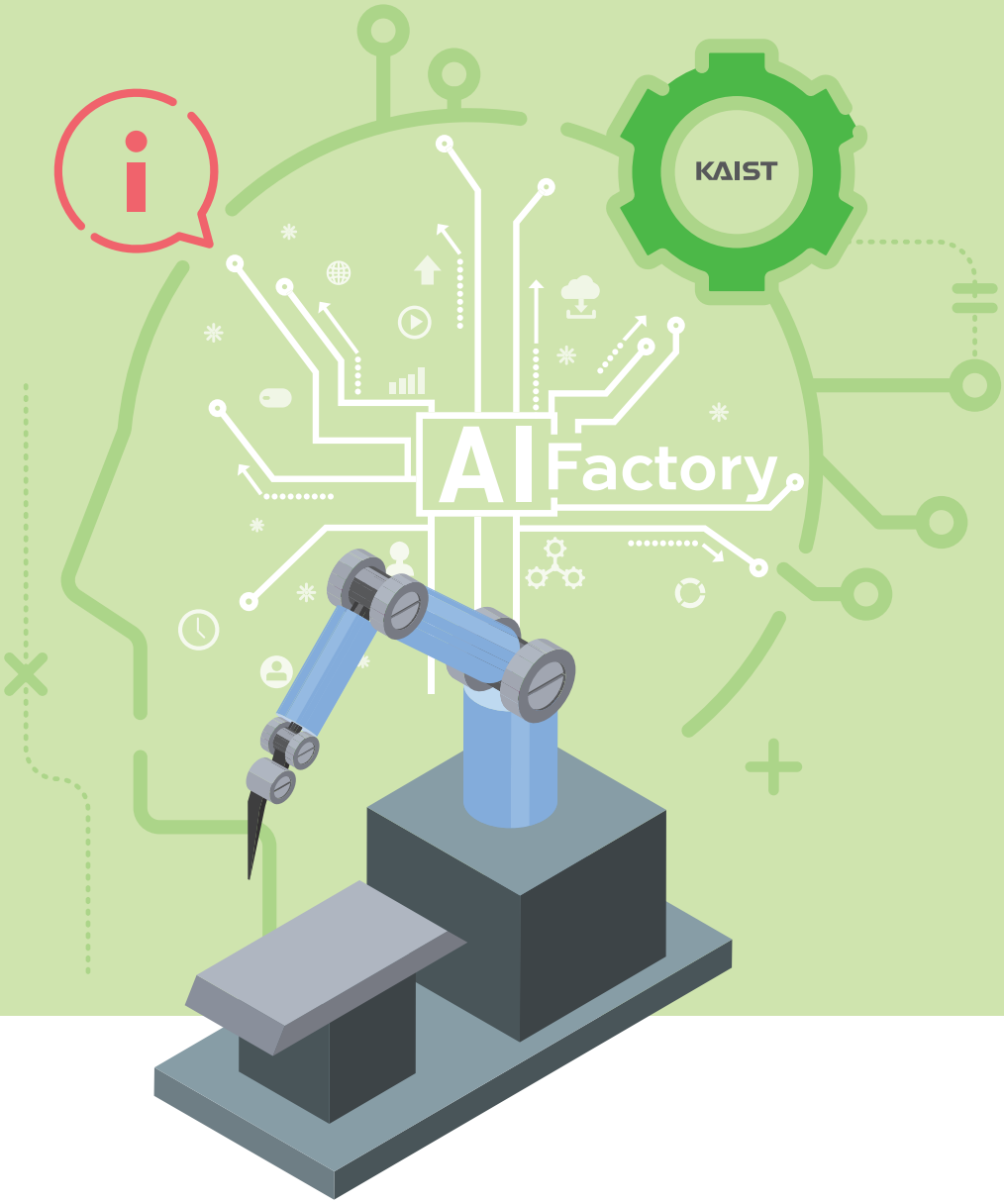
Guide book

「용접기 AI 데이터셋」 분석실습 가이드북

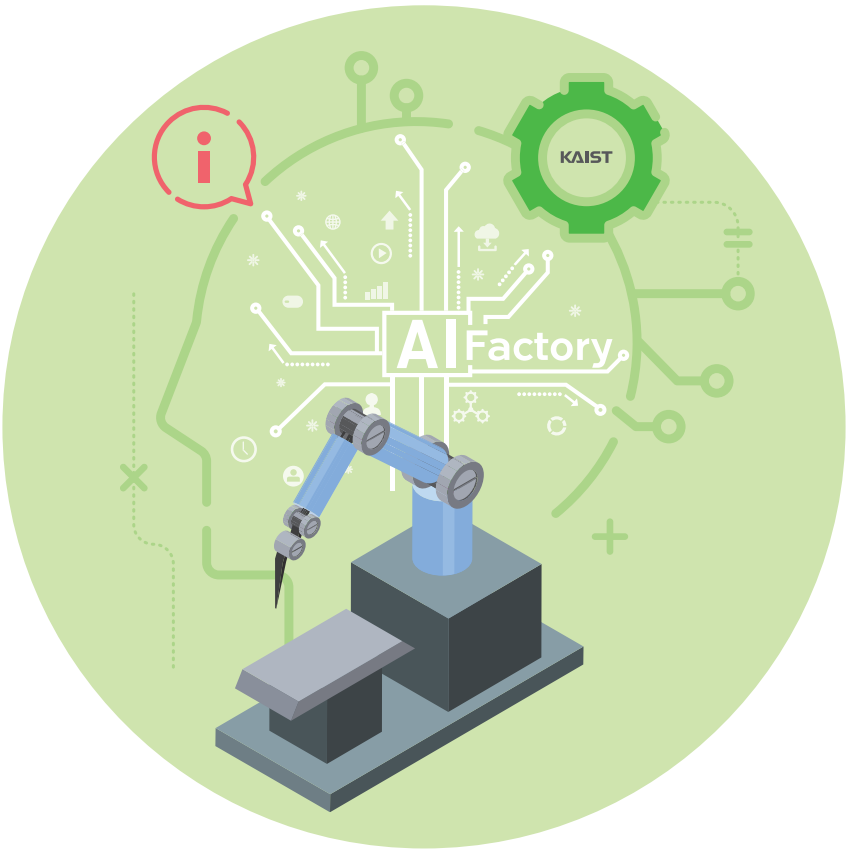


중소벤처기업부

KOSMO
스마트제조혁신추진단

KAIST

「용접기 AI 데이터셋」
분석실습 가이드북



중소벤처기업부
KOSMO
스마트제조혁신추진단
KAIST

Contents

「용접기 AI 데이터셋」 분석실습 가이드북

- **필 요 S W :** Python, Anaconda - Jupyter Notebook
- **필요 패키지 :** Pandas, matplotlib, scikit-learn
- **분 석 환 경 :** [운영체제] Ubuntu 14.0 이상, [CPU] Intel Xeon 2.3 GHz, [RAM] 13GB, [GPU] : Tesla K80
- **필요 데이터 :** Welding Data Set_01.xlsx(1차 가공데이터), scaled_data.csv(2차가공 데이터)
- **주 관 기 관 :** 한국과학기술원(KAIST)
- **수 행 기 관 :** 울산과학기술원(UNIST), 주식회사 이피엠솔루션즈

1 분석요약

No	구분		내용
1	**분석 목적** (현장 이슈, 목적)		- 본 인공지능 데이터 세트는 SPOT 용접공정에서 발생하는 품질문제를 해결하기위해, 생산 조건 최적화를 데이터 분석과 인공지능 모델을 개발하고 적용하는 방법을 학습한다. 용접공정의 문제점을 데이터 분석을 통하여 데이터 간의 상관관계를 찾고, 주요 문제별 원인 인자를 분석하는 과정과 다양한 기계 학습 알고리즘을 활용한 용접공정 불량예측모델과 생산조건최적화모델을 만드는 과정을 학습할 수 있다.
2	**데이터셋 형태 및 수집방법**		- **분석에 사용된 변수명 :** weld force(용접 가압력), weld current(전류), weld voltage(전압), weld time(통전 시간) - **수집 방법 :** 생산 Lot 단위 불량 이력 데이터 - **확장자 :** csv
3	**데이터 개수** **데이터셋 총량**		- **데이터 개수 :** Row 수(23,901개) - **데이터셋 총량 :** 548 KB
4	**분석적용 알고리즘**	**알고리즘**	Anomaly detection (이상치 탐지) - 오토인코더
		알고리즘 간략소개	- 이상치 탐지는 데이터 안에서 이상적인 패턴이나 비정상적인 샘플들을 찾는 알고리즘이다. 이상치 탐지에도 여러 종류가 있으며 용접기 데이터 에서 비지도 기반 이상치 탐지를 사용한다. 여기서 사용할 이상치 탐지 는 정상 샘플을 이용하여 정상 샘플이 가지는 분포의 경계를 학습한 뒤, 이 경계의 바깥에 있는 샘플들을 비정상 샘플이라고 판별하는 알고리즘 이다. 그리고 이상치 탐지에 사용될 오토인코더는 비지도 학습 방식을 사용하는 딥러닝 모델 중 하나이다. 데이터의 입력과 출력 형태가 똑같 이 나오며, 입력을 받았을 때, 최대한 입력과 똑같은 형태의 데이터를 출력하면서 입력데이터에 대한 표현과 패턴을 학습하는 모델이다
5	**분석결과 및 시사점**		- 용접공정에서 발생하는 4가지 데이터(용접 가압력, 전류, 전압, 통전시간)를 분석하여 불량품을 예측하기위해 AI 모델을 학습시켰다. 개별 불량품 예측에 대한 정량적 평가는 어렵지만, 일별 불량품 개수에 대한 예측은 적정 수준을 보였다. - 본 모델을 사용하는 제조기업이 보다 많은 데이터 특성을 사용한다면 더 효과적인 분석이 가능 할 것이고, 이를 통해 용접공정에서의 불량예측으로 제품의 품질을 향상하는데 기여할 것이다."

2 분석 실습

1. 분석 개요

1.1 분석 배경

1) 공정(설비) 정의 및 특징

• 용접 개요

- 용접이란 용접부를 용융상태로 하던가 용융상태로 하지 않을 정도로 가열한 부재 또는 상온 상태의 부재를 서로 접속하고 압력을 가하여 접합하는 이음 방법을 말한다. 즉 두 개의 금속을 접합하여 하나로 만드는 작업으로서 나사 또는 리벳을 사용하는 기계적 이음과는 다르며 일반으로 용접으로 이음된 것은 분해할 수 없다.
  용접은 융접, 압접, 납땜, 에너지 이용원에 따른 용접 등 크게 4가지로 분류될 수 있다. 본 실습은 융접 용접의 일종인 Spot 용접에 다루고자 한다.

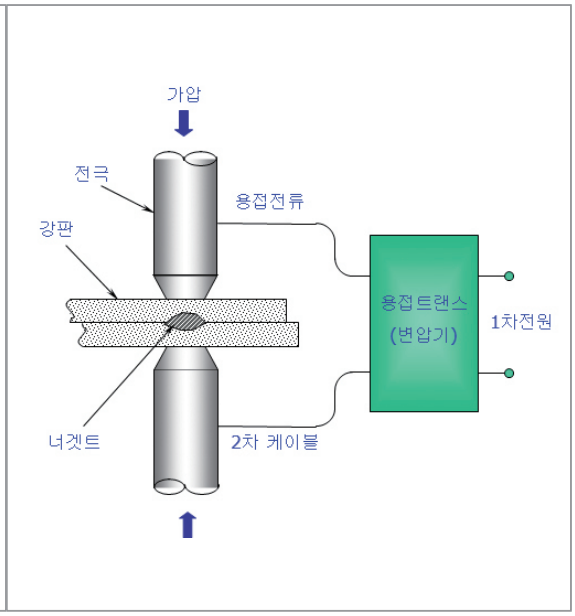
• Spot 용접

- 용접하고자 하는 재료를 전극(TIP)사이에 두고 가압하면서 전류를 통하면, 줄의 법칙에 의한 저항열이 발생한다. 이때 발생하는 저항열을 이용하여 접합부를 가열 융합하는 용접법으로 작업속도가 빠르고, 재료비가 절약되며 용접표면이 평평하고 깨끗하여 차체부품에 많이 이용되는 용접법이다.

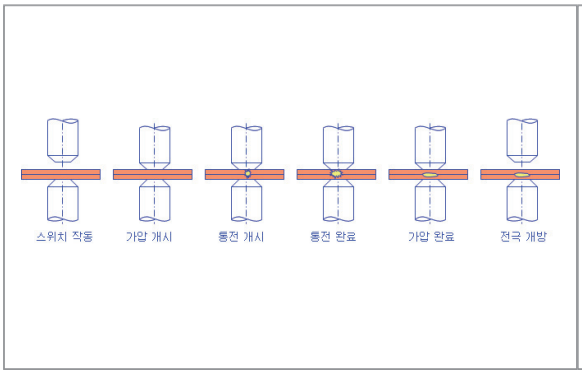
• 공정(실습설비) 개요

- 저항용접 발열량은 Q = 0.24 (I 전류, R 판의 저항, T 통전시간)이며, Spot 용접의 4대 요소는 용접전류, 통전시간, 가압력, 전극이다. 용접전류는 교류를 주로 사용하며, 발열량이 I*I에 비례하므로 전류 값은 용접결과에 중요한 변수로, 판 두께가 두꺼울수록 전류 값은 커진다. 통전시간은 너깃의 경을 제어하기 위해 통전시간으로 발열과 방열의 적절한 균형을 조절해야 한다. 열전도가 좋은 재료는 대전류 통전시간을 짧게 한다. 가압력이 크면 저항이 작아져서 유효발열량은 떨어지나, 작아지면 접촉저항 분포가 불균일하여 스파크가 발생하므로 조절이 필요하다. 전극의 촉부 면적은 전류밀도와 연관되어 용접품질에 영향을 미치며, 냉각 여부도 용접품질과 전극 마모율에 영향 변수이다.

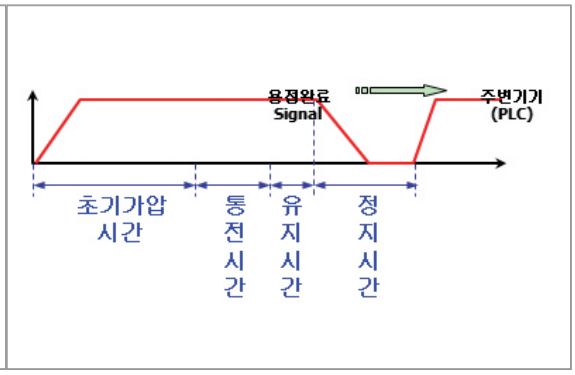
[그림 1] 용접 설비



[그림 2] Spot 용접 구성도



[그림 3] 용접 공정 순서



[그림 4] 용접시간별 구성

- **용접과정**

  - 타이머 전원 스위치를 켜고, 급수 및 에어밸브를 개방한다.
  - 용접 전원 스위치 켠 다음 건 스위치를 누르면서 동작을 확인한다.
  - 초기 너깃 시험을 통하여 설비를 조정한 다음 양호한 상태에서 작업을 연속적으로 실시한다.
  - 작업을 완료한 후 타이머와 용접 전원 스위치를 끄고 급수 밸브 및 에어밸브 잠금 및 전원을 끈다.

  * 판과 판을 밀착시키는 초기 가압시간을 거쳐 통전시간, 그리고 용접부가 응고하는데 소요되는 유지시간을 거친 다음 다시 처음 작동할 때까지의 정지시간으로 운영된다.

2)이슈 사항(Pain point)

• 공정(설비)상의 문제 현황

- 용접 후 과한 용접 파임, 판의 들뜸, 용접 불균일, 용접 크랙 발생 등 용접 불량 발생하고 있으며, 이에 따른 재작업으로 재료비 및 인건비 등 부가적인 비용 상승과 제품 공급 후 발생하는 용접 불량에 따른 고객 클레임 등의 이슈가 발생하고 있다.
  이를 개선하기 위해 기존 통계적 분석 방법을 벗어나 용접 두께에 따른 최적의 용접 조건 결정 모델 및 불량 예측과 원인 분석 관련하여 생산 조건 최적화를 데이터 분석과 불량 예측을 인공지능 모델을 개발하여 적용하고자 하는 목적이 있다.

• 문제해결 장애 요인

- 로봇 기반의 용접 자동화 공정이 아닌 작업자 기반으로 용접작업이 되고 있어 판 두께에 따른 용접 조건을 경험 및 직관에 의한 조정과 검사 또한 육안에 의존하고 있어 인공지능 모델에 필요한 자료를 수집하는 데 어려움이 있다.

• 극복 방안

- 용접의 전류, 가압력, 통전시간을 PLC와 연동하여 MES에 수집하고, 판의 두께는 ERP-MES의 데이터를 용접 작업 오더 데이터를 가지고 활용하였으며, 검사는 별도 인원 투입으로 양품/불량품, 그리고 불량품의 등급을 분류하여 기록하여 분석을 위해 라벨링 하였다.

1.2 분석목표

1) 분석목표 설정

- 용접공정의 문제점을 데이터 분석을 통하여 데이터 간의 상관관계를 찾고 주요 인자를 분석하는 과정과 다양한 기계 학습 알고리즘을 활용한 용접공정 불량 원인 분석 및 생산 최적 조건 모델 개발 및 적용을 통해 Spot 용접공정의 품질을 현 수준 대비 30% 개선하는 것을 목표로 하고자 한다.

2) 데이터 정의 및 소개

공정 변수 조건		내 용
독립변수	소재 두께	용접하고자 하는 강판의 두께
	용접 가압력	용접 지점에서 가해지는 압력
	전류 세기	용접 지점에서 측정된 전류
	전압 세기	용접 지점에서 측정된 전압
	통전 시간	전극에 용접전류를 통한 시간
종속변수	불량 여부	제품 각각에 대한 불량 선별값 (표시 없음 / Unlabeled)

3) 제조데이터 분석 기대 효과

- 용접공정에서 수집되는 다양한 변수들에 따라 양품 및 불량품을 예측할 수 있는 모델을 제공한다.

4) 시사점(implication) 요약기술

- 자동차 부품 D사 공장에는 육안으로 용접 양품과 불량을 검사하고 있으나, 양품과 불량 판별하고 불량은 유형별 세분화하여 판별할 수 있는 인공지능 기법의 적용과 불량의 세분된 유형에 따라 용접 조건 데이터와의 상관관계를 통해 원인 인자 도출 및 영향 인자의 우선순위(Priority)를 결정하는 인공지능 모델과 이의 결과 기반 재현 검증, 그리고 양품과 불량의 예측 모델 또한 필요한 상황이다.
  이에 용접공정의 검사 판정 데이터 및 생산 조건 자료를 수집하여 가공/전처리 및 인공지능 모델 개발과 제조 공정의 적용 및 검증을 통한 수행으로 열악한 중소기업에 빅데이터 및 인공지능 기반 기술로 실질적인 품질 및 비용 절감 개선에 기여했다는 점에 대해서 시사하는 바가 크다고 판단된다.

2. 분석 실습

2.1 제조데이터 소개

1) 데이터 수집 방법

- **제조 분야** : 자동차 부품 (용접)
- **제조 공정명** : 자동차 부품 용접
- **수집 장비** : Spot 용접 공정대상 PLC-MES Data 및 Mongo DB
- **수집 기간** : 2020/03/24 ~ 2020/04/07 (약 15일)
- **수집 주기** : 약 8초 주기로 4지점에서의 센서 데이터 수집
  각 지점 별 0.072초동안 데이터를 수집한다

2) 데이터 유형/구조

- **비식별화 원본 데이터** : 공정 과정에서 수집된 센서 데이터에서 비식별화 및 품질 전처리를 수행한 데이터셋이다
- **학습 통합 데이터 (label 데이터)** : "2.1 제조데이터 소개" 의 "1) 1차 가공 데이터"와 같은 비식별화 원본 데이터의 경우 모델의 입력으로 들어가 위해 전처리 과정이 필요하다. 용접기 데이터의 특성인 생산 품목, 작업 시간 그리고 소재 두께들은 데이터 값이 1가지로 모든 샘플에서 동일하므로 모델의 입력으로 알맞지 않다. 따라서 그 특성들은 제외한, 용접 가압력, 전류, 전압, 그리고 통전 시간을 가지고 있는 열만 남겨둔다. 남겨둔 특성(용접가압력, 전류, 전압, 통전시간)의 경우 모델 훈련의 용이성을 위한 정규화 전처리 과정을 적용한다. 해당 전처리 과정을 거친 학습 통합 데이터를 사용하여 모델을 학습시킨다.

