```
import os
import datetime

import IPython
import IPython.display
import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import tensorflow as tf
```

```
zip_path = tf.keras.utils.get_file(
    origin='https://storage.googleapis.com/tensorflow/tf-keras-datasets/jena_climate_2009_2016.csv.zip',
    fname='jena_climate_2009_2016.csv.zip',
    extract=True)
csv_path, _ = os.path.splitext(zip_path)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/jena_climate_2009_2016.csv.zip
13574144/13568290 [==============================] - 0s 0us/step
13582336/13568290 [==============================] - 0s 0us/step
```

```
df = pd.read_csv(csv_path)
df.head(10)
```

1 to 10 of 10 entries Filter

index	Date Time	p (mbar)	T (degC)	Tpot (K)	Tdew (degC)	rh (%)	VPmax (mbar)	VPact (mbar)	VPdef (mbar
0	01.01.2009 00:10:00	996.52	-8.02	265.4	-8.9	93.3	3.33	3.11	0.22
1	01.01.2009 00:20:00	996.57	-8.41	265.01	-9.28	93.4	3.23	3.02	0.21
2	01.01.2009 00:30:00	996.53	-8.51	264.91	-9.31	93.9	3.21	3.01	0.2
3	01.01.2009 00:40:00	996.51	-8.31	265.12	-9.07	94.2	3.26	3.07	0.19
4	01.01.2009 00:50:00	996.51	-8.27	265.15	-9.04	94.1	3.27	3.08	0.19
5	01.01.2009 01:00:00	996.5	-8.05	265.38	-8.78	94.4	3.33	3.14	0.19
6	01.01.2009 01:10:00	996.5	-7.62	265.81	-8.3	94.8	3.44	3.26	0.18
7	01.01.2009 01:20:00	996.5	-7.62	265.81	-8.36	94.4	3.44	3.25	0.19
8	01.01.2009 01:30:00	996.5	-7.91	265.52	-8.73	93.8	3.36	3.15	0.21
9	01.01.2009 01:40:00	996.53	-8.43	264.99	-9.34	93.1	3.23	3.0	0.22

Show 25 per page

Like what you see? Visit the data table notebook to learn more about interactive tables

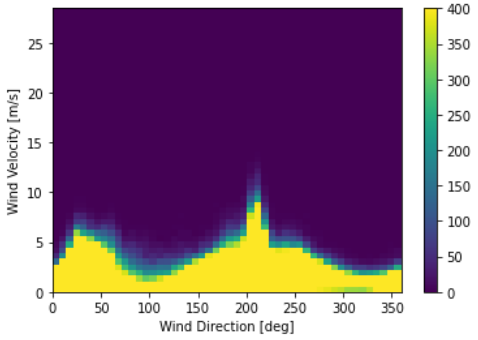
HANDLING ERRENEOUS DATA

```
wv = df['wv (m/s)']
bad_wv = wv == -9999.0
wv[bad_wv] = 0.0

max_wv = df['max. wv (m/s)']
bad_max_wv = max_wv == -9999.0
max_wv[bad_max_wv] = 0.0

# The above inplace edits are reflected in the DataFrame.
df['wv (m/s)'].mean()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  This is separate from the ipykernel package so we can avoid doing imports until
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  import sys
2.130190963759447
```

```
plt.hist2d(df['wd (deg)'], df['wv (m/s)'], bins=(50, 50), vmax=400)
plt.colorbar()
plt.xlabel('Wind Direction [deg]')
plt.ylabel('Wind Velocity [m/s]')
```

