```
!pip install prophet tensorflow
```

Show hidden output

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

#prophet
from prophet import Prophet
from prophet.plot import plot_plotly, plot_components_plotly

#tensorflow
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout

#sklearn
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

```
def gen_url_resorce(base_url, resorce_pack, resource_id, year=0):
  if year !=0:
    resource_id = resource_id.replace("(year)", str(year))
  url = base_url + resorce_pack + resource_id
  return url

def resorce_in_year_interval(base_url, resorce_pack, resource_mask, start_year, end_year):
  df = pd.DataFrame()
  for year in range(start_year, end_year + 1):
    url = gen_url_resorce(base_url, resorce_pack, resource_mask, year)
    df_year = pd.read_csv(url, sep=';')
    df = pd.concat([df, df_year])
  return df

base_url =  "https://ons-aws-prod-opendata.s3.amazonaws.com/dataset/"
```

Recolhimento e Pre-processamento dos Dados

Endereços padronizados dos endpoints para coleta facilitada e contínua dos dados, permitindo consulta das bases de séries temporais anuais.

```
#dados hidrológicos diários
resorce_pack_dh_di = "dados_hidrologicos_di/"
resource_mask_dh_di = "DADOS_HIDROLOGICOS_RES_(year).csv"

#dados diários de ENA por reservatório
resorce_pack_ena_di = 'ena_reservatorio_di/'
resource_mask_ena_di = "ENA_DIARIO_RESERVATORIOS_(year).csv"

#dados reservatórios
resource_pack_reservatorios = "reservatorio/"
resource_reservatorio_id = "RESERVATORIOS.csv"
```

Dados referentes a todos os reservatórios cadastrados, levando em consideração apenas instâncias únicas, nescessário devido a replicação dos dados entre os reservatórios P. Afonso 1, 2 e 3.

```
resource_reservatorios_url = gen_url_resorce(base_url, resource_pack_reservatorios, resource_reservato
reservatorios = pd.read_csv(resource_reservatorios_url, sep=';')
#remover duplicatas
reservatorios = reservatorios.drop_duplicates(subset=['cod_resplanejamento'])
reservatorios.head()
```

	nom_reservatorio	tip_reservatorio	cod_resplanejamento	cod_posto	nom_usina	ceg
0	14 DE JULHO	FIO DAGUA	99	284	14 DE JULHO	UHE.PH.RS.000012-4.01

1	A. VERMELHA	RESERVATÓRIO COM USINA	18	18	AGUA VERMELHA	UHE.PH.MG.000041-8.01
2	AIMORES	FIO DAGUA	143	148	AIMORÉS	UHE.PH.MG.000042-6.01
3	ANTA	RESERVATÓRIO SEM USINA	128	129	ANTA	UHE.PH.MG.029458-6.01
4	APOLONIO SALES	FIO DAGUA	173	173	APOLÔNIO SALES	UHE.PH.AL.001510-5.01

5 rows × 24 columns

Consulta e concatenação dos dados diários de Energia Natural Afluente (ENA) por reservatório no período de 2021 a 2025. Esse intervalo foi escolhido devido à proximidade temporal com eventos climáticos significativos, que impactaram diretamente a disponibilidade hídrica na região, como a eventos de seca e cheias recentes. Como pode ser notado nas seguintes notícias.

https://cbhsaofrancisco.org.br/noticias/novidades/mesmo-com-dois-anos-consecutivos-de-cheia-os-problemas-da-bacia-do-sao-francisco-nao-devem-ser-minimizados/

https://manguejornalismo.org/rio-sao-francisco-esta-secando-pesquisa-alerta-que-60-de-sua-vazao-ja-foi-reduzida/

```
resorce_ena_di_interval = resorce_in_year_interval(base_url, resorce_pack_ena_di, resource_mask_ena_di
#exibir not numbers
resorce_ena_di_interval = resorce_ena_di_interval[~pd.to_numeric(resorce_ena_di_interval['cod_resplane
resorce_ena_di_interval['cod_resplanejamento'] = resorce_ena_di_interval['cod_resplanejamento'].astype
print(resorce_ena_di_interval.isnull().sum())
resorce_ena_di_interval.head()
```

```
nom_reservatorio                      0
cod_resplanejamento                   0
tip_reservatorio                      0
nom_bacia                             0
nom_ree                             858
id_subsistema                         0
```

```
nom_subsistema                          0
ena_data                                0
ena_bruta_res_mwmed                  1834
ena_bruta_res_percentualmlt          1834
ena_armazenavel_res_mwmed            1834
ena_armazenavel_res_percentualmlt    1834
ena_queda_bruta                      1834
mlt_ena                              1834
dtype: int64
```

	nom_reservatorio	cod_resplanejamento	tip_reservatorio	nom_bacia	nom_ree	id_subsistema	nom_sub
0	14 DE JULHO	99	Fio dagua	JACUI	SUL	S	
1	A. VERMELHA	18	Reservatório com Usina	GRANDE	PARANA	SE	S
2	AIMORES	143	Fio dagua	DOCE	SUDESTE	SE	S
3	B. BONITA	37	Reservatório com Usina	TIETE	PARANA	SE	S
4	B.COQUEIROS	312	Fio dagua	PARANAIBA	PARANA	SE	S

Porque a estação do ano influencia diretamente o regime de chuvas e, consequentemente, o volume dos reservatórios, sendo essencial para capturar padrões sazonais na previsão de ENA.

```
def get_season_by_date(date):
    month = date.month
    if month in [12, 1, 2]:
        return 1 #verão
    elif month in [3, 4, 5]:
        return 2 #Outono
    elif month in [6, 7, 8]:
        return 3 #inverno
    elif month in [9, 10, 11]:
        return 4 # primavera
    else:
```

```
        return 0
```

Cruza os dados das duas bases de dados permitindo a comparação de séries temporais.

```
def merge_ena_di_reservatorios(df_ena_di, reservatorios):
  df_ena_di_reservatorios = pd.merge(df_ena_di, reservatorios, on='cod_resplanejamento', how='left')
  return df_ena_di_reservatorios
```

Prepara os dados para treinamento filtrando por reservatório e levanto em consideração as features de maior relevância encontradas, sendo elas ENA, (nosso objetivo de predição), a data da coleta do dado(relevante para o estudo da série temporal no modelo Prophet) e a partir desta ano, mes e dia da coleta, o volume d'água calculado sobre o valor de ENA e a produtividade específica de cada reservatório e a estação do ano no hemisfério sul.

```
def prepare_ena_di_reservatorio(df, rese):
  df_ena_di = df.copy()
  df_ena_di = df_ena_di[df_ena_di['cod_resplanejamento'] == rese]
  df_ena_di['ena_data'] = pd.to_datetime(df_ena_di['ena_data'])
  df_ena_di = df_ena_di.rename(columns={'ena_data': 'ds', 'ena_bruta_res_mwmed': 'y'})
  df_ena_di['vol_calc'] = df_ena_di['y'] / df_ena_di['val_produtibilidadeespecifica']
  df_ena_di['season'] = df_ena_di['ds'].apply(get_season_by_date)
  df_ena_di['year'] = df_ena_di['ds'].dt.year
  df_ena_di['month'] = df_ena_di['ds'].dt.month
  df_ena_di['day'] = df_ena_di['ds'].dt.day
  df_ena_di = df_ena_di[['ds', 'y', 'vol_calc', 'season', 'year', 'month', 'day']]
  return df_ena_di
```

```
Start coding or generate with AI.
```

```
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
import matplotlib.pyplot as plt
import seaborn as sns
```

```
def plot_acf_pacf(df, column):
  plt.figure(figsize=(14, 6))
  plt.subplot(1, 2, 1)
  plot_acf(df['y'].dropna(), ax=plt.gca(), lags=365)
  plt.title('Função de Autocorrelação (ACF) da ENA')

  plt.subplot(1, 2, 2)
  plot_pacf(df['y'].dropna(), ax=plt.gca(), lags=365)
  plt.title('Função de Autocorrelação Parcial (PACF) da ENA')

  plt.tight_layout()
  plt.show()
```

```
def normalize_columns(df):
  df_norm = df.copy()
  for col in df.columns:
    if col != 'ds' and col != 'season':
      df_norm[col] = (df[col] - df[col].min()) / (df[col].max() - df[col].min())
  return df_norm
```

Treinamento e Teste do Modelo Prophet

O Prophet é uma biblioteca desenvolvida pelo Facebook (agora Meta) para previsão de séries temporais, especialmente quando há tendências não lineares e forte sazonalidade, sendo fácil de usar e interpretar.

Após o treinamento do modelo, é essencial avaliar sua performance e interpretar os resultados. Para isso, foram implementadas funções auxiliares que calculam métricas de erro e geram visualizações do ajuste e da decomposição dos componentes previstos.

As métricas utilizadas são:

- MAE (Erro Médio Absoluto): mede o desvio médio entre valores reais e previstos.
- RMSE (Raiz do Erro Quadrático Médio): penaliza mais fortemente erros grandes, útil para avaliar robustez do

modelo.

- MAPE (Erro Percentual Médio Absoluto): expressa o erro em termos percentuais, facilitando a interpretação relativa ao valor real.

Além das métricas, também são gerados gráficos do forecast (previsão em relação aos dados reais) e dos componentes do modelo (tendência, sazonalidade anual/semanal e possíveis efeitos de feriados).

Essas ferramentas permitem não apenas quantificar o desempenho do modelo, mas também compreender melhor o comportamento temporal da série de ENA em cada reservatório analisado.

```
def avalute_prophet_model(performance):

  performance['error'] = performance['y'] - performance['yhat']
  performance['absolute_error'] = performance['error'].abs()

  mae = performance['absolute_error'].mean()
  rmse = (performance['error']**2).mean()**0.5

  epsilon = 1e-10
  mape = np.mean(np.abs((performance['y'] - performance['yhat']) / (performance['y'] + epsilon))) * 1

  print("### Métricas de Avaliação do Modelo ###")
  print(f"Erro Médio Absoluto (MAE): {mae:.2f} MWméd")
  print(f"Raiz do Erro Quadrático Médio (RMSE): {rmse:.2f} MWméd")
  print(f"Erro Percentual Médio Absoluto (MAPE): {mape:.2f}%")

  real_mean = performance['y'].mean()
  print(f"\nContexto: O valor médio real de ENA no período foi de {real_mean:.2f} MWméd.")
  print(f"Um erro (MAE) de {mae:.2f} representa aproximadamente {(mae/real_mean*100):.2f}% do valor r

  return mae, rmse, mape

def plot_forecast(model, forecast, test_data, rese):
  fig1 = model.plot(forecast, xlabel='Date', ylabel='ENA (Normalized)')
  fig1.gca().set_title(f'Forecast for {rese}')
  fig1.gca().plot(test_data['ds'], test_data['y'], 'o', color='red', markersize=2, label='Real')
```

```
fig1.gca().plot(test_data['ds'], test_data['y'], 'o', color='red', markersize=2, label='Real')
fig1.gca().legend()
fig1.gca().title.set_text(f'Forecast for {rese}')
plt.savefig(f'{rese}_forecast.png')
plt.show()

fig2 = model.plot_components(forecast)
fig2.gca().set_title(f'Forecast Components for {rese}')
plt.savefig(f'{rese}_forecast_components.png')
plt.show()
```

A função `train_prophet_model` treina e testa um modelo Prophet para previsão de séries temporais.

- Normaliza os dados de entrada
- Divide em treino (datas < `limit_date`) e teste (datas ≥ `limit_date`)
- Ajusta o modelo Prophet com os dados de treino
- Gera previsões para o período de teste
- Avalia a performance com métricas (MAE, RMSE, MAPE)
- Plota gráficos de previsão e componentes
- Exporta resultados em CSV e retorna um DataFrame com valores reais e previstos

```
def train_prophet_model(rese_data, rese, limit_date='2025-01-01'):
  y_min = rese_data['y'].min()
  y_max = rese_data['y'].max()
  data = rese_data.copy()
  scaled = normalize_columns(rese_data)
  df_scaled = scaled.copy()
  df_scaled['ds'] = scaled['ds'].values

  train_data = df_scaled[df_scaled['ds'] < limit_date]
  test_data = df_scaled[df_scaled['ds'] >= limit_date]

  m = Prophet()
  m.fit(train_data)
```

```
    periods = len(test_data)
    future = m.make_future_dataframe(periods=periods)

    forecast = m.predict(future)
    forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail()
    performance = pd.merge(rese_data, forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']], on='ds')

    avalute_prophet_model(performance)

    plot_forecast(m,forecast, test_data, rese)

    df_prev = pd.DataFrame({'ds': data[data['ds'] >= limit_date]['ds'], 'ena_di': test_data['y'], 'ena_
    print(df_prev)
    df_prev.to_csv(f'{rese}_prophet_prev.csv', index=False)
    return df_prev
```

Treinamento e Teste do Modelo de LSTM

O código abaixo apresenta funções auxiliares para treinar e avaliar um modelo LSTM na previsão da ENA diária. Elas incluem a criação de janelas temporais da série, a definição da arquitetura da rede (duas camadas LSTM, dropout e saída densa), a plotagem dos resultados reais vs. previstos e o cálculo de métricas de desempenho como MAE, MSE, RMSE, R^2 e MAPE.

```
def create_sequences(data, target_col, window_size=30):
    X, y = [], []
    for i in range(len(data) - window_size):
        X.append(data[i:i+window_size])
        y.append(data[i+window_size, target_col])
    return np.array(X), np.array(y)

def create_model(input_shape):
    model = Sequential()
    model.add(LSTM(64, return_sequences=True, input_shape=input_shape))
```

```
    model.add(LSTM(64, return_sequences=True, input_shape=input_shape))
    model.add(Dropout(0.2))
    model.add(LSTM(32))
    model.add(Dense(1))
    return model

def plot_lstm_results(rese, dates_test, y_test_inv, y_pred_inv):
  plt.figure(figsize=(12,5))
  plt.plot(dates_test, y_test_inv, label="Real", color="blue")
  plt.plot(dates_test, y_pred_inv, label="Previsto", color="red")
  plt.xlabel("Data")
  plt.ylabel("ENA (MWmed)")
  plt.title(f"Previsão LSTM - ENA diária de {rese}")
  plt.legend()
  plt.savefig(f'{rese}_lstm_results.png')
  plt.show()

def mean_absolute_percentage_error(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    # evitar divisão por zero
    mask = y_true != 0
    return np.mean(np.abs((y_true[mask] - y_pred[mask]) / y_true[mask])) * 100

def avalute_lstm_model(y_test_inv, y_pred_inv):
  mae = mean_absolute_error(y_test_inv, y_pred_inv)
  mse = mean_squared_error(y_test_inv, y_pred_inv)
  rmse = np.sqrt(mse)
  r2 = r2_score(y_test_inv, y_pred_inv)
  mape = mean_absolute_percentage_error(y_test_inv, y_pred_inv)

  print(f"MAE: {mae:.2f}")
  print(f"MSE: {mse:.2f}")
  print(f"RMSE: {rmse:.2f}")
  print(f"R²: {r2:.2f}")
  print(f"MAPE: {mape:.2f}%")
```

A função treina um modelo LSTM para prever a ENA diária de um reservatório. Primeiro, normaliza os dados e cria janelas

temporais para capturar o histórico recente. Divide os dados entre treino e teste, ajusta a rede LSTM e salva o modelo. Em seguida, faz previsões, reverte a normalização e registra os valores reais e previstos em um CSV. Por fim, avalia o desempenho com métricas e plota os resultados para visualização.

```
def train_lstm_model(rese_data, rese):
  df_ena = rese_data.copy()

  scaler = MinMaxScaler()
  cols  = ['y', 'vol_calc', 'month', 'day']
  scaled = scaler.fit_transform(df_ena[cols])

  df_scaled = pd.DataFrame(scaled, columns=cols)
  df_scaled['ds'] = df_ena['ds'].values

  # Converter para numpy
  data = df_scaled[cols].values

  # Usando coluna 0 (y = ENA) como alvo
  window_size = 30
  X, y = create_sequences(data, target_col=0, window_size=window_size)
  dates_y = df_ena['ds'].values[window_size:]
  print("Shape X:", X.shape)  # (amostras, janela, features)
  print("Shape y:", y.shape)

  split = int(0.8 * len(X))
  X_train, X_test = X[:split], X[split:]
  y_train, y_test = y[:split], y[split:]
  dates_train, dates_test = dates_y[:split], dates_y[split:]

  model = create_model(input_shape=(X_train.shape[1], X_train.shape[2]))
  model.compile(optimizer='adam', loss='mse')
  history = model.fit(X_train, y_train, epochs=30, batch_size=32,
                      validation_data=(X_test, y_test))
  model.save(f'{rese}_model.keras')

  y_pred = model.predict(X_test)
```

```
    y_test_inv = scaler.inverse_transform(
        np.hstack((y_test.reshape(-1,1),
                   X_test[:, -1, 1:]))  # mantém features vol_calc, month, day
    )[:,0]

    y_pred_inv = scaler.inverse_transform(
        np.hstack((y_pred, X_test[:, -1, 1:]))
    )[:,0]

    df_prev = pd.DataFrame({'ds': dates_test, 'ena_di': y_test_inv, 'ena_di_prev': y_pred_inv, 'rese':
    df_prev.to_csv(f'{rese}_lstm_prev.csv', index=False)

    avalute_lstm_model(y_test_inv, y_pred_inv)
    plot_lstm_results(rese, dates_test, y_test_inv, y_pred_inv)
    return df_prev
```

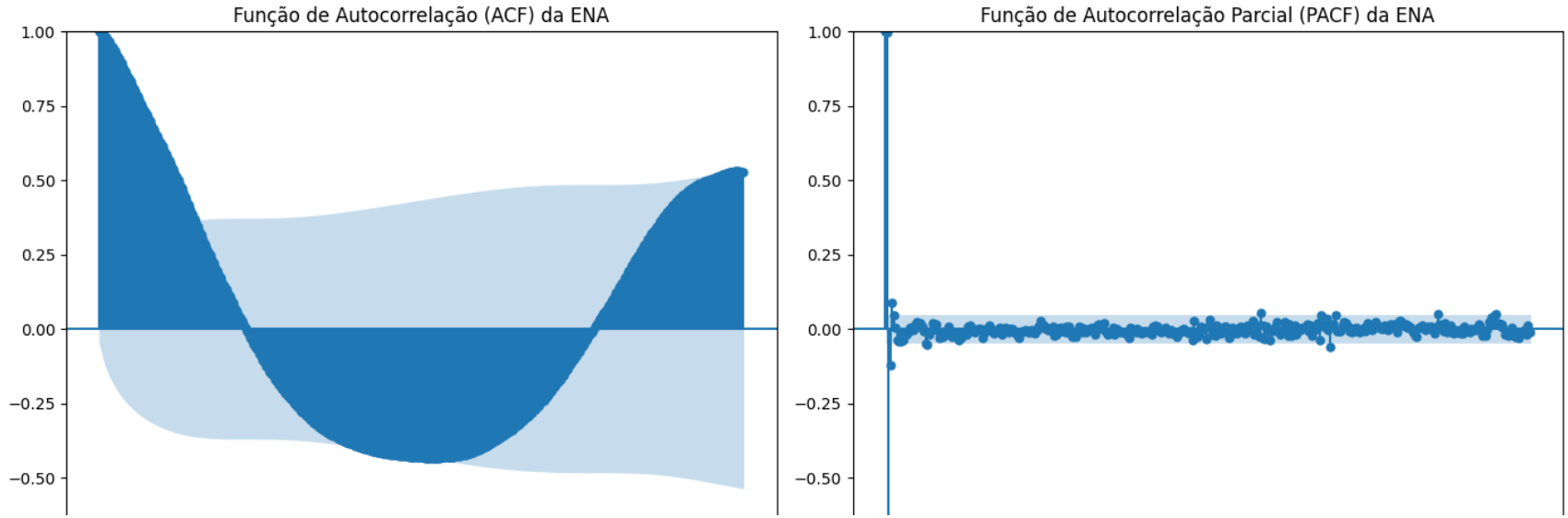
Esse trecho de código realiza o processamento e treinamento automático para cada bacia do Rio São Francisco. De forma prosaica:

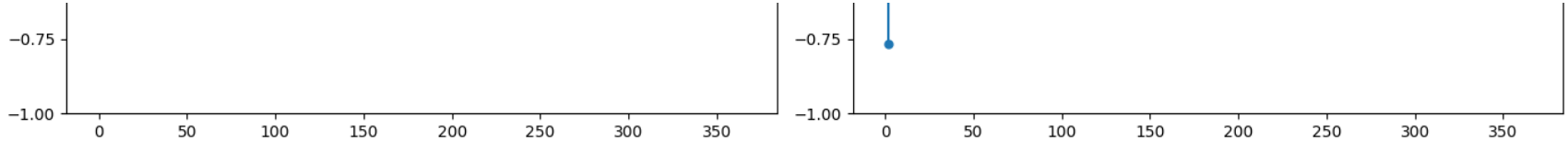
Ele começa mesclando os dados de ENA diária com os reservatórios. Em seguida, define um dicionário com os principais reservatórios do Rio São Francisco, incluindo seus códigos e nomes. Para cada bacia, o código:

- Imprime o nome da bacia atual.
- Prepara os dados específicos da bacia chamando prepare_ena_di_reservatorio.
- Analisa a autocorrelação e autocorrelação parcial da série com plot_acf_pacf.
- Treina o modelo Prophet para prever a ENA diária.
- Treina o modelo LSTM para prever a ENA diária.

