딥러닝을 통한 서울시 미세먼지 예측

동국대학교
이종수

dongguk UNIVERSITY

Table of Contents

서론

- 기존 예측 시스템 조사

분석 절차

- 데이터 수집
- 데이터 전처리
- CNN
- RNN
- 학습 및 예측
- 시각화

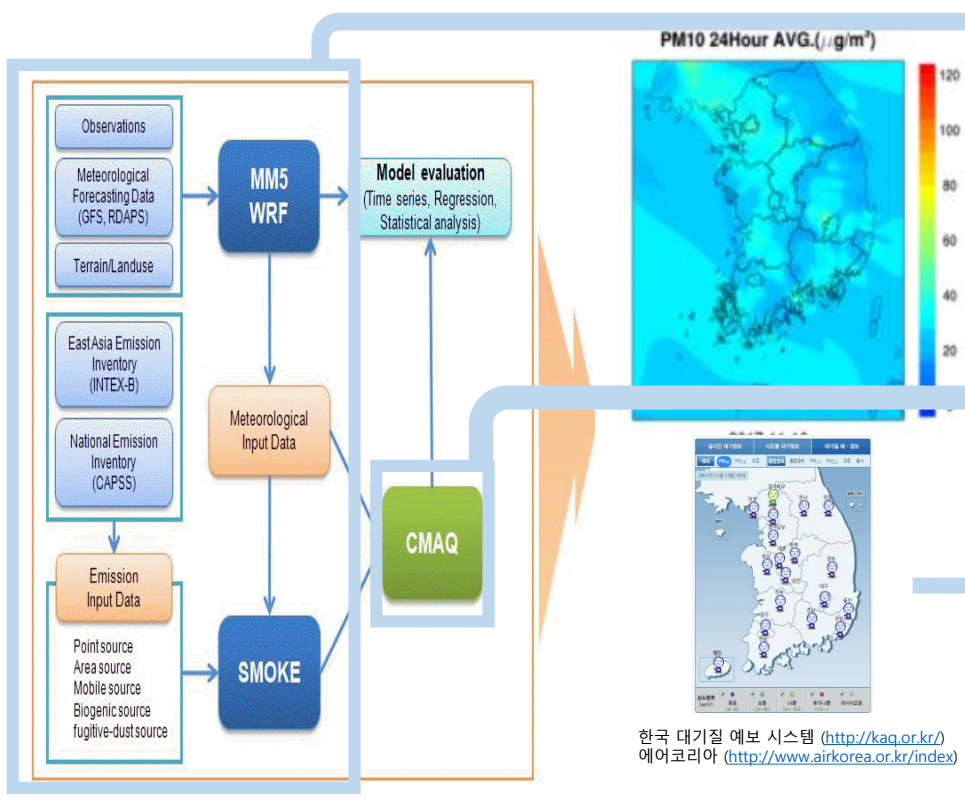
결론

- 한계점
- 해결방안 및 활용

dongguk UNIVERSITY

IDEA

기존 미세먼지 예측 시스템



한국 대기질 예보 시스템 (http://kaq.or.kr/)
에어코리아 (http://www.airkorea.or.kr/index)

예보하는 데 필요한 데이터 종류/수가 많음

- 대기자료(GFS, RDAPS, 지형 등), 배출량자료 등

계산의 복잡성이 높음

- 대기자료를 통해 미세먼지의 이동경로를 구한 후, 중국과 한국의 평상시 대기오염물질 배출 총량 값을 넣어 계산

시/도별 예보만 제공

- 구/군별 상세 예보는 찾아볼 수 없음

dongguk UNIVERSITY

WHAT IF ?

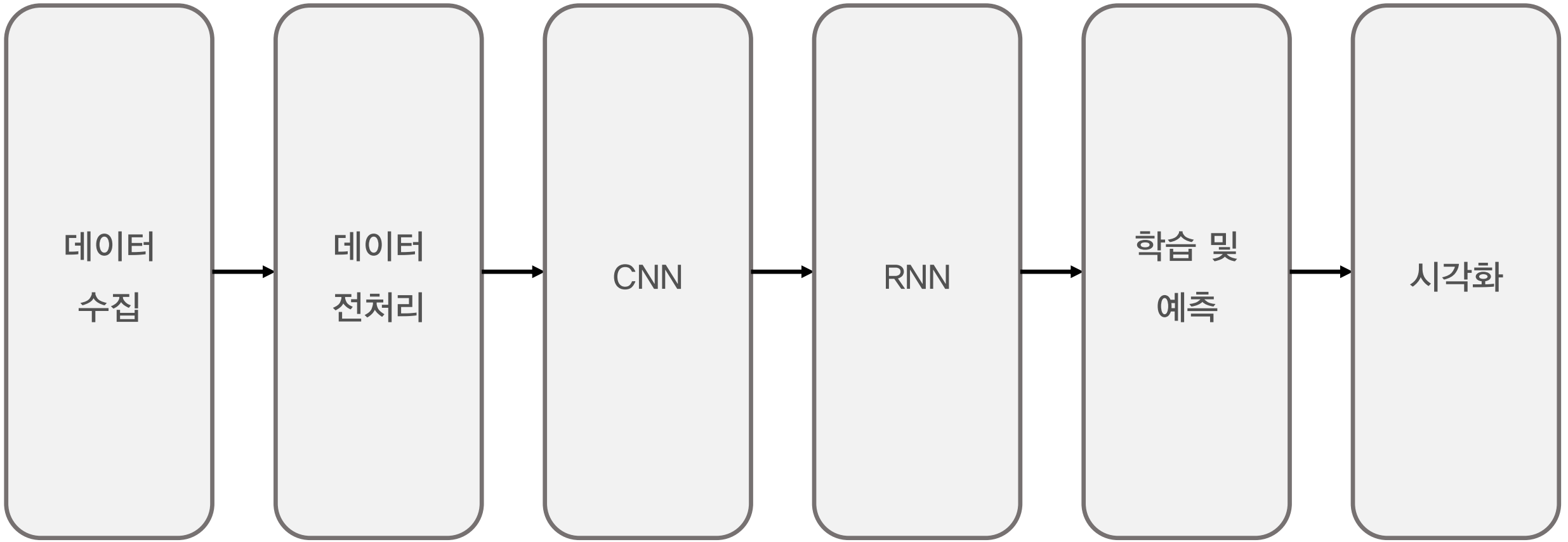
과거 대기 데이터

+

딥러닝

“낮은 비용으로 더욱 자세한 예측 시스템을 구축해 활용 가능”

Procedure



“ 수집, 처리, 분석, 표현 언어 : Python ”

사용 모듈 : numpy, pandas, Tensorflow, matplotlib, folium 등

Procedure – (1) Building Dataset

서울시 일별 평균 대기오염도 정보 (http://data.seoul.go.kr/openinf/sheetview.jsp?infId=OA-2218)

