In [1]:

```
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]:

```
total_seoul_data = pd.read_csv("./Total_Data_Seoul.csv", engine = "python")

total_seoul_data
```

Out[2]:

	측정일시	SO2	CO	O3	NO2	PM10	Seoul_Temp(°C)	Seoul_Precipitation(mm)	Seoul_
0	2020-01-01 01:00:00	0.002	0.5	0.011	0.024	19.0	-5.9	1.556	
1	2020-01-01 02:00:00	0.002	0.6	0.005	0.030	19.0	-5.7	1.556	
2	2020-01-01 03:00:00	0.002	0.6	0.002	0.033	27.0	-5.6	0.000	
3	2020-01-01 04:00:00	0.002	0.6	0.003	0.031	20.0	-5.4	1.556	
4	2020-01-01 05:00:00	0.002	0.7	0.003	0.031	21.0	-5.2	1.556	
...	...	...	...	...	...	...	...	...	
8778	2020-12-31 19:00:00	0.002	0.4	0.016	0.023	26.0	-7.1	1.556	
8779	2020-12-31 20:00:00	0.002	0.4	0.014	0.026	29.0	-7.1	1.556	
8780	2020-12-31 21:00:00	0.002	0.4	0.017	0.021	23.0	-7.2	1.556	
8781	2020-12-31 22:00:00	0.002	0.4	0.025	0.013	28.0	-7.4	1.556	
8782	2020-12-31 23:00:00	0.002	0.3	0.030	0.008	24.0	-7.6	1.556	

8783 rows × 11 columns

In [3]:

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

total_seoul_data
```

Out[3]:

	SO2	CO	O3	NO2	PM10	Seoul_Temp(°C)	Seoul_Precipitation(mm)	Seoul_Wind_Spe
0	0.002	0.5	0.011	0.024	19.0	-5.9	1.556	
1	0.002	0.6	0.005	0.030	19.0	-5.7	1.556	
2	0.002	0.6	0.002	0.033	27.0	-5.6	0.000	
3	0.002	0.6	0.003	0.031	20.0	-5.4	1.556	
4	0.002	0.7	0.003	0.031	21.0	-5.2	1.556	
...	...	...	...	...	...	...	...	
8778	0.002	0.4	0.016	0.023	26.0	-7.1	1.556	
8779	0.002	0.4	0.014	0.026	29.0	-7.1	1.556	
8780	0.002	0.4	0.017	0.021	23.0	-7.2	1.556	
8781	0.002	0.4	0.025	0.013	28.0	-7.4	1.556	
8782	0.002	0.3	0.030	0.008	24.0	-7.6	1.556	

8783 rows × 10 columns

In [13]:

```
def extract_inputoutput(dataframe, lookback_time = 3, predict_time = 1):
    dfx = pd.DataFrame()
    dfy = pd.DataFrame()

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

        if i % 1000 == 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["PM10"][i + lookback_time]])
        dfy = pd.concat([dfy, rowy], ignore_index = True)

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

In [14]:

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

```
0
1000
2000
3000
4000
5000
6000
7000
8000
X, Y 데이터 분류 완료!
```

In [15]:

```
x
```

Out[15]:

	SO2	CO	O3	NO2	PM10	Seoul_Temp(°C)	Seoul_Precipitation(mm)	Seoul_Wind_Spe
0	0.002	0.5	0.011	0.024	19.0	-5.9	1.556	
1	0.002	0.6	0.005	0.030	19.0	-5.7	1.556	
2	0.002	0.6	0.002	0.033	27.0	-5.6	0.000	
3	0.002	0.6	0.003	0.031	20.0	-5.4	1.556	
4	0.002	0.7	0.003	0.031	21.0	-5.2	1.556	
...	...	...	...	...	...	...	...	
8775	0.002	0.3	0.031	0.008	20.0	-5.5	1.556	
8776	0.002	0.3	0.029	0.010	24.0	-6.1	1.556	
8777	0.002	0.3	0.027	0.011	24.0	-6.7	0.000	
8778	0.002	0.4	0.016	0.023	26.0	-7.1	1.556	
8779	0.002	0.4	0.014	0.026	29.0	-7.1	1.556	

8780 rows × 30 columns

In [16]:

```
t
```

Out[16]:

	0
0	20.0
1	21.0
2	23.0
3	22.0
4	21.0
...	...
8775	26.0
8776	29.0
8777	23.0
8778	28.0
8779	24.0

8780 rows × 1 columns

In [17]:

```
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 :  (7024, 30)
t_train shape :  (7024, 1)
x_test shape :  (1756, 30)
t_test shape :  (1756, 1)
```

In [18]:

```
timesteps = 3
feature = 10

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 :  (7024, 3, 10)
t_train shape :  (7024, 1)
reshape 후 x_test shape :  (1756, 3, 10)
t_test shape :  (1756, 1)
```

In [21]:

```
cell_size = 128
timesteps = 3
feature = 10

model = Sequential(name = "Airpollution_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: "Airpollution_LSTM"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 3, 128)            71168

 lstm_1 (LSTM)               (None, 128)               131584

 dense (Dense)               (None, 1)                 129

=================================================================
Total params: 202,881
Trainable params: 202,881
Non-trainable params: 0
_________________________________________________________________
```

In [22]:

```
model.fit(x_train, t_train, epochs=50, batch_size = 32)
```

```
Epoch 1/50
220/220 [==============================] - 5s 7ms/step - loss: 466.5694 - root_mean_
squared_error: 21.6002
Epoch 2/50
220/220 [==============================] - 2s 8ms/step - loss: 173.8214 - root_mean_
squared_error: 13.1841
Epoch 3/50
220/220 [==============================] - 2s 7ms/step - loss: 121.6646 - root_mean_
squared_error: 11.0302
Epoch 4/50
220/220 [==============================] - 2s 8ms/step - loss: 100.5315 - root_mean_
squared_error: 10.0265
Epoch 5/50
220/220 [==============================] - 2s 8ms/step - loss: 89.7378 - root_mean_s
quared_error: 9.4730
Epoch 6/50
220/220 [==============================] - 2s 8ms/step - loss: 81.6670 - root_mean_s
quared_error: 9.0370
Epoch 7/50
220/220 [==============================] - 2s 10ms/step - loss: 76.9656 - root_mean_
squared_error: 8.7730
Epoch 8/50
220/220 [==============================] - 2s 10ms/step - loss: 72.2548 - root_mean_
squared_error: 8.5003
Epoch 9/50
220/220 [==============================] - 2s 10ms/step - loss: 69.6578 - root_mean_
squared_error: 8.3461
Epoch 10/50
220/220 [==============================] - 2s 10ms/step - loss: 66.8509 - root_mean_
squared_error: 8.1762
Epoch 11/50
220/220 [==============================] - 2s 11ms/step - loss: 64.3068 - root_mean_
squared_error: 8.0192
Epoch 12/50
220/220 [==============================] - 2s 10ms/step - loss: 62.0702 - root_mean_
squared_error: 7.8785
Epoch 13/50
220/220 [==============================] - 2s 9ms/step - loss: 60.1310 - root_mean_s
quared_error: 7.7544
Epoch 14/50
220/220 [==============================] - 2s 9ms/step - loss: 58.7601 - root_mean_s
quared_error: 7.6655
Epoch 15/50
220/220 [==============================] - 2s 9ms/step - loss: 57.7093 - root_mean_s
quared_error: 7.5967
Epoch 16/50
220/220 [==============================] - 2s 9ms/step - loss: 56.6996 - root_mean_s
quared_error: 7.5299
Epoch 17/50
220/220 [==============================] - 2s 9ms/step - loss: 54.3914 - root_mean_s
quared_error: 7.3751
Epoch 18/50
220/220 [==============================] - 2s 9ms/step - loss: 53.4576 - root_mean_s
quared_error: 7.3115
Epoch 19/50
220/220 [==============================] - 2s 9ms/step - loss: 52.3788 - root_mean_s
quared_error: 7.2373
Epoch 20/50
220/220 [==============================] - 2s 9ms/step - loss: 51.6317 - root_mean_s
quared_error: 7.1855
Epoch 21/50
```

