Group Project Presentation

미세먼지 예측 모델

기상인자와 오염 배출 인자를 중심으로

발표자 —김용성 배상준 | 코드 작성 — 변성일 | PPT 제작 — 차혜선

CONTENTS

01 주제 선정

02 데이터 선정

03 데이터 전처리

04 모델 학습

05 모델 학습 결과

06 피드백

주제 선정

선정 주제 : 미세먼지 예측 모델

미세먼지가 사회에 미치는 영향

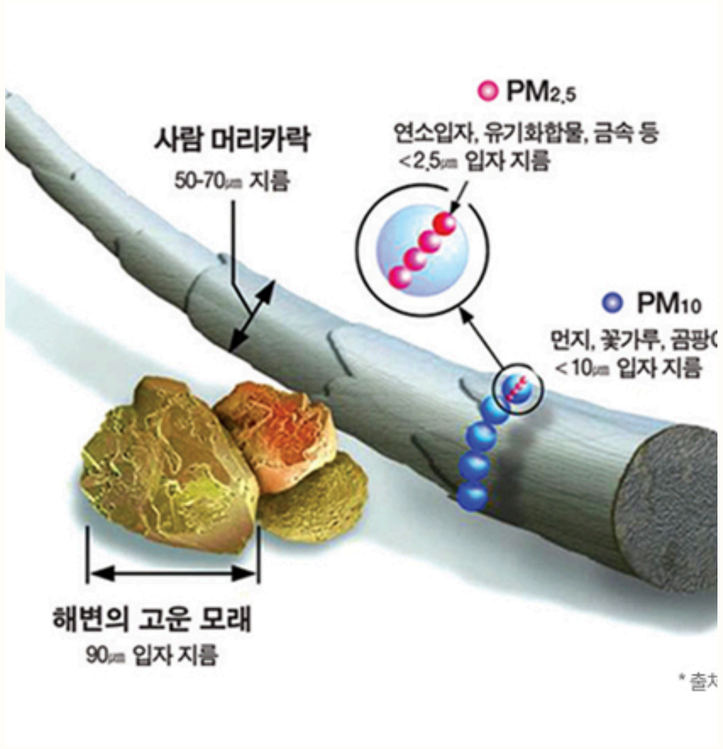
>> 건강

>> 경제

>> 환경오염

Group Project Presentation

주제 선정



미세먼지란?

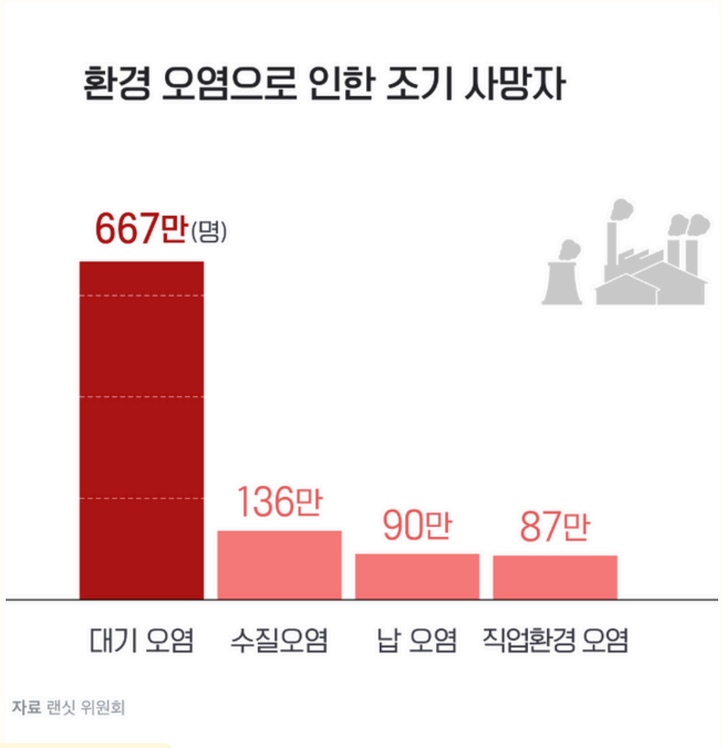
대기 중에 떠다니거나 흩날려 내려오는 입자상물질인 먼지 중 흡입성 먼지.

>> 미세먼지 : 입자의 지름이 10마이크로미터(㎛) 이하인 먼지(PM-10)

>> 초미세먼지 : 입자의 지름이 2.5마이크로미터(㎛) 이하인 먼지(PM-2.5)