1	번호	측정일시	측정소명	이산화질소	오존농도(p	일산화탄소	아황산가스	미세먼지(μ	초미세먼지
2	1	20170101	강남구	0.04	0.002	0.8	0.005	80	58
3	2	20170101	강동구	0.039	0.003	0.8	0.004	80	54
4	3	20170101	강북구	0.046	0.003	1	0.005	78	65
5	4	20170101	강서구	0.053	0.005	1	0.006	75	54
6	5	20170101	관악구	0.064	0.005	1.3	0.006	75	56
7	6	20170101	광진구	0.045	0.005	0.8	0.004	80	71
8	7	20170101	구로구	0.036	0.006	1	0.007	79	54
9	8	20170101	금천구	0.053	0.006	1	0.005	66	45
10	9	20170101	노원구	0.052	0.004	1.2	0.008	82	65
11	10	20170101	도봉구	0.052	0.003	0.9	0.006	78	47
12	11	20170101	동대문구	0.046	0.003	1	0.006	87	62
13	12	20170101	동작구	0.058	0.004	1.1	0.005	72	58
14	13	20170101	마포구	0.055	0.004	1.2	0.005	76	68
15	14	20170101	서대문구	0.039	0.008	0.8	0.005	91	50
16	15	20170101	서초구	0.042	0.005	1.2	0.005	87	51
17	16	20170101	성동구	0.045	0.003	0.8	0.005	77	53
18	17	20170101	성북구	0.048	0.003	0.9	0.005	66	44
19	18	20170101	송파구	0.04	0.004	1.2	0.005	64	49
20	19	20170101	양천구	0.046	0.004	1.1	0.005	80	51

“ 대기 데이터 API를 통해 크롤링하는 방식으로 수집
2010년부터 2017년까지의 시계열 데이터 ”

Procedure – (1) Building Dataset

Attributes

측정일시, 측정소명, 이산화질소, 오존농도, 일산화탄소, 아황산가스, 미세먼지, 초미세먼지

Training Dataset (3.6MB)

Row : 69739 / Column : 8 / 2010-01-01 ~ 2017-08-31

Test Dataset (92KB)

Row : 1875 / Column : 8 / 2017-09-01 ~ 2017-11-12

Procedure – (2) Preprocessing

1. 명목형 자료의 수치화

```
data_df['측정소명'] = data_df['측정소명'].map({'강남구':0, '강동구':1, '강북구':2, '강서구':3, '관악구':4, '광진구':5,
                                        '구로구':6, '금천구':7, '노원구':8, '도봉구':9, '동대문구':10, '동작구':11,
                                        '마포구':12, '서대문구':13, '서초구':14, '성동구':15, '성북구':16, '송파구':17,
                                        '양천구':18, '영등포구':19, '용산구':20, '은평구':21, '종로구':22, '중구':23, '중랑구':24})
```

2. 초미세먼지 측정 이후 데이터 선택

```
data_df = data_df[(data_df['측정일시']>20131231)]
```

3. 공백 데이터를 해당 열의 평균으로 대체

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

4. Normalization

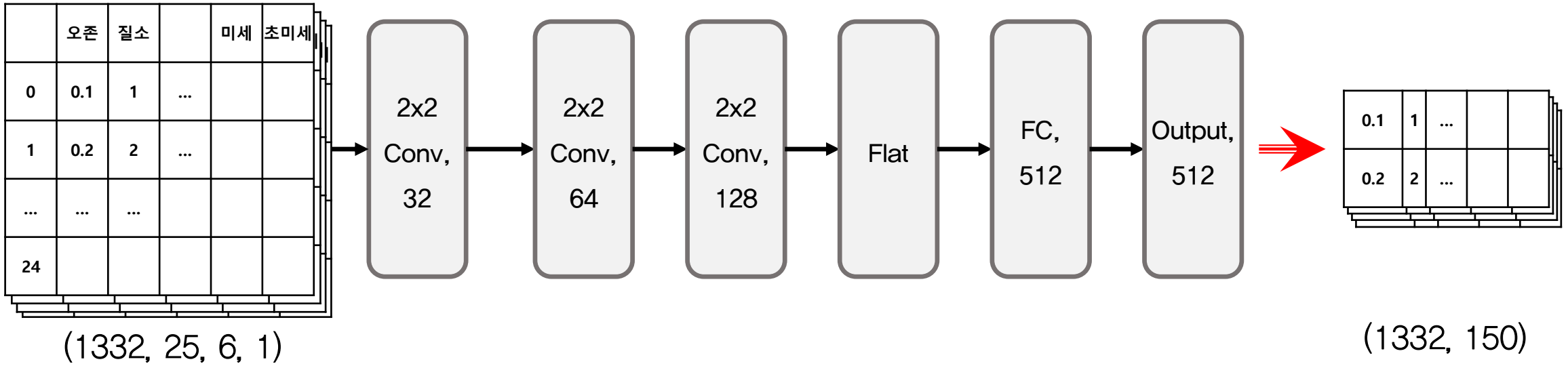
```
def MinMaxScaler(data):
    numerator = data - np.min(data, 0)
    denominator = np.max(data, 0) - np.min(data, 0)
    return numerator / (denominator + 1e-7), np.min(data, 0), np.max(data, 0)
```

5. CNN을 위해 25(지역구 수), 6(속성 수)의 이미지 형태로 변환

```
data = np.reshape(data_dff[:,2:], [-1, 25, 6, 1])
```

dongguk UNIVERSITY

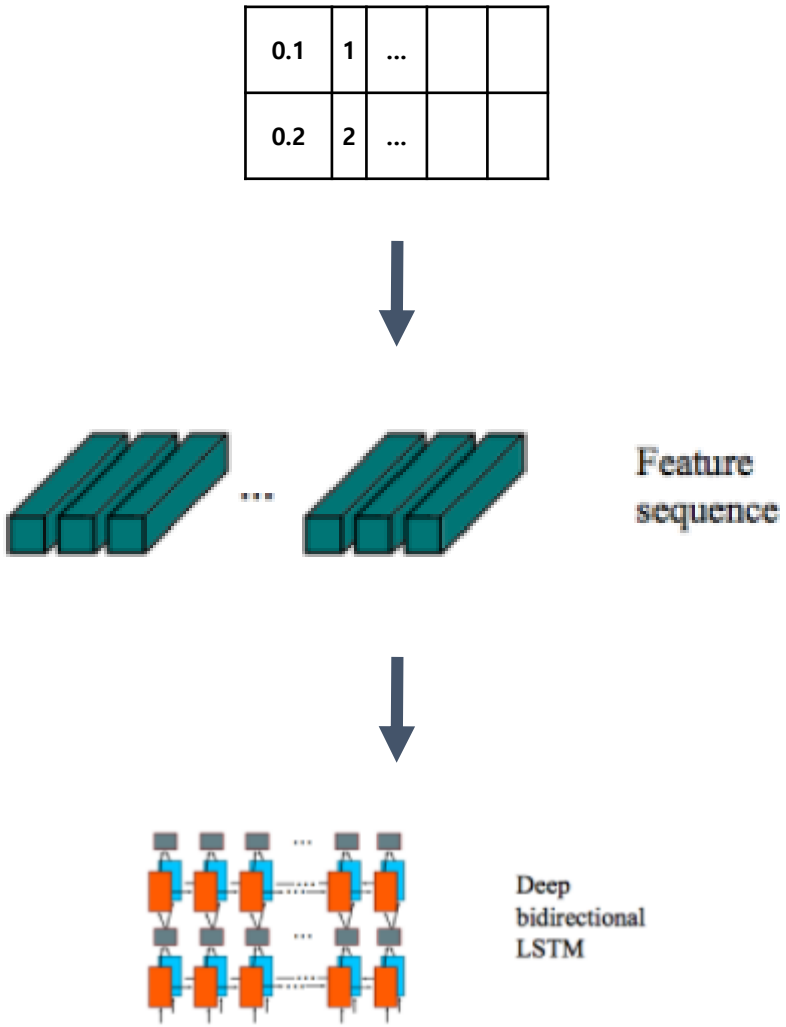
Procedure – (3) CNN




```
X = tf.placeholder(np.float32, shape=[None, img_height, img_width, 1])
conv1 = tf.layers.conv2d(inputs=X, filters=32, kernel_size=[2, 2], padding="same", activation=tf.nn.relu)
conv2 = tf.layers.conv2d(inputs=conv1, filters=64, kernel_size=[2, 2], padding="same", activation=tf.nn.relu)
conv3 = tf.layers.conv2d(inputs=conv2, filters=128, kernel_size=[2, 2], padding="same", activation=tf.nn.relu)
pool3_flat = tf.reshape(conv3, [-1, 6 * 25 * 128])
fc1 = tf.layers.dense(inputs=pool3_flat, units=512, activation=tf.nn.relu)
fc2 = tf.layers.dense(inputs=fc1, units=512, activation=tf.nn.relu)
output = tf.layers.dense(inputs=fc2, units=150)
```

