INSTITUTO FEDERAL DE SANTA CATARINA

YAGO CASTRO ROSA

Desenvolvimento de uma Ferramenta para Detecção e Predição de Falhas em Trackers para Usinas Fotovoltaicas de Grande Porte

São José - SC

Julho/2025

DESENVOLVIMENTO DE UMA FERRAMENTA PARA DETECÇÃO E PREDIÇÃO DE FALHAS EM TRACKERS PARA USINAS FOTOVOLTAICAS DE GRANDE PORTE

Monografia apresentada ao Curso de Engenharia de Telecomunicações do campus São José do Instituto Federal de Santa Catarina para a obtenção do diploma de Engenheiro de Telecomunicações.

Orientador: Prof. Arliones Stevert Hoeller Junior, Dr.

Coorientador: Prof. Mário de Noronha Neto, Dr.

São José - SC

Julho/2025

Yago Castro Rosa

Desenvolvimento de uma Ferramenta para Detecção e Predição de Falhas em Trackers para Usinas Fotovoltaicas de Grande Porte

São José - SC, 30 de Julho de 2025:

Documento assinado digitalmente
ARLIONES STEVERT HOELLER JUNIOR
Data: 20/08/2025 10:18:42-0300
CPF: ***.699.639-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Arliones Stevert Hoeller Junior, Dr.
Orientador
Instituto Federal de Santa Catarina

Documento assinado digitalmente
MARIO DE NORONHA NETO
Data: 29/08/2025 14:03:14-0300
CPF: ***.859.519-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Mário de Noronha Neto, Dr.
Coorientador
Instituto Federal de Santa Catarina

Documento assinado digitalmente
ODILSON TADEU VALLE
Data: 27/08/2025 10:50:46-0300
CPF: ***.485.599-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Odilson Tadeu Valle, Dr.
Instituto Federal de Santa Catarina

Documento assinado digitalmente
RAMON MAYOR MARTINS
Data: 20/08/2025 10:21:07-0300
CPF: ***.866.756-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Ramon Mayor Martins, Dr.
Instituto Federal de Santa Catarina

AGRADECIMENTOS

Gostaria de iniciar agradecendo à minha família, incluindo minha mãe, meus avós, minha tia e meu namorado, que estiveram ao meu lado desde o início desta jornada e que hoje colhem comigo os frutos dessa conquista. Sem o apoio e a presença de cada um deles, esta realização não teria sido possível.

Este trabalho foi desenvolvido no contexto do projeto de Pesquisa e Desenvolvimento realizado em parceria entre o Instituto Federal de Santa Catarina (IFSC), a Universidade Federal de Santa Catarina (UFSC) e a empresa ENGIE Brasil. Agradeço a todos os profissionais da ENGIE que sempre estiveram dispostos a colaborar e a contribuir para a obtenção dos melhores resultados. Agradeço também aos professores envolvidos, em especial ao meu orientador Arliones Stevert Hoeller Junior e ao meu coorientador Mário de Noronha Neto, cujas orientações e ensinamentos foram fundamentais para meu crescimento profissional. Aos colegas de bolsa Bruno Castro Valle, Gabriel Luiz Espíndola Pedro, João Vitor Rosa Negri e Lucas Coelho Raupp, deixo meu sincero agradecimento pelo apoio e companheirismo ao longo do projeto.

Por fim, agradeço aos professores e demais servidores do IFSC Campus São José que fizeram parte da minha formação durante os cinco anos e meio de graduação e os quatro anos de ensino médio, dos quais guardo profunda admiração.

RESUMO

Este trabalho apresenta o desenvolvimento de uma ferramenta baseada em aprendizado de máquina para detecção e predição de falhas em trackers de usinas fotovoltaicas. A metodologia combina análise física, por meio da comparação entre ângulos reais e teóricos, com técnicas de *machine learning*. A detecção de anomalias permitiu classificar os trackers como saudáveis ou falhos, enquanto os modelos preditivos, um autoencoder não supervisionado e dois modelos XGBoost (classificador e regressor), anteciparam falhas com até dois dias de antecedência. Os melhores resultados foram obtidos para o horizonte de um dia utilizando o modelo regressor, com revocação de 73% e precisão de 80%. A solução inclui uma interface interativa em Streamlit para visualização e acompanhamento das falhas, contribuindo para um monitoramento proativo e eficiente da operação. Como diferencial, destaca-se a integração de abordagens supervisionadas e não supervisionadas, aliadas à modelagem física, otimizadas com métricas voltadas à detecção de eventos raros.

Palavras-chave: Trackers. XGBoost. Autoencoder. Predição de Falhas. Detecção de Falhas.

ABSTRACT

This work presents the development of a machine learning-based tool for fault detection and prediction in photovoltaic plant trackers. The methodology combines physical analysis, through the comparison between actual and theoretical angles, with machine learning techniques. The anomaly detection approach enabled the classification of trackers as healthy or faulty, while the predictive models, an unsupervised autoencoder and two XGBoost models (classifier and regressor), anticipated failures up to two days in advance. The best results were obtained for the one-day horizon using the regressor model, with a recall of 73% and a precision of 80%. The solution includes an interactive Streamlit interface for fault visualization and monitoring, contributing to proactive and efficient operational management. A key feature of the work is the integration of supervised and unsupervised approaches, combined with physical modeling and optimized using metrics focused on rare event detection.

Keywords: Trackers. XGBoost. Autoencoder. Fault Prediction. Fault Detection.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

CA Corrente Alternada.

CC Corrente Contínua.

GW Gigawatt.

ML Aprendizado de Máquina (Machine Learning).

MPPT Rastreamento do Ponto de Máxima Potência (Maximum Power Point Tracking).

NREL Laboratório Nacional de Energia Renovável (National Renewable Energy Laboratory).

SPA Algoritmo de Posição Solar (Solar Position Algorithm).

VLS-PV Usinas Fotovoltaicas de Grande Porte (Very Large Scale Photovoltaic Power Generation).

SUMÁRIO

1 INTRODUÇÃO

A geração de energia por meio de Usinas Fotovoltaicas de Grande Porte (Very Large Scale Photovoltaic Power Generation) (VLS-PV) é essencial para atender à crescente demanda energética global de forma sustentável. Segundo o Ministério de Minas e Energia do Brasil (2023), entre janeiro e agosto de 2023, dos 7 Gigawatt (GW) adicionados à capacidade energética do país, 3 GW foram provenientes exclusivamente de energia fotovoltaica, que apresentou o maior aumento percentual em comparação com outras fontes de energia.

No entanto, a eficiência desses sistemas é afetada por diversos fatores de degradação. Segundo Rahman et al. (2023), embora a vida útil média de um painel solar seja de 25 anos, elementos como poeira, descoloração, delaminação, rachaduras, umidade e altas temperaturas podem comprometer sua durabilidade. Estudos apontam que a degradação ocorre a uma taxa média de 0,5% ao ano, resultando em uma redução gradual do desempenho ao longo do tempo (JORDAN; KURTZ, 2013).

Além dos painéis solares, outros componentes da usina também se degradam com o tempo, impactando diretamente a eficiência da geração. Os *trackers*, responsáveis por ajustar a inclinação dos painéis solares para otimizar a captação da luz ao longo do dia, podem aumentar a geração de energia em até 35% em comparação com sistemas de painéis fixos (MELO; MOREIRA; VILLALVA, 2020). No entanto, falhas mecânicas ou eletrônicas podem fazer com que o *tracker* permaneça plano em relação ao eixo horizontal ou travado em um ângulo específico. Este último cenário é especialmente crítico, pois, quando travado, a estrutura fica mais vulnerável a danos em situações de ventania (VALENTÍN et al., 2022).

Nesse contexto, técnicas de aprendizado de máquina surgem como uma solução promissora para antecipar falhas, evitando danos severos aos sistemas fotovoltaicos e custos relacionados à reposição de equipamentos. Ferramentas como o TensorFlow permitem a implementação de modelos preditivos capazes de monitorar e identificar padrões anômalos em conjuntos de dados, como séries temporais dos ângulos dos *trackers*, possibilitando a adoção de medidas preventivas antes que ocorram avarias.

Dessa forma, este trabalho investiga como a aplicação de um algoritmo de aprendizado de máquina pode aprimorar a gestão e a manutenção de *trackers*.

1.1 Objetivos

1.1.1 Objetivo geral

Desenvolver uma ferramenta para a detecção e predição de falhas em *trackers* de usinas fotovoltaicas de grande porte.

1.1.2 Objetivos específicos

- Coletar e tratar os dados de séries temporais das posições angulares dos *trackers*;
- Desenvolver técnicas e estratégias para detecção de falhas em *trackers*;
- Desenvolver técnicas para predição de falhas em *trackers*;
- Desenvolver interface gráfica para visualizar as séries temporais e os resultados das análises;

2 FUNDAMENTAÇÃO TEÓRICA

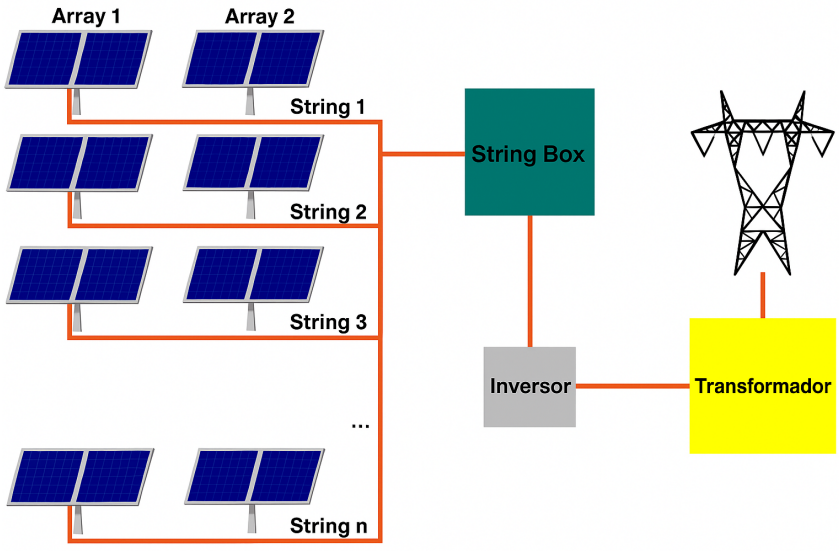
Este capítulo apresenta os principais conceitos utilizados neste trabalho, descrevendo seus fundamentos e a relação com VLS-PVs. A seção 2.1 explora o funcionamento de uma VLS-PV, destacando os componentes mais relevantes para a pesquisa. Na seção 2.2, são exploradas as principais causas de falhas em *trackers*, incluindo falhas mecânicas e eventos climáticos. Também serão apresentados neste capítulo o *Gradient Boosting*, na seção 2.3, e o Autoencoder, na seção 2.4, dois modelos que serão utilizados para a predição de falhas em trackers. Por fim, na seção 2.5, são apresentadas as bibliotecas pvlib, Streamlit e XGBoost, bem como a aplicação *web* Jupyter Notebook, que serão ferramentas utilizadas no desenvolvimento do trabalho.

2.1 Elementos de uma Usina Fotovoltaica

Conforme destacado por Anser et al. (2020) e ilustrado na Figura 1, o processo de conversão da energia solar em elétrica inicia-se com a captação da radiação solar pelos painéis fotovoltaicos, que transformam a energia dos fótons da luz solar em eletricidade. Esses painéis são organizados em *strings*, formando *arrays* fotovoltaicos, que consistem em múltiplos painéis conectados eletricamente em série para aumentar a capacidade de geração.

As *strings* são conectadas a uma String Box, que atua como um ponto de con-

Figura 1 – Topologia simplificada de uma usina fotovoltaica.



Fonte: Adaptado de Anser et al. (2020).

vergência e proteção, agrupando os circuitos elétricos antes de encaminhá-los ao inversor. O inversor é responsável por converter a Corrente Contínua (CC), gerada pelos painéis solares, em Corrente Alternada (CA), tornando-a adequada para uso na rede elétrica.

Após essa conversão, a energia passa pelo transformador, que ajusta a tensão da corrente alternada para níveis compatíveis com a distribuição. Por fim, a eletricidade é integrada à rede elétrica externa, podendo ser consumida localmente ou injetada no sistema de distribuição.

2.1.1 Inversor

Em cenários de VLS-PVs, os inversores têm a função de converter a corrente CC gerada pelas *strings* em corrente CA, que será distribuída para a rede elétrica, garantindo que a forma de onda alternada seja o mais próxima possível da original. Pereira e Gonçalves (2008) destacam algumas das principais funções do inversor na geração fotovoltaica:

- Controle dos dispositivos na rede: o inversor deve isolar da rede as *strings* não energizadas, garantindo a segurança dos operadores. Além disso, ele é responsável por desconectar *strings* que apresentem níveis de tensão, corrente ou frequência fora dos padrões pré-estabelecidos.
- Relatórios de funcionamento: o inversor pode armazenar e transmitir informações relevantes sobre os dispositivos conectados, incluindo dados de corrente, tensão e temperatura dos módulos.
- Ajuste do Rastreamento do Ponto de Máxima Potência (Maximum Power Point Tracking) (MPPT): monitora e ajusta os valores de tensão e corrente das *strings* para que operem próximo ao seu ponto de maior potência, que varia conforme a radiação solar.

2.1.2 String Box

De acordo com Simão (2021), a String Box garante a conexão segura entre as *strings* e o inversor, protegendo o sistema contra variações na corrente CC que poderiam danificá-lo. Dentro da própria String Box, as *strings* são conectadas em paralelo, com um barramento ligado ao terminal positivo e outro ao negativo. Para assegurar a segurança e a confiabilidade da operação, a String Box conta com diversos mecanismos de proteção, dentre os quais Simão (2021) destaca:

- Dispositivos de proteção contra surto: protegem o inversor contra descargas elétricas, minimizando os riscos de danos causados por sobretensões.

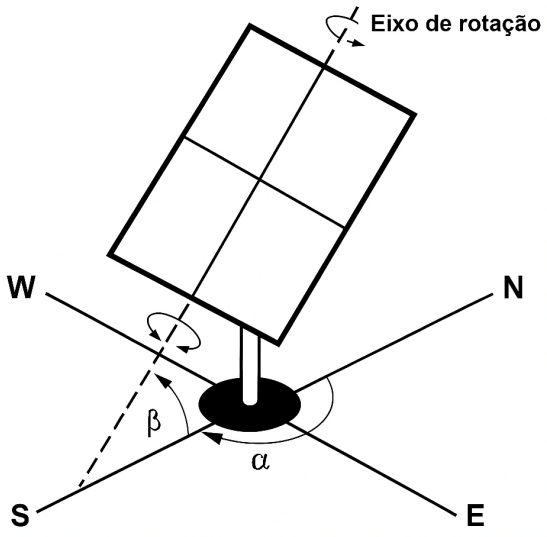
- Chave seccionadora: permite o isolamento do circuito, garantindo segurança durante manutenções e prevenindo riscos elétricos para pessoas próximas.
- Disjuntores ou fusíveis: protegem o circuito contra sobrecargas e curto-circuitos, evitando falhas que possam comprometer a operação do sistema.

