Desenvolvimento de um *Pipeline* Completo para Coleta, Modelagem Preditiva e Visualização de Dados Meteorológicos do INMET

Ana C. Gomes[1], Gabriel Albuquerque[1], Gustavo Mourato[1], Paulo H. Rosado[1], Thomaz R. Lima[1], Sophia Gallindo[1], Vinícius de Andrade[1],

[1]CESAR School

{acgs,gmca,gmam,phrf,trl,sagp,vaj}@cesar.school

Abstract. *This paper presents the development of a complete end-to-end pipeline for analyzing meteorological data from INMET and uncovering meaningful climatic patterns across the state of Pernambuco. Beyond the architectural implementation, the project emphasizes the extraction of environmental insights from variables such as temperature, humidity and wind speed, enabling the identification of distinct regional behaviors. The analysis reveals a strong climatic contrast between the semi-arid conditions of the Sertão and the milder, more humid environments of the Agreste and Zona da Mata. By structuring a workflow that integrates data ingestion, processing, feature engineering and predictive modeling, alongside experiment tracking with MLflow, the system supports the interpretation of long-term trends such as persistent heat waves, chronic dry-air exposure, wind corridors and humidity saturation zones. These findings are consolidated and made accessible through interactive dashboards on the ThingsBoard platform, allowing stakeholders to visualize both raw observations and predictive inferences to better understand local climatic dynamics.*

Resumo. *Este trabalho apresenta o desenvolvimento de um pipeline completo para análise de dados meteorológicos do INMET, com ênfase não apenas na arquitetura técnica, mas principalmente na interpretação dos padrões climáticos observados em Pernambuco. A partir das variáveis de temperatura, umidade e velocidade do vento, o sistema permite revelar contrastes marcantes entre regiões, destacando o calor extremo e a secura persistente no Sertão em oposição ao clima mais úmido, estável e ventoso do Agreste e da Zona da Mata. O fluxo de trabalho integra ingestão, tratamento, engenharia de atributos e modelagem preditiva, bem como o registro de experimentos no MLflow, possibilitando a identificação de tendências relevantes, como amplitudes térmicas elevadas, longos períodos de ar seco, regimes de vento constantes e zonas de saturação de umidade. A visualização final no ThingsBoard consolida esses achados, oferecendo dashboards interativos que auxiliam na compreensão das dinâmicas climáticas regionais e no suporte a análises ambientais mais aprofundadas.*

Sumário

1. Introdução

A variação da temperatura ao longo do dia é um dos fatores mais importantes para o entendimento das condições climáticas e para a tomada de decisão em setores que dependem fortemente do clima, como agricultura, energia e planejamento urbano. Prever a temperatura de cada hora com precisão ajuda a antecipar mudanças bruscas, reduzir riscos operacionais e oferecer informações mais confiáveis para análises ambientais e estudos meteorológicos.

Neste projeto, o objetivo central é estimar a temperatura horária a partir de dados reais coletados das estações automáticas do INMET. A escolha dessa problemática se justifica pela forte relação entre a temperatura e outras variáveis meteorológicas, com destaque para a umidade relativa do ar, velocidade do vento e a própria temperatura registrada previamente. Essas variáveis influenciam diretamente a dinâmica térmica da atmosfera e permitem a construção de modelos preditivos capazes de capturar padrões sazonais, variações diárias e comportamentos característicos do clima de Pernambuco.

Com base no conjunto de dados disponibilizado, foi estruturado um processo analítico que inclui preparação dos dados, interpretação dos padrões identificados e construção de um modelo preditivo. O foco do estudo está na capacidade de identificar como cada variável contribui para o comportamento da temperatura ao longo do tempo e de que forma essas relações podem ser utilizadas para gerar previsões confiáveis.

2. Descrição analítica do *dataset*

O *dataset* utilizado neste projeto contém informações meteorológicas coletadas pelas estações automáticas do INMET, incluindo tanto metadados das estações quanto observações horárias das variáveis climáticas. Os identificadores do *dataset* descrevem a localização e as características estruturais de cada estação:

- **Região**: região geográfica onde a estação está localizada;
- **UF**: unidade federativa correspondente;
- **Estação**: município de referência da estação meteorológica;
- **Código**: identificador único da estação;
- **Latitude e Longitude**: coordenadas geográficas da estação;
- **Altitude**: elevação da estação acima do nível do mar;
- **Data de fundação**: data em que a estação foi instalada.

As colunas de observação horária incluem medidas importantes para caracterização do clima, como data, hora e variáveis físicas registradas automaticamente. Entre elas, o *dataset* apresenta:

- **Data (AAAA/MM/DD)**: data da observação;
- **Hora UTC (HHMM UTC)**: horário de coleta;
- **Precipitação total (mm)**: quantidade de chuva acumulada na última hora;
- **Pressão atmosférica ao nível da estação (mB)**;

- **Pressão máxima e mínima da hora anterior (mB)**;
- **Radiação solar global (Kj/m²)**;
- **Temperatura do ar - bulbo seco (°C)**;
- **Temperatura do ponto de orvalho (°C)**;
- **Temperaturas máxima e mínima da hora anterior (°C)**;
- **Umidade relativa horária (%)**
- **Umidade relativa máxima e mínima na hora anterior (%)**;
- **Direção do vento** (°);
- **Velocidade horária do vento (m/s)**;
- **Rajada máxima (m/s)**.

2.1 Variáveis utilizadas no modelo

Para a construção da solução proposta, o projeto utilizou o seguinte subconjunto de colunas, reorganizado no código por meio do *column_mapping*:

Coluna Original	Variável (renomeada)
TEMPERATURA DO AR - BULBO SECO, HORARIA (°C)	***temperature***
UMIDADE RELATIVA DO AR, HORARIA (%)	***humidity***
VENTO, VELOCIDADE HORARIA (m/s)	***wind_speed***
Data	***timestamp***
Hora UTC	***time***

Tabela 1.0 - Variáveis a serem consideradas na análise.

2.1.1 Justificativa da escolha das colunas

A seleção das variáveis acima foi orientada pela relevância direta delas no comportamento térmico da atmosfera. As colunas escolhidas representam os principais elementos físicos que influenciam ou refletem a variação da temperatura ao longo do dia:

- **Temperatura do ar (*temperature*)** é a variável-alvo e referência principal para análise.
- **Umidade relativa (*humidity*)** afeta a taxa de resfriamento e aquecimento do ar, influenciando a temperatura percebida e real.
- **Velocidade do vento (*wind_speed*)** altera a troca de calor entre superfície e atmosfera, impactando diretamente oscilações térmicas.

