UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE ENGENHARIA DE PRODUÇÃO E SISTEMAS
CURSO DE ENGENHARIA DE PRODUÇÃO ELÉTRICA

Marcella de Matos Bittencourt

Aplicação de Modelos Preditivos para a Geração de Energia Solar Fotovoltaica

Florianópolis
2022

Marcella de Matos Bittencourt

Aplicação de Modelos Preditivos para a Geração de Energia Solar Fotovoltaica

Trabalho de Conclusão do Curso de Graduação em Engenharia de Produção Elétrica do Centro Tecnológico da Universidade Federal de Santa Catarina como requisito para a obtenção do título de Bacharel em Engenharia Elétrica com Habilitação em Engenharia de Produção
Orientador: Prof. Mauricio Uriona Maldonado, Dr..

Florianópolis
2022

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC

Bittencourt, Marcella de Matos
Aplicação de Modelos Preditivos para a Geração de Energia Solar Fotovoltaica / Marcella de Matos Bittencourt ; orientador, Mauricio Uriona Maldonado, 2022.
104 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, , Graduação em , Florianópolis, 2022.

Inclui referências.

1. . 2. Machine Learning. 3. Energia Solar Fotovoltaica. 4. Predição de Geração de Energia. I. Maldonado, Mauricio Uriona. II. Universidade Federal de Santa Catarina. Graduação em . III. Título.

Marcella de Matos Bittencourt

Aplicação de Modelos Preditivos para a Geração de Energia Solar Fotovoltaica

Este Trabalho Conclusão de Curso foi julgado adequado para obtenção do Título de Bacharel em Engenharia Elétrica com Habilitação em Engenharia de Produção e aprovado em sua forma final pelo Curso Engenharia de Produção Elétrica

Florianópolis, 21 de dezembro de 2022.

_____

Prof.(a) Mônica Maria Mendes Luna, Dr.
Coordenador do Curso

Banca Examinadora:

Documento assinado digitalmente
Mauricio Uriona Maldonado
Data: 21/12/2022 16:53:14-0300
CPF: ***.972.939-**
Verifique as assinaturas em https://v.ufsc.br

_____

Prof. Mauricio Uriona Maldonado, Dr.
Orientador
Universidade Federal de Santa Catarina

_____

Prof. Lynceo Falavigna Braghirolli, Dr.
Avaliador
Universidade Federal de Santa Catarina

_____

Prof. Rafael Marcuzzo, Dr.
Avaliador
Universidade do Estado de Santa Catarina – Campus Joinville

AGRADECIMENTOS

Dedico esse trabalho a meus pais, irmãos, minhas avós, meus sogros e familiares que deram o suporte necessário durante toda a jornada. Agradeço especialmente a minha namorada Maria Eduarda, que foi responsável por todo suporte e incentivo para que esse trabalho fosse concluído. Gostaria de dedicar também a todos os Professores que foram responsáveis pela minha formação ao longo da minha vida. E por fim, agradecer a Universidade Federal de Santa Catarina e meus colegas que me proporcionaram o ambiente de crescimento ao longo da graduação.

RESUMO

A previsibilidade de produção de energia elétrica é um dos grandes desafios do sistema elétrico brasileiro dada a atual expansão da geração proveniente de fontes como a fotovoltaica e eólica. O presente trabalho objetiva aplicar modelos preditivos baseados em Machine Learning para a previsão de potência de saída de placas fotovoltaicas. Para isto, foram utilizados dados de doze usinas solares instaladas no hemisfério norte, os quais foram classificados e submetidos a modelos de aprendizado de máquina baseados em Ridge, Árvore de Decisão e XGBoost. A partir dos resultados obtidos se realizou uma análise comparativa entre dados reais e os previstos pelos modelos utilizando as métricas MAE, MSE e R^2. O modelo de aprendizagem XGBoost apresentou maior assertividade na previsão e a média dos seus valores de erro a partir das métricas utilizadas foram 3,71 para MAE, 22,75 para MSE e 0,39 para R^2. Os resultados apontam para o fato de que a utilização de sistemas de previsões computacionais se constitui como uma boa forma de prever produção de energia solar fotovoltaica a curto prazo.

Palavras-chave: Ridge. Árvore de Decisão. XGBoost. Energia Fotovoltaica.

ABSTRACT

The predictability of electricity production is one of the great challenges of the Brazilian electrical system given the current expansion of generation from sources such as photovoltaic and wind. The present work aims to apply predictive models based on Machine Learning to predict the output power of photovoltaic panels. For this, data from twelve solar plants installed in the northern hemisphere were used, which were classified and submitted to machine learning models based on Ridge, Decision Tree and XGBoost. From the results obtained, a comparative analysis was carried out between real data and those predicted by the models using the MAE, MSE and R^2 metrics. The XGBoost learning model showed more assertiveness in the prediction and the average of its error values from the metrics used were 3.71 for MAE, 22.75 for MSE and 0.39 for R^2. The results point to the fact that the use of computational forecasting systems is a good way to predict the production of photovoltaic solar energy in the short term.

Keywords: Ridge. Decision Tree. XGBoost. Photovoltaic Energy.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ANEEL - Agência Nacional de Energia Elétrica

ARIMA - Autoregressive Integrated Moving Average

CA - Corrente alternada

CC - Corrente contínua

IA - Inteligência Artificial

IEA - Agência Internacional de Energia *(International Energy Agency*)

IRENA - Agência Internacional de Energia Renovável *(International Renewable Energy Agency)*

JDMT - Jonathan Dickinson State Park

LSTM - Long Short-Term Memory

MAE - Erro Médio Absoluto

MLP - Multilayer Perceptron

MNANG - Aeroporto Internacional de Minneapolis-Saint Paul

MSE - Erro Médio Quadrado

ONS - Operador Nacional do Sistema

OOP - Programação Orientada a Objetos

R^2 - Coeficiente de Determinação

RMSE - Raiz quadrada do erro-médio

RNA - Redes Neurais Artificiais

SIN - Sistema Interligado Nacional

USAFA - Academia da Força Aérea EUA

XGBoost - eXtreme Gradient Boosting

SUMÁRIO

1 INTRODUÇÃO

Em um mundo que necessita cada vez mais de Energia e onde discutem-se a necessidade de que essa Energia seja limpa e renovável a importância da matriz fotovoltaica e de que estudos relacionados à mesma sejam realizados é visível. Apesar do Brasil ter uma das matrizes energéticas mais limpas do mundo, tendo 62,5% da sua matriz constituída por fonte hidroelétrica segundo o Operador Nacional do Sistema Elétrico (ONS, 2022), se faz necessário a diversificação da matriz energética já que alterações nos regimes de chuvas, principalmente onde estão as maiores usinas, causa insegurança energética.

Com isso, uma alternativa na busca de diversificar a matriz energética e ao mesmo manter o caráter limpo e renovável da matriz brasileira é a utilização da fonte fotovoltaica considerando a alta irradiação solar e o clima do Brasil. Porém, a geração de Energia Elétrica utilizando fontes fotovoltaicas depende de diversos fatores ingovernáveis da natureza o que leva a uma dificuldade de previsão de geração dessa energia.

No presente trabalho será aplicado a tecnologia de Machine Learning utilizado dados de 12 Usinas Fotovoltaicas para antever a produção de Energia Elétrica Fotovoltaica, com o intuito de tornar mais previsível e atrativa essa fonte energética.

1.1 DESCRIÇÃO DO PROBLEMA

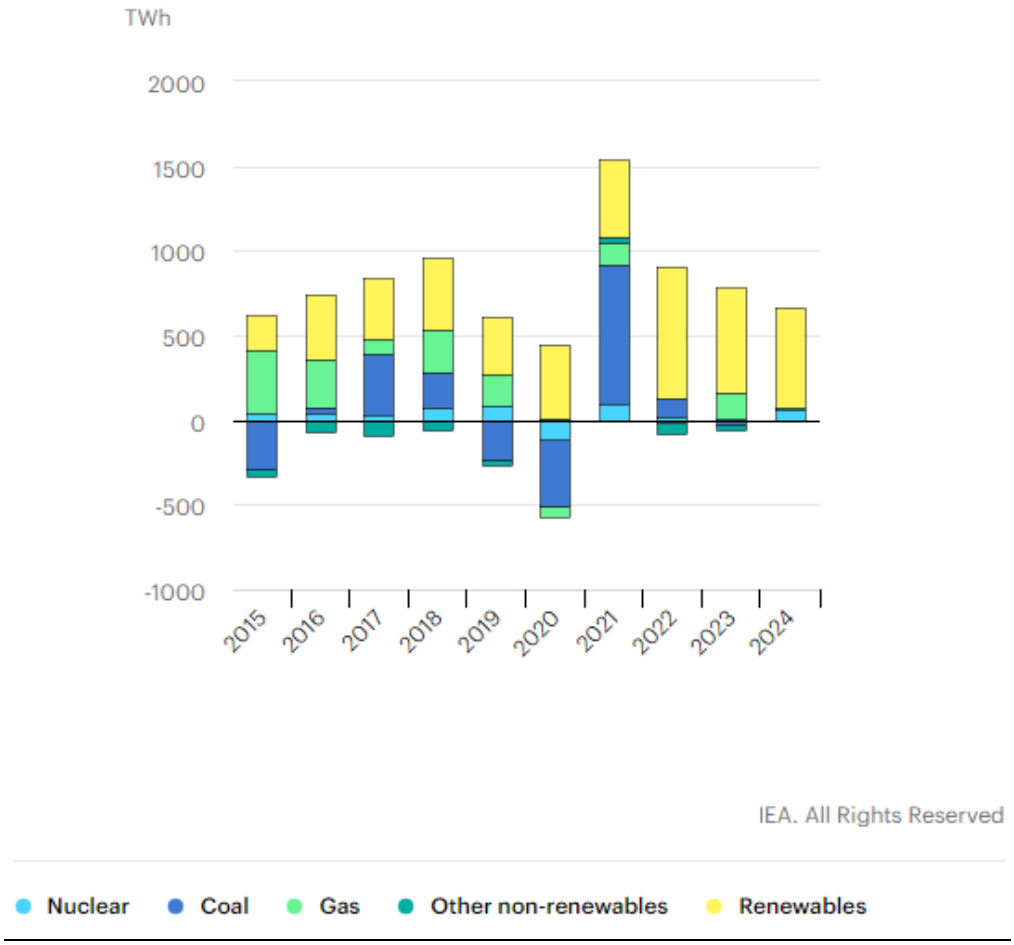
Nos dias atuais, em que o mundo se encontra cada vez mais dependente do uso da energia elétrica visto que a mesma desempenha papéis fundamentais (TURMINA et al., 2018) é importante que se considere as fontes pelas quais ela é produzida a fim de diminuir os impactos negativos provenientes do processo de geração de eletricidade. Além do fato dos recursos naturais serem finitos, há uma grande preocupação quanto aos impactos ambientais gerados a partir do uso de algumas matrizes energéticas, direcionando para a busca e utilização de fontes alternativas de energia que contribuam tanto na garantia da sustentabilidade e da preservação do meio ambiente quanto para o desenvolvimento econômico e social (REZENDE, 2019).

A utilização dessas fontes alternativas gera a chamada energia limpa, que são provenientes de matérias naturais e renováveis, não poluentes e inesgotáveis (PINTO, 2013). A principal diferença da modalidade energética renovável é que a mesma não emite gases

prejudiciais na atmosfera durante sua produção, o que ocorre na utilização de fontes provenientes do petróleo, gás natural, carvão mineral, etc., gerando menores impactos nos ecossistemas. Dessa forma, por possuírem grandes benefícios, a busca pela utilização de matrizes energéticas renováveis se torna cada vez mais necessária e presentes, sendo motivada principalmente pela escassez de recursos naturais e impactos ambientais - o que resulta em um maior reconhecimento e representação das mesmas nas matrizes energéticas mundiais.

A Figura 1 ilustra o acréscimo e decréscimos na geração de energia elétrica no mundo de diversas matrizes, em amarelo destaca-se o constante e expressivo crescimento das matrizes renováveis, principalmente em 2022 e a sua projeção até 2024.

Figura 1 - Mudanças na geração de energia elétrica global de 2015 à 2024.



Fonte: Agência Internacional de Energia - IEA (2022).

A energia solar pode ser utilizada a partir de várias formas, possibilitando a utilização direta para a iluminação, para o aquecimento de fluidos, aquecimento de ambientes, para a geração de potência mecânica ou elétrica como fonte de energia térmica, ou então ser convertida

diretamente em energia elétrica a partir do efeito fotovoltaico (DOS SANTOS et al., 2021). A partir disso, como importante fonte de energia limpa e renovável encontra-se a energia solar fotovoltaica, que dentre as diversas possíveis aplicações da energia solar, apresenta-se como uma das mais eficientes formas de gerar potência elétrica (RÜTHER, 2004). Nela, se utiliza de células fotovoltaicas para a conversão da radiação solar em energia elétrica, a partir do chamado efeito fotovoltaico (BEZERRA, 2021).

De forma sucinta, a energia solar fotovoltaica é gerada através da conversão direta de radiação solar em eletricidade. Esse processo ocorre através do uso de células fotovoltaicas, compostas por um material semicondutor (geralmente silício), que são submetidas a radiação solar e converte tal energia em potencial elétrico a partir do efeito fotoelétrico ou fotovoltaico (IMHOFF, 2017). O funcionamento do sistema fotovoltaico demanda da utilização de equipamentos auxiliares, como por exemplo o caso de inversores, visto que a energia elétrica gerada inicialmente é de corrente contínua (CC), precisando ser transformada em corrente alternada (CA).

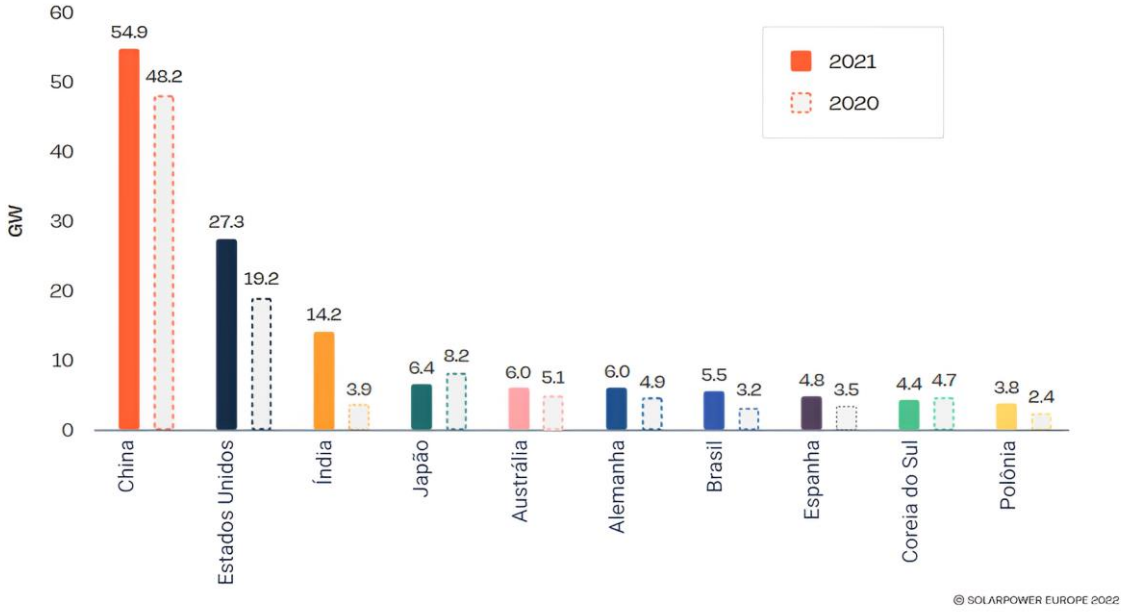
Conforme o Global Solar Atlas (2022), a radiação solar passível de aproveitamento para geração de energia elétrica apresenta altos valores no mundo todo, possibilitando que a utilização do efeito fotovoltaico para a geração de energia elétrica conquiste cada vez mais reconhecimento na composição das matrizes energéticas mundiais, apresentando-se como uma tecnologia em constante avanço (LANA et al., 2015). Conforme apresentado por Bezerra (2021), a utilização da fonte solar na geração de energia elétrica mundial em 2019 correspondia a 2,68% do total gerado, que mesmo ainda sendo um número pequeno, corresponde a um crescimento expressivo e exponencial.

Já entre 2010 e 2019 a capacidade de geração de energia elétrica a partir da matriz solar cresceu cerca de 30,8% ao ano no mundo todo. No final de 2019, a capacidade instalada de energia solar mundial alcançou 586,4 GW, o que foi 20,0% acima do esperado e aproximadamente 15 vezes maior do que era em 2010 (BEZERRA, 2021). Bezerra (2021) ainda cita que esse aumento expressivo da geração solar tende a se manter nos próximos anos.

Já em 2015 a China alcançou a liderança da capacidade instalada de geração fotovoltaica do mundo, correspondendo a 25,8% da produção global. Já a Alemanha, líder anterior em tal setor, vem perdendo participação relativa ao longo dos anos. A capacidade instalada de geração fotovoltaica no Brasil ainda é pouco expressiva, representando em 2019 apenas 0,4% da capacidade instalada mundial em geração fotovoltaica (BEZERRA, 2021).

Conforme Figura 2, em 2021 os 10 países com maior capacidade de geração de energia elétrica de matriz fotovoltaica instalada eram: China, Estados Unidos, Índia, Japão, Austrália, Alemanha, Brasil, Espanha, Coreia do Sul e Polônia. A Figura 2 ainda apresenta a evolução da capacidade de geração instalada entre 2020 e 2021:

Figura 2 - 10 países com maior capacidade de geração de energia elétrica de matriz fotovoltaica instalada e sua evolução entre 2020 e 2021.



Fonte: SolarPower Europe (2022).

A partir das informações apresentadas acima, fica evidente a importância que a energia elétrica de matriz fotovoltaica vem adquirindo e conquistando ao decorrer dos anos. Tal fato direciona para a necessidade de adequados modelos de previsão, que dentre outros objetivos, possibilitem melhorar e orientar a previsão do retorno de investimentos, ajudando no aumento da rentabilidade e na confiabilidade da área (DAS et al., 2018).

Conforme apresentado por Voyant et al. (2017), a previsão da potência de saída dos sistemas solares é de extrema relevância e necessidade, sendo indispensáveis para a redução de custos na produção e para uma efetiva operação da rede elétrica, possibilitando o gerenciamento ideal dos fluxos de energia, construção de estimativas de reservas, programação do sistema elétrico, etc. Atualmente são utilizados diversos métodos de previsão para antever a geração de energia fotovoltaica, que são divididos em quatro grandes grupos: inteligência artificial,

abordagens estatísticas, métodos físicos e também híbridos (ALKHAYAT, MEHMOOD, 2021).

Os autores Urzagasti et al. (2021) apontam que os métodos baseados em Machine Learning são os que apresentam as melhores predições. O uso de séries temporais e regressão associados a Machine Learning vem obtendo resultados satisfatórios na previsão de geração de energia fotovoltaica, visto que o conceito de séries temporais utiliza parâmetros importantes, tais como: variações do tempo, clima e sazonalidade (ANDRADE et al. 2021). As redes neurais artificiais também ocupam posição importante em tal predição pelo fato de que as mesmas possuem capacidade de "aprender" padrões através de treinamento, o que gera vantagem sob a especificidade de que os dados envoltos à geração de energia fotovoltaica não são lineares (CUNHA, SOBEL, 2021). Outros algoritmos como árvore de decisão, regressões lineares e XGBoost também se mostram eficazes.

De forma geral, as previsões de curto prazo são utilizadas para o controle de geração, gerenciamento ideal dos fluxos de energia e gerenciamento de planta de geração. Já modelos de longo prazo são utilizados para avaliar o comprometimento de unidade, balanceamento de carga e sua programação. Tais previsões, tanto de curto, médio e longo prazo, também são importantes para o planejamento da infraestrutura das plantas de geração (NESPOLI et al., 2019).

A previsão de geração de energia geralmente se realiza considerando as próximas horas (0h-6h, curto prazo ou 6h-24h, médio prazo) ou para os próximos dias (24h ou mais, longo prazo). A previsão de curto prazo muitas vezes acaba não sendo suficiente ou efetiva para um controle eficiente da operação da rede elétrica - priorizando-se as de médio ou longo prazo (MELLIT, PAVAN, 2010).

Importante ressaltar que tais modelos de previsão são de extrema utilização e importância em países que o mercado de eletricidade se dá a partir da estimativa de geração do dia seguinte, visto que tais modelos de previsão permitem maior rentabilidade econômica e um melhor planejamento energético para o país (DAS et al., 2018).

O número de artigos publicados sobre o tema de previsão de energia fotovoltaica vem crescendo de forma expressiva nos últimos anos, os quais utilizam diferentes combinações entre métodos de previsão e seus horizontes. Dentre estes destaca-se alguns, como o que combina modelo de inteligência artificial com previsão de longo prazo (SILVA et al. 2022), inteligência

artificial com curto e médio prazo (SANTOS, 2019), de comparação entre técnicas mais eficazes de previsão para 24h (NESPOLI et al., 2019), entre outros.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

O presente trabalho tem como objetivo propor um modelo de previsão para geração solar fotovoltaica mais preciso baseado em Machine Learning.

1.2.2 Objetivos Específicos

Dentro do objetivo geral apresentado os objetivos específicos são:

- Classificar os dados referentes às usinas analisadas.
- Definir métodos de Machine Learning que serão utilizados.
- Treinar os modelos de Machine Learning previamente escolhidos.
- Comparar os modelos treinados utilizando como critério indicadores de acurácia.

1.2 JUSTIFICATIVA

Levando em consideração as tecnologias atuais, há uma dificuldade de armazenar grandes volumes de energia e as cargas aplicadas ao sistema demandam dele flexibilidade em se adaptar constantemente (ROHLEDER, 2021). A partir disso, se faz necessário que as matrizes energéticas tenham boa previsibilidade, proporcionando uma operação estável do sistema e a confiabilidade das mesmas (ANTONIOLLI, 2021).

A geração de energia fotovoltaica vem se destacando nos últimos anos e se tornando um dos mercados mais promissores no campo de energias renováveis (BEZERRA, 2021), porém a mesma possui sua previsão como desafio. Levando em consideração sua grande expansão e o fato de que a mesma está condicionada a diversos fatores não lineares, como a irradiação solar e demais fatores climáticos, modelos eficazes para sua previsão se tornam cada vez mais necessários - campo que Machine Learning ganha destaque (BERGAMO, RAMOS, 2022).

A literatura apresenta que os benefícios de uma boa previsão de geração são inúmeros, desde melhora na análise da viabilidade de empreendimentos de energia solar (SILVA et al., 2020), na avaliação da necessidade de complementações do abastecimento com outras fontes (MARINHO et al., 2020), diminuição dos gastos e melhorias na manutenção, armazenamento e comercialização da energia (DOMINGOS et al. 2020).

Se tratando de modelos de previsão para geração de energia elétrica, a inteligência artificial aplicada a Machine Learning se destaca visto que se constitui como uma ferramenta com a capacidade de aprender padrões (LUDERMIR, 2021). A literatura confirma a eficiência de tal método, que é utilizado a partir de algoritmos diferentes (BERGAMO, RAMOS, 2022; DA SILVA, 2022; URZAGASTI, 2021; WENTZ, 2021).

Especificamente para a previsão de geração de energia fotovoltaica, que com já dito, está condicionada a fatores não lineares, Domingos et al. (2020) afirma que Machine Learning também se consolida como uma das abordagens mais populares, sendo utilizada principalmente para a previsão de curto prazo.

Ao utilizar tais ferramentas, são enfrentados diversos desafios. Como o apresentado por Santos (2019) ainda não existe uma maneira de construção de um modelo perfeito de Machine Learning, de forma com que a seleção e adequação dos parâmetros utilizados geralmente se dá através de tentativa e erro. Além disso, como já citado, no geral os estudos preveem a curto prazo, visto a dificuldade de construção de um modelo que comporte a previsão de longo prazo. Outro desafio apresentado por Sonia et al. (2017) refere-se à necessidade de um grande conjunto de dados de treinamento para que se possa atingir melhor desempenho dos modelos.

Perante a necessidade de construção de modelos cada vez mais precisos para a previsão de energia fotovoltaica e estudos sobre os mesmos, que considerem suas especificidades, o presente trabalho se propõe a analisar a utilização de diferentes tipos de algoritmos de Machine Learning na aplicação em questão. Como dito, predições mais assertivas são de imensa importância por diversos motivos, como a atração de mais investimentos financeiros - visto que seria possível dimensionar de forma mais confiável a projeções de retornos financeiros.

1.3 ESTRUTURA DO TRABALHO

O primeiro capítulo do trabalho possui caráter introdutório, apresentando uma contextualização e delimitando o problema central do trabalho. Neste capítulo realiza-se a

definição do objetivo geral e dos objetivos específicos, determinando a direção que o trabalho seguirá.

O segundo capítulo, fundamentação teórica, possui o objetivo de aproximação e fundamentação dos conceitos que são abordados ao longo do estudo a partir de revisões bibliográficas e referenciais teóricos. O mesmo comporta os tópicos “Energia Elétrica Fotovoltaica”, “Modelagem Preditiva”, “Machine Learning” e por fim “Síntese de Pesquisas”.

O terceiro capítulo aborda os procedimentos metodológicos, que se constituem como os passos e instrumentos que serão utilizados no decorrer do estudo. Além disso, em tal capítulo aborda-se o conjunto de dados que serão utilizados no capítulo seguinte.

O quarto capítulo, o desenvolvimento, trata do estudo em si a partir do que já foi delimitado nos capítulos anteriores. Tal capítulo aborda a análise exploratória dos dados, segmentação dos dados, aplicação dos algoritmos, apresentação e comparação dos resultados. Constitui-se como um dos principais capítulos do estudo visto que é nele que se aplica tudo aquilo conceituado na fundamentação teórica a fim de buscar atender os objetivos delimitados.

Por fim, o quinto e último capítulo trata sobre as conclusões e resultados obtidos a partir do estudo, além de conter apontamentos importantes identificados ao longo do desenvolvimento do mesmo.

2 FUNDAMENTAÇÃO TEÓRICA

A seguir serão apresentados os conceitos mais relevantes para a contextualização do trabalho. Serão abordados os tópicos sobre Energia Elétrica Fotovoltaica que será a matriz energética analisada neste trabalho e a Modelagem Preditiva que será utilizada para a análise dos dados.

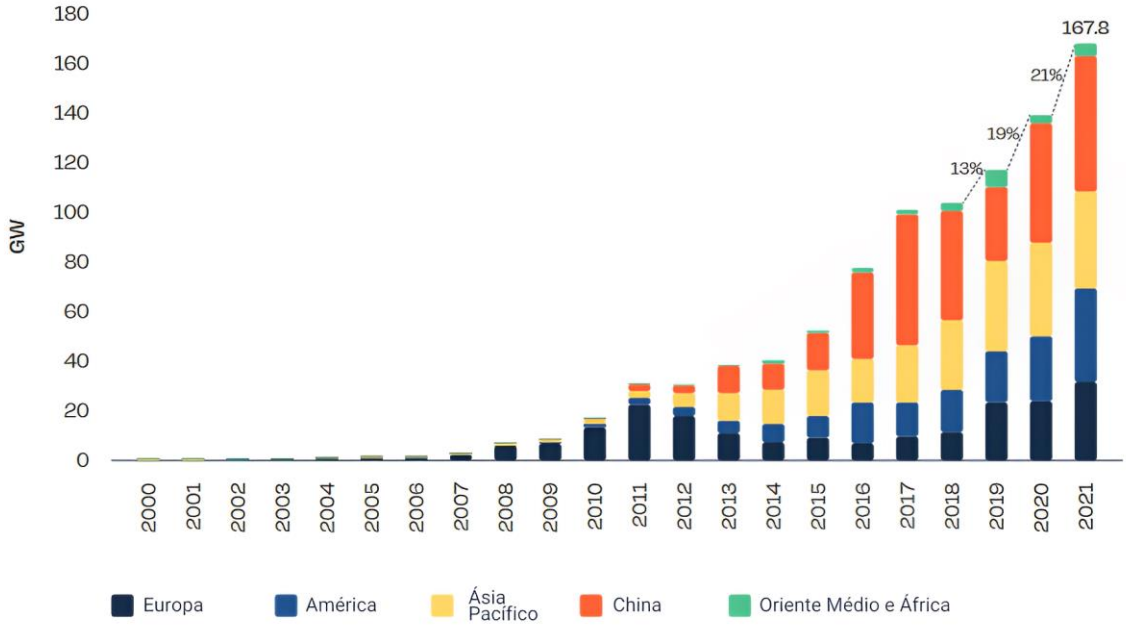
2.1 ENERGIA ELÉTRICA FOTOVOLTAICA

A cada dia o mundo aumenta sua demanda por energia elétrica, a qual é utilizada tanto durante o desenvolvimento de atividades cotidianas quanto na indústria. A maioria da população desconhece qual a origem e fonte da energia elétrica que está utilizando, mas é possível observar um avanço nas discussões sobre as consequências da geração de energia por meio de fontes poluidoras, fazendo com que o mercado energético passe a buscar por alternativas que gerem os menores impactos ambientais possíveis (OLIVEIRA, 2011).

