Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Departamento de Engenharia Elétrica

Anyelle Keila Farias de Queiroz

Ciência de Dados aplicada a Dados Abertos do Operador Nacional do Sistema Elétrico Brasileiro

Campina Grande, PB
Novembro de 2024

Anyelle Keila Farias de Queiroz

Ciência de Dados aplicada a Dados Abertos do Operador Nacional do Sistema Elétrico Brasileiro

Trabalho de Conclusão de Curso submetido à Coordenação do Curso de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para obtenção do grau de Bacharel em Engenharia Elétrica.

Orientador: Prof. Dr. Edmar Candeia Gurjão

Campina Grande, PB
Novembro de 2024

Anyelle Keila Farias de Queiroz

Ciência de Dados aplicada a Dados Abertos do Operador Nacional do Sistema Elétrico Brasileiro

Trabalho de Conclusão de Curso submetido à Coordenação do Curso de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para obtenção do grau de Bacharel em Engenharia Elétrica.

Trabalho aprovado em: _____ / _____ / 2024
Nota: _____

Prof. Dr. Edmar Candeia Gurjão
Orientador

Prof. Dr. Gutemberg Gonçalves dos Santos Júnior
Avaliador

Campina Grande, PB
Novembro de 2024

Agradecimentos

Agradeço primeiramente a Deus por Sua infinita graça e bondade.

Ao meu pai, Raniere, que muito mais do que um exemplo — é um herói. Obrigada pelas incontáveis renúncias, pelas horas dedicadas, e por estar sempre ao meu lado, mesmo quando o caminho parecia árduo. Esta vitória é tão sua quanto minha, pois através do seu exemplo, aprendi que com esforço e retidão, tudo é possível.

À minha mãe, Anamaria, que com sua alegria, leveza e imenso amor, trouxe cor e energia aos meus dias. Sua força e resiliência me inspiram constantemente. Se hoje estou onde estou, é porque cresci à sombra da mulher incrível que você é.

Ao meu irmão, Clayton, meu parceiro de lutas e de sonhos. Deus sabia que eu precisava de você, e por isso me deu o melhor amigo e incentivador que eu poderia ter. Cada conquista só tem verdadeiro sentido porque posso compartilhá-la com você.

Ao meu marido, Hugo, o amor da minha vida e meu grande apoio. Você esteve ao meu lado em cada passo desta graduação, enfrentou comigo inúmeras batalhas, e me deu forças quando eu mais precisei. Sem você, nada disso seria possível.

Ao meu orientador, Edmar, uma verdadeira inspiração como professor e mentor. Que privilégio ter aprendido com alguém que compartilha não só o conhecimento, mas também as conquistas e o caminho a ser trilhado. Sua paciência, dedicação e sabedoria fizeram toda a diferença em minha trajetória, e por isso, minha gratidão será eterna.

"Porque o Senhor dá a sabedoria;
da Sua boca procedem o conhecimento e o entendimento."
Provérbios 2:6

Resumo

Este trabalho tem como objetivo analisar os principais conceitos e técnicas de ciência de dados, aplicando-os aos dados abertos do Operador Nacional do Sistema Elétrico (ONS), com foco especial na Micro e Minigeração Distribuída (MMGD). Para isso, foram desenvolvidos relatórios interativos que permitem uma análise aprofundada dos dados do ONS, além da criação de modelos que visam prever o consumo de energia em diversas regiões do país durante dias específicos, especialmente em situações de variações abruptas de consumo e geração que podem impactar a estabilidade do sistema elétrico. Foram empregadas técnicas de aprendizado de máquina na construção desses modelos, com o objetivo de oferecer subsídios para a tomada de decisões estratégicas, melhorando a resposta do sistema elétrico e contribuindo para um futuro mais sustentável na gestão da energia.

Palavras-chave: Aprendizado de Máquina, Ciência de Dados, Operador Nacional do Sistema Elétrico (ONS).

Abstract

This paper aims to analyze the main concepts and techniques of data science, applying them to open data from the National Electric System Operator (ONS), with a special focus on Micro and Mini Distributed Generation (MMGD). To this end, interactive reports were developed that allow an in-depth analysis of the ONS data, in addition to the creation of models that aim to predict energy consumption in different regions of the country during specific days, especially in situations of abrupt variations in consumption and generation that can impact the stability of the electrical system. Machine learning techniques were used to build these models, with the aim of providing support for strategic decision-making, improving the response of the electrical system and contributing to a more sustainable future in energy management.

Keywords: Machine Learning, Data Science, National Electric System Operator (ONS).

Lista de Figuras

Lista de Abreviaturas e Siglas

AR - Modelos Autorregressivos (do inglês, *Autoregressive Models*)
ARIMA - Modelo Auto-regressivo Integrado de Médias Móveis
ARMA - Modelos Autorregressivos de Média Móvel
CCEE - Câmara de Comercialização de Energia Elétrica
DL - Aprendizado Profundo (do inglês, *Deep Learning*)
IA - Inteligência Artificial
ISO - Operador Independente do Sistema (do inglês, *Independent System Operator*)
ML - Aprendizado de Máquina (do inglês, *Machine Learning*)
MMGD - Micro e Minigeração Distribuída
ONS - Operador Nacional do Sistema Elétrico
SEB - Operação e Planejamento do Sistema Elétrico Brasileiro
SIN - Sistema Interligado Nacional
VAR - Modelos Vetoriais Autorregressivos (do inglês, *Vector Autoregressive Models*)

Sumário

1 Introdução

De acordo com a publicação do Fórum Econômico Mundial (2019) [1], estamos vivenciando uma era em que os conjuntos de dados atingem proporções gigantescas, medidos em zettabytes. Esse cenário de dados volumosos impulsionou o surgimento da ciência de dados, uma disciplina que se dedica à mineração e análise dessas grandes quantidades de informações para identificar padrões e extrair informações valiosas.

A ciência de dados é uma combinação hábil de matemática e estatística, programação especializada, inteligência artificial (IA) e aprendizado de máquina. Ao empregar essas técnicas, empresas tornam-se capazes de processar e interpretar informações, resultando na identificação de padrões cruciais. Possuir o domínio sobre essas técnicas permite que empresas otimizem sua eficiência, controlem custos, identifiquem novas oportunidades de mercado e, em última instância, ganhem uma vantagem competitiva no mercado.

Neste trabalho, será utilizada ciência de dados para analisar e aplicar métodos preditivos nos dados disponibilizados pelo Operador Nacional do Sistema Elétrico (ONS). O Portal de Dados Abertos do ONS foi desenvolvido com o objetivo de facilitar, melhorar e democratizar o acesso e consumo de dados históricos da operação do Sistema Interligado Nacional (SIN). O portal desempenha um papel fundamental na ampliação da digitalização do Operador, tornando-o mais orientado a dados e consolidando-o como um centro de informações valiosas para o setor elétrico e para a sociedade brasileira. Com o acesso às informações fornecidas por este veículo, surge a oportunidade de explorar grandes volumes de dados e aplicar técnicas de ciência de dados.

O objetivo deste trabalho é analisar a demanda e a geração do Sistema Interligado Nacional (SIN) brasileiro, comparando-as com a geração e o consumo da Micro e Mini Geração Distribuída (MMGD). A partir dessa análise comparativa, será aprofundado o estudo sobre o impacto da MMGD no sistema, especialmente no fenômeno conhecido como "rampa"— termo que descreve variações rápidas e acentuadas na demanda de energia em intervalos curtos, exigindo ajustes dinâmicos do sistema para manter o equilíbrio entre consumo e oferta. Com o aumento contínuo da inserção de MMGD, esses efeitos de rampa no consumo tornam-se mais pronunciados. Este trabalho busca desenvolver um modelo que preveja esses efeitos, contribuindo para o planejamento e a estabilidade do sistema elétrico.

1.1 Objetivos

1.1.1 Objetivo Geral

O objetivo geral deste trabalho é realizar uma análise aprofundada dos dados de Micro e Minigeração Distribuída (MMGD), utilizando técnicas de ciência de dados para entender o impacto no Sistema Interligado Nacional (SIN) e desenvolver modelos preditivos capazes de identificar e antecipar variações abruptas de consumo e geração de energia visando auxiliar na gestão eficiente e na operação em tempo real do sistema elétrico.

1.1.2 Objetivos Específicos

- Analisar as plataformas de dados disponibilizadas pelo Operador Nacional do Sistema Elétrico (ONS), identificando as tabelas e informações relevantes relacionadas à Micro e Minigeração Distribuída (MMGD).
- Aplicar técnicas de pré-processamento de dados para limpar, transformar e organizar os conjuntos de dados, facilitando a análise subsequente.
- Desenvolver visualizações interativas (*dashboards*) para monitorar a geração de MMGD e suas variações ao longo do tempo, permitindo uma melhor compreensão do comportamento dos dados.
- Implementar modelos de aprendizado de máquina, focando em técnicas preditivas para identificar e prever variações abruptas de consumo de energia (rampas).

1.2 Metodologia

Inicialmente, foi realizada uma análise detalhada das plataformas de dados disponibilizadas pelas instituições para identificar as tabelas que melhor atendessem aos objetivos. Em seguida, foram aplicadas técnicas de pré-processamento e visualização de dados, com o objetivo de criar *dashboards*, que constituíram o produto inicial do trabalho. Posteriormente, foram conduzidas pesquisas para identificar modelos adequados aos dados selecionados, considerando problemas de classificação ou regressão. Ao final, foi apresentado um modelo de predição capaz de fornecer novas informações relevantes.

2 Fundamentação Teórica

2.1 Ciência de Dados

A recente premiação do Nobel de Física de 2024 a John Hopfield e Geoffrey Hinton, em reconhecimento às suas contribuições fundamentais ao aprendizado de máquina e às redes neurais artificiais, destaca a crescente relevância da ciência de dados no mundo atual [2]. A ciência de dados envolve a coleta, análise e interpretação de vastos volumes de dados, permitindo a extração de informações valiosas e a tomada de decisões bem fundamentadas. Por meio de técnicas avançadas de estatística, aprendizado de máquina e inteligência artificial, essa disciplina transforma dados brutos em informações poderosas, que podem moldar estratégias e ações em diversas áreas, como saúde, negócios, engenharia e mais.

Atualmente, estamos vivendo uma mudança significativa em que o foco se desloca para uma riqueza de aplicações práticas da ciência de dados. Essa transformação é impulsionada por várias razões, entre elas a fusão da computação e das comunicações, que revolucionou a forma como interagimos com a informação. A capacidade aprimorada de observar, coletar e armazenar dados nas ciências naturais, no comércio e em muitos outros setores exige uma reavaliação de nossa compreensão sobre os dados e das metodologias empregadas para manipulá-los no contexto moderno. Além disso, o surgimento da web e das redes sociais como componentes centrais da vida cotidiana não apenas apresenta novos desafios, mas também abre um leque de oportunidades para a teoria e a prática da ciência de dados [3].

É importante ressaltar que a ciência de dados vai muito além da simples análise de dados já existentes. Ela abrange a criação de novos métodos e algoritmos projetados para lidar com dados complexos e em grande escala. Isso inclui o desenvolvimento de modelos preditivos que ajudam a antecipar tendências, a análise de padrões que revelam comportamentos ocultos e a implementação de sistemas que são capazes de aprender e se adaptar ao longo do tempo. A interdisciplinaridade é uma característica fundamental dessa área, reunindo conhecimentos provenientes de matemática, estatística, ciência da computação e áreas específicas de aplicação, como saúde, finanças, marketing, engenharia e muito mais.

A capacidade de transformar dados em informações acionáveis tem um impacto profundo em diversas esferas. No setor da saúde, por exemplo, a ciência de dados pode ser utilizada para prever surtos de doenças, personalizar tratamentos e otimizar processos de diagnóstico. No comércio, as técnicas de ciência de dados são essenciais para a otimização de cadeias de suprimentos, previsão de demanda e aprimoramento da experiência do cliente por meio de recomendações personalizadas. Já na engenharia, essa disciplina é

utilizada para monitorar e prever falhas em sistemas complexos, melhorar a eficiência de processos produtivos e desenvolver novas tecnologias inovadoras.

Assim, a ciência de dados não apenas se destaca como uma área de estudo em ascensão, mas também como uma força motriz que está moldando o futuro de diversas indústrias, tornando-se um componente essencial para o desenvolvimento e a inovação em um mundo cada vez mais orientado por dados.

2.2 Ferramentas de Ciência de Dados

As ferramentas de ciência de dados desempenham um papel fundamental na extração de informações relevantes e no desenvolvimento de soluções eficazes para diversos desafios. Com o uso adequado dessas ferramentas, é possível processar grandes volumes de dados, identificar padrões e gerar análises detalhadas que apoiam a tomada de decisões estratégicas.

2.2.1 Power BI

O Power BI Desktop é uma ferramenta robusta de análise e visualização de dados, que permite aos analistas transformar dados em relatórios interativos, proporcionando aos usuários finais informações anteriormente não visíveis. No setor financeiro, o Power BI pode automatizar a geração de demonstrações de lucros e perdas (P&L) e realizar análises detalhadas de custos ao longo do tempo. Na construção civil, ele permite identificar variações nos prazos de conclusão de projetos, levando em consideração a composição das equipes ou fatores geográficos. No varejo, a ferramenta auxilia na identificação de produtos com melhor desempenho, bem como aqueles com potencial de crescimento, permitindo a realização de análises preditivas e de cenários hipotéticos para impulsioná-los estrategicamente. Na engenharia elétrica, o Power BI pode ser utilizado para monitorar o desempenho de sistemas elétricos, identificar padrões de consumo de energia, otimizar a manutenção preventiva de equipamentos e prever falhas, contribuindo para uma operação mais eficiente e segura.

Segundo dados apresentados pela Microsoft no *Business Applications Summit* de 2021, 97% das empresas da Fortune 500 [4] utilizam o Power BI de alguma forma. Isso demonstra que a tecnologia é amplamente confiável e um investimento estratégico de tempo e recursos, especialmente para aqueles que buscam informações capazes de impulsionar transformações significativas em seus negócios.

O Power BI hoje consiste em uma ampla variedade de produtos que permitem aos usuários criar e consumir relatórios a partir de seus dados. De acordo com a Microsoft (no momento da publicação), esses são todos os componentes que compõem a família de

produtos do Power BI: *Power BI Desktop, Power BI service, Power BI Report Builder, Power BI Report Server on premises, Power BI Mobile, Power BI Embedded.* (Figura 1)

Figura 1: Power BI

Fonte: Power BI, Principais Benefícios e Aplicações, 2019.[5]

Esses componentes oferecem uma gama de funcionalidades, permitindo que os usuários acessem e analisem dados de diversas fontes, colaborem em tempo real e compartilhem *dashboards* de maneira eficaz. O Power BI Desktop, por exemplo, é ideal para a criação de relatórios ricos e detalhados, enquanto o Power BI Service facilita o compartilhamento e a colaboração em relatórios na nuvem.

Essas ferramentas trabalham em conjunto para fornecer uma solução completa de *Business Intelligence*, capacitando organizações a transformar dados brutos em informações acionáveis que podem conduzir estratégias e decisões de negócios mais eficazes. Com a contínua evolução da plataforma e a adição de novos recursos, o Power BI se estabelece como uma das principais soluções de análise de dados no mercado atual.

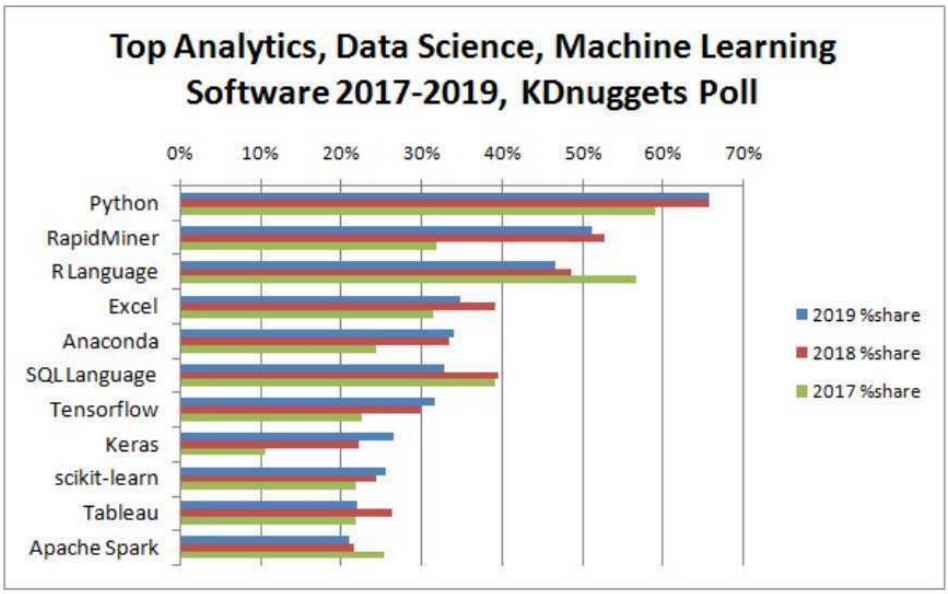
2.2.2 Python

Historicamente, uma ampla gama de diferentes linguagens de programação e ambientes foram usados para permitir a pesquisa de análise de dados, aprendizado de máquina e o desenvolvimento de aplicativos. No entanto, com o crescimento exponencial da popularidade da linguagem *Python* na comunidade de computação científica ao longo da última década, as bibliotecas mais modernas de aprendizado de máquina e aprendizado profundo passaram a ser desenvolvidas em *Python* [6].

Com seu foco principal na legibilidade, *Python* é uma linguagem de programação interpretada de alto nível, amplamente reconhecida por ser fácil de aprender, mas ainda

capaz de aproveitar o poder das linguagens de programação de nível de sistema quando necessário. Além dos benefícios da linguagem em si, a comunidade em torno das ferramentas e bibliotecas disponíveis torna o *Python* particularmente atraente para cargas de trabalho em ciência de dados, aprendizado de máquina e computação científica. De acordo com uma pesquisa da KDnuggets [7] que entrevistou mais de 1.800 participantes sobre preferências em análise, ciência de dados e aprendizado de máquina, *Python* manteve sua posição no topo da linguagem mais amplamente usada em 2019. (Figura 2)

Figura 2: Pesquisa KDnuggets sobre preferências em análise



Fonte: Python leads the 11 top Data Science [7]

A quantidade de dados coletados e gerados hoje é enorme, e os números continuam a crescer em taxas recordes, causando a necessidade de ferramentas que sejam tão performáticas quanto fáceis de usar. A abordagem mais comum para alavancar os pontos fortes do *Python*, como facilidade de uso e ao mesmo tempo garantir eficiência computacional, é desenvolver bibliotecas *Python* eficientes que implementem código de nível inferior escrito em linguagens estaticamente tipadas, como Fortran, C/C++ e CUDA. Nos últimos anos, esforços substanciais estão sendo gastos no desenvolvimento de bibliotecas performáticas, mas amigáveis, para computação científica e aprendizado de máquina.

Nos últimos dez anos, a comunidade *Python* cresceu substancialmente. De acordo com um relatório do *GitHub* [8], esse avanço foi impulsionado principalmente pela rápida expansão de profissionais e entusiastas da ciência de dados. Esse crescimento se deve, em grande parte, à facilidade de uso da linguagem *Python* e ao ecossistema robusto que ela oferece, tornando-a uma escolha ideal para essa área. Isso também se deve à viabilidade do aprendizado profundo, ao crescimento da infraestrutura de nuvem e às

soluções escaláveis de processamento de dados, capazes de lidar com grandes volumes de informações, viabilizando fluxos de trabalho anteriormente intratáveis em um tempo razoável. Esses recursos de computação simples, escaláveis e acelerados permitiram uma insurgência de recursos digitais úteis que estão ajudando a moldar ainda mais a ciência de dados em seu próprio campo distinto, atraindo indivíduos de muitas origens e disciplinas diferentes.

2.2.3 Bibliotecas

As bibliotecas *Python* são coleções de código pré-desenvolvido que ampliam as funcionalidades nativas da linguagem, oferecendo um conjunto de ferramentas e funções prontas para uso. Elas permitem que desenvolvedores realizem tarefas específicas de maneira eficiente, eliminando a necessidade de criar soluções do zero. Essas bibliotecas são amplamente utilizadas em diversas áreas, e em especial na ciência de dados, fornecendo recursos valiosos para análise, extração de informações e visualização de dados. Nesta seção, são listadas as bibliotecas utilizadas: *Statsmodels, Pandas, NumPy, Prophet e Darts.* (Figura 3)

Figura 3: Bibliotecas *Statsmodels, Pandas, NumPy, Prophet e Darts.*

Fonte: Própria Autora.

Statsmodels [1] é um pacote *Python* que fornece uma ampla gama de ferramentas para cálculos estatísticos. Além dos modelos iniciais, como regressão linear, modelos lineares robustos, modelos lineares generalizados e modelos para dados discretos, o *Statsmodels* oferece funcionalidades avançadas para a análise de séries temporais. A versão mais recente inclui ferramentas para estatísticas descritivas, testes estatísticos e diversas classes de modelos lineares voltados para séries temporais, como modelos autorregressivos (AR),

[1] https://www.statsmodels.org/

modelos autorregressivos de média móvel (ARMA) e modelos vetoriais autorregressivos (VAR) [9].

Séries temporais são conjuntos de dados ordenados ao longo do tempo, o que impõe características estocásticas específicas a esses dados. Os modelos presentes no *Statsmodels* assumem que as observações são contínuas, o tempo é discreto e igualmente espaçado, e que não há dados ausentes. Esse tipo de estrutura é comum em diversos campos, especialmente em economia e finanças, onde é utilizada para analisar variáveis como produção nacional, força de trabalho, preços, valores de mercado de ações e volumes de vendas. A ferramenta continua a ser expandida, com novos modelos e extensões em desenvolvimento, proporcionando maior suporte a análises mais complexas de séries temporais.

Pandas [2] é uma biblioteca *Python* que oferece estruturas de dados de alto desempenho e ferramentas projetadas para facilitar análises de dados robustas e eficientes. O principal objetivo da biblioteca é permitir que programadores extraiam rapidamente informações valiosas a partir de grandes conjuntos de dados. Desenvolvido inicialmente em 2008 por Wes McKinney e lançado ao público em 2009 [10], *Pandas* continua a ser mantido e aprimorado por uma comunidade ativa de colaboradores e diversas organizações.

A concepção original do *Pandas* foi voltada para o setor financeiro, com foco em manipulação de séries temporais e processamento de dados históricos, especialmente no mercado de ações. Para enfrentar desafios desse mercado, era essencial uma ferramenta que possibilitasse a recuperação, indexação, limpeza, transformação, combinação e análise de dados tanto unidimensionais quanto multidimensionais, incluindo dados de tipos heterogêneos, que fossem automaticamente alinhados com base em rótulos de índice comuns.*Pandas* foi criado para atender a essas demandas, incorporando uma vasta gama de recursos poderosos, como indexação eficiente, tratamento de dados faltantes e operações rápidas de agregação e remodelagem.

NumPy [3] é um pacote fundamental para a computação científica em *Python*, oferecendo suporte avançado para manipulação de arrays N-dimensionais, operações de transmissão, funções matemáticas como álgebra linear e integração com código C/C++/Fortran. Ele permite que usuários realizem cálculos de alta performance de forma eficiente, preenchendo lacunas na manipulação de dados no *Python*. [11]

Embora *Python* ofereça diferentes estruturas de dados, como listas e dicionários, esses métodos podem ser limitados em termos de eficiência computacional. As listas do *Python*, por exemplo, podem armazenar uma variedade de tipos de objetos, mas realizar operações sobre seus elementos requer loops iterativos, o que pode ser lento. O *NumPy*, por sua

[2] https://pandas.pydata.org
[3] https://numpy.org

vez, oferece o objeto ndarray, que é otimizado para operações rápidas e eficientes sobre grandes volumes de dados.

O ndarray difere das listas *Python* por exigir que todos os elementos armazenados sejam do mesmo tipo, como inteiros, floats ou strings. Enquanto as listas permitem a mistura de diferentes tipos de dados, essa restrição do ndarray é compensada pelo ganho substancial em desempenho, uma vez que as operações sobre arrays *NumPy* são significativamente mais rápidas. Isso o torna uma ferramenta essencial para tarefas computacionais intensivas, como processamento numérico, análise de dados e modelagem científica.

Facebook Prophet [4] é um algoritmo de previsão desenvolvido pela equipe de ciência de dados do *Facebook* em 2017, projetado para ser escalável, rápido e preciso, adequado para diversas aplicações, desde a previsão de vendas no comércio eletrônico até a previsão de padrões climáticos. O algoritmo baseia-se na modelagem de séries temporais como uma combinação de três componentes principais: tendência, sazonalidade e ruído. Ao decompor os dados nesses elementos, o algoritmo consegue gerar previsões precisas, capturando os padrões subjacentes presentes nos dados.