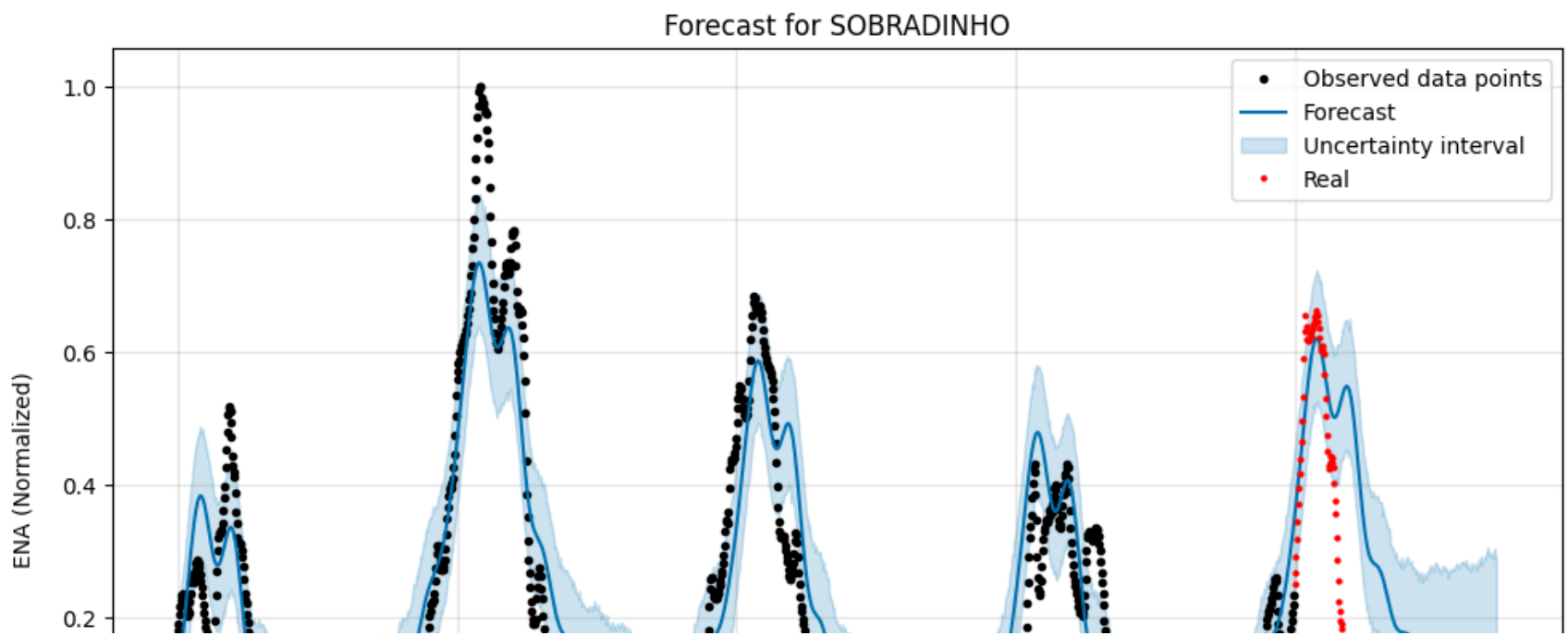
```
ena_res = merge_ena_di_reservatorios(resorce_ena_di_interval, reservatorios)
reservatorios_sao_francisco = {
    169: {"name": "SOBRADINHO", "id": "SFSOBR"},
    174: {"name": "P. AFONSO 1,2,3", "id": "SFP123"},
    172: {"name": "LUIZ GONZAGA", "id": "SFLGON"},
    178: {"name": "XINGO", "id": "SFXING"},
    156: {"name": "TRÊS MARIAS", "id": "SFTMAR"},
    173: {"name": "APOLONIO SALES", "id": "SFMOXO"},
    175: {"name": "P. AFONSO 4", "id": "SFPAF4"}
}


for rese in reservatorios_sao_francisco:
  print(f'Reservatório: {reservatorios_sao_francisco[rese]["name"]}')

  preparared_ena_res = prepare_ena_di_reservatorio(ena_res, rese)
  plot_acf_pacf(preparared_ena_res, 'y')
  rese_name = reservatorios_sao_francisco[rese]["name"]
  train_prophet_model(preparared_ena_res, rese_name)
  train_lstm_model(preparared_ena_res, rese_name)
```

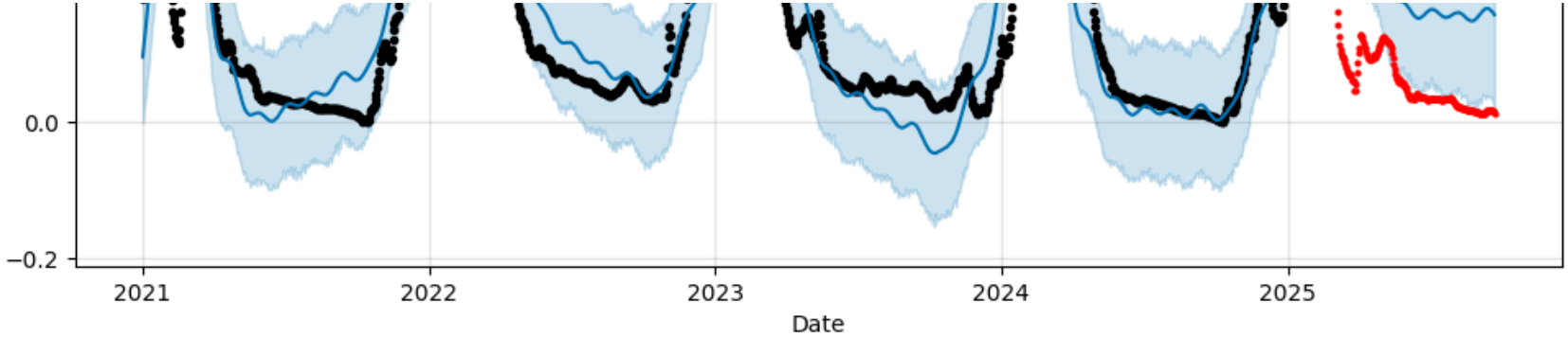
```
Reservatório: SOBRADINHO
```

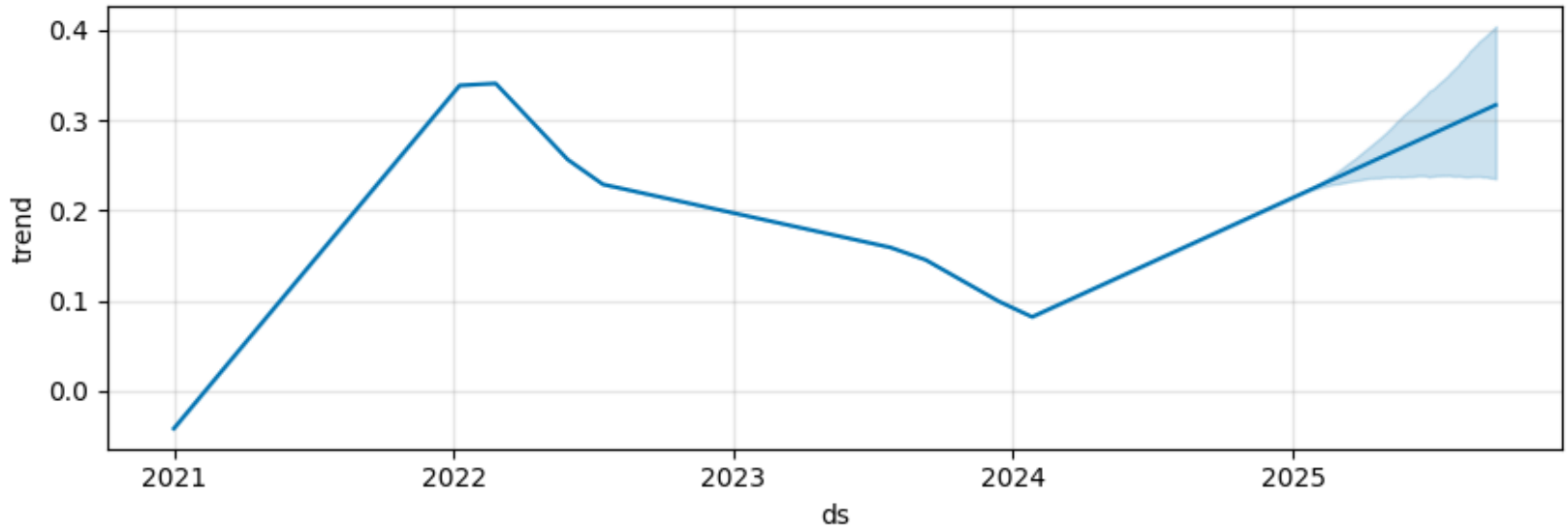


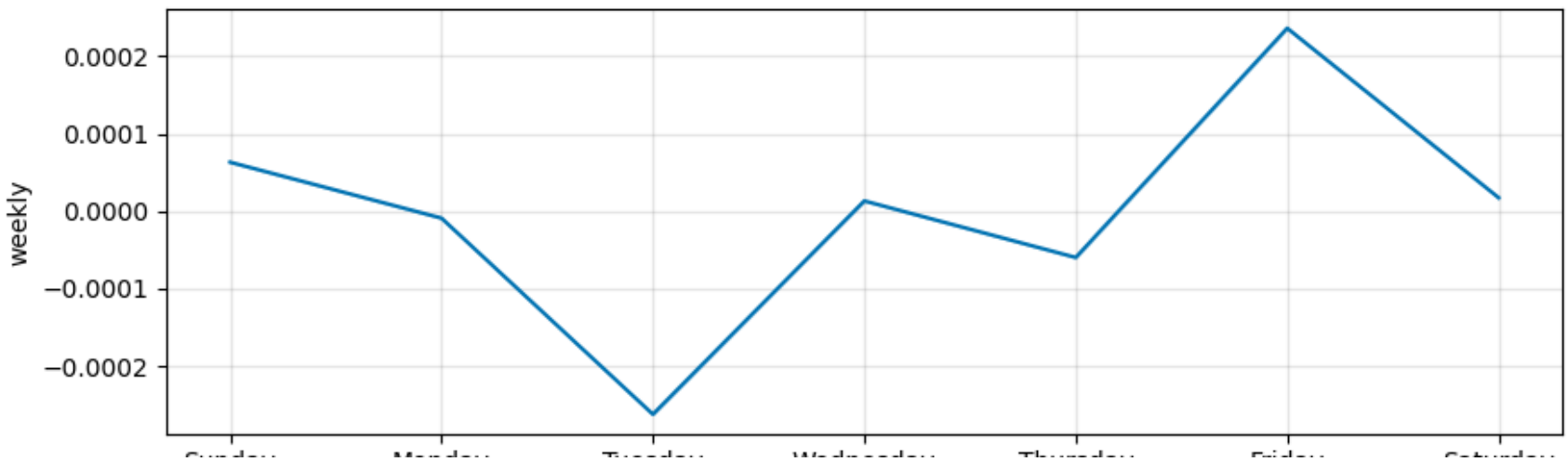


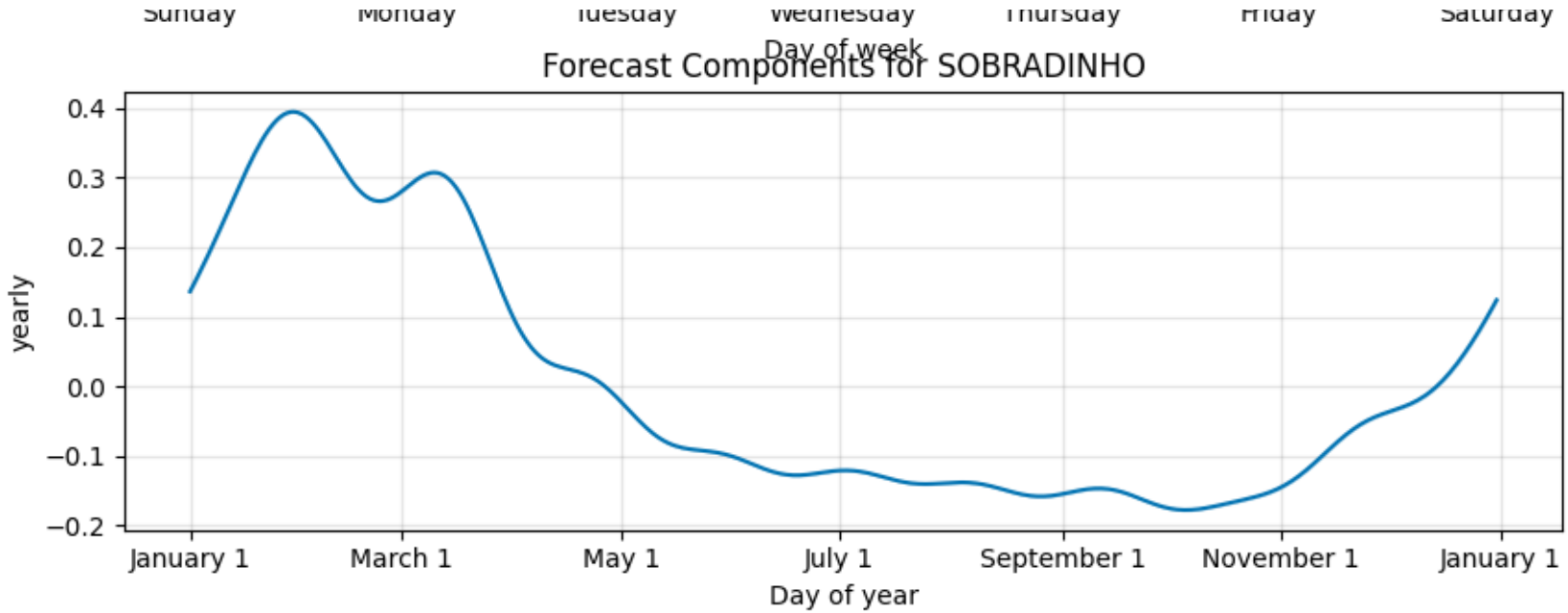

```
INFO:prophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to override this.
DEBUG:cmdstanpy:input tempfile: /tmp/tmp20qzy7c_/nzhm7ei5.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmp20qzy7c_/r79okrb8.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.12/dist-packages/prophet/stan_model/prophet_mo
10:46:26 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
10:46:26 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
### Métricas de Avaliação do Modelo ###
Erro Médio Absoluto (MAE): 414.65 MWméd
Raiz do Erro Quadrático Médio (RMSE): 573.12 MWméd
Erro Percentual Médio Absoluto (MAPE): 99.95%

Contexto: O valor médio real de ENA no período foi de 414.85 MWméd.
Um erro (MAE) de 414.65 representa aproximadamente 99.95% do valor médio.
```




0.0
−0.2
2021
2022
2023
2024
2025
Date


0.4
0.3
0.2
0.1
0.0
trend
2021
2022
2023
2024
2025
ds

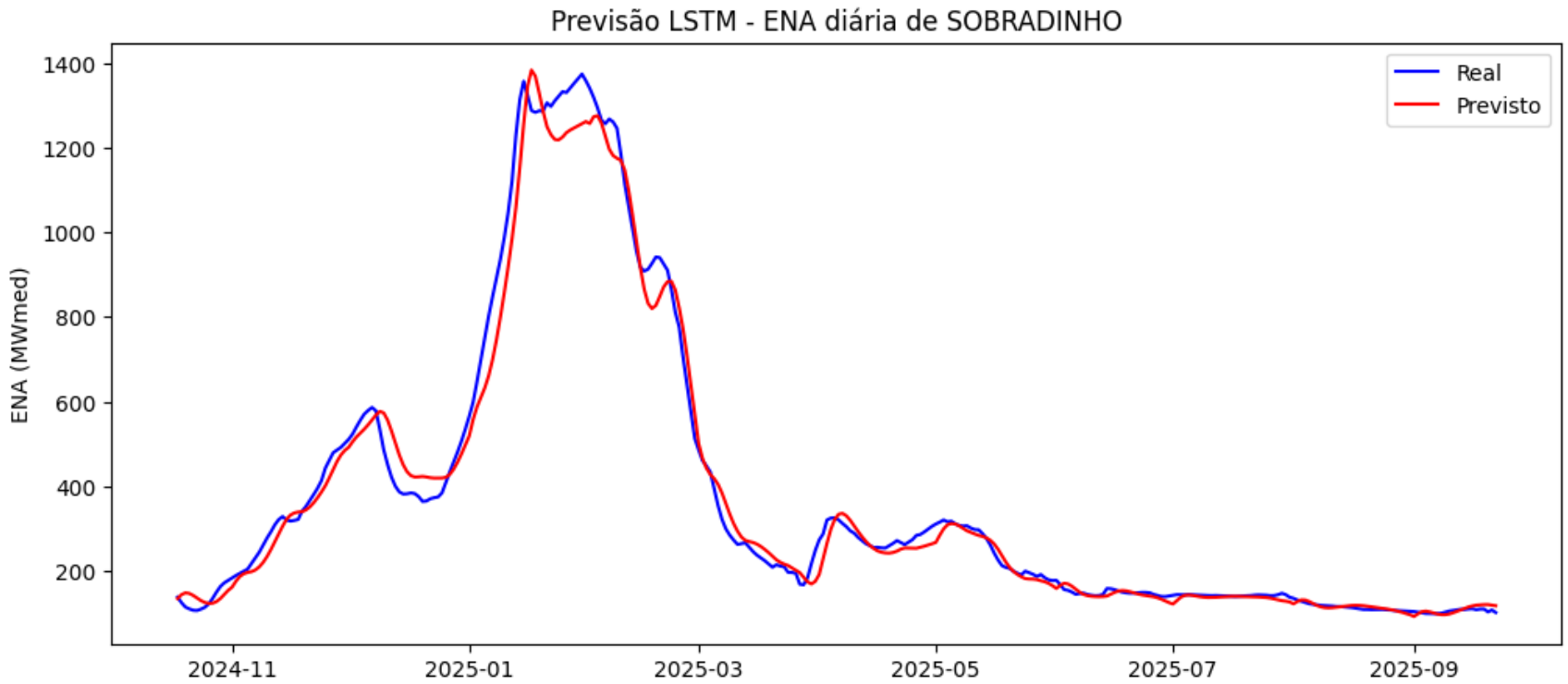

0.0002
0.0001
0.0000
−0.0001
−0.0002
weekly




```
                ds    ena_di  ena_di_prev        rese
0              NaT       NaN     0.095206  SOBRADINHO
1              NaT       NaN     0.105983  SOBRADINHO
2              NaT       NaN     0.117240  SOBRADINHO
3              NaT       NaN     0.128593  SOBRADINHO
4              NaT       NaN     0.139977  SOBRADINHO
...            ...       ...          ...         ...
261423 2025-09-18  0.016570          NaN  SOBRADINHO
261578 2025-09-19  0.016645          NaN  SOBRADINHO
261733 2025-09-20  0.013180          NaN  SOBRADINHO
261888 2025-09-21  0.015481          NaN  SOBRADINHO
262043 2025-09-22  0.012446          NaN  SOBRADINHO

[1991 rows x 4 columns]
Shape X: (1696, 30, 4)
Shape y: (1696,)
Epoch 1/30
/usr/local/lib/python3.12/dist-packages/keras/src/layers/rnn/rnn.py:199: UserWarning: Do not pass an
  super().__init__(**kwargs)
43/43 ━━━━━━━━━━━━━━━━━━━━ 5s 33ms/step - loss: 0.0155 - val_loss: 0.0034
Epoch 2/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0032 - val_loss: 0.0020
Epoch 3/30
```

