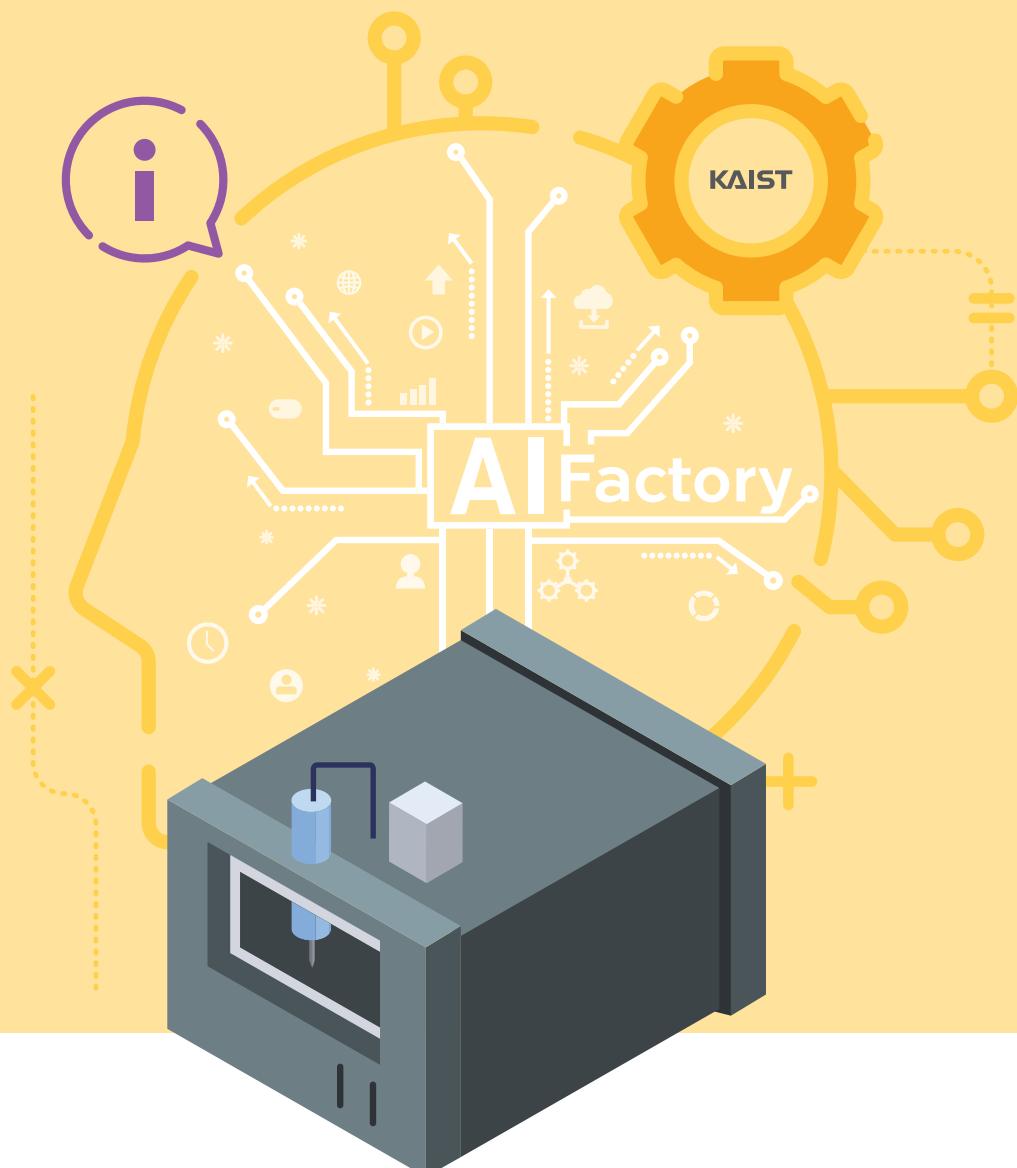
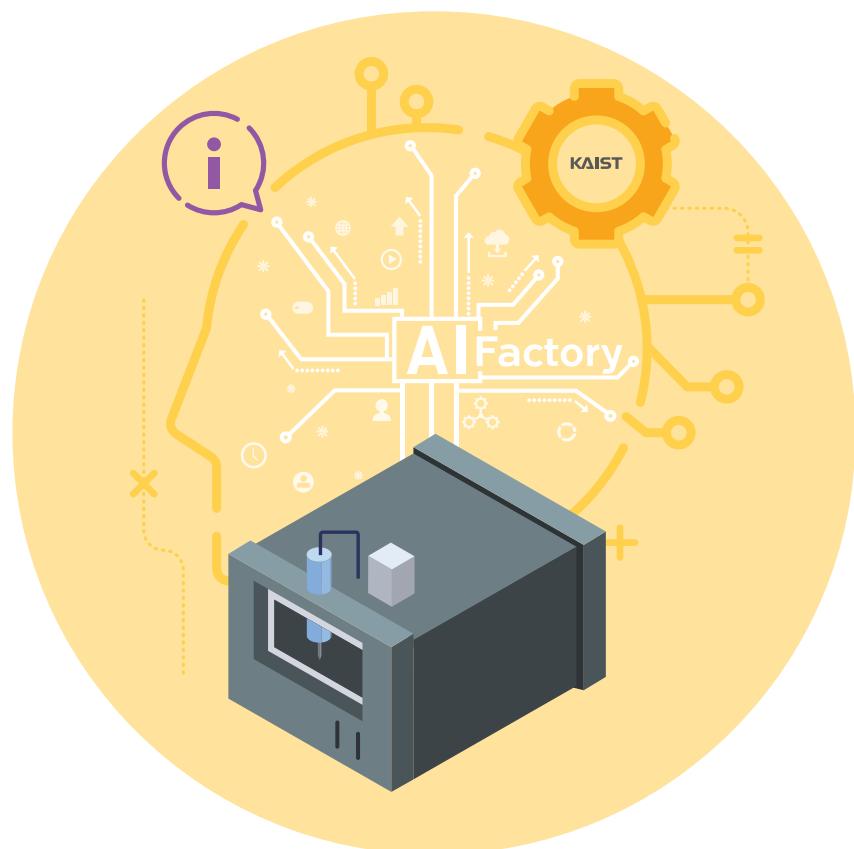


『CNC 머신 AI 데이터셋』 분석실습 가이드북

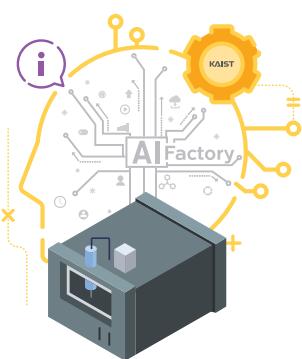


「CNC 머신 AI 데이터셋」 분석실습 가이드북



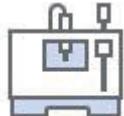
Contents

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	08
	1) 분석목표	
	2) 제조 데이터 정의 및 소개	
	3) 제조 데이터 분석 기대효과	
	4) 시사점(implication) 요약기술	
	2. 분석실습	09
	2.1 제조데이터 소개	09
	1) 데이터 수집 방법	
	2) 데이터 유형/구조	
	2.2 분석 모델 소개	17
	1) AI 분석모델	
	2.3 분석 체험	20
	1) 필요SW, 패키지 설치 방법 및 절차 가이드	
	2) 분석 단계별 프로세스 – Flow Chart	
	[단계 ①] 라이브러리 및 데이터 불러오기	
	[단계 ②] 데이터 종류 및 개수 확인	
	[단계 ③] 데이터 정제(2차 전처리)	
	[단계 ④] 데이터 라벨링	
	[단계 ⑤] 학습/검증/평가 데이터 분리	
	[단계 ⑥] AI 모델 구축	
	[단계 ⑦] AI 모델 훈련	
	[단계 ⑧] 결과 분석 및 해석	
	3. 유사 타 현장의 「CNC 머신 AI 데이터셋」 분석 적용	43
부 록	분석환경 구축을 위한 설치 가이드	44



「CNC 머신 AI 데이터셋」 분석실습 가이드북

- 필요 SW : Python
- 필요 패키지 : Tensorflow, Keras, numpy, padas, glob, sklearn, scipy
- 분석 환경 : [CPU] Intel(R) Xeon(R) Gold 6126 @ 2.60Hz, [GPU] NVIDIA GeForce RTX 2080 Ti, [Memory] 32GB
- 필요 데이터 : train.csv, experiment.csv, X_train.csv, X_test.csv, Y_train.csv, Y_test.csv
- 주 관 기관 : 한국과학기술원(KAIST)
- 수 행 기관 : 울산과학기술원(UNIST), 주식회사 이피엠솔루션즈



1 분석요약

No	구분	내용
1	분석 목적 (현장 이슈, 목적)	- CNC가공 공정의 데이터 분석을 통한 품질 문제 해결과 생산성 향상은 스마트팩토리 부분의 필수적인 요소로 자리 잡고 있다. CNC 설비와 네트워크 연결을 통해 생산데이터를 수집하고 가공조건에 따른 가공불량 예측과정을 학습할 수 있다.
2	데이터셋 형태 및 수집방법	- 분석에 사용된 변수명 : X, Y, Z, 스피드 등의 변수 총 48가지 - 수집 방법 : CNC 프로그램, 파이썬 - 확장자 : csv
3	데이터 개수 데이터셋 총량	- 데이터 개수 : 96,169개 - 데이터셋 총량 : 14.62 MB
4	분석적용 알고리즘	알고리즘 딥뉴럴네트워크 (Deep Neural Network)
		알고리즘 간략소개 - 지도학습(Supervised learning) 중 하나인, 딥뉴럴네트워크(Deep Neural Network)는 다수의 신경망을 연속적으로 쌓고, 최종적으로 각 라벨별 확률을 출력하도록 그 특징을 학습하고 업데이트를 반복하는 모델이다. 이를 통해 가공한 데이터를 학습 및 테스트할 수 있다.
5	분석결과 및 시사점	- CNC공정의 생산데이터에 따른 제품의 불량률을 예측할 수 있는 AI기법을 적용하여 CNC 가공공정의 양품과 불량품을 분류 할 수 있는 모델을 도출하였다. - CNC 머신의 변수 48가지를 사용하여 양품과 불량품을 판별하기 위해, 데이터를 가공/전처리 후 학습된 AI모델을 통해 유사한 CNC 가공 설비에 테스트 후 활용 할 수 있을 것으로 기대된다.

2 분석 실습

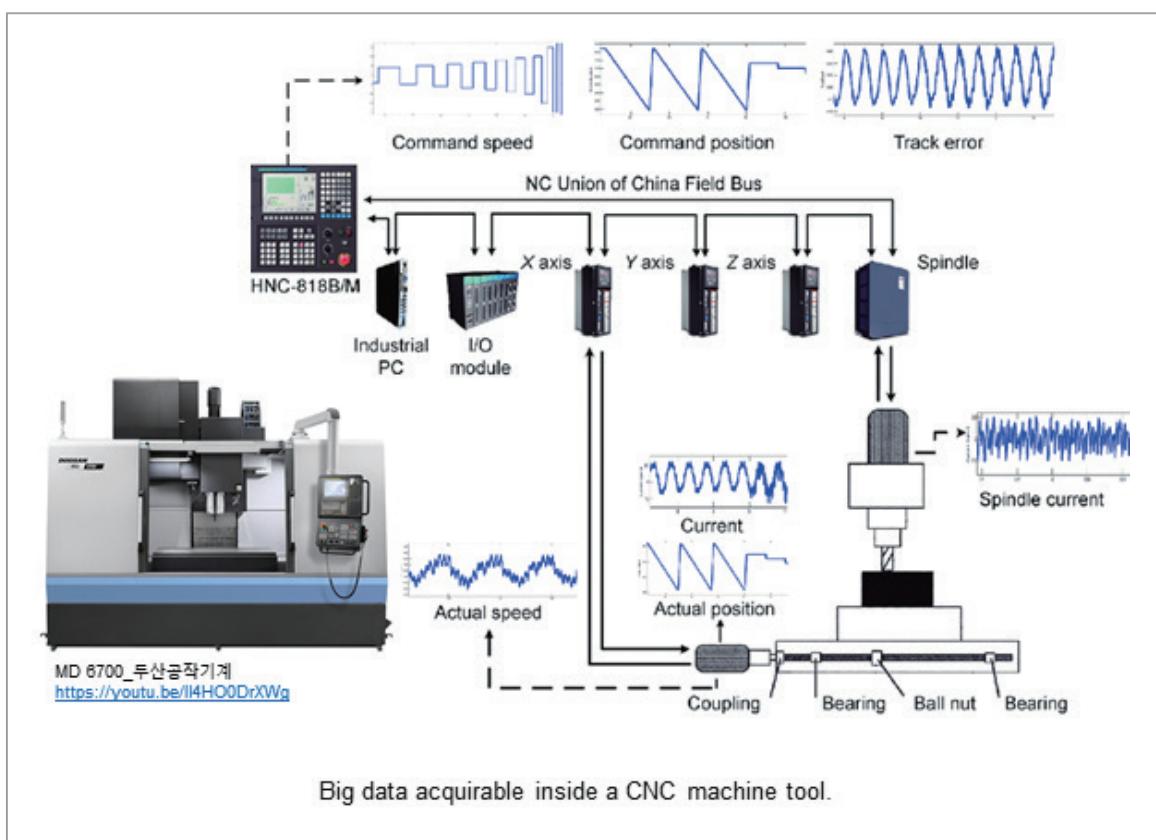
1. 분석 개요

1.1 분석 배경

1) 공정(설비) 정의 및 특징

- 예전의 범용공작기계(선반, 밀링)에서는 공구의 움직임을 수동 핸들 조작에 의해 이루 어졌지만 CNC 공작기계는 그 움직임을 CNC 프로그램에 의해 자동 제어하여 가공하는 방법이다.

(Computerized Numerical Control.) CNC 가공에서는 기계가 NC 작업을 수행하고 컴퓨터 프로그램은 개체에 맞게 사용자 정의되며 CNC 가공 언어 (G-code)를 사용하여 기계가 프로그래밍되어 이송 속도, 좌표, 위치 및 속도와 같은 모든 기능을 기본적으로 제어하게 된다. 따라서 CNC 가공을 통해 컴퓨터의 위치와 속도를 정확하게 지정 할 수 있다. 재질과 형상에 크게 구속이 없는 만능 성형가공으로 금형 제작이전 목업 제품의 제작에도 많이 쓰인다.



CNC 공정 전체 개요도

- CNC 절삭 공구

- (플랫, 볼, 불) 헤드



헤드 공구 예

홈파기, 포켓 생성, 수직 벽 깎기 등에 쓰인다. 각기 다른 형태를 띠고 있으며 이 형태를 이용하여 구조를 생성한다.

- 드릴



드릴 예

구멍을 뚫는 작업에 쓰이는 가장 빠르고 경제적인 툴이다. 표준 규격 외의 지름을 가진 구멍은 맞춤으로 제작된 드릴을 이용하거나 플랫 헤드 등으로 추가적인 조형을 하기도 한다.

- 슬롯 커터



슬롯 커터 예

공작물의 슬롯을 제작하는 데 쓰이는 툴이다. 일반적으로 슬롯 커터는 절삭 날보다 샤프트가 좁아 T자 깎기가 가능하다.

- 텁



금속 공정에 사용하는 텁 예

나사산을 형성하는 데 쓰인다. 정교한 피팅을 위하여 정밀하고 일정한 안정성이 필요하다.

- 페이스 밀링 커터



페이스 밀링 커터 예

넓은 면을 고르게 깎는 데에 능한 도구이다. 깎는 면적이 넓어 전처리 과정에 사용되기도 한다.

2) 이슈사항(Pain point)

- 공정(설비)상의 문제현황

- CNC 절삭가공에 있어서 절삭공구가 마찰에 의해 마모되거나 절삭력의 순간 변경으로 인해 파손으로 마모 한계치에 도달하게 되면 가공 정밀도가 급격하게 떨어지게 된다. 또한 절삭력 증가로 인해 공작기계의 수명에도 영향을 미치므로 생산제품의 품질을 균일하게 하고 재현성을 향상시키기 위해 공구수명에 따른 가공볼륨 예측은 반드시 필요한 사항이다.

- 문제해결 장애요인

- CNC가공품은 공정상 제작중에 생기는 문제점을 실시간 인지하기 힘들며 제품성형이 끝난 뒤에야 비로소 제품의 불량여부를 알 수 있다.

• 극복방안

- CNC 가공머신에서 제공하는 설정값 대비 실제 절삭공구가 움직이는 값, 전류, 전압 데이터등과 스픬들 토크 등의 값을 수집하고 절삭공구수명에 따른 가공불량 예측을 하고자 노력하였다.

1.2 분석 목표

1) 분석목표

- CNC 설비와 네트워크 연결을 통해 실제 가공생산데이터를 수집하여 공구수명에 따른 가공불량 예측을 다양한 기계학습 알고리즘을 활용하여 해결하고자 한다.

2) 데이터 정의 및 소개

- CNC 설비와 네트워크 연결을 통해 수집한 생산데이터를 1, 2차 전처리 과정을 거친 가공 데이터를 사용하고, 공정의 정지 유무를 양품과 불량품 2가지의 '라벨 데이터'로 정의한다.

데이터 변수 종류	개수
샘플 관련 변수	6개
기계의 X축 관련 변수	11개
기계의 Y축 관련 변수	11개
기계의 Z축 관련 변수	11개
기계의 스플들 관련 변수	12개
기타 변수	4개

데이터 변수의 종류 및 개수

3) 데이터 분석 기대효과

- 유사한 CNC 가공 설비에 테스트 후 활용할 수 있을 것으로 판단된다.

4) 시사점(implication) 요약기술

- 자동차 부품 J사 공장에는 CNC 장비에서 작업 중에 어떤 요인으로 인한 불량이 발생되는지에 대한 내용을 파악하기가 힘들었다. 항상 CNC 가공이 완결된 뒤에 품질검사를 시행하여 양품과 불량을 판단하였으며 불량이 발생하였을 시 결과물을 본 후에 조치를 취할 수 있어 많은 손실이 생겼었다. 따라서 CNC 설비의 데이터를 기준으로 가공불량 예측 모델을 생성하고, 결과 기반 재현 검증, 그리고 양품과 불량의 예측 모델을 구성하고자 한다.

이에 CNC 가공의 검사 판정 데이터 및 생산 조건 데이터를 수집하여 가공/전처리 및

AI 모델 개발과 제조공정의 적용 및 검증을 통한 수행으로 열악한 중소기업에 빅데이터 및 AI를 적용하여 실질적인 품질 및 비용절감 개선에 기여했다는 점에서 시사하는 바가 크다고 판단된다.

2. 분석실습

2.1 제조데이터 설명

1) 데이터 수집 방법

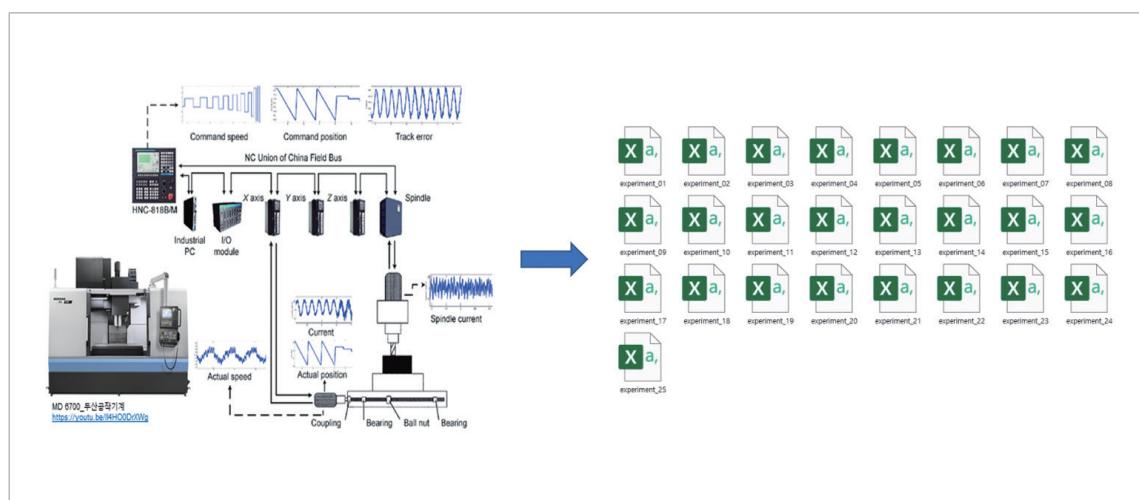
- 제조 분야 : 자동차 부품
- 제조 공정명 : 자동차 부품 CNC 가공 공정
- 수집 장비 : 공장 내 CNC 가공 장비 Data, 조건 및 품질 Data
- 수집 기간 : 10/19~10/23 (5일)
- 수집 주기 : 1 sec

2) 데이터 유형/구조

데이터셋 구조, 칼럼 수, 데이터 개수, AI 데이터셋 주요 변수 정의

• Original Data

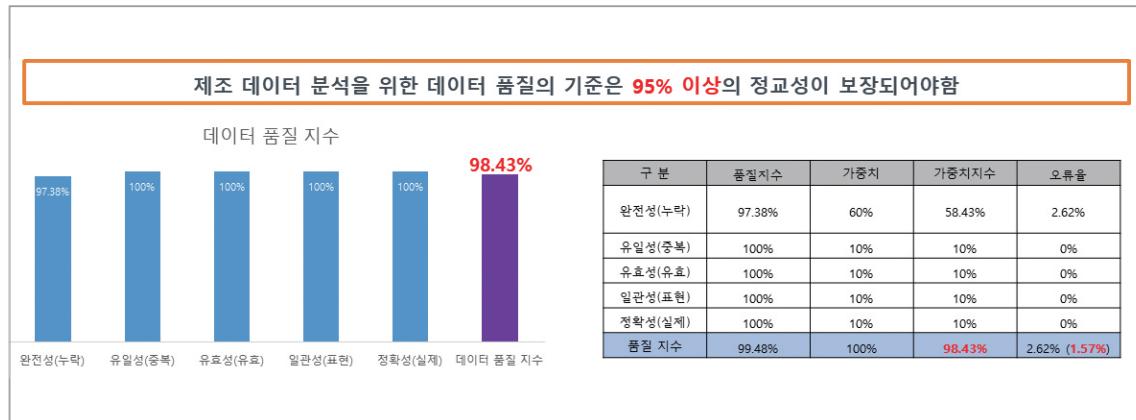
- 실제 공정에서 발생한 데이터를 받아서 그대로 저장한다. 실제 공정에서 발생하는 데이터들은 값이 의미 없거나 누락 및 오타가 발생하여 데이터 품질이 떨어진다. 이에 품질이 낮은 데이터를 이용하면 좋은 결과를 얻을 수 없기 때문에 데이터 품질 전처리는 데이터 분석에서 가장 중요한 단계이다.



CNC 공정에서 데이터 취득 및 저장

• 1차 가공 데이터

- Original D 지수를 파악하고, 데이터 전처리를 통해 품질 지수를 향상시킨다.



1차 데이터 가공을 위한 품질 지수

• 데이터 품질지수

- 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 반영이 되어야 한다.