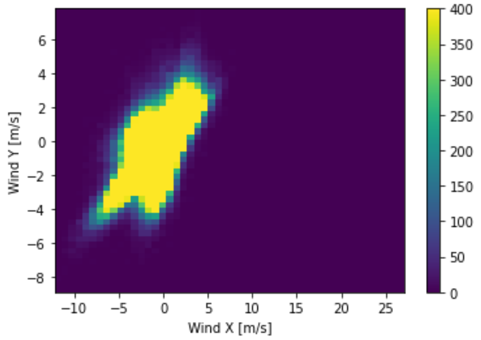
```
Text(0, 0.5, 'Wind Velocity [m/s]')
```



```
wv = df.pop('wv (m/s)')
max_wv = df.pop('max. wv (m/s)')

# Convert to radians.
wd_rad = df.pop('wd (deg)')*np.pi / 180

# Calculate the wind x and y components.
df['Wx'] = wv*np.cos(wd_rad)
df['Wy'] = wv*np.sin(wd_rad)

# Calculate the max wind x and y components.
df['max Wx'] = max_wv*np.cos(wd_rad)
df['max Wy'] = max_wv*np.sin(wd_rad)
```

```
plt.hist2d(df['Wx'], df['Wy'], bins=(50, 50), vmax=400)
plt.colorbar()
plt.xlabel('Wind X [m/s]')
plt.ylabel('Wind Y [m/s]')
ax = plt.gca()
ax.axis('tight')
```

```
(-12.185637751588763,
 27.064703747937347,
 -8.898421828413506,
 7.849152333233395)
```



```
date_time = pd.to_datetime(df.pop('Date Time'), format='%d.%m.%Y %H:%M:%S')
dt = date_time.tolist()
print(dt[0:10])
print(type(dt[0]))
```

```
[Timestamp('2009-01-01 00:10:00'), Timestamp('2009-01-01 00:20:00'), Timestamp('2009-01-01 00:30:00'), Timestamp('2009-01-01 00:40:00'), Ti
<class 'pandas._libs.tslibs.timestamps.Timestamp'>
```

```
timestamp_s = date_time.map(pd.Timestamp.timestamp)
timestamp_s
```

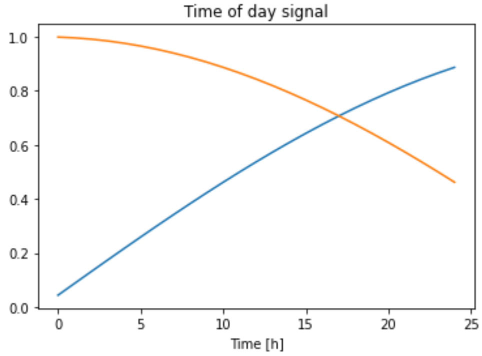
```
0         1.230769e+09
1         1.230769e+09
2         1.230770e+09
3         1.230770e+09
```

```
4         1.230771e+09
              ...
420546    1.483226e+09
420547    1.483227e+09
420548    1.483228e+09
420549    1.483228e+09
420550    1.483229e+09
Name: Date Time, Length: 420551, dtype: float64
```

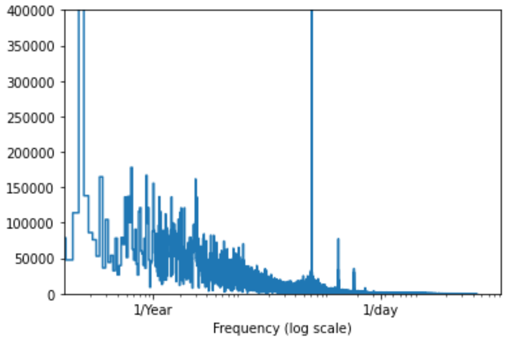
```
day = 24*60*60
year = (365.2425)*day

df['Day sin'] = np.sin(timestamp_s * (2 * np.pi / day))
df['Day cos'] = np.cos(timestamp_s * (2 * np.pi / day))
df['Year sin'] = np.sin(timestamp_s * (2 * np.pi / year))
df['Year cos'] = np.cos(timestamp_s * (2 * np.pi / year))
```

```
plt.plot(np.array(df['Day sin'])[:25])
plt.plot(np.array(df['Day cos'])[:25])
plt.xlabel('Time [h]')
plt.title('Time of day signal')
```

```
Text(0.5, 1.0, 'Time of day signal')
```



```
fft = tf.signal.rfft(df['T (degC)'])
f_per_dataset = np.arange(0, len(fft))

n_samples_h = len(df['T (degC)'])
hours_per_year = 24*365.2524
years_per_dataset = n_samples_h/(hours_per_year)

f_per_year = f_per_dataset/years_per_dataset
plt.step(f_per_year, np.abs(fft))
plt.xscale('log')
plt.ylim(0, 400000)
plt.xlim([0.1, max(plt.xlim())])
plt.xticks([1, 365.2524], labels=['1/Year', '1/day'])
_ = plt.xlabel('Frequency (log scale)')
```

