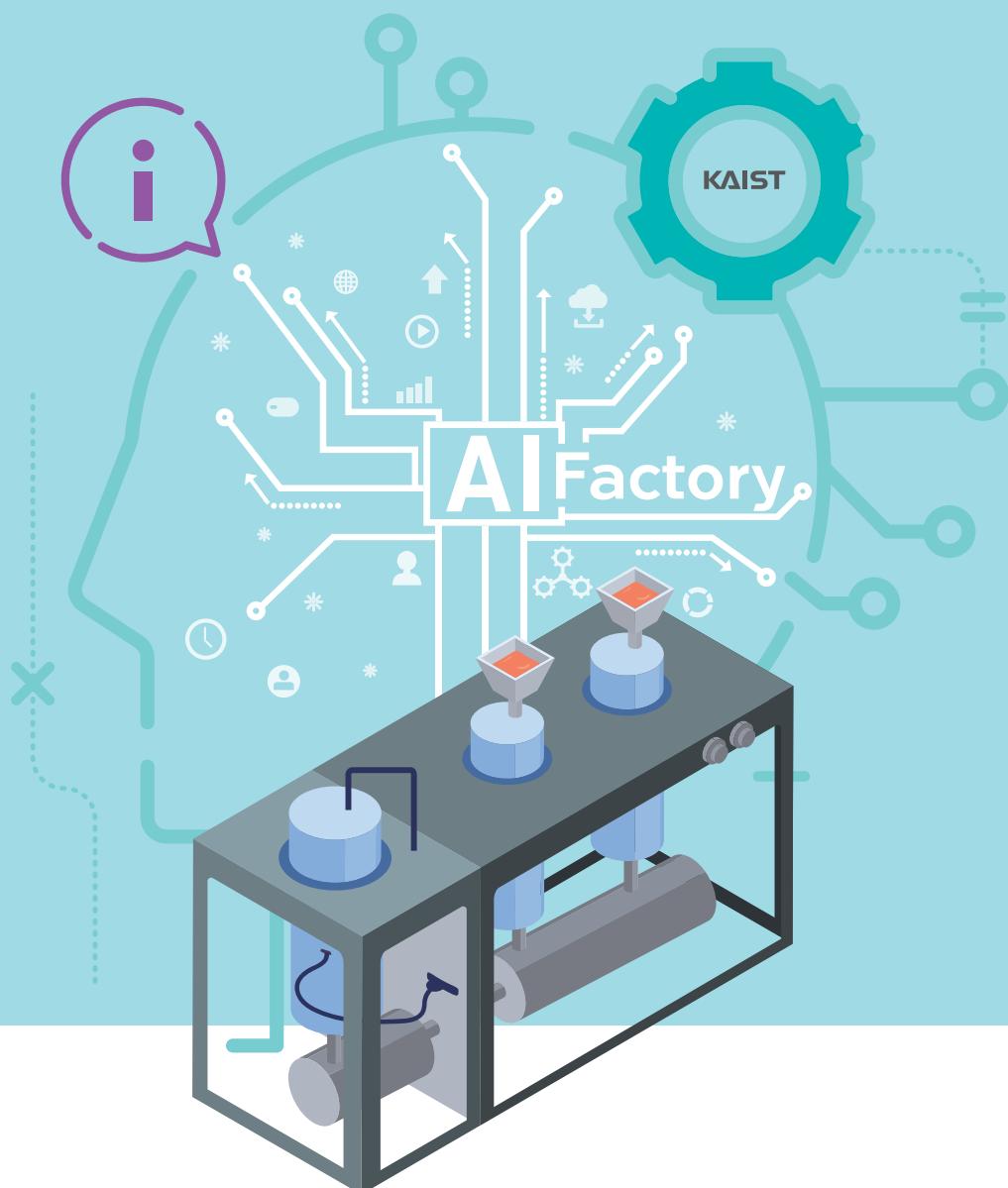
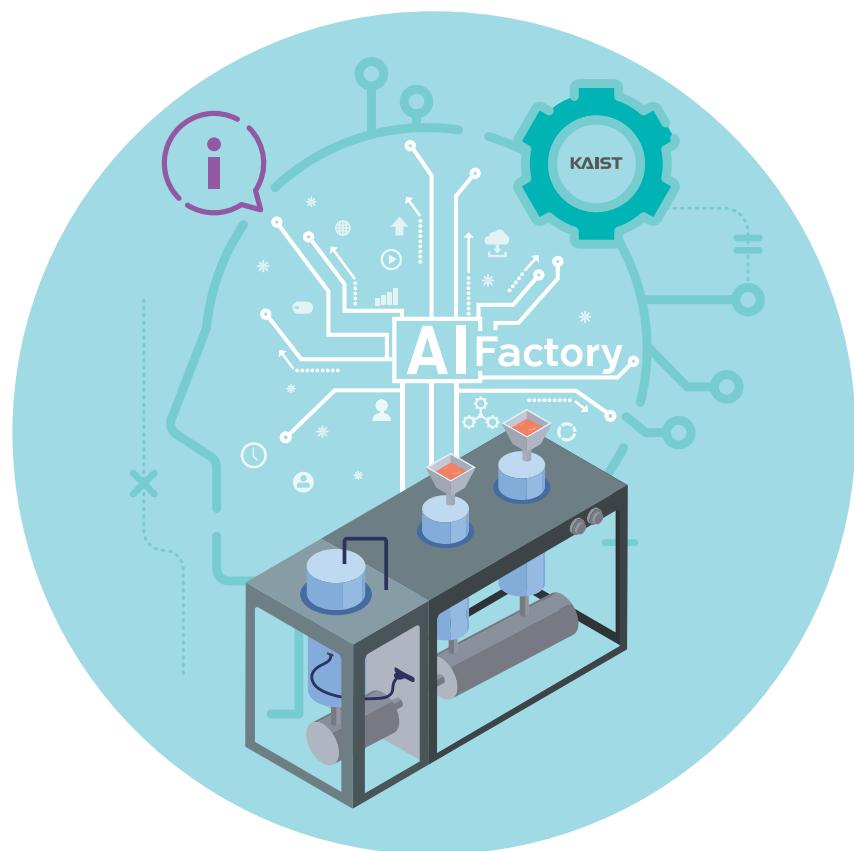


「살균기 AI 데이터셋」 분석실습 가이드북

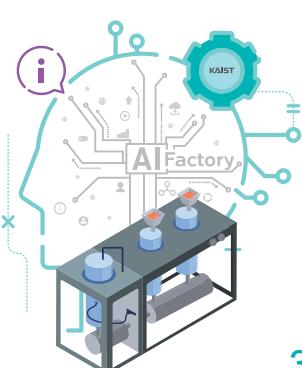


「살균기 AI 데이터셋」 분석실습 가이드북



Contents

1 분석요약	04
2 분석 실습	
1. 분석 개요	05
1.1 분석 배경	05
1) 공정(설비) 개요	
2) 이슈사항(Pain point)	
1.2 분석 목표	06
1) 분석목표	
2) 제조 데이터 정의 및 소개	
3) 제조 데이터 분석 기대효과	
4) 시사점(implication) 요약기술	
2. 분석실습	08
2.1 제조데이터 소개	08
1) 데이터 수집 방법	
2) 데이터 유형/구조	
2.2 분석 모델 소개	10
1) AI 분석모델	
2.3 분석 체험	17
1) 분석 단계별 Process	
2) 분석 실습	
[단계 ①] 라이브러리/데이터 불러오기	
[단계 ②] 데이터 종류 및 개수 확인	
[단계 ③] 데이터 정제(전처리)	
[단계 ④] 데이터 특성 파악	
[단계 ⑤] 학습/평가 데이터 분리	
[단계 ⑥] 의사결정나무 변수 설정 및 실행	
[단계 ⑦] 의사결정나무 시각화	
[단계 ⑧] 결과 분석 및 해석	
3. 유사 타 현장의 「살균기AI데이터셋」 분석 적용	39
부 록	
분석환경 구축을 위한 설치 가이드	40
Graphviz 설치 가이드	51



「살균기 AI 데이터셋」 분석실습 가이드북

- 필요 SW : Jupyter Notebook, Anaconda 등
- 필요 패키지 : Numpy, Pandas, Matplotlib, Os, Pydotplus, Seaborn, Graphviz, Scikit-learn, IPython
- 분석 환경 : [CPU] i5-10400, [GPU] gtx 1660 super, [RAM] 32GB
- 필요 데이터 : pasteurizer.csv
- 주관 기관 : 한국과학기술원(KAIST)
- 수행 기관 : (주)임피스, 한양대학교 산학협력단, 주식회사 아큐라소프트



1 분석요약

No	구분	내용
1	분석 목적 (현장 이슈, 목적)	<ul style="list-style-type: none">- 전처리 공정의 두 번째 단계인 살균공정(살균기)은 용해공정에서 액상상태로 혼합된 원재료 100°C 이하의 온도에서 30분 이상 저온살균 및 교반하는 작업을 하기 때문에 후공정 및 원제품 품질관리를 위해서는 본 공정의 품질이 매우 중요하다.- 이에 따라 용해탱크 운영데이터 분석을 통한 품질예측에 대한 의미가 커지며, 특히 불량 발생 후 원인분석을 진행하는 결과론적 품질분석이 아닌 예견된 불량을 효율적으로 방지할 수 있는 예지적 품질분석이 필요하게 된다.
2	데이터셋 형태 및 수집방법	<ul style="list-style-type: none">- 분석에 사용된 변수 : 살균상태, 살균온도, 양품/불량여부- 데이터 수집 방법 : PLC, DBMS(RDB)- 데이터셋 파일 확장자 : CSV
3	데이터 개수 데이터셋 총량	<ul style="list-style-type: none">- 데이터 개수 : 210,794건- 데이터셋 총량 : 6.11MB
4	분석적용 알고리즘	알고리즘 의사결정나무(Decision Tree)를 이용하여 품질 불량 선별에 중요한 설비운영데이터를 선별 및 예측하고 예측 정확도를 확인한다.
		알고리즘 간략소개 머신러닝에서 분류 또는 예측 모형으로 사용되는 지도학습 방법론으로, 중요 변수 중심의 의사결정 규칙을 “나무구조”로 도표화하여 분석을 수행하며, 목표변수가 범주형인 경우와 연속형인 경우로 나누어 활용할 수 있다.
5	분석결과 및 시사점	<ul style="list-style-type: none">- 공정중 살균기 설비운영값을 기준으로 제품의 최종 품질을 예측할 수 있는 AI 기법을 적용하여 공정 운영변수의 변화가 품질에 미칠 영향 모델링을 도출한다.- 살균공정의 설비운영 데이터와 최종품질검사 데이터를 수집하고, 데이터 가공/전처리, AI 모델 개발과 제조공정의 적용 및 검증을 통해 열악한 중소기업에 빅데이터 및 AI 기술을 적용하여 실질적인 품질향상 및 비용절감에 기여한다는 점에서 시사하는 바가 크다고 판단된다.

2 분석 실습

1. 분석 개요

1.1 분석 배경

1) 공정(설비) 개요

• 공정(설비) 정의 및 특징

- 가열살균공정은 열처리를 통해 액상 식품의 미생물을 사멸시켜 식품의 안전성과 보존성을 향상시키는 공정이다. 상압에서 100°C 이하의 온도에서 진행되는 저온살균 방식, 고압의 증기 또는 100°C 이상의 열수 속에서 진행하는 고온살균 방식, 그 외에도 마이크로파 살균, 원적외선 살균 등이 있다.
- 본 가이드북의 살균공정은 분말 유크림, 기능성 조제 분말 등을 생산하는 식품제조업의 저온살균공정으로, 용해공정에서 용해·혼합된 내용물을 50~70°C의 온도에서 30분 이상 살균 및 교반하는 작업을 한다.
- 살균공정은 원재료의 전처리를 수행하는 두 번째 단계이니만큼 본 공정의 품질이 후 공정 및 완제품 품질에 미치는 영향이 크다. 살균공정은 특히 HACCP¹ 주요관리점(CCP)이기 때문에 식품안전관리인증기준 이행 측면에서도 매우 중요한 공정이다.

2) 이슈사항(Pain point)

• 공정(설비)상의 문제현황

- 상술한 바와 같이 제품품질 및 식품안전 보장 측면에서 살균공정은 매우 중요하나, HACCP의 CCP 한계기준은 식품안전 측면의 기준이기 때문에 반드시 제품품질을 보장한다고는 할 수 없다.² 따라서 CCP 한계기준은 항상 최대치, 최소치 또는 범위로 지정되어 있으며, 제조현장에서 생산제품 및 설비에 따라 기준 내 공정/설비 운영값을 설정하도록 한다.
- 살균공정은 일반적으로 살균온도와 살균시간을 지표로 CCP 이행여부를 판단한다. 제조현장에서는 기준 자료, 시험결과 등을 바탕으로 제품별로 CCP 기준을 준수하면서 동시에 목표 품질을 위한 공정기준값을 설정하여 공정을 수행하는데, 가열방식의 특성상 살균공정 동안 내용물의 온도는 일정하게 유지되지 않고 계속해서 변동한다. 살

1) 식품안전관리인증기준 HACCP(Hazard Analysis & Critical Control Point)은 식품의 원재료 생산에서부터 제조, 가공, 보존, 유통단계를 거쳐 최종 소비자가 섭취하기 전까지의 각 단계에서 발생할 우려가 있는 위해요소(HA)를 규명하고, 이를 중점적으로 관리하기 위한 중요관리점(CCP)을 결정하여 자주적이고 효율적인 관리로 식품의 안전성을 확보하기 위한 과학적인 위생관리체계이다.

2) 한계기준은 CCP에서 관리되어야 할 생물학적, 화학적 또는 물리적 위해요소를 예방, 제거 또는 허용 가능한 안전한 수준까지 감소시킬 수 있는 최대치 또는 최소치를 말하며, 안전성을 보장할 수 있는 과학적 근거에 기초하여 설정한다. 한계기준을 결정할 때는 법적 요구조건과 연구논문이나 식품 관련 전문서적, 전문가 조언, 생산 공정의 기본자료 등 여러 조건을 고려하여 결정한다.

균기의 내부 센서가 이를 감지하여 설정온도가 유지되도록 이에 맞춰 가열온도를 변경하지만, 온도가 조정되는 동안의 시차가 있기 때문에 어느 정도의 온도변화는 불가피하며, 이에 따른 품질 영향은 현장에서 확인되지 못하고 있다.

• 문제해결 장애요인

- 살균기가 온도변화를 감지하는 시점에 온도설정을 변경하여도 내용물 전체에 영향을 미치는 시차는 항상 있으며, 특히 노후화된 설비는 센서 또는 가열기능이 불안정하기 때문에 설비 자체적으로 적정 온도가 항상 유지되길 기대하는 것은 어렵다. 이 때문에 대부분은 작업자가 현장에서 온도 모니터링을 하는데, 이마저도 경험과 직감에 의존한 주관적 판단으로 설정온도를 조정하기 때문에 일관적이지 않다.

• 극복방안

- 살균공정의 CCP 이행과 제품품질을 둘 다 확보하기 위해서는 지금과 같이 노후화된 살균기의 자체 기능에만 의존하거나 작업자의 주관적 판단에 의존한 대처 방법이 아닌 데이터에 기반한 품질예측 방법이 필요하다. 살균여부의 주요 지표인 살균온도와 제품 불량 데이터의 머신학습을 통하여 살균공정 중 실시간으로 품질을 예측하고 이를 공정제어에 이용할 수 있다.

1.2 분석 목표

1) 분석목표

- 가열방식의 특성상 살균공정의 내용물은 설정온도를 유지하지 못하고 계속해서 변동 하며, 설비센서 또는 현장작업자는 설비 PLC 또는 모니터링 시스템에 표시되는 온도 변화추이를 바탕으로 설정온도를 조정하여 살균온도가 CCP 한계기준과 공정기준을 이탈하지 않도록 한다. 그러나 대부분의 중소제조업체에서 사용하는 노후화된 설비는 데이터에 기반한 고도화된 기능을 갖추지 못하고 있고, 작업자는 주관적 판단에 의지함에 따라 원활하고 일관적인 공정제어가 이뤄지지 못하고 있는 현실이다.
- 이에 살균공정의 CCP 이행여부를 결정하고 제품품질(품미, 특성 등)에 영향을 미치는 살균온도 데이터와 최종품질검사를 수행하여 획득한 제품 불량여부 데이터를 활용하여 머신을 학습시키고, 공정 중의 설비운영 데이터를 학습모델에 적용하여 완제품 품질을 예측하고자 한다.

2) 제조 데이터 정의 및 소개

- 살균공정의 CCP 이행여부를 결정하고 제품품질에 영향을 미치는 살균온도 데이터와 최종품질검사를 수행하여 획득한 제품 불량여부 데이터를 활용하여 품질예측을 하기 위한 머신학습 데이터이다.
- 설비운영 데이터는 살균기의 PLC를 통해 일반적으로 수집할 수 있는 데이터를 범위로 하였으며, 품질데이터는 식품제조업에서 일반적으로 진행하는 최종검사 결과를 범위로 하여, 본 가이드북에 따라 학습하고 유사 현장에 적용하는 데 있어 기존의 설비 및 프로세스 외에 추가 작업이 필요 없도록 하였다.

3) 제조 데이터 분석 기대효과

- 본 분석을 통해 설비운영값과 생산품질의 연관성을 도출하여, 설비운영조건에 따라 생산품질을 예측할 수 있게 하며, 생산성은 높이고 불량률은 낮추는 설비운영 최적화를 위한 분석 데이터셋을 구축한다. 공정설비 운영데이터에 따라 생산품질을 예측할 수 있는 제품 생산 공정에 넓게 활용할 수 있을 것으로 판단된다.
- 특히, 불량 발생 후 원인분석을 진행하는 결과론적 품질분석이 아닌 예견된 불량을 효율적으로 방지할 수 있는 예지적 품질분석을 수행할 수 있게 되어, AI 추론을 통한 생산 시점의 실시간 불량 발생 예측 알람을 활용할 수 있을 것으로 판단된다.
- 또한, 공정중 실시간 불량 예측과 이를 기반으로 한 공정제어를 구현함으로써 데이터에 기반한 'CCP 기준이탈 모니터링 및 품질예측'이 가능한 AI Smart HACCP 시스템을 구축할 수 있는 기반을 마련할 수 있게 된다.

4) 시사점(implication) 요약기술

- 식품제조업체 H사 공장에서는 노후화되고, 기능이 고도화되지 않은 설비의 한계로 인해 가열살균 중 내부온도가 유지되지 못하고 변동하는 상황에 대해 작업자의 판단으로 설비운영 셋팅값을 조정하여야 했기 때문에 일관적인 품질관리에 어려움이 있었다.
- 따라서 공정중 설비운영값을 기준으로 제품의 최종 품질을 예측할 수 있는 AI 기법을 적용하여 각 공정 운영변수의 변화가 품질에 미칠 영향 모델링을 도출하고자 한다. 살균공정 설비의 생산 데이터를 예제로 기본적인 데이터 탐색을 통해 전체 데이터 및 각 변수의 특성을 파악하는 방법을 학습하고, 이를 바탕으로 생산품 품질에 주된 영향을 미치는 공정변수 및 그 연관성 등을 파악하여 본다. 영향 인자의 우선순위를 결정하고, 최종 품질을 예측하는 AI 모델을 구성하고자 한다.
- 살균공정의 설비운영 데이터와 최종품질검사 데이터를 수집하고, 데이터 가공/전처리, AI 모델 개발과 제조공정의 적용 및 검증을 통해 열악한 중소기업에 빅데이터 및 AI 기술을 적용하여, 실질적인 품질향상 및 비용절감에 기여한다는 점에서 시사하는 바가 크다고 판단된다.

2. 분석실습

2.1 제조데이터 소개

1) 데이터 수집 방법

- 제조 분야 : 분무건조공법을 이용한 분말유크림 제조
- 제조 공정명 : 혼합살균
- 수집장비 : PLC(설비데이터) 및 DBMS(품질데이터)
- 수집기간 : 2020년 3월 4일 ~ 2020년 11월 11일 (약 8.5개월)
- 수집주기 : 불규칙 (※전처리 후 사이클타임 약 30분)

2) 데이터 유형/구조

STD_DT	MIXA_PASTEUR_STATE	MIXB_PASTEUR_STATE	MIXA_PASTEUR_TEMP	MIXB_PASTEUR_TEMP	INSP
2020-03-04 6:00	1	1	551	524	OK
2020-03-04 6:30	1	1	584	536	OK
2020-03-04 7:00	1	1	584	536	OK
2020-03-04 7:30	1	1	585	536	OK
2020-03-04 8:00	1	1	585	536	OK
2020-03-04 8:30	1	1	585	536	OK
2020-03-04 9:00	1	1	585	537	OK
2020-03-04 9:30	1	1	585	538	OK
2020-03-04 10:00	1	1	585	541	OK
2020-03-04 10:30	1	1	585	543	OK
2020-03-04 11:00	1	1	585	545	OK
2020-03-04 11:30	1	1	585	548	OK
2020-03-04 12:00	1	1	584	551	OK
2020-03-04 12:30	1	1	584	554	OK
2020-03-04 13:00	1	1	582	556	OK
2020-03-04 13:15	1		551	524	NG
2020-03-04 13:30	1	1	580	558	OK
2020-03-04 14:00	1	1	577	561	OK
2020-03-04 14:30	1	1	573	564	OK
2020-03-04 15:00	1	1	572	566	OK
2020-03-04 15:30	1	1	566	569	OK
2020-03-04 16:00	1	1	559	570	OK
2020-03-04 16:30	1	1	554	571	OK
2020-03-04 17:00	1	1	551	572	OK
2020-03-04 17:30	1	1	549	574	OK
2020-03-04 18:00	1	1	546	575	OK
2020-03-04 18:30	1	1	544	576	OK
2020-03-04 19:00	1	1	543	576	OK
2020-03-04 19:30	1	1	542	577	OK
2020-03-04 20:00	1	1	540	578	OK

[그림 1] 살균기 데이터 스크린샷

- 데이터 크기, 데이터 수량 : 6개 칼럼, 210,794개의 관측치

• 데이터 속성정의 표 :

변수명	설명	데이터 타입
STD_DT	날짜, 시간(YYYY-MM-DD HH:MM:SS)	object
MIXA_PASTEUR_STATE	살균기A 가동상태	float64
MIXB_PASTEUR_STATE	살균기B 가동상태	float64
MIXA_PASTEUR_TEMP	살균기A 살균온도	float64
MIXB_PASTEUR_TEMP	살균기B 살균온도	float64
INSP	불량여부	object

* object = 문자, float64 = 실수

- 본 가이드북에서 사례로 활용한 H사에서는 생산량에 따라 살균공정을 설비 2대에 나눠 병렬적으로 진행하기도 함에 따라 살균공정 품질예측 분석을 위한 데이터셋은 살균기 2대(MIXA, MIXB)의 공정(설비운영)변수와 생산된 제품의 최종 품질판정(불량여부) 값으로 구성한다.
- 본 가이드북의 데이터셋은 데이터 수집일시(STD_DT), 살균기A 가동상태(MIXA_PASTEUR_STATE), 살균기B 가동상태(MIXB_PASTEUR_STATE), 살균기A 살균온도(MIXA_PASTEUR_TEMP), 살균기B 살균온도(MIXB_PASTEUR_TEMP), 불량여부(INSP) 등 6개 칼럼으로 이루어져 있으며, 가동상태(0, 1)와 불량여부(OK, NG)는 명목형 범주형이고, 살균온도는 연속형 데이터이다.
- 살균기 가동상태(_PASTEUR_STATE) 값 0은 ‘정지’, 1은 ‘운전’을 의미하며, 불량여부(INSP) 값 OK는 ‘양품’, NG는 ‘불량’을 의미한다. 또한, 살균온도 데이터(_PASTEUR_TEMP)는 소수점 1자리가 생략되어 있기 때문에 값 nnn은 실제로 nn.n을 의미한다 (예: 501 → 50.1°C)

• 기술통계 :

구분	MIXA_PASTEUR_STATE	MIXB_PASTEUR_STATE	MIXA_PASTEUR_TEMP	MIXB_PASTEUR_TEMP
개 수	11,135	10,255	201,423	198,802
평 균	50,327	633	567	18,626
표준편차	528,690	6,408	69	811,171
최 소 값	0	0	0	0
중 간 값	1	1	570	569
최 대 값	5,603,841	65,536	772	42,795,010
최 빈 값	1	1	571	564

* 실제 알고리즘에 사용하는 데이터 중 수치 데이터만 기술

- 위 기술통계에 따르면 원시 데이터에는 가동상태의 최대값이 5,603,841로 나오거나, 살균온도의 최대값이 42,795,010이 나오는 등 비현실적, 비정상적인 값들이 있는데, 이는 설비오류 또는 데이터 수집 장치의 일시적 오류로 인한 값들로 보이며, 이같은 이상치들은 데이터 전처리 및 정제 작업 시 제거될 예정이다.

