Wage LSTM Model - Training (1997-2020), Training (2021-2023)

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.preprocessing import MinMaxScaler, LabelEncoder
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Input
from sklearn.model_selection import TimeSeriesSplit

import gdown
import time
```

In [ ]:
```
# Load the dataset
url = 'https://drive.google.com/uc?id=1wjTiPLhi938Ro-jfjVHF0d_YPvsLaRc3'

# Download the file
output = 'data_wage.csv'
gdown.download(url, output, quiet=False)

# Check the file content
with open(output, 'r') as file:
    content = file.read()
    print("File content preview:")
    print(content[:500])

# Load the CSV file
try:
    data = pd.read_csv(output, delimiter=',')
    print(data.head())
except pd.errors.ParserError as e:
    print("Error parsing CSV file:", e)
```

```
Downloading...
From: https://drive.google.com/uc?id=1wjTiPLhi938Ro-jfjVHF0d_YPvsLaRc3
To: d:\OneDrive (Personal)\OneDrive\~ TMU 2023\CIND 820 - Big Data Analytics Project
\06 - Initial Results & Code (10%)\data_wage.csv
100%|██████████| 80.1M/80.1M [00:01<00:00, 43.0MB/s]
```

```
File content preview:
ref_date,geo,wages,type_of_work,sex,age_group,value,occupation_classification,noc,se
x_binary,age_group_numeric,geo_code,date_ordinal,year,month
1997-01-01,newfoundland and labrador,average hourly wage rate,full-time employees,ma
les,25 to 54 years,18.7,legislative and senior management occupations,00,1,1,210,729
025,1997,1
1997-02-01,newfoundland and labrador,average hourly wage rate,full-time employees,ma
les,25 to 54 years,18.48,legislative and senior management occupations,00,1,1,210,72
9056,1997
     ref_date                          geo                     wages  \
0  1997-01-01  newfoundland and labrador  average hourly wage rate
1  1997-02-01  newfoundland and labrador  average hourly wage rate
2  1997-03-01  newfoundland and labrador  average hourly wage rate
3  1997-04-01  newfoundland and labrador  average hourly wage rate
4  1997-05-01  newfoundland and labrador  average hourly wage rate

          type_of_work    sex       age_group  value  \
0  full-time employees  males  25 to 54 years  18.70
1  full-time employees  males  25 to 54 years  18.48
2  full-time employees  males  25 to 54 years  27.87
3  full-time employees  males  25 to 54 years  23.32
4  full-time employees  males  25 to 54 years  23.08

                      occupation_classification noc  sex_binary  \
0  legislative and senior management occupations  00           1
1  legislative and senior management occupations  00           1
2  legislative and senior management occupations  00           1
3  legislative and senior management occupations  00           1
4  legislative and senior management occupations  00           1

   age_group_numeric  geo_code  date_ordinal  year  month
0                  1       210        729025  1997      1
1                  1       210        729056  1997      2
2                  1       210        729084  1997      3
3                  1       210        729115  1997      4
4                  1       210        729145  1997      5
```

In [ ]:
```
# Convert 'ref_date' to datetime
data['ref_date'] = pd.to_datetime(data['ref_date'], format='%Y-%m-%d')

# Label encode the 'occupation_classification' column
data['occupation_classification'] = data['occupation_classification'].astype(str)
label_encoder = LabelEncoder()
data['occupation_code'] = label_encoder.fit_transform(data['occupation_classificati
```

In [ ]:
```
# Normalize the data
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(data[['value', 'date_ordinal', 'sex_binary', 'ag

# Function to create sequences for forecasting
def create_sequences(data, seq_length):
    xs, ys = [], []
    for i in range(len(data) - seq_length):
        x = data[i:i+seq_length]
        y = data[i+seq_length, 0]
        xs.append(x)
```

```
        ys.append(y)
    return np.array(xs), np.array(ys)

# Define sequence length
SEQ_LENGTH = 12

# Split the data into training and testing sets based on the date
train_data = data[data['ref_date'] < '2021-01-01']
test_data = data[data['ref_date'] >= '2021-01-01']

# Normalize training and testing data separately
scaled_train_data = scaler.fit_transform(train_data[['value', 'date_ordinal', 'year
scaled_test_data = scaler.transform(test_data[['value', 'date_ordinal', 'year', 'mo
```

In [ ]:

```
# Create sequences for training and testing
X_train, y_train = create_sequences(scaled_train_data, SEQ_LENGTH)
X_test, y_test = create_sequences(scaled_test_data, SEQ_LENGTH)

# Define cross-validation procedure
tscv = TimeSeriesSplit(n_splits=5)
cv_mse_scores = []

# Perform cross-validation
for train_index, val_index in tscv.split(X_train):
    X_train_cv, X_val_cv = X_train[train_index], X_train[val_index]
    y_train_cv, y_val_cv = y_train[train_index], y_train[val_index]

    # Define the LSTM model for forecasting
    model = Sequential()
    model.add(Input(shape=(SEQ_LENGTH, X_train.shape[2])))
    model.add(LSTM(50, activation='relu'))
    model.add(Dense(1))
    model.compile(optimizer='adam', loss='mse')

    # Train the model
    start = time.time()
    history = model.fit(X_train_cv, y_train_cv, epochs=200, batch_size=32, validati
    end = time.time()

    # Evaluate the model
    val_loss = model.evaluate(X_val_cv, y_val_cv, verbose=0)
    cv_mse_scores.append(val_loss)

    print(f"Validation Loss: {val_loss}, Training time: {end - start} seconds")

print('Cross-Validation Mean Squared Error:', np.mean(cv_mse_scores))
```

```
Validation Loss: 0.0041073705069720745, Training time: 1485.4001214504242 seconds
Validation Loss: 0.0027351335156708956, Training time: 2564.582067012787 seconds
Validation Loss: 0.0022012367844581604, Training time: 3694.7265124320984 seconds
Validation Loss: 0.0035521364770829678, Training time: 5118.316879749298 seconds
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
Cell In[7], line 23
     21 # Train the model
     22 start = time.time()
---> 23 history = model.fit(X_train_cv, y_train_cv, epochs=200, batch_size=32, valid
ation_data=(X_val_cv, y_val_cv), verbose=0)
     24 end = time.time()
     26 # Evaluate the model

File c:\Users\nesha\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras
\src\utils\traceback_utils.py:117, in filter_traceback.<locals>.error_handler(*args,
**kwargs)
    115 filtered_tb = None
    116 try:
--> 117     return fn(*args, **kwargs)
    118 except Exception as e:
    119     filtered_tb = _process_traceback_frames(e.__traceback__)

File c:\Users\nesha\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras
\src\backend\tensorflow\trainer.py:339, in TensorFlowTrainer.fit(self, x, y, batch_s
ize, epochs, verbose, callbacks, validation_split, validation_data, shuffle, class_w
eight, sample_weight, initial_epoch, steps_per_epoch, validation_steps, validation_b
atch_size, validation_freq)
    328 if getattr(self, "_eval_epoch_iterator", None) is None:
    329     self._eval_epoch_iterator = TFEpochIterator(
    330         x=val_x,
    331         y=val_y,
   (...)
    337         shuffle=False,
    338     )
--> 339 val_logs = self.evaluate(
    340     x=val_x,
    341     y=val_y,
    342     sample_weight=val_sample_weight,
    343     batch_size=validation_batch_size or batch_size,
    344     steps=validation_steps,
    345     callbacks=callbacks,
    346     return_dict=True,
    347     _use_cached_eval_dataset=True,
    348 )
    349 val_logs = {
    350     "val_" + name: val for name, val in val_logs.items()
    351 }
    352 epoch_logs.update(val_logs)