<예시>

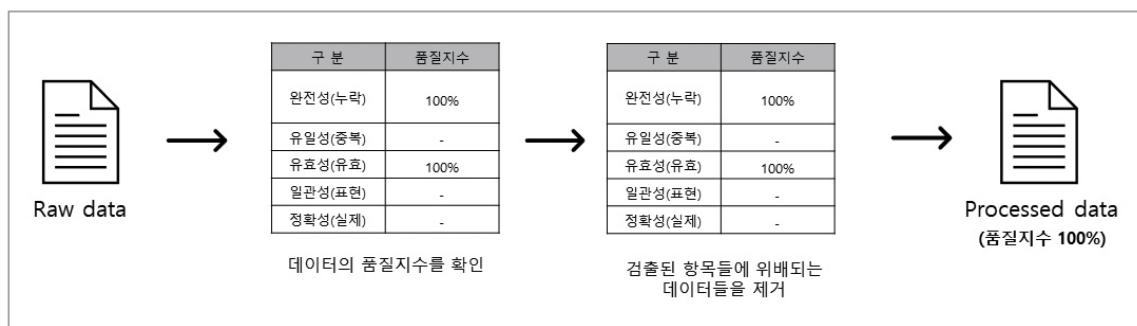
idx	Machine_Name	Item_N	working time	Press time(ms)	Pressure	Pressure	Pressure
1843	Press-01	ED5260	2020-05-07 0:00		275.2	273.2	548.4
1844	Press-01	ED5260	2020-05-07 0:00		275.5	273.1	548.6
1885	Press-01	ED5260	2020-05-07 0:00		274.8	276	550.8
1886	Press-01	ED5260	2020-05-07 0:00		275.6	273.2	548.8
1619	Press-01	ED5260	2020-05-15 0:00		274.8	267	541.8
561	Press-01	ED5260	2020-05-20 0:00		274.9	269	543.9
755	Press-01	ED5260	2020-05-26 0:00		274.9	269	543.9
3879	Press-01	ED5260	2020-05-28 0:00		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
105	Press-01	ED5260	8159-02-28 0:00		549	274.5	269
107	Press-01	ED5260	8277-03-19 0:00		550.2	275.1	267
108	Press-01	ED5260	8336-03-29 0:00		550.2	275.1	267
111	Press-01	ED5260	8513-04-27 0:00		550.2	275.1	267
136	Press-01	ED5260	9988-12-25 0:00		549.8	274.9	267
2	Press-01	ED5260	1900-01-00 0:00		550	275	267
138	Press-01	ED5260	1900-01-00 0:00		549.2	274.6	267
140	Press-01	ED5260	1900-01-00 0:00		550.2	275.1	267
142	Press-01	ED5260	1900-01-00 0:00		551	275.5	267

-> working time에 유효하지 않은 날짜형식이 들어있으므로 유효성을 위배한다.

품질 지수 측정 예시



1차 데이터 가공 절차

- 데이터 파일은 전체 실험 샘플의 결과 데이터 1개와 각 실험 샘플의 상세 데이터가 열(row) 별로 저장되어있는 25개의 파일, 총 26개로 구성되어 있으며 모두 csv파일 형태이다. 실데이터 스크린샷은 다음과 같다.

No	material	feedrate	clamp_pre	tool_cond	machining	passed_visual_inspection
1	aluminum	6	4	unworn	yes	yes
2	aluminum	20	4	unworn	yes	yes
3	aluminum	6	3	unworn	yes	yes
4	aluminum	6	2.5	unworn	no	
5	aluminum	20	3	unworn	no	
6	aluminum	6	4	worn	yes	no
7	aluminum	20	4	worn	no	
8	aluminum	20	4	worn	yes	no
9	aluminum	15	4	worn	yes	no
10	aluminum	12	4	worn	yes	no
11	aluminum	3	4	unworn	yes	yes
12	aluminum	3	3	unworn	yes	yes
13	aluminum	3	4	worn	yes	yes
14	aluminum	3	3	worn	yes	yes
15	aluminum	6	3	worn	yes	yes
16	aluminum	20	3	worn	no	
17	aluminum	3	2.5	unworn	yes	yes
18	aluminum	3	2.5	worn	yes	yes
19	aluminum	15	4	worn	yes	no
20	aluminum	12	4	unworn	no	
21	aluminum	3	4	unworn	yes	no
22	aluminum	20	3	worn	yes	yes
23	aluminum	3	4	worn	no	
24	aluminum	3	3	unworn	yes	yes
25	aluminum	6	2.5	worn	yes	yes

train.csv의 실데이터 스크린샷

X_ActualP	X_ActualV	X_ActualA	X_SetPos	X_SetVel	X_SetAcc	X_Current	X_DCBus	X_Output	X_Output	X_Output	X_ActualP	Y_ActualP	Y_ActualV	Y_SetPos	Y_SetVel	Y_SetAcc	Y_Current	Y_DCBus	Y_Output	Y_Output	Z_ActualP	Z_ActualV	Z_SetPos			
202	-4	-4	203	-4	-0.18	0.0027	328	2.77	-1.43E-06	162	3.975	-2.25	162	4	4	0.539	0.0167	328	1.84	6.43E-07	123	4	4	12		
202	-6.8	-346	202	-9.6	-354	-10.0	0.186	328	23.3	0.00448	162	-15.8	-746	161	-20.6	-643	-14.5	0.281	325	37.8	0.0126	123	-16.3	-708	12	
200	-13.8	-2.25	200	-13.9	3.999905	-8.59	0.14	328	20.6	0.00533	158	-28.5	-4	158	-28.3	3.999905	-7.79	0.139	327	49.4	0.00943	119	-29.7	41.5	11	
198	-14	-2.5	198	-13.9	3.999905	-6.11	0.13	327	30.3	0.00489	155	-28.6	-58.5	155	-28.3	3.999905	-8.13	0.156	325	47.6	0.0105	116	-29.7	-2.25	11	
197	-13.9	-14.8	196	-13.9	4.000095	-5.7	0.114	328	30.5	0.00425	152	-28	142	152	-28.3	3.999905	-13.8	0.202	326	47.1	0.0135	113	-29.6	22.8	11	
195	-13.6	85.2	195	-13.9	4.000191	-5.85	0.128	328	30.9	0.00474	149	-28.4	-8.5	145	-28.3	3.999905	-8.56	0.179	326	49.6	0.0113	109	-29.7	-2.25	10	
193	-13.9	-2.25	193	-13.9	4.000095	-5.84	0.0955	328	29.3	0.00345	145	-28.3	85.2	145	-28.3	3.999905	-8.56	0.179	326	50	0.0124	106	-29.5	66.5	10	
191	-13.8	54	191	-13.9	3.999809	-7.31	0.117	327	29.1	0.00441	142	-28.3	54	142	-28.3	3.999905	-8	0.185	325	50.8	0.0128	102.6	-29.7	22.8	102	
189	-13.9	29	189	-13.9	4	-7.98	0.113	328	29.4	0.00417	139	-28.3	41.5	139	-28.3	4	-9.96	0.154	327	49.7	0.0102	99.1	-29.6	22.8	98.1	
188	-13.9	16.5	187	-13.9	3.999905	-7.44	0.138	328	29.8	0.00501	136	-28.7	139	-28.3	4	-6.29	0.119	325	48.1	0.00784	95.8	-29.7	-33.5	95.4		
186	-14.1	-52.3	186	-13.9	3.999905	-4.52	0.114	327	31.3	0.00483	133	-28.1	117	133	-28.3	4	-8.62	0.186	325	48.6	0.0126	92.5	-29.7	-21	92	
184	-13.9	66.5	184	-13.9	4.000191	-5.53	0.128	328	31.4	0.00485	129	-28.3	16.5	129	-28.3	3.999905	-16.9	0.181	326	50.6	0.0124	89	-29.7	4	88	
183	-13.9	-33.0	182	-13.9	4.000095	-7.57	0.122	327	30.4	0.00485	126	-28.4	22.6	126	-28.3	3.999905	-17.4	0.174	325	50.4	0.0122	87	-29.6	10.25	85	
180	-13.8	110	180	-13.9	4.000095	-8.41	0.133	327	28.5	0.005	123	-28.3	34	123	-28.3	3.999905	-8.39	0.177	327	49.4	0.0112	82.4	-29.7	4.32	82	
179	-13.8	47.8	178	-13.9	3.999905	-8.81	0.131	328	30.3	0.00478	120	-28.2	135	119	-28.3	3.999905	-11.3	0.165	327	51.2	0.0111	78.9	-29.6	54	78	
177	-14	-21	177	-13.9	4.000191	-5.78	0.127	327	29.9	0.00475	116	-28.3	41.5	116	-28.3	3.999905	-10.8	0.155	325	49.3	0.0114	75.6	-29.5	-2.25	75	
175	-13.8	22.8	175	-13.9	4.000095	-7.39	0.127	327	30.2	0.00468	113	-28.2	47.8	113	-28.3	3.999905	-12.8	0.172	325	43.2	0.0114	72.3	-29.5	35.3	7	
173	-13.9	10.25	173	-13.9	3.999905	-6.08	0.125	328	31	0.00472	114	-28.3	16.5	114	-28.3	4	-12.6	0.196	326	47.3	0.0134	68.7	-29.8	-46	68	
171	-13.8	10.25	171	-13.9	3.999905	-6.89	0.131	327	31.8	0.00491	107	-28.3	41.5	107	-28.3	3.999905	-11.4	0.185	325	50.2	0.0126	65.5	-29.7	-21	65	
170	-13.8	54	170	-13.9	4.000095	-6.42	0.111	328	29.8	0.00441	103.6	-28.4	4	103.3	-28.3	3.999905	-11	0.161	326	46.6	0.0109	52.1	-29.7	10.25	61	
168	-13.9	-39.8	168	-13.9	4.000191	-7.26	0.112	328	29.8	0.00416	100.2	-28.1	91.5	99.9	-28.3	3.999905	-4	-12.2	0.203	326	49.4	0.0138	58.6	-29.6	47.8	58
166	-14	22.8	166	-13.9	3.999905	-4.17	0.0986	328	28.4	0.00361	97.1	-28.1	123	96.8	-28.3	3.999905	-12.3	0.176	325	48.6	0.0117	55.4	-29.7	10.25	55	
164	-14.1	-89.7	164	-13.9	3.999809	-5.34	0.107	328	30.5	0.00394	93.9	-28.3	-14.8	93.6	-28.3	3.999905	-10.5	0.181	326	50	0.0123	52	-29.7	10.25	51	
162	-14	-33.5	162	-13.9	4	-5.22	0.111	328	30.1	0.00409	90.5	-28.3	60.3	90	-28.3	4.000099	-10.2	0.19	326	47.6	0.013	48.5	-29.7	-8.5	48	
161	-13.9	-8.5	161	-13.9	3.999905	-6.7	0.13	328	30.3	0.00414	87.4	-28.2	22.8	87.1	-28.3	3.999905	-14.6	0.202	326	43.8	0.0139	45.3	-29.8	10.25	4	
159	-13.7	28.7	159	-13.9	3.999905	-7.63	0.139	328	30.3	0.00518	84.1	-28.5	-64.7	83.9	-28.3	3.999905	-37.1	0.174	326	46.9	0.0119	41.9	-29.6	16.5	41	
157	-13.8	-21	157	-13.9	4.000095	-6.66	0.134	328	30.7	0.00499	80.8	-28	23	80.5	-28.3	3.999905	-37.1	0.208	326	50.2	0.0144	38.4	-29.8	-58.5	36	
155	-13.2	192	155	-13.9	-10.3	362	-1.97	0.0687	328	23.5	0.00212	77.7	-26.9	423	77.3	-21.9	651	-3.21	0.0867	326	40.6	0.00515	35.2	-28.8	254	3
155	-14	35.3	155	-14	4	1.17	0.032	329	1.81	1.32E-05	77	4.05	10.25	77	4	4	1.18	0.0235	328	0.991	0	34.5	4.05	4	34	
155	3.925	-33.5	155	4	4	0.45	0.0276	329	4.82	-1.44E-05	77	3.975	-8.5	77	4	4	0.7	0.0235	328	3.32	-1.54E-06	34.5	3.975	-8.5	34	
155	4.075	47.8	155	4	4	0.651	0.0402	328	2	1.02E-05	77	4.025	29	77	4	4	-1.54	0.0205	327	4.47	2.77E-06	34.5	4.05	29	34	
155	4.075	41.5	155	4	4	0.878	0.0377	327	1.56	7.50E-06	77	4.025	10.25	77	4	-0.1	0.0184	325	1.82	1.77E-06	34.2	-1.88	-21	34		

experiment_*.csv의 실데이터 스크린샷

• 데이터 속성정의 표

- 실데이터의 설명 및 속성값은 다음과 같다.

Input Data	Description
No	작업 번호
material	작업 소재
feed rate	Tool 이동 속도 (mm/s)
clamp pressure	소재 Clamping 압력 (bar)
Output Data	Description
tool condition	일정시간 사용한 Tool, 새로운 Tool
machine_completed	가공 완료 여부
passed_visual_inspection	육안 검사 결과
Data Set	Number of row
experiment_01.csv	1056
experiment_02.csv	1669
experiment_03.csv	1522
experiment_04.csv	533
experiment_05.csv	463
experiment_06.csv	1297
experiment_07.csv	566
experiment_08.csv	606
experiment_09.csv	741
experiment_10.csv	1302
experiment_11.csv	2315
experiment_12.csv	2277
experiment_13.csv	2234
experiment_14.csv	2333

Data Set	Number of row
experiment_15.csv	1382
experiment_16.csv	603
experiment_17.csv	2151
experiment_18.csv	2254
experiment_19.csv	566
experiment_20.csv	606
experiment_21.csv	1669
experiment_22.csv	566
experiment_23.csv	463
experiment_24.csv	2333
experiment_25.csv	566

Attributes name	Description	Type	Unit
X_ActualPosition	부품의 실제 x 위치	float64	mm
X_ActualVelocity	부품의 실제 x 속도	float64	mm/s
X_ActualAcceleration	부품의 실제 x 가속도	float64	mm/s/s
X_SetPosition	부품의 참조 x 위치	float64	mm
X_SetVelocity	부품의 참조 x 속도	float64	mm/s
X_SetAcceleration	부품의 참조 x 가속도	float64	mm/s/s
X_CurrentFeedback	x 전류	float64	A
X_DCBusVoltage	x 전압	float64	V
X_OutputCurrent	x 아웃풋 전류	float64	A
X_OutputVoltage	x 아웃풋 전압	float64	V
X_OutputPower	x 아웃풋 전원	float64	kw
Y_ActualPosition	부품의 실제 y 위치	float64	mm
Y_ActualVelocity	부품의 실제 y 속도	float64	mm/s
Y_ActualAcceleration	부품의 실제 y 가속도	float64	mm/s/s
Y_SetPosition	부품의 참조 y 위치	float64	mm
Y_SetVelocity	부품의 참조 y 속도	float64	mm/s
Y_SetAcceleration	부품의 참조 y 가속도	float64	mm/s/s
Y_CurrentFeedback	y 전류	float64	A
Y_DCBusVoltage	y 전압	float64	V
Y_OutputCurrent	y 아웃풋 전류	float64	A
Y_OutputVoltage	y 아웃풋 전압	float64	V
Y_OutputPower	y 아웃풋 전원	float64	kw
Z_ActualPosition	부품의 실제 z 위치	float64	mm
Z_ActualVelocity	부품의 실제 z 속도	float64	mm/s
Z_ActualAcceleration	부품의 실제 z 가속도	float64	mm/s/s
Z_SetPosition	부품의 참조 z 위치	float64	mm
Z_SetVelocity	부품의 참조 z 속도	float64	mm/s

Attributes name	Description	Type	Unit
Z_SetAcceleration	부품의 참조 z 가속도	float64	mm/s/s
Z_CurrentFeedback	z 전류	float64	A
Z_DCBusVoltage	z 전압	float64	V
Z_OutputCurrent	z 아웃풋 전류	float64	A
Z_OutputVoltage	z 아웃풋 전압	float64	V
Z_OutputPower	z 아웃풋 전원	float64	kw
S_ActualPosition	스핀들의 실제 위치	float64	mm
S_ActualVelocity	스핀들의 실제 속도	float64	mm/s
S_ActualAcceleration	스핀들의 실제 가속도	float64	mm/s/s
S_SetPosition	스핀들의 설정 위치	float64	mm
S_SetVelocity	스핀들의 설정 속도	float64	mm/s
S_SetAcceleration	스핀들의 설정 가속도	float64	mm/s/s
S_CurrentFeedback	스핀들 전류	float64	A
S_DCBusVoltage	스핀들 전압	float64	V
S_OutputCurrent	스핀들 아웃풋 전류	float64	A
S_OutputVoltage	스핀들 아웃풋 전압	float64	V
S_OutputPower	스핀들 아웃풋 전원	float64	kw
S_SystemInertia	토크 관성	float64	kg*m^2
M_CURRENT_PROGRAM_NUMBER	프로그램이 CNC에 나열된 번호	float64	
M_sequence_number	실행 중인 G-code 라인	float64	
M_CURRENT_FEEDRATE	스핀들의 순간 공급 속도	float64	
Machining_Process	현재 수행 중인 가공 단계	float64	

* float64 = 실수

• 2차 가공 데이터

- 1차 가공 데이터를 AI 모델 훈련을 위해 추가 가공한 데이터를 2차 가공 데이터라고 한다. 실제 AI 모델 훈련에 사용하기 위해 필요한 속성값을 필터링하고, 실사용 데이터만을 남긴다. 2차 가공 데이터는 X_train.csv, X_test.csv와 Y_train.csv, Y_test.csv 파일 4개로 구성되어 있다. X_train.csv와 X_test.csv 두 파일은 학습 및 평가 데이터셋 파일이며, (샘플수)x(독립변수)의 사이즈로 되어있다. 각각 18806x48, 13242x48 이다. Y_train.csv와 Y_test.csv 두 파일은 학습 및 평가 데이터셋의 라벨링 파일이며, (샘플수)x(라벨수)의 사이즈로 되어있다. 각각 18806x2, 13242x2 이다.

- 양품/불량품 기준 :

- 라벨은 양품, 불량품 2가지이며 분류 기준은 다음과 같다.

라벨	공정완료 유무 (machining_process)	육안검사 통과 유무 (passed_visual_inspection)
양품	O	O
불량품	X	X
	O	X

양품 및 불량품 분류 기준

- 실제 샘플의 양품 및 불량품은 다음과 같다.



실제 샘플 사진 : 좌)양품, 우)불량품

- 기술통계 :

- 2차 가공 데이터 독립변수의 기술통계는 다음과 같다.

구분	개수	평균	표준편차	최소값	중간값	최대값	최빈값
X_ActualPosition	18806	167.7072	20.82049	144	161	203	202
X_ActualVelocity	18806	3.652443	6.525964	-17.3	4	55.3	4
X_ActualAcceleration	18806	4.611939	102.7972	-1275	4	1444	4
X_SetPosition	18806	167.7053	20.82236	144	161	203	202
X_SetVelocity	18806	3.661146	6.537311	-17	4	55	5
X_SetAcceleration	18806	4.593118	86.56424	-997	4	1005	5
X_CurrentFeedback	18806	-0.52877	4.149693	-23.4	-0.533	27.1	-1.1
X_DCBusVoltage	18806	0.059127	0.044941	2.78E-19	5.39E-02	3.80E-01	2.79E-19
X_OutputCurrent	18806	326.6577	1.662001	320	327	331	327
X_OutputVoltage	18806	8.15966	9.960586	0	4.735	75.4	0
X_OutputPower	18806	0.000819	0.002038	-0.00606	1.88E-05	0.0388	0
Y_ActualPosition	18806	110.333	31.4084	75.4	100.45	163	162
Y_ActualVelocity	18806	3.470461	6.879965	-29.7	4	55.3	4
Y_ActualAcceleration	18806	5.240007	97.76467	-1255	4	1465	4

구분	개수	평균	표준편차	최소값	중간값	최대값	최빈값
Y_SetPosition	18806	110.329	31.40747	75.4	100.4	163	162
Y_SetVelocity	18806	3.492666	6.875905	-29.4	4	55	5
Y_SetAcceleration	18806	6.919201	96.74036	-997	4	1005	5
Y_CurrentFeedback	18806	-0.06994	4.452276	-27.8	0.06115	30.7	-1.33
Y_DC Bus Voltage	18806	0.057169	0.054494	2.68E-19	3.10E-02	4.30E-01	2.68E-19
Y_OutputCurrent	18806	325.904	2.417372	319	326	333	326
Y_OutputVoltage	18806	7.255391	10.70918	0	2.9	76.8	0
Y_OutputPower	18806	0.000879	0.002703	-0.00492	7.6E-07	0.0424	0
Z_ActualPosition	18806	61.15775	36.93214	30.5	34.7	124	32.5
Z_ActualVelocity	18806	3.6159	8.291877	-48.5	4	55.9	5
Z_ActualAcceleration	18806	4.194874	67.97119	-1255	4	1273	5
Z_SetPosition	18806	61.15469	36.92991	30.5	34.7	124	32.5
Z_SetVelocity	18806	3.616605	8.286712	-47	4	55	5
Z_SetAcceleration	18806	4.256775	64.26634	-997	4	1005	5
Z_CurrentFeedback	18806	0	0	0	0	0	0
Z_DC Bus Voltage	18806	0	0	0	0	0	0
Z_OutputCurrent	18806	0	0	0	0	0	0
Z_OutputVoltage	18806	0	0	0	0	0	0
Z_OutputPower	18806	-205.674	1107.363	-2147	-256.75	2155	-1156
S_ActualPosition	18806	37.24095	24.13951	2.98	56.3	58.6	58.3
S_ActualVelocity	18806	4.126496	28.91924	-147	4	153	5
S_ActualAcceleration	18806	-205.399	1107.401	-2147	-256.25	2155	-1156
S_SetPosition	18806	36.84016	24.33453	3	56.3	58.3	58.3
S_SetVelocity	18806	4.499468	6.739726	2.999999	4	105	5
S_SetAcceleration	18806	12.33538	11.45001	-8.28	16.9	75.4	-1.52
S_CurrentFeedback	18806	0.546167	0.489973	0	0.81	3.16	0
S_DC Bus Voltage	18806	323.3792	4.847668	290	323	332	327
S_OutputCurrent	18806	68.1149	58.37324	0	116	130	0
S_OutputVoltage	18806	0.103512	0.090861	-0.00296	0.154	0.569	0
S_OutputPower	18806	16.00495	0.857813	15	16	17	17
S_SystemInertia	18806	1.009252	0.572358	0	1	4	1
M_CURRENT_PROGRAM_NUMBER	18806	37.20026	43.00002	0	19	135	0
M_sequence_number	18806	23.36866	20.62185	3	15	50	50
M_CURRENT_FEEDRATE	18806	3.954748485	2.616196	0	4	8	1

구분	명칭
독립변수	샘플 별 특성 48개(X, Y, Z, 스피드 등)
종속변수	양품
	불량품

2차 가공 데이터 독립/종속 변수 정의

* 독립변수란, 다른 변수에 영향을 받지 않는 변수, 입력값을 나타낸다. 종속변수란 독립변수의 변화에 따라 바뀌는 변수이며, 결과값을 나타낸다.

2.2 분석 모델 소개

1) AI 분석모델

- 해당 AI 방법론(알고리즘) 선정 이유 기술

- CNC 가공 데이터 (1차 전처리 데이터) 분석 방법 : 지도학습(supervised learning)을 적용한 데이터 분석

- 제품의 CNC 공정결과는 크게 양품 및 불량품 2가지로 분류할 수 있다. 2가지 케이스에 대해 각각 별개의 라벨로 설정하여 데이터 터셋을 구성한다. 가공조건에 따른 최적의 가공불량 예측과정을 위해 이에 좋은 성능을 보이는 심층 신경망을 디자인하여 학습 및 테스트 하였다.

