

「2024년 임업통계 스마트활용 경진대회」 양식[아이디어 기획]

1) 추진배경 및 필요성

산림청 및 각 지역에서 예비 임업인을 양성하고 무료로 교육하는 정책을 보여주는 것들을 기사를 통해 확인했습니다. 이를 통해 임업인을 양성하는 것을 현정책 기조로 판단하고, 예비 임업인들이 필요한 것이 무엇일까를 생각했습니다. 실제 저희가 임업계에 뛰어들고자 한다면, 어디서 어떤 작물을 재배할 것인지부터 결정해야한다고 생각합니다.

그런데 현재 재배적지를 알고자 해도 한눈에 쉽게 알 수 있는 방법이 없었습니다. 물론, 한국임업진흥원의 공공데이터에 “단기임산물재배적지도” 데이터셋이 존재하지만, 코딩에 대한 지식이 없는 사람들이 이 자료를 확인하기엔 어려움이 있을 것이라 생각합니다. 또한 임산물이 자라나는 데에 어떤 기후 및 토양 특징이 있어야하는지에 대한 정보를 한 곳에 확인할 수 있는 곳도 없어 현재 임업계 종사를 앞두고 있는 사람들에게 정보를 찾는 데에 있어 어려움이 있을 것이라고 생각했습니다.

이런 현정책 기조와 문제를 파악해보니, 임업계 진출을 앞둔 예비임업인 양성 및 교육을 위해서는 임산물의 재배적지에 대한 정보를 한 곳으로 모아 쉽게 확인할 수 있는 매체가 필요함을 깨달았습니다. 거기서 더 나아가 소비자가 임산물 가격 추이를 확인할 수 있는 카테고리 또한 만들어 임업인들과 소비자들 모두에게 접근성 좋은 웹사이트를 만들었습니다.

2) 아이디어 기획 세부 내용

1. 임산물 재배환경 추천 서비스

- 임산물 재배에 적합한 환경 조건(토양,기후 조건 등)을 분석하여 최적의 재배 지역을 추천
- 사용자가 선택한 임산물에 대한 유리한 재배 지역 정보 제공
- 실시간 날씨 정보와 연계하여 재배 환경 변화에 따른 실시간 알림 서비스
- 재배 환경 개선을 위한 맞춤형 팁 제공

1.1. 핵심 기술

- GIS(지리정보시스템) 기술 : 지도를 기반으로 지역별 재배 환경 데이터를 시각화
- 빅데이터 분석 : 토양깊이유형, 토성코드, 토양형코드, 토양유효수분량 데이터, 임산물 생산량 데이터, 연평균 기온, 최저-최고 기온, 상대습도, 강수량 데이터를 수집하고 가공하여 연관성, 상관성 분석

◆ 상관성 분석 구체적 과정 설명(이하 <코드 및 결과1&2>참조)

기후데이터에서 어떤 변수가 생산량과 연관성이있는지 파악하기 위해 상관분석을 실시했습니다. 하지만 토양데이터는 범주형데이터이기 때문에 카이제곱분석, ANOVA분석을 사용해 분석했습니다. 토양데이터와 생산량과의 관계를 확인하기 위한 분석에서 P-value값이 0.1 이내에 해당하는 변수를 1차적으로 추출했습니다. 기후데이터에선 상관성을 확인하기 위해 히트맵 등으로 상관관계를 파악할 수 있는 시각화도 진행했습니다.

구체적으로, 기후데이터는 상관계수를 분석해 각 작물에 대해 상관계수 절댓값이 0.2 이상인 기후 변수를 1차적으로 유의하다 판단하였습니다. 다음으로 토양데이터와 기후데이터에서 각각 1차적으로 중요하다고 판단한 요인들 중 SVM과 로지스틱 회귀등의 통계적 분석방법을 활용하여 2차적

으로 중요 요인들을 각 작물별로 선정했습니다.

◆ SVM, 랜덤포레스트, 로지스틱 회귀 구체적 설명(이하 <코드 및 결과 3>참조)

SVM과 랜덤포레스트, 로지스틱회귀 분석을 돌릴 때 범주형변수를 수치형으로 변환해줘야 하기 때문에 수치형 변수로 변환을 진행한 후 각 품목별 생산량의 평균으로 품목 생산량 이진화를 진행하고 이를 토대로 SVM모델, 랜덤포레스트, 로지스틱 회귀 분석을 학습시키고 중요도를 판단했습니다.

-SVM: 성능을 확인하기 위해 AUC 성능평가를 실시한 결과, 마는 성능이 좋음을 확인할 수 있지만, 나머지 것들은 성능이 다소 떨어짐을 확인 할 수 있습니다.

-랜덤포레스트 : 복분자딸기, 오갈피, 더덕, 생표고의 모델 성능이 좋음을 확인하였습니다

-로지스틱회귀: 밤, 도라지의 모델성능이 좋음을 확인하였습니다.

>결론: 즉, 마는 SVM 모델을 통해서, 복분자딸기, 생표고, 더덕, 오갈피는 랜덤포레스트가 더 좋은 성능, 밤, 도라지 같은 경우에는 로지스틱회귀에서 좋은 결과를 보인다는 것을 확인할 수 있습니다. 생산량으로 미루어보아 해당 모델들을 통해 각 임산물에 어떤 조건이 얼마만큼의 중요도를 갖는지를 파악할 수 있고, 더 나아가 각 조건이 들어맞는 지역을 매칭해 추천해줄 수 있습니다.

◆ 상관성 분석 결과(음의 상관관계가 두 개 이상인 품목)에 관한 분석

-오갈피: 오갈피는 최저/고기온과 연평균에서 모두 음의 상관관계를 보이며, 기온이 너무 낮거나 높으면 생육에 부정적인 영향을 미칩니다. 최저기온이 지나치게 낮을 경우 뿌리의 생장이 저해되고, 최고기온이 높으면 잎이 마르거나 생육 정지가 발생할 수 있습니다.

오갈피는 연평균 기온이 10~15도일 때 뿌리와 잎의 생장이 가장 활발합니다. 이 범위를 벗어나 기온이 높아질수록 생장 속도가 느려지고 생산량이 감소할 가능성이 높습니다.

과도한 습도는 곰팡이병이나 뿌리 썩음과 같은 병해를 유발할 가능성이 커 오갈피 생육에 부정적인 영향을 미칩니다. 적정 상대습도(50~70%)를 유지하는 것이 중요합니다.

오갈피는 낮과 밤의 온도차가 클수록(15~20℃) 뿌리의 양분 축적과 품질 향상에 유리합니다. 특히 고지대(해발 200~800m)에서 재배되는 오갈피는 낮은 기온과 큰 일교차 덕분에 품질이 더 우수한 경우가 많습니다.

-생표고: 강수량과 상대습도는 표고버섯 생장에 중요한 변수입니다. 최적 수분함량은 50%이며, 수분이 부족(23% 이하)할 경우 균사가 사멸합니다. 자실체(버섯 본체)는 85~95%의 수분을 함유하며, 적정 습도를 유지하지 않으면 품질과 수량에 직접적인 영향을 미칩니다. 온도의 영향: 표고균은 24~28℃에서 가장 잘 성장하며, 45℃까지도 생존이 가능합니다. 온도는 자실체의 발생 속도, 형태, 수량에 영향을 미치지만, 최근 톱밥(배지) 재배 방식의 확산으로 인해 상대적으로 영향력이 약해진 것으로 보입니다. 결론: 강수량과 상대습도는 표고버섯의 생존과 생장에 필수적인 주요 기후 변수입니다. 온도는 중요한 변수이지만, 표고버섯의 특성과 재배 방식 변화로 인해 영향력은 상대적으로 작아진 것으로 판단됩니다. 위 내용을 통해 표고버섯 재배 시 물과 습도 관리의 중요성이 강조됩니다.

-더덕: 더덕은 최저기온, 연평균기온, 최고기온, 상대습도에서 음의 상관관계 온도차이에서 양의 상관관계를 보입니다. 최저기온과 최고기온: 너무 높거나 낮으면 더덕의 생장에 부정적일 것으로

예측되었습니다. 예를 들어, 최저기온이 낮으면 더덕의 생장에 스트레스를 줄 수 있어 생산량이 감소한다. 연평균기온: 10~15°C 더덕은 연평균 기온이 이 범위 내에 있을 때 뿌리의 성장과 발달이 가장 활발합니다.. 즉 연평균 기온이 높다면 더덕의 생장에 안좋은 영향을 끼칠 것으로 예측됩니다. 상대습도 : 과도한 습도는 병해 발생 가능성을 높여 더덕의 생장에 좋지 않다. 온도차이 : 낮과 밤의 온도차가 클수록 더덕 뿌리의 당분 축적과 품질 향상에 유리합니다. 이 예로 고지대(해발 300~800m)에서 재배되는 더덕이 품질이 더 좋은 경우가 많습니다.

-도라지: 도라지의 생산량은 강수량, 온도차이, 최고기온에 크게 영향을 받으며, 적절한 강수와 온도차가 생육에 중요한 역할을 합니다. 도라지가 잘 자라는 조건은 월간 강수량이 충분히 많은 지역(100~150mm 이상), 낮과 밤의 온도차가 큰 환경(15~20°C), 최고기온이 25~30°C 범위에 있을 때 최적의 생육을 보입니다. 강수량이 부족하거나 과도하면 생장에 부정적인 영향을 줄 수 있으므로 배수와 관수 관리가 중요합니다. 특히, 적당한 일교차는 도라지 뿌리의 품질과 당분 축적을 높이는 데 유리합니다. 이러한 환경에서 도라지의 생산량과 품질을 극대화할 수 있습니다.

- ◆ <코드 및 결과 3&4 참조> SVM, 랜덤포레스트, 로지스틱회귀로 중요도 추출된 각 품목별 토양과 기후를 구분하여 변수Top3를 추출했습니다. 다만, 상대 습도를 포함한 기후데이터를 타 데이터와 조건을 맞추기 위해 4분위수로 조정해 범주화를 했습니다. 또한 지역별로 각기 다른 토양 및 기후 데이터 특성을 갖고 있습니다. 그래서 해당 품목 중요도와 지역 변수 특성과 매칭해 품목별로 조건에 맞는 최적의 지역을 추출할 수 있습니다.

->로지스틱회귀분석의 경우 음수의 경우 단기생산물 성장의 부정적인 영향을 끼친다고 볼 수 있기 때문에 밤과 도라지는 양수의 값만 우선적으로 추출 하지만 도라지의 경우는 주요변수를 추출하였을때 3개가 나와 TOP3를 추출할 때 음수인것도 고려함

->랜덤포레스트같은 경우에는 양수,음수가 나오지않아 값이 큰 것 기준으로 TOP3를 추출

->SVM도 로지스틱회귀분석과 유사하게 양수/음수값으로 도출되는데 마같은 경우도 주요변수가 3개만 도출되어 TOP3를 할 때 음수도 고려함

※토양관련변수들은 범주형에서 수치형으로 변환하면서 각각의 변수들을 확실하게 구분가능
ex)토성코드-> '토성코드_1','토성코드_2','토성코드_3','토성코드_4' 하지만 기후관련데이터는 특정값을 구분할 수 없어 범주화를 진행하여 생산물 별 가장 많은 지역의 값을 참고하여 추천지역을 뽑아냈습니다.

