Fortaleza LSTM infodengue

Ivina Lorena Oliveira Moura

May 2025

1 Dengue nas Américas

A dengue é uma infecção viral transmitida por mosquitos contaminados, que pode atingir indivíduos de qualquer idade. Os sintomas variam de febre leve a intensa, com dores de cabeça, atrás dos olhos, musculares, nas articulações e manchas na pele. Em casos graves, pode levar a choque, dificuldade para respirar e falência de órgãos. A doença segue um padrão sazonal, com maior incidência no verão e período chuvoso — no Hemisfério Sul, predomina no início do ano, enquanto no Hemisfério Norte, no segundo semestre (Org24b). Segundo as informações da (Org24a) nas Américas a incidência da dengue cresceu drasticamente nas últimas décadas.

Segundos as estatísticas:

- Cerca de 500 milhões de pessoas nas Américas correm o risco de contrair dengue.
- Os quatro sorotipos da dengue (DENV-1, DENV-2, DENV-3 e DEN-V 4) circulam pelas Américas e, em alguns casos, simultaneamente.
- Nas Américas, o Aedes aegypti é o mosquito vetor da dengue e está amplamente distribuído por todo o território.

A Organização Mundial de Saúde (OMS) apresenta uma grande preocupação com o aumento de casos no Brasil, esse que lidera o número de casos de dengue no mundo, com 2,9 milhões registrados em 2023 (Tok23).

A crise climática, com o aumento das temperaturas globais, é apontada como um dos principais fatores para essa elevação, pois permite que o mosquito Aedes aegypti, vetor da dengue, sobreviva em locais antes desfavoráveis. O fenômeno El Niño de 2023 também intensificou os impactos do aquecimento global e das mudanças climáticas. Em nível mundial, a OMS registrou mais de 5 milhões de infecções e 5 mil óbitos por dengue. A maior parte desses casos, cerca de 80% ou 4,1 milhões, foi reportada nas Américas, seguidas pelo Sudeste Asiático e Pacífico Ocidental. No continente americano, o Brasil lidera em número de casos, seguido por Peru e México. Os dados abrangem o período de 1º de janeiro a 11 de dezembro (Org24b, Tok23)

2 Aplicação da LSTM nos dados de Fortaleza

A LSTM é um tipo especial de rede neural recorrente (RNN) projetada para aprender dependências de longo prazo em sequências de dados. A rede possui uma estrutura em cadeia que contém quatro redes neurais e diferentes blocos de memória chamados células. A informação é retida pelas células e as manipulações de memória são feitas pelos portões (*gates*) (Dee23). Sobre o fluxo de operação dos *gates*:

- Portão de esquecimento (*forget gate*): decide o que esquecer.
- Portão de entrada (*input gate*): decide o que armazenar.
- Portão de saída (*output gate*): decide o que enviar para a próxima etapa.

Nesse contexto, com o objetivo de analisar e prever a incidência de casos de dengue, foi desenvolvida uma aplicação de rede neural recorrente (LSTM) utilizando dados da cidade de Fortaleza-CE, disponíveis por meio da plataforma InfoDengue (Inf24).

Algorithm 1 Operações em uma célula LSTM

Require: Entrada atual x_t, estado oculto anterior h_{t-1}, estado da célula anterior C_{t-1}
Ensure: Novo estado oculto h_t, novo estado da célula C_t

1: $f_t \leftarrow \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$ ▷ Portão de esquecimento
2: $i_t \leftarrow \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$ ▷ Portão de entrada
3: $\tilde{C}_t \leftarrow \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$ ▷ Candidata a novo estado
4: $C_t \leftarrow f_t * C_{t-1} + i_t * \tilde{C}_t$ ▷ Atualização do estado da célula
5: $o_t \leftarrow \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$ ▷ Portão de saída
6: $h_t \leftarrow o_t * \tanh(C_t)$ ▷ Atualização do estado oculto

2.1 Fonte dos dados

Os dados utilizados nesta aplicação foram extraídos da API do InfoDengue, contendo registros semanais de casos estimados e reportados de dengue no município de Fortaleza, Ceará. Para esta análise, os dados foram agregados por mês, visando uma melhor estruturação para modelagem com redes neurais.