- 근처 픽셀과 영향을 주고 받는 Convolution 과정을 통해 지역간의 연관성을 반영한 수치로 변환

Procedure – (4) RNN




```
trainX = []
trainY = trainY[seq_length:,:]
for i in range(0, len(x) - seq_length):
    _x = x[i:i + seq_length]
    _y = x[i + seq_length]
    trainX.append(_x)
trainY = np.reshape(trainY,[-1,150])
np.array(trainX).shape, trainY.shape

testX = []
testY = testY[seq_length:,:]
for i in range(0, len(test) - seq_length + 1):
    _x = test[i:i + seq_length]
    testX.append(_x)
testY = np.reshape(testY,[-1,150])

np.array(testX).shape, testY.shape
```

```
# train Parameters
seq_length = 7
data_dim = 6
hidden_dim = 1024
output_dim = 1024
num_layers = 2
```

```
X = tf.placeholder(tf.float32, [None, seq_length, 150])
Y = tf.placeholder(tf.float32, [None, 150])

cell = tf.contrib.rnn.MultiRNNCell([(tf.contrib.rnn.BasicLSTMCell(hidden_dim,state_is_tuple=True))
                                    for i in range(num_layers)], state_is_tuple=True)
drop = tf.contrib.rnn.DropoutWrapper(cell, output_keep_prob=0.7) # 드랍아웃한 셀
outputs, _states = tf.nn.dynamic_rnn(drop, X, dtype=tf.float32)

Y_pred = tf.contrib.layers.fully_connected(
    outputs[:, -1], output_dim, activation_fn=None)

W1 = tf.Variable(tf.truncated_normal([output_dim, 150], stddev=0.1))
b1 = tf.Variable(tf.random_normal([150]))
z1 = tf.matmul(Y_pred, W1) + b1

logits = tf.matmul(Y_pred, W1) + b1
```

- 시계열 데이터에서 패턴을 인식하는 인공 신경망 형성

dongguk UNIVERSITY

Procedure – (5) Training / Prediction

주요 학습 변수

Loss = $\sum(H(x_i) - y_i)^2$ / Optimizer = Adam / Learning Rate = 0.001

```
learning_rate = 0.001
iterations = 2500

# cost/loss
loss = tf.reduce_sum(tf.square(logits - Y))  # sum
# optimizer
optimizer = tf.train.AdamOptimizer(learning_rate)
train = optimizer.minimize(loss)
```

변수 선정 기준

과적합을 방지하기 위해 예측 값과 실제 값의 RMSE = $\sqrt{\frac{1}{n}\sum(x_i - y_i)^2}$ 를 가장 작게 만드는 변수 집합을 변수로 선정 / TRIAL AND ERROR

(Number of CNN Layer / Filter, RNN Layer, 학습 횟수 등)

학습 횟수	미세먼지 RMSE	초미세먼지 RMSE
1000	17.78	8.80
…	…	…
2000	15.26	8.55
2250	15.30	8.86
2500	16.50	9.39
2750	16.74	10.07
3000	18.19	10.57
4000	18.37	11.19
5000	20.60	12.56
…	…	…
10000	25.00	13.04

예측 값

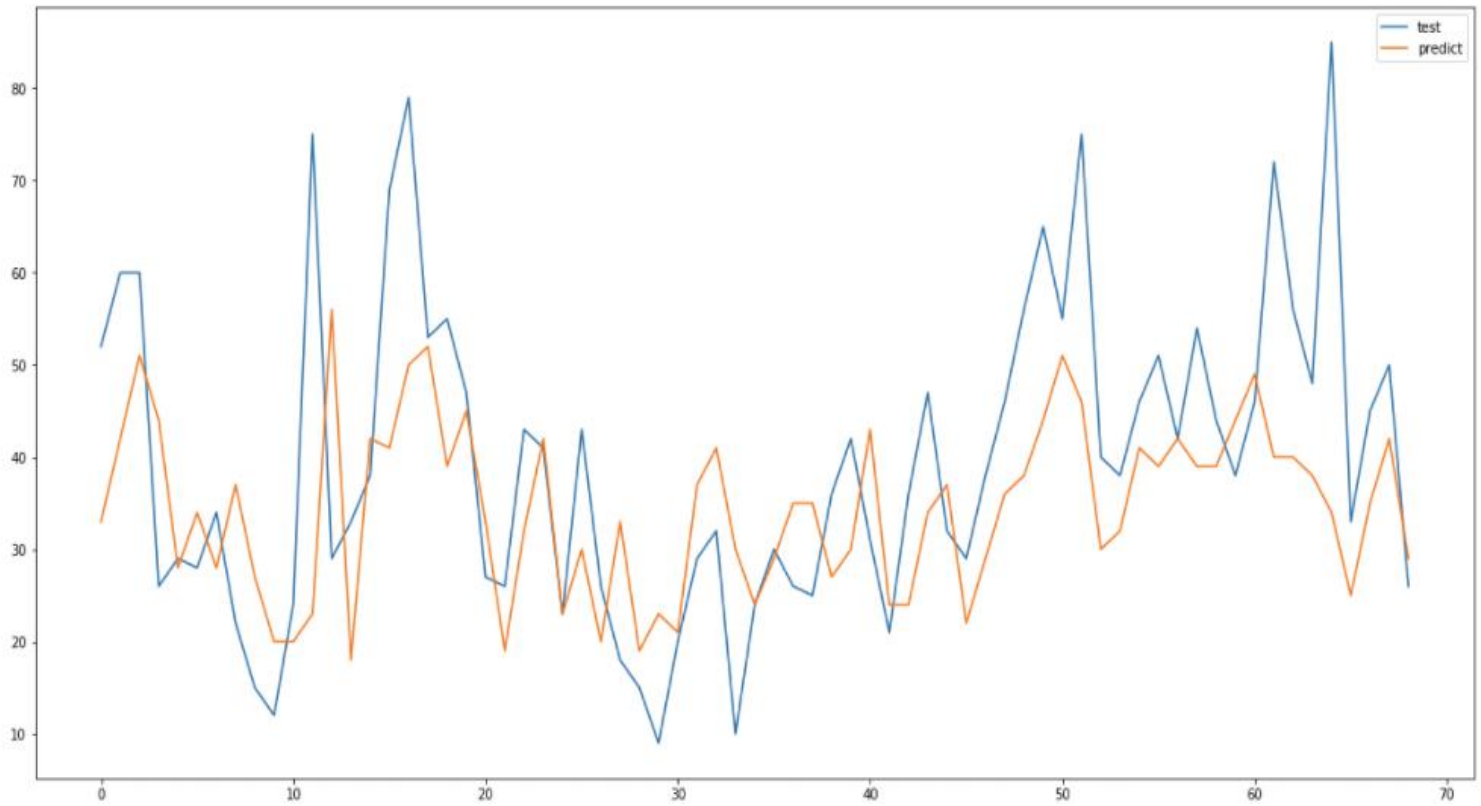
20170910	강남구	51	31	62	33
20170910	강동구	57	31	53	31
20170910	강북구	41	29	36	32
20170910	강서구	58	30	64	43
20170910	관악구	47	33	50	37
20170910	광진구	49	31	37	29
20170910	구로구	49	29	57	43
20170910	금천구	41	32	49	39
20170910	노원구	45	29	38	31
20170910	도봉구	44	29	48	33
20170910	동대문구	45	28	53	34
20170910	동작구	49	30	53	44