- **1차 가공 데이터** : Welding Data Set_01.xlsx

idx	Machine_Name	Item No	working time	Thickness 1	Thickness 2(m	weld force(ba	weld current(kA)	weld Voltage(v)	weld time
1	Spot-01	65235-25800	2020-03-24 0:00	0.7	0.7	2.33	14.57	2.70	72
2	Spot-01	65235-25800	2020-03-24 0:00	0.7	0.7	2.36	14.57	2.70	72
3	Spot-01	65235-25800	2020-03-24 0:00	0.7	0.7	2.37	14.54	2.70	71
4	Spot-01	65235-25800	2020-03-24 0:00	0.7	0.7	2.37	14.54	2.70	72
5	Spot-01	65235-25800	2020-03-24 0:00	0.7	0.7	2.36	14.56	2.70	72
6	Spot-01	65235-25800	2020-03-24 0:00	0.7	0.7	2.34	14.56	2.70	71
7	Spot-01	65235-25800	2020-03-24 0:00	0.7	0.7	2.35	14.53	2.70	72
8	Spot-01	65235-25800	2020-03-24 0:00	0.7	0.7	2.33	14.53	2.70	70
9	Spot-01	65235-25800	2020-03-24 0:00	0.7	0.7	2.32	14.57	2.70	71
10	Spot-01	65235-25800	2020-03-24 0:00	0.7	0.7	2.32	14.57	2.70	72
11	Spot-01	65235-25800	2020-03-24 0:00	0.7	0.7	2.33	14.59	2.70	72
12	Spot-01	65235-25800	2020-03-24 0:00	0.7	0.7	2.36	14.59	2.70	72
13	Spot-01	65235-25800	2020-03-24 0:00	0.7	0.7	2.37	14.59	2.70	73
14	Spot-01	65235-25800	2020-03-24 0:00	0.7	0.7	2.37	14.59	2.70	71

[그림 5] Welding_RawDataSet.csv 예시

물품 각각에 대한 생산 조건을 xlsx(엑셀파일)의 형태로 제공된다. 용접 데이터의 경우, 물품 각각에 대한 양품 불량 여부는 알 수 없다.

	A	B	C	D	E	F	G
1	idx	Machine_Name	Item No	working time	defect	defect type	
2	1	Spot-01	65235-25800	2020-03-24 0:00	2	1	파임불량
3	2	Spot-01	65235-25800	2020-03-24 0:00	1	2	용접부족
4	3	Spot-01	65235-25800	2020-03-24 0:00	1	3	크랙발생
5	4	Spot-01	65235-25800	2020-03-25 0:00	3	1	
6	5	Spot-01	65235-25800	2020-03-25 0:00	2	2	
7	6	Spot-01	65235-25800	2020-03-25 0:00	3	3	

[그림 6] 비식별화 원본 데이터 예시(용접)

개별 물품에 대한 양품 혹은 불량 여부는 알 수 없지만, 일별 불량 수 및 불량 유형에 대한 정보를 알 수 있다. 이를 이용하여, 비지도 학습에서 필요한 임계 값을 추정할 수 있다.

• 데이터 속성 정의표

Data	항목 설명	수집 범위	Unit
idx	생산순번	-	-
Machine_Name	생산설비	-	-
Item No	생산품목	-	-
working time	작업시간	-	-
Thickness 1	소재 두께 1	0.3~2.3	두께 (mm)
Thickness 2	소재 두께 2	0.3~2.3	두께 (mm)
weld force	용접 가압력	1.00 ~ 12.00	압력 (bar)
weld current	전류	12.00 ~ 18.00	전류 세기 (kA)
weld Voltage	전압	1.50 ~ 3.50	전압 세기 (V)
weld time	통전시간	30 ~ 120	시간 (ms)

• 2차 가공 데이터 : scaled_data.csv

	weld force(bar)	weld current(kA)	weld voltage(v)	weld time(ms)
0	0.004481651	0.001716984	0.002505742	0.000460829
1	0.005411804	0.000500787	0.002505742	0.000921659
2	0.004989007	0.000357705	0.507830445	0.000921659
3	0.005242686	0.000357705	0.002505742	0.000921659
4	0.005327245	0.000143082	0.002505742	0.000460829
5	0.005327245	0.000143082	0.002505742	0.000921659
6	0.005242686	0.000286164	0.002505742	0.000921659
7	0.005073567	0.000286164	0.002505742	0.000460829
8	0.005158126	7.15E-05	0.002505742	0.000921659
9	0.004989007	7.15E-05	0.508874504	0
10	0.004904448	0.000357705	0.002505742	0.000460829

[그림 7] 학습 통합 데이터 예시(용접))

물품 각각에 대한 생산 조건을 csv(column separated value)의 형태로 제공된다. 용접 데이터의 경우, 물품 각각에 대한 양품 불량 여부는 알 수 없다. 또한 1차 가공 데이터에 있었던 생산 품목, 작업 시간, 그리고 소재 두께 특성들은 모든 샘플에서 동일한 값을 가지므로 모델의 학습에 적절하지 않은 특성들이다. 따라서 2차 가공 데이터는 위의 특성들을 제외한 4가지 특성(용접 가압력, 전류, 전압, 통전 시간)만 남겨둔 전처리 데이터이다. 또한 모델의 입력으로 들어가기 위한 정규화 전처리 과정을 거친 데이터로 제공된다.

• 데이터 속성 정의표

Data	항목 설명	수집 범위	Unit
weld force	용접 가압력	1.00~12.00	압력 (bar)
weld current	전류	12.00~18.00	전류 세기 (kA)
weld Voltage	전압	1.50~3.50	전압 세기 (V)
weld time	통전시간	30~120	시간 (ms)

• 기술 통계

구 분	개 수	평 균	표준편차	최소값	중간값	최대값	최빈값
weld force	11921	3.022	4.254	1.74	2.34	120	2.31
weld current	11924	14.932	3.617	14.52	14.73	154.3	14.73
weld Voltage	11924	2.846	3.056	2.46	2.7	98.24	2.7
weld time	11916	76.963	85.564	70	72	2240	72

• 독립변수/ 종속변수 정의

- 독립변수란, 다른 변수에 영향을 받지 않는 변수로, 입력값이나 원인을 나타낸다.
- 종속변수란, 독립변수의 변화에 따라 어떻게 변화하는지를 알고 싶은 변수로, 결과물이나 효과를 나타낸다.

공정 변수 조건	내 용
독립변수	소재 두께
	용접 가압력
	전류
	전압
	통전 시간
종속변수	(표시 없음)

• **데이터 (품질) 전처리**

- 품질 전처리 목적

◦ 실제 공정에서 발생하는 데이터들은 값이 의미 없거나 누락 및 오타가 발생하여 데이터 품질이 떨어진다. 품질이 낮은 데이터를 이용하면 좋은 결과를 얻을 수 없으므로 데이터 품질 전처리는 데이터 분석에서 가장 중요한 단계이다.

◦ 따라서 데이터들의 5가지 품질 지수를 파악하고, 데이터 전처리를 통해 품질 지수를 향상한다.

• **데이터 품질 지수**

◦ **완전성(Completeness)** : 필수항목에 누락이 없어야 한다.

◦ **유일성(Uniqueness)** : 데이터 항목은 유일해야 하며 중복되어 서는 안된다.

◦ **유효성(Validity)** : 데이터 항목은 정해진 데이터 유효범위 및 도메인을 충족해야 한다.

◦ **일관성(Consistency)** : 데이터가 지켜야 할 구조, 값, 표현되는 형태가 일관되게 정의되고, 서로 일치해야 한다.

◦ **정확성(Accuracy)** : 실제 존재하는 객체의 표현 값이 정확히 반영되어야 한다.

<예시>

idx	Machine_Nam	Item N	working time	Press time(ms	Pressure	Pressure	Pressure
1843	Press-01	ED5260	2020-05-07 0:0		275.2	273.2	548.4
1844	Press-01	ED5260	2020-05-07 0:0		275.5	273.1	548.6
1885	Press-01	ED5260	2020-05-07 0:0		274.8	276	550.8
1886	Press-01	ED5260	2020-05-07 0:0		275.6	273.2	548.8
1619	Press-01	ED5260	2020-05-15 0:0		274.8	267	541.8
561	Press-01	ED5260	2020-05-20 0:0		274.9	269	543.9
755	Press-01	ED5260	2020-05-26 0:0		274.9	269	543.9
3879	Press-01	ED5260	2020-05-28 0:0		275.2	267	542.2
744	Press-01	ED5260	1900-01-00 0:00		274.9	269	543.9

-> 컬럼에 null 값이 들어가 있으므로 완전성을 위배한다.

idx	Machine_Nam	Item N	working time	Press time(ms	Pressure	Pressure	Pressure
2912	Press-01	ED5260	1900-01-00 0:00	551	275.5	269	1644.5
105	Press-01	ED5260	8159-02-28 0:00	549	274.5	269	543.5
107	Press-01	ED5260	8277-03-19 0:00	550.2	275.1	267	542.1
108	Press-01	ED5260	8336-03-29 0:00	550.2	275.1	267	542.1
111	Press-01	ED5260	8513-04-27 0:00	550.2	275.1	267	542.1
136	Press-01	ED5260	9988-12-25 0:00	549.8	274.9	267	541.9
2	Press-01	ED5260	1900-01-00 0:00	550	275	267	542
138	Press-01	ED5260	1900-01-00 0:00	549.2	274.6	267	541.6
140	Press-01	ED5260	1900-01-00 0:00	550.2	275.1	267	542.1
142	Press-01	ED5260	1900-01-00 0:00	551	275.5	267	542.5

-> working time에 유효하지 않은 날짜형식이 들어있으므로 유효성을 위배한다.

[그림 8] 데이터 유효성 검사

- **데이터 품질 지수 :** 세부 설명

◦ 완전성 품질 지수 = ((1-결측치)/전체 데이터 수) * 100

① Null 값이 30%이상인 데이터들은 데이터의 완전성이 떨어지기 때문에 열별 Null값의 비율을 확인하여 삭제한다.

② 데이터의 결측치를 확인하기 위해 isnull()함수를 사용한 뒤 sum()함수를 이용하여 총 결측치 개수를 구한다

③ 구한 결측치의 개수를 이용하여 완전성 품질 지수를 구한다.

◦ 유일성 품질 지수 = ((1-중복데이터 수)/전체 데이터 수) * 100

① 이 데이터는 유일한 값을 가지는 열이 존재하지 않으므로 유일성을 판단하지 않는다.

② 유일성을 판단하고 싶다면 idx와 workingtime 두 열을 이용하여 합성키를 만든 뒤 기본키로 설정하여 중복을 판단하면 된다.

◦ 유효성 품질 지수

① 데이터가 유효범위 내에 있는가?

② 데이터가 형식에 맞는가?

③ 수집된 날짜 안에 들어가 있는가? 등을 검증하는 것이다.

◦ 일관성 품질 지수 = (일관성 만족 데이터 수/전체 데이터 수)*100

여기 용접 데이터는 다른 테이블을 참조하는 열이 없으므로 일관성을 판단하지 않는다.

◦ 정확성 품질 지수 = (1-(정확성 위배 데이터 수/전체 데이터 수))*100

여기 용접 데이터는 상관관계가 있는 데이터가 존재하지 않기 때문에 정확성을 판단하지 않는다.

- 용접 데이터 품질 지수는 다음과 같다.

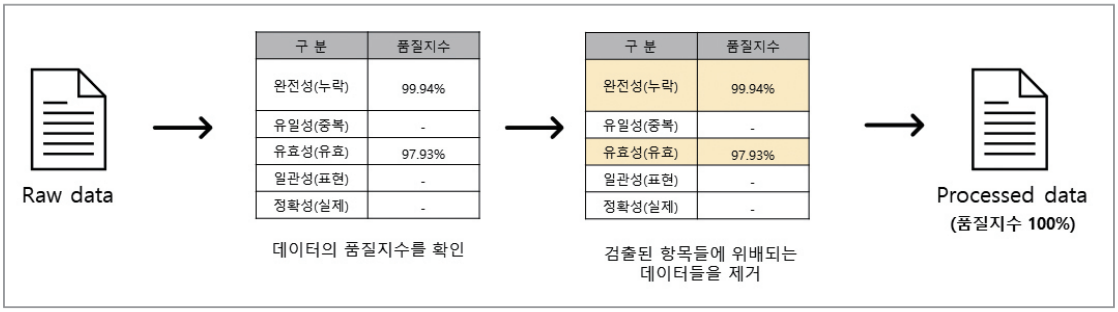
구 분	품질 지수	가중치	가중치 지수	오류율
완전성(누락)	99.94%	80%	79.95%	0.06%
유일성(중복)	-	-	-	-
유효성(유효)	97.93%	20%	19.59%	2.07%
일관성(표현)	-	-	-	-
정확성(실제)	-	-	-	-
품질 지수	98.94%	100%	**99.54%**	**0.46%**

[표 1] 용접 데이터 품질 지수

- 데이터 전처리 방법

  ◦ **개발 언어** : 파이썬(Python)

  ◦ **개발환경** : 주피터 노트북(Jupyter notebook)



[그림 9] 데이터 전처리 과정 도식화

- 데이터 전처리 결과물

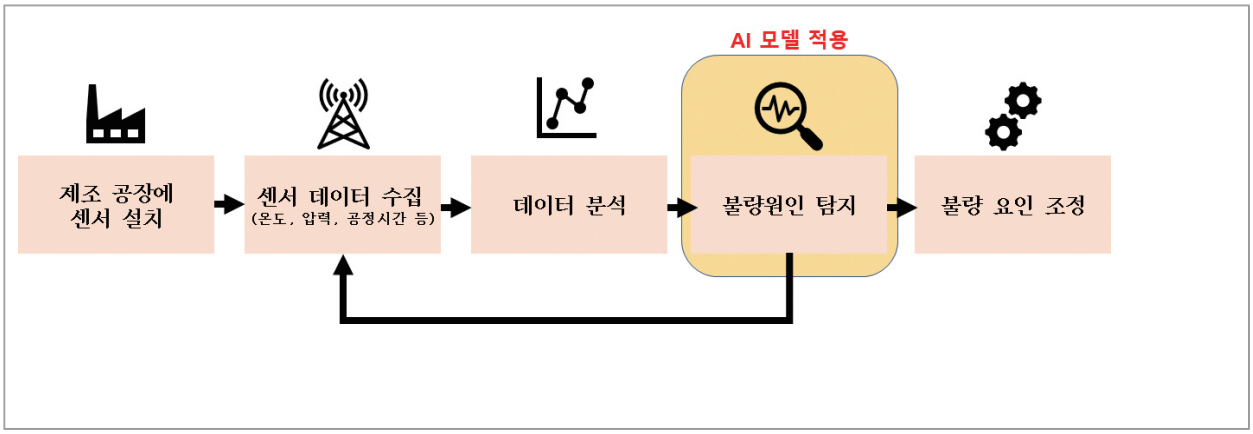
prod_dt

	idx	Machine_Name	Item No	working time	Thickness 1(mm)	Thickness 2(mm)	weld force(bar)	weld current(kA)	weld Voltage(v)	weld time(ms)	cooling temperature
0	2	Spot-01	65235-25800	2020-03-24	0.7	0.7	2.36	14.57	2.70	72.0	21.4
1	3	Spot-01	65235-25800	2020-03-24	0.7	0.7	2.37	14.54	2.70	71.0	21.4
2	4	Spot-01	65235-25800	2020-03-24	0.7	0.7	2.37	14.54	2.70	72.0	21.4
3	5	Spot-01	65235-25800	2020-03-24	0.7	0.7	2.36	14.56	2.70	72.0	21.4
4	6	Spot-01	65235-25800	2020-03-24	0.7	0.7	2.34	14.56	2.70	71.0	21.4
...	...	...	...	...	...	...	...	...	...	...	...
11687	665	Spot-01	65235-25800	2020-04-07	0.7	0.7	2.37	14.55	2.70	72.0	21.7
11688	666	Spot-01	65235-25800	2020-04-07	0.7	0.7	2.37	14.60	2.71	72.0	21.7
11689	667	Spot-01	65235-25800	2020-04-07	0.7	0.7	2.37	14.60	2.71	71.0	21.7
11690	668	Spot-01	65235-25800	2020-04-07	0.7	0.7	2.35	14.53	2.71	71.0	21.7
11691	669	Spot-01	65235-25800	2020-04-07	0.7	0.7	2.35	14.53	2.71	71.0	21.6

11692 rows × 11 columns

총 11,692개의 데이터로 약 2.16%(258개) Outlier로 검출되었다.

2.2 분석 모델 소개



[그림 10] 제조 공정에서의 데이터 흐름 및 인공지능 모델 적용 단계 도식화

1) 인공지능(AI) 분석모델

• 해당 AI 방법론(알고리즘) 선정 이유 기술

- 용접기 데이터 분석 방법 : 비지도 학습 적용한 데이터 분석

◦ 이상 탐지는 전체 데이터에서 비정상 샘플(불량품)을 찾아내는 방법론으로 종종 사용된다. 비용적인 부분 때문에 양품과 불량품 정보가 없어 목표치 없다. 따라서 용접기 데이터는 목표치가 없으면 사용할 수 있는 비지도 학습을 적용한다. 따라서 비지도 학습의 이상 탐지 방법을 적용한다.

생산품목	작업시간	소재 두께	용접 가압력	전류	전압	통전시간	불량 여부
65243	2020-07-12 0:00	13.2	9	24	2	60	(데이터 없음)
65243	2020-07-13 0:00	13.2	10	23	8	61	(데이터 없음)
65243	2020-07-14 0:00	13.2	8	21	8	67	(데이터 없음)

[표 2] 데이터 예시

• 적용하고자 하는 AI 분석 방법론(알고리즘)의 구체적 소개

- 비지도 학습이란?

◦ 비지도 학습이란 지도 학습의 반대 개념이며 데이터들이 목표치(label)들이 주어지지 않은 상태에서 훈련하는 방식을 비지도 학습이라고 불린다. 예를 들어, 양품과 불량품에 대한 데이터를 학습할 때에 아래 테이블과 같이 목표치(양품과 불량품에 대한 정보)가 존재한다면 지도 학습이며, 목표치가 존재하지 않는다면 비지도 학습이라고 할 수 있다. 또한, 아래와 같은 목표치가 존재하는 데이터들을 레이블 데이터(labeled data)라 부르며, 목표치가 존재하지 않는 데이터들을 레이블 없는 데이터(unlabeled data)라 부른다.

