LSTM을 활용한 우리동네 미세먼지 농도 예측

2ternals 팀장 : 손예린
팀원 : 김창현, 권희정, 이우섭, 이해성

발표일자 : 2022/ 01/14

Contents

01. Introduction
02. Process
03. Code & Results
04. Problem & Solution
05. Reviews

Introduction

LSTM을 활용한 우리동네 미세먼지 농도 예측

- Team Members

  2ternals
  팀장 : 손예린
  팀원 : 김창현, 권희정, 이우섭, 이해성

LSTM을 활용한 우리동네 미세먼지 농도 예측

기상데이터(미세먼지, 초미세먼지)를 이용한
LSTM 모델 기반의 서울시 미세먼지 농도 예측 프로그램

- Work Schedule

2022/01/07	주제선정
2022/01/10	데이터 수집
2022/01/11 ~ 2022/01/12	개발
2022/01/13	수정 및 보완
2022/01/14	발표

- Skills

Numpy

Pandas

Tensorflow

Matplotlib

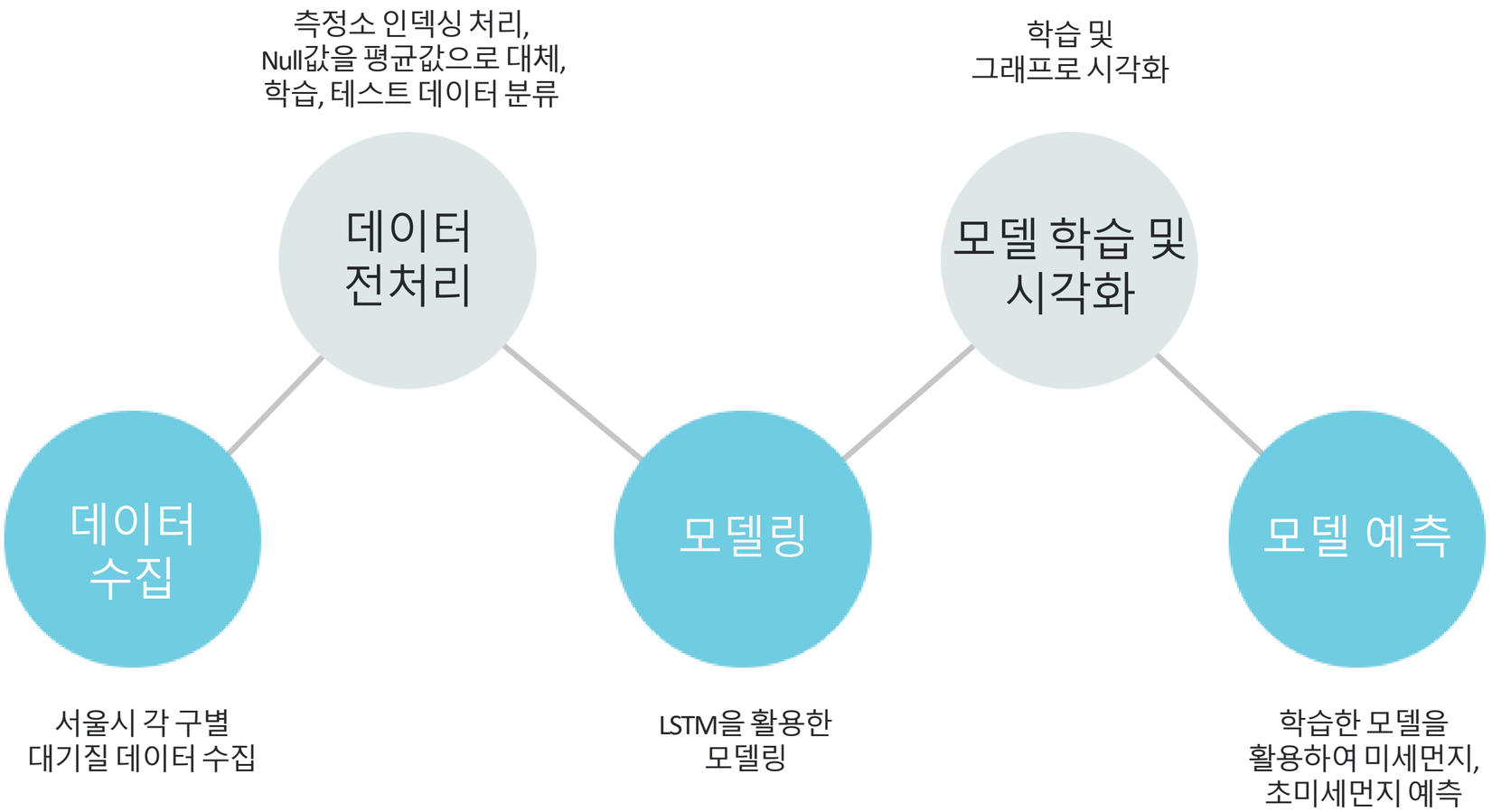
- Work Dataset & Rule

기상 데이터 : 에어코리아, 케이웨더, 서울시 공공데이터
LSTM 모델 기반

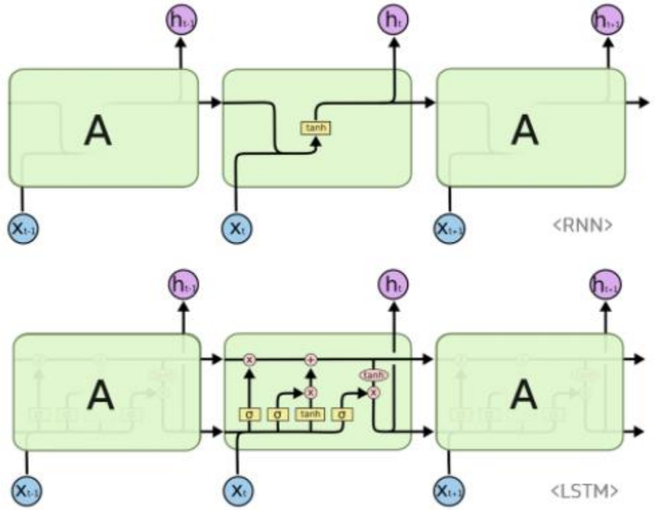
- Roles and Responsibilities

데이터 수집	데이터 전처리	모델 구축 및 예측	파라미터 튜닝	시각화	발표
김창현, 권희정	김창현	전원	손예린, 이우섭	손예린, 이해성	권희정

Process

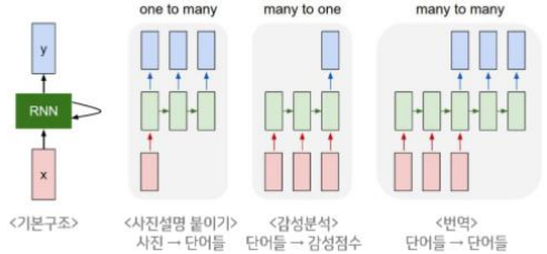

데이터 수집
서울시 각 구별
대기질 데이터 수집
데이터 전처리
측정소 인덱싱 처리,
Null값을 평균값으로 대체,
학습, 테스트 데이터 분류
모델링
LSTM을 활용한
모델링
모델 학습 및 시각화
학습 및
그래프로 시각화
모델 예측
학습한 모델을
활용하여 미세먼지,
초미세먼지 예측

