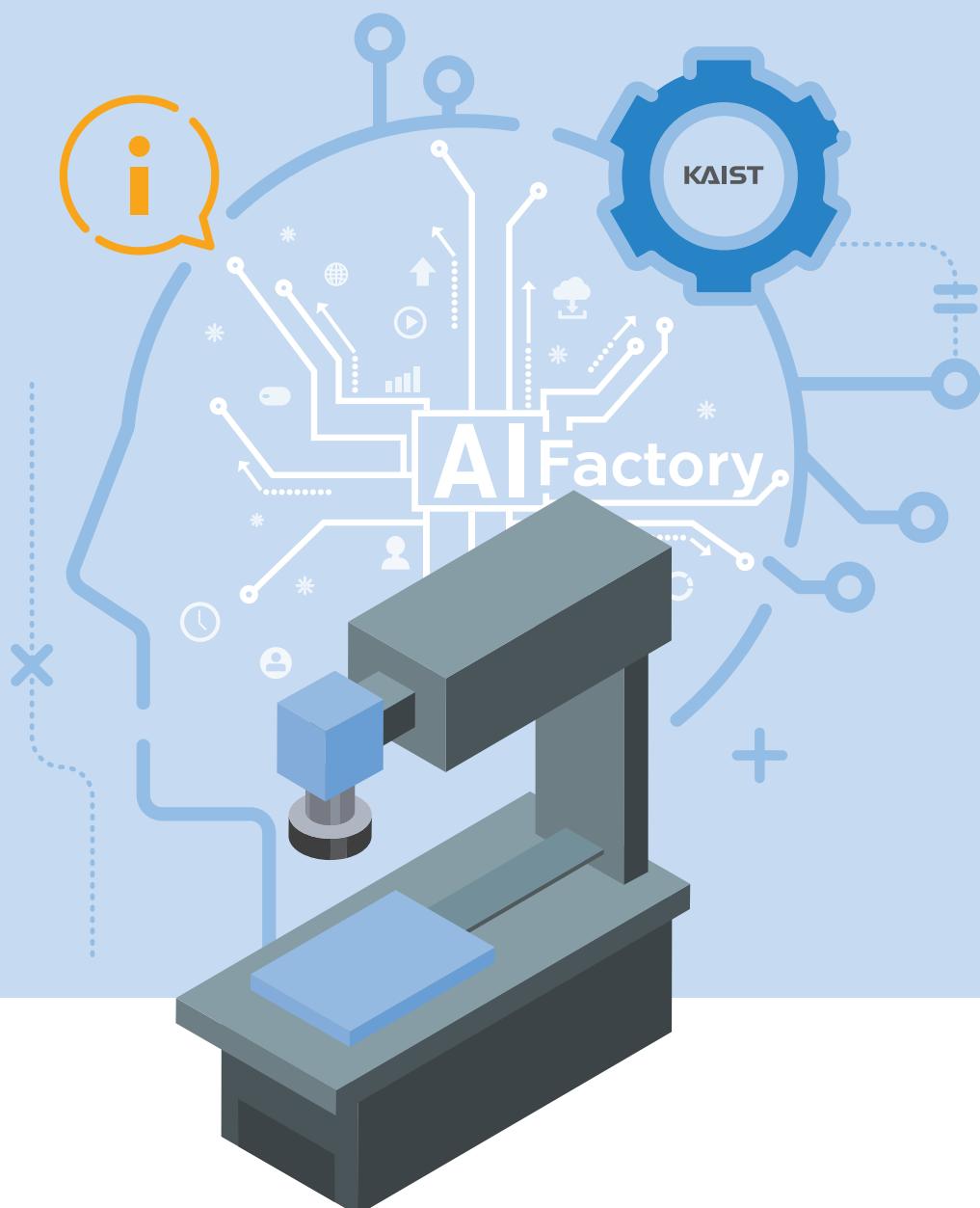
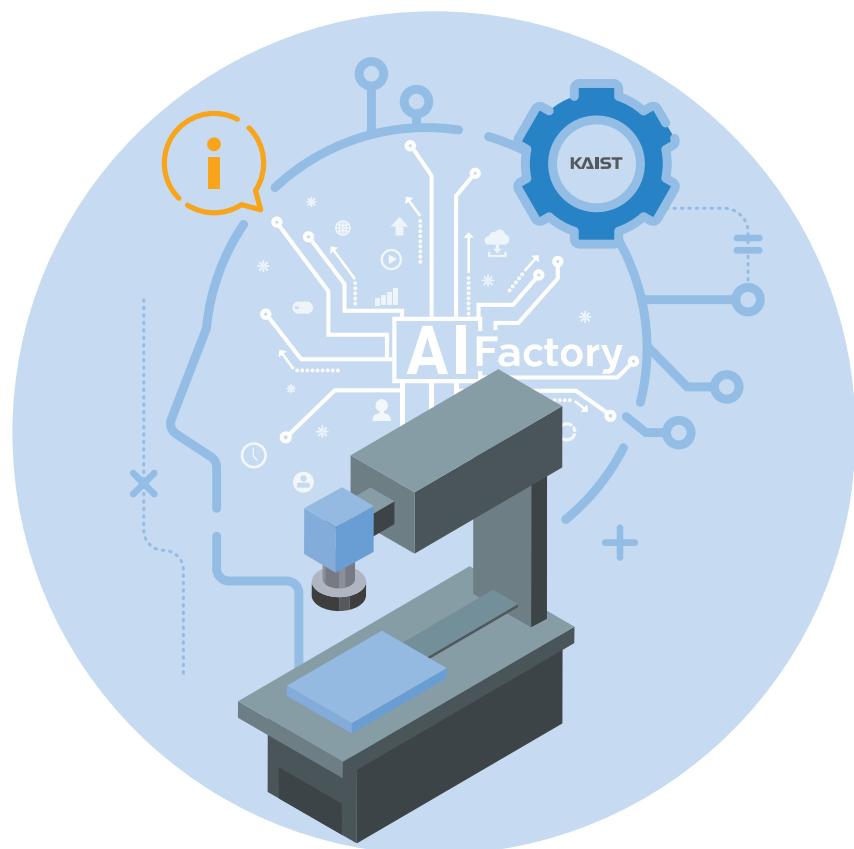


「머신비전 AI 데이터셋」 분석실습 가이드북



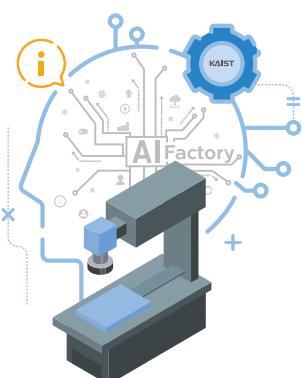
「머신비전 AI 데이터셋」

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 설정	
	2) 제조 데이터 정의 및 소개	
	3) 제조 데이터 분석 기대효과	
	4) 시사점(implication) 요약기술	
	2. 분석실습	10
	2.1 제조데이터 소개	10
	1) 데이터 수집 방법	
	2) 데이터 유형/구조	
	2.2 분석 모델 소개	13
	1) AI 분석모델	
	2.3 분석 체험	15
	1) 필요SW, 패키지 설치 방법 및 절차 가이드	
	2) 분석 단계별 Process	
	[단계 ①] 라이브러리 및 데이터 불러오기	
	[단계 ②] 데이터 시각화	
	[단계 ③] 데이터 정제(전처리)	
	[단계 ④] 라벨 데이터의 불러오기 및 정제	
	[단계 ⑤] 학습/평가 데이터 분리 및 서포트 벡터 머신 학습/평가	
	[단계 ⑥] 실제 상황에 적용 및 결과 분석	
	3. 유사 타 현장의 「머신비전 AI 데이터셋」 분석 적용	33
부 록	분석환경 구축을 위한 설치 가이드	34



「머신비전 AI 데이터셋」 분석실습 가이드북

- 필요 SW : Python
- 필요 패키지 : Numpy, matplotlib, opencv, sklearn, skimage, json
- 분석 환경 : [CPU] Intel(R) Xeon(R) E5-2690 v3 @ 2.60Hz 2.60GHz, [Memory] 128GB, [GPU] NVIDIA Gforce GTX 980
- 필요 데이터 : left_data.csv, right_data.csv, left_label.json, right_label.json
- 주 관 기관 : 한국과학기술원(KAIST)
- 수행 기관 : 울산과학기술원(UNIST), 주식회사 이피엠솔루션즈



1

분석요약

No	구분	내용
1	분석 목적 (현장 이슈, 목적)	<ul style="list-style-type: none">- 자동차 원드실드 사이드몰딩을 생산하기위한 가스 사출성형 시, 불량 발생 시점에서 공정을 멈추거나 제어하는 것이 어렵다. 두께가 동일하지않은 제품은 수축, 뒤틀림 및 싱크가 발생할 수 있으므로 이 공정에서의 가스 양 조절은 중요하다.- 일정시간동안 열화상 카메라를 통해 주입된 가스의 온도와 표면 두께의 온도 비교하여 가스의 양을 측정·불량품을 판정한다.
2	데이터셋 형태 및 수집방법	<ul style="list-style-type: none">- 분석에 사용된 변수명 : 320x256의 열화상 이미지의 모든 온도 raw data, 제품 두께 데이터- 수집 방법 : 학습 데이터는 사출 제품을 일정시간안에 열화상 카메라로 촬영하여 취득. 라벨 데이터는 수기 기록(두께)- 확장자 : csv, json
3	데이터 개수 데이터셋 총량	<ul style="list-style-type: none">- 데이터 개수 : 1,674개- 데이터셋 총량 : 450 MB
4	분석적용 알고리즘	알고리즘 비선형 SVM Machine (좌 우 제품에 각각 적용)
		알고리즘 간략소개 - N차원 데이터를 N-1차원의 초평면(hyperplane)을 분류 경계면으로 이용하여 분류하는 방법. 주어진 데이터를 분류하는 최대 마진의 분류 경계면을 결정하는데 영향을 주는 관측 데이터를 서포트 벡터라고 함. 초평면으로 정확히 분류할 수 없는 데이터의 경우 일부 오분류를 허용하는 소프트 마진을 이용하거나, 커널(kernel)을 이용하여 분류할 수 있는 공간으로 투영하는 기법을 사용할 수 있음.
5	분석결과 및 시사점	<ul style="list-style-type: none">- 적외선 카메라 등을 통하여 취득한 이미지에 포함된 비정형 샘플데이터에서 온도데이터를 분석하고, 컴퓨터 비전기술을 통하여 AI모델을 구축하였다.- 표면 온도를 측정할 수 있는 다양한 공정에서, 제품 온도 분포에 따른 품질 예측을 통해 불량품 발생 및 불량품 재사용 등을 줄이고, 효율적인 생산이 가능할 것으로 기대한다.

2 분석 실습

1. 분석 개요

1.1 분석 배경

1) 공정(설비) 개요

• 공정(설비) 정의 및 특징

- 고분자 수지 재료를 사용하는 윈드실드 사이드몰딩 사출 생산에서 적용되는 공정이다. 성형 방법은 사출 단계 종료 시점에 용융 스트림으로 불활성 가스를 압력에 의해 주입하는 가스 사출성형이다. 가스보다 먼저 용융된 고분자 코어에 충전 및 보압 단계를 진행하여 속이 빈 제품을 생성하게 된다. 제품 생산 후 속이 빈 단면이 설계 사양대로 생성되었는지 여부를 판별하는 머신비전 기능을 제공하여 양품과 불량품을 육안으로 식별하지 않고 자동으로 판별하는 공정이다.

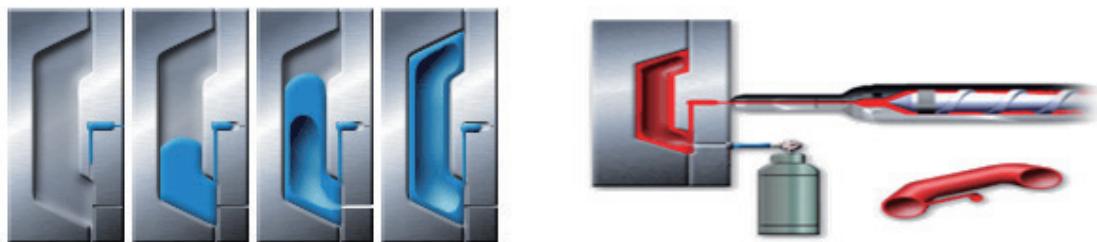
원드실드 사이드몰딩 (Windshield Side Molding)

- 주요기능 : 전면유리의 양끝단을 마감 하는 외장 몰딩으로 주행시 발생하는 소음 및 오염 등을 방지
- 전면 유리 수리(교체)시 탈착 부분



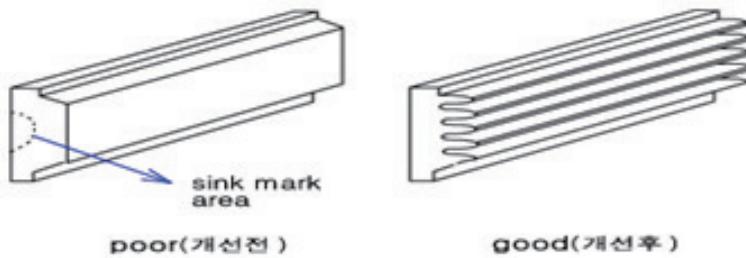
[그림 1] 원드실드 사이드 몰딩

가스 사출성형



[그림 2] 가스 사출 성형의 이해

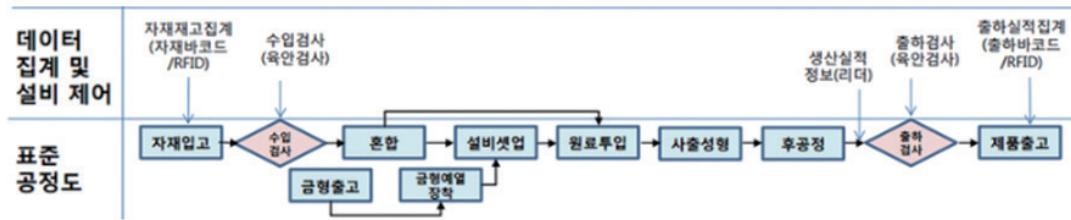
- 사출 안쪽이 지나치게 두껍거나 리브(Rib)가 있을 때, 성형 후 후변형에 의해 움푹 들어가는 외관불량인 Sink mark가 발생하거나, 비틀림(warpage)등 발생
 - 이를 방지하기 위해 단면 두께가 두껍거나 리브(Rib)가 위치한 부분에 가스홀을 만들어 Sink warpage를 방지한다.
 - 이점 : Sink mark / warpage 방지, 재료 절감 및 경량화
 - 사이드 몰딩은 금속 몸체와 유리 장착 단면 구조 특성상 단면이 두껍고, 리브(Rib)가 형성되는 몰딩 디자인이 불가피하다. 따라서 제안적인 디자인 특성을 감안하여 가스 사출 성형이 최적의 성형방법으로 적용 된다.
-



[그림 3] Sink mark의 예

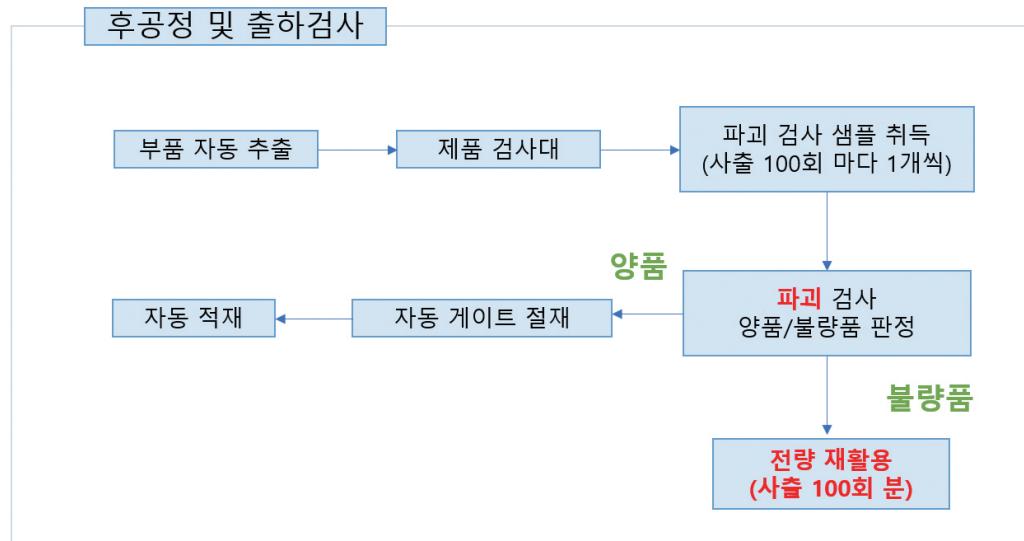
머신비전 공정

표준 공정 : 원료 투입 ▶ 사출 성형 ▶ 후공정 ▶ 출하검사



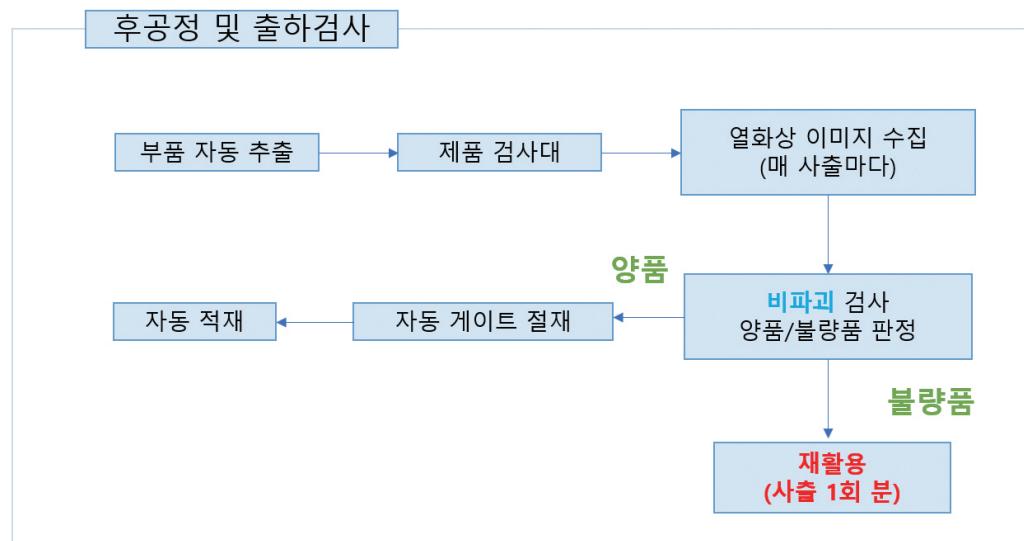
[그림 4] 표준 공정과 기능의 구성 (출처 : 민관합동 스마트공장 추진단)

머신비전 적용 전 후공정 및 출하검사 세부 공정도



[그림 5] 머신비전 적용 전 후공정 및 출하검사 세부 공정도

머신비전 적용 후 후공정 및 출하검사 세부 공정도



[그림 6] 머신비전 적용 후 후공정 및 출하검사 세부 공정도

2) 이슈사항(Pain point)

• 공정(설비)상의 문제현황

- 일반적으로 가스 성형 제품은 상대적으로 일정한 벽 두께로 설계되어 있다. 그러나 간단한 제품을 제외하면 단면의 두께를 동일하게 설계할 수 없고, 제품의 단면마다 다르게 충전 및 보압 된다. 이에 따라 성형 전체에 수축 효과가 발생하고 뒤틀림 및 싱크가 발생할 수 있다. 사용되는 기체와 용융재료의 물리적 특성 차이에 따라 기체를 안정적으로 주입하는 것이 어렵고 이에 따라 생산되는 제품마다 캐비티에 침투되는 폴리머의 양 및 두께가 일정하지 않을 수 있다.

• 문제해결 장애요인

- 성형된 제품이 냉각되어 수축한 후 불량이 발생하고 이를 확인할 수 있으므로 불량 발생 시점에서 공정을 멈추거나 제어하는 것이 어렵다. 제품의 불량을 확인하기 위하여 파괴 검사를 수행하여야 하므로 모든 제품에 대해 불량 여부를 확인하기 어렵고, 일정 기한을 두고 대표 샘플을 취득하여 파괴 검사를 수행할 수 밖에 없으며, 불량 판정인 경우에 앞뒤 일정 개수의 제품을 모두 재활용할 수밖에 없으므로 시간 손실이 발생한다.