- 적용하고자 하는 AI 분석 방법론(알고리즘)의 구체적 소개

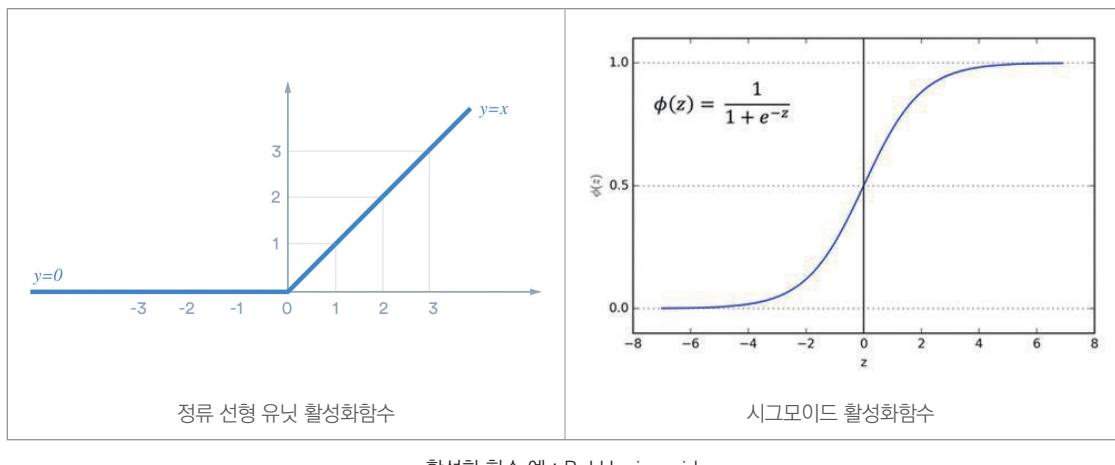
- 심층 신경망(deep neural network)

- 심층 신경망(deep neural network)은 신경망이라 불리는 구조를 연속적으로 깊게 쌓은 후 분류 및 회귀 등의 다양한 목적을 위해 학습시키는 AI 알고리즘이다. 신경망은 아래 그림처럼 여러 노드로 구성되어 있으며 은닉층(hidden layer)이라고도 불린다. 신경망에 속하는 노드는 입력층의 데이터를 받고 행렬 곱 등의 연산을 거친 후 결과값을 출력한다. 이러한 신경망은 선형적인 변환 뿐만 아니라 여러 활성화 함수(activation function)라 부르는 비선형적인 연산을 수행하기도 한다. 활성화 함수는 일반적으로 시그모이드(sigmoid) 함수와 정류 선형 유닛(Rectified Linear Unit)함수를 사용한다. 정류 선형 유닛 함수는 0보다 작은 입력은 0으로 만들고, 0보다 큰 입력은 그대로 출력한다. 시그모이드 함수는 입력값을 0과 1 사이의 결과값으로 제한된다.

신경망의 이러한 특성을 여러 층으로 연속적으로 쌓아가게 될수록 진가를 발휘하는데, 매우 깊은 신경망은 수많은 함수를 구현할 수 있다는 장점이 있다. 신경망을 현재 경사 하강법이라는 기법을 사용하여 학습을 진행한다. 신경망의 예측 결과값과 데이터의 실제값이 어느 정도 일치하는지 등의 기준으로 신경망의 성능을 평가하게 되는데 이러한 평가 수치를 손실함수(loss function)라고 부른다. 각 신경망의 모수(parameter)를 변화시킴에 따라 이 손실함수의 값은 변화하게 되므로, 모델의 학습 과정에서는 손실함수를 최소화하는 신경망의 모수를 찾는 방식으로 학습을 진행하게 된다. 분류를 위한 심층 신경망은 임의의 다수의 신경망을 연속적으로 쌓고, 최종적으로 목표치(양품 혹은 불량품)인 확률을 출력하도록 구성한다. 신경망의 구조

적 특성상 노드의 개수, 노드별 활성화함수, 신경망의 개수, 손실함수 등의 조작 가능한 값이 매우 많이 존재한다. 따라서 모델 학습시에 주어진 데이터에 적합한 값을 찾는 것이 중요하다.

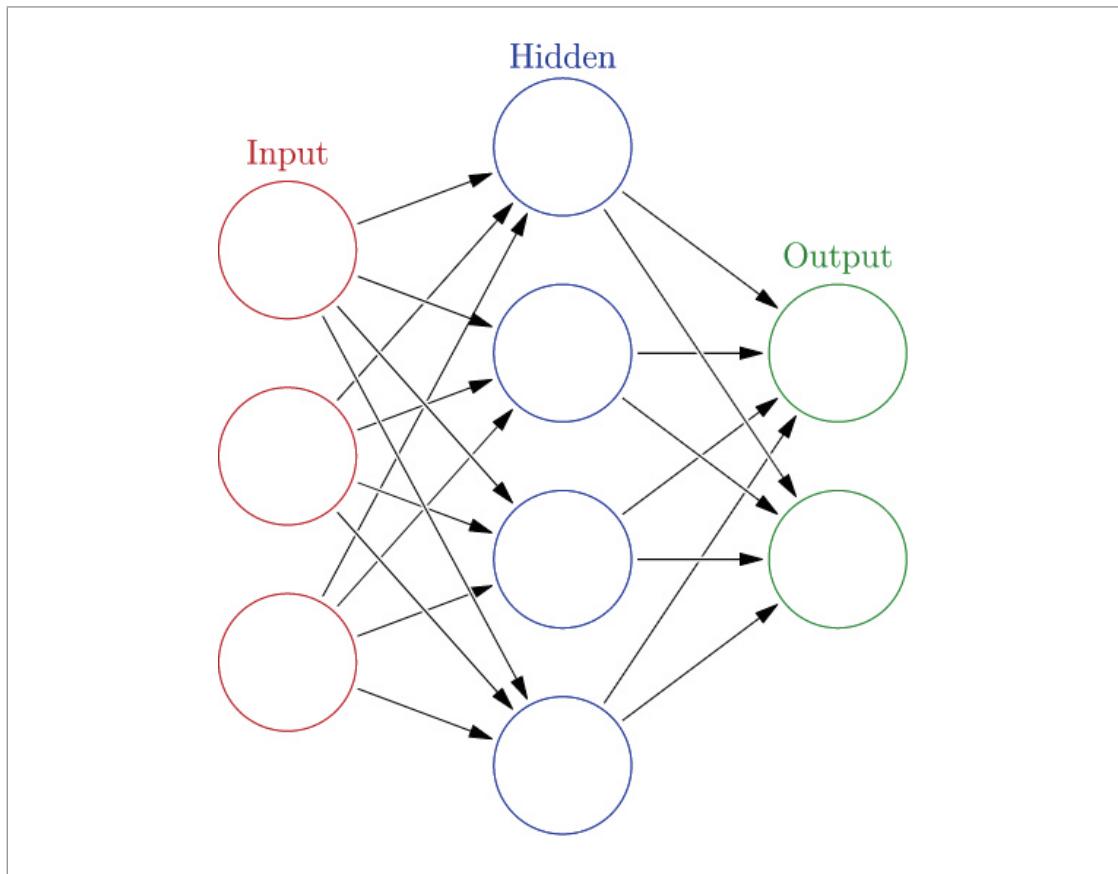
모델의 학습 진행 상황을 판단하는 손실함수(loss)는 학습 목표에 따라 여러 가지가 쓰인다. 양품과 불량품을 가려내는 경우에는 이 진 교차엔트로피(binary_crossentropy)를 사용하며, 교차엔트로피는 분류 목적의 AI 알고리즘을 학습시킬 때 가장 많이 사용되는 함수로서, 실제 목표치(0 또는 1)와 근접할 때 낮은 값을 가지고 반대의 경우에 높은 값을 가지게 된다.



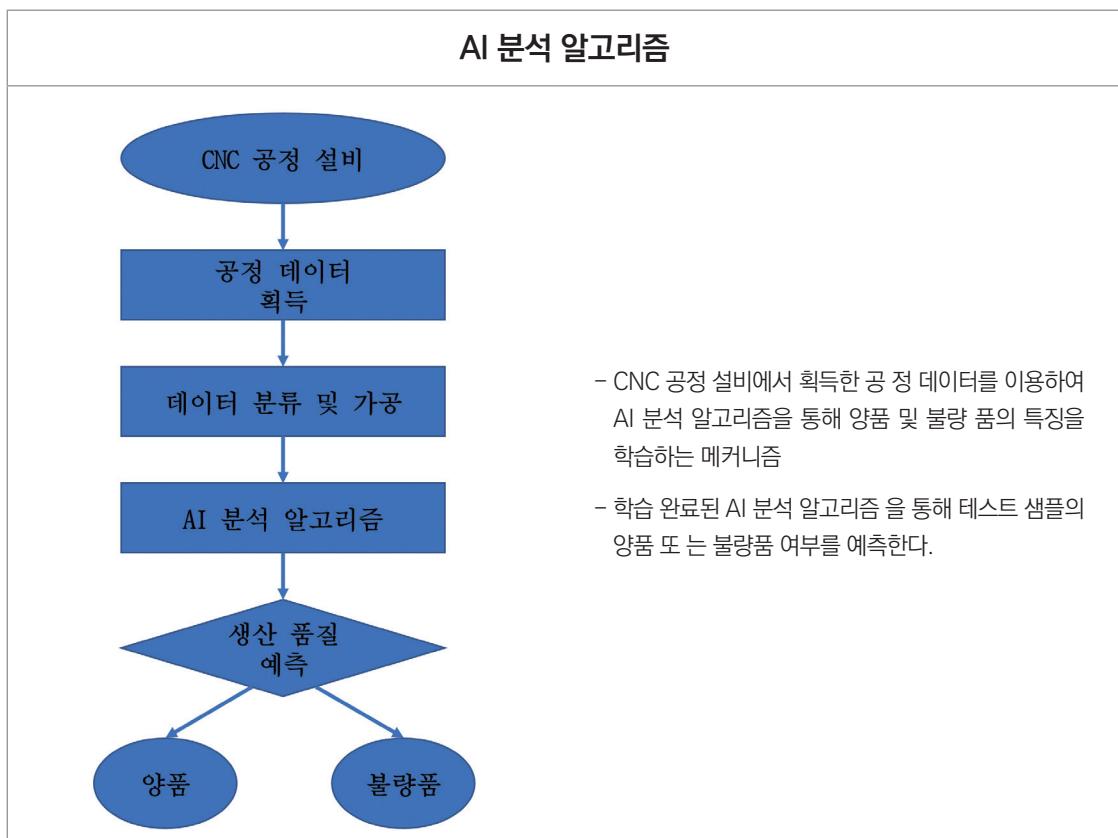
• AI 분석 방법론(알고리즘) 구축 절차 설명

- 지도학습 상황에서 활용되는 AI 분석 알고리즘

- 지도학습을 이용한 여러 가지 알고리즘 중 인공신경망 (artificial neural network)이라는 방법이 있다. 이 방법은 기계 학습과 인지과학에서 생물학의 신경망에서 영감을 얻은 통계학적 학습 알고리즘이다. 인공신경망은 시냅스의 결합으로 네트워크를 형성한 인공 뉴런(노드)이 학습을 통해 시냅스의 결합 세기를 변화 시켜, 문제 해결 능력을 가지는 모델 전반을 가리킨다. 일반적으로 인공 신경망은 노드들의 그룹으로 연결되어 있으며 이들은 뇌의 방 대한 뉴런의 네트워크과 유사하다. 위 그림에서 각 원모양의 노드는 인공 뉴런을 나타내고 화살표는 하나의 뉴런의 출력에서 다른 하나의 뉴런으로의 입력을 나타낸다. 본 AI 분석 방법론에서는 위의 인공신경망에서 다수의 뉴런을 쌓은 형태인 심층 신경망(deep neural network)을 활용한다.



인공 신경망 예시



AI 분석 알고리즘 도식

2.3 분석 체험

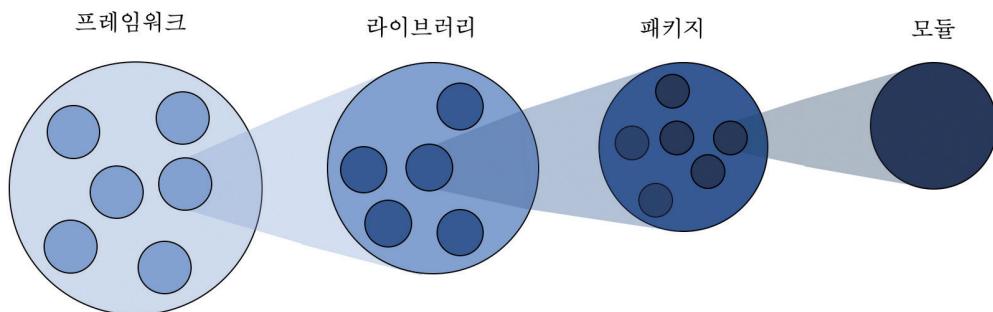
1) 필요SW, 패키지 설치 방법 및 절차 가이드

- SW 설치는 ‘부록(분석환경 구축을 위한 설치 가이드)’ 참고

• 패키지 설치 가이드

- 패키지 설치 전 파일 관련 용어 정리

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



파일 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-py3-none-any.whl (60 kB)
    100% |██████████| 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->datetime) (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 keras==2.1.6  
!pip install tensorflow-gpu==1.15  
!pip install matplotlib  
!pip install -U scikit-learn  
!pip install pandas
```

* 이번 분석을 위한 필요 패키지 및 모듈은 위와 같다. 가이드북 실습 전에 설치하면 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
Pygments	2.9.0

* 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

예) 다양한 임포트 방법으로 Press_RawDataSet.xlsx를 읽어보기

i. import

import를 사용하여 해당 모듈 전체를 임포트한다.

```
import pandas  
pandas.read_csv('./label_data.csv')
```

pandas를 임포트를 하고 ‘.’을 사용하여 pandas 의‘read_csv’ 함수를 이용하여 ‘label_data.csv’ 파일을 불러온다.

ii. from, import

해당 모듈에서 특정한 타입만 임포트한다.

예) pandas에서 ‘read_csv’만 임포트

```
from pandas import read_csv  
read_csv('./label_data.csv')
```

from, import를 사용해서 필요한 함수만 import로 사용할 수 있다.

iii. * import

해당 모듈 내에 정의된 모든 것을 임포트하나 다른 모듈들과 섞일 수 있으므로 일 반적으로 사용이 권장되지 않는다.

```
from pandas import *
```

iv. as

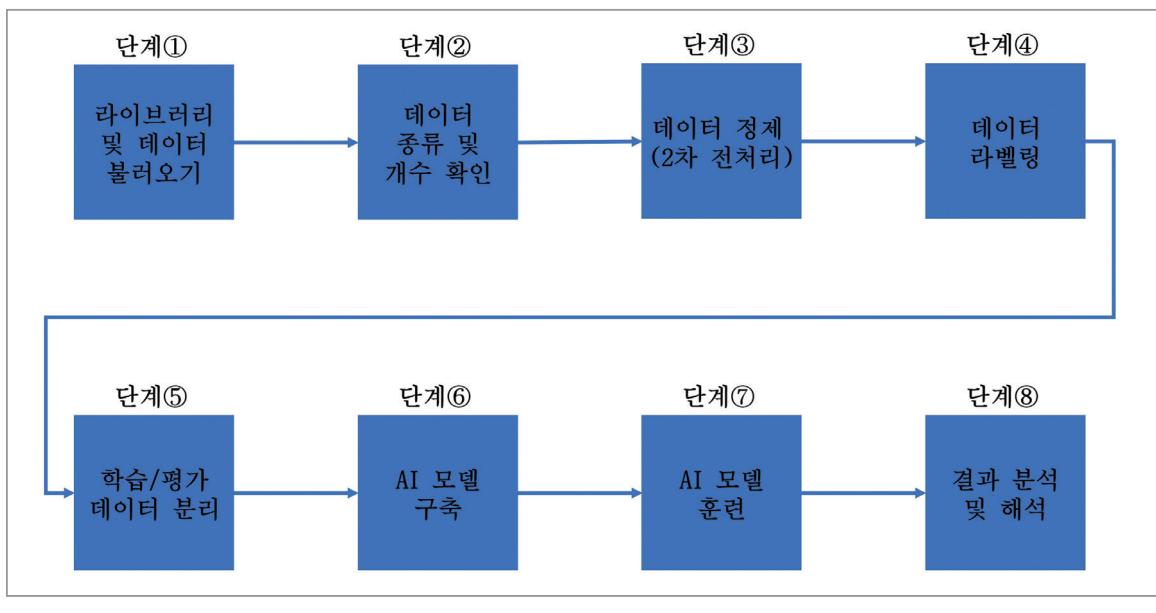
모듈 임포트할 때, 별명(alias)를 지정할 때 사용한다.

```
import pandas as pd  
pd.read_csv('./label_data.csv')
```

‘pandas’를 ‘pd’로 별명 지정 후에는 ‘pd’로 불러올 수 있다.

* 한번 설치한 모듈 및 패키지는 영구적이며 필요시 버전 업그레이드가 필요하다. 또한, 분석, 시각화 등 상황에 따라 필요한 패키지를 임포트 한다.

2) 분석 단계별 프로세스 - Flow Chart



[단계①] 라이브러리 및 데이터 불러오기

제일 먼저, 전체 단계 진행에 필요한 라이브러리와 심층 신경망 학습을 위한 데이터 불러오기 방법을 설명하고, 실습해보는 예제를 준비하였다.

①-1. 필요 라이브러리

```
from __future__ import print_function
import pandas as pd
import numpy as np
import glob
from sklearn.preprocessing import MinMaxScaler, StandardScaler

# import function libraries
import numpy as np
import keras
import os, sys, math, copy
import scipy.io as sio
import tensorflow as tf
from keras.models import Model, Sequential
from keras.engine import Layer, InputSpec
from keras.optimizers import RMSprop, SGD, Adam
from keras import initializers, regularizers, constraints
from keras.callbacks import ModelCheckpoint, LearningRateScheduler, History
from keras.layers import Dense, Dropout, Activation, Flatten, Input
from keras import backend as K
from keras.utils import np_utils

sys.setrecursionlimit(10000)

import matplotlib.pyplot as plt
```

[그림 1] 필요 라이브러리 import

- 그림 1의 실습 코드와 같이 분석 체험에 필요한 모든 라이브러리를 추가한다.

①-2. 훈련 데이터 불러오기

```
train_sample = pd.read_csv("train.csv", header=0, encoding='utf-8')
path = r'./CNC Virtual Data set _v2'
all_files = glob.glob(path + "W*.csv")
# change data type
train_sample_np = np.array(train_sample.copy())

# load csv file
li_df= []
for filename in all_files:
    df = pd.read_csv(filename, index_col=None, header=0)
    li_df.append(df)
```

[그림 2] train.csv 데이터 파일 불러오기

- 그림 2는 전체 26개의 csv파일의 데이터를 불러오는 예제이다. `read_csv()`는 경로의 csv 파일을 받아 변수 `train_sample`로 출력한다. ‘train.csv’ 파일은 CNC 가공 실적 데이터 파일로, 25개 샘플에 대한 데이터를 포함하고 있다. 해당 파일을 불러

을 시, 현재 작성 중인 python 코드 파일과 ‘train.csv’파일이 같은 폴더에 저장되어 있어야 한다.

train_sample의 데이터 타입은 DataFrame으로, 앞으로 사용할 때 불편하기 때문에 python 사용 시에 많이 쓰이는 numpy array 형태의 데이터 타입으로 변환하기 위해 np.array() 함수를 이용한다. 변수 train_sample_np라는 데이터를 얻을 수 있다.

csv 파일의 폴더가 다른 경로에 있을 경우, 경로 설정은 다음과 같다. 25개 샘플 각각에 대한 데이터는 먼저 해당 데이터들이 들어있는 파일 경로를 변수 path에 저장한다. path 설정 방법은 그림 3과 같다. 개별 생산 단위 데이터 파일인 ‘CNC Virtual Data set_v2’파일의 위치를 확인하고, 해당 파일의 파일 경로를 복사한다. 복사한 파일 경로는 path의 ‘ ’안에 붙여넣기 한다. 이후, glob함수를 이용하여 파일 경로에 있는 모든 csv파일을 변수 li_df 저장한다.



[그림 3] path의 경로 설정 예시

[단계②] 데이터 종류 및 개수 확인

단계②에서는 앞선 단계①에서 불러온 데이터의 종류 및 개수를 확인한다. 이를 통해 데이터가 포함하는 독립 변수를 확인하고, 분류 기준에 따른 샘플 개수를 확인할 수 있다.

②-1. 데이터 종류 확인

train_sample

No	material	feedrate	clamp_pressure	tool_condition	machining_finalized	passed_visual_inspection
0	1 aluminum	6	4.0	unworn	yes	yes
1	2 aluminum	20	4.0	unworn	yes	yes
2	3 aluminum	6	3.0	unworn	yes	yes
3	4 aluminum	6	2.5	unworn	no	NaN
4	5 aluminum	20	3.0	unworn	no	NaN
5	6 aluminum	6	4.0	worn	yes	no
6	7 aluminum	20	4.0	worn	no	NaN
7	8 aluminum	20	4.0	worn	yes	no
8	9 aluminum	15	4.0	worn	yes	no
9	10 aluminum	12	4.0	worn	yes	no
10	11 aluminum	3	4.0	unworn	yes	yes
11	12 aluminum	3	3.0	unworn	yes	yes
12	13 aluminum	3	4.0	worn	yes	yes
13	14 aluminum	3	3.0	worn	yes	yes
14	15 aluminum	6	3.0	worn	yes	yes
15	16 aluminum	20	3.0	worn	no	NaN
16	17 aluminum	3	2.5	unworn	yes	yes
17	18 aluminum	3	2.5	worn	yes	yes
18	19 aluminum	15	4.0	worn	yes	no
19	20 aluminum	12	4.0	unworn	no	NaN
20	21 aluminum	3	4.0	unworn	yes	no
21	22 aluminum	20	3.0	worn	yes	yes
22	23 aluminum	3	4.0	worn	no	NaN
23	24 aluminum	3	3.0	unworn	yes	yes
24	25 aluminum	6	2.5	worn	yes	yes

[그림 4] 변수 train_sample의 데이터 종류 확인

- 그림 4를 보자. train_sample을 확인하면 train.csv 파일의 데이터가 그대로 불러온 것을 확인할 수 있다. 행 순서대로 No, material, feedrate, clamp_pressure, tool_condition, machining_finalized, passed_visual_inspection 정보를 담고 있다.

df

	X_ActualPosition	X_ActualVelocity	X_ActualAcceleration		X_SetPosition	X_SetVelocity						
	X_SetAcceleration		X_CurrentFeedback		X_DC BusVoltage	X_OutputCurrent						
	X_OutputVoltage ...	S_CurrentFeedback		S_DC BusVoltage	S_OutputCurrent							
	S_OutputVoltage	S_OutputPower	S_SystemInertia	M_CURRENT_PROGRAM_NUMBER								
	M_sequence_number	M_CURRENT_FEEDRATE		Machining_Process								
0	176.0	4.9750	-1.250	176.0	5.0	5.0	-1.420	0.0227	327	0.355	.	.
	0.499	2.710000e-19		327	0.0	0.000003		17	1	2	5	0
	Prep											
1	176.0	5.0250	23.775	176.0	5.0	5.0	-1.740	0.0224	327	0.589	.	.
	0.790	2.710000e-19		327	0.0	-0.000003		17	1	0	5	0
	Prep											
2	176.0	4.9750	-1.250	176.0	5.0	5.0	0.180	0.0329	327	2.190	.	.
	-1.300	2.710000e-19		327	0.0	-0.000006		17	1	0	5	0
	Prep											
3	176.0	4.9625	-13.750	176.0	5.0	5.0	-0.619	0.0198	327	1.010	.	.
	-3.810	2.710000e-19		327	0.0	0.000002		17	1	0	5	0
	Prep											
4	176.0	4.9750	-4.400	176.0	5.0	5.0	-0.779	0.0311	327	0.372	.	.
	0.790	2.710000e-19		327	0.0	0.000000		17	1	0	5	0
	Prep											
...
560	179.0	3.5625	8.150	179.0	3.5	5.0	-0.941	0.0215	328	1.790	.	.
	0.244	2.770000e-19		328	0.0	-0.000003		17	1	0	2	0
	End											
561	178.5	3.5350	30.000	178.5	3.5	5.0	-0.780	0.0202	328	1.070	.	.
	0.244	2.770000e-19		328	0.0	0.000000		17	1	0	2	0
	End											
562	178.5	3.5475	17.500	178.5	3.5	5.0	0.501	0.0190	328	0.990	.	.
	0.128	2.770000e-19		328	0.0	0.000000		17	1	0	2	0
	End											
563	178.5	3.4475	11.250	178.5	3.5	5.0	-0.298	0.0193	328	1.370	.	.
	0.657	2.770000e-19		328	0.0	-0.000008		17	1	0	2	0
	End											
564	178.0	3.5625	1.850	178.0	3.5	5.0	0.340	0.0258	328	1.090	.	.
	0.244	2.770000e-19		328	0.0	-0.000003		17	1	0	2	0
	End											

[그림 5] 샘플이 포함하는 독립변수 확인

- 단계①에서 우리는 각 샘플의 48개 독립변수 데이터를 변수 df에 불러왔고, 전체 샘플 데이터를 변수 li_df에 불러왔다. 그림 5와 같이 변수 df를 확인해보자.
각 행은 1차 전처리 데이터 변수인 독립변수 48개의 정보를 담고 있다.