A partir do que foi exposto acima, as energias renováveis passam a ocupar um lugar importante no fornecimento de eletricidade visto seus baixos impactos ao meio ambiente, sendo a energia fotovoltaica um grande exemplo delas (OLIVEIRA, 2011).

Nos últimos anos vem se expandindo bastante a capacidade instalada de geração fotovoltaica pelo mundo, incluindo no Brasil. Porém a participação brasileira ainda é pequena visto a produção mundial que está ilustrada na Figura 3. Em contrapartida o Brasil tem um potencial gigantesco para produção de Energia Elétrica Fotovoltaica, conforme Sauaia (2017).

Figura 3 - Evolução da capacidade instalada de geração fotovoltaica no mundo.



Fonte: SolarPower Europe (2022).

Segundo Castro (2002), a Energia Elétrica Fotovoltaica é gerada a partir do efeito fotoelétrico, onde um material semicondutor, geralmente o silício, quando submetido a radiação solar converte essa energia em potencial elétrico. Com isso, todos os fatores climáticos que afetam a radiação solar afetam também a produção de Energia Elétrica Fotovoltaica.

Existem diversos modelos e marcas de painéis solares, com várias potências de geração e várias tensões de saída, esses se diferenciam principalmente pelo tipo de estrutura cristalina do silício, podendo ser monocristalinas e policristalinas (LEMOS, 2022). Ainda conforme apresentado por Lemos (2022), células monocristalinas são mais eficientes, logo necessitam de menor espaço para gerar a mesma quantidade de energia e são recomendadas principalmente em locais onde há uma menor incidência de luz solar. Já as policristalinas são por sua vez menos eficientes, porém são uma solução mais barata e sua fabricação gera menos resíduos.

A energia gerada em aerogeradores e geradores hidráulicos, em maioria das vezes, geram energia de corrente alternada (CA), que é o padrão utilizado no mundo para transmissão e distribuição de energia, salvo poucas exceções, visto sua vantagem diante das perdas e facilidade de alterações nos níveis de tensão (PASCUETTI, 2021). Já a corrente que sai das células que constituem os painéis solares é contínua (CC) e para que se integre no sistema de transmissão necessita ser convertida (RODRIGUES et al., 2022).

2.2 MODELAGEM PREDITIVA

Existem muitas vantagens em poder prever o comportamento de diversos cenários, fato que pode ser concretizado a partir do uso de técnicas computacionais que vem se desenvolvendo e sendo utilizadas em diversos problemas de diferentes áreas, viabilizando a modelagem preditiva. No setor energético é possível observar que essas ferramentas vêm sendo cada vez mais utilizadas e obtendo resultados positivos (BERGAMO, RAMOS, 2022), podendo ser aplicadas em matrizes energéticas que possuem interferência direta de fatores climáticos, como o caso de usinas eólicas (CRUZ et al., 2022), usinas hidrelétricas sem reservatório chamadas de fio d'água (OLIVEIRA, 2021), e a matriz que será a analisada neste estudo que é a solar (CUNHA, SOBEL, 2021).

A Modelagem Preditiva busca um método baseado em modelos matemáticos e estatísticos como uma forma de antever um evento futuro ou saídas em um determinado período de tempo (BERGAMO, RAMOS, 2022). Esses modelos se baseiam em relacionar dados passados com a projeção de dados futuros. Recentemente se tem empregado inteligência artificial nas modelagens preditivas, a fim de refinar os resultados, com o avanço da capacidade de processamento das máquinas a aplicação de técnicas computacionais para resolução de problemas com muitas variáveis ficou cada vez mais acessível e rápida (BARI et al., 2020).

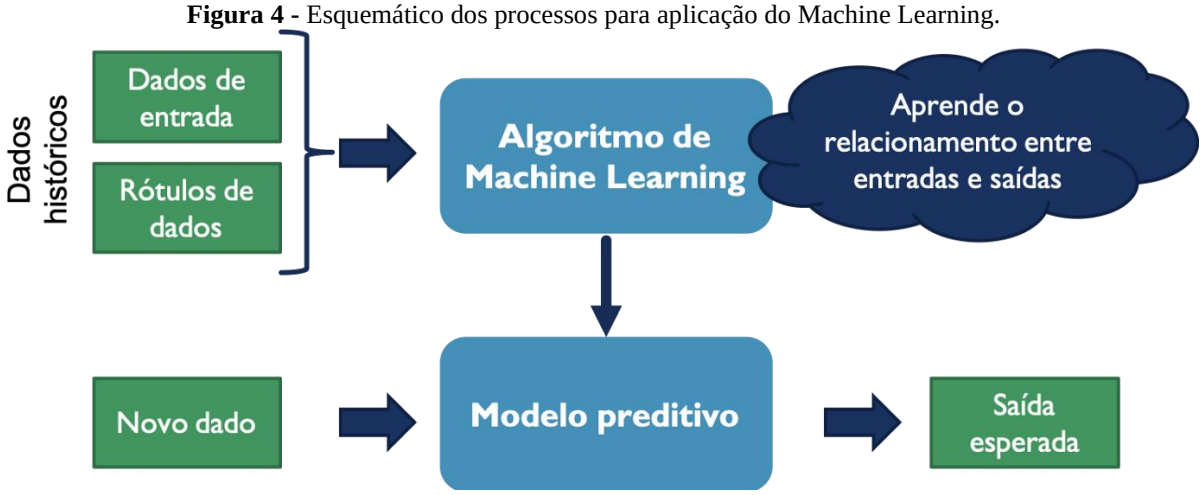
A inteligência artificial, que Kaufman (2019) delimita como a ciência e engenharia de criação de máquinas com funções que poderiam ser exercidas por cérebros, envolve capacidades como a de raciocínio, aprendizagem e aplicação de regras lógicas. Russell e Norvig (2013), definem a inteligência artificial a partir de quatro categorias: sistemas que pensam como seres humanos, sistemas que pensam racionalmente, sistemas que agem como seres humanos, e sistemas que agem racionalmente. Uma das técnicas de inteligência artificial é o Machine Learning, ou aprendizado de máquina, que conforme Liu e Zhang (2016) explicita é a modelagem de um algoritmo computacional com o objetivo de aprender com a repetição.

2.3 MACHINE LEARNING

O termo Machine Learning surge como um subcampo da inteligência artificial que possui como finalidade possibilitar aos computadores a capacidade de aprender sem serem programados (KAUFMAN, SANTAELLA, 2020). A mesma constitui-se como um processo

para solução de problemas específicos a partir da construção algorítmica de modelos estatísticos baseados em determinados conjuntos de dados (BURKOV, 2019 apud KAUFMAN, SANTAELLA, 2020). A Figura 4 apresenta um esquemático dos processos para aplicação do Machine Learning.

Figura 4 - Esquemático dos processos para aplicação do Machine Learning.



Fonte: Escovedo e Koshiyama (2020).

Desta maneira, na aplicação da técnica Machine Learning, a partir da exposição a um determinado conjunto de dados, que representam as variáveis e a solução de um determinado problema, os sistemas de aprendizado de máquina adquirem a capacidade de tomada de decisões baseando-se nas conclusões obtidas a partir da inferência indutiva sobre o conjunto de dados.

Antes de aplicar propriamente os algoritmos de Machine Learning ainda há a etapa de pré-processamento de dados. Conforme Sivakumar e Gunasundar (2017) apresentam, tal etapa é composta por diversas fases, sendo um momento de limpeza dos dados em busca de uma melhor assertividade do modelo. Nesta etapa, retira-se dados e valores que não são relevantes, ajusta-se dados e valores que não se encontram nos padrões, etc. - tornando o conjunto de dados pronto para os próximos passos de análises e treinamento.

Segundo Ludermir (2021), a inferência indutiva é um dos principais métodos para construção de conhecimento e predição de eventos futuros sobre o fenômeno a ser estudado. No aprendizado de máquina a inferência indutiva é composta por duas etapas, sendo elas o processo de indução e a de dedução. Na primeira etapa o algoritmo estabelece relações

funcionais entre as variáveis independentes e as variáveis dependentes. Já na segunda etapa, utiliza-se a relação obtida na etapa anterior para realizar a classificação do conjunto de dados.

Ainda segundo o autor, os sistemas de aprendizado que utilizam da inferência indutiva podem ser classificados em duas classes: os supervisionados e os não supervisionados. Os sistemas supervisionados serão abordados a seguir, e nos sistemas não supervisionados a máquina analisa os dados oferecidos como treino para identificar padrões e encontrar agrupamentos de dados que se assemelham, os classificando.

No setor energético, a aplicação de técnicas de aprendizado de máquina origina estudos que possibilitam, entre outras coisas, a implementação de planos energéticos e políticas públicas mais assertivas (BORGES, 2021), o que é um importante passo na direção da segurança energética e da parametrização das tomadas de decisões.

2.3.1 Sistema de Aprendizado Supervisionado

Nos sistemas de aprendizado supervisionado a máquina recebe um conjunto de dados para treinamento, servindo como exemplo para identificar as relações entre as variáveis independentes e os rótulos das classes pertencentes.

Dessa maneira, o aprendizado supervisionado requer um conjunto de dados já conhecidos que serão divididos em variáveis de entrada e variáveis de saída de forma com que os dados já rotulados passam a servir de aprendizado para que o modelo consiga estimar novos dados.

Conforme apresentado por Osisanwo et. al (2017):

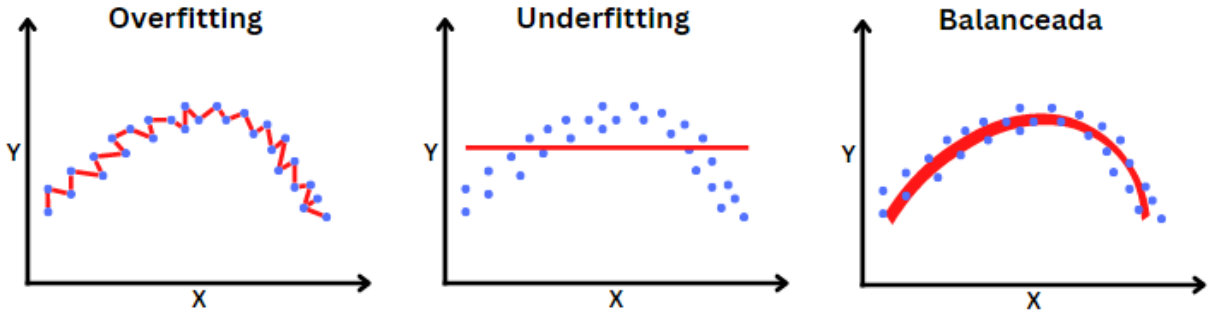
> O Aprendizado de Máquina Supervisionado (SML) é a busca por algoritmos que raciocinam a partir de instâncias fornecidas externamente para produzir hipóteses gerais, que então fazem previsões sobre instâncias futuras (p. 128).

Nesse ramo da aprendizagem de máquina existe uma função mapeadora, a qual é conhecida como hipótese, que buscará aproximar os dados de entrada com os dados de saída. Conforme apresentado por Silva (2020), essa aprendizagem tem como finalidade a escolha de um modelo que estabelece o domínio da hipótese e uma técnica de otimização que seja capaz de encontrar a hipótese mais apurada nesse domínio. Sobre o processo de otimização, o autor coloca que o mesmo consiste na minimização da função de perda durante o treinamento, que

levará também a redução da taxa de erro - que representa o quão longe a hipótese está da solução ideal (GOODFELLOW, BENGIO, COURVILLE, 2016 apud SILVA, 2020).

Silva (2020) ainda afirma que se deve ater ao fato de que baixas taxas de erro não significam obrigatoriamente que a hipótese será apurada para casos de valores desconhecidos - o que se constitui como um dos principais desafios em Machine Learning. Esse fato pode levar a problemas de *underfitting* e *overfitting*, conforme Figura 5, em que o primeiro ocorre quando a solução é muito simples para um problema complexo, gerando uma hipótese muito genérica, e o segundo quando a solução é muito complexa para um problema simples, gerando uma hipótese muito específica (GOODFELLOW, BENGIO, COURVILLE, 2016 apud SILVA, 2020).

Figura 5 - Exemplificação de Overfitting e Underfitting.



Fonte: Autor (2022).

Existem diversos algoritmos de Machine Learning e abordagens, em que cada um deles apresentará melhor resultado em determinadas situações - de forma com que conforme o problema, determinados algoritmos terão melhor eficiência e resultados. O autor Mahesh (2020) explicita que os Sistemas de Aprendizado Supervisionados podem ser classificados em: de classificação, de regressão e de previsão.

Aqueles enquadrados na categoria de classificação são eficientes em modelos relacionados a tarefas de classificação, em que o algoritmo se torna capaz de obter conclusões a partir dos valores observados e apontar a qual categoria os novos dados observados se enquadram. Os algoritmos enquadrados como de regressão são capazes de estimar e compreender as relações existentes entre as variáveis de entrada e saída. E por fim, os de previsão são responsáveis por prever dados futuros a partir daqueles já existentes.

Entre os métodos de aprendizado supervisionado estão os que serão utilizados na aplicação deste trabalho: Regressão Linear Ridge, Árvore de Decisão e XGBoost.

2.3.1.1 Regressão Linear Ridge

Conforme apresentado por Gonfalonieri (2019), um dos algoritmos de Machine Learning mais utilizados é o de Regressão Linear, que possui uma implementação rápida e simples, além de possuir saída de fácil interpretação. A mesma é denominada dessa maneira por consistir-se em uma reta traçada a partir de uma relação em um diagrama de dispersão.

A Regressão Linear é um algoritmo supervisionado de Machine Learning que possibilita a estimativa do valor de algo baseando-se em uma série de outros dados históricos. Existem 2 tipos de Regressão Linear: a simples e a múltipla.

A Regressão Linear Simples refere-se ao algoritmo que possui somente uma variável independente para a predição. Já a Regressão Linear Múltipla é quando há mais de uma variável independente para a predição - que será utilizada no presente trabalho.

De forma geral, na Regressão Linear Múltipla, a partir de um conjunto de variáveis de entrada verifica-se como a mesma possibilita explicar a variável de saída. O autor Pascual (2018) apresenta que tal modelo estabelece uma relação linear entre as variáveis de entrada e as de saída, possuindo como objetivo encontrar uma reta que possibilite uma boa definição dos dados - de forma a minimizar a diferença entre os dados de testes (entrada) e a saída calculada pelo modelo.

Matematicamente a relação linear pode ser representada da seguinte maneira (1):

$$Y_1 = \alpha + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \ \ldots + E_n \qquad (1)$$

Em tal equação: Y_1 é a variável de saída; α é o valor constante do modelo; X são os valores das variáveis analisadas; β são os coeficientes do modelo e E apresenta os resíduos do modelo.

A diferença entre o valor real e a estimativa gerada pelo algoritmo é chamada de resíduo, que pode ser calculada em cada ponto do conjunto de dados, servindo de um bom parâmetro para avaliação do desempenho do modelo. Caso o conjunto de resíduos seja pequeno, significa que o modelo possibilita uma boa previsão de saída, sendo resíduos grandes, o modelo não atingiu sucesso.

Como apresentado por Matos (2021), durante a aplicação do algoritmo de Regressão Linear, o conjunto de dados utilizados como variáveis de entrada podem variar bastante entre si, o que gera a necessidade de normalização dos mesmos de forma antecipada - o que possibilitará um melhor desempenho do modelo.

A Regressão Linear Ridge, que é um modelo de Regressão Linear Múltipla, apresenta uma técnica de regulação que favorece a aplicação em dados altamente correlacionados (CHIANG et al., 2018). A mesma possui como característica uma suavização dos atributos correlacionados, convergindo para uma solução mais estável e restringindo o modelo a fim de evitar o Overfitting (DA SILVA, 2017). A regulação dos dados em tal algoritmo é feita através de um parâmetro denominado alpha, que pode variar entre 0 e infinito, que quanto maior, mais suavizados serão os atributos.

2.3.1.2 Árvore de decisão

Conforme os autores Norvig e Russell (2013) apresentam, o modelo de Árvore de Decisão representa uma função que obtém como entrada um conjunto de valores e retorna apenas decisão, ou seja, um único valor de saída - que é resultado de uma sequência de testes. Tais autores colocam que um diferencial do presente modelo é o fato de que é possível um ser humano compreender a razão da saída do algoritmo de aprendizagem, visto a lógica seguida.

O algoritmo de Árvore de Decisão pode se enquadrar tanto na categoria de classificação quanto de regressão, a qual "divide o problema em subproblemas menores que podem então ser resolvidos de forma recursiva" (NORVIG, RUSSELL, 2013, p. 814), em que a árvore é composta por galhos e nós: nós de divisão e nós folha.

Os nós da árvore representam um evento ou escolha, correspondendo aos testes dos valores de entrada, as arestas do gráfico representam as regras ou condições, que são seguidas de novas ramificações de nós - valores resultantes da entrada submetida às condições.

Conforme apresenta Carvalho et al. (2011), os nós de divisão possuem dois ou mais sucessores, possuindo um teste condicional nos valores de um atributo para realizar a divisão - indicando qual o próximo nó conforme o valor. Tratando dos nós folha, os mesmos ficam nas pontas das árvores e são como rótulos, com os valores das classes do conjunto de dados.

É possível dizer que cada nó da árvore contém uma questão, que poderá ser respondida como verdadeira ou falsa. A partir da resposta o algoritmo segue para uma direção, que caso

atinja um nó intermediário, o processo de decisão se repete. No momento em que o algoritmo atinge um nó folha da árvore, indicará o valor final.

2.3.1.3 XGBoost

O modelo *eXtreme Gradient Boosting* (XGBoost) trabalha a partir de algoritmos de Árvores de Decisão, impulsionando os mesmos a partir da aplicação de um modelo linear eficiente - possibilitando que sua execução seja mais rápida que demais soluções populares. Dessa maneira, conforme apresentado por Dhaliwal et al. (2018), o XGBoost busca otimizar os resultados obtidos a partir do treinamento de Árvore de Decisões, vários modelos fracos são combinados para gerar um modelo forte - fazendo com que elas se auto regulem. O presente modelo está sendo amplamente explorado e utilizado, visto sua eficiência na obtenção de resultados, além de ser possível de aplicar em diversos âmbitos como de classificação e regressão de dados - sendo capaz de resolvê-los com uma quantidade mínima de recursos.

O XGBoost possui inúmeros outros diferenciais, a qual também deve-se evidenciar sua regularização, visto que a mesma evita o overfitting de dados (DHALIWAL et al. 2018). Para além disso, o mesmo é um classificador flexível, possibilitando que o usuário defina os parâmetros do modelo, importantes para que se possa atingir os resultados desejados.

Como posto acima, a seleção dos parâmetros é um fator decisivo que afeta diretamente o desempenho do modelo. Os parâmetros do XGBoost podem ser classificados em três tipos: parâmetros gerais, parâmetros de reforço e parâmetros de tarefa. A partir do site oficial da biblioteca XGBoost serão abordados a seguir alguns dos mais utilizados, e que serão utilizados no presente trabalho:

O parâmetro "Booster" define o tipo de booster que o modelo irá utilizar, podendo ser ‘gbtree’, ‘gblinear’ ou ‘dart’, em que os dois primeiros são baseados em árvore e o último em funções lineares. Para algoritmos de regressão, as três opções podem ser utilizadas.

O parâmetro “Eta” é responsável por controlar a taxa de aprendizado, podendo ser ajustado para evitar o overfitting. Após cada etapa o modelo reduz os pesos dos recursos buscando atingir melhores resultados. Quanto menor sua taxa, mais lento será o processamento, visto que aumentará a exigência quanto aos resultados. Seu valor padrão é [0,3].

O parâmetro “Gamma” realiza o controle da regularização, sendo responsável pelo parâmetro de poda (remoção de sub-nós de um nó de decisão) ou adição de novos nós, que

também previne o overfitting. Quanto maior o valor de tal parâmetro, maior a regularização do modelo. Seu valor padrão é [0].

O parâmetro “Max_depth” define a profundidade que a árvore terá, um bom acerto deste parâmetro levará a uma melhora significativa do desempenho computacional, visto que quanto maior a profundidade, mais complexo o modelo - se tornando mais propenso a overfitting. Seu valor padrão é [6].

Por fim, o parâmetro “Colsample_bytree” é responsável por determinar a fração de recursos (variáveis) que serão utilizadas para a construção de cada árvore. A biblioteca não apresenta um valor padrão para tal parâmetro. Tais parâmetros bases sugeridos pela biblioteca servem de ponto de partida para o início da etapa de treinamento do algoritmo, que conforme sua resposta e eficiência, poderão ser ajustados - até que se alcance melhores resultados.

2.3.2 Métricas de Avaliação

Com o objetivo de avaliar a precisão e desempenho dos modelos, utiliza-se métricas que são capazes de sintetizar os resultados em um único valor - o qual representa a capacidade preditiva dos algoritmos. Existem diversas métricas de avaliação, em que cada uma delas possui determinadas especificidades. Entre elas destaca-se: Erro Médio Absoluto, Erro Médio Quadrado e Coeficiente de Determinação (R^2).

O Erro Médio Absoluto (MAE) realiza a média dos erros absolutos entre os valores de teste (y_i) e os valores de treino ($\hat{y}_i$) em n observações (2):

$$MAE = \frac{\sum_1^{i=n} |\hat{y}_i - y_i|}{n} \quad (2)$$

Tal métrica possui como um de seus pontos positivos a fácil interpretação do erro, visto que a unidade de medida avaliada não é alterada.

A métrica Erro Médio Quadrado (MSE) eleva ao quadrado o valor residual do MAE ($\hat{y}_i - y_i$), apresentando o desvio-padrão médio dos valores gerados pelos modelos em relação aos valores de treino (3):

$$MSE = \frac{\sum_1^{i=n} (\hat{y}_i - y_i)^2}{n} \quad (3)$$

A MSE se constitui como uma maneira mais rígida de avaliação do erro por tratá-lo na forma quadrática.

Como outra métrica amplamente difundida está o R², que avalia a dispersão dos pontos de dados em uma linha de regressão. Quanto maior o valor de R², menor é a diferença entre os dados reais e os dados previstos, onde $(\hat{y}_i)$ representa os valores do conjunto de dados de treino, (y_i) os valores previstos pelo modelo e (Ỹ) a média dos valores do conjunto de treino. O mesmo é descrito pela fórmula a seguir (4):

$$R^2 = 1 - \frac{\sum_1^{i=n} (\hat{y}_i - y_i)^2}{\sum_1^{i=n} (\hat{y}_i - \tilde{Y})^2} \quad (4)$$

Tal métrica é amplamente utilizada em diversos estudos (PASION, 2020; TANEV, STANEV, 2021; URZAGASTI et al., 2021), apresentando-se como de extrema relevância no que tange a avaliação de modelos preditivos.

2.4 SÍNTESE DE PESQUISAS

Ao longo dos anos diversos trabalhos vêm sendo construídos com o objetivo de construir e analisar modelos de Machine Learning aplicados à predição de energia solar fotovoltaica, o qual no geral apresentam grande eficiência. São utilizados diversos algoritmos, como o caso de Redes Neurais, Árvore de Decisão e XGBoost.

O artigo do qual foi retirado o dataset para a realização do presente trabalho (PASION et al., 2020) se propôs analisar diversas técnicas de Machine Learning para a prever a saída de energia para painéis solares a partir de variáveis independentes que não continham dados de irradiação solar. Desenvolveu-se o estudo através de dados coletados em durante 14 meses em 12 locais do hemisfério norte. Como métrica de avaliação utilizou-se Raiz quadrada do erro-médio (RMSE), MAE, e R². Para RMSE os valores variaram entre 1,754 e 5,027, com um valor médio de 3,174. Tratando de MAE, os valores variaram menos, entre 1,176 e 3,896, com um valor médio de 2,280. Por fim, os valores de R² apresentaram variações entre 0,501 e 0,939, com valor médio de 0,771.

Tanev e Stanev (2021) desenvolveram um estudo sobre a geração de energia de uma usina solar fotovoltaica usando vários métodos de Machine Learning e diferentes variáveis independentes, combinando os mesmos em busca dos melhores resultados. Como algoritmos utilizados estão: Regressão Linear, Regressão Linear Poly Features, Regressão Linear Lasso, Regressão Linear Ridge, Árvore de Decisão, Random Forest. Para avaliar os resultados obtidos utilizou-se as métricas R^2 e RMSE, em que na primeira os valores variaram entre 0,988247 e 0,993671.

Urzagasti et al. (2021) realizaram um trabalho de análise e comparação de acurácia de predição de modelos de RNA a partir de dois datasets diferentes, submetidos a três diferentes horizontes de predição de curto prazo - posteriormente submetidas às métricas R^2 e RMSE Para R^2, o primeiro dataset apresentou valores médios maiores ou iguais a 0,9, já o segundo menores ou iguais a 0,80. Como resultados finais obteve-se o fato de que a acurácia dos modelos é diretamente influenciada pelo uso de diferentes variáveis meteorológicas e o tamanho da base de dados. Além disso, foi possível concluir que quanto maior o horizonte de predição, menor a assertividade do modelo.

Gupta et al. (2021) desenvolveram um trabalho utilizando o algoritmo de Árvore de Decisão a partir de um conjunto de dados de valores diários de energia solar gerados durante cinco anos por uma usina de energia solar de 10MW da Gujarat Power Corporation Limited, na Índia. Tal algoritmo foi submetido às métricas de avaliação MSE e MAE. Observa-se que tal algoritmo apresenta resultados muito positivos, com alta precisão, baixos erros e fácil interpretação - indicando ser importante ferramenta para previsão de geração de energia fotovoltaica.

Da Silva (2022) realizou um estudo aplicando Machine Learning para a prever a geração de uma usina solar fotovoltaica de 160 MW de potência instalada, no Ceará. Seu estudo abrangeu treze diferentes algoritmos, estando estes baseados em RNAs e XGBoost. Como métricas para avaliação de tais aplicações foram utilizadas as métricas RMSE e MAE, em que o modelo XGBoost apresentou melhor resultado - com RMSE de 24,68 e MSE de 17,94.

Os autores Bergamo e Ramos (2022) desenvolveram um trabalho com o objetivo de estudar o comportamento da geração fotovoltaica de uma usina solar utilizando, entre outras ferramentas, o Machine Learning. Entre os algoritmos utilizados para tal estão: Random Forest, Support Vector Machine, XGBoost e MLP. Como métricas de avaliação foram utilizadas MAE, RMSE e MAPE. Após a realização de todas as etapas do estudo pode-se observar que todos os

modelos obtiveram alta assertividade, em que o XGBoost foi o segundo melhor com assertividade de 96.934%, apresentando MAE de 0,8264.

A seguir, tabela com síntese das informações basilares dos trabalhos referidos acima.