주제 선정

Group Project Presentation



출처 : KBS 뉴스

미세먼지가 사회에 미치는 영향

>> 건강

대기오염 물질로 인한 호흡기 질환 증가

혈류를 통한 전신 작용 — 세포 노화 및 염증 반응 촉진

대기 오염에 따른 연간 사망자 추산 700만명

Group Project Presentation

주제 선정

출처 : 국민일보

미세먼지가 사회에 미치는 영향

>> 경제

소비자의 외부활동 위축 — 소비 경직으로 인한 **경제 침체**

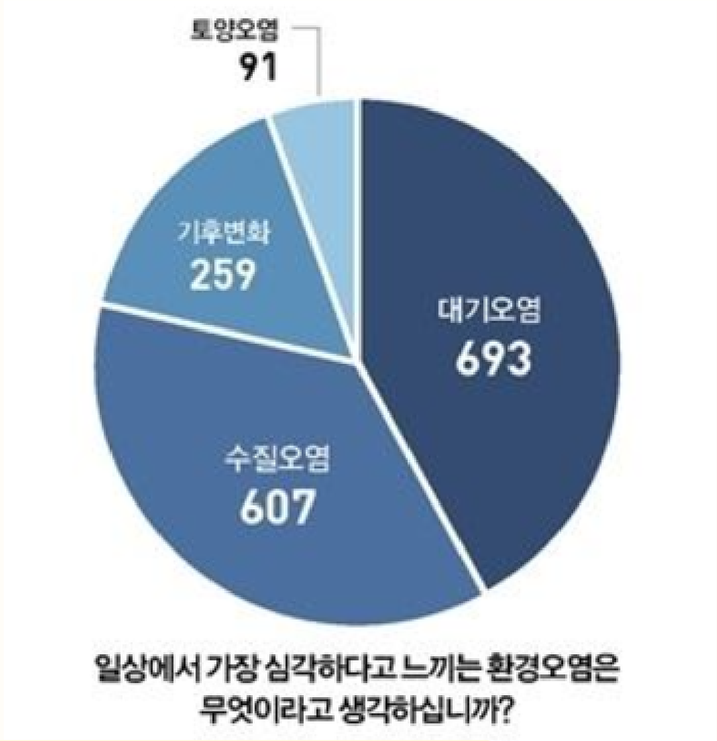
국내 여행을 기피하는 해외 여행객 증가

깨끗한 작업 환경이 필요한 반도체 산업

— 미세 공정 제품의 불량률 증가

Group Project Presentation

주제 선정



출처 : 아름다운 세상을 위한 공동체포럼

미세먼지가 사회에 미치는 영향

>> 환경 문제

미세먼지로 인한 산성비 — 토양 황폐화 및 생태계 피해

미세먼지로 인한 일조량 감소 — 작물 생육 저하 및 농민 피해

녹조 현상 심화 — 해양 생태계 파괴

Group Project Presentation

데이터 선정

기상 인자(풍향, 풍속, 강수량)

서울시 시간 평균 미세먼지 농도를 모델링 한 결과 미세먼지 농도에 가장 큰 영향을 미치는 것은 풍향과 풍속이었으며 이 두 변수를

박애경 외 2인,「서울시 미세먼지 농도에 영향을 미치는 요인 분석 : 기상 요인 및 장거리 이동 물질 중 일산화탄소를 중심으로」

또한 서풍계열 풍향비율도 한국의 초미세먼지 농도에 유의미한 (+)영향을 주는 것으로 나타났다. 이를 통해 중국 측 요인

박순애, 신현재,「한국의 초미세먼지(PM2.5)의 영향요인 분석 : 풍향을 고려한 계절성 원인을 중심으로」

기후변수 중에서는 강수량과 온도가 미세먼지에 음(-)의 영향을 미치고 황사는 양(+)의 영향을 미치는 것으로 나타났다.

김형건,「미세먼지 원인 요소들의 영향력 변화 추정: 경유를 중심으로」

데이터 선정

인구 수 / 자동차 대 수

경유 소비는 미세먼지를 증가시키고, 황사일수가 많아질수록 미세먼지의 농도가 높아진다는 결과는 선행연구들의 결과나 일반적인 기대에 부합하는 결과로 보인다. 두 모형 간

김형건, 「미세먼지 원인 요소들의 영향력 변화 추정: 경유를 중심으로」

둘째, 춘천에서 가장 높은 $PM_{2.5}$ 배출량을 나타내는 일차 배출원은 도로이동오염원으로, 조대입자에 비해 $PM_{2.5}$의 비율이 높은 배출원 특성을 지닌다.

조성환, 「강원도 춘천과 영월에서 측정한 미세먼지 농도 특성 및 고농도 원인 분석」

실질적으로 도로수송 부문 동력기관에서 배출되는 미세먼지(PM_{10})의 상당 부분은 노후경유차에서 배출되는 것이다.

배충식, 박현욱, 「미세먼지와 자동차」

Group Project Presentation

데이터 선정

전력 판매량

특히 석탄 화력 발전소는 1차 미세먼지 및 전구물질 중 하나인 황산화물(SO_x)의 주요한 배출원이다.

유태종, 유동현, 「화학 수송 모델과 기상 모델을 이용한 석탄 화력 발전소에서 배출되는 미세먼지의 거동 분석」

우리나라 석탄화력발전소에는 총 먼지 측정장치는 있으나 미세먼지 농도를 측정하기 위한 설비는 없는 실정으로 발전소 미세먼지 배출현황이 정확히 파악되지 못하는 실정이다. 한전과

강연욱 외 3인, 「국내 화력발전소 배출 미세먼지 저감 및 모니터링 기술 분석」

강원도는 인구수는 적으나 대규모의 시멘트 및 석회 공장이 위치해 있어 미세먼지 배출량이 높은 지역이다. 본 연구에서는 시멘트 및 석회 공장에서의 배출로

조성환, 「강원도 춘천과 영월에서 측정한 미세먼지 농도 특성 및 고농도 원인 분석」

데이터 선정

대기오염물질

특히 오존 농도의 증가는 대기 중 광화학스모그 때문이며 그 원인 물질이 질소산화물(NOx)과 휘발성유기화합물(VOCs)임을 고려할 때, 가스 상태의 물질이 입자상 물질로 변화하는 기전을 통해 알 수 있는 것은 기후변화가 대기 중 미세먼지 농도에 영향을 줄 수 있다는 점이다(Fuzzi et al., 2015). Tai,

이런 변화는 대기오염물질 거동(확산과 이동)에 영향을 주기 때문에 미세먼지 농도 변화와 관련성이 있을 것으로 생각한다.

(초)미세먼지는 주로 산업시설, 자동차, 난방 및 에너지 사용 등으로 인해 직접적으로 1차 배출되기도 하고, 황산염, 질산염과 같이 대기 중 반응에 의해 2차 생성되기도 한다.

