21KDT 데이터 사이언스/엔지니어링 전문가 1,2회차
데이터 시각화 프로젝트

기상환경 데이터를 활용한 서울시 미세먼지 예측

11조 UTU 박경빈 이우재 허재혁 강아름

목차

구성원

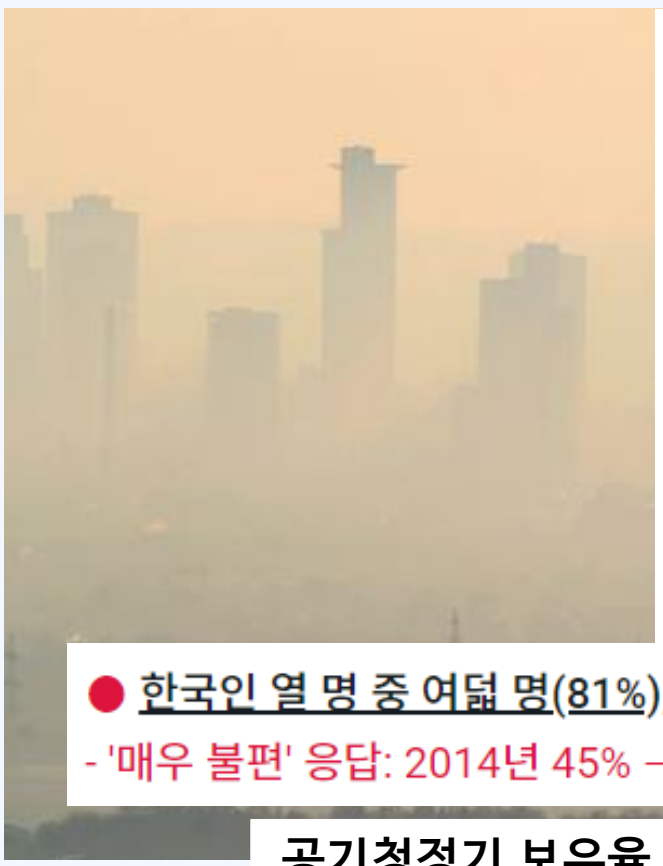
박경빈(팀장)
모델링 & 시각화

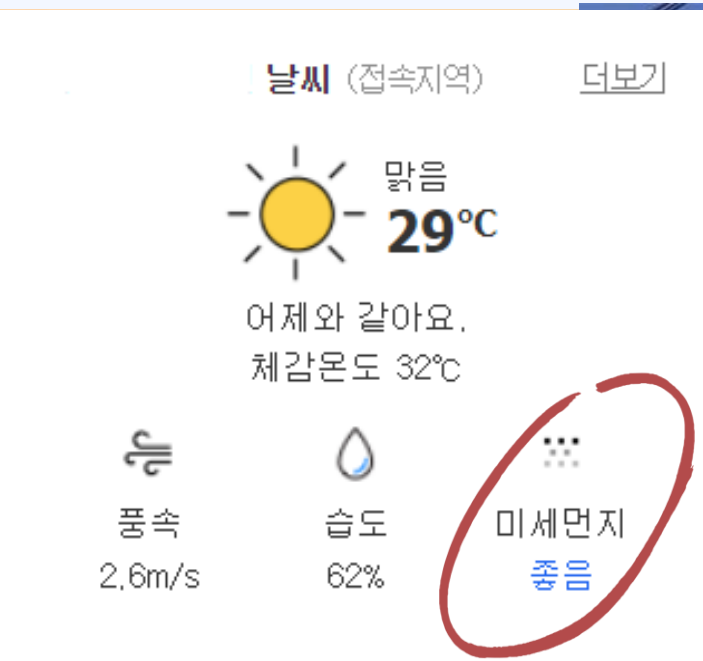
강아름
모델링 & 시각화

이우재
전처리 & 모델링

허재혁
전처리 & 모델링

프로젝트 기획 배경 및 목표





● 한국인 열 명 중 여덟 명(81%), '미세먼지 때문에 불편하다'

- '매우 불편' 응답: 2014년 45% → 2017년 5월 57% → 2019년 1월 57%

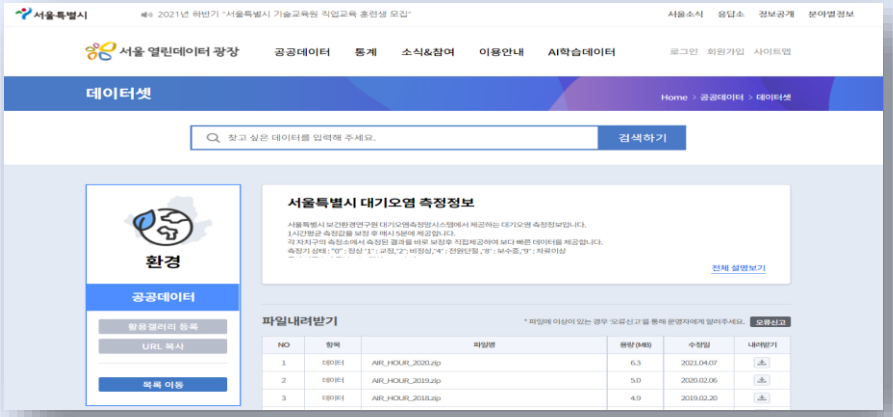
공기청정기 보유율 2006년 15% 수준,
고농도 미세먼지 재난 상황을 겪었던 2019년 상반기 40%대 진입, 2020년 50% 돌파

→ 미세먼지에 대한 관심 증가

통계 출처: 한국갤럽 마켓70 2021 (5) 주방·생활 가전제품 26종 보유율 조사일 : 2021/03/18

데이터 수집

미세먼지 data 수집



	A	B	C	D	E	F	G
1	측정일시	측정소 코드	측정항목 코드	평균값	측정기 상태	국가 기준초과 구분	지자체 지군초과 구분
2	43101	101	1	0.006	0	0	0
3	43101	101	3	0.021	0	0	0
4	43101	101	5	0.4	0	0	0
5	43101	101	6	0.022	0	0	0
6	43101	101	8	29	0	0	0
7	43101	101	9	17	0	0	0
8	43101	102	1	0.004	0	0	0
9	43101	102	3	0.02	0	0	0
10	43101	102	5	0.5	0	0	0
11	43101	102	6	0.02	0	0	0
12	43101	102	8	34	0	0	0
13	43101	102	9	19	0	0	0
14	43101	103	1	0.004	0	0	0
15	43101	103	3	0.023	0	0	0
16	43101	103	5	0.3	0	0	0
17	43101	103	6	0.013	0	0	0
18	43101	103	8	29	0	0	0
19	43101	103	9	14	0	0	0
20	43101	104	1	0.003	0	0	0
21	43101	104	3	0.018	0	0	0
22	43101	104	5	0.6	0	0	0

서울 열린데이터 광장:
서울특별시 대기오염 측정정보

기상 data 수집

Station Code	Name	Date	Temp	Prec	WS	Humi	Pres	cbwd
108	서울	2018-01-01	-1.3		1.4	39.1	1016.8	290
108	서울	2018-01-02	-1.8		1.8	42	1018.1	290
108	서울	2018-01-03	-4.7		2.2	42.3	1019.9	290
108	서울	2018-01-04	-4.7		1.4	43	1016.5	290
108	서울	2018-01-05	-3		1.7	48.4	1010.3	290
108	서울	2018-01-06	-2.8		1.5	45.8	1012.9	290
108	서울	2018-01-07	-0.8		0.8	42.1	1010.7	90
108	서울	2018-01-08	1.3	0.9	1.8	51.9	1000.6	270
108	서울	2018-01-09	-4.2	0.5	2.9	59.4	1002.8	290
108	서울	2018-01-10	-7.5	0.3	3	52	1007.9	290
108	서울	2018-01-11	-11.1		2.8	49.8	1015.9	290
108	서울	2018-01-12	-10.2	0	1.7	35.4	1020.6	290
108	서울	2018-01-13	-4.4	0.4	0.8	67.3	1021.1	50
108	서울	2018-01-14	0.6		1.2	64.1	1017.9	90
108	서울	2018-01-15	4.7	0.2	1.4	63.9	1012.8	290
108	서울	2018-01-16	3.5	0	0.7	59.1	1007.6	20
108	서울	2018-01-17	4.5		1.2	64.1	1007.2	290
108	서울	2018-01-18	2.1	0	1.9	59.6	1012.9	290
108	서울	2018-01-19	1		1.6	43	1013.5	290
108	서울	2018-01-20	2.2		1.6	57.6	1012	270
108	서울	2018-01-21	0.4		1	36.6	1012.7	290
108	서울	2018-01-22	0.2	3.3	1.5	56.5	1005.2	290
108	서울	2018-01-23	-11.7		3.7	42	1013.4	290
108	서울	2018-01-24	-14.2		3.4	31.6	1018.3	290
108	서울	2018-01-25	-14		2.8	32.6	1019.1	340
108	서울	2018-01-26	-14.8		2.6	34.9	1018.9	320
108	서울	2018-01-27	-9.9		1.4	37.4	1017.8	270
108	서울	2018-01-28	-6.3		2.6	36	1011.6	290