실제로 우리가 도출해낸게 정말 유의미한지 판단하기위해 조사를 통해 검증하였습니다.

ex)밤같은 경우 토성코드_1이 주요한 변수로 도출되었는데 토성코드_1은 양토를 의미합니다. 실제로 밤의 재배에서 양토가 적합하고 복분자의 경우에는 토양유효수분량_3으로 복분자딸기같은 경우 지하수위가 낮은곳에서 잘자라는데 지하수위가 낮다는 것은 지하수가 깊어져서 토양이 충분한 수분을 공급받기 어려운 상황을 의미하여 토양에 저장되는 수분량이 적어져서 유효수분량이 낮아 질수있다는걸 의미합니다 실제로 토양유효수분량_3은 유효수분이 양호하여 너무 과습하지도 않고 건조하지도 않아 좋다는 걸 볼 수 있습니다.

2. 임산물 판매량 정보

- 특정 임산물의 연도별 판매량 데이터를 그래프로 제공

2.1. 핵심 기술

- 사용자 "맞춤형" 필터링 : 특정 임산물별 판매량 정보 확인 가능
- 연도별 데이터 시각화 : 특정 임산물 판매량을 연도를 기준으로 그래프를 통해 시각화
- 데이터 통합 : 임산물별 판매량 데이터를 수집, 가공하여 연도별로 정리

3. 임산물 가격 정보

- 특정 임산물의 연도별 가격 데이터를 그래프로 제공

3.1. 핵심 기술

- 사용자 "맞춤형" 필터링 : 특정 임산물별 가격 정보 확인 가능
- 연도별 데이터 시각화 : 특정 임산물 가격을 연도를 기준으로 그래프를 통해 시각화
- 데이터 통합 : 임산물별 가격 데이터를 수집, 가공하여 연도별로 정리

4. 지역별 임산물 생산 정보

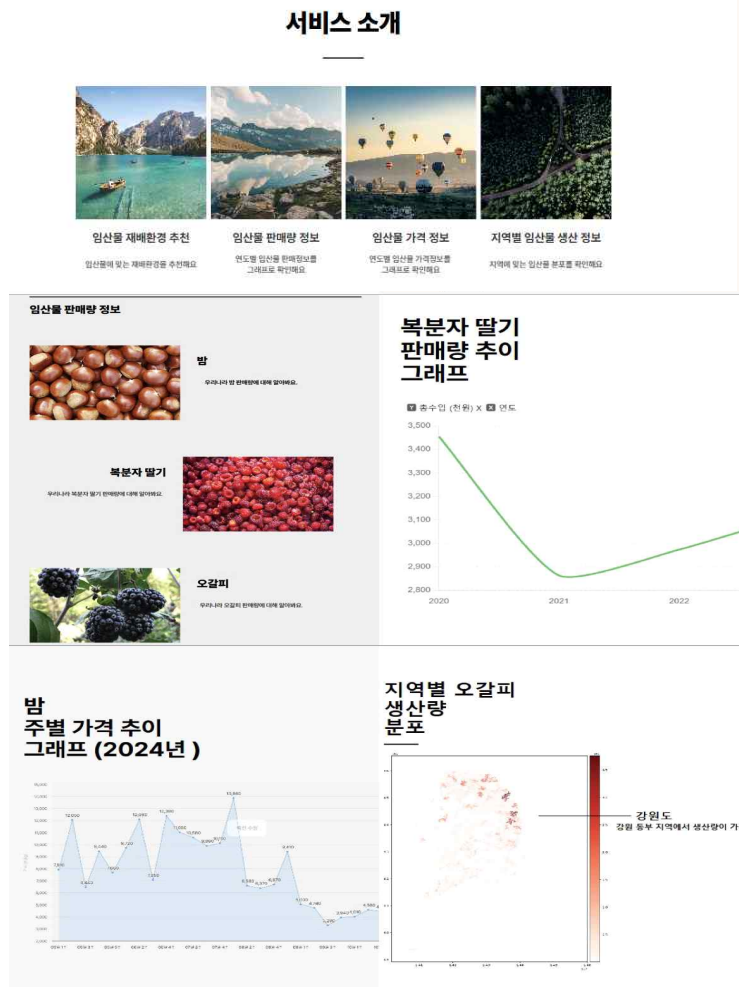
- 특정 임산물의 생산량을 지리적으로 시각화하여 지도 상에서 제공

4.1. 핵심 기술

- 지리적 시각화 : python의 geopandas 모듈을 사용하여 임산물의 생산량을 지리 데이터와 결합시키고, 지도 형식으로 시각화하여 생산량의 산포도와 밀집도를 보여줌

[웹페이지 이미지 및 링크]

링크: <https://app.publr.co/channels/L2NoYW5uZWxzLzE3MDUx/pages/home>



밤 재배시 고려사항 top3

1	최고기온
2	강수량(mm)
3	토성코드_1

밤 재배지역 추천

1	경상남도 거창군
2	대전광역시 대덕구
3	대전광역시 유성구
4	전라북도 남원시
5	충청남도 부여군

[데이터 전처리]

1. 임업6종(생산량데이터) (필수데이터활용)

- 접근 경로: kfss.forest.go.kr-> 임산물생산조사(kosis.kr)

<다운받은 파일의 형태>

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	임산물별(1)	임산물별(2)	임산물별(3)	2022	2022	2022	2022	2022	2022	2022	2022	2022	2022	2022	2022	2022	2022	2022	2022
2	임산물별(1)	임산물별(2)	임산물별(3)	인천광역시	인천광역시	중구	중구	동구	동구	미추홀구	미추홀구	연수구	연수구	남동구	남동구	부평구	부평구	계양구	계양구
3	임산물별(1)	임산물별(2)	임산물별(3)	생산량	생산액 (원)	생산량	생산액 (원)	생산량	생산액 (원)	생산량	생산액 (원)	생산량	생산액 (원)	생산량	생산액 (원)	생산량	생산액 (원)	생산량	생산액 (원)
4	합계	합계	합계	-	15381210531	-	2.27E+08	-	40064000	-	2.51E+08	-	62464000	-	7.85E+08	-	1.03E+09	-	3.59E+08
5	조경재	조경재	소계	-	1698415000	-	-	-	-	-	-	-	-	-	-	-	-	-	-
6	조경재	조경수	단풍나무류 (그루)	50	5000000	-	-	-	-	-	-	-	-	-	-	-	-	-	-
7	조경재	조경수	느티나무류 (그루)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
8	조경재	조경수	동백나무 (그루)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
9	조경재	조경수	회양목 (그루)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
10	조경재	조경수	주목 (그루)	649	91420000	-	-	-	-	-	-	-	-	-	-	-	-	-	-
11	조경재	조경수	철쭉류 (그루)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
12	조경재	조경수	소나무 (그루)	20	8000000	-	-	-	-	-	-	-	-	-	-	-	-	-	-
13	조경재	조경수	배롱나무 (그루)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
14	조경재	조경수	벚나무 (그루)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	조경재	조경수	이팝나무 (그루)	260	39000000	-	-	-	-	-	-	-	-	-	-	-	-	-	-
16	조경재	조경수	기타교목류 (그루)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
17	조경재	조경수	기타관목류 (그루)	17500	54995000	-	-	-	-	-	-	-	-	-	-	-	-	-	-
18	조경재	분재소재	해송 (그루)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
19	조경재	분재소재	향나무 (그루)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
20	조경재	분재소재	소사나무 (그루)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
21	조경재	분재소재	소나무 (그루)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
22	조경재	분재소재	철쭉 (그루)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
23	조경재	분재소재	기타분재소재 (그루)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
24	조경재	분재완재	해송 (그루)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
25	조경재	분재완재	향나무 (그루)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
26	조경재	분재완재	소사나무 (그루)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

<전처리>

- 지역을 기준으로, 2023년 임산물 주요 10대 임산물을 기반으로 '밤', '복분자딸기', '오갈피', '마', '도라지', '더덕', '생표고'만을 추출했다. 이하 첨부된 표가 정제된 데이터이다.
- 추출 방식: ex) =FILTER(A3:O147, C3:C147="밤 (kg)")



<산림통계시스템-통계보고서PDF발체>

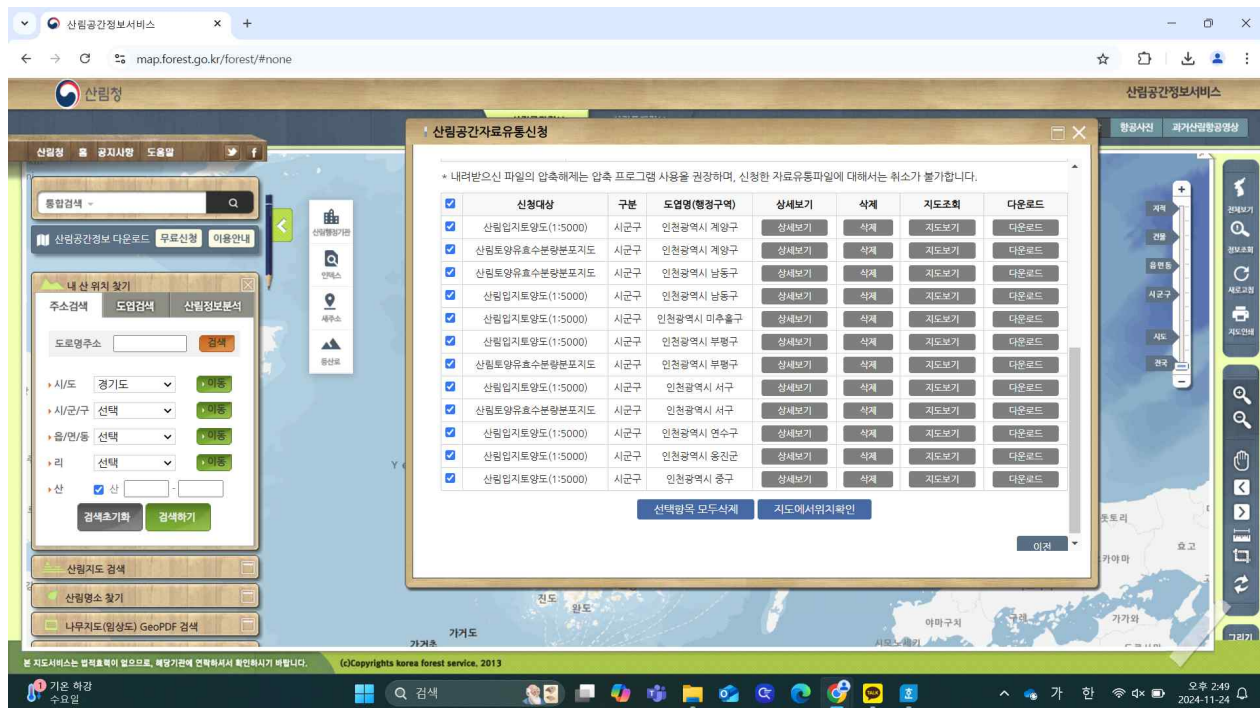
번호	지역	밤 (kg)	복분자딸기	오갈피 (kg)	마 (kg)	도라지 (kg)	더덕 (kg)	생표고 (kg)
1	제주특별자치도	110	28155	79727	-	405321	4965809	109409
2	제주시	110	28155	79727	-	405321	4047105	67650
3	서귀포시	-	-	-	-	-	918704	41759
4	강원도	210955	140046	113875	13209	1649108	6485061	
5	춘천시	29511	70	2700	90	9261	89324	
6	원주시	43619	986	7053	-	30962	3698	
7	강릉시	4009	10	238	-	1235	1740	
8	동해시	1691	-	-	-	-	-	
9	태백시	-	-	1279	-	1491	22117	
10	속초시	581	-	-	-	-	-	
11	삼척시	3689	1106	27588	-	655607	1034771	
12	홍천군	294	-	13721	-	605600	1275833	
13	횡성군	10229	128101	1584	6027	125396	1548165	
14	영월군	769	-	9550	-	11065	22547	
15	평창군	3275	606	11096	-	50440	1730324	
16	정선군	-	20	13077	-	62020	615419	
17	철원군	-	-	3998	-	1727	15123	
18	화천군	278	-	2632	-	61455	60517	
19	양구군	1437	-	12317	2216	30	39224	
20	인제군	100542	7728	5940	4876	6519	8436	
21	고성군	5706	173	1000	-	3322	10016	

2. 토양코드&토양유효수분량

- 접근 경로: <https://map.forest.go.kr/forest/#none>

<다운받은 파일의 형태>

- SHP 파일형태



<전처리>

- 산림입지토양도(1:5000)에서 "토양깊이유형", "토성코드", "토양형코드"를 추출함

*토양깊이유형: 'SLDPT_TPCD'

*토성코드: 'SCSTX_CD'

*토양형코드: 'SLTP_CD'

- 산림토양유효수분량분포지도에서 "gridcode"를 추출함

*해당 예시 코드는 인천만 해당하지만, 다른 지역 또한 같은 코드로 전처리 진행함.

