UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
CAMPUS DE SÃO JOÃO DA BOA VISTA

JEAN ZANELLA CORREIA

Modelagem computacional de arranjo em série de antenas *patch* retangular utilizando métodos de aprendizado de máquina

São João da Boa Vista
2022

Jean Zanella Correia

Modelagem computacional de arranjo em série de antenas *patch* retangular utilizando métodos de aprendizado de máquina

Trabalho de Graduação apresentado ao Conselho de Curso de Graduação em Engenharia Eletrônica e de Telecomunicações do Campus de São João da Boa Vista, Universidade Estatual Paulista, como parte dos requisitos para obtenção do diploma de Graduação em Engenharia Eletrônica e de Telecomunicações .

Orientador: Profº Dr. Rafael Abrantes Penchel

São João da Boa Vista
2022

C824m

Correia, Jean Zanella
Modelagem computacional de arranjo em série de antenas patch retangular utilizando métodos de aprendizado de máquina / Jean Zanella Correia. -- São João da Boa Vista, 2022
49 f. : il., tabs., fotos

Trabalho de conclusão de curso (Bacharelado - Engenharia de Telecomunicações) - Universidade Estadual Paulista (Unesp), Faculdade de Engenharia, São João da Boa Vista
Orientador: Rafael Abrantes Penchel

1. Matemática. 2. Antenas (Eletrônica). 3. Inteligência artificial. 4. Algoritmos. 5. Aprendizado do computador. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca da Faculdade de Engenharia, São João da Boa Vista. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"
FACULDADE DE ENGENHARIA - CÂMPUS DE SÃO JOÃO DA BOA VISTA
GRADUAÇÃO EM ENGENHARIA ELETRÔNICA E DE TELECOMUNICAÇÕES

TRABALHO DE CONCLUSÃO DE CURSO

MODELAGEM COMPUTACIONAL DE ARRANJO EM SÉRIE DE ANTENA PATCH RETANGULAR UTILIZANDO MÉTODOS DE APRENDIZADO DE MÁQUINA

Aluno: Jean Zanella Correia
Orientador: Prof. Dr. Rafael Abrantes Penchel

Banca Examinadora:

- Rafael Abrantes Penchel (Orientador)
- Ivan Aritz Aldaya Garde (Examinador)
- Renan Alves dos Santos (Examinador)

A ata da defesa com as respectivas assinaturas dos membros encontra-se no prontuário do aluno (Expediente nº 199/2021)

São João da Boa Vista, 18 de agosto de 2022

DADOS CURRICULARES

JEAN ZANELLA CORREIA

NASCIMENTO	29/07/1998
FILIAÇÃO	Tânia Regina Mariano Zanella José Aparecido Correia
2017 / 2022	Graduação em Engenharia Eletrônica e de Telecomunicações UNESP–SJBV

...

RESUMO

Conforme o acesso à tecnologia aumenta exponencialmente nos dias de hoje, a utilização e a pesquisa de dispositivos que operam em ondas milimétricas cresce. Uma opção para o uso, é a antena de *microstrip*. Para aumentar a diretividade e o ganho desse tipo de antena, é necessário utilizar estruturas periódicas, como o arranjo em série, já que possui menores percas na rede de alimentação do que a paralela. Embora essa configuração possibilite o direcionamento de feixe e aumente da diretividade, o processo implica no surgimento de lóbulos laterais, e no alto custo de processamento computacional em simulações devido ser eletricamente grande. Portanto, o propósito do trabalho é modelar computacionalmente uma antena de arranjo em série com 8 elementos, a fim de realizar previsões precisas da diretividade e do nível de lóbulo lateral a partir de variações na largura das plaquetas. A partir das previsões é possível encontrar valores ótimos de parâmetros geométricos, coeficientes de redução da largura do elemento, que reduzam o nível de lóbulo lateral num modelo de menor custo de processamento computacional. A modelagem será feita utilizando métodos de aprendizado de máquina para regressão, já que os valores de diretividade e nível de lóbulo lateral são escalares. Para atingir tais objetivos é necessário calcular o nível de lóbulo lateral de cada configuração, analisar os dados e construir os modelos preditivos. Os modelos de aprendizagem supervisionado construídos em *Python* foram: Regressão Linear Múltipla, Regressão por Árvore de Decisão, Regressão por Árvores de Decisão Aleatórias e Redes Neurais Artificias. O projeto computacional construído atingiu o objetivo principal do trabalho, sendo capaz de realizar previsões assertivas em pouco tempo de processamento.

PALAVRAS-CHAVE: ANTENA, DIRETIVIDADE, NÍVEL DE LÓBULO LATERAL, APRENDIZADO DE MÁQUINA

ABSTRACT

As access to technology increases exponentially these days, the use and research of devices that operate in millimeter waves grows. One option for use is the microstrip antenna. To increase the directivity and gain of this type of antenna, it is necessary to use periodic structures, such as the series arrangement, since less losses in the supply network than in parallel. Although this configuration allows beam steering and increases directivity, the process implies the emergence of side lobes, and the high cost of computational processing in simulations due to being electrically large. Therefore, the purpose of this work is to computationally model a series array antenna with 8 elements, in order to perform accurate measurements of directivity and side lobe level from variations in the amplitude of the plates. From the estimates it is possible to find the values of the geometric parameters, the optimal width reduction elements, which reduced the side lobe level in a computational processing cost model. Modeling will be done using steering capability machine equipment as the side lobe values are scaled. To achieve these goals it is necessary to calculate the side lobe level of each configuration, analyze the data and build the predictive models. The supervised learning models built in Python were: Multiple Linear Regression, Decision Tree Regression, Random Decision Tree Regression and Artificial Neural Networks. The computational design built the main objective of the work, being able to perform assertive processing in a short time of work.

KEYWORDS: ANTENNA, DIRECTIVITY, RATE SIDELOBE LEVEL, MACHINE LEARNING

SUMÁRIO

1 INTRODUÇÃO

Conforme o acesso à tecnologia aumenta exponencialmente nos dias de hoje, a utilização de dispositivos que operam em ondas milimétricas cresce. A tecnologia voltada para operação em ondas milimétricas denominada *mmWave*, é uma parte do conjunto de tecnologias do 5G, por exemplo. Este espectro pode ser usado para comunicações sem fio de alta velocidade, como visto no recente padrão *Wi-Fi 802.11ad* que está operando em 60GHz. Essa faixa de frequência é fundamental para o funcionamento da rede móvel de quinta geração em zonas densas da cidade, onde há necessidade de alta capacidade para curtas distâncias. Portanto, pesquisas voltadas para antenas que operem nessa faixa de frequência estão em alta.

Uma opção para o uso, é a antena de *microstrip*. Além de ser de fácil manipulação, possui construção simples de pequena dimensão, favorecendo o baixo custo. Sua desvantagem principal é possuir baixa diretividade, implicando em ganho mínimo quando comparado a outras antenas convencionais. Uma opção para aumentar a diretividade é a utilização de estruturas periódicas dos elementos, como o arranjo em série. Embora possibilite o direcionamento de feixe e aumente a diretividade, essa configuração geométrica implica no surgimento de lóbulos laterias, que interfere outras frequências vizinhas.

Como essa classe de antenas é eletricamente grande, conforme os elementos do arranjo cresce, o custo de processamento computacional aumenta, levando muito mais tempo para realizar simulações. Uma alternativa de modelagem buscando velocidade e assertividade, é o aprendizado de máquina. O aprendizado de máquina vêm sendo a estrela principal do campo de estudo da inteligência artificial, possibilitando às máquinas auxiliarem na resolução de muitos problemas. Envolver o *machine learning* requer conhecimentos em análise de dados, já que a base de dados deve estar adequada para bom desempenho do algoritmo. A linguagem de programação escolhida é a mais utilizada no ramo, *Python*, *open-source* e orientada a objetos, que facilita o aprendizado por possuir boas práticas de programação visando a interpretação de terceiros. A base de dados que foi exportada do *software* de simulação *Ansys HFSS*, além de conter dados referentes a diretividade, possui quatro parâmetros geométricos construtivos que configuram a largura do *patches*, que interferem no diagrama de radiação e nos níveis de lóbulos secundários.

O propósito do trabalho é modelar computacionalmente uma antena de arranjo em série com 8 elementos, a fim de realizar previsões precisas da diretividade e do nível de lóbulo lateral a partir de variações na largura das plaquetas. A partir das previsões é possível encontrar valores ótimos de parâmetros geométricos, coeficientes de redução da largura do elemento, que reduzam o nível de lóbulo lateral num modelo de menor custo de processamento computacional. A modelagem será feita utilizando métodos de aprendizado de máquina para regressão, já que os valores de diretividade e nível de lóbulo lateral são escalares.

1.1 MOTIVAÇÃO E JUSTIFICATIVA

A demanda por aproveitar os espectros disponíveis na faixa de onda milimétricas, faz ser necessário a utilização de antenas que possibilitam tal operação. Antenas *patch* de arranjo em série possuem vantagens nessas aplicações pelo aumento da diretividade sem redes de alimentações complexas, mantendo baixo custo e pequenas dimensões. O custo da alta diretividade numa aplicação de setorização, conceito de cobertura numa região específica por um grupo de canais a fim de reduzir a interferência, é o surgimento de altos níveis de lóbulos laterais. Os lóbulos laterais atrapalham na operação de uma antena que deve ser diretiva, causando sobreposição de frequência. Além disso, grandes arranjos de antenas dificultam a simulação computacional em questão de tempo, por ficarem eletricamente grandes, causando maior custo de processamento.

A inteligência artificial está num ritmo de crescimento exponencial, principalmente depois da pandemia causada pelo Coronavírus. O aprendizado de máquina vêm sendo a estrela principal do campo de estudo, possibilitando às máquinas auxiliarem na resolução de muitos problemas. Sobretudo, o aprendizado de máquina permite o aumento da produtividade pela velocidade e assertividade, diminuindo o custo de processamento.

Portanto, o aprendizado de máquina se torna uma interessante opção na resolução dos problemas apresentados. A velocidade diminuirá o custo de processamento e a assertividade possibilita novas previsões, a fim de encontrar bons resultados de diretividade com baixo nível de lóbulo lateral. Será uma abordagem prática teórica, modelando o comportamento da diretividade e nível de lóbulo lateral de uma antena arranjada com 8 plaquetas em série.

1.2 OBJETIVOS

Esse trabalho tem como objetivo realizar um estudo de aprendizado de máquina comum e profundo aplicados na previsão da diretividade e no nível de lóbulo lateral de um arranjo em série de antena *patch* retangular. Os objetivos específicos para o cumprimento do propósito são:

- Entendimento do problema e obtenção da base de dados provindas de uma antena simulada pelo *Ansys HFSS*;
- Análise exploratória com perspectiva analítica, a fim de compreender os dados de forma ampla;
- Cálculo do *RSLL* (Taxa de lóbulo lateral [dB]);
- Construção de modelos de aprendizagem supervisionados para aplicação de regressão;
- Construção de um modelo de *deep learning*, no caso redes neurais artificiais.
- Analisar os resultados e obter conclusões objetivas sobre a pesquisa.

1.3 ORGANIZAÇÃO DO TRABALHO

A organização do trabalho é composta por cinco capítulos. No primeiro capítulo é feita uma introdução do tema, motivação e objetivos do trabalho. No capítulo 2 é introduzido a teoria da antena

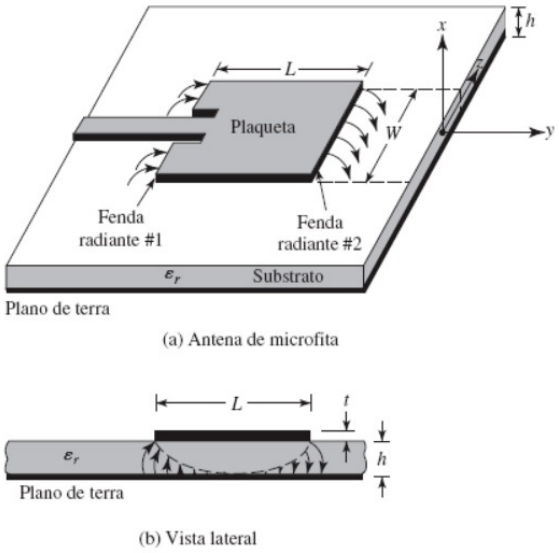
utilizada para extração da base de dados. No terceiro capítulo é feita uma revisão teórica de Ciência de Dados como um todo, citando e comentando os campos envolvidos de estudo. O quarto capítulo é o capítulo de projeto, detalhando o desenvolvimento e explicando os resultados obtidos. A conclusão por final, representa as considerações finais e trabalhos futuros.

2 ANTENA DE *MICROSTRIP* E ARRANJOS EM SÉRIE

As antenas de *microstrip* são de fácil manipulação, simples construção, baixo perfil e baixo custo. Elas são de grande importância para diversas aplicações aeronáuticas e espaciais e estão sendo utilizadas atualmente para aplicações comerciais, tais como radares, televisão, *IoT*, redes de telefonia móvel, etc. As desvantagens críticas desse tipo de antena é possuir fraco desempenho de varredura e pequena largura de banda (BALANIS, 2012).

Essa antena é constituída por dois condutores paralelos, em que o terra é o plano de referência e a antena, também chamada de plaqueta, é o elemento radiador. Eles são separados por substrato dielétrico com altura h e constante dielétrica ϵ_r, como indica a Figura 1.

Figura 1 – Antena de *microstrip*



Fonte: Adaptado (BALANIS, 2012)

A figura Figura 1 ilustra uma antena *patch* retangular com largura W, comprimento L e espessura do condutor t.

Primeiramente, a largura W do *patch* que gera uma boa eficiência de radiação (BALANIS, 2012) é dado como

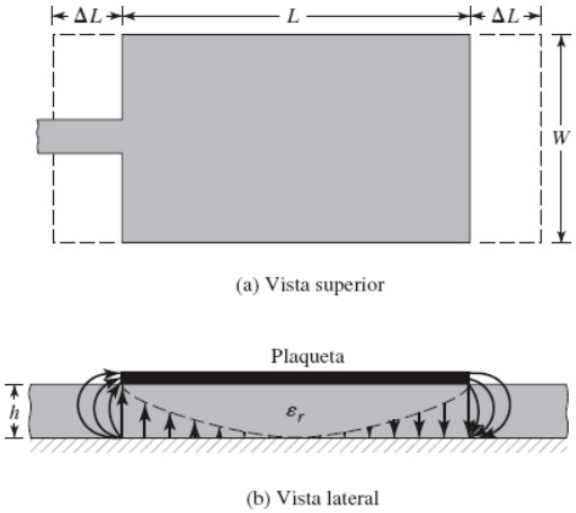
$$W = \frac{v_0}{2f_r}\sqrt{\frac{2}{\varepsilon_r + 1}}, \tag{2.1}$$

onde v_0 é a velocidade da luz no vácuo e f_r é a frequência de ressonância.