기상청 기상자료개방포털:
서울특별시 일일 종관기상관측 정보

데이터 전처리

데이터 병합

Station code	Date	SO2	CO	O3	NO2	PM10	PM25	Temp	Prec	WS	Humi	Pres	cbwd
111121	2018-01-01	0.004833333333	0.5541666667	0.01458333333	0.02929166667	32.66666667	17.79166667	-1.3	0	1.4	39.1	1016.8	290
111122	2018-01-01	0.005416666667	0.5416666667	0.011375	0.03375	49.54166667	19.89974937	-1.3	0	1.4	39.1	1016.8	290
111123	2018-01-01	0.007208333333	0.6041666667	0.01529166667	0.03025	34.375	17.33333333	-1.3	0	1.4	39.1	1016.8	290
111124	2018-01-01	0.006333333333	0.4333333333	0.01154166667	0.03416666667	37.125	19.89974937	-1.3	0	1.4	39.1	1016.8	290
111125	2018-01-01	0.003666667	0.495833333	0.02083333333	0.02870833333	36.20833333	19.89974937	-1.3	0	1.4	39.1	1016.8	290
111131	2018-01-01	0.004791666667	0.3416666667	0.01116666667	0.02933333333	39.08333333	19.33333333	-1.3	0	1.4	39.1	1016.8	290
111141	2018-01-01	0.005416666667	0.6708333333	0.012	0.0375	41.58333333	21.70833333	-1.3	0	1.4	39.1	1016.8	290
111142	2018-01-01	0.0055	0.495833333	0.009291666667	0.03291666667	48	22.875	-1.3	0	1.4	39.1	1016.8	290
111143	2018-01-01	0.005	0.65	0.007833333333	0.03408333333	48.29166667	19.89974937	-1.3	0	1.4	39.1	1016.8	290
111151	2018-01-01	0.009625	0.7208333333	0.009833333333	0.03725	48.25	22.66666667	-1.3	0	1.4	39.1	1016.8	290
111152	2018-01-01	0.006375	0.5791666667	0.01279166667	0.032125	46	20.58333333	-1.3	0	1.4	39.1	1016.8	290
111154	2018-01-01	0.008208333333	0.6066956522	0.009375	0.041375	43.33333333	19.89974937	-1.3	0	1.4	39.1	1016.8	290
111161	2018-01-01	0.006958333333	0.7083333333	0.009625	0.04070833333	43.04166667	19	-1.3	0	1.4	39.1	1016.8	290
111162	2018-01-01	0.006416666667	0.7791666667	0.00675	0.050375	56.16666667	19.89974937	-1.3	0	1.4	39.1	1016.8	290
111171	2018-01-01	0.005583333333	0.625	0.01554166667	0.03016666667	44.75	21.82608696	-1.3	0	1.4	39.1	1016.8	290
111181	2018-01-01	0.003208333333	0.65	0.013875	0.01691666667	40.875	21.16666667	-1.3	0	1.4	39.1	1016.8	290
111191	2018-01-01	0.003666667	0.495833333	0.02083333333	0.02870833333	36.20833333	19.89974937	-1.3	0	1.4	39.1	1016.8	290
111201	2018-01-01	0.005583333333	0.6375	0.01116666667	0.03445833333	39.20833333	20.95833333	-1.3	0	1.4	39.1	1016.8	290
111202	2018-01-01	0.00775	0.8633333333	0.005375	0.04541666667	58.43478261	19.89974937	-1.3	0	1.4	39.1	1016.8	290
111212	2018-01-01	0.004375	0.6958333333	0.01120833333	0.03775	36.20833333	13.41666667	-1.3	0	1.4	39.1	1016.8	290
111213	2018-01-01	0.006291666667	0.6416666667	0.006875	0.04545833333	50.58333333	19.89974937	-1.3	0	1.4	39.1	1016.8	290
111221	2018-01-01	0.007375	0.6333333333	0.0075	0.02520833333	47.875	28.25	-1.3	0	1.4	39.1	1016.8	290
111231	2018-01-01	0.007333333333	0.7083333333	0.01016666667	0.030625	47.83333333	24.04166667	-1.3	0	1.4	39.1	1016.8	290
111232	2018-01-01	0.006166666667	0.6291666667	0.006333333333	0.05041666667	51.45833333	19.89974937	-1.3	0	1.4	39.1	1016.8	290
111241	2018-01-01	0.0045	0.5541666667	0.01183333333	0.03625	40.16666667	16.75	-1.3	0	1.4	39.1	1016.8	290
111242	2018-01-01	0.005333333333	0.7375	0.004916666667	0.05166666667	48.91666667	19.89974937	-1.3	0	1.4	39.1	1016.8	290
111251	2018-01-01	0.007625	0.4791666667	0.007916666667	0.03716666667	38.33333333	23.75	-1.3	0	1.4	39.1	1016.8	290
111261	2018-01-01	0.006166666667	0.6333333333	0.009208333333	0.034375	34.41666667	22.33333333	-1.3	0	1.4	39.1	1016.8	290
111262	2018-01-01	0.005375	0.5791666667	0.01016666667	0.03129166667	49.42857143	24.04166667	-1.3	0	1.4	39.1	1016.8	290

- 미세먼지 & 기상 데이터 병합

파이썬 활용 전처리

```
airdata.head()
```

	Station code	Date	SO2	CO	O3	NO2	PM10	PM25
0	111121	2018-01-01	0.004833	0.554167	0.014583	0.029292	32.666667	17.791667
1	111121	2018-01-02	0.004625	0.783333	0.012208	0.036292	32.125000	19.666667
2	111121	2018-01-03	0.004583	0.491667	0.017833	0.020625	29.583333	17.583333
3	111121	2018-01-04	0.004625	0.662500	0.008792	0.036042	38.708333	23.833333
4	111121	2018-01-05	0.005083	0.766667	0.006292	0.044083	47.875000	32.708333

```
airdata.isnull().sum()
```

```
Station code        0
Date                0
SO2               917
CO                961
O3                815
NO2               880
PM10             1267
PM25             2480
dtype: int64
```

```
median1=airdata.SO2.median()
```

```
airdata.SO2.fillna(median1,inplace=True)
```

```
median2=airdata.CO.median()
median3=airdata.O3.median()
median4=airdata.NO2.median()
median5=airdata.PM10.median()
median6=airdata.PM25.median()
```

```
airdata.CO.fillna(median2,inplace=True)
airdata.O3.fillna(median3,inplace=True)
airdata.NO2.fillna(median4,inplace=True)
airdata.PM10.fillna(median5,inplace=True)
airdata.PM25.fillna(median6,inplace=True)
```

