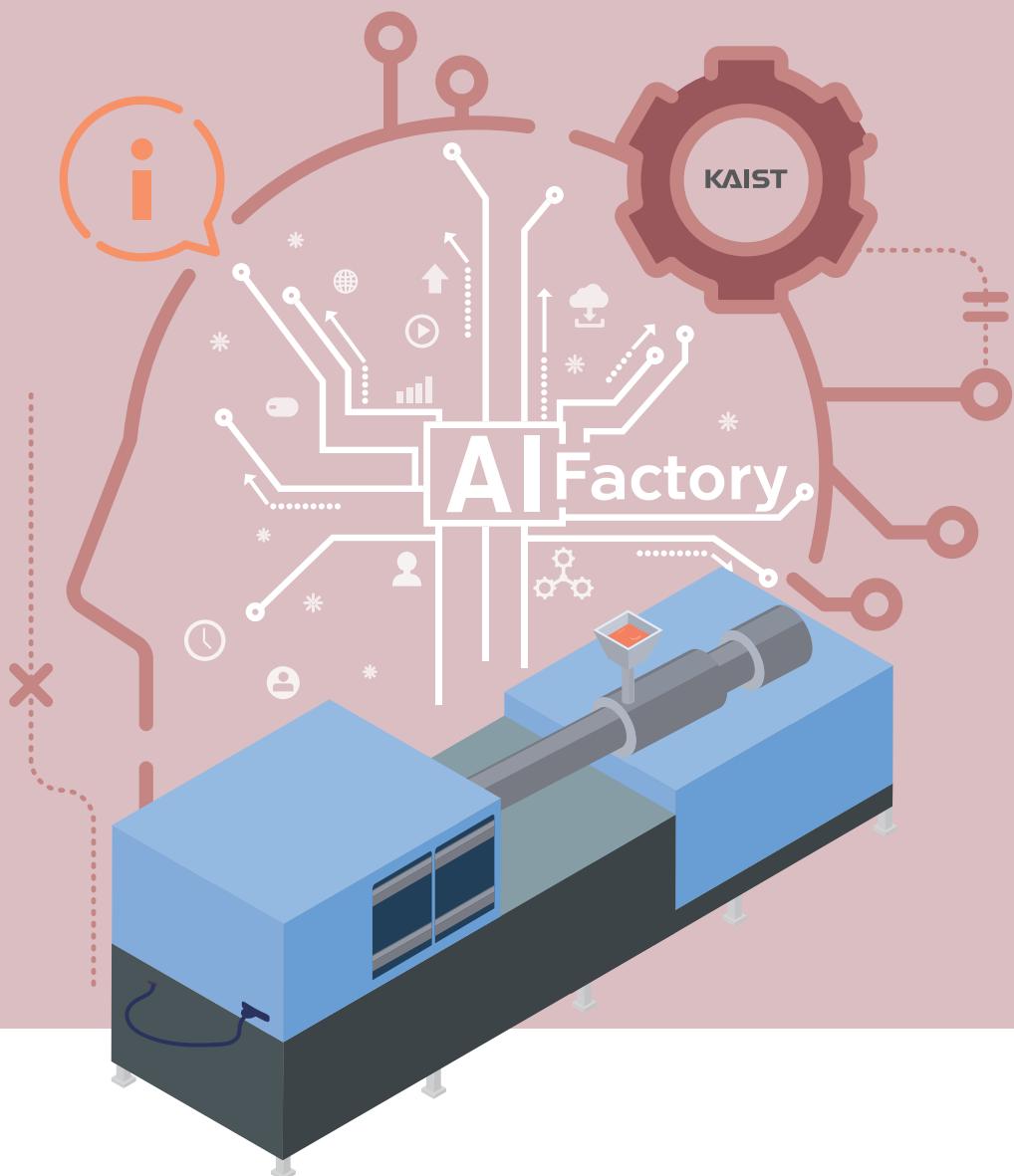
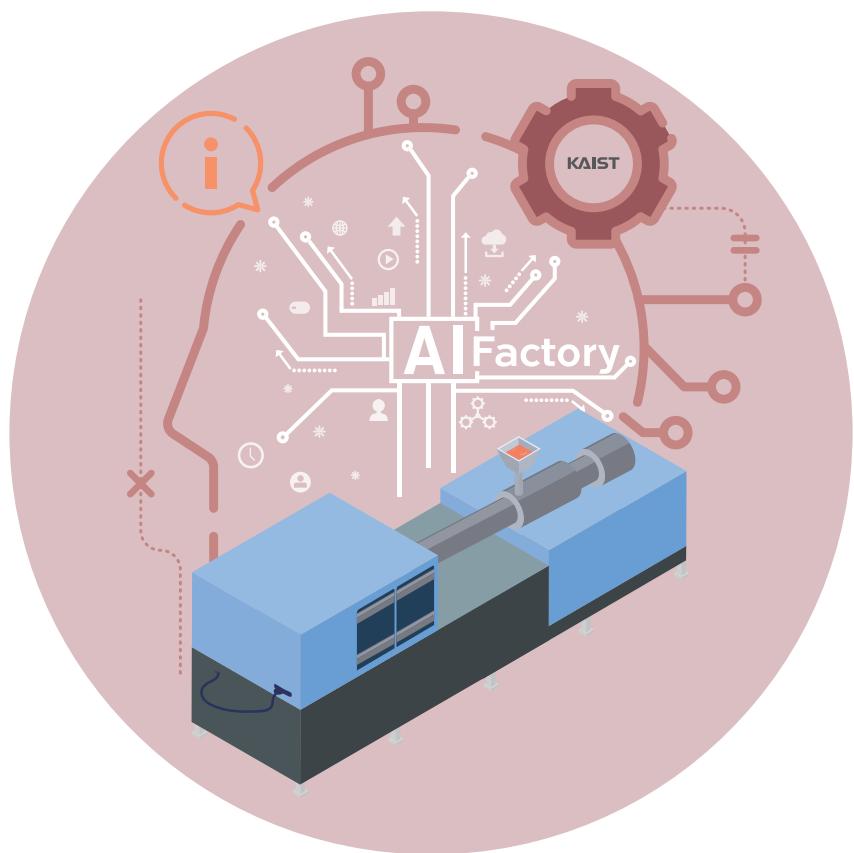


# 『사출성형기 AI 데이터셋』 분석실습 가이드북

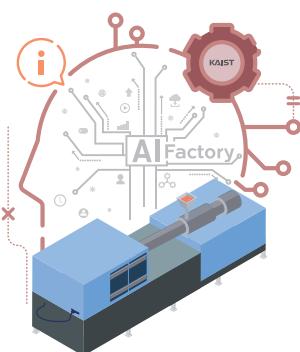


# 「사출성형기 AI 데이터셋」 분석실습 가이드북



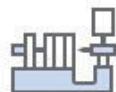
# Contents

<b>1 분석요약</b>	<b>04</b>
<b>2 분석 실습</b>	
<b>1. 분석 개요</b>	<b>05</b>
<b>1.1 분석 배경</b>	<b>05</b>
1) 공정(설비) 개요	
2) 이슈사항(Pain point)	
<b>1.2 분석 목표</b>	<b>15</b>
1) 분석목표	
2) 제조 데이터 정의 및 소개	
3) 제조 데이터 분석 기대효과	
4) 시사점(implication) 요약기술	
<b>2. 분석실습</b>	<b>16</b>
<b>2.1 제조데이터 소개</b>	<b>16</b>
1) 데이터 수집 방법	
2) 데이터 유형/구조	
3) 데이터 (품질) 전처리	
<b>2.2 분석 모델 소개</b>	<b>26</b>
1) AI 분석모델	
<b>2.3 분석 체험</b>	<b>39</b>
1) 필요 SW, 패키지 설치 방법 및 절차 가이드	
2) 분석 단계별 프로세스	
[단계 ①] 라이브러리/데이터 불러오기	
[단계 ②] 데이터 종류 및 개수 확인	
[단계 ③] 데이터 정제(전처리)	
[단계 ④] 데이터 특성 파악	
[단계 ⑤] 학습/평가 데이터 분리	
[단계 ⑥] 오토인코더 모델 구축 (잡음 제거)	
[단계 ⑦] 손실 함수, 옵티마이저 정의 및 모델 컴파일, 훈련	
[단계 ⑧] 임계값 정의 및 예측값과 복원 오차 확인	
[단계 ⑨] 결과 분석 및 해석	
<b>3. 유사 타 현장의 「사출성형기 AI 데이터셋」 분석 적용</b>	<b>86</b>
<b>부 록</b>	
<b>데이터 품질 전처리 [실습 코드]</b>	<b>87</b>
<b>분석환경 구축을 위한 설치 가이드</b>	<b>94</b>



# 「사출성형기 AI 데이터셋」 분석실습 가이드북

- 필요 SW : Python, Anaconda – Jupyter Notebook
- 필요 패키지 : Pandas, matplotlib, scikit-learn
- 분석 환경 : [운영체제] Ubuntu 14.0 이상, [CPU] Intel Xeon 2.3 GHz, [GPU] Tesla K80, [RAM] 13GB
- 필요 데이터 : labeled\_data.csv, unlabeled\_data.csv (1차 가공 데이터), moldset\_labeled.csv (1차 가공 데이터), supervised\_label\_cn7.csv (2차 가공 데이터), moldset\_labeled\_cn7.csv, moldset\_unlabeled\_cn7.csv(2차 가공 데이터 중 cn7 제품에 대한 데이터셋), moldset\_labeled\_rg3.csv, moldset\_unlabeled\_rg3.csv(2차 가공 데이터 중 rg3 제품에 대한 데이터셋)
- 주 관 기관 : 한국과학기술원(KAIST)
- 수 행 기관 : 울산과학기술원(UNIST), 주식회사 이피엠솔루션즈



## 1 분석요약

No	구분	내용				
1	분석 목적 (현장 이슈, 목적)	- 사출 공정에서 발생하는 품질문제를 해결하기 위해 현장에서 생성되는 데이터셋을 AI를 이용하여 분석할 수 있다. 데이터 간의 상관관계를 찾고 다양한 AI 알고리즘을 활용한 사출 공정 불량 예측 모델을 학습할 수 있다.				
2	데이터셋 형태 및 수집방법	- 분석에 사용된 변수명 : Injection_time, Filling_time 외 25개 - 데이터 수집 방법 : Lot 단위 사출 조건 및 일별 불량률에 대한 MES 데이터 이력관리 - 데이터셋 파일 확장자 : csv				
3	데이터 개수 데이터셋 총량	- 데이터 개수 : 886,227개 - 데이터셋 총량 : 35.2 MB				
4	분석적용 알고리즘	<table border="1"><tr><td>알고리즘</td><td>라벨이 있는 데이터와 라벨이 없는 데이터를 모두 훈련에 사용하여 학습된 모델을 다양한 기계학습 모델 (Support vector machine, Deep Neural Network 등)의 결괏값 비교를 통하여 최적의 알고리즘을 선택한다.</td></tr><tr><td>알고리즘 간략소개</td><td>- 오토인코더는 출력값을 입력값과 근사하게 모델을 학습시키는 알고리즘이다. - 랜덤포레스트(Random Forest)는 분류를 위한 다수의 결정 트리(Decision Tree)를 구성하고, 이를 통해 양상불린 결과를 사용하는 트리 기반(가지가 뻗은 나무 모양)의 모델이다. 이런 결정 트리에서는, 트리의 깊이, 잎 개수 등의 파라미터를 조절 할 수 있다. - 심층신경망 (Deep Neural Network)는 분류를 위해 다수의 신경망을 연속적으로 쌓고, 최종적으로 각 클래스별 확률을 출력하도록 각 신경망의 모수를 업데이트하는 모델이다. - 서포트 벡터 머신(Support vector machine, SVM)은 기계 학습의 분야 중 하나로 패턴 인식, 자료 분석을 위한 지도 학습 모델이며, 주로 분류와 회귀 분석을 위해 사용한다. 일반적으로 초평면 또는 초평면들의 집합으로 구성되어 있으며, 좋은 분류를 위해서는 어떤 분류된 점에 대해서 가장 가까운 학습 데이터와 가장 먼 거리를 가지는 초평면을 찾아야 한다. - 가우시안 나이브 베이즈는 베이즈 정리를 적용한 확률적인 분류 알고리즘이다. 모든 특성이 서로 독립적이며, 각 특성이 정규분포를 따르는 연속적인 변수일 경우 사용한다.</td></tr></table>	알고리즘	라벨이 있는 데이터와 라벨이 없는 데이터를 모두 훈련에 사용하여 학습된 모델을 다양한 기계학습 모델 (Support vector machine, Deep Neural Network 등)의 결괏값 비교를 통하여 최적의 알고리즘을 선택한다.	알고리즘 간략소개	- 오토인코더는 출력값을 입력값과 근사하게 모델을 학습시키는 알고리즘이다. - 랜덤포레스트(Random Forest)는 분류를 위한 다수의 결정 트리(Decision Tree)를 구성하고, 이를 통해 양상불린 결과를 사용하는 트리 기반(가지가 뻗은 나무 모양)의 모델이다. 이런 결정 트리에서는, 트리의 깊이, 잎 개수 등의 파라미터를 조절 할 수 있다. - 심층신경망 (Deep Neural Network)는 분류를 위해 다수의 신경망을 연속적으로 쌓고, 최종적으로 각 클래스별 확률을 출력하도록 각 신경망의 모수를 업데이트하는 모델이다. - 서포트 벡터 머신(Support vector machine, SVM)은 기계 학습의 분야 중 하나로 패턴 인식, 자료 분석을 위한 지도 학습 모델이며, 주로 분류와 회귀 분석을 위해 사용한다. 일반적으로 초평면 또는 초평면들의 집합으로 구성되어 있으며, 좋은 분류를 위해서는 어떤 분류된 점에 대해서 가장 가까운 학습 데이터와 가장 먼 거리를 가지는 초평면을 찾아야 한다. - 가우시안 나이브 베이즈는 베이즈 정리를 적용한 확률적인 분류 알고리즘이다. 모든 특성이 서로 독립적이며, 각 특성이 정규분포를 따르는 연속적인 변수일 경우 사용한다.
알고리즘	라벨이 있는 데이터와 라벨이 없는 데이터를 모두 훈련에 사용하여 학습된 모델을 다양한 기계학습 모델 (Support vector machine, Deep Neural Network 등)의 결괏값 비교를 통하여 최적의 알고리즘을 선택한다.					
알고리즘 간략소개	- 오토인코더는 출력값을 입력값과 근사하게 모델을 학습시키는 알고리즘이다. - 랜덤포레스트(Random Forest)는 분류를 위한 다수의 결정 트리(Decision Tree)를 구성하고, 이를 통해 양상불린 결과를 사용하는 트리 기반(가지가 뻗은 나무 모양)의 모델이다. 이런 결정 트리에서는, 트리의 깊이, 잎 개수 등의 파라미터를 조절 할 수 있다. - 심층신경망 (Deep Neural Network)는 분류를 위해 다수의 신경망을 연속적으로 쌓고, 최종적으로 각 클래스별 확률을 출력하도록 각 신경망의 모수를 업데이트하는 모델이다. - 서포트 벡터 머신(Support vector machine, SVM)은 기계 학습의 분야 중 하나로 패턴 인식, 자료 분석을 위한 지도 학습 모델이며, 주로 분류와 회귀 분석을 위해 사용한다. 일반적으로 초평면 또는 초평면들의 집합으로 구성되어 있으며, 좋은 분류를 위해서는 어떤 분류된 점에 대해서 가장 가까운 학습 데이터와 가장 먼 거리를 가지는 초평면을 찾아야 한다. - 가우시안 나이브 베이즈는 베이즈 정리를 적용한 확률적인 분류 알고리즘이다. 모든 특성이 서로 독립적이며, 각 특성이 정규분포를 따르는 연속적인 변수일 경우 사용한다.					
5	분석결과 및 시사점	- 사출 공정에서 생산된 제품의 불량률을 예측하기 위해서 다양한 방법으로 학습된 AI모델의 성능을 비교하고, 최적의 알고리즘을 선택하는 과정을 통해 최종 모델을 도출하였다. - 사출 공정 시 발생되는 데이터를 사용하여 데이터의 가공/전처리 후 개발된 AI 모델을 통해 사출공정을 가진 중소 제조기업현장의 실질적인 품질향상 및 비용 절감을 기대한다.				

## 2 분석 실습

### 1. 분석 개요

#### 1.1 분석 배경

##### 1) 공정(설비) 정의 및 특징

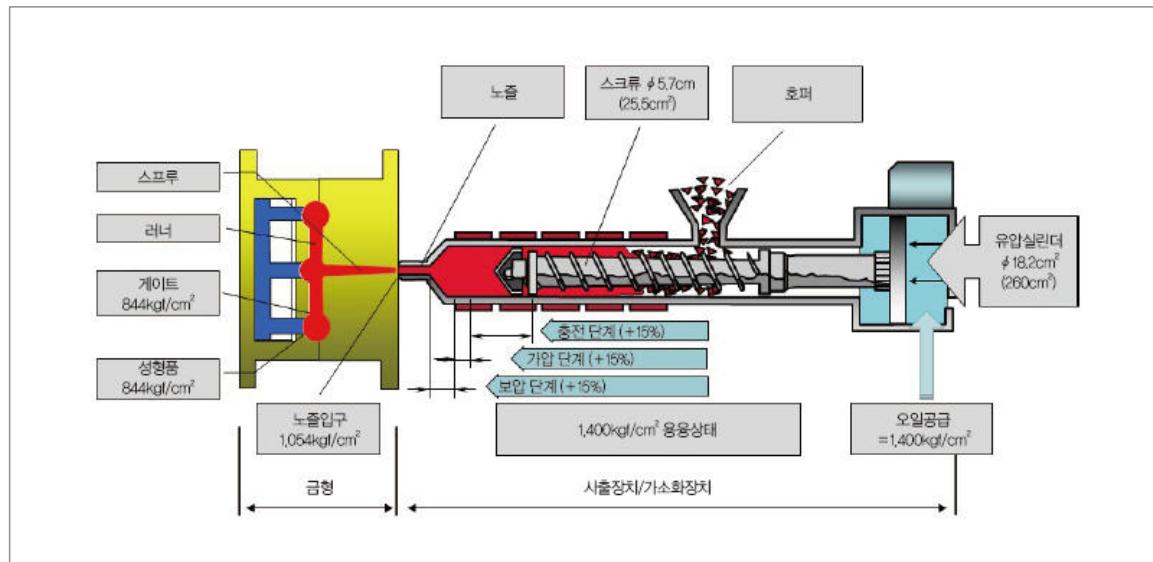
###### • 사출성형 정의

- 사출성형이란 플라스틱 성형법 중의 한 방법으로서, 열가소성 수지를 가열해서 유동 상태로 되었을 때 금형의 공동부(Cavity, 이하 ‘캐비티’)에 가압 주입하여 금형 내에서 냉각시킴으로써, 금형의 공동부에 상당하는 성형품을 만드는 방법이다.

###### • 사출성형 특징

- ① 성형 사이클이 짧고, 성형 능률이 좋다. (대량 생산 가능)
- ② 광범위한 수지의 성형 (열가소성 수지, 열경화성 수지)
- ③ 자동제어화가 가능하다.
- ④ 현재 대형 성형품의 성형 가능하다.
- ⑤ 정밀도가 높아 복잡한 모양 가능하다.

###### • 사출성형 원리



[그림 1] 살균기 데이터 스크린샷

- 사출성형 과정은 크게 구분하여 충전 단계, 가압 단계, 보압 단계의 3단계로 나눌 수 있다.

### ① 충전 단계 (Filling Phase)

- 사출기의 스크류가 전진함에 따라 처음에는 금형의 캐비티 내로 수지가 유입된다. 이 단계를 충전 단계라고 하며 수지가 금형의 캐비티에 채워질 때까지 지속된다.

### ② 가압 단계 (Pressurisation Phase)

- 스크류가 더욱 전진하여 금형의 캐비티에 압력이 가해지는 단계를 가압 단계라고 한다. 캐비티가 충전되면 속도는 감소되며 계속 천천히 전진한다. 이것은 수지가 압축성의 재료이기 때문이며, 이로써 충전 단계에서 수지를 추가로 15%만큼 더 캐비티 내로 밀어 넣을 수 있다. 수지의 압축성은 노즐을 막은 후 스크류를 전진시키면 확인할 수 있다. 스크류는 압력이 가해질 때 앞으로 신속히 전진하지만, 압력을 제거하면 스프링 백(Spring Back) 현상으로 인해 뒤로 밀려난다.

### ③ 보압 단계 (Compensating Phase)

- 가압 단계 이후에도 스크류는 완전히 정지하지 않고 한동안 천천히 계속해서 전진 한다. 일반적으로 수지는 용융 상태에서 고체 상태로 될 때 약 25%의 체적 변화가 생긴다. 이와 같은 현상은 심한 요철이 생긴다. 이때 캐비티와 성형품의 체적 차이는 상기의 체적 변화 때문이다. 이 단계를 보압 단계라고 하며 상기와 같은 체적 차이를 보상하기 위한 것이다. 성형품의 체적 변화는 일반적으로 25%인데 비하여, 압축 단계에서는 15%만을 압축하므로 항상 보압 단계가 필요하다.

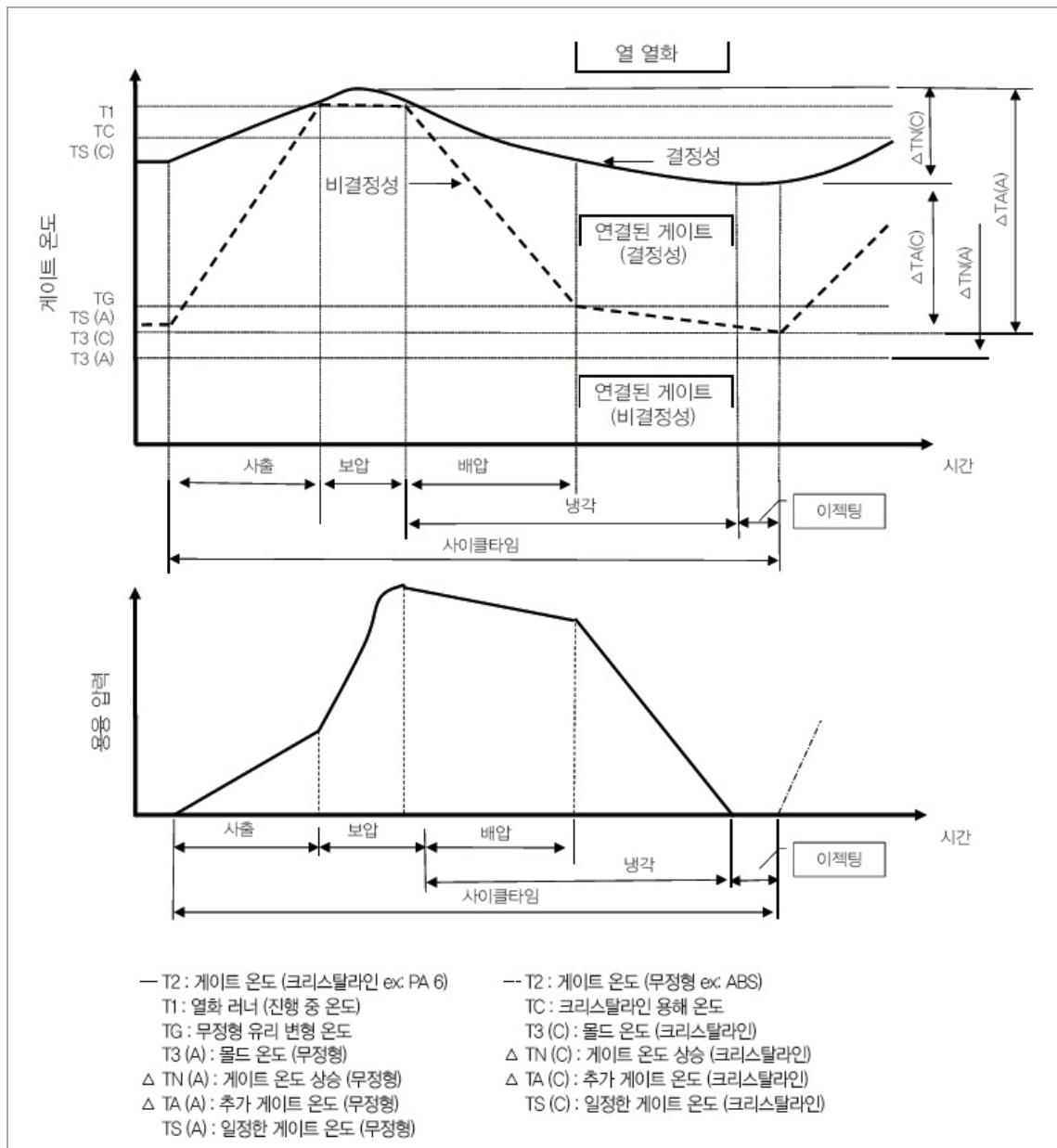
- 아래의 8단계를 사출성형 공정 1사이클이라고 하는데, [그림 2]는 1사이클 시간 동안 이루어지는 과정을 도식적으로 표현한 것이다. 열가소성 수지의 비결정성 수지와 결정성 수지를 게이트 온도와 사출압력으로 표시했다.

- ① 재료에 흡입된 수분을 제거하기 위해 재료를 1차 건조시킨 다음 사출성형기에 부착된 재료 받이실인 ‘호퍼’에 채워지고, 사출기의 실린더 내에 있는 사출스크루의 회전에 의해 원재료는 호퍼의 밑 부분에 있는 것부터 실린더 내로 유입되어 가소화된다. 이때 실린더는 외부가 히터로 감겨 있어 유입된 재료는 가열 용융되어 유동성을 갖는다.
- ② 사출성형기의 노즐부(고정측 체결부)에 금형의 상형이 체결되고 이젝터부(가동측 체결부)에 금형의 하형이 체결되어진 상태에서, 사출기의 가동측이 고정측 방향으로 이동함으로써 금형의 하형과 상형이 닫힌다. 이때 금형이 강한 사출압에 열리지 않도록 강한 조임력으로 금형이 닫혀져야 한다.
- ③ 금형의 상형 부위 유동 부분의 시작점인 스프루에 사출기의 노즐을 접촉시키고 난 다음, 유암실린더로 사출스크루를 전진시켜 가열실린더 안에 있는 수지를, 금형의 유동 부위를 거쳐 캐비티 내에 완전하게 골고루 충전되도록 높은 압력으로 사출한다. 이때 강한 사출압에 의해 금형이 열리지 않도록 강한 클램핑력이 금형에 작용해야 한다. 그렇지 않으면 금형

이 열려 제품의 열림면에 플래시가 발생한다.

- ④ 금형 안에 충전된 플라스틱은 제품을 고화시키기 위해 금형을 냉각시키는 공정에 의해 수축된다. 캐비티 안에서 플라스틱이 수축될 경우 제품의 품질(치수정밀도 및 외관)이 저하되므로 이것을 방지하기 위해 고화되지 않은 상태에서 사출압을 계속 유지해 준다. 이것을 보압이라고 한다. 이렇게 하면 재료가 수축되어 캐비티에 공간이 생긴 부위에 원재료를 더 보충할 수 있게 되어 품질 저하를 막을 수 있게 된다. 또한 강한 압력으로 사출을 하였기 때문에 강력한 압력으로 뒤에서 받쳐주지 않으면 원재료가 역류하여 밖으로 흘러나올 수 있다. 이 두 가지를 위해 필요한 공정이 보압 공정이며, 이 공정은 게이트가 굳어 원재료가 흘러나오지 않거나 보충할 수 없을 때까지 지속된다.
- ⑤ 금형을 냉각시켜 제품이 고화되는 동안 사출실린더 내의 사출스크루는 유압 모터에 의해 회전되면서 호퍼로부터 재료를 공급받아 스크루의 산 사이를 통해 노즐 쪽으로 보낸다. 이 때 재료는 실린더 벽면과의 사이에 압축되어 마찰되므로 열이 발생하고, 밴드히터에 의해 가열되므로 용융되어 앞부분인 노즐 쪽으로 이동된다. 노즐 부분인 앞부분에 원재료가 계속 채워지고 스크루는 수지 계량을 위해 설치한 리미트 스위치를 누를 때까지 후진한다.
- ⑥ 사출기의 가소화 작업이 끝난 후 사출기의 노즐이 뒤로 후퇴하고 금형이 열린다.
- ⑦ 금형이 열린 후 사출기의 가동축 부위에 설치되어 있는 이젝터 봉이 유압실린더의 작동에 의해 앞으로 전진하여 금형에 설치되어 있는 이젝터 봉을 앞으로 전진시켜 이젝터 판을 밀고 이젝터 핀이 제품을 밀어내어 취출시킨다.
- ⑧ 다시 금형이 닫힌다.

- 이러한 공정을 계속 반복하는 것이 성형 공정의 원리이며 ①~⑧까지 각 공정이 수행된 것을 1사이클이라고 한다.



[그림 2] 결정성 수지와 비결정 수지의 게이트 주변 온도 변화 성형 조건 도식화 (출처:박균명 박사 \_ 한국생산기술연구원 금형기술센터)

### • 성형조건의 설정

- 성형품의 품질저하에 영향을 주는 성형조건에 있어서는 수지온도, 사출속도와 압력, 균일한 냉각에 의해서 형성된다. 이러한 관계에 있어서 문제가 야기되고 있는 것이며 성형품의 형상, 크기 사용 재료에 따라 성형성은 더욱 복잡하게 되어 각 성형품별로 표준화를 작성하지 않으면 안된다. 일반적인 방법으로는

- 온도를 일정하게 하고 저압 (고압) 저속 (고속)
- 압력을 일정하게 하고 고온 (저온) 고속 (저속)
- 온도를 일정하게 하고 고온 (저온) 고압 (고압)

의 순으로 조합을 이루어 사출조건을 정한다.

## ① 실린더 온도

- 소재의 온도, 재료 grade, 금형구조, 사출성형기에 따라 차이가 있으며 제품표면에 광택을 요구하는 성형품의 성형에는 실린더 온도를 높게 설정하면 안되며, 실린더 온도를 높이면 충격강도가 높아지는 경우가 있으나 이러한 요인은 재료의 특성 별로 차이가 있다. 극단적으로 온도를 높게 설정하면 재료의 열분해, 플래시(flash)가 발생하기 쉽다. 제품 두께가 두꺼운 제품은 유동저항이 적기 때문에 실린더 온도는 재료의 유동성과 크게 영향을 미친다.

## ② 사출압력에 따른 사출속도

- 일반적으로 사출속도와 압력은 초기 주입(primary injection)시에 높게 설정하여 정한다. 두꺼운 제품이 성형에 있어서는 속도를 빠르게 하는 것이 중요하다. 캐비티 내의 수지온도를 균일하게 하여야 강도, 밀도를 균일하게 할 수 있다. 사출압력과 속도는 ‘cavity flow route’의 단면적에 관계가 있으며 제품 설계 시(금형설계) gate 위치에 유의를 해야 한다.

## ③ 금형온도

- 금형온도 설정도 일반적으로 높게 설정하며 이유는 유동저항을 적게 해주는데 있으며 또한 잔류응력을 적게 해주는데 있다. 금형온도를 높여주면 광택의 효과를 얻을 수 있으며 cavity core의 금형표면 온도를 일정하게 유지시켜 주는 것 이 이상적이다. 금형온도를 높여주면 성형수축이 크게 발생하는 경우가 있어 재료의 특성별로 조건설정이 중요하며 외관제품은 금형온도를 높게, 기능 및 치수가 중요한 제품은 금형온도를 낮게 해주어야 한다.

## ④ 사출시간

- 성형품의 형상, 크기에 따라 차이가 있으며 완전 충진과 외관강의 문제를 결정하는 요인이 된다. 두꺼운 제품에는 sink 또는 void의 발생을 방지 해야 되며 gate의 폭이 짧거나 작거나 하는 경우에 따라 사출압력, 재료의 온도와 상관관계가 있다. 사출시간이 짧으면 외관상으로 주름 무늬(wrinkle mark)가 발생하며 void발생으로 인한 충격강도 저하의 요인이 될 수 있다.

## ⑤ 냉각시간

- 실린더온도, 금형온도, 스프루, 런너 및 성형품의 두께에 따라 크게 변한다. 충분한 냉각시간으로서 성형품의 변형을 적게 해주면 성형 사이클이 길어지고 너무 짧

게 하면 취출 후 경시 변화에 의한 평면 밀도를 이루면서 왜곡(Distortion), 변형(Warpage), 또는 균열(Cracking)이 발생하게 된다.

## 2) 이슈사항(Pain point)

### • 공정의 문제 상황

- 사출성형에서는 각종 성형불량이 발생한다. 이를 불량현상을 해결하기 위해서는 우선 현상자체를 물리·화학적으로 잘 이해해 둘 필요가 있지만 개개의 경우에 대해서 신속히 그 원인을 규명해서 적절한 대책을 세우는 것은 쉬운 일이 아니다. 원인이 되는 요인의 몇 가지가 복잡하게 동시에 얹혀서 나오기도 하고, 한 가지 불량현상의 대책을 세우는 것에 대해서 다른 불량현상을 해결하는 경우도 있을 것이다. 불량을 발생시키는 중요한 인자에 대한 이해가 반드시 필요한 부분이라 하겠다. 하여 이하에 사출성형의 중요한 불량현상과 그 직접적인 원인에 대해서 설명해 그들 원인에 직결하는 요인을 대략적으로 정리하였으며 실제 공정에서의 이슈사항들을 같이 나열하였다.

### • 불량현상과 원인

#### a. 셔트숏

◦ 셔트숏(Shot Shot)은 금형 내에서의 원료 플라스틱의 충전량이 부족함으로 인해 발생하는 상태이다. 게이트로부터 떨어진 부분에서 발생하는 잔물결의 주름도 셔트숏 현상이다. 직접적인 원인은 원료 공급불량, 충전압력 부족, 금형 내 유동중의 원료고화, 금형 내의 공기 저항 등으로 분류할 수 있다. 이들 원인은 싱크마크, 플로우마크 등의 현상원인과 공통으로, 특히 싱크마크와 셔트숏은 현상적으로 구별하기 어려운 일도 있다.



[그림 3] 불량현상 – 셔트숏

### b. 플래시

- 플래시(flash)는 금형의 성형품 주입부 이외의 부분으로 용융 플라스틱이 유출되어 고화하는 현상이다. 금형의 접합부에서 발생한다.



[그림 4] 불량현상 - 플래시

### c. 싱크마크

- 싱크마크(Sink Mark)는 성형품 표면에 발생하는 함몰현상이다. 전형적인 싱크마크는 부분적으로 두꺼운 부분에 생기는 것으로 금형내에서 압축되어진 용융 플라스틱의 온도 저하에 의한 체적의 감소와 압력 저하에 동반하는 체적의 증가 균형이 깨지는 것이 직접원인이다. 원리적으로는 성형수축성의 문제와 같다. 부분적으로 큰 성형수축이 일어나서 그것이 성형품 표면에 존재한다면 싱크마크가 되고 성형품 내부가 되면 공동이 된다. 더욱 싱크마크는 전형적인 함몰외관이 다른 선상에로 들어가 얇은 요철이 다수 생기는 등 여러 가지 모습으로 나타나는 경우도 있다.



[그림 5] 불량현상 - 싱크마크

#### d. 플로우마크

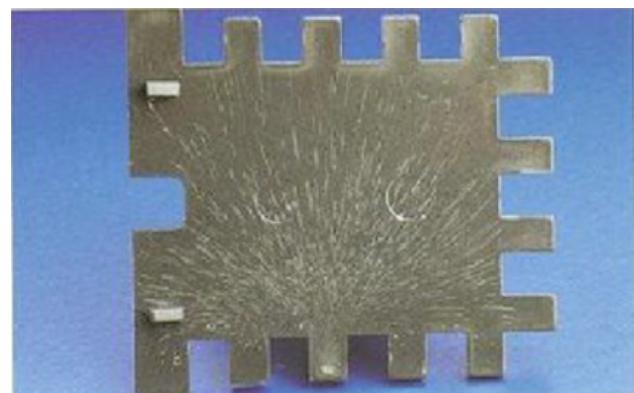
- 플로우마크(Flow Mark)는 성형품 표면에 게이트를 중심으로 동심원상의 광택 또는 표면이 거칠어짐을 발생시키는 현상이다. 이것은 용융원료가 금형 내를 주름상으로 파동하면서 흘러서 금형면과 보다 많이 접촉한 부분이 앞서 고화되기 때문에 일어나는 것이라고 생각할 수 있는데 아직 충분히 해명되고 있지 않은 점도 있다. 금형유동시 수지의 점도가 크면 발생하기 쉽다.



[그림 6] 불량현상 - 플로우마크

#### e. 은조흔

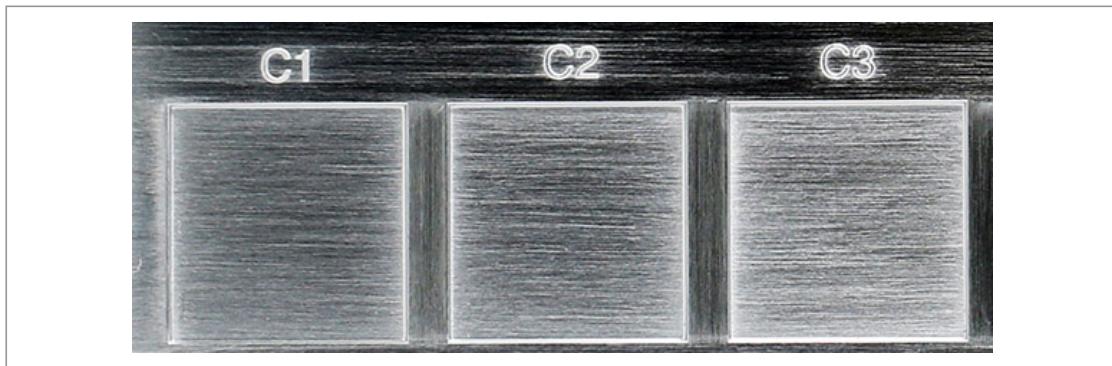
- 은조흔(Silver Streak)은 성형품의 표면에 원료가 흐르는 방향을 따라서 은백의 조흔(條痕)이 생기는 현상이다. 이것은 성형품이 고화되기 전에 그 표면을 가스가 지나가 흔적이다.



[그림 7] 불량현상 - 은조흔

#### f. 광택불량

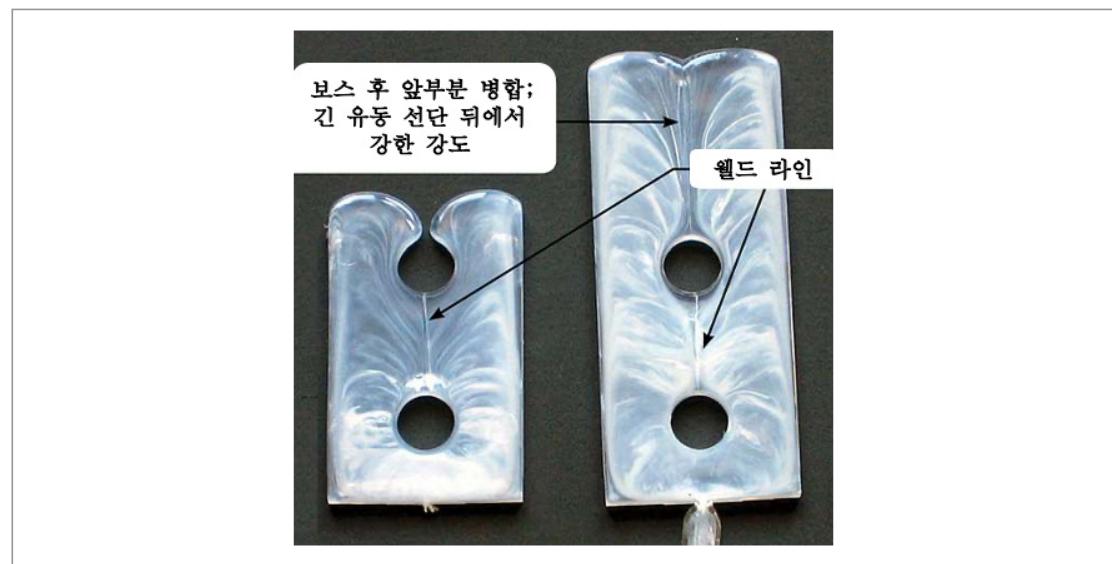
- 광택불량과 흐름(Dull Surface)은 성형품의 표면이 일면에 광택이 나쁘게 되기도 하고 무색 투명품에서는 유백의 흐림을 일으키는 현상이다. 이들은 금형면에 플라스틱이 충분히 압착되지 못하기도 하고 노즐로부터 배출된 가스(수증기, 휘발분 등)가 금형표면에 응축해서 플라스틱 층과 금형면의 직접 접착을 방해하는 것이 원인이다. 금형에 이형제를 너무 도포해도 마찬가지이다.



[그림 8] 불량현상 – 광택불량

#### g. 웨드라인

- 웨드라인(Weld Line)은 금형 내에서 용융 플라스틱의 두 개 유동선단이 합류하여 용착한 부분에 발생하는 가는 선이다. 성형품에 빈틈부가 있는 경우 두께의 변화가 있어 두꺼운 부분으로부터 얇은 부분으로 일방적으로 원료를 흐르게 하지 않는 경우 게이트가 2개 이상있는 경우에는 웨드라인의 발생은 피할 수 없다. 불량현상으로써 웨드라인은 이들 유동선단이 용착불량에 의해서 외관적으로 눈에 띄고 강도도 현저하게 저하된다.