토양코드&토양유효수분량데이터 전처리	
코드	결과
<p><토양코드> - 예시지역: "인천"</p> <pre> import geopandas as gpd import pandas as pd region_mapping = { 1: "인천광역시 중구", 2: "인천광역시 미추홀구", 3: "인천광역시 연수구", 4: "인천광역시 남동구", 5: "인천광역시 부평구", 6: "인천광역시 계양구", 7: "인천광역시 서구", 8: "인천광역시 용진군" } data_frames = [] for i in range(1, 9): file_path = f'Desktop/인천토양/10/10.shp' df = gpd.read_file(file_path, encoding='cp949') </pre>	


```
df['지역'] = region_mapping[i]
data_frames.append(df)

<토양유효수분량데이터>- 예시지역: '인천'
import geopandas as gpd
import pandas as pd

region_mapping = {
    1: "인천광역시 남동구",
    2: "인천광역시 부평구",
    3: "인천광역시 계양구",
    4: "인천광역시 서구"
}

data_frames = []

for i in range(1, 5):
    file_path = f'Desktop/인천수분/{i}/{i}.shp'
    df = gpd.read_file(file_path, encoding='cp949')
    df['지역'] = region_mapping[i]
    data_frames.append(df)

merged_data = pd.concat(data_frames, ignore_index=True)

merged_data.rename(columns={
    'gridcode': '토양유효수분량'
}, inplace=True)

result = []

for region, group in merged_data.groupby('지역'):
    values = {
        "지역": region,
        "토양유효수분량": group['토양유효수분량'].mode()[0] if not group['토양유효수분량'].mode().empty
    else None
    }
    result.append(values)

result_df = pd.DataFrame(result)

print(result_df)

result_df.to_csv('Desktop/인천수분/인천수분.csv', index=False, encoding='utf-8-sig')
```

<토양코드 결과> - 예시지역: "인천"

지역	토양깊이유형	토성코드	토양형코드
0	인천광역시 계양구	20.0	2 2
1	인천광역시 남동구	20.0	2 2
2	인천광역시 미추홀구	20.0	1 1
3	인천광역시 부평구	20.0	1 1
4	인천광역시 서구	20.0	2 2
5	인천광역시 연수구	20.0	1 1
6	인천광역시 옹진군	20.0	2 2
7	인천광역시 중구	20.0	1 1

<토양유효수분량데이터 결과>- 예시지역: '인천'

지역	토양유효수분량
0	인천광역시 계양구 2
1	인천광역시 남동구 4
2	인천광역시 부평구 3
3	인천광역시 서구 3

3. 기후데이터

-접근 경로: 기후 : 공공데이터 포털(www.data.go.kr)> 우리나라 기후 평년값(최근 30년 데이터 셋)

<다운받은 파일의 형태>

-CSV파일

<전처리>

*결과에 인천만 예시로 첨부함

기후데이터 전처리																																																																		
코드	결과																																																																	
<pre>import pandas as pd from tabulate import tabulate file_path = r"C:\Users\woojoo00\Desktop\bda 조별활동\우리나라기후평년(년별)_1991 (1).xlsx" df2 = pd.read_excel(file_path, sheet_name='연평균값', skiprows=2, usecols=[0, 1, 3, 4, 5, 6, 10]) df2.columns = ['지점번호', '지점명', '연평균기온', '최고기온', '최저기온', '상대습도(%)', '강수량(mm)'] df2 = df2.dropna(how='all') print(tabulate(df2, headers='keys', tablefmt='grid', showindex=False)) df2['연평균기온'] seasn_data = df2[df2['지점명'] == '서산'];seasn_data result_df2 = pd.DataFrame(df2) result_df2.to_csv('Desktop/bda 조별활동/기후데이터.csv', index=False, encoding='utf-8-sig')</pre>	<table><tr><th>지역</th><th>연평균기온</th><th>최고기온</th><th>최저기온</th><th>상대습도(%)</th><th>강수량(mm)</th></tr><tr><td>인천광역시 계양구</td><td>12.5</td><td>16.6</td><td>9.1</td><td>68.8</td><td>1207.4</td></tr><tr><td>인천광역시 남동구</td><td>12.5</td><td>16.6</td><td>9.1</td><td>68.8</td><td>1207.4</td></tr><tr><td>인천광역시 미추홀구</td><td>12.5</td><td>16.6</td><td>9.1</td><td>68.8</td><td>1207.4</td></tr><tr><td>인천광역시 부평구</td><td>12.1</td><td>18</td><td>7</td><td></td><td>1142.6</td></tr><tr><td>인천광역시 서구</td><td>12.5</td><td>16.6</td><td>9.1</td><td>68.8</td><td>1207.4</td></tr><tr><td>인천광역시 연수구</td><td>12.8</td><td>17.5</td><td>8.7</td><td></td><td>1085.4</td></tr><tr><td>인천광역시 옹진군</td><td>12.5</td><td>16.6</td><td>9.1</td><td>68.8</td><td>1207.4</td></tr><tr><td>인천광역시 중구</td><td>12.5</td><td>16.6</td><td>9.1</td><td>68.8</td><td>1207.4</td></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>						지역	연평균기온	최고기온	최저기온	상대습도(%)	강수량(mm)	인천광역시 계양구	12.5	16.6	9.1	68.8	1207.4	인천광역시 남동구	12.5	16.6	9.1	68.8	1207.4	인천광역시 미추홀구	12.5	16.6	9.1	68.8	1207.4	인천광역시 부평구	12.1	18	7		1142.6	인천광역시 서구	12.5	16.6	9.1	68.8	1207.4	인천광역시 연수구	12.8	17.5	8.7		1085.4	인천광역시 옹진군	12.5	16.6	9.1	68.8	1207.4	인천광역시 중구	12.5	16.6	9.1	68.8	1207.4						
지역	연평균기온	최고기온	최저기온	상대습도(%)	강수량(mm)																																																													
인천광역시 계양구	12.5	16.6	9.1	68.8	1207.4																																																													
인천광역시 남동구	12.5	16.6	9.1	68.8	1207.4																																																													
인천광역시 미추홀구	12.5	16.6	9.1	68.8	1207.4																																																													
인천광역시 부평구	12.1	18	7		1142.6																																																													
인천광역시 서구	12.5	16.6	9.1	68.8	1207.4																																																													
인천광역시 연수구	12.8	17.5	8.7		1085.4																																																													
인천광역시 옹진군	12.5	16.6	9.1	68.8	1207.4																																																													
인천광역시 중구	12.5	16.6	9.1	68.8	1207.4																																																													

4. 합본

- 모든 데이터들을 합치고, 결측치 처리를 위한 코딩 진행
- “밤”, “복분자딸기”, “오갈피”, “마”, “도라지”, “더덕”, “생표고”가 모두 0인 열은 제거
- 토양깊이유형, 토성코드, 토양형코드, 토양유효수분량, 기후관련 모든 데이터의 결측치를 지역별로 그룹화 하여 최빈값으로 결측치를 처리함

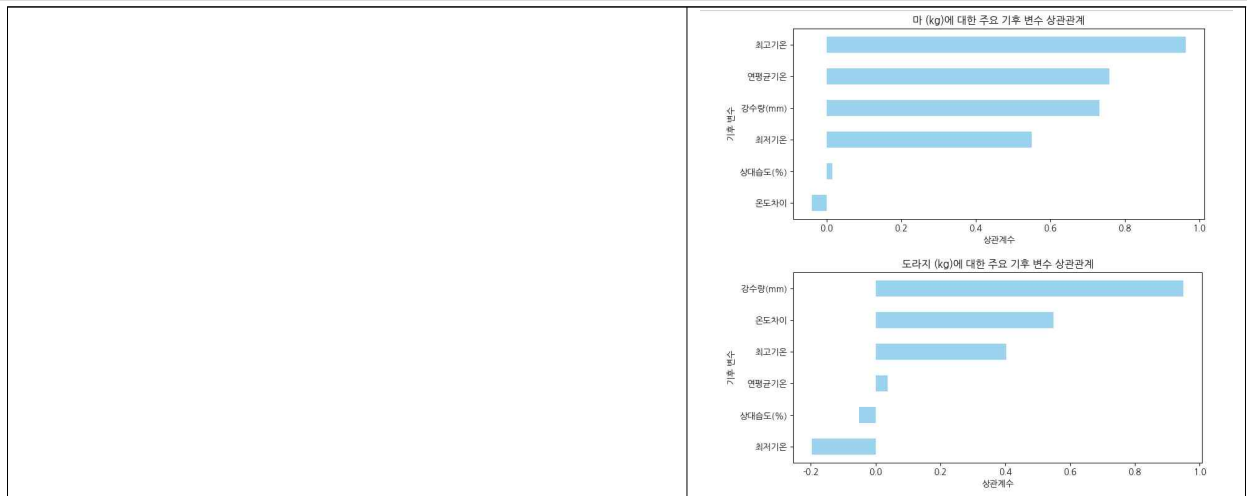
합본 결측치 제거	
코드	결과
<pre>import pandas as pd df1=pd.read_csv("C:/Users/user/Desktop/토양수분생산량기후.csv",encoding='euc-kr') df1[df1['밤 (kg)'].isna()&df1['복분자딸기 (kg)'].isna()&df1['오갈피 (kg)'].isna()] drop1=df1[df1[['밤 (kg)', '복분자딸기 (kg)', '오갈피 (kg)', '마 (kg)', '도라지 (kg)', '더덕 (kg)', '생표고 (kg)']].isna().all(axis=1)] df1 = df1.dropna(subset=['밤 (kg)', '복분자딸기 (kg)', '오갈피 (kg)', '마 (kg)', '도라지 (kg)', '더덕 (kg)', '생표고 (kg)'], how='all') df_copy=df1.copy() df_copy[['밤 (kg)', '복분자딸기 (kg)', '오갈피 (kg)', '마 (kg)', '도라지 (kg)', '더덕 (kg)', '생표고 (kg)']].fillna(0) road_regions = ['강원도', '경기도', '경상북도', '경상남도', '대구광역시', '부산광역시', '서울특별시', "세종특별자치시", "울산광역시", "인천광역시", "전라남도", "전라북도", "충청남도", "충청북도", "제주"] columns_to_impute = ['토양깊이유형', '토성코드', "토양형코드"] for col in columns_to_impute: df_copy[col] = df_copy.apply(lambda row: row[col] if pd.notnull(row[col]) else df_copy[df_copy['지역'].str.contains(next((region for region in broad_regions if region in row['지역']), ""), na=False)][col].mode().iloc[0] if not df_copy[df_copy['지역'].str.contains(next((region for region in broad_regions if region in row['지역']), ""), na=False)][col].mode().empty else row[col], axis=1) df_copy</pre>	