20170910	마포구	42	35	47	39
20170910	서대문구	52	30	49	38
20170910	서초구	47	29	53	32
20170910	성동구	53	31	56	31
20170910	성북구	51	31	60	42
20170910	송파구	54	29	47	30
20170910	양천구	51	36	57	45
20170910	영등포구	54	30	61	35
20170910	용산구	51	31	47	36
20170910	은평구	52	29	43	34
20170910	종로구	43	31	40	32
20170910	중구	51	29	46	35
20170910	중랑구	50	32	45	31

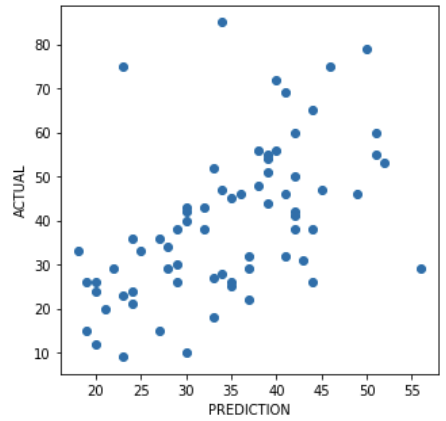
…

Procedure – (5) Training / Prediction

실제 값과의 비교 그래프 – 미세먼지 (성북구)



실제 데이터 예측 데이터



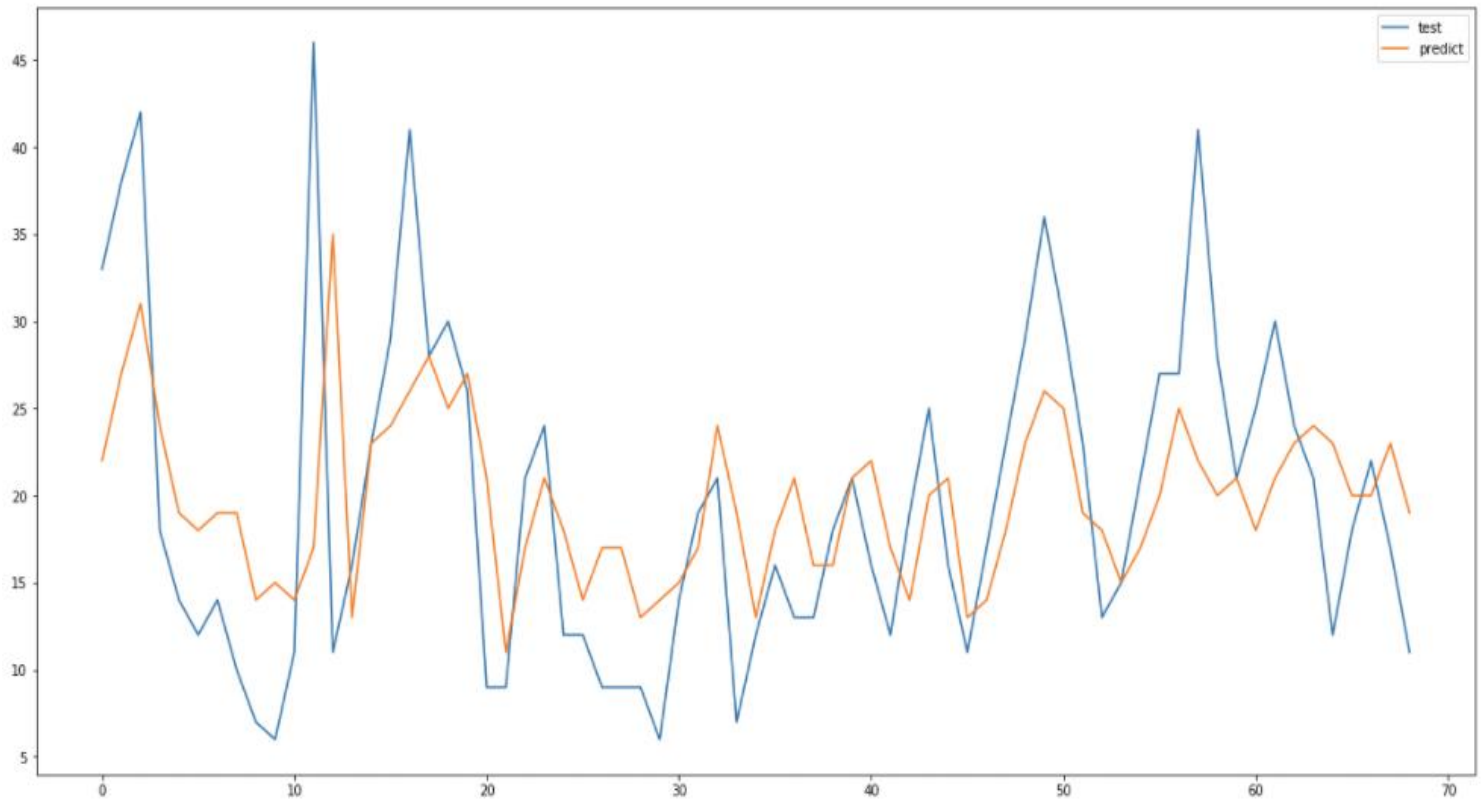
상관계수
0.53

```
plt.scatter(sb[:,2], sb[:,4])
plt.xlabel('PREDICTION')
plt.ylabel('ACTUAL')
plt.show()
print(np.corrcoef(np.array(sb[:,2], dtype=int), np.array(sb[:,4], dtype=int)))
```

