UNIVERSIDADE FEDERAL DO CEARÁ
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
PROGRAMA DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

JOEL KALIL DA SILVA PONTES

CALIBRAÇÃO DE DADOS SATELITAIS DE IRRADIAÇÃO SOLAR COM BASE EM MEDIÇÕES IN SITU: UMA ABORDAGEM DE REGRESSÃO LINEAR E CLUSTERIZAÇÃO DE GHI E DHI

FORTALEZA
2025

JOEL KALIL DA SILVA PONTES

CALIBRAÇÃO DE DADOS SATELITAIS DE IRRADIAÇÃO SOLAR COM BASE EM MEDIÇÕES IN SITU: UMA ABORDAGEM DE REGRESSÃO LINEAR E CLUSTERIZAÇÃO DE GHI E DIF

Trabalho de Conclusão de Curso apresentado ao Departamento de Engenharia Elétrica da Universidade Federal do Ceará como requisito parcial à obtenção do grau de bacharel em Engenharia Elétrica.

Orientador: Prof. Dr. Fernando Luiz Marcelo Antunes

FORTALEZA

2025

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

P858c Pontes, Joel Kalil da Silva.
Calibração de dados satelitais de irradiação solar com base em medições in situ : uma abordagem de regressão linear e clusterização de GHI e DIF / Joel Kalil da Silva Pontes. – 2025.
125 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Centro de Tecnologia, Curso de Engenharia Elétrica, Fortaleza, 2025.
Orientação: Profa. Dra. Fernando Luiz Marcelo Antunes.

1. Irradiação solar. 2. Calibração de dados. 3. Regressão linear. 4. Clusterização. I. Título.
CDD 621.3

JOEL KALIL DA SILVA PONTES

CALIBRAÇÃO DE DADOS SATELITAIS DE IRRADIAÇÃO SOLAR COM BASE EM MEDIÇÕES IN SITU: UMA ABORDAGEM DE REGRESSÃO LINEAR E CLUSTERIZAÇÃO DE GHI E DIF

Trabalho de Conclusão de Curso apresentado ao Departamento de Engenharia Elétrica da Universidade Federal do Ceará como requisito parcial à obtenção do grau de bacharel em Engenharia Elétrica.

Aprovada em: ___/___/_____.

BANCA EXAMINADORA

_____
Prof. Dr. Fernando Luiz Marcelo Antunes (Orientador)
Universidade Federal do Ceará (UFC)

_____
Dr. Menaouar Berrehil El Kattel
Universidade Federal do Ceará (UFC)

_____
Eng. Luis Guilherme Bastos de Castro
Casa dos Ventos Desenvolvimento

À minha família, especialmente aos meus pais, que me ensinaram que a educação é o bem mais valioso que podemos herdar.

AGRADECIMENTOS

Agradeço, em primeiro lugar, a todos os professores e referências que tive ao longo da minha trajetória – e que ainda hoje me inspiram. A cada um de vocês, devo parte significativa de quem me tornei. Dirijo um agradecimento especial ao ***Prof. Dr. Fernando Antunes***, pela total solicitude e atenção prestadas como meu orientador, e à ***Casa dos Ventos***, em especial, menciono o Lúcio, Luiz, assim como o time de engenharia solar, pelo apoio inestimável e pela prontidão em colaborar neste projeto.

Minha gratidão estende-se também à ***Universidade Federal do Ceará (UFC)***, onde vivi experiências que me marcaram não apenas academicamente, mas também no âmbito pessoal, graças às amizades construídas ali. À ***École Centrale de Lille***, por ter me acolhido e permitido expandir ainda mais meu conhecimento de mundo, além de proporcionar trocas culturais e acadêmicas riquíssimas. Aos amigos que fiz tanto na UFC quanto na ECL – e fora do ambiente acadêmico –, vocês são parte fundamental da minha formação e lapidação como indivíduo. Em especial, menciono o Caio, o Andreis, o João Lucas e o Mário, meus companheiros de casa durante os dois anos na França, que se tornaram verdadeiros irmãos para mim.

Agradeço também à família da minha namorada, que sempre me tratou como parte do próprio lar. À minha noiva, Hilda Vitória, companheira de vida desde o início do curso, pelo suporte constante em todas as dimensões – acadêmica e pessoal. E, por fim, mas não menos importante, expresso meu profundo reconhecimento à minha família, que jamais deixou de me apoiar, mesmo à distância. Ao meu irmão, cuja presença foi essencial durante a infância, e, sobretudo, aos meus pais, que continuam a ser minhas maiores referências, e a principal razão de eu ser quem sou hoje.

“Douter de tout ou tout croire, ce sont deux solutions également commodes, qui l’une et l’autre nous dispensent de réfléchir.”

(Henri Poincaré)

RESUMO

Este TCC apresenta uma metodologia para a calibração de dados de irradiação solar provenientes de satélite, utilizando medições in situ como referência. O estudo propõe uma abordagem inovadora que combina modelos de regressão linear com técnicas de clusterização para ajustar as componentes de Irradiância Global Horizontal (GHI) e Irradiância Difusa (DIF). Inicialmente, os dados são submetidos a um rigoroso pré-processamento, que inclui a remoção de valores ausentes e negativos, além da correção de inconsistências, garantindo a qualidade da base utilizada para a modelagem. Em seguida, são desenvolvidos modelos estatísticos que correlacionam as medições de campo com as estimativas de satélite, segmentando os dados em função do ângulo zenital solar para capturar comportamentos específicos em diferentes condições. Os resultados evidenciam ganhos significativos na precisão das estimativas, demonstrando que a integração de medições in situ com dados satelitais pode reduzir os vieses inerentes às estimativas e aumentar a confiabilidade para aplicações em dimensionamento de sistemas fotovoltaicos e estudos climáticos. Conclui-se que a abordagem proposta é robusta, viável em diferentes cenários geográficos e atmosféricos, e contribui de forma relevante para o aprimoramento das estimativas de irradiação solar.

Palavras-chave: irradiação solar; calibração de dados; regressão linear; clusterização.

ABSTRACT

This paper presents a methodology for the calibration of solar irradiance data from satellites, using in situ measurements as a reference. The study proposes an innovative approach that combines linear regression models with clustering techniques to adjust the Global Horizontal Irradiance (GHI) and Diffuse Irradiance (DIF) components. Initially, the data are subjected to rigorous pre-processing, which includes the removal of missing and negative values, in addition to the correction of inconsistencies, ensuring the quality of the basis used for modeling. Then, statistical models are developed that correlate the field measurements with the satellite estimates, segmenting the data according to the solar zenith angle to capture specific behaviors in different conditions. The results show significant gains in the accuracy of the estimates, demonstrating that the integration of in situ measurements with satellite data can reduce the biases inherent in the estimates and increase the reliability for applications in photovoltaic system sizing and climate studies. It is concluded that the proposed approach is robust, viable in different geographic and atmospheric scenarios, and contributes significantly to the improvement of solar irradiation estimates.

Keywords: solar irradiation; data calibration; linear regression; clustering.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

MME - Ministério de Minas e Energia

ETS - Espectro Extraterrestre

UV - Ultravioleta

NIR - Infravermelho Próximo

FIR - Infravermelho Distante

TSI - Irradiação Solar Total

UA - Unidade Astronômica

ZEN - Ângulo Zenital Solar

ELV - Ângulo de Elevação Solar

AZ - Ângulo Azimutal Solar

TOA - Topo da Atmosfera

GHI - Irradiância Global Horizontal

DIF - Irradiância Difusa Horizontal

DNI - Irradiância Direta Normal

RHI - Irradiação Refletida Horizontal

POA - Plano de Irradiância da Matriz

AI - Ângulo de Incidência

SVF - Fator de Visão do Céu

GVF - Fator de visão do solo

KD - Coeficiente de Atenuação Difusa

KT - Coeficiente de Proporção Global

IEC - Comissão Eletrotécnica Internacional

ISO - Organização Internacional para Padronização

WMO - Organização Meteorológica Mundial

CAMS - Serviço de Monitoramento Atmosférico Copernicus

NASA - Administração Nacional de Aeronáutica e Espaço

SUMÁRIO

1. INTRODUÇÃO

A crescente demanda por fontes de energia renováveis tem intensificado o interesse em métodos e ferramentas que auxiliem no planejamento e na análise de viabilidade de projetos solares. Nesse contexto, a disponibilidade de dados de irradiação solar desempenha papel fundamental para estimar com maior precisão o potencial energético de uma região, subsidiando a tomada de decisão em termos de dimensionamento de sistemas fotovoltaicos e previsão de geração ao longo do tempo. Atualmente, há uma ampla variedade de dados de satélite que fornecem estimativas de irradiação solar, além de outros parâmetros, como temperatura e velocidade do vento, para praticamente qualquer localidade do planeta, com séries históricas que ultrapassam duas décadas. Essas informações são disponibilizadas tanto de forma gratuita quanto paga, dependendo do fornecedor e do nível de detalhamento requerido.

Apesar da vantagem de oferecer longas séries temporais, os dados satelitais podem apresentar vieses e incertezas decorrentes de limitações de resolução, modelagem e condições atmosféricas não capturadas adequadamente. Por outro lado, a legislação brasileira, por meio do Ministério de Minas e Energia (MME), exige que, para a obtenção da Outorga em projetos de energia, sejam realizadas campanhas de medição in situ: no mínimo 36 meses para projetos eólicos e 12 meses para projetos solares. Esse procedimento é fundamental para comprovar a disponibilidade do recurso renovável (vento ou sol) na região de interesse. Assim, as medições locais tornam-se indispensáveis e, ao mesmo tempo, oferecem a oportunidade de ajuste (calibração) dos dados de satélite, corrigindo eventuais discrepâncias e melhorando a confiabilidade das estimativas.

Essa abordagem é particularmente relevante em projetos de energia solar, pois, diferentemente de outras fontes intermitentes, como a eólica, a campanha de medição obrigatória para caracterização local costuma ser mais curta, porém ainda insuficiente para abarcar toda a variabilidade interanual da irradiação. Ao combinar os dados de satélite, que oferecem séries de longo prazo, com medições de qualidade realizadas in situ, obtém-se um conjunto de informações mais robustas, assegurando maior precisão na previsão de geração de energia e na avaliação de riscos. O presente trabalho, intitulado "***Calibração de Dados Satelitais de Irradiação Solar com Base em Medições In Situ: Uma Abordagem de Regressão Linear e Clusterização de GHI e DIF***", propõe uma metodologia de calibração

por meio de modelos de regressão, com destaque para o papel da clusterização e das especificidades das componentes de irradiação global (GHI) e difusa (DIF). Enquanto a relação entre dados de GHI in situ e satelitais se mostra mais linear, a componente difusa tende a apresentar comportamento menos previsível, demandando técnicas de análise mais elaboradas para a remoção de vieses e consequente aprimoramento da qualidade dos dados.

1.1. Justificativa

A importância de uma base de dados confiável de irradiação solar abrange diversas áreas, como engenharia, meteorologia e climatologia. No setor de energia, essas informações são essenciais para estimativas mais precisas no planejamento de centrais fotovoltaicas. A redução das incertezas nos dados de irradiação reduz o risco financeiro e aumenta a eficiência nos investimentos, especialmente no dimensionamento de módulos solares e na projeção de geração ao longo do ciclo de vida do empreendimento.

Além disso, a ampla disponibilidade de dados de satélite, frequentemente superiores a 20 anos, contrasta com a exigência legal de campanhas de medição in situ, geralmente limitadas a 12 meses para projetos solares. Sem um processo de calibração adequado, essa diferença temporal pode gerar incertezas significativas na avaliação do potencial solar de uma área. Nesse contexto, o uso de modelos de regressão e estratégias de clusterização permite corrigir os vieses dos dados de satélite, fornecendo informações mais alinhadas à realidade local e melhorando a tomada de decisão em projetos de energia solar.

1.2. Objetivos

O presente estudo tem como objetivo principal desenvolver e avaliar diferentes estratégias de calibração de dados de satélite de irradiação solar, considerando tanto as componentes global (GHI) quanto a difusa (DIF). Para cumprir esse propósito, definem-se os seguintes objetivos específicos:

- Coletar e processar os dados de irradiação solar provenientes de medições in situ e de satélite, assegurando a qualidade da base de dados por meio da remoção de valores inconsistentes e da correção de casos incoerentes.

- Investigar padrões e vieses nos dados por meio de análise exploratória (EDA), identificando aspectos como heteroscedasticidade e variações específicas em determinados intervalos do ângulo zenital (ZEN), para fundamentar a escolha dos modelos de calibração.

- Desenvolver e aplicar modelos de regressão com diferentes abordagens, incluindo:

    - Regressão linear sem clusterização: utilizando todo o conjunto de dados de treinamento;
    - Regressão linear com clusterização: segmentando os dados em intervalos de ângulo zenital (clusters) para capturar variações específicas;
    - Regressão polinomial com clusterização: ajustando um modelo polinomial de grau superior (por exemplo, 4º grau) para cada intervalo de ângulo zenital, atendendo a cenários de maior complexidade, sobretudo em faixas de ZEN elevadas.

- Comparar o desempenho das diferentes estratégias de calibração (original, linear global, linear segmentada e polinomial segmentada) por meio de métricas estatísticas de avaliação, tanto em dados de treino quanto em dados de teste, a fim de verificar melhorias na precisão e robustez na estimativa de GHI e DIF.

- Validar o método final selecionado, destacando sua capacidade de manter boa acurácia e generalizar em condições reais de aplicação, evidenciando as melhorias em intervalos críticos de ângulo zenital e a importância da redução de vieses para o setor de energia solar.

1.3. Organização do Trabalho

Este trabalho está dividido em cinco capítulos, conforme descrito a seguir:

- Capítulo 1 - Introdução: Apresenta a contextualização do problema, a

justificativa do estudo, seus objetivos (geral e específicos) e a organização do Trabalho;

- Capítulo 2 - Fundamentação Teórica: Fornece uma revisão de literatura sobre os principais conceitos envolvidos no estudo, como conceitos de irradiação solar, dados e sensores de irradiação e modelos de regressão;
- Capítulo 3 - Metodologia: Detalha os procedimentos adotados para a coleta e preparação dos dados, as ferramentas e tecnologias utilizadas, a arquitetura dos modelos propostos, assim como o processo de treinamento e métricas de avaliação dos modelos;
- Capítulo 4 - Resultados: Apresenta os resultados obtidos durante os experimentos, com uma análise quantitativa das métricas de desempenho do modelo, além de gráficos para comparação dos resultados;
- Capítulo 5 - Conclusão e Trabalhos Futuros: Finaliza o trabalho discutindo as principais conclusões, as limitações encontradas e as possíveis melhorias e desdobramentos futuros do projeto.

2. FUNDAMENTAÇÃO TEÓRICA

A fundamentação teórica que se segue apresenta os principais aspectos relacionados à irradiação solar, começando pela definição de conceitos radiométricos e pela compreensão da geometria solar, passando pela caracterização da constante solar e dos modelos de céu claro (clear-sky). Em seguida, discute-se a importância dos dados e sensores de irradiação, com foco no funcionamento, calibração e manutenção dos equipamentos de medição em superfície e na análise das incertezas envolvidas. Por fim, abordam-se os dados de satélite, seus processos de geração e as limitações inerentes, destacando-se o papel fundamental que essas informações desempenham na estimativa e no monitoramento da radiação solar para diversas aplicações científicas e tecnológicas.

2.1. Conceitos de Irradiação Solar

A radiação solar é a principal fonte de energia que impulsiona os processos climáticos e a vida na Terra, sendo essencial para aplicações que vão desde a geração de energia fotovoltaica até estudos de balanço térmico e meteorologia. Seu entendimento envolve conceitos físicos, geométricos e atmosféricos que descrevem como a energia emitida pelo Sol percorre o espaço e interage com a atmosfera antes de alcançar a superfície terrestre. Na prática, essa análise exige o conhecimento de terminologia radiométricas, do comportamento do Sol como um corpo negro, da geometria que determina a posição solar ao longo do ano e das componentes da irradiação medidas em solo.

Este tópico apresenta os principais fundamentos relacionados à irradiação solar, abordando desde as definições de energia e fluxo radiante (terminologia radiométrica), passando pela conceituação de constante solar e suas variações, até a influência da geometria e da inclinação do eixo terrestre nas estações do ano. Discutem-se também as diferentes formas de irradiância solar — global, direta, difusa e refletida —, fundamentais para aplicações em análise de sistemas fotovoltaicos e climatologia. Por fim, introduzem-se os modelos de céu claro, que oferecem estimativas teóricas para comparação com condições reais, sendo ferramentas valiosas na calibração de dados e na compreensão dos fatores que afetam a disponibilidade de radiação solar em diferentes regiões e condições atmosféricas.

2.1.1. Terminologia Radiométrica

Antes de explorar o tema da radiação solar, é fundamental esclarecer algumas definições básicas relacionadas à energia, ao fluxo, à potência e a outros conceitos associados à radiância. Essas definições estão apresentadas na figura abaixo:

Tabela 1 - Terminologia Radiométrica e Unidades

Quantity	Symbol	Unit	Unit	Description
Radiant energy	Q	Joule	J	Energy
Radiant flux	Φ	Watt	W	Radiant energy per unit of time (radiant power)
Radiant intensity	I	Watt per steradian	W/sr	Power per unit solid angle
Radiant emittance	M	Watt per square meter	W/m²	Power emitted from a surface
Radiance	L	Watt per steradian per square meter	W/(sr·m²)	Power per unit solid angle per unit of projected source area
Irradiance	E	Watt per square meter	W/m²	Power incident on a unit area surface
Spectral irradiance	E_λ	Watt per square meter per nanometer	W/(m²·nm)	Power incident on a unit area surface per unit wavelength
Irradiation	H	Joule per square meter	J/m²	Energy accumulated on a unit area surface during a period; a more practical energy unit is kilowatt-hours per square meter (1 kWh / m² =

Fonte: [2]

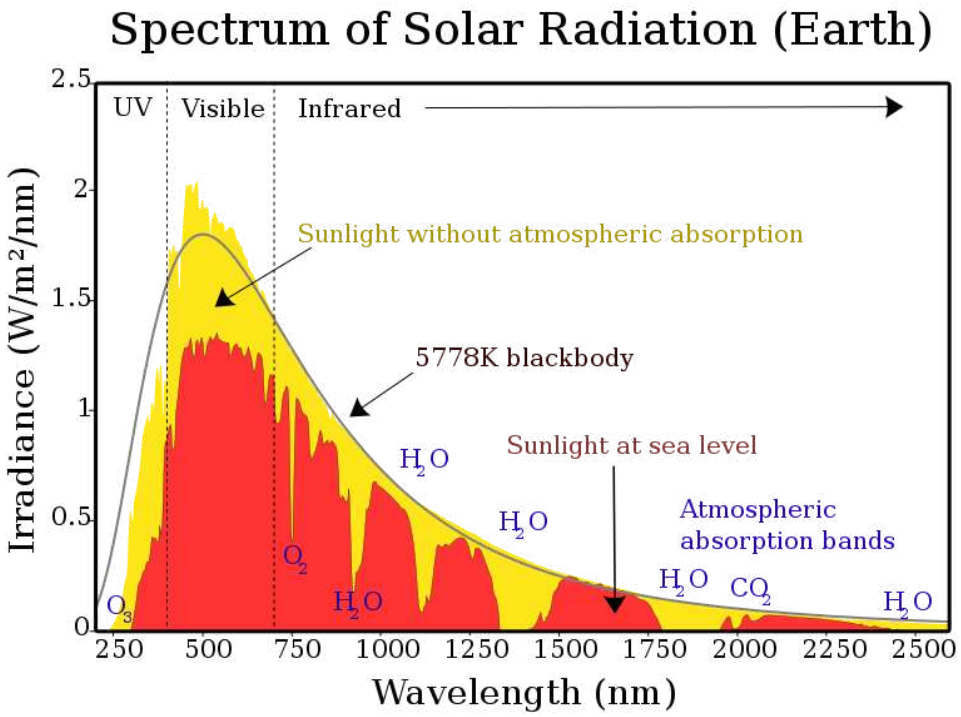
2.1.2. Irradiância Extraterrestre

Na física, é amplamente conhecido que qualquer objeto com temperatura absoluta acima de zero Kelvin emite radiação eletromagnética, fenômeno que ocorre devido ao movimento térmico das partículas em sua estrutura. O Sol, com sua temperatura de superfície estimada em aproximadamente 5800 K, comporta-se de maneira muito semelhante a um corpo negro ideal.

O conceito de corpo negro na física refere-se a um objeto ideal que absorve toda a radiação eletromagnética incidente, sem refletir ou transmitir nenhuma parte. Em contrapartida, ele emite radiação de maneira perfeitamente previsível, dependendo apenas de sua temperatura. A intensidade e a distribuição da radiação emitida por um corpo negro são descritas pela lei de Planck, que estabelece como a energia emitida varia com o comprimento de onda e a temperatura. Embora nenhum objeto no universo seja um corpo negro perfeito, o Sol se aproxima desse comportamento devido à sua emissão em um amplo espectro de comprimentos de onda.

A radiação solar abrange comprimentos de onda entre 290 nm e 4000 nm, faixa conhecida como espectro extraterrestre (ETS). Esse intervalo inclui três regiões principais do espectro eletromagnético: ultravioleta (UV), luz visível e o início do infravermelho próximo (NIR, do inglês near-infrared). A distribuição desse espectro, frequentemente utilizada em estudos sobre energia solar, pode ser visualizada na Figura 2. Estudos recentes, como o de Gueymard (2018a), indicam que aproximadamente 98,5% da irradiação solar extraterrestre está concentrada nessa faixa de 290-4000 nm. O restante da radiação está presente nas regiões mais distantes do espectro infravermelho, conhecidas como infravermelho distante (FIR, do inglês far-infrared).

Figura 1 - Espectro da radiação solar na Terra



Fonte: Wikimedia Commons

Compreender o comportamento do Sol como um corpo negro e a distribuição do seu espectro é essencial para o desenvolvimento de tecnologias de captação de energia solar. Esse conhecimento permite determinar não apenas a quantidade de energia disponível para conversão em eletricidade ou calor, mas também quais partes do espectro podem ser melhor aproveitadas em diferentes aplicações.

2.1.3. Constante Solar

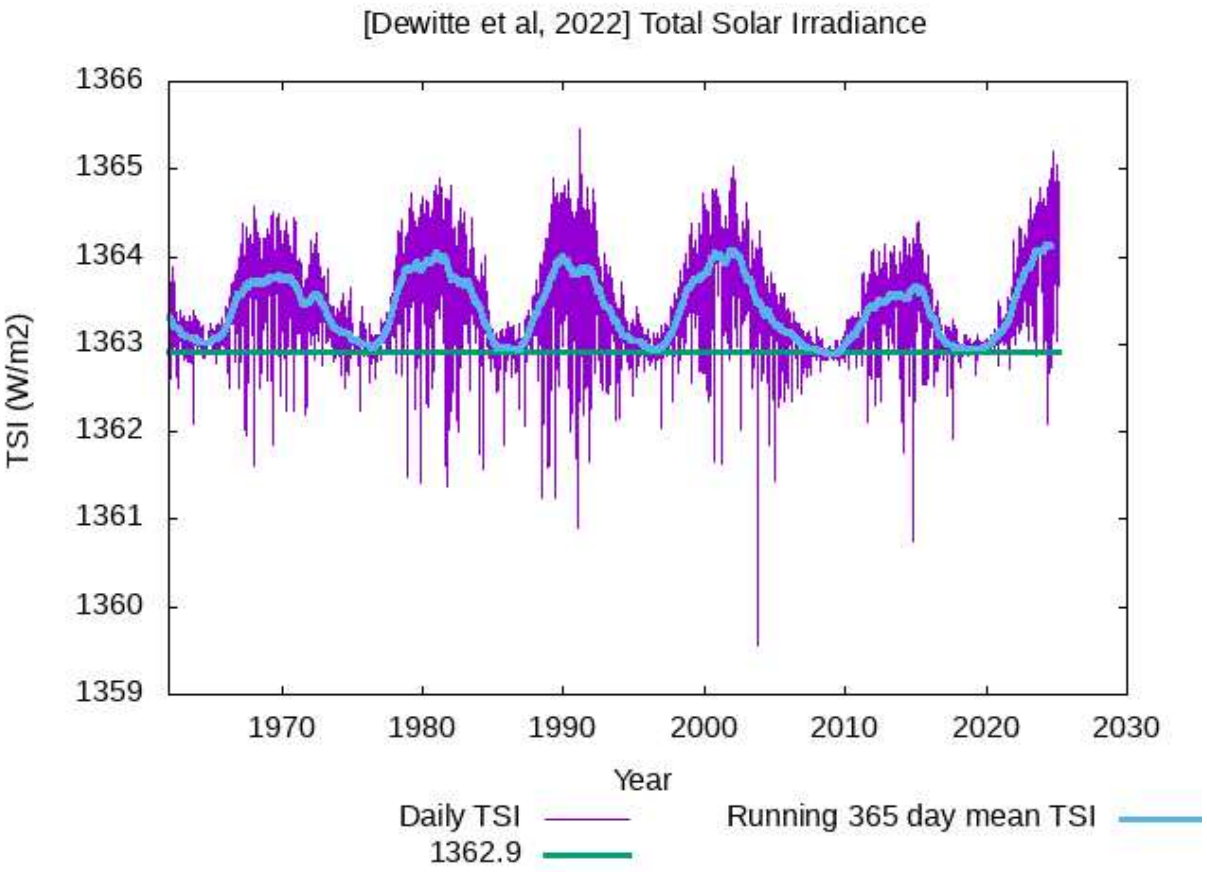
A potência radiante total emitida pelo Sol, denominada como irradiação solar total (ou Total Solar Irradiance - TSI), representa a quantidade de energia emitida por unidade de tempo em todas as direções do espaço. Embora a TSI seja, em grande parte, constante ao longo do tempo, medições precisas realizadas por satélites detectaram variações temporais sutis.

A TSI é calculada pela integração do espectro extraterrestre (ETS) quando a distância entre o Sol e a Terra é igual a 1 Unidade Astronômica (UA), aproximadamente 149,6 milhões de quilômetros. Inicialmente, acreditava-se que a emissão do Sol era perfeitamente constante, motivo pela qual o termo "constante solar" foi amplamente utilizado para descrever o valor médio dessa radiação recebida no topo da atmosfera terrestre. No entanto, com avanços tecnológicos e observações mais precisas, descobriu-se que a TSI apresenta pequenas flutuações temporais, relacionadas a ciclos solares, atividades de manchas solares e outras variações na dinâmica do Sol.

Atualmente, a constante solar é definida como a média de longo prazo da TSI e representa a energia solar incidente perpendicularmente a uma unidade de área no topo da atmosfera terrestre, a uma distância média de 1 UA. Embora pequenas, essas flutuações são relevantes para o estudo do clima e das interações entre o Sol e a Terra, pois até mesmo variações mínimas podem afetar a atmosfera terrestre e, a longo prazo, o balanço energético do planeta.

Historicamente, diferentes valores foram atribuídos à constante solar à medida que medições e métodos de cálculo evoluíram. Um dos valores mais recentes foi proposto por Gueymard (2018b), que, após uma reavaliação abrangente e a correção de décadas de dados obtidos por satélites, estabeleceu a constante solar em 1361,1 W/m². Este valor é amplamente utilizado em estudos climáticos, meteorológicos e em projetos de aproveitamento da energia solar.