온 도	압 력	공정 시간	양품 = 0/불량품 = 1
25	60	128	0
14	62	79	0
11	76	80	0
30	59	94	1

[표 3] 라벨링 된 데이터(labeled data)

온 도	압 력	공정 시간
25	60	128
14	62	79
11	76	80
30	59	94

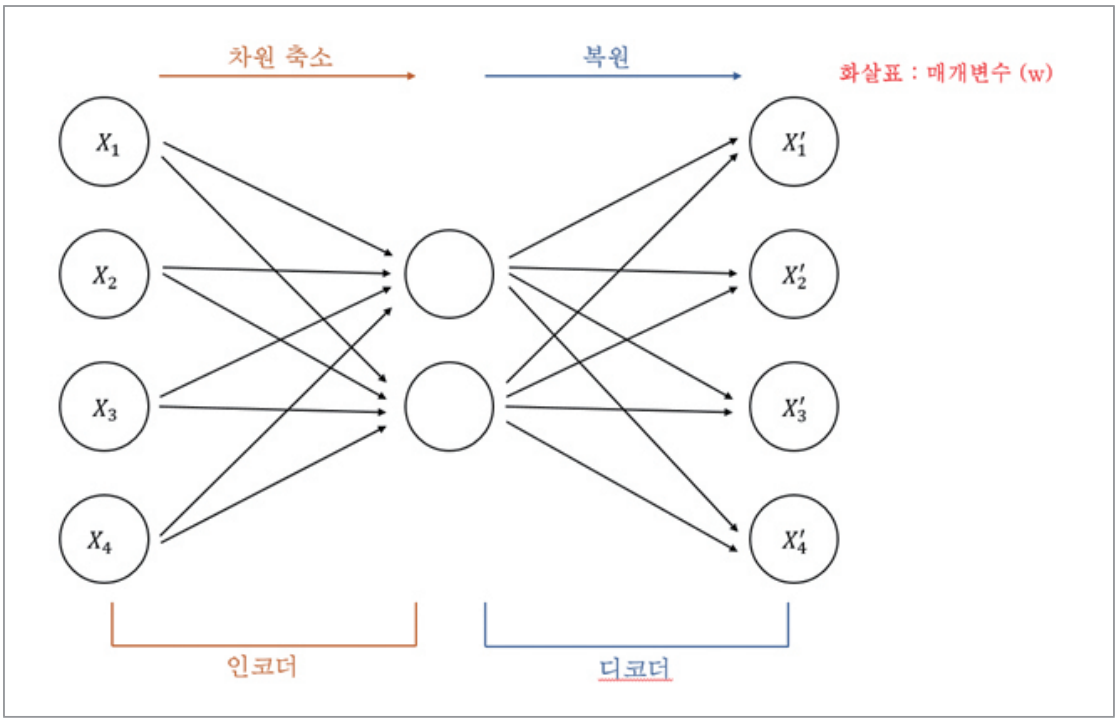
[표 4] 라벨링 안된 데이터(unlabeled data)

따라서, 용접기 데이터처럼 일별 데이터에 대한 불량품의 개수 정보는 있지만 각 데이터의 양품과 불량품에 대한 정보가 없으면, 결국 레이블 없는 데이터이기 때문에 비지도 학습을 적용한다.

• AI 분석 방법론(알고리즘) 구축 절차 설명

- 오토인코더(AutoEncoder)

◦ 오토인코더는 데이터에 대한 목표치 없이도 입력데이터에 대해서 효율적인 데이터 표현 및 특성을 학습할 수 있는, 비지도 학습 모델이면서 인공신경망 모델 중 하나이다 (출처: 핸즈온 머신 러닝-오렐리앙 게론). 기본적으로 심층 신경망처럼 여러 신경망을 쌓아 만들 수 있으며, 두 가지 구조로 되어 있다. 인코더 부분과 디코더 부분으로 이루어져 있으며, 인코더 부분에서 데이터의 입력에 대한 차원 축소가 일어나며 이 부분에서 차원 축소를 하면서 데이터의 효율적인 표현들을 학습한다. 디코더 부분에서는 앞에서 인코더가 축소한 데이터를 입력으로 받아 다시 원래 상태의 데이터로 복원시키는 기능이 있다. 결국, 입력의 형태와 출력의 형태가 같다는 점을 제외하면 심층 신경망과 같은 구조이다. 또한, 출력은 입력을 재현한다는 점에서 재구성이라고도 불린다. 쉽게 말해서, 오토인코더는 데이터 입력을 복제하는 과정이며 이의 부산물로서 데이터에 대한 효율적인 특성들을 학습하게 되는 것이다. 여기서 학습을 한다는 것은 오토인코더가 가지고 있는 매개변수들을 조정한다는 의미이며 매개변수는 아래의 그림처럼 입력을 받았을 때 다음 층으로 넘어갈 때 곱해주는 가중치라고 생각하면 된다. 오토인코더의 기본 구조는 아래의 그림과 같다.



[그림 11] 오토인코더의 구조

- 오토인코더의 구현

◦ 오토인코더는 자체적으로 비지도 학습 방식의 모델이라, 레이블 없는 데이터에 적합한 모델이다. 용접기 데이터 또한 개별 레이블 없는 데이터들이라 비지도 학습 데이터라 할 수 있으며, 이를 사용하여 오토인코더를 직접 학습시켜준다. 학습시키는 과정에서 학습에 대한 기준이 필요한데 이를 손실 함수라고 하며, 이를 최소화하는 방향으로 학습이 이루어진다. 오토인코더에서는 손실 함수로서 재구성 오류(Reconstruction error)를 사용한다. 또한, 재구성 오류를 계산하는 방식에도 여러 가지 방식이 있는데, 어떤 식으로 정의해주는지에 따라서 결과가 달라질 수 있다.
위와 같은 손실 함수를 최소화하는 방향으로 오토인코더는 매개변수들을 조정해 나가며 이를 "학습"이라고 한다.

2.3 분석 체험

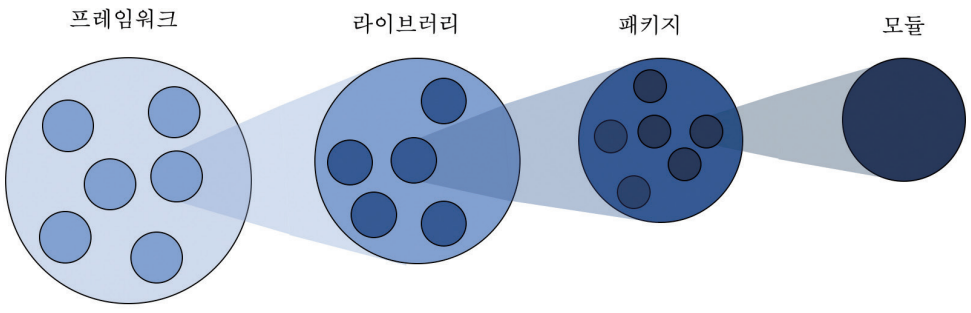
1) 필요 SW, 패키지 설치 방법 및 절차 가이드

• SW 설치는 '부록(분석환경 구축을 위한 설치 가이드)' 참고

• 패키지 설치 가이드

- 패키지 설치 전 파이썬 관련 용어 정리

프레임워크	- 라이브러리의 모음 - 프레임워크가 곧 프로그램의 구성요소이다.
라이브러리	- 여러 패키지의 모음 - 파이썬을 설치할 때, 기본적으로 설치되는 라이브러리를 표준라이브러리라고 한다. 파이썬 공식이 아닌 외부에서 개발한 모듈과 패키지를 묶어 외부 라이브러리라고 한다.
패키지	- 여러 모듈들의 모음 - 패키지 안에 여러 가지 폴더가 존재할 수 있다.
모 듈	- 특정 함수, 변수, 클래스 등이 구현되어 있는 파이썬 파이(.py)을 칭한다. 대표적으로 아래 모듈들이 존재한다. - 예) numpy: 수치해석 모듈, pandas: 데이터 분석 모듈



[그림 12] 파이썬 관련 용어 개념

- 패키지 설치 과정

① 주피터 노트북 내에서 패키지 및 모듈 설치하기

필요 패키지 및 모듈 설치

예) datetime 모듈 설치

```
!pip install datetime
```

```
In [5]: pip install datetime

        Defaulting to user installation because normal site-packages is not writeable
        Collecting datetime
          Downloading DateTime-4.3-py2.py3-none-any.whl (60 kB)
             |████████████████████████████████| 60 kB 1.3 MB/s eta 0:00:011
        Requirement already satisfied: zope.interface in ./Python/anaconda3/lib/python3.8/site-packages (from datetime) (4.
        7.1)
        Requirement already satisfied: pytz in ./Python/anaconda3/lib/python3.8/site-packages (from datetime) (2020.1)
        Requirement already satisfied: setuptools in ./Python/anaconda3/lib/python3.8/site-packages (from zope.interface->d
        atetime) (49.2.0.post20200714)
        Installing collected packages: datetime
        Successfully installed datetime-4.3
        Note: you may need to restart the kernel to use updated packages.
```

* pip install 패키지 및 모듈 이름은 설치를 뜻하며 영구적이다.

```
!pip install pandas
!pip install numpy
!pip install matplotlib
!pip install scikit-learn
!pip install seaborn
!pip install torch==1.7.0+cpu torchvision==0.8.1+cpu torchaudio===0.7.0 -f
https://download.pytorch.org/whl/torch_stable.html
```

* 이번 분석을 위한 필요 패키지 및 모듈은 pandas, numpy, matplotlib,sckit-learn, pytorch이기 때문에 위와 같이 시작 전에 설치를 하면 된다.

- 모듈 설치 확인하기

```
pip list
```

```
In [6]: pip list
        cryptography                  2.9.2
        cycler                        0.10.0
        Cython                        0.29.21
        cytoolz                       0.10.1
        dask                          2.20.0
        DateTime                      4.3
        decorator                     4.4.2
        defusedxml                    0.6.0
        diff-match-patch              20200713
        distributed                   2.20.0
        docutils                      0.16
```

* pip list를 실행하면 현재 설치된 패키지 목록을 볼 수 있다. 직전에 설치한 'datetime'이 목록에 있는 것을 확인 가능

② 패키지 및 모듈 임포트 하기

(예시) 다양한 임포트 방법으로 'Welding Data Set_01.xlsx'를 읽어보기

i. import

import를 사용하여 해당 모듈 전체를 임포트한다.

```
import pandas
pandas.read_excel('./Welding Data Set_01.xlsx')
```

pandas를 임포트를 하고 '.'을 사용하여 pandas 의'read_excel' 함수를 이용하여 'Welding Data Set_01.xlsx' 파일을 불러온다.

ii. from, import

해당 모듈에서 특정한 타입만 임포트한다.

예) pandas에서 'read_csv'만 임포트

```
from pandas import read_excel
read_excel('./Welding Data Set_01.xlsx')
```

from, import를 사용해서 필요한 함수만 import로 사용할 수 있다.

iii. * import

해당 모듈 내에 정의된 모든 것을 임포트 하나, 다른 모듈들과 섞일 수 있으므로 일반적으로 사용이 권장되지 않는다.

```
from pandas import *
```

iv. as

모듈 임포트할 때, 별명(alias)을 지정할 때 사용한다.

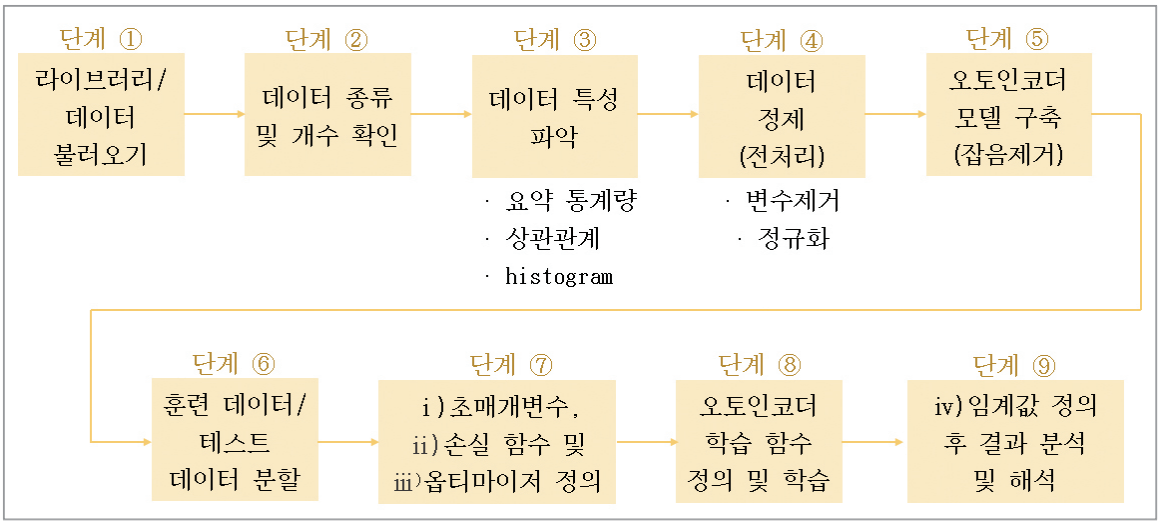
```
import pandas as pd
pd.read_excel('./Welding Data Set_01.xlsx')
```

'pandas'를'pd'로 별명 지정 후에는 'pd'로 불러올 수 있다.

※ 한번 설치한 모듈 및 패키지는 영구적이며 상황에 따라 버전 업그레이드가 필요하다. 또한, 분석, 시각화 등을 할 때 필요한 패키지를 임포트 한다.

2) 분석 단계별 프로세스

크게 아래의 흐름으로 데이터셋을 준비하고 모델을 구현할 수 있다.



[그림 13] 비지도 학습 적용을 위한 모델 구현 과정

i) **하이퍼 파라미터** : 모델을 만드는 데에 있어서 모델 자체가 학습할 수 있는 매개변수와 달리 사람이 직접 설정해 주어야 하는 변수

ii) **손실 함수** : 실제 값과 예측값(출력)의 차이를 수치화하는 함수로, 이 차이를 최소화하는 것이 목표

iii) **옵티마이저** : 손실 함수의 값을 줄여나가면서 네트워크를 업데이트하는 것으로, 어떤 옵티마이저를 사용하느냐에 따라 성능이 달라짐 iv)

v) **임계값** : 이상치 탐지를 위해 어떤 데이터를 이상치로 판별하는 기준이 되는 값이다. 판별기준은 손실값이다.

[단계 ①] 라이브러리/데이터 불러오기

①-1. 필요 데이터 다운로드 및 불러오기(import) 가이드

```
## 필요한 패키지 설치
from torch.utils.data import DataLoader
from torch.utils.data import TensorDataset
import torch
import numpy as np
import pandas as pd
import sklearn.metrics as metric
from sklearn import preprocessing
import torch.nn as nn
import matplotlib.pyplot as plt
```

[코드 1] 필요한 패키지 설치

```
##pandas의 read_excel함수를 사용하여 데이터 불러옴, 또한 index_col = "열 이름" 을 사용하여 인덱스 열 설정
welding_data = pd.read_excel("./Welding Data Set_01.xlsx", index_col ="idx")
welding_data.head()
#결과는 아래에서 확인 가능하다.
```

idx	Machine_Name	Item No	working time	Thickness 1(mm)	Thickness 2(mm)	weld current(kA)	weld Voltage(v)	weld time(ms)	weld force(bar)
1	Spot-01	65235-25800	2020-03-24	0.7	0.7	2.33	14.57	2.701	72.0
2	Spot-01	65235-25800	2020-03-24	0.7	0.7	2.36	14.57	2.701	72.0
3	Spot-01	65235-25800	2020-03-24	0.7	0.7	2.37	14.54	2.703	71.0
4	Spot-01	65235-25800	2020-03-24	0.7	0.7	2.37	14.54	2.703	72.0
5	Spot-01	65235-25800	2020-03-24	0.7	0.7	2.36	14.56	2.704	72.0

[코드 2] Pandas의 read_excel 함수를 이용한 데이터 불러오기

- Pandas의 read_excel 함수를 이용하여 데이터가 저장된 경로 찾고 입 력하여 데이터를 불러온다. 용접기 데이터의 경우 데이터 특성들 앞에 고유 번호가 붙어있어 첫 번째 열을 인덱스 열(목차)로 설정해준다. 그리고 데이터가 잘 읽어졌는지 데이터의 앞부분 4개까지 출력한다.

[단계 ②] 데이터 종류 및 개수 확인

②-1. 데이터 종류 및 개수 확인 가이드

```
## 데이터의 특성들이 어떤 값들을 가지고 있으며 몇 개씩 가지고 있는지 확인
for feature in welding_data:
    print(feature, welding_data[feature].value_counts()) #결과는 아래에서 확인 가능하다.
```

```
Machine_Name Spot-01    11939
Name: Machine_Name, dtype: int64
Item No 65235-25800    11939
Name: Item No, dtype: int64
working time 2020-04-02    2000
2020-03-31    1800
2020-03-27    1648
2020-03-30    1470
2020-03-25    1352
2020-03-24    1200
2020-03-26    1000
2020-04-03     800
2020-04-07     669
Name: working time, dtype: int64
Thickness 1(mm) 0.7    11939
Name: Thickness 1(mm), dtype: int64
Thickness 2(mm) 0.7    11939
Name: Thickness 2(mm), dtype: int64
weld force(bar) 2.31    1237
```

[코드 3] For 구문과 value_counts()함수를 이용한 특성 파악

[단계 ③] 데이터 특성 파악

③-1. Welding 제품/describe 함수를 통한 통계량 파악 가이드

- 용접 제품 'welding_data'의 클래스 변수가 존재하는 데이터의 변수별 요약 통계량을 확인한다.

```
welding_data.describe()  #결과는 아래에서 확인 가능하다.
```

	Thickness 1(mm)	Thickness 2(mm)	weld force(bar)	weld current(kA)	weld Voltage(v)	weld time(ms)
count	1.193900e+04	1.193900e+04	11939.000000	11939.000000	11939.000000	11939.000000
mean	7.000000e-01	7.000000e-01	2.787925	14.711208	2.704223	71.724123
std	1.113600e-13	1.113600e-13	1.455966	0.099000	0.024700	0.632049
min	7.000000e-01	7.000000e-01	1.740000	14.520000	2.464000	70.000000
25%	7.000000e-01	7.000000e-01	2.310000	14.610000	2.699000	71.000000
50%	7.000000e-01	7.000000e-01	2.340000	14.730000	2.702000	72.000000
75%	7.000000e-01	7.000000e-01	2.370000	14.750000	2.706000	72.000000
max	7.000000e-01	7.000000e-01	10.540000	15.070000	2.861000	73.000000

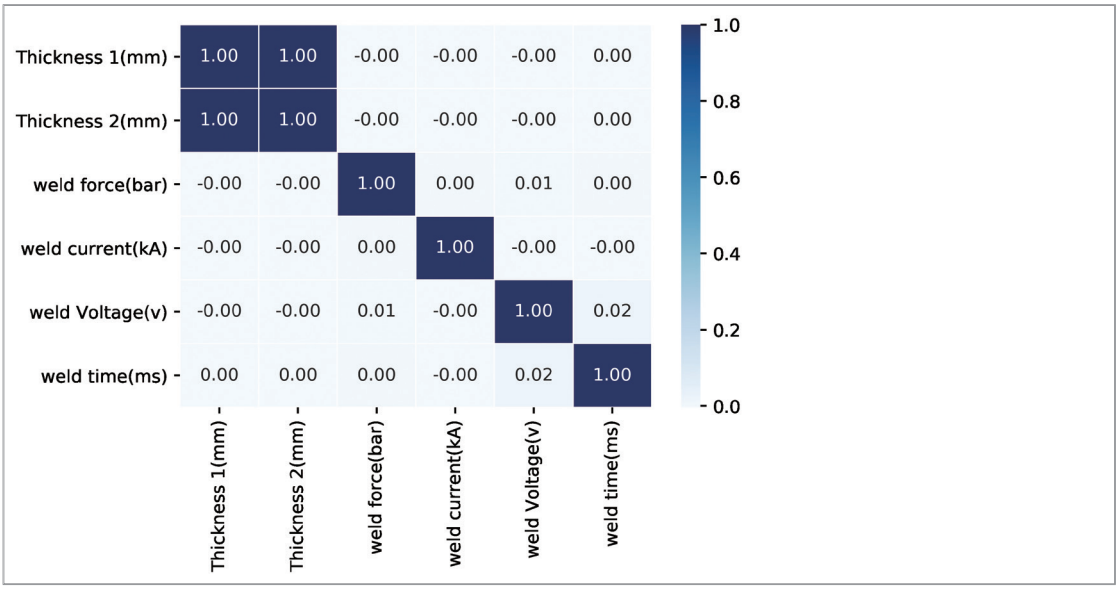
[코드 4] welding_data의 변수 별 요약 통계량

③-2. 용접 제품/corr 함수를 통한 변수 간 상관관계 파악 가이드

- 제품 'welding_data'의 클래스 변수가 존재하는 데이터의 변수 간 상관관계를 확인한다.