[그림 9] 불량현상 – 웨드라인

## h. 기포

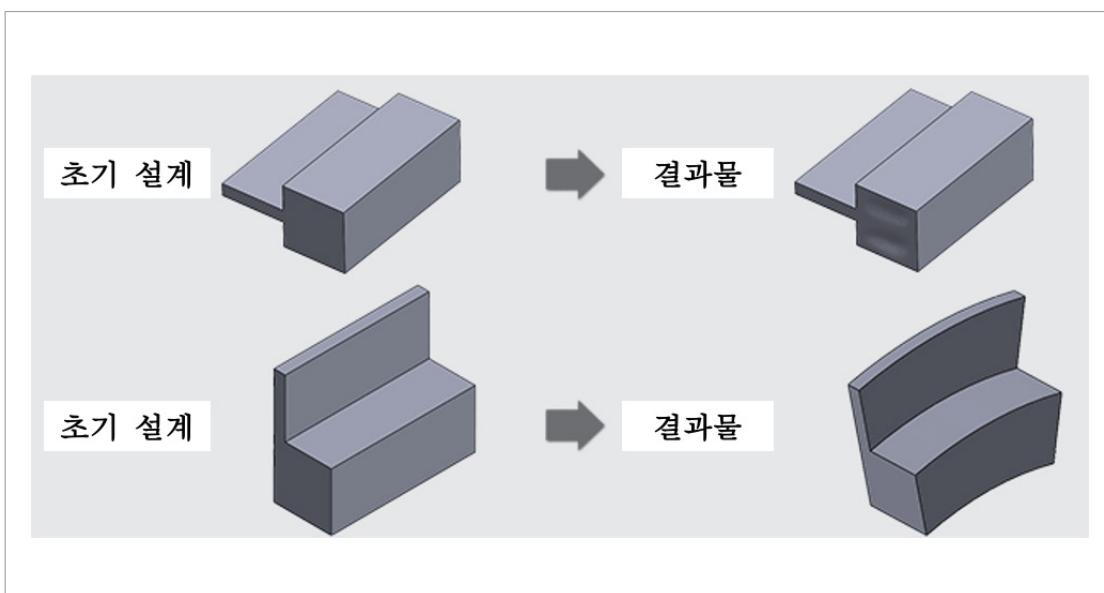
- 기포(Bubble)는 성형품 내부에 공洞을 일으키는 현상이다. 직접적인 원인은 은조 흔의 경우와 마찬가지로 가스이다. 다량의 가스가 노즐로부터 방출되어지기도 하고 금형 내에서 용융 플라스틱에 충분한 압력이 작용하지 않는 경우에 발생한다.



[그림 10] 불량현상 - 기포

## i. 변형 또는 흡

- 변형 또는 흡(Warpage)의 직접원인은 성형품의 잔류응력이다. 따라서 크레이징의 경우와 완전히 같은 원인에 의해서 그 영향이 치수 변화와 함께 나타난 것이다. 일반적으로 성형수축률이 크고, 그 수치가 불균일한 두께에 따른 냉각속도의 차이 등으로 인해 부분적으로 상당히 변화하는 폴리에틸렌, 폴리프로필렌 등에서는 특히 문제가 되는 일이 많다. 성형품의 설계면에서는 배려가 중요하다.



[그림 11] 불량현상 - 변형 또는 흡

- 사출성형 현장의 직접적인 이슈사항
  - a. 미성형, 가스자국 등 불량 발생 과다
  - b. 불량 발생 원인 분석 어려움, 불량 발생시 조치 시간 과다 발생
  - c. 근본 원인 제거 및 문제 해결 지식의 자산화 부족
  - d. 사출 Recipe 조건인 소재량, 온도, 압력, 속도 등의 설정을 경험에 의존

- 문제해결 장애요인

- 성형된 제품이 냉각되어 수축된 후 불량이 발생하고 이를 확인할 수 있기 때문에 불량 발생 시점에서 공정을 멈추거나 제어하는 것이 어렵다.

- 극복방안

- 아래에 기술하는 사출기의 주요 공정변수를 실시간으로 수집하고 불량원인분석을 통해 불량예측을 할 수 있다. 불량예측이 발생되면 불량원인인자를 표출하여 작업자가 빠른 조치를 취할 수 있게 되어 이를 공정 제어에 이용할 수 있다.

## 1.2 분석 목표

### 1) 분석목표

- 사출 공정에서 발생하는 품질문제와 생산조건 데이터들의 분석을 통하여 데이터 간의 상관관계를 찾고 주요 문제별 원인 인자를 분석하여 다양한 기계 학습 알고리즘을 활용, 사출 공정 불량예측 모델과 생산조건 최적화 모델을 만들고자 한다.

### 2) 제조 데이터 정의 및 소개

공정 변수 조건		내용
독립변수	온도 관련	스크류/실린더, 수지, 금형, 건조, 유압, 주변 환경
	압력 관련	충진 압력, 보압, 배압(계량시 발생하는 압력), 이형 압력, 형개 압력, 형체 압력
	시간 관련	충진 시간, 보압 시간, 냉각 시간, 건조 시간
	속도 관련	사출 속도, 스크류 회전 속도, 형개 속도, 이형(이젝팅) 속도
	양 관련	계량, 이형량, 쿠션량
종속변수	불량 여부	Y : 양품, N : 불량품 (Labeled), 표시 없음 (Unlabeled)

### 3) 제조 데이터 분석 기대효과

- 사출 공정에서 수집되는 다양한 변수들에 따라 양품과 불량품을 예측을 할 수 있는 모델을 제공한다.

#### 4) 시사점(implication) 요약기술

- 현재 자동차 부품 생성하는 사출 공정에서는, 육안으로 제품의 양품선별을 진행하지만, 사람의 의한 판단으로 품질의 균일성을 확보하기 힘들었다. 하여, 양품 선별 레이블의 유무에 따른 데이터셋 특성에 따라 적용할 수 있는 AI 모델을 구축하고, 이러한 모델을 통한 양품 선별을 할 수 있는 분석을 수행한다.
- 본 분석은, 사출 공정의 양품 선별값 및 생산 조건 데이터를 수집 후, 가공/전처리, AI 모델 개발과 제조공정의 적용 및 검증을 통해 중소기업에 빅데이터 및 AI를 적용하여 실질적인 품질 및 비용 절감 개선에 기여할 수 있다는 점에서 시사하는 바가 크다고 판단된다.

## 2. 분석실습

### 2.1 제조데이터 소개

#### 1) 데이터 수집 방법

- 제조 분야 : 자동차 앞유리 사이드 몰딩
- 제조 공정명 : 자동차 앞유리 사이드 몰딩사출 공정
- 수집장비 : 공장내 사출성형기 Data, 품질 Data
- 수집기간 : 2020년 10월 16일 ~ 2020년 11월 19일 (1개월)
- 수집주기 : 0.2 sec

#### 2) 데이터 유형/구조

##### • 비식별화 원본 데이터 :

- 공정 과정에서 수집된 센서 데이터에서 비식별화 및 품질 전처리를 수행한 데이터셋 이다

##### • 학습 통합 데이터 (label 데이터) :

- ‘2.3.1 지도 학습 활용 알고리즘 (이상 탐지)’의 ‘[단계 ③]데이터 정제 (전처리)’에서 확인할 수 있듯, 비식별화 원본 데이터에서 데이터 분석을 하고자 하는 제품(예 \_ “650톤-우진2호기”의 제품 “CN7 W/S SIDE MLD’G LH”, “CN7 W/S SIDE MLD’G RH”)만 추출한 뒤 두 데이터를 통합한 후, 가공 일자, 제품의 시리얼 넘버, 모든 값들이 0으로 채워져 있는 등의 불필요한 열(column)들을 제거하여 26개의 열만 남겨둔다.

- AutoEncoder 모델에 입력되는 데이터는 위에서 기술한 ‘학습 통합 레이블 데이터 (cn7)’에서 추가적으로 양품과 불량 데이터를 분리하고 종속 변수(“PassOrFail” 열)와 독립 변수(“PassOrFail” 열을 제외한 모든 열)를 구분하였으며, 해당 과정은 ‘[단계 ⑤] 학습 및 평가 데이터 분리’부터 확인할 수 있다.

## • 1차 가공 데이터

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
_id	TimeStamp	PART_FACT_PART_NAME	PART_NAME	EQUIP_CD	EQUIP_NAME	PassOrFail	Reason		Injection_T	Filling_Tim	Plasticizing	Cycle_Time	Clamp_Clos	Cushion_Pc
2	5f8928bb5	2020-10-16 4:57	2020-10-16	24 CN7 W/S SIDE MLD'G RH	S14	650우진-2호기	Y	None	9.59	4.47	16.92	59.52	7.13	653.41
3	5f8928de5	2020-10-16 4:58	2020-10-16	24 CN7 W/S SIDE MLD'G RH	S14	650우진-2호기	Y	None	9.6	4.48	16.91	59.58	7.13	653.41
4	5f8928df9	2020-10-16 4:58	2020-10-16	23 CN7 W/S SIDE MLD'G LH	S14	650우진-2호기	Y	None	9.6	4.48	16.91	59.58	7.13	653.41
5	5f8928f39	2020-10-16 4:59	2020-10-16	23 CN7 W/S SIDE MLD'G LH	S14	650우진-2호기	Y	None	9.59	4.48	16.91	59.56	7.13	653.42
6	5f8928f59	2020-10-16 4:59	2020-10-16	24 CN7 W/S SIDE MLD'G RH	S14	650우진-2호기	Y	None	9.59	4.48	16.91	59.56	7.13	653.42
7	5f89294b5	2020-10-16 5:00	2020-10-16	24 CN7 W/S SIDE MLD'G RH	S14	650우진-2호기	Y	None	9.58	4.46	16.9	59.58	7.13	653.41
8	5f89294c9	2020-10-16 5:00	2020-10-16	23 CN7 W/S SIDE MLD'G LH	S14	650우진-2호기	Y	None	9.58	4.46	16.9	59.58	7.13	653.41
9	5f8929985	2020-10-16 5:01	2020-10-16	24 CN7 W/S SIDE MLD'G RH	S14	650우진-2호기	Y	None	9.58	4.46	16.92	59.56	7.13	653.41
10	5f8929995	2020-10-16 5:01	2020-10-16	23 CN7 W/S SIDE MLD'G LH	S14	650우진-2호기	Y	None	9.58	4.46	16.92	59.56	7.13	653.41
11	5f8929a85	2020-10-16 5:02	2020-10-16	24 CN7 W/S SIDE MLD'G RH	S14	650우진-2호기	Y	None	9.57	4.45	16.91	59.52	7.14	653.41
12	5f8929a95	2020-10-16 5:02	2020-10-16	23 CN7 W/S SIDE MLD'G LH	S14	650우진-2호기	Y	None	9.57	4.45	16.91	59.52	7.14	653.41
13	5f8929eb5	2020-10-16 5:03	2020-10-16	24 CN7 W/S SIDE MLD'G RH	S14	650우진-2호기	Y	None	9.56	4.45	16.9	59.52	7.14	653.41
14	5f8929ec9	2020-10-16 5:03	2020-10-16	23 CN7 W/S SIDE MLD'G LH	S14	650우진-2호기	Y	None	9.56	4.45	16.9	59.52	7.14	653.41
15	5f892a235	2020-10-16 5:04	2020-10-16	23 CN7 W/S SIDE MLD'G LH	S14	650우진-2호기	Y	None	9.57	4.45	16.9	59.52	7.13	653.41

[그림 12] labeled\_data.csv 예시

- 1차 가공 데이터는 2가지가 제공이 된다. 물품 각각에 대한 양품 선별 여부가 있는 데이터 (labeled\_data), 물품 각각에 대한 양품 선별 여부는 없이, 일별 불량품 개수만 추계된 데이터 (unlabeled\_data), 총 2가지 데이터가 csv(column separated value)의 형태로 추출된다.
- 두 데이터 셋 모두 같은 데이터를 수집하고, 단 'labeled\_data'의 경우에만 "PassorFail"이라는 데이터 열이 마지막 열(제일 오른쪽)에 추가되어, 각 물품에 대한 개별 불량 여부가 같이 제공된다.

### 〈지도 학습용〉

- 불량 선별 여부 포함 데이터 (labeled): labeled\_data.csv

### 〈준지도 학습용〉

- 불량 선별 여부 포함 데이터 (labeled): moldset\_labeled.csv

불량 선별 여부 불포함 데이터 (unlabeled): unlabeled\_data.csv

## • 데이터 속성정의 표

Attributes name	Description	Unit
<b>id</b>	제조 공정 id	-
<b>TimeStamp</b>	시간( YY:HH:MM:SS)	-
<b>PART_FACT_PLAN_DATE</b>	생산을 지시한 날짜	-
<b>PART_FACT_SERIAL</b>	생산을 지시한 품목에 대한 코드부여	-
<b>PART_NO*</b>	제품의 모델코드	-
<b>PART_NAME</b>	제품의 이름	-
<b>EQUIP_CD</b>	생산한 사출기 호기	-
<b>EQUIP_NAME</b>	생산한 사출기 모델명	-
<b>PassOrFail**</b>	사출되는 각 물품마다 검수자가 양품 선별을 하여 붙이는 레이블 값	0 : 불 1 : 합

Attributes name	Description	Unit
Reason**	검수자가 양품선별 시, 불량으로 확인된 제품에 대해 불량 유형을 나타냄	Unlabeled 에서는 확인불가
ERR_FACT_QTY	생산시 발생한 불량의 개수	수량 (개)
Injection_Time	고압+사출시간(고압(사출압) : 재료를 금형에 유입시킬때의 압력), 사출 시간 : 재료를 금형에 유입시키는데 소요되는 시간)	Sec (초)
Filling_Time	충진시간으로 사출기에서 금형으로 내용물이 주입되는 시간	Sec (초)
Plasticizing_Time	계량시간으로 재료를 스크류에 1번 생산할 만큼 용융되어 저장되는 시간	Sec (초)
Cycle_Time	1번의 제품생산에 소요되는 생산시간	Sec (초)
Clamp_Close_Time	제품이 생산되고 난후 열려있는 금형을 사출기가 닫아주고 빙틈이 없이 고정축과 이동축을 꽉 잡아주는데 걸리는 시간	Sec (초)
Cushion_Position	보압(사출압의 다음으로 가해지는 압력(금형내부압력을 조절하여 과충전을 방지))을 하기위한 스크류의 위치	mm
Switch_Over_Position	고압,보압절환위치(고압(사출압)에서 보압으로 진행될때의 위치))	mm
Plasticizing_Position	계량완료위치(계량을 마친 스크류의 위치)	mm
Clamp_Open_Position	제품이 생산되어 추출하기위해 금형이 열리고 난 위치	mm
Max_Injection_Speed	배럴에 계량되어 있는 용융수지가 금형으로 흘러들어가는데 측정되는 최대속도	mm/s
Max_Screw_RPM	사출을 위한 스크류의 최대속도	mm/s
Average_Screw_RPM	사출을 위한 스크류의 평균속도	mm/s
Max_Injection_Pressure	배럴에 계량되어 있는 용융수지가 금형으로 흘러들어가는데 가해지는 최대압력	MPa
Max_Switch_Over_Pressure	사출에서 보압(총진된 수지가 밀리지않게 압력을준다)으로 변환되는 압력	MPa
Max_Back_Pressure	수지가 계량이 되는중에 스크류가 밀려나는 현상을 저지하기위한 최대압력	MPa
Average_Back_Pressure	수지가 계량이 되는중에 스크류가 밀려나는 현상을 저지하기위한 평균압력	MPa
Barrel_Temperature_1	계량 및 사출시 수지가 일정하게 용융(녹임)을 유지하기위해 온도가 일정해야한다	섭씨 (°C)
Barrel_Temperature_2		섭씨 (°C)
Barrel_Temperature_3		섭씨 (°C)
Barrel_Temperature_4		섭씨 (°C)
Barrel_Temperature_5		섭씨 (°C)
Barrel_Temperature_6		섭씨 (°C)
Barrel_Temperature_7		섭씨 (°C)
Hopper_Temperature	재료주입구의 온도(충분히 건조시켜주며 재료가 용융되는시간을 절약시켜주기위해 온도가 높아야한다)	섭씨 (°C)

Attributes name	Description	Unit
Mold_Temperature_1		섭씨 (°C)
Mold_Temperature_2		섭씨 (°C)
Mold_Temperature_3		섭씨 (°C)
Mold_Temperature_4		섭씨 (°C)
Mold_Temperature_5		섭씨 (°C)
Mold_Temperature_6	재료주입구의 온도(충분히 건조시켜주며 재료가 용융되는시간을 절약시켜주기위해 온도가 높아야한다)	섭씨 (°C)
Mold_Temperature_7		섭씨 (°C)
Mold_Temperature_8		섭씨 (°C)
Mold_Temperature_9		섭씨 (°C)
Mold_Temperature_10		섭씨 (°C)
Mold_Temperature_11		섭씨 (°C)
Mold_Temperature_12		섭씨 (°C)

\* Part\_No는 제품의 모델번호(고유번호 아님)이고, 해당 데이터들은 데이터베이스에서 분석자가 어떤 속성을 불러오느냐에 따라 다른데, 'labeled\_data.csv'에서는 해당 속성을 불러오지 않았으나, 'moldset\_labeled.csv' 와 'unlabeled\_data.csv'에서는 확인할 수 있다.

\*\* PassOrFail, Reason은 클래스 여부를 확인할 수 있는 labeled에서만 확인할 수 있다.

(파일이름: labeled data.csv[지도학습], moldset\_labeled.csv[준지도학습])

## • 2차 가공 데이터

A	B	C	D	E	F	G	H	I	J	K	L	M
1	PassOrFail	Injection_T	Filling_Tim	Plasticizing	Cycle_Time	Clamp_Clo	Cushion_Pc	Plasticizing	Clamp_Ope	Max_Inject	Max_Screw	Average_Sc
2	0	1.679123	1.491397	-0.47802	0.361047	0.97858	-0.71116	1.532018	0	-1.1524	-0.79573	-0.13202
3	1	0.2080358	1.894372	-0.5271	1.886261	0.97858	-0.71116	1.471608	0	-1.3471	0.049553	-0.13202
4	2	0.2080358	1.894372	-0.5271	1.886261	0.97858	-0.71116	1.471608	0	-1.3471	0.049553	-0.13202
5	3	0.1679123	1.894372	-0.5271	1.377857	0.97858	1.202502	1.471608	0	-1.3471	1.740153	-0.13202
6	4	0.1679123	1.894372	-0.5271	1.377857	0.97858	1.202502	1.471608	0	-1.3471	1.740153	-0.13202
7	5	0.1277888	1.088442	-0.57618	1.886261	0.97858	-0.71116	1.471608	0	-0.76302	0.894853	-0.13202
8	6	0.1277888	1.088442	-0.57618	1.886261	0.97858	-0.71116	1.471608	0	-0.76302	0.894853	-0.13202
9	7	0.1277888	1.088442	-0.47802	1.377857	0.97858	-0.71116	1.532018	0	-0.95771	-1.64103	-1.29059
10	8	0.1277888	1.088442	-0.47802	1.377857	0.97858	-0.71116	1.532018	0	-0.95771	-1.64103	-1.29059

[그림 13] moldset\_labeled\_cn7.csv 예시

A	B	C	D	E	F	G	H	I	J	K	L	M
1	PassOrFail	Injection_T	Filling_Tim	Plasticizing	Cycle_Time	Clamp_Clo	Cushion_Pc	Plasticizing	Clamp_Ope	Max_Inject	Max_Screw	Average_Sc
2	114610	1.346829	-2.76886	0.961909	1.843294	1.029767	1.031111	0.849819	0.861854	2.489188	1.025136	1.031548
3	114688	1.360046	-2.72031	0.96328	1.847956	1.029767	1.031048	0.863559	0.861854	2.446877	1.025136	1.031548
4	114919	1.355641	-2.72031	0.96602	1.843294	1.029767	1.03108	0.864246	0.861854	2.446877	1.025136	1.031548
5	115304	1.360046	-2.72031	0.960539	1.843294	1.029767	1.031143	0.864246	0.861854	2.455339	1.025136	1.031548
6	115582	1.360046	-2.72031	0.967391	1.847956	1.029767	1.031048	0.864933	0.861854	2.446877	1.018644	1.031548
7	115743	1.355641	-2.73649	0.961909	1.843294	1.029767	1.031143	0.864246	0.861854	2.463802	1.018644	1.031548
8	115784	1.355641	-2.72031	0.967391	1.843294	1.029767	1.031143	0.864246	0.861854	2.455339	1.012153	1.031548
9	116725	1.421726	-2.49371	0.998909	1.855726	1.034739	1.031017	0.933634	0.861854	2.303019	1.018644	1.032232
10	143072	-0.49914	1.973436	-0.93193	-0.70829	-0.76023	-0.96633	-0.86632	-0.22983	-0.65031	-0.96769	-0.96977

[그림 14] moldset\_unlabeled\_cn7.csv 예시

- 2차 가공 데이터는 본 데이터셋에서는 4가지가 제공이 된다. 1차 가공데이터 중, 전처리를 통하여, 실제 분석을 진행하는 제품 2개(자동차 사이드 몰딩, 2가지 제품 : cn7, rg3)에 대하여 각각 labeled\_data 및 unlabeled\_data를 csv 파일 형태로 가공한 데이터이다.

#### 〈 지도 학습용 〉

- 불량 선별 여부 포함 데이터 (labeled): supervised\_label\_cn7.csv

#### 〈 준지도 학습용 〉

- 불량 선별 여부 포함 데이터 (labeled): moldset\_labeled\_cn7.csv
- 불량 선별 여부 포함 데이터 (labeled): moldset\_labeled\_rg3.csv
- 불량 선별 여부 불포함 데이터 (unlabeled): moldset\_unlabeled\_cn7.csv
- 불량 선별 여부 불포함 데이터 (unlabeled): moldset\_unlabeled\_rg3.csv

### • 데이터 속성정의 표

Attributes name	Description	Unit
PassOrFail*	사출되는 각 물품마다 검수자가 양품 선별을 하여 붙이는 레이블 값	-
Injection_Time	고압+사출시간(고압(사출압) : 재료를 금형에 유입시킬때의 압력), 사출시간 : 재료를 금형에 유입시키는데 소요되는 시간)	Sec (초)
Filling_Time	충진시간으로 사출기에서 금형으로 내용물이 주입되는 시간	Sec (초)
Plasticizing_Time	계량시간으로 재료를 스크류에 1번 생산할 만큼 용융되어 저장되는 시간	Sec (초)
Cycle_Time	1번의 제품생산에 소요되는 생산시간	Sec (초)
Clamp_Close_Time	제품이 생산되고 난후 열려있는 금형을 사출기가 닫아주고 빈틈이 없이 고정축과 이동축을 꽉 잡아주는데 걸리는 시간	Sec (초)
Cushion_Position	보압(사출압의 다음으로 가해지는 압력(금형내부압력을 조절하여 과충전을 방지))을 하기위한 스크류의 위치	mm
Plasticizing_Position	계량완료위치(계량을 마친 스크류의 위치)	mm
Clamp_Open_Position	제품이 생산되어 추출하기위해 금형이 열리고 난 위치	mm
Max_Injection_Speed	배럴에 계량되어 있는 용융수지가 금형으로 흘러들어가는데 측정되는 최대 속도	mm/s
Max_Screw_RPM	사출을 위한 스크류의 최대속도	mm/s
Average_Screw_RPM	사출을 위한 스크류의 평균속도	mm/s
Max_Injection_Pressure	배럴에 계량되어 있는 용융수지가 금형으로 흘러들어가는데 가해지는 최대 압력	MPa
Max_Switch_Over_Pressure	사출에서 보압(충진된 수지가 밀리지않게 압력을준다)으로 변환되는 압력	MPa
Max_Back_Pressure	수지가 계량이 되는중에 스크류가 밀려나는 현상을 저지하기위한 최대압력	MPa
Average_Back_Pressure	수지가 계량이 되는중에 스크류가 밀려나는 현상을 저지하기위한 평균압력	MPa

Attributes name	Description	Unit
Barrel_Temperature_1		섭씨 (°C)
Barrel_Temperature_2		섭씨 (°C)
Barrel_Temperature_3		섭씨 (°C)
Barrel_Temperature_4	계량 및 사출시 수지가 일정하게 용융(녹임)을 유지하기위해 온도가 일정해야한다	섭씨 (°C)
Barrel_Temperature_5		섭씨 (°C)
Barrel_Temperature_6		섭씨 (°C)
Hopper_Temperature	재료주입구의 온도(충분히 건조시켜주며 재료가 용융되는시간을 절약시켜 주기위해 온도가 높아야한다)	섭씨 (°C)
Mold_Temperature_3	재료주입구의 온도(충분히 건조시켜주며 재료가 용융되는시간을 절약시켜 주기위해 온도가 높아야한다)	섭씨 (°C)
Mold_Temperature_4		섭씨 (°C)

\* PassorFail은 불량여부 선별이된 데이터에서만 확인할 수 있다.

파일 이름: 'supervised\_label\_cn7.csv', 'moldset\_labeled.cn7.csv', 'moldset\_labeled\_rg3.csv'.

### • 기술 통계 (준지도학습의 1차가공 데이터 활용)

#### 1) moldset\_labeled.csv (불량여부 클래스가 있는 데이터)

구분	개수	평균	표준편차	최소값	중간값	최대값	최빈값
Injection_Time	2607	5.71	4.24	1.05	9.52	13.39	1.06
Filling_Time	2607	2.86	1.76	0.93	4.41	8.27	0.94
Plasticizing_Time	2607	15.14	2.03	12.80	16.78	21.10	12.90
Cycle_Time	2607	60.54	1.13	58.96	59.58	62.36	61.78
Clamp_Close_Time	2607	6.98	0.16	6.79	7.12	7.18	7.12
Cushion_Position	2607	653.80	0.42	653.39	653.45	654.29	653.41
Switch_Over_Position	2607	1.01	25.65	0.00	0.00	655.31	0.00
Plasticizing_Position	2607	61.79	7.48	53.55	68.48	68.86	53.59
Clamp_Open_Position	2607	356.29	320.34	4.63	647.99	647.99	647.99
Max_Injection_Speed	2607	88.64	36.07	38.50	56.30	128.50	128.30
Max_Screw_RPM	2607	30.80	0.12	30.40	30.80	31.20	30.80
Average_Screw_RPM	2607	291.60	0.98	290.40	292.40	293.90	292.50
Max_Injection_Pressure	2607	142.18	0.48	140.70	142.20	147.40	141.80
Max_Switch_Over_Pressure	2607	127.33	10.10	109.70	136.00	146.70	136.40
Max_Back_Pressure	2607	46.18	9.04	36.30	59.80	71.90	38.40
Average_Back_Pressure	2607	60.13	1.10	57.70	276.80	87.10	59.60
Barrel_Temperature_1	2607	280.60	4.90	274.80	275.70	287.10	276.00
Barrel_Temperature_2	2607	279.78	4.96	274.20	275.50	286.50	275.30
Barrel_Temperature_3	2607	279.53	4.99	274.10	271.50	285.80	275.00
Barrel_Temperature_4	2607	272.66	2.58	268.80	255.40	276.80	269.60
Barrel_Temperature_5	2607	259.54	5.00	253.50	230.30	266.40	255.00
Barrel_Temperature_6	2607	232.25	2.49	229.30	0.00	235.50	230.10

구분	개수	평균	표준편차	최소값	중간값	최대값	최빈값
Barrel_Temperature_7	2607	0.00	0.00	0.00	66.60	0.00	0.00
Hopper_Temperature	2607	66.72	1.59	62.60	0.00	70.60	66.50
Mold_Temperature_1	2607	0.00	0.00	0.00	0.00	0.00	0.00
Mold_Temperature_2	2607	0.00	0.00	0.00	0.00	0.00	0.00
Mold_Temperature_3	2607	22.81	1.27	19.60	23.80	25.30	25.00
Mold_Temperature_4	2607	24.18	1.69	21.00	0.00	27.80	23.80
Mold_Temperature_5	2607	0.00	0.00	0.00	0.00	0.00	0.00
Mold_Temperature_6	2607	0.00	0.00	0.00	0.00	0.00	0.00
Mold_Temperature_7	2607	0.00	0.00	0.00	0.00	0.00	0.00
Mold_Temperature_8	2607	0.00	0.00	0.00	0.00	0.00	0.00
Mold_Temperature_9	2607	0.00	0.00	0.00	0.00	0.00	0.00
Mold_Temperature_10	2607	0.00	0.00	0.00	0.00	0.00	0.00
Mold_Temperature_11	2607	0.00	0.00	0.00	0.00	0.00	0.00
Mold_Temperature_12	2607	0.00	0.00	0.00	0.00	0.00	0.00

## 2) unlabeled\_data.csv (불량여부 클래스가 있는 데이터)

구분	개수	평균	표준편차	최소값	중간값	최대값	최빈값
Injection_Time	795315	9.45	5.57	0.13	9.52	29.10	5.00
Filling_Time	795315	3.90	1.65	0.00	3.35	29.10	5.00
Plasticizing_Time	795315	11.23	7.62	0.20	10.95	468.20	2.80
Cycle_Time	795315	58.03	16.92	0.00	64.42	648.39	35.60
Clamp_Close_Time	795315	5.58	1.91	2.11	6.08	30.10	3.40
Cushion_Position	795315	215.04	296.93	0.00	18.20	655.33	17.30
Switch_Over_Position	795315	10.88	19.05	0.00	10.00	655.34	0.00
Plasticizing_Position	795315	63.56	40.96	31.27	59.90	392.90	59.90
Clamp_Open_Position	795315	561.10	369.23	0.00	522.50	1901.00	647.99
Max_Injection_Speed	795315	47.58	27.67	0.00	38.80	128.80	59.90
Max_Screw_RPM	795315	22.22	18.47	0.00	27.00	106.30	0.00
Average_Screw_RPM	795315	104.96	132.14	0.00	25.50	484.20	0.00
Max_Injection_Pressure	795315	118.59	34.73	0.00	131.70	199.90	80.60
Max_Switch_Over_Pressure	795315	101.22	31.70	0.10	102.30	6553.10	69.90
Max_Back_Pressure	795315	15.50	20.12	0.00	5.20	121.50	5.30
Average_Back_Pressure	795315	27.84	28.32	0.00	22.10	165.20	0.00
Barrel_Temperature_1	795315	88.72	124.63	0.00	0.00	990.20	0.00
Barrel_Temperature_2	795315	262.88	13.27	189.40	264.80	340.30	264.80
Barrel_Temperature_3	795315	266.19	11.71	189.00	269.90	290.30	269.90
Barrel_Temperature_4	795315	264.33	11.42	179.40	269.80	290.10	270.00
Barrel_Temperature_5	795315	257.66	12.90	179.60	259.90	290.20	254.90
Barrel_Temperature_6	795315	241.97	15.35	167.80	239.90	276.80	239.90
Barrel_Temperature_7	795315	22.46	15.80	0.00	30.20	50.50	0.00

구분	개수	평균	표준편차	최소값	중간값	최대값	최빈값
Hopper_Temperature	795315	38.44	27.36	0.00	56.70	75.10	0.00
Mold_Temperature_1	795315	0.00	0.00	0.00	0.00	0.00	0.00
Mold_Temperature_2	795315	0.00	0.00	0.00	0.00	0.00	0.00
Mold_Temperature_3	795315	7.67	11.48	0.00	0.00	39.20	0.00
Mold_Temperature_4	795315	8.94	13.61	0.00	0.00	51.10	0.00
Mold_Temperature_5	795315	0.00	0.00	0.00	0.00	0.00	0.00
Mold_Temperature_6	795315	0.00	0.00	0.00	0.00	0.00	0.00
Mold_Temperature_7	795315	0.00	0.00	0.00	0.00	0.00	0.00
Mold_Temperature_8	795315	0.00	0.00	0.00	0.00	0.00	0.00
Mold_Temperature_9	795315	0.00	0.00	0.00	0.00	0.00	0.00
Mold_Temperature_10	795315	0.00	0.00	0.00	0.00	0.00	0.00
Mold_Temperature_11	795315	0.00	0.00	0.00	0.00	0.00	0.00
Mold_Temperature_12	795315	0.00	0.00	0.00	0.00	0.00	0.00

#### • 독립변수/ 종속변수 정의

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

구분	구분	비고
독립변수	Injection_Time	
	Filling_Time	
	Plasticizing_Time	
	Cycle_Time	
	Clamp_Close_Time	
	Cushion_Position	
	Switch_Over_Position	
	Plasticizing_Position	
	Clamp_Open_Position	
	Max_Injection_Speed	
	Max_Screw_RPM	
	Average_Screw_RPM	
	Max_Injection_Pressure	
	Max_Switch_Over_Pressure	
	Max_Back_Pressure	
	Average_Back_Pressure	
	Barrel_Temperature_1	
	Barrel_Temperature_2	
	Barrel_Temperature_3	

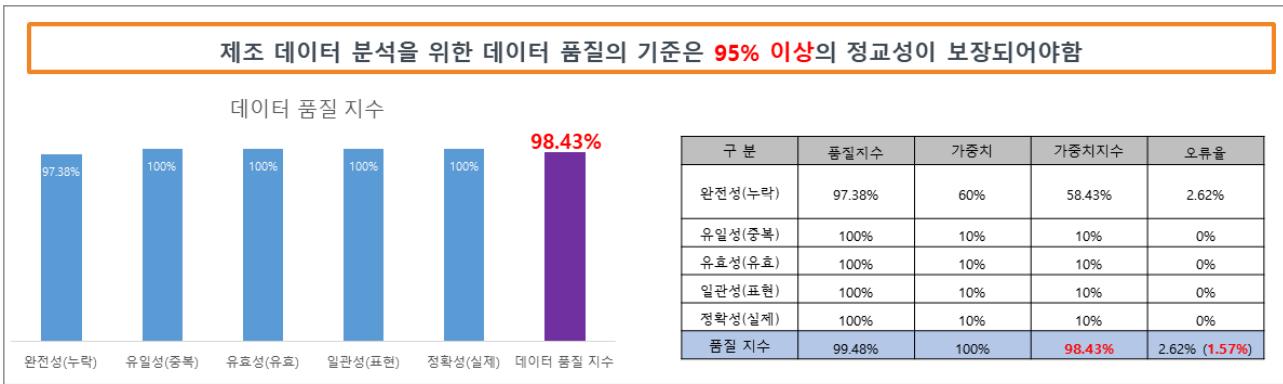
구분	구분	비고
독립변수	Barrel_Temperature_4	
	Barrel_Temperature_5	
	Barrel_Temperature_6	
	Barrel_Temperature_7	
	Hopper_Temperature	
	Mold_Temperature_1	
	Mold_Temperature_2	
	Mold_Temperature_3	
	Mold_Temperature_4	
	Mold_Temperature_5	
	Mold_Temperature_6	
	Mold_Temperature_7	
종속변수	Mold_Temperature_8	
	Mold_Temperature_9	
	Mold_Temperature_10	
	Mold_Temperature_11	
	Mold_Temperature_12	
종속변수	PassOrFail	*Label 데이터의 경우, 1/0 표시 1 : 양품, 0 : 불량품 *Unlabel 데이터의 경우, 'Null(비어있음)' 표시

\* 종속변수, 독립변수에 대한 설명은 위 1차 가공 데이터, 2차 가공 데이터에서 정리된 것과 같다

### 3) 데이터 (품질) 전처리

- 데이터 품질 전처리 목적

- 실제 공정에서 발생하는 데이터들은 값이 의미 없거나 누락 및 오타가 발생하여 데이터 품질이 떨어진다.
- 아무리 좋은 분석 방법론을 가지고 있더라도 품질이 낮은
- 데이터를 이용하면 좋은 결과를 얻을 수 없기 때문에, 데이터
- 품질 전처리는 데이터 분석에서 가장 중요한 단계이다.
- 따라서 데이터들의 5가지 품질 지수를 파악하고, 데이터 전처리를 통해 품질 지수를 향상시킨다.



[그림 15] 데이터 품질 지수

#### • 데이터 품질 지수

- 완전성(Completeness) : 필수항목에 누락이 없어야 한다.
- 유일성(Uniqueness) : 데이터 항목은 유일해야 하며 중복되어 서는 안된다.
- 유효성(Validity) : 데이터 항목은 정해진 데이터 유효범위 및 도메인을 충족해야 한다.
- 일관성(Consistency) : 데이터가 지켜야 할 구조, 값, 표현되는 형태가 일관되게 정의되고, 서로 일치해야 한다.
- 정확성(Accuracy) : 실제 존재하는 객체의 표현 값이 정확히 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_Name	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.

[그림 16] 데이터 유효성 검사

### - 데이터 품질 지수 : 세부 설명

◦ 완전성 품질 지수 =  $((1-\text{결측치})/\text{전체 데이터 수}) \times 100$

- ① Null 값이 30% 이상인 데이터들은 데이터의 완전성이 떨어지기 때문에 열별 Null값의 비율을 확인하여 삭제한다.
- ② 데이터의 결측치를 확인하기 위해 isnull()함수를 사용한 뒤 sum( )함수를 이용하여 총 결측치 개수를 구한다
- ③ 구한 결측치의 개수를 이용하여 완전성 품질 지수를 구한다.

◦ 유일성 품질 지수 =  $((1-\text{중복데이터 수})/\text{전체 데이터 수}) \times 100$

- ① 이 데이터는 유일한 값을 가지는 열이 존재하지 않으므로 유일성을 판단하지 않는다.
- ② 유일성을 판단하고 싶다면 idx와 workingtime 두 열을 이용하여 합성키를 만든 뒤 기본키로 설정하여 중복을 판단하면 된다.

◦ 유효성 품질 지수

- ① 데이터가 유효범위 내에 있는가?
- ② 데이터가 형식에 맞는가?
- ③ 수집된 날짜 안에 들어가 있는가? 등을 검증하는 것이다.

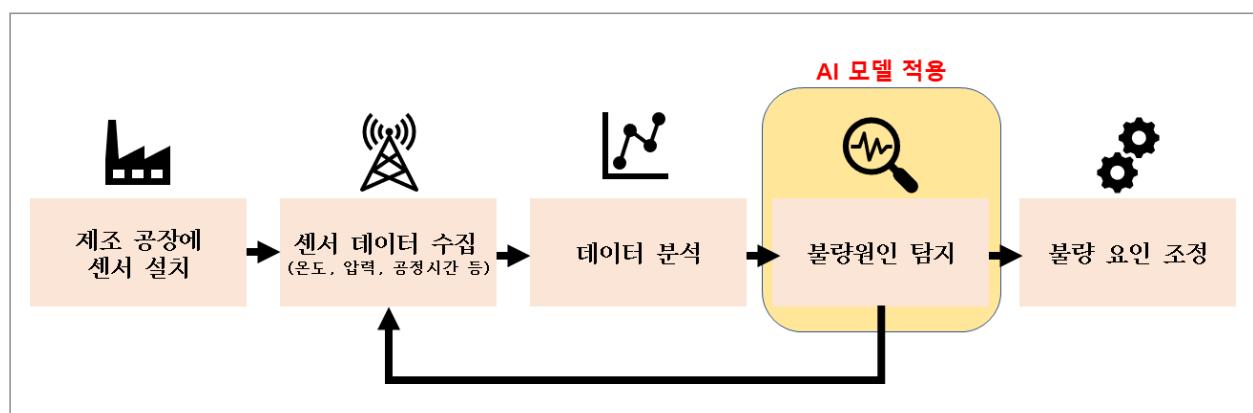
◦ 일관성 품질 지수 =  $(\text{일관성 만족 데이터 수}/\text{전체 데이터 수}) \times 100$

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

◦ 정확성 품질 지수 =  $(1-(\text{정확성 위배 데이터 수}/\text{전체 데이터 수})) \times 100$

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

## 2.2 분석 모델 소개



[그림 17] : 제조 공정에서의 데이터 흐름 및 AI 모델 적용 단계 도식화

## 2.2.1 지도 학습 방법론 (이상 탐지)

### 1) AI 분석모델

- 해당 AI 방법론(알고리즘) 선정 이유 기술

- 사출성형기 데이터 분석 방법 : 이상 탐지(Anomaly detection)

- 이상 탐지는 전체 데이터에서 비정상 샘플(불량품)을 찾아내는 방법론으로 종종 사용된다. 대부분의 제조업에서 나오는 샘플데이터의 경우 정상 샘플에 비해 비정상 샘플의 수가 크게 적다는 점에 착안하여 이상 탐지 모델을 해결책으로 제시한다. 이상(Anomaly)을 어떻게 정의하고 접근하는지는 아래에서 서술한다.

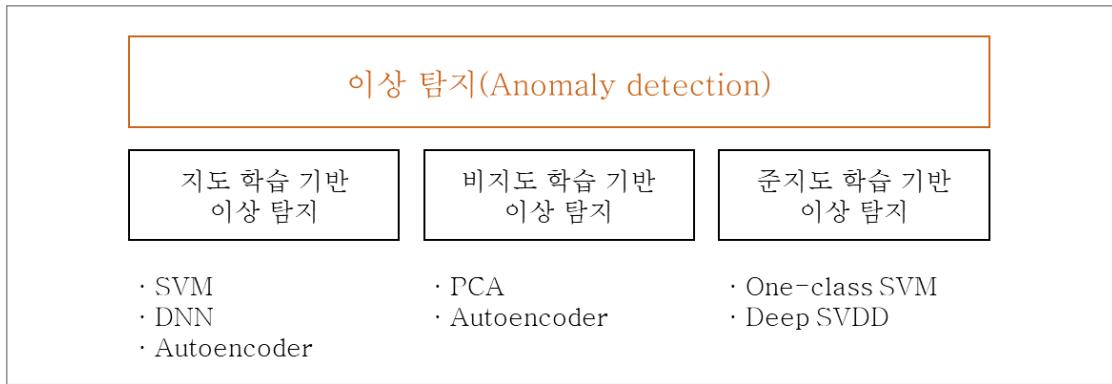
- 적용하고자 하는 AI 분석 방법론(알고리즘의 구체적 소개)

- 이상 탐지란?

