기후 변화에 대응하는 농작물 수확량 예측

목차

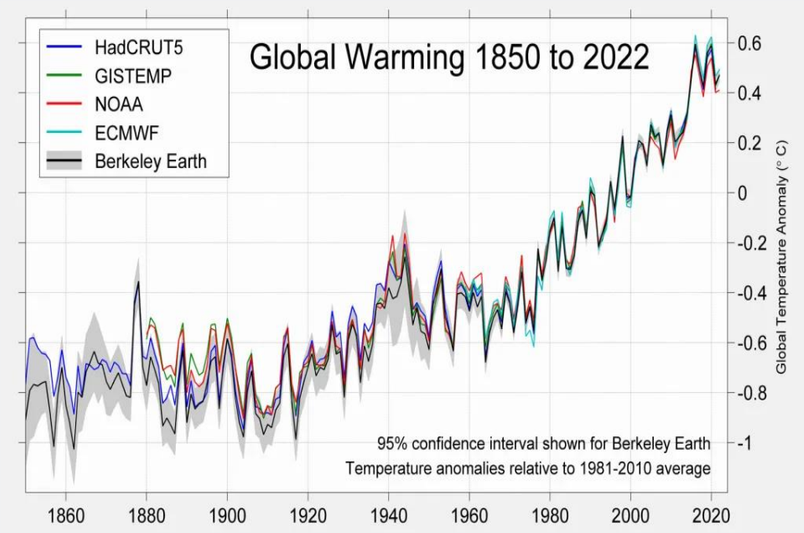
- 문제상황
- 선정 기준
- 데이터 선정
- 날씨 및 농작물 데이터 추출
- 웹 스크래핑 과정
- 데이터 정제
- 머신 제작

- 날씨 예측

- 수확량 예측

- 소감 및 담당업무

문제 상황



지구 기온 상승 그래프

재배 한계선 상승

농작물 재배에 유불리?

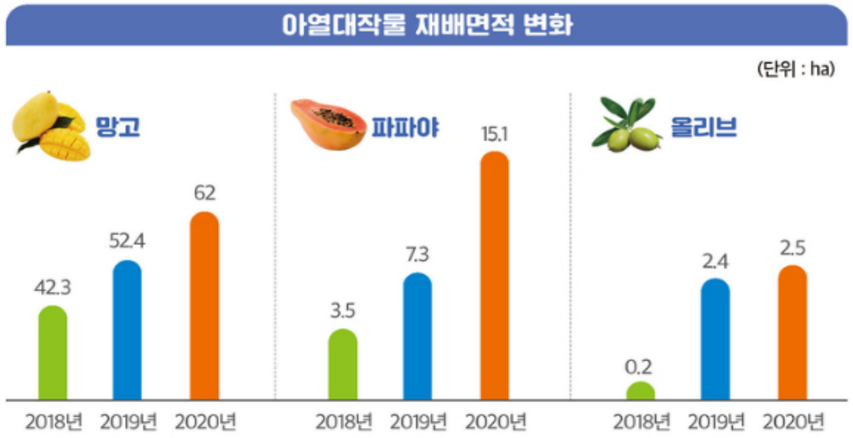
문제 상황

국내 아열대작물 재배농가, 1,376호 311.4ha… 망고, 파파야, 용과, 올리브 등 꾸준히 증가

농촌진흥청은 지난 2008년부터 아열대작물 연구를 시작해 현재까지 총 52종의 아열대작물을 도입했으며, 이 중 국내 환경에 맞는 22종을 선발했다. 특히 백향과, 망고, 용과 등 과수 8종, 여주, 롱빈, 아티초크 등 채소 7종 등 총 15개 작물의 재배기술을 개발해 보급하고 있다. 특히, 아열대 과수 중 망고는 열풍기, 히트펌프, 다겹보온커튼 등을 이용해 에너지를 46% 절감하는 기술 개발과 함께 나무 키를 낮게 키우는 방법으로 노동력을 36% 절감하고 상품률 또한 20% 높이는 기술을 개발해 농가에 보급하고 있다.

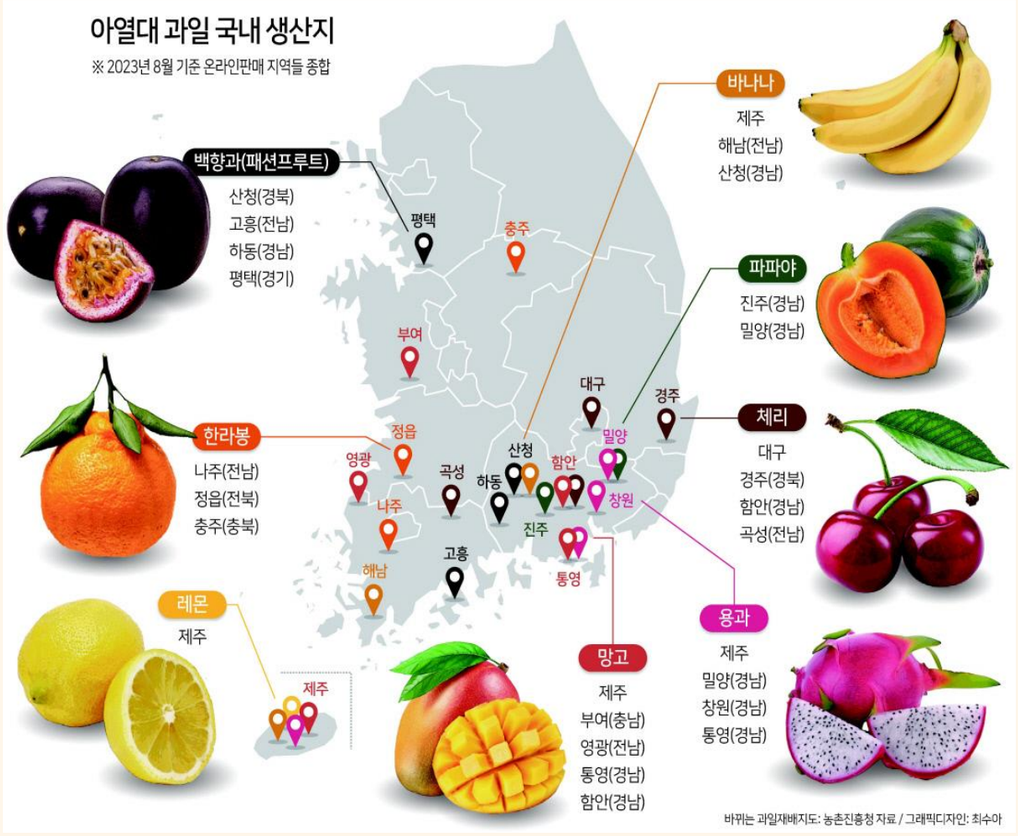
농촌진흥청이 지난 2020년 2월 전국 농촌진흥기관을 통해 국내 재배 중인 아열대작물 22종에 대한 재배현황을 조사한 결과에 따르면, 2020년 2월 기준으로 국내 전체 아열대작물 재배농가는 1,376호로 파악됐다. 재배면적은 311.4헥타르(ha), 생산량은 5,697.3톤이다. 이 중 아열대 채소 재배농가는 848호, 재배면적은 147.4ha, 생산량은 2,819.5톤이며, 아열대 과수 재배농가는 528호, 재배면적은 164ha, 생산량은 2,877.8톤이다.

조사 작물은 오크라, 삼채, 여주, 공심채, 강황, 사탕무, 얌빈, 게욱, 롱빈, 아티초크, 인디언시금치, 차요테 등 채소 12종과 망고, 백향과, 용과, 올리브, 파파야, 아떼모아, 구아바, 훼이조아, 바나나, 커피 등 과수 10종이다.



아열대 채소 가운데 재배면적이 넓은 작목은 ▲ 여주(59.9ha, 232농가) ▲강황(46.6ha, 367농가) ▲ 삼채(15.9ha, 71농가) 순이다. 아열대 과수는 ▲ 망고(62.0ha, 159농가) ▲ 백향과(36.5ha, 156농가) ▲ 바나나(29.3ha, 61농가) 순이다.

