Predicting temperature with tensorflow

기상청 자료로 온도 예측하기

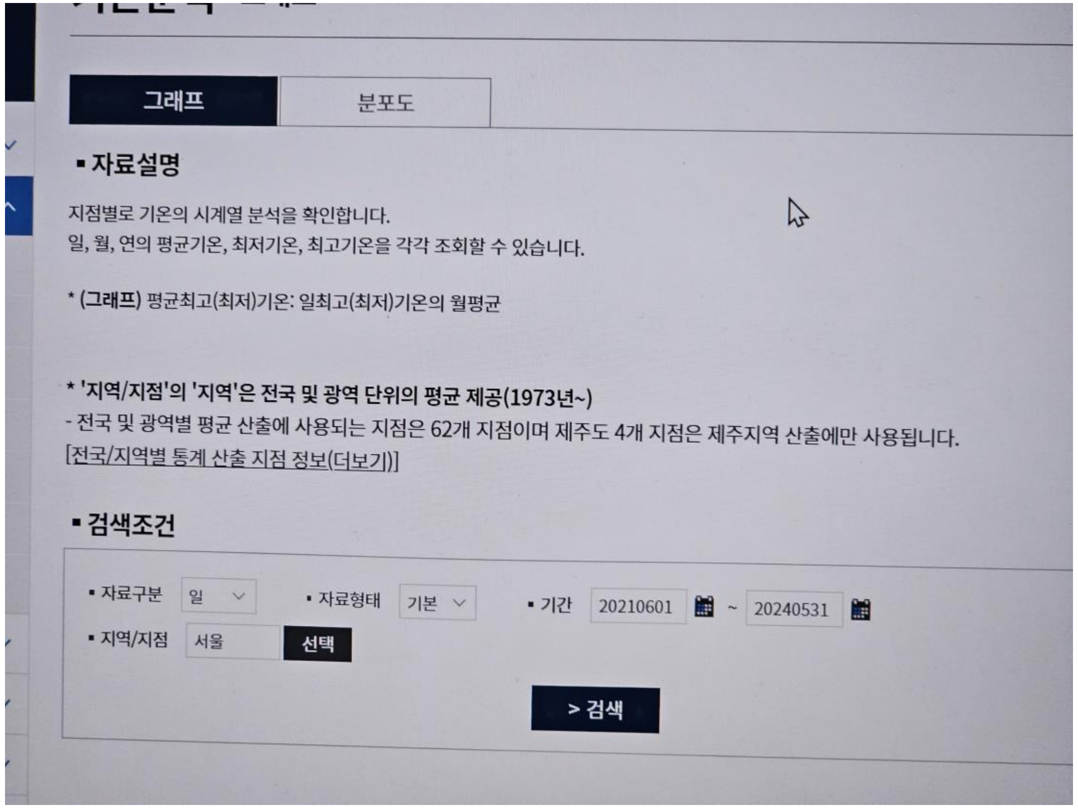
2023042124_김민지

Contents

- Source of data
- Deep learning model: LSTM
- My source code
- Problem

Source of data

- KMA(기상청)
- https://data.kma.go.kr/stcs/grnd/grndTaList.do?pgmNo=70 (기상청 기상자료개방포털)
- 2021-06-01 ~ 2024-05-31



▪ 자료설명

지점별로 기온의 시계열 분석을 확인합니다.
일, 월, 연의 평균기온, 최저기온, 최고기온을 각각 조회할 수 있습니다.

* **(그래프)** 평균최고(최저)기온: 일최고(최저)기온의 월평균

*** '지역/지점'의 '지역'은 전국 및 광역 단위의 평균 제공(1973년~)**
- 전국 및 광역별 평균 산출에 사용되는 지점은 62개 지점이며 제주도 4개 지점은 제주지역 산출에만 사용됩니다.
[전국/지역별 통계 산출 지점 정보(더보기)]

