Ungraded Lab: Using a multi-layer LSTM for forecasting

In this lab, you will use the same RNN architecure in the first lab but will instead stack LSTM layers instead of `SimpleRNN` .

Imports

In [1]:
```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
```

Utilities

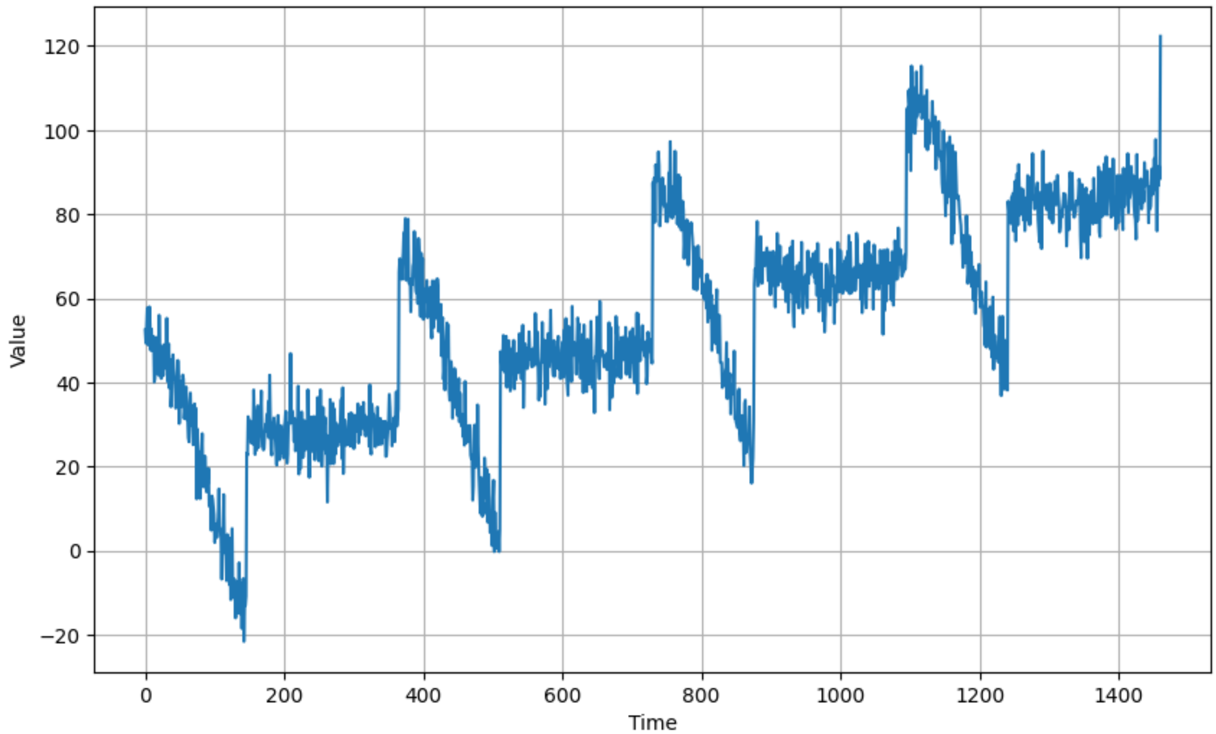
In [2]:
```
def plot_series(time, series, format="-", start=0, end=None):
    """
    Visualizes time series data

    Args:
      time (array of int) - contains the time steps
      series (array of int) - contains the measurements for each time step
      format - line style when plotting the graph
      start - first time step to plot
      end - last time step to plot
    """

    # Setup dimensions of the graph figure
    plt.figure(figsize=(10, 6))

    if type(series) is tuple:

      for series_num in series:
        # Plot the time series data
        plt.plot(time[start:end], series_num[start:end], format)

    else:
      # Plot the time series data
      plt.plot(time[start:end], series[start:end], format)

    # Label the x-axis
    plt.xlabel("Time")

    # Label the y-axis
    plt.ylabel("Value")

    # Overlay a grid on the graph
    plt.grid(True)

    # Draw the graph on screen
    plt.show()

def trend(time, slope=0):
    """
    Generates synthetic data that follows a straight line given a slope value.

    Args:
      time (array of int) - contains the time steps
      slope (float) - determines the direction and steepness of the line

    Returns:
      series (array of float) - measurements that follow a straight line
    """

    # Compute the linear series given the slope
    series = slope * time

    return series

def seasonal_pattern(season_time):
    """
    Just an arbitrary pattern, you can change it if you wish

    Args:
```

```
      season_time (array of float) - contains the measurements per time step

    Returns:
      data_pattern (array of float) -  contains revised measurement values according
                                  to the defined pattern
    """

    # Generate the values using an arbitrary pattern
    data_pattern = np.where(season_time < 0.4,
                    np.cos(season_time * 2 * np.pi),
                    1 / np.exp(3 * season_time))

    return data_pattern

def seasonality(time, period, amplitude=1, phase=0):
    """
    Repeats the same pattern at each period

    Args:
      time (array of int) - contains the time steps
      period (int) - number of time steps before the pattern repeats
      amplitude (int) - peak measured value in a period
      phase (int) - number of time steps to shift the measured values

    Returns:
      data_pattern (array of float) - seasonal data scaled by the defined amplitude
    """

    # Define the measured values per period
    season_time = ((time + phase) % period) / period

    # Generates the seasonal data scaled by the defined amplitude
    data_pattern = amplitude * seasonal_pattern(season_time)

    return data_pattern

def noise(time, noise_level=1, seed=None):
    """Generates a normally distributed noisy signal

    Args:
      time (array of int) - contains the time steps
      noise_level (float) - scaling factor for the generated signal
      seed (int) - number generator seed for repeatability

    Returns:
      noise (array of float) - the noisy signal
    """

    # Initialize the random number generator
    rnd = np.random.RandomState(seed)

    # Generate a random number for each time step and scale by the noise level
    noise = rnd.randn(len(time)) * noise_level

    return noise
```