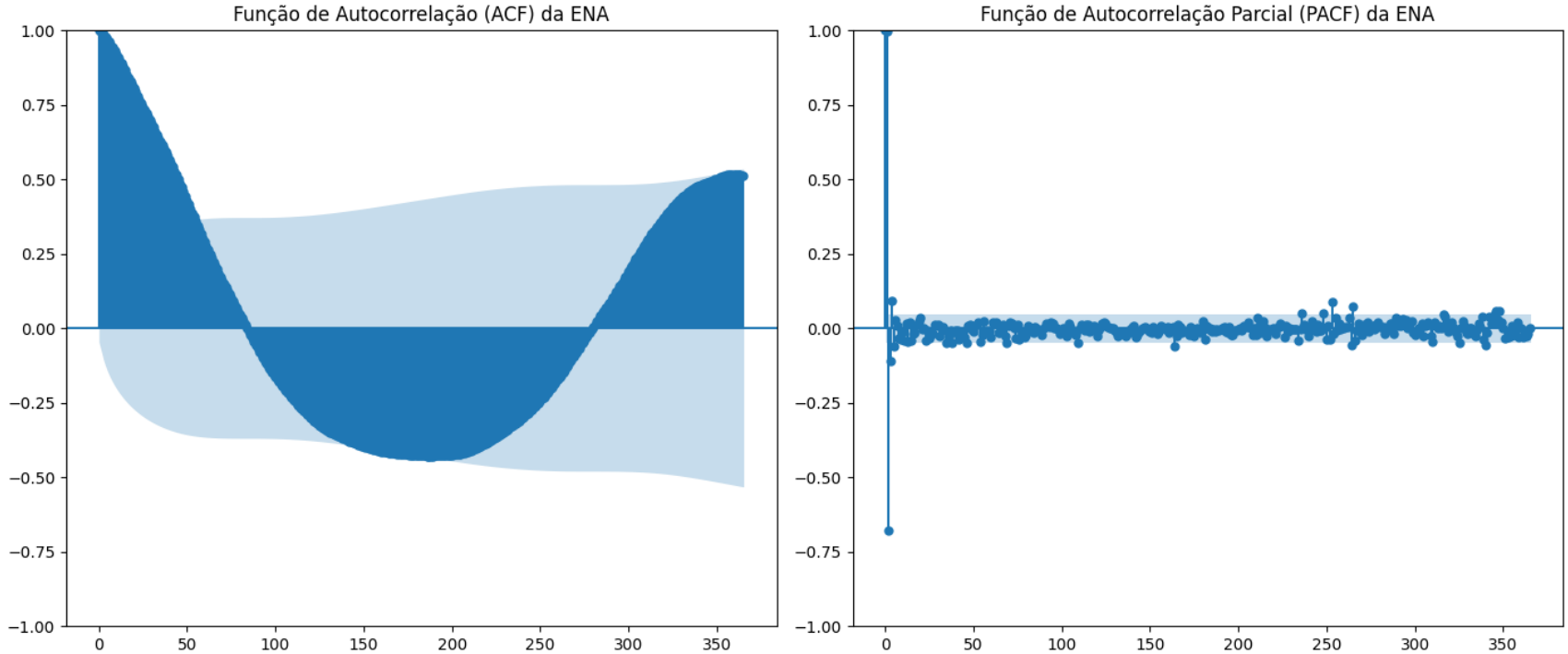
```
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0024 - val_loss: 0.0013
Epoch 4/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0015 - val_loss: 0.0010
Epoch 5/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0016 - val_loss: 9.5085e-04
Epoch 6/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0014 - val_loss: 7.7828e-04
Epoch 7/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0012 - val_loss: 7.1550e-04
Epoch 8/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 30ms/step - loss: 0.0011 - val_loss: 7.3473e-04
Epoch 9/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 40ms/step - loss: 0.0012 - val_loss: 7.2189e-04
Epoch 10/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0012 - val_loss: 6.9258e-04
Epoch 11/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 9.2090e-04 - val_loss: 9.9078e-04
Epoch 12/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 8.7753e-04 - val_loss: 8.7754e-04
Epoch 13/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 8.9639e-04 - val_loss: 6.3070e-04
Epoch 14/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 7.2440e-04 - val_loss: 7.6236e-04
Epoch 15/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 9.0194e-04 - val_loss: 7.9733e-04
Epoch 16/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 8.8693e-04 - val_loss: 8.2179e-04
Epoch 17/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 9.1243e-04 - val_loss: 5.8888e-04
Epoch 18/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 6.2897e-04 - val_loss: 6.5082e-04
Epoch 19/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 38ms/step - loss: 6.8583e-04 - val_loss: 9.0599e-04
Epoch 20/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - loss: 7.3357e-04 - val_loss: 5.5042e-04
Epoch 21/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 5.3893e-04 - val_loss: 7.7801e-04
Epoch 22/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 7.2184e-04 - val_loss: 5.8186e-04
Epoch 23/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 5.2009e-04 - val_loss: 6.7229e-04
```

```
Epoch 24/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 6.6139e-04 - val_loss: 5.2451e-04
Epoch 25/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 5.3078e-04 - val_loss: 4.6649e-04
Epoch 26/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 4.5029e-04 - val_loss: 4.8540e-04
Epoch 27/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 5.0269e-04 - val_loss: 4.5782e-04
Epoch 28/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 5.1778e-04 - val_loss: 4.5505e-04
Epoch 29/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 38ms/step - loss: 4.3791e-04 - val_loss: 4.4658e-04
Epoch 30/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 24ms/step - loss: 4.1598e-04 - val_loss: 4.8727e-04
11/11 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step
MAE: 27.58
MSE: 1862.02
RMSE: 43.15
R²: 0.98
MAPE: 7.21%
```



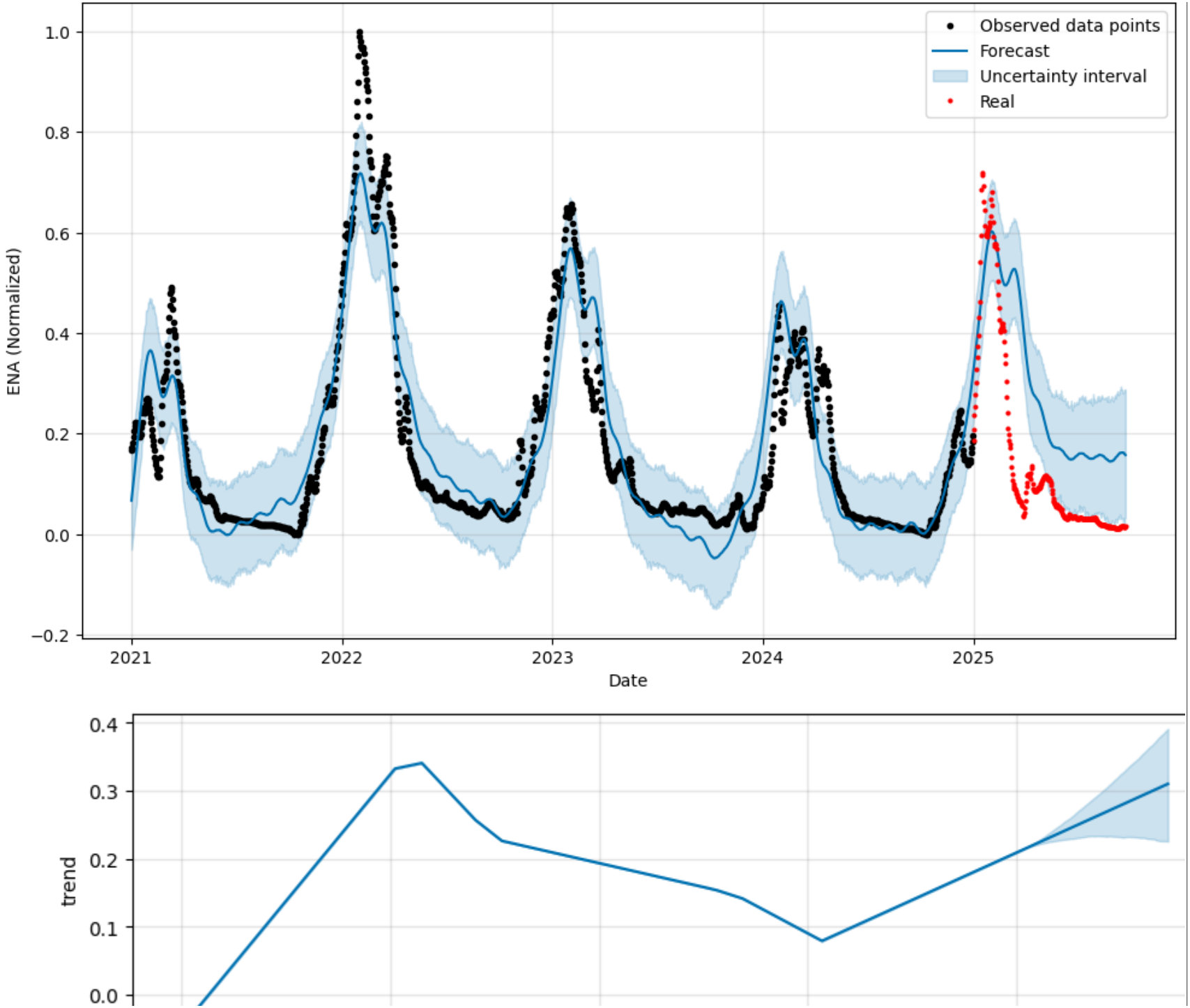
Data

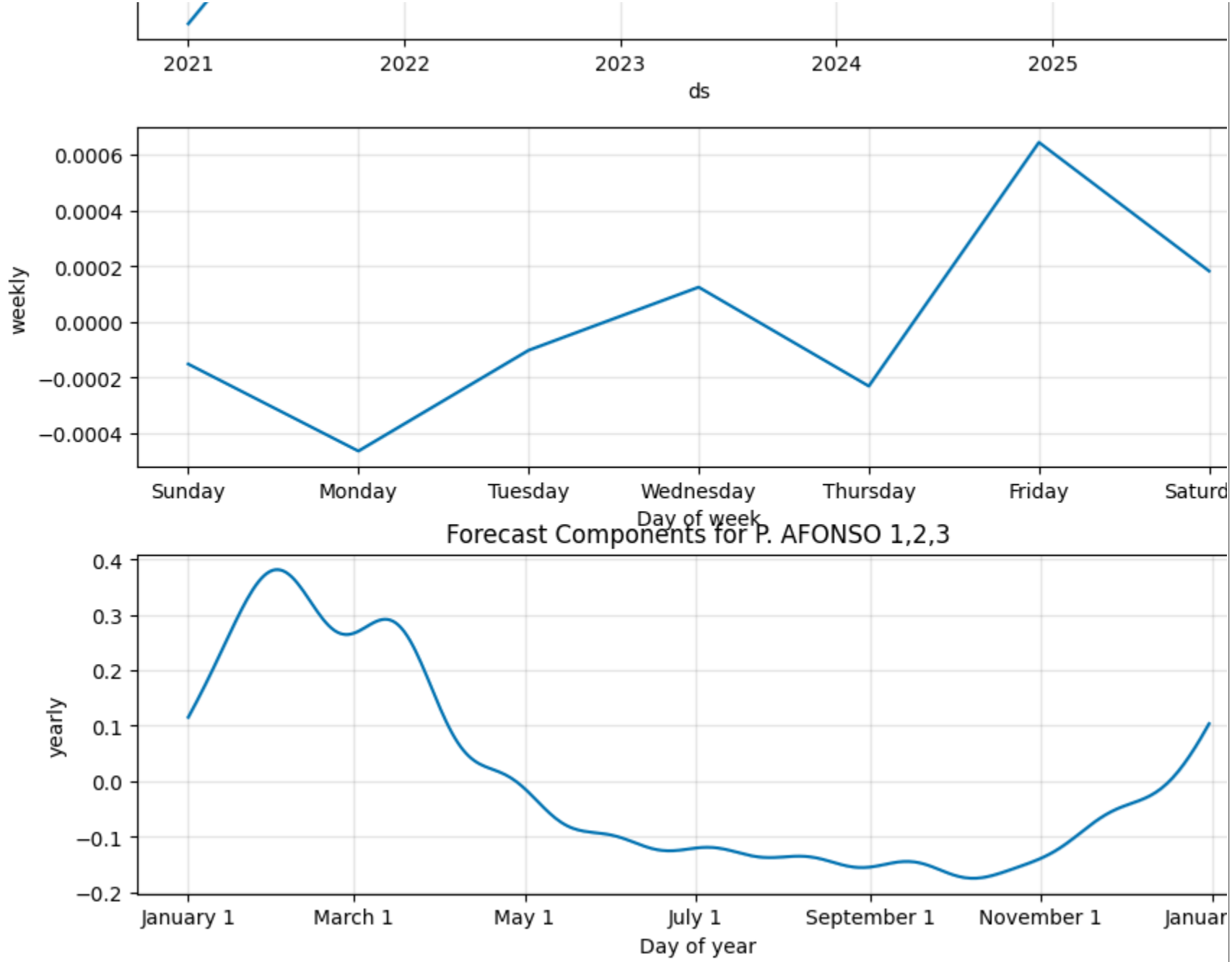
```
Reservatório: P. AFONSO 1,2,3
```




```
INFO:prophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to override this.
DEBUG:cmdstanpy:input tempfile: /tmp/tmp20qzy7c_/si09brvw.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmp20qzy7c_/ujvmi__x.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.12/dist-packages/prophet/stan_model/prophet_mo
10:47:10 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
10:47:11 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
### Métricas de Avaliação do Modelo ###
Erro Médio Absoluto (MAE): 658.13 MWméd
Raiz do Erro Quadrático Médio (RMSE): 914.26 MWméd
Erro Percentual Médio Absoluto (MAPE): 99.97%

Contexto: O valor médio real de ENA no período foi de 658.32 MWméd.
Um erro (MAE) de 658.13 representa aproximadamente 99.97% do valor médio.
```

Forecast for P. AFONSO 1,2,3


Observed data points
Forecast
Uncertainty interval
Real
ENA (Normalized)
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
2021
2022
2023
2024
2025
Date
trend
0.4
0.3
0.2
0.1
0.0

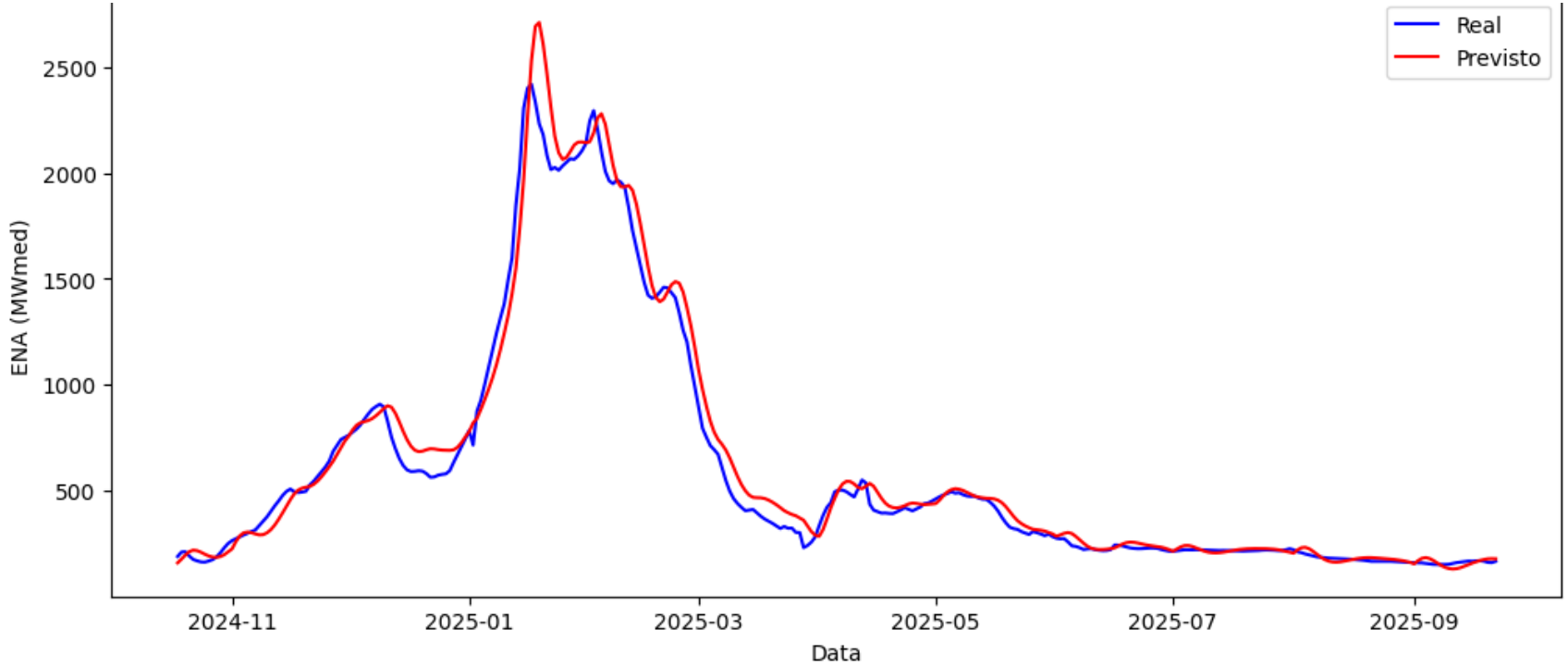



```
          ds    ena_di  ena_di_prev            rese
0        NaT       NaN     0.067210  P. AFONSO 1,2,3
1        NaT       NaN     0.077092  P. AFONSO 1,2,3
```

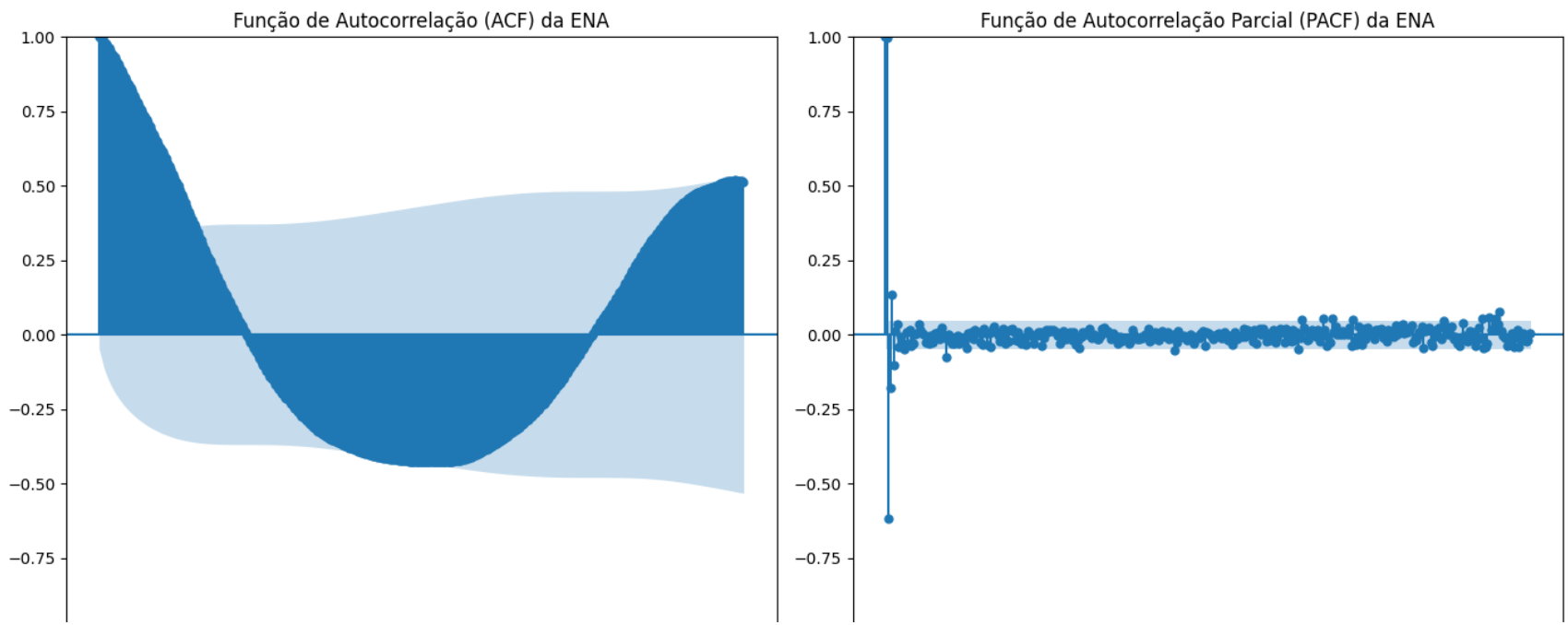
```
2             NaT       NaN     0.087255  P. AFONSO 1,2,3
3             NaT       NaN     0.097589  P. AFONSO 1,2,3
4             NaT       NaN     0.108742  P. AFONSO 1,2,3
...           ...       ...          ...              ...
261375 2025-09-18  0.016127          NaN  P. AFONSO 1,2,3
261530 2025-09-19  0.015383          NaN  P. AFONSO 1,2,3
261685 2025-09-20  0.013675          NaN  P. AFONSO 1,2,3
261840 2025-09-21  0.013365          NaN  P. AFONSO 1,2,3
261995 2025-09-22  0.015318          NaN  P. AFONSO 1,2,3

[1991 rows x 4 columns]
Shape X: (1696, 30, 4)
Shape y: (1696,)
Epoch 1/30
/usr/local/lib/python3.12/dist-packages/keras/src/layers/rnn/rnn.py:199: UserWarning: Do not pass an
  super().__init__(**kwargs)
43/43 ━━━━━━━━━━━━━━━━━━━━ 4s 34ms/step - loss: 0.0180 - val_loss: 0.0042
Epoch 2/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 42ms/step - loss: 0.0034 - val_loss: 0.0022
Epoch 3/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0024 - val_loss: 0.0017
Epoch 4/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0021 - val_loss: 0.0014
Epoch 5/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0017 - val_loss: 0.0013
Epoch 6/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0013 - val_loss: 0.0015
Epoch 7/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0013 - val_loss: 0.0012
Epoch 8/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0013 - val_loss: 0.0012
Epoch 9/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0013 - val_loss: 0.0013
Epoch 10/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0013 - val_loss: 0.0012
Epoch 11/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0012 - val_loss: 0.0011
Epoch 12/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - loss: 8.5350e-04 - val_loss: 0.0011
Epoch 13/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 25ms/step - loss: 9.5716e-04 - val_loss: 0.0011
```

