In [1]:

```
import os

import tensorflow as tf
import numpy as np

import pandas as pd

from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
from tensorflow.keras.losses import mse

from tensorflow.keras.metrics import RootMeanSquaredError, mean_squared_error

from tensorflow.keras.optimizers import Adam
import matplotlib.pyplot as plt
```

In [2]:

```
zip_path = tf.keras.utils.get_file(
    origin = "http://storage.googleapis.com/tensorflow/tf-keras-datasets/jena_climate_2009_2016.csv.
    fname = "jena_climate_2009_2016.csv.zip",
    extract = True)
```

In [3]:

```
csv_path, csv_extension = os.path.splitext(zip_path)

print("파일 경로 : ", csv_path, "파일 확장자명 : ", csv_extension)
```

```
파일 경로 :  C:₩Users₩wogml₩.keras₩datasets₩jena_climate_2009_2016.csv 파일 확장자명
:  .zip
```

In [4]:

```
import numpy as np

import pandas as pd
import tensorflow as tf

import os

from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
from tensorflow.keras.losses import mse

from tensorflow.keras.metrics import RootMeanSquaredError, mean_squared_error

from tensorflow.keras.optimizers import Adam
import matplotlib.pyplot as plt
```

In [5]:

```
weather_data = pd.read_csv(csv_path)

weather_data
```

Out[5]:

	Date Time	p (mbar)	T (degC)	Tpot (K)	Tdew (degC)	rh (%)	VPmax (mbar)	VPact (mbar)	VPdef (mbar)	sh (g/kg)	(mm
0	01.01.2009 00:10:00	996.52	-8.02	265.40	-8.90	93.30	3.33	3.11	0.22	1.94	
1	01.01.2009 00:20:00	996.57	-8.41	265.01	-9.28	93.40	3.23	3.02	0.21	1.89	
2	01.01.2009 00:30:00	996.53	-8.51	264.91	-9.31	93.90	3.21	3.01	0.20	1.88	
3	01.01.2009 00:40:00	996.51	-8.31	265.12	-9.07	94.20	3.26	3.07	0.19	1.92	
4	01.01.2009 00:50:00	996.51	-8.27	265.15	-9.04	94.10	3.27	3.08	0.19	1.92	
...	...	...	...	...	...	...	...	...	...	...	
420546	31.12.2016 23:20:00	1000.07	-4.05	269.10	-8.13	73.10	4.52	3.30	1.22	2.06	
420547	31.12.2016 23:30:00	999.93	-3.35	269.81	-8.06	69.71	4.77	3.32	1.44	2.07	
420548	31.12.2016 23:40:00	999.82	-3.16	270.01	-8.21	67.91	4.84	3.28	1.55	2.05	
420549	31.12.2016 23:50:00	999.81	-4.23	268.94	-8.53	71.80	4.46	3.20	1.26	1.99	
420550	01.01.2017 00:00:00	999.82	-4.82	268.36	-8.42	75.70	4.27	3.23	1.04	2.01	

420551 rows × 15 columns

In [6]:

```
print("Date Time 컬럼 데이터의 type : ", type(weather_data["Date Time"][0]))
```

```
Date Time 컬럼 데이터의 type :  <class 'str'>
```

In [7]:

```
for i in range(len(weather_data)):
    if weather_data["Date Time"][i][6:10] == '2016':
        print("2016년 데이터는 ", str(i), "번째 행부터 시작입니다.")
        break
```

```
2016년 데이터는  368291 번째 행부터 시작입니다.
```

In [8]:

```
weather_data = weather_data.iloc[368291:]

weather_data = weather_data.reset_index(drop = True)

weather_data
```

Out[8]:

	Date Time	p (mbar)	T (degC)	Tpot (K)	Tdew (degC)	rh (%)	VPmax (mbar)	VPact (mbar)	VPdef (mbar)	sh (g/kg)	(mmo
0	01.01.2016 00:00:00	999.08	-0.01	273.22	-0.44	96.90	6.10	5.91	0.19	3.69	
1	01.01.2016 00:10:00	999.03	0.01	273.25	-0.41	97.00	6.11	5.93	0.18	3.70	
2	01.01.2016 00:20:00	999.07	0.06	273.29	-0.36	97.00	6.13	5.95	0.18	3.71	
3	01.01.2016 00:30:00	999.09	0.07	273.30	-0.36	96.90	6.14	5.95	0.19	3.71	
4	01.01.2016 00:40:00	999.09	-0.05	273.18	-0.50	96.80	6.09	5.89	0.19	3.68	
...	...	...	...	...	...	...	...	...	...	...	
52255	31.12.2016 23:20:00	1000.07	-4.05	269.10	-8.13	73.10	4.52	3.30	1.22	2.06	
52256	31.12.2016 23:30:00	999.93	-3.35	269.81	-8.06	69.71	4.77	3.32	1.44	2.07	
52257	31.12.2016 23:40:00	999.82	-3.16	270.01	-8.21	67.91	4.84	3.28	1.55	2.05	
52258	31.12.2016 23:50:00	999.81	-4.23	268.94	-8.53	71.80	4.46	3.20	1.26	1.99	
52259	01.01.2017 00:00:00	999.82	-4.82	268.36	-8.42	75.70	4.27	3.23	1.04	2.01	

52260 rows × 15 columns

In [9]:

```
weather_data = weather_data.iloc[:, 1:]

weather_data
```

Out[9]:

	p (mbar)	T (degC)	Tpot (K)	Tdew (degC)	rh (%)	VPmax (mbar)	VPact (mbar)	VPdef (mbar)	sh (g/kg)	H2OC (mmol/mol)	(g/m³
0	999.08	-0.01	273.22	-0.44	96.90	6.10	5.91	0.19	3.69	5.92	1271
1	999.03	0.01	273.25	-0.41	97.00	6.11	5.93	0.18	3.70	5.94	1271
2	999.07	0.06	273.29	-0.36	97.00	6.13	5.95	0.18	3.71	5.96	1270
3	999.09	0.07	273.30	-0.36	96.90	6.14	5.95	0.19	3.71	5.96	1270
4	999.09	-0.05	273.18	-0.50	96.80	6.09	5.89	0.19	3.68	5.90	1271
...	...	...	...	...	...	...	...	...	...	...	
52255	1000.07	-4.05	269.10	-8.13	73.10	4.52	3.30	1.22	2.06	3.30	1292
52256	999.93	-3.35	269.81	-8.06	69.71	4.77	3.32	1.44	2.07	3.32	1289
52257	999.82	-3.16	270.01	-8.21	67.91	4.84	3.28	1.55	2.05	3.28	1288
52258	999.81	-4.23	268.94	-8.53	71.80	4.46	3.20	1.26	1.99	3.20	1293
52259	999.82	-4.82	268.36	-8.42	75.70	4.27	3.23	1.04	2.01	3.23	1296

52260 rows × 14 columns

In [10]:

```
def extract_inputoutput(dataframe, lookback_time = 2, predict_time = 1):
    dfx = pd.DataFrame()
    dfy = pd.DataFrame()

    for i in range(len(dataframe) - (lookback_time - 1) - (predict_time)):

        if i % 10000 == 0:
            print(i)

        rowx = []

        for timestep in range(lookback_time):
            dfRename = dataframe.iloc[[i + timestep]]
            dfRename.index = [i]
            rowx.append(dfRename)

        rowx = pd.concat(rowx, axis = 1)
        dfx = pd.concat([dfx, rowx])

        rowx = []
        rowy = pd.DataFrame([dataframe["T (degC)"][i + lookback_time]])
        dfy = pd.concat([dfy, rowy], ignore_index = True)

    print("X, Y 데이터 분류 완료!")
    return dfx, dfy
```

In [11]:

```
x, t = extract_inputoutput(weather_data)
```

```
0
10000
20000
30000
40000
50000
X, Y 데이터 분류 완료!
```