Generate the Synthetic Data

In [3]:

```
# Parameters
time = np.arange(4 * 365 + 1, dtype="float32")
baseline = 10
amplitude = 40
slope = 0.05
noise_level = 5

# Create the series
series = baseline + trend(time, slope) + seasonality(time, period=365, amplitude=amplitude)

# Update with noise
series += noise(time, noise_level, seed=42)

# Plot the results
plot_series(time, series)
```



Split the Dataset

In [4]:
```
# Define the split time
split_time = 1000

# Get the train set
time_train = time[:split_time]
x_train = series[:split_time]

# Get the validation set
time_valid = time[split_time:]
x_valid = series[split_time:]
```

Prepare Features and Labels

In [5]:
```
# Parameters
window_size = 20
batch_size = 32
shuffle_buffer_size = 1000
```

In [6]:
```
def windowed_dataset(series, window_size, batch_size, shuffle_buffer):
    """Generates dataset windows

    Args:
      series (array of float) - contains the values of the time series
      window_size (int) - the number of time steps to include in the feature
      batch_size (int) - the batch size
      shuffle_buffer(int) - buffer size to use for the shuffle method

    Returns:
      dataset (TF Dataset) - TF Dataset containing time windows
    """

    # Add an axis for the feature dimension of RNN layers
    series = tf.expand_dims(series, axis=-1)

    # Generate a TF Dataset from the series values
    dataset = tf.data.Dataset.from_tensor_slices(series)

    # Window the data but only take those with the specified size
    dataset = dataset.window(window_size + 1, shift=1, drop_remainder=True)

    # Flatten the windows by putting its elements in a single batch
```

```
    dataset = dataset.flat_map(lambda window: window.batch(window_size + 1))

    # Create tuples with features and labels
    dataset = dataset.map(lambda window: (window[:-1], window[-1]))

    # Shuffle the windows
    dataset = dataset.shuffle(shuffle_buffer)

    # Create batches of windows
    dataset = dataset.batch(batch_size)

    # Optimize the dataset for training
    dataset = dataset.cache().prefetch(1)

    return dataset
```

In [7]:
```
# Generate the dataset windows
dataset = windowed_dataset(x_train, window_size, batch_size, shuffle_buffer_size)
```

Build the Model

As mentioned, you will swap `SimpleRNN` for `LSTM` in this lab. It is also set as bidirectional below but feel free to revise later and see what results you get. LSTMs are much more complex in their internal architecture than simpleRNNs. It implements a cell state that allows it to remember sequences better than simple implementations. This added complexity results in a bigger set of parameters to train and you'll see that when you print the model summary below.

In [8]:
```
# Build the Model
model_tune = tf.keras.models.Sequential([
    tf.keras.Input(shape=(window_size, 1)),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(32, return_sequences=True)),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(32)),
    tf.keras.layers.Dense(1),
    tf.keras.layers.Lambda(lambda x: x * 100.0)
])

# Print the model summary
model_tune.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
bidirectional (Bidirectional)	(None, 20, 64)	8,704
bidirectional_1 (Bidirectional)	(None, 64)	24,832
dense (Dense)	(None, 1)	65
lambda (Lambda)	(None, 1)	0

Total params: 33,601 (131.25 KB)
Trainable params: 33,601 (131.25 KB)
Non-trainable params: 0 (0.00 B)

Tune the Learning Rate

As usual, you will pick a learning rate by running the tuning code below.

In [9]:
```
# Set the learning rate scheduler
lr_schedule = tf.keras.callbacks.LearningRateScheduler(
    lambda epoch: 1e-8 * 10**(epoch / 20))

# Initialize the optimizer
optimizer = tf.keras.optimizers.SGD(momentum=0.9)

# Set the training parameters
model_tune.compile(loss=tf.keras.losses.Huber(), optimizer=optimizer)

# Train the model
history = model_tune.fit(dataset, epochs=100, callbacks=[lr_schedule])
```

```
Epoch 1/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 3s 6ms/step - loss: 25.9677 - learning_rate: 1.0000e-08
Epoch 2/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 25.8166 - learning_rate: 1.1220e-08
Epoch 3/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 25.6226 - learning_rate: 1.2589e-08
Epoch 4/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 25.4159 - learning_rate: 1.4125e-08
Epoch 5/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 25.1914 - learning_rate: 1.5849e-08
Epoch 6/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 24.9346 - learning_rate: 1.7783e-08
Epoch 7/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 24.6494 - learning_rate: 1.9953e-08
Epoch 8/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 24.3334 - learning_rate: 2.2387e-08
Epoch 9/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 23.9843 - learning_rate: 2.5119e-08
Epoch 10/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 23.6082 - learning_rate: 2.8184e-08
Epoch 11/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 23.2080 - learning_rate: 3.1623e-08
Epoch 12/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 22.7789 - learning_rate: 3.5481e-08
Epoch 13/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 22.3378 - learning_rate: 3.9811e-08
Epoch 14/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 21.8734 - learning_rate: 4.4668e-08
Epoch 15/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 21.4225 - learning_rate: 5.0119e-08
Epoch 16/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 20.9776 - learning_rate: 5.6234e-08
Epoch 17/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 20.5352 - learning_rate: 6.3096e-08
Epoch 18/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 20.0785 - learning_rate: 7.0795e-08
Epoch 19/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 19.5876 - learning_rate: 7.9433e-08
Epoch 20/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 19.0520 - learning_rate: 8.9125e-08
Epoch 21/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 18.4632 - learning_rate: 1.0000e-07
Epoch 22/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 17.8136 - learning_rate: 1.1220e-07
Epoch 23/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 17.0986 - learning_rate: 1.2589e-07
Epoch 24/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 16.3107 - learning_rate: 1.4125e-07
Epoch 25/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 15.4398 - learning_rate: 1.5849e-07
Epoch 26/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 14.4814 - learning_rate: 1.7783e-07
Epoch 27/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 13.4190 - learning_rate: 1.9953e-07
Epoch 28/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 12.2325 - learning_rate: 2.2387e-07
Epoch 29/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 10.9616 - learning_rate: 2.5119e-07
Epoch 30/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 9.7130 - learning_rate: 2.8184e-07
Epoch 31/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 8.6223 - learning_rate: 3.1623e-07
Epoch 32/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 7.7907 - learning_rate: 3.5481e-07
Epoch 33/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 7.2619 - learning_rate: 3.9811e-07
Epoch 34/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 6.9522 - learning_rate: 4.4668e-07
Epoch 35/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 6.7418 - learning_rate: 5.0119e-07
Epoch 36/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 6.5812 - learning_rate: 5.6234e-07
Epoch 37/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 6.4447 - learning_rate: 6.3096e-07
Epoch 38/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 6.3066 - learning_rate: 7.0795e-07
Epoch 39/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 6.1649 - learning_rate: 7.9433e-07
```

