Filho", Faculdade de Ciências, Bauru, 2023 III. Título

UNESP - Câmpus de Bauru

UNIVERSIDADE ESTADUAL PAULISTA

Câmpus de Bauru

ATA DA DEFESA PÚBLICA DA DISSERTAÇÃO DE MESTRADO DE MARCELO VELLUDO DE SOUZA PRADO, DISCENTE DO PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO, DA FACULDADE DE CIÊNCIAS - CÂMPUS DE BAURU.

Aos 08 dias do mês de março do ano de 2023, às 14:00 horas, por meio de Videoconferência, realizou-se a defesa de DISSERTAÇÃO DE MESTRADO de MARCELO VELLUDO DE SOUZA PRADO, intitulada **Tomografia de Hamiltoniano para Sistemas de Dois Qubits Utilizando Aprendizado de Máquina**. A Comissão Examinadora foi constituída pelos seguintes membros: Prof. Dr. FELIPE FERNANDES FANCHINI (Orientador(a) - Participação Virtual) do(a) Departamento de Física / Unesp Faculdade de Ciências Campus de Bauru, Prof. Dr. JOAO PAULO PAPA (Participação Virtual) do Depto de Computação / FC Bauru Unesp, Prof. Dr. LEONARDO KLEBER CASTELANO (Participação Virtual) do Departamento de Física / Universidade Federal de São Carlos. Após a exposição pelo mestrando e arguição pelos membros da Comissão Examinadora que participaram do ato, de forma presencial e/ou virtual, o discente recebeu o conceito final APROVADO .

Nada mais havendo, foi lavrada a presente ata, que após lida e aprovada, foi assinada pelo(a) Presidente(a) da Comissão Examinadora.

Prof. Dr. FELIPE FERNANDES FANCHINI

Marcelo Velludo de Souza Prado

Tomografia de Hamiltoniano para Sistemas de Dois Qubits Utilizando Aprendizado de Máquina

Dissertação apresentada como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação, junto ao Programa de Pós-Graduação em Ciência da Computação, do Instituto de Biociências, Letras e Ciências Exatas da Universidade EstaduaPaulista "Júlio de Mesquita Filho", Câmpus de São José do Rio Preto.

Comissão Examinadora

Prof. Dr. Felipe F. Fanchini
UNESP - Câmpus de Bauru
Orientador

Convidado Externo
Instituição

Convidado Interno
UNESP - Câmpus de Bauru

Bauru
26 de janeiro de 2023

Agradecimentos

Ao meu amigo, Dr.Bernardo Dias Batista Guimarães, por me apoiar em meu sonho mais ousado.

Ao meu orientador, pelos ensinamentos e especialmente pela sua paciência, compreensão e apoio por cada fase superada durante este projeto.

Aos meus familiares, e ao programa de pós-graduação, pelo acolhimento durante os anos de mestrado.

Resumo

Para sistemas quânticos fechados, a energia é uma observável que pode ser representada por uma matriz hermitiana chamada de hamiltoniano. Com a hamiltoniano que descreve o sistema quântico, podemos determinar a sua dinâmica através da equação de Schrödinger, que diz a taxa de variação de um estado $|\psi(t)\rangle$ com respeito a um determinado instante t. Resolvendo a equação para um estado inicial t consegue-se determinar a evolução do sistema. Utilizando o hamiltoniano que descreve um sistema de duplos pontos quânticos, e medindo as observáveis locais, aplicamos o modelo de aprendizado de máquina Extra-Trees com o objetivo de realizar a tomografia completa do hamiltoniano. Ficou demonstrado a efetividade da solução em diversas situações e, afim de obter diretrizes para auxiliar o físico experimental, realizamos analises exploratórias com o intuito de melhorar o entendimento sobre o problema de tomografia de hamiltoniano.

Palavras-chave: Informação quântica. Computação quântica. Tomografia de hamiltoniano. Aprendizado de máquina. Regressão. Extra-Trees.

Abstract

For closed quantum systems, energy is an observable that can be represented by a Hermitian matrix called the Hamiltonian. With the Hamiltonian that describes the quantum system, we can determine its dynamics through the Schrödinger equation, which tells the rate of change of a state $|\psi(t)\rangle$ with respect to a given instant t. By solving the equation for an initial state we can determine the system's evolution. Using the Hamiltonian that describes a system of double quantum dots, and measuring the local observables, we applied the Extra-Trees machine learning model in order to perform a complete tomography of the Hamiltonian. We demonstrated the effectiveness of the solution in several situations and, in order to obtain guidelines to help the experimental physicist, we performed an exploratory analysis to improve the understanding of the Hamiltonian tomography problem.

Keywords: Quantum information. Quantum Computing. Hamiltonian tomography. Machine learning. Regression. Extra-Trees.

Lista de ilustrações

Lista de abreviaturas e siglas

QD	Pontos Quântico
DQD	Duplos Pontos Quânticos
ML	Aprendizado de Máquina (Machine Learning)
S-T_0	Singleto-Tripleto
SVM	Máquina de Vetores de Suporte (Support Vector Machine)
LightGBM	Light Gradient Boosting Machine
GBDT	Gradient Boosting Decision Tree
MSE	Média do Erro ao Quadrado (Mean Squared Error)
MAE	Média do Erro Absoluto (Mean Absolute Error)
MAPE	Média Percentualdo Erro Absoluto (Mean Absolute Percentage Error)
CART	Árvore de Regressão e Classificação (Classification and Regression Trees)
R2	R-Quadrado(R-Squared)
Extra-Trees	Árvore extremamente randomizada (Extremly Randomized Trees)

Sumário

1 Introdução

O desenvolvimento do processador de informação quântica vêm introduzindo técnicas diferenciadas que estão abrindo o caminho para realizarmos o denominado *avanço tecnológico quântico* (ARCARI et al., 2014; OKAMOTO et al., 2009). Entre esses avanços, spin qubits em pontos quânticos (QDs) estão certamente entre um dos sistemas mais marcantes por causa de sua escalabilidade e miniaturização (IMAMOGLU et al., 1999; GREILICH et al., 2011; JEFFERSON et al., 2002). Vale frisar no entanto que a leitura e controle de um único spin em QDs é ainda um grande desafio, de forma que o desenvolvimento de sensores de carga e campos de controle robustos, que permitem suas observações e controle, é um campo de pesquisa de grande interesse (PETTA et al., 2005; TAYLOR et al., 2007).

Mais recentemente, um progresso significativo na implementação de um esquema alternativo em duplos pontos quânticos (DQDs) foi alcançado (MAUNE et al., 2012). Nesta abordagem, os estados singleto e tripleto de dois elétrons são usados para representar um qubit lógico e, por meio deste aparato, um experimento inovador mostra que um conjunto universal de portas quânticas pode ser alcançado (SHULMAN et al., 2012). É baseando-se nestas implementações que iremos desenvolver nossos estudos.

Nosso objetivo neste projeto é, utilizando aprendizado de máquina, determinar a intensidade de acoplamento entre os qubits observando os mesmos localmente. Esta tarefa é relevante pois determinar o acoplamento experimentalmente é uma tarefa geralmente difícil de forma que métodos numéricos poderão auxiliar os experimentais neste procedimento. Esta proposta tem como objetivo, portanto, a aplicações de aprendizado de máquina, que é um campo da inteligência artificial, em física quântica, sendo o foco, a estimativa da constante de acoplamento entre os qubits.

Vale frisar que a inteligência artificial é um amplo campo de pesquisa que visa simular a inteligência humana usando algoritmos que são programados para realizar habilidades humanas. Atualmente, o estudo da inteligência artificial abrange vários ramos, como aprendizado de máquina (ML), aprendizado por reforço e aprendizagem profunda. O aprendizado de máquina recentemente se tornou uma ferramenta crucial para extrair informações úteis com a rápida expansão na formação de dados. Atualmente é utilizado em várias áreas de pesquisa como medicina, química, biologia e física (JORDAN; MITCHELL, 2015).

Este projeto pretende estudar a dinâmica de QDs com a ajuda de instrumentos de ML. Para tal, na próxima seção 1.1 discutimos brevemente os dois trabalhos principais que serão utilizados em nossos estudos. Na sequência, exploramos a fundamentação teórica sobre o assunto que foi dividido em duas seções, física quântica 2.1 e aprendizagem de máquina 2.2. A seção 3 aborda a metodologia utilizada, na seção 4 abordamos os resultados da proposta e,

em sequência, seguimos com a conclusão 5.

1.1 Trabalhos Relacionados

Para a física quântica, tarefas preditivas como estimar fidelidade, verificar emaranhamento e medir correlações são essenciais para construção, calibragem e controle de sistemas quânticos (HUANG; KUENG; PRESKILL, 2020)(GRAY et al., 2018). Os parâmetros para as tarefas preditivas são feitas utilizando medidas. Uma medida informativa na mecânica quântica é destrutiva (a função de onda colapsa) e apenas leva a uma saída probabilística (Lei de Born). Então, vários experimentos idênticos são necessários para estimar de maneira acurada um único parâmetro de um sistema quântico desejado (HUANG; KUENG; PRESKILL, 2020). Em suma, reconstruir uma descrição completa do sistema quântico composto por n elementos (pontos quânticos, qubits, etc) necessita um número de repetições nas medidas que cresce exponencialmente com n, assim como a memória e poder computacional necessários (HUANG; KUENG; PRESKILL, 2020)(GRAY et al., 2018). Neste sentido, as técnicas de aprendizado de máquina vem surgindo como um ferramenta de suma importância.

Por exemplo, em (GRAY et al., 2018) foi proposto um esquema para medir o emaranhamento entre subsistemas arbitrários com auxílio de um esquema de aprendizado de máquina. O objetivo foi alcançado sem conhecimento a priori sobre o estado inicial. Em (HUANG et al., 2022) foi desenvolvido um protocolo para determinar o grau de emaranhamento através de informações locais e aprendizado de máquina, utilizando uma rede neural totalmente conectada (FCNN) e uma rede neural recorrente de memória de termo longo e curto (LSTM). Os dois artigos evidenciam que tal medida para ser obtida necessitaria de tomografias completas do estado, que são dispendiosas. Ainda expõem as dificuldades em se determinar o grau de emaranhamento, visto que não é uma medida que pode ser observada diretamente.

Nossa abordagem difere de (GRAY et al., 2018) e (HUANG et al., 2022) em seu objetivo. Nos artigos em questão, o aprendizado de máquina é utilizado para predizer a dinâmica do emaranhamento e em nosso trabalho, o objetivo é determinar os parâmetros que compõem o hamiltoniano. O processo de reconstrução, ou fazer a tomografia do hamiltoniano que atua no sistema, é um assunto que tem ganhado destaque recentemente por abordar diretamente um problema da computação quântica dos termos recentes. A tomografia de hamiltoniano é utilizada em diversas situações. Por exemplo, em (GREENAWAY et al., 2022) a tomografia do hamiltoniano é utilizada para avaliar a adequação dos qubits transmon com frequências e interações fixas a fim de realizar simulações quânticas de sistemas com spin. Além disso, em (SIVA et al., 2022) os autores citam tal relevância em relação a caracterização de portas lógicas quânticas, entes fundamentais para todo e qualquer processo de computação quântica. O objetivo de (SIVA et al., 2022) é reconstruir completamente o hamiltoniano em um sistema quântico utilizando transmons e medidas fracas que não perturbam o sistema.

Neste projeto de pesquisa, temos objetivos similares à aqueles propostos por (GREENAWAY et al., 2022) e (SIVA et al., 2022), onde faremos uso das técnicas de aprendizagem de máquina a fim de tomografarmos o hamiltoniano típico de duplos pontos quânticos. Este projeto, toma como base o manuscrito (CASTELANO; FANCHINI; BERRADA, 2016), onde desenvolve-se um modelo teórico para descrever a dinâmica dissipativa de qubits singleto-tripleto (S-T_0) em pontos quânticos de GaAs. Apesar do foco neste trabalho ser diferente, acreditamos que uma breve explanação é relevante para a contextualização do problema. No artigo em questão os autores, usando a concorrência (uma medida de emaranhamento) obtida experimentalmente como guia, mostraram que cada qubit lógico pode ser descrito através do acoplamento com seu próprio ambiente, pois o efeito de decoerência pode ser descrito por canais de defasagem independentes. No artigo, dada a correta descrição do ambiente, foi estudado a dinâmica da concorrência em função da temperatura, a constante de acoplamento entre o sistema e o ambiente, do tempo de preparo do estado inicial e do acoplamento entre o qubits. O artigo em questão demonstrou que, embora a redução da constante de acoplamento do ambiente modifique a dinâmica de emaranhamento, a temperatura surge como uma variável crucial e uma variação de milikelvins modifica significativamente a geração de estados emaranhados. Além disso, o artigo mostrou que o acoplamento entre os qubits, juntamente com o tempo de preparação, afeta fortemente a dinâmica do emaranhamento. Vale frisar que em nossos estudos, para gerar a base inicial para escolha do modelo, tratamos o sistema como fechado, desconsiderando o acoplamento com o meio ambiente. Após a escolha do modelo e a demonstração do potencial das ferramentas de machine learning, aumentamos a complexidade do problema e passamos a considerar o sistema aberto.

Além disso, neste trabalho também levamos em consideração o manuscrito (FANCHINI et al., 2021), onde os autores aplicaram métodos de aprendizado de máquina em sistemas quânticos abertos considerando o estudo da caracterização dos efeitos da memória que surgem dinamicamente ao longo da evolução temporal dos sistemas abertos, à medida que interagem com o ambiente circundante. Neste trabalho os autores mostraram que, usando técnicas de aprendizado de máquina, em particular, algoritmos de máquina de vetores de suporte (SVM), é possível estimar o grau de não Markovianidade em dois modelos de sistemas abertos. A abordagem, segundo os autores, tem o potencial de ser experimentalmente viável para estimar o grau de não markovianidade, uma vez que requer uma única ou no máximo duas rodadas de tomografia do estado quântico.

Neste trabalho iremos fazer uso da metodologia desenvolvida em (FANCHINI et al., 2021) a fim de estudarmos o sistema físico dos pontos quânticos. Em suma, como já dito anteriormente, o objetivo é utilizar os resultados das medidas locais como características, a fim de determinamos a constante de acoplamento entre os qubits fazendo o uso da aprendizagem de máquina.

2 Fundamentação Teórica

2.1 Física Quântica

A física tem como objetivo estudar a dinâmica de sistemas, sendo a dinâmica o estado da partícula em um certo instante de tempo. Assim, dado um estado inicial, e com as leis que regem o sistema, conseguimos determinar o estado da partícula dado um instante de tempo, em outras palavras, estudar sua evolução temporal. Com base neste fato, a proposta do trabalho em questão, visa estudar as dinâmicas dos sistemas quânticos de dois qubits com o intuito de estimarmos o acoplamento entre eles.

Em sistemas quânticos, o estado da partícula é dado pela função de onda. A função de onda na mecânica quântica descreve o estado quântico de um sistema de uma ou mais partículas, contendo informações sobre o sistema. As informações podem ser quantidades associadas às partículas tais como a energia, momento angular, ou mesmo a posição, que podem ser obtidos a partir da função de onda por meio de operações matemáticas que descrevem a sua interação com os dispositivos de observação. Assim, a função de onda é uma entidade central na mecânica quântica.

Para estudar a dinâmica em um sistema quântico temos que, dado um estado inicial $|\Psi(0)\rangle$ e a configuração do sistema - campos externos, por exemplo - podemos obter $|\Psi(t)\rangle$. Na eq. (1) temos a equação de Schrödinger, sendo i o numero imaginário, $\hbar$ a constante de Planck e H um operador Hermitiano conhecido como hamiltoniano. O termo Hermitiano se refere a um grupo especial de matrizes que se enquadram na definição $A^\dagger = A$, onde A é uma matriz quadrada (n-por-n), e o operador $\dagger$ é o transposto complexo conjugado da matriz. Neste caso, o Hamiltoniano pode ser interpretado como o total da energia do sistema, além de balizar a evolução do estado ao longo do tempo, que é determinada pela equação de Schrödinger:

$$i\hbar \frac{d|\psi(t)\rangle}{dt} = H|\psi(t)\rangle \tag{1}$$

onde, a solução pode ser escrita como:

$$|\psi(t)\rangle = U(t)|\psi(0)\rangle. \tag{2}$$

Aqui, $|\psi(0)\rangle$ é o estado inicial do sistema, $|\psi(t)\rangle$ o estado no tempo t e, $U(t)$ é um operador unitário chamado de matriz evolução temporal. O operador $U(t)$ é definido pelo hamiltoniano do sistema que, para H independente do tempo, pode ser descrito por:

$$U(t) = e^{-iHt}, \tag{3}$$

onde aqui consideramos $\hbar = 1$. Rescrevendo na forma de operador densidade (uma maneira alternativa para descrever o estado quântico), temos:

$$\rho(t) = U(t)\rho(0)U^{\dagger}(t), \tag{4}$$

onde $\rho(t) = |\psi(t)\rangle\langle\psi(t)|$. Enfatizando que estamos utilizando a notação bra-ket, comum na computação quântica, a qual o *ket* $|\psi(t)\rangle$ é um vetor coluna e *bra* $\langle\psi(t)|$ é o transposto complexo conjugado de $\psi(t)$, resultando em um vetor linha.

A configuração do sistema no caso estudado inicialmente é representada pelo hamiltoniano da Eq. (5), lembrando que estamos tratando um sistema de dois qubits. O hamiltoniano efetivo para o sistema de dois duplos pontos quânticos, de forma simplificada, pode ser descrito como:

$$H = a(\sigma_z^1 \otimes \mathbf{I}) + b(\mathbf{I} \otimes \sigma_z^2) + J\ (\sigma_z^1 \otimes \sigma_z^2) + c(\mathbf{I} \otimes \sigma_x^2) + d(\sigma_x^1 \otimes \mathbf{I}) \tag{5}$$

onde, $\sigma_{x,z}$ são as matrizes de Pauli,

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \tag{6}$$

$\mathbf{I}$ é a matriz identidade, e o índice representa os qubits 1 e 2. O simbolo $\otimes$ é a operação tensorial. Vale salientar também que os termos multiplicado por a, b, c e d, correspondem a campos externos e a variável J está relacionada a constante de acoplamento entre os qubits. O objetivo inicial deste trabalho foi descobrir o valor de J, fazendo uso de aprendizado de máquina através da observação dos qubits de maneira individual e localmente. Como explorado em (CASTELANO; FANCHINI; BERRADA, 2016), este hamiltoniano é capaz de implementar o controle quântico universal, podendo levar um estado inicial para qualquer posição no espaço de Hilbert. Desta maneira, o Hamiltoniano pode ser usado para manipular os estados quânticos de um sistema de uma maneira precisa e flexível, podendo executar tarefas como a simulação quântica, computação quântica, e comunicação quântica.

Com o hamiltoniano final descrito, podemos retornar a discussão sobre as medidas quânticas, que é o meio pelo qual observamos o estado quântico. Como foi explorado, quando efetua-se a medida em um sistema quântico, o estado do sistema é reduzido de uma superposição de múltiplos estados possíveis para um único estado definido, o que chamamos de colapso do sistema. De forma geral, medidas são representadas em mecânica quântica por um operador chamado de observável, que atua no estado de um sistema e produz uma saída correspondente ao seu auto-valor. Exemplos de observáveis em mecânica quântica incluem posição, momento, energia, spin, e momento angular. Cada observável é associado com um operador correspondente, e os auto-valores do operador representam os valores possíveis que as observáveis podem assumir. Por exemplo, o operador de posição possui auto-valores das possíveis posições das partículas, e o Hamiltoniano que é o operador que representa a energia, possui auto-valores que representam os possíveis níveis de energia de um sistema quântico.

No caso apresentado neste manuscrito, vamos utilizar como observáveis os operadores de Pauli, relembrando que o estado evolui com o tempo, como representado pela eq. 4. Sendo assim, serão as matrizes de Pauli que nos trarão informações sobre o estado, em diversos instantes de tempo. Como estamos trabalhando em um sistema de dois qubits, podemos extrair as medidas de ambos, se atentando também que estaremos focando em medidas locais. Em outras palavras, as observáveis que iremos utilizar serão $(\sigma_k \otimes \mathbf{I})$ e $(\mathbf{I} \otimes \sigma_k)$, com $k = \{x, y, z\}$.

Ainda sobre o método de medida utilizado por este manuscrito, como será demonstrado na metodologia, foi necessário apenas medidas locais para cumprir o objetivo final de determinar as variáveis do hamiltoniano. Em mecânica quântica, medida local é a medida performada em um subsistema vindo de um sistema quântico maior. Traduzindo para o nosso sistema, como estamos trabalhando com um sistema de dois qubits, uma medida local é aquela que é realizada em apenas um qubit. Ou seja, um único qubit faz o papel de nosso subsistema.

O valor esperado dos qubits, quando se trata de medidas locais, pode ser calculado utilizando os traços das matrizes seguindo a equação abaixo:

$$\begin{aligned} O_k^1(t) &= \mathrm{Tr}\ \{(\sigma_k \otimes \mathbf{I})\rho(t)\} \\ O_k^2(t) &= \mathrm{Tr}\ \{(\mathbf{I} \otimes \sigma_k)\rho(t)\}, \end{aligned} \tag{7}$$

onde $k = \{x, y, z\}$ representando cada matriz como descrito na equação 6. A outra traço é a somatória dos elementos da diagonal, sendo σ_k as matrizes de Pauli. É importante notar que a partir das duas equações expostas temos na verdade seis, pois existem 3 matrizes de Pauli, e necessitamos das 3 para fazer a tomografia completa em todos as bases do sistema.

