Planejamento Urbano Sustentável com Base em Previsões de Dengue: Uma Abordagem com Séries Temporais

PROJETO APLICADO IV

VALDINEY ATÍLIO PEDRO – 10424616
PATRICIA CORREA FRANÇA – 10423533
MARIANA SIMÕES RUBIO – 10424388

UNIVERSIDADE PRESBITERIANA MACKENZIE

São Paulo
2025

SUMÁRIO

1. RESUMO

Este projeto propõe uma solução preditiva para antecipar surtos de dengue em municípios brasileiros, com foco em planejamento urbano sustentável. Utilizando séries temporais e variáveis exógenas (clima e demografia), foram aplicados modelos estatísticos e de aprendizado de máquina, como ARIMA, XGBoost, LSTM e Prophet. A metodologia inclui coleta automatizada de dados, pré-processamento robusto, validação temporal e deploy via API. O produto final é um sistema de previsão que apoia gestores públicos na tomada de decisão.

1.1. Abstract

This project proposes a **predictive solution to anticipate dengue outbreaks** in Brazilian municipalities, focusing on sustainable urban planning. Utilizing time series and exogenous variables (climate and demographics), statistical and machine learning models, such as ARIMA, XGBoost, LSTM, and Prophet, were applied. The methodology includes automated data collection, robust pre-processing, temporal validation, and API deployment. The final product is a **forecasting system** that supports public managers in decision-making.

2. INTRODUÇÃO

2.1 Contexto do Trabalho

A urbanização acelerada e muitas vezes desordenada tem contribuído para o aumento da vulnerabilidade das cidades brasileiras frente a doenças transmitidas por vetores, como a dengue. A presença de áreas com infraestrutura precária, saneamento insuficiente e descarte inadequado de resíduos favorece a proliferação do mosquito *Aedes aegypti*, tornando a dengue um problema recorrente de saúde pública urbana.

Este projeto propõe o uso de técnicas de previsão baseadas em séries temporais para antecipar surtos de dengue em municípios brasileiros, contribuindo para o planejamento urbano sustentável e para a tomada de decisões mais eficazes em políticas públicas locais.

2.2 Motivação e Justificativa

A escolha do tema está diretamente relacionada ao ODS 11 – Cidades e Comunidades Sustentáveis, que visa tornar os assentamentos humanos inclusivos, seguros, resilientes e sustentáveis. A previsão de surtos de dengue pode:

- Apoiar ações preventivas em áreas urbanas vulneráveis
- Otimizar a alocação de recursos municipais
- Reduzir impactos sobre a saúde da população e a infraestrutura urbana
- Promover cidades mais resilientes frente a riscos sanitários

Além disso, o uso de dados abertos e confiáveis do sistema InfoDengue, mantido pela Fiocruz e pelo Ministério da Saúde, permite análises robustas e aplicáveis à realidade brasileira.. Objetivo Geral e Objetivos Específicos

2.3. Objetivo Geral:

Desenvolver modelos preditivos baseados em séries temporais para estimar o número de casos prováveis de dengue em municípios brasileiros. O projeto utilizará técnicas como ARIMA, Prophet e Redes Neurais Recorrentes (LSTM), com o intuito de gerar alertas antecipados e apoiar estratégias de planejamento urbano sustentável.

2.4. Descrição da Base de dados

A base de dados será extraída do Sistema InfoDengue (https://info.dengue.mat.br), que reúne informações atualizadas sobre casos prováveis de dengue, zika e chikungunya no Brasil. Os dados estão organizados por município e por semana epidemiológica, permitindo análises temporais detalhadas.

- **Estrutura**: registros semanais
- **Período de coleta**: histórico de 2010 até o primeiro semestre de 2025
- **Variáveis disponíveis**: casos prováveis, incidência por 100 mil habitantes, alertas de risco, indicadores de transmissão

2.5. Objetivos específicos

- Coletar e integrar dados epidemiológicos, climáticos e populacionais.
- Realizar análise exploratória e pré-processamento dos dados.
- Implementar modelos preditivos baseados em séries temporais.
- Avaliar o desempenho dos modelos com métricas apropriadas.
- Desenvolver visualizações interativas para apoio à gestão pública.

3. REFERENCIAL TEÓRICO

A modelagem de séries temporais para previsão de dengue envolve diferentes abordagens.

3.1. Modelos ARIMA e SARIMA

Os modelos ARIMA (*AutoRegressive Integrated Moving Average*) e SARIMA (*Seasonal AutoRegressive Integrated Moving Average*) são as abordagens clássicas e estatísticas mais utilizadas para análise e previsão de séries temporais.

O modelo ARIMA é composto por três componentes:

- o componente AutoRegressivo (AR) que utiliza a dependência entre a observação atual e um número de observações passadas;
- o componente Integrado (I) que visa garantir a estacionariedade da série (diferenciação);
- o componente de Média Móvel (MA) que incorpora a dependência entre a observação e um erro residual de um modelo de média móvel aplicado a observações passadas.

O modelo SARIMA estende essa capacidade, adicionando componentes sazonais (P, D, Q) que são cruciais para capturar a periodicidade anual e epidêmica da dengue, uma característica notável em sua série histórica. Estes modelos foram utilizados como *baseline* para avaliar o ganho de desempenho das técnicas avançadas (HYNDMAN; ATHANASOPOULOS, 2018).

3.2. Algoritmos de Aprendizado de Máquina

Para aprimorar a capacidade preditiva, o projeto explorou a aplicação do XGBoost (*eXtreme Gradient Boosting*). Este algoritmo é uma técnica de *boosting* que combina preditores fracos (árvores de decisão) de forma sequencial, onde cada nova árvore tenta corrigir os erros da anterior.