```
Epoch 40/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 6.0240 - learning_rate: 8.9125e-07
Epoch 41/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 5.8996 - learning_rate: 1.0000e-06
Epoch 42/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 5.7906 - learning_rate: 1.1220e-06
Epoch 43/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 5.6856 - learning_rate: 1.2589e-06
Epoch 44/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 5.5861 - learning_rate: 1.4125e-06
Epoch 45/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 5.4906 - learning_rate: 1.5849e-06
Epoch 46/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 5.3873 - learning_rate: 1.7783e-06
Epoch 47/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 5.2881 - learning_rate: 1.9953e-06
Epoch 48/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 5.2029 - learning_rate: 2.2387e-06
Epoch 49/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 5.1593 - learning_rate: 2.5119e-06
Epoch 50/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 5.1459 - learning_rate: 2.8184e-06
Epoch 51/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 5.1207 - learning_rate: 3.1623e-06
Epoch 52/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 5.0978 - learning_rate: 3.5481e-06
Epoch 53/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 5.1220 - learning_rate: 3.9811e-06
Epoch 54/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 5.1379 - learning_rate: 4.4668e-06
Epoch 55/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.9924 - learning_rate: 5.0119e-06
Epoch 56/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.7892 - learning_rate: 5.6234e-06
Epoch 57/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 4.7842 - learning_rate: 6.3096e-06
Epoch 58/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.6106 - learning_rate: 7.0795e-06
Epoch 59/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.6254 - learning_rate: 7.9433e-06
Epoch 60/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.7063 - learning_rate: 8.9125e-06
Epoch 61/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 4.6146 - learning_rate: 1.0000e-05
Epoch 62/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 4.5576 - learning_rate: 1.1220e-05
Epoch 63/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 4.6942 - learning_rate: 1.2589e-05
Epoch 64/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 4.7850 - learning_rate: 1.4125e-05
Epoch 65/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 4.6712 - learning_rate: 1.5849e-05
Epoch 66/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 4.8722 - learning_rate: 1.7783e-05
Epoch 67/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 5.0902 - learning_rate: 1.9953e-05
Epoch 68/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 5.1375 - learning_rate: 2.2387e-05
Epoch 69/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 5.3312 - learning_rate: 2.5119e-05
Epoch 70/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 5.0312 - learning_rate: 2.8184e-05
Epoch 71/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 5.6808 - learning_rate: 3.1623e-05
Epoch 72/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 5.7493 - learning_rate: 3.5481e-05
Epoch 73/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 5.5139 - learning_rate: 3.9811e-05
Epoch 74/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 5.6715 - learning_rate: 4.4668e-05
Epoch 75/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 6.2505 - learning_rate: 5.0119e-05
Epoch 76/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 6.0349 - learning_rate: 5.6234e-05
Epoch 77/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 6.6410 - learning_rate: 6.3096e-05
Epoch 78/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 9.0754 - learning_rate: 7.0795e-05
```

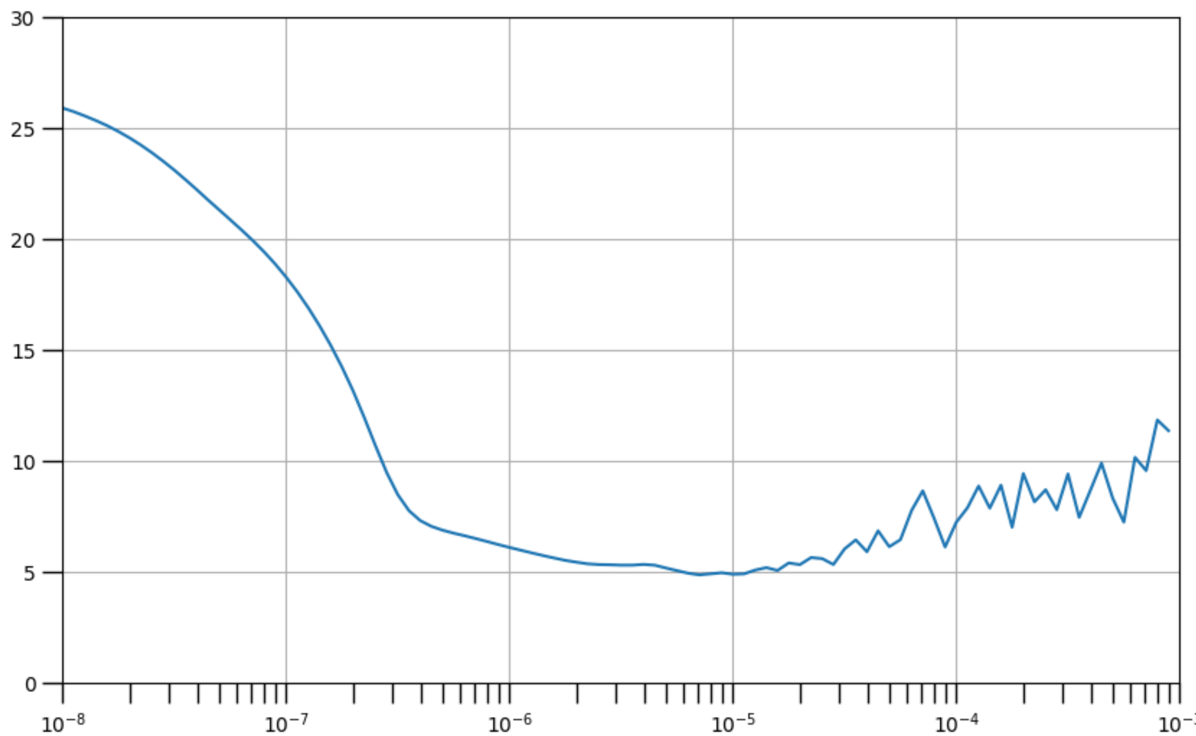
```
Epoch 79/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 7.4770 - learning_rate: 7.9433e-05
Epoch 80/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 5.3599 - learning_rate: 8.9125e-05
Epoch 81/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 7.5421 - learning_rate: 1.0000e-04
Epoch 82/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 7.5513 - learning_rate: 1.1220e-04
Epoch 83/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 9.0821 - learning_rate: 1.2589e-04
Epoch 84/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 6.9848 - learning_rate: 1.4125e-04
Epoch 85/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 7.7550 - learning_rate: 1.5849e-04
Epoch 86/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 6.2030 - learning_rate: 1.7783e-04
Epoch 87/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 7.5104 - learning_rate: 1.9953e-04
Epoch 88/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 7.6245 - learning_rate: 2.2387e-04
Epoch 89/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 7.9235 - learning_rate: 2.5119e-04
Epoch 90/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 6.4471 - learning_rate: 2.8184e-04
Epoch 91/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 8.5426 - learning_rate: 3.1623e-04
Epoch 92/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 6.5610 - learning_rate: 3.5481e-04
Epoch 93/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 8.1545 - learning_rate: 3.9811e-04
Epoch 94/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 11.3164 - learning_rate: 4.4668e-04
Epoch 95/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 6.8449 - learning_rate: 5.0119e-04
Epoch 96/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 8.0550 - learning_rate: 5.6234e-04
Epoch 97/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 10.4993 - learning_rate: 6.3096e-04
Epoch 98/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 9.3931 - learning_rate: 7.0795e-04
Epoch 99/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 10.5945 - learning_rate: 7.9433e-04
Epoch 100/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 10.4864 - learning_rate: 8.9125e-04
```

In [10]:

```python
# Define the learning rate array
lrs = 1e-8 * (10 ** (np.arange(100) / 20))

# Set the figure size
plt.figure(figsize=(10, 6))

# Set the grid
plt.grid(True)

# Plot the loss in log scale
plt.semilogx(lrs, history.history["loss"])

# Increase the tickmarks size
plt.tick_params('both', length=10, width=1, which='both')

# Set the plot boundaries
plt.axis([1e-8, 1e-3, 0, 30])
```

Out[10]:

```
(1e-08, 0.001, 0.0, 30.0)
```



Train the Model

You can then proceed to train the model with your chosen learning rate.

Tip: When experimenting and you find yourself running different iterations of a model, you may want to use the `clear_session()` *method to declutter memory used by Keras. This is added in the first line below.*

In [11]:
```
# Reset states generated by Keras
tf.keras.backend.clear_session()

# Build the model
model = tf.keras.models.Sequential([
    tf.keras.Input(shape=(window_size, 1)),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(32, return_sequences=True)),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(32)),
    tf.keras.layers.Dense(1),
    tf.keras.layers.Lambda(lambda x: x * 100.0)
])

# Set the learning rate
learning_rate = 2e-6

# Set the optimizer
optimizer = tf.keras.optimizers.SGD(learning_rate=learning_rate, momentum=0.9)

# Set the training parameters
model.compile(loss=tf.keras.losses.Huber(),
              optimizer=optimizer,
              metrics=["mae"])