- 결측치 제거
- 데이터 스케일링

예측모델 - 선형 회귀

선형 회귀 모델 생성

PM10 : 정확도 = 0.168 → 매우 떨어짐

```
from sklearn.model_selection import train_test_split
# 목표 변수 : 'SO2', 'CO', 'O3', 'NO2', 'PM10', 'PM25'
x = df_dust[['Temp', 'Prec', 'WS', 'Humi', 'Pres', 'cbwd']]
y = df_dust[['PM10']]

x_train, x_test, y_train, y_test = train_test_split(x, y, train_size=0.8, test_size=0.2)
```

```
# 모델 생성
# from sklearn.linear_model import LinearRegression
mlr1 = LinearRegression()
mlr1.fit(x_train, y_train)
```

```
LinearRegression()
```

```
# 회귀계수 확인
print(mlr1.coef_)
```

```
[[-0.84
```

```
# 상수
print(m
```

```
[490.52
```

```
round(m
```

```
0.168
```

PM25 : 정확도 = 0.153 → 매우 떨어짐

```
z = df_dust[['PM25']]

x_train, x_test, z_train, z_test = train_test_split(x, z, train_size=0.8, test_size=0.2)
```

```
# 모델 생성
mlr2 = LinearRegression()
mlr2.fit(x_train, z_train)
```

```
LinearRegression()
```

```
# 회귀계수 확인
print(mlr2.coef_)
```

```
[[-0.50676889 -0.2239161  -6.40522696  0.13947856 -0.1758375   0.01852609]]
```

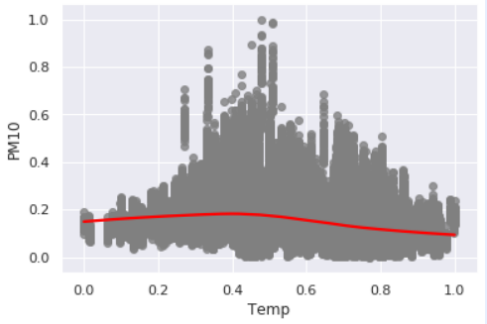
```
# 상수 확인
print(mlr2.intercept_)
```

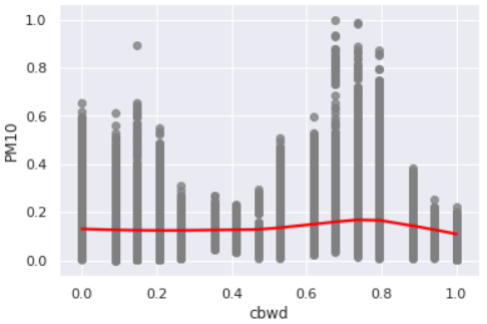
```
[208.19247258]
```

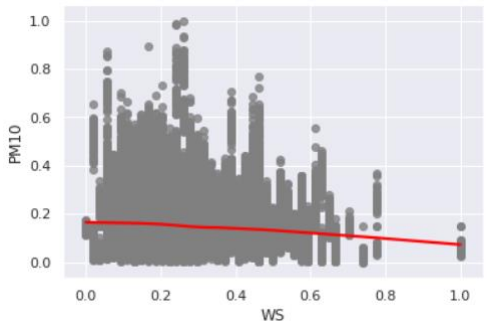
```
round(mlr2.score(x_train, z_train), 3)
```