2.1.3 *Tracker*

Conforme mencionado no Capítulo 1, o *tracker* é um dispositivo projetado para otimizar a conversão da energia solar em eletricidade, ajustando continuamente a posição dos painéis solares para mantê-los perpendicularmente alinhados com o sol, maximizando a captação da radiação solar. Fora do período de incidência solar, o *tracker* é mantido em uma posição de repouso chamada de *stow*, que costuma ser de aproximadamente 0°. Este ângulo é escolhido para minimizar efeitos climáticos nos painéis durante o período noturno, visto que um *tracker* mantido na horizontal sofre menos interferência de ventanias.

Tudorache e Kreindler (2010) informam que os *trackers* podem ser classificados de acordo com o número de eixos de rotação. Os uniaxiais ajustam a posição dos painéis solares no sentido leste-oeste, acompanhando o movimento diário do sol, sendo possível visualizar um exemplo na Figura 2. Já os biaxiais possuem o mesmo ajuste, mas também realizam correções no ângulo de altitude, permitindo um alinhamento mais preciso ao longo do dia e das estações do ano. Esses sistemas podem ser configurados como *tilt and roll*, ajustando a declinação e o ângulo horário, ou como *azimuth-elevation*, que controla a rotação em azimute e elevação (ETTALBI et al., 2018).

Figura 2 – Representação de um *tracker* uniaxial.



Fonte: Tudorache e Kreindler (2010).

Embora os sistemas de rastreamento solar aumentem a eficiência da captação de energia, eles também implicam em partes móveis e sistemas de controle, o que pode elevar os custos de implementação e manutenção. Dessa forma, os *trackers* de um eixo são frequentemente considerados a melhor solução para pequenas usinas fotovoltaicas, pois oferecem um bom compromisso entre eficiência e viabilidade econômica. Além disso, esses sistemas geralmente possuem um ajuste manual de inclinação no segundo eixo, que pode ser regulado em intervalos ao longo do ano para otimizar a captação da radiação solar conforme as mudanças sazonais (TUDORACHE; KREINDLER, 2010).

2.2 Principais Causas de Falhas em Trackers

Por serem dispositivos mecânicos expostos ao ambiente externo e a diferentes condições climáticas, os trackers podem ser submetidos a uma série de condições que afetam o seu funcionamento. Dentre estas possíveis falhas, destacam-se falhas mecânicas nos componentes e as ocasionadas por fenômenos climáticos.

2.2.1 Falhas mecânicas

Os componentes mecânicos responsáveis por permitir o movimento diário dos trackers são fundamentais para assegurar o bom funcionamento do sistema. Segundo Elerath (2017), os principais elementos que compõem um tracker e que estão sujeitos a falhas são:

- Redutor de giro: componente crítico responsável pela rotação e fixação do tracker.
- Motor: converte energia elétrica em movimento mecânico.
- Controlador: determina a posição angular do tracker com base em dados como latitude, longitude, data e horário.

Elerath (2017) também destaca que o percentual de perda varia de acordo com a arquitetura adotada na implantação dos trackers, que pode ser composta por linhas independentes ou linhas ligadas. Na arquitetura de linhas independentes, cada tracker opera de forma autônoma, com seu próprio motor e controlador. Já na arquitetura de linhas ligadas, diversas linhas são mecanicamente conectadas e dependem de um único conjunto motriz, o que as torna mais suscetíveis a falhas.

2.2.2 Falhas por fenômenos climáticos

Na literatura, a causa mais frequentemente associada às falhas em trackers está relacionada a eventos climáticos, especialmente ventanias. O efeito provocado pelo vento sobre a estrutura recebe o nome de *torsional galloping*.

O *torsional galloping* é um tipo de instabilidade que pode ocorrer em estruturas longas e leves, como os painéis solares instalados sobre trackers. Segundo Young et al. (2020), esse fenômeno tem início quando o vento atinge lateralmente o painel, provocando um pequeno movimento de torção, ou seja, um leve giro sobre o próprio eixo. Em condições normais, esse movimento tenderia a cessar naturalmente. No entanto, em determinadas situações, o vento acaba reforçando esse movimento, fazendo com que o painel oscile cada vez mais, criando um ciclo de autoalimentação.

Isso ocorre porque, ao girar, o painel altera a forma como o vento interage com sua superfície. Essa nova condição de escoamento gera forças que, em vez de reduzir a torção, a intensificam, alimentando ainda mais o movimento. O resultado é um "*loop*" de oscilações, no qual o próprio vento mantém o painel em movimento.

À medida que essas oscilações aumentam, a estrutura de suporte do painel é submetida a tensões cada vez maiores. Se o sistema de travamento ou a rigidez da estrutura não forem suficientes para conter o movimento, os componentes começam a se desgastar rapidamente ou até mesmo a se romper (VALENTÍN et al., 2022). Em casos mais severos, o tracker pode colapsar completamente, gerando danos estruturais e interrompendo a geração de energia naquela região da usina.

A Figura 3 ilustra um caso em que o *torsional galloping* afetou de forma mais severa a lateral esquerda de um tracker, onde ocorreu o colapso completo da estrutura e a queda de diversos painéis. No centro do tracker, observa-se uma torção evidente ao longo do eixo longitudinal, indicando que a estrutura foi submetida a esforços significativos, embora ainda tenha preservado parte de sua integridade. Já a extremidade direita se mostrou

Figura 3 – Danos estruturais causados por *torsional galloping*.

Fonte: Young et al. (2020).

mais preservada, sem sinais aparentes de danos, o que evidencia que o efeito da torção foi assimétrico e concentrou-se principalmente no lado oposto ao sentido predominante do vento.

2.3 *Gradient Boosting*

Gradient Boosting é um método de *ensemble* que constrói um modelo forte aditivo combinando vários classificadores simples, sendo geralmente árvores de decisão, de forma sequencial. A ideia geral do *boosting* é adicionar novos modelos fracos ao conjunto a cada iteração, focando nos erros que o modelo atual cometeu (HASTIE; TIBSHIRANI; FRIEDMAN, 2009). No *Gradient Boosting*, esse processo é guiado pelo gradiente da função de perda: a cada passo, ajusta-se o próximo aprendiz para aproximar o gradiente negativo da perda em relação às previsões atuais (NATEKIN; KNOLL, 2013). Dessa forma, o algoritmo constrói progressivamente um modelo aditivo que minimiza uma dada função de perda como, por exemplo, erro quadrático para regressão ou *log-loss* para classificação, por meio de descida de gradiente em espaço de funções (FRIEDMAN, 2001).

Friedman (2001) formalizou essa abordagem, mostrando que o *boosting* pode ser visto como uma otimização numérica em espaço funcional, usando expansões aditivas em etapas sucessivas. O *Gradient Boosting* inicia com uma previsão inicial como, por exemplo, a constante que minimiza a perda no conjunto de treino, e então repete M iterações. A cada iteração, o algoritmo calcula os pseudo-resíduos, ou seja, o gradiente negativo da perda, treina um modelo base para prever esses resíduos, e então atualiza o modelo somando a contribuição desse novo aprendiz. Dessa forma, cada novo modelo corrige parcialmente os erros remanescentes do modelo acumulado até então (HASTIE; TIBSHIRANI; FRIEDMAN, 2009).

Isso resulta em um modelo final $f_M(x)$ dado pela soma dos M aprendizes fracos ajustados sequencialmente. Além disso, o *Gradient Boosting* é muito flexível quanto à escolha da função de perda; pode-se usar qualquer função diferenciável (FRIEDMAN, 2001; NATEKIN; KNOLL, 2013). Em particular, Friedman (2001) propôs variantes que usam árvores de decisão como aprendizes base (*TreeBoost*), produzindo modelos robustos e interpretáveis tanto para regressão quanto para classificação. Na prática, essas árvores de decisão fracas e com profundidade limitada capturam interações de baixa ordem e são combinadas de forma a reduzir viés enquanto se controla a variância. A abordagem de *Gradient Boosting* tem demonstrado alto desempenho preditivo em diversos problemas, embora demande ajuste de hiperparâmetros como número de iterações M, taxa de aprendizagem e profundidade das árvores para evitar sobreajuste (*overfitting*).

2.3.1 Algoritmo de *Gradient Boosting*

O procedimento básico do algoritmo de *Gradient Boosting* pode ser resumido em etapas numeradas, conforme descrito por Friedman (2001) e Hastie, Tibshirani e Friedman (2009). Para esta seção, foram usadas árvores de decisão como aprendizes base, mas o método é aplicável a qualquer regressor ou classificador fraco. Seja $L(y, f(x))$ a função de perda a ser minimizada, e (x_i, y_i) os dados de treino:

1. Inicialização: o modelo inicial $f_0(x)$ é definido como a constante que minimiza a perda total. Por exemplo, para erro quadrático $L(y, \gamma) = (y - \gamma)^2$, $f_0(x)$ é a média dos alvos. Em geral (HASTIE; TIBSHIRANI; FRIEDMAN, 2009):

$$f_0(x) = \arg\min_{\gamma} \sum_{i=1}^{N} L(y_i, \gamma).$$

2. Para cada iteração $m = 1, 2, \ldots, M$:

   a) Calcula-se os pseudo-resíduos para cada amostra i como o gradiente negativo da perda no modelo atual (FRIEDMAN, 2001; NATEKIN; KNOLL, 2013):

$$r_{im} = - \left. \frac{\partial L(y_i, f(x_i))}{\partial f(x_i)} \right|_{f=f_{m-1}}.$$

   b) Ajusta-se uma árvore de decisão (ou outro modelo fraco) aos resíduos r_{im}, obtendo regiões terminais (folhas) R_{jm} para $j = 1, \ldots, J_m$ (FRIEDMAN, 2001).

   c) Para cada região (folha) R_{jm}, é encontrada a saída γ_{jm} que minimiza a perda localmente (FRIEDMAN, 2001):

$$\gamma_{jm} = \arg\min_{\gamma} \sum_{x_i \in R_{jm}} L\Big(y_i,\ f_{m-1}(x_i) + \gamma\Big).$$

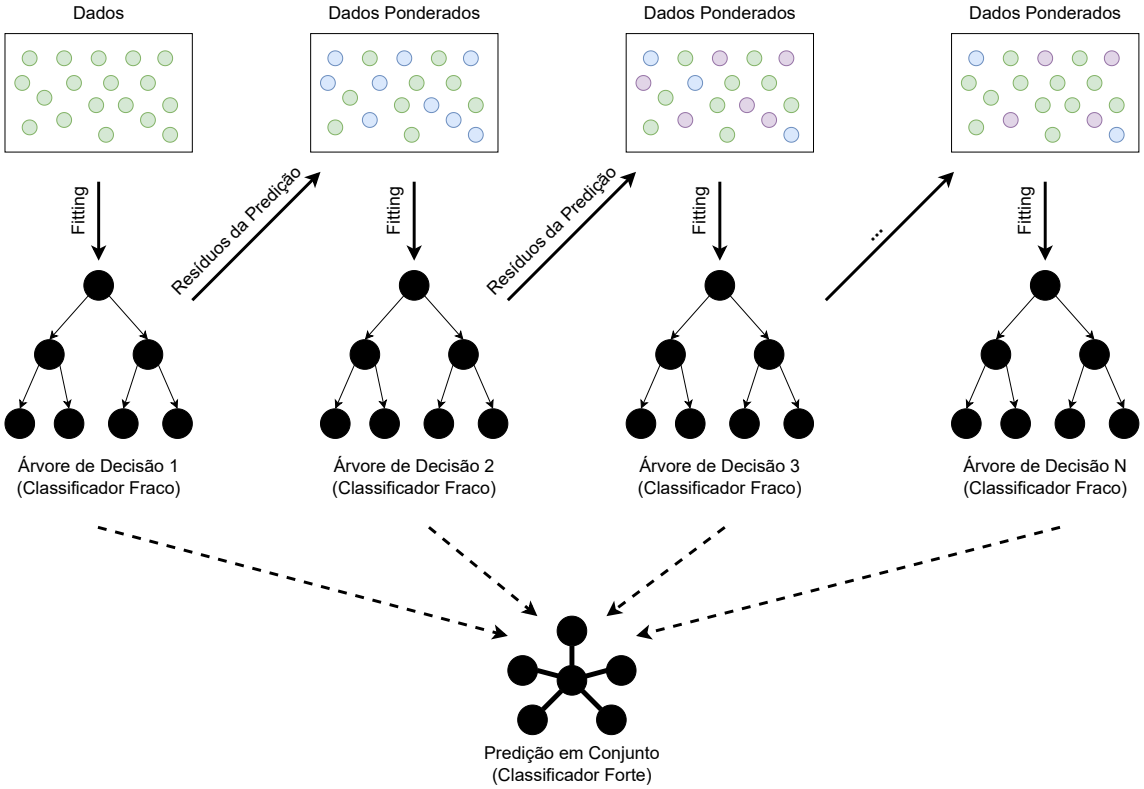
   d) Atualiza-se o modelo aditivo adicionando a contribuição da nova árvore (FRIEDMAN, 2001):

$$f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm}\, \mathbf{1}(x \in R_{jm}).$$

3. Modelo final: após M iterações, o modelo final é $f_M(x)$, sendo a soma de todas as árvores ajustadas. O processo de adicionar modelos continua até esgotar M ou convergir para um erro suficientemente baixo.

Esse algoritmo implementa uma otimização estágio-a-estágio: cada árvore nova corrige erros residuais do conjunto anterior, movendo-se na direção negativa do gradiente da perda (FRIEDMAN, 2001). Em implementações práticas, incluem-se taxa de

Figura 4 – Arquitetura de uma árvore de decisões utilizando *Gradient Boosting*.



Fonte: Adaptado de Deng et al. (2021).

aprendizado (*shrinkage*) e amostragem estocástica para controle de *overfitting*, mas o procedimento básico acima captura o funcionamento fundamental do *Gradient Boosting*.