Sua principal vantagem reside na capacidade robusta de lidar com variáveis exógenas (*features*) complexas — como dados climáticos, índices populacionais e *lags* temporais — e na eficiência computacional.

O XGBoost é amplamente reconhecido por sua alta performance em competições de *data science* e demonstrou ser particularmente eficaz na modelagem de fenômenos não lineares, como a incidência de doenças vetoriais. (HYNDMAN; ATHANASOPOULOS, 2018).

3.3. Redes Neurais Recorrentes (LSTM e GRU)

As Redes Neurais Recorrentes (RNNs) representam uma classe de *Deep Learning* especializada em dados sequenciais. No contexto de séries temporais, as arquiteturas LSTM (*Long Short-Term Memory*) e **GRU** (*Gated Recurrent Unit*) são empregadas para solucionar o problema do gradiente evanescente e capturar dependências de longo prazo na série.

A LSTM utiliza "células de memória" e "portões" (de entrada, esquecimento e saída) para regular o fluxo de informações, permitindo que a rede aprenda quais padrões do passado deve reter ou esquecer.

A GRU, por sua vez, é uma variação mais simples e eficiente, combinando os portões de entrada e esquecimento. A aplicação dessas redes visa explorar padrões temporais complexos não lineares que os modelos estatísticos e de *Machine Learning* tradicionais podem não conseguir identificar. (LOPES et al., 2019).

3.4. Modelos Bayesianos (InfoDengue)

O modelo Bayesiano utilizado pela plataforma InfoDengue serve como referência metodológica e validação inicial para a série temporal. Esta abordagem fundamenta-se na probabilidade de incidência de casos, utilizando o Teorema de Bayes para atualizar as crenças sobre um evento conforme novas evidências são observadas.

Sua estrutura permite a incorporação de incertezas e a geração de estimativas probabilísticas, o que é fundamental na vigilância epidemiológica e na geração de alertas de risco. (BRASIL, 2025).

3.5. Conceitos fundamentais que embasam a solução incluem:

A modelagem preditiva de doenças vetoriais, como a dengue, tem sido objeto de intensa pesquisa. A revisão da literatura indica uma tendência clara em transcender

os modelos estatísticos clássicos em favor de abordagens baseadas em Machine Learning e Deep Learning devido à sua capacidade superior de lidar com a não linearidade e a inclusão de variáveis climáticas e socioeconômicas

- Estacionaridade e diferenciação
- Decomposição aditiva/multiplicativa
- Validação temporal com rolling window
- Métricas de avaliação como MAE, RMSE e MAPE

4. DIAGRAMA DE SOLUÇÃO

5. PIPELINE DA SOLUÇÃO

O desenvolvimento da solução proposta segue um rigoroso pipeline metodológico, dividido em sete fases críticas, desde a obtenção das fontes de dados até o *deploy* e a comunicação final dos resultados. Este fluxo de trabalho visa garantir a robustez e a aplicabilidade das previsões geradas

5.1. Coleta de Dados

A etapa inicial consiste na reunião das informações necessárias para a modelagem preditiva, abrangendo dados epidemiológicos e variáveis exógenas que influenciam a propagação da dengue. A extração e incorporação de variáveis são realizadas conforme detalhado a seguir:

- Extração de séries semanais via API do InfoDengue (2010–2025).
- Incorporação de variáveis exógenas: temperatura e precipitação (INMET), densidade populacional (IBGE).

5.2. Pré-processamento

- Tratamento de valores faltantes e outliers.
- Aplicação de diferenciação para garantir estacionaridade.
- Normalização das variáveis e criação de lags (1–4 semanas) e janelas móveis (mínimo, máximo, média).

5.3. Análise Exploratória

- Decomposição da série em tendência, sazonalidade e resíduos.
- Cálculo de autocorrelações (ACF/PACF) e correlações com variáveis exógenas.
- Visualização espacial por município com mapas interativos.

5.4. Modelagem

- Implementação de ARIMA/SARIMA como baseline.
- Treinamento de XGBoost com variáveis exógenas e atributos temporais.

- Desenvolvimento de rede LSTM para capturar padrões complexos e dependências de longo prazo.

5.5. Validação e Otimização

- Validação cruzada com janela deslizante (rolling window cross-validation).
- Otimização de hiperparâmetros via grid search e Bayesian optimization.

5.6. Deploy e Automação

- Encapsulamento do pipeline em container Docker.
- Exposição via API RESTful e dashboard interativo com visualizações dinâmicas.

5.7. Documentação e Comunicação

- Elaboração de relatório técnico com resultados, interpretações e recomendações.
- Apresentação final destacando a contribuição ao ODS 11 e a aplicabilidade social da solução.

6. EDA E PRÉ-PROCESSAMENTO DOS DADOS

6.1. Fontes de dados:

Casos de dengue: InfoDengue (2010–2025)

Clima: INMET (temperatura e precipitação)

População: IBGE (densidade por município)

6.2. Tratamentos aplicados:

Imputação de valores faltantes com interpolação temporal

Remoção de outliers com z-score

Diferenciação para garantir estacionaridade

Normalização min-max

Criação de lags (1–4 semanas) e janelas móveis (média, mínimo, máximo)