Figura 2 - Variações Temporais do TSI



Fonte: Solar Influences Data Analysis Center - Total Solar Irradiance

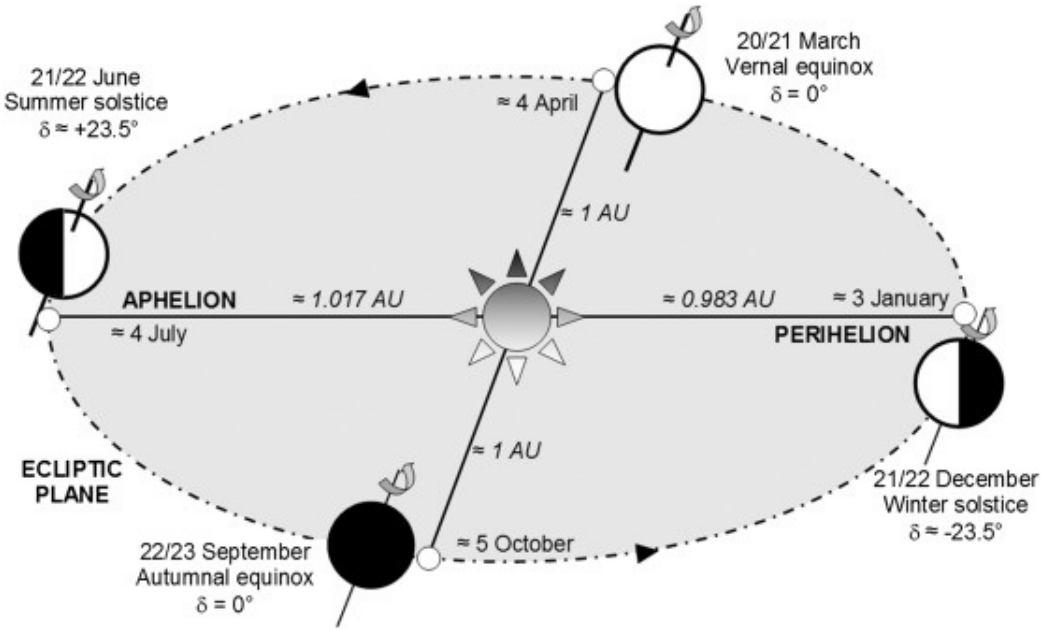
A constância relativa da TSI, combinada com o conhecimento das pequenas variações cíclicas, é um elemento crucial para compreender não apenas o fluxo de energia recebido pela Terra, mas também sua influência no sistema climático global. Estudos contínuos sobre a TSI e suas variações contribuem significativamente para o entendimento de fenômenos como aquecimento global, mudanças climáticas e a evolução dos ciclos solares.

2.1.4. Geometria Solar

Ao longo do ano, a Terra percorre uma órbita elíptica ao redor do Sol, resultando em mudanças contínuas na posição relativa entre o nosso planeta e a estrela central. Esse movimento orbital, aliado à inclinação do eixo terrestre, provoca variações sazonais nas condições climáticas e nos padrões de radiação solar que atingem diferentes regiões da superfície terrestre. Essas variações influenciam diretamente fenômenos como a duração dos dias e noites, as estações do ano e a intensidade da radiação solar incidente.

A órbita da Terra ao redor do Sol é ligeiramente elíptica, com o Sol localizado em um dos focos da elipse. Como resultado, a distância entre a Terra e o Sol varia ao longo do ano, influenciando levemente a quantidade de energia solar recebida. Entretanto, a inclinação do eixo terrestre (aproximadamente 23,5° em relação à perpendicular do plano orbital) tem o papel mais significativo na definição das estações do ano, pois altera a incidência dos raios solares nos diferentes hemisférios.

Figura 3 - Órbita da Terra



Fonte: Wikipedia

Para compreender o comportamento do Sol em relação à Terra ao longo das estações, utilizam-se definições angulares específicas que permitem descrever e calcular sua posição no céu. Esses ângulos são essenciais para determinar a irradiação solar em diferentes momentos e locais, viabilizando análises detalhadas sobre a distribuição da radiação e seu impacto no clima, na agricultura e nas tecnologias solares. Os principais ângulos utilizados são:

- ***Declinação Solar (δ):*** É o ângulo formado entre o plano do equador terrestre e a linha imaginária que conecta o centro do Sol ao centro da Terra. A declinação solar varia ao longo do ano, atingindo aproximadamente +23,5° no solstício de verão do hemisfério norte e -23,5° no solstício de inverno.

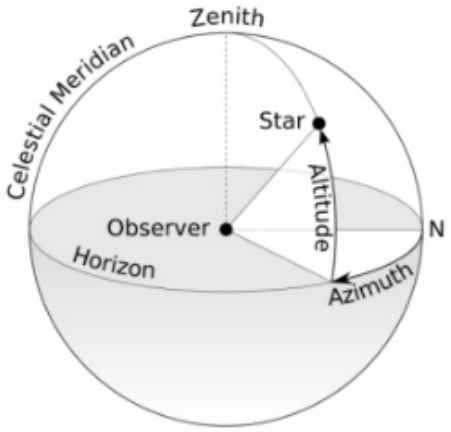
- ***Ângulo Zenital Solar (ZEN):*** É o ângulo formado entre a direção do Sol e a vertical local (uma linha imaginária perpendicular ao solo em um ponto específico). Esse ângulo indica o quão alto ou baixo o Sol está no céu em relação ao observador.

- ***Ângulo de Elevação Solar (ELV):*** É o ângulo formado entre a direção do Sol e o plano do horizonte. Ele indica a altura do Sol no céu em relação ao horizonte e está relacionado ao ângulo zenital pela equação:

$$ZEN + ELV = 90° \quad (1)$$

- ***Ângulo Azimutal Solar (AZ):*** Também conhecido como azimute solar, é o ângulo formado entre a projeção da direção do Sol no plano horizontal e a direção do norte. Ele descreve a posição do Sol em relação aos pontos cardeais.

Figura 4 - Ângulos do Sol em relação à Terra



Fonte: Wikipédia

Esses ângulos não apenas ajudam a descrever a trajetória do Sol no céu ao longo do dia, mas também são cruciais para o cálculo da irradiação solar em superfícies inclinadas ou horizontais. Por exemplo, a declinação solar é necessária para determinar o ângulo de incidência da radiação em diferentes latitudes, enquanto os ângulos zenital e de elevação ajudam a modelar a intensidade e a duração da radiação recebida em um local específico.

2.1.5. Irradiância Solar

A irradiância solar corresponde à quantidade de energia radiante do Sol incidente sobre uma determinada superfície, por unidade de área e tempo, geralmente expressa em W/m². Ao atravessar a atmosfera até alcançar a superfície terrestre, a radiação solar sofre diversos processos, como absorção por gases (ozônio, vapor d'água etc.), espalhamento por moléculas e partículas em suspensão, e reflexão em superfícies. Esses processos afetam diretamente a quantidade e o tipo de radiação que alcança o solo.

A irradiância solar é classificada em diferentes componentes, dependendo do ponto de análise:

- ***No topo da atmosfera (TOA):***

    - ***Constante Solar:*** Representa a irradiância máxima recebida pela Terra antes de interações atmosféricas. É considerada a radiação incidente perpendicularmente ao plano da atmosfera a uma distância média de 1 Unidade Astronômica (UA).

- ***Na superfície terrestre:***

    - ***Irradiação Global Horizontal (GHI):*** Corresponde à irradiação solar total incidente sobre uma superfície horizontal no solo, englobando tanto a componente direta quanto a difusa.
    - ***Irradiação Direta Normal (DNI):*** Corresponde à irradiação solar direta incidente sobre uma superfície perpendicular à direção do Sol, sem considerar a contribuição da irradiação difusa ou refletida.
    - ***Irradiação Difusa Horizontal (DIF):*** Corresponde à fração da irradiação solar que atinge o solo de forma indireta, após sofrer espalhamento na atmosfera.
    - ***Irradiação Refletida Horizontal (RHI):*** É a irradiação refletida a partir do solo ou outras superfícies, influenciada pelo albedo do solo, sendo a

fração de irradiação refletida em relação à irradiação incidente.

Essas componentes solares estão interligadas e podem ser descritas por relações matemáticas envolvendo o ângulo zenital solar (ZEN), entre outras variáveis. As principais relações são:

- ***Para radiação global horizontal:***

$$GHI = DNI \times cos(ZEN) + DIF \quad (2)$$

- ***Para radiação global inclinada no plano de um painel solar:***

$$POA = DNI \times cos(AI) + DIF \times SVF + RHI \times GVF \quad (3)$$

Onde:

- ***POA (Plane of Array Irradiance):*** Irradiância no plano inclinado do painel solar.
- ***AI (Angle of Incidence):*** Ângulo de incidência da radiação solar sobre a superfície inclinada.
- ***SVF (Sky View Factor):*** Fator que representa a fração do céu visível a partir de uma superfície.
- ***GVF (Ground View Factor):*** Fator que representa a fração de radiação visível do solo em relação a uma superfície.

Além das relações mencionadas, dois coeficientes são amplamente utilizados em modelos e algoritmos para descrever a distribuição e proporção da radiação solar:

- ***Coeficiente de Atenuação Difusa (KD):***

Esse coeficiente mede a fração de radiação difusa em relação à radiação global na superfície terrestre. Um valor alto de KA indica maior proporção de radiação difusa em relação à radiação total recebida. É calculado pela relação:

$$KD = DIF \div GHI \quad (4)$$

- ***Coeficiente de Proporção Global (KT):***

O coeficiente Kt avalia a quantidade de radiação solar efetivamente recebida na superfície terrestre em relação à radiação disponível no topo da atmosfera. Ele considera os efeitos de geometria solar e atenuação atmosférica. Sua fórmula é dada por:

$$KT = GHI \div (TOA \times cos(ZEN)) \quad (5)$$

Esses coeficientes são particularmente úteis para estudar o impacto da atmosfera na distribuição da radiação solar, bem como para o dimensionamento e a análise de desempenho de sistemas solares.

O entendimento dessas componentes e relações é essencial para a modelagem da radiação solar em diferentes condições climáticas, latitudes e configurações de sistemas solares, permitindo otimizar o aproveitamento da energia solar em aplicações práticas.

2.1.6. Modelos de Céu Claro (Clear-Sky)

Os modelos de céu claro (clear-sky models) são ferramentas essenciais no estudo da energia solar, fornecendo estimativas teóricas da radiação solar em condições ideais, ou seja, na ausência de nuvens e com atmosfera limpa. Esses modelos são amplamente utilizados para comparar dados reais de irradiância com valores ideais, calcular o índice de clareza (clearness index, Kt) e validar medições de irradiância solar. Entre os vários modelos disponíveis, o modelo de Ineichen é particularmente popular devido à sua precisão e simplicidade, sendo o modelo adotado nesta pesquisa.

O modelo de Ineichen é um modelo empírico de céu claro amplamente utilizado para estimar as componentes da irradiância solar: irradiância global horizontal (GHI), irradiância direta normal (DNI) e irradiância difusa horizontal (DIF). Ele considera fatores como a posição solar, massa de ar, turbidez atmosférica e altitude para gerar estimativas confiáveis da radiação solar sob condições de céu claro.

- Principais Parâmetros do Modelo

- ***Ângulo zenital aparente:*** Reflete a posição do Sol no céu e é crucial para o cálculo das componentes de irradiância.
- ***Massa de ar relativa e absoluta:*** Representa o caminho percorrido pela radiação solar através da atmosfera. A massa de ar relativa é dependente do ângulo zenital, enquanto a absoluta é ajustada pela pressão atmosférica.
- ***Turbidez de Linke:*** Um parâmetro que caracteriza a transparência da atmosfera, influenciada por partículas, vapor d'água e aerossóis.
- ***Altitude do local:*** Afeta a pressão atmosférica e, consequentemente, a atenuação da radiação solar.
- ***Irradiação extraterrestre:*** A radiação solar incidente no topo da atmosfera, considerada como referência em condições ideais.

O modelo de Ineichen utiliza uma combinação de relações empíricas para calcular GHI, DNI e DHI. Essas relações são ajustadas para incluir os efeitos da turbidez atmosférica (parâmetro de Linke) e da altitude, garantindo estimativas robustas em diferentes condições geográficas.

2.2. Dados e Sensores de Irradiação Solar

2.2.1. Sensores de Superfície

Os sensores de superfície são instrumentos fundamentais para medir e monitorar as componentes da radiação solar que atingem a superfície terrestre. Entre os mais utilizados estão os piranômetros, os pireliômetros e os albedômetros, que permitem medir a radiação solar global, direta e refletida, respectivamente. A seguir, detalha-se o funcionamento dos piranômetros e pireliômetros, além de questões importantes como calibração, manutenção e fatores que afetam as medições.

2.2.1.1. Funcionamento dos Sensores

A medição dos diferentes componentes da radiação solar, sejam eles diretos, difusos ou refletidos, é fundamental para uma série de aplicações, incluindo estudos climáticos, projetos de energia solar e pesquisas em balanço de energia na superfície terrestre.

Nesse contexto, surgem diferentes tipos de sensores, cada um projetado para mensurar um aspecto específico do espectro solar. Na sequência, apresenta-se o funcionamento dos principais dispositivos utilizados para esses fins: piranômetro, pireliômetro e albedômetro.

O piranômetro é destinado à medição da irradiância global incidente em uma superfície plana. Essa medição abrange tanto a radiação direta proveniente do Sol quanto a radiação difusa resultante do espalhamento atmosférico. O componente central do piranômetro é um sensor térmico (usualmente baseado em termopilha ou termopar), responsável por converter a radiação incidente em um sinal elétrico diretamente proporcional à intensidade luminosa recebida. Esse sensor é recoberto por uma cúpula de vidro ou quartzo, cuja função é permitir a passagem de radiação em uma ampla faixa espectral, incluindo tanto a região do visível quanto a do infravermelho próximo. Dessa forma, o piranômetro fornece uma leitura confiável e abrangente do fluxo de energia solar na superfície analisada.

Em contraste com o piranômetro, cujo princípio consiste em captar a radiação global, o pireliômetro foi concebido especificamente para medir a radiação direta normal (DNI). Para tal, o equipamento permanece apontado diretamente para o Sol, geralmente por meio de um rastreador solar que ajusta seu alinhamento ao longo do dia. Este arranjo garante que a medição seja livre de componentes difusos, possibilitando uma avaliação precisa da fração direta da irradiância. Dessa forma, o pireliômetro é fundamental em estudos que demandam uma análise mais detalhada da contribuição direta do Sol, a exemplo de aplicações em concentradores solares.

O albedômetro, por sua vez, está voltado à determinação do albedo de uma superfície, definindo-se este como a razão entre a radiação refletida e a radiação solar incidente. Na prática, o albedômetro consiste em dois piranômetros acoplados: um voltado para cima, medindo a irradiância global incidente, e outro orientado para baixo, capturando a radiação refletida pela superfície subjacente. A partir dessas duas leituras, é possível calcular o albedo, quantificando o grau de refletividade de diferentes materiais ou formações naturais (por exemplo, gelo, vegetação, solo exposto, etc.). Em estudos climáticos, essa grandeza é essencial para estimar o balanço de energia e avaliar o impacto de mudanças na cobertura terrestre.

2.2.1.2. Padrões Internacionais de Medição

A precisão e confiabilidade das medições com piranômetros e pireliômetros dependem de seguir padrões internacionais amplamente aceitos, como os definidos pela ***IEC 61274-1***, que referencia os seguintes normativos importantes:

- ***ISO 9060:2018***

Define e classifica piranômetros e pireliômetros com base em suas características de desempenho. Os instrumentos são divididos em categorias como classe secundária, primeira classe e padrão secundário (secondary standard), de acordo com critérios como resposta espectral, resposta direcional e linearidade.

Tabela 2 - Tabela de classificação dos piranômetros pela ISO 9060 e WMO

ISO CLASSIFICATION TABLE**			
ISO CLASS	SECONDARY STANDARD	FIRST CLASS	SECOND CLASS
Specification limit			
Response time (95 %)	15 s	30 s	60 s
Zero offset a (response to 200 W/m² net thermal radiation)	+ 7 W/m²	+ 15 W/m²	+ 30 W/m²
Zero offset b (response to 5 K/h in ambient temperature)	± 2 W/m²	± 4 W/m²	± 8 W/m²
Non-stability (change per year)	± 0.8 %	± 1.5 %	± 3 %
Non-linearity (100 to 1000 W/m²)	± 0.5 %	± 1 %	± 3 %
Directional response	± 10 W/m²	± 20 W/m²	± 30 W/m²
Spectral selectivity (350 to 1 500 x 10^{-9} m) (WMO 300 to 3 000 x 10^{-9} m)	± 3 %	± 5 %	± 10 %
Temperature response (interval of 50 K)**	2 %	4 %	8 %
Tilt response (0 to 90 ° at 1000 W/m²)	± 0.5 %	± 2 %	± 5 %
ADDITIONAL WMO SPECIFICATIONS			
WMO CLASS	HIGH QUALITY	GOOD QUALITY	MODERATE QUALITY
WMO: achievable accuracy for daily sums*	2 %	5 %	10 %
WMO: achievable accuracy for hourly sums*	3 %	8 %	20 %
WMO: achievable accuracy for minute sums*	not specified	not specified	not specified
WMO: resolution (smallest detectable change)	1 W/m²	5 W/m²	10 W/m²
CONFORMITY TESTING***			
ISO 9060	individual instrument only: all specs must comply	group compliance	group compliance

Fonte: [11]

- ***ISO 9847:1992***

Especifica os procedimentos para calibração de piranômetros em campo por comparação com um piranômetro de referência. Este padrão cobre a calibração em condições controladas internas.

- ***ISO 9846:1993***

Detalha a calibração de piranômetros utilizando pireliômetros como referência em medições externas. Este método é frequentemente usado para garantir a rastreabilidade das medições a padrões internacionais.

2.2.1.3. Calibração e Manutenção

A confiabilidade e a precisão dos dados obtidos na medição de radiação solar dependem, em grande medida, de um programa rigoroso de calibração. Em conformidade com normas internacionais, recomenda-se que a calibração seja realizada periodicamente, incluindo tanto procedimentos em laboratório (calibração interna) quanto avaliações de campo (calibração externa). Na calibração interna, utilizam-se fontes-padrão em condições controladas, permitindo verificar a sensibilidade do sensor e identificar possíveis desvios sistemáticos. Já a calibração externa consiste na comparação do instrumento (piranômetro ou pireliômetro) com um dispositivo de referência sob luz solar natural, possibilitando a aferição do desempenho em condições reais de operação.

Além da calibração, a manutenção dos sensores é fundamental para garantir medições precisas ao longo do tempo. A limpeza periódica das cúpulas (de vidro ou quartzo) é uma das etapas mais críticas, pois a presença de poeira ou outras partículas sobre a superfície de medição pode atenuar a transmissão de radiação, comprometendo a exatidão dos valores registrados. No caso dos pireliômetros, a verificação constante do alinhamento com o Sol é igualmente relevante, uma vez que qualquer desvio angular pode introduzir erros significativos nas medições da irradiação direta normal (DNI). Por fim, recomenda-se a realização de recalibrações sistemáticas, geralmente em intervalos anuais ou bienais, uma vez que o envelhecimento natural dos componentes – especialmente aqueles sensíveis à radiação, como termopilhas e termopares – pode alterar gradualmente a resposta do instrumento ao longo do tempo.

2.2.1.4. Fatores que Afetam as Medições

A precisão e a confiabilidade das medições de radiação solar estão sujeitas a diversos fatores que podem, direta ou indiretamente, influenciar o desempenho dos sensores. Um destes fatores é o acúmulo de poeira ou sujeira na superfície de medição, comum em ambientes com alta concentração de partículas em suspensão, o que pode reduzir sensivelmente a transmissão da radiação incidente. Além disso, o envelhecimento natural dos materiais que compõem o sensor provoca degradações em sua sensibilidade ao longo do tempo, fazendo com que a resposta inicial se modifique gradualmente. Outro aspecto relevante é a formação de condensação ou gelo sobre a cúpula de proteção, o que dificulta a passagem da radiação e pode gerar leituras subestimadas.

Por fim, condições climáticas extremas, tais como ventos intensos, chuvas torrenciais ou variações bruscas de temperatura, podem comprometer a estabilidade física do equipamento, resultando em possíveis erros de alinhamento ou danos em componentes sensíveis.

2.2.1.5. Incerteza nas Medições

Mesmo com procedimentos de calibração e manutenção em dia, as medições efetuadas por piranômetros e pireliômetros estão intrinsecamente associadas a determinados graus de incerteza. Um dos fatores determinantes é a classe do instrumento: dispositivos classificados como padrão secundário tendem a apresentar margens de erro menores em comparação aos de primeira classe ou classe secundária, devido às especificações mais rigorosas de fabricação e calibração.

A resposta direcional do sensor também afeta as medições, uma vez que variações no ângulo de incidência da radiação podem introduzir discrepâncias entre o valor real e o valor medido. Além disso, as condições ambientais locais, incluindo flutuações de temperatura, pressão atmosférica e umidade, exercem influência tanto sobre o próprio sensor quanto sobre a propagação da radiação, resultando em pequenos desvios nos dados registrados.

2.2.2. Dados de Satélite

O uso de dados de satélite é essencial para estimar a irradiância solar em locais onde não há medições terrestres diretas disponíveis. Essa abordagem combina observações de satélites geoestacionários ou orbitais com modelos atmosféricos para calcular componentes como irradiância global horizontal (GHI), irradiância difusa (DIF) e irradiância direta normal (DNI). A seguir, são descritos os principais passos do processo de geração desses dados, bem como os desafios e limitações associados.

2.2.2.1. Processo de Geração de Dados de Satélite

A obtenção de estimativas de irradiância a partir de sensores em órbita envolve um conjunto estruturado de etapas que integram observações atmosféricas, modelos de transferência radiativa e parâmetros meteorológicos globais. Inicialmente, realiza-se a modelagem de céu claro, na qual se calculam as irradiâncias de GHI, DNI e DIF, assumindo a ausência de nuvens. Esse processo depende de parâmetros como a massa de ar — definida conforme o ângulo zenital solar — e de propriedades atmosféricas, incluindo teor de vapor d'água, presença de aerossóis e a concentração de ozônio. Modelos empíricos e físicos, tais como Ineichen ou Bird, são frequentemente empregados nessa etapa para fornecer uma estimativa inicial das condições ideais de irradiância.

Em seguida, passa-se à detecção e análise de nuvens, por meio do processamento de imagens de satélite provenientes, por exemplo, de sensores MODIS (NASA) ou EUMETSAT. Esse procedimento visa identificar cobertura, altura e espessura das nuvens, utilizando algoritmos de limiar ("thresholding") ou classificadores baseados em aprendizado de máquina para segmentar as diferentes regiões cobertas. As informações sobre nuvens obtidas nesse estágio são, então, incorporadas aos modelos de transferência radiativa, ajustando as estimativas de irradiância para condições reais de nebulosidade.

A correção de aerossóis e vapor d'água é outro fator crucial, pois ambos afetam de forma significativa a atenuação da radiação solar. Dados relacionados à concentração de aerossóis são usualmente extraídos de re-análises atmosféricas globais, como as fornecidas pelo Copernicus Atmosphere Monitoring Service (CAMS), enquanto o conteúdo de vapor d'água é estimado a partir de informações registradas por sensores infravermelhos instalados

em satélites. Assim, ajusta-se a transmissão da radiação conforme a presença dessas partículas em suspensão e a quantidade de umidade no ar.

Após essas correções, procede-se ao cálculo das componentes de irradiância:

- A GHI é obtida pela soma das contribuições direta e difusa, corrigidas pelos efeitos atmosféricos.
- A DNI representa a parte do fluxo solar que não foi dispersa pela atmosfera.
- A DIF é estimada como a diferença entre GHI e DNI, ajustada pelo ângulo zenital.

Por fim, realiza-se a validação e ajuste dos modelos com base em comparações entre as estimativas de satélite e medições de superfície captadas por piranômetros e pireliômetros. A partir dessas comparações, realiza-se a calibração final dos modelos, adequando-os às condições atmosféricas específicas de cada região e garantindo maior confiabilidade nos valores de irradiância estimados.

2.2.2.2. Problemas e Limitações dos Dados de Satélite

Embora os produtos de satélite proporcionem ampla cobertura espacial e temporal, tornando-se ferramentas valiosas para o estudo da radiação solar em escala global, sua utilização está sujeita a algumas restrições. O chamado efeito nugget ilustra bem esse cenário: em escalas espaciais muito pequenas, podem surgir discrepâncias significativas entre medições terrestres pontuais e estimativas derivadas de satélite. Isso ocorre em razão da resolução espacial dos sensores orbitais (em geral de 1 a 3 km) não captarem a variabilidade local de terreno, nuvens ou aerossóis, especialmente em regiões de topografia complexa ou dinâmica atmosférica intensa.

A resolução espacial e temporal constitui, portanto, outro desafio. Embora intervalos de 15 minutos a 1 hora sejam adequados para muitas aplicações, determinadas análises — como a avaliação de eventos de curta duração, a exemplo de passagens rápidas de nuvens — exigem uma frequência de coleta mais elevada. Além disso, a dimensão dos pixels de satélite pode ser insuficiente para mapear detalhadamente pequenas áreas urbanizadas ou regiões com variação acentuada no uso e cobertura do solo.

As incertezas de modelagem também não devem ser subestimadas. O êxito da estimativa depende tanto da precisão dos modelos de transferência radiativa quanto da qualidade dos bancos de dados atmosféricos sobre aerossóis, vapor d'água e outros constituintes. Modelos empíricos podem falhar em regiões com condições muito particulares, como áreas industriais com alta densidade de partículas. Por fim, a influência de nuvens constitui outro elemento crítico: determinar a espessura óptica de forma precisa a partir de imagens de satélite é complexo, especialmente em cenários com nebulosidade densa ou em zonas costeiras, onde há frequente formação de nuvens baixas. Essas limitações salientam a importância de procedimentos de validação contínuos e de ajustes regionais para reduzir incertezas e garantir que os dados de satélite sejam efetivamente representativos das condições locais.

2.3. Modelos de Regressão

2.3.1. Regressão Linear

A *Regressão Linear* é uma técnica estatística fundamental que visa modelar a relação entre uma variável dependente (geralmente denotada por y) e uma ou mais variáveis independentes (geralmente denotadas por x_1, x_2, ..., x_n). No caso mais simples, em que há apenas uma variável preditora (também chamada variável explicativa), chamamos de *Regressão Linear Simples*.

2.3.1.1. Modelagem Matemática

Na regressão linear simples, assumimos que existe uma relação aproximadamente linear entre x e y. Dessa forma, podemos escrever o modelo como:

$$y = \beta_0 + \beta_1 \cdot x + \varepsilon \tag{6}$$

onde:

- β_0 é o coeficiente linear ou intercepto (o valor de y quando $x = 0$),
- β_1 é o coeficiente angular (indica a inclinação da reta e, portanto, a relação de variação

de y em função de x),

- ε é o termo de erro ou ruído, que representa a discrepância entre o valor observado e o valor previsto pelo modelo.

Já na *Regressão Linear Múltipla*, quando se tem n variáveis preditoras $x_1, x_2, \ldots, x_n$, o modelo é escrito como:

$$y = \beta_0 + \beta_1 \cdot x + \beta_2 \cdot x_2 + \ldots + \beta_n \cdot x_n + \varepsilon \quad (7)$$

2.3.1.2. Ajuste do Modelo (Método dos Mínimos Quadrados)

O método mais comum para ajustar (estimar os coeficientes $\beta_0, \beta_1, \ldots$) é o *Método dos Mínimos Quadrados Ordinários (OLS)*. Ele busca minimizar a soma dos quadrados dos resíduos (diferença entre valores observados e valores preditos):

$$min_{\beta_0, \beta_1, \ldots, \beta_n} \sum_{i=1}^{N} (y_i - \widehat{y_i})^2 \quad (8)$$

onde:

- N é o número de observações,
- y_i é o valor real da variável dependente para a i-ésima observação,
- $\widehat{y_i}$ é o valor previsto pelo modelo para a i-ésima observação.

Em termos matriciais, podemos representar todas as observações por:

$$y = X \cdot \beta + \varepsilon \quad (9)$$

em que:

- y é o vetor $(N \times 1)$ com os valores observados,
- X é a matriz $(N \times (n + 1))$ que contém uma coluna de 1s (para o intercepto) e as demais colunas correspondentes às variáveis preditoras,

- β é o vetor $(n + 1 \times 1)$ de coeficientes (incluindo β_0),
- ε é o vetor $(N \times 1)$ de erros.

A solução ótima em OLS é dada por:

$$\widehat{\beta} = (X^T X)^{-1} X^T y \tag{10}$$

desde que $X^T X$ seja invertível (ou pseudo-inversa em casos mais complexos).