A Figura 4 ilustra o funcionamento do algoritmo XGBoost por meio da técnica de *gradient boosting*. Nela, as cores das circunferências indicam os dados utilizados em cada etapa do treinamento. Os pontos verdes representam os dados originais utilizados para treinar a primeira árvore. Em seguida, os pontos azuis indicam os resíduos (erros) da primeira predição, combinados aos dados originais, que são usados como entrada da segunda árvore. O processo se repete com os pontos roxos, que representam os resíduos atualizados após a segunda árvore, ainda com base nos dados originais. Ao final, a predição total é obtida pela soma ponderada das contribuições de todas as árvores anteriores, cada uma treinada com foco em corrigir os erros das etapas anteriores.

2.3.2 Propriedades do *Gradient Boosting*

O *Gradient Boosting* destaca-se pela sua flexibilidade e bom desempenho. Ele pode lidar com diferentes tipos de problemas, como regressão, classificação binária ou multiclasse, simplesmente escolhendo a função de perda adequada. Usar árvores de decisão como base confere ao método robustez diante de dados ruidosos ou não estruturados

(FRIEDMAN, 2001). Além disso, por trabalhar em espaço de funções, ele aproveita a informação de forma sequencial, focando nos pontos de difícil predição a cada etapa (HASTIE; TIBSHIRANI; FRIEDMAN, 2009). Em suma, o *Gradient Boosting* constrói um modelo preditivo forte com baixo viés adicionando progressivamente aprendizes fracos que corrigem os erros dos predecessores (NATEKIN; KNOLL, 2013).

2.4 Autoencoder

As redes neurais possuem a capacidade de analisar grandes volumes de dados, identificar padrões e, com base nisso, categorizá-los e realizar predições. Dentre os diversos modelos existentes, os autoencoders se destacam como uma arquitetura de redes neurais projetada para aprendizado não supervisionado. Eles extraem uma representação compactada dos dados de entrada, denominada codificação ou encoder, sem a necessidade de categorização explícita. Em seguida, o decodificador (*decoder*) reconstrói os dados, buscando recuperar a informação original transmitida (BARMAN; HASNAT; NAG, 2022).

A Figura 5 apresenta a estrutura de um modelo autoencoder denominado Deep Autoencoder, que se caracteriza pelo uso de quatro ou cinco camadas para a fase de codificação e a mesma quantidade para a fase de decodificação. Essa ilustração busca exemplificar o funcionamento de um autoencoder, sendo o Deep Autoencoder escolhido como modelo representativo para demonstrar sua estrutura e operação.

Vale destacar que a escolha pelo uso de autoencoder neste trabalho foi motivada pela familiaridade do autor e de seu orientador com esse tipo de rede neural. Por se tratar de uma abordagem não supervisionada, capaz de aprender o comportamento normal dos dados e identificar desvios com base no erro de reconstrução, o autoencoder se mostrou uma solução viável para o problema proposto de detecção de falhas sutis em trackers solares.

2.4.1 Arquitetura de um autoencoder

Ainda segundo Barman, Hasnat e Nag (2022), existem três componentes principais presentes na arquitetura do autoencoder:

- Encoder: componente responsável por comprimir a informação de entrada no sistema. Para isso, utiliza camadas ocultas que reduzem progressivamente o tamanho dos dados enquanto extraem padrões relevantes.
- Bottleneck: após atingir o nível desejado de abstração durante o processo de codificação, a última camada oculta armazena esses dados. Essa camada, chamada *bottleneck* ou espaço latente, pode ser ajustada para priorizar o desempenho, redu-

zindo o grau de compressão, ou para aumentar a compactação, tornando o processo mais complexo.

- *Decoder*: por fim, as camadas ocultas expandem progressivamente a dimensionalidade dos dados para reconstruir a informação codificada no espaço latente, retornando-a a um formato próximo do original.

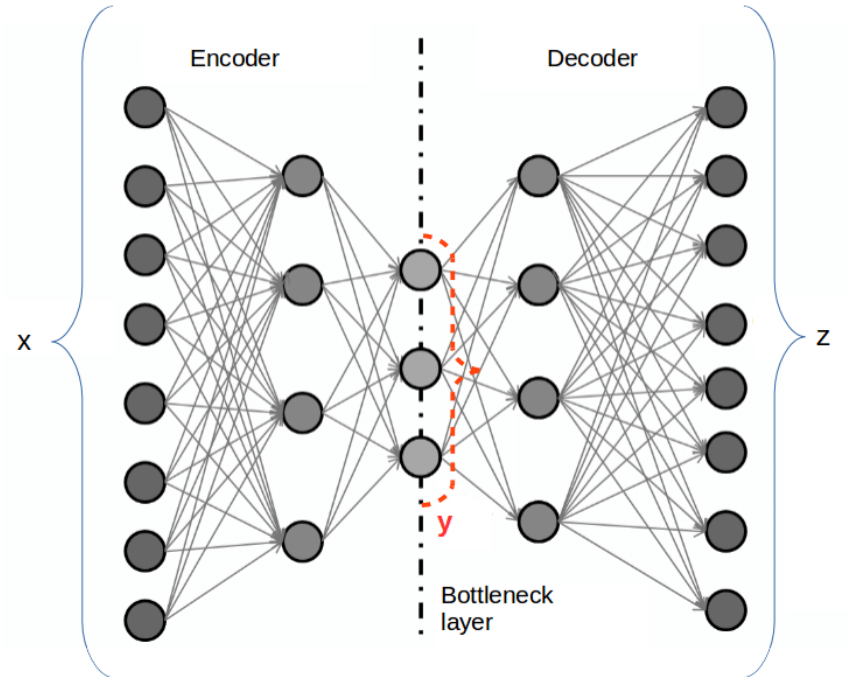
2.4.2 Hiperparâmetros no autoencoder

Os hiperparâmetros são valores ajustáveis no processo Aprendizado de Máquina (Machine Learning) (ML) podendo ser definidos tanto antes quanto durante o treinamento do modelo, com o objetivo de otimizar seu desempenho de acordo com a aplicação. Nesse contexto Berahmand et al. (2024) destacou algumas das variáveis mais relevantes a serem ajustadas no treinamento de autoencoders.

2.4.2.1 Quantidade de camadas ocultas

Valor definido antes do treinamento que determina a capacidade do modelo de detectar padrões complexos nos dados. A escolha do número adequado de camadas é essencial para a aplicação, pois um maior número de camadas pode aumentar o poder de representação do modelo, permitindo uma melhor extração de características. No entanto, um excesso de camadas pode levar ao sobreajuste (*overfitting*), situação em que o modelo

Figura 5 – Estrutura básica do Autoencoder.



Fonte: Barman, Hasnat e Nag (2022).

Figura 6 – Resultados para diferentes valores de taxa de aprendizado.

Fonte: Adaptado de Jordan (2018).

se ajusta tão bem aos dados de treinamento que perde a capacidade de generalizar e prever corretamente novos dados.

2.4.2.2 Taxa de aprendizado (*Learning Rate*)

Define o tamanho do passo dado pelo modelo durante o processo de otimização dos pesos. Para encontrar um valor adequado, a taxa de aprendizado é ajustada iterativamente ao longo do treinamento, permitindo que o modelo se aproxime gradualmente do mínimo da função de erro.

Se a taxa de aprendizado for muito alta, os ajustes nos pesos podem ser excessivos, causando a oscilação dos valores e impedindo a convergência para um local ideal, fenômeno conhecido como *overshooting*. Por outro lado, uma taxa muito baixa pode resultar em um aprendizado excessivamente lento ou até impedir que o modelo alcance um mínimo adequado, levando a um treinamento ineficiente. A Figura 6 demonstra a configuração da taxa de aprendizado com valores muito baixo, adequado e muito alto, respectivamente, evidenciando a importância de conhecer o *dataset* utilizado para evitar indisposições no treinamento.

2.4.2.3 Número de épocas (*Number of Epochs*)

Parâmetro que define quantas passagens completas pelo conjunto de treinamento serão realizadas. Assim como a quantidade de camadas ocultas, um número excessivo de épocas pode resultar em *overfitting*, tornando o modelo altamente ajustado aos dados de treinamento e, consequentemente, menos eficaz na generalização para novos dados.

Para mitigar esse problema, pode-se utilizar a técnica de parada precoce (*early stopping*). De acordo com Prechelt (2000), esse método consiste em dividir os dados em dois conjuntos: um para treinamento e outro para validação. O modelo é treinado apenas com o conjunto de treinamento, enquanto o conjunto de validação é utilizado para monitorar o erro ao longo das épocas. Quando o erro de validação atinge um patamar

satisfatório, o treinamento é interrompido, evitando que o modelo continue aprendendo padrões específicos dos dados de treinamento e perca sua capacidade de generalização.

2.5 Ferramentas utilizadas

O desenvolvimento da ferramenta proposta neste trabalho contou com o suporte de bibliotecas especializadas em análise de dados, modelagem de sistemas fotovoltaicos e construção de interfaces interativas. Cada uma dessas ferramentas foi escolhida com base na sua adequação às necessidades específicas do projeto, como o processamento de séries temporais, simulação de posições solares e implementação de modelos de aprendizado de máquina.

2.5.1 Jupyter Notebook

O Jupyter Notebook é uma aplicação *web* interativa amplamente utilizada para escrever e executar código de forma dinâmica, integrando programação, visualização de dados e documentação em um único ambiente. Segundo sua documentação oficial (PROJECT JUPYTER, 2024), ele permite a criação de documentos compostos por células de código executável, textos formatados com Markdown, visualizações gráficas e outros elementos interativos, sendo bastante adotado em áreas como ciência de dados, estatística, aprendizado de máquina e ensino.

Seu funcionamento é baseado em uma estrutura de células. As células podem ser do tipo código, que são executadas por um *kernel*, ou do tipo texto, escritas em Markdown, permitindo a inserção de títulos, descrições, imagens, entre outros elementos. O *kernel* é responsável por processar o código, executar os comandos e retornar os resultados diretamente na interface do usuário.

2.5.2 Biblioteca pvlib

A pvlib é uma biblioteca em Python que incorpora as funções do Algoritmo de Posição Solar (Solar Position Algorithm) (SPA) desenvolvido pelo Laboratório Nacional de Energia Renovável (National Renewable Energy Laboratory) (NREL), sendo amplamente utilizada para modelagem e análise de sistemas fotovoltaicos. Seu funcionamento se baseia na utilização de diferentes parâmetros, dependendo da função utilizada, como coordenadas geográficas da usina, fuso horário, dados meteorológicos e características dos painéis solares. Com essas informações, a biblioteca permite calcular a posição solar, irradiância, posição angular teórica dos *trackers* e outras variáveis essenciais para a análise e otimização da geração fotovoltaica (NAMBIAR et al., 2022).

2.5.3 Streamlit

O Streamlit é uma biblioteca em Python voltada para o desenvolvimento de aplicações *web* interativas (*dashboards*), com foco no uso de dados e facilidade de configuração. Sua principal vantagem é a simplicidade, permitindo que programadores criem *front-ends* sem a necessidade de conhecimento em HTML, JavaScript ou CSS. Além disso, o Streamlit é compatível com diversos formatos de entrada de dados e recarrega facilmente o conteúdo exibido após serem realizadas alterações no código (AKKEM; KUMAR; VARANASI, 2023).

2.5.4 XGBoost

O XGBoost, ou eXtreme *Gradient Boosting*, é uma versão otimizada e eficiente do algoritmo de *Gradient Boosting*, proposta por Chen e Guestrin (2016). Diferente das abordagens mais tradicionais, o XGBoost trouxe melhorias no desempenho e na forma como o modelo é construído, o que fez com que ele fosse utilizado em problemas práticos, incluindo manutenção preditiva.

Enquanto o *Gradient Boosting* clássico cria modelos somando pequenas árvores para minimizar uma função de erro, o XGBoost vai além e adiciona termos de regularização diretamente na função que o modelo tenta otimizar. Essa regularização ajuda a evitar que o modelo fique muito complexo, tornando as soluções mais simples e com maior chance de funcionar bem em novos conjuntos de dados (CHEN; GUESTRIN, 2016). Além disso, o XGBoost usa informações da segunda derivada da função de perda, o que significa que ele considera mais do que apenas o gradiente para fazer os ajustes, tornando o treinamento mais rápido e com resultados mais precisos.

Segundo Chen e Guestrin (2016), o algoritmo também foi pensado para se adaptar a conjuntos dados esparsos, que são muito comuns em aplicações industriais. O XGBoost conta com um recurso que consegue lidar com valores ausentes de forma mais eficiente, atribuindo para cada divisão da árvore uma direção padrão para os registros que não possuem valor naquela variável. Essa estratégia simplifica o processamento e mantém o bom desempenho do modelo mesmo quando há muitos dados faltantes.

Na prática, o XGBoost tem se destacado em aplicações de manutenção preditiva. Um estudo realizado por Salim, Hebri e Besma (2022) mostrou que o XGBoost teve um desempenho melhor do que algoritmos como Random Forest, SVM e AdaBoost na detecção de falhas em compressores industriais. Além disso, em pesquisas voltadas para a previsão da vida útil restante de motores a jato, o XGBoost apresentou bons resultados e foi uma das técnicas mais bem avaliadas no benchmark NASA PHM 2021 (COHEN; HUAN; NI, 2023). Esses resultados reforçam que o XGBoost é uma ferramenta confiável e eficiente para aplicações industriais, especialmente em projetos de manutenção preditiva,

onde é comum lidar com dados ruidosos, incompletos e desbalanceados.

A escolha do XGBoost foi motivada pelo crescente número de estudos que têm demonstrado bons resultados na aplicação do algoritmo para detecção de falhas em sistemas fotovoltaicos. Nassreddine et al. (2025) utilizaram o XGBoost para classificar falhas em painéis solares, alcançando acurácia próxima de 99%. De forma semelhante, Zhao et al. (2024) combinaram o algoritmo com técnicas de otimização, aumentando a eficiência da detecção em arranjos fotovoltaicos e superando abordagens tradicionais.