Com W já conhecido, é preciso verificar se $W/h > 1$. Sendo cumprida esta condição, a constante dielétrica efetiva pode ser escrita como

$$\varepsilon_{ref} = \frac{\varepsilon_r + 1}{2} + \frac{\varepsilon_r - 1}{2}1 + 12\frac{h}{W}^{-1/2}. \tag{2.2}$$

Figura 2 – Comprimento efetivo da antena



Fonte: Adaptado (BALANIS, 2012)

Como mostrado na figura Figura 2, para determinar o comprimento L_{total}, leva-se os efeitos de borda em consideração (BALANIS, 2012), ocasionando em um maior comprimento elétrico L do *patch*. Por isso, é preciso calcular o comprimento incremental ΔL, como é mostrado na equação (2.3).

$$\Delta L = 0,412h\frac{\varepsilon_{ref} + 0,3\frac{W}{h} + 0,264}{(\varepsilon_{ref} - 0,258)\frac{W}{h} + 0,8}. \tag{2.3}$$

Calculado o comprimento incremental, é possível determinar o comprimento efetivo, dado como

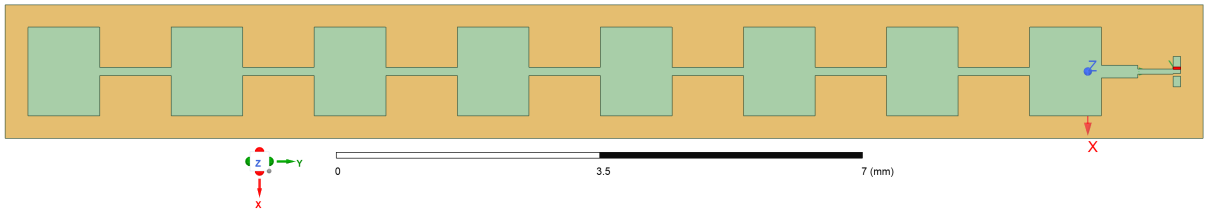
$$L = \frac{1}{2f_r\sqrt{\varepsilon_{ref}}\sqrt{\mu_0\varepsilon_0}} - 2\Delta L, \tag{2.4}$$

onde μ_o é a permeabilidade magnética do vácuo e ε_0 é a constante dielétrica do vácuo.

2.1 ARRANJOS EM SÉRIE DE ANTENAS *PATCH* RETANGULARES

Para utilizar antenas de plaqueta com maior ganho é necessário configurar um arranjo de antenas em série, configuração operacional de estrutura periódica com simplicidade na rede de alimentação e sem percas como ocorre no caso paralelo. Os arranjos em série de antenas são arranjos lineares que podem ser alimentadas através de uma linha de transmissão. A fase dos elementos radiantes são determinadas pelo espaçamento ao longo da linha de transmissão, conforme a frequência é alterada ocorre uma mudança de fase progressiva, mudando a direção do feixe principal. Para que a antena tenha um funcionamento coeso, é preciso que as n plaquetas ressonem em fase para garantir maior energia na transmissão, alcançando maior diretividade. Conforme o número de elementos aumenta, a antena tende a desempenhar melhor em questão de ganho e largura de banda.

Embora possibilite o direcionamento de feixe e aumente a diretividade, essa configuração geométrica implica no surgimento de lóbulos laterais. Os lóbulos secundários são prejudiciais numa aplicação que envolve alto aproveitamento de energia com boa diretividade. Numa aplicação de setorização,

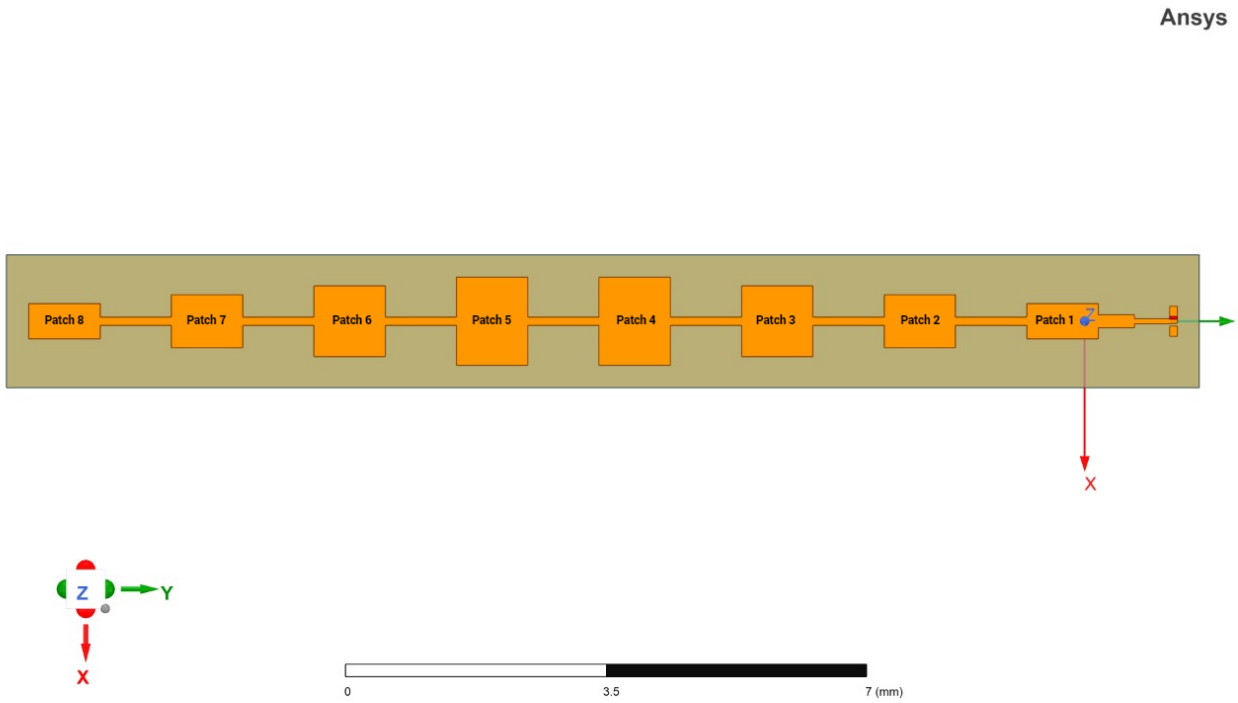


Fonte: Próprio autor

Figura 3 – Antena de arranjo em série com 8 plaquetas retangulares uniformes

varredura de frequência para cobertura de um grupo de canais, ocorre sobreposição de frequências prejudicando o desempenho da aplicação. A definição de nível de lóbulo lateral que é utilizado no trabalho é uma taxa expressa em [dB] e está nomeado como *RSLL* (*Ratio Sidelobe Level*), diferença entre a amplitude de pico do lóbulo principal e a maior amplitude de lóbulo lateral. É importante citar que o *RSLL* é mais sensível aos efeitos de fase, sendo mais irregular que a diretividade.

Uma das possíveis maneiras de realizar um afunilamento (do inglês *taper*) é variar a largura dos elementos retangulares em pares, e de forma empírica, foi observado que essas variações modificam a diretividade e o nível de lóbulo lateral da antena. Esse afunilamento costuma ser variado em conjunto a distância entre os elementos, que nesse caso está cravada em aproximadamente $(\lambda_0/2)$, comprimento de onda referente a 60GHz. Portanto foi configurado quatro coeficientes de redução para as plaquetas, citados como parâmetros geométricos, já que cada coeficiente reduz a largura de dois elementos mudando as características de radiação da antena. Pela Figura 4 e Tabela 1 é possível entender como a mudança dos parâmetros reflete na geometria das plaquetas da antena.



Fonte: Próprio autor

Figura 4 – Antena de arranjo em série com 8 plaquetas retangulares não uniformes

Tabela 1 – Características dos parâmetros geométricos *alphas*

Parâmetros	Plaquetas alteradas	Intervalo de variação
Alpha 1	Patch 1 e 8	0,2 - 0,6
Alpha 2	Patch 2 e 7	0,4 - 0,8
Alpha 3	Patch 3 e 4	0,6 - 1,0
Alpha 4	Patch 5 e 6	0,8 - 1,2

Fonte: Próprio autor

Essa antena projetada pelo *ANSYS HFSS* é eletricamente grande e influencia no custo de processamento computacional, requisitando melhor configuração de *hardware* da máquina para não estender o tempo de processamento. Além disso, a redução de lóbulo lateral nessa configuração se torna difícil devido não possuir uma expressão analítica que modela essas variações. Portanto a modelagem dessa antena será feita utilizando inteligência artificial, com métodos de aprendizado de máquina comum e profundo, visando realizar previsões dos valores de diretividade e *RSLL* pela variação dos parâmetros geométricos.

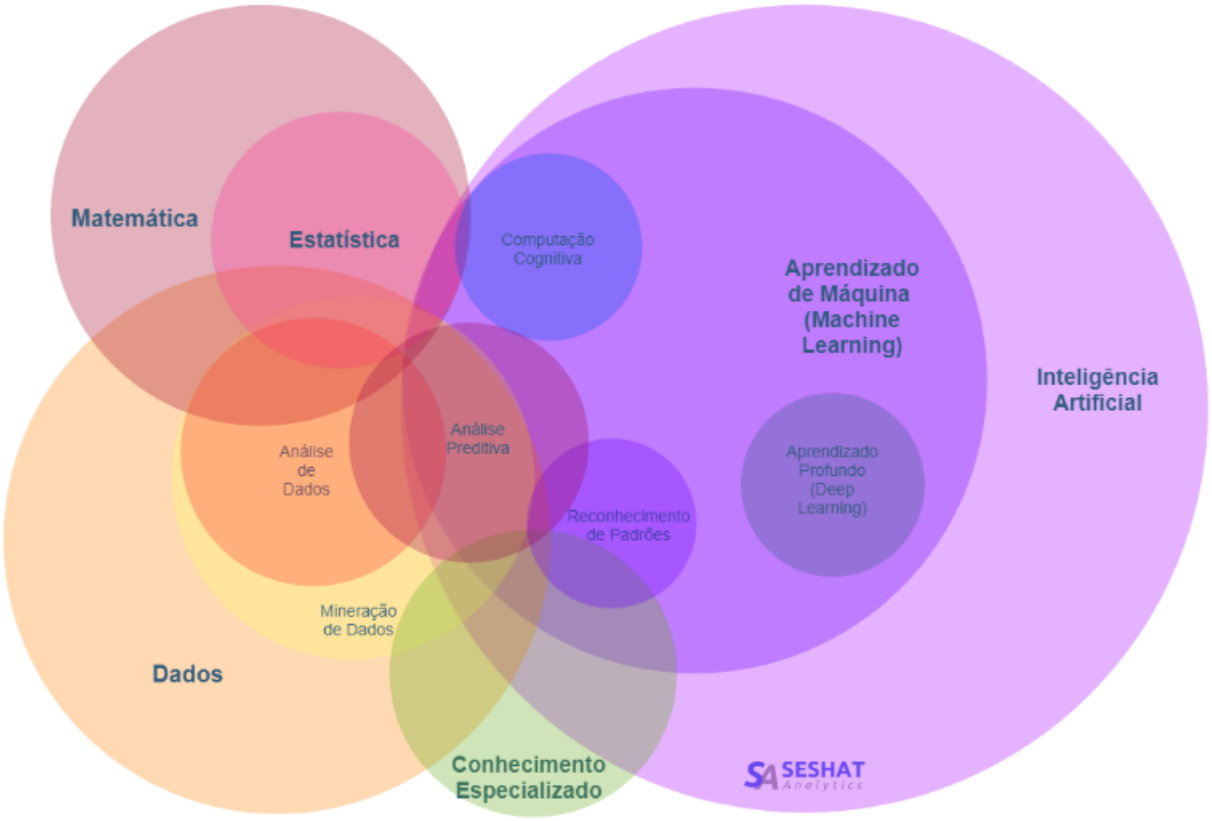
3 APRENDIZADO DE MÁQUINA PARA MODELAGEM

3.1 INTRODUÇÃO

Machine Learning é o termo em inglês para aprendizado de máquina. Como o próprio nome sugere, a máquina "aprende". Para isso, ela utiliza amostras de dados e assimila essas informações através de métodos estatísticos que estão integrados aos algoritmos da máquina, permitindo-os analisar as preferências dos usuários e, a partir disso, fazer previsões.

O aprendizado de máquina possui uma derivação, chamado de aprendizado de máquina profundo. Ambas estão relacionadas a Ciência de Dados e são uma área de Inteligência Artificial. Está ilustrado o diagrama de Ciência de Dados pela Figura 5 para entender melhor onde estão alocados os termos citados.

Figura 5 – Exemplo de uma definição ilustrativa de Ciência de Dados



Fonte: (GALDINO, 2022)

Em termos simples, Inteligência Artificial é um conjunto de sistemas ou máquinas que assemelham-se a inteligência humana para realização de tarefas com base nas informações recebidas, com capacidade de aprimoramento independente. Ela está mais relacionada ao processo de análise de dados e pensamento superpoderoso do que qualquer função em particular. Vale ressaltar que a IA não pretende substituir os seres humanos e sim contribuir para as tomadas de decisões (WHAT... 2022).

Já a ciência de dados, pode ser entendida como uma combinação entre análise de dados de forma geral, matemática e inteligência artificial. Também é um conjunto de recursos como algoritmos, princípios e ferramentas que tem com objetivo descobrir padrões explícitos ou implícitos nos dados brutos. Para trabalhar com análise de dados com gigantesca quantidade de informações, é preciso conhecer as características desse mecanismo de análise:

- Volume: grande quantidade de informação;
- Velocidade: a cada segundo novos elementos são agregados ao banco de dados;
- Variedade: relação com a natureza diversa das informações.

A ciência de dados é um dos campos mais interessantes na atualidade, pois tem aplicação em diversas áreas e grande valor para as organizações, já que o crescente armazenamento de informações se bem interpretados, tornam as decisões mais assertivas e revela tendências do mercado. Essas vantagens refletem em pesquisas como do presente trabalho, que tem como aplicação principal realizar previsões com baixo custo de processamento computacional conforme o algoritmo aprende os padrões da base de dados.

3.2 CONCEITOS BÁSICOS DE ANÁLISE DE DADOS

A análise de dados é um processo com aplicação de técnicas, como estatísticas e lógicas. Essas técnicas ajudam a avaliar as informações que podem ser úteis ou não para o processo. A partir destas informações, é possível tomar decisões mais assertivas e orientadas para resultados (MCKINNEY, 2018).

A análise de dados é de extrema importância e é dividida em quatro grupos:

1. **Análise descritiva**

   Esta é a análise mais simples de se executar, pois ela consiste em descrever dados já obtidos (O QUE... 2022). Basicamente, resume-se a relatórios e apresentações de informações pontuais, como analisar o desempenho em física de alunos de uma escola através da frequência em sala de aula, e a partir disso inferir que aqueles com menor porcentagem de faltas obtiveram as melhores notas, logo uma possível solução para quem ficou abaixo da média é estar mais presente em aula.

2. **Análise exploratória**

   É uma análise que contém a parte descritiva, porém, com auxílio de técnicas de regressão e estudo da variância, faz um estudo sobre a correlação das variáveis do problema (O QUE... 2022).

3. **Análise preditiva**

   A análise preditiva utiliza padrões que ocorrem na base de dados para criar um modelo que, caso certas condições sejam cumpridas, consegue prever possíveis cenários futuros (O QUE... 2022). Como exemplo, se uma pessoa tem uma rotina bem definida, é possível prever que o horário de almoço dela nas sextas-feiras é às 12 horas.

4. **Análise prescritiva**

   Como em uma consulta médica, em que após avaliação dos sintomas do paciente junto com avaliação física e de exames e o médico prescreve um remédio, esta análise usa como base justamente uma análise pré-realizada, como a preditiva. Esta análise por ser dependente de previsões e predições, é necessário fazer o uso de algoritmos, aprendizado de máquina e inteligência artificial, devido a alta mutabilidade dos dados. Portante, a prescrição ocorre após uma previsão e tem o intuito de direcionar recursos para obter melhores resultados (O QUE... 2022).

3.3 CONCEITOS DE APRENDIZADO DE MÁQUINA

3.3.1 Funcionamento do *Machine Learning*

Para que um algoritmo de *machine learning* consiga aprender, é necessário que ele contenha estas três qualidades:

- **Decisão**

  Como os algoritmos de *machine learning* são responsáveis por fazerem a classificação ou predição de dados, a partir de uma base de dados, será feita uma análise e os dados serão listados como rotulados ou não rotulados, e assim será encontrada estimativas sobre um padrão dos dados (IBM, 2020).

- **Função de erro**

  A função de erro tem como objetivo verificar as predições feitas pelo modelo de ML. Para fazer essa validação, é necessário ter um exemplo prático já conhecido. Se a medida de erro entre o modelo e o exemplo estiver dentro de um valor aceitável, então este será utilizado para fazer as estimativas do padrão dos dados (IBM, 2020).