File c:\Users\nesha\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras
\src\utils\traceback_utils.py:117, in filter_traceback.<locals>.error_handler(*args,
**kwargs)
    115 filtered_tb = None
    116 try:
--> 117     return fn(*args, **kwargs)
    118 except Exception as e:
    119     filtered_tb = _process_traceback_frames(e.__traceback__)

File c:\Users\nesha\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras
```

```
\src\backend\tensorflow\trainer.py:425, in TensorFlowTrainer.evaluate(self, x, y, ba
tch_size, verbose, sample_weight, steps, callbacks, return_dict, **kwargs)
    423 for step, iterator in epoch_iterator.enumerate_epoch():
    424     callbacks.on_test_batch_begin(step)
--> 425     logs = self.test_function(iterator)
    426     logs = self._pythonify_logs(logs)
    427     callbacks.on_test_batch_end(step, logs)

File c:\Users\nesha\AppData\Local\Programs\Python\Python311\Lib\site-packages\tensor
flow\python\util\traceback_utils.py:150, in filter_traceback.<locals>.error_handler
(*args, **kwargs)
    148 filtered_tb = None
    149 try:
--> 150   return fn(*args, **kwargs)
    151 except Exception as e:
    152   filtered_tb = _process_traceback_frames(e.__traceback__)

File c:\Users\nesha\AppData\Local\Programs\Python\Python311\Lib\site-packages\tensor
flow\python\eager\polymorphic_function\polymorphic_function.py:833, in Function.__ca
ll__(self, *args, **kwds)
    830 compiler = "xla" if self._jit_compile else "nonXla"
    832 with OptionalXlaContext(self._jit_compile):
--> 833   result = self._call(*args, **kwds)
    835 new_tracing_count = self.experimental_get_tracing_count()
    836 without_tracing = (tracing_count == new_tracing_count)

File c:\Users\nesha\AppData\Local\Programs\Python\Python311\Lib\site-packages\tensor
flow\python\eager\polymorphic_function\polymorphic_function.py:878, in Function._cal
l(self, *args, **kwds)
    875 self._lock.release()
    876 # In this case we have not created variables on the first call. So we can
    877 # run the first trace but we should fail if variables are created.
--> 878 results = tracing_compilation.call_function(
    879     args, kwds, self._variable_creation_config
    880 )
    881 if self._created_variables:
    882   raise ValueError("Creating variables on a non-first call to a function"
    883                    " decorated with tf.function.")

File c:\Users\nesha\AppData\Local\Programs\Python\Python311\Lib\site-packages\tensor
flow\python\eager\polymorphic_function\tracing_compilation.py:139, in call_function
(args, kwargs, tracing_options)
    137 bound_args = function.function_type.bind(*args, **kwargs)
    138 flat_inputs = function.function_type.unpack_inputs(bound_args)
--> 139 return function._call_flat(  # pylint: disable=protected-access
    140     flat_inputs, captured_inputs=function.captured_inputs
    141 )

File c:\Users\nesha\AppData\Local\Programs\Python\Python311\Lib\site-packages\tensor
flow\python\eager\polymorphic_function\concrete_function.py:1322, in ConcreteFunctio
n._call_flat(self, tensor_inputs, captured_inputs)
   1318 possible_gradient_type = gradients_util.PossibleTapeGradientTypes(args)
   1319 if (possible_gradient_type == gradients_util.POSSIBLE_GRADIENT_TYPES_NONE
   1320     and executing_eagerly):
   1321   # No tape is watching; skip to running the function.
-> 1322   return self._inference_function.call_preflattened(args)
```

```
   1323 forward_backward = self._select_forward_and_backward_functions(
   1324     args,
   1325     possible_gradient_type,
   1326     executing_eagerly)
   1327 forward_function, args_with_tangents = forward_backward.forward()

File c:\Users\nesha\AppData\Local\Programs\Python\Python311\Lib\site-packages\tensorflow\python\eager\polymorphic_function\atomic_function.py:216, in AtomicFunction.call_preflattened(self, args)
    214 def call_preflattened(self, args: Sequence[core.Tensor]) -> Any:
    215   """Calls with flattened tensor inputs and returns the structured output."""
--> 216   flat_outputs = self.call_flat(*args)
    217   return self.function_type.pack_output(flat_outputs)

File c:\Users\nesha\AppData\Local\Programs\Python\Python311\Lib\site-packages\tensorflow\python\eager\polymorphic_function\atomic_function.py:251, in AtomicFunction.call_flat(self, *args)
    249 with record.stop_recording():
    250   if self._bound_context.executing_eagerly():
--> 251     outputs = self._bound_context.call_function(
    252         self.name,
    253         list(args),
    254         len(self.function_type.flat_outputs),
    255     )
    256   else:
    257     outputs = make_call_op_in_graph(
    258         self,
    259         list(args),
    260         self._bound_context.function_call_options.as_attrs(),
    261     )

File c:\Users\nesha\AppData\Local\Programs\Python\Python311\Lib\site-packages\tensorflow\python\eager\context.py:1500, in Context.call_function(self, name, tensor_inputs, num_outputs)
   1498 cancellation_context = cancellation.context()
   1499 if cancellation_context is None:
-> 1500   outputs = execute.execute(
   1501       name.decode("utf-8"),
   1502       num_outputs=num_outputs,
   1503       inputs=tensor_inputs,
   1504       attrs=attrs,
   1505       ctx=self,
   1506   )
   1507 else:
   1508   outputs = execute.execute_with_cancellation(
   1509       name.decode("utf-8"),
   1510       num_outputs=num_outputs,
   (...)
   1514       cancellation_manager=cancellation_context,
   1515   )

File c:\Users\nesha\AppData\Local\Programs\Python\Python311\Lib\site-packages\tensorflow\python\eager\execute.py:53, in quick_execute(op_name, num_outputs, inputs, attrs, ctx, name)
     51 try:
```

```
     52   ctx.ensure_initialized()
---> 53   tensors = pywrap_tfe.TFE_Py_Execute(ctx._handle, device_name, op_name,
     54                                       inputs, attrs, num_outputs)
     55 except core._NotOkStatusException as e:
     56   if name is not None:

KeyboardInterrupt:
```

In [ ]:
```
# Create sequences for training and testing
X_train, y_train = create_sequences(scaled_train_data, SEQ_LENGTH)
X_test, y_test = create_sequences(scaled_test_data, SEQ_LENGTH)
```

In [ ]:
```
# Define the LSTM model for forecasting
model = Sequential()
model.add(Input(shape=(SEQ_LENGTH, X_train.shape[2])))
model.add(LSTM(50, activation='relu'))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse')

print(model.summary())

# Train the model
start = time.time()
history = model.fit(X_train, y_train, epochs=500, batch_size=32, validation_split=0
end = time.time()

