TDE02 - Séries Temporais Redes Neurais Recorrentes

RNN - Recurrent Neural Networks

Modelo

No TDE02 usaremos as redes neurais recorrentes para fazer a previsão de dados de desempenho utilizando redes neurais recorrentes com o seguinte mdelo:

- Primeira camada: `Lambda` (formata a entrada para a segunda camada)
- Segunda e terceira camadas: `simpleRNN`
- Quarta camada: `Dense`
- Quarta camada: `Lambda` (formata a saída da rede)
- Função perda de `Huber` (treinamento)

Importar bibliotecas

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

Conectar com google Drive

Funções

Cálcular de métricas de previsão

```
# Métricas de previsão
def metricas(previsto, observado):
    erro = previsto - observado                       # erro
    me = np.mean(erro)                                # ME
    mse = np.square(erro).mean()                      # MSE
    rmse = np.sqrt(mse)                               # RMSE
    mae = np.abs(erro).mean()                         # MAE
    mpe = (erro / observado).mean()                   # MPE
    mape = np.abs(erro / observado).mean()            # MAPE
    mins = np.amin(np.hstack([previsto[:,None],
                              observado[:,None]]), axis=1)
    maxs = np.amax(np.hstack([previsto[:,None],
                              observado[:,None]]), axis=1)
    minmax = 1 - np.mean(mins/maxs)                   # MINMAX
    return({'ME':me, 'MSE':mse, 'RMSE':rmse,
```

```
            'MAE': mae, 'MPE': mpe, 'MAPE':mape,
            'MIN-MAX':minmax})
```

Plotar séries

```
### Plotar séries temporais
def plotar_series(tempo, series, format="-", inicio=0, fim=None):

    # Dimensões da figura
    plt.figure(figsize=(10, 6))

    if type(series) is tuple:
        for series_num in series:
            # Plotar valores x tempo
            plt.plot(tempo[inicio:fim], series_num[inicio:fim],
format)
    else:
        # Plotar valores x tempo
        plt.plot(tempo[inicio:fim], series[inicio:fim], format)

    # Rótulo do eixo x
    plt.xlabel("Tempo")

    # Rótulo do eixo y
    plt.ylabel("Valor")

    # Plotar grid
    plt.grid(True)

    # Mostrar a gráfico
    plt.show()
```

Função para janelamento e lotes

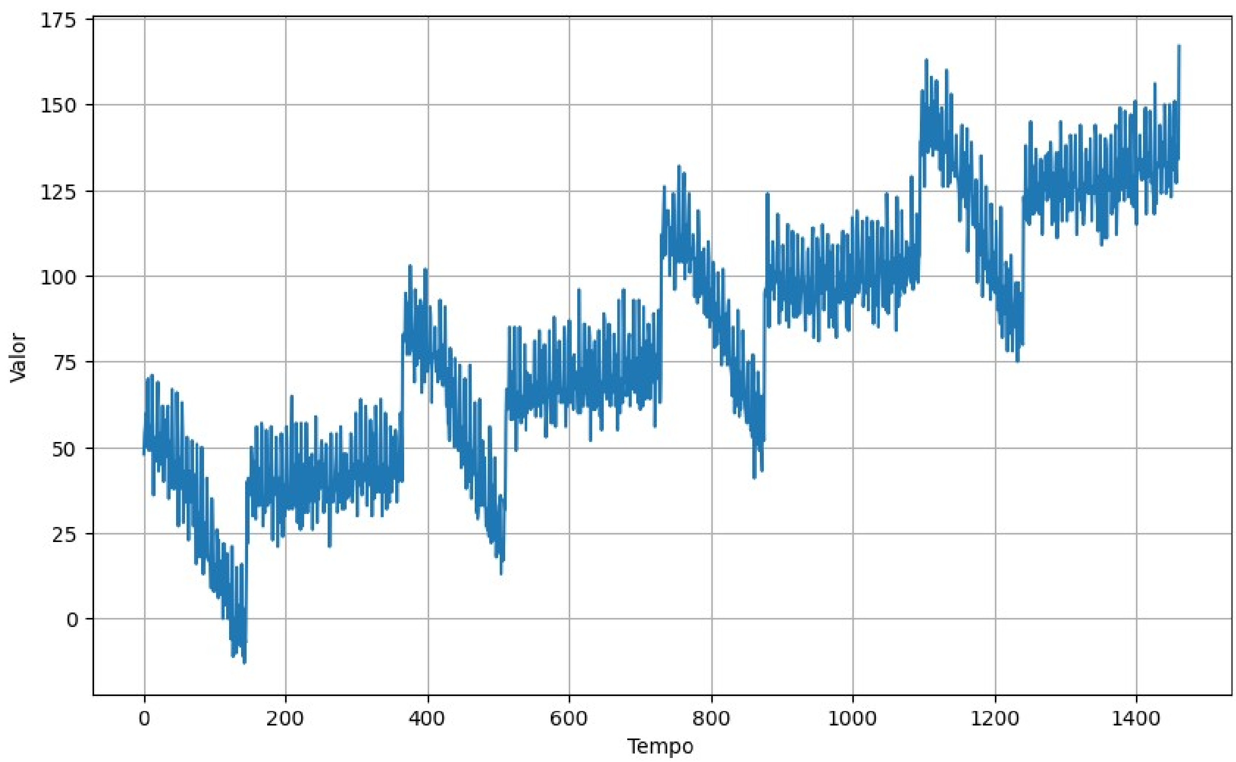
```
def janelamento_lotes(serie, tam_janela, tam_lote,
buffer_embaralhamento):
     # Cria um dataset TF Dataset a partir dos valores da serie
    dataset = tf.data.Dataset.from_tensor_slices(serie)

    # Janelamento dos dados
    dataset = dataset.window(tam_janela + 1, shift=1,
drop_remainder=True)

    # Ajustar as janelas (flatten) colocando seus elementos em lotes
    dataset = dataset.flat_map(lambda window: window.batch(tam_janela
+ 1))

    # Criar tuplas com variáveis (features) e rótulos (labels)
    dataset = dataset.map(lambda window: (window[:-1], window[-1]))
```

```
    # Embaralhar janelas
    dataset = dataset.shuffle(buffer_embaralhamento)

    # Criar lotes de treinamento
    dataset = dataset.batch(tam_lote).prefetch(1)

    return dataset
```

Preparar dados

- Carregar os valores da série temporal do arquivo **serie_sintetica_D.csv**
- Gerar os valores de tempo
- Plotar a série

```
# Ler os dados
serie = np.loadtxt("/content/serie_sintetica_D.csv", delimiter=",",
dtype="float32")
# Criar array com o tempo
tempo = np.arange(len(serie), dtype="float32")
# Plotar série
plotar_series(tempo, serie)
```