- 이상 탐지(anomaly detection)란 자료에서 예상과는 다른 패턴을 보이는 개체 또는 자료를 찾는 것을 일컫는다. 정상(normal) 데이터와 비정상(abnormal) 데이터를 구별해내는 문제를 해결하는 방법론이다. 이상 탐지는 기존 통계 기법부터 머신 러닝 및 딥러닝에 이르기까지 다양한 방법론을 제시함으로써 여러 응용 분야에서 다양한 형태로 연구되고 있다. 이상 감지 기술이 적용 되는 상황은 비정상 데이터의 수가 정상 데이터에 비해 적은 경우가 대다수이기 때문에 정상 데이터를 잘 학습하고, 학습된 정상 데이터의 특징과 구분되는 비정상 데이터의 특징을 추출할 수 있는 모델과 학습 알고리즘이 필요하다. 데이터 집합에서 비정상적인 데이터를 제거하기 위해 전처리(pre-processing) 시 종종 사용된다.

- 딥러닝을 이용한 이상 탐지와 그 종류

- 딥러닝은 머신 러닝의 하위 개념으로, 딥러닝의 등장으로 인해 인공지능 및 머신 러닝 분야의 실용성이 강조되었다. 일반적인 머신 러닝 알고리즘을 사용 할 때에는 따로 특징 추출(feature extraction)을 위한 전처리 과정을 거쳐야 하는 반면, 딥러닝은 전처리 과정이 전체 학습 프로세스에 포함되어 있어보다 효율적이다. 일반적인 머신 러닝 알고리즘으로는 분석하기 어려운 복잡한문서, 소리, 이미지, 영상 등의 비정형(unstructured) 데이터를 사용할 수 있으며, 그 양이 많을수록 시스템의 성능이 좋아진다. 학습데이터의 레이블 유무에 따라 크게 지도 학습(Supervised Learning), 비지도 학습(Unsupervised Learning), 준지도 학습(Semi-Supervised Learning)으로 분류될 수 있다. 위 세 종류의 학습 방식을 기반으로 이상 탐지 모델을 구분할 수 있다.



[그림 18] 레이블 유무에 따른 이상 탐지 분류 및 알고리즘

### - 지도 학습 기반 이상치 탐지 (Supervised Learning based Anomaly detection)

#### i. 설명

- 사용되는 모든 데이터들, 정상 샘플과 비정상 샘플들이 label이 존재 할 경우 지도 학습 방식으로 훈련이 이루어지기 때문에 ‘지도 학습 기반 이상치 탐지’라고 한다. 즉, 정상 샘플과 비정상 샘플들의 패턴과 특징들을 학습하고 입력이 들어왔을 때, 그 입력이 정상 샘플인지 비정상 샘플인지 판단하는 분류 문제와 매우 유사하다고 볼 수 있다.

#### ii. 장단점

장점	단점	해결책
모든 데이터들이 정상/비정상 샘플들로 나누어져 있기 때문에, 다른 학습 방식들보다 높은 정확성을 가진다.	대체적으로 이상 탐지에 사용되는 데이터들의 경우 정상 샘플이 비정상 샘플보다 월등히 많기 때문에 데이터 불균형 문제가 빈번하게 발생한다.	많은 비정상 샘플의 확보를 위한 데이터 증강 기법, 입력과 출력 사이의 차이를 정의하는 비용 함수 설계 등이 있다.

#### iii. 대표 알고리즘

- 지도 학습 기반 이상치 탐지 시스템에서는 훈련 방식은 흔한 기계 학습의 분류나 예측 모델을 만드는 것과 크게 다르지 않다. 따라서 정상/비정상 샘플에 대한 데이터만 충분히 가지고 있다면 기계 학습 기법으로는 서포트 벡터 머신(Support Vector Machine, SVM), 랜덤 포레스트(Random Forest), 딥러닝 기법으로는 심층 신경망(Deep Neural Network, DNN), 오토인코더(AutoEncoder, AE) 등이 사용될 수 있다.
- 비지도 학습 기반 이상치 탐지 (Unsupervised Learning based Anomaly detection)

### i. 설명

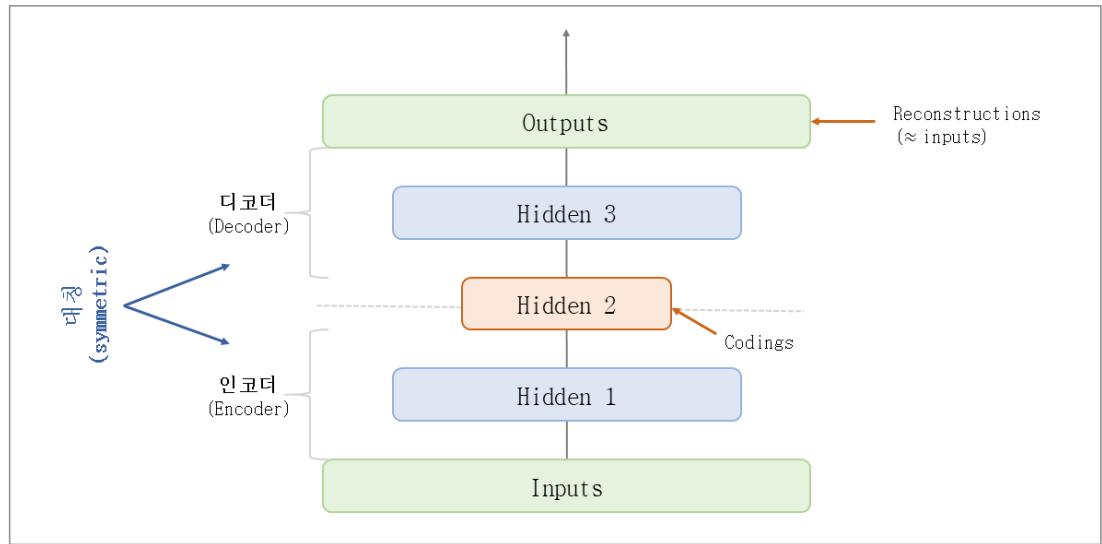
- 지도 학습 기반의 이상 탐지의 경우, 정상과 이상을 구분할 수 있는 레이블을 확보하는 과정이 필요하기 때문에 비용 및 시간 측면에서 비효율적이다. 비지도 학습기반의 이상 탐지는 대부분의 데이터가 정상 샘플이라는 가정을 하여 레이블 없이 학습을 시키는 방법론이다. 고유 데이터특성을 학습함으로써 정상 데이터와 비정상 데이터를 분리해낼 수 있다. 비지도 학습 기반의 이상 탐지 모델은 아래의 특징을 가지고 있다고 가정한다.
- 정상과 불량 영역의 잠재 특징 공간(latent feature space)을 구별할 수 있다.
- 소수의 데이터 샘플이 비정상, 대부분의 데이터 샘플은 정상이다.
- 거리 혹은 밀도와 같은 데이터 집합의 내적 특징을 기반으로 하여 이상 점수(anomalous score)를 산출해낼 수 있다.

### ii. 장단점

장점	단점
데이터에 정상/불량 여부의 레이블이 필요하지 않아 비용 측면에서 효율적이다.	<ul style="list-style-type: none"><li>- 복잡하고 고차원 공간에서 데이터 내 공통점을 학습하는 것에 어려움이 있어 양품 판별 정확도를 계산하기 어렵다.</li><li>- 직접 조정해야 하는 하이퍼파라미터(hyper-parameter) 및 데이터 손상에 민감하다.</li></ul>

### iii. 대표 알고리즘

- 주성분 분석(Principal component analysis, PCA)은 차원 축소 및 복원을 통한 비정상 샘플을 검출하며, 선형 구조의 모델이다. 이러한 전통적인 방법보다 심층 신경망을 가진 모델이 더욱 좋은 성능을 보이는데, 그중 오토인코더 기반의 모델이 비선형의 구조를 가진 이상치 탐지에서는 가장 흔히 사용되는 구조이다. 오토인코더란 데이터들을 효율적으로 표현할 수 있는 coding을 학습할 수 있는 인공 신경망이며, 특성 추출의 목적으로 사용될 수 있다. 오토인코더가 입력을 받아 효율적인 내부 표현으로 바꾸고 이를 다시 해석하여 입력 값과 가까워 보이는 출력을 한다. 오토인코더는 인코더(encoder)와 디코더(decoder), 두 부분으로 구성되어 있다.
- 인코더 : 인지 네트워크(recognition network)라고 하며, 입력을 내부 표현으로 변환한다.
- 디코더 : 생성 네트워크(generative network)라고 하며, 내부 표현을 출력으로 변환한다.



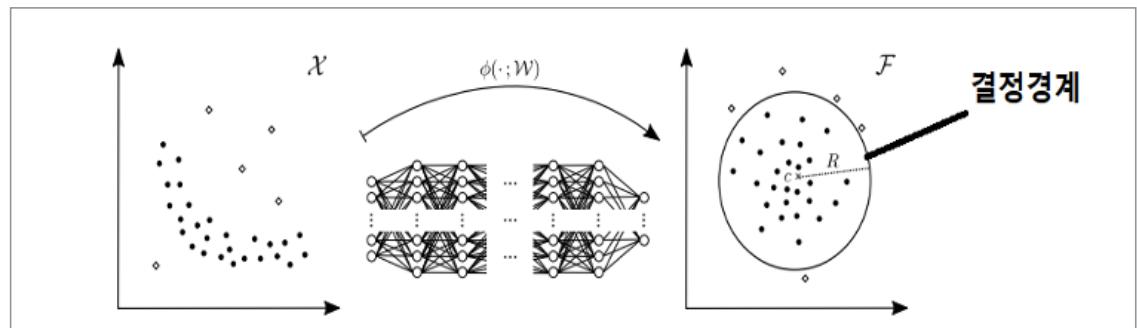
[그림 19] 오토인코더의 구조

입력을 재구성하여 출력을 만들어내기 때문에, 이때 발생하는 비용을 재구성 손실 혹은 복원 오차(reconstruction error)라고 한다. 입력과 복원된 출력의 복원 오차가 특정 역치 값을 넘어설 경우 이상, 작을 경우 정상으로 간주함으로써 불량 유무를 확인할 수 있다.

- 준지도 학습 기반 이상치 탐지 (Semi-Supervised Learning based anomaly detection)

### i . 설명

- 사용되는 데이터들이 부분적으로 레이블링이 되어 있는 경우, 준지도 학습 방식으로 훈련을 시킬 수 있다. 실제 데이터는 레이블링이 되어 있는 경우보다, 안 되어 있는 경우나 부분적으로 되어 있는 경우가 더 많다. 따라서 데이터가 부분적으로만 레이블링이 되어 있거나, 데이터 불균형 문제가 너무 심각하여 지도 학습 방식을 사용하기 힘들 경우, 정상적인 샘플들만 사용하여 학습을 시키기도 하는데 이런 방식을 ‘준지도 학습 기반 이상 탐지’라고 한다. 즉, 이 방법은 정상 샘플들만 사용하여 정상 샘플들이 분포하고 있는 지역의 결정 경계(Decision boundary, Discriminative boundary)를 학습하여 이 밖의 데이터들은 모두 비정상 샘플이라고 판별하는 방식의 모델이다.



[그림 20] 결정 경계 학습

## ii. 장단점

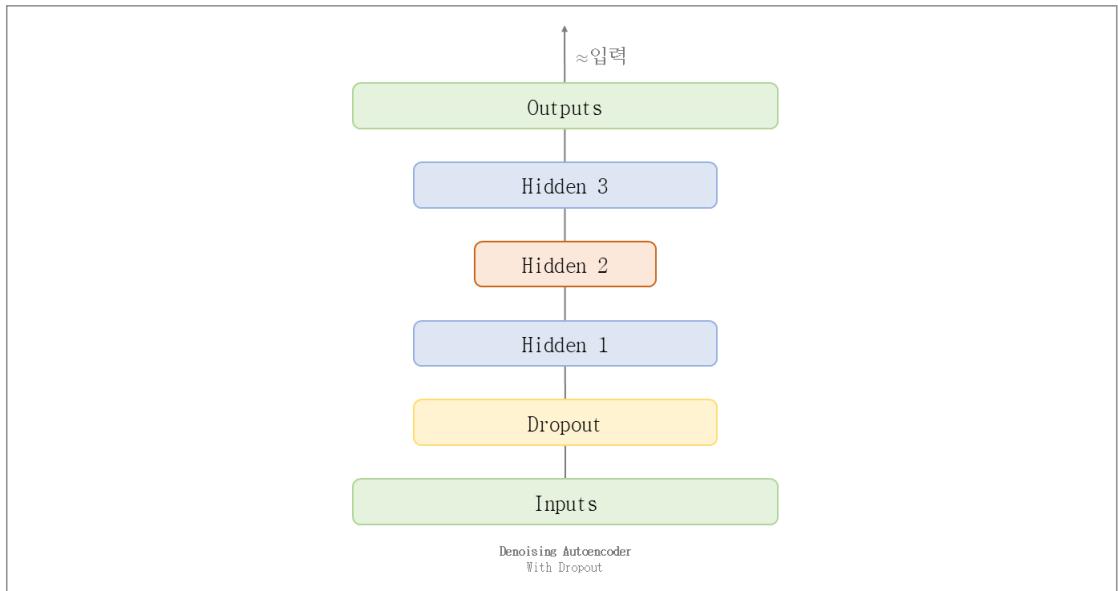
장점	단점
<p>정상 샘플만 있어도 학습이 가능하다. 비정상 샘플은 정상 샘플에 비해 월등히 데이터 수가 적으며, 비정상 샘플을 얻는 과정에서 많은 비용이 발생하므로 정상 샘플만 사용한 학습 방식은 비용 측면에서 효율적이다.</p>	<ul style="list-style-type: none"><li>- 비정상 샘플에 대한 정확한 레이블이 존재하지 않아 비정상 샘플에 대한 특성을 정확히 학습하기 어렵고, 그로 인한 지도 학습 기반 방식에 비해 상대적으로 낮은 정확도를 보인다.</li><li>- ‘정상 샘플이 분포하는 경계 밖의 데이터는 비정상 샘플이라고 판별한다’는 가정은 크게 편향된 결과를 가져올 수 있으며 실제로 비정상 샘플의 분포와 정상 샘플의 분포가 많은 부분 겹쳐있는 경우 비정상 샘플임에도 정상 샘플로 인식할 가능성이 높다.</li></ul>

## iii. 대표 알고리즘

- 준지도 학습기반 이상 탐지에 사용되었던 대표적인 알고리즘으로는 기계 학습 기법의 One-Class SVM이 있다. 딥러닝 기법과 결합하여 심층신경망을 활용한 Deep Support Vector Data Description(Deep SVDD) 등이 있다.

### • AI 분석 방법론(알고리즘) 구축 절차 설명

- **사용 알고리즘:** 지도 학습 기반의 이상탐지: 잡음제거 오토인코더(Denoising AutoEncoder)
  - 잡음제거 오토인코더(Denoising AutoEncoder)란 오토인코더가 의미 있는 특성(feature)을 학습하도록 제약을 주는 방법으로, 입력 데이터에 잡음(noise)을 추가하고 잡음이 없는 원본 입력을 복원하도록 학습시키는 것이다. 단순히 입력을 출력으로 복사하지 못하므로 데이터에 있는 패턴을 찾게 되는 방법으로, 드롭아웃(dropout)을 통해 잡음을 발생시킬 수 있다. 여기서 드롭아웃이란 입력층과 출력층 사이의 은닉층(hidden layer)이나 입력층의 일부 뉴런(유닛, 노드)을 무작위로 생략하는 것이며, 신경망 모델이 더 견고하고 안정적으로 학습할 수 있도록 도와주는 역할을 한다. 정상 데이터로만 학습된 잡음제거 오토인코더의 경우, 학습한 적이 없는 불량 데이터가 입력으로 들어왔을 때보다 정상 데이터를 입력으로 받을 때 작은 복원 오차를 가지게 되어 불량 유무를 분류할 수 있게 된다.



[그림 21] 잡음제거 오토인코더의 구조

## 2.2.2 준지도 학습 방법론

### 1) AI 분석모델

- 해당 AI 방법론(알고리즘) 선정 이유 기술

- 사출기 데이터 분석 방법: 준지도 학습(Semi-Supervised Learning)을 적용한 데이터 분석

- 현재 대부분의 제조기업에서 사용하는 사출기의 특성상, 하루에 사출기에서 생산되는 총 생산량, 전체 불량품의 개수 등은 확인이 가능하지만, 사출되는 제품 각각에 대한 양품 선별은 현장 전문가의 검수 외에는 알기 어렵다. 이러한 제품 각각에 대한 양불 여부는 각 제품의 데이터 중에서 클래스 변수(목표치)라고 부를 수 있는데, 이러한 클래스 변수를 만들어내는 과정은 큰 비용이 들고 긴 시간이 소요된다. 따라서 사출기 관련 데이터에서는 클래스 변수가 없는 데이터가 대다수인 상황에서 사출기의 불량요인을 효율적으로 예측할 수 있는 준지도 학습(Semi-Supervised Learning)을 적용한다.

- 적용하고자 하는 AI 분석 방법론(알고리즘)의 구체적 소개

- 준지도 학습이란?

- 기계 학습은 주어진 학습 데이터의 클래스 변수 유무에 따라 크게 지도 학습(Supervised-Learning), 비지도 학습(Unsupervised-Learning), 준지도 학습(Semi-Supervised Learning)으로 분류된다. 지도 학습은 클래스 변수가 존재하는 데이터를 사용하여 모델을 학습시켜 원하는 클래스 변수를 예측하도록 하는 학습 방

법이다. 반면에 비지도 학습은 클래스 변수가 존재하지 않는 데이터만을 사용하며, 데이터의 주어진 특성을 활용하여 모델을 학습시킨다. 준지도 학습은 클래스 변수가 있는 데이터와 그렇지 않은 데이터 모두를 활용하여 모델 학습을 수행하는 방법이다.

준지도 학습(Semi-Supervised Learning)은 분류 및 회귀 등 다양한 목적을 위해 사용되며, 클래스 변수가 주어진 데이터와 그렇지 않은 데이터 모두를 사용할 수 있다는 장점이 있다. 일반적인 지도 학습의 경우, <표 1>과 같이 데이터의 클래스 변수(예: 성별)를 예측하기 위해 데이터의 다른 변수(예: 키, 체중)들을 사용한다. 그러나 실제로는 사출성형뿐만 아니라 대부분의 기계 학습이 응용되는 분야에서 이러한 라벨링을 하는 것은 큰 비용과 시간이 소요된다. 따라서 비교적 쉽게 얻을 수 있는, <표 2>와 같은 클래스 변수가 주어지지 않은 데이터를 효율적으로 활용하여 클래스 변수가 주어진 데이터만으로 학습된 모델보다 나은 성능을 내는 것이 준지도 학습의 목적이다.

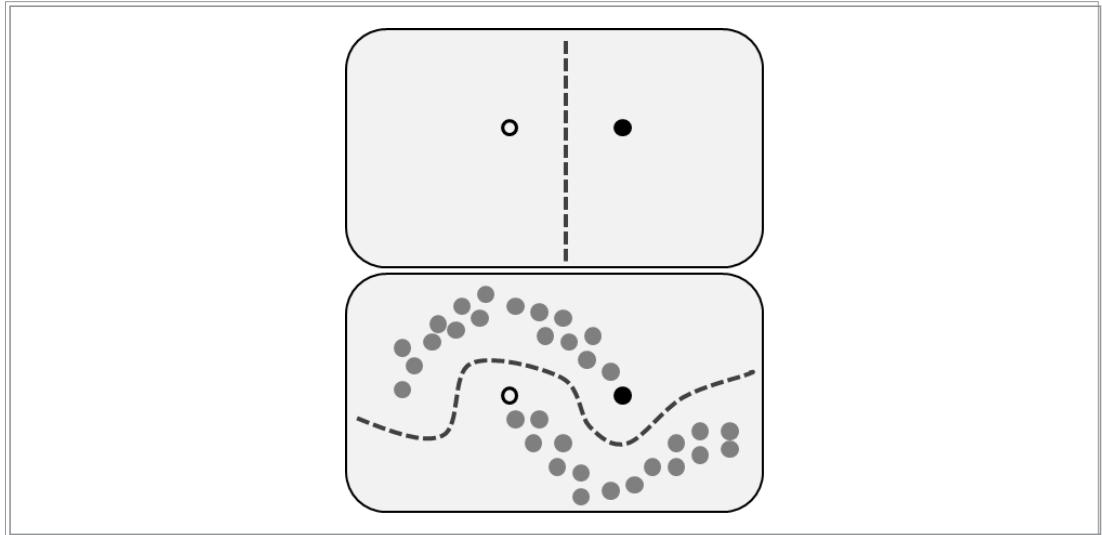
다양의, 클래스 변수가 주어지지 않은 데이터도 사용하여 학습시킨 모델은 소량의 클래스 변수가 주어진 데이터만을 사용해 학습시킨 모델보다 좋은 결과를 가져다주곤 하는데, 그 예시가 바로 [그림 22]이다. 흰 동그라미는 여성, 검은 동그라미는 남성인 데이터 포인트라고 가정하자. 두 개의 클래스 변수, 즉 성별이 주어진 두 개만의 데이터 포인트를 가지고 성별을 분류할 수 있는 모델을 만들면, 모델은 학습을 통해 점선과 같은 결정 경계를 찾게 된다. 그러나 소수의 데이터만을 가지고 만든 결정 경계는 부정확할 수 있다. 추가적으로 두 번째 그림과 같이 회색의 클래스 변수가 없는 데이터들을 학습에 사용하게 되면, 보다 실제와 비슷한 결정 경계를 가지게 되고, 모델을 더 적합하게 학습시킬 수 있다.

키	체중	성별
150	40	여성
180	80	남성

[표 1] 클래스 변수가 주어진 데이터 예시

키	체중	성별
155	45	X(모름)
160	45	X(모름)
165	60	X(모름)
175	80	X(모름)
190	90	X(모름)

[표 2] 클래스 변수가 주어지지 않은 데이터 예시



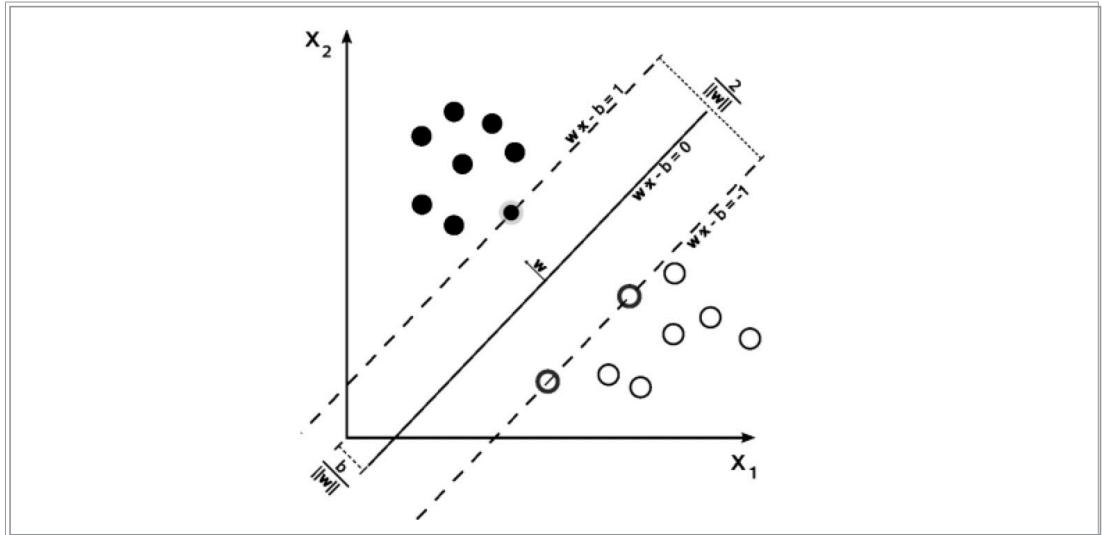
[그림 22] 준지도 학습의 예시

#### • AI 분석 방법론(알고리즘) 구축 절차 설명

- 준지도 학습 상황에서 활용되는 AI 분석 알고리즘
  - 준지도 학습은 전통적인 지도 학습 및 비지도 학습에 사용되는 AI 방법론들을 동일하게 이용하여 구현할 수 있다. 본 사출기 데이터 분석에서는 기계 학습 분야에서의 범용적인 알고리즘 4가지를 사용한다. 전통적인 기계 학습 알고리즘인 서포트 벡터 머신(Support Vector Machine), 랜덤 포레스트(Random Forest), 가우시안 나이브 베이즈(Gaussian Naïve Bayes)와 최근 각광을 받고 있는 심층 신경망(Neural Network)을 사용한다. 다양한 기계 학습 모델을 적용함으로써 주어진 사출기 데이터를 최대한 활용하는 AI 방법론을 탐색한다.
- 서포트 벡터 머신(Support Vector Machine)
  - 서포트 벡터 머신은 널리 사용되는 기계 학습 알고리즘으로, 분류와 회귀를 위해 사용되는 전통적인 방법론이다. 데이터를 사용해 모델을 학습시킨다는 것은 서포트 벡터 머신이 두 가지 종류의 데이터, 즉 양품과 불량품을 구분할 수 있는 초평면(hyperplane)이라고도 불리는 결정 경계를 찾게 한다는 것이다. 이러한 초평면에서 가장 가깝게 위치한 각 집단(양품, 불량품)에서의 데이터를 서포트 벡터라고 부른다. [그림 23]에서의 각 점선 위에 위치한 데이터 포인트들이 바로 서포트 벡터이다. 이렇게 선택된 두 개의 서로 다른 집단에 속한 서포트 벡터 사이의 거리를 마진(margin)이라고 부르며 [그림 23]에서 두 점선 사이의 수직거리로 나타낸다. 모델은 학습 과정에서 마진을 최대화하는 방향으로, [그림 23]의 실선으로 표현되는 초평면을 바꾸어가며 최적의 결정 경계를 찾게 된다.

서포트 벡터 머신은 특별히 데이터를 다른 차원으로 사상시키기도 한다. 데이터의 변수의 개수, 즉 사출기 데이터의 압력, 온도 등의 개수를 차원 수라고 볼 수 있는데 데이터를 선형으로 사상한 뒤 모델로 하여금 초평면을 찾게 하는 것이 기본적이다.

또한, 커널(kernel)이라고 불리는 기법을 활용하여 고차원의 공간으로 데이터를 사상시킨 후 사상된 고차원의 공간에서 초평면을 찾도록 하는 방법을 사용하기도 한다. 많이 사용되는 커널로는 다항식(polynomial) 커널, 방사 기저 함수(radial basis function) 커널 등이 존재한다. 모델 학습 시에는 하이퍼 파라미터라고 불리는 사용자가 조정가능한 값들이 존재한다. 서포트 벡터 머신의 경우 C값과 gamma값 등을 조정가능한데, C값은 모델의 마진을 계산하는 식을 조정함으로써 모델의 적합도를 조절하는 데에 사용되고 gamma값은 커널의 식을 조정하는 데에 사용된다.



[그림 23] 서포트 벡터 머신

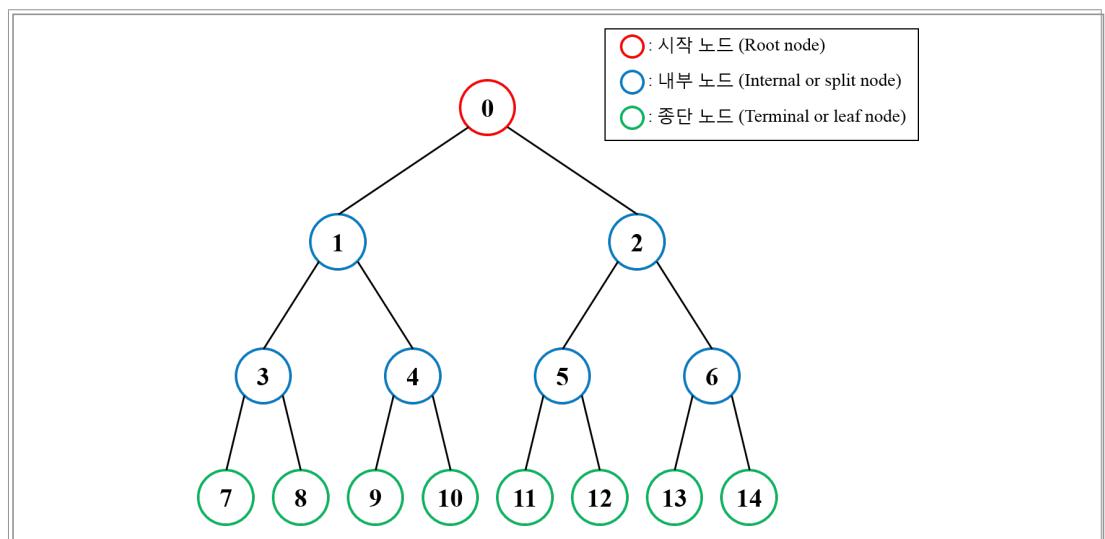
#### - 랜덤 포레스트(Random Forest)

- 랜덤 포레스트는 결정 트리(decision tree) [그림 15]라는 기계 학습 알고리즘을 활용하는 범용적인 모델로서 분류와 회귀 목적으로 자주 사용된다. 기계 학습에서 여러 학습 알고리즘을 사용하고 취합함으로써 각각의 모델을 따로 사용하는 경우보다 뛰어난 성능을 얻는 방식을 앙상블(ensemble) 학습방법이라고 한다. 랜덤 포레스트는 이러한 앙상블 학습방법을 활용하여 높은 정확도와 좋은 일반화 성능을 내는 것으로 알려져 있다.

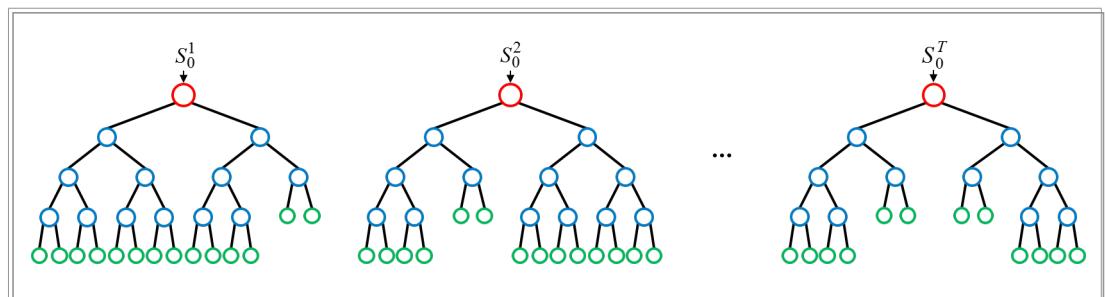
랜덤 포레스트는 여러 개의 결정 트리 [그림 25]를 구성하고 각각의 결정 트리로부터의 결과값을 취합하는 과정을 통해 학습을 진행한다. 결정 트리는 [그림 24]와 같은 구조를 가지며 계층적인 구조로 이루어진 노드(node)와 에지(edge)의 집합으로 구성된다. 이를 그대로 결정을 내리기 위해 스무고개처럼 결정 과정을 나무 형태로 나타낸 것이라고 이해하면 편하다. 각 결정 트리는 학습 단계에서 노드별 분할 함수, 즉 결정 단계별 기준을 최적화하는 방향으로 학습된다. 이러한 학습 과정의 특성 때문에 학습이 완료된 트리는 항상 동일한 결과를 내게 된다.

랜덤 포레스트는 이러한 방식으로 학습되는 결정 트리를 임의의 개수만큼 여러 개 사용하게 되는데, 각 결정 트리가 서로 다른 특성을 가지도록 학습시킴으로써 모델의

전체적인 일반화 성능을 제고한다. 이 방법을 배깅(bagging)이라 부른다. 배깅을 적용하여 기존의 데이터를 여러 개의 조금씩 다른 데이터로 분할한 후, 각 결정 트리를 서로 다른 데이터에 학습시킨 다음 서로 다른 특성을 학습하게 된 결정 트리들의 결과값을 결합하는 방법을 일컫는다. 조정 가능한 하이퍼 파라미터로는 랜덤 포레스트 내부의 결정 트리의 개수, 결정 트리의 최대 깊이(높이), 가장 끝 노드의 최대 수 등 여러 조정 가능한 값이 존재한다. 랜덤 포레스트는 결국 서로 다른 결정 트리의 결과값을 종합함으로써 모델의 일반화 성능 및 견고함을 제고한다는 장점이 있다. 그러나 결정 트리의 특징인 과대적합(overfitting) 및 비정상 데이터에 취약하다는 점은 여전히 랜덤 포레스트에 내재한 단점이다.



[그림 24] 결정 트리



[그림 25] 여러 개의 결정 트리로 구성된 랜덤 포레스트

### - 가우시안 나이브 베이즈(Gaussian Naïve Bayes)

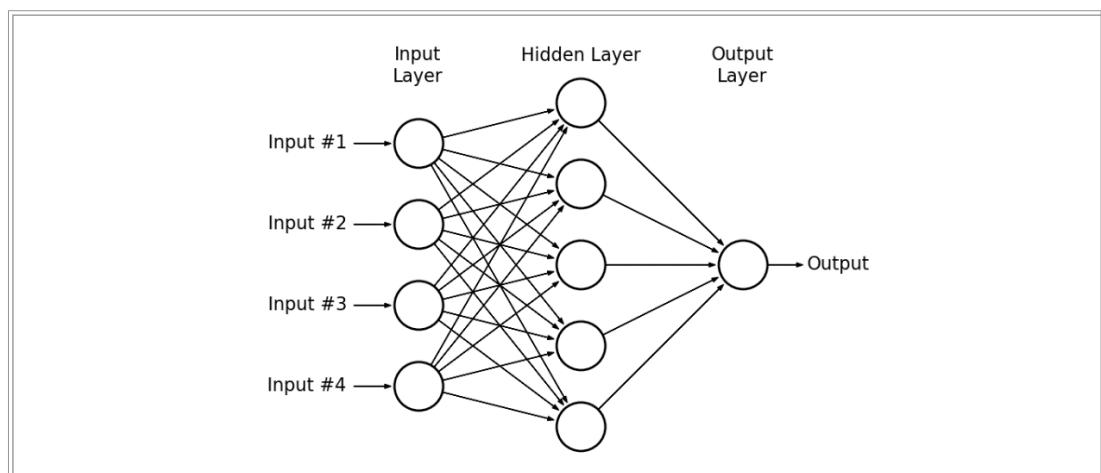
- 가우시안 나이브 베이즈는 베이즈 정리(Bayes theorem)를 적용한 확률적인 분류 알고리즘이다. 데이터에 여러 변수가 존재할 때 일반적으로 대부분의 변수는 서로 독립적이지 않다. 하지만, 나이브 베이즈는 모든 특성이 서로 독립적이라는 강한 가정하에 생성된 조건부 확률 모델이다. 특성이 연속적인 변수일 경우에는 각 특성이 정규분포(Gaussian)를 따른다고 가정하게 되며 이러한 가정하에 학습되는 모델을 가우시안 나이브 베이즈라고 부른다. 나이브 베이즈는 비교적 적은 데이터를 사용하

여도 필요한 파라미터를 충분히 추정할 수 있다는 장점이 있다. 이는 학습 시에 주어진 데이터를 사용하여 확률을 계산하는 방식으로 모델이 학습되기 때문이며, 실제로 강한 가정을 취함에도 불구하고 준수한 성능을 보이기 때문에, 효율적인 학습이라고 불린다.

#### - 심층 신경망(Deep Neural Network)

- 심층 신경망은 신경망이라 불리는 구조를 연속적으로 깊게 쌓은 후 분류 및 회귀 등의 다양한 목적을 위해 학습시키는 AI 알고리즘이다. 신경망은 [그림 25]와 같이 입력층(input layer), 은닉층(hidden layer), 출력층(output layer)으로 구성된다. 은닉층은 입력층의 데이터를 받고 행렬 곱 등의 연산을 거친 후 출력값을 배출한다. 이러한 신경망은 선형적인 변환뿐만 아니라 여러 활성화 함수(activation function)라 부르는 비선형적인 연산을 수행하기도 한다. 여러 겹의 은닉층을 가진 신경망은 수많은 함수를 구현할 수 있는 장점이 있으며, 일반적으로 우수한 성능을 보이는 것으로 알려져 있다.

신경망은 경사 하강법(gradient descent)을 이용하여 학습시킬 수 있다. 또한, 신경망의 예측 결과값과 데이터의 실제값이 어느 정도 일치하는지 등의 기준으로 신경망의 성능을 평가하게 되는데 이러한 평가 수치를 손실 함수(loss function)라고 부른다. 각 신경망의 파라미터(모수)를 변화시킴에 따라 이 손실 함수의 값은 변화하게 되므로, 모델의 학습 과정에서는 손실 함수를 최소화하는 신경망의 모수를 찾는 방식으로 학습을 진행하게 된다. 분류를 위한 심층 신경망은 임의의 다수의 신경망을 연속적으로 쌓고, 최종적으로 클래스 변수(양품 혹은 불량품)인 확률을 출력하도록 구성한다. 여러 개의 신경망을 쌓게 되면 보다 복잡하거나 높은 차원에 존재하고 결정 경계를 찾을 수 있으므로 최근 신경망 연구의 대다수는 매우 깊은 모델을 구축하여 사용한다. 신경망의 구조적 특성상 노드의 개수, 노드별 활성화 함수, 신경망의 개수, 손실 함수 등의 조작 가능한 값이 매우 많이 존재한다. 따라서 모델 학습 시에 주어진 데이터에 적합한 값들을 찾는 것이 중요하다.



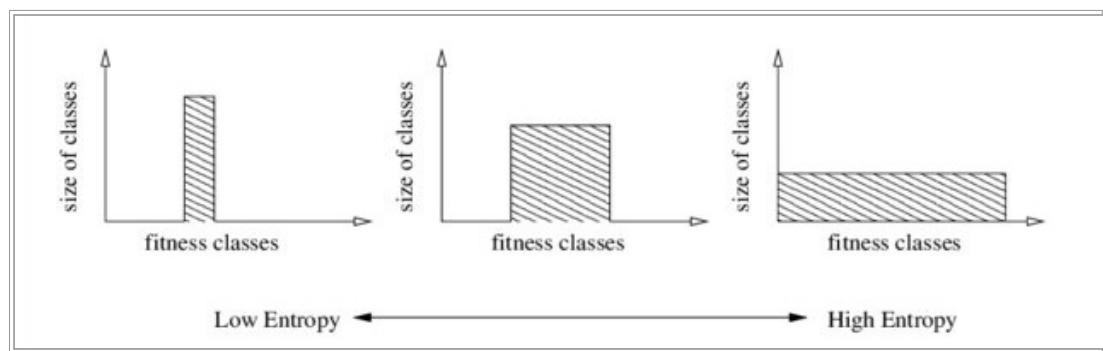
[그림 25] 신경망 예시

## - 준지도 학습의 구현

준지도 학습 상황에서 앞서 설명한 다양한 AI 방법론들을 활용하게 되는데, 사출기 데이터 및 분석 상황에 적합한 접근이 필요하다. 먼저 사출기 데이터 중에서 클래스 변수를 가진 데이터만을 사용하여 앞서 언급된 기계 학습 모델들을 학습시킨다. 이후 학습된 모델을 가지고 클래스 변수가 존재하지 않는 데이터를 사용하여 학습을 진행한다. 이러한 준지도 학습 상황에서의 구현을 위하여 특별히 엔트로피 최소화(entropy minimization)와 수도 레이블링(pseudo-labeling) 개념을 활용한다.

주어진 데이터를 사용해 학습된 기계 학습 모델은 단순히 결과값뿐만 아니라 각 결과값별 확률을 출력할 수 있다. 이러한 확률값은 학습된 모델의 신뢰도(confidence), 혹은 확신의 정도라고 해석할 수 있다. 이처럼 모델의 출력을 확률로 나타낼 때, 엔트로피(entropy)를 계산할 수 있게 된다. [그림 26]과 같이 모델이 하나의 클래스 변수에 대해 높은 확률로 예측하는 상황은 낮은 엔트로피를 가지게 되는데, 이는 동시에 모델이 확신을 가지고 예측을 한다고 이해할 수 있다. 준지도 학습 상황에서 소수의 클래스 변수를 가진 데이터로 학습된 모델을 사용하여 클래스 변수가 없는 데이터에 대해 예측할 때, 가장 확실한 예측을 하는 데이터는 실제값과 클래스 변수가 일치할 가능성이 크다고 볼 수 있다. 따라서 학습 과정에서 모델의 예측의 엔트로피가 낮은 데이터를 우선으로 선택하는 것이 준지도 학습에서 적합한 방법이다.