합본 그룹화 및 결측치 제거	
코드	결과
<pre>import pandas as pd data = [] df1=pd.read_csv('C:/Users/kh009/OneDrive/바탕화면/토양수분/merged_all_regions.csv', encoding='UTF-8-SIG') df2 = pd.read_csv('C:/Users/kh009/OneDrive/바탕 화면/토양수분/sangsanrrang.csv', encoding='EUC-KR') df3 = pd.read_csv('C:/Users/kh009/OneDrive/바탕 화면/토양수분/기후결합.csv', encoding='UTF-8-SIG') merged_1 = pd.merge(df1, df2, on='지역', how='outer') final_merged = pd.merge(merged_1, df3, on='지역', how='outer') print(final_merged) final_output_file = 'C:/Users/kh009/OneDrive/바탕 화면/토양수분/토양수분생산량기후.csv' final_merged.to_csv(final_output_file, index=False, encoding='utf-8-sig') print(f"Final merged file saved as: {final_output_file}")</pre>	

[코드 및 결과]

<코드 및 결과 1>

상관분석 시각화	
코드	결과
<pre> import pandas as pd import seaborn as sns import matplotlib.pyplot as plt import matplotlib import platform matplotlib.rcParams['font.sans-serif'] = ['NanumGothic'] matplotlib.rcParams['axes.unicode_minus'] = False file_path = "C:/Users/user/Desktop/진짜임 이계 썬.csv" data = pd.read_csv(file_path, encoding='euc-kr') data['온도차이'] = data['최고기온'] - data['최저기온'] crops = ["밤 (kg)", "복분자딸기 (kg)", "오갈피 (kg)", "더덕 (kg)", "생표고 (kg)", '마 (kg)', '도라지 (kg)'] independent_vars = ["연평균기온", "최고기온", "최저기온", "상대습도(%)", "강수량(mm)", "온도차이"] filtered_data = data[(data[crops] > 0).all(axis=1)].copy() variables = independent_vars + crops filtered_data[variables] = filtered_data[variables].fillna(filtered_data[variables].mean()) filtered_correlation_matrix = filtered_data[variables].corr() print("상관관계 행렬:") print(filtered_correlation_matrix) top_n = 6 main_climate_factors = {} for crop in crops: corr_series = filtered_correlation_matrix.loc[crop, independent_vars] sorted_corr = corr_series.abs().sort_values(ascending=False) top_vars = sorted_corr.head(top_n).index.tolist() # 실제 상관계수 값 저장 (원래 부호 유지) main_factors = corr_series[top_vars] main_climate_factors[crop] = main_factors for crop, factors in main_climate_factors.items(): print(f"{crop}에 영향을 미치는 주요 기후 변수 (상관계수 순위):") for rank, (var, corr) in enumerate(factors.items(), start=1): print(f" {rank}등: {var} (상관계수: {corr:.2f})") plt.figure(figsize=(12, 8)) sns.heatmap(filtered_correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f", linewidths=(0.5)) plt.title("상관 히트맵: 작물과 환경 변수 (필터링된 데이터)") plt.xticks(rotation=45, ha='right') plt.tight_layout() plt.show() for crop, factors in main_climate_factors.items(): plt.figure(figsize=(8, 4)) factors_sorted = factors.sort_values(ascending=True) # 막대그래프를 보기 좋게 정렬 factors_sorted.plot(kind='barh', color='skyblue') plt.title(f"{crop}에 대한 주요 기후 변수 상관관계") plt.xlabel("상관계수") plt.ylabel("기후 변수") plt.tight_layout() plt.show() </pre>	



<코드 및 결과2>

토양데이터	
코드	결과
<pre> import pandas as pd from scipy.stats import chi2_contingency, f_oneway file_path = 'C:/Users/user/Desktop/진짜임 이계 썬.csv' data = pd.read_csv(file_path, encoding='cp949') categorical_columns = ['토양깊이유형', '토성코드', '토양형코드', '토양유효수분량'] for col in categorical_columns: data[col] = data[col].astype('object') production_columns = ['밤 (kg)', '복분자딸기 (kg)', '오갈피 (kg)', '마 (kg)', '도라지 (kg)', '더덕 (kg)', '생표고 (kg)'] for col in production_columns: data[col] = pd.to_numeric(data[col], errors='coerce').fillna(0) def categorize_by_quantiles(data, production_col): filtered_data = data[data[production_col] != 0] quantiles = filtered_data[production_col].quantile([0.25, 0.5, 0.75]).values bins = [0, quantiles[0], quantiles[1], quantiles[2], filtered_data[production_col].max()+1] labels = ['Low', 'Medium-Low', 'Medium-High', 'High'] categorized = pd.cut(filtered_data[production_col], bins=bins, labels=labels, include_lowest=True) return categorized def perform_analysis_for_production(data, production_col, categorical_cols): filtered_data = data[data[production_col] != 0].copy() filtered_data[production_col + '_category'] = categorize_by_quantiles(filtered_data, production_col) chi2_results = {} for cat_col in categorical_cols: chi2_results[cat_col] = { 'p-value': chi2_contingency(pd.crosstab(filtered_data[cat_col], filtered_data[production_col + '_category']))[1] } anova_results = {} for cat_col in categorical_cols: anova_results[cat_col] = { 'p-value': f_oneway(*[group[production_col].values for name, group in filtered_data.groupby(cat_col)])[1] } chi2_results_df = pd.DataFrame.from_dict(chi2_results, orient='index', columns=['p-value']) anova_results_df = pd.DataFrame.from_dict(anova_results, orient='index', columns=['p-value']) return chi2_results_df, anova_results_df </pre>	<pre> ==== All Results ==== 밤 (kg) - Chi-Square Results: p-value 토양깊이유형 0.634062 토성코드 0.218194 토양형코드 0.687409 토양유효수분량 0.978081 밤 (kg) - ANOVA Results: p-value 토양깊이유형 0.878330 토성코드 0.070395 토양형코드 0.351903 토양유효수분량 0.532277 복분자딸기 (kg) - Chi-Square Results: p-value 토양깊이유형 0.354658 토성코드 0.519526 토양형코드 0.136203 토양유효수분량 0.283187 복분자딸기 (kg) - ANOVA Results: p-value 토양깊이유형 0.973020 토성코드 0.740754 토양형코드 0.554722 토양유효수분량 0.000004 오갈피 (kg) - Chi-Square Results: p-value 토양깊이유형 0.937710 토성코드 0.620558 토양형코드 0.390885 토양유효수분량 0.367888 오갈피 (kg) - ANOVA Results: p-value 토양깊이유형 0.811761 토성코드 0.436857 토양형코드 0.953100 토양유효수분량 0.096380 마 (kg) - Chi-Square Results: p-value 토양깊이유형 0.288619 토성코드 0.516913 토양형코드 0.733140 토양유효수분량 0.136004 마 (kg) - ANOVA Results: p-value 토양깊이유형 0.726274 토성코드 0.357121 토양형코드 0.392110 토양유효수분량 0.643374 도라지 (kg) - Chi-Square Results: </pre>

<pre> all_results = {} significant_results_cleaned = {} for production_col in production_columns: print(f"Performing analysis for {production_col}...\n") chi2_results, anova_results = perform_analysis_for_production(data, production_col, categorical_columns) all_results[production_col] = { 'Chi-Square': chi2_results, 'ANOVA': anova_results } chi2_significant = chi2_results[chi2_results['p-value'] < 0.1] anova_significant = anova_results[anova_results['p-value'] < 0.1] if not chi2_significant.empty or not anova_significant.empty: significant_results_cleaned[production_col] = { 'Chi-Square': chi2_significant, 'ANOVA': anova_significant } print("\n==== All Results =====") for production_col, results in all_results.items(): print(f"\n{production_col} - Chi-Square Results:") print(results['Chi-Square']) print(f"\n{production_col} - ANOVA Results:") print(results['ANOVA']) print("\n==== Significant Results (p-value < 0.1) =====") for production_col, results in significant_results_cleaned.items(): print(f"\n{production_col}:") if not results['Chi-Square'].empty: print("\nChi-Square Significant Results:") print(results['Chi-Square']) if not results['ANOVA'].empty: print("\nANOVA Significant Results:") print(results['ANOVA']) </pre>	<pre> p-value 토양깊이유형 0.607435 토성코드 0.641535 토양형코드 0.497741 토양유효수분량 0.953369 도라지 (kg) - ANOVA Results: p-value 토양깊이유형 0.869143 토성코드 0.774737 토양형코드 0.504751 토양유효수분량 0.124696 더덕 (kg) - Chi-Square Results: p-value 토양깊이유형 0.376734 토성코드 0.458934 토양형코드 0.532508 토양유효수분량 0.443737 더덕 (kg) - ANOVA Results: p-value 토양깊이유형 8.342797e-01 토성코드 1.865364e-02 토양형코드 2.463748e-01 토양유효수분량 3.074451e-08 생표고 (kg) - Chi-Square Results: p-value 토양깊이유형 0.557808 토성코드 0.132472 토양형코드 0.204570 토양유효수분량 0.026544 생표고 (kg) - ANOVA Results: p-value 토양깊이유형 0.524341 토성코드 0.369413 토양형코드 0.779517 토양유효수분량 0.010500 ==== Significant Results (p-value < 0.1) ===== 밤 (kg): ANOVA Significant Results: p-value 토성코드 0.070395 복분자딸기 (kg): ANOVA Significant Results: p-value 토양유효수분량 0.000004 오갈피 (kg): ANOVA Significant Results: p-value 토양유효수분량 0.09638 더덕 (kg): ANOVA Significant Results: p-value 토성코드 1.865364e-02 토양유효수분량 3.074451e-08 생표고 (kg): Chi-Square Significant Results: p-value 토양유효수분량 0.026544 ANOVA Significant Results: p-value 토양유효수분량 0.0105 </pre>
---	---