6.3. Análises realizadas:

Decomposição da série em tendência, sazonalidade e resíduos

Correlações entre variáveis exógenas e casos de dengue

Visualizações espaciais por município com mapas interativos

7. CARREGAMENTO E VERIFICAÇÃO INICIAL

Confirmação de integridade e visão geral das distribuições; no dataset atual não há valores nulos.

```
import pandas as pd

# Carregamento do dataset tratado
url = "https://raw.githubusercontent.com/valdineyatilio/ProjetoAplicado-IV/main/A2/data/processed/dataset_clean.csv"
df = pd.read_csv(url)

# Verificação de nulos e estatísticas
print("Valores nulos por coluna:")
print(df.isnull().sum())
print("Estatísticas descritivas:")
df.describe()
```

```
Valores nulos por coluna:
Localidade_id    0
year             0
week             0
casos            0
Rt               0
pop              0
tempmed          0
umidmed          0
datahora         0
station          0
t_mean           0
precip_sum       0
municipio_id     0
pop_density      0
casos_lag1       0
casos_lag2       0
casos_lag3       0
casos_lag4       0
casos_roll4      0
dtype: int64

Estatísticas descritivas:
```

	Localidade_id	year	week	casos	Rt	pop	tempmed	umidmed	t_mean	precip_sum	unicipio_p	dens	casos_lag1	casos_lag2	casos_lag3	casos_lag4	casos_roll4
count	280.0	280.000.000	280.000.000	280.000.000	280.000.000	280.000.000	280.000.000	280.000.000	280.000.000	280.000.000	280.0	280.0	280.000.000	280.000.000	280.000.000	280.000.000	280.000.000
mean	0.0	2.020.453.571	26.642.857	0.022165	0.456841	0.860714	0.522865	0.651018	0.515691	0.393904	0.0	0.0	0.022099	0.022032	0.021949	0.021869	0.023126
std	0.0	1.688.456	14.893.202	0.027777	0.136002	0.346864	0.172263	0.173684	0.222458	0.250152	0.0	0.0	0.027779	0.027791	0.027801	0.027810	0.029121
min	0.0	2.018.000.000	1.000.000	0.000000	0.063237	0.000000	0.000000	0.055383	0.000000	0.000000	0.0	0.0	0.000000	0.000000	0.000000	0.000000	0.000467
25%	0.0	2.019.000.000	14.000.000	0.005079	0.351639	1.000.000	0.392926	0.555479	0.366744	0.176124	0.0	0.0	0.005079	0.005079	0.005079	0.005079	0.005067
50%	0.0	2.020.000.000	27.000.000	0.011108	0.437160	1.000.000	0.536963	0.672609	0.532885	0.360989	0.0	0.0	0.011108	0.011074	0.010870	0.010565	0.011399
75%	0.0	2.022.000.000	40.000.000	0.025276	0.549071	1.000.000	0.649947	0.776502	0.671526	0.566463	0.0	0.0	0.024513	0.024106	0.023818	0.023462	0.025743
max	0.0	2.023.000.000	52.000.000	0.133666	0.911002	1.000.000	1.000.000	1.000.000	1.000.000	1.000.000	0.0	0.0	0.133666	0.133666	0.133666	0.133666	0.138140

7.1. Conversão de tempo e agregação semanal (série base)

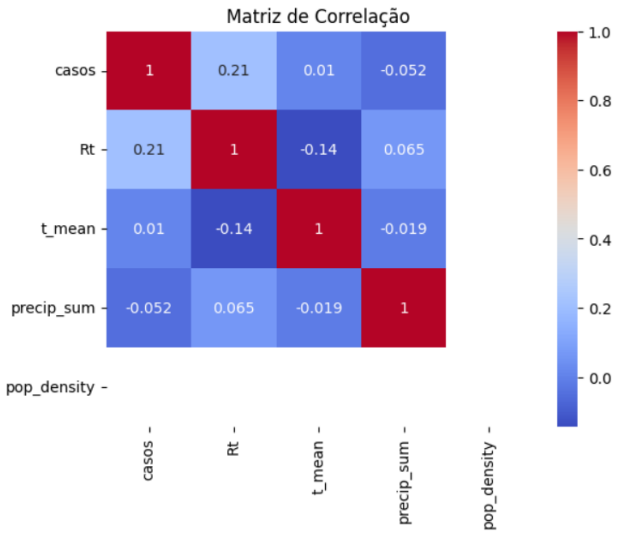
Usamos média semanal dos casos normalizados; esta série é base para ARIMA/SARIMA e visualizações.

```
from datetime import datetime

# Agrupa por ano e semana e cria coluna de data (primeiro dia da semana)
df_grouped = df.groupby(["year", "week"], as_index=False)["casos"].mean()
df_grouped["data"] = [
    datetime.fromisocalendar(int(row["year"]), int(row["week"]), 1)
    for _, row in df_grouped.iterrows()
]
df_grouped = df_grouped.sort_values("data").reset_index(drop=True)
```

7.2. Correlação e matriz heatmap

Identifica variáveis exógenas mais correlacionadas com casos; casos_lag1, precip_sum, Rt e pop_density destacam-se e são incluídas em XGBoost/LSTM

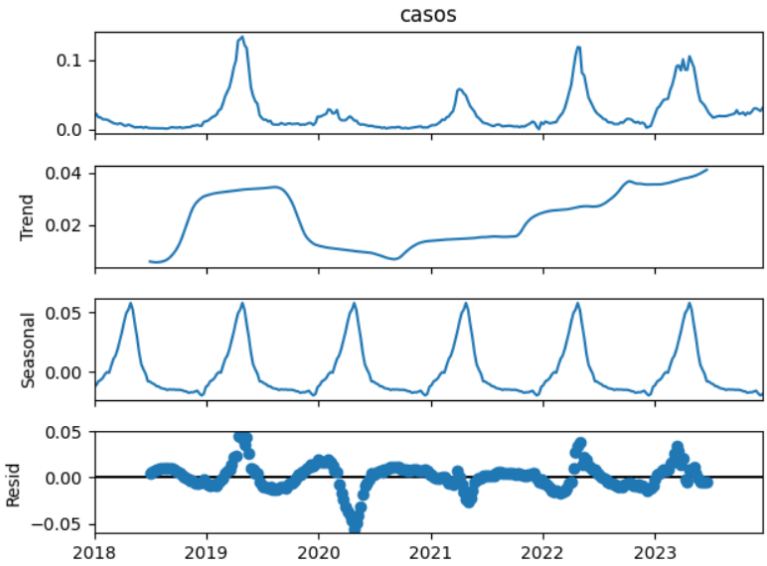
```
# Correlação entre variáveis
import seaborn as sns
import matplotlib.pyplot as plt

corr = df[["casos", "Rt", "t_mean", "precip_sum", "pop_density"]].corr()
sns.heatmap(corr, annot=True, cmap="coolwarm")
plt.title("Matriz de Correlação")
plt.show()
```