- **Otimização** Usando a função de erro, é possível verificar se existem parâmetros que podem ser mudados no modelo para que ele se aproxime ainda mais dos valores do exemplo conhecido (IBM, 2020). Para isso, é necessário ajustar o algoritmo para fazer uma varredura de maneira constante e automática, fazendo com que as variáveis do problema sejam ajustadas de uma forma em que os valores da aproximação se aproximem do exemplo conhecido.

3.3.2 Tipos de *Machine Learning*

3.3.2.1 Aprendizado de máquina supervisionado

No *machine learning* supervisionado, a partir de um conjunto de dados que já tem rótulos e saída definidas. o algoritmo é treinado para classificar dados ou prever resultados com uma precisão mais apurada. Este treinamento é feito conforme os dados entram, em que os pesos das variáveis são ajustados para obter uma melhor aproximação. Estes ajustes são obtidos usando validação cruzada, para que não haja sub-ajuste ou super-ajuste (IBM, 2020). Alguns conceitos usados para ML supervisionado são redes neurais, regressão linear e logística, classificação de múltiplas classes etc.

3.3.2.2 Aprendizado de máquina não supervisionado

Diferente do ML supervisionado, o não supervisionado trabalha com algoritmos de ML que vão analisar e agrupar dados não rotulados (IBM, 2020). Por conta disso, ele é um processo mais independente, já que o algoritmo aprende a identificar padrões complexos e ocultos sem que haja necessidade de intervenção humana.

3.3.2.3 Aprendizado de máquina semi-supervisionado

Este é um meio termo entre os anteriores, em que o aprendizado do algoritmo consiste em usar um conjunto menor de dados já rotulados para orientar e classificar como os recursos de um conjunto maior de dados não rotulado será extraído (IBM, 2020).

3.3.2.4 Aprendizado de máquina de reforço

A solução deste problema é feito por tentativa e erro e é frequentemente aplicado em aprendizado de máquina semi-supervisionado. Conforme o algoritmo avança em suas tentativas, os testes completamente aleatórios começam a apresentar possíveis modelos de solução até conseguir desenvolver táticas e métodos sofisticados para resolver o problema proposto (IBM, 2020).

3.3.3 Tipos de soluções de problemas de aprendizado de máquina

- Classificação: problema de ML supervisionado ou semi-supervisionado que se baseia em categorias. As amostras de dados passadas ao treinamento da máquina possuem rótulos indicativos de categorias, com o objetivo de categorizar os dados de teste/validação.
- Regressão: problema de ML supervisionado ou semi-supervisionado que se baseia em valor numérico único escalar. Com mesma analogia de treinamento e teste, o objetivo é a predição desse valor. Exemplo: realizar a predição do peso corporal de uma pessoa com base nas amostras de n pessoas semelhantes.
- Clusterização: problema de ML não supervisionado que consiste em reconhecimento de características numa amostra de dados sem rótulos que categorizam-nas.

3.3.4 Etapa de treinamento e teste/validação computacional

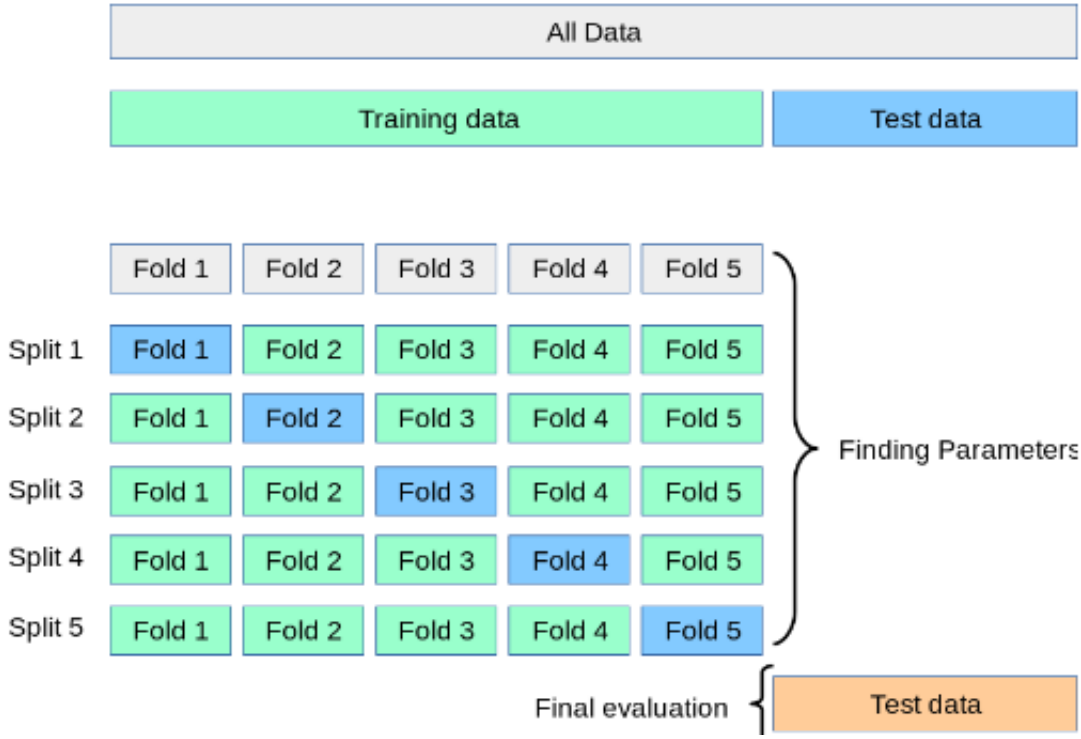
Essa etapa consiste em subdividir os dados que serão utilizados pelo algoritmo, em subconjuntos de treino e teste. Dividir de maneira simples os dados, por exemplo, os 75% primeiros dados são de treinamento e os últimos 25% de teste, gera problemas dependendo das características dos dados. Se nesse exemplo os dados de treinamento não tiverem todas as características do conjunto completo dos dados, na hora de realizar a avaliação, o modelo pode não se comportar muito bem, já que o subconjunto de teste pode conter amostras nunca vistas pela etapa de treinamento. Uma alternativa para realizar o treinamento e teste de forma balanceada, é a validação cruzada.

3.3.4.1 Validação Cruzada

Validação cruzada é uma técnica para avaliar os modelos de aprendizado de máquina. Essa avaliação é por meio do treinamento de vários modelos em subconjuntos de dados de entrada disponíveis e avaliação deles no subconjunto complementar dos dados. Use a validação cruzada para detectar sobreajuste, ou seja, a não generalização de um padrão (PEDREGOSA et al., 2011). Essa definição pode ser melhor compreendida pela Figura 6.

Portanto a validação cruzada é eficiente contra esse problema, além dos problemas de sobreajuste. A solução implícita da validação cruzada é separar uma parte dos dados para treino e teste criando várias combinações diferentes, afim de fazer todos os dados serem de treino e teste e um subconjunto que pode ser previamente separado pode ser utilizado para a validação final.

Figura 6 – Validação Cruzada



Fonte: (PEDREGOSA et al., 2011)

3.3.5 Métricas utilizadas em problemas de regressão

3.3.5.1 Coeficiente de determinação R-quadrado

Quando é feita uma regressão, cria-se uma reta que tende a se aproximar dos pontos reais da amostra. Neste contexto, o coeficiente R-quadrado é calculado com o intuito de mostrar o quão boa (ou ruim) é a regressão realizada. Para medir isto, ele leva em consideração a distância ao quadrado entre a linha de regressão e os pontos reais da amostra, chamada de soma dos quadrados do resíduos (SQ_{res}) e a compara com a diferença entre uma aproximação feita utilizando a média dos valores da amostra com os valores reais dos dados, chamada de soma dos quadrados total (SQ_T) (CHEIN, 2019).

Com isso, é possível escrever uma expressão para R^2, dada em função de SQ_{res} e SQ_T, como mostra a equação (3.1).

$$R^2 = \frac{SQ_T - SQ_{res}}{SQ_T}. \tag{3.1}$$

A partir da equação (3.1), é possível aferir algumas coisas. Se o ajuste feito através da regressão englobar todos os pontos reais de forma perfeita, então SQ_{res} é nulo, e consequentemente R^2 é um. A interpretação deste valor diz que o modelo de regressão foi capaz de explicar toda a variabilidade em relação a média. Porém, quando o valor de R^2 é nulo, isso significa que a regressão feita é igual a aproximação feita pela média, que representa o pior cenário possível.

Portanto, quanto maior o valor de R^2, melhor é a relação entre o modelo de regressão, pois ele consegue explicar uma boa parte da variabilidade relação relação a média. Se R^2 tender a zero, então o modelo não consegue explicar praticamente nada da variabilidade em relação a média.

3.3.5.2 *Mean squared error* - Erro médio quadrático

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y_i}\right)^2 \tag{3.2}$$

3.3.5.3 *Root mean squared error* - Raiz do erro médio quadrático

$$RMSE = \sqrt{\sum_{i=1}^{n}\frac{\left(\hat{Y_i} - Y_i\right)^2}{n}} \tag{3.3}$$

3.3.5.4 Entropia

A entropia é uma relação de quão pura ou impura é a informação analisada. Quanto maior o valor da entropia, mais heterogêneo é o conjunto de dados. Com isto, é possível chegar a uma expressão para a entropia considerando um conjunto de entrada (S) que pode ter c classes distintas (SILVA, 2005), dada como

$$\text{Entropia}\,(S) = -\sum_{i=1}^{c} p_i \log_2 pi. \tag{3.4}$$

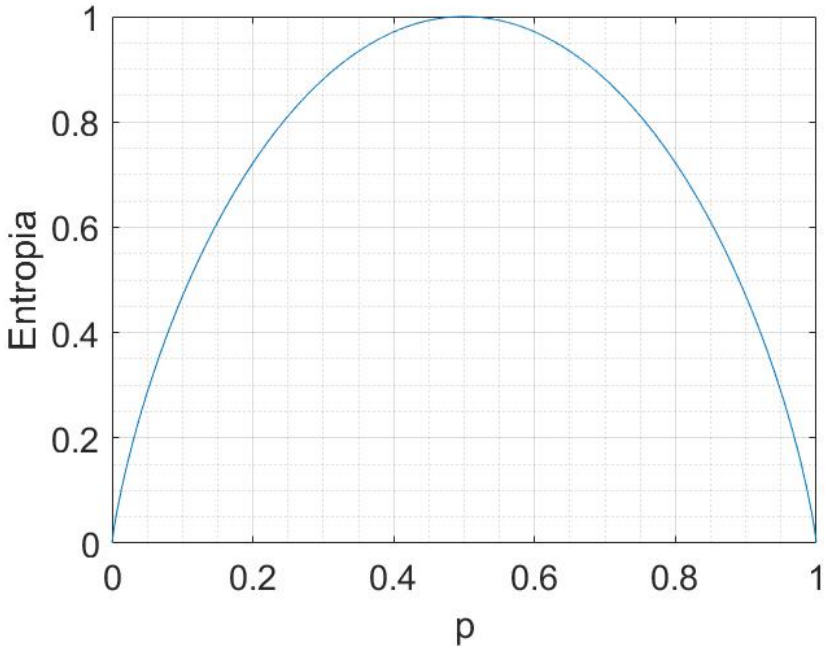
Algumas observações e considerações

- Será considerado $\log_2 0 = 0$;
- $\log_2 1 = 0$

A partir da equação (3.4), quando existe a probabilidade de 0,5 de um certo evento ocorrer, então existe a probabilidade de 0,5 para ele não acontecer. Isto coloca a ferramenta de decisão sobre um grau de incerteza grande, pois para estes valores a entropia assumirá o valor 1, que é seu máximo. Para que a decisão seja tomada com o menor erro possível, é necessário que os dados de entrada sejam classificados de forma que apresentem uma maior homogeneidade, isto significa que quanto maior a probabilidade de um evento acontecer é em relação ao de não acontecer, menor serão os erros de

decisão pois a entropia será menor que 1. Quanto mais próximo o valor da entropia é de 0, mais acurada é a decisão tomada. Isto pode ser visto de maneira mais clara através da figuraFigura 7, que ilustra o gráfico da entropia gerado pela equação (3.4).

Figura 7 – Representação gráfica da entropia



Fonte: Próprio autor.

Pela figura Figura 7, é possível verificar o que foi dito no parágrafo anterior, em que a entropia vai ser máxima quando a probabilidade de um evento acontecer é $p = 0.5$, a probabilidade de ele não acontecer é $1 - p = 0.5$. Se o valor $p = 0$, então $1 - p = 1$, logo, pela equação (3.4), Entropia $(S) = 0$. O mesmo acontece se $p = 1$. Portanto, quanto mais homogênea é a variável, melhor é a certeza da decisão.

Outra métrica que é utilizada para é o ganho de informação. Ele mede a diminuição da entropia esperada quando é utilizado um certo atributo AT para particionar um conjunto de dados S (SILVA, 2005).

Dado um conjunto AT, então os valores que este conjunto pode assumir serão dados como P(AT). Com isso, dado que x é um elemento do conjunto P(AT), um subconjunto S_x é criado pelos dados AT = x. A partir disso, a entropia em função da partição de S em função de AT é dada como (SILVA, 2005)

$$\mathrm{E}\,(AT) = \sum_{x \epsilon P(AT)} \rho \mathrm{Entropia}(S_x), \tag{3.5}$$

onde $\rho = \frac{|\text{Nº de amostras de } S_x|}{|\text{Nº de amostras de } S|}$.

Portanto, o ganho de informação é dado por

$$\mathrm{GI} = \mathrm{Entropia}(S) - \mathrm{E}(AT). \tag{3.6}$$

Portanto, a árvore de decisão tem o objetivo de diminuir a entropia (torna mais homogênea a variável que está sendo analisada), ter poucos nós e ser consistente com o conjunto de dados (SILVA, 2005).

3.3.5.5 Índice Gini

O índice Gini mede a heterogeneidade dos dados. Ele é dado como

$$\text{IG} = 1 - \sum_{i=1}^{c} p_i^2, \tag{3.7}$$

onde p_i é frequência relativa de cada classe em cada nó e c é o número de classes (SILVA, 2005).

Semelhante a entropia, quando seu valor é nulo, o nó é puro. E, quanto mais próximo do valor um, mais impuro o nó se torna.

A diferença de quando usar o índice Gini para a entropia é que ele tende a a isolar os registros que representam a classe que aparece mais frequentemente em um ramo (SILVA, 2005). Já a entropia faz um balaço do número de registros feitos em cada ramo da árvore.

3.3.6 *Overfitting* e *Underfitting*

O Sobre-ajuste do inglês *Overfitting*, termo comumente utilizado no Brasil pelos profissionais do ramo, se refere a um modelo que se ajusta muito bem aos dados, porém possui pouca generalização para novos dados. O *Overfitting* é mais provável em modelos não paramétricos e não lineares que têm mais flexibilidade ao aprender uma função de destino.

A falta de ajuste do inglês *Underfitting*, em contraste, se refere a um modelo que não consegue se ajustar aos dados e muito menos generalizar para novos. A falta de ajuste geralmente não é tão discutido quanto o sobre-ajuste, pois é fácil de detectar dada uma boa métrica de desempenho.

O objetivo é selecionar um modelo num ponto de modelagem ideal entre os dois fenômenos. Para chegar nesse objetivo é interessante utilizar um modelo de aprendizagem que se adapta bem aos dados, mas possui uma complexidade adequada para o problema. Se essa complexidade for alta, os dados de testes não serão bem interpretados caso tenham pequenas diferenças dos dados de treinamento, ou seja, o algoritmo decorou o conjunto de dados de treino e não sabe se adequar pois não é genérico.

3.3.7 Hiperparâmetros

Até o momento foi visto que a partir de alguns parâmetros, um modelo de *machine learning* consegue aprender e ajustar o valor destes parâmetros para obter modelos ainda melhores. Porém, os hiperparâmetos não podem ser definidos durante o processo de aprendizagem. Seus valores serão escolhidos manualmente, isso é feito antes do algoritmo estar em execução. Depois disso, serão avaliados diversos modelos com diferentes valores de hiperparâmetros, até que um dos valores escolhidos gere um bom modelo.