2.2 Pré-processamento

Para inicio do pré-processamento os dados foram lidos e colocados em ordem, de 2014 para 2024, abrangendo 10 anos de dados para a rede que será modelada.

```
1
2 df = pd.read_csv("Fortaleza-dengue␣(2).csv")
3 df=df.sort_values(by='data_iniSE')
4
5 # Seleciona apenas a coluna de casos estimados e datas
6 df_modificado = df[['casos_est']].copy()
7 datas = df[['data_iniSE']].copy()
```

Após a coleta dos dados, realizou-se a normalização dos valores do banco de dados. Essa etapa garante que o modelo de aprendizado de máquina consiga aprender de forma mais eficaz. Ao ajustar os dados para uma escala comum, geralmente entre 0 e 1, a normalização simplifica significativamente o processo de aprendizagem do modelo.

```
1 normalizador = MinMaxScaler(feature_range=(0, 1))
2 df_normalizado = normalizador.fit_transform(df_modificado)
```

Para usar LSTM ou outros modelos de séries temporais, é obrigatório criar "janelas" de dados. Isso porque esses modelos aprendem olhando para sequências de dados para prever o futuro. Então, os dados precisam ser organizados como sequências de valores passados para prever o próximo. Aqui, os dados foram modificados para usar essas "janelas deslizantes". No código:

```
1
2 # Criacao de janelas de tempo
3 window_size = 4  # 4 semanas (1 mes)
4 previsao = []
5 valor_real = []
6
7 for i in range(window_size, len(df_normalizado)):
8     janela = df_normalizado[i-window_size:i, 0]
9     previsao.append(janela)
10    valor_real.append(df_normalizado[i, 0])
11
12 # Converte para arrays numpy
13 previsao = np.array(previsao)
14 valor_real = np.array(valor_real)
```

O modelo LSTM espera os dados de entrada na forma [amostras, passos de tempo características], pois cada **amostra** representa uma sequência temporal completa, os ***timesteps*** define quantos pontos no tempo o modelo deve considerar de forma sequencial para aprender um padrão e as ***features*** define cada ponto no tempo onde pode ter uma ou mais variáveis. então:

```
previsao = np.reshape(previsao, (previsao.shape[0], previsao.shape[1], 1))
```

Inicialmente, dividimos o conjunto entre treinamento e teste, nessa implementação definimos 80% para treinamento e 20% dos dados para teste e separamos dos dados em treino e teste.

```
# Divide em treino e teste
tam_treinamento = int(len(previsao) * 0.8)
X_treinamento = previsao[:tam_treinamento]
y_treinamento = valor_real[:tam_treinamento]
x_teste = previsao[tam_treinamento:]
y_teste = valor_real[tam_treinamento:]
```

Para construir e treinar o modelo, foi usado um modelo simples do Keras. Esse modelo empilha as camadas uma após a outra. A primeira camada do tipo LSTM tem 100 neurônios. A 'forma da entrada' (input shape) define como os dados vão entrar, como o número de momentos no tempo e uma informação por momento (por exemplo, casos de dengue a cada mês). O 'return_sequences=True' é importante porque outras camadas LSTM serão adicionadas depois, então a saída precisa continuar sendo uma sequência, e não só o último resultado." O Dropout(0.3) serve para evitar overfitting, "desligando" 30% dos neurônios aleatoriamente durante o treino.

```
modelo = Sequential()
modelo.add(LSTM(units=100, return_sequences=True, input_shape=(previsao.shape
    [1], 1)))
modelo.add(Dropout(0.3))

modelo.add(LSTM(units=50, return_sequences=True))
modelo.add(Dropout(0.3))

modelo.add(LSTM(units=50))
modelo.add(Dropout(0.3))
modelo.add(Dense(units=1))

modelo.compile(optimizer='adam',
               loss='mean_squared_error',
               metrics=[MeanAbsoluteError(), RootMeanSquaredError()])


modelo.fit(X_treinamento, y_treinamento, batch_size=32, epochs=100, verbose=1)
```

Em seguida ocorre a previsão dos conjuntos de treinamento, teste e do conjunto de semanas futuras:

```
previsao_treinamento_lstm = modelo.predict(X_treinamento)
previsao_treinamento_desnormalizada = normalizador.inverse_transform(
    previsao_treinamento_lstm)
y_treinamento_desnormalizado = normalizador.inverse_transform(y_treinamento.
    reshape(-1, 1))


previsao_lstm = modelo.predict(x_teste)
previsao_teste_desnormalizada = normalizador.inverse_transform(previsao_lstm)
y_teste_desnormalizado = normalizador.inverse_transform(y_teste.reshape(-1, 1)
    )
previsao_futuro = modelo.predict(previsao)
```

- Cálculo das métricas

Cálculo de métricas para treinamento

Mean_absolute_error: calcula o erro absoluto médio entre os valores reais e previstos.