O componente de tendência reflete a direção geral da série temporal, seja ela crescente ou decrescente ao longo do tempo. Esse comportamento é modelado por meio de uma regressão linear por partes, o que confere flexibilidade na adaptação do modelo às variações observadas. O componente de sazonalidade, por sua vez, capta padrões periódicos recorrentes, como tendências semanais ou mensais. O componente de ruído representa as flutuações aleatórias nos dados, que não podem ser explicadas pelos componentes de tendência ou sazonalidade. [12]

Facebook Prophet utiliza uma abordagem Bayesiana para modelar os dados de séries temporais, estimando a distribuição posterior dos parâmetros do modelo em vez de apenas estimativas pontuais. Com isso, o algoritmo consegue gerar previsões probabilísticas que fornecem uma medida de incerteza em torno da previsão central, permitindo uma análise mais robusta e informada dos resultados.

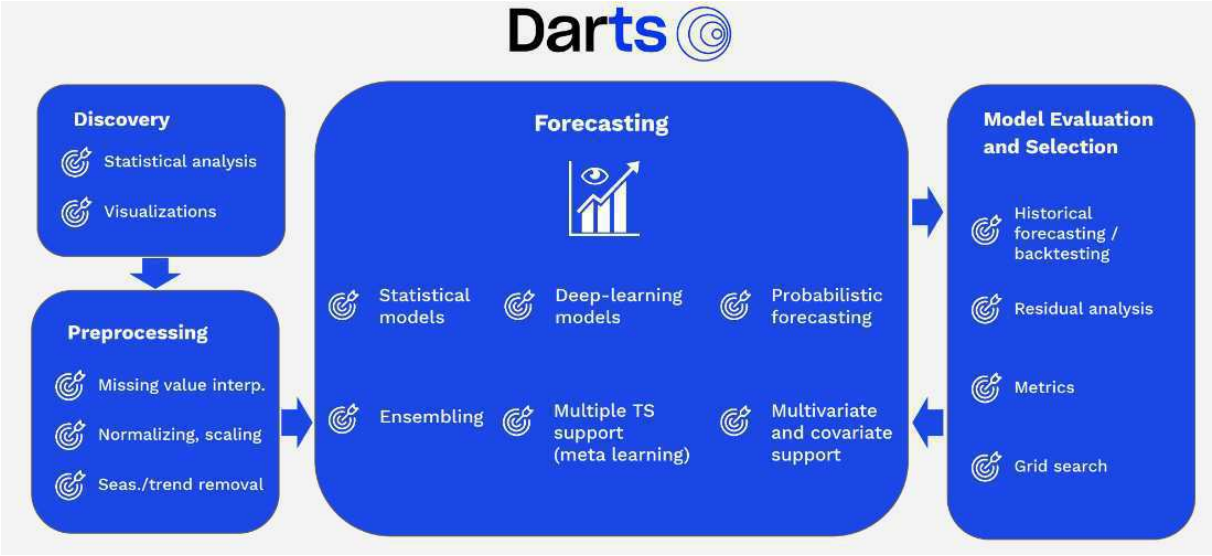
Darts é uma biblioteca *Python* criada em 2020 pela empresa *Unit8* [13] [5], especializada em soluções baseadas em dados. O principal objetivo da *Darts* é facilitar a implementação de modelos de previsão e detecção de anomalias em séries temporais. A biblioteca oferece uma interface unificada para diferentes tipos de modelos, permitindo que usuários trabalhem com algoritmos clássicos, como ARIMA, e também com redes neurais profundas, com suporte para previsões univariadas e multivariadas, além de recursos para previsão

[4] https://facebook.github.io/prophet/
[5] https://unit8co.github.io/darts/

probabilística. [14]

Darts também possui recursos robustos para a detecção de anomalias, facilitando, por exemplo, a aplicação de modelos para calcular pontuações de anomalias em séries temporais. A biblioteca permite encapsular qualquer um de seus modelos de previsão ou filtragem para criar modelos completos de detecção de anomalias. Além disso, os modelos baseados em aprendizado de máquina da biblioteca podem ser treinados em grandes volumes de dados contendo múltiplas séries temporais, sendo que alguns oferecem suporte avançado para previsões probabilísticas, aumentando a precisão e a robustez das previsões. (Figura 4)

Figura 4: *DARTS* - A biblioteca com foco em suavizar a experiência de aprendizado de máquina de séries temporais de ponta a ponta em *Python.*



Fonte: Darts: Time Series Made Easy in Python [13]

Séries temporais representam pontos de dados ao longo do tempo, estando presentes em diversos aspectos da natureza e dos negócios: temperaturas, batimentos cardíacos, dinâmica populacional, tráfego de internet, estoques, inventários, vendas, pedidos e produção industrial, entre outros. Em muitos casos, o processamento e a previsão eficientes dessas séries temporais têm o potencial de gerar vantagens significativas. Previsões podem ajudar empresas a ajustar suas estratégias com antecedência, como planejar a produção de forma proativa, ou otimizar operações ao detectar anomalias em sistemas complexos.

Embora existam diversos modelos e ferramentas para lidar com séries temporais, trabalhar com elas ainda pode ser desafiador, devido às complexidades inerentes a cada método e à falta de uma abordagem unificada para sua aplicação. Neste trabalho, a *Darts* será usada para prever informações críticas a partir dos dados fornecidos pelo Operador Nacional do Sistema Elétrico (ONS), com o objetivo de gerar informações estratégicas e

apoiar a tomada de decisões. A biblioteca será a principal ferramenta de aprendizado de máquina para analisar e prever padrões de consumo e geração de energia, além de detectar possíveis anomalias no sistema elétrico.

Darts foi escolhida por ser uma biblioteca *Python* desenvolvida para simplificar o trabalho com séries temporais, oferecendo uma experiência semelhante a outras bibliotecas amplamente conhecidas, como o *scikit-learn*. Seu foco é tornar mais fácil a previsão de séries temporais e a detecção de anomalias, com suporte para uma vasta gama de modelos, desde os clássicos, como ARIMA, até redes neurais profundas.

Principais vantagens:

- Interface intuitiva para definir e ajustar modelos de forma simples.
- Suporte a diversos tipos de dados de séries temporais.
- Modelos integrados como ARIMA (*AutoRegressive Integrated Moving Average*), Suavização Exponencial, Prophet e LSTM (*Long Short-Term Memory*).
- Ferramentas para ajuste de hiperparâmetros e seleção de modelos, como validação cruzada e *grid search*.
- Recursos para exploração e análise de dados de séries temporais e dos resultados dos modelos.

2.3 Ciência de Dados aplicada a Engenharia Elétrica

O campo da engenharia elétrica tem liderado a inovação tecnológica, impulsionando avanços em setores como sistemas de energia, controle, telecomunicações e eletrônica. Recentemente, a ciência de dados trouxe mudanças significativas para esses domínios, oferecendo novas possibilidades. Com ferramentas como aprendizado de máquina e aprendizado profundo engenheiros agora podem analisar grandes volumes de dados, otimizar o desempenho de sistemas e prever tendências com uma precisão sem precedentes, abrindo novas fronteiras para soluções mais eficientes e inteligentes.

A integração da ciência de dados na engenharia elétrica vai além de um simples avanço tecnológico; é uma verdadeira mudança de paradigma. Com o uso estratégico de ciência de dados, engenheiros eletricistas podem elevar a eficiência, a confiabilidade e a segurança dos sistemas a novos patamares. Essa união possibilita a manutenção preditiva, o monitoramento em tempo real e o controle adaptativo, resultando em redução de custos e melhorias na qualidade do serviço. Em uma era em que os dados são o novo petróleo,

essa convergência entre ciência de dados e engenharia elétrica desbloqueia um vasto leque de novas oportunidades e inovações. [15]

A eletrotécnica tem se beneficiado significativamente da aplicação da ciência de dados, especialmente em áreas como manutenção preditiva e previsão de carga. A manutenção preditiva, por exemplo, utiliza algoritmos de aprendizado de máquina para analisar dados históricos de sistemas de energia e prever falhas de equipamentos antes que ocorram, minimizando o tempo de inatividade e os custos de manutenção. Concessionárias podem monitorar transformadores em tempo real com sensores e programar a manutenção apenas quando necessário, otimizando recursos. Além disso, a previsão de carga, essencial para a eficiência operacional, também é aprimorada por técnicas de aprendizado de máquina, como análise de regressão e redes neurais. Essas técnicas utilizam dados históricos e fatores externos, como o clima, para prever a demanda futura de eletricidade, permitindo que concessionárias ajustem a geração de energia de maneira mais eficaz, evitando desperdícios e melhorando a estabilidade da rede. Essas aplicações já estão em uso em diversas regiões, trazendo melhorias substanciais no gerenciamento de energia. [15]

Em controle, as aplicações de engenharia têm se beneficiado enormemente da integração da ciência de dados, especialmente no que diz respeito a sistemas adaptativos e sistemas autônomos. Nos sistemas de controle adaptativos, algoritmos de aprendizado de máquina podem aprimorar a performance ao prever o comportamento do sistema e ajustar os parâmetros de controle em tempo real, garantindo a otimização mesmo sob condições variáveis. Por exemplo, na automação industrial, modelos de aprendizado de máquina analisam dados históricos de processos para otimizar estratégias de controle, resultando em maior eficiência e redução do tempo de inatividade. Por outro lado, o aprendizado por reforço se destaca na criação de sistemas autônomos que aprendem e se adaptam por meio de interações com o ambiente. Na engenharia de controle, algoritmos de aprendizado por reforço são usados para projetar controladores para sistemas complexos, como braços robóticos e veículos autônomos, que aprendem políticas ótimas através de tentativas e erros, melhorando seu desempenho ao longo do tempo. Um exemplo notável são os controladores baseados em aprendizado por reforço para drones, que conseguem navegar e realizar tarefas de forma autônoma, demonstrando o potencial transformador dessa tecnologia na engenharia de controle.[15]

Em telecomunicações, as aplicações de engenharia têm sido profundamente transformadas pela integração da ciência de dados, especialmente nas áreas de processamento de sinais, otimização de redes e detecção de anomalias. No processamento de sinais, que é fundamental para a engenharia de telecomunicações, técnicas de aprendizado de máquina, como redes neurais e máquinas de vetores de suporte, estão sendo utilizadas para automatizar tarefas complexas, como redução de ruído, classificação de sinais e extração de

características. Por exemplo, algoritmos de aprendizado de máquina podem filtrar ruídos de sinais de áudio, melhorando a clareza e a qualidade das comunicações de voz. Além disso, as redes de telecomunicações, que são intrinsecamente complexas, exigem otimização contínua para garantir uma operação eficiente. Modelos de aprendizado profundo podem analisar grandes volumes de dados de rede para otimizar roteamento, alocação de recursos e gerenciamento de tráfego. Ao prever congestionamentos de rede e identificar os melhores caminhos para a transmissão de dados, esses algoritmos podem melhorar o desempenho da rede e reduzir a latência. Implementações bem-sucedidas de aprendizado profundo para otimização de redes têm resultado em melhorias significativas na qualidade do serviço e na experiência do usuário. Por fim, as redes de telecomunicações estão sujeitas a várias anomalias, como falhas de hardware, ataques cibernéticos e padrões de tráfego inesperados. Algoritmos de aprendizado de máquina podem detectar essas anomalias em tempo real ao analisar dados da rede e identificar desvios do comportamento normal. Por exemplo, modelos de detecção de anomalias podem identificar picos incomuns no tráfego da rede que podem indicar um ataque cibernético, permitindo medidas de mitigação rápidas. A implementação de detecção de anomalias baseada em aprendizado de máquina melhora a confiabilidade e a segurança das redes de telecomunicações.[15]

Por fim, em eletrônica as aplicações da engenharia estão passando por uma transformação significativa com a integração da inteligência artificial e do aprendizado de máquina, trazendo avanços em áreas como design de circuitos, fabricação de semicondutores, controle de qualidade e sistemas embarcados. O design de circuitos eletrônicos, que tradicionalmente é um processo complexo e demorado, pode ser automatizado com técnicas de IA, que geram layouts ótimos com base em requisitos especificados. Modelos de aprendizado de máquina analisam grandes volumes de dados de design para identificar padrões e sugerir melhorias, acelerando o processo e reduzindo a ocorrência de erros. Ferramentas de design de circuitos impulsionadas por IA têm sido utilizadas com sucesso para criar componentes eletrônicos eficientes e confiáveis.[15]

Na fabricação de semicondutores, onde vários processos complexos devem ser rigorosamente controlados para garantir a qualidade do produto, as técnicas de aprendizado de máquina podem otimizar esses processos ao analisar dados de produção e identificar fatores que influenciam o rendimento e a qualidade. Por exemplo, modelos de aprendizado de máquina podem prever os resultados das etapas de fabricação com base em dados de sensores, permitindo ajustes em tempo real que melhoram o rendimento e reduzem defeitos. Além disso, o controle de qualidade na produção de eletrônicos é essencial para garantir a confiabilidade e o desempenho dos produtos finais. Algoritmos de aprendizado de máquina podem prever problemas de qualidade analisando dados de diversas etapas do processo produtivo, permitindo que sistemas de controle de qualidade preditiva identifiquem padrões que indicam possíveis defeitos, possibilitando que os fabricantes tomem

ações corretivas antes que os produtos cheguem ao mercado. Essa abordagem não apenas reduz desperdícios, mas também melhora a qualidade do produto e aumenta a satisfação do cliente.[15]

2.4 Operador Nacional do Sistema Elétrico

2.4.1 Introdução ao Operador Nacional do Sistema Elétrico

O Operador Nacional do Sistema Elétrico (ONS) foi instituído no Brasil para garantir o controle e a operação integrada do sistema elétrico nacional, inspirado em modelos internacionais como *Independent System Operator (ISO)*. O ONS é uma entidade privada e independente que conta com a participação de representantes dos principais agentes do setor elétrico, como geradores, transmissores, distribuidores, consumidores e o governo. Sua principal função é coordenar e supervisionar a operação da rede básica de transmissão de energia elétrica em todo o país, garantindo que a geração e a transmissão ocorram de forma eficiente e segura. [16]

Através de Contratos de Prestação de Serviços de Transmissão, as empresas proprietárias da infraestrutura de transmissão cedem ao ONS o controle operacional de seus ativos. Em contrapartida, recebem receitas que cobrem os custos operacionais e os investimentos realizados. Isso permite ao ONS assumir a responsabilidade pelo gerenciamento centralizado da rede, promovendo uma operação equilibrada e economicamente viável.

As principais atribuições do ONS incluem:

- **Garantir o livre acesso à rede de transmissão** de forma transparente e não discriminatória, permitindo que diferentes agentes do mercado possam utilizá-la de maneira equitativa.
- **Otimizar a operação do sistema elétrico**, incluindo o planejamento da geração e o despacho centralizado de energia, de forma a atender à demanda com o menor custo e maior confiabilidade possível.
- **Incentivar a expansão da infraestrutura elétrica** ao menor custo, promovendo o desenvolvimento sustentável do sistema de transmissão de energia.
- **Administrar as redes básicas de transmissão**, garantindo que a operação seja eficiente e segura para atender a demanda energética do país.

Adicionalmente, o modelo de governança do ONS segue um formato colegiado, garantindo maior transparência e representatividade. Sua estrutura organizacional inclui uma Assembleia Geral, um Conselho de Administração, uma Diretoria Executiva e um

Conselho Fiscal. O ONS também conta com um Comitê de Arbitragem, responsável por mediar e resolver conflitos entre os agentes do setor elétrico, assegurando um ambiente de cooperação e equilíbrio nas operações.

Esse modelo fortalece a confiança no sistema elétrico brasileiro, permitindo maior eficiência operacional e alinhamento com padrões internacionais, ao mesmo tempo que assegura a continuidade no fornecimento de energia de forma estável e economicamente sustentável.

2.4.2 DATATHONS do ONS: Inovação para o Futuro do Setor Elétrico

Diante de um cenário de transformações cada vez mais rápidas e significativas, o Operador Nacional do Sistema Elétrico (ONS) decidiu tomar a iniciativa para se preparar para os desafios do futuro. Foi assim que, em 2020, surgiram os DATATHONS, competições de inovação que agora já estão em sua quinta edição. A ideia por trás dessas competições é mobilizar estudantes de diversas áreas do conhecimento para desenvolver soluções tecnológicas inovadoras.

O DATATHONS também promove uma maior integração entre o setor elétrico e a comunidade de inovação, incentivando o desenvolvimento de tecnologias que possam ser aplicadas em áreas como energias renováveis, otimização de redes e sistemas de monitoramento em tempo real. As soluções geradas por essas competições têm o potencial de transformar a maneira como o setor opera, reforçando o compromisso do ONS com a modernização e a segurança do sistema elétrico brasileiro.

Nas edições anteriores dos DATATHONS, os temas foram projetados para enfrentar desafios específicos e complexos do setor elétrico. Um dos desafios, por exemplo, tratou da Inteligência de Dados de Carga, que busca melhorar a precisão e o gerenciamento da previsão de demanda elétrica. Além disso, houve um foco em Remoção de Viés na Previsão de Vento, um aspecto crítico para garantir previsões mais precisas e eficientes da geração eólica, minimizando erros e melhorando o planejamento energético. Outro desafio importante envolveu o uso de inteligência artificial para validação de dados de contratos de conexão de transmissão [17], um processo essencial para garantir que as conexões entre diferentes partes do sistema elétrico estejam corretamente documentadas e operando dentro dos parâmetros exigidos, contribuindo para uma operação mais confiável e segura. Na quinta e última edição, o tema central foi: "Impactos das Fontes de Energia Renovável na Operação em Tempo Real do Sistema Interligado Nacional (SIN)".

2.4.3 5ª edição DATATHONS: Impactos das Fontes de Energia Renovável na Operação em Tempo Real do Sistema Interligado Nacional (SIN)

No cenário atual da matriz energética, as energias renováveis representam fontes fundamentais impulsionando a transição para sistemas mais sustentáveis. Este paradigma refere-se a recursos naturais que, através de processos naturais ou inesgotáveis, são capazes de gerar energia. A relevância das energias renováveis reside em sua capacidade de fornecer eletricidade de maneira ecologicamente responsável, contribuindo para a redução das emissões de gases de efeito estufa e promovendo a segurança energética.

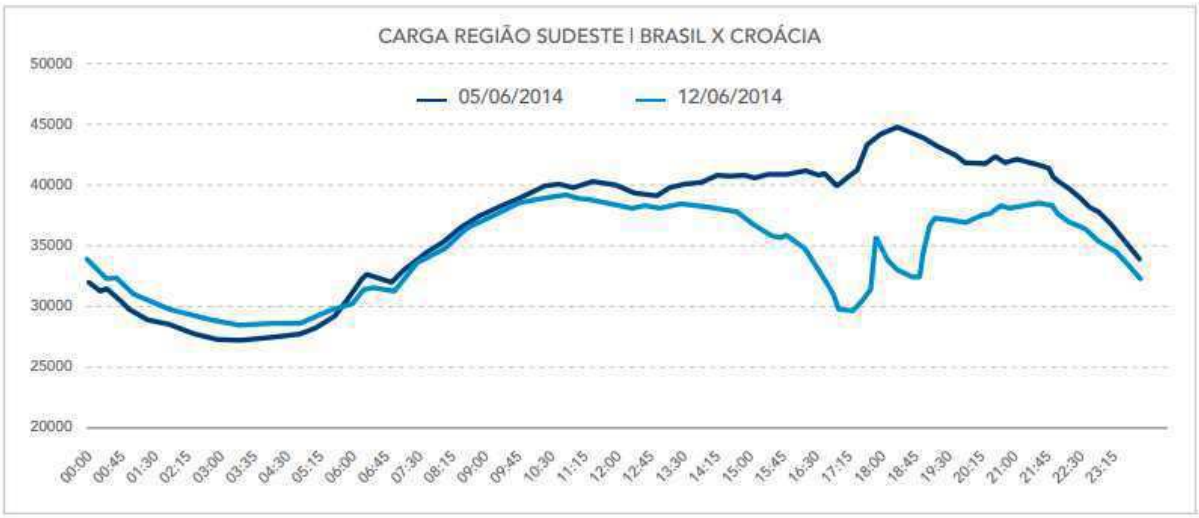
Dentre os exemplos mais preeminentes de energias renováveis, destacam-se a energia solar, proveniente da captação da luz do sol por meio de painéis fotovoltaicos, e a energia eólica, obtida pela conversão da energia cinética do vento em eletricidade. Além disso, a energia hidrelétrica aproveita o potencial energético da água, enquanto a biomassa utiliza matéria orgânica para a geração de eletricidade. Estas fontes, alinhadas às demandas técnicas e ambientais, consolidam-se como pilares essenciais na construção de um mix energético eficiente e sustentável.

A Micro e Minigeração Distribuída (MMGD) emerge como um componente estratégico nesse contexto, representando a descentralização da produção de energia. Este modelo permite que consumidores individuais ou empresas gerem parte de sua eletricidade, muitas vezes por meio de instalações fotovoltaicas ou turbinas eólicas de pequena escala. No ambiente técnico, a MMGD aumenta o nível de incerteza nas previsões de carga e de geração, além de criar situações desafiadoras para a Operação e Planejamento do Sistema Elétrico Brasileiro (SEB). [18]

As rampas de carga, como são chamadas as variações abruptas de consumo, são observadas em situações muito específicas, como em jogos de copa do mundo, em que o consumo varia muito nos horários de início e término dos jogos. Como pode ser visto na Figura 4, a curva em azul escuro representa a curva de carga de um dia normal e em azul

claro a curva de carga de um dia com jogo do Brasil na Copa do Mundo.

Figura 5: Curva de Carga da Região Sudeste comparativa entre dias sem e com jogo do Brasil da Copa do Mundo FIFA 2014.



Fonte: O setor elétrico que não para na hora dos jogos do Brasil na Copa do Mundo. [19]

Com a inserção cada vez maior de fontes renováveis intermitentes, ou MMGD, essas variações abruptas de carga acabam se fazendo mais presentes de forma disfarçada. Observando a carga global, que representa o total de consumo do Brasil, mais as perdas em linhas e equipamento de transmissão, não se consegue perceber de forma direta esse efeito. Porém, quando abatemos a carga atendida por fontes intermitentes, como solar e eólica, obtemos a carga que o ONS efetivamente opera e pode-se observar o efeito de rampa causado por essas fontes.

De forma a garantir o suprimento de energia nessas situações, o Operador dispõe do uso das usinas hidroelétricas para atendimento, pois seu tempo de resposta é rápido, diferente das usinas térmicas que, de maneira geral, não conseguem variar sua geração de forma rápida. No entanto, essas usinas dependem de recursos hídricos e existem muitas restrições devido aos usos múltiplos da água. Quantificar melhor essas variações em situações diferentes é um insumo extremamente relevante para a operação e planejamento do sistema, de forma a garantir um uso otimizado e seguro dos recursos disponíveis.

2.5 Descrição das Tabelas de Dados

Nesta seção, são descritas em detalhe as tabelas de dados fornecidas pela competição do DATATHONS, acompanhadas de explicações sobre as colunas que compõem cada uma. O objetivo é proporcionar um entendimento completo da estrutura dos dados e da função

de cada variável, assegurando uma aplicação eficiente das informações na resolução dos desafios propostos.

2.5.1 Dados de Carga Verificada

Os dados de Carga Verificada contêm informações detalhadas sobre o comportamento das cargas em diferentes áreas e momentos, permitindo uma análise precisa do sistema elétrico. A tabela contêm 561 mil linhas de observações e 11 colunas. A seguir, apresenta-se uma explicação das colunas que compõem essa tabela e seus respectivos significados:

- **Código da Área de Carga:** A coluna identifica a área geográfica à qual a carga está associada. Os dados são balanceados entre as regiões Sudeste e Centro-Oeste (SECO), Sul (S), Nordeste (NE) e Norte (N), com cada uma dessas classes contendo aproximadamente 140 mil registros. Isso assegura uma distribuição uniforme dos dados entre as diferentes regiões, permitindo uma análise equitativa do comportamento de carga em cada área.

- **Data de Referência:** Indica a data exata em que a medição da carga foi realizada, servindo como ponto de referência temporal para todas as demais variáveis.

- **Data de Referência no Intervalo da Semi Hora:** Essa coluna complementa a anterior, mantendo a mesma data de referência, mas acrescentando a informação referente ao intervalo de meia hora. Em outras palavras, isso significa que há medições diárias entre 1º de janeiro de 2023 e 31 de dezembro de 2023, com observações registradas a cada meia hora ao longo do dia. Assim, para cada região, são capturadas 24 observações diárias, proporcionando uma análise granular e contínua do comportamento da carga ao longo do ano.

- **Valor da Carga Global Consistida:** Valor da carga global consistida em MWmed integralizada no final do intervalo da semi hora. A carga global é o total de energia demandada por todas as regiões ou áreas interligadas no sistema elétrico em um dado momento. Esse valor inclui o consumo de todas as unidades consumidoras conectadas a rede elétrica. O termo "consistida"refere-se ao fato de que os dados de carga passaram por um processo de validação para garantir que não há erros de medição.

- **Valor da Consistência:** A coluna corresponde a parcela de consistências feitas na carga para correção de falhas de medidas e dias atípicos.