```
220/220 [==============================] - 2s 9ms/step - loss: 50.7672 - root_mean_s
quared_error: 7.1251
Epoch 22/50
220/220 [==============================] - 2s 9ms/step - loss: 49.8035 - root_mean_s
quared_error: 7.0572
Epoch 23/50
220/220 [==============================] - 2s 9ms/step - loss: 48.4841 - root_mean_s
quared_error: 6.9630
Epoch 24/50
220/220 [==============================] - 2s 9ms/step - loss: 48.2438 - root_mean_s
quared_error: 6.9458
Epoch 25/50
220/220 [==============================] - 2s 9ms/step - loss: 47.0145 - root_mean_s
quared_error: 6.8567
Epoch 26/50
220/220 [==============================] - 3s 11ms/step - loss: 46.7781 - root_mean_
squared_error: 6.8395
Epoch 27/50
220/220 [==============================] - 2s 11ms/step - loss: 45.9979 - root_mean_
squared_error: 6.7822
Epoch 28/50
220/220 [==============================] - 3s 13ms/step - loss: 45.8148 - root_mean_
squared_error: 6.7687
Epoch 29/50
220/220 [==============================] - 3s 12ms/step - loss: 44.9345 - root_mean_
squared_error: 6.7033
Epoch 30/50
220/220 [==============================] - 3s 14ms/step - loss: 43.6488 - root_mean_
squared_error: 6.6067
Epoch 31/50
220/220 [==============================] - 3s 12ms/step - loss: 42.4937 - root_mean_
squared_error: 6.5187
Epoch 32/50
220/220 [==============================] - 3s 12ms/step - loss: 43.4613 - root_mean_
squared_error: 6.5925
Epoch 33/50
220/220 [==============================] - 2s 11ms/step - loss: 43.6707 - root_mean_
squared_error: 6.6084
Epoch 34/50
220/220 [==============================] - 2s 10ms/step - loss: 42.1462 - root_mean_
squared_error: 6.4920
Epoch 35/50
220/220 [==============================] - 3s 12ms/step - loss: 41.7496 - root_mean_
squared_error: 6.4614
Epoch 36/50
220/220 [==============================] - 3s 12ms/step - loss: 41.9993 - root_mean_
squared_error: 6.4807
Epoch 37/50
220/220 [==============================] - 3s 13ms/step - loss: 40.8436 - root_mean_
squared_error: 6.3909
Epoch 38/50
220/220 [==============================] - 2s 11ms/step - loss: 39.9990 - root_mean_
squared_error: 6.3245
Epoch 39/50
220/220 [==============================] - 3s 11ms/step - loss: 40.2268 - root_mean_
squared_error: 6.3425
Epoch 40/50
220/220 [==============================] - 2s 11ms/step - loss: 40.2728 - root_mean_
squared_error: 6.3461
Epoch 41/50
220/220 [==============================] - 2s 10ms/step - loss: 39.5220 - root_mean_
```

```
squared_error: 6.2867
Epoch 42/50
220/220 [==============================] - 2s 10ms/step - loss: 39.4626 - root_mean_
squared_error: 6.2819
Epoch 43/50
220/220 [==============================] - 2s 10ms/step - loss: 39.5821 - root_mean_
squared_error: 6.2914
Epoch 44/50
220/220 [==============================] - 2s 10ms/step - loss: 39.2123 - root_mean_
squared_error: 6.2620
Epoch 45/50
220/220 [==============================] - 2s 11ms/step - loss: 38.6658 - root_mean_
squared_error: 6.2182
Epoch 46/50
220/220 [==============================] - 2s 10ms/step - loss: 38.9896 - root_mean_
squared_error: 6.2442
Epoch 47/50
220/220 [==============================] - 2s 10ms/step - loss: 37.8260 - root_mean_
squared_error: 6.1503
Epoch 48/50
220/220 [==============================] - 2s 10ms/step - loss: 37.6471 - root_mean_
squared_error: 6.1357
Epoch 49/50
220/220 [==============================] - 2s 10ms/step - loss: 37.8318 - root_mean_
squared_error: 6.1508
Epoch 50/50
220/220 [==============================] - 2s 10ms/step - loss: 38.3088 - root_mean_
squared_error: 6.1894
Out[22]:

<keras.callbacks.History at 0x20f336c9f40>


In [23]:
```

```
pred = model.predict(x_test)

for i in range(1, 10):
    print("PM10 예측 : ", round(pred[i][0], 2), "/ 정답 : ", round(t_test[i][0], 2))
```

```
55/55 [==============================] - 1s 4ms/step
PM10 예측 :  73.3 / 정답 :  75.0
PM10 예측 :  75.58 / 정답 :  70.0
PM10 예측 :  71.48 / 정답 :  74.0
PM10 예측 :  80.84 / 정답 :  65.0
PM10 예측 :  65.59 / 정답 :  59.0
PM10 예측 :  60.93 / 정답 :  56.0
PM10 예측 :  57.16 / 정답 :  52.0
PM10 예측 :  52.53 / 정답 :  53.0
PM10 예측 :  53.39 / 정답 :  51.0
```

In [24]:

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

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

```
55/55 [==============================] - 1s 3ms/step - loss: 77.7209 - root_mean_squ
ared_error: 8.8159
test loss (MSE) :  77.720871
test RMSE :  8.82
```

In [ ]: