



# Experiência na publicação de bibliotecas python na otimização de processos de consolidação de medições na EDP Espírito Santo

**Tema:** Planejamento da Expansão

**Autores:** Henrique Domingos Panceri; Joao Marcus Ramos Bacalhau

**Co-Autores:** -

**Empresa:** EDP Espírito Santo Distribuição de Energia S.A.

---

## Resumo

A aquisição e consolidação de dados em redes de distribuição de média tensão apresenta desafios significativos para as concessionárias de energia. Inconsistências nos dados, valores ausentes e falhas em equipamentos dificultam, em especial, análises de planejamento da expansão. Para solucionar esses problemas, este artigo propõe uma solução baseada em Python para o pré-processamento e imputação de dados faltantes em séries temporais de medições dos alimentadores. Foi desenvolvida uma biblioteca estruturada em Python, publicada no PyPI, contendo funções modulares para limpeza de dados, detecção de outliers e reconstrução de séries temporais. A implementação inclui controle de versão via GitHub e documentação completa no ReadTheDocs, garantindo acessibilidade e manutenção eficiente.

A metodologia foi aplicada ao Plano Anual de Obras de uma distribuidora, resultando em uma redução expressiva no tempo de processamento dos dados, de aproximadamente 19 horas para uma hora, melhorando a eficiência e a precisão das avaliações do desempenho da rede. Além disso, a integração com o Databricks permitiu maior escalabilidade e análises avançadas. Os resultados demonstram que a automação e a padronização do tratamento de dados aprimoram a tomada de decisão no contexto do planejamento da expansão.

## 1. Introdução

### 1. Desafios na Consolidação de Medições de alimentadores em Redes de Distribuição de Média Tensão

As redes de distribuição de média tensão representam um componente crucial no sistema elétrico, conectando subestações aos consumidores finais. Apesar de sua importância, a consolidação de medições nessas redes apresenta uma série de desafios técnicos e operacionais, que impactam diretamente o planejamento e a operação eficiente do sistema de distribuição.

Um dos principais desafios está relacionado à aquisição e armazenamento de dados. A coleta de informações ao longo do sistema envolve uma cadeia de atores, desde os sensores que realizam as medições até os sistemas do Operador do Sistema de Distribuição (Distribution System Operator – DSO), onde os dados são armazenados. Cada etapa do processo está sujeita a interferências, como falhas de equipamentos e condições climáticas adversas, que podem prejudicar a transferência de dados. Além disso, paradas de equipamentos para manutenção preventiva e manobras de carga, comuns no contexto das

distribuidoras, também geram lacunas e dados fora do padrão (outliers). A grande variabilidade de dados coletados por diferentes tecnologias de relés e métodos de amostragem torna o processo de consolidação ainda mais desafiador. Por exemplo, enquanto alguns relés coletam dados a cada cinco minutos, outros o fazem a cada minuto, criando inconsistências temporais. Esses fatores dificultam a integridade e a confiabilidade das informações utilizadas no planejamento do sistema.

A ocorrência de dados ausentes é outro grande obstáculo. Quando valores estão ausentes, seja por falhas de comunicação ou manutenção dos equipamentos, as análises baseadas nesses dados tornam-se suscetíveis a erros. A simples exclusão de amostras incompletas pode levar a resultados enviesados ou imprecisos. Por isso, o problema da imputação de valores ausentes tem se tornado uma área central na análise de dados incompletos. Nesse contexto, o trabalho “Missing Data Analysis and Imputation Method for Medium Voltage Distribution Network Feeders” (BACALHAU, 2020) traz uma abordagem diferente para o problema em questão, propondo um método de imputação baseado na correlação de séries temporais e na normalização de padrões diários, ajustados linearmente aos valores mínimo e máximo das séries. Esse método garante maior precisão na recuperação de dados ausentes, fortalecendo a qualidade das informações utilizadas no planejamento e na operação das redes de distribuição de média tensão.

Inicialmente a implementação e testes foram feitos utilizando bases temporais de uma distribuidora de energia elétrica. Entretanto, viu-se a necessidade da aplicação do estudo utilizando ferramenta computacional robusta, ao qual o Python se mostrou adequado.