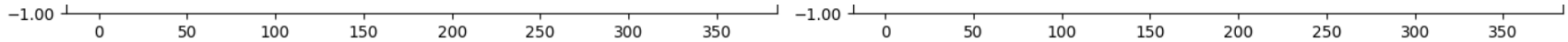
```
Epoch 14/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 9.8304e-04 - val_loss: 0.0013
Epoch 15/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 9.9595e-04 - val_loss: 9.7860e-04
Epoch 16/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 9.3383e-04 - val_loss: 0.0013
Epoch 17/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 8.5825e-04 - val_loss: 9.3787e-04
Epoch 18/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 8.0744e-04 - val_loss: 0.0011
Epoch 19/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 8.5765e-04 - val_loss: 0.0012
Epoch 20/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 7.3286e-04 - val_loss: 8.8899e-04
Epoch 21/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 36ms/step - loss: 5.9399e-04 - val_loss: 9.1801e-04
Epoch 22/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 26ms/step - loss: 6.6410e-04 - val_loss: 0.0012
Epoch 23/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 8.4370e-04 - val_loss: 9.2544e-04
Epoch 24/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 6.4982e-04 - val_loss: 7.8751e-04
Epoch 25/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - loss: 6.0404e-04 - val_loss: 9.8150e-04
Epoch 26/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 5.6511e-04 - val_loss: 7.7272e-04
Epoch 27/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 4.8743e-04 - val_loss: 6.4681e-04
Epoch 28/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 5.4907e-04 - val_loss: 7.3341e-04
Epoch 29/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 5.4523e-04 - val_loss: 7.8083e-04
Epoch 30/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 39ms/step - loss: 4.6586e-04 - val_loss: 7.5423e-04
11/11 ━━━━━━━━━━━━━━━━━━━━ 1s 53ms/step
MAE: 54.00
MSE: 7693.00
RMSE: 87.71
R²: 0.98
MAPE: 9.66%
```

Previsão LSTM - ENA diária de P. AFONSO 1,2,3

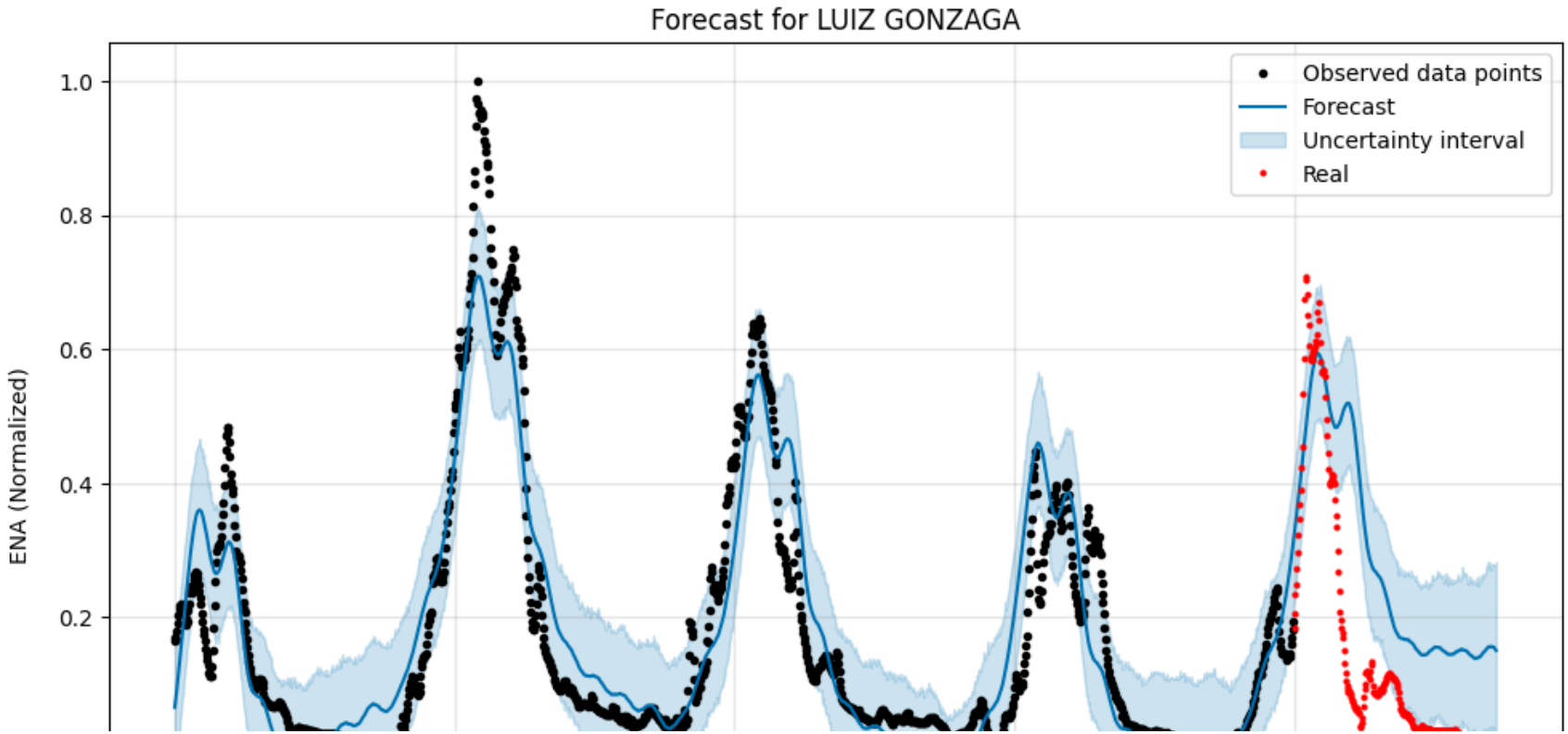


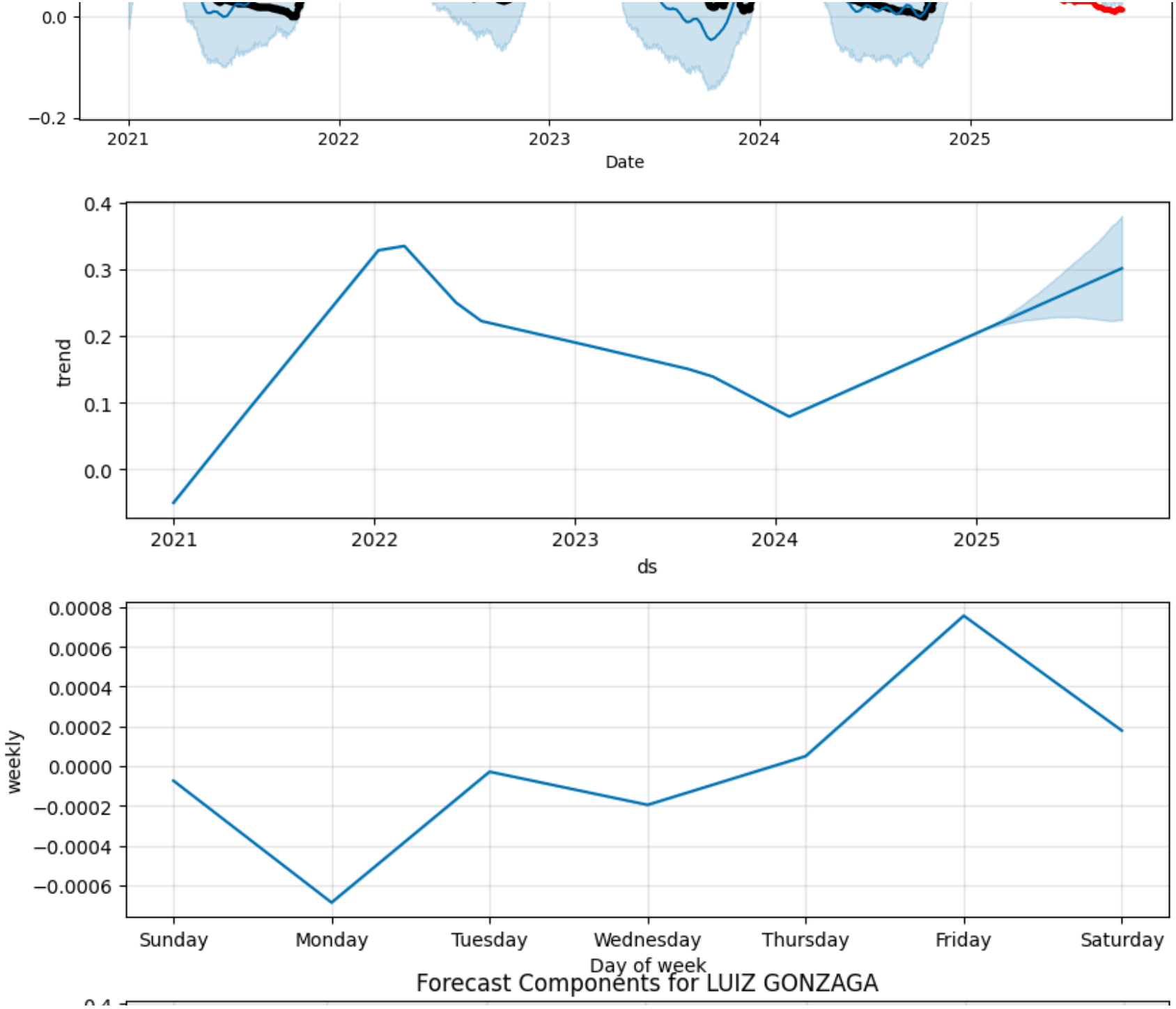

```
Reservatório: LUIZ GONZAGA
```

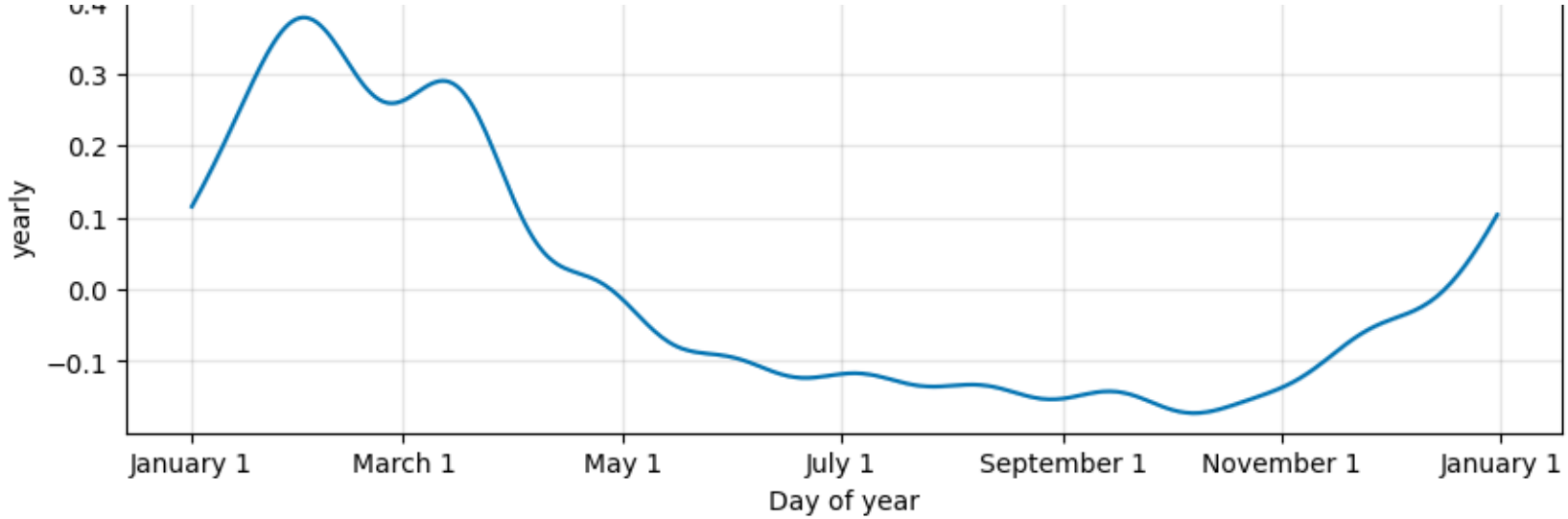





```
INFO:prophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to override this.
DEBUG:cmdstanpy:input tempfile: /tmp/tmp20qzy7c_/vkyu3bft.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmp20qzy7c_/o7iz84ts.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.12/dist-packages/prophet/stan_model/prophet_mo
10:47:57 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
10:47:57 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
### Métricas de Avaliação do Modelo ###
Erro Médio Absoluto (MAE): 766.35 MWméd
Raiz do Erro Quadrático Médio (RMSE): 1064.74 MWméd
Erro Percentual Médio Absoluto (MAPE): 99.98%

Contexto: O valor médio real de ENA no período foi de 766.53 MWméd.
Um erro (MAE) de 766.35 representa aproximadamente 99.98% do valor médio.
```



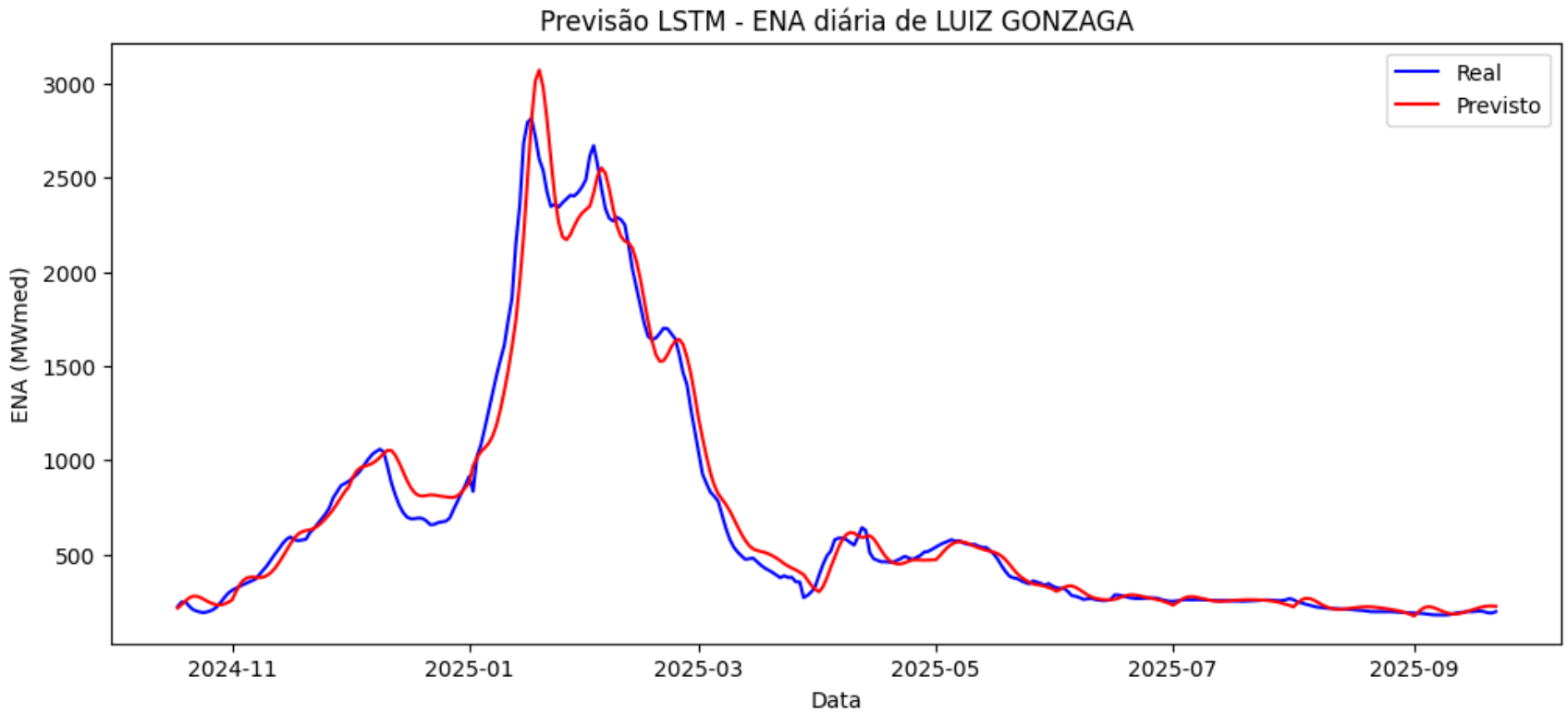

0.0
−0.2
2021
2022
2023
2024
2025
Date
0.4
0.3
0.2
0.1
0.0
trend
2021
2022
2023
2024
2025
ds
0.0008
0.0006
0.0004
0.0002
0.0000
−0.0002
−0.0004
−0.0006
weekly
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Day of week
Forecast Components for LUIZ GONZAGA




```
                ds    ena_di  ena_di_prev          rese
0              NaT       NaN     0.065570  LUIZ GONZAGA
1              NaT       NaN     0.075106  LUIZ GONZAGA
2              NaT       NaN     0.085138  LUIZ GONZAGA
3              NaT       NaN     0.094980  LUIZ GONZAGA
4              NaT       NaN     0.106265  LUIZ GONZAGA
...            ...       ...          ...           ...
261362 2025-09-18  0.015870          NaN  LUIZ GONZAGA
261517 2025-09-19  0.015138          NaN  LUIZ GONZAGA
261672 2025-09-20  0.013457          NaN  LUIZ GONZAGA
261827 2025-09-21  0.013152          NaN  LUIZ GONZAGA
261982 2025-09-22  0.015074          NaN  LUIZ GONZAGA

[1991 rows x 4 columns]
Shape X: (1696, 30, 4)
Shape y: (1696,)
Epoch 1/30
/usr/local/lib/python3.12/dist-packages/keras/src/layers/rnn/rnn.py:199: UserWarning: Do not pass an
  super().__init__(**kwargs)
43/43 ━━━━━━━━━━━━━━━━━━━━ 4s 35ms/step - loss: 0.0461 - val_loss: 0.0048
Epoch 2/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0034 - val_loss: 0.0030
Epoch 3/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 38ms/step - loss: 0.0027 - val_loss: 0.0024
Epoch 4/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 40ms/step - loss: 0.0021 - val_loss: 0.0020
```

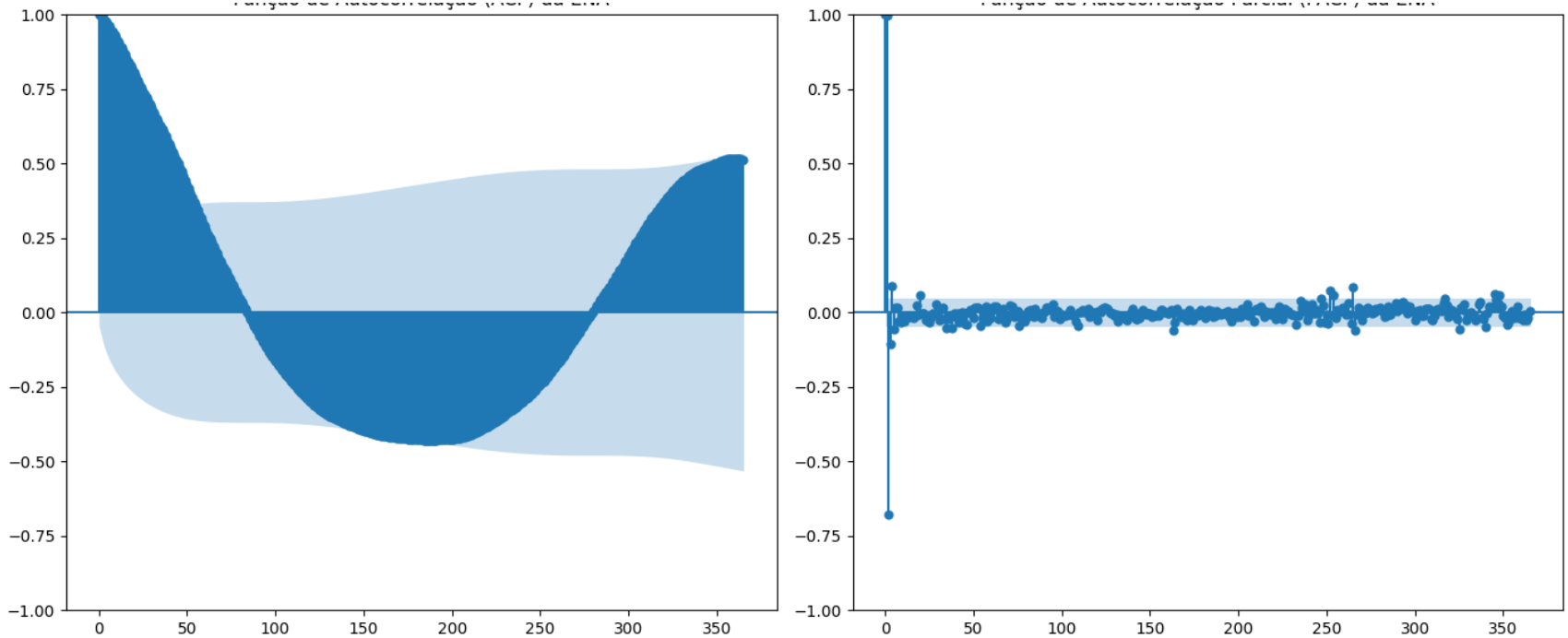
```
Epoch 5/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - loss: 0.0018 - val_loss: 0.0017
Epoch 6/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - loss: 0.0016 - val_loss: 0.0018
Epoch 7/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - loss: 0.0018 - val_loss: 0.0017
Epoch 8/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - loss: 0.0015 - val_loss: 0.0017
Epoch 9/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0013 - val_loss: 0.0015
Epoch 10/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0012 - val_loss: 0.0015
Epoch 11/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - loss: 0.0014 - val_loss: 0.0014
Epoch 12/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 9.6050e-04 - val_loss: 0.0014
Epoch 13/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 38ms/step - loss: 9.1838e-04 - val_loss: 0.0014
Epoch 14/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 26ms/step - loss: 0.0011 - val_loss: 0.0014
Epoch 15/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - loss: 0.0011 - val_loss: 0.0013
Epoch 16/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - loss: 9.8459e-04 - val_loss: 0.0014
Epoch 17/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0011 - val_loss: 0.0012
Epoch 18/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 8.7913e-04 - val_loss: 0.0012
Epoch 19/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0010 - val_loss: 0.0014
Epoch 20/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - loss: 8.7796e-04 - val_loss: 0.0011
Epoch 21/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - loss: 7.9823e-04 - val_loss: 0.0010
Epoch 22/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 37ms/step - loss: 9.0783e-04 - val_loss: 0.0010
Epoch 23/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 26ms/step - loss: 8.1524e-04 - val_loss: 9.7034e-04
Epoch 24/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 6.7399e-04 - val_loss: 9.4340e-04
Epoch 25/30
```

```
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 7.4547e-04 - val_loss: 9.9821e-04
Epoch 26/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - loss: 6.6809e-04 - val_loss: 9.6068e-04
Epoch 27/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - loss: 8.2543e-04 - val_loss: 8.7701e-04
Epoch 28/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 6.6731e-04 - val_loss: 7.9550e-04
Epoch 29/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - loss: 7.3144e-04 - val_loss: 8.7729e-04
Epoch 30/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 6.1042e-04 - val_loss: 7.3612e-04
11/11 ━━━━━━━━━━━━━━━━━━━━ 1s 44ms/step
MAE: 62.75
MSE: 10513.95
RMSE: 102.54
R²: 0.98
MAPE: 9.16%
```