# Train the model
history = model.fit(dataset,epochs=100)
```

```
Epoch 1/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 2s 6ms/step - loss: 66.4583 - mae: 66.9578
Epoch 2/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 13.0632 - mae: 13.5602
Epoch 3/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 9.1585 - mae: 9.6362
Epoch 4/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 8.4469 - mae: 8.9343
Epoch 5/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 8.0986 - mae: 8.5859
Epoch 6/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 7.7979 - mae: 8.2862
Epoch 7/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 7.4471 - mae: 7.9360
Epoch 8/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 6.9718 - mae: 7.4576
Epoch 9/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 6.6138 - mae: 7.0946
Epoch 10/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 6.4142 - mae: 6.8958
Epoch 11/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 6.2424 - mae: 6.7223
Epoch 12/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 6.1135 - mae: 6.5937
Epoch 13/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 6.0016 - mae: 6.4827
Epoch 14/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 5.8965 - mae: 6.3775
Epoch 15/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 5.8003 - mae: 6.2805
Epoch 16/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 5.7236 - mae: 6.2035
Epoch 17/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 5.6583 - mae: 6.1391
Epoch 18/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 5.5891 - mae: 6.0658
Epoch 19/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 5.5209 - mae: 6.0012
Epoch 20/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 5.4769 - mae: 5.9569
Epoch 21/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 5.4207 - mae: 5.9000
Epoch 22/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 5.4302 - mae: 5.9100
Epoch 23/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 5.4410 - mae: 5.9174
Epoch 24/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 5.3457 - mae: 5.8226
Epoch 25/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 5.2259 - mae: 5.7030
Epoch 26/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 5.2093 - mae: 5.6842
Epoch 27/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 5.1578 - mae: 5.6312
Epoch 28/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 5.0689 - mae: 5.5448
Epoch 29/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 5.0629 - mae: 5.5380
Epoch 30/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 5.0776 - mae: 5.5513
Epoch 31/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 5.0193 - mae: 5.4938
Epoch 32/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 4.9592 - mae: 5.4346
Epoch 33/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 4.9672 - mae: 5.4412
Epoch 34/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 4.9442 - mae: 5.4185
Epoch 35/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 4.8918 - mae: 5.3673
Epoch 36/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 4.9072 - mae: 5.3808
Epoch 37/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 4.8872 - mae: 5.3614
Epoch 38/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 4.8553 - mae: 5.3288
Epoch 39/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.8453 - mae: 5.3191
```

```
Epoch 40/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.8168 - mae: 5.2904
Epoch 41/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.8075 - mae: 5.2803
Epoch 42/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.7873 - mae: 5.2610
Epoch 43/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.7738 - mae: 5.2472
Epoch 44/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.7603 - mae: 5.2334
Epoch 45/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.7484 - mae: 5.2206
Epoch 46/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.7369 - mae: 5.2088
Epoch 47/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.7270 - mae: 5.1978
Epoch 48/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.7168 - mae: 5.1871
Epoch 49/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.7077 - mae: 5.1774
Epoch 50/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.6993 - mae: 5.1686
Epoch 51/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.6913 - mae: 5.1611
Epoch 52/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.6828 - mae: 5.1531
Epoch 53/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.6748 - mae: 5.1453
Epoch 54/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.6677 - mae: 5.1388
Epoch 55/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.6599 - mae: 5.1317
Epoch 56/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.6535 - mae: 5.1257
Epoch 57/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.6466 - mae: 5.1190
Epoch 58/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.6404 - mae: 5.1131
Epoch 59/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.6342 - mae: 5.1069
Epoch 60/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.6285 - mae: 5.1013
Epoch 61/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.6224 - mae: 5.0951
Epoch 62/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.6165 - mae: 5.0889
Epoch 63/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.6109 - mae: 5.0833
Epoch 64/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.6044 - mae: 5.0767
Epoch 65/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.5984 - mae: 5.0706
Epoch 66/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.5917 - mae: 5.0639
Epoch 67/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.5849 - mae: 5.0572
Epoch 68/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.5802 - mae: 5.0519
Epoch 69/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.5744 - mae: 5.0462
Epoch 70/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.5687 - mae: 5.0402
Epoch 71/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.5623 - mae: 5.0338
Epoch 72/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.5565 - mae: 5.0275
Epoch 73/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.5502 - mae: 5.0212
Epoch 74/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.5441 - mae: 5.0149
Epoch 75/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.5389 - mae: 5.0093
Epoch 76/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.5349 - mae: 5.0053
Epoch 77/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.5279 - mae: 4.9984
Epoch 78/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.5234 - mae: 4.9940
```