• 독립변수/종속변수 정의 :

구분	명칭
독립변수	살균기A 가동상태
	살균기B 가동상태
	살균기A 살균온도
	살균기B 살균온도
종속변수	불량여부

* 독립변수란 다른 변수에 영향을 받지 않는 변수로, 입력값이나 원인을 나타낸다.
종속변수란 독립변수의 변화에 따라 어떻게 변하는지를 알고자 하는 변수를 말하며, 결과물이나 효과를 나타낸다.

- 본 분석에서는 원료 용해혼합액의 살균상태에 영향을 미쳐 최종 제품품질에 영향을 줄 것으로 생각되는 공정변수로 살균기 2대의 가동상태와 살균온도 등 4개의 변수를 고려하였다. 따라서 설비운영변수(가동상태, 온도)가 독립변수, 품질결과값(불량여부)이 종속변수가 된다.

2.2 분석 모델 소개

1) AI 분석모델

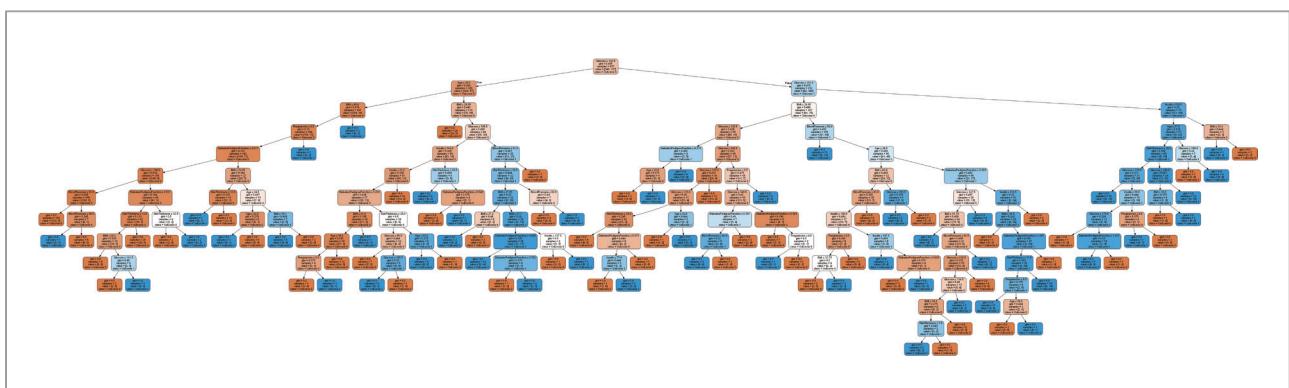
• 해당 AI 방법론(알고리즘) 선정 이유 기술

- 살균기 데이터 분석 방법 : 의사결정나무
 - 의사결정나무는 직관적인 머신러닝 알고리즘으로, 학습을 통해 데이터의 규칙을 자동으로 찾아내어 분류와 예측을 하는 강력한 알고리즈다.
 - 연속형 혹은 범주형의 입력변수와 목표변수를 모두 취급 가능하고 분류 규칙이 명확하여 해석이 용이하다. 선형성, 정규성 등의 가정이 필요하지 않아 전처리 과정에 큰 영향을 받지 않는다. 또한, 두 개 이상의 변수가 목표변수에 어떤 영향을 주는지 쉽게 파악 가능하다.

- 적용하고자 하는 AI 분석 방법론(알고리즘)의 구체적 소개

- 의사결정나무 알고리즘의 이해

- 의사결정나무(Decision Tree) 알고리즘은 머신러닝에서 분류 또는 예측 모형으로 사용되는 지도학습 방법론으로, 의사결정 규칙을 “나무구조”로 도표화하여 분석을 수행한다.
 - 의사결정나무는 [그림 2]와 같은 형태로 이뤄져 있다. 가장 상위의 마디를 뿌리노드라고 하고 그 아래 규칙노드들이 위치한다. 각 노드는 규칙조건에 따라서 분할된다. 더이상 나누어지지 않는 말단의 노드는 리프노드라고 하며 결정된 클래스 값이 된다. 즉, 원본 데이터에서 하나의 규칙을 만들 때마다 규칙 노드를 만들어서 가지를 치면서 내려가는 형태이다.



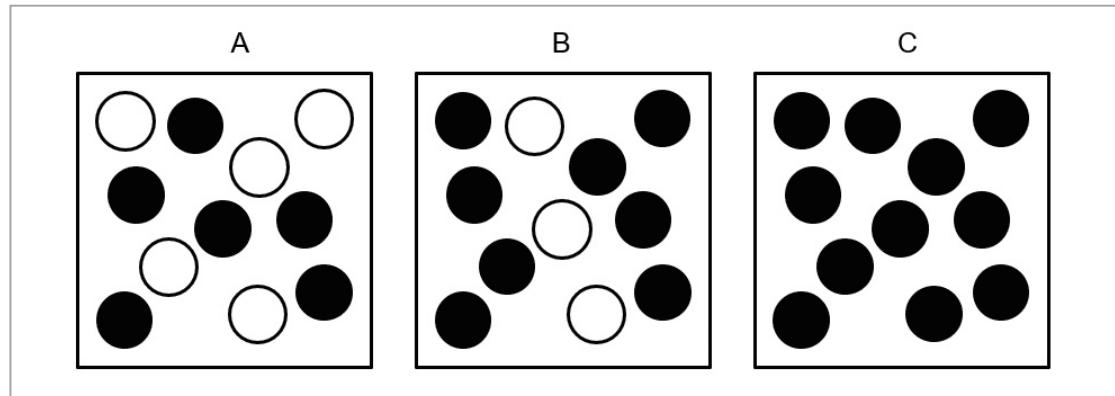
[그림 2] 의사결정나무 모형도

- 의사결정나무는 일반적으로 if / else 형태로 규칙을 표현하는데 이는 일종의 스무고개를 반복하는 구조이다. 예를 들어, ‘데이터가 10보다 큰 값인가?’라는 규칙을 만들고, 결과가 참인 것과 그렇지 않은 값을 분할하여 나누는 형태이다. 이처럼 규칙을 직관적으로 이해하기 쉽다는 장점이 있지만, 과적합이 발생하기 쉽다는 단점 또한 존재한다. 따라서 과적합이 발생하는 것을 방지하기 위한 제어가 필요하다.

- 과적합의 발생과 제어

- 과적합(Overfitting)이란 머신러닝을 할 때 학습 데이터의 패턴을 과도하게 학습하는 것을 의미한다. 간단히 이야기하면 샘플 데이터만 과도하게 학습하여 샘플 데이터에 대해서는 높은 예측력을 가지지만, 실제 데이터를 적용하면 성능이 떨어지는 것을 말한다. 그러므로 머신러닝을 설계하는 과정에서 과적합이 발생하지 않도록 유의해야 한다. 특히 의사결정나무 알고리즘은 가지치기를 하는 과정에서 다양한 규칙을 생성하기 때문에 과적합이 발생할 가능성이 큰 알고리즘이다. 따라서 의사결정 알고리즘을 사용할 때에는 과적합의 발생을 방지하기 위한 방법을 적용해야 한다.

- 과적합을 방지하기 위해서는 균일도가 높은 데이터셋이 나타나도록 분할해야 한다. 먼저 균일한 데이터셋에 대해 알아보자.



[그림 3] 데이터셋 혼잡도 예시

- 위의 [그림 3]에서 가장 균일한 데이터는 C이다. 그다음으로 균일도가 높은 것은 B, 마지막이 A가 된다. 모두 동일한 데이터로 구성된 것일수록 균일하고 다른 데이터가 섞여 있을수록 균일도가 낮아진다. 이와 같이 데이터의 균일도에 따라서 데이터를 분할하는데 필요한 정보의 양에 차이가 생긴다. 눈을 감은 상태에서 C와 A의 데이터를 뽑아 어떤 색인지 맞출 때, C의 데이터는 모두 검은색인 것을 쉽게 예측할 수 있지만, A는 추가적인 정보가 있어야 예측을 할 수 있기 때문이다.
- 의사결정나무의 노드는 정보 균일도를 측정하여 균일도가 높은 데이터셋을 먼저 뽑아서 규칙을 생성한다. 균일도가 높은 데이터셋으로 나눠지도록 상위 데이터셋을 만들고 아래로 가지를 치는 과정을 반복하며 값을 예측한다. 의사결정나무 알고리즘에서 정보 균일도를 측정하는 두 가지 대표적인 방법은 엔트로피를 사용한 정보이득 (information gain)과 지니계수이다.
- 먼저 정보이득 개념에 대해 알아보자. 정보이득은 엔트로피 개념을 바탕으로 한다. 엔트로피는 데이터의 혼잡도를 의미하는 것으로, 혼잡도가 높을수록(데이터가 균일하지 않을수록) 엔트로피가 증가한다. 따라서 엔트로피를 측정하면 데이터의 균일도를 측정할 수 있다. 결국, 균일한 데이터는 엔트로피 지수가 낮게 측정된다. 정보 이득 지수는 1에서 엔트로피지수를 뺀 값으로, $[1 - \text{엔트로피지수}]$ 의 형태로 이뤄져 있다. 앞서 살펴보았듯이 균일한 데이터는 엔트로피 지수가 작고 따라서 정보이득 지수는 크게 설정된다. 이를 토대로 의사결정나무는 정보이득이 높은 방향으로 데이터를 분할한다.
- 지니계수는 경제학에서 불평등 지수를 나타낼 때 사용한 값이다. 경제학자 코라도 지니(Corrado Gini)의 이름을 따라 만들어진 계수로서 0은 가장 불평등한 상태를 의미하고 1은 가장 평등한 상태를 의미한다. 데이터의 균일도 측면에서 살펴보면 모든 데이터가 단일한 값으로만 이뤄져 있을 때, 값이 고르게 퍼져있지 않기 때문에 지

지니계수는 0을 향한다. 반대로 여러 값이 섞인 데이터는 지니계수가 증가한다. 따라서 의사결정나무는 지니계수가 낮은 쪽으로 데이터를 분할한다.

- 의사결정나무 평가

- 머신러닝 프로세스는 데이터 가공 및 변환, 모델 트레이닝 및 예측, 평가의 단계로 이뤄져 있다. 머신러닝 모델의 성능은 다양한 방법으로 평가할 수 있다. 일반적으로 사용되는 방법은 실제 데이터와 예측 데이터의 차이가 얼마나 나타나는지 확인하는 방법이다. 하지만 이 방법 이외에도 여러 방법이 존재한다.

다음은 성능평가 지표들이다.

- * 정확도
- * 오차행렬
- * 정밀도
- * 재현율
- * F1 스코어
- * ROC-AUC

- 정확도는 다음과 같은 식으로 이뤄져 있다.

「예측 결과가 동일한 데이터 건수 ÷ 전체 예측 데이터 건수」

정확도는 예측 데이터와 실제 데이터의 동일한 정도를 판단하는 지표이다. 직관적으로 모델의 성능을 나타내는 지표이지만 이진분류의 경우 데이터가 어떤 구성으로 이뤄져 있는지에 따라서 성능을 왜곡할 가능성이 있다. 따라서 정확도 하나만으로 성능을 평가하지 않는다. 정확도만으로 모델의 성능을 평가하는 것이 바람직하지 못한 사례를 살펴보도록 하자.

- 90개의 검은 공과 10개의 흰 공으로 이뤄진 상자에서 공의 색깔을 예측하는 모델이 있다고 가정해보자. 100개의 공 가운데 검은색 공이 90개에 이르기 때문에 모든 공이 검은색이라고 예측하는 단순한 모델의 경우에도 정확도는 90%로 상당히 정확하게 나타난다. 이처럼 불균형한 레이블 값을 가진 데이터셋에서 모델의 성능을 평가할 때, 적합한 방법이 아니다. 이러한 단점을 극복하기 위해 다양한 분류지표와 함께 성능을 평가해야 한다. 다음은 오차행렬(Confusion Matrix)에 대해 살펴보도록 한다.
- 오차행렬은 이진분류에서 성능지표로 주로 활용되고 있다. 예측오류가 얼마인지, 어떤 유형의 오류가 발생하였는지를 나타내는 지표이다. 오차행렬은 다음 [그림 4]의 4분면 행렬에서 실제 값과 예측값이 어떤 유형으로 나타나는지 보여준다.

		Condition(실제)	
		Positive	Negative
Prediction (예측)	Positive	TP – True Positive	FP – False Positive
	Negative	FN – False Negative	TN – True Negative

[그림 4] 오차행렬(Confusion Matrix)

- TN, FP, FN, TP는 예측값과 실제값의 긍정값과 부정값의 결합 방식에 따라 결정된다. 예를 들어 TP는 참-양성의 의미로서 앞의 True는 예측과 실제가 동일하다는 의미이고 뒤쪽의 Positive는 예측을 긍정으로 하였다는 의미이다. 사이킷런(scikit-learn)에서는 오차행렬을 구하기 위한 API로 `confusion_matrix()`를 제공하고 있다.
- TN, FP, FN, TP는 알고리즘의 예측 성능을 판별할 수 있는 정보를 제공한다. 각각의 값을 조합하여 정확도, 정밀도, 재현율을 확인할 수 있다. 앞서 살펴보았던 정확도(Accuracy)를 구하는 공식은 다음과 같다.

$$\text{정확도} = (TN + TP) / (TN + FP + FN + TP)$$

- 하지만 정확도는 분류 모델의 성능을 측정하는 여러 지표 가운데 한가지일 뿐이다. 불균형한 데이터셋을 평가할 때는 정밀도와 재현율이 더 선호되는 평가지표이다.
- 정밀도는 데이터셋의 예측 성능의 평가에 초점을 맞춘 지표이다. 긍정으로 예측을 한 레이블 가운데 실제값이 예측값과 동일하게 긍정으로 일치한 비율을 의미한다. 정밀도는 다음과 같은 공식으로 계산된다.

$$\text{정밀도} = TP / (TP + FP)$$

양성예측 성능에 초점을 두고 평가하기 때문에 양성 예측도라고도 한다.

- 재현율 또한 데이터셋의 예측 성능의 평가에 초점을 맞춘 지표이다. 재현율은 실제 값이 긍정인 레이블 가운데 예측과 실제값이 긍정으로 일치한 데이터의 비율을 의미한다. 민감도(Sensitivity) 또는 TPR(True Positive Rate)이라고도 불린다. 재현율은 다음과 같은 공식으로 계산된다.

$$\text{재현율} = TP / (FN + TP)$$

- 정밀도와 재현율은 모델의 목적에 따라 어떤 것이 중요한지 결정된다. 재현율은 실제 긍정인 레이블을 부정으로 잘못 판단하면 큰 손실이 발생하는 경우에 중요 지표로 사용된다. 예를 들어, 암 환자 진단 모델은 재현율을 중요한 지표로 간주한다. 실제 암인 환자를 음성으로 판단하면 환자의 목숨에 큰 영향을 미치기 때문이다. 반면 실제로 음성인 환자를 암에 걸린 것으로 판단하여도 추가적인 검사를 하는 것 정도의 손실이 발생하기 때문에 이런 경우, 재현율을 중요 지표로 간주한다. 일반적으로

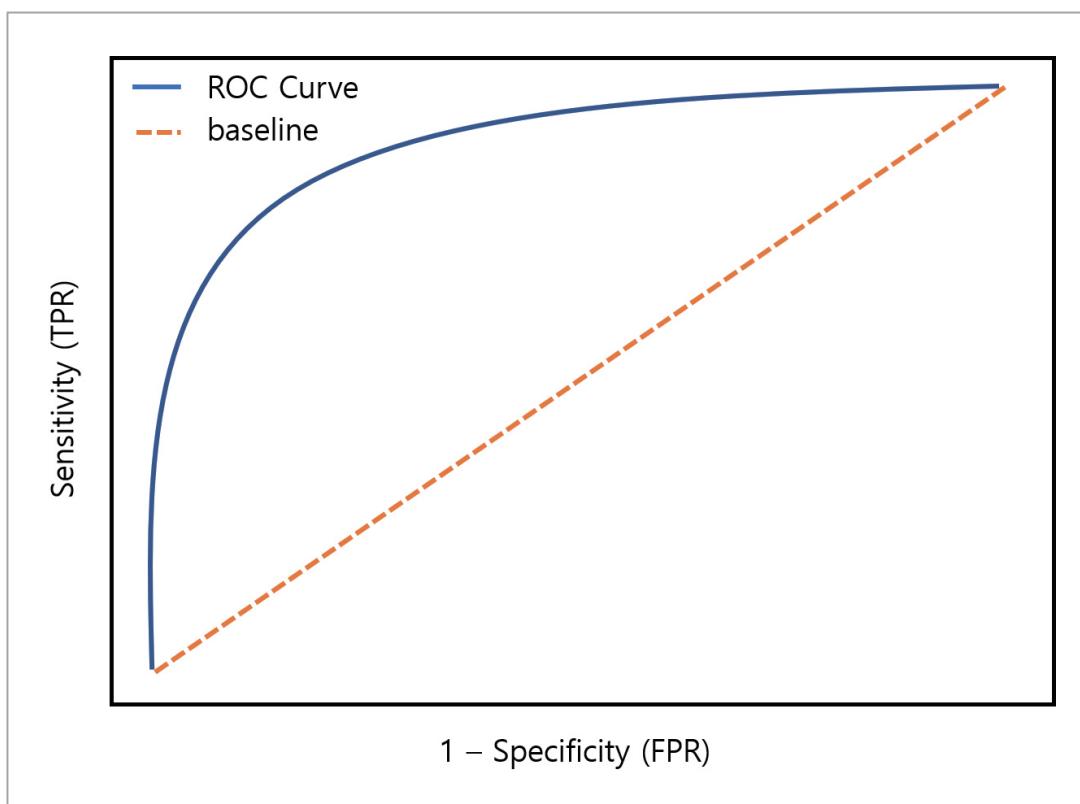
재현율이 더욱 중요한 지표로 간주되지만 스팸메일 여부를 판단하는 모델의 경우와 같이 정밀도가 더 중요한 경우도 존재한다. 실제로 스팸인 메일을 음성으로 판단하여도 스팸메일을 받는 정도의 손해가 발생하지만, 일반 메일을 스팸으로 분류한다면 메일을 받지 못하는 불편을 끼치기 때문이다.

- 가장 좋은 경우는 두 지표 모두 높은 경우이지만, 두 지표는 상호 보완적인 성격을 지니기 때문에 한쪽을 높이면 다른 지표는 낮은 값을 가지는 경우가 일반적이다. 이러한 현상을 정밀도/재현율 트레이드오프(Trade off)라고 한다.
- F1 스코어는 정밀도와 재현율을 종합적으로 고려한 지표이다. 두 지표가 어느 한쪽으로 치우치지 않을 때 높은 값을 가지게 된다. F1 스코어는 다음과 같은 공식으로 계산된다.

$$F1 \text{ 스코어} = 2 * \{(정밀도 * 재현율) / (정밀도 + 재현율)\}$$

사이킷런(scikit-learn)은 F1 스코어를 계산하기 위해 `f1_score()` API를 제공하고 있다.

- ROC 곡선과 AUC 점수는 민감도와 특이도의 상충관계를 파악하기 위해 사용되는 지표이다. ROC 곡선은 FPR(False Positive 비율) 대비 TPR(True Positive 비율)이 어떻게 변화하는지 나타내는 곡선이다. AUC는 Area Under the Curve의 약자로 ROC의 아래 부분 면적을 나타낸 것이다.



[그림 5] ROC-AUC 그래프

- [그림 5]에서 파란 선이 ROC 곡선을 나타내며 파란선 아래 부분이 AUC를 의미한다. AUC가 클수록 특이도를 떨 손상시킨 상태에서 최적의 민감도를 산출하였다는 의미가 된다. 따라서 분석을 할 때 AUC 점수를 확인하여 분석 결과를 타당성 있게 평가하여야 한다.

- AI 분석 방법론(알고리즘) 구축 절차 설명

- 변수 분할

- 전체 변수에서 독립변수와 종속변수를 분할한다. 이어서 학습 데이터와 테스트 데이터로 분할한다. 살균기 데이터의 경우 온도와 가동상태 관련 데이터를 독립변수로 설정하고 품질결과 변수를 종속변수로 설정하게 된다. 독립변수와 종속변수의 구별이 끝나면 해당 값들의 일부는 학습 데이터로, 일부는 테스트 데이터로 나누어 머신러닝을 진행한다.

- 의사결정나무 실행 및 시각화

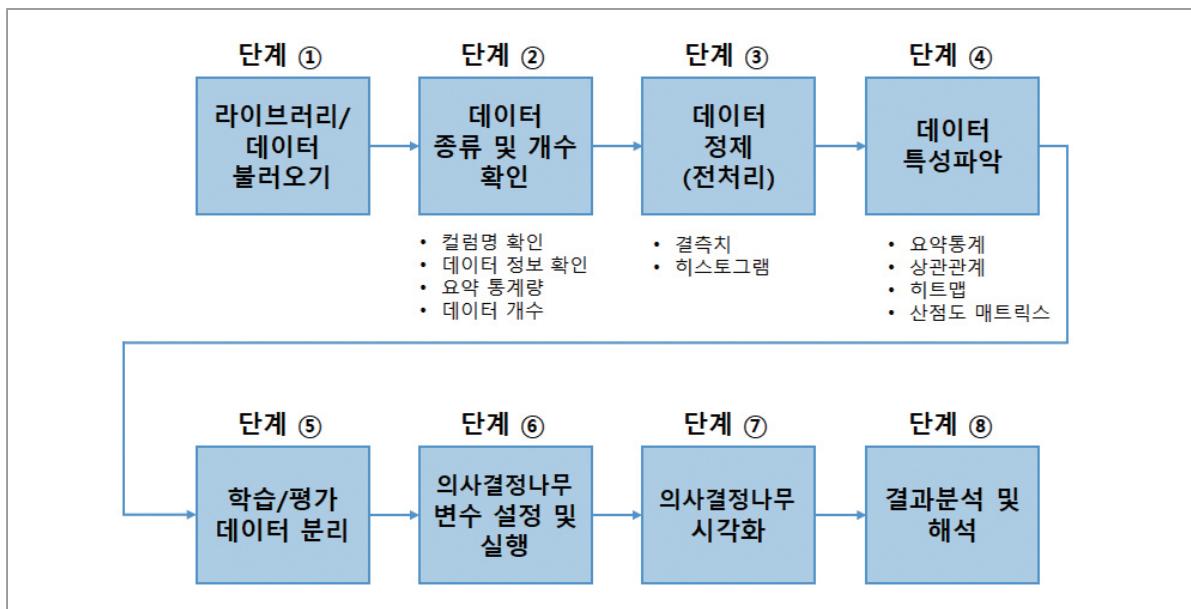
- 의사결정 나무 알고리즘을 활용하여 데이터를 분류한다. 이때, 과적합이 발생하지 않도록 max_depth를 조정하여 적절한 수준을 찾아가도록 한다. 추가적으로 시각화를 통해 어떤 규칙으로 모델링이 이뤄졌는지 파악한다.

- 모델 평가 및 해석

- 컴퓨터 매트릭스를 활용하여 모델의 성능을 평가한다. 이를 통해 정확도, 정밀도, 재현율, 민감도, 특이도와 F1-score를 산출한다. 이 과정을 거쳐 불량으로 예측한 데이터가 실제 불량 데이터와 얼마나 일치하는지 확인하여 효과적인 파라미터를 결정한다. 또한, 시각화 결과를 살펴보며 변수별로 어떤 규칙에 의해 분류가 이뤄졌는지, 양품 판정을 위한 변수별 기준을 확인한다.

2.3 분석 체험

1) 분석 단계별 Process



[그림 6] 분석 단계 Flow Chart

2) 분석 실습

[단계 ①] 라이브러리/데이터 불러오기

①-1. Graphviz 설치하기

- 가장 먼저 데이터 시각화 툴의 하나인 Graphviz를 '[부록] Graphviz 설치 가이드'를 참고하여 설치한다. 본 분석에서 Graphviz는 의사결정나무(Decision Tree) 모델 시각화에 사용될 것이다.