3.4 MÉTODOS DE *MACHINE LEARNING* PARA REGRESSÃO

3.4.1 Regressão linear simples

Este é o modelo mais básico de regressão linear, em que apenas uma variável explicativa X_i gera a variável dependente Y_i(FÁVERO, 2015). A equação que descreve este modelo de regressão é dado como

$$Y_i = a + b_1 X_{1i} + u_i, \tag{3.8}$$

onde a é o intercepto, b_1 é o coeficiente linear associado à variável explicativa X_i e u_i é o termo do erro, em que é comparada a distância entre os pontos reais e os pontos gerados pelo modelo de regressão linear.

Para encontrar o intercepto, o coeficiente linear e o termo de erro é preciso fazer o uso do método dos mínimos quadrados, no qual a aproximação será feita considerando um conjunto de dados reais (FÁVERO, 2015). Sendo este conjunto composto pela variável independente X_i e variável dependente Y_i, é possível encontrar o coeficiente angular da seguinte maneira:

$$b_1 = \frac{\sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y})}{\sum_{i=1}^{n}(X_i - \overline{X})^2}, \tag{3.9}$$

onde $\overline{X}$ e $\overline{Y}$ são o valores médios da variável X e Y, respectivamente.

O intercepto é dado como

$$b_0 = \overline{Y} - b_1 \overline{X}, \tag{3.10}$$

e o termo de erro é

$$u_i = \overline{Y_i} - \overline{Y}. \tag{3.11}$$

3.4.2 Regressão linear múltipla

A regressão linear múltipla oferece a possibilidade do estudo da relação de duas ou mais variáveis explicativas (X_i), que se apresentam na forma linear, e uma variável dependente (Yi) (FÁVERO, 2015). Portanto, a equação que define uma regressão linear múltipla é dada como

$$Y_i = a + b_1 X_{1i} + b_2 X_{2i} + ... + b_n X_{ni} + u_i. \tag{3.12}$$

Sabendo a expressão da regressão, agora basta definir os coeficientes da equação. Para um problema de duas variáveis explicativas (FÁVERO, 2015), tem-se

$$Y_i = a + b_1 X_{1i} + b_2 X_{2i} + u_i. \tag{3.13}$$

Para determinar os coeficientes b_1 e b_2, é necessário utilizar encontrar a solução do sistemas de equações (FÁVERO, 2015) mostrado na equação (3.15).

$$\sum_{i=1}^{n} Y_i X_{1i} - \frac{\sum_{i=1}^{n} Y_i \sum_{i=1}^{n} X_{1i}}{n} =$$

$$b_1 \left[\sum_{i=1}^{n} (X_{1i})^2 - \frac{(\sum_{i=1}^{n} X_{1i})^2}{n}\right] + b_2 \left[\sum_{i=1}^{n} X_{1i}X_{2i} - \frac{(\sum_{i=1}^{n} X_{1i})(\sum_{i=1}^{n} X_{2i})}{n}\right]; \quad (3.14)$$

$$\sum_{i=1}^{n} Y_i X_{2i} - \frac{\sum_{i=1}^{n} Y_i \sum_{i=1}^{n} X_{2i}}{n} =$$

$$b_1 \left[\sum_{i=1}^{n} X_{1i}X_{2i} - \frac{(\sum_{i=1}^{n} X_{1i})(\sum_{i=1}^{n} X_{2i})}{n}\right] + b_2 \left[\sum_{i=1}^{n} (X_{2i})^2 - \frac{(\sum_{i=1}^{n} X_{2i})^2}{n}\right]. \quad (3.15)$$

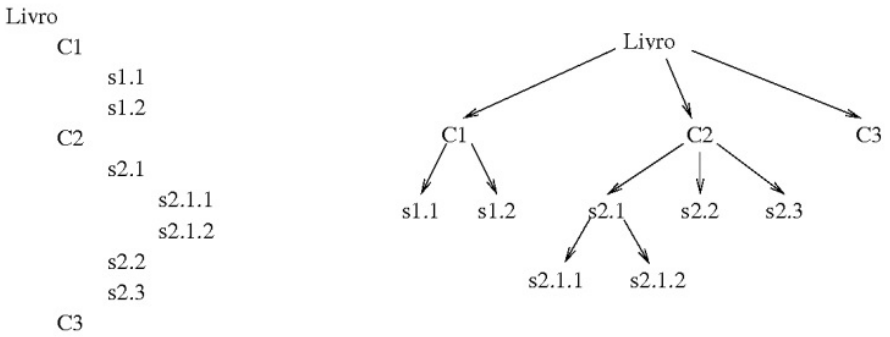
E, para determinar o intercepto, tem-se

$$a = \overline{Y} - b_1\overline{X_1} - b_2\overline{X_2}. \quad (3.16)$$

3.4.3 Árvores de decisão

Uma árvore nada mais é uma estrutura que possuí uma raiz, que é o ponto em que a árvore começa a ser construídas e, a partir dela, ter vários ramos que se conectam a nós e que consequentemente terminam nas folhas. Em uma árvore de decisão, são avaliadas todas as possibilidades de um evento que parte de um nó inicial (raiz) que se ramifica em outros nós, que acabam por formar ramos, até chegar nas folhas, que são os nós finais. Como exemplo, suponha que existe um livro que está dividido em capítulos C1, C2 e C3, e que estes estão subdivididos como mostra a figura Figura 8 (LAURETTO, 2010). Com estes dados, considerando que o livro é o nó inicial, da raiz saem três ramos que chegam nos nós C1, C2 e C3, que são os capítulos do livro. De C1 saem dois ramos que se encontram com os nós s1.1 e s1.2, suas seções. Já de C2 saem três ramos que se encontram com os nós s2.1, s2.2 e s2.3, porém s2.1 se ramifica em mais dois ramos, que chegam aos nós s2.1.1 e s2.1.2, que são subseções. Por fim, como é possível observar na figura Figura 8, o nó C3 não apresenta ramificações, logo, esta árvore está completa.

Figura 8 – Árvore de decisão construída a partir do sumário de um livro



Fonte: (LAURETTO, 2010)

Foi visto um exemplo de como estruturar uma árvore genérica, mas o intuito desta abordagem é escolher variáveis que fazem o evento que está sendo estudado apresente o menor grau de incerteza possível, fazendo com que o processo seja mais acurado. Para isso, existem algumas métricas que podem ser calculadas que vão mostrar a relação de quanta informação se tem a partir de uma determinada variável. Neste trabalho, serão abordadas os conceitos de entropia e ganho de informação.

3.4.3.1 Podagem

3.4.3.1.1 Pré-Podagem

Conforme a árvore está sendo construída, nó a nó, é verificado o valor do ganho de informação, e se o valor obtido for menor que um limiar preestabelecido, então o nó vira uma folha, ou seja, não sairão mais ramos dele.

3.4.3.1.2 Pós-Podagem

Esta podagem é realizada quando a árvore já está completa, e tem como objetivo encontrar a sub-árvore com melhor desempenho (SILVA, 2005).

Para executar a podagem, é preciso seguir os seguintes passos:

1. Varrer toda a estrutura da árvore;
2. Calcular o erro considerando se o nó virar uma folha (todos os ramos e nós abaixo dele seriam eliminados) e o erro para caso não haja poda;
3. Se a diferença entre os erros encontrados for maior do que um valor preestabelecido, então o nó vira uma folha; caso contrário, o nó se mantém e a verificação é feita novamente para os nós abaixo dele.

Este processo se repete até que sejam obtidas diversas árvores podadas. Por fim, é usado um modelo de teste já feito para efeitos de comparação com as árvores podadas geradas, e aquela que apresentar o menor erro de classificação de dados com relação aos dados do modelo de teste é escolhida (SILVA, 2005).

3.5 MÉTODO DE *DEEP LEARNING* PARA REGRESSÃO

3.5.1 Redes Neurais Artificiais

3.5.1.1 Introdução

Uma rede neural é uma predição modelada pela forma de como o cérebro funciona, é constituída por um conjunto de neurônios interligados, formando um potente sistema de armazenamento de conhecimento gerido por exemplos propositais apresentados, usado para inferir sobre novas situações desconhecidas. Esse modelo tem inspiração biológica pela semelhança de funcionamento, a grosso modo, entre o cérebro e o modelo computacional (FURTADO, 2019).

3.5.1.2 Inspiração Biológica

O sistema nervoso é um dos responsáveis por coordenar as atividades dos diversos tecidos e órgãos do corpo humano, inclusive, o tecido nervoso tem em sua constituição os neurônios. A neuróglia, estrutura de sustentação do tecido nervoso, compõe-se de células que atuam na formação de circuitos neurais, essas células envolvem completamente os neurônios, possibilitando a formação de

independentes circuitos neurais que impedem a propagação desordenada dos impulsos. Os neurônios apresentam a propriedade de responder prontamente a estímulos, resposta que se propaga a outros neurônios, glândulas e músculos (FURTADO, 2019).

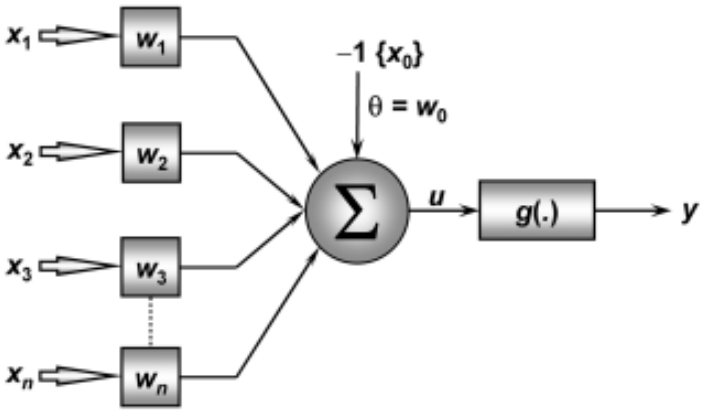
O neurônio é formado por um corpo celular que contém o núcleo, dendritos que são os numerosos prolongamentos especializados em receber estímulos e o axônio, prolongamento único especializado em transmitir informações a outras células, sejam nervosas, musculares ou glandulares. Basicamente, as informações que são recebidas pelos dendritos e núcleo são repassadas pelos axônios, transmissão conhecida como sinapse (FURTADO, 2019).

"Estima-se que esta rede neural biológica, com características bem excêntricas, seja constituída por cerca de 100 bilhões de neurônios"(SILVA, 2010, p.30).

3.5.1.3 *Perceptron*

As redes neurais artificias são baseadas em sistemas nervosos biológicos que envolvem as unidades processadoras, neurônios artificiais e modelos não-lineares simplificados de neurônios humanos. A representação ilustrativa da Figura 9, mostra o modelo mais simples de um neurônio que engloba características como paralelismo e alta conectividade. O conjunto de entrada $x_1, x_2, x_3, ..., x_n$ recebem os advindos sinais que são análogos aos impulsos elétricos captados pelos dendritos (SILVA, 2010).

Figura 9 – Perceptron



Fonte: (SILVA, 2010)

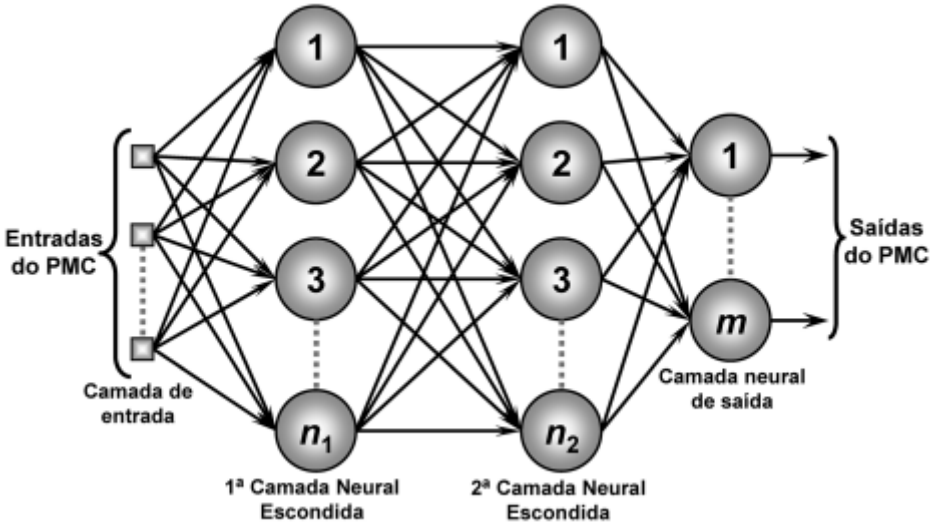
Esse modelo é conhecido como *Perceptron* proposto por *McCulloch* e *Pits* e idealizado por *Rosenblatt*. Sua simplicidade e única saída se deve ao fato de possuir somente um neurônio, mas teve o potencial de atrair diversos pesquisadores do ramo para evoluções significativas (SILVA, 2010).

As entradas, as quais representam informações, são inicialmente ponderadas pelos pesos W_j a fim de quantificar a importância de cada uma frente os objetivos, mapeando o comportamento da entrada e saída do processo. Em seguida o resultado dessa combinação linear é somado ao limiar de ativação Θ, também conhecido como *bias* b, e repassado como argumento para a função de ativação, que produz um retorno na saída y (SILVA, 2010).

A principal contribuição da função de ativação é o poder de decisão de ativação para um neurônio. Ela é a transformação não linear realizada ao longo do sinal de entrada. Esta saída transformada é então enviada para a próxima camada de neurônios como entrada.

3.5.1.4 Redes *Perceptron* multicamadas

Figura 10 – Rede *Perceptron* multicamadas (MLP)



Fonte: (SILVA, 2010)

A caracterização de uma rede multicamadas é conter pelo menos uma camada intermediária de neurônios, denominada por muitos como escondida, situada entre a camada de entrada e saída. Essa rede pertence à arquitetura *feedforward* de camadas múltiplas, modelo de treinamento/aprendizado também supervisionado. Essa arquitetura basicamente define que o sentido que as informações viajam é da camada de entrada para a camada de saída, e nunca retrocede. Isso se explica melhor observando a Figura 10, que o fluxo de informações percorre a camada escondida provindas da camada de entrada com destino à camada de saída (SILVA, 2010).

"Observa-se ainda que na rede MLP convencional inexiste qualquer tipo de realimentação de valores produzidos pela camada neural de saídas ou pelas próprias camadas neurais intermediárias" (SILVA, 2010, p.92).

O início da grande popularidade da MLP foram nos anos 1990, já que as redes passaram e permitir sua implementação no processo de treinamento, graças ao algoritmo de aprendizagem denominado *backpropagation*.

Esse algoritmo trabalha com o fluxo *backward*, ou *feedback*, e usa o erro ocorrido na camada de saída para recalcular os valores dos pesos presentes na camada de entrada, mas isso ocorre de trás para frente, ou seja, os pesos são atualizados desde a camada de saída até chegar na camada de entrada, com objetivo de diminuir o erro no próximo processo.

3.5.1.4.1 Treinamento de uma MLP

É entendido que uma rede neural contém uma camada de saída e uma função de ativação de saída, que por motivos matemáticos, colabora com a otimização dos parâmetros ponderados. Para ocorrer uma otimização, é necessário ter uma medida numérica de qualidade, já que as tentativas dos recálculos de pesos não são por tentativas e erros. Um dos nomes dessa medida qualitativa, é função de custo, que reduz todos os aspectos bons e ruins do sistema à um único valor escalar, permitindo uma comparação entre soluções. Portanto, a escolha da função de custo está diretamente ligada ao objetivo do problema, assim como a camada de saída e sua função de ativação. O problema do presente trabalho, como já citado, trata-se da regressão. A principal métrica de regressão para medir a distância dos resultados inferidos e dos reais, é a distância de *Manhattan*, denominada no ramo como função de custo L1, utilizada no desenvolvimento prático do trabalho (TREINANDO... 2022).