Nosso objetivo então é, fazendo o uso das observáveis da eq. 7 como medidas do sistema descrito pelo hamiltoniano, determinar a constante de acoplamento do hamiltoniano e os campos externos. A maneira como vamos determinar os seis alvos, é por aprendizado de máquina. Neste sentido, as observáveis descritas na equação 7 são as características, enquanto os seis alvos são os parâmetros do hamiltoniano. Alvos e características são denominações vindas de aprendizado de máquina, que é a forma como buscamos solucionar o problema tratado no manuscrito e que vamos abordar na próxima seção.

2.2 Aprendizado de Máquina

O Aprendizado de Máquina (ML) é um ramo multidisciplinar dentro da computação, que faz uso de ferramentas estatísticas e matemáticas, de modo à fornecer às máquinas a capacidade de aprender de forma autônoma. Para que máquinas aprendam utiliza-se experiências, observações ou análises de padrões dentro de um determinado conjunto de dados sem à necessidade de programação explícita para a solução do problema (SCHMIDT et al., 2019).

Os algoritmos de aprendizado de máquina são aplicados para tarefas de classificação, regressão e agrupamento. Os métodos de aprendizado de máquina geralmente precisam passar

por um treinamento em um conjunto especifico de dados. Após o treinamento, o algoritmo é capaz de fazer generalizações em dados não vistos durante o treinamento (KADAM, 2020). Os algoritmos de aprendizado de máquina podem ser divididos em 3 categorias:

- Supervisionados: No aprendizado supervisionado, usamos dados rotulados para os dados de treinamento, ou seja dados que sabemos a resposta para a variável que deseja-se prever. Os dados alimentam o algoritmo e são usados para treinar o modelo. Depois que o modelo é treinado com base nos dados rotulados, pode-se usar dados não rotulados no modelo e obter uma nova resposta. No final do treinamento, o que se têm é uma equação, ou um objeto chamado árvore de decisão, que busca descrever com base nos dados das características, a variável alvo.

- Não-Supervisionados: No aprendizado não supervisionado, os dados de treinamento são não-rotulados. Sem o aspecto de dados rotulados, a resposta não pode ser guiada para o algoritmo, que é de onde o termo não supervisionado se origina. Mesmo sem um rótulo nos dados, os métodos supervisionados também passam por uma fase de treinamento como os supervisionados. O modelo treinado pesquisa por um padrão nos dados. O padrão buscado depende do algoritmo utilizado e sua forma de medir semelhança, por exemplo; utilizando a distância euclidiana entre dois pontos de dados, onde pontos mais próximos possuem maior semelhança e pontos mais distantes são menos semelhantes. Geralmente esta estratégia é utilizada para definir grupos, ou traçar perfis semelhantes, em outras palavras, verificar a possível existência de um agrupamento natural dos dados. Os métodos deste conjunto, como possuem a natureza de agrupar os dados semelhantes, são utilizadas para melhorar o conhecimento dos dados, ou por falta de acesso a dados rotulados. Importante notar que a rotulação de dados, na prática, pode ser um trabalho demorado e custoso.

- Aprendizagem por Reforço: O aprendizado por reforço tenta resolver as tarefas por tentativa e erro. Neste tipo de aprendizado normalmente se tem um ambiente e, de acordo com o objetivo, define-se recompensas para o modelo tentar aprender por conta própria. É uma estratégia muito utilizada para treinar robôs a fazer tarefas como andar, ou pegar outros objetos.

Para abordar o problema tratado neste projeto, serão utilizados algoritmos supervisionados, pois a nossa base de dados é rotulada com as equações demonstradas em 2.1, o passo a passo será descrito em 3.3.

A categoria supervisionada pode ser dividida em dois tipos com base na tarefa realizada. A primeira tarefa é a classificação, que tem como objetivo separar os dados em classes distintas. O segundo tipo, é chamado de regressão, que tem como objetivo modelar variáveis continuas (RAY, 2019). Como o principal objetivo deste projeto busca variáveis continuas, e não uma

classe, na perspectiva de aprendizado de máquina, o problema trata-se de regressão, portanto será utilizado um algoritmo para tal fim.

O projeto em sua fase de seleção de modelos, explorou diversos tipos diferentes de modelos de regressão, com base na figura 6 (Figura retomada no capítulo 3), nota-se que os algoritmos de ML classificados como algoritmos de conjunto, se destacam pelo bom desempenho. De forma breve, modelos conjunto (ou utilizando o nome inglês, ensemble), tratam de modelos ML que agrupam as respostas de diversos modelos, formando apenas uma resposta final. Sabendo que o algoritmo escolhido é um algoritmo de conjunto, na seção 2.2.2 será abordado a classe de algoritmos e a motivação para sua escolha. Antes do aprofundamento nos métodos, será tratado a definição de regressão.

2.2.1 Regressão

Nesse trabalho se assume um típico problema de regressão. Com um potencialmente infinito de espaço de entrada X, o objetivo é induzir uma função $\hat{f} : X \rightarrow R$ que aproxima uma função verdadeira desconhecida f,

$$MSE\ (\hat{f}) = E\,[(\hat{f} - f)^2]. \tag{8}$$

A função $\hat{f}$ é obtida pelo algoritmo de indução (o aprendiz, modelo, algoritmo de aprendizado de máquina), aplicado nos dados. Os dados consistem de um conjunto finito de n exemplos na forma $\{(x_1, f(x_1)), \ldots, (x_n, f(x_n))\}$, onde os termos que formam cada par representam; o conjunto de características (x), e o resultado esperado para o respectivo conjunto de característica $f((x))$. Neste trabalho também podemos fazer uso de outros nomes para x e $f(x)$; x pode ser chamado de caracteristicas ou variáveis independentes, $f(x)$ pode ser chamado de variável dependente, alvo ou simplesmente y. A função $\hat{f}$ é chamada de modelo ou preditor. A qualidade da aproximação da função $\hat{f}$ em relação à função real desconhecida f é definida pela generalização do erro, que pode ser definida pela média do erro ao quadrado (MSE), presente na eq. 8. No entanto, existem outras opções de generalizações de erro para predições numéricas, a generalização apenas necessita medir a distancia entre a função real e a aproximação $\hat{f}$, apesar da maioria dos trabalhos utilizarem MSE (MENDES-MOREIRA et al., 2012). Podemos citar como alternativas a média absoluta do erro (MAE), ou a média absoluta percentual do erro (MAPE), que podem ser empregadas em situações especificas, por exemplo para evitar distorções do termo ao quadrado da eq. 8. Daremos enfoque nas métricas na seção (3.4).

No âmbito do problema tratado neste manuscrito, como foi demonstrado na seção 2.1, será induzida 5 funções através do algoritmo Extremly Randomized Trees (Extra-Trees). Sendo uma função para cada variável alvo: J_1, J_2, J_{12}, $\Delta B_{z,1}$ e $\Delta B_{z,2}$, presentes na equação 9. Como descrito nesta seção, as funções são induzidas através do conjunto de característica

No caso tratado neste manuscrito, o nosso conjunto de características são as observáveis que foram apresentadas na seção 2.1, sua respectiva equação está demonstrada em 7.

Com a introdução de aprendizado de máquina e a definição de regressão apresentados, e feitas as correlações com o objetivo final deste projeto, na próxima seção será abordado a classe do algoritmo escolhido - métodos de conjunto. Nesta seção foi falado brevemente sobre o erro e foi dada uma forma de medi-lo com a equação 8. O erro apresentado é, na verdade, uma soma que pode ser decomposta em 3 partes: erro irredutível, Bias, Variância. Métodos de conjunto surgiram como forma de diminuir os dois erros causados pela algoritmo de aprendizado de máquina, ou pelos dados, o Bias e a Variância.

2.2.2 Métodos de Conjunto

Os modelos de aprendizado de máquina sofrem com problemas relacionados a compensação Bias-Variância, o que pode resultar em uma baixa performance do modelo. Visando corrigir problemas da compensação bias-variância surgiu os métodos de conjunto (ensemble) (MIENYE; SUN, 2022). Métodos de conjunto possuem atualmente bons desempenhos para números consideráveis de aplicações e, notoriamente, ganhou seu espaço em competições de problemas práticos como o Netflix Prize (BELL; KOREN, 2007; BELL; KOREN; VOLINSKY, 2010) e em muitos outros contextos como será demonstrado nesta seção, fazendo com o que este estilo de aprendizado faça parte de todo praticante de machine learning nos dias atuais (MERENTITIS;DEBES, 2015). Tais métodos, como o próprio nome sugere, aumentam a performance das predições combinando o treinamento de múltiplos modelos e de suas predições (SAGI; ROKACH, 2018), formando um conjunto. Vamos explorar o problema da compensação Bias-Variância para entendermos como a ideia para este tipo de aprendizado surgiu, e depois vamos prosseguir com aprendizado com conjunto. Com o Bias-Variância conseguiremos entender melhor o sucesso deste algoritmo e o motivo pelo qual foi utilizado neste trabalho.

2.2.2.1 Compensação Bias-Variância

Uma maneira útil de analisar a habilidade de predição de um algoritmo é examinando as diferentes fontes de erro que podem ocorrer. Além do erro irredutível, inerente ao próprio conjunto de dados ou ao próprio problema tratado, temos também dois tipos de erros que podem ocorrer de acordo com o algoritmo utilizado. Estes erros podem ser controlados pelo algoritmo e são chamados de Bias e Variância (FENG et al., 2022).

Com base no que foi visto na seção 2.2.1, onde definimos regressão, de forma geral estamos induzindo uma função $\hat{f}$ para tentar encontrar uma função f desconhecida. Na escolha de um algoritmo de aprendizado de máquina para induzir $\hat{f}$, é assumido que o algoritmo é capaz de chegar à função f desconhecida, o erro desta escolha é chamado de bias. Como exemplo, assuma que o algoritmo em questão consiga ajustar uma função de segundo grau formando uma parábola. Então temos que, apesar do algoritmo tentar encaixar a sua parábola de uma

forma ótima nos dados que formam a função verdadeira f, a caso a função verdadeira f não seja de fato uma parábola, vai haver um erro chamado de Bias. Por outro lado, se permitirmos que nossa função $\hat{f}$ assuma um grau próximo de infinito, a sua forma será flexível suficiente para alcançar qualquer ponto dos dados com o devido treinamento para "encaixá-la". Como sempre trabalha-se com dados finitos, por melhor que a curva possa ser flexível para alcançar todos os pontos de forma ótima, ela será apenas perfeitamente adaptada aos dados que lhe foi apresentado, tendo assim uma variância em dados não vistos. Este erro é chamado de variância. A escolha do algoritmo de aprendizado de máquina é feito de modo a equilibrar o bias e a variância para obter uma melhor performance prática.

Na prática, existem várias maneiras de controlar o bias ou a variância de um algoritmo. Como foi utilizado no exemplo, pode-se permitir que um algoritmo considere uma função mais complexa para mapear os dados, ou quando se utiliza árvore de decisão, pode-se permitir que sua profundidade aumente. Também deve-se levar em conta incrementar a quantidade de dados para decrementar a variância. Nota-se que algoritmos que predizem bem, são aqueles que controlam a compensação entre o bias e a variância de forma a minimizar o erro total, apesar de sua fonte (FENG et al., 2022). Como foi visto no exemplo dado nesta seção, o bias e variância atuam como uma troca, por isso o nome "compensação"(traduzido do inglês, Trade-off). O aprendizado por conjunto foi desenvolvido como uma forma de romper com essa compensação. De fato, muito da boa performance prática do aprendizado por conjunto é devido ao fato deste rompimento. Na próxima seção será explorado como o aprendizado por conjunto consegue romper a compensação, e o funcionamento do paradigma de aprendizado que o levou a uma boa performance na resolução do problema em questão, como será demonstrado na seção 4.1.

2.2.2.2 Aprendizado com Conjunto

Aprendizado por conjunto é um termo que agrega métodos que combinam múltiplos modelos para fazer a decisão, tipicamente em tarefas de aprendizado supervisionado. A premissa principal de um método de aprendizagem com conjunto é: combinando múltiplos modelos, o erro de um único modelo provavelmente será compensado pelos outros modelos do conjunto, resultando em uma melhora de desempenho em relação à utilizar um modelo único (SAGI; ROKACH, 2018; MIENYE; SUN, 2022; KUMAR; KAUR; GOSAIN, 2022).

A combinação de modelos é intuitiva, no nosso cotidiano passamos por diversas situações onde utilizamos a ideia base do aprendizado por conjunto. Por exemplo, em um sistema da votos, estamos tentando utilizar a sabedoria da maioria para solucionar um problema, sejam em eleições presidenciais ou em um conselho. Um sistema de votação funciona para problemas de classificação, e de fato o voto majoritário é utilizado (MOREIRA et al., 2012) em aprendizado por conjunto. Por outro lado, em regressão poderiamos supor o seguinte contexto: temos um pote com algumas bolas, e o objetivo é calcular a quantidade de bolas no pote. Podemos

considerar a possibilidade de contar as bolas no pote dependendo da quantidade, caso fosse inviável, uma outra opção seria utilizar o aprendizado por conjunto perguntando para algumas pessoas um palpite sobre quantas bolas existem no pote. Poderíamos então tirar uma média aritmética para conseguir um palpite final mais próximo. De fato está também é um estratégia utilizada em aprendizado de máquina, como é o caso do algoritmo Extra-Trees utilizado neste projeto. Aprendizado de conjunto, trata de construir um conjunto de aprendizes bases e aglutinar suas respostas com alguma regra (Voto majoritário, média aritmética), de forma que a variância total e o bias é reduzido (RINCY; GUPTA, 2020). É possível combinar qualquer algoritmo de machine learning, porém métodos com um menor custo computacional e que possuem variância são mais comuns.

Em 2014 o artigo (FERNÁNDEZ-DELGADO et al., 2014) comparou 179 classificadores de 17 famílias usando 121 base de dados, assim como alguns desafios práticos, e atestou que random forest, que faz parte dos métodos de conjunto, obteve desempenho superior na maior gama de problemas em relação aos demais. Foram explorados os motivos pela eficiência do aprendizado por conjunto, em (DIETTERICHL, 2002; POLIKAR, 2006; RINCY; GUPTA, 2020), foram expostas as seguintes conclusões:

1) Evita Overfitting: algoritmos de aprendizado, busca a melhor hipótese no espaço. Quando se tem apenas uma pequena quantidade de dados disponível, o algoritmo tem a tendência para encontrar uma hipótese que prediz todo o data set de treino perfeitamente, enquanto faz predições pobres para instâncias não vistas no treino. Em outras palavras, a melhor hipótese para o data-set de treinamento, pode não condizer com a melhor hipótese para a situação real. Tal comportamento do modelo é chamado de overfitting. Como métodos ensemble uniatravés de uma regra as hipóteses de vários modelos, as chances do modelo escolher apenas a melhor hipótese contida nos dados de treino e não a melhor hipótese que soluciona o problema é reduzida.

2) Vantagem em Pontos de confusão: aprendizes (learners) geralmente possuem uma aleatoriedade vinda do algoritmo (ex: Ponto de inicialização aleatório), e os aprendizes fazem uma busca local que é suscetível a mínimos locais, mesmo com o resguardo de uma base de dados grande o suficiente. Com os aprendizes trabalhando em conjunto, e com a aleatoriedade, mesmo se algum aprendiz ficar preso em um ponto de ótimo local, os outros aprendizes podem compensar e com isso temos uma melhor representação da função que estamos buscando.

3) Representação: a hipótese ótima pode estar fora do espaço de qualquer modelo único. Combinando diferentes modelos, o espaço de hipótese pode ser estendido e então, pode-se encontrar uma melhor hipótese que seja mais compatível com o problema. A figura 1 demonstra como a união de varias hipóteses (a), pode resultar em uma hipótese mais eficaz em (b). Outro exemplo de problema na representação é de quando os dados

utilizados para o treino não representam bem situações reais, algoritmos de ensemble com a extensão do espaço de hipótese também pode ajudar neste problema, aumentando a generalização (MERENTITIS; DEBES, 2015).

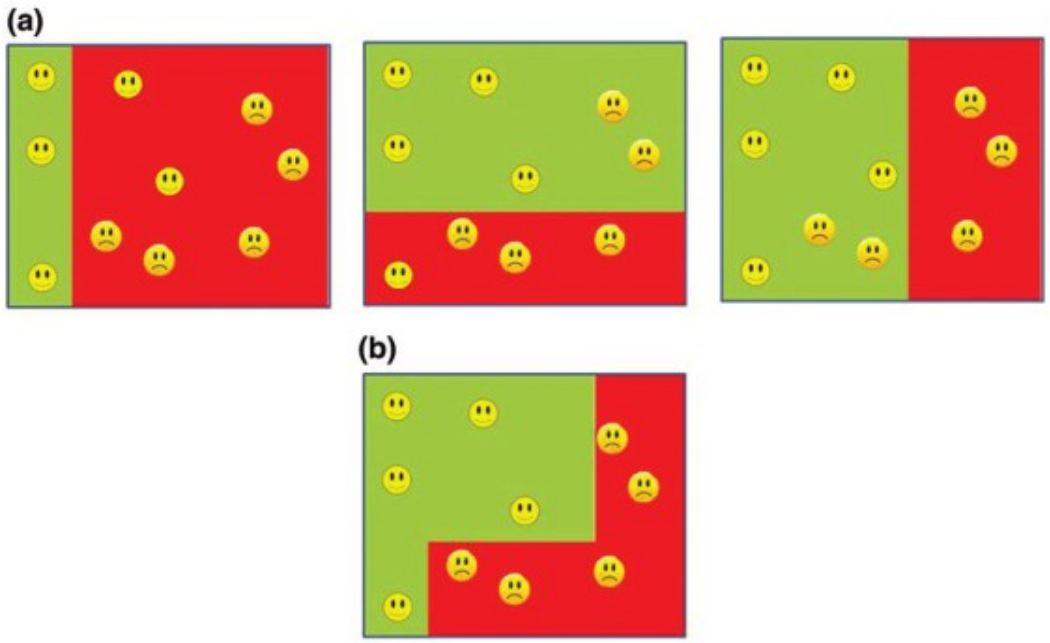

Figura 1 – Exemplo de aperfeiçoamento de hipótese em métodos ensemble (SAGI; ROKACH, 2018).

Aprendizado por conjunto é o processo que utiliza um conjunto de modelos, cada um deles aplicando seu próprio processo de treinamento de forma independente, para após o treinamento os seus resultados poderem ser integrados, formando uma predição final única (MENDES-MOREIRA et al., 2012). A chave para ensemble é: aprendizes fracos podem formar um aprendiz forte (SCHAPIRE, 1990b; SCHAPIRE, 1992; SCHAPIRE; SINGER, 1999; HASTIE; TIBSHIRANI; FRIEDMAN, 2009a), onde aprendiz fraco se trata de um aprendiz que performa apenas razoavelmente melhor que uma predição completamente aleatória (MENDES-MOREIRA et al., 2012). Importante notar que apesar de serem vários métodos muitas vezes iguais, existem diversas estratégias que tornam as respostas de um método diferente das respostas do outro método, de forma a garantir uma variação das hipóteses ótimas entre os modelos. Vamos aprofundar o tema agora e buscar por opções de como unir os modelos de forma a garantir a redução da compensação bias-variância.

2.2.3 Impulsionar (Boosting)

Impulsionar (Boosting) faz referência a sequência de aprendizes. Começando com um modelo base fraco, ele é treinado iterativamente, cada iteração adicionando à previsão do

modelo anterior para produzir um modelo forte. Normalmente é utilizado o mesmo modelo base como uma árvore de decisão. O modelo base é treinado da seguinte forma: na primeira iteração o modelo é treinado sem nenhum peso nas amostras do conjunto (dataset) de treino, na sequencia as amostras são ajustadas com um peso, de acordo com o erro que o primeiro modelo cometeu, as amostras com os pesos modificados são utilizadas para alimentar o próximo modelo. Este processo iterativo é repetido conforme a necessidade do problema. Nota-se que as amostras que são difíceis do modelo capturar, vão recebendo mais peso com o passar das iterações até que o modelo consiga capturá-la de forma adequada (MIENYE et al., 2022). É um procedimento simples capaz de aumentar a performance do modelo base consideravelmente (HASTIE; TIBSHIRANI; FRIEDMAN, 2009b). Impulsionar pode ser visto como uma forma de agregar modelos verticalmente.

Impulsionar tem maior foco em reduzir mais o bias do que a variância (OZA; TUMER, 2008). Como amostras classificadas erradas recebem maior peso, fazendo com que o aprendiz foque nestas amostras, então se o aprendiz está mais tendenciado a errar em uma amostra, essa amostra vaireceber mais peso, e então, o bias será ajustado. No entanto, esse mesmo modo iterativo de aprendizado, faz com que esse aprendizado seja inviável para dados com ruído, pois o peso dado para dados com ruídos será usualmente maior que os pesos das outras amostras, ocasionando que o algoritmo foque mais nas amostras com ruído, podendo resultar em overfitting.

Boosting foi discutido primeiramente no artigo de 1990 por Schapire (SCHAPIRE, 1990a), em resposta a pergunta levantada por Kearns e Valiant em (LI; YU; ZHOU, 2012a), se um conjunto de aprendizes fracos poderia produzir um aprendiz forte. Posteriormente, a discussão levou ao desenvolvimento de diversos algoritmos de boosting, incluindo AdaBoost (FREUND; SCHAPIRE, 1999) e XGBoost (CHEN; GUESTRIN, 2016), que são os algoritmos clássicos de boosting que também foram testados neste projeto. Para melhor detalhe confira a figura 6 com os resultados. Na categoria de boosting, que foi testado neste projeto, os dois métodos que merecem destaque neste trabalho são o CatBoost (PROKHORENKOVA et al., 2018) e Light Gradient Boosting Machine (LightGBM) (KE et al., 2017). Interessante notar que como o artigo originalo CatBoost (PROKHORENKOVA et al., 2018) constatou, neste trabalho também verificamos que o CatBoost obteve melhores resultados relativo aos baseados em gradiente (lightgbm, xgboost, gbr em 6).