①-2. 필요 라이브러리 불러오기

- 분석에 앞서 우선 필요한 Package를 불러와야 한다(import). Package는 한마디로, 사용하고자 하는 계산식 혹은 알고리즘 관련 함수를 모아놓은 모듈들의 모음이라고 할 수 있다. 예를 들어, 함수를 이용하여 원의 면적을 계산하겠다면 계산식 안에 pi가 필요할 것이다. 이때 pi는 수학 관련 함수를 모아놓은 math 모듈에서 가져올 수 있다. math 모듈을 가져오지 않으면 직접 작성한 수식에 pi를 같이 적더라도 코드가 실행되지 않는다. 꼭 계산식이 아니더라도 다양한 머신러닝/딥러닝 알고리즘을 단 몇 줄만으로 수행하기 위해서는 이와 관련된 기능이 내장된 패키지를 설치하고 불러오는 과정이 필수적이다. 다음 과정을 통해 필요한 Package를 설치하고 불러오는 방법을 확인한다.

```
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
import graphviz
import seaborn as sns
from IPython.display import Image
import pydotplus
import pandas as pd
import numpy as np
import os
```

[그림 7] 필요 패키지 및 모듈을 불러오기 위한 코드

- Package와 모듈을 불러오는 코드는 ‘from 패키지명 import 모듈명’을 기본으로 한다. 아래 코드와 결과를 보면, pandas_profiling이라는 패키지 안의 ProfileReport 를 불러오기 위해 ‘from pandas_profiling import ProfileReport’를 입력하였다. 하지만 [ModuleNotFoundError : No module named ‘pandas_profiling’] 이라는 오류가 발생한 것을 볼 수 있다. 이러한 오류가 나는 이유는 현재 pandas_profiling이라는 Package가 설치되어있지 않기 때문이다.

```
from pandas_profiling import ProfileReport
```

[그림 8] 필요 패키지 및 모듈을 불러오기 위한 코드

```
ModuleNotFoundError Traceback (most recent call last)
<ipython-input-3-e2a33329b6f0> in <module>
      1 from pandas_profiling import ProfileReport
ModuleNotFoundError: No module named 'pandas_profiling'
```

[그림 9] Package 미설치 시 발생하는 오류

- 이를 해결하기 위해서는 Package를 설치해야 하며 설치 기본 코드는 ‘!pip install 패키지명’이다. 따라서 아래의 예시에서는 ‘!pip install pandas-profiling’을 입력하였다. 이 줄을 실행하여 Package가 잘 설치되었다면 아래 결과와 같이 ‘Successfully installed ~.’라는 문구를 확인할 수 있다. 결과 내용 중의 WARNING 메시지는 Package 설치 여부에 크게 영향을 미치지 않는 부분이니 ‘Successfully installed ~.’ 문구만 필수적으로 확인하도록 한다.

```
!pip install pandas-profiling
```

[그림 10] Package 설치를 위한 코드

```

Requirement already satisfied: testpath in c:\users\ingull\anaconda3\lib\site-packages (from nbconvert->notebook>=4.4.1->widgetsnbextension>=3.5.0->ipywidgets>=7.5.1->pandas-profiling) (0.4.2)
Requirement already satisfied: defusedxml in c:\users\ingull\anaconda3\lib\site-packages (from nbconvert->notebook>=4.4.1->widgetsnbextension>=3.5.0->ipywidgets>=7.5.1->pandas-profiling) (0.5.0)
Requirement already satisfied: webencodings in c:\users\ingull\anaconda3\lib\site-packages (from bleach->nbconvert->notebook>=4.4.1->widgetsnbextension>=3.5.0->ipywidgets>=7.5.1->pandas-profiling) (0.5.1)
Installing collected packages: widgetsnbextension, ipywidgets, requests, pandas-profiling
  Attempting uninstall: ipywidgets
    Found existing installation: ipywidgets 7.4.2
    Uninstalling ipywidgets-7.4.2:
      Successfully uninstalled ipywidgets-7.4.2
  Attempting uninstall: requests
    Found existing installation: requests 2.21.0
    Uninstalling requests-2.21.0:
      Successfully uninstalled requests-2.21.0
Successfully installed ipywidgets-7.5.1 pandas-profiling-2.6.0 requests-2.23.0 widgetsnbextension-3.5.1
WARNING: You are using pip version 20.0.2; however, version 20.1 is available.
You should consider upgrading via the 'c:\users\ingull\anaconda3\python.exe -m pip install --upgrade pip' command.

```

[그림 11] Package가 성공적으로 설치되었을 때 출력되는 문구

- 앞서 언급했던 Package와 모듈을 불러오는 코드인 ‘from pandas_profiling import ProfiledReport’을 입력해보면 오류 없이 실행되는 모습을 볼 수 있다. Package와 모듈을 불러오는 것은 우선적으로 기능만을 불러오는 것이기 때문에 별도의 결과값이 표시되지 않는다. 제대로 실행되었을 경우, 코드 실행줄의 왼쪽을 보면 숫자가 표기된다. 추가로 언급하자면, [*] 표시는 코드가 실행중이며 아직 완료되지 않은 상태이고 오류 없이 [숫자]로 표시되면 코드가 정상적으로 실행 완료되었다는 의미이다.

```
In [3]: from pandas_profiling import ProfileReport
```

[그림 12] 필요 패키지 및 모듈을 불러오기 위한 코드

- 위의 과정에서 설명한 내용을 바탕으로 이번 실습에서 Package 설치를 해야 하는 라이브러리들은 다음과 같다.

```

matplotlib
scikit-learn
graphviz
seaborn
IPython
pydotplus
pandas
numpy

```

[그림 13] 본 실습을 위해 설치가 필요한 Package 목록

- Jupyter Notebook에서 !pip install 패키지명을 입력하여 위에서 열거한 모든 것을 설치해야 실습과정에서 에러가 발생하지 않고 실습을 진행할 수 있다.

```

!pip install matplotlib
!pip install scikit-learn
!pip install graphviz
!pip install seaborn
!pip install IPython
!pip install pydotplus
!pip install pandas
!pip install numpy

```

[그림 14] 본 실습을 위해 필요한 Package 설치 코드

```

Building wheel for pydotplus (setup.py): started
Building wheel for pydotplus (setup.py): finished with status 'done'
Created wheel for pydotplus: filename=pydotplus-2.0.2-py3-none-any.whl size=24572 sha256=9af9a
f44a62bec1f7bed738a12725077cef79dbd9265b05033ca9c1b4eb951c
Stored in directory: c:\users\█████\appdata\local\pip\cache\wheels\1f\5c\ba\f931f74fcac8f48b1
8ae597279203b1c1f921c76249c2b6f66
Successfully built pydotplus
Installing collected packages: pydotplus
Successfully installed pydotplus-2.0.2
Requirement already satisfied: pandas in d:\dev\anaconda3\envs\test\lib\site-packages (1.1.4)
Requirement already satisfied: python-dateutil>=2.7.3 in d:\dev\anaconda3\envs\test\lib\site-packages (from pandas) (2.8.1)
Requirement already satisfied: pytz>=2017.2 in d:\dev\anaconda3\envs\test\lib\site-packages (from pandas) (2020.4)
Requirement already satisfied: numpy>=1.15.4 in d:\dev\anaconda3\envs\test\lib\site-packages (from pandas) (1.19.4)
Requirement already satisfied: six>=1.5 in d:\dev\anaconda3\envs\test\lib\site-packages (from python-dateutil>=2.7.3->pandas) (1.15.0)
Requirement already satisfied: numpy in d:\dev\anaconda3\envs\test\lib\site-packages (1.19.4)

```

[그림 15] Package가 성공적으로 설치되었을 때 출력되는 문구

- 본 가이드북에서 사용하는 주요 라이브러리의 버전은 다음과 같다.

graphviz 0.15

pydotplus 2.0.2

①-3. 데이터 불러오기

```

path = "C:/Users/your_path"
os.chdir(path)

df = pd.read_csv("pasteurizer.csv")

```

[그림 16] 데이터를 불러오기 위한 코드

- pd.read_csv("데이터파일_경로명")을 통해 csv 형태의 데이터를 판다스(pandas)의 DataFrame으로 읽어 들인다. 이 예제에서는 csv 형태이기 때문에 read_csv()를 사용하였지만, excel, csv, hdf5 형식인 데이터도 불러올 수 있다. 이런 형식일 경우에는 read_excel(), read_csv(), read_hdf()를 사용하면 된다.

①-4. 데이터 확인

```
df
```

[그림 17] 데이터 내용 확인을 위한 코드

	STD_DT	MIXA_PASTEUR_STATE	MIXB_PASTEUR_STATE	MIXA_PASTEUR_TEMP	MIXB_PASTEUR_TEMP	INSP
0	2020-03-04 6:00	1.0	1.0	551.0	524.0	OK
1	2020-03-04 6:30	1.0	1.0	584.0	536.0	OK
2	2020-03-04 7:00	1.0	1.0	584.0	536.0	OK
3	2020-03-04 7:30	1.0	1.0	585.0	536.0	OK
4	2020-03-04 8:00	1.0	1.0	585.0	536.0	OK
...
210789	2020-11-11 18:34	NaN	NaN	586.0	566.0	OK
210790	2020-11-11 18:35	NaN	NaN	582.0	569.0	OK
210791	2020-11-11 18:36	NaN	NaN	577.0	572.0	OK
210792	2020-11-11 18:37	NaN	NaN	572.0	575.0	OK
210793	2020-11-11 18:38	NaN	NaN	568.0	576.0	OK

210794 rows × 6 columns

[그림 18] 데이터 내용 확인 결과

- 불러온 데이터프레임의 변수 df를 입력하면 해당 내용을 확인해 볼 수 있다. 데이터 양이 많기 때문에 제일 처음과 끝 5개씩만 나오고 있다. 만약 처음 n개 혹은 마지막 n개를 확인해보고 싶을 경우에는 아래와 같이 입력하면 된다.

```
df.head(20)
```

[그림 19] 데이터의 첫 n행 확인을 위한 코드 (n=20)

	STD_DT	MIXA_PASTEUR_STATE	MIXB_PASTEUR_STATE	MIXA_PASTEUR_TEMP	MIXB_PASTEUR_TEMP	INSP
0	2020-03-04 6:00	1.0	1.0	551.0	524.0	OK
1	2020-03-04 6:30	1.0	1.0	584.0	536.0	OK
2	2020-03-04 7:00	1.0	1.0	584.0	536.0	OK
3	2020-03-04 7:30	1.0	1.0	585.0	536.0	OK
4	2020-03-04 8:00	1.0	1.0	585.0	536.0	OK
5	2020-03-04 8:30	1.0	1.0	585.0	536.0	OK
6	2020-03-04 9:00	1.0	1.0	585.0	537.0	OK
7	2020-03-04 9:30	1.0	1.0	585.0	538.0	OK
8	2020-03-04 10:00	1.0	1.0	585.0	541.0	OK
9	2020-03-04 10:30	1.0	1.0	585.0	543.0	OK
10	2020-03-04 11:00	1.0	1.0	585.0	545.0	OK
11	2020-03-04 11:30	1.0	1.0	585.0	548.0	OK
12	2020-03-04 12:00	1.0	1.0	584.0	551.0	OK
13	2020-03-04 12:30	1.0	1.0	584.0	554.0	OK
14	2020-03-04 13:00	1.0	1.0	582.0	556.0	OK
16	2020-03-04 13:30	1.0	1.0	580.0	558.0	OK
17	2020-03-04 14:00	1.0	1.0	577.0	561.0	OK
18	2020-03-04 14:30	1.0	1.0	573.0	564.0	OK
19	2020-03-04 15:00	1.0	1.0	572.0	566.0	OK
20	2020-03-04 15:30	1.0	1.0	566.0	569.0	OK

[그림 20] 데이터의 첫 n행 확인 결과 (n=20)

- 처음 20개를 불러오는 코드이며, 팔호() 안에 개수를 입력하면 된다. 만약 숫자를 입력하지 않고 df.head()만 입력하면 기본값으로 5개만 나온다.

```
df.tail()
```

[그림 21] 데이터의 마지막 5행 확인을 위한 코드

	STD_DT	MIXA_PASTEUR_STATE	MIXB_PASTEUR_STATE	MIXA_PASTEUR_TEMP	MIXB_PASTEUR_TEMP	INSP
160703	2020-09-15 15:00		1.0	1.0	560.0	580.0 OK
160734	2020-09-15 15:30		1.0	1.0	588.0	555.0 OK
160765	2020-09-15 16:00		1.0	1.0	588.0	550.0 OK
160796	2020-09-15 16:30		1.0	1.0	595.0	555.0 OK
160827	2020-09-15 17:00		1.0	1.0	546.0	595.0 OK

[그림 22] 데이터의 마지막 5행 확인 결과

- 마지막 데이터를 가져오는 코드로 디폴트값이 5개만 출력된다.

[단계 ②] 데이터 종류 및 개수 확인

②-1. 칼럼명 확인

```
df.columns
```

[그림 23] 데이터의 칼럼명 확인을 위한 코드

```
Index(['STD_DT', 'MIXA_PASTEUR_STATE', 'MIXB_PASTEUR_STATE',
       'MIXA_PASTEUR_TEMP', 'MIXB_PASTEUR_TEMP', 'INSP'],
      dtype='object')
```

[그림 24] 데이터의 칼럼명 확인 결과

- 데이터프레임에 들어있는 칼럼명을 확인하는 코드로 ['STD_DT', 'MIXA_PASTEUR_STATE', 'MIXB_PASTEUR_STATE', 'MIXA_PASTEUR_TEMP', 'MIXB_PASTEUR_TEMP', 'INSP'] 총 6개의 칼럼을 확인할 수 있다.

②-2. 데이터 정보 확인

```
df.info()
```

[그림 25] 데이터의 기본 구조 확인을 위한 코드

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 210794 entries, 0 to 210793
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   STD_DT          210794 non-null   object  
 1   MIXA_PASTEUR_STATE 11135 non-null   float64 
 2   MIXB_PASTEUR_STATE 10255 non-null   float64 
 3   MIXA_PASTEUR_TEMP  201423 non-null   float64 
 4   MIXB_PASTEUR_TEMP  198802 non-null   float64 
 5   INSP             210794 non-null   object  
dtypes: float64(4), object(2)
memory usage: 9.6+ MB

```

[그림 26] 데이터 기본 구조 확인 결과

- 불리들인 데이터를 구성하고 있는 요소들의 칼럼명, 칼럼별 null이 아닌 데이터 개수, 데이터 타입을 확인할 수 있다. 이 데이터는 float64 형이 4개, object형이 2개라고 나온다.

②-3. 요약 통계량 확인

```
df.describe()
```

[그림 27] 데이터의 요약통계량 확인을 위한 코드

	MIXA_PASTEUR_STATE	MIXB_PASTEUR_STATE	MIXA_PASTEUR_TEMP	MIXB_PASTEUR_TEMP
count	1.113500e+04	10255.000000	201423.000000	1.988020e+05
mean	5.032693e+04	633.200390	566.867528	1.862568e+04
std	5.286901e+05	6408.270847	69.061703	8.111731e+05
min	0.000000e+00	0.000000	0.000000	0.000000e+00
25%	0.000000e+00	0.000000	543.000000	5.420000e+02
50%	1.000000e+00	1.000000	570.000000	5.690000e+02
75%	1.000000e+00	1.000000	596.000000	5.950000e+02
max	5.603841e+06	65536.000000	772.000000	4.279501e+07

[그림 28] 데이터의 요약통계량 확인 결과

- describe() 명령을 통해 다양한 요약 통계를 확인할 수 있다. 각 항목에 대한 설명은 다음과 같다.

* count : null이 아닌 행의 개수

* mean : 평균

* std : 표준편차

* min : 최소값

* 25% : 1/4 분위

* 50% : 중앙값

* 75% : 3/4 분위

* max : 최대값

②-4. 데이터 개수 확인

```
df.shape
```

[그림 29] 데이터의 크기(행, 열 개수) 확인을 위한 코드

(210794, 6)

[그림 30] 데이터의 크기(행, 열 개수) 확인 결과

- 데이터프레임의 크기는 shape 명령을 통해 (행, 열)의 개수를 확인할 수 있다. 이 데이터는 행이 210,794개이고 열이 6개인 데이터임을 보여준다.

②-5. 특정 칼럼 개수 확인

```
df['INSP'].value_counts()
```

[그림 31] 특정 칼럼의 각 값의 개수 확인을 위한 코드

```
OK    133010  
NG    77784  
Name: INSP, dtype: int64
```

[그림 32] 특정 칼럼의 각 값의 개수 확인 결과

- 특정 칼럼의 개수는 value_counts()를 통해서 확인할 수 있다. INSP는 OK와 NG로 표기된 값이 각각 133,010개, 77,784개 들어있다.

②-6. 칼럼별 null 개수 확인

```
df.isna().sum()
```

[그림 33] 각 칼럼의 null 개수 확인을 위한 코드

```
STD_DT          0  
MIXA_PASTEUR_STATE 199659  
MIXB_PASTEUR_STATE 200539  
MIXA_PASTEUR_TEMP   9371  
MIXB_PASTEUR_TEMP   11992  
INSP             0  
dtype: int64
```

[그림 34] 각 칼럼의 null 개수 확인 결과

- 각 칼럼에 포함되어 있는 null의 개수를 확인하기 위해 사용하는 코드로 상당히 많은 행들이 null 값이 포함된 것으로 보여진다. 다음 단계인 전처리 단계에서 이러한 값을 제거할 필요가 있다.

[단계 ③] 데이터 정제(전처리)

③-1. 결측치 제거

```
df = df.dropna()
```

[그림 35] null 값이 있는 행을 삭제하는 코드

- dropna()를 통해 null 값이 있는 행은 삭제한다.

③-2. 결측치 제거 결과 확인

```
df.isna().sum()
```

[그림 36] 각 칼럼의 null 개수 확인을 위한 코드

```
STD_DT          0  
MIXA_PASTEUR_STATE 0  
MIXB_PASTEUR_STATE 0  
MIXA_PASTEUR_TEMP   0  
MIXB_PASTEUR_TEMP   0  
INSP            0  
dtype: int64
```

[그림 37] 각 칼럼의 null 개수 확인 결과

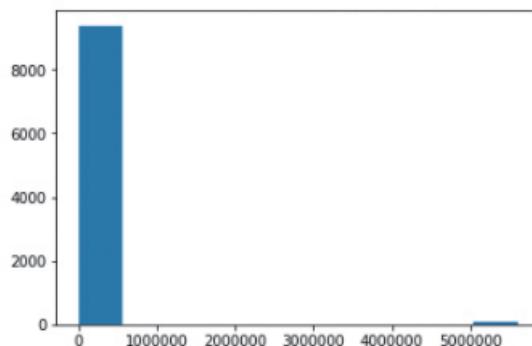
- 삭제가 제대로 이루어졌는지 확인하기 위해 다시 칼럼별 null 개수를 확인해보니 모두 0으로 나타나는 것을 확인할 수 있다.

③-3. 히스토그램 확인

```
plt.hist(df['MIXA_PASTEUR_STATE'])
```

[그림 38] 히스토그램을 그리기 위한 코드

```
(array([9383.,     0.,     0.,     0.,     0.,     0.,     0.,     0.,     0.,  
       100.]),  
 array([ 0. ,  560384.1, 1120768.2, 1681152.3, 2241536.4, 2801920.5,  
    3362304.6, 3922688.7, 4483072.8, 5043456.9, 5603841. ],  
<a list of 10 Patch objects>)
```



[그림 39] 히스토그램 그리기 결과

- 데이터의 분포, 중심 경향, 치우침 정도 등을 파악하기 위해 주로 사용하는 히스토그램을 통해 특정 칼럼의 데이터를 살펴볼 수 있다. 이 칼럼의 값은 9,383개의 행이 0~560384 사이에 분포하고, 나머지 100개가 5043456~5603841 사이에 분포하고 있는 것으로 나타난다.

③-4. 특정 칼럼의 unique 값 확인

```
df['MIXA_PASTEUR_STATE'].unique()
```

[그림 40] 특정 칼럼의 유일값 확인을 위한 코드

```
array([1.000000e+00, 5.603841e+06, 0.000000e+00])
```

[그림 41] 특정 칼럼의 유일값 확인 결과

- unique()를 통해 특정 칼럼의 유일값을 파악할 수 있다. MIXA_PASTEUR_STATE 칼럼에는 총 3개의 유일한 값이 존재하며, 그 값은 0, 1 그리고 5603841로 나타난다. 이 칼럼은 상태값을 나타내는데 0과 1이 아닌 다른 값들은 잘못된 값이기 때문에 분석에서 제거할 필요가 있다.

③-5. 특정 조건의 값 변경

```
df = df[df.MIXA_PASTEUR_STATE < 2]
```

[그림 42] 특정 조건에 해당하는 값만 저장하는 코드

- MIXA_PASTEUR_STATE 값이 0 또는 1인 행만 유지하기 위해 2 이하인 값들에 대해서만 다시 df에 넣는 코드이다.

```
df
```

[그림 43] 데이터 확인을 위한 코드

	STD_DT	MIXA_PASTEUR_STATE	MIXB_PASTEUR_STATE	MIXA_PASTEUR_TEMP	MIXB_PASTEUR_TEMP	INSP
0	2020-03-04 6:00	1.0	1.0	551.0	524.0	OK
1	2020-03-04 6:30	1.0	1.0	584.0	536.0	OK
2	2020-03-04 7:00	1.0	1.0	584.0	536.0	OK
3	2020-03-04 7:30	1.0	1.0	585.0	536.0	OK
4	2020-03-04 8:00	1.0	1.0	585.0	536.0	OK
...
160703	2020-09-15 15:00	1.0	1.0	560.0	580.0	OK
160734	2020-09-15 15:30	1.0	1.0	588.0	555.0	OK
160765	2020-09-15 16:00	1.0	1.0	588.0	550.0	OK
160796	2020-09-15 16:30	1.0	1.0	595.0	555.0	OK
160827	2020-09-15 17:00	1.0	1.0	546.0	595.0	OK

9383 rows × 6 columns

[그림 44] 데이터 확인 결과

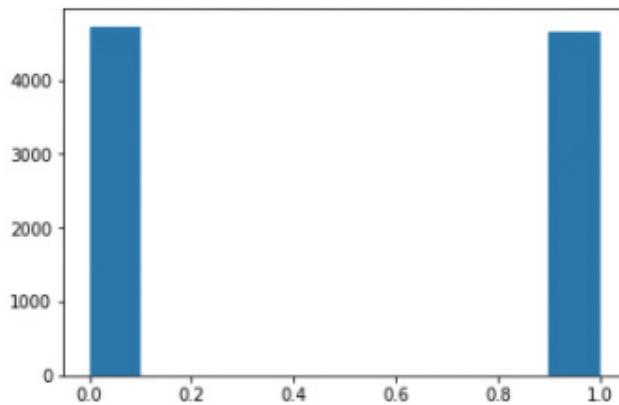
- 데이터프레임의 데이터를 확인해보면 이제 총 9,383개의 행이 존재하는 것을 확인 할 수 있다.

③-6. 다른 모든 칼럼의 히스토그램 확인

```
plt.hist(df['MIXA_PASTEUR_STATE'])
```

[그림 45] 특정 칼럼의 히스토그램을 그리기 위한 코드

```
(array([4729., 0., 0., 0., 0., 0., 0., 0., 0., 0.],  
      [4654.]),  
 array([0., 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.]),  
<a list of 10 Patch objects>)
```

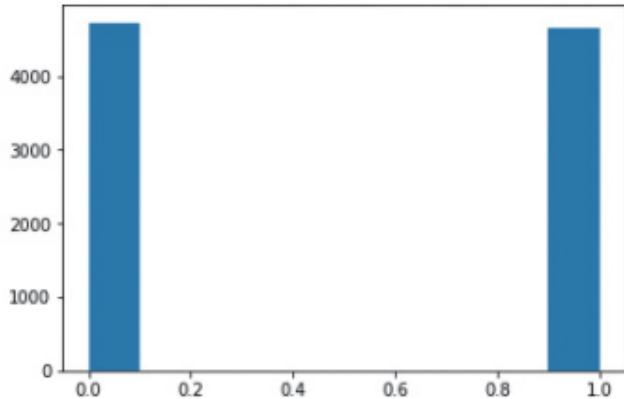


[그림 46] 특정 칼럼의 히스토그램 그리기 결과

```
plt.hist(df['MIXB_PASTEUR_STATE'])
```

[그림 47] : 특정 칼럼의 히스토그램을 그리기 위한 코드

```
(array([4729., 0., 0., 0., 0., 0., 0., 0., 0.,  
       4654.]),  
 array([0., 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.]),  
<a list of 10 Patch objects>)
```

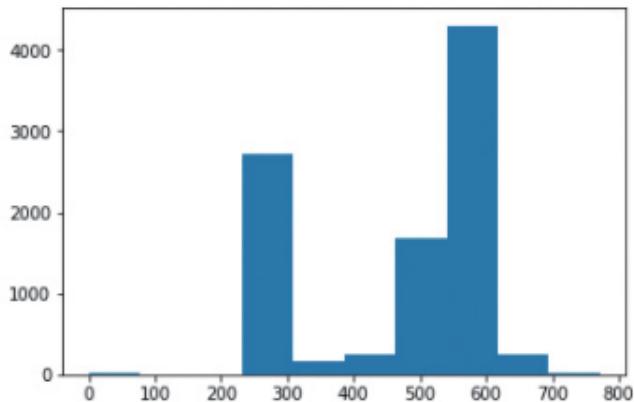


[그림 48] 특정 칼럼의 히스토그램 그리기 결과