Mean_squared_error: calcula o erro quadrático médio entre os valores reais e previstos.

O erro absoluto médio (MAE) é a média dos erros absolutos entre os valores reais e previstos.

```
mae_treinamento = mean_absolute_error(y_treinamento_desnormalizado,
    previsao_treinamento_desnormalizada)
rmse_treinamento = np.sqrt(mean_squared_error(y_treinamento_desnormalizado,
    previsao_treinamento_desnormalizada))
mape_treinamento = np.mean(np.abs((y_treinamento_desnormalizado -
    previsao_treinamento_desnormalizada) / y_treinamento_desnormalizado)) *
    100

mae_teste = mean_absolute_error(y_teste_desnormalizado,
    previsao_teste_desnormalizada)
rmse_teste = np.sqrt(mean_squared_error(y_teste_desnormalizado,
    previsao_teste_desnormalizada))
mape_teste = np.mean(np.abs((y_teste_desnormalizado -
    previsao_teste_desnormalizada) / y_teste_desnormalizado)) * 100

print(f"\nTotal de amostras: {len(df_normalizado)}")
print(f"Treinamento: {len(X_treinamento)} amostras ({len(X_treinamento)/len(
    previsao):.1%})")
print(f"Teste: {len(x_teste)} amostras ({len(x_teste)/len(previsao):.1%})")

print("\nM tricas de TREINAMENTO:")
print(f"MAPE: {mape_treinamento:.2f}% | RMSE: {rmse_treinamento:.2f} | MAE: {
    mae_treinamento:.2f}")

print("\nM tricas de TESTE:")
print(f"MAPE: {mape_teste:.2f}% | RMSE: {rmse_teste:.2f} | MAE: {mae_teste:.2f
    }")
```

Neste estágio, é realizada a programação necessária para gerar previsões futuras, abrangendo um horizonte de 104 semanas, equivalente a aproximadamente dois anos à frente.

```
n = 104
janela_atual = df_normalizado[-window_size:].reshape(1, window_size, 1)
previsoes_futuras = []
```

Em seguida, realiza-se a geração de previsões futuras de forma incremental, isto é, semana a semana, utilizando exclusivamente as saídas anteriores do próprio modelo, sem recorrer a dados reais futuros. Essa abordagem é conhecida como predição recursiva (ou autopredição), sendo amplamente empregada em séries temporais para simular a evolução dos dados com base nas previsões previamente obtidas.

```
    for _ in range(n):
    #modelo.predict(...) retorna uma matriz de previs o, por isso:
    #[0][0] pega o primeiro valor da previs o.
    #Armazena a previs o na vari vel proxima_pred.
    proxima_pred = modelo.predict(janela_atual, verbose=0)[0][0]
    #est  guardando todas as previs es futuras sequenciais.
    previsoes_futuras.append(proxima_pred)
    janela_atual = np.append(janela_atual[:, 1:, :], [[[proxima_pred]]], axis
        =1)
    #Desnormaliza as previsoes futuras
    previsoes_futuras = np.array(previsoes_futuras).reshape(-1, 1)
    previsoes_futuras_desnormalizadas = normalizador.inverse_transform(
        previsoes_futuras)

    #Geracao de datas futuras
    ultima_data = pd.to_datetime(datas.iloc[-1, 0])
    datas_futuras = [ultima_data + datetime.timedelta(weeks=i+1) for i in
        range(n)]
```