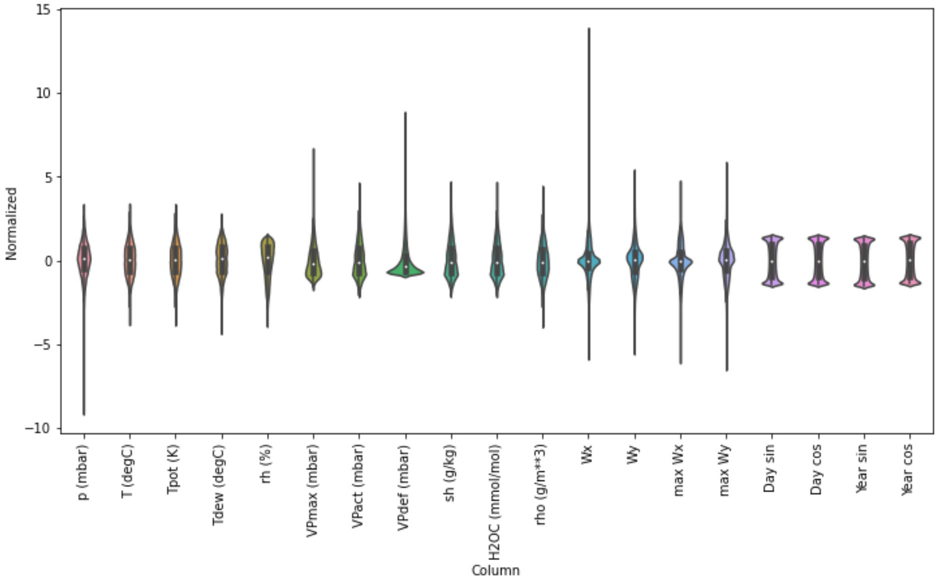


```
column_indices = {name: i for i, name in enumerate(df.columns)}

n = len(df)
train_df = df[0:int(n*0.7)]
val_df = df[int(n*0.7):int(n*0.9)]
test_df = df[int(n*0.9):]

num_features = df.shape[1]
```