# Convert elapsed time to minutes and seconds
elapsed_time = end - start
minutes = int(elapsed_time // 60)
seconds = int(elapsed_time % 60)

print(f"\nTraining time: {minutes} minutes and {seconds} seconds")
```

Model: "sequential"

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 50)	11,800
dense (Dense)	(None, 1)	51

Total params: 11,851 (46.29 KB)

Trainable params: 11,851 (46.29 KB)

Non-trainable params: 0 (0.00 B)

None
Epoch 1/500
1517/9427 ━━━━━━━━━━━━━━━━━━━━ **27s** 3ms/step - loss: 0.0096

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
Cell In[5], line 16
     14 # Train the model
     15 start = time.time()
---> 16 history = model.fit(X_train, y_train, epochs=500, batch_size=32, validation_
split=0.2)
     17 end = time.time()
     19 # Convert elapsed time to minutes and seconds

File c:\Users\nesha\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras
\src\utils\traceback_utils.py:117, in filter_traceback.<locals>.error_handler(*args,
**kwargs)
    115 filtered_tb = None
    116 try:
--> 117     return fn(*args, **kwargs)
    118 except Exception as e:
    119     filtered_tb = _process_traceback_frames(e.__traceback__)

File c:\Users\nesha\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras
\src\backend\tensorflow\trainer.py:314, in TensorFlowTrainer.fit(self, x, y, batch_s
ize, epochs, verbose, callbacks, validation_split, validation_data, shuffle, class_w
eight, sample_weight, initial_epoch, steps_per_epoch, validation_steps, validation_b
atch_size, validation_freq)
    312 for step, iterator in epoch_iterator.enumerate_epoch():
    313     callbacks.on_train_batch_begin(step)
--> 314     logs = self.train_function(iterator)
    315     logs = self._pythonify_logs(logs)
    316     callbacks.on_train_batch_end(step, logs)

File c:\Users\nesha\AppData\Local\Programs\Python\Python311\Lib\site-packages\tensor
flow\python\util\traceback_utils.py:150, in filter_traceback.<locals>.error_handler
(*args, **kwargs)
    148 filtered_tb = None
    149 try:
--> 150   return fn(*args, **kwargs)
    151 except Exception as e:
    152   filtered_tb = _process_traceback_frames(e.__traceback__)

File c:\Users\nesha\AppData\Local\Programs\Python\Python311\Lib\site-packages\tensor
flow\python\eager\polymorphic_function\polymorphic_function.py:833, in Function.__ca
ll__(self, *args, **kwds)
    830 compiler = "xla" if self._jit_compile else "nonXla"
    832 with OptionalXlaContext(self._jit_compile):
--> 833   result = self._call(*args, **kwds)
    835 new_tracing_count = self.experimental_get_tracing_count()
    836 without_tracing = (tracing_count == new_tracing_count)

File c:\Users\nesha\AppData\Local\Programs\Python\Python311\Lib\site-packages\tensor
flow\python\eager\polymorphic_function\polymorphic_function.py:878, in Function._cal
l(self, *args, **kwds)
    875 self._lock.release()
    876 # In this case we have not created variables on the first call. So we can
    877 # run the first trace but we should fail if variables are created.
--> 878 results = tracing_compilation.call_function(
    879     args, kwds, self._variable_creation_config
```

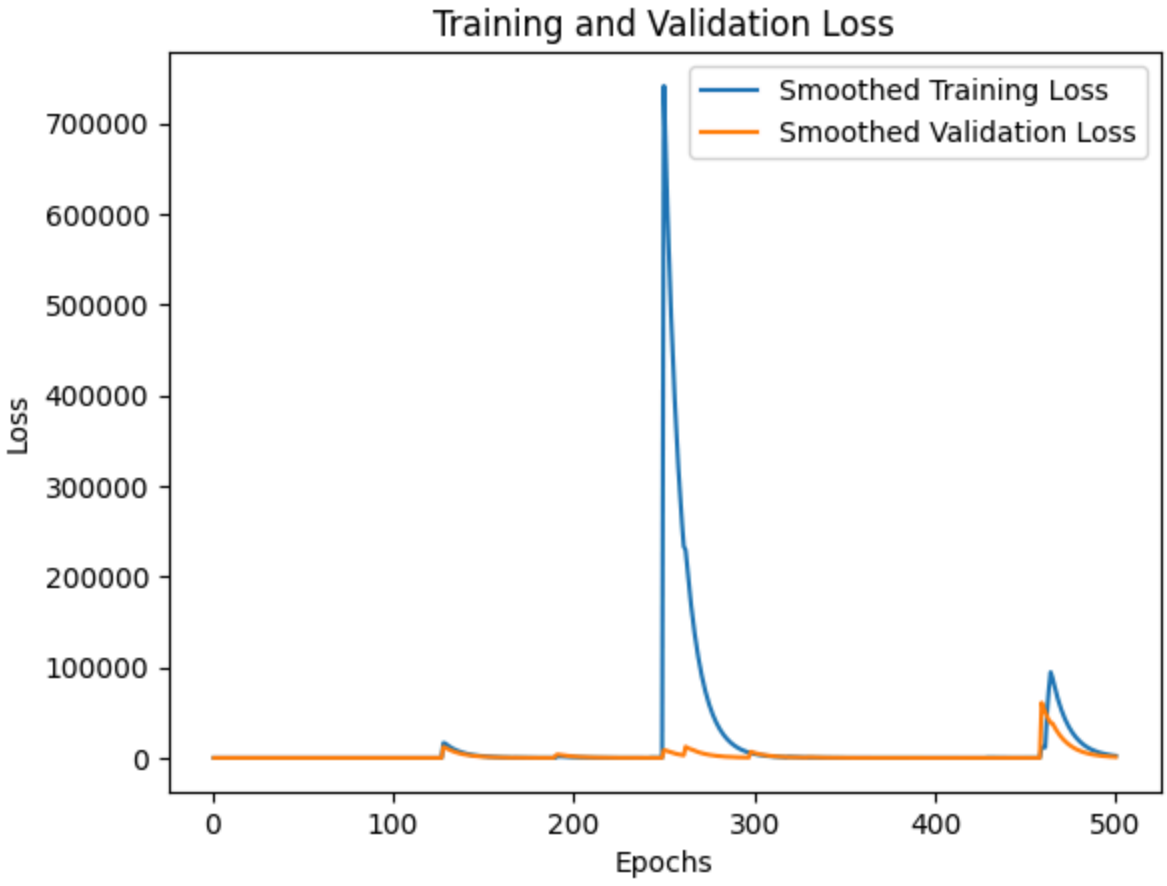
```
    880 )
    881 if self._created_variables:
    882   raise ValueError("Creating variables on a non-first call to a function"
    883                    " decorated with tf.function.")

File c:\Users\nesha\AppData\Local\Programs\Python\Python311\Lib\site-packages\tensorflow\python\eager\polymorphic_function\tracing_compilation.py:139, in call_function(args, kwargs, tracing_options)
    137 bound_args = function.function_type.bind(*args, **kwargs)
    138 flat_inputs = function.function_type.unpack_inputs(bound_args)
--> 139 return function._call_flat(  # pylint: disable=protected-access
    140     flat_inputs, captured_inputs=function.captured_inputs
    141 )

File c:\Users\nesha\AppData\Local\Programs\Python\Python311\Lib\site-packages\tensorflow\python\eager\polymorphic_function\concrete_function.py:1322, in ConcreteFunction._call_flat(self, tensor_inputs, captured_inputs)
   1318 possible_gradient_type = gradients_util.PossibleTapeGradientTypes(args)
   1319 if (possible_gradient_type == gradients_util.POSSIBLE_GRADIENT_TYPES_NONE
   1320     and executing_eagerly):
   1321   # No tape is watching; skip to running the function.
-> 1322   return self._inference_function.call_preflattened(args)
   1323 forward_backward = self._select_forward_and_backward_functions(
   1324     args,
   1325     possible_gradient_type,
   1326     executing_eagerly)
   1327 forward_function, args_with_tangents = forward_backward.forward()

File c:\Users\nesha\AppData\Local\Programs\Python\Python311\Lib\site-packages\tensorflow\python\eager\polymorphic_function\atomic_function.py:216, in AtomicFunction.call_preflattened(self, args)
    214 def call_preflattened(self, args: Sequence[core.Tensor]) -> Any:
    215   """Calls with flattened tensor inputs and returns the structured output."""
--> 216   flat_outputs = self.call_flat(*args)
    217   return self.function_type.pack_output(flat_outputs)

File c:\Users\nesha\AppData\Local\Programs\Python\Python311\Lib\site-packages\tensorflow\python\eager\polymorphic_function\atomic_function.py:251, in AtomicFunction.call_flat(self, *args)
    249 with record.stop_recording():
    250   if self._bound_context.executing_eagerly():
--> 251     outputs = self._bound_context.call_function(
    252         self.name,
    253         list(args),
    254         len(self.function_type.flat_outputs),
    255     )
    256   else:
    257     outputs = make_call_op_in_graph(
    258         self,
    259         list(args),
    260         self._bound_context.function_call_options.as_attrs(),
    261     )

File c:\Users\nesha\AppData\Local\Programs\Python\Python311\Lib\site-packages\tensorflow\python\eager\context.py:1500, in Context.call_function(self, name, tensor_input
```

```
s, num_outputs)
   1498 cancellation_context = cancellation.context()
   1499 if cancellation_context is None:
-> 1500   outputs = execute.execute(
   1501       name.decode("utf-8"),
   1502       num_outputs=num_outputs,
   1503       inputs=tensor_inputs,
   1504       attrs=attrs,
   1505       ctx=self,
   1506   )
   1507 else:
   1508   outputs = execute.execute_with_cancellation(
   1509       name.decode("utf-8"),
   1510       num_outputs=num_outputs,
   (...)
   1514       cancellation_manager=cancellation_context,
   1515   )

File c:\Users\nesha\AppData\Local\Programs\Python\Python311\Lib\site-packages\tensor
flow\python\eager\execute.py:53, in quick_execute(op_name, num_outputs, inputs, attr
s, ctx, name)
     51 try:
     52   ctx.ensure_initialized()
---> 53   tensors = pywrap_tfe.TFE_Py_Execute(ctx._handle, device_name, op_name,
     54                                       inputs, attrs, num_outputs)
     55 except core._NotOkStatusException as e:
     56   if name is not None:

KeyboardInterrupt:
```

