Serviço Web para Imputação de Dados em Séries Temporais Univariadas

Jeremias Lima Abreu[1], **Douglas Almeida Vidal**[1], **Glauco Estacio Gonçalves**[1]

[1]Instituto de Tecnologia – Universidade Federal do Pará (UFPA)

jeremias.abreu@itec.ufpa.br, vidalstm998@gmail.com, glaucogoncalves@ufpa.br

Abstract. *A common problem in environmental data recording is gaps, which can be mitigated by data imputation. Although there are computational libraries for imputation, these require learning new technologies and programming languages. To meet this need, this work presents a web service for imputing time series data, which allows a client application to request imputations on data with gaps. In addition to describing the architecture of this service, this article shows results of a rigorous evaluation of service performance in terms of imputation time and quality, in order to provide a global view of the trade-off between execution time and accuracy of each algorithm.*

Resumo. *Um problema comum na coleta de dados ambientais são as lacunas, as quais podem ser mitigadas pela imputação de dados. Embora haja bibliotecas computacionais para imputação, estas exigem o aprendizado de novas tecnologias e programação. Para suprir esta necessidade, este trabalho apresenta um serviço web para imputação de dados em séries temporais, o qual permite que uma aplicação cliente solicite imputações em dados com lacunas. Além da descrição da arquitetura deste serviço, este artigo mostra resultados de uma rigorosa avaliação de desempenho do serviço em termos de tempo e qualidade da imputação, de modo a fornecer uma visão global do compromisso entre o tempo de execução e a precisão de cada algoritmo.*

1. Introdução

O avanço da tecnologia possibilitou a criação de dispositivos para realizar a coleta de dados com as mais diversas finalidades. Como exemplo, pode-se citar as estações meteorológicas que coletam informações do tempo da região onde está instalada, definindo uma série temporal dos parâmetros registrados [Huang et al. 2020].

No processo automático de coleta de dados, um problema comum são as falhas na coleta de amostras, que podem ser provenientes de diversos fatores como o desgaste dos dispositivos que realizam os registros ou problemas de conexão com as bases de dados, levando à perda de dados [Addi et al. 2022]. Um estudo da rede de estações meteorológicas do Instituto Nacional de Meteorologia (INMET) do ano de 2017 [Bezerra et al. 2019], por exemplo, mostrou que das 490 estações no Brasil naquele ano, 25% apresentavam falhas ao menos uma vez na semana. Tal ausência dos dados coletados pode impactar processos que dependam da completude do conjunto de dados. No contexto anteriormente mencionado, a ausência de dados proveniente de estações meteorológicas poderia, por exemplo, impedir a emissão de um boletim meteorológico de uma região, causando deficiências nos processos agrícolas naquele período de tempo.

Para mitigar os problemas decorrentes da ausência de dados, pode-se empregar técnicas de imputação de dados, que consiste em estimar uma informação faltante em uma coletânea de dados e substituir os elementos em falta com os elementos fictícios estimados (imputados) [Dhevi 2014], a fim de reduzir os efeitos que seriam ocasionados pelo dado ausente durante a etapa de análise.

Atualmente, várias soluções e bibliotecas computacionais, como a *missingpy*[1], estão disponíveis para aplicar métodos de imputação a dados faltantes. No entanto, o uso dessas soluções muitas vezes exige conhecimento avançado em programação e tempo significativo para compreender a tecnologia e adaptá-la às análises específicas. Além disso, o desempenho computacional pode se tornar um obstáculo, dependendo da complexidade dos métodos de imputação e da quantidade de dados.

Como alternativa, este trabalho propõe a construção de um Serviço Web para Imputação de Dados em Séries Temporais Univariadas. Essa solução visa simplificar a integração para os usuários, oferecendo escalabilidade para lidar com desafios de desempenho. No melhor do conhecimento dos autores, não foram identificados trabalhos que apresentem serviços semelhantes para imputação de dados.