NORMALIZE DATA

```
train_mean = train_df.mean()
train_std = train_df.std()

train_df = (train_df - train_mean) / train_std
val_df = (val_df - train_mean) / train_std
test_df = (test_df - train_mean) / train_std
```

```
df_std = (df - train_mean) / train_std
df_std = df_std.melt(var_name='Column', value_name='Normalized')
plt.figure(figsize=(12, 6))
ax = sns.violinplot(x='Column', y='Normalized', data=df_std)
_ = ax.set_xticklabels(df.keys(), rotation=90)
```



```
class WindowGenerator():
  def __init__(self, input_width, label_width, shift,
               train_df=train_df, val_df=val_df, test_df=test_df,
               label_columns=None):
    # Store the raw data.
    self.train_df = train_df
    self.val_df = val_df
    self.test_df = test_df

    # Work out the label column indices.
    self.label_columns = label_columns
    if label_columns is not None:
      self.label_columns_indices = {name: i for i, name in
                                    enumerate(label_columns)}
    self.column_indices = {name: i for i, name in
                           enumerate(train_df.columns)}


    # Work out the window parameters.
    self.input_width = input_width
    self.label_width = label_width
    self.shift = shift

    self.total_window_size = input_width + shift

    self.input_slice = slice(0, input_width)
    self.input_indices = np.arange(self.total_window_size)[self.input_slice]

    self.label_start = self.total_window_size - self.label_width
    self.labels_slice = slice(self.label_start, None)
    self.label_indices = np.arange(self.total_window_size)[self.labels_slice]



  def __repr__(self):
        return '\n'.join([
        f'Total window size: {self.total_window_size}',
        f'Input indices: {self.input_indices}',
        f'Label indices: {self.label_indices}',
        f'Label column name(s): {self.label_columns}'])
```

```
w1 = WindowGenerator(input_width=5, label_width=1, shift=24, label_columns=['T (degC)'])
w1
```

```
<__main__.WindowGenerator at 0x7fc60f8c9750>
```

```
w2 = WindowGenerator(input_width=5, label_width=1, shift=1, label_columns=['T (degC)'])
w2
```

```
<__main__.WindowGenerator at 0x7fc60f87cb10>
```

```
def split_window(self, features):
  inputs = features[:, self.input_slice, :]
  labels = features[:, self.labels_slice, :]
  if self.label_columns is not None:
    labels = tf.stack(
        [labels[:, :, self.column_indices[name]] for name in self.label_columns],
        axis=-1)

  # Slicing doesn't preserve static shape information, so set the shapes
  # manually. This way the `tf.data.Datasets` are easier to inspect.
  inputs.set_shape([None, self.input_width, None])
  labels.set_shape([None, self.label_width, None])

  return inputs, labels

WindowGenerator.split_window = split_window
```

```
example_window = tf.stack([np.array(train_df[:w2.total_window_size]),
                           np.array(train_df[100:100+w2.total_window_size]),
                           np.array(train_df[200:200+w2.total_window_size])])

example_inputs, example_labels = w2.split_window(example_window)

print('All shapes are: (batch, time, features)')
print(f'Window shape: {example_window.shape}')
print(f'Inputs shape: {example_inputs.shape}')
print(f'Labels shape: {example_labels.shape}')
```

```
All shapes are: (batch, time, features)
Window shape: (3, 6, 19)
Inputs shape: (3, 5, 19)
Labels shape: (3, 1, 1)
```

```
CONV_WIDTH = 5
conv_window = WindowGenerator(
    input_width=CONV_WIDTH,
    label_width=10,
    shift=1,
    label_columns=['T (degC)'])

conv_window
#conv_window.plot()
#plt.title("Given 5 hours of inputs, predict 10 hour into the future.")
```

```
<__main__.WindowGenerator at 0x7fc60f8bbe90>
```

```
conv_model = tf.keras.Sequential([
    tf.keras.layers.Conv1D(filters=32,
                           kernel_size=(CONV_WIDTH,),
                           activation='relu'),
    tf.keras.layers.Dense(units=64, activation='relu'),
    tf.keras.layers.Dense(units=1),
])
```

```
dense = tf.keras.Sequential([
    # Shape: (time, features) => (time*features)
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(units=32, activation='relu'),
    tf.keras.layers.Dense(units=32, activation='relu'),
    tf.keras.layers.Dense(units=1),
    # Add back the time dimension.
    # Shape: (outputs) => (1, outputs)
    tf.keras.layers.Reshape([1, -1]),
])
```

```
def make_dataset(self, data):