데이터 전처리

Group Project Presentation

기상 인자 데이터

필요없는 열 — '지점' 제거

열 이름 가독성 높이기

열 순서 재배치

```
[ ]  1 weather_factor.head()
     2 # 기상인자 데이터 확인
```

	지점	지점명	일시	평균기온(°C)	평균상대습도(%)	월합강수량(00~24h만)(mm)	평균풍속(m/s)	최다풍향(16방위)
0	90	속초	2011-01	-2.7	33	12.0	2.7	290
1	90	속초	2011-02	2.1	62	105.6	2.2	290
2	90	속초	2011-03	5.3	45	20.4	2.7	290
3	90	속초	2011-04	11.4	54	115.7	3.2	290
4	90	속초	2011-05	14.3	71	65.3	2.3	360

	일시	시군구	평균기온	평균상대습도	강수량	평균풍속	최다풍향
0	2011-01	속초	-2.7	33	12.0	2.7	290
1	2011-02	속초	2.1	62	105.6	2.2	290
2	2011-03	속초	5.3	45	20.4	2.7	290
3	2011-04	속초	11.4	54	115.7	3.2	290
4	2011-05	속초	14.3	71	65.3	2.3	360

데이터 전처리

Group Project Presentation

기상 인자 데이터

결측치 파악

```
missing_values = weather_factor.isnull().sum()
print(missing_values)
```

```
일시              0
시군구             0
평균기온           15
평균상대습도         42
강수량         27426
평균풍속           14
최다풍향           93
dtype: int64
```

```
weather_factor['강수량'] = weather_factor['강수량'].fillna(0)
```

강수량 결측치 >> 0으로 대체

```
weather_factor.fillna(method='ffill', inplace=True)
```

강수량 외의 결측치 >> 하루 전 데이터로 대체

데이터 전처리

Group Project Presentation

기상 인자 데이터

누락 데이터 확인

```python
date_range = pd.date_range(start='2011-01-01', end='2019-12-31')

missing_dates = {}
regions = weather_factor['시군구'].unique()

for region in regions:
    region_data = weather_factor[weather_factor['시군구'] == region]
    available_dates = pd.to_datetime(region_data['일시']).dt.date
    missing = set(date_range.date) - set(available_dates)
    missing_dates[region] = sorted(missing)

for region, missing in missing_dates.items():
    print(f"Region: {region}, Missing Dates: {len(missing)}")
    print(missing)
```

필요한 날짜 범위 생성

지역별 데이터 확인

출력

데이터 전처리

Group Project Presentation

기상 인자 데이터

```
missing_dates = [pd.Timestamp('2011-07-25'), pd.Timestamp('2011-08-18')]
region_name = '정선군'
```

데이터 프레임에 누락된 날짜 추가

```
for missing_date in missing_dates:
    weather_factor = pd.concat([
        weather_factor,
        pd.DataFrame({'시군구': [region_name], '일시': [missing_date], '평균기온': [None], '평균상대습도': [None],
                      '강수량': [None], '평균풍속': [None], '최다풍향': [None]})
    ], ignore_index=True)
```

누락된 행 추가 — 시군구 / 일시

데이터 전처리

Group Project Presentation

기상 인자 데이터

```
14 weather_factor['일시'] = pd.to_datetime(weather_factor['일시'])
15 weather_factor = weather_factor.sort_values(by=['시군구', '일시'])
```

지역 / 날짜 기준 정렬

```
18 weather_factor_interpolated = weather_factor.groupby('시군구').apply(lambda group: group.interpolate(method='linear'))
```

보간법으로 결측값 처리

보간법 : 결측치의 앞, 뒤 데이터를 이용하여 적정값을 추정하는 방법

```
21 weather_factor = weather_factor.drop_duplicates(subset=['일시', '시군구'], keep='first')
```

중복행 제거

데이터 전처리

Group Project Presentation

기상 인자 데이터

새로운 열 생성

>> 서풍 계열

>> 계절

논문, 「한국의 초미세먼지(PM2.5)의 영향요인 분석 : 풍향을 고려한 계절성 원인을 중심으로」

중 **'서풍 계열 풍향의 비율에 따른 국내 미세먼지 농도의 유의미한 변화'**를 근거로 함.

```
data_plus['서풍계열'] = data_plus['최다풍향'].apply(
    lambda x: 3 if 247.5 <= x <= 292.5 else
              2 if 225 <= x <= 315 else
              1 if 202.5 <= x <= 337.5 else
              0)
```

'서풍 계열' 열 생성

```
def assign_season(month):
    if month in [3, 4, 5]:
        return '봄'
    elif month in [6, 7, 8]:
        return '여름'
    elif month in [9, 10, 11]:
        return '가을'
    else:
        return '겨울'
```

'계절' 열 생성

데이터 전처리

Group Project Presentation

자동차 데이터

필요없는 열 제거

열 이름 가독성 높이기

강원지역 / 2011년~2019년

— **데이터 필터링**

```
1 cars.columns
```

```
Index(['월(Monthly)', '시도명', '시군구', '관용', '자가용', '영업용', '계', '관용.1', '자가용.1',
       '영업용.1', '계.1', '관용.2', '자가용.2', '영업용.2', '계.2', '관용.3', '자가용.3',
       '영업용.3', '계.3', '관용.4', '자가용.4', '영업용.4', '계.4'],
      dtype='object')
```

필요한 데이터만 선별하여 저장

```
1 cars
```

	일시	시군구	차량수
0	2011-01-01	강릉시	89310
1	2011-01-02	강릉시	89310
2	2011-01-03	강릉시	89310
3	2011-01-04	강릉시	89310
4	2011-01-05	강릉시	89310

데이터 전처리

Group Project Presentation

자동차 데이터

기상인자 / 대기오염 데이터
>> **일** 단위 데이터

자동차 데이터
>> **월** 단위 데이터

= **데이터 단위 통일** 필요

데이터 복사하여 동일값 처리

예시) 1월 : 200 > > 1월 1일 : 200 … 1월 31일 : 200

```
daily_data = []

for i, row in cars.iterrows():
    start_date = pd.to_datetime(row['일시'] + '-01')  해당 월의 첫날

    days_in_month = start_date.days_in_month  해당 월의 모든 날짜 생성
    dates = pd.date_range(start=start_date, periods=days_in_month)

    daily_values = [row['차량수']] * days_in_month  동일값 복사

    daily_data.extend(zip(dates, [row['시군구']] * days_in_month, daily_values))  날짜와 '시군구' 결합

cars = pd.DataFrame(daily_data, columns=['일시', '시군구', '차량수'])
```