- **Data (*timestamp*)** e **hora (*time*)** são essenciais para reconstruir a série temporal e capturar padrões diários e sazonais.

Esse conjunto de variáveis permite modelar de forma consistente as dinâmicas atmosféricas que condicionam a temperatura horária, gerando previsões mais assertivas e acuradas.

3. Arquitetura da Solução

A arquitetura do projeto integra os principais serviços necessários ao fluxo de ingestão, processamento e visualização dos dados. A aplicação *FastAPI* realiza a coleta e envia os arquivos brutos para o *AWS S3*. Esses dados são então estruturados no *Neon PostgreSQL*, onde ficam disponíveis para análise. O *JupyterLab* é utilizado para explorar, tratar, treinar e testar o modelo preditivo, enquanto o *MLflow* registra experimentos, métricas e artefatos, contando com um *PostgreSQL* local para metadados e um volume *Docker* para armazenamento dos modelos. Por fim, o *ThingsBoard* apresenta os resultados e predições por meio de *dashboards* interativos. Essa combinação de serviços garante um *pipeline* completo, do armazenamento bruto à visualização final.

Os principais serviços orquestrados via *Docker* e suas funções são:

Container	Porta	Função
FastAPI	8060	*API REST* para ingestão e processamento de dados
ThingsBoard	9090	Plataforma *IoT* utilizada para visualização e construção de *dashboards*
PostgreSQL (local)	5432	Banco de dados relacional usado pelo *MLflow* para metadados
JupyterLab	8888	Ambiente interativo para análise exploratória e *notebooks*
MLflow	5000	Serviço de tracking de experimentos e versionamento de modelos

Tabela 2.0 - Serviços Orquestrados

O *JupyterLab* funciona como o ambiente central de desenvolvimento e análise: é nele que os dados estruturados são consultados, tratados e onde o modelo de previsão de temperatura horária é treinado. O *MLflow*, por sua vez, registra parâmetros de treino, métricas e versões de modelo, utilizando o *PostgreSQL* em container para armazenar os metadados e um volume *Docker* dedicado (*/mlflow/artifacts*) para salvar os artefatos de *ML*.

O *ThingsBoard* é o responsável pela camada de visualização, recebendo dados e/ou predições para montagem de *dashboards* interativos.

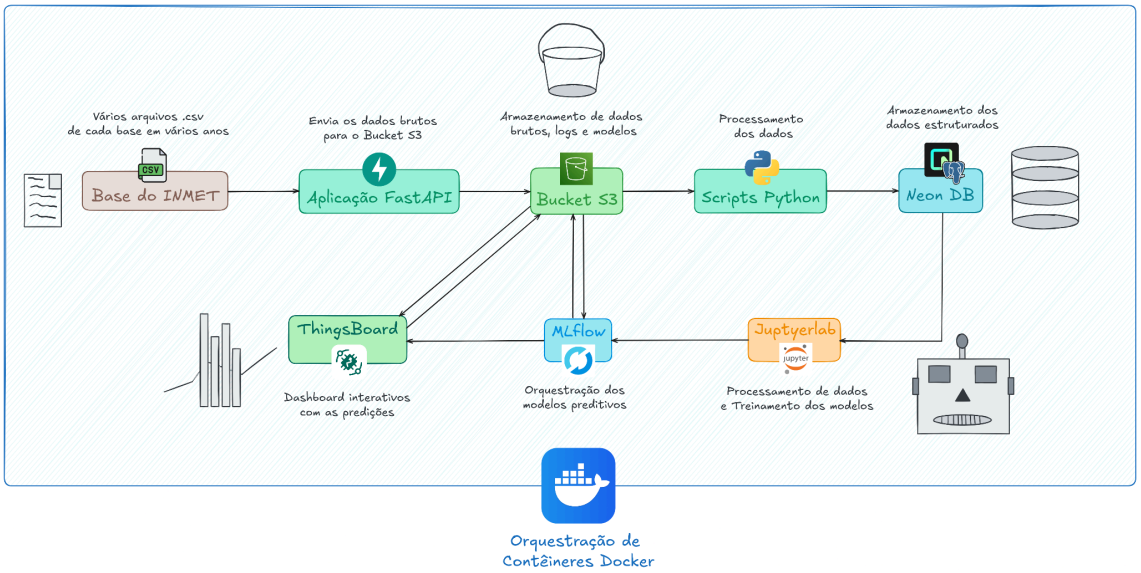

Imagem 1.0 - Diagrama de Arquitetura

3.1 Camadas de armazenamento

A solução utiliza diferentes camadas de armazenamento, separando dados brutos, dados estruturados e artefatos de machine learning:

Tipo	Tecnologia	Localização
Arquivos brutos	***AWS S3***	Nuvem
Dados estruturados	***Neon PostgreSQL***	Nuvem (serverless)
Metadados do *MLflow*	***PostgreSQL***	Container local

Artefatos de *ML*	***Volume Docker***	Local *(/mlflow/artifacts)*

Tabela 3.0 - Camadas de Armazenamento

- Os arquivos brutos oriundos das estações do INMET são armazenados em um bucket único no *AWS S3*.
- Após o processamento inicial, os dados são carregados e organizados em tabelas no *Neon PostgreSQL*, que assume o papel de banco relacional na nuvem com dados estruturados, que também são salvos no *AWS S3*.
- O *MLflow* utiliza um *PostgreSQL* local para registrar execuções, parâmetros e métricas dos experimentos.
- Os artefatos de modelo (por exemplo, arquivos serializados) são armazenados em um volume *Docker*, garantindo persistência local entre reinicializações dos contêineres.

A opção por utilizar um único *bucket S3* para dados brutos foi motivada pelo volume relativamente pequeno de dados do projeto e pela necessidade de manter o custo da infraestrutura mais baixo. Separar em múltiplos buckets, embora possível, não traria ganho significativo de organização para o escopo atual, mas aumentaria a complexidade e o custo de gerenciamento na nuvem.