  data = np.array(data, dtype=np.float32)
  ds = tf.keras.utils.timeseries_dataset_from_array(
      data=data,
      targets=None,
      sequence_length=self.total_window_size,
      sequence_stride=1,
      shuffle=True,
      batch_size=32,)

  ds = ds.map(self.split_window)

  return ds

WindowGenerator.make_dataset = make_dataset
```

```
@property
def train(self):
  return self.make_dataset(self.train_df)

@property
def val(self):
  return self.make_dataset(self.val_df)

@property
def test(self):
  return self.make_dataset(self.test_df)

@property
def example(self):
  """Get and cache an example batch of `inputs, labels` for plotting."""
  result = getattr(self, '_example', None)
  if result is None:
    # No example batch was found, so get one from the `.train` dataset
    result = next(iter(self.train))
    # And cache it for next time
    self._example = result
  return result

WindowGenerator.train = train
WindowGenerator.val = val
WindowGenerator.test = test
WindowGenerator.example = example
```

```
MAX_EPOCHS = 2

def compile_and_fit(model, window, patience=2):
  early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
                                                    patience=patience,
                                                    mode='min')

  model.compile(loss=tf.losses.MeanSquaredError(),
                optimizer=tf.optimizers.Adam(),
                metrics=[tf.metrics.MeanAbsoluteError()])

  history = model.fit(window.train, epochs=MAX_EPOCHS,
                      validation_data=window.val,
                      callbacks=[early_stopping])
  return history
```

```
history = compile_and_fit(dense, conv_window)
```

```
      app.launch_new_instance()
    File "/usr/local/lib/python3.7/dist-packages/traitlets/config/application.py", line 846,
in launch_instance
      app.start()
    File "/usr/local/lib/python3.7/dist-packages/ipykernel/kernelapp.py", line 499, in start
      self.io_loop.start()
    File "/usr/local/lib/python3.7/dist-packages/tornado/platform/asyncio.py", line 132, in
start
      self.asyncio_loop.run_forever()
    File "/usr/lib/python3.7/asyncio/base_events.py", line 541, in run_forever
      self._run_once()
    File "/usr/lib/python3.7/asyncio/base_events.py", line 1786, in _run_once
      handle._run()
    File "/usr/lib/python3.7/asyncio/events.py", line 88, in _run
      self._context.run(self._callback, *self._args)
    File "/usr/local/lib/python3.7/dist-packages/tornado/platform/asyncio.py", line 122, in
_handle_events
      handler_func(fileobj, events)
    File "/usr/local/lib/python3.7/dist-packages/tornado/stack_context.py", line 300, in
null_wrapper
      return fn(*args, **kwargs)
    File "/usr/local/lib/python3.7/dist-packages/zmq/eventloop/zmqstream.py", line 577, in
_handle_events
      self._handle_recv()
    File "/usr/local/lib/python3.7/dist-packages/zmq/eventloop/zmqstream.py", line 606, in
_handle_recv
      self._run_callback(callback, msg)
    File "/usr/local/lib/python3.7/dist-packages/zmq/eventloop/zmqstream.py", line 556, in
_run_callback
      callback(*args, **kwargs)
    File "/usr/local/lib/python3.7/dist-packages/tornado/stack_context.py", line 300, in
null_wrapper
      return fn(*args, **kwargs)
    File "/usr/local/lib/python3.7/dist-packages/ipykernel/kernelbase.py", line 283, in
dispatcher
      return self.dispatch_shell(stream, msg)
    File "/usr/local/lib/python3.7/dist-packages/ipykernel/kernelbase.py", line 233, in
dispatch_shell
      handler(stream, idents, msg)
    File "/usr/local/lib/python3.7/dist-packages/ipykernel/kernelbase.py", line 399, in
execute_request
      user_expressions, allow_stdin)
    File "/usr/local/lib/python3.7/dist-packages/ipykernel/ipkernel.py", line 208, in
do_execute
      res = shell.run_cell(code, store_history=store_history, silent=silent)
    File "/usr/local/lib/python3.7/dist-packages/ipykernel/zmqshell.py", line 537, in
run_cell
      return super(ZMQInteractiveShell, self).run_cell(*args, **kwargs)
    File "/usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.py", line
2718, in run_cell
      interactivity=interactivity, compiler=compiler, result=result)
    File "/usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.py", line
2822, in run_ast_nodes
      if self.run_code(code, result):
    File "/usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.py", line
2882, in run_code