Process – Modeling



RNN

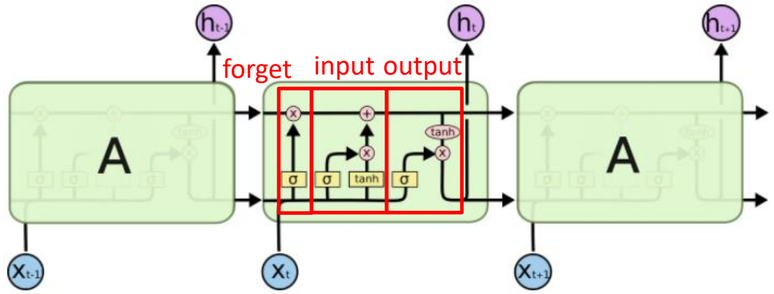
Recurrent Neural Networks



- 과거의 데이터가 미래에 영향을 줌
- 역전파 과정에서 기울기 손실 발생
- 장기 의존성 문제점 발생

LSTM

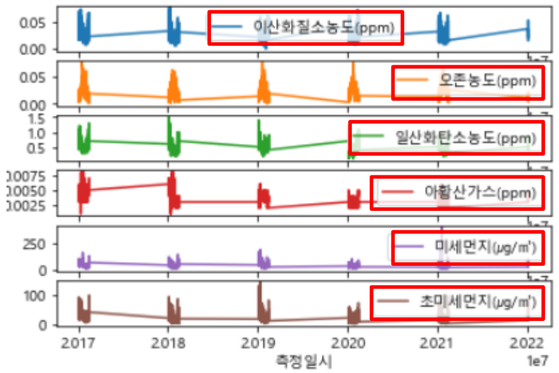
Long Short-Term Memory



- 신경망 사이의 뉴런이 재귀하는 구조
- 현재 시점의 정보를 바탕으로 과거 내용의 보존 여부 계산, 그 결과에 현재 정보 추가 후 다음 시점으로 전달
- 시간 흐름에 따라 연속성을 갖는 시계열 데이터를 이용한 예측 모델 학습에 강점을 가짐

Code & Results

```
import platform
if platform.system() == 'Darwin': #맥
        plt.rc('font', family='AppleGothic')
elif platform.system() == 'Windows': #윈도우
        plt.rc('font', family='Malgun Gothic')
elif platform.system() == 'Linux': #리눅스 (코랩)
        plt.rc('font', family='Malgun Gothic')
plt.rcParams['axes.unicode_minus'] = False #한글 폰트 사용시 마이너스 폰트 깨짐 해결
```



Code & Results

```
df['이산화질소농도(ppm)'] = df['이산화질소농도(ppm)'].fillna(df['이산화질소농도(ppm)'].mean()).astype(float)
df['오존농도(ppm)'] = df['오존농도(ppm)'].fillna(df['오존농도(ppm)'].mean()).astype(float)
df['일산화탄소농도(ppm)'] = df['일산화탄소농도(ppm)'].fillna(df['일산화탄소농도(ppm)'].mean()).astype(float)
df['아황산가스(ppm)'] = df['아황산가스(ppm)'].fillna(df['아황산가스(ppm)'].mean()).astype(float)
df['미세먼지(㎍/㎥)'] = df['미세먼지(㎍/㎥)'].fillna(df['미세먼지(㎍/㎥)'].mean()).astype(float)
df['초미세먼지(㎍/㎥)'] = df['초미세먼지(㎍/㎥)'].fillna(df['초미세먼지(㎍/㎥)'].mean()).astype(float)
```

결측치를 평균값으로 대체

```
측정일시                0
측정소명                0
이산화질소농도(ppm)     1201
오존농도(ppm)         1103
일산화탄소농도(ppm)     1270
아황산가스(ppm)        1187
미세먼지(㎍/㎥)         1651
초미세먼지(㎍/㎥)        2177
dtype: int64
```

```
측정일시                0
측정소명                0
이산화질소농도(ppm)      0
오존농도(ppm)         0
일산화탄소농도(ppm)      0
아황산가스(ppm)         0
미세먼지(㎍/㎥)          0
초미세먼지(㎍/㎥)         0
dtype: int64
```

Code & Results

```python
def load_time_series_data(data, sequence_length):
    window_length = sequence_length + 1
    x_data = []
    y_data = []
    for i in range(0, len(data) - window_length + 1):
        window = data[i:i + window_length, :]
        x_data.append(window[:-1, :])
        y_data.append(window[-1, [-1]])
    x_data = np.array(x_data)
    y_data = np.array(y_data)

    return x_data, y_data
```

```python
transformer = MinMaxScaler()
data = transformer.fit_transform(data)

sequence_length = 3
x_data, y_data = load_time_series_data(data, sequence_length)
x_data = x_data.reshape(len(x_data), -1)
print(x_data.shape)
print(y_data.shape)
```

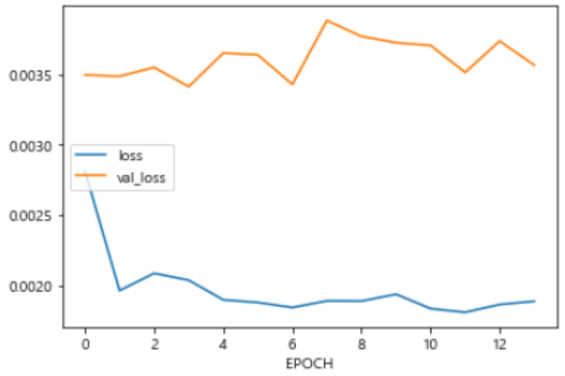
```
(1235, 3)
(1235, 1)
(530, 3)
(530, 1)
```