일 단위 데이터 프레임 생성

데이터 전처리

Group Project Presentation

인구 수 데이터

자동차 데이터와 유사한 데이터 전처리

>> 데이터 단위 통일

```
daily_data = []

for i, row in population.iterrows():
    start_date = pd.to_datetime(row['일시'] + '-01')

    days_in_month = start_date.days_in_month
    dates = pd.date_range(start=start_date, periods=days_in_month)

    daily_values = [row['인구수']] * days_in_month

    daily_data.extend(zip([row['시군구']] * days_in_month, dates, daily_values))

population = pd.DataFrame(daily_data, columns=['시군구', '일시', '인구수'])
```

	시군구	일시	인구수
0	속초시	2011-01-01	36636
1	속초시	2011-01-02	36636
2	속초시	2011-01-03	36636
3	속초시	2011-01-04	36636
4	속초시	2011-01-05	36636

데이터 전처리

Group Project Presentation

전력 판매량 데이터

자동차 데이터와 유사한
데이터 전처리

>> 데이터 단위 통일

```
daily_data = []

for i, row in electricity.iterrows():
    start_date = pd.to_datetime(row['일시'] + '-01')

    days_in_month = start_date.days_in_month
    dates = pd.date_range(start=start_date, periods=days_in_month)

    daily_values = [row['전력량']] * days_in_month

    daily_data.extend(zip([row['시군구']] * days_in_month, dates, daily_values))

electricity = pd.DataFrame(daily_data, columns=['시군구', '일시', '전력량'])
```

	시군구	일시	전력량
0	강릉시	2011-01-01	169,825
1	강릉시	2011-01-02	169,825
2	강릉시	2011-01-03	169,825
3	강릉시	2011-01-04	169,825
4	강릉시	2011-01-05	169,825

데이터 전처리

Group Project Presentation

대기오염 데이터

SO2 : 아황산가스
CO : 일산화탄소
O3 : 오존
NO2 : 이산화질소

>> 미세먼지 농도에 영향을 미치는 오염물질들

	지역	측정일시	SO2	CO	O3	NO2	PM10
0	강원 춘천시	2010010101	0.0060	0.4	0.024	0.0080	30.0
1	강원 춘천시	2010010102	0.0060	0.5	0.017	0.0130	28.0
2	강원 춘천시	2010010103	0.0040	0.4	0.019	0.0070	28.0
3	강원 춘천시	2010010104	0.0040	0.4	0.020	0.0060	29.0
4	강원 춘천시	2010010105	0.0050	0.6	0.016	0.0100	28.0

다른 데이터와 맞지 않는 지역 표기 형식

시간 단위로 기록된 데이터

데이터 전처리

Group Project Presentation

대기오염 데이터

데이터 단위 통일

시간 단위 데이터를
일(평균) 단위 데이터로 **변환**

```
2 air['지역'] = air['지역'].str.replace('강원 ', '', regex=False)
```
시군구 형식 통일

'측정일시'를 일시로 변환

```
3 air['일시'] = air['측정일시'].str[:8]
4
5 air = air.groupby(['일시', '지역']).mean(numeric_only=True).reset_index()
```
'일시'와 '지역'을 기준으로 그룹화하여 **평균** 계산

```
2 air.replace(-999, np.nan, inplace=True)
```
-999인 값 >> 결측치로 대체

	일시	시군구	SO2	CO	O3	NO2	PM10
3285	2011-01-01	강릉시	0.006083	0.458333	0.026208	0.012250	25.500000
3286	2011-01-01	고성군	0.002596	0.316667	0.038167	0.003554	25.666667
3287	2011-01-01	동해시	0.003708	0.433333	0.029708	0.012792	17.166667
3288	2011-01-01	삼척시	0.004708	0.504167	0.030375	0.010083	24.750000
3289	2011-01-01	양구군	0.002021	0.595833	0.024042	0.005071	32.083333

데이터 전처리

Group Project Presentation

데이터 병합 및 결측치 처리

강원 지역의 '시군구' 데이터 중
강릉 / 동해 / 원주 / 정선 / 춘천
지역 선별

>> 관광객 방문 현황과 인구수 통계를 바탕으로 선정

```
[ ]  1 target_cities = ['춘천시', '정선군', '강릉시', '원주시', '동해시']
```

시군별 전체 관광객 방문 현황

단위(명)

구분	2024년 8월			2024년 7월			2023년 8월			전월대비		전년대비	
시군명	외지인	외국인	전체	외지인	외국인	전체	외지인	외국인	전체	증감량	증감률	증감량	증감률
춘천시	2,808,096	16,219	2,824,315	2,389,772	20,387	2,410,159	2,927,204	12,334	2,939,538	414,156	17.18%	-115,223	-3.92%
원주시	3,143,749	6,078	3,149,827	2,753,905	6,758	2,760,663	2,909,827	3,852	2,913,679	389,164	14.10%	236,148	8.10%
강릉시	3,831,461	8,847	3,840,308	2,973,705	8,630	2,982,335	3,595,890	5,433	3,601,323				
동해시	1,275,001	3,717	1,278,718	1,050,168	3,537	1,053,705	1,221,217	2,822	1,224,039				
태백시	650,832	697	651,529	499,524	1,041	500,565	589,841	408	590,249				
속초시	3,163,948	7,629	3,171,577	2,351,759	8,160	2,359,919	2,855,048	5,025	2,860,073				
삼척시	1,486,986	2,514	1,489,500	1,124,008	2,415	1,126,423	1,368,731	2,037	1,370,768				
홍천군	2,702,812	4,059	2,706,871	2,043,426	4,165	2,047,591	1,953,109	2,246	1,955,355				
횡성군	1,347,861	1,749	1,349,610	978,359	2,385	980,744	1,293,282	1,428	1,294,710				
영월군	981,722	808	982,530	686,672	892	687,564	931,382	563	931,945				
평창군	2,176,804	3,541	2,180,345	1,483,360	3,114	1,486,474	2,070,985	2,676	2,073,661				
정선군	1,226,075	924	1,226,999	899,759	986	900,745	1,139,474	731	1,140,205				

데이터 전처리

Group Project Presentation

데이터 병합 및 결측치 처리

앞서 처리한 각 데이터들을 하나로 합침

```
1 data = pd.concat([cars, weather_factor, air, population, electricity], axis=1)
```