2.3.2. Regressão Polinomial

A *Regressão Polinomial* é uma extensão natural da regressão linear quando a relação entre a(s) variável(is) independente(s) e a variável dependente y não é adequadamente capturada por uma reta (função linear simples). Em vez de ajustarmos uma reta, ajustamos uma *função polinomial* para representar melhor a curvatura dos dados.

2.3.2.1. Modelagem Matemática

Para um único preditor x, o modelo de regressão polinomial de grau d pode ser escrito como:

$$y = \beta_0 + \beta_1 \cdot x^1 + \beta_2 \cdot x^2 + \beta_3 \cdot x^3 + \ldots + \beta_d \cdot x^d + \varepsilon \tag{11}$$

Note que, apesar de ser chamado "polinomial", ainda trata-se de um modelo linear do ponto de vista dos coeficientes $\beta_0, \beta_1, \ldots, \beta_d$. A não linearidade está na transformação das variáveis $(x, x^2, x^3, \ldots)$.

2.3.2.2. Forma Matricial

Assim como na regressão linear simples ou múltipla, podemos escrever o modelo polinomial de forma matricial. Suponha que tenhamos N observações para a variável x. Definimos uma matriz de projeto X_{poly} da seguinte forma:

$$X_{\text{poly}} = \begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^d \\ 1 & x_2 & x_2^2 & \dots & x_2^d \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_N & x_N^2 & \dots & x_N^d \end{bmatrix} \tag{12}$$

e o vetor de parâmetros:

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_d \end{bmatrix} \tag{13}$$

O modelo é:

$$y = X_{poly} \cdot \beta + \varepsilon \tag{14}$$

O processo de ajuste (estimação de β) por Mínimos Quadrados Ordinários continua o mesmo, resultando na solução:

$$\hat{\beta} = (X^T{}_{poly} \cdot X_{poly})^{-1} \cdot X^T{}_{poly} \cdot y \tag{15}$$

2.3.2.3. Escolha do Grau do Polinômio

A escolha do grau d do polinômio é crítica:

- Um polinômio de grau muito baixo pode sub-ajustar (underfitting), não capturando a curvatura real dos dados.
- Um polinômio de grau muito alto pode sobre-ajustar (overfitting), resultando em um modelo extremamente sensível a ruídos e que não generaliza bem.

Na prática, métodos de validação cruzada (cross-validation) são usados para auxiliar na escolha do grau mais adequado, equilibrando complexidade e capacidade de generalização

3. METODOLOGIA

A metodologia adotada neste trabalho foi planejada de modo a assegurar resultados robustos e comparáveis a pesquisas recentes na área de modelagem de irradiância solar. Todo o processo metodológico foi estruturado em etapas bem definidas, abrangendo desde a coleta e pré-processamento dos dados de cinco diferentes locais até a engenharia de atributos, análise exploratória e posterior treinamento e avaliação de modelos para estimar a irradiância global (GHI) e difusa (DIF).

3.1. Aquisição e Organização dos Dados

Esta seção detalha a aquisição e organização dos dados meteorológicos e de irradiância solar utilizados no TCC. Aborda a origem dos dados (Solargis e torres de medição da Casa dos Ventos), descreve a estrutura dos dados brutos, e explica os procedimentos de leitura e armazenamento dos dados usando a biblioteca Pandas.

3.1.1. Fonte dos Dados

Os dados utilizados neste TCC foram fornecidos pela Casa dos Ventos, sendo proveniente de torres de medições. Por uma questão de sigilo, as coordenadas dos sites serão ocultadas, e os nomes serão trocados pelos nomes das cidades mais próximas. Dessa forma, teremos os dados de Tianguá-CE, Lajes-RN, Barra-BA, Nova Alvorada do Sul-MS e Goianésia-GO. Os dados consistem em:

- Dados de satélite da empresa Solargis, calculados a partir de dados de satélite GOESR e GOES e de dados atmosféricos (ECMWF, NOAA e NASA).
- Dados medidos em site, através dos sensores SPN1 (piranômetro com padrão de qualidade “first-class”, que mede GHI e DIF) e SR20-D2 (piranômetro com padrão de qualidade “secondary standard”, que mede GHI).

Os dados de satélite do Solargis possuem tempo de amostragem de 15 minutos e período de medição que varia entre 01 de janeiro de 1999 e a data em que foram solicitados os dados (2023-2024, dependendo do site).

Já os dados dos sensores possuem uma variação conforme o período em que está cada campanha de medição de dados, ressaltando que o exigido em uma campanha de medição solar é de 12 meses, porém os dados em questão possuem entre 12-36 meses de medições, com um período de amostragem de 1 minuto.

3.1.2. Estrutura dos Dados Brutos

Os dados do Solargis são disponibilizados no formato .csv, e possuem um "header" com informações referentes a procedência dos dados, período de dados, informações do cliente, entre outros. Dentro da parte dos dados, temos os seguintes campos:

- Date: Data da medição,
- Time: Tempo da medição,
- GHI: Irradiância Global Horizontal,
- DNI: Irradiância Direta Normal,
- DIF: Irradiância Difusa Horizontal,
- flagR: Flag de identificação de qualidade de nuvem,
- SE: Ângulo de altitude solar,
- SA: Ângulo de aspecto solar,
- TEMP: Temperatura do ar,
- AP: Pressão atmosférica,
- RH: Humidade relativa,
- WS: Velocidade do vento a 10 m,
- WG: Rajada de vento a 10 m,
- WD: Direção do vento a 10 m,
- PREC: Taxa de precipitação,
- PWAT: Água precipitável

Já os dados do site da Casa dos Ventos, são exportados no formato .txt, e utilizaremos os campos de GHI de cada piranômetro SR20-D2 (duas unidades) e o GHI e DIF do SPN1.

3.1.3. Leitura e Armazenamento

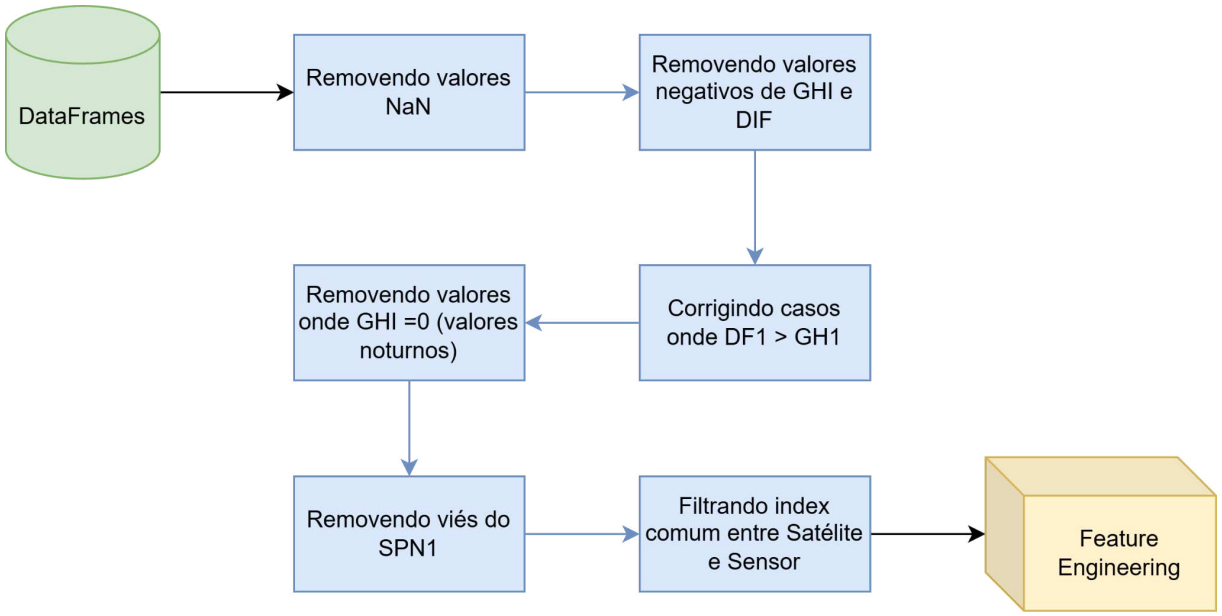
Para fazer a leitura dos dados, foi feita a remoção do cabeçalho dos arquivos originais, sobrando assim apenas o conteúdo no formato .csv (padrão utilizado para "timeseries" solares). Dessa forma, utiliza-se a função 'read_csv' da biblioteca Pandas para fazer a leitura do arquivo.

Após a leitura, salvamos os dados como um "DataFrame" da biblioteca Pandas, que nos permite trabalhar utilizando os dados com uma grande facilidade. Ressaltando que os dados foram amostrados (usando a média) para um intervalo de 30 minutos. Dessa forma, ambos os "DataFrames" possuem os mesmos intervalos de tempo de medição.

3.2. Pré-Processamento dos Dados

Esta seção detalha o pré-processamento dos dados utilizados no TCC. Aborda cada procedimento aplicado aos dados brutos. Ressaltando que esses procedimentos foram aplicados em sua maioria nos dados da Casa dos Ventos, pois os fornecidos pelo Solargis já haviam sido pré-processados pela própria empresa. Abaixo tem-se um diagrama com o fluxo dos procedimentos aplicados.

Figura 5 - Fluxograma de procedimentos do pré-processamento.



Fonte: Autoria própria

3.2.1. Remoção de Valores Ausentes (NaN)

A remoção de valores ausentes (“NaN” ou “Not a Number”) constitui uma etapa fundamental no pré-processamento, especialmente em modelagens de regressão, pois a presença de dados faltantes pode comprometer tanto o desempenho dos algoritmos quanto a interpretabilidade dos resultados. Em geral, esses valores podem surgir por diversas razões, como períodos de manutenção dos equipamentos de medição, troca de sensores ou mesmo falhas temporárias no registro dos dados. Em alguns casos, fatores externos, como problemas de comunicação ou falta de energia também podem ocasionar lacunas na aquisição das medições.

Para assegurar a consistência do conjunto de dados, optou-se por descartar linhas ou intervalos de tempo em que as variáveis de interesse apresentassem NaN de forma extensa ou para variáveis críticas na modelagem. Essa decisão se justifica pelo baixo volume relativo de dados faltantes, o que torna a remoção mais viável e menos propensa a introduzir viés, além de evitar os desafios e incertezas inerentes à imputação de valores, sobretudo quando não há uma informação sólida sobre a distribuição original das medidas. Desse modo, busca-se preservar a integridade estatística do conjunto de dados, minimizando impactos adversos no processo de treinamento dos modelos de regressão.

3.2.2. Remoção de Valores Negativos

A presença de valores negativos no conjunto de dados de irradiância solar representa um contrassenso físico, pois a irradiância não pode assumir valores abaixo de zero em condições normais de medição. Muitas vezes, essas leituras negativas surgem durante períodos noturnos, momento em que a irradiância efetivamente não está sendo captada pelo sensor, mas ainda assim podem ocorrer pequenos desvios nos instrumentos de medição. Além disso, vieses dos sensores ou problemas técnicos, como falhas de calibração e ruídos elétricos, podem levar a registros espúrios negativos mesmo durante o dia.

Para garantir a coerência dos dados e evitar que esses registros prejudiquem o processo de modelagem, optou-se pela remoção de todos os valores negativos do conjunto de dados. Essa etapa de limpeza contribui para a integridade dos modelos de regressão subsequentes, eliminando pontos que não refletem a realidade física da irradiância solar.

3.2.3. Correção de Casos “DIF > GHI”

A condição em que a irradiância difusa (DIF) supera a irradiância global (GHI) contraria os princípios físicos que fundamentam a medição solar, pois, em condições normais, a GHI consiste na soma da componente direta e da componente difusa. No presente estudo, a medição de DIF foi realizada exclusivamente por meio do sensor SPN1, o que possibilitou a identificação de inconsistências em períodos específicos, sobretudo no início e no final do dia. Esses valores anômalos podem ocorrer devido a falhas pontuais do equipamento, problemas de calibração ou ruídos na aquisição dos dados de irradiância difusa em horários de baixa irradiância.

Diante dessa situação, optou-se por corrigir os registros em que, $DIF > GHI$, fazendo $DIF = GHI$, de modo a manter a coerência física dos dados. Essa medida visa preservar a qualidade do conjunto de dados para as etapas posteriores de análise e modelagem, evitando que resultados distorcidos afetem as estimativas de desempenho dos modelos de regressão.

3.2.4. Remoção de Valores GHI = 0

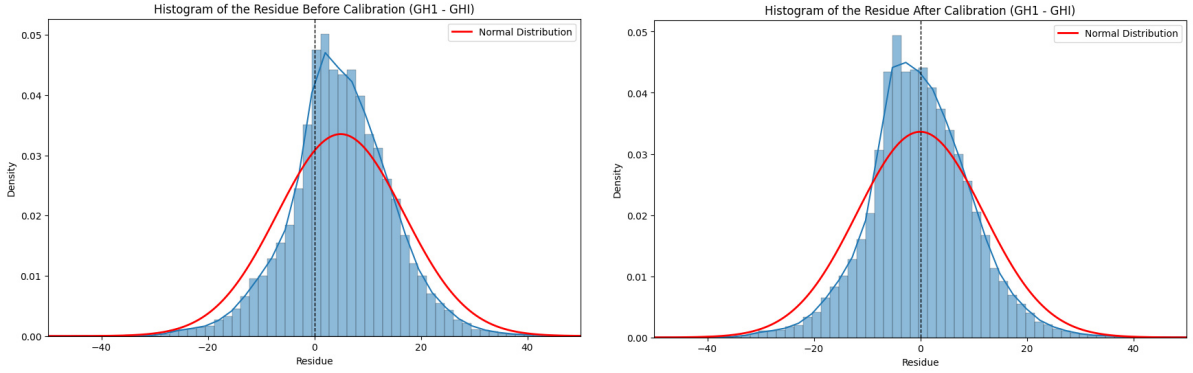
Após a etapa de remoção de valores negativos, correspondente principalmente aos registros noturnos, observou-se que parte dos dados de irradiância global (GHI) ainda apresentava valores exatamente iguais a zero. No caso dos dados provenientes do Solargis, é comum que os períodos noturnos sejam representados de forma padronizada com GHI = 0, refletindo fielmente a ausência de irradiância. Em contraste, nos sensores em campo, é raro encontrar medições de GHI exatamente nulas, pois mesmo em condições de céu noturno, podem surgir leituras muito próximas de zero, mas não exatamente zero, devido ao ruído de fundo e a pequenos desvios instrumentais.

Nesse contexto, optou-se por remover todos os registros em que GHI = 0, uma vez que eles correspondem essencialmente às condições noturnas ou a dados potencialmente não representativos para a análise de períodos diurnos. Tal procedimento simplifica o conjunto de dados e evita que linhas noturnas, sem significado para os modelos de irradiância, tenham influência de maneira indevida nas etapas de modelagem e avaliação.

3.2.5. Remoção de Viés do Sensor de Brilho Solar (SPN1)

O sensor de brilho solar SPN1 utiliza um conjunto de lentes simultaneamente para medir tanto a irradiância global (GHI) quanto a irradiância difusa (DIF). Em seu princípio de funcionamento, uma das lentes permanece totalmente exposta à incidência direta do sol, enquanto a outra permanece na sombra, permitindo a separação das componentes global e difusa. Apesar de ser classificado como um sensor de primeira classe, na mesma torre de medição são empregados sensores do tipo SR20-D2 (secondary standard), que apresentam menores margens de incerteza.

Figura 6 - Exemplo de viés no GHI do SPN1.



Fonte: Autoria própria

Para ajustar o viés do SPN1 em relação ao SR20-D2 na medição de GHI, foi estabelecida uma regressão linear simples, tendo como entrada os valores de GHI medidos pelo SPN1 e, como saída, os valores de GHI registrados pelo SR20-D2. Em seguida, a mesma relação de calibração foi aplicada aos valores de DIF obtidos pelo SPN1. Essa abordagem é justificada pelo fabricante, que assegura que as lentes responsáveis pela medição de GHI são as mesmas utilizadas na medição de DIF, de modo que a correção de viés para a componente global pode ser estendida de forma confiável para a componente difusa. Dessa maneira, garante-se maior consistência nos valores medidos pelo SPN1 antes de prosseguir para as etapas de análise e modelagem.

3.2.6. Filtragem do Índice Comum

A fim de assegurar a consistência temporal entre os dados de satélite e os valores

coletados pelos sensores em campo, procedeu-se à filtragem dos índices (“timestamps”) comuns a ambos os conjuntos. Essa etapa foi necessária para garantir que, ao comparar ou combinar as duas fontes de informação, cada medição de irradiância refere-se exatamente ao mesmo momento no tempo. Dessa forma, evitam-se discrepâncias ocasionadas por diferenças nos intervalos de coleta, harmonizando o conjunto de dados e viabilizando análises e modelagens mais precisas.

3.3. Engenharia de Atributos

Nesta etapa, foram gerados atributos adicionais para enriquecer o conjunto de dados e refinar a modelagem da irradiância. Entre eles destaca-se o índice de claridade (KT) e a fração difusa (KD), cujas expressões foram apresentadas na fundamentação teórica e baseiam-se em relações consagradas na literatura especializada. Além disso, também foram calculados ângulos solares (zenital e azimutal) e derivados (cosseno do ângulo zenital por exemplo), a fim de captar variações geométricas que influenciam a distribuição de irradiância ao longo do dia e do ano. Tais variáveis fornecem uma visão mais completa do comportamento da radiação solar e contribuem para o desenvolvimento de modelos mais robustos nas etapas seguintes.

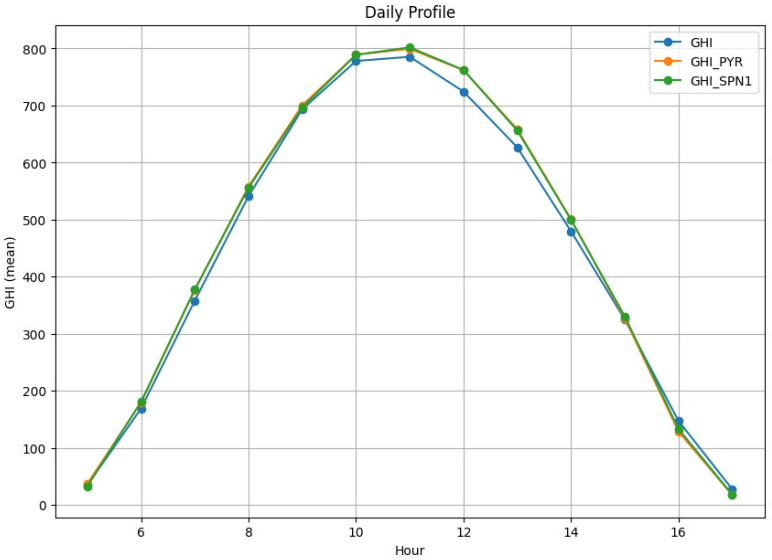
3.4. Análise Exploratória dos Dados

Nesta etapa, foram descritos os gráficos utilizados para analisar os dados. A maioria dessas representações é amplamente empregada na literatura especializada sobre irradiação solar.

3.4.1. Perfil Diário

Nesta fase, gera-se um gráfico do comportamento médio horário de GHI, o que permite avaliar se os dados de diferentes fontes (sensores e satélite) estão alinhados na mesma referência de tempo. Além disso, esse perfil diário possibilita verificar possíveis padrões recorrentes de irradiância ao longo do dia, como picos em horários específicos ou discrepâncias que possam indicar falhas nos sensores ou desalinhamento temporal.

Figura 7 - Exemplo de Perfil Diário.

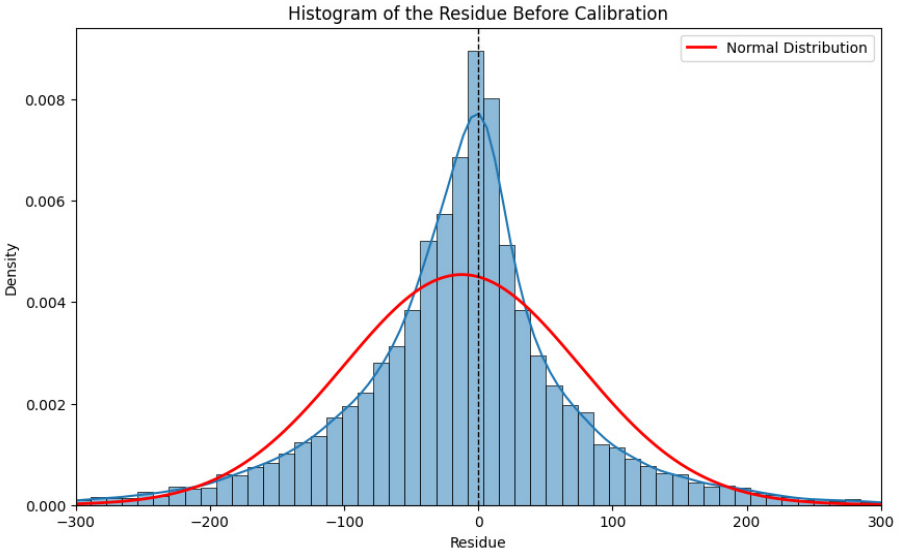


Fonte: Autoria própria

3.4.2. Análise de Viés: GHI e DIF

Em seguida, realiza-se uma análise de viés entre as medições de GHI e DIF fornecidas pelo satélite (por exemplo, Solargis) e as medições de campo. Esse passo busca identificar se há uma tendência sistemática do satélite subestimar ou superestimar a irradiância, tanto global quanto difusa. Neste caso, utilizou-se um histograma do resíduo (erro) entre o GHI do sensor e o GHI do satélite.

Figura 8 - Exemplo de Histograma do resíduo.

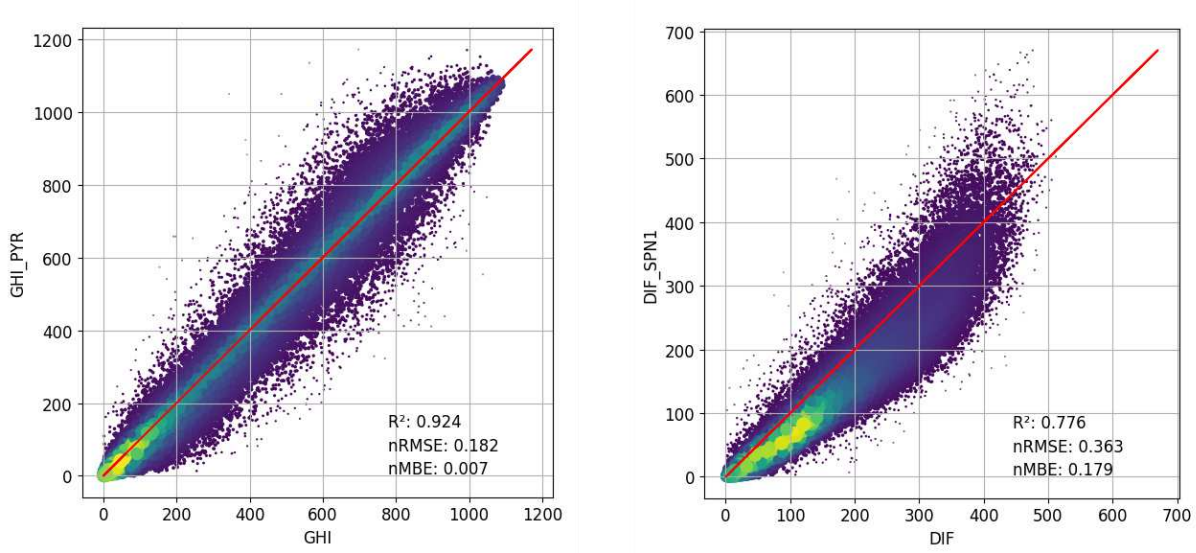


Fonte: Autoria própria

3.4.3. Comparação Satélite x Sensor

Por fim, traçam-se gráficos de dispersão colocando GHI do satélite versus GHI do sensor de campo, bem como DIF. Tais gráficos permitem examinar se a relação entre ambas as fontes de dados é fundamentalmente linear ou se apresenta desvios em determinados intervalos de irradiância. Essa comparação fornece informações valiosas sobre o grau de ajuste entre as medições satelitais e em site, contribuindo para identificar lacunas ou inconsistências antes de prosseguir para as fases de modelagem.

Figura 9 - Exemplo de comparação de GHI e DIF de satélite e sensor.



Fonte: Autoria própria

3.4.4. Análise de Heteroscedasticidade

Por fim, analisou-se a heteroscedasticidade do GHI e da DIF. A heteroscedasticidade é um fenômeno estatístico em que a variabilidade dos erros de um modelo de regressão não é constante ao longo dos valores preditores. Isso significa que a dispersão dos resíduos muda em diferentes níveis da variável independente, violando uma das principais suposições da regressão linear clássica: a homoscedasticidade (ou seja, erros com variância constante). Desta forma, para identificar esse comportamento, pode-se utilizar de gráficos do resíduo e do quadrado do resíduo, que torna a variação do erro mais notória.

3.5. Modelagem das Regressões

Nesta etapa, são descritos os procedimentos utilizados para definir os parâmetros que serão utilizadas nas regressões de GHI e DIF, assim como a explicação da abordagem utilizada nos modelos de regressão definidos para cada irradiância.

3.5.1. Definição das features

Analisou-se a correlação entre as variáveis disponíveis na série temporal do Solargis, que são as possíveis entradas do modelo de Regressão. Desta forma, definiu-se como entrada apenas a irradiação em questão (GHI ou DIF) e o cosseno do ângulo zenital (valores entre 0 e 1 são melhor interpretados pelas regressões), as outras variáveis não apresentaram correlação alta, com exceção de elementos redundantes, como, por exemplo, GHI e GHI clearsky, ou ângulo zenital e ângulo de elevação, que são variáveis que já possuem alta correlação entre si, então é redundante sua utilização.

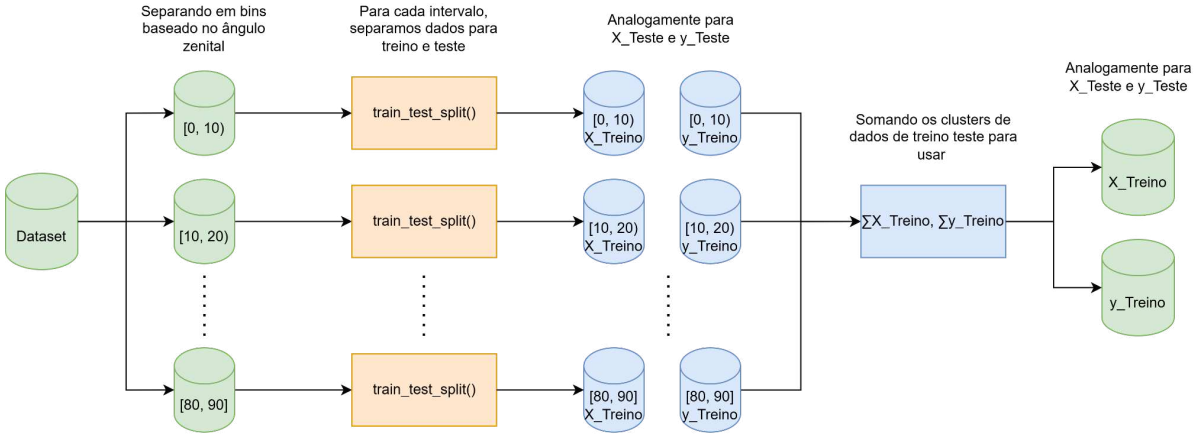
3.5.2. Separação em Conjuntos de Treino e Teste

Nesta etapa, optou-se por um método de particionamento que assegura a presença equilibrada de exemplos em cada faixa de ângulo zenital (ZEN) tanto no conjunto de treino quanto no de teste. Para isso, o conjunto de dados foi segmentado em ***bins*** de ZEN (agrupados de 10º em 10º, por exemplo, 0 a 10°, 10 a 20°, etc.), e em seguida foi aplicada a função `train_test_split` do sklearn, separadamente para cada bin, com uma proporção de 80% dos dados destinada ao treinamento e 20% ao teste. Dessa forma, evita-se que determinada faixa de ângulo zenital fique sub- ou super-representada em uma das divisões, o que poderia enviesar o treinamento e a avaliação do modelo.

