



Follow Data

-환경데이터 분석, AI 전문가-

환경정보 융합 빅데이터 플랫폼 데이터셋 분석하기



환경부



Follow Data

목 차

Followdata 알아보기

1. 시작하기-----	1
2. 내파일의 주요 기능-----	2
3. 시뮬레이터의 주요 기능-----	3
4. 시뮬레이터 실행하기-----	4
5. 분석생성-----	5
6. 분석 실행하기-----	6

Followdata 분석실습

1. 분석개요-----	9
2. 분석실습-----	10





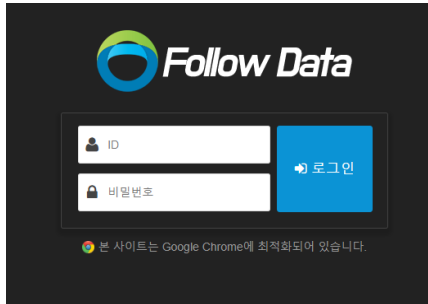
Follow Data

알아보기



1 시작하기

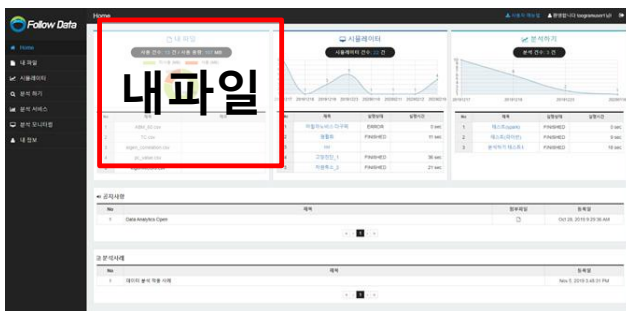
시스템 접속



파이썬 기반의 분석을 지원하는 FollowData 서비스입니다.

환경 데이터 포털에 등록된 계정으로 등록된 계정으로
Follow Data 서비스에 로그인 합니다.

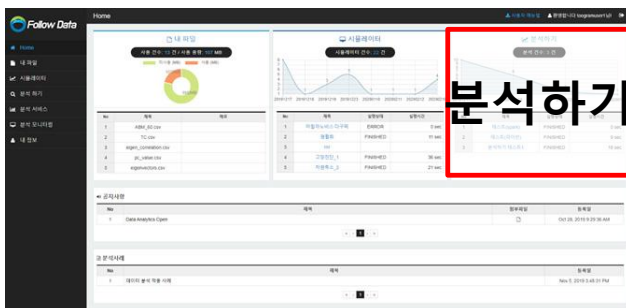
메인화면



메인화면의 내파일 영역에서는 사용자가 업로드 한 파일의 총 개수와 최신 파일목록을 최대 5개까지 확인할 수 있고, 사용 및 미사용 용량을 차트로 확인이 가능합니다.



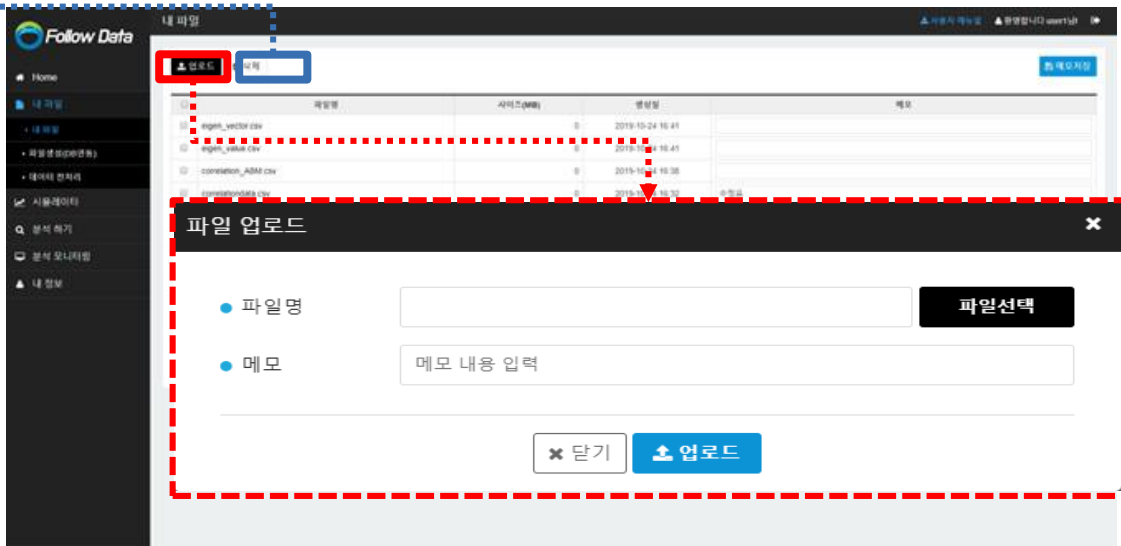
시뮬레이터 영역에서도 5개 가지의 목록확인 가능하고 날짜별 분석 건수를 그래프로 확인할 수 있습니다



새로운 데이터를 분석할 수 있습니다. 5개까지의 분석 목록 확인이 가능합니다.

2 내파일의 주요 기능

내파일 등록



The screenshot shows the '내파일' (My Files) interface. At the top, there are buttons for '업로드' (Upload) and '등록' (Register). Below is a table listing files:

이름	파일명	사이즈(MB)	생성일	비고
1	egon_vector.csv	0	2019-10-24 16:41	
2	egon_value.csv	0	2019-10-24 16:41	
3	connection_ABM.csv	0	2019-10-24 16:38	
4	connectiondata.csv	0	2019-10-24 16:37	수정됨

The '파일 업로드' (Upload File) modal is open, showing the following fields and buttons:

- 파일명** (File Name): Input field with a '파일선택' (Select File) button.
- 메모** (Memo): Input field with the placeholder '메모 내용 입력' (Enter memo content).
- Buttons**: '닫기' (Close), '업로드' (Upload).

내파일에서는 분석할 파일을 등록합니다.

- ① "업로드" 버튼을 클릭
 - ② 파일 업로드(확장자가 *.csv인 파일만 가능)
- ※ 사용자에게 주어진 업로드 가능한 총 용량을 초과시 업로드 불가

내파일 삭제

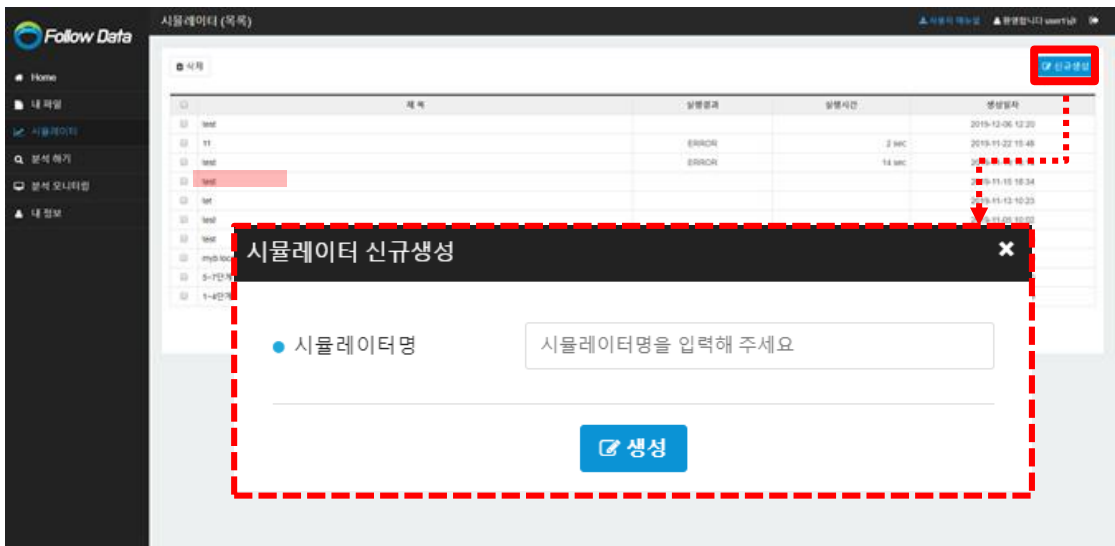
파일을 삭제 하겠습니까?

✕ 취소

✓ 삭제

3 시뮬레이터의 주요 기능

시뮬레이터 등록



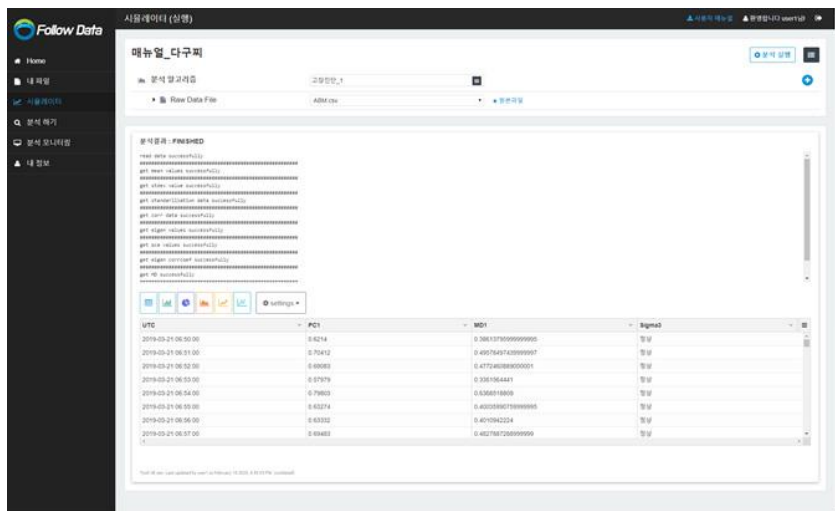
시뮬레이터 신규생성

- ① "신규생성" 버튼을 클릭
- ② 시뮬레이터 명 입력 후 생성

시뮬레이터 제목 선택시 시뮬레이터 실행화면으로 이동

시뮬레이터 실행

관리자가 튜닝한 알고리즘을 선택하여 사용자 데이터의 분석서비스를 진행하고 해당 분석결과를 다양한 형태의 그래프로 확인·관리 가능한 시뮬레이터 실행 화면



4 시뮬레이터 실행하기

시뮬레이터 실행

1 분석 알고리즘 검색

2 분석 알고리즘 선택

검색조건 : 검색조건을 입력하세요.

알고리즘명

- Eigen_correlation_PS_V1.0
- Eigen_correlation_PY_V1.0
- MD_PS_V1.0
- MD_PY_V1.0
- PC값계산_PS_V1.0
- PC값계산_PY_V1.0
- 고유벡터/고유값_PS_V1.0
- 고유벡터/고유값_PY_V1.0
- 고장전단_1
- 기본통계값_PY_V0.1
- 다중회귀_PS_V1.0
- 다중회귀_PY_V1.0
- 단순회귀_PS_V1.0
- 단순회귀_PY_V1.0
- 데이터표준화_PS_V1.0

분석명: 고장전단_1

분석방법: 주성분 분석을 이용한 고장전단 알고리즘
실제 데이터를 이용해서 정상/비정상 판단

입력 데이터: CSV파일(헤더, 인덱스, 데이터)

UTC	센서명	센서명	센서명	센서명
2019-03-21 06:50	45.3	45.3	45.3	45.3
2019-03-21 06:51	45.3	45.3	45.3	45.3
2019-03-21 06:52	45.2	45.2	45.2	45.2

출력 데이터: 주성분 점수(PC), 중심 간 거리(MD), 정상/비정상, 정상/비정상 갯수

UTC	주성분 점수(PC)	중심 간 거리(MD)	정상/비정상
2019-03-21 06:50	0.6214	0.386138	정상
2019-03-21 06:51	0.70412	0.495785	정상
2019-03-21 06:52	0.69083	0.477246	정상

적용 분야: 기계 부품 고장전단, 선박 부품 고장전단, 차량 부품 고장전단 등

입력 정보: Raw Data File

결과 정보:

닫기 분석 알고리즘 선택

① 분석알고리즘 검색

② 분석알고리즘 선택

③ 입력정보 선택

※ 내파일에 등록된 CSV 파일 선택

④ 분석실행

위의 과정을 통해 "분석 실행" 버튼을 클릭하여 시뮬레이터를 실행합니다.

사용자의 데이터 파일과 Follow Data 서비스의 알고리즘을 매핑하여 시뮬레이터가 실행됨을 확인할 수 있습니다.

⑤ 실행결과 확인

분석결과 : RAW DATA

UTC	PC1	MD1	logname
2019-03-21 06:50:00	0.6214	0.386138	정상
2019-03-21 06:51:00	0.70412	0.495785	정상
2019-03-21 06:52:00	0.69083	0.477246	정상

5 분석 생성

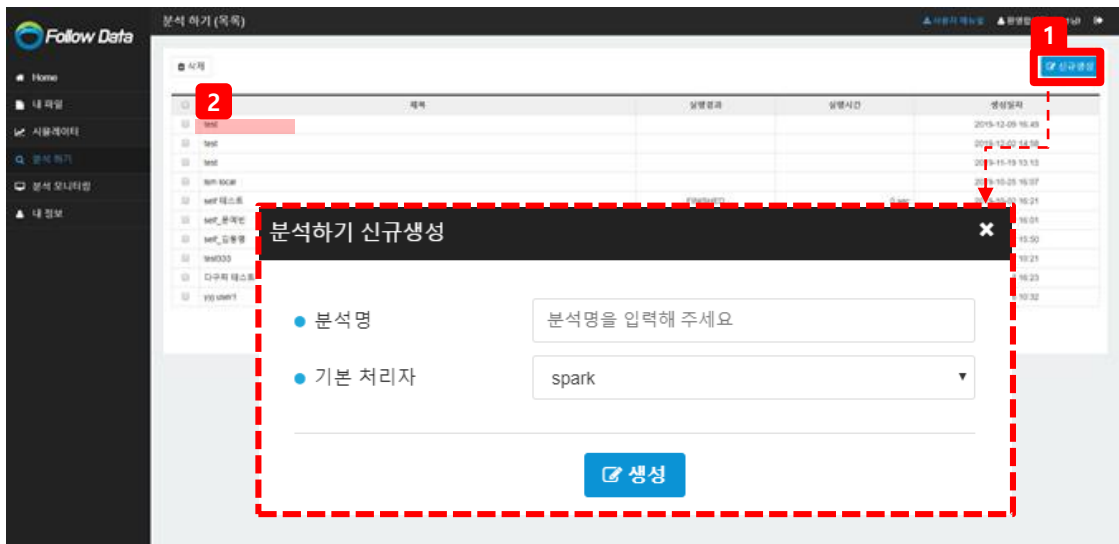
※ 먼저 알아두기 ※

분석 하기 메뉴에서 생성한 데이터분석 서비스 목록 관리

분석 하기 메뉴에서는 **사용자가 직접 작성한 SQL 쿼리 및 코드로 데이터 분석을 진행**하고, 분석결과를 다양한 형태의 **그래프를 통해 시각적으로 확인**할 수 있습니다.

분석 서비스를 진행하기 위해서는 분석에 필요한 SQL 쿼리 및 코드를 작성하고 실행한 후, 해당 결과물을 확인 및 관리할 수 있는 분석 실행 화면을 생성해야 합니다. 또한, 하나의 분석 실행 화면에서 여러 개의 분석 작업을 사용자가 직접 추가, 수정, 삭제하여 분석 실행 화면을 관리함으로써 분석코드 작성 / 실행 / 시각화를 손쉽게 할 수 있습니다.

분석 생성

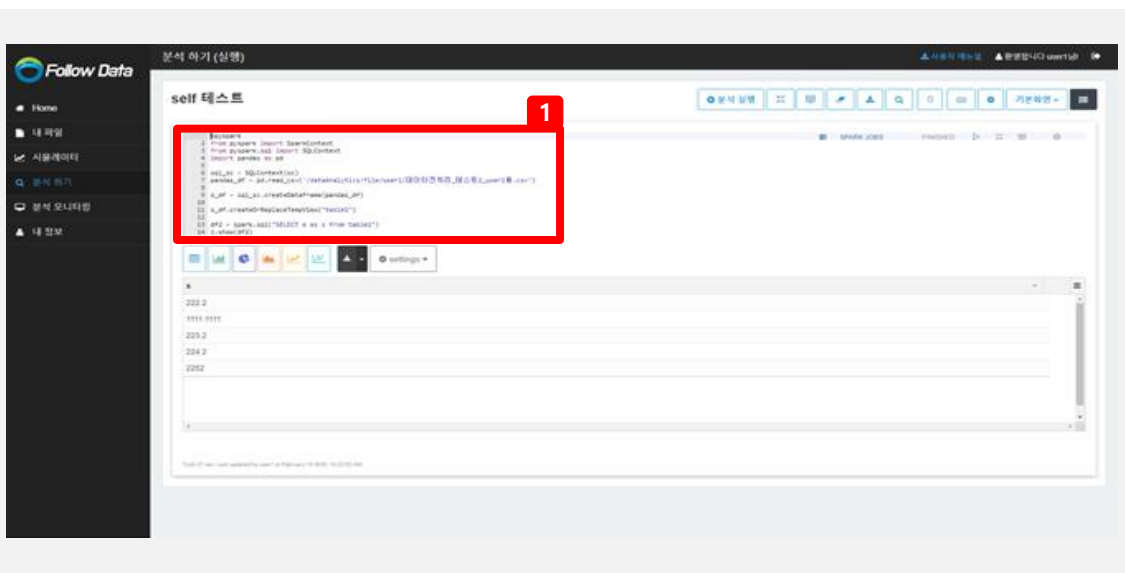


- ① 분석 목록 생성(분석 신규생성 버튼 클릭)
- ② 제목선택하여 분석화면으로 넘어가기

"기본 처리자" 값은 **spark** 또는 **python**을 선택가능(상세화면에서 변경 가능)
하며 **분석 실행 화면**의 첫 **분석 처리자** 값으로 자동설정

6 분석 실행하기

분석 실행



① SQL 코드 작성

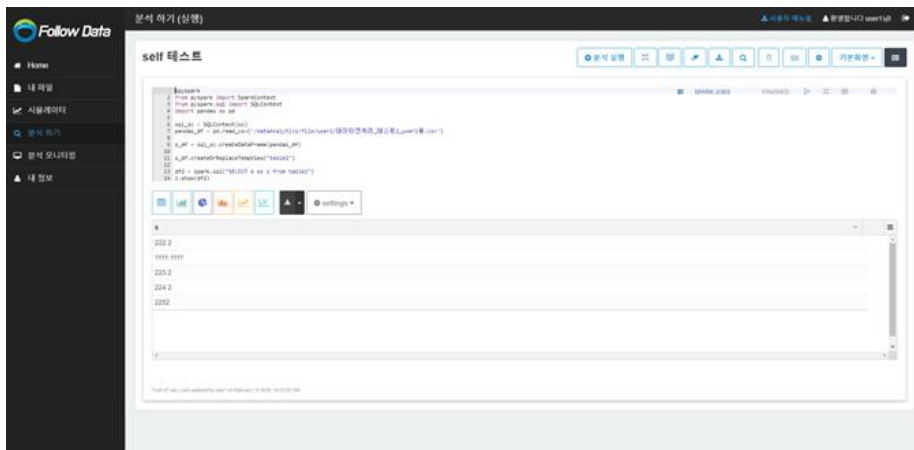
사용자가 직접 작성한 SQL 쿼리 및 코드로 데이터 분석을 진행

② 분석 실행(실행버튼 클릭)

③ 실행상태 확인

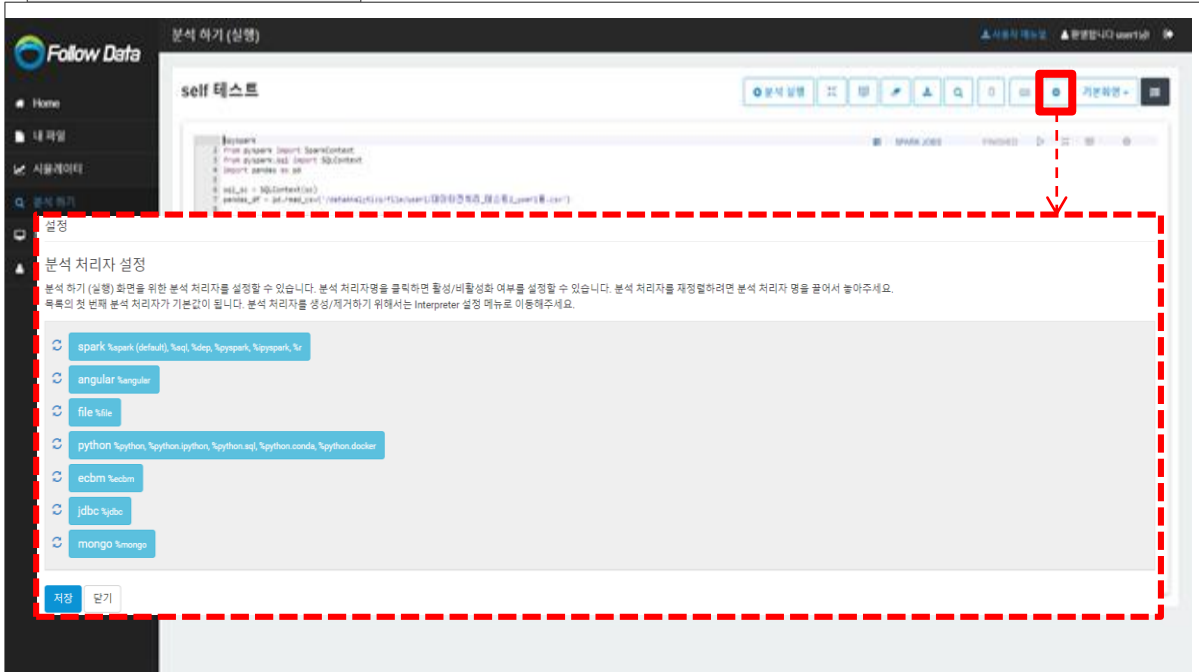
해당 **분석 작업**의 실행 상태(실행 대기, 완료, 일시정지, 에러, 보류, 실행 중)를 확인할 수 있습니다. 가능, 실행중인 경우 %로 진행률을 확인

RUNNING 41% 

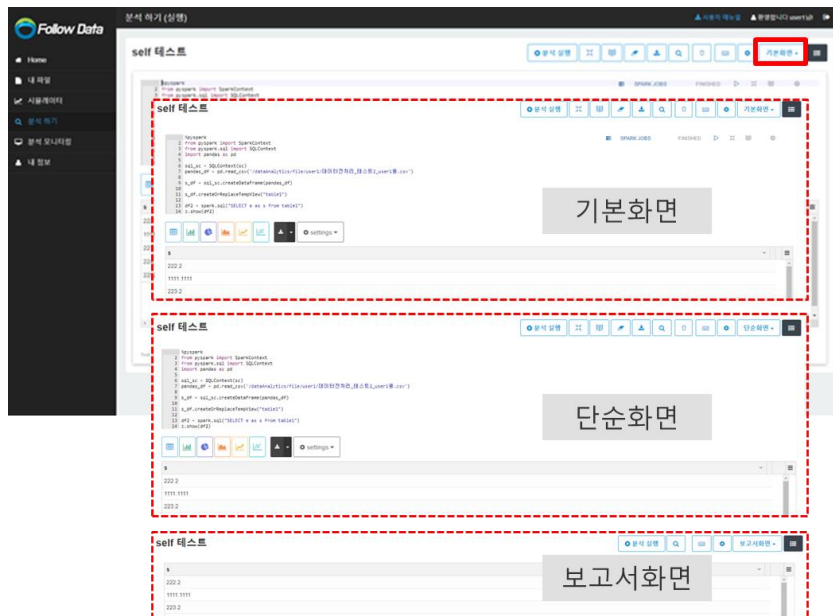


④ 실행결과 확인: 그래프 아이콘 활용 → 시각적 확인 가능

분석환경 설정



“톱니바퀴 아이콘”을 클릭하여 해당 분석 작업의 분석 처리자 값을 설정할 수 있습니다.
 분석 처리자 값의 사용여부를 조정할 수 있습니다.
 “새로고침 아이콘”을 클릭하여 해당 분석처리자의 상태를 초기화할 수 있습니다.



그밖의 화면 설정



Follow Data

분석사례(실습)



1 분석 개요

- 분석목표 : 특정지역의 다음날 미세먼지 예측
- 분석 데이터 : 특정 지역의 3개월간 미세먼지 측정값
- 분석모델 : ARIMA 모델을 이용한 단기 예측

분석환경

데이터 수집
(엑셀처리)

전처리
(수학 함수)

통계분석
(예측)

통계분석
(예측)

환경 데이터 포털

Pandas



엑셀파일
처리 패키지



NumPy

다차원 배열 등
수학 관련
작업 패키지



statsmodel

회귀분석과
시계열분석 등
전통적인
통계분석 패키지

matplotlib

통계분석 관련
다양한 시각화



Follow Data



python

분석데이터

예측분석

검증데이터

100,000 건 이상의 데이터는 'File' 탭에서 다운로드 가능합니다.

Sheet Chart File Link

번호	파일명	크기(kb)	등록일	다운로드(회)
18	[대기오염실시간공개시스템], 전국대기오염측정정보(지점목록), 2020_1분기.zip	15,348	2020-07-06	0
17	[대기오염실시간공개시스템], 전국대기오염측정정보(지점목록), 2019_4분기.zip	22,421	2020-07-06	0
16	[대기오염실시간공개시스템], 전국대기오염측정정보(지점목록), 2018_4분기.p	31,119	2019-10-16	5

- 시점 : 2018년 4분기 데이터
- 지역:경남양산 북부동(PM10미세먼지)
- 데이터명:전국대기오염측정정보

Sheet Chart File Link

번호	파일명	크기(kb)	등록일	다운로드(회)
3	[대기오염실시간공개시스템], 전국대기오염측정정보, 2019_3분기(7월~9월).zip	23,816	2019-10-16	5
2	[대기오염실시간공개시스템], 전국대기오염측정정보, 2019_2분기(4월~6월).zip	23,782	2019-10-16	4
1	[대기오염실시간공개시스템], 전국대기오염측정정보, 2019_1분기(1월~3월).zip	22,966	2019-10-16	10

- 시점 : 2019년 1분기 데이터
- 지역:경남양산 북부동(PM10미세먼지)
- 데이터명:전국대기오염측정정보

2 분석 실습

데이터
업로드

내파일 접속하여 데이터 등록하기

파일 업로드

파일명

파일선택

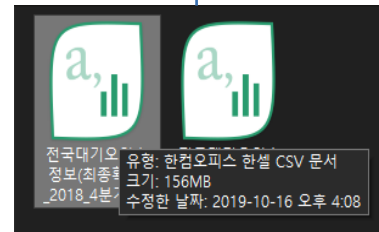
메모

메모 내용 입력

닫기

업로드

※환경데이터 포털에서 저장한
2개의 데이터파일을 등록합니다.



데이터
분석하기
등록

분석 신규 등록

분석하기 메뉴 접속

분석하기 신규생성

분석명

경남 양산시 북부동 대기질 예측

기본 처리자

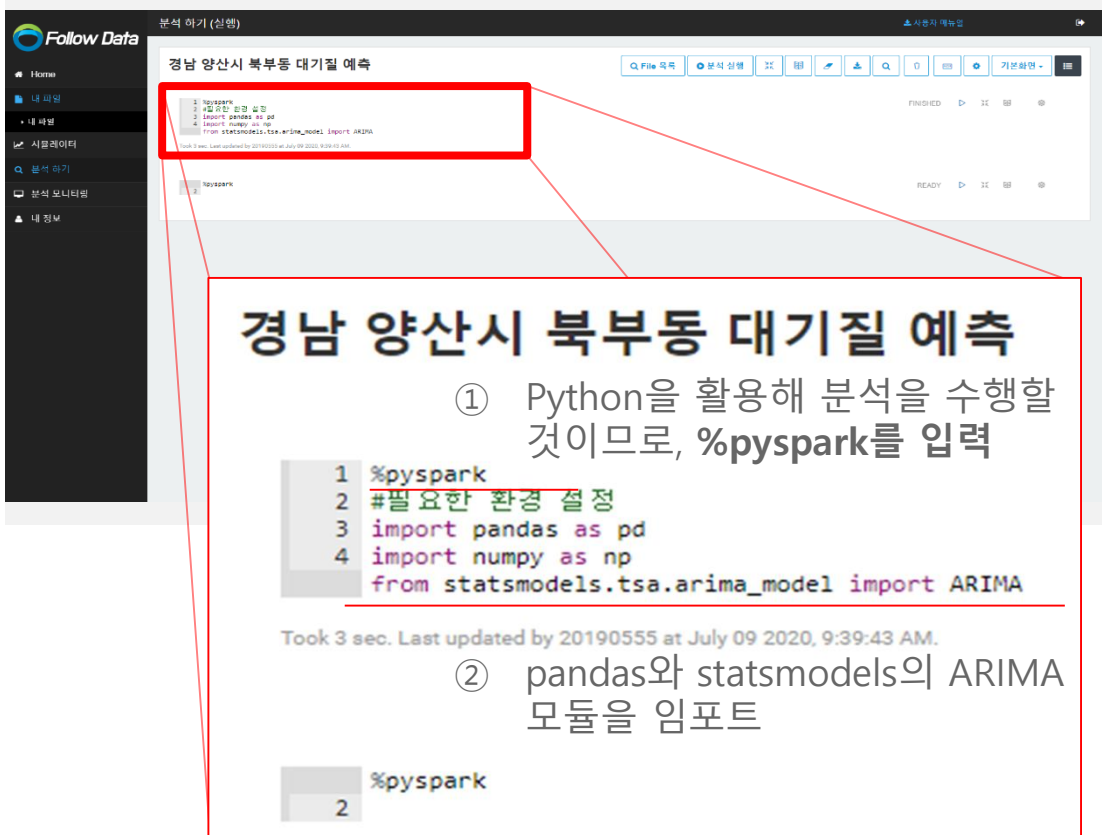
spark

생성

※분석작업을 신규 생성합니다.

기본환경 설정하기

기본환경
설정



경남 양산시 북부동 대기질 예측

① Python을 활용해 분석을 수행할 것이므로, %pyspark를 입력

```
1 %pyspark
2 #필요한 환경 설정
3 import pandas as pd
4 import numpy as np
5 from statsmodels.tsa.arima_model import ARIMA
```

Took 3 sec. Last updated by 20190555 at July 09 2020, 9:39:43 AM.

② pandas와 statsmodels의 ARIMA 모듈을 임포트

```
%pyspark
2
```

<<코드>>

```
#pandas와 numpy를 각각 pd, np라는 이름으로 활용할 것입니다.
import pandas as pd
import numpy as np
#matplotlib 패키지에서 pyplot 모듈만 가져와 plt라는 이름으로 사용
from matplotlib import pyplot as plt
#statsmodels 패키지에서 ARIMA 모듈만 등록
from statsmodels.tsa.arima_model import ARIMA
```

Shift + Enter 키를 누르면 코드가 적용되며, 밑에 새로운 코드를 입력할 수 있는 창이 나타납니다. 새 창에 계속 코드를 적습니다.

데이터 확인하기

데이터를 분석에 적합한 형태로 전처리

- 데이터를 측정소, 측정대상, 측정시간에 따라 처리
- 시계열 예측에 적합한 형태로 보정

```
%pyspart
air_pollution_raw = pd.read_csv('dataAnalytics/file/20190555/전국
대기오염측정보(최종확정)_2018_4분기.csv')
Air_pollution_raw.info()
```

데이터 전처리

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 849878 entries, 0 to 849877
Data columns (total 13 columns):
지역명                849878 non-null object
측정망명              849878 non-null object
측정소코드            849878 non-null int64
측정소명              849878 non-null object
측정시점              849878 non-null int64
마황산가스농도(ppm)   788541 non-null float64
일산화탄소농도(ppm)  786349 non-null float64
오존농도(ppm)         793482 non-null float64
미산화질소농도(ppm)  793510 non-null float64
미세먼지10농도(µg/m³) 786583 non-null float64
미세먼지25농도(µg/m³) 735320 non-null float64
주소                  849878 non-null object
최종수정일시          849878 non-null object
dtypes: float64(6), int64(2), object(5)
memory usage: 84.34 MB
```

업로드한 csv를 읽은 후, 기본 정보를 확인합니다.

미세먼지10농도의 데이터형(float64)은 숫자로 계산 가능한 데이터형임을 확인합니다.

데이터 추출 하기

예측에 활용할 측정소(경남 양산시 북부동)를 선택합니다.
해당 측정소의 측정소코드(238361) 확인 후 데이터에서 측정
소코드가 일치하는 행들만 추출하여 저장

```
yangsan_air_pollution = air_pollution_raw[air_pollution_raw['측정소코드'] == 238361]
yangsan_air_pollution
```

	지역명	측정망명	...	주소	최종수정일시
114	경남	양산시	도시대기	경남 양산시 북부동 북안남 5번길 21	20190801 12:10:40
491	경남	양산시	도시대기	경남 양산시 북부동 북안남 5번길 21	20190801 12:10:40
868	경남	양산시	도시대기	경남 양산시 북부동 북안남 5번길 21	20190801 12:10:40
1245	경남	양산시	도시대기	경남 양산시 북부동 북안남 5번길 21	20190801 12:10:40
1622	경남	양산시	도시대기	경남 양산시 북부동 북안남 5번길 21	20190801 12:10:40
1999	경남	양산시	도시대기	경남 양산시 북부동 북안남 5번길 21	20190801 12:10:40
2376	경남	양산시	도시대기	경남 양산시 북부동 북안남 5번길 21	20190801 12:10:40
2753	경남	양산시	도시대기	경남 양산시 북부동 북안남 5번길 21	20190801 12:10:40
3130	경남	양산시	도시대기	경남 양산시 북부동 북안남 5번길 21	20190801 12:10:40
3507	경남	양산시	도시대기	경남 양산시 북부동 북안남 5번길 21	20190801 12:10:40
3884	경남	양산시	도시대기	경남 양산시 북부동 북안남 5번길 21	20190801 12:10:40
4261	경남	양산시	도시대기	경남 양산시 북부동 북안남 5번길 21	20190801 12:10:40
4638	경남	양산시	도시대기	경남 양산시 북부동 북안남 5번길 21	20190801 12:10:40
5015	경남	양산시	도시대기	경남 양산시 북부동 북안남 5번길 21	20190801 12:10:40
5392	경남	양산시	도시대기	경남 양산시 북부동 북안남 5번길 21	20190801 12:10:40
5769	경남	양산시	도시대기	경남 양산시 북부동 북안남 5번길 21	20190801 12:10:40
6146	경남	양산시	도시대기	경남 양산시 북부동 북안남 5번길 21	20190801 12:10:40

데이터
전처리

시계열 분석용 데이터 추출 하기

```
yangsan_pm10 = pd.DataFrame({'target': yangsan_air_pollution['미세먼지10농도( $\mu\text{g}/\text{m}^3$ )']}).reset_index(drop=True)
yangsan_pm10.head()
```

```
target
0      24.0
1      17.0
2      19.0
3      18.0
4      21.0
```


결측값 확인 및 처리

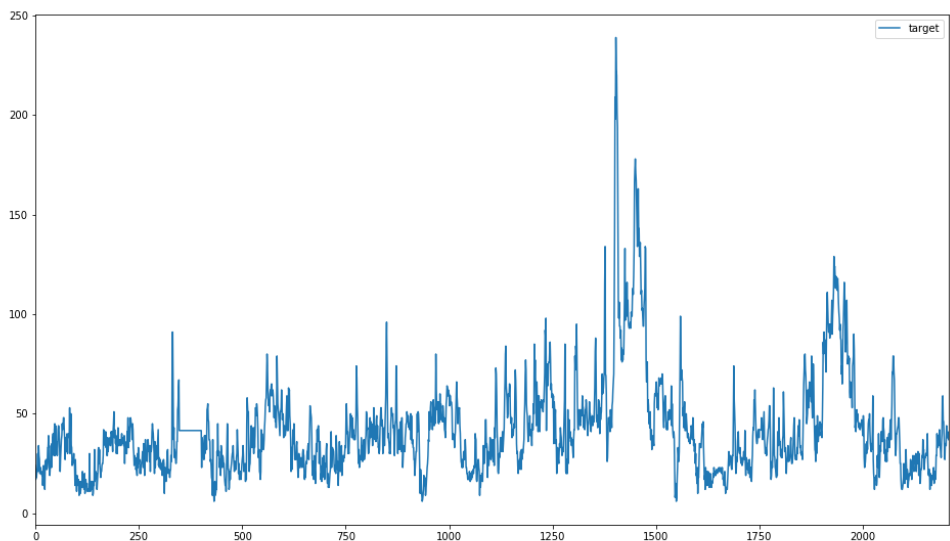
결측값은 시계열 예측에 치명적이므로 반드시 처리해 줘야 합니다. 이번 분석에서는 측정값의 평균으로 결측값을 대체하겠습니다. 이후 단기 예측을 위해 필요한 데이터를 선별합니다.

```
yangsan_pm10.fillna(yangsan_pm10['target'].mean(),  
inplace=True)  
yangsan_pm10[yangsan_pm10.isna()['target'] == True]
```

전처리 데이터 확인

전처리된 자료의 결측자료 잔존 여부등을 시각화를 통해 확인

```
#사이즈가 16x9인 그림판을 만듭니다.  
fig, ax = plt.subplots(figsize=(16,9))  
#그림 정보에 DataFrame의 plot 내장함수로 그린 그래프를 추가합니다.  
ax = yangsan_pm10.plot(ax=ax)  
#그림을 표시합니다.  
plt.show()
```



※ 위의 코드는 모두 같은 셀에 입력후 Shift+Enter로 실행

ARIMA 시계열 분석 수행

ARIMA는 시계열 데이터가 일정한 추세 위에 무작위성을 더한 분포라고 가정하고, 미래에 나타날 데이터의 추세와 분포를 파악하는 전통적 통계분석입니다.

차분(I)으로 시계열 데이터를 정상화한 후, 자기회귀(AR)정도와 이동평균(MA) 추세를 파악하여 미래의 데이터를 예측합니다.

```
model = ARIMA(yangsan, order=(1,1,1))
```

```
#모델 피팅(상수항 없이 피팅)
```

```
model_fitted = model.fit(trend='nc')
```

```
#피팅 결과 요약
```

```
model_fitted.summary()
```

시계열 분석

Dep. Variable:	D.target	No. Observations:	70			
Model:	1 <u>ARIMA(1, 1, 1)</u>	Log Likelihood	-221.427			
Method:	css-mle	S.D. of innovations	5.708			
Date:	Mon, 06 Jul 2020	AIC	<u>448.854</u> 2			
Time:	19:20:28	BIC	455.600			
Sample:	12-29-2018	HQIC	451.534			
	- 12-31-2018					
=====						
	coef	std err	z	2 P> z	[0.025	0.975]

ar.L1.D.target	0.7218	0.177	4.070	0.000	0.374	1.069
ma.L1.D.target	-0.9090	0.116	-7.827	0.000	-1.137	-0.681
Roots						

① ARIMA 차수

statsmodels는 fit 함수를 통해 초기값과 초기차수가 주어진 ARIMA 모델을 최적화

summary 함수를 통해 최적화된 모델의 AR(자기회귀), I(차분), MA(이동평균) 각각의 차수를 확인

② AIC 지표

모델의 성능을 간단히 평가할 수 있는 지표

③ T검정 값 확인

T검정은 모델의 통계적 유의성을 표현하는 지표
※0.05 이하여야 일반적으로 유효한 모델

ARIMA 모델 자동 최적화

statsmodels의 ARIMA 모델은 최적 차수를 자동 계산할 수 없으므로, 가능한 차수를 적용한 모든 모델들의 AIC값을 비교해 최적 차수를 도출합니다.

```
#임의의 높은 AIC값을 설정, 아래 숫자는 임의의 큰 값
optimal_aic = 2147483647
#AR의 차수는 0~5 중에 결정
for p in range(6):
    #I의 차수는 0~2중에 결정
    for d in range(3):
        #MA의 차수는 0~5중에 결정
        for q in range(6):
            try:
                #각 경우에 ARIMA 모델을 생성하여 최적화
                model = ARIMA(yangsan_pm10,
                              order=(p,d,q)).fit(trend='nc', disp=0)
                #AIC가 저장된 AIC보다 낮다면 최적 모델로 설정
                if optimal_aic > model.aic:
                    optimal_model = model
                    optimal_aic = model.aic
            except:
                pass
#최적 모델의 정보를 호출
optimal_model.summary()
```

※ 108개의 모델을 최적화하므로, 몇 분 정도의 시간이 소요

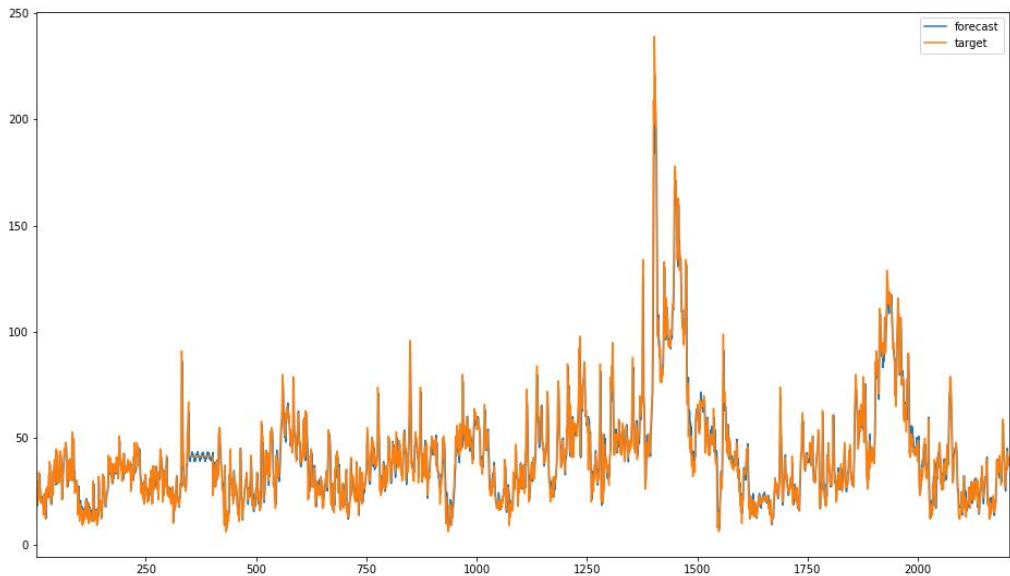
① ARIMA 차수

T검정 값

ARIMA Model		P> z
=====		-----
Dep. Variable:	D.target	
Model:	<u>ARIMA(5, 1, 5)</u>	0.000
Method:	css-mle	0.000
-	-	0.000

예측결과 시각화

```
fig, ax = plt.subplots(figsize=(16,9))
#ARIMAResults형의 plot_predict 함수는 반환값을 fig에 넣습니다.
fig = optimal_model.plot_predict(ax=ax)
plt.show()
```



예측결과 검증데이터 추가

```
# 2019년 1분기 데이터 읽기
air_pollution_19q1_raw = pd.read_csv('/dataAnalytics/file/20190555/전
국대기오염측정보_2019_1분기(1월~3월).csv')
# 양산 북부동 측정소의 첫 24개 데이터
yangsan_air_pollution_19q1_head =
air_pollution_19q1_raw[air_pollution_19q1_raw['측정소코드'] ==
238361].head(24)
# pm10 값을 measured라는 이름으로 추출 → 원본 데이터의 순번
마지막 값이 2207이므로, 2208~2231번을 붙여 정리
yangsan_pm10_19q1_head =
pd.DataFrame({'measured':pd.to_numeric(yangsan_air_pollution_19q1_h
ead['PM10 농도(μg/m³)'])}).set_index(np.array(range(2208,2232)))
# 데이터를 조회
yangsan_pm10_19q1_head
```

모델 검증 확인

수집한 실제 측정값을 모델의 추정값과 비교

```
fig, ax = plt.subplots(figsize=(16,9))  
#2100구간부터 2232 구간까지, 실제 측정값을 추가하여 그림  
#니다.  
fig = optimal_model.plot_predict(start=2100, end=2232,  
ax=yangsan_pm10_19q1_head.plot(ax=ax))  
plt.show()
```

모델
검증