      exec(code_obj, self.user_global_ns, self.user_ns)
    File "<ipython-input-41-0e27aedfcfad>", line 1, in <module>
```

```
history = compile_and_fit(conv_model, conv_window)
```

```
      app.launch_new_instance()
    File "/usr/local/lib/python3.7/dist-packages/traitlets/config/application.py", line 846,
in launch_instance
      app.start()
    File "/usr/local/lib/python3.7/dist-packages/ipykernel/kernelapp.py", line 499, in start
      self.io_loop.start()
    File "/usr/local/lib/python3.7/dist-packages/tornado/platform/asyncio.py", line 132, in
start
      self.asyncio_loop.run_forever()
    File "/usr/lib/python3.7/asyncio/base_events.py", line 541, in run_forever
      self._run_once()
    File "/usr/lib/python3.7/asyncio/base_events.py", line 1786, in _run_once
      handle._run()
    File "/usr/lib/python3.7/asyncio/events.py", line 88, in _run
      self._context.run(self._callback, *self._args)
    File "/usr/local/lib/python3.7/dist-packages/tornado/platform/asyncio.py", line 122, in
_handle_events
      handler_func(fileobj, events)
    File "/usr/local/lib/python3.7/dist-packages/tornado/stack_context.py", line 300, in
null_wrapper
      return fn(*args, **kwargs)
    File "/usr/local/lib/python3.7/dist-packages/zmq/eventloop/zmqstream.py", line 577, in
_handle_events
      self._handle_recv()
    File "/usr/local/lib/python3.7/dist-packages/zmq/eventloop/zmqstream.py", line 606, in
_handle_recv
      self._run_callback(callback, msg)
    File "/usr/local/lib/python3.7/dist-packages/zmq/eventloop/zmqstream.py", line 556, in
_run_callback
      callback(*args, **kwargs)
    File "/usr/local/lib/python3.7/dist-packages/tornado/stack_context.py", line 300, in
null_wrapper
      return fn(*args, **kwargs)
    File "/usr/local/lib/python3.7/dist-packages/ipykernel/kernelbase.py", line 283, in
dispatcher
      return self.dispatch_shell(stream, msg)
    File "/usr/local/lib/python3.7/dist-packages/ipykernel/kernelbase.py", line 233, in
dispatch_shell
      handler(stream, idents, msg)
    File "/usr/local/lib/python3.7/dist-packages/ipykernel/kernelbase.py", line 399, in
execute_request
      user_expressions, allow_stdin)
    File "/usr/local/lib/python3.7/dist-packages/ipykernel/ipkernel.py", line 208, in
do_execute
      res = shell.run_cell(code, store_history=store_history, silent=silent)
    File "/usr/local/lib/python3.7/dist-packages/ipykernel/zmqshell.py", line 537, in
run_cell
      return super(ZMQInteractiveShell, self).run_cell(*args, **kwargs)
    File "/usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.py", line
2718, in run_cell
      interactivity=interactivity, compiler=compiler, result=result)
    File "/usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.py", line
2822, in run_ast_nodes
      if self.run_code(code, result):
    File "/usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.py", line
2882, in run_code
      exec(code_obj, self.user_global_ns, self.user_ns)
    File "<ipython-input-43-8c19829a56aa>", line 1, in <module>
```

```
OUT_STEPS = 24
multi_window = WindowGenerator(input_width=24,
                               label_width=OUT_STEPS,
                               shift=OUT_STEPS)

multi_window.plot()
multi_window
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-45-0fa5a0a302db> in <module>()
      4                                    shift=OUT_STEPS)
      5 
----> 6 multi_window.plot()
      7 multi_window

AttributeError: 'WindowGenerator' object has no attribute 'plot'
```

```
multi_lstm_model = tf.keras.Sequential([
    # Shape [batch, time, features] => [batch, lstm_units].
    # Adding more `lstm_units` just overfits more quickly.
    tf.keras.layers.LSTM(32, return_sequences=False),
    # Shape => [batch, out_steps*features].
    tf.keras.layers.Dense(OUT_STEPS*num_features),
    # Shape => [batch, out_steps, features].
    tf.keras.layers.Reshape([OUT_STEPS, num_features])
])
```

```
history = compile_and_fit(multi_lstm_model, multi_window)
```

```
Epoch 1/2
9199/9199 [==============================] - 132s 14ms/step - loss: 0.1037 - mean_absolute_error: 0.1763 - val_loss: 0.0874 - val_mean_abso
Epoch 2/2
2186/9199 [======>.......................] - ETA: 1:23 - loss: 0.0865 - mean_absolute_error: 0.1539
```

Executing (2m 52s) Cell › compile_and_fit() › error_handler() › fit() › error_handler() › __call__() › _call() › __call__() › _call_flat() › call() › quick_execute()