Dividir os dados

```
# Definir o tamanho do conjunto de treinamento
tam_trein = 1000
```

```
# Conjunto de treinamento
tempo_trein = tempo[:tam_trein]
x_trein = serie[:tam_trein]

# Conjunto de validação
tempo_valid = tempo[tam_trein:]
x_valid = serie[tam_trein:]
```

Preparar dataset para treinamento

Parâmetros

```
# Parameteros
tam_janela = 20
tam_lote = 32
tam_buffer_embaralhamento = 1000
```

Criar o dataset

```
# Criar o dataset de treinamento
dataset = janelamento_lotes(x_trein, tam_janela, tam_lote,
tam_buffer_embaralhamento)
```

Imprimir formato de variáveis (features) e rótulos (labels)

```
# Imprimir formato de variáveis e rótulos
for janela in dataset.take(1):
  print(f'Formato das variáveis: {janela[0].shape}')
  print(f'Formato dos rótulos: {janela[1].shape}')

Formato das variáveis: (32, 20)
Formato dos rótulos: (32,)
```

Construir o modelo

Modelo

Modelo sequencial composto por 5 camadas:

Primeira camada do tipo lambda função Lambda. Transforma a entrada para um tensor tridimensional no formato [tamanho do lote, intervalos de tempo, features], conforme exigido pela camada SimpleRNN (ver documentação). A função `Lambda` remodela o formato do dataset de [tam_janela, tam_lote] para [tam_janela, tam_lote, 1], adicionando uma dimensão no último eixo da entrada. A quantidade de pontos de dados na janela (tam_janela) são mapeados para a mesma quantidade de intervalos de tempo da RNN ao definir o parâmetro `input_shape` igual ao tamanho da janela (`tam_janela`).

Segunda e a terceira camadas do tipo SimpleRNN. O primeiro argumento para camadas `SimpleRNN` é um inteiro positivo com a dimensionalidade da saída, que deve ser múltiplo do tamanho da janela. A segunda camada deverá ter o argumento `return_sequences` definido como `True` para encaminhar sua saída de volta para a entrada da terceira camada.

Quarta camada do tipo densa (`Dense`).

Quinta camada do tipo função `Lambda`. Incluída para facilitar o treinamento. Configura a saída para valores próximos aos valores observados na série. Como as camadas SimpleRNN usam a função de ativação `tanh` que definem um intervalo de saída entre [-1,1] e os valores observados na série estão próximos de 100, a reformatação é usada uma camada Lambda() que multiplica o tamanho da saída por 100.

```python
modelo_RNN = tf.keras.models.Sequential([
  tf.keras.layers.Lambda(lambda x: tf.expand_dims(x, axis=-1),
                    input_shape=[tam_janela]),
  tf.keras.layers.SimpleRNN(2*tam_janela, return_sequences=True),
  tf.keras.layers.SimpleRNN(2*tam_janela),
  tf.keras.layers.Dense(1),
  tf.keras.layers.Lambda(lambda x: x * 100.0)
])

# Imprimir sumário
modelo_RNN.summary()
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/
lambda_layer.py:65: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(**kwargs)

Model: "sequential"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┓
━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                          ┃ Output Shape                 ┃
Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┩
━━━━━━━━━━━━━━━━━┩
│ lambda (Lambda)                       │ (None, 20, 1)                │
0 │
├───────────────────────────────────────┼──────────────────────────────┤
─────────────────┤
│ simple_rnn (SimpleRNN)                │ (None, 20, 40)               │
1,680 │
├───────────────────────────────────────┼──────────────────────────────┤
─────────────────┤
│ simple_rnn_1 (SimpleRNN)              │ (None, 40)                   │
3,240 │
├───────────────────────────────────────┼──────────────────────────────┤
```

```
────────────────────┤
│ dense (Dense)                                    │ (None, 1)                             │
41 │
├──────────────────────────────────────────────────┼───────────────────────────────────────┼
────────────────────┤
│ lambda_1 (Lambda)                                │ (None, 1)                             │
0 │
└──────────────────────────────────────────────────┴───────────────────────────────────────┴
────────────────────┘

 Total params: 4,961 (19.38 KB)

 Trainable params: 4,961 (19.38 KB)

 Non-trainable params: 0 (0.00 B)
```

Ajustar a taxa de aprendizagem

Treinar o modelo para ajuste de taxa de aprendizagem

Definir parâmetros:

- Função LearningRateScheduler para ajuste no callback
- Algoritmo de otimização: SGD
- Função de perda: Huber para minimizar a sensibilidade a outliers.