▪ 검색조건

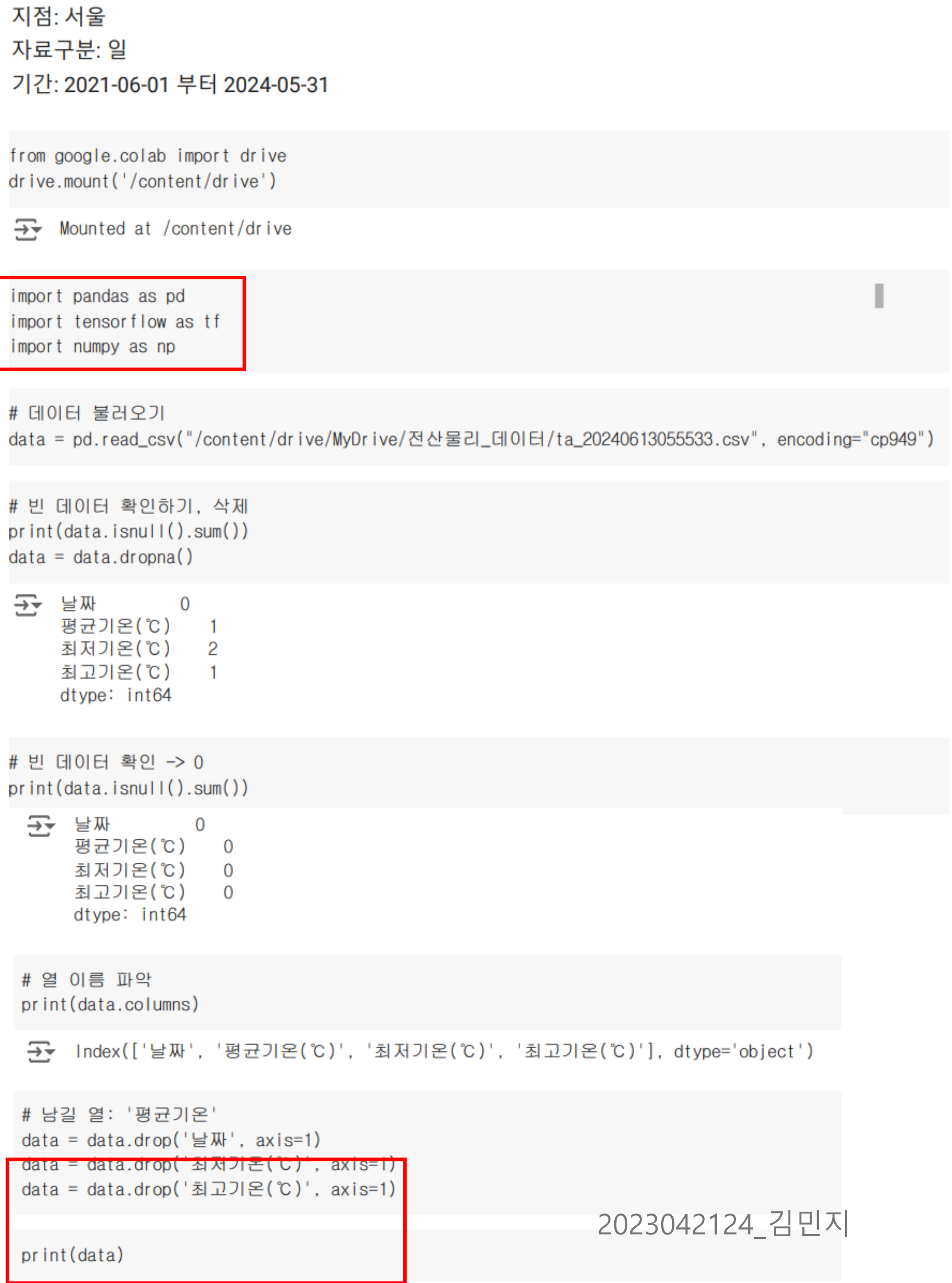
▪ 자료구분 일 ▪ 자료형태 기본 ▪ 기간 20210601 ~ 20240531
▪ 지역/지점 서울 선택

deep learning model: LSTM

- LSTM(Long short-term memory) is a type of recurrent neural network(RNN) aimed at dealing with the vanishing gradient problem present in RNN.