3 METODOLOGIA

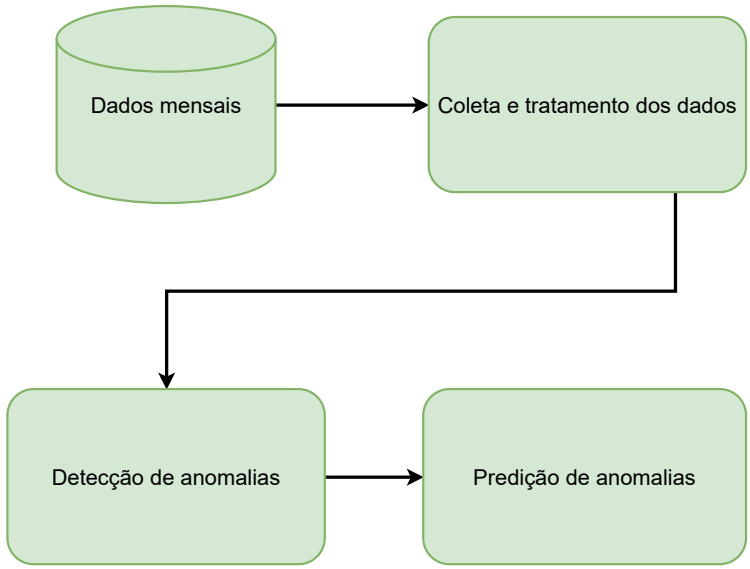
Este capítulo descreve as etapas realizadas para o desenvolvimento da ferramenta de detecção e predição de falhas em *trackers* solares. A metodologia proposta integra diferentes técnicas de análise de dados, aprendizado de máquina e visualização interativa, buscando criar uma solução prática e aplicável ao contexto real de usinas fotovoltaicas.

A Figura 7 apresenta o diagrama geral da metodologia adotada, ilustrando de forma resumida o fluxo do trabalho, desde a coleta e tratamento dos dados até a validação dos modelos e a construção da interface final. Cada etapa será detalhada nos próximos tópicos, respeitando a sequência indicada no fluxograma.

3.1 Coleta e Tratamento dos Dados

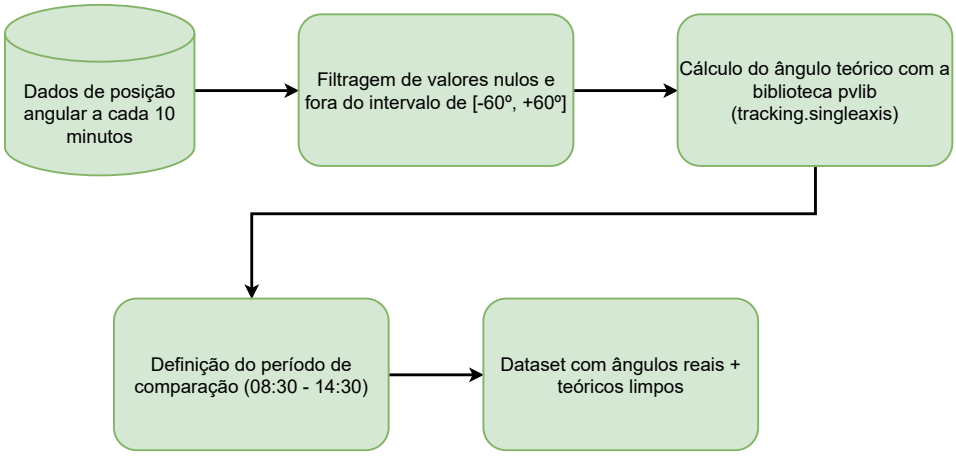
A Figura 8 apresenta o fluxo geral de coleta e tratamento dos dados utilizados neste trabalho. A detecção de falhas nos *trackers* inicia-se com a coleta dos dados de posição angular, registrados em intervalos de 10 minutos e organizados em um *DataFrame* que armazena o nome do *tracker* e sua respectiva posição angular. Foram obtidos seis meses de dados para todos os *trackers* da usina fotovoltaica. Em seguida, esses dados passaram por um processo de filtragem para eliminar valores nulos e aqueles que excedem os limites

Figura 7 – Diagrama geral da metodologia de detecção e predição de falhas.



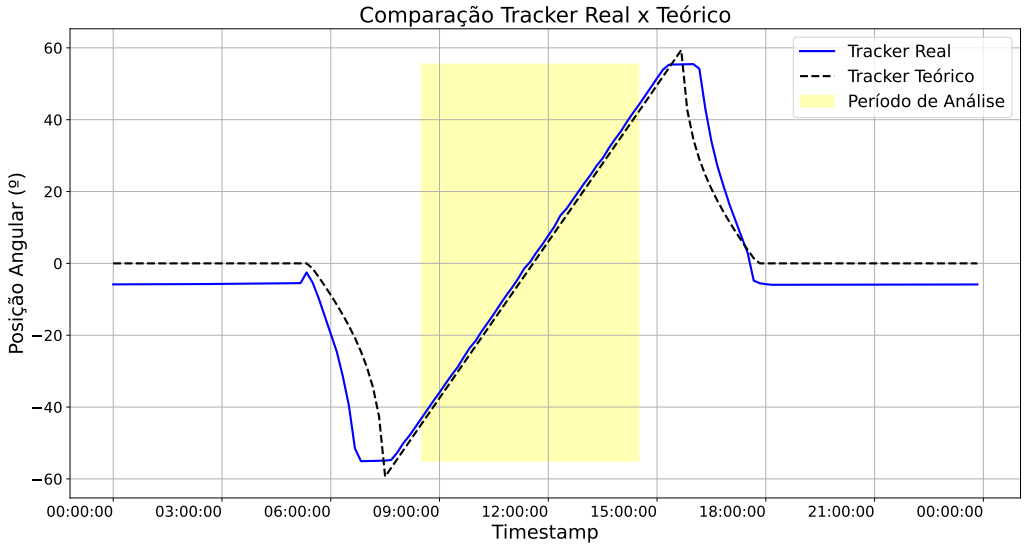
Fonte: Autoria própria.

Figura 8 – Diagrama da coleta e tratamento de dados.



Fonte: Autoria própria.

Figura 9 – Comparação entre ângulo real e teórico.



Fonte: Autoria própria.

operacionais do equipamento. No contexto da usina fotovoltaica analisada neste estudo, o intervalo permitido para os *trackers* varia entre $-60°$ e $60°$. Portanto, qualquer valor fora desse limite é considerado um erro de leitura e, consequentemente, descartado.

Posteriormente, a biblioteca "pvlib" é utilizada para calcular a posição angular teórica de cada *tracker*, considerando parâmetros como latitude, longitude e fuso horário da usina, por meio da função "tracking.singleaxis". Esse valor teórico serve como referência para a comparação com os ângulos registrados, permitindo categorizar as amostras como anômalas ou não.

Embora a posição angular de referência seja uma métrica eficaz para a detec-

ção de falhas, nem todos os instantes de tempo apresentam convergência entre os ângulos medidos e os valores simulados. Conforme ilustrado na Figura 9, que apresenta o comportamento de um *tracker* ao longo do dia, o melhor intervalo para realizar essa comparação corresponde ao período de incidência solar, que, para esta usina, ocorre entre 08:30 e 14:30, representado pela área destacada.

3.2 Detecção de falhas

A detecção de falhas é composta por duas etapas principais: a análise das diferenças entre amostras consecutivas, com o intuito de identificar falhas a partir de variações abruptas nos ângulos registrados, e a comparação direta entre o ângulo do *tracker* e o valor teórico gerado pela biblioteca pvlib.

Para realizar essas análises, foi necessário inicialmente definir o *dataset* a ser utilizado. A série temporal coletada corresponde a um dia de dados para cada estação do ano, sendo sempre selecionado o primeiro dia de cada estação, garantindo assim um espaçamento uniforme entre os casos analisados.

3.2.1 Análise de diferença angular

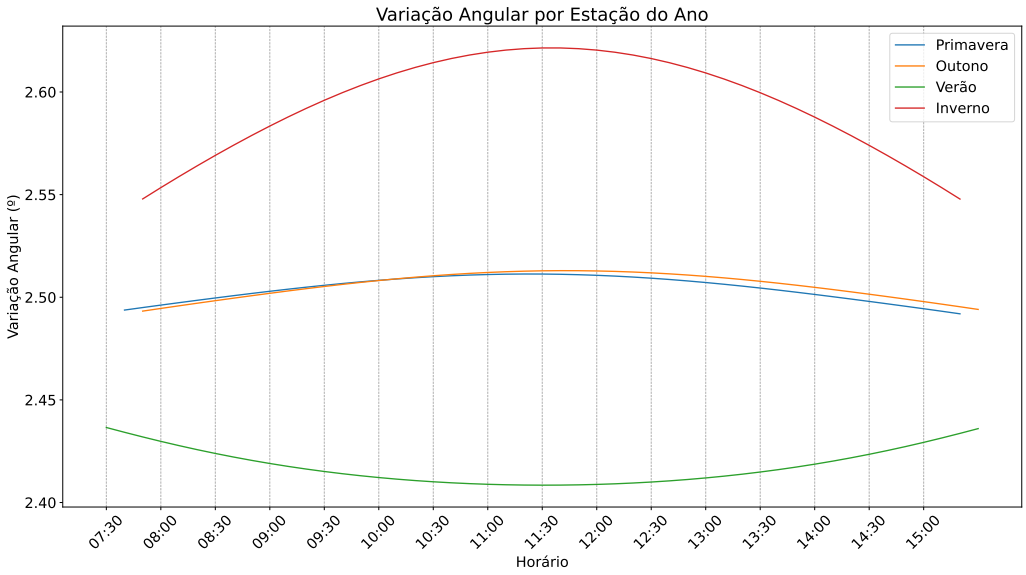
Utilizando um *dataset* composto por um dia de cada estação do ano, buscou-se avaliar o comportamento da diferença entre as amostras para cada reta correspondente. Esse diferencial, calculado através do método "*diff()*" em séries temporais, revela a variação do ângulo de uma amostra para a próxima, indicando a movimentação do *tracker* ao longo do dia. Na prática, isso significa que cada ponto no gráfico reflete o quanto o *tracker* se moveu em relação ao instante anterior. A análise dos resultados, transpostos na Figura 10, evidencia comportamentos distintos dos *trackers* ao longo das estações do ano.

No Inverno, representado pela linha vermelha, observa-se uma variação angular mais acentuada em comparação com as demais estações. Esse comportamento indica que o *tracker* realiza movimentos mais intensos para acompanhar o Sol, cuja trajetória, nesse período, é mais inclinada no céu.

No Verão, a linha verde revela uma diferença angular menor e mais estável ao longo do tempo, refletindo um movimento mais suave do *tracker*. Isso ocorre porque, durante essa estação, o Sol está mais perpendicular em relação ao painel, reduzindo a necessidade de ajustes angulares para otimizar a captação de luz solar.

Primavera e Outono, simbolizados pelas linhas azul e laranja, respectivamente, apresentam curvas bastante semelhantes. Essa proximidade sugere que, nesses períodos, a trajetória solar é intermediária em relação aos extremos observados no Inverno e no

Figura 10 – Gráfico de diferença angular por estação do ano.



Fonte: Autoria própria.

Verão, resultando em ajustes angulares menos abruptos por parte dos *trackers*.

Ao longo do dia, as diferenças de posição começam em níveis mais baixos, aumentam até atingir um pico, que coincide com o meio do dia, e depois diminuem novamente. Esse comportamento reflete o percurso natural do Sol, que demanda maior movimento angular dos *trackers* no início da manhã e no final da tarde, enquanto o deslocamento se estabiliza no período central do dia.

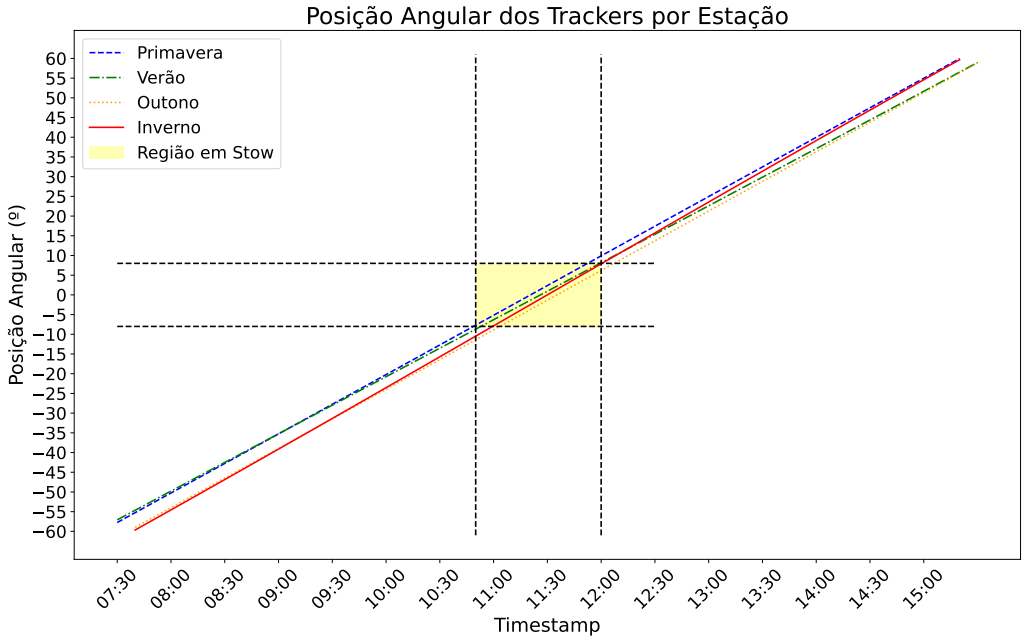
3.2.2 Comparação angular direta

Na etapa de implementação da comparação direta, foi realizado um estudo sobre o comportamento da curva de cada estação do ano, com o objetivo de definir um limiar de confiança na comparação entre a posição angular real do *tracker* e o ângulo teórico gerado pela biblioteca "pvlib". Quando os valores registrados permanecem dentro desse intervalo, o *tracker* é considerado "saudável"; caso contrário, a amostra é classificada como anômala.

É essencial estabelecer um valor de comparação que seja seguro e abrangente, considerando as variações sazonais. Embora um intervalo de 6° fosse suficiente para algumas estações, em outras ele gerava alertas falsos. Além disso, *trackers* travados em *stow* poderiam ser erroneamente classificados como normais durante períodos em que a posição angular da usina se aproxima do ângulo de *stow*, tornando necessária a remoção desse intervalo da análise para evitar falsos positivos.

Para essa avaliação, foi utilizado o mesmo *dataset* apresentado na subseção 3.2.1, a fim de comparar o comportamento angular das curvas para cada estação do ano. A Figura 11 apresenta essas curvas, com a região destacada em amarelo indicando o período

Figura 11 – Gráfico de comparação angular entre estações do ano.



Fonte: Autoria própria.

removido da análise por estar próximo ao ângulo de *stow*, especificamente entre 10:50 e 12:00.