Treinar o modelo

```
# Função de callback para escalonamento de taxa de aprendizagem
lr_schedule = tf.keras.callbacks.LearningRateScheduler(
    lambda epoch: 1e-8 * 10**(epoch / 20))

# Definir algoritmo de otimização
optimizador = tf.keras.optimizers.SGD(momentum=0.9)

# Definir função de perda
modelo_RNN.compile(loss=tf.keras.losses.Huber(),
optimizer=optimizador)

# Treinar o modelo
resultado_treinamento = modelo_RNN.fit(dataset, epochs=100,
callbacks=[lr_schedule])

Epoch 1/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 3s 10ms/step - loss: 20.5146 -
learning_rate: 1.0000e-08
Epoch 2/100

/usr/lib/python3.10/contextlib.py:153: UserWarning: Your input ran out
of data; interrupting training. Make sure that your dataset or
```

```
generator can generate at least `steps_per_epoch * epochs` batches.
You may need to use the `.repeat()` function when building your
dataset.
  self.gen.throw(typ, value, traceback)

31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 18.5910 -
learning_rate: 1.1220e-08
Epoch 3/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 17.5031 -
learning_rate: 1.2589e-08
Epoch 4/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 18.0856 -
learning_rate: 1.4125e-08
Epoch 5/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 18.3818 -
learning_rate: 1.5849e-08
Epoch 6/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 18.6415 -
learning_rate: 1.7783e-08
Epoch 7/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 17.4237 -
learning_rate: 1.9953e-08
Epoch 8/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 16.7274 -
learning_rate: 2.2387e-08
Epoch 9/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 17.0898 -
learning_rate: 2.5119e-08
Epoch 10/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - loss: 16.9835 -
learning_rate: 2.8184e-08
Epoch 11/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 18ms/step - loss: 17.0992 -
learning_rate: 3.1623e-08
Epoch 12/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 16.6900 -
learning_rate: 3.5481e-08
Epoch 13/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 16.6142 -
learning_rate: 3.9811e-08
Epoch 14/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 16.7250 -
learning_rate: 4.4668e-08
Epoch 15/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 15.4876 -
learning_rate: 5.0119e-08
Epoch 16/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 15.6782 -
learning_rate: 5.6234e-08
Epoch 17/100
```

```
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 15.5584 -
learning_rate: 6.3096e-08
Epoch 18/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 14.9415 -
learning_rate: 7.0795e-08
Epoch 19/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 14.7870 -
learning_rate: 7.9433e-08
Epoch 20/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 14.3104 -
learning_rate: 8.9125e-08
Epoch 21/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 14.1978 -
learning_rate: 1.0000e-07
Epoch 22/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 13.7860 -
learning_rate: 1.1220e-07
Epoch 23/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - loss: 13.4100 -
learning_rate: 1.2589e-07
Epoch 24/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 13.5280 -
learning_rate: 1.4125e-07
Epoch 25/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 13.6873 -
learning_rate: 1.5849e-07
Epoch 26/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 12.9754 -
learning_rate: 1.7783e-07
Epoch 27/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 12.7676 -
learning_rate: 1.9953e-07
Epoch 28/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - loss: 12.8392 -
learning_rate: 2.2387e-07
Epoch 29/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 18ms/step - loss: 11.1665 -
learning_rate: 2.5119e-07
Epoch 30/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 18ms/step - loss: 12.0939 -
learning_rate: 2.8184e-07
Epoch 31/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - loss: 12.1196 -
learning_rate: 3.1623e-07
Epoch 32/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 11.2174 -
learning_rate: 3.5481e-07
Epoch 33/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - loss: 10.7991 -
```

```
learning_rate: 3.9811e-07
Epoch 34/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 11.6812 -
learning_rate: 4.4668e-07
Epoch 35/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 11.8122 -
learning_rate: 5.0119e-07
Epoch 36/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 10.8729 -
learning_rate: 5.6234e-07
Epoch 37/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 11.5444 -
learning_rate: 6.3096e-07
Epoch 38/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - loss: 10.9216 -
learning_rate: 7.0795e-07
Epoch 39/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 10.2995 -
learning_rate: 7.9433e-07
Epoch 40/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 10.6656 -
learning_rate: 8.9125e-07
Epoch 41/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - loss: 11.7357 -
learning_rate: 1.0000e-06
Epoch 42/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 10.8912 -
learning_rate: 1.1220e-06
Epoch 43/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 12.3886 -
learning_rate: 1.2589e-06
Epoch 44/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 10.2559 -
learning_rate: 1.4125e-06
Epoch 45/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 11.2771 -
learning_rate: 1.5849e-06
Epoch 46/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 11.1895 -
learning_rate: 1.7783e-06
Epoch 47/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 10.2717 -
learning_rate: 1.9953e-06
Epoch 48/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - loss: 11.5808 -
learning_rate: 2.2387e-06
Epoch 49/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - loss: 11.0075 -
learning_rate: 2.5119e-06
```

```
Epoch 50/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - loss: 10.0990 -
learning_rate: 2.8184e-06
Epoch 51/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 10.6549 -
learning_rate: 3.1623e-06
Epoch 52/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 9.6731 -
learning_rate: 3.5481e-06
Epoch 53/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - loss: 10.2547 -
learning_rate: 3.9811e-06
Epoch 54/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 9.4757 -
learning_rate: 4.4668e-06
Epoch 55/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 16.6387 -
learning_rate: 5.0119e-06
Epoch 56/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 11.1754 -
learning_rate: 5.6234e-06
Epoch 57/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - loss: 9.0353 -
learning_rate: 6.3096e-06
Epoch 58/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 10.5498 -
learning_rate: 7.0795e-06
Epoch 59/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 10.0655 -
learning_rate: 7.9433e-06
Epoch 60/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 12.4224 -
learning_rate: 8.9125e-06
Epoch 61/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 9.6848 -
learning_rate: 1.0000e-05
Epoch 62/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 13.8264 -
learning_rate: 1.1220e-05
Epoch 63/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 10.6117 -
learning_rate: 1.2589e-05
Epoch 64/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 18.6919 -
learning_rate: 1.4125e-05
Epoch 65/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 15.5350 -
learning_rate: 1.5849e-05
Epoch 66/100
```

```
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 13.7956 -
learning_rate: 1.7783e-05
Epoch 67/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 13.9408 -
learning_rate: 1.9953e-05
Epoch 68/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 14.8531 -
learning_rate: 2.2387e-05
Epoch 69/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - loss: 15.0892 -
learning_rate: 2.5119e-05
Epoch 70/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 17ms/step - loss: 12.0099 -
learning_rate: 2.8184e-05
Epoch 71/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 19ms/step - loss: 16.7686 -
learning_rate: 3.1623e-05
Epoch 72/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 13.4157 -
learning_rate: 3.5481e-05
Epoch 73/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 20.7957 -
learning_rate: 3.9811e-05
Epoch 74/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 15.8446 -
learning_rate: 4.4668e-05
Epoch 75/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 17.2856 -
learning_rate: 5.0119e-05
Epoch 76/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - loss: 18.1880 -
learning_rate: 5.6234e-05
Epoch 77/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 22.9493 -
learning_rate: 6.3096e-05
Epoch 78/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 22.5715 -
learning_rate: 7.0795e-05
Epoch 79/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 21.7857 -
learning_rate: 7.9433e-05
Epoch 80/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 23.3585 -
learning_rate: 8.9125e-05
Epoch 81/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 19.5045 -
learning_rate: 1.0000e-04
Epoch 82/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 22.6758 -
```

```
learning_rate: 1.1220e-04
Epoch 83/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - loss: 23.2631 -
learning_rate: 1.2589e-04
Epoch 84/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 24.3410 -
learning_rate: 1.4125e-04
Epoch 85/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 23.2483 -
learning_rate: 1.5849e-04
Epoch 86/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 23.8898 -
learning_rate: 1.7783e-04
Epoch 87/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 22.3567 -
learning_rate: 1.9953e-04
Epoch 88/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - loss: 25.4452 -
learning_rate: 2.2387e-04
Epoch 89/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - loss: 28.4218 -
learning_rate: 2.5119e-04
Epoch 90/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 18ms/step - loss: 24.3553 -
learning_rate: 2.8184e-04
Epoch 91/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 17ms/step - loss: 27.4770 -
learning_rate: 3.1623e-04
Epoch 92/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 23.3223 -
learning_rate: 3.5481e-04
Epoch 93/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 22.3032 -
learning_rate: 3.9811e-04
Epoch 94/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 25.5876 -
learning_rate: 4.4668e-04
Epoch 95/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 22.1305 -
learning_rate: 5.0119e-04
Epoch 96/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 25.5256 -
learning_rate: 5.6234e-04
Epoch 97/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 25.8699 -
learning_rate: 6.3096e-04
Epoch 98/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 26.5455 -
learning_rate: 7.0795e-04
```