```
Epoch 79/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.5190 - mae: 4.9898
Epoch 80/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.5141 - mae: 4.9848
Epoch 81/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.5092 - mae: 4.9799
Epoch 82/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.5056 - mae: 4.9766
Epoch 83/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.5009 - mae: 4.9720
Epoch 84/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.4963 - mae: 4.9675
Epoch 85/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.4951 - mae: 4.9660
Epoch 86/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.4903 - mae: 4.9613
Epoch 87/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.4872 - mae: 4.9578
Epoch 88/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.4835 - mae: 4.9538
Epoch 89/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.4831 - mae: 4.9543
Epoch 90/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.4776 - mae: 4.9493
Epoch 91/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.4737 - mae: 4.9466
Epoch 92/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.4731 - mae: 4.9465
Epoch 93/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.4712 - mae: 4.9454
Epoch 94/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 4.4694 - mae: 4.9441
Epoch 95/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.4654 - mae: 4.9402
Epoch 96/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 4.4672 - mae: 4.9421
Epoch 97/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.4658 - mae: 4.9409
Epoch 98/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.4634 - mae: 4.9385
Epoch 99/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.4611 - mae: 4.9362
Epoch 100/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.4599 - mae: 4.9349
```

Model Prediction

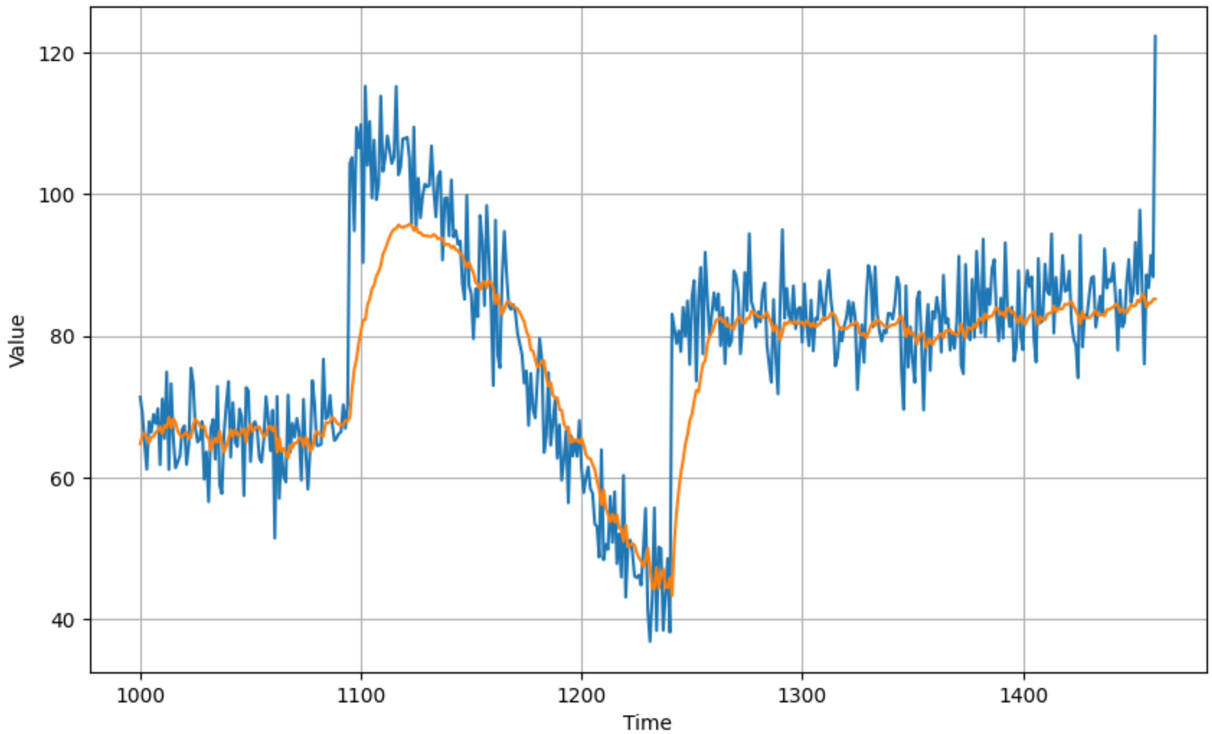
You will then generate batches of windows to generate predictions that align with the validation set.

In [12]:

```python
def model_forecast(model, series, window_size, batch_size):
    """Uses an input model to generate predictions on data windows

    Args:
      model (TF Keras Model) - model that accepts data windows
      series (array of float) - contains the values of the time series
      window_size (int) - the number of time steps to include in the window
      batch_size (int) - the batch size

    Returns:
      forecast (numpy array) - array containing predictions
    """

    # Add an axis for the feature dimension of RNN layers
    series = tf.expand_dims(series, axis=-1)

    # Generate a TF Dataset from the series values
    dataset = tf.data.Dataset.from_tensor_slices(series)

    # Window the data but only take those with the specified size
    dataset = dataset.window(window_size, shift=1, drop_remainder=True)

    # Flatten the windows by putting its elements in a single batch
    dataset = dataset.flat_map(lambda w: w.batch(window_size))

    # Create batches of windows
    dataset = dataset.batch(batch_size).prefetch(1)
```

```python
    # Get predictions on the entire dataset
    forecast = model.predict(dataset, verbose=0)

    return forecast
```

In [13]:
```python
# Reduce the original series
forecast_series = series[split_time-window_size:-1]

# Use helper function to generate predictions
forecast = model_forecast(model, forecast_series, window_size, batch_size)

# Drop single dimensional axis
results = forecast.squeeze()

# Plot the results
plot_series(time_valid, (x_valid, results))
```



You can then generate the metrics to evaluate the model's performance.

In [14]:
```python
# Compute the MSE and MAE
print(tf.keras.metrics.mse(x_valid, results).numpy())
print(tf.keras.metrics.mae(x_valid, results).numpy())
```

```
67.75228
5.7619185
```

Wrap Up

This concludes this short exercise on using LSTMs for time series forecasting. Next week, you will build upon this and add convolutions. Then, you will start to move away from synthetic data and use real-world datasets. See you there!

Optional: Including a Validation Set while Training

Back in the first course of this specialization, you saw how you can also monitor the performance of your model against a validation set while training. You can also do that for this lab.

First, you need to generate a `val_set` which are data windows and labels that your model can accept. You can simply reuse the `windowed_dataset` function for that and you can pass in the `x_valid` points to generate the windows.

In [15]:
```
# Generate data windows of the validation set
val_set = windowed_dataset(x_valid, window_size, batch_size, shuffle_buffer_size)
```

You can then do the same training as before but pass in the `val_set` to the `validation_data` parameter of the `fit()` method.

In [16]:
```
# Reset states generated by Keras
tf.keras.backend.clear_session()

# Build the model
model = tf.keras.models.Sequential([
    tf.keras.Input(shape=(window_size, 1)),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(32, return_sequences=True)),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(32)),
    tf.keras.layers.Dense(1),
    tf.keras.layers.Lambda(lambda x: x * 100.0)
])

# Set the learning rate
learning_rate = 2e-6

# Set the optimizer
optimizer = tf.keras.optimizers.SGD(learning_rate=learning_rate, momentum=0.9)

# Set the training parameters
model.compile(loss=tf.keras.losses.Huber(),
              optimizer=optimizer,
              metrics=["mae"])