3.2 Fluxo de dados

De forma resumida, o fluxo de dados implementado segue a mesma lógica geral descrita na especificação do projeto com adaptações pontuais:

1. Ingestão: a *API FastAPI* recebe os dados meteorológicos do *INMET* e os armazena no *ThingsBoard* que o envia para o *bucket S3* através da *FastAPI*. Quando necessário, a própria API acessa esse *bucket*, recupera os dados já processados e os envia diretamente ao *ThingsBoard* para atualização imediata do *dashboard*.
2. Estruturação: a partir dos arquivos armazenados no *S3*, os dados são processados e carregados para o *PostgreSQL* em nuvem, onde são organizados em tabelas adequadas para consulta, análise e preparação posterior para modelagem.
3. Análise e modelagem: o *JupyterLab* acessa o banco estruturado, realiza o tratamento dos dados, cria as variáveis derivadas e treina o modelo de previsão da temperatura horária. Durante todo esse processo, o *MLflow* registra parâmetros, métricas e versões do modelo, utilizando o *Neon* para metadados e o volume local /mlflow/artifacts para armazenar os artefatos do modelo.
4. Visualização: sempre que o *MLflow* recebe uma nova execução, um monitor implementado no código detecta a atualização, acessa novamente o *bucket S3* para recuperar os dados processados e envia os resultados atualizados ao *ThingsBoard*. Dessa forma, o *dashboard* permanece continuamente sincronizado com os dados mais recentes e com as predições geradas pelo modelo.

3.3 Comparação com a arquitetura proposta e justificativas

Em relação à arquitetura mínima sugerida na especificação, a solução mantém todos os blocos conceituais exigidos, mas com algumas substituições tecnológicas e simplificações justificadas:

***Snowflake → Neon PostgreSQL**:*

A especificação recomenda o uso do *Snowflake* como *data warehouse* na nuvem para armazenar os dados tratados. No projeto, esse papel foi assumido pelo *Neon PostgreSQL*, um serviço de *PostgreSQL serverless*. A escolha foi feita principalmente por questões de custo (*Neon* oferece camada gratuita) e por manter a mesma lógica de "banco relacional em nuvem com dados estruturados".

MinIO/S3* e número de *buckets:

O requisito original prevê *MinIO* ou *AWS S3* para armazenamento de dados brutos e modelos. A implementação utiliza apenas um *bucket S3* para armazenar os arquivos brutos, enquanto os modelos e artefatos de *ML* são armazenados localmente em volume *Docker* e no mesmo *bucket* que os dados brutos. Essa decisão foi tomada porque:

- O volume de dados e modelos é reduzido;
- A separação em múltiplos *buckets* não traria benefício prático significativo;

***Jupyter Notebook → JupyterLab**:*

A especificação menciona *"Jupyter Notebook"* como ambiente de análise. A solução utiliza *JupyterLab*, que inclui o suporte a *notebooks*, mas oferece uma interface mais completa para organização de arquivos e execução de código, sem alterar o papel dessa camada na arquitetura.

ThingsBoard:

O requisito considera *ThingsBoard* com *Trendz* para visualização, como alternativas equivalentes. A solução utilizou *ThingsBoard* puro, atendendo à necessidade de possuir um *dashboard* interativo conectado ao *pipeline*. Essa escolha foi feita considerando que o *ThingsBoard* já possui ferramentas suficientes para a visualização e análise dos dados. Além disso, O *Trendz* tem triggers automáticos para quando acontecer uma atualização de dados do *ThingsBoard*, porém como não estamos trabalhando com gestão contínua de dados, não houve necessidade de utilizar o *plugin*.

Em síntese, a arquitetura implementada respeita o fluxo conceitual definido na especificação (ingestão → armazenamento em nuvem → base estruturada → modelagem → visualização), ao mesmo tempo em que adapta algumas tecnologias para

minimizar custo, reduzir complexidade desnecessária e aproveitar melhor os recursos disponíveis.

4. Modelagem

A modelagem foi estruturada a partir de um fluxo que começa no tratamento dos dados recebidos do banco, que já chegam previamente filtrados por um *pipeline* inicial contendo apenas as colunas de identificador, estação, data, hora, temperatura, umidade e velocidade do vento. A partir desse conjunto reduzido, inicia-se a criação de novas variáveis derivadas e segue-se até a fase de imputação e experimentação das diferentes configurações do modelo.

A primeira etapa consistiu no carregamento dos registros contendo os identificadores, as informações de data e hora separadas, a estação meteorológica e as variáveis de interesse. Como a hora vinha armazenada no formato *string "HHMM UTC"*, foi necessário extrair apenas os dois primeiros caracteres para obter a hora como valor inteiro. Um *timestamp* temporário foi criado internamente para ordenar corretamente a série temporal e extrair *features* derivadas, porém as colunas de data e hora foram mantidas separadas nos dados finais, preservando a estrutura original do banco de dados.

A partir do processamento temporal, foram criadas dez novas *features* derivadas, organizadas para capturar padrões sazonais e cíclicos nos dados meteorológicos. As primeiras representavam componentes tradicionais do tempo: ano, mês, dia, hora, dia da semana e dia do ano, permitindo ao modelo reconhecer ciclos naturais e comportamentos recorrentes. Em seguida, foram geradas versões cíclicas de mês e hora utilizando funções seno e cosseno (hora_sin, hora_cos, mes_sin, mes_cos), garantindo que valores extremos fossem tratados como vizinhos diretos — preservando, por exemplo, a continuidade entre 23 horas e 0 horas. Além das *features* temporais, foi aplicada uma codificação *One-Hot* da variável referente à estação meteorológica, gerando colunas binárias para cada localidade presente no *dataset*. Essa combinação de *features* temporais e espaciais fornece contexto suficiente para que o modelo de imputação estime valores faltantes considerando tanto o momento da medição quanto às características específicas de cada estação, sem impor uma ordem artificial entre elas.

Para facilitar o processamento subsequente e permitir a execução modular do *pipeline*, os dados tratados foram serializados em arquivos no formato *pickle (PKL)*, um formato binário nativo do *Python* que preserva integralmente a estrutura dos objetos, incluindo tipos de dados e índices do *pandas*. Para cada estação meteorológica, são gerados dois arquivos: um contendo o *DataFrame* filtrado com os registros daquela localidade (dados_tratados_{estacao}.pkl), e outro com os metadados do tratamento (metadata_tratamento_{estacao}.pkl), que inclui informações como nomes das colunas alvo e *features*, período dos dados, e contagem de valores faltantes. Essa separação por estação permite o processamento paralelo ou independente de cada série temporal no *notebook* de imputação, além de evitar a necessidade de reconexão com o banco de dados em etapas posteriores do *pipeline*.

Após essas etapas, o conjunto final passou a incluir identificadores, as variáveis meteorológicas principais: temperatura, umidade e velocidade do vento, dez *features* temporais e o conjunto das colunas referentes às estações. Com essa estrutura, iniciou-se a fase de imputação destinada a corrigir valores ausentes nas variáveis meteorológicas. Antes de realizar a imputação definitiva, uma parte dos valores originalmente completos foi temporariamente mascarada, permitindo comparar o valor real com o valor reconstruído.