A otimização do processo tem como objetivo diminuir essa função de erro, para isso, os pesos são alterados iterativamente para verificar o impacto de cada mudança. Para avaliar a melhor mudança, é feita a derivada dessa função, ou para múltiplas dimensões, o gradiente. A derivada é como um "medidor" qualitativo dessas iterações, já que indica a direção e quantização de cada modificação nos ponderadores. Em outras palavras, a função de custo determinar o quão longe o inferido está do real, e a derivada determina se essa alteração está no sentido e quanto mais próximo está da minimização. Então, a otimização é o uso dessa informação para escolher a próxima alteração de pesos com objetivo de aproximar os resultados.

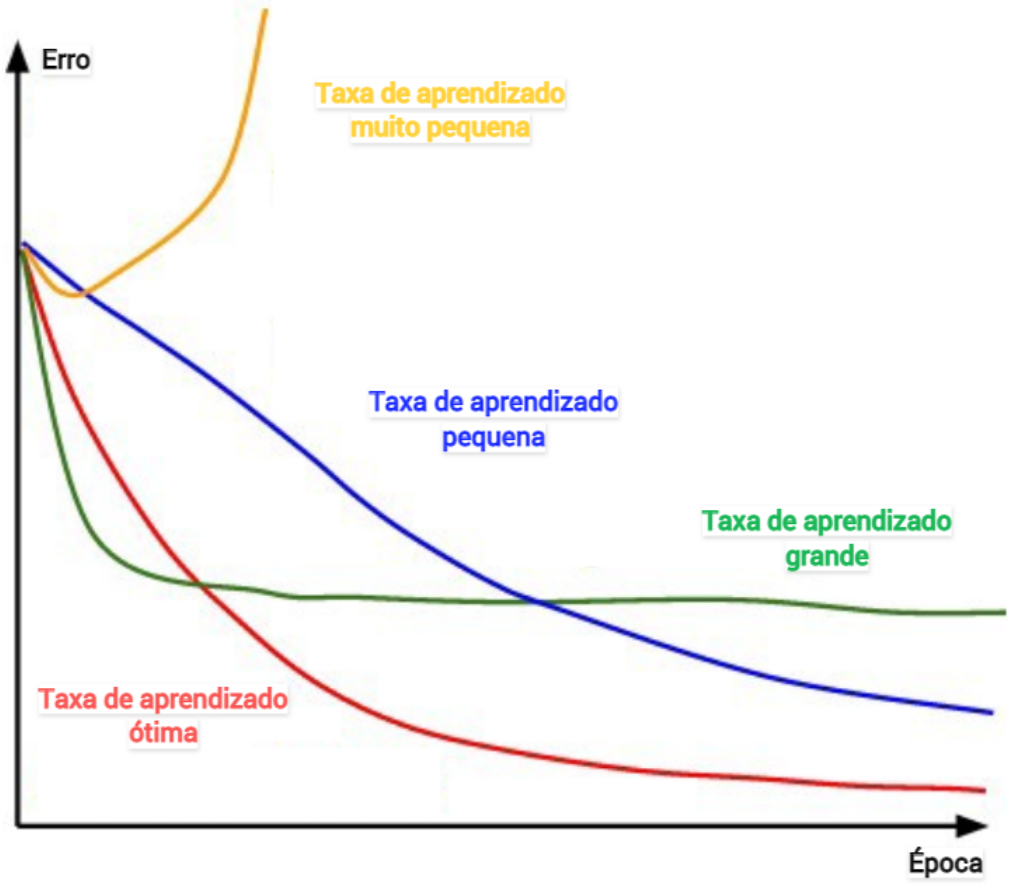
O otimizador primário e clássico é chamado de Descida do Gradiente, dado pela equação (3.17), ela consiste em subtrair o valor do gradiente (∇f) dos pesos (W) da rede. O vetor de derivadas tem um multiplicador *alpha* que pondera a subtração dos pesos controlando o passo de otimização, denominado taxa de aprendizado (TREINANDO... 2022).

$$W_i = W_i - \alpha * \nabla f_i \tag{3.17}$$

A taxa de aprendizado é um dos fatores mais importantes da assertividade de uma rede neural, como ele é um termo independente, sua alteração é feita pelo operador do programa. O efeito de aumentar ou diminuir a taxa de aprendizado é simples de entender. Se a taxa de aprendizado for muito baixa, o modelo vai precisar de muitas iterações para obter um resultado ótimo, levando tempo e custando desempenho computacional. De forma contrária, se a taxa de aprendizado for muito alta, o modelo pulará etapas com passos grandes, não encontrando o erro mínimo facilmente, precisando também de mais iterações que o necessário, caso essa configuração convirja. A Figura 11 ajuda a entender as consequências provindas das alterações do valor de *learning rate*, no eixo y o valor escalar da função de custo e no eixo x as iterações mencionadas acima. Portanto, esse hiperparâmetro deve ser ajustado com assertividade, obtida por testes ou otimizações de hiperparâmetros.

Outros otimizadores foram surgindo como evoluções do Gradiente Descendente, seguindo uma evolução temporal, e cada otimizador requer um ajuste específico de hiperparâmetros. O otimizador utilizado no projeto foi o *Adam*, muito utilizado na atualidade para resolver a maioria dos problemas. O *Adam* possui alguns hiperparâmetros que podem ser modificados, porém muitos estudos revelam

Figura 11 – Comportamento da variação escalar da taxa de aprendizado



Fonte: Adaptado de (*ZULKIFLI*, 2022)

que os principais a serem explorados e modificados são a taxa de aprendizado e o decaimento de pesos (*Weight Decay*). O último se refere a regularização do modelo de modo a evitar o sobreajuste, quanto maior seu valor, mais simples seu modelo se torna. Se o modelo necessitar de mais complexidade para ser assertivo, esse valor deverá ser menor. Ele é uma penalidade L2, que difere do L1 elevando o módulo das diferenças individuais ao quadrado, sendo mais sensível a *outliers* e mais eficiente contra superespecializações. De forma geral, seu papel é generalizar melhor o modelo penalizando grandes saltos dos valores dos pesos, diminuindo o salto escalar que provoca complexidade ao modelo e consequentemente sobreajuste.

Para trabalhar com problemas do mundo real é necessário trabalhar com algum *framework* especializado em *deep learning* para facilitar a implementação de uma rede neural. O *framework* utilizado no presente trabalho é o *PyTorch*, muito utilizado no ramo e com arquitetura já pensada para pessoas que têm um pouco de conhecimento em *machine learning*. Esse e outros *frameworks* utilizam como estrutura de dados o tensor.

O tensor é apenas uma nomenclatura para um *array* de dimensão maior que 2, ou seja, *array* $3D, 4D, ..., nD$.

4 PROJETO DE APRENDIZADO DE MÁQUINA

4.1 CONHECENDO A BASE DE DADOS

O projeto de aprendizado de máquina utiliza um *dataset* exportado do *software* de simulação de antenas Ansys HFSS. A exportação foi provinda dos resultados de uma simulação da antena ilustrada na Figura 4, essa simulação tem como característica a influência na diretividade da antena pela variação da largura dos *patches*, como citado no Capítulo 2. O *dataset* foi exportado com formato *CSV*, 2.567.513 linhas, 8 colunas e tamanho de 250 MB, mediano quando comparado a outros *datasets* que são utilizados no ramo de ciência de dados. O *dataset* está disposto na Figura 12.

Figura 12 – Base de Dados Exportada do *HFSS*

	alpha1 []	alpha2 []	alpha3 []	alpha4 []	Freq [GHz]	Phi [deg]	Theta [deg]	dB(DirTotal) []
0	0.200202	0.627273	0.778384	0.940808	60	90	-180.0	-1.551429
1	0.200202	0.627273	0.778384	0.940808	60	90	-179.9	-1.575115
2	0.200202	0.627273	0.778384	0.940808	60	90	-179.8	-1.599650
3	0.200202	0.627273	0.778384	0.940808	60	90	-179.7	-1.625035
4	0.200202	0.627273	0.778384	0.940808	60	90	-179.6	-1.651271
...	...	...	...	...	...	...	...	...
2567508	0.599798	0.636970	0.806263	0.953737	60	90	179.6	-1.617326
2567509	0.599798	0.636970	0.806263	0.953737	60	90	179.7	-1.613945
2567510	0.599798	0.636970	0.806263	0.953737	60	90	179.8	-1.611334
2567511	0.599798	0.636970	0.806263	0.953737	60	90	179.9	-1.609493
2567512	0.599798	0.636970	0.806263	0.953737	60	90	180.0	-1.608427

2567513 rows × 8 columns

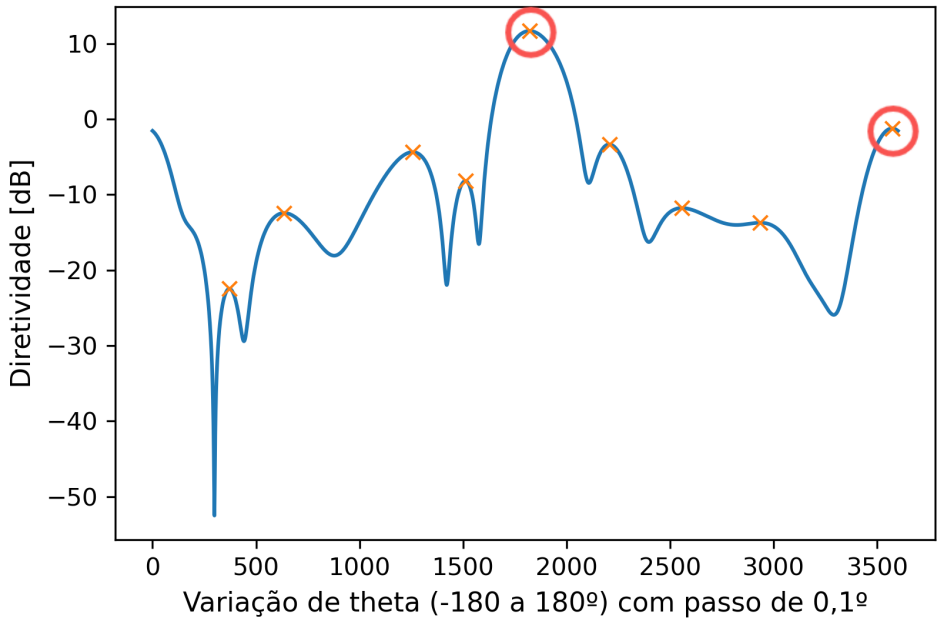
Fonte: Próprio autor

O arquivo foi carregado pelo *Jupyter's Notebook*, ambiente de desenvolvimento *web* voltado para aplicações de Inteligência Artificial, devido sua interface flexível e moderna. Todo o desenvolvimento de análise de dados e aprendizado de máquina foi realizado por ele.

Observa-se que os dados estão orientados por linhas, e duas colunas do *dataset* são desnecessárias: Freq [GHz] Phi [deg]. É notório que os dados de *alpha 1 - alpha 4* se repetem para cada passo de *Theta* (0,1°), até que seu valor chegue ao máximo: 180°. Então para cada combinação de valores dos parâmetros, o *Theta* varia de -180 a 180°, e para cada passo do ângulo é dado uma diretividade. O objetivo é trabalhar com os valores máximos da diretividade para cada arranjo da antena, ou seja, para cada combinação dos parâmetros geométricos existe um valor máximo da diretividade que se encontra num valor de *Theta*.

Para cumprir o objetivo, o primeiro passo é criar outra base de dados com todas as combinações possíveis dos parâmetros *alphas*, utilizando a função *groupby*() da biblioteca Pandas. Esse segundo *dataset* possui 713 linhas e 4 colunas, já que a divisão de 2.567.513 pela quantidade de valores de ângulos *Theta* (3600) é igual a 713. Essa segunda base de dados é importante para ser usada como referência de procura para a função *loc*() também do Pandas. Essa função, a cada iteração sobre os parâmetros, seleciona um subgrupo contendo as 3600 linhas referentes daquele arranjo. Após isso, basta encontrar o índice da linha referente à diretividade máxima. Nessa mesma subcoleção é passada a função *find-peaks*() da biblioteca *Scipy* seção *Signal*, com finalidade de encontrar todos os picos de diretividade, para posteriormente subtrair a maior diretividade da segunda maior, obtendo o *RSLL*. Uma ilustração das 713 possíveis desse processo está disposta na Figura 13. Contendo todos os índices e todos os *RSLL*, é necessário somente selecionar as linhas de interesse num novo *dataset*, com colunas renomeadas e sem as colunas desnecessárias citadas acima.

Figura 13 – Gráfico do diagrama de radiação 2D



Fonte: Próprio autor

Por último foi plotado a descrição dos dados dado pela função *describe()* do Pandas (Figura 15), que traz algumas informações importantes para o conhecimento dos dados. Uma observação importante para os dados serem filtrados pela diretividade máxima, é que os valores de *Theta* estão entre - 0,3 e 8,3°, uma faixa de valores inclusas na largura de feixe da antena.

O próximo passo é iniciar a visualização com diferentes estratégias de gráficos para o começo da interpretação, etapa denominada análise de dados.

Figura 14 – *Dataset* Extraoficial

	Alpha 1	Alpha 2	Alpha 3	Alpha 4	Theta [deg]	Dir [dB]	RSLL [dB]
0	0.200202	0.627273	0.778384	0.940808	2.1	11.681918	12.908683
1	0.200606	0.672121	0.784444	1.027677	1.1	11.713436	13.039437
2	0.201010	0.603838	0.796162	1.080606	0.7	11.650144	13.409771
3	0.201414	0.632929	0.908081	1.005455	0.8	11.725322	13.387417
4	0.202222	0.508485	0.731515	0.969495	2.6	11.435685	12.995963
...	...	...	...	...	...	...	...
708	0.597778	0.649495	0.723838	1.078182	2.5	12.302141	13.282003
709	0.598182	0.599798	0.869697	0.986869	2.6	12.515220	14.293194
710	0.598990	0.545657	0.708485	0.997778	4.3	12.251252	12.623841
711	0.599394	0.612323	0.787677	1.101616	2.2	12.384422	14.287221
712	0.599798	0.636970	0.806263	0.953737	3.0	12.516022	14.124159

713 rows × 7 columns

Fonte: Próprio autor

Figura 15 – Resumo de Estatísticas dos Dados

	Alpha 1	Alpha 2	Alpha 3	Alpha 4	Theta [deg]	Dir [dB]	RSLL [dB]
count	713.000000	713.000000	713.000000	713.000000	713.000000	713.000000	713.000000
mean	0.395701	0.602765	0.800645	1.002551	2.807854	11.958846	13.247797
std	0.115530	0.114935	0.115451	0.115041	1.725355	0.296849	0.954797
min	0.200202	0.400202	0.600606	0.800202	-0.300000	11.257175	10.541277
25%	0.297980	0.504444	0.703232	0.902828	1.600000	11.745287	12.926212
50%	0.394949	0.604242	0.798182	1.001010	2.600000	11.962068	13.455851
75%	0.496364	0.700808	0.898384	1.101212	3.900000	12.188981	13.870875
max	0.599798	0.799798	0.999798	1.199394	8.300000	12.603641	17.191139

Fonte: Próprio autor

4.2 ANÁLISE DE DADOS

Nessa segunda parte do desenvolvimento do projeto, é importante conhecer alguns princípios importantes e comumente utilizados pelos analistas e cientistas de dados como citado no capítulo anterior. O primeiro plote é o mais básico da biblioteca *Matplotlib* seção *Pyplot*: Gráfico de linhas e marcadores *matplotlib.pyplot.plot*().