A cada iteração, os dados do bin correspondente são divididos em `X_train`, `y_train` e `X_test`, `y_test`, com o embaralhamento (`shuffle=True`) controlado por uma semente fixa (`random_state=69`), para garantir reprodutibilidade do cenário. Em seguida, todos os subconjuntos de treino são armazenados em listas e, ao final do processo, concatenados para formar o conjunto final de treino. Então, o mesmo procedimento é realizado para compor o conjunto de teste. Essa estratégia permite manter a coerência das proporções de treino e teste ao longo de toda a gama de ângulos zenitais, garantindo que o

modelo seja treinado e validado de forma equilibrada em relação à variação angular.

Figura 10 - Diagrama com o procedimento de separação dos dados para treinamento de modelos.



Fonte: Autoria própria

3.5.3. Definição dos Modelos de Regressão

Para a construção e avaliação dos modelos de regressão voltados à estimação da irradiância medida (GHI, do SR20-D2 ou DIF, do SPN1), considerando as variáveis de entrada disponíveis (especialmente a irradiância estimada pelo modelo – GHI ou DIF – e o cosseno do ângulo zenital), foram definidas três abordagens metodológicas principais. A primeira consiste em utilizar todos os dados de treinamento para ajustar um único modelo de Regressão Linear, enquanto a segunda se baseia em segmentar o domínio do ângulo zenital (ZEN) em diferentes faixas e, então, calibrar um modelo de Regressão Linear específico para cada uma dessas faixas. Por fim, a terceira abordagem também segmenta o domínio de ZEN, mas emprega uma expansão polinomial nas variáveis de entrada, gerando termos adicionais (como GHI^2, $GHI \times \cos(ZEN)$, $\cos^2(ZEN)$, dentre outros) para permitir uma maior flexibilidade no ajuste.

Na primeira abordagem, denominada “Regressão Linear Única”, todo o conjunto de treinamento previamente definido (com a separação em treino e teste) é utilizado para ajustar um único modelo linear. As variáveis GHI (ou DIF) e cos(ZEN) são organizadas em um vetor X, enquanto o valor medido pelos sensores compõe o vetor alvo y. Em seguida, aplica-se a classe ***LinearRegression*** do ***scikit-learn*** para estimar os coeficientes β_0, β_1 *e* β_2. O objetivo dessa etapa é oferecer um modelo-base (baseline) simples, de modo que suas

previsões possam ser comparadas às de técnicas mais sofisticadas.

A segunda abordagem, chamada “Regressão Linear Segmentada por Faixa de Ângulo Zenital (Piecewise Linear)”, busca lidar com eventuais mudanças no comportamento da irradiância ao longo da variação de ZEN. Para isso, definem-se “bins” ou intervalos de ZEN (por exemplo, de 0° a 10°, de 10° a 20°, e assim por diante) e, para cada bin, filtra-se o conjunto de dados de treino, retendo apenas amostras cujo ângulo zenital pertença àquele intervalo. Em seguida, ajusta-se um modelo de Regressão Linear restrito aos dados desse bin, armazenando-se, para cada intervalo, os coeficientes estimados, intercepto e número de amostras utilizadas. Para prever o valor de ghi e dif em um novo instante, é necessário identificar em qual bin o respectivo ZEN cai e, então, aplicar o modelo daquele bin para gerar a estimativa. Essa abordagem permite que cada submodelo reflita a relação entre as variáveis naquele intervalo angular específico, mas também reduz a quantidade de pontos disponíveis para cada ajuste, podendo aumentar a variabilidade.

Na terceira abordagem, “Regressão Polinomial Segmentada por Faixa de Ângulo Zenital (Piecewise Polynomial)”, mantém-se a segmentação do ZEN em bins, porém, em vez de ajustar diretamente uma regressão linear simples, realiza-se uma expansão polinomial das variáveis GHI (ou DIF) e cos(ZEN). Essa expansão inclui termos como GHI^2, $\cos^2(ZEN)$ e interações como GHI×cos(ZEN). Para tanto, utiliza-se a classe ***PolynomialFeatures***, definindo um grau ***d*** (por exemplo, grau 4), e então realiza-se um ajuste linear nos termos polinomiais resultantes. Cada bin, portanto, possui um transformador polinomial próprio e um modelo de regressão correspondente, que são armazenados para posterior uso em previsão. Dessa forma, ao receber um novo valor de ZEN, identifica-se o bin e aplica-se o transformador polinomial, antes de usar o regressor linear treinado para aquela faixa específica de ZEN. Essa variação permite capturar de forma mais robusta relações não lineares, mas exige cuidado para não incorrer em sobreajuste (overfitting), especialmente quando a quantidade de dados em cada bin é limitada ou quando o grau polinomial é elevado.

3.6. Métricas de avaliação dos modelos

A avaliação dos modelos foi conduzida a partir de diferentes métricas de erro, todas normalizadas pela média dos valores observados, além da verificação do coeficiente de determinação (R^2). Em particular, empregou-se o Root Mean Square Error normalizado

(nRMSE), o Mean Bias Error normalizado (nMBE) e o Mean Absolute Error normalizado (nMAE), definidos para um conjunto de N amostras onde $\widehat{y_i}$ representa o valor predito e y_i o valor observado, e $\overline{y}$ é a média dos valores observados dada por

$$\overline{y} = (1 \div N) \times \sum_{i=1}^{N} y_i \tag{16}$$

enquanto o nRMSE é dada por

$$nRMSE = \left(\sqrt{(1 \div N) \times \sum_{i=1}^{N} (\widehat{y_i} - y_i)^2} \right) \div \overline{y} \tag{17}$$

e o nMBE, por sua vez, segue a forma

$$nMBE = \left((1 \div N) \times \sum_{i=1}^{N} (\widehat{y_i} - y_i) \right) \div \overline{y} \tag{18}$$

por fim, o nMAe, é dado por

$$nMAE = \left((1 \div N) \times \sum_{i=1}^{N} \left| \widehat{y_i} - y_i \right| \right) \div \overline{y} \tag{19}$$

Tais medidas, por serem "normalizadas" pela média $\overline{y}$, permitem uma comparação relativa dos erros em diferentes intervalos de magnitude da variável alvo. Além disso, utilizou-se o coeficiente de determinação (R^2), calculado a partir de:

$$R^2 = 1 - \left[\left(\sum_{i=1}^{N} (\widehat{y_i} - y_i)^2 \right) \div \left(\sum_{i=1}^{N} (y_i - \overline{y})^2 \right) \right] \tag{20}$$

que fornece uma indicação de quanto da variabilidade dos dados o modelo é capaz de explicar. Para facilitar a análise gráfica, foi implementada a função `***plot_performance_comparison***`, que gera, em um único painel, tanto o diagrama de

dispersão entre valores previstos e observados (com coloração baseada na densidade de pontos e uma linha de referência 1:1) quanto um histograma das diferenças ($\widehat{y} - y$). Essa função também imprime no console os valores das métricas de desempenho (R^2, nRMSE e nMBE), possibilitando uma avaliação visual e quantitativa dos resultados, bem como a identificação de eventuais tendências de superestimativa ou subestimativa ao longo da faixa de valores preditos.

4. RESULTADOS

Os resultados obtidos neste trabalho foram avaliados com base nos desempenhos dos diferentes modelos de regressão (Linear Único, Linear por Faixas de Ângulo Zenital e Polinomial por Faixas) aplicados a cinco bases de dados (Tianguá - CE, Lajes - RN, Barra - BA, Goianésia - GO e Nova Alvorada do Sul - MS). Para cada localidade, analisaram-se as métricas de desempenho, como R^2 (coeficiente de determinação), nRMSE (erro quadrático médio normalizado), nMBE (erro médio tendencioso normalizado) e nMAE (erro absoluto médio normalizado). Além disso, foram realizadas comparações entre os valores estimados e os valores medidos de GHI (Global Horizontal Irradiance) e DIF (Difusa).

Na Tabela 3, apresenta-se um panorama do viés (nMBE) das componentes GHI e DIF do Solargis em cada site, evidenciando a existência de tendência de subestimativa ou superestimativa antes da aplicação dos modelos de correção via regressão. Esses resultados servem de base para verificar o impacto das metodologias propostas na redução desse viés.

Tabela 3 - Viés das componentes GHI e DIF do Solargis para cada site.

	nMBE GHI (%)	nMBE DIF (%)
Tiangua - CE	0.6665	17.9160
Lajes - RN	-2.5530	11.8077
Barra - BA	1.9473	14.8468
Goianésia - GO	0.1178	3.5103
Nova Alvorada do Sul - MS	-0.0795	14.1650

Fonte: Autoria própria.

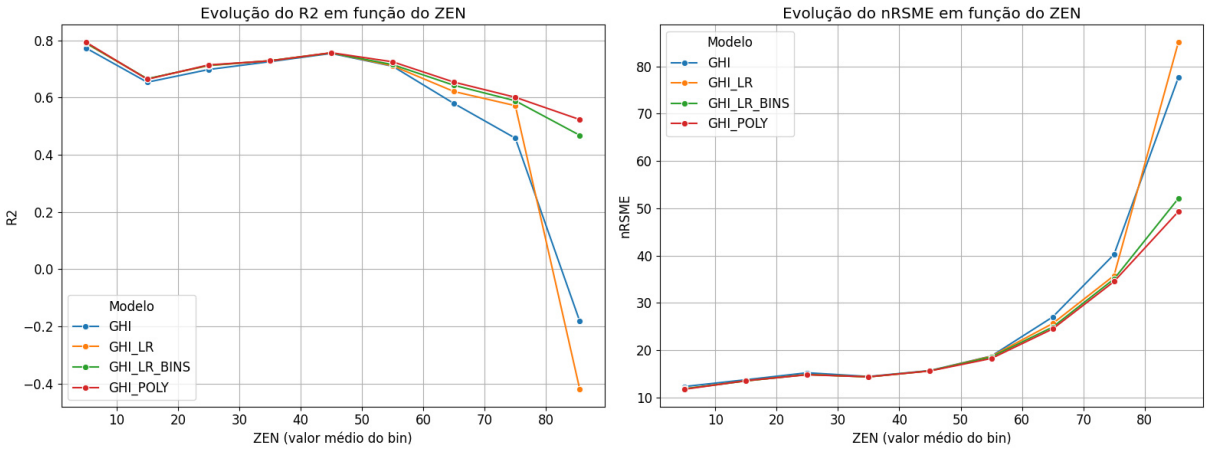
4.1. Avaliação e comparação de métricas em função do ZEN

Na sequência, cada modelo foi avaliado em função do ângulo zenital (ZEN), de modo a verificar possíveis variações no desempenho para faixas de ZEN distintas. Para ilustrar essas análises, as Figuras 11 a 20 mostram a evolução dos principais indicadores – tais como R^2, nRMSE, nMBE e nMAE – ao longo de diferentes intervalos de ZEN, para cada um dos cinco sítios estudados. Essas figuras permitem observar como a qualidade do ajuste, o viés e o erro médio variam conforme o ângulo zenital, possibilitando identificar regiões em que o modelo apresenta comportamento mais consistente ou em que possa ocorrer um aumento do erro.

4.1.1. Análise da componente global (GHI)

Em linhas gerais, observam-se melhorias significativas nas estimativas de GHI ao se aplicar modelos segmentados ou polinomiais, sobretudo em ângulos zenitais mais elevados, em que a irradiância tende a apresentar maior dispersão. Já em faixas de ZEN menores, a melhoria é por vezes menos pronunciada, uma vez que o Solargis, em alguns casos, já apresenta menor erro nessa condição. Ainda assim, a comparação detalhada dos resultados em cada sítio revela que a escolha do modelo mais apropriado pode variar em função de fatores climáticos e geográficos, tais como latitude e regime de nuvens.

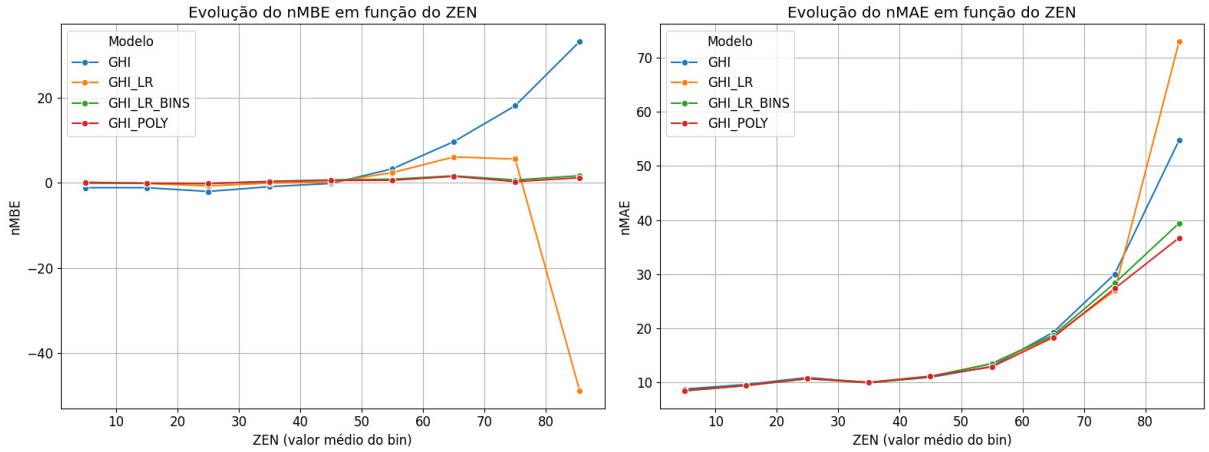
Figura 11 - Evolução do R² e do nRMSE do GHI em função do ZEN, site de Tianguá.



Fonte: Autoria própria.

Na Figura 11, pode-se observar que os modelos que foram treinados por bins de ângulo zênite, se demonstraram mais eficientes conforme os valores do ângulo zenital se aproximam de 90°. Vale ressaltar também, que para valores entre 80° à 90° de ângulo zenital o coeficiente R2 apontou valores negativos, mostrando que não existe correlação entre o calibrado e o original, fato que é explicado pelo aumento brusco do nRMSE nessa faixa de ângulo zenital. Para valores inferiores a 50° de ângulo zenital, observa-se pouca diferença entre os modelos, com os modelos treinados por bins levemente melhores.

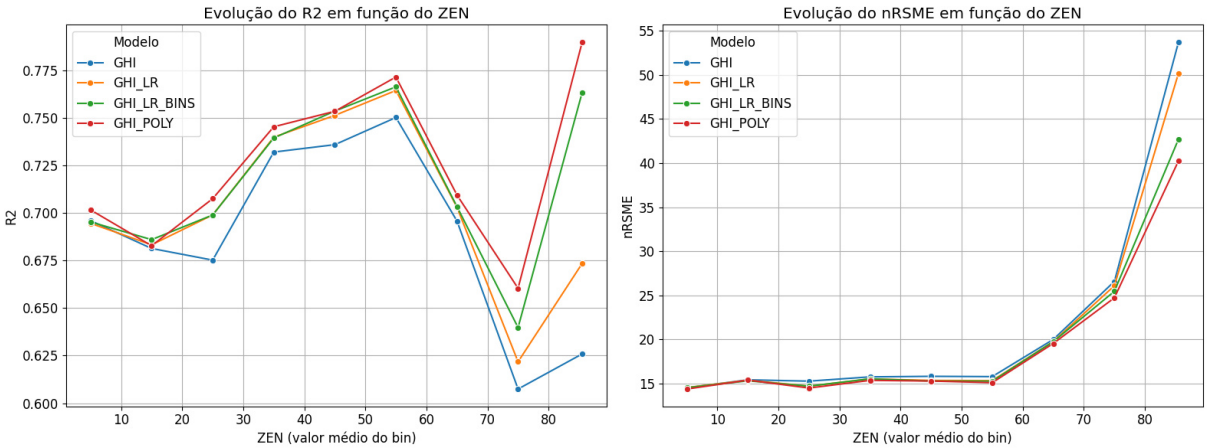
Figura 12 - Evolução do nMBE e do NMAE do GHI em função do ZEN, site de Tianguá.



Fonte: Autoria própria.

Na Figura 12, observa-se que os dados de Tianguá possuem um viés positivo considerável para valores com ângulo zenital superior a 50°, e também observa-se que a calibração por regressão linear (‘DIF_LR’) falha nesse mesmo intervalo, porém a metodologia proposta de calibração em bins, mantém o viés próximo a 0 em todo o intervalo de ângulo zenital, se demonstrando mais robusta. Esse melhor desempenho da metodologia apresentada também é observado na redução do erro absoluto para ângulos superiores a 80°.

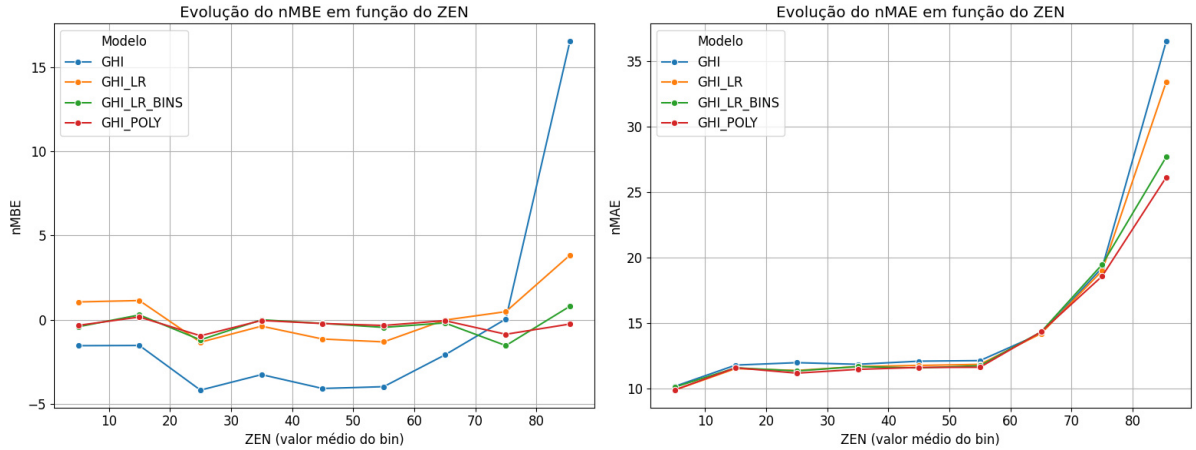
Figura 13 - Evolução do R² e do nRMSE do GHI em função do ZEN, site de Lajes.



Fonte: Autoria própria.

Na Figura 13, devido a oscilação dos valores ser mais próxima, temos uma intervalo menor de plotagem no eixo y, o que torna mais claro o desempenho superior dos modelos em bins, apesar de não ser uma diferença grande.

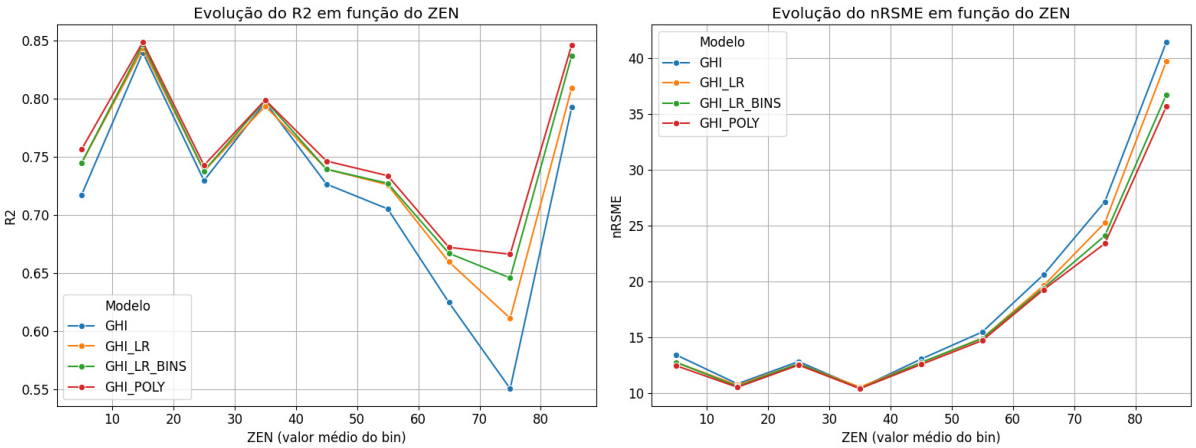
Figura 14 - Evolução do nMBE e do NMAE do GHI em função do ZEN, site de Lajes.



Fonte: Autoria própria.

Na Figura 14, observa-se um viés nos dados de Lajes, que se reflete em todos os dados de medição, com tendência negativa, também pode-se observar que os modelos de calibração em bins mostram-se mais constantes, com erros mais próximos de 0 para todos os intervalos.

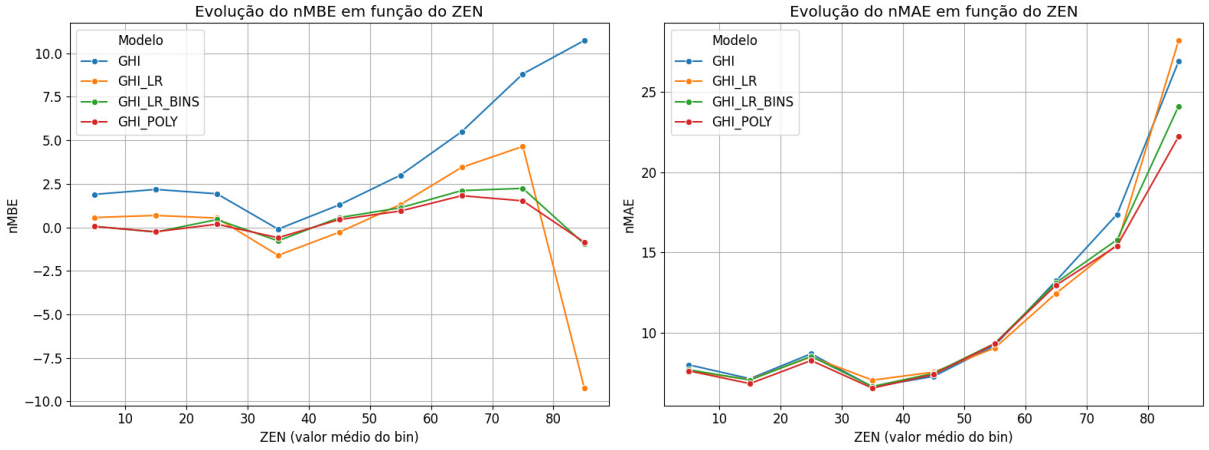
Figura 15 - Evolução do R² e do nRMSE do GHI em função do ZEN, site de Barra.



Fonte: Autoria própria.

Na Figura 15, observa-se resultados similares aos da Figura 13.

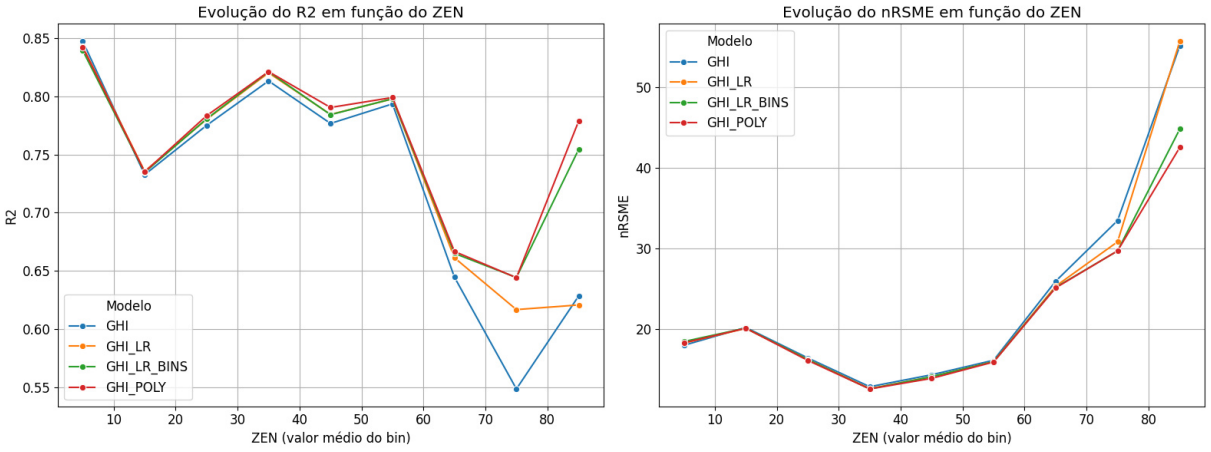
Figura 16 - Evolução do nMBE e do NMAE do GHI em função do ZEN, site de Barra.



Fonte: Autoria própria.

Na Figura 16, observa-se que para valores superiores a 50° de ângulo zenital, todos os modelos apresentaram erros consideráveis, apesar de superiores aos dados originais, considerando que para os outros sites os modelos com calibração em bins haviam atingido valores próximos de 0.

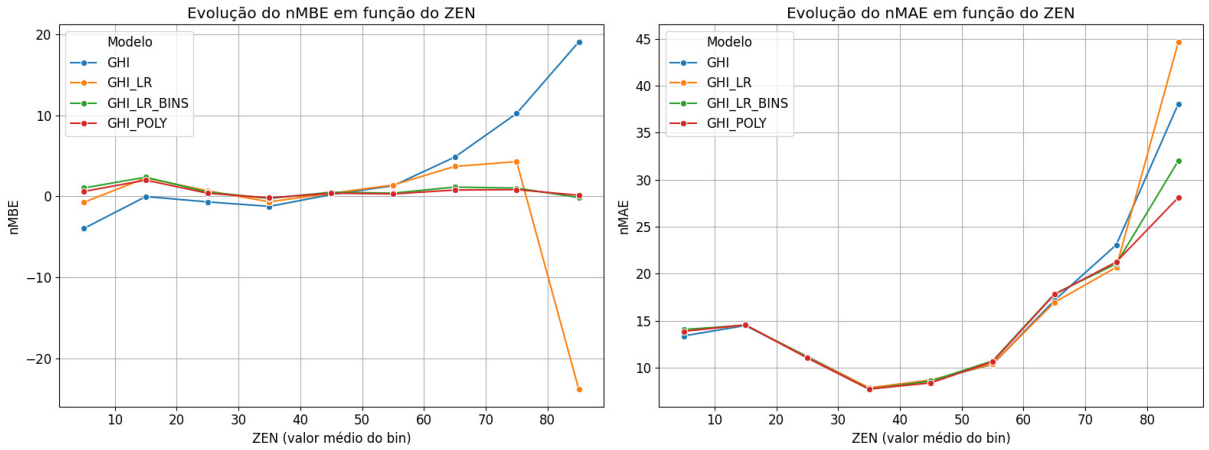
Figura 17 - Evolução do R² e do nRMSE do GHI em função do ZEN, site de Goianésia.



Fonte: Autoria própria.

Na Figura 17, observa-se valores bem próximos de R2 e nRMSE, destoando apenas para intervalos superiores a 70°de ângulo zenital, semelhante aos outros sites.

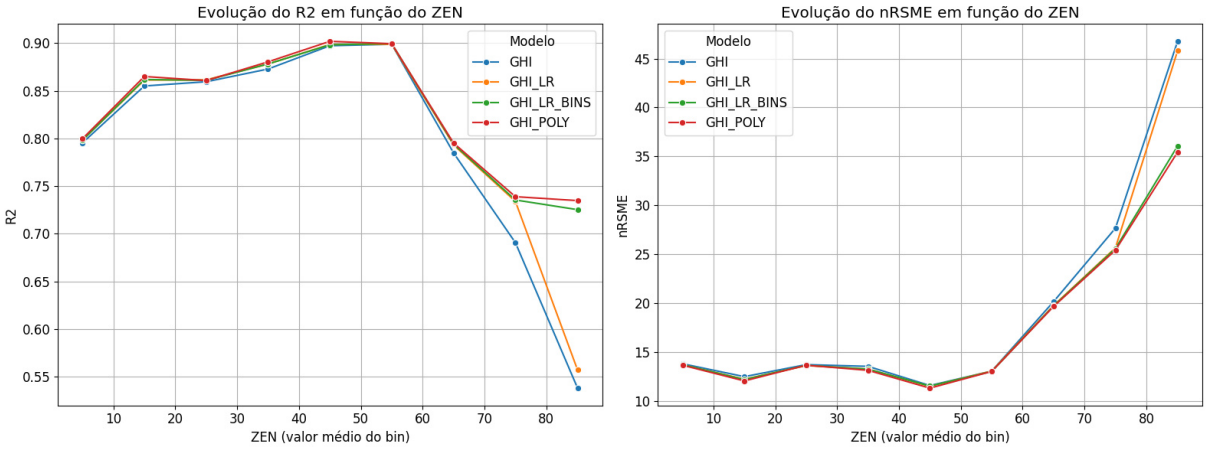
Figura 18 - Evolução do nMBE e do NMAE do GHI em função do ZEN, site de Goianésia.