```
print(all_files)
```

```
['C:\\Users\\Server\\Desktop\\hjcho\\CNC\\CNC Virtual Data set _v2\\experiment_01.csv',
'C:\\Users\\Server\\Desktop\\hjcho\\CNC\\CNC Virtual Data set _v2\\experiment_02.csv',
'C:\\Users\\Server\\Desktop\\hjcho\\CNC\\CNC Virtual Data set _v2\\experiment_03.csv',
'C:\\Users\\Server\\Desktop\\hjcho\\CNC\\CNC Virtual Data set _v2\\experiment_04.csv',
'C:\\Users\\Server\\Desktop\\hjcho\\CNC\\CNC Virtual Data set _v2\\experiment_05.csv',
'C:\\Users\\Server\\Desktop\\hjcho\\CNC\\CNC Virtual Data set _v2\\experiment_06.csv',
'C:\\Users\\Server\\Desktop\\hjcho\\CNC\\CNC Virtual Data set _v2\\experiment_07.csv',
'C:\\Users\\Server\\Desktop\\hjcho\\CNC\\CNC Virtual Data set _v2\\experiment_08.csv',
'C:\\Users\\Server\\Desktop\\hjcho\\CNC\\CNC Virtual Data set _v2\\experiment_09.csv',
'C:\\Users\\Server\\Desktop\\hjcho\\CNC\\CNC Virtual Data set _v2\\experiment_10.csv',
'C:\\Users\\Server\\Desktop\\hjcho\\CNC\\CNC Virtual Data set _v2\\experiment_11.csv',
'C:\\Users\\Server\\Desktop\\hjcho\\CNC\\CNC Virtual Data set _v2\\experiment_12.csv',
'C:\\Users\\Server\\Desktop\\hjcho\\CNC\\CNC Virtual Data set _v2\\experiment_13.csv',
'C:\\Users\\Server\\Desktop\\hjcho\\CNC\\CNC Virtual Data set _v2\\experiment_14.csv',
'C:\\Users\\Server\\Desktop\\hjcho\\CNC\\CNC Virtual Data set _v2\\experiment_15.csv',
'C:\\Users\\Server\\Desktop\\hjcho\\CNC\\CNC Virtual Data set _v2\\experiment_16.csv',
'C:\\Users\\Server\\Desktop\\hjcho\\CNC\\CNC Virtual Data set _v2\\experiment_17.csv',
'C:\\Users\\Server\\Desktop\\hjcho\\CNC\\CNC Virtual Data set _v2\\experiment_18.csv',
'C:\\Users\\Server\\Desktop\\hjcho\\CNC\\CNC Virtual Data set _v2\\experiment_19.csv',
'C:\\Users\\Server\\Desktop\\hjcho\\CNC\\CNC Virtual Data set _v2\\experiment_20.csv',
'C:\\Users\\Server\\Desktop\\hjcho\\CNC\\CNC Virtual Data set _v2\\experiment_21.csv',
'C:\\Users\\Server\\Desktop\\hjcho\\CNC\\CNC Virtual Data set _v2\\experiment_22.csv',
'C:\\Users\\Server\\Desktop\\hjcho\\CNC\\CNC Virtual Data set _v2\\experiment_23.csv',
'C:\\Users\\Server\\Desktop\\hjcho\\CNC\\CNC Virtual Data set _v2\\experiment_24.csv',
'C:\\Users\\Server\\Desktop\\hjcho\\CNC\\CNC Virtual Data set _v2\\experiment_25.csv']
```

[그림 6] 샘플 별 데이터 파일의 경로 확인

- `print()` 함수를 통해 `all_files`에 저장된 전체 25개 샘플 데이터 파일의 경로 및 파일명을 볼 수 있다. 그림 6을 참고하자.

②-2. 데이터 개수 확인

```
# count the number of pass/fail items
nb_pass = 0
nb_pass_half = 0
nb_defective = 0
for i in range(len(train_sample_np)):
    if train_sample_np[i,5] == 'no':
        nb_defective += 1
    if train_sample_np[i,5] == 'yes' and train_sample_np[i,6] == 'yes':
        nb_pass += 1
    if train_sample_np[i,5] == 'yes' and train_sample_np[i,6] == 'no':
        nb_pass_half += 1

print('양품 샘플 개수 : ', nb_pass)
print('공정 마쳤으나 육안검사 통과 못한 샘플 개수 : ', nb_pass_half)
print('공정 중지된 샘플 개수 : ', nb_defective)
print('전체 샘플 개수 : ', nb_pass + nb_pass_half + nb_defective)
```

```
양품 샘플 개수 : 13
공정 마쳤으나 육안검사 통과 못한 샘플 개수 : 6
공정 중지된 샘플 개수 : 6
전체 샘플 개수 : 25
```

[그림 7] 종속변수 별 샘플 수 확인

- 읽어온 데이터 파일의 전체 25개의 샘플 중에서, 2-1에서 살펴본대로 train_sample의 6, 7번째 행을 참고하여 종속변수 별 샘플 수를 세기 위해 if문이 포함된 for문을 동작시킨다.

3가지의 공정 결과를 분류하기 위해, train_sample_np에서 6,7 번째 행의 데이터를 확인한다. 6번째 행은 machining_finalized 항목으로, 공정이 끝까지 진행되었는지의 여부를 yes/no로 나타낸다. 7번째 행은 passed_visual_inspection 항목으로, 공정이 끝난 샘플에 대한 육안검사 통과 여부를 yes/no로 나타내고 있다. 공정이 끝까지 진행되지 않은 샘플은 passed_visual_inspection 데이터가 NaN으로 나타난다.

for문을 이용하여 25개 샘플을 차례대로 훑으며, 각 if문에서 해당 데이터를 확인한다. 변수 nb_pass, nb_pass_half, nb_defective는 각각 공정완료 및 육안검사 통과한 샘플, 공정 완료되었으나 육안검사에서 통과하지 못한 샘플, 공정미완료 샘플의 수를 기록하였다. 그림 7을 확인하자.

[단계③] 데이터 정제(2차 전처리)

이번 단계에서는, 앞서 불러온 데이터(1차 전처리)를 AI 모델 학습 및 테스트를 위한 정제(2차 전처리) 작업을 진행하는 예제를 제공한다.

③-1. 사용자 정의 함수 선언

```
def tool_condition(input):
    for i in range(len(input)):
        if input[i,4] == 'unworn':
            input[i,4] = 0
        else:
            input[i,4] = 1
    return input
```

```
def item_inspection(input):
    for i in range(len(input)):
        if input[i,5] == 'no':
            input[i,6] = 2
        elif input[i,5] == 'yes' and input[i,6] == 'no':
            input[i,6] = 1
        elif input[i,5] == 'yes' and input[i,6] == 'yes':
            input[i,6] = 0
    return input
```

```

def machining_process(input):
    for i in range(len(input)):
        if input[i,47] =='Prep':
            input[i,47] =0
        elif input[i,47] =='Layer 1 Up':
            input[i,47] =1
        elif input[i,47] =='Layer 1 Down':
            input[i,47] =2
        elif input[i,47] =='Layer 2 Up':
            input[i,47] =3
        elif input[i,47] =='Layer 2 Down':
            input[i,47] =4
        elif input[i,47] =='Layer 3 Up':
            input[i,47] =5
        elif input[i,47] =='Layer 3 Down':
            input[i,47] =6
        elif input[i,47] =='Repositioning':
            input[i,47] =7
        elif input[i,47] =='End'or'end':
            input[i,47] =8
        elif input[i,47] =='Starting':
            input[i,47] =9
    return input

```

[그림 8] 데이터 정제를 위한 사용자 함수 정의

- 효과적인 심층 신경망 학습을 위해서는 데이터 전처리 과정이 필수적이다. 해당 데이터를 효과적으로 정제하기 위해 사용자 함수를 정의하였다. 3개의 사용자 정의 함수를 그림 8에서 소개한다.

- tool_condition()
 - tool_condition() 함수는 입력값으로 불러온 ‘train.csv’ 파일을 받고, 출력값으로 각 샘플의 tool_condition에 대해 ‘마모되지않음(unworn)’ 이면 0을, ’마모됨(worn)’ 이면 1을 갖는다. for문으로 전체 샘플에 대해 확인하고, if문으로 해당 샘플의 tool_condition을 확인한다.

- item_inspection()
 - item_inspection() 함수는 입력값으로 불러온 ‘train.csv’ 파일을 받고, 출력값으로 각 샘플의 machining_finalized(공정완료)와 passed_visual_inspection(육안검사) 두 가지 항목을 확인하고, 공정미완료(2), 공정완료 및 육안검사 불합격(1), 공정완료 및 육안검사 합격(0)의 3가지 값을 갖는다. for문으로 전체 샘플에 대해 확인하고, if문으로 해당 샘플의 machining_finalized(공정완료)와 passed_visual_inspection(육안검사) 여부를 확인한다.

- machining_process()

- machining_process() 함수는 입력값으로 앞선 두 사용자 함수를 거친 출력값 data_pass, data_pass_half, data_fail을 갖는다. 해당 변수들은 3-2에서 상세하게 설명되어 있다. 개별 생산 단위 데이터에는 machining_process 항목이 있다. 이 항목은 기계의 공정상태를 나타내는 데이터로, 총 10가지로 되어있으며 다음과 같다 : Prep, Layer 1 Up/Down, Layer 2 Up/Down, Layer 3 Up/Down, Repositioning, End(end), Starting.

해당 데이터는 학습 데이터로 사용 시에 문자가 아닌 숫자이어야 하기 때문에 순서대로 0~9로 대체하는 작업이 필요하다. 이 작업을 machining_process()가 진행한다. for문으로 전체 열에 대해 확인하고, if문으로 해당 열의 machining_process(공정과정)의 상태를 확인한 후, 해당하는 숫자(0~9)를 대체하여 출력값으로 갖는다.

③-2. 사용자 정의 함수의 사용을 통한 데이터 분류

```
# Modifying train.csv for training,  
# - [tool_condition] : unworn/worn -> 0 / 1  
# - [item_inspection] : machining_finalized & passed -> yes & yes / yes & no / no : 0 / 1 / 2  
# - delete 'material' column and 'No' column  
train_sample_info = np.array(train_sample_np.copy())  
train_sample_info = tool_condition(train_sample_info)  
train_sample_info = item_inspection(train_sample_info)  
print(train_sample_info)
```

```
[1 'aluminum' 6 4.0 0 'yes' 0]  
[2 'aluminum' 20 4.0 0 'yes' 0]  
[3 'aluminum' 6 3.0 0 'yes' 0]  
[4 'aluminum' 6 2.5 0 'no' 2]  
[5 'aluminum' 20 3.0 0 'no' 2]  
[6 'aluminum' 6 4.0 1 'yes' 1]  
[7 'aluminum' 20 4.0 1 'no' 2]  
[8 'aluminum' 20 4.0 1 'yes' 1]  
[9 'aluminum' 15 4.0 1 'yes' 1]  
[10 'aluminum' 12 4.0 1 'yes' 1]  
[11 'aluminum' 3 4.0 0 'yes' 0]  
[12 'aluminum' 3 3.0 0 'yes' 0]  
[13 'aluminum' 3 4.0 1 'yes' 0]  
[14 'aluminum' 3 3.0 1 'yes' 0]  
[15 'aluminum' 6 3.0 1 'yes' 0]  
[16 'aluminum' 20 3.0 1 'no' 2]  
[17 'aluminum' 3 2.5 0 'yes' 0]  
[18 'aluminum' 3 2.5 1 'yes' 0]  
[19 'aluminum' 15 4.0 1 'yes' 1]  
[20 'aluminum' 12 4.0 0 'no' 2]  
[21 'aluminum' 3 4.0 0 'yes' 1]  
[22 'aluminum' 20 3.0 1 'yes' 0]  
[23 'aluminum' 3 4.0 1 'no' 2]  
[24 'aluminum' 3 3.0 0 'yes' 0]  
[25 'aluminum' 6 2.5 1 'yes' 0]]
```

[그림 9] 함수 사용 후 변수의 데이터 확인

- 그림 9와 같이, train_sample_np 데이터를 train_sample_info라는 변수로 새롭게 복사하고, 앞서 선언한 함수들 중 tool_condition()과 item_inspection() 함수들을 사용한다.

tool_condition()과 item_inspection() 두 함수를 사용한 결과는 그림 10과 같다. 각 열은 7행으로 이루어져 있다. train.csv 파일의 구성과 마찬가지로, 차례대로 No, material, feedrate, clamp_pressure, tool_condition, machining_finalized, passed_visual_inspection에 대한 데이터이다. tool_condition 데이터는 기존에는 unworn/worn이었으나, tool_condition() 함수 사용으로 0/1로 바뀌었다. 마찬가지로 마지막 행 passed_visual_inspection 역시 기존엔 'yes'/'no'/'NaN'으로

표시되었으나 item_inspection() 함수 사용으로 0/1/2로 바뀌었다. 그 결과, 데이터 중 문자로 되어있던 값이 정수로 변환된다.

```
train_sample_info = np.delete(train_sample_info,5,1)
train_sample_info = np.delete(train_sample_info,0,1)
train_sample_info = np.delete(train_sample_info,0,1)
print(train_sample_info)
```

```
[[6 4.0 0 0]
 [20 4.0 0 0]
 [6 3.0 0 0]
 [6 2.5 0 2]
 [20 3.0 0 2]
 [6 4.0 1 1]
 [20 4.0 1 2]
 [20 4.0 1 1]
 [15 4.0 1 1]
 [12 4.0 1 1]
 [3 4.0 0 0]
 [3 3.0 0 0]
 [3 4.0 1 0]
 [3 3.0 1 0]
 [6 3.0 1 0]
 [20 3.0 1 2]
 [3 2.5 0 0]
 [3 2.5 1 0]
 [15 4.0 1 1]
 [12 4.0 0 2]
 [3 4.0 0 1]
 [20 3.0 1 0]
 [3 4.0 1 2]
 [3 3.0 0 0]
 [6 2.5 1 0]]
```

[그림 10] 불필요한 데이터 행 삭제

- 하지만 현재의 변수 train_sample_info에는 학습에 필요 없는 행을 가지고 있다. 첫 번째 행(No.)과 두 번째 행(material), 그리고 다섯 번째 행(machining_finalized)을 삭제하도록 한다. np.delete() 함수를 이용하여 데이터의 행을 삭제할 수 있다. 그 결과 그림 11과 같다.

각 열은 4개의 행으로 되어있으며, 순서대로 feedrate, clamp_pressure, tool_condition() 함수로 정해진 값, item_inspection() 함수로 정해진 값으로 이루어져 있다. 모두 숫자 데이터로 이루어져 있어 학습 데이터로 사용 가능하다.

```

k = 0
li_pass = []
li_pass_half = []
li_fail = []
for filename in all_files:
    df = pd.read_csv(filename, index_col=None, header=0)

    if train_sample_info[k,3] == 0:
        li_pass.append(df)
    elif train_sample_info[k,3] == 1:
        li_pass_half.append(df)
    else :
        li_fail.append(df)

    k += 1

frame01 = pd.concat(li_pass, axis=0, ignore_index=True)
frame02 = pd.concat(li_pass_half, axis=0, ignore_index=True)
frame03 = pd.concat(li_fail, axis=0, ignore_index=True)
data_pass = np.array(frame01.copy())
data_pass_half = np.array(frame02.copy())
data_fail = np.array(frame03.copy())
print('공정완료 및 육안검사 합격한 전체 데이터 수 : ',len(data_pass))
print('공정완료 및 육안검사 불합격한 전체 데이터 수 : ',len(data_pass_half))
print('공정 미완료한 전체 데이터 수 : ',len(data_fail))

```

공정완료 및 육안검사 합격한 전체 데이터 수 : 22645
 공정완료 및 육안검사 불합격한 전체 데이터 수 : 6175
 공정 미완료한 전체 데이터 수 : 3228

[그림 11] 데이터 정제 전 불필요한 변수 삭제

- 정리한 데이터를 바탕으로, 학습에 사용할 각 샘플데이터들을 분류하는 작업을 진행한다. 앞선 단계에서 all_files에 저장한 정보를 이용하여 각각의 파일에서 데이터를 불러올 수 있다. 샘플 데이터를 분류한 후 모이게 될 변수 li_pass, li_pass_half, li_fail을 선언한다. for문을 이용하여 25개 샘플들에 순서대로 접근하여 해당 샘플의 값을 확인한 후 해당 변수에 모은다. if문을 통해 해당 샘플의 값을 확인하며, append() 함수를 이용하여 선언한 데이터 변수에 해당 샘플 데이터를 이어 붙인다. li_pass, li_pass_half, li_fail은 리스트 형태의 데이터이다. 주로 사용하는 numpy array 형태로 바꾸기 위해, 먼저 세 변수들을 pd.concat() 함수를 이용하여 샘플 별로 구분되어 있던 index를 무시하도록 한다. 이후 np.array()와 copy() 함수들을 이용하여 DataFrame 타입인 데이터 변수 frame01, frame02, frame03을 데이터만 복제한 후 numpy array 타입인 data_pass, data_pass_half, data_fail로 변수 선언한다.

이렇게 분류한 샘플 데이터 변수 전체의 개수는 그림 12와 같다.

```
print(data_pass.shape)
print(data_pass_half.shape)
print(data_fail.shape)
```

```
(22645, 48)
(6175, 48)
(3228, 48)
```

[그림 12] 분류한 데이터 및 독립변수 개수 확인

- print() 함수를 이용하여 분류 데이터의 사이즈를 확인한다. data_pass는 총 22645열, data_pass_half는 총 6175열, data_fail은 총 3228열로 이루어져 있으며, 세 변수 모두 48행의 독립변수 데이터를 가지고 있다.
단계②에서 확인한 13개의 양품 데이터는 data_pass라는 하나의 변수로 저장되었다. 공정을 마쳤으나 육안검사를 통과하지 못한 6개의 불량품 데이터는 data_pass_half라는 변수에, 공정을 마치지 못한 6개의 불량품 데이터는 data_fail이라는 변수에 저장되었음을 확인할 수 있다.