- **Valor da Carga Supervisionada:** Parcela de carga atendida por geração supervisionada e intercâmbios, provenientes do sistema de supervisão do ONS. Refere-se à porcão da demanda de energia elétrica que é suprida por usinas de geração supervisionadas pelo ONS, ou seja, aquelas que fazem parte do Sistema Interligado

Nacional (SIN) e cuja operação é monitorada e controlada diretamente pelo ONS. Isso inclui grandes geradores de energia, como hidrelétricas, termelétricas e eólicas, que tem sua geração ajustada conforme a necessidade do sistema.

- **Valor da Carga Não Supervisionada:** Parcela da carga atendida por geração não supervisionada provenientes do sistema de medição de faturamento da CCEE. A geração não supervisionada é aquela que não esta sob controle direto e em tempo real do ONS. Isso inclui usinas menores ou geradores distribuídos, como pequenas centrais hidrelétricas, plantas solares de pequeno porte, geradores locais ou mesmo autoprodutores de energia. Esses geradores não participam do despacho centralizado do ONS, ou seja, o ONS não controla ou ajusta a geração dessas unidades.

- **Valor da Carga Global:** Refere-se aos valores de carga global que representam o total da demanda de energia elétrica no SIN. Esses valores são usados para medir o consumo total de energia por todos os consumidores conectados ao SIN que fazem parte do sistema de geração e distribuição de energia no Brasil.

- **Valor da Carga Global Líquida sem MMGD:** Refere-se aos valores de carga global líquida de MMGD (Micro e Mini Geração Distribuída) que é a carga total consumida em uma área descontando a energia gerada por sistemas MMGD. A Micro geração é caracterizada por sistemas com potência de até 75 KW, geralmente instalada em residências ou pequenos comércios. A Mini Geração é caracterizada por sistemas com potência entre 75 KW e 5 MW, esses sistemas podem ser instalados em empresas, industrias ou grandes condomínios. Esses sistemas geram energia localmente e, quando há excedente, essa energia pode ser injetada na rede elétrica, sendo abatida do consumo total por meio de um sistema de compensação. A Carga Global Líquida sem MMGD é o valor da carga total menos a energia gerada por esses sistemas de micro e mini geração. Ou seja, o valor representa a energia que não foi suprida por esses pequenos geradores e que teve que ser fornecida pelas usinas supervisionadas pelo ONS.

- **Valor da Carga Atendida por MMGD:** Parcela da carga de energia elétrica que foi atendida por sistemas de micro e mini geração distribuída como painéis solares e pequenas turbinas eólicas, instalados por consumidores. Esses valores representam a energia que os consumidores geraram e usaram localmente, sem precisar de rede elétrica tradicional.

2.5.2 Dados de Dias Especiais

Os dados relacionados aos Dias Especiais, fornecidos pela competição, contêm 816 linhas de observações e 8 colunas. A seguir, apresenta-se uma explicação das colunas que compõem essa tabela e seus respectivos significados:

- **Identificador do Dia Especial:** Identificador único de cada dia especial.
- **Código da Área de Carga:** A coluna identifica a área geográfica à qual o dia especial está associado. Os dados são balanceados entre as regiões Sudeste e Centro-Oeste (SECO), Sul (S), Nordeste (NE) e Norte (N).
- **Data do Dia Especial:** Data correspondente ao dia especial.
- **Nome do Dia Especial:** Nome correspondente ao dia especial, exemplos: Jogos da Copa do Mundo, Carnaval, Natal.
- **Descrição do Dia Especial:** Descrição correspondente ao dia especial, exemplos: Feriado, Semi feriado, Após feriado.
- **Identificadores para Tipos de Dias Especiais:** Identificador único de cada tipo de dia especial.

2.5.3 Dados de Carga sem MMGD

A tabela de Carga Verificada sem Micro e Mini Geração Distribuída contêm informações detalhadas sobre o comportamento das cargas em diferentes áreas e momentos, permitindo uma análise precisa do sistema elétrico. A tabela contêm aproximadamente 12 milhões linhas de observações e 3 colunas. A seguir, apresenta-se uma explicação das colunas que compõem essa tabela e seus respectivos significados:

- **Instante:** Data e hora de referência. Os dados correspondem a observações registradas a cada minuto ao longo do dia, entre os anos de 2018 e 2024 divididos por região.
- **Valor:** A coluna representa a demanda média integralizada no início do intervalo de medição, expressa em megawatts médios (MWmed). Esse valor reflete a carga elétrica média consumida em um determinado período, considerando as variações ao longo do tempo.
- **Código da Área de Carga:** A coluna identifica a área geográfica à qual o dia especial está associado. Os dados são balanceados entre as regiões Sudeste e Centro-Oeste (SECO), Sul (S), Nordeste (NE) e Norte (N).

2.5.4 Capacidade Instalada de Usina MMGD

Os dados relacionados à Carga Instalada de Micro e Mini Geração Distribuída contêm aproximadamente 85 mil linhas de observações e 5 colunas. A seguir, apresenta-se uma explicação das colunas que compõem essa tabela e seus respectivos significados:

- **Código da Área de Carga:** A coluna identifica a área geográfica à qual o dia especial está associado. Os dados são balanceados entre as regiões Sudeste e Centro-Oeste (SECO), Sul (S), Nordeste (NE) e Norte (N).

- **Tipo de Geração da Usina:** Esta coluna classifica a usina de acordo com o tipo de fonte de energia que ela utiliza para gerar eletricidade. As categorias incluem Central Geradora Hidrelétrica, Central Geradora Eólica, Central Geradora Fotovoltaica e Central Geradora Térmica. As quatro classes identificadas estão equilibradas com 21 mil linhas de observação cada. Essa classificação é importante para entender a contribuição de cada tipo de fonte na matriz energética.

- **Data de Referência da Potência Instalada:** Indica a data em que a potência instalada da usina foi registrada. Essa informação é crucial para acompanhar análises históricas e comparativas.

- **Valor da Potência Instalada:** Refere-se à capacidade total de geração de energia da usina, medida em quilowatts (kW).

- **Valor da Potência Instalada Acumulada:** Esta coluna representa a soma total da potência instalada nas usinas registradas até a data de referência.

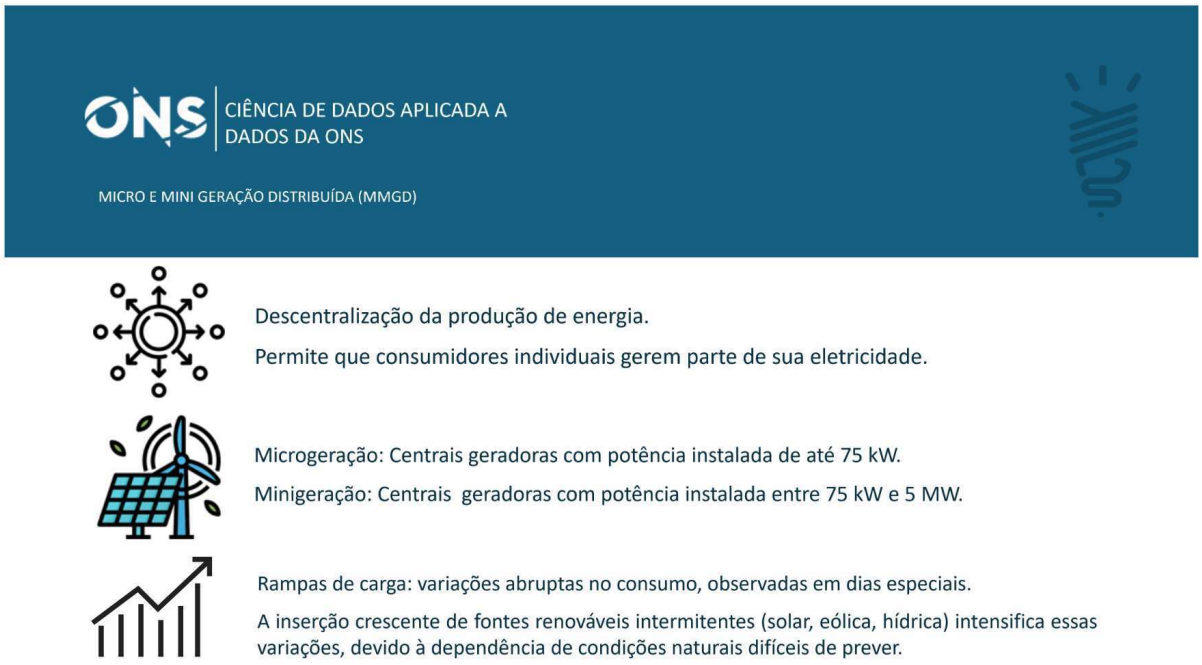
3 Resultados e Discussões

3.1 Visualizações e Análises no Power BI

A primeira etapa para a resolução de um problema como o apresentado envolve a análise detalhada dos dados disponibilizados. Essa análise é essencial para entender as características e padrões dos dados, o que permitirá uma melhor compreensão do problema e ajudará na escolha das abordagens adequadas para solucioná-lo. Nesse contexto, utiliza-se o Power BI como ferramenta principal para conduzir essa análise, aproveitando seus recursos de visualização e exploração de dados para obter uma visão clara e informada sobre as informações fornecidas.

A primeira página do *dashboard* (Figura 6) é fundamental para contextualizar o problema abordado neste trabalho. Ela destaca a descentralização da produção de energia impulsionada pela Micro e Mini Geração Distribuída (MMGD) e expõe os principais desafios técnicos dessa mudança, como as rampas de carga — variações abruptas no consumo de energia. A página também aborda o impacto das fontes renováveis intermitentes, como solar e eólica, que intensificam essas variações devido à dependência de condições naturais.

Figura 6: Página 1 - Contextualização da Descentralização Energética e Desafios da MMGD.



Fonte: Própria Autora.

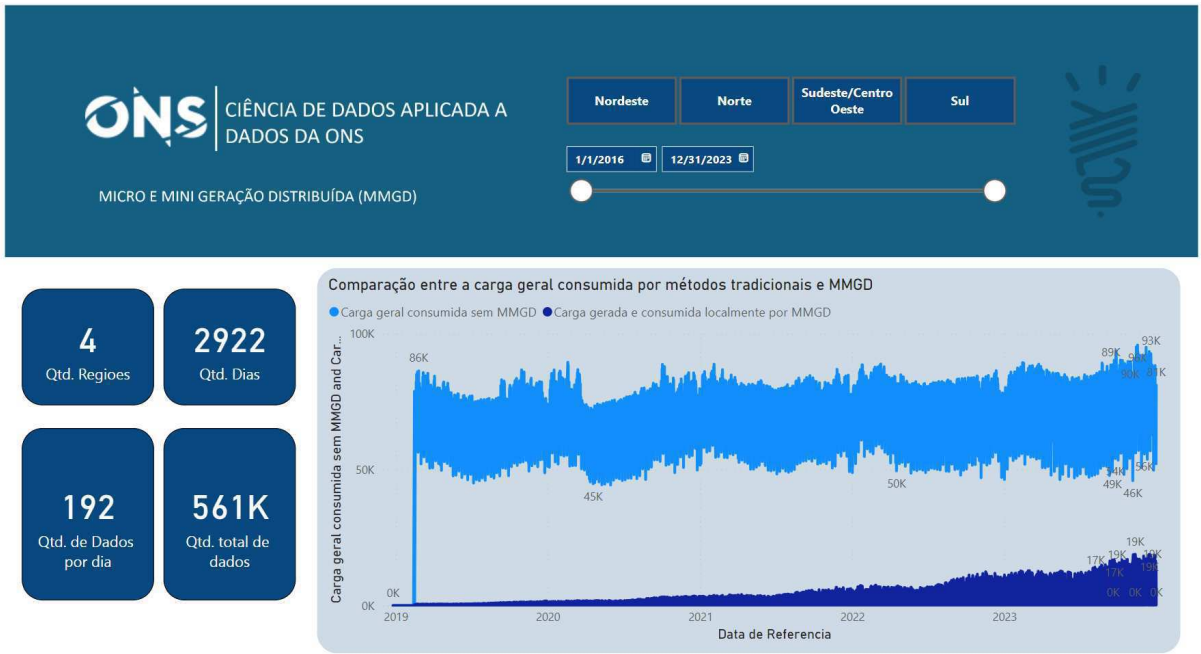
A segunda página do *dashboard* (Figura 7) exibe a análise da primeira tabela de dados disponibilizada pelo ONS: Dados de Carga Verificada. Na página é representada uma visão comparativa da carga geral consumida por métodos tradicionais versus a carga

geral consumida pela Micro e Mini Geração Distribuída (MMGD). Os principais dados apresentados incluem:

- Quantidade de Regiões: 4 regiões estão sendo monitoradas no gráfico.
- Quantidade de Dias: 2922 dias de dados coletados.
- Quantidade de Dados por Dia: 192 registros por dia, capturando a variação da demanda e geração de energia por região.
- Quantidade Total de Dados: Mais de 561 mil registros totais no banco de dados.

O gráfico na parte inferior da Figura 7 faz uma comparação direta entre a carga geral consumida sem MMGD e a carga consumida por MMGD. Ele mostra que, ao longo do tempo, a participação da MMGD no consumo de energia tem aumentado de forma significativa, evidenciada pela crescente área escura no gráfico, representando a contribuição das energias renováveis. Isso destaca a relevância crescente da MMGD na matriz energética e a sua contribuição para a descentralização da geração de energia.

Figura 7: Página 2 - Dados de Carga Verificada.



Fonte: Própria Autora.

A terceira página do *dashboard* (Figura 8) apresenta uma comparação do consumo total de carga utilizando métodos supervisionados e não supervisionados. O gráfico exibe duas linhas principais:

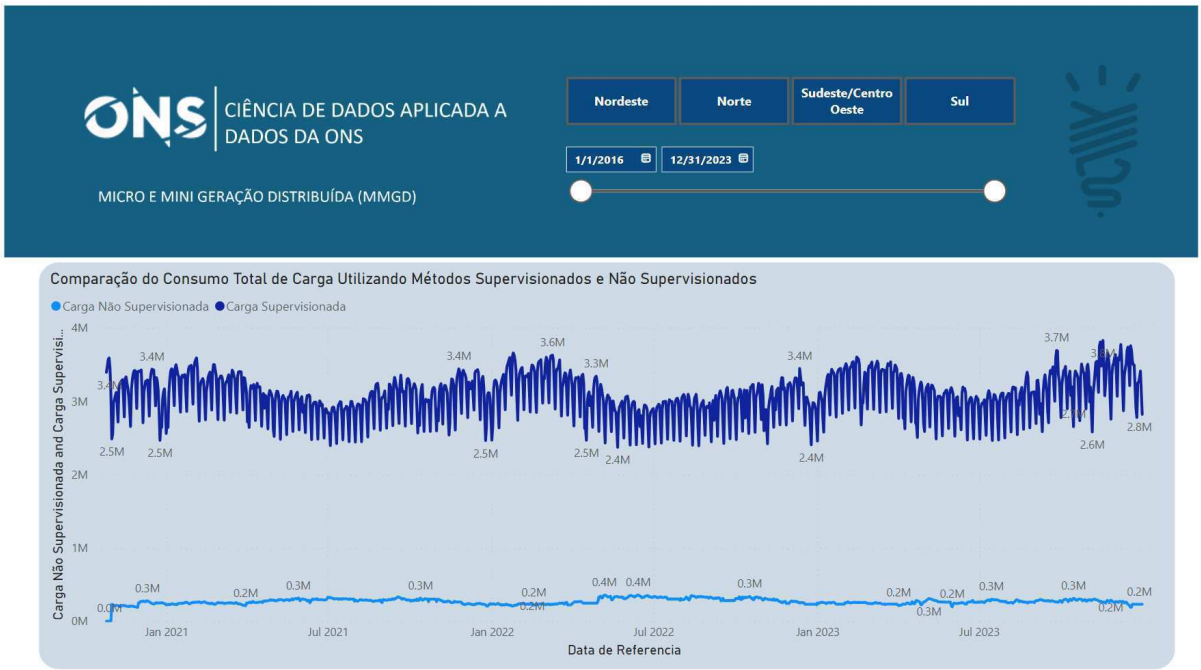
- **Carga Supervisionada:** Representada pela linha azul escuro, mostra uma variação da carga supervisionada (porcão da demanda de energia elétrica que é suprida por

usinas de geração supervisionadas pelo ONS, ou seja, aquelas que fazem parte do SIN) ao longo do tempo, com picos de até 3.7 milhões de MWmed.

- **Carga Não Supervisionada:** Representada pela linha azul claro, com variação bem menor, atingindo picos de aproximadamente 0.3 milhões de MWmed. Esta linha representa a carga não supervisionada, ou seja, aquela que não está sob controle direto e em tempo real do ONS.

A comparação evidencia a diferença na magnitude entre os métodos, destacando a predominância da carga supervisionada na matriz energética. Contudo, a carga não supervisionada, apesar de representar uma parcela menor, é relevante no contexto da MMGD, pois ela ilustra a produção de energia descentralizada e local, associada a fontes renováveis.

Figura 8: Página 3 - Dados de Carga Verificada.



Fonte: Própria Autora.

A quarta página do *dashboard* (Figura 9) exibe a Carga Global Consistida por Região, destacando o consumo de energia nas diferentes regiões do Brasil (Nordeste, Norte, Sudeste/Centro Oeste, e Sul) ao longo do tempo. Cada linha representa o comportamento da carga em uma das regiões.

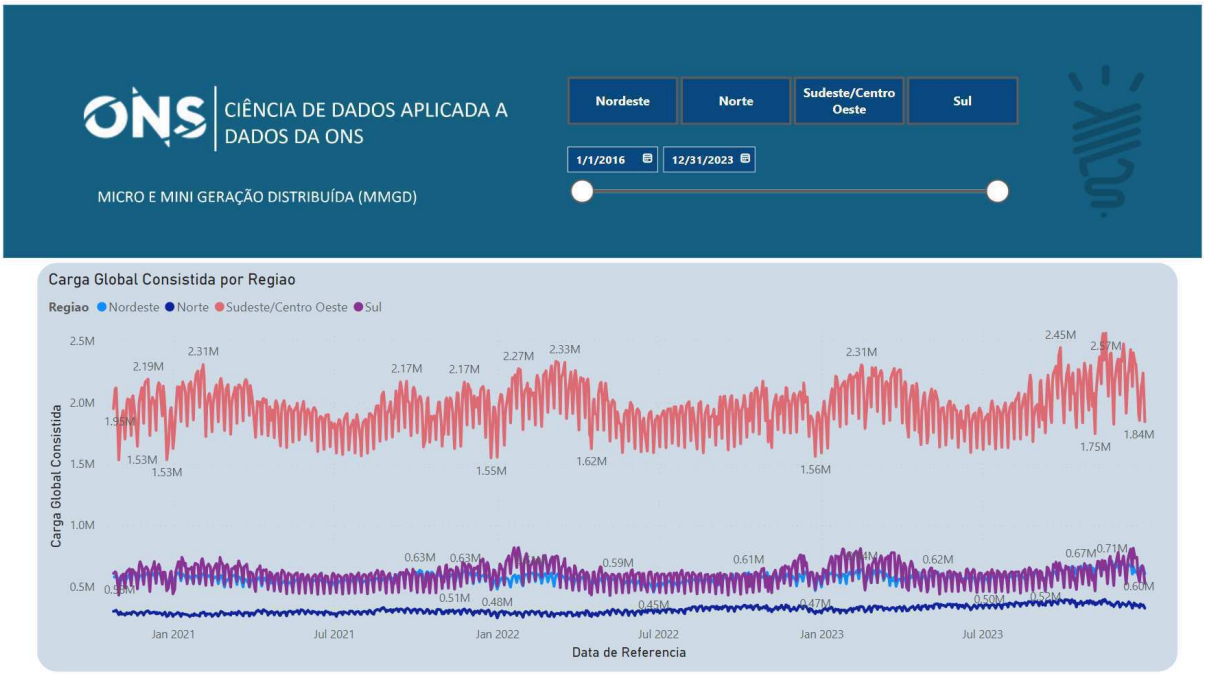
- **Sudeste/Centro Oeste (vermelho):** É a região com o maior consumo de energia, com picos variando entre 2.19M e 2.57M MWmed ao longo do período. Isso reflete a alta concentração de demanda nessas regiões, que incluem grandes centros urbanos e industriais.

- **Nordeste (azul claro) e Sul (roxo):** Essas regiões apresentam níveis intermediários de consumo, com valores que oscilam entre 0.45M e 0.50M MWmed no Nordeste, e entre 0.60M e 0.70M MWmed no Sul, mostrando menor demanda em comparação ao Sudeste/Centro Oeste.

- **Norte (azul escuro):** A região Norte tem o menor consumo de energia, com valores entre 0.47M e 0.52M MWmed, refletindo uma menor densidade populacional e industrial em relação às outras regiões.

O gráfico permite comparar o comportamento da carga entre regiões ao longo de quase três anos, evidenciando as variações sazonais e a predominância do consumo no Sudeste/Centro Oeste.

Figura 9: Página 4 - Dados de Carga Verificada.



Fonte: Própria Autora.

A quinta página do *dashboard* (Figura 10) mostra a potência instalada de diferentes tipos de usinas MMGD nas regiões do Brasil, além da potência total acumulada ao longo do tempo.

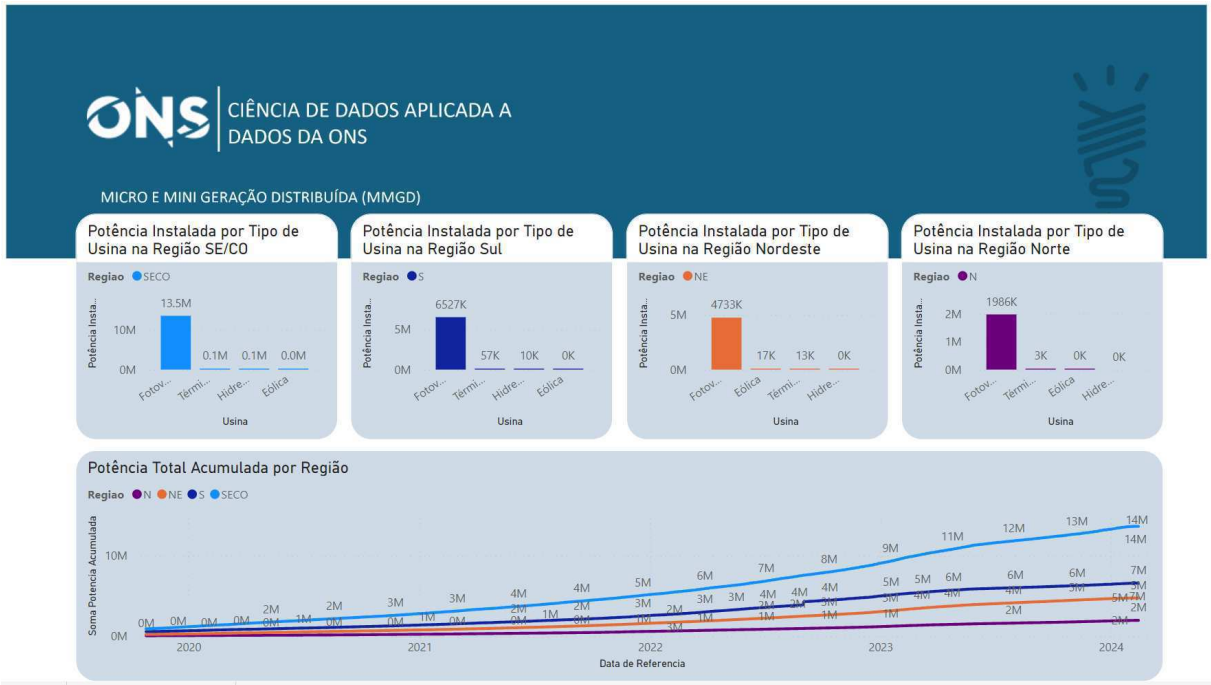
- Na região Sudeste/Centro Oeste, a maior parte da potência instalada é de usinas fotovoltaicas, com 13.5M kW, seguida de pequenas contribuições de térmicas, hídricas e eólicas, cada uma com cerca de 0.1M kW.

- Na região Sul, a maior parte da potência instalada também vem de usinas fotovoltaicas, com 6527k kW. Há pequenas contribuições de térmicas (57k), hídricas (10k) e de usinas eólicas.

- No Nordeste, a potência instalada é mais distribuída, com 4733k kW de usinas fotovoltaicas e 17k kW de eólicas, além de pequenas contribuições de térmicas (13k).
- No Norte, a menor potência instalada, com 1986k kW de usinas fotovoltaicas e 3k kW de térmicas.

O gráfico inferior mostra o crescimento da potência acumulada de geração distribuída ao longo dos anos. A região Sudeste/Centro Oeste apresenta o maior crescimento, ultrapassando 14M kW em 2024, seguida pelo Sul com 7M kW, Nordeste com 6M kW, e Norte com pouco mais de 2M kW.