- 상관관계 분석을 통해 두 변수 간의 선형적인 관계 존재 여부를 확인할 수 있다. 분석에 사용된 상관 계수는 보편적으로 사용되는 피어슨 상관 계수로서 -1과 1 사이의 값을 가진다. 상관 계수의 절대값이 1에 가까울수록 변수 간의 선형관계가 강하다고 볼 수 있으며, 값이 양수일 때는 양적인 선형관계, 값이 음수일 때는 음적인 선형관계를 가진다. 상관관계 분석은 비선형적인 관계를 확인하기 어렵지만, 변수 간의 선형관계를 정량적으로 확인할 수 있다는 장점이 있다. 상관계수의 절댓값이 0.1과 0.3 사이이면 약한 선형관계, 0.3과 0.7 사이이면 뚜렷한 선형관계, 0.7 이상인 경우 강한 선형관계를 나타낸다고 이해할 수 있다.

```
welding_data.corr()  #결과는 아래에서 확인 가능하다.
```



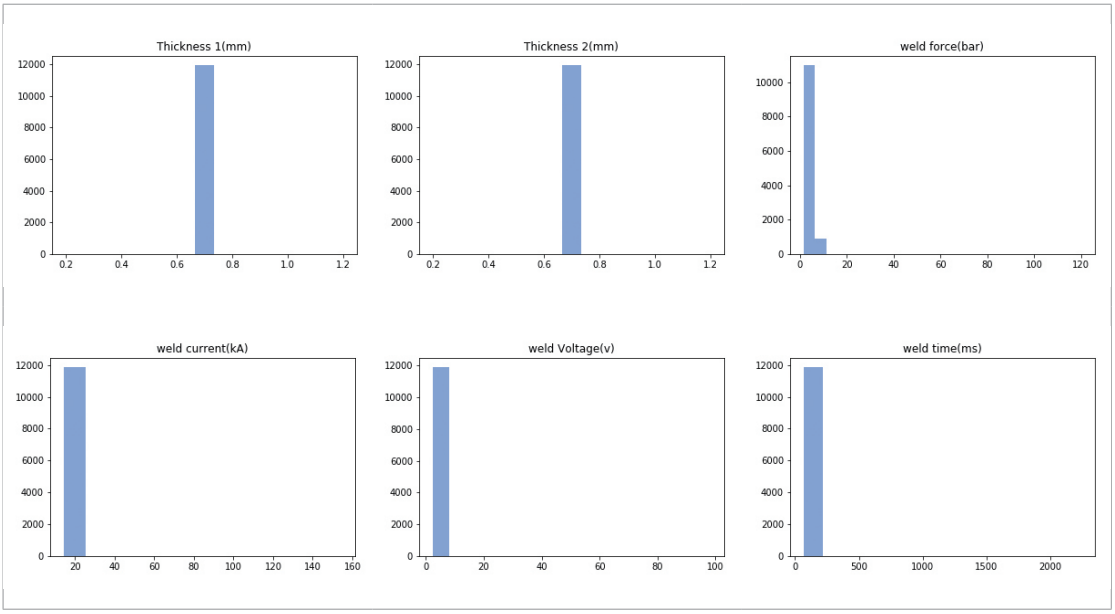
[코드 5] welding_data 상관관계

- 이 그림은 변수 간의 상관계수를 시각화한 히트 맵으로서, 각 칸의 값은 해당되는 X축, Y축의 변수 간의 상관계수이며, 우측의 범례처럼 상관계수 값마다 다른 색을 가진다. 히스토그램은 도수분포표를 그래프로 나타낸 것으로서, X축은 특정 변수의 구간이며 Y축은 구간에 해당하는 데이터의 개수이다. 히스토그램을 통한 시각화는 하나의 변수에 대한 데이터의 분포를 어림잡아 볼 수 있다는 장점이 있다. 히스토그램의 패턴이 단봉, 쌍봉인지 등의 모달리티(modality)와 왼쪽 혹은 오른쪽으로 쏠려있는 정도인 비대칭도 등을 확인 할 수 있다.

③-3. 용접 제품/Histogram을 통한 변수별 데이터 파악 가이드

- 제품 'welding_data'의 클래스 변수가 존재하는 데이터의 변수별 히스토그램을 확인한다. 변수별로 막대 그래프의 개수를 설정해주고, 변수마다 해당 막대 그래프의 개수가 할당될 수 있도록 index와 value로 설정해준다. matplotlib 패키지의 pyplot(plt로 표현)을 이용하여 히스토그램을 그려준다. subplot() 함수를 이용하여 한꺼번에 모든 변수의 히스토그램 결과를 볼 수 있도록 한다.

```
## 히스토그램을 그리기 위한 for 구문
b = [15,15,25,13,17,15,35]
for index, value in enumerate(welding_data.columns):
    plt.figure(index)
    plt.hist(welding_data[value], bins = b[index], facecolor = (144 /255,171 /255,221 /255))
    plt.title(value)
    plt.savefig("Welding/"+ value +'.png')
```



[코드 6] 데이터 histogram

[단계 ④] 데이터 정제 (전처리)

- 데이터 품질검사, 데이터 정제(전처리)·가공 실습 내용

※ 이미지 데이터인 경우 고품질 이미지 데이터셋을 만들기 위한 변환 및 특징(feature) 추출 방안 (ex)데이터 라벨링(data labeling))

④-1. 불필요 데이터 제외 가이드

```
## 용접기 데이터에서 필요없는 부분(생산품목, 작업시간, 소재두께)들을 제외
new_welding_data = welding_data.iloc[:, 5:]
new_welding_data.head()   #결과는 아래에서 확인 가능하다.
```

idx	weld force(bar)	weld current(kA)	weld Voltage(v)	weld time(ms)
1	2.33	14.57	2.701	72.0
2	2.36	14.57	2.701	72.0
3	2.37	14.54	2.703	71.0
4	2.37	14.54	2.703	72.0
5	2.36	14.56	2.704	72.0

[코드 7] Pandas의 iloc함수를 이용한 데이터 슬라이싱

④-2. MinMaxScaler를 통한 데이터 정규화 가이드

```
## sklearn의 preprocessing모듈에 들어있는 MinMaxScaler함수를 이용해 정규화 적용
scaler = preprocessing.MinMaxScaler()
scaler.fit(new_welding_data)
scaled_data = scaler.transform(new_welding_data)
```

[코드 8] Sklearn의 MinMaxScaler 함수를 이용한 최소 최대 정규화

- 각 특성을 sklearn 패키지의 preprocessing 모듈에 들어있는 MinMaxScaler 함수를 사용하여 최소-최대 정규화를 적용한다. 여기서 sklearn같은 큰 패키지 같은 경우 그 안에 여러 모듈을 가지고 있으며 각 모듈은 또 여러 함수를 가지고 있다.

[단계 ⑤] 오토인코더 모델 구축 (잡음제거)

⑤-1. 오토인코더 모델 구축 가이드

- AI 분석모델 구축을 위한 방법론(알고리즘) 적용 및 학습 네트워크 구축 실습

input_size	입력의 크기, 숫자로 입력을 받음
hidden_size	신경망층의 크기, 리스트 형태로 입력을 받음
output_size	출력의 크기, 숫자로 입력을 받음
nn.Linear()	완전연결망
nn.RReLU()	Leaky Rectified Linear Unit 활성 함수
nn.RReLU()	인공신경망을 담는 컨테이너, 여기에 각 신경망 층들을 더해주어 신경망을 만듦

```
## AutoEncoder 클래스 구현
class AutoEncoder(nn.Module):
    def __init__(self, input_size, hidden_size, output_size):
        super(AutoEncoder, self).__init__()
        ## initialize
        self.input_size = input_size
        self.hidden_size = hidden_size
        self.output_size = output_size
        ##오토인코더 구현
        self.AutoEncoder = nn.Sequential(
        ## 인코더 부분
        nn.Linear(input_size, hidden_size[0]),
        nn.RReLU(),
        nn.Linear(hidden_size[0], output_size),
        nn.RReLU(),
        ## 디코더 부분
        nn.Linear(output_size, hidden_size[0]),
        nn.RReLU(),
        nn.Linear(hidden_size[0], input_size)
        )
    def forward(self, inputs):
        output =self.AutoEncoder(inputs)

        return output
```

[코드 9] Pytorch를 이용한 오토인코더 구현

- 사용할 오토인코더 모델은 Pytorch package를 사용하여 구현한다. 여기서 Pytorch package는 사용자가 직접 각 신경망 층을 어떤 식으로 구성할지 조절할 수 있다. 여기서는 인코더 부분에서 완전연결층 2개와 디코더 부분에서 완전연결층 2개로 구성한다. 완전연결층이란 한 층의 모든 노드와 다음 층의 모든 노드가 연결된 층을 말한다. 노드는 데이터의 특성 하나를 담을 수 있는 단위이다. 코드 구현 과정을 자세히 설명하자면, Class AutoEncoder(nn.Module)을 사용해 오토인코더의 클래스를 정의해준다. 클래스란 파이썬 프로그래밍의 기본 단위의 객체이며 쉽게 말해 변수와 함수를 포함하는 집단이라고 생각하면 된다. 그리고 클래스를 정의해주면서 같이 정해주어야 할 입력 값으로 input_size, hidden_size, output_size를 __init__(self, input_size, hidden_size, output_size)와 같이 정의해준다.
- 그런 다음 앞에서 정의한 값들을 self.input_size = input_size와 같이 클래스 안의 변수로 만들어준다. 그리고 nn.Sequential 함수를 사용하여 오토인코더 안에 들어갈 신경망층 들을 담을 컨테이너를 만들어준다. self.AutoEncoder = nn.Sequential() 이 컨테이너 안에 자기가 원하는 신경망들을 이어서 정의해주면 되는데, 여기서는 인코더 부분에서 완전연결망(nn.Linear) 2개와 그리고 그에 따른 Leaky Rectified Linear Unit 활성 함수, nn.RReLU(),를 사용하여 정의해주었다. 활성 함수는 각 노드에서 입력값들을 어떤 식으로 출력을 해줄지 정해주는 함

수이다. 디코더 부분에서도 마찬가지로 인코더와 같이 완전연결망 2개와 Leaky Rectified Linear Unit 활성 함수를 사용하였으며, 출력 부분에서는 활성 함수를 사용하지 않았다.

[단계 ⑥] 훈련 데이터/테스트 데이터 분할

⑥-1. 훈련 데이터/테스트 데이터 분할 가이드

훈련 데이터(Train_data)	**데이터 개수 :** 8470개
	불량품 개수 : 28개
테스트 데이터(Test_data)	**데이터 개수 :** 3469
	불량품 개수 : 11개

```
##기존이 데이터를 텐서 형태로 변환, 그리고 훈련세트와 테스트세트로 나눔
train_data = torch.Tensor(scaled_data[:8470]) ## [:8470] 처음부터 8469번까지의 데이터를 훈련세트로 지정
print(len(train_data))
test_data = torch.Tensor(scaled_data[8470:]) ## [8470:] 8470번째 데이터부터 끝까지를 테스트세트로 지정
print(len(test_data))  #결과는 아래에서 확인 가능하다.
```

```
8470
3469
```

[코드 10] Pytorch의 ToTensor 함수를 이용한 입력 변환

- 기존에 불러왔었던 데이터를 pytorch의 Tensor함수를 사용하여 오토인코더에 들어갈 입력 값으로 넣기 위한 텐서로 변환한다. 여기서 텐서란 간단하게 데이터의 배열이다. 위와 같은 pytorch로 구현한 오토인코더는 텐서의 형태로만 입력을 받는다. 그리고 기본적인 인덱스 슬라이싱(자르기)을 이용해 훈련세트와 테스트세트로 나눈다.

[단계 ⑦] 하이퍼파라미터, 손실 함수 및 옵티마이저 정의

⑦-1. 하이퍼파라미터, 손실 함수 및 옵티마이저 정의 가이드

```
## 훈련 하이퍼파라미터
epoch =50
batch_size =64
lr =0.01
## 모델 하이퍼파라미터
input_size =len(train_data[0])
hidden_size = [3]
output_size =2
## 손실 함수로 제곱근 오차 사용
criterion = nn.MSELoss()
## 매개변수 조정 방식으로 Adam사용
optimizer = torch.optim.Adam
##오토인코더 정의
AutoEncoder = AutoEncoder(input_size, hidden_size, output_size)
```

[코드 11] 하이퍼파라미터에 대한 정의

nn.MSELoss()	평균 제곱근 오차 계산 함수
torch.optim.Adam	매개 변수 조정 방식으로 Adam으로 설정
AutoEncoder(input, hidden, output)	앞에서 구현한 오토인코더 클래스를 정의

[그림 14] 에포크와 배치사이즈

- 하이퍼파라미터(hyper-parameter)에 대한 정의. 하이퍼파라미터란 모델을 만드는 데에 있어서 모델 자체가 학습할 수 있는 매개변수와는 달리 사람이 직접 설정해 주어야 하는 변수이다. 여기서는 학습 하이퍼파라미터로 에포크, 배치사이즈, 학습률 그리고 모델 하이퍼파라미터로 입력 크기, 신경망의 크기, 출력 크기, 손실 함수, 매개변수 조정 방법(Optimizer)을 정의했다. 배치사이즈는 배치 학습에서 사용되는 데이터를 작은 묶은 단위이다.
- 이렇게 작은 묶음 단위로 모델에 차례대로 입력시켜 학습하는 것이 한 번에 모든 데이터를 학습하는 것보다 효율적이기 때문에 배치 학습 방식을 적용한다. 또한, 에포크는 전체 데이터를 한 번 학습하는 단위로서, 10 에포크일 경우 전체 데이터를 10번 학습하는 것이다. 마지막으로 학습률은 매개변수를 조정하는 정도를 나타내는 계

수이다. 여기서는 손실 함수로 평균 제곱근 오차와 조정 방법으로 Adam optimizer를 사용했다. 평균 제곱근 오차는 각 데이터의 입력값과 출력값의 차이를 제곱하여 평균을 낸 다음 제곱근을 적용해주는 계산방식이다. Adam optimizer는 여러 매개변수 조정 방법 중 보편적으로 자주 쓰이는 기울기를 계산하여 매개변수를 조정해주는 경사 하강법을 업그레이드한 방식이다.

- 앞에서 말한 하이퍼파라미터들을 직관적으로 epoch = 50과 같이 정의해준다. 또한, 입력값의 길이는 len(train_data[0])과 같이 데이터 하나의 길이를 측정하여 정의해준다. 손실 함수와 매개변수 조정 방식 같은 경우는, pytorch에서 제공하는 nn.MSELoss()함수와 torch.optim.Adam함수를 사용해 criterion = nn.MSELoss()와 optimizer = torch.optim.Adam같이 정의해주어 손실 값 계산에 사용할 수 있게 해준다. 마지막으로, 정의한 하이퍼파라미터를 사용하여 앞에서 정의한 오토인코더 클래스를 정의해준다. AutoEncoder = AutoEncoder(input_size, hidden_size, output_size).

[단계 ⑧] 오토인코더 학습 함수 정의 및 학습

⑧-1. 학습 함수 정의 가이드

AutoEncoder.parameters( )	현재 오토인코더에 내재되어 있는 매개 변수
DataLoader(data, batch_size, shuffle)	Pytorch에서 제공하는 함수로, 데이터를 배치 사이즈로 나누며 shuffle은 훈련과정에서 데이터를 섞는지에 대한 여부
optim.zero_grad( )	매개변수를 0으로 초기화시켜주는 함수
loss.backward( ) optim.step( )	매개변수를 손실 값을 기준으로 조정해주는 함수
{}{}.format(a, b)	문장을 출력해줄 경우, {}자리에 특정 변수값(a,b가 가지는 값)을 넣어 출력해주는 함수
running_loss += loss.item( )	데이터를 학습하면서 발생하는 손실 값을 추적하기 위한 손실 값 기록. running_loss를 통해 현재 손실값이 얼만지 알 수 있음.

```
## 학습 함수에 대한 정의
def train_net(AutoEncoder, data, criterion, epochs, lr_rate =0.01):
## Optimizer에 대한 정의
    optim = optimizer(AutoEncoder.parameters(), lr =lr_rate)
## 배치 학습을 시키기 위한 데이터 변환
    data_iter = DataLoader(data, batch_size =batch_size, shuffle =True)
## 에포크 학습
    for epoch in range(1, epochs +1):
        running_loss =0.0
        for x in data_iter:
            ## 매개변수 0으로 초기화
            optim.zero_grad()
            output = AutoEncoder(x)
            ## 입력값과 출력값간의 차이인 손실값
            loss = criterion(x, output)
            ## 손실값을 기준으로 매개변수 조정
            loss.backward()
            optim.step()
            running_loss += loss.item()

        ## 각 에포크마다 손실 값 표기
        print("epoch: {}, loss: {:.2f}".format(epoch, running_loss))
    return AutoEncoder
```

[코드 12] 모델 학습 함수 정의

- 오토인코더를 학습시키기 위한 학습 함수 정의. 학습을 에포크만큼 반복시키기 위해 for 구문 사용한다. For 구문 안에서 첫 번째로 매개변수 조정 방식을 오토인코더의 매개변수와 학습률을 넣어 정의해준다, optim = optimizer(AutoEncoder.parameters(), lr = lr_rate. 다음으로 배치 학습을 시키기 위한 데이터를 DataLoader()함수를 사용하여 정의해준다, data_iter = DataLoader(data, batch_size = batch_size, shuffle = True). 다음으로 For 구문을 중첩하여 에포크마다 손실 값들을 출력하게 해주며, 배치마다 학습하며 매개변수들을 조정해준다. for x in data_iter:부터 running_loss += loss.item() 코드까지 매개변수를 조정해주는 과정이며, print("epoch:{}.loss: {:2f}".format(epoch, running_loss))가 손실 값을 출력해주는 코드이다.

⑧-2. 오토인코더 학습 과정 가이드

- AI 분석모델 학습 내용 (학습-검증-평가 비율, 학습조건, 모델 튜닝)

```
## 학습 함수를 이용한 오토인코더 학습
AutoEncoder = train_net(AutoEncoder, train_data, criterion, epoch, lr)
```

```
epoch: 1, loss: 6.84
epoch: 2, loss: 2.39
epoch: 3, loss: 2.29
epoch: 4, loss: 2.15
epoch: 5, loss: 1.55
epoch: 6, loss: 0.83
epoch: 7, loss: 0.76
epoch: 8, loss: 0.75
epoch: 9, loss: 0.75
epoch: 10, loss: 0.74
epoch: 11, loss: 0.74
epoch: 12, loss: 0.74
epoch: 13, loss: 0.74
epoch: 14, loss: 0.73
epoch: 15, loss: 0.65
epoch: 16, loss: 0.49
epoch: 17, loss: 0.40
epoch: 18, loss: 0.37
epoch: 19, loss: 0.36
epoch: 20, loss: 0.34
epoch: 21, loss: 0.34
epoch: 22, loss: 0.33
epoch: 23, loss: 0.33
epoch: 24, loss: 0.32
epoch: 25, loss: 0.32
epoch: 26, loss: 0.31
epoch: 27, loss: 0.33
epoch: 28, loss: 0.31
epoch: 29, loss: 0.32
epoch: 30, loss: 0.32
...
```

[코드 13] 학습 함수를 이용한 오토인코더 학습

- 이제 앞에서 정의해준 하이퍼파라미터와 학습 함수를 이용하여 오토인코더를 학습시켜준다, AutoEncoder = train_net(AutoEncoder, train_data, criterion, epoch, lr). 학습 함수에서도 정의했던 것처럼 오토인코더를 학습하며 에포크마다 손실 값을 위와 같이 출력해준다.