Este artigo está estruturado da seguinte forma: a Seção 2 apresenta as especificações que serão atendidas pelo sistema, tal como detalhes da arquitetura e da implementação do serviço. A seção Seção 3 apresenta o conjunto de dados e o ambiente de experimentação utilizados nos dois experimentos de avaliação do serviço, cujos detalhes e resultados são discutidos na Seção 4 e Seção 5. Por fim, a Seção 6 apresenta conclusões do trabalho e sugere futuras direções.

2. Serviço web para Imputação de Séries Temporais Univariadas

Essa seção especifica os requisitos do serviço proposto (Subseção 2.1) e também discute sua arquitetura (Subseção 2.2) e implementação (Subseção 2.3).

2.1. Especificação

O serviço web proposto segue o padrão de projeto conhecido como *Resource API* [Daigneau 2011] e adere aos princípios REST (*REpresentational State Transfer*). Neste padrão de projeto utiliza-se o protocolo HTTP (*HyperText Transfer Protocol*) para comunicação entre cliente e servidor. Ao usá-lo, busca-se beneficiar-se ao máximo dos recursos disponíveis no protocolo HTTP para prover resultados ao cliente de forma padronizada e otimizando o uso de rede. Todas as operações de requisição (métodos) e códigos de retorno (status) citados nessa seção são referências aos métodos e status HTTP.

Tendo em vista que o tempo para imputação pode variar dependendo do tamanho da série temporal e do algoritmo escolhido, o serviço proposto utiliza o padrão de projeto *Request/Acknowledge* [Daigneau 2011] para desacoplar o pedido de imputação da sua resposta. Assim, o serviço disponibiliza duas operações possíveis: a criação de imputação e a recuperação de dados de uma imputação. A primeira operação permite solicitar a imputação de uma série temporal usando um algoritmo de imputação escolhido pelo cliente, dentre os disponíveis no serviço. Esta operação posterga a realização da imputação e entrega ao cliente um identificador que pode ser usado para a segunda operação, que permite a recuperação dos dados imputados.

[1] `https://pypi.org/project/missingpy/`

Outro requisito fundamental deste serviço é que o serviço seja capaz de receber novas funcionalidades sem comprometer o funcionamento de clientes anteriormente implementados. A versão atual do serviço oferece a opção de 12 algoritmos de imputação para séries univariadas: *mean*, *median*, *mode*, *random*, *most frequent*, *linear interpolation*, *spline interpolation*, *barycentric interpolation*, *polynomial interpolation*, *locf*, *nocb* e *normal unit variance*. A adição de novos algoritmos ou a melhoria de um dos existentes não deve exigir nenhuma modificação no cliente. Tal requisito impacta a organização da operação de criação de imputação, que deve permitir a expansão da quantidade de algoritmos de imputação, incluindo algoritmos que precisam de parâmetros extras, sem a necessidade de alterações profundas no cliente.

Um último requisito deste serviço é que ele seja capaz de atuar com séries temporais longas. Enquanto a operação de imputação deve receber toda a série com os dados e lacunas, a operação de recuperação deve oferecer um meio para retornar somente os dados imputados, diminuindo a quantidade de dados transmitida na resposta.

2.2. Arquitetura

Na arquitetura do sistema proposto, mostrada na Figura 1, destacam-se 2 componentes principais: o *Imputation Creator and Retriever* e o *Imputation Processor*. Enquanto o primeiro comunica-se com o Cliente e trata as requisições de imputação e consulta, o segundo é responsável por efetivamente realizar a imputação, conforme consome as requisições de uma fila. O Banco de Dados é usado para armazenar a série temporal imputada e metadados adicionais, os quais podem ser consultados a qualquer tempo.