• 극복방안

- 용융 된 고분자 코어와 가스의 양에 따라 제품의 각 단면에 충전 및 보압된 고분자의 양이 변하게 되고, 이에 따라 제품이 식는 과정에서 단면의 온도가 다르게 분포된다. 제품 성형 직후에 제품의 열화상 이미지를 취득하고 머신 비전 기술을 이용하여 관심 영역의 온도 분포 데이터 분석을 수행하고 불량을 즉시 판정하여 이를 공정 제어에 이용할 수 있다.

1.2 분석 목표

1) 분석목표 설정

- 사출 시 제품의 금형에 가스가 과다하거나 부족한 경우, 양품과 다르게 제품 내에 과한 빈공간이나 적은 빈공간이 생길 수 있으며, 이로 인해 제품의 뒤틀림이나 내구성 저하를 유발할 수 있다. 제품의 표면 온도가 제품 내의 빈공간의 영향을 받기 때문에, 이후 제공하는 가이드북에서는 제품의 표면 온도와, 파괴 검사를 수행하여 획득한 제품 내 빈공간과 표면 사이의 두께를 활용하여 머신을 학습시키고, 사출 후 제품의 표면 온도 데이터를 이용한 양품/불량품을 판정하고자 한다.



[그림 7] (좌) 가스 성형 시 발생하는 불량 유형, (우) 양품과 불량품의 차이

2) 제조 데이터 정의 및 소개

- 제품의 표면 온도를 촬영하여 취득한 열화상 로우 데이터(raw data)를 ‘학습 데이터’로 사용하고, 파괴 검사를 수행하여 획득한 제품 내 빈공간과 표면 사이의 두께를 ‘라벨 데이터’로 정의한다.

3) 제조 데이터 분석 기대효과

- 제품의 양품/불량품 판정을 표면 온도로 판별이 가능한 모든 제품 생산 공정에 넓게 활용이 가능할 것으로 보인다. 특히, 제품이 식은 이후에 양품/불량품 판단이 가능한 경우, 본 분석방법을 적용하면 제품이 생산된 직후 양품/불량품 판단이 가능하고, 제품이 냉각된 이후 파괴 검사를 수행하고, 불량일 경우 앞뒤 일정 기간의 제품을 버리거나 재활용해야 하는 공정을 최소화함으로, 시간과 재료의 손실을 최소화할 수 있을 것으로 보인다.

4) 시사점(implication) 요약기술

- 자동차 부품 H사 공장에는 파괴 검사와 육안 계측을 통해 사출제품의 양품과 불량을 검사하고 있었으나 파괴 검사를 모든 샘플에 수행 할 수 없고 사람에 의한 판단으로 품질의 균일성을 확보하기 힘들었다. 또한, 제품이 냉각된 이후 양품과 불량을 구분할 수 있었기 때문에 일정기간동안 생산된 제품 중 샘플을 취득하여 파괴 검사를 수행하고 불량으로 판별이 나면 앞뒤 제품 전부를 재활용하는 등의 불량으로 인한 시간 손실이 컸다. 따라서 양품과 불량을 열화상 데이터를 이용해 제품 생산 즉시 판별할 수 있는 AI 기법을 적용하고자 하였다. 사출 공정의 파괴 검사와 두께 측정 자료를 수집하여 가공/전처리 및 AI 모델 개발과 제조공정의 적용 및 검증을 통한 수행으로 열악한 중소기업에 AI를 이용한 판별법을 적용하여 실질적인 품질 향상 및 파괴 검사로 인해 소모되는 비용 개선에 기여했다는 점에서 시사하는 바가 크다고 판단된다.

2. 분석실습

2.1 제조데이터 소개

1) 데이터 수집 방법

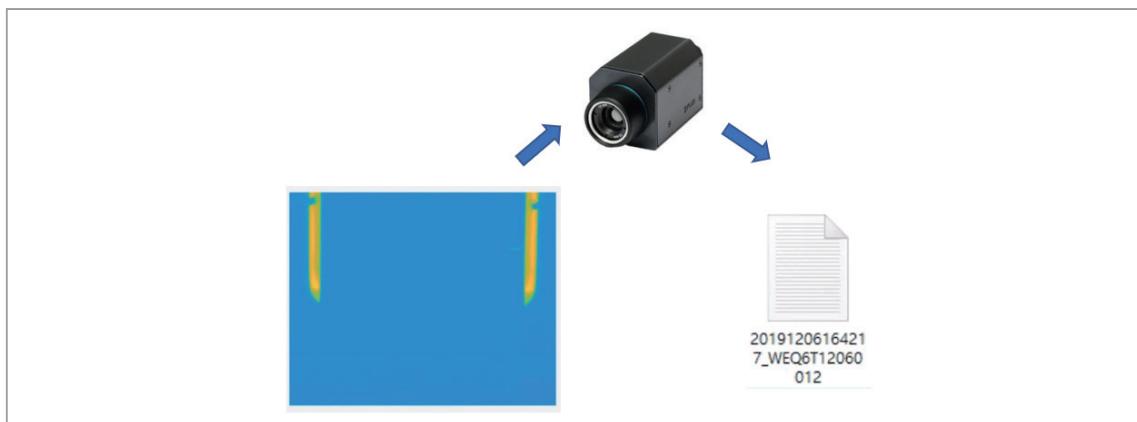
- 제조 분야 : 가스사출성형공법을 이용한 윈드실드 사이드 몰딩
- 제조 공정명 : 머신비전
- 수집장비 : IR카메라
- 수집 기간(주기) : 사이클 타임 약 60초

2) 데이터 유형/구조

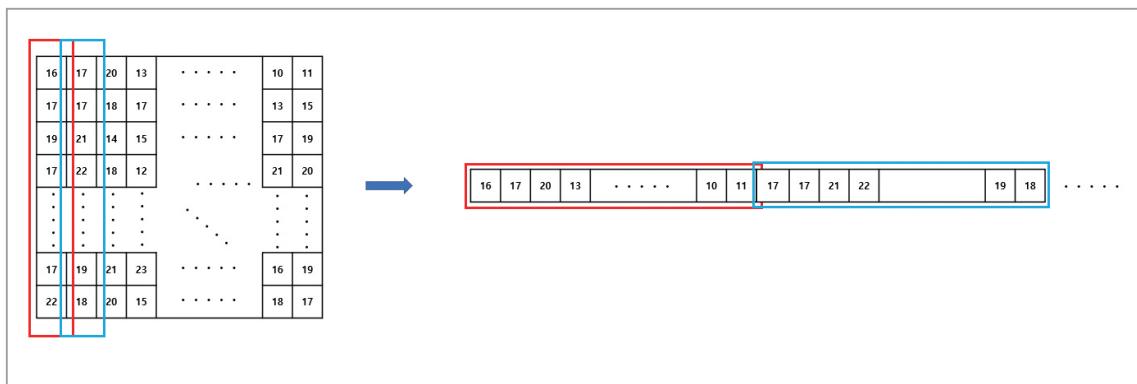
- 데이터셋 구조, 칼럼 수, 데이터 개수, AI 데이터셋 주요 변수 정의

- 원본 데이터(Original Data)

원본 데이터(Original Data)는 사출 공정이 끝난 후 제품을 촬영한 열화상 이미지의 온도로우 데이터(raw data)와 파괴 검수 담당자가 측정하여 수기로 기록한 두께 정보로 이루어져 있다. 총 430여개의 온도 데이터와 두께 정보에서, 좌우 제품별로 얻어지는 온도 데이터를 하나의 파일에 모으고, 수기 기록된 두께 정보를 디지털화하고, 누락된 두께 정보를 제거한 데이터를 1차 가공 데이터라고 한다.



[그림 8] 열화상 카메라로 제품 촬영하여 정보 저장



[그림 9] 온도 raw data 이미지의 벡터화

20191206164217_WEQ&T12060012 - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
17.28393,17.30563,17.41397,17.37063,17.3923,17.47894,17.52221,17.65191,17.69512,17.71^
21,17.54385,17.52221,17.54385,17.50057,17.52221,17.56545,17.6087,17.56545,17.52221,17
,17.58709,17.54385,17.3923,17.30563,17.50057,17.47894,17.54385,17.65191,17.65191,17.8
8192,52.3652,52.66644,51.40932,49.29745,47.57903,46.91817,47.1664,47.57903,48.0888,47
03,17.58709,17.58709,17.54385,17.58709,17.52221,17.54385,17.54385,17.54385,1
9324,17.91085,18.01855,17.97549,18.01855,17.95394,17.82461,18.14761,17.997,17.9324,18
09,17.73831,17.78149,17.86773,17.95394,17.91085,17.95394,17.97549,17.997,17.91085,17.
6,17.3273,17.30563,17.34896,17.34896,17.26223,17.26223,17.3273,17.41397,17.47894,17.4
,44.40813,47.69433,48.98821,49.31368,49.65453,50.80029,52.33346,54.49065,56.6133,57.5
7.45727,17.45727,17.58709,17.52221,17.50057,17.56545,17.58709,17.50057,17.56545,17.60
7.45727,17.47894,17.50057,17.54385,17.58709,17.52221,17.58709,17.52221,17.52221,17.54
```

[그림 10] 벡터화되어 txt 파일 형태로 저장된 온도 raw data

W/No	LH	RH	W/No	LH	RH
0001			0037	0.90	1.22
0002			0038	0.94	1.11
0003			0039	0.84	1.10
0004			0040	0.80	1.08
0005			0041	0.93	1.18
0006	0.86	1.01	0042	0.82	1.10
0007	0.71	1.08	0043	0.75	1.18
0008	0.74	1.02	0044	0.80	1.01
0009	0.73	1.05	0045	0.84	1.15

[그림 11] 파괴 검사 후 수기 라벨 데이터

• 1차 가공 데이터

- 1차 가공 데이터는 데이터 파일(csv)와 라벨 파일(json)으로 이루어진다. 데이터 파일은 256x320 해상도의 열화상 이미지 온도로우 데이터(raw data)가 vector 형태로 열(row)별로 저장되어 있다. 각 칼럼(column)은 열화상 이미지에서 특정 위치의 온도 데이터이다.
- 한 이미지에 제품 두 개가 포함되며, 한 이미지에서 좌 혹은 우측 제품 중 하나의 두께가 누락 된 경우, 다른 쪽 제품은 여전히 사용할 수 있기 때문에, 원본 데이터(Original Data)에서 좌측(left)과 우측(right)으로 나누어 데이터 파일과 라벨 파일로 정리하였다. 라벨 파일에는 디지털화된 두께 정보가 저장되어 있다.

1차 가공 데이터 유형/구조

x: left_data.csv

0 이미지데이터1	d1(1x1)	d1(2x1)	d1(3x1)	...	d1(256x320)
0 이미지데이터2	d2(1x1)	d2(2x1)	d2(3x1)	...	d2(256x320)
...
0 이미지데이터414	d414(1x1)	d414(2x1)	d414(3x1)	...	d414(256x320)

y: left_label.json

라벨1	데이터1의 단면두께
라벨2	데이터2의 단면두께
...	...
라벨414	데이터414의 단면두께

*right side에 대해서도 동일 구조임

양품·불량품 기준

- 양품 : 0.8mm < 단면두께 < 1.5mm
- 불량품 : 단면두께 <= 0.8mm 또는 단면두께 >= 1.5mm

[표 1] 1차 가공 데이터셋 구조

• 2차 가공 데이터

- 2차 가공 데이터는 1차 가공 데이터와 마찬가지로 데이터 파일(csv)와 라벨 파일(json)으로 이루어진다. 1차 가공 데이터에 전처리 기술들을 적용하여, 서포트 벡터 머신을 통해 학습하기 직전의 데이터를 모아 데이터 파일을 형성하였다. 열화상 이미지에서의 관심 영역의 온도 데이터가 1x80사이즈의 벡터(vector) 형태로 열(row)별로 저장되어 있다.

라벨 파일에는 디지털화된 두께 정보와 기준값을 이용하여 0(불량품) 혹은 1(양품)으로 재태깅 작업을 수행한 라벨 데이터가 포함된다.

2차 가공 데이터 유형/구조

x: left_data.csv

열 데이터1	d1(1x1)	d1(1x2)	d1(1x3)	...	d1(1x80)
열 데이터2	d2(1x1)	d2(1x2)	d2(1x3)	...	d2(1x80)
...
열 데이터414	d414(1x1)	d414(1x2)	d414(1x3)	...	d414(1x80)

y: left_label.json

라벨1	데이터1의 양품 / 불량품 정보
라벨2	데이터2의 양품 / 불량품 정보
...	...
라벨414	데이터414의 양품 / 불량품 정보

*right side에 대해서도 동일 구조임

[표 2] 2차 가공 데이터셋 구조

2.2 분석 모델 소개

1) AI 분석모델

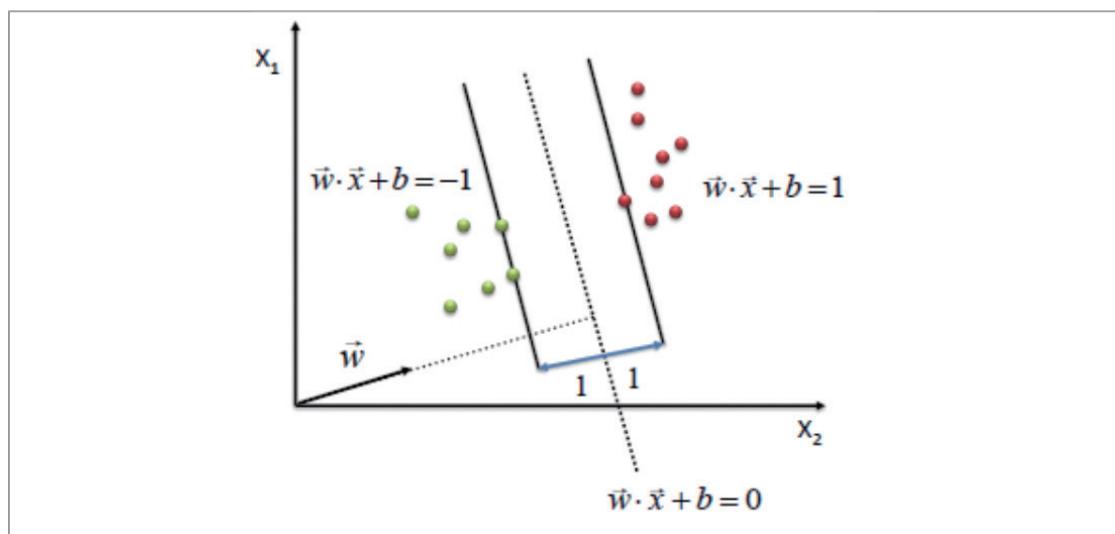
- 해당 AI 방법론(알고리즘) 선정 이유 기술

- 머신비전 데이터 분석 방법 : 서포트 벡터 머신(Support Vector Machine)
 - 서포트 벡터 머신은 AI 알고리즘 학습용 데이터가 충분하다고 판단할 때 사용할 수 있는 방법이다. 서포트 벡터 머신은 마진(Margin)을 최대화하여 초평면(hyperplane)을 결정하기 때문에, 알고리즘 학습시 없었던 유형의 데이터에 대해서도 강인함을 보인다. 따라서 실환경 적용에 적합하다고 판단하여 선정하였다.

- 적용하고자 하는 AI 분석 방법론(알고리즘)의 구체적 소개

- 서포트 벡터 머신 (Support Vector Machine , SVM)

- 서포트 벡터 머신을 학습시킨다는 것은, 양품과 불량품의 데이터를 해당 데이터들의 특징을 이용하여 어느 공간상에 위치시켰을 때, 두 집단을 분류하는 초평면(hyperplane)을 찾는다는 말과 동일하다. 각 집단에서 초평면에 가장 가까운 데이터를 서포트 벡터라고 하며, 서포트 벡터의 거리, 마진(Margin)이 가장 최대화되는 초평면을 학습 과정에서 찾는다. 초평면에 사용되는 커널(Kernel)의 기법에 따라 선형과 비선형 서포트 벡터 머신으로 분류되며, 선형 서포트 벡터 머신의 개념도는 아래와 같다.



[그림 12] Support Vector Machine 알고리즘 개념도

- 해당 개념도는 $y \in \{-1,1\}$ 로 레이블링된 벡터 x 의 분류에 이용되는 예시이다. 그림과 같이 빨간색으로 표현된 $y=1$ 인 데이터와 초록색으로 표현된 $y=-1$ 인 데이터를 나누는 공간상의 직선의 방정식을 구하여 이를 분류에 이용한다. 두 종류의 데이터 간에 가장 큰 마진을 가질 수 있는 직선의 방정식을 찾아 이용한다. 이 직선의 방정

식을 나타내는 벡터 w 를 서포트 벡터라고 한다. 주어진 데이터를 이용하여 서포트 벡터를 학습하고 나면, 앞으로 주어지는 벡터를 아래의 분류 기법을 이용하여 분류 할 수 있다.

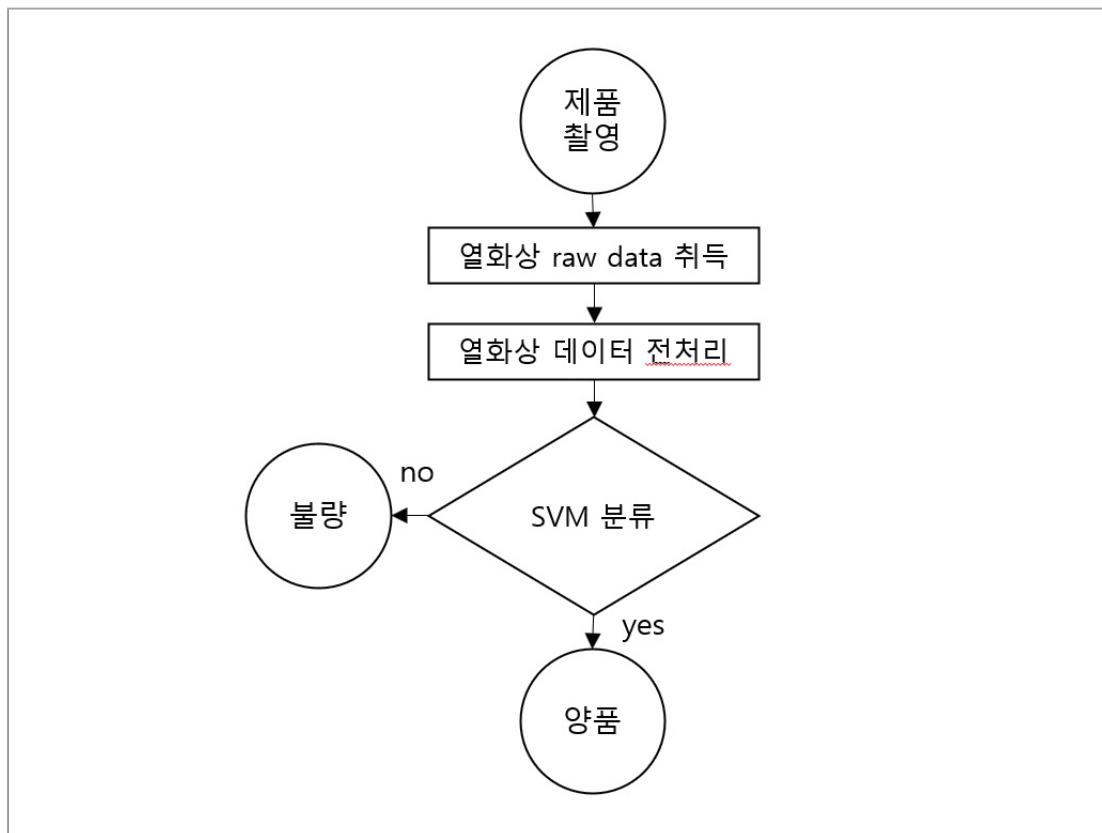
$$x \text{ is class 1 if } wx - b > 0$$

$$x \text{ is class -1 if } wx - b < 0$$

- AI 분석 방법론(알고리즘) 구축 절차 설명

- 서포트 벡터 머신 AI 분석 알고리즘 구축 절차

- 알고리즘을 구축하기 위한 절차는 다음과 같다. ① 학습 데이터 수집 ② 학습 데이터 전처리 ③ 서포트 벡터 머신 알고리즘 훈련 ④ 실제 적용. 실제 적용단계에서 알고리즘의 양품/불량품 판정이 명확하지 않다면, 3번과 4번 과정을 반복하며 데이터를 축적하여 3번 과정에 사용될 데이터의 양과 질을 개선해 나간다.
- 실제 적용단계가 끝난 후, 서포트 벡터 머신을 열화상 데이터를 이용한 양품/불량품 판정에 적용하는 과정을 도식화하면 아래와 같다. 해당 과정을 통해 제품 비파괴 검사가 가능하다.



[그림 13] 서포트 벡터 머신을 이용한 양품/불량품 판정

2.3 분석 체험

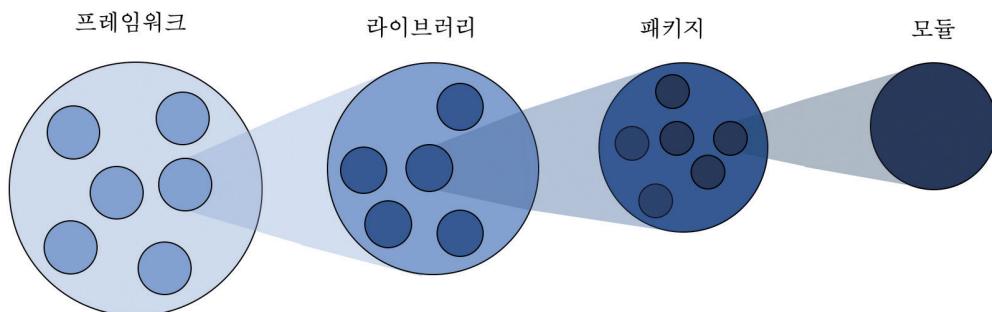
1) 필요 SW, 패키지 설치 방법 및 절차 가이드

- SW 설치는 ‘부록(분석환경 구축을 위한 설치 가이드)’ 참고

• 패키지 설치 가이드

- 패키지 설치 전 파이썬 관련 용어 정리

프레임워크	<ul style="list-style-type: none">- 라이브러리의 모음- 프레임워크가 곧 프로그램의 구성요소이다.
라이브러리	<ul style="list-style-type: none">- 여러 패키지의 모음- 파이썬을 설치할 때, 기본적으로 설치되는 라이브러리를 표준라이브러리라고 한다. 파일 공식이 아닌 외부에서 개발한 모듈과 패키지를 묶어 외부 라이브러리라고 한다.
패키지	<ul style="list-style-type: none">- 여러 모듈들의 모음- 패키지 안에 여러 가지 폴더가 존재할 수 있다.
모듈	<ul style="list-style-type: none">- 특정 함수, 변수, 클래스 등이 구현되어 있는 파일 파일(.py)을 칭한다. 대표적으로 아래 모듈들이 존재한다.- 예) numpy: 수치해석 모듈, pandas: 데이터 분석 모듈



[그림 16] 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->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 numpy  
!pip install matplotlib  
!pip install scikit-image  
!pip install opencv-python  
!pip install sklearn
```