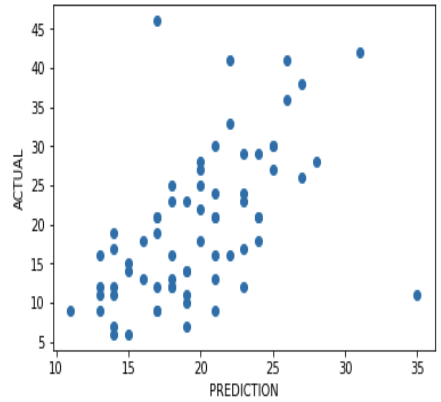
```
fig = plt.figure(figsize=(4, 2))
plt.plot(test1[0:100, i], label='test')
plt.plot(predict1[0:100, i], label='predict')
plt.legend()
plt.show()
```

Procedure – (5) Training / Prediction

실제 값과의 비교 그래프 – 초미세먼지 (성북구)



실제 데이터 예측 데이터



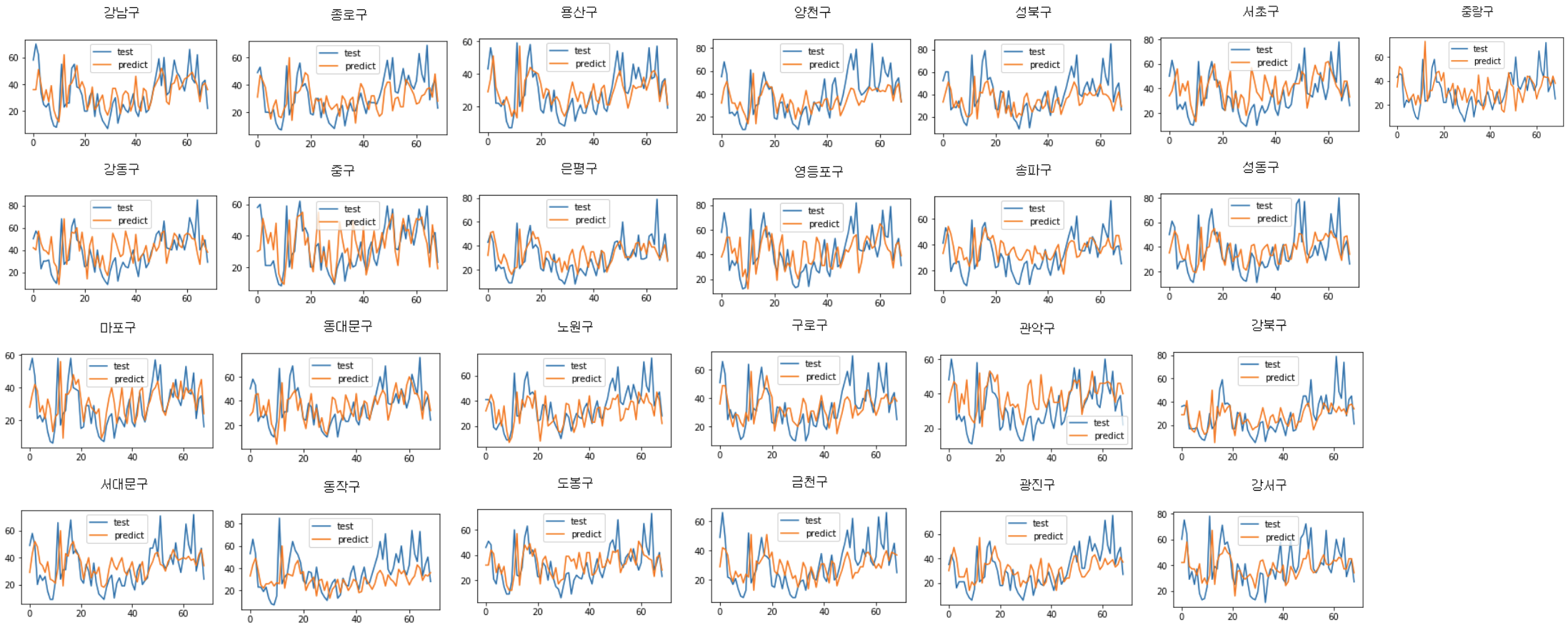
상관계수
0.55

```
plt.scatter(sb[:,3], sb[:,5])
plt.xlabel('PREDICTION')
plt.ylabel('ACTUAL')
plt.show()
print(np.corrcoef(np.array(sb[:,3], dtype=int), np.array(sb[:,5], dtype=int)))
```

```
fig = plt.figure(figsize=(4, 2))
plt.plot(test1[0:100, i], label='test')
plt.plot(predict1[0:100, i], label='predict')
plt.legend()
plt.show()
```