Como os artigos (PAUL; LUCA, ; HOCHACHKA et al., 2007) discutiram no passado, e como a análise que será desenvolvida na metodologia e as próximas seções demonstram, ainda nos dias de hoje boosting produz bons resultados, especialmente em situações onde os recursos computacionais são limitantes e, contudo, ainda se deseja uma boa performance.

2.2.4 Ensacamento (Bagging)

Ao contrário de boosting que agrega os modelos de forma sequencial, Ensacamento (bagging) trabalha com os modelos paralelamente. Em boosting os modelos base se diferenciam um do outro por utilizarem pesos diferentes nas amostras, bagging, por sua vez, não utiliza pesos mas também possui estratégias próprias para que os modelos base se diferenciem uns dos outros. Para diferenciar os modelos, bagging pode seguir uma estratégia heterogênea, mas é mais comum utilizar modelos homogêneos e buscar a variância entre os modelos de outras maneiras que começaremos a explicar nesta sub-seção. Abordagem homogênea é quando é repetido o mesmo algoritmo, em contra ponto, heterogênea é a abordagem que usa modelos diferentes. Daremos aprofundamento também na sub-seção 2.2.7, pois o método Extra-Trees, que foi utilizado como solução do problema tratado neste trabalho, pertence a esta classe de método de ensemble.

Para entender esta abordagem, é importante notar que alterando o conjunto de aprendizado pode levar a modificações significativas no modelo, o que por sua vez pode melhorar a acurácia (BREIMAN, 1996a). A técnica mais comum para buscar a variância do modelo é o bootstrap. O bootstrap é uma técnica vinda da estatística, consiste em replicar os dados de treinamento formando sub-conjuntos com reposição. Descrevendo em detalhes, é escolhido aleatoriamente n amostras em um conjunto, o elemento que foi escolhido não é retirado do conjunto (por isso o termo com reposição), tendo chances de ser escolhido aleatoriamente novamente. O principal ponto em bagging é a construção do conjunto de dados apresentado ao modelo. Ainda sobre os dados de treinamento, os sub-sets formados podem fazer uma seleção de características aleatória para aumentar a variabilidade mesmo sendo o mesmo modelo.

Bagging tem maiores chances de aumentar a performance de um aprendiz base, se o algoritmo utilizado para treino for instável. Um algoritmo instável, significativamente altera sua habilidade de generalização quando modificações são feitas nos dados de entrada. Bagging foca em reduzir a variância mais do que o bias, ao contrário do que foi falado anteriormente para boosting. Então, bagging performa otimamente quando os membros do ensemble possuem alta variância e um baixo bias. Um exemplo de algoritmo instável é decision tree, e de algoritmos estáveis temos o K-nearest neighbor (KNN) e o naïve Bayes. Então, como o que foi KNN e naive Bayes não vão ser boas opções para bagging enquanto decision tree será uma boa escolha (OZA; TUMER, 2008), justificando então o uso majoritário de árvores nesse tipo de algoritmos.

Como exemplo principal da classe temos random forest, introduzido em (BREIMAN, 2001), que em 2014 foi indicado no artigo (FERNÁNDEZ-DELGADO et al., 2014) como um dos principais algoritmos da época, e até hoje random forest ainda obtém uma performance competitiva com um custo de implementação baixo, o que o torna uma boa opção em problemas práticos. No problema tratado neste artigo também alcançamos uma boa performance com o algoritmo em questão, o que motiva seu destaque.

A agregação bagging foi desenvolvida em 1996 por Breiman (BREIMAN, 1996b) para aperfeiçoar a performance de modelo ML em problemas de classificação, combinando as predições de conjuntos de treinos separados de maneira aleatória da base de treino total(bootstrap). A maior vantagem de Bagging é reduzir a variância sem incrementar o bias. Bagging é uma boa opção para conjunto de dados grandes pois sub-particiona os dados, reduzindo assim o tempo de treinamento (ALELYANI, 2021). A desvantagem deste tipo de abordagem é que o uso de vários modelos limita a habilidade de interpretar a predição. Por exemplo, se utilizássemos uma decision tree, conseguiríamos interpretar seus resultados, mas como temos um conjunto de varias árvore a interpretabilidade do modelo vai se tornando difícil.

2.2.5 Empilhamento (Stacking)

Na abordagem empilhamento (Stacking), os aprendizes independentes são combinados por outro aprendiz. Os aprendizes independentes podem ser chamados de aprendizes cardinal, ou nível-0, e o aprendiz que combina de meta aprendiz,ou nível-1 (POLIKAR, 2012). O conceito de stacking é treinar os aprendizes cardinais com o data-set inicial, então é gerado um novo data-set denominado contemporâneo para ser aplicado no meta aprendiz. A saída gerada pelo aprendiz cardinal será o data-set que alimenta o meta aprendiz . Os aprendizes são obtidos aplicando algoritmos de aprendizados diversificados, porém, pode-se aplicar o mesmo algoritmo formando um empilhamento (stack) uniforme.

Como stacking utiliza diversos algoritmos base,ele não precisa de técnicas como bootstrap para produzir modelos diversos. E como a técnica utiliza um modelo, chamado de level-1, para combinar diversos modelo do level-0, a técnica consegue aprender quais predições são melhores. Diferentemente de boosting que precisa de etapas sequenciais para corrigir o bias, a técnica consegue aprender quando algum aprendiz cardinal está errando, e corrigi-lo isso pegando o resultado de outro aprendiz cardinal.

Stacking foi introduzido em 1992 por Wolpert (WOLPERT, 1992) com a finalidade de reduzir o erro na generalização em problemas de machine learning. Stacking é útil em situações onde modelos ML são únicos na captura de algum aspecto em particular de um tarefa ; então, a abordagem de stacking utiliza um modelo ML separado para aprender quando utilizar a predição dos vários modelos (WITTEN; FRANK; HALL, 2011). A desvantagem desta técnica é que é fácil ocorrer vazamento de informações no treinamento, o que torna a técnica de difícil implementação e por este motivo não é uma técnica muito explorada em machine learning (WITTEN; FRANK; HALL, 2011). Também uma limitação importante é o alto custo computacional da técnica.

O sucesso da técnica de ensemble tem como seus pontos de sustentação a acurácia e a diversidade dos aprendizes base (LI; YU; ZHOU, 2012b). Um modelo de aprendizado de máquina é considerado acurado se ele possui uma boa habilidade de generalização em instâncias ainda não vistas. Os modelos são considerados diversos se os erros cometidos nas instâncias

não vistas não são os mesmos erros (BIAN et al., 2020). Por exemplo, em um processo de ML comum temos dois conjuntos, treinamento e validação, o modelo é treinado no conjunto de treinamento, então as instâncias não vistas são as do conjunto validação, que são utilizadas para verificar a habilidade de generalização do modelo (dada por métricas como a acurácia ou MSE). Por outro lado, se quando é passada as amostras de validação os modelos erram nas mesmas amostras, eles não são diversos. Diversidade é dita como sendo a diferença entre os aprendizes bases (ALSHDAIFAT; AL-HASSAN; AL-OQAILY, 2020). Não existe uma regra para atingir uma boa diversidade no modelo. Os modelos de ensemble geralmente tentam alcançar diversidade heuristicamente ou implicitamente. Como exemplo, bagging utiliza sub-conjunto dos dados de treino, por sua vez, boosting faz uso de pesos para tentar alcançar uma boa diversidade (MIENYE; SUN, 2022).

Sumarizando os três tipos de aprendizado por conjunto, impulsionar, ensacamento e empilhamento, temos a figura 2. Impulsionar tem como característica aprendizes um em sequencia do outro. Geralmente trabalhando com o peso das amostras para melhorar a performance do modelo. O passo 1 simboliza o processo de alimentar o aprendiz, e o 2 e 3 significa corrigir o peso das amostras e alimentar o próximo modelo pelo quanto de interações que se deseja. Ensacamento por sua vez trabalha com bootstrap para aumentar a variância dos aprendizes. Como pode ser observado no passo 1 desse algoritmo, o data-set é o mesmo para todos aprendizes, o próprio algoritmo do aprendiz se encarrega de particionar os dados aleatoriamente. No passo 2 é utilizada alguma regra para agrupar os resultados, como média aritmética para problemas de regressão, ou voto majoritário para problemas de classificação. Empilhamento, por sua vez, é a técnica mais complexa das 3. Ela possui dois níveis, 0 e 1. No nível 0 aprendizes distintos constroem uma nova base de dados, exposta no passo 2, essa nova base alimenta um novo aprendiz no nível 1 no passo 3. Este aprendiz fica encarregado de aprender qual aprendiz é melhor em qual momento, então combinando as respostas e nos levando a um resultado final no passo 4.

Agora que sabemos onde o método Extra-Trees se encaixa no meio de tantas opções de algoritmos de aprendizado de máquina, vamos começar a aprofundar o funcionamento sobre o método escolhido. Para isso vamos começar introduzindo o algoritmo para árvores de classificação e regressão (CART).

2.2.6 Árvores de Classificação ou Regressão (CART)

Como o nome sugere CART, sabendo que o acrônimo se refere a Classification and Regression Trees, pode ser usado para problemas de classificação e regressão. Como já foi explorado a distinção se dá na variável alvo. Em problemas de classificação se tenta descobrir uma classe (ex. vai chover amanhã? sim ou não), enquanto problemas de regressão se procura um valor numérico que pode ser um conjunto infinito de valores (ex. preço de casa). Como exposto anteriormente ambos os problemas fazem parte do aprendizado supervisionado.

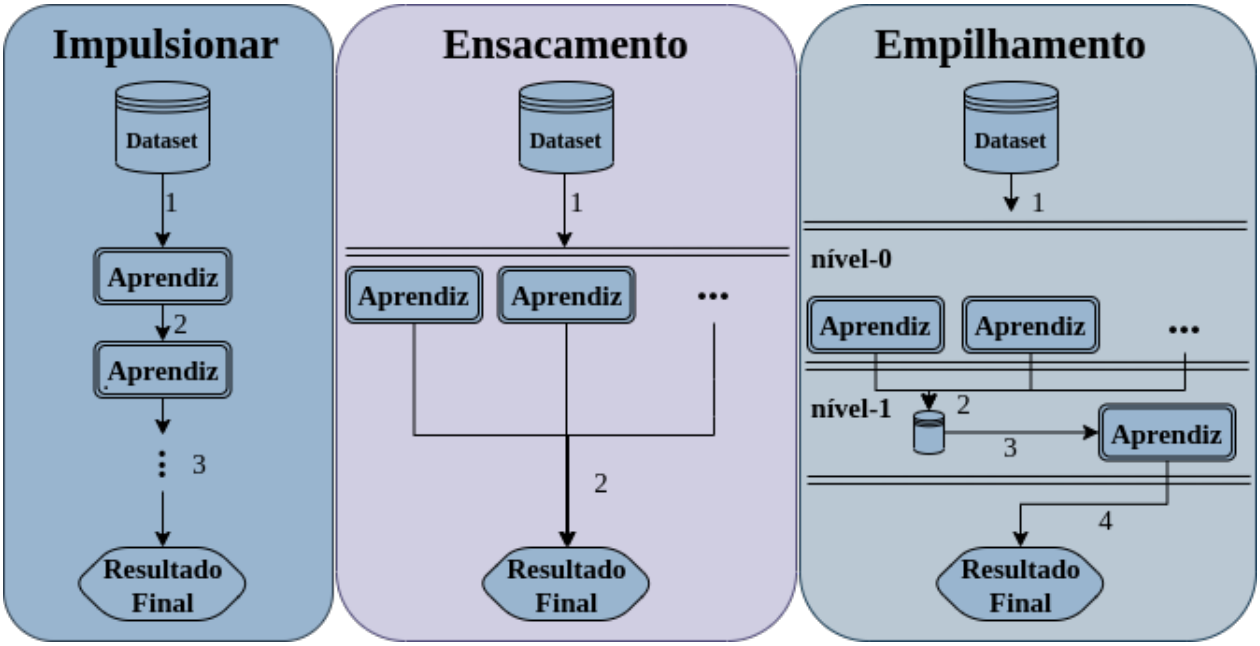

Figura 2 – Sumário aprendizado por conjunto.

O algoritmo CART pode ser melhor explicado com um exemplo prático, e para fins didáticos o problema será de classificação, porém facilmente pode-se transpor o conceito para a regressão. Supondo o objetivo é distinguir entre dois tipos de fruta, laranjas e tangerina. Sabe-se que laranjas tipicamente possuem um diâmetro entre 6-10 cm e, são maiores que as tangerinas com um diâmetro entre 4-8 cm. Sabe-se também que as tangerinas tendem a ser ligeiramente mais escuras que as laranjas, então, usando uma escala de cor (1 = escuro até 10 = claro). Com essas informações, consultando os dados que teríamos a disposição, conseguimos estimar algumas possíveis regras para a árvore:

1) Diâmetro $\leq$ 7cm

2) Cor $\leq$ 5

3) Cor $\leq$ 6

Aplicando as possíveis regras estimadas pode se obter uma árvore que ajudará a fazer a distinção em questão. A árvore pode ser vista na figura 3. Note que com as regras 2 e 3 regem sub-conjuntos distinto dos dados.

Na figura 3 ainda conseguimos ter um bom exemplo da estrutura de uma árvore. Nota-se que a árvore possui 3 estruturas padrões: Nó raiz, ramos e folhas. Fazendo a transcrição da analogia, enfatizamos então que no caso os ramos seriam as decisões tomadas a partir da regra, que por sua vez são referenciados pelos Nós (raiz e subsequentes), as folhas seriam as decisões finais. A ideia então é que apresentado um dado novo - e extraindo as mesmas características que foram usados para o treinamento da árvore -, a árvore criada no treinamento

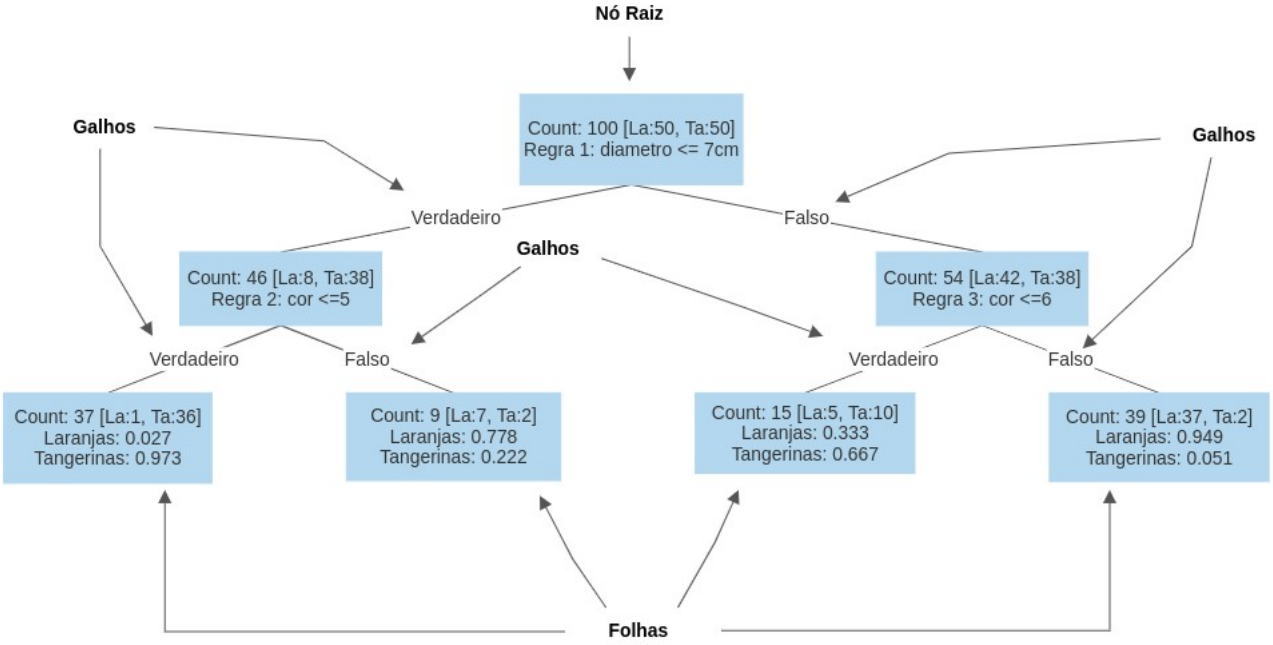

Figura 3 – Exemplo de implementação das regras.

do algoritmo seria capaz de aplicar as regras até que o dado percorra toda a árvore. No final chegaríamos ao nó folha nos levando a resposta (classe ou variável). Ainda sobre o exemplo, para fins didáticos, estipulamos que o nosso problema é composto por 100 exemplos, que pode ser localizado na figura por count (contador). Entre colchetes estão a quantidade respectiva a cada classe (La:Laranja, Ta: Tangerina). Os 100 exemplos iniciais vão percorrendo cada nó da árvore de acordo com as respostas dadas pelas suas características previamente extraídas. No quadrado finaltambém é exposto a porcentagem das classe que foram "filtradas" por cada folha, com a classe correta destacada pela cor. Observamos que com apenas estas 3 regras, em nosso problema didático, a árvore ainda se confunde em diversos casos, mas atinge uma acurácia de90%. Ou seja dos 100 exemplos de entrada, 90 amostras teriam sidos classificados corretamente com as 3 regras colocadas nesta ordem e, 10 amostras teriam sido classificadas incorretamente.

Algumas questões o exemplo deixa de cobrir, como por exemplo: Como o algoritmo CART sabe a melhor divisão? De fato existem diversas respostas corretas para essa questão, podemos citar aquipara problemas de classificação o grau de impureza: Gini, e a entropia. Gini sendo um coeficiente variando de 0 a 1, onde 0 representa a igualdade de todas as amostras, ou seja é observado no nó em questão que todas as amostras pertencem a apenas uma classe. Entropia mede a desordem do sistema, ou impureza, o ganho na informação da divisão dado pela regra pode ser medido pela redução que se dá na entropia.

Na imagem 2.2.6 é demonstrado o funcionamento da divisão utilizando Gini, com apenas uma parte do mesmo exemplo explorado anteriormente. Para problemas de regressão

podemos utilizar o mínimo erro quadrado, ou qualquer outra que se adapte melhor aos dados em questão. Neste artigo o Extra-Trees utiliza MSE como métrica padrão da divisão, importante não confundir com as métricas que são utilizadas para avaliar o modelo montado, a confusão pode acontecer devido as métricas poderem ser usadas em ambos os casos. O conceito por trás das métricas é apenas uma forma de se avaliar o quão boa vai ser sua divisão. Para escolher os pontos das features onde será realizada a divisão, cada método tem sua própria abordagem, mas como um exemplo para desenvolver o conceito, pode se fazer o conjunto de todos os pontos possíveis e testá-los utilizando a métrica escolhida. Em termos práticos essa abordagem não é viável, pois essa abordagem pode facilmente gerar um número infinito de pontos, por isso cada método irá desenvolver uma abordagem específica para resolver este problema. A re-amostragem em sub-conjuntos, ou bootstrap apresentada na sub-secção 2.2.2, ajuda a reduzir o custo computacional nesse ponto, e é um dos motivos por sua intensa utilização. Importante notar que a divisão e seleção dos data sets pode acontecer a nível colunas e também de linhas. Vale enfatizar que a perda de informação gerada pelos sub-sets é compensada quando se analisa o contexto de múltiplas árvores implementando este modelo.

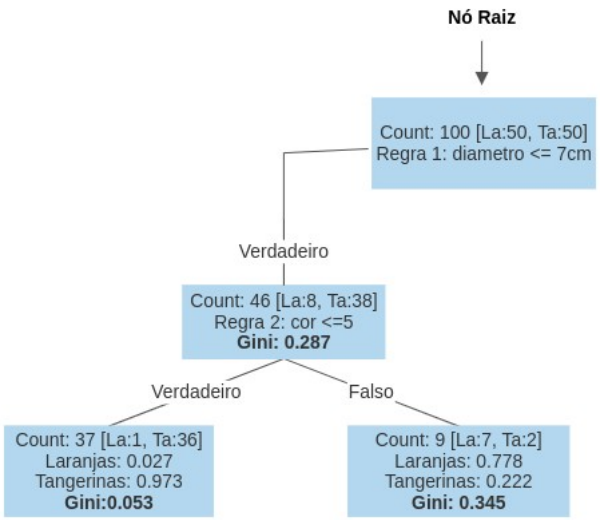

Figura 4 – Gini.

2.2.7 Algoritmo Extra-Trees

Extra trees é um método de ensemble que foi validado em diversos contextos, como será mostrado nesta seção. Atualmente é tido como um método que possui performance comparável a de outros modelos do estado da arte em aprendizado de máquina. Essencialmente o algoritmo consiste em randomizar fortemente os atributos e a escolha do ponto de corte quando se está dividindo um nó da árvore (GEURTS; ERNST; WEHENKEL, 2006). Por exemplo, quando se está dividindo um nó, existe um conjunto de atributos (também chamado de características ou

features), esse conjunto é escolhido aleatoriamente, e o ponto de corte de cada atributo é feito de maneira aleatória. Na seção anterior falamos que podia fazer uso de Gini juntamente com bootstrap para estabelecer esse ponto de corte, Extra-Trees pode não fazer uso de bootstrap, que é um diferencial para esse método. Se trata de uma nova solução que trabalha com pontos de corte totalmente aleatórios, por isso o nome árvores extremamente randomizadas (extremely ramdomized trees), ou apenas Extra-Trees. Após realizar o ponto de corte de cada atributo utilizando um sub-conjunto (nota-se que o algoritmo também pode trabalhar com o conjunto completo), se faz uso de uma métrica adequada (exemplo: Gini, MSE) para verificar qual ponto de corte é o melhor deste grupo, o melhor então é escolhido para a divisão do nó. Os pontos de corte são realizados de maneira completamente aleatória mas a escolha é feita pelo melhor na métrica em questão.