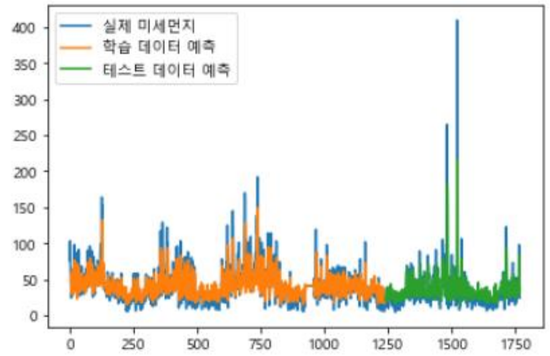
Train data, test data 분류

```python
x_train, x_test, y_train, y_test = train_test_split(x_data, y_data, test_size=0.3, shuffle=False)
```

Code & Results

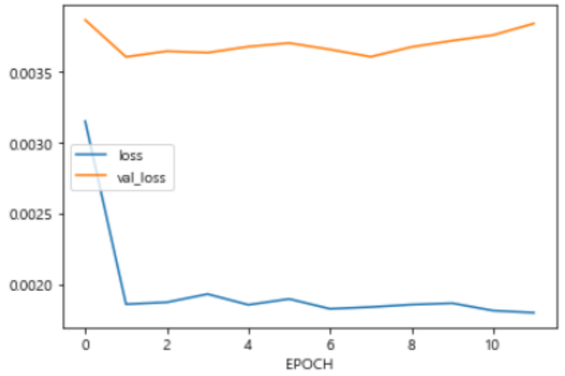
```
tf.keras.layers.Dense(units=256, activation='tanh')(input)
tf.keras.layers.Dense(units=128, activation='tanh')(net)
tf.keras.layers.Dense(units=32, activation='relu')(net)
tf.keras.layers.Dense(units=1)(net)
```

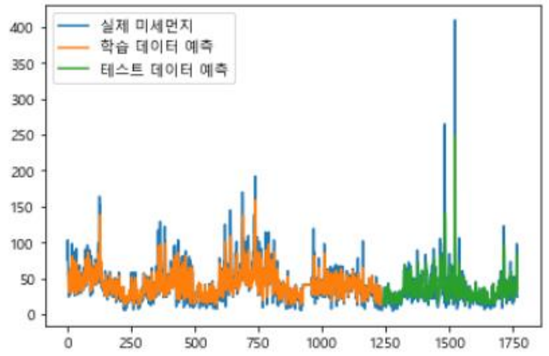




30.180792

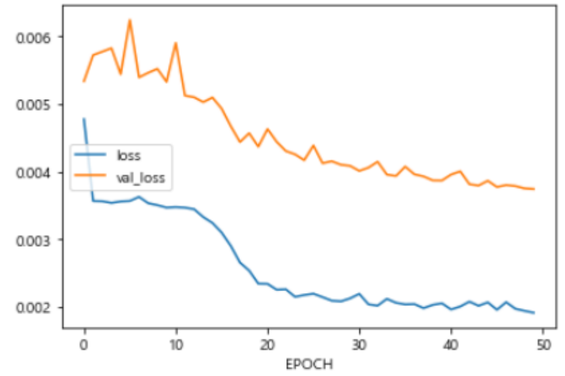
```
tf.keras.layers.Dense(units=256, activation='relu')(input)
tf.keras.layers.Dense(units=128, activation='relu')(net)
tf.keras.layers.Dense(units=32, activation='relu')(net)
tf.keras.layers.Dense(units=1)(net)
```