	일시	시군구	차량수	평균기온	평균상대습도	강수량	평균풍속	최다풍향	SO2	CO	...	인구수	전력량	PM10	PM10-1d	서풍계열	계절_숫자	계절_가을	계절_겨울	계절_봄	계절_여름
0	2011-01-01	강릉시	89310	-1.5	71.4	6.0	3.2	320	0.006083	0.458333	...	89915	169825	25.500000	35.541667	1	4	False	True	False	False
1	2011-01-02	강릉시	89310	0.6	66.4	0.5	1.9	270	0.006625	0.454167	...	89915	169825	20.708333	25.500000	3	4	False	True	False	False
2	2011-01-03	강릉시	89310	-0.9	83.6	6.5	2.1	270	0.005750	0.487500	...	89915	169825	20.916667	20.708333	3	4	False	True	False	False
3	2011-01-04	강릉시	89310	-0.1	47.1	0.0	3.7	250	0.005667	0.533333	...	89915	169825	36.666667	20.916667	3	4	False	True	False	False
4	2011-01-05	강릉시	89310	-0.6	46.5	0.0	4.7	250	0.006048	0.504762	...	89915	169825	50.333333	36.666667	3	4	False	True	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
16063	2019-12-27	춘천시	134390	-2.5	59.3	0.0	1.1	270	0.002375	0.570833	...	123006	163734041	40.541667	50.541667	3	4	False	True	False	False
16064	2019-12-28	춘천시	134390	-1.5	66.0	0.0	0.4	20	0.002938	0.750000	...	123006	163734041	55.312500	40.541667	0	4	False	True	False	False
16065	2019-12-29	춘천시	134390	0.5	76.3	0.0	0.3	360	0.003458	0.966667	...	123006	163734041	68.187500	55.312500	0	4	False	True	False	False
16066	2019-12-30	춘천시	134390	2.2	68.8	0.5	1.6	250	0.002417	0.737500	...	123006	163734041	44.208333	68.187500	3	4	False	True	False	False
16067	2019-12-31	춘천시	134390	-6.9	44.3	0.0	2.0	200	0.002083	0.412500	...	123006	163734041	27.687500	44.208333	0	4	False	True	False	False

데이터 전처리

Group Project Presentation

데이터 병합 및 결측치 처리

각 열 별 결측치 처리

>> **선형 보간법**

>> 결측치 0으로 대체

>> **최빈값**

>> 결측치 있는 행 삭제

```
missing_counts = data.isnull().sum() 결측치 개수 확인

missing_percentage = (data.isnull().sum() / len(data)) * 100  결측치 비율 확인
```

각 열 별 결측치 처리

```
data['평균기온'] = data['평균기온'].interpolate(method='linear')  선형 보간법 사용
```

연속적인 데이터의 결측치를 처리할 때 효과적인 방법

>> 연속성 유지 가능

>> 데이터 왜곡 최소화

>> 단순해서 계산 비용이 낮음

데이터 전처리

Group Project Presentation

데이터 병합 및 결측치 처리

각 열 별 결측치 처리

>> **선형 보간법**

>> 결측치 0으로 대체

>> **최빈값**

>> 결측치 있는 행 삭제

각 열 별 결측치 처리

```
 3 data['평균상대습도'] = data['평균상대습도'].interpolate(method='linear')
 5 data['평균풍속'] = data['평균풍속'].interpolate(method='linear')
10 data['SO2'] = data['SO2'].interpolate(method='linear')
11 data['CO'] = data['CO'].interpolate(method='linear')
12 data['O3'] = data['O3'].interpolate(method='linear')
13 data['NO2'] = data['NO2'].interpolate(method='linear')
```

선형 보간법 사용

```
 4 data['강수량'] = data['강수량'].fillna(0)
```

결측치 0으로 대체

```
6 most_frequent_wind_direction = data['최다풍향'].mode()[0]
7 data['최다풍향'] = data['최다풍향'].fillna(most_frequent_wind_direction)
```

최빈값 사용

최빈값 : 데이터셋에서 가장 빈도수가 높은 값

범주형 / 명목형 데이터에 주로 사용함

```
16 data = data.dropna(subset=['PM10'])
```

PM10의 결측치가 있는 행 삭제

Group Project Presentation

모델 학습

피처 / 타겟 데이터 분리

```
# PM10 예측을 위해 Feature와 Target을 정의
features = ['차량수', '평균기온', '평균상대습도', '강수량', '평균풍속', '최다풍향',
            'SO2','CO', 'O3', 'NO2', '인구수', '전력량', 'PM10-1d', '서풍계열',
            '계절_가을', '계절_겨울', '계절_봄', '계절_여름']
target = 'PM10'

# '일시' 열을 datetime 형식으로 변환
data_plus['일시'] = pd.to_datetime(data_plus['일시'])

# 연도 정보 추출
data_plus['연도'] = data_plus['일시'].dt.year
```

```
# Train/Test 데이터 분리 (2011-2017년: Train, 2018-2019년: Test)
train_data = data_plus[data_plus['연도'].between(2011, 2017)]
test_data = data_plus[data_plus['연도'].between(2018, 2019)]

# Feature와 Target 분리
X_train = train_data[features]
y_train = train_data[target]
X_test = test_data[features]
y_test = test_data[target]
```

모델 학습

Group Project Presentation

하이퍼 파라미터 최적화

하이퍼 파라미터 최적화

>> 최적 하이퍼 파라미터 도출

예시) 랜덤 포레스트의 하이퍼 파라미터 최적화

```
param_grid = {'n_estimators': [50, 100, 200],
              'max_depth': [10, 20, 30]}
```

트리 개수

최대 깊이

그리드 서치 방식 사용

```
grid_search = GridSearchCV(
    estimator=rf_model,
    param_grid=param_grid,
    cv=3
    scoring='neg_mean_squared_error',
    verbose=2,
    n_jobs=-1
```

3-FOLD 교차 검증

MSE >> 음수 반환

병렬 처리

```
Best Parameters: {'max_depth': 10, 'n_estimators': 200}
```

모델 학습

LSTM

RNN(순환신경망)의 한 종류

순환 구조를 통해 **과거의 학습을 현재 학습에 반영**하는 딥러닝 알고리즘

RNN의 **기울기 소멸 문제**를 **해결**하기 위해 개발된 모델

은닉층이 깊어질수록 기울기가 작아져 **학습 성능이 저하되는 문제**

직전 데이터 뿐만 아니라, 더 거시적으로 과거 데이터를 고려하여 미래 데이터를 예측 가능함.