Fonte: Autoria própria.

Na Figura 18, observa-se na evolução do nMBE o mesmo comportamento encontrado na Figura 17, em que para valores superiores a 90° de ângulo zenital, a calibração por regressão linear (‘DIF_LR’) apresenta um pico negativo superior ao erro dos dados sem calibração. Esse comportamento não é observado para os modelos de calibração em bins.

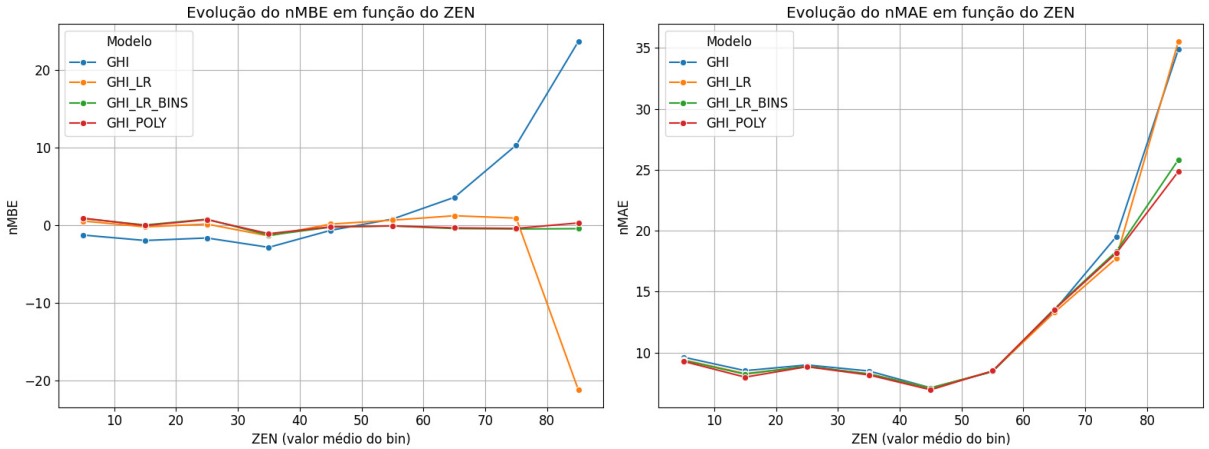
Figura 19 - Evolução do R² e do nRMSE do GHI em função do ZEN, site de Nova Alvorada do Sul.



Fonte: Autoria própria.

Na Figura 19, observa-se valores bem próximos de R2 e nRMSE, destoando apenas para intervalos superiores a 70°de ângulo zenital, semelhante aos outros sites.

Figura 20 - Evolução do nMBE e do NMAE do GHI em função do ZEN, site de Nova Alvorada do Sul.



Fonte: Autoria própria.

Na Figura 20, observa-se novamente o mesmo comportamento que ocorre na Figura 18 e 17, citado anteriormente.

Nas Figuras apresentadas, nota-se que os valores de R^2 tendem a se manter em patamares mais elevados nos intervalos de ângulo zenital (ZEN) baixos a intermediários, indicando que, nessas condições, tanto o dado original do Solargis (GHI) quanto os modelos de regressão (GHI_LR, GHI_LR_BINS e GHI_POLY) conseguem explicar uma fração considerável da variabilidade observada. Já quando o ZEN ultrapassa 60°–70°, há uma queda acentuada de R^2 em vários sítios, evidenciando maior dificuldade em prever a irradiância global à medida que o Sol se aproxima do horizonte.

As métricas de erro, especialmente o nRMSE e o nMAE, confirmam essa tendência de aumento progressivo conforme o ângulo zenital cresce, refletindo o acréscimo de incertezas e a maior dispersão dos dados de irradiância em condições de sol baixo. Mesmo nessa situação, as abordagens segmentadas (GHI_LR_BINS) e polinomiais (GHI_POLY) costumam oferecer reduções de erro um pouco mais acentuadas em comparação à regressão linear simples (GHI_LR). Em contrapartida, para ZEN baixos (até cerca de 40°–50°), os ganhos podem ser menores, pois a qualidade do dado original (GHI) já é suficientemente boa em muitos casos.

Quanto ao viés (nMBE), alguns sítios exibem valores ligeiramente positivos (superestimativa) para o dado original do Solargis nos ângulos zenitais médios ou altos, enquanto outros apresentam leve subestimativa em faixas específicas de ZEN. Os métodos de

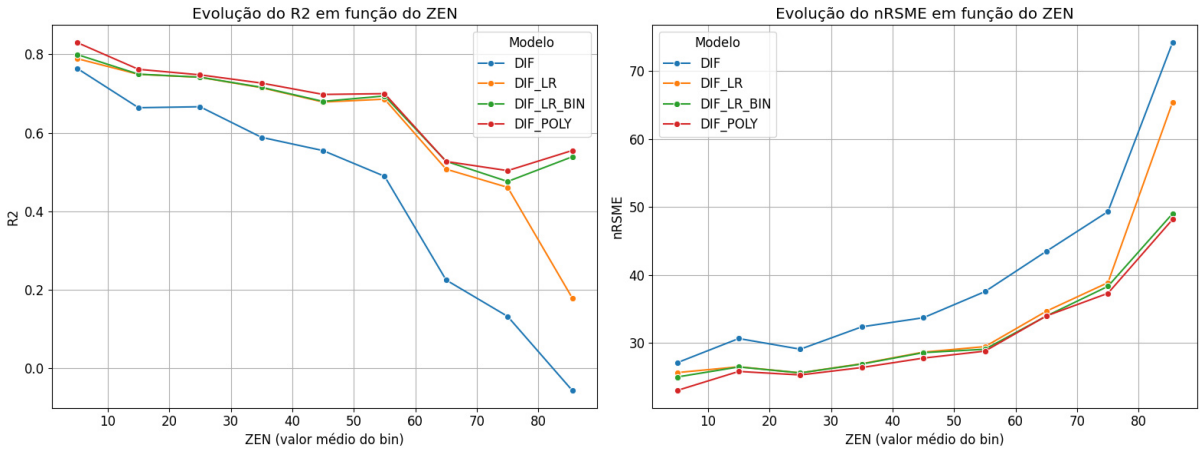
correção, em geral, reduzem essa tendência de forma razoável na maior parte dos intervalos, mas pode haver comportamentos atípicos em ângulos muito extremos, o que reforça a necessidade de análise específica para cada local e faixa angular.

De modo geral, os resultados mostram que o desempenho dos modelos varia ao longo do ângulo zenital, sendo recomendável uma abordagem que considere essa variação caso se deseje obter correções consistentes em toda a gama de posições solares. Além disso, a comparação entre GHI_LR_BINS e GHI_POLY sugere que ambas as estratégias de segmentação (por regressões lineares ou por regressões polinomiais em intervalos de zênite) são capazes de atenuar os erros e o viés do dado original, ainda que existam diferenças pontuais em cada sítio.

4.1.2. Análise da componente difusa (DIF)

Para a componente difusa (DIF), foi conduzida avaliação semelhante, observando-se novamente as métricas R^2, nRMSE, nMBE e nMAE em diferentes faixas de ZEN. De modo geral, o dado original do Solargis (DIF) apresenta viés e erro um pouco mais elevados em ângulos zenitais intermediários a altos, em razão do aumento do espalhamento atmosférico. Ainda assim, a aplicação das correções via regressão tende a reduzir esses valores, sobretudo com o uso de modelos segmentados ou polinomiais.

Figura 21 - Evolução do R^2 e do nRMSE da DIF em função do ZEN, site de Tianguá.

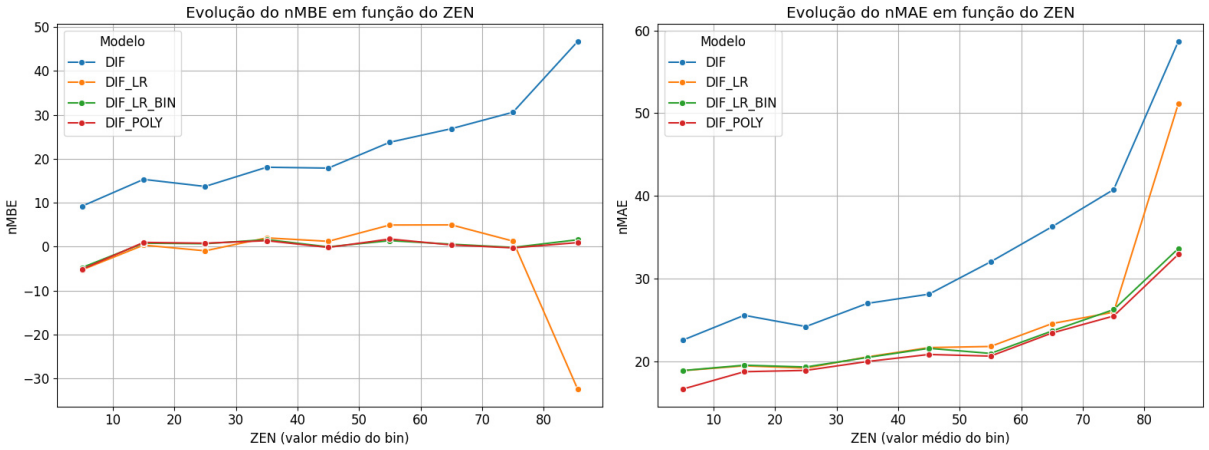


Fonte: Autoria própria.

Na Figura 21, observa-se que existe uma melhora considerável na métrica R2, para todo o intervalo de ângulo zenital, independente do método de calibração, porém os

métodos de calibração em bins, mostram-se mais efetivos.

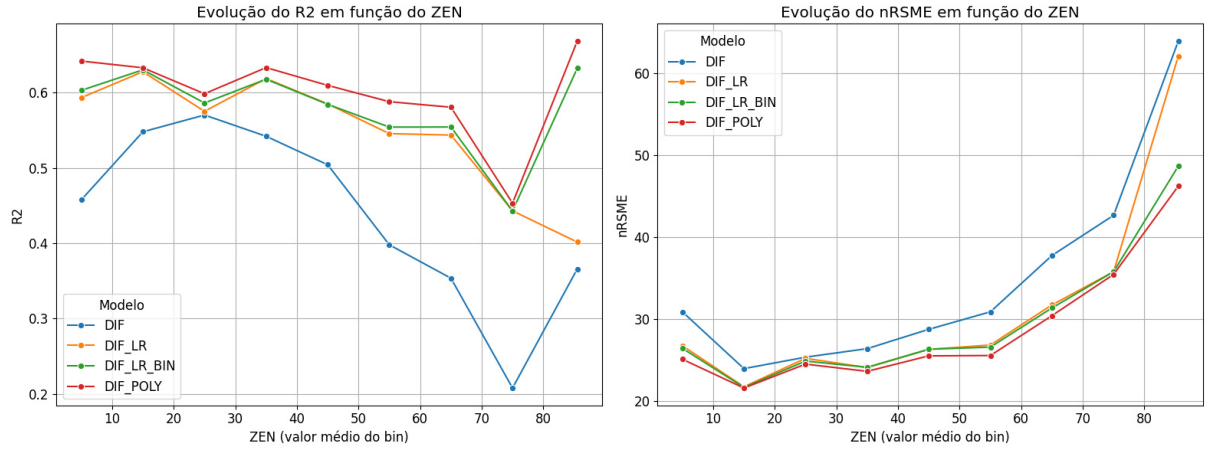
Figura 22 - Evolução do nMBE e do NMAE da DIF em função do ZEN, site de Tianguá.



Fonte: Autoria própria.

Na Figura 22, observa-se o alto viés nos dados de irradiação difusa do site de Tianguá, com valores superiores a 10% para praticamente todos os intervalos de ângulo zenital. Também observa-se que as regressões foram efetivas na remoção de viés dos dados. Vale ressaltar que observa-se o mesmo problema que ocorria na irradiação global, com a calibração por regressão linear (‘DIF_LR’), para ângulos superiores a 80°.

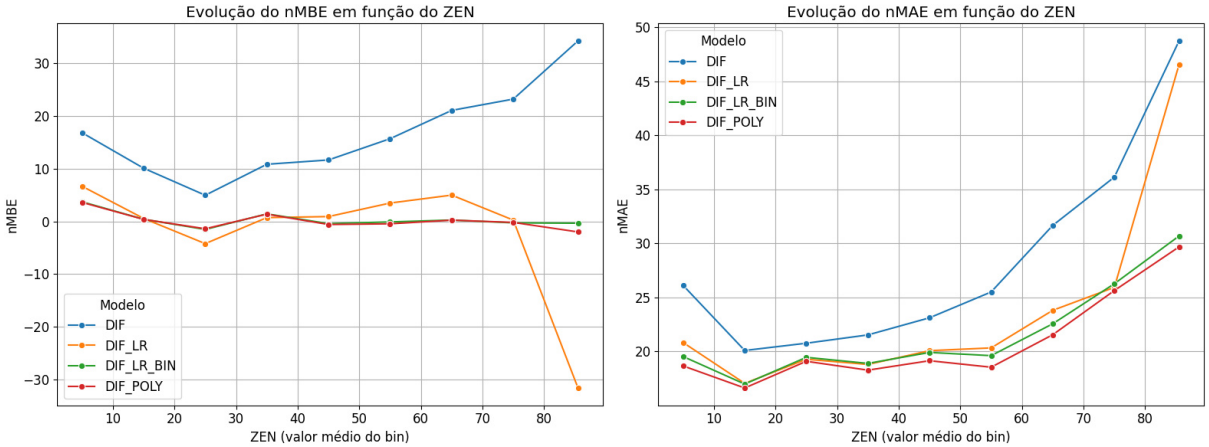
Figura 23 - Evolução do R² e do nRMSE da DIF em função do ZEN, site de Lajes.



Fonte: Autoria própria.

Na Figura 23, observa-se que existe uma melhora considerável nas métricas R2 e nRMSE, para todo o intervalo de ângulo zenital, independente do método de calibração, porém os métodos de calibração em bins, mostram-se mais efetivos, com destaque para o método polinomial, que consegue se distanciar consideravelmente em R2.

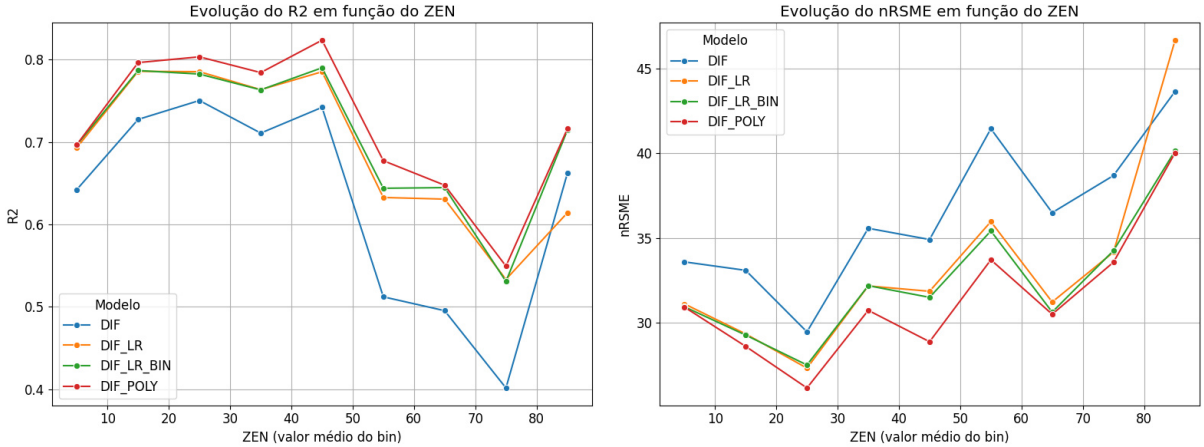
Figura 24 - Evolução do nMBE e do NMAE da DIF em função do ZEN, site de Lajes.



Fonte: Autoria própria.

Na Figura 24, observa-se o mesmo comportamento visto na Figura 22.

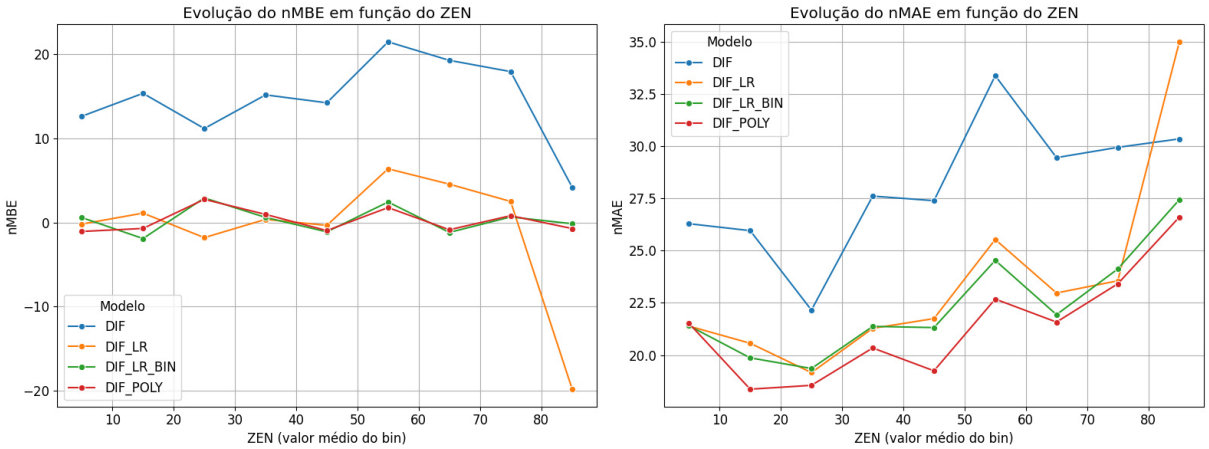
Figura 25 - Evolução do R^2 e do nRMSE da DIF em função do ZEN, site de Barra.



Fonte: Autoria própria.

Na Figura 25, observa-se de forma geral o mesmo notado nas Figuras 23 e 21, mas vale ressaltar que para valores de ângulo zenital superiores a 80°, a calibração por regressão linear (‘DIF_LR’) mostrou-se inferior ao dados sem calibração, o que havia sido notado apenas no nMBE, como citado anteriormente. Também observa-se que a regressão polinomial, para certos intervalos de ângulos zenital, mostra-se consideravelmente superior em relação às regressões lineares.

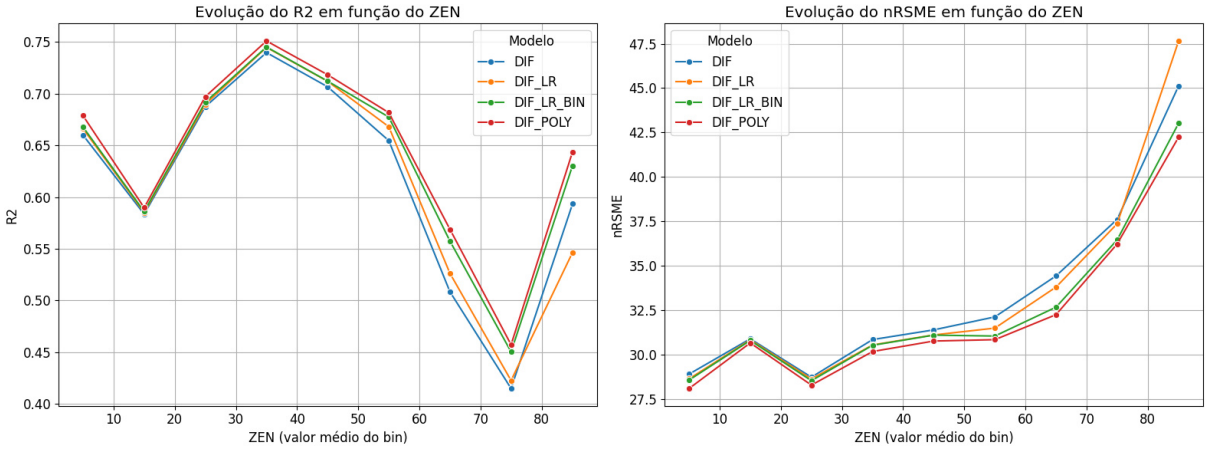
Figura 26 - Evolução do nMBE e do NMAE da DIF em função do ZEN, site de Barra.



Fonte: Autoria própria.

Na Figura 26, observa-se o mesmo comportamento visto na Figura 22 e 24, destacando uma maior redução de nMAE, como pode-se observar para a redução de aproximadamente 10% entre os dados 'DIF' e 'DIF_POLY' (calibração por bins com regressão polinomial). Ressaltando, que para este site de Barra, a regressão polinomial desempenhou uma redução em valores de erro absoluto considerável em relação a regressão linear.

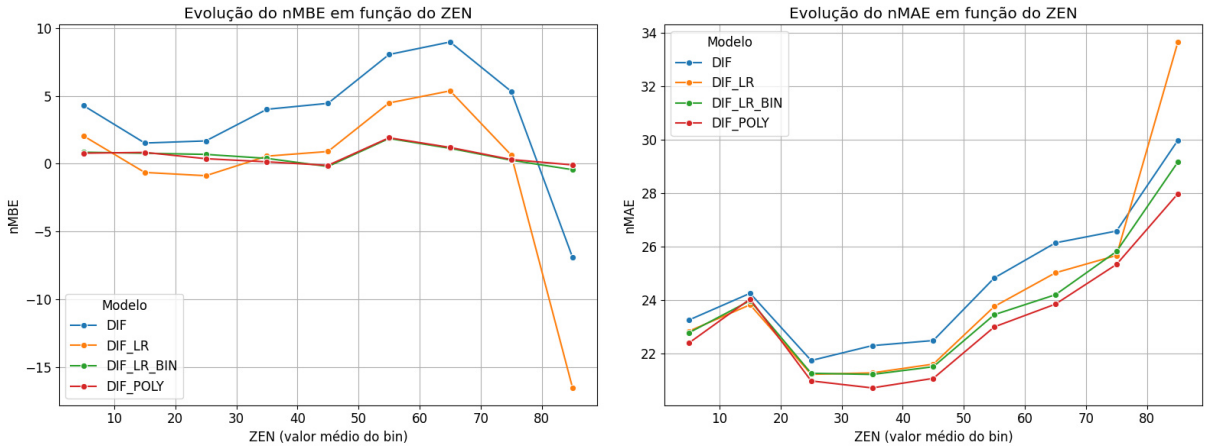
Figura 27 - Evolução do R² e do nRMSE da DIF em função do ZEN, site de Goianésia.



Fonte: Autoria própria.

Na Figura 27, observa-se métricas bem próximas e com mesma tendência, comportamento que não havia sido visto nos sites do Nordeste (Tianguá, Lajes e Barra). Ressaltando que o site de Goianésia se encontra em um local de geografia mais plana, o que influencia na irradiação difusa.

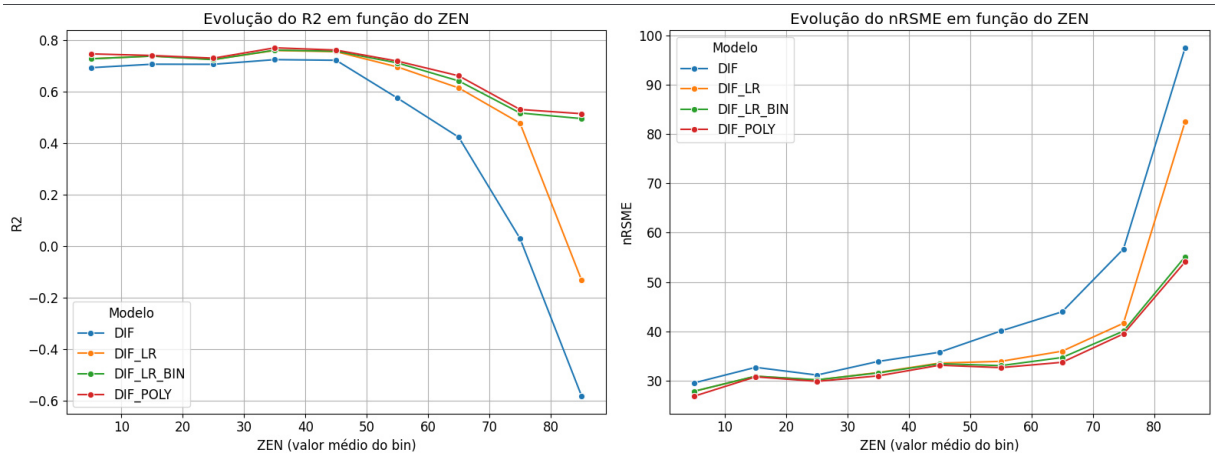
Figura 28 - Evolução do nMBE e do NMAE da DIF em função do ZEN, site de Goianésia.



Fonte: Autoria própria.

Na Figura 28, observa-se que o viés (nMBE) é inferior a 10%, muito menor que os vieses observados anteriormente. Além disso, observa-se novamente o mesmo comportamento da calibração por regressão linear (‘DIF_LR’) para valores superiores a 80° de ângulo zenital.

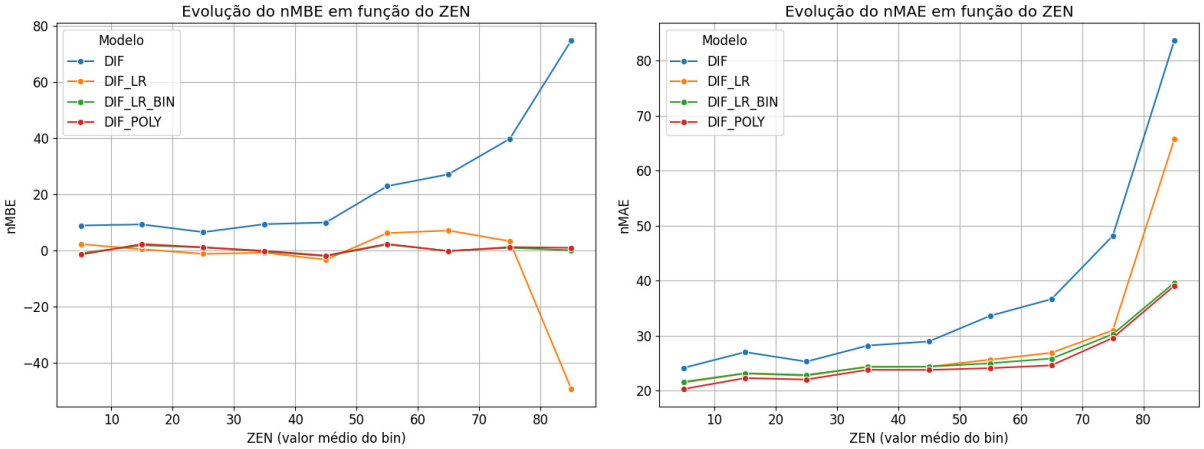
Figura 29 - Evolução do R² e do nRMSE da DIF em função do ZEN, site de Nova Alvorada do Sul.



Fonte: Autoria própria.

Na Figura 29, observa-se um comportamento similar ao encontrado nos sites do Nordeste, mas nesse caso para valores de ângulo zenital inferiores a 50°, temos métricas mais próximas.

Figura 30 - Evolução do nMBE e do NMAE da DIF em função do ZEN, site de Nova Alvorada do Sul.



Fonte: Autoria própria.

Na Figura 30, observa-se um viés superior a 10% para praticamente todos os intervalos de ângulo zenital, assim como o mesmo comportamento citado anteriormente para a calibração por regressão linear (‘DIF_LR’).