```
# Modifying experiment data
# - machining_process : From "Prep" to "End" -> 0~9
data_pass = machining_process(data_pass)
data_pass_half = machining_process(data_pass_half)
data_fail = machining_process(data_fail)
```

[그림 13] 사용자 정의 함수 machining_process()의 사용

- 마지막으로, 각 변수에 대해 machining_process() 함수를 사용하여 마지막 행에 있는 독립변수 Machining_Process의 값을 변환한다. 문자데이터이기 때문에 학습 할 수 있도록 정수 데이터로 변환해준다.

③-3. 데이터셋 구성

```
# label 0/1 --> data01 / data02+data03
data01 = data_pass[0:3228+6175,:]
data02 = data_pass_half[0:6175,:]
data03 = data_fail[0:3228,:]
data = np.concatenate((data01,data02),axis=0);
data = np.concatenate((data,data03),axis=0);
data_all= data_pass[3228+6175:22645,:]

print((data))
print(data.shape)
print(data_all.shape)
```

```

[[202.04.04.0 ... 0508]
[202.0-6.8-346.0 ... 4500]
[200.0-13.8-2.25 ... 7500]
...
[155.09.875-64.6 ... 0508]
[155.59.95-52.125 ... 0508]
[156.010.176.125 ... 0508]]
(18806, 48)
(13242, 48)

```

[그림 14] 데이터 라벨링을 위한 사전 분류

- 3가지로 분류한 변수들을(data_pass, data_pass_half, data_fail) 심층 신경망 학습을 위한 양품과 불량품(pass/fail)의 데이터셋으로 재구성한다. 각각 22654개, 6175개, 3228개가 있고, data_pass_half와 data_fail을 불량품, data_pass를 양품으로 정의한다.

심층 신경망의 학습 및 성능을 더 효과적으로 하기 위해서는 라벨 간 균일한 데이터 샘플 수를 설정하고, 전처리 과정을 진행하는 것이 중요하다. 불량품은 총 9403개이며, 샘플 수의 균일성을 위해 양품 22654개 중에서 9403개를 뽑는다. 총 18806 개의 데이터를 심층 신경망 학습을 위한 데이터셋으로 재구성한다.

새롭게 정의한 변수 data01은 양품 22654개의 데이터 중 9403개를 뽑아 구성한 데이터이다. 또한, data02와 data03은 각각 6175개의 data_pass_half 데이터와 3228개의 data_fail 데이터이다. 하나의 data 변수로 합치기 위해 np.concatenate() 함수를 사용하여 차례대로 합친다. print() 함수를 사용하여 새롭게 구성한 data의 내용과 사이즈를 확인해보면 그림15와 같다.

결과창에서 볼 수 있듯이, data에는 각 샘플의 열 데이터가 이어 붙여진 형태로 저장되어 있고, 사이즈는 18806x48로 data01, data02, data03의 샘플 수를 합한 18806개를 나타내고 있다.

또한, 학습 완료 후에 사용할 평가 데이터셋으로 남은 data_pass의 샘플 데이터들을 data_all이라는 변수로 선언한다. 평가 데이터셋의 샘플 수는 결과창에 나타난 대로 13242개이며, 변수의 사이즈는 13242x48이다.

③-4. 데이터 정제(2차 전처리)

```

sc = MinMaxScaler()
X_train = sc.fit_transform(data)
X_train = np.array(X_train)
X_test = sc.fit_transform(data_all)
X_test = np.array(X_test)

```

[그림 15] 데이터 정제

- 새롭게 구성한 변수 data와 data_all에 대한 2차 전처리 과정을 진행한다.
2차 전처리는 각 열에 대해 사이킷런 package에서 제공하는 MinMaxScaler() 함수를 이용한다. 각 열의 데이터에 대해 최소값과 최대값을 구하고, 그 열의 데이터값 각각을 0에서 1 사이의 값으로 재조정한다.
sc라는 변수에 MinMaxScaler()를 선언한다. data 및 data_all을 입력값으로 하는 sc.fit_transform 함수를 이용하여 2차 전처리가 완료된 X_train과 X_test라는 변수를 얻는다. X_train이 우리가 심층 신경망 학습에 사용하게 될 2차 전처리 과정이 완료된 데이터셋이며, X_test는 학습 완료 후 평가에 사용할 데이터셋이다.

[단계④] 데이터 라벨링

앞선 단계에서 분류한 데이터를 바탕으로, 단계④에서는 해당 데이터에 대해 양품 및 불량품에 해당하는 라벨을 붙여준다. 데이터 라벨링 작업 예제를 제공한다.

④-1. 라벨 데이터 제작

```
# make label data
Y_train = np.zeros((len(X_train),1),dtype='int')
Y_test = np.zeros((len(X_test),1),dtype='int')
l = int(Y_train.shape[0]/2)
Y_train[0:l,:] = 0
Y_train[l:l*2,:] = 1
print(Y_train)
```

```
[0]
[0]
[0]
...
[1]
[1]
[1]]
```

[그림 16] 라벨 데이터 제작 및 확인

- 심층 신경망 학습을 위해서는 데이터셋 뿐만 아니라 라벨 데이터 역시 필요하다. 앞서 제작한 데이터셋 X_train과 X_test의 각 샘플에 대한 라벨 데이터를 제작한다. 그림 17을 보자. 먼저 라벨 데이터 변수 Y_train과 Y_test를 선언한다. np.zeros() 함수를 이용하여 numpy array 형태로 선언한다. np.zeros() 함수는 설정한 사이즈로 원소 값이 전부 0인 numpy array를 생성해준다. Y_train의 길이는 X_train의 샘플 개수와 같아야 하므로, 설정 사이즈는 (len(X_train),1)로 한다. Y_test 역시 마찬가지로 설정한다. 또한, Y_train의 원소값은 라벨을 의미하기 때문에 0 또는 1

로 선언해야 한다. X_train 데이터셋 재구성 시, 앞의 절반은 라벨값 0에 해당하는 양품을, 뒤의 절반은 라벨값 1에 해당하는 불량품으로 구성하였다. 마찬가지로 Y_train의 앞의 절반은 0으로, 뒤의 절반은 1로 선언하면 된다.

반면, Y_test는 평가 데이터셋 X_test가 전부 양품인 샘플로 이루어져 있으므로 라벨값으로 0을 채우면 된다.

print() 함수를 이용하여 제작한 라벨 데이터 변수 Y_train을 확인해보자. 의도한대로 라벨값 0으로 시작하여 라벨값 1로 끝나는 것을 볼 수 있다.

[단계⑤] 학습/검증/평가 데이터 분리

이전 단계들을 진행하면서 AI 모델 학습 및 테스트를 위해 데이터를 분류하고 라벨링 작업도 진행하였다. 단계⑤에서는 학습/검증/평가 데이터를 분리하는 방법을 예제로 제공한다.

⑤-1. 학습/검증/평가 데이터 구성

- 학습 시작 전, fit() 함수에서 validation_split 옵션을 설정을 통해 입력한 데이터셋을 설정한 비율로 학습 데이터셋과 검증 데이터셋으로 분리할 수 있다. 학습 데이터셋은 전체 학습 데이터셋 X_train의 90%를 사용하였고, 검증 데이터셋은 전체 데이터셋의 10%를 사용하였다. fit() 함수에서의 검증 데이터셋 비율 설정은 뒤에 이어지는 단계7의 7-1. AI 모델 훈련에서 해당 함수를 사용하기 때문에 설명으로만 대체하였다.

평가에는 평가 데이터셋 X_test를 사용한다. 각 데이터셋의 개수는 아래 표와 같다.

	전체 개수
학습 데이터셋	16925 개
검증 데이터셋	1881 개
평가 데이터셋	13242 개

[표 1] 항목별 데이터셋 개수

[단계⑥] AI 모델 구축

단계⑤ 까지 진행했다면, AI 모델 학습을 위해 파라미터를 설정하고 AI 모델을 디자인하는 마지막 단계가 남았다. 단계⑥에서는 이에 대한 예제를 제공한다.

⑥-1. AI 모델 파라미터 설정

```
nb_classes =2  
batch_size =1024  
epochs =300  
lr = 1e-4
```

[그림 17] AI 학습 모델의 파라미터 설정

- AI 모델 학습 시작 전 사용자가 설정하는 파라미터는 다음과 같다 : 학습 데이터셋 라벨 종류의 개수, 학습 데이터셋 배치 사이즈, 학습 반복 횟수(에폭수), 러닝레이트. 본 예제에서 제작한 데이터셋은 양품/불량품 2가지의 라벨로 이루어져 있다. 따라서 라벨 종류의 개수로 nb_classes = 2 로 선언한다.
- 배치 사이즈는 데이터셋 전체를 몇 개로 쪼개서 학습을 진행할지 결정하는 파라미터이다. 예를 들어 총 데이터가 100개, 배치 사이즈가 10이면, 한번의 학습을 10개의 데이터에 대해 진행한다는 의미이다.
- 학습 반복 횟수(에폭수)는 모든 데이터셋을 한 번 학습하는 횟수를 나타내는 파라미터이다. 설정한 수만큼 학습 횟수를 진행한다.
- 러닝레이트는 AI 모델이 성능을 향상시키기 위하여 학습할 때 완급을 조절하는 파라미터이다. 러닝레이트의 값이 너무 크면 범위를 벗어나 학습이 진행되지 않고, 너무 작으면 학습시간이 크게 증가하거나 목표 성능에 도달하지 못하게 된다. 일반적인 학습에서 0.001~0.0001의 값으로 설정한다.
- 예제에서 볼 수 있듯이, 본 가이드북에서는 배치사이즈는 1024, 에폭수는 300, 러닝레이트는 0.0001로 설정하여 진행하였다. 표 2에 AI 모델 파라미터 설정값이 나타나 있다.

설정값	
라벨 개수 (nb_classes)	2
배치 사이즈 (batch_size)	1024
학습 반복 횟수 (epochs)	300
러닝레이트 (lr)	1e-4 (0.0001)

[표 2] 학습 조건별 설정값

⑥-2. AI 데이터셋 준비

```
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
Y_train = np_utils.to_categorical(Y_train, nb_classes)
Y_test = np_utils.to_categorical(Y_test, nb_classes)
print(X_train.shape)
print(X_test.shape)
print(Y_train.shape)
print(Y_test.shape)
```

```
(18806, 48)
(13242, 48)
(18806, 2)
(13242, 2)
```

[그림 18] AI 모델 학습을 위한 데이터셋 준비

- 2차 가공까지 완료한 데이터셋 X_train, X_test와 라벨 데이터 Y_train, Y_test를 준비한다.

라벨 데이터의 경우, 원-핫 인코딩 기법을 이용하여 0과 1로 된 데이터를 변환하여야 한다. 원-핫 인코딩(one-hot encoding)이란 라벨 데이터를 벡터로 표현하고, 해당 라벨을 표현할 인덱스에 1을 표시하고 나머지 인덱스에는 0을 표시하는 방법이다. 이렇게 표현된 벡터를 원-핫 벡터(one-hot vector)라고 한다. 원-핫 벡터의 길이는 라벨의 개수로 결정된다. 예를 들어, 예제의 라벨의 개수는 2개이다. 따라서 원-핫 벡터의 길이는 그림 18에서 볼 수 있듯이 (18806, 2)와 (13242, 2)로, (샘플 수 x 라벨수) 가 된다.

⑥-3. AI 모델 디자인

```
model = Sequential()
model.add(Dense(128, activation ='relu', input_dim =48))
model.add(Dropout(0.3))
model.add(Dense(256, activation ='relu'))
model.add(Dropout(0.3))
model.add(Dense(512, activation ='relu'))
model.add(Dropout(0.3))
model.add(Dense(512, activation ='relu'))
model.add(Dropout(0.3))
model.add(Dense(256, activation ='relu'))
model.add(Dropout(0.3))
model.add(Dense(128, activation ='relu'))
model.add(Dropout(0.3))
model.add(Dense(nb_classes, activation='sigmoid'))
model_checkpoint = ModelCheckpoint('weight_CNC_binary.mat', monitor ='val_acc', save_best_only =True)
opt=Adam(lr)
model.summary()
model.compile(optimizer=opt,loss ='binary_crossentropy',
              metrics=['accuracy'])
history = History()
print('.....model is defined.....')
```

[그림 19] AI 학습 모델 디자인

- 준비된 데이터를 학습 할 AI 모델을 디자인한다. 본 예제에서는 AI 모델을 Sequential 형태로 디자인하였다. 뜻 그대로 레이어 들이 연속해서 이어진 형태이다. model.add() 함수를 이용하여 레이어를 차례대로 쌓는다. 레이어의 종류는 함수의 옵션으로 입력하며, 텐스 레이어와 드랍아웃 레이어를 사용하였다. 완전연결층인 텐스 레이어 사이에 드랍아웃 레이어를 추가하였다.

활성화 함수는 마지막 텐스 레이어에서만 시그모이드 함수를 사용하였고, 다른 텐스 레이어에서는 정류 선형 유닛 함수를 사용하였다.

모델의 학습 진행 상황을 판단하는 손실함수는 이진 교차엔트로피를 사용했으며, 신경망의 모수를 최적화하는 역할을 하는 최적화 알고리즘(optimizer)은 Adam을 사용하였다.

모델이 전체 데이터에 대해 학습하는 매 에폭마다 각 시점의 모델의 학습 데이터에 대한 정확도(acc), 손실도(loss), 그리고 검증 데이터에 대한 정확도(val_acc), 손실

도(val_loss)가 출력되도록 하였다.

ModelCheckpoint() 함수는 AI 모델이 학습을 진행하는 동안에 매 에폭 종료시 검증 데이터에 대한 정확도(val_acc)를 확인하고, 가장 좋은 값을 기록하였을 때의 모델 파라미터(weight)를 저장한다. 함수 이름대로 모델 파라미터에 대해 체크포인트를 설정한다고 하면 이해하기 쉽다. AI 모델이 의도대로 구축되었는지 summary() 함수를 통해 확인할 수 있다.

또한, History() 함수를 사용하면 모델의 학습과정 동안의 정확도, 손실도 등을 기록할 수 있다.

[단계⑦] AI 모델 훈련

이번 단계에서는 AI 모델의 훈련 및 진행 방법에 대해 예제를 제공한다.

⑦-1. AI 모델 훈련

```
model.fit(X_train, Y_train, verbose=2, batch_size =batch_size, epochs =epochs, validation_split =0.1
shuffle =True, callbacks =[history])
model.save_weights('weight_CNC_binary.mat')
```

```
Train on 16925 samples, validate on 1881 samples
Epoch 1/300
 - 5s - loss: 0.6846 - acc: 0.5471 - val_loss: 0.7834 - val_acc: 0.0000e+00
Epoch 2/300
 - 0s - loss: 0.6740 - acc: 0.5560 - val_loss: 0.7761 - val_acc: 0.1058
Epoch 3/300
 - 0s - loss: 0.6610 - acc: 0.5658 - val_loss: 0.7682 - val_acc: 0.5303
Epoch 4/300
 - 0s - loss: 0.6448 - acc: 0.6080 - val_loss: 0.7267 - val_acc: 0.6909
Epoch 5/300
 - 0s - loss: 0.6237 - acc: 0.6551 - val_loss: 0.6568 - val_acc: 0.7759
Epoch 6/300
 - 0s - loss: 0.5958 - acc: 0.6781 - val_loss: 0.5609 - val_acc: 0.8963
Epoch 7/300
 - 0s - loss: 0.5663 - acc: 0.6990 - val_loss: 0.5015 - val_acc: 0.9599
Epoch 8/300
```

[그림 20] AI 모델 훈련 및 진행

- AI 모델 구축이 완료 되었다면 fit() 함수를 통해 학습을 시작할 수 있다. fit() 함수의 shuffle 옵션을 이용하면 손쉽게 학습 데이터셋을 매 에폭마다 무작위로 섞어서 사용할 수 있게 된다.

마지막으로, save_weights() 함수는 학습이 끝난 AI 모델의 파라미터값을 미리 정한 파일명과 확장자로 저장한다. 그림 20은 AI 모델의 훈련 진행 상황을 보여주는 스크린샷 결과이다.

[단계8] 결과 분석 및 해석

마지막으로, AI 모델의 훈련이 완료된 후, 훈련동안의 학습 및 평가 데이터셋에 대한 정확도와 손실값을 확인해보는 예제를 진행한다.

⑧-1. AI 모델 훈련 결과 출력

```
loss_and_metrics = model.evaluate(X_train,Y_train,batch_size=32)
print(loss_and_metrics)

loss_and_metrics2 = model.evaluate(X_test,Y_test,batch_size=32)
print(loss_and_metrics2)
```

```
18806/18806 [=====] - 2s 95us/step
[0.23372000538212398, 0.9719504413485058]
13242/13242 [=====] - 1s 100us/step
[0.8747636941664975, 0.9150808035040024]
```

[그림 21] AI 훈련 모델 테스트 결과

데이터셋 종류	양품/불량품 분류 정확도
검증 데이터셋	99.60
전체 데이터셋	97.18%
평가 데이터셋	91.51%

[표 4] AI 모델 훈련 및 테스트 결과

- 각 데이터셋에 대한 학습이 완료된 AI 모델의 성능을 확인한다. `model.evaluate()` 함수를 이용하여 성능을 확인할 수 있다. 학습 데이터셋과 평가 데이터셋에 대하여 성능을 확인해보자.

그림 23의 결과를 보면 각각 2개의 값을 출력하는 것을 확인할 수 있다. 첫 번째 값은 손실도값이며, 두 번째 값은 정확도값이다.

전체 데이터셋 `X_train`에 대한 성능은 0.9719로 백분율로 환산하면 97.19%이다.

평가 데이터셋 `X_test`에 대한 성능은 0.9151로 백분율로 환산하면 91.51%이다.

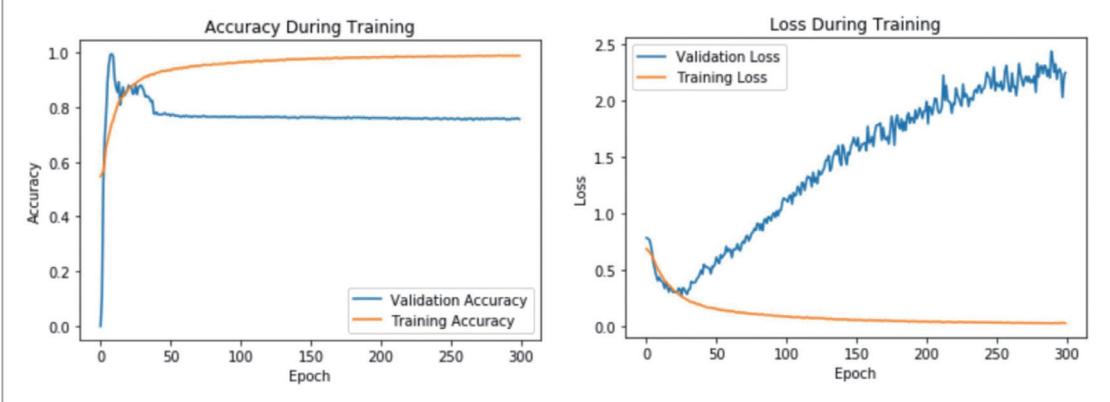
검증 데이터셋에 대한 성능은 학습 중 검증정확도 최고값인 99.60%이다. 결과를 정리하면 표 4와 같다.

```
plt.plot(history.history['val_acc'])
plt.plot(history.history['acc'])
plt.title('Accuracy During Training')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(['Validation Accuracy','Training Accuracy'])
```