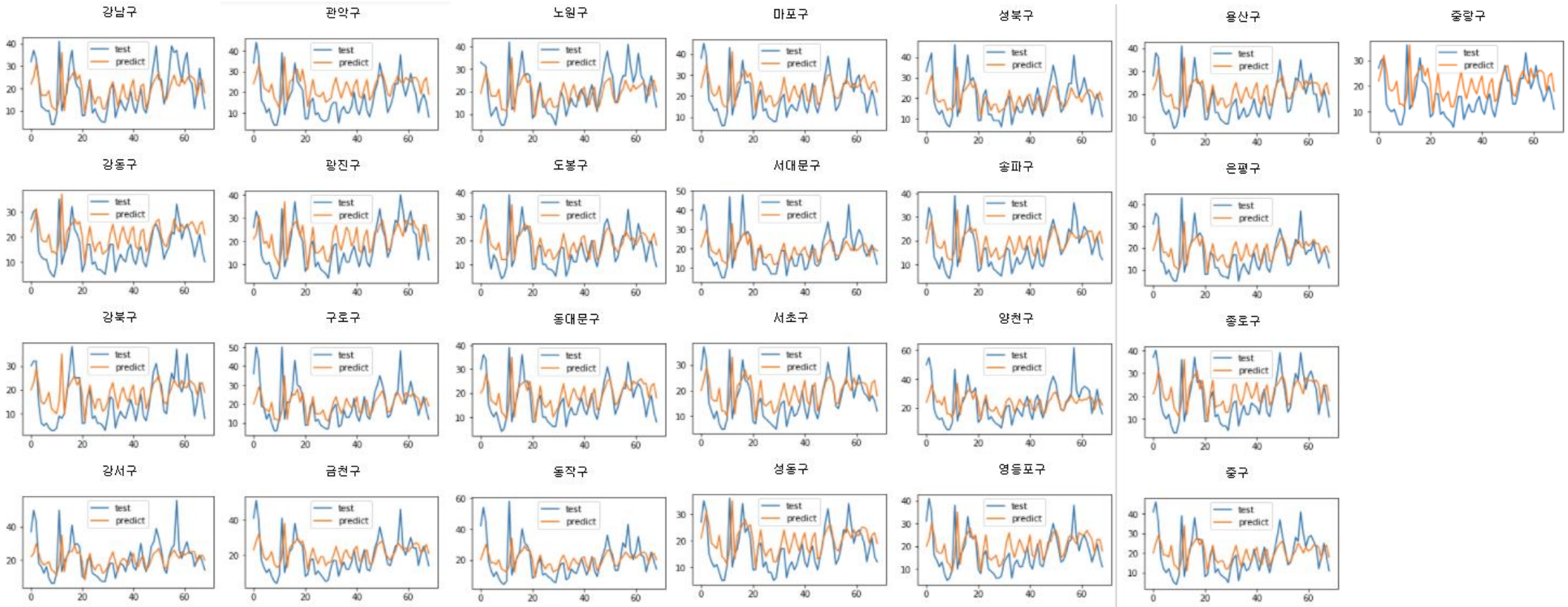
Procedure – (5) Training / Prediction

실제 값과의 비교 그래프 – 미세먼지



Procedure – (5) Training / Prediction

실제 값과의 비교 그래프 – 초미세먼지



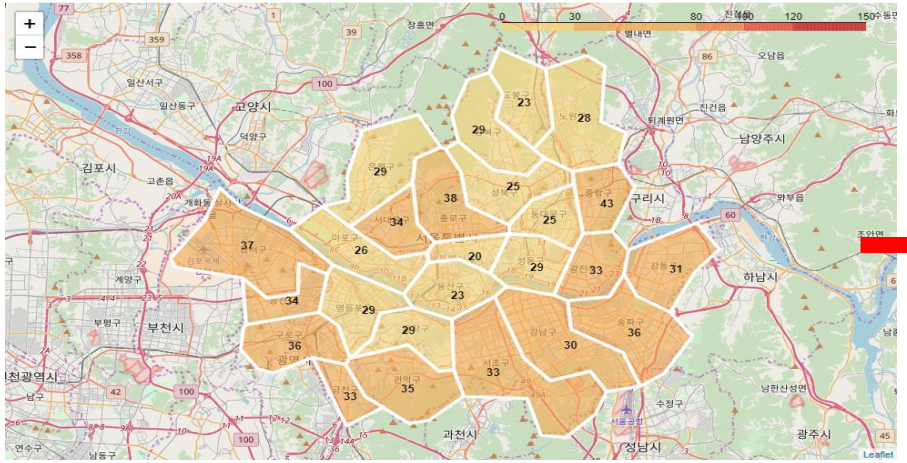
실제 데이터 예측 데이터

dongguk UNIVERSITY

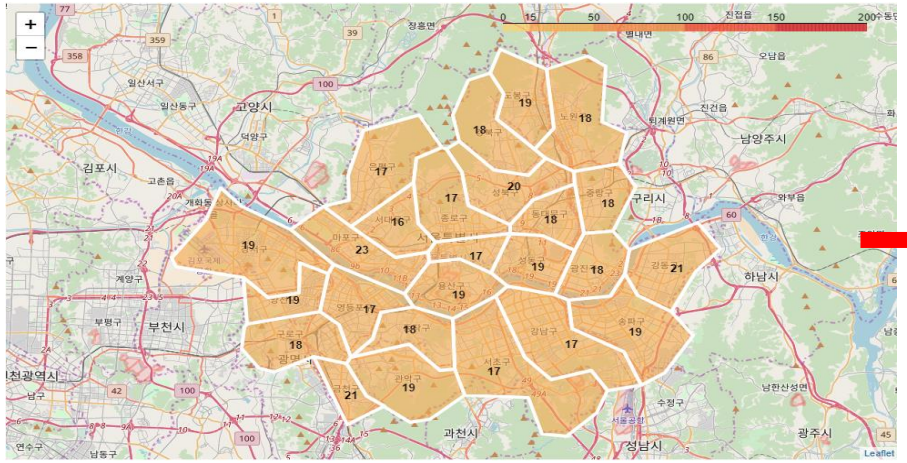
Procedure – (6) Visualization

	좋음	보통	나쁨	매우나쁨
미세먼지	0-30	31-80	81-150	151-
초미세먼지	0-15	16-50	51-100	100-

예측 Data Visualization

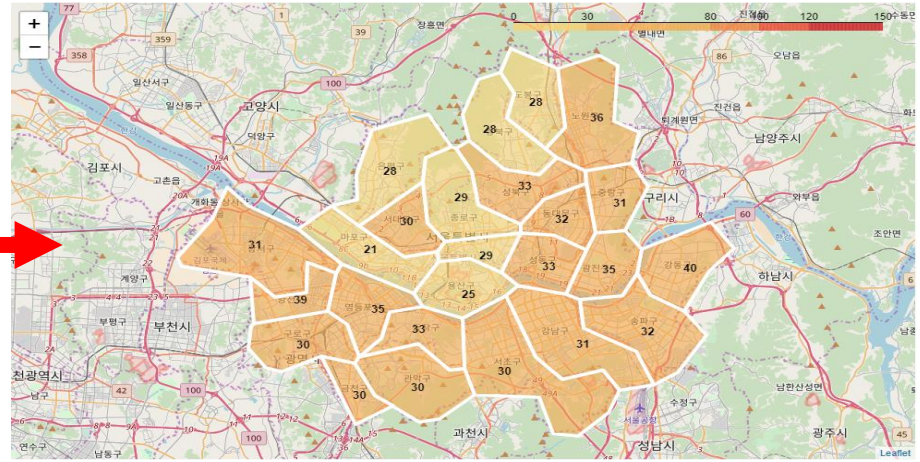


2017년 11월 12일 예측 미세먼지

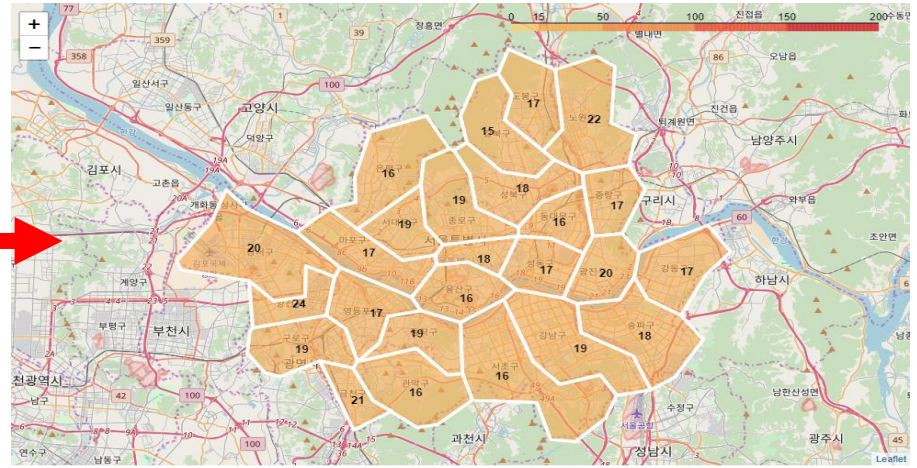


2017년 11월 12일 예측 초미세먼지

실제 Data Visualization



2017년 11월 12일 실제 미세먼지



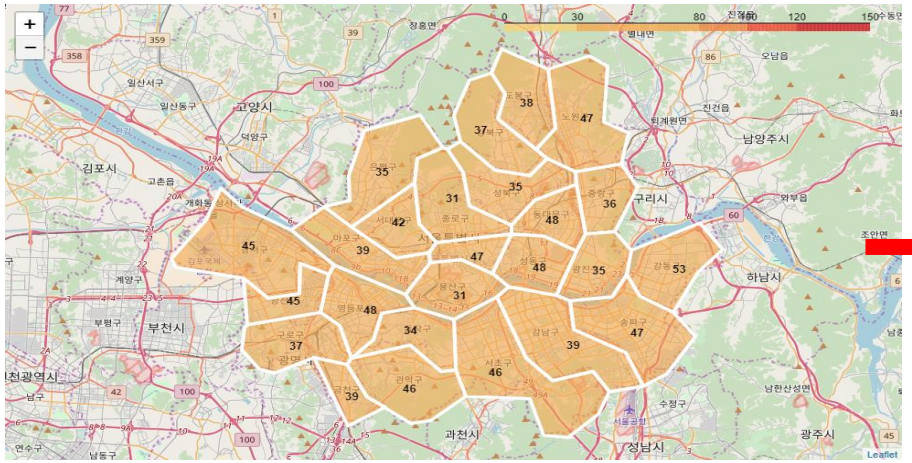
2017년 11월 12일 실제 초미세먼지

dongguk UNIVERSITY

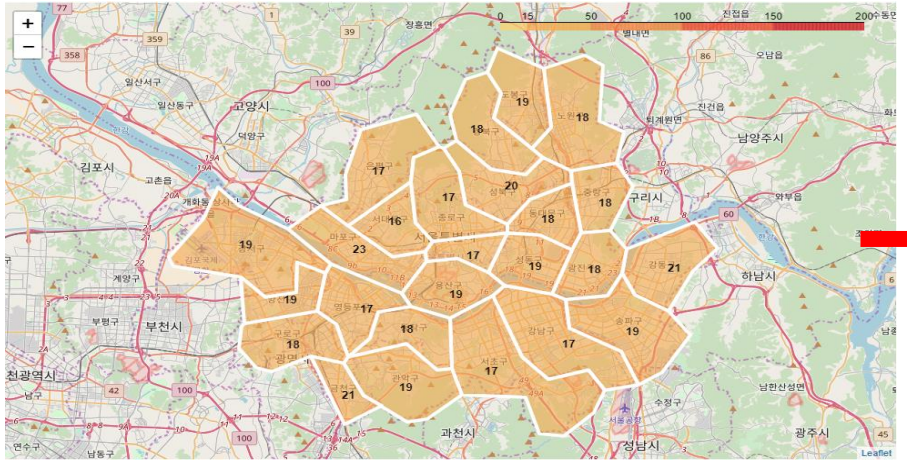
Procedure – (6) Visualization

	좋음	보통	나쁨	매우나쁨
미세먼지	0-30	31-80	81-150	151-
초미세먼지	0-15	16-50	51-100	100-

예측 Data Visualization

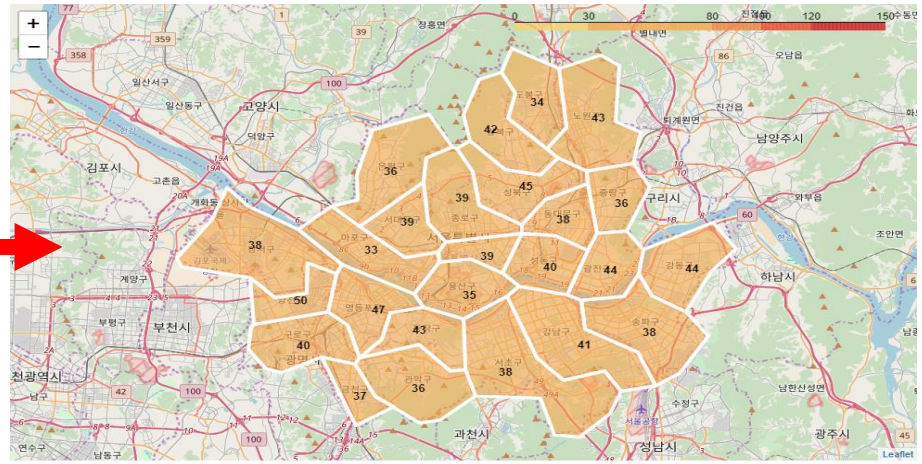


2017년 11월 13일 예측 미세먼지

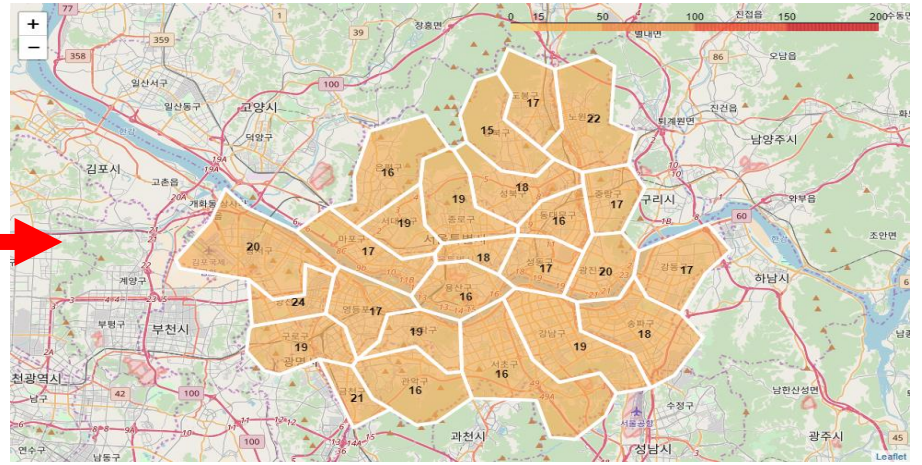


2017년 11월 13일 예측 초미세먼지