* 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  
entrypoints       0.3
```

* 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

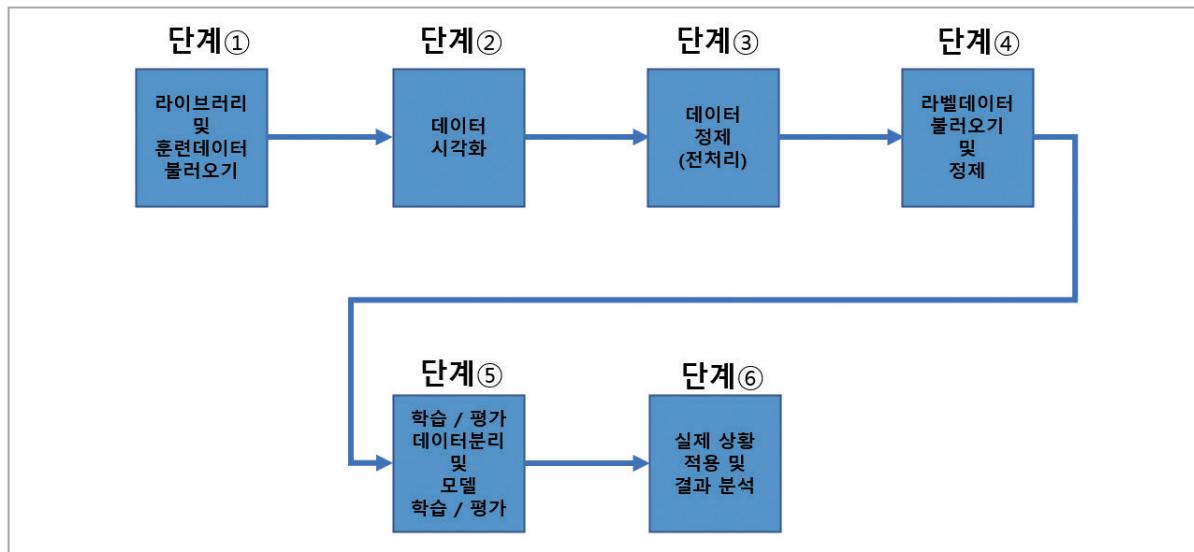
모듈 임포트할 때, 별명(alias)를 지정할 때 사용한다.

```
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 Process



[그림 17] 비지도 학습 적용을 위한 모델 구현 과정

[단계①] 라이브러리 및 데이터 불러오기

이 단계에서는 열화상 데이터를 이용하여 서포트 벡터를 훈련시키기 위해, 1차 가공 데이터 다운로드 및 불러오기 방법을 설명하고, 실습해보는 예제를 준비하였다.

①-1. 필요 라이브러리

```
import numpy as np
import matplotlib.pyplot as plt
from skimage.morphology import skeletonize
import cv2
import json
from sklearn.svm import SVC
```

[코드 1] numpy package import 예제

- 해당 코드를 실행하면, python에서 package를 사용할 수 있게 된다. 사용상 편의를 위해, 일부 함수들은 편의를 위해 축약어를 사용하여 호출할 수 있도록 하였다.

①-2. 훈련 데이터 불러오기

```
file_name ='right_data.csv'# 우측 데이터 활용
vectors = np.zeros((423,81920)) # 전체데이터를 저장할 변수 선언
```

[코드 2] 파일 이름 지정 및 변수 선언

- 코드 2는 불러올 파일 이름을 지정하고, 불러올 데이터를 저장할 변수를 선언하는 예제이다. 해당 예제에서는 오른쪽 제품들을 사용하였다. 데이터 파일은 현재 작성 중인 코드 파일과 동일한 폴더 내에 위치하여야 한다.

```
r =open(file_name,mode='r') # 데이터 파일 open
lines = r.readlines() # 파일에서 변수로 값 불러오기
```

[코드 3] 파일 open 및 데이터 읽어오기

- 코드 3은 실제로 파일을 읽어 그 안에 존재하는 데이터를 불러오는 예제이다. 읽기 전용으로 설정하여 변수 r에 파일을 불러왔으며, 변수 r에 저장되어있는 열화상 데이터값을 변수 lines에 불러오도록 하였다.
- 불러온 데이터 r에는 파일의 정보들이 저장되어있으며, 읽어온 열화상 데이터들은 아래 코드4와 같다.

```
r
```

```
<_io.TextIOWrapper name='right_data.csv' mode='r' encoding='cp949'>
```

```
lines[0]
```

```
'17.392,17.349,17.436,17.522,17.544,17.587,17.738,17.825,17.825,17.911,17.868,17.868,17.868,17.95  
4,18.105,18.083,18.319,18.491,18.641,19.302,20.274,22.665,27.99,39.456,46.835,50.575,52.92,53.409,52.476  
,50.414,48.711,47.793,47.579,47.957,47.99,47.1,44.324,39.509,27.226,19.408,18.362,18.083,17.889,17.803,1  
7.544,17.717,17.63,17.565,17.501,17.306,17.392,17.565,17.803,17.825,17.868,17.868,17.868,17.868,1  
7.825,17.825,17.889,17.889,17.932,17.846,17.825,17.846,17.868,17.868,17.825,17.803,17.889,17.889,17.803,  
17.825,17.846,17.846,17.825,17.825,17.803,17.63,17.609,17.501,17.565,17.674,17.652,17.695,17.587,17.76,1  
7.76,17.803,17.825,17.846,17.825,17.825,17.803,17.781,17.738,17.781,17.738,17.717,17.652,17.76,17.781,17  
.76,17.803,17.76,17.846,17.652,17.738,17.781,17.674,17.695,17.609,17.717,17.674,17.674,17.63,17.652,17.6  
52,17.738,17.781,17.825,17.825,17.76,17.781,17.803,17.846,17.868,17.803,17.803,17.803,17.781,17.7  
6,17.781,17.803,17.803,17.781,17.76,17.738,17.695,17.717,17.738,17.717,17.63,17.609,17.63,17.63,17.565,1  
7.544,17.479,17.501,17.501,17.479,17.392,17.392,17.414,17.371,17.436,17.392,17.436,17.262,17.306,  
17.262,17.349,17.262,17.241,17.262,17.284,17.197,17.197,17.262,17.284,17.306,17.327,17.349,17.371,17.45  
7,17.457,17.544,17.587,17.587,17.63,17.544,17.522,17.522,17.544,17.522,17.609,17.349,17.349,17.414,17.41  
4,17.436,17.349,17.414,17.457,17.457,17.457,17.436,17.414,17.436,17.392,17.501,17.522,...
```

[코드 4] 파일 open 및 데이터 읽어오기 결과

- 이렇게 불러온 데이터는, 우측 모든 제품의 열화상 데이터를 가지고 있다. 제품별로 데이터를 나누고, 변수의 행별로 데이터를 저장하기 위해 코드 5를 실행한다.
- 코드 5 - 코드 6은 설명을 위해 code를 나누어 두었다. 실습 시에는 코드 10을 참고하여 사용하기 바란다.

```
for data in range(0,423):  
    numbers = lines[data].split(',') #각 행에서 comma로 데이터를 구분하여 변수에 저장
```

[코드 5] 원하는 위치의 데이터 원하는 형태로 가져오기 예제 (split)

- 모든 열화상 데이터 정보가 담겨있는 lines 함수에서, 콤마(',')를 기준으로 split 함수를 수행하여 데이터값을 분류하고 열 이미지의 크기만큼 데이터를 차례로 불러와 저장한다.

```
flir_vector = [] # 행 데이터를 저장할 변수 선언  
for i in range(0,81920):  
    flir_vector.append(float(numbers[i])) # 데이터를 순서대로 붙여 행에 저장  
  
vectors[data,:] = np.transpose(flir_vector) # 전체 데이터 변수에 저장
```

[코드 6] 원하는 위치의 데이터 원하는 형태로 가져오기 예제2 (float, append, transpose)

- 콤마(',') 단위로 분리된 데이터가 들어있는 numbers 함수에서, 열화상 이미지의 크기인 $256 \times 320 = 81920$ 만큼의 데이터를, float 형태로 가져와 flir_vector 변수에 저장한다. append 함수는 데이터값을 하나씩 옆으로 붙여 주는 함수이다.
- 코드 5의 split 함수와 코드 6의 append 함수의 적용 결과는 다음과 같다

```
lines[data] #각 행에서 comma로 데이터를 구분하여 변수에 저장
```

```
'23.728,23.626,23.646,23.667,23.809,23.728,23.728,23.748,23.748,23.728,23.667,23.829,23.89,23.911,24.032  
,24.194,24.255,24.397,24.558,25.201,26.179,28.01,32.183,41.031,46.935,49.897,51.553,52.063,51.585,50.35,  
49.346,48.711,48.646,48.678,48.695,47.875,45.803,41.932,31.788,25.642,24.719,24.316,24.215,24.174,24.13  
4,24.134,23.992,23.931,23.809,23.748,23.687,23.748,23.809,23.829,23.85,23.829,23.829,23.87,23.809,23.85,  
23.809,23.809,23.809,23.789,23.748,23.646,23.748,23.809,23.728,23.707,23.789,23.687,23.707,23.728,23.68  
7,23.789,23.809,23.748,23.748,23.728,23.687,23.707,23.646,23.646,23.585,23.585,23.545,23.524,23.5  
04,23.545,23.545,23.545,23.524,23.545,23.565,23.565,23.565,23.565,23.545,23.545,23.565,23.565,23.585,23.  
585,23.606,23.524,23.463,23.585,23.524,23.626,23.524,23.422,23.565,23.565,23.606,23.483,23.565,23.463,2  
3.483,23.463,23.545,23.463,23.402,23.585,23.585,23.585,23.626,23.585,23.565,23.524,23.545,...
```

```
lines[data].split(',') #각 행에서 comma로 데이터를 구분하여 변수에 저장
```

```
['23.728',  
'23.626',  
'23.646',  
'23.667',  
'23.809',  
'23.728',  
'23.728',  
...]
```

[코드 7] 원하는 위치의 데이터 원하는 형태로 가져오기 결과 (split)

```
flir_vector
```

```
['23.728',  
'23.626',  
'23.646',  
'23.667',  
'23.809',  
'23.728',  
'23.728',  
...]
```

[코드 8] for 문 안에서 float, append 함수 실행 결과

- append 함수를 통해 완성된 81920x1 크기의 데이터를 transpose 함수를 이용하여 1x81920 크기의 벡터로 변환하고, 이 벡터를 vectors 변수의 행에 저장한다. for 문 안에서 해당 과정을 반복하면 vectors의 행별로 열화상 데이터들이 저장된다. 코드 8의 수행 결과는 다음과 같다.

```
vectors
```

```
array([[17.392, 17.349, 17.436, ..., 16.259, 16.215, 16.193],  
       [17.846, 17.889, 17.932, ..., 16.522, 16.522, 16.522],  
       [17.284, 17.306, 17.414, ..., 16.369, 16.347, 16.325],  
       ...,  
       [23.789, 23.545, 23.626, ..., 22.685, 22.706, 22.85 ],  
       [23.809, 23.748, 23.768, ..., 22.809, 22.891, 22.87 ],  
       [23.728, 23.626, 23.646, ..., 22.685, 22.747, 22.788]])
```

[코드 9] 원하는 위치의 데이터 원하는 형태로 가져오기 예제2 결과

```

file_name ='right_data.csv'# 우측 데이터 활용
vectors = np.zeros((423,81920)) # 전체데이터를 저장할 변수 선언

r =open(file_name,mode='r') # 데이터 파일 open
lines = r.readlines() # 파일에서 변수로 값 불러오기

for data in range(0,423):

    numbers = lines[data].split(',') #각 행에서 comma 로 데이터를 구분하여 변수에 저장
    flir_vector = [] # 행 데이터를 저장할 변수 선언

    for i in range(0,81920):
        flir_vector.append(float(numbers[i])) # 데이터를 순서대로 붙여 행에 저장

    vectors[data,:] = np.transpose(flir_vector) # 전체 데이터 변수에 저장

```

[코드 10] 데이터 불러와 가공하기 전체 코드

- 이제 모든 열화상 데이터를 제품별로 변수의 각 행에 저장하였으며, 다음으로는 불러온 열화상 데이터를 눈으로 확인한다.

[단계②] 데이터 시각화

이 단계에서는 vectors 변수에 저장된 열화상 데이터들이 어떤 이미지인지 직접 눈으로 확인한다.

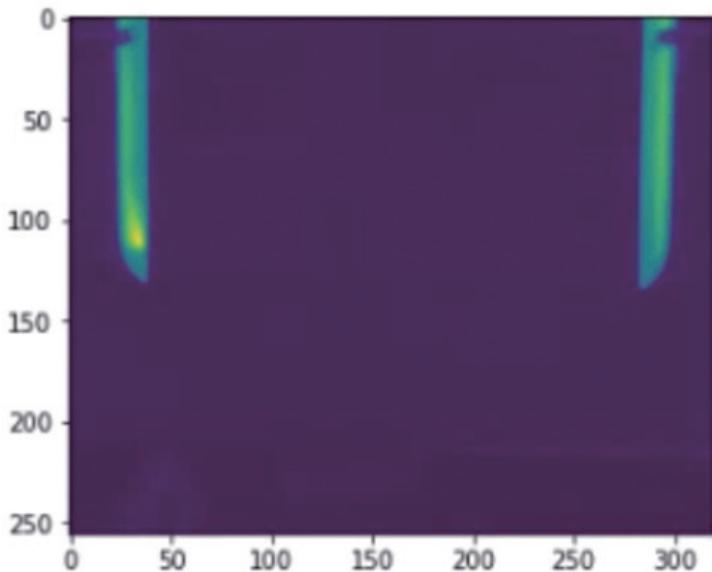
②-1. 데이터 시각화

```
import matplotlib.pyplot as plt
```

[코드 11] 시각화 package matplotlib import 예제

- matlabplotlib package를 cmd를 이용하여 설치한 뒤, python에서 사용할 수 있도록 import 한다.
- 데이터를 시각화하여 디스플레이하기 위해, figure 1번 창을 열어준다. 이후 vectors에 저장된 첫 번째 열화상 데이터를 불러오고, reshape 함수를 이용하여 256x320 사이즈의 이미지로 만들어 준다. 이후 imshow 함수를 이용하여 이미지를 figure 1번 창에 디스플레이하고, show 함수를 이용하여 사용자가 볼 수 있도록 출력한다.

```
plt.figure(1)
plt.imshow(np.reshape(vectors[1,:],[256,320]))
plt.show()
```



[코드 12] 열화상 데이터 시각화

- 보이는 것과 같이, 변수 vectors에 데이터 파일에 들어있던 열화상 이미지의 온도 데이터가 들어온 것을 확인할 수 있다.

[단계③] 데이터 정제(전처리)

이 단계에서는 열화상 데이터를 서포트 벡터 머신에 훈련시키기 위해, 필요한 영역의 데이터를 추출하여 2차 가공 데이터를 생성하는 전처리 예제를 제공한다.

③-1. threshold mask 추출 예시

```
# 전처리 예제
from skimage.morphology import skeletonize
import cv2
```

[코드 13] 전처리용 package import 예제 (skimage, opencv)

- 전처리를 수행하기 위해 opencv package와 skimage package를 import 한다. opencv package(cv2)는 ‘pip install cv2’ 가 아닌 ‘pip install opencv-python’ 명령어를 사용하여 인스톨 할 수 있다.
- 코드 14 - 코드 19 까지는 설명을 위해 for 문을 나누어 두었다. 실습시에는 코드 24를 사용하기 바란다.

```

processed_vectors = np.zeros((423,80)) # 전처리 적용된 전체 데이터 저장용 변수 선언

for j in range(0,423):
    flir_image = np.reshape(vectors[j,:],[256,320]) # data 이미지화

```

[코드 14] 데이터 reshape 예제

- import 이후 전처리를 적용한 데이터를 저장할 수 있도록 변수 processed_vectors를 선언한다. 이 변수에는 전처리가 적용된 1x80짜리 벡터를 제품의 개수 만큼 행에 저장한다. for 문을 열어 제품의 개수만큼 돌아가도록 설정하고, reshape 함수를 통해 데이터를 이미지화한다.