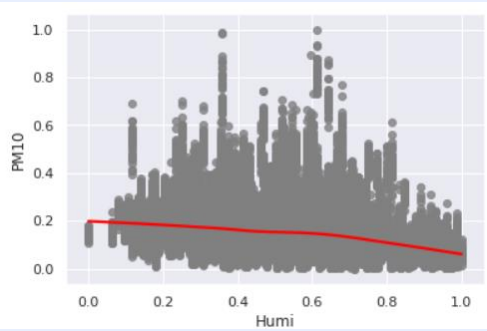
```
0.153
```

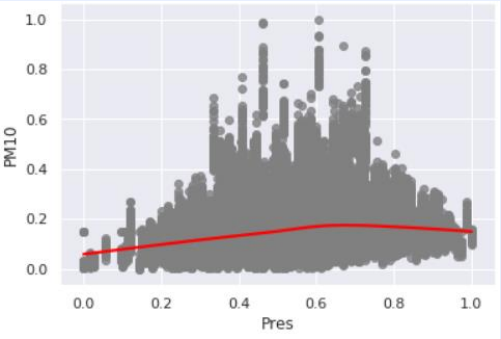
시각화

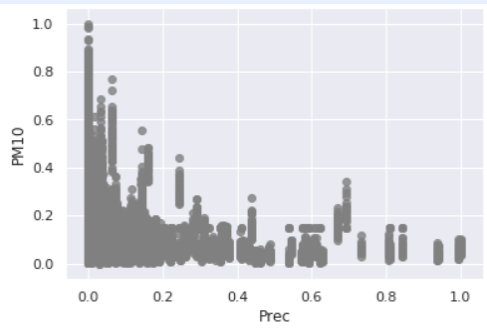






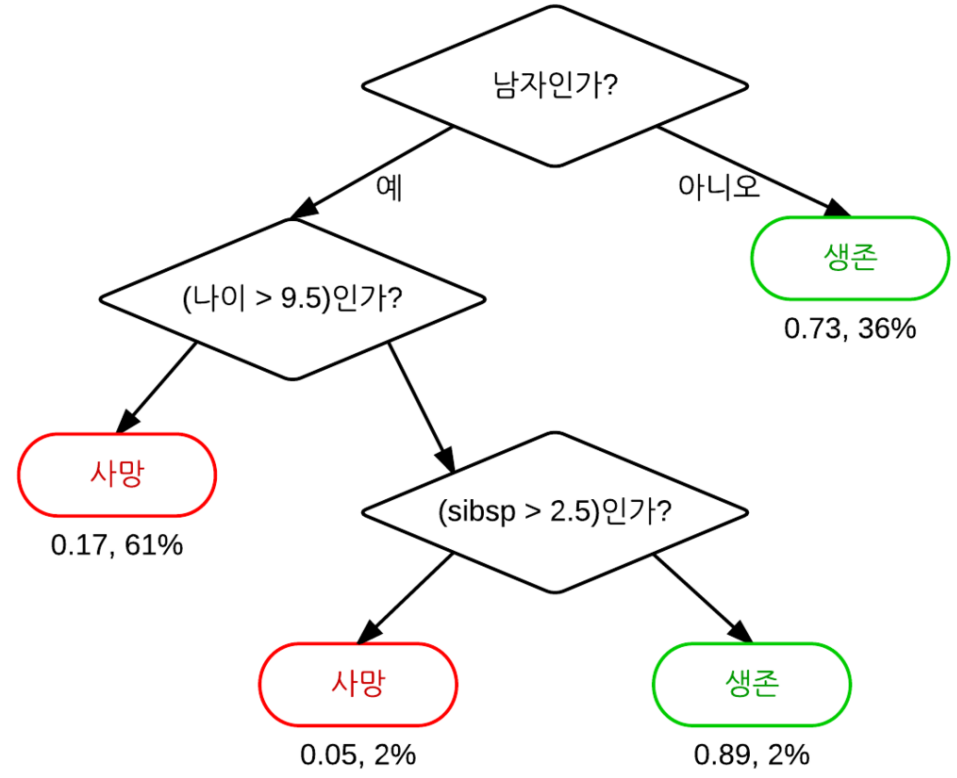






예측모델 - 결정 트리

결정 트리

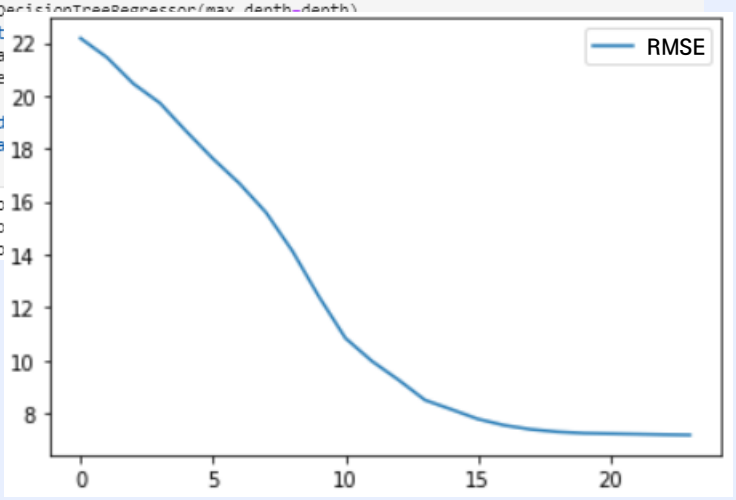


- 특정 기준에 따라 데이터 구분
- 자식 노드의 불순도가 낮도록 설정
- Leaf Node/Terminal Node의 복잡성 낮추는 것이 목표

모델 생성

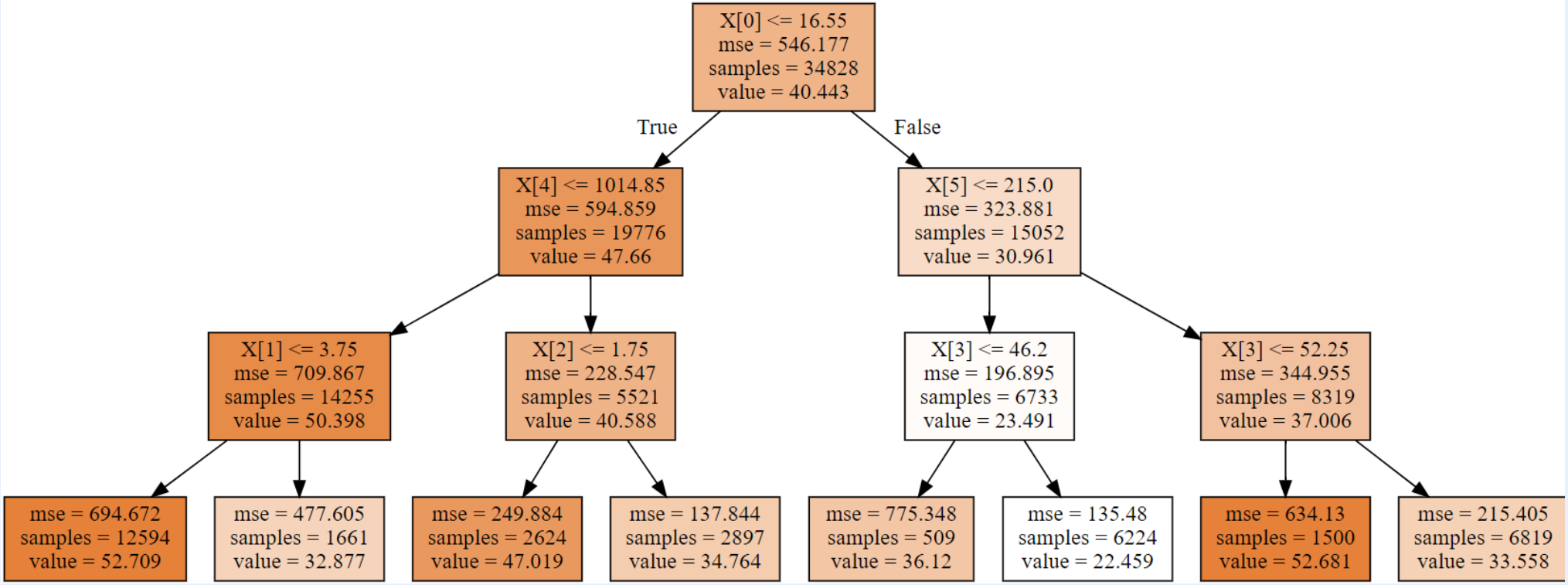
MimMaxScaler - PM10

```
[13]: x = data[['Temp', 'Prec', 'WS', 'Humi', 'Pres', 'cbwd']]
      trans=MinMaxScaler()
      X=trans.fit_transform(x)
      y = data['PM10']

      x_train, x_test, y_train, y_test = train_test_split(X, y, train_size=0.8, test_size=0.2)

[14]: estimator = DecisionTreeRegressor(max_depth=5)
      estimator.fit(x_train,y_train)

[14]: DecisionTreeRegressor(max_depth=5)

[15]: mse= mean_squared_error(y_test,y_pred)
      r2 = r2_score(y_test,y_pred)
      print('MSE : ',mse)
      print('R2 Score : ', r2)

      MSE :  1062.688248208727
      R2 Score :  -0.9681880238461189

[18]: scores=[]
      for depth in range(1,25):
          estimator = DecisionTreeRegressor(max_depth=depth)
          estimator.fit
          y_pred=estima
          r2 = r2_score
          print('depth
          scores.append
      scores=pd.DataFra
      scores.plot()

      depth =  1 R2 Sco
      depth =  2 R2 Sco
      depth =  3 R2 Sco
```



예측모델

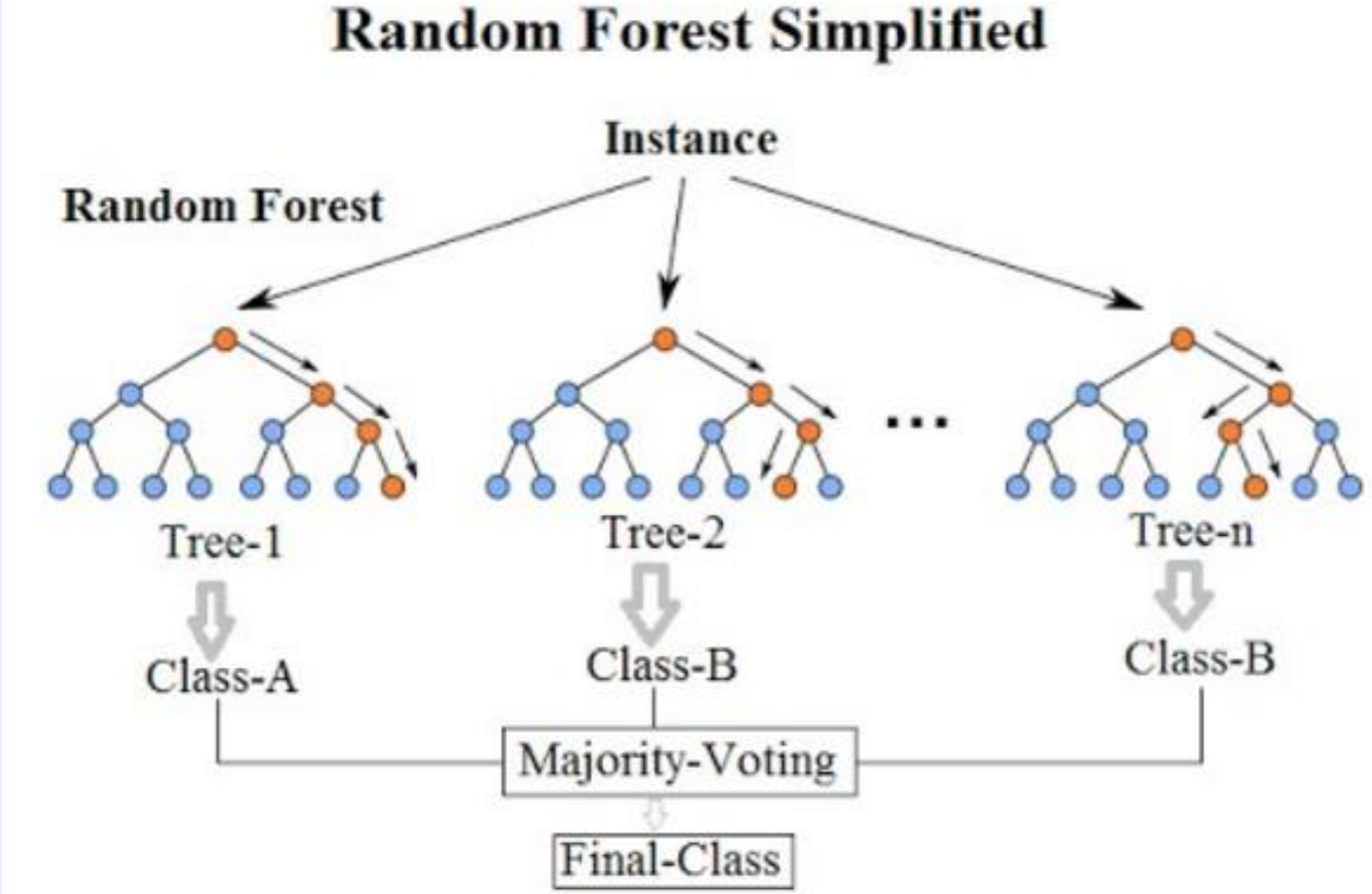
시각화



PM 10 / max depth = 4

예측모델 - Random Forest

Random Forest 랜덤 포레스트



예측모델

Random Forest 랜덤 포레스트

- 배깅을 적용한 결정트리의 앙상블
- 포레스트의 크기 T
  : 트리의 개수를 결정
- 최대 허용 깊이 D
  : 하나의 트리에서 루트 노드부터
  종단 노드까지 노드 수를 결정