In [12]:

```
x
```

Out[12]:

	p (mbar)	T (degC)	Tpot (K)	Tdew (degC)	rh (%)	VPmax (mbar)	VPact (mbar)	VPdef (mbar)	sh (g/kg)	H2OC (mmol/mol)	...	
0	999.08	-0.01	273.22	-0.44	96.90	6.10	5.91	0.19	3.69	5.92	...	9
1	999.03	0.01	273.25	-0.41	97.00	6.11	5.93	0.18	3.70	5.94	...	9
2	999.07	0.06	273.29	-0.36	97.00	6.13	5.95	0.18	3.71	5.96	...	9
3	999.09	0.07	273.30	-0.36	96.90	6.14	5.95	0.19	3.71	5.96	...	9
4	999.09	-0.05	273.18	-0.50	96.80	6.09	5.89	0.19	3.68	5.90	...	9
...	...	...	...	...	...	...	...	...	...	...	...	
52253	1000.21	-3.76	269.39	-7.95	72.50	4.62	3.35	1.27	2.09	3.35	...	7
52254	1000.11	-3.93	269.23	-8.09	72.60	4.56	3.31	1.25	2.06	3.31	...	7
52255	1000.07	-4.05	269.10	-8.13	73.10	4.52	3.30	1.22	2.06	3.30	...	6
52256	999.93	-3.35	269.81	-8.06	69.71	4.77	3.32	1.44	2.07	3.32	...	6
52257	999.82	-3.16	270.01	-8.21	67.91	4.84	3.28	1.55	2.05	3.28	...	7

52258 rows × 28 columns

In [13]:

```
t
```

Out[13]:

	0
0	0.06
1	0.07
2	-0.05
3	0.07
4	-0.05
...	...
52253	-4.05
52254	-3.35
52255	-3.16
52256	-4.23
52257	-4.82

52258 rows × 1 columns

In [14]:

```
x_train, x_test, t_train, t_test = train_test_split(x, t, test_size=0.2, shuffle = False)

print("x_train shape : ", x_train.shape)
print("t_train shape : ", t_train.shape)
print("x_test shape : ", x_test.shape)
print("t_test shape : ", t_test.shape)
```

```
x_train shape :  (41806, 28)
t_train shape :  (41806, 1)
x_test shape :  (10452, 28)
t_test shape :  (10452, 1)
```

In [15]:

```
timesteps = 2
feature = 14

x_train = np.array(x_train)
x_train = x_train.reshape(x_train.shape[0], timesteps, feature)

x_test = np.array(x_test)
x_test = x_test.reshape(x_test.shape[0], timesteps, feature)

t_train = np.array(t_train)
t_test = np.array(t_test)

print("reshape 후 x_train shape : ", x_train.shape)
print("t_train shape : ", t_train.shape)
print("reshape 후 x_test shape : ", x_test.shape)
print("t_test shape : ", t_test.shape)
```

```
reshape 후 x_train shape :  (41806, 2, 14)
t_train shape :  (41806, 1)
reshape 후 x_test shape :  (10452, 2, 14)
t_test shape :  (10452, 1)
```

In [16]:

```
cell_size = 128
timesteps = 2
feature = 14

model = Sequential(name = "Temp_LSTM")

model.add(LSTM(cell_size, input_shape = (timesteps, feature), return_sequences = True))
model.add(LSTM(cell_size))

model.add(Dense(1))

model.compile(loss = mse, optimizer = Adam(learning_rate=0.001), metrics = [RootMeanSquaredError()]

model.summary()
```

```
Model: "Temp_LSTM"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 2, 128)            73216

 lstm_1 (LSTM)               (None, 128)               131584

 dense (Dense)               (None, 1)                 129

=================================================================
Total params: 204,929
Trainable params: 204,929
Non-trainable params: 0
_________________________________________________________________
```