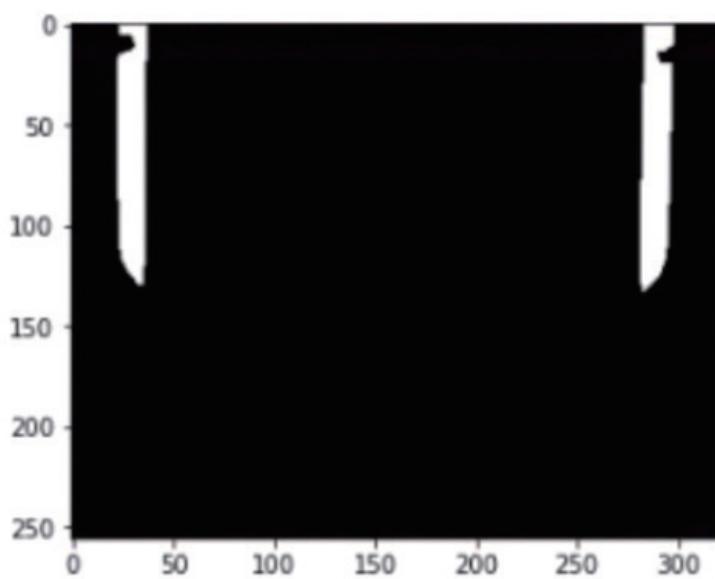
```

ret, mask = cv2.threshold(flir_image,41,255,cv2.THRESH_BINARY) # data 마스크 제작
mask = mask/255 # 0부터 1까지로 normalization

```

[코드 15] 이미지 threshold 적용 예제 및 normalize 적용 예제

- opencv의 threshold 함수를 사용하여 이미지에서 기준값(threshold) 이상의 데이터값을 가진 경우 데이터값을 255로, 이하의 경우 0으로 만든다. 이후 모폴로지(morphology) 연산을 수행하기 위해서, 255를 1로 만드는 정규화(normalization)를 수행한다.
- 이렇게 형성된 mask를 확인하여 보면 아래와 같다. mask 변수를 시각화하기 위한 코드는 전체 코드 뒤에 소개한다.



[그림 14] mask 변수의 시각화

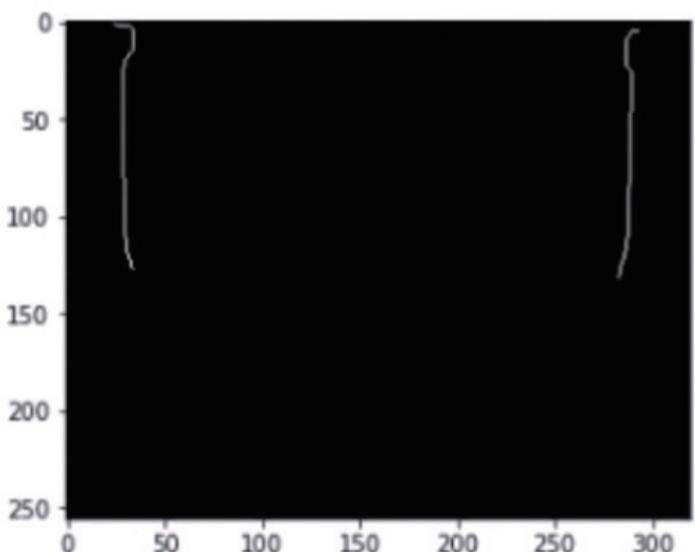
③-2. mask에서 skeleton 이미지 추출 예시

- 만들어진 mask 이미지에, skeleton 함수를 적용한다. skeleton 함수는 1값을 가진 영역을 축소한다. 결과는 다음과 같다.

```
skeleton = skeletonize(mask) # skeleton 적용
```

[코드 16] skeleton 함수 적용

- skeleton 변수를 시각화 하면 아래와 같다. 시각화를 위한 코드는 전체 코드 뒤에 소개한다.



[그림 15] skeleton 변수의 시각화

③-3. 합성 이미지 제작 예시

```
# vector 추출
mixed = flir_image*skeleton # skeleton 위치의 픽셀 값 이미지화
mixed_right = mixed[0:255,160:319] # 우측 제품만 추출
```

[코드 17] Region of Interest 추출 예제

- 열화상 이미지와, skeleton 이미지에 요소간 곱셈을 적용하면, 원하는 영역의 열데이터만 존재하는 합성 이미지를 만들어 낼 수 있다. 예시의 경우 우측 제품만을 이용할 것이기 때문에, 우측 영역 제품만을 떼어낸다.

③-4. 관심 영역에서 원하는 데이터 값 추출 예시

```
min_thresh = 0

res = np.where(mixed_right>min_thresh)[0] # 제품의 시작 좌표 확인
maxn = np.max(res) # 제품의 ROI 시작좌표
minn = np.max(res) - 80 # 제품의 ROI 끝 좌표
```

[코드 18] 이미지 내 제품 위치 추출 예제

- numpy의 where 함수와 논리 연산을 이용하여 합성 이미지에서 0이 아닌 값이 존재하는, skeleton 위치의 픽셀 값만 존재하는 영역의 실제 좌표를 취득한다.
- 제품이 존재하는 위치의 시작점과 끝점만 사용하면 되기 때문에, 최소값과 최대값만을 max 함수와 min 함수를 통해 추출하여 사용한다.

```

num =0
vector = np.zeros((80,1))

for i in range(minn,maxn):
    ii = i-(minn)+1
    iii = maxn - ii # 제품의 시작 좌표부터 데이터 값 저장을 저장하기 위한 인덱스 변경
    index = np.where(mixed_right[iii,0:319] >0)[0] # 각 행에서 데이터가 존재하는 열인덱스 저장
    vector[num,0] = mixed_right[iii,index] # 열 데이터값 저장
    num = num +1

vector = np.transpose(vector)
processed_vectors[j,0:80] = vector # 전처리 완료 데이터 저장

```

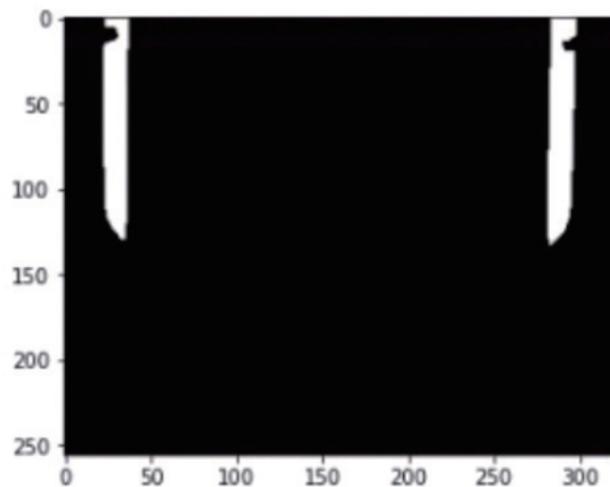
[코드 19] 제품 위치에서 원하는 순서로 데이터 가져오기 예제

- 제품의 시작점과 끝점까지의 픽셀 값을 저장하는 for문을 제작한다. python의 for 문이 range의 역순 증가를 허용하지 않기 때문에, num 과 변수 ii, iii를 이용하여 역순 증가를 구현한다. for 문을 이용하여 열화상 이미지의 행 별로 접근하여, where 함수를 이용해 각 행에서 데이터 값이 존재하는 좌표를 취득하고, 그 좌표에서의 값을 순서대로 저장하여 정보가 저장된 vector를 생성한다. 이후 데이터 별로 만들어진 vector를 processed_vectors 변수에 저장한다.

③-5. 전처리 데이터의 시각화

- 3-1 ~ 3-4 코드의 결과로 생성된 변수들은 아래 코드들을 이용해 시각화할 수 있다.
- mask 변수를 시각화 하기 위해 아래의 코드를 실행한다. imshow 함수 안의 cmap = plt.cm.gray는 이미지를 흑백으로 보여준다.

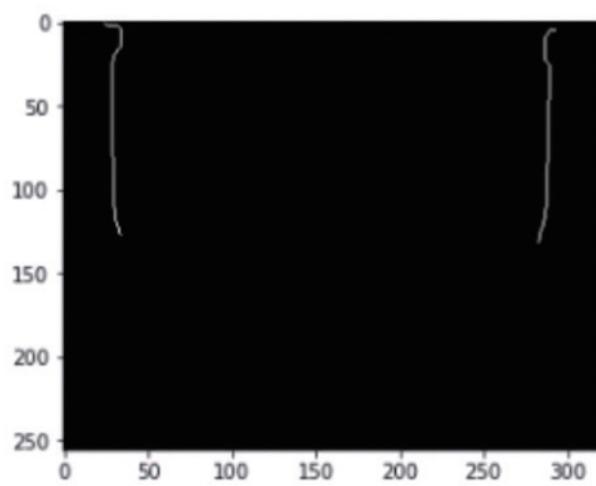
```
plt.figure(1)
plt.imshow(mask,cmap=plt.cm.gray)
plt.show()
```



[코드 20] mask 데이터 시각화

- skeleton 코드는 동일한 코드에 변수명 만을 교체하여 시각화 할 수 있다.

```
plt.figure(1)
plt.imshow(skeleton,cmap=plt.cm.gray)
plt.show()
```



[코드 21] skeleton 적용 및 시각화

- 제품의 좌표를 저장하는 res 변수는 호출하면 아래와 같다.

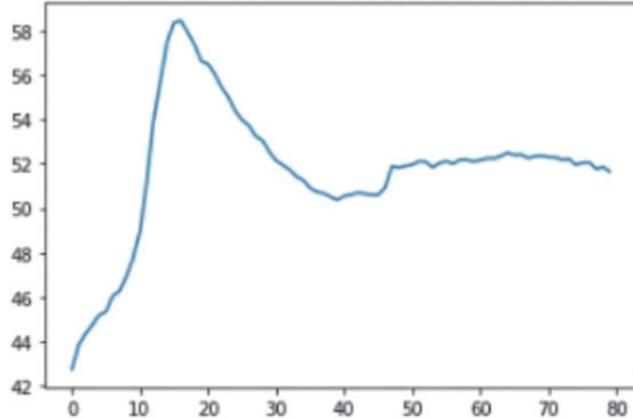
```
res
```

```
array([ 5,  5,  5,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14,
       15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27,
       28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40,
       41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53,
       54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66,
       67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79,
       80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92,
       93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105,
       106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118,
       119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131,
       132], dtype=int64)
```

[코드 22] 이미지 내 제품 위치 추출 결과

- processed_vector 들의 최고점(peak)의 높낮이와 위치가, 제품들의 온도를 구별할 수 있는 중요한 정보가 된다. 예시를 들기 위해 423번 제품의 전처리 데이터를 시각화한다.

```
plt.figure(1)
plt.plot(np.arange(0,80),processed_vectors[422])
plt.show()
```



[코드 23] 추출 데이터 가져오기 결과 시각화

```

from skimage.morphology import skeletonize
import cv2

processed_vectors = np.zeros((423,80)) # 전처리 적용된 전체 데이터 저장용 변수 선언

for j in range(0,423):

    flir_image = np.reshape(vectors[j,:][256:320]) # data 이미지화
    ret, mask = cv2.threshold(flir_image, 41, 255, cv2.THRESH_BINARY) # data 마스크 제작

    mask = mask/255 # 0부터 1까지로 normalization

    skeleton = skeletonize(mask) # skeleton 적용

    # vector 추출
    mixed = flir_image*skeleton # skeleton 위치의 픽셀 값 이미지화
    mixed_right = mixed[0:255,160:319] # 우측 제품만 추출

    min_thresh = 0

    res = np.where(mixed_right > min_thresh)[0] # 제품의 시작 좌표 확인

    maxn = np.max(res) # 제품의 ROI 시작좌표
    minn = np.max(res) - 80 # 제품의 ROI 끝 좌표

    num = 0

    vector = np.zeros((80,1))

    for i in range(minn,maxn):
        ii = i-(minn)+1
        iii = maxn - ii # 제품의 시작 좌표부터 데이터 값 저장하기 위한 인덱스 변경

        index = np.where(mixed_right[iii,0:319] > 0)[0] # 각 행에서 데이터가 존재하는 열 인덱스 저장

        vector[num,0] = mixed_right[iii,index] # 열 데이터값 저장
        num = num + 1

    vector = np.transpose(vector)
    processed_vectors[j,0:80] = vector # 전처리 완료 데이터 저장

```

[코드 24] 전처리 전체 코드

[단계④] 라벨 데이터의 불러오기 및 정제

전 단계에서 학습에 사용될 2차 가공 데이터가 준비되었기 때문에, 이번 단계에서는 정답 역할을 하는 1차 가공 데이터 라벨 파일을 불러와서 원하는 형태로 가공하여 2차 가공 라벨 데이터를 생성한다. 라벨 데이터는 제품의 두께 값을 가지고 있으며, 두께가 0.8~1.5 범위 안의 제품은 양품으로, 아닌 제품은 불량으로 정의한다. 본 예시에서는 양품은 1, 불량품은 0으로 저장한다.

- 라벨 데이터는 json 파일로 저장되어있기 때문에, json package를 불러온다. 이후 with open ~ as 함수를 사용하여, json 파일을 열어준다. open 함수에 파일이름과 모드(예제에서는 r, read 모드)를 기입해 사용한다. as 뒤의 infile 은 해당 json 파일이 포함하고 있는 데이터의 개수만큼 정의된다. json.load(infile)을 이용하여, 해당 json 파일이 포함하고 있는 모든 데이터를 불러와 newlist 변수에 저장한다.

```

import json
with open('right_label.json', 'r') as infile:
    newlist = json.load(infile)

print(newlist)

```

```

[1.1, 1.1, 1.08, 1.05, 1.11, 1.13, 1.04, 0.94, 1.08, 1.01, 1.02, 1.01, 1.42, 1.8, 1.12, 1.78, 1.78, 1.68, 1.18, 1.48, 1.67, 1.6, 1.54, 1.6, 1.59, 1.54, 1.21, 1.52, 1.19, 1.27, 1.4, 1.35, 1.35, 1.19, 1.41, 1.39, 1.45, 1.47, 1.53, 1.57, 1.4, 1.61, 1.48, 1.54, 1.56, 1.45, 1.49, 1.32, 1.11, 1.04, 1.19, 1.03, 1.16, 1.08, 1.13, 1.16, 1.07, 1.06, 1.14, 1.15, 1.19, 1.16, 1.21, 1.1, 1.15, 1.16, 1.15, 1.2, 1.12, 1.2, 1.22, 1.11, 1.1, 1.08, 1.18, 1.1, 1.18, 1.09, 1.15, 1.13, 1.15, 1.16, 1.2, 1.2, 1.18, 1.18, 1.2, 1.14, 1.23, 1.39, 1.44, 1.56, 1.52, 1.42, 1.56, 1.47, 1.26, 1.37, 1.26, 1.24, 1.34, 1.47, 1.32, 1.3, 1.17, 1.1, 1.08, 1.13, 1.14, 1.14, 1.07, 1.11, 1.1, 1.11, 1.11, 1.18, 1.37, 1.36, 1.44, 1.21, 1.18, 1.48, 1.18, 1.15, 1.17, 1.22, 1.1, 1.16, 1.12, 1.16, 1.12, 1.1, 1.15, 1.12, 1.13, 1.14, 1.12, 1.13, 1.12, 1.14, 1.15, 1.18, 2.23, 2.15, 2.16, 2.13, 2.13, 2.12, 2.12, 1.39, 1.29, 1.48, 1.4, 1.2, 1.6, 1.29, 1.15, 1.16, 1.22, 1.67, 1.45, 1.36, 1.44, 1.16, 1.17, 1.41, 1.27, 1.48, 1.46, 1.26, 1.59, 1.27, 1.4, 1.14, 1.34, 1.38, 1.45, 1.4, 1.17, 1.07, 1.04, 1.07, 1.1, 1.07, 1.1, 1.14, 1.08, 1.14, 1.13, 1.07, 1.13, 1.1, 1.08, 1.12, 1.04, 1.15, 1.07, 1.05, 1.14, 1.17, 1.11, 1.17, 1.15, 1.14, 1.43, 1.2, 1.31, 1.42, 1.55, 1.14, 1.37, 1.13, 1.59, 1.2, 1.36, 1.44, 1.84, 1.93, 1.89, 1.9, 1.96, 1.91, 1.86, 1.96, 1.92, 1.9, 1.88, 1.96, 1.89, 1.95, 1.95, 1.86, 1.92, 1.91, 1.91, 1.94, 1.91, 1.85, 1.94, 1.96, 1.91, 1.97, 1.94, 1.92, 1.98, 1.9, 2.03, 2.03, 1.9, 1.97, 1.99, 1.94, 1.9, 1.98, 1.91, 1.92, 1.9, 1.99, 1.96, 1.92, 1.99, 1.99, 1.98, 1.96, 1.97, 1.93, 1.92, 1.94, 1.99, 1.96, 1.98, 0.96, 0.98, 0.95, 0.95, 0.95, 0.92, 0.93, 0.95, 0.95, 0.97, 0.96, 0.99, 0.96, 0.97, 0.97, 0.97, 0.97, 0.98, 0.98, 0.98, 0.96, 0.97, 0.9, 1.08, 0.97, 1.04, 0.98, 1.03, 0.99, 0.98, 1.06, 0.96, 0.97, 0.97, 0.98, 0.93, 0.93, 0.97, 1.0, 1.14, 1.09, 1.03, 1.13, 1.18, 1.16, 1.23, 1.22, 1.17, 1.48, 1.33, 1.33, 1.17, 1.6, 1.2, 1.2, 1.2, 1.15, 1.56, 1.3, 1.23, 1.2, 1.17, 1.13, 1.16, 1.19, 1.2, 1.21, 1.17, 1.2, 1.16, 1.48, 1.19, 1.22, 1.15, 1.44, 1.13, 1.16, 1.17, 1.16, 1.14, 1.21, 1.13, 1.43, 1.29, 1.22, 1.28, 1.14, 1.45, 1.36, 1.37, 1.45, 1.21, 1.19, 1.19, 1.18, 1.2, 1.16, 1.15, 1.16, 1.2, 1.19, 1.14, 1.16, 1.18, 1.21, 1.1, 1.17, 1.2, 1.14, 1.16, 1.16, 1.17, 1.16, 1.24, 1.14, 1.22, 1.2, 1.2, 1.13, 1.16, 1.14, 1.15, 1.16, 1.15, 1.15, 1.17, 1.23, 1.22, 1.15, 1.18, 1.18, 1.21, 1.29, 1.2, 1.3, 1.38, 1.26, 1.17, 1.2, 1.3, 1.21, 1.28, 1.18, 1.27, 1.18, 1.21, 1.19, 1.35]

```

[코드 25] 제품 두께 파일 불러오기 예제

- 두께 데이터가 저장된 newlist 변수를 이용하여, 양품과 불량품을 판단하고 label을 만든다. thres1 과 thres2 에 기준값을 정의하고, for 문과 if 문, 논리 연산을 이용하여 양품과 불량품을 판단한다.

```

thres1 =0.8
thres2 =1.5

label = np.zeros((len(newlist),1))
num =0

for val in newlist:

    if val<thres1 :
        label[num] =0
    elif thres1<val<thres2 :
        label[num] =1
    elif val>thres2 :
        label[num] =0
    num = num +1

label

```

```
array([[1.],  
       [1.],  
       [1.],  
       [1.],  
       [1.],  
       [1.],  
       [1.],  
       [1.],  
       [1.],  
       [1.],  
       [1.],  
       [1.],  
       [1.],  
       [1.],  
       [1.],  
       [1.],  
       [1.],  
       [1.],  
       [0.],  
       [1.],  
       [0.],  
       [0.],  
       [0.],  
       [1.],  
       [1.],  
       [0.],  
       [0.],  
       [0.],...]
```

[코드 26] threshold를 통해 제품 label 결정 예제

[단계⑤] 학습/평가 데이터 분리 및 서포트 벡터 머신 학습/평가

이 단계에서는 전처리 데이터를 랜덤하게 4등분하여 3/4은 훈련에, 1/4는 테스트에 사용하려 한다. 이를 위해, 2차 가공 데이터를 랜덤하게 나누는 예시를 제공하고, 나누어진 데이터로 서포트 벡터 머신을 학습하고 결과를 확인하는 과정을 제공한다.

⑤-1. 서포트 벡터 머신 학습을 위한 데이터 셔플

```
import warnings # 경고문을 꺼주는 코드입니다  
warnings.filterwarnings(action='ignore') # 경고문을 꺼주는 코드입니다  
from sklearn.svm import SVC  
  
datas = np.hstack((processed_vectors,label)) # 데이터를 랜덤으로 섞기 전 라벨과 데이터 결합  
wrong_guess_stack = 0
```

[코드 27] 3:1 학습 및 검증을 위한 package import 및 데이터 세팅 예제

- 우선, 서포트 벡터 머신을 사용하기 위하여 sklearn의 svm 함수를 약어인 SVC로 import 한다. 데이터를 랜덤으로 섞기 전에, hstack 함수를 사용하여 423x80인 전처리 데이터에 423x1 label 벡터를 결합하여 423x81 벡터를 만들어 준다. wrong_guess_stack 변수는 이후 오분류 결과를 저장하기 위하여 먼저 선언해 주었다.
 - 아래 코드 28 - 30 은 설명을 위해 for 문을 나누어 두었다. 실습시에는 코드 32를 참고하여 사용하기 바란다.