- LSTM is used to analyze data placed at regular intervals (time series).

source code



지점: 서울

자료구분: 일

기간: 2021-06-01 부터 2024-05-31

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import pandas as pd
import tensorflow as tf
import numpy as np
```

```
# 데이터 불러오기
data = pd.read_csv("/content/drive/MyDrive/전산물리_데이터/ta_20240613055533.csv", encoding="cp949")
```

```
# 빈 데이터 확인하기, 삭제
print(data.isnull().sum())
data = data.dropna()
```

```
날짜          0
평균기온(℃)    1
최저기온(℃)    2
최고기온(℃)    1
dtype: int64
```

```
# 빈 데이터 확인 -> 0
print(data.isnull().sum())
```

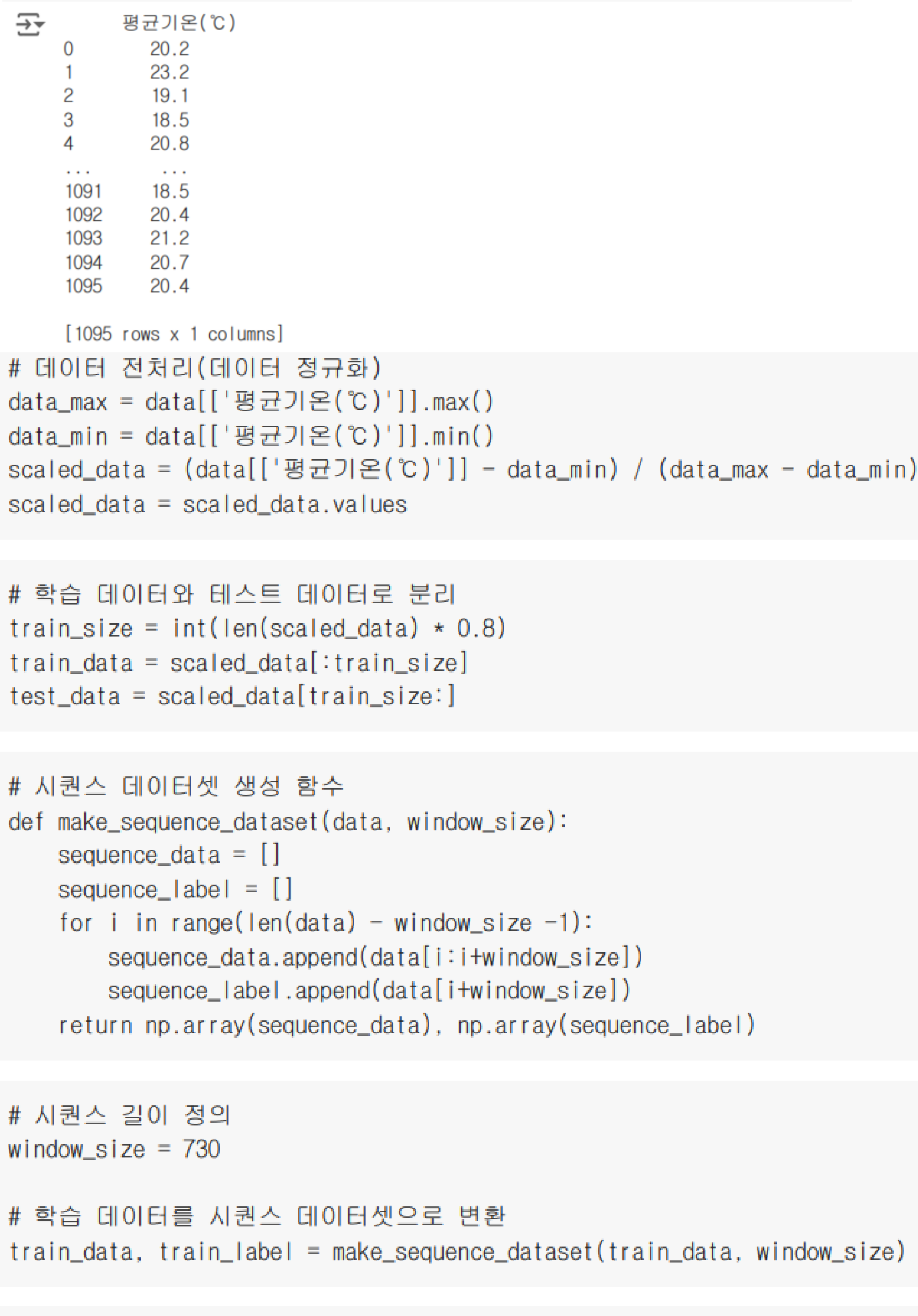
```
날짜          0
평균기온(℃)    0
최저기온(℃)    0
최고기온(℃)    0
dtype: int64
```