Nas Figuras apresentadas, nota-se que todas as métricas de erros destoam com maior amplitude que observado com a componente de irradiação global (GHI), com exceção de Goianésia, que foi o único site em que as métricas estão mais próximas, devido a questões geográficas, como citado anteriormente. Vale destacar, com exceção de Goianésia, que todos os sites demonstraram viés (nMBE) superiores a 10% para todo o intervalo de ZEN.

Quanto aos sites do Nordeste, observa-se diversos pontos onde o nRMSE e o nMBA mostram diferenças superiores a 5%, 10% em comparação entre os dados de satélite e os dados calibrados, o que mostra que além do viés, também é possível reduzir consideravelmente o erro absoluto, independente do intervalo de ângulo zenital.

De modo geral, os resultados reforçam que o desempenho dos modelos varia ao longo do ângulo zenital e que a qualidade dos dados de irradiação difusa da fonte de satélite é bem inferior à dos dados de irradiação global, devido às dificuldades na modelagem, geografia etc. Diante disso, a calibração de difusa mostra-se indispensável, além de que observou-se ganhos na utilização de metodologia de calibração em clusters, com destaque para a regressão polinomial, que conseguiu se destacar em relação a regressão linear, principalmente por conta da irradiação difusa possuir mais não-linearidades.

5. CONCLUSÃO

Este trabalho apresentou o desenvolvimento e a avaliação de diferentes metodologias de regressão (Linear Único, Linear por Faixas de Ângulo Zenital e Polinomial por Faixas) aplicadas à estimação da irradiância global (GHI) e difusa (DIF) em cinco localidades distintas (Tianguá - CE, Lajes - RN, Barra - BA, Goianésia - GO e Nova Alvorada do Sul - MS). Foram analisadas métricas como R^2, nRMSE, nMBE e nMAE, além de gráficos que relacionam o desempenho dos modelos às variações do ângulo zenital (ZEN). Os resultados mostram que os métodos segmentados ou polinomiais, de modo geral, suplantam o modelo linear único em cenários de maior dispersão dos dados ou de ângulos zenitais mais elevados, proporcionando reduções no viés e nos erros médios.

A partir das análises realizadas, constatou-se que:

- A segmentação em faixas de ZEN contribui para ajustar melhor os modelos às variações de irradiância, tanto global quanto difusa, principalmente em situações extremas (sol muito baixo ou muito alto no horizonte).
- A abordagem polinomial tende a capturar relações mais complexas entre a irradiância e o ângulo zenital, resultando em erros ligeiramente menores que a regressão linear segmentada em diversos intervalos.
- A irradiância difusa (DIF) apresenta naturalmente maior variabilidade em ângulos zenitais extremos, mas ainda assim pode ser beneficiada pelos modelos propostos.

Dessa forma, o trabalho contribuiu ao demonstrar que a consideração explícita do ângulo zenital e a introdução de termos polinomiais podem aperfeiçoar a acurácia de estimativas de irradiância, ampliando a confiabilidade em aplicativos que dependam de dados solares mais ajustados, como planejamento energético e dimensionamento de sistemas fotovoltaicos.

5.1. Limitações

Apesar das melhorias observadas, algumas limitações devem ser destacadas:

- Restrição de Faixas de ZEN: Em algumas localidades, o volume de dados em ângulos zenitais muito elevados (próximos de 90°) ou muito baixos (próximos de 0°) mostrou-se limitado, o que pode dificultar o refinamento dos modelos nessas condições.
- Generalização para Outros Locais: Os resultados podem não se estender de forma imediata a regiões com condições climáticas ou padrões de nebulosidade significativamente diferentes, exigindo uma etapa de revalidação ou recalibração dos modelos.
- Complexidade Computacional: A abordagem polinomial e a segmentação por bins de ZEN demandam um volume maior de ajustes e armazenamento de parâmetros, o que pode ser um entrave em aplicações com recursos computacionais mais restritos.

5.2. Trabalhos Futuros

Para dar continuidade aos resultados obtidos neste trabalho, sugere-se:

Incorporação de Variáveis Meteorológicas

- Ventos, Umidade e Temperatura: Integrar variáveis adicionais pode revelar correlações importantes e aprimorar a qualidade das estimativas de GHI e DIF.

Otimização de Bins ou Modelos de Clustering

- Abordagens Dinâmicas: Em vez de segmentar o ângulo zenital em intervalos fixos, explorar técnicas de clustering que agrupe faixas de ZEN de maneira adaptativa, considerando também o erro de previsão.

Aplicação de Métodos de Machine Learning Avançados

- Redes Neurais ou Modelos Ensembling: Investigar modelos como Random Forest, Gradient Boosted Trees ou redes neurais profundas para avaliar se conseguem capturar variações não lineares de maneira mais eficiente.

Extensão para Outras Regiões e Base de Dados

- Validação Abrangente: Expandir a análise para outros locais com perfis climáticos distintos, a fim de avaliar a robustez dos modelos em contextos mais amplos.

Avaliação Econômica e Prática

- Aplicação em Projetos Solares: Examinar o impacto da redução de erro nas estimativas de produção fotovoltaica, incluindo estudos de custo-benefício para diferentes configurações de sistema.

5.3. Considerações Finais

Com as análises e resultados obtidos, conclui-se que a inclusão do ângulo zenital de forma segmentada ou via regressão polinomial é uma estratégia promissora para aperfeiçoar as estimativas de GHI e DIF em cenários reais. Embora ainda haja espaço para aprimoramentos e validações adicionais, o conjunto de metodologias apresentadas constitui uma base sólida para aplicações que demandem maior exatidão em dados de irradiância, contribuindo para o avanço de soluções no âmbito de energias renováveis e planejamento energético em distintas regiões do país.

REFERÊNCIAS

[1] Wald, Lucien. "Basics in Solar Radiation at Earth Surface." Lecture Notes, Edição 1, Janeiro, 2021. MINES ParisTech, PSL Research University, O.I.E. – Observation, Impacts, Energy Center, Sophia Antipolis, França.

[2] Sengupta, M., Habte, A., Wilbert, S., Gueymard, C., & Remund, J. (Eds.). (2021, Abril). Best Practices Handbook for the Collection and Use of Solar Resource Data for Solar Energy Applications: Third Edition. National Renewable Energy Laboratory, German Aerospace Center (DLR), Solar Consulting Services, Meteotest AG.

[3] Atlas Brasileiro de Energia Solar, INPE (Instituto Nacional de Pesquisas Espaciais), 2017. Disponível em: http://labren.ccst.inpe.br/

[4] Mermoud, A., & Lejeune, T. (Year of publication). "Performance Assessment of a Simulation Model for PV Modules of Any Available Technology". Institute for Environmental Sciences / Energy Group, University of Geneva, Battelle, Bât. D, 7, route de Drize, CH-1227 Carouge – Switzerland.

[5] Taylor, J. R. (1997). "An Introduction to Error Analysis: The Study of Uncertainties in Physical Measurements". Second Edition. Professor of Physics, University of Colorado.

[6] METEO 300, Fundamentals of Atmospheric Science, 6.4 The Solar Spectrum. Acesso em 02 de janeiro de 2024. Fonte: https://www.e-education.psu.edu/.

[7] Notton, G. (2017). "Solar Radiation for Energy Applications". University of Corsica Pasquale Paoli, Ajaccio, França. Version of Record 14 July 2017.

[8] J. A. Duffie, W. -A. Beckman, Solar Engineering of Thermal Processes, John Wiley and Sons, N-Y. 2° Ed., 1991.

[9] Webb, N. (2014, 27 de janeiro). SPN1 Technical Fact Sheet (versão 1.2d). Delta-T Devices Ltd. Acesso em 20 de fevereiro de 2024. Fonte: https://delta-t.co.uk/wp-content/uploads/2016/10/SPN1-Technical-Fact-Sheet-v1.2_c_F.pdf

[10] Badosa, J., Wood, J., Blanc, P., Long, C. N., Vuilleumier, L., Demengel, D., and Haeffelin, M.: Solar irradiances measured using SPN1 radiometers: uncertainties and clues for development, Atmos. Meas. Tech., 7, 4267–4283, https://doi.org/10.5194/amt-7-4267-2014, 2014.

[11] Hukseflux. (2024). USER MANUAL SR20-D2: Digital secondary standard pyranometer with Modbus RTU and 4-20 mA output (Versão 2301). Acesso em 20 de fevereiro de 2024. Fonte: https://www.hukseflux.com/uploads/product-documents/SR20-D2_manual_v2301.pdf.

[12] International Organization for Standardization. (2018). ISO 9060:2018: Solar energy –

Specification and classification of instruments for measuring hemispherical solar radiation. Acesso em 20 de fevereiro de 2024. Fonte: https://www.iso.org/standard/67464.html

[13] INEICHEN, Pierre, et al. Dynamic global-to-direct irradiance conversion models. ASHRAE Transactions, 1992, vol. 98, no. 1, p. 354-369.

[14] Cebecauer, T., & Suri, M. (2024). Site-adaptation of satellite-based DNI and GHI time series: overview and SolarGIS approach. Solar Energy, AIP Conf. Proc. 1734, 150002 (2016). Acesso em 20 de fevereiro de 2024. Fonte: https://doi.org/10.1063/1.4949234.

[15] Copernicus Atmosphere Monitoring Service. (2022). User Guide to the CAMS Radiation Service (CRS) Status December 2022 (Ref: CAMS2_73_2021SC1_D3.2.1_2022_UserGuide_v1). Issued by DLR / M. Schroedter-Homscheidt. Retrieved from https://atmosphere.copernicus.eu/sites/default/files/2022-01/CAMS2_73_2021SC1_D3.2.1_2021_UserGuide_v1.pdf.