As distribuições dos valores dos parâmetros que estão na Figura 16 implicam que *Alpha 1* está ordenado de forma crescente. Foi verificado a ocorrência do fato e tal comportamento é explicado

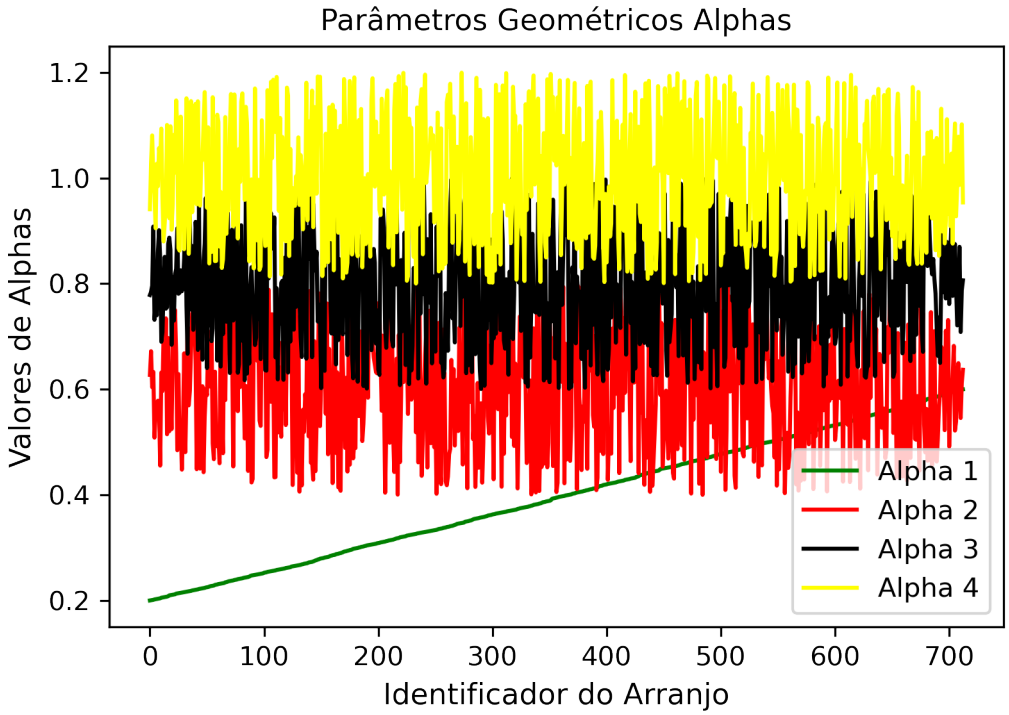


Figura 16 – Fonte: Próprio autor

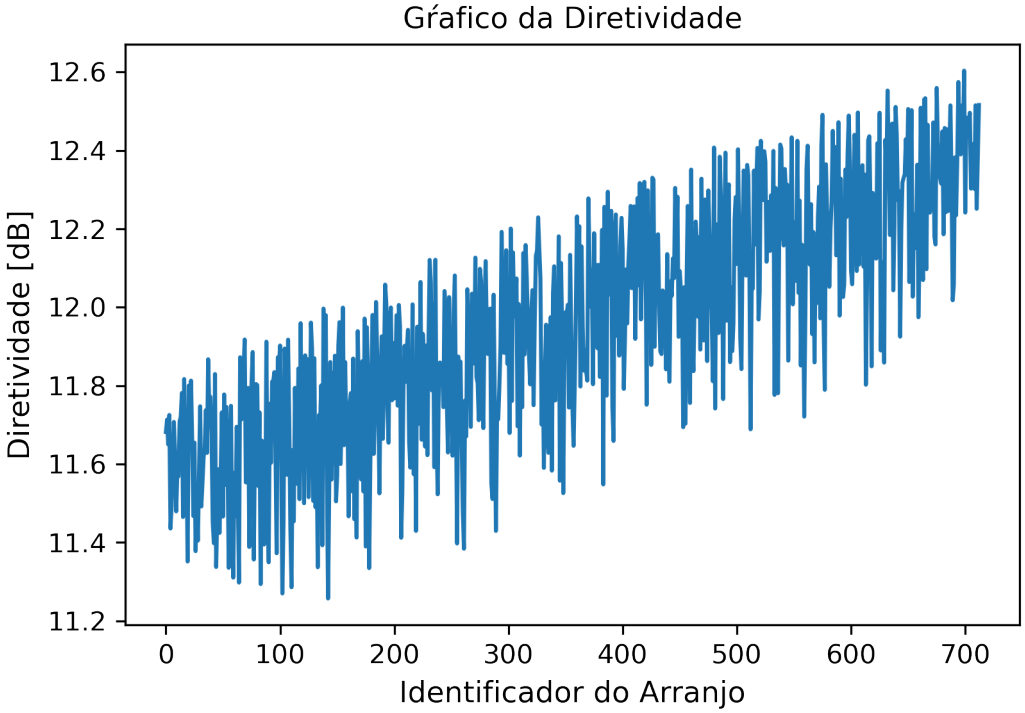


Figura 17 – Fonte: Próprio autor

pelo modo de exportação do software *Ansys HFSS*, que ordena os dados pelo primeiro parâmetro da coleção.

O plote da diretividade está disposto pela Figura 17, nele é interpretável mas não comprovável que o *Alpha 1* influencia o comportamento da diretividade também ser crescente, semelhante ao parâmetro.

Da mesma forma, é viável verificar o comportamento dos valores de *RSLL*. A Figura 18 proporciona o entendimento de que a taxa de lóbulo lateral não tem um comportamento semelhante á diretividade. Portanto, é interessante buscar outros métodos para avaliar tais semelhanças e/ou diferenças.

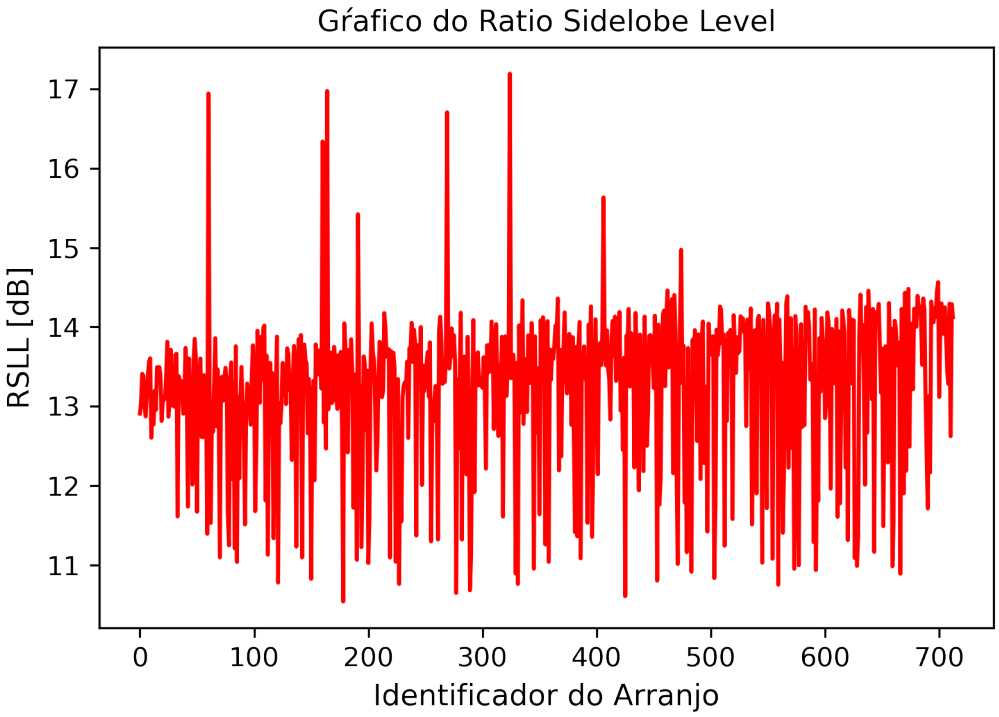


Figura 18 – Fonte: Próprio autor

Outro recurso que pode ajudar é a matriz de correlação (Figura 19), com ela é possível ver como as variáveis dependentes estão relacionadas com as variáveis independentes, já que o objetivo é fazer a predição dos valores da diretividade e *RSLL*.

Figura 19 – Mapa de correlações

	Alpha 1	Alpha 2	Alpha 3	Alpha 4	Theta	Dir [dB]	RSLL
Alpha 1	1.00	-0.02	0.01	-0.01	0.22	0.79	0.13
Alpha 2	-0.02	1.00	0.04	-0.01	-0.37	0.44	0.00
Alpha 3	0.01	0.04	1.00	0.02	-0.54	0.29	0.74
Alpha 4	-0.01	-0.01	0.02	1.00	-0.69	-0.22	-0.03
Theta	0.22	-0.37	-0.54	-0.69	1.00	0.00	-0.33
Dir [dB]	0.79	0.44	0.29	-0.22	0.00	1.00	0.38
RSLL	0.13	0.00	0.74	-0.03	-0.33	0.38	1.00

Fonte: Próprio autor

A matriz de correlação mostra a relação linear entre as colunas da base da dados. Analisando

as relações entre os *targets* e as *features*, pode-se concluir que a diretividade tem alta relação linear com os parâmetros construtivos, principalmente *Alpha 1* e *Alpha 2*. Já o *RSLL* mostra-se ter relação linear apenas com *Alpha 3*. Nesse contexto, é interessante visualizar essas relações lineares de modo ilustrativo, utilizando *Seaborn.pairplot*, plote de distribuições bivariadas de pares. Esse tipo de plote dispersivo é muito utilizado na atualidade antes de qualquer inclusão de aprendizado de máquina, ainda mais se for um problema de regressão, como o do presente trabalho.

Na Figura 20 se confirma as relações lineares comentadas acima sobre a diretividade. O *RSLL* no mapa de correlações mostra-se ter uma relação linear com o parâmetro *Alpha 3*, porém pela imagem do gráfico de dispersão, essa relação parece se aproximar de uma curva. Na parte de resultados do projeto, será entendível tais afirmações.

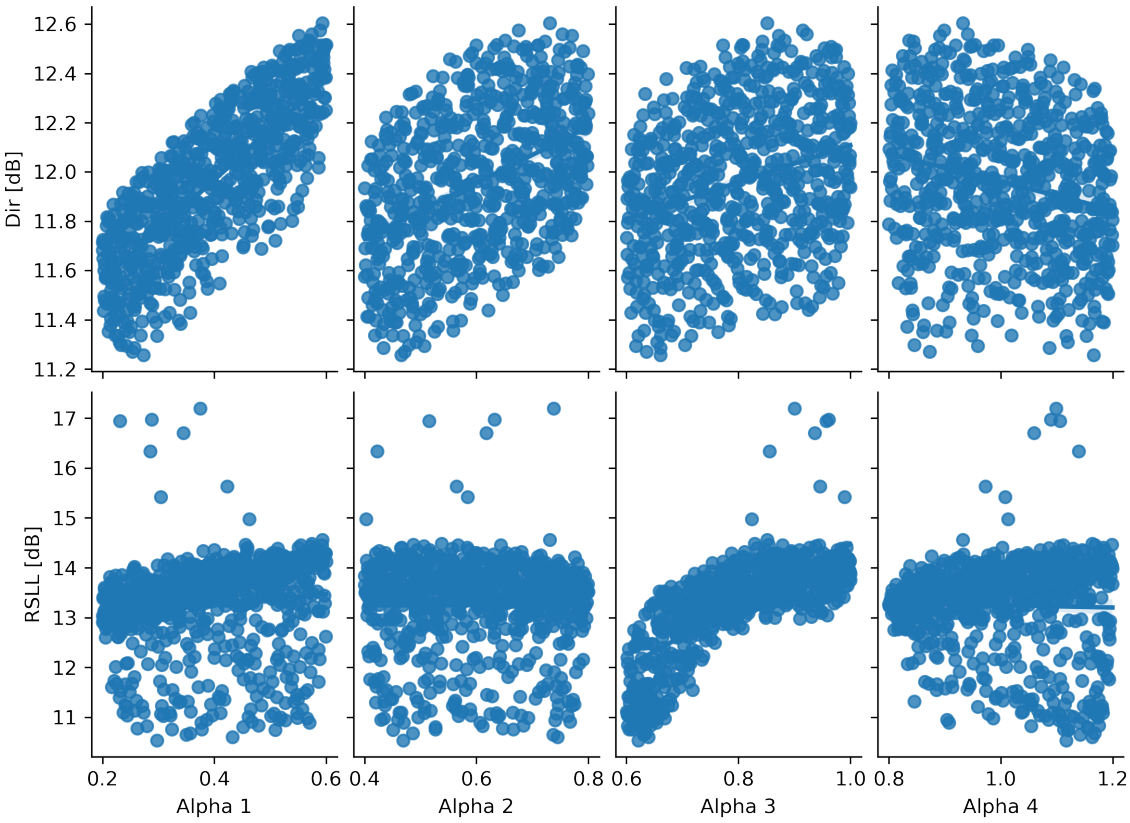


Figura 20 – Fonte: Próprio autor

Antes de iniciar o aprendizado de máquina, os dados para aplicações de regressão não temporais, costumam ser embaralhados. Esse embaralhamento é extremamente importante para a máquina não aprender que o comportamento crescente entre *Alpha 1* e a *Diretividade*, por exemplo, deve ser considerado no treinamento. Imagine que novos dados de validação não estejam distribuídos dessa maneira, as previsões do modelo supervisionado terá baixa eficácia. Uma forma de embaralhar os dados é usar a função *Sample* da biblioteca Pandas, passando como argumento um número inteiro para reprodutibilidade, dessa maneira toda vez que o código for processado, a mesma aleatoriedade será imposta, um processo não genuinamente aleatório e sim pseudo-aleatório. Após essa modificação referenciadas pelas linhas da coleção de dados, os dados ficaram ilustrado pela composta Figura 21.

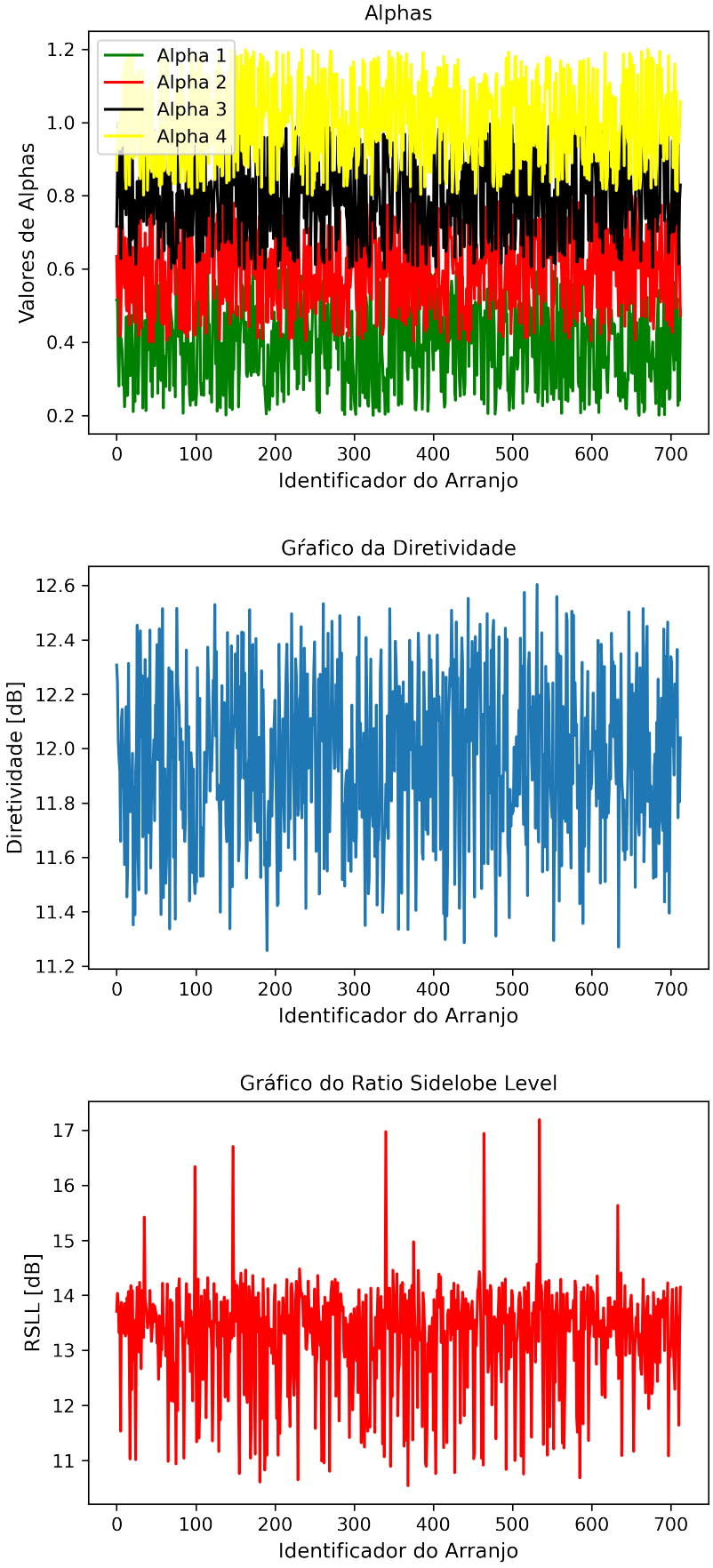


Figura 21 – Figura composta do resultado de embaralhamento dos dados. Fonte: Próprio autor

4.3 REGRESSÃO LINEAR

A função *LinearRegression* da biblioteca *Scikit-Learn* seção *Linear_Model*, tem suporte para atuar com regressões lineares simples e múltiplas. Como existem quatro parâmetros que são as variáveis independentes do problema, a regressão linear múltipla será automaticamente utilizada pela função. A primeira parte de todo projeto de treino, teste e talvez validação, requer a separação dos dados, isso é feito pela função *Train_Test_Split* da mesma biblioteca, porém da seção *Model_Selection*. Para todos os modelos que serão apresentados, com exceção para o uso de validação cruzada, os dados de teste serão 20% do total do *dataset*. O treinamento do algoritmo é para ajustar os coeficientes da função para minimizar o erro. Esse erro pode ser medido utilizando a métrica *R2*, que quanto mais próximo de 1 melhor o treinamento foi concluído.