5. Metodologia

A qualidade da imputação foi avaliada utilizando uma técnica de mascaramento, na qual 10% dos valores conhecidos foram temporariamente ocultados para simular dados faltantes. Isso permitiu comparar os valores imputados com os valores reais e calcular métricas objetivas de desempenho. As métricas utilizadas foram o *RMSE (Root Mean Squared Error)*, que penaliza erros maiores de forma quadrática; o *MAE (Mean Absolute Error)*, que oferece uma medida menos sensível a valores extremos; e o R^2 (coeficiente de determinação), que quantifica a parcela da variância explicada pela imputação. Além disso, gráficos de dispersão comparando valores reais versus imputados foram gerados para cada variável alvo, onde a linha diagonal representa a predição perfeita, permitindo visualizar o comportamento do processo e verificar sua adequação.

O algoritmo de imputação utilizado foi o *IterativeImputer* do *scikit-learn*, configurado com um *RandomForestRegressor* como estimador base. Essa escolha permite capturar relações não-lineares entre as variáveis meteorológicas e as *features* temporais durante o processo de estimação dos valores faltantes. Foram testadas três configurações distintas do modelo, denominadas Leve, Médio e Robusto, variando o número de estimadores (5, 10 e 15), a profundidade máxima das árvores (2, 3 e 4) e o número de iterações do imputador (2, 3 e 4). Os hiperparâmetros foram definidos a partir de considerações sobre o tempo de execução efetivo dos modelos. Todavia, o planejamento inicial previa a utilização de hiperparâmetros 8 vezes maiores. Cada configuração foi avaliada sistematicamente para todas as estações meteorológicas, permitindo comparar alternativas e identificar aquela que melhor concilia precisão e tempo de processamento.

Após a validação com os dados mascarados, a imputação foi aplicada aos dados originais completos, preenchendo apenas os valores efetivamente ausentes e mantendo os identificadores originais (id, data e hora) para preservar a rastreabilidade e permitir a atualização no banco de dados. Os resultados foram exportados em múltiplos formatos: arquivos *PKL* com o *dataset* completo imputado, arquivos *CSV* preparados para atualização no banco *Neon*, e arquivos de métricas para análise posterior.

O fluxo opera de maneira integrada ao *MLflow* para rastreamento de experimentos. A cada execução do *notebook*, são registrados automaticamente os parâmetros do modelo (*n_estimators, max_depth, max_iter*), as métricas de avaliação por variável e agregadas, os gráficos de avaliação como artefatos, e o próprio modelo treinado é versionado no *Model Registry* do *MLflow*. Cada estação meteorológica gera um modelo registrado independente, permitindo rastrear o histórico de versões e comparar desempenhos entre diferentes configurações e localidades.

Ao final do processamento, todos os arquivos gerados, incluindo os *datasets* imputados, *CSVs* para atualização, gráficos de avaliação e artefatos do *MLflow*, são automaticamente enviados para o *bucket S3*, organizados em estruturas de diretórios que facilitam a recuperação e análise posterior. Esse conjunto de etapas, desde o carregamento dos dados tratados, mascaramento para avaliação, imputação com *RandomForest*, experimentação com múltiplas configurações e integração com *MLflow* e *S3*, compõe o núcleo do *pipeline* de imputação que garante a qualidade dos dados utilizados nas etapas finais do projeto.

6. Análise de gráficos e insights

6.1 Panorama Climático de Pernambuco

A análise climática realizada com base nos dados meteorológicos de 12 estações revela um panorama marcadamente heterogêneo dentro de Pernambuco. A partir das variáveis de temperatura, umidade e velocidade do vento, torna-se possível compreender como diferentes combinações de fatores moldam realidades microclimáticas. O *dashboard* geral de séries temporais (Imagem 2.0 - Visão Geral das Estações Meteorológicas) permite visualizar comparativamente o comportamento de todas as estações ao longo do período analisado. Uma observação a ser feita é que os gráficos tendem a mascarar os *outliers*, pois para visualização os mesmos precisam ser agregados (pela média). Para acessar os dados de maneira completa, é necessário manipular os parâmetros de agregação do gráfico.