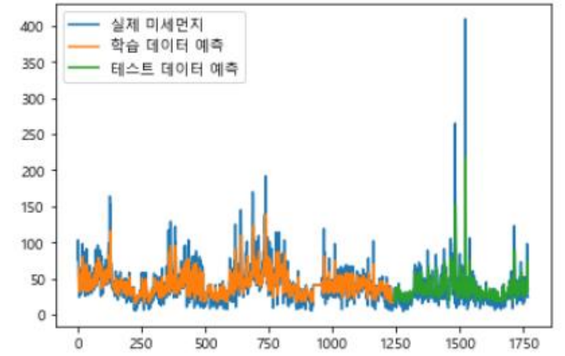




26.038826

```
tf.keras.layers.Dense(units=256, activation='softmax')(input)
tf.keras.layers.Dense(units=128, activation='relu')(net)
tf.keras.layers.Dense(units=32, activation='relu')(net)
tf.keras.layers.Dense(units=1)(net)
```



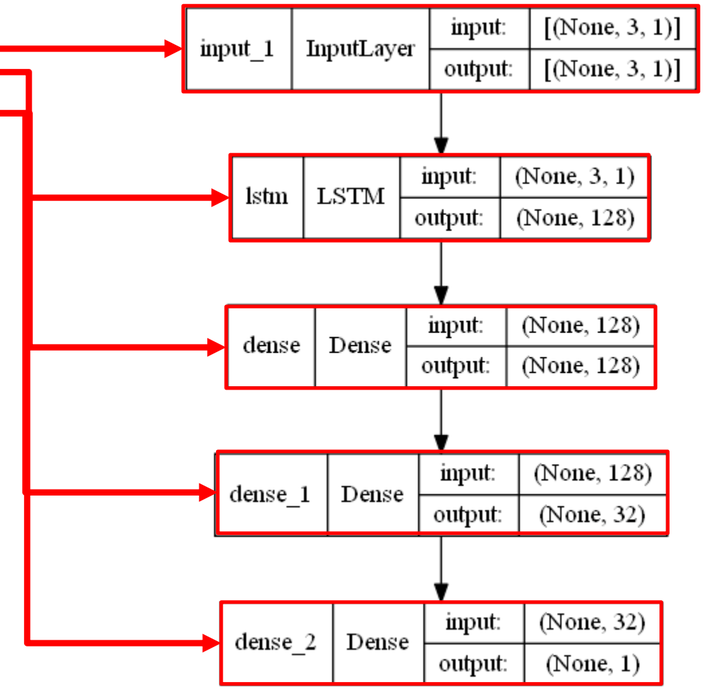


28.28347

Code & Results

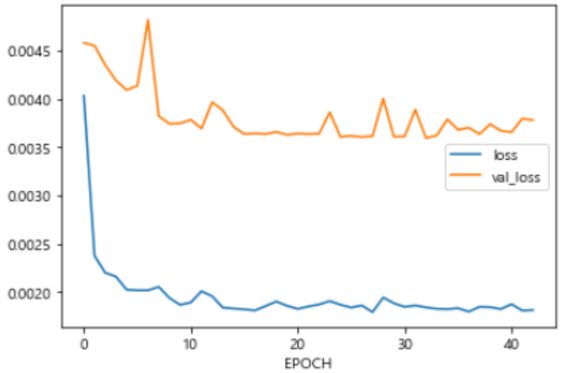
```
input = tf.keras.layers.Input(shape=(3,1))
net = tf.keras.layers.LSTM(units=128, activation='relu')(input)
net = tf.keras.layers.Dense(units=128, activation='relu')(net)
net = tf.keras.layers.Dense(units=32, activation='relu')(net)
net = tf.keras.layers.Dense(units=1)(net)
model = tf.keras.models.Model(input, net)
```

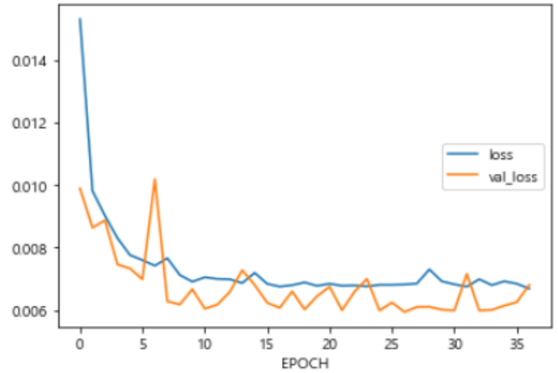
```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 3, 1)]            0

 lstm (LSTM)                 (None, 128)               66560

 dense (Dense)               (None, 128)               16512

 dense_1 (Dense)             (None, 32)                4128

 dense_2 (Dense)             (None, 1)                 33

=================================================================
Total params: 87,233
Trainable params: 87,233
Non-trainable params: 0
_________________________________________________________________
```