기후데이터	
코드	결과
<pre> import pandas as pd import seaborn as sns import matplotlib.pyplot as plt import matplotlib import platform matplotlib.rcParams['font.sans-serif'] = ['NanumGothic'] matplotlib.rcParams['axes.unicode_minus'] = False set_font() file_path = "C:/Users/woojoo00/Desktop/진짜임 이계 찐.csv" data = pd.read_csv(file_path, encoding='euc-kr') data['온도차이'] = data['최고기온'] - data['최저기온'] crops = ["밤 (kg)", "복분자딸기 (kg)", "오갈피 (kg)", "더덕 (kg)", "생표고 (kg)", "마 (kg)", "도라지 (kg)"] independent_vars = ["연평균기온", "최고기온", "최저기온", "상대습도(%)", "강수량(mm)", "온도차이"] filtered_data = data[(data[crops] > 0).all(axis=1)].copy() variables = independent_vars + crops filtered_data[variables] = filtered_data[variables].fillna(filtered_data[variables].mean()) filtered_correlation_matrix = filtered_data[variables].corr() print("상관관계 행렬:") print(filtered_correlation_matrix) threshold = 0.2 # 절댓값 기준 main_climate_factors = {} for crop in crops: corr_series = filtered_correlation_matrix.loc[crop, independent_vars] significant_corr = corr_series[abs(corr_series) >= threshold] sorted_corr = significant_corr.reindex(significant_corr.abs().sort_values(ascending=False).index) main_climate_factors[crop] = sorted_corr for crop, factors in main_climate_factors.items(): print(f"\n{crop}에 영향을 미치는 주요 기후 변수 (상관계수 절댓값 >= {threshold} 기준):") if not factors.empty: for rank, (var, corr) in enumerate(factors.items(), start=1): print(f" {rank}등: {var} (상관계수: {corr:.2f})") else: print(" 유의미한 기후 변수가 없습니다.") mask = filtered_correlation_matrix.abs() >= threshold plt.figure(figsize=(12, 10)) sns.heatmap(filtered_correlation_matrix.where(mask), annot=True, cmap="coolwarm", fmt=".2f", linewidths=0.5, cbar=True) plt.title("상관 히트맵: 작물과 환경 변수 (상관계수 절댓값 >= 0.2)") plt.xticks(rotation=45, ha='right') plt.yticks(rotation=0) plt.tight_layout() plt.show() for crop, factors in main_climate_factors.items(): if not factors.empty: plt.figure(figsize=(8, 4)) factors_sorted = factors.reindex(factors.abs().sort_values(ascending=True).index) factors_sorted.plot(kind='barh', color='skyblue') plt.title(f"{crop}에 대한 주요 기후 변수 상관관계") plt.xlabel("상관계수") plt.ylabel("기후 변수") plt.tight_layout() plt.show() else: print(f"\n{crop}에 대해 상관관계 절댓값이 {threshold} 이상인 기후 변수가 없어 그래프를 생성하지 않습니다.") </pre>	<p>밤 (kg)에 영향을 미치는 주요 기후 변수 (상관계수 절댓값 >= 0.2 기준):</p> <ol style="list-style-type: none"> 1등: 최고기온 (상관계수: 0.92) 2등: 연평균기온 (상관계수: 0.78) 3등: 강수량(mm) (상관계수: 0.75) 4등: 최저기온 (상관계수: 0.60) 5등: 상대습도(%) (상관계수: 0.24) <p>복분자딸기 (kg)에 영향을 미치는 주요 기후 변수 (상관계수 절댓값 >= 0.2 기준):</p> <ol style="list-style-type: none"> 1등: 최고기온 (상관계수: 0.71) 2등: 강수량(mm) (상관계수: 0.71) 3등: 상대습도(%) (상관계수: 0.63) 4등: 연평균기온 (상관계수: 0.54) 5등: 최저기온 (상관계수: 0.41) <p>오갈피 (kg)에 영향을 미치는 주요 기후 변수 (상관계수 절댓값 >= 0.2 기준):</p> <ol style="list-style-type: none"> 1등: 연평균기온 (상관계수: -0.83) 2등: 최저기온 (상관계수: -0.83) 3등: 온도차이 (상관계수: 0.68) 4등: 최고기온 (상관계수: -0.58) <p>더덕 (kg)에 영향을 미치는 주요 기후 변수 (상관계수 절댓값 >= 0.2 기준):</p> <ol style="list-style-type: none"> 1등: 최저기온 (상관계수: -0.96) 2등: 연평균기온 (상관계수: -0.93) 3등: 온도차이 (상관계수: 0.76) 4등: 최고기온 (상관계수: -0.71) 5등: 상대습도(%) (상관계수: -0.39) <p>생표고 (kg)에 영향을 미치는 주요 기후 변수 (상관계수 절댓값 >= 0.2 기준):</p> <ol style="list-style-type: none"> 1등: 강수량(mm) (상관계수: -0.40) 2등: 상대습도(%) (상관계수: -0.28) 3등: 연평균기온 (상관계수: -0.25) 4등: 최저기온 (상관계수: -0.20) <p>마 (kg)에 영향을 미치는 주요 기후 변수 (상관계수 절댓값 >= 0.2 기준):</p> <ol style="list-style-type: none"> 1등: 최고기온 (상관계수: 0.96) 2등: 연평균기온 (상관계수: 0.76) 3등: 강수량(mm) (상관계수: 0.73) 4등: 최저기온 (상관계수: 0.55) <p>도라지 (kg)에 영향을 미치는 주요 기후 변수 (상관계수 절댓값 >= 0.2 기준):</p> <ol style="list-style-type: none"> 1등: 강수량(mm) (상관계수: 0.95) 2등: 온도차이 (상관계수: 0.55) 3등: 최고기온 (상관계수: 0.40)

<코드 및 결과 3>

SVM	
코드	결과
<pre> from sklearn.svm import SVC from sklearn.model_selection import train_test_split from sklearn.preprocessing import LabelEncoder import numpy as np import pandas as pd file_path = "C:/Users/user/Desktop/진짜임 이계 찐.csv" data = pd.read_csv(file_path, encoding='euc-kr') </pre>	<p>밤 - 주요 변수 중요도:</p> <p>토성코드_1: 0.5763146268864068 토성코드_2: 0.2873379003105292 토성코드_3: -0.4152795908181084 토성코드_4: -0.4483729363788458 연평균기온: -0.20878056847918636 최고기온: 0.38757820931001774 강수량(mm): 0.0022087768804794905</p> <p>복분자딸기 - 주요 변수 중요도:</p> <p>토양유해수분량_1: 0.0 토양유해수분량_2: 0.05578239439902832</p>

```

data['온도차이']=data['최고기온']-data['최저기온']
data = pd.get_dummies(data, columns=['토성코드', '토양깊이유형','토양유효수분량'])

feature_sets = {
    '밤': ['토성코드_1','토성코드_2','토성코드_3','토성코드_4','연평균기온', '최고기온', '강수량(mm)',
    '북분자팔기': ['토양유효수분량_1','토양유효수분량_2','토양유효수분량_3','토양유효수분량_4','토양유효수분량_5', '최고기온', '상대습도(%)', '강수량(mm)', '오갈피: ['토양유효수분량_1','토양유효수분량_2','토양유효수분량_3','토양유효수분량_4','토양유효수분량_5', '연평균기온', '최고기온', '최저기온', '상대습도(%)', '강수량(mm)', '온도차이'], '더덕': ['토성코드_1','토성코드_2','토성코드_3','토성코드_4','토양유효수분량_1','토양유효수분량_2','토양유효수분량_3','토양유효수분량_4','토양유효수분량_5', '연평균기온', '최고기온', '최저기온', '상대습도(%)', '강수량(mm)', '온도차이'], '생표고: ['토양유효수분량_2','토양유효수분량_3','토양유효수분량_4', '연평균기온', '최고기온', '최저기온', '강수량(mm)', '온도차이','최고기온']

}

data_cleaned = data.dropna()

def convert_to_binary_by_mean(y):
    mean_value = np.mean(y)
    return np.where(y >= mean_value, 1, 0)

def get_important_features_for_classification(data, target_col, features):
    X = data[features]
    y = data[target_col]

    y_binary = convert_to_binary_by_mean(y)

    if len(np.unique(y_binary)) == 1:
        print("경고: (target_col)에서 클래스가 하나만 존재합니다. 분류를 수행할 수 없습니다.")
        return None, None

    le = LabelEncoder()
    for col in X.select_dtypes(include=['object']).columns:
        X[col] = le.fit_transform(X[col])

    X_train, X_test, y_train, y_test = train_test_split(X, y_binary, test_size=0.2, random_state=42)

    model = SVC(kernel='linear')
    model.fit(X_train, y_train)

    feature_importance = model.coef_

    return feature_importance, features

for item, features in feature_sets.items():
    target_col = item + ' (kg)'
    feature_importance, features = get_important_features_for_classification(data_cleaned, target_col, features)

    if feature_importance is not None:
        print(f'{item} - 주요 변수 중요도:')
        for feature, importance in zip(features, feature_importance[0]):
            print(f'{feature}: {importance}')
        print()

<성능확인코드>

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import roc_auc_score, roc_curve
import matplotlib.pyplot as plt

file_path = 'C:/Users/user/Desktop/진짜임 이계 썬.csv'
data = pd.read_csv(file_path, encoding='euc-kr')
data.columns = data.columns.str.replace(r' W(\^)*W)', '', regex=True).str.strip()
data['온도차이'] = data['최고기온'] - data['최저기온']
data['도라지'] = data['도라지'].astype(int)

data = pd.get_dummies(data, columns=['토성코드', '토양깊이유형','토양유효수분량'])
feature_sets = {
    '밤': ['토성코드_1','토성코드_2','토성코드_3','토성코드_4','연평균기온', '최고기온', '강수량(mm)',
    '북분자팔기': ['토양유효수분량_1','토양유효수분량_2','토양유효수분량_3','토양유효수분량_4','토양유효수분량_5', '최고기온', '상대습도(%)', '강수량(mm)', '오갈피: ['토양유효수분량_1','토양유효수분량_2','토양유효수분량_3','토양유효수분량_4','토양유효수분량_5', '연평균기온', '최고기온', '최저기온', '상대습도(%)', '강수량(mm)', '온도차이'], '더덕': ['토성코드_1','토성코드_2','토성코드_3','토성코드_4','토양유효수분량_1','토양유효수분량_2','토양유효수분량_3','토양유효수분량_4','토양유효수분량_5', '연평균기온', '최고기온', '최저기온', '상대습도(%)', '강수량(mm)', '온도차이'], '생표고: ['토양유효수분량_2','토양유효수분량_3','토양유효수분량_4', '연평균기온', '최고기온', '최저기온', '강수량(mm)', '온도차이','최고기온']

}

def convert_to_binary_by_mean(y):
    mean_value = np.mean(y)
    return np.where(y >= mean_value, 1, 0)

def process_and_evaluate(data, product, features):
    target = convert_to_binary_by_mean(data[product])
    X = data[features]

```

토양유효수분량_3: -0.011107396351961007
 토양유효수분량_4: -0.04467499804711972
 토양유효수분량_5: 0.0
 최고기온: 0.09824030132670103
 상대습도(%): 0.0032675237866328644
 강수량(mm): -1.6094401871669106e-05