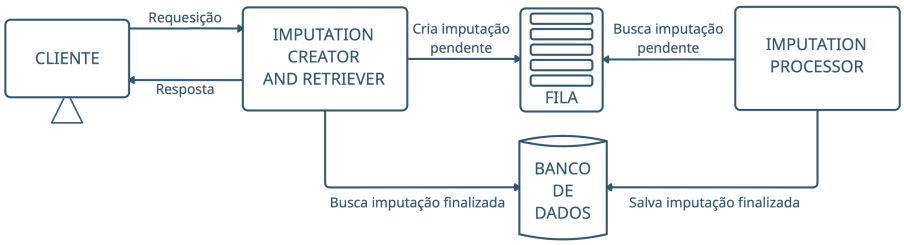

Figura 1. Arquitetura do serviço web

O *Imputation Creator and Retriever* analisa requisições, determinando o tipo da operação desejada (criação ou recuperação) com base na rota utilizada. A requisição de criação de imputação é feita usando o método POST na rota */imputation*. O corpo da requisição deve seguir a estrutura mostrada no Listing 1, onde a série temporal é passada como valor da chave *time_series* no formato de um *array* contendo números reais e *null* na posição dos valores faltantes. O algoritmo pode ser escolhido através do campo *method* no formato de uma *string*. E, por fim, o campo *order* aceita números inteiros no intervalo $[1, 5]$, que são limites típicos dos algoritmos usados.

Listing 1. Estrutura do JSON aceito na criação de imputação

```
{
    "time_series": [0, null],
    "method": "string",
    "order": 0
}
```

Quando a requisição recebida é válida (conforme a sintaxe e restrições acima), o *Imputation Creator and Retriever* armazena a série temporal no banco de dados, junto a um identificador único gerado pelo serviço, o algoritmo escolhido e seus parâmetros. Além disso, uma resposta com o status HTTP 201 CREATED é enviada. No corpo da resposta, o cliente receberá, no formato JSON, o identificador único da requisição. Caso a requisição seja inválida, o serviço responde com status HTTP 400 BAD REQUEST, contendo em seu corpo uma mensagem de erro informando o problema da requisição.

Com o identificador recebido na requisição de criação, o cliente pode, a qualquer tempo, verificar o status do processo de imputação e recuperar a série imputada. Para tanto, o cliente deve fazer uma requisição GET para a rota */imputation/<id>*, onde o chave *<id>* é preenchida com o identificador recebido na resposta da requisição de criação.

Ao receber uma requisição de recuperação, o *Imputation Creator and Retriever* verifica a validade do identificador fornecido. Caso o identificador não seja válido, uma resposta de erro com status *404 NOT FOUND* é retornada. Se a solicitação existe, uma resposta com status *200 OK* é retornada contendo o status da imputação - criada, em processamento, finalizada ou erro. Uma requisição apenas criada está aguardando para ser processada. Neste caso, o corpo da resposta *200 OK* contém o campo *status* com o valor *created*, e os campos *imputed_data* e *imputed_indexes* são *arrays* vazios. Já o status *processing* indica que a imputação já está em andamento, mas não concluída.

Quando o processo de imputação já foi concluído, o status da resposta é *finished*. O campo *imputed_indexes* contém um *array* de inteiros indicando os índices nas posições da série temporal original onde os valores foram imputados. A série temporal completa após a imputação pode ser encontrada no campo *imputed_data*. Para diminuir a quantidade de dados transferidas na resposta, o cliente pode optar por receber apenas os dados imputados, e não toda a série, usando o parâmetro *onlyImputed* na requisição GET.

Finalmente, caso haja um erro irrecuperável durante a imputação da série, o campo *status* na resposta será *error*. O campo *message* dentro de *error* conterá informação sobre o erro ocorrido. Os campos *imputed_data* e *imputed_indexes* têm arrays vazios.

O *Imputation Processor* atua de forma independente do *Imputation Creator and Retriever* sendo responsável por coletar as requisições de imputação da fila e processá-las aplicando o algoritmo e os parâmetros indicados pelo cliente. A Figura 2 apresenta um diagrama do funcionamento do *Imputation Processor* em conjunto com a fila de requisições. Utilizando o monitoramento por *pooling*, cujo intervalo de tempo de consulta é um parâmetro de configuração do serviço web, o *Imputation Processor* consulta a fila por imputações pendentes. Após receber os dados de uma imputação, a *Imputation Processor* processa a série temporal preenchendo cada uma das lacunas da série. Após a imputação da série temporal, o banco de dados é atualizado com a série temporal imputada. Ao longo deste processo, o status da requisição no banco é atualizado pelo *Imputation Processor*, passando de *created* para *processing* no momento em que é retirada da fila, e em seguida para *finished*, quando as imputações são salvas no banco de dados.