수도 레이블링(pseudo-labeling)은 학습된 모델을 이용해 클래스 변수가 없는 데이터에 대해 예측한 출력을 가상의 클래스 변수로 생각하고 추후 학습 시에 사용하는 방법이다. 클래스 변수가 존재하지 않는 데이터 중의 일부에 대해 새로운 가상의 클래스 변수를 부여하고 클래스 변수가 존재하는 데이터로 병합하는 과정을 반복함으로써 모델이 학습되는 데이터의 다양성을 확보하게 된다. 이 과정에서 엔트로피가 낮은 모델 예측값을 가진 순서대로 가상의 클래스 변수를 부여함으로써 가장 타당한 가상의 클래스 변수를 부여하게 된다. 이러한 방법은 클래스 변수가 없는 데이터를 활용하여 바람직한 결정 경계를 찾고, 좋은 분류 모델을 구성한다는 준지도 학습의 본래 목적과 부합하게 된다.



[그림 26] 정보 엔트로피 예시

## 2.3 분석 체험

### 2.3.1 지도 학습 활용 알고리즘 (이상 탐지) – 지도 학습 적용을 위한 모델 구현 과정

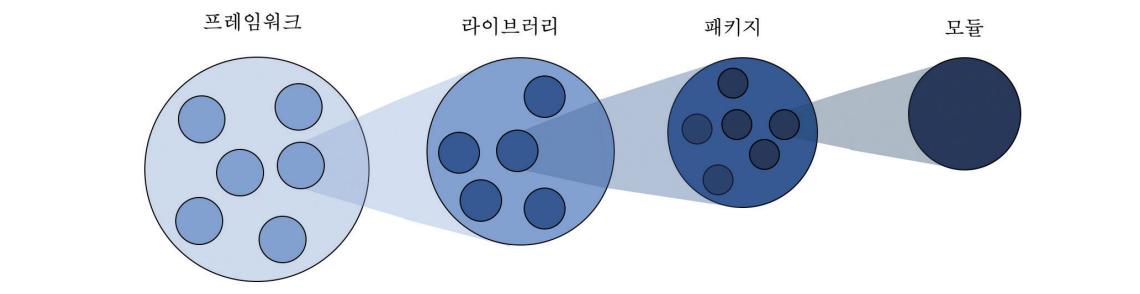
#### 1) 필요 SW, 패키지 설치 방법 및 절차 가이드

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



[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 pandas  
!pip install numpy  
!pip install matplotlib  
!pip install scikit-learn  
!pip install keras  
!pip install datetime  
!pip install seaborn
```

\* 이번 분석을 위한 필요 패키지 및 모듈은 pandas, numpy, matplotlib, scikit-learn, keras, datetime, seaborn이기 때문에 위와 같이 시작 전에 설치를 하면 된다.

### - 모듈 설치 확인하기

```
pip list
```

cycler	0.10.0
Cython	0.29.21
cytoolz	0.10.1
dask	2.20.0
DateTime	4.3
decorator	4.4.2
defusedxml	0.6.0
diff-match-patch	20200713
distributed	2.20.0
docutils	0.16

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

## ② 패키지 및 모듈 임포트 하기

예) 다양한 임포트 방법으로 Press\_RawDataSet.xlsx를 읽어보기

### i. import

import를 사용하여 해당 모듈 전체를 임포트한다.

```
import pandas  
pandas.read_excel('./Press_RawDataSet.xlsx')
```

pandas를 임포트를 하고 ‘.’을 사용하여 pandas 의‘read\_excel’ 함수를 이용하여 ‘Press\_RawDataSet.xlsx’ 파일을 불러온다.

### ii. from, import

해당 모듈에서 특정한 타입만 임포트한다.

예) pandas에서 ‘read\_excel’만 임포트

```
from pandas import read_excel  
read_excel('./Press_RawDataSet.xlsx')
```

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

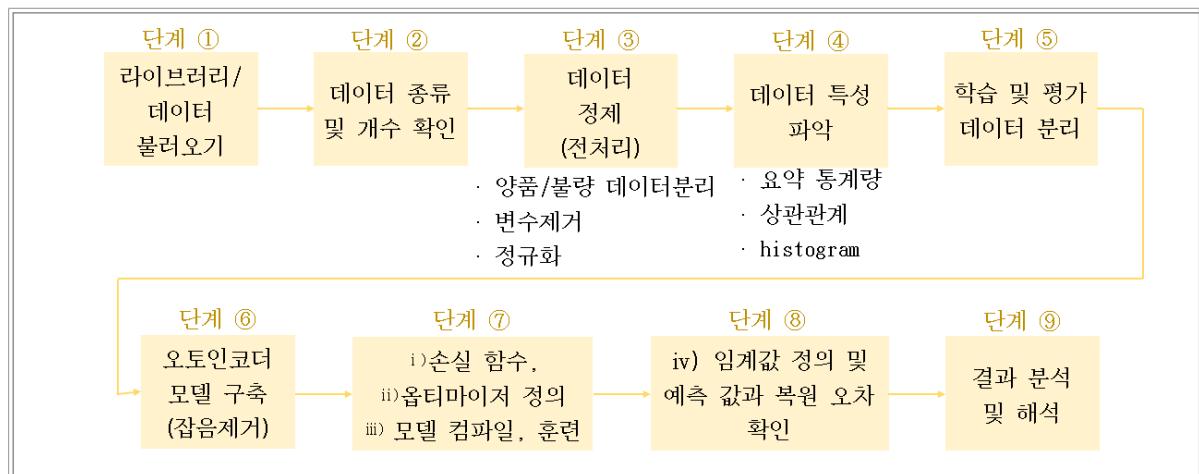
모듈 임포트할 때, 별명(alias)를 지정할 때 사용한다.

```
import pandas as pd  
pd.read_excel('Press_RawDataSet.xlsx')
```

‘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



[그림 28] 지도 학습 적용을 위한 모델 구현 과정

- i) 손실 함수: 실제값과 예측값(출력)의 차이를 수치화하는 함수. 차이를 최소화 하는 것이 목표이다.
- ii) 옵티마이저: 손실 함수의 값을 줄여 나가면서 네트워크를 업데이트하는 것. 성능을 결정한다.
- iii) 컴파일 및 훈련: 학습 데이터를 통한 신경망 학습, 평가 데이터를 통한 신경망 훈련도 검사, 임계치 설정, 예측값과의 오차 확인, 시각화를 진행한다.
- iv) 임계값: 이상치 탐지를 위해 어떤 데이터를 이상치로 판별할 지의 기준이 되는 값이다. 판별기준은 손실값이다.

## [단계 ①] 라이브러리/데이터 불러오기

### ①-1. 필요 데이터 다운로드 및 불러오기(import) 가이드

```
import numpy as np
import pandas as pd
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler
```

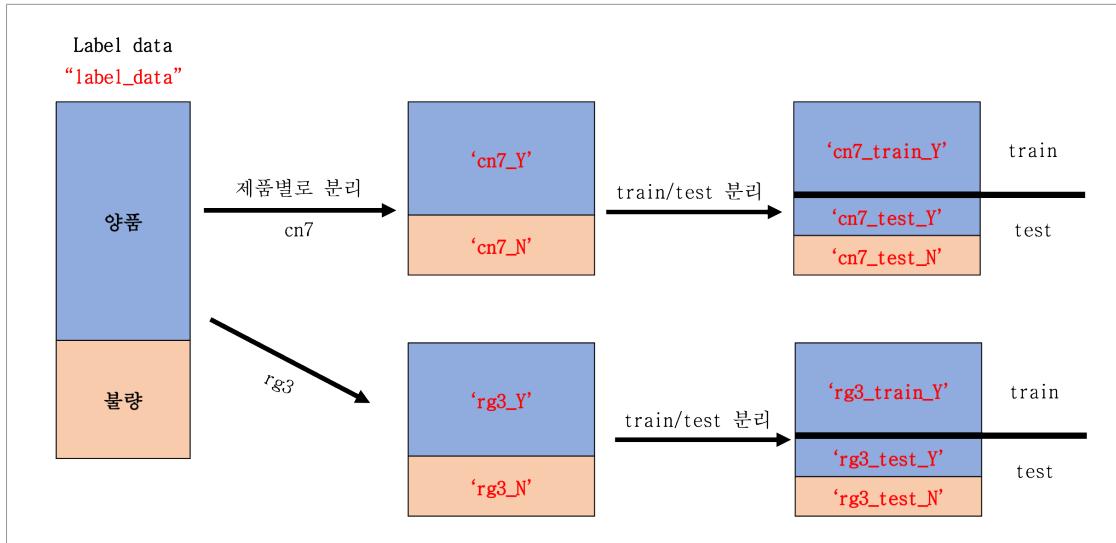
[코드 1] 필요 라이브러리 불러오기

- 레이블 된 데이터를 불러온다. csv 파일을 불러올 때에는 pandas의 ‘read\_csv(‘파일 경로’)’ 기능을 사용한다. 지도 학습 기반의 이상 탐지 알고리즘 방식이므로 클래스 변수가 존재하는 데이터를 사용한다.

```
label_data = pd.read_csv('./labeled_data.csv')
```

[코드 2] 클래스 변수가 존재하는 데이터 불러오기

아래와 마찬가지로 클래스 변수가 존재하는 데이터를 양품 및 불량, 학습 및 평가 데이터셋으로 분리할 것이다.



[그림 29] 데이터 양품/불량 분리 및 학습/평가 데이터 분리

## [단계 ②] 데이터 종류 및 개수 확인

### ②-1. 데이터 종류 및 개수 확인 가이드

- 데이터의 호기 이름값을 가진 ‘EQUIP\_NAME’의 종류 및 개수를 불러온다. 호기 종 데이터가 확보된 ‘650톤-우진2호기’만 사용한다.

```
label_data['EQUIP_NAME'].value_counts() # 결과는 아래에서 확인 가능하다.
```

```
650톤-우진2호기    7992  
650톤-우진        2  
1800TON-우진      2  
Name: EQUIP_NAME, dtype: int64
```

[코드 3] 데이터의 호기 종류 및 개수 확인

- 데이터의 제품 이름값을 가진 'PART\_NAME'의 종류 및 개수를 불러온다. 제품 중 데이터가 충분히 확보된 'CN7', 'RG3'만 사용한다.

```
label_data['PART_NAME'].value_counts() # 결과는 아래에서 확인 가능하다.
```

```
CN7 W/S SIDE MLD'G RH      3371  
CN7 W/S SIDE MLD'G LH      3365  
RG3 MOLD'G W/SHLD, RH      628  
RG3 MOLD'G W/SHLD, LH      628  
SP2 CVR ROOF RACK CTR, RH   2  
JX1 W/S SIDE MLD'G RH      2  
Name: PART_NAME, dtype: int64
```

[코드 4] 데이터의 제품 종류 및 개수 확인

### [단계 ③] 데이터 정제(전처리)

#### ③-1. 불필요 데이터 제거 가이드

- 입력 데이터에서 필요 없는 변수(feature) 제거를 위한 'make\_input' 함수를 정의 한다, 동일한 기능을 여러 번 사용할 때에는 아래와 같이 'def' 기능으로 함수를 정의 해 준다.

```
def make_input(data, machine_name ,product_name):  
    machine_ = data['EQUIP_NAME'] == machine_name  
    product_ = data['PART_NAME'] == product_name  
    data = data[machine_ & product_]  
  
    # 불필요하다고 판단된 columns  
    # pandas package의 라이브러리인 'drop'을 통해 지정한 열(변수) 제거  
  
    data.drop(['_id', 'TimeStamp', 'PART_FACT_PLAN_DATE', 'Reason',  
              'PART_FACT_SERIAL','PART_NAME','EQUIP_CD', 'EQUIP_NAME',  
              # 값이 모두 0인 변수들 제거  
              'Mold_Temperature_1',           'Mold_Temperature_2',           'Mold_Temperature_5',  
              'Mold_Temperature_6',           'Mold_Temperature_7',           'Mold_Temperature_8',  
              'Mold_Temperature_9',           'Mold_Temperature_10',          'Mold_Temperature_11',  
              'Mold_Temperature_12'],  
              axis=1, inplace=True)  
  
    return data
```

[코드 5] 필요 없는 변수 제거를 위한 'make\_input' 함수 정의

## - 함수 실행

```
# 데이터 초기 종류  
machine_name = "650톤-우진2호기"  
  
# 데이터 제품 종류  
product_name = ["CN7 W/S SIDE MLD'G LH", "CN7 W/S SIDE MLD'G RH", "RG3 MOLD'G W/SHLD,  
LH", "RG3 MOLD'G W/SHLD, RH"]  
  
# "650톤-우진2호기'의 "CN7 W/S SIDE MLD'G LH" 데이터만 변수를 제거하여 가져옴  
cn7lh = make_input(label_data, machine_name, product_name[0])  
  
# "650톤-우진2호기'의 "CN7 W/S SIDE MLD'G RH" 데이터만 변수를 제거하여 가져옴  
cn7rh = make_input(label_data, machine_name, product_name[1])  
  
# "650톤-우진2호기'의 "RG3 MOLD'G W/SHLD, LH" 데이터만 변수를 제거하여 가져옴  
rg3lh = make_input(label_data, machine_name, product_name[2])  
  
# "650톤-우진2호기'의 "RG3 MOLD'G W/SHLD, RH" 데이터만 변수를 제거하여 가져옴  
rg3rh = make_input(label_data, machine_name, product_name[3])  
  
# 동일한 제품의 LH와 RH는 합쳐줌  
cn7 = pd.concat([cn7lh, cn7rh], ignore_index=True)  
rg3 = pd.concat([rg3lh, rg3rh], ignore_index=True)  
  
# rg3에는 'Plasticizing_Position' 변수값이 모두 0이므로 추가적으로 삭제  
rg3.drop(['Plasticizing_Position'], axis=1, inplace=True)
```

[코드 6] 'make\_input' 함수 실행

## [단계 ④] 데이터 특성 파악

아래부터는 CN7 제품의 모델링만 진행하였으며, 이름만 바꾸어 동일한 방식으로 RG3의 모델링도 동일하게 진행할 수 있다.

### ④-1. CN7 제품/describe 함수를 통한 통계량 파악 가이드

- 통계 분석 알고리즘 및 시각화 도구 등을 활용한 변수들의 분포 파악, 변수 간의 관계 파악을 위한 실습 내용
  - 제품 'CN7' 데이터의 'PassOrFail' 변수가 'Y'와 'N'으로 구성되어 있으므로, 데이터 시각화를 위해 각각 0과 1로 바꾼다.

```
cn7['PassOrFail'] = cn7['PassOrFail'].replace('Y', 0).replace('N', 1)
```

[코드 7] 양품(Y)은 0으로, 불량(N)은 1로 바꾸기

- 제품 'CN7'의 클래스 변수가 존재하는 데이터의 변수별 요약 통계량을 확인한다.

```
cn7.describe() # 결과는 아래에서 확인 가능하다.
```

	PassOrFail	Injection_Time	Filling_Time	Plasticizing_Time	Cycle_Time	Clamp_Close_Time	Cushion_Position	Plasticizing_Position
<b>count</b>	6736.000000	6736.000000	6736.000000	6736.000000	6736.000000	6736.000000	6736.000000	6736.000000
<b>mean</b>	0.005790	9.580064	4.448425	16.820433	59.549314	7.113639	653.440859	68.381941
<b>std</b>	0.075876	0.180526	0.140286	0.288946	0.372267	0.075843	0.115004	0.648307
<b>min</b>	0.000000	9.360000	3.350000	16.469999	58.840000	6.070000	653.390015	59.759998
<b>25%</b>	0.000000	9.530000	4.420000	16.629999	59.480000	7.120000	653.429993	68.320000
<b>50%</b>	0.000000	9.570000	4.450000	16.820000	59.520000	7.120000	653.429993	68.360001
<b>75%</b>	0.000000	9.600000	4.480000	16.910000	59.540001	7.120000	653.440002	68.510002
<b>max</b>	1.000000	13.390000	8.270000	21.100000	64.349998	7.180000	655.000000	68.860001
	Clamp_Open_Position	Max_Injection_Speed	Max_Screw_RPM	Average_Screw_RPM	Max_Injection_Pressure	Max_Switch_Over_Pressure	Max_Back_Pressure	
	6736.000000	6736.000000	6736.000000	6736.000000	6736.000000	6736.000000	6736.000000	6736.000000
	644.899046	55.523085	30.674674	125.583744	142.090559	136.518646	37.842132	
	42.170577	1.005340	0.141148	126.836972	1.985764	0.754736	1.768085	
	69.639999	38.500000	30.299999	29.200001	140.699997	128.399994	21.700001	
	647.989990	55.099998	30.600000	29.200001	141.800003	136.300003	37.599998	
	647.989990	55.400002	30.700001	29.200001	141.899994	136.500000	37.900002	
	647.989990	55.900002	30.799999	292.399994	142.100006	136.800003	38.200001	
	647.989990	64.800003	31.200001	293.899994	169.100006	146.699997	75.199997	
	Average_Back_Pressure	Barrel_Temperature_1	Barrel_Temperature_2	Barrel_Temperature_3	Barrel_Temperature_4	Barrel_Temperature_5	Barrel_Temperature_6	
	6736.000000	6736.000000	6736.000000	6736.000000	6736.000000	6736.000000	6736.000000	6736.000000
	59.347209	275.965024	275.134961	274.858091	270.285823	254.924346	229.971512	
	3.530820	2.302366	1.887356	1.854329	2.018081	1.134566	0.428762	
	13.300000	244.699997	249.000000	249.600006	244.399994	239.699997	224.600006	
	59.400002	275.799988	275.000000	274.799988	269.700012	254.800003	229.800003	
	59.500000	276.100006	275.299988	275.000000	270.399994	255.000000	230.000000	
	59.700001	276.399994	275.500000	275.200012	271.100006	255.199997	230.100006	
	90.800003	277.899994	276.500000	276.000000	272.399994	256.299988	230.699997	
	Hopper_Temperature	Mold_Temperature_3	Mold_Temperature_4					
	6736.000000	6736.000000	6736.000000					
	66.663094	22.074228	23.473619					
	2.433782	1.171389	1.370780					
	38.500000	19.100000	20.600000					
	65.599998	21.200001	22.600000					
	67.000000	21.900000	23.299999					
	67.800003	22.799999	24.200001					
	70.599998	25.299999	27.799999					

[코드 8] 제품 'CN7'의 클래스 변수별 통계량 확인

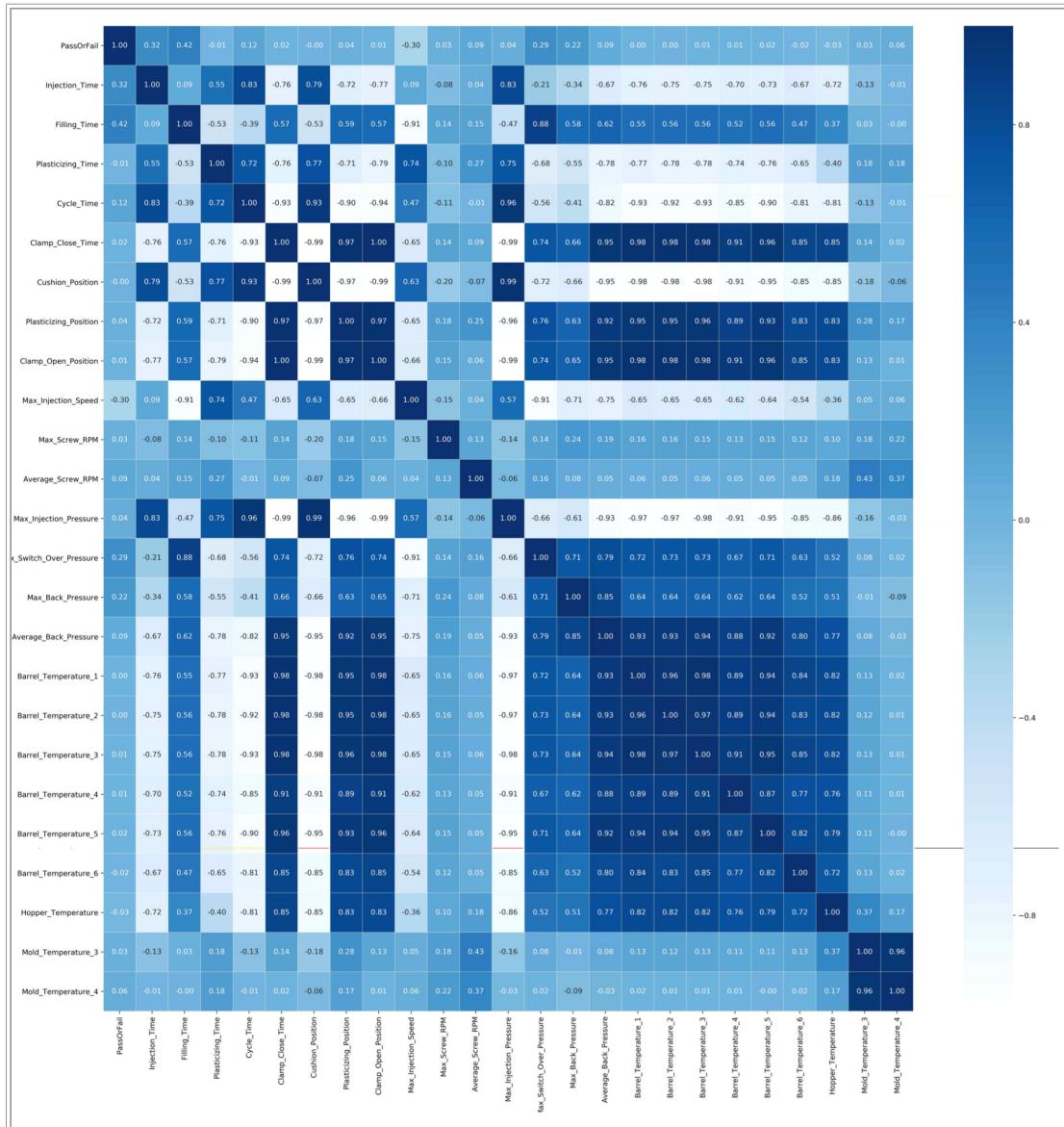
#### ④-2. CN7 제품/corr 함수를 통한 변수 간 상관관계 파악 가이드

- 제품 'CN7'의 클래스 변수가 존재하는 데이터의 변수 간 상관관계를 확인한다.
- 상관관계 분석을 통해 두 변수 간의 선형적인 관계 존재 여부를 확인할 수 있다. 분석에 사용된 상관 계수는 보편적으로 사용되는 피어슨 상관 계수로서 -1과 1 사이의 값을 가진다. 상관 계수의 절대값이 1에 가까울수록 변수 간의 선형관계가 강하다고 볼 수 있으며, 값이 양수일 때는 양적인 선형관계, 값이 음수일 때는 음적인 선형관계를 가진다. 상관관계 분석은 비선형적인 관계를 확인하기 어렵지만, 변수 간의 선형관계를 정량적으로 확인할 수 있다는 장점이 있다. 상관계수의 절댓값이 0.1과 0.3 사이이면 약한 선형관계, 0.3과 0.7 사이이면 뚜렷한 선형관계, 0.7 이상인 경우 강한 선형관계를 나타낸다고 이해할 수 있다. 다음 장에서 상관관계를 확인할 수 있는 코드 및 결과값을 확인할 수 있다.

```

plt.subplots(figsize=(25,25))
sns.heatmap(data = cn7.corr(), linewidths=0.1, annot=True, fmt = '.2f', cmap='Blues')
# 결과는 다음 장에서 확인 가능하다.

```



[코드 9] 제품 'CN7'의 클래스 변수별 상관관계 확인

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

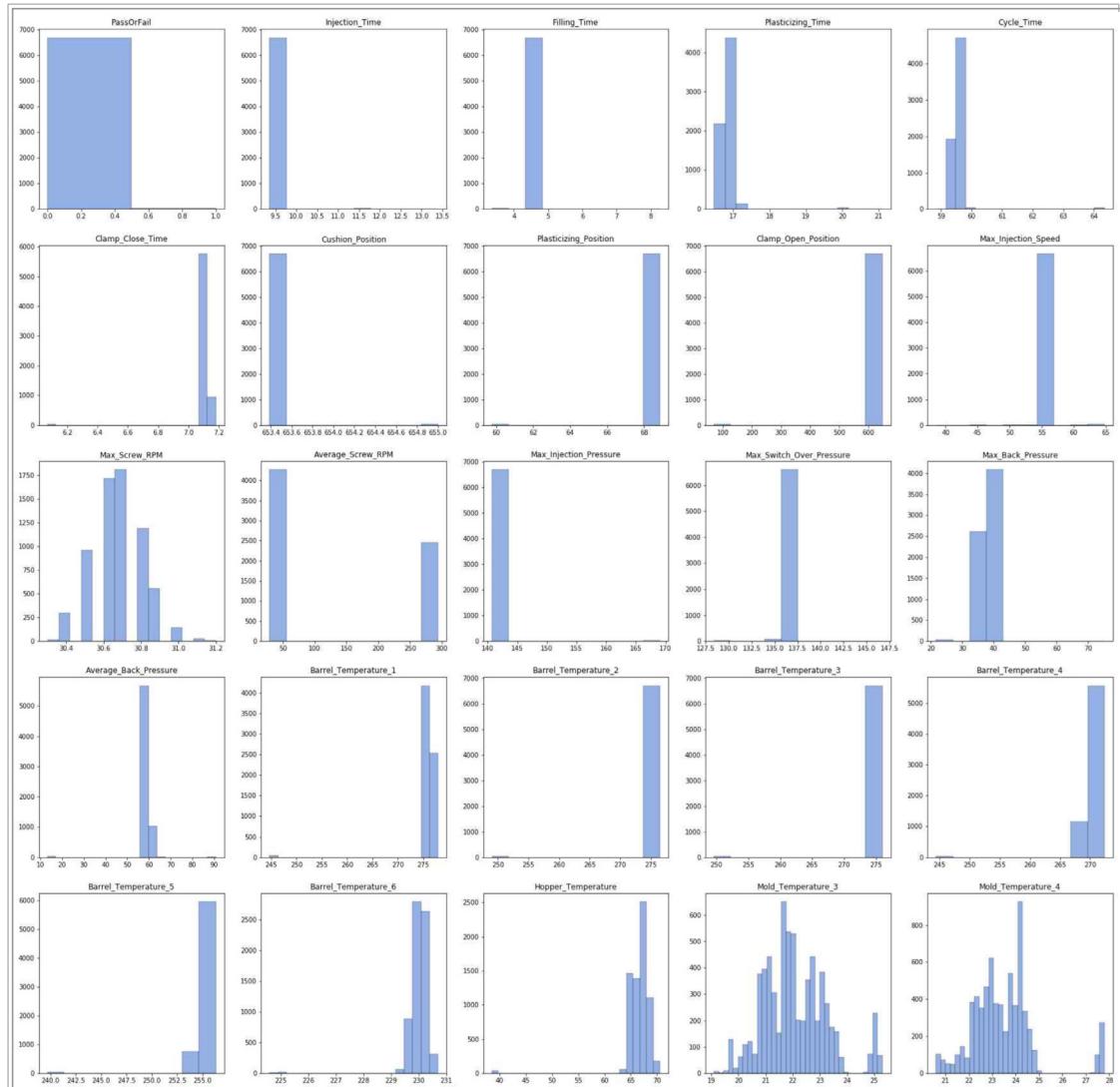
#### ④-3. CN7 제품/Histogram을 통한 변수별 데이터 파악 가이드

- 제품 'CN7'의 클래스 변수가 존재하는 데이터의 변수별 히스토그램을 확인한다. 각 변수별로 막대그래프의 개수를 설정해주고, CN7의 각 변수(총 26개)마다 해당 막대 그래프의 개수가 할당될 수 있도록 index와 value로 설정해준다. matplotlib 패키지의 pyplot(plt로 표현)을 이용하여 히스토그램을 그려준다. subplot() 함수를 이용하여 한번에 모든 변수의 히스토그램 결과를 볼 수 있도록 한다.

```
plt.figure(figsize = (30,30))

# 각 변수의 막대그래프 개수
bin = [2,10,10,15,17,20,10,10,10,10,15,10,10,10,10,10,20,20,10,10,10,10,10,20,25,35,35]

for index, value in enumerate(cn7):
    sub = plt.subplot(5, 5, index +1)
    sub.hist(cn7[value], bins = bin[index], facecolor = (144/255,171/255,221/255), linewidth=.3,
edgecolor ='black')
    plt.title(value) #결과는 아래에서 확인 가능하다.
```



[코드 10] 제품 'CN7'의 클래스 변수별 히스토그램 확인

## [단계 ⑤] 학습/평가 데이터 분리

### ⑤-1. 양품 및 불량 데이터 파악 가이드

- 데이터 품질검사, 데이터 정제(전처리)·가공 실습 내용
  - 데이터의 불량 여부 값을 가진 ‘PassOrFail’의 종류 및 개수를 불러오며, 0은 양품, 1은 불량을 의미한다.

```
cn7['PassOrFail'].value_counts() # 결과는 아래에서 확인 가능하다.
```

```
0    6697  
1     39  
Name: PassOrFail, dtype: int64
```

[코드 11] 양품 및 불량 데이터 개수 확인

### ⑤-2. 데이터 분리 및 불필요 데이터 제거 가이드

- 잡음제거 오토인코더에는 정상 데이터만 학습시키므로 데이터 분리가 필수적이다. cn\_Y는 양품 데이터만, cn7\_N는 불량 데이터만 포함하게 된다. 데이터를 분리 후 지정한 변수명은 아래와 같다.

```
# 양품  
cn7_Y = cn7[cn7['PassOrFail']==0]  
print('CN7의 양품 개수:', len(cn7_Y))  
# 불량  
cn7_N = cn7[cn7['PassOrFail']==1]  
print('CN7의 불량 개수:', len(cn7_N))
```

```
CN7의 양품 개수: 6697  
CN7의 불량 개수: 39
```

[코드 12] 양품 및 불량 데이터 분리

- 잡음제거 오토인코더 모델에 학습시킬 데이터의 형태는 클래스 변수가 없어야 하므로 변수(‘PassOrFail’)를 제거한다.

```
# 양품  
cn7_Y.drop(['PassOrFail'] ,axis=1, inplace=True)  
# 불량  
cn7_N.drop(['PassOrFail'] ,axis=1, inplace=True)
```

[코드 13] 추가적인 변수(‘PassOrFail’) 제거

### ⑤-3. MinMaxscaler를 통한 데이터 정규화 가이드

- sklearn package의 전처리(preprocessing) 함수 중 MinMaxScaler()를 사용하여 raw data를 스케일링한다. raw data는 변수마다 수치의 크기와 평균, 분산 등의 통계치가 다르고, 특히 지나치게 낮거나 높은 값인 아웃라이어(outlier)가 존재할 수

있으므로 모델의 성능에 악영향을 미칠 수 있다. 스케일링을 통해 모델의 성능을 제고하고 데이터를 정제하는 결과를 얻게 된다.

```
scaler = MinMaxScaler()
# 양품
cn7_Y = scaler.fit_transform(cn7_Y)
# 불량
cn7_N = scaler.fit_transform(cn7_N)
```

[코드 14] 정규화

#### ⑤-4. 학습 데이터/평가 데이터 분리 가이드

- 학습 데이터와 평가 데이터를 6:4의 비율로 분리한다.

\*데이터의 특성상 정확히 10배수가 되지 않을 경우, 대략 6:4 정도로 분류한다

```
# 학습 데이터(양품)
cn7_train_Y = cn7_Y[:4000]

# 평가 데이터(양품)
cn7_test_Y = cn7_Y[4000:]

# 평가 데이터(불량)
cn7_test_N = cn7_N

print('CN7의 양품 학습 데이터셋 개수:', len(cn7_train_Y))
print('CN7의 양품 평가 데이터셋 개수:', len(cn7_test_Y))
print('CN7의 불량 평가 데이터셋 개수:', len(cn7_test_N)) #결과는 아래에서 확인 가능하다.
```

```
CN7의 양품 학습 데이터셋 개수: 4000
CN7의 양품 평가 데이터셋 개수: 2697
CN7의 불량 평가 데이터셋 개수: 39
```

[코드 15] 학습 및 평가 데이터 분리

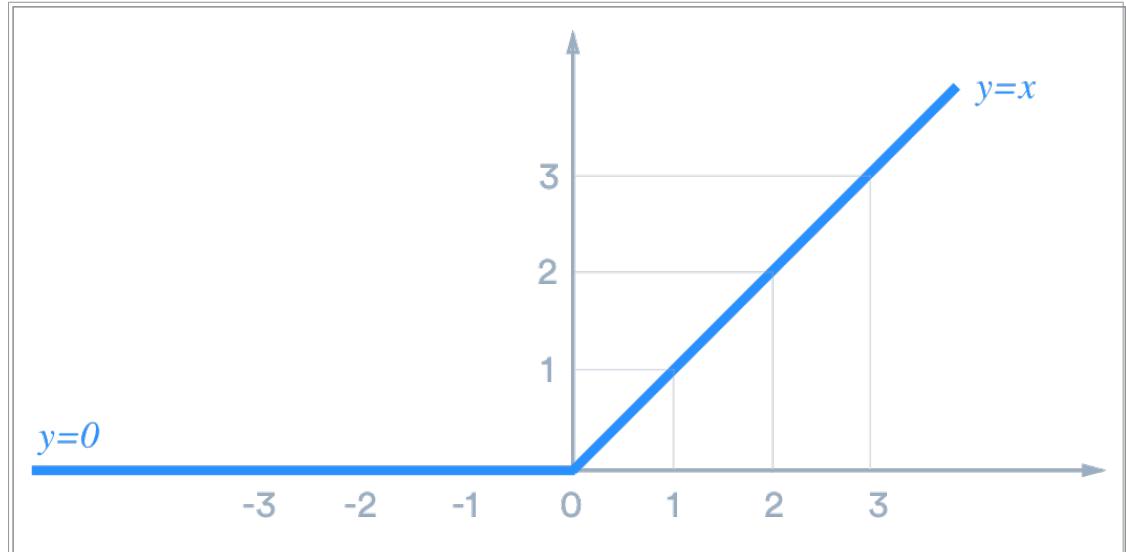
#### [단계 ⑥] 오토인코더 모델 구축 (잡음 제거)

##### ⑥-1. 오토인코더 모델 구축 가이드

- AI 분석모델 구축을 위한 방법론(알고리즘) 적용 및 학습 네트워크 구축 실습
  - (텐서플로 구현) 잡음제거 오토인코더 모델 구축

Tensorflow Keras package의 Sequential 모델을 사용하여 순서대로 연결된 층을 일렬로 쌓아서 구성한다. 인코더와 디코더, 두 개를 합친 오토인코더 모델을 구성할 때 Sequential을 사용한다. 훈련에 필요한 계산과 모델을 표현한 구조를 구성하는 과정에서 입력층, 은닉층, 출력층의 크기와 학습 속도 및 드롭아웃 비율을 직접 설정해줄 수 있으며, 이렇게 직접 설정해주는 파라미터를 하이퍼파라미터(hyper-parameter)라고 한다. 첫 번째 레이어로 Dropout 은닉층을 추가하고 드롭아웃 비율을 0.3으로 설정한다. 뉴런을 15개 혹은 5개를 가진 Dense 은닉층을 추가하고, ‘Relu’ 활성화 함수를 사용하여 층마다 각자 가중치 행렬을 관리하도록

한다. ‘Relu’는 [그림 30]과 같이 0보다 작은 입력은 0으로 만들고, 0보다 큰 입력은 그대로 내보내게 되면서 연산 비용을 줄임으로써 빠른 학습을 가능하게 한다.



[그림 30] ReLU 활성화함수

```
# 인코더
dropout_encoder = Sequential([
    Dropout(0.3),
    Dense(15, activation="relu"), # 첫 번째 은닉층
    Dense(5, activation="relu") # 두 번째 은닉층
])

# 디코더
dropout_decoder = Sequential([
    Dense(15, activation="relu", input_shape=[5]), # 세 번째 은닉층
    Dense(cn7_train_Y.shape[1], activation="relu"), # 출력
])

dropout_AE = Sequential([dropout_encoder, dropout_decoder])
```

[코드 16] 잡음제거 오토인코더 모델 구축

## [단계 ⑦] 손실 함수, 옵티마이저 정의 및 모델 컴파일, 훈련

### ⑦-1. 손실 함수 및 옵티마이저 정의 및 모델 컴파일 가이드

- 모델을 만들고 나서 compile() 메서드를 호출하여 손실 함수로는 평균 제곱 오차 (Mean Squared Error, MSE)를 사용하고, 옵티마이저로는 Adam을 사용한다. 훈련 과 평가 시에 정확도를 측정하기 위해 “accuracy”로 평가 지표로 설정한다.

### ⑦-2. 모델 훈련 가이드

- 모델을 훈련하기 위해서 fit() 메서드를 호출한다. 복원 모델이기 때문에 입력과 출력 모두 동일하게 양품의 학습 데이터(cn7\_train\_Y)로 설정한다. 데이터를 작은 뮤

은 단위로 쪼갠 미니배치(mini-batch)를 순차적으로 모델에 주입하는 것이 한 번에 모든 학습 데이터를 주입하는 것보다 빠르고 적은 비용(손실)으로 학습할 수 있다. 배치 크기를 30개로 설정함으로써 데이터를 30개씩 쪼개고, 에포크(epoch)를 30으로 두어 모델을 30번 학습시킨다. 검증 데이터는 학습 데이터의 20%를 사용한다. 학습 에포크마다 처리한 데이터 개수와 데이터마다 걸린 평균 학습 시간, 학습 데이터와 검증 데이터에 대한 손실과 정확도를 출력한다. 학습 손실이 감소하고 정확도가 올라가고 있으므로 잘 학습되고 있는 것 확인할 수 있다.

콜백(callback)은 모델의 fit() 메서드가 호출될 때 전달되는 객체이다. 훈련하는 동안 모델은 여러 지점에서 콜백을 호출하여 모델의 상태와 성능에 대한 모든 정보에 접근하고 훈련을 중지하거나 모델을 저장하고, 가중치를 적재하거나 모델 상태를 변경할 수 있다. 이 모델에서는 클래스 콜백 함수의 조기종료(early stopping)를 사용하여 검증 손실값이 더 이상 향상되지 않을 때 훈련을 중지한다. monitor는 관찰하고자 하는 항목이고, 검증 손실값이나 검증 정확도인 ‘val\_loss’나 ‘val\_acc’가 주로 사용된다. patience는 개선이 없다고 바로 종료하지 않고 개선이 없는 에포크를 얼마나 기다려 줄 것인가를 지정하는 값이다. mode는 관찰 항목에 대해 개선이 없다고 판단하기 위한 기준이며, 관찰 항목이 ‘val\_loss’인 경우에는 감소되는 것이 멈출 때 종료되어야 하므로 ‘min’으로 설정한다.