Tabela 1 - Síntese dos trabalhos referidos

Autores	**Proposta**	**Modelos Utilizados**	**Variáveis**	**Métricas Utilizadas**
Pasion (2020)	Utilização de diversos algoritmos de Machine Learning para prever geração de energia solar fotovoltaica sem o dado de irradiação solar	Rede neural de feedforward multicamadas (MFFNN), Máquina de aumento de gradiente (GBM), Modelos combinados e/ou com validação cruzada, Modelagem linear generalizada (GLM), Random Forest distribuída (DRF) e Random Forest distribuída extremamente aleatória (XRT)	Local, data, horário, latitude, longitude, altitude, mês, hora, umidade, temperatura ambiente, potência de saída, velocidade do vento, visibilidade, pressão e teto de nuvens.	R^2, MAE e RMSE
Tanev e Stanev (2021)	Previsão de geração de potência e energia de uma usina fotovoltaica usando vários métodos de Machine Learning, comprovando que o uso de dados de irradiação solar e de temperatura ambiente são suficiente para tal	Regressão Linear, Regressão Linear Polynomial Features, Regressão Linear Lasso, Regressão Linear Ridge, Árvore de Decisão, Random Forest	Irradiação solar, temperatura ambiente, módulo temperatura e velocidade do vento	R^2 e RMSE
Urzagasti et al. (2021)	Comparação de acurácia de predição de modelos de RNA já publicado anteriormente, a partir de dois datasets distintos e três diferentes horizontes de predição de curto prazo	RNA	Temperatura do ar, ângulo azimutal, ângulo zenital, mês, hora e minuto, irradiação solar.	R^2 e RMSE
Gupta et al. (2021)	Aplicação do modelo de Árvore de Decisão para prever a geração de energia de uma usina solar fotovoltaica, buscando relacionar os atributos climáticos com a potência de saída gerada.	Árvore de Decisão	Velocidade média do vento, temperatura máxima do dia, temperatura mínima do dia, precipitação do dia, disponibilidade da placa,	MSE e MAE

			disponibilidade do equipamento, irradiação solar e saída da placa (KWh).	
Da Silva (2022)	Aplicação de Machine Learning para a previsão da geração de uma usina solar fotovoltaica de 160 MW de potência instalada para para um intervalo de um ano a partir dos dados utilizados para treino	RNA, XGBoost e híbridos	Dados de geração e , temperatura ambiente mínima e máxima, precipitação irradiação média, umidade relativa do ar e velocidade do vento.	RMSE e MAE
Bergamo e Ramos (2022)	Estudo do comportamento da geração fotovoltaica de uma usina solar, de forma com que seja possível subsidiar estudos em outras usinas similares. Como forma de validação realizou-se um estudo de previsibilidade da geração da energia elétrica.	XGBoost, SVR, MLP e Random Forest.	Precipitação, temperatura, vento, radiação global, geração elétrica, etc.	MAE, RMSE

Fonte: Autor, 2022.

Existem diversos outros trabalhos que abordam a temática, sendo mais comuns aqueles que utilizam os algoritmos de RNA (ROCHA et al., 2019; URZAGASTI, 2021; YADAV et al., 2014). Todos eles confirmam a hipótese de que Machine Learning se constitui como uma ferramenta muito eficiente para a predição de geração de energia fotovoltaica, mesmo considerando os desafios ainda existentes e enfrentados.

3 METODOLOGIA

Nesta seção será apresentada a metodologia aplicada, inicialmente serão tratados do tipo de pesquisa adotado e posteriormente das etapas a serem seguidas. O presente estudo enquadra-se como de natureza aplicada, possuindo abordagem quantitativa, objetivo exploratório e método *Design Science Research*.

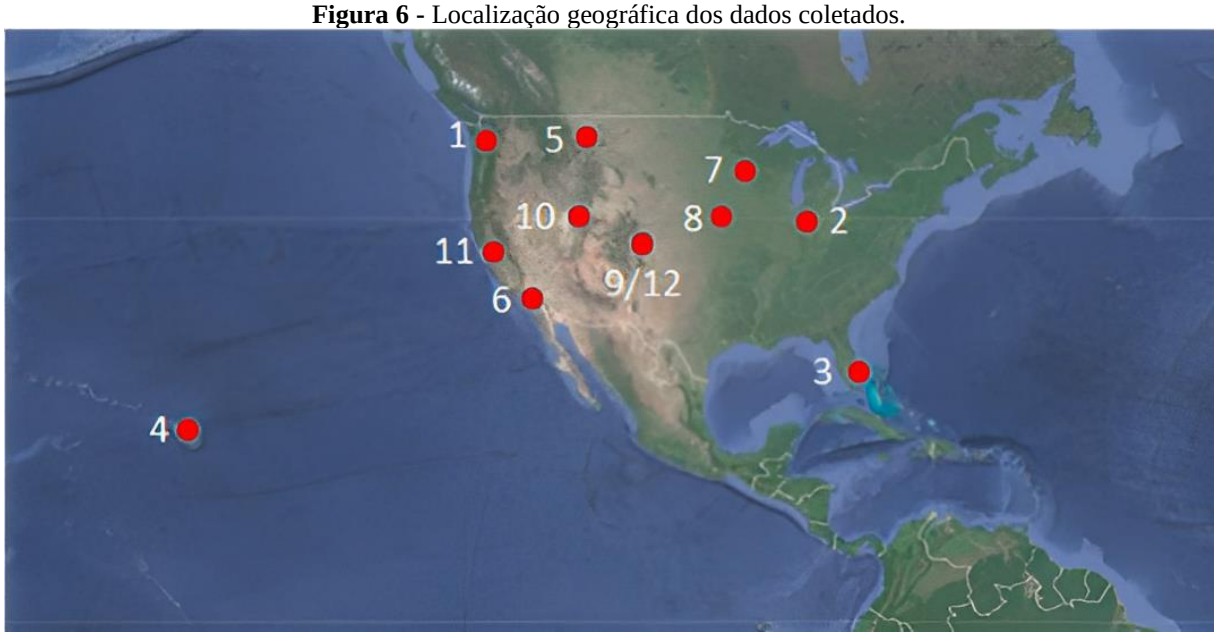
A utilização da ferramenta de aprendizado de máquina evidencia a natureza aplicada do presente trabalho visto que a mesma será utilizada para solucionar um problema específico, de abordagem quantitativa.

O método de *Design Science Research* se caracteriza como um rigoroso processo que visa projetar artefatos para a resolução de problemas, avaliação do que foi projetado ou do que está funcionando e para comunicação dos resultados obtidos. Na presente pesquisa o artefato em questão é o método que se constitui como um conjunto de passos, que pode ser um algoritmo, usado para executar uma tarefa (LACERDA et al., 2013).

3.1 DADOS DA PESQUISA

O conjunto de dados que serão utilizados na pesquisa é público e está disponibilizado na plataforma Kaggle, que é uma comunidade da área de ciência de dados. Este conjunto de dados foi utilizado no artigo *Machine Learning Modeling of Horizontal Photovoltaics Using Weather and Location Data* (PASION et al., 2020) publicado no jornal *Energies*. Este dataset contém as saídas de energia de painéis fotovoltaicos horizontais distribuídos em 12 instalações globais da Força Aérea Americana pelo hemisfério norte, sua localização geográfica está ilustrada na Figura 6 e os locais estão identificados na Tabela 1, ao longo de 14 meses.

Figura 6 - Localização geográfica dos dados coletados.



Fonte: Pasion et al. (2020).

Tabela 2 - Localização geográfica dos dados coletados.

Número no Mapa	Local	Identificação nos Dados	Estado	Latitude (Graus)	Longitude (Graus)
1	Camp Murray	Camp Murray	Washington	47.11°	122.57°
2	Grissom	Grissom	Indiana	40.67°	86.15°
3	Jonathan Dickinson State Park	JDMT	Florida	26.98°	80.11°
4	Kahului	Kahului	Hawaii	20.89°	156.44°
5	Malmstrom	Malmstrom	Montana	47.52°	111.18°
6	March	March	California	33.9°	117.26°
7	Aeroporto Internacional de Minneapolis-Saint Paul	MNANG	Minnesota	44.89°	93.2°
8	Offutt	Offutt	Nebraska	41.13°	95.75°
9	Peterson	Peterson	Colorado	38.82°	104.71°

10	Hill Weber	Hill Weber	Utah	41.15°	111.99°
11	Travis	Travis	California	38.16°	121.56°
12	Academia da Força Aérea EUA	USAFA	Colorado	38.95°	104.83°

Fonte: Pasion et al. (2020).

Os dados foram coletados de painéis solares Renogy® policristalinos, 50 Watts e 12 Volts integrados ao sistema computacional Raspberry Pi® que foi utilizado para registrar as informações de potência do painel, temperatura, umidade, data e hora, a cada 15 minutos. Solicitou-se que nos locais onde se encontravam os painéis que houvesse limpeza frequente sempre que observado coberturas de poeira ou neve, já que isso afeta diretamente a produção de energia, porém não há dados da frequência com que esse serviço foi realizado.

As variáveis independentes contidas no conjunto de dados incluem: localização (latitude e longitude), data, hora amostrada, altitude, estação, umidade, temperatura ambiente, potência do painel solar, velocidade do vento, visibilidade, pressão e teto de nuvens, suas variáveis e unidades estão demonstradas na Tabela 2.

O conjunto de dados está em formato .csv e possui 21045 amostras para cada variável independente que estão distribuídas em 17 colunas. Porém há uma quantidade de amostras diferentes para cada localidade, que deve ser levado em consideração no momento da análise comparativa entre as localidades.

Tabela 3 - Variáveis e suas unidades.

Variável	Unidade
Potência de Saída	Watts
Latitude	Graus
Humidade	Percentual
Temperatura Ambiente	Graus Celsius
Velocidade do Vento	km/h
Visibilidade	km
Pressão	Milibares
Teto de Nuvens	km

Altitude	m

Fonte: Pasion et al. (2020).

3.2 FERRAMENTAS

Para o desenvolvimento dos algoritmos de Machine Learning é necessária a utilização de alguma linguagem de programação. Entre as diversas existentes, Python se destaca sendo amplamente utilizada, de alto nível, fácil elaboração e interpretação (BORGES, 2014). A linguagem Python é gratuita, podendo ser utilizada livremente em praticamente qualquer sistema operacional de computadores sem necessidade de adquirir licenças.

Conforme Diaz (2022), Python é uma linguagem de Programação Orientada a Objetos (OOP), que se constitui como um paradigma (teoria para solucionar problemas) de programação que permite problemas complexos serem tratados como objetos. Dessa maneira, de forma sucinta, OOPs são conjuntos de conceitos e padrões utilizados para a resolução de problemas como objetos. Na linguagem Python, como em demais OOPs, o objeto é uma única coleção de dados (atributos) e comportamento (métodos) - de forma com que se realiza a construção de objetos para o armazenamento de dados, em que os mesmos contêm funcionalidades específicas.

Devido sua flexibilidade, a mesma vem possibilitando e agregando o processo de expansão da utilização de inteligência artificial em diversos âmbitos visto que embora a mesma seja simples, é poderosa e pode ser utilizada em grandes projetos. A mesma permite que grandes resultados sejam obtidos de forma rápida (MENEZES, 2010).

Para que seja possível utilizar o Python, é necessário o uso de um interpretador, software que interpreta os comandos escritos através da linguagem e os executa - transformando o ‘texto’ em sistemas propriamente (MENEZES, 2010). O interpretador também é responsável pela verificação dos códigos, indicando caso exista algum erro. Anteriormente os interpretadores se tratavam de programas instalados diretamente na máquina, mas atualmente os mesmos já existem em nuvem, como o caso do Google Colab - será abordado posteriormente.

Para além disso, a grande relevância e utilização do Python também se deve ao fato de que a mesma possui uma comunidade ativa e uma vasta gama de bibliotecas, que são conjuntos de códigos Python associados que implementam recursos e simplificam a aplicação, tornando-a ainda mais rápida.

Como algumas relevantes bibliotecas: Numpy, Pandas, MatplitLib, Sklearning e XGBoost.

NumPy é uma biblioteca matemática de código aberto que se constitui como uma das mais utilizadas de sua categoria. A mesma possui tanto recursos matemáticos quanto para análise de dados, sendo destinada a realização de operações multidimensionais. Ela é amplamente utilizada no campo da ciência de dados, visto que permite uma grande facilidade de execução de cálculos numéricos, além de eficientes armazenamento e processamento dos dados.

Dessa maneira, NumPy possui diversas funções matemáticas para operações facilitadas além de operações eficientes para tratamento e limpeza de dados, geração de subconjuntos e filtragens, estatísticas descritivas, manipulação de dados relacionais, manipulação de dados em grupos, etc. (MULINARI, 2022).

A biblioteca MatplotLib também se constitui como uma biblioteca matemática de código aberto, porém mais utilizada para a criação de apresentações de dados em forma gráfica. Ela possui sua impressão baseada em pontos coordenados, oferecendo ferramentas que são capazes de plotar gráficos de forma bi e tridimensional (HOMEM, 2020).

A biblioteca Pandas é a principal ferramenta em Python destinada a análise de dados, a qual possibilita a manipulação de grandes conjuntos de dados e o tratamento dos mesmos de forma ágil (MCKINNEY, 2017). Ela se constitui como uma das bibliotecas mais populares, possuindo alto desempenho e eficiência, introduzindo ferramentas que facilitam as tarefas analíticas.

Pandas é construído a partir das bibliotecas MatplotLib, de visualização de dados, e NumPy, de operações matemáticas. Dessa maneira, por ser a união de tais bibliotecas, Pandas permite o acesso a métodos de ambas bibliotecas a partir de um número menor de códigos (COUTINHO, 2021).

Já a biblioteca Scikit-Learn, mais conhecida como SkLearn, também é de código aberto, porém possui funções específicas para aplicação prática de Machine Learning - tanto de aprendizado supervisionado quanto não-supervisionado. SkLearn está organizada em diversos módulos, em cada um deles possui ferramentas e funções específicas, como: ferramentas para ajuste de modelos, pré-processamento de dados, seleção de modelos, avaliação de modelos, entre outras (PEDREGOSA *et al.*, 2011).

A biblioteca em questão possui diversos algoritmos e modelos de Machine Learning integrados, como o caso dos modelos de Regressão Linear e Árvore de Decisão já citados anteriormente, os quais podem ter seus parâmetros ajustados conforme necessidade. Como diferenciais de tal biblioteca aponta-se seu conjunto de ferramentas, como a que possibilita que os melhores parâmetros sejam encontrados de forma automática, e as que facilitam o processo de avaliação dos modelos.

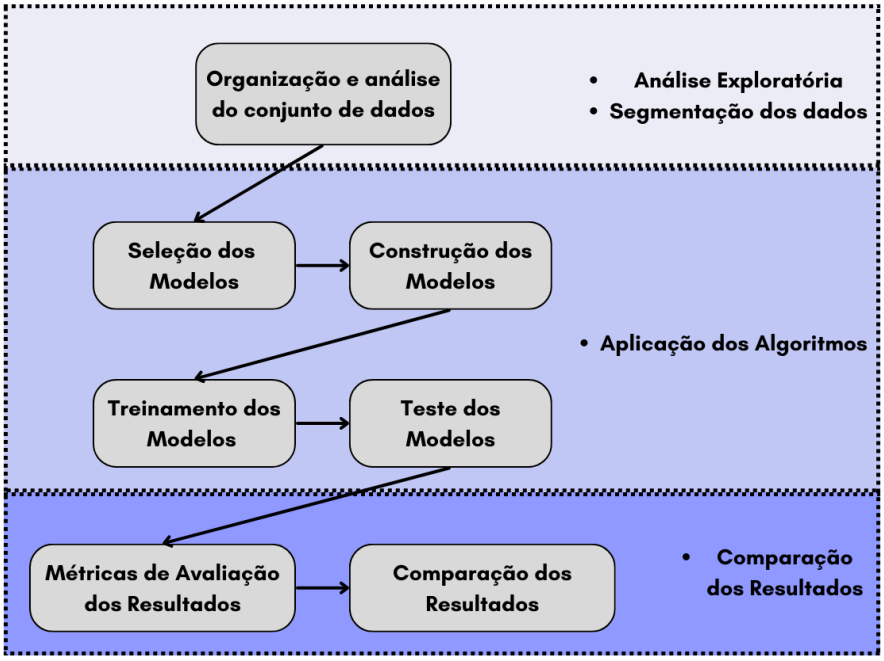
Por fim, a biblioteca XGBoost é de código aberto e possui como objetivo ser eficiente e flexível, sendo responsável pela implementação de algoritmos de estrutura Gradient Boosting (DEVELOPERS, 2021) - como o caso do modelo XGBoost, já desenvolvido anteriormente. Ela permite o ajuste de determinados parâmetros do algoritmo e possui diversas ferramentas, como as que possibilitam a avaliação da aplicação.

Como já desenvolvido anteriormente, para que seja possível utilizar o Python é necessário o uso de um interpretador, como o Google Colaboratory, mais conhecido como Google Colab, que é uma plataforma virtual gratuita que fornece o ambiente Jupyter Notebook. O mesmo possibilita a programação em linguagem Python diretamente do navegador, sem qualquer tipo de instalação ou configuração na máquina, desprezando a necessidade de equipamentos potentes (COLABORATORY, 2022). De forma padrão, o Google Colab já possui as bibliotecas citadas anteriormente instaladas, precisando apenas serem importadas para possibilitar seu uso.

3.3 ETAPAS METODOLÓGICAS

Para a realização do estudo foram utilizadas as seguintes etapas: organização do conjunto de dados, seleção dos modelos, aplicação dos modelos, treinamento dos modelos, definição de uma métrica de comparação e a comparação dos resultados, todas as etapas estão ilustradas na Figura 7. Em todas as etapas mencionadas foram utilizadas a linguagem de programação Python.

Figura 7 - Fluxograma das Etapas.



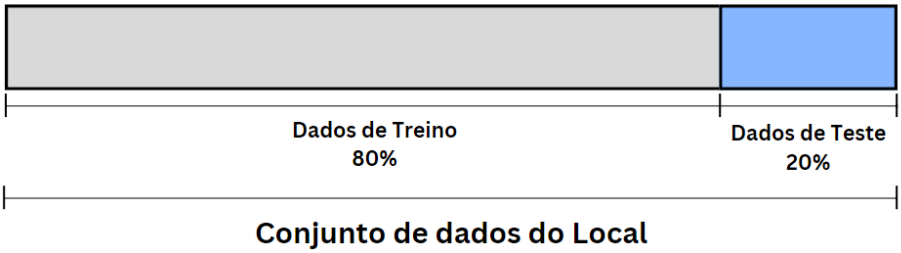
Fonte: Autor, 2022.

3.3.1 Organização e análise do conjunto de dados

Na etapa da organização e análise do conjunto de dados foram separados e segmentados os dados coletados, avaliando quais tipos de dados possuiriam relevância no estudo e de que forma eles teriam utilidade no desenvolvimento do estudo em questão. Dessa maneira, esta etapa buscou alcançar o objetivo específico de classificação dos dados referentes às usinas analisadas.

Da base de dados utilizada foi retirada apenas uma coluna que apresentava dados reincidentes que estavam contidos em outras colunas, os quais se referiam a variáveis de data e horário. Posteriormente a esta ação foram divididos os dados por local, ficando assim doze conjuntos de dados. Cada um desses doze conjuntos foram divididos em dados de treino (80%) e dados de teste (20%), sem aleatoriedade. Ou seja, os 80% primeiros dados do conjunto foram selecionados para treino e os 20% restantes para dados de teste, conforme ilustrado na Figura 8.

Figura 8 - Amostra de treino e teste.



Fonte: Autor (2022).

3.3.2 Seleção dos Modelos

Após a delimitação dos dados a serem utilizados se iniciou a seleção dos modelos mais adequados para o alcance do objetivo geral da pesquisa. Tal seleção se deu a partir de modelos de Machine Learning previamente existentes na literatura e contemplou o segundo objetivo específico do estudo. Os modelos selecionados levaram em consideração modelos que já estavam consolidados pela literatura e que não necessitavam de grande capacidade de processamento.

3.3.3 Construção dos modelos

Realizadas as etapas anteriores se passou para a de aplicação dos dados nos três modelos selecionados. Para isto foram utilizadas bibliotecas amplamente conhecidas em Python aplicadas diretamente no Google Colab, a fim de construir algoritmos para aplicação dos três modelos nas doze localidades separadamente.

Como já citado no decorrer do trabalho, um dos princípios de Machine Learning é a necessidade de treinamento da máquina, o qual foi realizado na etapa seguinte - gerando os resultados finais utilizados na posterior análise.

3.3.4 Treinamento dos modelos

Após a etapa de construção dos modelos foi possível aplicar os dados para treino, já separados anteriormente. Neste momento, foram aplicados os dados de treino de cada localidade em cada modelo separadamente, gerando assim doze máquinas para cada modelo.

Os modelos foram treinados com o objetivo de prever a potência de saída dos painéis solares com base nas variáveis independentes do conjunto de dados utilizados.

3.3.5 Teste dos modelos

Esta etapa objetivou testar a assertividade dos modelos, comparando os dados reais e os previstos pelos algoritmos. Para isto, foram confrontados os dados separados anteriormente como de teste (20% do conjunto total) com os dados de saída previstos pelos modelos.

Dessa maneira, as variáveis independentes do conjunto de teste (temperatura, umidade, etc.) de cada localidade foram submetidas aos modelos de forma a preverem variáveis dependentes, que são as potências de saída. A partir disso, realizou-se a comparação entre as variáveis dependentes previstas pelos modelos com as potências de saída reais contidas no conjunto teste.

Os dados previstos pelos modelos para cada localidade foram plotados em forma gráfica para comparação visual, possibilitando a análise do resultado real da potência de saída com os dados previstos pelas máquinas.

3.3.6 Métrica de avaliação dos resultados

Para tornar possível a realização da análise das previsões foi preciso definir métricas comparativas de acurácia. Desta forma, a etapa em questão visou a definição de métricas que possibilitasse o estabelecimento e identificação do modelo que se destaca para a resolução do problema central do estudo.

Foram definidas três métricas para comparação: MAE, MSE e R^2. Tais métricas foram selecionadas por já serem amplamente utilizadas na literatura e possuírem objetivos diferentes entre si. Com isso, tornou-se possível a comparação dos modelos através de tais métricas distintas.

Os resultados dos cálculos de erros foram baseados na comparação dos resultados previstos pela máquina com a saída real (dados de teste) da potência de saída das placas em cada localidade.

3.3.7 Comparação dos resultados

Após os resultados de erro (MAE, MSE e R^2) calculados em cada localidade para cada modelo, foram obtidos doze resultados para cada erro e cada modelo respectivamente. Para que fosse possível a comparação dos erros dos modelos foi calculada uma média simples entre eles, gerando assim um único valor para cada modelo referente a cada métrica. Desta maneira, a presente etapa buscou atingir o objetivo específico de comparação dos modelos treinados.

Por fim, após a realização das etapas anteriormente mencionadas, se iniciou a análise dos resultados obtidos no estudo. Nesta etapa se explicitou as conclusões, de forma com que se tornou possível a proposição um modelo de previsão para a geração de energia elétrica fotovoltaica a partir da observação dos dados de doze usinas.

4 RESULTADOS

Nesta sessão serão apresentadas a análise exploratória dos dados, bem como os resultados das aplicações de Machine Learning e o comparativo dos resultados obtidos a partir do conjunto de dados apresentado anteriormente. A partir disso, objetiva-se prever resultados de potência de saída de forma mais assertiva, utilizando diferentes modelos. O apêndice L contém o código desenvolvido.

Durante o desenvolvimento do estudo houve uma grande dificuldade no que tange a geração de *Overfitting* por parte dos modelos, associados a diversos fatores. Como primeiro fator podemos citar a pequena quantidade de dados, já que os mesmos foram separados em grupos por localidade, como segundo temos a alta variância dos dados e por fim, a tendência natural de *Overfitting* dos modelos de Árvore de Decisão (no caso deste estudo, o de Árvore de Decisão e o XGBoost).

Assim, as máquinas quando submetidas aos dados de treino acabavam não “aprendendo” e sim “decorando” entradas e saídas. Isso ficou evidente nas métricas calculadas e necessitou consecutivos ajustes para que fosse possível melhor adequar os modelos ao problema proposto.

4.1 ANÁLISE EXPLORATÓRIA

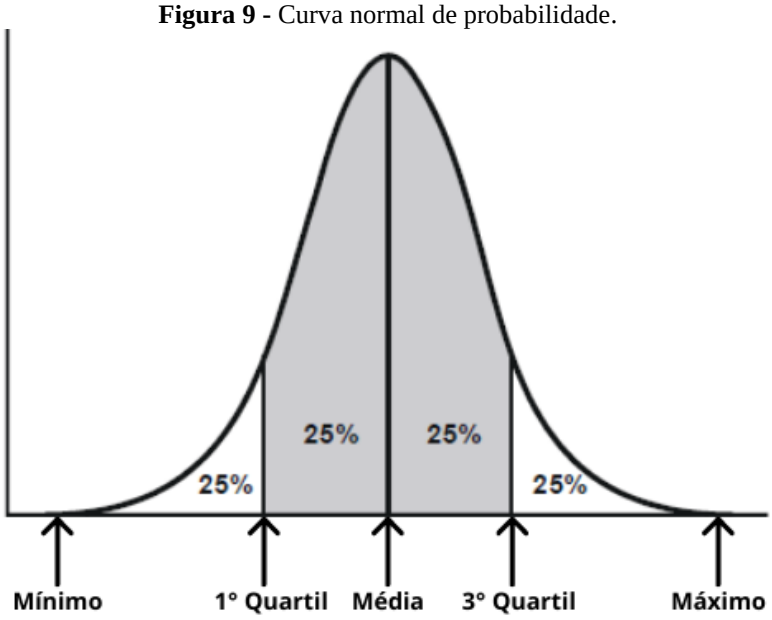
Para que se possa compreender a natureza de cada variável independente será realizada a análise exploratória dos dados. Foi retirada a coluna “YRMODAHRMI” que representa um compilado de ano, mês, hora e minuto. E também a coluna "Season" que se refere às estações do ano em que o dado foi obtido. Não serão mostrados na análise exploratória os dados das colunas “Date”, “Month”, “Hour”, “Time”, “Latitude” e “Longitude”, porém esses serão utilizados posteriormente na aplicação dos métodos.

As colunas “Date” e a “Time” foram transformadas posteriormente em uma única coluna que contém a informação detalhada de data, hora e minuto que aquela amostra foi coletada.

Os dados foram organizados conforme a curva normal de probabilidade para facilitar a visualização utilizando a biblioteca Pandas, na Figura 9 está apresentada a curva normal de

probabilidade e seus pontos principais. A Tabela 4 mostra as variáveis e suas unidades, bem como seus valores máximos, mínimos, médias, mediana, desvio padrão, 1º Quartil e 3º Quartil.

Figura 9 - Curva normal de probabilidade.



Fonte: Autor (2022).

Tabela 4 - Avaliação das variáveis independentes.

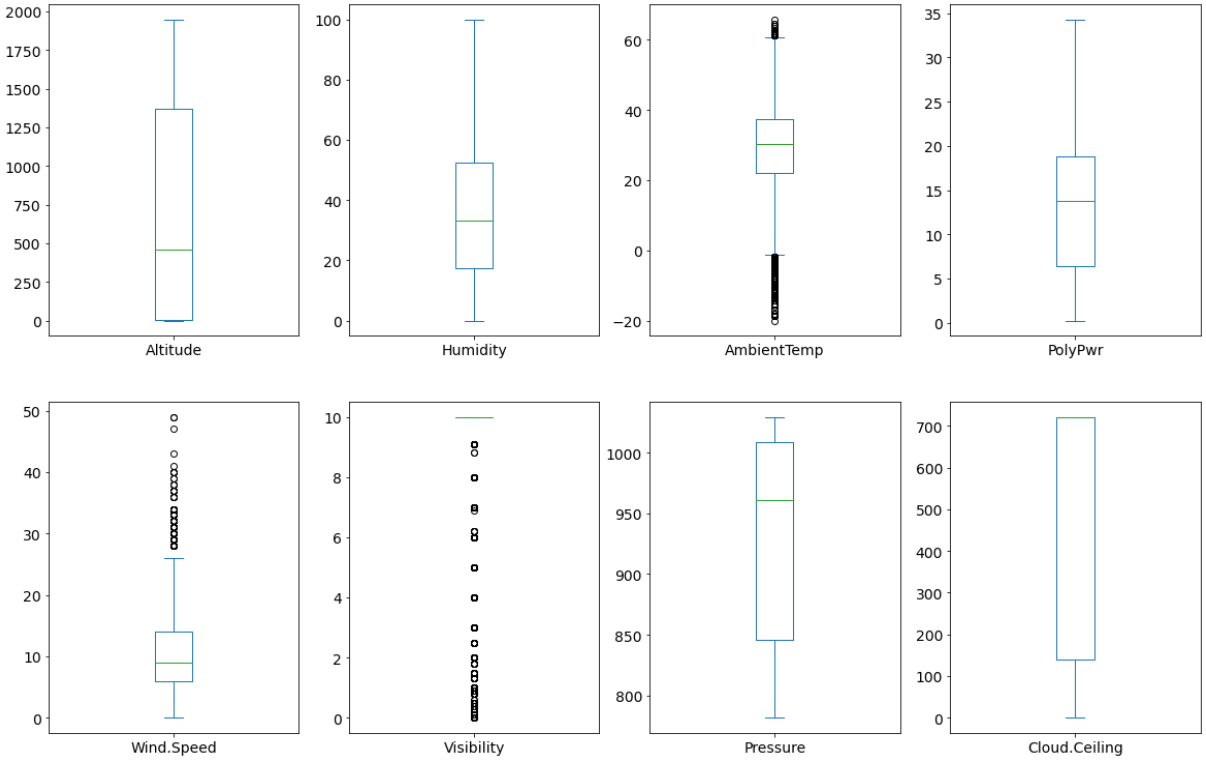
	Time	Latitude	Longitude	Altitude	Humidity	Ambient Temp	PolyPwr	Wind. Speed	Visibility	Pressure	Cloud. Ceiling
N° Amostras	21045	21045	21045	21045	21045	21045	21045	21045	21045	21045	21045
Média	1267,48	38,21	-108,59	798,84	37,12	29,28	12,97	10,31	9,70	925,94	515,96
Desvio Padrão	167,60	6,32	16,36	770,68	23,82	12,36	7,12	6,38	1,35	85,21	301,90
Mínimo	1000	20,89	-156,44	1	0	-19,98	0,25	0	0	781,70	0
1° Quartil	1100	38,16	-117,26	2	17,53	21,91	6,40	6	10	845,50	140
Mediana	1300	38,95	111,18	458	33,12	30,28	13,79	9	10	961,10	722
3° Quartil	1400	41,15	-104,71	1370	52,59	37,47	18,86	14	10	1008,90	722
Máximo	1545	47,52	-80,11	1947	99,98	65,73	34,28	49	10	1029,50	722

Fonte: Autor (2022).