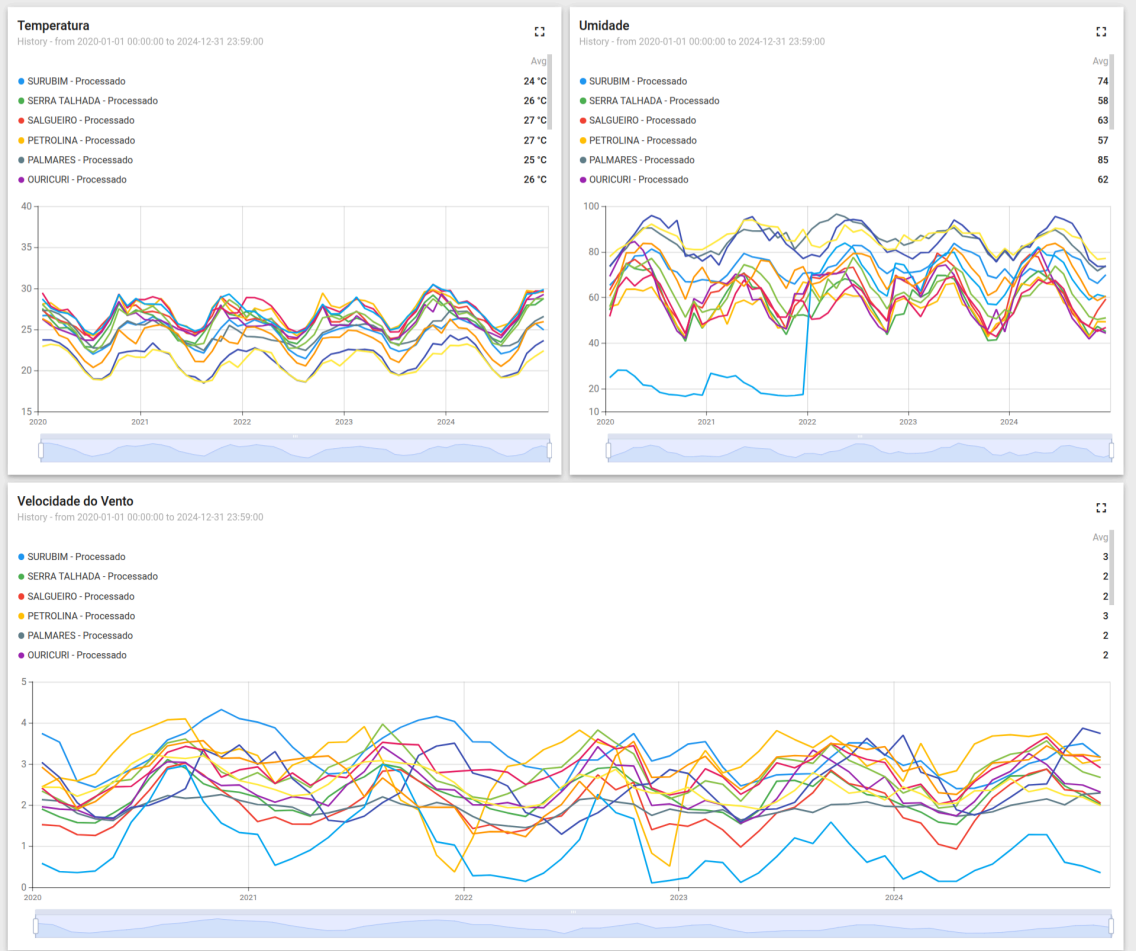

Imagem 2.0 - Visão Geral das Estações Meteorológicas

Logo de início, percebe-se uma divisão estrutural entre Sertão e Agreste/Zona da Mata. O Sertão, representado pelas estações de Cabrobó, Floresta, Ibimirim, Ouricuri, Petrolina, Salgueiro e Serra Talhada, se destaca pelo calor intenso, pela secura persistente e pelas amplitudes térmicas elevadas. Já o Agreste e a Zona da Mata, com as estações de Arcoverde, Caruaru, Garanhuns, Surubim e Palmares, aparecem como regiões notavelmente mais úmidas, com atmosferas estáveis, temperaturas mais moderadas e regimes de vento peculiares.

Cidade	Temp. Mediana (°C)	Umidade Média (%)	Horas de Secura Extrema	Horas de Vento Forte
Caruaru	20,5	86	0	251

Garanhuns	20,6	85	0	443
Palmares	24,0	85	1	308
Arcoverde	22,9	71	33	674
Surubim	23,6	74	11	3.940

Tabela 4.0 - Valores de temperatura mediana, umidade relativa e horas de eventos extremos por cidade.

No Sertão, Floresta aparece como o epicentro do calor, atingindo a maior temperatura registrada em toda a série (39,7°C), conforme observável nas séries temporais do *dashboard* específico desta estação. Cabrobó apresenta um perfil de secura impressionante com 14.930 horas de umidade abaixo de 20%, posicionando a cidade como a mais crítica para riscos respiratórios. Ibimirim registra a maior amplitude térmica entre mínima (12,5°C) e máxima (38,2°C), totalizando 25,7°C de variação. A tabela a seguir apresenta a distribuição térmica completa das cidades do Sertão.

Cidade	Temp. Mín (°C)	Temp. Mediana (°C)	Temp. Máx (°C)	Horas Secura Extrema	Horas Calor Extremo
Floresta	15,6	26,8	39,7	237	1.237
Cabrobó	16,1	26,7	39,6	14.930	895
Ibimirim	12,5	25,2	38,2	159	599
Ouricuri	14,1	25,0	38,4	380	395

Petrolina	17,6	27,0	38,5	102	589
Salgueiro	14,6	26,1	39,0	143	884
Serra Talhada	15,3	25,2	37,8	682	458

Tabela 5.0 - Estatísticas descritivas de temperatura e duração de eventos climáticos extremos por município.

Um achado particularmente interessante, observável nas séries temporais de cada estação, é o horário dos picos de calor. Ao contrário do senso comum que associa o meio-dia às temperaturas mais elevadas, observa-se que Cabrobó atinge 32,0°C às 18h com umidade crítica de apenas 36,5%, enquanto Floresta registra 32,2°C às 17h com umidade de 37,9%. Essa inércia térmica significativa tem implicações diretas em políticas de saúde e planejamento urbano.