```
# 손실함수 옵티마이저 정의
dropout_AE.compile(loss="mse", optimizer=Adam(lr=0.01), metrics=['accuracy'])
# 모델 훈련
history = dropout_AE.fit(cn7_train_X, cn7_train_Y, batch_size=30, epochs=30, validation_split=0.2,
callbacks=[EarlyStopping(monitor="val_loss", patience=7, mode="min")])
# 결과는 아래에서 확인 가능하다.
```

```
Epoch 1/30
107/107 [=====] - 0s 2ms/step - loss: 0.1226 - accuracy: 0.4272 - val_loss: 0.1041 - val_accuracy: 0.1863
Epoch 2/30
107/107 [=====] - 0s 757us/step - loss: 0.0891 - accuracy: 0.7253 - val_loss: 0.0808 - val_accuracy: 0.9688
Epoch 3/30
107/107 [=====] - 0s 722us/step - loss: 0.0771 - accuracy: 0.8784 - val_loss: 0.0754 - val_accuracy: 0.9312
Epoch 4/30
107/107 [=====] - 0s 707us/step - loss: 0.0759 - accuracy: 0.8641 - val_loss: 0.0765 - val_accuracy: 0.9688
Epoch 5/30
107/107 [=====] - 0s 702us/step - loss: 0.0753 - accuracy: 0.8650 - val_loss: 0.0731 - val_accuracy: 0.9675
Epoch 6/30
107/107 [=====] - 0s 737us/step - loss: 0.0747 - accuracy: 0.8625 - val_loss: 0.0739 - val_accuracy: 0.9688
Epoch 7/30
107/107 [=====] - 0s 740us/step - loss: 0.0746 - accuracy: 0.8656 - val_loss: 0.0734 - val_accuracy: 0.9688
Epoch 8/30
107/107 [=====] - 0s 713us/step - loss: 0.0745 - accuracy: 0.8647 - val_loss: 0.0722 - val_accuracy: 0.9675
Epoch 9/30
107/107 [=====] - 0s 690us/step - loss: 0.0745 - accuracy: 0.8631 - val_loss: 0.0733 - val_accuracy: 0.9688
```

```

Epoch 10/30
107/107 [=====] - 0s 743us/step - loss: 0.0744 - accuracy: 0.8666 - val_loss: 0.0728 - val_accuracy: 0.9675
Epoch 11/30
107/107 [=====] - 0s 910us/step - loss: 0.0737 - accuracy: 0.8628 - val_loss: 0.0729 - val_accuracy: 0.9688
Epoch 12/30
107/107 [=====] - 0s 925us/step - loss: 0.0733 - accuracy: 0.8681 - val_loss: 0.0733 - val_accuracy: 0.9688
Epoch 13/30
107/107 [=====] - 0s 810us/step - loss: 0.0727 - accuracy: 0.8741 - val_loss: 0.0731 - val_accuracy: 0.9688
Epoch 14/30
107/107 [=====] - 0s 755us/step - loss: 0.0727 - accuracy: 0.8603 - val_loss: 0.0722 - val_accuracy: 0.9688
Epoch 15/30
107/107 [=====] - 0s 753us/step - loss: 0.0724 - accuracy: 0.8566 - val_loss: 0.0714 - val_accuracy: 0.9675
Epoch 16/30
107/107 [=====] - 0s 752us/step - loss: 0.0724 - accuracy: 0.8547 - val_loss: 0.0728 - val_accuracy: 0.9688
Epoch 17/30
107/107 [=====] - 0s 771us/step - loss: 0.0724 - accuracy: 0.8672 - val_loss: 0.0727 - val_accuracy: 0.9688
Epoch 18/30
107/107 [=====] - 0s 813us/step - loss: 0.0722 - accuracy: 0.8537 - val_loss: 0.0725 - val_accuracy: 0.9688
Epoch 19/30
107/107 [=====] - 0s 825us/step - loss: 0.0724 - accuracy: 0.8503 - val_loss: 0.0730 - val_accuracy: 0.9688
Epoch 20/30
107/107 [=====] - 0s 779us/step - loss: 0.0719 - accuracy: 0.8537 - val_loss: 0.0709 - val_accuracy: 0.9688
Epoch 21/30
107/107 [=====] - 0s 846us/step - loss: 0.0721 - accuracy: 0.8506 - val_loss: 0.0721 - val_accuracy: 0.9675
Epoch 22/30
107/107 [=====] - 0s 840us/step - loss: 0.0720 - accuracy: 0.8562 - val_loss: 0.0698 - val_accuracy: 0.9688
Epoch 23/30
107/107 [=====] - 0s 744us/step - loss: 0.0686 - accuracy: 0.8509 - val_loss: 0.0661 - val_accuracy: 0.9675
Epoch 24/30
107/107 [=====] - 0s 797us/step - loss: 0.0687 - accuracy: 0.8497 - val_loss: 0.0705 - val_accuracy: 0.9688
Epoch 25/30
107/107 [=====] - 0s 797us/step - loss: 0.0685 - accuracy: 0.8616 - val_loss: 0.0680 - val_accuracy: 0.9688
Epoch 26/30
107/107 [=====] - 0s 796us/step - loss: 0.0681 - accuracy: 0.8562 - val_loss: 0.0656 - val_accuracy: 0.9513
Epoch 27/30
107/107 [=====] - 0s 751us/step - loss: 0.0680 - accuracy: 0.8575 - val_loss: 0.0700 - val_accuracy: 0.9688
Epoch 28/30
107/107 [=====] - 0s 814us/step - loss: 0.0679 - accuracy: 0.8556 - val_loss: 0.0680 - val_accuracy: 0.9675
Epoch 29/30
107/107 [=====] - 0s 855us/step - loss: 0.0680 - accuracy: 0.8584 - val_loss: 0.0670 - val_accuracy: 0.9675
Epoch 30/30
107/107 [=====] - 0s 811us/step - loss: 0.0680 - accuracy: 0.8534 - val_loss: 0.0687 - val_accuracy: 0.9688

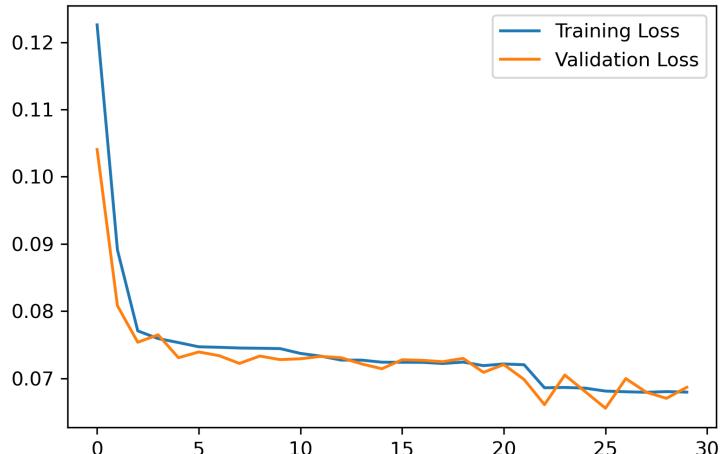
```

[코드 17] 모델 컴파일 및 훈련

- 머신러닝 및 딥러닝 모델은 매번 같은 학습 결과를 도출하지는 않기 때문에, 위 학습 결과와 다른 결과가 나타날 수 있다.

학습이 진행되면서 매 에포크마다 손실값(loss)과 정확도(위 그림에서 ‘acc’)가 변화하는 것을 확인할 수 있다. 일반적으로 손실값은 점점 줄어들고 정확도는 증가하는 것을 보고 학습이 잘 진행되고 있다고 판단한다. 이때 주의해야 할 점은 학습 손실값(training loss)은 감소하지만 검증 손실값(validation loss)가 증가한다면 학습 데이터에 모델이 과적합되고 있다는 의미이므로 에포크 수를 줄여야 한다. matplotlib 패키지를 활용하여 에포크별 학습 진행 상황을 확인할 수 있다. ‘plot’ 함수로 학습 손실값과 검증 손실값을 그래프로 그리고 ‘legend’ 함수로 범례를 만들어 준 후, ‘show’ 함수로 해당 그래프를 시각화한다.

```
plt.plot(history.history["loss"], label="Training Loss")
plt.plot(history.history["val_loss"], label="Validation Loss")
plt.legend()
plt.show() #결과는 아래에서 확인 가능하다.
```

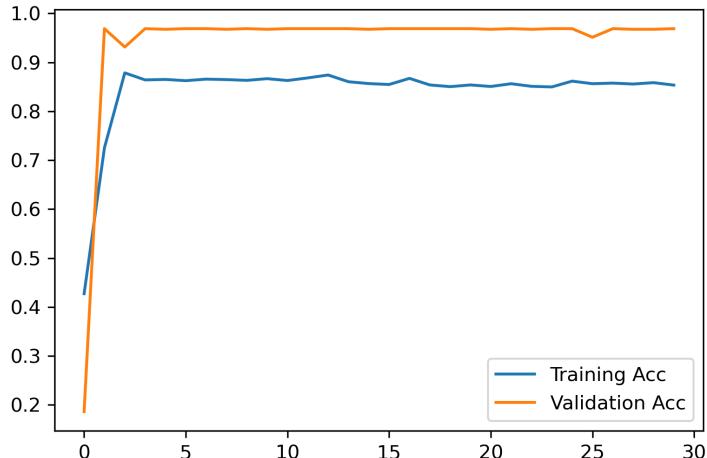


[코드 18] 에포크별 손실값 변화

```

plt.plot(history.history["accuracy"], label="Training Acc")
plt.plot(history.history["val_accuracy"], label="Validation Acc")
plt.legend()
plt.show() #결과는 아래에서 확인 가능하다.

```



[코드 19] 에포크별 정확도 변화

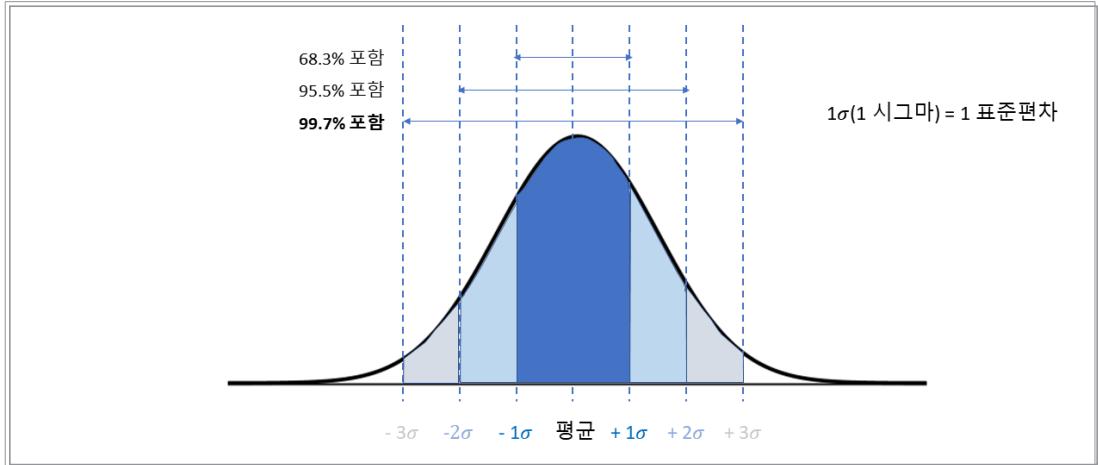
## [단계 ⑧] 임계값 정의 및 예측값과 복원 오차 확인

### ⑧-1. 임계값 정의 가이드

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

- 양품 및 불량 구분을 위한 임계치 설정

양품과 불량을 구분하기 위해서는, 잡음제거 오토인코더 모델을 각각 거치고 나온 양품 데이터와 불량 데이터의 복원 오차를 구별할 수 있는 임계치가 필요하다. 양품 데이터로만 학습된 모델은 양품 데이터가 들어왔을 때 더 잘 복원해낼 것이므로 작은 복원 오차값을 가지지만, 불량 데이터가 들어왔을 때에는 덜 복원해내므로 높은 복원 오차값을 가지게 된다. 해당 알고리즘에서는 임계치로 3시그마 규칙(three-sigma rule)을 적용한다. 3시그마 규칙은 평균에서 양쪽으로 3 표준편차 범위에 거의 모든 값들(99.7%)가 들어간다는 가정인데, 잡음제거 오토인코더 모델의 정상 데이터 예측값의 3시그마 이상의 값을 적용하여 임계치를 설정한다. 아래의 예시에는 5시그마 값을 사용하였다.



[그림 31] 3시그마 규칙 – 1시그마, 2시그마, 3시그마에 대한 데이터 범위 도식화

```
# 학습 데이터의 예측값

cn7_train_pred = dropout_AE.predict(cn7_train_Y)
# 학습 데이터의 복원 오차 (예측값 - 실제 값)
cn7_train_loss = np.mean(np.square(cn7_train_pred - cn7_train_Y), axis=1)

# 임계치
threshold = np.mean(cn7_train_loss) + 5*np.std(cn7_train_loss)

print("복원 오류 임계치: ", threshold) # 결과는 아래에서 확인 가능하다.
```

복원 오류 임계치: 0.08615016806460124

[코드 20] 양품 및 불량 구분을 위한 임계치 설정

### ⑧-2. 데이터 각각의 예측값과 복원 오차 확인 가이드

- 평가 데이터의 양품 및 불량 데이터 각각의 예측값과 복원 오차를 확인한다. 복원 오차가 임계치(threshold)보다 크면 모델이 익숙하지 않은 패턴을 보고 있음을 유추하고 해당 데이터를 불량이라고 판단할 수 있다.

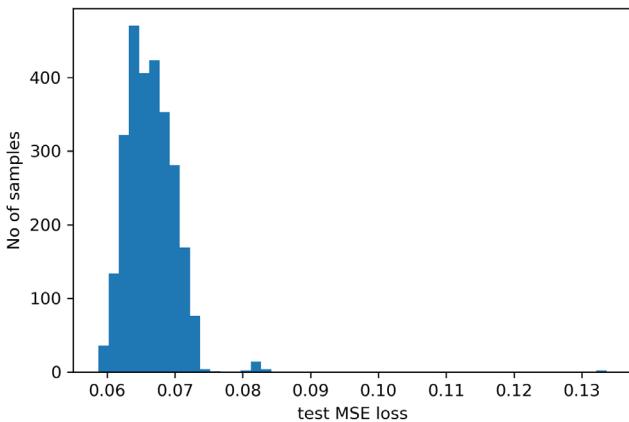
```
# 평가 데이터의 양품

# 예측값
cn7_predict_Y = dropout_AE.predict(cn7_test_Y)

# 양품 평가 데이터의 복원 오차 (예측값 - 실제 값)
cn7_test_Y_mse = np.mean(np.square(cn7_predict_Y - cn7_test_Y), axis=1)

# 시각화
plt.hist(cn7_test_Y_mse, bins=50)
plt.xlabel("test MSE loss")
plt.ylabel("No of samples")
plt.show()

# 불량으로 판단한 데이터 확인
cv7_test_Y_anomalies = cn7_test_Y_mse > threshold
print("불량 개수: ", np.sum(cv7_test_Y_anomalies)) # 결과는 아래에서 확인 가능하다.
```

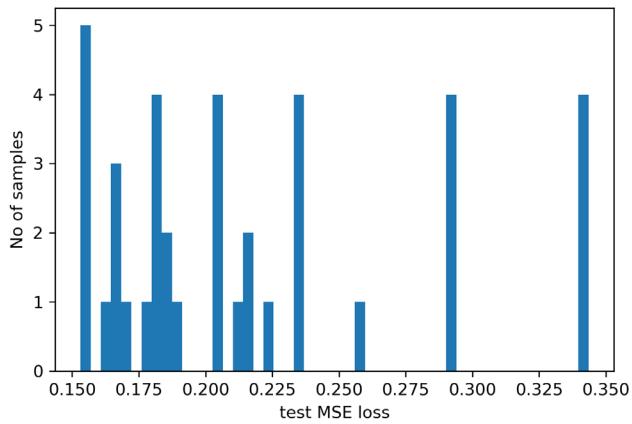


[코드 21] 평가 데이터 중 양품의 예측값 및 복원 오차 확인

```
# 평가 데이터의 불량
# 예측값
cn7_predict_N = dropout_AE.predict(cn7_test_N)
# 불량 평가 데이터의 복원 오차 (예측값 - 실제 값)
cn7_test_N_mse = np.mean(np.square(cn7_predict_N - cn7_test_N), axis=1)

# 시각화
plt.hist(cn7_test_N_mse, bins=50)
plt.xlabel("test MSE loss")
plt.ylabel("No of samples")
plt.show()

# 불량으로 판단한 데이터 확인
cv7_test_N_anomalies = cn7_test_N_mse > threshold
print("불량 개수: ", np.sum(cv7_test_N_anomalies)) # 결과는 아래에서 확인 가능하다.
```



[코드 22] 평가 데이터 중 불량의 예측값 및 복원 오차 확인

## [단계 ⑨] 결과 분석 및 해석

### ⑨-1. 분석 결과 도출 가이드

#### - 분석 결과값 도출 (시각화 필요)

- 혼동 행렬(confusion matrix)로 결과를 확인하기 위하여 평가 데이터의 양품과 불량을 결합하여 실제 값과 예측값 형태로 정리한다.

```
cn7_true = np.concatenate([np.zeros(len(cv7_test_Y_anomalies)), np.ones(len(cv7_test_N_anomalies))])
```

[코드 23] 평가 데이터의 실제 값

```
cn7_prediction = np.concatenate([cv7_test_Y_anomalies, cv7_test_N_anomalies])
```

[코드 24] 평가 데이터의 예측값

- sklearn package에서 제공하는 confusion\_matrix 활용하여 혼동 행렬을 구한다. 혼동 행렬은 [코드 25]와 같이 실제값과 예측값을 비교하는 데에 사용되며 정확도, 정밀도, 재현율은 혼동 행렬에 속하는 값으로 계산할 수 있다. 혼동 행렬 내부에는 TP, TN, FP, FN라는 네 가지 값이 위치한다. TN는 정상(양품)의 클래스 변수를 가진 데이터를 올바르게 정상으로 예측한 개수, TP는 불량의 클래스 변수를 가진 데이터를 올바르게 불량으로 예측한 개수이다. FP는 불량의 클래스 변수를 가진 데이터를 정상으로 잘못 예측한 개수, FN은 정상의 클래스 변수를 가진 데이터를 불량으로 잘못 예측한 개수이다.

		실제 클래스 (Actual class)	
		정상	불량
예측 클래스 (Predicted class)	정상	TP (True Positive)	FP (False Positive)
	불량	FN (False Negative)	TN (True Negative)

[표 3] 혼동 행렬

```
from sklearn.metrics import confusion_matrix
confusion_matrix(cn7_true, cn7_prediction) # 결과는 아래에서 확인 가능하다.
```

```
array([[2695,     2],
       [    0,  39]])
```

[코드 25] 혼동 행렬로 확인

- sklearn package에서 제공하는 accuracy\_score, precision\_score, recall\_score, f1\_score 활용하여 정확도, 정밀도, 재현율 및 F1 score를 확인한다. 이 값을 사용하여 정확도는  $\frac{TP+TN}{TP+TN+FP+FN}$ , 정밀도는  $\frac{TP}{TP+FP}$ , 재현율은  $\frac{TP}{TP+FN}$ 로 계산된다. 해석하자면, 정확도는 모델의 예측이 맞은(양품과 불량품 모두) 비율을 나

타내며, 정밀도는 모델이 불량품(1)이라고 예측한 것 중에서 옳게 예측한 비율을 의미하고, 재현율은 실제 불량품(1) 중에서 모델이 불량품으로 예측한 비율을 의미한다. 데이터의 클래스 불균형(class imbalance)이 심한 상황이고, 실제 공정에서 불량품을 어느 정도 잘 탐지하느냐가 중요하므로 재현율(recall)의 중요도가 비교적 높다고 할 수 있다.

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
print("정확도:", accuracy_score(cn7_true, cn7_prediction))
print("정밀도:", precision_score(cn7_true, cn7_prediction))
print("재현율:", recall_score(cn7_true, cn7_prediction))
print("F1:", f1_score(cn7_true, cn7_prediction)) # 결과는 아래에서 확인 가능하다.
```

```
정확도: 0.9992690058479532
정밀도: 0.9512195121951219
재현율: 1.0
F1: 0.975
```

[코드 26] : 평가 데이터의 정확도, 정밀도, 재현율 및 F1 score

## ⑨-2. 분석결과에 대한 해석 가이드

### - 분석결과에 대한 논의 및 해석(implication)

- 정확도는 높은 반면 그 외의 평가 지표 결과가 낮은 이유는 데이터 수의 부족이 될 수 있다. 정확도의 경우 전체 데이터 중에서 옳게 분류한 개수를 의미하는 반면, 그 외의 평가 지표는 정상 데이터 중에서, 혹은 불량 데이터 중에서 옳게 분류한 데이터의 수를 계산한다. 그로 인해 해당 지표들은 특히 불량의 총 개수에 민감하게 바뀔 수 있는데, 성능 결과가 낮다면 전반적인 성능을 높이기 위해서는 불량 데이터의 수를 확보함으로써 보다 많은 정보를 모델에 학습시키고 덜 민감한 성능 지표를 갖도록 해야 한다. 정밀도와 재현율은 서로 트레이드 오프(trade-off) 관계를 가지고 있는데, 그 이유는 정상과 불량을 구분(분류)하는 임계값이 줄어들면 정상을 불량이라고 판단한 개수인 FP의 수는 증가하지만 불량을 정상 데이터로 판단한 개수인 FN이 감소하게 되어 정밀도는 감소하는 반면 재현율은 증가한다. 낮은 정밀도를 높이기 위해서 4시그마 혹은 5시그마로 임계값을 높이게 되면 정밀도는 개선될 수 있을지라도 재현율이 낮아지므로, 적절한 정밀도와 재현율을 가질 수 있게 하는 임계값을 설정하는 것 또한 중요하다.

## 2.3.2 준지도 학습 활용 알고리즘 – 준지도 학습 적용을 위한 모델 구현 과정

### 1) 분석 단계별 프로세스



[그림 32] 준지도 학습 적용을 위한 모델 구현 과정

- i) 클래스 변수가 있는 데이터는 labeled data, 클래스 변수가 없는 데이터는 unlabeled data
- ii) 층화 추출법을 이용하여 클래스 별 양품/불량 데이터의 비율 유지
- iii) 일부 unlabeled의 클래스 변수에 예측한 값을 부여하여 labeled data에 통합하고, 이를 unlabeled data 가 없을 때까지 반복
- iv) 학습데이터를 통한 신경망 학습, 평가데이터를 통한 신경망 훈련도 검사, 임계치 설정, 예측값과의 오차 확인, 시각화

#### [단계 ①] 라이브러리/데이터 불러오기

##### ①-1. 필요 데이터 다운로드 및 불러오기(import) 가이드

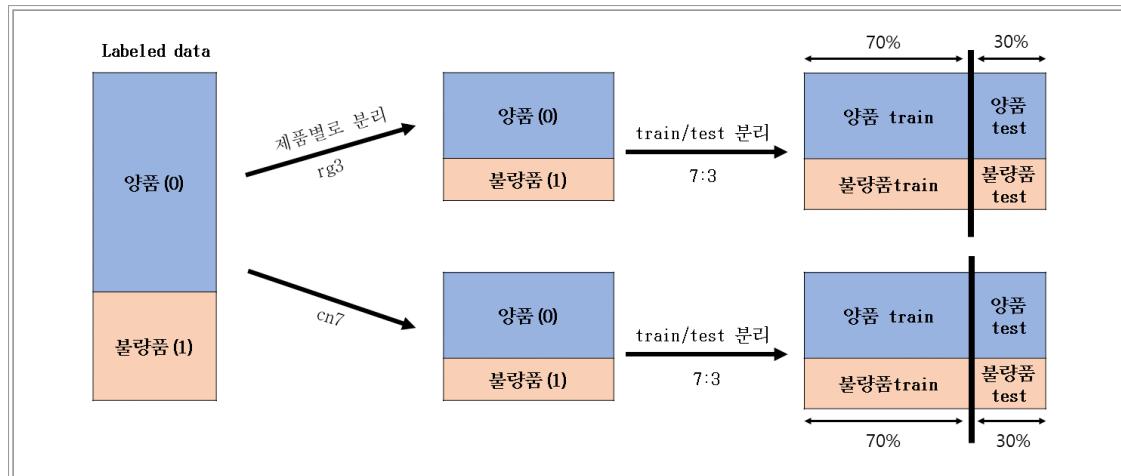
- 모델 학습을 위한 데이터 준비과정은 크게 두 가지이다. 클래스 변수가 존재하는 데이터를 [그림 33]과 같이 제품 별로 분류한 뒤, 학습에 용이하도록 학습 데이터와 평가 데이터로 구분한다. 이 과정에서 클래스 별 데이터 개수를 유지하기 위해 층화추출법을 사용한다. 두 번째로 클래스 변수가 없는 데이터도 동일하게 제품 별로 분류한 뒤 사용한다. 클래스 변수가 없는 데이터는 학습에만 사용되므로 평가 데이터로 귀속시킬 필요가 없다.

```

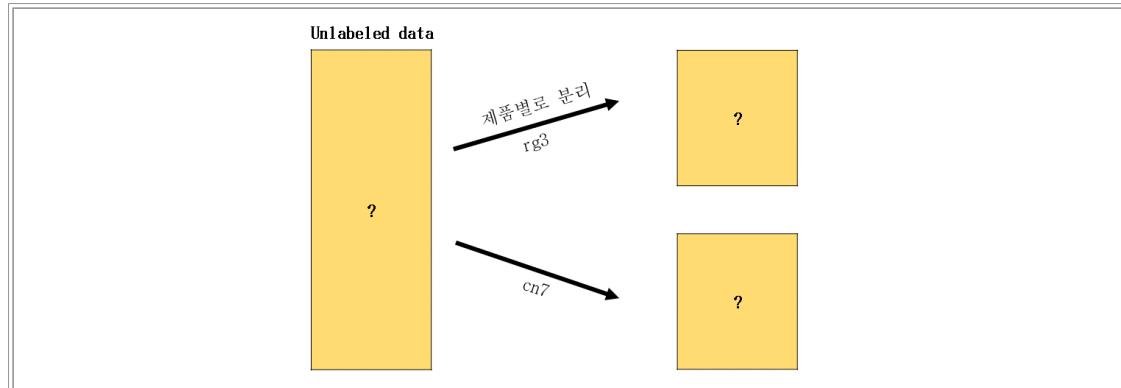
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import sklearn
from sklearn.model_selection import train_test_split, StratifiedShuffleSplit
from sklearn.metrics import classification_report, precision_score, recall_score, roc_auc_score,
accuracy_score, f1_score
import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras import optimizers
import tensorflow as tf
from tensorflow.keras import backend as k
import seaborn as sns
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
from tensorflow.keras.layers import Dense, Dropout

```

[코드 27] 준지도 학습을 위한 패키지 불러오기



[그림 33] : 클래스 변수가 존재하는 데이터셋 준비



[그림 34] : 클래스 변수가 존재하지 않는 데이터셋 준비

- python으로 사용 가능한 pandas package 활용
- pandas의 read\_csv 함수를 이용하여 데이터가 저장된 컴퓨터 내의 경로를 찾아 입력하고 데이터를 불러온다.

```

#pandas의 read_csv 함수를 사용하여 데이터를 불러옴
moldset_labeled = pd.read_csv(r'./moldset_labeled.csv', low_memory=False, index_col=False)

```

[코드 28] 데이터 불러오기

## [단계 ②] 데이터 종류 및 개수 확인

### ②-1. 데이터 종류 및 개수 확인 가이드

- 데이터 내에 존재하는 분석하고자 하는 사출기의 이름을 확인한다.

```
#pandas dataframe의 value_counts 함수를 사용  
moldset_labeled['EQUIP_NAME'].value_counts() #결과는 아래에서 확인 가능하다.
```

```
650톤-우진2호기    2607  
Name: EQUIP_NAME, dtype: int64
```

[코드 29] 데이터 이름 및 타입 확인하기

- 데이터 내에 분석하고자 하는 제품 코드와 해당하는 데이터 개수를 확인한다.

```
#분석에 사용되는 제품 코드  
parts = ['86141AA000', '86131AA000', '86141T1000', '86131T1000']  
moldset_labeled['PART_NO'].value_counts() #결과는 아래에서 확인 가능하다.
```

```
86141AA000    713  
86131AA000    712  
86131T1000    591  
86141T1000    591  
Name: PART_NO, dtype: int64
```

[코드 30] 데이터 개수 확인하기

- 불러온 데이터를 확인한다.

```
#dataframe 이름을 사용  
moldset_labeled.head() #결과는 아래에서 확인 가능하다.
```

_id	TimeStamp	PART_FACT_PLAN_DATE	PART_FACT_SERIAL	PART_NO	PART_NAME	EQUIP_CD	EQUIP_NAME	PassOrFail
0	5f8928bb9c0189cc666ef19b	2020-10-16 04:57:47	2020-10-16 오전 12:00:00	24	86141AA000	CN7 W/S SIDE MLD'G RH	S14	650톤-우진2호기 0
1	5f8928de9c0189cc666ef20b	2020-10-16 04:58:48	2020-10-16 오전 12:00:00	24	86141AA000	CN7 W/S SIDE MLD'G RH	S14	650톤-우진2호기 0
2	5f8928df9c0189cc666ef213	2020-10-16 04:58:48	2020-10-16 오전 12:00:00	23	86131AA000	CN7 W/S SIDE MLD'G LH	S14	650톤-우진2호기 0
3	5f8928f39c0189cc666ef25e	2020-10-16 04:59:48	2020-10-16 오전 12:00:00	23	86131AA000	CN7 W/S SIDE MLD'G LH	S14	650톤-우진2호기 0
4	5f8928f59c0189cc666ef265	2020-10-16 04:59:48	2020-10-16 오전 12:00:00	24	86141AA000	CN7 W/S SIDE MLD'G RH	S14	650톤-우진2호기 0

5 rows × 46 columns

[코드 31] 데이터 불러오기 및 확인하기

## [단계 ③] 데이터 특성 파악

### ③-1. 불필요 데이터 제거 가이드

- 통계 분석 알고리즘 및 시각화 도구 등을 활용한 변수들의 분포 파악, 변수 간의 관계 파악을 위한 실습 내용
  - 클래스 변수가 존재하는 데이터와 클래스 변수가 존재하지 않는 데이터를 모두 사

용하지만, 2.3.1의 지도 학습 활용 알고리즘에서 클래스 변수가 존재하는 데이터를 시각화 하였으므로 아래는 클래스 변수가 존재하지 않는 데이터를 시각화 한다.  
(시각화 자료 다음 장)

- 데이터 내의 특성(변수)을 확인한 후에 값이 존재하지 않는 등의 불필요한 특성을 데이터에서 삭제한다.

```
#불필요한 특성을 dataframe에서 삭제함.
moldset_labeled.drop(columns={'_id', 'TimeStamp', 'PART_FACT_PLAN_DATE', 'PART_FACT_SERIAL',
'PART_NAME', 'EQUIP_CD', 'EQUIP_NAME', 'Reason',
'Mold_Temperature_1', "Mold_Temperature_2", "Barrel_Temperature_7",
"Switch_Over_Position",
"Mold_Temperature_5", "Mold_Temperature_6", "Mold_Temperature_7",
"Mold_Temperature_8", "Mold_Temperature_9", "Mold_Temperature_10",
"Mold_Temperature_11", "Mold_Temperature_12"}, inplace=True)
```

[코드 32] 데이터 특성 중 불필요한 성질을 데이터셋에서 제거하기

- 레이블이 없는 데이터를 동일한 방법으로 불러온다. 위에서와 같이, 불필요한 특성을 데이터에서 삭제하는 과정을 거친다.

```
moldset_unlabeled = pd.read_csv(r'unlabeled_data.csv', index_col=0)

#불필요한 특성을 dataframe에서 삭제함.
moldset_unlabeled.drop(columns={'_id', 'TimeStamp', 'PART_FACT_PLAN_DATE', 'PART_FACT_SERIAL',
'PART_NAME', 'EQUIP_CD', 'EQUIP_NAME', 'ERR_FACT_QTY',
'Mold_Temperature_1', "Mold_Temperature_2", "Barrel_Temperature_7",
"Switch_Over_Position",
"Mold_Temperature_5", "Mold_Temperature_6", "Mold_Temperature_7",
"Mold_Temperature_8", "Mold_Temperature_9", "Mold_Temperature_10",
"Mold_Temperature_11", "Mold_Temperature_12"}, inplace=True)
```

[코드 33] 데이터 특성 중 불필요한 성질을 데이터셋에서 제거하기

- 데이터를 원하는 제품 코드를 가지고 분리하여 따로 선언한다.

```
#사출성형 시 분석하는 제품을 제품 코드로 선택 후 데이터셋을 분리함
moldset_labeled_cn7      = moldset_labeled[(moldset_labeled['PART_NO']=='86131AA000') |
(moldset_labeled['PART_NO']=='86141AA000')]
moldset_labeled_cn7.drop(columns={'PART_NO'}, inplace=True)

moldset_labeled_rg3      = moldset_labeled[(moldset_labeled['PART_NO']=='86131T1000') |
(moldset_labeled['PART_NO']=='86141T1000')]
moldset_labeled_rg3.drop(columns={'PART_NO'}, inplace=True)

moldset_unlabeled_cn7    = moldset_unlabeled[(moldset_unlabeled['PART_NO']=='86131AA000') |
(moldset_unlabeled['PART_NO']=='86141AA000')]
moldset_unlabeled_cn7.drop(columns={'PART_NO'}, inplace=True)

moldset_unlabeled_rg3    = moldset_unlabeled[(moldset_unlabeled['PART_NO']=='86131T1000') |
(moldset_unlabeled['PART_NO']=='86141T1000')]
moldset_unlabeled_rg3.drop(columns={'PART_NO'}, inplace=True)
```

[코드 34] 원하는 제품 코드로 데이터 분류하기

### ③-2. CN7 제품/describe 함수를 통한 통계량 파악 가이드

- 제품 'CN7'의 클래스 변수가 존재하지 않는 데이터의 변수별 요약 통계량을 확인 한다.

```
moldset_unlabeled_cn7.describe() # 결과는 아래에서 확인 가능하다.
```

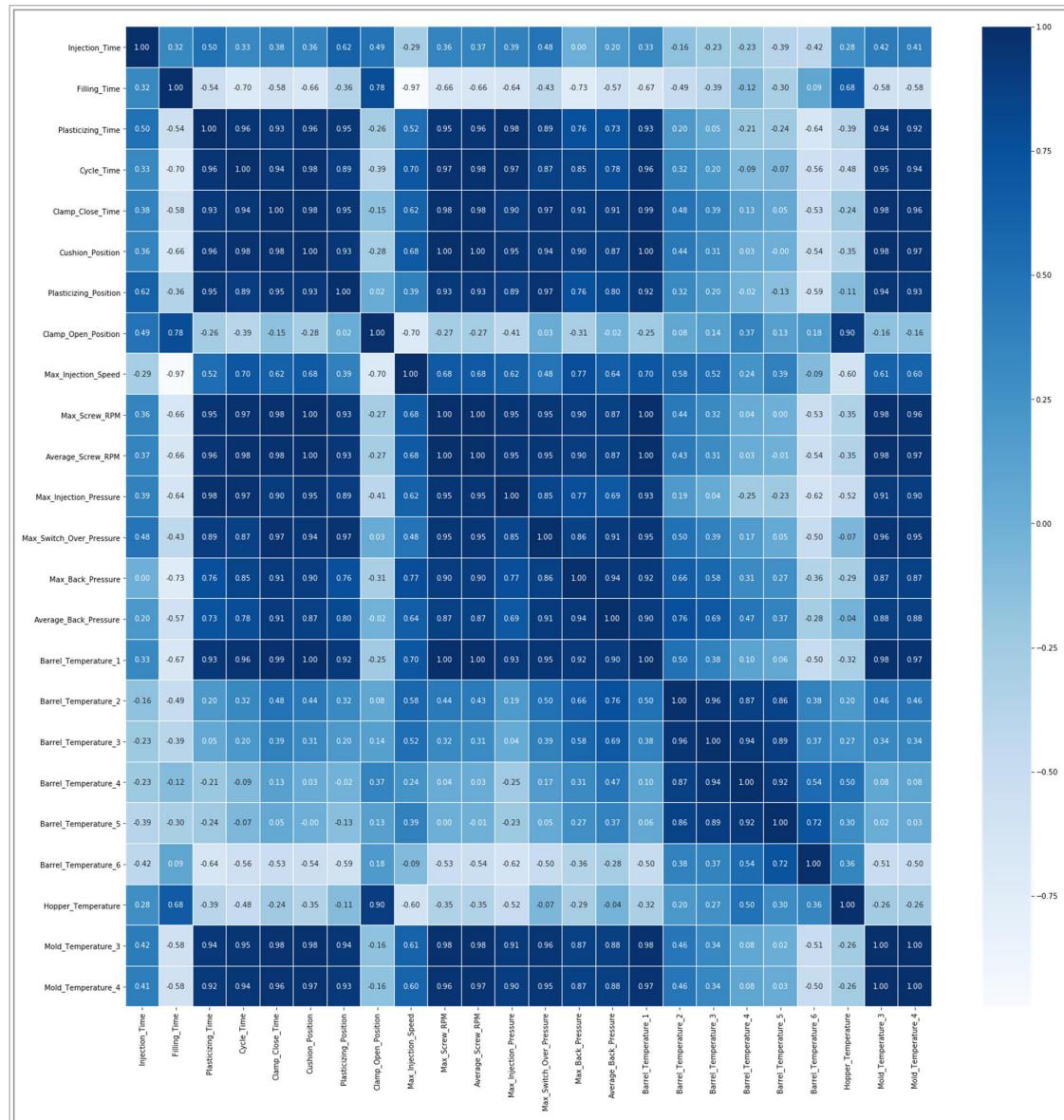
	Injection_Time	Filling_Time	Plasticizing_Time	Cycle_Time	Clamp_Close_Time	...
count	52547.000000	52547.000000	52547.000000	52547.000000	52547.000000	...
mean	6.424337	4.218894	10.436911	50.905432	5.141773	...
std	3.020516	1.444531	7.396193	13.369645	1.868678	...
min	1.000000	0.300000	0.500000	0.000000	2.800000	...
25%	5.000000	3.890000	2.800000	36.600000	3.200000	...
50%	5.000000	4.490000	12.810000	59.520000	5.990000	...
75%	9.560000	5.000000	16.860000	63.600000	7.120000	...
max	11.400000	6.000000	97.080000	119.880000	7.270000	...
	Cushion_Position	Plasticizing_Position	Clamp_Open_Position	Max_Injection_Speed	Max_Screw_RPM	...
	52547.000000	52547.000000	52547.000000	52547.000000	52547.000000	...
	359.175364	52.581079	434.434858	56.389434	16.472642	...
	317.320806	13.087495	235.942618	29.437641	15.334763	...
	15.800000	32.180000	0.000000	25.100000	0.000000	...
	17.300000	40.300000	485.900000	36.500000	0.000000	...
	653.410000	52.660000	522.200000	49.900000	30.300000	...
	653.560000	68.470000	647.990000	56.100000	30.700000	...
	655.330000	75.750000	652.000000	128.200000	62.900000	...
	Average_Back_Pressure	Barrel_Temperature_1	Barrel_Temperature_2	Barrel_Temperature_3	Barrel_Temperature_4	...
	52547.000000	52547.000000	52547.000000	52547.000000	52547.000000	...
	27.234318	145.529239	267.087272	268.502999	265.401555	...
	29.014509	135.601133	11.345312	10.124143	8.909881	...
	0.000000	0.000000	244.200000	231.600000	239.100000	...
	0.000000	0.000000	264.600000	264.900000	264.800000	...
	12.200000	245.100000	264.900000	269.900000	269.400000	...
	59.300000	276.100000	275.400000	275.000000	270.500000	...
	112.900000	287.500000	286.200000	285.800000	279.000000	...
	Barrel_Temperature_5	Barrel_Temperature_6	Hopper_Temperature	Mold_Temperature_3	Mold_Temperature_4	...
	52547.000000	52547.000000	52547.000000	52547.000000	52547.000000	...
	254.059596	233.254549	57.740971	12.841428	14.730523	...
	7.613118	6.257067	10.945172	12.181921	14.162900	...
	234.400000	219.200000	34.300000	0.000000	0.000000	...
	254.600000	229.800000	57.800000	0.000000	0.000000	...
	255.000000	234.700000	61.600000	20.500000	22.000000	...
	259.600000	239.900000	65.200000	23.400000	26.700000	...
	267.200000	242.800000	73.400000	38.600000	48.600000	...

[코드 35] 제품 'CN7'의 클래스 변수별 상관관계 확인

### ③-3. CN7 제품/corr 함수를 통한 변수 간 상관관계 파악 가이드

- 제품 'CN7'의 클래스 변수가 존재하지 않는 데이터의 변수 간 상관관계를 확인한다.

```
plt.subplots(figsize=(25,25))
sns.heatmap(data      = moldset_unlabeled_cn7.corr(),    linewidths=0.1,annot=True,   fmt     ='.2f',
cmap='Blues')
#결과는 아래에서 확인 가능하다.
```



[코드 36] 제품 'CN7'의 클래스 변수별 상관관계 확인

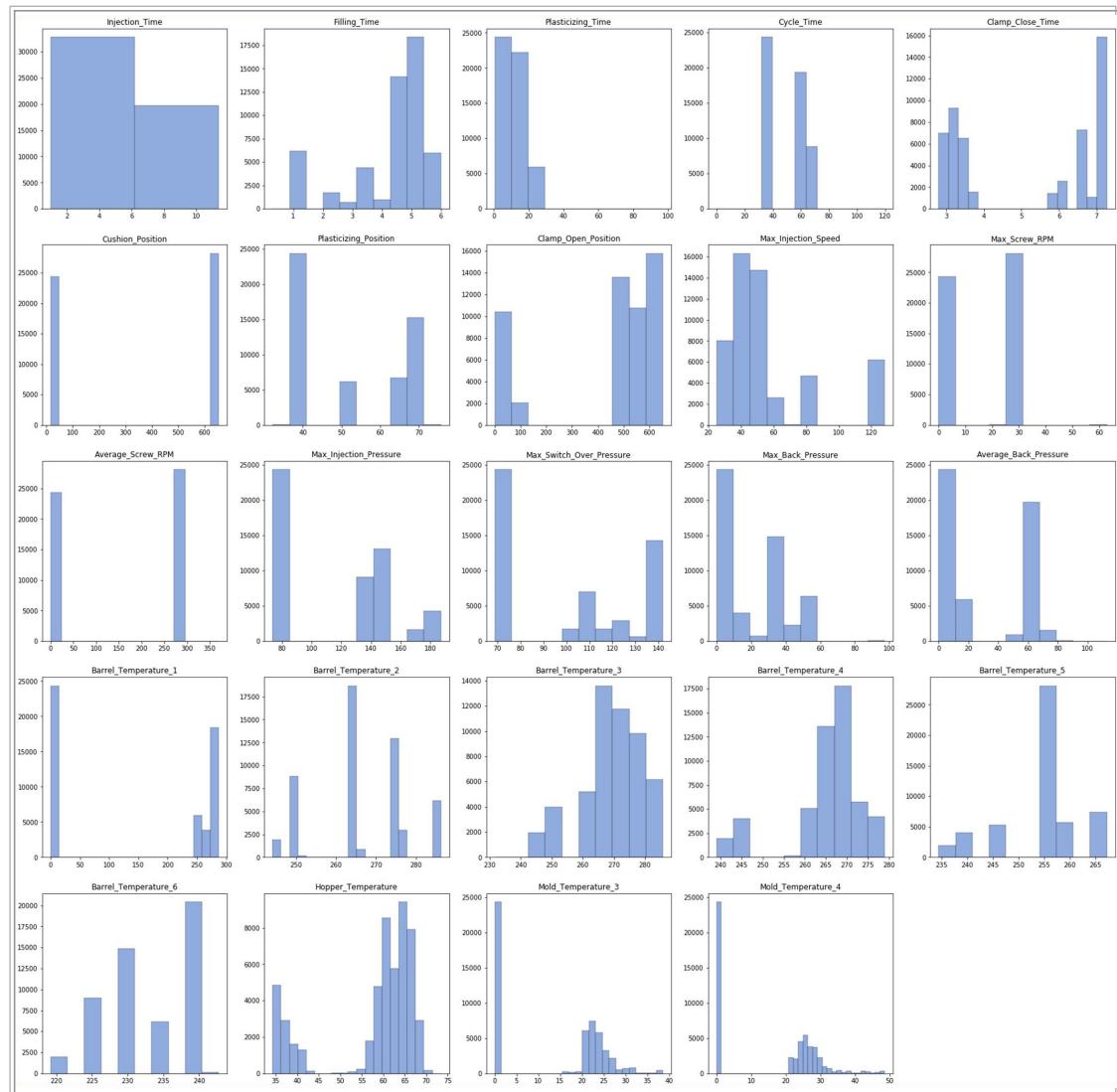
### ③-4. CN7 제품/Histogram을 통한 변수별 데이터 파악 가이드

- 제품 'CN7'의 클래스 변수가 존재하지 않는 데이터의 변수별 히스토그램을 확인한다.

```
plt.figure(figsize = (30,30))

# 각 변수의 막대그래프 개수
bin = [2,10,10,15,17,20,10,10,10,10,15,10,10,10,10,20,20,10,10,10,10,20,25,35,35]

for index, value in enumerate(moldset_unlabeled_cn7):
    sub = plt.subplot(5,5,index + 1)
    sub.hist(moldset_unlabeled_cn7[value], bins = bin[index], facecolor = (144/255,171/255,221/255),
    linewidth=.3, edgecolor ='black')
    plt.title(value) #결과는 아래에서 확인 가능하다.
```



[코드 37] 제품 'CN7'의 클래스 변수별 히스토그램 확인

### ③-5. RG3 제품/describe 함수를 통한 통계량 파악 가이드

- 제품 'RG3'의 클래스 변수가 존재하지 않는 데이터의 변수별 요약 통계량을 확인 한다.