Para que os dados da Tabela 4 fiquem mais claros foram gerados gráficos de caixas utilizando a biblioteca MatplotLib, Figura 10, de cada uma das variáveis independentes, nele é

possível identificar as informações sobre a distribuição dos dados além de identificar a existência de *Outliers*, que são aqueles atípicos que se diferem drasticamente dos outros pontos.

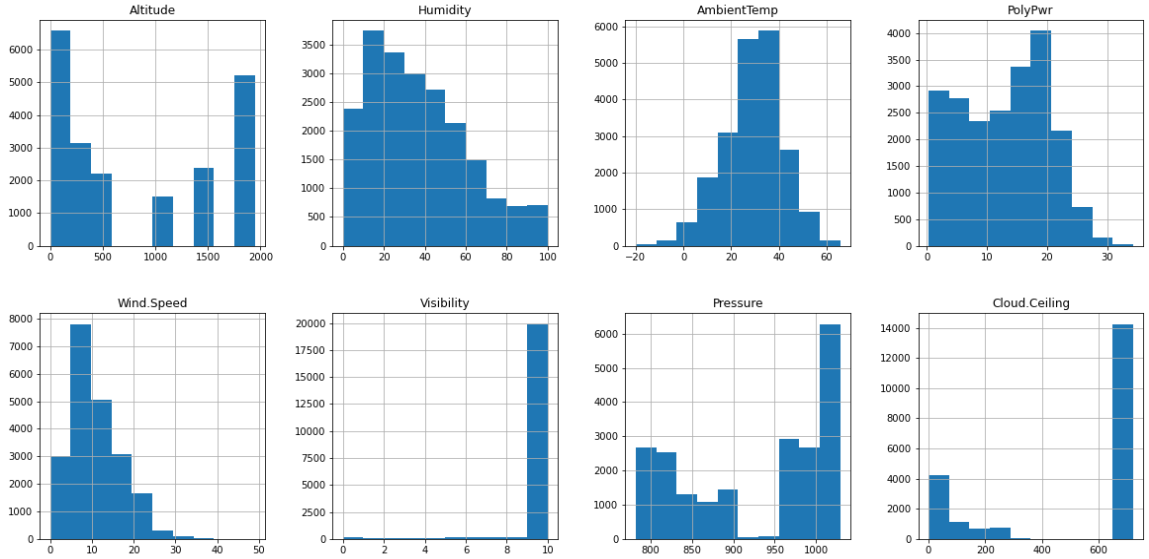
Figura 10 - Gráfico de Caixa das variáveis independentes.



Fonte: Autor (2022).

Outra forma de visualização dos gráficos é em forma de histograma como a curva de probabilidade, como mostrado na Figura 11, também gerada utilizando a biblioteca MatplotLib. Todas essas são formas de melhor compreender a natureza dos dados.

Figura 11 - Histograma das variáveis independentes.



Fonte: Autor (2022)

4.2 SEGMENTAÇÃO DOS DADOS

Primeiramente o conjunto de dados foi separado por localidade, assim foi obtido doze grupos de dados distintos e estes posteriormente foram divididos cada um em outras duas partes, dados de treino e dados de teste. Para obtenção dos dados de treino foram utilizados os 70% primeiros dados da amostra de cada local, ou seja, os dados não foram coletados de maneira aleatória no grupo do conjunto de dados. Com isso, os 30% dos dados restantes foram utilizados para testar os modelos, de maneira com que se tornasse possível a comparação entre os dados reais de saída com os dados previstos pelo algoritmo.

A Tabela 5 apresenta a quantidade de amostras para cada uma das localidades, já que as mesmas não possuem um número de amostras padrão entre si, bem como o número de amostras utilizadas para treino dos algoritmos e a parte utilizada para teste.

Tabela 5 - Tamanho das Amostras de cada local.

Local	Amostras por local	Treinamento (70%)	Teste (30%)
Camp Murray	1113	779	334
Grissom	1487	1041	446
Hill Weber	2384	1669	715
JDMT	1779	1245	534
Kahului	941	659	282
Malmstrom	1517	1062	455

March AFB	2204	1543	661
MNANG	780	546	234
Offutt	881	617	264
Peterson	2640	1848	792
Travis	2746	1922	824
USAFA	2573	1801	772

Fonte: Autor (2022).

4.3 APLICAÇÃO DOS ALGORITMOS

Foram aplicados três algoritmos diferentes para cada uma das doze localidades: Ridge, Árvore de Decisão e XGBoost para prever as Potências de Saída com base nos outros parâmetros já mencionados.

O modelo de Ridge foi selecionado como primeiro modelo por sua simplicidade e baixa necessidade de processamento, se constituindo como um bom ponto de partida para as comparações. Foram utilizados dois parâmetros o de normalização e o alpha, sendo o último ajustado em 0,5.

O segundo modelo escolhido foi o de Árvore de Decisão, que apesar de ter uma implementação simples também apresenta resultados eficientes. E por fim, para melhorar os resultados obtidos pelo modelo de Árvore de Decisão foi escolhido o modelo de XGBoost, que aplica melhorias ao modelo de Árvore de Decisão, sendo assim possível obter alto desempenho e resultados ainda melhores.

Cada algoritmo foi treinado usando o mesmo conjunto de dados em cada localidade e seus resultados previstos foram comparados com os mesmos resultados reais do conjunto de dados. Conforme já citado, 70% dos dados de cada local foram utilizados para treinar o modelo, chamados de amostra de treino. Os 30% restantes foram utilizados para realizar o comparativo com a previsão feita pelos modelos.

Para a confecção dos modelos foi utilizada a linguagem de programação Python através do Google Colab, aplicando a biblioteca Sklearn para os modelos de Ridge e Árvore de Decisão, e a biblioteca XGBoost foi utilizada para aplicação do modelo XGBoost.

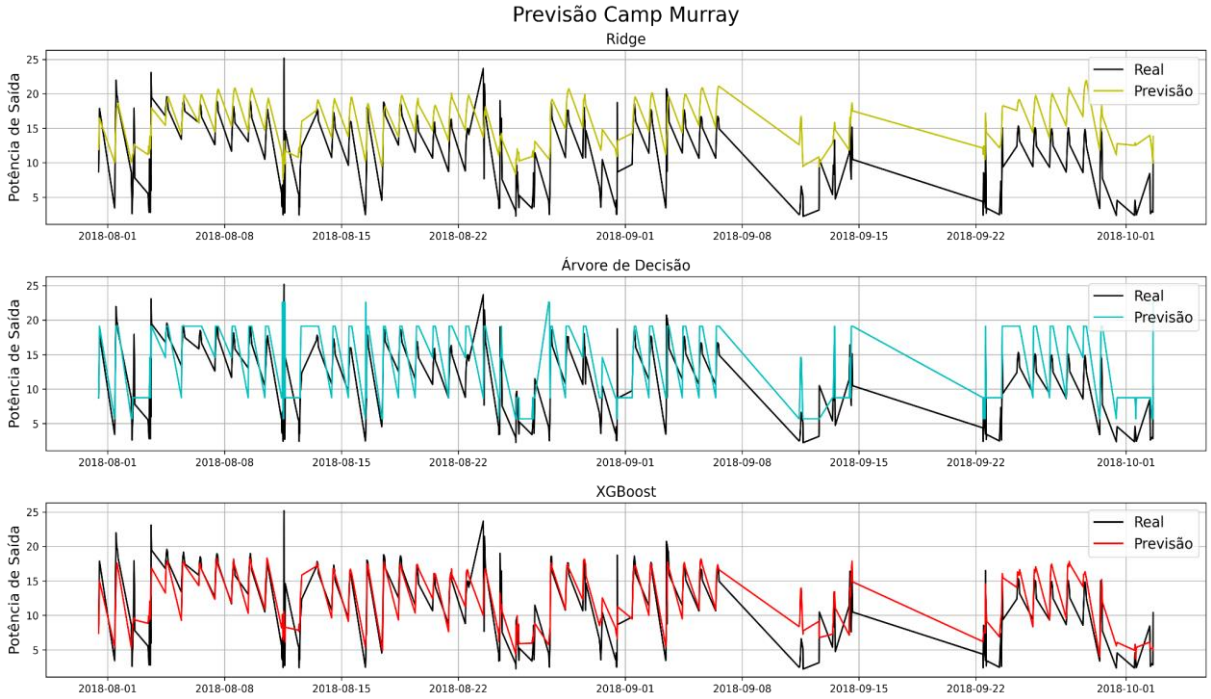
Para o modelo de Ridge foi utilizada a aplicação do parâmetro de normalização. Já para o modelo de Árvore de Decisão foi utilizada uma profundidade máxima de até 3 nós até a folha.

E para o XGBoost também foi utilizado profundidade máxima até a de 3 nós até a folha, além dos parâmetros: booster = "gbtree"; eta = 0.01; max_depth = 3; colsample_bytree = 0.1 e gamma = 0.5. Estes foram ajustados por tentativa e erro, a fim de obter uma melhor aproximação dos dados reais.

A seguir estão os resultados comparativos dos três modelos para cada localidade. Os resultados foram plotados utilizando as bibliotecas NumPy, Pandas e MatplotLib, de forma que fosse possível visualizar a diferença entre a potência de saída real, que são os dados de teste que foram divididos anteriormente, e a prevista pelos modelos.

A Figura 12 apresenta a previsão dos três modelos comparada com os dados reais (dados de teste) para a localidade de Camp Murray. As demais previsões das outras onze localidades se encontram nos apêndices de A à K do presente trabalho.

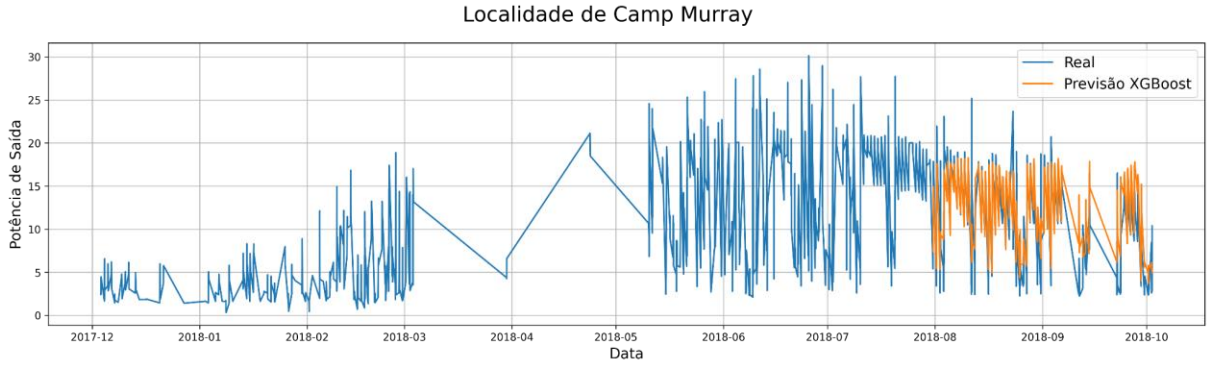
Figura 12 - Previsão dos Modelos para a localidade de Camp Murray.



Fonte: Autor (2022).

A Figura 13 apresenta a potência de saída para todas as amostras e é possível perceber a divisão dos dados utilizados para o treino do modelo e dos dados que foram utilizados para o teste. Em laranja está a previsão realizada utilizando o modelo XGBoost. Com isso, ficou ilustrada que a previsão consegue acompanhar a tendência dos dados e se aproxima bastante da curva traçada pelos dados reais em azul.

Figura 13 - Potência de saída para a localidade de Camp Murray e a previsão do XGBoost.



Fonte: Autor (2022).

4.4 COMPARAÇÃO DOS RESULTADOS

Para comparar os dados foram utilizadas três métricas: MAE, MSE e R^2. Isso porque nenhuma métrica sozinha seria capaz de avaliar a qualidade dos modelos, por isso a combinação de três métricas a fim de melhor descrever o comportamento dos modelos.

A Tabela 6 apresenta o MAE de cada modelo em cada localidade. Para comparar os modelos foi calculada uma média simples entre os resultados das doze localidades para cada modelo, objetivando facilitar a análise comparativa entre os modelos. Como o MAE representa a diferença entre o valor real e o previsto de forma absoluta, o menor valor de Erro Médio Absoluto demonstra que, considerando apenas essa métrica, o melhor modelo é o XGBoost - seguido do modelo de Árvore de Decisão e por último o modelo de Ridge. Os valores em negrito correspondem ao menor valor de MAE dentre os modelos para cada localidade.

Tabela 6 - Métrica de MAE de cada localidade para cada modelo.

Local	Ridge MAE	Árvore MAE	XGBoost MAE
Camp Murray	4,499759	3,7369925	**2,468416**
Grissom	4,3564905	3,532631	**3,4111905**
Hill Weber	4,3564905	3,532631	**3,411191**
JDMT	4,3564905	3,532631	**3,4111905**
Kahului	**5,3924435**	6,455702	6,0700591
Malmstrom	4,3759776	**4,313744**	4,3889594
March AFB	2,797221	2,537446	**2,5298294**

MNANG	4,297606	4,6505609	**4,0341001**
Offutt	4,2976064	4,6505609	**4,0341001**
Peterson	5,0224189	5,4228617	**4,4633355**
Travis	4,0406365	3,1787952	**3,1225907**
USAFA	**3,1515036**	3,5707992	3,2127881
Média	**4,245387**	**4,0929463**	**3,713146**

Fonte: Autor (2022).

A Tabela 7 apresenta o MSE de cada modelo em cada localidade e para comparar os modelos foi calculada uma média simples de todos os resultados para cada modelo, como na métrica anterior. Quanto menor o valor dessa métrica melhor é o modelo. Assim, o XGBoost seguido da Árvore de Decisão e do modelo Ridge. Os valores em negrito correspondem ao menor valor de MSE dentre os modelos para cada localidade.

Tabela 7 - Métrica de MSE de cada localidade para cada modelo.

Local	Ridge MSE	Árvore MSE	XGBoost MSE
Camp Murray	28,5613118	24,2570057	**11,410932**
Grissom	27,836287	23,8940628	**19,6847245**
Hill Weber	27,836287	23,8940628	**19,684725**
JDMT	27,836287	23,8940628	**19,6847245**
Kahului	**40,992382**	52,670455	47,6209669
Malmstrom	32,2356982	33,508105	**27,5610837**
March AFB	12,816102	11,8410053	**11,4880383**
MNANG	27,035222	39,9893792	**25,3426515**
Offutt	27,0352223	39,9893792	**25,3426515**
Peterson	34,4059289	44,0431989	**31,58448**
Travis	22,8873612	19,2080281	**17,1775667**
USAFA	17,4220713	20,5563381	**16,4079471**
Média	**27,2416801**	**29,8120902**	**22,749208**

Fonte: Autor (2022).

A Tabela 8 apresenta o R^2 de cada modelo em cada localidade. Para comparar os modelos foi calculada uma média dos valores para cada modelo. Como essa métrica mede um

coeficiente de determinação, é possível dizer que em termos gerais quanto mais próximo de 1 esse valor estiver melhor seria o modelo. Levando em considerando apenas essa métrica, o XGBoost obteve novamente o melhor resultado, e diferente do esperado, a Ridge foi considerada melhor que a Árvore de Decisão. Os valores em negrito correspondem ao maior valor de R^2 dentre os modelos para cada localidade.

Observando os gráficos comparativos apresentados anteriormente é possível identificar que o modelo de Árvore de Decisão ocasionou algumas previsões pontuais muito diferentes dos valores reais, podendo isso ter afetado a métrica, indicando uma qualidade pior da Árvore de Decisão nesse sentido.

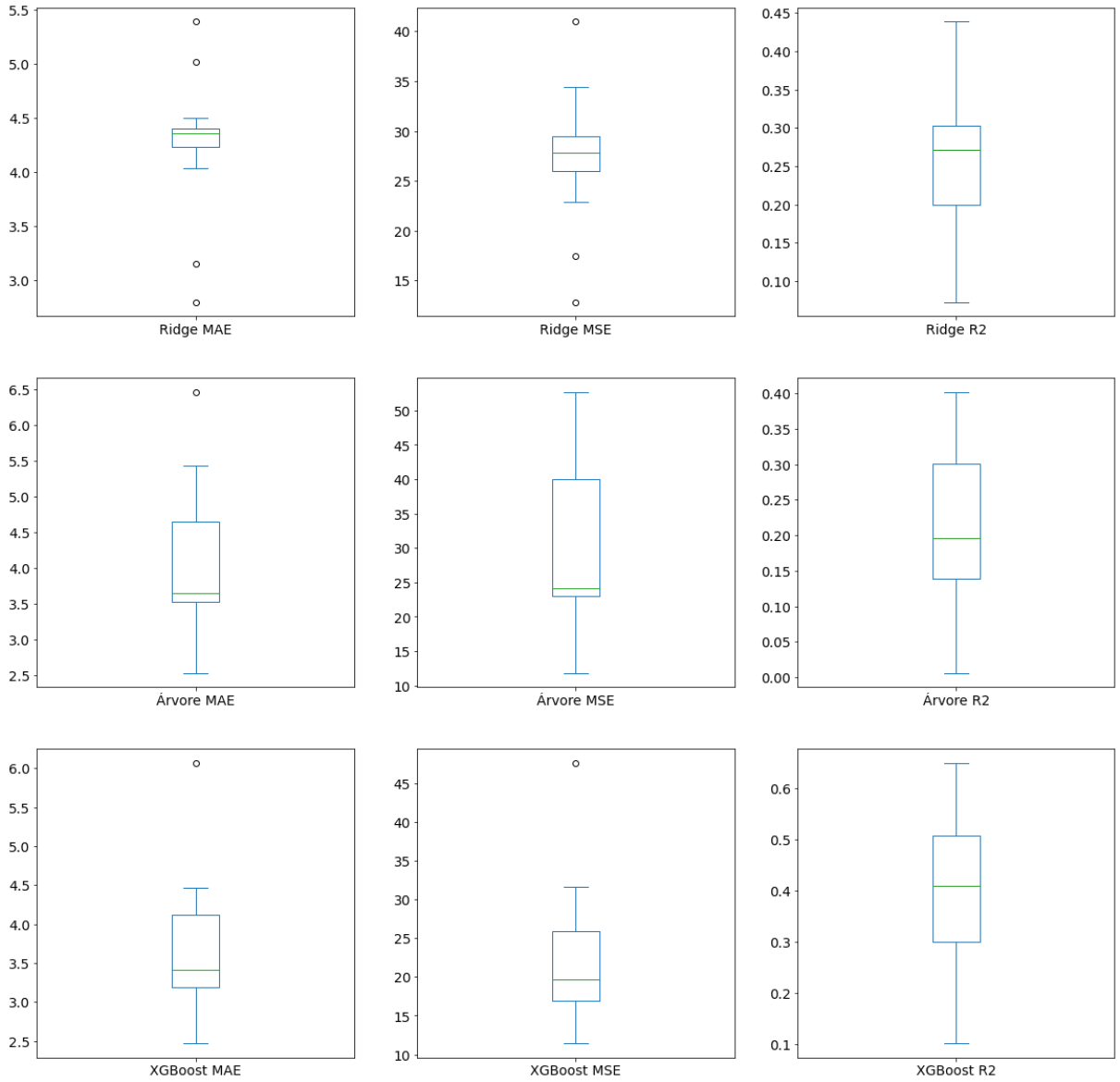
Tabela 8 - Métrica de R^2 de cada localidade para cada modelo.

Local	**Ridge R^2**	**Árvore R^2**	**XGBoost R^2**
Camp Murray	0,1219601	0,2542843	**0,649202**
Grissom	0,3029152	0,4016376	**0,5070491**
Hill Weber	0,3029152	0,4016376	**0,507049**
JDMT	0,3029152	0,4016376	**0,5070491**
Kahului	**0,2265455**	0,006201	0,1014757
Malmstrom	0,1771543	0,144675	**0,2964781**
March AFB	0,207245	0,267561	**0,289394**
MNANG	0,438851	0,1699716	**0,4739823**
Offutt	0,438851	0,169972	**0,473982**
Peterson	0,2853709	0,0851998	**0,343974**
Travis	0,0725483	0,2216439	**0,3039231**
USAFA	0,2577215	0,1241841	**0,3009289**
Média	**0,2612494**	**0,220717**	**0,3962073**

Fonte: Autor (2022).

Na Figura X, apresenta-se o boxplot feito a partir dos resultados das métricas, nela é possível visualizar a média dos valores, bem como seus valores máximos e mínimos. Assim, podemos observar que o XGBoost apresentou menores intervalos nas métricas MAE e MSE, ou seja, seus valores máximos e mínimos são mais próximos.

Figura 14 - Boxplot comparativo das métricas.



Fonte: Autor (2022).

O melhor desempenho do XGBoost já era esperado, pois esse aplica ajustes no modelo de Árvore de Decisão. Porém, foi surpreendente que o modelo de Ridge tenha apresentado resultados satisfatórios dada sua simplicidade.

4.4.1 Comparação com trabalhos recentes

Devido às características dos dados de nossa pesquisa, e da variação de medidas de geração de energia, a métrica de avaliação R^2 torna-se a mais simples e passível de comparação com demais estudos.

Tanev e Stanev (2021) obtiveram como resultado de R^2 do modelo Ridge o valor 0,991, em que o aplicado no presente trabalho atingiu uma média de 0,261. A discrepância entre os valores se deve a diversos fatores, entre eles o dataset utilizado. No caso do estudo desenvolvido por Tanev e Stanev (2021), o mesmo utilizou um conjunto grande de dados (178 dias, com cerca de 288 amostras cada), que apresentavam variáveis mais simples - contando com irradiação solar. O próprio estudo apresenta que a utilização das variáveis de irradiação solar, temperatura ambiente e vento do dataset em uma simples Regressão Linear já foram suficientes para atingir um valor de R^2 acima de 0,97 - concluindo ao fim que a utilização de tais dados é suficiente para uma boa predição.

Os mesmos autores, ao aplicarem o modelo de Árvore de Decisão obtiveram como R^2 o valor 0,987, em que no atual estudo, para o mesmo modelo e métrica, a média do resultado foi 0,220. Novamente é possível observar o desempenho muito inferior atingido, que pode ser justificado pelo mesmo fator mencionado para o modelo anterior.

Ao comparar os resultados obtidos a partir dos estudos de Pasion et al. (2020) e o atual, que contam com o mesmo dataset, os resultados obtidos pelo último também se apresentam inferiores. Pasion et al. (2021), a partir de diversos modelos, atingiu R^2 de: 0,939 para Random Forest Distribuída, 0,924 para Árvores Extremamente Aleatórias, 0,868 para *Stacked ensemble build*, 0,802 para máquina de aumento de gradiente, 0,593 para *Deep Learning* e 0,502 para Modelo linear generalizado. No atual trabalho, o modelo XGBoost que atingiu melhor resultado para R^2, de 0,396, ainda apresenta distância dos mencionados anteriormente. O fator dataset não possui influência por ser o mesmo, mas há a complexidade e robustez dos modelos utilizados por Pasion et al. (2020), os quais necessitam de mais poder computacional - diferindo-se da proposta dos modelos utilizados no atual trabalho.

Em seu estudo, os autores Urzagasti et al. (2021) aplicaram modelos de RNA utilizando dois datasets diferentes, com diferentes horizontes de predição de curto prazo. Ambos os datasets possuíam entre as variáveis dados de irradiação solar, mas o segundo era mais extenso, sendo possível observar diferença entre os dados de R^2 para o mesmo modelo e horizonte: para o dataset 1 o valor médio foi 0,763 e para o dataset 2 foi 0,930. A partir disso, contrastando com o trabalho atual que obteve desempenho bastante inferior, de modelos de menor

processamento e complexidade, é possível supor que a partir de um dataset de melhor qualidade (tanto de extensão quanto de variáveis) submetidos aos mesmos algoritmos já utilizados seria possível atingir resultados mais satisfatórios.

5 CONCLUSÕES E RECOMENDAÇÕES

5.1 CONCLUSÕES

O presente trabalho buscou comparar três modelos de Machine Learning para o problema de previsão de energia elétrica fotovoltaica. Conclui-se que os objetivos, tanto gerais como específicos, foram alcançados.

O primeiro objetivo específico, de classificação dos dados referentes às usinas analisadas, foi devidamente abordado e desenvolvido no capítulo 4.1, com a análise exploratória dos dados. O segundo e terceiro objetivos específicos, referentes à definição dos modelos de Machine Learning e treinamento dos mesmos, foram desenvolvidos no capítulo 4.3 com a aplicação dos algoritmos. O quarto e último objetivo específico, que trata da comparação dos modelos treinados a partir de indicadores de acurácia, foi desenvolvido no capítulo 4.4, de comparação dos resultados.

Por fim, foi possível observar tanto de maneira gráfica como nos comparativos das métricas de avaliação que o modelo XGBoost obteve melhor desempenho, seguido do modelo de Árvore de Decisão e por último o modelo que apresentou o pior desempenho foi o de Ridge - atingindo o objetivo geral do estudo.

Conclui-se que o Machine Learning é uma ferramenta poderosa para realizar previsões e suas aplicações podem ser as mais diversas possíveis, mesmo que ainda enfrentando alguns desafios como a necessidade de amplo conjunto de dados para previsões mais assertivas. Algoritmos como o XGBoost podem potencializar a assertividade das previsões e demandar menos processamento comparados a outros algoritmos mais difundidos.

A partir da comparação com demais trabalhos já presentes na literatura, foi possível observar que os resultados obtidos através dos modelos são inferiores. Isto deve-se a combinação de diversos motivos, como o caso da simplicidade dos algoritmos utilizados (que necessitam de baixo poder computacional), da divisão realizada no dataset - que um grande conjunto de dados se transformou em 12 pequenos, além da inexistência da variável de irradiação solar - que a literatura comprovou como relevante para acurácia dos modelos.

A previsibilidade de produção de energia elétrica é um dos grandes desafios do sistema elétrico brasileiro dada a atual expansão da geração proveniente de fontes como a fotovoltaica e eólica. A utilização de sistemas de previsões computacionais é uma boa forma de prever tal

produção, a aplicação de modelos que possam otimizar os recursos de processamento computacional evidencia-se necessária, levando em conta as dimensões do Sistema Interligado Nacional (SIN) e a quantidade de dados.

Assim, através dos resultados obtidos é possível ressaltar a necessidade de futuros estudos visando melhores aplicações e obtenção de dados, além de análise de outros modelos de Machine Learning e utilização de outras métricas para melhor compreender o comportamento dos dados.