[단계 ⑨] 임계값 정의 후 결과 분석 및 해석

⑨-1. 임계값 정의 가이드

```
## 훈련세트의 손실값 이용한 임계값 정의
train_loss_chart = []
for data in train_data:
    output = AutoEncoder(data)
    loss = criterion(output, data)
    train_loss_chart.append(loss.item())

threshold = np.mean(train_loss_chart) + np.std(train_loss_chart)*8
print("Threshold :", threshold)  #결과는 아래에서 확인 가능하다.
```

```
Threshold : 0.08386612018531674
```

[코드 14] 훈련세트의 손실값들을 이용한 임계값 정의

⑨-2. 결과 분석 가이드

- 분석 결과값 도출

```
## 훈련세트의 손실값 이용한 임계값 정의
test_loss_chart = []
for data in train_data:
    output = AutoEncoder(data)
    loss = criterion(output, data)
    test_loss_chart.append(loss.item())

outlier = list(test_loss_chart >= threshold)
outlier.count(True)  #결과는 아래에서 확인 가능하다.
```

```
13
```

[코드 15] 테스트 데이터에 대한 불량품 예

- 위에서 정한 임계값과 훈련한 오토인코더로 테스트 데이터에 적용한다. For 구문을 사용하여 테스트 데이터를 하나씩 오토인코더에 입력시키며 손실 값 들을 측정한다, output=AutoEncoder(data)와 loss = criterion(output, data). 각 데이터의 손실 값들을 측정하고, test_loss_chart라는 표에 test_loss_chart.append(loss.item()) 과 같이 기록해준다. 마지막으로 기록한 표에서 임계값 이상인 손실 값들을 추출한다. outlier = list(test_loss_chart >= threshold)와 outlier.count(True)를 사용해 불량품이 몇 개인지 출력한다. 위에서는 테스트 데이터에서 13개의 불량품이 나온 것을 알 수 있다.

⑨-3. 분석 결과에 대한 해석

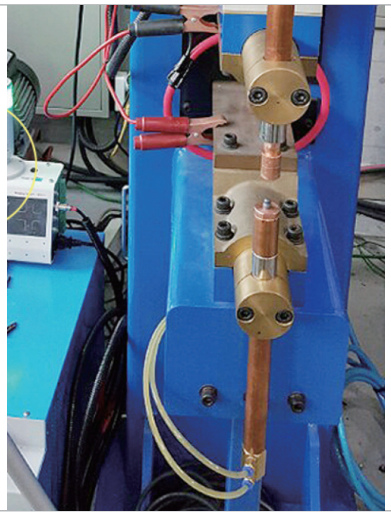
◦ 분석 결과에 대한 논의 및 해석(implication)

- 용접기 데이터의 경우에는 모든 데이터가 개별로 목표치가 주어지지 않아 모델의 정량적 평가는 힘들며, 일별 불량품 개수를 비교한 정성적 평가만 가능하다. 현재 테스트 데이터에서는 3469개의 제품 중 11개의 제품이 불량이며, 오토인코더의 결과는 13개의 제품이 불량품이라 판별한다. 이는 일별 불량품 개수 예측의 측면으로 본다면 나쁘지 않은 결과일 수 있으나, 개별 제품의 불량품 예측이 맞았는지 틀렸는지는 알 수가 없어 정확한 모델의 성능을 측정할 수 없다. 즉, 일별 불량품 개수에 대한 예측에 대한 성능은 2개 정도의 오차를 보이며, 정량적 개별 제품에 대한 불량품 판별의 성능을 보기 위해서는 목표치가 주어진 데이터가 있어야 한다. 또한, 데이터 특성적인 면에서 현재 데이터 같은 경우 딥러닝을 적용하기에는 데이터의 특성이 부족해 보인다. 따라서 딥러닝, 기계 학습의 분석적 이점들을 살리기 위해서는 좀 더 많은 데이터 특성의 확보가 필요해 보인다.

3. 유사 타현장의 「용접기 AI 데이터셋」 분석 적용

3.1 본 분석이 적용 가능한 제조현장 소개

- 주로 자동차 부품산업에 적용되며, 승용차 1대 조립 시 총용접 수 3,400 ~3,800개의 타점 중 98% 비중을 차지할 정도로 널리 사용되고 있다. 기업 중 작업자 의존하여 용접하는 공정이 있지만, 점차 로봇 기반 용접공정 자동화를 하는 기업들이 늘어나고 있다.
- 작업자에 의한 용접공정은 용접품의 위치 정보 및 용접 후 검사 결과 데이터 수집이 쉽지 않으나 자동화 공정의 경우 로봇 티칭에 따른 좌표정보와 PLC의 전류, 통전시간, 가압력 데이터 수집과 검사 자동화에서 결과를 용이하게 수집하는 공정에서는 본 분석을 적용하기 매우 유리하다고 볼 수 있다.



[그림 15] 용접 설비 공정

3.2 본 「용접기 AI 데이터셋」 분석을 원용하여 타 제조현장 적용 시, 주요고려사항

- 상기 데이터셋의 스킴은 그대로 활용할 수 있으며, 데이터 수집 시 기존 시스템과의 인터페이스 방식을 고려해야 한다.
- 두 번째는 용접 후 검사 데이터 수집 시 불량 유형별 관리하는지 파악이 필요하다. 대부분 검사를 맨눈으로 하거나 일부 비전으로 하고 있지만, 양품, 불합격 관리 위주로 하며, 불량 유형별 관리는 수기로 관리하고 있고, 이러한 데이터와 용접 설비 조건 데이터와 데이터 매핑이 되어 있지 않은 경우가 대부분이며, 이런 경우 검사 자동화를 통해 검사를 자동 판정하고, 판정 결과를 공정 설비 조건과 매핑하여 상관관계 정의를 하여야 한다.

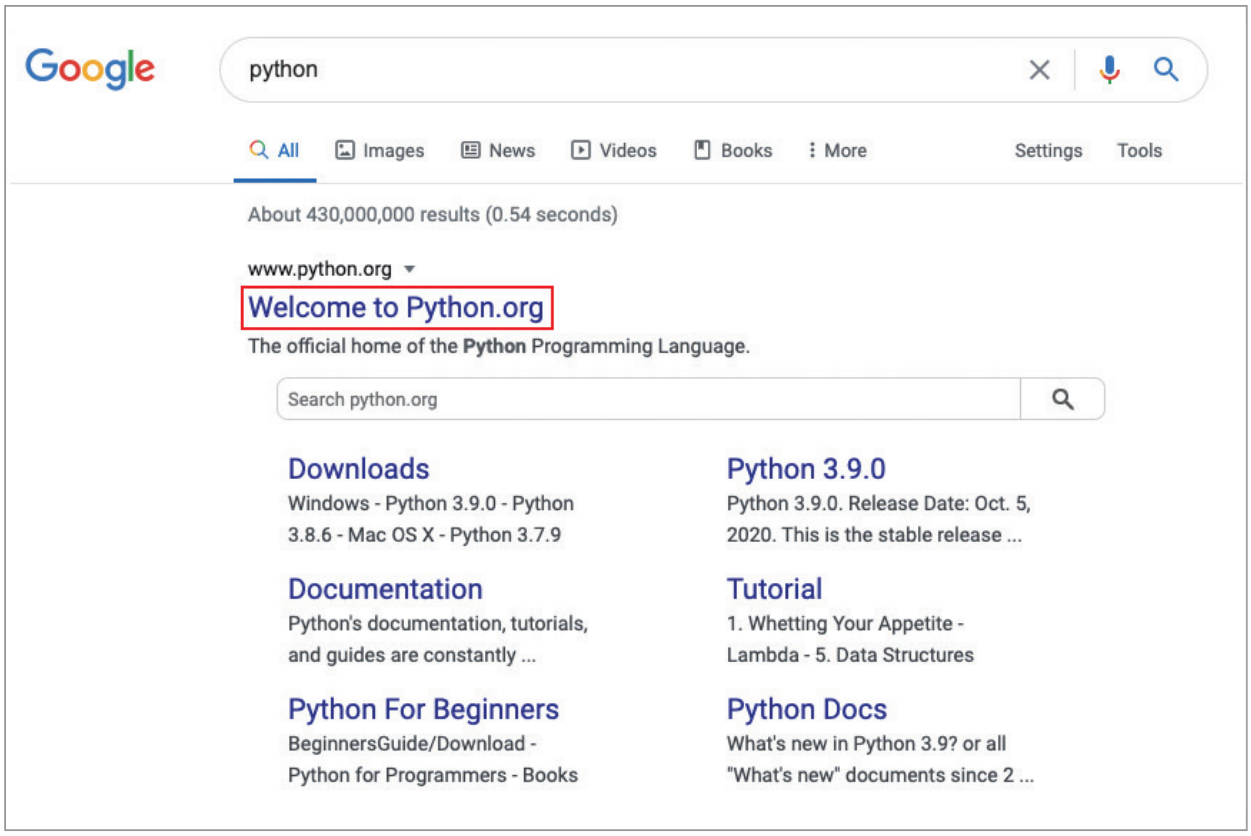
부록 파일 설치방법

1. 파이썬(python) 설치

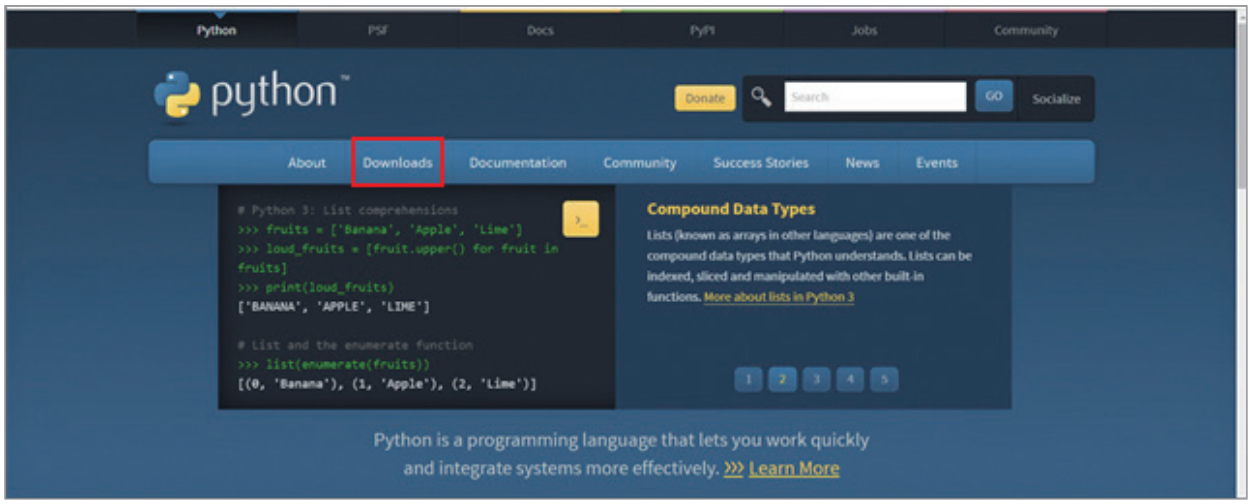
파이썬이란, 컴퓨터 언어 및 데이터 분석에 활발하게 쓰이는 도구입니다. 데이터 분석을 위해서 다운로드 및 설치가 간편하고 활용도가 높은 파이썬을 설치하고 적용하여 봅니다.

① google.com 등의 검색 엔진에 'python'을 검색

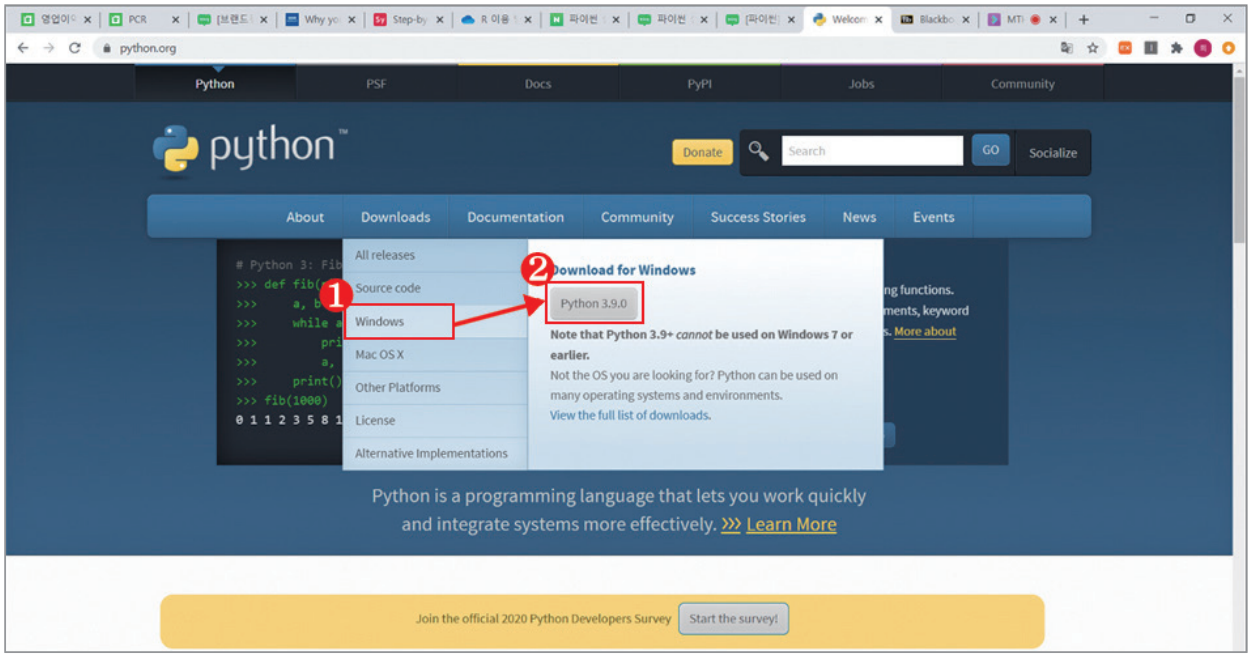
② 제일 처음에 보이는 'Welcome to python.org'를 클릭



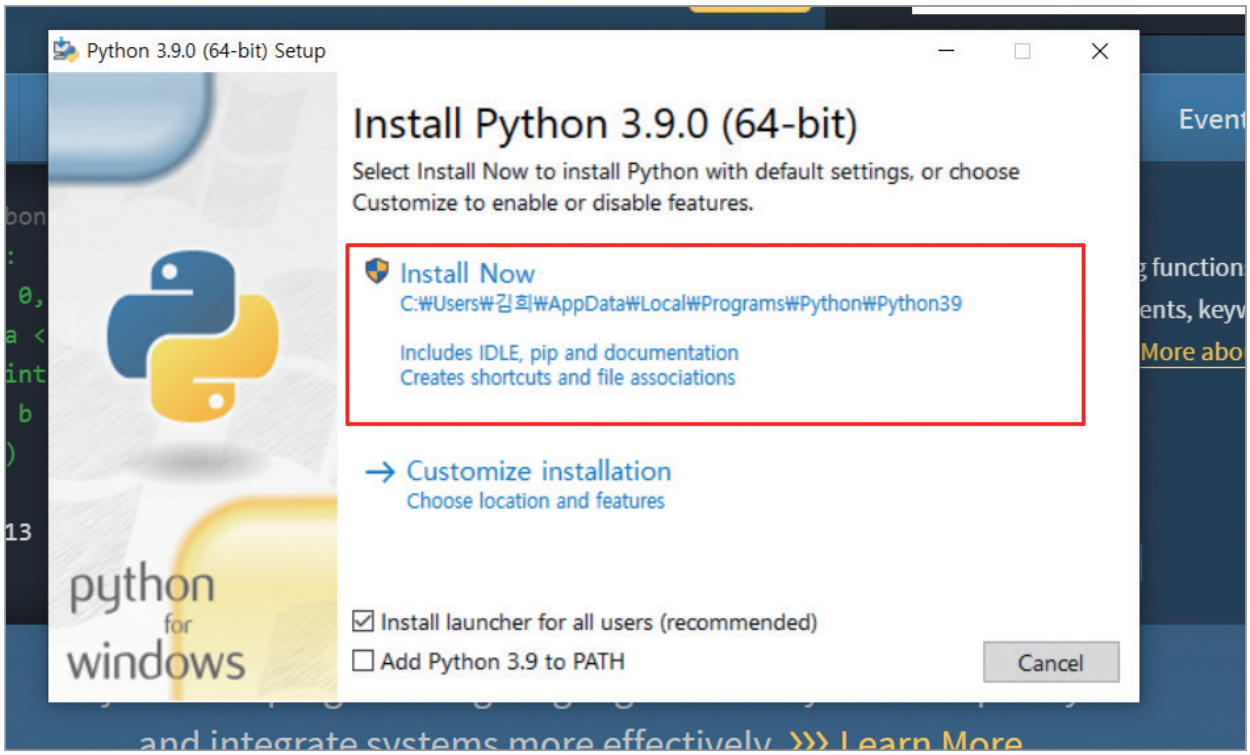
③ 클릭해서 보이는 페이지 정면의, 왼쪽 2번째 'Downloads' 클릭



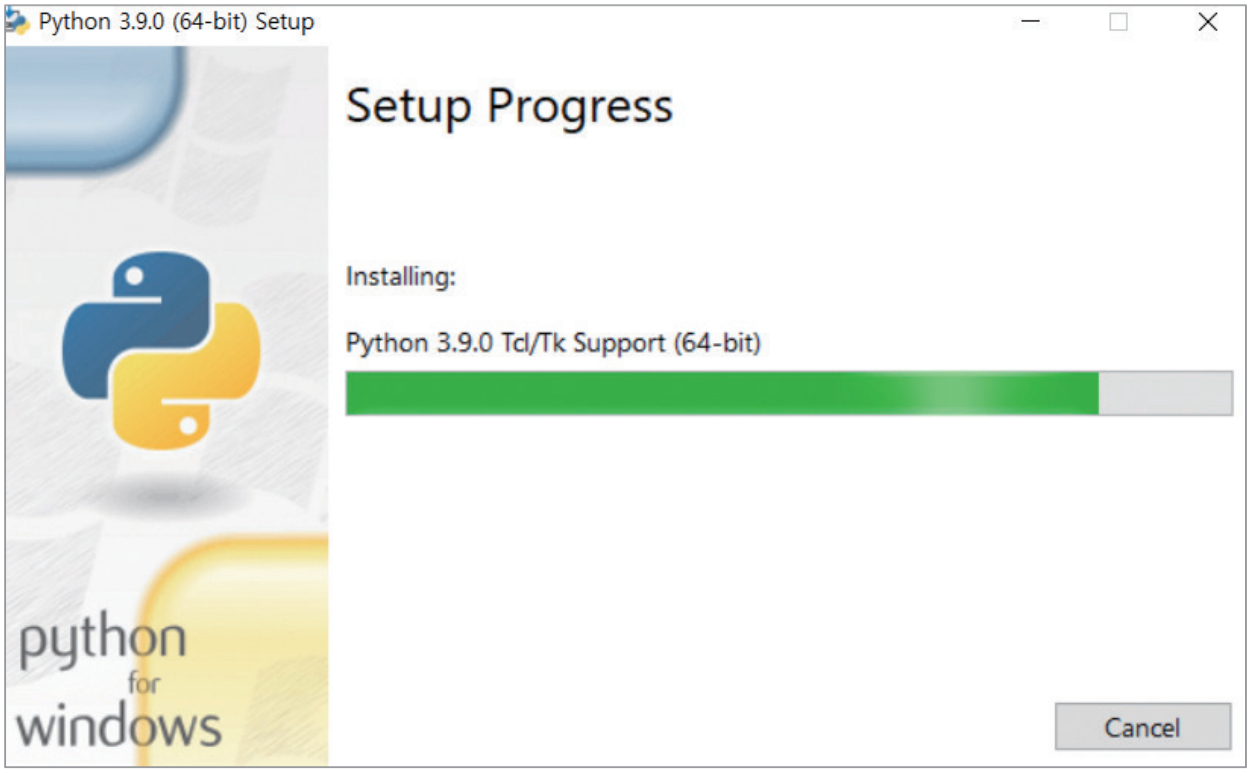
④ 위에서 3번째, Windows 탭을 선택한 후, python3.9.0 다운로드
(python3.9.0은 숫자가 업데이트 될 수 있습니다)