⑤-2. 서포트 벡터 머신 학습

```
for i in range(0,100):
    np.random.shuffle(datas) # 데이터를 랜덤 셔플
    
    sX = datas[0:300,0:80] # 훈련용 데이터
    sXt = datas[301:423,0:80] # 테스트용 데이터
    sY = datas[0:300,80:81] # 훈련용 라벨
    sYt = datas[301:423,80:81] # 테스트용 라벨
    
    model = SVC(kernel='rbf') # 비선형 SVM kernel 모델 설정
    model.fit(sX,sY) # SVM model 훈련
```

[코드 28] 학습용 데이터와 검증용 데이터의 분리 및 SVM 훈련 예제

- 훈련을 100회 반복하여 서포트 벡터 머신의 평균 성능을 확인하기 위하여 for문을 셔플 단계 이전에 추가하여 주었다. np.random.shuffle 함수를 이용하여 데이터를 셔플한다. 해당 함수는 변수에 직접 작용해 인풋이 된 데이터를 셔플하여 반환하고, for문 내에서 매 시도마다 다른 방식으로 셔플된 데이터를 제공한다.
- 셔플 된 데이터에서 데이터와 라벨을 분리하고, 학습용 데이터와 테스트용 데이터를 나누어 준다. SVC 함수를 이용해 학습에 사용할 비선형 커널을 선택하고, fit 함수를 이용하여 서포트 벡터 머신을 훈련한다. 다른 비선형 커널인 다항 (polynomial) 커널은 'poly', 선형 커널은 'linear' 옵션으로 사용이 가능하다. 훈련용 데이터인 sX와 훈련용 라벨인 sY인 라벨을 이용하여 서포트 벡터 머신을 학습시킨다.

⑤-3. 학습된 서포트 벡터 머신을 이용한 분류

```
s_result = model.predict(sXt)
```

[코드 29] 서포트 벡터 모델을 이용한 예측 코드

- predict 함수를 이용하여 테스트용 데이터를 분류한다. 해당 결과는 아래와 같으며, 함수 사용에 관련된 오류 메세지가 뜨더라도 분류에는 문제가 없다.

⑤-4. 서포트 벡터 머신의 성능 평가

```
wrong_guess = (np.sum(np.abs(s_result-np.transpose(sYt)))) # 틀린 예측 개수
wrong_guess_stack = wrong_guess_stack + wrong_guess
```

[코드 30] 성능 확인용 오분류 개수 저장 예제

- 서포트 벡터 머신의 성능을 평가하기 위하여, 틀린 분류의 개수를 저장한다. 분류 결과는 1x122, 데이터 라벨은 122x1 이기 때문에 transpose로 사이즈를 맞춘다. 분류 결과가 0과 1이기 때문에, 요소간 차를 계산하면, 오분류 데이터는 -1 혹은 1 값

을 가지게 된다. 절대값 함수인 abs를 이용하여, 모든 오분류 데이터가 1 값을 가질 수 있게 하고, sum 함수를 이용하여 합쳐, 틀린 예측의 개수를 저장한다.

- 매 훈련마다 오분류 데이터 개수가 wrong_guess_stack 변수에 축적되며 훈련이 마무리 되고 난 뒤 아래 코드로 평균 오분류율을 확인한다.
- 훈련 데이터가 랜덤으로 셔플되기 때문에, 평균 오분류율 값이 매 시도마다 조금씩 달라질 수 있다.

```
(wrong_guess_stack/100/122)*100 # 100 = for 문 반복 횟수, 122 = test set size, *100 = 백분률
```

```
4.563934426229509
```

[코드 31] threshold를 통해 제품 label 결정 예제

```
import warnings # 경고문을 꺼주는 코드입니다
warnings.filterwarnings(action='ignore')
# SVM 훈련 코드입니다.

from sklearn.svm import SVC

datas = np.hstack((processed_vectors,label)) # 데이터를 랜덤으로 섞기 전 라벨과 데이터 결합
wrong_guess_stack = 0

for i in range(0,100):
    np.random.shuffle(datas) # 데이터를 랜덤 셔플

    sX = datas[0:300,0:80] # 훈련용 데이터
    sXt = datas[301:423,0:80] # 테스트용 데이터
    sY = datas[0:300,80:81] # 훈련용 라벨
    sYt = datas[301:423,80:81] # 테스트용 라벨

    model = SVC(kernel='rbf') # 비선형 SVM kernel 모델 설정
    model.fit(sX,sY) # SVM model 훈련
    s_result = model.predict(sXt)

    wrong_guess = (np.sum(np.abs(s_result-np.transpose(sYt)))) # 틀린 예측 개수
    wrong_guess_stack = wrong_guess_stack + wrong_guess

(wrong_guess_stack/100/122)*100 # 100 = for 문 반복 횟수, 122 = test set size, *100 = 백분률
```

[코드 32] 3:1 학습 및 검증 전체코드 예제(RBF)

- 위 코드를 활용하여 열화상 데이터의 커널별 평균 오분류율을 계산하여 보면 다음과 같다. 표의 결과를 바탕으로, rbf 커널을 선택하였다. 검증용 총 데이터 122개에 대한 결과이다.

	linear	polynomial	RBF
오분류 제품 개수	6.37	6.56	5.21
오분류율(%)	5.31	5.37	4.56

[표 3] 선형 및 비선형 커널 테스트 결과

[단계⑥] 실제 상황에 적용 및 결과 분석

- 실제 분석 모델에서는 위의 예제에 사용된 모든 데이터를 SVM 훈련에 사용하였고, RBF 커널을 사용하였다. 검증 테스트에서 Type 1 과 Type 2 의 케이스를 나누어 SVM 성능을 검증하였으며, Type 1 은 양품을 불량품이라고 판단하는 경우, Type 2 는 불량품을 양품이라고 판단하는 경우를 말한다.
- 적외선 카메라 등을 통하여 취득된 이미지에서 포함된 샘플 데이터에서 온도 데이터를 컴퓨터 비전 기술을 통하여 추출하여 이용할 수 있고, 제품 온도 분포에 차이가 생기는 경우라면 품질을 예측할 수 있는 제품 생산 공정에 넓게 활용할 수 있을 것으로 판단된다.

3. 유사 태현장의 「머신비전 AI 데이터셋」 분석 적용

3.1 본 분석이 적용 가능한 제조현장 소개

- 본 분석은 가스 성형 사출 방식을 사용하는 제조업 계열 중 제품 내부 충진의 비율에 따라 사출 뒤 일정시간 동안 제품의 온도 분포가 부위별로 다르게 나타나는 현장에서 적용이 가능하다. 대부분의 가스성형사출 방식을 사용하는 부품들은 제품이 식어야 양품/불량품 판단이 가능하기 때문에, 불량이 발생하더라도 발생 시점에 확인이 불가능하며, 식는 시간 동안 생성된 모든 제품들을 폐기해야하는 단점이 있다. 또한, 제품 내부에 결함이 생기는 경우 파괴검사를 수행해야 하는데, 모든 제품에 파괴 검사를 수행할 수는 없다.
- 본 분석을 적용하면, 일정기간동안 생산된 제품 중 샘플을 취득하여 파괴 검사를 수행하고 불량으로 판별이 나면 앞뒤 제품 전부를 폐기하는 방식에서 벗어나, 매 사출마다 제품의 양품과 불량품 여부를 제품 파괴 없이 파악이 가능해진다. 따라서, 품질 향상 및 파괴 검사로 인해 소모되는 시간과 비용 개선에 실질적 효과를 보일 것으로 기대된다.

3.2 본 「머신비전 AI 데이터셋」 분석을 원용하여 태현장 적용 시, 주요 고려사항

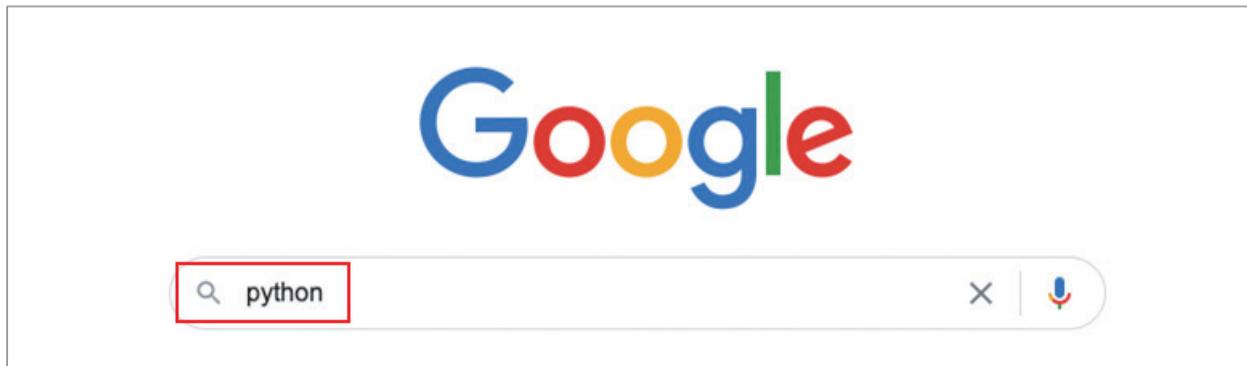
- 본 분석은 열화상 카메라가 제품을 되도록 고정된 장소에서 촬영하여야 한다. 전처리 알고리즘에 어느 정도의 떨림 방지, 위치 변경에 대한 대비책이 포함되어 있으나, 머신 학습을 위해 데이터의 연속성이 필수적임으로 피사체인 제품과 카메라의 촬영 위치가 바뀌어서는 안된다. 또한, 촬영 현장의 온도 등이 급격하게 바뀌는 상황에서는 소프트웨어적으로 알고리즘을 추가적인 상쇄 전처리나 물리적 방법을 사용한 촬영 현장의 온도 유지가 필요하다.



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 official Python website at www.python.org. The search bar at the top contains the word "python". Below the search bar, there are navigation links for All, Images, News, Videos, Books, More, Settings, and Tools. The main content area displays the Python 3.9.0 release information, including the release date (Oct. 5, 2020), and links for Downloads, Documentation, Python For Beginners, Tutorial, and Python Docs.

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 ...

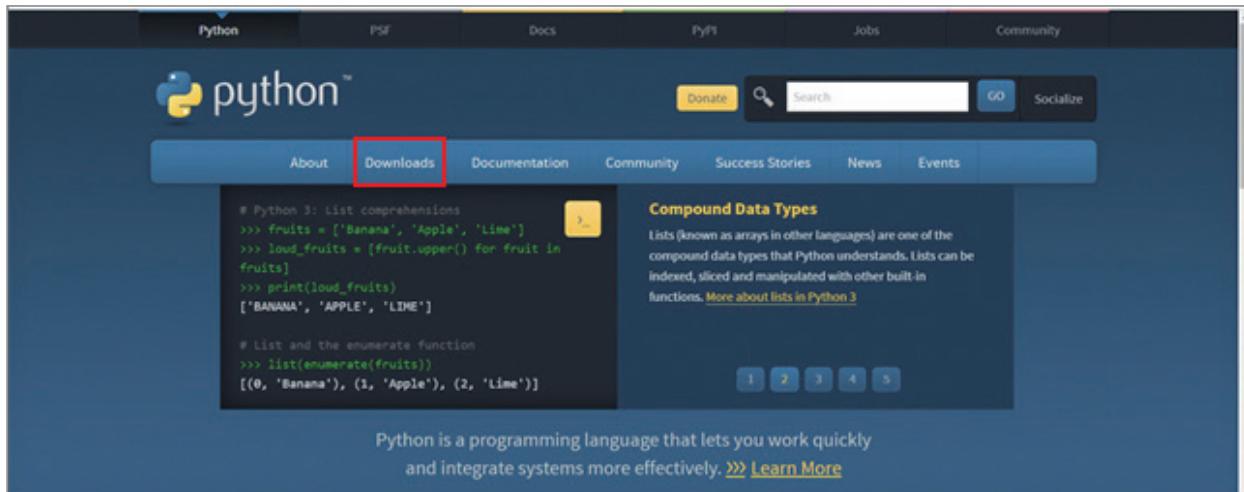
Python For Beginners
BeginnersGuide/Download - Python for Programmers - Books

Python 3.9.0
Python 3.9.0. Release Date: Oct. 5, 2020. This is the stable release ...

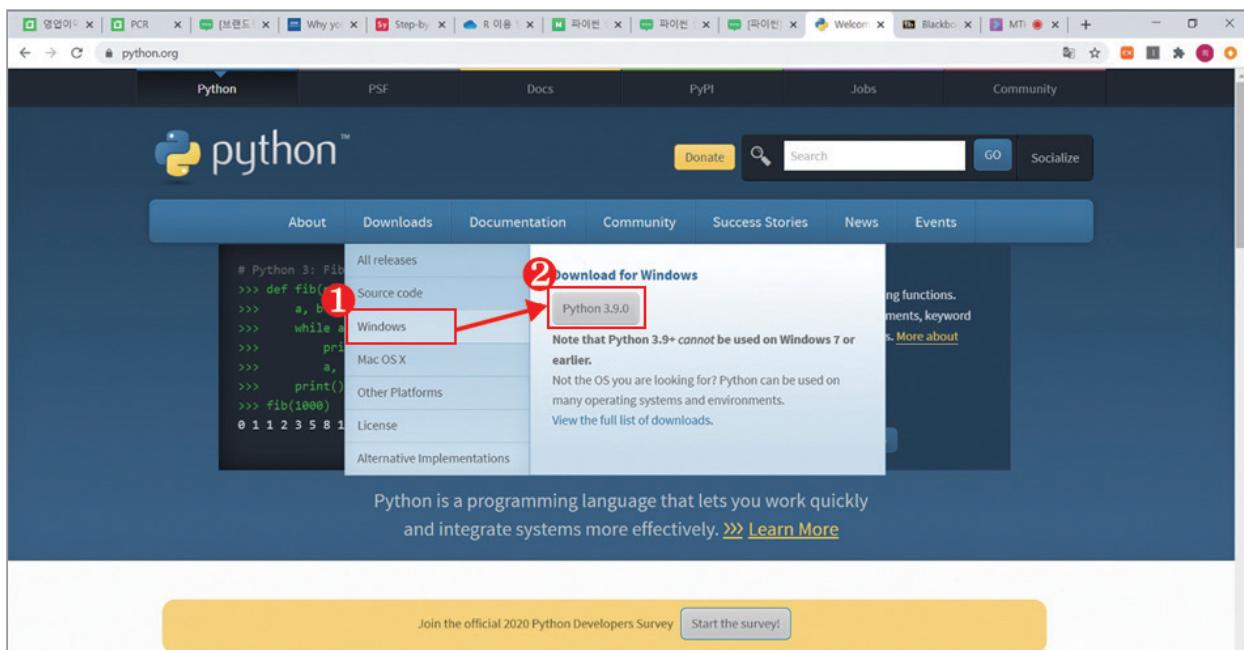
Tutorial
1. Whetting Your Appetite - Lambda - 5. Data Structures

Python Docs
What's new in Python 3.9? or all "What's new" documents since 2 ...

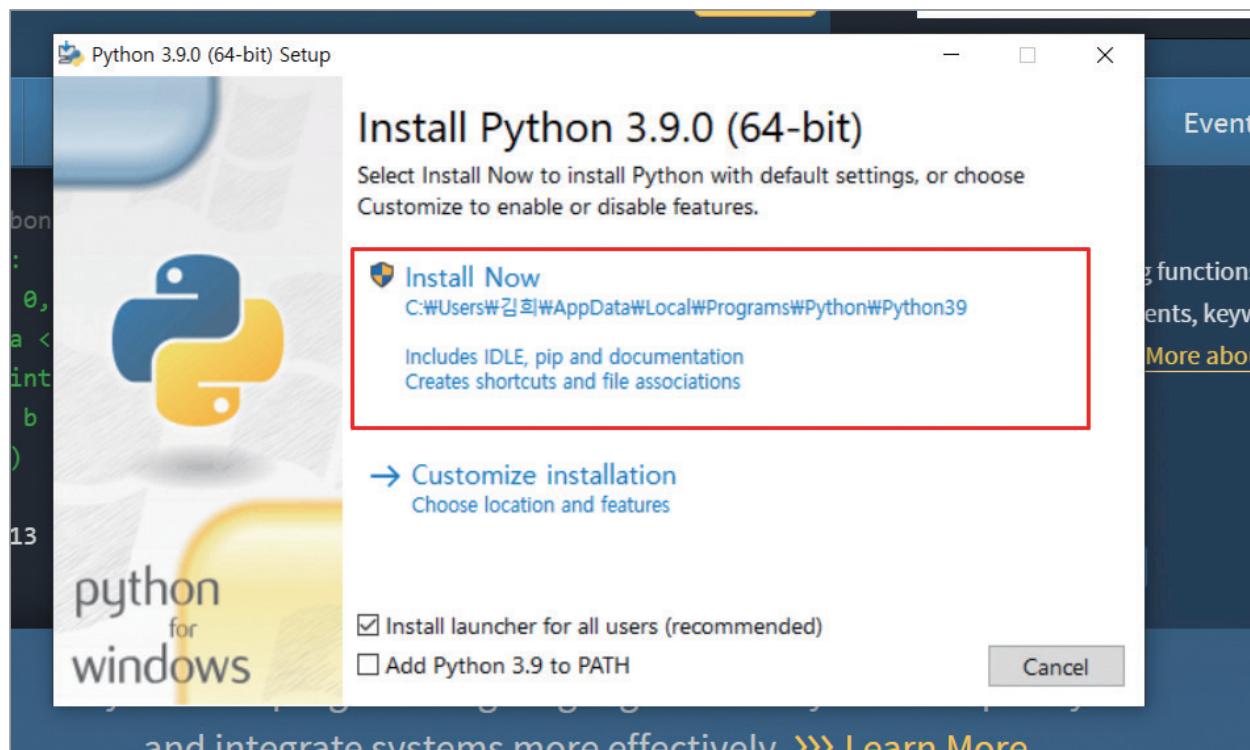
③ 클릭해서 보이는 페이지 정면의, 왼쪽 2번째 'Downloads' 클릭



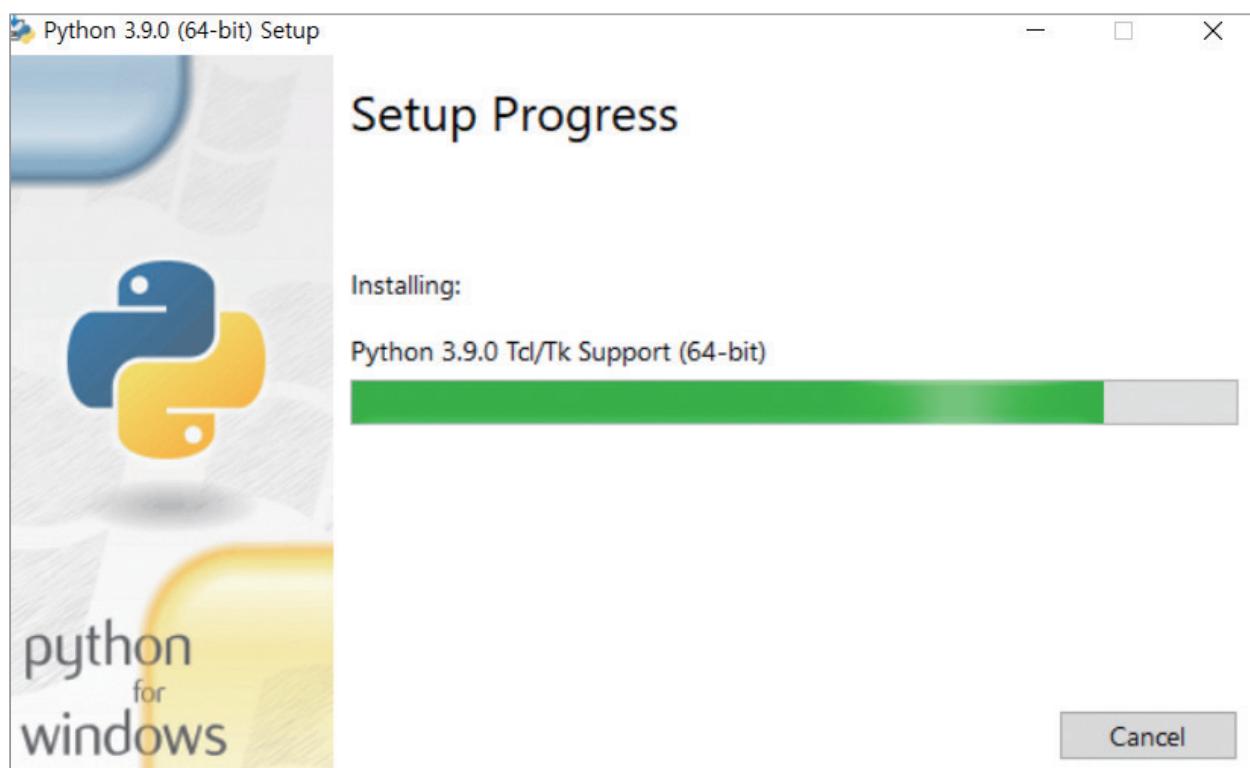
④ 위에서 3번째, Windows 탭을 선택한 후, python3.9.0 다운로드
(python3.9.0은 숫자가 업데이트 될 수 있습니다)