```
# 열 이름 파악
print(data.columns)
```

```
Index(['날짜', '평균기온(℃)', '최저기온(℃)', '최고기온(℃)'], dtype='object')
```

```
# 남길 열: '평균기온'
data = data.drop('날짜', axis=1)
data = data.drop('최저기온(℃)', axis=1)
data = data.drop('최고기온(℃)', axis=1)

print(data)
```

use '평균 기온'



```
      평균기온(℃)
0         20.2
1         23.2
2         19.1
3         18.5
4         20.8
...        ...
1091      18.5
1092      20.4
1093      21.2
1094      20.7
1095      20.4

[1095 rows x 1 columns]
```

```
# 데이터 전처리(데이터 정규화)
data_max = data[['평균기온(℃)']].max()
data_min = data[['평균기온(℃)']].min()
scaled_data = (data[['평균기온(℃)']] - data_min) / (data_max - data_min)
scaled_data = scaled_data.values
```

```
# 학습 데이터와 테스트 데이터로 분리
train_size = int(len(scaled_data) * 0.8)
train_data = scaled_data[:train_size]
test_data = scaled_data[train_size:]
```

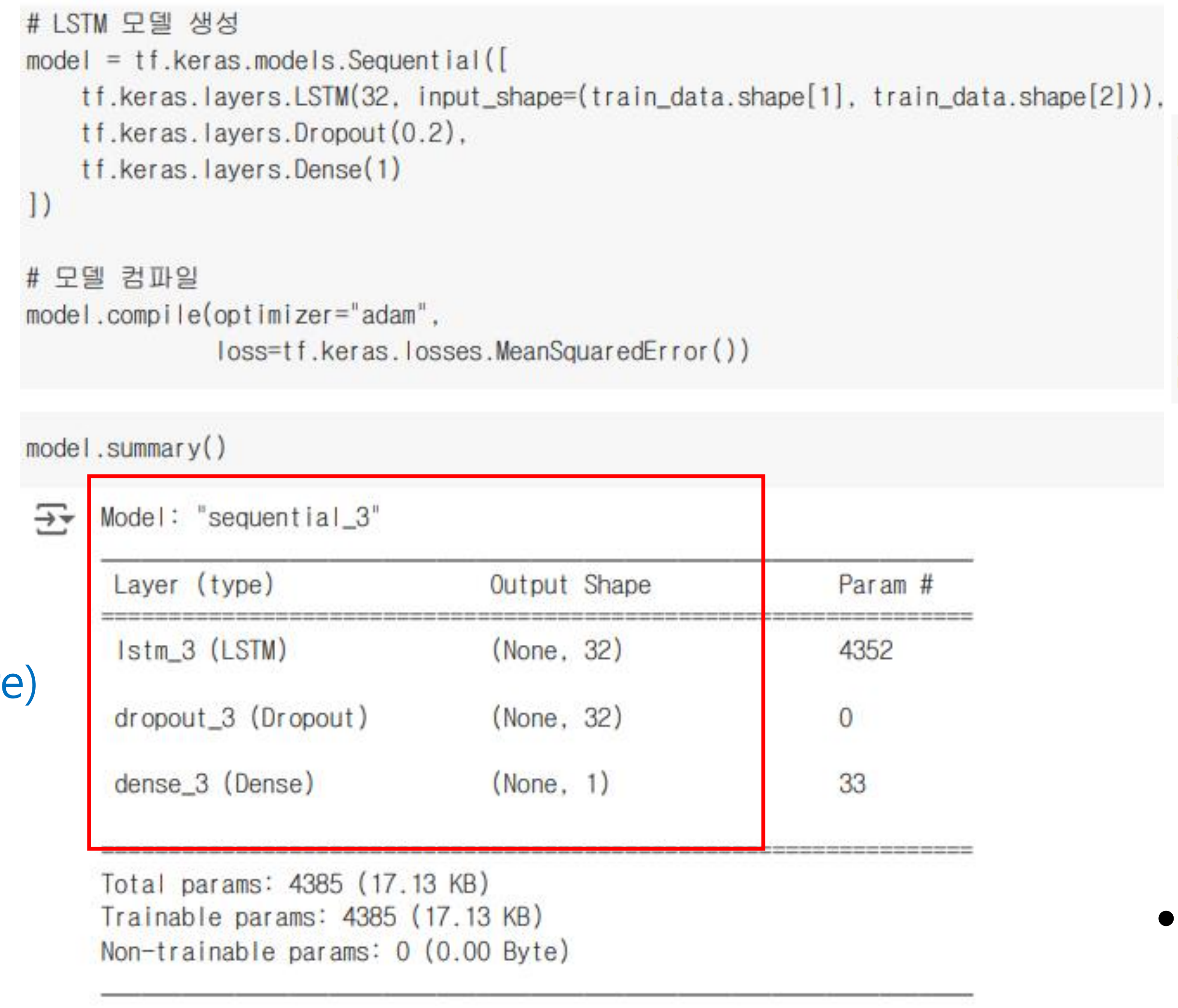
```
# 시퀀스 데이터셋 생성 함수
def make_sequence_dataset(data, window_size):
    sequence_data = []
    sequence_label = []
    for i in range(len(data) - window_size -1):
        sequence_data.append(data[i:i+window_size])
        sequence_label.append(data[i+window_size])
    return np.array(sequence_data), np.array(sequence_label)
```