5.2 TRABALHOS FUTUROS

O objetivo geral do presente trabalho, que buscava a proposição de um modelo de previsão para geração solar fotovoltaica mais preciso baseado em Machine Learning, foi atingido. Porém, durante o decorrer e conclusão do mesmo identificou-se outras diversas especificidades que ainda se constituem como desafios e devem ser estudadas e desenvolvidas.

A partir do presente trabalho, o mesmo pode ser melhorado e ajustado de forma com que seu método seja mais eficiente e conclusivo - utilizando outros datasets, parâmetros, métricas de avaliação e metodologias de análise.

Como já citado no decorrer do trabalho, a literatura aponta que maiores horizontes de predição levam a menor assertividade e desempenho dos modelos. Dessa maneira, outra sugestão de futuro trabalho é o desenvolvimento de modelos baseados em Machine Learning que possibilitam a predição de geração de energia fotovoltaica a um horizonte maior, possibilitando que tais dados possam ser utilizados para um planejamento energético de longo prazo.

Além disso, determinados algoritmos de Machine Learning, como os utilizados no presente trabalho, necessitam de uma grande e específica base de dados para que se tornem eficientes, as quais nem sempre estão disponíveis. Como outra sugestão de trabalho futuro está o desenvolvimento de um algoritmo que atinja considerável assertividade mesmo com dados mais básicos, possibilitando que instalações de menor porte ou até mesmo particulares possam ter acesso a uma predição segura daquilo que será gerado por seus equipamentos.

A forte expansão de outras diversas fontes de energias renováveis e limpas que também dependem de fatores não lineares, como o caso da energia eólica, também nos leva a novas

possibilidades de trabalhos futuros - podendo eles estarem relacionados a modelos de predição baseados em Machine Learning para tais fontes.

REFERÊNCIAS

ANDRADE, Fillipe de A. et al. Previsão da Geração de Energia Fotovoltaica Utilizando Inteligência Artificial em Séries Temporais. **Simpósio Brasileiro de Automação Inteligente - SBAI.** 2021. Disponível em: https://www.sba.org.br/open_journal_systems/index.php/sbai/article/view/2608. Acesso em: 20 mar. 2022.

ANTONIOLLI, Eduarda et al. Revisão sistemática: Aplicação de Redes Neurais para Previsão do Consumo de Energia Elétrica. **REVISTA DE ENGENHARIA E TECNOLOGIA**, v. 13, n. 2, 2021. Disponível em: https://revistas.uepg.br/index.php/ret/article/view/17956. Acesso em: 18 dez. 2022.

ALKHAYAT, Ghadah; MEHMOOD, Rashid. A review and taxonomy of wind and solar energy forecasting methods based on deep learning. **Energy and AI**, v. 4, p. 100060, 2021. Disponível em: https://www.sciencedirect.com/science/article/pii/S2666546821000148. Acesso em: 08 nov. 2022.

BARI, Anasse; CHAOUCHI, Mohamed; JUNG, Tommy. **Análise Preditiva para leigos**. Alta Books, 2020.

BERGAMO, Henrique Postingel; RAMOS, Lucas Gomes dos. **Análise preditiva da geração fotovoltaica via algoritmos de inteligência computacional: modelagem e estudo de caso da Usina Solar Bom Jesus da Lapa-BA**. Universidade Estadual Paulista (UNESP), 2022. Disponível em: https://repositorio.unesp.br/handle/11449/216985. Acesso em: 08 nov. 2022.

BEZERRA, Francisco Diniz. Energia solar. **Banco do Nordeste do Brasil**, 2021. Disponível em: https://www.bnb.gov.br/s482-dspace/handle/123456789/227. Acesso em: 22 out. 2022.

BORGES, Fabricio Quadros. Planejamento público de matrizes elétricas sustentáveis e inteligência artificial. **Desenvolvimento em Debate,** v. 9, n. 3, p. 191-211, 2021. Disponível em: https://web.archive.org/web/20220511132933id_/https://inctpped.ie.ufrj.br/desenvolvimentoemdebate/pdf/revista_dd_v9_n3_fabricio_e_fabrini_quadros_borges.pdf. Acesso em: 19 dez. 2022.

BORGES, Luiz Eduardo. **Python para desenvolvedores: aborda Python 3.3.** Novatec Editora, 2014.

BRAGA, Magno Cesar Fernandes. **Previsão de geração fotovoltaica utilizando o método Cascade Forward Back Propagation de redes neurais artificiais.** 2022. 57 f. Trabalho de Conclusão de Curso (Graduação) - Engenharia Elétrica, Universidade Federal do Ceará. Fortaleza, 2022. Disponível em: https://repositorio.ufc.br/bitstream/riufc/65865/3/2022_tcc_mcfbraga.pdf#cite.marinho2020resultados. Acesso em: 08 nov. 2022.

BRASIL, Agência Nacional de Energia Elétrica (ANEEL). **Atlas da Energia Elétrica do Brasil.** 2.ed. Brasília - Distrito Federal, 2005.

CARNEIRO, Tatiane Carolyne et al. Review on photovoltaic power and solar resource forecasting: current status and trends. **Journal of Solar Energy Engineering**, v. 144, n. 1, 2022. Disponível em: https://asmedigitalcollection.asme.org/solarenergyengineering/article-abstract/144/1/010801/1114259/Review-on-Photovoltaic-Power-and-Solar-Resource. Acesso em: 18 dez. 2022.

CARVALHO, André et al. **Inteligência Artificial: Uma abordagem de Aprendizado de Máquina.** Rio de Janeiro: Ltc, 2011

CASTRO, Rui M. G. **Introdução à energia fotovoltaica.** Universidade Técnica de Lisboa, Instituto Superior Técnico, 2002. Disponível em: https://silo.tips/download/introduao-a-energia-fotovoltaica. Acesso em: 22 out. 2022.

CHEN, Tianqi; GUESTRIN, Carlos. XGBoost: A Scalable Tree Boosting System. **22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,** p. 785-794, 2016. Disponível em: https://arxiv.org/abs/1603.02754. Acesso em: 22 out. 2022.

CHIANG, Wanchih et al. A study of exact ridge regression for big data. **2018 IEEE International Conference on Big Data (Big Data)**. IEEE, 2018. p. 3821-3830. Disponível em: https://ieeexplore.ieee.org/abstract/document/8622274?casa_token=rjNfDZq81R8AAAAA:GZn8Xo77d4QynbhsxeEkVen-3AcLpBi9xpd7b7ssMwcCsUYJ_w-wv-7fzUvIq9UXmoq-b9BkLgc. Acesso em: 18 dez. 2022.

COLABORATORY, Colaboratory. **Conheça o Colab.** 2022. Disponível em: https://colab.research.google.com/notebooks/intro.ipynb. Acesso em: 22 out. 2022.

COUTINHO, Thiago. **O que é a biblioteca Pandas?** Voitto. Cidade, junho de 2021. Disponível em: https://www.voitto.com.br/blog/artigo/biblioteca-pandas. Acesso em: 22 out. 2022.

COUNTRY, Rankings. **Agência Internacional de Energia Renovável - IRENA**, 2022. Disponível em: https://irena.org/Statistics/View-Data-by-Topic/Capacity-and-Generation/Country-Rankings. Acesso em: 20 mar. 2022.

CRUZ, Josélio C. et al. **Estado da Arte para Previsão de Eventos de Rampas Eólicas no Sistema de Energia.** 2022. Disponível em: https://www.sba.org.br/cba2022/wp-content/uploads/artigos_cba2022/paper_3569.pdf. Acesso em: 18 dez. 2022.

CUNHA, Henrique Queiroz; SOBEL, Leonardo Farias. Aplicação de Redes Neurais Artificiais na Predição de Geração de Energia Fotovoltaica no Nordeste do Brasil. **Revista de Engenharia e Pesquisa Aplicada**, v. 6, n. 5, p. 73-80, 2021. Disponível em: http://www.revistas.poli.br/~anais/index.php/repa/article/view/1767. Acesso em: 08 nov. 2022.

DA SILVA, Francisco Eduardo Mendes et al. Short-term renewable electric energy generation forecast in the state of Ceará using prophet regression. **Research, Society and Development**, v. 11, n. 7, p. 01-09, 2022. Disponível em: https://rsdjournal.org/index.php/rsd/article/view/29579. Acesso em: 08 nov. 2022.

DA SILVA, Larissa Camila Ferreira. **Modelo de Transferência de Aprendizagem baseado em Regressão Linear Regularizada**. Trabalho de Conclusão de Curso (Graduação) - Engenharia de Computação, Universidade Federal de Pernambuco. Recife, 2017. Disponível em: https://www.cin.ufpe.br/~tg/2017-1/lcfs_tg.pdf. Acesso em: 18 dez. 2022.

DAS, Utpal Kumar et al. Forecasting of photovoltaic power generation and model optimization: A review. **Renewable and Sustainable Energy Reviews**, v. 81, p. 912-928, 2018. Disponível em: https://www.sciencedirect.com/science/article/abs/pii/S1364032117311620?via%3Dihub. Acesso em: 18 dez. 2022.

DAMACENO, Laura. **Regressão Linear?** Medium. Junho de 2021. Disponível em: https://medium.com/@lauradamaceno/regress%C3%A3o-linear-6a7f247c3e29. Acesso em: 22 out. 2022.

DE SOUSA OLIVEIRA, Patrícia. **Otimização da Agenda de Manutenção das Turbinas de uma Usina Hidrelétrica**. 2021. Dissertação (mestrado) - Departamento de Engenharia Industrial, Pontifícia Universidade Católica do Rio de Janeiro. Rio de Janeiro, 2021. Disponível em: https://www.maxwell.vrac.puc-rio.br/52477/52477.PDF. Acesso em: 18 dez. 2022.

DEVELOPERS, Scikit-Learn. **3.3 Metrics and scoring: quantifying the quality of predictions.** 2022. Disponível em: https://scikit-learn.org/stable/modules/model_evaluation.html#median-absolute-error. Acesso em: 22 out. 2022.

DEVELOPERS, xgboost. **XGBoost Documentation.** 2021. Disponível em: https://xgboost.readthedocs.io/en/stable/. Acesso em: 22 out. 2022.

_____. **XGBoost Parameters.** 2021. Disponível em: https://xgboost.readthedocs.io/en/stable/parameter.html. Acesso em: 22 out. 2022.

DIAZ, Daniel. **Um Guia para Iniciantes em Programação Orientada a Objetos (POO) Python.** Kinsta, 5 de out. 2022. Disponível em: https://kinsta.com/pt/blog/programacao-orientada-objetos-python/. Acesso em: 19 out. 2022.

DHALIWAL, Sukhpreet Singh; NAHID, Abdullah-Al; ABBAS, Robert. Effective intrusion detection system using XGBoost. **Information,** v. 9, n. 7, p. 149, 2018. Disponível em: https://www.mdpi.com/2078-2489/9/7/149. Acesso em: 22 out. 2022.

DOMINGOS, Samira Fontes et. al. Estado da Arte para a Previsão da Radiação Solar. **ANAIS VII Congresso Brasileiro de Energia Solar - CBENS 2018**. 2020. Disponível em: https://anaiscbens.emnuvens.com.br/cbens/article/view/742. Acesso em: 20 mar. 2022.

DOS SANTOS, Cochiran Pereira et al. Análise de materiais utilizados na produção de diodos emissores de luz para possíveis aplicações em painéis solares fotovoltaicos. **Ciência e Engenharia de Materiais: Conceitos, Fundamentos e Aplicações,** v. 1, n. 1, p. 60-67, 2021. Disponível em: https://www.editoracientifica.com.br/articles/code/210805656. Acesso em: 18 dez. 2022.

ESCOVEDO, Tatiana; KOSHIYAMA, Adriano S. **Introdução à Data Science - Algoritmos de Machine Learning e métodos de análise**. São Paulo, Ed. Casa do Código, 2020.

EUROPE, SolarPower. **Global market outlook for solar power/2022–2026.** Solar Power Europe: Brussels, Belgium, 2022. Disponível em: https://api.solarpowereurope.org/uploads/Solar_Power_Europe_Global_Market_Outlook_for_Solar_Power_2022_2026_V01_b720ac6c3c.pdf. Acesso em: 08 nov. 2022.

GIL, Antonio Carlos. **Métodos e técnicas de pesquisa social**. 6. ed., São Paulo: Atlas, 2008.

GLOBAL Photovoltaic Power Potential by Country. **Global Solar Atlas,** 2022. Disponível em: https://globalsolaratlas.info/global-pv-potential-study. Acesso em: 20 mar. 2022.

GONFALONIERI, Alexandre. **How to Implement Machine Learning For Predictive Maintenance.** Towards Data Science, 7 nov. 2019. Disponível em: https://towardsdatascience.com/how-to-implement-machine-learning-for-predictive-maintenance-4633cdbe4860. Acesso em: 15 out. 2022.

GUPTA, Aakash et al. Solar energy prediction using decision tree regressor. In: **2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS)**. IEEE, 2021. p. 489-495. Disponível em: https://ieeexplore.ieee.org/document/9432322. Acesso em: 08 nov. 2022.

HOMEM, William Ludovico. **Apostila de Machine Learning.** PET Engenharia Mecânica, UFES, 2020. Disponível em: https://petengenhariamecanica.ufes.br/sites/petengenhariamecanica.ufes.br/files/field/anexo/apostila_do_minicurso_de_machine_learning.pdf. Acesso em: 22 out. 2022.

IMHOFF, Johninson. **Desenvolvimento de Conversores Estáticos para Sistemas Fotovoltaicos Autônomos**. Dissertação de Mestrado apresentada à Escola de Engenharia Elétrica da Universidade Federal de Santa Maria, Santa Maria. 2007. 146 f. Disponível em: https://repositorio.ufsm.br/handle/1/8608. Acesso em: 22 out. 2022.

KAUFMAN, Dora. **A inteligência artificial irá suplantar a inteligência humana?** São Paulo: Estação das Letras e Cores, 2019.

KAUFMAN, Dora; SANTAELLA, Lucia. O papel dos algoritmos de inteligência artificial nas redes sociais. **Revista Famecos**, v. 27, 2020. Disponível em: https://revistaseletronicas.pucrs.br/ojs/index.php/revistafamecos/article/view/34074. Acesso em: 22 out. 2022.

LACERDA, Daniel Pacheco et al. Design Science Research: método de pesquisa para a engenharia de produção. **Gestão & produção**, v. 20, p. 741-761, 2013. Disponível em: https://www.scielo.br/j/gp/a/3CZmL4JJxLmxCv6b3pnQ8pq/?format=pdf&lang=pt. Acesso em: 22 out. 2022.

LANA, Luana Teixeira Costa et al. Energia solar fotovoltaica: revisão bibliográfica. **Engenharias On-line**, v. 1, n. 2, p. 21-33, 2015. Disponível em: http://revista.fumec.br/index.php/eol/article/view/3574. Acesso em: 22 out. 2022.

LEMOS, Eliardo Vinicius Bezerra. **Estudo de caso de projeto de instalação de painéis fotovoltaicos.** Trabalho de Conclusão de Curso (Graduação) - Engenharia Elétrica, Universidade Federal de Campina Grande. Campina Grande, 2022. Disponível em: http://dspace.sti.ufcg.edu.br:8080/xmlui/handle/riufcg/27417. Acesso em: 18 dez. 2022.

LEVA, Sonia et al. Analysis and validation of 24 hours ahead neural network forecasting of photovoltaic output power. **Mathematics and computers in simulation**, v. 131, p. 88-100, 2017. Disponível em: https://www.sciencedirect.com/science/article/abs/pii/S0378475415001238. Acesso em: 08 nov. 2022.

LIU, Zhao; ZHANG, Ziang. Solar forecasting by K-Nearest Neighbors method with weather classification and physical model. In: **2016 North American Power Symposium (NAPS)**. IEEE, 2016. p. 1-6. Disponível em: https://ieeexplore.ieee.org/abstract/document/7747859/. Acesso em: 22 out. 2022.

LUDERMIR, Teresa Bernarda. Inteligência Artificial e Aprendizado de Máquina: estado atual e tendências. **Estudos Avançados**, v. 35, p. 85-94, 2021. Disponível em: https://www.revistas.usp.br/eav/article/view/185035. Acesso em: 08 nov. 2022.

MACHADO, Carolina T.; MIRANDA, Fabio S. Energia Solar Fotovoltaica: uma breve revisão. **Revista Virtual de Química**, v. 7, n. 1, p. 126-143, 2015. Disponível em: https://rvq-sub.sbq.org.br/index.php/rvq/article/view/664. Acesso em: 20 mar. 2022.

MAHESH, Batta. Machine Learning Algorithms - A Review. **International Journal of Science and Research (IJSR).** v. 9, n. 1, p. 381-386, 2020. Disponível em: https://www.ijsr.net/archive/v9i1/ART20203995.pdf. Acesso em: 08 nov. 2022.

MARINHO, Felipe Pinto et al. Resultados preliminares de previsão de irradiação solar de curto prazo através da combinação de processamento de imagens com algoritmos de aprendizagem de máquina. In: **VII Congresso Brasileiro de Energia Solar-CBENS 2018**. 2020. Disponível em: https://anaiscbens.emnuvens.com.br/cbens/article/view/756. Acesso em: 08 nov. 2022.

MATOS, Gonçalo Ribeiro de. **Machine learning aplicado à gestão de activos físicos industriais**. Tese de Doutorado, Instituto Superior de Engenharia de Lisboa, Lisboa, 2021. Disponível em: https://repositorio.ipl.pt/bitstream/10400.21/13524/1/Disserta%C3%A7%C3%A3o.pdf. Acesso em: 18 out. 2022.

MCKINNEY, Wes. **Python for Data Analysis.** 2. ed. Sebastopol: O'Reilly Media, 2017.

MELLIT, Adel; PAVAN, Alessandro Massi. A 24-h forecast of solar irradiance using artificial neural network: Application for performance prediction of a grid-connected PV plant at Trieste, Italy. **Solar energy**, v. 84, n. 5, p. 807-821, 2010. Disponível em: https://www.sciencedirect.com/science/article/abs/pii/S0038092X10000782. Acesso em: 08 nov. 2022.

MENEZES, Nilo Ney Coutinho. **Introdução a programação com Python.** São Paulo: Novatec, 2010.

MULINARI, Bruna. **Numpy Python: O que é, vantagens e tutorial inicial.** Harve - escola de inovação. Curitiba, s.d. Disponível em: https://harve.com.br/blog/programacao-python-blog/numpy-python-o-que-e-vantagens-e-tutorial-inicial/. Acesso em: 22 out. 2022.

NESPOLI, Alfredo et al. Day-ahead photovoltaic forecasting: A comparison of the most effective techniques. **Energies**, v. 12, n. 9, p. 1621, 2019. Disponível em: https://www.mdpi.com/1996-1073/12/9/1621. Acesso em: 08 nov. 2022.

OHLSON, James Arvid; KIM, Seil. Linear valuation without OLS: the theil-sen estimation approach. **Review Accounting Studies,** 2015, p. 395-435. Disponível em: http://link.springer.com/article/10.1007%2Fs11142-014-9300-0. Acesso em: 22 out. 2022.

OLIVEIRA, Thalles Rodrigues de. **Geração De Energia X Impacto Ambiental.** 2011. Trabalho de Conclusão de Curso (Graduação em Engenharia Elétrica) - Universidade do Estado de Minas Gerais, Minas Gerais, 2011. Disponível em: http://www.waltenomartins.com.br/tcc_2011_Thalles.pdf. Acesso em: 10 de jul. de 2018.

ONS, Operador Nacional do Sistema Elétrico. **O sistema em números**. 2022. Disponível em: http://www.ons.org.br/paginas/sobre-o-sin/o-sistema-em-numeros. Acesso em: 23 de mar. de 2022.

OSISANWO, F. Y. et al. Supervised machine learning algorithms: classification and comparison. **International Journal of Computer Trends and Technology (IJCTT)**, v. 48, n. 3, p. 128-138, 2017. Disponível em: https://ir.tech-u.edu.ng/344/1/AkinsolaJET-IJCTT-V48P126.pdf. Acesso em: 22 out. 2022.

PASCUAL, Christian. **Tutorial: Understanding Regression Error Metrics in Python.** Dataquest. Setembro de 2018. Disponível em: https://www.dataquest.io/blog/understanding-regression-error-metrics/. Acesso em: 22 out. 2022.

PASCUETTI, Douglas Yuri Leite. **Alternativas para regulação de velocidade de turbinas em micro centrais hidrelétricas**. Trabalho de Conclusão de Curso (Graduação) - Engenharia Elétrica,

Universidade Tecnológica Federal do Paraná. Pato Branco, 2021. Disponível em: http://repositorio.utfpr.edu.br/jspui/handle/1/28056. Acesso em: 18 dez. 2022.

PASION, Christil et al. Machine Learning Modeling of Horizontal Photovoltaics Using Weather and Location Data. **Energies,** v. 13, n. 10, p. 2570, 2020. Disponível em: https://www.mdpi.com/1996-1073/13/10/2570. Acesso em: 22 out. 2022.

PEDREGOSA, Fabian et al. Scikit-learn: Machine learning in Python. **The Journal of Machine Learning Research,** v. 12, p. 2825-2830, 2011. Disponível em: https://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf?ref=https://githubhelp.com. Acesso em: 22 out. 2022.

PINTO, Milton de Oliveira. **Fundamentos de Energia Eólica**. Rio de Janeiro: LTC, 2013.

REZENDE, Jaqueline Oliveira. A importância da Energia Solar para o Desenvolvimento Sustentável. **Atena Editora**, 2019.

ROCHA, P. A. et al. Estimation of daily, weekly and monthly global solar radiation using ANNs and a long data set: a case study of Fortaleza, in Brazilian Northeast region. **International Journal of Energy and Environmental Engineering,** v. 10, n. 3, p. 319–334, 2019. Disponível em: https://link.springer.com/article/10.1007/s40095-019-0313-0. Acesso em: 18 dez. 2022.

RODRIGUES, Gabriel Ferreira; MINOTTI, Cristiano; FLORIAN, Fabiana. Energia Fotovoltaica - Aplicação Sistema On Grid em Residência. **RECIMA21 - Revista Científica Multidisciplinar-ISSN 2675-6218**, v. 3, n. 1, p. e3122434-e3122434, 2022. Disponível em: https://www.recima21.com.br/index.php/recima21/article/view/2434. Acesso em: 18 dez. 2022.

ROHLEDER, Diego. **Caracterização de um acumulador hidráulico para armazenamento de energia solar fotovoltaica.** Trabalho de Conclusão de Curso (Graduação) - Engenharia Ambiental e Sanitária, Universidade Federal da Fronteira Sul. Cerro Largo, 2021. Disponível em: https://rd.uffs.edu.br/handle/prefix/5262. Acesso em: 18 dez. 2022.

RUSSELL, Stuart; NORVIG, Peter. **Inteligência artificial.** Tradução Regina Célia Simille. Rio de Janeiro: Campus Elsevier, 2013.

RÜTHER, Ricardo. **Edifícios Solares Fotovoltaicos:** O potencial da geração solar fotovoltaica integrada a edificações urbanas e interligada à rede elétrica pública no Brasil. Editora Labsolar, Florianópolis, 2004. Disponível em: https://fotovoltaica.ufsc.br/sistemas/livros/livro-edificios-solares-fotovoltaicos.pdf. Acesso em: 22 out. 2022.

SANTOS, Silvio Silva dos. **Previsão da produção de uma usina fotovoltaica usando redes neurais artificiais**. 2019. 65f. Trabalho de Conclusão de Curso (Graduação) - Engenharia Elétrica, Universidade Federal de Goiás. Goiânia, 2019. Disponível em: https://repositorio.bc.ufg.br/handle/ri/18991. Acesso em: 08 nov. 2022.

SAUAIA, Rodrigo Lopes. **Energia solar fotovoltaica: panorama, oportunidades e desafios.** Associação Brasileira de Energia Solar Fotovoltaica (ABSOLAR). São Paulo, 2017. Disponível em:https://sitefiespstorage.blob.core.windows.net/uploads/2017/09/rodrigo_6899.pdf. Acesso em: 22 out. 2022.

SEMOLINI, Robinson. **Support Vector Machines, Inferência Transdutiva e o Problema de Classificação**. Dissertação de Mestrado, Universidade Estadual de Campinas - UNICAMP, Engenharia Elétrica e de Computação, 2002, Campinas/SP. Disponível em:

https://www.dca.fee.unicamp.br/~vonzuben/theses/semolini_mest/semolini_tese_mest.pdf. Acesso em: 22 out. 2022.

SILVA, Lucas Tavares da. **Aprendizado de máquina aplicado na previsão da geração de energia elétrica de uma usina solar fotovoltaica.** Trabalho de Conclusão de Curso (Graduação) - Engenharia Elétrica, Universidade Federal do Ceará. Fortaleza, 2022. Disponível em: https://repositorio.ufc.br/handle/riufc/65867. Acesso em: 18 dez. 2022.

SILVA, Rafael Gomes da. **Análise Comparativa entre os Modelos CNN e LSTM para Predição de Fluxo do Tráfego Urbano na Cidade de Recife.** 2020. 61 f. TCC (Graduação) - Curso de Ciência da Computação, Escola de Ciências Exatas e da Computação, Pontifícia Universidade Católica de Goiás, Goiânia, 2020. Disponível em: https://repositorio.pucgoias.edu.br/jspui/handle/123456789/360. Acesso em: 15 de out. 2022.

SILVA, Rafael Martini et al. Estimativa da irradiância local utilizando inteligência artificial. In: **VII Congresso Brasileiro de Energia Solar-CBENS 2018**. 2020. Disponível em: https://anaiscbens.emnuvens.com.br/cbens/article/view/744. Acesso em: 08 nov. 2022.

SINGH, Neha; JENA, Satyaranjan; PANIGRAHI, Chinmoy Kumar. A novel application of Decision Tree classifier in solar irradiance prediction. **Materials Today: Proceedings**, v. 58, p. 316-323, 2022. Disponível em: https://www.sciencedirect.com/science/article/pii/S2214785322008124. Acesso em: 08 nov. 2022.

SIVAKUMAR, Aravinth; GUNASUNDARI, Ramalingam. A Survey on Data Preprocessing Techniques for Bioinformatics and Web Usage Mining. **International Journal of Pure and Applied Mathematics**, v. 117, n. 20, p. 785-794, 2017. Disponível em: https://www.semanticscholar.org/paper/A-Survey-on-Data-Preprocessing-Techniques-for-and-Sivakumar-Gunasundari/d365e3372a7b42d615e57ce99e0738d6ebb3094c. Acesso em: 08 nov. 2022.

SOUZA, Emanuel de. **Implementando Regressão Linear Simples em Python.** Medium. Janeiro de 2019. Disponível em: https://medium.com/data-hackers/implementando-regress%C3%A3o-linear-simples-em-python-91df53b920a8. Acesso em: 22 out. 2022.

TANEV, Tanyo; STANEV, Rad. Modeling of Photovoltaic Power Plant Electricity Generation Using Machine Learning Methods. **2021 17th Conference on Electrical Machines, Drives and Power Systems (ELMA)**. IEEE, 2021. p. 1-6. Disponível em: https://ieeexplore.ieee.org/abstract/document/9503066/?casa_token=Jma5yl_QQaIAAAAA:J0BUW6SjpM6FyRv_cHdjs2rU9fNVMumUMBbl7o7WNoOycAx7Cfps7YPE9mByddpDeNAvDhKnriI. Acesso em: 18 dez. 2022.

TURMINA, Eliana, et al. Avaliação de impactos ambientais gerados na implantação e operação de subestação de energia elétrica: um estudo de caso em Palhoça, SC. **Revista de Ciências Agroveterinárias**, Lages, v. 17, n. 4, p. 589-598, 2018. Disponível em: https://www.revistas.udesc.br/index.php/agroveterinaria/article/view/10482. Acesso em: 18 dez. 2022.

URZAGASTI, Carlos Alejandro et al. Comparação da Acurácia de Modelos de Redes Neurais Artificiais na Predição da Irradiância Solar e Geração de Energia Fotovoltaica. In: **Anais do XVIII Congresso Latino-Americano de Software Livre e Tecnologias Abertas**. SBC, 2021. p. 53-58. Disponível em: https://sol.sbc.org.br/index.php/latinoware/article/view/19905. Acesso em: 08 nov. 2022.