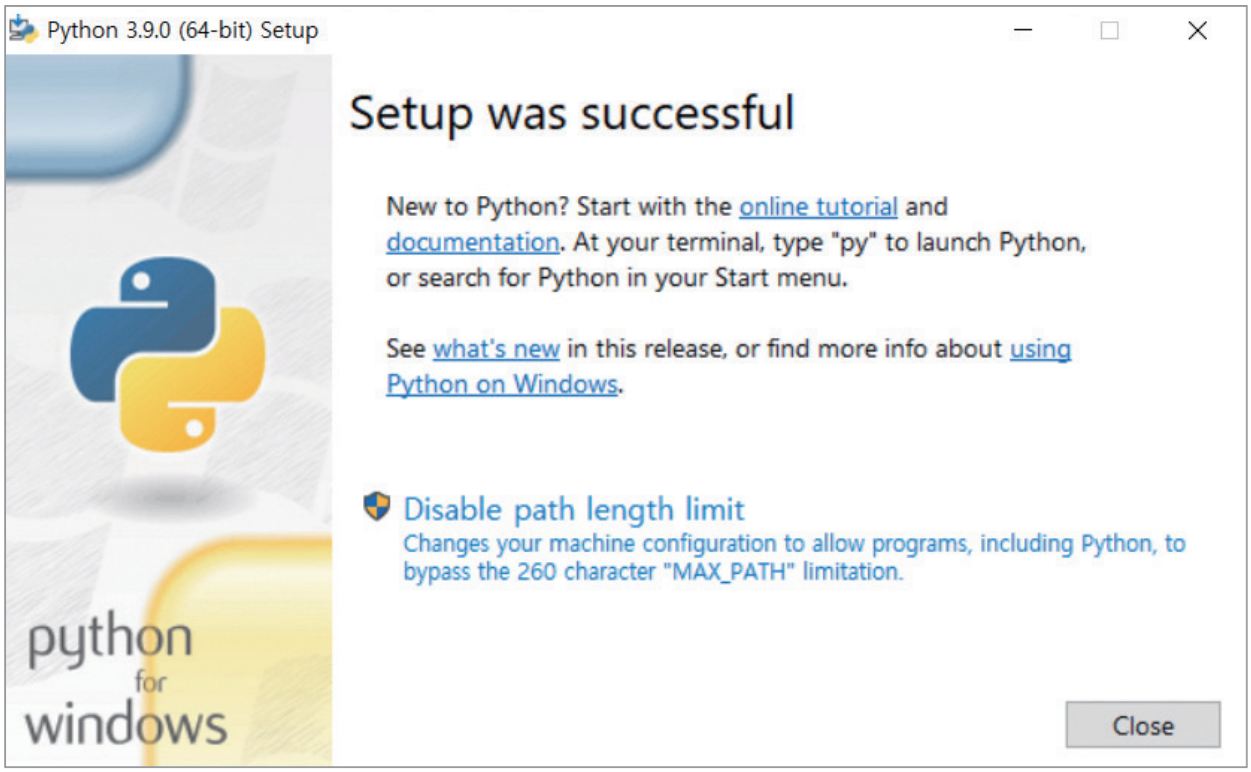
⑤ 아래와 같은 설치창이 뜨면, 'Install Now'를 클릭



⑥ 아래와 같은 설치 진행창이 완료가 될 때까지 유지



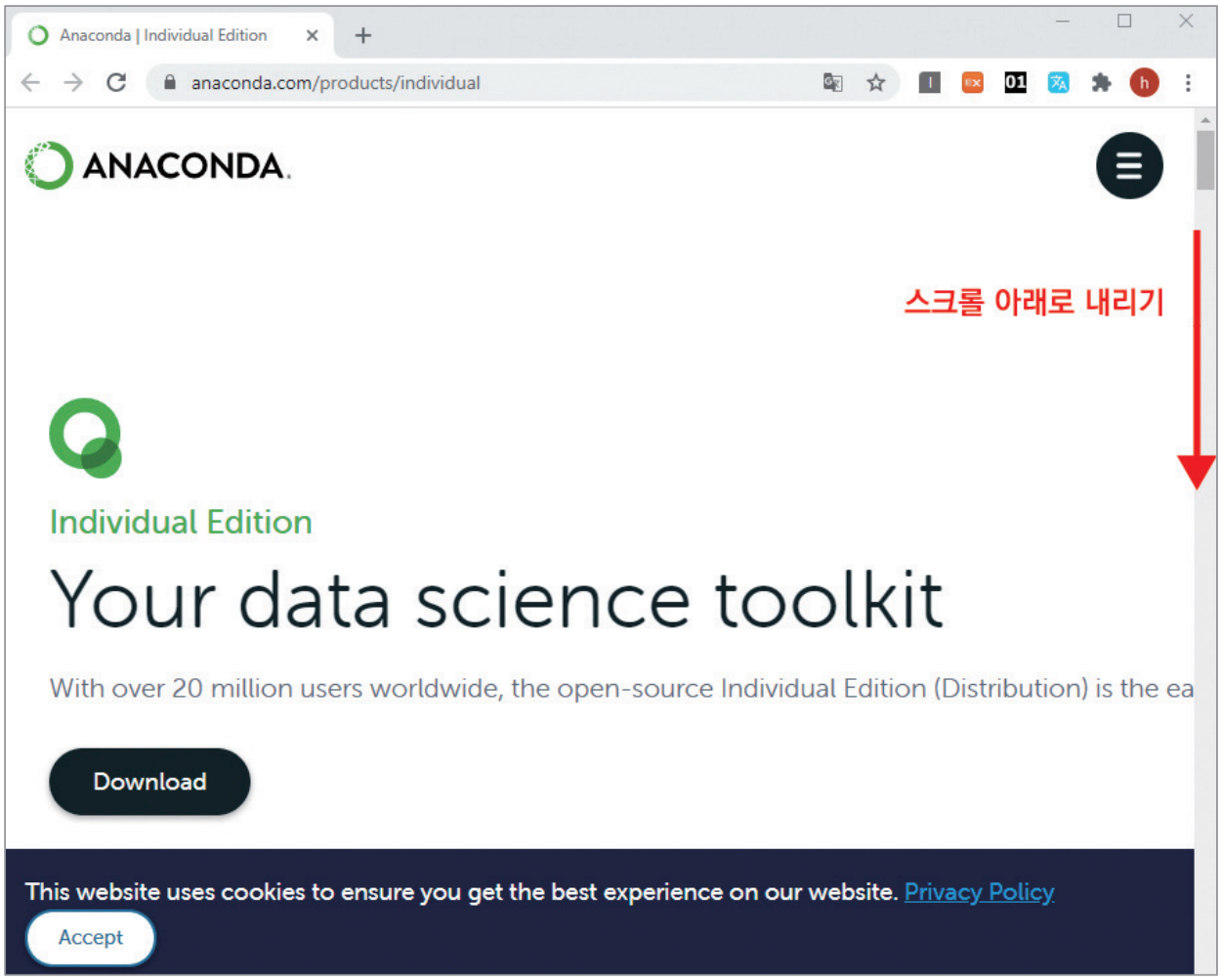
⑦ 완료가 되면 아래와 같은 창이 뜨는 것 확인 후 종료 **[설치완료]**



2. 아나콘다(anaconda) 설치

아나콘다란? 파이썬과 같은 분석 도구를 사용할 때 필요한 고급 기능 및 분석을 보조하는 도구입니다. 아나콘다를 설치함으로써 많은 기능들을 바로 쓸 수 있고, 결과물들 또한 쉽게 볼 수 있는 기능을 지원합니다. 아나콘다를 설치하고 분석을 할 수 있는 환경을 만들어봅니다.

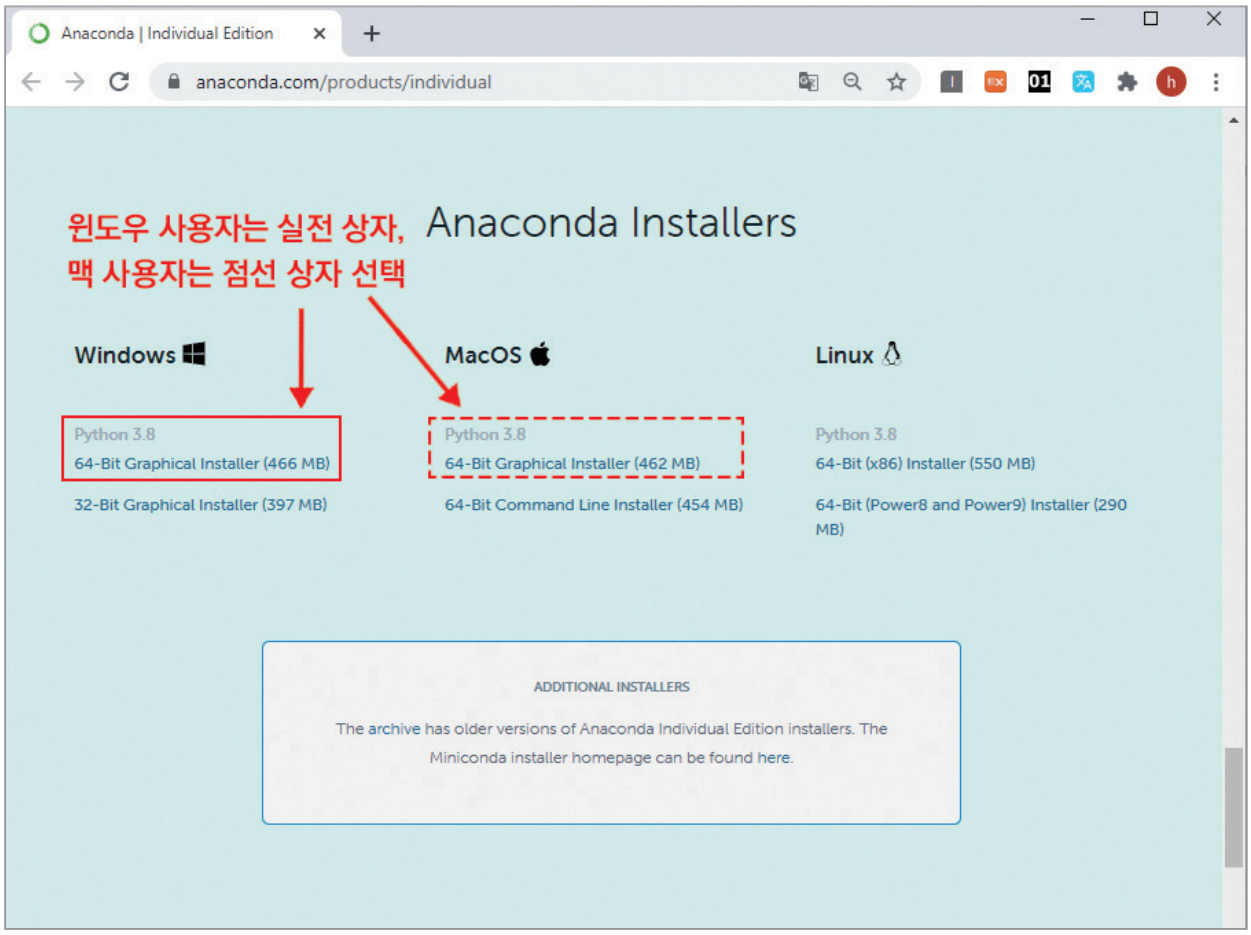
① https://www.anaconda.com/distribution/ 로 접속 후 스크롤 내림



② 스크롤을 다음과 같은 화면이 나올 때 까지 아래로 내린 후, 컴퓨터 사용환경에 맞는 파일 다운받기 (본 부록은 Windows 설치 기준)