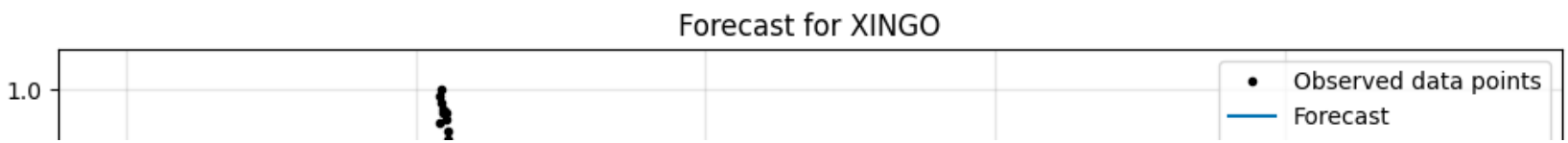


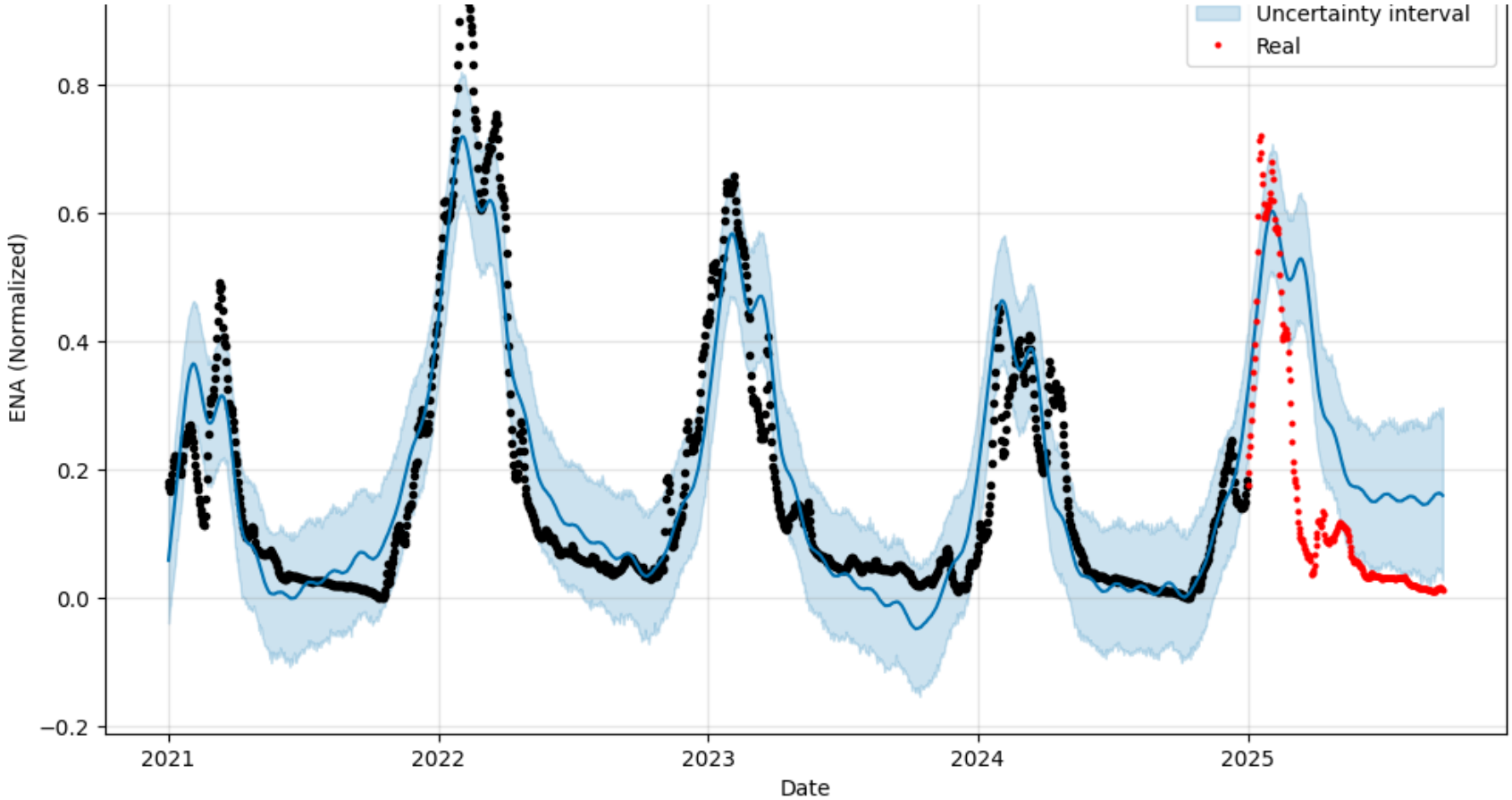

```
Reservatório: XINGO
```

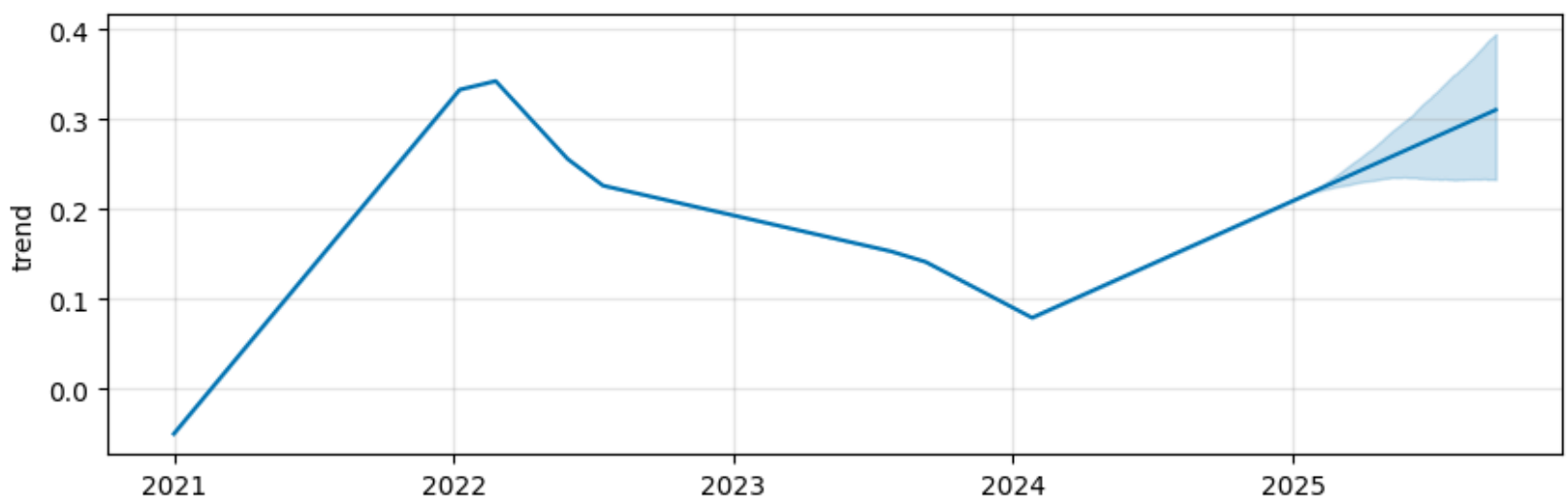
Função de Autocorrelação (ACF) da ENA Função de Autocorrelação Parcial (PACF) da ENA

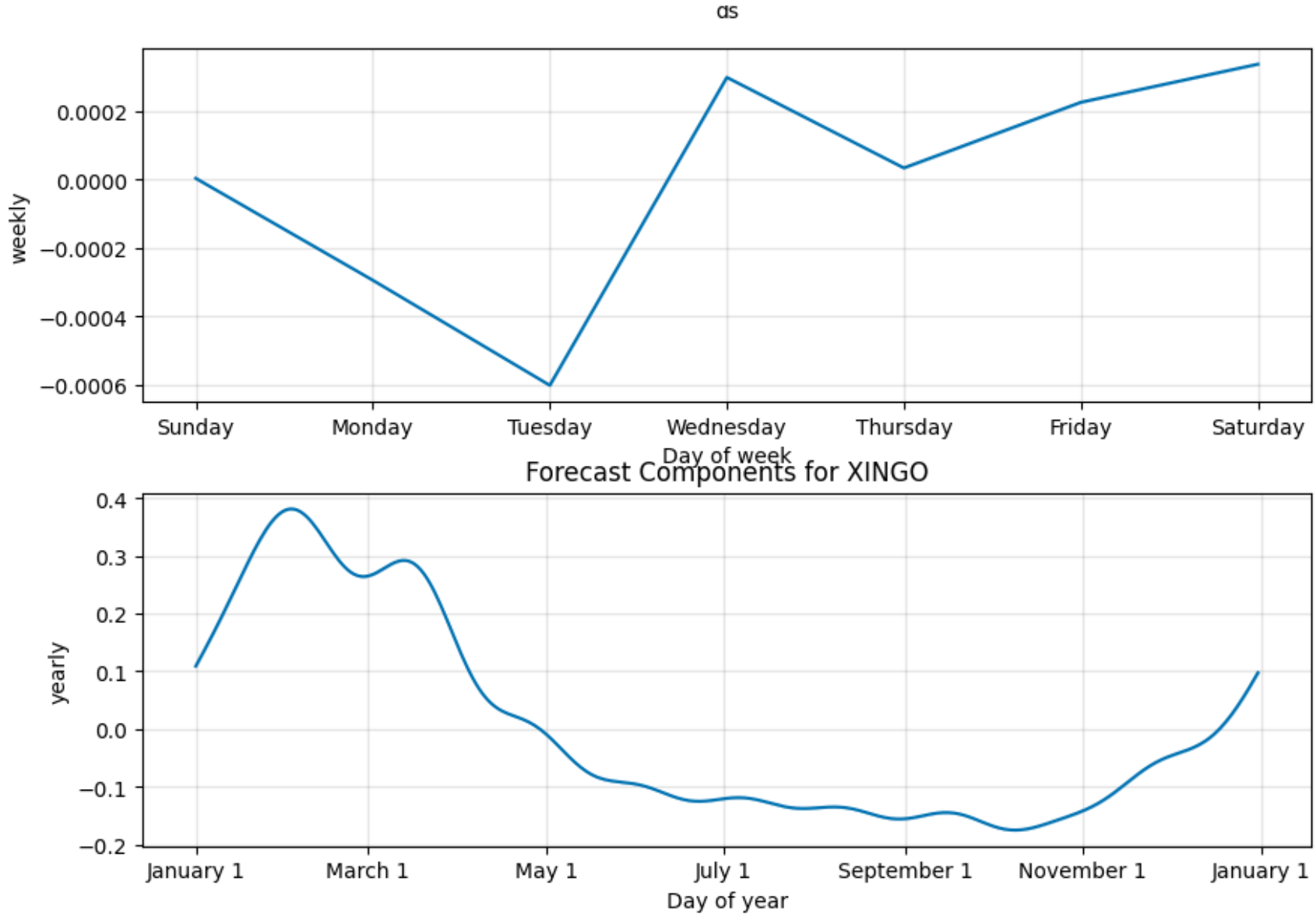



```
INFO:prophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to override this.
DEBUG:cmdstanpy:input tempfile: /tmp/tmp20qzy7c_/sz0vn37o.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmp20qzy7c_/5xroyv1x.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.12/dist-packages/prophet/stan_model/prophet_mo
10:48:45 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
10:48:45 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
### Métricas de Avaliação do Modelo ###
Erro Médio Absoluto (MAE): 1868.09 MWméd
Raiz do Erro Quadrático Médio (RMSE): 2594.35 MWméd
Erro Percentual Médio Absoluto (MAPE): 99.99%

Contexto: O valor médio real de ENA no período foi de 1868.28 MWméd.
Um erro (MAE) de 1868.09 representa aproximadamente 99.99% do valor médio.
```




Uncertainty interval
Real
ENA (Normalized)
0.8
0.6
0.4
0.2
0.0
−0.2
2021
2022
2023
2024
2025
Date

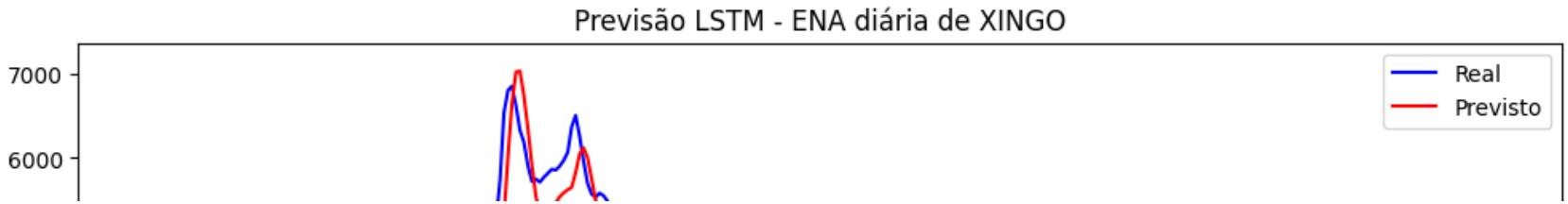

trend
0.4
0.3
0.2
0.1
0.0
2021
2022
2023
2024
2025

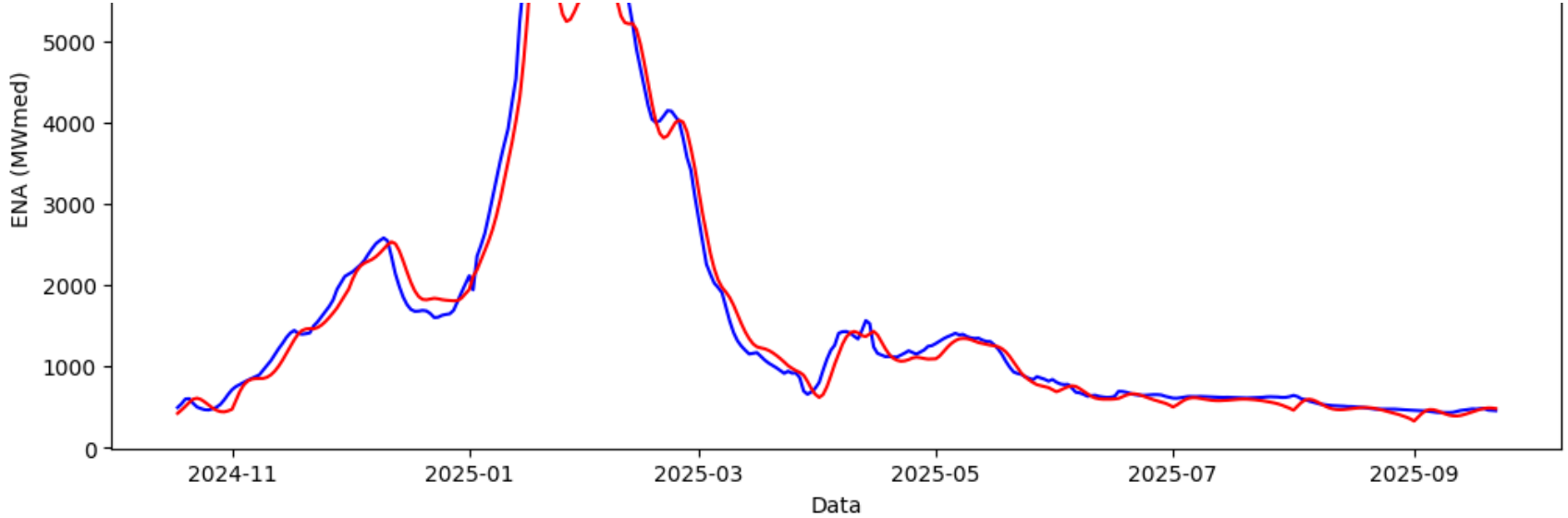



```
                ds    ena_di  ena_di_prev   rese
0              NaT       NaN     0.058699  XINGO
1              NaT       NaN     0.069019  XINGO
2              NaT       NaN     0.079022  XINGO
3              NaT       NaN     0.089184  XINGO
4              NaT       NaN     0.099456  XINGO
...            ...       ...          ...    ...
261434 2025-09-18  0.015884          NaN  XINGO
```