```
plt.hist(df['MIXA_PASTEUR_TEMP'])
```

[그림 49] 특정 칼럼의 히스토그램을 그리기 위한 코드

```
(array([ 7., 0., 0., 2720., 158., 241., 1688., 4304., 238.,  
       27.]),  
 array([ 0., 77.2, 154.4, 231.6, 308.8, 386., 463.2, 540.4, 617.6,  
       694.8, 772.]),  
<a list of 10 Patch objects>)
```

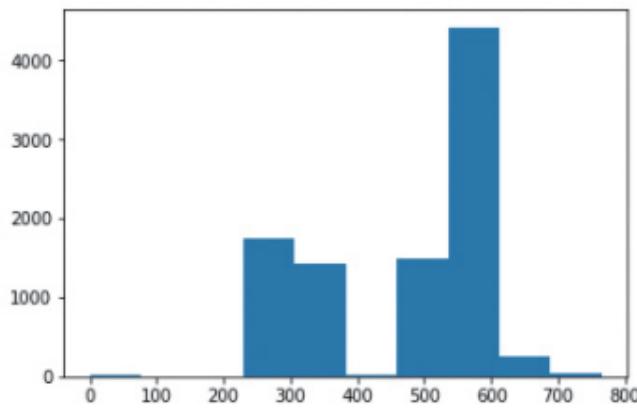


[그림 50] 특정 칼럼의 히스토그램 그리기 결과

```
plt.hist(df['MIXB_PASTEUR_TEMP'])
```

[그림 51] 특정 칼럼의 히스토그램을 그리기 위한 코드

```
(array([ 7., 0., 0., 1749., 1423., 10., 1487., 4418., 250.,
       39.]),
 array([ 0., 76.5, 153., 229.5, 306., 382.5, 459., 535.5, 612., 688.5, 765.]),
 <a list of 10 Patch objects>)
```



[그림 52] 특정 칼럼의 히스토그램 그리기 결과

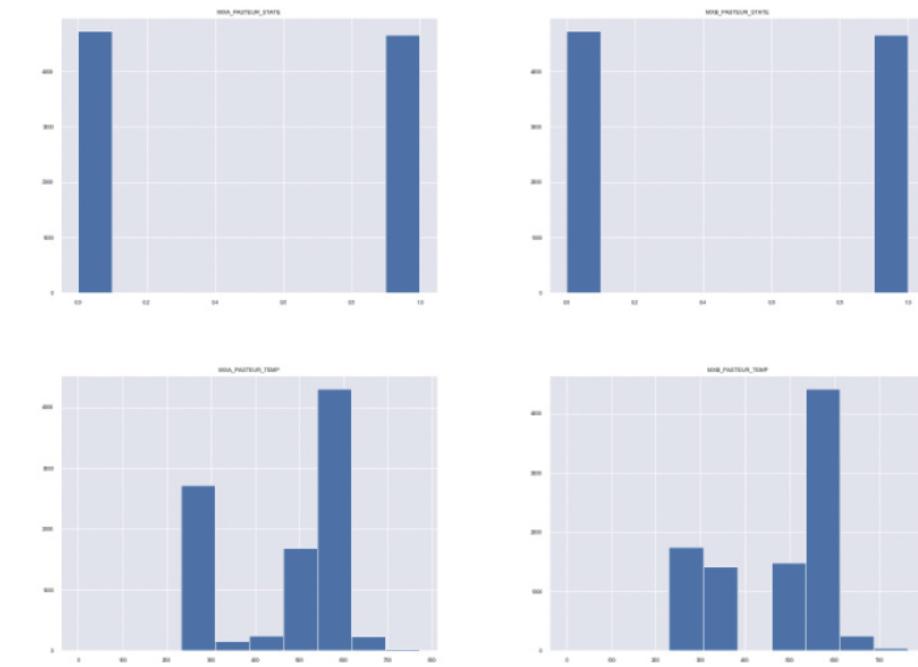
[단계 ④] 데이터 특성 파악

④-1. 요약 통계

```
df.hist(bins=10, figsize=(20, 15))
```

[그림 53] 모든 칼럼의 히스토그램을 한 번에 그리기 위한 코드

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x000002C3540CA438>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x000002C3544E7A58>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x000002C354514438>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x000002C354532E10>]],
      dtype=object)
```



[그림 54] 모든 칼럼의 히스토그램을 한 번에 그린 결과

- 데이터의 분포를 한눈에 볼 수 있는 코드로는 앞절에서는 특정 칼럼별로 확인을 하였는데, 위 코드와 같이 한눈에 볼 수도 있다. bins는 몇 개의 구간으로 나눌 것인지 를 지정하는 파라미터이다. 이 값이 커질수록 더 세분화된 그래프를 볼 수 있다.

④-2. 특성 간의 상관관계

```
df.corr()
```

[그림 55] 상관관계 분석 수행 코드

	MIXA_PASTEUR_STATE	MIXB_PASTEUR_STATE	MIXA_PASTEUR_TEMP	MIXB_PASTEUR_TEMP
MIXA_PASTEUR_STATE	1.000000	1.000000	0.523328	0.465706
MIXB_PASTEUR_STATE	1.000000	1.000000	0.523328	0.465706
MIXA_PASTEUR_TEMP	0.523328	0.523328	1.000000	0.920102
MIXB_PASTEUR_TEMP	0.465706	0.465706	0.920102	1.000000

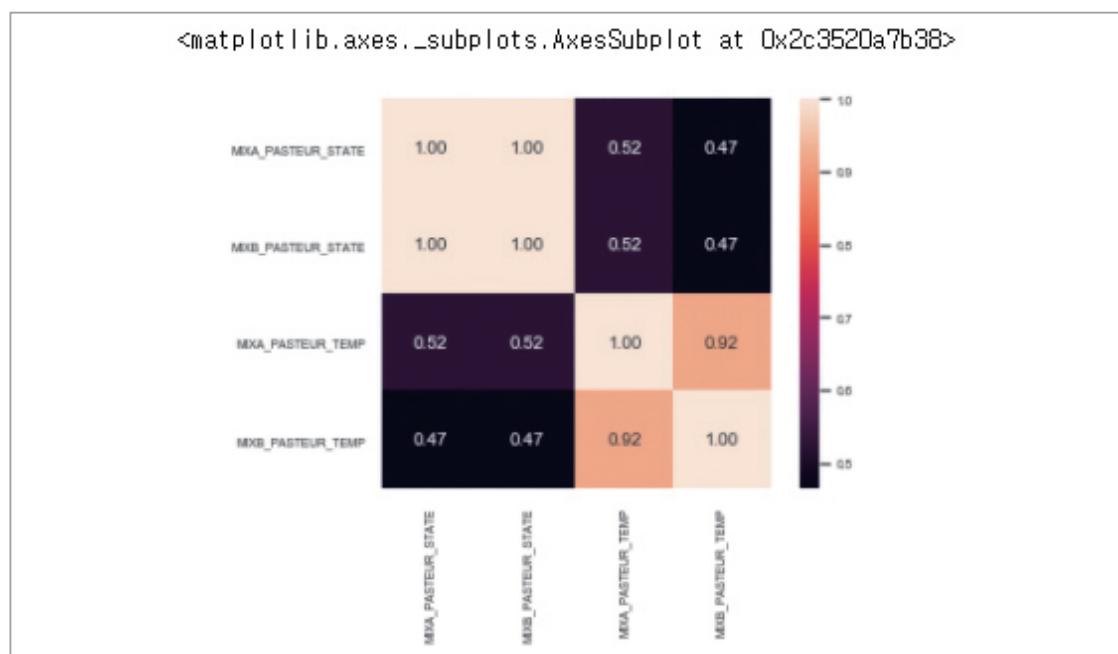
[그림 56] 상관관계 분석 수행 결과

- 4개의 특성 중 상관도가 제일 높은 특성은 온도를 나타내는 MIXA_PASTEUR_TEMP와 MIXB_PASTEUR_TEMP로 수치는 0.92 정도로 나타난다.

④-3. 히트맵

```
names = ['MIXA_PASTEUR_STATE', 'MIXB_PASTEUR_STATE', 'MIXA_PASTEUR_TEMP', 'MIXB_PASTEUR_TEMP']
cm = np.corrcoef(df[names].values.T)
sns.set(font_scale=0.6)
sns.heatmap(cm, cbar=True, annot=True, square=True, fmt='.2f', annot_kws={'size': 10}, yticklabels=names, xticklabels=names)
```

[그림 57] 히트맵을 그리기 위한 코드



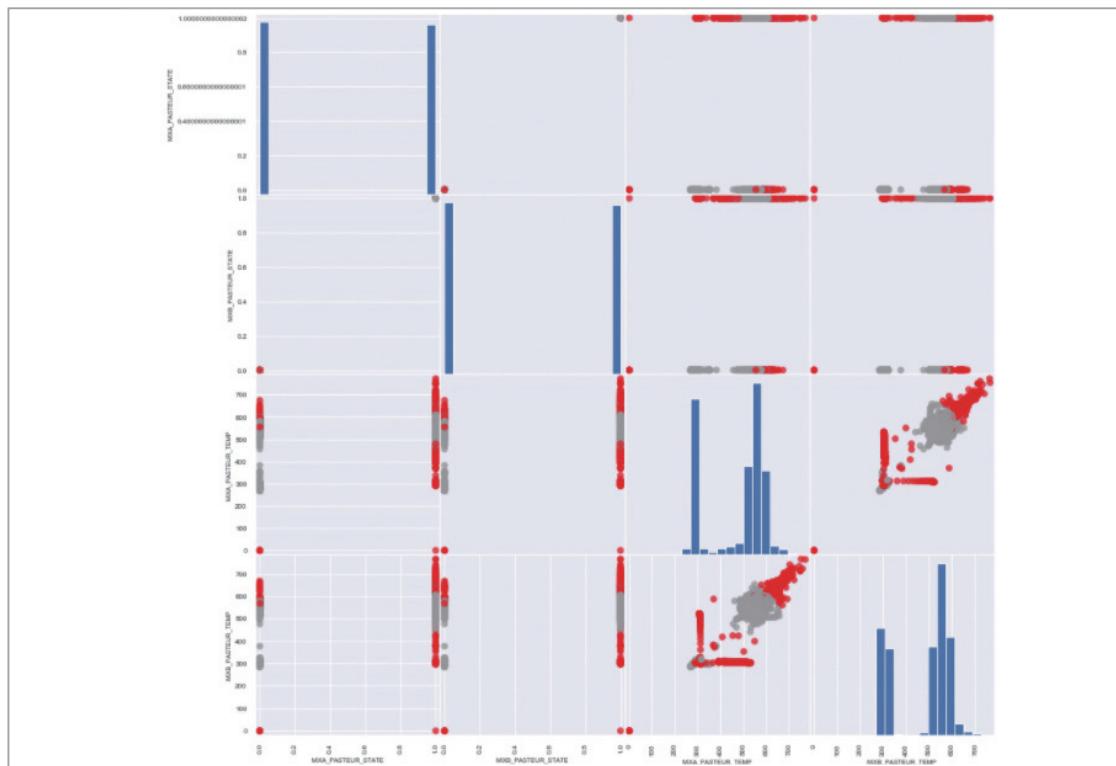
[그림 58] 히트맵을 그린 결과

- 히트맵을 통해서 특성 간의 상관관계를 나타낼 수 있다. 히트맵이란 열을 뜻하는 히트(heat)와 지도를 뜻하는 맵(map)을 결합한 단어로, 색상으로 표현할 수 있는 다양한 정보를 일정한 이미지 위에 열분포 형태의 그래프로 출력할 수 있다.³⁾
- 위에서 실행한 코드의 결과로 나온 히트맵은 4개의 특성을 간의 상관관계를 나타내는 것으로 검정색에서 연주황색으로 색상이 변할수록 상관관계가 높은 것을 의미한다. 4개의 특성 중 MIXA_PASTEUR_TEMP와 MIXB_PASTEUR_TEMP 특성이 정적 관계로 상관계수가 0.92이다. 4-2에서 구성한 상관행렬은 특성의 개수가 많아질수록 많은 숫자로 구성되기 때문에 특성들의 관계를 파악하는 것이 쉽지 않다. 따라서 이와 같이 상관행렬을 히트맵으로 만들게 되면 특성들 간의 관계를 쉽게 파악할 수 있으며, 이 때문에 특성들 간의 관련성을 알아보고자 하는 경우에 자주 쓰이는 방식 중 하나이다. 참고로 상관계수가 높다고 하여 인과관계가 있다고 볼 수는 없다.

④-4. 산점도 매트릭스

```
y = df.iloc[:, -1:].values
y = np.where(y == 'OK', 1, 0)
y = y.ravel()
pd.plotting.scatter_matrix(df, c=y, figsize=(15, 15), marker='o',
                           hist_kwds={'bins':20}, s=60, alpha=.8, cmap=plt.cm.Set1)
plt.show()
```

[그림 59] 산점도 매트릭스를 그리기 위한 코드



[그림 60] 산점도 매트릭스를 그린 결과

3) 참조: https://ko.wikipedia.org/wiki/히트_맵

- 산점도 매트릭스를 통해 특성 간의 관계를 일목요연하게 볼 수 있다. 산점도는 데이터를 x축과 y축에 점으로 표현한 그래프로서 두 특성 간의 관계를 표현할 때 사용 한다. 위 그래프에서는 양품인 경우 회색, 불량인 경우 빨간색 점으로 표현되고 있다. 특성들 간의 상관관계가 높은 MIXA_PASTEUR_TEMP와 MIXB_PASTEUR_TEMP의 분포는 양품인 경우 그래프(4행 3열에 위치한 서브그래프) 가운데 위치하고 있고, 불량인 경우는 빨간색 점과 같이 우측 상단과 ↘자 모형으로 패턴을 이루고 있는 것을 확인할 수 있다. 따라서 비교적 양품과 불량에 대한 구분이 특성 패턴을 이루고 있는 것을 짐작해 볼 수 있다.

[단계 ⑤] 학습/평가 데이터 분리

⑤-1. 학습/평가 데이터 분리

```
# 측정데이터와 레이블(정답)을 분리
X = df.iloc[:, 1:5].values
y = df.iloc[:, -1:].values
# print(y[-1:].reshape(-1)) # iloc으로 받아오면 numpy의 ndarray 형이다.
y = np.where(y == 'OK', 1, 0)
y = y.ravel() # 레이블을 1차원으로 변경하자.
print(X.shape, y.shape)
```

[그림 61] 측정데이터와 레이블(정답) 분리를 위한 코드

```
(9383, 4) (9383,)
```

[그림 62] 측정데이터와 레이블(정답) 분리 결과 확인

```
# 훈련셋과 테스트셋 분리
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)
```

[그림 63] 훈련 데이터와 테스트 데이터 분리를 위한 코드

```
X_train
```

[그림 64] 훈련 데이터 확인을 위한 코드

```
array([[ 0.,  0., 526., 584.],
       [ 0.,  0., 571., 526.],
       [ 0.,  0., 294., 314.],
       ...,
       [ 1.,  1., 543., 578.],
       [ 1.,  1., 594., 583.],
       [ 1.,  1., 582., 542.]])
```

[그림 65] 훈련 데이터 확인 결과

- 머신러닝 과정은 데이터의 학습과 데이터 검정의 단계로 나눠진다. 단계 ⑤에서는 전체 데이터를 학습용 데이터와 테스트 데이터로 분할한다. 먼저 전체 데이터를 독

립변수로 이뤄진 행렬과 종속변수로 이뤄진 행렬로 구분한다. 그다음, train_test_split을 활용하여 각 행렬을 학습용 데이터와 테스트 데이터로 분할한다.

[단계 ⑥] 의사결정나무 변수 설정 및 실행

⑥-1. 의사결정나무 알고리즘 불러오기

```
dt_clf = DecisionTreeClassifier(max_depth=3)
dt_clf = dt_clf.fit(X_train, y_train)
dt_prediction = dt_clf.predict(X_test)
```

[그림 66] 의사결정나무 알고리즘을 불러오기 위한 코드

- DecisionTreeClassifier()를 활용하여 의사결정나무 알고리즘을 불러오고, 해당 함수를 dt_clf라는 이름으로 설정한다. max_depth는 의사결정나무가 가지치는 깊이를 설정하는 함수로 크게 설정하면 가지를 많이 치게 되어 해석의 용이성이 떨어진다. 이번 샘플 데이터 분석에서는 max_depth를 3으로 설정하여 분석하였으나 이는 실제 데이터 분석 과정에서 조정이 가능하다.

⑥-2. 모델에 데이터 학습시키기

- dt_clf.fit을 사용하여 알고리즘에 데이터를 학습시킨다. 이때 학습시킬 데이터는 단계 ⑤에서 설정한 X_train, y_train이다.

⑥-3. 데이터 예측하기

- dt_clf.predict를 사용하여 훈련된 모델의 테스트를 하는 단계이다.

[단계 ⑦] 의사결정나무 시각화

⑦-1. 시각화를 위한 사전작업

```
feature_names = df.columns.tolist()
feature_names = feature_names[1:5]
target_name = np.array(['0', '1'])
```

[그림 67] 시각화 라이브러리 변수 생성을 위한 코드

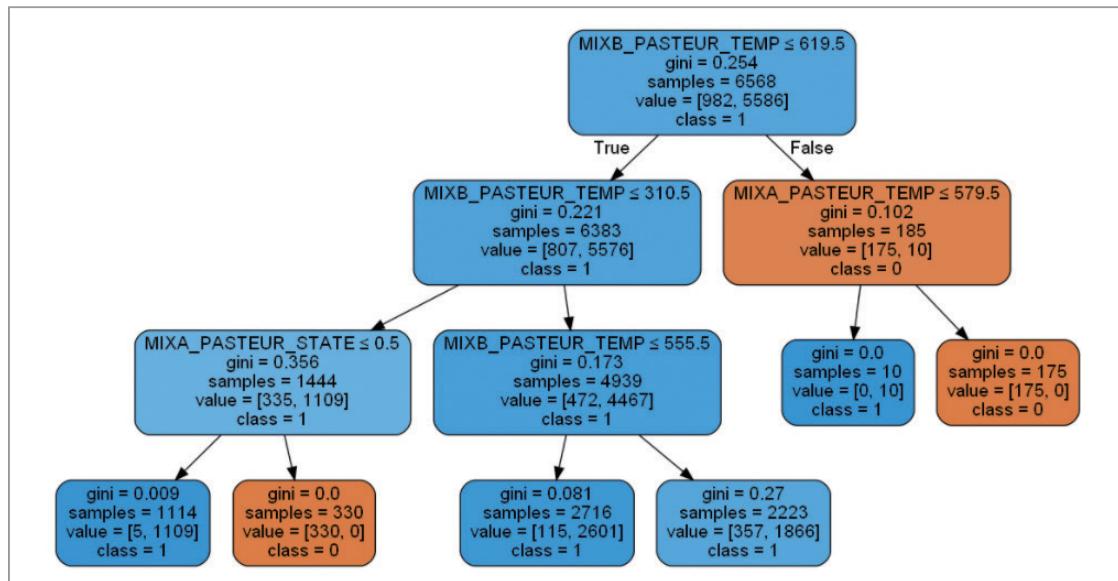
- 시각화 라이브러리에 삽입해야 하는 변수들을 생성하는 단계이다. 원본 데이터의 변수 이름을 추출하고 타겟 변수 이름을 0, 1로 설정한다.

⑦-2. Graphviz로 시각화하기

```
dt_dot_data = tree.export_graphviz(dt_clf,
    feature_names = feature_names,
    class_names = target_name,
    filled = True, rounded = True,
    special_characters = True)
dt_graph = pydotplus.graph_from_dot_data(dt_dot_data)
Image(dt_graph.create_png())
```

[그림 68] 의사결정나무 시각화 수행을 위한 코드

- `tree.export_graphviz()`는 의사결정나무 시각화를 위한 라이브러리이다. 이 라이브러리를 사용하여 가지치기가 어떻게 이루어졌는지 확인할 수 있다. 이 라이브러리의 파라미터는 다음과 같다.
 - * `feature_names(좌측), class_names` : 사전에 지정해준 변수명을 등호 옆에 입력해야 실행이 된다.
 - * `filled` : True일 시 노드들을 색칠한다(노드가 많을 경우 사용하는 게 적절). 아닐 시, False 대입
 - * `rounded`: 노드 박스 모서리를 둥글게 표현하고 싶을 시 True, 아닐 시 False
 - * `special_characters` : 변수명을 다 표현하고 싶을 시 True, 아닐 시 False
- 파라미터 설정이 끝나면 시각화를 도와주는 `pypdotplus.graph_from_dot_data`를 사용하여 시각화 작업을 마무리한다. 끝으로 `image()` 함수를 활용하여 시각화를 표출한다. 다음은 샘플 데이터로 의사결정나무를 시각화한 것이다.

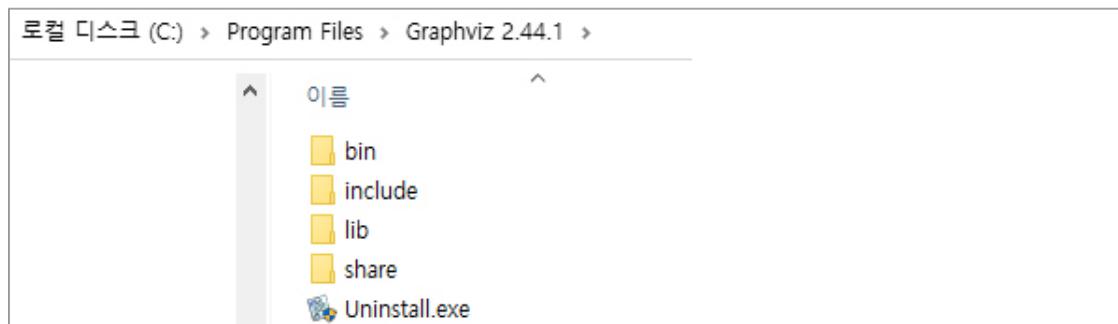


[그림 69] 의사결정나무 시각화 수행 결과

- 만약 의사결정나무 시각화 단계에서 오류가 발생할 경우, 다음과 같은 명령어를 입력하여 경로지정을 하면 문제를 해결할 수 있다. 다만, Graphviz가 설치된 경로는 사용자의 컴퓨터마다 다를 수 있으니, 설치된 위치를 확인 후 각자의 환경에 맞게 입력하여야 한다. 일반적으로 C:/Program Files 또는 C:/Program Files (x86) 폴더 안에 설치된다.

```
# 위의 코드에서 에러가 발생하지 않은 경우에는 이 라인을 실행할 필요가 없습니다.  
import os  
os.environ['PATH'] += os.pathsep + "C:/Program Files/Graphviz 2.44.1/bin/"
```

[그림 70] Graphviz 라이브러리 경로 지정을 위한 코드



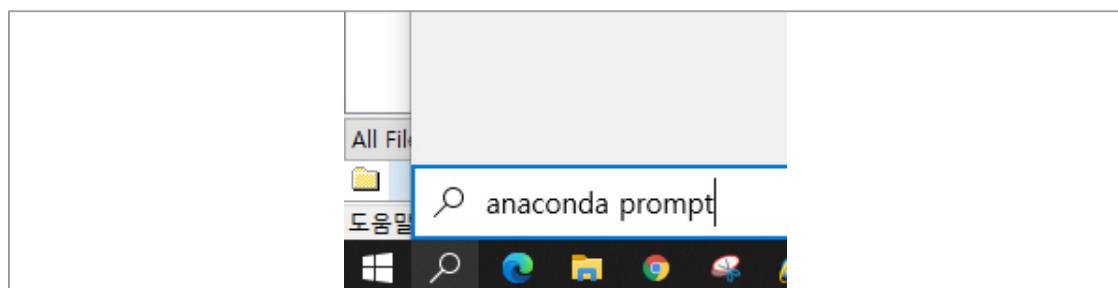
[그림 71] Graphviz 설치 위치 확인

- 위의 절차대로 진행하였음에도 아래와 같은 에러가 발생하는 경우가 있을 수 있다.

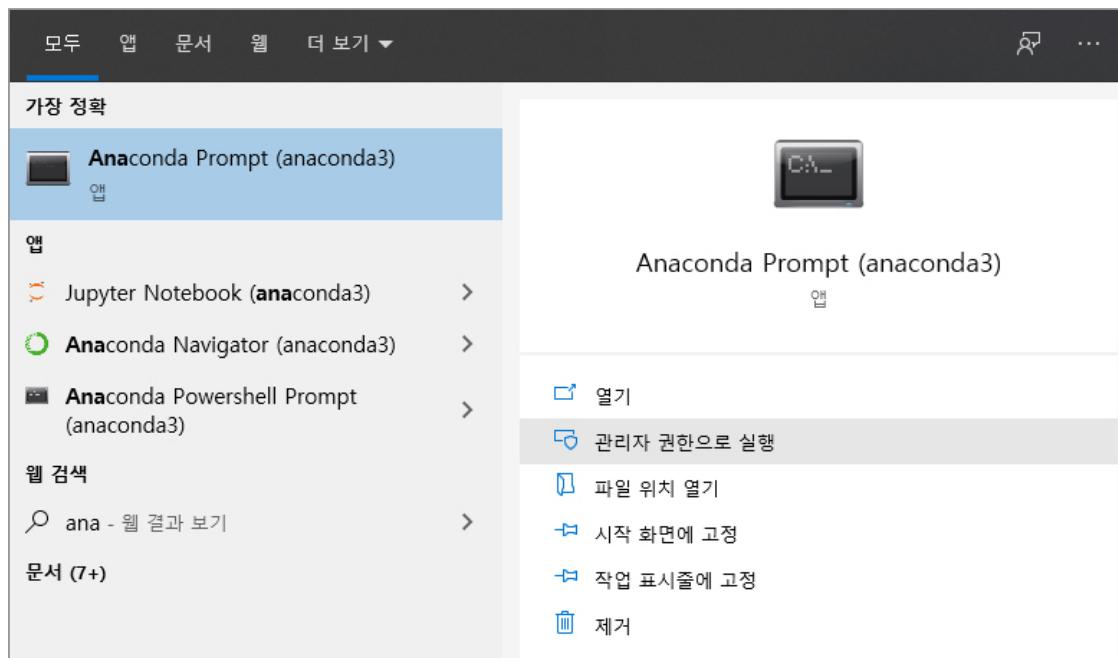
```
<Error>  
InvocationException : Program terminated with status: 1. stderr follows: Format:  
"png" not recognized. Use one of:
```

[그림 72] Graphviz 실행 오류 메시지

- 위와 같은 에러가 발생한 경우에는 Windows key를 누르고 “Anaconda Prompt”를 입력하여 관리자 권한으로 해당 앱을 실행한다.