2.3. Implementação

A linguagem de programação Python 3.10.8 foi selecionada como a linguagem de desenvolvimento do serviço devido à ampla variedade de bibliotecas disponíveis, especialmente para manipulação de dados [Kadiyala and Kumar 2017]. As bibliotecas AutoIm-

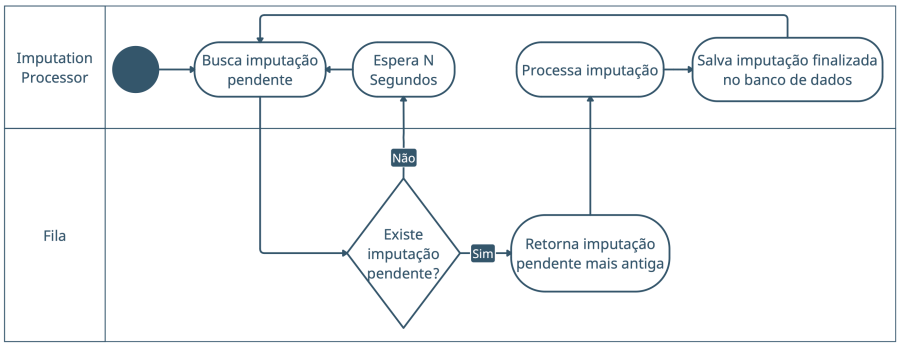

Figura 2. Diagrama de atividades do processamento de imputação

pute 0.13.0 e Scikit-learn 1.1.3 foram escolhidas para implementação dos algoritmos de imputação, já que são amplamente utilizadas em trabalhos que fazem uso da imputação de séries univariadas [Moritz and Bartz-Beielstein 2017] [Flores et al. 2019]. O código completo do serviço de imputação está disponível no GitHub[2].

O *framework* Flask 2.1.2 [Idris et al. 2020] foi adotado para implementar a API do serviço de imputação devido à sua simplicidade e robustez. Esta biblioteca oferece mecanismos de alto nível para tarefas comuns em aplicações web, como criação de rotas e manipulação de dados nas requisições, simplificando o desenvolvimento da API. Para o armazenamento de dados, o MongoDB 4.4.4 foi escolhido por ser um banco de dados de código aberto focado em desempenho [Chauhan 2019], com suporte oficial para Python.

A integração entre os serviços usa a abordagem arquitetônica de microsserviços [Larrucea et al. 2018], em que cada serviço possui única e bem definida responsabilidade e pode ser desenvolvido, testado e implantado independentemente. Esta arquitetura permite ainda escalar os serviços de forma independente, sendo possível implantar o serviço no hardware mais adequado para as tarefas que este realiza.

3. Conjunto de Dados e Ambiente de Testes

Este artigo apresenta resultados da avaliação de qualidade dos algoritmos usados para imputação (Seção 4), assim como a avaliação de performance do sistema proposto (Seção 5). Estes experimentos têm dois principais objetivos: caracterizar o impacto das estratégias de projeto e implementação no desempenho geral do serviço; e dar ao leitor uma visão acerca de como o serviço e os algoritmos suportados podem ser utilizados na prática.

Dados meteorológicos do INMET[3] (Instituto Nacional de Meteorologia) foram escolhidos para serem usados tanto nos testes de performance do serviço web quanto para os testes de qualidade dos algoritmos de imputação. Esses dados consistem de coletas diárias da temperatura média em Belém-PA, com 112 amostras no total, que foram registradas do dia 01/08/2022 ao dia 20/11/2022.

Essa série temporal foi escolhida por não possuir nenhuma lacuna originalmente, ou seja, todas as 112 amostras são medições reais. Deve-se observar que a busca desta série envolveu observar os dados de toda a série temporal de 2022 em busca de um trecho

[2]Repositório do trabalho: `https://github.com/j-abreu/imputation-web-service`
[3]https://portal.inmet.gov.br/

contínuo sem lacunas, e este período de Agosto de 2022 a Novembro de 2022 foi o de maior quantidade de dias sem lacunas encontrado. Para simular falhas na amostragem dos dados, gerou-se uma segunda versão da série temporal com aproximadamente 10% das amostras substituídas por valores nulos, os dias foram removidos de forma aleatória seguindo uma distribuição uniforme, de modo que cada dia tem a mesma probabilidade de ser removido. Desta forma, a série com dados originais pode ser usada para comparar a qualidade dos diferentes algoritmos de imputação.