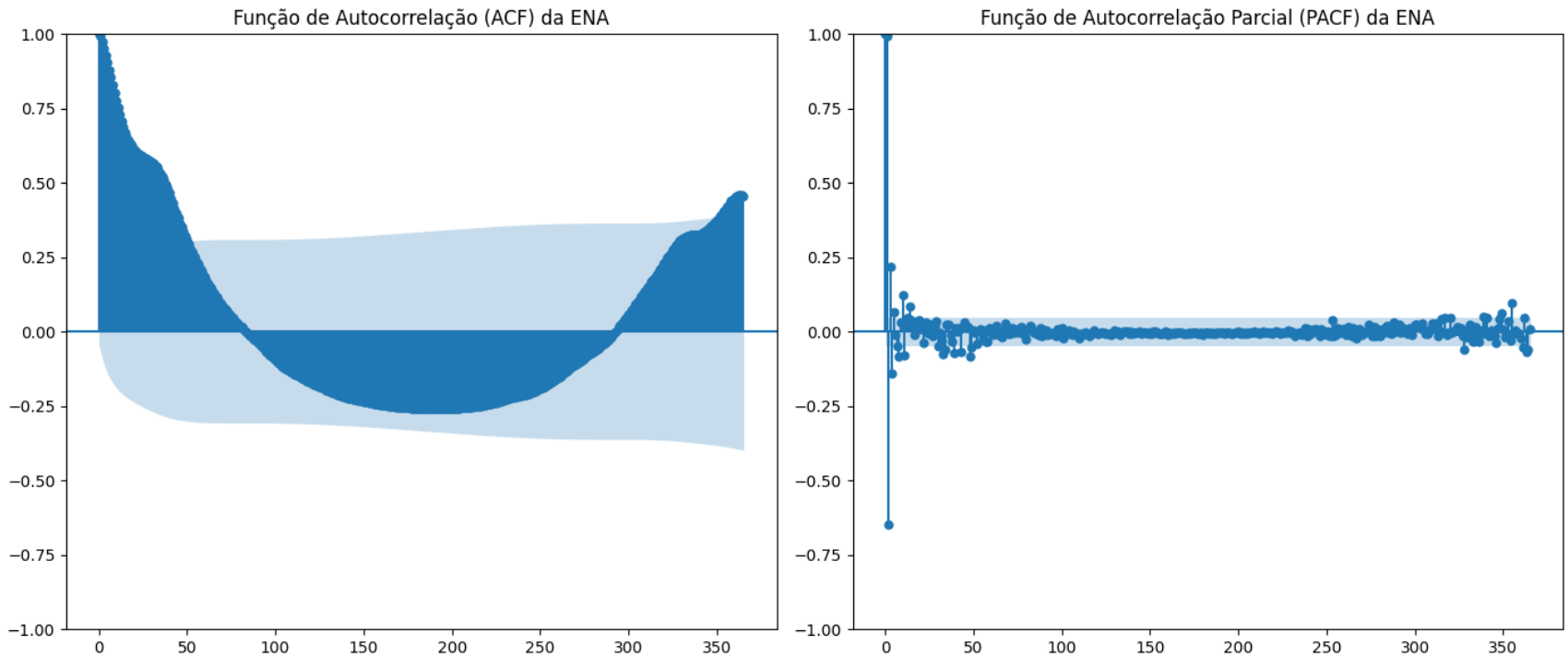
```
261454 2025-09-18  0.015884          NaN  XINGO
261589 2025-09-19  0.016127          NaN  XINGO
261744 2025-09-20  0.013956          NaN  XINGO
261899 2025-09-21  0.013675          NaN  XINGO
262054 2025-09-22  0.013246          NaN  XINGO

[1991 rows x 4 columns]
Shape X: (1696, 30, 4)
Shape y: (1696,)
Epoch 1/30
/usr/local/lib/python3.12/dist-packages/keras/src/layers/rnn/rnn.py:199: UserWarning: Do not pass an
  super().__init__(**kwargs)
43/43 ━━━━━━━━━━━━━━━━━━━━ 4s 37ms/step - loss: 0.0299 - val_loss: 0.0040
Epoch 2/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - loss: 0.0029 - val_loss: 0.0023
Epoch 3/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - loss: 0.0025 - val_loss: 0.0017
Epoch 4/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 3s 40ms/step - loss: 0.0021 - val_loss: 0.0014
Epoch 5/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 36ms/step - loss: 0.0015 - val_loss: 0.0016
Epoch 6/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - loss: 0.0015 - val_loss: 0.0013
Epoch 7/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 3s 30ms/step - loss: 0.0014 - val_loss: 0.0013
Epoch 8/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 30ms/step - loss: 0.0012 - val_loss: 0.0013
Epoch 9/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - loss: 0.0013 - val_loss: 0.0012
Epoch 10/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - loss: 0.0011 - val_loss: 0.0011
Epoch 11/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 30ms/step - loss: 0.0012 - val_loss: 0.0012
Epoch 12/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 42ms/step - loss: 0.0011 - val_loss: 0.0010
Epoch 13/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 42ms/step - loss: 0.0011 - val_loss: 0.0015
Epoch 14/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - loss: 0.0012 - val_loss: 9.9483e-04
Epoch 15/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - loss: 0.0011 - val_loss: 0.0013
Epoch 16/30
```

```
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - loss: 0.0010 - val_loss: 9.2632e-04
Epoch 17/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 3s 29ms/step - loss: 8.9538e-04 - val_loss: 9.0073e-04
Epoch 18/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - loss: 8.3747e-04 - val_loss: 9.2632e-04
Epoch 19/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 3s 38ms/step - loss: 9.7401e-04 - val_loss: 9.6886e-04
Epoch 20/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 29ms/step - loss: 7.8425e-04 - val_loss: 7.5299e-04
Epoch 21/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - loss: 8.1434e-04 - val_loss: 7.2055e-04
Epoch 22/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - loss: 7.4397e-04 - val_loss: 6.9521e-04
Epoch 23/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 30ms/step - loss: 6.7774e-04 - val_loss: 6.6945e-04
Epoch 24/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - loss: 7.3401e-04 - val_loss: 7.2217e-04
Epoch 25/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - loss: 6.4636e-04 - val_loss: 6.4681e-04
Epoch 26/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 3s 32ms/step - loss: 6.2285e-04 - val_loss: 6.6373e-04
Epoch 27/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 41ms/step - loss: 5.8530e-04 - val_loss: 6.1335e-04
Epoch 28/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 40ms/step - loss: 5.8607e-04 - val_loss: 5.6361e-04
Epoch 29/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 30ms/step - loss: 6.0523e-04 - val_loss: 5.6563e-04
Epoch 30/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 30ms/step - loss: 6.3394e-04 - val_loss: 5.6174e-04
11/11 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step
MAE: 141.07
MSE: 46118.51
RMSE: 214.75
R²: 0.98
MAPE: 8.93%
```



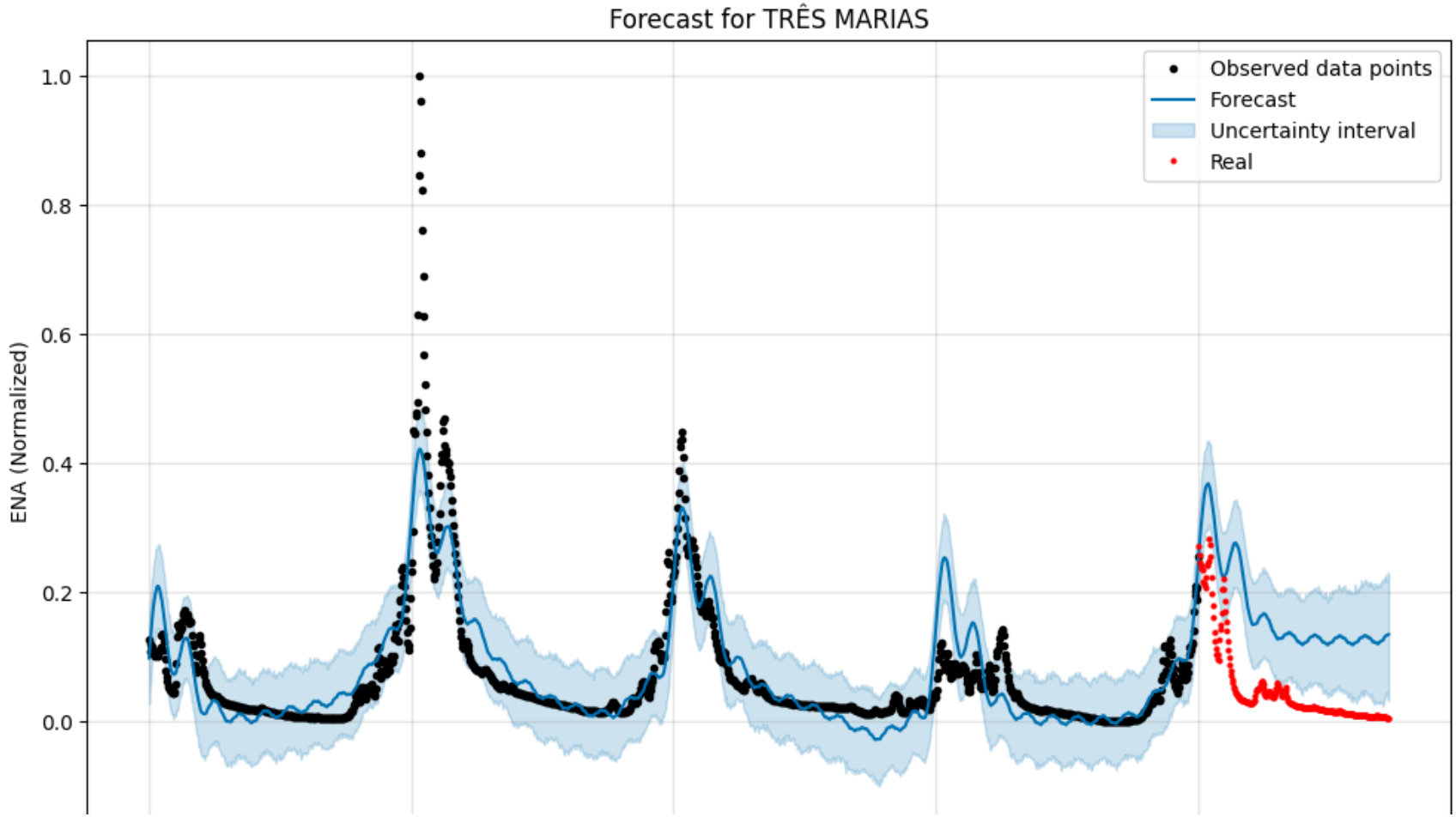


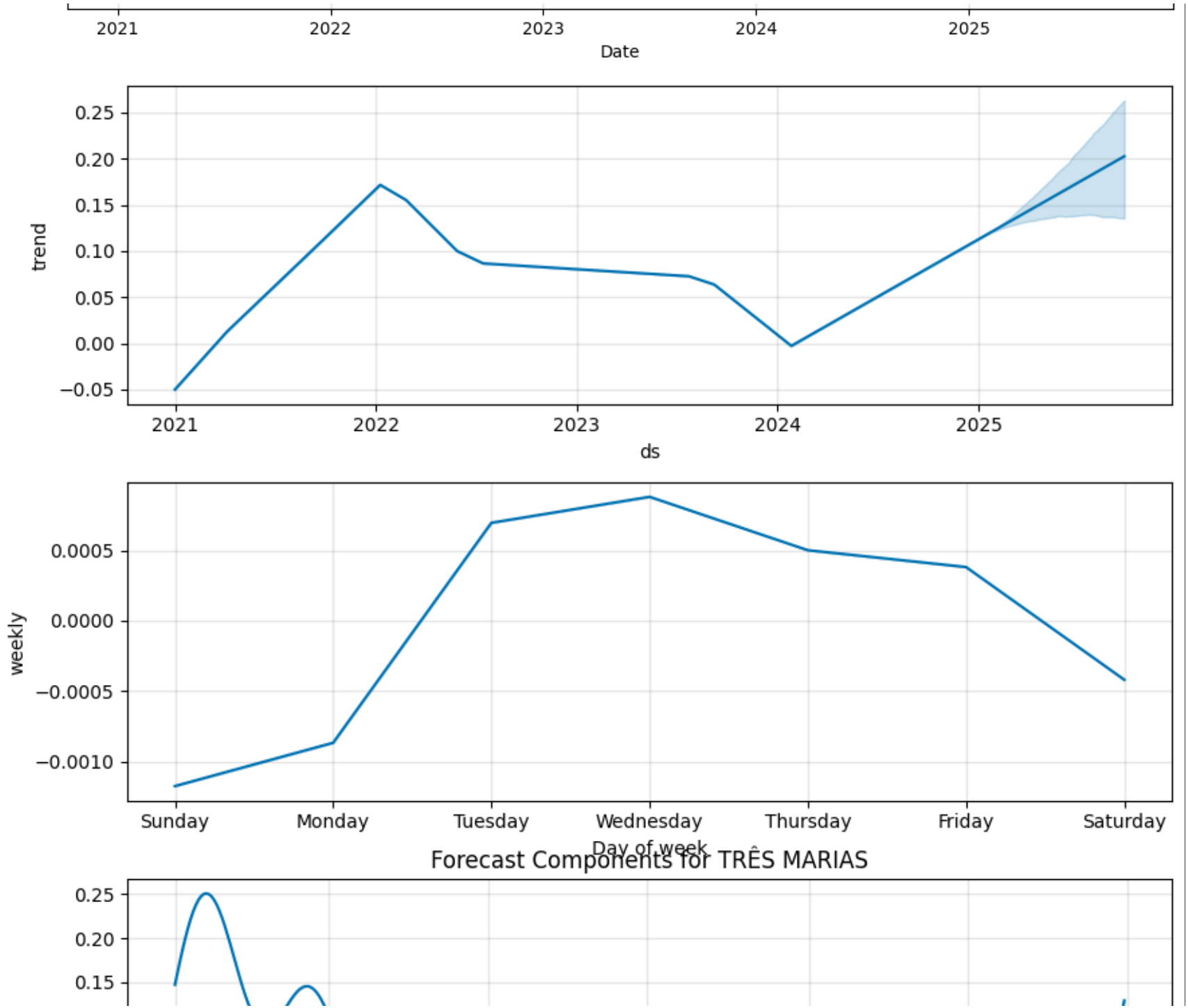

```
Reservatório: TRÊS MARIAS
```

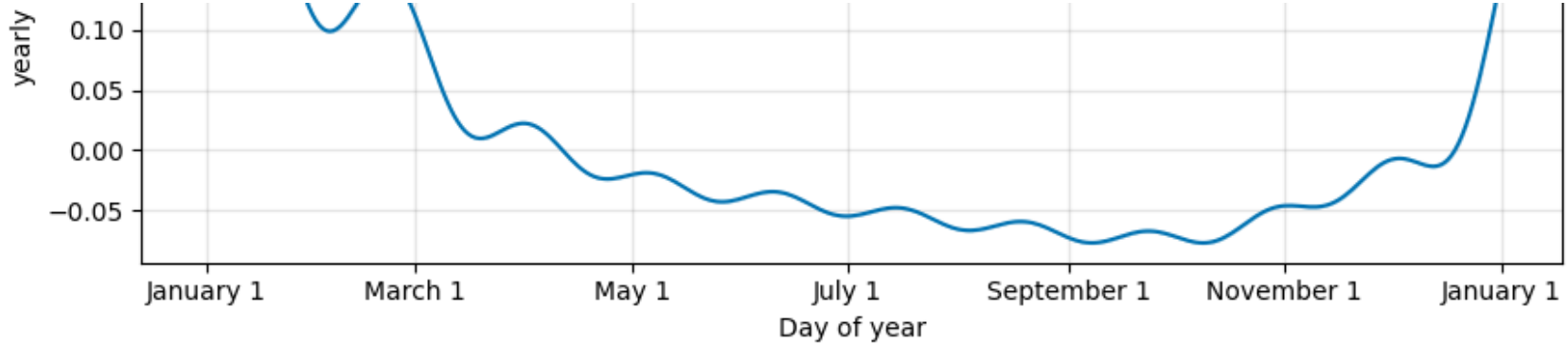



```
INFO:prophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to override this.
DEBUG:cmdstanpy:input tempfile: /tmp/tmp20qzy7c_/jyp3icmg.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmp20qzy7c_/kgqkttqs.json
```

```
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.12/dist-packages/prophet/stan_model/prophet_mo
10:49:42 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
10:49:42 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
### Métricas de Avaliação do Modelo ###
Erro Médio Absoluto (MAE): 235.02 MWméd
Raiz do Erro Quadrático Médio (RMSE): 396.57 MWméd
Erro Percentual Médio Absoluto (MAPE): 99.95%

Contexto: O valor médio real de ENA no período foi de 235.11 MWméd.
Um erro (MAE) de 235.02 representa aproximadamente 99.96% do valor médio.
```



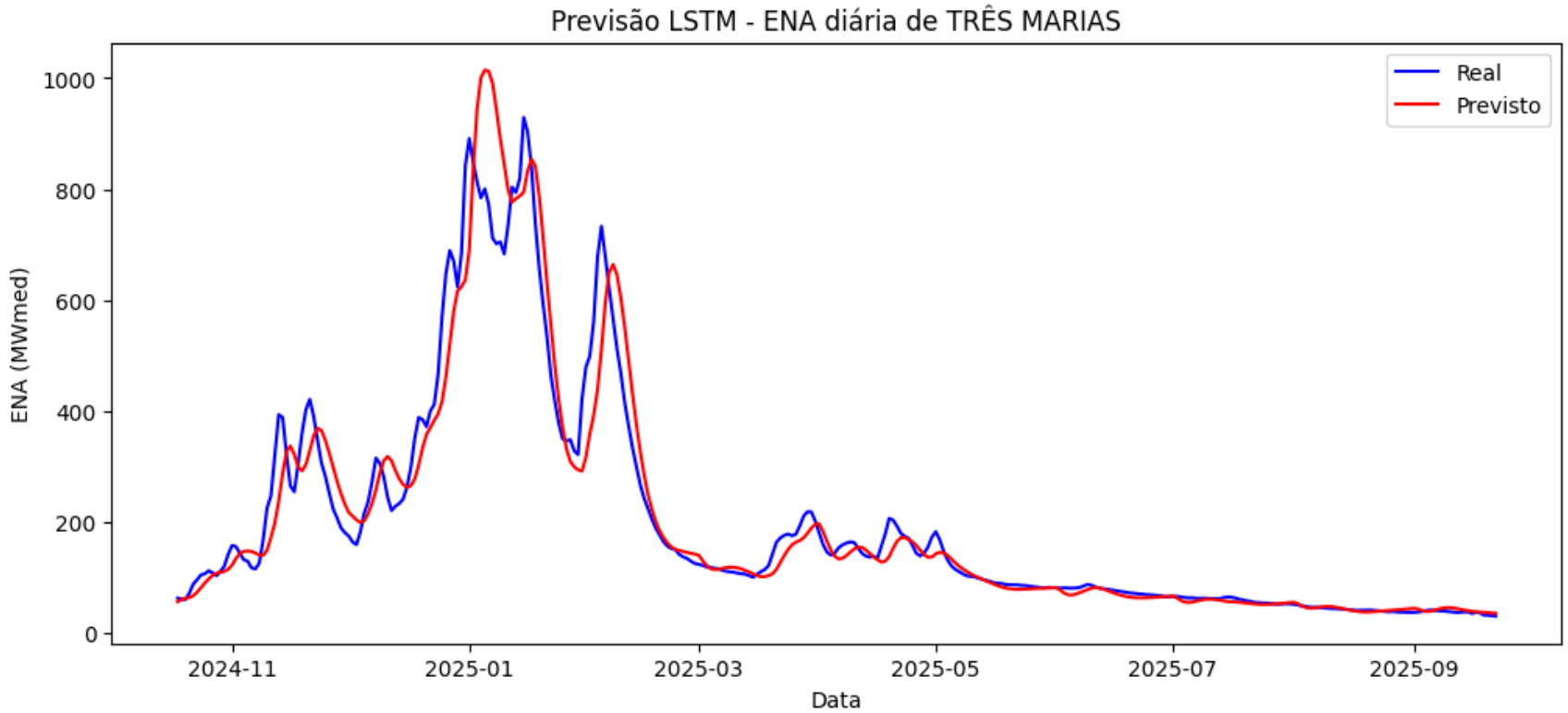

2021
2022
2023
2024
2025
Date
trend
0.25
0.20
0.15
0.10
0.05
0.00
−0.05
ds
weekly
0.0005
0.0000
−0.0005
−0.0010
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Day of week
Forecast Components for TRÊS MARIAS



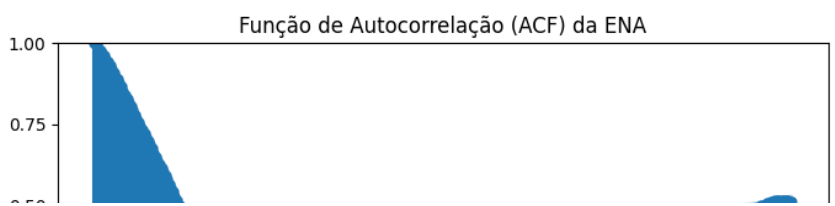

```
                ds    ena_di  ena_di_prev         rese
0             NaT       NaN     0.097658  TRÊS MARIAS
1             NaT       NaN     0.111657  TRÊS MARIAS
2             NaT       NaN     0.125259  TRÊS MARIAS
3             NaT       NaN     0.139284  TRÊS MARIAS
4             NaT       NaN     0.153731  TRÊS MARIAS
...           ...       ...          ...          ...
261431 2025-09-18  0.006631          NaN  TRÊS MARIAS
261586 2025-09-19  0.005432          NaN  TRÊS MARIAS
261741 2025-09-20  0.005469          NaN  TRÊS MARIAS
261896 2025-09-21  0.005121          NaN  TRÊS MARIAS
262051 2025-09-22  0.004948          NaN  TRÊS MARIAS

[1991 rows x 4 columns]
Shape X: (1696, 30, 4)
Shape y: (1696,)
Epoch 1/30
/usr/local/lib/python3.12/dist-packages/keras/src/layers/rnn/rnn.py:199: UserWarning: Do not pass an
  super().__init__(**kwargs)
43/43 ━━━━━━━━━━━━━━━━━━━━ 5s 54ms/step - loss: 0.0100 - val_loss: 0.0010
Epoch 2/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 32ms/step - loss: 0.0018 - val_loss: 6.5581e-04
Epoch 3/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - loss: 0.0011 - val_loss: 9.1154e-04
Epoch 4/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - loss: 0.0011 - val_loss: 7.5067e-04
Epoch 5/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 3s 31ms/step - loss: 6.0668e-04 - val_loss: 7.7632e-04
Epoch 6/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - loss: 8.9214e-04 - val_loss: 6.5774e-04
Epoch 7/30
```