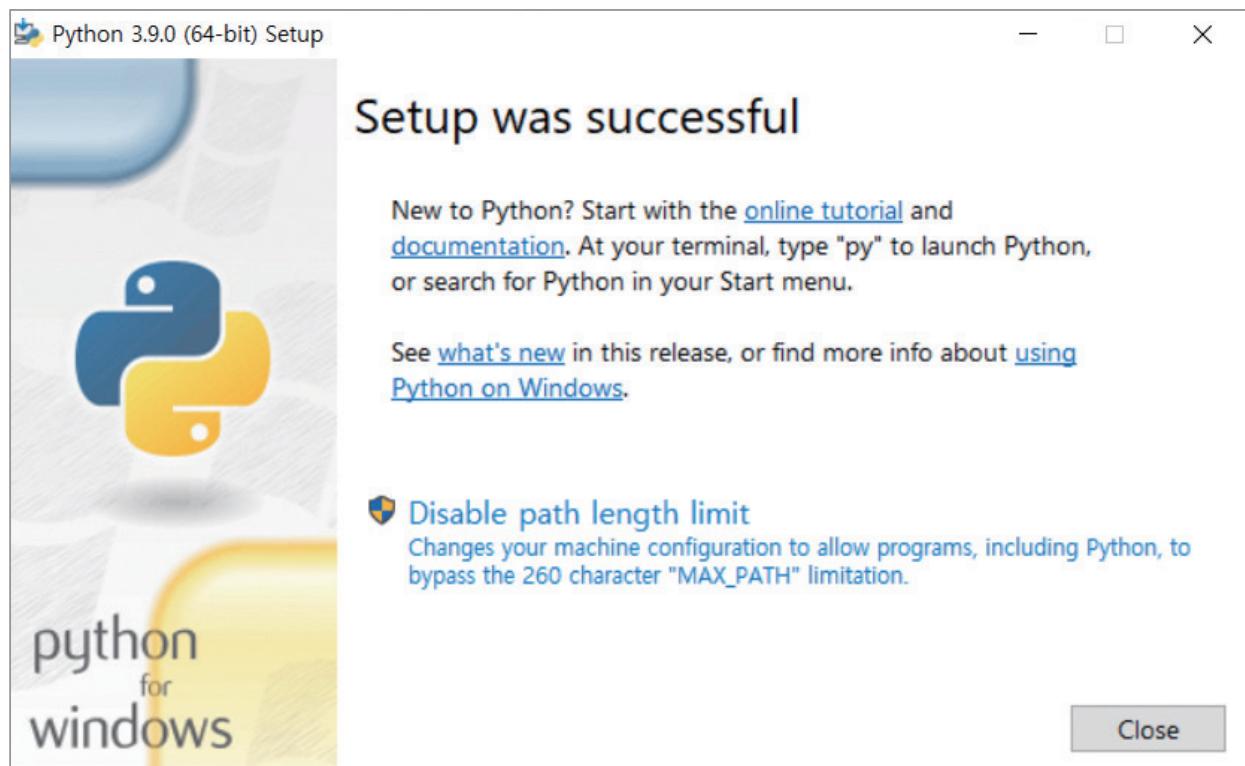
⑤ 아래와 같은 설치창이 뜨면, 'Install Now'를 클릭



⑥ 아래와 같은 설치 진행창이 완료가 될 때까지 유지



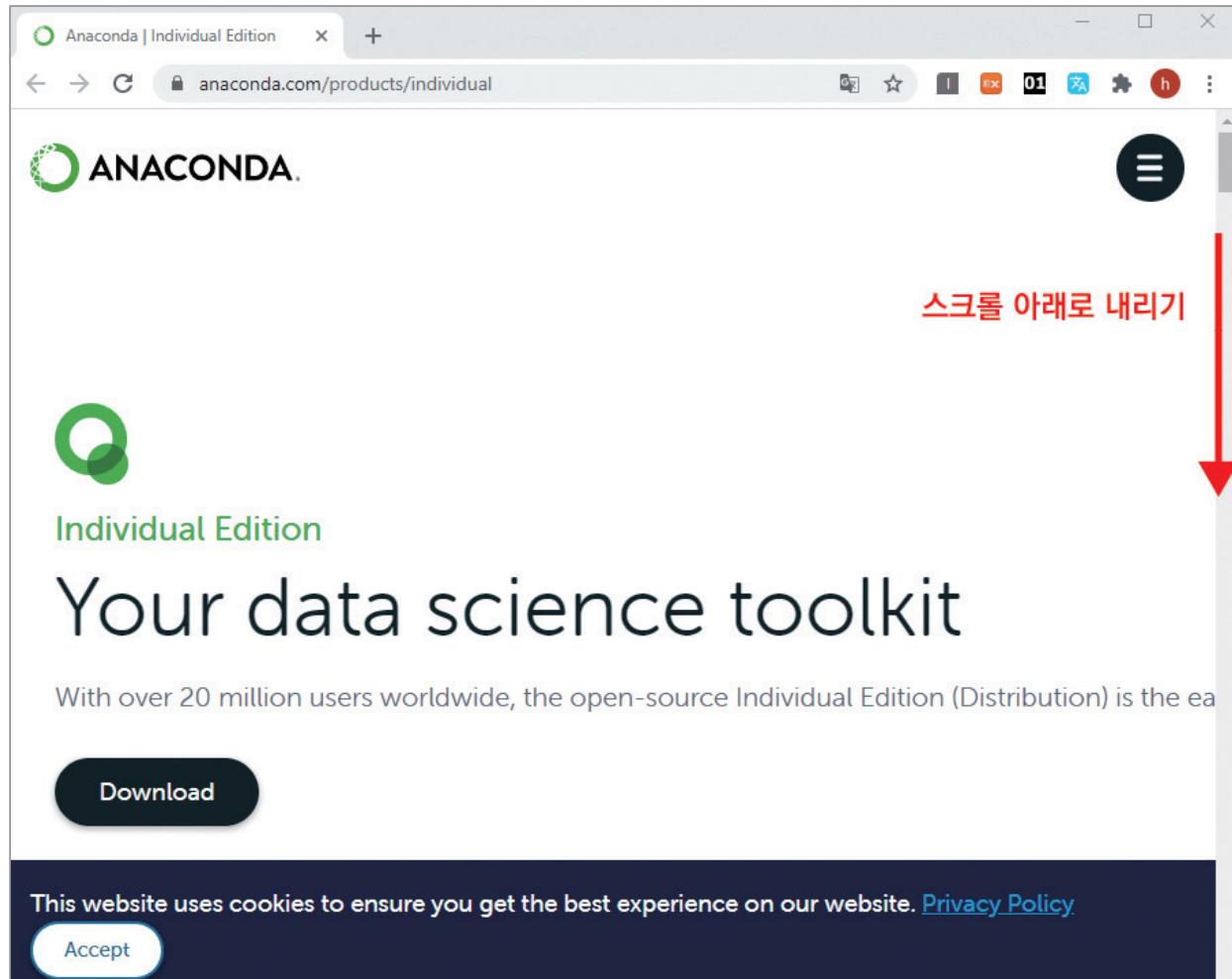
⑦ 완료가 되면 아래와 같은 창이 뜨는 것 확인 후 종료 [설치완료]



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.

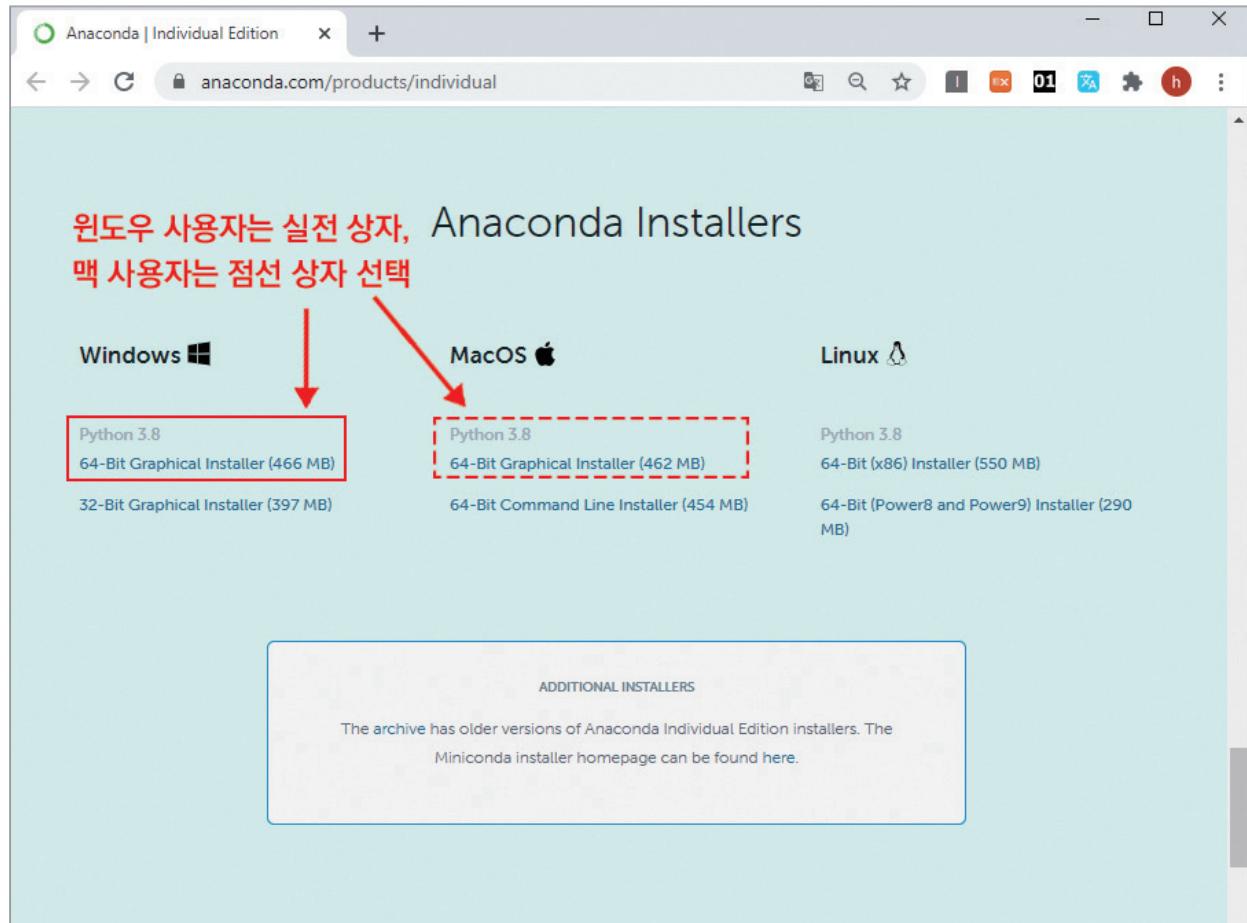
- ① <https://www.anaconda.com/distribution/> 로 접속 후 스크롤 내림