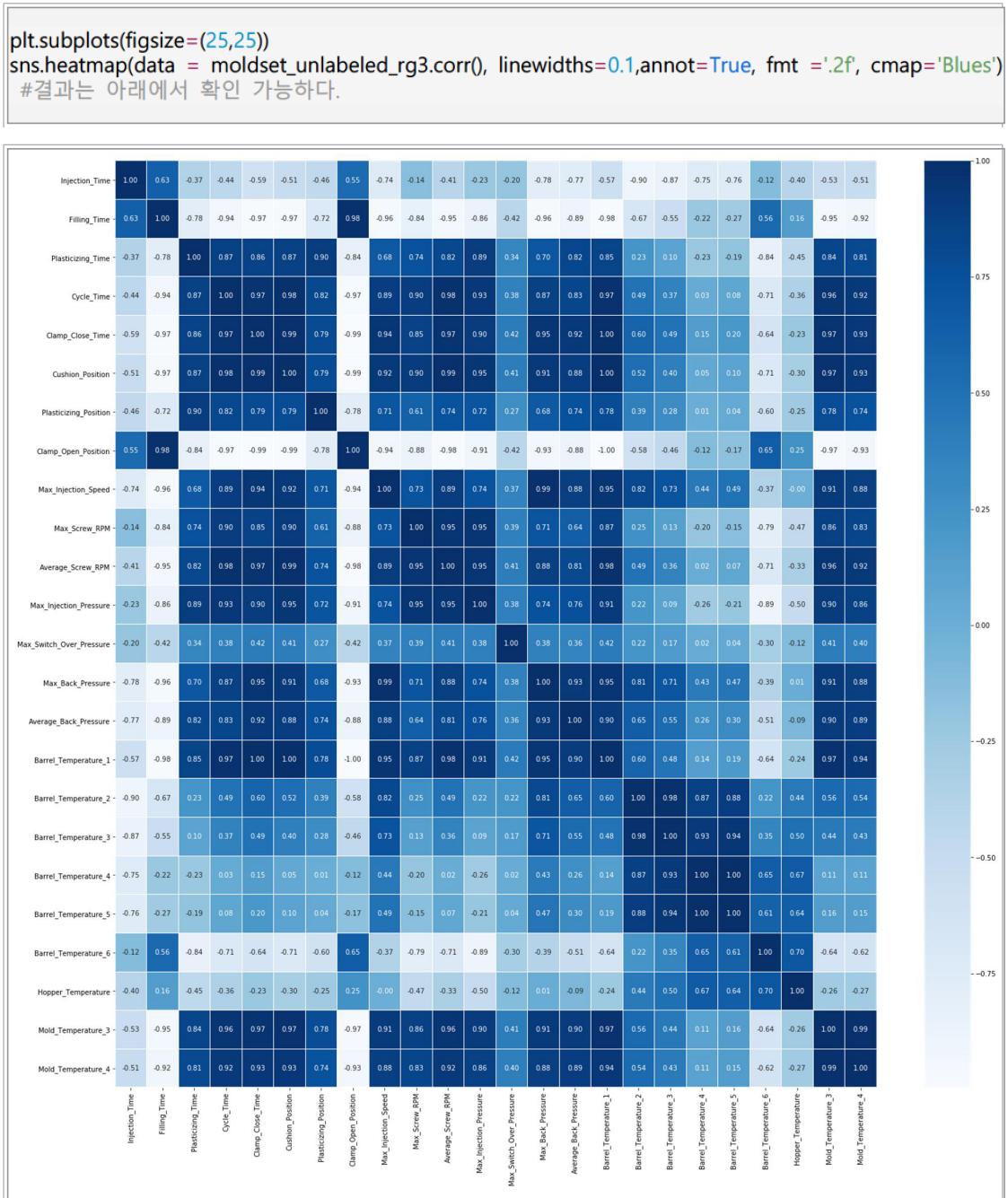
`moldset_unlabeled_rg3.describe()` #결과는 아래에서 확인 가능하다.

	Injection_Time	Filling_Time	Plasticizing_Time	Cycle_Time	Clamp_Close_Time	...
count	37477.000000	37477.000000	37477.000000	37477.000000	37477.000000	...
mean	3.515335	2.778947	9.631108	53.469702	5.357147	...
std	2.837536	2.087176	6.238300	12.758191	1.633415	...
min	0.130000	0.000000	0.400000	0.000000	2.800000	...
25%	1.050000	0.930000	2.900000	38.900000	3.400000	...
50%	2.630000	1.650000	12.800000	61.760000	6.640000	...
75%	5.000000	5.000000	13.030000	63.600000	6.740000	...
max	11.390000	6.000000	38.710000	76.190000	15.100000	...
Cushion_Position Plasticizing_Position Clamp_Open_Position Max_Injection_Speed Max_Screw_RPM ...						
37477.000000	37477.000000	37477.000000	37477.000000	37477.000000	37477.000000	...
405.104732	48.223423	209.289599	86.657328	20.604886	...	
310.466273	7.846830	243.368693	41.854656	18.345535	...	
0.000000	33.350000	0.000000	0.000000	0.000000	0.000000	...
18.700000	40.500000	4.630000	39.900000	0.000000	...	
654.250000	52.540000	35.630000	79.500000	30.700000	...	
654.280000	53.570000	522.100000	127.500000	30.900000	...	
655.160000	72.760000	654.990000	128.800000	62.900000	...	
Average_Back_Pressure Barrel_Temperature_1 Barrel_Temperature_2 Barrel_Temperature_3 Barrel_Temperature_4						
37477.000000	37477.000000	37477.000000	37477.000000	37477.000000	37477.000000	...
36.580767	169.379563	273.179625	274.535830	269.174136	...	
33.234145	136.544147	12.796782	12.004843	9.526382	...	
0.000000	0.000000	244.200000	234.400000	239.100000	...	
0.000000	0.000000	264.800000	269.900000	269.800000	...	
58.200000	246.000000	265.400000	270.100000	270.200000	...	
60.800000	286.000000	285.100000	285.000000	275.200000	...	
136.000000	287.600000	286.700000	286.000000	278.100000	...	
Barrel_Temperature_5 Barrel_Temperature_6 Hopper_Temperature Mold_Temperature_3 Mold_Temperature_4						
37477.000000	37477.000000	37477.000000	37477.000000	37477.000000	37477.000000	...
259.571657	235.694215	58.029599	14.053569	15.813630	...	
8.061772	4.830513	9.782343	11.607758	13.580681	...	
234.400000	219.500000	33.200000	0.000000	0.000000	...	
259.800000	234.800000	58.000000	0.000000	0.000000	...	
260.300000	235.100000	61.300000	20.500000	22.000000	...	
265.000000	239.900000	64.100000	22.400000	24.000000	...	
266.900000	241.000000	71.200000	34.800000	45.900000	...	

[코드 38] 제품 'RG3'의 클래스 변수별 통계량 확인

### ③-6. RG3 제품/corr 함수를 통한 변수 간 상관관계 파악 가이드

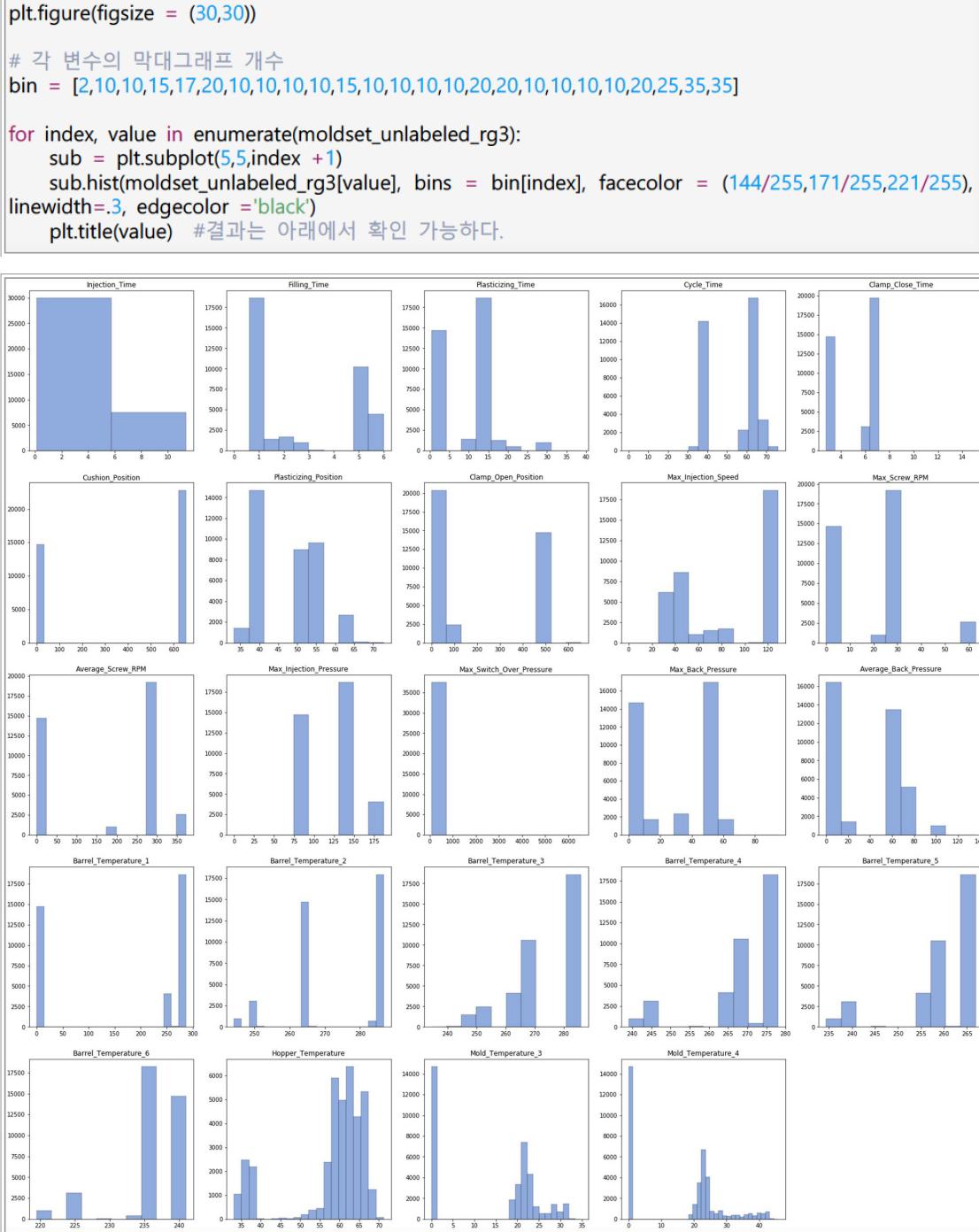
- 제품 'RG3'의 클래스 변수가 존재하지 않는 데이터의 변수 간 상관관계를 확인한다.



[코드 39] 제품 'RG3'의 클래스 변수별 상관관계 확인

### ③-7. RG3 제품/Histogram을 통한 변수별 데이터 파악 가이드

- 제품 'RG3'의 클래스 변수가 존재하지 않는 데이터의 변수별 히스토그램을 확인 한다.



[코드 40] 제품 'RG3'의 클래스 변수별 히스토그램 확인

## [단계 ④] 데이터 정제(전처리)

### ④-1. 불필요 데이터 제거 가이드

- 데이터 품질검사, 데이터 정제(전처리)·가공 실습 내용

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

◦ 데이터 내의 목푯값인 ‘PassOrFail’ 열(column)의 값이 문자열 ‘Y’와 ‘N’으로 되어 있으므로, 이를 각각 ‘0’(양품)과 ‘1’(불량품)로 변환해준다.

```
moldset_labeled_rg3['PassOrFail'] = moldset_labeled_rg3['PassOrFail'].map({0:int(0), 1:int(1)})  
moldset_labeled_cn7['PassOrFail'] = moldset_labeled_cn7['PassOrFail'].map({0:int(0), 1:int(1)})
```

[코드 41] 데이터 목푯값 데이터 형식 변경하기 \_ 실수에서 정수 형태로

### ④-2. 양품 및 불량품 개수 확인 가이드

- 데이터 내의 양품(0)과 불량품(1) 개수를 확인한다.

```
labeled_data = [moldset_labeled_cn7, moldset_labeled_rg3]  
for d in labeled_data:  
    print('양품 수: {}'.format(d[d['PassOrFail']==0].shape[0]))  
    print('불량품 수: {}'.format(d[d['PassOrFail']==1].shape[0]))  
    print('*'*10) # 결과는 아래에서 확인 가능하다.
```

```
양품 수: 1398  
불량품 수: 27  
=====  
양품 수: 1157  
불량품 수: 25  
=====
```

[코드 42] 데이터 내 양품 및 불량품 수 확인하기

### ④-3. StandardScaler를 통한 데이터 정규화 가이드

- 전처리 과정으로 값의 정규화를 진행한다. sklearn package에서 제공하는 전처리 함수인 StandardScaler를 사용하여 데이터를 정규화한다.

```
from sklearn.preprocessing import StandardScaler  
  
data = [moldset_labeled_cn7, moldset_labeled_rg3, moldset_unlabeled_cn7, moldset_unlabeled_rg3]  
for d in data:  
    for column in d.columns:  
        if column !='PassOrFail':  
            sc = StandardScaler()  
            d[[column]] = sc.fit_transform(d[[column]])
```

[코드 43] 데이터 정규화 \_ sklearn package 활용

### ④-4. 전처리 데이터 저장 가이드

- 전처리가 완료된 데이터를 사용자가 원하는 컴퓨터의 경로에 csv 파일 형태로 저장

한다. 추후 데이터를 불러올 때는 처음에 사용한 pandas의 read\_csv 함수를 사용하여 동일한 방법으로 불러오면 된다.

```
moldset_labeled_cn7.to_csv(r'moldset_labeled_cn7.csv')  
moldset_labeled_rg3.to_csv(r'moldset_labeled_rg3.csv')
```

[코드 44] 레이블이 있는 데이터를 csv 파일 형식으로 저장하기

- 레이블이 없는 데이터도 동일한 방식으로 저장한다.

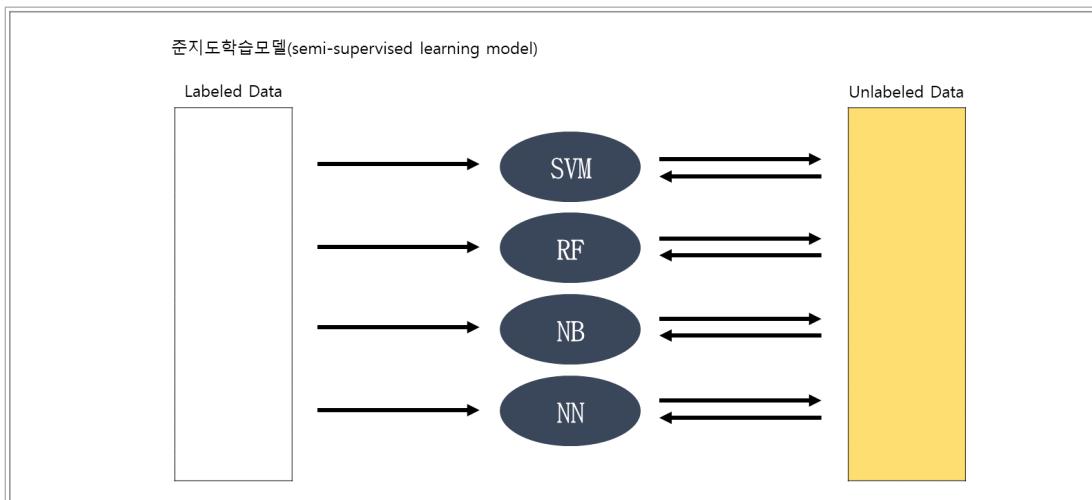
```
moldset_unlabeled_cn7.to_csv(r'moldset_unlabeled_cn7.csv')  
moldset_unlabeled_rg3.to_csv(r'moldset_unlabeled_rg3.csv')
```

[코드 45] 레이블이 없는 데이터를 csv 파일 형식으로 저장하기

## [단계 ⑤] 모델 구축 - I

### ⑤-1. 모델 구축을 위한 패키지 불러오기

- AI 분석모델 구축을 위한 방법론(알고리즘) 적용 및 학습 네트워크 구축 실습
  - 사용할 기계 학습 모델은 sklearn package와 Keras package를 활용하여 구현한다. 전체적인 준지도 학습 방법은 [그림 35]과 같다. 클래스 변수가 있는 데이터를 사용하여 일차적으로 모델을 학습시킨 후, 클래스 변수가 없는 데이터를 선별적으로 추가하는 과정을 반복하며 모델을 개선하는 방식이다.



[그림 35] 준지도 학습 모델 도식화

### ⑤-2. 절대값 반환 함수 생성 가이드

- 학습된 모델이 예측하는 클래스별 확률값 중에서 큰 값의 절대값을 반환하는 함수를 만들어서 사용한다. 이 함수는 준지도 학습 상황에서 엔트로피 최소화를 위한 계산에 사용된다. 아래 두 함수는 동일한 기능을 하는데, 모델의 출력이 각 클래스 변수별 확률로 나타내어질 때 이를 입력으로 받아 가장 큰 확률값을 출력하게 된다. 첫 번째 함수는 행렬을 입력으로 받게 되는데, 행렬은 각 행이 데이터 포인트, 열이 클래스인 형식이다. 이 경우, 각 행마다 가장 큰 값을 가진 클래스 변수만을 사용하여 하나의 벡터를 출력하게 만든다. 두 번째 함수는 0부터 1 사이의 값을 가지는 벡터를 입력으로 받아, 0과 1 중 가까운 클래스 변수만을 선택하여 하나의 벡터로 출력하게 한다.

```
def confident_prediction(df):
    result = []

    for i in range(len(df)):
        if df[i][0] >= df[i][1]:
            result.append(df[i][0])
        else:
            result.append(df[i][1])
    return result

#심층 신경망에서 사용
def confident_prediction(df):
    result = []

    for i in range(len(df)):
        if df[i] >= 0.5:
            result.append(df[i])
        else:
            result.append(1-df[i])
    return result
```

[코드 46] 심층 신경망 사용

### ⑤-3. 평가지표 함수 가이드

- sklearn package의 평가지표 함수를 활용하여 혼동 행렬(confusion matrix), 정확도(accuracy), 정밀도(precision), 재현율(recall)이 반환되는 함수를 만들고 사용한다.

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

def evaluation(y,y_pred):
    print("Accuracy: {:.2f}".format(accuracy_score(y,y_pred)))
    print("Precision: {:.2f}".format(precision_score(y,y_pred)))
    print("Recall: {:.2f}".format(recall_score(y,y_pred)))
    print(roc_auc_score(y,y_pred))
    print(f1_score(y,y_pred))
    print(confusion_matrix(y,y_pred))
```

[코드 47] 평가지표 함수를 활용하여 혼동행렬, 정확도, 정밀도, 재현율을 반환하기

## [단계 ⑥] 모델 구축 - II

### ⑥-1. 데이터 불러오기 및 학습/평가 데이터 분리

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

- 원하는 경로에 저장이 되어있는 모든 데이터(클래스 변수가 있는 것과 없는 것 모두)를 불러오고, 학습 데이터와 평가 데이터를 층화 추출법으로 분리한다. 층화 추출법은 데이터의 클래스 변수의 비율(양품과 불량품의 비율)을 고려하여 학습 데이터와 평가 데이터를 분리하는 방법으로서, 비율이 학습과 평가 시에 동일하게 유지되기 때문에 과적합을 방지하고 모델의 성능을 향상시킬 수 있다. 학습 데이터와 평가 데이터의 비율은 일반적으로 7:3 혹은 8:2 정도로 설정한다.

```
class DataLoader():

    def __init__(self):
        #클래스 변수가 존재하는 데이터를 불러옴
        moldset_labeled      = pd.read_csv(r'./moldset_labeled_cn7.csv',      low_memory=False,
index_col=False)
        moldset_labeled.drop(columns=['Unnamed: 0'], inplace=True)
        #클래스 변수가 존재하지 않는 데이터를 불러옴
        moldset_unlabeled    = pd.read_csv(r'./moldset_unlabeled_cn7.csv',    low_memory=False,
index_col=False)
        moldset_unlabeled.drop(columns=['Unnamed: 0'], inplace=True)
        #평가 데이터의 비율을 test_size로 지정함
        sss = StratifiedShuffleSplit(n_splits=1, test_size=0.3, random_state=42)
        for train_index, test_index in sss.split(moldset_labeled.loc[:, moldset_labeled.columns != 'PassOrFail'], moldset_labeled['PassOrFail']):
            moldset_labeled_train_X = moldset_labeled.loc[:, moldset_labeled.columns != 'PassOrFail'].iloc[train_index]
            moldset_labeled_test_X = moldset_labeled.loc[:, moldset_labeled.columns != 'PassOrFail'].iloc[test_index]
            moldset_labeled_train_Y = moldset_labeled['PassOrFail'].iloc[train_index]
            moldset_labeled_test_Y = moldset_labeled['PassOrFail'].iloc[test_index]
            #학습데이터
            self.moldset_labeled_train_X = moldset_labeled_train_X
            #평가데이터
            self.moldset_labeled_test_X = moldset_labeled_test_X
            #학습데이터의 클래스 변수
            self.moldset_labeled_train_Y = moldset_labeled_train_Y
            #평가데이터의 클래스 변수
            self.moldset_labeled_test_Y = moldset_labeled_test_Y
            #클래스 변수가 없는 데이터
            self.moldset_unlabeled = moldset_unlabeled
```

[코드 48] 클래스 변수 있는 데이터, 클래스 변수 없는 데이터 모두 불러온 후 학습 데이터셋 지정

### ⑥-2. labeled data를 이용한 준지도 학습 가이드

- 클래스 변수가 존재하는 데이터(labeled data)만을 사용하여 준지도 학습 모델에 사용될 하위모델의 하이퍼 파라미터를 조정한다. 하위모델은 기계 학습 모델인 서포트 벡터 머신(Support Vector Machine), 가우시안 나이브 베이즈(Gaussian Naïve Bayes), 랜덤 포레스트(Random Forest)로, 각 모델별 조정가능한, 학습 이전에 설정해야 하는 하이퍼 파라미터가 존재한다. 실제로 기계 학습 모델의 성능은 하이퍼 파라미터에 의해 좌지우지되는 경우가 빈번하며, 데이터에 적합한 각 모

델별 하이퍼 파라미터를 찾고 모델 학습 시에 적절한 값을 사용하는 것이 중요하다. sklearn package의 GridSearch와 StratifiedKFold 함수를 사용하여 데이터를 K 개의 뮤음으로 나누고 반복적으로 적합한 하이퍼 파라미터를 변경해가며 탐색하여 최적의 하이퍼 파라미터를 찾아주는 과정을 진행한다.

```
from sklearn.model_selection import GridSearchCV, KFold, StratifiedKFold
from sklearn.svm import SVC
#탐색할 하이퍼 파라미터의 최대, 최소 범위를 설정한다.
tuned_parameters = [{"kernel": ["rbf"], "gamma": [i*1e-4 for i in range(1, 100)], "C": [i*1e-4 for i in range(1, 100)]}]
scores = ['accuracy', 'precision', 'recall']
weights = {0:100.0, 1:1.0}
#충회죽법을 활용하여 클래스 별 비율이 유지되도록 한다.
kf = StratifiedKFold(random_state=42, n_splits=5, shuffle=True)
X = pd.read_csv(r'./moldset_labeled_cn7.csv', low_memory=False, index_col=False)
X.drop(columns={'Unnamed: 0'}, inplace=True)
y = X.pop('PassOrFail')
for score in scores:
    clf = GridSearchCV(SVC(class_weight=weights), scoring=score, param_grid=tuned_parameters,
    n_jobs=-1, cv=kf, refit=True, verbose=0)
    clf.fit(X, y)
    #최적의 하이퍼 파라미터 짹를 반환한다.
print('Best Params: {}'.format(clf.best_params_)) #결과는 아래에서 확인 가능하다.
```

```
Best Params: {'C': 0.197, 'gamma': 0.71, 'kernel': 'rbf'}
...
Best Params: {'bootstrap': False, 'max_depth': 10, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 1200}
```

[코드 49] 하이퍼 파라미터 변경하며 최적의 값을 찾아주는 과정

### ⑥-3. 모델 학습 및 평가 함수 가이드

- 준지도 학습 상황으로 모델을 학습시키고 평가하는 함수를 만들고 사용함. 클래스 변수가 존재하지 않는 데이터를 반복적으로 가져와서 사용하는 방식이므로 어느 정도 비율로 데이터를 반복적으로 가져올지 정하는 percentage와 전체 클래스 변수가 없는 데이터의 몇 퍼센트를 사용할지 지정하는 unlabeled\_usage 인자를 생성한다. 이후 데이터를 불러오게 되는데, 독립변수와 종속변수 그리고 클래스 변수가 없는 데이터의 독립변수를 따로 불러온다.

```
def train_and_evaluate(percentage=10, unlabeled_usage=90):

    data = DataLoader()
    X_test = data.moldset_labeled_test_X
    Y_test = data.moldset_labeled_test_Y
    X_with_label = data.moldset_labeled_train_X
    Y_with_label = data.moldset_labeled_train_Y
    moldset_unlabeled = data.moldset_unlabeled

    #남은 클래스 변수가 존재하지 않는 데이터의 개수
    num_left_unlabeled = int(moldset_unlabeled.shape[0]*(100-unlabeled_usage)*0.01)
```

[코드 50] 준지도 학습 모델을 학습 시키고 평가하는 함수

## [단계 ⑦] 모델 구축 - III. sklearn Package

### ⑦-1. 준지도 학습 모델 : 학습 및 평가 모델 구축

```
#7-1-1. 모델 학습 및 평가 함수 가이드
def train_and_evaluate(percentage=10, unlabeled_usage=90):

    data = DataLoader()
    X_test = data.moldset_labeled_test_X
    Y_test = data.moldset_labeled_test_Y
    X_with_label = data.moldset_labeled_train_X
    Y_with_label = data.moldset_labeled_train_Y
    moldset_unlabeled = data.moldset_unlabeled
    without_label = moldset_unlabeled

    #남은 클래스 변수가 존재하지 않는 데이터의 개수
    num_left_unlabeled = int(moldset_unlabeled.shape[0]*(100-unlabeled_usage)*0.01)

#7-1-2. 구현된 모델 불러오기 / 모델 정의 가이드
while True:
    if without_label.shape[0] >= num_left_unlabeled:
        #클래스 별 데이터 비율
        weights = {0:100.0, 1:1.0}
        #최적의 하이퍼 파라미터 사용
        model = SVC(C=0.001, kernel='rbf', gamma=1e-2, class_weight=weights, probability=True, random_state=42)
        model.fit(X_with_label, Y_with_label)
        y_pred = model.predict(X_test)
        evaluation(Y_test, y_pred)

    #7-1-3. Unlabeled 데이터에 대한 예측
    prob = model.predict_proba(without_label)
    confident_prob = confident_prediction(prob)
    confident_prediction(prob)
    without_label['confidence'] = confident_prob

    #이 과정에서 내림차순 정렬
    without_label = without_label.sort_values(by=['confidence'], ascending=False)
    length = without_label.shape[0]
    cutting_index = int(length*(percentage*0.01))
    chosen_without_label = without_label.iloc[:cutting_index, :]
    chosen_without_label.drop(columns=['confidence'], inplace=True)
    notchosen_without_label = without_label.iloc[cutting_index:, :].drop(columns=['confidence'])
    pseudo_label = pd.DataFrame(model.predict(chosen_without_label))

    #7-1-4. Labeled 데이터와 통합 후 학습
    #데이터 업데이트
    X_with_label = pd.concat([X_with_label, chosen_without_label])
    Y_with_label = pd.concat([Y_with_label, pseudo_label])
    without_label = notchosen_without_label
    y_pred = model.predict(X_test)
    evaluation(Y_test, y_pred)

else:
    print()
    print("ALL DONE, UNLABELED USED: {:.0%}".format(without_label.shape[0]/num_left_unlabeled))
    break #결과는 아래에서 확인 가능하다.
```

[코드 50] 준지도학습 모델 학습 및 평가모델 구축 \_ 전체 코드

- [코드50]에서 전체 모델 학습 및 평가 함수를 확인하였다. 이 전체 코드 내부에서 중요한 의미를 가지는 부분들을 소분류로 나누어서 자세히 다루어 본다. 아래의 코드들은 위의 [코드50]의 일부분으로, 실습시에는 [코드50]만 활용하여야 한다.

## ⑦-1-1. 모델 학습 및 평가 함수 가이드

```
def train_and_evaluate(percentage=10, unlabeled_usage=90):

    data = DataLoader()
    X_test = data.moldset_labeled_test_X
    Y_test = data.moldset_labeled_test_Y
    X_with_label = data.moldset_labeled_train_X
    Y_with_label = data.moldset_labeled_train_Y
    moldset_unlabeled = data.moldset_unlabeled
    without_label = moldset_unlabeled

        #남은 클래스 변수가 존재하지 않는 데이터의 개수
    num_left_unlabeled = int(moldset_unlabeled.shape[0]*(100-unlabeled_usage)*0.01)
```

[코드 50-1] 준지도 학습 모델을 학습 시키고 평가하는 함수의 변수 정의

- 준지도 학습 상황으로 모델을 학습시키고 평가하는 함수를 만들고 사용함. 클래스 변수가 존재하지 않는 데이터를 반복적으로 가져와서 사용하는 방식이므로 어느 정도 비율로 데이터를 반복적으로 가져올지 정하는 percentage와 전체 클래스 변수가 없는 데이터의 몇 퍼센트를 사용할지 지정하는 unlabeled\_usage 인자를 생성한다. 이후 데이터를 불러오게 되는데, 독립변수와 종속변수 그리고 클래스 변수가 없는 데이터의 독립변수를 따로 불러온다.

## ⑦-1-2. 구현된 모델 불러오기/모델 정의 가이드

```
while True:
    if without_label.shape[0] >= num_left_unlabeled:
        #클래스 별 데이터 비율
        weights = {0:100.0, 1:1.0}
        #최적의 하이퍼 파라미터 사용
        model = SVC(C=0.001, kernel='rbf', gamma=1e-2, class_weight=weights,
probability=True, random_state=42)
        model.fit(X_with_label, Y_with_label)
        y_pred = model.predict(X_test)
        evaluation(Y_test, y_pred)
```

[코드 50-2] 구현된 모델 불러오는 부분

- sklearn package에 구현되어있는 모델을 불러오고 이전에 찾은 최적의 하이퍼 파라미터를 사용하여 모델을 정의한다. 모델의 fit 함수를 활용하여 모델을 독립변수와 종속변수를 사용하여 학습시키고 평가 데이터에 대해 평가를 진행한다.

### ⑦-1-3. 구현된 모델 불러오기/모델 정의 가이드

```
prob = model.predict_proba(without_label)
confident_prob = confident_prediction(prob)
confident_prediction(prob)
without_label['confidence'] = confident_prob

#이 과정에서 내림차순 정렬
without_label = without_label.sort_values(by=['confidence'], ascending=False)
length = without_label.shape[0]
cutting_index = int(length*(percentage*0.01))
chosen_without_label = without_label.iloc[:cutting_index, :]
chosen_without_label.drop(columns={'confidence'}, inplace=True)
notchosen_without_label = without_label.iloc[cutting_index:, :].drop(columns={'confidence'})
pseudo_label = pd.DataFrame(model.predict(chosen_without_label))
```

[코드 50-3] unlabeled 데이터에 대하여 예측하기

- 이전에 학습시킨 모델을 사용하여 새롭게 목표값이 없는 데이터에 대한 예측을 실시한다. 예측 결과는 각 클래스별(양품 혹은 불량품) 확률로 출력이 된다. 이전에 만든 함수를 활용하여 엔트로피가 가장 낮은(가장 모델이 높은 확신을 가지고 출력한) 예측값을 반환한 뒤에 클래스 변수가 없는 데이터를 이 확신도를 기준으로 내림차순 정렬한다. 이후 정렬된 데이터에서 상위 10%(전 단계에서 설정한 percentage)의 클래스 변수가 없는 데이터에 새로운 예측값을 사용해 클래스 변수(pseudo-label)를 부여한다. 가상의 클래스 변수가 생긴 데이터는 기존의 클래스 변수가 없는 데이터에서 분리해내고, 기존의 클래스 변수가 없는 데이터는 그대로 둔다.

### ⑦-1-4. Labeled 데이터와 통합 후 학습

```
#데이터 업데이트
X_with_label = pd.concat([X_with_label, chosen_without_label])
Y_with_label = pd.concat([Y_with_label, pseudo_label])
without_label = notchosen_without_label
y_pred = model.predict(X_test)
evaluation(Y_test, y_pred)

else:
    print()
    print("ALL DONE, UNLABELED USED: {:.0%}".format(without_label.shape[0]/num_left_unlabeled))
    break #결과는 아래에서 확인 가능하다.
```

[코드 50-4] labeled 데이터와 통합 후 학습

- 새로운 클래스 변수가 생긴 데이터를 기존의 클래스 변수가 있는 데이터와 통합하고, 나머지 클래스 변수가 없는 데이터는 유지한다. 위의 과정은 클래스 변수가 없는 데이터의 일정량을 모두 사용하게 되는 시점까지 반복되며, 이후에는 학습과 평가 과정이 종료된다.

## ⑦-2. 저장한 함수 실행하기

```
train_and_evaluate()
```

```
Accuracy: 0.92
Precision: 0.45
Recall: 0.93
0.9230145093547156
0.6024096385542168
[[25  31]
 [ 2  357]]
```

```
ALL DONE, UNLABELED USED: 90%
```

[코드51] 저장한 학습 및 평가 함수 수행하기

- 출력값들 중 제일 아래에서 최종 함수 실행 결과물을 확인할 수 있다.
- 학습을 위한 데이터의 특성 상 레이블 간의 불균형이 심하기 때문에, 모델이 학습하는 데에 어려움을 겪을 수 있다.
- 실습 상황에서는 이러한 문제를 해결하기 위해 오버 샘플링, 언더 샘플링 등의 방법을 사용하여 문제를 해결할 수 있다. 오버 샘플링은 개수가 적은 레이블에 속하는 데 이터의 비중을 임의로 높이는 방법으로서, 새로운 데이터를 생성하거나 기존의 데이터를 복제하는 등의 방식을 사용한다. 언더 샘플링은 반대로 수가 많은 레이블의 데이터를 적게 샘플링하여 두 레이블 각각의 데이터 비율을 맞추는 방법이다.
- 이러한 방법론은 불균형의 정도와 종류에 따라 적용할 수 있는 방법이 달라질 수 있고, 임의로 데이터를 조정하여 사용하는 방법이라고 할 수 있다. 따라서, 실제 실습 상황에서는 가이드북에 기재된 값과 상이한 결과가 나타날 수 있다.

## ⑦-3. 최종 출력값 통일을 위한 클래스 변수 변환 함수 제작

```
def predict(df):
    result = []

    for i in range(len(df)):
        if df[i] >= 0.5:
            result.append(1)
        else:
            result.append(0)
    return result
```

[코드 52] 심층 신경망의 최종 출력값의 통일을 위하여, 클래스 변수 변환 함수 만들기

- 심층 신경망의 최종 출력은 하나의 확률값이기 때문에, 이를 클래스 변수로 바꾸어 주는 함수를 만들어 사용한다. 확률값이 0.5 이상일 경우 불량(1)로 예측하고, 0.5 미만의 경우 양품(0)으로 클래스 변수를 생성한다.

## [단계 ⑧] 모델 구축 - IV. Keras Package

### ⑧-1. 학습 및 평가 함수 제작

```
#8-1-1. Keras Package를 활용하여 학습에 필요한 변수 생성
def train_and_evaluate(percentage=10, unlabeled_usage=90):
    data = DataLoader()
    X_test = data.moldset_labeled_test_X
    Y_test = data.moldset_labeled_test_Y
    X_with_label = data.moldset_labeled_train_X
    Y_with_label = data.moldset_labeled_train_Y
    moldset_unlabeled = data.moldset_unlabeled
    without_label = moldset_unlabeled
    #남은 클래스 변수가 존재하지 않는 데이터의 개수
    num_left_unlabeled = int(moldset_unlabeled.shape[0]*(100-unlabeled_usage)*0.01)

    #8-1-2. 신경망을 순차적으로 쌓음
    model = Sequential()
    #units은 출력 차원 수
    model.add(Dense(units=32, activation='relu', input_dim=24))
    model.add(Dense(units=64, activation='relu'))
    model.add(Dropout(0.25))
    model.add(Dense(units=32, activation='relu'))
    model.add(Dropout(0.2))
    model.add(Dense(units=16, activation='relu'))
    model.add(Dense(units=1, activation='sigmoid'))
    model.compile(loss='binary_crossentropy',
                  optimizer='adam',
                  metrics=[accuracy,tf.keras.metrics.Precision(name='precision'),tf.keras.metrics.Recall(name='recall')])

    #8-1-3. 실증 신경망 모델을 이용하여 학습 진행하기
    while True:
        if without_label.shape[0] >= num_left_unlabeled:
            val_precision = 0
            val_recall = 0
            patience = 10
            cnt = 0

            for epoch in range(100):
                history = model.fit(X_with_label, Y_with_label, epochs=1, validation_split=0.3)
                history = history.history
                if cnt>= patience:
                    break
                if history['val_precision'][0] >= val_precision and history['val_recall'][0] >= val_recall:
                    val_precision = history['val_precision'][0]
                    val_recall = history['val_recall'][0]
                    cnt = 0
                else:
                    cnt+=1

            #8-1-4. 학습시킨 모델을 이용하여 목표값이 없는 데이터에 대한 예측 실행
            prob = model.predict_proba(without_label)
            confident_prob = confident_prediction_1(prob)
            without_label['confidence'] = confident_prob
            without_label = without_label.sort_values(by=['confidence'], ascending=False)
            length = without_label.shape[0]
            cutting_index = int(length*(percentage*0.01))
            chosen_without_label = without_label.iloc[:cutting_index, :]
            chosen_without_label.drop(columns=['confidence'], inplace=True)
            notchosen_without_label = without_label.iloc[cutting_index:, :].drop(columns=['confidence'])
            pseudo_label = pd.DataFrame(predict(model.predict(chosen_without_label)))

            #8-1-5. UPDATE
            X_with_label = pd.concat([X_with_label, chosen_without_label])
            Y_with_label = pd.concat([Y_with_label, pseudo_label])
            without_label = notchosen_without_label

        else:
            print()
            print("ALL DONE, UNLABELED USED: {:.0%}".format(without_label.shape[0]/num_left_unlabeled))
            model.save('NN_lr3.h5')
            break #결과는 아래에서 확인 가능하다.
```

[코드 53] Keras package를 활용하여, 준지도 학습 과정 진행하기 \_ 학습 및 평가

- [코드53]에서 전체 모델 학습 및 평가 함수를 확인하였다. 이 전체 코드 내부에서 중요한 의미를 가지는 부분들을 소분류로 나누어서 자세히 다루어 본다. 아래의 코드들은 위의 [코드53]의 일부분으로, 실습시에는 [코드53]만 활용하여야 한다.

### ⑧-1-1. 준지도 학습 평가를 위한 변수 생성

```
def train_and_evaluate(percentage=10, unlabeled_usage=90):  
  
    data = DataLoader()  
    X_test = data.moldset_labeled_test_X  
    Y_test = data.moldset_labeled_test_Y  
    X_with_label = data.moldset_labeled_train_X  
    Y_with_label = data.moldset_labeled_train_Y  
    moldset_unlabeled = data.moldset_unlabeled  
    without_label = moldset_unlabeled  
  