```

plt.plot(history.history['val_loss'])
plt.plot(history.history['loss'])
plt.title('Loss During Training')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(['Validation Loss','Training Loss'])
plt.show()

```



[그림 22] AI 모델 훈련 동안의 정확도/손실값 변화

- AI 모델 훈련동안의 정확도와 손실값의 변화 그래프는 그림 23와 같다. 비교적 이른 에폭수에서 AI 모델 훈련이 수렴하였음을 확인할 수 있다.
해당 예제 실행 시, 설치한 keras와 tensorflow의 버전에 따라 정확도의 변수명이 다를 수 있다. ‘val_acc’ 및 ‘acc’에서 오류 발생 시, 각각 val_accuracy 와 accuracy 로 바꿔서 실행하도록 한다.

⑧-2. AI 모델 훈련 결과 분석 및 해석

- CNC 가공 설비를 통하여 취득한 데이터에서 적절한 전처리 가공을 진행하고, 지도 학습 방식의 AI 모델을 설계하고 학습하여 양품과 불량품 분류를 해낼 수 있다. 설정한 라벨링 기준에 따르면, 학습된 모델은 공정 완료 유무에 따른 양품과 불량품의 분류 및 예측 가능하였다. 하지만 육안검사 통과 유무에 따른 양품과 불량품의 분류 케이스는 앞선 분류 기준의 성능과 비교했을 때, 향상된 성능을 필요로 한다고 판단된다. 이를 위해서는 해당 분류 기준의 성능 향상을 위한 데이터셋이 추가로 필요로 하다고 판단된다.
또한, 평가 데이터셋의 양품 및 불량품 라벨 데이터 결과와 실제 라벨 데이터 결과 비교를 통해 공구수명에 따른 가공불량을 예측할 수 있을 것으로 판단된다.

3. 유사 태현장의 「CNC 머신 AI 데이터셋」 분석 적용

3.1 본 분석이 적용 가능한 제조현장 소개

- CNC로 제어 할 수 있는 도구로는 선반, 밀링 머신, 라우터 및 그라인더가 있으며 이 공정을 사용하는 제조현장에는 본 분석이 적용 가능하다. 또한 본 분석이 적용된 CNC 가공의 신뢰성으로 인해 더욱 복잡한 형상을 생성할 수 있으며 3차원 형상을 정확히 생성하는 데 도움이 될 것으로 생각된다. 또한 높은 정밀도 또는 반복적인 작업이 필요한 작업에 적용을 한다면 좋은 성과가 기대된다.

3.2 본 「CNC 머신 AI 데이터셋」 분석을 원용하여 타 제조현장 적용 시, 주요 고려사항

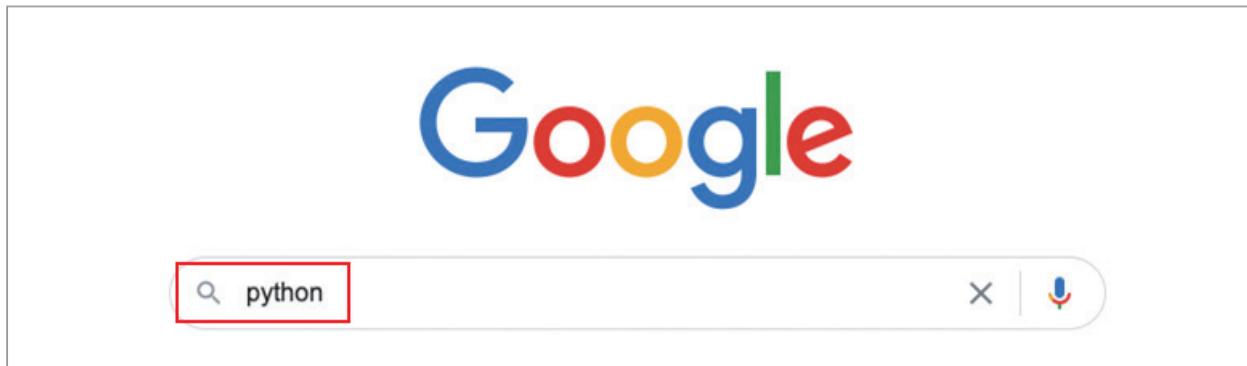
- 컴퓨터 수치 제어(CNC) 방식의 제조 공정에서 나오는 데이터를 이용하여 정상과 불량품을 구별하는 불량검출 알고리즘에 다음과 같은 애로사항 및 개선점이 존재한다.
 - a. 불량검출 실패사례 발생 및 검출할 수 있는 종류가 제한적이다.
 - b. 여러 공정에 통용할 수 있는 자동 불량 검출 알고리즘의 부재
- 따라서 공구의 마모도 측정 및 파손 예측 모델 개발 필요하고 데이터 분석을 위한 전처리 방법 모색 및 정상과 불량 생산품을 명확하게 구별하는 데이터의 수학적 특징 정의 및 분류 기준 설정이 목표가 되어야 한다.



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.

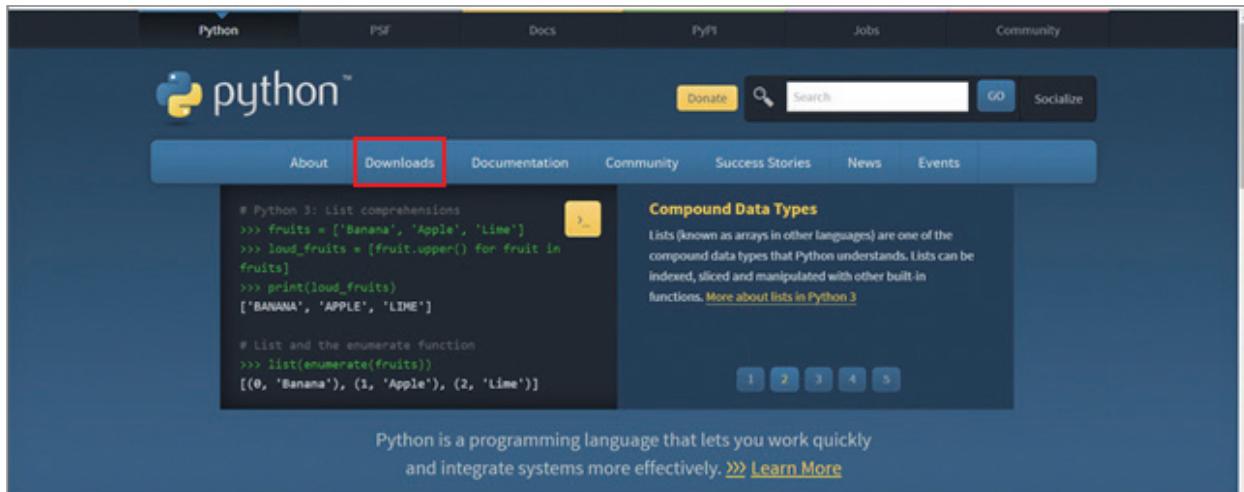
① google.com 등의 검색 엔진에 'python'을 검색



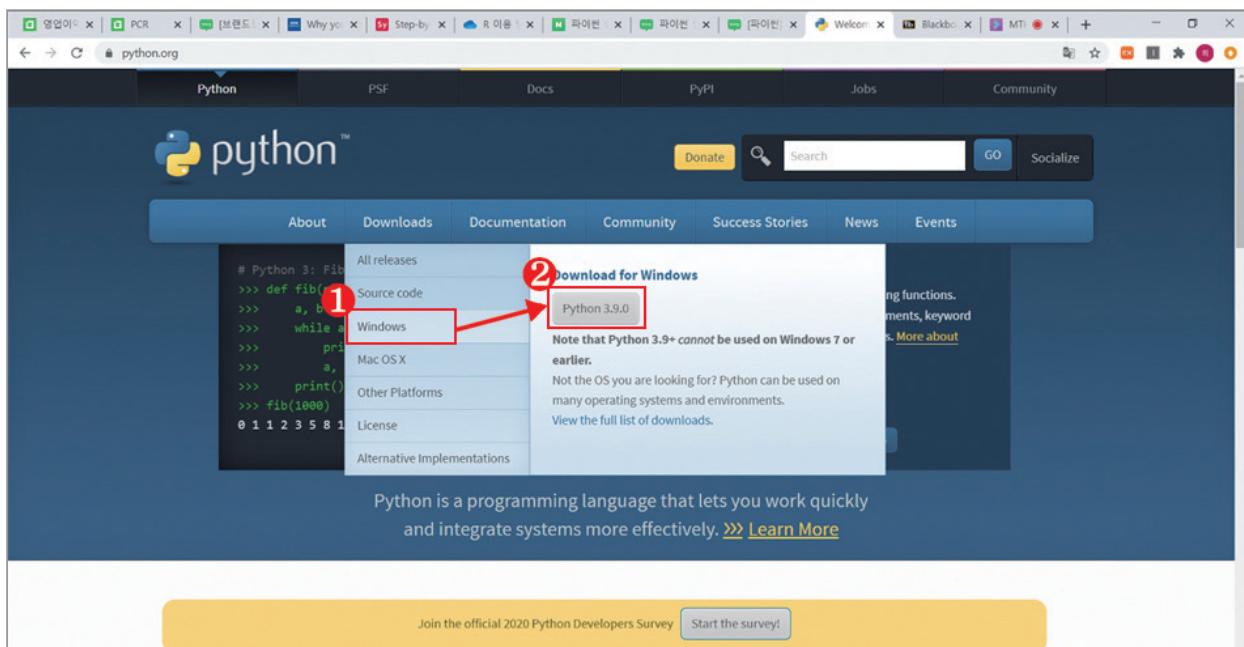
② 제일 처음에 보이는 'Welcome to python.org'를 클릭

The screenshot shows the Python.org homepage. The search bar at the top contains the word "python". Below the search bar, there are navigation links for All, Images, News, Videos, Books, and More. A link to "www.python.org" is shown, followed by a large button labeled "Welcome to Python.org" which is highlighted with a red box. Below this button, the text "The official home of the Python Programming Language." is visible. On the left side, there are sections for "Downloads" (Windows - Python 3.9.0 - Python 3.8.6 - Mac OS X - Python 3.7.9), "Documentation" (Python's documentation, tutorials, and guides are constantly ...), and "Python For Beginners" (BeginnersGuide/Download - Python for Programmers - Books). On the right side, there are sections for "Python 3.9.0" (Python 3.9.0. Release Date: Oct. 5, 2020. This is the stable release ...), "Tutorial" (1. Whetting Your Appetite - Lambda - 5. Data Structures), and "Python Docs" (What's new in Python 3.9? or all "What's new" documents since 2 ...).