A partir desse estudo, foi definido um intervalo de confiança com amplitude de 8°, representado por duas linhas horizontais no gráfico da Figura 11. O valor de 8° foi escolhido com base na análise sazonal das diferenças angulares observadas. Durante essa avaliação, verificou-se que limiares menores, como 6°, resultavam em um número maior de falsos positivos em estações mais instáveis, como outono e primavera. O intervalo de 8° demonstrou ser o menor valor que assegurava a cobertura das variações máximas nas quatro estações, garantindo uma boa sensibilidade sem comprometer a precisão da detecção. Além disso, esse valor ajuda a prevenir classificações equivocadas de *trackers* travados em posição de *stow* como saudáveis, uma vez que os dados do intervalo crítico do meio-dia foram removidos da análise.

Sendo assim, esse intervalo cobre, de forma segura, todas as estações do ano para a usina analisada, garantindo um desempenho uniforme do algoritmo de detecção ao longo de todo o período. Vale ressaltar que este valor pode variar dependendo do comportamento da usina cujos dados de posição angular são obtidos.

3.2.3 Classificação dos *trackers*

É possível combinar os dois métodos mencionados na subseção 3.2.2 e na subseção 3.2.3 para identificar a presença de falhas na série temporal das posições angulares dos *trackers*. O processo inicia-se pela filtragem dos valores de diferença angular superiores a

1°, de modo que apenas variações significativas no movimento sejam consideradas para a análise. Essa etapa visa eliminar pequenos ruídos que poderiam distorcer a interpretação dos dados.

Na sequência, são calculados a mediana e o desvio padrão das diferenças filtradas. A mediana é utilizada como referência central, uma vez que é menos suscetível a influências de valores extremos, enquanto o desvio padrão mensura a dispersão dos valores em torno dessa mediana, permitindo identificar o quanto essas variações se afastam de um comportamento típico.

Com esses parâmetros definidos, é criada uma nova coluna booleana denominada "Estado" no *DataFrame*, inicialmente configurada como "*False*" para todos os registros, indicando a ausência de falhas. A identificação de falhas ocorre em uma etapa subsequente, onde o código verifica se a diferença angular (Δ) está significativamente fora do padrão esperado. Para isso, são aplicadas duas condições baseadas na mediana e no desvio padrão da distribuição das diferenças angulares. Um valor é considerado anômalo se satisfizer a seguinte inequação:

$$\Delta < \text{Mediana} - \sigma \quad \text{ou} \quad \Delta > \text{Mediana} + 1{,}5 \cdot \sigma.$$

Caso qualquer uma das condições citadas acima seja atendida, o estado do *tracker* para aquele instante é alterado para "*True*", indicando a presença de uma possível falha.

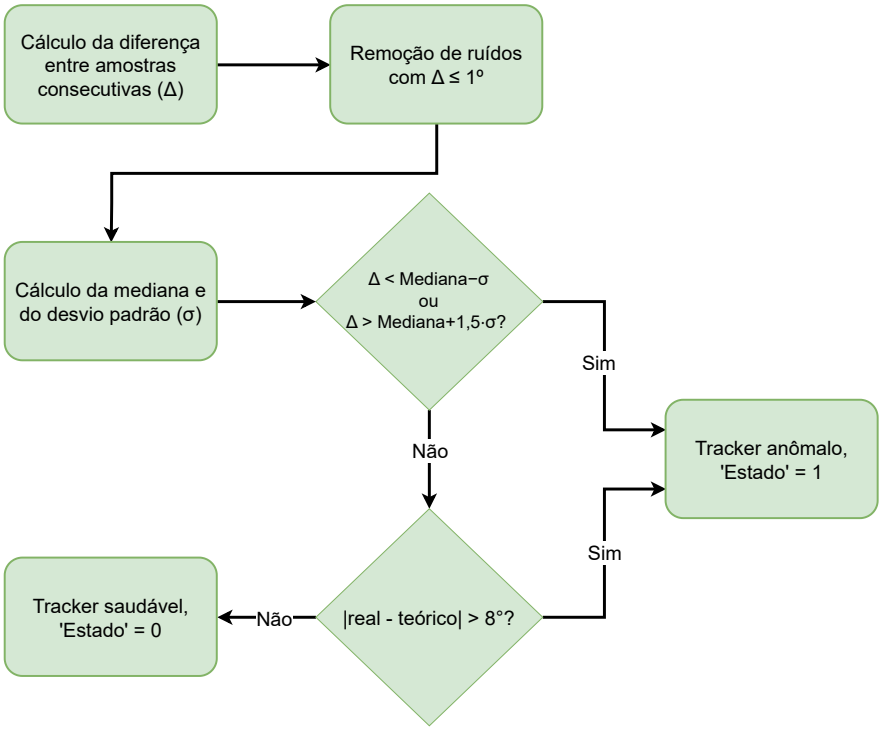
Utilizou-se uma equação assimétrica, pois falhas em *trackers* solares tendem a gerar desvios positivos mais pronunciados, enquanto desvios negativos são geralmente associados a variações operacionais normais. Sendo assim, não houve a necessidade de acrescentar um coeficiente para que os desvios negativos fossem detectados. Vale ressaltar que o fator 1,5 deve ser ajustado de acordo com o perfil de comportamento dos *trackers* da usina, já que uma abordagem generalista pode não ser precisa para detectar descontinuidades em diferentes modelos de equipamento, padrões de operação e estações do ano.

Por fim, é realizada a detecção por comparação direta, onde cada amostra da série temporal é confrontada com o valor teórico de referência. Caso a diferença absoluta entre o valor medido e o estimado ultrapasse 8°, a amostra também é marcada como "*True*".

Vale destacar que, enquanto a técnica de comparação angular direta é mais eficaz para detectar falhas persistentes nos trackers, a análise da diferença angular permite identificar desvios sutis de comportamento ao longo do dia, que nem sempre indicam uma falha constante. Dessa forma, a escolha entre as duas abordagens pode ser feita com base nas métricas desejadas para a aplicação. Neste trabalho, optou-se por utilizar ambas.

Sendo assim, a Figura 12 ilustra o processo completo de detecção de falhas adotado

Figura 12 – Diagrama de detecção de falhas.



Fonte: Autoria própria.

neste trabalho, destacando as duas abordagens complementares utilizadas.

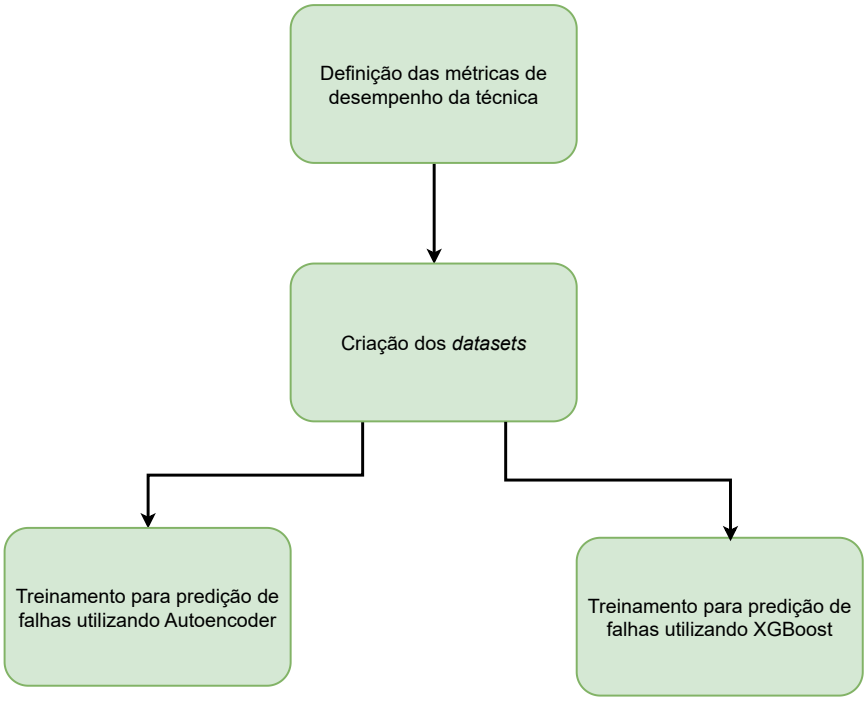
A interface gráfica demonstrando os resultados obtidos para a detecção de falhas está disponível na seção 4.1.

3.3 Predição de falhas

Uma vez tendo definido o perfil de indisponibilidade dos *trackers* por meio das operações realizadas na seção 3.2, foi possível iniciar o processo de predição de falhas. Para tanto, dividiu-se o processo em algumas etapas, demonstradas na Figura 13.

A primeira etapa consiste na definição dos critérios da análise, onde foram estabelecidos o percentual mínimo aceitável de precisão e *recall*, o período futuro em que a falha deve ser prevista, entre outros critérios que serão detalhados no próximo tópico. Em seguida, foram criados os *dataset* de treino, utilizando como base os critérios definidos na etapa anterior. Por fim, foram realizados os treinamentos considerando diferentes critérios variáveis, como a quantidade de dias analisados, o método de treinamento e o valor mínimo de indisponibilidade por dia para classificar uma falha. Cada uma dessas combinações resultou na criação de um novo *dataset* ajustado para o cenário específico de

Figura 13 – Diagrama das etapas para realizar a predição de falhas.



Fonte: Autoria própria.

cada teste. Os resultados obtidos com cada uma das técnicas aplicadas serão apresentados no Capítulo 4, onde também serão comparados para definir qual delas teve o melhor desempenho de acordo com as métricas estabelecidas.

3.3.1 Métricas de desempenho da técnica

A presença de falhas nos *trackers* representa a minoria das amostras do conjunto de dados, com cerca de 4%, como será mostrado na seção 4.1. Por conta disso, utilizar apenas métricas comuns para treinamento de modelos, como acurácia, não é indicado, já que um modelo que sempre prevê o estado normal teria acurácia alta sem identificar nenhuma falha real (HANCOCK; KHOSHGOFTAAR; JOHNSON, 2023).

Nesse cenário, é mais importante adotar métricas que realmente avaliem o desempenho na classe minoritária. A revocação (*recall*), por exemplo, mede a proporção de falhas detectadas corretamente e será a métrica principal para avaliar os modelos. A precisão (*precision*), que mostra quantas das previsões de falha estavam corretas, também será considerada, mas em segunda escala de prioridade. A média entre essas duas métricas, o *F1-score*, não será usada com prioridade neste trabalho por conta do forte desbalanceamento das classes. Além disso, curvas de desempenho como a ROC-AUC podem

ser usadas, mas, em cenários desbalanceados, a PR-AUC (área sob a curva Precisão-Revocação) costuma trazer informações mais relevantes por focar no desempenho sobre a classe minoritária (HANCOCK; KHOSHGOFTAAR; JOHNSON, 2023).

Com isso, as métricas principais adotadas são:

- Revocação: mostra quantas das falhas reais foram corretamente identificadas. É a métrica mais importante, pois falhas não detectadas podem gerar consequências graves (HAN; WEI; HUANG, 2024).
- Precisão: mostra, entre todas as vezes que o modelo previu falha, quantas eram de fato falhas. Alta precisão ajuda a reduzir alarmes falsos.
- PR-AUC: representa a área sob a curva de precisão e revocação. Em conjuntos desbalanceados, é mais confiável para avaliar o desempenho do modelo ao lidar com eventos raros (HANCOCK; KHOSHGOFTAAR; JOHNSON, 2023).

Por fim, definiu-se que o modelo deve prever falhas com pelo menos um dia de antecedência. Essa decisão se baseia na ideia de que intervalos menores poderiam ser inviáveis na prática, já que a mobilização de uma equipe de vistoria exige tempo e planejamento. Além disso, buscou-se priorizar modelos que apresentassem as menores perdas possíveis nas métricas de precisão e revocação, permitindo que a previsão de falhas pudesse ser feita com dois ou mais dias de antecedência sem comprometer gravemente a confiabilidade dos alertas.

3.3.2 Criação do *dataset*

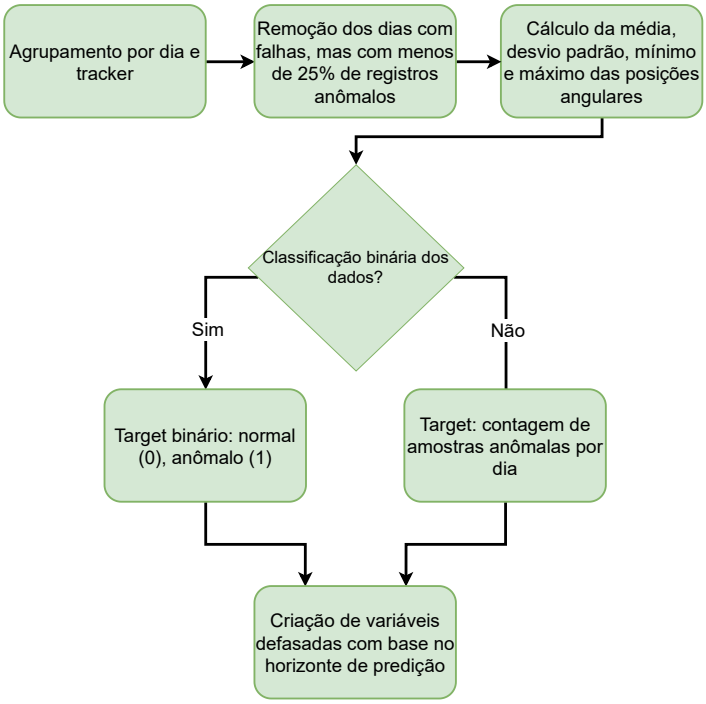
Para viabilizar a análise preditiva de falhas em *trackers* solares, foram criados dois conjuntos de dados diferentes a partir da classificação feita na seção subseção 3.2.3, em que cada amostra temporal dos *trackers* foi rotulada como saudável ou anômala. Ambos os conjuntos organizam as informações de forma agregada por dia e por *tracker*, mas diferem na maneira como representam o evento de falha. A Figura 14 contém o diagrama que demonstra a criação dos dois *datasets* que serão usados no treinamento dos modelos de predição.