[그림 73] Anaconda Prompt 검색



[그림 74] Anaconda Prompt 앱

- Anaconda Prompt가 실행되면 “dot -c”를 하단의 그림과 같이 입력하고 엔터를 누른다.



[그림 75] dot -c 실행 화면

- 엔터를 누르면 별다른 결과는 나오지 않으며, 다음으로 이전에 실행한 주피터 노트북을 종료하고 다시 노트북을 실행하여 처음부터 코드를 실행하면 된다.

[단계 ⑧] 결과 분석 및 해석

⑧-1. 분석 결과 확인하기 (Confusion Matrix)

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, roc_auc_score
from sklearn.metrics import f1_score, confusion_matrix, precision_recall_curve, roc_curve
```

[그림 76] 모델 평가를 위한 측정지표를 활용하기 위해 필요한 라이브러리를 불러오는 코드

- 의사결정나무의 분석 결과는 컴퓨터 매트릭스를 통해 확인할 수 있다. 이를 위해 필요한 라이브러리를 불러온다.

```

def get_clf_eval(y_test=None, pred=None):
    confusion = confusion_matrix(y_test, pred)
    accuracy = accuracy_score(y_test, pred)
    precision = precision_score(y_test, pred)
    recall = recall_score(y_test, pred)
    f1 = f1_score(y_test, pred)
    roc_auc = roc_auc_score(y_test, pred)
    print('오차 행렬')
    print(confusion)
    print('정확도: {:.4f}, 정밀도: {:.4f}, 재현율: {:.4f}, F1: {:.4f}, AUC: {:.4f}'.format(accuracy, precision, recall, f1, roc_auc))

get_clf_eval(y_test, dt_prediction)

```

[그림 77] get_clf_eval 함수 정의 및 생성을 위한 코드

- 그다음, get_clf_eval 함수를 입력한다. 이는 정확도, 정밀도, 재현율, f1 스코어 auc 점수, 오차행렬을 한 번에 불러오기 위한 함수이다. 함수를 생성하는 코드를 입력하고 결과를 출력하기 위한 get_clf_eval 함수를 넣으면 분석 결과에 대한 값을 한 번에 출력할 수 있다.

오차 행렬 [[237 196] [1 2381]] 정확도 : 0.9300, 정밀도: 0.9239, 재현율: 0.9996, F1 : 0.9603, AUC: 0.7735
--

[그림 78] 컨퓨전 매트릭스, 정확도, 정밀도, 재현율, F1-score, AUC 측정 결과

- 오차 행렬의 값은 시계방향으로 TP, FP, FN, TN을 의미한다. 그리고 이를 바탕으로 정확도, 정밀도, 재현율이 계산된다. Max_depth에 따라서 정확도 등 값이 개선되나 과적합이 발생하고 결과 해석에 용이하지 않다는 단점이 있어서 적절히 조정하는 것이 필요하다.

⑧-2. 의사결정나무 해석하기

- 시각화된 그래프를 살펴보면 class = 1(양품)로 분류된 기준이 무엇인지 확인할 수 있다. 따라서 클래스 1로 분류되는 가지들을 따라가면 양품의 규칙을 확인할 수 있다. 예를 들어, 루트 노드에서 MIXB_PASTEUR_TEMP가 619.5보다 작은 경우 양품으로 분류하였고, 초과하는 경우는 불량으로 분류하였음을 알 수 있다. 그다음, MIXB_PASTEUR_TEMP가 619.5를 초과하는 경우 중에서 MIXA_PASTEUR_TEMP가 579.5 이하인 경우는 양품으로 분류하였고, 초과하는 경우는 불량으로 분류하였다.
 - MIXB_PASTEUR_TEMP가 619.5 이하인 경우는 양품으로 분류가 되어 있긴 하지만, 지니 계수가 상대적으로 크기 때문에 max_depth를 높여서 가지치기를 이어나간다면 추가적인 규칙에 의해 분할될 가능성이 있다.
- 결과를 다시 정리하면 다음과 같다.

- 1) MIXB_PASTEUR_TEMP가 619.5를 초과하고, MIXA_PASTEUR_TEMP가 579.5 이하인 경우, 양품으로 분류
- 2) MIXB_PASTEUR_TEMP가 619.5를 초과하고, MIXA_PASTEUR_TEMP가 579.5 초과인 경우, 불량으로 분류
- 3) MIXB_PASTEUR_TEMP가 619.5를 이하인 경우, 양품으로 분류하나 추가 규칙에 의해 양품여부가 달라질 수도 있다.

⑧-3. 제조현장 관점에서 분석결과 해석하기

- 상기 의사결정나무 분석결과를 제조현장 관점에서 다시 해석하면,
 - 1) 살균기B의 살균온도는 61.95°C 이상, 살균기A의 살균온도는 57.95°C 이하로 운영하는 경우 양품 생산이 예측된다.
 - 2) 또한, 살균기B의 살균온도를 61.95°C 이하에서 운영할 때도 양품 생산이 가능하지만 보장되진 않는다.
- * 앞서 2.1절에서 설명한 바와 같이 MIXA_PASTEUR_TEMP와 MIXB_PASTEUR_TEMP 변수의 데이터는 실제로 소수점 1자리가 생략되어 있어 625는 62.5를 의미한다.
- 이와 같은 결과는 제품품질만 고려하였을 때의 설비운영조건이므로, 실제 운영시에는 CCP 이행기준을 고려하여 온도 범위를 설정하여야 한다. 즉, 예를 들어, 살균공정에 대한 CCP 기준이 55°C 이상이라면, 살균기A는 55~57.95°C, 살균기B는 61.95°C 이상의 조건 안에서 운영해야 양품을 생산할 수 있을 것이다.

3. 유사 타 현장의 「살균기 AI 데이터셋」 분석 적용

3.1 본 분석이 적용 가능한 제조현장 소개

- 공정중에 내부 온도변화를 예측하여 자동으로 설정값을 수시 변경할 수 있는 최첨단의 살균기를 사용하고 있지 않은 한, 생산라인에 살균공정을 포함하는 대부분의 중소기업 식품 제조업체에서는 본 가이드북에서 제시된 것과 동일한 문제를 안고 있을 것이다. 이와 같은 어려움이 있는 산업현장에서 만약 공정운영 데이터를 설비 PLC 또는 센서를 통해 확인 및 수집이 가능하고, 품질검사 결과를 전산화하여 관리하고 있다면 본 분석은 적용 가능하다.

3.2 본 「살균기 AI 데이터셋」 분석을 원용하여 타 제조현장 적용시, 주요 고려사항

- 생산제품 및 투입원료에 따라 분석 결과가 달라지기 때문에 해당 제품의 데이터로 분석 모델을 다시 학습시켜야 한다.
- 살균공정에서 측정되는 다른 변수(공정데이터, 환경데이터, 품질데이터 등)들도 포함하여 모델을 학습시켜 사용할 경우 더 정확한 모델을 생성할 수도 있다.
- 본 분석에서 활용한 특성값은 타 제조현장의 상황과 맞지 않을 수 있기 때문에 현장 전문가의 의견을 반영하여 적용 여부를 결정해야 한다.

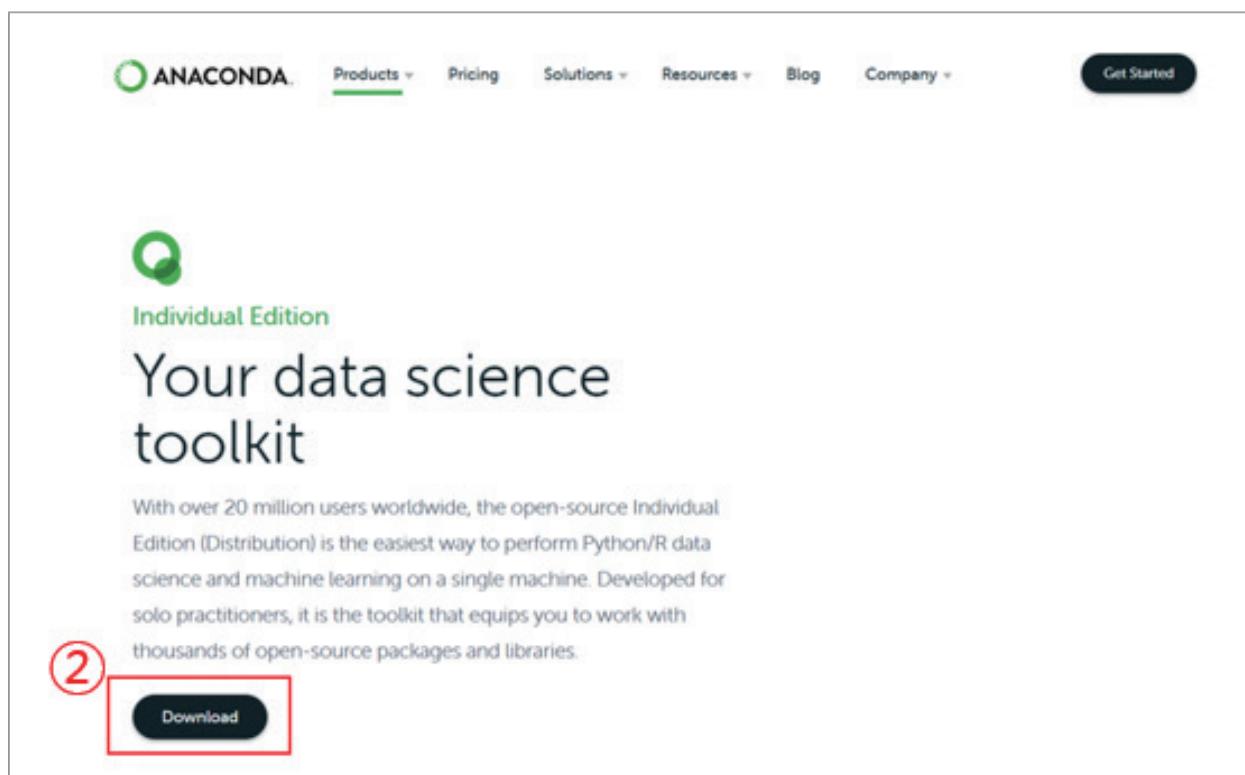
● Python을 사용하는 이유

- 데이터를 분석 및 예측하기 위해 Python 프로그래밍 언어를 사용하는 이유는 다음과 같다.
첫째, 비교적 읽고 쓰기 쉬운 프로그래밍 언어이며, 둘째, 효율적인 메모리 관리 기능을 갖추고 있고, 셋째, 머신러닝 프레임워크와 라이브러리가 풍부하다는 장점이 있다.

● Python 플랫폼 : Anaconda

- 먼저 Python 플랫폼인 Anaconda를 설치한다. Anaconda는 핵심적인 과학 및 수학 Python 라이브러리를 포함한 패키지이며, 머신러닝에 필수적인 툴이다. 쉽게 말해서, Anaconda를 설치하면 분석과 예측을 수행하기 위해 필요한 다양한 패키지들을 쉽게 설치할 수 있으며, 코드를 직접 입력하고 실행시킬 ‘개발환경’을 Anaconda Navigator라는 하나의 통합된 공간에서 편리하게 설치할 수 있다. 분석에 들어가기에 앞서 다음 순서에 맞춰 Anaconda를 설치한다.

● Anaconda Installer 설치

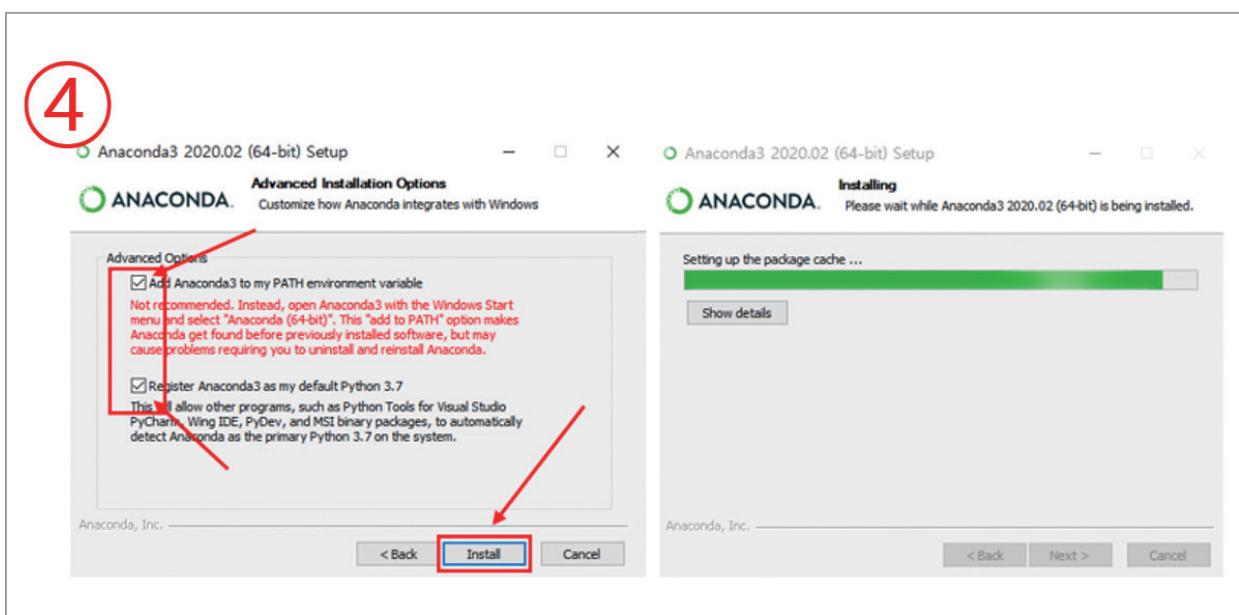
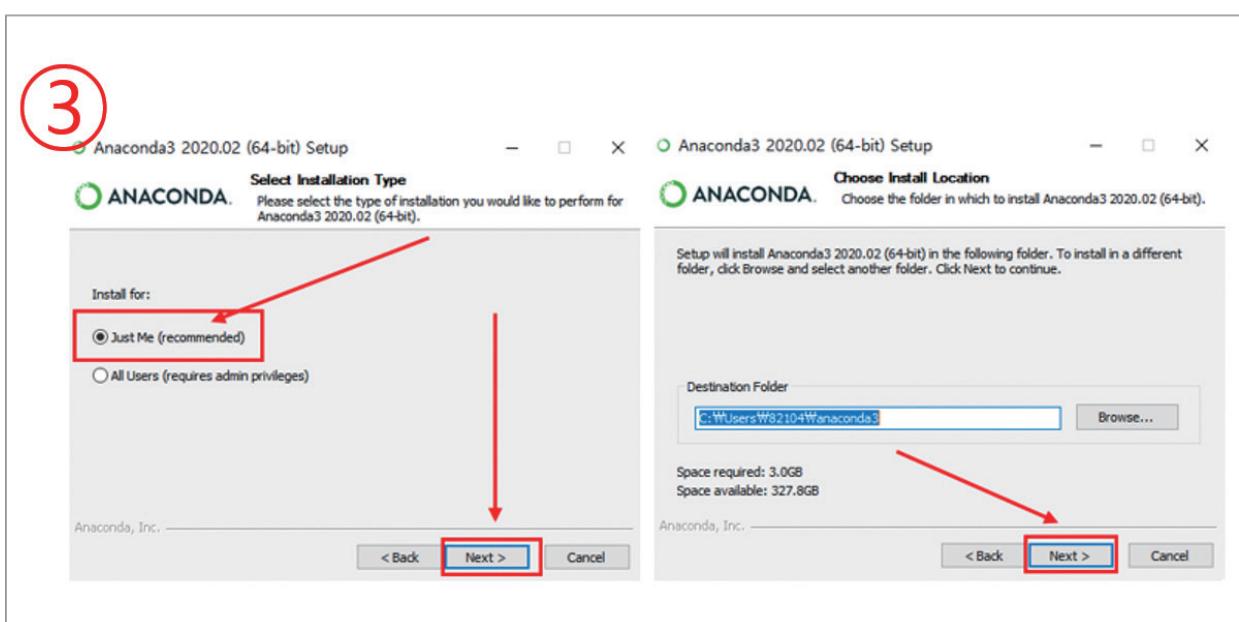
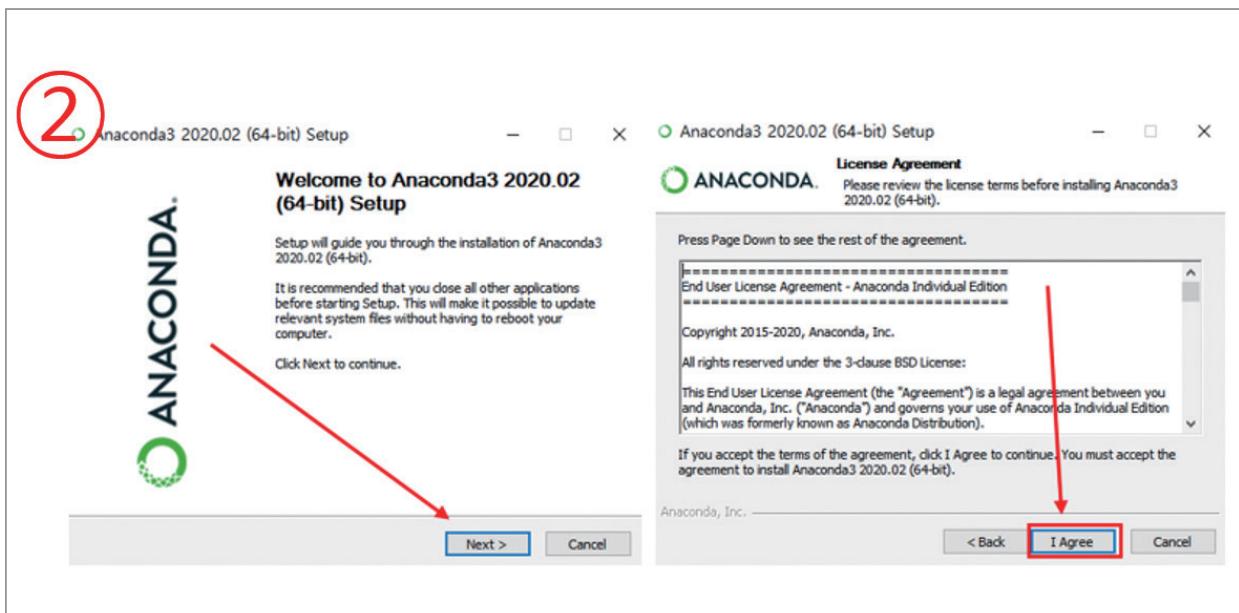


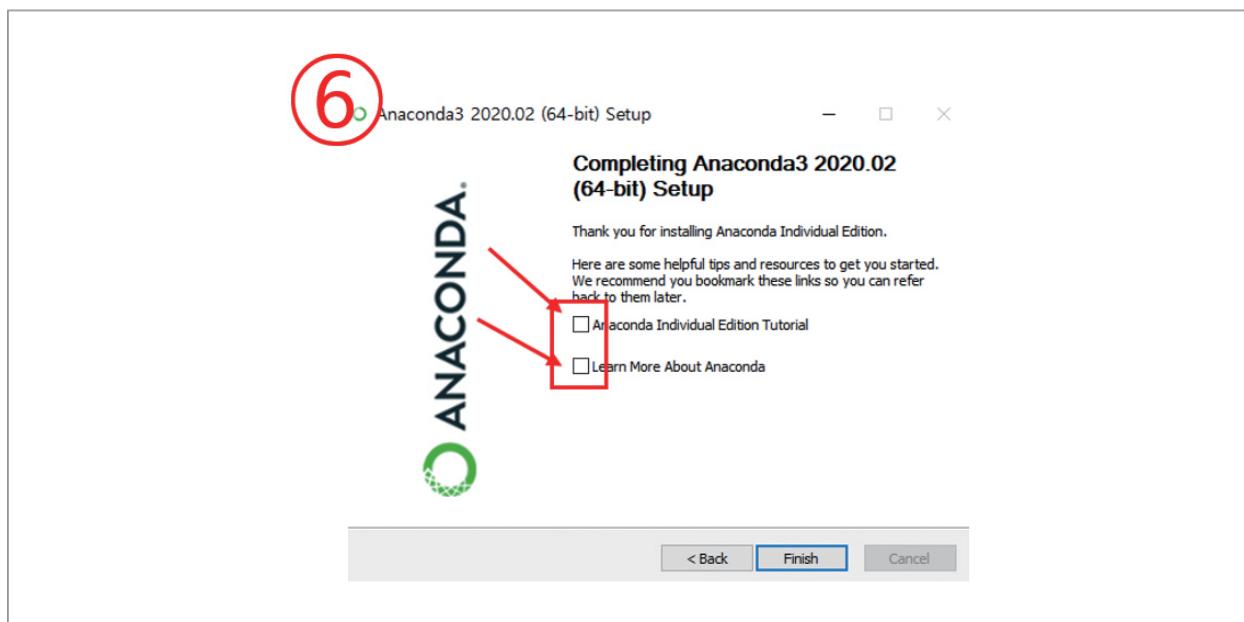
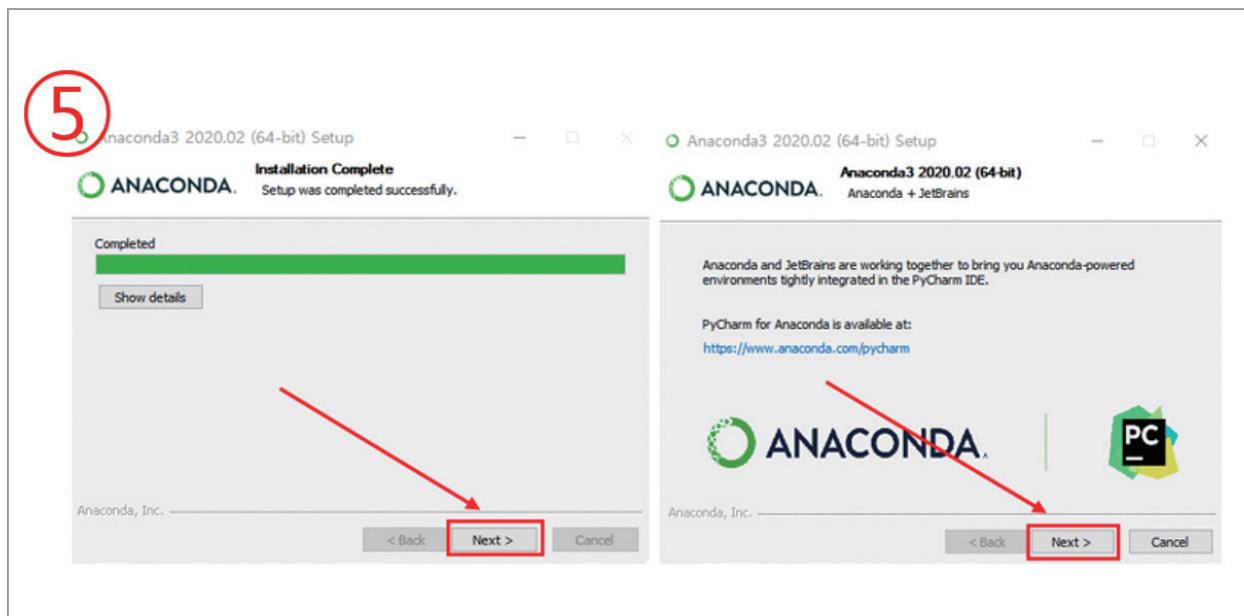


- ① www.anaconda.com/download 주소로 접속한다.
- ② 'Download' 버튼을 클릭한다.
- ③ Windows, MacOS, Linux 버전 중 자신의 PC 운영체제와 일치하는 버전을 선택하여 Installer를 다운로드 한다.

◉ Anaconda 실행파일 설치(Windows 64-bit 기준)

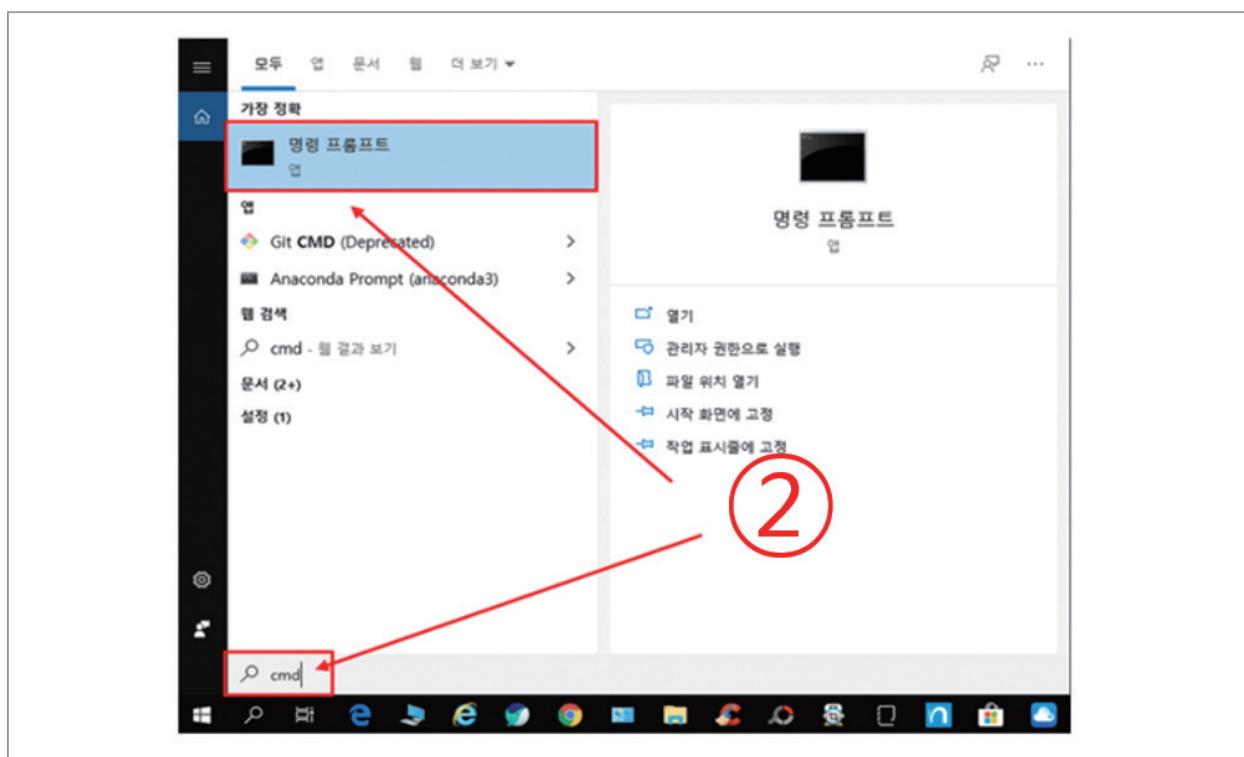
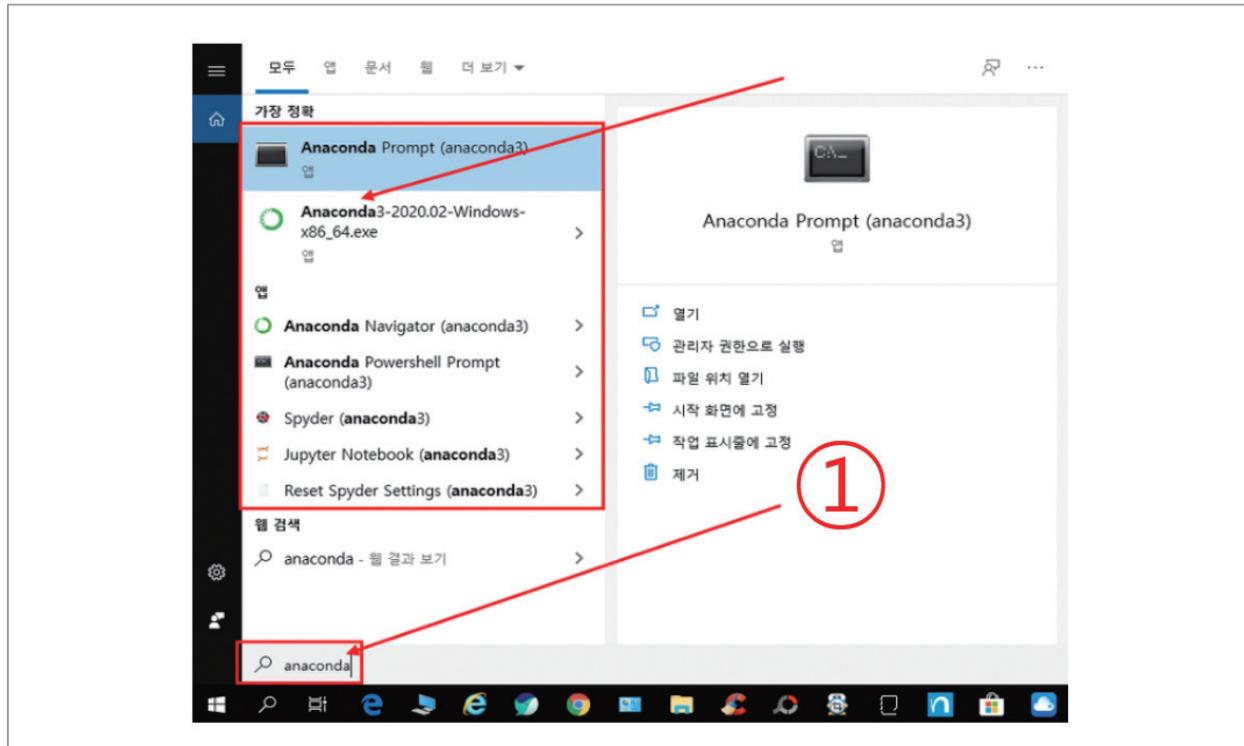




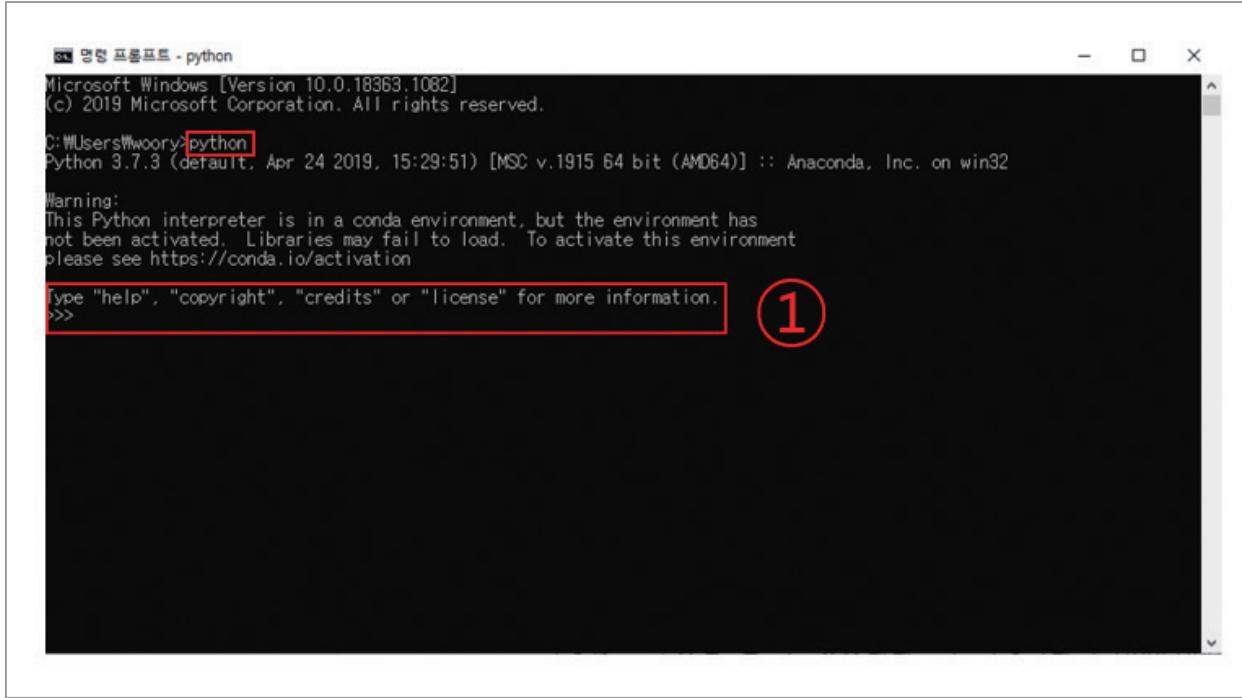


- ① 설치된 실행 파일을 열 때는 [마우스 오른쪽]-[관리자 권한으로 실행]을 클릭한다.
- ② 별도의 설정 없이 Next 버튼을 눌러 다음 단계로 진행한다.
- ③ 'Just me(recommended)'를 체크하고 넘어가서 다운로드 경로는 별도로 설정하지 않고 바로 넘어간다.
- ④ 체크박스를 모두 체크한 후 Install을 클릭한다.
- ⑤ 별도의 설정 없이 Next 버튼을 눌러 진행한다.
- ⑥ 두 가지 체크박스는 Anaconda와 관련된 튜토리얼과 교육자료에 관한 것을 설치 및 연결할 것인지 묻는 창이므로 체크를 전부 해제하고 Finish를 클릭한다.

① Anaconda 설치 확인

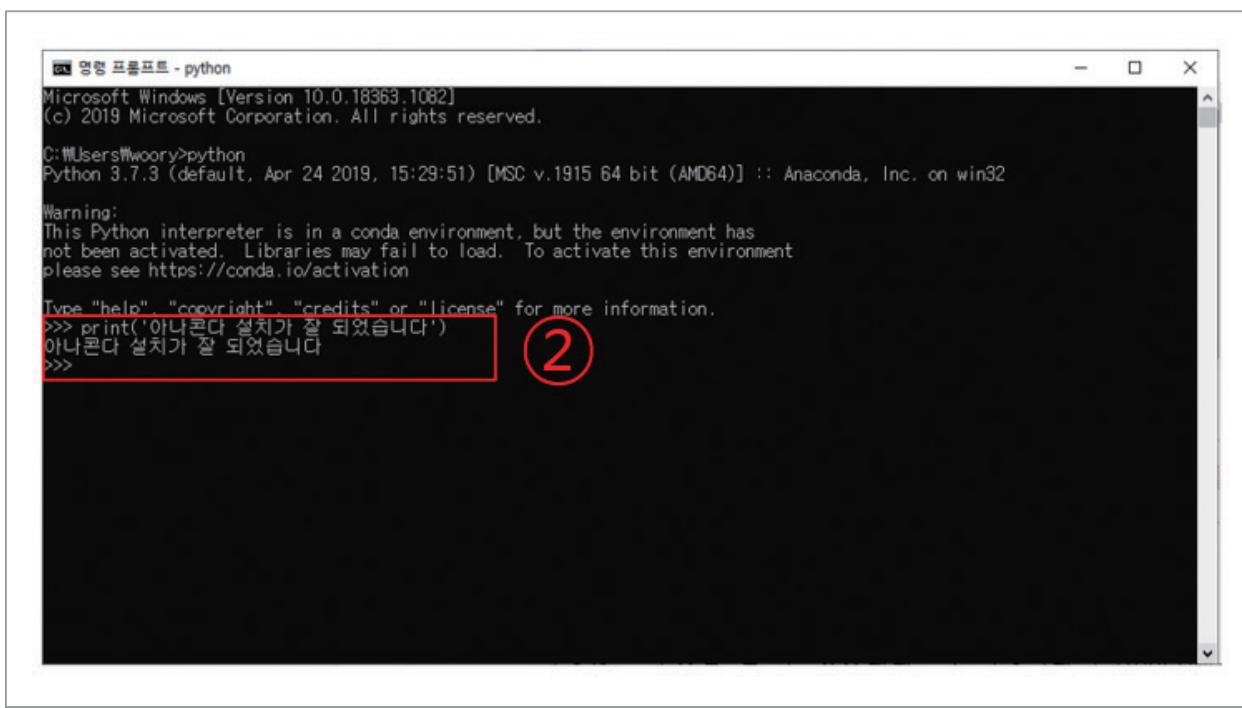


- ① 설치가 완료되면 Windows 좌측 하단의 돋보기를 클릭하여 Anaconda를 입력한 후 설치된 것을 확인한다.
- ② Anaconda를 통해 Python이 잘 설치되었는지 최종적으로 확인하기 위해 'cmd'를 입력한 후 '명령 프롬프트' 앱을 클릭한다.



```
C:\Users\woory>python
Microsoft Windows [Version 10.0.18363.1082]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\woory>python
Python 3.7.3 (default, Apr 24 2019, 15:29:51) [MSC v.1915 64 bit (AMD64)] :: Anaconda, Inc. on win32
Warning:
This Python interpreter is in a conda environment, but the environment has
not been activated. Libraries may fail to load. To activate this environment
please see https://conda.io/activation
Type "help", "copyright", "credits" or "license" for more information.
>>>
```



```
C:\Users\woory>python
Microsoft Windows [Version 10.0.18363.1082]
(c) 2019 Microsoft Corporation. All rights reserved.

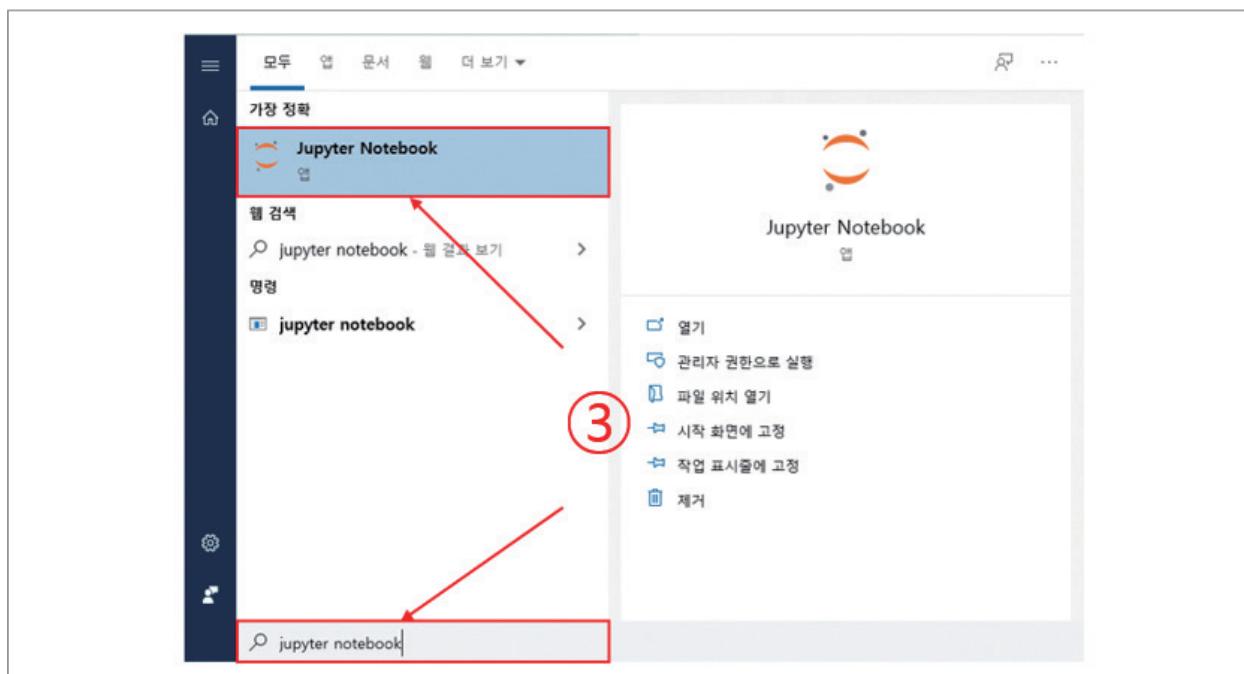
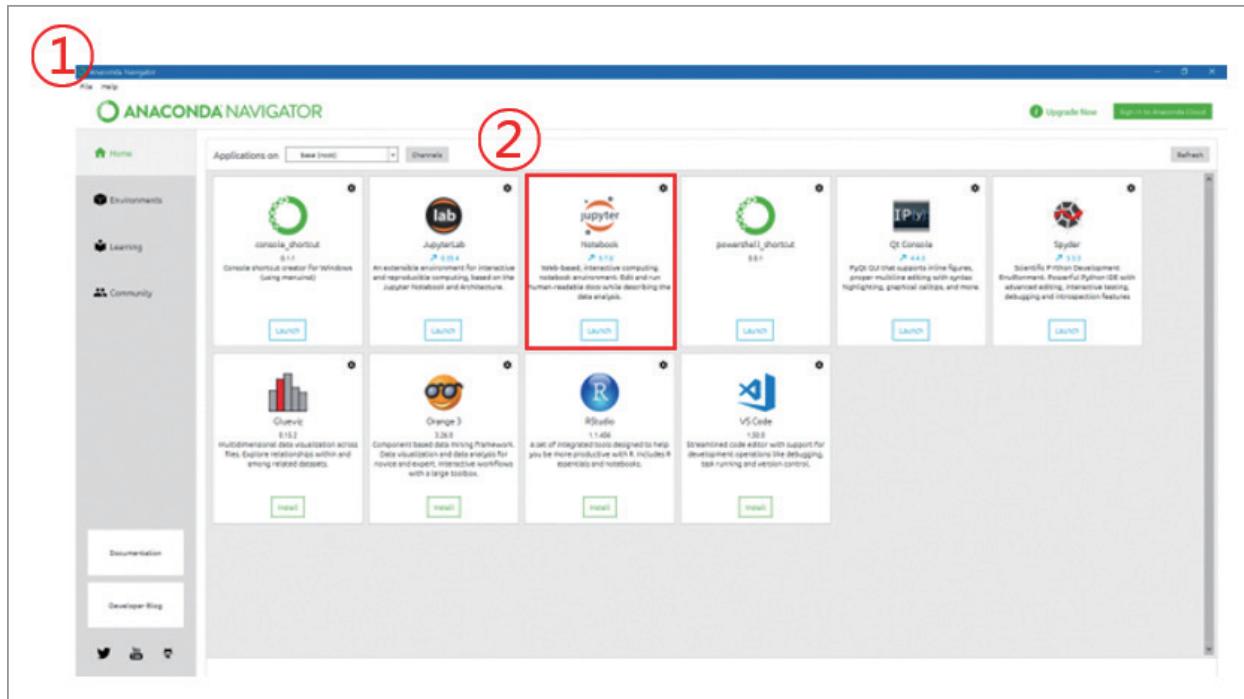
C:\Users\woory>python
Python 3.7.3 (default, Apr 24 2019, 15:29:51) [MSC v.1915 64 bit (AMD64)] :: Anaconda, Inc. on win32
Warning:
This Python interpreter is in a conda environment, but the environment has
not been activated. Libraries may fail to load. To activate this environment
please see https://conda.io/activation
Type "help", "copyright", "credits" or "license" for more information.
>>> print('아나콘다 설치가 잘 되었습니다.')
아나콘다 설치가 잘 되었습니다.
>>>
```

- ① 'python'을 입력한 후 Enter를 눌러 Python이 실행되는 모습을 확인한다.
- ② 최종적으로 코드 실행이 정상적으로 작동하는지 확인하기 위해 '>>>' 옆에 'print('아나콘다가 설치가 잘 되었습니다.')'를 입력하고 Enter를 눌러 해당 문자열이 그대로 화면에 출력되는 모습을 확인한다.

◎ Python 개발환경 : Jupyter Notebook

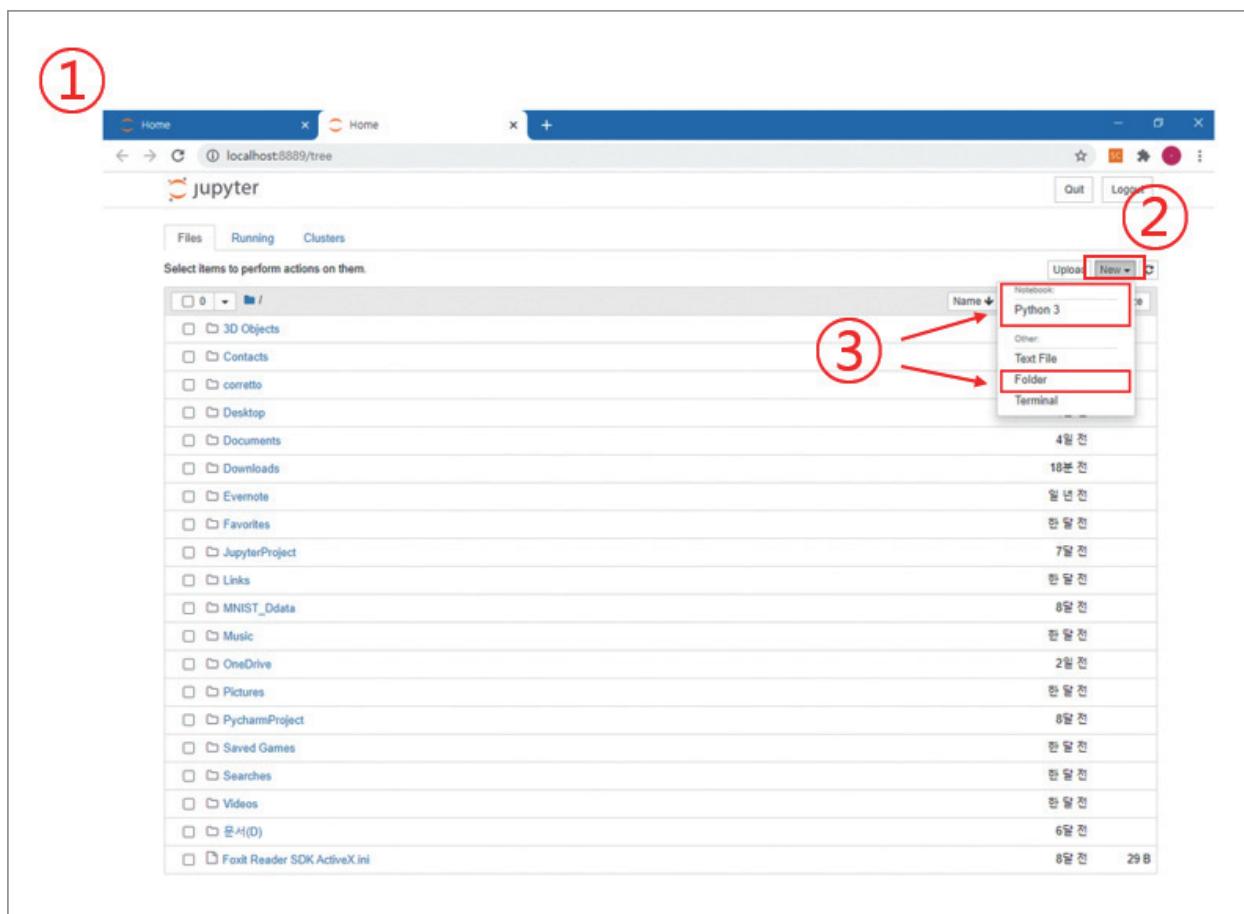
- Jupyter Notebook이란 ‘컴퓨터에게 Python 언어로 명령을 내렸을 때 즉각적인 응답이 오는 대화형 개발환경’이다. 여기서 말하는 ‘Notebook’은 일종의 ‘브라우저 웹페이지’ 형태이다. 즉, Jupyter Notebook이라는 웹페이지 상에서 Python 코드를 한 줄씩 입력하여 시행하고 분석 결과를 얻는다.

◎ Jupyter Notebook 설치



- ① Anaconda Navigator를 검색 후 실행한다.
- ② Jupyter Notebook이 'Launch' 상태인지 확인한다.
(※ 'Install' 상태일 경우, 클릭하여 설치해야 한다.)
- ③ Windows 화면 좌측 하단의 돋보기 버튼을 눌러 'jupyter notebook'을 입력하고 실행 앱이 있는지 확인한다.
(※ Anaconda Navigator를 매번 실행하지 않아도 Jupyter Notebook 앱을 통해 곧바로 실행이 가능하다.)

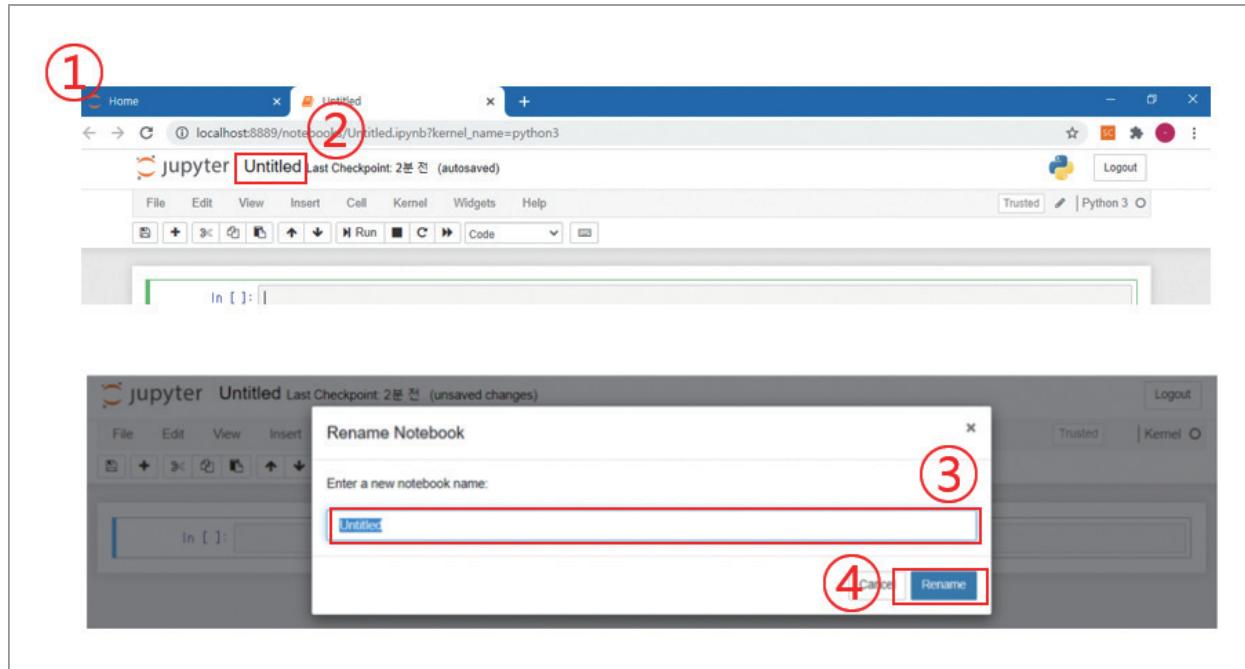
◉ Jupyter Notebook 실행 후 Notebook 생성



- ① Jupyter Notebook 앱을 클릭하여 웹페이지가 실행되는 것을 확인한다.
- ② notebook을 하나를 생성하기 위해 'New' 버튼을 클릭한다.
- ③ 현재 위치에 코드 파일 notebook을 바로 생성하려면 'Python 3'을 클릭하고, 새로운 폴더를 생성 후 그 안에 생성하려면 'Folder'를 클릭한다.
(※ 새로운 폴더 생성 후 진행할 경우, 그 폴더 안에서 [New]-[Python 3]를 동일하게 클릭한다.)

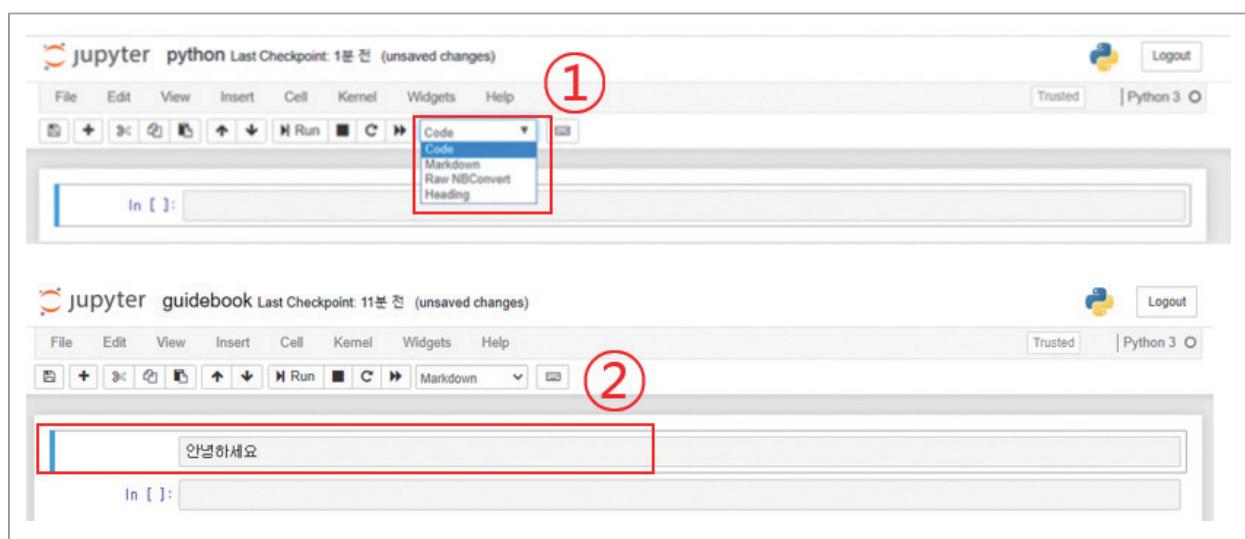
◉ Notebook 기본 활용법 : 제목 설정 및 주석 사용

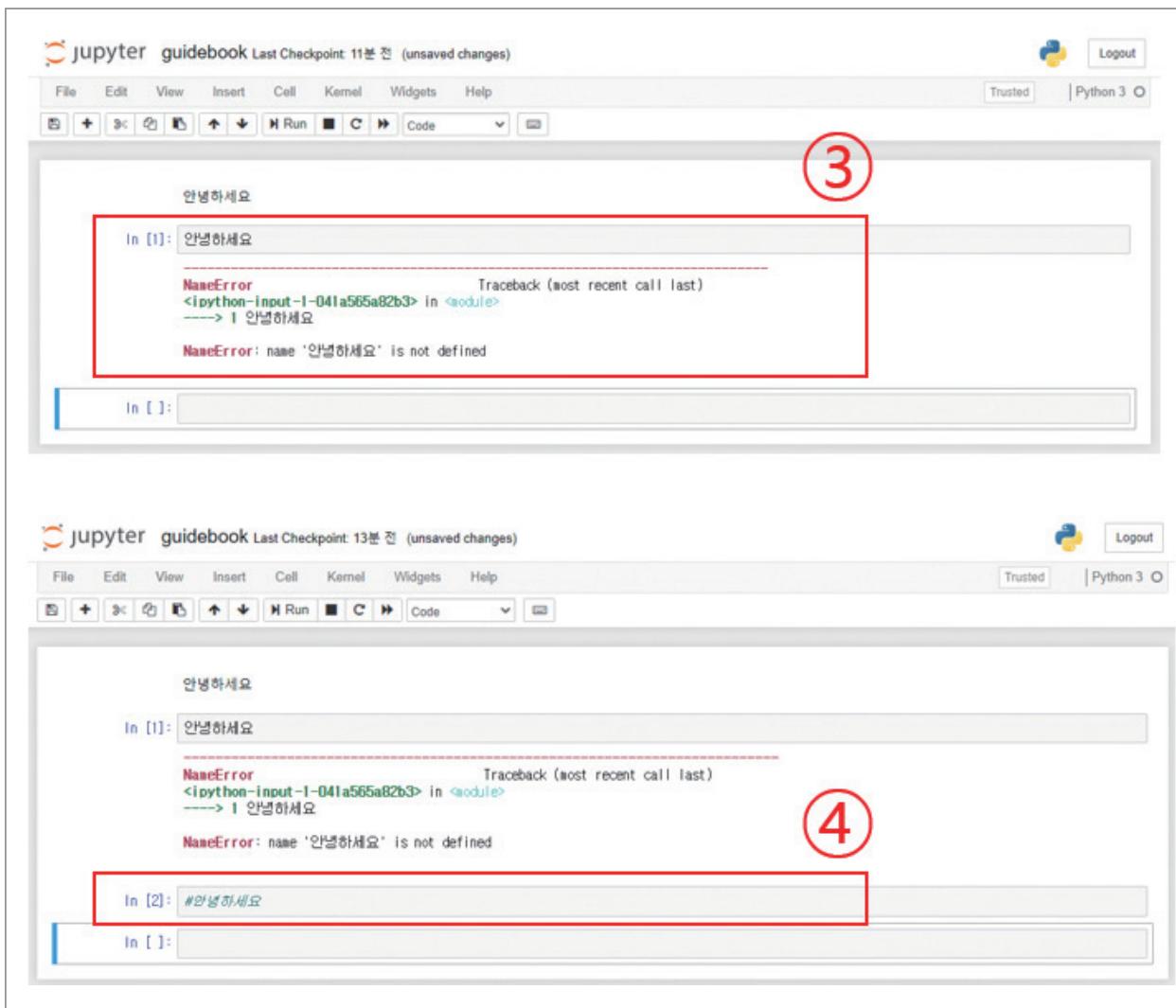
- 먼저 제목을 설정할 경우 다음과 같은 과정을 거친다.



- ① 새롭게 생성된 notebook을 연다.
- ② 처음 생성한 notebook의 제목을 확인하고 수정할 경우 제목을 두 번 클릭한다.
- ③ Rename Notebook 창에 수정하고자 하는 notebook의 제목을 입력한다.
- ④ 수정을 완료하기 위해 'Rename' 버튼을 클릭한다.

- 다음으로 코드 내에 주석(설명)을 기입할 경우 다음과 같은 과정을 거친다. ‘마크다운’(Markdown)을 이용하여 코드 실행 줄을 메모장 형식으로 변형하거나 이러한 변형 과정 없이 코드를 입력한 실행 줄 내에 곧바로 기입할 수 있다.





- ① 마크다운을 사용하기 위해 코드 실행 줄 하나를 선택해두고 ‘Code’로 설정되어 있는 버튼을 클릭하여 ‘Markdown’을 선택한다.
- ② 메모장 형식으로 코드 실행 줄이 변경된 것을 확인하고 주석 내용을 입력하여 Shift+Enter 키를 눌러 주석을 설정한다.
- ③ 마크다운으로 변경하지 않은 코드 실행 줄에 주석을 입력할 경우 오류가 발생하는 것을 확인 할 수 있다.
- ④ 마크다운 변경 없이 주석을 기입할 경우 ‘#’을 먼저 입력한 후 그 뒤에 설명을 기입한다.

● Notebook 기본 활용법 : 단축키

- 자주 활용하는 단축기는 해당 코드 줄을 클릭해놓은 상태에서 활용한다.

- ① Shift + Enter : 해당 코드 줄을 실행(Run)한다.
- ② m : 해당 코드 줄을 마크다운으로 자동 변경한다.
- ③ y : 해당 마크다운 줄을 코드 줄로 자동 변경한다.
- ④ a : 바로 위에 코드 줄을 추가한다.
- ⑤ b : 바로 아래에 코드 줄을 추가한다.
- ⑥ x : 해당 코드 줄을 삭제한다.

◎ 인스톨 패키지 다운받기

graphviz를 설치하기 위해서는 graphviz 홈페이지에서 별도의 인스톨러 파일을 다운로드하여 로컬 환경에 설치하여야 한다. 여기서는 Windows 64비트 버전을 예로 설명한다.

- ① graphviz 홈페이지(<https://graphviz.org/download/>)에 접속하여 [Download] 메뉴로 들어간다.

The screenshot shows the Graphviz website's download section. At the top, there is a navigation bar with links: About, Download, Gallery, Documentation, Theory and Publications, License, Resources, Credits, FAQ, Contact, and Issues/Bugs. Below the navigation bar, there is a large teal header with the text "Graphviz - Graph Visualization Software" and a small icon of a magnifying glass over a graph. The main content area has a white background and contains several sections: "Download" (highlighted in blue), "Source Code" (in dark blue), "Executable Packages" (in dark blue), and "Linux" (in dark blue). Under "Linux", it says: "We do not provide precompiled packages any more. You may find it useful to try one of the following third-party sites." At the bottom of the page, there is a footer note: "[그림 1] Graphviz 홈페이지의 Download 화면".

② Windows 환경을 위한 인스톨 패키지 메뉴를 찾아 링크를 따라 들어간다. 관련 링크는 아래의 그림과 같다.

The screenshot shows the official Graphviz website's navigation bar at the top, followed by a list of installation options for Windows. The 'Stable Windows install packages' link is highlighted with a red box. Below it, there are links for Cygwin Ports, WinGraphviz, Chocolatey packages, and the Windows Package Manager. Further down, instructions for building Graphviz on Windows are provided, along with links for Mac and Homebrew.

- [Development Windows install packages](#)
- [Stable Windows install packages](#)**
- [Cygwin Ports*](#) provides a port of Graphviz to Cygwin.
- [WinGraphviz*](#) Win32/COM object (dot/neato library for Visual Basic and ASP).
- [Chocolatey packages Graphviz for Windows.](#)
- `choco install graphviz`
- [Windows Package Manager](#) provides [Graphviz Windows packages](#).
- `winget install graphviz`

Mostly correct notes for building Graphviz on Windows can be found [here](#).

Mac

- [MacPorts*](#) provides both stable and development versions of Graphviz and the Mac GUI Graphviz.app. These can be obtained via the ports [graphviz](#), [graphviz-devel](#), [graphviz-gui](#) and [graphviz-gui-devel](#).
- `sudo port install graphviz`
- [Homebrew*](#) has a [Graphviz port](#).

[그림 2] 윈도우 환경에 맞는 인스톨 패키지 메뉴

③ 해당 메뉴를 선택하면 나오는 화면에서는 [10]이라는 메뉴를 선택한다.

The screenshot shows the 'Index of /Packages/stable/windows' directory listing. It contains two items: a link to the parent directory and a link to '10/'.