## **2. Soluções para organização e disseminação de scripts**

### **1. Python**

Python é uma linguagem de programação de alto nível reconhecida por sua simplicidade e legibilidade, tornando-se ideal para iniciantes e desenvolvedores experientes. Sua sintaxe clara, similar ao inglês, reduz a curva de aprendizado e aumenta a produtividade, permitindo a escrita de programas em menos linhas de código em comparação com linguagens como C++ ou Java (Peta, 2022; Kelly, 2015). Além disso, suporta múltiplos paradigmas de programação, como orientação a objetos, funcional e imperativa, o que amplia sua flexibilidade e aplicabilidade (Kelly, 2015).

A versatilidade de Python possibilita seu uso em diversas áreas, incluindo desenvolvimento web, análise de dados, aprendizado de máquina e automação (Giapros, 2022; Galimullin, 2023). Bibliotecas como NumPy e Pandas são essenciais para análise de dados, permitindo a implementação de funcionalidades complexas de forma eficiente ("Python for web development", 2022; Seyidova & Shakhayev, 2023).

Python conta com uma comunidade ativa que contribui para sua evolução contínua, tornando-o confiável e popular (Peta, 2022). Empresas como Google, Amazon e Microsoft utilizam amplamente a linguagem, validando sua relevância no mercado (Peta, 2022). No campo da automação, Python se destaca por facilitar tarefas repetitivas, aumentando a produtividade e reduzindo custos (Galimullin, 2023). Sua compatibilidade com sistemas e plataformas reforça sua utilidade em soluções integradas (Khan et al., 2023).

Apesar de suas vantagens, Python apresenta limitações, como menor desempenho em relação a linguagens compiladas e ineficácia em aplicações de baixo nível ou tempo real. Ainda assim, sua legibilidade, suporte comunitário e bibliotecas extensivas consolidam sua posição como uma ferramenta poderosa para múltiplas aplicações.

### **2. PyPI**

O Índice de Pacotes Python (PyPI) é o repositório central de pacotes de software Python, essencial para o ecossistema da linguagem. Ele facilita a distribuição e o gerenciamento de bibliotecas, promovendo a colaboração na comunidade (Bommarito & Bommarito, 2019). Como repositório global, o PyPI permite

que desenvolvedores publiquem pacotes de forma acessível, reforçando os princípios de código aberto e incentivando a reutilização de software (Kranas et al., 2022).

O PyPI hospeda milhões de bibliotecas, abrangendo diversas aplicações (Zadka, 2021). Ao oferecer ferramentas prontas, aumenta a produtividade dos desenvolvedores. Seu crescimento, com taxa média anual de 47% nos últimos 15 anos, destaca sua influência no desenvolvimento de software (Bommarito & Bommarito, 2019). A integração com ferramentas como o pip simplifica a instalação de pacotes, incentivando desenvolvedores a compartilhar projetos e promovendo maior adoção pela comunidade (Takefuji, 2021). Além disso, a plataforma possibilita feedback colaborativo e melhorias contínuas, impactando positivamente a qualidade dos pacotes (Sitthithanasakul et al., 2022). O suporte ao gerenciamento de dependências aumenta a confiabilidade de sistemas complexos (Wang et al., 2020).

Outro diferencial é sua acessibilidade. A publicação no PyPI amplia o alcance global das bibliotecas, superando barreiras de visibilidade (Günther et al., 2021). Ao implementar padrões e melhores práticas, o PyPI assegura qualidade e longevidade dos pacotes, fomentando um ecossistema sustentável (Zadka, 2021; Valiev et al., 2018).

Apesar de suas vantagens, o PyPI enfrenta desafios, como conflitos de dependências, que podem comprometer projetos inteiros. Ferramentas como o Watchman ajudam a mitigar esses problemas, mas evidenciam a necessidade de aprimoramentos contínuos (Wang et al., 2020).

Em síntese, o PyPI vai além de um repositório: é um motor de crescimento e colaboração na comunidade Python. Sua organização e acessibilidade impulsionam a inovação e fortalecem a evolução do ecossistema Python.

### 3. **Git e Github**

Git e GitHub são ferramentas essenciais para o controle de versão e colaboração em projetos. Git é um sistema de controle de versão distribuído que permite rastrear alterações em arquivos e diretórios, mantendo a integridade dos projetos (Ghodke & Chavan, 2024). Desenvolvedores podem criar repositórios locais clonando repositórios remotos, possibilitando trabalho offline.