Vamos dar sequência com o aprofundamento do algoritmo extra-tree primeiramente com uma revisão bibliográfica, após vamos retomar com a descrição do algoritmo para melhor aprofundamento.

2.2.7.1 Revisão Extra-Trees

Em (AQEEL; MAJID, 2022), o método foi testado juntamente com gradient Boosting, XGBoost, Support Vector Machine, Decision Tree e Random Forest, e foi escolhido dentre estes como modelo para solucionar o problema de classificar possíveis drogas no combate de COVID-19. A proposta era identificar drogas possivelmente eficazes no combate a COVID-19. Aproveitando o contexto atual, em (PATIL, 2022), o método Extra-Trees foi utilizado em um esquema multi-modelo, de voto majoritário, atingindo uma performance de 96% de acurácia, precisão e F1-measure (acurácia, precisão e F1-measure são métricas de sucesso do algoritmo, quanto maior melhor), no problema de classificação de Fake news. Em (MUNIR; PARK; HONG, 2022) foi utilizado por fazer parte dos modelos de ML explicáveis - que suas predições podem ser compreendidas - e mostrou potencial para auxílio no controle autônomo de internet de tudo (IoE) na sexta geração de redes wireless (6G). Nota-se que apesar do modelo ter sido selecionado por sua capacidade de ser explicado - inerente de modelos baseados em árvores - o modelo perde está capacidade conforme o aumento no numero dos estimadores (aprendizes, preditores, classificadores, regressores). De fato, quanto mais estimadores se tem, mais difícil explicar o modelo. O modelo Extra-Trees também foi utilizado como estratégia de imputação de datasets em (LLERA et al., 2022), sendo imputação de data-set a tarefa de atribuir valores a colunas que possuem uma parte faltante, comum em problema práticos de big data. No caso, foi demonstrado que estratégias de imputação univariantes, como substituir pela média ou mediana, é pior que utilizar estratégias multivariadas com modelos de regressão. Ainda, sobre o artigo (LLERA et al., 2022), o data-set em questão era utilizado na busca de correlações com o autismo, onde tinha-se a necessidade de utilizar uma estratégia efetiva de imputação para obter as correlações com mais eficiência. O modelo Extra-Trees também mostrou sua eficiência em diversos outros problemas (XIE et al., 2020; PU et al., 2021; ISHAQ et al., 2021;

SAEED et al., 2021; ALKAWAZ et al., 2022). Neste trabalho, ficou constatado que dentre os modelos testados inicialmente, Extra-Trees teve um papel de destaque na solução do problema de tomografia de hamiltonianos quânticos que pretendemos trabalhar neste projeto. Como fizemos uso do algoritmo de Extra-Trees, vamos agora introduzir o seu funcionamento de forma detalhada.

2.2.7.2 Descrição do Algoritmo

Extra-Trees constrói um conjunto de árvores de decisão ou regressão sem podas de acordo com o procedimento clássico, top-down (de cima para baixo). Existem duas principais diferenças com relação ao Extra-Trees e outros algoritmos baseados em conjuntos de árvore segundo o artigo original (GEURTS; ERNST; WEHENKEL, 2006). As principais diferenças são: ele divide os nós da árvore escolhendo pontos de corte completamente aleatórios, e o algoritmo em questão utiliza o conjunto de amostras de treino por completo. Foi falado na seção 2.2.4 que as técnicas recorriam a técnicas de re-amostragem (ex. bootstrapping), para crescer suas árvores para solucionar dois problemas: alto custo computacional na busca por pontos de corte no espaço de característica, aumentar o grau de variabilidade entre os modelos paralelos. Note, então, que com os pontos de corte na característica sendo utilizados de maneira aleatório, se consegue solucionar estes dois problemas.

Algoritmo 1: Divisão da árvore - Pseudocódigo (Para atributos numéricos)

1 **Função** *Divide_Nó*(S):
  entrada: Subset de aprendizado S, correspondente ao nó que deseja realizar divisão
  saída: Uma divisão $[a < a_c]$ ou vazio
2 **se** `Parar_divisão`(S) **então retorna** *vazio*;
3 **senão**
4 - Selecione K atributos, a_1, ..., a_k dentre todos atributos candidatos em S;
5 - Sorteie K divisões s_1, ..., s_k, onde s_i = `Escolhe_Divisão_Aleatória`(S, a_i), para todo $i = 1$, ..., K;
6 **retorna** *divisão* s_* *sendo que* $Score(s_*, S) = max_{\{1,...,k\}} Score(s_i, S)$
7 **Fim**
8 **Função** *Escolhe_Divisão_Aleatória*(S, a)
  entrada: Um subconjunto S e um atributo a
  saída: Uma divisão
9 - a^S_{max} e a^S_{min} denotam o valor máximo e mínimo de a em S;
10 - Sorteie um ponto de corte a_c de maneira uniforme em $[a^S_{min}, a^S_{max}]$;
11 **retorna** *divisão* $[a < a_c]$
12 **Fim**
13 **Função** *Parar_Divisão*(S)
  entrada: Um subconjunto S
  saída: Uma divisão
14 **se** $|S| < N_{min}$ **então retorna** *Verdadeiro*;
15 **se** *Todos atributos constantes em* S **então retorna** *Verdadeiro*;
16 **se** *Todas as saídas constantes em* S **então retorna** *Verdadeiro*;
17 **senão retorna** *Falso*;
18 **Fim**

O algoritmo 1 possuitrês funções distintas, a principaDivide_Nó e duas auxiliares Escolhe_Divisão_Aleatória e Parar_Divisão. Vamos começar explicando as funções auxiliares de trás para frente, por questões de simplicidade. A função Parar_Divisão, tem como entrada um subconjunto$S$, o objetivo dela é dizer se as divisões chegaram ao final,ou seja, se já dividimos os nós até as folhas.Para distinguir um nó de uma folha temos que verificar as condições presentes nas linhas 14,15,16, todas elas retornam um valor verdadeiro que nos diz que devemos parar de dividir o nó. As condições são:

- $|S| < N_{min}$, a qual verifica se o subconjunto S possui a quantidade mínima de samples para divisão do nó.
- Todos atributos constantes em S, se os atributos não possuem mais um ponto de mínimo e máximo, sendo uma constante, então não temos condições de dividí-los pois não

conseguiríamos dividir as amostras fazendo esse processo.

- Todas as saídas constantes, analogamente ao problema dos atributos, se alcançado um nó puro, ou seja, onde todas as samples levam ao mesmo resultado, então podemos parar a divisão retornando verdadeiro.

Notadamente apenas o primeiro item é um critério que podemos calibrar em aplicações práticas.

A função Escolhe_divisão_Aleatória, recebe como entrada um sub-conjunto S referente ao nó local que se está realizando a divisão no momento, e um atributo a. O objetivo da função é escolher uma divisão aleatória para o atributo que foi passado para ela. Para realizar essa tarefa, o primeiro passo na linha 9, verifica o valor máximo e o valor mínimo do atributo a dentro de S. Com o valor máximo e mínimo calculado, o algoritmo sorteia um ponto de corte de modo uniforme (linha 10), ou seja sem que tenha algum peso atuando no sorteio. Após o sorteio o ponto de corte é então retornado, simbolizado por $[a < a_c]$. A função principal, Divide_Nó(S), por sua vez, tem como entrada o sub-set S. O objetivo da função é dividir o nó. Então, primeiramente é testado se o sub-set S ainda possui todos os critérios para realizar a divisão. Para realizar essa tarefa, exposta na linha 2, é utilizada a função auxiliar Parar_Divisão com os seus critérios. Se Parar_Divisão retornar verdadeiro, a divisão é interrompida e é retornado vazio, caso contrário a divisão dá sequência com a seleção de K atributos dentre todos os atributos candidatos em S. Para cada atributo é então utilizada a segunda função auxiliar, Escolhe_Divisão_Aleatória para extrair K pontos de corte dentre os K atributos selecionados. Com os K ponto de corte calculados, se obtém um score para cada um deles, o score pode ser calculado com o grau de impureza gini ou média quadrada do erro, como foi exposto na seção 2.2.6.

O procedimento descrito em 1 possui dois parâmetros: K, o número de atributos selecionados aleatoriamente em cada nó e N_{min}, a quantidade mínima de samples para se dividir o nó. O procedimento então forma apenas uma árvore, para completar o número de M árvores desejado, o procedimento é repetido diversas vezes com as samples originais (completas) para gerar um modelo de conjunto (ensemble model). As predições das árvores são então agregadas para gerar a predição final, por voto majoritário para problemas de classificação e média aritmética em problemas de regresão.

Os parâmetros K, N_{min} e M possuem diferentes efeitos: K determina a força no processo de seleção de atributos, N_{min} a força da média do ruído de saída, e M a força da redução da variância na agregação do modelo de conjunto. Esses parâmetros podem ser adaptados especificamente para o problema de maneira manual ou automática e com técnicas de validação cruzada (cross-validation). No entando, é preferido utilizar as configurações padrões para maximizar o uso computacional (GEURTS; ERNST; WEHENKEL, 2006).

Em uma perspectiva da variância referente ao bias, a racionalização por trás do método Extra-Trees é que uma randomização explicita do ponto de corte e atributo, combinado com

a média do conjunto, deve ser capaz de reduzir a variância mais intensamente do que uma randomização fraca proposta por outros métodos. Random Forest, algoritmo de bagging, não faz uso de pontos de corte aleatório, por exemplo. O uso da base original de aprendizado completa ao invés de réplicas bootstrap é motivado neste algoritmo para minimizar o bias. Do ponto de vista computacional, a complexidade do procedimento de crescimento das árvores é, assumindo que a árvore é balanceada, da ordem de $N \log N$ com respeito ao tamanho das amostras de aprendizado, assim como outros métodos de crescimento de árvore. No entanto, dado a simplicidade da divisão do nó, é esperado uma maior eficiência computacional do que em outros métodos de conjunto que localmente otimiza o ponto de corte.

3 Metodologia

Para explicar o desenvolvimento da metodologia, vamos primeiro introduzir as base de dados utilizadas em cada etapa do projeto, dando detalhes de sua construção. Para atingir o objetivo finalde realizar a tomografia de um hamiltoniano quântico de dois qubits, começamos simulando a dinâmica imposta pela Eq(5). Inicialmente iremos focar na determinação do acoplamentos mas, como será demonstrado nessa seção, com o sucesso de nossa abordagem, seguimos ampliando o escopo do modelo a fim de realizarmos a tomografia completa do hamiltoniano de dois qubits, identificando o grau de acoplamento juntamente com os campos externos.

A modelagem inicialfoi realizada em duas fases,a fase de seleção do modelo,e a fase de análise do modelo escolhido. Na fase de seleção do modelo, foi feita a utilização da ferramenta PyCaret (ALI, 2020) para implementação dos modelos de aprendizado de máquina e análise dos resultados. A decisão finalsobre a escolha do modelo levou em conta as métricas expostas pela tabela de resultados 6 e os motivos discutidos na fundamentação presente neste trabalho.

A partir da escolha do modelo e seu potencialtendo sido validado pela fase de seleção, começamos a fase de análise. Nesta etapa fazemos uso de um Hamiltoniano mais rigoroso, que descreve de forma apropriada a dinâmica do duplo ponto quântico. O Hamiltoniano efetivo é dado por:

$$\hat{H}_{2-qubit} = \frac{1}{2}\Big[J_1\sigma_z^{(1)}\otimes\mathbf{I} + J_2\mathbf{I}\otimes\sigma_z^{(2)} + \frac{J_{12}}{2}\left(\sigma_z^{(1)}\otimes\sigma_z^{(2)} - \sigma_z^{(1)}\otimes\mathbf{I} - \mathbf{I}\otimes\sigma_z^{(2)}\right) + \Delta B_{z,1}\sigma_x^{(1)}\otimes\mathbf{I} + \Delta B_{z,2}\mathbf{I}\otimes\sigma_x^{(2)}\Big]. \tag{9}$$

Como podemos observar, a estrutura do Hamiltoniano é similar à aquela dada pela Eq. (5). Apesar da mudança de nome das variáveis das duas equações, os hamiltonianos são equivalentes, apesar da inclusão das razões e da adição de duas operações locais$\sigma_z$em cada qubit com a atuação da variávelJ_{12}. Importante notar que mudando o hamiltoniano, as outras partes que compõem o cálculo da base ainda permaneceram inalteradas, como o cálculo das observáveis, e as propriedades referentes às características utilizadas para treino do modelo. Além disso, para esta fase foirealizado um aumento na base de dados além de um novo intervalo de valor dos parâmetros do hamiltoniano. Nesta fase também foi feito diversos experimentos para melhor estudo do problema.

Os experimentos mais relevantes para a solução do problema estão presentes no capítulo 4, dentre os experimentos temos: visualização do comportamento das observáveis, visualização do erro conforme os tempos em que a dinâmica foi observada, visualização dos erros em relação ao alvo, visualização do erro conforme quantidade de exemplos utilizados. Os experimentos

estão descritos em detalhe no capítulo 4. Importante notar que com a fase de seleção do modelo concluída, o projeto entra em um ciclo de refinamento onde se realiza testes, e com os resultados dos testes se inicia uma nova etapa de análise com modificações, um processo natural em problemas de ciência de dados.

Com o sucesso na predição do valor J_{12} presente no hamiltoniano eq. 9, prosseguimos ampliando o escopo do projeto para atingir a tomografia completa do hamiltoniano. Nesse passo buscamos determinar além do termo de acoplamento as variáveis referentes aos campos J_1, J_2, $\Delta B_{z,1}$ e $\Delta B_{z,2}$, totalizando as 5 variáveis que compõem o hamiltoniano e atingindo o objetivo final de realizar a tomografia completa.

Para tratar como esse projeto tentou solucionar o problema de regressão, em busca de encontrar uma ferramenta prática para físicos experimentais utilizando métodos de aprendizagem de máquina, nas próximas seções será dado sequência com uma introdução sobre as ferramentas utilizadas, formação da base utilizada e configurações.

3.1 Ferramentas

Neste projeto foi necessário o uso de diversas ferramentas para realizar as tarefas de criar o data-set, manipular o data-set, implementar os modelos e visualizar os resultados. Nessa seção será explorada as ferramentas principais e como elas ajudaram a solucionar o problema de tomografia do hamiltoniano quântico. Vale ressaltar que as bibliotecas citadas aqui são do ambiente de desenvolvimento Python. A motivação para o uso do ambiente Python se deve ao fato de se tratar do principal ambiente para desenvolvimento de projetos envolvendo aprendizado de máquina, possuindo inúmeros recursos implementados e, amplamente testados, possibilitando realizar o controle dos dados e desenvolvimento de modelos de aprendizado de máquina de forma rápida, segura e consistente.

A principio, o projeto utilizou a biblioteca PyCaret com a função de preparar os dados e implementar os modelos. Na fase de análise do modelo escolhido, foi implementado o modelo escolhido com a biblioteca scikit-learn. Em ambas as fases também foi feito o uso da biblioteca Pandas juntamente com Numpy para manipular os dados, também foi feito o uso da matplotlib para visualizar os resultados. Em ambas as fases foi utilizada a linguagem python. Dentre as bibliotecas citadas, em especial, vamos introduzir brevemente as bibliotecas referentes a implementação dos modelos e manipulação dos dados.

PyCaret (ALI, 2020) é uma biblioteca de aprendizado de máquina de código aberto em Python que visa reduzir o tempo do processo de ML. Ele permite que se realizem experimentos completos com rapidez e eficiência. Em comparação com outras bibliotecas de aprendizado de máquina de código aberto, PyCaret é uma biblioteca alternativa que utiliza um padrão de baixa quantidade de linhas de código que pode ser usada para realizar tarefas complexas de aprendizado de máquina. Todas as operações realizadas no PyCaret são armazenadas

automaticamente em um pipeline personalizado que é totalmente orquestrado para implantação. A biblioteca foi selecionada por facilitar a implementação e adequação da base e parâmetros de diferentes modelos. Apesar de ser uma biblioteca com pouco tempo de desenvolvimento, seu lançamento é do ano de 2020, a biblioteca atua apenas como um agregador (wrapper), uma interface que unifica diversas bibliotecas que são amplamente utilizadas e foram amplamente testadas para resolver problemas práticos de aprendizado de máquina. Como exemplo de tais bibliotecas utilizadas internamente temos scikit-learn, XGBoost, Microsoft LightGBM, spaCy entre outras.

Scikit-learn (PEDREGOSA et al., 2011) foi iniciado em 2007 como um projeto Google Summer of Code de David Cournapeau. Mais tarde naquele ano, Matthieu Brucher começou a trabalhar neste projeto como parte de sua tese. Em 2010, Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort e Vincent Michel assumiram a liderança do projeto e fizeram o primeiro lançamento público. Desde então, vários lançamentos surgiram e uma comunidade internacional tem liderado o desenvolvimento. Scikit-learn é uma das bibliotecas mais utilizadas para desenvolvimento de modelos de ML por fornecer implementações de diversos modelos ML, e por ser uma biblioteca de fácil uso. Neste projeto, o modelo final proposto foi desenvolvido utilizando a implementação do algoritmo extra-trees presente nesta biblioteca.

Como foi citado, utilizamos a biblioteca Pandas, que é uma biblioteca que permite que você trabalhe com seus dados em forma de data frame. Data frame é uma forma de estruturar os dados, onde se transforma os dados em formato tabular, com linhas representando cada entrada, e colunas representando as características e alvo de cada entrada. Em um projeto de dados, é comum possuir situações onde a necessidade de separar os dados de várias formas diferente. Como exemplo, tivemos situações onde foi necessário separar os dados em conjuntos distintos de treinamento e testes, ou então, separar os dados apenas com algumas colunas. Ainda sobre manipulação dos dados para criar os dados que alimentam o Data frame, foi utilizada a biblioteca numpy. Numpy é uma biblioteca que fornece rotinas ao Python que permitem processar dados de maneira eficiente, em outras palavras o numpy é utilizado para os cálculos. Aproveitando, vamos agora descrever os passos para a criação da base de dados.

3.2 Bases de Dados

A base de dados para aplicação dos métodos de regressão foi construída seguindo os seguintes passos:

1) Escolhemos um estado $|\Psi(0)\rangle$ como estado inicial;

2) Tomamos o hamiltoniano H que descreve o sistema quântico para dois qubits;

3) Calculamos a dinâmica e consequentemente $|\Psi(t)\rangle$;

4) Calculamos os valores esperados $O_k(t)$ descritos pela Eq. (7);

Com relação ao primeiro passo,o estado inicial considerado na fase de seleção de modelos foi:

$$|\Psi(0)\rangle = |\uparrow\uparrow\rangle \tag{10}$$

Na fase de análise do modelo escolhido, o estado inicial foi trocado para o estado:

$$|\Psi(0)\rangle = \frac{1}{2}(|\uparrow\uparrow\rangle + |\uparrow\downarrow\rangle + |\downarrow\uparrow\rangle + |\downarrow\downarrow\rangle) \tag{11}$$

No segundo passo, quando definimos o hamiltoniano H, para a fase de seleção do modelo o hamiltoniano definido pela equação 5 foi configurado inicialmente com c e d sendo 1, a e b, variando em intervalo de $[0, 1]$, e J variando entre $[0, 10]$ com passo de 0.1 para todos. Calculamos então a dinâmica com os valores esperados calculados, com o tempo t variando em um intervalo $[5, 25]$ com passos de 5. Na fase de análise do modelo escolhido, o hamiltoniano definido é passou por algumas alterações e o que anteriormente era definido pela eq. 5, nesta etapa foi dado pela equação 9. Vale enfatizar que as alterações de uma equação para outra são: Os parâmetros sofreram alteração no nome, o que era a, b, J , c e d, foram transformados em J_1, J_2, J_{12}, $\Delta B_{z,1}$ e $\Delta B_{z,2}$ respectivamente. Note que foi adicionado as devidas proporções em cada termo.

O fluxo completo para criação da base está presente na imagem 5. Como pode ser visualizado, primeiramente é realizado os cálculos descrito nos passos, conforme citado no começo da seção. No passo 2 temos o hamiltoniano 9 e no passo 3 a dinâmica do sistema é calculada. No quarto passo calculamos, finalmente, o que é chamado de observáveis do sistema. Como é um sistema composto e as observações são locais, podemos observar os dois qubits, portanto temos duas equações. Ainda no quarto passo, note que usamos três observáveis para cada qubit σ_x, σ_y e σ_z, ou seja, de acordo com o modo como se deseja observar usamos uma matriz de Pauli diferente. Observe que as matrizes estão definidas na primeira linha do passo 4 e na equação da observável ela esta denotada por σ_k com $k = \{x, y, z\}$

Após os passos, vamos obter um conjunto de dados, para organizar esses dados temos as tarefas de manipulação dos dados. As observáveis são funções que dependem do tempo, como está explícito no passo 4. Como foi dito, temos duas equações, uma para cada qubit. Cada qubit pode ser observado de três formas diferentes, possibilitando transformar as duas equações em seis, uma para cada observável. Agregando a informação que as observáveis trazem em função do tempo, e notando que observar os dois qubits de três formas diferentes pode nos dar seis equações, então temos a tabela de características demonstrada na figura. Cada observável forma uma coluna diferente na tabela de características, e para cada tempo que se deseja observar, podemos ter até seis colunas a mais por tempo. Ainda no agregamento do data-set, temos outra tabela denominada alvo. Teremos uma tabela alvo diferente para cada parâmetro indicado pelas setas vermelhas do passo 2 que desejamos obter do algoritmo de

aprendizado de máquina. Na figura foi dado como exemplo J_{12}, voltando no passo 2 dos cálculos, cada variávelselecionada com a seta vermelha se transforma em uma tabela alvo diferente. Determinar todas as variáveis que estão sublinhadas pela seta vermelha no passo 2 é o que chamamos de tomografia completa do hamiltoniano e é o objetivo ideal desse projeto. Após agregar as tabelas, as mesmas são divididas conforme a necessidade em data-set. Neste projeto a divisão comum foi treino e teste, mas também foi utilizado a técnica de cross-validation.