    #남은 클래스 변수가 존재하지 않는 데이터의 개수  
    num_left_unlabeled = int(moldset_unlabeled.shape[0]*(100-unlabeled_usage)*0.01)
```

[코드 53-1] 준지도 학습 평가를 위한 변수 생성

- Keras package를 활용한 심층 신경망은 모델의 정의와 학습 설정만 상이할 뿐 나머지 준지도 학습 과정은 동일하게 진행하면 된다. `train_and_evaluate`이라는 함수를 만들고 같은 방법으로 데이터를 불러온다.

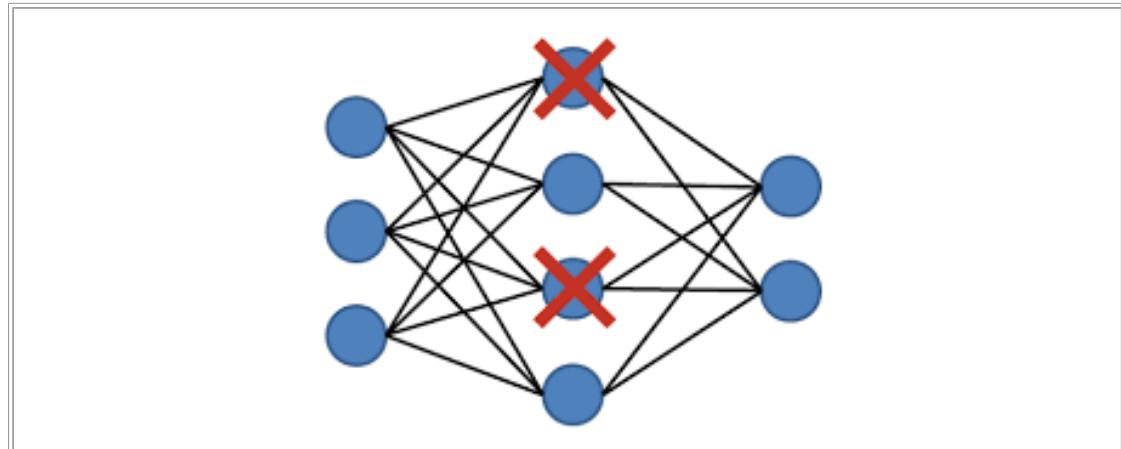
### ⑧-1-2. Sequential 모델에 신경망 쌓기

```
#신경망을 순차적으로 쌓음  
model = Sequential()  
    #units은 출력 차원 수  
model.add(Dense(units=32, activation='relu', input_dim=24))  
model.add(Dense(units=64, activation='relu'))  
model.add(Dropout(0.25))  
model.add(Dense(units=32, activation='relu'))  
model.add(Dropout(0.2))  
model.add(Dense(units=16, activation='relu'))  
model.add(Dense(units=1, activation='sigmoid'))  
model.compile(loss='binary_crossentropy',  
                optimizer='adam',  
                metrics=['accuracy',tf.keras.metrics.Precision(name='precision'),tf.keras.metrics.Recall(name='recall')])
```

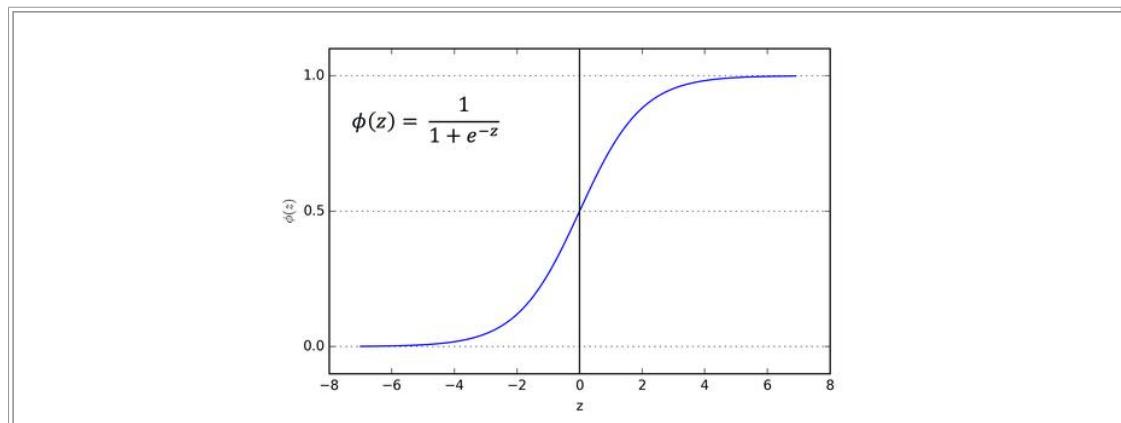
[코드 53-2] 신경망을 순차적으로 쌓아보기

- 이후 Keras package의 Sequential 모델을 생성하고, 사용하려고 하는 신경망인 완전연결층의 차원수(units)와 활성화함수(activation function)을 지정한다. 완전연결층 사이에 드랍아웃(dropout) [그림 36]이라는 추가적인 신경망을 연결하게 된다. 드랍아웃은 정해진 비율만큼 각 신경망의 출력을 0으로 만들게 되며 이는 모델의 견고함(robustness)을 제고하고 과대적합을 방지하게 하는 규제(regularization)의 일종이다. 중간 부분의 신경망에는 ReLU [그림 30]라고 부르는 좋은 성능을 내는 활성화함수를 사용했다. 가장 마지막의 신경망은 확률로 출력이 되게 하기 위한 시그모이드(sigmoid) 함수가 활성화함수로 사용된다. 시그모이드 함수의 식은 광고, 결과값은 [그림 37]과 같이 0과 1사이로 제한된다. 모델의 학습 정도를 가늠하는 손실 함수(loss)는 이진 교차엔트로피(binary\_crossentropy)를 사용했으며 교차엔트로피는 분류 목적의 AI 알고리즘을 학습시킬 때 가장 많이 사용되는 함수로서, 실제 클래스 변수(0 또는 1)와 근접할 때 낮은 값을 가지고 반대의 경우에 높은 값을

가지게 된다. 신경망의 파라미터를 최적화하는 역할을 하는 옵티마이저(optimizer)는 현재 가장 많이 활용되는 Adam을 사용했다. 모델이 전체 데이터를 순회하며 신경망의 파라미터를 학습하는 과정마다 각 시점의 모델의 정확도(accuracy), 정밀도(precision), 그리고 재현율(recall)이 출력되도록 한다.



[그림 36] 드랍 아웃 연결



[그림 37] 시그모이드 활성화함수

### ⑧-1-3. Label 데이터에 대해 모델 학습

```
while True:  
    if without_label.shape[0] >= num_left_unlabeled:  
  
        val_precision = 0  
        val_recall = 0  
        patience = 10  
        cnt = 0  
  
        for epoch in range(100):  
            history = model.fit(X_with_label, Y_with_label, epochs=1, validation_split=0.3)  
            history = history.history  
  
            if cnt >= patience:  
                break  
  
            if history['val_precision'][0] >= val_precision and history['val_recall'][0] >= val_recall:  
                val_precision = history['val_precision'][0]  
                val_recall = history['val_recall'][0]  
                cnt = 0  
  
        else:  
            cnt += 1
```

[코드 53-3] 심층 신경망 모델을 이용하여 학습 진행하기

- 이전에 선언한 심층 신경망 모델을 사용하고, 불리온 데이터를 이용해 학습을 진행 한다. 에폭(epoch)은 모델이 전체 데이터를 한번 다 사용했다는 단위이다. 이를 최대 100번으로 임의로 설정한 후, 모델의 fit 함수를 활용하여 모델을 독립변수와 종속변수를 사용하여 학습을 시작한다. sklearn package를 사용한 분석과 다르게, 심층 신경망에서는 검증 데이터셋을 학습시에 자동으로 사용하여, 이후에 실시하는 평가 데이터셋에 대한 일반화 성능을 짐작한다. patience로 설정한 에폭의 수만큼 검증 데이터셋에 대한 평가에서의 정밀도와 재현율의 개선이 이루어지지 않으면 이후 단계로 넘어가게 되고, 개선이 이루어지면 최대 100 에폭까지 학습이 진행된다.

### ⑧-1-4. Unlabel 데이터에 대해 예측

```
prob = model.predict_proba(without_label)  
confident_prob = confident_prediction_1(prob)  
without_label['confidence'] = confident_prob  
without_label = without_label.sort_values(by=['confidence'], ascending=False)  
length = without_label.shape[0]  
cutting_index = int(length*(percentage*0.01))  
chosen_without_label = without_label.iloc[:cutting_index, :]  
chosen_without_label.drop(columns={'confidence'}, inplace=True)  
notchosen_without_label = without_label.iloc[cutting_index:, :].drop(columns={'confidence'})  
pseudo_label = pd.DataFrame(predict(model.predict(chosen_without_label)))
```

[코드 53-4] 학습시킨 모델을 이용하여 목표값이 없는 데이터에 대한 예측 실행하기

- 이전에 학습시킨 모델을 사용하여 새롭게 목표값이 없는 데이터에 대한 예측을 시한다. 예측 결과는 각 클래스별(양품 혹은 불량품) 확률로 출력이 된다. 이전에 만

든 함수를 활용하여 엔트로피가 가장 낮은(가장 모델이 높은 확신을 가지고 출력한) 예측값을 반환한 뒤에 클래스 변수가 없는 데이터를 이 확신도를 기준으로 내림차순 정렬한다. 이후 정렬된 데이터에서 상위 10%(전 단계에서 설정한 percentage)의 클래스 변수가 없는 데이터에 새로운 예측값을 사용해 클래스 변수(pseudo-label)를 부여한다. 가상의 클래스 변수가 생긴 데이터는 기존의 클래스 변수가 없는 데이터에서 분리해내고, 기존의 클래스 변수가 없는 데이터는 그대로 둔다.

### ⑧-1-5. Labeled 데이터와 통합 후 학습

- 새로운 클래스 변수가 생긴 데이터를 기존의 클래스 변수가 있는 데이터와 통합하고, 나머지 클래스 변수가 없는 데이터는 유지한다. 위의 과정은 클래스 변수가 없는 데이터의 일정량을 모두 사용하게 되는 시점까지 반복되며, 이후에는 학습과 평가 과정이 종료된다. 최종적으로 학습된 모델은 원하는 이름에 h5라는 확장자명을 붙여 저장하고, 저장된 모델은 추후에 불러올 수 있다.

```
#UPDATE
X_with_label = pd.concat([X_with_label, chosen_without_label])
Y_with_label = pd.concat([Y_with_label, pseudo_label])
without_label = notchosen_without_label

else:
    print()
    print("ALL DONE, UNLABELED USED:
{:.0%}".format(without_label.shape[0]/num_left_unlabeled))
    model.save('NN_rg3.h5')
    break      #결과는 아래에서 확인 가능하다.
```

[코드 53-5] 새로운 클래스 변수가 생긴 데이터를 기준 데이터와 통합 후 학습, 평과 과정 종료

### ⑧-1-6. 저장한 함수 실행하기

```
train_and_evaluate()
```

```
Train on 701 samples, validate on 301 samples
Epoch 1/1
701/701 [=====] - 1s 1ms/step - loss: 0.5908 - accuracy: 0.7989 - precision: 0.3525 - recall: 0.2184 - val_loss: 0.5251 - val_accuracy: 0.7973 - val_precision: 0.3684 - val_recall: 0.0861
Train on 701 samples, validate on 301 samples
Epoch 1/1
701/701 [=====] - 0s 325us/step - loss: 0.4881 - accuracy: 0.8131 - precision: 0.3684 - recall: 0.0549 - val_loss: 0.4880 - val_accuracy: 0.7973 - val_precision: 0.3684 - val_recall: 0.0393
...
...
...
Train on 701 samples, validate on 301 samples
Epoch 1/1
701/701 [=====] - 0s 233us/step - loss: 0.2619 - accuracy: 0.8402 - precision: 0.6407 - recall: 0.0244 - val_loss: 0.2381 - val_accuracy: 0.8605 - val_precision: 0.7037 - val_recall: 0.0345
```

[코드 59] 새로운 클래스 변수가 생긴 데이터를 기준 데이터와 통합 후 학습, 평과 과정 종료

- 출력값들 중 제일 아래에서 최종 함수 실행 결과물을 확인할 수 있다.

### ⑧-1-7. 심층 신경망 모델 불러오기

```
from tensorflow.keras.models import load_model

data = DataLoader()
Y_test = data.moldset_labeled_test_Y
X_test = data.moldset_labeled_test_X

model = load_model('NN_rg3.h5')
```

[코드 53-7] 심층 신경망 모델 불러오기

- 이전에 저장한 심층 신경망 모델을 load\_model 함수로 불러온다. 평가 데이터를 다시 불러오기 위해 이전에 만든 DataLoader 함수를 다시 사용한다.

### ⑧-2. 모델을 통한 예측 후 평가 실시

- 불러온 모델을 사용하여 평가 데이터에 대한 클래스 변수를 예측하고, 실제 클래스 변수를 사용해 evaluation 함수로 평가를 실시한다.

```
y_pred = predict(model.predict(X_test))

evaluation(Y_test, y_pred) # 결과는 아래에서 확인 가능하다.
```

```
Accuracy: 0.98
Precision: 0.89
Recall: 1.00
0.9855907780979828
0.9431818181818181
```

```
[[337 10]
 [ 0 83]]
```

[코드 54] 실제 클래스 변수를 사용하여 평가하기

## [단계 ⑨] 결과 분석 및 해석

### ⑨-1. 분석 결과값 도출

- 분석 결과값 도출
  - 평가 지표를 활용한 분석 결과값

모델	정확도	정밀도	재현율	ROC-AUC	F1
서포트 벡터 머신	0.85	0.77	0.33	0.65	0.46
랜덤 포레스트	0.98	0.91	1.00	0.98	0.95
가우시안 나이브 베이즈	0.22	0.20	1.00	0.51	0.33
심층 신경망	0.98	0.89	1.00	0.98	0.94

[표 4] 제품 rg3 결과값

모델	정확도	정밀도	재현율	ROC-AUC	F1
서포트 벡터 머신	0.92	0.93	0.45	0.72	0.61
랜덤 포레스트	0.99	0.97	1.00	0.99	0.98
가우시안 나이브 베이즈	0.77	0.37	1.00	0.86	0.54
심층 신경망	0.99	0.90	1.00	0.99	0.95

[표 5] 제품 cn7 결과값

#### - 혼동 행렬로 나타낸 분석 결과값

	실제 불량(1)	실제 양품(0)
불량(1) 예측	27	8
양품(0) 예측	56	339

[표 6] 서포트 벡터 머신 rg3

	실제 불량(1)	실제 양품(0)
불량(1) 예측	83	8
양품(0) 예측	0	339

[표 7] 랜덤 포레스트 rg3

	실제 불량(1)	실제 양품(0)
불량(1) 예측	83	337
양품(0) 예측	0	10

[표 8] 가우시안 나이브 베이즈 rg3

	실제 불량(1)	실제 양품(0)
불량(1) 예측	83	10
양품(0) 예측	0	337

[표 9] 심층 신경망 rg3

	실제 불량(1)	실제 양품(0)
불량(1) 예측	25	2
양품(0) 예측	31	357

[표 10] 서포트 벡터 머신 cn7

	실제 불량(1)	실제 양품(0)
불량(1) 예측	56	2
양품(0) 예측	0	357

[표 11] 랜덤 포레스트 cn7

	실제 불량(1)	실제 양품(0)
불량(1) 예측	56	96
양품(0) 예측	0	263

[표 12] 가우시안 나이브 베이즈 cn7

	실제 불량(1)	실제 양품(0)
불량(1) 예측	56	6
양품(0) 예측	0	353

[표 13] 심층 신경망 cn7

#### ⑨-2. 분석 결과 해석

##### - 분석결과에 대한 논의 및 해석(implication)

###### ◦ 전통적인 기계 학습 방법론

분석에 사용된 두 가지 제품(rg3, cn7) 각각에 대해 AI 분석 방법론을 적용한 결과 알고리즘별로 각 데이터셋에 대해 비슷한 성능을 보였다. 랜덤 포레스트와 심층 신경망이 여러 평가 지표에서 우수한 성능을 보임을 알 수 있다. 앞서 설명된 바와 같이 모델의 성능은 정확도도 중요하지만, 실제 공정에서 발생하는 불량품을 얼마나 잘 예측하느냐를 평가하는 재현율(recall)도 매우 중요하다고 볼 수 있다. 그러나 재현율만 높다고 우수한 모델은 아닐 수 있다. 왜냐하면, 모든 예측을 불량(1)으로 하게 되면, 다른 평가지표들을 무시한 채 재현율만을 높일 수 있기 때문이다. 이러한 점을 고려할 때, 사용된 AI 방법론 중 랜덤 포레스트와 심층 신경망은 균형

있는 평가지표에서의 결과를 보여줌으로써 훌륭한 성능을 가진다고 할 수 있다.

#### - 심층 신경망

심층 신경망은 모델의 학습 시에 다른 AI 방법론보다 고려해야 할 사항이 많다. 신경망의 개수, 각 신경망의 노드 수(차원 수), 손실 함수, 에폭 수 등의 무수히 많은 하이퍼 파라미터가 존재하기 때문이다. 또한 신경망을 깊고 넓게 구축할수록 모델이 주어진 학습데이터에 대해 과대적합될 가능성이 높아진다. 이러한 학습 시의 난이도가 존재함에도 불구하고, 최근 학계 및 산업에서의 심층 신경망에 대한 관심과 연구 결과들은 심층 신경망의 잠재적 가능성을 방증한다.

\* (Manufacturing Data translator 관점) 제조데이터 트랜스레이터 관점에서 결과에 대한 해석과 함의 기술

### 3. 유사 타 현장의 「사출성형기 AI 데이터셋」 분석 적용

#### 3.1 본 분석이 적용 가능한 제조현장 소개

- 사출성형에는 많은 방식이 있다. 대표적으로 용융시킨 원료(주로 열가소성 수지)를 금형에 주입하고 냉각시켜 금형을 열어 배출시키며 모양을 완성하는 방식을 뜻하는 사출성형(injection molding), 유동성이 있는 원료를 넣고 틀 모양을 따라 눌러 제품을 빼내며 만 들어지는 압출성형(extrusion molding)이 있는데 제품의 유형에 따라 다른 유형의 공정을 적용한다. 하지만 아래와 같은 공통적인 프로세스가 존재하는데 이와 같은 사출 프로세스를 적용하는 금형사출 제조현장에는 본 분석을 적용할 수 있다.

\* 공통적인 금형 사출 프로세스

- a. 건조된 중합체를 호퍼에 넣고 제품의 스펙에 따라 착색 안료 또는 강화 첨가제를 추가하고 원료는 배럴로 공급됨과 동시에 가열되어 혼합된다.
- b. 액체 상태의 재료가 스크루를 통해 적절한 온도와 압력을 가진 상태로 이송되고 재료가 노즐을 통해 금형으로 가득 주입되어 형상을 갖춘다.
- c. 재료가 응고되면 금형이 열리고, 이젝터 핀이 전진하며 사출물을 이탈시킨다. a~c 과정이 빠르게 반복된다..

#### 3.2 본 「사출성형기 AI 데이터셋」 분석을 원용하여 타 제조현장 적용 시, 주요 고려사항

- 개별추적 가능한 불량데이터 관리가 필요하다. 사출성형품의 특성상 제품이 쏟아져 나오는 경우가 있는데 반드시 개별추적 가능한 시스템이 구축되어야 하며 불량 유형별로 데이터 관리가 되어야 한다.
- 수집하는 데이터의 형태에 대한 논의 및 관리가 필요하다. 현재는, 현장에서 수집되는 데이터와 AI 분석을 위한 데이터에는 많은 전처리 과정이 필요하다. 데이터 수집을 하면서 현장 실무자와 AI 분석 전문가가 꾸준히 커뮤니케이션하면서 데이터 수집 방향 및 형태에 대해서 논의하여 수집 환경을 구축하는 것이 중요하다.

## 부록 데이터 품질 전처리 [실습 코드]

앞선 ‘2.분석 실습’에서 데이터 품질 전처리에 필요한 5가지 특성에 대하여 논의하였다. (완전성, 유일성, 유효성, 일관성, 정확성)

### • 데이터 품질 지수 : 세부 설명

- 완전성 품질 지수 =  $((1-\text{결측치})/\text{전체 데이터수}) \times 100$

① Null 값이 30%이상인 데이터들은 데이터의 완전성이 떨어지기 때문에 컬럼별 Null값의 비율을 확인하여 삭제한다. 이 데이터는 30%를 넘는 컬럼이 존재하지 않으므로 컬럼을 제거하지 않는다.

```
perc=30
dt.isnull().sum()/len(dt)*100
```

_id	0.0
TimeStamp	0.0
PART_FACT_PLAN_DATE	0.0
PART_FACT_SERIAL	0.0
Shot_Number	0.0
PART_NO	0.0
PART_NAME	0.0
EQUIP_CD	0.0
EQUIP_NAME	0.0
WorkingNum	0.0
PassingorFail	0.0
Reason	0.0
Injection_Time	0.0
Filling_Time	0.0
Plasticizing_Time	0.0
Cycle_Time	0.0
Clamp_Close_Time	0.0
Cushion_Position	0.0
Switch_Over_Position	0.0
Plasticizing_Position	0.0
Clam_Open_Position	0.0
Max_Injection_Speed	0.0

```
perc=30
dt.isnull().sum()/len(dt)*100>perc
```

_id	False
TimeStamp	False
PART_FACT_PLAN_DATE	False
PART_FACT_SERIAL	False
Shot_Number	False
PART_NO	False
PART_NAME	False
EQUIP_CD	False
EQUIP_NAME	False
WorkingNum	False
PassingorFail	False
Reason	False
Injection_Time	False
Filling_Time	False
Plasticizing_Time	False
Cycle_Time	False
Clamp_Close_Time	False
Cushion_Position	False
Switch_Over_Position	False
Plasticizing_Position	False
Clam_Open_Position	False
Max_Injection_Speed	False



[코드 62] : 분석 도구를 활용한 완전성 품질 지수 확인

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

idx	Machine_Name	Item No	working time	Press time(ms)	Pressure 1	Pressure 2	Pressure 5
0	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...
64354	False	False	False	False	False	True	False
64355	False	False	False	False	False	True	False
64356	False	False	False	False	False	True	False
64357	False	False	False	False	False	True	False
64358	False	False	False	False	False	True	False

```
dt.isnull().sum()
_id           0
TimeStamp     0
PART_FACT_PLAN_DATE 0
PART_FACT_SERIAL 0
Shot_Number    0
PART_NO         0
PART_NAME       0
EQUIP_CD        0
EQUIP_NAME      0
WorkingNum      0
PassOrFail       0
Reason          0
Injection_Time   0
Filling_Time     0
Plasticizing_Time 0
```

```
cmpt_len = dt.isnull().sum().sum()
print(cmpt_len)
0
```

[그림 38] : 분석 도구를 활용한 데이터 결측치 확인

③ 구한 결측치의 개수를 이용하여 완전성 품질지수를 구한다.

```
print("결측치 = %d개 \n완전성 지수 : %.2f%% "%(cmpt_len,(1-cmpt_len/len(dt))*100))
#결과는 아래에서 확인 가능하다.
```

```
결측치 = 0개
완전성 지수 : 100.00%
```

[코드 63] : 분석 도구를 활용한 완전성 품질 지수 구하기

- 유일성 품질지수 = ((유일한 데이터수)/전체 데이터수) × 100

: 이 데이터는 '\_id' 컬럼이 유일한 값을 가지는 기본키이다.

① unique() 함수를 이용하여 '\_id' 컬럼이 가지는 유일한 값을 확인하고, 유일한 데이터 개수를 구한다.

```
dt['_id'].unique()
len(dt['_id'].unique())
```

```
array(['5f8928b99c0189cc666ef194','5f892
8bb9c0189cc666ef19b','5f8928de9c0189cc
666ef20b',...,'5fa112879c0189cc66dabe50',
'5fa112bb9c0189cc66dac22a','5fa112bc9c0
189cc66dac23a'],dtype=object)
```

[코드 64] : 분석 도구를 활용한 유일성 품질지수 값 확인하기

② 구한 유일한 데이터 개수를 이용하여 유일성 품질지수를 구한다.

```
uniq_len=len(dt['id'].unique())
print("유일성 지수 : %.2f%%"%(uniq_len/len(dt)*100)) #결과는 아래에서 확인 가능하다.
```

유일성 지수 : 100.00%

[코드 65] : 분석 도구를 활용한 유일성 품질지수 구하기

### - 유효성 품질지수

① 데이터가 유효범위내에 들어가 있는가

: Data set에 정의된 수집 범위를 이용하여 조건(유효범위)을 모두 충족하는 데이터들을 vald\_dt에 저장한다.

The screenshot shows a Jupyter Notebook cell containing a data frame and some Python code. The data frame has columns for various time-related parameters like 'Injection\_Time', 'Filling\_Time', etc., and their corresponding ranges. A large orange bracket on the right side groups several rows of data. To the right of this bracket is a block of Python code with comments explaining the filtering conditions (a) through (w). The code uses range operators (gt, lt, gte, lte) to filter rows based on specific temperature ranges.

Injection_Time	고압+사출시간(고압(사출압) : 재료를 금형에 유입시킬때의 압력), 사출시간: 재료를 금형에 유입시키는동안 소요되는 시간)	9.4~13.4	9~14
Filling_Time	충전시간으로 사용(제품을 금형으로 내용물이 주입되는 시간)	3.3~8.3	3~9
Plasticizing_Time	계량시간으로 재료를 스크류에 1번 생산할 만큼 품용되어 저장되는 시간	16.5~21.1	16~22
Cycle_Time	1번의 제품생산에 소요되는 생산시간	50.8~64.3	50~65
Clamp_Close_Time	제품이 생산되고 난후 열려있는 금형을 사용하기가 당아주고 번거로워서 제품을 제작하는 데 걸리는 시간	6.1~7.2	6~8
Cushion_Position	보입(사출압의 다음으로 가해지는 압력(금형내부 압력을 조절하여 압력을 방지)을 하기위한 스트류의 위치)	65.3~655	651~656
Switch_Open_Position	고압,보입을 허용하지고(사출압)에서 압력으로 전환되는 위치	0~0	0~0
Plasticizing_Position	계량한 뒤 제품을 미리 스크류의 위치)	59.8~68.9	59~69
Clamp_Open_Position	제품이 생산되어 추출하기 위해 금형이 열리고 난 위치	69.6~648	70~649
Max_Injection_Speed	배럴에 저항되어 있는 품용수치가 금형으로 흘러 들어가는데 속도는 최대속도	38.5~64.8	38~65
Max_Screw_RPM	제품을 제작하는 속도	30.3~31.2	30~32
Average_Screw_RPM	사출을 위한 스트류의 평균속도	29.2~293.9	29~294
Max_Injection_Pressure	배럴에 저항되어 있는 품용수치가 금형으로 흘러 들어가는데 가해지는 최대압력	140.7~169.1	140~167
Max_Switch_Over_Pressure	사출에서 보입(충진)될 수가 일리지않게 압력을 준다으로 변환되는 압력	128.4~146.7	128~147
Max_Bck_Pressure	수치가 계량되는 스트류의 위치나마는 현상	21.9~75.2	21~76
Average_Bck_Pressure	수치가 계량되는 스트류가 일어나는 현상을 저지하기 위한 평균압력	13.4~90.8	13~91
Barrel_Temperature_1	244.7~277.5	244~278	
Barrel_Temperature_2	249~276.5	248~277	
Barrel_Temperature_3	244.7~277.5	244~277	
Barrel_Temperature_4	244.8~272.4	244~273	
Barrel_Temperature_5	239.7~256.1	239~257	
Barrel_Temperature_6	224.6~230.7	224~231	
Barrel_Temperature_7	0~0	0~0	

<data set>

조건(a)~(w)

```
# 조건(t)
t_lower_rng = dt['Barrel_Temperature_4'] >= 244
t_upper_rng = dt['Barrel_Temperature_4'] <= 273

# 조건(u)
u_lower_rng = dt['Barrel_Temperature_5'] >= 239
u_upper_rng = dt['Barrel_Temperature_5'] <= 257

# 조건(v)
v_lower_rng = dt['Barrel_Temperature_6'] >= 224
v_upper_rng = dt['Barrel_Temperature_6'] <= 231

# 조건(w)
w_lower_rng = dt['Barrel_Temperature_7'] >= 0
w_upper_rng = dt['Barrel_Temperature_7'] <= 0

vald_dt = dt[a_lower_rng & a_upper_rng & b_lower_rng & b_upper_rng
& c_lower_rng & c_upper_rng & d_lower_rng & d_upper_rng
& e_lower_rng & e_upper_rng & f_lower_rng & f_upper_rng
& g_lower_rng & g_upper_rng & h_lower_rng & h_upper_rng
& i_lower_rng & i_upper_rng & j_lower_rng & j_upper_rng
& k_lower_rng & k_upper_rng & l_lower_rng & l_upper_rng
& m_lower_rng & m_upper_rng & n_lower_rng & n_upper_rng
& o_lower_rng & o_upper_rng & p_lower_rng & p_upper_rng
& q_lower_rng & q_upper_rng & r_lower_rng & r_upper_rng
& s_lower_rng & s_upper_rng & t_lower_rng & t_upper_rng
& u_lower_rng & u_upper_rng & v_lower_rng & v_upper_rng
& w_lower_rng & w_upper_rng]
```

[그림 39] : 분석 도구를 활용하여 수집된 데이터 저장하기

: 유효 범위를 충족하는 데이터들은 3,957개로 약 18개의 데이터가 수집 범위를 벗어났다는 것을 알 수 있다.

```
len(vald_dt)#결과는 아래에서 확인 가능하다.
```

3957

[코드 66] : 데이터 개수 유효성

② 데이터가 형식에 맞는가

: 'TimeStamp'와 'PART\_FACT\_PLAN\_DATE' 컬럼의 타입을 datetime으로 변경할 때 errors='coerce' 옵션을 추가한다. errors='coerce' 옵션은 올바르지 않은 날짜형식을 null 값으로 채워주는 옵션이다.

```

vald_dt['TimeStamp']=pd.to_datetime(vald_dt['TimeStamp'],format='%Y-%m-%dT%H:%M:%S',erro
rs='coerce')
vald_dt['PART_FACT_PLAN_DATE']=pd.to_datetime(vald_dt['PART_FACT_PLAN_DATE'],format='%Y-
%m-%d 오전 12:00:00',errors='coerce')

```

[코드 67] : 분석 도구를 활용하여 데이터 형식 맞춰주기

: 'TimeStamp'와 'PART\_FACT\_PLAN\_DATE'가 null인 경우를 확인해보면 모든 데이터의 형식이 올바르다는 것을 알 수 있다.

```
vald_dt[vald_dt['PART_FACT_PLAN_DATE'].isnull()] #결과는 아래에서 확인 가능하다.
```

```
_id TimeStamp PART_FACT_PLAN_DATE PART_FACT_SERIAL Shot_Number PART_NO PART_NAME
EQUIP_CD EQUIP_NAME
```

```
vald_dt[vald_dt['TimeStamp'].isnull()] #결과는 아래에서 확인 가능하다.
```

```
_id TimeStamp PART_FACT_PLAN_DATE PART_FACT_SERIAL Shot_Number PART_NO PART_NAME
EQUIP_CD EQUIP_NAME
```

[코드 68] : 분석 도구를 활용하여 데이터의 형식 확인하기

③ 수집된 날짜 안에 들어가 있는가. 등을 검증하는 것이다.

: 2020.10.16 ~ 2020.11.03에 수집된 데이터이기 때문에, 데이터의 날짜가 이 기간을 벗어나지 않았는지 확인한다.

```

import datetime
d0= pd.Timestamp(datetime.date(2020,10,16)) #date 객체1
d1= pd.Timestamp(datetime.date(2020,11,4)) #date 객체2
con1=vald_dt['TimeStamp']>=d0
con2=vald_dt['TimeStamp']<=d1
vald_dt=vald_dt[con1&con2]

```

[코드 69] : 분석 도구를 활용하여 데이터 수집 기간 확인하기

▶ 따라서 3가지 경우를 모두 만족하는 데이터는 vald\_dt에 저장되게 되고, 최종적으로 완성된 데이터를 이용하여 유효성 품질지수를 구한다.

- 유효성 품질지수 = (유효성만족 데이터수/전체데이터 수) × 100

```

vald_len=len(vald_dt)
print("유효성 지수 : %.2f%%"%(vald_len/len(dt)*100)) #결과는 아래에서 확인 가능하다.

```

유효성 지수 : 99.55 %

[코드 70] : 분석 도구를 활용하여 유효성 품질지수 최종 확인하기

- 일관성 품질지수 = (일관성만족 데이터수/전체데이터 수) × 100  
여기 사출성형 데이터는 다른 데잍블을 참조하는 컬럼이 없으므로 일관성을 판단하지 않는다.
- 정확성 품질지수 = (1-(정확성 위배데이터수/전체데이터수)) × 100
  - : 데이터를 확인해보면 'PassOrFail' 컬럼의 값에 따라서 'Reason' 컬럼에 영향을 주는 것을 알 수 있다. 'PassOrFail'이 Y면 'Reason'은 None이고, N이면 None이 아닌 불량유형의 값이 들어가야 하므로, 두 가지 경우를 만족하는지 확인한다.
  - : 이 데이터에서는 두 컬럼의 값이 잘못 기입된 값이 없으므로 정확성 품질지수는 100%이다.
  - : 라벨링이 되어있지 않은 데이터는 두 컬럼이 존재하지 않기 때문에 정확성을 위배하는지 확인할 필요가 없다.

dt[['PassOrFail', 'Reason']]		
	PassOrFail	Reason
45	Y	None
46	Y	None
47	Y	None
48	N	가스
49	Y	None

(1) PassOrFail=='Y' 일 때, Reason=='None'  
`a = dt[dt['PassOrFail']=='Y']  
b = dt[dt['Reason']=='None']  
print(len(a)-len(b))`

(2) PassOrFail=='N' 일 때, Reason!='None'  
`a = dt[dt['PassOrFail']=='N']  
b = dt[dt['Reason']!=None]  
print(len(a)-len(b))`

[그림 40] : 분석 도구를 활용하여 Pass or Fail 확인하기

- ▶ 최종적으로 구한 3가지 데이터 품질지수에 적절한 가중치를 부여하여 품질지수를 구한다.

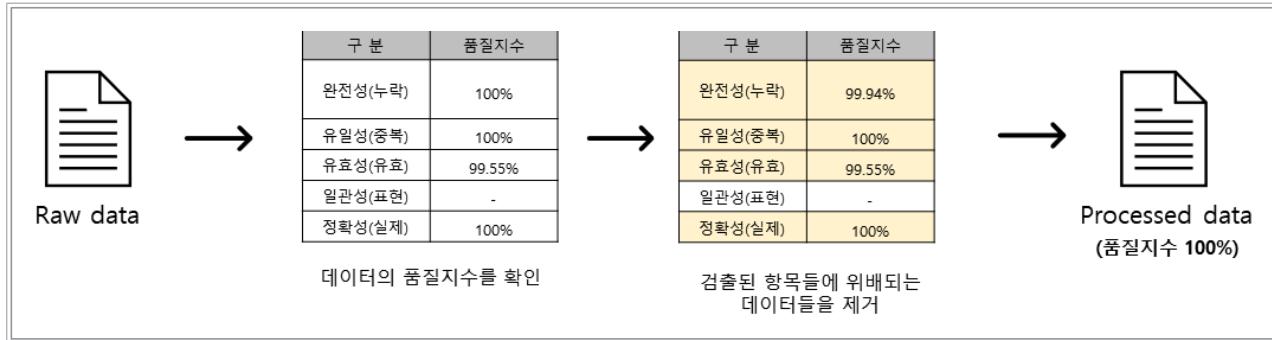
- 가중치지수 = 품질지수 × 가중치
- 데이터 품질지수 =  $\sum$  가중치지수

구 분	품질지수	가중치	가중치지수	오류율
완전성(누락)	100%	70%	70%	0%
유일성(중복)	100%	10%	10%	0%
유효성(유효)	99.55%	10%	9.96%	0.45%
일관성(표현)	-	-	-	-
정확성(실제)	100%	10%	10%	0%
품질 지수	99.89%	100%	99.96%	0.04%

[그림 41] : 데이터 품질지수

## • 데이터 전처리 방법

- 개발언어 : 파이썬(Python)
- 개발환경 : 주피터랩(Jupyterlab)



[그림 42] : 데이터 전처리 과정 도식화

- 앞에서 구한 데이터 품질지수를 높이기 위해 유효성을 위배하는 데이터들을 삭제하여 품질 전처리를 진행한다.

① 데이터를 불러온 뒤, 변수에 저장한다.

```
file_name = r'CN7.csv'
dt = pd.read_csv(file_name, low_memory=False, index_col=None, encoding='utf-8')
```

② 유효성 품질지수를 높이기 위해, 유효범위에 들어가는 데이터들만 남겨준다.

The screenshot shows a Jupyter Notebook cell with the following code:

```
# (1) 데이터가 유효 범위 안에 들어가 있는가?
# 조건(a)
a_lower_rng = dt['Injection_Time'] >= 9
a_upper_rng = dt['Injection_Time'] <= 14

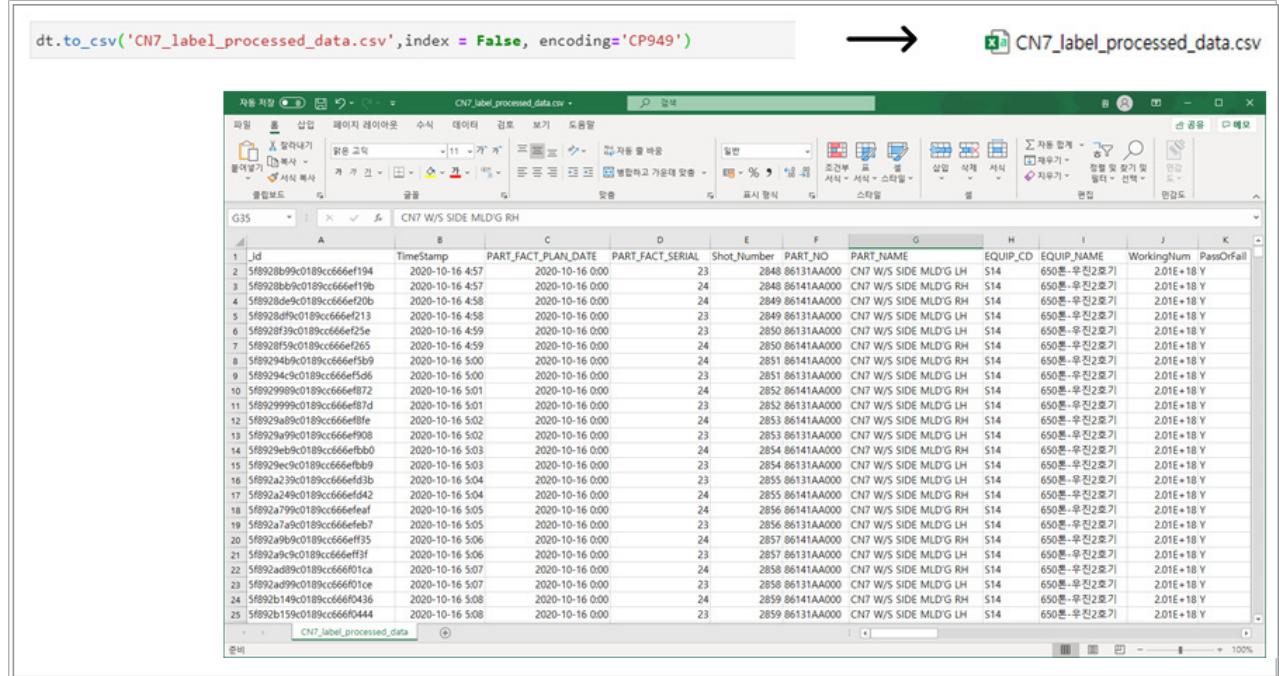
# 조건(b)
b_lower_rng = dt['Filling_Time'] >= 3
b_upper_rng = dt['Filling_Time'] <= 9

# 조건(c)
w_lower_rng = dt['Barrel_Temperature_7'] >= 0
w_upper_rng = dt['Barrel_Temperature_7'] <= 0

dt = dt[a_lower_rng & a_upper_rng & b_lower_rng & b_upper_rng
        & c_lower_rng & c_upper_rng & d_lower_rng & d_upper_rng
        & e_lower_rng & e_upper_rng & f_lower_rng & f_upper_rng
        & g_lower_rng & g_upper_rng & h_lower_rng & h_upper_rng
        & i_lower_rng & i_upper_rng & j_lower_rng & j_upper_rng
        & k_lower_rng & k_upper_rng & l_lower_rng & l_upper_rng
        & m_lower_rng & m_upper_rng & n_lower_rng & n_upper_rng
        & o_lower_rng & o_upper_rng & p_lower_rng & p_upper_rng
        & q_lower_rng & q_upper_rng & r_lower_rng & r_upper_rng
        & s_lower_rng & s_upper_rng & t_lower_rng & t_upper_rng
        & u_lower_rng & u_upper_rng & v_lower_rng & v_upper_rng
        & w_lower_rng & w_upper_rng]
```

To the right, a pandas DataFrame is shown with columns: Id, TimeStamp, PART\_FACT\_PLAN\_DATE, PART\_FACT\_SERIAL, Shot\_Number, PART\_NO, PART\_NAME, EQUIP\_CD, and E. The data includes rows 0 through 3974, with row 3957 noted as having 49 columns.

③ 모든 과정이 끝나면 데이터 품질 지수가 100%인 데이터가 완성되고, `to_csv()`함수를 이용하여 데이터를 저장한다. 데이터에 한글이 포함되어 있으므로 `encoding='CP949'` 조건을 추가해준다.



## • 데이터 전처리 완료

- `read_csv()`를 이용하여 전처리가 완료된 데이터를 읽어온다.