MIN-MAX SCALAR로 데이터 정규화

```
23 scaler_X_lstm = MinMaxScaler()
24 scaler_y_lstm = MinMaxScaler()
25 X_train = scaler_X_lstm.fit_transform(X_train)
26 X_test = scaler_X_lstm.transform(X_test)
27 y_train = scaler_y_lstm.fit_transform(y_train.reshape(-1, 1))
28 y_test = scaler_y_lstm.transform(y_test.reshape(-1, 1))
```

입력 데이터와 타겟 데이터를 별도로 정규화
>> 각각의 데이터 특성에 맞춰 독립적으로 변환함.

MIN-MAX SCALAR

데이터의 최소값을 0, 최대값을 1로 변환하는 데이터 정규화 방식

분류 모델보다 **회귀 모델에 주로 사용**함

모델 학습

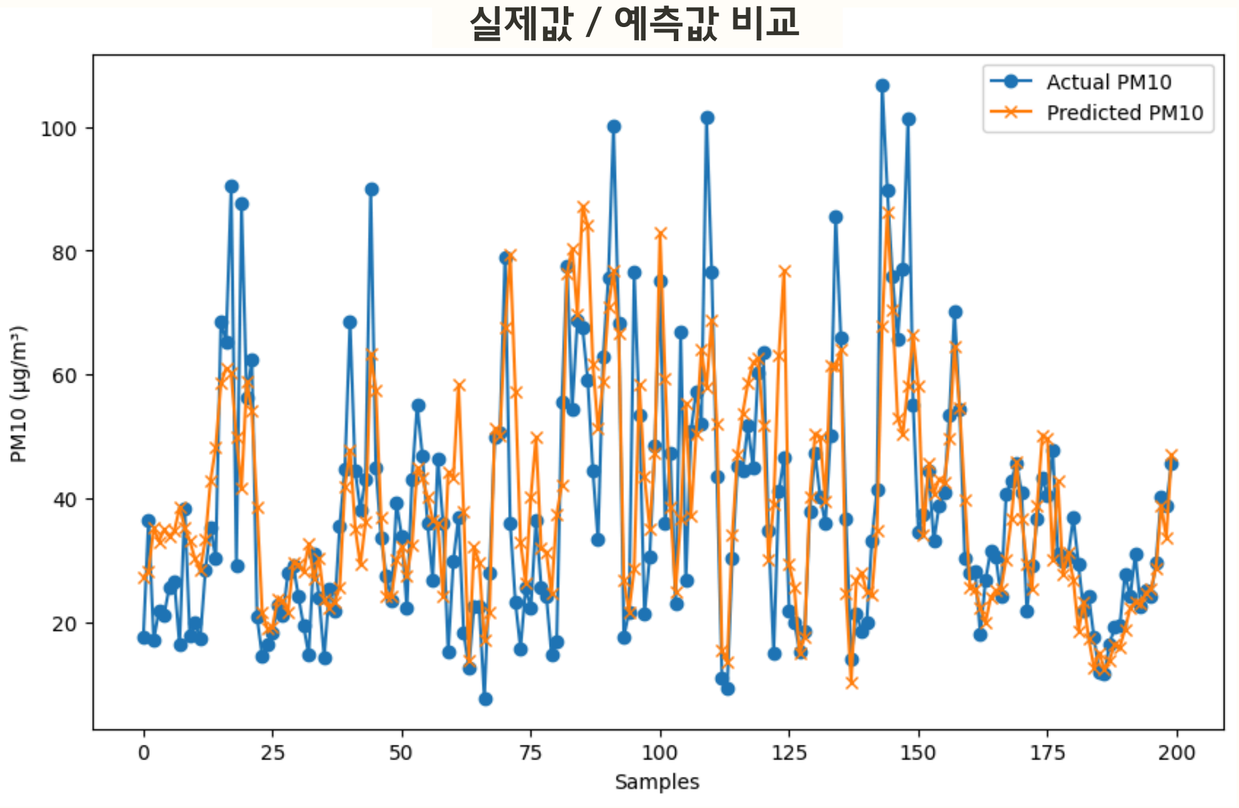
LSTM

LSTM의 출력층에 적용할 **활성화 함수**로 **RELU 함수 선택**