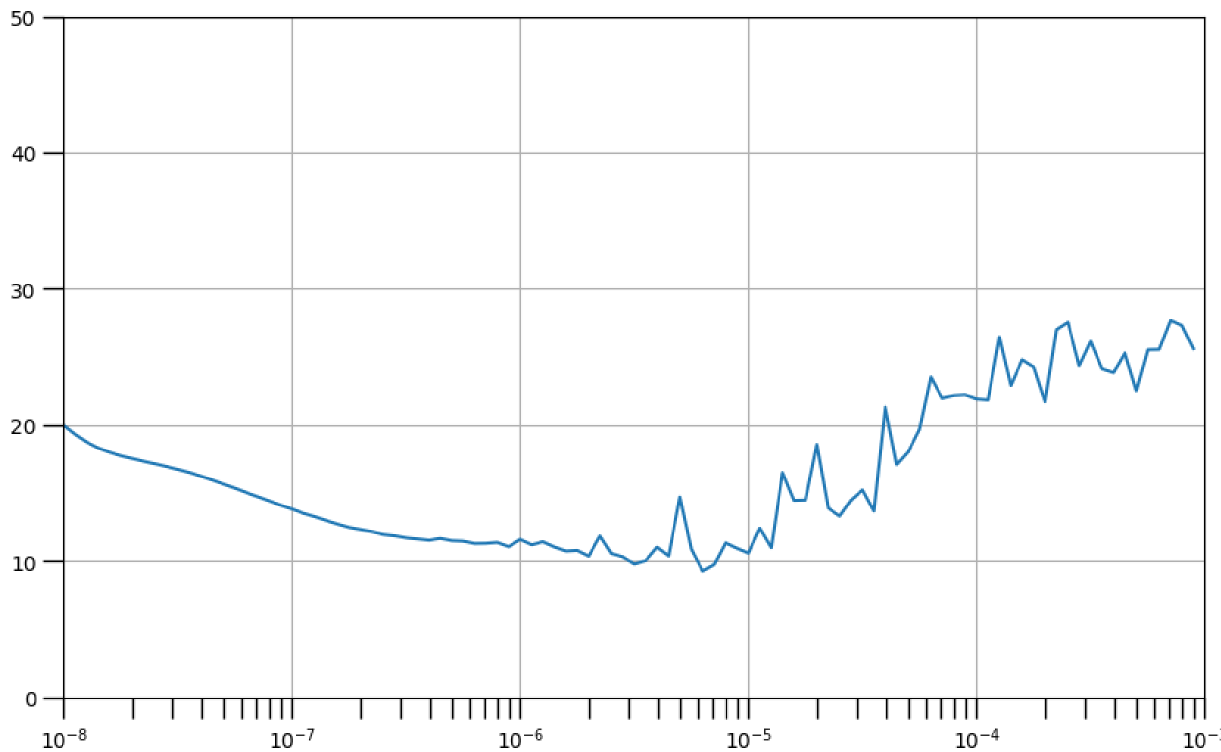
```
Epoch 99/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 26.5924 -
learning_rate: 7.9433e-04
Epoch 100/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 27.7154 -
learning_rate: 8.9125e-04
```

Visualizar perdas para selecionar taxa de aprendizagem

Plotar perda em função da taxa de aprendizagem

```
# Definir o array de taxas de aprendizagem
taxas_aprendizagem = 1e-8 * (10 ** (np.arange(100) / 20))

# Definir tamanho da figura
plt.figure(figsize=(10, 6))

# Definir grid
plt.grid(True)

# Plotar perda em escala logarítmica
plt.semilogx(taxas_aprendizagem,
resultado_treinamento.history["loss"])

# Aumentar tamanho das marcas de ticks
plt.tick_params('both', length=10, width=1, which='both')

# Definir limites de plotagem
plt.axis([1e-8, 1e-3, 0, 50])
```

```
(1e-08, 0.001, 0.0, 50.0)
```

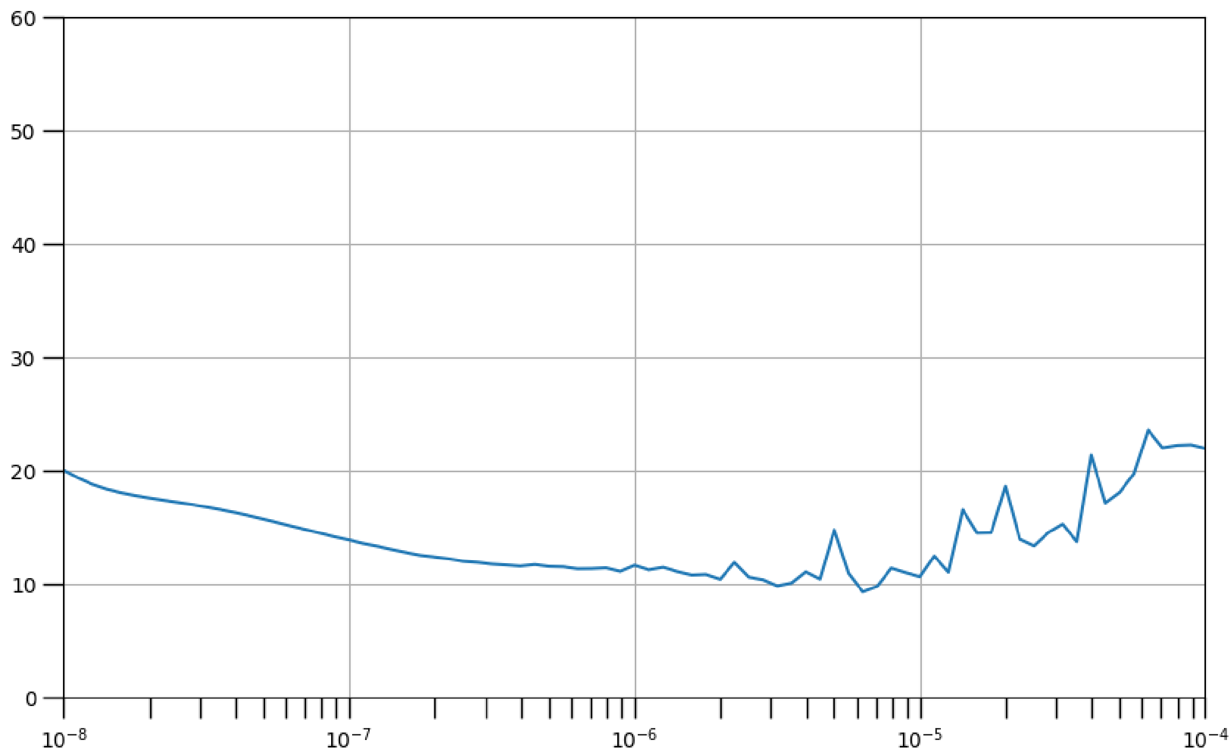


Alterar os limites do gráfico (zoom in)