Extra Trees 엑스트라 트리

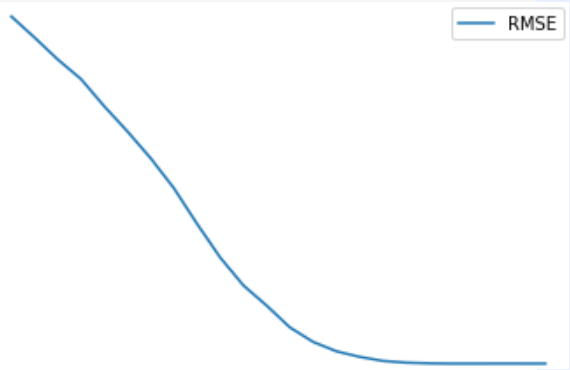
- Extremely Randomized Trees
- 비복원 추출
- 분할(Split)시 최적의
  항목(Feature)을 찾는 랜덤
  포레스트와 달리 랜덤지정
- 랜덤 포레스트 보다 빠른 속도

예측모델

랜덤 포레스트 모델

```
# minmaxscaler
# 0,1 사이로
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

scaler_col = ['Temp', 'Prec', 'WS', 'Humi', 'Pres', 'cbwd', 'PM10']
df_scaled = scaler.fit_transform(df[scaler_col])
df_scaled = pd.DataFrame(df_scaled)
df_scaled.columns = scaler_col

df_scaled.head()
```

	Temp	Prec	WS	Hun
0	0.278351	0.0	0.148148	0.26801
1	0.278351	0.0	0.148148	0.26801
2	0.278351	0.0	0.148148	0.26801
3	0.278351	0.0	0.148148	0.26801
4	0.278351	0.0	0.148148	0.26801




```
feature_cols = ['Temp', 'Prec', 'WS', 'Humi', 'Pres', 'cbwd']
label_cols = ['PM10']
x = df[feature_cols]
y = df[label_cols]
```

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2)
```

```
from sklearn.ensemble import RandomForestRegressor # 회귀트리(모델)

model = RandomForestRegressor(n_estimators=400, min_samples_split=3)
model.fit(X = x_train, y = y_train)
```

엑스트라 트리 모델

MinMaxScaler - PM 10

```
# 기온, 강수량, 풍속, 습도, 대기압, 풍향
x = data[['Temp', 'Prec', 'WS', 'Humi', 'Pres', 'cbwd']]
trans=MinMaxScaler()
X=trans.fit_transform(x)

y=data['PM10']

x_train, x_test, y_train, y_test = train_test_split(X, y, train_size=0.8, test_size=0.2)
```

```
etr=ExtraTreesRegressor()
etr.fit(x_train,y_train)
```

```
ExtraTre
```

```
round(et
```

```
0.91
```

```
y_pred=e
r2=r2_sc
print('R
```

```
R2: 0.91
```

MinMaxScaler - PM 2.5

```
x = data[['Temp', 'Prec', 'WS', 'Humi', 'Pres', 'cbwd']]
trans=MinMaxScaler()
X=trans.fit_transform(x)

y=data['PM25']

x_train, x_test, y_train, y_test = train_test_split(X, y, train_size=0.8, test_size=0.2)
```

```
etr=ExtraTreesRegressor()
etr.fit(x_train,y_train)
```

```
ExtraTreesRegressor()
```

```
round(etr.score(x_train, y_train),3)
```

```
0.882
```

```
round(etr.score(x_test,y_test),3)
```

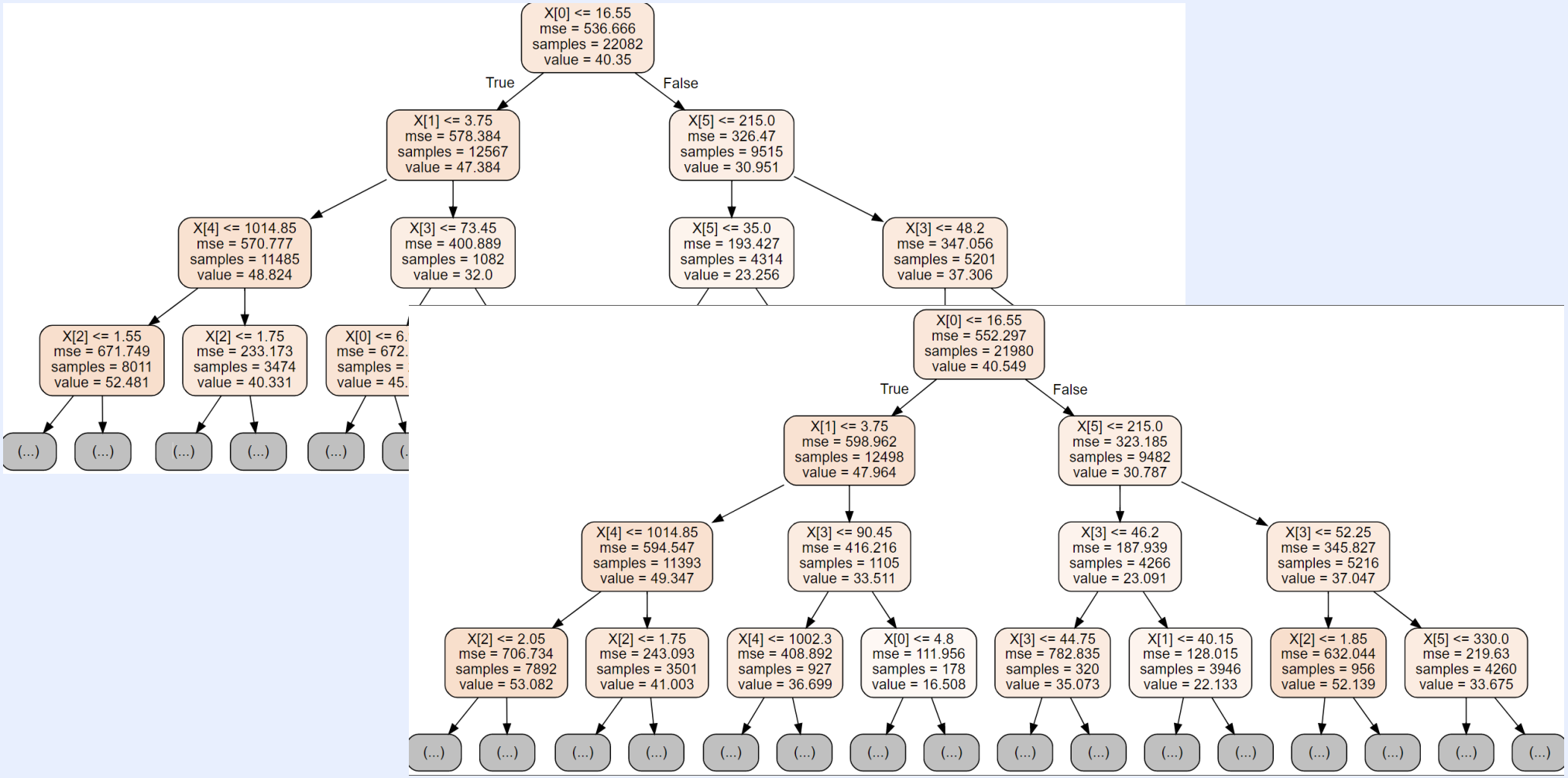
```
0.87
```

```
y_pred=etr.predict(x_test)
r2=r2_score(y_test,y_pred)
print('R2: %.3f'%r2)
```

```
R2: 0.870
```

예측모델

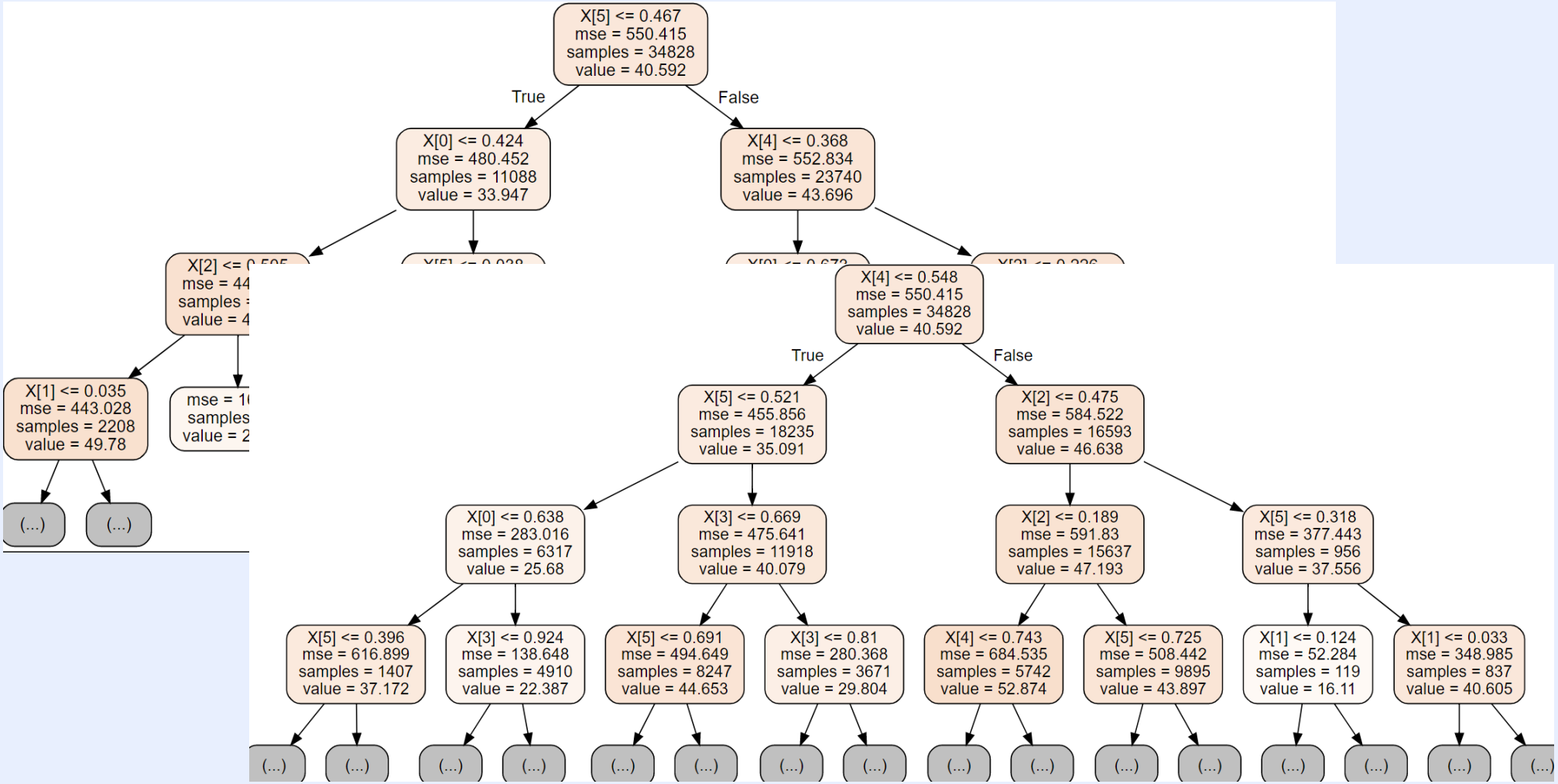
랜덤 포레스트 시각화



PM 10 / estimators_[1, 2]

예측모델

엑스트라 트리 시각화



PM 10 / estimators_[1, 2]

예측모델 - 시계열 모델

ARIMA

- 자기 회귀 모델(Auto Regressive), 이동 평균 모델(Moving Average), 차분(Integrated)을 합친 모델
- 적절한 매개 변수 조합을 찾는 것이 관건(pmdarima)

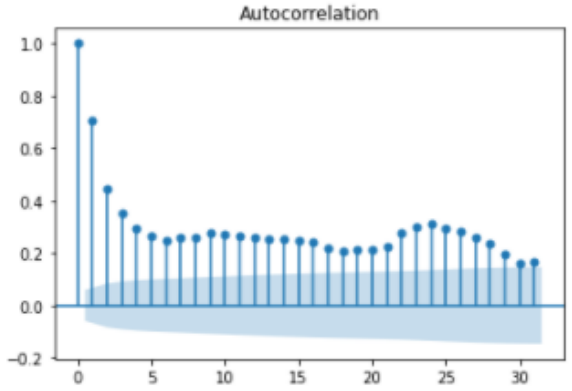
Prophet

- 계절성이 강하고 데이터가 많을수록 적합
- 이상치와 결측치에 대한 영향을 적게 받음

예측모델 - 시계열 모델

ARIMA 모델

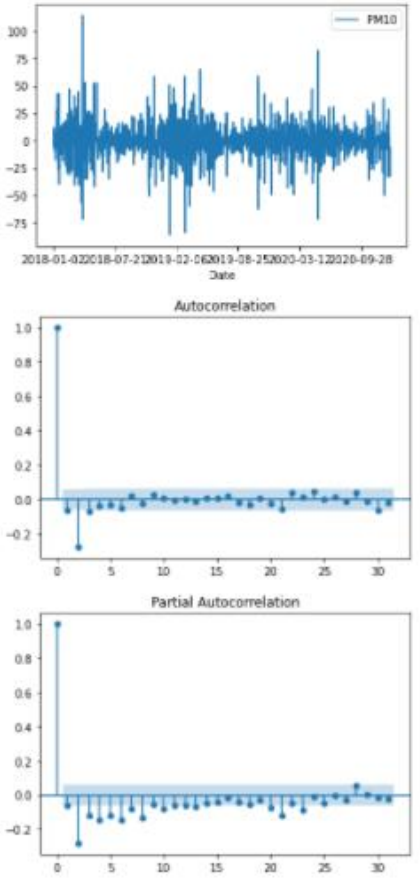
```
import matplotlib.pyplot as plt
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

plot_acf(data1)
plot_pacf(data1)
plt.show()
```



ACF가 양수이므로 AR 모델을 사용한다. arima(p,d,0)

1차 차분

```
diff_1=data1.diff(periods=1).iloc[1:]
diff_1.plot()
plot_acf(diff_1)
plot_pacf(diff_1)
plt.show()
```