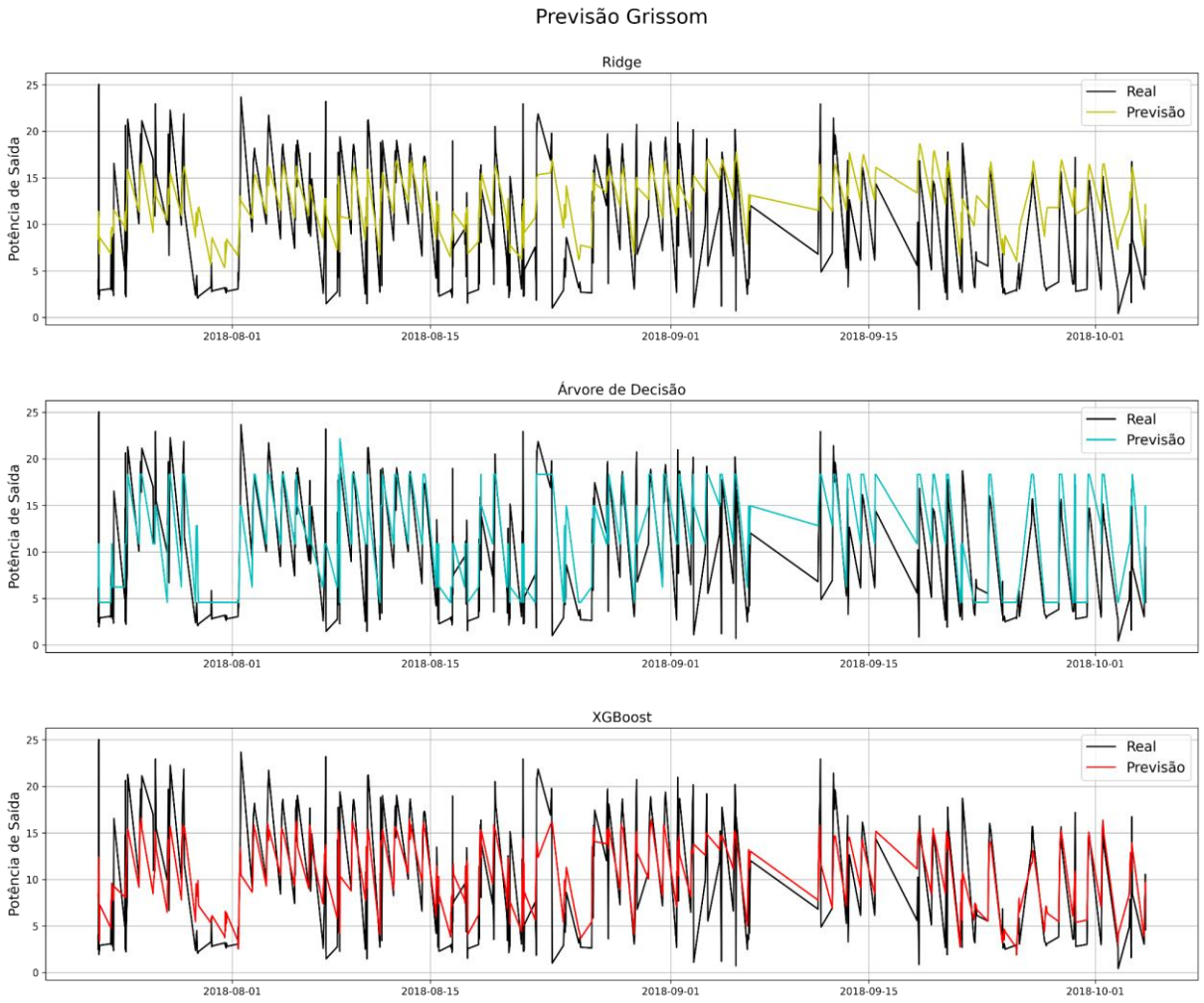
VOYANT, Cyril et al. Machine learning methods for solar radiation forecasting: A review. **Renewable Energy**, v. 105, p. 569-582, 2017. Disponível em: https://www.sciencedirect.com/science/article/abs/pii/S0960148116311648. Acesso em: 08 nov. 2022.

WENTZ, Victor Hugo. **Avaliação de modelos de inteligência artificial aplicados à predição de irradiância solar**. 2021. 75 f. Trabalho de Conclusão de Curso (Graduação) - Engenharia Física, Universidade Federal da Integração Latino-Americana,. Foz de Iguaçu, 2021. Disponível em: https://dspace.unila.edu.br/handle/123456789/6319. Acesso em: 08 nov. 2022.

WILLIAMS, Jada; WAGNER, Torrey. **Northern Hemisphere Horizontal Photovoltaic Power Output Data for 12 Sites.** Mendeley Data, 2019. Disponível em: https://data.mendeley.com/datasets/hfhwmn8w24. Acesso em: 22 out. 2022.

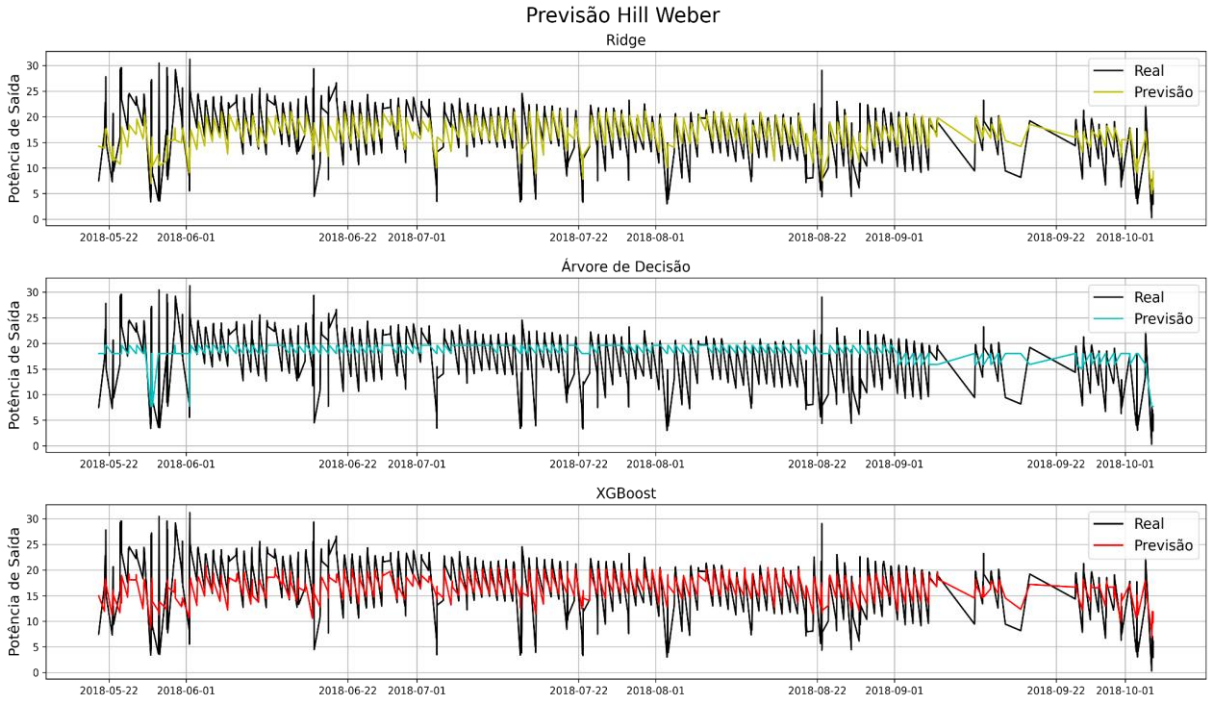
YADAV, Amit Kumar et al. Selection of most relevant input parameters using WEKA for artificial neural network based solar radiation prediction models. **Renewable and Sustainable Energy Reviews,** v. 31, p. 509–519, 2014. Disponível em: https://www.sciencedirect.com/science/article/pii/S1364032113008228?casa_token=iBzu6FUCuTQAAAAA:aXp_e_rI3Rtw0rKctVUZxcqAlkDoTloIbIwxIkAtDnokySug_RJdaUjJ1WBYkoMu4RJhybLcD0o. Acesso em: 18 dez. 2022.

APÊNDICE A – PREVISÃO DOS MODELOS: FIGURA DE RESULTADOS OBTIDOS PELOS TRÊS MODELOS PARA A LOCALIDADE DE GRISSOM



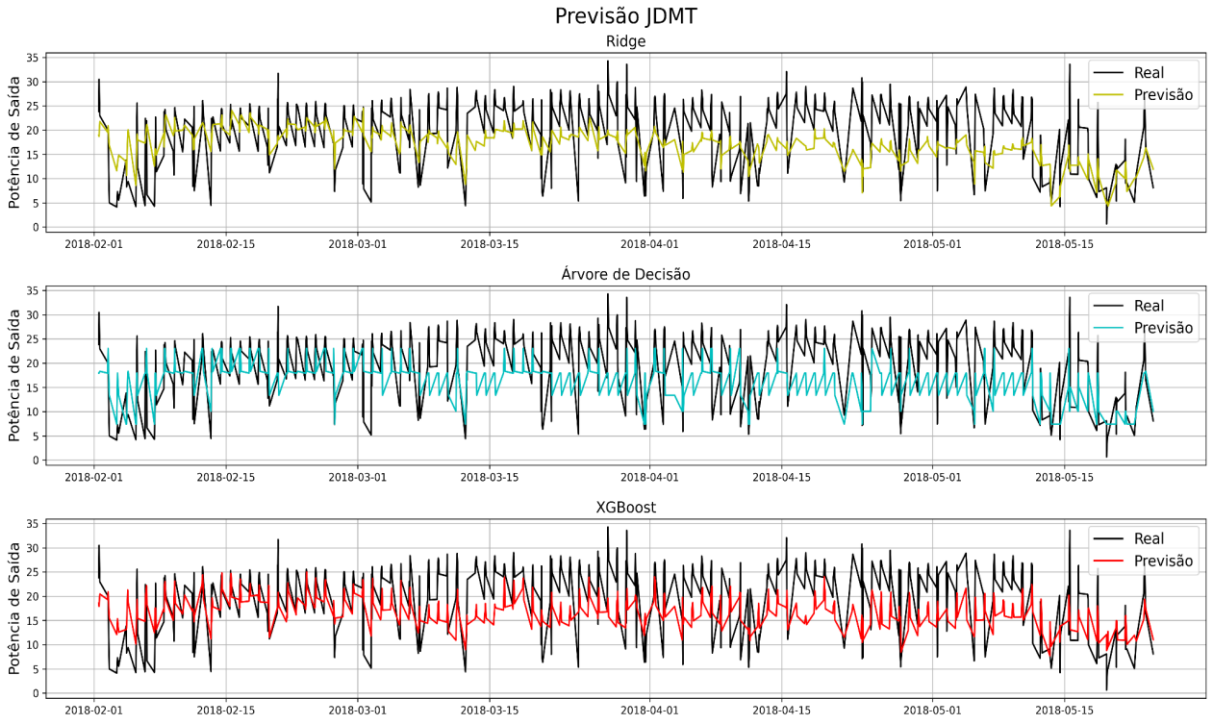
Fonte: Autor (2022).

APÊNDICE B – PREVISÃO DOS MODELOS: FIGURA DE RESULTADOS OBTIDOS PELOS TRÊS MODELOS PARA A LOCALIDADE DE HILL WEBER



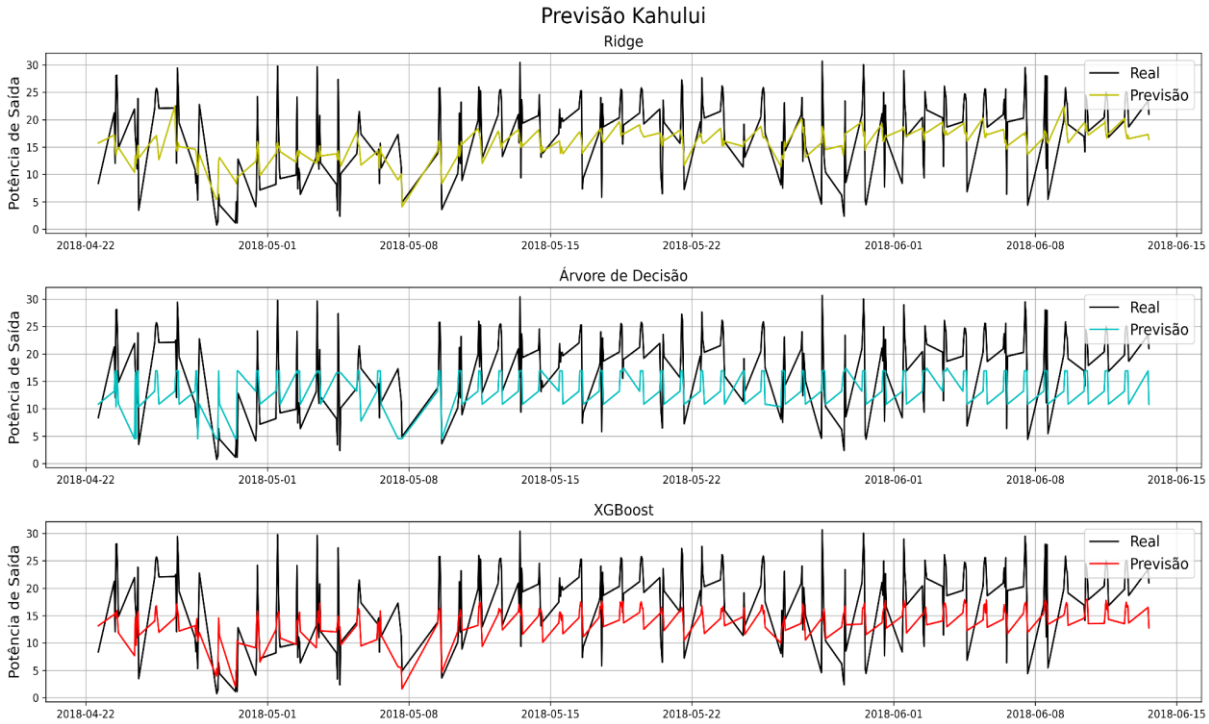
Fonte: Autor (2022).

APÊNDICE C – PREVISÃO DOS MODELOS: FIGURA DE RESULTADOS OBTIDOS PELOS TRÊS MODELOS PARA A LOCALIDADE DE JDMT



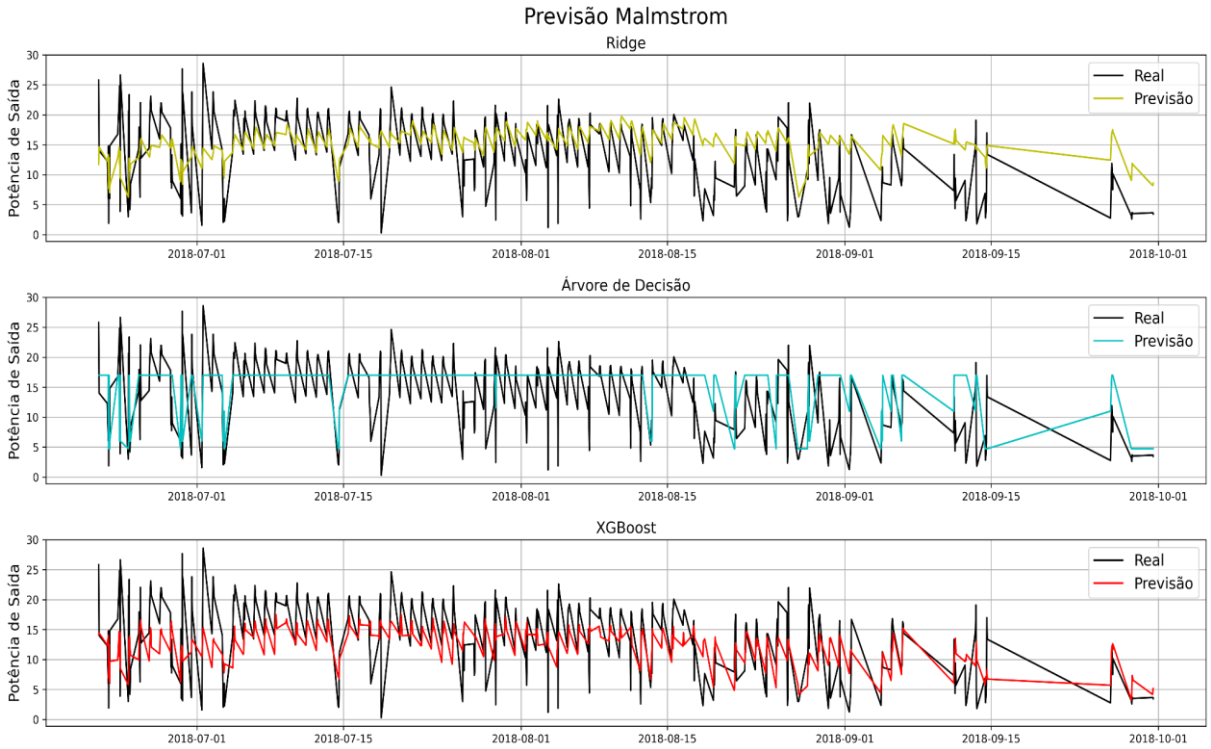
Fonte: Autor (2022).

APÊNDICE D – PREVISÃO DOS MODELOS: FIGURA DE RESULTADOS OBTIDOS PELOS TRÊS MODELOS PARA A LOCALIDADE DE KAHULUI



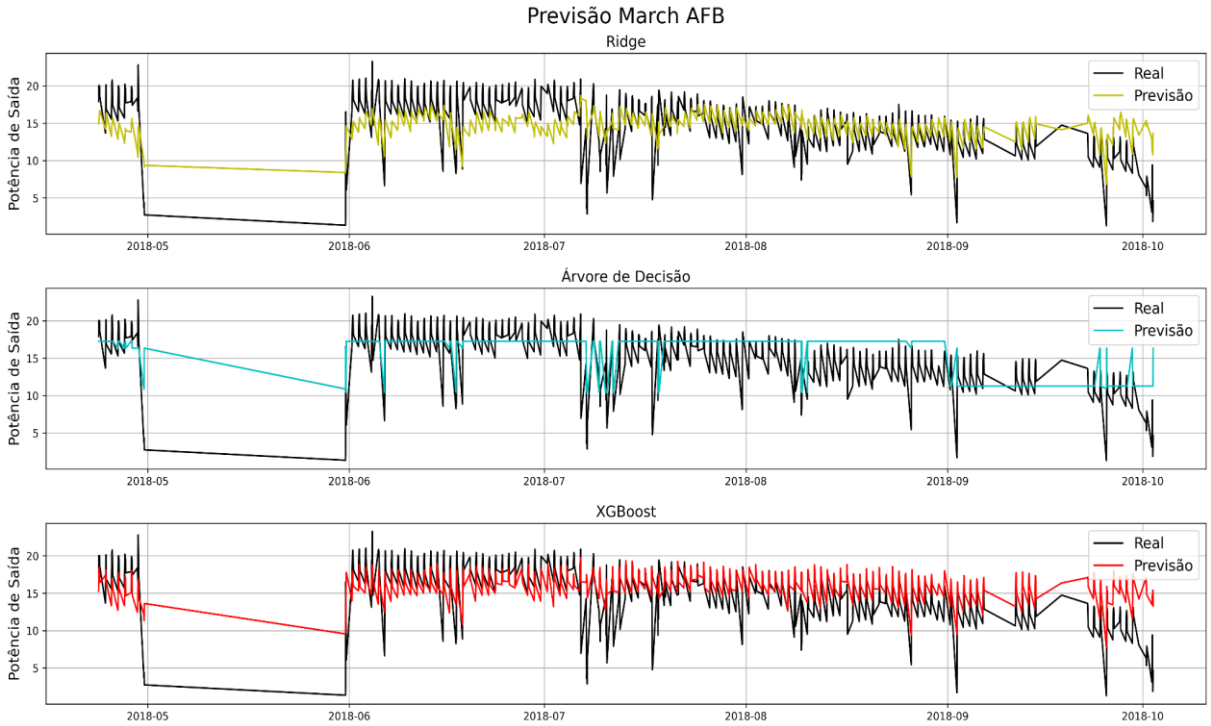
Fonte: Autor (2022).

APÊNDICE E – PREVISÃO DOS MODELOS: FIGURA DE RESULTADOS OBTIDOS PELOS TRÊS MODELOS PARA A LOCALIDADE DE MALMSTROM



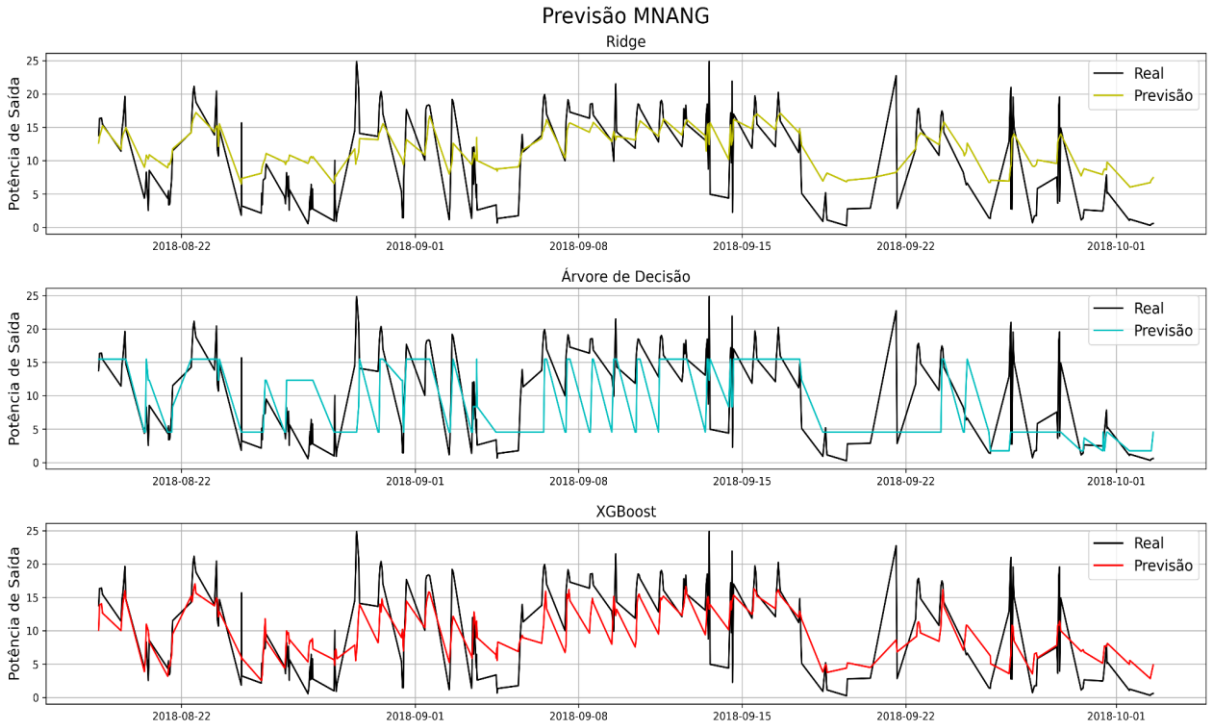
Fonte: Autor (2022).

APÊNDICE F – PREVISÃO DOS MODELOS: FIGURA DE RESULTADOS OBTIDOS PELOS TRÊS MODELOS PARA A LOCALIDADE DE MARCH AFB



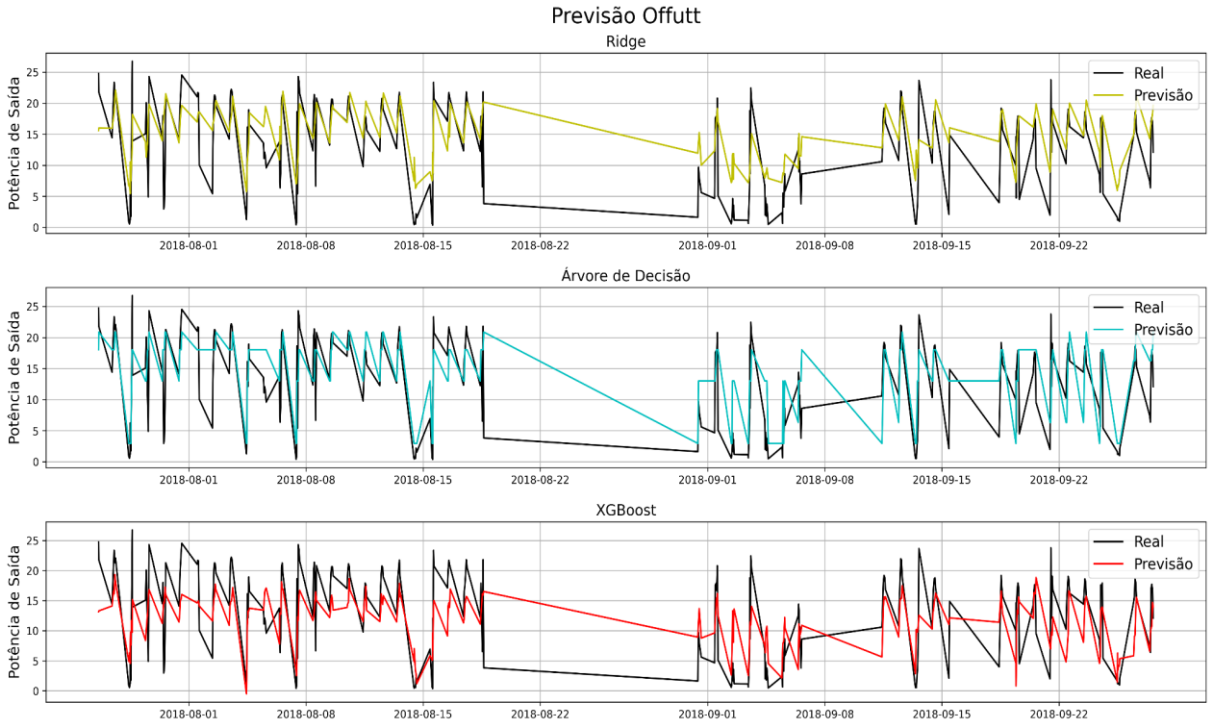
Fonte: Autor (2022).

APÊNDICE G – PREVISÃO DOS MODELOS: FIGURA DE RESULTADOS OBTIDOS PELOS TRÊS MODELOS PARA A LOCALIDADE DE MNANG



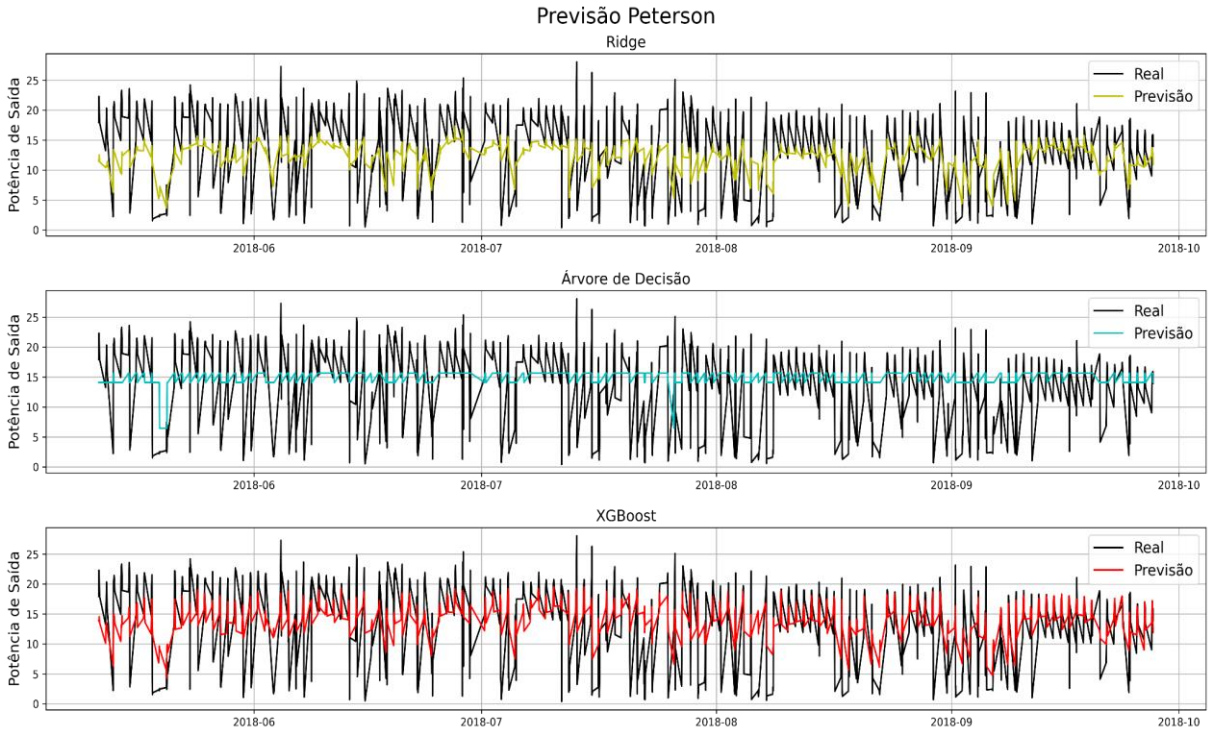
Fonte: Autor (2022).