실제 Data Visualization



2017년 11월 13일 실제 미세먼지

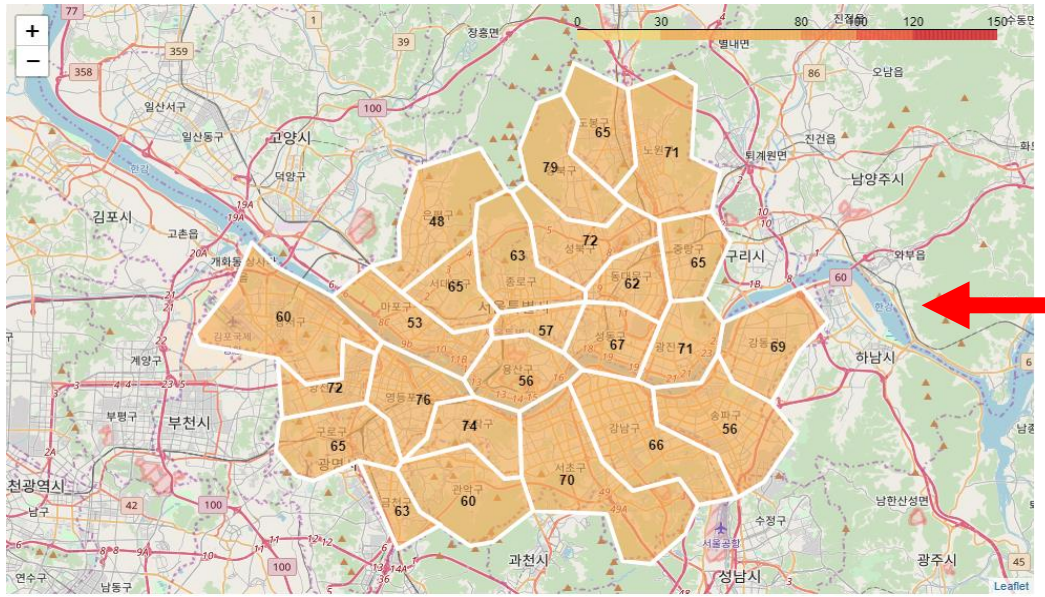


2017년 11월 13일 실제 초미세먼지

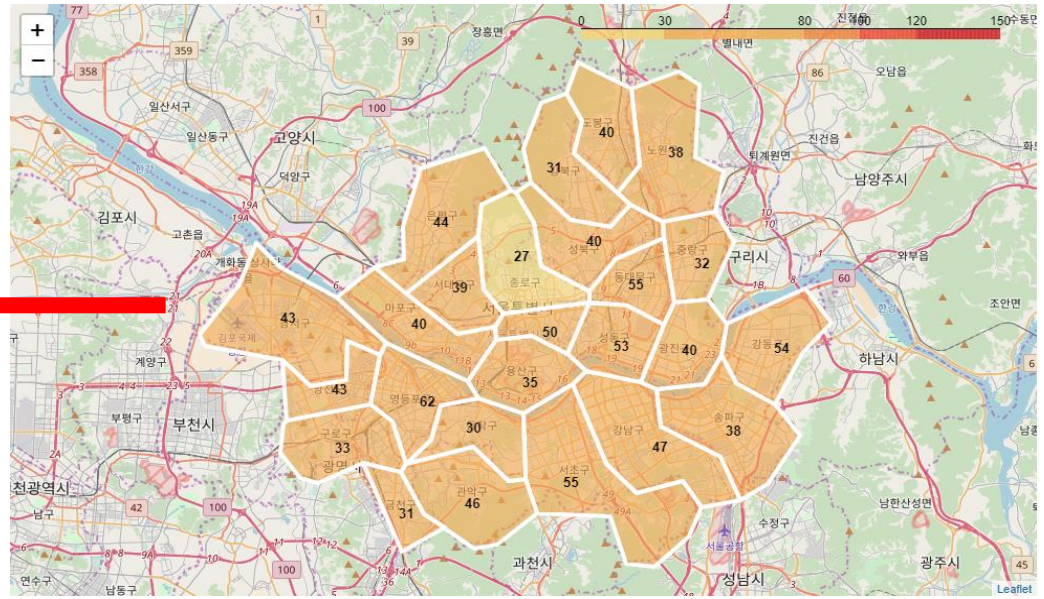
dongguk UNIVERSITY

Procedure – (6) Visualization

	좋음	보통	나쁨	매우나쁨
미세먼지	0-30	31-80	81-150	151-
초미세먼지	0-15	16-50	51-100	100-



2017년 11월 8일 실제 미세먼지



2017년 11월 8일 예측 미세먼지

2017년 11월 8일 실제 예보

```
geo_path = 'skorea_municipalities_geo_simple.json'
geo_str = json.load(open(geo_path, encoding='utf-8'))
XY = pd.read_csv('XY.csv')
XY = pd.DataFrame(data = np.array(XY)[:,2:4], index=np.array(XY)[:,0], columns=['위도', '경도'])
date = 20171104
map1111_1 = folium.Map(location=[37.5702, 126.982], zoom_start=11, tiles='openstreetmap')
map1111_1.choropleth(geo_str, data = data[data['측정일시']==date]['예상미세'],
                     columns = [data[data['측정일시']==date].values[:,1], data[data['측정일시']==date]['예상미세']],
                     line_color='white',line_weight=4, fill_color= 'YlOrRd', key_on = 'feature.id',