```
p=range(0,3)
d=range(1,2)
q=range(0,3)

pdq=list(itertools.product(p,d,q))

aic=[]
```

```
for i in pdq:
    model = ARIMA(data1,order=i)
    model_fit=model.fit()
    aic.append(round(model_fit.aic,2))
```

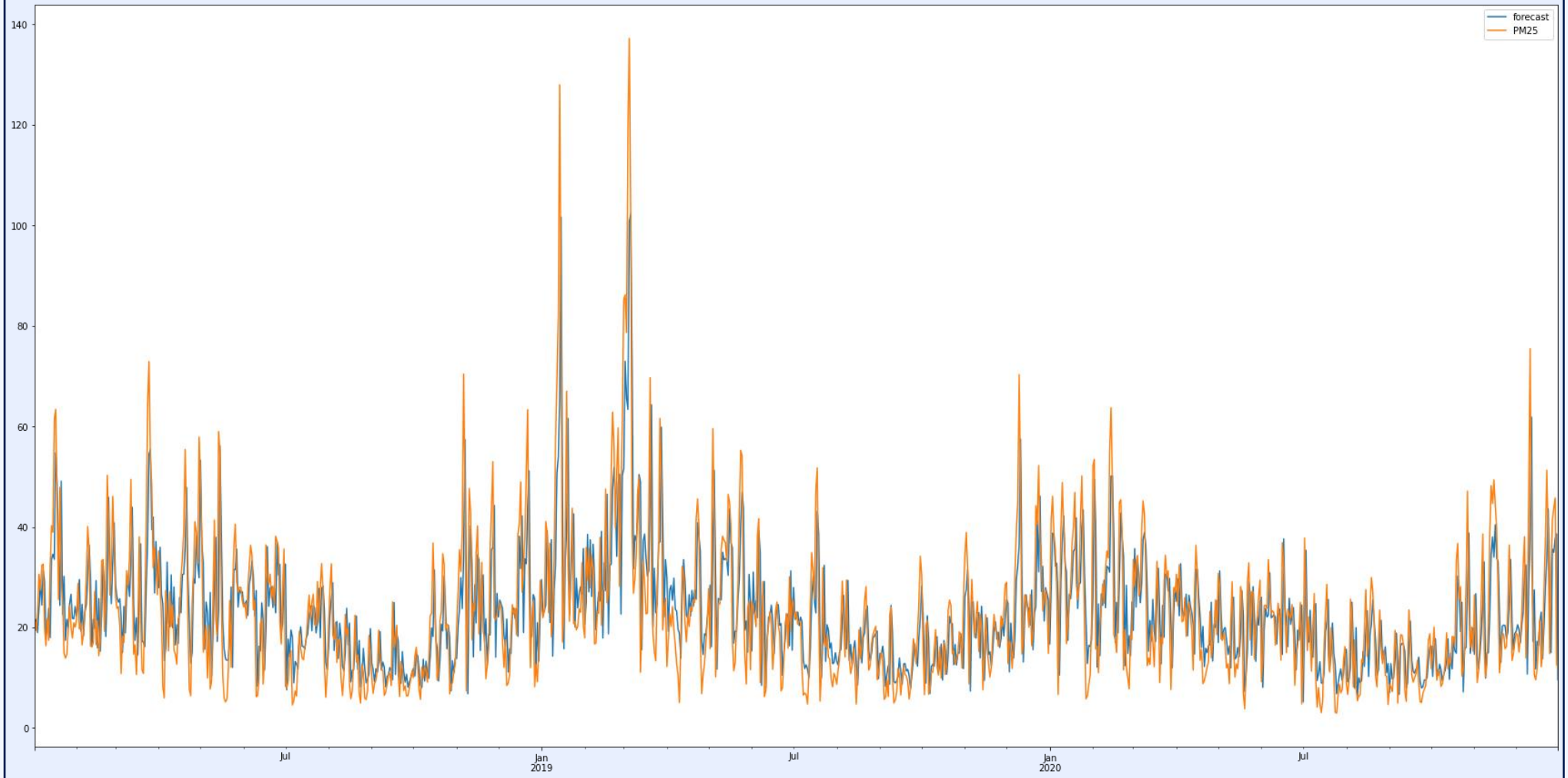
```
pd.DataFrame({'(p,d,q)': pdq, 'AIC': aic[:9]})
```

	(p,d,q)	AIC
0	(0, 1, 0)	9338.98
1	(0, 1, 1)	9330.16
2	(0, 1, 2)	9139.52
3	(1, 1, 0)	9336.49
4	(1, 1, 1)	9138.17
5	(1, 1, 2)	9098.98
6	(2, 1, 0)	9248.62
7	(2, 1, 1)	9105.88
8	(2, 1, 2)	9100.08

```
model = ARIMA(data1, order=(1,1,2))
model_fit = model.fit(trend='nc', full_output=True, disp=True)
print(model_fit.summary())
```

```
                             ARIMA Model Results
==============================================================================
Dep. Variable:                 D.PM10   No. Observations:                 1095
Model:                 ARIMA(1, 1, 2)   Log Likelihood               -4544.509
Method:                       css-mle   S.D. of innovations             15.343
Date:                Fri, 16 Jul 2021   AIC                           9097.018
Time:                        11:43:40   BIC                           9117.012
Sample:                    01-02-2018   HQIC                          9104.584
                         - 12-31-2020
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1.D.PM10     0.3657      0.050      7.256      0.000       0.267       0.465
ma.L1.D.PM10    -0.6097      0.051    -11.867      0.000      -0.710      -0.509
ma.L2.D.PM10    -0.3260      0.046     -7.130      0.000      -0.416      -0.236
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1            2.7343           +0.0000j            2.7343            0.0000
MA.1            1.0503           +0.0000j            1.0503            0.0000
MA.2           -2.9207           +0.0000j            2.9207            0.5000
```

예측모델 - ARIMA 시각화



예측모델 - 시계열 모델

Prophet 모델

PM10

```
PM10=data.groupby('Date').mean()
```

```
PM10.drop(['Station code','SO2','CO','O3','NO2','PM25','Temp','Prec','WS','Humi','Pres','
```

```
PM10.reset_index(inplace=True)
```

```
PM10.columns=['ds','y']
```

```
prophet = Prophet(seasonality_mode='multiplicative',
                  yearly_seasonality=True,
                  weekly_seasonality=True, daily_seasonality=True,
                  changepoint_prior_scale=0.5)
prophet.fit(PM10)
```

```
INFO:numexpr.utils:NumExpr defaulting to 2 threads.
<fbprophet.forecaster.Prophet at 0x7f4180957dd0>
```

```
future_data = prophet.make_future_dataframe(periods=60, freq='d')
forecast_data = prophet.predict(future_data)
forecast_data[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail(5)
```

	ds	yhat	yhat_lower	yhat_upper
1151	2021-02-25	60.091501	32.402469	88.673747
1152	2021-02-26	64.004163	35.387356	96.324936
1153	2021-02-27	65.218504	35.531412	95.552216
1154	2021-02-28	60.783802	32.584014	92.368172
1155	2021-03-01	65.651140	37.626611	97.208619

PM25

```
PM25=data.groupby('Date').mean()
```

```
PM25.drop(['Station code','SO2','CO','O3','NO2','PM10','Temp','Prec','WS','Humi','Pres
```

```
PM25.reset_index(inplace=True)
```

```
PM25.columns=['ds','y']
```

```
prophet = Prophet(seasonality_mode='multiplicative',
                  yearly_seasonality=True,
                  weekly_seasonality=True, daily_seasonality=True,
                  changepoint_prior_scale=0.5)
prophet.fit(PM25)
```