APÊNDICE H – PREVISÃO DOS MODELOS: FIGURA DE RESULTADOS OBTIDOS PELOS TRÊS MODELOS PARA A LOCALIDADE DE OFFUTT



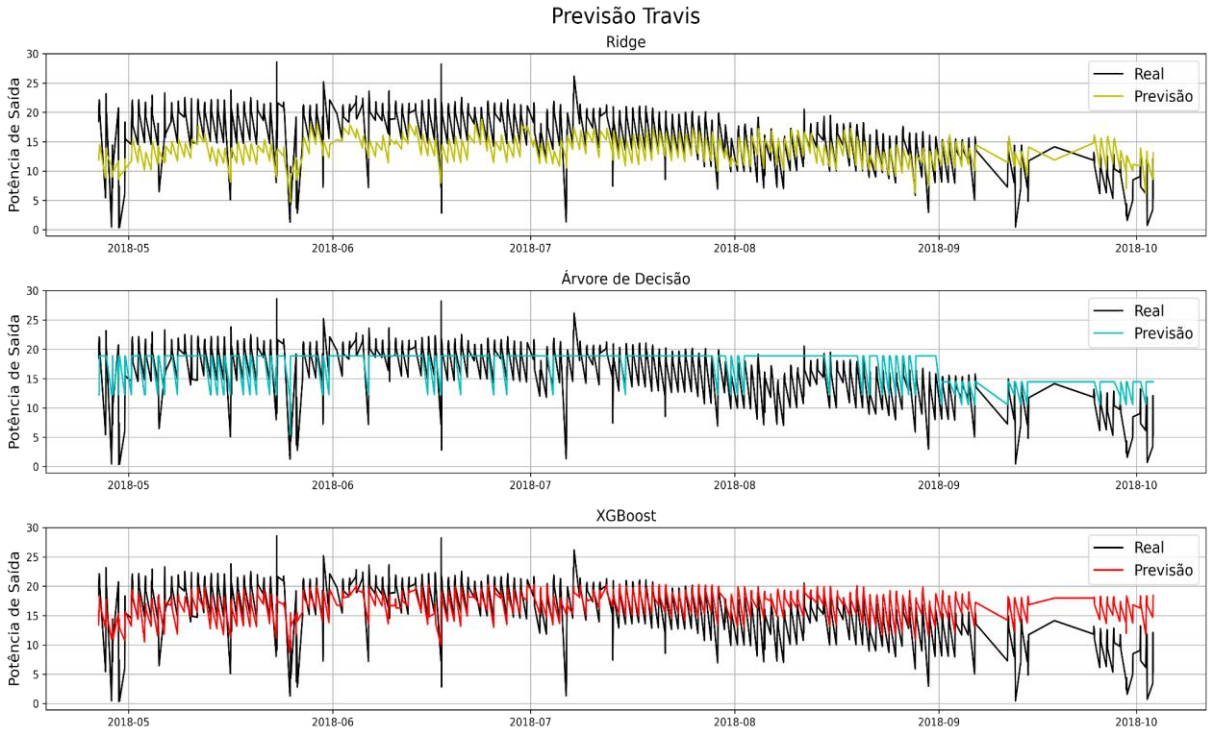
Fonte: Autor (2022).

APÊNDICE I – PREVISÃO DOS MODELOS: FIGURA DE RESULTADOS OBTIDOS PELOS TRÊS MODELOS PARA A LOCALIDADE DE PETERSON



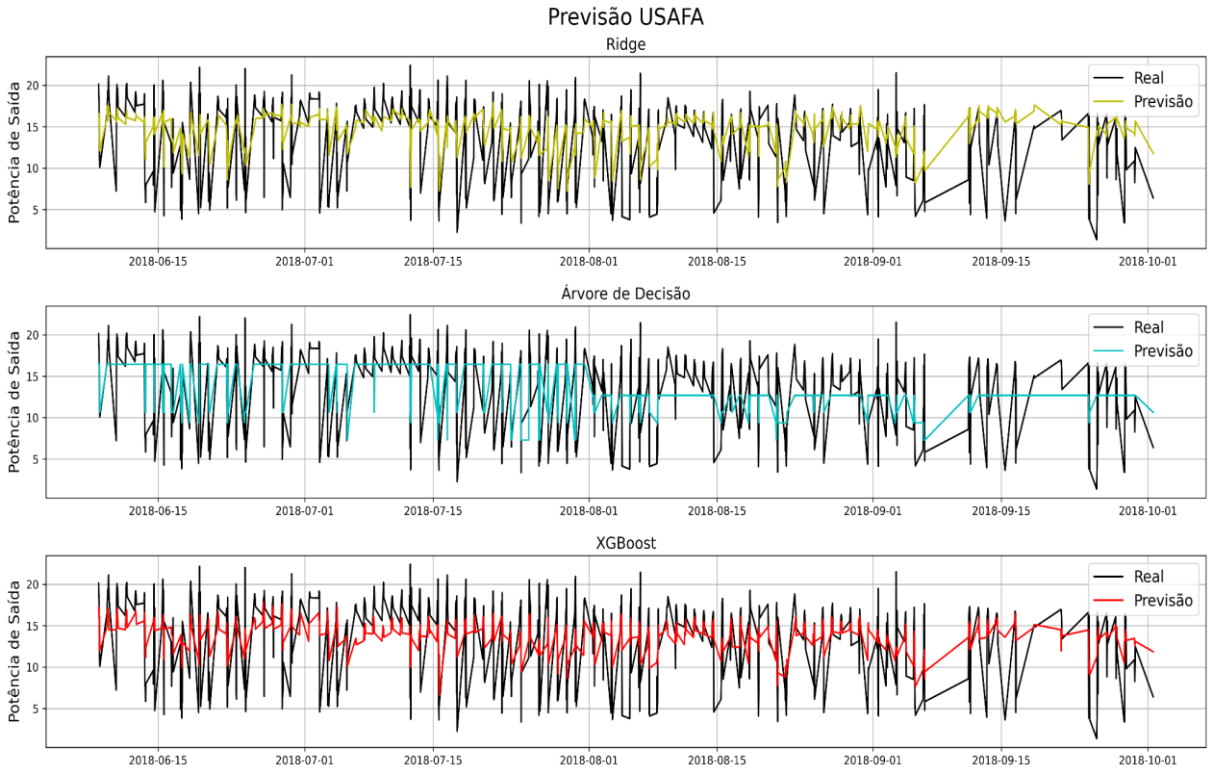
Fonte: Autor (2022).

APÊNDICE J - PREVISÃO DOS MODELOS: FIGURA DE RESULTADOS OBTIDOS PELOS TRÊS MODELOS PARA A LOCALIDADE DE TRAVIS



Fonte: Autor (2022).

APÊNDICE K – PREVISÃO DOS MODELOS: FIGURA DE RESULTADOS OBTIDOS PELOS TRÊS MODELOS PARA A LOCALIDADE DE USAFA



Fonte: Autor (2022).