                     threshold_scale=[0, 40, 80, 90, 100, 120])

print(date," 예측 미세먼지")
for n in range(25):
    folium.Marker(location = [float(XY.values[n, 0]),float(XY.values[n, 1])], icon=DivIcon(
        icon_size=(0,0),
        icon_anchor=(0,0),
        html='<div style="font-size: 10pt; font-weight:bold ; color : black">{}</div>'
        .format(int(data[data['측정일시']==date].values[n, 2])),
        )).add_to(map1111_1)
map1111_1
```

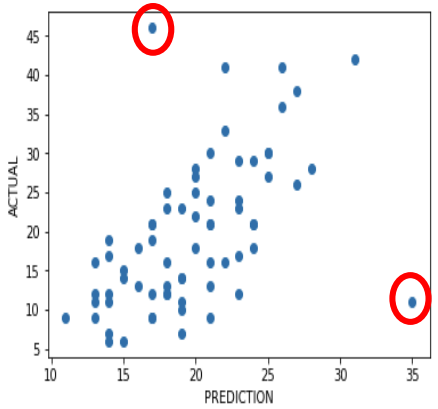
Conclusion

한계

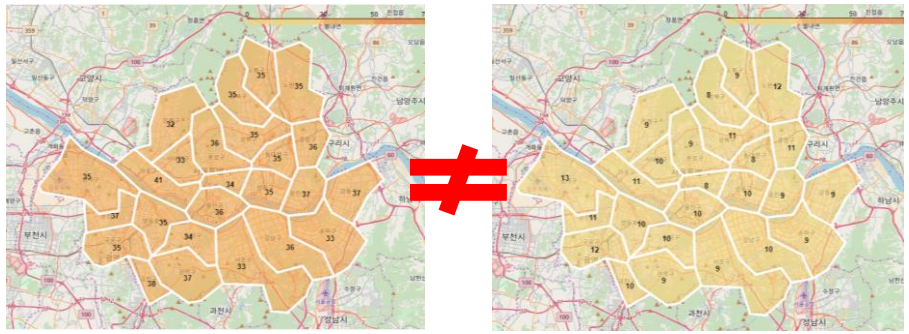
1. 데이터 수 부족
2. 시간단위 특성 미반영
3. 불균형 데이터

...

예상 값과 실제 값이 불일치한 날의 발생



(예시) 9월 20일 초미세먼지



개선방안 및 활용

1. 시간성 및 배출량 자료를 포함한 증가된 수의 데이터와 개선된 하드웨어를 바탕으로 분석 시, 기존의 시스템보다 더욱 낮은 비용으로 간단하게 구 단위의 자료까지 예측 가능
2. 지하철 승하차 인원 데이터와 함께 분석하여, 미세먼지가 높을 것으로 예상되는 지역의 승하차 인원이 많은 역의 미세먼지 관리
3. 주거/상가 데이터와 함께 분석하여, 원인 분석 후 미세먼지 문제 해결 및 억제 방안 마련


0
30
80
120
150
25
19
23
27
31
27
31
21
35
18
30
33
21
38
29
31
22
24
32
20
28
32
36
25
32
감사합니다.
고양시
김포시
부천시
하남시
구리시
남양주시
과천시
성남시
광주시
Leaflet