In [ ]:

```
# Define the smoothing function
def smooth_curve(points, factor=0.9):
    smoothed_points = []
    for point in points:
        if smoothed_points:
            previous = smoothed_points[-1]
            smoothed_points.append(previous * factor + point * (1 - factor))
        else:
            smoothed_points.append(point)
    return smoothed_points

# Retrieve loss and validation loss from history
loss = history.history['loss']
val_loss = history.history['val_loss']

# Smooth the loss curves
smoothed_loss = smooth_curve(loss)
smoothed_val_loss = smooth_curve(val_loss)

# Plot smoothed training and validation loss
plt.plot(range(1, len(smoothed_loss) + 1), smoothed_loss, label='Smoothed Training
plt.plot(range(1, len(smoothed_val_loss) + 1), smoothed_val_loss, label='Smoothed V
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
```

```
plt.show()

# Identify the epoch with the lowest validation loss
best_epoch = np.argmin(smoothed_val_loss) + 1
print(f"Best epoch based on validation loss: {best_epoch}")


# Refine the LSTM model
model = Sequential()
model.add(Input(shape=(SEQ_LENGTH, X_train.shape[2])))
model.add(LSTM(50, activation='relu'))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse')

print(model.summary())

# Train the model
start = time.time()
history = model.fit(X_train, y_train, epochs=best_epoch, batch_size=32, validation_
end = time.time()

print(f"\nTraining time: {minutes} minutes and {seconds} seconds")
```




```
 Best epoch based on validation loss: 56
Model: "sequential_1"
```

Layer (type)	Output Shape	Param #
lstm_1 (LSTM)	(None, 50)	11,800
dense_1 (Dense)	(None, 1)	51

Total params: 11,851 (46.29 KB)

Trainable params: 11,851 (46.29 KB)