GitHub complementa o Git com uma interface web que facilita a colaboração, oferecendo ferramentas como Issues, Pull Requests e Discussions para comunicação e gestão de tarefas (Guzura, 2023). Ele também integra fluxos de trabalho automatizados e pipelines CI/CD, otimizando o desenvolvimento moderno (Chen et al., 2024). Além disso, o GitHub tem sido amplamente usado na pesquisa científica para gerenciar códigos e dados, promovendo a ciência aberta e a reprodutibilidade (Guzura, 2023; Chen et al., 2024). Na educação, sua inclusão em currículos ensina controle de versão e programação colaborativa, preparando estudantes para práticas modernas de desenvolvimento (Herlambang et al., 2023).

Os benefícios dessas ferramentas são evidentes: elas estruturam a gestão de alterações, facilitam a colaboração e asseguram a integridade dos dados, sendo indispensáveis em projetos com múltiplos colaboradores (Coker, 2024; Davis, 2015). No entanto, iniciantes podem enfrentar desafios devido à curva de aprendizado, especialmente com a interface de linha de comando do Git (Guzura, 2023). Apesar disso, recursos educacionais têm tornado essas ferramentas mais acessíveis, ampliando sua adoção em diferentes campos.

Originalmente projetados para o desenvolvimento de software, Git e GitHub expandiram sua utilidade, tornando-se ferramentas robustas para colaboração em domínios diversos. Apesar da complexidade inicial, seus benefícios em promover eficiência e reprodutibilidade destacam sua relevância em múltiplas disciplinas.

### 4. **Documentação**

ReadTheDocs é uma plataforma web que automatiza a construção, versionamento e hospedagem de documentações para projetos de software. Integra-se a sistemas de controle de versão, como GitHub e GitLab, permitindo atualizações contínuas e reduzindo o esforço manual em formatação e compilação. Utiliza o Sphinx para gerar documentação a partir de arquivos reStructuredText ou Markdown, garantindo consistência e alinhamento com o código mais recente.

A plataforma destaca-se pela hospedagem gratuita para projetos de código aberto, suporte a domínios personalizados e funcionalidade de busca integrada, tornando a navegação prática e acessível. Para projetos privados, embora haja custos, os recursos robustos oferecidos continuam atraentes. Além disso, ReadTheDocs promove colaboração eficiente com suporte a pull requests e revisões, facilitando feedback e melhorias contínuas.

Entre os benefícios estão a automação, que padroniza o conteúdo entre versões, e a gratuidade para projetos abertos, uma economia relevante para desenvolvedores independentes. A funcionalidade de busca e a documentação versionada também aprimoram a experiência do usuário.

Contudo, desafios incluem a familiaridade inicial com reStructuredText ou Markdown e custos para projetos privados. Apesar disso, a robustez e facilidade de uso consolidam o ReadTheDocs como uma escolha popular.

## 2. Desenvolvimento

### 1. Objetivo

O objetivo deste trabalho foi consolidar as metodologias e códigos previamente desenvolvidos em Python para o pré-processamento e preenchimento de lacunas em dados de séries temporais de alimentadores de média tensão, integrando-os em uma biblioteca Python. Essa biblioteca foi publicada no repositório PyPI, com documentação acessível ao público, além de contar com controle de versões e rastreamento de bugs por meio de um repositório Git. O objetivo final foi viabilizar o uso desses códigos em diversas aplicações dentro da distribuidora, ao mesmo tempo em que se promove a disseminação da solução, facilita sua manutenção e amplia sua reutilização.

### 2. Metodologia

A publicação de bibliotecas na linguagem Python exigiu, em diversos casos, a reescrita de códigos originalmente implementados como scripts locais, transformando-os em funções. Para facilitar o uso por parte do usuário final, essas funções foram organizadas em módulos específicos:

- **Clean:** contém funções para eliminação de *outliers*.
- **Util:** reúne funções utilitárias, como o cálculo do percentual de lacunas nos dados, além de outras funções auxiliares usadas por diferentes módulos.
- **Fill:** implementa métodos de preenchimento de lacunas, incluindo o NSSC.
- **Examples:** disponibiliza exemplos de uso da biblioteca para novos usuários.

Além da conversão dos códigos locais para funções modulares, foi necessário criar documentação explicativa em formato *Markdown* para cada função. Também foram implementadas validações dos dados de entrada, acompanhadas de *warnings* e mensagens de aviso para alertar os usuários sobre possíveis inconsistências nos dados fornecidos.

Para organizar todo o processo de portabilidade dos códigos, que durou aproximadamente um ano e oito meses, utilizou-se o GitHub como repositório GIT. Os testes foram realizados na plataforma *Test PyPI*, que permite a publicação e a execução de testes antes do lançamento oficial da biblioteca.