APÊNDICE - JUPYTER NOTEBOOK DO CÓDIGO DO TRABALHO

```
# %% [markdown]
# ## Timeseries from XX - XX
# _Data provided by Casa dos Ventos Desenvolvimento._

# %% [markdown]
# ## 1. Problem Definition
#
# Satellite solarimetric data are very general and have several years of measurements (more than 20), but when compared with sensor data, they have considerable inaccuracy. That said, the proposal is to use data from measurement campaigns (3 years of sensor measurements) to calibrate the entire satellite time series, to improve data quality.

# %% [markdown]
# ## 2. Data Mining
#
# Satellite data from Solargis or other Satellite source.

# %% [markdown]
# ### 2.1. Imports
#
# Imports and functions declarations.

# %%
# Imports
# Libraries
import numpy as np
import pandas as pd
import seaborn as sns
import scipy.stats as stats
import statsmodels.api as sm
import matplotlib.pyplot as plt

# Classes
from typing import Union
from pvlib import (
    atmosphere,
    clearsky,
    irradiance,
    solarposition,
)
```

```
from datetime import timedelta
from scipy.stats import gaussian_kde
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures

# Functions
from scipy.stats import mode
from tabulate import tabulate
from pvlib.tools import sind, cosd
from sklearn.metrics import r2_score
from statsmodels.stats.diagnostic import het_breuschpagan, het_white

# %%
def add_ghi_column(data: pd.DataFrame) -> pd.DataFrame:

    data_copy = data.copy()
    data_copy['GHI'] = (data_copy['PR1'] + data_copy['PR2']) / 2

    return data_copy

def add_kd_column(sat_data: pd.DataFrame,
                  ghi_column_name: str,
                  dif_column_name: str) -> pd.DataFrame:
      # Create a copy of the input DataFrame to avoid modifying the
original data
    sat_data_copy = sat_data.copy()

    # Calculate the Diffuse-to-Global Horizontal Irradiance Ratio (KD)
            sat_data_copy['KD']    =    sat_data_copy[dif_column_name]    /
sat_data_copy[ghi_column_name]

     # Replace NaN values in 'KD' column with 1 to handle division by
zero
    sat_data_copy.loc[pd.isna(sat_data_copy['KD']), 'KD'] = 1
    sat_data_copy.loc[np.isinf(sat_data_copy['KD']), 'KD'] = 1

    return sat_data_copy

def add_kt_column(data: pd.DataFrame,
                  ghi_column_name: str,
                  toa_column_name: str,
                  elv_column_name: str,
) -> pd.DataFrame:
```

```
    # Create a copy of the input DataFrame to avoid modifying the
original data
    data_copy = data.copy()

    # Calculate the Transmittance Ratio (KT)
                data_copy['KT']    =    data_copy[ghi_column_name]    /
(data_copy[toa_column_name] * sind(data_copy[elv_column_name]))

    # Replace NaN values in 'KT' column with 0
    data_copy.loc[pd.isna(data_copy['KT']), 'KT'] = 0

    # Replace infinite KT values with 0 to avoid incorrect Direct
Normal Irradiance (DNI) values
     data_copy.loc[(data_copy['KT'] == np.inf) | (data_copy['KT'] ==
np.NINF), 'KT'] = 0

    return data_copy

def add_solar_position_columns(data: pd.DataFrame,
                               dt_minutes: int,
                               latitude: float,
                               longitude: float,
                                   correct_sun_elevation: bool =True) ->
pd.DataFrame:

    # Create a copy of the input DataFrame to avoid modifying the
original data
    data_copy = data.copy()

    timestamp = data_copy.index + timedelta(minutes=dt_minutes / 2)
    # Get solar position by NREL SPA algorithm from timestamp
       sun_position_pvlib  =  solarposition.get_solarposition(timestamp,
latitude, longitude)
    # Get the extraterrestrial radiation from timestamp
    irr_toa = irradiance.get_extra_radiation(timestamp)

            sun_position_pvlib.index    =    sun_position_pvlib.index    -
timedelta(minutes=dt_minutes / 2)

    irr_toa.index = irr_toa.index - timedelta(minutes=dt_minutes / 2)

    sun_elevation_pvlib = sun_position_pvlib['apparent_elevation']
```

```
    if correct_sun_elevation:
                                        sun_elevation_pvlib_corrected     =
apply_elevation_correction(sun_elevation_pvlib)
                               sun_position_pvlib['apparent_elevation']    =
sun_elevation_pvlib_corrected
                          sun_position_pvlib['apparent_zenith']    =    90    -
sun_elevation_pvlib_corrected
    # Create new columns in the dataframe
    data_copy['ZEN'] = sun_position_pvlib['apparent_zenith']
    data_copy['ELV'] = sun_position_pvlib['apparent_elevation']
    data_copy['AZ'] = sun_position_pvlib['azimuth']
    data_copy['TOA'] = irr_toa

    return data_copy

def       apply_elevation_correction(elevation_angle:       Union[float,
np.array]):

    # Convert scalar input to numpy array
    elevation_angle = np.asarray(elevation_angle)

    # Apply elevation correction
    result = np.where(
        elevation_angle >= 7.0,
        elevation_angle,
        np.where(
            (-7.0 < elevation_angle) & (elevation_angle < 7.0),
            0.5 * elevation_angle + 3.5,
            0.0
        )
    )

    # If input was scalar, return scalar; otherwise, return numpy array
    return np.squeeze(result)

def get_windographer_avg_columns(data: pd.DataFrame) -> pd.DataFrame:
    # List all column names from the input DataFrame
    columns_list = list(data.columns)

    # Extract the columns that contain 'AVG'
    columns_avg = [col for col in columns_list if 'AVG' in col]
```

```
    # Select the 'AVG' columns from the DataFrame
    data_avg = data.loc[:, columns_avg]

    # List all column names of the filtered DataFrame
    columns_list_avg = list(data_avg.columns)

    # Shorten each column name to its first three characters
    column_shortnames_avg = [x[:3] for x in columns_list_avg]

    # Rename the columns in the filtered DataFrame
    data_avg.columns = column_shortnames_avg

    return data_avg

def get_common_index(dataframes):
    # Ensure there are at least two DataFrames in the array
    if len(dataframes) < 2:
        raise ValueError("At least two DataFrames are required.")

    # Extract indices from the first DataFrame
    common_idx = set(dataframes[0].index)

    # Intersect indices with each subsequent DataFrame
    for df in dataframes[1:]:
        common_idx.intersection_update(df.index)

    return common_idx

def get_clear_sky_values(timestamp: pd.Timestamp,
                         latitude: float,
                         longitude: float,
                         altitude: float) -> pd.DataFrame:
    # Get solar position using pvlib
       sun_position_pvlib  =  solarposition.get_solarposition(timestamp,
latitude, longitude)

    # Get apparent elevation and apply correction
    sun_elevation_pvlib = apply_elevation_correction(
        elevation_angle=sun_position_pvlib['apparent_elevation']
    )

    sun_position_pvlib['apparent_elevation'] = sun_elevation_pvlib
    sun_position_pvlib['apparent_zenith'] = 90 - sun_elevation_pvlib
```

```
    # Get relative and absolute airmass
    rel_airmass = atmosphere.get_relative_airmass(
        zenith=sun_position_pvlib['apparent_zenith']
    )
    abs_airmass = atmosphere.get_absolute_airmass(
        airmass_relative=rel_airmass,
        pressure=atmosphere.alt2pres(altitude)
    )

    # Linke turbidity
     linke_turb = 2#clearsky.lookup_linke_turbidity(timestamp, latitude,
longitude)

    # Calculate clear-sky values
    clear_sky_day = clearsky.ineichen(
        apparent_zenith=sun_position_pvlib['apparent_zenith'],
        airmass_absolute=abs_airmass,
        linke_turbidity=linke_turb,
        altitude=altitude
    )

    clear_sky_day.rename(columns={
        'ghi': 'GHI',
        'dhi': 'DIF',
        'dni': 'DNI'
    }, inplace=True)

    # Update 'ZEN' in clear_sky_day with apparent zenith angle
    clear_sky_day['ZEN'] = sun_position_pvlib['apparent_zenith']

    # Get extraterrestrial radiation at the top of the atmosphere
    clear_sky_day['TOA'] = irradiance.get_extra_radiation(timestamp)

    # Get apparent elevation
    clear_sky_day['ELV'] = sun_position_pvlib['apparent_elevation']

      # Calculate clear-sky index (kt) based on GHI, extraterrestrial
radiation, and apparent elevation
    clear_sky_day = add_kt_column(clear_sky_day, "GHI", "TOA", "ELV")

    # Return the calculated clear-sky values
    return clear_sky_day
```

```
def get_minutes_resolution_from_time(time_list: list, format: str=None)
-> int:

    if isinstance(time_list, pd.DatetimeIndex):
        time_series = time_list
    else:
        # Convert the time series string to datetime
        time_series = pd.to_datetime(time_list, format=format)

       # If time have't at least 2 elements, we can descover the
resolution
    # So it raises an ValueError
    if len(time_series)<2:
        raise ValueError("Not enough elements in list.")

    # Calculate the time difference between consecutive timestamps
    time_diff = time_series.diff()[1:]

    # Converting Timedelta to seconds
    time_diff = [diff.total_seconds() for diff in time_diff]

    # Calculate the mode of time differences in seconds
    mode_time_diff_sec = mode(time_diff)[0]

    # Convert mode time difference to minutes
    resolution_minutes = mode_time_diff_sec // 60

    return resolution_minutes

def predict_piecewise_ghi(
    zen_value: list,
    ghi_value: list,
    bin_models: list
) -> list:

    cos_zen = cosd(zen_value)  # Convert to cosine

    for model in bin_models:
        (low, high) = model["zen_range"]
        if low <= zen_value < high:
            # Use the corresponding bin's regression model
            ghi_slope = model["slope_GHI"]
```

```
            czen_slope = model["slope_cZEN"]
            intercept = model["intercept"]

                return intercept + ghi_slope * ghi_value + czen_slope *
cos_zen

    return np.nan

def predict_piecewise_dif(
    zen_value: list,
    dif_value: list,
    bin_models: list
) -> list:

    cos_zen = cosd(zen_value)  # Convert to cosine

    for model in bin_models:
        (low, high) = model["zen_range"]
        if low <= zen_value < high:
            # Use the corresponding bin's regression model
            dif_slope = model["slope_DIF"]
            czen_slope = model["slope_cZEN"]
            intercept = model["intercept"]

                return intercept + dif_slope * dif_value + czen_slope *
cos_zen

    return np.nan

def fit_piecewise_polynomial(
    df: pd.DataFrame,
    zen_bins: list,
    degree: int,
    feature_cols: list = ["GHI", "cZEN"],
    target_col: str = "GHI_PYR"
):

    poly_models = []

    for (low, high) in zen_bins:
        # Filter the data for this ZEN bin
        mask = (df["ZEN"] >= low) & (df["ZEN"] < high)
        df_bin = df[mask]
```

```
        # Prepare X and y
        X = df_bin[feature_cols].values  # e.g., [DIF, cZEN]
        y = df_bin[target_col].values

        # Polynomial expansion for this bin
                poly_transformer  =  PolynomialFeatures(degree=degree,
include_bias=True)
        X_poly = poly_transformer.fit_transform(X)

           # Fit a simple linear regression on the polynomial-expanded
features
        reg = LinearRegression().fit(X_poly, y)

        # Compute MSE (just for reference)
        y_pred = reg.predict(X_poly)
        nrmse = normalized_rmse(y_pred, y)

        poly_models.append({
            "zen_range": (low, high),
            "model": reg,
            "poly_transformer": poly_transformer,
            "nrmse": nrmse,
            "n_samples": len(X)
        })

                    print(f"[Degree={degree}]  ZEN  [{low},  {high}):
nRMSE={nrmse:.3f}, n={len(X)}")

    return poly_models

def predict_piecewise_polynomial(
    zen_value: float,
    param_value: float,
    bin_models: list
) -> float:

    # Compute cos(zen) in degrees
    cos_zen = cosd(zen_value)

    # Check each bin; use whichever bin the current zen_value fits into
    for model_info in bin_models:
        (low, high) = model_info["zen_range"]
```

```
        if low <= zen_value < high:
            # Grab the polynomial transformer and the regression model
            poly_transformer = model_info["poly_transformer"]
            reg = model_info["model"]

            # Construct our 2D feature row: [ [DIF, cZEN] ]
            X = np.array([[param_value, cos_zen]])

               # Transform using the stored polynomial transformer for
that bin
            X_poly = poly_transformer.transform(X)

            # Predict
            y_pred = reg.predict(X_poly)
            return y_pred[0]

    # If ZEN does not fall into any bin, return NaN
    return np.nan

def normalized_rmse(predicted, observed):
    rmse = np.sqrt(np.mean((predicted - observed) ** 2))
    return rmse / np.mean(observed)

def normalized_mbe(predicted, observed):
    mbe = np.mean(predicted - observed)
    return mbe / np.mean(observed)

def normalized_mae(predicted, observed):
    mba = np.mean(np.abs(predicted - observed))
    return mba / np.mean(observed)

def plot_performance_comparison(
    x: np.ndarray,
    y: np.ndarray,
    filter_array: np.ndarray = None,
    filter_threshold: float = None,
    x_label: str = 'Predicted',
    y_label: str = 'Actual',
    diff_label: str = 'Difference',
    diff_count_label: str = 'Count',
    diff_title: str = 'Histogram of Differences'
) -> None:
    # Determine which points to include based on optional filtering
```

```
    if filter_array is not None and filter_threshold is not None:
        valid_filter = filter_array > filter_threshold
    else:
        valid_filter = np.ones_like(x, dtype=bool)

    # Setup the figure and axes
    plt.rcParams["figure.figsize"] = (12, 6)
    plt.rc('font', size=12)

    fig, axes = plt.subplots(1, 2)
    axes = axes.flatten()

    # Calculate point density for scatter plot
    xy = np.vstack([x[valid_filter], y[valid_filter]])
    z = gaussian_kde(xy)(xy)
    size_marker_factor = 100 / z.max()

    # Scatter plot with density-based coloring
        axes[0].scatter(x[valid_filter], y[valid_filter], c=z, s=z *
size_marker_factor)
     axes[0].plot(y[valid_filter], y[valid_filter], color='red')  # 1:1
reference line
    axes[0].set_xlabel(x_label)
    axes[0].set_ylabel(y_label)

    # Compute and annotate metrics
    r2_value = r2_score(y[valid_filter], x[valid_filter])
    nrmse_value = normalized_rmse(x[valid_filter], y[valid_filter])
    nmbe_value = normalized_mbe(x[valid_filter], y[valid_filter])

     axes[0].annotate('R²: ' + format(r2_value, '.3f'), xy=(0.65, 0.15),
xycoords='axes fraction')
     axes[0].annotate('nRMSE: ' + format(nrmse_value, '.3f'), xy=(0.65,
0.10), xycoords='axes fraction')
      axes[0].annotate('nMBE: ' + format(nmbe_value, '.3f'), xy=(0.65,
0.05), xycoords='axes fraction')
    axes[0].grid(True)

    # Differences plot (x - y)
    diff = x - y

    # Histogram of differences
       axes[1].hist(diff[valid_filter], range=(diff[valid_filter].min(),
```

```
diff[valid_filter].max()), bins=50)
    axes[1].set_xlabel(diff_label)
    axes[1].set_ylabel(diff_count_label)
    axes[1].set_title(diff_title)
    axes[1].grid(True)

    plt.tight_layout()
    plt.show()

    # Print metrics
    print('R²: ' + str(r2_value))
    print('nRMSE [%]: ' + str(round(nrmse_value, 2)))
    print('nMBE [%]: ' + str(round(nmbe_value, 2)))

# %% [markdown]
# ### 2.2. Reading Data
#
# Reading CSVs with Satellite and Sensor data.

# %%
# Path to each file
sensor_data_path   = '../data/xxxx.csv'
solargis_data_path  = '../data/xxxx.csv'

# %%
# This data can be get from Solargis header
# Getting Latitude, Longitude, Elevation and Calm Treshold
latitude, longitude, altitude, calm_treshold =  'xxxx', 'xxxx', 'xxxx',
'xxxx'

# %%
# Reading sensor time series
sensor_ts = pd.read_csv(sensor_data_path)

# Getting average columns
sensor_ts_avg = get_windographer_avg_columns(sensor_ts)

# Get only the desired columns
# DF1 and GH1 = Diffuse and Global from SPN1
# PR1 and PR2 = Global from SR20-D2 (two sensors)
sensor_ts_avg = sensor_ts_avg[['DF1', 'GH1', 'PR1', 'PR2', 'TP1']]

# %%
```

```
# The sensor data is in 1 min frequency and the satellite data is in 15
min frequancy.
# So we will resample the sensor data to 30min
sensor_ts_avg = sensor_ts_avg.resample('30min').mean()

# %%
# Read satellite and sensor data
solargis_ts = pd.read_csv(solargis_data_path)
solargis_ts = solargis_ts.resample('30min').mean()

# %% [markdown]
# ## 3. Preparing Data

# %% [markdown]
# ### 3.1. Removing NaN

# %%
# I have sensor data of 2 identicals sensors, to avoid bias.
# So I'll make the mean of them and create the GHI column.
sensor_ts_avg = add_ghi_column(sensor_ts_avg)

# %%
# Then drop the pyranometer 1 and 2 columns
sensor_ts_avg.drop(columns=['PR1', 'PR2'], inplace=True)

# Drop NaN from sensor data
sensor_ts_avg.dropna(inplace=True)

# Drop NaN from satallite data
solargis_ts.dropna(inplace=True)

# Drop time column
solargis_ts.drop(['time'], inplace=True, axis=1)

# %% [markdown]
# ### 3.2. Removing Negative Values

# %%
# DF1, GH1 and GHI can not be negative values.
# So w'll get those index to fix it.
df1_negative_idx = sensor_ts_avg[
    sensor_ts_avg["DF1"] < 0
].index
```

```
gh1_negative_idx = sensor_ts_avg[
    sensor_ts_avg["GH1"] < 0
].index

ghi_negative_idx = sensor_ts_avg[
    sensor_ts_avg["GHI"] < 0
].index

# %%
# Replacing those values for 0.
# DF1
sensor_ts_avg.loc[
    sensor_ts_avg.index.isin(df1_negative_idx), "DF1"
] = sensor_ts_avg.loc[
    sensor_ts_avg.index.isin(df1_negative_idx), "DF1"
].clip(lower=0)

# GH1
sensor_ts_avg.loc[
    sensor_ts_avg.index.isin(gh1_negative_idx), "GH1"
] = sensor_ts_avg.loc[
    sensor_ts_avg.index.isin(gh1_negative_idx), "GH1"
].clip(lower=0)

# GHI
sensor_ts_avg.loc[
    sensor_ts_avg.index.isin(ghi_negative_idx), "GHI"
] = sensor_ts_avg.loc[
    sensor_ts_avg.index.isin(ghi_negative_idx), "GHI"
].clip(lower=0)

# %% [markdown]
# ### 3.3. Fixing cases when DF1 > GH1

# %%
# Replacing DF1 for GH1 when DF1 > GH1
sensor_ts_avg.loc[
    sensor_ts_avg["DF1"] > sensor_ts_avg["GH1"], "DF1"
] = sensor_ts_avg["GH1"]


# %% [markdown]
```

```
# ### 3.4. Removing GHI = 0 values

# %%
# Using the secondary standard pyranometer as reference
# to filter the night values (GHI = 0)
sensor_ts_avg = sensor_ts_avg[sensor_ts_avg["GHI"] != 0]

# %% [markdown]
# ### 3.4. Removing Bias from Sunshine (SPN1)
#
# Linear regression is an effective method for reducing bias, as we will see shortly. In this case, I applied it primarily because the SPN1 is a first-class pyranometer, whereas the SR20-D2 is a secondary standard. Additionally, since the GHI is composed of two SR20-D2 sensors, as previously mentioned, we can use them to calibrate the SPN1 sensor's GHI. Once calibrated, we can apply the same linear regression to the diffuse component. This approach is valid because the SPN1 uses the same set of sensors to capture both GHI and DIF.

# %%
# Analyzing residue histogram
residue = sensor_ts_avg["GH1"] - sensor_ts_avg["GHI"]

# Calculate the mean and standard deviation of the residue
mu = residue.mean()
sigma = residue.std()

# Set up the plot with density scaling so that the PDF can be overlaid directly.
plt.figure(figsize=(10, 6))
sns.histplot(residue, kde=True, bins=300, stat="density")
plt.xlim(-50, 50)

# Create an array of x values for the normal distribution curve
x = np.linspace(-50, 50, 1000)
# Compute the normal distribution PDF using the calculated mu and sigma
pdf = stats.norm.pdf(x, mu, sigma)

# Plot the normal distribution as a red line
plt.plot(x, pdf, color='red', lw=2, label='Normal Distribution')

# Add a dashed vertical line at x=0
plt.axvline(x=0, color='black', linestyle='--', linewidth=1)
```

```
plt.title("Histogram of the Residue Before Calibration (GH1 - GHI)")
plt.xlabel("Residue")
plt.ylabel("Density")
plt.legend()
plt.show()


# %%
# We can fix bias with linear regression
sunshine_lr = LinearRegression()

sunshine_lr.fit(
    sensor_ts_avg["GH1"].values.reshape(-1, 1),
    sensor_ts_avg["GHI"].values.reshape(-1, 1)
)

sensor_ts_avg["GH1"] = sunshine_lr.predict(
    sensor_ts_avg["GH1"].values.reshape(-1, 1)
)

# %%
# Analyzing residue histogram
residue = sensor_ts_avg["GH1"] - sensor_ts_avg["GHI"]

# Calculate the mean and standard deviation of the residue
mu = residue.mean()
sigma = residue.std()

# Set up the plot with density scaling so that the PDF can be overlaid
directly.
plt.figure(figsize=(10, 6))
sns.histplot(residue, kde=True, bins=300, stat="density")
plt.xlim(-50, 50)

# Create an array of x values for the normal distribution curve
x = np.linspace(-50, 50, 1000)
# Compute the normal distribution PDF using the calculated mu and sigma
pdf = stats.norm.pdf(x, mu, sigma)

# Plot the normal distribution as a red line
plt.plot(x, pdf, color='red', lw=2, label='Normal Distribution')
```

```
# Add a dashed vertical line at x=0
plt.axvline(x=0, color='black', linestyle='--', linewidth=1)

plt.title("Histogram of the Residue After Calibration (GH1 - GHI)")
plt.xlabel("Residue")
plt.ylabel("Density")
plt.legend()
plt.show()

# %%
# We can use the same calibration for SPN1 diffuse.
sensor_ts_avg["DF1"] = sunshine_lr.predict(
    sensor_ts_avg["DF1"].values.reshape(-1, 1)
)

# %%
# After Calibration, we can have some negative values, so
# w'll repeat this:
sensor_ts_avg = sensor_ts_avg[
    (sensor_ts_avg["DF1"] >= 0) &
    (sensor_ts_avg["GH1"] >= 0)
]

# %% [markdown]
# ### 3.5. Filtering common index

# %%
# Get common index
common_index = get_common_index([solargis_ts, sensor_ts_avg])

# %%
# Filter both dataframes to common index
solargis_filtered = solargis_ts[solargis_ts.index.isin(common_index)]
sensor_filtered = sensor_ts_avg[sensor_ts_avg.index.isin(common_index)]

# %% [markdown]
# ## 4. Feature Engineering

# %% [markdown]
# ### 4.1. Adding new features

# %%
# Add some parameters for station
```

```
# Get resolution
dt_minutes = get_minutes_resolution_from_time(sensor_filtered.index)

# %%
# Add solar position
sensor_filtered = add_solar_position_columns(
    data=sensor_filtered,
    dt_minutes=dt_minutes,
    latitude=latitude,
    longitude=longitude,
    correct_sun_elevation=True
)

# Add KD Column
sensor_filtered = add_kd_column(sensor_filtered, 'GH1', 'DF1')

# Add KT Column
sensor_filtered = add_kt_column(sensor_filtered, 'GHI', 'TOA', 'ELV')

# Calculating DNI using SPN1 data
sensor_filtered['DNI']        =        (sensor_filtered['GH1']        -
sensor_filtered['DF1']) / cosd(sensor_filtered['ZEN'])

# %%
# Add some parameters for Solargis
solargis_filtered = add_solar_position_columns(
    data=solargis_filtered,
    dt_minutes=dt_minutes,
    latitude=latitude,
    longitude=longitude,
    correct_sun_elevation=True
)

# Add KD column
solargis_filtered = add_kd_column(solargis_filtered, 'GHI', 'DIF')

# Add KT column
solargis_filtered   =   add_kt_column(solargis_filtered,   'GHI',   'TOA',
'ELV')

# %%
# Get ClearSky values
cs_df = get_clear_sky_values(
```

```
    solargis_filtered.index,
    latitude,
    longitude,
    altitude
)

# Add ClearSky Columns
solargis_filtered['GHI_CS'] = cs_df['GHI']
solargis_filtered['DNI_CS'] = cs_df['DNI']
solargis_filtered['DIF_CS'] = cs_df['DIF']
solargis_filtered['KT_CS'] = cs_df['KT']

# %%
# SE and SA is Sun Elevation and Sun Azimuth from solargis, but we'll
use the ZEN and AZ calculated.
df = solargis_filtered[
    [
        'GHI', 'DNI', 'DIF', 'flagR', 'TEMP',
        'AP', 'RH', 'WS', 'WG', 'WD', 'PREC',
        'PWAT', 'ZEN', 'ELV', 'AZ', 'TOA', 'KD',
        'KT', 'GHI_CS', 'DNI_CS', 'DIF_CS', 'KT_CS'
    ]
].copy()

# %%
# Adding sensor data
df.loc[:, 'GHI_SPN1'] = sensor_filtered['GH1'].copy()
df.loc[:, 'DIF_SPN1'] = sensor_filtered['DF1'].copy()
df.loc[:, 'KD_SPN1'] = sensor_filtered['KD'].copy()
df.loc[:, 'GHI_PYR'] = sensor_filtered['GHI'].copy()
df.loc[:, 'KT_PYR'] = sensor_filtered['KT'].copy()

# %%
# W'll use cos(ZEN) instead of ZEN in Regressions
df["cZEN"] = cosd(df["ZEN"])

# %% [markdown]
# ### 4.2. Removing KD and KT out of range

# %%
# Remove any absurd value of KT
df = df[
    ((df["KT_PYR"] < 0) | (df["KT_PYR"] > 1)) == False
```

```
]

# %%
# Remove any absurd value of KD
df = df[
    ((df["KD_SPN1"] < 0) | (df["KD_SPN1"] > 1)) == False
]

# %% [markdown]
# ## 5. Exploratory Data Analysis

# %% [markdown]
# ### 5.1. Daily Profile and Sun Path
#
# Analyze the data based in Zenithal and Azimuthal angles.

# %%
# Daily Profile Graph
# Here we'll visualize the mean GHI distribution by hour.
#  It's useful to visualize if all sources are in the same hour
reference.

# Get hour from index
df['hour'] = df.index.hour

# Mean GHI by hour
hourly_ghi = df.groupby('hour')['GHI'].mean()
hourly_ghi_pyr = df.groupby('hour')['GHI_PYR'].mean()
hourly_ghi_spn1 = df.groupby('hour')['GHI_SPN1'].mean()


plt.figure(figsize=(10, 7))
hourly_ghi.plot(marker='o')
hourly_ghi_pyr.plot(marker='o')
hourly_ghi_spn1.plot(marker='o')
plt.title('Daily Profile')
plt.xlabel('Hour')
plt.ylabel('GHI (mean)')
plt.grid()
plt.legend()
plt.show()

# %%
```

```
# SunPath Graph
# Converting Azimuthal angles to radians
theta = np.radians(df['AZ'].values)

# Use elevation as radius
r = df['ELV'].values

fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(polar=True)

# Using GHI as color
sc = ax.scatter(theta, r, c=df['GHI_PYR'], cmap='viridis', s=5)
plt.colorbar(sc, label='GHI')
ax.set_title('SunPath (Azimuthal vs Elevation)')

# 0° in North
ax.set_theta_zero_location("N")

# Clockwise
ax.set_theta_direction(-1)

# Max Elevation = 90°
ax.set_rmax(90)

plt.show()


# %% [markdown]
# ### 5.2. Solargis GHI and DIF Bias
#
# Analyze the data bias for GHI and DIF.

# %%
# Analyzing residue histogram
residue = df["GHI"] - df["GHI_PYR"]

# Calculate the mean and standard deviation of the residue
mu = residue.mean()
sigma = residue.std()

# Set up the plot with density scaling so that the PDF can be overlaid
directly.
plt.figure(figsize=(10, 6))
```

```
sns.histplot(residue, kde=True, bins=100, stat="density")
plt.xlim(-300, 300)

# Create an array of x values for the normal distribution curve
x = np.linspace(-300, 300, 1000)
# Compute the normal distribution PDF using the calculated mu and sigma
pdf = stats.norm.pdf(x, mu, sigma)

# Plot the normal distribution as a red line
plt.plot(x, pdf, color='red', lw=2, label='Normal Distribution')

# Add a dashed vertical line at x=0
plt.axvline(x=0, color='black', linestyle='--', linewidth=1)

plt.title("Histogram of the Residue Before Calibration")
plt.xlabel("Residue")
plt.ylabel("Density")
plt.legend()
plt.show()

# %%
# Analyzing residue histogram
residue = df["DIF"] - df["DIF_SPN1"]

# Calculate the mean and standard deviation of the residue
mu = residue.mean()
sigma = residue.std()

# Set up the plot with density scaling so that the PDF can be overlaid
directly.
plt.figure(figsize=(10, 6))
sns.histplot(residue, kde=True, bins=100, stat="density")
plt.xlim(-200, 200)

# Create an array of x values for the normal distribution curve
x = np.linspace(-200, 200, 1000)
# Compute the normal distribution PDF using the calculated mu and sigma
pdf = stats.norm.pdf(x, mu, sigma)

# Plot the normal distribution as a red line
plt.plot(x, pdf, color='red', lw=2, label='Normal Distribution')

# Add a dashed vertical line at x=0
```

```
plt.axvline(x=0, color='black', linestyle='--', linewidth=1)

plt.title("Histogram of the Residue Before Calibration")
plt.xlabel("Residue")
plt.ylabel("Density")
plt.legend()
plt.show()

# %% [markdown]
# ### 5.3. Satellite x Sensor
#
# Analyzing GHI, DIF, KT, KD from satellite versus sensor. Also getting
R², NRMSE and NMBE.

# %%
# Plot GHI x GHI Pyranometer
x = df["GHI"].values
y = df["GHI_PYR"].values

plot_performance_comparison(
    x=x,
    y=y,
    x_label='GHI',
    y_label='GHI_PYR',
    diff_label='GHI - GHI_PYR',
    diff_title='Histograma: GHI - GHI_PYR'
)

plt.show()

# %%
# Plot DIF x DIF Pyranometer
x = df["DIF"].values
y = df["DIF_SPN1"].values

plot_performance_comparison(
    x=x,
    y=y,
    x_label='DIF',
    y_label='DIF_SPN1',
    diff_label='DIF - DIF_SPN1',
    diff_title='Histograma: DIF - DIF_SPN1'
)
```

```
# %%
# Plot KT x KT Pyranometer
x = df["KT"].values
y = df["KT_PYR"].values

plot_performance_comparison(
    x=x,
    y=y,
    x_label='KT',
    y_label='KT_PYR',
    diff_label='KT - KT_PYR',
    diff_title='Histograma: KT - KT_PYR'
)

plt.show()

# %%
# Plot KD x KD Pyranometer
x = df["KD"].values
y = df["KD_SPN1"].values

plot_performance_comparison(
    x=x,
    y=y,
    x_label='KD',
    y_label='KD_SPN1',
    diff_label='KD - KD_SPN1',
    diff_title='Histograma: KD - KD_SPN1'
)

plt.show()

# %% [markdown]
# ### 5.4. Heteroscedasticity Analyze
#
# Heteroscedasticity refers to the presence of non-constant variance in
the errors of a regression model, which can undermine the reliability
of standard statistical inferences. To detect heteroscedasticity, we
can use statistical tests such as Breusch-Pagan and White.
#
# ##### **Breusch-Pagan Test**
#  The Breusch-Pagan test assesses whether the variance of residuals
```

```
depends on the independent variables. It provides:
# - **LM statistic (Lagrange Multiplier)** and its **p-value**
# - **F statistic** and its **p-value**
#
# To interpret the results:
# - If the p-value (especially for the LM or F statistic) is **less than 0.05** (or the defined significance level **α**), reject the null hypothesis (**H₀**) of homoscedasticity → **evidence of heteroscedasticity** is present.
# - If the p-value is **greater than 0.05**, do not reject **H₀** → **no evidence of heteroscedasticity** is found.
#
# ##### **White Test**
# The White test operates similarly but does not assume a specific functional form of heteroscedasticity. It also provides:
# - **LM statistic** and **p-value**
# - **F statistic** and **p-value**
#
# The interpretation follows the same principle:
# - If the p-value is below the significance threshold, reject **H₀**, suggesting **heteroscedasticity** in the model.
# - If the p-value is above the threshold, do not reject **H₀**, indicating **homoscedasticity** (constant variance of residuals).
#
# Both tests help determine whether the variance of errors changes across observations, which, if present, may require corrective measures such as weighted least squares (WLS) or robust standard errors to ensure reliable inference.

# %% [markdown]
# #### 4.4.1. GHI

# %%
# Create subplots
fig, axs = plt.subplots(nrows=1, ncols=2, figsize=(12, 5))

# Calculating Residue
residue = df["GHI"] - df["GHI_PYR"]

# Scatter plot with marker size and color based on density
xy = np.vstack([df["GHI_PYR"], residue])
z = gaussian_kde(xy)(xy)
size_marker_factor = 100 / z.max()
```

```
# Scatter plot
sc = axs[0].scatter(
    df["GHI_PYR"],
    residue,
    s=z * size_marker_factor,
    c=z,
    cmap='viridis'
)

# Adding labels and title to the subplot
axs[0].set_xlabel('GHI SR20-D2')
axs[0].set_ylabel('MBE')
axs[0].set_title("Bias Error of Solargis GHI")
axs[0].set_ylim(-750, 750)

# Add trendline
x = np.array(df["GHI_PYR"])
y = np.array(residue)
z = np.polyfit(x, y, 1)
p = np.poly1d(z)
axs[0].plot(x, p(x), "r--", label='Trend Line')

# Scatter plot with marker size and color based on density
xy = np.vstack([df["GHI_PYR"], residue**2])
z = gaussian_kde(xy)(xy)
size_marker_factor = 100 / z.max()

# Scatter plot
sc = axs[1].scatter(
    df["GHI_PYR"],
    residue**2,
    s=z * size_marker_factor,
    c=z,
    cmap='viridis'
)

# Adding labels and title to the subplot
axs[1].set_xlabel('GHI SR20-D2')
axs[1].set_ylabel('Squared MBE')
axs[1].set_title("Squared Bias Error of Solargis GHI")
axs[1].set_ylim(0,100000)
```

```
# Add trendline
x = np.array(df["GHI_PYR"])
y = np.array(residue**2)
z = np.polyfit(x, y, 1)
p = np.poly1d(z)
axs[1].plot(x, p(x), "r--", label='Trend Line')

# Adjust layout to prevent clipping of titles
plt.tight_layout()

# Show the plot
plt.show()

# %%
# Independent variable
X = df["GHI"]