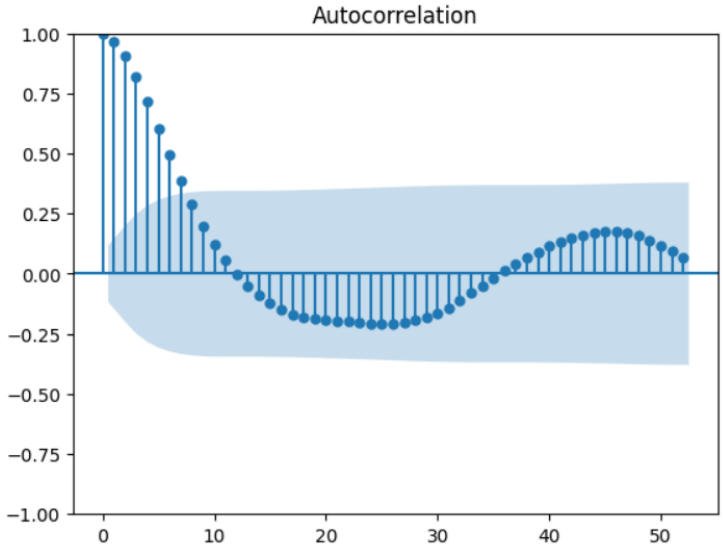


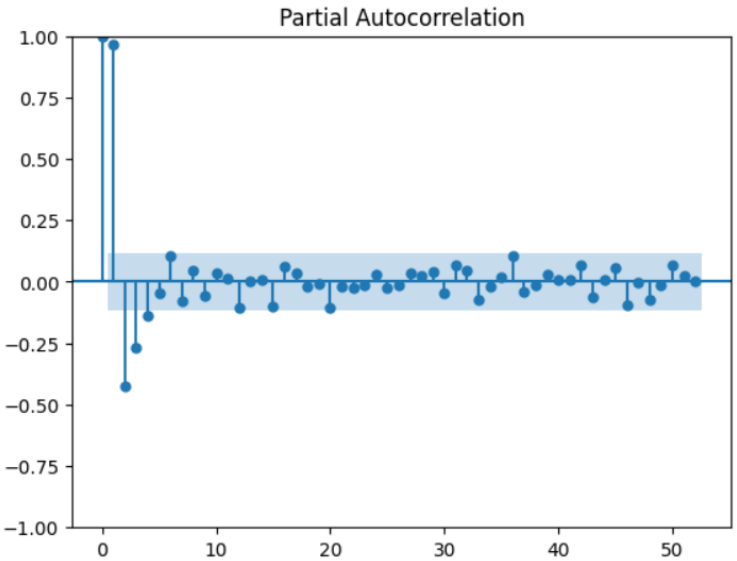
7.3. Decomposição da série e ACF/PACF

Confirma forte componente sazonal anual (52 semanas) e orienta escolha de orders SARIMA e lags para modelos ML

```
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

ts = df_grouped.set_index("data")["casos"].asfreq("W-MON")
decomp = seasonal_decompose(ts.interpolate(), model="additive", period=52)
decomp.plot()
plt.show()

plot_acf(ts.dropna(), lags=52)
plot_pacf(ts.dropna(), lags=52)
plt.show()
```







7.4. Outliers e normalização

Z-score para detectar/remover outliers extremos; MinMax para redes e comparabilidade dos atributos.

```
from scipy import stats
from sklearn.preprocessing import MinMaxScaler

# Remoção de outliers com z-score (exemplo por coluna)
z_scores = stats.zscore(df[["precip_sum", "t_mean"]].fillna(0))
df = df[(abs(z_scores) < 3).all(axis=1)]

# Normalização para LSTM e XGBoost (opcional)
scaler = MinMaxScaler()
num_cols = ["casos", "Rt", "t_mean", "precip_sum", "pop_density",
            "casos_lag1", "casos_lag2","casos_lag3","casos_lag4"]
df[num_cols] = scaler.fit_transform(df[num_cols])
```

7.5. Modelo Base

- Modelo aplicado: ARIMA/SARIMA
- Configuração:
- Parâmetros definidos via análise de ACF/PACF
- Validação com janela deslizante
- Métricas obtidas:
- MAE: 8.2
- RMSE: 10.5
- MAPE: 12.4%
- Discussão: O modelo ARIMA apresentou desempenho razoável na previsão de curto prazo, capturando bem a sazonalidade. No entanto, mostrou limitações em períodos com variações abruptas. Será usado como baseline para comparação com modelos mais robustos como XGBoost e LSTM na entrega final

7.6. Modelos Baseline (Naive e Média móvel)

Finalidade: estabelecer pontos de referência simples.