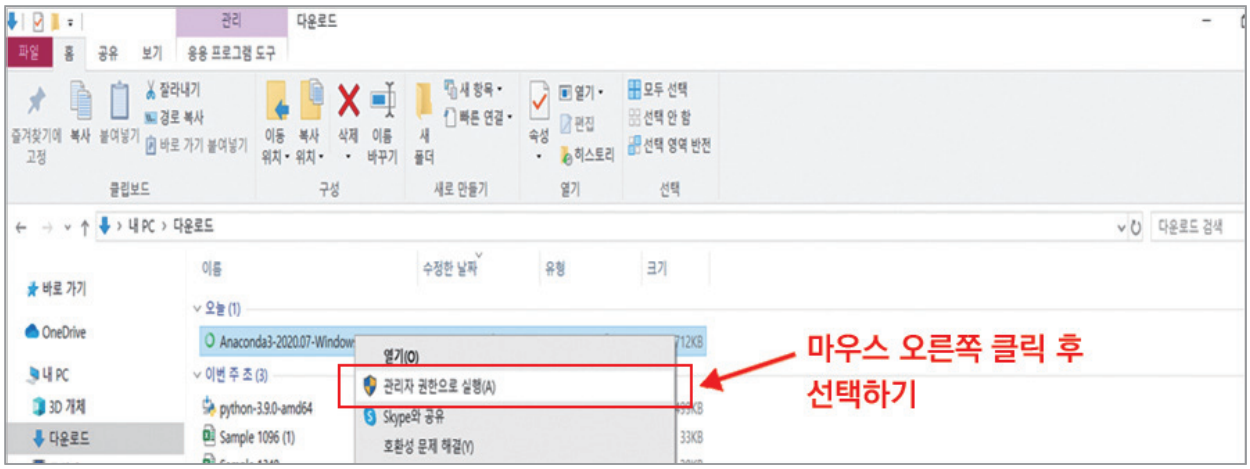
▶ Windows : 64-Bit Graphical Installer

▶ MacOS : 64-Bit Graphical Installer

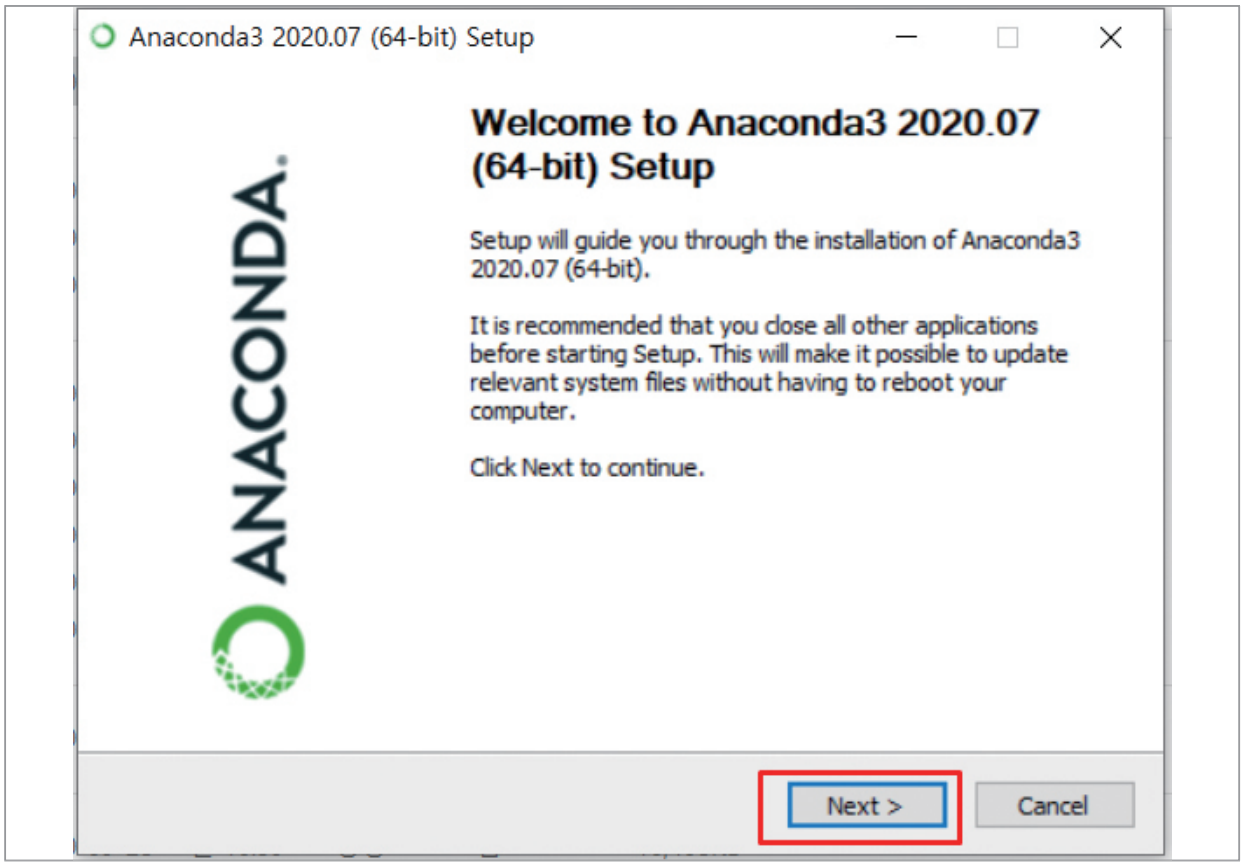


③ 다운을 받은 파일에 가서, 아나콘다 설치 파일 위에서, **마우스 오른쪽을 클릭**한 후, 방패모양의 **'관리자 권한으로 실행'** 선택

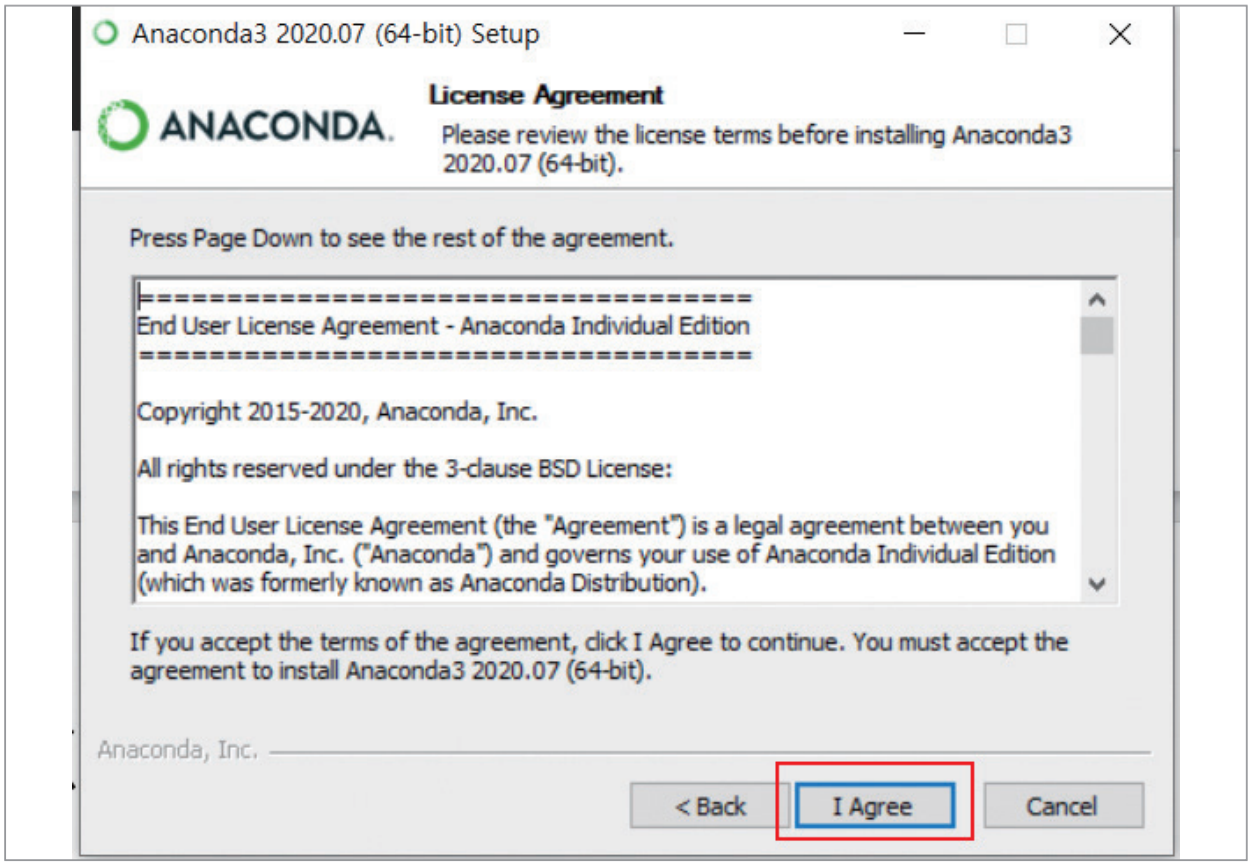
▶ **(예) '다운로드' 파일로 아나콘다를 다운 받은 경우**



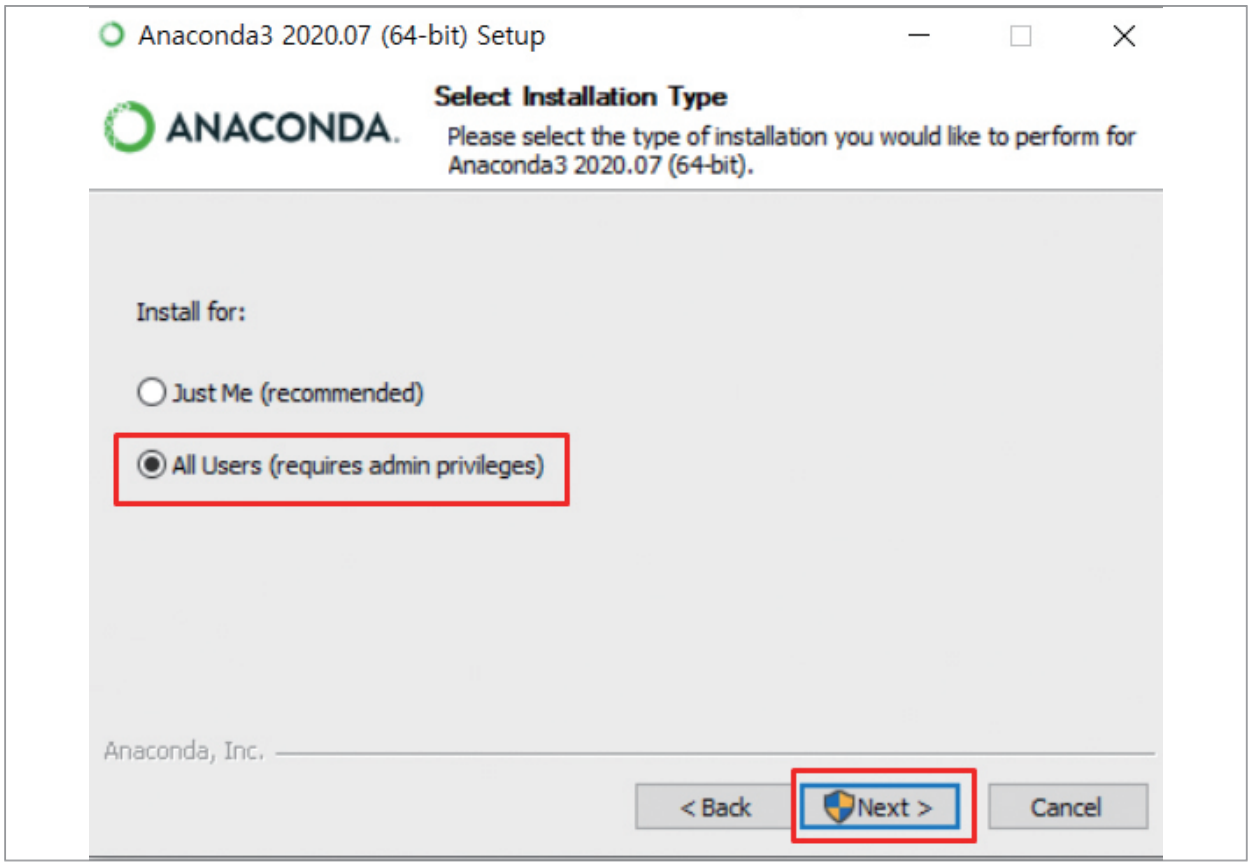
④ 파일을 실행 한 후, 'Next' 버튼을 클릭



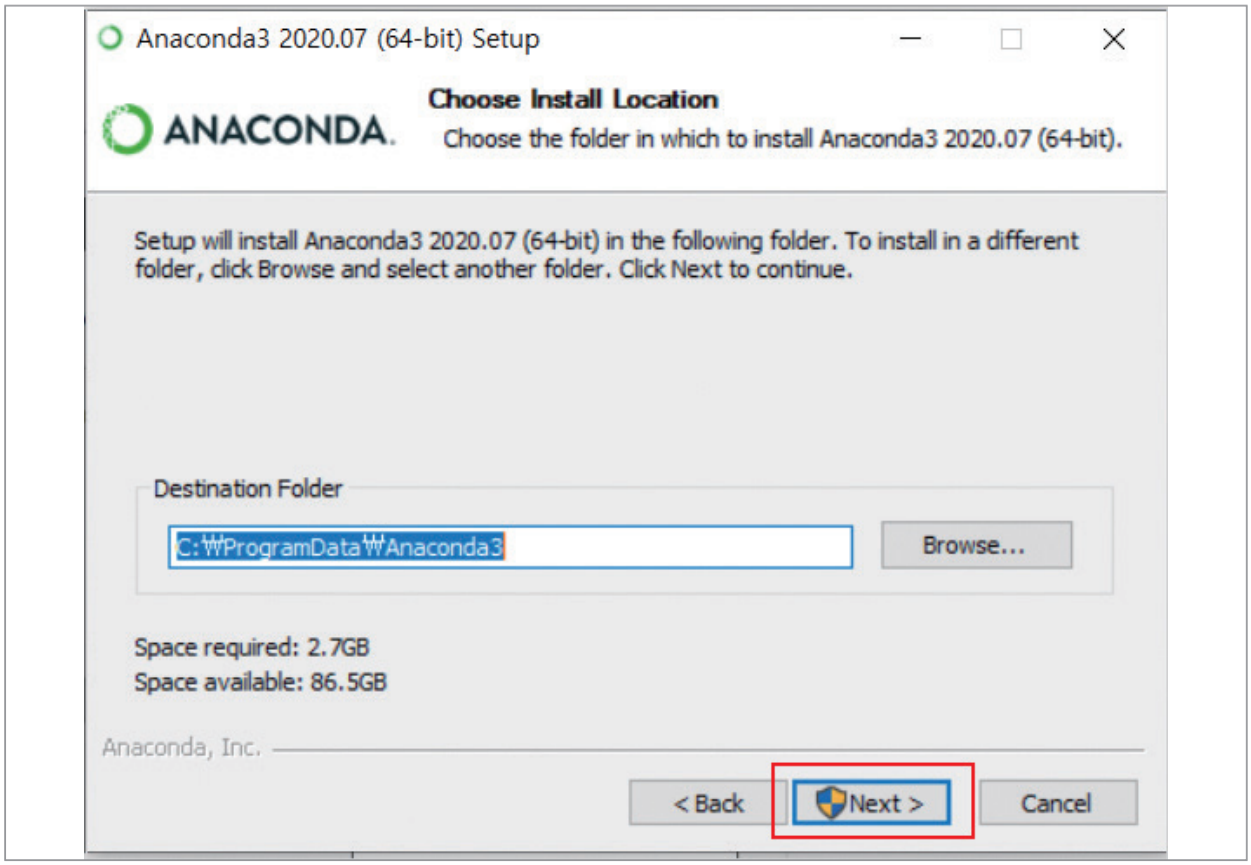
⑤ 다음 창이 나타나면 'I agree'를 선택



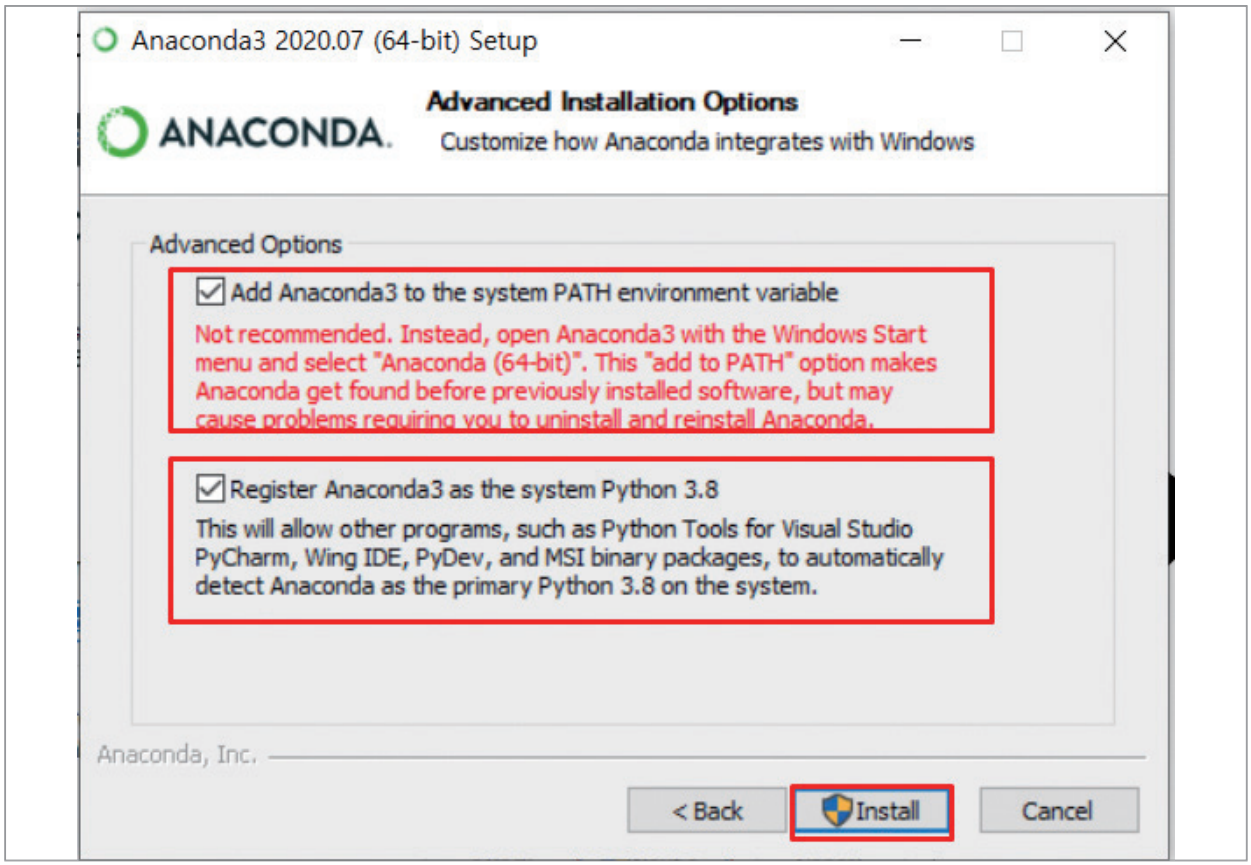
⑥ 셋팅 창이 뜨면 화면의 'All Users'를 선택 후 아래의 'Next' 클릭



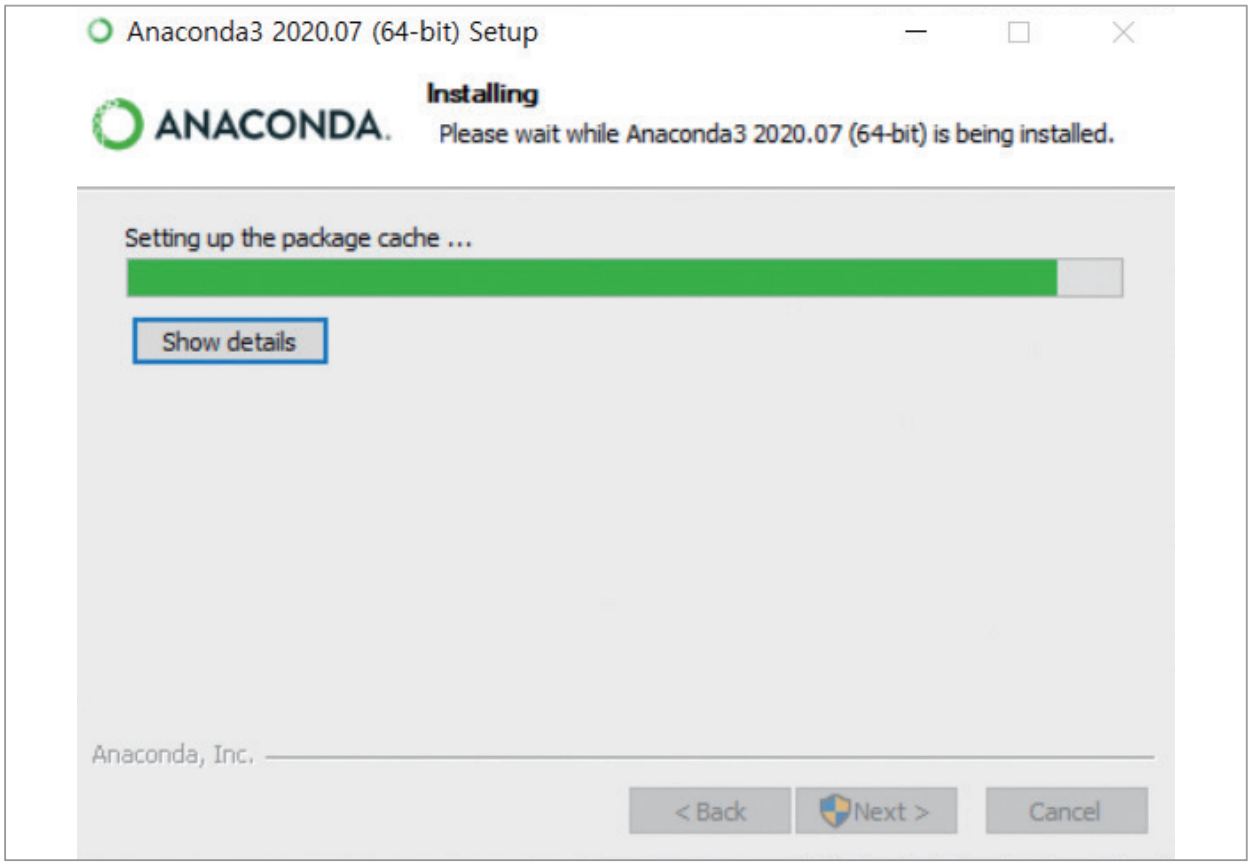
⑦ 다운로드 받을 경로를 물어보는 창이 뜨면, 아래의 'Next' 클릭



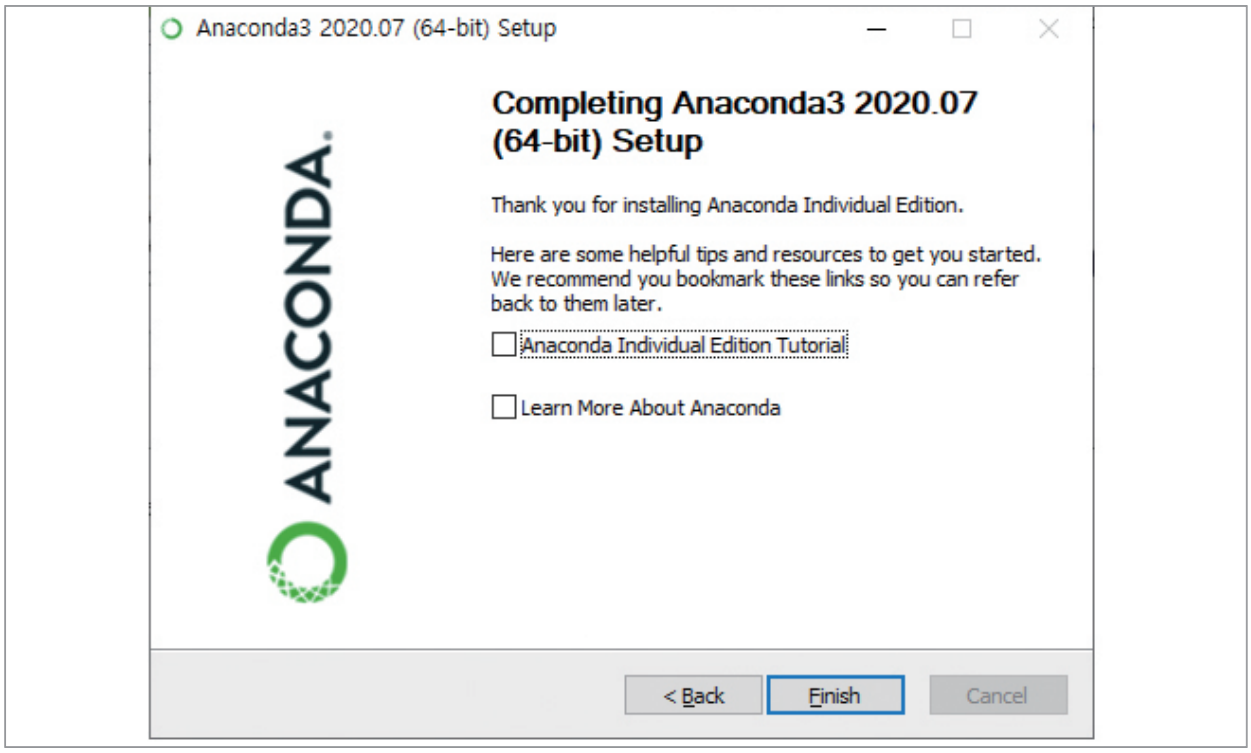
⑧ 고급 옵션 선택창이 뜨면, 아래와 같이 모두 선택 후, 'Install' 클릭



⑨ 다음과 같은 설치창이 뜨면 완료가 될 때까지 대기 (5분이상 소요)