APÊNDICE L – CÓDIGO DESENVOLVIDO PARA REALIZAR AS PREVISÕES

```
# Excluindo Colunas que não serão utilizadas (YRMODAHRMI  , Season)
df.drop(["YRMODAHRMI", "Season"], axis=1, inplace=True)
print(df.head(5)) # Mostrando as 5 primeiras Linhas do df Alterado
print(df.shape)   # Mostrando o tamanho do df (Linhas,Colunas)
print(df.dtypes)  # Mostrando o tipo de dado de cada Coluna

df["DateTime"] = df["Date"].astype(str) + df["Time"].astype(str)
df["DateTime"] = pd.to_datetime(df["DateTime"], format='%Y%m%d%H%M')
df.drop(["Date", "Time"], axis=1, inplace=True)
pd.get_dummies("DateTime")
print(df.head(5)) # Mostrando as 5 primeiras Linhas do df Alterado
print(df.shape)   # Mostrando o tamanho do df (Linhas,Colunas)
print(df.dtypes)  # Mostrando o tipo de dado de cada Coluna

## Dividindo em 12 grupos -> por Local
## Cada grupo tem um tamanho (quantidade de linhas diferente)

groups = df.groupby(df.Location)

Camp_Murray = groups.get_group("Camp Murray")     #1
Grissom = groups.get_group("Grissom")             #2
Hill_Weber = groups.get_group("Hill Weber")       #3
JDMT = groups.get_group("JDMT")                   #4
Kahului = groups.get_group("Kahului")             #5
Malmstrom = groups.get_group("Malmstrom")         #6
March_AFB = groups.get_group("March AFB")         #7
MNANG = groups.get_group("MNANG")                 #8
Offutt = groups.get_group("Offutt")               #9
Peterson = groups.get_group("Peterson")           #10
Travis = groups.get_group("Travis")               #11
USAFA = groups.get_group("USAFA")                 #12

print("Camp Murray shape => ", Camp_Murray.shape)
print("Grissom shape => ", Grissom.shape)
print("Hill Weber shape => ", Hill_Weber.shape)
print("JDMT shape => ", JDMT.shape)
print("Kahului shape => ", Kahului.shape)
print("Malmstrom shape => ", Malmstrom.shape)
print("March_AFB shape => ", March_AFB.shape)
print("MNANG shape => ", MNANG.shape)
print("Offutt shape => ", Offutt.shape)
print("Peterson shape => ", Peterson.shape)
print("Travis shape => ", Travis.shape)
```

```
print("USAFA shape => ", USAFA.shape)
## Dividindo os grupos
## Camp Murray  1

features = ['Latitude', 'Longitude',  'Altitude', 'Month', 'Hour',  'Humidity',
'AmbientTemp','Wind.Speed', 'Visibility', 'Pressure', 'Cloud.Ceiling', 'DateTime']
X1 = Camp_Murray[features]
y1 = np.array(Camp_Murray['PolyPwr']).reshape(-1, 1)

X1_train, X1_test, y1_train, y1_test = train_test_split(X1, y1, test_size=0.3,
shuffle=False) # 70% treino e 30% teste

date1 = Camp_Murray['DateTime']
date1_train, date1_test = train_test_split(date1, test_size=0.3, shuffle=False)

# Árvore Grissom  2
X2 = Grissom[features]
y2 = np.array(Grissom['PolyPwr']).reshape(-1, 1)

X2_train, X2_test, y2_train, y2_test = train_test_split(X2, y2, test_size=0.3,
shuffle=False) # 70% treino e 30% teste

date2 = Grissom['DateTime']
date2_train, date2_test = train_test_split(date2, test_size=0.3, shuffle=False)

# Árvore Hill_Weber  3
X3 = Hill_Weber[features]
y3 = np.array(Hill_Weber['PolyPwr']).reshape(-1, 1)

X3_train, X3_test, y3_train, y3_test = train_test_split(X3, y3, test_size=0.3,
shuffle=False) # 70% treino e 30% teste

date3 = Hill_Weber['DateTime']
date3_train, date3_test = train_test_split(date3, test_size=0.3, shuffle=False)

# Árvore JDMT  4
X4 = JDMT[features]
y4 = np.array(JDMT['PolyPwr']).reshape(-1, 1)

X4_train, X4_test, y4_train, y4_test = train_test_split(X4, y4, test_size=0.3,
shuffle=False) # 70% treino e 30% teste

date4 = JDMT['DateTime']
date4_train, date4_test = train_test_split(date4, test_size=0.3, shuffle=False)

# Árvore Kahului  5
```

```
X5 = Kahului[features]
y5 = np.array(Kahului['PolyPwr']).reshape(-1, 1)

X5_train, X5_test, y5_train, y5_test = train_test_split(X5, y5, test_size=0.3,
shuffle=False) # 70% treino e 30% teste

date5 = Kahului['DateTime']
date5_train, date5_test = train_test_split(date5, test_size=0.3, shuffle=False)

# Árvore Malmstrom  6
X6 = Malmstrom[features]
y6 = np.array(Malmstrom['PolyPwr']).reshape(-1, 1)

X6_train, X6_test, y6_train, y6_test = train_test_split(X6, y6, test_size=0.3,
shuffle=False) # 70% treino e 30% teste

date6 = Malmstrom['DateTime']
date6_train, date6_test = train_test_split(date6, test_size=0.3, shuffle=False)

# Árvore March_AFB  7
X7 = March_AFB[features]
y7 = np.array(March_AFB['PolyPwr']).reshape(-1, 1)

X7_train, X7_test, y7_train, y7_test = train_test_split(X7, y7, test_size=0.3,
shuffle=False) # 70% treino e 30% teste

date7 = March_AFB['DateTime']
date7_train, date7_test = train_test_split(date7, test_size=0.3, shuffle=False)

# Árvore MNANG  8
X8 = MNANG[features]
y8 = np.array(MNANG['PolyPwr']).reshape(-1, 1)

X8_train, X8_test, y8_train, y8_test = train_test_split(X8, y8, test_size=0.3,
shuffle=False) # 70% treino e 30% teste

date8 = MNANG['DateTime']
date8_train, date8_test = train_test_split(date8, test_size=0.3, shuffle=False)

# Árvore Offutt  9
X9 = Offutt[features]
y9 = np.array(Offutt['PolyPwr']).reshape(-1, 1)

X9_train, X9_test, y9_train, y9_test = train_test_split(X9, y9, test_size=0.3,
shuffle=False) # 70% treino e 30% teste
```

```
date9 = Offutt['DateTime']
date9_train, date9_test = train_test_split(date9, test_size=0.3, shuffle=False)

# Árvore Peterson  10
X10 = Peterson[features]
y10 = np.array(Peterson['PolyPwr']).reshape(-1, 1)

X10_train, X10_test, y10_train, y10_test = train_test_split(X10, y10,
test_size=0.3, shuffle=False) # 70% treino e 30% teste

date10 = Peterson['DateTime']
date10_train, date10_test = train_test_split(date10, test_size=0.3, shuffle=False)

# Árvore Travis  11
X11 = Travis[features]
y11 = np.array(Travis['PolyPwr']).reshape(-1, 1)

X11_train, X11_test, y11_train, y11_test = train_test_split(X11, y11,
test_size=0.3, shuffle=False) # 70% treino e 30% teste

date11 = Travis['DateTime']
date11_train, date11_test = train_test_split(date11, test_size=0.3, shuffle=False)

# Árvore USAFA  12
X12 = USAFA[features]
y12 = np.array(USAFA['PolyPwr']).reshape(-1, 1)

X12_train, X12_test, y12_train, y12_test = train_test_split(X12, y12,
test_size=0.3, shuffle=False) # 70% treino e 30% teste

date12 = USAFA['DateTime']
date12_train, date12_test = train_test_split(date12, test_size=0.3, shuffle=False)


## Camp Murray  1

# Regressão Linear (Ridge)
sklearn.linear_model.Ridge = Ridge
linear_reg1 = Ridge(normalize=True, alpha=0.5)
X1_train['DateTime'] = X1_train['DateTime'].values.astype(float)
X1_test['DateTime'] = X1_test['DateTime'].values.astype(float)
linear_reg1.fit(X1_train, y1_train)
y1_pred_linear = linear_reg1.predict(X1_test)
```

```
# Árvore de decisão
clf1 = tree.DecisionTreeRegressor(max_depth=3)
clf1.fit(X1_train,y1_train)

y1_pred_clf = clf1.predict(X1_test)

# XGBoost
XGB1 = XGBRegressor(booster = "gbtree", eta = 0.01, max_depth=3, colsample_bytree =
0.1, gamma = 0.5)
XGB1.fit(X1_train, y1_train, verbose=False)

y1_pred_XGB = XGB1.predict(X1_test)

# PLOT

plt.figure(figsize=(20,10), dpi=300)
plt.suptitle('Previsão Camp Murray', fontsize=20)


# linear
plt.subplot(311)
plt.plot(date1_test, y1_test, color='k')
plt.plot(date1_test, y1_pred_linear,  color='y')
plt.ylabel('Potência de Saída', fontsize=14)
plt.title('Ridge', fontsize=14)
plt.grid(True)
plt.legend(['Real', 'Previsão'], fontsize=14, loc=1)

# árvore
plt.subplot(312)
plt.plot(date1_test, y1_test, color='k')
plt.plot(date1_test, y1_pred_clf, color='c')
plt.ylabel('Potência de Saída', fontsize=14)
plt.title('Árvore de Decisão', fontsize=14)
plt.grid(True)
plt.legend(['Real', 'Previsão'], fontsize=14, loc=1)

# XGBoost
plt.subplot(313)
plt.plot(date1_test, y1_test, color='k')
plt.plot(date1_test, y1_pred_XGB,  color='r')
plt.ylabel('Potência de Saída', fontsize=14)
plt.title('XGBoost', fontsize=14)
plt.grid(True)
plt.legend(['Real', 'Previsão'], fontsize=14, loc=1)
```

```
plt.subplots_adjust(top=0.92, bottom=0.08, left=0.1, right=0.9, hspace=0.3,
wspace=0.1)

plt.show()

print("RL Erro Médio Absoluto = ", (mean_absolute_error(y1_test, y1_pred_linear)))
print("RL Erro Quadrado Médio = ", mean_squared_error(y1_test, y1_pred_linear))
print("RL R² = ", r2_score(y1_test, y1_pred_linear)),
print("RL Median absolute error = ", metrics.median_absolute_error(y1_test,
y1_pred_linear))

print("CLF Erro Médio Absoluto = ", (mean_absolute_error(y1_test, y1_pred_clf)))
print("CLF Erro Quadrado Médio = ", mean_squared_error(y1_test, y1_pred_clf))
print("CLF R² = ", r2_score(y1_test, y1_pred_clf)),
print("CLF Median absolute error = ", metrics.median_absolute_error(y1_test,
y1_pred_clf))

print("XGB Erro Médio Absoluto = ", (mean_absolute_error(y1_test, y1_pred_XGB)))
print("XGB Erro Quadrado Médio = ", mean_squared_error(y1_test, y1_pred_XGB))
print("XGB R² = ", r2_score(y1_test, y1_pred_XGB)),
print("XGB Median absolute error = ", metrics.median_absolute_error(y1_test,
y1_pred_XGB))

plt.figure(figsize=(20,5), dpi=300)
plt.grid(True)
plt.suptitle('Localidade de Camp Murray', fontsize=20)
plt.ylabel('Potência de Saída', fontsize=14)
plt.xlabel('Data', fontsize=14)
plt.plot(date1, Camp_Murray['PolyPwr'])
plt.plot(date1_test, y1_pred_XGB)
plt.legend(['Real', 'Previsão XGBoost'], fontsize=14, loc=1)
plt.show()

## Grissom  2

# Regressão Linear (Ridge)
sklearn.linear_model.Ridge = Ridge
linear_reg2 = Ridge(normalize=True, alpha=0.5)
X2_train['DateTime'] = X2_train['DateTime'].values.astype(float)
X2_test['DateTime'] = X2_test['DateTime'].values.astype(float)
linear_reg2.fit(X2_train, y2_train)
y2_pred_linear = linear_reg2.predict(X2_test)


# Árvore de decisão
clf2 = tree.DecisionTreeRegressor(max_depth=3)
```

```
clf2.fit(X2_train,y2_train)

y2_pred_clf = clf2.predict(X2_test)

# XGBoost
XGB2 = XGBRegressor(booster = "gbtree", eta = 0.01, max_depth=3, colsample_bytree =
0.1, gamma = 0.5)
XGB2.fit(X2_train, y2_train, verbose=False)

y2_pred_XGB = XGB2.predict(X2_test)

# PLOT

plt.figure(figsize=(20,15), dpi=300)
plt.suptitle('Previsão Grissom', fontsize=20)


# linear
plt.subplot(311)
plt.plot(date2_test, y2_test, color='k')
plt.plot(date2_test, y2_pred_linear,  color='y')
plt.ylabel('Potência de Saída', fontsize=14)
plt.title('Ridge', fontsize=14)
plt.grid(True)
plt.legend(['Real', 'Previsão'], fontsize=14, loc=1)

# árvore
plt.subplot(312)
plt.plot(date2_test, y2_test, color='k')
plt.plot(date2_test, y2_pred_clf, color='c')
plt.ylabel('Potência de Saída', fontsize=14)
plt.title('Árvore de Decisão', fontsize=14)
plt.grid(True)
plt.legend(['Real', 'Previsão'], fontsize=14, loc=1)

# XGBoost
plt.subplot(313)
plt.plot(date2_test, y2_test, color='k')
plt.plot(date2_test, y2_pred_XGB,  color='r')
plt.ylabel('Potência de Saída', fontsize=14)
plt.title('XGBoost', fontsize=14)
plt.grid(True)
plt.legend(['Real', 'Previsão'], fontsize=14, loc=1)
plt.subplots_adjust(top=0.92, bottom=0.08, left=0.1, right=0.9, hspace=0.3,
wspace=0.1)
```

```
plt.show()

print("RL Erro Médio Absoluto = ", (mean_absolute_error(y2_test, y2_pred_linear)))
print("RL Erro Quadrado Médio = ", mean_squared_error(y2_test, y2_pred_linear))
print("RL R² = ", r2_score(y2_test, y2_pred_linear)),
print("RL Median absolute error = ", metrics.median_absolute_error(y2_test,
y2_pred_linear))

print("CLF Erro Médio Absoluto = ", (mean_absolute_error(y2_test, y2_pred_clf)))
print("CLF Erro Quadrado Médio = ", mean_squared_error(y2_test, y2_pred_clf))
print("CLF R² = ", r2_score(y2_test, y2_pred_clf)),
print("CLF Median absolute error = ", metrics.median_absolute_error(y2_test,
y2_pred_clf))

print("XGB Erro Médio Absoluto = ", (mean_absolute_error(y2_test, y2_pred_XGB)))
print("XGB Erro Quadrado Médio = ", mean_squared_error(y2_test, y2_pred_XGB))
print("XGB R² = ", r2_score(y2_test, y2_pred_XGB)),
print("XGB Median absolute error = ", metrics.median_absolute_error(y2_test,
y2_pred_XGB))


## Hill_Weber  3

# Regressão Linear (Ridge)
sklearn.linear_model.Ridge = Ridge
linear_reg3 = Ridge(normalize=True, alpha=0.5)
X3_train['DateTime'] = X3_train['DateTime'].values.astype(float)
X3_test['DateTime'] = X3_test['DateTime'].values.astype(float)
linear_reg3.fit(X3_train, y3_train)
y3_pred_linear = linear_reg3.predict(X3_test)


# Árvore de decisão
clf3 = tree.DecisionTreeRegressor(max_depth=3)
clf3.fit(X3_train,y3_train)

y3_pred_clf = clf3.predict(X3_test)

# XGBoost
XGB3 = XGBRegressor(booster = "gbtree", eta = 0.01, max_depth=3, colsample_bytree =
0.1, gamma = 0.5)
XGB3.fit(X3_train, y3_train, verbose=False)

y3_pred_XGB = XGB3.predict(X3_test)

# PLOT
```

```
plt.figure(figsize=(20,10), dpi=300)
plt.suptitle('Previsão Hill Weber ', fontsize=20)


# linear
plt.subplot(311)
plt.plot(date3_test, y3_test, color='k')
plt.plot(date3_test, y3_pred_linear,  color='y')
plt.ylabel('Potência de Saída', fontsize=14)
plt.title('Ridge', fontsize=14)
plt.grid(True)
plt.legend(['Real', 'Previsão'], fontsize=14, loc=1)

# árvore
plt.subplot(312)
plt.plot(date3_test, y3_test, color='k')
plt.plot(date3_test, y3_pred_clf, color='c')
plt.ylabel('Potência de Saída', fontsize=14)
plt.title('Árvore de Decisão', fontsize=14)
plt.grid(True)
plt.legend(['Real', 'Previsão'], fontsize=14, loc=1)

# XGBoost
plt.subplot(313)
plt.plot(date3_test, y3_test, color='k')
plt.plot(date3_test, y3_pred_XGB,  color='r')
plt.ylabel('Potência de Saída', fontsize=14)
plt.title('XGBoost', fontsize=14)
plt.grid(True)
plt.legend(['Real', 'Previsão'], fontsize=14, loc=1)
plt.subplots_adjust(top=0.92, bottom=0.08, left=0.1, right=0.9, hspace=0.3,
wspace=0.1)

plt.show()

print("RL Erro Médio Absoluto = ", (mean_absolute_error(y3_test, y3_pred_linear)))
print("RL Erro Quadrado Médio = ", mean_squared_error(y3_test, y3_pred_linear))
print("RL R² = ", r2_score(y3_test, y3_pred_linear)),
print("RL Median absolute error = ", metrics.median_absolute_error(y3_test,
y3_pred_linear))

print("CLF Erro Médio Absoluto = ", (mean_absolute_error(y3_test, y3_pred_clf)))
print("CLF Erro Quadrado Médio = ", mean_squared_error(y3_test, y3_pred_clf))
print("CLF R² = ", r2_score(y3_test, y3_pred_clf)),
```

```
print("CLF Median absolute error = ", metrics.median_absolute_error(y3_test,
y3_pred_clf))

print("XGB Erro Médio Absoluto = ", (mean_absolute_error(y3_test, y3_pred_XGB)))
print("XGB Erro Quadrado Médio = ", mean_squared_error(y3_test, y3_pred_XGB))
print("XGB R² = ", r2_score(y3_test, y3_pred_XGB)),
print("XGB Median absolute error = ", metrics.median_absolute_error(y3_test,
y3_pred_XGB))


## JDMT  4

# Regressão Linear (Ridge)
sklearn.linear_model.Ridge = Ridge
linear_reg4 = Ridge(normalize=True, alpha=0.5)
X4_train['DateTime'] = X4_train['DateTime'].values.astype(float)
X4_test['DateTime'] = X4_test['DateTime'].values.astype(float)
linear_reg4.fit(X4_train, y4_train)
y4_pred_linear = linear_reg4.predict(X4_test)


# Árvore de decisão
clf4 = tree.DecisionTreeRegressor(max_depth=3)
clf4.fit(X4_train,y4_train)

y4_pred_clf = clf4.predict(X4_test)

# XGBoost
XGB4 = XGBRegressor(booster = "gbtree", eta = 0.01, max_depth=3, colsample_bytree =
0.1, gamma = 0.5)
XGB4.fit(X4_train, y4_train, verbose=False)

y4_pred_XGB = XGB4.predict(X4_test)

# PLOT

plt.figure(figsize=(20,10), dpi=300)
plt.suptitle('Previsão JDMT', fontsize=20)


# linear
plt.subplot(311)
plt.plot(date4_test, y4_test, color='k')
plt.plot(date4_test, y4_pred_linear,  color='y')
plt.ylabel('Potência de Saída', fontsize=14)
plt.title('Ridge', fontsize=14)
```

```
plt.grid(True)
plt.legend(['Real', 'Previsão'], fontsize=14, loc=1)

# árvore
plt.subplot(312)
plt.plot(date4_test, y4_test, color='k')
plt.plot(date4_test, y4_pred_clf, color='c')
plt.ylabel('Potência de Saída', fontsize=14)
plt.title('Árvore de Decisão', fontsize=14)
plt.grid(True)
plt.legend(['Real', 'Previsão'], fontsize=14, loc=1)

# XGBoost
plt.subplot(313)
plt.plot(date4_test, y4_test, color='k')
plt.plot(date4_test, y4_pred_XGB,  color='r')
plt.ylabel('Potência de Saída', fontsize=14)
plt.title('XGBoost', fontsize=14)
plt.grid(True)
plt.legend(['Real', 'Previsão'], fontsize=14, loc=1)
plt.subplots_adjust(top=0.92, bottom=0.08, left=0.1, right=0.9, hspace=0.3,
wspace=0.1)

plt.show()

print("RL Erro Médio Absoluto = ", (mean_absolute_error(y4_test, y4_pred_linear)))
print("RL Erro Quadrado Médio = ", mean_squared_error(y4_test, y4_pred_linear))
print("RL R² = ", r2_score(y4_test, y4_pred_linear)),
print("RL Median absolute error = ", metrics.median_absolute_error(y4_test,
y4_pred_linear))

print("CLF Erro Médio Absoluto = ", (mean_absolute_error(y4_test, y4_pred_clf)))
print("CLF Erro Quadrado Médio = ", mean_squared_error(y4_test, y4_pred_clf))
print("CLF R² = ", r2_score(y4_test, y4_pred_clf)),
print("CLF Median absolute error = ", metrics.median_absolute_error(y4_test,
y4_pred_clf))

print("XGB Erro Médio Absoluto = ", (mean_absolute_error(y4_test, y4_pred_XGB)))
print("XGB Erro Quadrado Médio = ", mean_squared_error(y4_test, y4_pred_XGB))
print("XGB R² = ", r2_score(y4_test, y4_pred_XGB)),
print("XGB Median absolute error = ", metrics.median_absolute_error(y4_test,
y4_pred_XGB))


## Kahului  5
```

```
# Regressão Linear (Ridge)
sklearn.linear_model.Ridge = Ridge
linear_reg5 = Ridge(normalize=True, alpha=0.5)
X5_train['DateTime'] = X5_train['DateTime'].values.astype(float)
X5_test['DateTime'] = X5_test['DateTime'].values.astype(float)
linear_reg5.fit(X5_train, y5_train)
y5_pred_linear = linear_reg5.predict(X5_test)


# Árvore de decisão
clf5 = tree.DecisionTreeRegressor(max_depth=3)
clf5.fit(X5_train,y5_train)

y5_pred_clf = clf5.predict(X5_test)

# XGBoost
XGB5 = XGBRegressor(booster = "gbtree", eta = 0.01, max_depth=3, colsample_bytree =
0.1, gamma = 0.5)
XGB5.fit(X5_train, y5_train, verbose=False)

y5_pred_XGB = XGB5.predict(X5_test)

# PLOT

plt.figure(figsize=(20,10), dpi=300)
plt.suptitle('Previsão Kahului', fontsize=20)


# linear
plt.subplot(311)
plt.plot(date5_test, y5_test, color='k')
plt.plot(date5_test, y5_pred_linear,  color='y')
plt.ylabel('Potência de Saída', fontsize=14)
plt.title('Ridge', fontsize=14)
plt.grid(True)
plt.legend(['Real', 'Previsão'], fontsize=14, loc=1)

# árvore
plt.subplot(312)
plt.plot(date5_test, y5_test, color='k')
plt.plot(date5_test, y5_pred_clf, color='c')
plt.ylabel('Potência de Saída', fontsize=14)
plt.title('Árvore de Decisão', fontsize=14)
plt.grid(True)
plt.legend(['Real', 'Previsão'], fontsize=14, loc=1)
```

```
# XGBoost
plt.subplot(313)
plt.plot(date5_test, y5_test, color='k')
plt.plot(date5_test, y5_pred_XGB,  color='r')
plt.ylabel('Potência de Saída', fontsize=14)
plt.title('XGBoost', fontsize=14)
plt.grid(True)
plt.legend(['Real', 'Previsão'], fontsize=14, loc=1)
plt.subplots_adjust(top=0.92, bottom=0.08, left=0.1, right=0.9, hspace=0.3,
wspace=0.1)

plt.show()

print("RL Erro Médio Absoluto = ", (mean_absolute_error(y5_test, y5_pred_linear)))
print("RL Erro Quadrado Médio = ", mean_squared_error(y5_test, y5_pred_linear))
print("RL R² = ", r2_score(y5_test, y5_pred_linear)),
print("RL Median absolute error = ", metrics.median_absolute_error(y5_test,
y5_pred_linear))

print("CLF Erro Médio Absoluto = ", (mean_absolute_error(y5_test, y5_pred_clf)))
print("CLF Erro Quadrado Médio = ", mean_squared_error(y5_test, y5_pred_clf))
print("CLF R² = ", r2_score(y5_test, y5_pred_clf)),
print("CLF Median absolute error = ", metrics.median_absolute_error(y5_test,
y5_pred_clf))

print("XGB Erro Médio Absoluto = ", (mean_absolute_error(y5_test, y5_pred_XGB)))
print("XGB Erro Quadrado Médio = ", mean_squared_error(y5_test, y5_pred_XGB))
print("XGB R² = ", r2_score(y5_test, y5_pred_XGB)),
print("XGB Median absolute error = ", metrics.median_absolute_error(y5_test,
y5_pred_XGB))

## Malmstrom 6

# Regressão Linear (Ridge)
sklearn.linear_model.Ridge = Ridge
linear_reg6 = Ridge(normalize=True, alpha=0.5)
X6_train['DateTime'] = X6_train['DateTime'].values.astype(float)
X6_test['DateTime'] = X6_test['DateTime'].values.astype(float)
linear_reg6.fit(X6_train, y6_train)
y6_pred_linear = linear_reg6.predict(X6_test)


# Árvore de decisão
clf6 = tree.DecisionTreeRegressor(max_depth=3)
clf6.fit(X6_train,y6_train)
```

```
y6_pred_clf = clf6.predict(X6_test)

# XGBoost
XGB6 = XGBRegressor(booster = "gbtree", eta = 0.01, max_depth=3, colsample_bytree =
0.1, gamma = 0.5)
XGB6.fit(X6_train, y6_train, verbose=False)

y6_pred_XGB = XGB6.predict(X6_test)

# PLOT

plt.figure(figsize=(20,10), dpi=300)
plt.suptitle('Previsão Malmstrom', fontsize=20)


# linear
plt.subplot(311)
plt.plot(date6_test, y6_test, color='k')
plt.plot(date6_test, y6_pred_linear,  color='y')
plt.ylabel('Potência de Saída', fontsize=14)
plt.title('Ridge', fontsize=14)
plt.grid(True)
plt.legend(['Real', 'Previsão'], fontsize=14, loc=1)

# árvore
plt.subplot(312)
plt.plot(date6_test, y6_test, color='k')
plt.plot(date6_test, y6_pred_clf, color='c')
plt.ylabel('Potência de Saída', fontsize=14)
plt.title('Árvore de Decisão', fontsize=14)
plt.grid(True)
plt.legend(['Real', 'Previsão'], fontsize=14, loc=1)

# XGBoost
plt.subplot(313)
plt.plot(date6_test, y6_test, color='k')
plt.plot(date6_test, y6_pred_XGB,  color='r')
plt.ylabel('Potência de Saída', fontsize=14)
plt.title('XGBoost', fontsize=14)
plt.grid(True)
plt.legend(['Real', 'Previsão'], fontsize=14, loc=1)
plt.subplots_adjust(top=0.92, bottom=0.08, left=0.1, right=0.9, hspace=0.3,
wspace=0.1)

plt.show()
```

```
print("RL Erro Médio Absoluto = ", (mean_absolute_error(y6_test, y6_pred_linear)))
print("RL Erro Quadrado Médio = ", mean_squared_error(y6_test, y6_pred_linear))
print("RL R² = ", r2_score(y6_test, y6_pred_linear)),
print("RL Median absolute error = ", metrics.median_absolute_error(y6_test,
y6_pred_linear))

print("CLF Erro Médio Absoluto = ", (mean_absolute_error(y6_test, y6_pred_clf)))
print("CLF Erro Quadrado Médio = ", mean_squared_error(y6_test, y6_pred_clf))
print("CLF R² = ", r2_score(y6_test, y6_pred_clf)),
print("CLF Median absolute error = ", metrics.median_absolute_error(y6_test,
y6_pred_clf))

print("XGB Erro Médio Absoluto = ", (mean_absolute_error(y6_test, y6_pred_XGB)))
print("XGB Erro Quadrado Médio = ", mean_squared_error(y6_test, y6_pred_XGB))
print("XGB R² = ", r2_score(y6_test, y6_pred_XGB)),
print("XGB Median absolute error = ", metrics.median_absolute_error(y6_test,
y6_pred_XGB))


## March AFB 7

# Regressão Linear (Ridge)
sklearn.linear_model.Ridge = Ridge
linear_reg7 = Ridge(normalize=True, alpha=0.5)
X7_train['DateTime'] = X7_train['DateTime'].values.astype(float)
X7_test['DateTime'] = X7_test['DateTime'].values.astype(float)
linear_reg7.fit(X7_train, y7_train)
y7_pred_linear = linear_reg7.predict(X7_test)


# Árvore de decisão
clf7 = tree.DecisionTreeRegressor(max_depth=3)
clf7.fit(X7_train,y7_train)

y7_pred_clf = clf7.predict(X7_test)

# XGBoost
XGB7 = XGBRegressor(booster = "gbtree", eta = 0.01, max_depth=3, colsample_bytree =
0.1, gamma = 0.5)
XGB7.fit(X7_train, y7_train, verbose=False)

y7_pred_XGB = XGB7.predict(X7_test)

# PLOT

plt.figure(figsize=(20,10), dpi=300)
```

```
plt.suptitle('Previsão March AFB', fontsize=20)


# linear
plt.subplot(311)
plt.plot(date7_test, y7_test, color='k')
plt.plot(date7_test, y7_pred_linear,  color='y')
plt.ylabel('Potência de Saída', fontsize=14)
plt.title('Ridge', fontsize=14)
plt.grid(True)
plt.legend(['Real', 'Previsão'], fontsize=14, loc=1)

# árvore
plt.subplot(312)
plt.plot(date7_test, y7_test, color='k')
plt.plot(date7_test, y7_pred_clf, color='c')
plt.ylabel('Potência de Saída', fontsize=14)
plt.title('Árvore de Decisão', fontsize=14)
plt.grid(True)
plt.legend(['Real', 'Previsão'], fontsize=14, loc=1)

# XGBoost
plt.subplot(313)
plt.plot(date7_test, y7_test, color='k')
plt.plot(date7_test, y7_pred_XGB,  color='r')
plt.ylabel('Potência de Saída', fontsize=14)
plt.title('XGBoost', fontsize=14)
plt.grid(True)
plt.legend(['Real', 'Previsão'], fontsize=14, loc=1)
plt.subplots_adjust(top=0.92, bottom=0.08, left=0.1, right=0.9, hspace=0.3,
wspace=0.1)

plt.show()

print("RL Erro Médio Absoluto = ", (mean_absolute_error(y7_test, y7_pred_linear)))
print("RL Erro Quadrado Médio = ", mean_squared_error(y7_test, y7_pred_linear))
print("RL R² = ", r2_score(y7_test, y7_pred_linear)),
print("RL Median absolute error = ", metrics.median_absolute_error(y7_test,
y7_pred_linear))

print("CLF Erro Médio Absoluto = ", (mean_absolute_error(y7_test, y7_pred_clf)))
print("CLF Erro Quadrado Médio = ", mean_squared_error(y7_test, y7_pred_clf))
print("CLF R² = ", r2_score(y7_test, y7_pred_clf)),
print("CLF Median absolute error = ", metrics.median_absolute_error(y7_test,
y7_pred_clf))
```

```
print("XGB Erro Médio Absoluto = ", (mean_absolute_error(y7_test, y7_pred_XGB)))
print("XGB Erro Quadrado Médio = ", mean_squared_error(y7_test, y7_pred_XGB))
print("XGB R² = ", r2_score(y7_test, y7_pred_XGB)),
print("XGB Median absolute error = ", metrics.median_absolute_error(y7_test,
y7_pred_XGB))

## March MNANG  8

# Regressão Linear (Ridge)
sklearn.linear_model.Ridge = Ridge
linear_reg8 = Ridge(normalize=True, alpha=0.5)
X8_train['DateTime'] = X8_train['DateTime'].values.astype(float)
X8_test['DateTime'] = X8_test['DateTime'].values.astype(float)
linear_reg8.fit(X8_train, y8_train)
y8_pred_linear = linear_reg8.predict(X8_test)


# Árvore de decisão
clf8 = tree.DecisionTreeRegressor(max_depth=3)
clf8.fit(X8_train,y8_train)

y8_pred_clf = clf8.predict(X8_test)

# XGBoost
XGB8 = XGBRegressor(booster = "gbtree", eta = 0.01, max_depth=3, colsample_bytree =
0.1, gamma = 0.5)
XGB8.fit(X8_train, y8_train, verbose=False)

y8_pred_XGB = XGB8.predict(X8_test)

# PLOT

plt.figure(figsize=(20,10), dpi=300)
plt.suptitle('Previsão MNANG', fontsize=20)


# linear
plt.subplot(311)
plt.plot(date8_test, y8_test, color='k')
plt.plot(date8_test, y8_pred_linear,  color='y')
plt.ylabel('Potência de Saída', fontsize=14)
plt.title('Ridge', fontsize=14)
plt.grid(True)
plt.legend(['Real', 'Previsão'], fontsize=14, loc=1)

# árvore
```

```
plt.subplot(312)
plt.plot(date8_test, y8_test, color='k')
plt.plot(date8_test, y8_pred_clf, color='c')
plt.ylabel('Potência de Saída', fontsize=14)
plt.title('Árvore de Decisão', fontsize=14)
plt.grid(True)
plt.legend(['Real', 'Previsão'], fontsize=14, loc=1)

# XGBoost
plt.subplot(313)
plt.plot(date8_test, y8_test, color='k')
plt.plot(date8_test, y8_pred_XGB,  color='r')
plt.ylabel('Potência de Saída', fontsize=14)
plt.title('XGBoost', fontsize=14)
plt.grid(True)
plt.legend(['Real', 'Previsão'], fontsize=14, loc=1)
plt.subplots_adjust(top=0.92, bottom=0.08, left=0.1, right=0.9, hspace=0.3,
wspace=0.1)

plt.show()

print("RL Erro Médio Absoluto = ", (mean_absolute_error(y8_test, y8_pred_linear)))
print("RL Erro Quadrado Médio = ", mean_squared_error(y8_test, y8_pred_linear))
print("RL R² = ", r2_score(y8_test, y8_pred_linear)),
print("RL Median absolute error = ", metrics.median_absolute_error(y8_test,
y8_pred_linear))

print("CLF Erro Médio Absoluto = ", (mean_absolute_error(y8_test, y8_pred_clf)))
print("CLF Erro Quadrado Médio = ", mean_squared_error(y8_test, y8_pred_clf))
print("CLF R² = ", r2_score(y8_test, y8_pred_clf)),
print("CLF Median absolute error = ", metrics.median_absolute_error(y8_test,
y8_pred_clf))

print("XGB Erro Médio Absoluto = ", (mean_absolute_error(y8_test, y8_pred_XGB)))
print("XGB Erro Quadrado Médio = ", mean_squared_error(y8_test, y8_pred_XGB))
print("XGB R² = ", r2_score(y8_test, y8_pred_XGB)),
print("XGB Median absolute error = ", metrics.median_absolute_error(y8_test,
y8_pred_XGB))


## March Offutt   9

# Regressão Linear (Ridge)
sklearn.linear_model.Ridge = Ridge
linear_reg9 = Ridge(normalize=True, alpha=0.5)
X9_train['DateTime'] = X9_train['DateTime'].values.astype(float)
```

```
X9_test['DateTime'] = X9_test['DateTime'].values.astype(float)
linear_reg9.fit(X9_train, y9_train)
y9_pred_linear = linear_reg9.predict(X9_test)


# Árvore de decisão
clf9 = tree.DecisionTreeRegressor(max_depth=3)
clf9.fit(X9_train,y9_train)

y9_pred_clf = clf9.predict(X9_test)

# XGBoost
XGB9 = XGBRegressor(booster = "gbtree", eta = 0.01, max_depth=3, colsample_bytree =
0.1, gamma = 0.5)
XGB9.fit(X9_train, y9_train, verbose=False)

y9_pred_XGB = XGB9.predict(X9_test)

# PLOT

plt.figure(figsize=(20,10), dpi=300)
plt.suptitle('Previsão Offutt', fontsize=20)


# linear
plt.subplot(311)
plt.plot(date9_test, y9_test, color='k')
plt.plot(date9_test, y9_pred_linear,  color='y')
plt.ylabel('Potência de Saída', fontsize=14)
plt.title('Ridge', fontsize=14)
plt.grid(True)
plt.legend(['Real', 'Previsão'], fontsize=14, loc=1)

# árvore
plt.subplot(312)
plt.plot(date9_test, y9_test, color='k')
plt.plot(date9_test, y9_pred_clf, color='c')
plt.ylabel('Potência de Saída', fontsize=14)
plt.title('Árvore de Decisão', fontsize=14)
plt.grid(True)
plt.legend(['Real', 'Previsão'], fontsize=14, loc=1)

# XGBoost
plt.subplot(313)
plt.plot(date9_test, y9_test, color='k')
plt.plot(date9_test, y9_pred_XGB,  color='r')
```

```
plt.ylabel('Potência de Saída', fontsize=14)
plt.title('XGBoost', fontsize=14)
plt.grid(True)
plt.legend(['Real', 'Previsão'], fontsize=14, loc=1)
plt.subplots_adjust(top=0.92, bottom=0.08, left=0.1, right=0.9, hspace=0.3,
wspace=0.1)

plt.show()

print("RL Erro Médio Absoluto = ", (mean_absolute_error(y9_test, y9_pred_linear)))
print("RL Erro Quadrado Médio = ", mean_squared_error(y9_test, y9_pred_linear))
print("RL R² = ", r2_score(y9_test, y9_pred_linear)),
print("RL Median absolute error = ", metrics.median_absolute_error(y9_test,
y9_pred_linear))

print("CLF Erro Médio Absoluto = ", (mean_absolute_error(y9_test, y9_pred_clf)))
print("CLF Erro Quadrado Médio = ", mean_squared_error(y9_test, y9_pred_clf))
print("CLF R² = ", r2_score(y9_test, y9_pred_clf)),
print("CLF Median absolute error = ", metrics.median_absolute_error(y9_test,
y9_pred_clf))

print("XGB Erro Médio Absoluto = ", (mean_absolute_error(y9_test, y9_pred_XGB)))
print("XGB Erro Quadrado Médio = ", mean_squared_error(y9_test, y9_pred_XGB))
print("XGB R² = ", r2_score(y9_test, y9_pred_XGB)),
print("XGB Median absolute error = ", metrics.median_absolute_error(y9_test,
y9_pred_XGB))


## Peterson  10

# Regressão Linear (Ridge)
sklearn.linear_model.Ridge = Ridge
linear_reg10 = Ridge(normalize=True, alpha=0.5)
X10_train['DateTime'] = X10_train['DateTime'].values.astype(float)
X10_test['DateTime'] = X10_test['DateTime'].values.astype(float)
linear_reg10.fit(X10_train, y10_train)
y10_pred_linear = linear_reg10.predict(X10_test)


# Árvore de decisão
clf10 = tree.DecisionTreeRegressor(max_depth=3)
clf10.fit(X10_train,y10_train)

y10_pred_clf = clf10.predict(X10_test)

# XGBoost
```

```
XGB10 = XGBRegressor(booster = "gbtree", eta = 0.01, max_depth=3, colsample_bytree
= 0.1, gamma = 0.5)
XGB10.fit(X10_train, y10_train, verbose=False)

y10_pred_XGB = XGB10.predict(X10_test)

# PLOT

plt.figure(figsize=(20,10), dpi=300)
plt.suptitle('Previsão Peterson', fontsize=20)


# linear
plt.subplot(311)
plt.plot(date10_test, y10_test, color='k')
plt.plot(date10_test, y10_pred_linear,  color='y')
plt.ylabel('Potência de Saída', fontsize=14)
plt.title('Ridge', fontsize=14)
plt.grid(True)
plt.legend(['Real', 'Previsão'], fontsize=14, loc=1)

# árvore
plt.subplot(312)
plt.plot(date10_test, y10_test, color='k')
plt.plot(date10_test, y10_pred_clf, color='c')
plt.ylabel('Potência de Saída', fontsize=14)
plt.title('Árvore de Decisão', fontsize=14)
plt.grid(True)
plt.legend(['Real', 'Previsão'], fontsize=14, loc=1)

# XGBoost
plt.subplot(313)
plt.plot(date10_test, y10_test, color='k')
plt.plot(date10_test, y10_pred_XGB,  color='r')
plt.ylabel('Potência de Saída', fontsize=14)
plt.title('XGBoost', fontsize=14)
plt.grid(True)
plt.legend(['Real', 'Previsão'], fontsize=14, loc=1)
plt.subplots_adjust(top=0.92, bottom=0.08, left=0.1, right=0.9, hspace=0.3,
wspace=0.1)

plt.show()

print("RL Erro Médio Absoluto = ", (mean_absolute_error(y10_test,
y10_pred_linear)))
print("RL Erro Quadrado Médio = ", mean_squared_error(y10_test, y10_pred_linear))
```

```
print("RL R² = ", r2_score(y10_test, y10_pred_linear)),
print("RL Median absolute error = ", metrics.median_absolute_error(y10_test,
y10_pred_linear))

print("CLF Erro Médio Absoluto = ", (mean_absolute_error(y10_test, y10_pred_clf)))
print("CLF Erro Quadrado Médio = ", mean_squared_error(y10_test, y10_pred_clf))
print("CLF R² = ", r2_score(y10_test, y10_pred_clf)),
print("CLF Median absolute error = ", metrics.median_absolute_error(y10_test,
y10_pred_clf))

print("XGB Erro Médio Absoluto = ", (mean_absolute_error(y10_test, y10_pred_XGB)))
print("XGB Erro Quadrado Médio = ", mean_squared_error(y10_test, y10_pred_XGB))
print("XGB R² = ", r2_score(y10_test, y10_pred_XGB)),
print("XGB Median absolute error = ", metrics.median_absolute_error(y10_test,
y10_pred_XGB))

## Travis  11

# Regressão Linear (Ridge)
sklearn.linear_model.Ridge = Ridge
linear_reg11 = Ridge(normalize=True, alpha=0.5)
X11_train['DateTime'] = X11_train['DateTime'].values.astype(float)
X11_test['DateTime'] = X11_test['DateTime'].values.astype(float)
linear_reg11.fit(X11_train, y11_train)
y11_pred_linear = linear_reg11.predict(X11_test)


# Árvore de decisão
clf11 = tree.DecisionTreeRegressor(max_depth=3)
clf11.fit(X11_train,y11_train)

y11_pred_clf = clf11.predict(X11_test)

# XGBoost
XGB11 = XGBRegressor(booster = "gbtree", eta = 0.01, max_depth=3, colsample_bytree
= 0.1, gamma = 0.5)
XGB11.fit(X11_train, y11_train, verbose=False)

y11_pred_XGB = XGB11.predict(X11_test)

# PLOT

plt.figure(figsize=(20,10), dpi=300)
plt.suptitle('Previsão Travis', fontsize=20)
```

```
# linear
plt.subplot(311)
plt.plot(date11_test, y11_test, color='k')
plt.plot(date11_test, y11_pred_linear,  color='y')
plt.ylabel('Potência de Saída', fontsize=14)
plt.title('Ridge', fontsize=14)
plt.grid(True)
plt.legend(['Real', 'Previsão'], fontsize=14, loc=1)

# árvore
plt.subplot(312)
plt.plot(date11_test, y11_test, color='k')
plt.plot(date11_test, y11_pred_clf, color='c')
plt.ylabel('Potência de Saída', fontsize=14)
plt.title('Árvore de Decisão', fontsize=14)
plt.grid(True)
plt.legend(['Real', 'Previsão'], fontsize=14, loc=1)

# XGBoost
plt.subplot(313)
plt.plot(date11_test, y11_test, color='k')
plt.plot(date11_test, y11_pred_XGB,  color='r')
plt.ylabel('Potência de Saída', fontsize=14)
plt.title('XGBoost', fontsize=14)
plt.grid(True)
plt.legend(['Real', 'Previsão'], fontsize=14, loc=1)
plt.subplots_adjust(top=0.92, bottom=0.08, left=0.1, right=0.9, hspace=0.3,
wspace=0.1)

plt.show()

print("RL Erro Médio Absoluto = ", (mean_absolute_error(y11_test,
y11_pred_linear)))
print("RL Erro Quadrado Médio = ", mean_squared_error(y11_test, y11_pred_linear))
print("RL R² = ", r2_score(y11_test, y11_pred_linear)),
print("RL Median absolute error = ", metrics.median_absolute_error(y11_test,
y11_pred_linear))

print("CLF Erro Médio Absoluto = ", (mean_absolute_error(y11_test, y11_pred_clf)))
print("CLF Erro Quadrado Médio = ", mean_squared_error(y11_test, y11_pred_clf))
print("CLF R² = ", r2_score(y11_test, y11_pred_clf)),
print("CLF Median absolute error = ", metrics.median_absolute_error(y11_test,
y11_pred_clf))

print("XGB Erro Médio Absoluto = ", (mean_absolute_error(y11_test, y11_pred_XGB)))
print("XGB Erro Quadrado Médio = ", mean_squared_error(y11_test, y11_pred_XGB))
```

```
print("XGB R² = ", r2_score(y11_test, y11_pred_XGB)),
print("XGB Median absolute error = ", metrics.median_absolute_error(y11_test,
y11_pred_XGB))


## USAFA  12

# Regressão Linear (Ridge)
sklearn.linear_model.Ridge = Ridge
linear_reg12 = Ridge(normalize=True, alpha=0.5)
X12_train['DateTime'] = X12_train['DateTime'].values.astype(float)
X12_test['DateTime'] = X12_test['DateTime'].values.astype(float)
linear_reg12.fit(X12_train, y12_train)
y12_pred_linear = linear_reg12.predict(X12_test)


# Árvore de decisão
clf12 = tree.DecisionTreeRegressor(max_depth=3)
clf12.fit(X12_train,y12_train)

y12_pred_clf = clf12.predict(X12_test)

# XGBoost
XGB12 = XGBRegressor(booster = "gbtree", eta = 0.01, max_depth=3, colsample_bytree
= 0.1, gamma = 0.5)
XGB12.fit(X12_train, y12_train, verbose=False)

y12_pred_XGB = XGB12.predict(X12_test)

# PLOT

plt.figure(figsize=(20,10), dpi=300)
plt.suptitle('Previsão USAFA', fontsize=20)


# linear
plt.subplot(311)
plt.plot(date12_test, y12_test, color='k')
plt.plot(date12_test, y12_pred_linear,  color='y')
plt.ylabel('Potência de Saída', fontsize=14)
plt.title('Ridge', fontsize=14)
plt.grid(True)
plt.legend(['Real', 'Previsão'], fontsize=14, loc=1)

# árvore
plt.subplot(312)
```

```
plt.plot(date12_test, y12_test, color='k')
plt.plot(date12_test, y12_pred_clf, color='c')
plt.ylabel('Potência de Saída', fontsize=14)
plt.title('Árvore de Decisão', fontsize=14)
plt.grid(True)
plt.legend(['Real', 'Previsão'], fontsize=14, loc=1)

# XGBoost
plt.subplot(313)
plt.plot(date12_test, y12_test, color='k')
plt.plot(date12_test, y12_pred_XGB,  color='r')
plt.ylabel('Potência de Saída', fontsize=14)
plt.title('XGBoost', fontsize=14)
plt.grid(True)
plt.legend(['Real', 'Previsão'], fontsize=14, loc=1)
plt.subplots_adjust(top=0.92, bottom=0.08, left=0.1, right=0.9, hspace=0.3,
wspace=0.1)

plt.show()

print("RL Erro Médio Absoluto = ", (mean_absolute_error(y12_test,
y12_pred_linear)))
print("RL Erro Quadrado Médio = ", mean_squared_error(y12_test, y12_pred_linear))
print("RL R² = ", r2_score(y12_test, y12_pred_linear)),
print("RL Median absolute error = ", metrics.median_absolute_error(y12_test,
y12_pred_linear))

print("CLF Erro Médio Absoluto = ", (mean_absolute_error(y12_test, y12_pred_clf)))
print("CLF Erro Quadrado Médio = ", mean_squared_error(y12_test, y12_pred_clf))
print("CLF R² = ", r2_score(y12_test, y12_pred_clf)),
print("CLF Median absolute error = ", metrics.median_absolute_error(y12_test,
y12_pred_clf))

print("XGB Erro Médio Absoluto = ", (mean_absolute_error(y12_test, y12_pred_XGB)))
print("XGB Erro Quadrado Médio = ", mean_squared_error(y12_test, y12_pred_XGB))
print("XGB R² = ", r2_score(y12_test, y12_pred_XGB)),
print("XGB Median absolute error = ", metrics.median_absolute_error(y12_test,
y12_pred_XGB))
```