Non-trainable params: 0 (0.00 B)

```
None
Epoch 1/56
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 31s 3ms/step - loss: 0.0047 - val_loss: 0.0031
Epoch 2/56
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 29s 3ms/step - loss: 0.0027 - val_loss: 0.0031
Epoch 3/56
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 29s 3ms/step - loss: 0.0026 - val_loss: 0.0030
Epoch 4/56
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 29s 3ms/step - loss: 0.0025 - val_loss: 0.0030
Epoch 5/56
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 29s 3ms/step - loss: 0.0026 - val_loss: 0.0029
Epoch 6/56
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 29s 3ms/step - loss: 0.0025 - val_loss: 0.0030
Epoch 7/56
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 29s 3ms/step - loss: 0.0025 - val_loss: 0.0029
Epoch 8/56
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 29s 3ms/step - loss: 0.0025 - val_loss: 0.0029
Epoch 9/56
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 30s 3ms/step - loss: 0.0024 - val_loss: 0.0031
Epoch 10/56
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 29s 3ms/step - loss: 0.0024 - val_loss: 0.0029
Epoch 11/56
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 29s 3ms/step - loss: 0.0024 - val_loss: 0.0030
Epoch 12/56
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 29s 3ms/step - loss: 0.0024 - val_loss: 0.0028
Epoch 13/56
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 29s 3ms/step - loss: 0.0024 - val_loss: 0.0029
Epoch 14/56
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 29s 3ms/step - loss: 0.0024 - val_loss: 0.0030
Epoch 15/56
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 30s 3ms/step - loss: 0.0024 - val_loss: 0.0030
Epoch 16/56
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 29s 3ms/step - loss: 0.0024 - val_loss: 0.0028
Epoch 17/56
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 29s 3ms/step - loss: 0.0024 - val_loss: 0.0028
Epoch 18/56
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 29s 3ms/step - loss: 0.0024 - val_loss: 0.0031
Epoch 19/56
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 29s 3ms/step - loss: 0.0024 - val_loss: 0.0029
Epoch 20/56
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 30s 3ms/step - loss: 0.0024 - val_loss: 0.0028
Epoch 21/56
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 29s 3ms/step - loss: 0.0023 - val_loss: 0.0028
Epoch 22/56
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 29s 3ms/step - loss: 0.0023 - val_loss: 0.0028
Epoch 23/56
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 29s 3ms/step - loss: 0.0024 - val_loss: 0.0030
Epoch 24/56
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 29s 3ms/step - loss: 0.0023 - val_loss: 0.0029
Epoch 25/56
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 29s 3ms/step - loss: 0.0023 - val_loss: 0.0028
Epoch 26/56
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 29s 3ms/step - loss: 0.0023 - val_loss: 0.0030
Epoch 27/56
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 29s 3ms/step - loss: 0.0023 - val_loss: 0.0030
Epoch 28/56
```

```
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 29s 3ms/step - loss: 0.0023 - val_loss: 0.0029
Epoch 29/56
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 29s 3ms/step - loss: 0.0023 - val_loss: 0.0029
Epoch 30/56
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 30s 3ms/step - loss: 0.0023 - val_loss: 0.0028
Epoch 31/56
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 29s 3ms/step - loss: 0.0023 - val_loss: 0.0029
Epoch 32/56
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 30s 3ms/step - loss: 0.0023 - val_loss: 0.0028
Epoch 33/56
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 30s 3ms/step - loss: 0.0023 - val_loss: 0.0028
Epoch 34/56
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 30s 3ms/step - loss: 0.0023 - val_loss: 0.0028
Epoch 35/56
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 29s 3ms/step - loss: 0.0023 - val_loss: 0.0029
Epoch 36/56
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 30s 3ms/step - loss: 0.0023 - val_loss: 0.0028
Epoch 37/56
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 30s 3ms/step - loss: 0.0023 - val_loss: 0.0028
Epoch 38/56
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 30s 3ms/step - loss: 0.0023 - val_loss: 0.0028
Epoch 39/56
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 29s 3ms/step - loss: 0.0023 - val_loss: 0.0027
Epoch 40/56
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 30s 3ms/step - loss: 0.0023 - val_loss: 0.0028
Epoch 41/56
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 29s 3ms/step - loss: 0.0023 - val_loss: 0.0028
Epoch 42/56
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 30s 3ms/step - loss: 0.0023 - val_loss: 0.0028
Epoch 43/56
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 29s 3ms/step - loss: 0.0023 - val_loss: 0.0029
Epoch 44/56
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 30s 3ms/step - loss: 0.0023 - val_loss: 0.0027
Epoch 45/56
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 29s 3ms/step - loss: 0.0023 - val_loss: 0.0029
Epoch 46/56
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 30s 3ms/step - loss: 0.0023 - val_loss: 0.0028
Epoch 47/56
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 29s 3ms/step - loss: 0.0023 - val_loss: 0.0027
Epoch 48/56
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 30s 3ms/step - loss: 0.0023 - val_loss: 0.0027
Epoch 49/56
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 29s 3ms/step - loss: 0.0022 - val_loss: 0.0027
Epoch 50/56
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 30s 3ms/step - loss: 0.0023 - val_loss: 0.0028
Epoch 51/56
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 29s 3ms/step - loss: 0.0023 - val_loss: 0.0027
Epoch 52/56
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 30s 3ms/step - loss: 0.0023 - val_loss: 0.0028
Epoch 53/56
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 29s 3ms/step - loss: 21.4818 - val_loss: 0.0064
Epoch 54/56
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 30s 3ms/step - loss: 0.0112 - val_loss: 0.0036
Epoch 55/56
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 30s 3ms/step - loss: 0.0028 - val_loss: 0.0029
Epoch 56/56
```

```
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 30s 3ms/step - loss: 0.0024 - val_loss: 0.0028

Training time: 256 minutes and 14 seconds
```

In [ ]:
```python
# Evaluate the model
loss = model.evaluate(X_test, y_test)
print(f'Test Loss: {loss}')

# Make predictions
predictions = model.predict(X_test)

# Inverse transform the predictions and the actual values
predictions_inv = scaler.inverse_transform(np.concatenate((predictions, np.zeros((p
y_test_inv = scaler.inverse_transform(np.concatenate((y_test.reshape(-1, 1), np.zer

# Compare the first few predictions with the actual values
comparison = pd.DataFrame({'Actual': y_test_inv, 'Predicted': predictions_inv})
print(comparison.head())
```

```
1516/1516 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - loss: 0.0225
Test Loss: 0.022555621340870857
1516/1516 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step
   Actual  Predicted
0   48.51  21.124145
1   49.43  31.216181
2   49.78  33.773115
3   50.11  47.519050
4   52.02  41.261429
```