```
# 시퀀스 길이 정의
window_size = 730

# 학습 데이터를 시퀀스 데이터셋으로 변환
train_data, train_label = make_sequence_dataset(train_data, window_size)
```

2023042124_김민지

source code

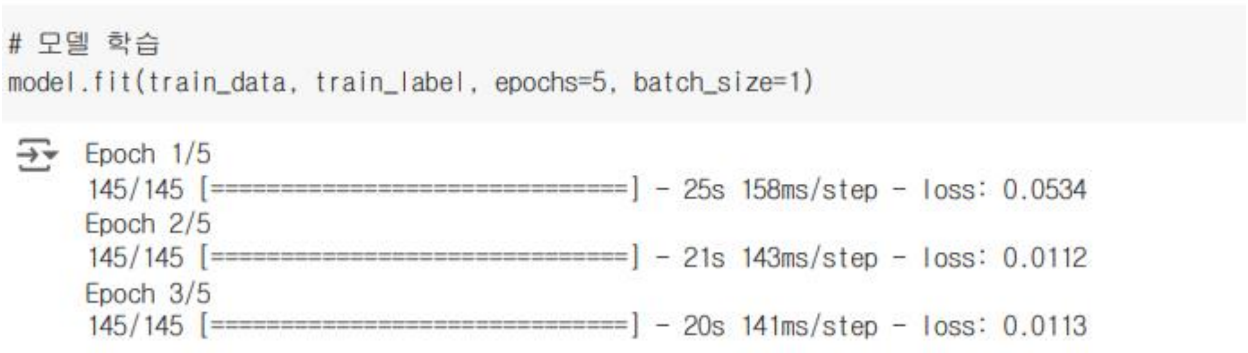
- LSTM
- Dropout
- Dence -> 1 (temperature)



```
# LSTM 모델 생성
model = tf.keras.models.Sequential([
    tf.keras.layers.LSTM(32, input_shape=(train_data.shape[1], train_data.shape[2])),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(1)
])