```
future_data = prophet.make_future_dataframe(periods=60, freq='d')
forecast_data = prophet.predict(future_data)
forecast_data[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail(5)
```

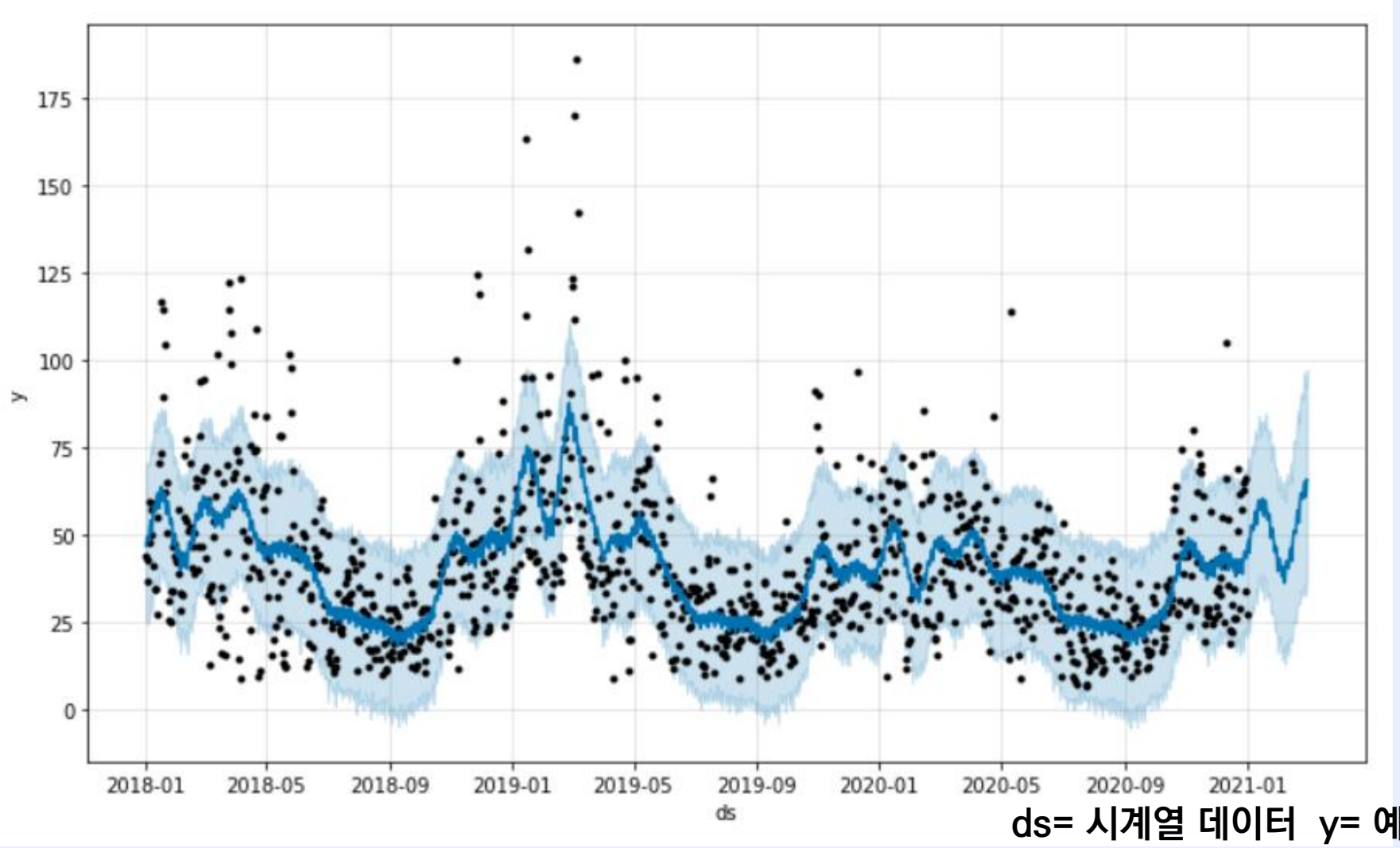
```
y_test=tset.PM25[:60]
x_test=forecast_data[['yhat']][-60:]
```

```
mae = mean_absolute_error(x_test,y_test)
mse = mean_squared_error(x_test,y_test)
print('MAE: %.3f' % mae)
print('MSE: %.3f' % mse)
```

```
fig1 = prophet.plot(forecast_data,xlabel='Date',ylabel='PM25')
```

예측모델 - 시계열 모델

Prophet 시각화



ds= 시계열 데이터 y= 예측 값

예측모델 - RMSE

이름	RMSE
Linear Regression	21.106
Decision Tree	6.986
Random Forest	7.037
Extra Tree	7.057
ARIMA	14.463
Prophet	23.934

결론 및 기대효과

- 국내 대기 분석을 통한 미세먼지 예측 가능
- 외부 요소 병합시 더 높은 예측률 기대

- 미세먼지 예보를 통해 배출저감 정책 및 조치의 정확성·효과성 향상 및 그에 따른 국민 생활 환경, 삶의 질 향상 기대 가능
- 일시적으로 악화된 기상 여건에 의해 언제든지 고농도 미세먼지 사례 발생 가능하므로 더욱 강화된 배출 감축 목표 설정 필요

개발 후기 및 느낀 점

박경빈

교육 때 따라서 하는 데이터 전처리와 다르게, 직접 데이터를 수집해서 전처리하고 모델링을 해보니 느낌이 매우 달랐다. 데이터를 보며 새로운 인사이트를 얻어내는 것에 흥미를 느낄 수 있었고, 데이터 사이언스라는 분야가 매력적인 분야라고 느꼈다. 전체적인 과정을 경험할 수 있는 의미 있는 시간이었다.

강아름

인사이트에 맞는 데이터 찾는 것 보다는 데이터를 기반으로 인사이트를 도출하는 편이 좋다는 것을 느꼈고, 여러가지 모델의 특성과 비슷한 모델의 경우에는 차이점까지 알 수 있어서 유익한 프로젝트였다.

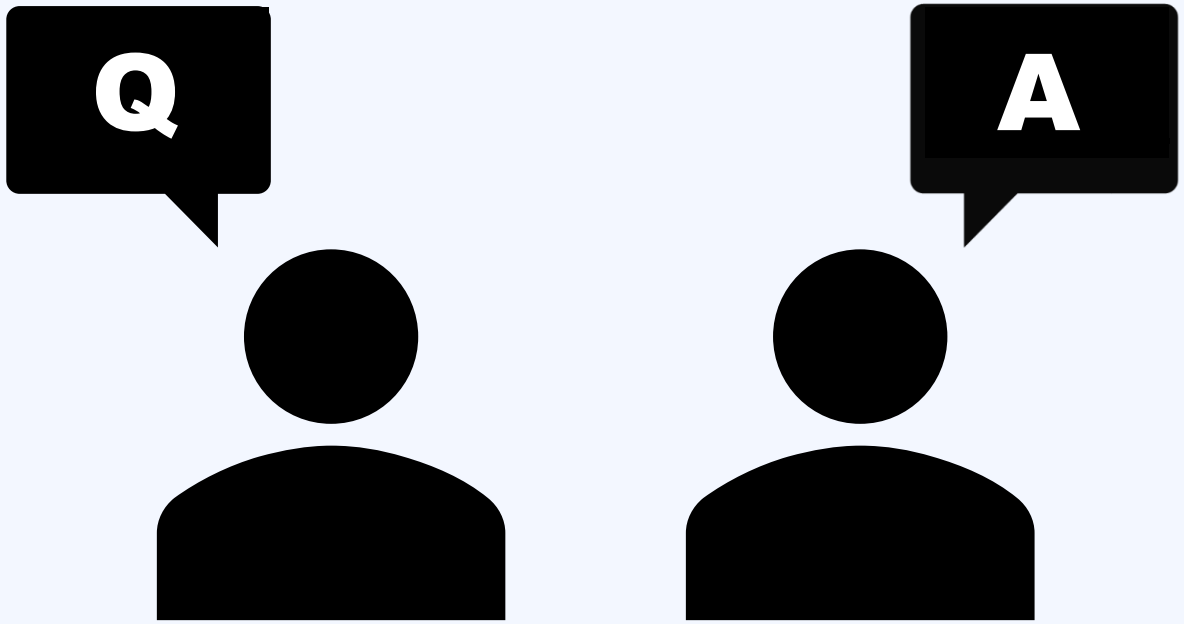
이우재

정제된 데이터셋만 보다가 결측치가 많은 데이터를 직접 정제하고 쓸 수 있도록 만드는 과정이 쉽지는 않았다. 그렇지만 이런 과정을 직접 해보고 함께 분석도 하면서 인사이트를 도출해보니 생각보다 흥미로운 시간이었다.

허재혁

교육 과정에서 배운 이론을 실제로 적용해서 내 손으로 직접 데이터를 전처리, 분석하는 과정에서 많이 배울 수 있었다. 데이터 사이언스 분야는 프로그래밍도 프로그래밍이지만, 모델들에 대한 이해와 모델을 통해 얻을 수 있는 결과를 어떻게 해석하는 지가 매우 중요하다고 느꼈다

Q&A



감사합니다