O primeiro conjunto foi construído com foco em classificação binária, ou seja, para identificar se houve ou não falha em determinado dia. Para isso, os dados brutos com amostras a cada 10 minutos foram agrupados em registros diários por *tracker*, sendo calculadas variáveis estatísticas como média, desvio padrão, valor mínimo e valor máximo da posição angular ao longo do dia. Também foi optado por remover da análise os dias em que os *trackers* apresentaram falhas em menos de 25% das amostras diárias, uma vez que esse comportamento raramente está associado a falhas reais e, com frequência, se

deve a pequenas flutuações ou instabilidades momentâneas. A presença desses registros intermediários tende a introduzir ruído no conjunto de dados e prejudicar a capacidade do modelo em aprender padrões consistentes que caracterizem falhas significativas. Além disso, a exclusão desse tipo de dado contribuiu para a construção de um conjunto de treinamento mais coeso, com menor ambiguidade entre os rótulos. A variável de saída, chamada "Estado", recebe o valor 1 caso tenha sido detectada ao menos uma falha relevante naquele dia (i.e., com mais de 25% de duração), ou 0 caso contrário. Esse *dataset* permite o treinamento de classificadores supervisionados como o XGBoost, com foco em métricas como *recall* e precisão, visando avaliar a capacidade do modelo em detectar a ocorrência de falhas, mesmo que pontuais.

Já o segundo conjunto de dados foi pensado para quantificar o número de falhas diárias, tratando o problema como uma regressão. A estrutura de consolidação é semelhante à anterior: os dados foram agregados por dia e por *tracker*, com os mesmos parâmetros estatísticos sendo calculados como variáveis de entrada. A diferença está na variável de saída, que agora representa a quantidade total de falhas detectadas no dia, permitindo a aplicação de modelos como o XGBoost Regressor. Após o treinamento, é possível ainda transformar as previsões contínuas em classificações para realizar análises complementares (por exemplo, considerar qualquer valor maior ou igual a um como falha).

Figura 14 – Diagrama de criação dos *datasets* para predição de falhas.



Fonte: Autoria própria.

Alguns aspectos são comuns aos dois *datasets*. A conversão da granularidade de 10 minutos para diária teve como objetivo:

- Reduzir a complexidade do problema.
- Facilitar a criação de variáveis.
- Mitigar a descontinuidade dos dados de entrada devido à natureza da técnica de detecção de falhas.
- Garantir a coerência com o tipo de previsão desejada, que considera o comportamento dos *trackers* ao longo do dia, e não em intervalos curtos como minutos ou horas.

Também foram criadas variáveis de *lag* temporal, ou seja, valores de dias anteriores como entrada para o modelo. Isso possibilita que o algoritmo aprenda com padrões históricos, como, por exemplo, utilizando informações dos dias $t-1$, $t-2$, *etc*, para prever o comportamento no dia t.

Por fim, a normalização dos dados foi considerada especialmente para o uso com Autoencoder, dado que esse tipo de modelo é sensível à escala das variáveis. O Autoencoder foi treinado apenas com dados de *trackers* saudáveis, já que não utiliza a variável de saída diretamente. Por outro lado, os modelos supervisionados como o XGBoost utilizaram dados de ambos os estados, com ajuste no parâmetro `scale_pos_weight` para lidar com o desbalanceamento natural entre as classes.

3.3.3 Predição de falhas com Autoencoder

O Autoencoder é um tipo de modelo que realiza seu treinamento exclusivamente com base nos dados saudáveis do conjunto, ou seja, em situações onde não há falhas. Por essa razão, ambos os *datasets* descritos na subseção 3.3.2 tenderiam a apresentar desempenho semelhante. Ainda assim, optou-se por utilizar o *dataset* que quantifica o número de falhas por dia, ao invés do formato binário, com o objetivo de facilitar a demonstração dos resultados. Para que um dia seja considerado anômalo, é suficiente que exista ao menos uma amostra do *tracker* com uma diferença angular elevada em relação ao ângulo teórico, ou com uma variação interna significativa entre as próprias amostras do dia. Essa abordagem busca garantir o menor erro de classificação possível no processo de treinamento do modelo. A Figura 15 ilustra o passo a passo da elaboração do modelo de predição de falhas com autoencoder, desde o preparo dos dados até a avaliação dos resultados.

Primeiramente, foi definido um horizonte de previsão, ou seja, a quantidade de dias à frente para a qual o modelo deveria indicar a presença ou ausência de falhas. A partir

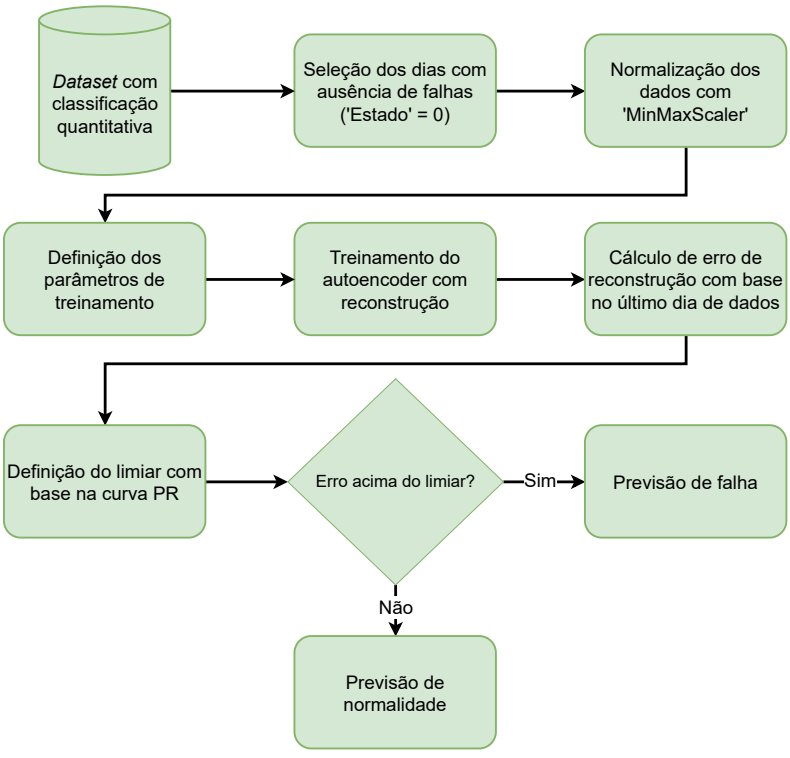
disso, criou-se uma nova coluna de data-alvo ao somar esse intervalo ao *timestamp* original. Em seguida, associou-se a cada dia o valor de falha correspondente ao dia seguinte. A variável-alvo utilizada na avaliação, mas não no treinamento do Autoencoder, foi chamada de `Estado_t+1d`, representando se houve ou não falha no dia seguinte.

Com a estrutura montada, as amostras saudáveis foram normalizadas utilizando o *MinMaxScaler*, e o modelo foi treinado exclusivamente com registros considerados normais (sem falha), com o objetivo de aprender o padrão de comportamento esperado dos *trackers*. O modelo adotado foi um *autoencoder* simples, composto por uma camada de entrada com dimensão igual ao número de variáveis explicativas (sendo elas: média, desvio padrão, mínimo e máximo), uma camada densa de codificação com 4 neurônios e função de ativação $ReLU$, seguida por uma camada de decodificação com ativação linear, projetada para reconstruir os dados de entrada.

A compilação do modelo foi realizada utilizando o otimizador *Adam*, com taxa de aprendizado de 0,001, e função de perda baseada no erro quadrático médio (*mean squared error*). O treinamento ocorreu por até 100 épocas, com tamanho de lote de 64 amostras, empregando o mecanismo de *early stopping* com paciência de 5 épocas, de modo a evitar sobreajuste e preservar os melhores pesos obtidos durante o processo de aprendizado.

Após o treinamento, a etapa de predição foi realizada considerando apenas o último

Figura 15 – Diagrama de predição de falhas com Autoencoder.



Fonte: Autoria própria.

dia disponível para cada *tracker*. Nessa etapa, o erro de reconstrução foi calculado para cada amostra, comparando os dados originais com os valores gerados pelo modelo. A ideia é que amostras com comportamento anômalo apresentem um erro de reconstrução mais alto, já que esses padrões não foram aprendidos durante o treinamento.

A avaliação final consistiu em aplicar um limiar sobre esse erro de reconstrução para classificar se o *tracker* está em risco de falha ou não. Esse limiar foi definido com base na curva precisão-revocação, escolhendo o ponto que maximiza o *F1-score*. Por fim, foram calculadas as métricas de precisão, revocação e *F1-score* para avaliar o desempenho do modelo com base nas falhas reais do dia seguinte. Os *trackers* com maior erro de reconstrução e classificados como positivos foram listados como "em risco".

Os resultados dos testes com o autoencoder podem ser encontrados na subseção 4.2.1.

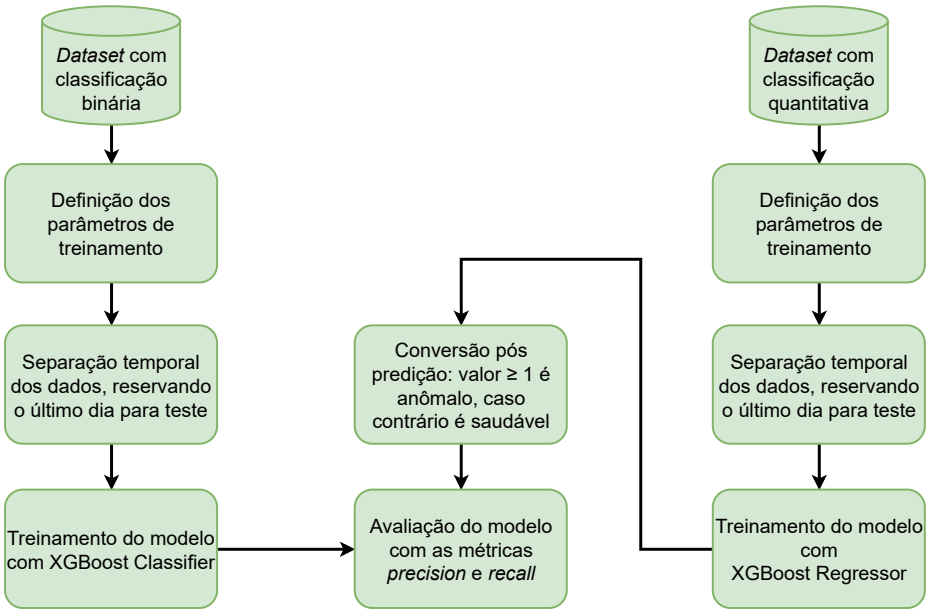
3.3.4 Predição de falhas com XGBoost

A técnica de predição com XGBoost foi aplicada por meio de duas abordagens complementares: uma baseada em classificação binária e outra em regressão quantitativa. Em ambas, foram utilizadas variáveis estatísticas extraídas diariamente (média, desvio padrão, mínimo e máximo da posição angular), combinadas com variáveis defasadas (lags), para capturar padrões temporais de comportamento dos *trackers*. A separação entre treino e teste foi feita de forma temporal, reservando o último dia de cada *tracker* para teste e o restante para treinamento. Os dois modelos adotados e como eles foram treinados está demonstrado na Figura 16.

Na abordagem de classificação binária, a variável-alvo representava a ocorrência ou não de falha em um horizonte futuro de 1 a 3 dias. O modelo *XGBoost* Classifier foi treinado com 300 estimadores e com o parâmetro `scale_pos_weight` ajustado conforme o desbalanceamento entre as classes. A métrica de avaliação escolhida foi a área sob a curva de precisão-revocação (PR-AUC), e o limiar de decisão foi otimizado com base no maior valor de *F1-score*. Com essa configuração, foi possível identificar os *trackers* com maior probabilidade de falha iminente. Os resultados estão disponíveis na subseção 4.2.2.

Na segunda abordagem, o modelo *XGBoost Regressor* foi utilizado para prever a quantidade diária de falhas, mantendo a mesma estrutura de variáveis e o particionamento dos dados. O treinamento foi realizado com 300 árvores, profundidade máxima de 5, taxa de aprendizado de 0,05 e amostragem parcial de colunas e linhas. Os resultados foram avaliados com as métricas do treinamento com Autoencoder: precisão, revocação e *F1-score*. Para fins comparativos, os valores previstos foram posteriormente convertidos em classes binárias (falha prevista ≥ 1), permitindo a análise do desempenho também sob a ótica da classificação. Essa conversão possibilitou comparar diretamente os dois métodos

Figura 16 – Diagrama de predição de falhas com XGBoost.



Fonte: Autoria própria.

e orientar a tomada de decisão sobre quais *trackers* apresentavam maior risco de falha iminente. Os resultados estão presentes na subseção 4.2.3.

4 RESULTADOS

Neste tópico, serão apresentados os resultados dos algoritmos de detecção e predição de falhas, acompanhados de discussões sobre os resultados e comparações entre os diferentes métodos adotados.

4.1 Detecção de falhas

Uma vez tendo classificado os *trackers* como normais ou anômalos, foi possível calcular o percentual de indisponibilidade por *tracker* durante o período que engloba os dados. Para tanto, foi utilizada a biblioteca Streamlit em Python para desenvolver um *dashboard* contendo as informações necessárias para a visualização dos resultados. Ao acessar a página por meio do link da URL, são exibidas informações relacionadas à análise descrita na **??**, como o intervalo de tempo da análise, a posição de *stow* da usina e a tolerância angular utilizada para classificar uma falha.