Após a publicação, testes adicionais foram conduzidos para garantir o funcionamento adequado de todas as funções, bem como a validação dos exemplos elaborados.

Com as funções devidamente comentadas, os códigos Markdown foram compilados e publicados no site ReadTheDocs, tornando a consulta e a busca por referências da biblioteca mais fáceis.

- **Aplicação dos Resultados**

1. **Plano de Obras**

Anualmente, as distribuidoras de energia elétrica realizam estudos com o intuito de avaliar o desempenho do sistema elétrico sob sua concessão e propor melhorias no sistema, a fim de mitigar possíveis violações nos critérios de qualidade e continuidade do fornecimento de energia. Esses estudos são baseados em dados históricos anuais de grandezas elétricas por alimentador. Como resultado dos estudos, é definido um conjunto de obras, que é proposto e discutido quanto à sua aplicabilidade e viabilidade financeira. A este conjunto de obras, dá-se o nome de Plano de Obras Anual (POA).

Devido ao custo envolvido, ainda não é possível a implementação de medição específica em todos os pontos de interesse. Por isso, também são utilizados os dados de equipamentos de proteção da rede elétrica, assumindo o erro de precisão associado a esse tipo de equipamento. Para cada alimentador, são esperados 105.120 registros por grandeza, representando uma taxa de amostragem de 5 minutos. Entretanto, ao realizar a análise, é muito comum observar lacunas de dados faltantes. Também, observa-se a influência de manobras realizadas de forma temporária na rede, que mascaram o comportamento padrão do alimentador. A ferramenta discutida nesse artigo se mostrou poderosa para o saneamento dos dados e a reconstrução das curvas.

O objetivo geral é obter os níveis de carregamento máximo e mínimo, e a performance da tensão e do fator de potência. Por isso, são avaliadas as grandezas de tensão, corrente e fator de potência, trifásicas, para se obter os valores de energia e demanda, integralizados. Diante dos dados a serem analisados, foi utilizada a biblioteca *MVDataProcessing* no Python para aplicação da metodologia de análise:

1. **Preparação do arquivo base**

Devido ao volume de dados, as medições foram extraídas do sistema da distribuidora e armazenadas em arquivo do tipo “.feather”. Este arquivo reúne os dados de todos os alimentadores a serem estudados. Deste arquivo, novos arquivos por alimentador foram criados, contendo os dados de Corrente por fase da saída do alimentador (IA, IB e IC), tensão do barramento ao qual cada alimentador está conectado (VA, VB e VC), Demanda ativa (MW), Demanda Reativa (MVar) e fator de potência. Naturalmente, o arquivo por alimentador reunirá os dados registrados no período de extração, e as lacunas são mantidas no arquivo para que a série temporal da massa de dados seja preservada. Esta etapa é comum a qualquer metodologia aplicada, pois se refere a seleção da base de dados a ser estudada. Entretanto, a forma de armazenamento dos dados de medição dos alimentadores pode implicar em aumento de produtividade de processo, se for possível a extração por alimentador de maneira direta e individual.

2. **Tratamento dos dados**

Nesta parte, os arquivos são subamostrados para uma taxa de atualização de 10 minutos, utilizando a função *DataSynchronization* da biblioteca *MVDataProcessing*. Com base nos registros de manobras temporárias do centro de operações da distribuidora, é possível inserir lacunas conhecidas para que estes momentos não influenciem nas análises reais a serem efetuadas. Para isso, utilizou-se a