Figura 10: Página 5 - Potência Instalada MMGD.



Fonte: Própria Autora.

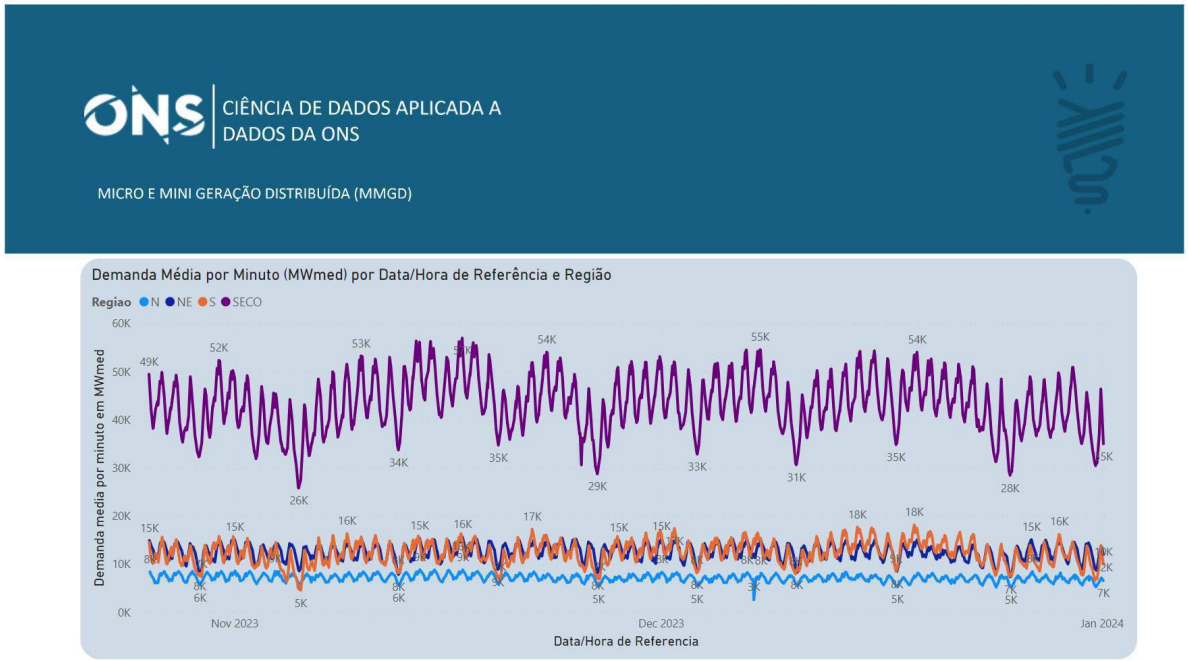
A sexta página do *dashboard* (Figura 11) exibe um gráfico de linhas que apresenta a Demanda Média por Minuto (MWmed) ao longo do tempo com dados referentes a **tabela Dados de Carga sem MMGD**. O gráfico é segmentado por regiões do Brasil: Norte (N), Nordeste (NE) e Sudeste/Centro Oeste (SECO). O gráfico cobre o período de novembro de 2023 até janeiro de 2024, com uma variação de demanda de energia entre essas regiões.

- Região Sudeste e Centro Oeste (em roxo) tem uma demanda média significativamente maior, variando entre 26k MWmed e 55k MWmed, com picos mais acentuados em determinados períodos.
- Região Sul (em laranja) e Região Nordeste (em azul escuro) apresenta uma demanda média por minuto menor, entre 8k MWmed e 18k MWmed, com flutuações regulares.

- Região Norte (em azul claro) tem a menor demanda entre as quatro regiões mostradas, com valores de aproximadamente 3k MWmed a 8k MWmed.

O gráfico destaca um padrão cíclico nas curvas, que provavelmente está relacionado ao consumo de energia que varia ao longo do dia e da semana, com picos e quedas regulares, especialmente mais evidentes na Região Sudeste e Centro Oeste.

Figura 11: Página 6 - Dados de Carga sem MMGD.



Fonte: Própria Autora.

A sétima página do *dashboard* (Figura 12) apresenta uma visão geral da quantidade de dias especiais registrados entre 1 de janeiro de 2016 e 26 de dezembro de 2023, com um total de 816 dias especiais. A tabela central lista dias específicos, como o Natal e feriados relacionados, para cada região (Norte, Nordeste, Sul e Sudeste/Centro-Oeste), incluindo suas respectivas datas.

Os gráficos à direita mostram a distribuição da quantidade de dias especiais por região, onde o Nordeste (NE) lidera com 2,6 mil dias, seguido pelas outras regiões com valores próximos. O gráfico inferior mostra a quantidade de dias especiais por ano, variando entre 88 e 121 dias ao longo dos últimos anos.

Figura 12: Página 7 - Análise Dias Especiais.

Fonte: Própria Autora.

A oitava página do *dashboard* (Figura 13) mostra a Carga Global Consumida no dia 28 de novembro de 2022, que coincide com o jogo da Copa do Mundo entre Brasil e Suíça. No eixo Y, temos a carga consumida ao longo do dia, enquanto o eixo X mostra as horas, começando às 3h da manhã e indo até a meia-noite. O comportamento da carga se estabelece da seguinte forma:

- **Antes do Jogo:** A carga global consumida estava aumentando constantemente desde o início da manhã. Por volta das 11:00 - 12:00, a carga atinge um pico de 81.6K MWmed, pouco antes do início do jogo, que começou às 13 horas.
- **Durante o Jogo:** Após o início do jogo há uma queda acentuada na carga, caindo para cerca de 73.3K durante o jogo, o que pode indicar uma redução no consumo de energia, em decorrência das pessoas assistindo ao jogo e menos atividades industriais ou comerciais.
- **Após o Jogo:** Por volta das 15:00, após o término do jogo, a carga diminui para 71.6K. Após isso, há uma recuperação no consumo, chegando a outro pico de 80.5K por volta das 18:00, seguido de uma leve oscilação até o fim do dia.

O gráfico ilustra bem o impacto que um grande evento como um jogo de futebol de Copa do Mundo pode ter no consumo de energia. Rampas podem ser identificadas nas quedas e subidas bruscas no gráfico. Por exemplo, a queda rápida de 81.6K para 73.3K MWmed entre 12:00 e 13:00. Essas variações abruptas no consumo de energia são

influenciadas por comportamentos sociais durante o jogo, como menos uso de energia em atividades produtivas.

Figura 13: Página 8 - Dia de Jogo de Copa do Mundo.

Fonte: Própria Autora.

A nona página do *dashboard* (Figura 14) apresenta a variação da carga global de energia elétrica consumida no Brasil no dia 24 de dezembro de 2016, véspera de Natal, com base em dados do Operador Nacional do Sistema Elétrico (ONS). Nota-se uma rampa de subida à noite, culminando no pico de 65,7 mil megawatts às 21h. Esse comportamento reflete a variação típica de consumo em dias festivos, quando a demanda aumenta durante a noite devido às celebrações natalinas.

Figura 14: Página 9 - Dia Especial - Natal.

Fonte: Própria Autora.

O gráfico interativo (*dashboard*) demonstrou ser uma ferramenta extremamente valiosa no processo de análise de dados. Ele não apenas facilitou a visualização de padrões e tendências, mas também ajudou significativamente na identificação de informações relevantes que são cruciais para a etapa de criação de modelos preditivos. Essa abordagem visual permitiu uma compreensão mais profunda dos dados, contribuindo para decisões mais informadas e estratégias mais eficazes na modelagem preditiva.

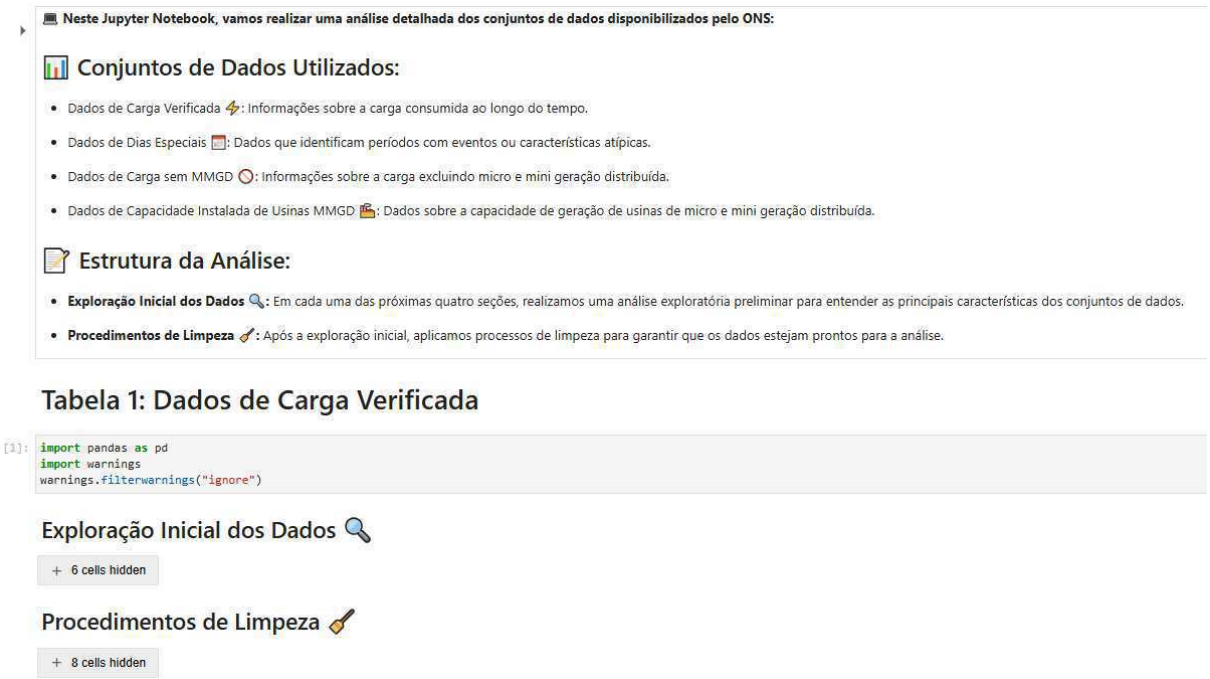
3.2 Desempenho do Modelo Darts e Previsões

No Anexo A - Parte I, são apresentados os resultados das análises realizadas nas quatro principais tabelas de dados: **Carga Verificada, Dias Especiais, Carga sem MMGD e Capacidade Instalada de Usinas MMGD.** Todas as etapas de análise foram realizadas em Python, com o auxílio da biblioteca Pandas para manipulação e tratamento dos dados.

A estrutura de análise aplicada a cada uma das tabelas segue um padrão consistente: inicialmente, realiza-se uma exploração inicial dos dados, na qual verifica-se as informações estatísticas básicas, a quantidade de valores nulos e a distribuição de classes. Essa etapa incluiu a análise de *outliers*, variáveis categóricas e a compreensão geral das características de cada tabela. Em seguida, aplicam-se procedimentos de limpeza. Utilizando a biblioteca Pandas, as colunas são transformadas para tipos de dados apropriados (como datetime e float), valores nulos e duplicados são removidos, de modo a garantir a integridade e consistência dos dados.

Essas etapas foram repetidas para cada uma das quatro tabelas, de modo que os dados estejam adequadamente preparados para análises e modelagens subsequentes. (Figura 15)

Figura 15: Análise Inicial.



Fonte: Própria Autora.

No Anexo A - Parte II, é conduzida uma análise agregada que envolve a combinação e o agrupamento de diferentes tabelas, com o objetivo de criar um conjunto de dados mais abrangente e centralizado. Essa etapa tem como finalidade integrar informações

complementares e organizar os dados para facilitar a análise posterior. A agregação é realizada especificamente nas tabelas de Dias Especiais e Carga Verificada, visando consolidar informações essenciais para o estudo.

Na tabela de Dias Especiais, o agrupamento é feito por dia especial, permitindo uma visão centralizada sobre quais regiões apresentam essa classificação em cada data específica. Esse agrupamento facilita a interpretação de eventos únicos e sua distribuição geográfica, oferecendo uma visão simplificada e integrada dos dias especiais por região.

Já na tabela de Carga Verificada, realiza-se um agrupamento por dia, onde são somados todos os valores de carga associados a cada data. Esse procedimento possibilita uma visão consolidada do consumo total diário, que é essencial para compreender a demanda energética em nível agregado e identificar possíveis padrões de consumo ao longo do tempo.

No Anexo A - Parte III, são desenvolvidos os modelos de previsão de carga elétrica. Essa seção está dividida em três partes principais: **Previsão de Carga Global por Região, Previsão de Carga para o Natal de 2024 e Previsão de Carga Global com MMGD.**

Na primeira subparte, Previsão de Carga Global por Região, o objetivo é criar um modelo específico para cada região do país, visando prever a carga elétrica. Para cada uma das quatro regiões (Sudeste/Centro-Oeste, Nordeste, Norte e Sul), o processo seguiu uma estrutura padronizada: inicialmente, realizou-se a preparação dos dados para aplicação do modelo, seguida pela criação de uma série temporal utilizando a biblioteca DARTS, que facilitou o manuseio e a organização dos dados. Em seguida, dividiu-se a série em conjuntos de treino e teste, procedendo-se ao desenvolvimento (*fitting*) do modelo Prophet, desenvolvido para cada região individualmente. A validação dos resultados foi conduzida com a métrica *Mean Absolute Percentage Error (MAPE)*, permitindo avaliar a precisão das previsões de forma quantitativa.

Os resultados dos modelos Prophet para cada região foram os seguintes:

- Sudeste/Centro-Oeste: MAPE = 13,54%
- Nordeste: MAPE = 5,55%
- Norte: MAPE = 12,51%
- Sul: MAPE = 15,88%

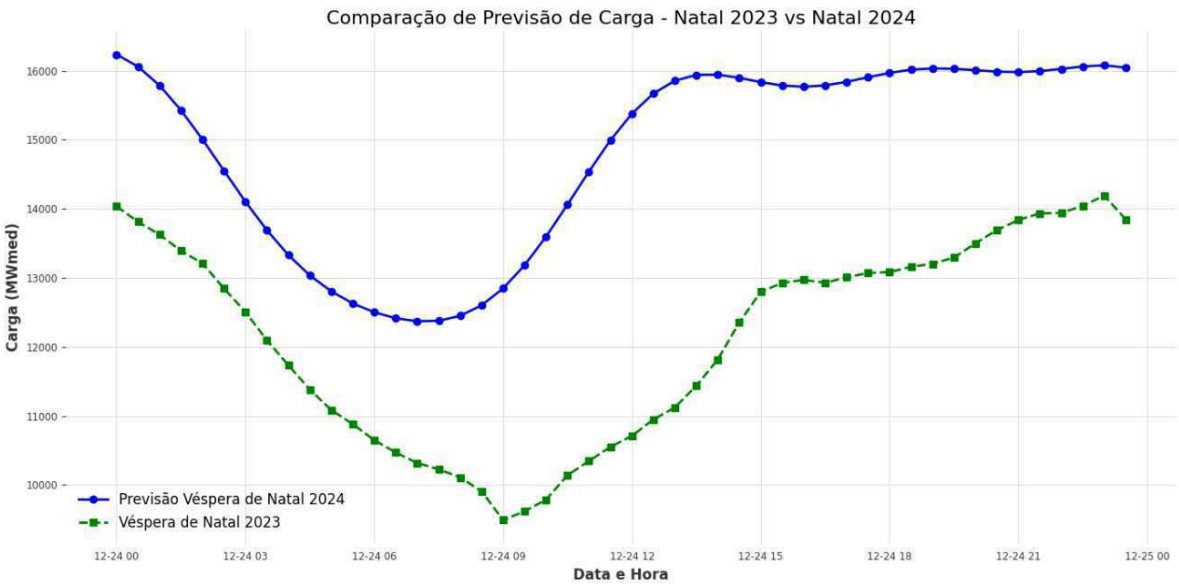
Esses valores de MAPE indicam o desempenho do modelo em cada região, com uma precisão mais elevada observada na região Nordeste, enquanto as regiões Sul e

Sudeste/Centro-Oeste apresentaram maiores desafios para previsão precisa da carga global.

Na segunda subparte, utiliza-se todo o conjunto de dados disponível para treinar um modelo mais robusto, capaz de realizar previsões para os próximos dias e horas. Entre essas previsões, destaca-se a projeção específica para o dia de Natal de 2024, com uma análise detalhada das rampas de carga. Essa análise permite avaliar flutuações significativas de demanda durante o período natalino, um momento que tradicionalmente apresenta variações notáveis de consumo.

O modelo foi configurado para realizar um forecast de 20.000 pontos no futuro, e, considerando que os dados são diários com intervalos de meia hora, essa projeção abrange as próximas 20.000 meias-horas. Esse procedimento possibilitou a previsão da carga para o dia 24 de dezembro de 2024 (véspera de Natal), conforme ilustrado na Figura 16. Nessa figura, a projeção para o Natal de 2024 é comparada com a carga registrada no Natal de 2023, permitindo validar a precisão do modelo. Observa-se uma semelhança considerável entre os padrões, incluindo a ocorrência de rampas nesse dia especial, o que sugere que o modelo é capaz de captar corretamente as variações de carga típicas do período.

Figura 16: Comparação de Previsão de Carga.



Fonte: Própria Autora.

Na última subparte, é desenvolvido o modelo de previsão da carga global com inclusão da Micro e Minigeração Distribuída (MMGD). Para iniciar esse processo, foram coletados dados de incidência solar fornecidos pelo Instituto Nacional de Pesquisas Espaciais (INPE), que disponibiliza essas informações por estado e por mês. Como os dados de carga estão divididos por regiões, foi necessário agrupar as informações de incidência solar

conforme essas regiões, além de ajustar o formato das datas para mantê-las consistentes com as outras tabelas utilizadas na análise.

Em seguida, realizou-se um merge entre a tabela de dados do INPE, após as adaptações, e a tabela de Carga Verificada, especificamente na coluna que contém os valores de carga MMGD. Com a tabela resultante, foi aplicado o modelo SARIMAX, o qual permite a utilização de variáveis exógenas – neste caso, a incidência solar – além de data e valor de carga. A avaliação do modelo resultou em uma Raiz do Erro Quadrático Médio (RMSE) de 12,6, indicando uma precisão satisfatória na previsão da carga com MMGD.

4 Conclusão e Proposta para Trabalhos Futuros

Este trabalho apresentou uma análise detalhada da aplicação de ciência de dados aos Dados Abertos do Operador Nacional do Sistema Elétrico (ONS), com foco na Micro e Minigeração Distribuída (MMGD). Utilizando uma abordagem abrangente, foram desenvolvidos modelos preditivos e técnicas de análise de dados que permitiram uma compreensão aprofundada sobre as demandas e variações de carga, especialmente em eventos que exigem flexibilidade e adaptabilidade do sistema elétrico.

A utilização de ferramentas como Python e bibliotecas específicas para manipulação de séries temporais mostrou-se essencial para o tratamento dos dados, enquanto técnicas avançadas de aprendizado de máquina, como o modelos Prophet e SARIMAX, possibilitaram incorporar variáveis exógenas como a incidência solar, melhorando a precisão das previsões. Os resultados dos modelos demonstraram que a previsão de carga pode ser realizada com boa precisão, especialmente em dias de grande variabilidade, como feriados e eventos sazonais. A previsão para o Natal de 2024, comparada com os dados de 2023, exemplificou a robustez dos modelos em identificar padrões de consumo semelhantes, inclusive nos eventos de rampa.

Além disso, os modelos desenvolvidos oferecem subsídios valiosos para a tomada de decisões estratégicas no setor elétrico, auxiliando na adaptação da geração e distribuição de energia em resposta a variações abruptas de demanda. Essa capacidade é especialmente relevante em um cenário de crescente inserção da MMGD, onde fontes intermitentes de energia podem impactar a estabilidade do sistema. Assim, a análise apresentada contribui para uma gestão energética mais sustentável e flexível, alinhada com a evolução tecnológica e as demandas do setor elétrico brasileiro.

Para aprimorar o modelo de previsão de carga e expandir sua aplicabilidade, sugere-se o uso de dados de incidência solar com maior granularidade, idealmente em frequências diárias ou horárias, para refletir melhor as oscilações reais da geração solar e seu impacto no consumo. Além disso, seria interessante adicionar novas variáveis exógenas, como dados meteorológicos e socioeconômicos, para melhorar a precisão do modelo em diferentes cenários e condições sazonais.

Outra linha promissora de investigação seria o desenvolvimento de modelos específicos para previsão da geração de MMGD, considerando variáveis que possam capturar a intermitência das fontes renováveis. Esse avanço permitiria avaliar de forma mais detalhada o papel da MMGD na matriz energética, oferecendo subsídios adicionais para uma integração mais eficiente e sustentável dessas fontes no sistema elétrico nacional.

Referências

[1] J. Desjardins. How much data is generated each day?, 2019.

[2] The Royal Swedish Academy of Sciences. The nobel prize, 2024.

[3] A. Blum, J. Hopcroft, and R. Kannan. *Foundations of Data Science*. Cambridge University Press, 2020.

[4] J. Arnold. *Learning Microsoft Power BI: Transforming Data into Insights*. O'Reilly Media, 2022.

[5] GRVPPE. Power bi: principais benefícios e aplicações, 2019.

[6] S. Raschka. *Python Machine Learning*. Packt Publishing, 2015.

[7] Gregory Piatetsky. Python leads the 11 top data science, machine learning platforms: Trends and analysis, 2019.

[8] Github. The state of open source software, 2024.

[9] Skipper Seabold Wes McKinney, Josef Perktold. Time series analysis in python with statsmodels. *Python in Science Conference*, 183:7, 2011.

[10] M. Heydt. *Learning pandas*. Packt Publishing, 2017.

[11] E. Bressert. *SciPy and NumPy*. Oreilly and Associate Series. O'Reilly, 2012.

[12] Pratyush Khare. Understanding fb prophet: A time series forecasting algorithm, 2023.

[13] Julien Herzen. Darts: Time series made easy in python, 2021.

[14] Umesh Kumawat. Darts : Learn how to simply perform time series analysis and forecasting in deep neural networks!, 2024.

[15] Abbas Seifossadat. Harnessing data science to drive advancements in electrical engineering, 2024.

[16] J.C.L. Pires and Banco Nacional de Desenvolvimento Econômico e Social (Brazil). Departamento Econômico. *Desafios da reestruturação do setor elétrico brasileiro*. Textos para discussão. BNDES, Area de Planejamento, Departamento Econômico-DEPEC, 2000.

[17] ONS. Dados abertos ons, 2024.

[18] ONS. Regulamento datathons - 5ª edição, 2024.

[19] Jayme Darriba Macêdo. O setor elÉtrico que nÃo para na hora dos jogos do brasil na copa do mundo, 2018.

Anexo - A

Anexo A

Anexo A - Parte I

💻 **Neste Jupyter Notebook, vamos realizar uma análise detalhada dos conjuntos de dados disponibilizados pelo ONS:**

📊 Conjuntos de Dados Utilizados:

- Dados de Carga Verificada ⚡: Informações sobre a carga consumida ao longo do tempo.
- Dados de Dias Especiais 📅: Dados que identificam períodos com eventos ou características atípicas.
- Dados de Carga sem MMGD 🚫: Informações sobre a carga excluindo micro e mini geração distribuída.
- Dados de Capacidade Instalada de Usinas MMGD 🏭: Dados sobre a capacidade de geração de usinas de micro e mini geração distribuída.

📝 Estrutura da Análise:

- **Exploração Inicial dos Dados 🔍:** Em cada uma das próximas quatro seções, realizamos uma análise exploratória preliminar para entender as principais características dos conjuntos de dados.
- **Procedimentos de Limpeza 🧹:** Após a exploração inicial, aplicamos processos de limpeza para garantir que os dados estejam prontos para a análise.