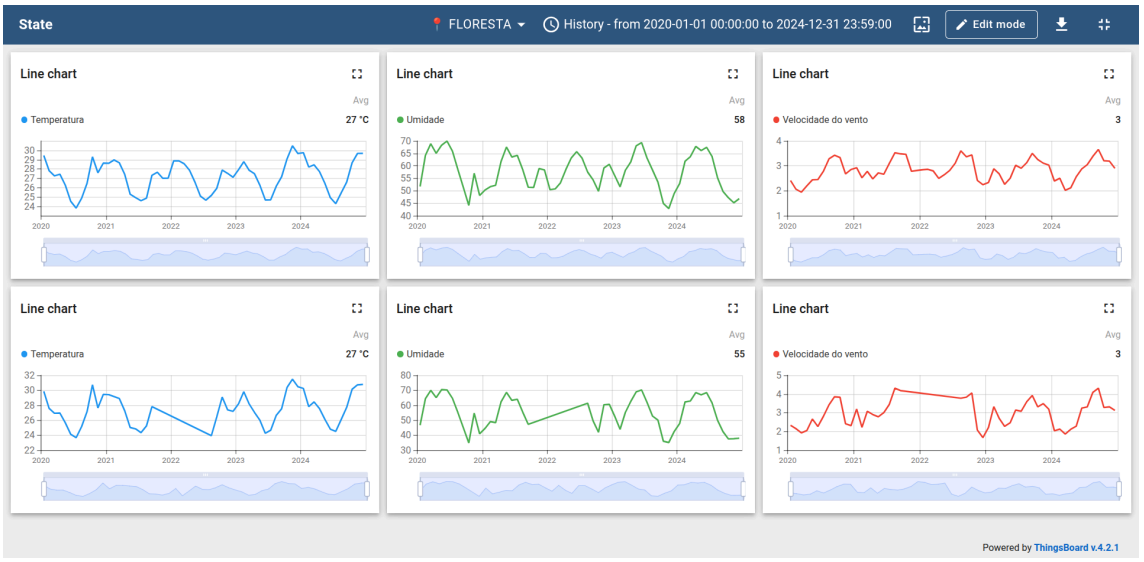

Imagem 3.0 - Visão da Estação Meteorológica de Floresta

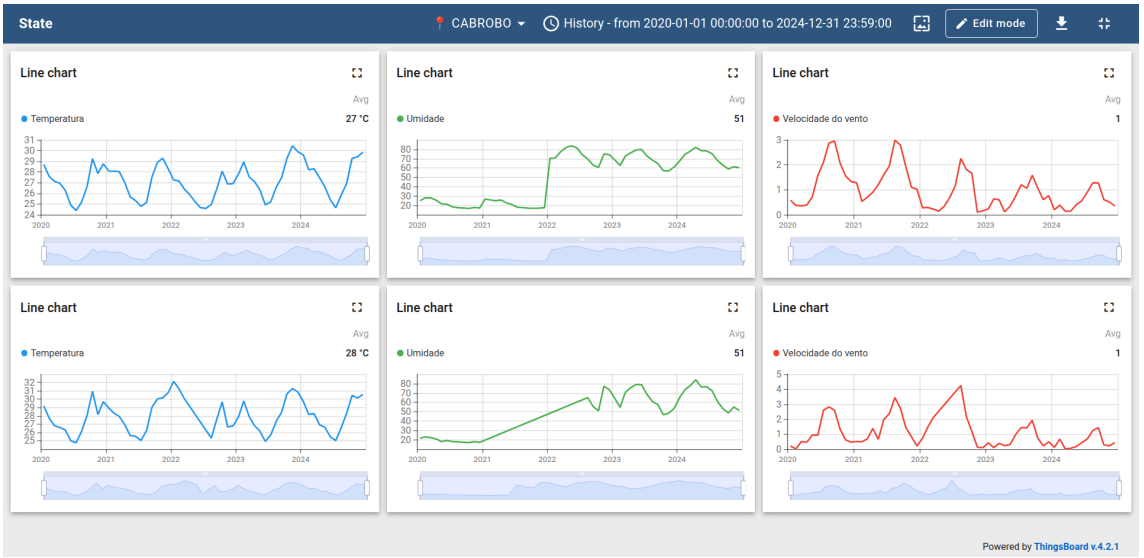

Imagem 4.0 - Visão da Estação Meteorológica de Cabrobó

Imagem 5.0 - Visão da Estação Meteorológica de Petrolina

Surubim se destaca como uma anomalia positiva no Agreste, com a maior média de velocidade do vento (3,2 m/s) e 9,0% do tempo em condição de vento forte, totalizando 3.940 horas em cinco anos. Esse regime confere à cidade enorme potencial para geração de energia eólica, conforme evidenciado nas séries temporais de velocidade do vento disponíveis no *dashboard* da estação. Em contraste, Caruaru registra zero horas de secura extrema e zero horas de calor extremo, com umidade média de 86%, evidenciando uma atmosfera continuamente hidratada. Garanhuns apresenta mais de 31.000 horas próximas da saturação, sugerindo alta formação de neblina.

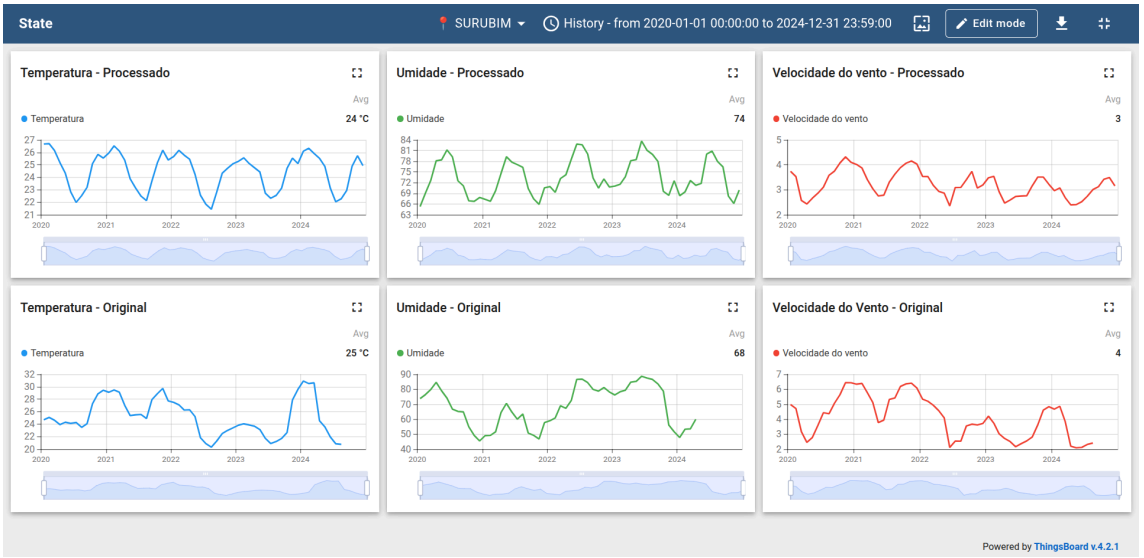

Imagem 6.0 - Visão da Estação Meteorológica de Surubim

Imagem 7.0 - Visão da Estação Meteorológica de Caruaru

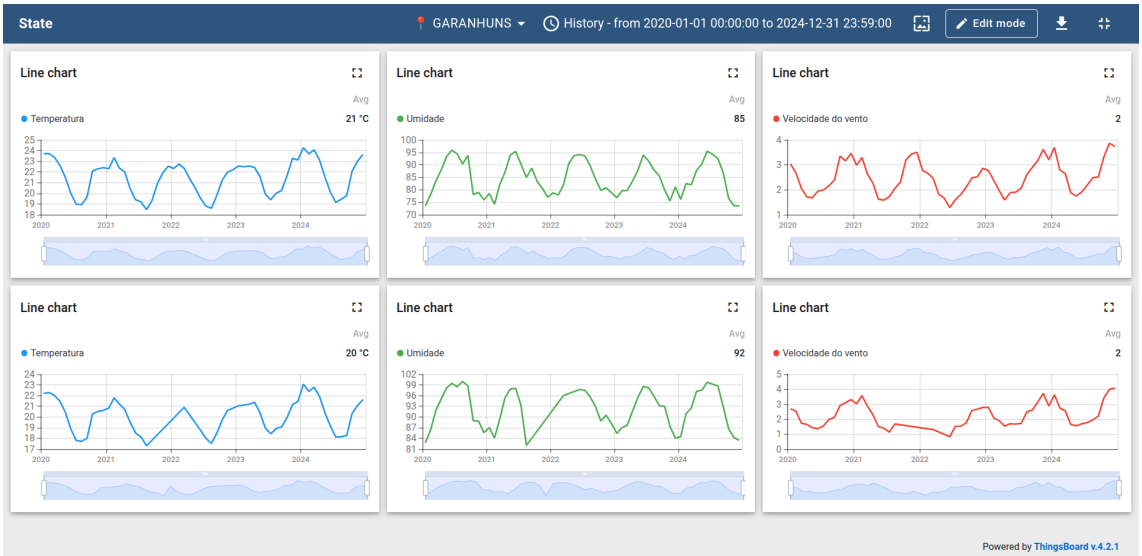

Imagem 8.0 - Visão da Estação Meteorológica de Garanhuns

A correlação entre temperatura e umidade revela padrões físicos distintos, observáveis ao comparar as séries temporais de cada variável nos *dashboards* individuais. Ibimirim e Surubim apresentam correlação quase perfeita (-0,94 e -0,92, respectivamente), demonstrando o comportamento atmosférico esperado onde temperatura alta implica umidade baixa. No entanto, Cabrobó aparece como anomalia com correlação de apenas -0,39, indicando que a cidade pode estar quente e úmida simultaneamente, possivelmente devido à proximidade com o Rio São Francisco ou irrigação agrícola intensiva.