In [ ]:
```python
# Wage LSTM Model - Training (1997-2020), Testing (2021-2023)
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler, LabelEncoder
from sklearn.model_selection import TimeSeriesSplit
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Input
import gdown
import time

# Load the dataset
url = 'https://drive.google.com/uc?id=1wjTiPLhi938Ro-jfjVHF0d_YPvsLaRc3'

# Download the file
output = 'data_wage.csv'
gdown.download(url, output, quiet=False)

# Check the file content
with open(output, 'r') as file:
    content = file.read()
    print("File content preview:")
    print(content[:500])

# Load the CSV file
```

```
try:
    data_wage = pd.read_csv(output, delimiter=',')
    print(data_wage.head())
except pd.errors.ParserError as e:
    print("Error parsing CSV file:", e)

# Convert 'ref_date' to datetime
data_wage['ref_date'] = pd.to_datetime(data_wage['ref_date'], format='%Y-%m-%d')

# Change 'ref_date' format to '%Y-%m'
data_wage['ref_date'] = data_wage['ref_date'].dt.to_period('M').dt.to_timestamp()

# Label encode the 'occupation_classification' column
data_wage['occupation_classification'] = data_wage['occupation_classification'].ast
label_encoder = LabelEncoder()
data_wage['occupation_code'] = label_encoder.fit_transform(data_wage['occupation_cl

# Normalize the data
scaler = MinMaxScaler(feature_range=(0, 1))
features = ['value', 'date_ordinal', 'sex_binary', 'age_group_numeric', 'geo_code',
scaled_data = scaler.fit_transform(data_wage[features])

# Function to create sequences for forecasting
def create_sequences(data, seq_length):
    xs, ys = [], []
    for i in range(len(data) - seq_length):
        x = data[i:i+seq_length]
        y = data[i+seq_length, 0]
        xs.append(x)
        ys.append(y)
    return np.array(xs), np.array(ys)

# Define sequence length
SEQ_LENGTH = 12

# Split the data into training and testing sets based on the date
train_data = data_wage[data_wage['ref_date'] < '2021-01']
test_data = data_wage[data_wage['ref_date'] >= '2021-01']

# Normalize training and testing data separately
scaled_train_data = scaler.fit_transform(train_data[features])
scaled_test_data = scaler.transform(test_data[features])

# Create sequences for training and testing
X_train, y_train = create_sequences(scaled_train_data, SEQ_LENGTH)
X_test, y_test = create_sequences(scaled_test_data, SEQ_LENGTH)

# Define cross-validation procedure
tscv = TimeSeriesSplit(n_splits=5)
cv_mse_scores = []

# Perform cross-validation
for train_index, val_index in tscv.split(X_train):
    X_train_cv, X_val_cv = X_train[train_index], X_train[val_index]
    y_train_cv, y_val_cv = y_train[train_index], y_train[val_index]
```

```
    # Define the LSTM model for forecasting
    model = Sequential()
    model.add(Input(shape=(SEQ_LENGTH, X_train.shape[2])))
    model.add(LSTM(50, activation='relu'))
    model.add(Dense(1))
    model.compile(optimizer='adam', loss='mse')

    # Train the model
    start = time.time()
    history = model.fit(X_train_cv, y_train_cv, epochs=100, batch_size=32, validati
    end = time.time()

    # Evaluate the model
    val_loss = model.evaluate(X_val_cv, y_val_cv, verbose=0)
    cv_mse_scores.append(val_loss)

    print(f"Validation Loss: {val_loss}, Training time: {end - start} seconds")

print('Cross-Validation Mean Squared Error:', np.mean(cv_mse_scores))

# Train the final model on the entire training set
model = Sequential()
model.add(Input(shape=(SEQ_LENGTH, X_train.shape[2])))
model.add(LSTM(50, activation='relu'))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse')

print(model.summary())

start = time.time()
history = model.fit(X_train, y_train, epochs=100, batch_size=32, validation_split=0
end = time.time()

# Convert elapsed time to minutes and seconds
elapsed_time = end - start
minutes = int(elapsed_time // 60)
seconds = int(elapsed_time % 60)

print(f"\nTraining time: {minutes} minutes and {seconds} seconds")

# Retrieve loss and validation loss from history
loss = history.history['loss']
val_loss = history.history['val_loss']

# Define the smoothing function
def smooth_curve(points, factor=0.9):
    smoothed_points = []
    for point in points:
        if smoothed_points:
            previous = smoothed_points[-1]
            smoothed_points.append(previous * factor + point * (1 - factor))
        else:
            smoothed_points.append(point)
    return smoothed_points

# Smooth the loss curves
```

```
smoothed_loss = smooth_curve(loss)
smoothed_val_loss = smooth_curve(val_loss)

# Plot smoothed training and validation loss
plt.plot(range(1, len(smoothed_loss) + 1), smoothed_loss, label='Smoothed Training
plt.plot(range(1, len(smoothed_val_loss) + 1), smoothed_val_loss, label='Smoothed V
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()

# Identify the epoch with the lowest validation loss
best_epoch = np.argmin(smoothed_val_loss) + 1
print(f"Best epoch based on validation loss: {best_epoch}")

# Refine the LSTM model
model = Sequential()
model.add(Input(shape=(SEQ_LENGTH, X_train.shape[2])))
model.add(LSTM(50, activation='relu'))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse')

print(model.summary())

# Train the model
start = time.time()
history = model.fit(X_train, y_train, epochs=best_epoch, batch_size=32, validation_
end = time.time()

print(f"\nTraining time: {minutes} minutes and {seconds} seconds")

# Evaluate the model
loss = model.evaluate(X_test, y_test)
print(f'Test Loss: {loss}')

# Make predictions
predictions = model.predict(X_test)

# Inverse transform the predictions and the actual values
predictions_inv = scaler.inverse_transform(np.concatenate((predictions, np.zeros((p
y_test_inv = scaler.inverse_transform(np.concatenate((y_test.reshape(-1, 1), np.zer

# Compare the first few predictions with the actual values
comparison = pd.DataFrame({'Actual': y_test_inv, 'Predicted': predictions_inv})
print(comparison.head())
```

```
Downloading...
From: https://drive.google.com/uc?id=1wjTiPLhi938Ro-jfjVHF0d_YPvsLaRc3
To: d:\OneDrive (Personal)\OneDrive\~ TMU 2023\CIND 820 - Big Data Analytics Project
\06 - Initial Results & Code (10%)\data_wage.csv
100%|██████████| 80.1M/80.1M [00:02<00:00, 38.7MB/s]
```

```
File content preview:
ref_date,geo,wages,type_of_work,sex,age_group,value,occupation_classification,noc,se
x_binary,age_group_numeric,geo_code,date_ordinal,year,month
1997-01-01,newfoundland and labrador,average hourly wage rate,full-time employees,ma
les,25 to 54 years,18.7,legislative and senior management occupations,00,1,1,210,729
025,1997,1
1997-02-01,newfoundland and labrador,average hourly wage rate,full-time employees,ma
les,25 to 54 years,18.48,legislative and senior management occupations,00,1,1,210,72
9056,1997
     ref_date                          geo                     wages  \
0  1997-01-01  newfoundland and labrador  average hourly wage rate
1  1997-02-01  newfoundland and labrador  average hourly wage rate
2  1997-03-01  newfoundland and labrador  average hourly wage rate
3  1997-04-01  newfoundland and labrador  average hourly wage rate
4  1997-05-01  newfoundland and labrador  average hourly wage rate

          type_of_work    sex       age_group  value  \
0  full-time employees  males  25 to 54 years  18.70
1  full-time employees  males  25 to 54 years  18.48
2  full-time employees  males  25 to 54 years  27.87
3  full-time employees  males  25 to 54 years  23.32
4  full-time employees  males  25 to 54 years  23.08

                      occupation_classification noc  sex_binary  \
0  legislative and senior management occupations  00           1
1  legislative and senior management occupations  00           1
2  legislative and senior management occupations  00           1
3  legislative and senior management occupations  00           1
4  legislative and senior management occupations  00           1

   age_group_numeric  geo_code  date_ordinal  year  month
0                  1       210        729025  1997      1
1                  1       210        729056  1997      2
2                  1       210        729084  1997      3
3                  1       210        729115  1997      4
4                  1       210        729145  1997      5
Validation Loss: 0.0027075393591076136, Training time: 710.9991371631622 seconds
Validation Loss: 0.0027307418640702963, Training time: 1157.2986750602722 seconds
Validation Loss: 0.0023171829525381327, Training time: 1626.4196791648865 seconds
Validation Loss: 0.003146085189655423, Training time: 2079.2018325328827 seconds
Validation Loss: 0.2398611158132553, Training time: 2571.675872325897 seconds
Cross-Validation Mean Squared Error: 0.05015253303572535
```

Model: "sequential_5"

Layer (type)	Output Shape	Param #
lstm_5 (LSTM)	(None, 50)	11,400
dense_5 (Dense)	(None, 1)	51

Total params: 11,451 (44.73 KB)

Trainable params: 11,451 (44.73 KB)