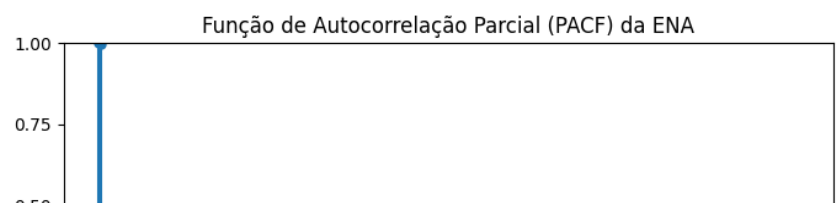
```
Epoch 7/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - loss: 7.2687e-04 - val_loss: 5.8017e-04
Epoch 8/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 3s 37ms/step - loss: 8.2532e-04 - val_loss: 6.2907e-04
Epoch 9/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - loss: 7.7429e-04 - val_loss: 8.0479e-04
Epoch 10/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 31ms/step - loss: 6.5980e-04 - val_loss: 5.6352e-04
Epoch 11/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 3s 31ms/step - loss: 8.4963e-04 - val_loss: 5.6824e-04
Epoch 12/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 3s 31ms/step - loss: 5.2747e-04 - val_loss: 5.5825e-04
Epoch 13/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 40ms/step - loss: 5.8631e-04 - val_loss: 7.3196e-04
Epoch 14/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 45ms/step - loss: 6.1579e-04 - val_loss: 5.7678e-04
Epoch 15/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - loss: 4.1879e-04 - val_loss: 6.7360e-04
Epoch 16/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - loss: 6.1581e-04 - val_loss: 4.4121e-04
Epoch 17/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 3s 32ms/step - loss: 4.7392e-04 - val_loss: 4.7657e-04
Epoch 18/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - loss: 8.5053e-04 - val_loss: 4.0533e-04
Epoch 19/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - loss: 5.0216e-04 - val_loss: 6.7536e-04
Epoch 20/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - loss: 5.1654e-04 - val_loss: 4.3004e-04
Epoch 21/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 39ms/step - loss: 6.7289e-04 - val_loss: 4.4832e-04
Epoch 22/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 34ms/step - loss: 4.8673e-04 - val_loss: 5.0393e-04
Epoch 23/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - loss: 4.7723e-04 - val_loss: 4.1237e-04
Epoch 24/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 3s 31ms/step - loss: 3.7070e-04 - val_loss: 3.3291e-04
Epoch 25/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - loss: 4.1476e-04 - val_loss: 3.6299e-04
Epoch 26/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - loss: 3.4399e-04 - val_loss: 4.0218e-04
Epoch 27/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 3.2368e-04 - val_loss: 5.3975e-04
```

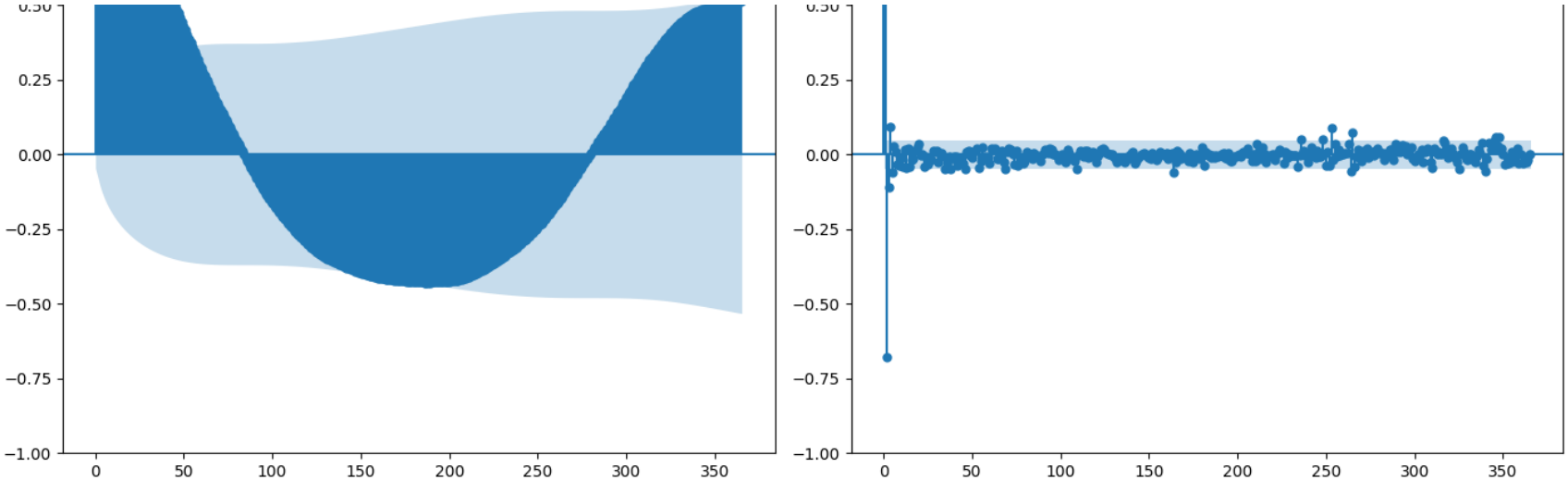
```
Epoch 28/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 40ms/step - loss: 4.3039e-04 - val_loss: 5.3966e-04
Epoch 29/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 43ms/step - loss: 4.9189e-04 - val_loss: 2.8565e-04
Epoch 30/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 34ms/step - loss: 3.5772e-04 - val_loss: 3.2833e-04
11/11 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step
MAE: 31.16
MSE: 3453.30
RMSE: 58.76
R²: 0.92
MAPE: 11.54%
```



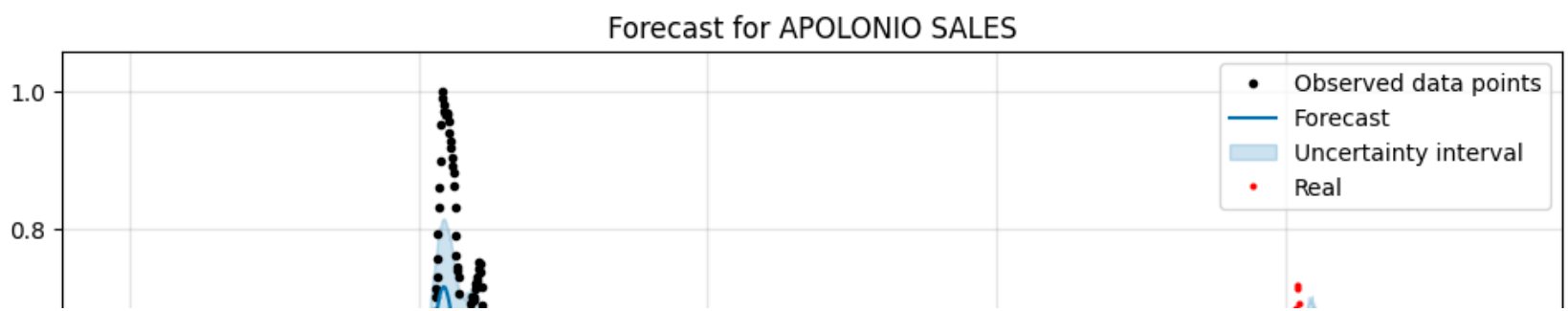

```
Reservatório: APOLONIO SALES
```

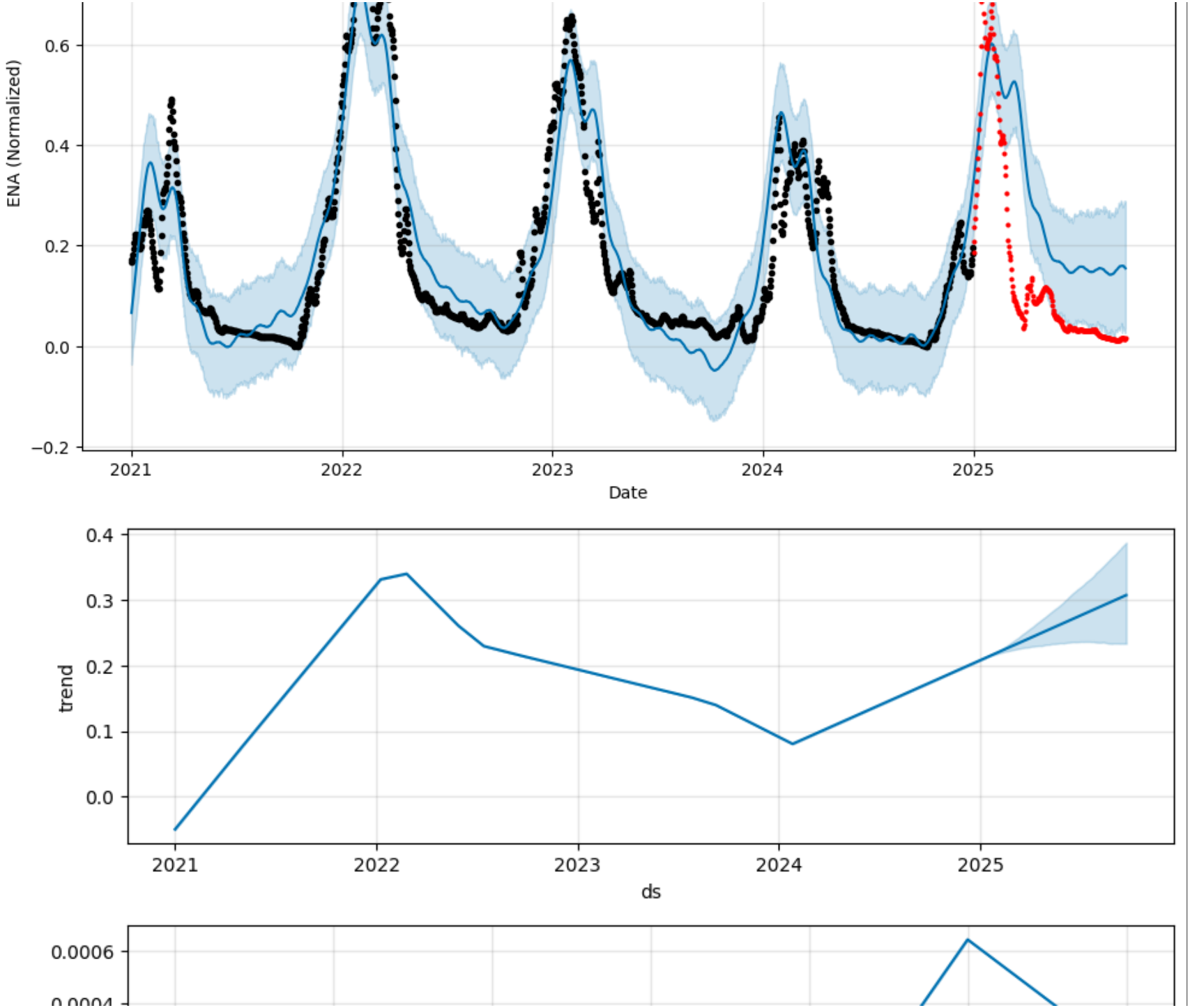


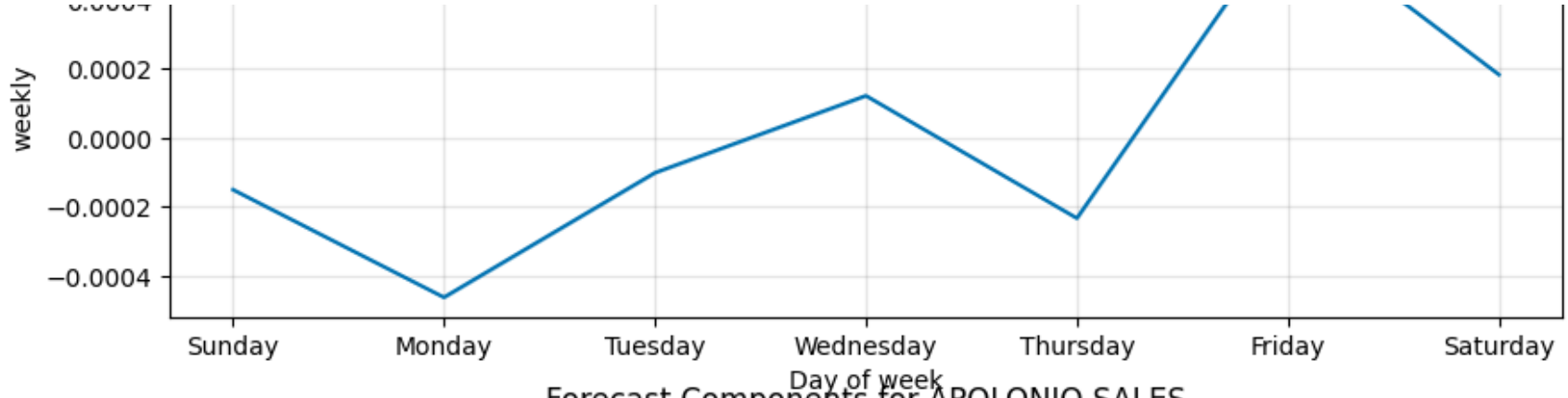


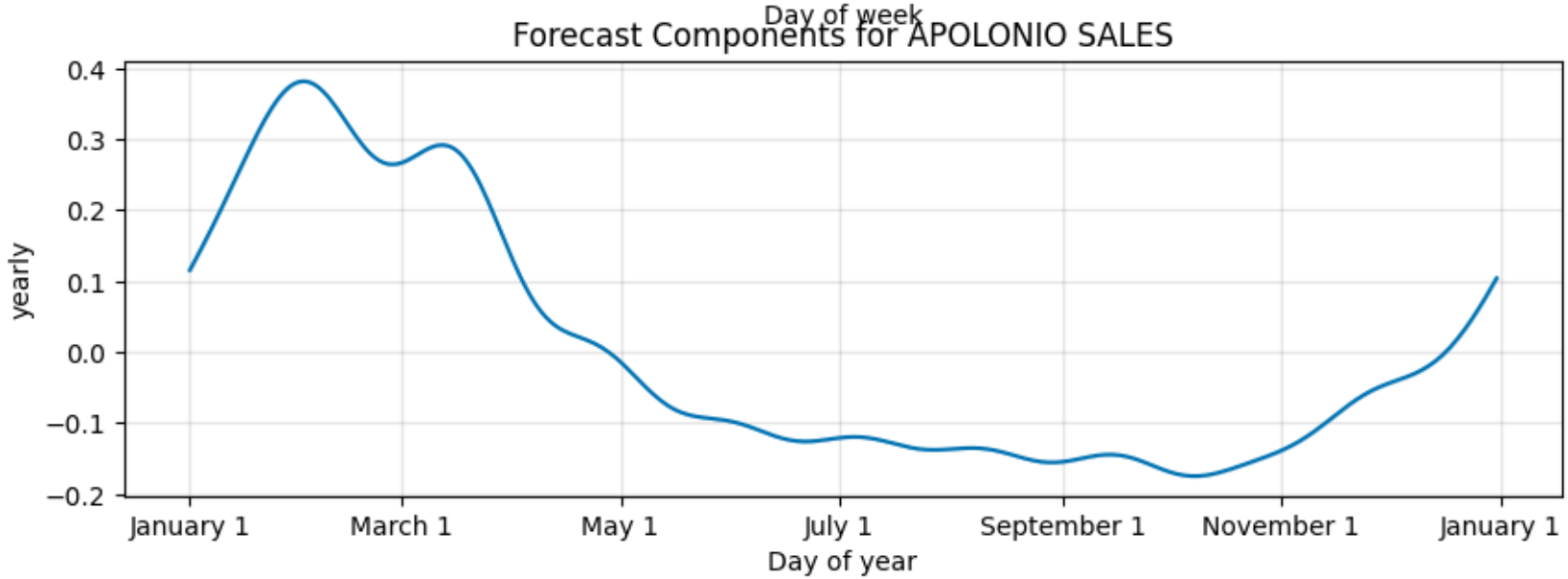



```
INFO:prophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to override this.
DEBUG:cmdstanpy:input tempfile: /tmp/tmp20qzy7c_/fz2cfp31.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmp20qzy7c_/5jf2akxs.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.12/dist-packages/prophet/stan_model/prophet_mo
10:50:43 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
10:50:44 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
### Métricas de Avaliação do Modelo ###
Erro Médio Absoluto (MAE): 168.55 MWméd
Raiz do Erro Quadrático Médio (RMSE): 234.16 MWméd
Erro Percentual Médio Absoluto (MAPE): 99.89%

Contexto: O valor médio real de ENA no período foi de 168.74 MWméd.
Um erro (MAE) de 168.55 representa aproximadamente 99.89% do valor médio.
```




ENA (Normalized)
0.6
0.4
0.2
0.0
−0.2
2021
2022
2023
2024
2025
Date
trend
0.4
0.3
0.2
0.1
0.0
2021
2022
2023
2024
2025
ds
0.0006
0.0004

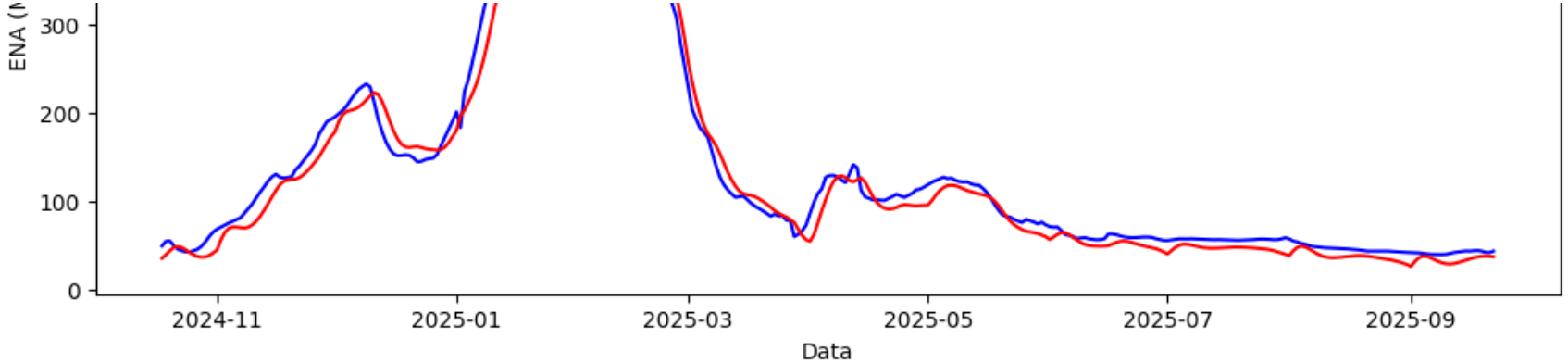




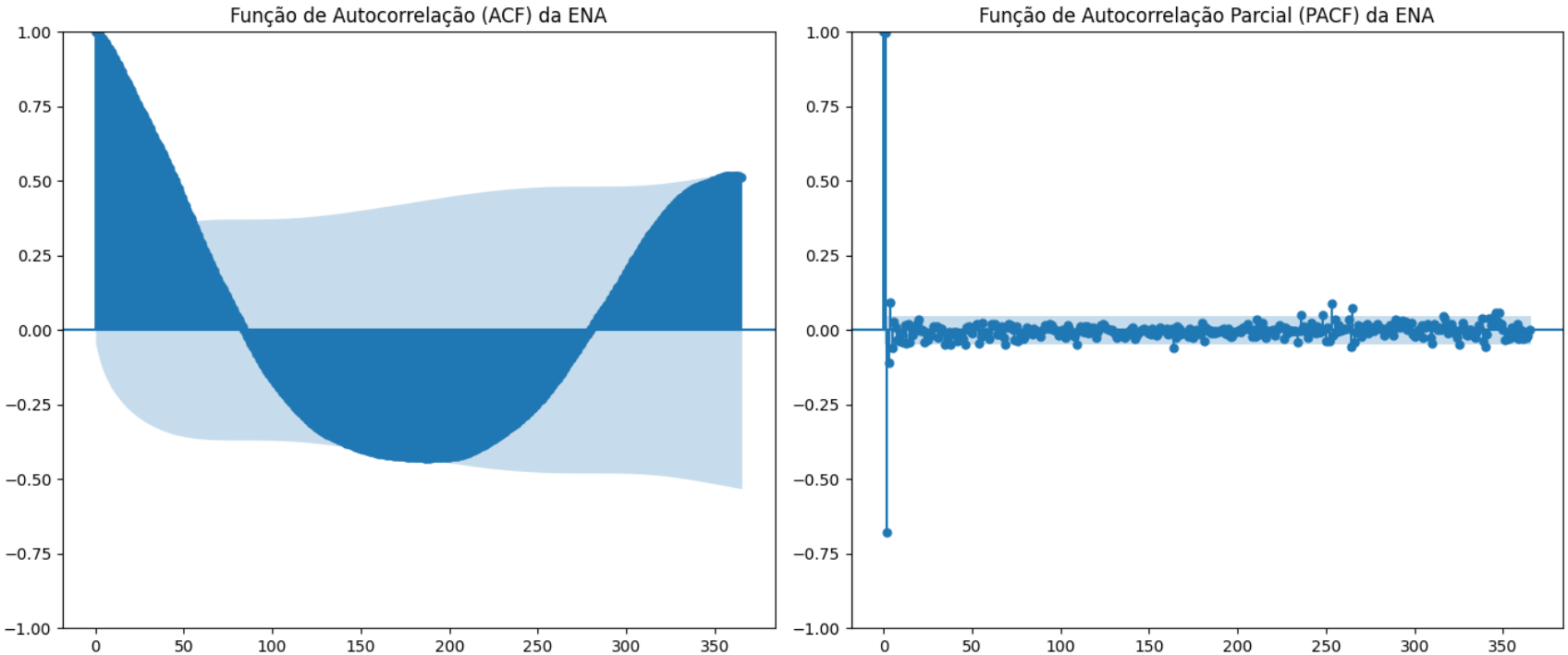

```
                ds    ena_di  ena_di_prev            rese
0              NaT       NaN     0.066360  APOLONIO SALES
1              NaT       NaN     0.076259  APOLONIO SALES
2              NaT       NaN     0.086440  APOLONIO SALES
3              NaT       NaN     0.096790  APOLONIO SALES
4              NaT       NaN     0.107960  APOLONIO SALES
...            ...       ...          ...             ...
261283 2025-09-18  0.016126          NaN  APOLONIO SALES
261438 2025-09-19  0.015384          NaN  APOLONIO SALES
261593 2025-09-20  0.013675          NaN  APOLONIO SALES
261748 2025-09-21  0.013366          NaN  APOLONIO SALES
261903 2025-09-22  0.015318          NaN  APOLONIO SALES
```

```
[1991 rows x 4 columns]
Shape X: (1696, 30, 4)
Shape y: (1696,)
Epoch 1/30
/usr/local/lib/python3.12/dist-packages/keras/src/layers/rnn/rnn.py:199: UserWarning: Do not pass an
  super().__init__(**kwargs)
43/43 ━━━━━━━━━━━━━━━━━━━━ 6s 52ms/step - loss: 0.0302 - val_loss: 0.0041
Epoch 2/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 46ms/step - loss: 0.0031 - val_loss: 0.0027
Epoch 3/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0024 - val_loss: 0.0018
Epoch 4/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0018 - val_loss: 0.0016
Epoch 5/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 3s 32ms/step - loss: 0.0018 - val_loss: 0.0014
Epoch 6/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - loss: 0.0019 - val_loss: 0.0013
Epoch 7/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 3s 33ms/step - loss: 0.0014 - val_loss: 0.0014
Epoch 8/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 3s 41ms/step - loss: 0.0015 - val_loss: 0.0013
Epoch 9/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0011 - val_loss: 0.0011
Epoch 10/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - loss: 9.9603e-04 - val_loss: 0.0011
Epoch 11/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0011 - val_loss: 0.0012
Epoch 12/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - loss: 0.0012 - val_loss: 0.0010
Epoch 13/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 34ms/step - loss: 8.4872e-04 - val_loss: 9.6283e-04
Epoch 14/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 3s 38ms/step - loss: 0.0010 - val_loss: 0.0011
Epoch 15/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 48ms/step - loss: 8.9382e-04 - val_loss: 0.0010
Epoch 16/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - loss: 9.0549e-04 - val_loss: 9.4060e-04
Epoch 17/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - loss: 8.4281e-04 - val_loss: 9.1264e-04
Epoch 18/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 9.2048e-04 - val_loss: 8.7772e-04
```

```
Epoch 19/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 34ms/step - loss: 8.2536e-04 - val_loss: 8.2840e-04
Epoch 20/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 3s 34ms/step - loss: 7.7533e-04 - val_loss: 8.8146e-04
Epoch 21/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 34ms/step - loss: 7.7607e-04 - val_loss: 8.9516e-04
Epoch 22/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 50ms/step - loss: 6.2930e-04 - val_loss: 8.0109e-04
Epoch 23/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 40ms/step - loss: 6.0943e-04 - val_loss: 7.9862e-04
Epoch 24/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 7.7472e-04 - val_loss: 7.3978e-04
Epoch 25/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - loss: 6.4676e-04 - val_loss: 8.0594e-04
Epoch 26/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 3s 33ms/step - loss: 7.1969e-04 - val_loss: 0.0010
Epoch 27/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 6.7767e-04 - val_loss: 8.3996e-04
Epoch 28/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 5.9189e-04 - val_loss: 7.5184e-04
Epoch 29/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 35ms/step - loss: 5.1229e-04 - val_loss: 6.8884e-04
Epoch 30/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 51ms/step - loss: 5.6337e-04 - val_loss: 6.9417e-04
11/11 ━━━━━━━━━━━━━━━━━━━━ 1s 49ms/step
MAE: 15.61
MSE: 465.17
RMSE: 21.57
R²: 0.98
MAPE: 13.00%
```