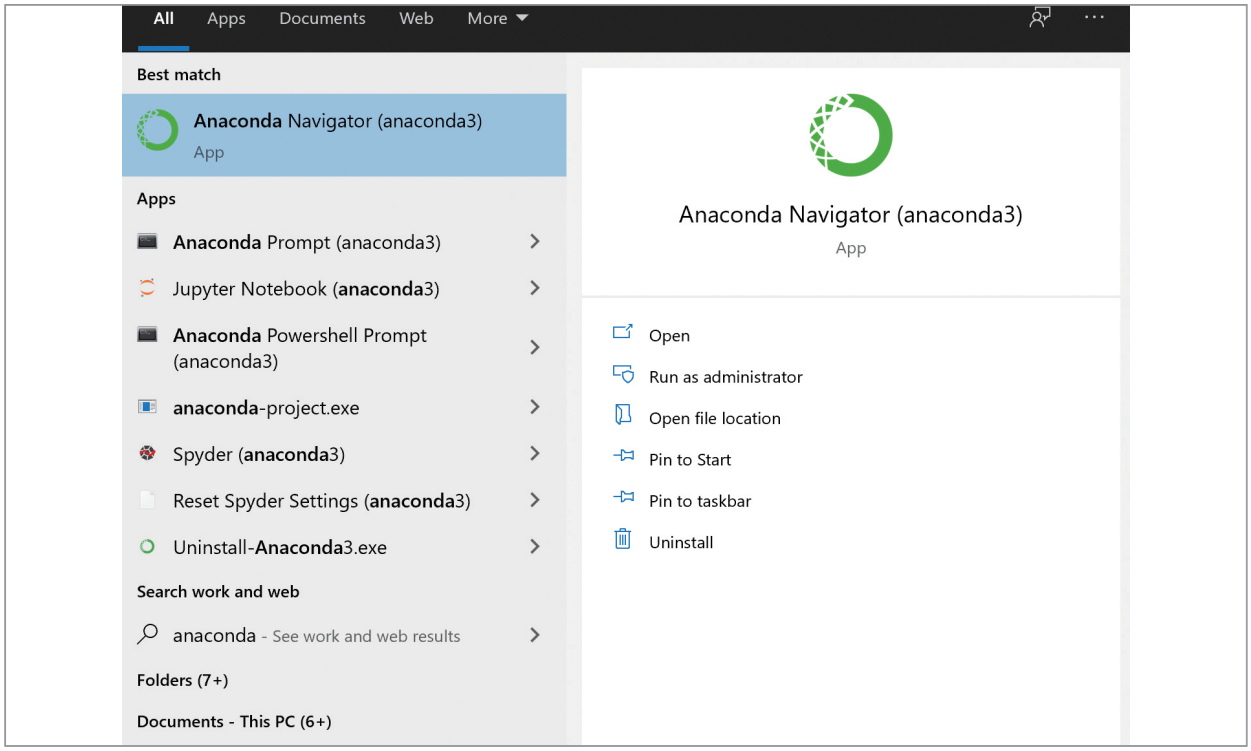


⑩ 마지막 화면에서, 모두 체크 해제한 후, 'Finish' 눌러 설치 완료



⑪ 설치 확인하기

화면상의 '홈( )'키를 눌러서 화면과 같이 anaconda prompt가 잘 깔렸는지 확인하기

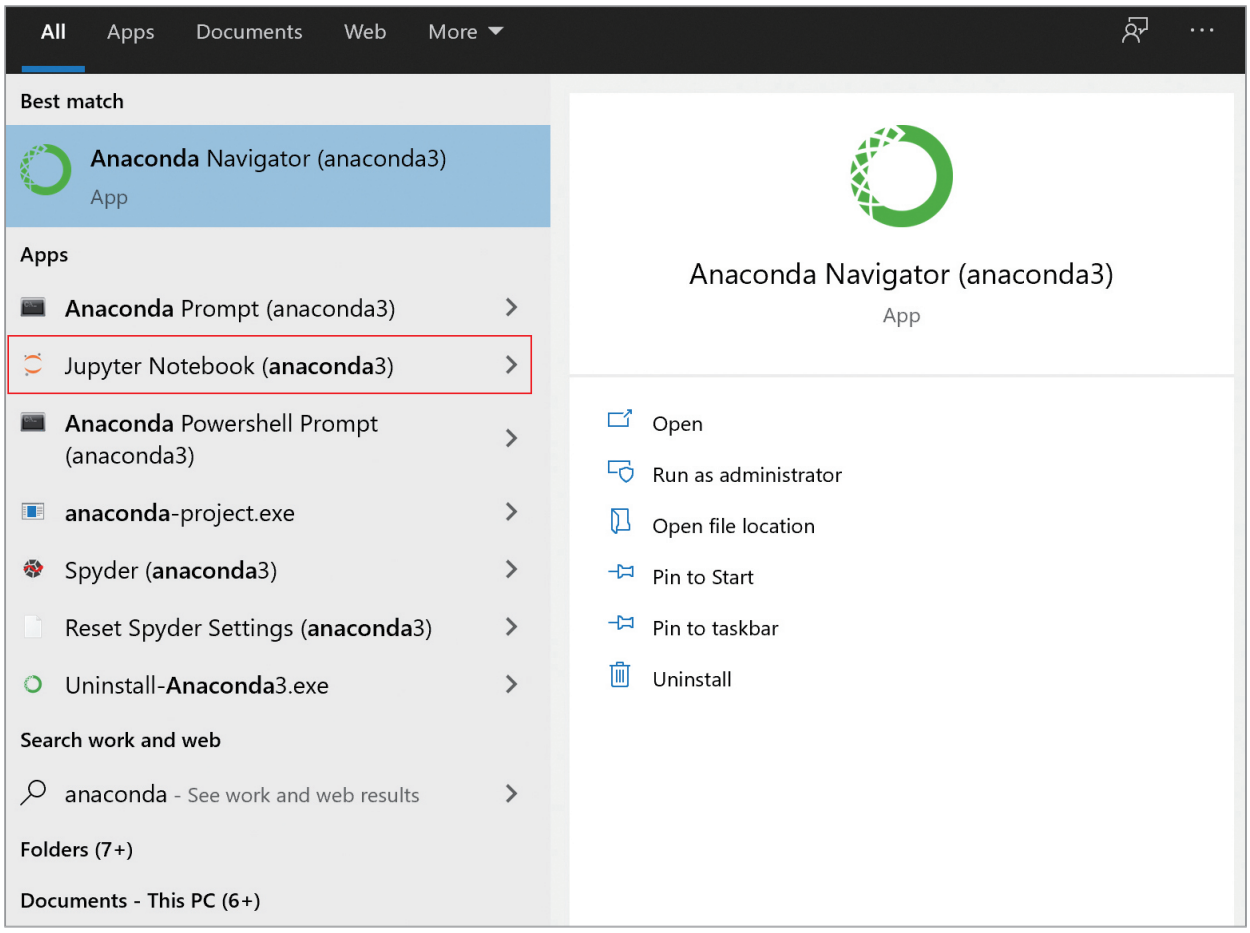


- Anaconda Navigator, Anaconda Prompt, Jupyter Notebook 등 다른 응용 프로그램들도 잘 깔려있는지 확인이 된다면, 설치 완료

3. 주피터 노트북 (Jupyter notebook) 실행

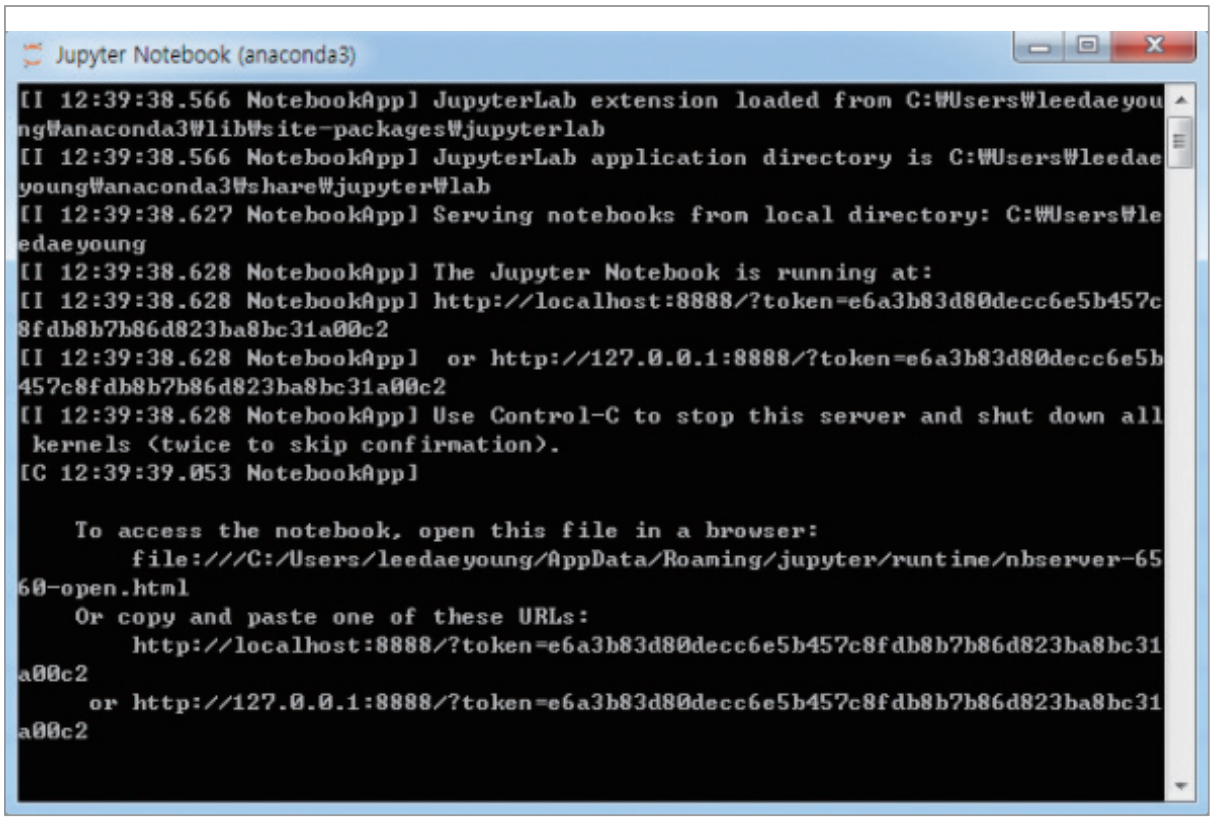
주피터 노트북이란, 실제로 코딩(분석 문장 작성)을 할 수 있는 도구이다. 쉽게 얘기하자면, 문서 도구로 마이크로소프트 사의 'Word' 파일이나, 국내에서는 '한글' 파일 등과 같은 도구라고 생각할 수 있다. 데이터 분석에 다양한 입력, 실행 도구가 있지만, 본 가이드북에서는 주피터 노트북을 활용하는 방법을 공유하기로 한다. [부록2]를 참고하여, Anaconda를 설치하였다면, 주피터 노트북(Jupyter notebook) 설치도 확인한다.

① 윈도우 키( )를 눌러서 시작화면을 연 후, anaconda를 검색.폴더 리스트 중 'Jupyter notebook(anaconda3)'를 실행한다.

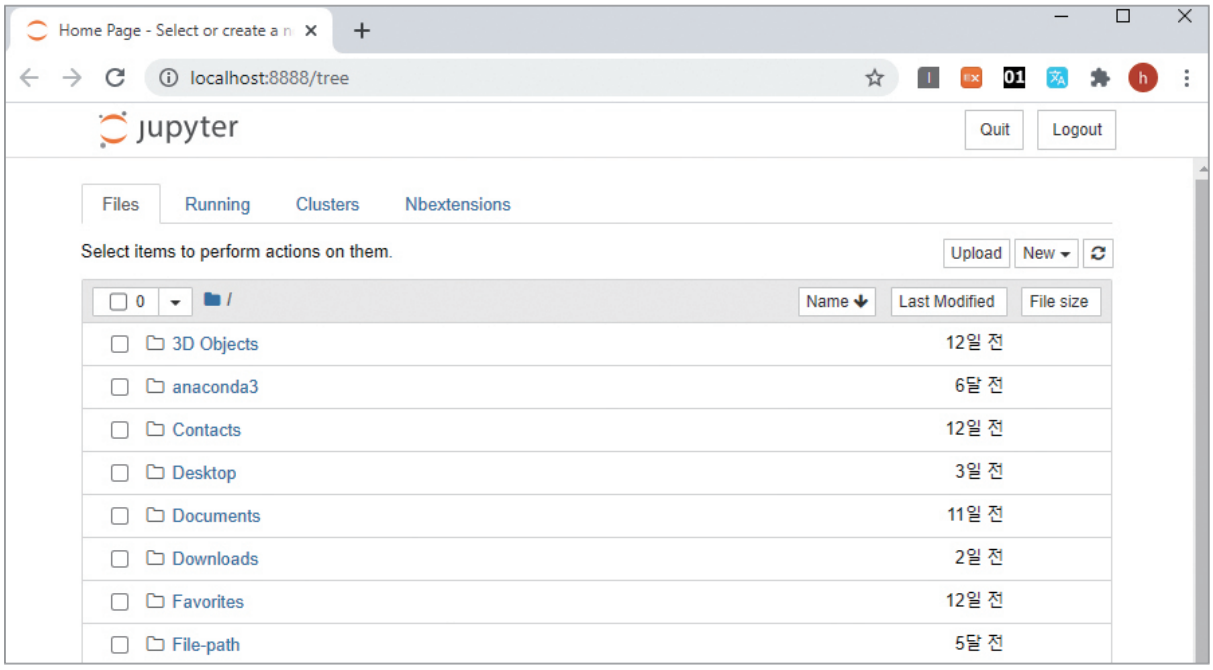


② jupyter notebook을 클릭하게 되면, 2개의 윈도우가 실행.

(1) 검은색 배경의 화면은, 주피터 노트북이 실행되는 환경에 대한 상태를 나타내주는 '상태 표시창' 같은 곳. **분석을 하는 동안 종료하면 안된다.**



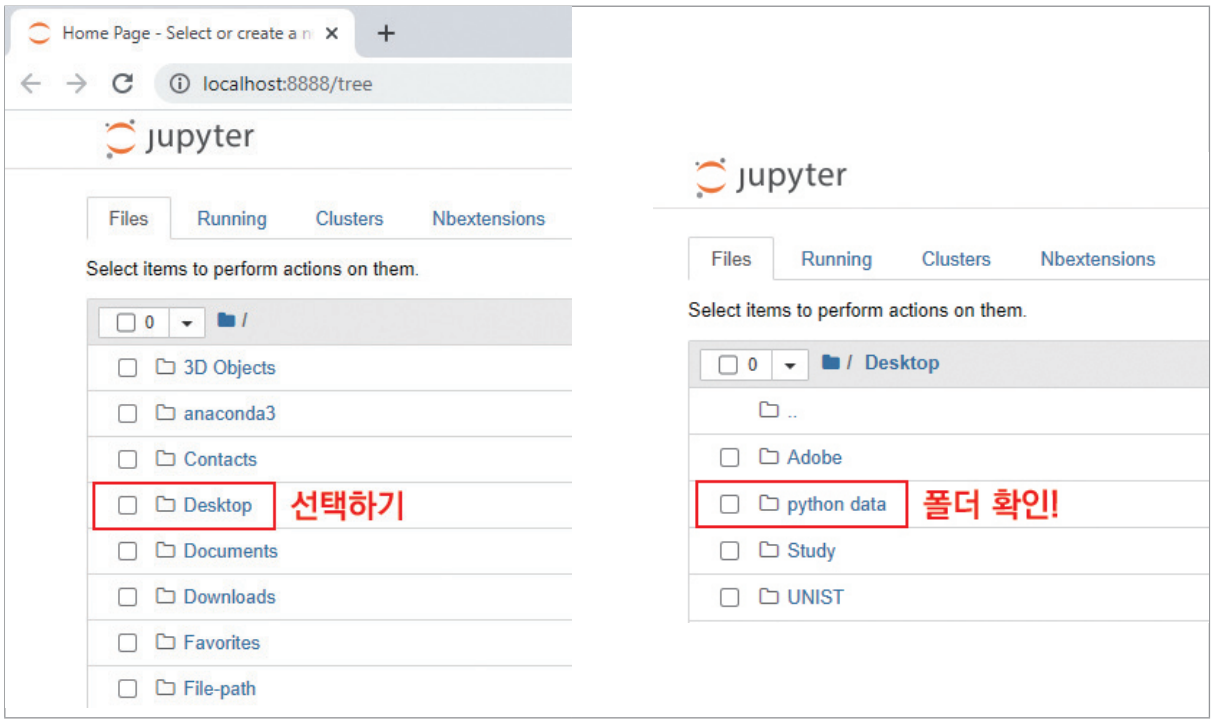
(2) 사용하는 인터넷 프로그램(크롬 / 인터넷 익스플로러)에 주피터 노트북이 열림. (다른 확장 프로그램 사용하고 싶다면, 기본 브라우저를 변경해주어야함). **이 창을 주로 사용.**



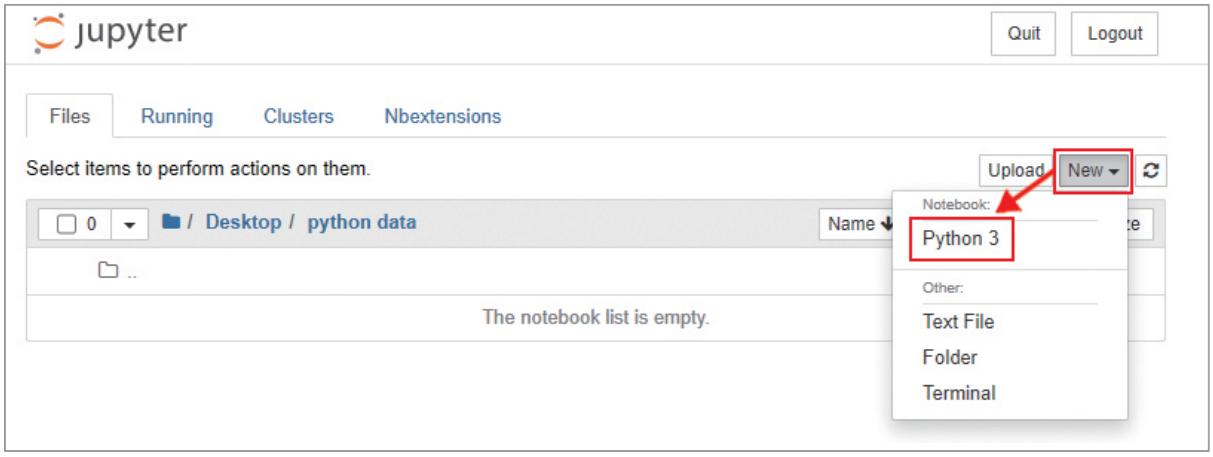
③ 데이터를 저장하고, 불러오고, 분석할 경로의 폴더를 하나 생성

▶ (예) '바탕화면'에 'python data' 폴더 생성 후 (미리 생성), 파이썬 코드 실행 후 저장하기

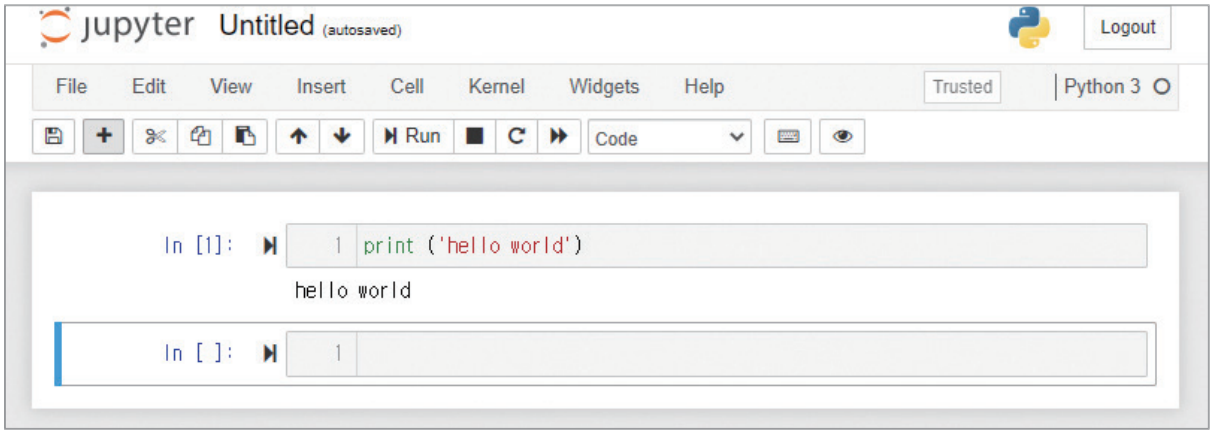
(1) 주피터에서 'python data' 폴더 확인



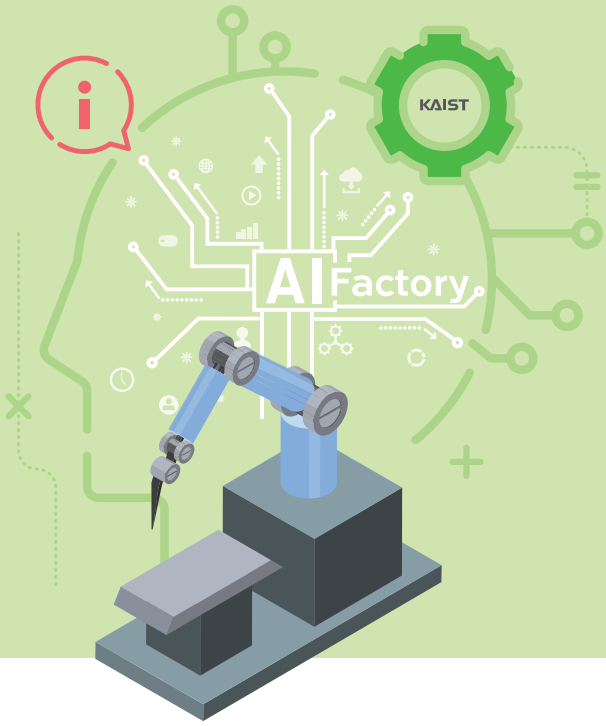
(2) python data에서 파이썬 파일 생성해보기: 오른쪽 상단의 'New'를 누른 후, 'Python 3' 선택



(3) 'hello world' 출력 확인해보기: 보이는 In[] 옆의 회색 창에 print('hello world') 입력 후, shift + Enter 키 눌러서 실행. : 자동으로 저장되며, 확장자는 '(파일이름).ipynb'로 저장



「용접기 AI 데이터셋」
분석실습 가이드북



중소벤처기업부

KOSMO
스마트제조혁신추진단

KAIST

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