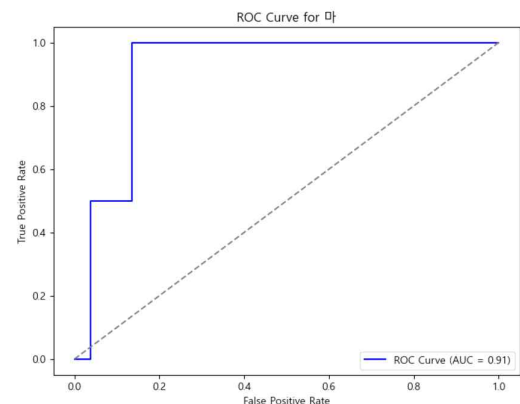
오갈피 - 주요 변수 중요도:
 토양유효수분량_1: 0.0
 토양유효수분량_2: 0.09443228072157606
 토양유효수분량_3: 0.18085921049787146
 토양유효수분량_4: 0.29432046963013825
 토양유효수분량_5: -0.5696119608496311
 연평균기온: -0.31821451218083396
 최고기온: 0.01161425416747619
 최저기온: 0.001250979315607026
 상대습도(%): -0.05465284916232349
 강수량(mm): 0.0007852369278680271
 온도차이: 0.010363274851822979

더덕 - 주요 변수 중요도:
 토성코드_1: -0.09128948088014699
 토성코드_2: 0.1392142959776399
 토성코드_3: -0.008135695096416207
 토성코드_4: -0.039789120001019695
 토양유효수분량_1: 0.11445620178256655
 토양유효수분량_2: -0.01993233637902254
 토양유효수분량_3: -0.08228485402320351
 토양유효수분량_4: -0.012239011380283493
 토양유효수분량_5: 0.0
 연평균기온: -0.14237441562566033
 최고기온: 0.024259693278779082
 최저기온: 0.07491782445855932
 상대습도(%): -0.016797184911254703
 강수량(mm): 0.0002972822114770679
 온도차이: -0.05065813117976781

생표고 - 주요 변수 중요도:
 토양유효수분량_2: 0.4239095453775228
 토양유효수분량_3: -0.12919326667957684
 토양유효수분량_4: 0.09004829847728324
 연평균기온: 0.26907487436341526
 최고기온: -0.09467007087956958
 최저기온: -0.20278730372764375
 강수량(mm): 0.0001741913604291767

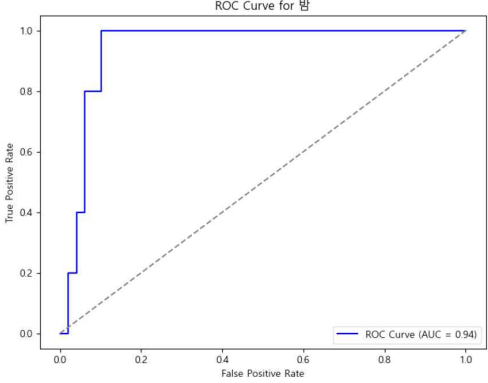
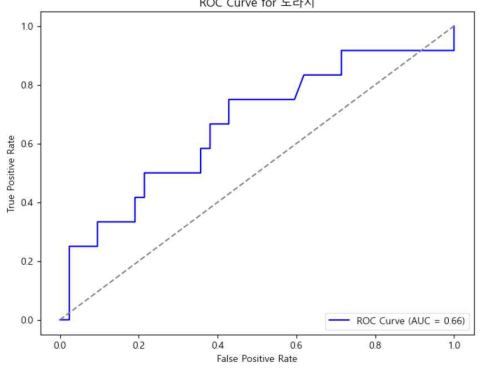
마 - 주요 변수 중요도:
 연평균기온: -0.0005724818957695277
 최고기온: 0.0003924961184011977
 최저기온: 0.00028507324983184645

도라지 - 주요 변수 중요도:
 강수량(mm): 0.0002764570481303963
 온도차이: 0.14833621943875386
 최고기온: -0.10871925869997767



<pre> scaler = StandardScaler() X_scaled = scaler.fit_transform(X) X_train, X_test, y_train, y_test = train_test_split(X_scaled, target, test_size=0.3, random_state=42) model = SVC(kernel='linear', probability=True, random_state=42) model.fit(X_train, y_train) y_pred_prob = model.predict_proba(X_test)[:, 1] auc = roc_auc_score(y_test, y_pred_prob) fpr, tpr, _ = roc_curve(y_test, y_pred_prob) plt.rcParams['font.family'] = 'Malgun Gothic' plt.rcParams['axes.unicode_minus'] = False plt.figure(figsize=(8, 6)) plt.plot(fpr, tpr, color='blue', label=f'ROC Curve (AUC = {auc:.2f})') plt.plot([0, 1], [0, 1], color='gray', linestyle='--') plt.xlabel('False Positive Rate') plt.ylabel('True Positive Rate') plt.title('ROC Curve for (product)') plt.legend(loc='lower right') plt.show() return auc auc_scores = {} for product, features in feature_sets.items(): auc = process_and_evaluate(data, product, features) auc_scores[product] = auc auc_scores </pre>	
---	--

로지스틱회귀	
코드	결과
<pre> import pandas as pd import numpy as np from sklearn.model_selection import train_test_split from sklearn.linear_model import LogisticRegression from sklearn.preprocessing import StandardScaler, LabelEncoder file_path = 'C:/Users/user/Desktop/진짜임 이계 켜.csv' data = pd.read_csv(file_path, encoding='euc-kr') data.columns = data.columns.str.replace(r' \[(^)*\]*', '', regex=True).str.strip() data['온도차이'] = data['최고기온'] - data['최저기온'] data['도라지'] = data['도라지'].astype(int) data = pd.get_dummies(data, columns=['토성코드', '토양깊이유형','토양유효수분량']) columns_by_product = { '밤': ['토성코드_1','토성코드_2','토성코드_3','토성코드_4','연평균기온','최고기온','강수량(mm)'], '복분자딸기': ['토양유효수분량_1','토양유효수분량_2','토양유효수분량_3','토양유효수분량_4','토양유효수분량_5','최고기온','상대습도(%)','강수량(mm)'], '오갈피': ['토양유효수분량_1','토양유효수분량_2','토양유효수분량_3','토양유효수분량_4','토양유효수분량_5','연평균기온','최고기온','최저기온','상대습도(%)','강수량(mm)','온도차이','더덕': ['토성코드_1','토성코드_2','토성코드_3','토성코드_4','토양유효수분량_1','토양유효수분량_2','토양유효수분량_3','토양유효수분량_4','토양유효수분량_5','연평균기온','최고기온','최저기온','상대습도(%)','강수량(mm)','온도차이','생표고': ['토양유효수분량_2','토양유효수분량_3','토양유효수분량_4','연평균기온','최고기온','최저기온','강수량(mm)'],'마': ['연평균기온','최고기온','최저기온','도라지','강수량(mm)','온도차이','최고기온'] } def convert_to_binary_by_mean(y): mean_value = np.mean(y) return np.where(y >= mean_value, 1, 0) def process_and_evaluate(data, product, features): target = convert_to_binary_by_mean(data[product]) X = data[features] scaler = StandardScaler() X_scaled = scaler.fit_transform(X) X_train, X_test, y_train, y_test = train_test_split(X_scaled, target, test_size=0.3, random_state=42) model = LogisticRegression(max_iter=1000) model.fit(X_train, y_train) feature_importance = model.coef_[0] return feature_importance feature_importances = {} for product, features in columns_by_product.items(): importance = process_and_evaluate(data, product, features) feature_importances[product] = importance for product, importance in feature_importances.items(): print(f'{product} 변수 중요도:') for feature, imp in zip(columns_by_product[product], importance): print(f' {feature}: {imp}')</pre>	<p>밤 변수 중요도:</p> <p>토성코드_1: 0.6422071867030641 토성코드_2: 0.10488056192062174 토성코드_3: -0.8219595259409562 토성코드_4: -0.632619163905879 연평균기온: -0.756872144026954 최고기온: 1.0851595164217822 강수량(mm): 1.0646583312746443</p> <p>복분자딸기 변수 중요도:</p> <p>토양유효수분량_1: 0.20954852761407966 토양유효수분량_2: 0.16020037875529464 토양유효수분량_3: 0.07383568509784506 토양유효수분량_4: -0.30217537682755985 토양유효수분량_5: 0.08197729542577069 최고기온: 0.3583540415324528 상대습도(%) : 0.7874233963732282 강수량(mm): 0.25106504668345236</p> <p>오갈피 변수 중요도:</p> <p>토양유효수분량_1: -0.13302218693504614 토양유효수분량_2: -0.13107450653704153 토양유효수분량_3: 0.059660048318121627 토양유효수분량_4: 0.23851773102406965 토양유효수분량_5: -0.3154004293561679 연평균기온: -0.670139003009114 최고기온: -0.2358737087016923 최저기온: -0.3143321443004134 상대습도(%) : -0.7007677373935917 강수량(mm): 0.5446265237190544 온도차이: 0.20411254814816954</p> <p>더덕 변수 중요도:</p> <p>토성코드_1: -0.16016034285132777 토성코드_2: 0.13522974770847638 토성코드_3: 0.18277320231104177 토성코드_4: -0.22212604415653106 토양유효수분량_1: 0.46904771394235556 토양유효수분량_2: -0.5906719396690789 토양유효수분량_3: -0.1503671601188886 토양유효수분량_4: 0.30944797583940725 토양유효수분량_5: 0.44518378518644053 연평균기온: -0.43634870585715724 최고기온: -0.3128784156288547 최저기온: 0.04955330866097385 상대습도(%) : 0.4264290268762058 강수량(mm): 0.004640412784972791 온도차이: -0.24845285380312532</p> <p>생표고 변수 중요도:</p> <p>토양유효수분량_2: 0.8738103966272557 토양유효수분량_3: -0.7617902708968525 토양유효수분량_4: 0.623438056987029 연평균기온: -0.18217611907358316 최고기온: 0.005499051559875496 최저기온: -0.164641773276043 강수량(mm): -0.3030259599260499</p> <p>마 변수 중요도:</p> <p>연평균기온: -0.10427616302621683 최고기온: 0.7336057878619041 최저기온: -0.6396770218527942</p> <p>도라지 변수 중요도:</p> <p>강수량(mm): 0.25450466275350053 온도차이: 0.4278946361665503 최고기온: -0.3241341178391088</p>
<성능확인 코드>	