아열대작물 가운데 망고, 파파야, 용과, 올리브의 재배면적은 최근 3년간 꾸준히 증가하고 있는 것으로 나타났



선정 기준

고온성 작물

저온성 작물

그 외

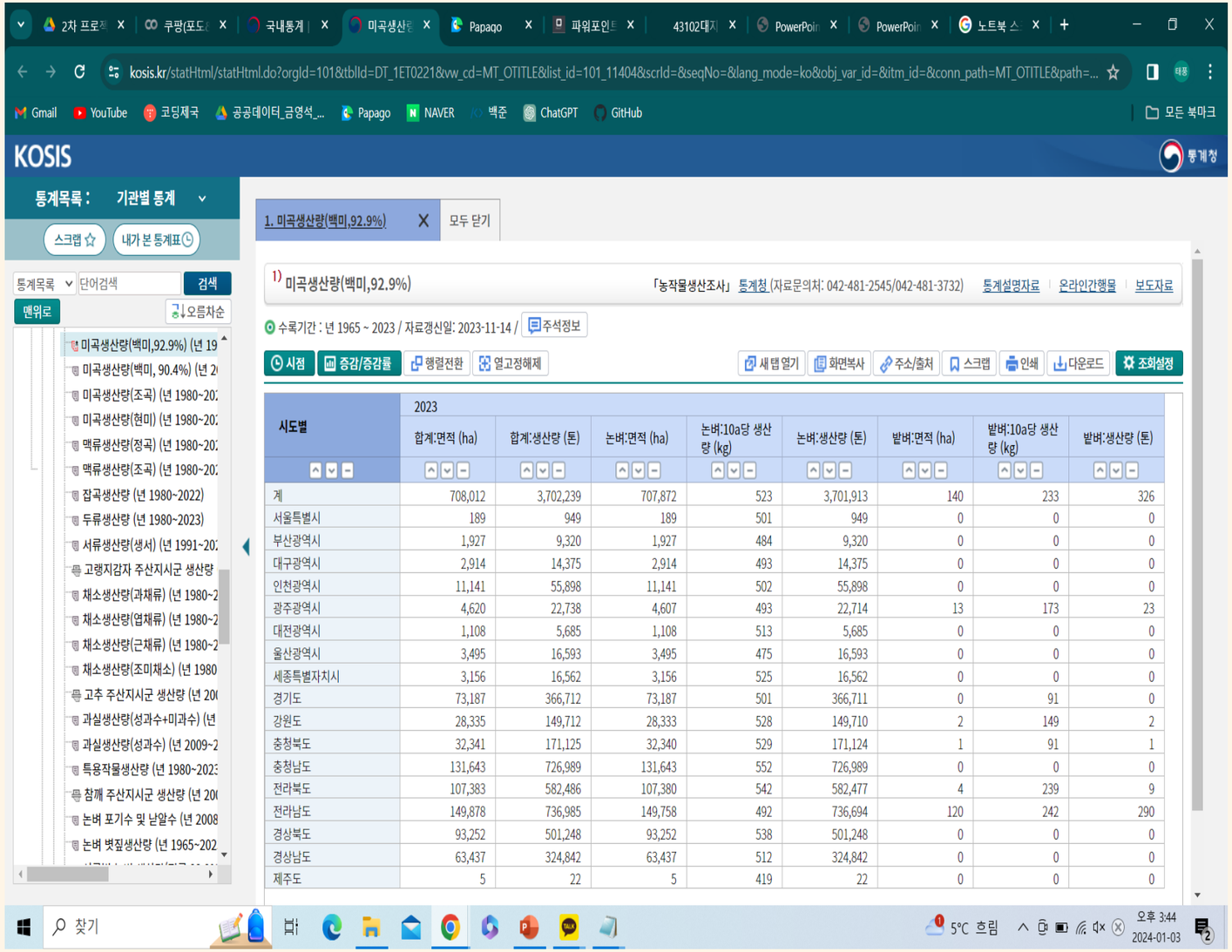
데이터 선정

기관명	데이터명	범위 및 이용과정
KOSIS 국가통계포털	농작물생산조사	1980 ~ 2023년 1년 시군구 10a당 생산량(kg)

기관명	데이터명	범위 및 이용과정
쿠팡	-	HTML 태그를 활용한 검색 데이터

기관명	데이터명	범위 및 이용과정
기상청 기상자료개방 포털	종관기상관측 (ASOS)	관측기관 개시일 ~ 2023년 지점별 전체 데이터

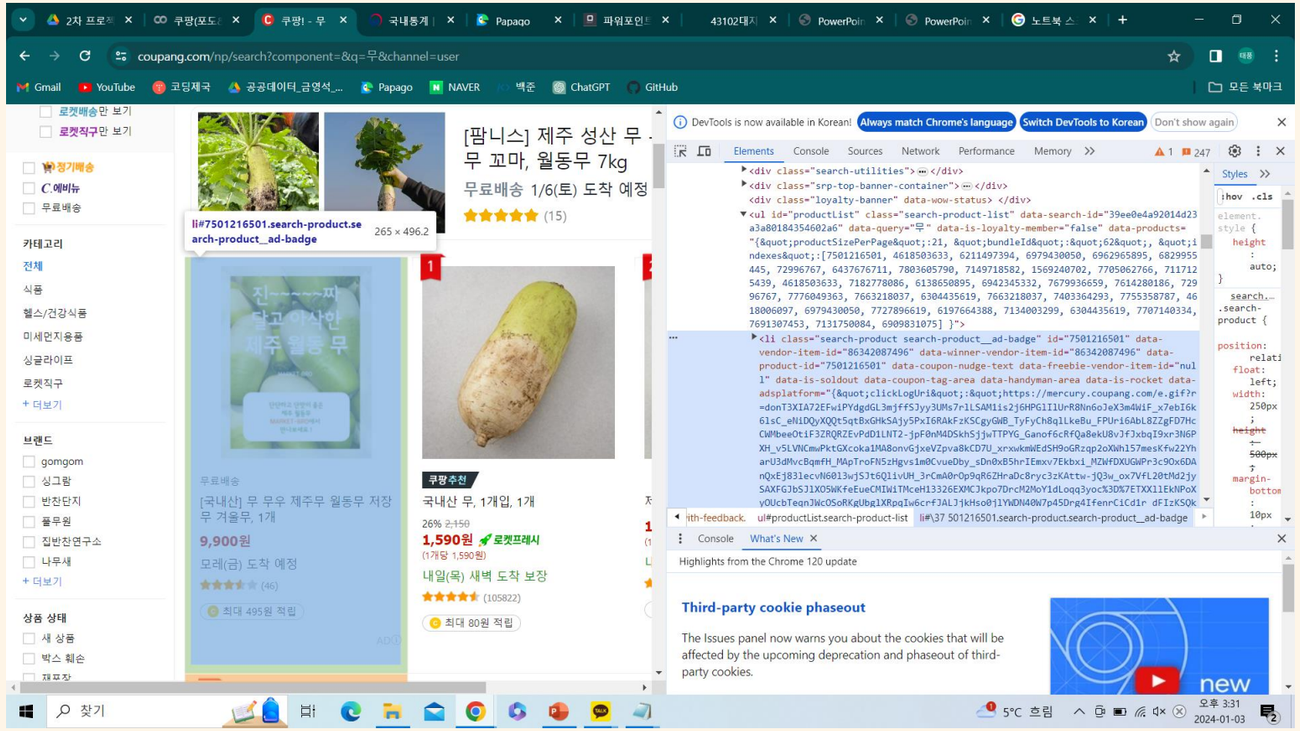
날씨 및 농작물 데이터 추출



시도별	2023							
	합계:면적 (ha)	합계:생산량 (톤)	논벼:면적 (ha)	논벼:10a당 생산량 (kg)	논벼:생산량 (톤)	밭벼:면적 (ha)	밭벼:10a당 생산량 (kg)	밭벼:생산량 (톤)
계	708,012	3,702,239	707,872	523	3,701,913	140	233	326
서울특별시	189	949	189	501	949	0	0	0
부산광역시	1,927	9,320	1,927	484	9,320	0	0	0
대구광역시	2,914	14,375	2,914	493	14,375	0	0	0
인천광역시	11,141	55,898	11,141	502	55,898	0	0	0
광주광역시	4,620	22,738	4,607	493	22,714	13	173	23
대전광역시	1,108	5,685	1,108	513	5,685	0	0	0
울산광역시	3,495	16,593	3,495	475	16,593	0	0	0
세종특별자치시	3,156	16,562	3,156	525	16,562	0	0	0
경기도	73,187	366,712	73,187	501	366,711	0	91	0
강원도	28,335	149,712	28,333	528	149,710	2	149	2
충청북도	32,341	171,125	32,340	529	171,124	1	91	1
충청남도	131,643	726,989	131,643	552	726,989	0	0	0
전라북도	107,383	582,486	107,380	542	582,477	4	239	9
전라남도	149,878	736,985	149,758	492	736,694	120	242	290
경상북도	93,252	501,248	93,252	538	501,248	0	0	0
경상남도	63,437	324,842	63,437	512	324,842	0	0	0
제주도	5	22	5	419	22	0	0	0

지점	시간	평균기온(℃)	최저기온(℃)	최고기온(℃)	평균최저기온(℃)	평균최고기온(℃)	평균 해면기압(hPa)
제주(184)	1923		-1.7	34.6			
제주(184)	1924	14.1	-2.7	33.5	11	18.1	
제주(184)	1925	14	-1.6	32.3	10.9	17.8	
제주(184)	1926	14.3	-2.9	35.7	11	18.2	1016.8
제주(184)	1927	14.5	-3.2	33.1	11.3	18.3	1015.9
제주(184)	1928	14.2	-2.3	31.4	11	17.9	1016.7
제주(184)	1929	14.6	-3.8	33.8	11.2	18.4	1016.9

웹 스크래핑 과정




```
import requests
import re
from bs4 import BeautifulSoup

keyword='무'
pageNum=1
proList=[]

def findSogoung(keyword, pageNum, proList):
    url='https://www.coupang.com/np/search?q='+keyword+'&channel=user&component=&eventCategory=SRP&trcid=&traid=&sorter=scoreDesc&minPrice=&maxPrice=&priceRange=&filterType=&listSize=60&filter=&isPriceRange=false&brand=&offerCondition=&rating=0&page='+str(p
    headers = {"User-Agent":"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/109.0.0.0 Safari/537.36", "Accept-Language": "ko-KR,ko;q=0.8,en-US;q=0.5,en;q=0.3"}

    res = requests.get(url, headers=headers)
    res.raise_for_status() # 웹페이지의 상태가 정상인지 확인

    soup = BeautifulSoup(res.text, "lxml") # 가져온 HTML 문서를 파서를 통해 BeautifulSoup 객체
    ul_tag = soup.select_one('#productList') # id가 productList인 ul 태그 선택

    if ul_tag:
        li_tags = ul_tag.find_all('li', class_=lambda x: x and 'ad-badge-text' not in x.split())


        for li_tag in li_tags:
            p_t = li_tag.select_one('div.name').text
            p_t = re.sub('ft', '', p_t)
            p_t = p_t.lstrip()
            proList.append(p_t)
            print('[' + str(len(proList)) + ']', p_t)
            print('='*50)

    else:
        print("해당하는 상품이 없습니다.")


while(len(proList)<300):
    proCnt=findSogoung(keyword, pageNum, proList)
```

```
[1] 무우 유기농 무 가을무 김장무 2kg, 1개
==================================================
[2] 국내산 무, 1개입, 1개
==================================================
[3] 곰곰 국내산 무, 2개입, 1개
==================================================
[4] 국내산 친환경 절단무, 400g, 1개
==================================================
[5] 제주무 10kg내외 겨울 김장무 제주 월동 무우 진맛갈, 10kg, 1개
==================================================
[6] [맛통령] 신선한 알타리무 맛있는 총각무, 1개, 2kg 1단
==================================================
[7] 국내산 절단무, 2입, 1개
==================================================
[8] 국내산 무, 5개입, 1개
==================================================
[9] [팜니스] 제주 성산 무 무우 월동무 꼬마, 월동무 7kg
==================================================
[10] 곰곰 친환경 무, 1개입, 1개
==================================================
[11] 석하 제주 월동 무말랭이, 450g, 1개
==================================================
[12] 강화 옛향기 순무 5kg 10kg 2023년 생산/밭에서 바로발송, 1개
==================================================
[13] 국내산 세척 다발무, 5kg, 1봉
==================================================
[14] 곰곰 국내산 무청제거 다발무, 5kg, 1개
==================================================
[15] 국내산 무, 3개입, 1개
==================================================
[16] 국내산 세척 무, 1개, 1개입
==================================================
[17] [동치미 사이즈] 제주무 월동무 제주도 무우 세척무, 5kg, 1개
==================================================
[18] 국내산 친환경 무, 1개입, 1개
==================================================
[19] 국내산 절단무 450g, 1개
==================================================
[20] 국내산 무, 5개입, 2개
==================================================
[21] 석하 제주 월동 무말랭이, 450g, 1개
==================================================
[22] 아루이팜 유기농 수박무 말랭이 200g 과일무, 2개
==================================================
[23] 맛있는 햇 무, 1박스, 5kg
==================================================
```

웹 스크래핑 과정

```
import pandas as pd

df=pd.DataFrame(proList, columns=[keyword])
df
```

	무
0	무우 유기농 무 가을무 김장무 2kg, 1개
1	국내산 무, 1개입, 1개
2	
3	
4	제주무 10kg내외 겨울 김
...	
355	제주 월동무
356	
357	강원도 무 고랭지
358	
359	[대박농수산] 햇 다발 무 동치미 무 무우

360 rows × 1 columns

```
[ ] from collections import Counter

    all_tokens = [token for tokens_list in df['형태소'] for token in tokens_list]
    token_counts = Counter(all_tokens)
```

```
형태소 등장 횟수 :
무 : 353번
개 : 348번
국내 : 228번
산 : 228번
개입 : 151번
친환경 : 60번
세척 : 60번
곰곰 : 49번
절단 : 48번
다발 : 44번
무우 : 39번
내외 : 32번
제주 : 30번
월 : 27번
동무 : 27번
입 : 24번
고랭 : 24번
지 : 24번
햇 : 22번
맛있는 : 21번
21번
8번
8번
8번
18번
8번
7번
3번
2번
2번
12번
1번
1번
10번
0번
```

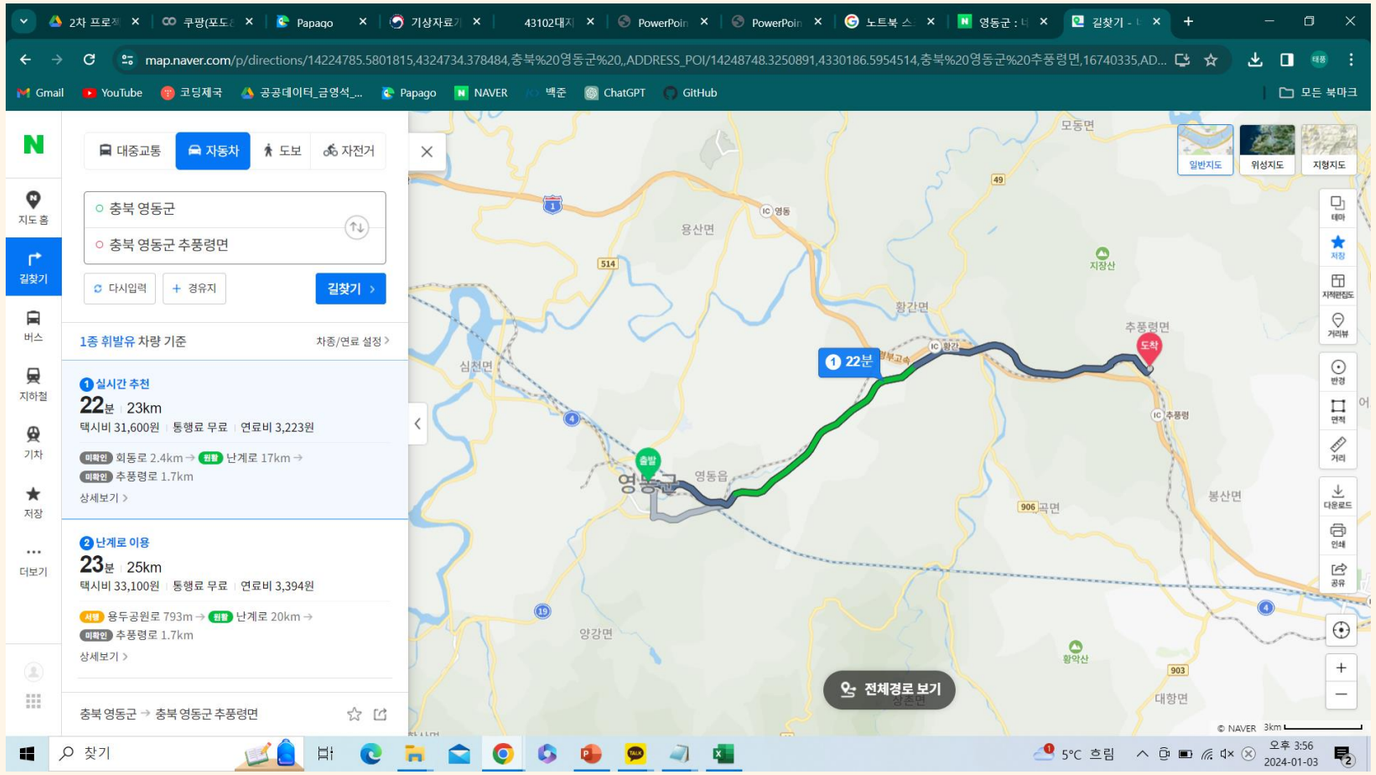
쌀	고추	배추	무	사과	포도
이천(경기도)	청양(충남)	해남(전남)	강화도(인천)	청송(경북)	상주(경북)
충주(충북)	밀양(경남)	진도(전남)	제주(제주도)	안동(경북)	영동(충북)
-	-	평창(강원도)	평창(강원도	충주(충북)	경산(경북)

	무
0	무우 유기농 무 가을무 김장무 개
1	국내산 무 개입 개
2	곰곰 국내산 무 개입 개
3	국내산 친환경 절단무 개
4	제주무 내외 겨울 김장무 제주 월동 무우 진맛깔 개

2	곰곰 국내산 무 개입 개	[곰곰, 국내, 산, 무, 개입, 개]
3	국내산 친환경 절단무 개	[국내, 산, 친환경, 절단, 무, 개]
4	제주무 내외 겨울 김장무 제주 월동 무우 진맛깔 개	[제, 주무, 내외, 겨울, 김장, 무, 제주, 월동, 무우, 진, 맛깔, 개]
...	...	...
355	제주 월동무 세척 무우 내외 개	[제주, 월, 동무, 세척, 무우, 내외, 개]
356	국내산 제주도 월동무 무우	[국내, 산, 제주도, 월, 동무, 무우]
357	강원도 무 고랭지 무우 햇무 김장무 국내산 무개 개	[강원도, 무, 고랭, 지, 무우, 햇무, 김장, 무, 국내, 산, 무, 개, 개]
358	제주도 친환경 무 개	[제주도, 친환경, 무, 개]
359	대박농수산 햇 다발 무 동치미 무 무우 홈무 달랑 겨울 내외 다발무내외 개	[대박, 농수산, 햇, 다발, 무, 동치미, 무, 무우, 홈무, 달랑, 겨울, 내외...

360 rows × 2 columns

데이터 정제



▪ 자료보기

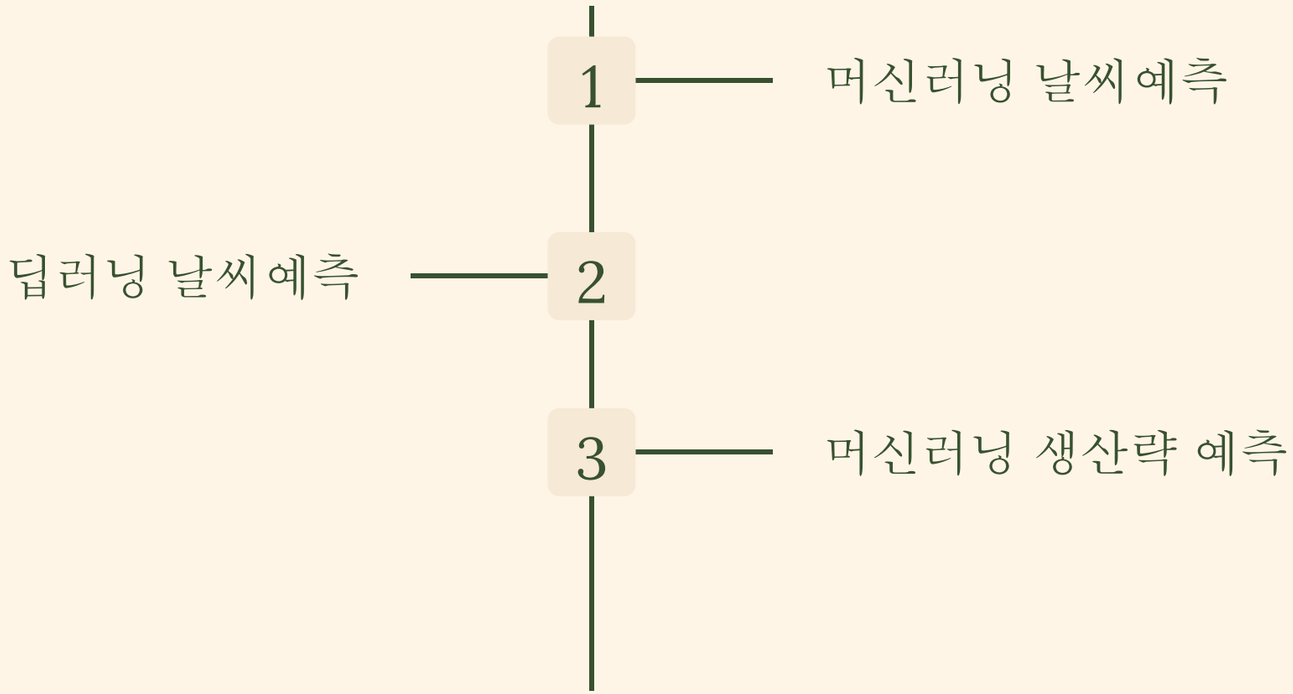
CSV Excel

※조회 결과는 10건만 표출 됩니다. 상세결과는 파일 다운로드를 이용해주세요

지점	시간	평균기온(℃)	최저기온(℃)	최고기온(℃)	평균최저기온(℃)	평균최고기온(℃)	평균해면기압(hPa)
추풍령(135)	1938			27.8			
추풍령(135)	1939		8.1	39.8			
추풍령(135)	1940		12.7				
추풍령(135)	1941		-11.2	27.8			
추풍령(135)	1942		-2.5	38.4			
추풍령(135)	1943		-17	28.9			
추풍령(135)	1944		-1.2	36.7			
추풍령(135)	1945						
추풍령(135)	1946						
추풍령(135)	1947						

영동군에는 기상관측소가 없어 거리가 가장 가까운 지점인 추풍령으로 선택 하는 모습

모델링 구현과정



머신러닝 날씨예측

```
df_data=data[['year','branch']]
# df_data=df_data.values.reshape(-1, 1)
df_data[-5:]
```

	year	branch
471	2019	203
472	2020	203
473	2021	203
474	2022	203
475	2023	203

```
df_labels=data.drop(['year','branch'], axis=1)
df_labels.head()
```

	avg_temp	min_temp	max_temp	avg_min_temp	avg_max_temp	avg_sea_pre	tot_rain	day_max_rain	avg_rel_hum	tot_sun_time	sun_rate	tot_sun	avg_wind	m
0	12.8	-14.0	34.7	7.6	18.5	1016.5	1299.15	120.6	71.3	2208.7	50.14	4877.39	1.8	
1	12.8	-14.0	34.7	7.6	18.5	1016.5	1299.15	120.6	71.3	2208.7	50.14	4877.39	1.8	
2	11.8	-17.5	36.6	5.9	18.5	1016.5	804.90	84.7	71.7	2824.3	63.70	6647.98	1.4	
3	10.5	-22.5	34.5	4.6	17.4	1016.5	1031.80	80.0	72.2	2661.8	60.04	8713.28	1.4	
4	11.9	-14.9	34.8	6.3	18.3	1016.5	1184.00	119.2	75.7	2520.5	56.86	8246.02	1.3	

```
Ridge Model:
r2_score: 0.13687716870361985
MSE: 97614.7088281032
RMSE: 312.43352705512126
--------------------------------------------------
Lasso Model:
r2_score: 0.13401157222441568
MSE: 97614.67689690001
RMSE: 312.43347595432215
--------------------------------------------------
DecisionTreeRegressor Model:
r2_score: 0.22495203190774388
MSE: 86844.73331682693
RMSE: 294.6943048598444
--------------------------------------------------
Linear Regression Model:
r2_score: 0.13687725145939011
MSE: 97614.70649177089
RMSE: 312.4335233161942
--------------------------------------------------
Random Forest Regression Model:
r2_score: 0.5430431642042646
MSE: 50248.25963152036
RMSE: 224.1612357913838
--------------------------------------------------
K-Nearest Neighbors Regression Model:
r2_score: 0.592378853152579
MSE: 48356.86071949126
RMSE: 219.9019343241238
--------------------------------------------------
Multilayer Perceptron Regression Model:
r2_score: -8.223393128198524
MSE: 101695.85901607196
RMSE: 318.8978817992869
```

```
# 데이터 분할
knn_params = {
    'n_neighbors': [3, 5, 7],
    'weights': ['uniform', 'distance'],
    'p': [1, 2]

}
best_knn_model = find_best_params(knn_model, knn_params, X_train, y_train, X_test, y_test)
```

```
최적의 파라미터: {'n_neighbors': 3, 'p': 2, 'weights': 'distance'}
Mean Squared Error: 46606.63727880946
Root Mean Squared Error: 215.8857042020371
```

```
knn_params = {
    'n_neighbors': [1,2,3,4],
    'weights': ['distance'],
    'p': [2]

}
best_knn_model = find_best_params(knn_model, knn_params, X_train, y_train, X_test, y_test)
```

```
최적의 파라미터: {'n_neighbors': 3, 'p': 2, 'weights': 'distance'}
Mean Squared Error: 46606.63727880946
Root Mean Squared Error: 215.8857042020371
```

```
best_knn_model = KNeighborsRegressor(n_neighbors = 3, weights='distance', p=2)
best_knn_model.fit(X_train, y_train)
```

```
KNeighborsRegressor
KNeighborsRegressor(n_neighbors=3, weights='distance')
```

딥러닝 날씨예측

3. 시리즈화 코드

1. 트레인 데이터 (80%)

```
def makeSeriesData(data, n_in=6, list=col_list):
  for col in list:
    for i in range(1, n_in+1):
      data[str(i)+col]=data[str(i-1)+col].shift(1)
  return data
```

							합계 강수량 (mm)	일 최다 강수량(mm)
							1251.9	130.9
							1227.0	133.7
2	14.0	-1.6	32.3	10.9	17.8	1016.3	1612.8	173.1
3	14.3	-2.9	35.7	11.0	18.2	1016.8	956.9	58.0
4	14.5	-3.2	33.1	11.3	18.3	1015.9	2042.1	301.2

d1	NaN	NaN	NaN	NaN	NaN
d2	**d1**	NaN	NaN	NaN	NaN
d2	**d3**	**d1**	NaN	NaN	NaN
d4	**d3**	**d2**	**d1**	NaN	NaN
d5	**d4**	**d3**	**d2**	**d1**	NaN
d6	**d5**	**d4**	**d3**	**d2**	**d1**

4. 시리즈화 된 df

```
temp_data.head()
```

	0avg_temp	0min_temp	0max_temp	0avg_min_temp	0avg_max_temp	0avg_sea_pre	0tot_rain	0day_max_rain	0avg_rel_hum	0tot_sun_time	...	3tot_small_Eva	4tot_small_Eva	5tot_small_Eva	6tot_small_Eva	1tot_big_Eva	2tot_big_Eva
0	15.0	-1.7	34.6	11.7	18.6	1016.3	1251.9	130.9	72.9	1610.7	...	NaN	NaN	NaN	NaN	NaN	NaN
1	14.1	-2.7	33.5	11.0	18.1	1016.3	1227.0	133.7	75.3	2105.5	...	NaN	NaN	NaN	NaN	744.1	NaN
2	14.0	-1.6	32.3	10.9	17.8	1016.3	1612.8	173.1	75.8	2019.0	...	NaN	NaN	NaN	NaN	744.1	744.1
3	14.3	-2.9	35.7	11.0	18.2	1016.8	956.9	58.0	72.7	2285.2	...	1127.3	NaN	NaN	NaN	744.1	744.1
4	14.5	-3.2	33.1	11.3	18.3	1015.9	2042.1	301.2	75.2	2187.0	...	1526.4	1127.3	NaN	NaN	744.1	744.1

5 rows × 112 columns

												합계 일사량 (MJ/m2)	평균 풍속 (m/s)	최다 풍향 (16방위)	합계 소형 증발량(mm)	합계 대형 증발량(mm)
												4597.30	3.1	340	1074.8	630.4
												5044.28	2.8	340	1285.8	747.7
												5086.58	3.5	20	1279.8	762.5
83	16.2	-1.8	34.0	13.5	19.4	1016.0	1527.4	161.5	64.3	1704.7	38.32	4791.09	3.2	20	1119.9	659.3
84	16.5	0.8	34.0	13.8	19.5	1016.0	2139.8	420.0	61.8	1671.8	39.07	4735.87	3.3	320	1198.2	715.9

딥러닝 날씨예측

5. df 결측치 행 제거

```
temp_data.dropna(axis=0, inplace=True)
temp_data
```

	0avg_temp	0min_temp	0max_temp	0avg_min_temp	0avg_max_temp	0avg_sea_pre	0tot_rain	0day_max_rain	0avg_rel_hum	0tot_sun_time	...	3tot_small_Eva	4tot_small_Eva	5tot_small_Eva	6tot_small_E
6	14.6	-3.8	33.8	11.2	18.4	1016.9	774.5	60.2	71.6	2407.4	...	1724.1	1441.3	1526.4	112
7	15.0	-2.1	35.5	11.9	18.8	1016.7	1233.5	91.0	75.3	2138.2	...	1572.4	1724.1	1441.3	152
8	14.2	-5.7	32.7	11.0	17.9	1016.6	1518.9	180.3	76.8	2072.8	...	1357.1	1572.4	1724.1	144
9	14.4	-2.4	35.7	11.2	18.2	1016.8	1131.2	160.7	73.4	2269.6	...	1638.4	1357.1	1572.4	172
10	14.2	-4.4	34.0	11.2	17.8	1016.6	1684.6	200.5	77.7	2122.2	...	1388.4	1638.4	1357.1	157
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
75	17.0	-2.9	37.4	14.1	20.0	1016.1	1581.1	219.0	74.9	1807.4	...	1266.9	1398.2	1210.8	127
76	16.0	-1.6	32.8	13.2	19.1	1015.8	2526.0	248.2	71.2	1656.3	...	1253.0	1266.9	1398.2	121
77	15.7	-1.7	36.0	12.7	18.8	1016.3	1189.4	147.5	66.3	2046.2	...	1273.0	1253.0	1266.9	139
78	16.1	-3.1	35.8	13.2	19.3	1016.3	1388.6	73.0	65.1	1910.1	...	1181.3	1273.0	1253.0	126
79	15.8	0.6	35.0	13.1	18.7	1016.1	1704.1	215.0	67.9	1794.9	...	1141.2	1181.3	1273.0	125

74 rows × 112 columns

6. 학습 데이터

```
X_train = X_train.sort_index(axis=1)
X_train
```

	1avg_max_temp	1avg_min_temp	1avg_rel_hum	1avg_sea_pre	1avg_temp	1avg_wind	1day_max_rain	1max_temp	1max_wind_dir	1min_temp	...	6day_max_rain	6max_temp	6max_wind_dir	6min_temp	6sun_rate	6tot_big_Eva	6tot_rain	6to
6	17.9	11.0	76.4	1016.7	14.2	4.7	106.0	31.4	360.0	-2.3	...	130.9	34.6	160.0	-1.7	42.98	744.1	1251.9	
7	18.4	11.2	71.6	1016.9	14.6	5.1	60.2	33.8	340.0	-3.8	...	133.7	33.5	340.0	-2.7	42.98	744.1	1227.0	
8	18.8	11.9	75.3	1016.7	15.0	5.0	91.0	35.5	340.0	-2.1	...	173.1	32.3	340.0	-1.6	42.98	744.1	1612.8	
9	17.9	11.0	76.8	1016.6	14.2	5.1	180.3	32.7	360.0	-5.7	...	58.0	35.7	340.0	-2.9	42.98	744.1	956.9	
10	18.2	11.2	73.4	1016.8	14.4	5.1	160.7	35.7	320.0	-2.4	...	301.2	33.1	360.0	-3.2	42.98	744.1	2042.1	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
75	19.4	13.5	70.6	1016.8	16.4	3.2	86.4	34.4	20.0	-0.9	...	59.4	34.5	290.0	0.4	42.64	839.0	1232.9	
76	20.0	14.1	74.9	1016.1	17.0	3.2	219.0	37.4	20.0	-2.9	...	132.2	34.5	290.0	-0.8	35.62	763.3	1552.8	
77	19.1	13.2	71.2	1015.8	16.0	3.4	248.2	32.8	340.0	-1.6	...	189.3	34.6	20.0	-0.6	48.58	770.0	1448.9	
78	18.8	12.7	66.3	1016.3	15.7	3.4	147.5	36.0	360.0	-1.7	...	204.5	35.8	20.0	0.3	43.77	646.1	1472.9	
79	19.3	13.2	65.1	1016.3	16.1	3.2	73.0	35.8	340.0	-3.1	...	47.1	35.5	20.0	-1.5	42.99	633.7	881.1	

74 rows × 96 columns

7. 검증 데이터

```
y_train=temp_data.iloc[:,:16]
y_train
```

	0avg_temp	0min_temp	0max_temp	0avg_min_temp	0avg_max_temp	0avg_sea_pre	0tot_rain	0day_max_rain	0avg_rel_hum	0tot_sun_time	0sun_rate	0tot_sun	0avg_wind	0max_wind_dir	0tot_small_Eva	0tot_big_Eva
6	14.6	-3.8	33.8	11.2	18.4	1016.9	774.5	60.2	71.6	2407.4	42.98	4465.545	5.1	340	1638.4	744.1
7	15.0	-2.1	35.5	11.9	18.8	1016.7	1233.5	91.0	75.3	2138.2	42.98	4465.545	5.0	340	1388.4	744.1
8	14.2	-5.7	32.7	11.0	17.9	1016.6	1518.9	180.3	76.8	2072.8	42.98	4465.545	5.1	360	1374.7	744.1
9	14.4	-2.4	35.7	11.2	18.2	1016.8	1131.2	160.7	73.4	2269.6	42.98	4465.545	5.1	320	1541.9	744.1
10	14.2	-4.4	34.0	11.2	17.8	1016.6	1684.6	200.5	77.7	2122.2	42.98	4465.545	5.0	340	1423.0	744.1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
75	17.0	-2.9	37.4	14.1	20.0	1016.1	1581.1	219.0	74.9	1807.4	40.66	4436.990	3.2	20	1181.3	597.2
76	16.0	-1.6	32.8	13.2	19.1	1015.8	2526.0	248.2	71.2	1656.3	37.26	4389.350	3.4	340	1141.2	528.2
77	15.7	-1.7	36.0	12.7	18.8	1016.3	1189.4	147.5	66.3	2046.2	45.92	4799.210	3.4	360	1248.9	744.1
78	16.1	-3.1	35.8	13.2	19.3	1016.3	1388.6	73.0	65.1	1910.1	42.97	4685.870	3.2	340	1189.9	673.1
79	15.8	0.6	35.0	13.1	18.7	1016.1	1704.1	215.0	67.9	1794.9	40.35	4559.420	3.2	340	1118.0	618.0

74 rows × 16 columns

8. 딥러닝 모델 생성

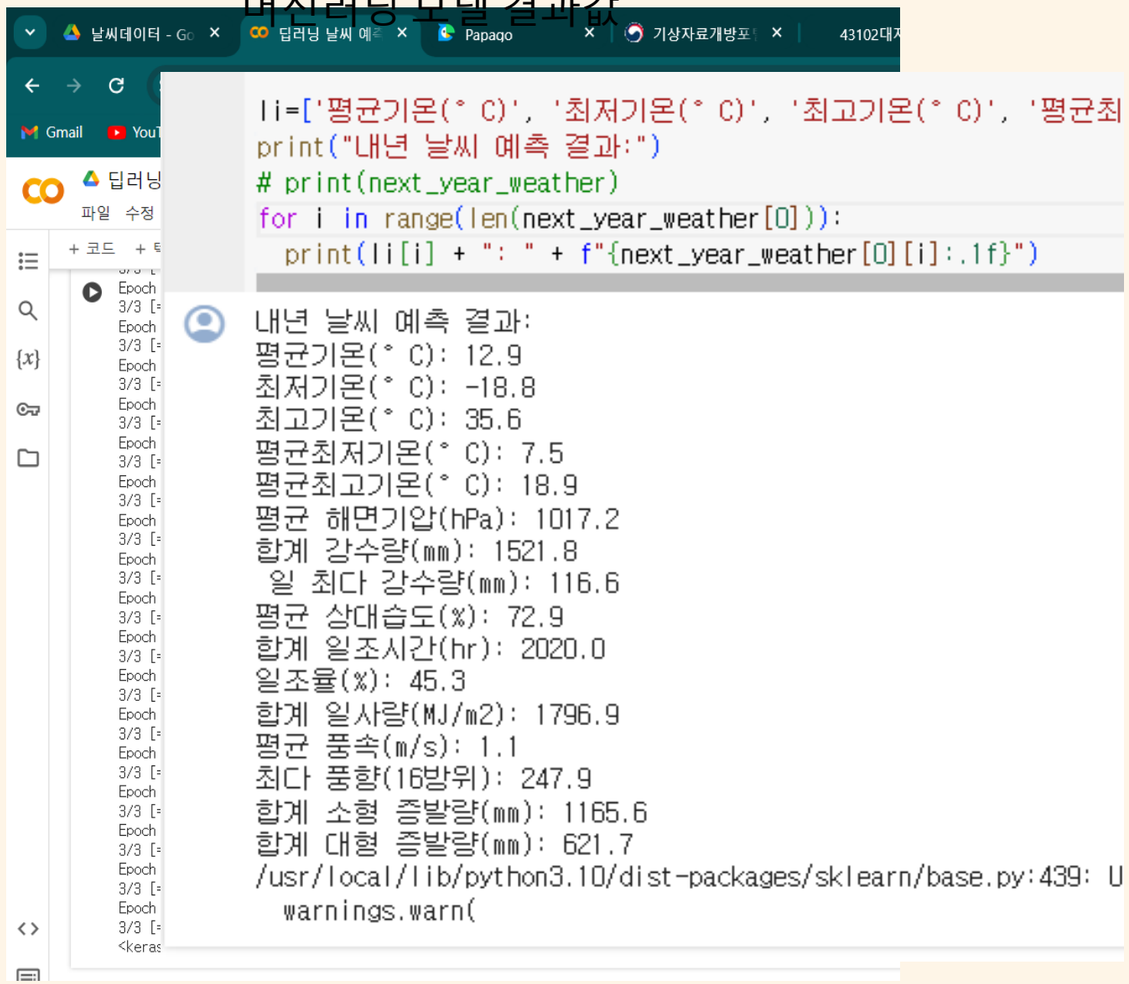
```
# LSTM 모델 생성
model = Sequential()
# 입력 데이터의 shape는 (X_train.shape[1], 1)로 지정
model.add(LSTM(64, activation='relu', input_shape=(6, 16)))
model.add(Dense(128, activation='relu'))
model.add(Dense(256, activation='relu'))
model.add(Dense(512, activation='relu'))
model.add(Dense(1024, activation='relu'))
model.add(Dense(units=y_train.shape[1]))
model.compile(optimizer='nadam', loss='mse')

# LSTM 모델 학습
X_train_reshaped = X_train.values.reshape((74, 6, 16))
model.fit(X_train_reshaped, y_train, epochs=3735, batch_size=32)
```

```
3/3 [==============================] - 0s 25ms/step - loss: 1566.8574
Epoch 3708/3735
3/3 [==============================] - 0s 24ms/step - loss: 555.5364
Epoch 3709/3735
3/3 [==============================] - 0s 24ms/step - loss: 742.4183
Epoch 3710/3735
3/3 [==============================] - 0s 28ms/step - loss: 242.5600
Epoch 3711/3735
3/3 [==============================] - 0s 25ms/step - loss: 252.7809
Epoch 3712/3735
3/3 [==============================] - 0s 30ms/step - loss: 151.2422
Epoch 3713/3735
3/3 [==============================] - 0s 25ms/step - loss: 127.9204
Epoch 3714/3735
3/3 [==============================] - 0s 28ms/step - loss: 190.2980
```

딥러닝 날씨예측

9. 모델 loss 값

머신러닝 모델 결과값



```
li=['평균기온(° C)', '최저기온(° C)', '최고기온(° C)', '평균최
print("내년 날씨 예측 결과:")
# print(next_year_weather)
for i in range(len(next_year_weather[0])):
  print(li[i] + ": " + f"{next_year_weather[0][i]:.1f}")
```

```
내년 날씨 예측 결과:
평균기온(° C): 12.9
최저기온(° C): -18.8
최고기온(° C): 35.6
평균최저기온(° C): 7.5
평균최고기온(° C): 18.9
평균 해면기압(hPa): 1017.2
합계 강수량(mm): 1521.8
 일 최다 강수량(mm): 116.6
평균 상대습도(%): 72.9
합계 일조시간(hr): 2020.0
일조율(%): 45.3
합계 일사량(MJ/m2): 1796.9
평균 풍속(m/s): 1.1
최다 풍향(16방위): 247.9
합계 소형 증발량(mm): 1165.6
합계 대형 증발량(mm): 621.7
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: U
  warnings.warn(
```

10. 딥러닝 모델 결과값

```
transposed_data = list(map(list, zip(*formatted_values_resha

# 데이터프레임 생성
formatted_values_reshaped = pd.DataFrame(transposed_data, c

# 열 벡터를 주어진 형식으로 출력
result = formatted_values_reshaped.to_dict(orient='records'
result = formatted_values_reshaped.iloc[0]
print(result)
```

```
1/1 [==============================] - 0s 24ms/step
평균최고기온(° C)          19.8
평균최저기온(° C)           6.6
평균 상대습도(%)          76.7
평균 해면기압(hPa)       1095.2
평균기온(° C)            22.0
평균 풍속(m/s)            0.9
일 최다 강수량(mm)        178.8
최고기온(° C)             48.3
최다 풍향(16방위)         282.3
최저기온(° C)             13.8
일조율(%)                42.8
합계 대형 증발량(mm)       667.8
합계 강수량(mm)          1609.1
합계 소형 증발량(mm)      1275.4
합계 일사량(MJ/m2)       5163.8
합계 일조시간(hr)         2111.3
Name: 0, dtype: float64
```

날씨예측 결과

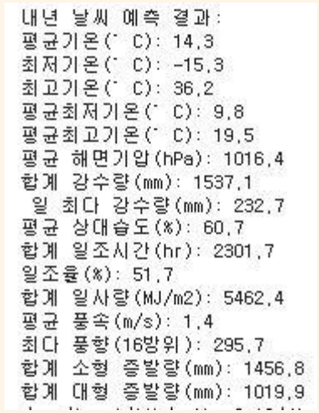
대관령

대전

밀양

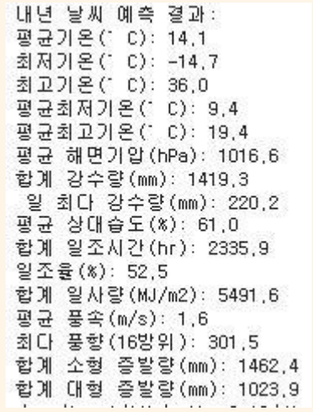
의성

이천



청주

해남



제주도

수확량 예측

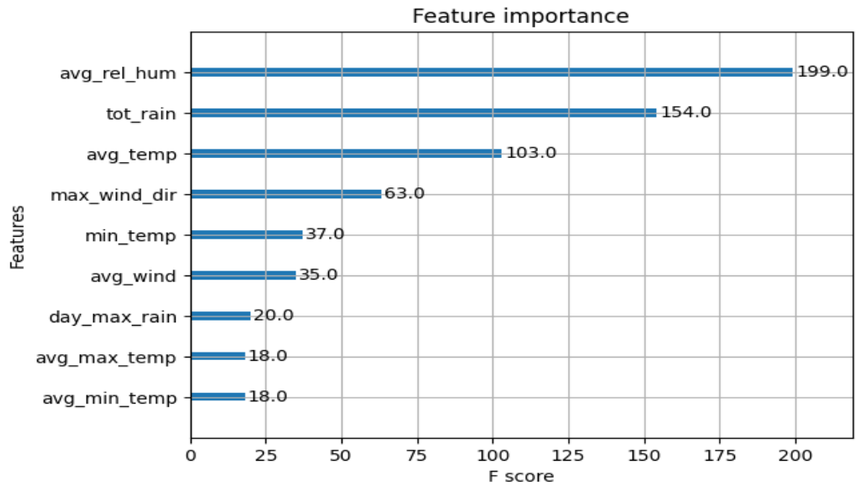
```
data=apple_weather.drop('apple', axis=1)
data=data.drop('일시', axis=1)

data.tail()
```

지점	avg_temp	min_temp	max_temp	avg_min_temp	avg_max_temp	tot_rain	day_max_rain	avg_rel_hum	tot_sun_time	sun_rate	avg_wind	max_wind_dir
136	12.4	-19.2	40.4	5.7	20.2	1079.4	88.5	64.7	2448.1	55.00	1.3	360
136	12.9	-13.7	37.6	6.3	20.7	831.3	88.5	65.1	2374.3	53.34	1.2	360
136	12.3	-15.9	35.9	6.1	19.5	1128.9	71.5	71.3	2174.3	48.85	1.1	290
136	12.4	-21.6	37.0	6.3	19.7	961.7	53.7	73.6	2079.9	46.72	1.0	290
136	12.2	-16.3	37.2	5.6	19.7	615.1	34.9	68.6	2295.8	51.58	1.1	290

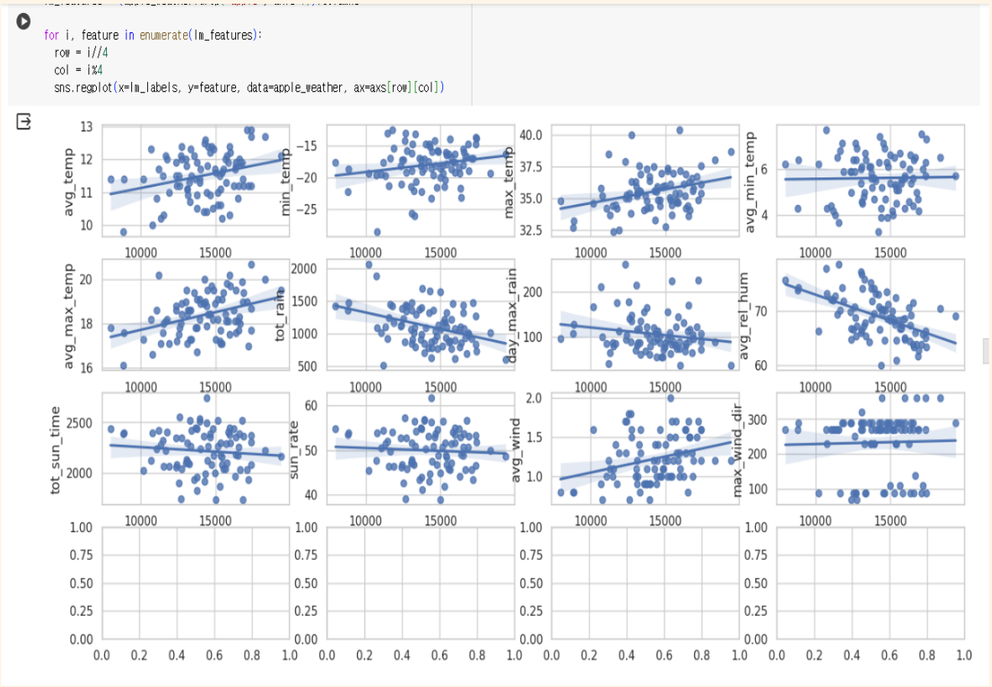
```
label = apple_weather[['apple']]
label.tail()
```

지점	apple
136	15940.0
136	17370.0
136	14860.0
136	14750.0
136	15980.0

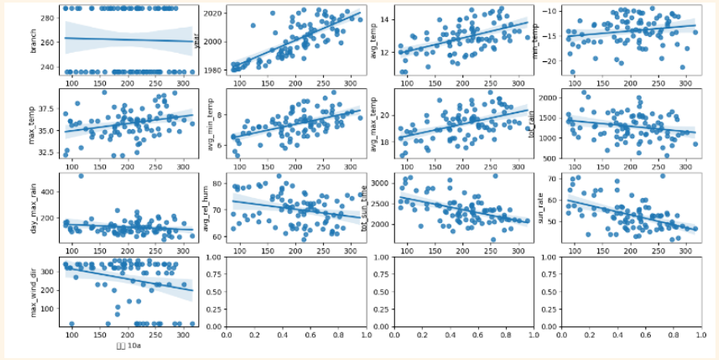
```
from xgboost import plot_importance
import matplotlib.pyplot as plt

# 특성 중요도 시각화
plot_importance(xg_reg)
plt.show()
```



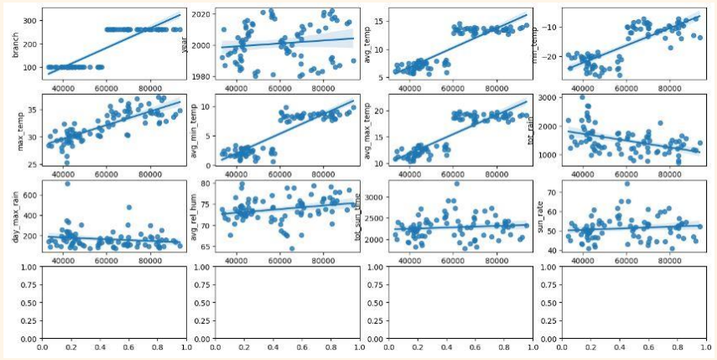

```
for i, feature in enumerate(lm_features):
  row = i//4
  col = i%4
  sns.regplot(x=lm_labels, y=feature, data=apple_weather, ax=axs[row][col])
```



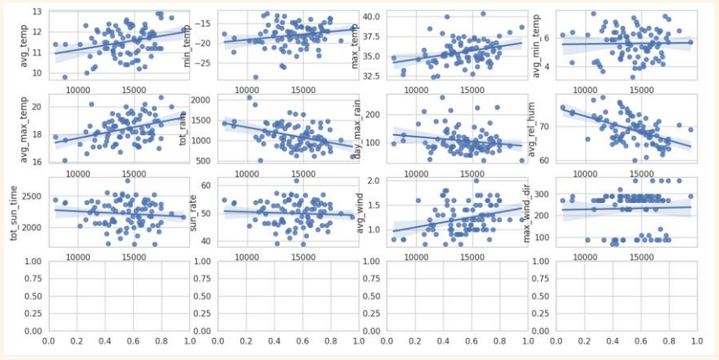
수확량 예측



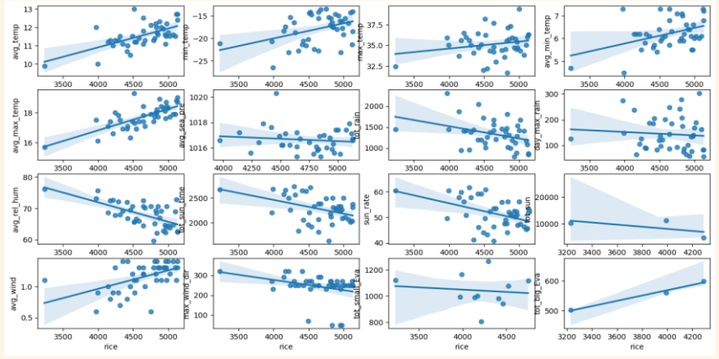
고추 산점도



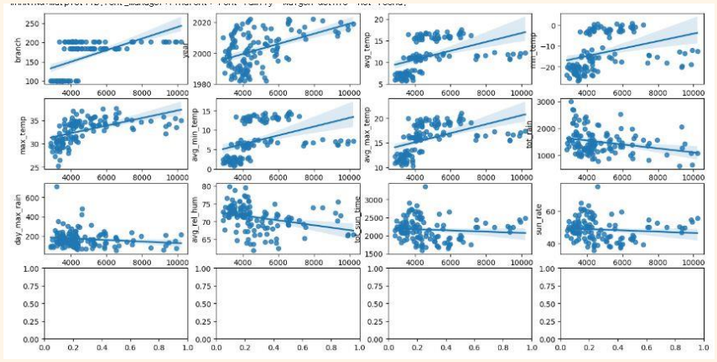
배추 산점도



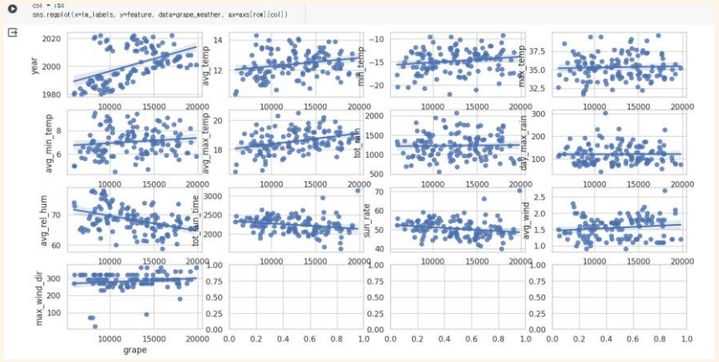
사과 산점도



쌀 산점도



무 산점도



포도 산점도

수확량 예측

```python
from sklearn.neighbors import KNeighborsRegressor
knn_model = KNeighborsRegressor()
knn_model.fit(X_train, y_train)

# k-최근접 이웃 회귀 모델
knn_y_pred = knn_model.predict(X_test)
knn_mse = mean_squared_error(y_test, knn_y_pred)
knn_r2 = r2_score(y_test, knn_y_pred)
print("K-Nearest Neighbors Regression Model:")
print("Mean Squared Error:", knn
print("R-squared:", knn_r2)
knn_rmse=np.sqrt(knn_mse)
print('tree_rmse : ', knn_rmse)
```

```
K-Nearest Neighbors Regression M
Mean Squared Error: 6779945.2442
R-squared: -0.21790987051613464
tree_rmse :  2603.832798830132
```

```python
from sklearn.tree import DecisionTreeRegressor

# 의사결정 트리 회귀 모델 생성 및 학습
tree_model = DecisionTreeRegressor()
tree_model.fit(X_train, y_train)

# DecisionTreeRegressor 모델 평가
tree_y_pred = tree_model.predict(X_test)
tree_mse = mean_squared_error(y_test, tree_y_pred)
tree_r2 = r2_score(y_test, tree_y_pred)
```

```python
from sklearn.ensemble import GradientBoostingRegressor

# 그래디언트 부스팅 회귀 모델 생성 및 학습
gb_model = GradientBoostingRegressor()
gb_model.fit(X_train, y_train)

# GradientBoostingRegressor 모델 평가
gb_y_pred = gb_model.predict(X_test)
gb_mse = mean_squared_error(y_test, gb_y_pred)
gb_r2 = r2_score(y_test, gb_y_pred)
print("GradientBoostingRegressor Model:")
print("Mean Squared Error:", gb_mse)
print("R-squared:", gb_r2)
gb_rmse=np.sqrt(gb_mse)
print('tree_rmse : ', gb_rmse)
```

```
GradientBoostingRegressor Model:
Mean Squared Error: 4749096.717909308
R-squared: 0.14689992907077198
tree_rmse :  2179.242234793853
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_g
  y = column_or_1d(y, warn=True)
```

```python
from sklearn.linear_model import LinearRegression


lr = LinearRegression()
lr.fit(X_train, y_train)
lr.score(X_test, y_test)

# LinearRegression 모델 평가
lr_y_pred = lr.predict(X_test)
lr_mse = mean_squared_error(y_test, lr_y_pred)
lr_r2 = r2_score(y_test, lr_y_pred)
print("lr Model:")
print("Mean Squared Error:", lr_mse)
print("R-squared:", lr_r2)
lr_rmse=np.sqrt(lr_mse)
print('tree_rmse : ', lr_rmse)
```

```
lr Model:
Mean Squared Error: 4641684.634512958
R-squared: 0.16619481005698644
tree_rmse :  2154.4569233365883
```

```python
from sklearn.linear_model import Lasso

# Lasso 회귀 모델 생성 및 학습
lasso_model = Lasso(alpha=1.0)  # alpha는
lasso_model.fit(X_train, y_train)

# Lasso 모델 평가
lasso_y_pred = lasso_model.predict(X_test)
lasso_mse = mean_squared_error(y_test, las
lasso_r2 = r2_score(y_test, lasso_y_pred)
print("Lasso Model:")
print("Mean Squared Error:", lasso_mse)
print("R-squared:", lasso_r2)
rmse=np.sqrt(lasso_mse)
print('tree_rmse : ', rmse)
```

```
Lasso Model:
Mean Squared Error: 4642227.717364218
R-squared: 0.1660972538170422
tree_rmse :  2154.582956714412
```

```python
from sklearn.linear_model import Ridge
from sklearn.metrics import mean_squared_error, r2_score


ridge_model = Ridge(alpha=1.0)  # alpha는 규제 강도를 나타냅니다
ridge_model.fit(X_train, y_train)

# Ridge 모델 평가
ridge_y_pred = ridge_model.predict(X_test)
ridge_mse = mean_squared_error(y_test, ridge_y_pred)
ridge_r2 = r2_score(y_test, ridge_y_pred)
print("Ridge Model:")
print("Mean Squared Error:", ridge_mse)
print("R-squared:", ridge_r2)
rmse=np.sqrt(ridge_mse)
print('tree_rmse : ', rmse)
```

```
Ridge Model:
Mean Squared Error: 4646827.44831496
R-squared: 0.16527098494247194
tree_rmse :  2155.6501219620404
```

```
<ipython-input-60-ec153584da92>:6: DataConversionWarning
  rf_model.fit(X_train, y_train)
Random Forest Regression Model:
Mean Squared Error: 4196466.31482696
R-squared: 0.2461712356098279
rmse :  2048.52784087182
```

수확량 예측

```python
import xgboost as xgb
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

# 데이터를 훈련 세트와 테스트 세트로 나눔
X_train, X_test, y_train, y_test = train_test_split(data, label, test_size=0.2, random_state=42)

# XGBoost 모델 초기화 및 학습
```

```python
# 최적의 하이퍼파라미터로 모델 초기화 및 훈련
xg_reg = xgb.XGBRegressor(objective='reg:squarederror')
xg_reg.fit(X_train, y_train)

# 새로운 데이터에 대한 예측 [평균 기온, 최다풍향, 합계강수량, 평균 상대습도]
new_data = np.array([[14.3, 295.7, 1537.1, 60.7]])
predictions = xg_reg.predict(new_data)

print("Predictions:", predictions)
```

```
Predictions: [15086.88]
```

```
내년 날씨 예측 결과:
평균기온(° C): 14.3
최저기온(° C): -15.3
최고기온(° C): 36.2
평균최저기온(° C): 9.8
평균최고기온(° C): 19.5
평균 해면기압(hPa): 1016.4
합계 강수량(mm): 1537.1
 일 최다 강수량(mm): 232.7
평균 상대습도(%): 60.7
합계 일조시간(hr): 2301.7
일조율(%): 51.7
합계 일사량(MJ/m2): 5462.4
평균 풍속(m/s): 1.4
최다 풍향(16방위): 295.7
합계 소형 증발량(mm): 1456.8
합계 대형 증발량(mm): 1019.9
```

9초

```python
# 데이터 분할
xg_params = {
    'n_estimators': [50, 100, 150],
    'learning_rate': [0.01, 0.1, 0.2],
    'max_depth': [3, 4, 5]
}
xg_reg = xgb.XGBRegressor(objective='reg:squarederror')
best_lasso_model = find_best_params(xg_reg, xg_params, X_train, y_train, X_test, y_test)
```

```
최적의 파라미터: {'learning_rate': 0.01, 'max_depth': 3, 'n_estimators': 150}
Mean Squared Error: 4339250.707892682
Root Mean Squared Error: 2083.0868219766267
```

```python
# 데이터 분할
xg_params = {
    'n_estimators': range(100, 201, 5),
    'learning_rate': [0.006,0.007,0.008,0.009, 0.01,0.02,0.03,0.04,0.05],
    'max_depth': [3]
}
xg_reg = xgb.XGBRegressor(objective='reg:squarederror')
best_lasso_model = find_best_params(xg_reg, xg_params, X_train, y_train, X_test, y_test)
```

```
최적의 파라미터: {'learning_rate': 0.02, 'max_depth': 3, 'n_estimators': 115}
Mean Squared Error: 4588261.051807139
Root Mean Squared Error: 2142.02265436366
```

활용 방안

선제적 대응 : 국가적으로 농작물을 수입할 때, 생산량의 예측값을 활용 농작물 수입. 해당 지역에서 기존 작물의 생산이 불리해 지기 전에 생산지를 다른곳으로 옮겨, 국내 생산량을 유지 할 수 있음

새로운 특산물 생산 : 해외 지역의 기후데이터를 바탕으로 우리나라에 해당 작물이 생산 가능한지 판단할 수 있음

소감
데이터 수집 후에 이틀이면 끝난다고 무시했는데, 최선의 값을 도출하는게 생각보다 만만치 않다.

담당업무
1. 주제 선정
2. 날씨 예측 및 생산량 알고리즘 구현
3. 지역 선정
4. Ppt제작
5. 발표