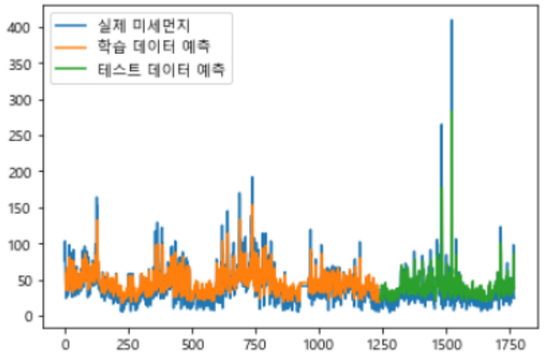


Code & Results

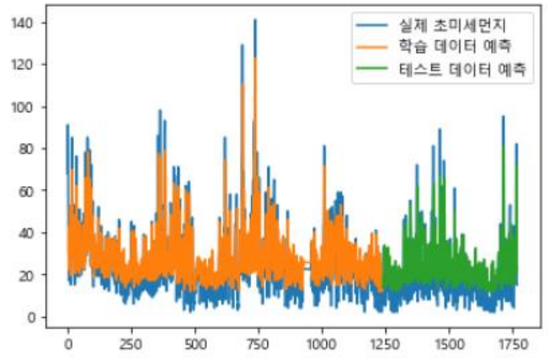
```
tf.keras.layers.LSTM(units=128, activation='relu')(input)
tf.keras.layers.Dense(units=128, activation='relu')(net)
tf.keras.layers.Dense(units=32, activation='relu')(net)
tf.keras.layers.Dense(units=1)(net)
```