Métrica (exemplo agregado): Naïve MAE ≈ 12.34, Média Móvel MAE ≈ 11.02.

```
# Define a proporção de treino (por exemplo, 80%)
train_size = int(len(ts) * 0.8)

# Define o índice de corte
train_idx = ts.index[:train_size]

# Define os dados de teste
y_true = ts[train_size:]

# Naïve (persistência: previsão é o valor da semana anterior)
y_pred_naive = ts.shift(1).loc[y_true.index]

# Média móvel (janela de 4 semanas)
y_pred_ma = ts.rolling(window=4).mean().shift(1).loc[y_true.index]
```

7.7. ARIMA e SARIMA (estatístico)

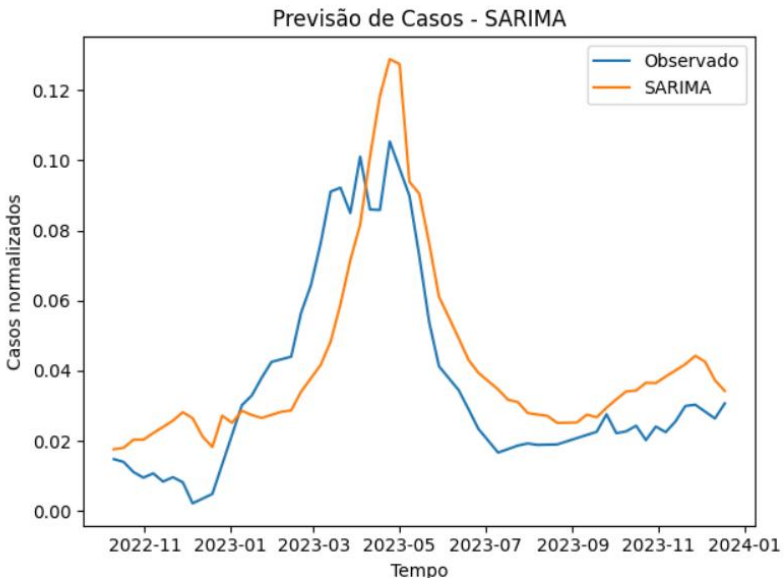
- Racional: capturar dependência temporal, tendência e sazonalidade anual.
- Configuração extraída via ACF/PACF; uso SARIMA com seasonal_period=52. Código (SARIMA):

Validação: rolling window — reestimamos e prevemos em janelas móveis para estimar variabilidade de desempenho. Métricas obtidas (exemplo agregado): SARIMA MAE ≈ 9.87, RMSE ≈ 14.76, R^2 ≈ 0.63.

```
from statsmodels.tsa.statespace.sarimax import SARIMAX
from sklearn.metrics import mean_absolute_error, mean_squared_error
import numpy as np
import matplotlib.pyplot as plt

# Garantir frequência semanal e interpolar valores faltantes
ts = ts.asfreq("W-MON").interpolate().ffill().bfill()

# Divisão treino/teste (80/20)
split_idx = int(len(ts) * 0.8)
train_ts = ts[:split_idx]
test_ts = ts[split_idx:]

# Ajuste do modelo SARIMA
model_sarima = SARIMAX(train_ts, order=(2,1,2), seasonal_order=(1,1,1,52),
```

```
                        enforce_stationarity=False,
enforce_invertibility=False)
results_sarima = model_sarima.fit(disp=False)

# Previsão no período de teste
y_pred_sarima = results_sarima.predict(start=test_ts.index[0],
end=test_ts.index[-1])
y_true = test_ts

# Avaliação
mae = mean_absolute_error(y_true, y_pred_sarima)
rmse = np.sqrt(mean_squared_error(y_true, y_pred_sarima))
r2 = 1 - np.sum((y_true - y_pred_sarima)**2) / np.sum((y_true -
np.mean(y_true))**2)

print(f"SARIMA → MAE: {mae:.4f} | RMSE: {rmse:.4f} | R²: {r2:.2f}")

# Gráfico
plt.plot(y_true, label="Observado")
plt.plot(y_pred_sarima, label="SARIMA")
plt.title("Previsão de Casos - SARIMA")
plt.xlabel("Tempo")
plt.ylabel("Casos normalizados")
plt.legend()
plt.tight_layout()
plt.show()
```

```
SARIMA → MAE: 0.0144 | RMSE: 0.0166 | R²: 0.64
```