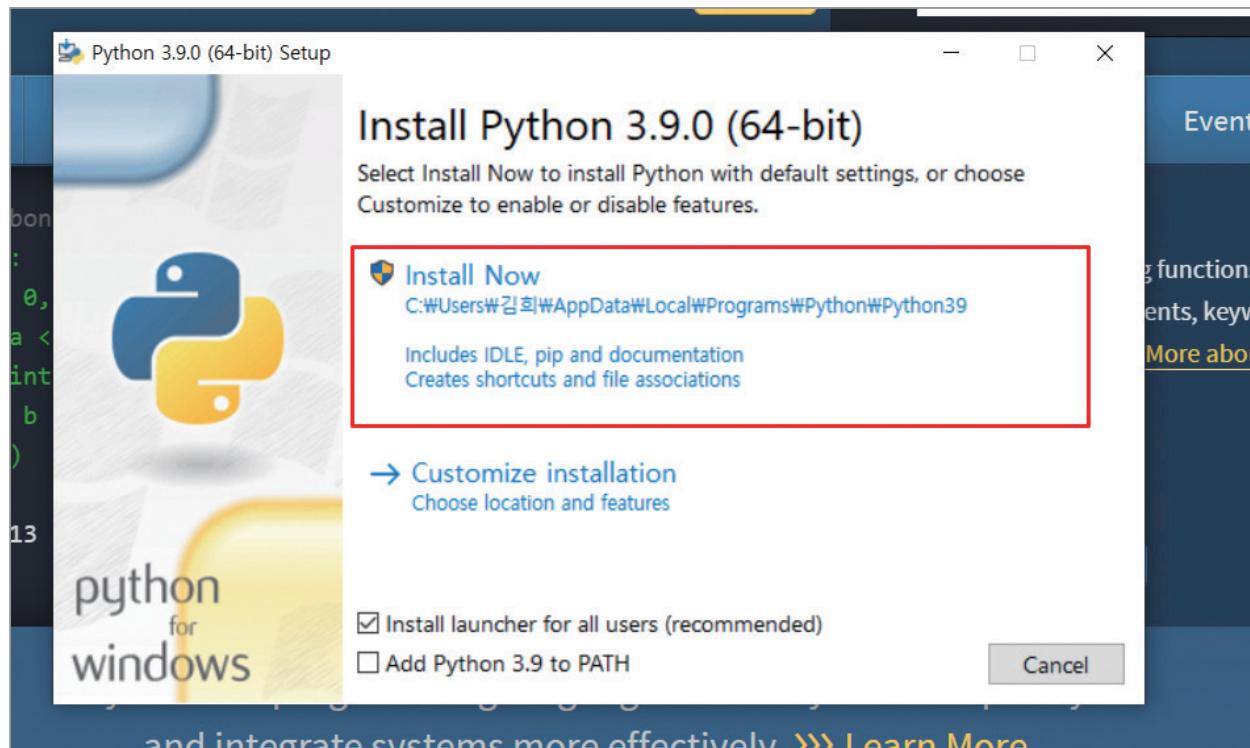
③ 클릭해서 보이는 페이지 정면의, 왼쪽 2번째 'Downloads' 클릭



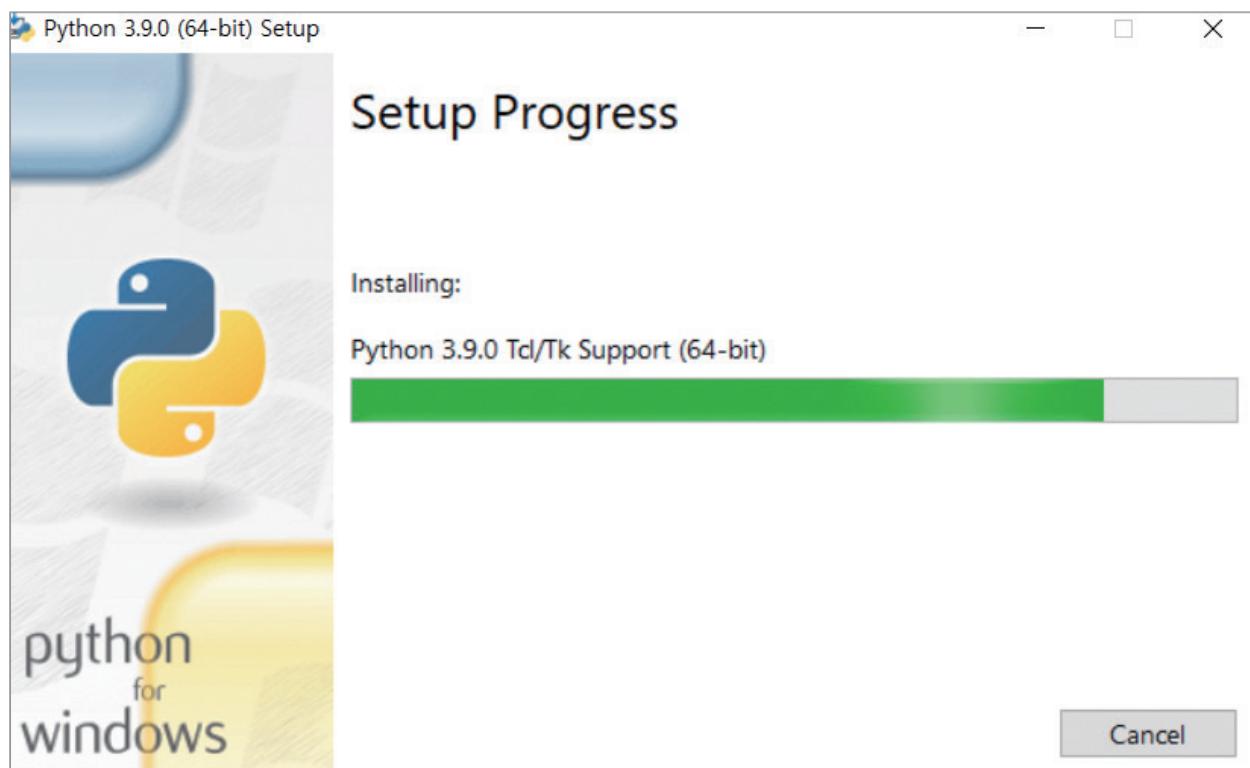
④ 위에서 3번째, Windows 탭을 선택한 후, python3.9.0 다운로드
(python3.9.0은 숫자가 업데이트 될 수 있습니다)



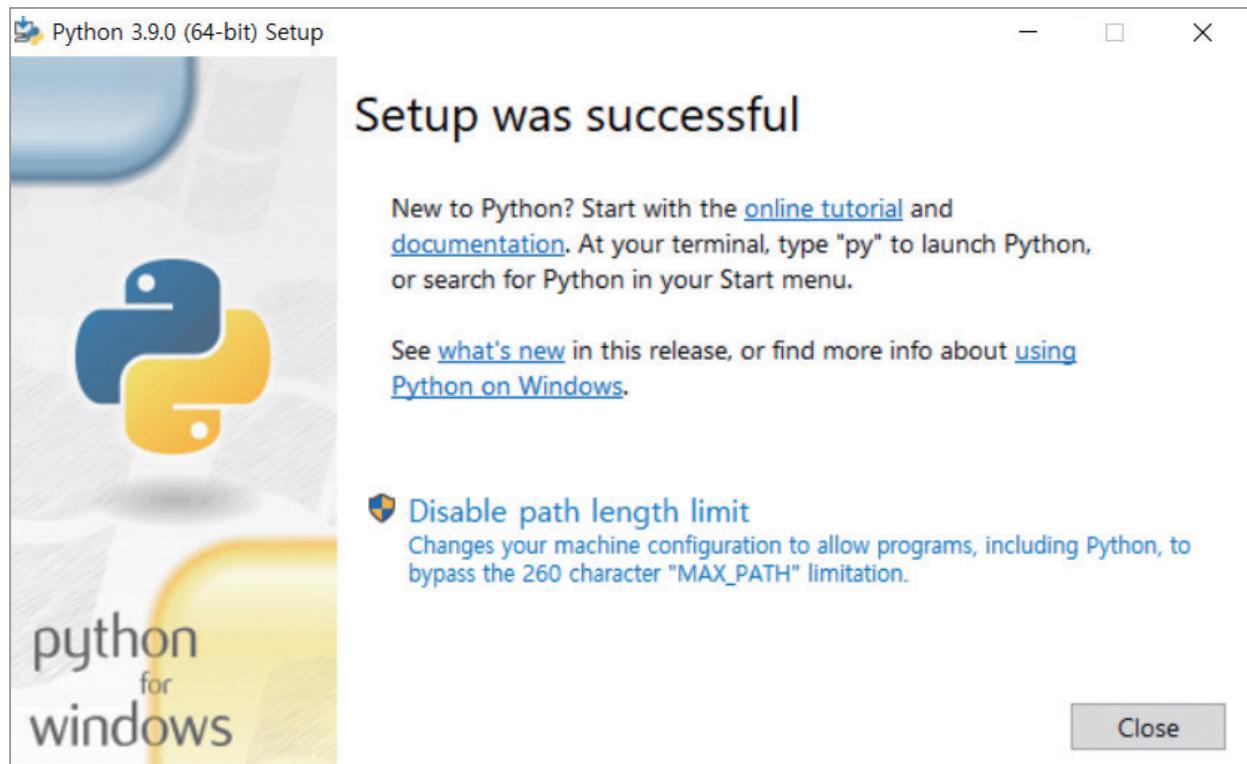
⑤ 아래와 같은 설치창이 뜨면, 'Install Now'를 클릭



⑥ 아래와 같은 설치 진행창이 완료가 될 때까지 유지



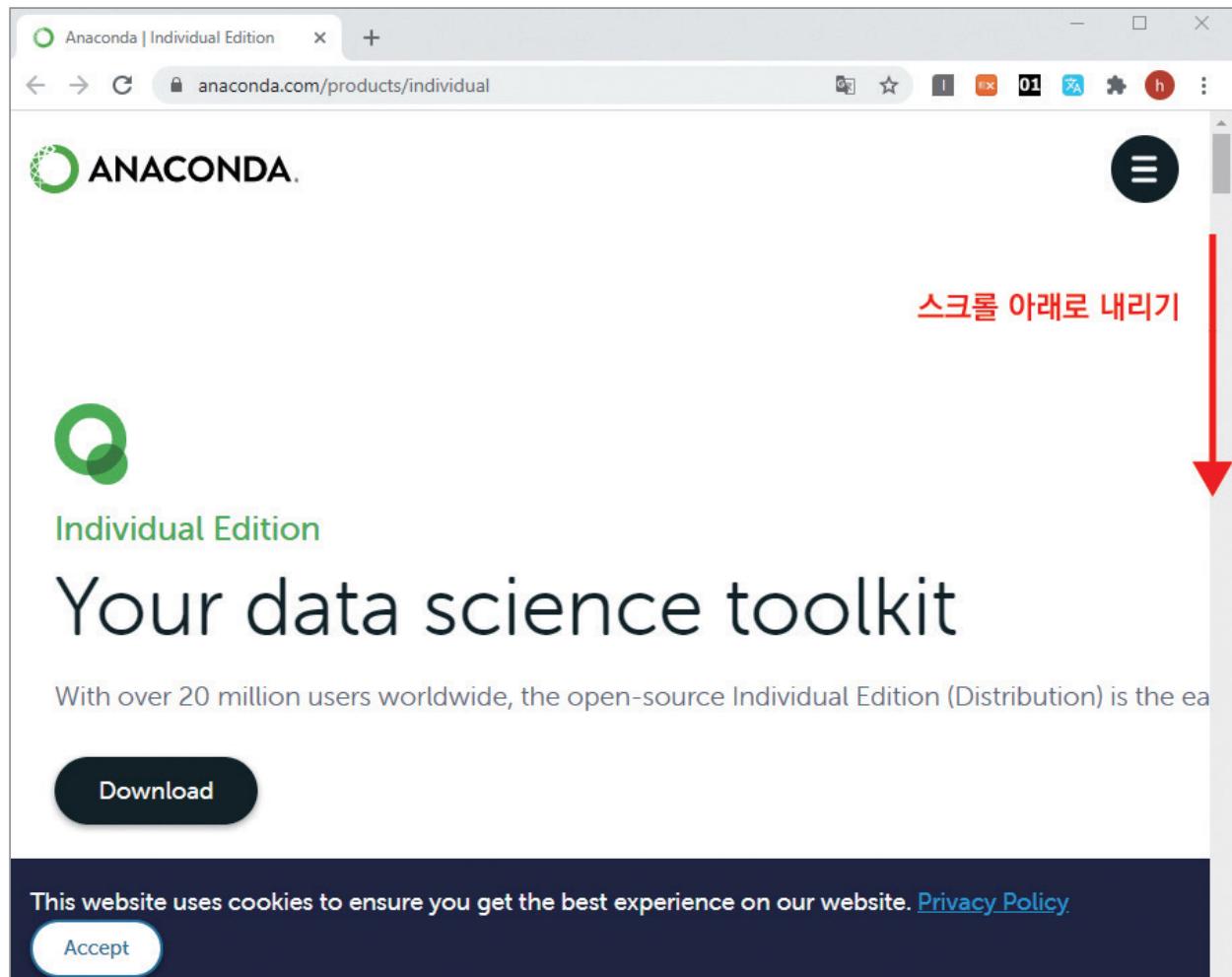
⑦ 완료가 되면 아래와 같은 창이 뜨는 것 확인 후 종료 [설치완료]



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, 결과물을 또한 쉽게 볼 수 있는 기능을 지원합니다. 아나콘다를 설치하고 분석을 할 수 있는 환경을 만들어봅니다.

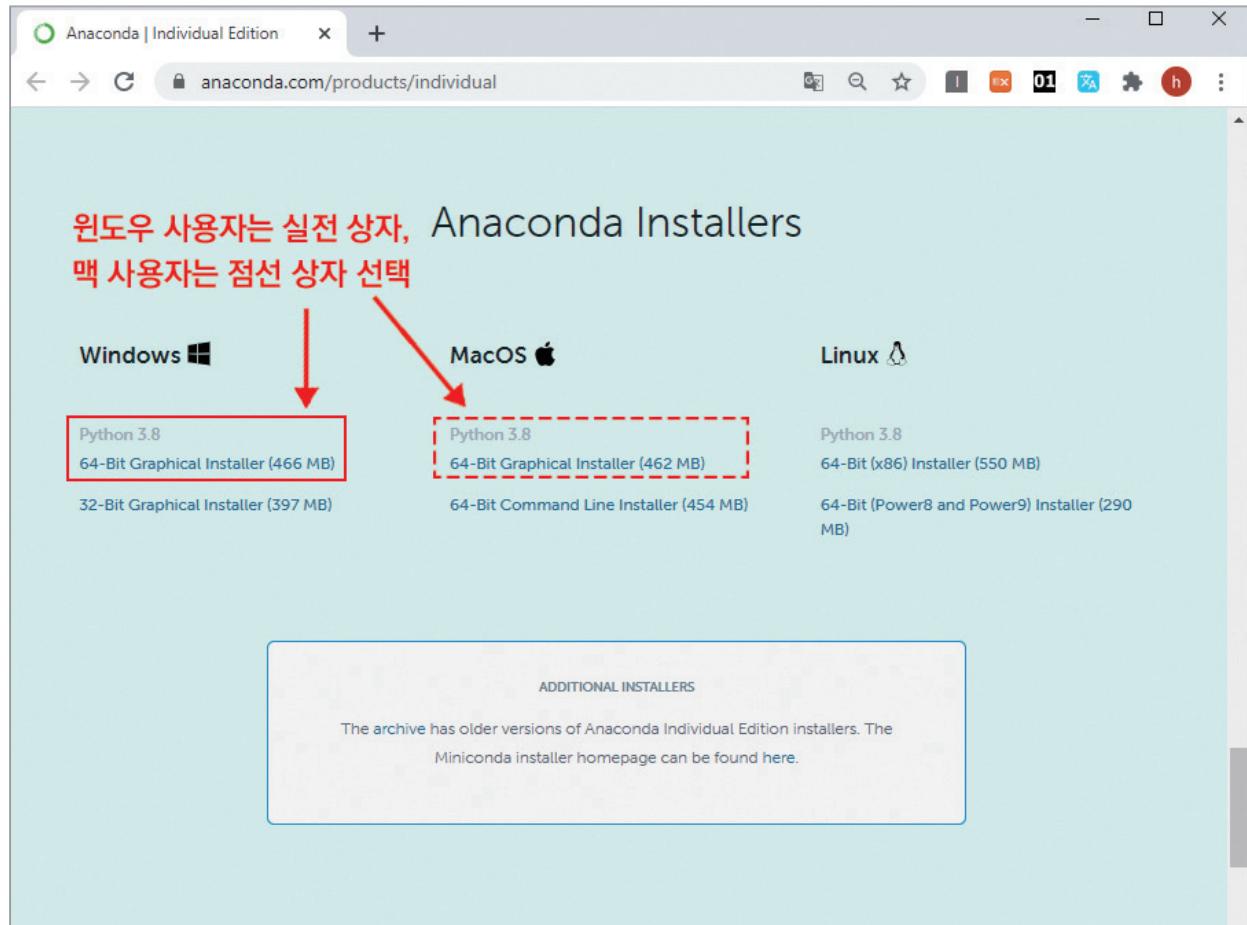
- ① <https://www.anaconda.com/distribution/> 로 접속 후 스크롤 내림



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

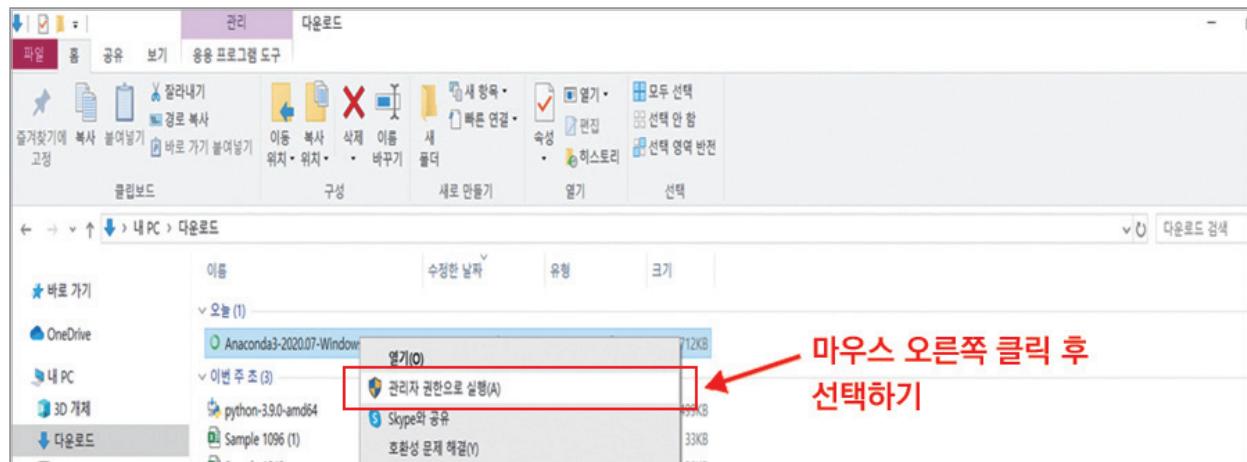
▶ Windows : 64-Bit Graphical Installer

▶ MacOS : 64-Bit Graphical Installer

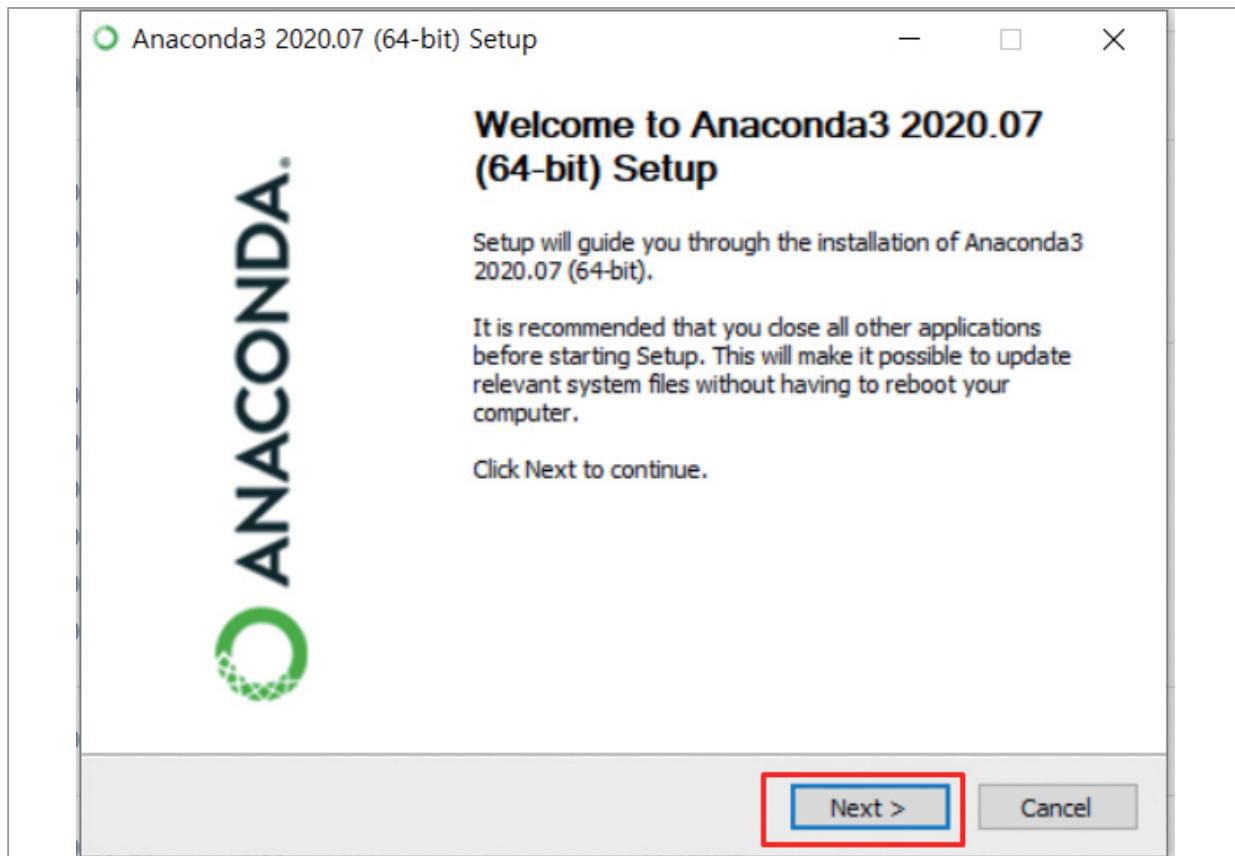


③ 다운을 받은 파일에 가서, 아나콘다 설치 파일 위에서, 마우스 오른쪽을 클릭한 후, 방패모양의 ‘관리자 권한으로 실행’ 선택

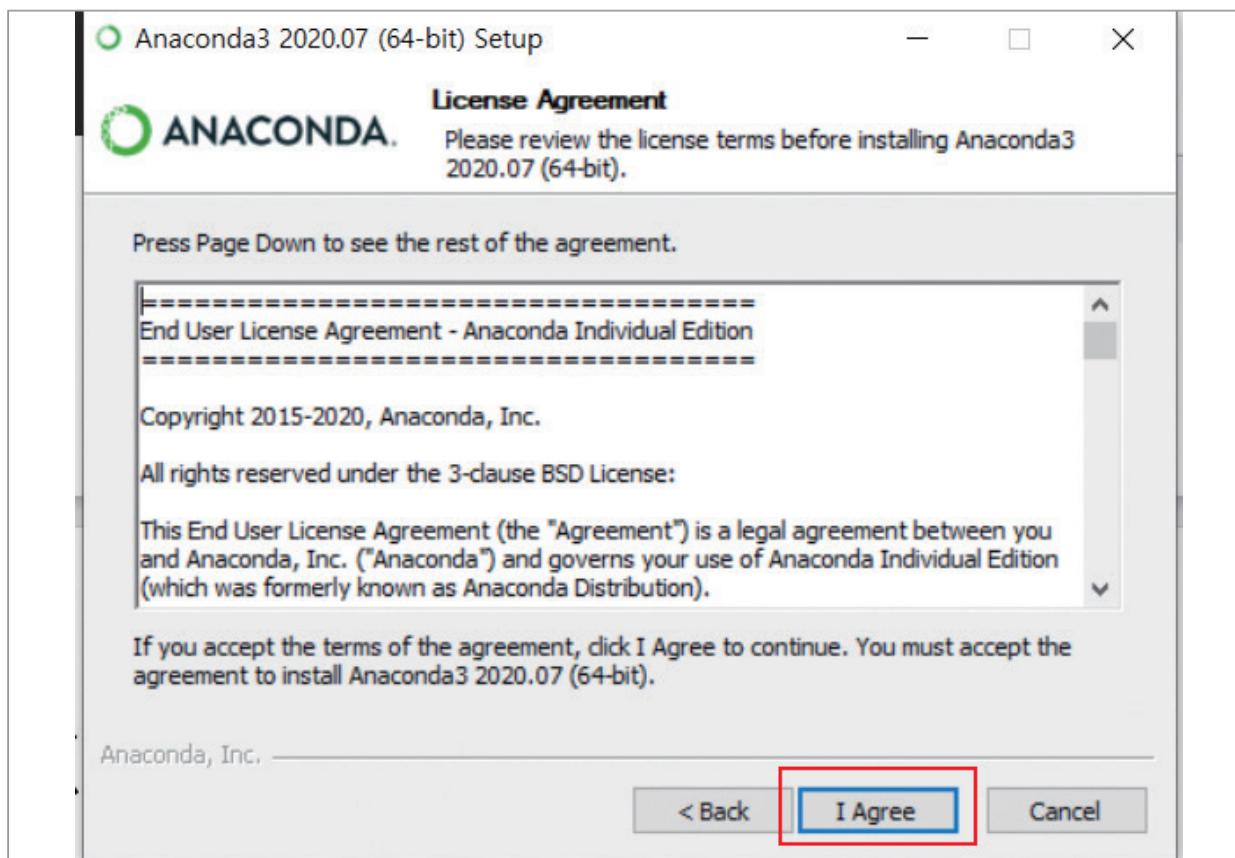
▶ (예) ‘다운로드’ 파일로 아나콘다를 다운 받은 경우



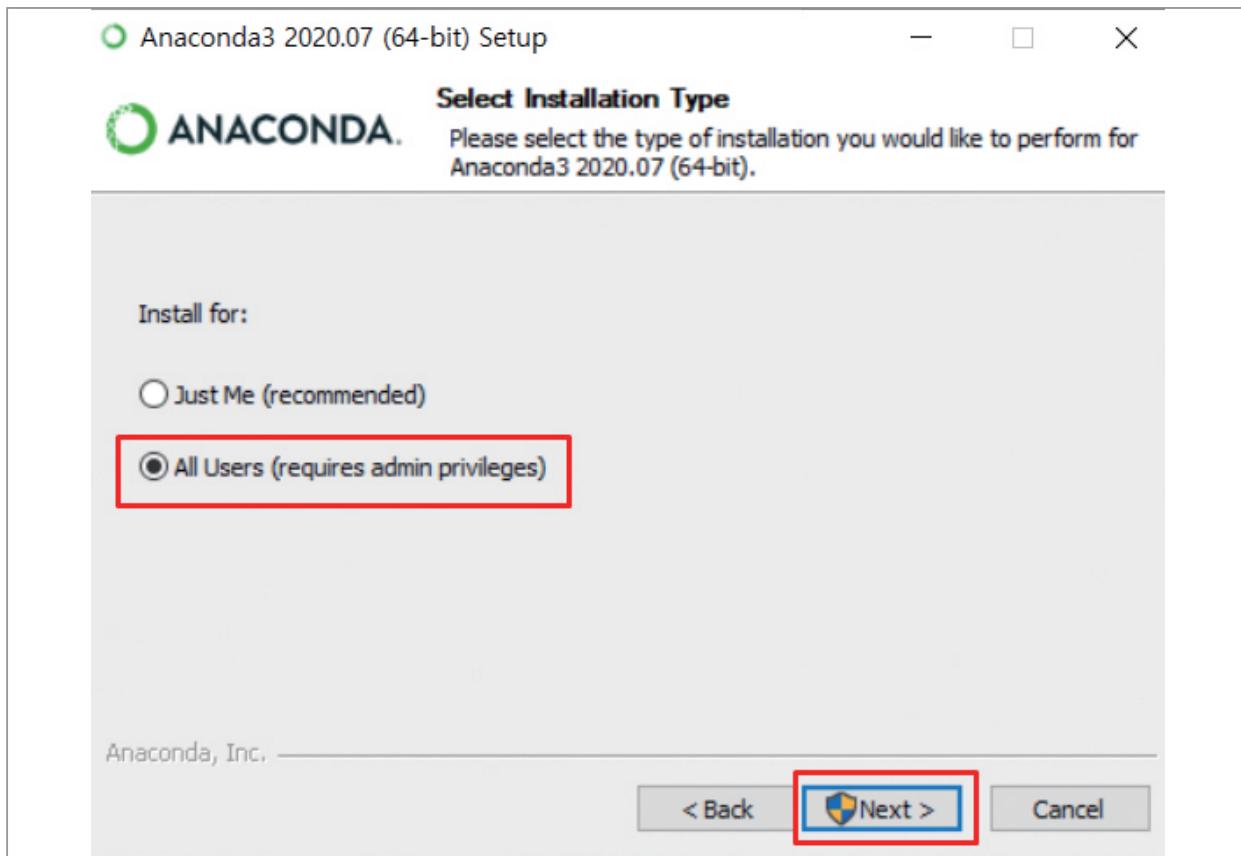
④ 파일을 실행 한 후, 'Next' 버튼을 클릭



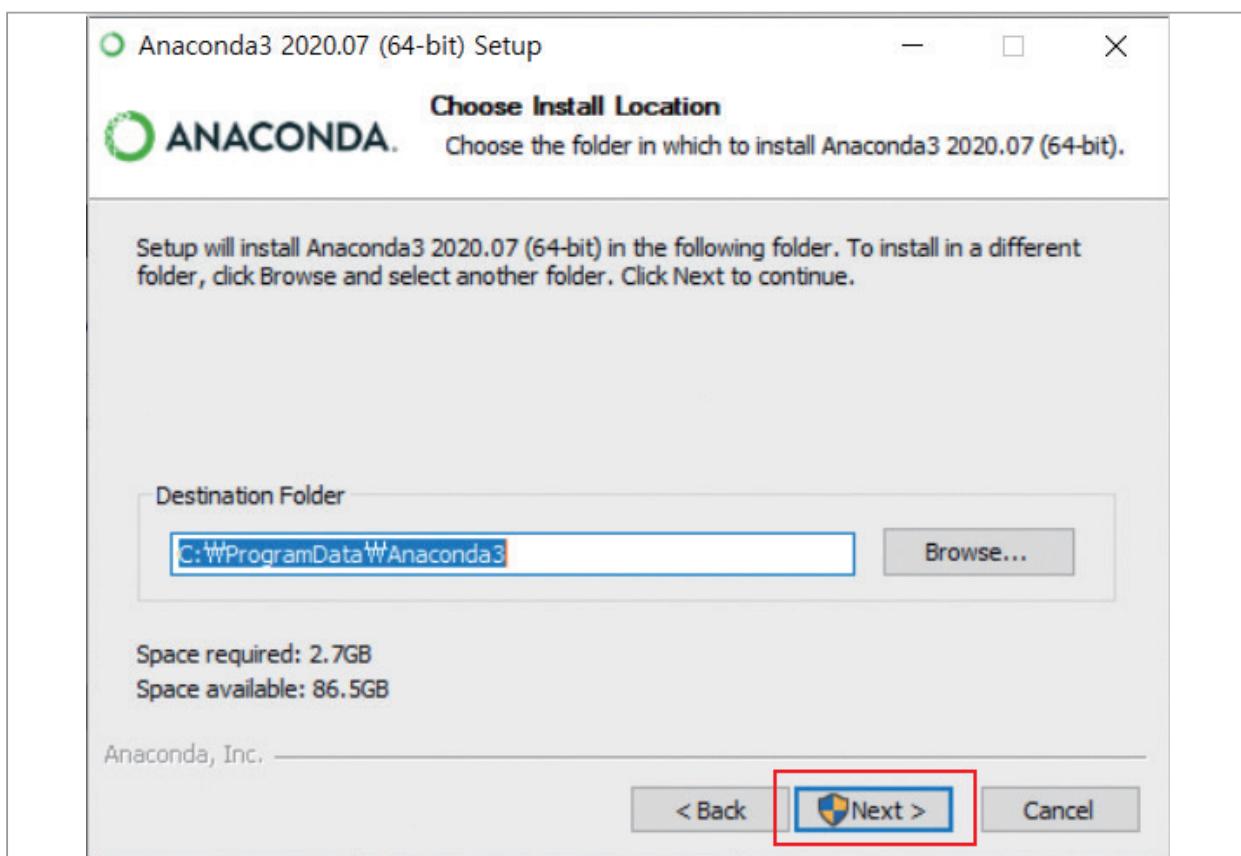
⑤ 다음 창이 나타나면 'I agree'를 선택



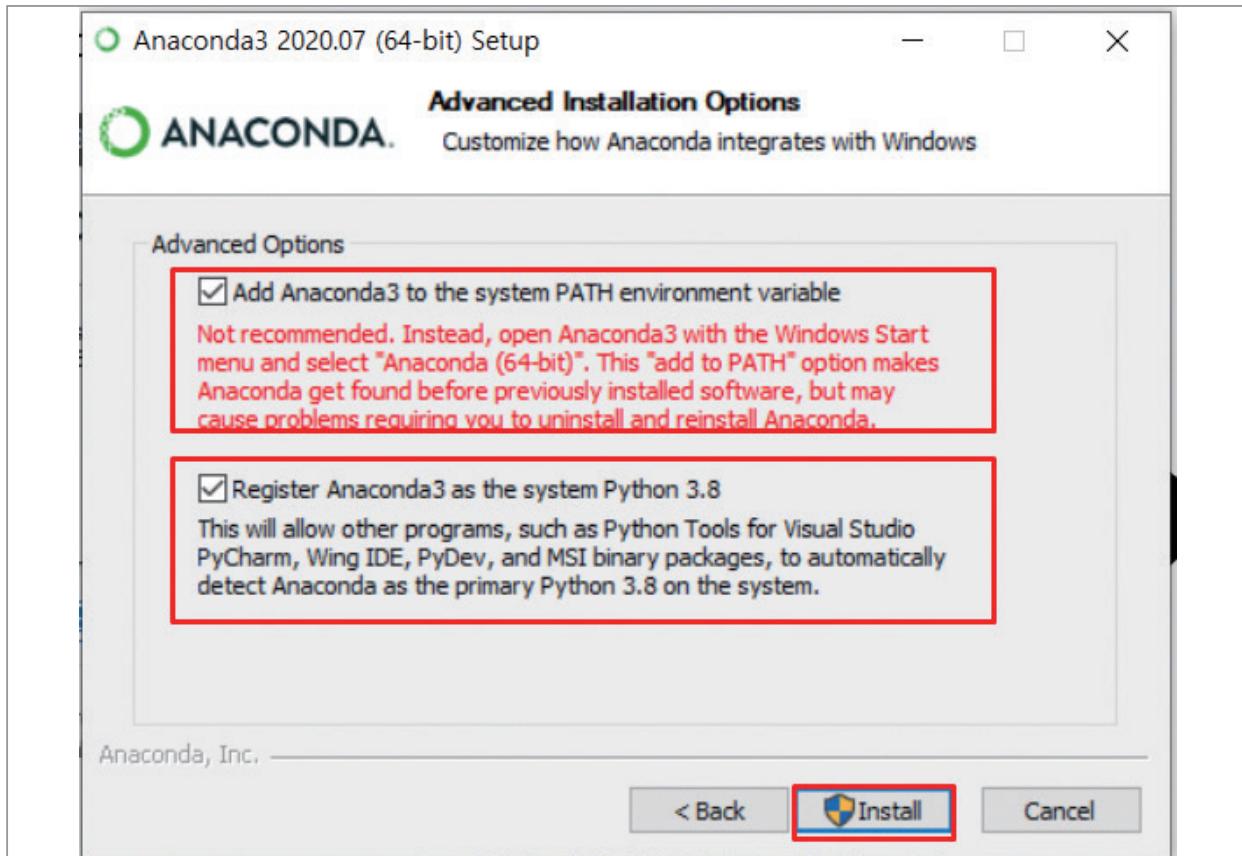
⑥ 셋팅 창이 뜨면 화면의 'All Users'를 선택 후 아래의 'Next' 클릭



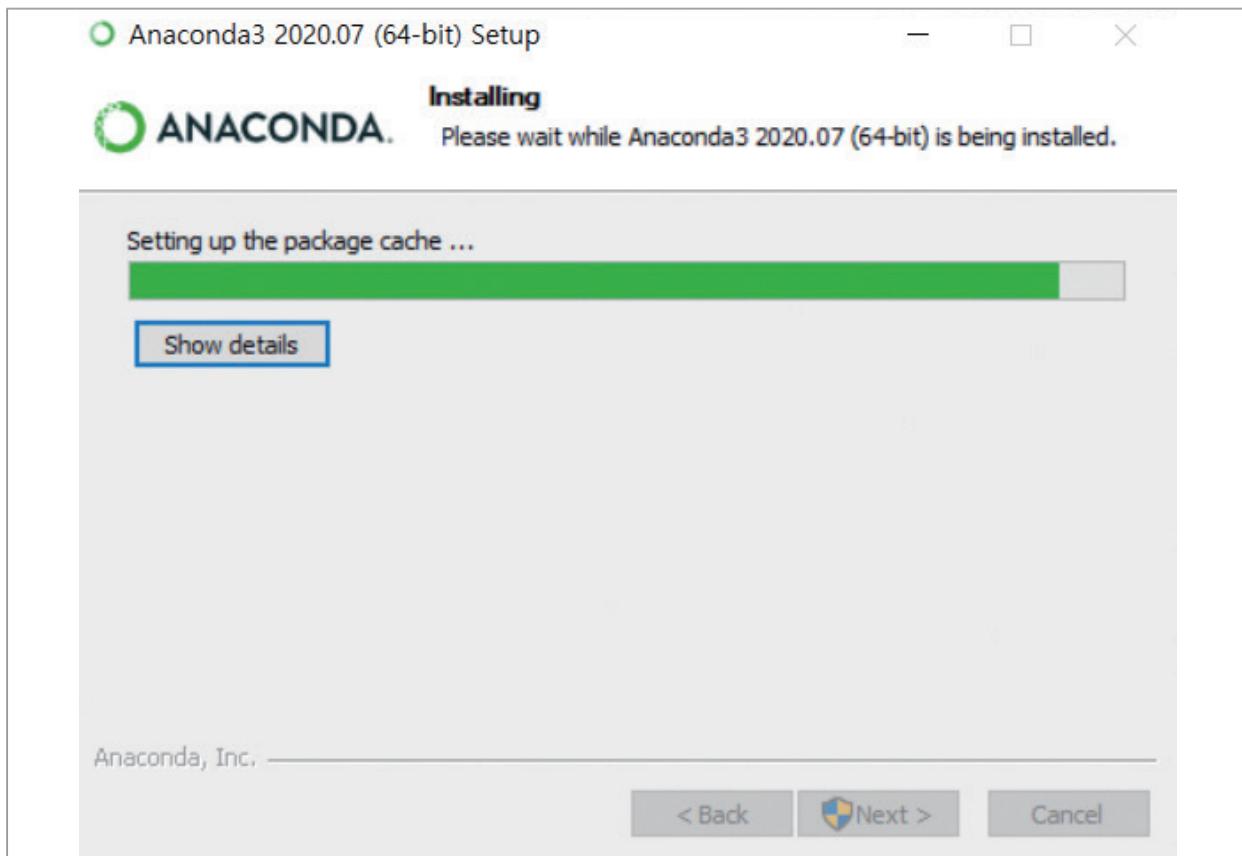
⑦ 다운로드 받을 경로를 물어보는 창이 뜨면, 아래의 'Next' 클릭



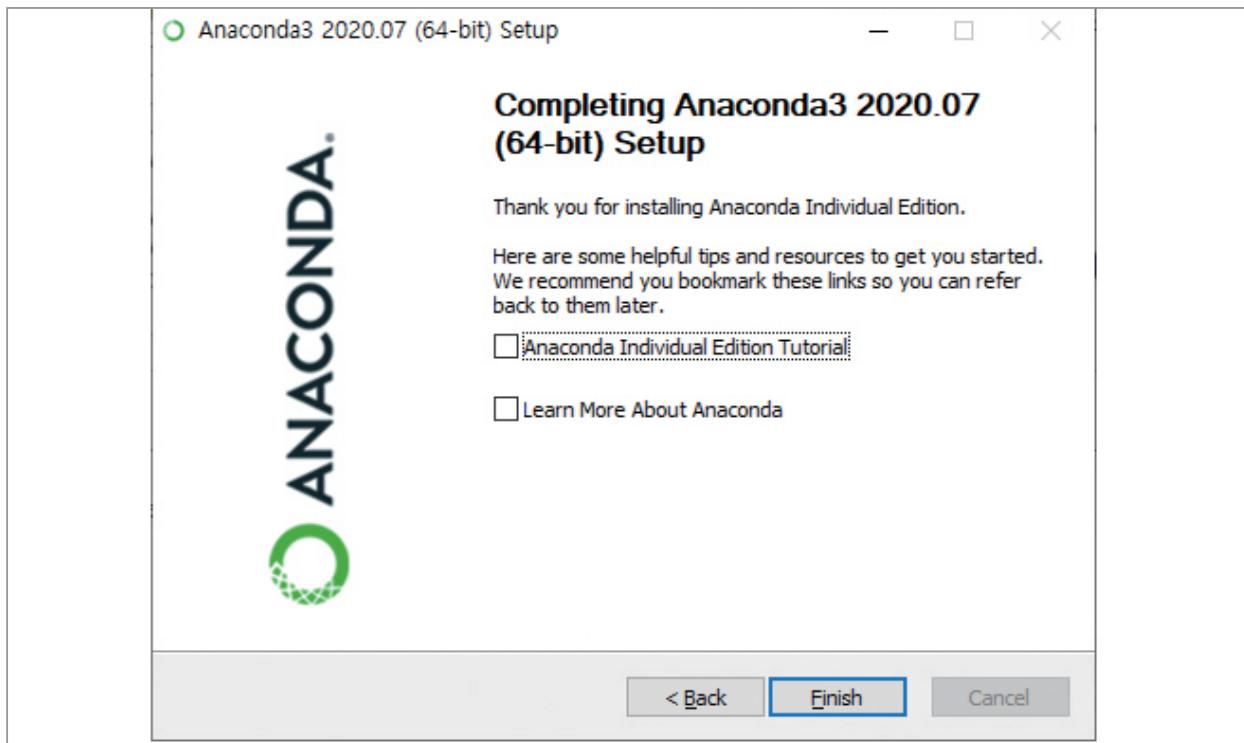
⑧ 고급 옵션 선택창이 뜨면, 아래와 같이 모두 선택 후, 'Install' 클릭



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

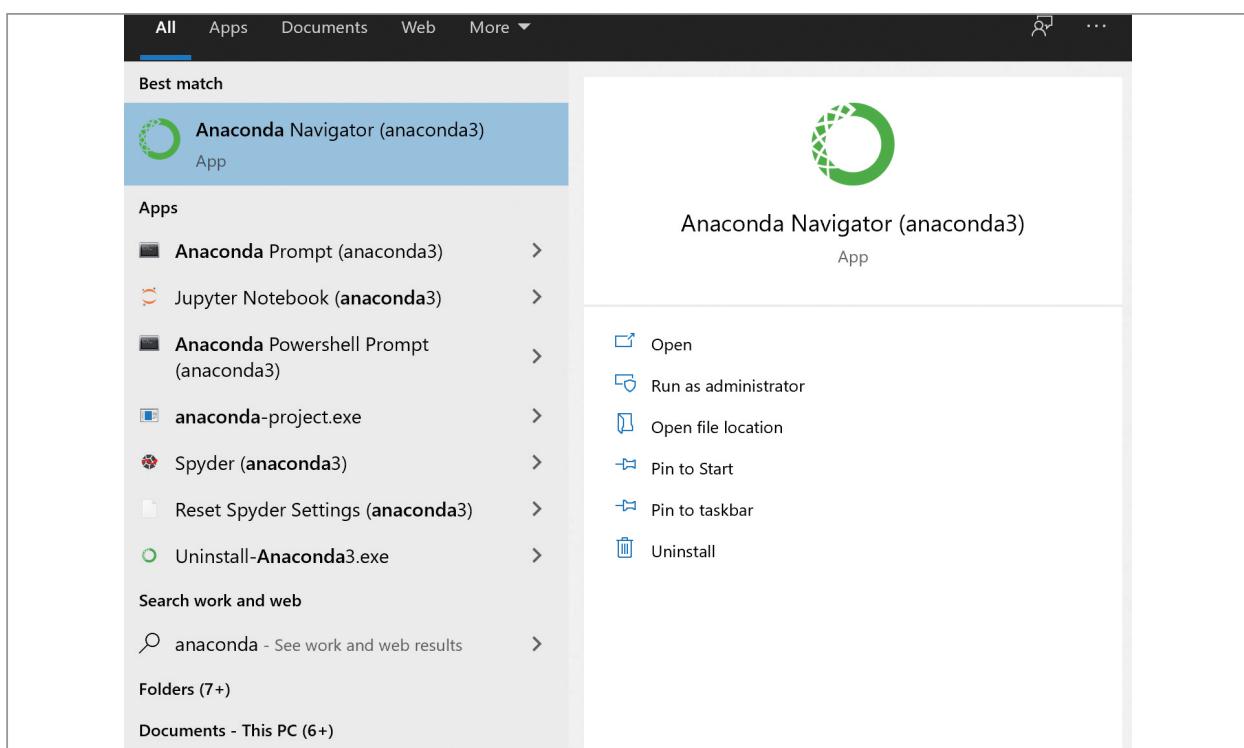


⑩ 마지막 화면에서, 모두 체크 해제한 후, 'Finish' 눌러 설치 완료



⑪ 설치 확인하기

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



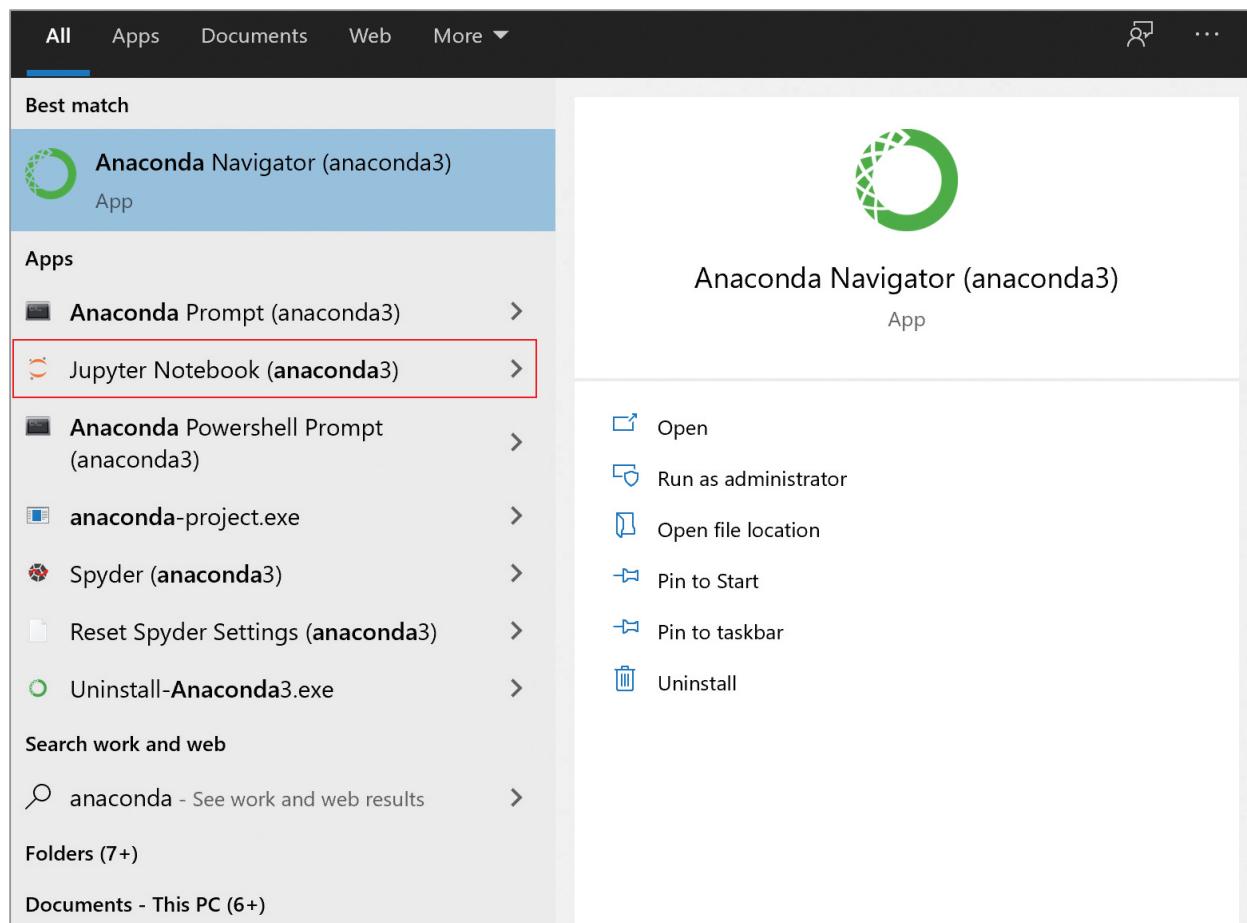
- Anaconda Navigator, Anaconda Prompt, Jupyter Notebook 등 다른 응용 프로그램들도 잘 깔려있는지 확인이 된다면, 설치 완료

3. 주피터 노트북 (Jupyter notebook) 실행



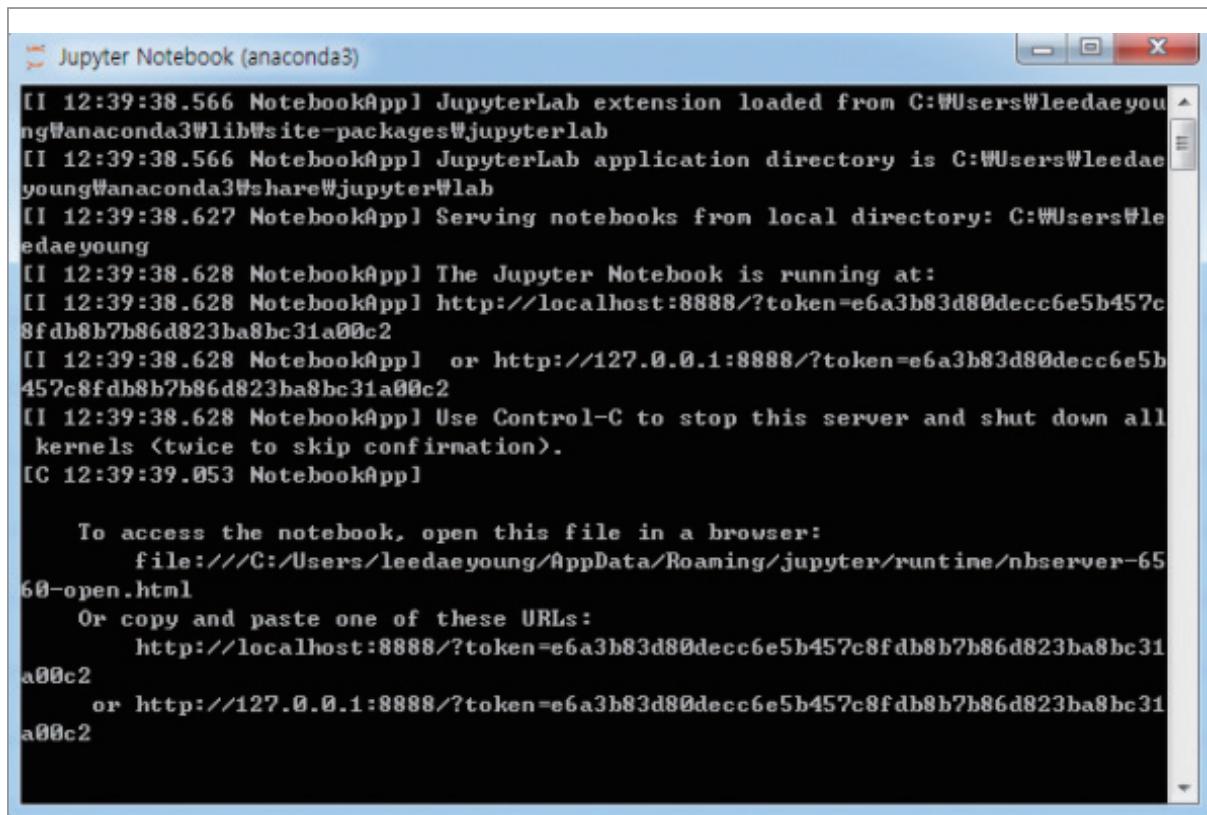
주피터 노트북이란, 실제로 코딩(분석 문장 작성)을 할 수 있는 도구이다. 쉽게 얘기하자면, 문서 도구로 마이크로소프트 사의 ‘Word’ 파일이나, 국내에서는 ‘한글’ 파일 등과 같은 도구라고 생각 할 수 있다. 데이터 분석에 다양한 입력, 실행 도구가 있지만, 본 가이드북에서는 주피터 노트북 을 활용하는 방법을 공유하기로 한다. [부록2]를 참고하여, Anaconda를 설치하였다면, 주피터 노트북(Jupyter notebook) 설치도 확인한다.

- ① 원도우 키()를 눌러서 시작화면을 연 후, anaconda를 검색.폴더 리스트 중 ‘Jupyter notebook(anaconda3)’를 실행한다.



② jupyter notebook을 클릭하게 되면, 2개의 윈도우가 실행.

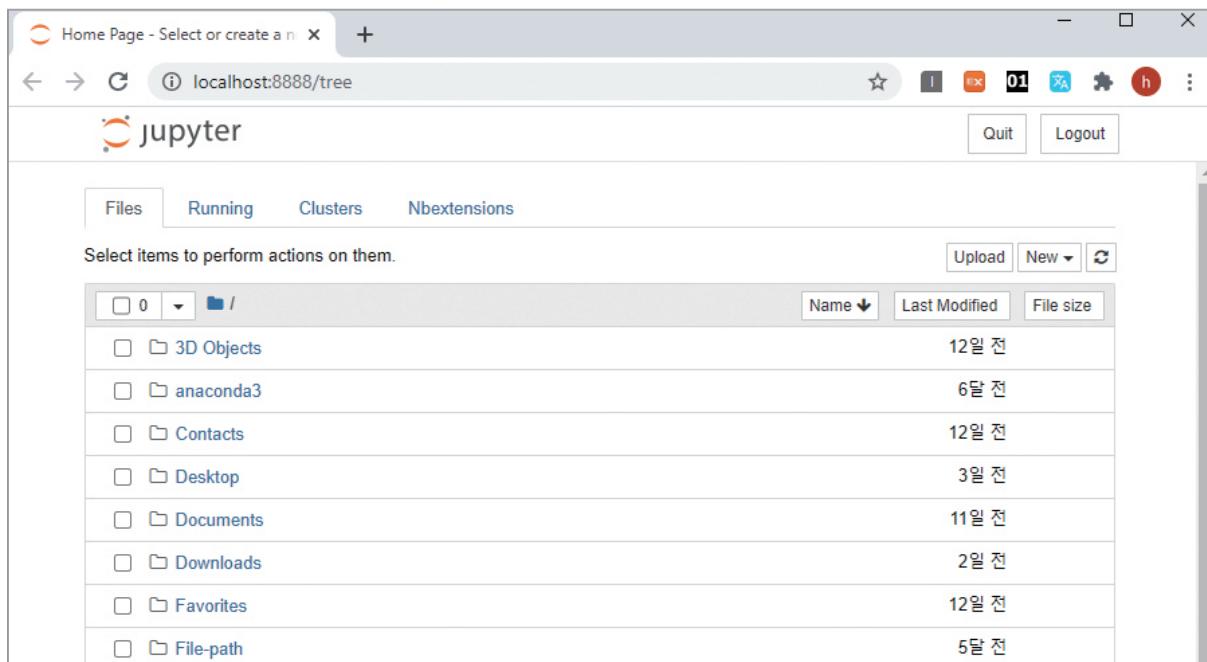
(1) 검은색 배경의 화면은, 주피터 노트북이 실행되는 환경에 대한 상태를 나타내주는 ‘상태 표시창’ 같은 곳. **분석을 하는 동안 종료하면 안된다.**