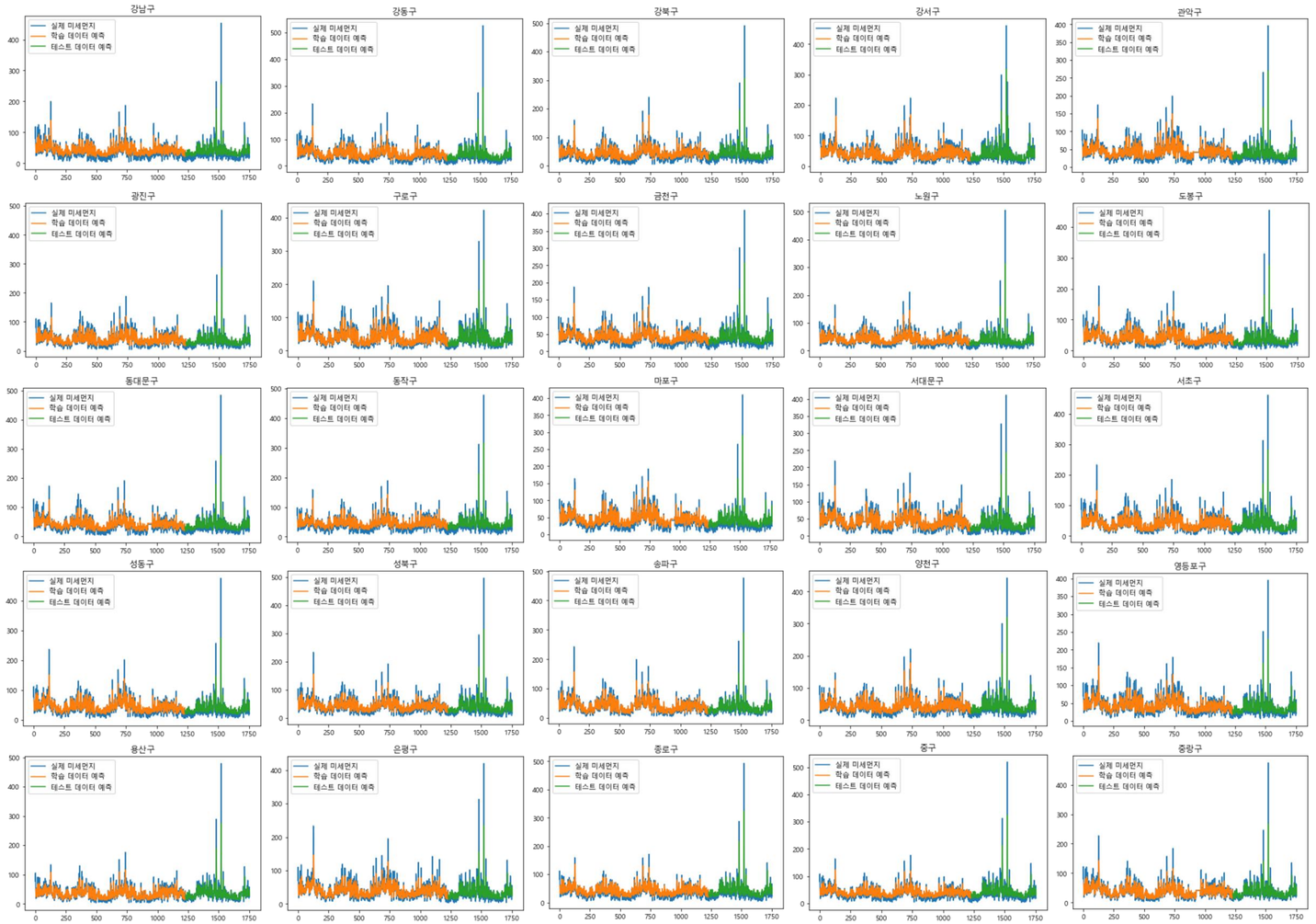


32.073906

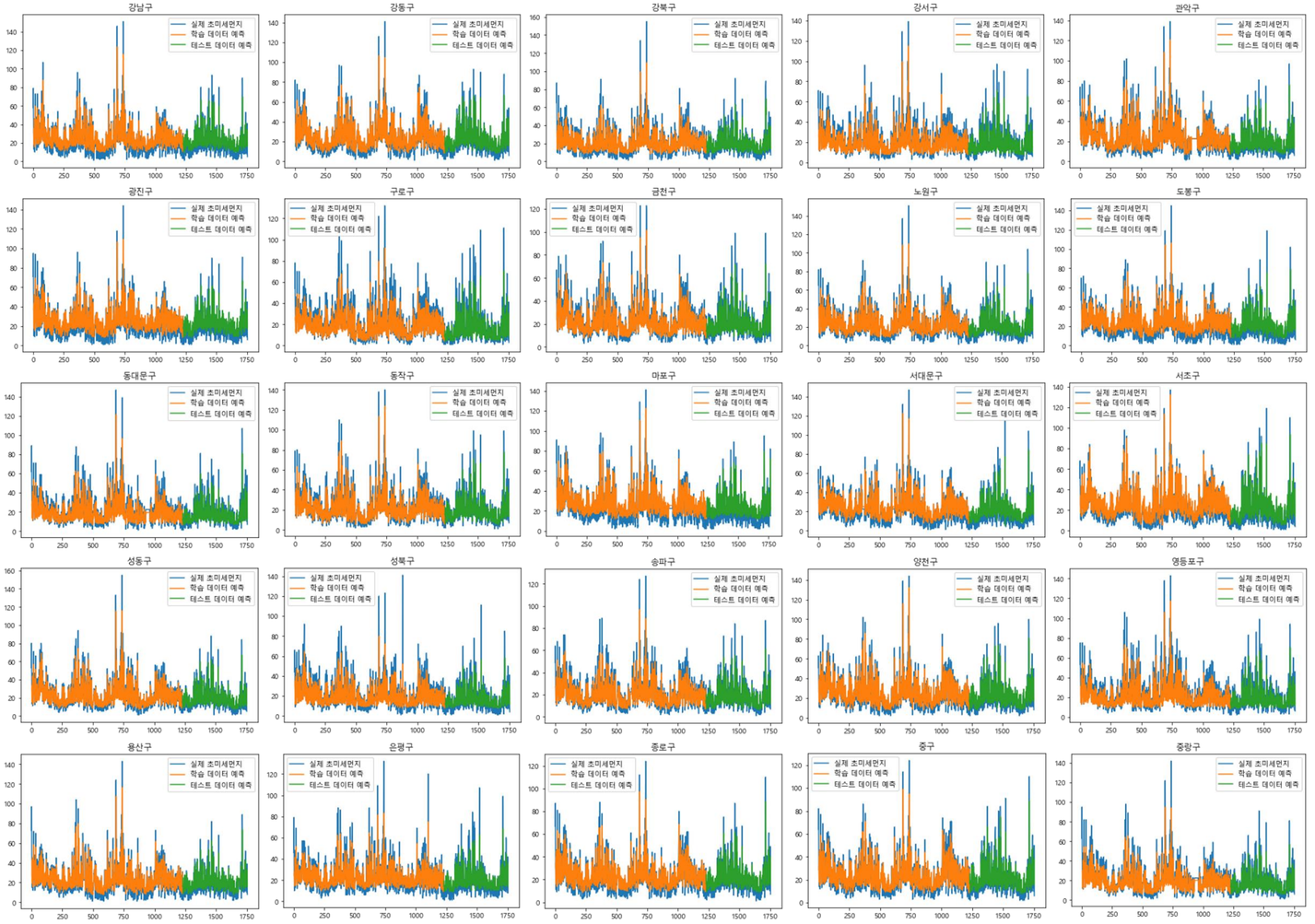


20.237158

Results 미세먼지



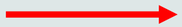
Results 초미세먼지



Problem & Solution

```
data = np.reshape(data_dff, [-1, 25, 6, 1])
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
```



```
def load_time_series_data(data, sequence_length):
    window_length = sequence_length + 1
    x_data = []
    y_data = []
    for i in range(0, len(data) - window_length + 1):
        window = data[i:i + window_length, :]
        x_data.append(window[:-1, :])
        y_data.append(window[-1, [-1]])
    x_data = np.array(x_data)
    y_data = np.array(y_data)

    return x_data, y_data
```

```
transformer = MinMaxScaler()
data = transformer.fit_transform(data)

sequence_length = 3
x_data, y_data = load_time_series_data(data, sequence_length)
x_data = x_data.reshape(len(x_data), -1)
print(x_data.shape)
print(y_data.shape)
```

```
x_train, x_test, y_train, y_test = train_test_split(x_data, y_data, test_size=0.3, shuffle=False)
```

Problem & Solution

```
input = tf.keras.layers.Input(shape=(sequence_length,))