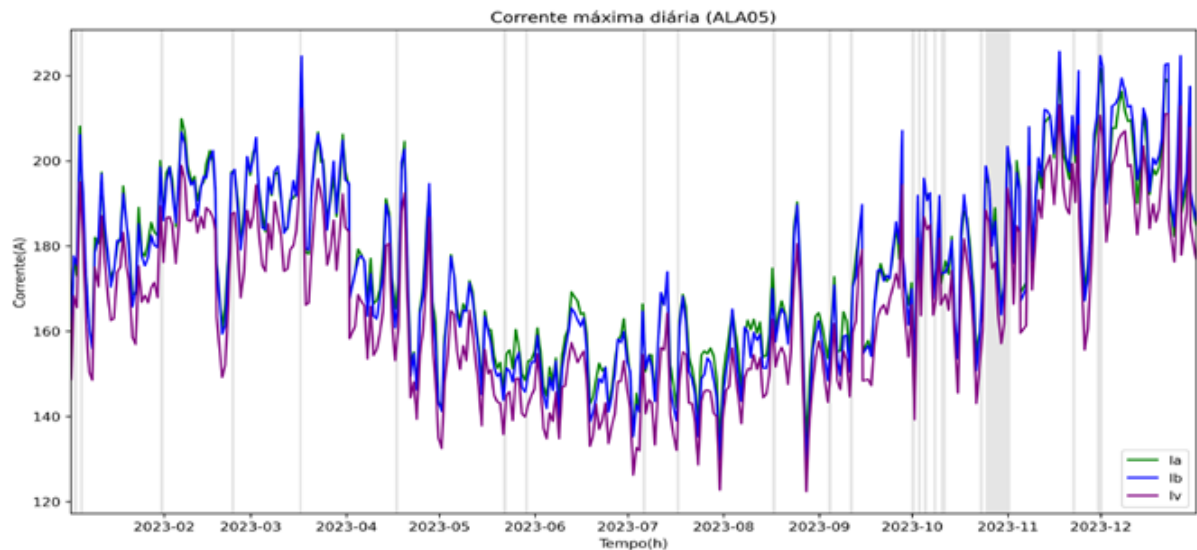
função `MarkNanPeriod` da biblioteca `MVDataProcessing`, para desconsiderar os dados registrados nos períodos inseridos. A função `CountMissingData` da biblioteca `MVDataProcessing` também é utilizada para acompanhar a evolução dos dados faltantes. É esperado que não haja uma perda significativa de dados devido a manobras. Entretanto, se não for possível identificar período satisfatório em que os dados representem o comportamento do alimentador, os dados não serão aproveitados e outras formas de estudo deverão ser avaliadas. Por isso, neste momento é crucial a correta utilização da biblioteca e a fiscalização contínua da integridade da massa de dados. Uma dificuldade enfrentada se refere a qualidade dos registros de manobra temporária. Após a tratativa dos dados, é comum observar curvas com amostras que, visivelmente, não representam o comportamento normal do alimentador. Por isso, é aplicada nova metodologia para identificação e saneamento de outliers:

- Utilização da função `RemoveOutliersHardThreshold` da biblioteca `MVDataProcessing`, para desconsiderar amostras em que é sabido erro. Exemplo: registros de tensão muito acima do nominal.
- Utilização da função `RemoveOutliersQuantile` da biblioteca `MVDataProcessing` para avaliação dos quantis da base de dados e exclusão dos outliers;
- Utilização da função `RemoveOutliersMMADMM` da biblioteca `MVDataProcessing` para avaliação da média móvel dos dados e exclusão das amostras com variação abrupta;
- Utilização da função `RemoveOutliersHistogram` da biblioteca `MVDataProcessing` para avaliação do histograma da base amostral;

Todas as funções possuem parâmetros definidos e avaliados com base na metodologia de teste prático. Ao final de cada análise de outliers, a saúde dos dados é medida para acompanhar o andamento da aproveitabilidade da base de dados. Um novo arquivo é gerado por alimentador, contendo a série temporal com dados que podem ser aproveitados.

- **Reconstrução dos dados**

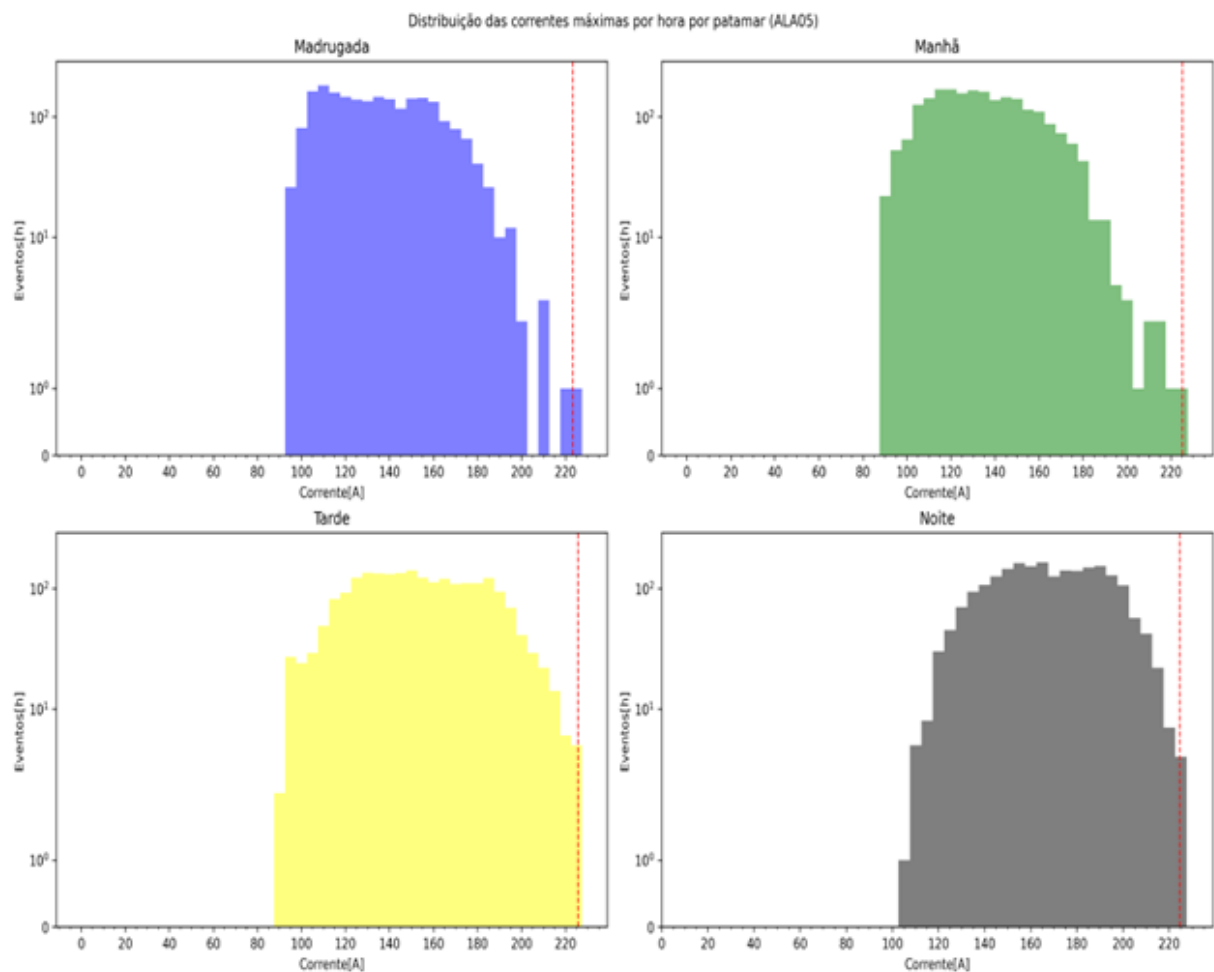
A partir dos dados já saneados de cada alimentador, a próxima etapa é reconstruir a série temporal, identificando uma curva de carga padrão para preenchimento das lacunas de dados. Para isso, inicia-se aplicando a função `PhaseProportionInput` da biblioteca `MVDataProcessing` para as grandezas de tensão e corrente. Esta função tem como objetivo reconstruir os dados de um sistema trifásico a partir dos dados existentes de uma fase pelo menos. Foi utilizada a interpolação simples para reconstrução das pequenas lacunas, antes de aplicação da função `NSSCInput` da biblioteca `MVDataProcessing`. Esta função identifica um padrão de carga semanal a partir dos dados, define uma curva, e a aplica aos dados faltantes da série original. Desta forma, as lacunas são preenchidas com uma média esperada do alimentador, a partir dos dados reais observados. Considerando o arquivo totalmente preenchido, após tratativas da base e reconstrução, as potências ativa e reativa, além das energias ativa e reativa, são calculadas a partir das grandezas de tensão, corrente e fator de potência, obtendo assim um valor de demanda integralizado para cada alimentador.

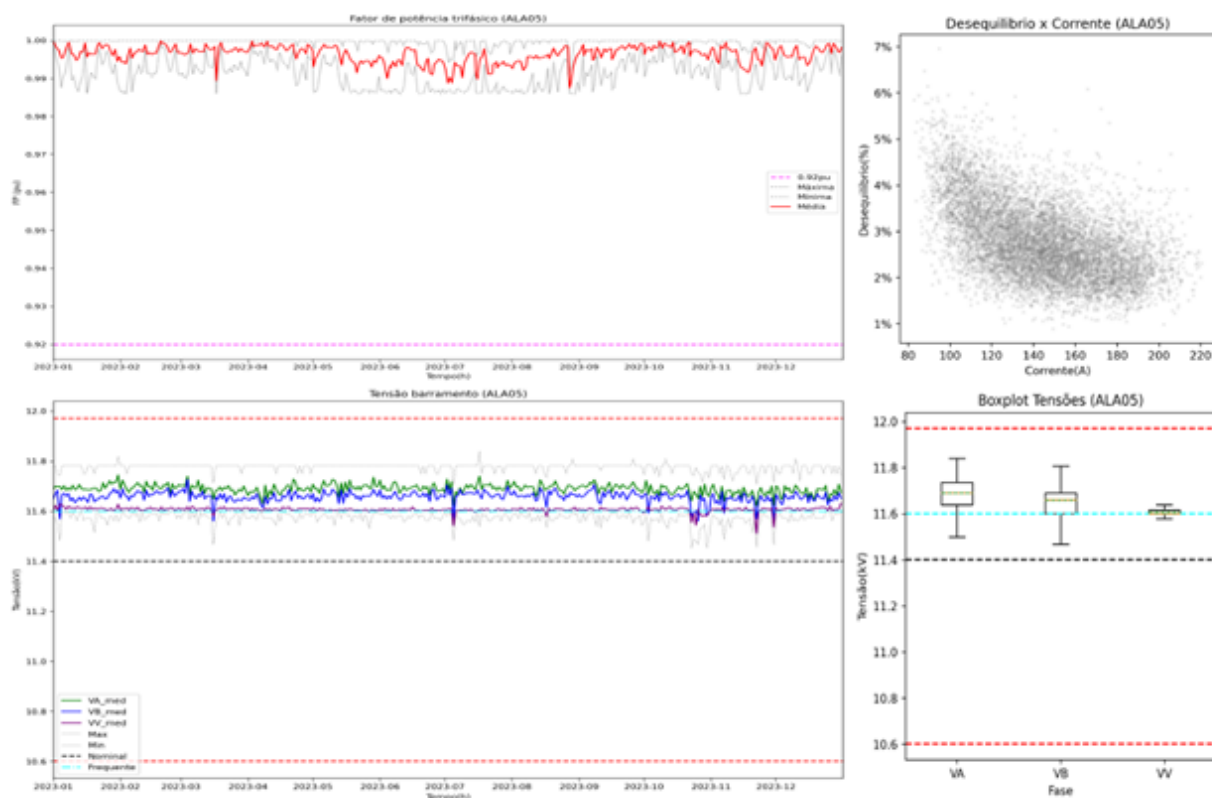


obs: os períodos em que os dados foram recompostos a partir da curva foram destacados com fundo cinza.

## • Resultados

Também foram utilizadas as funções padrão do Python para manipulação e criação de relatórios para cada alimentador, obtendo visões de máximas e mínimas por patamar, curvas de carga, curva de tensão, comportamento da corrente, desequilíbrio de corrente.





A reestruturação dos scripts em forma de biblioteca possibilitou não somente um ganho de organização e de facilidade de manutenção dos códigos como também um ganho de tempo de execução do processo, além de maior confiabilidade nos dados. Para o caso exemplificado neste artigo, o Plano Anual de Obras teve ganho de tempo de processamento dos dados, reduzindo de aproximadamente 19 horas para execução de todo o processo, para um tempo aproximado de uma hora. Os códigos elaborados são de fácil entendimento, e a aplicação da biblioteca MVDataProcessing facilitaram a elaboração da metodologia de processamento de dados utilizada para a definição das bases de dados por alimentador do Plano de Obras da distribuidora.

Em adição, a biblioteca criada pode ser utilizada via Databricks. A integração de uma nova biblioteca Python no Databricks muda completamente a forma como os dados são manipulados e analisados. Os usuários podem aproveitar funcionalidades avançadas de processamento de dados, análise, exclusão de amostras divergentes, definição e reconstrução de curvas, tudo dentro de um ambiente unificado. Isso não só aumenta a eficiência e a produtividade, mas também permite uma análise de dados mais profunda e integração com outras ferramentas existentes. A flexibilidade e a escalabilidade do Databricks, combinadas com as capacidades dessa nova biblioteca, oferecem uma solução poderosa para enfrentar desafios complexos de grandes volumes de dados.

### 3. Conclusão

A adoção dessa nova abordagem para a organização dos scripts de tratamento de dados trouxe benefícios significativos para o ambiente de produção. A automatização do processo reduziu consideravelmente a ocorrência de erros humanos, garantindo maior precisão e confiabilidade nas informações utilizadas para



análise. Além disso, a redução do tempo de processamento permitiu que grandes volumes de dados fossem tratados com maior eficiência, um fator essencial para estudos de expansão de alimentadores.

Outro aspecto crucial foi a implementação de um repositório centralizado para os códigos utilizados no tratamento de dados. Esse repositório não apenas facilitou o acesso e a reutilização dos scripts por diferentes usuários e equipes – independentemente do ambiente de execução, seja localmente ou em plataformas como Databricks – como também promoveu maior padronização e consistência nos processos analíticos. Dessa forma, as análises passaram a ser mais homogêneas, reduzindo discrepâncias entre resultados gerados por diferentes fontes e garantindo maior rastreabilidade e reprodutibilidade dos estudos.

No caso analisado, os impactos positivos foram evidentes. A otimização do tempo de processamento não só reduziu significativamente o esforço necessário para o tratamento dos dados, como também abriu novas possibilidades para análises mais aprofundadas dos fenômenos estudados. Isso permitiu que os especialistas focassem na interpretação dos resultados, em vez de dedicarem grande parte do tempo a ajustes manuais e correções de inconsistências nos dados.

Por fim, ao melhorar a organização, a automação e a confiabilidade do tratamento dos dados, essa metodologia se mostrou essencial para aprimorar o processo de tomada de decisão baseado em dados. Com informações mais precisas e acessíveis em menor tempo, os estudos podem ser conduzidos de forma mais eficiente, resultando em uma alocação mais estratégica dos recursos financeiros e humanos.

## 4. Referências bibliográficas

- Bommarito, M. J., & Bommarito, C. J. (2019). Understanding the Python Ecosystem: Trends and Insights.
- Günther, F., Smith, R., & Bell, A. (2021). Global Accessibility of Open-Source Software.
- Kranas, T., et al. (2022). Open-Source Innovation in Academia: The Role of PyPI.
- Sithithanasakul, A., et al. (2022). Community Dynamics in Open-Source Software.
- Takefuji, Y. (2021). Efficient Software Distribution with PyPI.
- Valiev, M., et al. (2018). The Collaborative Benefits of Python Package Repositories.
- Wang, Z., et al. (2020). Dependency Management Challenges in Large-Scale Software Projects.
- Zadka, M. (2021). Python Packaging Essentials.
- Chen, H., Smith, A., & Taylor, R. (2024). GitHub workflows in modern software development. *Journal of Technology Management*, 18(2), 45–67.
- Coker, R. (2024). Best practices in collaborative coding. *Software Engineering Quarterly*, 32(1), 112–130.
- Davis, J. (2015). The rise of Git and GitHub in education. *Educational Technology Today*, 10(3), 15–20.
- Ghodke, M., & Chavan, P. (2024). Version control essentials: Understanding Git. *Computing and Information Systems Review*, 29(3), 78–90.
- Guzura, M. (2023). GitHub for collaborative projects: Features and challenges. *International Journal of Software Engineering*, 21(4), 99–115.
- Herlambang, R., et al. (2023). Teaching collaborative programming using Git and GitHub. *Educational Computing Research*, 40(6), 55–72.
- ReadTheDocs: Automating and Hosting Your Documentation. Disponível em: <https://readthedocs.org>.
- Sphinx Documentation Generator. Disponível em: <https://www.sphinx-doc.org>.
- GitHub, Bitbucket, and GitLab Integration with ReadTheDocs. Disponível em: <https://docs.readthedocs.io>.
- Galimullin, I. (2023). Python and its evolution: A comprehensive review. *Journal of Programming Languages*, 10(3), 45–60.
- Giapros, T. (2022). Versatility of Python in modern software development. *Tech Innovations*, 8(1), 12–19.

- Kelly, T. (2015). The Python advantage: Simplicity and power in software design. *Software Engineering Today*, 6(2), 23–31.
- Khan, A., Ahmed, S., & Malik, R. (2023). Automation with Python: Trends and insights. *Global Technology Review*, 15(2), 100–110.
- M. Nivethitha, & Sarathambekai, P. (2019). Libraries for data analysis in Python. *Data Science Applications*, 5(2), 67–74.
- Peta, J. (2022). Python in practice: Benefits and challenges. *Open Source Journal*, 18(1), 89–97.
- Seyidova, N., & Shakhayev, I. (2023). Python frameworks for web development: An overview. *Digital Horizons*, 7(4), 55–70.
- "Python for web development: Tools and frameworks." (2022). *Tech Trends*.
- BACALHAU, J. Missing Data Analysis and Imputation Method for Medium Voltage Distribution Network Feeders, *Dissertação (Mestrado em Engenharia Elétrica) - Universidade Federal do Espírito Santo*. Vitória, p. 92, 2020.