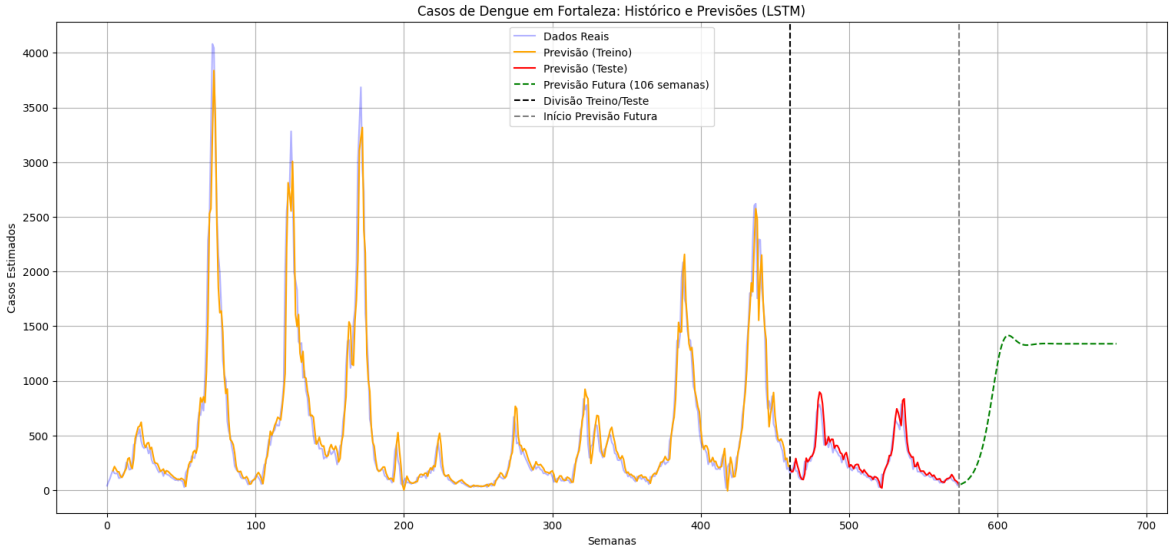
Por fim, realiza-se a plotagem consolidada contendo os dados históricos, as previsões sobre os conjuntos de treino e teste, bem como as projeções futuras. As previsões para as semanas futuras são então salvas em um arquivo no formato CSV para posterior análise.

```
plt.figure(figsize=(16, 6))

# 1. Todos os dados reais (hist rico)
plt.plot(normalizador.inverse_transform(df_normalizado), color='blue', alpha
    =0.3, label='Dados Reais')

# 2. Previs o do conjunto de treinamento
train_range = range(window_size, window_size + len(
    previsao_treinamento_desnormalizada))
plt.plot(train_range, previsao_treinamento_desnormalizada, color='orange',
    label='Previs o (Treino)')

# 3. Previs o do conjunto de teste
test_start = window_size + tam_treinamento
test_range = range(test_start, test_start + len(previsao_teste_desnormalizada)
    )
plt.plot(test_range, previsao_teste_desnormalizada, color='red', label='
    Previs o (Teste)')

# 4. Previs o futura
# Coloca as previs es futuras na sequ ncia dos dados j  existentes
futuro_start = len(df_normalizado)
futuro_range = range(futuro_start, futuro_start + len(
    previsoes_futuras_desnormalizadas))
plt.plot(futuro_range, previsoes_futuras_desnormalizadas, color='green',
    linestyle='--', label='Previs o Futura (52 semanas)')

# Linha divis ria treino/teste
plt.axvline(x=tam_treinamento + window_size, color='black', linestyle='--',
    label='Divis o Treino/Teste')

# Linha divis ria dados reais/previs o futura
plt.axvline(x=len(df_normalizado)-1, color='gray', linestyle='--', label='
    In cio Previs o Futura')

plt.title('Casos de Dengue em Fortaleza: Hist rico e Previs es (LSTM)')
plt.xlabel('Semanas')
plt.ylabel('Casos Estimados')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()

previsoes_futuras_df = pd.DataFrame({
    'Data':datas_futuras,
    'Previs o': previsoes_futuras_desnormalizadas.flatten()
})
previsoes_futuras_df.to_csv('previsoes_futuras_dengue.csv', index=False)
```

3 Resultados

Como resultado, para os dados semanais de Fortaleza no período de 2014 até Dezembro de 2024 e as previsões das próximas 104 semanas:

Figure 1: Casos de dengue da cidade de Fortaleza- histórico e previsões



Elaboração própria.

- Como resultados

Total de amostras: 575
Treinamento: 456 amostras (79.9%)
Teste: 115 amostras (20.1%)
Métricas de treinamento:
MAPE: 25.36% — RMSE: 160.31 — MAE: 88.68
Métricas de teste: MAPE: 23.81% — RMSE: 69.99 — MAE: 49.77
Link do projeto: GitHub.

References

aaa

Dee23 Deep Learning Book Brasil. Arquitetura de redes neurais long short-term memory (lstm), 2023. Acessado em 19 de maio de 2025.

Inf24 InfoDengue. Sistema de alerta de epidemias de dengue, zika e chikungunya. ⟨https://info.dengue.mat.br⟩, 2024. Acesso em: 19 maio 2025.

Org24a Pan American Health Organization. Dengue: Cases by country and year, 2024. Accessed: 2025-05-19.

Org24b Organização Pan-Americana da Saúde. Dengue, 2024. Acesso em: 19 maio 2025.

Tok23 Mariana Tokarnia. Brasil é país com mais casos de dengue no mundo, alerta OMS, 2023. Publicado pela Agência Brasil. Acesso em: October 1, 2025.