Ambos os testes foram realizados em ambientes controlados, usando instâncias EC2[4] da AWS[5]. Para a realização da avaliação de desempenho do serviço web foram usadas duas instâncias EC2 idênticas e conectadas por uma rede virtual privada (*Virtual Private Network* - VPC[6]) para eliminar qualquer influências que pudesse ser causada pela presença de tráfego cruzado na rede. Todas as máquina utilizadas possuíam as seguintes características: Processador Intel Xeon Platinum 8000 com 2 núcleos, 8GB de memória RAM, SSD de 16GB, Sistema operacional Ubuntu 20.04

4. Avaliação dos Algoritmos de Imputação

Para realizar os testes de qualidade, foi usado como fator o algoritmo de imputação, com os níveis correspondendo aos algoritmos: *mean*, *median*, *most frequent*, *linear interpolation*, *spline interpolation*, *barycentric interpolation*, *polynomial interpolation*, *mode*, *random*, *locf*, *nocb* e *normal unit variance*. Para os algortimos *spline interpolation* e *polynomial interpolation*, que necessitam do parâmetro *order*, escolheu-se o valor 3 para os testes, o qual indicou o melhor ajuste (menor erro médio) para este conjunto de dados.

Com a série temporal original e sua versão com lacunas, foi possível criar uma rotina que usa os algoritmos para imputar os dados faltantes e calcula o erro entre a série temporal com lacunas e a série temporal original. O cálculo do erro usou o Erro Absoluto Médio (MAE - *Mean Absolute Error*). Além do MAE, foram calculados a média e o desvio padrão do tempo de execução para um total de 100 execuções para cada algoritmo.

A Tabela 1 sumariza os resultados do experimento. A primeira coluna indica o nome do algoritmo; a segunda apresenta o MAE do respectivo algoritmo; a terceira mostra o tempo médio em milissegundos ($\bar{x}$) de uma imputação de cada algoritmo; e, por fim, a última coluna mostra o desvio padrão (s) no tempo para cada algoritmo. Os resultados do MAE mostram que, em geral, os algoritmos tem comportamentos próximos, com exceção do algoritmo *barycentric interpolation*. Dentre os demais, destaca-se o algoritmo de interpolação linear, que obteve o menor MAE. No geral, o MAE revela que, para a maioria dos algoritmos, o erro está próximo de meio grau Celsius.

No geral, os algoritmos exibem baixo desvio padrão do tempo de execução, indicando concentração dos tempos em torno da média. Apesar do *normal unit variance* apresentar o maior desvio padrão sua média também é significativamente maior, de modo que o coeficiente de variação é próximo aos dos demais algoritmos. Desta forma, pode-se usar a média do tempo de execução com segurança estatística no restante da análise.

Na métrica de tempo médio de execução, o algoritmo *normal unit variance* re-

[4]https://aws.amazon.com/ec2/

[5]https://aws.amazon.com/

[6]https://aws.amazon.com/vpc/

Tabela 1. Resultado dos testes de qualidade.

Algoritmo	**MAE**	$\bar{x}$ **(ms)**	s **(ms)**
linear interpolation	0.39389	2.09985	0.09569
most frequent	0.46771	0.46978	0.03457
mode	0.46771	1.74368	0.03434
locf	0.47449	1.81909	0.02869
median	0.54618	0.62262	0.02913
polynomial interpolation	3.0	0.54806	0.06367
spline interpolation	3.0	0.58265	0.05338
mean	0.62909	0.46683	0.04492
nocb	0.79088	1.83128	0.02193
random	0.78318	1.83878	0.03628
normal unit variance	0.92703	6.66171	0.30798
barycentric interpolation	4.5×10^{13}	3.22195	0.06590

gistra o maior tempo médio de execução, com pouco mais de 6,5 milissegundos, seguido pelos algoritmos de interpolação, que variam entre 2 e 4 milissegundos. Os demais algoritmos têm tempos médios de execução inferiores a 2 milissegundos, destacando-se os algoritmos *mean* e *most frequent* com 0,46683 e 0,46978 milissegundos, respectivamente.