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

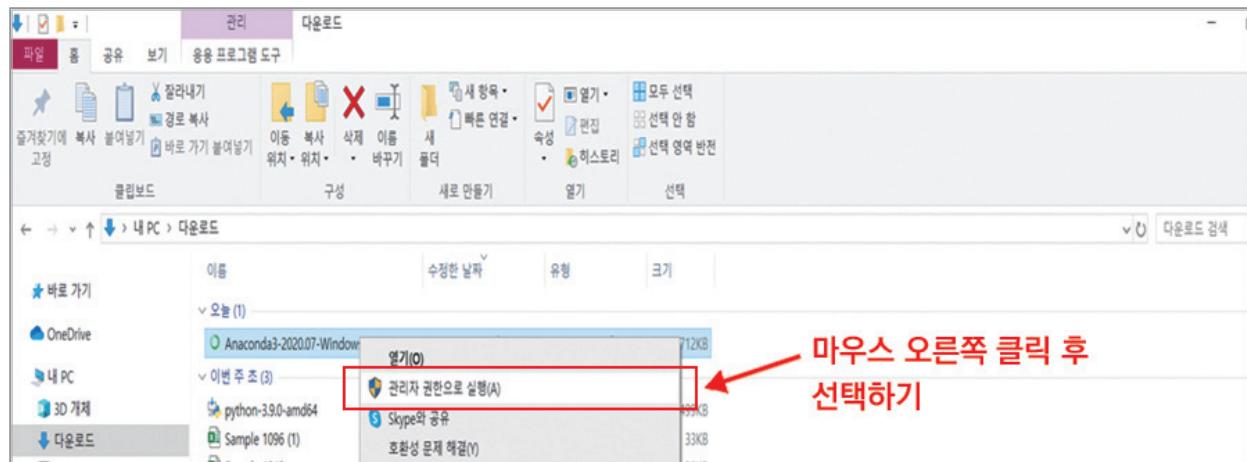
▶ Windows : 64-Bit Graphical Installer

▶ MacOS : 64-Bit Graphical Installer

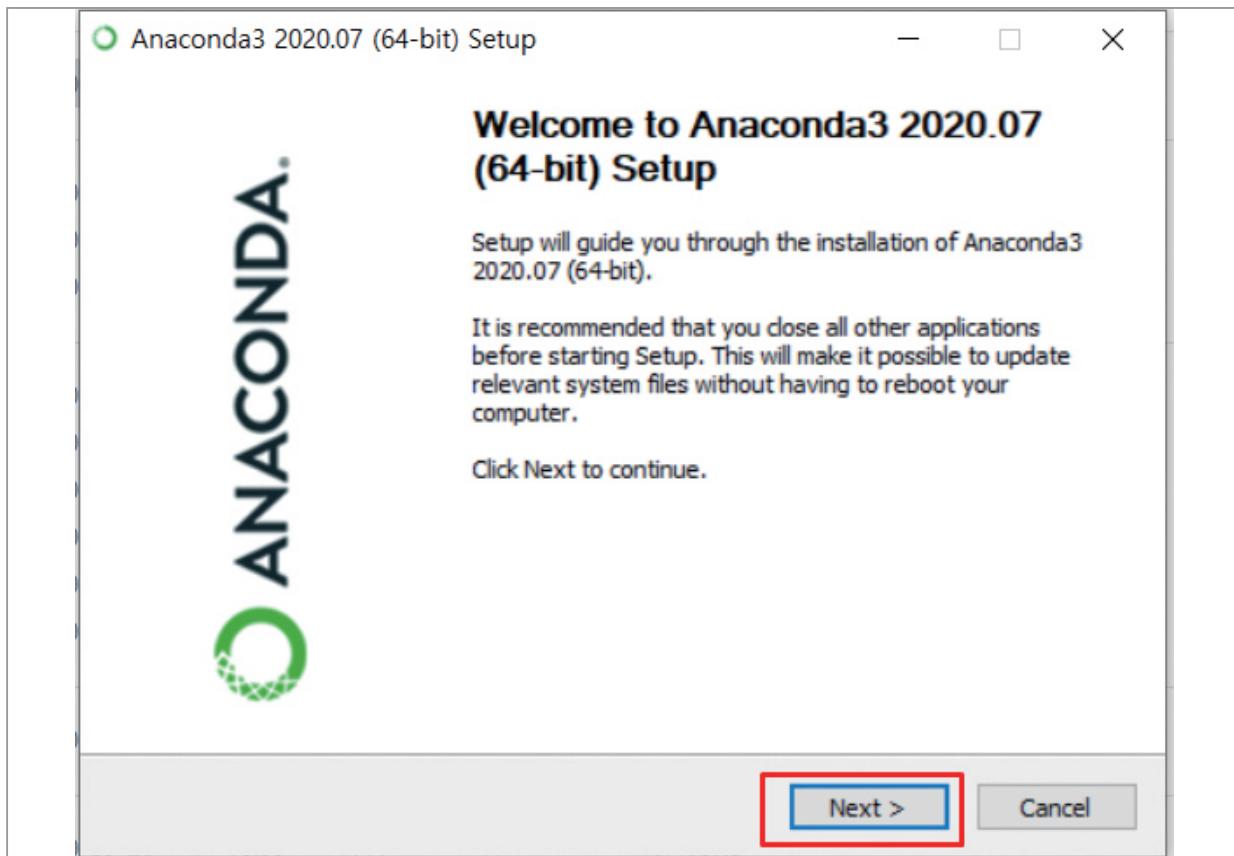


③ 다운을 받은 파일에 가서, 아나콘다 설치 파일 위에서, 마우스 오른쪽을 클릭한 후, 방패모양의 ‘관리자 권한으로 실행’ 선택

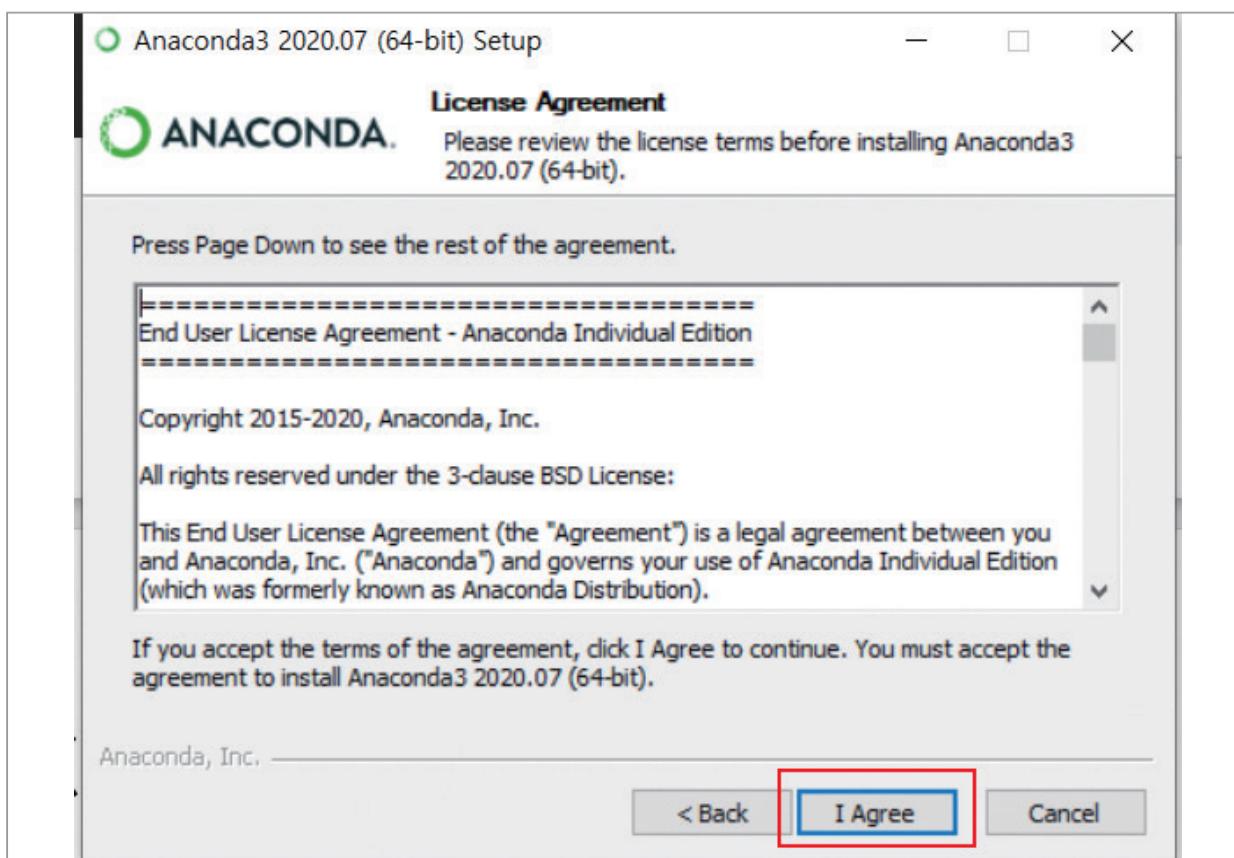
▶ (예) ‘다운로드’ 파일로 아나콘다를 다운 받은 경우



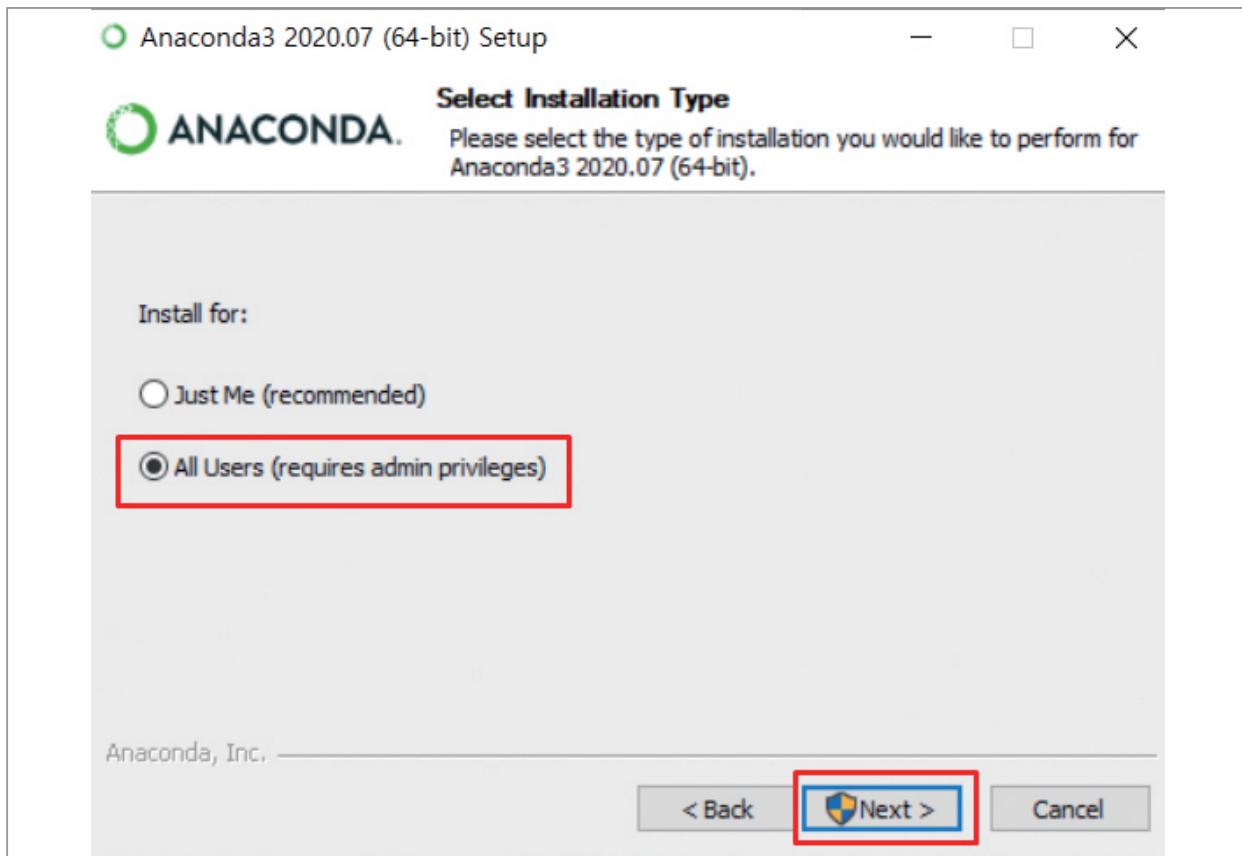
④ 파일을 실행 한 후, 'Next' 버튼을 클릭



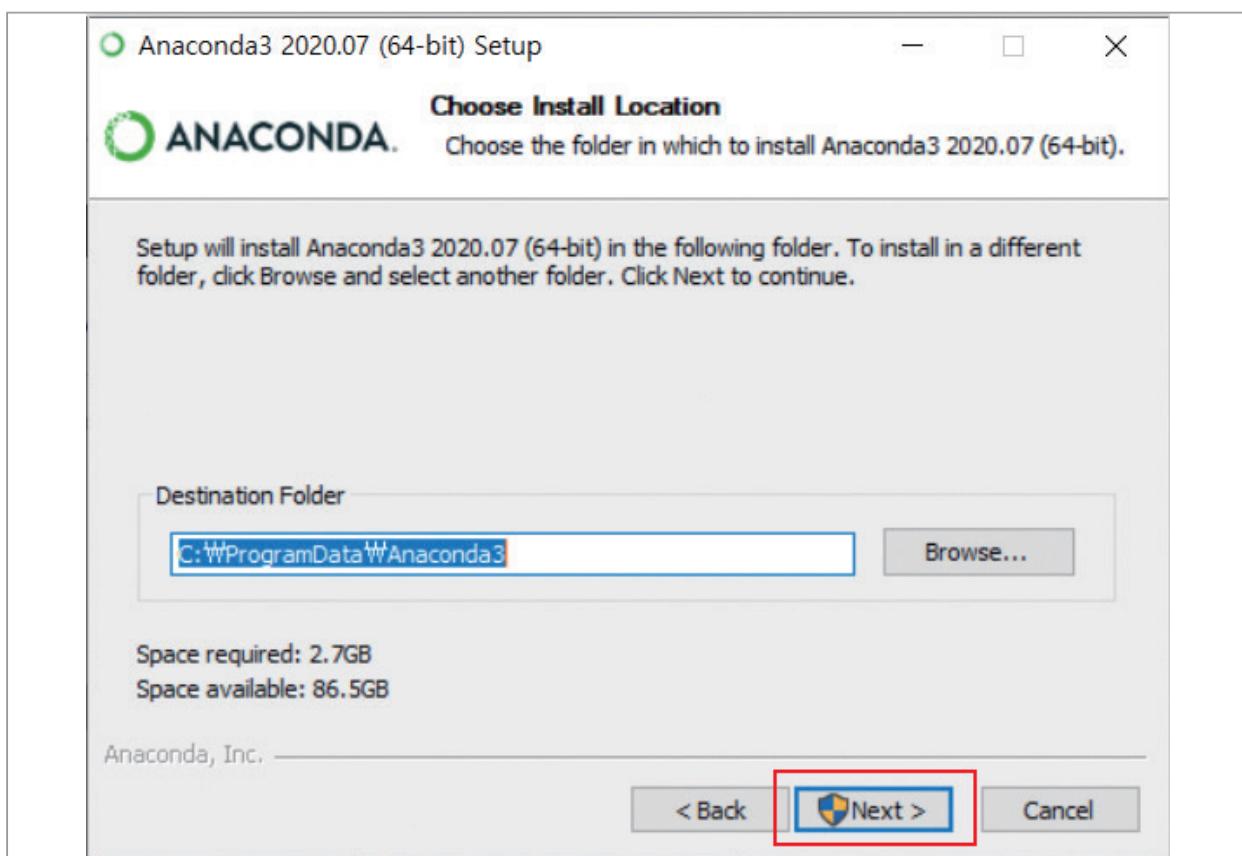
⑤ 다음 창이 나타나면 'I agree'를 선택



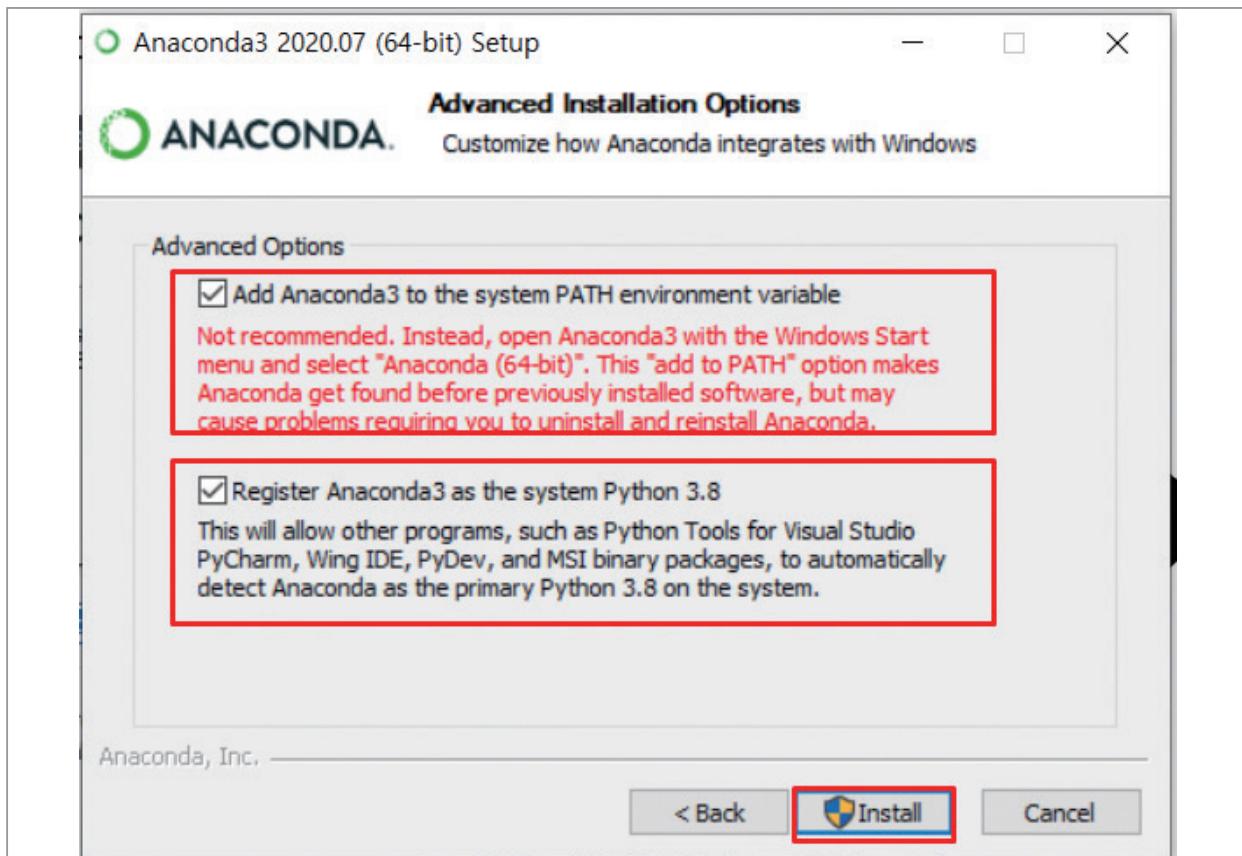
⑥ 셋팅 창이 뜨면 화면의 ‘All Users’를 선택 후 아래의 ‘Next’ 클릭



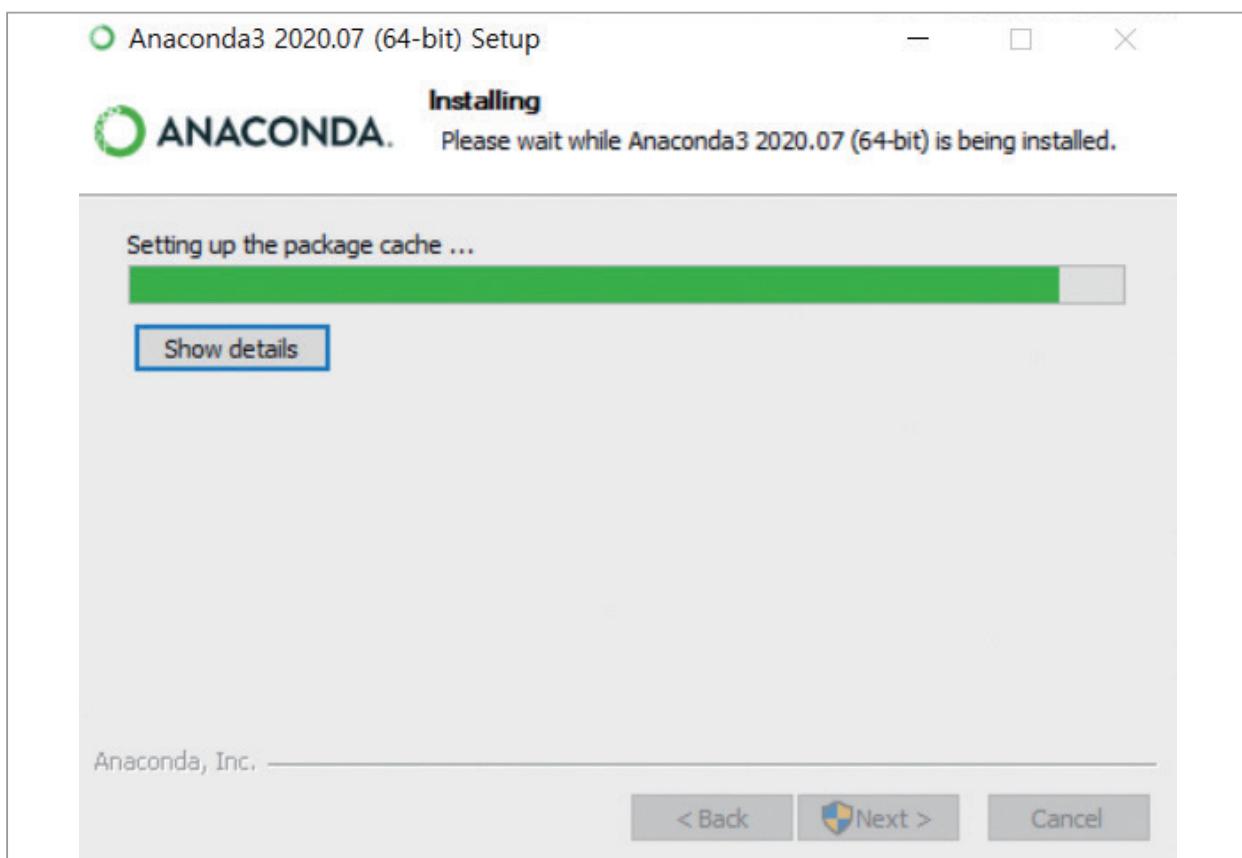
⑦ 다운로드 받을 경로를 물어보는 창이 뜨면, 아래의 ‘Next’ 클릭



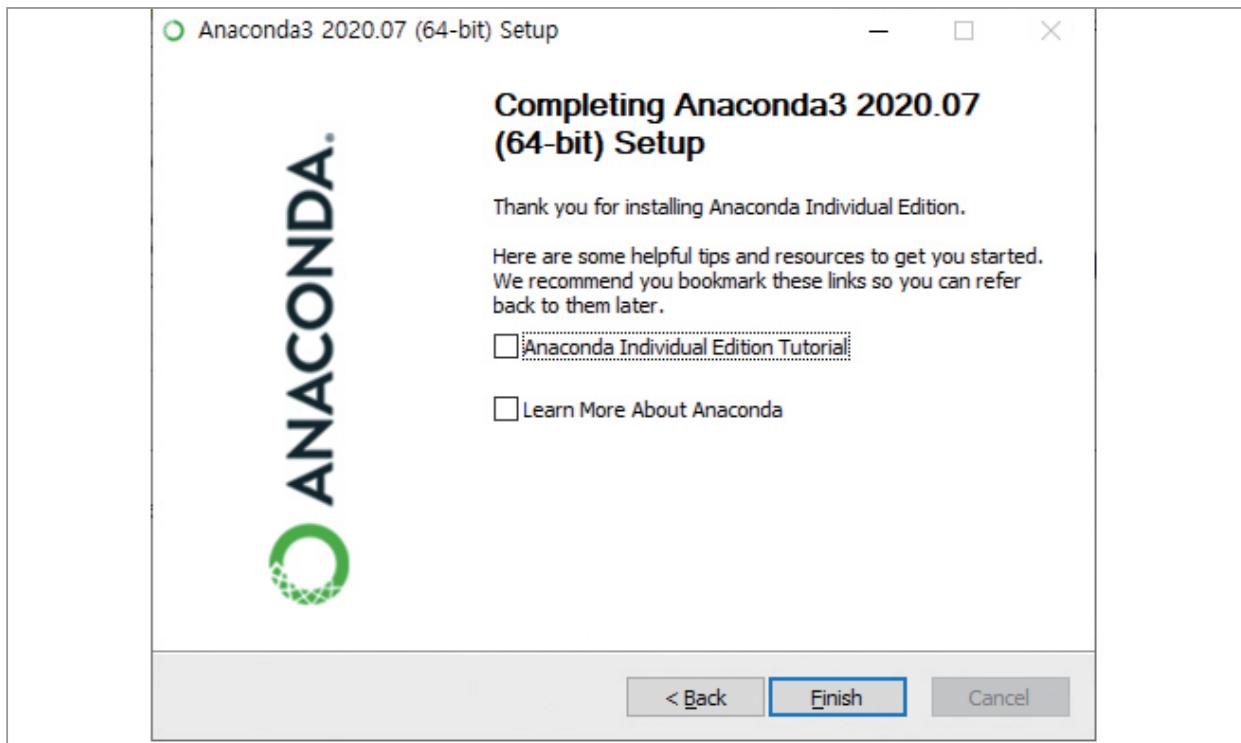
⑧ 고급 옵션 선택창이 뜨면, 아래와 같이 모두 선택 후, ‘Install’ 클릭