- [Parent Directory](#)
- [10/](#)

[그림 3] 사용자 환경에 맞는 인스톨러 다운로드 링크 선택(1)

④ 다음 화면에서는 [cmake] 링크를 선택한다.

Index of /Packages/stable/windows/10

- [Parent Directory](#)
- [cmake/](#)
- [msbuild/](#)

[그림 4] 사용자 환경에 맞는 인스톨러 다운로드 링크 선택(2)

⑤ 다음 화면에서는 [Release] 메뉴를 선택한다.

Index of /Packages/stable/windows/10/cmake

- [Parent Directory](#)
- [Release/](#)

[그림 5] 사용자 환경에 맞는 인스톨러 다운로드 링크 선택(3)

⑥ 다음 화면에서는 [x64] 메뉴를 선택한다. 만약 사용자의 Windows 환경이 32비트라면 [Win32]를 선택하면 된다.

Index of /Packages/stable/windows/10/cmake/Release

- [Parent Directory](#)
- [Win32/](#)
- [x64/](#)

[그림 6] 사용자 환경에 맞는 인스톨러 다운로드 링크 선택(4)

⑦ 다음 화면에서 graphviz-install-2.44.1-win64.exe를 선택하면 다운로드가 실행된다.

※ 현재 시점의 최신 버전은 2.44.1이며 버전은 변동될 수 있다.

Index of /Packages/stable/windows/10/cmake/Release/x64

- [Parent Directory](#)
- [graphviz-install-2.44.1-win64.exe](#)

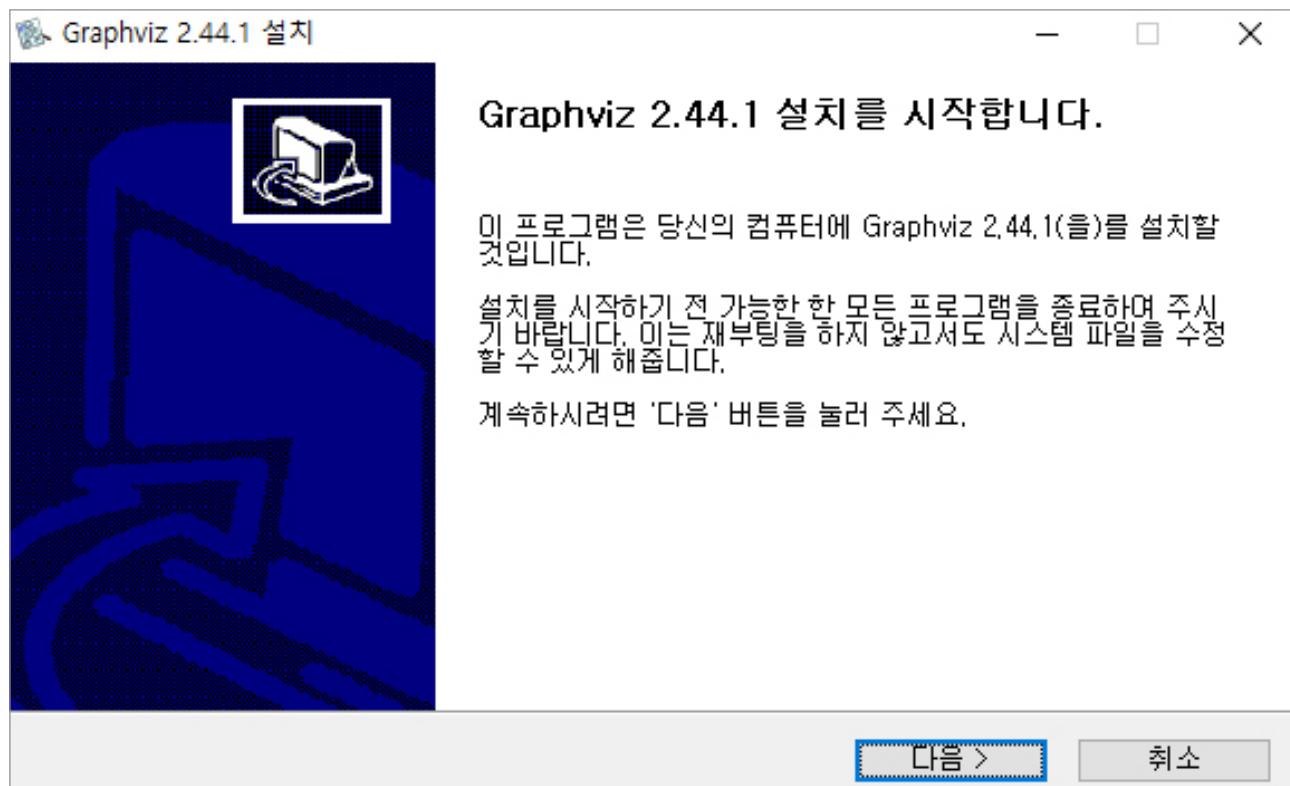
[그림 7] 사용자 환경에 맞는 인스톨러 다운로드 링크 선택(5)

⑧ 위 ①~⑦번의 순서에 따라 이동했을 때 도달하는 최종 URL은 아래와 같으며, 브라우저 검색창에 아래 URL을 입력하여 바로 해당 화면으로 이동하여도 된다.

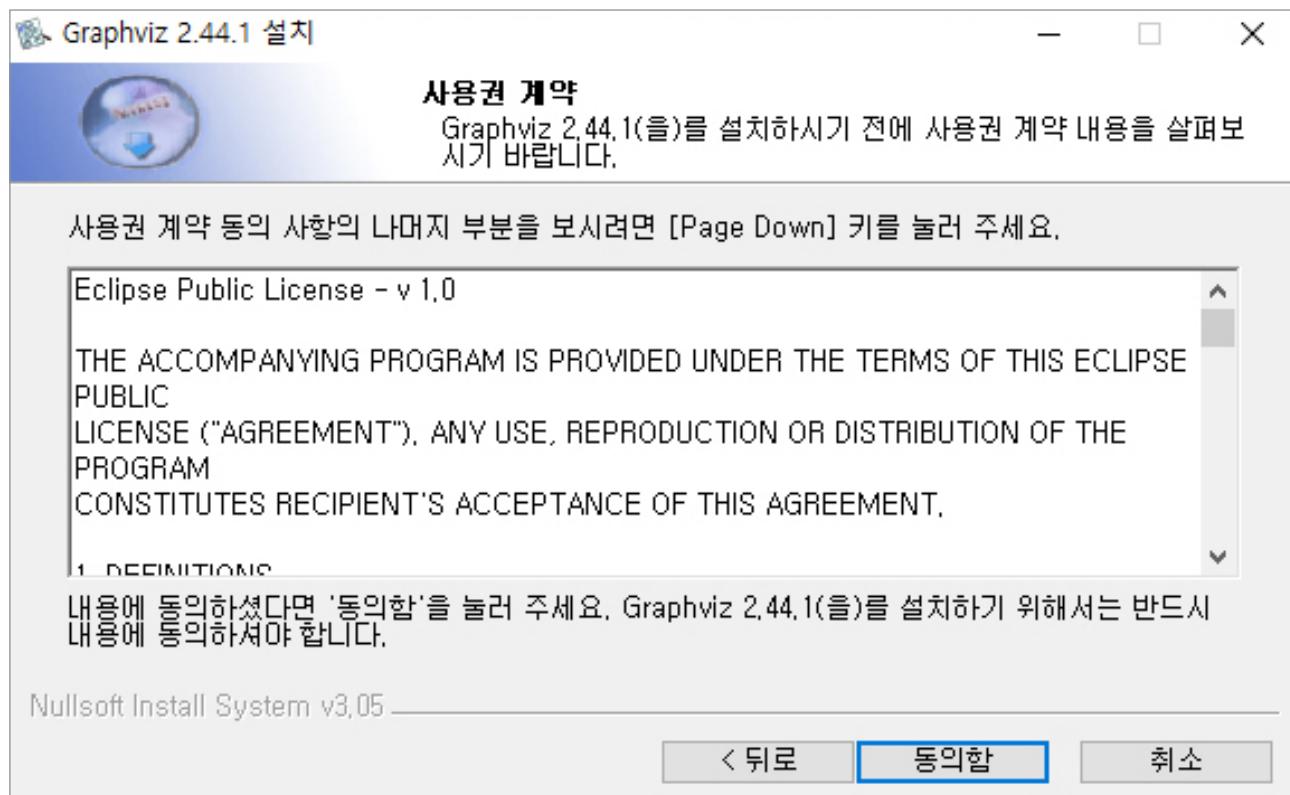
- 윈도우 64비트 : <https://www2.graphviz.org/Packages/stable/windows/10/cmake/Release/x64/>
- 윈도우 32비트 : <https://www2.graphviz.org/Packages/stable/windows/10/cmake/Release/Win32/>

◎ graphviz 설치하기

① 다운로드 받은 graphviz-install-2.44.1-win64.exe 파일을 실행하여 graphviz를 설치한다.

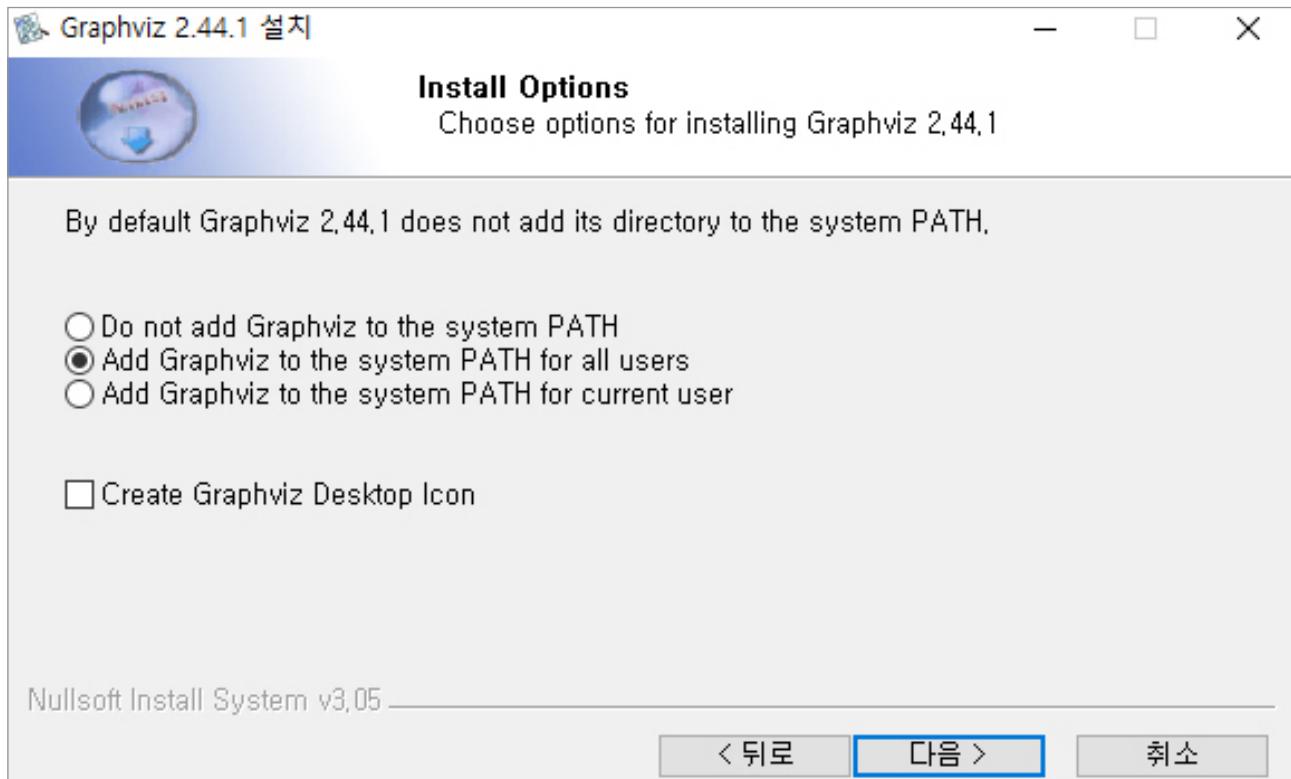


[그림 8] graphviz 설치 실행 화면



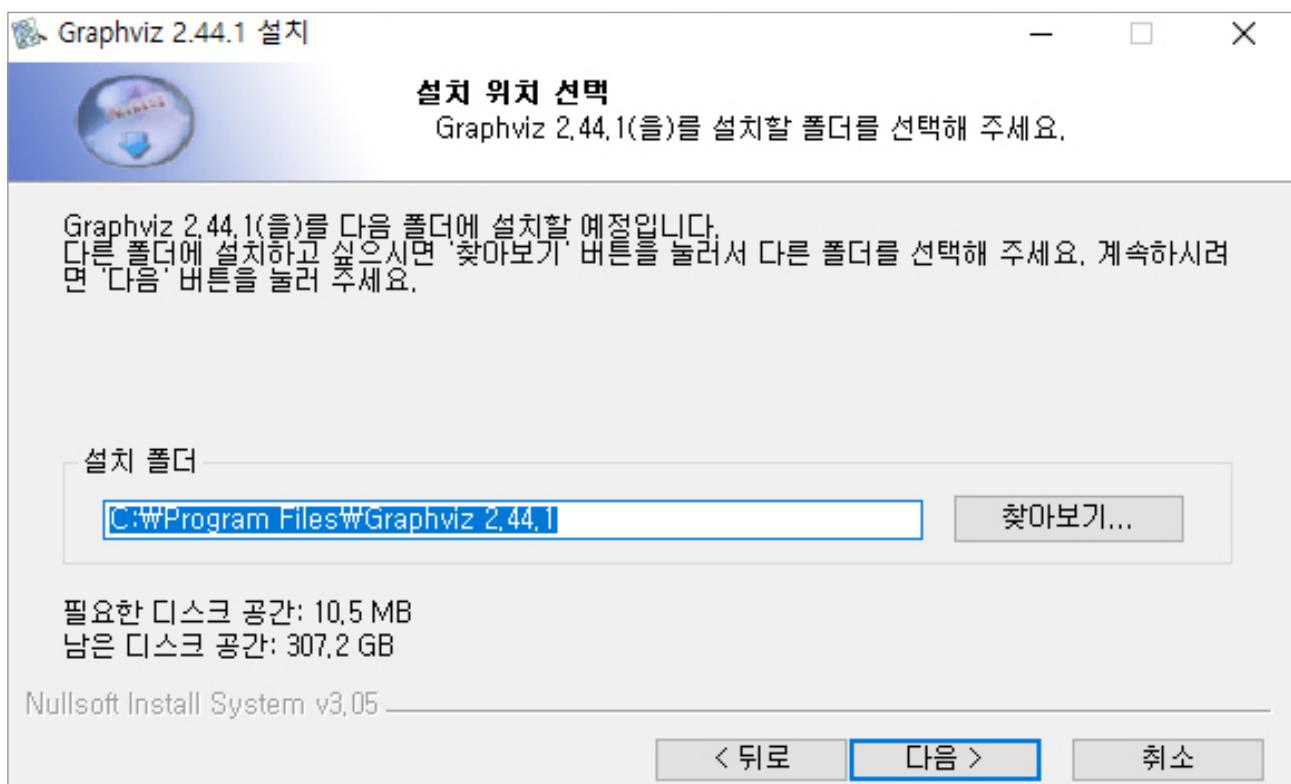
[그림 9] 설치 진행 화면

② [인스톨 옵션] 설치 화면에서 두 번째 옵션인 ‘Add Graphviz to the system PATH for all users’를 선택하고 다음을 클릭한다.



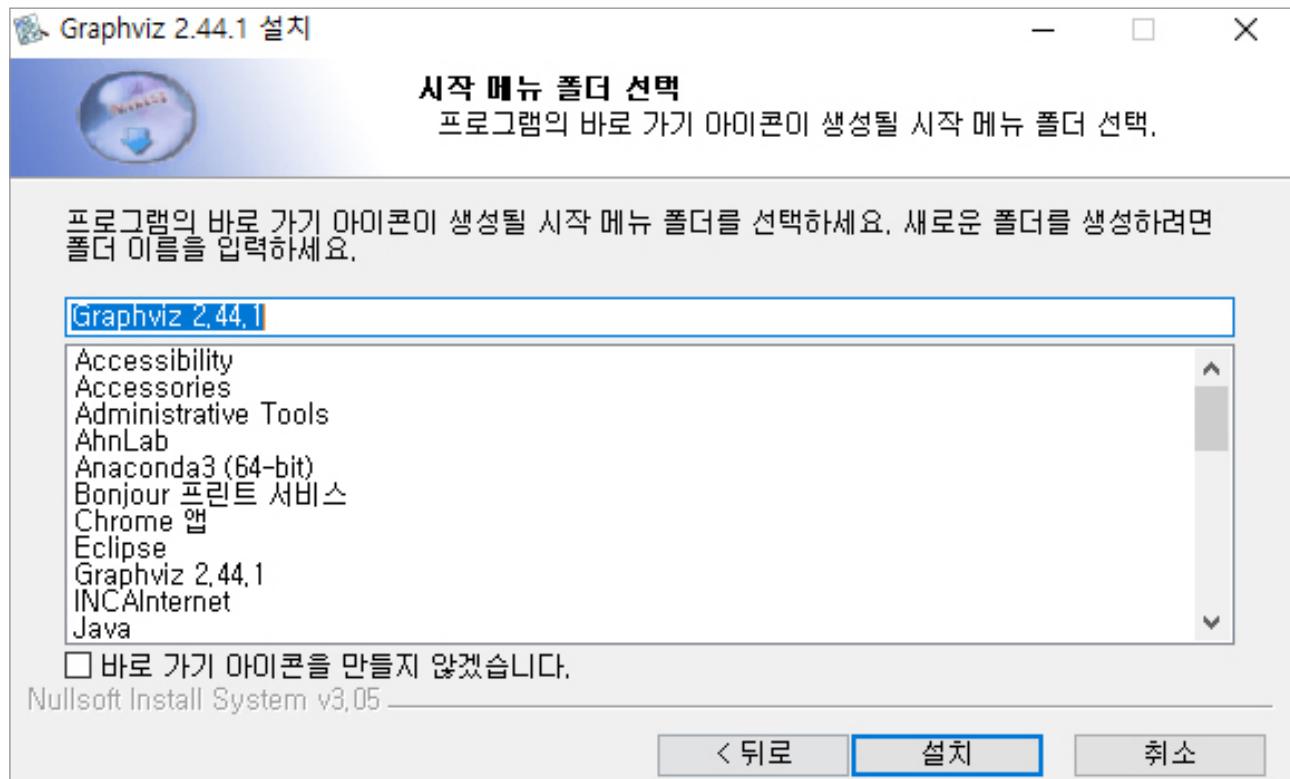
[그림 10] path 설정 화면

③ 다음 화면에서는 설치 폴더가 나오는데 디폴트로 나오는 경로를 기억해 두도록 한다.



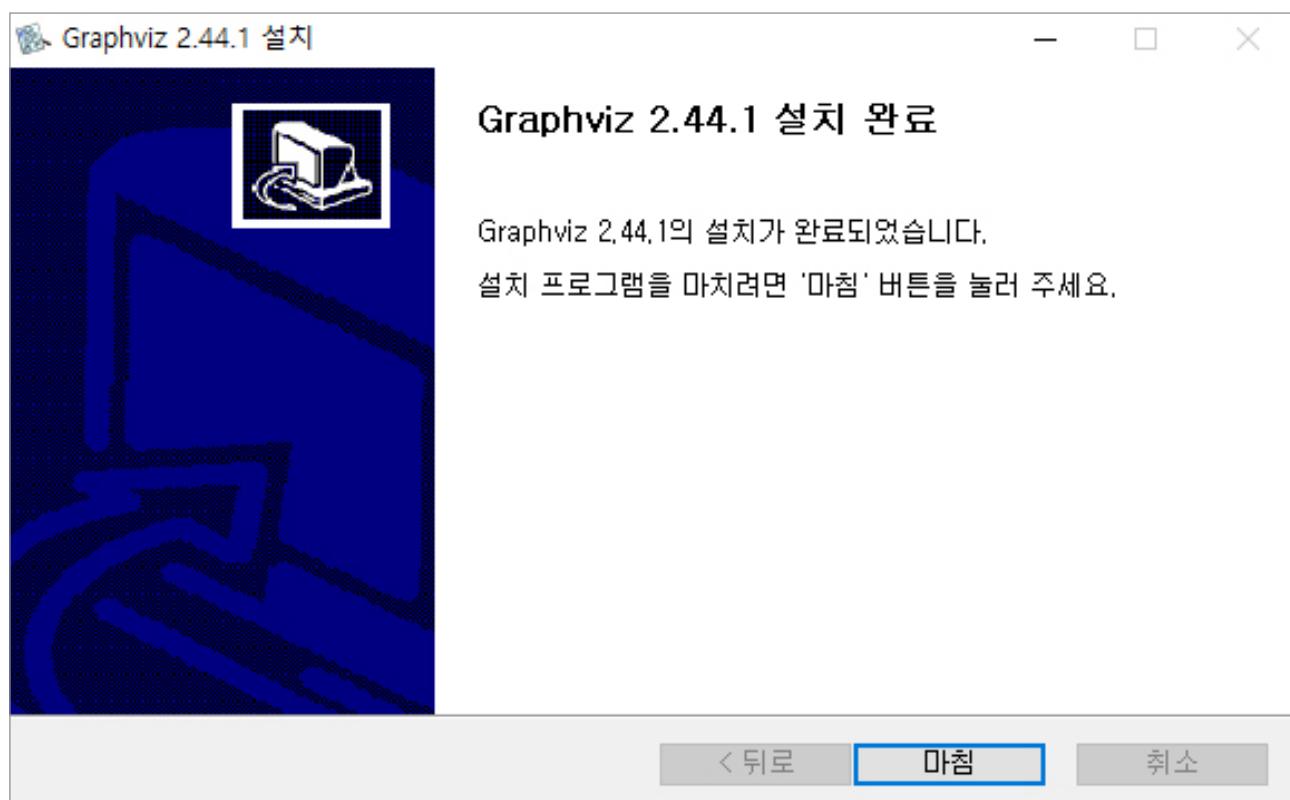
[그림 11] 설치 폴더 선택 화면

④ 마지막으로 나오는 [시작 메뉴 폴더 선택] 화면에서 설치 버튼을 클릭하여 설치를 시작한다.



[그림 12] 시작 메뉴 폴더 선택 화면

⑤ 설치가 완료되면 마침 버튼을 클릭하여 종료한다.



[그림 13] 설치 완료 화면

◉ 환경 변수 설정하기

- ① Windows 키를 눌러 검색란에 아래 그림과 같이 “시스템 환경 변수 편집”을 입력하여 나온 검색 결과를 선택한다.



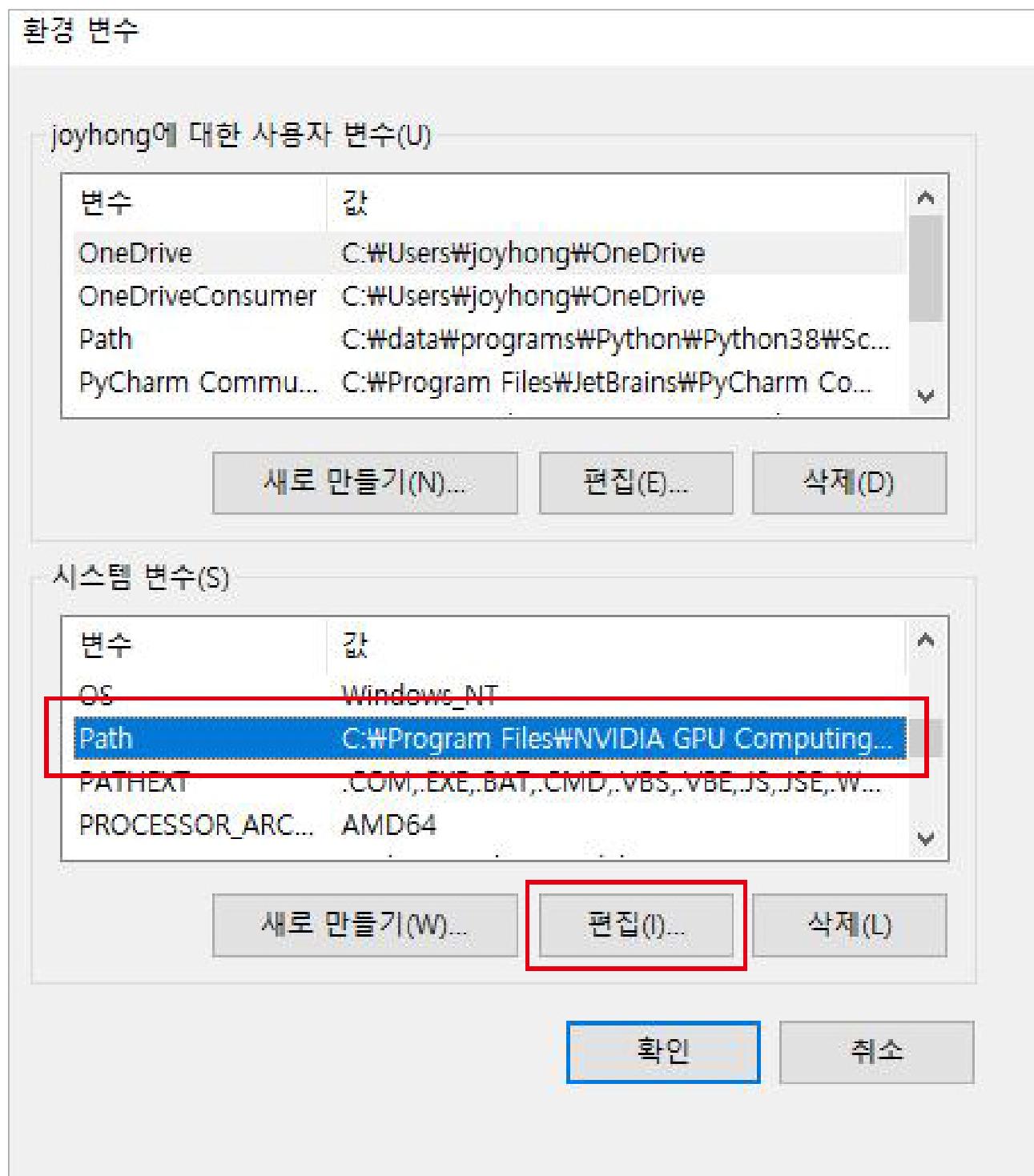
[그림 14] 시스템 환경 변수 편집 검색

- ② 시스템 속성 창이 나타나면 아래의 그림과 같이 [환경 변수] 버튼을 클릭하여 환경 변수 화면으로 이동한다.



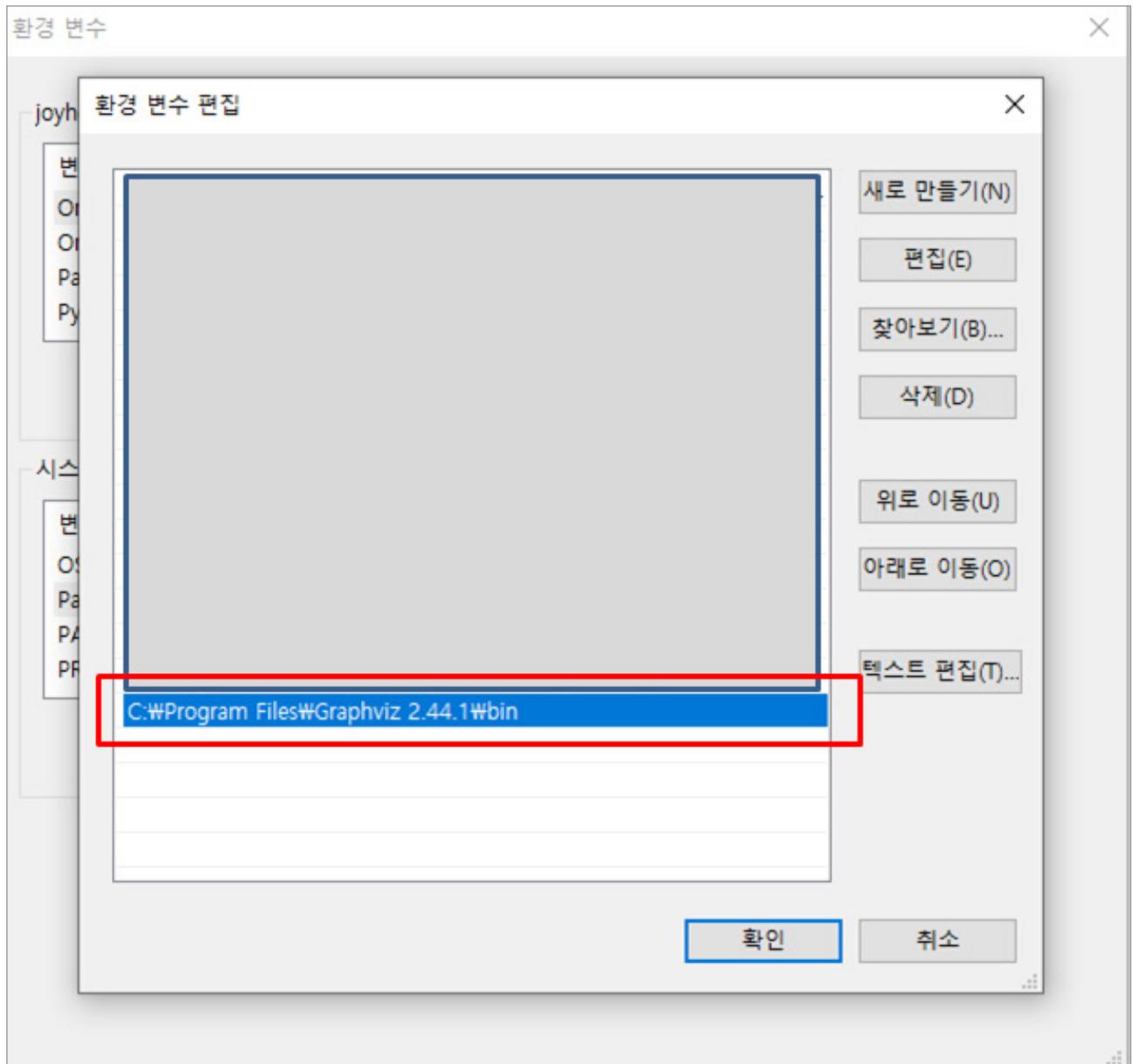
[그림 15] 환경 변수 선택

③ 시스템 변수란에 Path라는 변수를 찾아 편집 버튼을 선택한다.



[그림 16] 시스템 변수 편집

- ④ 환경 변수 편집 창에서 아래의 그림과 같이 “C:\ProgramFiles\Graphviz 2.44.1\bin”이 있는지 확인하고 만약 없다면 취소 버튼을 클릭하고 ⑤번을 진행하도록 한다.
만약 아래의 그림과 같이 환경 변수가 존재하면 설정이 완료된 것이다.

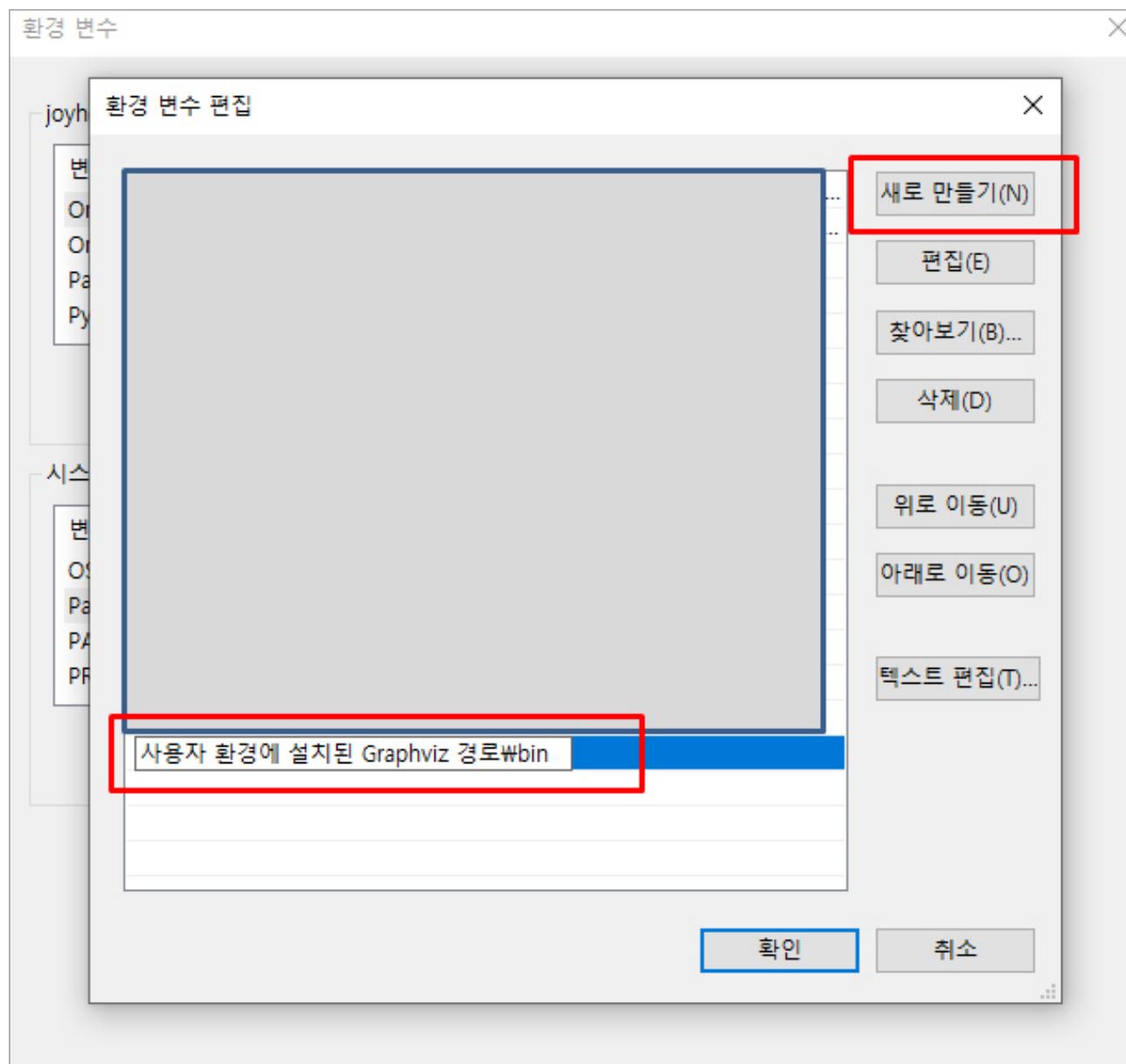


[그림 17] 환경 변수 확인

- ⑤ 시스템 환경 변수에 graphviz가 없을 경우, [새로 만들기] 버튼을 클릭 후 입력창에 사용자 환경에 설치된 graphviz 경로와 “\bin”을 붙여 입력 후 확인 버튼을 선택한다.

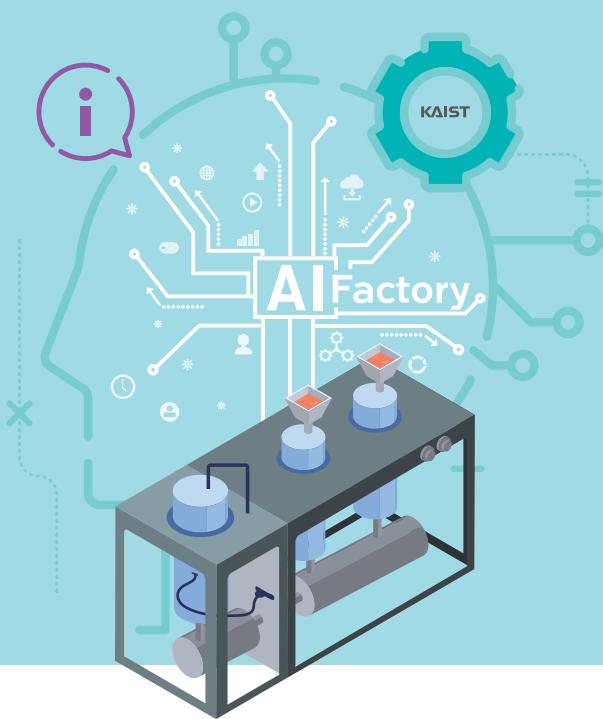
사용자 환경에 설치된 graphviz 경로는 graphviz 인스톨러를 통해 설치할 때 디폴트로 입력된 설치 폴더이다. 본 설치 가이드를 따라 진행했다면 “C:\ProgramFiles\Graphviz 2.44.1”에 설치가 되었을 것이고, 환경 변수 편집란에는 “C:\ProgramFiles\Graphviz 2.44.1\bin”을 입력하면 된다.

이상 모든 설정은 완료가 되었다.



[그림 18] 환경 변수 입력 화면

「살균기 AI 데이터셋」 분석실습 가이드북



중소벤처기업부



스마트제조혁신추진단



34141 대전광역시 유성구 대학로 291 한국과학기술원(KAIST)
T. (042)350-2114 F. (042)350-2210(2220)