Cidade	Original	Leve	Médio	Robusto
Ibimirim	-0,94	-0,94	-0,94	-0,94
Surubim	-0,92	-0,94	-0,93	-0,93
Arco Verde	-0,92	-0,91	-0,88	-0,90
Floresta	-0,91	-0,90	-0,88	-0,89
Serra Talhada	-0,90	-0,90	-0,85	-0,87

Garanhuns	-0,89	-0,88	-0,87	-0,87
Palmares	-0,89	-0,89	-0,88	-0,88
Petrolina	-0,87	-0,87	-0,86	-0,87
Caruaru	-0,86	-0,83	-0,87	-0,83
Ouricuri	-0,84	-0,79	-0,84	-0,83
Salgueiro	-0,83	-0,83	-0,83	-0,83
Cabrobó	-0,39	-0,35	-0,36	-0,36

Tabela 6.0 - Coeficientes de correlação temperatura-umidade: dados originais versus modelos gerados (Leve, Médio e Robusto).

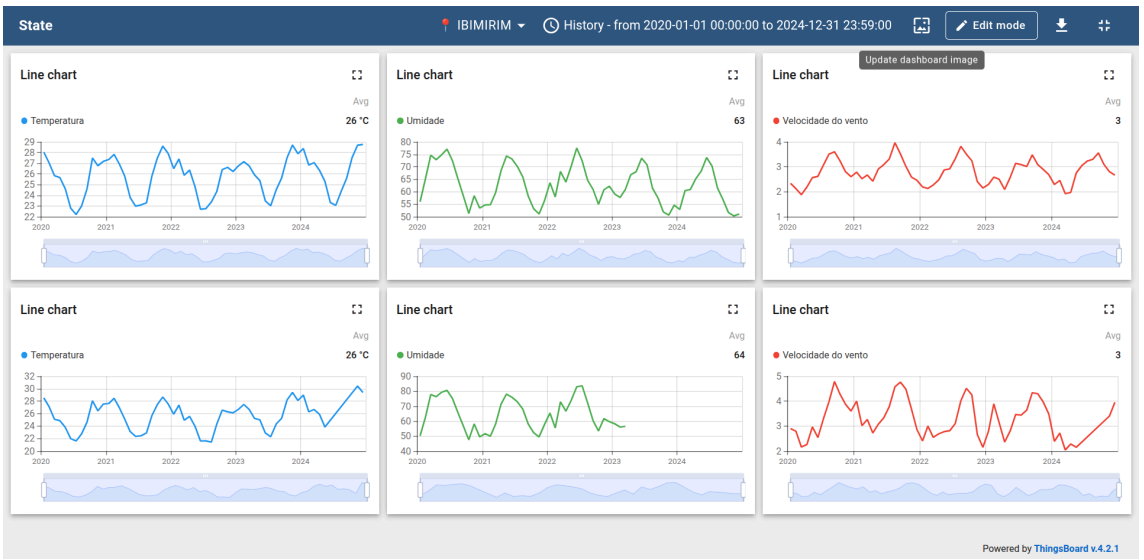

Imagem 9.0 - Visão da Estação Meteorológica de Ibimirim

6.2 Configuração e Desempenho dos Modelos de Imputação

Foram avaliadas três configurações do *IterativeImputer* com *RandomForestRegressor* como estimador base para todas as 12 estações meteorológicas, totalizando 36 modelos treinados e registrados no *MLflow*. O Modelo Leve utiliza 5 estimadores com profundidade máxima 2 e 2 iterações, o Modelo Médio utiliza 10 estimadores com profundidade 3 e 3 iterações, e o Modelo Robusto utiliza 15 estimadores com profundidade 4 e 4 iterações.

A preservação das medianas de temperatura foi avaliada comparando os valores originais com os imputados por cada modelo. O Modelo Leve apresentou desvios inferiores a 0,5°C em 10 das 12 cidades, com exceção de Petrolina (+1,6°C) e Garanhuns (-0,7°C). O Modelo Médio obteve desvios inferiores a 0,6°C em 11 cidades, com exceção de Serra Talhada (-1,2°C). O Modelo Robusto manteve desvios inferiores a 0,6°C em todas as 12 cidades.

Cidade	Original (°C)	Leve (°C)	Desvio Leve	Médio (°C)	Desvio Médio	Robusto (°C)	Desvio Robusto
Arco Verde	22,9	23,2	+0,3	23,1	+0,2	22,3	-0,6
Cabrobó	26,7	26,8	+0,1	27,0	+0,3	26,8	+0,1
Caruaru	20,5	20,5	0,0	20,5	0,0	20,2	-0,3
Floresta	26,8	26,1	-0,7	26,6	-0,2	26,3	-0,5
Garanhuns	20,6	19,9	-0,7	20,6	0,0	20,6	0,0
Ibimirim	25,2	25,3	+0,1	25,3	+0,1	25,3	+0,1
Ouricuri	25,0	25,2	+0,2	25,2	+0,2	25,2	+0,2

Palmares	24,0	23,4	-0,6	23,8	-0,2	23,9	-0,1
Petrolina	27,0	28,6	+1,6	26,4	-0,6	26,7	-0,3
Salgueiro	26,1	26,0	-0,1	26,6	+0,5	26,3	+0,2
Serra Talhada	25,2	25,0	-0,2	24,0	-1,2	24,7	-0,5
Surubim	23,6	23,9	+0,3	23,3	-0,3	23,9	+0,3

Tabela 6.0 - Análise de desvios de temperatura: dados observados versus resultados das simulações Leve, Média e Robusta.

6.3 Recomendações de Uso por Ambiente Computacional

Para dispositivos de borda *(Edge Computing)*, recomenda-se o Modelo Leve, que utiliza aproximadamente 50-100 *MB* de memória e executa inferências em menos de 1 segundo. Este modelo é adequado para sensores *IoT* nas estações meteorológicas, dispositivos como *Raspberry Pi* ou *ESP32*, e aplicações que exigem resposta em tempo real. O Modelo Leve apresentou desempenho excelente (desvio inferior a 0,3°C) em Caruaru, Cabrobó, Ibimirim, Salgueiro e Surubim, e desempenho adequado nas demais cidades, com exceção de Petrolina onde o desvio de +1,6°C recomenda o uso do Modelo Médio mesmo na borda.