<pre> import pandas as pd import numpy as np from sklearn.model_selection import train_test_split from sklearn.linear_model import LogisticRegression from sklearn.preprocessing import StandardScaler from sklearn.metrics import roc_auc_score, roc_curve import matplotlib.pyplot as plt file_path = 'C:/Users/user/Desktop/진짜임 이계 썬.csv' data = pd.read_csv(file_path, encoding='euc-kr') data.columns = data.columns.str.replace(r' \W{1,4} ', '', regex=True).str.strip() data['온도차이'] = data['최고기온'] - data['최저기온'] data['도라지'] = data['도라지'].astype(int) data = pd.get_dummies(data, columns=['토성코드', '토양깊이유형', '토양유효수분량']) columns_by_product = { '밤': ['토성코드_1', '토성코드_2', '토성코드_3', '토성코드_4', '연평균기온', '최고기온', '강수량(mm)', '복분자딸기': ['토양유효수분량_1', '토양유효수분량_2', '토양유효수분량_3', '토양유효수분량_4', '토양유효수분량_5', '최고기온', '상대습도(%)', '강수량(mm)', '오갈피': ['토양유효수분량_1', '토양유효수분량_2', '토양유효수분량_3', '토양유효수분량_4', '토양유효수분량_5', '연평균기온', '최고기온', '최저기온', '상대습도(%)', '강수량(mm)', '온도차이'], '생표고': ['토양유효수분량_2', '토양유효수분량_3', '토양유효수분량_4', '연평균기온', '최고기온', '최저기온', '강수량(mm)'], '마': ['연평균기온', '최고기온', '최저기온', '도라지', '강수량(mm)', '온도차이', '최고기온'] } def convert_to_binary_by_mean(y): mean_value = np.mean(y) return np.where(y >= mean_value, 1, 0) def process_and_evaluate(data, product, features): target = convert_to_binary_by_mean(data[product]) X = data[features] scaler = StandardScaler() X_scaled = scaler.fit_transform(X) X_train, X_test, y_train, y_test = train_test_split(X_scaled, target, test_size=0.3, random_state=42) model = LogisticRegression(max_iter=1000) model.fit(X_train, y_train) y_pred_prob = model.predict_proba(X_test)[:, 1] auc = roc_auc_score(y_test, y_pred_prob) fpr, tpr, _ = roc_curve(y_test, y_pred_prob) plt.rcParams['font.family'] = 'Malgun Gothic' plt.rcParams['axes.unicode_minus'] = False plt.figure(figsize=(8, 6)) plt.plot(fpr, tpr, color='blue', label=f'ROC Curve (AUC = {auc:.2f})') plt.plot([0, 1], [0, 1], color='gray', linestyle='--') plt.xlabel('False Positive Rate') plt.ylabel('True Positive Rate') plt.title(f'ROC Curve for {product}') plt.legend(loc='lower right') plt.show() return auc auc_scores = {} for product, features in columns_by_product.items(): auc = process_and_evaluate(data, product, features) auc_scores[product] = auc auc_scores </pre>	 <p>ROC Curve for 밤</p> <p>True Positive Rate vs False Positive Rate</p> <p>ROC Curve (AUC = 0.94)</p>  <p>ROC Curve for 도라지</p> <p>True Positive Rate vs False Positive Rate</p> <p>ROC Curve (AUC = 0.66)</p>
--	--

랜덤포레스트	
코드	결과
<pre> import pandas as pd import numpy as np from sklearn.model_selection import train_test_split from sklearn.ensemble import RandomForestClassifier from sklearn.preprocessing import StandardScaler file_path = 'C:/Users/user/Desktop/진짜임 이계 썬.csv' data = pd.read_csv(file_path, encoding='euc-kr') data.columns = data.columns.str.replace(r' \W{1,4} ', '', regex=True).str.strip() data = pd.get_dummies(data, columns=['토성코드', '토양깊이유형', '토양유효수분량']) data['온도차이'] = data['최고기온'] - data['최저기온'] data['도라지'] = data['도라지'].astype(int) columns_by_product = { '밤': ['토성코드_1', '토성코드_2', '토성코드_3', '토성코드_4', '연평균기온', '최고기온', '강수량(mm)', </pre>	<p>밤 변수 중요도:</p> <p>토성코드_1: 0.03255063765697429 토성코드_2: 0.05493609287700822 토성코드_3: 0.017343861162734286 토성코드_4: 0.0062363614139309695 연평균기온: 0.24782207775329607 최고기온: 0.23855452378444506 강수량(mm): 0.402556445351611</p> <p>복분자딸기 변수 중요도:</p> <p>토양유효수분량_1: 0.029126679506330674 토양유효수분량_2: 0.02001582456256473 토양유효수분량_3: 0.049649173854513215 토양유효수분량_4: 0.03768246047825507 토양유효수분량_5: 0.009739324722952969 최고기온: 0.2639553588710774 상대습도(%): 0.27022268981419245 강수량(mm): 0.31960848819011345</p> <p>오갈피 변수 중요도:</p> <p>토양유효수분량_1: 0.0001568509412665263 토양유효수분량_2: 0.0071238431856904295 토양유효수분량_3: 0.013499994895227106 토양유효수분량_4: 0.014767288170851546</p>

```

'복분자딸기': ['토양유효수분량_1','토양유효수분량_2','토양유효수분량_3','토양유효수분량_4','토양유효수
분량_5', '최고기온', '상대습도(%)', '강수량(mm)', '오갈피': ['토양유효수분량_1','토양유효수분량_2','토양
유효수분량_3','토양유효수분량_4','토양유효수분량_5', '연평균기온', '최고기온', '최저기온', '상대습도(%)',
'강수량(mm)', '온도차이', '더덕': ['토성코드_1','토성코드_2','토성코드_3','토성코드_4','토양유효수분량_1',
토양유효수분량_2','토양유효수분량_3','토양유효수분량_4','토양유효수분량_5', '연평균기온', '최고기온', '
최저기온', '상대습도(%)', '강수량(mm)', '온도차이','생표고': ['토양유효수분량_2','토양유효수분량_3','토
양유효수분량_4', '연평균기온', '최고기온', '최저기온', '강수량(mm)'],'마': ['연평균기온', '최고기온', '최저
기온'],'도라지':['강수량(mm)','온도차이','최고기온']
}

def convert_to_binary_by_mean(y):
    mean_value = np.mean(y)
    return np.where(y >= mean_value, 1, 0)

def process_and_evaluate(data, product, features):
    target = convert_to_binary_by_mean(data[product])
    X = data[features]
    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X)
    X_train, X_test, y_train, y_test = train_test_split(X_scaled, target, test_size=0.3, random_state=42)
    model = RandomForestClassifier(random_state=42)
    model.fit(X_train, y_train)
    feature_importance = model.feature_importances_
    return feature_importance

feature_importances = {}

for product, features in columns_by_product.items():
    importance = process_and_evaluate(data, product, features)
    feature_importances[product] = importance

for product, importance in feature_importances.items():
    print(f'{product} 변수 중요도:')
    for feature, imp in zip(columns_by_product[product], importance):
        print(f'    {feature}: {imp}')

<성능확인 코드>

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import roc_auc_score, roc_curve
import matplotlib.pyplot as plt

file_path = 'C:/Users/user/Desktop/진짜임 이계 썬.csv'
data = pd.read_csv(file_path, encoding='euc-kr')

data.columns = data.columns.str.replace(r' ₩([^\s]*)', '', regex=True).str.strip()

data['온도차이'] = data['최고기온'] - data['최저기온']
data['도라지'] = data['도라지'].astype(int)

data = pd.get_dummies(data, columns=['토성코드', '토양깊이유형','토양유효수분량'])

columns_by_product = {
    '밤': ['토성코드_1','토성코드_2','토성코드_3','토성코드_4','연평균기온', '최고기온', '강수량(mm)',
    '복분자딸기': ['토양유효수분량_1','토양유효수분량_2','토양유효수분량_3','토양유효수분량_4','토양유효수
분량_5', '최고기온', '상대습도(%)', '강수량(mm)', '오갈피': ['토양유효수분량_1','토양유효수분량_2','토양
유효수분량_3','토양유효수분량_4','토양유효수분량_5', '연평균기온', '최고기온', '최저기온', '상대습도(%)',
'강수량(mm)', '온도차이', '더덕': ['토양깊이유형_10','토양깊이유형_18','토양깊이유형_20','토양깊이유형
_30','토성코드_1','토성코드_2','토성코드_3','토성코드_4','토양유효수분량_1','토양유효수분량_2','토양유효
수분량_3','토양유효수분량_4','토양유효수분량_5', '연평균기온', '최고기온', '최저기온', '상대습도(%)',
'강수량(mm)', '온도차이','생표고': ['토양유효수분량_2','토양유효수분량_3','토양유효수분량_4', '연평균기
온', '최고기온', '최저기온', '강수량(mm)'],'마': ['연평균기온', '최고기온', '최저기온',
'도라지':['강수량(mm)','온도차이','최고기온']
}

def convert_to_binary_by_mean(y):
    mean_value = np.mean(y)
    return np.where(y >= mean_value, 1, 0)

def process_and_evaluate(data, product, features):
    target = convert_to_binary_by_mean(data[product])
    X = data[features]
    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X)
    X_train, X_test, y_train, y_test = train_test_split(X_scaled, target, test_size=0.3, random_state=42)
    model = RandomForestClassifier(random_state=42)
    model.fit(X_train, y_train)
    y_pred_prob = model.predict_proba(X_test[:, 1])
    auc = roc_auc_score(y_test, y_pred_prob)
    fpr, tpr, _ = roc_curve(y_test, y_pred_prob)
    plt.rcParams['font.family'] = 'Malgun Gothic'
    plt.rcParams['axes.unicode_minus'] = False
    plt.figure(figsize=(8, 6))
    plt.plot(fpr, tpr, color='blue', label=f'ROC Curve (AUC = {auc:.2f})')
    plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
    plt.xlabel('False Positive Rate')

```

토양유효수분량_5: 0.0010575362557897236
연평균기온: 0.19323426634428414
최고기온: 0.1263052994096206
최저기온: 0.2274131711876021
상대습도(%): 0.11579185193989916
강수량(mm): 0.15527129636611223
온도차이: 0.1453786013036564

더덕 변수 중요도:

토성코드_1: 0.008085586809663509
토성코드_2: 0.016738468728840543
토성코드_3: 0.006102716332241641
토성코드_4: 0.00030654287268478867
토양유효수분량_1: 0.027793706527307414
토양유효수분량_2: 0.01446174920947405
토양유효수분량_3: 0.0302098824581595
토양유효수분량_4: 0.01860198520284851
토양유효수분량_5: 0.03729052046459183
연평균기온: 0.20681568983566456
최고기온: 0.1040419859077218
최저기온: 0.12508360356357004
상대습도(%): 0.09894173653008088
강수량(mm): 0.1280377511557274
온도차이: 0.17360694214721387

생표고 변수 중요도:

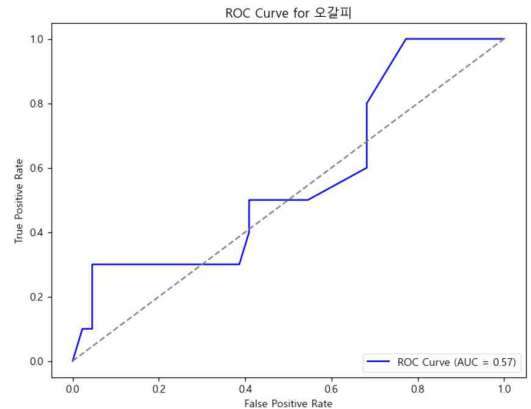
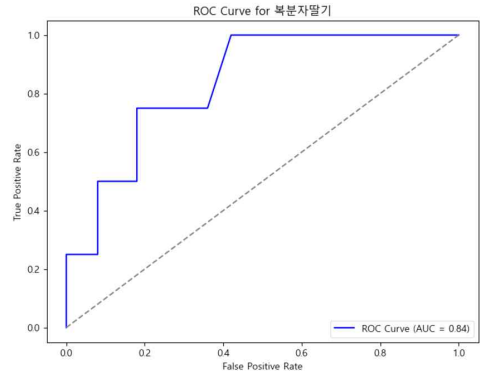
토양유효수분량_2: 0.07357439180615187
토양유효수분량_3: 0.058666339384862334
토양유효수분량_4: 0.02171360691451352
연평균기온: 0.15903222182250104
최고기온: 0.23027730879362168
최저기온: 0.20165634943277916
강수량(mm): 0.2550797818455704

마 변수 중요도:

연평균기온: 0.3062764310100684
최고기온: 0.3440403968610477
최저기온: 0.3496831721288839

도라지 변수 중요도:

강수량(mm): 0.3410267265077445
온도차이: 0.3426551204582461
최고기온: 0.3163181530340093



```

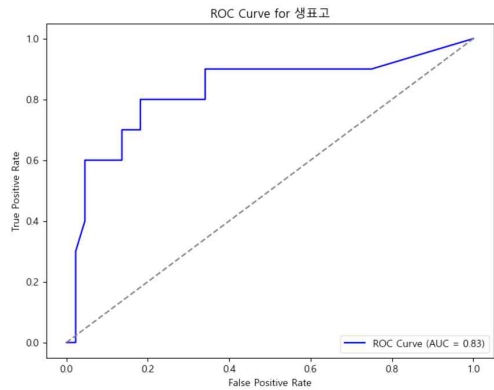
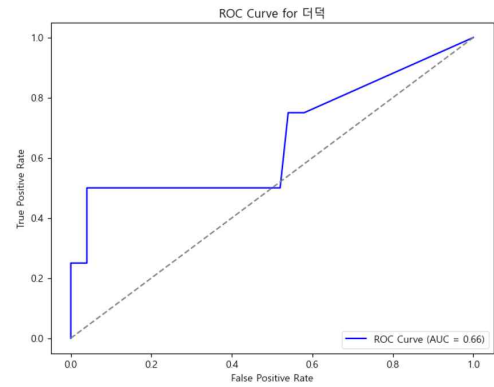
plt.ylabel('True Positive Rate')
plt.title('ROC Curve for {product}')
plt.legend(loc='lower right')
plt.show()
return auc

auc_scores = {}

for product, features in columns_by_product.items():
    auc = process_and_evaluate(data, product, features)
    auc_scores[product] = auc

auc_scores

```



<코드 및 결과4>

지역추출	
코드	결과(추출된 지역)
<pre> import pandas as pd file_path = r"C:/Users/user/Desktop/기후데이터_온도차이_범주화_완.csv" climate_data = pd.read_csv(file_path, encoding='cp949') filtered_data = climate_data[(climate_data['토성코드'] == 1) & (climate_data['최고기온_숫자'] == 3) & (climate_data['강수량(mm)_숫자'] == 3)] print('밤:', filtered_data['지역']) import pandas as pd file_path = r"C:/Users/user/Desktop/기후데이터_온도차이_범주화_완.csv" climate_data = pd.read_csv(file_path, encoding='cp949') filtered_data = climate_data[(climate_data['강수량(mm)_숫자'] == 2) & (climate_data['토양유효수분량'] == 3) & (climate_data['상대습도(%)_숫자'] == 4)] print('복분자딸기:', filtered_data['지역']) import pandas as pd file_path = r"C:/Users/user/Desktop/기후데이터_온도차이_범주화_완.csv" climate_data = pd.read_csv(file_path, encoding='cp949') filtered_data = climate_data[(climate_data['연평균기온_숫자'] == 2) & (climate_data['토양유효수분량'] == 4) & (climate_data['최저기온_숫자'] == 2)] print('오갈피:', filtered_data['지역']) import pandas as pd file_path = r"C:/Users/user/Desktop/기후데이터_온도차이_범주화_완.csv" filtered_data = climate_data[(climate_data['강수량(mm)_숫자'] == 3) & (climate_data['토양유효수분량'] == 2) & (climate_data['최고기온_숫자'] == 3)] print('생표고:', filtered_data['지역']) import pandas as pd file_path = r"C:/Users/user/Desktop/기후데이터_온도차이_범주화_완.csv" filtered_data = climate_data[(climate_data['연평균기온_숫자'] == 4) & (climate_data['토양유효수분량'] == 5) & (climate_data['온도차이_숫자'] == 4)] print('더덕:', filtered_data['지역']) import pandas as pd file_path = r"C:/Users/user/Desktop/기후데이터_온도차이_범주화_완.csv" filtered_data = climate_data[(climate_data['강수량(mm)_숫자'] == 2) & </pre>	<p>밤:</p> <ul style="list-style-type: none"> 44 경상남도 거창군 91 대전광역시 대덕구 94 대전광역시 유성구 138 전라북도 남원시 157 충청남도 부여군 <p>복분자딸기:</p> <ul style="list-style-type: none"> 105 인천광역시 강화군 108 인천광역시 미추홀구 110 인천광역시 서구 111 인천광역시 옹진군 112 인천광역시 중구 153 충청남도 금산군 <p>오갈피:</p> <ul style="list-style-type: none"> 8 강원도 원주 20 경기도 과천시 41 경기도 하남시 63 경상북도 고령군 65 경상북도 군위 66 경상북도 김천시 67 경상북도 문경시 69 경상북도 상주시 71 경상북도 안동시 77 경상북도 울릉군 78 경상북도 울진 122 전라남도보성군 154 충청남도 논산시 155 충청남도 당진시 157 충청남도 부여군 158 충청남도 서산시 161 충청남도 예산군 162 충청남도 천안시 165 충청남도 홍성군 <p>생표고:</p> <ul style="list-style-type: none"> 91 대전광역시 대덕구 94 대전광역시 유성구 138 전라북도 남원시 146 전라북도 전주시 147 전라북도 정읍시 <p>더덕:</p> <ul style="list-style-type: none"> 84 광주광역시 광산구 101 울산광역시 남구 <p>도라지:</p> <ul style="list-style-type: none"> 2 강원도 동해 3 강원도 삼척 7 강원도 영월 9 강원도 인제 16 강원도 화천 36 경기도 용인시 99 서울특별시 종로구 105 인천광역시 강화군 106 인천광역시 계양구 107 인천광역시 남동구 108 인천광역시 미추홀구

<pre> (climate_data['온도차이_숫자'] == 4)& (climate_data['최고기온_숫자'] == 1)]print('도라지:',filtered_data['지역']) import pandas as pd file_path = r"C:/Users/user/Desktop/기후데이터_온도차이_범주화_완.csv" filtered_data = climate_data[(climate_data['최저기온_숫자'] == 2) & (climate_data['최고기온_숫자'] == 2)& (climate_data['연평균기온_숫자'] == 2)]print('마:',filtered_data['지역']) </pre>	<pre> 110 인천광역시 서구 111 인천광역시 옹진군 112 인천광역시 중구 136 전라북도 군산시 158 충청남도 서산시 마: 19 경기도 고양시 20 경기도 과천시 29 경기도 안성시 35 경기도 오산시 41 경기도 하남시 71 경상북도 안동시 109 인천광역시 부평구 161 충청남도 예산군 162 충청남도 천안시 165 충청남도 홍성군 170 충청북도 옥천군 </pre>
---	---

3) 임업통계 등 데이터 적용 방안

[공공·민간데이터]

토양데이터로서 산림청의 “산림토양도(대축척 산림입지토양도)”에서 토양깊이유형, 토성코드, 토양형코드를 추출하고 “산림토양유효수분량” 데이터를 추출했습니다. 또한 결측치 제거 등의 전처리를 통해 최적의 데이터 셋을 구축했습니다. (이하 <코드 및 결과1>참조)토양데이터는 또한 기후데이터로서 기상청으로부터 “우리나라기후평균값”데이터셋을통해 연평균기온, 최고기온, 최저기온, 상대습도(%), 강수량(mm)을 추출했습니다.

[임업통계데이터]

임업통계데이터로 한국임업진흥원의 단기임산물재배적지도를 통해 웹페이지에 들어갈 시각화된 지도를 도출해냈습니다. (이하 <코드 및 결과2> 참조) 또한, KOSIS를 통해 산림청자료 임업소득조사데이터를 통해 연간 판매량 추이를 확인해 시각화했습니다.

<코드 및 결과 1>

결측치제거 전처리	
코드	결과
<pre> import pandas as pd df1=pd.read_csv("C:/Users/user/Desktop/토양수분생산량기후.csv") df1[df1['밤 (kg)'].isna()&df1['복분자딸기 (kg)'].isna()&df1['오갈피 (kg)'].isna()] df = df.dropna(subset=['밤 (kg)', '복분자딸기 (kg)', '오갈피 (kg)', '마 (kg)', '도라지 (kg)', '덕 (kg)', '생표고 (kg)', '능이 (kg)', how='all']) df </pre>	

<코드 및 결과 2>

시각화	
코드	결과
<pre> import pandas as pd import geopandas as gpd import fiona import numpy as np import matplotlib.pyplot as plt import geopandas as gpd from fiona.crs import from_epsg import matplotlib.pyplot as plt import numpy as np </pre>	

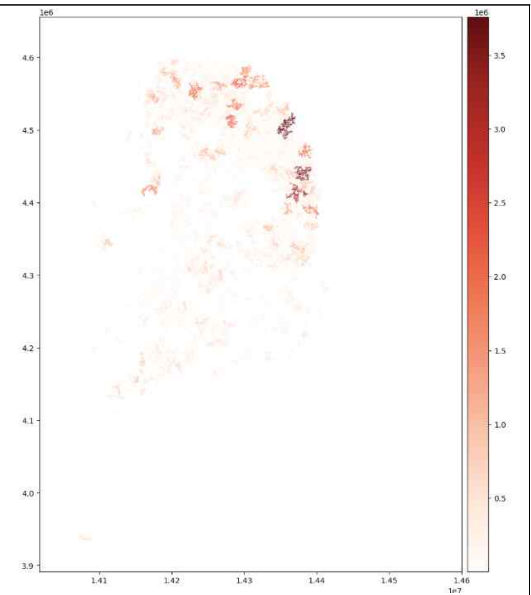
```

from mpl_toolkits.axes_grid1 import make_axes_locatable
오갈피=gpd.read_file('/Users/zjooo/Desktop/산림/오갈피(약용류)/오갈피_재배적지.shp', encoding='cp949')
오갈피2 = 오갈피.to_crs(epsg=3857)
fig = plt.figure(figsize=(18, 13))
ax = fig.subplots(1, 1)

divider = make_axes_locatable(ax)
cax = divider.append_axes("right", size="5%", pad=0.1)

오갈피2.plot("Shape_Leng", ax=ax, legend=True, cax=cax, cmap='Reds')
ax.set_title("재배적지", fontsize=25, pad=20)

```



4) 기대효과

1. 임업산업에 대한 진입 장벽 완화

- 정보 접근성 향상: 복잡한 공공 데이터를 시각적으로 제공해 예비 임업인들이 재배 적지와 관련 정보를 쉽게 이해하고 활용할 수 있음.
- 교육 및 자립 지원: 체계적인 데이터 기반 정보로 재배 작물 선정과 지역 결정에서 합리적이고 효율적인 의사결정이 가능.

2. 임산물 생산 극대화 및 품질 향상

- 최적 재배 환경 제공: 기후 및 토양 데이터를 분석해 작물에 적합한 재배 조건을 제안하여 생산량과 품질을 동시에 개선.
- 효율적인 자원 활용: 각 작물에 맞는 최적의 토양 분석과 기후 데이터를 통해 재배 실패 위험을 줄이고 자원을 효율적으로 배분 가능.

3. 임업 경제 활성화

- 소득 증대: 데이터 기반으로 생산성과 품질이 향상되면서 소비자와 직접 연결되어 임업인의 수익이 증가.
- 유통구조 간소화: 소비자가 플랫폼을 통해 임산물 가격 추이를 실시간으로 확인 가능하여 생산자와 소비자 간 연결성이 강화.