net = tf.keras.layers.LSTM(units=128, activation='relu')(input)
net = tf.keras.layers.Dense(units=128, activation='relu')(net)
net = tf.keras.layers.Dense(units=32, activation='relu')(net)
net = tf.keras.layers.Dense(units=1)(net)
model = tf.keras.models.Model(input, net)
```

```
input = tf.keras.layers.Input(shape=(3,1))
net = tf.keras.layers.LSTM(units=128, activation='relu')(input)
net = tf.keras.layers.Dense(units=128, activation='relu')(net)
net = tf.keras.layers.Dense(units=32, activation='relu')(net)
net = tf.keras.layers.Dense(units=1)(net)
model = tf.keras.models.Model(input, net)
```

Reviews

손예린

- 이번 딥러닝 프로젝트를 통해 서울의 연간 미세먼지, 초미세먼지 농도는 대체로 계절성을 보이지만 기상 외 변수에 의한 아웃 라이어도 종종 관측된다는 것을 알게 되었다. 따라서 보다 정확한 예측을 위해서는 다양한 변수를 포함한 모델 구축이 필요할 것 같다.

김창현

- 정제되지 않은 데이터를 이용한 것이 처음이어서 데이터 전처리의 중요성을 알게 되었다. 에러를 하나하나 해결해가면서 개발에 능숙해질 수 있었다.

권희정

- 데이터 전처리의 중요성을 깨달으면서 환경변수, 모델에 따른 결과값의 변화에 흥미를 가졌으며 딥러닝에 더욱 관심을 가지게 되었다.

이우섭

- 이번 미세먼지 예측에서 데이터 전처리, 모델 구축이 쉽지 않았지만 예측을 통해 눈으로 확인하는 것이 새로운 재미를 느낀 것 같다.

이해성

- 이번 프로젝트를 하면서 모델링을 하는 것에 대한 어려움과 데이터 수집 및 전처리의 중요성을 느낄 수 있었다.

Thank you