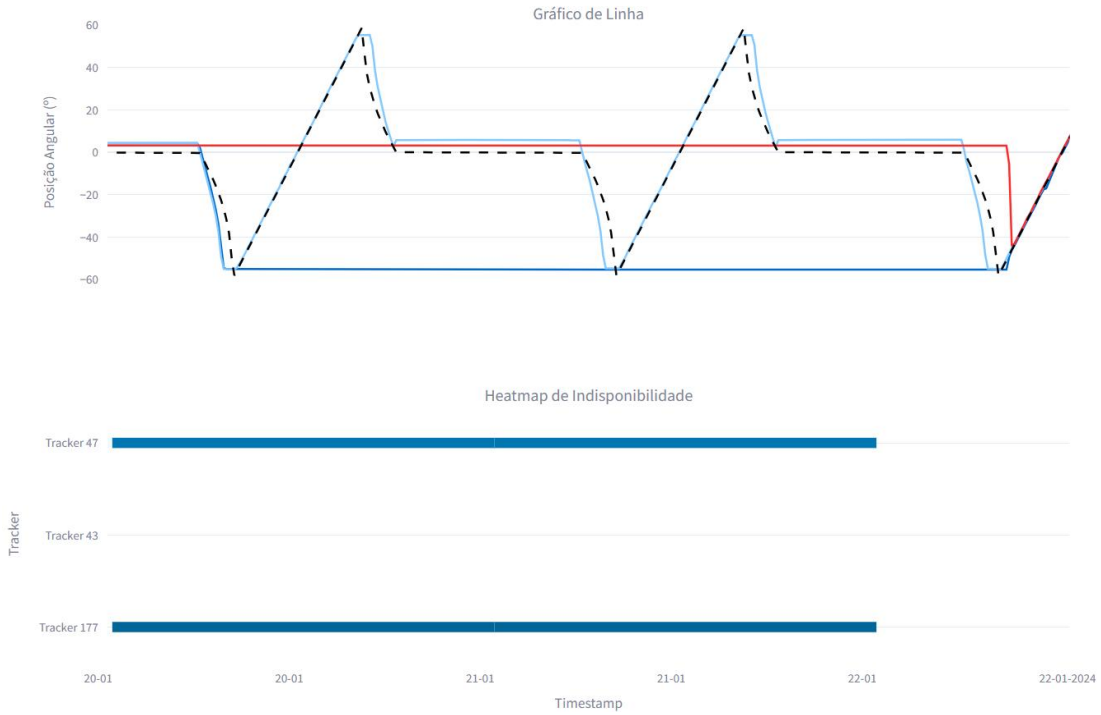
Assim que a opção "Iniciar Análise" é selecionada, a página processa as informações de posição angular dos *trackers* e é atualizada com os resultados da análise. No topo, foi adicionado um marcador que informa o percentual de *trackers* categorizados como normais ao longo do período analisado. Para esse cálculo, foi realizada a média entre todas as amostras com a coluna "Estado" definida como "*True*" e o total de amostras registradas. Esse indicador fornece um panorama geral da operação da usina no período selecionado, permitindo uma avaliação rápida do desempenho dos *trackers*.

A Figura 17 apresenta a porção inicial da página, contendo os parâmetros seleci-

Figura 17 – Seletor de parâmetros da página dos *trackers*.

Fonte: Autoria própria.

Figura 18 – Gráficos para análise de indisponibilidade.



Fonte: Autoria própria.

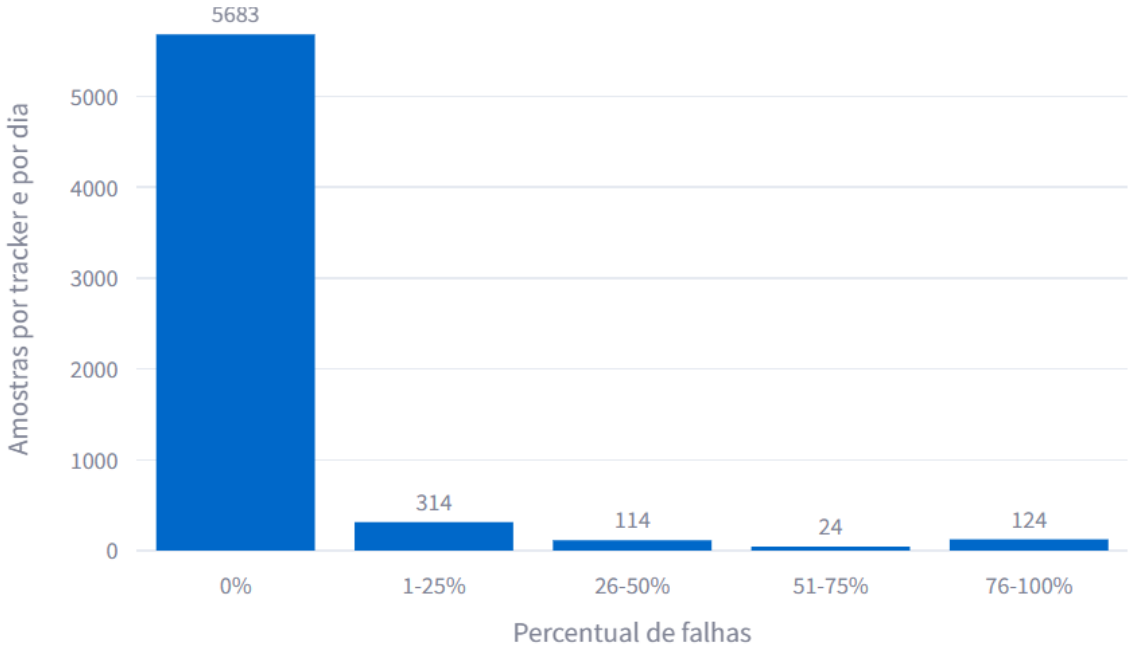
onáveis pelo usuário e a disponibilidade da usina no intervalo definido.

Logo abaixo, foi adicionado um gráfico de linhas que apresenta a posição angular dos *trackers* da usina em comparação com o *tracker* de referência gerado pela biblioteca pvlib. Esse gráfico oferece uma visão geral do comportamento das posições angulares ao longo do período analisado, permitindo a identificação de *outliers* por meio da comparação direta entre as curvas reais e a teórica.

O *dashboard* também conta com um gráfico em formato de *heatmap*, que exibe o percentual de indisponibilidade de cada *tracker* por dia. À medida que um *tracker* apresenta mais momentos de falha ao longo do dia, o retângulo correspondente no gráfico assume uma coloração mais intensa no gradiente de cor. Ao posicionar o cursor sobre um dos pontos, são exibidas informações do *tracker* naquele dia, como o nome, a data e o percentual de tempo em que ele permaneceu indisponível naquele dia. Esse indicador permite que *trackers* com falhas mais recorrentes se destaquem visualmente em relação àqueles com falhas pontuais, facilitando a identificação dos casos mais críticos e auxiliando na priorização das ações de manutenção.

A Figura 18 apresenta o gráfico de linhas com a posição angular dos *trackers*, seguido pelo *heatmap* que mostra, por meio do gradiente de cor, o percentual de indisponibilidade diária. Na figura, os *trackers* estão representados nas três condições principais de funcionamento: operação normal (azul claro), travamento em *stow* (vermelho) e travamento fora do intervalo de *stow* (azul escuro).

Figura 19 – Histograma de indisponibilidade dos *trackers*.



Fonte: Autoria própria.

Por fim, foi incluído um gráfico em formato de histograma para representar o percentual de falhas dos *trackers* com base na proporção de tempo em que ficaram indisponíveis ao longo de um dia. Cada ponto do gráfico corresponde a um *tracker* em um determinado dia e está agrupado em faixas de 25%. Como mostra a Figura 19, a maior parte dos *trackers* manteve comportamento saudável durante o período analisado, concentrando-se no grupo com 0% de falha, enquanto a faixa entre 1% e 25% corresponde ao segundo maior conjunto observado.

4.2 Predição de falhas

Esta seção apresenta os resultados obtidos com os diferentes métodos aplicados para a predição de falhas em *trackers* solares, utilizando técnicas de aprendizado de máquina. Foram avaliados dois modelos principais: o autoencoder, voltado à detecção de falhas em séries temporais, e o algoritmo XGBoost, utilizado tanto para classificação binária (XGBoost *Classifier*) quanto para regressão da quantidade de falhas (XGBoost Regressor).

A análise buscou verificar a capacidade de cada modelo em antecipar falhas reais com diferentes janelas de antecedência, variando de um até vários dias. Isso permitiu avaliar o desempenho preditivo das técnicas em cenários de curto e médio prazo, fundamentais para ações preventivas em usinas fotovoltaicas de grande porte.

A metodologia de implementação de cada modelo foi apresentada anteriormente,

na seção 3.3. Neste capítulo, o foco está nos resultados obtidos e na comparação entre os modelos com base nos critérios de desempenho definidos. Os resultados estão organizados por tipo de modelo e horizonte de previsão, facilitando a análise comparativa entre as abordagens. Adicionalmente, foi incluído um tópico comentando os testes realizados com um *dataset* estruturado a partir de amostras em intervalos de 10 minutos, ao invés de dados agregados por dia. A seção discute as limitações encontradas nessa abordagem e os motivos pelos quais não foi possível obter resultados satisfatórios com esse formato de entrada.

4.2.1 Resultados com Autoencoder

A técnica de predição baseada em Autoencoder foi aplicada com o objetivo de identificar falhas a partir da reconstrução de amostras saudáveis. O modelo foi treinado exclusivamente com dados de *trackers* em operação normal, e sua avaliação foi realizada considerando diferentes horizontes de previsão (1, 2 e 3 dias). O desempenho foi analisado por meio de métricas de classificação, especialmente *recall*, por ser a mais relevante no contexto de manutenção preditiva.

4.2.1.1 Predição com 1 dia de antecedência

Com 1 dia de antecedência, o modelo obteve um resultado mistos, com revocação de apenas 55% para a classe de falhas e precisão de 86%. Isso indica que pouco mais da metade das falhas reais foram corretamente antecipadas, não se mostrando a técnica ideal para o objetivo de prever falhas. A acurácia geral foi de 97%, e os *trackers* classificados com maior risco apresentaram erros de reconstrução significativamente elevados, indicando que o modelo teve facilidade em identificá-los. A matriz de confusão (Tabela 1) demonstra o bom desempenho na detecção da classe majoritária e resultados mistos na identificação de falhas, sendo 0 a ausencia de falhas e 1 a presença delas.

Tabela 1 – Matriz de confusão - Autoencoder (1 dia).

	Predito 0	**Predito 1**
Real 0	187	1
Real 1	5	6

4.2.1.2 Predição com 2 dias de antecedência

Para o horizonte de 2 dias, o modelo apresentou desempenho semelhante ao anterior. O *recall* foi de 50%, com precisão de 83%, demonstrando uma perda pequena ao adicionar 1 dia de horizonte. A matriz de confusão (Tabela 2) apresenta os resultados encontrados neste teste.

Tabela 2 – Matriz de confusão - Autoencoder (2 dias).

	Predito 0	**Predito 1**
Real 0	187	1
Real 1	5	5

4.2.1.3 Predição com 3 dias de antecedência

Ao considerar 3 dias de antecedência, observou-se uma queda mais acentuada no desempenho do modelo. O *recall* foi reduzido para 36%, com uma precisão de apenas 25%. A acurácia geral também caiu para 90%, e a maioria dos casos de falha não foi corretamente identificada. A matriz de confusão (Tabela 3) evidencia essa limitação. Sendo assim, esse horizonte foi definido como ponto de parada para predição, visto que os resultados obtidos a partir desse ponto não são mais considerados satisfatórios.

Tabela 3 – Matriz de confusão - Autoencoder (3 dias).

	Predito 0	**Predito 1**
Real 0	176	12
Real 1	7	4

4.2.1.4 Análise comparativa

Os resultados indicam que o desempenho do Autoencoder tende a diminuir conforme o horizonte de predição aumenta. Para janelas curtas (1 ou 2 dias), o modelo apresenta boa precisão, porém é capaz de detectar apenas metade das falhas. Já para predições com maior antecedência, a capacidade de generalização do modelo diminui, afetando negativamente a sensibilidade e a confiabilidade dos alertas gerados.

4.2.2 Resultados com XGBoost *Classifier*

A primeira estratégia adotada com o XGBoost consistiu em um modelo de classificação binária, cujo objetivo era identificar se um *tracker* apresentaria ou não falhas em determinado dia futuro. O desempenho do modelo variou conforme o horizonte de predição, com redução gradual nos índices de revocação à medida que se aumentava o horizonte da predição.

4.2.2.1 Predição com 1 dia de antecedência

Com um dia de antecedência, o modelo obteve um bom desempenho, alcançando revocação de 64% para a classe de falha, com precisão de 88%. A matriz de confusão (Tabela 4) mostra que 7 das 11 falhas foram corretamente identificadas, com apenas 4 falsos negativos. A alta precisão indica que a maior parte das predições de falha realmente

correspondiam a dias problemáticos, enquanto o *recall* sugere que ainda há espaço para melhorar a detecção de todas as falhas.

Tabela 4 – Matriz de confusão - XGBoost *Classifier* (1 dia).

	Predito 0	**Predito 1**
Real 0	190	1
Real 1	4	7

4.2.2.2 Predição com 2 dias de antecedência

Com dois dias de antecedência, a revocação caiu para 36%, enquanto a precisão se manteve relativamente alta, em 80%. O modelo passou a perder mais eventos de falha reais, com 7 das 11 falhas sendo classificadas incorretamente como dias normais, como pode ser observado na Tabela 5. Essa perda de sensibilidade é esperada com o aumento da janela temporal.

Tabela 5 – Matriz de confusão - XGBoost *Classifier* (2 dias).

	Predito 0	**Predito 1**
Real 0	190	1
Real 1	7	4

4.2.2.3 Predição com 3 dias de antecedência

Ao considerar três dias de antecedência, o modelo continuou com precisão elevada (75%), mas a revocação diminuiu ainda mais, chegando a 27%. Das 11 falhas presentes, apenas 3 foram corretamente previstas (Tabela 6), o que limita a aplicabilidade do modelo nesse horizonte para fins de manutenção preventiva imediata.

Tabela 6 – Matriz de confusão - XGBoost *Classifier* (3 dias)

	Predito 0	**Predito 1**
Real 0	190	1
Real 1	8	3

4.2.3 Resultados com XGBoost Regressor

A segunda abordagem consistiu em prever a quantidade diária de falhas por *trackers*, transformando posteriormente os valores previstos em categorias binárias (falha prevista quando o valor estimado era maior ou igual a 1). Essa técnica se mostrou mais eficaz em alguns cenários, especialmente no que diz respeito à revocação, com resultados mais equilibrados entre os dois extremos (curto e médio prazo).

4.2.3.1 Predição com 1 dia de antecedência

Com um dia de antecedência, o modelo atingiu revocação de 73% e precisão de 80%, superando a abordagem binária em sensibilidade. Isso significa que 8 das 11 falhas foram corretamente identificadas, com apenas 3 falsos negativos, conforme indicado na Tabela 7. A classificação quantitativa se mostrou vantajosa nesse cenário, ao captar mais eventos anômalos com antecedência adequada ao escopo do projeto.

Tabela 7 – Matriz de confusão - XGBoost Regressor (1 dia).

	Predito 0	**Predito 1**
Real 0	189	2
Real 1	3	8

4.2.3.2 Predição com 2 dias de antecedência

A performance do modelo se manteve estável, com revocação também em 73% e leve queda na precisão para 67%. Isso indica que o modelo ainda conseguiu antecipar 8 das 11 falhas, mesmo com o aumento do horizonte, como mostra a Tabela 8. Trata-se de um resultado positivo para aplicações de médio prazo, possibilitando intervenções com maior tempo de planejamento.