Observar onde o gráfico se torna instável.

```
# Tamanho da figura
plt.figure(figsize=(10, 6))

# Definir grid
plt.grid(True)

# Plotar perda em escala logarítmica
plt.semilogx(taxas_aprendizagem,
resultado_treinamento.history["loss"])

# Aumentar tamanho das marcas de ticks
plt.tick_params('both', length=10, width=1, which='both')

# Definir limites de plotagem
plt.axis([1e-8, 1e-4, 0, 60])
```

```
(1e-08, 0.0001, 0.0, 60.0)
```



Construir o modelo

```
# Construir modelo RNN
modelo_RNN = tf.keras.models.Sequential([
  tf.keras.layers.Lambda(lambda x: tf.expand_dims(x, axis=-1),
                      input_shape=[tam_janela]),
  tf.keras.layers.SimpleRNN(2*tam_janela, return_sequences=True),
  tf.keras.layers.SimpleRNN(2*tam_janela),
  tf.keras.layers.Dense(1),
  tf.keras.layers.Lambda(lambda x: x * 100.0)
])
```

Treinar o modelo

Usar a taxa de aprendizagem `1e-6` (pode testar com outros valores).

```
# Definir taxa de aprendizagem
taxa_de_aprendizagem = 1e-6
#taxa_de_aprendizagem = 1e-7

# Definir algoritmo de otimização
otimizador =
tf.keras.optimizers.SGD(learning_rate=taxa_de_aprendizagem,
momentum=0.9)

# Definir parâmetros
```

```python
modelo_RNN.compile(loss=tf.keras.losses.Huber(),
              optimizer=otimizador,
              metrics=["mae"])

# Treinar o modelo
resultado_treinamento_RNN = modelo_RNN.fit(dataset,epochs=100)
```

```
Epoch 1/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 3s 10ms/step - loss: 23.1926 - mae: 23.6882
Epoch 2/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 17.3315 - mae: 17.8253
Epoch 3/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 17ms/step - loss: 16.2537 - mae: 16.7486
Epoch 4/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 20ms/step - loss: 15.7438 - mae: 16.2378
Epoch 5/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - loss: 13.9522 - mae: 14.4456
Epoch 6/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 13.5374 - mae: 14.0317
Epoch 7/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 12.7374 - mae: 13.2316
Epoch 8/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 11.9055 - mae: 12.3939
Epoch 9/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 11.2650 - mae: 11.7557
Epoch 10/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 11.2251 - mae: 11.7144
Epoch 11/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 10.0258 - mae: 10.5121
Epoch 12/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 9.5520 - mae: 10.0392
Epoch 13/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 9.6412 - mae: 10.1291
Epoch 14/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 20ms/step - loss: 9.2726 - mae: 9.7608
Epoch 15/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 9.3874 - mae: 9.8730
Epoch 16/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 9.4785 - mae: 9.9649
Epoch 17/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 9.1791 - mae: 9.6636
Epoch 18/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 10.1642 - mae: 10.6559
Epoch 19/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 9.2424 - mae: 9.7298
Epoch 20/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 9.2997 - mae: 9.7828
Epoch 21/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 9.1802 - mae: 9.6652
Epoch 22/100
```

```
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - loss: 8.9598 - mae: 9.4470
Epoch 23/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 18ms/step - loss: 9.0067 - mae: 9.4943
Epoch 24/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 8.6963 - mae: 9.1830
Epoch 25/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 9.4800 - mae: 9.9648
Epoch 26/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 9.6580 - mae: 10.1465
Epoch 27/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 8.4966 - mae: 8.9843
Epoch 28/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 8.8624 - mae: 9.3537
Epoch 29/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 8.7135 - mae: 9.1973
Epoch 30/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 9.0215 - mae: 9.5142
Epoch 31/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 8.5199 - mae: 9.0089
Epoch 32/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 9.5104 - mae: 10.0021
Epoch 33/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 9.4572 - mae: 9.9442
Epoch 34/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 9.1335 - mae: 9.6225
Epoch 35/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - loss: 8.8147 - mae: 9.3029
Epoch 36/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 9.0467 - mae: 9.5332
Epoch 37/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 8.7472 - mae: 9.2372
Epoch 38/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 9.1512 - mae: 9.6373
Epoch 39/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 8.8695 - mae: 9.3594
Epoch 40/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 8.8767 - mae: 9.3635
Epoch 41/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 19ms/step - loss: 8.7956 - mae: 9.2838
Epoch 42/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 18ms/step - loss: 8.8724 - mae: 9.3602
Epoch 43/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 20ms/step - loss: 8.5634 - mae: 9.0512
Epoch 44/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 8.6246 - mae: 9.1082
Epoch 45/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 8.5236 - mae: 9.0112
Epoch 46/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 8.7534 - mae: 9.2401
```

```
Epoch 47/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 7.8875 - mae: 8.3724
Epoch 48/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 8.9622 - mae: 9.4534
Epoch 49/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 8.5391 - mae: 9.0258
Epoch 50/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 8.4414 - mae: 8.9292
Epoch 51/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 7.8235 - mae: 8.3056
Epoch 52/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - loss: 8.2772 - mae: 8.7610
Epoch 53/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 8.5267 - mae: 9.0168
Epoch 54/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 8.4197 - mae: 8.9050
Epoch 55/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 8.7124 - mae: 9.1965
Epoch 56/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 8.5725 - mae: 9.0585
Epoch 57/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 8.6811 - mae: 9.1690
Epoch 58/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 8.5075 - mae: 8.9959
Epoch 59/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 8.2983 - mae: 8.7847
Epoch 60/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 8.6940 - mae: 9.1816
Epoch 61/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 7.8754 - mae: 8.3643
Epoch 62/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 7.7349 - mae: 8.2181
Epoch 63/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - loss: 8.4891 - mae: 8.9697
Epoch 64/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 18ms/step - loss: 8.4668 - mae: 8.9522
Epoch 65/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 18ms/step - loss: 8.3536 - mae: 8.8408
Epoch 66/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - loss: 8.0448 - mae: 8.5267
Epoch 67/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 8.2099 - mae: 8.6991
Epoch 68/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - loss: 8.4243 - mae: 8.9090
Epoch 69/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 8.2717 - mae: 8.7543
Epoch 70/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 8.3773 - mae: 8.8627
Epoch 71/100
```

```
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 7.8381 - mae: 8.3215
Epoch 72/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 8.4879 - mae: 8.9711
Epoch 73/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 8.3853 - mae: 8.8692
Epoch 74/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 7.7638 - mae: 8.2492
Epoch 75/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 7.9307 - mae: 8.4135
Epoch 76/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 8.2927 - mae: 8.7765
Epoch 77/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 7.9129 - mae: 8.3995
Epoch 78/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 7.7349 - mae: 8.2190
Epoch 79/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 8.2716 - mae: 8.7544
Epoch 80/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 8.3659 - mae: 8.8530
Epoch 81/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 8.1915 - mae: 8.6778
Epoch 82/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 8.3183 - mae: 8.8022
Epoch 83/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 7.9840 - mae: 8.4719
Epoch 84/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - loss: 7.3551 - mae: 7.8360
Epoch 85/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 18ms/step - loss: 7.9362 - mae: 8.4173
Epoch 86/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 8.2312 - mae: 8.7188
Epoch 87/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 7.9010 - mae: 8.3863
Epoch 88/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 8.4020 - mae: 8.8904
Epoch 89/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - loss: 8.4453 - mae: 8.9326
Epoch 90/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 7.7182 - mae: 8.2002
Epoch 91/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 8.0101 - mae: 8.4956
Epoch 92/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 8.2163 - mae: 8.7012
Epoch 93/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 8.0939 - mae: 8.5794
Epoch 94/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 8.1932 - mae: 8.6803
Epoch 95/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - loss: 8.3882 - mae: 8.8762
```

```
Epoch 96/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 7.9637 - mae: 8.4489
Epoch 97/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 8.2900 - mae: 8.7715
Epoch 98/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 8.8865 - mae: 9.3701
Epoch 99/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 7.9113 - mae: 8.3912
Epoch 100/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 8.3719 - mae: 8.8586
```