Tabela 1: Dados de Carga Verificada

In [1]:
```
import pandas as pd
import warnings
warnings.filterwarnings("ignore")
```

Exploração Inicial dos Dados 🔍

In [2]:
```
dados_carga_verificada = pd.read_csv("dados_carga_verificada.csv", sep = ";", low_memo
dados_carga_verificada
```

Out[2]:

	cod_areacarga	dat_referencia	din_referenciautc	val_cargaglobalcons	val_consistencia	val_car
0	SECO	2016-01-01	2016-01-01T02:30:00Z	32604,021	0	326
1	SECO	2016-01-01	2016-01-01T03:00:00Z	32236,307	0	322
2	SECO	2016-01-01	2016-01-01T03:30:00Z	32312,062	0	323
3	SECO	2016-01-01	2016-01-01T04:00:00Z	32206,387	0	322
4	SECO	2016-01-01	2016-01-01T04:30:00Z	31973,025	0	319
...	...	...	...	...	...	
561027	N	2023-12-31	2024-01-01T01:00:00Z	6974,078	-5,3061523	693
561028	N	2023-12-31	2024-01-01T01:30:00Z	6861,34	8,870605	680
561029	N	2023-12-31	2024-01-01T02:00:00Z	6760,586	-3,9951172	671
561030	N	2023-12-31	2024-01-01T02:30:00Z	6677,354	-0,49316406	662
561031	N	2023-12-31	2024-01-01T03:00:00Z	6606,427	0,37646484	655

561032 rows × 11 columns

In [3]:
```
dados_carga_verificada.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 561032 entries, 0 to 561031
Data columns (total 11 columns):
 #   Column                Non-Null Count   Dtype 
---  ------                --------------   ----- 
 0   cod_areacarga         561032 non-null  object
 1   dat_referencia        561032 non-null  object
 2   din_referenciautc     561032 non-null  object
 3   val_cargaglobalcons   561032 non-null  object
 4   val_consistencia      561032 non-null  object
 5   val_cargasup          561032 non-null  object
 6   val_cargansup         561032 non-null  object
 7   val_cargaglobal       561032 non-null  object
 8   val_cargaglobalsmmgd  561032 non-null  object
 9   val_cargarvd          561032 non-null  int64 
 10  val_cargammgd         561032 non-null  object
dtypes: int64(1), object(10)
memory usage: 47.1+ MB
```

In [4]:
```
dados_carga_verificada.isnull().sum()
```

Out[4]:
```
cod_areacarga           0
dat_referencia          0
din_referenciautc       0
val_cargaglobalcons     0
val_consistencia        0
val_cargasup            0
val_cargansup           0
val_cargaglobal         0
val_cargaglobalsmmgd    0
val_cargarvd            0
val_cargammgd           0
dtype: int64
```

In [5]:
```
dados_carga_verificada.describe()
```

Out[5]:

	val_cargarvd
count	561032.0
mean	0.0
std	0.0
min	0.0
25%	0.0
50%	0.0
75%	0.0
max	0.0

In [6]:
```
dados_carga_verificada['cod_areacarga'].value_counts()
```

Out[6]:
```
SECO    140258
S       140258
NE      140258
N       140258
Name: cod_areacarga, dtype: int64
```

In [7]:
```
pd.DataFrame(dados_carga_verificada['dat_referencia'].value_counts()).reset_index().sc
```

Out[7]:

	index	dat_referencia
24	2016-01-01	192
2191	2016-01-02	192
1945	2016-01-03	192
1946	2016-01-04	192
1947	2016-01-05	192
...	...	...
979	2023-12-27	192
980	2023-12-28	192
981	2023-12-29	192
982	2023-12-30	192
2918	2023-12-31	192

2922 rows × 2 columns

Procedimentos de Limpeza 🧹

In [8]:
```
dados_carga_verificada['dat_referencia'] = pd.to_datetime(dados_carga_verificada['dat_
dados_carga_verificada['din_referenciautc'] = pd.to_datetime(dados_carga_verificada['d
```

In [9]:
```
dados_carga_verificada.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 561032 entries, 0 to 561031
Data columns (total 11 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   cod_areacarga         561032 non-null  object
 1   dat_referencia        561032 non-null  datetime64[ns]
 2   din_referenciautc     561032 non-null  datetime64[ns, UTC]
 3   val_cargaglobalcons   561032 non-null  object
 4   val_consistencia      561032 non-null  object
 5   val_cargasup          561032 non-null  object
 6   val_cargansup         561032 non-null  object
 7   val_cargaglobal       561032 non-null  object
 8   val_cargaglobalsmmgd  561032 non-null  object
 9   val_cargarvd          561032 non-null  int64
 10  val_cargammgd         561032 non-null  object
dtypes: datetime64[ns, UTC](1), datetime64[ns](1), int64(1), object(8)
memory usage: 47.1+ MB
```

In [10]:
```
dados_carga_verificada['val_cargaglobalcons'] = dados_carga_verificada['val_cargaglob
dados_carga_verificada['val_cargasup'] = dados_carga_verificada['val_cargasup'].str.re
dados_carga_verificada['val_cargaglobal'] = dados_carga_verificada['val_cargaglobal'].
dados_carga_verificada['val_cargaglobalsmmgd'] = dados_carga_verificada['val_cargaglob
dados_carga_verificada['val_cargammgd'] = dados_carga_verificada['val_cargammgd'].str.
```

In [11]:
```
dados_carga_verificada
```

Out[11]:

	cod_areacarga	dat_referencia	din_referenciautc	val_cargaglobalcons	val_consistencia	val_car
0	SECO	2016-01-01	2016-01-01 02:30:00+00:00	32604.021	0	3260
1	SECO	2016-01-01	2016-01-01 03:00:00+00:00	32236.307	0	3223
2	SECO	2016-01-01	2016-01-01 03:30:00+00:00	32312.062	0	3231
3	SECO	2016-01-01	2016-01-01 04:00:00+00:00	32206.387	0	3220
4	SECO	2016-01-01	2016-01-01 04:30:00+00:00	31973.025	0	3197
...	...	...	...	...	...	
561027	N	2023-12-31	2024-01-01 01:00:00+00:00	6974.078	-5,3061523	693
561028	N	2023-12-31	2024-01-01 01:30:00+00:00	6861.340	8,870605	680
561029	N	2023-12-31	2024-01-01 02:00:00+00:00	6760.586	-3,9951172	671
561030	N	2023-12-31	2024-01-01 02:30:00+00:00	6677.354	-0,49316406	662
561031	N	2023-12-31	2024-01-01 03:00:00+00:00	6606.427	0,37646484	655

561032 rows × 11 columns

In [12]:
```
dados_carga_verificada.isnull().sum()
```

Out[12]:
```
cod_areacarga           0
dat_referencia          0
din_referenciautc       0
val_cargaglobalcons     0
val_consistencia        0
val_cargasup            0
val_cargansup           0
val_cargaglobal         0
val_cargaglobalsmmgd    0
val_cargarvd            0
val_cargammgd           0
dtype: int64
```

In [13]:
```
dados_carga_verificada['val_consistencia'] = dados_carga_verificada['val_consistencia'
dados_carga_verificada['val_cargansup'] = dados_carga_verificada['val_cargansup'].str.
```

In [14]:
```
dados_carga_verificada.isnull().sum()
```

Out[14]:
```
cod_areacarga           0
dat_referencia          0
din_referenciautc       0
val_cargaglobalcons     0
val_consistencia        0
val_cargasup            0
val_cargansup           0
val_cargaglobal         0
val_cargaglobalsmmgd    0
val_cargarvd            0
val_cargammgd           0
dtype: int64
```

In [15]:
```
dados_carga_verificada.to_csv("dados_carga_verificada_limpo.csv")
```

Tabela 2: Dados de Dias Especiais

Exploração Inicial dos Dados 🔍

In [16]:
```
dados_dias_especiais = pd.read_csv("dados_diaespecial.csv", sep = ";")
dados_dias_especiais
```

Out[16]:

	id_diaespecial	cod_areacarga	dat_diaespecial	nom_diaespecial	dsc_tipodiaespecial	id_tipodiaesp
0	453	SECO	2018-06-17	DOM Jogo da Copa Br x Suíça	Jogos da Copa	
1	454	SECO	2018-06-22	SEX Jogo da Copa Br x Costa Rica	Jogos da Copa	
2	455	SECO	2018-06-27	QUA Jogo da Copa Br x Sérvia	Jogos da Copa	
3	457	SECO	2017-12-29	SEX DNC	Dias não considerados	
4	464	SECO	2018-07-02	SEG - Brasil x Mexico	Jogos da Copa	
...	...	...	...	...	...	
811	1005	N	2022-04-14	QUI Quinta feira Santa	Véspera de feriados (Redução a partir das 16hs)	
812	894	N	2022-04-15	SEX Sexta feira santa	Feriado (perfil domingo)	
813	895	N	2022-04-17	DOM Pascoa	Feriado (perfil domingo)	
814	896	N	2022-06-16	QUI Corpus Christi	Feriado (perfil sábado)	
815	897	N	2022-06-17	SEX Sexta Pos Corpus Christi	Semi-feriado nacional	

816 rows × 8 columns

In [17]:
```
dados_dias_especiais.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 816 entries, 0 to 815
Data columns (total 8 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   id_diaespecial      816 non-null    int64 
 1   cod_areacarga       816 non-null    object
 2   dat_diaespecial     816 non-null    object
 3   nom_diaespecial     816 non-null    object
 4   dsc_tipodiaespecial 816 non-null    object
 5   id_tipodiaespecial  816 non-null    int64 
 6   cod_prevcarga       816 non-null    int64 
 7   cod_simplificado    816 non-null    int64 
dtypes: int64(4), object(4)
memory usage: 51.1+ KB
```

In [18]:
```
dados_dias_especiais.describe()
```

Out[18]:

	id_diaespecial	id_tipodiaespecial	cod_prevcarga	cod_simplificado
count	816.000000	816.000000	816.000000	816.000000
mean	658.860294	12.397059	5.928922	0.990196
std	202.404994	6.106152	3.995684	0.749578
min	223.000000	1.000000	1.000000	0.000000
25%	464.000000	7.000000	1.000000	0.000000
50%	694.000000	14.000000	6.000000	1.000000
75%	840.000000	19.000000	10.000000	2.000000
max	1172.000000	19.000000	12.000000	2.000000

In [19]:
```
dados_dias_especiais['dsc_tipodiaespecial'].value_counts()
```

Out[19]:
```
Feriado (perfil sábado)                            210
Semi-feriado nacional                              101
Após feriado e dia especial (perfil de segunda)     93
Feriado (perfil domingo)                            68
Véspera de feriados (Redução a partir das 16hs)     60
Natal e Ano Novo                                    48
Dia Após Natal e Ano Novo                           48
Véspera de Natal e de Ano Novo                      44
Segunda de carnaval                                 32
Terça de carnaval                                   32
Quarta feira de cinzas                              32
Semi-feriado estadual                               25
Jogos da Copa                                       22
Dias não considerados                                1
Name: dsc_tipodiaespecial, dtype: int64
```

Procedimentos de Limpeza 🧹

In [20]:
```
dados_dias_especiais['dat_diaespecial'] = pd.to_datetime(dados_dias_especiais['dat_dia
```

In [21]:
```
dados_dias_especiais.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 816 entries, 0 to 815
Data columns (total 8 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   id_diaespecial       816 non-null    int64
 1   cod_areacarga        816 non-null    object
 2   dat_diaespecial      816 non-null    datetime64[ns]
 3   nom_diaespecial      816 non-null    object
 4   dsc_tipodiaespecial  816 non-null    object
 5   id_tipodiaespecial   816 non-null    int64
 6   cod_prevcarga        816 non-null    int64
 7   cod_simplificado     816 non-null    int64
dtypes: datetime64[ns](1), int64(4), object(3)
memory usage: 51.1+ KB
```

In [22]:
```
dados_dias_especiais.to_csv("dados_dias_especiais_limpo.csv")
```

Tabela 3: Dados de Carga Sem MMGD

Exploração Inicial dos Dados 🔍

In [23]:
```
dados_carga_smm = pd.read_csv("dados_carga_mm.csv", sep=';').sample(10000)
dados_carga_smm
```

Out[23]:

	Instante	Valor	Area
748173	2018-05-12T07:17:00Z	9555,13943684896	NE
1073901	2018-07-07T12:05:00Z	4696,92205403646	N
7520203	2021-07-31T23:31:00Z	12269,1336914063	S
10640391	2023-01-22T18:43:00Z	5500,13337402344	N
10271283	2022-11-19T10:55:00Z	5914,07936197917	N
...	...	...	...
6281143	2020-12-27T12:31:00Z	4839,68666992188	N
12157839	2023-10-14T13:34:00Z	33566,7533203125	SECO
2538500	2019-03-19T14:04:00Z	47186,3171223958	SECO
3000052	2019-06-08T02:36:00Z	33287,6505208333	SECO
6485086	2021-02-01T03:34:00Z	10891,0092773438	NE

10000 rows × 3 columns

In [24]:
```
dados_carga_smm.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10000 entries, 748173 to 6485086
Data columns (total 3 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   Instante  10000 non-null  object
 1   Valor     10000 non-null  object
 2   Area      10000 non-null  object
dtypes: object(3)
memory usage: 312.5+ KB
```

Procedimentos de Limpeza 🧹

In [25]:
```
dados_carga_smm['Instante'] = pd.to_datetime(dados_carga_smm['Instante'], format='%Y-%
```

In [26]:
```
dados_carga_smm['Valor'] = dados_carga_smm['Valor'].str.replace(',', '.').astype(float
```

In [27]:
```
dados_carga_smm.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10000 entries, 748173 to 6485086
Data columns (total 3 columns):
 #   Column    Non-Null Count  Dtype         
---  ------    --------------  -----         
 0   Instante  10000 non-null  datetime64[ns]
 1   Valor     10000 non-null  float64       
 2   Area      10000 non-null  object        
dtypes: datetime64[ns](1), float64(1), object(1)
memory usage: 312.5+ KB
```

In [28]:
```
dados_carga_smm.isnull().sum()
```

Out[28]:
```
Instante    0
Valor       0
Area        0
dtype: int64
```

In [29]:
```
dados_carga_smm.describe()
```

Out[29]:

	Valor
count	10000.000000
mean	16656.341699
std	13069.596442
min	3910.866081
25%	7479.837012
50%	11123.006706
75%	25935.084668
max	56717.914258

In [30]:
```
dados_carga_smm['Area'].value_counts()
```

Out[30]:
```
N       2543
SECO    2530
S       2474
NE      2453
Name: Area, dtype: int64
```

In [31]:
```
dados_carga_smm.to_csv("dados_carga_smm_limpo.csv")
```

Tabela 4: Dados de Capacidade Instalada de Usinas MMGD

Exploração Inicial dos Dados 🔍

In [32]:
```
capacidade_instalada = pd.read_csv("capacidade_instalada.csv", sep=';')
capacidade_instalada
```

Out[32]:

	cod_areacarga	sgl_tpgeracaousianeel	dat_referencia	val_potenciainstalada	val_potacum
0	N	CGH	2009-06-20	0	0
1	N	CGH	2009-06-21	0	0
2	N	CGH	2009-06-22	0	0
3	N	CGH	2009-06-23	0	0
4	N	CGH	2009-06-24	0	0
...	...	...	...	...	...
85659	SECO	UTE	2024-02-11	0	112142,399913847
85660	SECO	UTE	2024-02-12	0	112142,399913847
85661	SECO	UTE	2024-02-13	0	112142,399913847
85662	SECO	UTE	2024-02-14	0	112142,399913847
85663	SECO	UTE	2024-02-15	0	112142,399913847

85664 rows × 5 columns

In [33]:
```
capacidade_instalada.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 85664 entries, 0 to 85663
Data columns (total 5 columns):
 #   Column                Non-Null Count  Dtype 
---  ------                --------------  ----- 
 0   cod_areacarga         85664 non-null  object
 1   sgl_tpgeracaousianeel 85664 non-null  object
 2   dat_referencia        85664 non-null  object
 3   val_potenciainstalada 85664 non-null  object
 4   val_potacum           85664 non-null  object
dtypes: object(5)
memory usage: 3.3+ MB
```

In [34]:
```
capacidade_instalada.describe()
```

Out[34]:

	cod_areacarga	sgl_tpgeracaousianeel	dat_referencia	val_potenciainstalada	val_potacum
count	85664	85664	85664	85664	85664
unique	4	4	5354	10471	12158
top	N	CGH	2009-06-20	0	0
freq	21416	21416	16	73505	33153

In [35]:
```
capacidade_instalada['cod_areacarga'].value_counts()
```

Out[35]:
```
N       21416
NE      21416
S       21416
SECO    21416
Name: cod_areacarga, dtype: int64
```

In [36]:
```
capacidade_instalada['sgl_tpgeracaousianeel'].value_counts()
```

Out[36]:
```
CGH    21416
EOL    21416
UFV    21416
UTE    21416
Name: sgl_tpgeracaousianeel, dtype: int64
```

Procedimentos de Limpeza 🧹

In [37]:
```
capacidade_instalada['dat_referencia'] = pd.to_datetime(capacidade_instalada['dat_refe
```

In [38]:
```
capacidade_instalada['val_potenciainstalada'] = capacidade_instalada['val_potenciainst
capacidade_instalada['val_potacum'] = capacidade_instalada['val_potacum'].str.replace(
```

In [39]:
```
capacidade_instalada.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 85664 entries, 0 to 85663
Data columns (total 5 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   cod_areacarga          85664 non-null  object
 1   sgl_tpgeracaousianeel  85664 non-null  object
 2   dat_referencia         85664 non-null  datetime64[ns]
 3   val_potenciainstalada  85664 non-null  float64
 4   val_potacum            85664 non-null  float64
dtypes: datetime64[ns](1), float64(2), object(2)
memory usage: 3.3+ MB
```

In [40]:
```
capacidade_instalada.to_csv("capacidade_instalada_limpo.csv")
```

Anexo A - Parte II

Análise Agrupada

🔜 **Nesta seção, realizaremos uma análise agrupada,** onde faremos um merge entre as tabelas para consolidar as informações. O foco será agrupar os dados relacionados aos dias especiais, criando um conjunto de dados mais coeso, que poderá ser utilizado em futuros modelos preditivos.

In [41]:
```
dados_carga_verificada_limpo = pd.read_csv("dados_carga_verificada_limpo.csv")
dados_carga_verificada_limpo.head()
```

Out[41]:

	Unnamed: 0	cod_areacarga	dat_referencia	din_referenciautc	val_cargaglobalcons	val_consistencia	v
0	0	SECO	2016-01-01	2016-01-01 02:30:00+00:00	32604.021	0.0	
1	1	SECO	2016-01-01	2016-01-01 03:00:00+00:00	32236.307	0.0	
2	2	SECO	2016-01-01	2016-01-01 03:30:00+00:00	32312.062	0.0	
3	3	SECO	2016-01-01	2016-01-01 04:00:00+00:00	32206.387	0.0	
4	4	SECO	2016-01-01	2016-01-01 04:30:00+00:00	31973.025	0.0	

In [42]:
```
dados_dias_especiais_limpo = pd.read_csv("dados_dias_especiais_limpo.csv")
dados_dias_especiais_limpo.head()
```

Out[42]:

	Unnamed: 0	id_diaespecial	cod_areacarga	dat_diaespecial	nom_diaespecial	dsc_tipodiaespecial	id_t
0	0	453	SECO	2018-06-17	DOM Jogo da Copa Br x Suíça	Jogos da Copa	
1	1	454	SECO	2018-06-22	SEX Jogo da Copa Br x Costa Rica	Jogos da Copa	
2	2	455	SECO	2018-06-27	QUA Jogo da Copa Br x Sérvia	Jogos da Copa	
3	3	457	SECO	2017-12-29	SEX DNC	Dias não considerados	
4	4	464	SECO	2018-07-02	SEG - Brasil x Mexico	Jogos da Copa	

In [43]:
```
dados_carga_smm_limpo = pd.read_csv("dados_carga_smm_limpo.csv")
dados_carga_smm_limpo.head()
```

Out[43]:

	Unnamed: 0	Instante	Valor	Area
0	748173	2018-05-12 07:17:00	9555.139437	NE
1	1073901	2018-07-07 12:05:00	4696.922054	N
2	7520203	2021-07-31 23:31:00	12269.133691	S
3	10640391	2023-01-22 18:43:00	5500.133374	N
4	10271283	2022-11-19 10:55:00	5914.079362	N

In [44]:
```
capacidade_instalada_limpo = pd.read_csv("capacidade_instalada_limpo.csv")
capacidade_instalada_limpo.head()
```

Out[44]:

	Unnamed: 0	cod_areacarga	sgl_tpgeracaousianeel	dat_referencia	val_potenciainstalada	val_potacum
0	0	N	CGH	2009-06-20	0.0	0.0
1	1	N	CGH	2009-06-21	0.0	0.0
2	2	N	CGH	2009-06-22	0.0	0.0
3	3	N	CGH	2009-06-23	0.0	0.0
4	4	N	CGH	2009-06-24	0.0	0.0

In [45]:

```
dias_especiais_agg = dados_dias_especiais_limpo.groupby(['dat_diaespecial']).agg({
    'cod_areacarga': list,
    'dsc_tipodiaespecial': list,
    'nom_diaespecial': list,
    'id_tipodiaespecial': list
}).reset_index()

dias_especiais_agg
```

Out[45]:

	dat_diaespecial	cod_areacarga	dsc_tipodiaespecial	nom_diaespecial	id_tipodiaespecial
0	2016-01-01	[SECO, S, NE, N]	[Natal e Ano Novo, Natal e Ano Novo, Natal e A...	[SEX Ano Novo, SEX Ano Novo, SEX Ano Novo, SEX...	[13, 13, 13, 13]
1	2016-01-02	[SECO, S, NE, N]	[Dia Após Natal e Ano Novo, Dia Após Natal e A...	[SAB Pós Ano Novo, SAB Pós Ano Novo, SAB Pós A...	[14, 14, 14, 14]
2	2016-01-20	[SECO]	[Semi-feriado estadual]	[QUA Dia de São Sebastião]	[1]
3	2016-02-05	[SECO, S, NE, N]	[Véspera de feriados (Redução a partir das 16h...	[SEX Sexta de Carnaval, SEX Sexta de Carnaval,...	[6, 6, 6, 6]
4	2016-02-08	[SECO, S, NE, N]	[Segunda de carnaval, Segunda de carnaval, Seg...	[SEG Carnaval, SEG Carnaval, SEG Carnaval, SEG...	[15, 15, 15, 15]
...	...	...	...	...	...
225	2023-11-20	[SECO]	[Semi-feriado estadual]	[SEG Dia da Consciência Negra]	[1]
226	2023-12-08	[NE]	[Semi-feriado estadual]	[SEX Dia de Nossa Senhora da Conceição]	[1]
227	2023-12-24	[SECO, S, NE, N]	[Véspera de Natal e de Ano Novo, Véspera de Na...	[DOM Dia Anterior Natal, DOM Dia Anterior Nata...	[12, 12, 12, 12]
228	2023-12-25	[SECO, S, NE, N]	[Natal e Ano Novo, Natal e Ano Novo, Natal e A...	[SEG Natal, SEG Natal, SEG Natal, SEG Natal]	[13, 13, 13, 13]
229	2023-12-26	[SECO, S, NE, N]	[Dia Após Natal e Ano Novo, Dia Após Natal e A...	[TER Dia Apos Natal, TER Dia Apos Natal, TER D...	[14, 14, 14, 14]

230 rows × 5 columns

In [46]:
```
dados_carga_verificada_agg = dados_carga_verificada_limpo.groupby(['dat_referencia']).
    'cod_areacarga': list,
    'val_cargaglobalcons': sum,
    'val_consistencia': sum,
    'val_cargasup': sum,
    'val_cargansup': sum,
    'val_cargaglobal': sum,
    'val_cargaglobalsmmgd': sum,
    'val_cargarvd': sum,
    'val_cargammgd': sum
}).reset_index()
```

In [47]:
```
merge_final = dias_especiais_agg.merge(dados_carga_verificada_agg, how='left', left_or
```

Na tabela final resultante do merge, temos um foco específico nos **dias especiais**, com os dados **agrupados por regiões**. Além disso, todos os detalhes da **carga verificada** associados a

esses dias estão incluídos, permitindo uma análise mais segmentada e detalhada.

In [48]:
```
merge_final.head()
```

Out[48]:

	dat_diaespecial	cod_areacarga_x	dsc_tipodiaespecial	nom_diaespecial	id_tipodiaespecial	dat_refere
0	2016-01-01	[SECO, S, NE, N]	[Natal e Ano Novo, Natal e Ano Novo, Natal e A...	[SEX Ano Novo, SEX Ano Novo, SEX Ano Novo, SEX...	[13, 13, 13, 13]	2016-0
1	2016-01-02	[SECO, S, NE, N]	[Dia Após Natal e Ano Novo, Dia Após Natal e A...	[SAB Pós Ano Novo, SAB Pós Ano Novo, SAB Pós A...	[14, 14, 14, 14]	2016-0
2	2016-01-20	[SECO]	[Semi-feriado estadual]	[QUA Dia de São Sebastião]	[1]	2016-0
3	2016-02-05	[SECO, S, NE, N]	[Véspera de feriados (Redução a partir das 16h...	[SEX Sexta de Carnaval, SEX Sexta de Carnaval,...	[6, 6, 6, 6]	2016-0
4	2016-02-08	[SECO, S, NE, N]	[Segunda de carnaval, Segunda de carnaval, Seg...	[SEG Carnaval, SEG Carnaval, SEG Carnaval, SEG...	[15, 15, 15, 15]	2016-0

Anexo A - Parte III

🔮 Criação de Modelos de Previsão

A seguir, começaremos a desenvolver **modelos de predição**, divididos em três categorias principais:

1. **Previsão da Carga Global por Região 🗺️:**

- Este modelo será dividido em 4 submodelos, um para cada região, cada um focado na previsão da carga global de uma região específica. A ideia é capturar as particularidades de cada região e prever as variações de carga em períodos críticos.