# Train the model
history = model.fit(dataset,epochs=100, validation_data=val_set)
```

```
Epoch 1/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 2s 20ms/step - loss: 32.7407 - mae: 33.2375 - val_loss: 22.3182 - val_mae: 22.8143
Epoch 2/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 12.1673 - mae: 12.6610 - val_loss: 17.5748 - val_mae: 18.0723
Epoch 3/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 7.5763 - mae: 8.0522 - val_loss: 12.2577 - val_mae: 12.7495
Epoch 4/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 6.6635 - mae: 7.1468 - val_loss: 10.9568 - val_mae: 11.4456
Epoch 5/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 6.4514 - mae: 6.9334 - val_loss: 10.5222 - val_mae: 11.0129
Epoch 6/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 6.2836 - mae: 6.7668 - val_loss: 10.1590 - val_mae: 10.6476
Epoch 7/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 6.1421 - mae: 6.6274 - val_loss: 9.8933 - val_mae: 10.3809
Epoch 8/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 6.0093 - mae: 6.4942 - val_loss: 9.6813 - val_mae: 10.1698
Epoch 9/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 5.8863 - mae: 6.3704 - val_loss: 9.4973 - val_mae: 9.9870
Epoch 10/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 5.7738 - mae: 6.2579 - val_loss: 9.2777 - val_mae: 9.7662
Epoch 11/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 5.6705 - mae: 6.1548 - val_loss: 9.0967 - val_mae: 9.5842
Epoch 12/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 5.5715 - mae: 6.0554 - val_loss: 8.9124 - val_mae: 9.3985
Epoch 13/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 5.4816 - mae: 5.9662 - val_loss: 8.7294 - val_mae: 9.2144
Epoch 14/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 5.4034 - mae: 5.8877 - val_loss: 8.5282 - val_mae: 9.0127
Epoch 15/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 5.3326 - mae: 5.8144 - val_loss: 8.3353 - val_mae: 8.8217
Epoch 16/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 5.2656 - mae: 5.7466 - val_loss: 8.1559 - val_mae: 8.6416
Epoch 17/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 5.2053 - mae: 5.6837 - val_loss: 7.9849 - val_mae: 8.4695
Epoch 18/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 5.1496 - mae: 5.6280 - val_loss: 7.8284 - val_mae: 8.3121
Epoch 19/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 5.0995 - mae: 5.5794 - val_loss: 7.6667 - val_mae: 8.1507
Epoch 20/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 5.0525 - mae: 5.5340 - val_loss: 7.5235 - val_mae: 8.0081
Epoch 21/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 5.0088 - mae: 5.4902 - val_loss: 7.3718 - val_mae: 7.8558
Epoch 22/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.9697 - mae: 5.4509 - val_loss: 7.2433 - val_mae: 7.7270
Epoch 23/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.9356 - mae: 5.4172 - val_loss: 7.1243 - val_mae: 7.6078
Epoch 24/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.9049 - mae: 5.3861 - val_loss: 6.9928 - val_mae: 7.4759
Epoch 25/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.8762 - mae: 5.3568 - val_loss: 6.8856 - val_mae: 7.3666
Epoch 26/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.8490 - mae: 5.3294 - val_loss: 6.7913 - val_mae: 7.2710
Epoch 27/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.8234 - mae: 5.3043 - val_loss: 6.6893 - val_mae: 7.1695
Epoch 28/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.8007 - mae: 5.2826 - val_loss: 6.6051 - val_mae: 7.0867
Epoch 29/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.7772 - mae: 5.2603 - val_loss: 6.5213 - val_mae: 7.0046
Epoch 30/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.7569 - mae: 5.2409 - val_loss: 6.4488 - val_mae: 6.9322
Epoch 31/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.7361 - mae: 5.2207 - val_loss: 6.3685 - val_mae: 6.8514
Epoch 32/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.7185 - mae: 5.2031 - val_loss: 6.2921 - val_mae: 6.7734
Epoch 33/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.7022 - mae: 5.1866 - val_loss: 6.2262 - val_mae: 6.7063
Epoch 34/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.6849 - mae: 5.1693 - val_loss: 6.1569 - val_mae: 6.6366
Epoch 35/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.6700 - mae: 5.1546 - val_loss: 6.0915 - val_mae: 6.5731
Epoch 36/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.6556 - mae: 5.1402 - val_loss: 6.0228 - val_mae: 6.5063
Epoch 37/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.6436 - mae: 5.1277 - val_loss: 5.9600 - val_mae: 6.4455
Epoch 38/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.6337 - mae: 5.1170 - val_loss: 5.8997 - val_mae: 6.3855
Epoch 39/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.6229 - mae: 5.1052 - val_loss: 5.8415 - val_mae: 6.3266
```