Tabela 8 – Matriz de confusão - XGBoost Regressor (2 dias).

	Predito 0	**Predito 1**
Real 0	187	4
Real 1	3	8

4.2.3.3 Predição com 3 dias de antecedência

Com três dias de antecedência, o desempenho do modelo sofreu uma queda mais acentuada. A revocação caiu para 36%, igualando-se à abordagem binária nesse horizonte. A precisão também foi reduzida, ficando em 57%, conforme demonstrado na Tabela 9.

Tabela 9 – Matriz de confusão - XGBoost Regressor (3 dias).

	Predito 0	**Predito 1**
Real 0	188	3
Real 1	7	4

4.2.3.4 Análise comparativa entre as abordagens

A comparação entre os dois métodos indica que a abordagem com classificação quantitativa (XGBoost Regressor) apresentou desempenho mais equilibrado nos horizontes de 1 e 2 dias, com melhores índices de revocação em relação ao modelo binário. Já

o modelo de classificação binária (XGBoost *Classifier*) manteve maior estabilidade na precisão ao longo de todos os horizontes, sendo uma alternativa mais conservadora, com menor risco de falsos positivos.

Esses resultados mostram que, para aplicações em que o mais importante é captar o maior número possível de falhas (mesmo que haja alguns alarmes falsos), o modelo baseado em regressão se mostra mais adequado nos horizontes curtos, tornando-se o mais adequado para a finalidade deste trabalho.

4.2.4 Tentativa com dados não agregados por dia

Inicialmente, foi considerada a abordagem de predição com dados em alta frequência, utilizando amostras a cada 10 minutos ao longo do dia, de forma semelhante à metodologia empregada na detecção de falhas. No entanto, esse conjunto de dados estava limitado ao intervalo entre 08:30 e 14:30, o que resultava em apenas 29 amostras por dia. Ao tentar utilizar esse conjunto para prever falhas no dia seguinte, os resultados se mostraram insatisfatórios.

Essa limitação está associada ao desalinhamento entre a granularidade dos dados de entrada e a escala temporal da variável-alvo. Enquanto as entradas capturam variações locais e rápidas em um curto intervalo do dia, o objetivo da predição é estimar o comportamento agregado de um dia inteiro futuro. Como consequência, o modelo não consegue capturar relações causais de longo prazo, especialmente quando sinais relevantes de falha ocorrem fora da janela analisada.

Além disso, como o modelo tentava prever o mesmo evento do dia seguinte a cada 10 minutos, criava-se uma redundância de predições, com variações indesejadas e ruído estatístico. Esses fatores motivaram a adoção de uma abordagem alternativa, baseada em agregações diárias. Ao calcular estatísticas como média, desvio padrão, mínimo e máximo para cada dia, a entrada do modelo passou a representar de forma mais robusta o padrão de operação do *trackers*, além de se alinhar com a granularidade da variável-alvo.

Na comparação entre os resultados da tentativa com granularidade de 10 minutos e os obtidos com o modelo XGBoost Regressor utilizando agregações diárias (ambos com horizonte de 1 dia), observou-se um aumento de 40% na precisão e 25% na revocação, evidenciando a superioridade da abordagem baseada em dados diários.

5 CONCLUSÕES

Conclui-se que a abordagem proposta foi eficaz na detecção e predição a curto prazo de falhas em *trackers* solares. A ferramenta desenvolvida conseguiu integrar, de forma funcional, as etapas de processamento dos dados, detecção e predição via aprendizado de máquina, além de uma interface interativa para visualização e análise dos resultados. O sistema foi capaz de identificar *trackers* com comportamento anômalo e antecipar falhas com até dois dias de antecedência, o que pode ser decisivo para a manutenção preditiva em usinas de grande porte.

Os resultados mostraram que o modelo XGBoost Regressor teve maior sensibilidade para detectar falhas iminentes, principalmente nos horizontes de 1 e 2 dias. Já o modelo *Classifier* se destacou pela maior precisão, reduzindo o número de falsos positivos. Isso demonstra que as duas abordagens podem ser complementares, dependendo do tipo de ação desejada (maior abrangência na detecção ou maior segurança na decisão). Entretando, por normalmente ser priorizado o *recall* em cenários de manutenção preditiva, o resultado mais adequado encontra-se no modelo XGBoost Regressor.

Apesar dos resultados promissores, algumas limitações foram identificadas. O desempenho dos modelos caiu em horizontes superiores a dois dias, o que limita a capacidade de predição de longo prazo. Além disso, todo o estudo foi feito com dados históricos de um único parque fotovoltaico, sem integração em tempo real, o que restringe a generalização dos resultados. O desbalanceamento entre classes (dias com falha são raros) também foi um desafio, exigindo técnicas de ponderação e impactando especialmente a taxa de falsos negativos.

De forma geral, os resultados confirmam que é possível aplicar aprendizado de máquina para antecipar falhas de maneira prática e acessível, e que mesmo predições com poucos dias de antecedência já são suficientes para orientar decisões operacionais em campo. O trabalho desenvolvido abre caminho para futuras investigações voltadas à operação em tempo real, ao aumento da eficiência na manutenção e à escalabilidade da solução em diferentes contextos de usinas fotovoltaicas.

5.1 Trabalhos futuros

Como próximos passos, alguns caminhos podem ser explorados para ampliar a robustez e aplicabilidade da ferramenta:

- Integração em tempo real: Acoplar a ferramenta a sistemas SCADA ou fluxos de

dados contínuos via API, para que a detecção e a predição possam gerar alertas automáticos e acionáveis.

- Novos modelos: Investigar algoritmos mais complexos, como redes neurais recorrentes (LSTM/GRU) ou técnicas de aprendizado semi-supervisionado, que podem melhorar a captura de padrões temporais e lidar melhor com escassez de falhas.
- Ampliação do escopo: Incluir variáveis adicionais disponíveis nos sistemas SCADA, como dados meteorológicos ou status dos inversores, que podem agregar valor ao modelo.
- Análise de impacto econômico: Estimar o ganho em disponibilidade e a redução de custos operacionais decorrentes da aplicação da manutenção preditiva baseada na ferramenta, visando demonstrar o retorno sobre investimento.

REFERÊNCIAS

AKKEM, Y.; KUMAR, B. S.; VARANASI, A. Streamlit Application for Advanced Ensemble Learning Methods in Crop Recommendation Systems – A Review and Implementation. *Indian Journal of Science and Technology*, v. 16, n. 48, p. 4688–4702, dez. 2023. ISSN 0974-5645. Publisher: The Indian Society of Education and Environment. Disponível em: https://indjst.org/. 24

ANSER, M. K. et al. Assessing the integration of solar power projects: SWOT-based AHP–F-TOPSIS case study of Turkey. *Environmental Science and Pollution Research*, v. 27, n. 25, p. 31737–31749, set. 2020. ISSN 1614-7499. Disponível em: https://doi.org/10.1007/s11356-020-09092-6. 12

BARMAN, D.; HASNAT, A.; NAG, R. An introduction to autoencoders. In: _____. *Computation*. Iii. CSE-GCETTB, 2022. p. 14–23. Disponível em: https://www.researchgate.net/publication/379542201_an_introduction_to_autoencoders. 20, 21

BERAHMAND, K. et al. Autoencoders and their applications in machine learning: a survey. *Artificial Intelligence Review*, v. 57, n. 2, p. 28, fev. 2024. ISSN 1573-7462. Disponível em: https://doi.org/10.1007/s10462-023-10662-6. 21

CHEN, T.; GUESTRIN, C. Xgboost: A scalable tree boosting system. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: Association for Computing Machinery, 2016. (KDD '16), p. 785–794. ISBN 9781450342322. Disponível em: https://doi.org/10.1145/2939672.2939785. 24

COHEN, J.; HUAN, X.; NI, J. Fault prognosis of turbofan engines: Eventual failure prediction and remaining useful life estimation. *arXiv preprint arXiv:2303.12982*, 2023. Disponível em: https://arxiv.org/abs/2303.12982. 24

DENG, H. et al. Ensemble learning for the early prediction of neonatal jaundice with genetic features. *BMC Medical Informatics and Decision Making*, v. 21, 12 2021. Disponível em: https://doi.org/10.1186/s12911-021-01701-9. 19

ELERATH, J. G. Solar tracker effectiveness: It's all about availability. In: *2017 IEEE International Telecommunications Energy Conference)*. [s.n.], 2017. p. 156–162. ISSN: 0275-0473. Disponível em: https://ieeexplore.ieee.org/document/8214128. 15

ETTALBI, K. et al. Preliminary performance analysis of a CPV system with dual-axis tracker in a desert region. In: *2018 9th International Renewable Energy Congress (IREC)*. [s.n.], 2018. p. 1–6. ISSN: 2378-3451. Disponível em: https://ieeexplore.ieee.org/document/8362534. 14

FRIEDMAN, J. H. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, Institute of Mathematical Statistics, v. 29, n. 5, p. 1189 – 1232, 2001. Disponível em: https://doi.org/10.1214/aos/1013203451. 17, 18, 20

HAN, Y.; WEI, Z.; HUANG, G. An imbalance data quality monitoring based on smote-xgboost supported by edge computing. *Scientific Reports*, v. 14, p. 10151, 2024. Disponível em: https://doi.org/10.1038/s41598-024-60600-x. 34

HANCOCK, J. T.; KHOSHGOFTAAR, T. M.; JOHNSON, J. M. Evaluating classifier performance with highly imbalanced big data. *Journal of Big Data*, v. 10, p. 42, 2023. Disponível em: https://doi.org/10.1186/s40537-023-00724-5. 33, 34

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. H. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd. ed. New York: Springer, 2009. Disponível em: https://doi.org/10.1007/978-0-387-84858-7. 17, 18, 20

JORDAN, D. C.; KURTZ, S. R. Photovoltaic degradation rates—an analytical review. *Progress in Photovoltaics: Research and Applications*, v. 21, n. 1, p. 12–29, 2013. Disponível em: https://onlinelibrary.wiley.com/doi/abs/10.1002/pip.1182. 10

JORDAN, J. *Convolutional Neural Networks (CNNs)*. 2018. Acessado em: 7 fev. 2025. Disponível em: https://www.jeremyjordan.me/content/images/2018/02/Screen-Shot-2018-02-24-at-11.47.09-AM.png. 22

MELO, K. B. de; MOREIRA, H. S.; VILLALVA, M. G. Influence of solar position calculation methods applied to horizontal single-axis solar trackers on energy generation. *Energies*, MDPI, v. 13, n. 15, p. 3826, 2020. Disponível em: https://www.mdpi.com/1996-1073/13/15/3826. 10

Ministério de Minas e Energia do Brasil. *Brasil bate recorde de expansão da energia solar em 2023*. [S.l.], 2023. Disponível em: https://www.gov.br/mme/pt-br/assuntos/noticias/brasil-bate-recorde-de-expansao-da-energia-solar-em-2023. 10

NAMBIAR, S. M. et al. Comparison of sun-tracking and fixed solar panel data with python's pvlib classes. In: *2021 4th International Conference on Recent Trends in Computer Science and Technology (ICRTCST)*. [s.n.], 2022. p. 6–11. Disponível em: https://doi.org/10.1109/ICRTCST54752.2022.9781934. 23

NASSREDDINE, G. et al. Fault detection and classification for photovoltaic panel system using machine learning techniques. *Applied AI Letters*, v. 6, n. 2, p. e115, 2025. Disponível em: https://onlinelibrary.wiley.com/doi/abs/10.1002/ail2.115. 25

NATEKIN, A.; KNOLL, A. Gradient boosting machines, a tutorial. *Frontiers in Neurorobotics*, v. 7, p. 1–21, 2013. Disponível em: https://doi.org/10.3389/fnbot.2013.00021. 17, 18, 20

PEREIRA, O. L. S.; GONçALVES, F. F. Dimensionamento de inversores para sistemas fotovoltaicos conectados à rede elétrica: estudo de caso do sistema de tubarão – sc. *Revista Brasileira de Energia*, SBPE, v. 14, n. 1, p. 25–45, 2008. Disponível em: https://sbpe.org.br/index.php/rbe/article/view/214. 13

PRECHELT, L. Early stopping - but when? *Lecture Notes in Computer Science*, 03 2000. Disponível em: https://doi.org/10.1007/978-3-642-35289-8_5. 22

PROJECT JUPYTER. *Jupyter Notebook Documentation*. 2024. Accessed: 2025-06-08. Disponível em: https://jupyter-notebook.readthedocs.io/en/latest/. 23

RAHMAN, T. et al. Investigation of degradation of solar photovoltaics: A review of aging factors, impacts, and future directions toward sustainable energy management. *Energies*, v. 16, n. 9, 2023. ISSN 1996-1073. Disponível em: https://www.mdpi.com/1996-1073/16/9/3706. 10

SALIM, K.; HEBRI, S. A. R.; BESMA, S. Classification predictive maintenance using xgboost with genetic algorithm. *Revue d'Intelligence Artificielle*, IIETA, v. 36, n. 6, p. 833–845, 2022. Disponível em: https://doi.org/10.18280/ria.360603. 24

SIMÃO, T. E. *Criação de frameworks para suporte a análise de performance e quantificação de perda de energia em usinas fotovoltaicas*. 20 p. Monografia (Especialização) — Universidade Federal de Santa Catarina, Florianópolis, 2021. 13

TUDORACHE, T.; KREINDLER, L. Design of a solar tracker system for pv power plants. *Acta Polytechnica Hungarica*, v. 7, n. 1, p. 23–39, abr. 2010. Disponível em: https://www.researchgate.net/publication/45087905_Design_of_a_Solar_Tracker_System_for_PV_Power_Plants. 14, 15

VALENTÍN, D. et al. Failure investigation of a solar tracker due to wind-induced torsional galloping. *Engineering Failure Analysis*, v. 135, p. 3–4, 2022. Disponível em: https://doi.org/10.1016/j.engfailanal.2022.106137. 10, 16

YOUNG, E. et al. A fluid-structure interaction solver for investigating torsional galloping in solar-tracking photovoltaic panel arrays. *Journal of Renewable and Sustainable Energy*, v. 12, n. 6, p. 063503, nov. 2020. ISSN 1941-7012. Disponível em: https://doi.org/10.1063/5.0023757. 16

ZHAO, M. et al. Research on fault diagnosis method for photovoltaic array based on xgboost algorithm. *EAI Endorsed Transactions on Energy Web*, European Alliance for Innovation (EAI), v. 11, n. 4, p. e11, 2024. Disponível em: https://publications.eai.eu/index.php/ew/article/view/7224. 25