# 모델 컴파일
model.compile(optimizer="adam",
              loss=tf.keras.losses.MeanSquaredError())
```

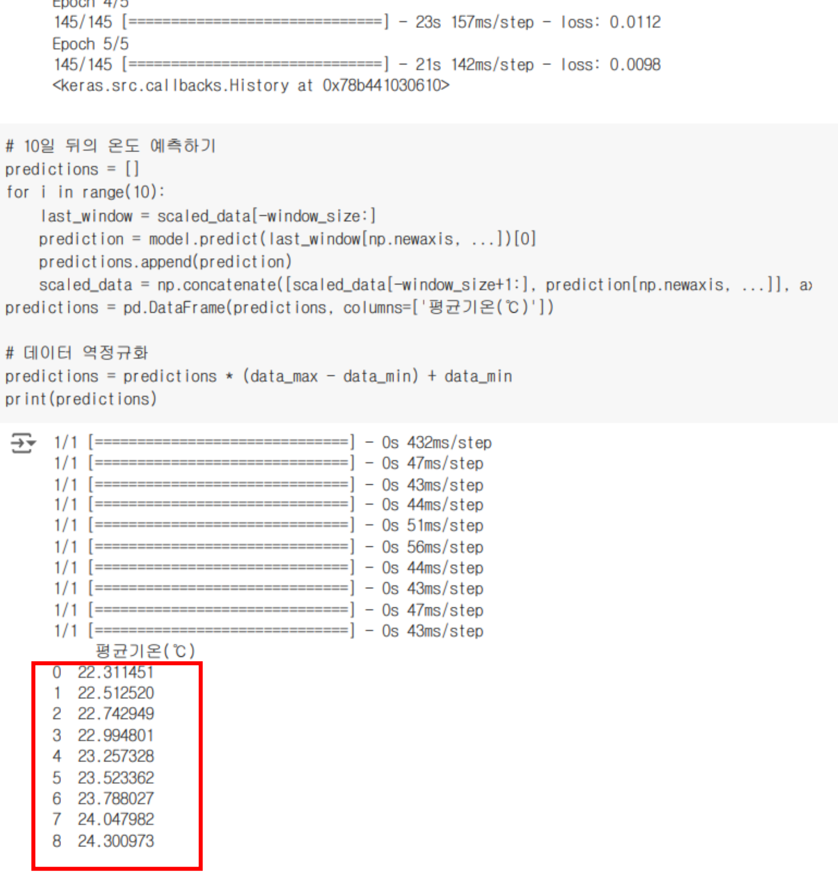
```
model.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_3 (LSTM)               (None, 32)                4352

 dropout_3 (Dropout)         (None, 32)                0

 dense_3 (Dense)             (None, 1)                 33

=================================================================
Total params: 4385 (17.13 KB)
Trainable params: 4385 (17.13 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```



```
# 모델 학습
model.fit(train_data, train_label, epochs=5, batch_size=1)
```

```
Epoch 1/5
145/145 [==============================] - 25s 158ms/step - loss: 0.0534
Epoch 2/5
145/145 [==============================] - 21s 143ms/step - loss: 0.0112
Epoch 3/5
145/145 [==============================] - 20s 141ms/step - loss: 0.0113
```



```
Epoch 4/5
145/145 [==============================] - 23s 157ms/step - loss: 0.0112
Epoch 5/5
145/145 [==============================] - 21s 142ms/step - loss: 0.0098
<keras.src.callbacks.History at 0x78b441030610>
```

```
# 10일 뒤의 온도 예측하기
predictions = []
for i in range(10):
    last_window = scaled_data[-window_size:]
    prediction = model.predict(last_window[np.newaxis, ...])[0]
    predictions.append(prediction)
    scaled_data = np.concatenate([scaled_data[-window_size+1:], prediction[np.newaxis, ...]], a
predictions = pd.DataFrame(predictions, columns=['평균기온(℃)'])

# 데이터 역정규화
predictions = predictions * (data_max - data_min) + data_min
print(predictions)
```

```
1/1 [==============================] - 0s 432ms/step
1/1 [==============================] - 0s 47ms/step
1/1 [==============================] - 0s 43ms/step
1/1 [==============================] - 0s 44ms/step
1/1 [==============================] - 0s 51ms/step
1/1 [==============================] - 0s 56ms/step
1/1 [==============================] - 0s 44ms/step
1/1 [==============================] - 0s 43ms/step
1/1 [==============================] - 0s 47ms/step
1/1 [==============================] - 0s 43ms/step
     평균기온(℃)
0  22.311451
1  22.512520
2  22.742949
3  22.994801
4  23.257328
5  23.523362
6  23.788027
7  24.047982
8  24.300973
```

- Problems

Thank you for listening

https://ko.wikipedia.org/wiki/%EC%8B%9C%EA%B3%84%EC%97%B4

https://en.wikipedia.org/wiki/Long_short-term_memory

https://ko.wikipedia.org/wiki/%EC%88%9C%ED%99%98_%EC%8B%A0%EA%B2%BD%EB%A7%9D

https://rubber-tree.tistory.com/115