Deve-se notar que, embora os algoritmos *most frequent* e *mode* apresentem o mesmo MAE (a imputação pela moda ou pelo valor mais frequente é, na realidade, o mesmo algoritmo), o tempo médio de *mode* é quatro vezes maior que o de *most frequent*. Isto ocorre porque estes algoritmos são implementados por bibliotecas distintas, sendo o *most frequent* implementado pela biblioteca Scikit-learn e o *mode* pela AutoImpute.

A análise conjunta do MAE e do tempo médio de execução, representada no gráfico de dispersão da Figura 3, destaca o compromisso entre qualidade e desempenho. O algoritmo de interpolação linear, com o menor MAE neste caso, tem um tempo de execução médio quase cinco vezes maior que o algoritmo *most frequent*. Esse compromisso sugere que, em larga escala, o usuário do serviço de imputação deve considerar esse equilíbrio entre qualidade da imputação e tempo de processamento, especialmente ao optar por algoritmos mais precisos que demandam mais recursos computacionais.

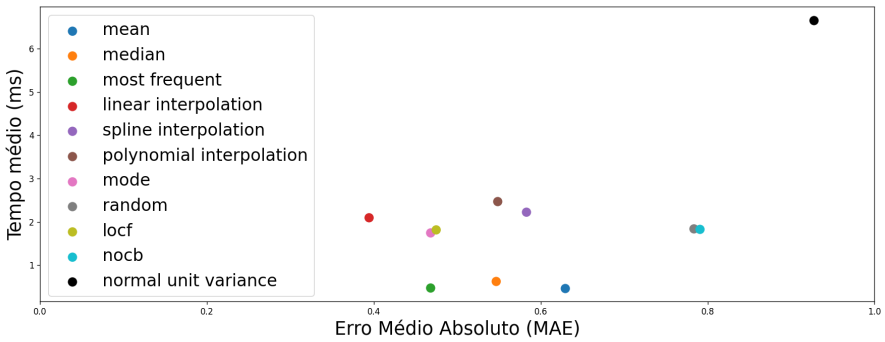

Figura 3. Gráfico de dispersão do MAE e do Tempo médio para cada algoritmo de imputação avaliado

5. Avaliação do Serviço Web

O serviço web proposto neste trabalho foi desenvolvido para ser escalável e manutenível. Além desses aspectos, é esperado que o sistema também apresente uma boa performance. Por isso, o serviço foi submetido à testes de desempenho em diferentes cenários.

Para geração das requisições foi utilizado o software JMeter[7] 5.5, uma ferramenta de código aberto capaz de realizar testes em sistemas cliente-servidor. Foi definido um plano de teste simulando o comportamento de usuários da seguinte forma: requisições HTTP distribuídas uniformemente ao longo de 30 segundos eram feitas para a criação de imputação e, para cada uma dessas, eram feitas requisições para recuperação dos dados enquanto os dados imputados não fossem retornados ou enquanto não fosse informado que houve um erro. Quando a resposta de uma requisição retornava os dados imputados ou informava que houve um erro, o fluxo para aquele usuário do teste era finalizado.

Como fatores do experimento, foram escolhidos o número total de usuários e o algoritmo de interpolação. Cada usuário neste experimento faz apenas uma solicitação para criação de imputação e repete solicitações de recuperação de dados, até que o processo de imputação tenha terminado e a série temporal seja retornada após imputação. Assim, o fator número total de usuários define quantos usuário simultâneos acessam o sistema. Para esse fator foram definidos 5 diferentes níveis, são eles: 1000, 2000, 4000, 6000 e 10000 usuários. O segundo fator diz repeito ao algoritmo escolhido na requisição, que são: *mean*, *median*, *most frequent*, *linear interpolation*, *spline interpolation*, *barycentric interpolation*, *polynomial interpolation*, *mode*, *random*, *locf*, *nocb* e *normal unit variance*. Para os algoritmos *polynomial interpolation* e *spline interpolation* foi definido como 3 o valor padrão para o parâmetro *order*, como no experimento anterior.