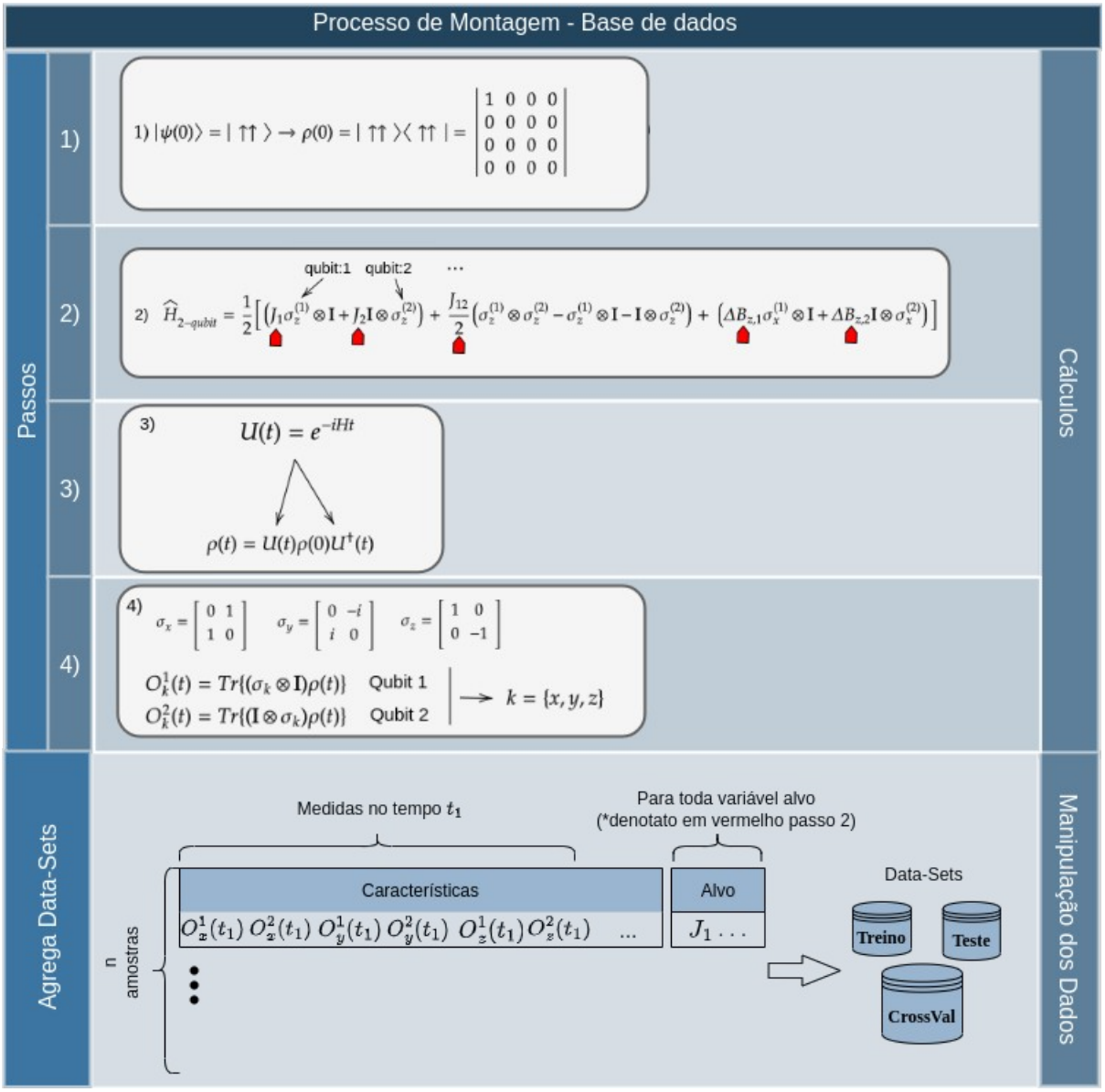

Figura 5 – Processo de criação da base de dados.

3.3 Configuração Experimental

3.3.1 Configuração para Seleção do Modelo

Com os passos discutidos na seção anterior, construímos a base de dados tanto para seleção do modelo quanto para sua análise, de forma que discutiremos brevemente sobre as configurações para treinamento do modelo.

Seguindo os passos, obtemos uma base de dados para aplicação dos métodos de ML onde se tem 6 valores esperados, calculados para cada um dos 5 tempos distintos, que formam por sua vez as características da base com J sendo o alvo que se deseja determinar.

É importante frisar que c, d foram fixados para testar a viabilidade do projeto e consideramos como alvo o parâmetro J, que tem a função de determina o grau de acoplamento entre os dois qubits. J é o alvo mais difícil de se determinar por conta de induzir uma interação entre dois qubits, conforme pode ser visualizado na equação 5. Como dito, o hamiltoniano foi configurado inicialmente com c e d igual a 1, a e b variando em intervalo de $[0, 1]$, e J variando entre $[0, 10]$ com passo de 0.1 para todos. As medida, foram feitas nos tempos 5, 10, 15, 20 e 25. Importante notar que a base de dados com os intervalos e passos descritos, possui próximo de 12.000 resultados e ela foi dividida com 70% dos dados para treino e 30% para testes. Também foi utilizado o método de validação cruzada K-fold com K setado em 10.

3.3.2 Configuração para Análise do Modelo

Sobre as configurações utilizadas na fase de análise do modelo escolhido, foi construída a base de dados da forma descrita na seção sobre base de dados. As características usadas para treinamento do modelo permaneceram inalteradas nesta fase. Mudanças na divisão da base foram incluídas para melhor confiabilidade dos resultados apresentados pelo modelo. A divisão usada continua 70% dos dados para treino e 30% para testes, porém foi incluída uma base de validação gerada aleatoriamente dentro das ranges estipuladas, e produzida sem qualquer alteração nos cálculos, no tamanho de 10.000 amostras. Sobre o tamanho total da tabela, foi pego um total de 1.000.000 de amostras com o intervalo proposto.

3.4 Métricas

As métricas servem para avaliar a performance de um algoritmo de aprendizado de máquina em um determinado conjunto de dados de teste. Elas podem ser utilizadas para comparar diferentes algoritmos, além de ajustar os seus hiperparâmetros. Neste trabalhos nós utilizamos a Média do Erro Absoluto e o R-Quadrado. Abaixo descrevemos brevemente cada uma delas.

3.4.1 Média do Erro Absoluto

Média do Erro Absoluto (MAE): Mede a magnitude do erro em um conjunto de predições, sem considerar a direção. É a média sobre a amostra de teste das diferenças absolutas entre a previsão e a observação real, onde todas as diferenças individuais têm peso igual.

$$\mathrm{MAE} = \frac{1}{n}\sum_{j=1}^{n} |y_j - \hat{y}_j| \,. \tag{12}$$

3.4.2 R-Quadrado

R-Quadrado (R2):Representa a proporção da variância explicada pelo seu modelo. Neste trabalho ela foiutilizada principalmente na fase de seleção do modelo, e é uma métrica relativa utilizada na comparação entre modelos. Em linhas gerais, um modelo que possui um R2 mais próximo de 1 pode ser visto como um modelo melhor para o caso onde se está aplicando. Um modelo que não explica nenhuma variância teria um R2 de 0.Um modelo com um R2 de 1 explicaria toda a variância.No entanto, se seu R2 for 1 em seu conjunto de teste, as informações utilizadas podem estar com problemas com suas informações ou o problema é bastante simples para o seu modelo aprender,

$$\mathrm{r}^2 = 1 - \frac{\sum_{j=1}^{n}(\mathrm{y_j} - \hat{\mathrm{y}}_\mathrm{j})^2}{\sum_{j=1}^{n}(\mathrm{y_j} - \bar{\mathrm{y}})^2}. \tag{13}$$

4 Experimentos e Resultados

Nesta seção vamos separar os experimentos feitos em duas fases: seleção do modelo, e análise do modelo escolhido, como foi explicitado no capítulo 3.

4.1 Seleção do modelo

Como o nome sugere e como exposto na metodologia, nesta fase foram feitos experimentos com o objetivo de encontrar o modelo que melhor encaixava com as características do nosso problema. Para tal, uma base de dados inicial foi construída seguindo as orientações contidas na metodologia. Então com o objetivo exposto, a biblioteca PyCaret foi aplicada obtendo os resultados presentes na figura 6. As colunas presentes na figura são respectivas as seguintes métricas: média do erro absoluto, média do erro ao quadrado, raiz da média do erro ao quadrado, R2, raiz da média do log do erro e a média absoluta da porcentagem do erro.

	Model	MAE	MSE	R2
et	Extra Trees Regressor	0.3586	0.3284	0.9613
catboost	CatBoost Regressor	0.4239	0.4000	0.9529
lightgbm	Light Gradient Boosting Machine	0.4345	0.4025	0.9526
rf	Random Forest Regressor	0.4419	0.4582	0.9460
xgboost	Extreme Gradient Boosting	0.5251	0.5502	0.9352
dt	Decision Tree Regressor	0.6549	1.2216	0.8560
gbr	Gradient Boosting Regressor	0.8664	1.2874	0.8484
knn	K Neighbors Regressor	0.8744	1.4761	0.8260
ada	AdaBoost Regressor	1.4130	3.0459	0.6412
br	Bayesian Ridge	1.9286	5.2784	0.3780
lar	Least Angle Regression	1.9284	5.2795	0.3779
ridge	Ridge Regression	1.9284	5.2795	0.3779
lr	Linear Regression	1.9284	5.2795	0.3779
huber	Huber Regressor	1.9190	5.3028	0.3752
omp	Orthogonal Matching Pursuit	2.1664	6.7032	0.2100
en	Elastic Net	2.3402	7.2488	0.1458
lasso	Lasso Regression	2.5131	8.4146	0.0084
llar	Lasso Least Angle Regression	2.5231	8.4899	-0.0004
par	Passive Aggressive Regressor	2.6954	11.5852	-0.3669

Figura 6 – Resultados fase seleção do modelo.

A biblioteca Pycaret fornece uma grande quantidade de métricas, facilitando o processo de analise para a busca do modelo. É importante notar que cada métrica que está na tabela é adequada a um determinado problema e situação especifica. Por exemplo, a média do erro ao quadrado (MSE) penaliza grandes erros pelo seu fator ao quadrado, em contra partida para erros pequenos, a MSE diminui o erro. É fácil observar a distorção causada pela MSE; um erro igual a 2 se torna 4 pelo termo ao quadrado, enquanto um erro de 0,2 se torna 0,04.

Nesta fase optamos por selecionar o modelo levando em consideração duas métricas, que estão descritas na seção 3.4. São elas: a média a absoluta do erro (MAE) e R2. Como foi dito, a MSE distorce os erros menores que 1. O problema de tomografia envolve alvos pequenos, então para evitar a distorção causada pela MSE, optamos pela MAE. A métrica R2 é uma métrica utilizada para comparar modelos, ela foi levada em consideração nessa fase por ser a situação onde ela se encaixa melhor. Ao todo foram testados 19 métodos de regressão. Os métodos que se adaptaram melhor ao problema, segundo as métricas em questão são os regressores Extra-Trees, CatBoost, LightGBM e Random Forest.

A figura 6, é um exemplo de como a biblioteca Pycaret ajudou nos testes para selecionar o modelo. A imagem mostra a saída que a biblioteca proporciona, com os resultados dos 19 métodos testados com o conjunto de métricas respectivo a cada modelo, cada linha é um modelo testado. As métricas foram extraídas por meio de validação cruzada usando k-fold com K configurado como 10. A validação cruzada é aplicada com o objetivo de eliminar o acaso dos testes com os modelos. Em uma situação sem validação cruzada, o modelo é treinado com uma parte dos dados, os outros dados são utilizados como validação (ou teste) para verificar como o modelo performa na prática. Não podemos treinar e testar com os mesmos dados pois o modelo de aprendizado de máquina, se for complexo o suficiente, pode aprender todas as respostas (alvo) do conjunto de treino. O modelo vai demonstrar boa performance quando na realidade ele apenas memorizou as respostas, esse problema é conhecido como overfitting. Ainda com os dados separados em treino e validação pode acontecer do modelo ser treinado com um conjunto especifico de dados onde ele performa melhor, mesmo não aprendendo muito bem as características dos dados. Por exemplo, as amostras mais difíceis para o modelo reconhecer podem ter sido selecionadas em sua parte de treino, e nos testes ficaram apenas as amostras de fácil identificação. Com isso o modelo apresentaria uma boa performance. Para garantir que capturamos as métricas o mais próximo da performance real dos modelos, utilizamos a validação cruzada. Com a validação cruzada k-fold configurado em 10, os dados disponíveis são divididos em 10 partes de tamanho igual selecionada de maneira aleatória. O modelo treina com nove partes e valida com a parte restante. As métricas são gravadas e então o processo é repetido mais 9 vezes. A cada vez o modelo é treinado do início e as métricas são gravadas. Quando se alcança o final das 10 repartições, é formada as métricas que foram gravadas nas iterações calculando a média. As métricas apresentadas na imagem, são, portanto, a média do resultado de 10 iterações da validação k-fold.

Como pode ser observado na primeira linha da tabela representada pela imagem 6, o regressor Extra-trees (Marcado poret na primeira coluna) obteve 0.3586 de média de erro absoluto. O método Extra-Trees obteve a melhor performance em todas as métricas proporcionada pela biblioteca pycaret. Adicionando este fato com os motivos apresentados na fundamentação teórica, o método escolhido foi então o Extra-Trees. Percebe-se também a relevância dos métodos ensemble mostrando boas métricas na tabela. Os métodos por conjunto testados nessa etapa, seguindo a abreviação da nomenclatura contida na primeira coluna, são: et, catboost, lightgbm, rf, xgboost, gbr, ada. Verifica-se na tabela que estes ocupam a parte superior da tabela, como demonstrado na fundamentação os métodos por conjunto são flexíveis e de fácil aplicação, o que justifica a boa performance desses modelos em relação a modelos classicos de aprendizado de máquina como a árvore de decisão (**dt**), Neighbors (**knn**) e Linear Regression (**lr**).

Com o modelo Extra-Trees escolhido, e com os resultados verificados. Vamos na próxima seção continuar aprofundando o problema, a partir desse momento do trabalho não vamos mais nos preocupar com os outros modelos, nossas análises tem o enfoque de aprimorar o entendimento do problema de tomografia do hamiltoniano quântico de dois qubits e o modelo Extra-Trees.

4.2 Análise do Modelo

Nesta fase começamos a explorar as características do problema de tomografia com o objetivo de aplicar de maneira eficiente o modelo de aprendizado de máquina e desenvolver maior entendimento sobre o problema. Para solucionar a questão foi feito o desenvolvimento e testes do modelo selecionado, o algoritmo de aprendizado de máquina por conjunto Extra-Trees. Essa fase é considerada a análise do modelo, onde foi implementado o modelo independente com outra biblioteca chamada scikit-learn, obtendo maior controle sobre o modelo selecionado na fase anterior. Análises foram realizadas com base no modelo construído com a biblioteca e vamos expor nesta seção.

4.2.1 Análise do Tempo: Determinando o Tempo das medidas

O primeiro aspecto analisado é o tempo. No problema de tomografia do hamiltoniano, o tempo não participa da equação 9, o hamiltoniano permanece o mesmo e a dinâmica do tempo é inserida na evolução com o operador $U(t)$ na eq.3 que participa do $\rho(t)$ na eq. 4, utilizada para calcular as observáveis representadas pela eq. 7. As observáveis fazem o papel das características no modelo de aprendizado de máquina, sendo as responsáveis por determinar a variável alvo. Qual o melhor conjunto de tempo para utilizar na solução do problema, é então a parte principal do problema de tomografia que estamos solucionando nesse projeto.

Até o ponto desta análise, foi escolhido o intervalo de 1 a 10 com passo de 1 para testar a efetividade do problema. Nesta análise, foi feito o calculo das novas características e foi testado tanto separadamente como combinações dos conjuntos para definir o melhor conjunto ou melhores conjuntos para resolução do problema. Os testes foram realizados utilizando validação cruzada k-fold com k=10, em uma base de dados contendo 400.000 elementos aleatórios, de forma a garantir que os resultados sejam sólidos. A validação k-fold treina o modelo em uma das K partes disponibilizada para validação, após o treinamento as k-1 partes restantes são utilizadas para extração da métrica desejada - no caso, porcentagem do erro médio absoluto (MAPE)-, após a métrica aferida o modelo é apagado e o processo acontece novamente com a próxima das k partes. Modelos de aprendizado de máquina são suscetíveis ao acaso, a validação k-fold é realizada para garantir que estamos comparando os resultados dos conjuntos de tempo de uma maneira justa, caso contrário poderia ocorrer de compararmos o melhor resultado de um conjunto com o pior cenário de outro conjunto, prejudicando assim as conclusões sobre os resultados.

Como dito anteriormente, a fim de gerarmos o conjunto de características, medidas são feitas em tempos específicos sobre os qubits. Naturalmente, podemos escolher os tempos da dinâmica que iremos efetuar tais medidas, de forma que os resultados de teste e treinamento podem variar dependendo dos tempos escolhidos. Aqui utilizamos 4 conjuntos de tempos distintos para a análise em questão. No caso, os quatro grupos de tempos selecionados foram;

de 1 a 2 com passo de 0.1; 1 a 10 com passo de 1; 50 a 500 com passo de 50; e 100 a 1000 com passo de 100. Todos os grupos possuem 10 elementos. O critério da escolha dos grupos foi feita buscando contemplar diversas ordens de grandeza.

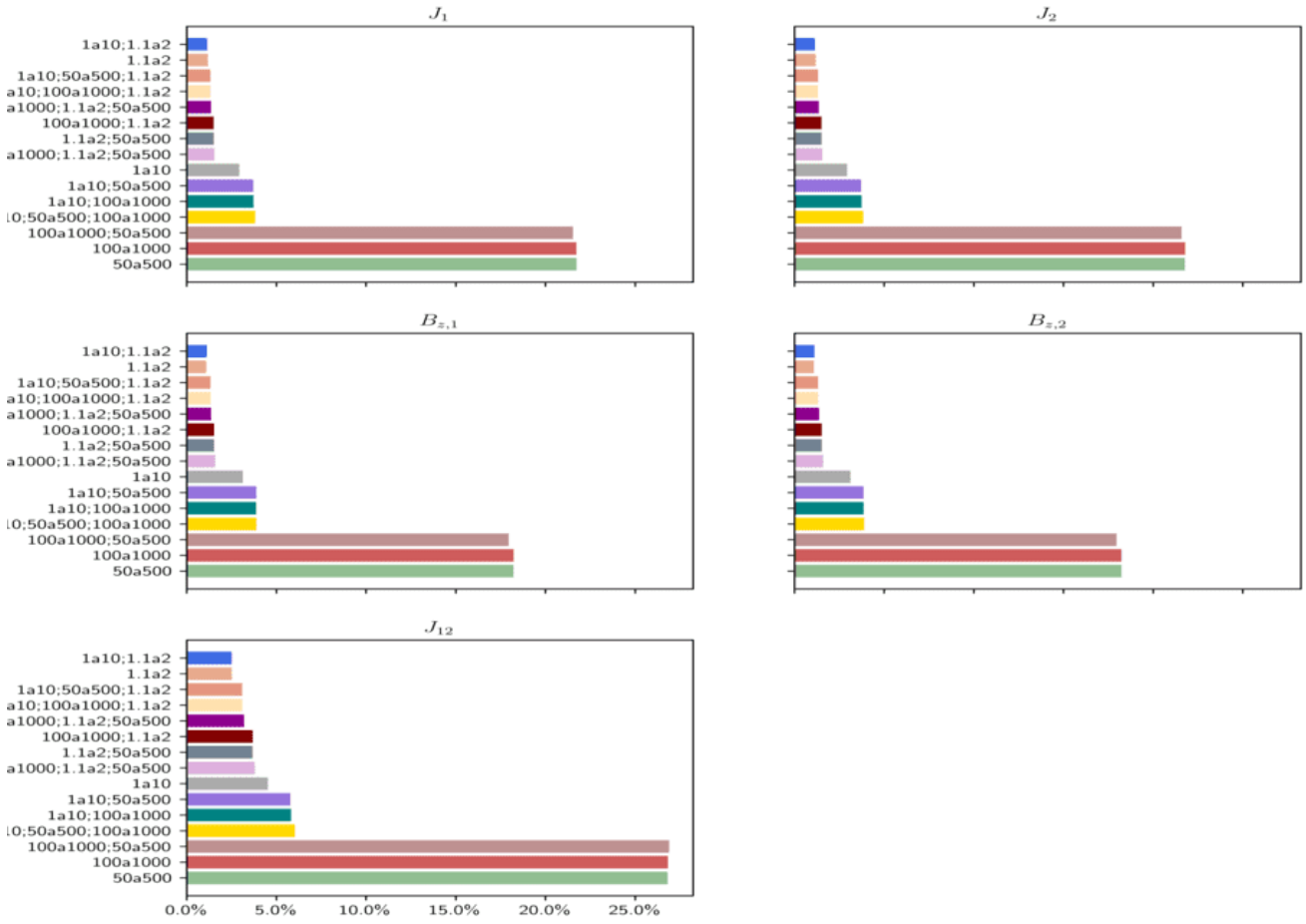

Figura 7 – MAPE conforme o tempo utilizado

Como pode ser observado nos gráficos da figura 7, em todos os alvos que estamos tratando neste problema, a melhor escolha de tempo é 1 a 10 juntamente com 1.1 a 2. Para os testes foram retirados as características sobrepostas, de maneira que de 1 a 10 está com passos de 1 enquanto 1.1 a 2 possui passos de 0.1, as medições no tempo 2 seriam sobrepostas por este motivo foram retiradas. Apesar do melhor conjunto de tempo, segundo análise gráfica, ser de 1 a 10 com 1.1 a 2, o grupo em questão não apresenta uma melhora significativa comparado com apenas capturar o grupo 1.1 a 2 isoladamente. Portanto, o grupo de tempo que será utilizado no restante do projeto será de 1.1 a 2 com passos de 0.1.