5개의 은닉층 설계

LSTM 모델의 적합도

— 결정계수 : **64.30%** / RMSE : **13.24**



모델 학습

랜덤 포레스트

모델의 적합도 측정

>> **결정계수**(R^2)를 통해 예측 정확도 파악

결정계수가 높을 수록 적합한 모델

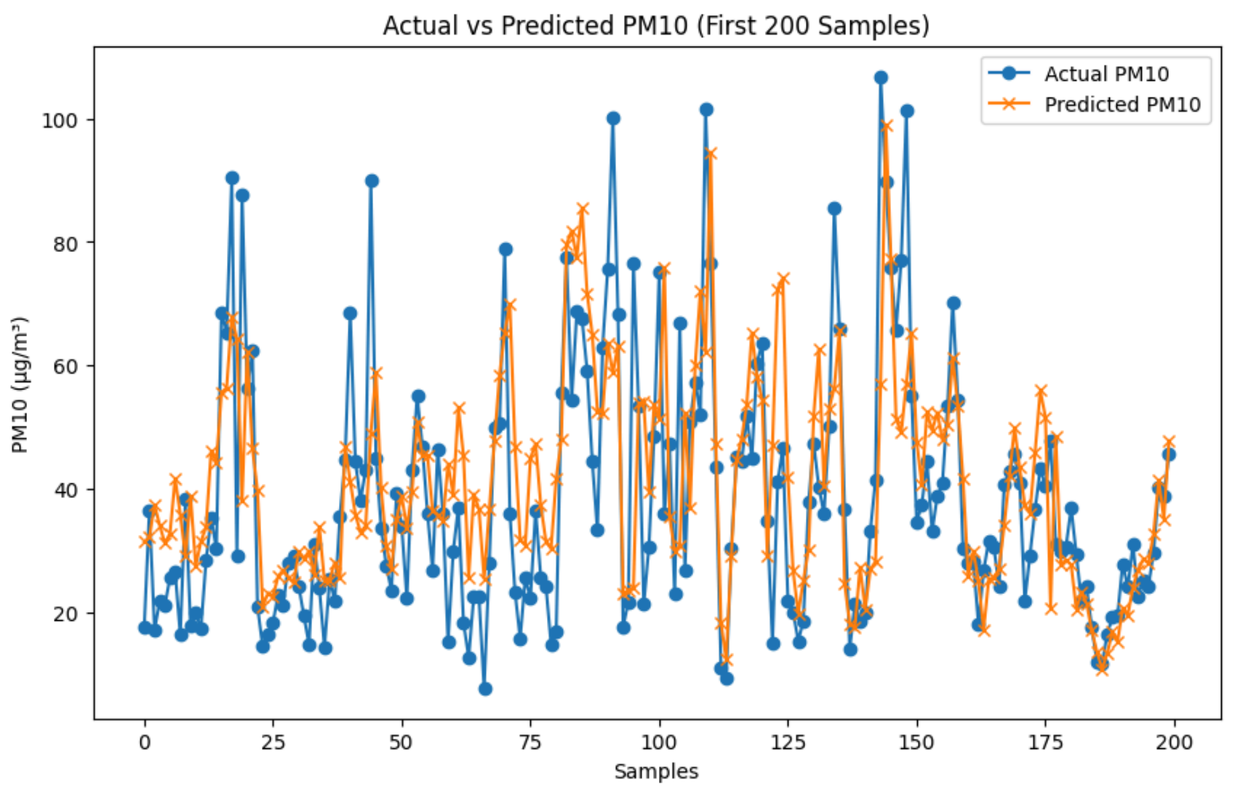
>> 모델에서 **예측한 값과 실제 값 간 평균의 차이** (RMSE)를 측정하여, 모델의 예측 정확도 파악

RMSE가 낮을 수록 적합한 모델

랜덤 포레스트 모델의 적합도

— 결정계수 : **62.96%** / RMSE : **13.48**

실제값 / 예측값 비교



모델 학습

랜덤 포레스트 - 잔차 학습

선형 회귀 모델 학습 >> **잔차 계산**

실제값 - 예측값

랜덤 포레스트 **잔차 학습**

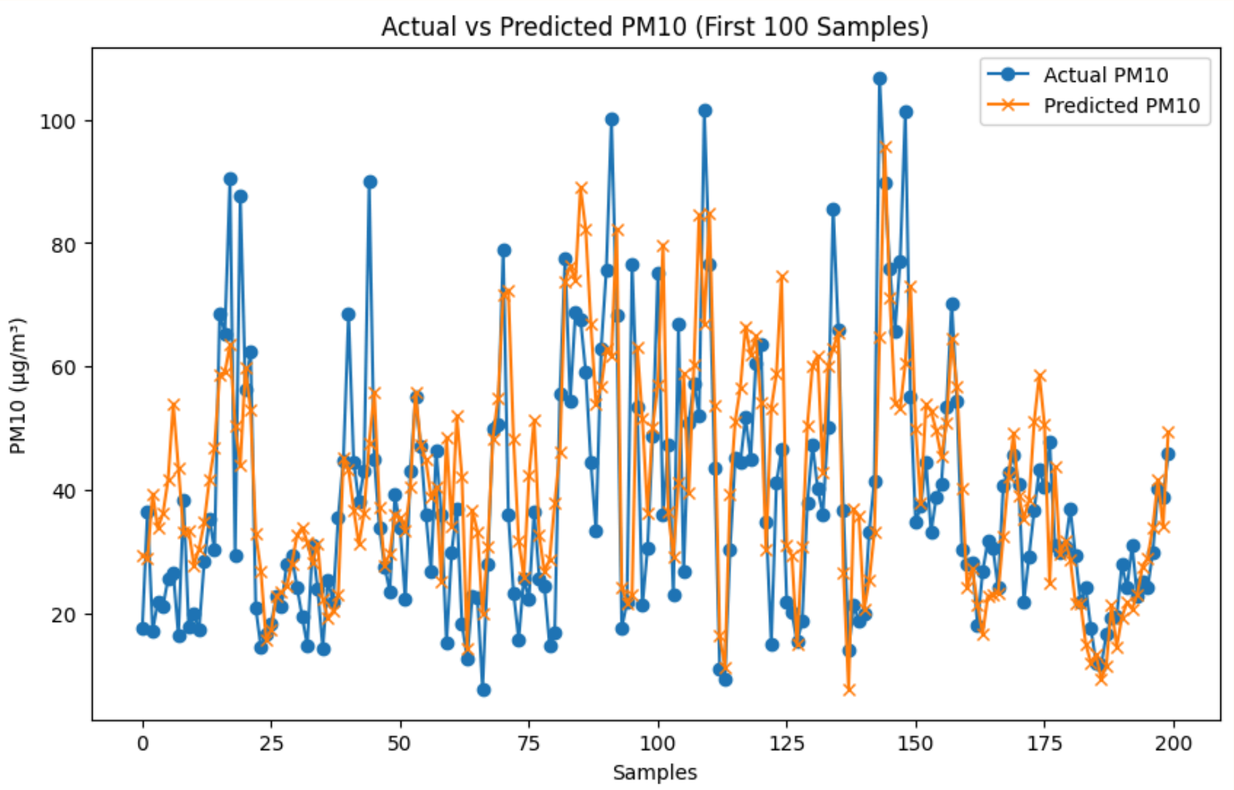
입력값과 출력값 간의 차이를 학습하는 방법

잔차를 통해 선형 회귀 모델이 설명하지 못한 **비선형적인 부분을 랜덤 포레스트가 보완하는 효과**를 얻음.