Para computação em névoa *(Fog Computing)*, recomenda-se o Modelo Médio, que utiliza aproximadamente 200-400 *MB* de memória e executa inferências entre 1 e 5 segundos. Este modelo é adequado para *gateways* regionais agregando dados de múltiplas estações, centros de monitoramento municipal e aplicações com latência moderada. O Modelo Médio apresentou desempenho excelente em Caruaru, Garanhuns, Ibimirim e Ouricuri, sendo particularmente recomendado para Petrolina onde corrigiu o problema observado no Modelo Leve. A exceção é Serra Talhada, onde o desvio de -1,2°C recomenda o uso do Modelo Robusto.

Para computação em nuvem (*Cloud Computing*), recomenda-se o Modelo Robusto, que utiliza aproximadamente 500 *MB* a 1 *GB* de memória e executa inferências entre 5 e 15 segundos. Este modelo é adequado para processamento em lote de séries históricas, treinamento e retreinamento periódico dos modelos, análises retrospectivas e geração de *datasets* para *dashboards*. O Modelo Robusto apresentou o melhor desempenho geral, com desvios inferiores a 0,6°C em todas as 12 cidades e preservação consistente das correlações.

Critério	**Borda (Leve)**	***Fog* (Médio)**	**Nuvem (Robusto)**
Latência	< 1s	1–5s	5–15s
Memória	< 100 *MB*	~300 *MB*	~700 *MB*
Precisão	10/12 cidades	11/12 cidades	12/12 cidades
Operação *Offline*	Sim	Sim	Requer conexão
Cidades Problemáticas	Petrolina	Serra Talhada	Nenhuma

Tabela 7.0 - Comparativo de latência, consumo de memória e precisão entre as arquiteturas de Borda, *Fog* e Nuvem.

A recomendação específica por estação considera tanto o desempenho observado quanto às características climáticas locais. Para Cabrobó, Caruaru, Ibimirim e Surubim, o Modelo Leve é suficiente para todas as camadas. Para Arco Verde, Floresta, Garanhuns, Ouricuri e Palmares, recomenda-se o Modelo Médio como padrão. Para Petrolina, o Modelo Médio é obrigatório mesmo na borda devido ao desvio observado no Modelo Leve. Para Salgueiro e Serra Talhada, o Modelo Robusto oferece a melhor precisão e é recomendado para aplicações críticas.

A arquitetura híbrida recomendada organiza o processamento em três camadas: na borda, cada uma das 12 estações executa o modelo apropriado (Leve para maioria, Médio para Petrolina); na névoa, *gateways* regionais do Sertão e do Agreste agregam dados usando o Modelo Médio; na nuvem, o *pipeline* completo com *MLflow*, *ThingsBoard* e *Trendz* processa os 36 modelos para análises históricas e *dashboards* consolidados, conforme exemplificado nos *dashboards* de séries temporais disponíveis para cada estação.

7. Conclusão

O projeto permitiu construir um *pipeline* integrado capaz de realizar coleta, tratamento, imputação, modelagem e visualização de dados meteorológicos do INMET, resultando em uma base consolidada e adequada para análises climáticas em escala regional. A combinação dessas etapas tornou possível identificar padrões térmicos, regimes de umidade e dinâmicas de vento que caracterizam de forma distinta as regiões do estado.

Os resultados evidenciaram dois regimes climáticos bem definidos. O Sertão apresentou condições de calor extremo, baixa umidade e amplitudes térmicas acentuadas, especialmente em municípios como Floresta, Cabrobó e Ibimirim, que concentram eventos críticos de temperatura e secura. Em contraste, o Agreste e a Zona da Mata exibiram ambientes mais úmidos, estáveis e de menor variabilidade térmica, com destaque para Caruaru, Garanhuns e Palmares. Surubim se destacou pela intensidade e constância dos ventos, reforçando seu potencial para aplicações voltadas à energia eólica.

No eixo metodológico, a imputação demonstrou boa estabilidade estatística. A técnica de mascaramento permitiu avaliar de forma objetiva as três configurações testadas, com métricas como *RMSE*, *MAE* e R^2 indicando preservação das distribuições originais e das correlações entre temperatura e umidade. O Modelo Robusto apresentou o melhor desempenho geral, enquanto os Modelos Leve e Médio atenderam adequadamente a cenários com restrições computacionais, permitindo flexibilizar a escolha de acordo com as necessidades de cada ambiente.

A análise comparativa entre arquiteturas de Borda, *Fog* e Nuvem permitiu estabelecer recomendações específicas para implantação, considerando desempenho por cidade, consumo de recursos e precisão dos modelos. Essa estrutura facilita o uso de soluções distribuídas para monitoramento climático, adaptando o processamento ao contexto operacional de cada aplicação.

Por fim, a integração com o *ThingsBoard* consolidou o *pipeline* como plataforma de visualização e interpretação, reunindo séries temporais, indicadores extremos e resultados de imputação em um único ambiente. Essa abordagem favorece a identificação de tendências como picos tardios de temperatura, longos períodos de secura no Sertão, saturação de umidade em regiões elevadas e padrões de vento no Agreste.

O conjunto de resultados demonstra que a solução desenvolvida oferece uma base consistente para análises climáticas, permitindo tanto o entendimento das dinâmicas atmosféricas regionais quanto a expansão para estudos futuros, seja pela inclusão de novas variáveis, ampliação territorial ou adoção de modelos preditivos mais sofisticados.

8. Referências

BREIMAN, Leo. Random Forests. Machine Learning, v. 45, n. 1, p. 5–32, 2001.

DIÁTAXIS. Diátaxis Documentation Framework. Daniele Procida. Disponível em: https://diataxis.fr

FASTAPI. FastAPI Documentation. Sebastián Ramírez. Disponível em: https://fastapi.tiangolo.com

MERMAID. Mermaid Official Documentation. Disponível em: https://mermaid.js.org

MLFLOW. MLflow Documentation. Disponível em: https://mlflow.org/docs/latest/index.html

RUBIN, Donald B. Multiple Imputation for Nonresponse in Surveys. New York: John Wiley & Sons, 1987.

SCIKIT-LEARN. User Guide. scikit-learn Documentation. Disponível em: https://scikit-learn.org/stable/user_guide.html

VAN BUUREN, Stef. Flexible Imputation of Missing Data. 2. ed. Boca Raton: CRC Press, 2018.

NEON. Neon Documentation. Disponível em: https://neon.tech/docs