Previsão

O modelo é maior do que os utilizados anteriormente e a natureza sequencial das RNNs tornam as previsões mais lentas. Nas RNNs as entradas passam por uma série de etapas de tempo sequenciais, em vez do processamento paralelo.

```
# Inicializar lista de previsões
forecast = []

# Selecionar pontos que estão alinhados com o conjunto de validação
serie_valid = serie[tam_trein - tam_janela:]

# Usar o modelo para prever um valor para cada janela
for indice in range(len(serie_valid) - tam_janela):
  forecast.append(modelo_RNN.predict(serie_valid[indice:indice +
tam_janela][np.newaxis]))

# Converter para formato esperado pela função plotar_series
previsao = np.array(forecast).squeeze()

# Plotar as séries de validação e de previsão
plotar_series(tempo_valid, (x_valid, previsao))
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 276ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 49ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 53ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
```

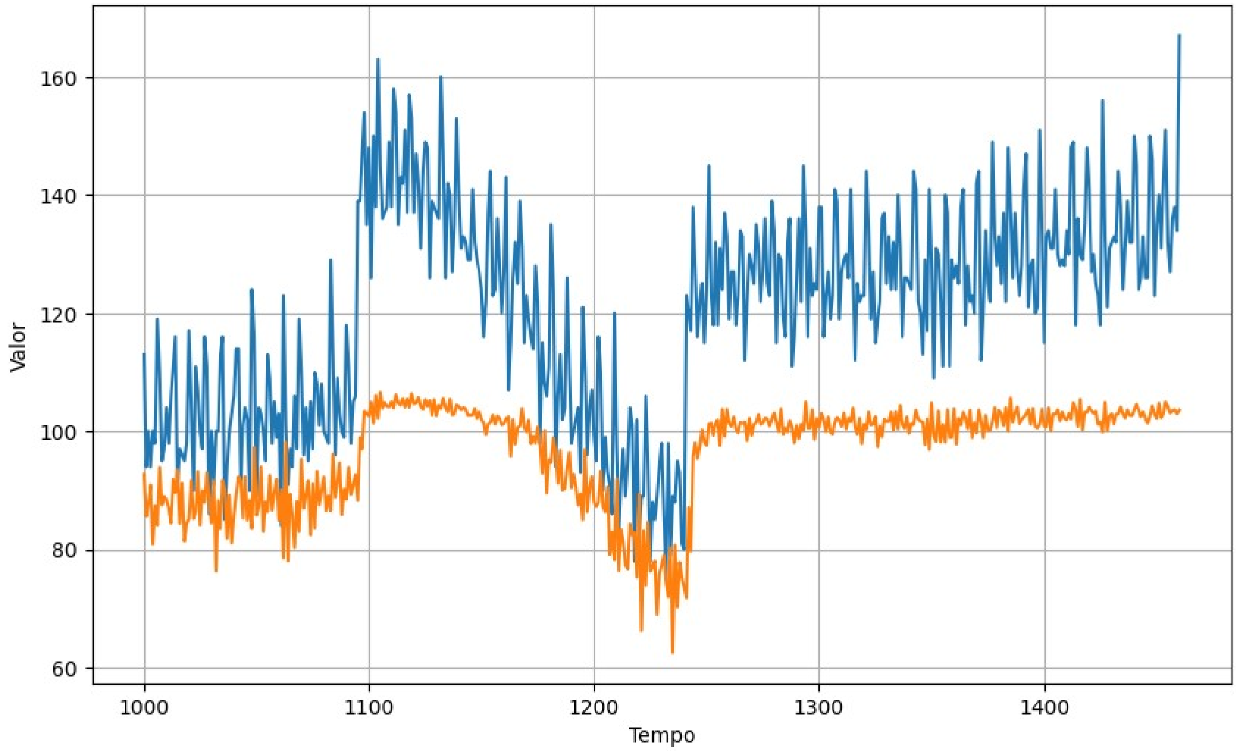
```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 78ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 133ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 74ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 172ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 109ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 102ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 152ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 112ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 139ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 94ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 49ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 54ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 51ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 97ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 51ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 51ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 50ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 108ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 108ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 126ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 85ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 309ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 82ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 83ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 104ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 83ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 92ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 97ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 90ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 52ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
```



Otimizar previsão

Criar dataset de treinamento

Inclui os mesmos passos realizados pela função `janelamento_lotes`, sem o embaralhamento de janelas.

```
# Serie de previsão
serie_forecast = serie[tam_trein - tam_janela:-1]

# Cria um dataset TF a partir dos valores da serie
dataset = tf.data.Dataset.from_tensor_slices(serie_forecast)

# Janelamento dos dados
dataset = dataset.window(tam_janela, shift=1, drop_remainder=True)

# Ajustar as janelas (flatten) colocando seus elementos em lotes
dataset = dataset.flat_map(lambda window: window.batch(tam_janela))
```

```
# Criar lotes de treinamento
dataset = dataset.batch(tam_lote).prefetch(1)
```