1. **Previsão da Carga para o Natal de 2024 🎄:**

Utilizando os dados completos e o modelo validado, faremos uma previsão específica para o Natal de 2024, com foco nas rampas de carga que podem ocorrer nesse período, em comparação com a demanda observada no Natal de 2023. Essa análise ajudará a prever possíveis flutuações de demanda em datas comemorativas.

1. **Previsão da Carga Global com MMGD 🌞:**

- O terceiro modelo será focado na previsão da carga global com MMGD. Para aumentar a precisão, dados de incidência solar serão incorporados a este modelo.

Todos esses modelos serão desenvolvidos utilizando a biblioteca **DARTS**, com o **modelo Prophet** escolhido como o principal método de previsão.

1️⃣ Modelo de Previsão da Carga Global por Região

In [49]:
```
!pip install pandas==1.4.4 "protobuf<3.20" darts==0.30.0 prophet==1.1.5 -q

import darts
import pandas as pd
from darts.dataprocessing.transformers.scaler import Scaler
from sklearn.preprocessing import MinMaxScaler
from darts.models import AutoARIMA, Prophet, TBATS
from darts.metrics import mape
from statsmodels.tsa.statespace.sarimax import SARIMAX
import numpy as np
from sklearn.metrics import mean_squared_error

print(f"Darts version: {darts.__version__}")
```

```
WARNING: There was an error checking the latest version of pip.
Darts version: 0.30.0
```

Modelo 1.1: Região Sudeste/Centro Oeste 🟥

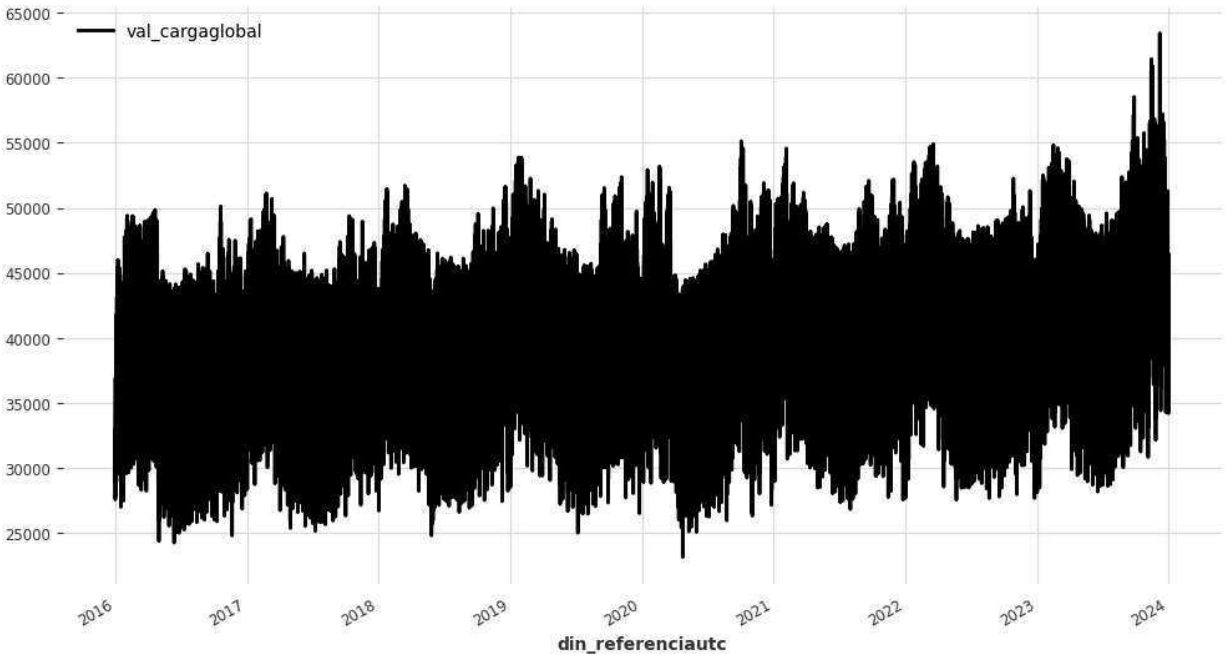
In [50]:
```
dados_carga_verificada_limpo_se = dados_carga_verificada_limpo[dados_carga_verificada_
```

In [51]:
```
dados_carga_verificada_limpo_se.info()
```

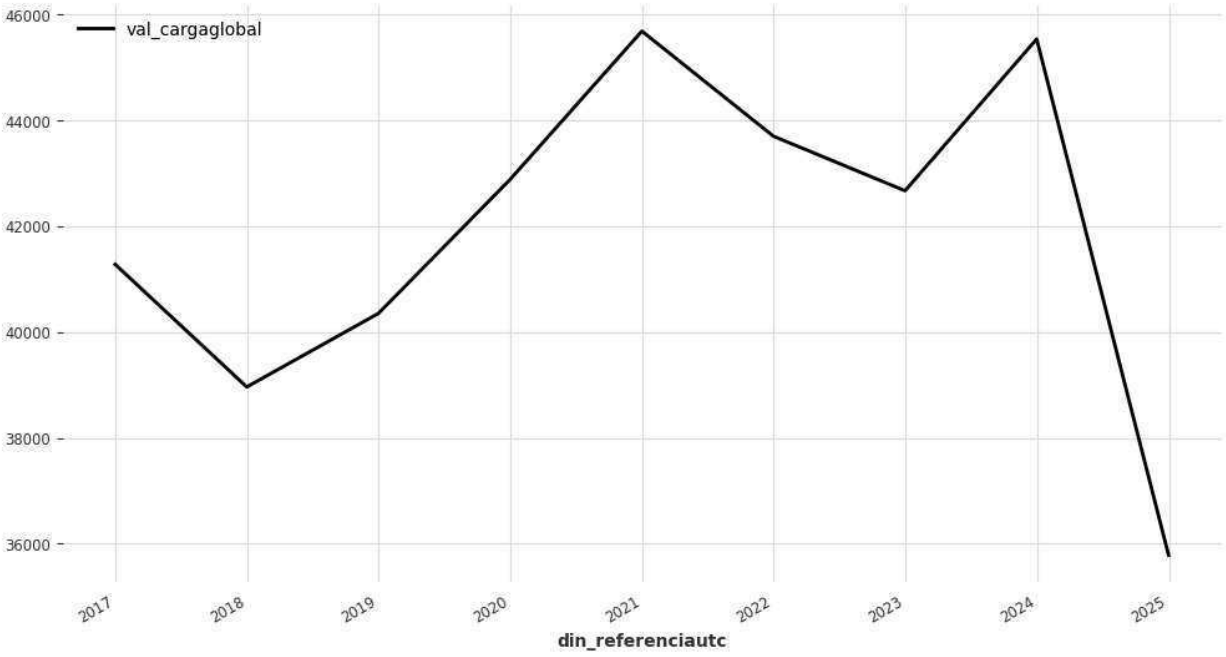
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 140258 entries, 0 to 140257
Data columns (total 12 columns):
 #   Column               Non-Null Count   Dtype  
---  ------               --------------   -----  
 0   Unnamed: 0           140258 non-null  int64  
 1   cod_areacarga        140258 non-null  object 
 2   dat_referencia       140258 non-null  object 
 3   din_referenciautc    140258 non-null  object 
 4   val_cargaglobalcons  140258 non-null  float64
 5   val_consistencia     140258 non-null  float64
 6   val_cargasup         140258 non-null  float64
 7   val_cargansup        140258 non-null  float64
 8   val_cargaglobal      140258 non-null  float64
 9   val_cargaglobalsmmgd 140258 non-null  float64
 10  val_cargarvd         140258 non-null  int64  
 11  val_cargammgd        140258 non-null  float64
dtypes: float64(7), int64(2), object(3)
memory usage: 13.9+ MB
```

In [52]:
```
dados_carga_verificada_limpo_se['din_referenciautc'] = pd.to_datetime(dados_carga_veri
```

In [53]:
```
import matplotlib.pyplot as plt
from darts import TimeSeries

# Cria a série temporal
series_seco = TimeSeries.from_dataframe(dados_carga_verificada_limpo_se, 'din_referenc

# Define o tamanho da figura
plt.figure(figsize=(12, 6))

series_seco.plot()
plt.show()
```

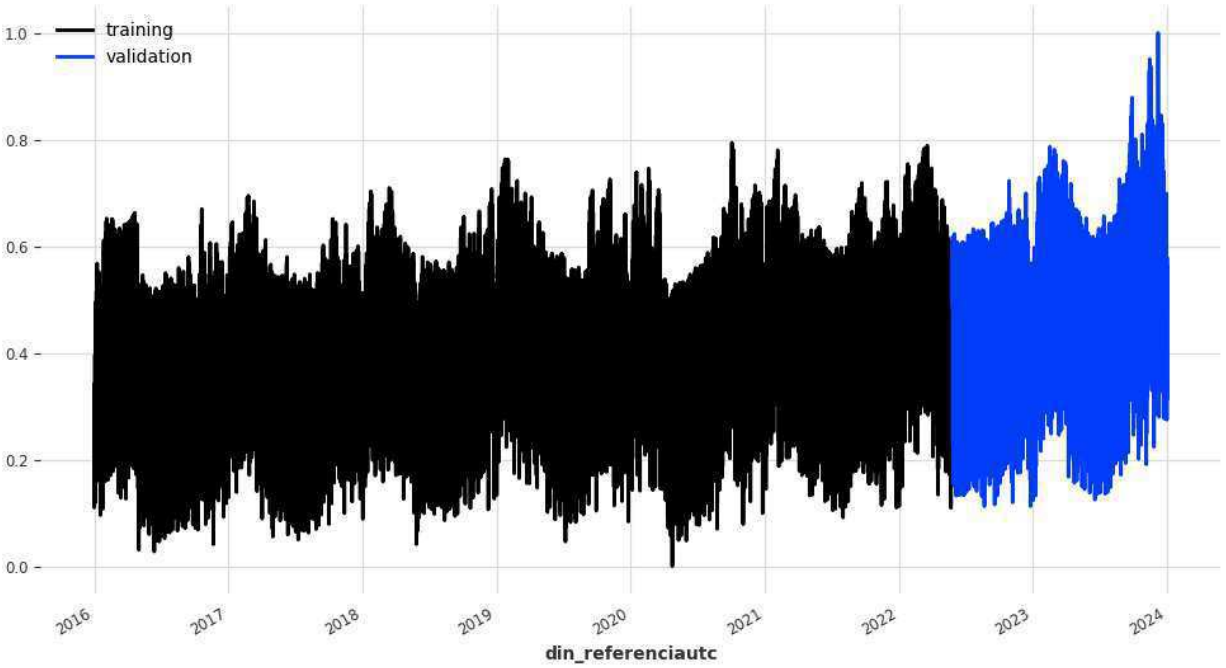


In [54]:
```
resampled_example = series_seco.resample("Y")

# Define o tamanho da figura (em polegadas)
plt.figure(figsize=(12, 6))  # Aumenta o tamanho para 12x6 polegadas

# Plota a série
resampled_example.plot()

# Mostra o gráfico
plt.show()
```



In [55]:
```
scaler = MinMaxScaler()
ts_transformer = Scaler(scaler)
scaled_ts_seco = ts_transformer.fit_transform(series_seco)
```

In [56]:
```
train, val = (scaled_ts_seco).split_before(0.8)# (we standardize by dividing by 100 si

# Define o tamanho da figura (em polegadas)
plt.figure(figsize=(12, 6))  # Aumenta o tamanho para 12x6 polegadas

# Plota a série
train.plot(label="training")
val.plot(label="validation")

# Mostra o gráfico
plt.show()
```



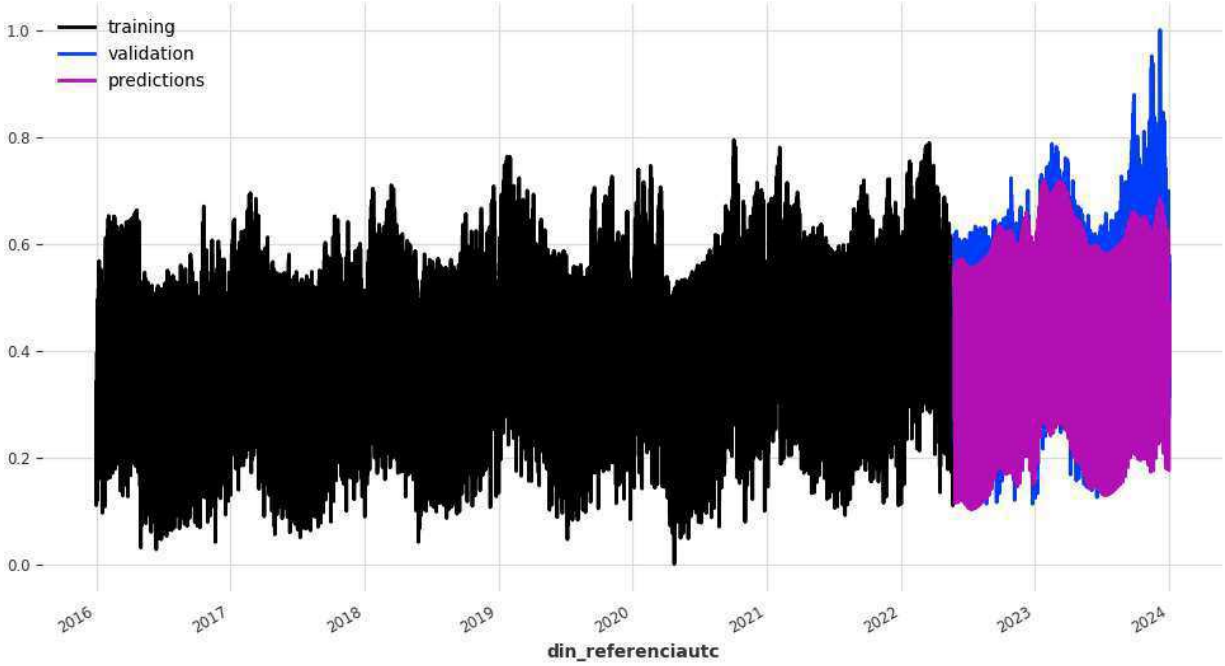
In [57]:
```
results_dict = {}

temp = Prophet()
temp.fit(train)
preds = temp.predict(len(val))
accuracy = mape(val, preds)
```

```
14:23:16 - cmdstanpy - INFO - Chain [1] start processing
14:25:47 - cmdstanpy - INFO - Chain [1] done processing
```

In [58]:
```
results_dict[f"Prophet SECO"] = accuracy

results_dict
```

Out[58]:
```
{'Prophet SECO': 13.539125846545344}
```

In [59]:
```
plt.figure(figsize=(12, 6))

train.plot(label="training")
val.plot(label="validation")
preds.plot(label="predictions")

plt.legend()
plt.show()
```



Modelo 1.2: Região Nordeste

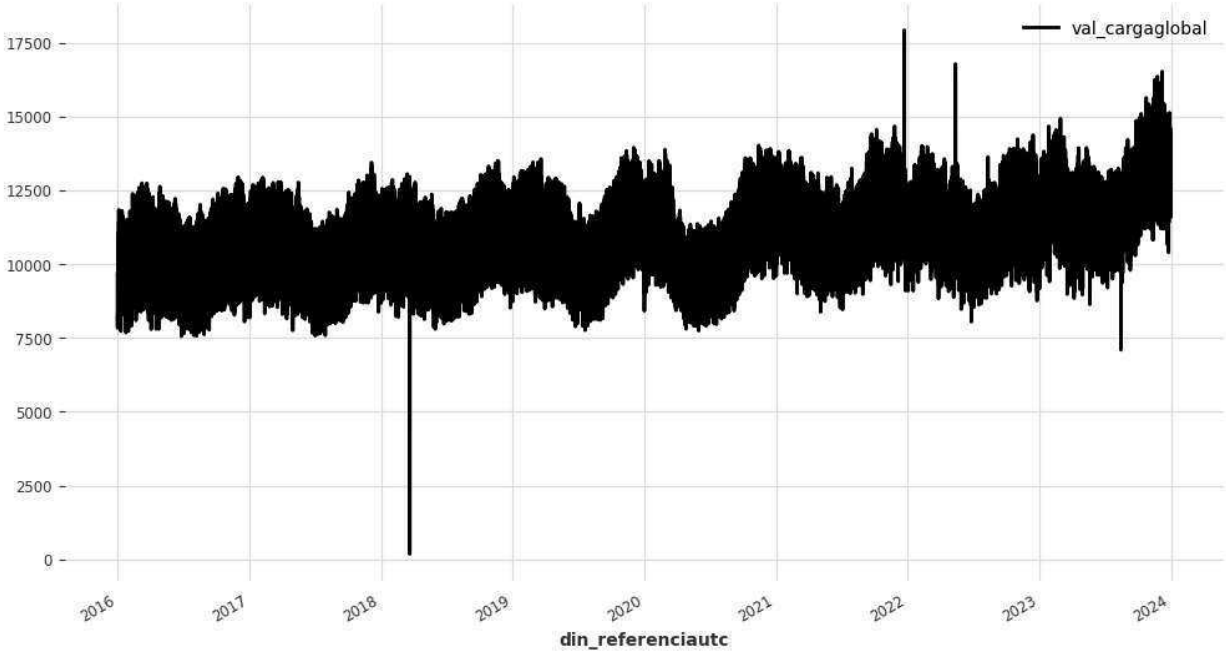
In [60]:
```
dados_carga_verificada_limpo_ne = dados_carga_verificada_limpo[dados_carga_verificada_
```

In [61]:
```
dados_carga_verificada_limpo_ne.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 140258 entries, 280516 to 420773
Data columns (total 12 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   Unnamed: 0            140258 non-null  int64
 1   cod_areacarga         140258 non-null  object
 2   dat_referencia        140258 non-null  object
 3   din_referenciautc     140258 non-null  object
 4   val_cargaglobalcons   140258 non-null  float64
 5   val_consistencia      140258 non-null  float64
 6   val_cargasup          140258 non-null  float64
 7   val_cargansup         140258 non-null  float64
 8   val_cargaglobal       140258 non-null  float64
 9   val_cargaglobalsmmgd  140258 non-null  float64
 10  val_cargarvd          140258 non-null  int64
 11  val_cargammgd         140258 non-null  float64
dtypes: float64(7), int64(2), object(3)
memory usage: 13.9+ MB
```

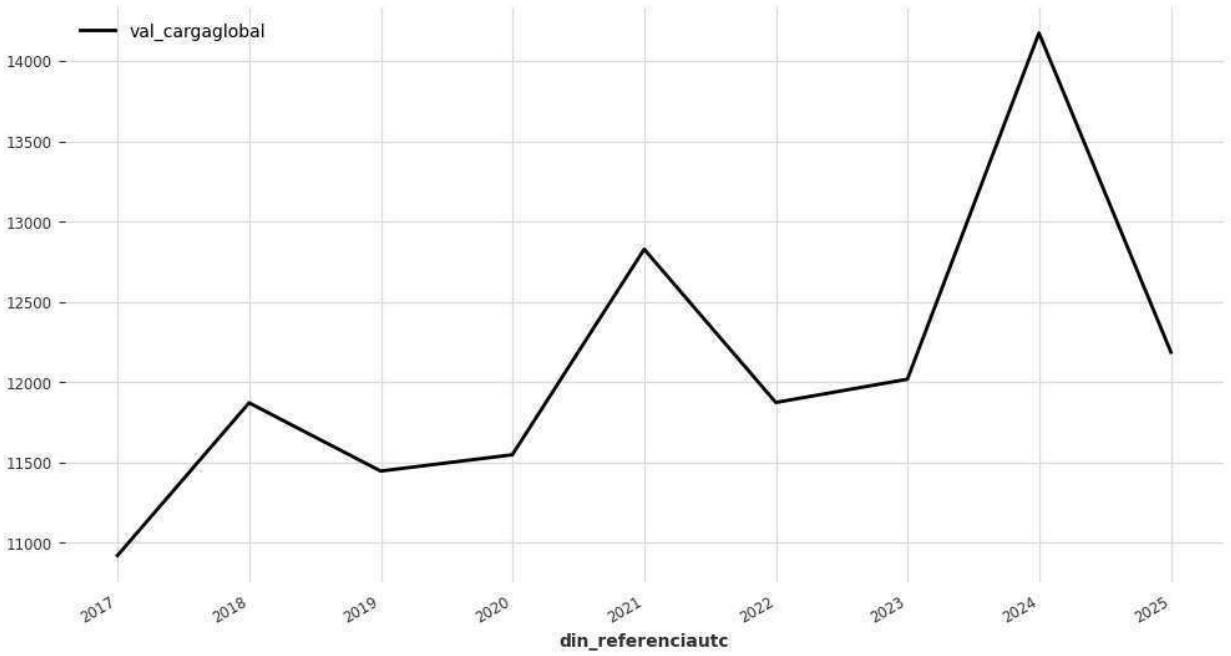
In [62]:
```
dados_carga_verificada_limpo_ne['din_referenciautc'] = pd.to_datetime(dados_carga_veri
```

In [63]:
```
import matplotlib.pyplot as plt
from darts import TimeSeries

# Cria a série temporal
series_ne = TimeSeries.from_dataframe(dados_carga_verificada_limpo_ne, 'din_referencia

# Define o tamanho da figura
plt.figure(figsize=(12, 6))

series_ne.plot()
plt.show()
```



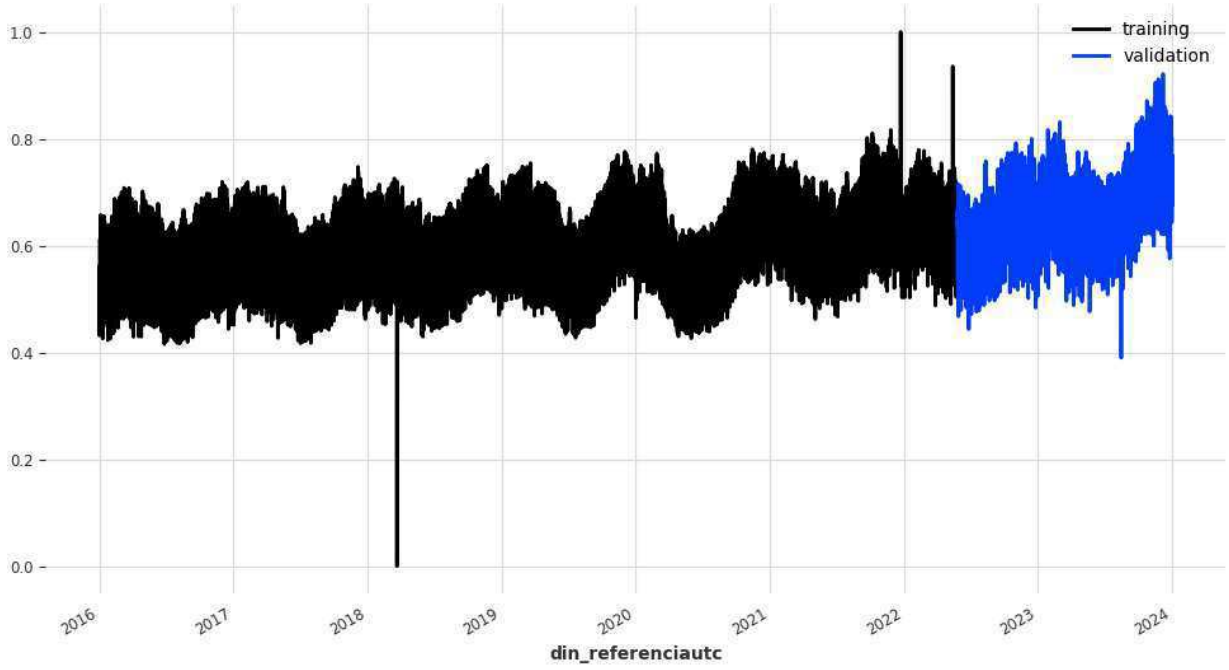
In [64]:
```
resampled_example = series_ne.resample("Y")

# Define o tamanho da figura (em polegadas)
plt.figure(figsize=(12, 6))  # Aumenta o tamanho para 12x6 polegadas
```

```
# Plota a série
resampled_example.plot()

# Mostra o gráfico
plt.show()
```



In [65]:
```
scaler = MinMaxScaler()
ts_transformer = Scaler(scaler)
scaled_ts_ne = ts_transformer.fit_transform(series_ne)
```

In [66]:
```
train, val = (scaled_ts_ne).split_before(0.8)# (we standardize by dividing by 100 sinc