Non-trainable params: 0 (0.00 B)

```
None
Epoch 1/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 28s 3ms/step - loss: 0.0049 - val_loss: 0.0031
Epoch 2/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 27s 3ms/step - loss: 0.0027 - val_loss: 0.0031
Epoch 3/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0027 - val_loss: 0.0030
Epoch 4/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0026 - val_loss: 0.0030
Epoch 5/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0026 - val_loss: 0.0030
Epoch 6/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0026 - val_loss: 0.0030
Epoch 7/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0026 - val_loss: 0.0030
Epoch 8/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0026 - val_loss: 0.0030
Epoch 9/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0026 - val_loss: 0.0030
Epoch 10/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0026 - val_loss: 0.0032
Epoch 11/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0026 - val_loss: 0.0030
Epoch 12/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0026 - val_loss: 0.0030
Epoch 13/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0026 - val_loss: 0.0030
Epoch 14/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0026 - val_loss: 0.0030
Epoch 15/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0026 - val_loss: 0.0030
Epoch 16/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0025 - val_loss: 0.0030
Epoch 17/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0025 - val_loss: 0.0030
Epoch 18/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0026 - val_loss: 0.0030
Epoch 19/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0026 - val_loss: 0.0030
Epoch 20/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0026 - val_loss: 0.0030
Epoch 21/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0025 - val_loss: 0.0030
Epoch 22/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0025 - val_loss: 0.0030
Epoch 23/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0026 - val_loss: 0.0029
Epoch 24/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0025 - val_loss: 0.0030
Epoch 25/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0025 - val_loss: 0.0029
Epoch 26/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0025 - val_loss: 0.0030
Epoch 27/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0025 - val_loss: 0.0030
Epoch 28/100
```

```
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0025 - val_loss: 0.0029
Epoch 29/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0025 - val_loss: 0.0031
Epoch 30/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0025 - val_loss: 0.0029
Epoch 31/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0025 - val_loss: 0.0031
Epoch 32/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0025 - val_loss: 0.0030
Epoch 33/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0025 - val_loss: 0.0029
Epoch 34/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0025 - val_loss: 0.0029
Epoch 35/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0025 - val_loss: 0.0029
Epoch 36/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0025 - val_loss: 0.0030
Epoch 37/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0025 - val_loss: 0.0029
Epoch 38/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0025 - val_loss: 0.0029
Epoch 39/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0025 - val_loss: 0.0029
Epoch 40/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0025 - val_loss: 0.0029
Epoch 41/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0025 - val_loss: 0.0030
Epoch 42/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0025 - val_loss: 0.0029
Epoch 43/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0025 - val_loss: 0.0029
Epoch 44/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0025 - val_loss: 0.0029
Epoch 45/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0025 - val_loss: 0.0029
Epoch 46/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0025 - val_loss: 0.0030
Epoch 47/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0025 - val_loss: 0.0029
Epoch 48/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0025 - val_loss: 0.0031
Epoch 49/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0025 - val_loss: 0.0030
Epoch 50/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0025 - val_loss: 0.0030
Epoch 51/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0025 - val_loss: 0.0030
Epoch 52/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0024 - val_loss: 0.0029
Epoch 53/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0025 - val_loss: 0.0029
Epoch 54/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0025 - val_loss: 0.0029
Epoch 55/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0025 - val_loss: 0.0029
Epoch 56/100
```

```
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0024 - val_loss: 0.0029
Epoch 57/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0025 - val_loss: 0.0029
Epoch 58/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0024 - val_loss: 0.0029
Epoch 59/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0024 - val_loss: 0.0029
Epoch 60/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0025 - val_loss: 0.0030
Epoch 61/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0024 - val_loss: 0.0029
Epoch 62/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0024 - val_loss: 0.0030
Epoch 63/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 27s 3ms/step - loss: 0.0024 - val_loss: 0.0030
Epoch 64/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0025 - val_loss: 0.0029
Epoch 65/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0025 - val_loss: 0.0029
Epoch 66/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0024 - val_loss: 0.0029
Epoch 67/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0025 - val_loss: 0.0029
Epoch 68/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0024 - val_loss: 0.0029
Epoch 69/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0024 - val_loss: 0.0029
Epoch 70/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0024 - val_loss: 0.0029
Epoch 71/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0024 - val_loss: 0.0030
Epoch 72/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0024 - val_loss: 0.0029
Epoch 73/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0024 - val_loss: 0.0030
Epoch 74/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0025 - val_loss: 0.0029
Epoch 75/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0024 - val_loss: 0.0029
Epoch 76/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0024 - val_loss: 0.0029
Epoch 77/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0025 - val_loss: 0.0030
Epoch 78/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0024 - val_loss: 0.0029
Epoch 79/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 27s 3ms/step - loss: 0.0024 - val_loss: 0.0030
Epoch 80/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0024 - val_loss: 0.0030
Epoch 81/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0033 - val_loss: 0.0029
Epoch 82/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0024 - val_loss: 0.0029
Epoch 83/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0024 - val_loss: 0.0029
Epoch 84/100
```

```
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0024 - val_loss: 0.0029
Epoch 85/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0024 - val_loss: 0.0029
Epoch 86/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 27s 3ms/step - loss: 0.0025 - val_loss: 0.0029
Epoch 87/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 27s 3ms/step - loss: 0.0024 - val_loss: 0.0029
Epoch 88/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 27s 3ms/step - loss: 0.0024 - val_loss: 0.0029
Epoch 89/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0024 - val_loss: 0.0029
Epoch 90/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0024 - val_loss: 0.0030
Epoch 91/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0024 - val_loss: 0.0029
Epoch 92/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0024 - val_loss: 0.0029
Epoch 93/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0024 - val_loss: 0.0033
Epoch 94/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 27s 3ms/step - loss: 0.0024 - val_loss: 0.0029
Epoch 95/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0024 - val_loss: 0.0029
Epoch 96/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0024 - val_loss: 0.0030
Epoch 97/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0024 - val_loss: 0.0029
Epoch 98/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0024 - val_loss: 0.0029
Epoch 99/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0024 - val_loss: 0.0029
Epoch 100/100
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0024 - val_loss: 0.0029

Training time: 43 minutes and 39 seconds
```


```
Best epoch based on validation loss: 89
```

Model: "sequential_6"

Layer (type)	Output Shape	Param #
lstm_6 (LSTM)	(None, 50)	11,400
dense_6 (Dense)	(None, 1)	51

Total params: 11,451 (44.73 KB)

Trainable params: 11,451 (44.73 KB)