A Figura 4(a) e a Figura 4(b) mostram diagramas de caixa (*boxplot*) do tempo em milissegundos para as requisições de criação e recuperação de imputações, respectivamente. Em ambos os casos, o algoritmo *normal unit variance* não foi incluído para facilitar a visualização, dado que obteve uma mediana próxima de 175 milissegundos, para criação, e 120 milissegundos para recuperação. Nos demais algoritmos, os tempos de criação ficaram entre 14 e 20 milissegundos, enquanto que para recuperação a maioria dos resultados está entre 17 e 19 milissegundos, com alguns outliers, como no *Most Frequent* com 16 milissegundos e no *spline interpolation* com 21 milissegundos.

Apesar do *normal unit variance* ter tempos mais elevados em comparação aos outros algoritmos, todos eles demonstram tempos de resposta eficientes para ambos os tipos de requisição, com médias abaixo de 200 milissegundos. Esse desempenho é atribuído à independência dos componentes do sistema. O microsserviço para criação e recuperação de imputações realiza operações leves de leitura ou escrita no banco de dados, sem depender do tempo de processamento da imputação em si. Dessa forma, o serviço web proporciona respostas instantâneas, mesmo sob grande carga de acessos simultâneos. A Figura 5 ilustra este ponto, apresentando o diagrama de caixa do tempo médio de resposta pras requisições de recuperação agrupados por número de usuários. No gráfico é possível o tempo de resposta manteve-se estável com o aumento do número de usuários.

Vale salientar que a precisão da ferramenta usada para a realização dos testes de performance é da ordem de milissegundos, e que o teste feito com 2000 usuários

[7]https://jmeter.apache.org/

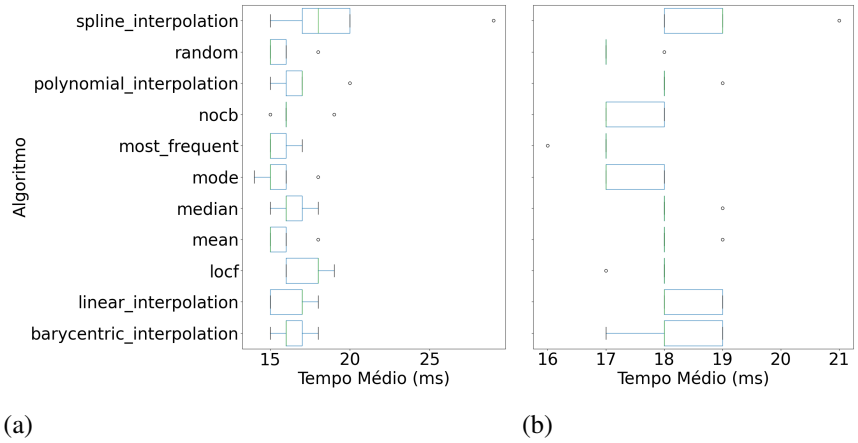

Figura 4. Tempo de resposta em milissegundos para requisições de: (a) criação e (b) recuperação de imputação, agrupados por algoritmo.

apresentou resultados apenas com os valores 17, 18 e 19 milissegundos, sendo 18ms o valor mais frequente, gerando assim um gráfico de caixa com apenas uma linha em 18ms e dois pontos, como é possível ver na Figura 5.