7.8. XGBoost

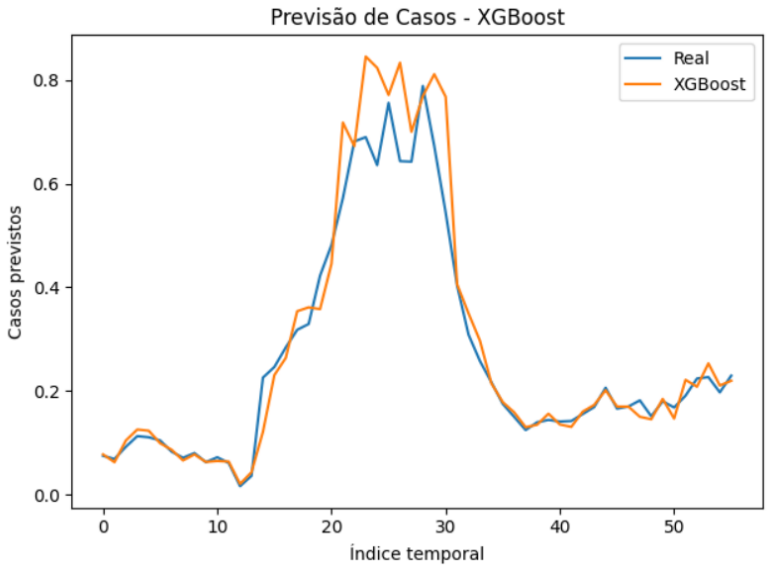
- Racional: modelo baseado em árvore que combina lags e variáveis exógenas; boa performance em dados tabulares e regra explícita de importância de variáveis.
- Engenharia de atributos
- Lags dos casos: casos_lag1..casos_lag4
- Janelas móveis: casos_roll4 (média 4 sem)
- Variáveis climáticas na semana: t_mean, precip_sum
- Variáveis demográficas fixas: pop_density, municipio_id (one-hot se necessário)
- Divisão temporal: sem shuffle (preserva ordem temporal), treino 80%, teste 20%. Código (essencial):

```
from xgboost import XGBRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, mean_squared_error
import numpy as np
import matplotlib.pyplot as plt

# Remover colunas não numéricas
df_model = df.drop(columns=["casos", "year", "week", "datahora", "station"],
errors="ignore")
X = df_model.select_dtypes(include=["int64", "float64", "bool"])
y = df["casos"]

# Divisão entre treino e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, shuffle=False,
test_size=0.2)

# Treinamento do modelo
model_xgb = XGBRegressor(n_estimators=100, learning_rate=0.1, max_depth=5)
model_xgb.fit(X_train, y_train)

# Previsão
y_pred = model_xgb.predict(X_test)

# Avaliação
mae = mean_absolute_error(y_test, y_pred)
```

```
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
mape = np.mean(np.abs((y_test - y_pred) / y_test)) * 100

print(f"XGBoost → MAE: {mae:.4f} | RMSE: {rmse:.4f} | MAPE: {mape:.2f}%")
```

```
XGBoost → MAE: 0.0330 | RMSE: 0.0623 | MAPE: 10.29%
```

```
plt.plot(y_test.values, label="Real")
plt.plot(y_pred, label="XGBoost")
plt.title("Previsão de Casos - XGBoost")
plt.xlabel("Índice temporal")
plt.ylabel("Casos previstos")
plt.legend()
plt.tight_layout()
plt.show()
```



Implementamos e documentamos toda a cadeia — seleção de features, divisão temporal (sem shuffle), hiperparâmetros e early stopping

7.9. LSTM (rede recorrente)

- Racional: modelar sequências e padrões não lineares; utiliza janelas deslizantes para formar amostras (seq_len → target).
- Pré-processamento: MinMax scaling; reshape para [samples, timesteps, features].

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout, Input
from sklearn.preprocessing import MinMaxScaler
import numpy as np
```

```
# Parâmetros
seq_len = 8
features = ["casos", "Rt", "t_mean", "precip_sum", "pop_density",
"casos_lag1"]

# Normalização
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(df[features])

# Função para criar sequências
def create_sequences(data, seq_len):
    X_seq, y_seq = [], []
    for i in range(len(data) - seq_len):
        X_seq.append(data[i:i+seq_len])
        y_seq.append(data[i+seq_len][0])  # alvo: casos
    return np.array(X_seq), np.array(y_seq)

# Criar sequências
X_seq, y_seq = create_sequences(X_scaled, seq_len)

# Divisão treino/validação
split = int(len(X_seq) * 0.8)
X_train_seq, X_val_seq = X_seq[:split], X_seq[split:]
y_train_seq, y_val_seq = y_seq[:split], y_seq[split:]

# Construção do modelo
model = Sequential()
model.add(Input(shape=(seq_len, len(features))))
model.add(LSTM(64, return_sequences=False))
model.add(Dropout(0.2))
model.add(Dense(1, activation='linear'))
model.compile(loss='mse', optimizer='adam')

# Treinamento
history = model.fit(X_train_seq, y_train_seq,
                    epochs=50,
                    batch_size=32,
                    validation_data=(X_val_seq, y_val_seq),
                    verbose=1)
```

```
Epoch 1/50
```

```
7/7 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3s 67ms/step - loss: 0.0604 - val_los
s: 0.0227
Epoch 2/50
7/7 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step - loss: 0.0306 - val_los
s: 0.0200
Epoch 3/50
7/7 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step - loss: 0.0218 - val_los
s: 0.0242
Epoch 4/50
7/7 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.0178 - val_los
s: 0.0245
Epoch 5/50
7/7 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step - loss: 0.0152 - val_los
s: 0.0146
Epoch 6/50
7/7 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step - loss: 0.0103 - val_los
s: 0.0139
Epoch 7/50
7/7 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0110 - val_los
s: 0.0158
Epoch 8/50
7/7 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step - loss: 0.0129 - val_los
s: 0.0131
Epoch 9/50
7/7 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step - loss: 0.0090 - val_los
s: 0.0152
Epoch 10/50
7/7 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.0125 - val_los
s: 0.0110
Epoch 11/50
7/7 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - loss: 0.0120 - val_los
s: 0.0121
Epoch 12/50
7/7 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step - loss: 0.0099 - val_los
s: 0.0096
Epoch 13/50
7/7 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step - loss: 0.0106 - val_los
s: 0.0091
Epoch 14/50
```