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

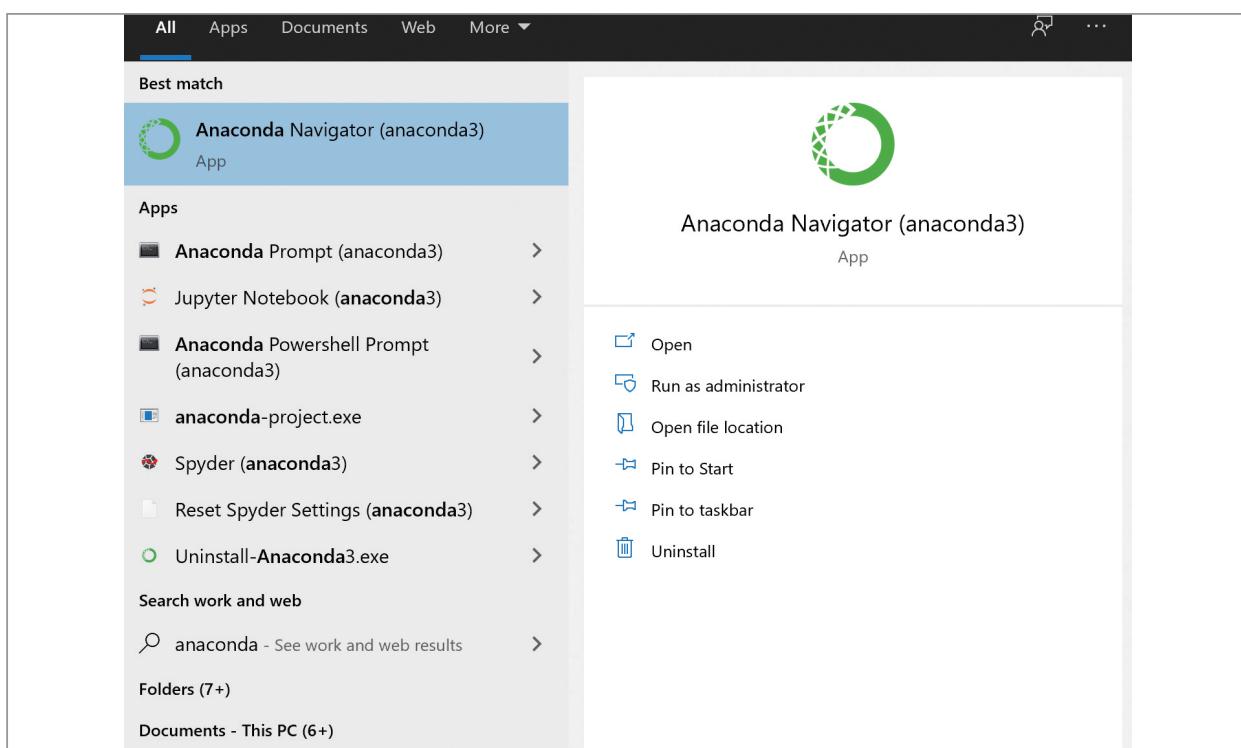


⑩ 마지막 화면에서, 모두 체크 해제한 후, ‘Finish’ 눌러 설치 완료



⑪ 설치 확인하기

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



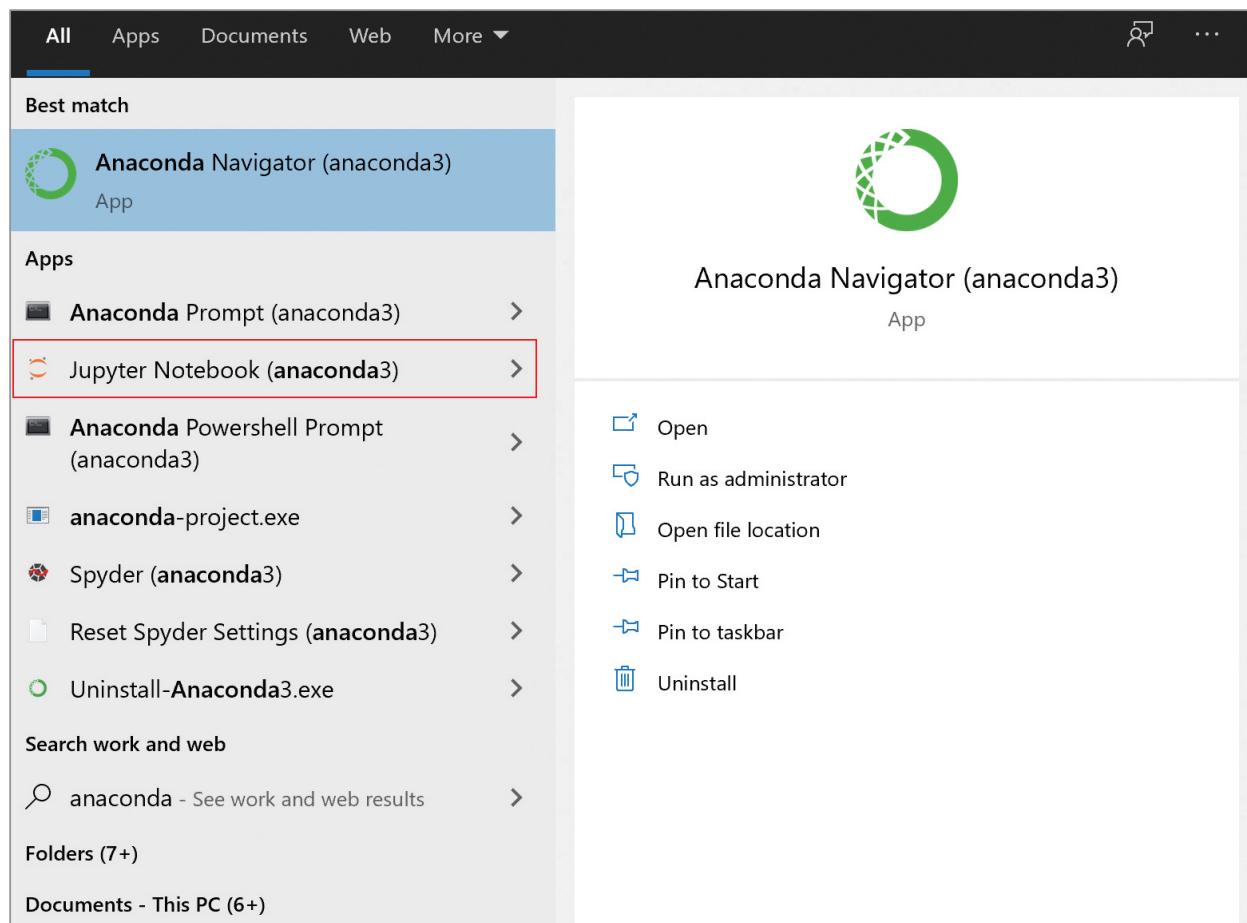
- Anaconda Navigator, Anaconda Prompt, Jupyter Notebook 등 다른 응용 프로그램들도 잘 깔려있는지 확인이 된다면, 설치 완료

3. 주피터 노트북 (Jupyter notebook) 실행



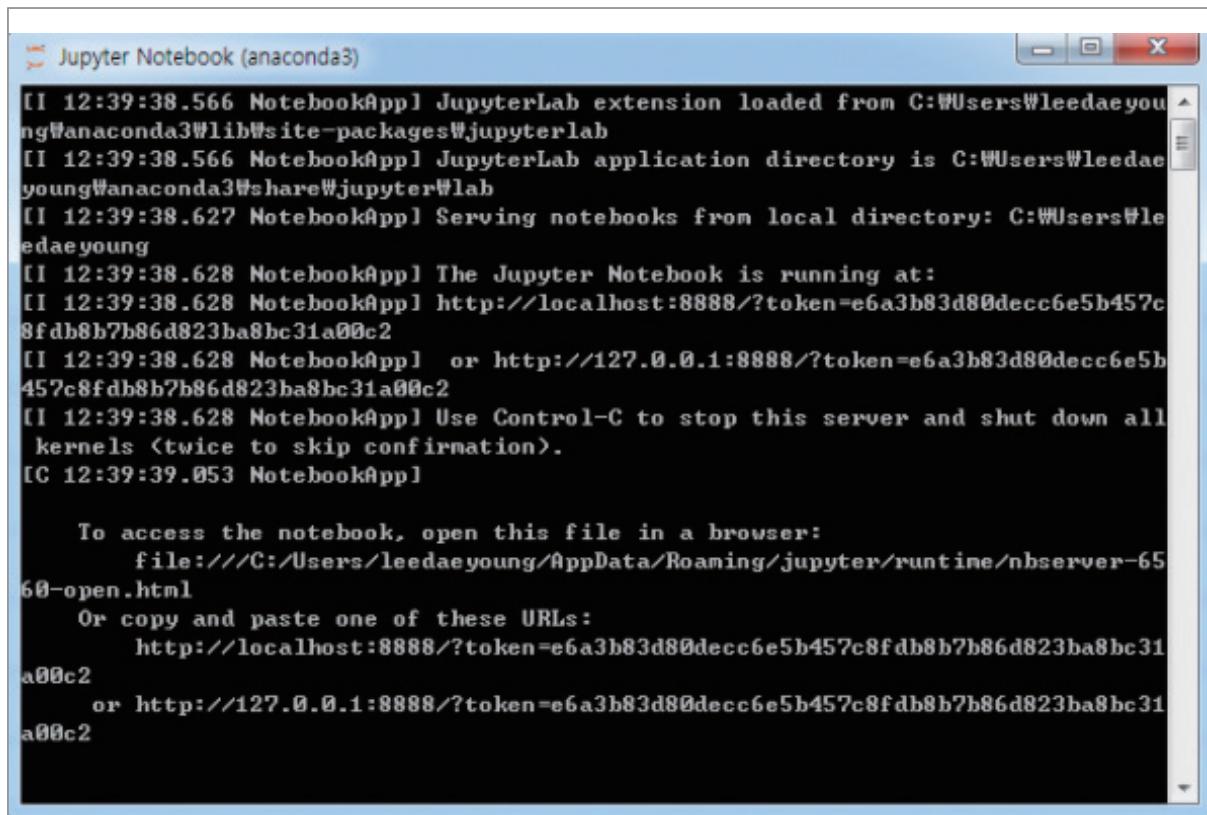
주피터 노트북이란, 실제로 코딩(분석 문장 작성)을 할 수 있는 도구이다. 쉽게 얘기하자면, 문서 도구로 마이크로소프트 사의 ‘Word’ 파일이나, 국내에서는 ‘한글’ 파일 등과 같은 도구라고 생각 할 수 있다. 데이터 분석에 다양한 입력, 실행 도구가 있지만, 본 가이드북에서는 주피터 노트북 을 활용하는 방법을 공유하기로 한다. [부록2]를 참고하여, Anaconda를 설치하였다면, 주피터 노트북(Jupyter notebook) 설치도 확인한다.

- 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.

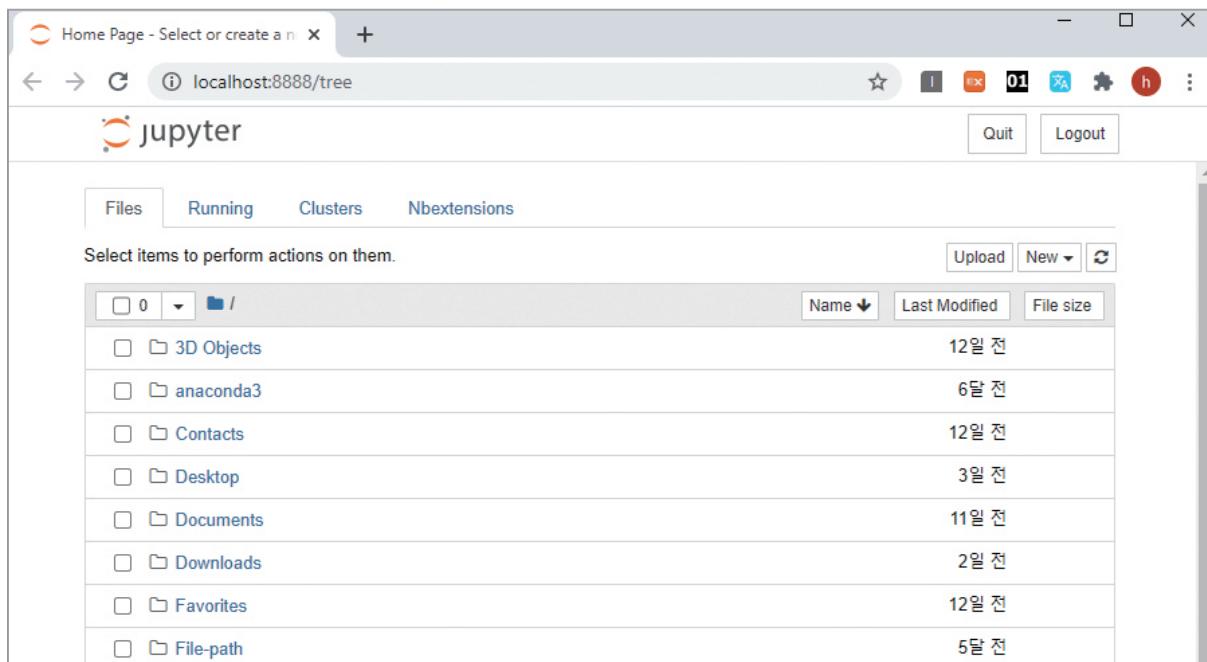
(1) 검은색 배경의 화면은, 주피터 노트북이 실행되는 환경에 대한 상태를 나타내주는 ‘상태 표시창’ 같은 곳. **분석을 하는 동안 종료하면 안된다.**



```
[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
```

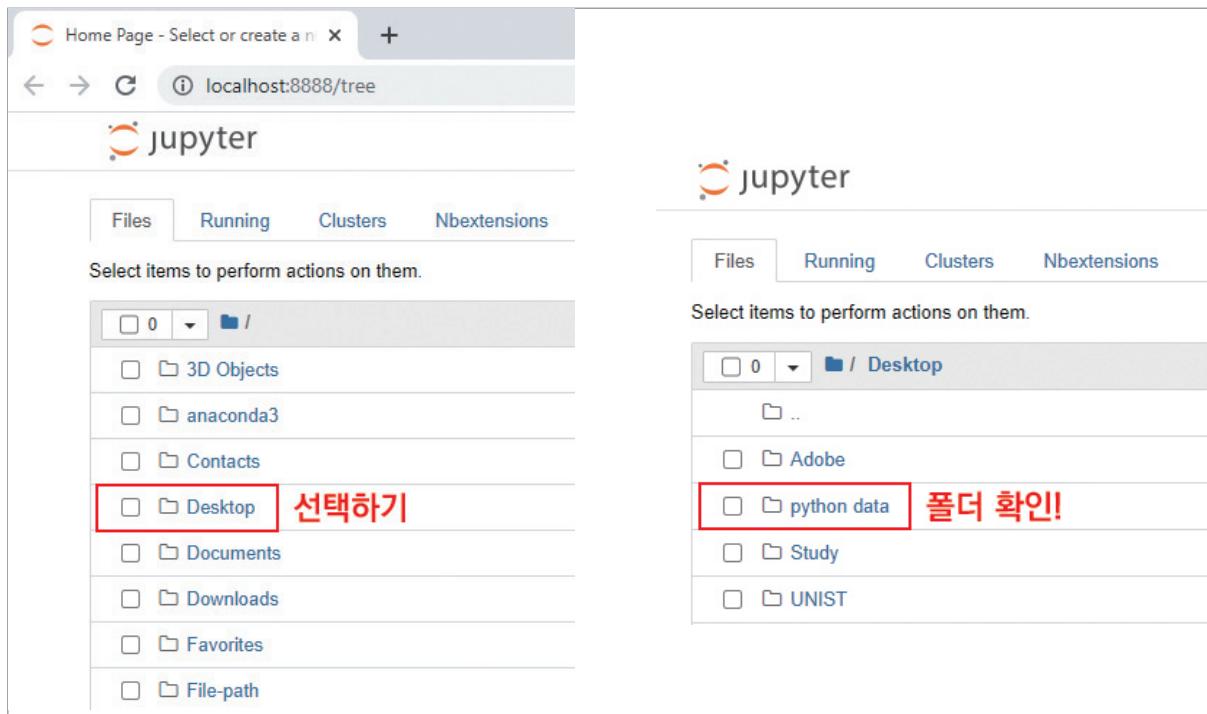
(2) 사용하는 인터넷 프로그램(크롬 / 인터넷 익스플로러)에 주피터 노트북이 열림. (다른 확장 프로그램 사용하고 싶다면, 기본 브라우저를 변경해주어야함). **이 창을 주로 사용.**



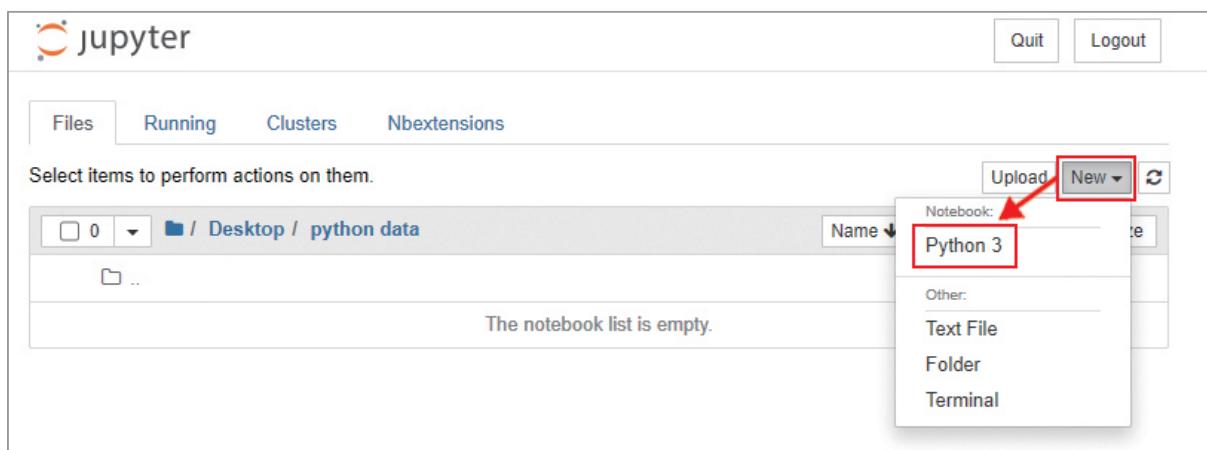
③ 데이터를 저장하고, 불러오고, 분석할 경로의 폴더를 하나 생성

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

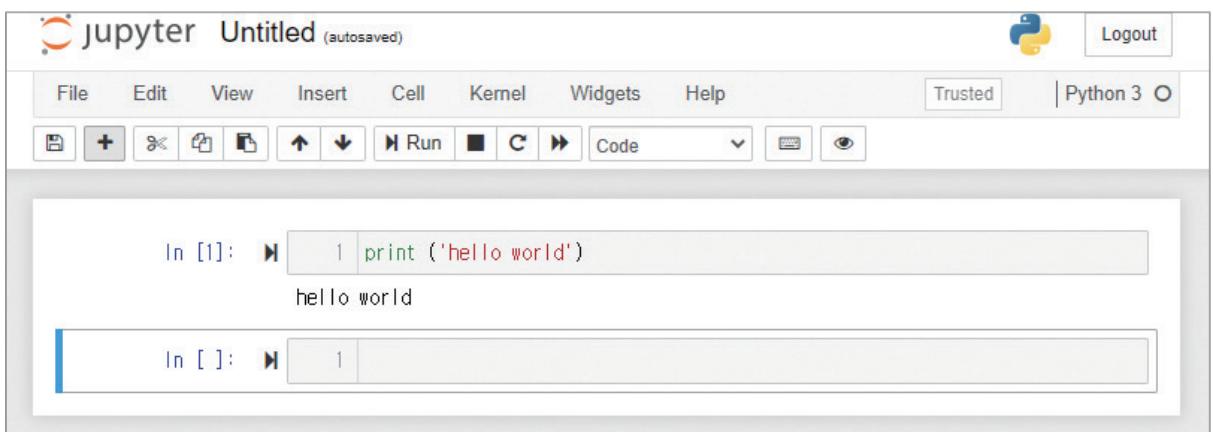
(1) 주피터에서 'python data' 폴더 확인



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



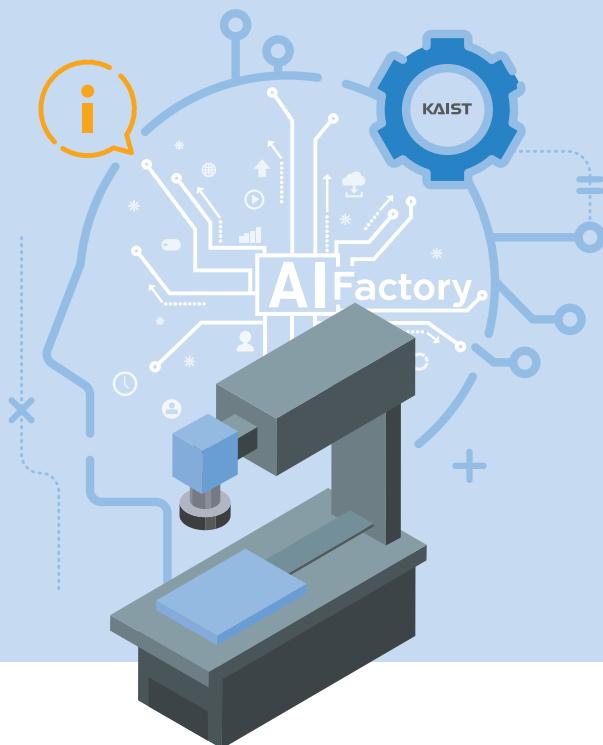
(3) 'hello world' 출력 확인해보기: 보이는 In[] 옆의 회색 창에 `print('hello world')` 입력 후, **shift + Enter** 키 눌러서 실행. : 자동으로 저장되며, 확장자는 '(파일이름).ipynb'로 저장



The screenshot shows a Jupyter Notebook interface with the title bar "jupyter 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 New, Open, Save, and Run. The main area displays two code cells. The first cell, labeled "In [1]:", contains the Python code `print ('hello world')`. The output of this cell, "hello world", is shown in the adjacent text cell, labeled "Out [1]:".

「머신비전 AI 데이터셋」

분석실습 가이드북



중소벤처기업부



스마트제조혁신추진단



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