In [17]:

```
model.fit(x_train, t_train, epochs=20, batch_size = 32)
```

```
Epoch 1/20
1307/1307 [==============================] - 24s 13ms/step - loss: 10.8442 - root_me
an_squared_error: 3.2930
Epoch 2/20
1307/1307 [==============================] - 16s 13ms/step - loss: 1.1526 - root_mea
n_squared_error: 1.0736
Epoch 3/20
1307/1307 [==============================] - 17s 13ms/step - loss: 1.0211 - root_mea
n_squared_error: 1.0105
Epoch 4/20
1307/1307 [==============================] - 17s 13ms/step - loss: 0.8664 - root_mea
n_squared_error: 0.9308
Epoch 5/20
1307/1307 [==============================] - 17s 13ms/step - loss: 0.7837 - root_mea
n_squared_error: 0.8853
Epoch 6/20
1307/1307 [==============================] - 17s 13ms/step - loss: 0.7308 - root_mea
n_squared_error: 0.8549
Epoch 7/20
1307/1307 [==============================] - 17s 13ms/step - loss: 0.6293 - root_mea
n_squared_error: 0.7933
Epoch 8/20
1307/1307 [==============================] - 17s 13ms/step - loss: 0.6107 - root_mea
n_squared_error: 0.7815
Epoch 9/20
1307/1307 [==============================] - 17s 13ms/step - loss: 0.6499 - root_mea
n_squared_error: 0.8062
Epoch 10/20
1307/1307 [==============================] - 17s 13ms/step - loss: 0.5708 - root_mea
n_squared_error: 0.7555
Epoch 11/20
1307/1307 [==============================] - 17s 13ms/step - loss: 0.5601 - root_mea
n_squared_error: 0.7484
Epoch 12/20
1307/1307 [==============================] - 17s 13ms/step - loss: 0.5073 - root_mea
n_squared_error: 0.7122
Epoch 13/20
1307/1307 [==============================] - 17s 13ms/step - loss: 0.4964 - root_mea
n_squared_error: 0.7046
Epoch 14/20
1307/1307 [==============================] - 18s 14ms/step - loss: 0.5218 - root_mea
n_squared_error: 0.7224
Epoch 15/20
1307/1307 [==============================] - 19s 14ms/step - loss: 0.4651 - root_mea
n_squared_error: 0.6820
Epoch 16/20
1307/1307 [==============================] - 19s 14ms/step - loss: 0.5303 - root_mea
n_squared_error: 0.7282
Epoch 17/20
1307/1307 [==============================] - 19s 14ms/step - loss: 0.5475 - root_mea
n_squared_error: 0.7399
Epoch 18/20
1307/1307 [==============================] - 19s 14ms/step - loss: 0.5037 - root_mea
n_squared_error: 0.7097
Epoch 19/20
1307/1307 [==============================] - 18s 14ms/step - loss: 0.4766 - root_mea
n_squared_error: 0.6904
Epoch 20/20
1307/1307 [==============================] - 18s 14ms/step - loss: 0.4760 - root_mea
n_squared_error: 0.6899
```

Out[17]:

<keras.callbacks.History at 0x21584bdb3d0>

In [18]:

```
pred = model.predict(x_test)

for i in range(1, 10):
    print("온도(T(deg C)) 예측 : ", round(pred[i][0], 2), "/ 정답 : ", round(t_test[i][0], 2))
```

```
327/327 [==============================] - 4s 7ms/step
온도(T(deg C)) 예측 :  6.14 / 정답 :  5.94
온도(T(deg C)) 예측 :  6.17 / 정답 :  6.16
온도(T(deg C)) 예측 :  6.22 / 정답 :  6.43
온도(T(deg C)) 예측 :  6.36 / 정답 :  6.58
온도(T(deg C)) 예측 :  6.47 / 정답 :  6.93
온도(T(deg C)) 예측 :  6.62 / 정답 :  7.21
온도(T(deg C)) 예측 :  6.83 / 정답 :  7.38
온도(T(deg C)) 예측 :  7.01 / 정답 :  7.56
온도(T(deg C)) 예측 :  7.12 / 정답 :  8.14
```

In [19]:

```
loss, rmse = model.evaluate(x_test, t_test, verbose = 1)

print("test loss (MSE) : ", round(loss, 6))
print("test RMSE : ", round(rmse, 2))
```

```
327/327 [==============================] - 4s 7ms/step - loss: 0.1818 - root_mean_sq
uared_error: 0.4263
test loss (MSE) :  0.181767
test RMSE :  0.43
```