```
[I 12:39:38.566 NotebookApp] JupyterLab extension loaded from C:\Users\leedaeyoung\Anaconda3\lib\site-packages\jupyterlab
[I 12:39:38.566 NotebookApp] JupyterLab application directory is C:\Users\leedaeyoung\Anaconda3\share\jupyter\lab
[I 12:39:38.627 NotebookApp] Serving notebooks from local directory: C:\Users\leedaeyoung
[I 12:39:38.628 NotebookApp] The Jupyter Notebook is running at:
[I 12:39:38.628 NotebookApp] http://localhost:8888/?token=e6a3b83d80decc6e5b457c8fdb8b7b86d823ba8bc31a00c2
[I 12:39:38.628 NotebookApp] or http://127.0.0.1:8888/?token=e6a3b83d80decc6e5b457c8fdb8b7b86d823ba8bc31a00c2
[I 12:39:38.628 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 12:39:39.053 NotebookApp]

    To access the notebook, open this file in a browser:
        file:///C:/Users/leedaeyoung/AppData/Roaming/jupyter/runtime/nbserver-6560-open.html
    Or copy and paste one of these URLs:
        http://localhost:8888/?token=e6a3b83d80decc6e5b457c8fdb8b7b86d823ba8bc31a00c2
        or http://127.0.0.1:8888/?token=e6a3b83d80decc6e5b457c8fdb8b7b86d823ba8bc31a00c2
```

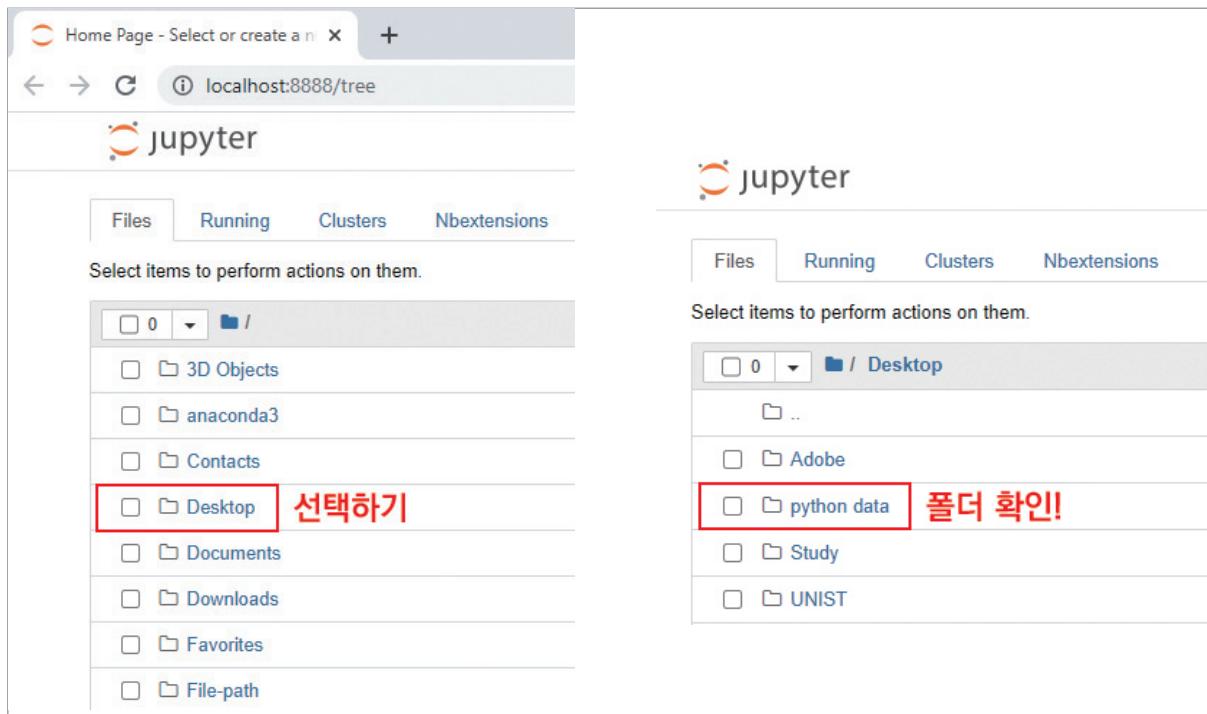
(2) 사용하는 인터넷 프로그램(크롬 / 인터넷 익스플로러)에 주피터 노트북이 열림. (다른 확장 프로그램 사용하고 싶다면, 기본 브라우저를 변경해주어야함). **이 창을 주로 사용.**



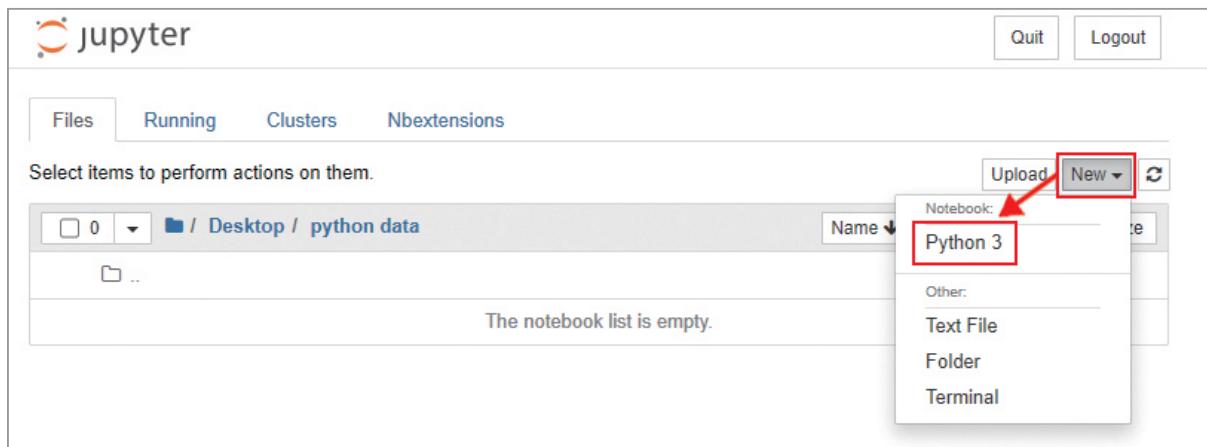
③ 데이터를 저장하고, 불러오고, 분석할 경로의 폴더를 하나 생성

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

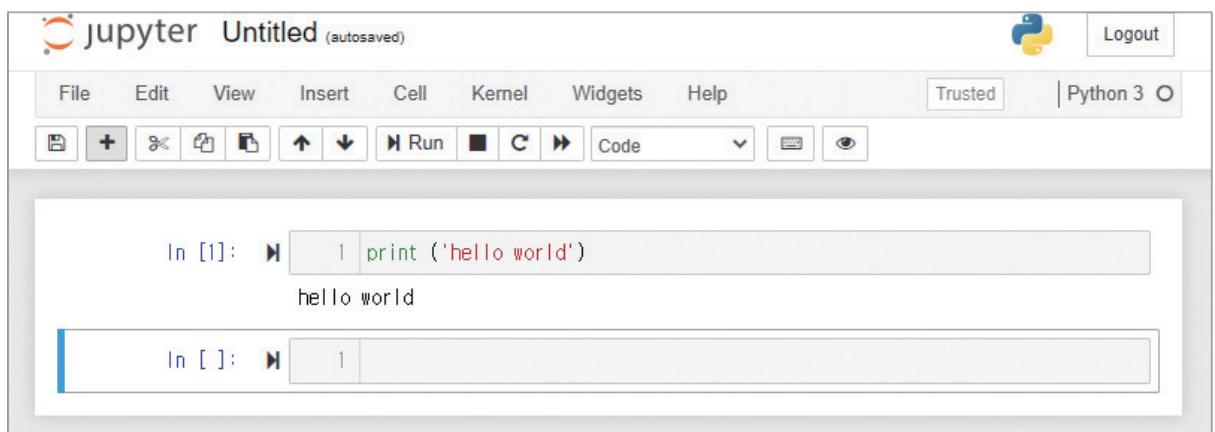
(1) 주피터에서 'python data' 폴더 확인



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

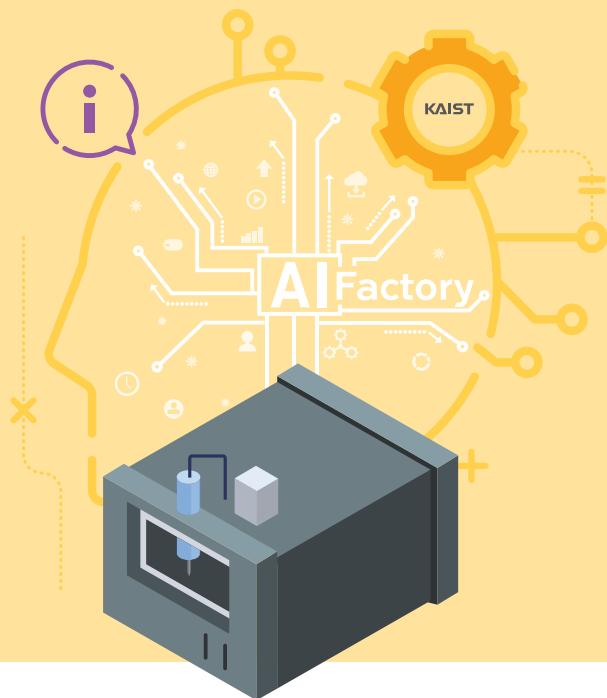


(3) 'hello world' 출력 확인해보기: 보이는 In[] 옆의 회색 창에 `print('hello world')` 입력 후, **shift + Enter** 키 눌러서 실행. : 자동으로 저장되며, 확장자는 '(파일이름).ipynb'로 저장



The screenshot shows a Jupyter Notebook window titled "Untitled (autosaved)". The menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. On the right, there are buttons for Trusted, Python 3, Logout, and a Python logo. Below the menu is a toolbar with icons for file operations like Open, Save, and Run. The main area contains two code cells. The first cell, labeled "In [1]", contains the Python code `print ('hello world')`. When run, it outputs "hello world". The second cell, labeled "In []", is currently empty.

『CNC 머신 AI 데이터셋』 분석실습 가이드북



중소벤처기업부



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