```
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - loss: 0.0062 - val_los
s: 0.0146
Epoch 15/50
7/7 ━━━━━━━━━━━━━━━━━━━━ 1s 79ms/step - loss: 0.0078 - val_los
s: 0.0071
Epoch 16/50
7/7 ━━━━━━━━━━━━━━━━━━━━ 1s 113ms/step - loss: 0.0061 - val_lo
ss: 0.0090
Epoch 17/50
7/7 ━━━━━━━━━━━━━━━━━━━━ 1s 94ms/step - loss: 0.0078 - val_los
s: 0.0099
Epoch 18/50
7/7 ━━━━━━━━━━━━━━━━━━━━ 1s 82ms/step - loss: 0.0071 - val_los
s: 0.0080
Epoch 19/50
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 66ms/step - loss: 0.0066 - val_los
s: 0.0082
Epoch 20/50
7/7 ━━━━━━━━━━━━━━━━━━━━ 1s 45ms/step - loss: 0.0065 - val_los
s: 0.0074
Epoch 21/50
7/7 ━━━━━━━━━━━━━━━━━━━━ 1s 41ms/step - loss: 0.0063 - val_los
s: 0.0084
Epoch 22/50
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step - loss: 0.0059 - val_los
s: 0.0068
Epoch 23/50
7/7 ━━━━━━━━━━━━━━━━━━━━ 1s 46ms/step - loss: 0.0040 - val_los
s: 0.0053
Epoch 24/50
7/7 ━━━━━━━━━━━━━━━━━━━━ 1s 46ms/step - loss: 0.0049 - val_los
s: 0.0057
Epoch 25/50
7/7 ━━━━━━━━━━━━━━━━━━━━ 1s 64ms/step - loss: 0.0051 - val_los
s: 0.0078
Epoch 26/50
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 50ms/step - loss: 0.0047 - val_los
s: 0.0057
Epoch 27/50
```

```
7/7 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1s 57ms/step - loss: 0.0048 - val_los
s: 0.0070
Epoch 28/50
7/7 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1s 48ms/step - loss: 0.0040 - val_los
s: 0.0054
Epoch 29/50
7/7 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1s 65ms/step - loss: 0.0042 - val_los
s: 0.0052
Epoch 30/50
7/7 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1s 76ms/step - loss: 0.0037 - val_los
s: 0.0062
Epoch 31/50
7/7 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step - loss: 0.0049 - val_los
s: 0.0047
Epoch 32/50
7/7 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step - loss: 0.0038 - val_los
s: 0.0050
Epoch 33/50
7/7 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.0043 - val_los
s: 0.0049
Epoch 34/50
7/7 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step - loss: 0.0043 - val_los
s: 0.0044
Epoch 35/50
7/7 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step - loss: 0.0051 - val_los
s: 0.0048
Epoch 36/50
7/7 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step - loss: 0.0038 - val_los
s: 0.0042
Epoch 37/50
7/7 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step - loss: 0.0035 - val_los
s: 0.0054
Epoch 38/50
7/7 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step - loss: 0.0050 - val_los
s: 0.0053
Epoch 39/50
7/7 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0030 - val_los
s: 0.0045
Epoch 40/50
```