# Define o tamanho da figura (em polegadas)
plt.figure(figsize=(12, 6))  # Aumenta o tamanho para 12x6 polegadas

# Plota a série
train.plot(label="training")
val.plot(label="validation")

# Mostra o gráfico
plt.show()
```



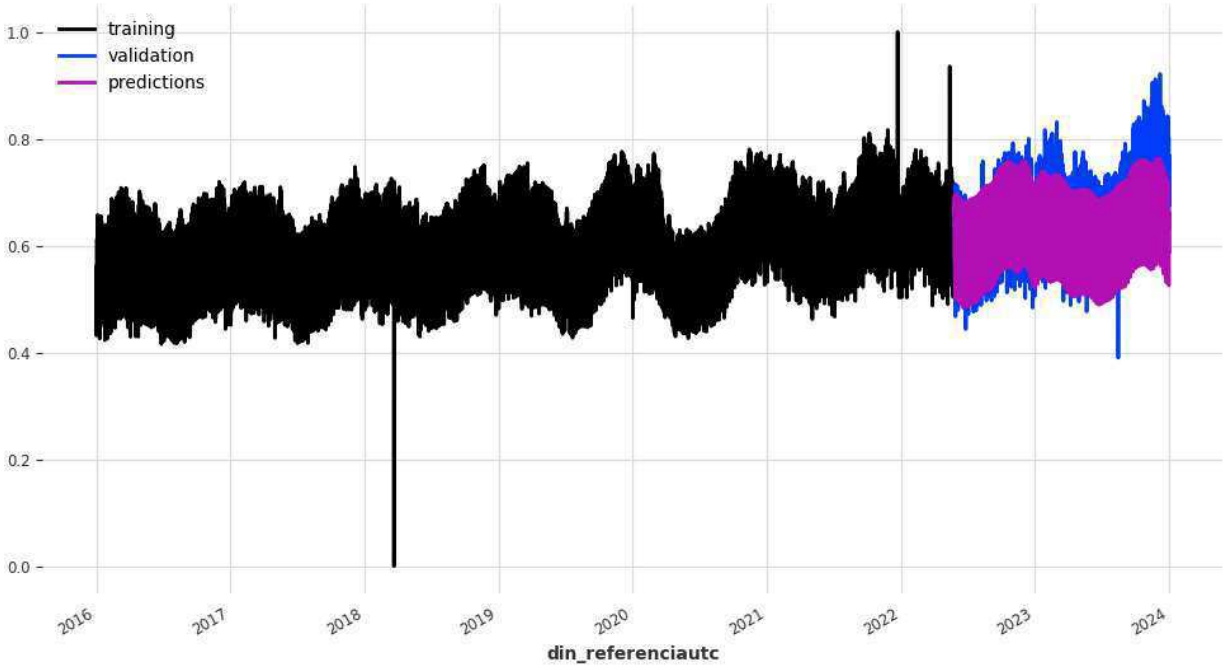
In [67]:
```
temp = Prophet()
temp.fit(train)
preds = temp.predict(len(val))
accuracy = mape(val, preds)
```

```
14:26:39 - cmdstanpy - INFO - Chain [1] start processing
14:28:22 - cmdstanpy - INFO - Chain [1] done processing
```

In [68]:
```
results_dict[f"Prophet NE"] = accuracy

results_dict
```

Out[68]:
```
{'Prophet SECO': 13.539125846545344, 'Prophet NE': 5.548868962998355}
```

In [69]:
```
plt.figure(figsize=(12, 6))

train.plot(label="training")
val.plot(label="validation")
preds.plot(label="predictions")

plt.legend()
plt.show()
```



Modelo 1.3: Região Norte

In [70]:
```
dados_carga_verificada_limpo_n = dados_carga_verificada_limpo[dados_carga_verificada_l
```

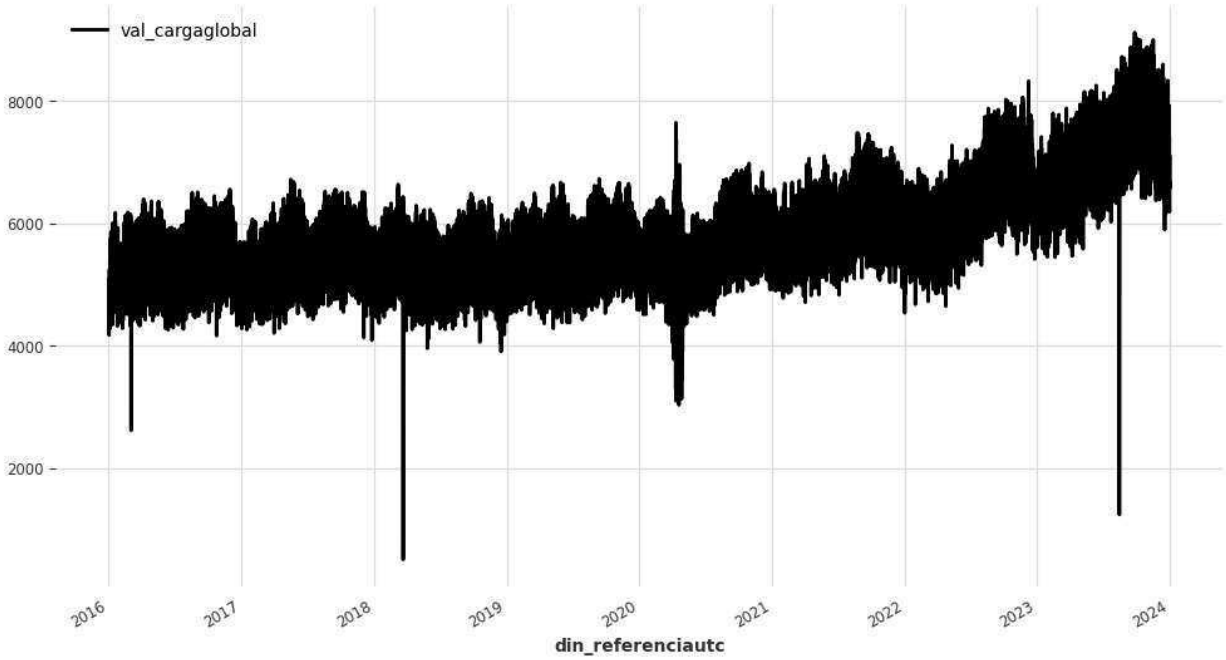
In [71]:
```
dados_carga_verificada_limpo_n.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 140258 entries, 420774 to 561031
Data columns (total 12 columns):
 #   Column                Non-Null Count   Dtype  
---  ------                --------------   -----  
 0   Unnamed: 0            140258 non-null  int64  
 1   cod_areacarga         140258 non-null  object 
 2   dat_referencia        140258 non-null  object 
 3   din_referenciautc     140258 non-null  object 
 4   val_cargaglobalcons   140258 non-null  float64
 5   val_consistencia      140258 non-null  float64
 6   val_cargasup          140258 non-null  float64
 7   val_cargansup         140258 non-null  float64
 8   val_cargaglobal       140258 non-null  float64
 9   val_cargaglobalsmmgd  140258 non-null  float64
 10  val_cargarvd          140258 non-null  int64  
 11  val_cargammgd         140258 non-null  float64
dtypes: float64(7), int64(2), object(3)
memory usage: 13.9+ MB
```

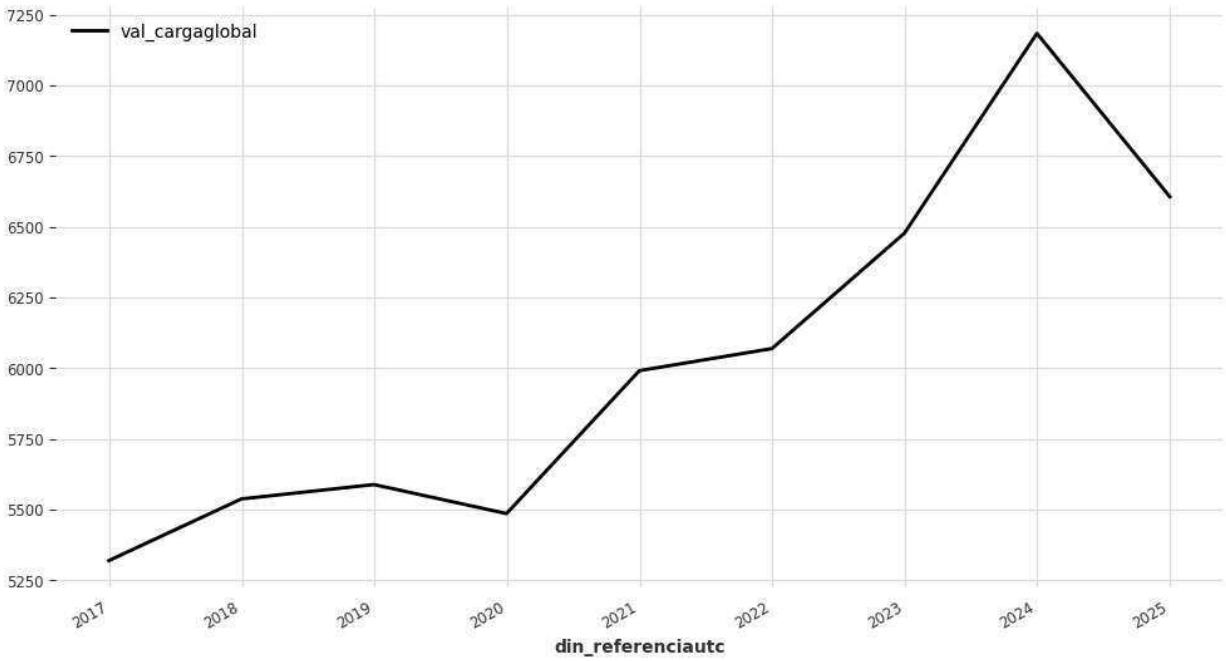
In [72]:
```
dados_carga_verificada_limpo_n['din_referenciautc'] = pd.to_datetime(dados_carga_verif
```

In [73]:
```
import matplotlib.pyplot as plt
from darts import TimeSeries

# Cria a série temporal
series_n = TimeSeries.from_dataframe(dados_carga_verificada_limpo_n, 'din_referenciaut

# Define o tamanho da figura
plt.figure(figsize=(12, 6))
```

```
series_n.plot()
plt.show()
```



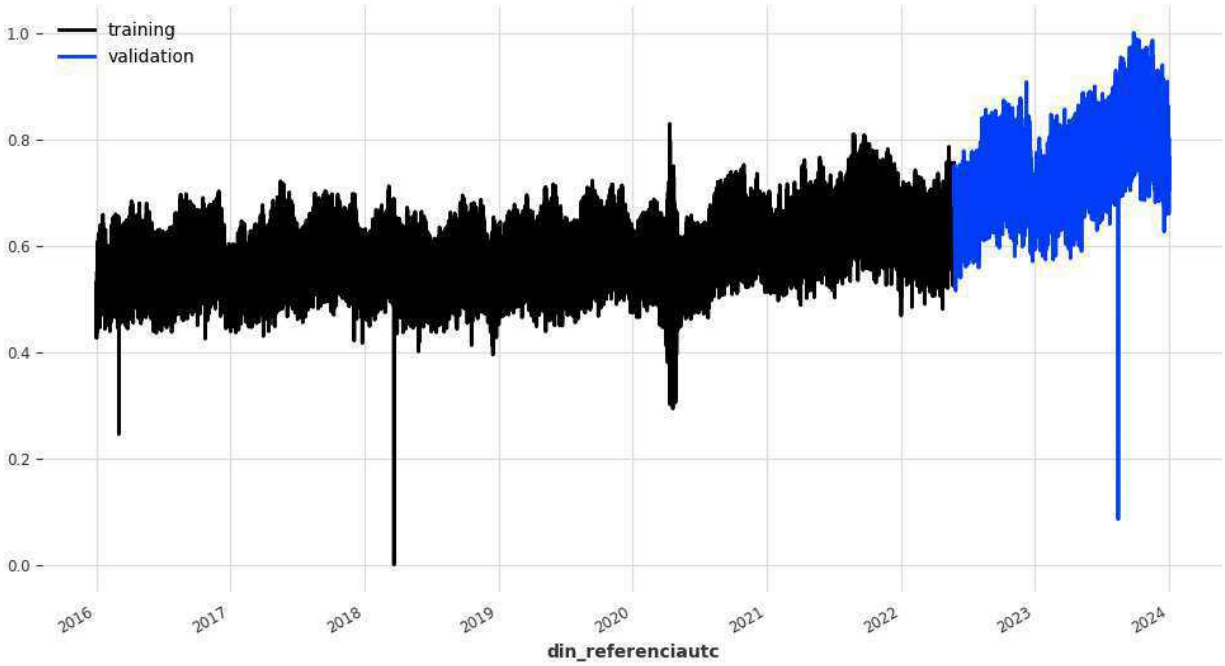
In [74]:
```
resampled_example = series_n.resample("Y")

# Define o tamanho da figura (em polegadas)
plt.figure(figsize=(12, 6))  # Aumenta o tamanho para 12x6 polegadas

# Plota a série
resampled_example.plot()

# Mostra o gráfico
plt.show()
```



In [75]:
```
scaler = MinMaxScaler()
ts_transformer = Scaler(scaler)
```

```
scaled_ts_n = ts_transformer.fit_transform(series_n)
```

In [76]:
```
train, val = (scaled_ts_n).split_before(0.8)# (we standardize by dividing by 100 since

# Define o tamanho da figura (em polegadas)
plt.figure(figsize=(12, 6))  # Aumenta o tamanho para 12x6 polegadas

# Plota a série
train.plot(label="training")
val.plot(label="validation")

# Mostra o gráfico
plt.show()
```



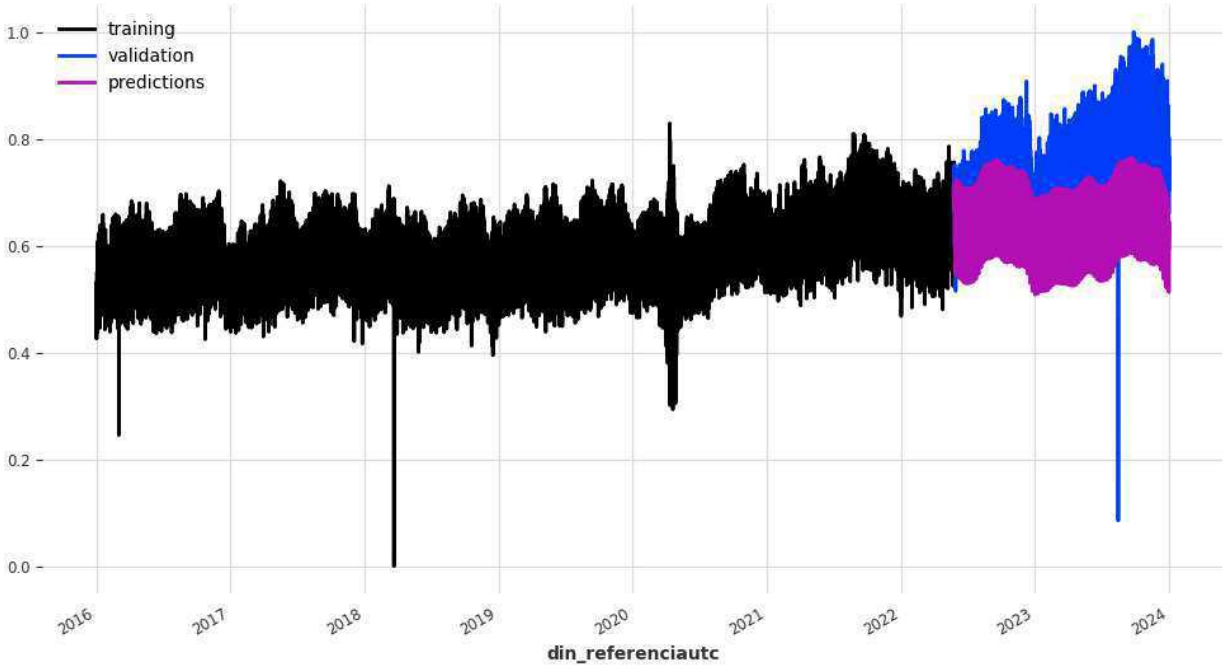
In [77]:
```
temp = Prophet()
temp.fit(train)
preds = temp.predict(len(val))
accuracy = mape(val, preds)
```

```
14:29:20 - cmdstanpy - INFO - Chain [1] start processing
14:30:48 - cmdstanpy - INFO - Chain [1] done processing
```

In [78]:
```
results_dict[f"Prophet N"] = accuracy

results_dict
```

Out[78]:
```
{'Prophet SECO': 13.539125846545344,
 'Prophet NE': 5.548868962998355,
 'Prophet N': 12.50964859039213}
```

In [79]:
```
plt.figure(figsize=(12, 6))

train.plot(label="training")
val.plot(label="validation")
preds.plot(label="predictions")

plt.legend()
plt.show()
```



Modelo 1.4: Região Sul

In [80]:
```
dados_carga_verificada_limpo_s = dados_carga_verificada_limpo[dados_carga_verificada_l
```

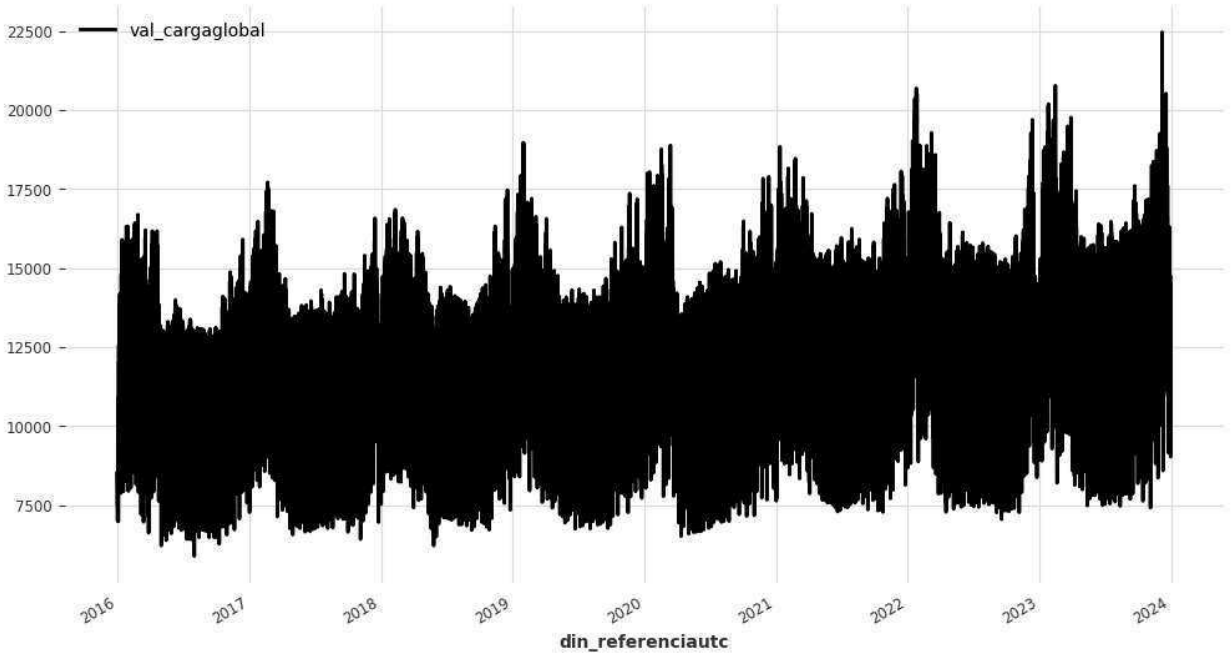
In [81]:
```
dados_carga_verificada_limpo_s.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 140258 entries, 140258 to 280515
Data columns (total 12 columns):
 #   Column                Non-Null Count   Dtype  
---  ------                --------------   -----  
 0   Unnamed: 0            140258 non-null  int64  
 1   cod_areacarga         140258 non-null  object 
 2   dat_referencia        140258 non-null  object 
 3   din_referenciautc     140258 non-null  object 
 4   val_cargaglobalcons   140258 non-null  float64
 5   val_consistencia      140258 non-null  float64
 6   val_cargasup          140258 non-null  float64
 7   val_cargansup         140258 non-null  float64
 8   val_cargaglobal       140258 non-null  float64
 9   val_cargaglobalsmmgd  140258 non-null  float64
 10  val_cargarvd          140258 non-null  int64  
 11  val_cargammgd         140258 non-null  float64
dtypes: float64(7), int64(2), object(3)
memory usage: 13.9+ MB
```

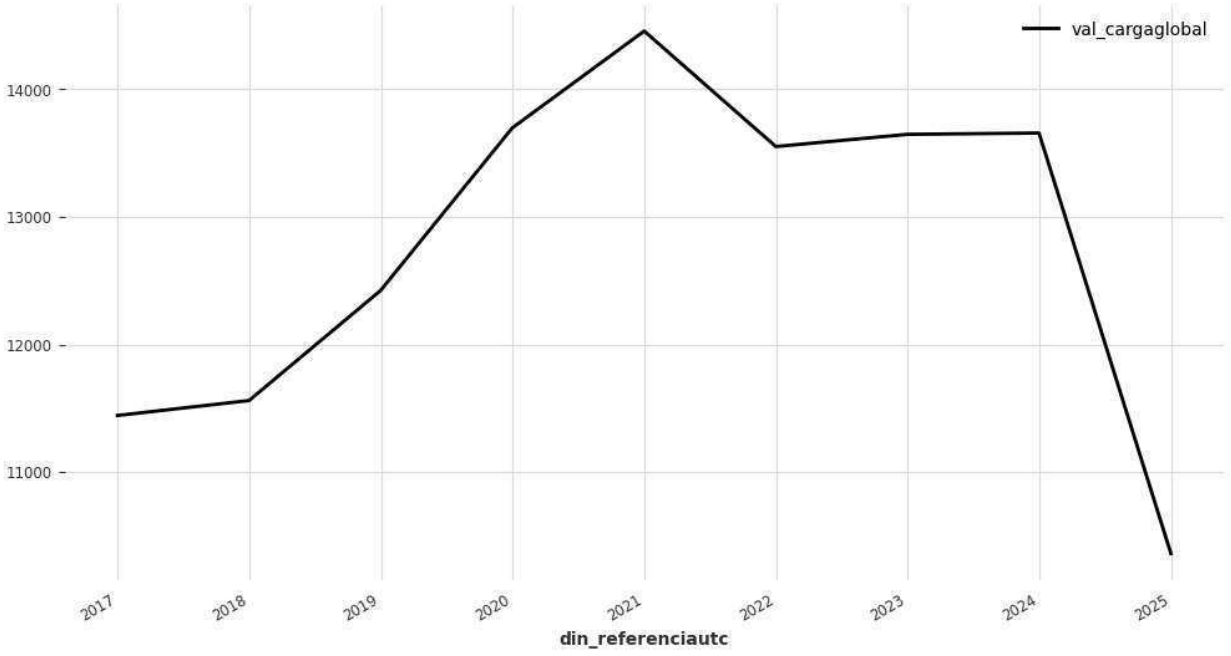
In [82]:
```
dados_carga_verificada_limpo_s['din_referenciautc'] = pd.to_datetime(dados_carga_verif
```

In [83]:
```
import matplotlib.pyplot as plt
from darts import TimeSeries

# Cria a série temporal
series_s = TimeSeries.from_dataframe(dados_carga_verificada_limpo_s, 'din_referenciaut

# Define o tamanho da figura
plt.figure(figsize=(12, 6))
```

```
series_s.plot()
plt.show()
```



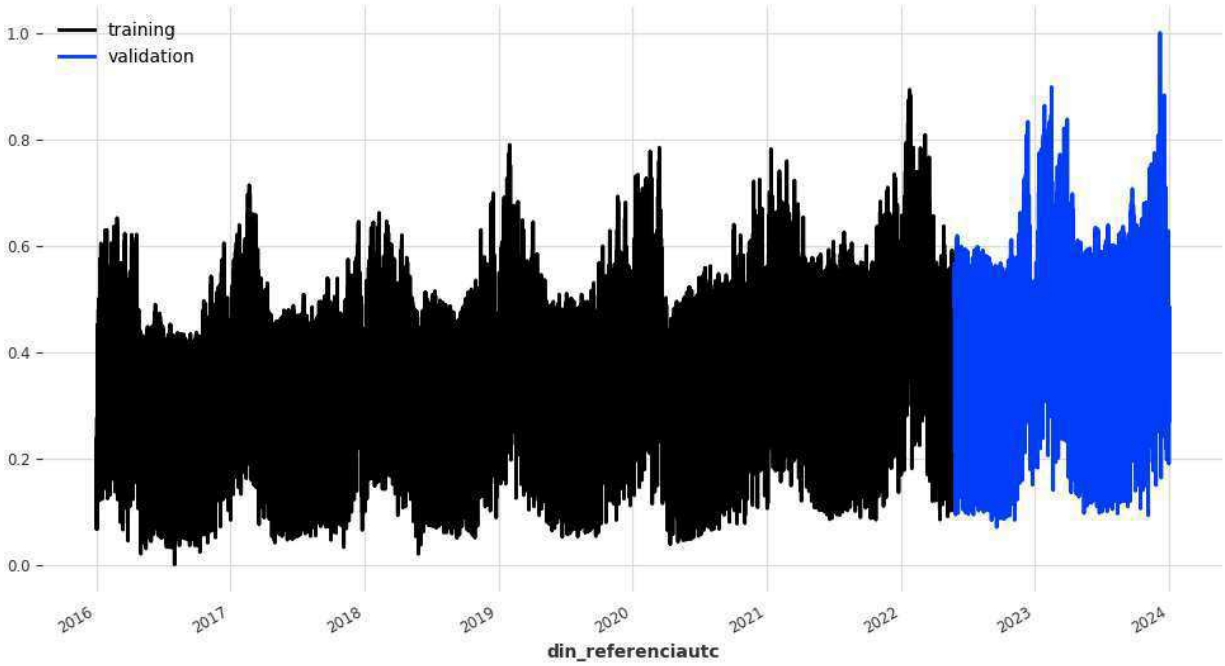
In [84]:
```
resampled_example = series_s.resample("Y")