4.2.2 Análise de influência do tamanho da Base de Dados

O objetivo da análise desta subseção é determinar o tamanho adequado para a base de dados para que se consiga solucionar o problema de modo eficiente. Em problemas de aprendizado de máquina, ter mais dados para solucionar um dado problema sempre é bem vindo e normalmente é a melhor alternativa quando se deseja obter uma melhor performance no modelo (CHOLLET, 2017). Em compensação, no problema que estamos tratando, uma base menor exige menos poder computacional e é mais fácil e mais rápido para o físico experimental obter. Levando em consideração os pontos levantados, a análise busca obter o ponto ótimo entre performance e tamanho da base.

Os resultados presentes no gráfico da figura 8 foram gerados utilizando uma base de dados construída em um intervalo variando no espaço de $[1, 5]$, de maneira aleatória e como probabilidades uniforme para todos os alvos; J_1, J_2, $B_{z,1}$, $B_{z,2}$ e J_{12}. O eixo y indica a porcentagem do erro, levando em consideração que para cada alvo é construído um modelo distinto, então a métrica porcentagem do erro absoluto (MAPE), é extraída de forma independente para todos os alvos considerados. O eixo x representa o tamanho da base de dados utilizada durante a fase de treinamento. A MAPE apresentada na figura é medida na base de validação, por ser a base que representa uma situação mais próxima à situação real do físico experimental.

Como pode ser observado, as curvas seguem um padrão de descendência conforme o tamanho da base aumenta. Trata-se de um comportamento normal em aprendizado de máquina, o que demonstra que o modelo proposto está aumentando sua capacidade de representação conforme os dados que são apresentados para o modelo aumenta. O alvo J_{12}, representado pela linha roxa, é o que o modelo tem mais dificuldade em predizer, começando com um erro de 4% e passando dos 2% quando se utiliza um pouco menos de 750.000 amostras. Note que este é o único elemento que infere uma interação entre os dois qubits retratados pelo hamiltoniano do sistema quântico. Visto que as medidas efetuadas para obtenção das características são locais, é naturalque sua determinação seja mais trabalhosa.

Os outros 4 alvos começam agrupados com um erro entre 2% e 1.5%, e se separam em dois grupos distintos. O grupo que obteve melhor performance é o que contém os alvos $B_{z,1}$ e $B_{z,2}$, na equação 9 eles atuam no qubit 1 e 2 respectivamente, eles atuam em operações com σ_x. O grupo restante é composto por J_1 e J_2, eles atuam no qubit 1 e 2 respectivamente. Os alvos que começam com J na equação, atuam em operações com σ_z. Notadamente as métricas de $B_{z,1}$ e $B_{z,2}$ são semelhantes, assim como J_1 e J_2, portanto os resultados encontram-se sobrepostos e apenas uma das linhas dos respectivos grupos estão aparente no gráfico, mas devem ser interpretadas como duas pois apenas estão sobrepostas. O mesmo comportamento pode ser admitido na figura 9.

Com a análise, foi demonstrado que com 0.75×10^6 amostras consegue-se obter uma

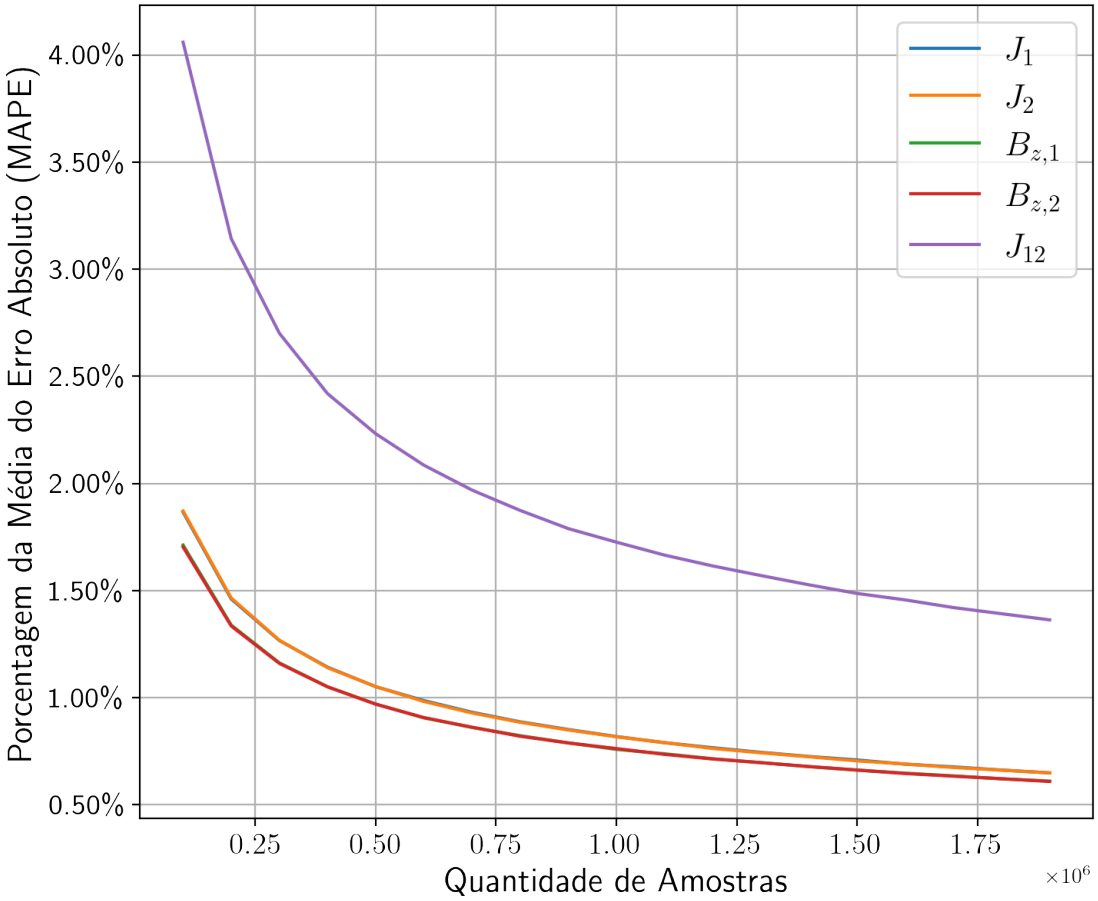

Figura 8 – Influência da Quantidade de Elementos no Treinamento

performance de aproximadamente 2% nas predições de J_{12} e menores do que 1% para os outros parâmetros. Pode ser observado a diminuição no ganho de performance conforme o número de amostras aumenta; de 0.1×10^6 até 0.75×10^6 amostras a MAPE reduz pela metade aproximadamente, enquanto que, respeitando o tamanho do intervalo, de 0.75×10^6 até 1.5×10^6 temos uma redução de aproximadamente 0.5%. Note que os resultados ainda não são os resultados finais, a MAPE ainda terá uma redução nas próximas análises que serão apresentadas na próxima seção. De fato, este resultado deve ser levado em conta apenas como um guia para balizarmos a quantidade de amostras necessárias para a nossa análise.

Assim, recapitulando o que foi exposto, até o momento temos: o modelo de aprendizado de máquina Extra-Trees se mostrou mais apto para a solução do problema de determinação da variável J na equação 5, dentre 19 métodos testados; O escopo do problema foi ampliado para a equação mais realista 9, capturando todas as variáveis da mesma, J_1, J_2, $B_{z,1}$, $B_{z,2}$, J_{12} formando então a tomografia completa do hamiltoniano.

Foi mostrado nessa seção que levando em consideração as três formas de calcular as observáveis e levando em consideração os dois qubits, conseguimos obter uma performance abaixo de 5% utilizando apenas 0.1×10^6 amostras para treino. Ainda, para obter a performance abaixo de 5% foi utilizado 6 características para cada tempo de medição. Os tempos variam de 1.1 à 2 com passos de 0.1. Seguindo estes passos, obtemos um total de 60 característica por amostra. A próxima análise, então, busca diminuir o número de características do modelo. Utilizar apenas as características essenciais facilita a predição do modelo, o que pode resultar em uma melhor performance, além de facilitar o desenvolvimento em situações práticas.

Na próxima seção vamos prosseguir analisando as características relevantes para determinar cada alvo, com o objetivo de melhorar o entendimento sobre o problema para aumentar a precisão do modelo. Em termos práticos, o nosso modelo ainda utiliza as observáveis de 3 modos diferentes para os 2 qubits em todas as ocasiões. Vale frisar que para cada medida, um custo experimental maior é necessário. Com a próxima análise buscamos então reduzir o número de medidas necessárias para a análise, para que o físico experimental apenas precise medir o seu sistema poucas vezes.

4.2.3 Análise: Importância das Características

A importância de características tem como objetivo estudar a influência de cada característica no alvo. Existem várias formas de realizar a análise de importância, nesse projeto focamos em uma análise prática dando um enfoque em auxiliar as decisões do físico experimental. Seguindo a linha mais prática, buscamos então demonstrar como possuir certas características e retirar outras vai impactar no resultado final da predição. Desta forma, o experimental terá uma noção da confiabilidade dos resultados dado as características que ele possui em mãos.

Em aprendizado de máquina, dependendo do tipo de modelo utilizado, podemos ter a possibilidade de calcular a importância de cada características no próprio modelo, por exemplo; em uma regressão linear com a equação de retas $y = w \times x + b$, a importância da característica x pode ser medida pelo próprio peso w que a multiplica quando se deseja determinar o alvo y. No caso proposto neste projeto, não temos uma equação de retas como representação final do modelo, temos um conjunto de árvores dado pelo algoritmo Extra-Trees. Uma das vantagens em utilizar árvores é a possibilidade de utilizar a estrutura final da árvore para calcular a importância das características. O modo de medir a importância das características com árvores é baseado na impuridade. A importância da característica é computada somando a redução do erro quadrado que a característica proporciona, a redução é então normalizada para ser transformada em uma porcentagem mais acessível.

A figura 9 traz as 15 características mais importantes para cada modelo em porcentagem. Os nomes das características foram dados com o seguinte protocolo: A primeira letra significa observável; A segunda letra é referente a uma das três possibilidades (x, y, z) de matriz de Pauli utilizada na equação da observável; O terceiro caractere é um número respectivo ao

qubit que a observável foi tomada; Os caracteres restantes $(T * .)$ faz referência ao tempo em que a observável foi medida, sendo os tempos totais de 1.1 a 2 com passos de 0.1. Com a imagem fica evidente que as primeiras características importam muito mais que as outras para determinar o alvo respectivo. Do ponto de vista prático, os gráficos sugerem que é possível reduzir o numero de características consideravelmente, lembrando que estamos utilizando 60 características no total e o gráfico apenas traz as 15 melhores.

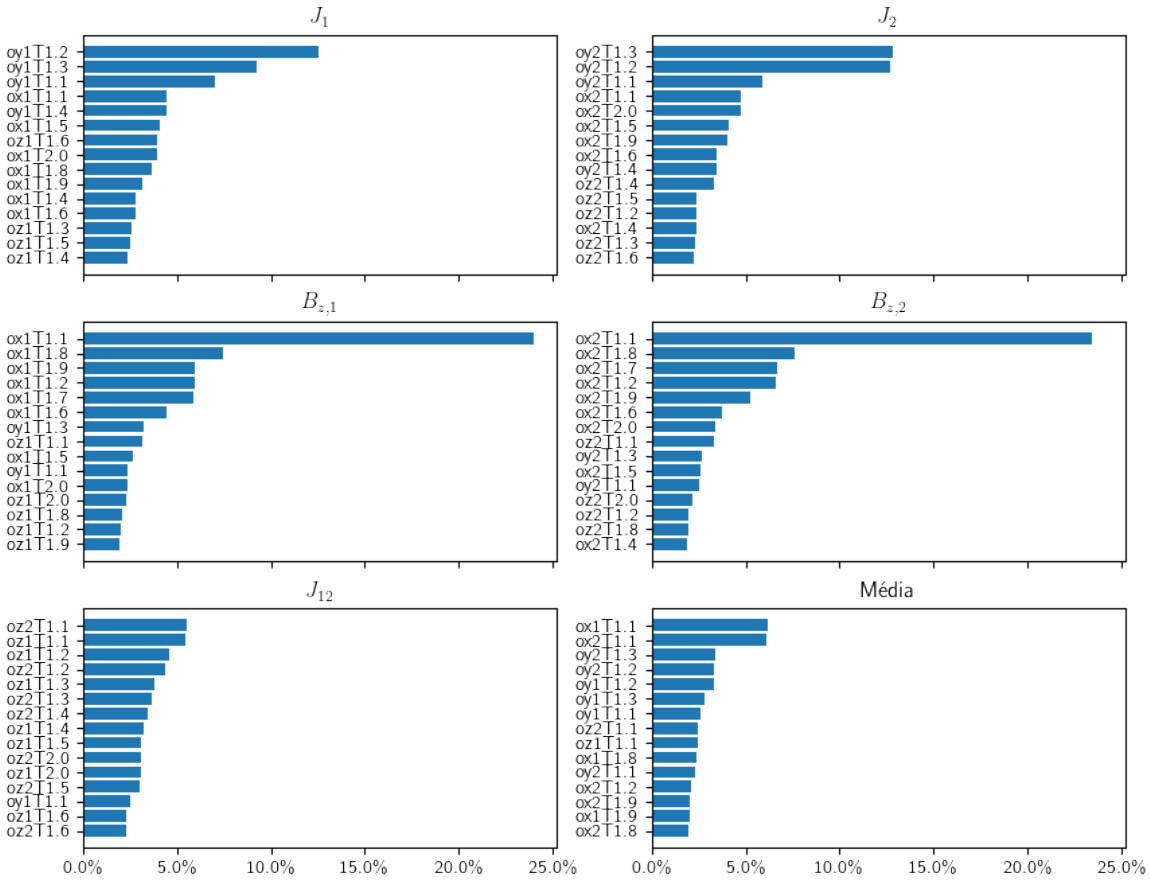

Figura 9 – Importância das Características: As 15 características mais significativas para o cada modelo

Visualizar as 60 características em uma única vez não seria prático e, mesmo visualizando, às vezes não é possível identificar os padrões implícitos. A figura 10 organiza o gráfico de características somando e agrupando as importâncias em categorias. As categorias selecionadas foram; qubit-1 ou qubit-2, sendo as features extraídas do qubit 1 ou 2; *ox*, *oz*, *oy*, as observáveis referentes as três matrizes de Pauli. No total foram cinco grupos. Separar em categorias coloca em evidência que o qubit 1, de fato, tem um papel de maior importância sobre o campo que atua sobre ele, como o modelo para J_1, $B_{z,1}$. Aproveitando, pelo mesmo motivo o qubit 2

recebe destaque em J_2 e $B_{z,2}$. Completando a análise sobre as características referente aos qubits, em $J_{1,2}$ os qubits 1 e 2 receberam igual importância, visto que é um modelo onde existem operações nos dois qubits. Estes pontos podem parecer óbvios tendo em mãos a equação do hamiltoniano, mas o modelo foi treinado apenas com as observáveis. É importantes garantir que, de fato, o modelo está conseguindo capturar o real significado do problema que estamos trabalhando, medindo a importância que o modelo dá as características. Esta é uma forma de validar a abordagem deste projeto, pois sabemos agora que o modelo e o hamiltoniano estão condizentes. Analisando as características referentes às matrizes de Pauli, os modelos que buscam prever J_1 e J_2 consideram mais importante medir o eixo Y e X, o Z influencia menos na decisão para a predição. Para $B_{z,1}$ e $B_{z,2}$ o eixo X é mais relevante, enquanto que o Z e Y tem menor importância na predição. Na análise das Médias temos que o qubit 1 e 2 receberam importâncias iguais como podia ser previsto.

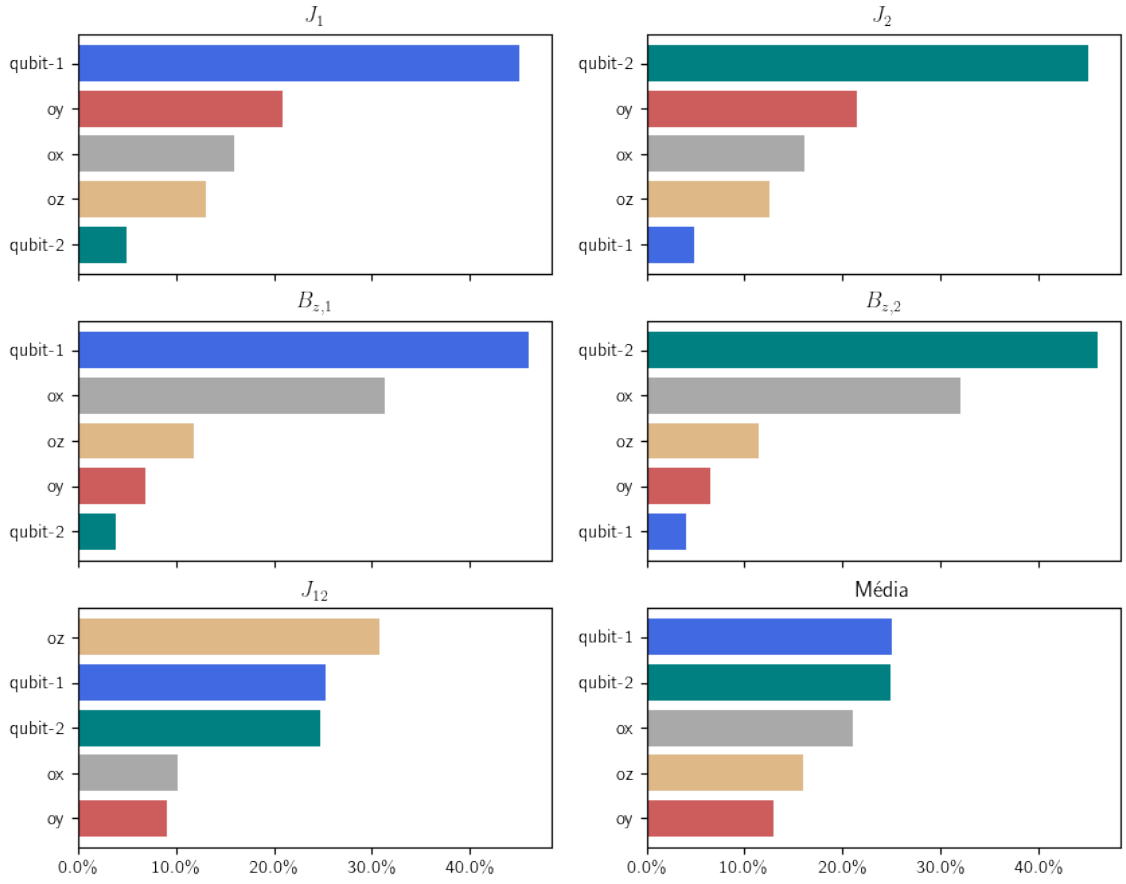

Figura 10 – Importância das Características: As categorias para cada modelo

De modo geral, todas características desempenham um papel importante em algum momento do modelo. Apenas essa análise não é o suficiente para descartar qualquer carac-

terística. Será dado continuidade aos testes, ainda considerando uma abordagem prática na próxima subseção. Ainda sobre o tópico características, nas próximas subseções será analisado em profundidade a questão de como os passos dos tempos, e as diferentes matrizes de Pauli, influenciam na previsão para o físico experimental. Para finalizar, na última sub-seção, sobre características do modelo, foi buscado um conjunto ótimo de características de forma eliminativa e utilizando validação cruzada.

4.2.4 Análise: Escolha das Características

Nessa seção, será demonstrado como o modelo performa em algumas situações que podem ser situações as quais o físico experimental se encontra, e de forma a complementar a seção 4.2.3 onde analisamos a importância das características, será dado um enfoque prático sobre a alteração da performance de acordo com as características selecionadas.

Para alcançar o objetivo da sub-seção foi realizado os seguintes testes: variação da performance conforme as medidas utilizadas, e variação da performance conforme a matriz de Pauli e qubit utilizada. Ambos os testes foram feitos para as cinco variáveis alvos que constituem o problema de tomografia de hamiltoniano. Os modelos foram treinados com 750.000 amostras, geradas de maneira aleatória no intervalo de 1 a 5 para todos os alvos J_1, J_2, $B_{z,1}$, $B_{z,2}$ e J_{12}. A base de teste possui 600.000 amostras formada de mesma maneira, mas com elementos disjuntos, em outras palavras, elementos que não estão presentes na base de treino. Para cada conjunto de característica testado foi treinado um modelo novo com a base de treino e fazendo o recorte necessário da base de treino com todas as colunas de características. Da mesma maneira, para cada alvo é necessário um novo modelo. Todas as métricas apresentadas foram medidas na base de teste, para garantir um resultado mais próximo de uma situação real.