# Dependent variable
y = df["GHI_PYR"]

# Add a constant (intercept) to the model
X_const = sm.add_constant(X)

# Fit the OLS (Linear Regression) model
modelo = sm.OLS(y, X_const).fit()

# Extract the model residuals and the matrix of explanatory variables
residuos = modelo.resid
exog = modelo.model.exog  # include the constant and GHI_PYR

# Breusch-Pagan test
bp_test = het_breuschpagan(residuos, exog)
lm_stat, lm_p_value, f_stat, f_p_value = bp_test

print("Breusch-Pagan test:")
print(f"LM estatístico = {lm_stat}")
print(f"LM p-valor     = {lm_p_value}")
print(f"F estatístico  = {f_stat}")
print(f"F p-valor      = {f_p_value}")

# White Test
white_test = het_white(residuos, exog)
lm_stat_w, lm_p_value_w, f_stat_w, f_p_value_w = white_test
```

```
print("\nWhite Test:")
print(f"LM estatístico = {lm_stat_w}")
print(f"LM p-valor     = {lm_p_value_w}")
print(f"F estatístico  = {f_stat_w}")
print(f"F p-valor      = {f_p_value_w}")


# %% [markdown]
# #### 4.4.2. DIF

# %%
# Create subplots
fig, axs = plt.subplots(nrows=1, ncols=2, figsize=(12, 5))

# Calculating Residue
residue = df["DIF"] - df["DIF_SPN1"]

# Scatter plot with marker size and color based on density
xy = np.vstack([df["DIF_SPN1"], residue])
z = gaussian_kde(xy)(xy)
size_marker_factor = 100 / z.max()

# Scatter plot
sc = axs[0].scatter(
    df["DIF_SPN1"],
    residue,
    s=z * size_marker_factor,
    c=z,
    cmap='viridis'
)

# Adding labels and title to the subplot
axs[0].set_xlabel('DIF SPN1')
axs[0].set_ylabel('MBE')
axs[0].set_title("Bias Error of Solargis DIF")
axs[0].set_ylim(-400, 400)

# Add trendline
x = np.array(df["DIF_SPN1"])
y = np.array(residue)
z = np.polyfit(x, y, 1)
p = np.poly1d(z)
```

```
axs[0].plot(x, p(x), "r--", label='Trend Line')

# Scatter plot with marker size and color based on density
xy = np.vstack([df["DIF_SPN1"], residue**2])
z = gaussian_kde(xy)(xy)
size_marker_factor = 100 / z.max()

# Scatter plot
sc = axs[1].scatter(
    df["DIF_SPN1"],
    residue**2,
    s=z * size_marker_factor,
    c=z,
    cmap='viridis'
)

# Adding labels and title to the subplot
axs[1].set_xlabel('DIF SPN1')
axs[1].set_ylabel('Squared MBE')
axs[1].set_title("Squared Bias Error of Solargis DIF")
axs[1].set_ylim(0,60000)

# Add trendline
x = np.array(df["DIF_SPN1"])
y = np.array(residue**2)
z = np.polyfit(x, y, 1)
p = np.poly1d(z)
axs[1].plot(x, p(x), "r--", label='Trend Line')

# Adjust layout to prevent clipping of titles
plt.tight_layout()

# Show the plot
plt.show()

# %%
# Independent variable
X = df["DIF"]

# Dependent variable
y = df["DIF_SPN1"]

# Add a constant (intercept) to the model
```

```
X_const = sm.add_constant(X)

# Fit the OLS (Linear Regression) model
modelo = sm.OLS(y, X_const).fit()

# Extract the model residuals and the matrix of explanatory variables
residuos = modelo.resid
exog = modelo.model.exog  # include the constant and DIF_SPN1

# Breusch-Pagan test
bp_test = het_breuschpagan(residuos, exog)
lm_stat, lm_p_value, f_stat, f_p_value = bp_test

print("Breusch-Pagan test:")
print(f"LM estatístico = {lm_stat}")
print(f"LM p-valor     = {lm_p_value}")
print(f"F estatístico  = {f_stat}")
print(f"F p-valor      = {f_p_value}")

# White Test
white_test = het_white(residuos, exog)
lm_stat_w, lm_p_value_w, f_stat_w, f_p_value_w = white_test

print("\nWhite Test:")
print(f"LM estatístico = {lm_stat_w}")
print(f"LM p-valor     = {lm_p_value_w}")
print(f"F estatístico  = {f_stat_w}")
print(f"F p-valor      = {f_p_value_w}")


# %% [markdown]
# ### 5.5. KT, KT Cleasky x ZEN

# %%
# Group the data by elevation (rounded to the nearest degree) and
compute the mean KT_CS per group.
# (If you actually want the maximum value per elevation, replace 'mean'
with 'max'.)
grouped = df.groupby(df['ELV'].round(0))['KT_CS'].mean().dropna()

# Optionally, restrict to elevations 1 through 90 (if needed)
grouped = grouped.loc[1:90]
```

```
# Extract the elevation (x) and corresponding KT_CS (y) values
x = grouped.index.values
y = grouped.values

# Fit an 8th degree polynomial to the grouped data
coefficients = np.polyfit(x, y, 8)
poly_fit = np.poly1d(coefficients)

# Generate a smooth set of x-values for plotting the fitted polynomial
x_smooth = np.linspace(x.min(), x.max(), 300)
y_smooth = poly_fit(x_smooth)

# Plot the original grouped data and the polynomial fit
plt.figure(figsize=(10, 6))
plt.scatter(x, y, label='Mean KT_CS per Elevation', color='blue')
plt.plot(x_smooth,  y_smooth,  label='8th  Order  Polynomial  Fit',
color='red')
plt.xlabel('Elevation (degrees)')
plt.ylabel('KT_CS')
plt.title('KT_CS vs Elevation with Polynomial Fit')
plt.legend()
plt.show()


# %%
# Create subplots
fig, ax = plt.subplots(1, 2, figsize=(18, 8), sharey=True, sharex=True)

labels = [
    'GHI Satellite',
    'GHI SR20-D2'
]

# Calculate the point density
xy = np.vstack([df['ELV'], df['KT']])
z = gaussian_kde(xy)(xy)
size_marker_factor = 10/(z.max())

scatter    =    ax[0].scatter(df['ELV'],    df['KT'],    c=df['GHI'],
cmap='plasma', marker='o', s=size_marker_factor, alpha=0.8)
ax[0].plot(x_smooth,  y_smooth,  color='red',  linewidth=3,  label='KT
Clearsky')
```

```
# Adding labels and title to the first subplot
ax[0].set_xlabel('Solar Elevation Angle (degrees)')
ax[0].set_ylabel('Clearness Index Solargis (KT)')
ax[0].set_title(labels[0])
ax[0].legend()

# Calculate the point density
xy = np.vstack([df['ELV'], df['KT_PYR']])
z = gaussian_kde(xy)(xy)
size_marker_factor = 10/(z.max())

scatter = ax[1].scatter(df['ELV'], df['KT_PYR'], c=df['GHI_PYR'],
cmap='plasma', marker='o', s=size_marker_factor, alpha=0.8)
ax[1].plot(x_smooth, y_smooth, color='red', linewidth=3, label='KT
Clearsky')

# Adding labels and title to the first subplot
ax[1].set_xlabel('Solar Elevation Angle (degrees)')
ax[1].set_ylabel('Clearness Index Solargis (KT)')
ax[1].set_title(labels[1])
ax[1].legend()

# Adjust layout to prevent clipping of titles
plt.tight_layout()

# Show the plot
plt.show()

# %% [markdown]
# ### 5.6. KD x KT

# %%
# Create a figure with 2 subplots side-by-side
fig, axes = plt.subplots(1, 2, figsize=(12, 6), sharey=True)

# First chart: KD vs. KT
sns.scatterplot(data=df, x='KT', y='KD', ax=axes[0], alpha=0.5)
axes[0].set_title("KD vs. KT")
axes[0].set_xlabel("KT")
axes[0].set_ylabel("KD")

# Second chart: KD_SPN1 vs. KT_PYR
sns.scatterplot(data=df, x='KT_PYR', y='KD_SPN1', ax=axes[1],
```

```
alpha=0.5)
axes[1].set_title("KD_SPN1 vs. KT_PYR")
axes[1].set_xlabel("KT_PYR")
axes[1].set_ylabel("KD_SPN1")

plt.tight_layout()
plt.show()

# %% [markdown]
# ### 5.7. GHI x ZEN, AZ

# %%
# Create a figure with 2 subplots side-by-side
fig, axes = plt.subplots(1, 2, figsize=(12, 6), sharey=True)

# First chart: GHI vs. ZEN
sns.scatterplot(
    data=df,
    x='ZEN',
    y='GHI',
    hue='AZ',                # Use the AZ column to color the points
    palette='viridis',       # Choose a colormap palette
    ax=axes[0],
    alpha=0.5
)
axes[0].set_title("GHI vs. ZEN")
axes[0].set_xlabel("ZEN")
axes[0].set_ylabel("GHI")
axes[0].legend(title="AZ")   # Optional: rename legend title

# Second chart: GHI_PYR vs. ZEN
sns.scatterplot(
    data=df,
    x='ZEN',
    y='GHI_PYR',
    hue='AZ',                # Use the AZ column to color the points
    palette='viridis',       # Same colormap for consistency
    ax=axes[1],
    alpha=0.5
)
axes[1].set_title("GHI_PYR vs. ZEN")
axes[1].set_xlabel("ZEN")
axes[1].set_ylabel("GHI_PYR")
```

```
axes[1].legend(title="AZ")    # Optional: rename legend title

plt.tight_layout()
plt.show()

# %%
# Create a figure with 2 subplots side-by-side
fig, axes = plt.subplots(1, 2, figsize=(12, 6), sharey=True)

# First chart: GHI vs. ZEN
sns.scatterplot(
    data=df,
    x='AZ',
    y='GHI',
    hue='ZEN',                 # Use the AZ column to color the points
    palette='viridis',        # Choose a colormap palette
    ax=axes[0],
    alpha=0.5
)
axes[0].set_title("GHI vs. AZ")
axes[0].set_xlabel("AZ")
axes[0].set_ylabel("GHI")
axes[0].legend(title="ZEN")    # Optional: rename legend title

# Second chart: GHI_PYR vs. ZEN
sns.scatterplot(
    data=df,
    x='AZ',
    y='GHI_PYR',
    hue='ZEN',                 # Use the AZ column to color the points
    palette='viridis',        # Same colormap for consistency
    ax=axes[1],
    alpha=0.5
)
axes[1].set_title("GHI_PYR vs. AZ")
axes[1].set_xlabel("AZ")
axes[1].set_ylabel("GHI_PYR")
axes[1].legend(title="ZEN")    # Optional: rename legend title

plt.tight_layout()
plt.show()

# %% [markdown]
```

```
# ### 5.8. DIF x ZEN, AZ

# %%
# Create a figure with 2 subplots side-by-side
fig, axes = plt.subplots(1, 2, figsize=(12, 6), sharey=True)

# First chart: GHI vs. ZEN
sns.scatterplot(
    data=df,
    x='ZEN',
    y='DIF',
    hue='AZ',                # Use the AZ column to color the points
    palette='viridis',       # Choose a colormap palette
    ax=axes[0],
    alpha=0.5
)
axes[0].set_title("DIF vs. ZEN")
axes[0].set_xlabel("ZEN")
axes[0].set_ylabel("DIF")
axes[0].legend(title="AZ")   # Optional: rename legend title

# Second chart: DIF_SPN1 vs. ZEN
sns.scatterplot(
    data=df,
    x='ZEN',
    y='DIF_SPN1',
    hue='AZ',                # Use the AZ column to color the points
    palette='viridis',       # Same colormap for consistency
    ax=axes[1],
    alpha=0.5
)
axes[1].set_title("DIF_SPN1 vs. ZEN")
axes[1].set_xlabel("ZEN")
axes[1].set_ylabel("DIF_SPN1")
axes[1].legend(title="AZ")   # Optional: rename legend title

plt.tight_layout()
plt.show()

# %%
# Create a figure with 2 subplots side-by-side
fig, axes = plt.subplots(1, 2, figsize=(12, 6), sharey=True)
```

```
# First chart: DIF vs. ZEN
sns.scatterplot(
    data=df,
    x='AZ',
    y='DIF',
    hue='ZEN',                # Use the AZ column to color the points
    palette='viridis',       # Choose a colormap palette
    ax=axes[0],
    alpha=0.5
)
axes[0].set_title("DIF vs. AZ")
axes[0].set_xlabel("AZ")
axes[0].set_ylabel("DIF")
axes[0].legend(title="ZEN")   # Optional: rename legend title

# Second chart: DIF_SPN1 vs. ZEN
sns.scatterplot(
    data=df,
    x='AZ',
    y='DIF_SPN1',
    hue='ZEN',                # Use the AZ column to color the points
    palette='viridis',       # Same colormap for consistency
    ax=axes[1],
    alpha=0.5
)
axes[1].set_title("DIF_SPN1 vs. AZ")
axes[1].set_xlabel("AZ")
axes[1].set_ylabel("DIF_SPN1")
axes[1].legend(title="ZEN")   # Optional: rename legend title

plt.tight_layout()
plt.show()

# %% [markdown]
# ### 5.8. Analyze GHI and DIF for ZEN bins

# %% [markdown]
# #### 5.8.1. GHI

# %%
# Visualization based in ZEN interval
# ZEN bins: (start, end)
zen_bins = [
```

```
    (0, 5), (5, 10),
    (10, 15), (15, 20),
    (20, 25), (25, 30),
    (30, 35), (35, 40),
    (40, 45), (45, 50),
    (50, 55), (55, 60),
    (60, 65), (65, 70),
    (70, 75), (75, 80),
    (80, 85), (85, 90)
]

# Prepare a 3x3 grid of subplots
fig, axs = plt.subplots(nrows=6, ncols=3, figsize=(18, 24))
axs = axs.ravel()  # Flatten the 2D array of axes for easy indexing

for i, (low, high) in enumerate(zen_bins):
   # Filter DataFrame for the given ZEN bin
    df_temp = df[(df["ZEN"] >= low) & (df["ZEN"] < high)]

    # If df_temp is empty, just skip plotting
    if df_temp.empty:
        axs[i].set_title(f"ZEN in [{low}, {high}) [No data]")
        axs[i].axis("off")
        continue

    x = df_temp["GHI"]
    y = df_temp["GHI_PYR"]

    # Density estimation
    xy = np.vstack([x, y])
    z = gaussian_kde(xy)(xy)

    # Adjust marker size based on the maximum density value
    size_marker_factor = 100 / z.max()

    # Scatter plot
    sc = axs[i].scatter(
        x,
        y,
        s=z * size_marker_factor,
        c=z,
        cmap='viridis'
    )
```

```
    # Plot identity line (y = x)
    axs[i].plot(x, x, color='red', ls='--')

    # Title, labels
    axs[i].set_title(f"ZEN in [{low}, {high})")
    axs[i].set_xlabel("GHI")
    axs[i].set_ylabel("GHI_PYR")

# Adjust layout
plt.tight_layout()
plt.show()

# %% [markdown]
# #### 5.8.2 DIF

# %%
# Visualization based in ZEN interval
# ZEN bins: (start, end)
zen_bins = [
    (0, 5), (5, 10),
    (10, 15), (15, 20),
    (20, 25), (25, 30),
    (30, 35), (35, 40),
    (40, 45), (45, 50),
    (50, 55), (55, 60),
    (60, 65), (65, 70),
    (70, 75), (75, 80),
    (80, 85), (85, 90)
]

# Prepare a 3x3 grid of subplots
fig, axs = plt.subplots(nrows=6, ncols=3, figsize=(18, 24))
axs = axs.ravel()  # Flatten the 2D array of axes for easy indexing

for i, (low, high) in enumerate(zen_bins):
   # Filter DataFrame for the given ZEN bin
    df_temp = df[(df["ZEN"] >= low) & (df["ZEN"] < high)]

    # If df_temp is empty, just skip plotting
    if df_temp.empty:
        axs[i].set_title(f"ZEN in [{low}, {high}) [No data]")
        axs[i].axis("off")
```

```
            continue

        x = df_temp["DIF"]
        y = df_temp["DIF_SPN1"]

        # Density estimation
        xy = np.vstack([x, y])
        z = gaussian_kde(xy)(xy)

        # Adjust marker size based on the maximum density value
        size_marker_factor = 100 / z.max()

        # Scatter plot
        sc = axs[i].scatter(
            x,
            y,
            s=z * size_marker_factor,
            c=z,
            cmap='viridis'
        )

        # Plot identity line (y = x)
        axs[i].plot(x, x, color='red', ls='--')

        # Title, labels
        axs[i].set_title(f"ZEN in [{low}, {high})")
        axs[i].set_xlabel("DIF")
        axs[i].set_ylabel("DIF_SPN1")

# Adjust layout
plt.tight_layout()
plt.show()

# %% [markdown]
# ## 6. Model Training

# %% [markdown]
# ### 6.1. Get correlation

# %%
# Getting correlation matrix
corr_matrix = df.corr()
```

```
# Filtering only the columns that we want to predict
corr_matrix = corr_matrix.loc[
    ['GHI_SPN1', 'DIF_SPN1', 'KD_SPN1', 'GHI_PYR'],
    :
]

# Set the size of the plot
plt.figure(figsize=(28, 4))

# Generate a heatmap
sns.heatmap(corr_matrix,    annot=True,    fmt=".2f",    cmap='coolwarm',
linewidths=0.5)

# Add title
plt.title('Correlation Matrix Heatmap')

# Show the plot
plt.show()

# %% [markdown]
# ### 6.2. Separating Training and Testing data

# %%
# Define bins
zen_bins = [
    (0, 10), (10, 20), (20, 30), (30, 40),
    (40, 50), (50, 60), (60, 70), (70, 80),
    (80, 91)
]

x_features = ["GHI", "DIF", "cZEN", "ZEN"]
y_target = ["GHI_PYR", "DIF_SPN1"]

# %%
from sklearn.model_selection import train_test_split

X_train = []
y_train = []
X_test = []
y_test = []

for (low, high) in zen_bins:
    # Filter the data for the current bin
```

```
    mask = (df["ZEN"] >= low) & (df["ZEN"] < high)
    df_bin = df[mask]

            X_train_bin,   X_test_bin,   y_train_bin,   y_test_bin   =
train_test_split(
        df_bin[x_features],
        df_bin[y_target],
        test_size=0.2,
        random_state=69,
        shuffle=True
    )

    X_train.append(X_train_bin)
    y_train.append(y_train_bin)
    X_test.append(X_test_bin)
    y_test.append(y_test_bin)

X_train = pd.concat(X_train)
y_train = pd.concat(y_train)
X_test = pd.concat(X_test)
y_test = pd.concat(y_test)

# %% [markdown]
# ### 6.3. Model for GHI
#
#  Fit  models  to  predict  GHI  Pyranometer  using  Solargis  data
(Satellite).

# %%
# Store regression models and coefficients
ghi_bin_models = []

for (low, high) in zen_bins:
    # Filter the data for the current bin
    mask = (X_train["ZEN"] >= low) & (X_train["ZEN"] < high)
    ghi_X_train_bin = X_train[mask]
    ghi_y_train_bin = y_train[mask]

    # Prepare feature matrix (GHI and cos(ZEN)) and target variable
    X = ghi_X_train_bin[["GHI", "cZEN"]].values
    y = ghi_y_train_bin["GHI_PYR"].values

    # Fit linear regression: GHI_PYR = a + b*GHI + c*cos(ZEN)
```

```
    reg = LinearRegression().fit(X, y)

    # Extract coefficients
    slope_GHI, slope_cZEN = reg.coef_
    intercept = reg.intercept_

    # Store model information
    ghi_bin_models.append({
        "zen_range": (low, high),
        "slope_GHI": slope_GHI,
        "slope_cZEN": slope_cZEN,
        "intercept": intercept,
        "model": reg
    })

            print(f"ZEN   [{low},   {high}):   GHI_slope={slope_GHI:.3f},
cZEN_slope={slope_cZEN:.3f}, intercept={intercept:.3f}, n={len(X)}")

# %%
# Prediction
ghi_train_pred_lr_bins = X_train.apply(
          lambda   row:   predict_piecewise_ghi(row["ZEN"],   row["GHI"],
ghi_bin_models), axis=1
)

ghi_test_pred_lr_bins = X_test.apply(
          lambda   row:   predict_piecewise_ghi(row["ZEN"],   row["GHI"],
ghi_bin_models), axis=1
)

ghi_pred_lr_bins = df.apply(
          lambda   row:   predict_piecewise_ghi(row["ZEN"],   row["GHI"],
ghi_bin_models), axis=1
)

# %%
# Applying Linear Regression for all training together, to comparate
with piecewise method.
lr = LinearRegression()

# Separate data
X = X_train[["GHI", "cZEN"]].values.reshape(-1, 2)
X2 = X_test[["GHI", "cZEN"]].values.reshape(-1, 2)
```

```
X3 = df[["GHI", "cZEN"]].values.reshape(-1, 2)
y = y_train[["GHI_PYR"]].values.reshape(-1, 1)

# Fitting
lr.fit(X, y)

# Prediction
ghi_train_pred_lr = lr.predict(X).reshape(1, -1)[0]
ghi_test_pred_lr = lr.predict(X2).reshape(1, -1)[0]
ghi_pred_lr = lr.predict(X3).reshape(1, -1)[0]

# %%
# Fit
poly_models = fit_piecewise_polynomial(
    df,                              # your DataFrame
    zen_bins,                        # ZEN bins
    4,
     feature_cols=["GHI", "cZEN"],    # polynomial will be generated from
these
    target_col="GHI_PYR"             # what we're predicting
)

# Predictions
ghi_train_pred_poly = X_train.apply(
      lambda row: predict_piecewise_polynomial(row["ZEN"], row["GHI"],
poly_models), axis=1
)

ghi_test_pred_poly = X_test.apply(
      lambda row: predict_piecewise_polynomial(row["ZEN"], row["GHI"],
poly_models), axis=1
)

ghi_pred_poly = df.apply(
      lambda row: predict_piecewise_polynomial(row["ZEN"], row["GHI"],
poly_models), axis=1
)

# %% [markdown]
# ### 6.4. Model for DIF
#
#  Fit models to predict DIF Pyranometer using Solargis data
(Satellite).
```

```
# %%
dif_bin_models = []  # Store regression models and coefficients

for (low, high) in zen_bins:
    # Filter the data for the current bin
    mask = (df["ZEN"] >= low) & (df["ZEN"] < high)
    df_bin = df[mask]

    if df_bin.empty:
        continue  # Skip bins without data

    # Prepare feature matrix (DIF and cos(ZEN)) and target variable
    X = df_bin[["DIF", "cZEN"]].values  # Already in 2D shape
    y = df_bin["DIF_SPN1"].values

    # Fit linear regression: DIF_SPN1 = a + b*DIF + c*cos(ZEN)
    reg = LinearRegression().fit(X, y)

    # Extract coefficients
    slope_DIF, slope_cZEN = reg.coef_
    intercept = reg.intercept_

    # Store model information
    dif_bin_models.append({
        "zen_range": (low, high),
        "slope_DIF": slope_DIF,
        "slope_cZEN": slope_cZEN,
        "intercept": intercept,
        "model": reg
    })

          print(f"ZEN   [{low},   {high}):   DIF_slope={slope_DIF:.3f},
cZEN_slope={slope_cZEN:.3f}, intercept={intercept:.3f}, n={len(X)}")

# %%
# Prediction
dif_train_pred_lr_bins = X_train.apply(
          lambda   row:   predict_piecewise_dif(row["ZEN"],   row["DIF"],
dif_bin_models), axis=1
)

dif_test_pred_lr_bins = X_test.apply(
```

```
        lambda row: predict_piecewise_dif(row["ZEN"], row["DIF"],
dif_bin_models), axis=1
)

dif_pred_lr_bins = df.apply(
        lambda row: predict_piecewise_dif(row["ZEN"], row["DIF"],
dif_bin_models), axis=1
)

# %%
# Applying Linear Regression for all training together, to comparate
with piecewise method.
lr = LinearRegression()

# Separate data
X = X_train[["DIF", "cZEN"]].values.reshape(-1, 2)
X2 = X_test[["DIF", "cZEN"]].values.reshape(-1, 2)
X3 = df[["DIF", "cZEN"]].values.reshape(-1, 2)
y = y_train[["DIF_SPN1"]].values.reshape(-1, 1)

# Fitting
lr.fit(X, y)

# Prediction
dif_train_pred_lr = lr.predict(X).reshape(1, -1)[0]
dif_test_pred_lr = lr.predict(X2).reshape(1, -1)[0]
dif_pred_lr = lr.predict(X3).reshape(1, -1)[0]

# %%
# Fit
poly_models = fit_piecewise_polynomial(
    df,                              # your DataFrame
    zen_bins,                        # ZEN bins
    4,
     feature_cols=["DIF", "cZEN"],    # polynomial will be generated from
these
    target_col="DIF_SPN1"            # what we're predicting
)

# Predictions
dif_train_pred_poly = X_train.apply(
      lambda row: predict_piecewise_polynomial(row["ZEN"], row["DIF"],
poly_models), axis=1
```

```
)

dif_test_pred_poly = X_test.apply(
      lambda row: predict_piecewise_polynomial(row["ZEN"], row["DIF"],
poly_models), axis=1
)

dif_pred_poly = df.apply(
      lambda row: predict_piecewise_polynomial(row["ZEN"], row["DIF"],
poly_models), axis=1
)


# %% [markdown]
# ## 7. Model Evaluation

# %% [markdown]
# ### 7.1. GHI

# %%

def compute_metrics(pred, actual):
    return [
        100 * normalized_rmse(pred, actual),
        100 * normalized_mbe(pred, actual),
        r2_score(actual, pred)
    ]

data = [
               ["Solargis    Train",    *compute_metrics(X_train["GHI"],
y_train["GHI_PYR"])],
                ["(LR)    Train",    *compute_metrics(ghi_train_pred_lr,
y_train["GHI_PYR"].values)],
       ["(LR  /  Bins)  Train",  *compute_metrics(ghi_train_pred_lr_bins,
y_train["GHI_PYR"])],
        ["(Poly  +  LR)  Train",  *compute_metrics(ghi_train_pred_poly,
y_train["GHI_PYR"])],

                ["Solargis    Test",    *compute_metrics(X_test["GHI"],
y_test["GHI_PYR"])],
                 ["(LR)    Test",    *compute_metrics(ghi_test_pred_lr,
y_test["GHI_PYR"].values)],
        ["(LR  /  Bins)  Test",  *compute_metrics(ghi_test_pred_lr_bins,
```

```
y_test["GHI_PYR"])],
          ["(Poly  +  LR)  Test",  *compute_metrics(ghi_test_pred_poly,
y_test["GHI_PYR"])],

            ["Solargis  Train  +  Test",  *compute_metrics(df["GHI"],
df["GHI_PYR"])],
             ["(LR)  Train  +  Test",  *compute_metrics(ghi_pred_lr,
df["GHI_PYR"])],
      ["(LR / Bins) Train + Test", *compute_metrics(ghi_pred_lr_bins,
df["GHI_PYR"])],
       ["(Poly  +  LR)  Train  +  Test",  *compute_metrics(ghi_pred_poly,
df["GHI_PYR"])],
]

headers = ["Dataset", "nRSME [%]", "nMBE [%]", "R²"]
print(tabulate(data, headers=headers, tablefmt="grid"))

# %%
# Applying the piecewise Linear Regression
df["GHI_lr"] = ghi_pred_lr
df["GHI_lr_bins"] = ghi_pred_lr_bins
df["GHI_poly"] = ghi_pred_poly

# %%
X_test["GHI_LR"] = ghi_test_pred_lr
X_test["GHI_LR_BINS"] = ghi_test_pred_lr_bins
X_test["GHI_POLY"] = ghi_test_pred_poly

# Lista para armazenar os resultados
results = []

# Lista com os nomes dos modelos
modelos = ["GHI", "GHI_LR", "GHI_LR_BINS", "GHI_POLY"]

# Itera sobre os bins de ZEN
for low, high in zen_bins:
    # Filtra o DataFrame para o intervalo de ZEN
    df_temp = X_test[(X_test["ZEN"] >= low) & (X_test["ZEN"] < high)]

    # Valor médio do ZEN para esse bin (usado no eixo x)
    zen_mid = (low + high) / 2.0

    # Variável alvo
```

```
    y = y_test[y_test.index.isin(df_temp.index)]["GHI_PYR"]

    # Para cada modelo, calcula as métricas e armazena os resultados
    for modelo in modelos:
        x = df_temp[modelo]

        r2_value    = r2_score(y, x)
        nrmse_value = 100 * normalized_rmse(x, y)
        nmbe_value  = 100 * normalized_mbe(x, y)
        nmae_value  = 100 * normalized_mae(x, y)

        results.append({
            "ZEN": zen_mid,
            "Modelo": modelo,
            "R2": r2_value,
            "nRSME": nrmse_value,
            "nMBE": nmbe_value,
            "nMAE": nmae_value
        })

# Converte a lista de resultados para um DataFrame
df_metrics = pd.DataFrame(results)

# Lista das métricas para plotagem
metricas = ["R2", "nRSME", "nMBE", "nMAE"]

# Cria uma figura com 2 linhas e 2 colunas de subplots
fig, axes = plt.subplots(2, 2, figsize=(16, 12))

# Itera sobre as métricas e plota cada uma em seu respectivo subplot
for i, metrica in enumerate(metricas):
    row = i // 2
    col = i % 2
          sns.lineplot(ax=axes[row,  col],  data=df_metrics,  x="ZEN",
y=metrica, hue="Modelo", marker="o")
    axes[row, col].set_title(f"Evolução do {metrica} em função do ZEN")
    axes[row, col].set_xlabel("ZEN (valor médio do bin)")
    axes[row, col].set_ylabel(metrica)
    axes[row, col].grid(True)
    axes[row, col].legend(title="Modelo", loc='best')

plt.tight_layout()
plt.show()
```

```
# %%
# Prepare a subplot grid with 4 columns
fig, axs = plt.subplots(nrows=len(zen_bins), ncols=4, figsize=(30, 50))

for i, (low, high) in enumerate(zen_bins):
    # Filter the DataFrame for the ZEN range
    df_temp = df[(df["ZEN"] >= low) & (df["ZEN"] < high)]

    # Common variable for y-axis
    y = df_temp["GHI_PYR"]

        for j, x_col in enumerate(["GHI", "GHI_lr", "GHI_lr_bins",
"GHI_poly"]):
        x = df_temp[x_col]

        # Density estimation
        xy = np.vstack([x, y])
        z = gaussian_kde(xy)(xy)
        size_marker_factor = 100 / z.max() if z.max() != 0 else 1

        # Scatter plot
        axs[i, j].scatter(
            x,
            y,
            s=z * size_marker_factor,
            c=z,
            cmap='viridis'
        )

        # Tendency Line:
        # Tendency Line:
        if x_col == "GHI_poly":
            # Fit a 4th-degree polynomial to the data
            p_coeffs = np.polyfit(x, y, 4)  # Degree 4 poly
            p = np.poly1d(p_coeffs)
            x_fit = np.linspace(x.min(), x.max(), 100)

                axs[i, j].plot(x_fit, p(x_fit), color='red', ls='--',
label='4th Degree Poly Fit')

        else:
            # Linear trend for other plots
```

```
            m, b = np.polyfit(x, y, 1)  # Linear regression (degree 1)
            x_fit = np.linspace(x.min(), x.max(), 100)

               axs[i, j].plot(x_fit, m*x_fit + b, color='red', ls='--',
label='Linear Fit')

             axs[i, j].set_title(f"ZEN in [{low}, {high})\n{x_col} vs
GHI_PYR")
        axs[i, j].set_xlabel(x_col)
        axs[i, j].set_ylabel("GHI_PYR")

        # Compute and annotate metrics
        r2_value = r2_score(y, x)
        nrmse_value = 100 * normalized_rmse(x, y)
        nmbe_value = 100 * normalized_mbe(x, y)

            axs[i, j].annotate(f'R²: {r2_value:.3f}', xy=(0.65, 0.15),
xycoords='axes fraction')
          axs[i, j].annotate(f'nRMSE [%]: {nrmse_value:.3f}', xy=(0.65,
0.10), xycoords='axes fraction')
            axs[i, j].annotate(f'nMBE [%]: {nmbe_value:.3f}', xy=(0.65,
0.05), xycoords='axes fraction')
        axs[i, j].legend()
        axs[i, j].grid(True)

# Adjust the layout and display the chart
plt.tight_layout()
plt.show()


# %% [markdown]
# ### 7.2. DIF

# %%
data = [
               ["Solargis    Train",    *compute_metrics(X_train["DIF"],
y_train["DIF_SPN1"])],
                ["(LR)    Train",    *compute_metrics(dif_train_pred_lr,
y_train["DIF_SPN1"])],
       ["(LR / Bins) Train", *compute_metrics(dif_train_pred_lr_bins,
y_train["DIF_SPN1"])],
         ["(Poly + LR) Train", *compute_metrics(dif_train_pred_poly,
y_train["DIF_SPN1"])],
```

```
                ["Solargis    Test",    *compute_metrics(X_test["DIF"],
y_test["DIF_SPN1"])],
                 ["(LR)    Test",    *compute_metrics(dif_test_pred_lr,
y_test["DIF_SPN1"])],
        ["(LR  /  Bins)  Test",  *compute_metrics(dif_test_pred_lr_bins,
y_test["DIF_SPN1"])],
         ["(Poly  +  LR)  Test",  *compute_metrics(dif_test_pred_poly,
y_test["DIF_SPN1"])],

           ["Solargis  Train  +  Test",  *compute_metrics(df["DIF"],
df["DIF_SPN1"])],
            ["(LR)  Train  +  Test",  *compute_metrics(dif_pred_lr,
df["DIF_SPN1"])],
      ["(LR / Bins) Train + Test", *compute_metrics(dif_pred_lr_bins,
df["DIF_SPN1"])],
       ["(Poly + LR) Train + Test", *compute_metrics(dif_pred_poly,
df["DIF_SPN1"])],
]

headers = ["Dataset", "nRSME [%]", "nMBE [%]", "R²"]
print(tabulate(data, headers=headers, tablefmt="grid"))

# %%
df["DIF_lr"] = dif_pred_lr
df["DIF_lr_bin"] = dif_pred_lr_bins
df["DIF_poly"] = dif_pred_poly

# %%
X_test["DIF_LR"] = dif_test_pred_lr
X_test["DIF_LR_BIN"] = dif_test_pred_lr_bins
X_test["DIF_POLY"] = dif_test_pred_poly

# Lista para armazenar os resultados
results = []

# Lista com os nomes dos modelos
modelos = ["DIF", "DIF_LR", "DIF_LR_BIN", "DIF_POLY"]

# Itera sobre os bins de ZEN
for low, high in zen_bins:
    # Filtra o DataFrame para o intervalo de ZEN
    df_temp = X_test[(X_test["ZEN"] >= low) & (X_test["ZEN"] < high)]
```

```
    # Valor médio do ZEN para esse bin (usado no eixo x)
    zen_mid = (low + high) / 2.0

    # Variável alvo
    y = y_test[y_test.index.isin(df_temp.index)]["DIF_SPN1"]

    # Para cada modelo, calcula as métricas e armazena os resultados
    for modelo in modelos:
        x = df_temp[modelo]

        r2_value    = r2_score(y, x)
        nrmse_value = 100 * normalized_rmse(x, y)
        nmbe_value  = 100 * normalized_mbe(x, y)
        nmae_value  = 100 * normalized_mae(x, y)

        results.append({
            "ZEN": zen_mid,
            "Modelo": modelo,
            "R2": r2_value,
            "nRSME": nrmse_value,
            "nMBE": nmbe_value,
            "nMAE": nmae_value
        })

# Converte a lista de resultados para um DataFrame
df_metrics = pd.DataFrame(results)

# Lista das métricas para plotagem
metricas = ["R2", "nRSME", "nMBE", "nMAE"]

# Cria uma figura com 2 linhas e 2 colunas de subplots
fig, axes = plt.subplots(2, 2, figsize=(16, 12))

# Itera sobre as métricas e plota cada uma em seu respectivo subplot
for i, metrica in enumerate(metricas):
    row = i // 2
    col = i % 2
          sns.lineplot(ax=axes[row,  col],  data=df_metrics,  x="ZEN",
y=metrica, hue="Modelo", marker="o")
    axes[row, col].set_title(f"Evolução do {metrica} em função do ZEN")
    axes[row, col].set_xlabel("ZEN (valor médio do bin)")
    axes[row, col].set_ylabel(metrica)
```

```
    axes[row, col].grid(True)
    axes[row, col].legend(title="Modelo", loc='best')

plt.tight_layout()
plt.show()

# %%
# Prepare a subplot grid with 4 columns
fig, axs = plt.subplots(nrows=len(zen_bins), ncols=4, figsize=(30, 50))

for i, (low, high) in enumerate(zen_bins):
    # Filter the DataFrame for the ZEN range
    df_temp = df[(df["ZEN"] >= low) & (df["ZEN"] < high)]

    # Common variable for y-axis
    y = df_temp["DIF_SPN1"]

        for j, x_col in enumerate(["DIF", "DIF_lr", "DIF_lr_bin",
"DIF_poly"]):
        x = df_temp[x_col]

        # Density estimation
        xy = np.vstack([x, y])
        z = gaussian_kde(xy)(xy)
        size_marker_factor = 100 / z.max() if z.max() != 0 else 1

        # Scatter plot
        axs[i, j].scatter(
            x,
            y,
            s=z * size_marker_factor,
            c=z,
            cmap='viridis'
        )

        # Tendency Line:
        if x_col == "DIF_poly":
            # Fit a 4th-degree polynomial to the data
            p_coeffs = np.polyfit(x, y, 4)  # Degree 4 poly
            p = np.poly1d(p_coeffs)
            x_fit = np.linspace(x.min(), x.max(), 100)

                axs[i, j].plot(x_fit, p(x_fit), color='red', ls='--',
```

```
label='4th Degree Poly Fit')

        else:
            # Linear trend for other plots
            m, b = np.polyfit(x, y, 1)  # Linear regression (degree 1)
            x_fit = np.linspace(x.min(), x.max(), 100)

               axs[i, j].plot(x_fit, m*x_fit + b, color='red', ls='--',
label='Linear Fit')

             axs[i, j].set_title(f"ZEN in [{low}, {high})\n{x_col} vs
DIF_SPN1")
        axs[i, j].set_xlabel(x_col)
        axs[i, j].set_ylabel("DIF_SPN1")

        # Compute and annotate metrics
        r2_value = r2_score(y, x)
        nrmse_value = 100 * normalized_rmse(x, y)
        nmbe_value = 100 * normalized_mbe(x, y)

            axs[i, j].annotate(f'R²: {r2_value:.3f}', xy=(0.65, 0.15),
xycoords='axes fraction')
          axs[i, j].annotate(f'nRMSE [%]: {nrmse_value:.3f}', xy=(0.65,
0.10), xycoords='axes fraction')
           axs[i, j].annotate(f'nMBE [%]: {nmbe_value:.3f}', xy=(0.65,
0.05), xycoords='axes fraction')
        axs[i, j].legend()
        axs[i, j].grid(True)

# Adjust the layout and display the chart
plt.tight_layout()
plt.show()
```