# Define o tamanho da figura (em polegadas)
plt.figure(figsize=(12, 6))  # Aumenta o tamanho para 12x6 polegadas

# Plota a série
resampled_example.plot()

# Mostra o gráfico
plt.show()
```



In [85]:
```
scaler = MinMaxScaler()
ts_transformer = Scaler(scaler)
```

```
scaled_ts_s = ts_transformer.fit_transform(series_s)
```

In [86]:
```
train, val = (scaled_ts_s).split_before(0.8)# (we standardize by dividing by 100 since

# Define o tamanho da figura (em polegadas)
plt.figure(figsize=(12, 6))  # Aumenta o tamanho para 12x6 polegadas

# Plota a série
train.plot(label="training")
val.plot(label="validation")

# Mostra o gráfico
plt.show()
```



In [87]:
```
temp = Prophet()
temp.fit(train)
preds = temp.predict(len(val))
accuracy = mape(val, preds)
```

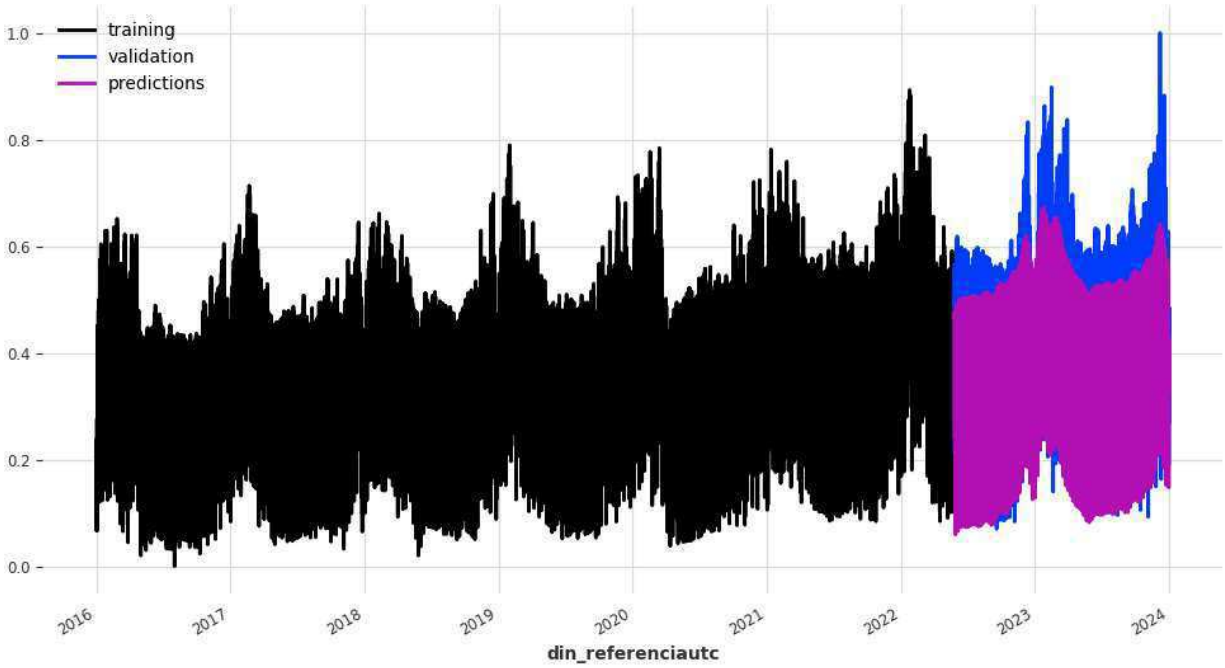
```
14:31:41 - cmdstanpy - INFO - Chain [1] start processing
14:32:52 - cmdstanpy - INFO - Chain [1] done processing
```

In [88]:
```
results_dict[f"Prophet S"] = accuracy

results_dict
```

Out[88]:
```
{'Prophet SECO': 13.539125846545344,
 'Prophet NE': 5.548868962998355,
 'Prophet N': 12.50964859039213,
 'Prophet S': 15.878125750929883}
```

O MAPE (Mean Absolute Percentage Error) é uma métrica amplamente utilizada para avaliar a precisão de modelos de previsão, especialmente em séries temporais. O MAPE é calculado como a média das diferenças absolutas entre os valores previstos e os valores reais, expressas como porcentagens dos valores reais. Por exemplo, um MAPE de 13.5% indica que, em média, as previsões do modelo estão erradas em 13.5% em relação aos valores observados.

In [89]:
```
plt.figure(figsize=(12, 6))

train.plot(label="training")
val.plot(label="validation")
preds.plot(label="predictions")

plt.legend()
plt.show()
```



In [90]:
```
dados_carga_verificada_limpo.head()
```

Out[90]:

	Unnamed: 0	cod_areacarga	dat_referencia	din_referenciautc	val_cargaglobalcons	val_consistencia	v
0	0	SECO	2016-01-01	2016-01-01 02:30:00+00:00	32604.021	0.0	
1	1	SECO	2016-01-01	2016-01-01 03:00:00+00:00	32236.307	0.0	
2	2	SECO	2016-01-01	2016-01-01 03:30:00+00:00	32312.062	0.0	
3	3	SECO	2016-01-01	2016-01-01 04:00:00+00:00	32206.387	0.0	
4	4	SECO	2016-01-01	2016-01-01 04:30:00+00:00	31973.025	0.0	

2 Previsão de Rampa de Carga - Dia Especial Natal

Após construir e validar o modelo inicial, usaremos **todo o conjunto de dados disponíveis** para treinar um novo modelo mais robusto. Este modelo será utilizado para realizar previsões para os próximos dias e horas ⏳. Entre essas previsões, vamos incluir especificamente o dia de Natal de 2024 🎄, onde faremos uma análise detalhada das rampas de carga. Isso nos permitirá avaliar as possíveis flutuações significativas de demanda durante o período natalino.

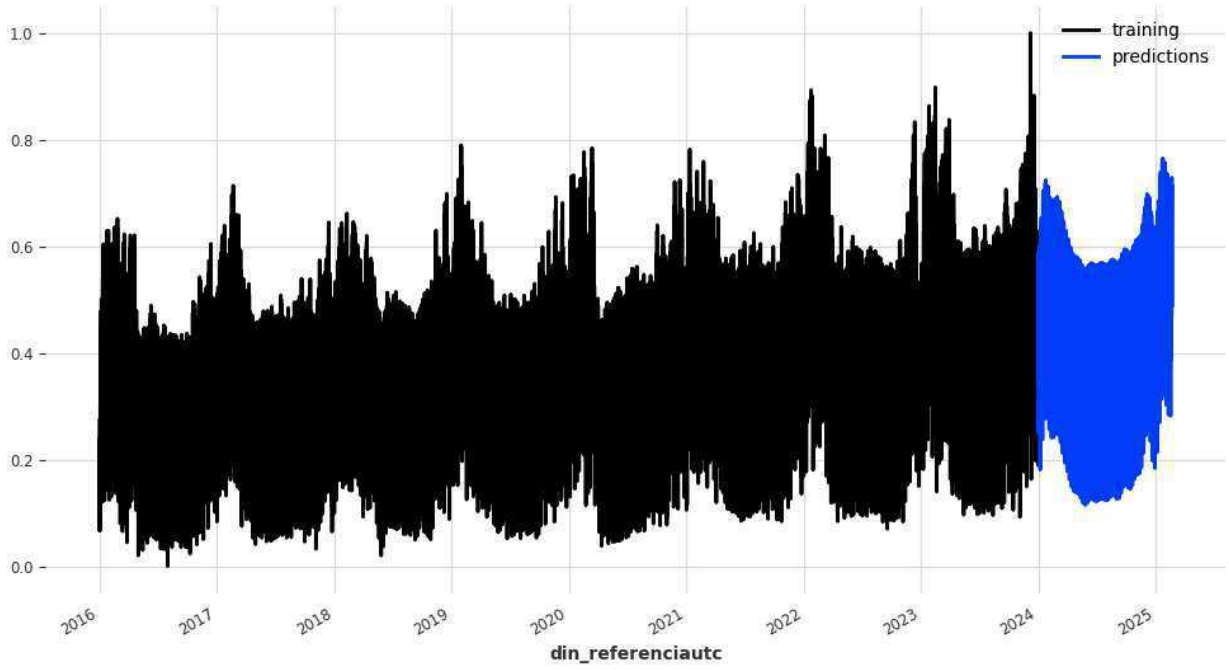
In [91]:
```
modelo = Prophet()
modelo.fit(scaled_ts_s)
```

```
14:33:33 - cmdstanpy - INFO - Chain [1] start processing
14:35:16 - cmdstanpy - INFO - Chain [1] done processing
```

Out[91]:
```
Prophet(add_seasonalities=None, country_holidays=None, suppress_stdout_stderror=True,
add_encoders=None, cap=None, floor=None)
```

In [92]:
```
preds = modelo.predict(20000)
```

In [93]:
```
plt.figure(figsize=(12, 6))

scaled_ts_s.plot(label="training")
preds.plot(label="predictions")

plt.legend()
plt.show()
```



In [94]:
```
# Reverter a transformação
predictions_real = ts_transformer.inverse_transform(preds)
```

In [95]:
```
forecast_df = predictions_real.pd_dataframe()
forecast_df.head()
```

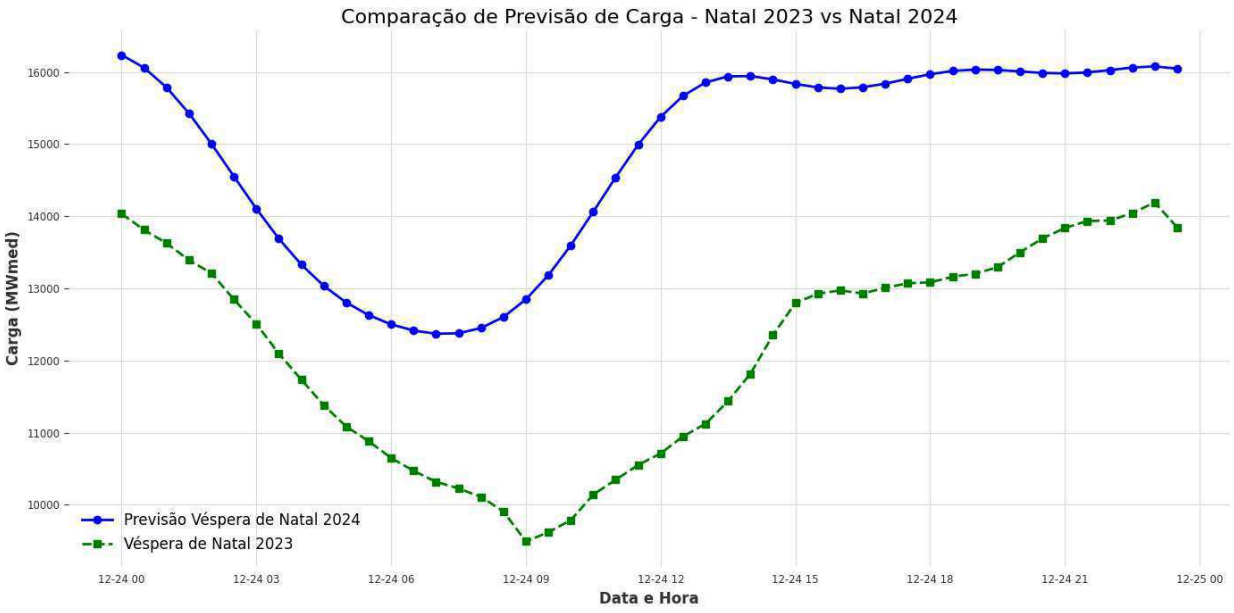
Out[95]:

din_referenciautc	component val_cargaglobal
2024-01-01 03:30:00	10906.593102
2024-01-01 04:00:00	10611.327555
2024-01-01 04:30:00	10382.411638
2024-01-01 05:00:00	10218.923855
2024-01-01 05:30:00	10113.124139

In [96]:
```
vespera_natal_24 = forecast_df[(forecast_df.index >= '2024-12-24 00:00:00') &
                               (forecast_df.index < '2024-12-25 00:00:00')]
```

In [97]:
```
series_s = series_s.pd_dataframe()

vespera_natal_23 = series_s[(series_s.index >= '2023-12-24 00:00:00') &
                            (series_s.index < '2023-12-25 00:00:00')]
```

In [98]:
```
plt.figure(figsize=(14, 7))

# Corrigindo para que fique sobreposto, apenas para efeito de visualizacao
vespera_natal_23.index = vespera_natal_23.index + pd.DateOffset(years=1)

plt.plot(vespera_natal_24.index, vespera_natal_24['val_cargaglobal'].values, label="Pr
plt.plot(vespera_natal_23.index, vespera_natal_23['val_cargaglobal'].values, label="Vé

plt.title('Comparação de Previsão de Carga - Natal 2023 vs Natal 2024', fontsize=16)
plt.xlabel('Data e Hora', fontsize=12)
plt.ylabel('Carga (MWmed)', fontsize=12)
plt.grid(True)

plt.legend(loc="best", fontsize=12, fancybox=True, shadow=True)

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



3 Modelo de Previsão da Carga Global com MMGD

(Com Incorporação de Dados de Incidência Solar)

Organização dos Dados Solares

Com base nos dados gerados pelo INPE e seus parceiros na nova edição do Atlas Brasileiro de Energia Solar, foi possível acessar informações detalhadas e realizar associações com os dados do ONS.

In [99]:
```
dados_solar = pd.read_csv("global_horizontal_means_sedes-munic.csv", sep=';')
dados_solar = dados_solar.drop(columns=['ID', 'LON', 'LAT', 'NAME', 'CLASS'])

dados_solar.head()
```

Out[99]:

	STATE	ANNUAL	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC
0	ACRE	4620	4655	4653	4268	4566	3945	4079	4285	4858	5102	5105	5157	4771
1	ACRE	4620	4655	4653	4268	4566	3945	4079	4285	4858	5102	5105	5157	4771
2	ACRE	4615	4678	4661	4231	4532	3917	4026	4283	4947	5136	5092	5102	4771
3	ACRE	4607	4629	4586	4281	4578	3962	4125	4332	4894	5068	5071	5077	4686
4	ACRE	4542	4486	4527	4264	4395	3957	3983	4228	4823	5070	5138	4952	4677

In [100…
```
regioes = {
    'N': ['AMAZONAS', 'RORAIMA', 'AMAPÁ', 'PARÁ', 'TOCANTINS', 'RONDÔNIA', 'ACRE'],
    'NE': ['MARANHÃO', 'PIAUÍ', 'CEARÁ', 'RIO GRANDE DO NORTE', 'PERNAMBUCO', 'PARAÍBA
    'SECO': ['SÃO PAULO', 'RIO DE JANEIRO', 'ESPÍRITO SANTO', 'MINAS GERAIS', 'MATO GR
    'S': ['PARANÁ', 'SANTA CATARINA', 'RIO GRANDE DO SUL']
}
```

In [101…
```
dados_solar['Regiao'] = dados_solar['STATE'].apply(lambda x: next((regiao for regiao,
dados_solar = dados_solar.drop(columns=['STATE', 'ANNUAL'])

dados_solar.head()
```

Out[101]:

	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	Regiao
0	4655	4653	4268	4566	3945	4079	4285	4858	5102	5105	5157	4771	N
1	4655	4653	4268	4566	3945	4079	4285	4858	5102	5105	5157	4771	N
2	4678	4661	4231	4532	3917	4026	4283	4947	5136	5092	5102	4771	N
3	4629	4586	4281	4578	3962	4125	4332	4894	5068	5071	5077	4686	N
4	4486	4527	4264	4395	3957	3983	4228	4823	5070	5138	4952	4677	N

In [102…
```
df_melted = dados_solar.melt(id_vars=['Regiao'], var_name='Mes', value_name='Incidenci
dados_solar_agg = df_melted.groupby(['Regiao', 'Mes'])['Incidencia_solar'].mean().rese
```

In [103…
```
mes_para_numero = {
    'JAN': '01', 'FEB': '02', 'MAR': '03', 'APR': '04', 'MAY': '05', 'JUN': '06',
    'JUL': '07', 'AUG': '08', 'SEP': '09', 'OCT': 10, 'NOV': 11, 'DEC': 12
}

# Converter a coluna 'Mes' usando o dicionário de mapeamento
dados_solar_agg['Mes'] = dados_solar_agg['Mes'].map(mes_para_numero)
```

In [104…
```
dados_solar_agg['Mes'] = '2023-' + dados_solar_agg['Mes'].astype(str)
dados_solar_agg['Mes'] = pd.to_datetime(dados_solar_agg['Mes'], format='%Y-%m').dt.to_

dados_solar_agg.head()
```

Out[104]:

	Regiao	Mes	Incidencia_solar
0	N	2023-04	4563.631111
1	N	2023-08	5386.668889
2	N	2023-12	4739.535556
3	N	2023-02	4654.571111
4	N	2023-01	4624.615556

Organização dos Dados de Valor de Carga de MMGD

In [105…
```
dados_carga_verificada_limpo.head()
```

Out[105]:

	Unnamed: 0	cod_areacarga	dat_referencia	din_referenciautc	val_cargaglobalcons	val_consistencia	v
0	0	SECO	2016-01-01	2016-01-01 02:30:00+00:00	32604.021	0.0	
1	1	SECO	2016-01-01	2016-01-01 03:00:00+00:00	32236.307	0.0	
2	2	SECO	2016-01-01	2016-01-01 03:30:00+00:00	32312.062	0.0	
3	3	SECO	2016-01-01	2016-01-01 04:00:00+00:00	32206.387	0.0	
4	4	SECO	2016-01-01	2016-01-01 04:30:00+00:00	31973.025	0.0	

In [106…
```
dados_carga_solar = dados_carga_verificada_limpo[['cod_areacarga', 'dat_referencia', '
dados_carga_solar['dat_referencia'] = pd.to_datetime(dados_carga_solar['dat_referencia

dados_carga_solar = dados_carga_solar[dados_carga_solar['dat_referencia'].dt.year == 2
```

In [107…
```
dados_carga_solar['dat_referencia'] = dados_carga_solar['dat_referencia'].dt.to_perioc
dados_carga_solar.head()
```

Out[107]:

	cod_areacarga	dat_referencia	val_cargammgd
122738	SECO	2023-01	110.07
122739	SECO	2023-01	110.03
122740	SECO	2023-01	109.97
122741	SECO	2023-01	110.13
122742	SECO	2023-01	110.33

In [108…
```
media_por_mes = dados_carga_solar.groupby(['cod_areacarga','dat_referencia'])['val_car
```

In [109…
```
merged_tabela = pd.merge(media_por_mes, dados_solar_agg, left_on=['cod_areacarga', 'da
merged_tabela.head()
```

Out[109]:

	cod_areacarga	dat_referencia	val_cargammgd	Regiao	Mes	Incidencia_solar
0	N	2023-01	209.859227	N	2023-01	4624.615556
1	N	2023-02	219.079315	N	2023-02	4654.571111
2	N	2023-03	205.610840	N	2023-03	4561.631111
3	N	2023-04	208.469340	N	2023-04	4563.631111
4	N	2023-05	248.533129	N	2023-05	4547.097778

Modelo de Previsão

In [110…
```
merged_tabela = merged_tabela[['cod_areacarga', 'dat_referencia', 'val_cargammgd', 'Ir
merged_tabela['dat_referencia'] = merged_tabela['dat_referencia'].dt.to_timestamp()
```

In [111…
```
merged_tabela = merged_tabela[:12]
merged_tabela.head()
```

Out[111]:

	cod_areacarga	dat_referencia	val_cargammgd	Incidencia_solar
0	N	2023-01-01	209.859227	4624.615556
1	N	2023-02-01	219.079315	4654.571111
2	N	2023-03-01	205.610840	4561.631111
3	N	2023-04-01	208.469340	4563.631111
4	N	2023-05-01	248.533129	4547.097778

In [112…
```
merged_tabela.set_index('dat_referencia', inplace=True)

train = merged_tabela.iloc[:10]
test = merged_tabela.iloc[10:]
```

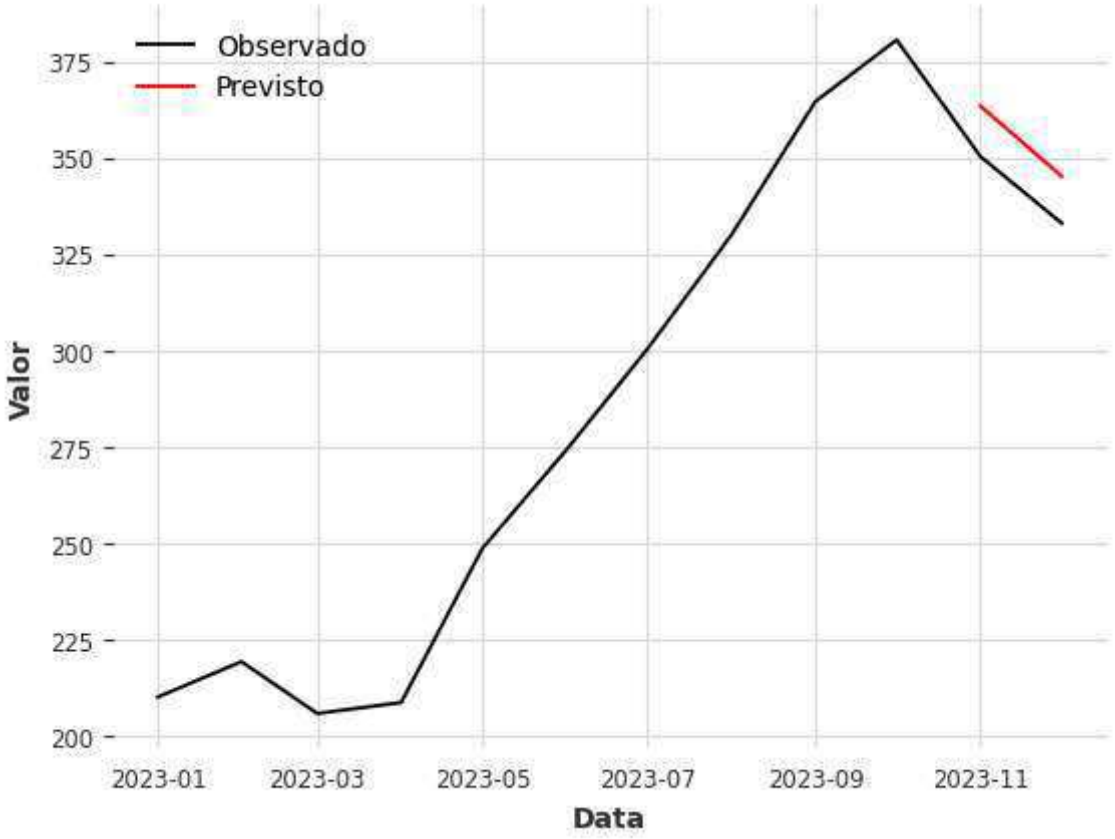
In [113…
```
# Modelo SARIMAX com variáveis exógenas
model = SARIMAX(train['val_cargammgd'],
                exog=train[['Incidencia_solar']],
                order=(1, 0, 0),
                seasonal_order=(0, 0, 0, 0))

model_fit = model.fit(disp=False)
```

```
/home/anyelle/.pyenv/versions/3.10.6/envs/lewagon/lib/python3.10/site-packages/statsm
odels/tsa/base/tsa_model.py:473: ValueWarning: No frequency information was provided,
so inferred frequency MS will be used.
  self._init_dates(dates, freq)
/home/anyelle/.pyenv/versions/3.10.6/envs/lewagon/lib/python3.10/site-packages/statsm
odels/tsa/base/tsa_model.py:473: ValueWarning: No frequency information was provided,
so inferred frequency MS will be used.
  self._init_dates(dates, freq)
```

In [114…
```
# Previsões
forecast = model_fit.predict(start=len(train), end=len(merged_tabela)-1, exog=test[['I

# Avaliação do modelo
rmse = np.sqrt(mean_squared_error(test['val_cargammgd'], forecast))
print(f'RMSE: {rmse}')
```

```
RMSE: 12.675884276157253
```

In [115…
```
plt.plot(merged_tabela['val_cargammgd'], label='Observado')
plt.plot(forecast, label='Previsto', color='red')
plt.xlabel('Data')
plt.ylabel('Valor')
plt.legend()
plt.show()
```



In [117…
```
np.mean(np.abs((test['val_cargammgd'] - forecast) / test['val_cargammgd'])) * 100
```

Out[117]:
```
3.707181322428095
```

In [ ]:
```

```