랜덤 포레스트 (+선형 회귀) 모델의 적합도

— 결정계수 : **63.12%** / RMSE : **13.46**

실제값 / 예측값 비교



모델 학습

XG 부스트

여러 앙상블(ENSEMBLE) 기법 중 2가지

>> 배깅 / 부스팅

배깅 알고리즘 : 랜덤 포레스트

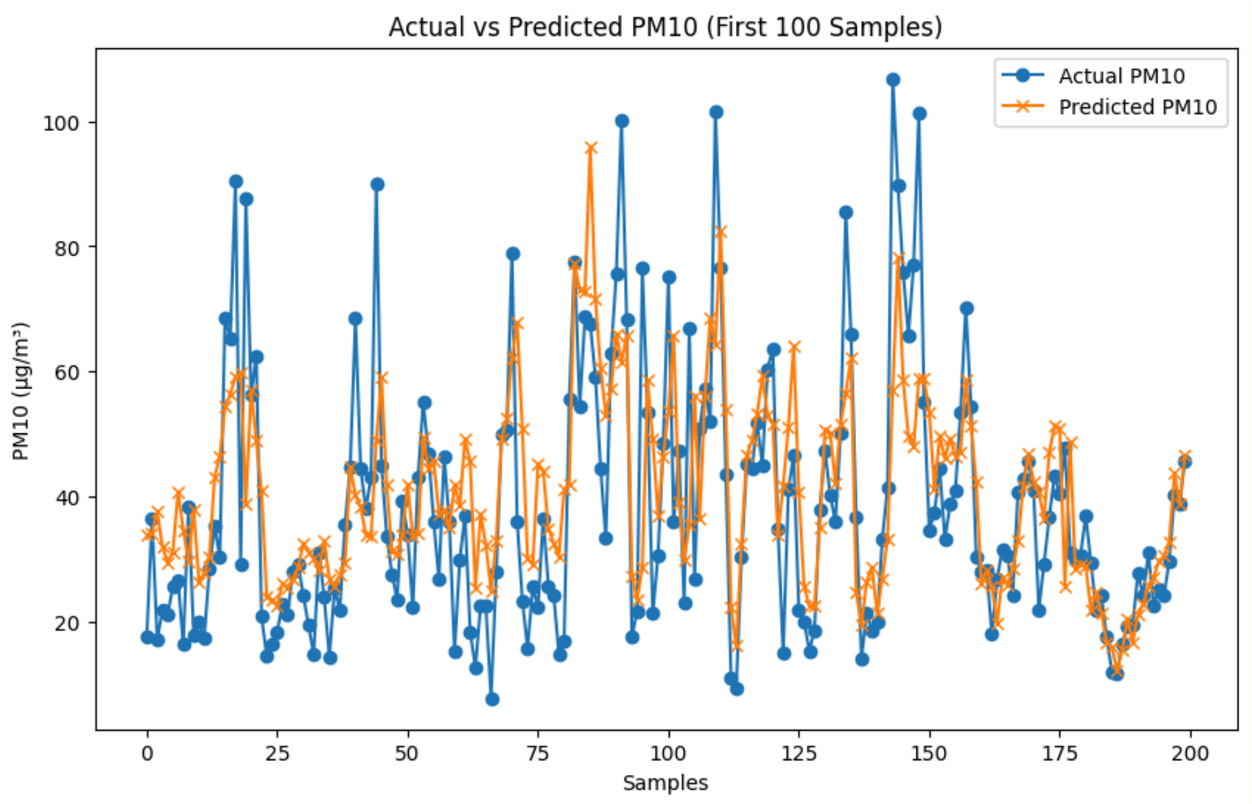
부스팅 알고리즘 : XG 부스트

오분류된 객체에 높은 가중치를 부여하는 방식으로 정확도를 높이는 모델

XG 부스트 모델의 적합도

— 결정계수 : **62.28%** / RMSE : **13.62**

실제값 / 예측값 비교



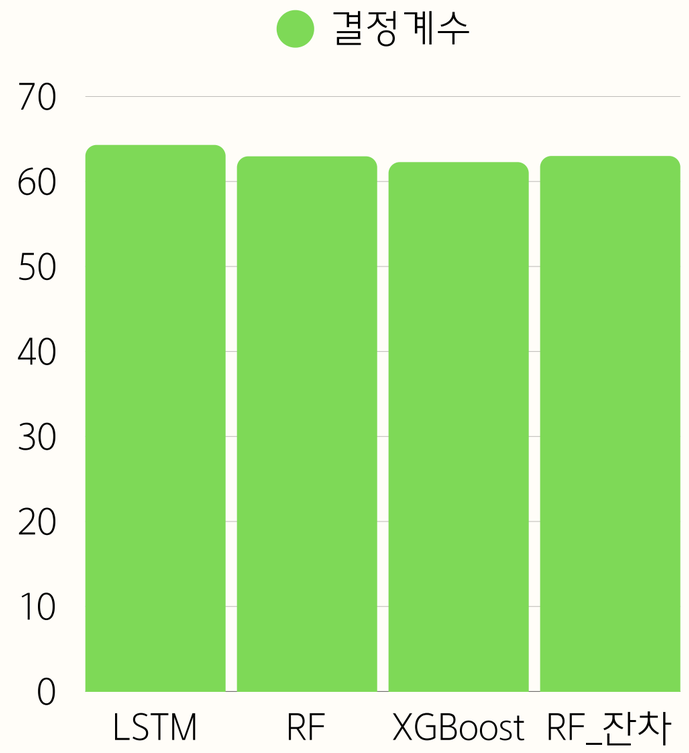
모델 학습 결과

가장 잘 학습된 모델

LSTM 모델

>> 64.30%의 정확도

각 모델 간 **유의미한 성능 차이를 보이지 않음**



피드백

- 데이터셋에 유의미한 변수 추가 및 삭제
- 모델 간 유의미한 성능 차이 관측 불가
- 64.30%의 낮은 예측 정확도 >> 성능 향상 필요

감사합니다