4.3.1 Diretividade

A primeira parte dessa modelagem de regressão, é para diretividade, *target* que apresentam maior relação linear com os parâmetros. A métrica *R2* de treinamento foi de 0,95. Além da avaliação no treinamento, foram observados essa e outras métricas na etapa de teste, métricas extras como EQM e REQM.

- EQM: 0,004;
- REQM: 0,063;
- *R2*: 0,95.

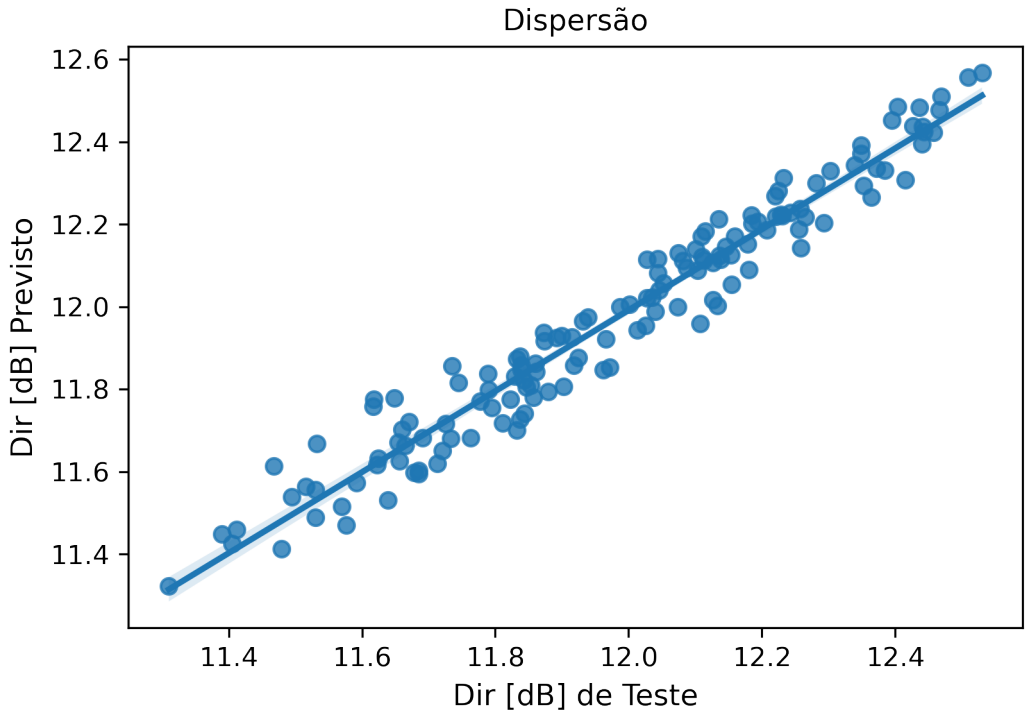


Figura 22 – Fonte: Próprio autor

Para esses resultados serem analisados, é necessário entender o que cada métrica representa para a avaliação do problema explicados no Capítulo 2. O *R2* se manteve em 0,95 na etapa de testes, bom resultado, porém será necessário reavaliar essa métrica. O erro quadrático médio e a raiz do erro quadrático médio estão como o esperado, próximo do valor nulo. Antes de confirmar o bom desempenho do programa, está ilustrado o gráfico de dispersão entre os valores reais e os preditos, que também ajuda a entender as métricas.

A Figura 22 mostra que o modelo responde bem, já que os valores preditos (marcadores redondos) estão a uma pequena distância euclidiana da reta (valores reais). A validação cruzada avalia o desempenho do modelo usando todo o conjunto de dados de treinamento para o treinamento e a avaliação, em vez de uma parte dele. Ela não mede somente a precisão de um modelo. Ela também dá uma ideia de quão representativo o conjunto de dados é e de quão sensível o modelo pode ser a variações nos dados. Portanto o seu uso é vantajoso para entender se o modelo está com problemas quanto para obter novos e talvez melhores resultados. A validação cruzada foi aplicada com algumas quantidades de *folds* porém sem variações relevantes, o modelo continuou desempenho de 0,95, já que a validação cruzada usa *R2* como métrica.

4.3.2 *RSLL*

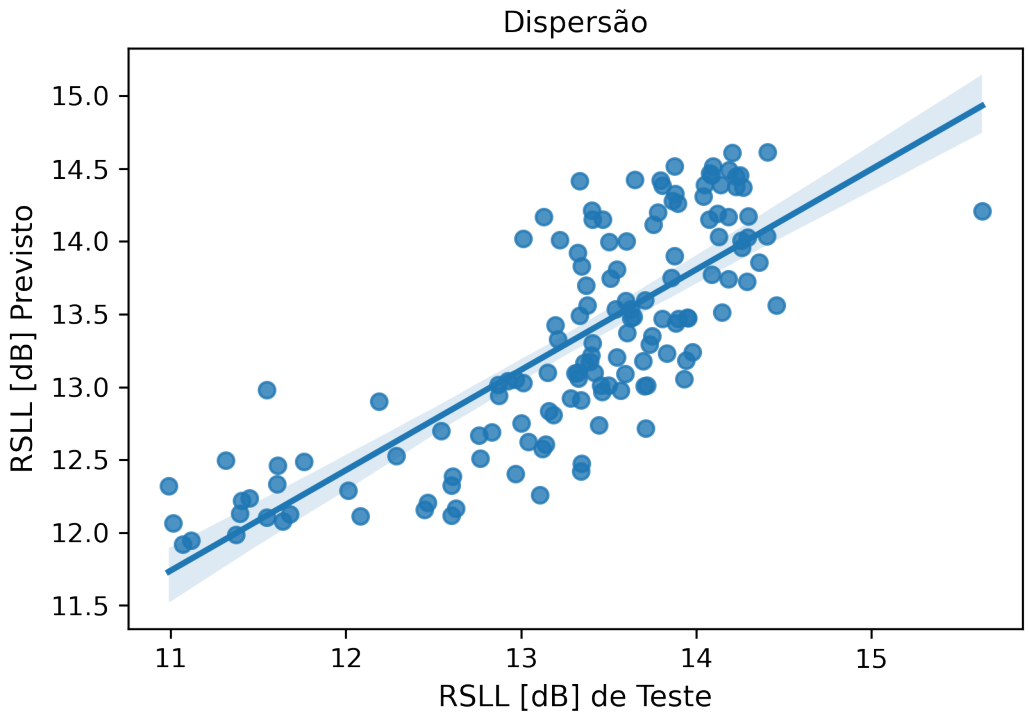


Figura 23 – Fonte: Próprio autor

O *RSLL* não apresentou boas relações lineares com as *features*. Após avaliar o modelo da mesma forma que o anterior, as métricas de testes foram as seguintes:

- EQM: 0,29;
- REQM: 0,54;

- *R2*: 0,62.

O erro quadrático médio e a raiz do erro quadrático médio da maior peso aos maiores erros, já que eles são elevados ao quadrado individualmente. Então é perceptível que o modelo apresenta alguns valores distantes do que pode ser modelo pelo algoritmo, que pode ser confirmado pelos valores que estão mais distantes do restante no gráfico disposto na Figura 18. Esses valores serão verificados para serem denominados *outliers* ou não. A imagem da dispersão dos valores preditos e reais da taxa de lóbulo lateral está disposta na Figura 23 e comprovam certas afirmações.

4.4 REGRESSÃO POR ÁRVORE DE DECISÃO

A árvore de decisão é um talvez o método mais famoso do aprendizado de máquina por ser versátil e conseguir resolver problemas complexos não-lineares. O código foi implementado utilizando o *DecisionTreeRegressor*(), validação cruzada e *GridSearch*(), um método de busca exaustivo de hiperparâmetros. Os hiperparâmetros que foram selecionados para a otimização foram: *max_depth*, *min_samples_leaf* e *min_samples_split*, profundidade máxima, amostras mínimas por nós externos (decisões) e amostras mínimas por nós internos (divisões). A profundidade máxima é um dos hiperparâmetros mais importantes para o desempenho do modelo. Se esse valor for muito grande, a árvore sera muito profunda e passará por mais nós antes de concluir o processo, podendo se super especializar nos dados de treinamento. O número mínimo de amostras por nós interno avalia o número de amostrar por nó, se o número de amostras for menor que o mínimo, esse nó virará uma folha (nó final, sem ramos). Já o número mínimo de amostras por nós externos garante um número mínimo de amostras em uma folha.

Para o caso da diretividade, alvo que possui alta relação linear com as variáveis independentes, o melhor modelo obtido possui as seguintes informações:

- Max_Depth = 11;
- Min_Samples_Leaf = 2;
- Min_Samples_Split = 4;
- R2[%] de Treino = 99;
- R2[%] de Teste = 90.

Já para o *RSLL*:

- Max_Depth = 5;
- Min_Samples_Leaf = 16;
- Min_Samples_Split = 2;
- R2[%] de Treino = 82;
- R2[%] de Teste = 75.

A principal conclusão que pode ser tirada para uma árvore de decisão, é que a profundidade máxima para o segundo caso é o dobro do primeiro. Isso se explica na diferença das distribuições dos dados. Os dados da diretividade estão concentrados num menor espectro de valores que o de *RSLL*, isso implica que a árvore consegue se adaptar melhor no primeiro que no segundo, já que a profundidade da árvore é menor e o mínimo de amostras por folha é maior, implicando numa árvore mais genérica.

4.5 *RANDOM FOREST*

O algoritmo *Random Forest* é uma combinação da simplicidade das árvores de decisão com maior flexibilidade e aleatoriedade para melhor precisão. São dezenas de árvores simples combinadas para predizer um bom resultado. A seleção de atributos é por aleatoriedade e não como anteriormente pelo cálculo de impureza. Essa método utiliza a técnica *Bagging* (*Bootstrapping Aggregation*), que tem como principal função tentar evitar o *overfitting*. Ele faz isso construindo vários estimadores que seleciona amostras aleatórias até fazer uma média de todos os estimadores construídos. Portanto, esse modelo tem ótimos motivos para ser utilizado, denominado *RandomForestRegressor* da Seção *Ensemble* da biblioteca *Scikit-Learn*.

REFAZERFoi estruturado da mesma forma que o anterior, seus resultados estão dispostos abaixo.

Diretividade:

- Max_Depth = 10;
- Min_Samples_Leaf = 1;
- Min_Samples_Split = 2;
- Num_estimators: 200
- R2[%] de Teste = 96.

RSLL:

- Max_Depth = 10;
- Min_Samples_Leaf = 1;
- Min_Samples_Split = 2;
- Num_estimators: 200
- R2[%] de Teste = 83.

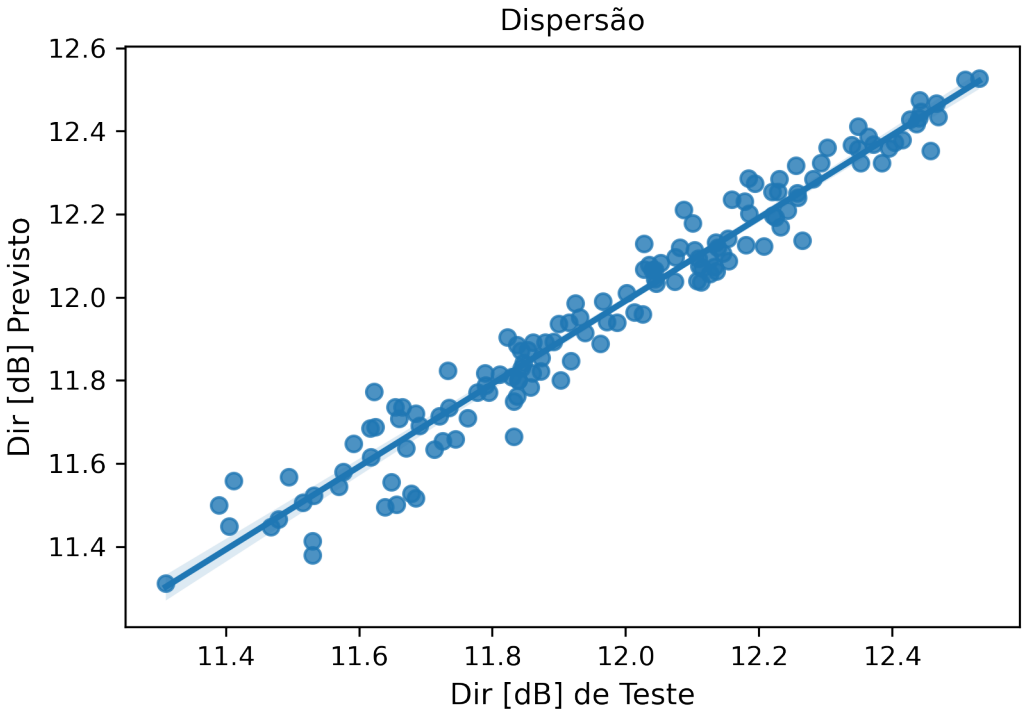


Figura 24 – Fonte: Próprio autor

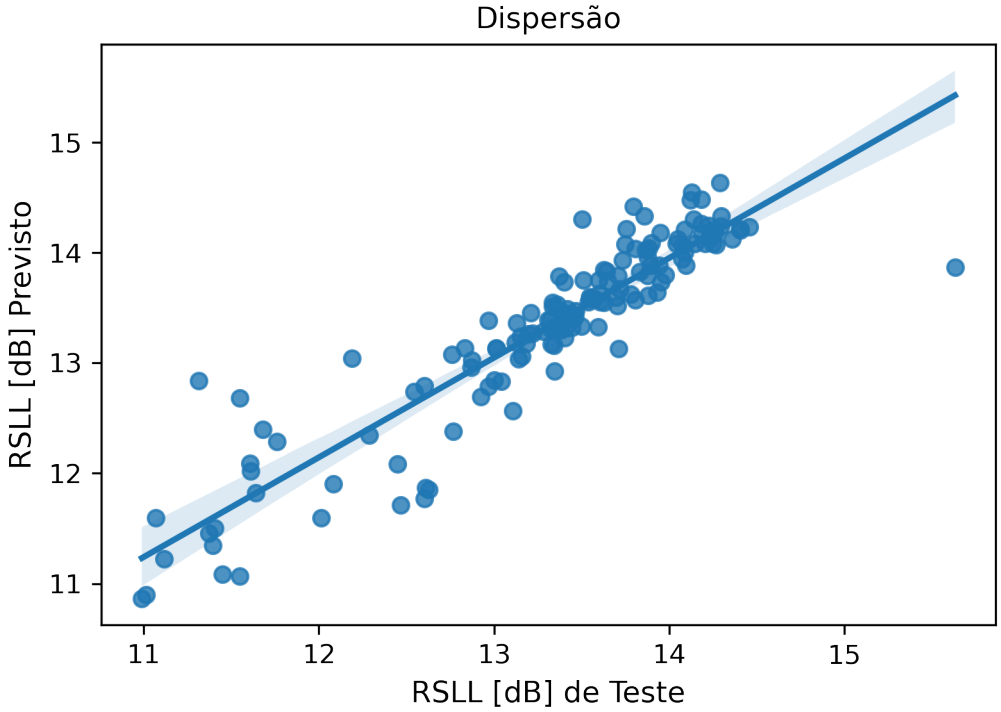


Figura 25 – Fonte: Próprio autor

As dispersões dispostas pela Figura 24 e Figura 25 são as melhores obtidas nos modelos de aprendizado de máquina mostrado no trabalho. Esses gráficos revelam que a evolução na métrica de desempenho foi obtida e comprovadas pelos mesmos, já que a distância euclidiana dos previstos

diminuíram em relação aos reis, tornando o modelo mais assertivo e também mais genérico para novos dados.