Non-trainable params: 0 (0.00 B)

```
None
Epoch 1/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 29s 3ms/step - loss: 0.0058 - val_loss: 0.0031
Epoch 2/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0027 - val_loss: 0.0030
Epoch 3/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0026 - val_loss: 0.0030
Epoch 4/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0026 - val_loss: 0.0031
Epoch 5/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 27s 3ms/step - loss: 0.0026 - val_loss: 0.0031
Epoch 6/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0026 - val_loss: 0.0031
Epoch 7/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 27s 3ms/step - loss: 0.0026 - val_loss: 0.0030
Epoch 8/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 27s 3ms/step - loss: 0.0026 - val_loss: 0.0030
Epoch 9/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 27s 3ms/step - loss: 0.0026 - val_loss: 0.0030
Epoch 10/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 26s 3ms/step - loss: 0.0025 - val_loss: 0.0030
Epoch 11/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 27s 3ms/step - loss: 0.0026 - val_loss: 0.0030
Epoch 12/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 27s 3ms/step - loss: 0.0026 - val_loss: 0.0030
Epoch 13/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 27s 3ms/step - loss: 0.0026 - val_loss: 0.0031
Epoch 14/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 27s 3ms/step - loss: 0.0025 - val_loss: 0.0030
Epoch 15/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 27s 3ms/step - loss: 0.0026 - val_loss: 0.0030
Epoch 16/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 32s 3ms/step - loss: 0.0025 - val_loss: 0.0030
Epoch 17/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 7331s 778ms/step - loss: 0.0025 - val_loss: 0.0030
Epoch 18/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 63s 7ms/step - loss: 0.0025 - val_loss: 0.0030
Epoch 19/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 57s 6ms/step - loss: 0.0025 - val_loss: 0.0030
Epoch 20/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 58s 6ms/step - loss: 0.0025 - val_loss: 0.0030
Epoch 21/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 58s 6ms/step - loss: 0.0025 - val_loss: 0.0030
Epoch 22/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 59s 6ms/step - loss: 0.0026 - val_loss: 0.0030
Epoch 23/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 57s 6ms/step - loss: 0.0025 - val_loss: 0.0030
Epoch 24/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 58s 6ms/step - loss: 0.0025 - val_loss: 0.0030
Epoch 25/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 58s 6ms/step - loss: 0.0025 - val_loss: 0.0030
Epoch 26/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 58s 6ms/step - loss: 0.0025 - val_loss: 0.0029
Epoch 27/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 58s 6ms/step - loss: 0.0025 - val_loss: 0.0030
Epoch 28/89
```

```
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 58s 6ms/step - loss: 0.0025 - val_loss: 0.0029
Epoch 29/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 57s 6ms/step - loss: 0.0025 - val_loss: 0.0029
Epoch 30/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 58s 6ms/step - loss: 0.0025 - val_loss: 0.0029
Epoch 31/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 57s 6ms/step - loss: 0.0025 - val_loss: 0.0030
Epoch 32/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 59s 6ms/step - loss: 0.0025 - val_loss: 0.0029
Epoch 33/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 58s 6ms/step - loss: 0.0025 - val_loss: 0.0029
Epoch 34/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 58s 6ms/step - loss: 0.0025 - val_loss: 0.0029
Epoch 35/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 58s 6ms/step - loss: 0.0025 - val_loss: 0.0029
Epoch 36/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 58s 6ms/step - loss: 0.0025 - val_loss: 0.0031
Epoch 37/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 60s 6ms/step - loss: 0.0025 - val_loss: 0.0030
Epoch 38/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 7346s 779ms/step - loss: 0.0025 - val_loss: 0.0029
Epoch 39/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 77s 8ms/step - loss: 0.0025 - val_loss: 0.0029
Epoch 40/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 77s 8ms/step - loss: 0.0025 - val_loss: 0.0030
Epoch 41/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 76s 8ms/step - loss: 0.0025 - val_loss: 0.0029
Epoch 42/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 76s 8ms/step - loss: 0.0025 - val_loss: 0.0030
Epoch 43/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 76s 8ms/step - loss: 0.0025 - val_loss: 0.0029
Epoch 44/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 76s 8ms/step - loss: 0.0025 - val_loss: 0.0029
Epoch 45/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 76s 8ms/step - loss: 0.0025 - val_loss: 0.0029
Epoch 46/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 76s 8ms/step - loss: 0.0025 - val_loss: 0.0030
Epoch 47/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 76s 8ms/step - loss: 0.0025 - val_loss: 0.0029
Epoch 48/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 77s 8ms/step - loss: 0.0025 - val_loss: 0.0029
Epoch 49/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 77s 8ms/step - loss: 0.0024 - val_loss: 0.0029
Epoch 50/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 77s 8ms/step - loss: 0.0025 - val_loss: 0.0029
Epoch 51/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 77s 8ms/step - loss: 0.0024 - val_loss: 0.0030
Epoch 52/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 77s 8ms/step - loss: 0.0025 - val_loss: 0.0030
Epoch 53/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 7336s 778ms/step - loss: 0.0025 - val_loss: 0.0030
Epoch 54/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 89s 9ms/step - loss: 0.0025 - val_loss: 0.0029
Epoch 55/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 75s 8ms/step - loss: 0.0025 - val_loss: 0.0029
Epoch 56/89
```

```
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 75s 8ms/step - loss: 0.0024 - val_loss: 0.0029
Epoch 57/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 75s 8ms/step - loss: 0.0025 - val_loss: 0.0030
Epoch 58/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 75s 8ms/step - loss: 0.0024 - val_loss: 0.0029
Epoch 59/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 74s 8ms/step - loss: 0.0024 - val_loss: 0.0029
Epoch 60/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 76s 8ms/step - loss: 0.0025 - val_loss: 0.0029
Epoch 61/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 75s 8ms/step - loss: 0.0025 - val_loss: 0.0031
Epoch 62/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 74s 8ms/step - loss: 0.0025 - val_loss: 0.0029
Epoch 63/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 74s 8ms/step - loss: 0.0024 - val_loss: 0.0029
Epoch 64/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 74s 8ms/step - loss: 0.0024 - val_loss: 0.0029
Epoch 65/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 75s 8ms/step - loss: 0.0024 - val_loss: 0.0030
Epoch 66/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 74s 8ms/step - loss: 0.0024 - val_loss: 0.0030
Epoch 67/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 74s 8ms/step - loss: 0.0024 - val_loss: 0.0029
Epoch 68/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 74s 8ms/step - loss: 0.0025 - val_loss: 0.0029
Epoch 69/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 74s 8ms/step - loss: 0.0024 - val_loss: 0.0029
Epoch 70/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 7342s 779ms/step - loss: 0.0025 - val_loss: 0.0030
Epoch 71/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 77s 8ms/step - loss: 0.0024 - val_loss: 0.0029
Epoch 72/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 73s 8ms/step - loss: 0.0024 - val_loss: 0.0029
Epoch 73/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 72s 8ms/step - loss: 0.0024 - val_loss: 0.0029
Epoch 74/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 73s 8ms/step - loss: 0.0024 - val_loss: 0.0029
Epoch 75/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 73s 8ms/step - loss: 0.0024 - val_loss: 0.0029
Epoch 76/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 73s 8ms/step - loss: 0.0024 - val_loss: 0.0029
Epoch 77/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 72s 8ms/step - loss: 0.0024 - val_loss: 0.0029
Epoch 78/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 73s 8ms/step - loss: 0.0024 - val_loss: 0.0029
Epoch 79/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 72s 8ms/step - loss: 141.0687 - val_loss: 0.0283
Epoch 80/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 72s 8ms/step - loss: 0.1419 - val_loss: 0.0062
Epoch 81/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 72s 8ms/step - loss: 0.0101 - val_loss: 0.0043
Epoch 82/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 72s 8ms/step - loss: 0.0033 - val_loss: 0.0031
Epoch 83/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 72s 8ms/step - loss: 0.0027 - val_loss: 0.0031
Epoch 84/89
```

```
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 72s 8ms/step - loss: 0.0026 - val_loss: 0.0029
Epoch 85/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 72s 8ms/step - loss: 0.0025 - val_loss: 0.0029
Epoch 86/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 4008s 425ms/step - loss: 0.0025 - val_loss: 0.0029
Epoch 87/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 220424s 23s/step - loss: 5.6695 - val_loss: 2.9401
Epoch 88/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 34s 4ms/step - loss: 0.1000 - val_loss: 0.0212
Epoch 89/89
9427/9427 ━━━━━━━━━━━━━━━━━━━━ 41s 4ms/step - loss: 0.0072 - val_loss: 0.0892

Training time: 43 minutes and 39 seconds
1516/1516 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - loss: 0.2851
Test Loss: 0.852683961391449
1516/1516 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step
   Actual  Predicted
0   48.51  51.434662
1   49.43  48.974869
2   49.78  50.224376
3   50.11  48.432188
4   52.02  47.580685
```