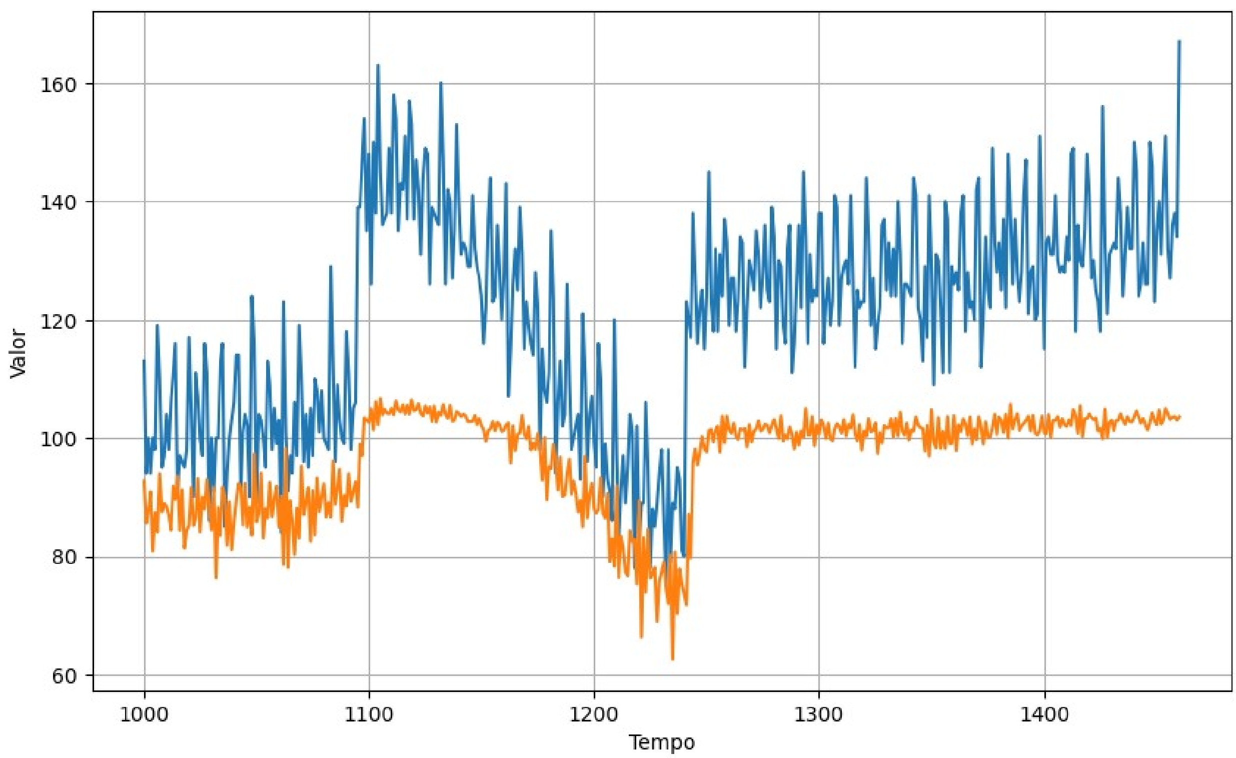
Previsão do conjunto de validação

Diferentemente do loop utilizado na previsão do modelo preliminar que processa uma janela por vez na previsão de cada lote, a função de previsão do modelo é invocada recebendo todo o dataset. A paralelização de lotes é realizada automaticamente pelo tensorflow.*

```
forecast = modelo_RNN.predict(dataset)

15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
```

Visualizar a previsão

```
# Ajustar formato
previsao = forecast.squeeze()

# Plotar resultados
plotar_series(tempo_valid, (x_valid, previsao))
```



Calcular MAPE

```
# Calcular MAPE
mape = tf.keras.metrics.MeanAbsolutePercentageError()
```

```
mape = tf.keras.metrics.mse =
tf.keras.metrics.MeanAbsolutePercentageError()
mape.update_state(x_valid, previsao)

MAPE = mape.result().numpy()/100
print(MAPE)

0.19200124740600585
```

TDE02 - Comparar desempenho de modelos

Utilizar a série sintética para comparar o modelo RNN com os modelo Baseline, AUTOARIMA, GB e RN.

- Calcular e imprimir méticas do modelo ST_RNN
- Salvar as métricas no arquivo Métricas_Previsão.csv onde estão salvas as métricas de todos os modelos estudados até agora
- Plotar a métrica MAPE para os modelos: ST_Baseline, ST_AUTOARIMA, STGB, ST_RN, ST_RNN

Para realizar os passos anteriores você pode consultar o código fornecido nas avaliações formativas. Não é necessário refazer os códigos de outros modelos porque nas avaliações formativas as métricas de desempenho foram salvas no arquivo Métricas_Previsão.csv.

Calcular e imprimir métricas do modelo ST_RNN (valor 0,4)

```
metricas_st_rnn = metricas(previsao, x_valid)
print("Métricas do modelo ST_RNN:")
for metrica, valor in metricas_st_rnn.items():
  print(f"{metrica}: {valor}")

metricas_df = pd.DataFrame({'Modelo': ['ST_RNN'],
                          'ME': [metricas_st_rnn['ME']],
                          'MSE': [metricas_st_rnn['MSE']],
                          'RMSE': [metricas_st_rnn['RMSE']],
                          'MAE': [metricas_st_rnn['MAE']],
                          'MPE': [metricas_st_rnn['MPE']],
                          'MAPE': [metricas_st_rnn['MAPE']],
                          'MIN-MAX': [metricas_st_rnn['MIN-MAX']]})

try:
  metricas_anteriores = pd.read_csv('Métricas_Previsão.csv')
  metricas_df = pd.concat([metricas_anteriores, metricas_df],
ignore_index=True)
except FileNotFoundError:
```

```
    pass

metricas_df.to_csv('Métricas_Previsão.csv', index=False)
```

```
Métricas do modelo ST_RNN:
ME: -24.150157928466797
MSE: 735.4100952148438
RMSE: 27.118446350097656
MAE: 24.289058685302734
MPE: -0.190353661775589
MAPE: 0.19200123846530914
MIN-MAX: 0.19195806980133057
```

Salvar méticas para comparação (valor 0,4)