4.6 REDES NEURAIS ARTIFICIAIS

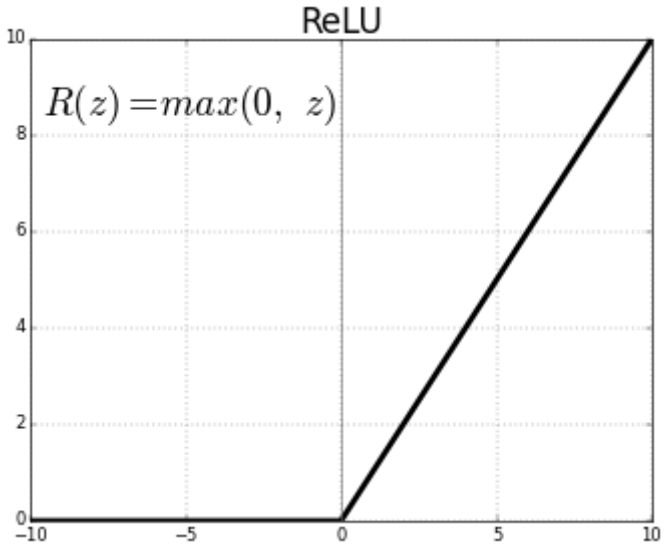
Os dados que foram usados para a implementação da rede neural são os da **??**, os mesmo utilizados para todos os modelos computacionais. O notebook utilizado foi o *Google Colab*, pela facilidade de utilizar placas de vídeo para o processamento mais ágil. Antes de implementar o modelo apresentado, foram feitas várias análises exploratórias para compreender experimentalmente o funcionamento do conjunto.

Como já foi falado, os alvos das previsões são os rótulos Diretividade e *RSLL*. A primeira etapa do desenvolvimento é atribuir uma arquitetura para a rede neural, ela foi implementada com uma camada de entrada, uma camada escondida e uma camada de saída, todas lineares.

- Camada de entrada: 4 neurônios;
- Camada escondida: 96 neurônios e e respectivas funções de ativação;
- Camada de saída: 2 neurônios e 2 funções de ativação.

Existem diversas funções de ativação que podem ser utilizadas, a escolhida foi a *ReLu*(), amplamente utilizada na atualidade por sua não-linearidade. Seu comportamento está ilustrado pela Figura 26. O neurônio é somente ativado se o valor de retorno do gradiente for positivo, dessa forma alguns neurônios são bloqueados, tornando a rede eficiente e com menos custo de processamento computacional. A expressão $R(z) = max(0, z)$ indica o comportamento da função *ReLU*, já que $z = W * x + b$, onde W é o peso e b o *bias*.

Figura 26 – *ReLU*



Fonte: (RELU... 2022)

Após a arquitetura ser atribuída à rede, deve ser instanciados a função de custo e o otimizador, *L1Loss* e *Adam* respectivamente. Essas instancias são importantes para definir o processo de treinamento e validação, caracterizados por hiperparâmetros de treinamento: *batches*, *epochs* e iterações.

Definições:

- Iteração: um passo de otimização, obedecendo o fluxo de treinamento, sendo os dois primeiros de *forward* e os dois penúltimos *backward*:

  1. Operação das entradas da rede;
  2. Cálculo da função de perda;
  3. Cálculo do gradiente;
  4. Atualização de pesos;
  5. Volta para o primeiro passo.

- *Batch*: quantidades de amostras vistas por uma iteração. Essa quantidade interfere no comportamento da divergência e deve ser assertivo, portanto deve ser encontrado através de testes ou otimizações.

- *Epoch*: uma época é completada quando todas as amostras do conjunto de treino forma vistas pelo modelo. Em geral são testadas dezenas ou centenas de época para avaliar a convergência do modelo.

Todos os hiperparâmetros foram ajustados pelo Otimizador de Hiperparâmetros *Optuna*, tal *framework* foi utilizado de maneira superficial para ajudar os testes manuais a teres melhor eficácia. Os valores dos hiperarâmetros estão dispostos na Tabela 2.

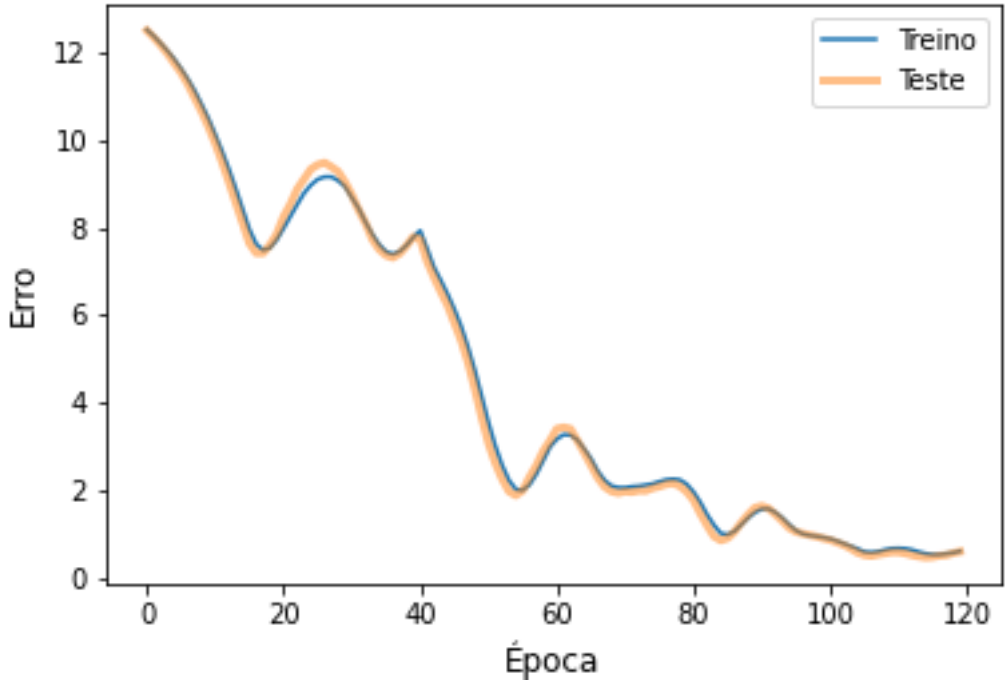
Tabela 2 – Hiperparâmetros otimizados

Hiperparâmetros	*Valor*
Tamanho do lote	5
Número de épocas	120
Taxa de aprendizado	$6,03e-05$
Decaimento de peso	$8,08e-06$

Fonte: Próprio autor

O modelo de uma rede neural, projetado para convergir com os valores das diretividades e das taxas de lóbulos laterais, obteve um bom resultado após as otimizações. A curva de convergência da função de perca pela quantidade de épocas está ilustrada pela Figura 27. O valor da função de erro relacionada aos dados de teste do modelo equivale a 4% do erro inicial da primeira época. A curva de convergência se assemelha ao modelo ideal de um aprendizado constante, embora haja algumas variações que podem estar relacionadas ao *RSLL*.

Figura 27 – Convergência da rede neural



Fonte: Próprio autor

5 CONCLUSÕES

O dispositivo simulado que foi utilizado para a extração de dados foi uma antena com 8 plaquetas arranjadas em série. É possível inferir que quando comparada a antena *patch* simples, a configuração de arranjo em série possui vantagens como direcionamento de feixe e aumento da diretividade, e desvantagens como surgimento de lóbulos laterais e alto custo de processamento computacional devido ser eletricamente grande. Conforme é adicionado elementos no arranjo, a tendência é de potencializar tais características.

Para realizar predições de diretividade e *RSLL*, foram utilizados modelos de aprendizagem de máquina supervisionado: regressão linear múltipla, regressão por árvore de decisão, regressão por árvores de decisão aleatórias e regressão utilizando redes neurais artificias. Para a construção desses modelos, foi necessário realizar uma análise de dados completa visando o entendimento e remanejo dos dados, além de calcular o nível de lóbulo lateral para cada configuração de largura das plaquetas. Ressalta-se que os parâmetros geométricos que reduzem as larguras dos *patches* implicam na mudança das características de radiação.

A análise de dados permitiu concluir que a diretividade possui alta relação linear com a mudança da largura dos *patches*, já o *RSLL*, não. Além disso foi notado que essas alterações dos valores de *RSLL* não possui explicação matemática analítica, por isso métodos mais complexos são interessantes para a modelagem da antena. A evolução de complexidade dos modelos contribuiu para a assertividade preditiva, já que árvores de decisão aleatória e redes neurais possuem maior capacidade de resolução de problemas devido suas arquiteturas de processamento. Os resultados obtidos podem ser reavaliados pela Tabela 3. A diretividade foi prevista com boa assertividade em todos os modelos, embora o *RSLL* não ter apresentado tal característica pelo método de regressão linear e regressão por árvore de decisão. Uma possível causa das ruins métricas do segundo alvo é que os dados de *RSLL* possui alguns valores discrepantes, possíveis *outliers*, que degrada a capacidade de aprendizagem do algoritmo para a realização dos testes. Uma etapa para trabalhos futuros é analisar se a função que calcula os níveis está configurada corretamente para captar os dois maiores picos da diretividade.

Tabela 3 – Métrica R2 para cada modelo e alvo preditivo

Método	Diretividade	*RSLL*
Regressão linear múltipla	0,95	0,62
Regressão por árvore de decisão	0,90	0,75
Regressão por árvores de decisão aleatórias	0,96	0,83
Regressão utilizando redes neurais artificiais	0,92	0,92

Fonte: Próprio autor

O projeto computacional construído atingiu o objetivo principal do trabalho, sendo capaz de realizar previsões assertivas em pouco tempo de processamento. O melhor candidato para realizar predição de diretividade é o método de regressão linear múltipla, já que é o mais simples e rápido com apenas 0,01 de diferença para o segundo mais assertivo, regressão por árvores de decisão aleatórias. Já

para predições de *RSLL*, as redes neurais artificiais foram as mais eficazes. Embora isso aconteça, seria interessante realizar predições de alvo único, ou seja, uma rede neural que realize regressão para o prognóstico de diretividade e outra para o *RSLL*, já que essas medidas não possuem mesmas características de correlação com as variáveis de entrada do sistema. Além disso, se novos dados de outros arranjos forem agregados ao modelo, novas previsões poderão ser realizadas. Essas consultas proporcionam ao pesquisador encontrar os melhores valores dos parâmetros *alphas* para menor nível de lóbulo lateral, diminuindo a interferência em outras frequências. Pesquisas futuras também podem ser direcionadas ao otimizador de hiperparâmetros *Optuna*, *framework* potente e tecnológico e bom candidato a ser o principal do ramo.

REFERÊNCIAS

BALANIS, C. **Antenna Theory: Analysis and Design**. Wiley, 2012. ISBN 9781118585733. Disponível em: <https://books.google.com.br/books?id=v1PSZ48DnuEC>.

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

LAURETTO, Marcelo S. **Árvores de Decisão**. 2010. Disponível em: <https://edisciplinas.usp.br/pluginfile.php/4469825/mod_resource/content/1/ArvoresDecisao_normalsize.pdf>. Acesso em: 06 ago. 2022.

O QUE é ciência de dados? Disponível em: <https://aws.amazon.com/pt/what-is/data-science/>. Acesso em: 06 ago. 2022.

IBM Cloud Education, **Machine Learning.** 2020. Disponível em: <https://www.ibm.com/br-pt/cloud/learn/machine-learning>. Acesso em: 04 jul. 2022.

CHEIN, Flávia. **Introdução aos modelos de regressão linear.** Brasília: Enap, 2019. 76 p. Disponível em: <https://repositorio.enap.gov.br/bitstream/1/4788/1/Livro_Regress~ao%20Linear.pdf.> Acesso em: 06 ago. 2022.

O QUE é Ciência de Dados? Disponível em: <https://www.oracle.com/br/what-is-data-science/.> Acesso em: 02 jul. 2022.

FÁVERO, Luiz P. **Análise de Dados.** Rio de Janeiro. Grupo GEN, 2015. ISBN 9788595153226. E-book. Disponível em: <https://integrada.minhabiblioteca.com.br/#/books/9788595153226/>. Acesso em: 10 ago. 2022.

Silva, Ivan Nunes da. **Redes neurais artificiais: para engenharia e ciências aplicadas**. São Paulo: Artliber Editora. ISBN: 9788588098534. Acesso em: 15 jun. 2022.

JACKSON, D. R.; OLINER, A. A. Leaky-wave antennas. **Modern Antenna Handbook**. John Wiley Sons, Ltd, 2008. cap. 7, p. 325–367. ISBN 9780470294154. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1002/9780470294154.ch7>

VALIDAÇÃO cruzada: **Avaliando o desempenho do estimador**. 2022. Disponível em: <https://scikit-learn.org/stable/modules/cross_validation.html#cross-validation.> Acesso em: 2 ago. 2022.

SINGLE-LAYER Neural Networks and Gradient Descent. Disponível em: <https://sebastianraschka.com/Articles/2015_singlelayer_neurons.html.> Acesso em: 15 jul. 2022.

TREINANDO uma Rede Neural: **Deep Learning com PyTorch.** Disponível em: <https://cursos.alura.com.br/course/treinando-rede-neural-pytorch.Acessoem:10jun.2022.>

ZULKIFLI, Hafidz.**UNDERSTANDING Learning Rate**.Disponível em: <https://towardsdatascience.com/https-medium-com-dashingaditya-rakhecha-understanding-learning-rate-dd5da26bb6de>. Acesso em: 01 jun. 2022.

WHAT is AI? Disponível em: <https://www.oracle.com/br/artificial-intelligence/what-is-ai/>. Acesso em: 05 jun. 2022.

RELU: Not a Differentiable Function: Why used in Gradient Based Optimization? and Other Generalizations of ReLU. Disponível em: <https://medium.com/@kanchansarkar/relu-not-a-differentiable-function-why-used-in-gradient-based-optimization-7fef3a4cecec>. Acesso em: 06 ago. 2022.

GALDINO, Renata. **O que é Ciência de Dados e os 6 passos para adoção de uma cultura orientada a dados.** Disponível em: <https://medium.com/@renatagaldino/6-passos-para-lidar-com-um-novo-problema-em-data-science-ef6ef88df475.> Acesso em: 16 ago. 2022.