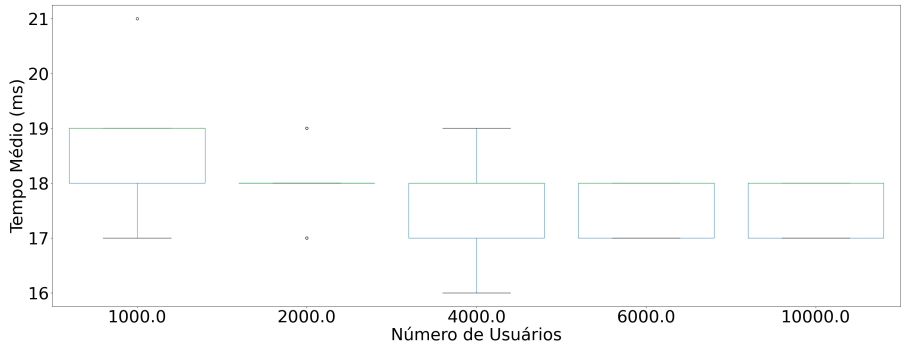

Figura 5. Tempo de resposta da recuperação de dados por número de usuários

6. Conclusão

Neste trabalho, foi proposto um serviço web REST para imputação de dados em séries temporais univariadas, permitindo a utilização de diversos algoritmos. O sistema foi construído seguindo padrões de projeto de engenharia de software reconhecidos, buscando garantir escalabilidade, manutenibilidade e desempenho.

O trabalho contribui à comunidade científica com um software open-source para imputação em séries temporais univariadas via web, o qual pode ser utilizado em sistemas e aplicações de engenharia e análise de dados que lidam com dados faltantes.

Para avaliar o desempenho, um experimento fatorial considerou o número de usuários e algoritmo de imputação. O serviço respondeu as requisições em menos de meio segundo para até 10.000 usuários. Além disso, os algoritmos foram avaliados quanto ao erro médio absoluto e tempo médio de execução. Onde se viu que diferentes implementações do mesmo algoritmo podem ter a mesma precisão, mas com tempo de processamento diferentes, como foi o caso dos algoritmos *mode* e *most frequest*. Isto ressalta a importância

de uma escolha adequada de bibliotecas para implementação de serviços, tendo em vista seu impacto no desempenho em um cenário de larga escala.

Destaca-se a escalabilidade da aplicação, desenvolvida em microserviços, permitindo seu uso em diferentes servidores. Como perspectivas futuras, propõe-se estender o serviço para lidar com a imputação em séries temporais de múltiplas variáveis e a agregação de novos algoritmos de imputação.

Agradecimentos

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

Referências

Addi, M., Gyasi-Agyei, Y., Obuobie, E., and Amekudzi, L. K. (2022). Evaluation of imputation techniques for infilling missing daily rainfall records on river basins in ghana. *Hydrological Sciences Journal*, 67(4):613–627.

Bezerra, D., Junior, J., Gonçalves, G., and Medeiros, V. (2019). Avaliação de disponibilidade de estações de medição meteorológica. In *Anais do X Workshop de Computação Aplicada a Gestão do Meio Ambiente e Recursos Naturais*, pages 1–10, Porto Alegre, RS, Brasil. SBC.

Chauhan, A. (2019). A review on various aspects of mongodb databases. *International Journal of Engineering Research & Technology (IJERT)*, 8(5).

Daigneau, R. (2011). *Service Design Patterns: fundamental design solutions for SOAP/WSDL and restful Web Services*. Addison-Wesley.

Dhevi, A. S. (2014). Imputing missing values using inverse distance weighted interpolation for time series data. In *2014 Sixth international conference on advanced computing (ICoAC)*, pages 255–259. IEEE.

Flores, A., Tito, H., and Silva, C. (2019). Local average of nearest neighbors: Univariate time series imputation. *International Journal of Advanced Computer Science and Applications*, 10(8).

Huang, Z.-Q., Chen, Y.-C., and Wen, C.-Y. (2020). Real-time weather monitoring and prediction using city buses and machine learning. *Sensors*, 20(18):5173.

Idris, N., Foozy, C. F. M., and Shamala, P. (2020). A generic review of web technology: Django and flask. *International Journal of Advanced Science Computing and Engineering*, 2(1):34–40.

Kadiyala, A. and Kumar, A. (2017). Applications of python to evaluate environmental data science problems. *Environmental Progress & Sustainable Energy*, 36(6):1580–1586.

Larrucea, X., Santamaria, I., Colomo-Palacios, R., and Ebert, C. (2018). Microservices. *IEEE Software*, 35(3):96–100.

Moritz, S. and Bartz-Beielstein, T. (2017). imputets: time series missing value imputation in r. *R Journal*, 9(1):207.