```
Epoch 40/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.6148 - mae: 5.0966 - val_loss: 5.7922 - val_mae: 6.2759
Epoch 41/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.6074 - mae: 5.0898 - val_loss: 5.7488 - val_mae: 6.2310
Epoch 42/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.6022 - mae: 5.0851 - val_loss: 5.7136 - val_mae: 6.1948
Epoch 43/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.5977 - mae: 5.0801 - val_loss: 5.6800 - val_mae: 6.1602
Epoch 44/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.5970 - mae: 5.0785 - val_loss: 5.6620 - val_mae: 6.1420
Epoch 45/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.5955 - mae: 5.0765 - val_loss: 5.6343 - val_mae: 6.1135
Epoch 46/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.5952 - mae: 5.0757 - val_loss: 5.6149 - val_mae: 6.0935
Epoch 47/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.5916 - mae: 5.0716 - val_loss: 5.5969 - val_mae: 6.0750
Epoch 48/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.5879 - mae: 5.0671 - val_loss: 5.5776 - val_mae: 6.0554
Epoch 49/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.5851 - mae: 5.0639 - val_loss: 5.5592 - val_mae: 6.0370
Epoch 50/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.5838 - mae: 5.0622 - val_loss: 5.5416 - val_mae: 6.0193
Epoch 51/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.5824 - mae: 5.0601 - val_loss: 5.5252 - val_mae: 6.0026
Epoch 52/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.5796 - mae: 5.0571 - val_loss: 5.5108 - val_mae: 5.9879
Epoch 53/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.5776 - mae: 5.0546 - val_loss: 5.4933 - val_mae: 5.9698
Epoch 54/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.5752 - mae: 5.0518 - val_loss: 5.4780 - val_mae: 5.9542
Epoch 55/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.5717 - mae: 5.0479 - val_loss: 5.4667 - val_mae: 5.9428
Epoch 56/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.5678 - mae: 5.0438 - val_loss: 5.4545 - val_mae: 5.9305
Epoch 57/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.5671 - mae: 5.0427 - val_loss: 5.4443 - val_mae: 5.9202
Epoch 58/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.5653 - mae: 5.0407 - val_loss: 5.4292 - val_mae: 5.9049
Epoch 59/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.5665 - mae: 5.0415 - val_loss: 5.4206 - val_mae: 5.8963
Epoch 60/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.5647 - mae: 5.0396 - val_loss: 5.4108 - val_mae: 5.8864
Epoch 61/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.5623 - mae: 5.0372 - val_loss: 5.4002 - val_mae: 5.8757
Epoch 62/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.5585 - mae: 5.0334 - val_loss: 5.3895 - val_mae: 5.8649
Epoch 63/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.5558 - mae: 5.0307 - val_loss: 5.3799 - val_mae: 5.8550
Epoch 64/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.5536 - mae: 5.0282 - val_loss: 5.3691 - val_mae: 5.8438
Epoch 65/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.5507 - mae: 5.0254 - val_loss: 5.3596 - val_mae: 5.8338
Epoch 66/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.5497 - mae: 5.0244 - val_loss: 5.3521 - val_mae: 5.8263
Epoch 67/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.5500 - mae: 5.0247 - val_loss: 5.3417 - val_mae: 5.8159
Epoch 68/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.5482 - mae: 5.0228 - val_loss: 5.3317 - val_mae: 5.8060
Epoch 69/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.5449 - mae: 5.0193 - val_loss: 5.3245 - val_mae: 5.7986
Epoch 70/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.5414 - mae: 5.0157 - val_loss: 5.3187 - val_mae: 5.7926
Epoch 71/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.5403 - mae: 5.0148 - val_loss: 5.3119 - val_mae: 5.7857
Epoch 72/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.5403 - mae: 5.0150 - val_loss: 5.3008 - val_mae: 5.7747
Epoch 73/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.5405 - mae: 5.0151 - val_loss: 5.2955 - val_mae: 5.7692
Epoch 74/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.5391 - mae: 5.0138 - val_loss: 5.2897 - val_mae: 5.7633
Epoch 75/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.5378 - mae: 5.0126 - val_loss: 5.2818 - val_mae: 5.7551
Epoch 76/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.5361 - mae: 5.0108 - val_loss: 5.2743 - val_mae: 5.7472
Epoch 77/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.5325 - mae: 5.0071 - val_loss: 5.2668 - val_mae: 5.7394
Epoch 78/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.5297 - mae: 5.0041 - val_loss: 5.2608 - val_mae: 5.7333
```

```
Epoch 79/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.5271 - mae: 5.0013 - val_loss: 5.2526 - val_mae: 5.7249
Epoch 80/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.5252 - mae: 4.9993 - val_loss: 5.2458 - val_mae: 5.7178
Epoch 81/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.5213 - mae: 4.9952 - val_loss: 5.2408 - val_mae: 5.7125
Epoch 82/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.5189 - mae: 4.9929 - val_loss: 5.2351 - val_mae: 5.7066
Epoch 83/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 4.5177 - mae: 4.9917 - val_loss: 5.2313 - val_mae: 5.7027
Epoch 84/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.5144 - mae: 4.9884 - val_loss: 5.2235 - val_mae: 5.6947
Epoch 85/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.5118 - mae: 4.9855 - val_loss: 5.2161 - val_mae: 5.6871
Epoch 86/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.5109 - mae: 4.9846 - val_loss: 5.2133 - val_mae: 5.6841
Epoch 87/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.5102 - mae: 4.9843 - val_loss: 5.2095 - val_mae: 5.6802
Epoch 88/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.5089 - mae: 4.9830 - val_loss: 5.2033 - val_mae: 5.6738
Epoch 89/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.5069 - mae: 4.9810 - val_loss: 5.1995 - val_mae: 5.6698
Epoch 90/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.5044 - mae: 4.9786 - val_loss: 5.1953 - val_mae: 5.6656
Epoch 91/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.5029 - mae: 4.9772 - val_loss: 5.1908 - val_mae: 5.6610
Epoch 92/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.5031 - mae: 4.9775 - val_loss: 5.1859 - val_mae: 5.6562
Epoch 93/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.5035 - mae: 4.9780 - val_loss: 5.1819 - val_mae: 5.6522
Epoch 94/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.5026 - mae: 4.9771 - val_loss: 5.1791 - val_mae: 5.6493
Epoch 95/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.4996 - mae: 4.9739 - val_loss: 5.1737 - val_mae: 5.6438
Epoch 96/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.4980 - mae: 4.9723 - val_loss: 5.1715 - val_mae: 5.6416
Epoch 97/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.4938 - mae: 4.9681 - val_loss: 5.1681 - val_mae: 5.6381
Epoch 98/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.4916 - mae: 4.9658 - val_loss: 5.1627 - val_mae: 5.6326
Epoch 99/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.4912 - mae: 4.9653 - val_loss: 5.1595 - val_mae: 5.6293
Epoch 100/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 4.4899 - mae: 4.9640 - val_loss: 5.1534 - val_mae: 5.6234
```