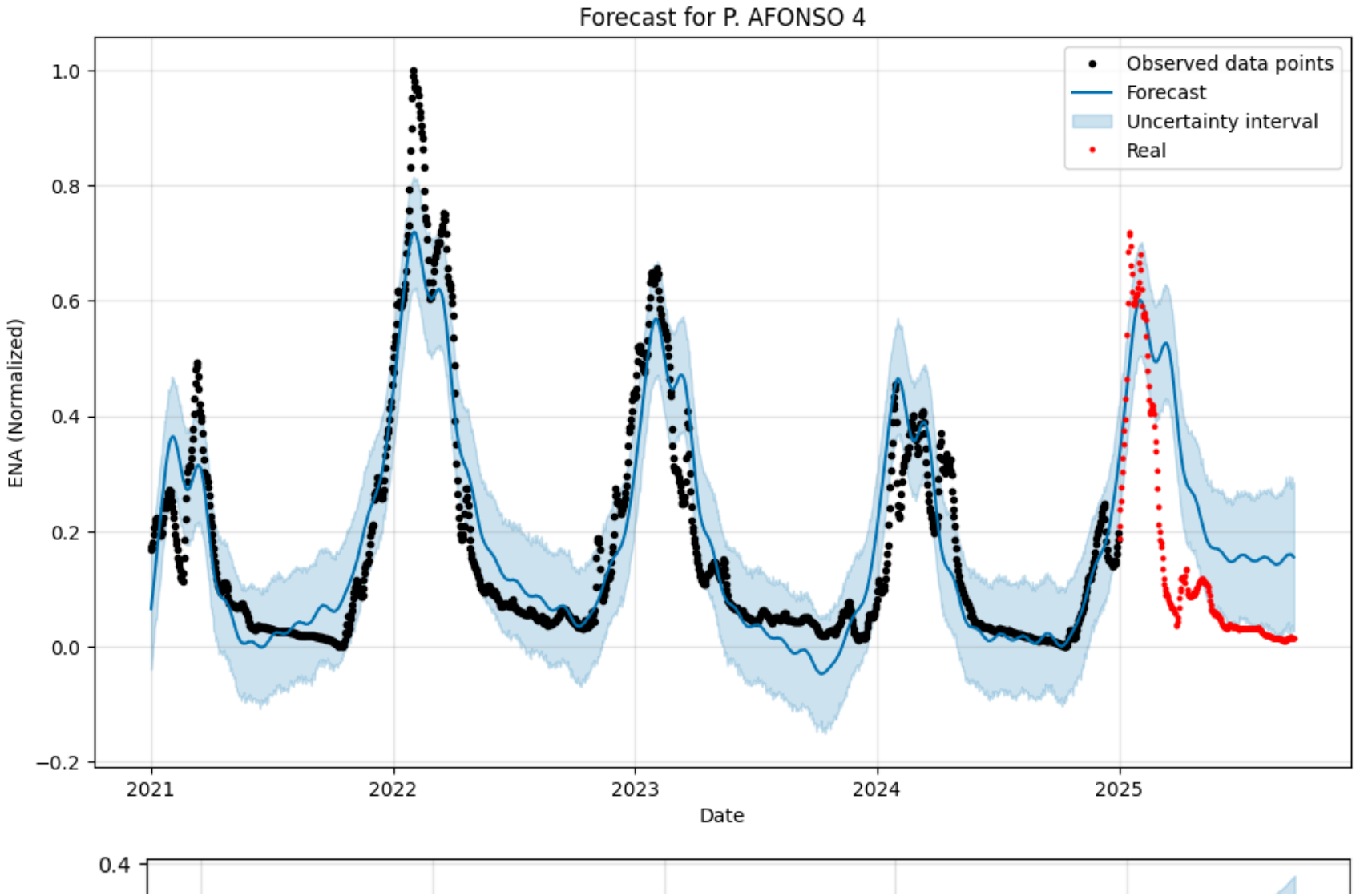


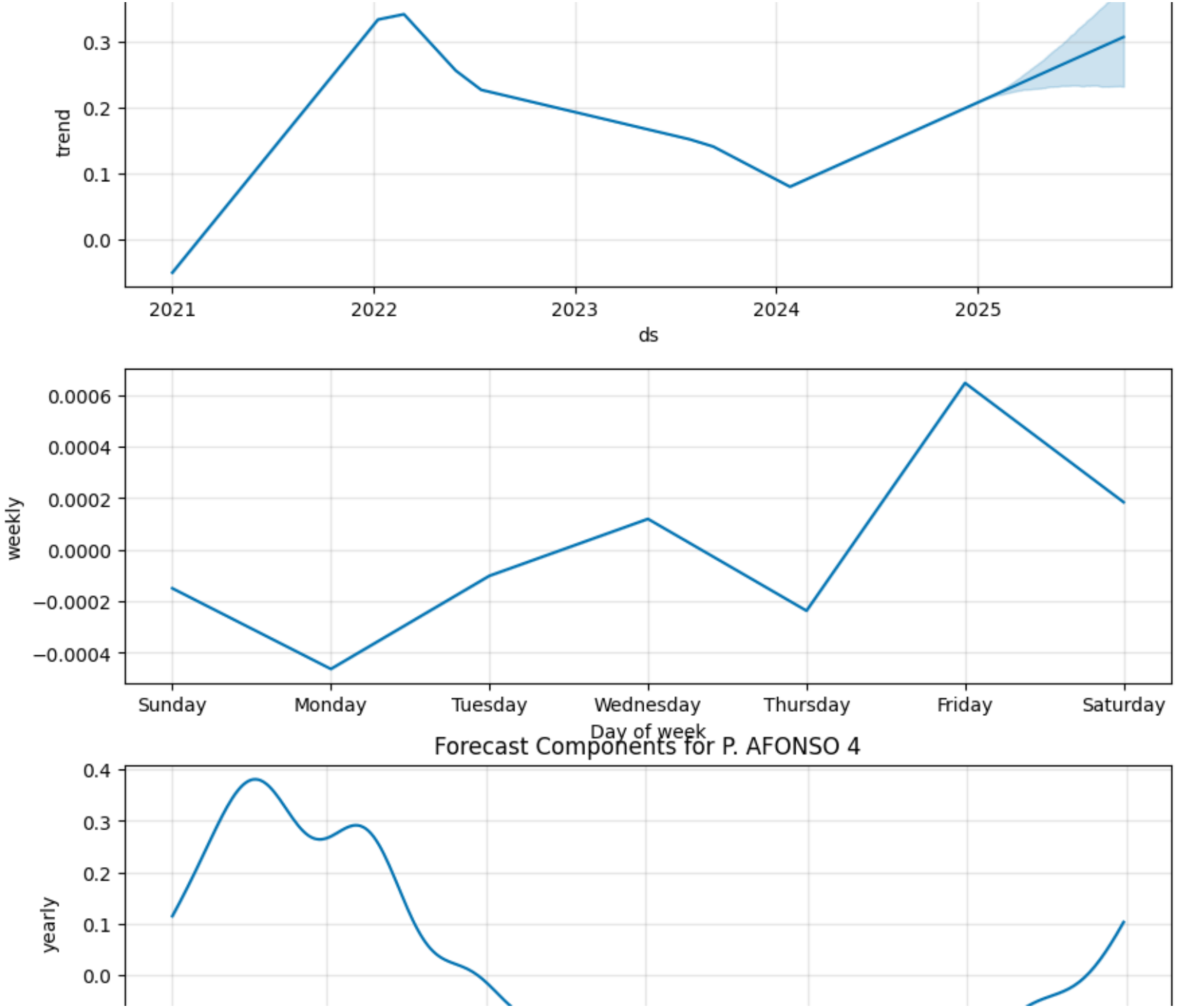

```
Reservatório: P. AFONSO 4
```

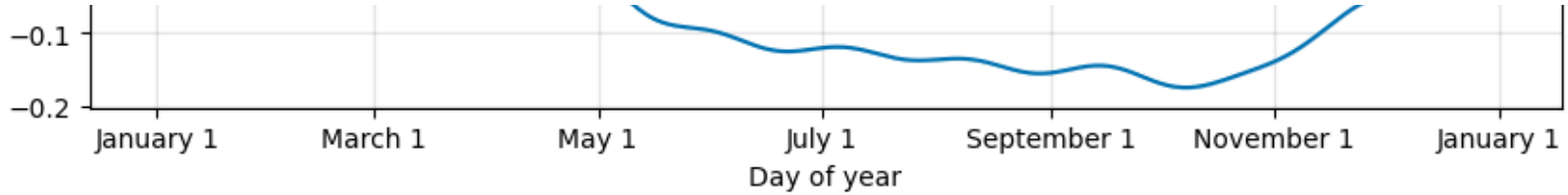



```
INFO:prophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to override this.
DEBUG:cmdstanpy:input tempfile: /tmp/tmp20qzy7c_/70m8a75j.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmp20qzy7c_/_rooiqlc.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.12/dist-packages/prophet/stan_model/prophet_mo
10:51:47 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
```

```
10:51:47 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
### Métricas de Avaliação do Modelo ###
Erro Médio Absoluto (MAE): 942.07 MWméd
Raiz do Erro Quadrático Médio (RMSE): 1308.71 MWméd
Erro Percentual Médio Absoluto (MAPE): 99.98%

Contexto: O valor médio real de ENA no período foi de 942.26 MWméd.
Um erro (MAE) de 942.07 representa aproximadamente 99.98% do valor médio.
```



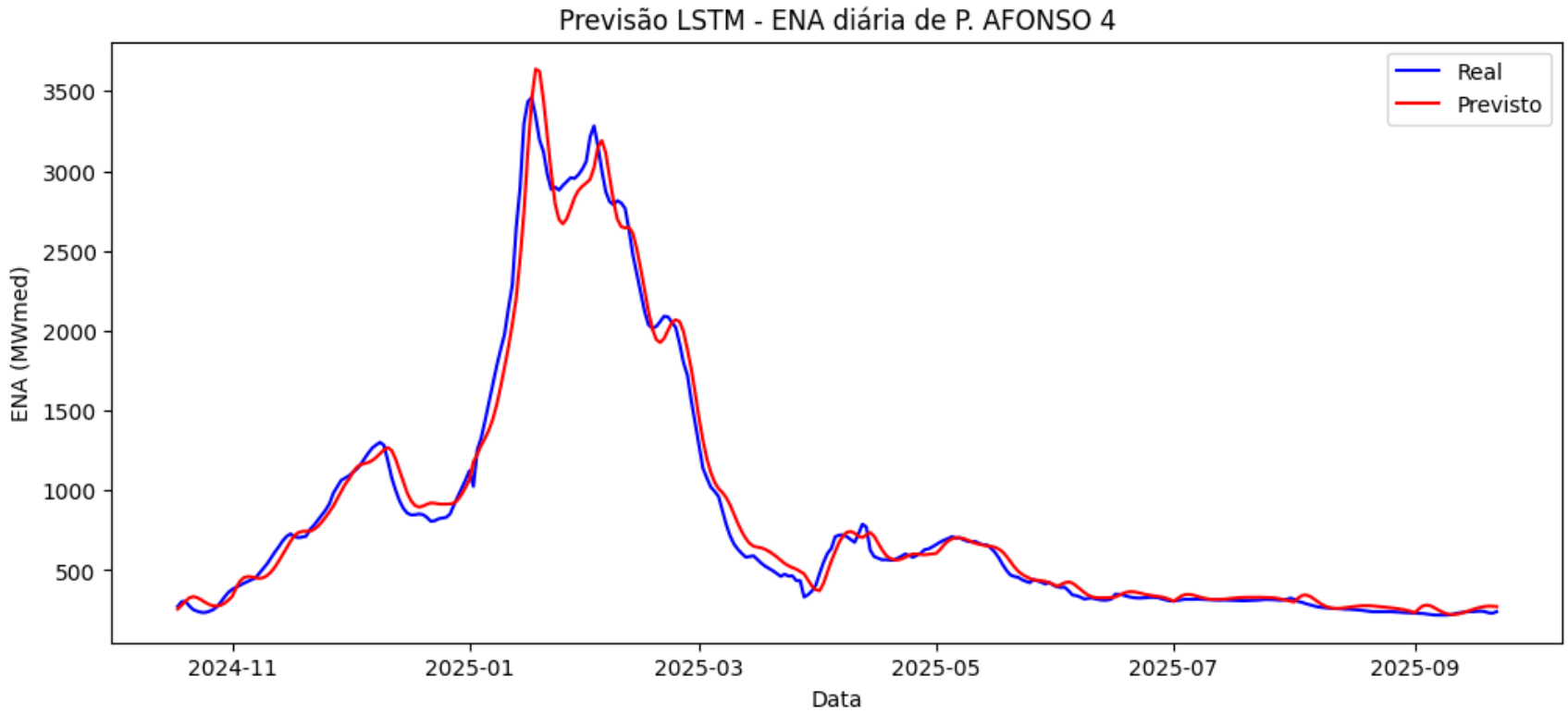

trend
0.3
0.2
0.1
0.0
2021
2022
2023
2024
2025
ds
weekly
0.0006
0.0004
0.0002
0.0000
−0.0002
−0.0004
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Day of week
Forecast Components for P. AFONSO 4
yearly
0.4
0.3
0.2
0.1
0.0




```
                ds    ena_di  ena_di_prev         rese
0              NaT       NaN     0.065472  P. AFONSO 4
1              NaT       NaN     0.075366  P. AFONSO 4
2              NaT       NaN     0.085542  P. AFONSO 4
3              NaT       NaN     0.095888  P. AFONSO 4
4              NaT       NaN     0.107053  P. AFONSO 4
...            ...       ...          ...          ...
261376 2025-09-18  0.016127          NaN  P. AFONSO 4
261531 2025-09-19  0.015383          NaN  P. AFONSO 4
261686 2025-09-20  0.013675          NaN  P. AFONSO 4
261841 2025-09-21  0.013365          NaN  P. AFONSO 4
261996 2025-09-22  0.015318          NaN  P. AFONSO 4

[1991 rows x 4 columns]
Shape X: (1696, 30, 4)
Shape y: (1696,)
Epoch 1/30
/usr/local/lib/python3.12/dist-packages/keras/src/layers/rnn/rnn.py:199: UserWarning: Do not pass an
  super().__init__(**kwargs)
43/43 ━━━━━━━━━━━━━━━━━━━━ 5s 43ms/step - loss: 0.0243 - val_loss: 0.0049
Epoch 2/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 3s 45ms/step - loss: 0.0035 - val_loss: 0.0028
Epoch 3/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 49ms/step - loss: 0.0030 - val_loss: 0.0018
Epoch 4/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 36ms/step - loss: 0.0021 - val_loss: 0.0017
Epoch 5/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 36ms/step - loss: 0.0018 - val_loss: 0.0015
Epoch 6/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 34ms/step - loss: 0.0015 - val_loss: 0.0014
Epoch 7/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 3s 34ms/step - loss: 0.0014 - val_loss: 0.0012
Epoch 8/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 35ms/step - loss: 0.0013 - val_loss: 0.0012
Epoch 9/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 38ms/step - loss: 0.0012 - val_loss: 0.0014
```

```
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 38ms/step - loss: 0.0012 - val_loss: 0.0014
Epoch 10/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 43ms/step - loss: 0.0011 - val_loss: 0.0013
Epoch 11/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 44ms/step - loss: 0.0011 - val_loss: 0.0011
Epoch 12/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 34ms/step - loss: 9.0204e-04 - val_loss: 0.0012
Epoch 13/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 3s 34ms/step - loss: 9.6657e-04 - val_loss: 0.0010
Epoch 14/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 36ms/step - loss: 8.5697e-04 - val_loss: 0.0010
Epoch 15/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 34ms/step - loss: 7.8633e-04 - val_loss: 0.0011
Epoch 16/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 38ms/step - loss: 7.1211e-04 - val_loss: 8.8394e-04
Epoch 17/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 50ms/step - loss: 8.4206e-04 - val_loss: 8.8448e-04
Epoch 18/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 34ms/step - loss: 7.9708e-04 - val_loss: 7.8678e-04
Epoch 19/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 36ms/step - loss: 7.6095e-04 - val_loss: 8.5910e-04
Epoch 20/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 35ms/step - loss: 7.9907e-04 - val_loss: 7.1172e-04
Epoch 21/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 36ms/step - loss: 6.5934e-04 - val_loss: 8.0521e-04
Epoch 22/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - loss: 7.0108e-04 - val_loss: 7.5679e-04
Epoch 23/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 35ms/step - loss: 7.9342e-04 - val_loss: 6.5438e-04
Epoch 24/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 3s 49ms/step - loss: 6.4714e-04 - val_loss: 6.7210e-04
Epoch 25/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 34ms/step - loss: 5.9056e-04 - val_loss: 6.7536e-04
Epoch 26/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 35ms/step - loss: 6.0181e-04 - val_loss: 5.7502e-04
Epoch 27/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 35ms/step - loss: 4.8965e-04 - val_loss: 5.9450e-04
Epoch 28/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 34ms/step - loss: 6.4027e-04 - val_loss: 5.8857e-04
Epoch 29/30
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 34ms/step - loss: 5.8721e-04 - val_loss: 8.4291e-04
Epoch 30/30
```

```
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 35ms/step - loss: 5.6484e-04 - val_loss: 4.9303e-04
11/11 ━━━━━━━━━━━━━━━━━━━━ 1s 48ms/step
MAE: 65.39
MSE: 10302.40
RMSE: 101.50
R²: 0.98
MAPE: 8.47%
```



Todos os dados previstos por ambos modelos são armazenos em formato csv, assim como os gráficos de teste para acompanhar a eficência do treinamento.

```
Start coding or generate with AI.
```