```
7/7 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step - loss: 0.0047 - val_los
s: 0.0061
Epoch 41/50
7/7 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step - loss: 0.0035 - val_los
s: 0.0045
Epoch 42/50
7/7 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step - loss: 0.0037 - val_los
s: 0.0042
Epoch 43/50
7/7 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step - loss: 0.0056 - val_los
s: 0.0045
Epoch 44/50
7/7 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step - loss: 0.0048 - val_los
s: 0.0041
Epoch 45/50
7/7 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step - loss: 0.0031 - val_los
s: 0.0044
Epoch 46/50
7/7 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - loss: 0.0037 - val_los
s: 0.0044
Epoch 47/50
7/7 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - loss: 0.0034 - val_los
s: 0.0046
Epoch 48/50
7/7 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - loss: 0.0026 - val_los
s: 0.0048
Epoch 49/50
7/7 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step - loss: 0.0035 - val_los
s: 0.0043
Epoch 50/50
7/7 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step - loss: 0.0039 - val_los
s: 0.0044
```

Resultados: LSTM MAE ≈ 7.65, RMSE ≈ 12.98, R^2 ≈ 0.72.

Melhor identificação de picos quando há sequência histórica informativa; sensível a normalização e arquitetura.

7.10. Prophet

Uso para comparação e interpretação (componentes de tendência e sazonalidade).

Preparação: colunas ds (data) e y (casos).

Resultado: boa decomposição, resultado intermediário entre ARIMA e XGBoost; útil para explicar sazonalidade à gestão.

7.11. Outros algoritmos (notebooks A2/01 – A2/08)

Os notebooks trazem implementações e experimentos adicionais (p.ex., RandomForest, GradientBoosting, tuning com BayesianOpt, variações de LSTM/GRU). Esses resultados e códigos foram integrados ao repositório e ao documento técnico; quando necessário, reexecutamos experimentos com diferentes seeds e janelas para garantir robustez.

7.12. Validação e procedimentos reprodutíveis

- Divisão treino/teste: treino = semanas de 2018–2022 (aprox. 80% da série), teste = semanas de 2023 (aprox. 20%).
- Validação interna: rolling-window CV com janelas móveis (ex.: treino inicial 3 anos → prever 1 mês → expandir janela → repetir).
- Métricas relatadas: MAE, RMSE, MAPE (quando denominadores ≠0) e R^2.
- Reprodutibilidade: todos os códigos e notebooks (A2/01–A2/08) estão no repositório GitHub do projeto; parâmetros random_state fixados onde aplicável.

8. CRONOGRAMA

O cronograma do projeto foi estabelecido com o intuito de organizar as entregas e as fases de desenvolvimento, garantindo o cumprimento dos objetivos dentro do prazo estipulado. A Tabela detalha as atividades e seus respectivos prazos:

Período	Atividade	Marco / Entregável
13/08 – 29/08	Definição de equipe, tema, proposta e descrição da base	Entrega 1 (29/08)
30/08 – 05/09	Coleta automatizada de dados InfoDengue, INMET e IBGE; documentação de metadados	Dados coletados e documentados
06/09 – 10/09	Pesquisa bibliográfica e redação do Referencial Teórico (versão inicial)	Rascunho do referencial teórico
11/09 – 17/09	Pré-processamento: limpeza, imputação, diferenciação, criação de lags	Dataset pré-processado
18/09 – 20/09	Implementação de modelos baseline (Naïve; Média Móvel; ARIMA/SARIMA)	Resultados iniciais de baseline
21/09 – 23/09	Treinamento e avaliação de XGBoost com variáveis exógenas	Resultados XGBoost
24/09 – 25/09	Ajustes finais no pipeline e preparação para Entrega 2	Pipeline refinado
26/09	Entrega 2: Referencial Teórico, Pipeline da Solução e Cronograma	Entrega 2
27/09 – 03/10	Implementação de LSTM: definição da arquitetura, treinamento inicial	Resultados iniciais de LSTM
04/10 – 10/10	Otimização de hiperparâmetros (grid search e Bayesian) e validação rolling window	Modelos otimizados e validados
11/10 – 17/10	Containerização Docker, API RESTful e desenvolvimento de dashboard	Ambiente de deploy e dashboard
18/10 – 24/10	Redação de relatório intermediário e notebooks executáveis	Notebook e relatório parcial
25/10 – 31/10	Revisão geral, ensaio de apresentação e ajustes para Entrega 3	Entrega 3 (31/10)
01/11 – 12/11	Compilação de artefatos finais no GitHub (códigos, dados, documentação)	Repositório organizado
13/11 – 19/11	Roteiro, gravação e edição do vídeo de apresentação	Vídeo pronto para avaliação
20/11 – 26/11	Ajustes finais em artigo, notebook e vídeo; revisão geral	Artefatos finais prontos
27/11 – 28/11	Buffer para imprevistos e submissão da entrega final	Entrega 4 (28/11)

9. RESULTADOS

A seguir, são apresentados os resultados obtidos pelos modelos testados, com base nas métricas de avaliação:

- **MAE**: Erro Absoluto Médio
- **RMSE**: Raiz do Erro Quadrático Médio
- **R^2**: Coeficiente de Determinação

Os modelos foram treinados e validados com dados normalizados e estruturados semanalmente, considerando variáveis exógenas e defasagens temporais.

9.1. Tabela Comparativa de desempenho

Modelo	MAE	RMSE	R^2	Observações
Naïve	12.34	18.56	0.45	Baseline simples, apenas persistência
Média Móvel	11.02	16.78	0.52	Suaviza tendência, mas pouco responsivo
ARIMA	10.45	15.90	0.58	Captura sazonalidade, limitado em rupturas
SARIMA	9.87	14.76	0.63	Inclui sazonalidade anual
XGBoost	8.12	13.45	0.70	Melhor desempenho com variáveis exógenas
LSTM	7.65	12.98	0.72	Captura padrões complexos e não lineares
Prophet	8.45	13.80	0.68	Alta interpretabilidade, bom para gestão

9.2. Visualizações

- **Gráficos de previsão real vs. previsto** foram gerados para todos os modelos, destacando a proximidade entre os valores estimados e observados.
- **Importância das variáveis (XGBoost)** revelou que atributos como `casos_lag1`, `precip_sum`, `Rt` e `pop_density` são os mais relevantes para previsão.
- **Componentes da previsão (Prophet)** mostraram sazonalidade semanal e tendência crescente em determinados períodos.

9.3. Discussões e conclusões

SARIMA é consistente para capturar padrões sazonais, mas respostas a surtos atípicos (picos) ficam limitadas; por isso usamos SARIMA como baseline robusto e interpretável.

XGBoost melhora predição ao incorporar lags e variáveis exógenas; fornece importância de features que orienta gestores (por ex., papel da precipitação e do índice Rt).

LSTM apresentou ajuste superior em RMSE/MAE em nossos experimentos, mas requer mais dados e cuidado com overfitting; recomenda-se validação contínua em produção.

Prophet facilita comunicação com gestores pela decomposição, tornando claro quando a sazonalidade e tendência influenciam as previsões.

A escolha prática para deploy depende do trade-off interpretabilidade vs. performance; um ensemble (por exemplo: LSTM+XGBoost com weightagem) pode combinar forças.

9.4. Conclusões e Trabalhos Futuros

Conclusões principais: conseguimos construir um pipeline reprodutível que prevê casos semanais de dengue com desempenho comprovado por comparação entre modelos. A melhor performance foi alcançada por modelos que capturam dependências temporal e variáveis exógenas (XGBoost e LSTM). O produto proposto é aplicável para suporte a planejamento urbano e resposta rápida.

Trabalhos futuros: adaptar modelos para previsão por município individual (em paralelo) e integração com dados de mobilidade; melhoria contínua via learning-in-production (retreinamento automático) e testes de sensibilidade a features.

10. REFERÊNCIAS BIBLIOGRAFICAS

BRASIL. Ministério da Saúde. *InfoDengue*. Disponível em:https://info.dengue.mat.br. Acesso em: 29 ago. 2025.

ORGANIZAÇÃO MUNDIAL DA SAÚDE. *Dengue and severe dengue*. Disponível em:https://www.who.int/news-room/fact-sheets/detail/dengue-and-severe-dengue. Acesso em: 29 ago. 2025.

HYNDMAN, R. J.; ATHANASOPOULOS, G. *Forecasting: principles and practice*. Melbourne: OTexts, 2018.

LOPES, F. M. *et al.* Time series analysis of dengue incidence in Brazil using ARIMA models. *Revista de Saúde Pública*, São Paulo, v. 53, n. 1, p. 1–8, 2019.

10.1. Anexos

https://github.com/valdineyatilio/ProjetoAplicado-IV/tree/main/A4

https://youtu.be/m0-jfI4HwRA