O teste de performance segundo as medidas vistas pelo modelo, levam em consideração dois aspectos. Para o físico experimental, quanto menos medidas ele puder determinar o hamiltoniano menos trabalho é necessário e mais simples e prático. Para treinar um modelo de aprendizado de máquina, o modelo pode não desempenhar bem quando se tem muitas características, e é computacionalmente mais dispendioso. A figura 11, demonstra, separando para cada alvo, a performance da métrica MAPE conforme os tempos de medida utilizados. As medidas de tempo utilizadas variam de 1.1 a 2 com passo de 0.1, então o gráfico começa com o erro utilizando apenas o tempo 1.1 como medida. O segundo ponto no eixo x é o 2, o que designa 2 medidas, portanto as medidas consideradas são 1.1 e 1.2, o ponto no eixo x igual a 4, considera 4 medidas, efetuadas nos tempos 1.1, 1.2, 1.3 e 1.4, e assim por diante para outros valores de x. O gráfico demonstra que considerando apenas duas medidas, 1.1 e 1.2, é possível determinar com uma precisão de 2.5% todas as variáveis alvo com exceção de J_{12}. Conforme mostrado na subseção 4.2.2 esta é a variável mais difícil de determinar, alcançando o nível de 2.5% apenas quando se utilizando 6 medidas. Com o gráfico percebe-se a existência de um ponto ótimo em 4 medidas para os alvos J_1, J_2, $B_{z,1}$ e $B_{z,2}$, de forma que medir os tempos

restantes até 2 não se mostra vantajoso para estes alvos. Com 6 medidas é possível determinar com uma MAPE de 2.5% todos os alvos para a tomográfica completa do hamiltoniano.

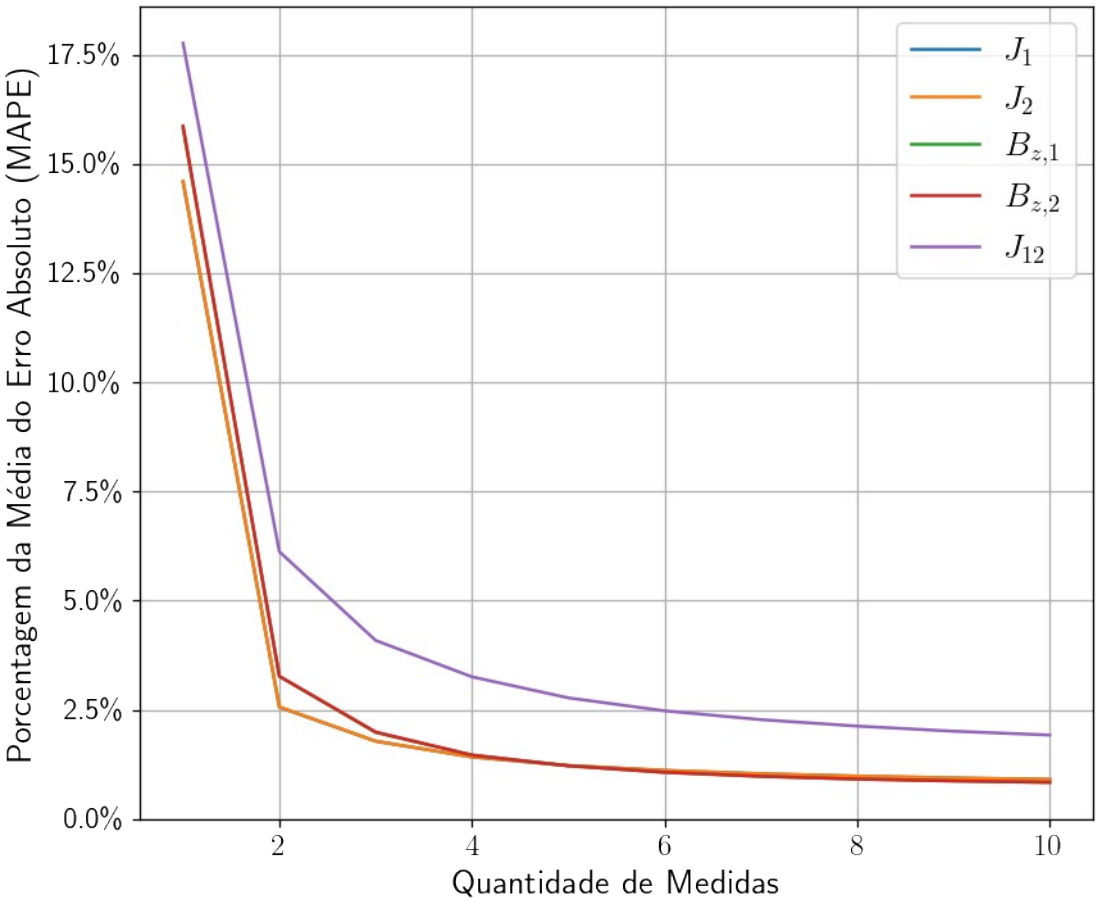

Figura 11 – Escolha das Características: Variação da MAPE Conforme medidas

Complementando a 4.2.3, foi realizado o teste da variação da performance conforme a matriz de Pauli e qubit utilizado. Em situações práticas pode se ter apenas disponível algumas características, a análise em questão busca responder a precisão que se pode ter utilizando apenas alguns grupos de características. A análise confirma a importância das características demonstrada na sub-seção anterior dando as métricas práticas para cada grupo de característica. O gráfico presente na figura 12, demonstra para cada alvo, qual a precisão da MAPE que se deve esperar para cada grupo de características destacado. Os grupos selecionados são: apenas o qubit 1, ou apenas qubit 2, ou utilizando apenas uma das 3 matrizes de Pauli (σx, σy, σz). Note que, confirmando os indícios da importância de características da subseção anterior, os alvos dependentes do qubit 1 ou qubit 2 mostram uma performance semelhante a de se utilizar a tabela completa de características. Em outras palavras, utilizando apenas as medidas realizadas no qubit que atua no alvo desejado é o bastante para determinar o mesmo. O gráfico

demonstra que é possível alcançar uma performance abaixo de 5% em no mínimo 2 ocasiões para todos os alvos.

Com as análises da sub-seção, conseguimos ter uma noção prática do percentual do erro absoluto em várias ocasiões. Os resultados apresentados podem ser utilizados como guia para os experimentais obterem a performance desejada, nas diversas situações práticas. A próxima análise aprofunda os testes das características, buscando encontrar o conjunto ótimo de características para cada alvo.

4.2.5 Análise: Conjunto Ótimo

Para buscar um conjunto ótimo dentre as características, foi ampliado os testes feitos na sub-seção 4.2.4. Nessa sub-seção foi mantido os mesmos aspectos nos dados de treino e teste, porém o conjunto de características testados foi significativamente maior. Na sub-seção anterior testamos a influência isolada das características, nessa subseção, como o foco é buscado o conjunto ótimo, foi testada um amplo conjunto de características.

A figura 13 demonstra os gráficos da MAPE separada por alvo, e seguindo a mesma nomenclatura utilizada para as características em todo esse projeto; o primeiro caractere, B, significava observável; o segundo caractere podem ser 3 opções, x, y e z, cada um correspondente a matriz de Pauli utilizada; o primeiro número significa o qubit medido. Quando o número do qubit é omitido, significa que ambos os qubits foram levados em consideração. Como podemos observar, com exceção de $B_{z,1}$ e $B_{z,2}$, o conjunto de características completo, ou seja, envolvendo as medidas x, y e z nos dois qubits, é o mais significativo. Para $B_{z,1}$ e $B_{z,2}$ notamos que somente as medidas y e z, em seus respectivos qubits, são as que apresentam melhor resultado. Apesar do qubit 1 ou o 2 ser mais significativo, os gráficos indicam que possuem situações onde olhar para o outro qubit é um critério relevante ainda. Isso acontece pois o termo $\sigma_z \otimes \sigma_z$ do hamiltoniano cria um canal de comunicação entre os qubits, de forma que uma medida no qubit 2, pode trazer informações sobre o qubit 1 e vice-versa.

As características foram testadas e analisadas de inúmeras formas nas subseções anteriores e nesta, e foi demonstrado que os tempos variando de 0.1 a 2, incluindo as duas extremidades, e as características com as 3 matrizes de Pauli e com os dois qubits estão de acordo com a resolução do problema. Na próxima seção, para finalizar as análises será utilizada outro modo de visualizar o problema. Agora com suas características bem definidas, será demonstrado como o modelo performa conforme a variação no alvo.

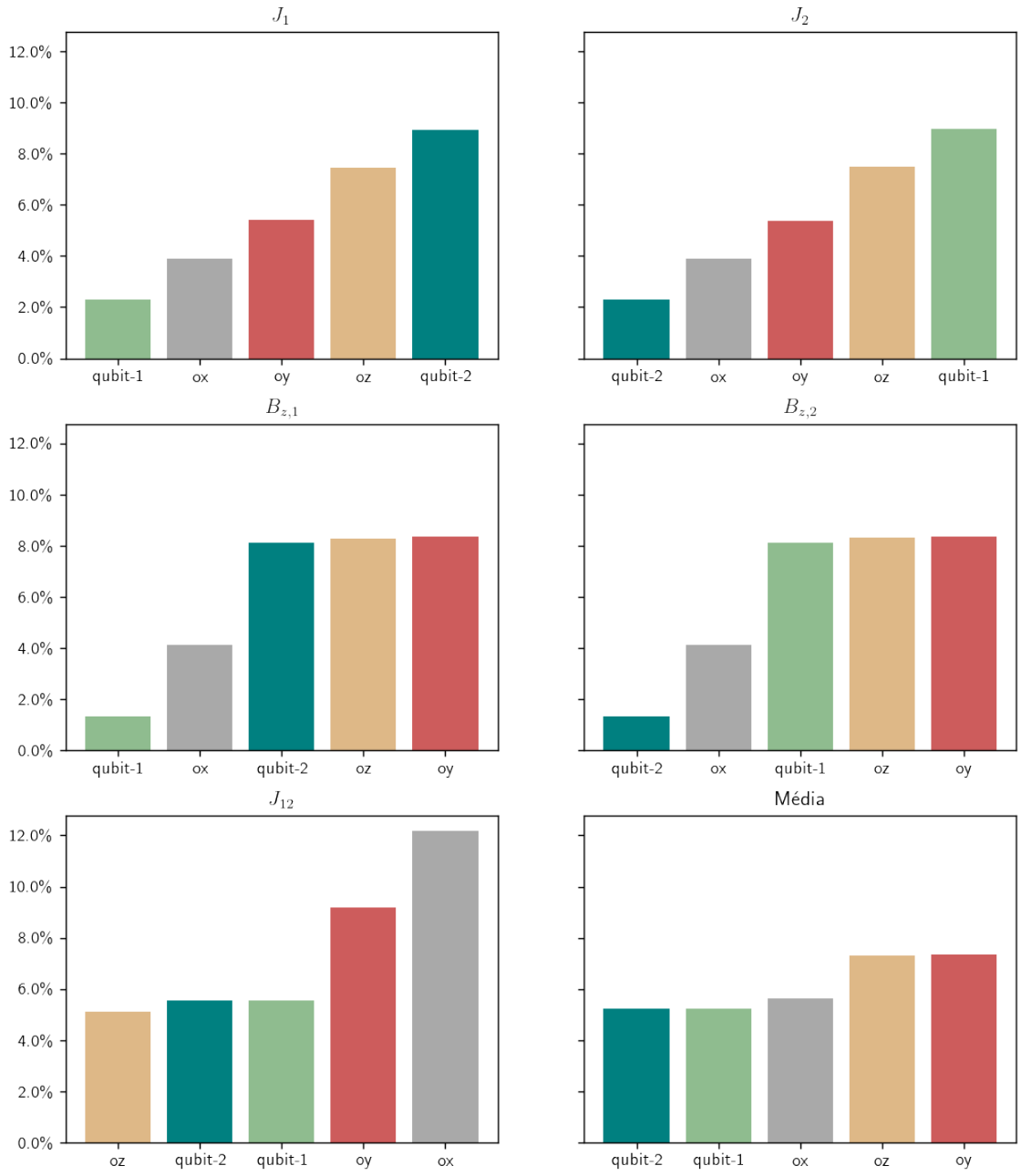

Figura 12 –Escolha das Características: Variação da MAPE Conforme categoria de características

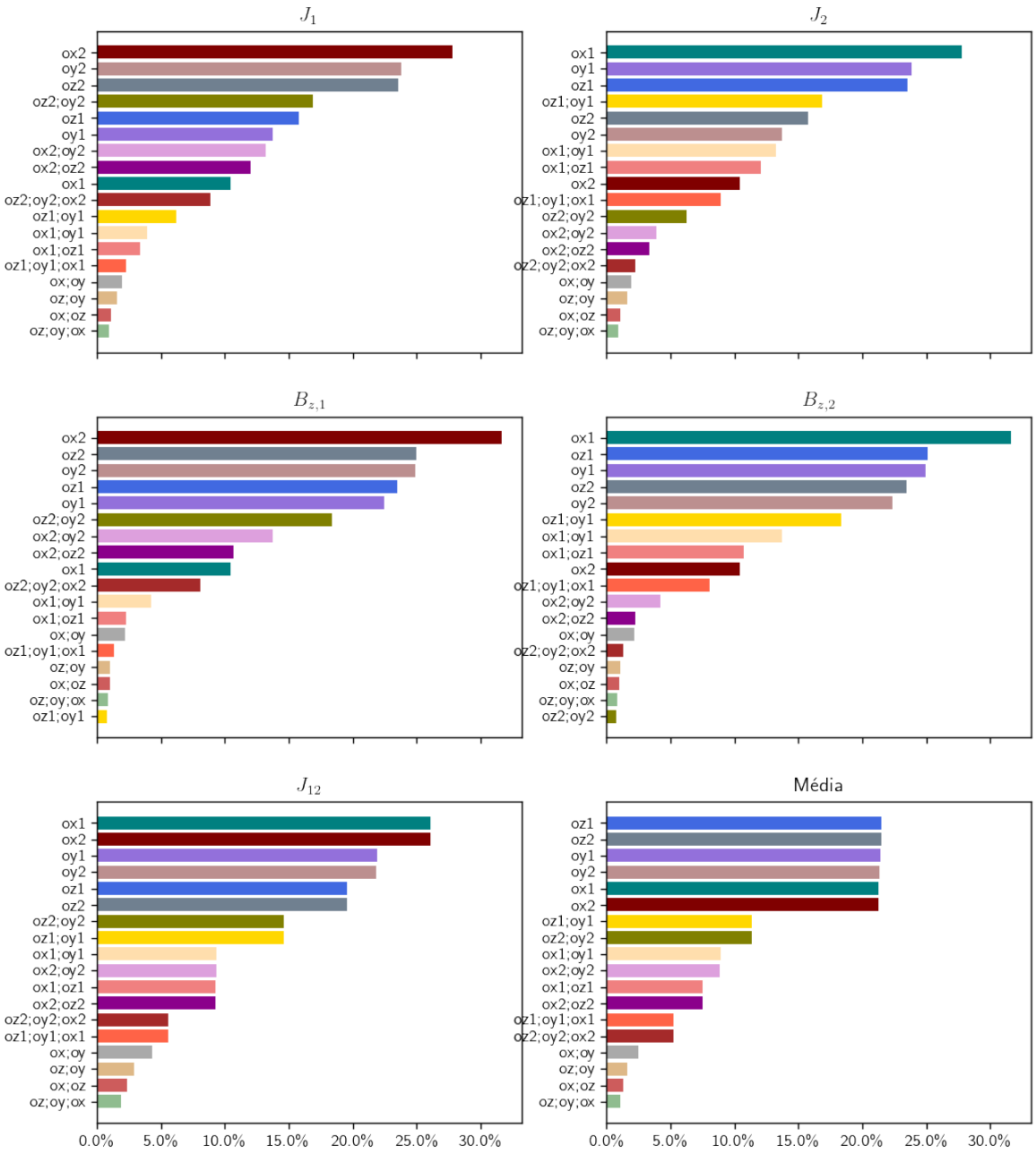

Figura 13 –Escolha das Características: Variação da MAPE Conforme combinações de características

4.2.6 Análise: Erro Conforme variação do alvo

Com os modelos de aprendizado de máquina definidos em termo das características, nessa subseção o objetivo é determinar a variação do erro pelo alvo. Para realizar a análise da subseção, os modelos foram treinados com uma base composta por 750.000 amostras geradas aleatoriamente no intervalo dos alvos. Todos os alvos estão variando em um intervalo de 1 a 5. Recapitulando, na sub-seção 4.2.2 foi demonstrado qual é a performance esperada para cada modelo utilizando uma base de treinamento com o número de amostras. Em linhas gerais, para todos os alvos com exceção de J_{12} espera-se uma média menor que 1%. Para J_{12} espera uma média de 2%. As métricas foram extraídas em uma base disjunta da base de treinamento, contendo 600.000 amostras geradas aleatoriamente no espaço em questão.

Os erros pelo alvo, mostram que em suas extremidades o erro aumenta em todos os alvos. Como o modelo utilizado é uma árvore, os erros nas extremidades podem ser maiores devido a maneira como a mesma é construída. A árvore de decisão utiliza a soma da média dos erros ao quadrado para avaliar a performance nas divisões dos nós e isso pode se torna um problema nas extremidades. De fato, nas extremidades temos menos amostras para somar na métrica de redução do erro médio ao quadrado(MSE), o que pode levar o modelo a escolher outro critério para dividir o nó da árvore.

O erro nas extremidades podem ser corrigidos utilizando pesos nas amostras da extremidade ou então apenas adicionando mais amostras para treino nas faixas de extremidade. Isso levaria o modelo a prestar mais atenção nas extremidade. Os modelos, mesmo com erros altos nas extremidades, não passam de 5% em todos os casos exceto J_{12}, onde o erro não ultrapassa 8%. Como demonstra o gráfico 14 os erros não possuem muitos desníveis e possuem níveis abaixo de 2% até mesmo para J_{12}, o alvo de maior dificuldade para predizer.

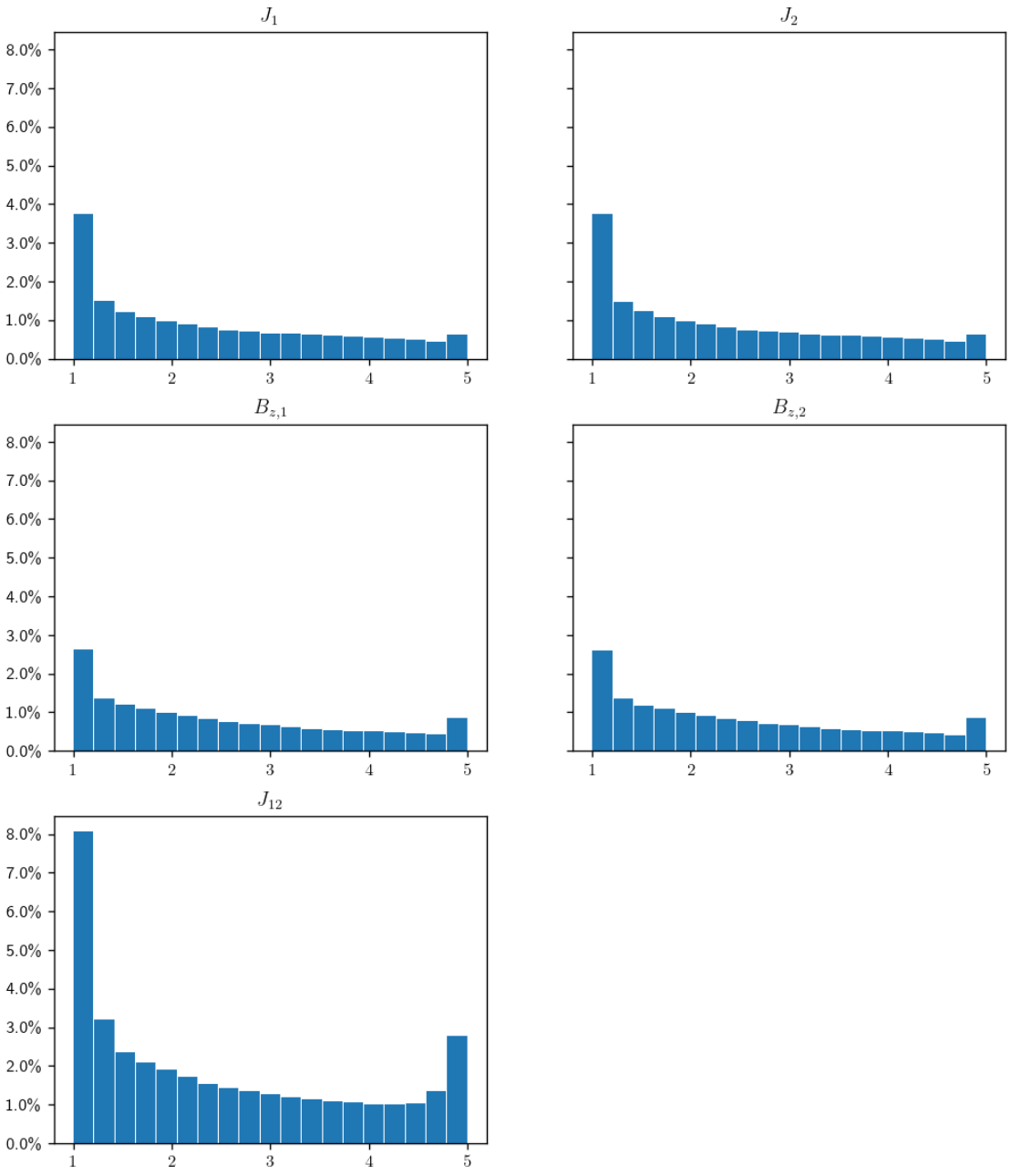

Figura 14 – Erro pelos alvos

5 Conclusão

Este projeto buscou solucionar o problema de tomografia de hamiltoniano utilizando o método de aprendizado de máquina Extra-Trees. A resolução do problema de tomografia de hamiltoniano possui potencial de auxiliar físicos experimentais em situação práticas, onde se deseja estipular os parâmetros do hamiltoniano utilizando um número mínimo de medidas sobre o sistema. As medições para estipular os parâmetros são dispendiosas e a cada medição existe a necessidade de restaurar todo o sistema novamente para efetuar uma nova medida. O projeto desenvolvido, demonstrou que é possível, com apenas 36 medidas locais totais (3 eixos, 2 qubits e 6 tempos), determinar com precisão todos os parâmetros que definem o hamiltoniano, o que economiza recursos em tarefas experimentais.