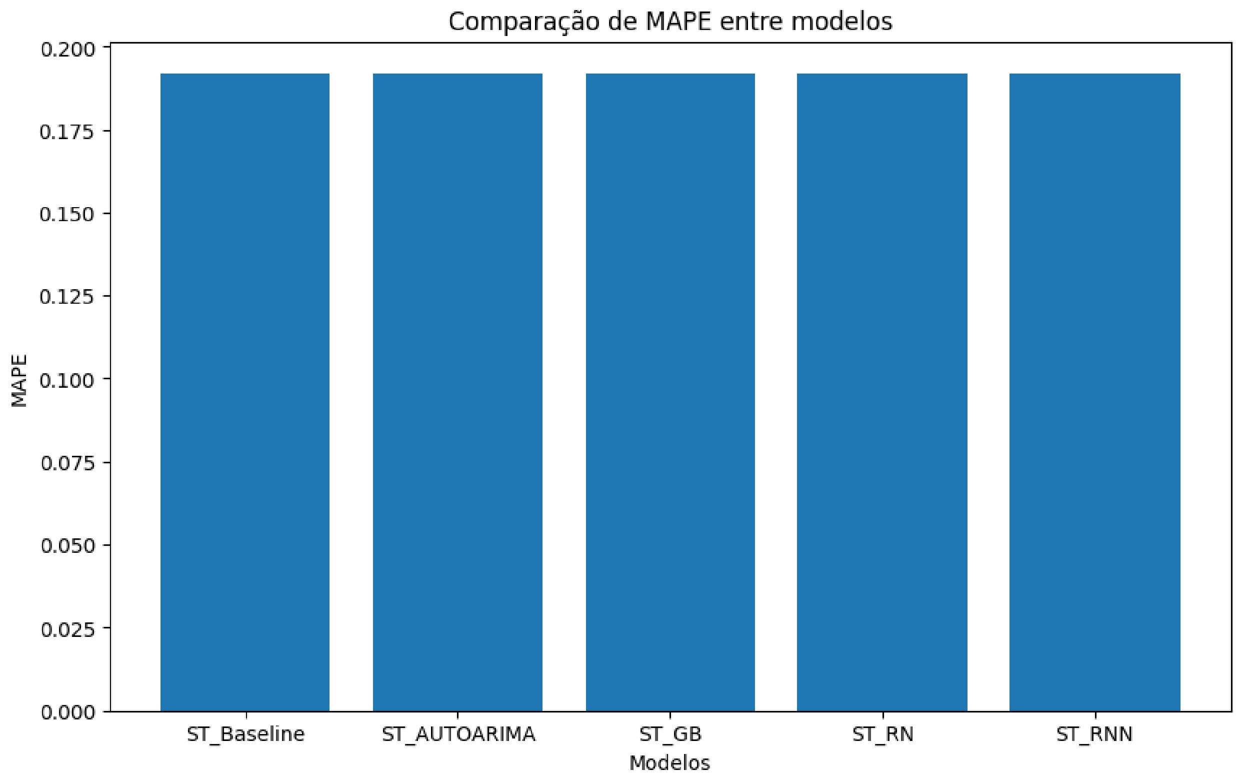
```
try:
  dfMetricas = pd.read_csv('Métricas_Previsão.csv')
except FileNotFoundError:
  dfMetricas = pd.DataFrame()

if 'ST_RNN' not in dfMetricas.columns:
  dfMetricas['ST_RNN'] = None

dfMetricas.to_csv('Métricas_Previsão.csv', index=False)
```

Plotar a mética MAPE (valor 0,4)

Modelos: ST_Baseline, ST_AUTOARIMA, ST_GB, ST_RN, ST_RNN

```
try:
  dfMetricas = pd.read_csv('Métricas_Previsão.csv')
except FileNotFoundError:
  print("Arquivo 'Métricas_Previsão.csv' não encontrado.")
  dfMetricas = pd.DataFrame()

if 'Modelo' in dfMetricas.columns and 'MAPE' in dfMetricas.columns:
  modelos = ['ST_Baseline', 'ST_AUTOARIMA', 'ST_GB', 'ST_RN',
'ST_RNN']
  mape_values = []
  for modelo in modelos:
    if modelo in dfMetricas['Modelo'].values:
        mape = dfMetricas.loc[dfMetricas['Modelo'] == modelo,
'MAPE'].values[0]
        mape_values.append(mape)
    else:
        print(f"Modelo {modelo} não encontrado no arquivo
'Métricas_Previsão.csv'.")
```

```
  plt.figure(figsize=(10, 6))
  plt.bar(modelos, mape_values)
  plt.xlabel('Modelos')
  plt.ylabel('MAPE')
  plt.title('Comparação de MAPE entre modelos')
  plt.show()
else:
  print("Colunas 'Modelo' ou 'MAPE' não encontradas no arquivo
'Métricas_Previsão.csv'.")

Modelo ST_Baseline não encontrado no arquivo 'Métricas_Previsão.csv'.
Modelo ST_AUTOARIMA não encontrado no arquivo 'Métricas_Previsão.csv'.
Modelo ST_GB não encontrado no arquivo 'Métricas_Previsão.csv'.
Modelo ST_RN não encontrado no arquivo 'Métricas_Previsão.csv'.
```



Conclusão (valor 0,8)

Colocar na célula abaixo a sua conclusão a respeito dos modelos de previsão de séries temporais estudados

- Baseline
- ARIMA

- Gradiete Boosting
- Redes Neurais
- Redes Neurais Recorrentes

Conclusão: ARIMA e Gradient Boosting mostraram bons resultados para capturar padrões estacionários e complexos, respectivamente. No entanto, dependem da preparação dos dados e da interpretação correta dos parâmetros. Redes Neurais e RNNs se destacam em capturar relações não lineares e dinâmicas, especialmente em séries com padrões complexos. Porém, podem ser mais desafiadores de treinar e otimizar, além de demandar maior poder computacional. A escolha do modelo ideal depende de fatores como a complexidade da série temporal, a disponibilidade de dados, a necessidade de interpretabilidade e os recursos computacionais disponíveis.

Considerando os resultados do presente trabalho, pode-se observar que o modelo RNN, apesar de demandar maior poder computacional, tende a ter um desempenho superior aos demais modelos, principalmente em séries temporais mais complexas, obtendo menor MAPE. Isso é devido à sua capacidade de capturar dependências ao longo do tempo e aprender representações complexas dos dados.

Recomenda-se considerar RNNs como uma boa opção para previsão de séries temporais em cenários complexos, porém, é crucial avaliar a complexidade do problema, requisitos de tempo de processamento e recursos computacionais disponíveis antes de implementar o modelo.