Ainda em tempo, o manuscrito descreveu em detalhes os passos para elaboração do modelo de aprendizado de máquina. Começando pelas equações utilizadas para descrever o sistema, formação das tabelas para treinamento e teste, escolha do modelo, escolha do tempo de medida das observáveis do hamiltoniano, escolha das características e análise do erro, demonstrando minunciosamente a importância de cada aspecto do problema em questão. O modelo foi testado de maneira exaustiva utilizando técnica de validação cruzada k-fold, 19 modelos de aprendizado de máquina dentre eles support vector machine, random forest e KNN. Ainda utilizando validação cruzada, foi testado mais de 18 conjuntos de características para cada um dos parâmetros do hamiltoniano. Sobre os tempos de medida, foram testados 4 grupos distintos de tempos e suas combinações e, após a escolha do melhor conjunto, foi testado a performance da utilização dos mesmos até sua totalidade de 10 medições. Para cada um dos 5 modelos foi explorada a importância das características, assim como a performance de cada uma no modelo. Por fim, foi exposto o erro por cada um dos alvos. Todos os testes mencionados foram feitos com o propósito de serem utilizados como guia para o físico experimental ter ferramentas para desenvolver o modelo que mais atenda seus propósitos práticos.

Para futuro direcionamento deste projeto é proposto utilização de métodos de aprendizado profundo. O projeto conseguiu estabelecer uma boa base de referência para solução do problema de tomografia. A partir dessas bases estabelecidas no projeto, e com a análise de características, é possível buscar novas soluções com métodos de maior complexidade, sobretudo os métodos de aprendizado profundo, como as redes neurais recorrentes, que ganharam destaque em tempos recentes. Elas conseguem capturar os aspectos temporais das características, tendo um bom potencial para o problema em questão.

O projeto buscou estabelecer um guia para o desenvolvimento de modelo para a tomografia de hamiltoniano. Hoje em dia, com os avanços na computação quântica é possível treinar o modelo com amostras realistas, fazendo uso de computadores quânticos. Além disso,

atualmente temos modelos teóricos para simularmos os erros dos experimentos de forma que uma análise mais realista também pode ser considerada. Naturalmente, quanto mais próximo os dados forem da realidade do experimental, melhor será o treinamento do modelo. Sendo assim, este é um outro aspecto para um futuro direcionamento do projeto.

Referências

ALELYANI, S. Stable bagging feature selection on medical data. *Journal of Big Data*, v. 8, 01 2021.

ALI, M. *PyCaret: An open source, low-code machine learning library in Python*. [S.l.], 2020. PyCaret version 2.3. Disponível em: <https://www.pycaret.org>.

ALKAWAZ, A.; ABDELLATIF, A.; KANESAN, J.; KHAIRUDDIN, A.; GHENI, H. Day-ahead electricity price forecasting based on hybrid regression model. *IEEE Access*, 10 2022.

ALSHDAIFAT, E.; AL-HASSAN, M.; AL-OQAILY, A. Effective heterogeneous ensemble classification: An alternative approach for selecting base classifiers. *ICT Express*, v. 7, 12 2020.

AQEEL, I.; MAJID, A. *Hybrid Approach to Identify Druglikeness Leading Compounds against COVID-19 3CL Protease*. arXiv, 2022. Disponível em: <https://arxiv.org/abs/2208.06362>.

ARCARI, M.; SÖLLNER, I.; JAVADI, A.; HANSEN, S. L.; MAHMOODIAN, S.; LIU, J.; THYRRESTRUP, H.; LEE, E. H.; SONG, J. D.; STOBBE, S.; LODAHL, P. Near-unity coupling efficiency of a quantum emitter to a photonic crystal waveguide. *Phys. Rev. Lett.*, American Physical Society, v. 113, p. 093603, Aug 2014. Disponível em: <https://link.aps.org/doi/10.1103/PhysRevLett.113.093603>.

BELL, R. M.; KOREN, Y. Lessons from the netflix prize challenge. *SIGKDD Explor. Newsl.*, Association for Computing Machinery, New York, NY, USA, v. 9, n. 2, p. 75–79, dec 2007. ISSN 1931-0145. Disponível em: <https://doi.org/10.1145/1345448.1345465>.

BELL, R. M.; KOREN, Y.; VOLINSKY, C. All together now: A perspective on the netflix prize. *CHANCE*, v. 23, p. 24 – 29, 2010.

BIAN, Y.; WANG, Y.; YAO, Y.; CHEN, H. Ensemble pruning based on objection maximization with a general distributed framework. *IEEE Transactions on Neural Networks and Learning Systems*, v. 31, n. 9, p. 3766–3774, 2020.

BREIMAN, L. Bagging predictors. *Machine Learning*, v. 24, p. 123–140, 08 1996.

BREIMAN, L. Bagging predictors. *Machine Learning*, v. 24, n. 2, p. 123–140, Aug 1996. ISSN 1573-0565. Disponível em: <https://doi.org/10.1007/BF00058655>.

BREIMAN, L. Random forests. *Machine Learning*, v. 45, n. 1, p. 5–32, Oct 2001. ISSN 1573-0565. Disponível em: <https://doi.org/10.1023/A:1010933404324>.

CASTELANO, L. K.; FANCHINI, F. F.; BERRADA, K. Open quantum system description of singlet-triplet qubits in quantum dots. *Physical Review B*, American Physical Society (APS), v. 94, n. 23, Dec 2016. ISSN 2469-9969. Disponível em: <http://dx.doi.org/10.1103/PhysRevB.94.235433>.

CHEN, T.; GUESTRIN, C. Xgboost: A scalable tree boosting system. In: . [S.l.: s.n.], 2016. p. 785–794.

CHOLLET, F. *Deep Learning with Python*. [S.l.]: Manning, 2017. ISBN 9781617294433.

DIETTERICHL, T. G. Ensemble learning. In: ARBIB, M. (Ed.). *The Handbook of Brain Theory and NeuralNetworks*. [S.l.]: MIT Press, 2002. p. 405–408.

FANCHINI, F. F.; KARPAT, G.; ROSSATTO, D. Z.; NORAMBUENA, A.; COTO, R. Estimating the degree of non-markovianity using machine learning. *PhysicalReview A*, American Physical Society (APS), v. 103, n. 2, Feb 2021. ISSN 2469-9934. Disponívelem: <http://dx.doi.org/10.1103/PhysRevA.103.022425>.

FENG, Y.; GRADWOHL, R.; HARTLINE, J.; JOHNSEN, A.; NEKIPELOV, D. Bias-variance games. In: *Proceedings of the 23rd ACM Conference on Economics and Computation*. New York, NY, USA: Association for Computing Machinery, 2022. (EC '22), p. 328–329. ISBN 9781450391504. Disponívelem: <https://doi.org/10.1145/3490486.3538248>.

FERNÁNDEZ-DELGADO, M.; CERNADAS, E.; BARRO, S.; AMORIM, D. Do we need hundreds of classifiers to solve real world classification problems? *J. Mach. Learn. Res.*, JMLR.org, v. 15, n. 1, p. 3133–3181, jan 2014. ISSN 1532-4435.

FREUND, Y.; SCHAPIRE, R. A short introduction to boosting. *Journal of Japanese Society for ArtificialIntelligence*, v. 14, p. 771–780, 10 1999.

GEURTS, P.; ERNST, D.; WEHENKEL, L. Extremely randomized trees. *Machine Learning*, v. 63, n. 1, p. 3–42, Apr 2006. ISSN 1573-0565. Disponívelem: <https://doi.org/10.1007/s10994-006-6226-1>.

GRAY, J.; BANCHI, L.; BAYAT, A.; BOSE, S. Machine-learning-assisted many-body entanglement measurement. *Phys. Rev. Lett.*, American Physical Society, v. 121, p. 150503, Oct 2018. Disponívelem: <https://link.aps.org/doi/10.1103/PhysRevLett.121.150503>.

GREENAWAY, S.; SMITH, A.; MINTERT, F.; MALZ, D. *Analogue Quantum Simulation with Fixed-Frequency Transmon Qubits*. arXiv, 2022. Disponívelem: <https://arxiv.org/abs/2211.16439>.

GREILICH, A.; CARTER, S.; KIM, D.; BRACKER, A.; GAMMON, D. Optical control of one and two hole spins in interacting quantum dots. *Nature Photonics - NAT PHOTONICS*, v. 5, 07 2011.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. Ensemble learning. In: ___. *The Elements of StatisticalLearning: Data Mining, Inference, and Prediction*. New York, NY: Springer New York, 2009. p. 605–624. ISBN 978-0-387-84858-7. Disponívelem: <https://doi.org/10.1007/978-0-387-84858-7_16>.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. Ensemble learning. In: ___. *The Elements of StatisticalLearning: Data Mining, Inference, and Prediction*. New York, NY: Springer New York, 2009. p. 605–624. ISBN 978-0-387-84858-7. Disponívelem: <https://doi.org/10.1007/978-0-387-84858-7_16>.

HOCHACHKA, W.; CARUANA, R.; FINK, D.; MUNSON, A.; RIEDEWALD, M.; SOROKINA, D.; KELLING, S. Data-mining discovery of pattern and process in ecological systems. *The Journal of Wildlife Management*, v. 71, p. 2427 – 2437, 09 2007.

HUANG, H.-Y.; KUENG, R.; PRESKILL, J. Predicting many properties of a quantum system from very few measurements. *Nature Physics*, v. 16, n. 10, p. 1050–1057, Oct 2020. ISSN 1745-2481. Disponívelem: <https://doi.org/10.1038/s41567-020-0932-7>.

HUANG, Y.; CHE, L.; WEI, C.; XU, F.; NIE, X.; LI, J.; LU, D.; XIN, T. *Measuring Quantum Entanglement from Local Information by Machine Learning*. arXiv, 2022. Disponível em: <https://arxiv.org/abs/2209.08501>.

IMAMOGLU, A.; AWSCHALOM, D. D.; BURKARD, G.; DIVINCENZO, D. P.; LOSS, D.; SHERWIN, M.; SMALL, A. Quantum information processing using quantum dot spins and cavity qed. *Phys. Rev. Lett.*, American Physical Society, v. 83, p. 4204–4207, Nov 1999. Disponível em: <https://link.aps.org/doi/10.1103/PhysRevLett.83.4204>.

ISHAQ, A.; SADIQ, S.; UMER, M.; ULLAH, S.; MIRJALILI, S.; RUPAPARA, V.; NAPPI, M. Improving the prediction of heart failure patients' survival using smote and effective data mining techniques. *IEEE Access*, v. 9, p. 39707–39716, 2021. ISSN 2169-3536.

JEFFERSON, J. H.; FEARN, M.; TIPTON, D. L. J.; SPILLER, T. P. Two-electron quantum dots as scalable qubits. *Phys. Rev. A*, American Physical Society, v. 66, p. 042328, Oct 2002. Disponível em: <https://link.aps.org/doi/10.1103/PhysRevA.66.042328>.

JORDAN, M.; MITCHELL, T. Machine learning: Trends, perspectives, and prospects. *Science (New York, N.Y.)*, v. 349, p. 255–60, 07 2015.

KADAM, V. Regression techniques in machine learning &applications: A review. *International Journal for Research in Applied Science and Engineering Technology*, v. 8, p. 826–830, 2020.

KE, G.; MENG, Q.; FINLEY, T.; WANG, T.; CHEN, W.; MA, W.; YE, Q.; LIU, T.-Y. Lightgbm: A highly efficient gradient boosting decision tree. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2017. (NIPS'17), p. 3149–3157. ISBN 9781510860964.

KUMAR, S.; KAUR, P.; GOSAIN, A. A comprehensive survey on ensemble methods. In: *2022 IEEE 7th International conference for Convergence in Technology (I2CT)*. [S.l.: s.n.], 2022. p. 1–7.

LI, N.; YU, Y.; ZHOU, Z.-H. Diversity regularized ensemble pruning. In: . [S.l.: s.n.], 2012. ISBN 978-3-642-33459-7.

LI, N.; YU, Y.; ZHOU, Z.-H. Diversity regularized ensemble pruning. In: . [S.l.: s.n.], 2012. ISBN 978-3-642-33459-7.

LLERA, A.; BRAMMER, M.; OAKLEY, B.; TILLMANN, J.; ZABIHI, M.; MEI, T.; CHARMAN, T.; ECKER, C.; ACQUA, F. D.; BANASCHEWSKI, T.; MOESSNANG, C.; BARON-COHEN, S.; HOLT, R.; DURSTON, S.; MURPHY, D.; LOTH, E.; BUITELAAR, J. K.; FLORIS, D. L.; BECKMANN, C. F. *Evaluation of data imputation strategies in complex, deeply-phenotyped data sets: the case of the EU-AIMS Longitudinal European Autism Project*. arXiv, 2022. Disponível em: <https://arxiv.org/abs/2201.09753>.

MAUNE, B. M.; BORSELLI, M. G.; HUANG, B.; LADD, T. D.; DEELMAN, P. W.; HOLABIRD, K. S.; KISELEV, A. A.; ALVARADO-RODRIGUEZ, I.; ROSS, R. S.; SCHMITZ, A. E.; SOKOLICH, M.; WATSON, C. A.; GYURE, M. F.; HUNTER, A. T. Coherent singlet-triplet oscillations in a silicon-based double quantum dot. *Nature*, v. 481, n. 7381, p. 344–347, Jan 2012. ISSN 1476-4687. Disponível em: <https://doi.org/10.1038/nature10707>.

MENDES-MOREIRA, J. a.; SOARES, C.; JORGE, A. M.; SOUSA, J. F. D. Ensemble approaches for regression: A survey. *ACM Comput. Surv.*, Association for Computing Machinery, New York, NY, USA, v. 45, n. 1, dec 2012. ISSN 0360-0300. Disponível em: <https://doi.org/10.1145/2379776.2379786>.

MERENTITIS, A.; DEBES, C. Many hands make light work - on ensemble learning techniques for data fusion in remote sensing. *Geoscience and Remote Sensing Magazine, IEEE*, v. 3, p. 86–99, 09 2015.

MIENYE, D.; OBAIDO, G.; ARULEBA, K.; DADA, O. Enhanced prediction of chronic kidney disease using feature selection and boosted classifiers. In: ____. [S.l.: s.n.], 2022. p. 527–537. ISBN 978-3-030-96307-1.

MIENYE, I. D.; SUN, Y. A survey of ensemble learning: Concepts, algorithms, applications, and prospects. *IEEE Access*, v. 10, p. 99129–99149, 2022.

MOREIRA, J.; SOARES, C.; JORGE, A.; SOUSA, J. Ensemble approaches for regression: A survey. *ACM Computing Surveys*, v. 45, p. 10:1–10:40, 11 2012.

MUNIR, M. S.; PARK, S.-B.; HONG, C. S. *An Explainable Artificial Intelligence Framework for Quality-Aware IoE Service Delivery*. arXiv, 2022. Disponível em: <https://arxiv.org/abs/2201.10822>.

OKAMOTO, R.; O'BRIEN, J. L.; HOFMANN, H. F.; NAGATA, T.; SASAKI, K.; TAKEUCHI, S. Experimental realization of an optical entanglement filter. In: *CLEO/Europe and EQEC 2009 Conference Digest*. [S.l.]: Optical Society of America, 2009. p. EA3.

OZA, N.; TUMER, K. Classifier ensembles: Select real-world applications. *Information Fusion*, v. 9, p. 4–20, 01 2008.

PATIL, D. R. *Fake News Detection Using Majority Voting Technique*. arXiv, 2022. Disponível em: <https://arxiv.org/abs/2203.09936>.

PAUL, M. J.; LUCA, M. In: _____. *Machine Learning For Dummies* New York, NY: For Dummies. p. 430–435. ISBN 9781119245513. Disponível em: <https://www.oreilly.com/library/view/machine-learning-for/9781119245513/>.

PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O.; BLONDEL, M.; PRETTENHOFER, P.; WEISS, R.; DUBOURG, V.; VANDERPLAS, J.; PASSOS, A.; COURNAPEAU, D.; BRUCHER, M.; PERROT, M.; DUCHESNAY, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011.

PETTA, J. R.; JOHNSON, A. C.; TAYLOR, J. M.; LAIRD, E. A.; YACOBY, A.; LUKIN, M. D.; MARCUS, C. M.; HANSON, M. P.; GOSSARD, A. C. Coherent manipulation of coupled electron spins in semiconductor quantum dots. *Science*, v. 309, n. 5744, p. 2180–2184, 2005. Disponível em: <https://www.science.org/doi/abs/10.1126/science.1116955>.

POLIKAR, R. Ensemble based systems in decision making. *IEEE Circuits and Systems Magazine*, v. 6, n. 3, p. 21–45, 2006.

POLIKAR, R. Ensemble learning. In: _____. *Ensemble Machine Learning: Methods and Applications*. Boston, MA: Springer US, 2012. p. 1–34. ISBN 978-1-4419-9326-7. Disponível em: <https://doi.org/10.1007/978-1-4419-9326-7_1>.

PROKHORENKOVA, L.; GUSEV, G.; VOROBEV, A.; DOROGUSH, A. V.; GULIN, A. Catboost: Unbiased boosting with categorical features. In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2018. (NIPS'18), p. 6639–6649.

PU, J.; ZHAO, X.; DONG, P.; WANG, Q.; YUE, Q. Extracting information on rocky desertification from satellite images: A comparative study. *Remote Sensing*, v. 13, n. 13, 2021. ISSN 2072-4292. Disponível em: <https://www.mdpi.com/2072-4292/13/13/2497>.

RAY, S. A quick review of machine learning algorithms. *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*, p. 35–39, 2019.

RINCY, T. N.; GUPTA, R. Ensemble learning techniques and its efficiency in machine learning: A survey. In: *2nd International Conference on Data, Engineering and Applications (IDEA)*. [S.l.: s.n.], 2020. p. 1–6.

SAEED, U.; JAN, S. U.; LEE, Y.-D.; KOO, I. Fault diagnosis based on extremely randomized trees in wireless sensor networks. *Reliability Engineering System Safety*, v. 205, p. 107284, 2021. ISSN 0951-8320. Disponível em: <https://www.sciencedirect.com/science/article/pii/S095183202030781X>.

SAGI, O.; ROKACH, L. Ensemble learning: A survey. *WIREs Data Mining and Knowledge Discovery*, v. 8, n. 4, p. e1249, 2018. Disponível em: <https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/widm.1249>.

SCHAPIRE, R. The strength of weak learnability. In: . [S.l.: s.n.], 1990. v. 5, p. 28–33. ISBN 0-8186-1982-1.

SCHAPIRE, R. E. The strength of weak learnability. *Machine Learning*, v. 5, n. 2, p. 197–227, Jun 1990. ISSN 1573-0565. Disponível em: <https://doi.org/10.1007/BF00116037>.

SCHAPIRE, R. E. *The Design and Analysis of Efficient Learning Algorithms*. Cambridge, MA, USA: MIT Press, 1992. ISBN 0262193256.

SCHAPIRE, R. E.; SINGER, Y. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, v. 37, n. 3, p. 297–336, Dec 1999. ISSN 1573-0565. Disponível em: <https://doi.org/10.1023/A:1007614523901>.

SCHMIDT, J.; MARQUES, M. R. G.; BOTTI, S.; MARQUES, M. A. L. Recent advances and applications of machine learning in solid-state materials science. *npj Computational Materials*, v. 5, p. 1–36, 2019.

SHULMAN, M. D.; DIAL, O. E.; HARVEY, S. P.; BLUHM, H.; UMANSKY, V.; YACOBY, A. Demonstration of entanglement of electrostatically coupled singlet-triplet qubits. *Science*, v. 336, n. 6078, p. 202–205, 2012. Disponível em: <https://www.science.org/doi/abs/10.1126/science.1217692>.

SIVA, K.; KOOLSTRA, G.; STEINMETZ, J.; LIVINGSTON, W. P.; DAS, D.; CHEN, L.; KREIKEBAUM, J. M.; STEVENSON, N.; JüNGER, C.; SANTIAGO, D. I.; SIDDIQI, I.; JORDAN, A. N. *Time-Dependent Hamiltonian Reconstruction using Continuous Weak Measurements*. arXiv, 2022. Disponível em: <https://arxiv.org/abs/2211.07718>.

TAYLOR, J. M.; PETTA, J. R.; JOHNSON, A. C.; YACOBY, A.; MARCUS, C. M.; LUKIN, M. D. Relaxation, dephasing, and quantum control of electron spins in double quantum dots. *Phys. Rev. B*, American PhysicalSociety, v. 76, p. 035315, Jul 2007. Disponível em: <https://link.aps.org/doi/10.1103/PhysRevB.76.035315>.

WITTEN, I. H.; FRANK, E.; HALL, M. A. Chapter 12 - the knowledge flow interface. In: WITTEN, I. H.; FRANK, E.; HALL, M. A. (Ed.). *Data Mining: Practical Machine Learning Tools and Techniques (Third Edition)*. Third edition. Boston: Morgan Kaufmann, 2011, (The Morgan Kaufmann Series in Data Management Systems). p. 495–503. ISBN 978-0-12-374856-0. Disponível em: <https://www.sciencedirect.com/science/article/pii/B9780123748560000122>.

WOLPERT, D. H. Stacked generalization. *Neural Networks*, v. 5, n. 2, p. 241–259, 1992. ISSN 0893-6080. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0893608005800231>.

XIE, Y.; ZHU, C.; HU, R.; ZHU, Z. A coarse-to-fine approach for intelligent logging lithology identification with extremely randomized trees. *Mathematical Geosciences*, v. 53, 07 2020.