```
file_name = r'CN7_label_processed_data.csv'
prod_dt = pd.read_csv(file_name, low_memory=False, index_col=None, encoding='CP949')
```

- 가공된 데이터는 총 3,957개로 데이터로 약 0.45%(18개)가 outlier로 검출되었다.

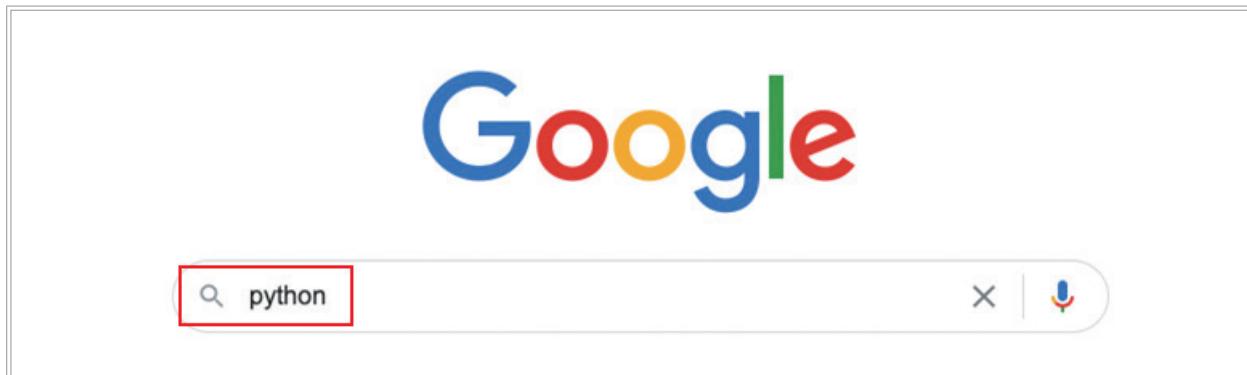
	_Id	TimeStamp	PART_FACT_PLAN_DATE	PART_FACT_SERIAL	Shot_Number	PART_NO	PART_NAME	EQUIP_CD	EQUIP_NAME	WorkingNum	Mold_Temperature_4	Mold_Temperature_5
0	5f8928b99c0189cc666ef194	2020-10-16 04:57:47	2020-10-16 00:00:00	23	2848	86131AA000	CN7 W/S SIDE MLD'G LH	\$14	650톤-우진2호기	2.010160e+18	...	27.500000
1	5f8928bb9c0189cc666ef19b	2020-10-16 04:57:47	2020-10-16 00:00:00	24	2848	86141AA000	CN7 W/S SIDE MLD'G RH	\$14	650톤-우진2호기	2.010160e+18	...	27.500000
2	5f8928de9c0189cc666ef20b	2020-10-16 04:58:48	2020-10-16 00:00:00	24	2849	86141AA000	CN7 W/S SIDE MLD'G RH	\$14	650톤-우진2호기	2.010160e+18	...	27.600000
3	5f8928de9c0189cc666ef213	2020-10-16 04:58:48	2020-10-16 00:00:00	23	2849	86141AA000	CN7 W/S SIDE MLD'G LH	\$14	650톤-우진2호기	2.010160e+18	...	27.600000
3954	5fa112879c0189cc66dabe50	2020-11-03 08:18:37	2020-11-03 00:00:00	6	7496	86131AA000	CN7 W/S SIDE MLD'G LH	\$14	650톤-우진2호기	2.011030e+18	...	22.400000
3955	5fa112bb9c0189cc66dac22a	2020-11-03 08:19:35	2020-11-03 00:00:00	6	7497	86131AA000	CN7 W/S SIDE MLD'G LH	\$14	650톤-우진2호기	2.011030e+18	...	22.299999
3956	5fa112bc9c0189cc66dac23a	2020-11-03 08:19:35	2020-11-03 00:00:00	7	7497	86141AA000	CN7 W/S SIDE MLD'G RH	\$14	650톤-우진2호기	2.011030e+18	...	22.299999



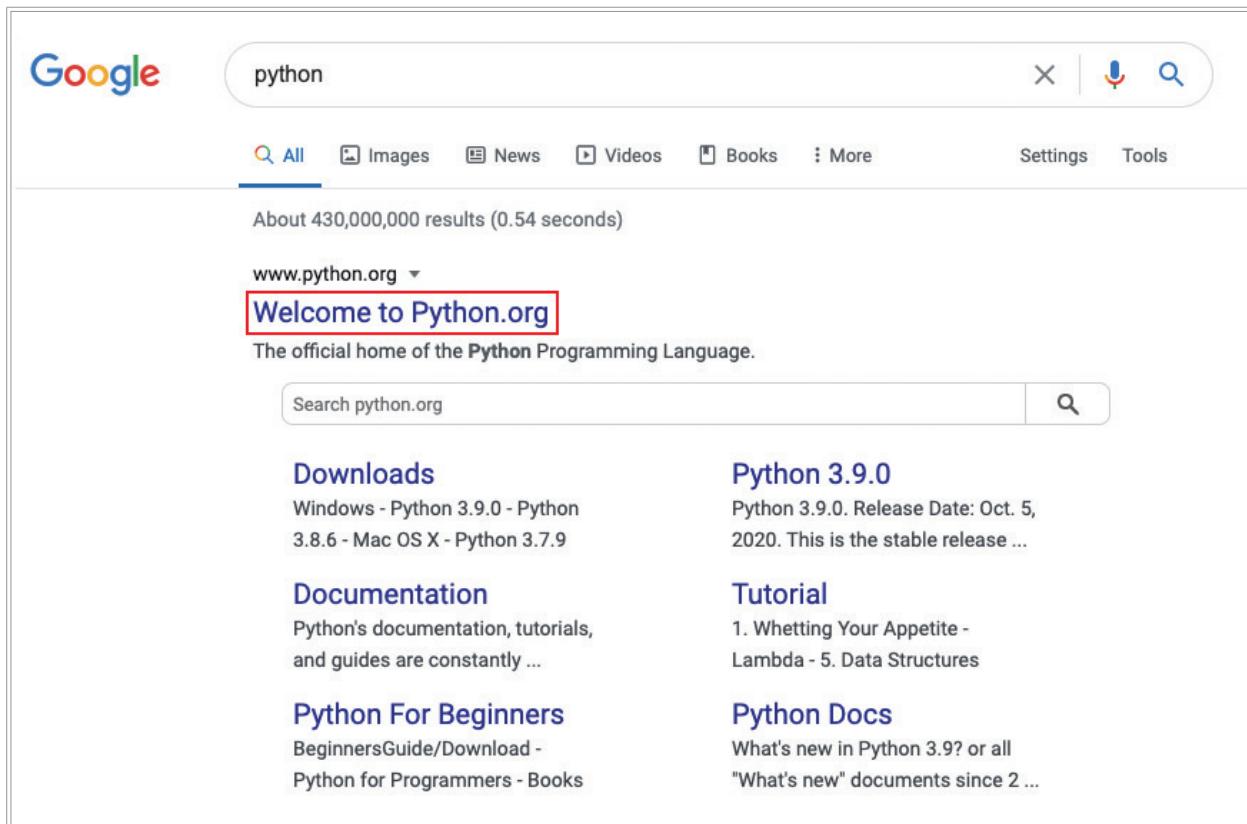
### 1. 파이썬(python) 설치

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

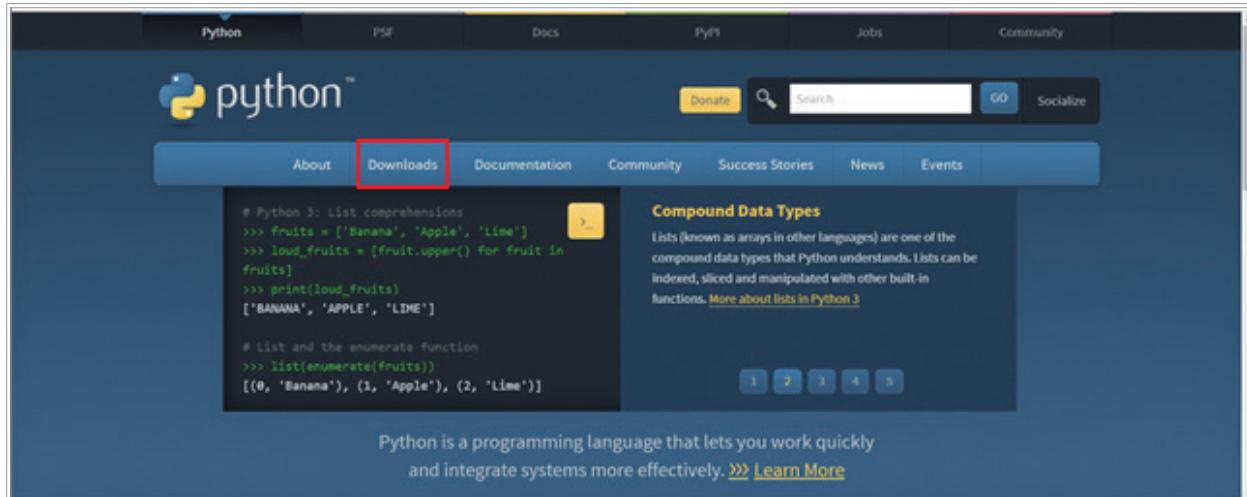
① google.com 등의 검색 엔진에 ‘python’을 검색



② 제일 처음에 보이는 ‘Welcome to python.org’를 클릭

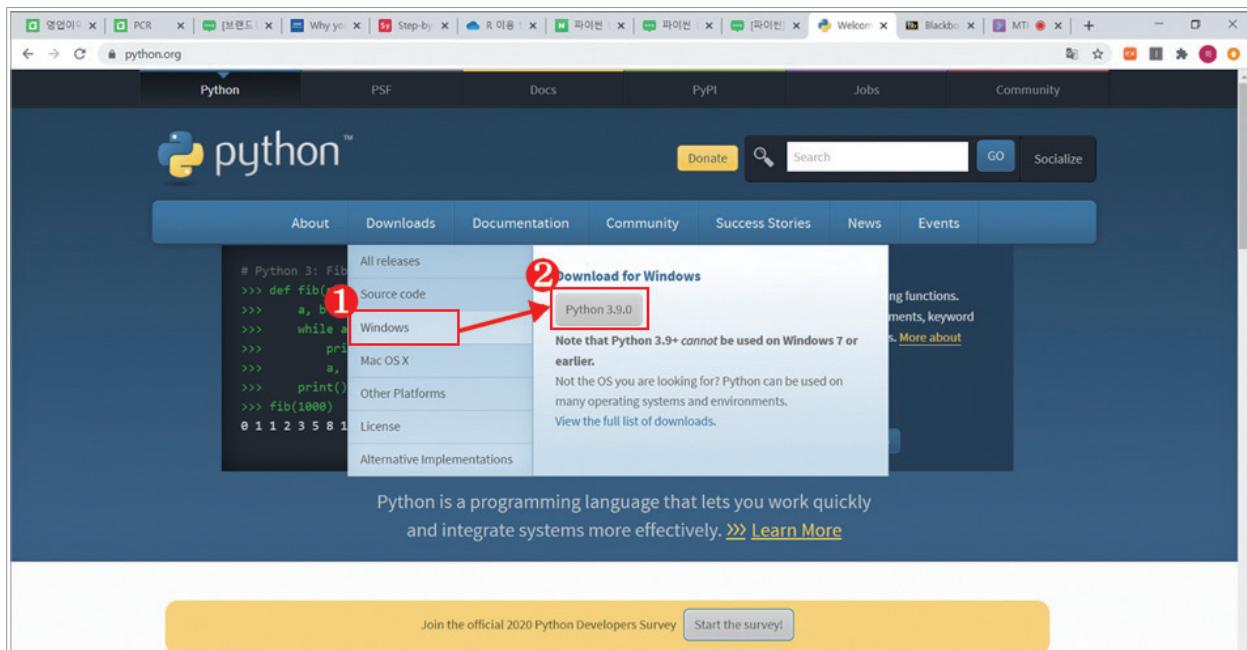


③ 클릭해서 보이는 페이지 정면의, 왼쪽 2번째 'Downloads' 클릭

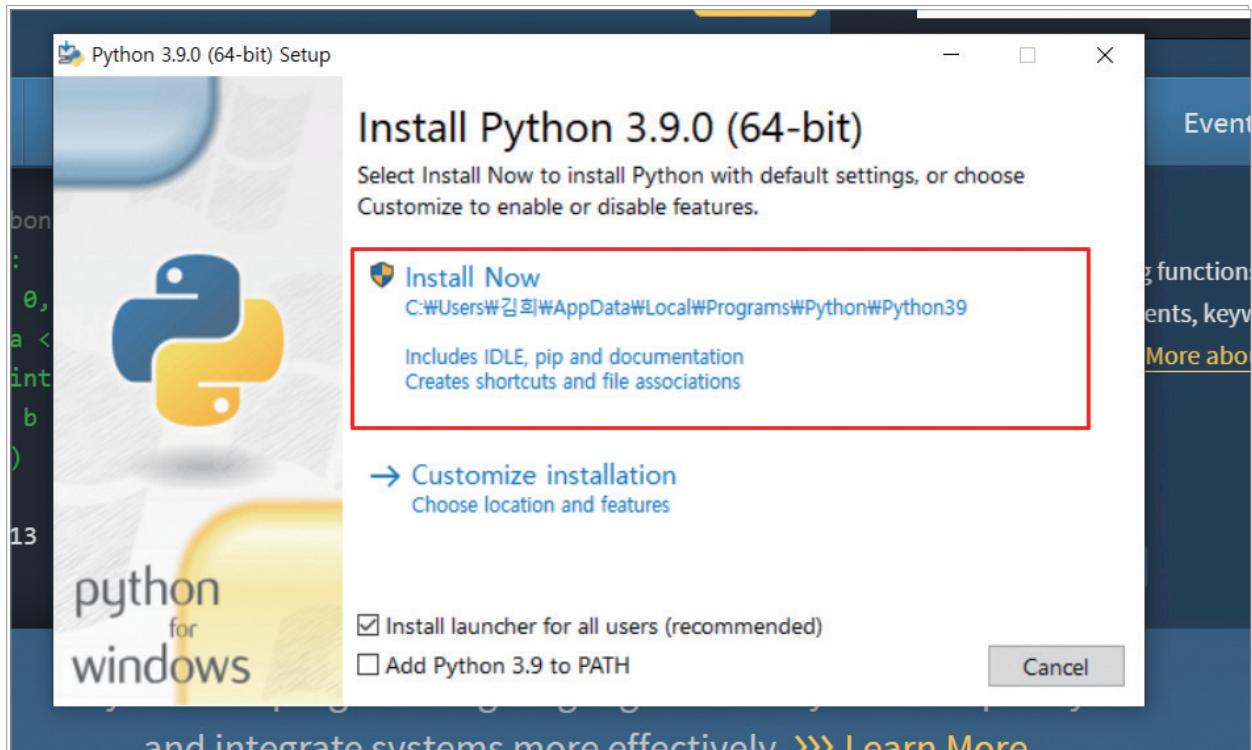


④ 위에서 3번째, Windows 탭을 선택한 후, python3.9.0 다운로드

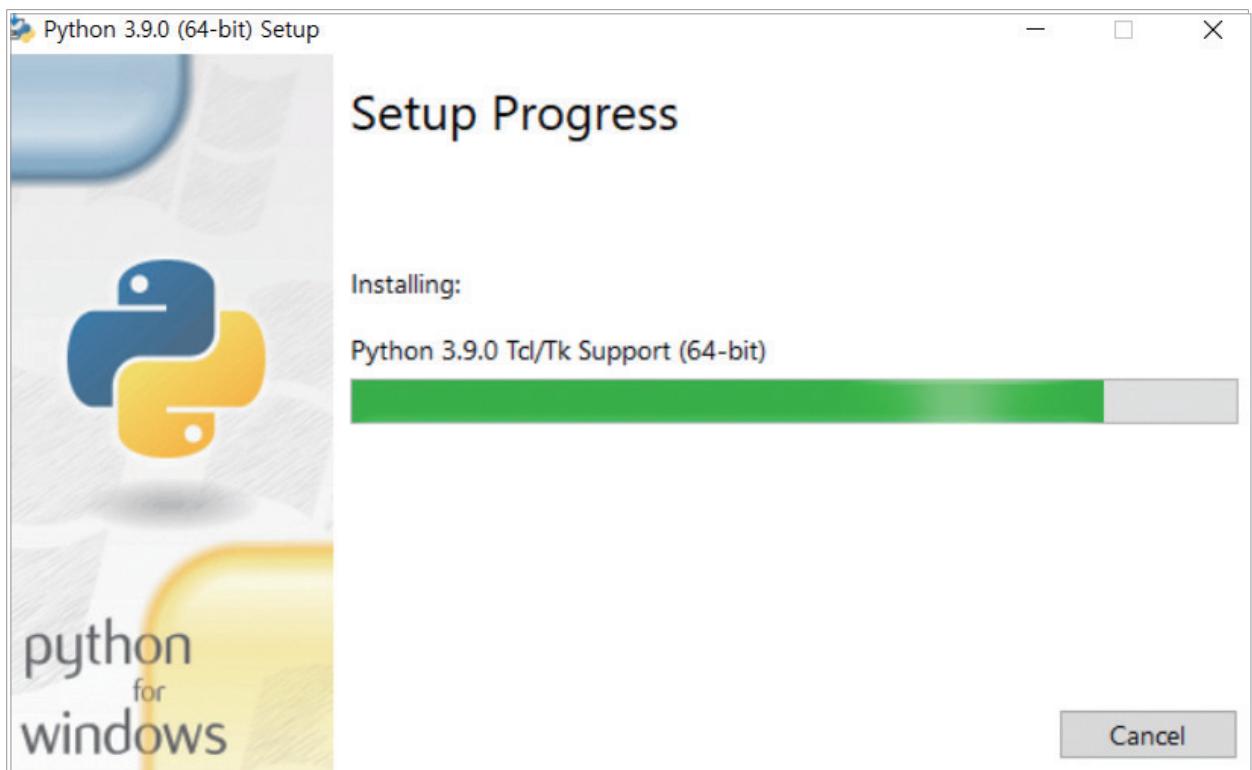
(python3.9.0은 숫자가 업데이트 될 수 있습니다)



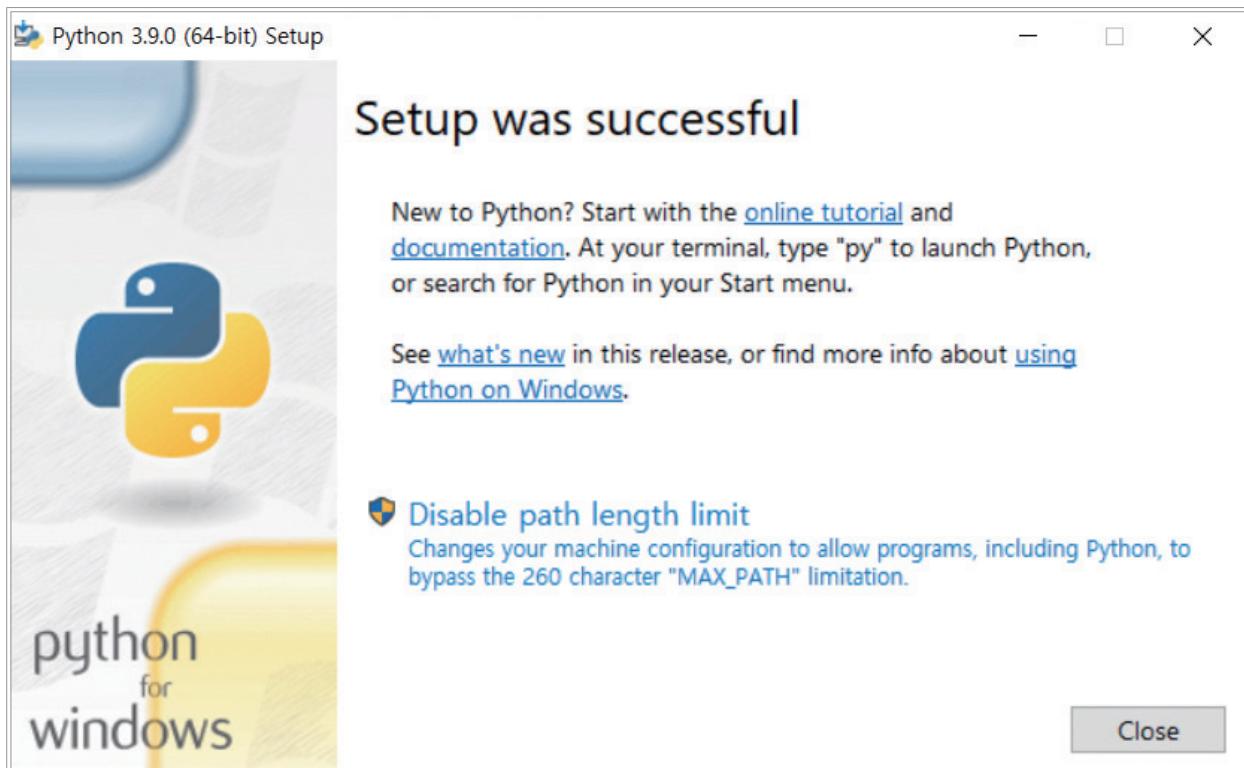
⑤ 아래와 같은 설치창이 뜨면, 'Install Now'를 클릭



⑥ 아래와 같은 설치 진행창이 완료가 될 때까지 유지



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

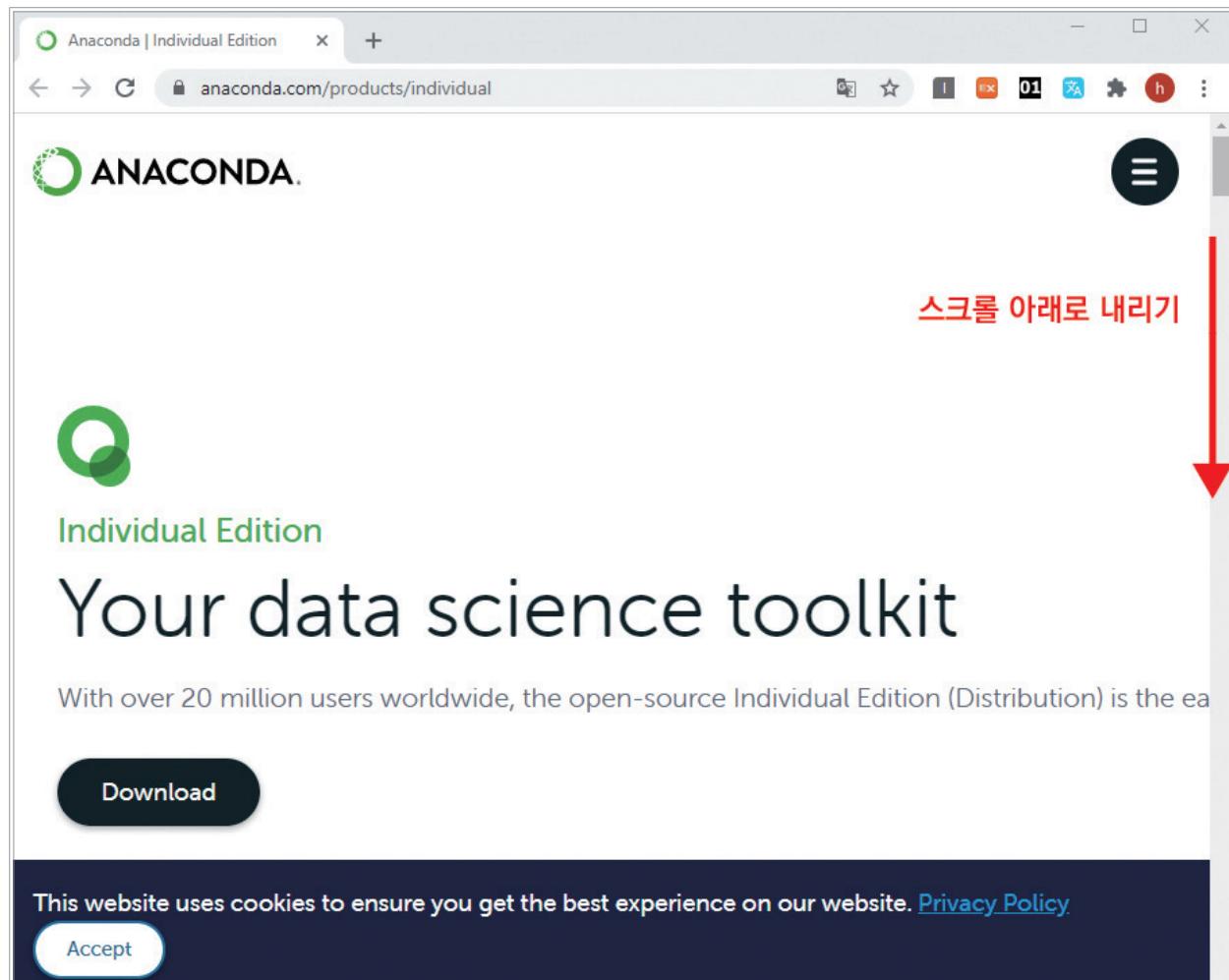


## 2. 아나콘다(anaconda) 설치



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

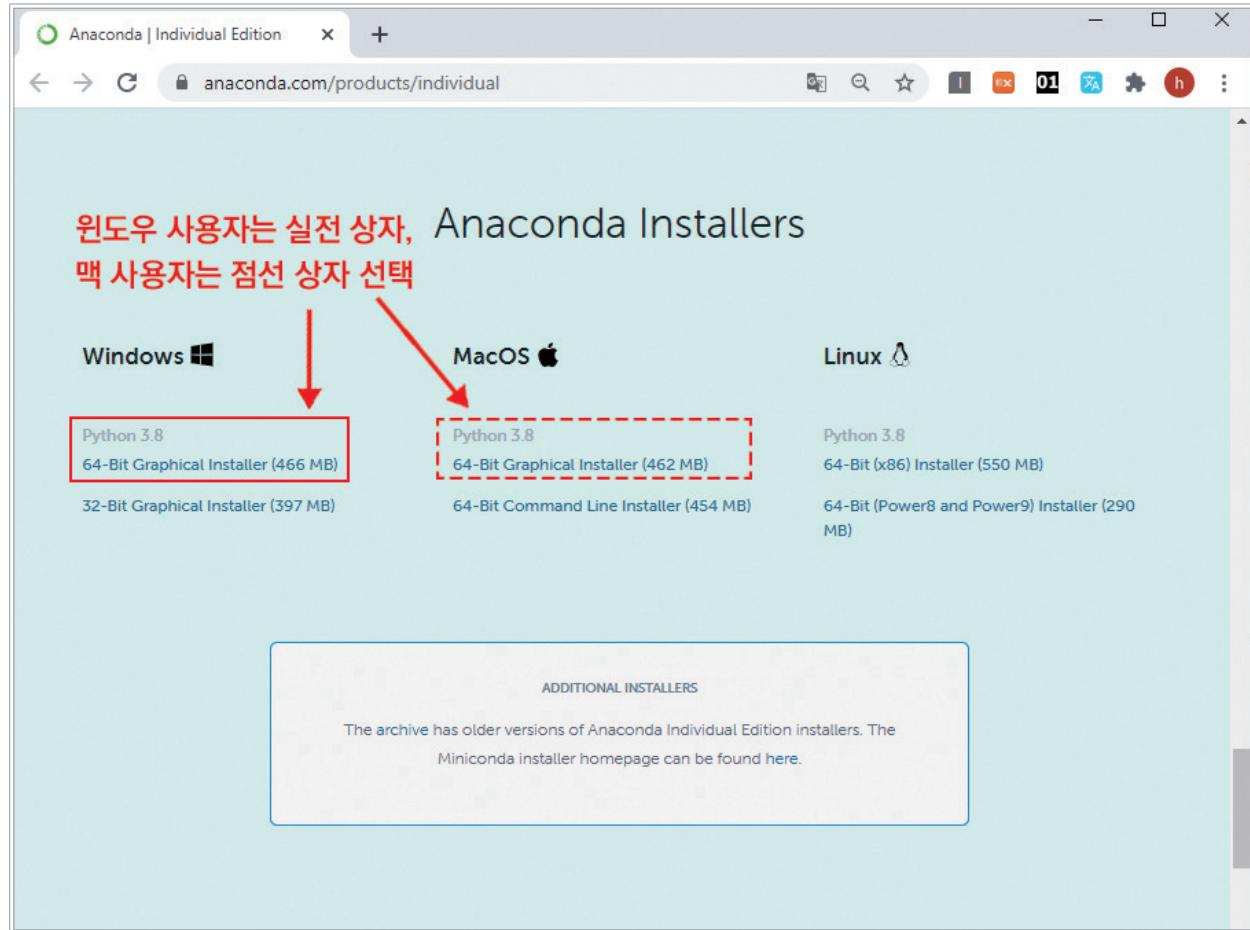
- ① <https://www.anaconda.com/distribution/> 로 접속 후 스크롤 내림



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

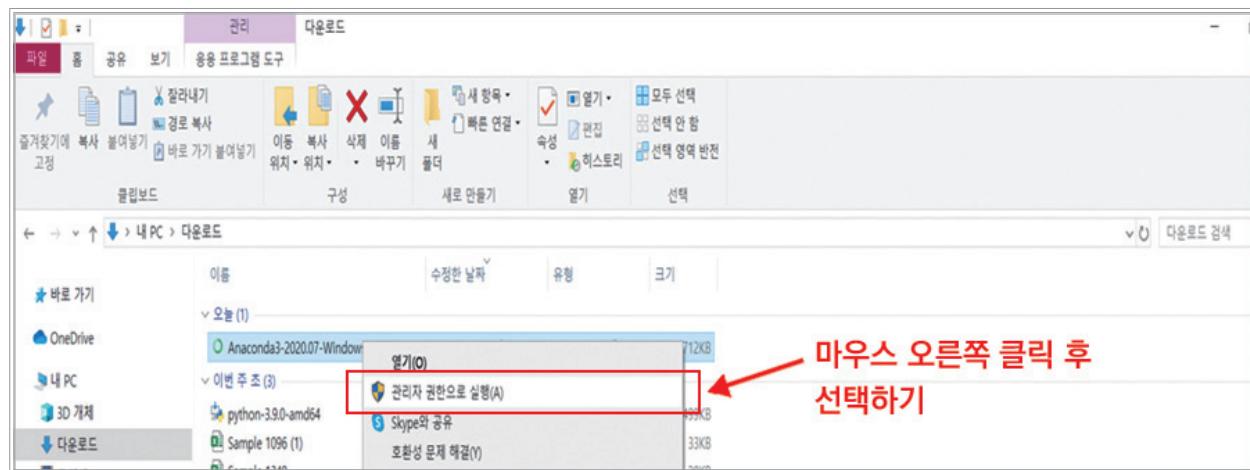
▶ Windows : 64-Bit Graphical Installer

▶ MacOS : 64-Bit Graphical Installer

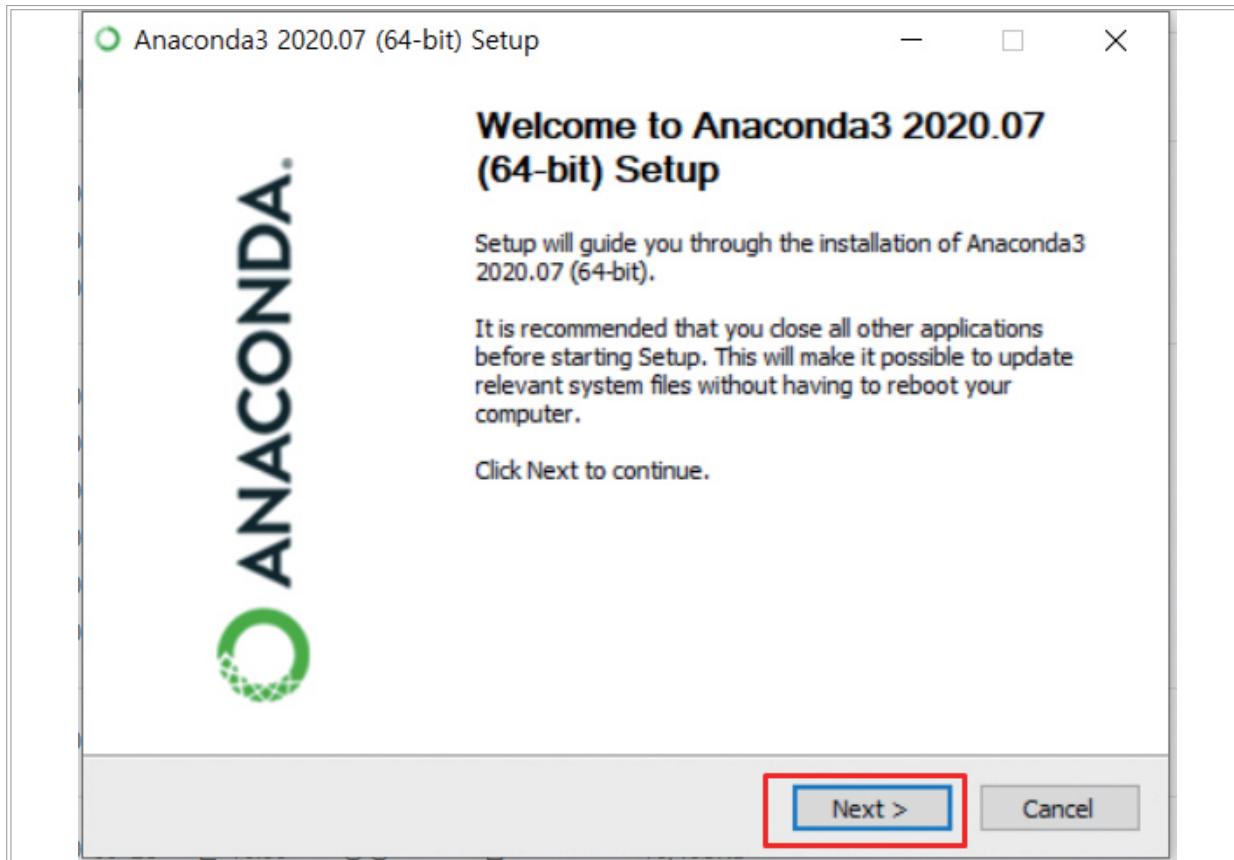


- ③ 다운을 받은 파일에 가서, 아나콘다 설치 파일 위에서, 마우스 오른쪽을 클릭한 후, 방패모양의 ‘관리자 권한으로 실행’ 선택

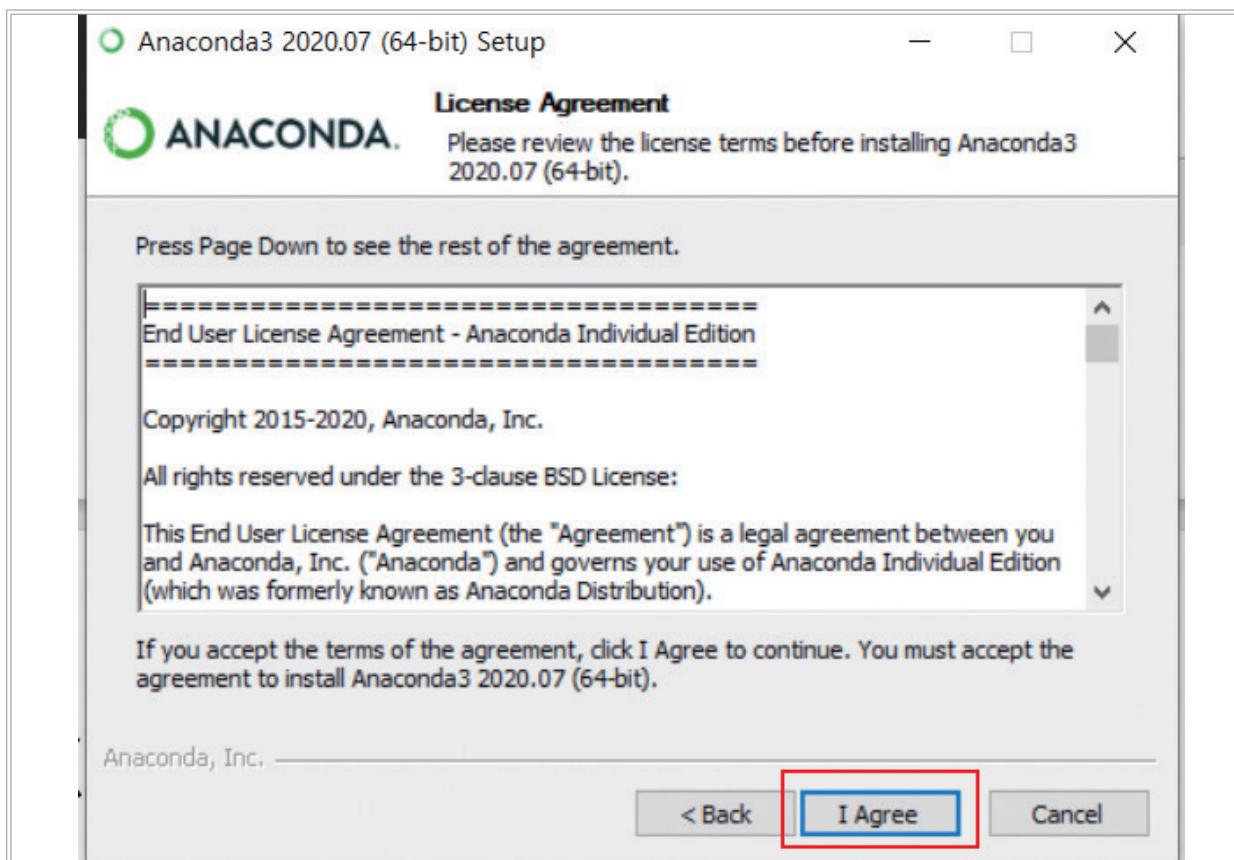
▶ (예) ‘다운로드’ 파일로 아나콘다를 다운 받은 경우



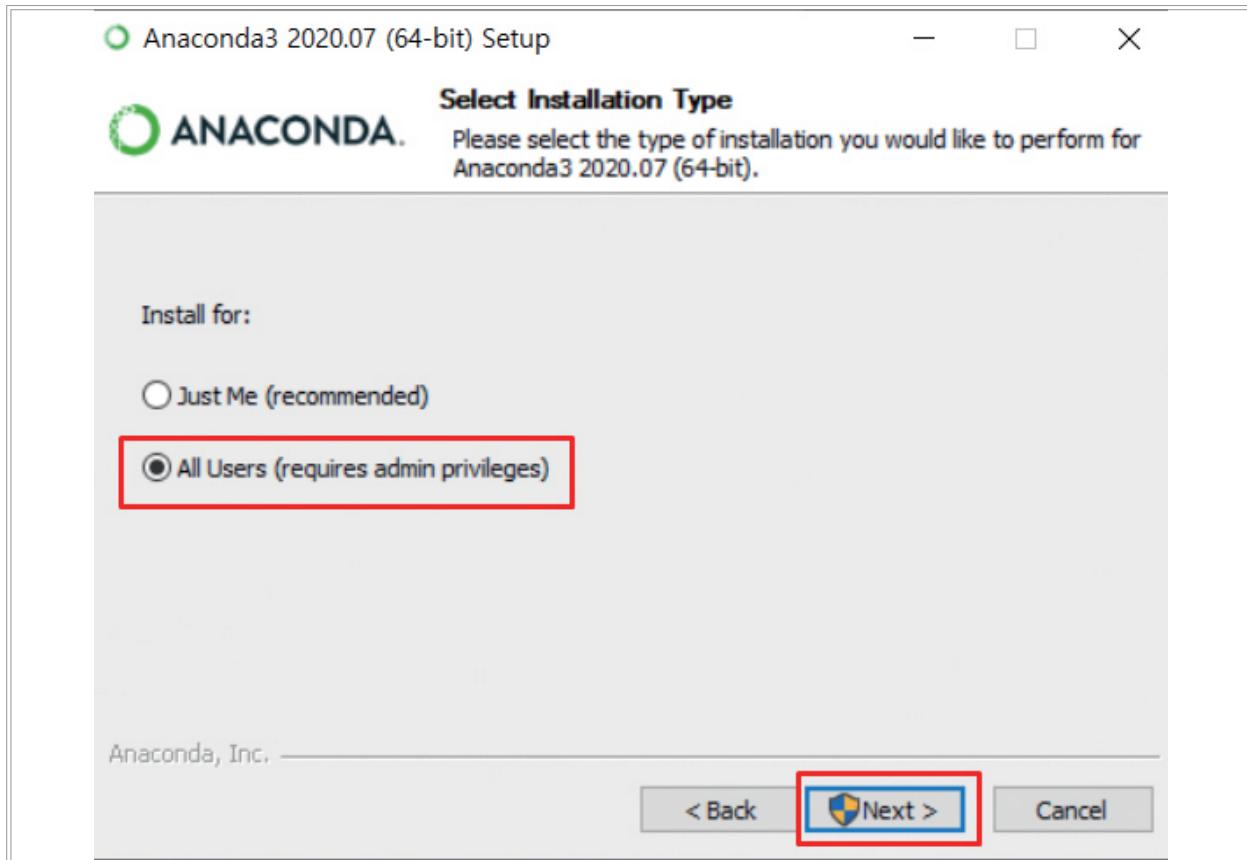
④ 파일을 실행 한 후, 'Next' 버튼을 클릭



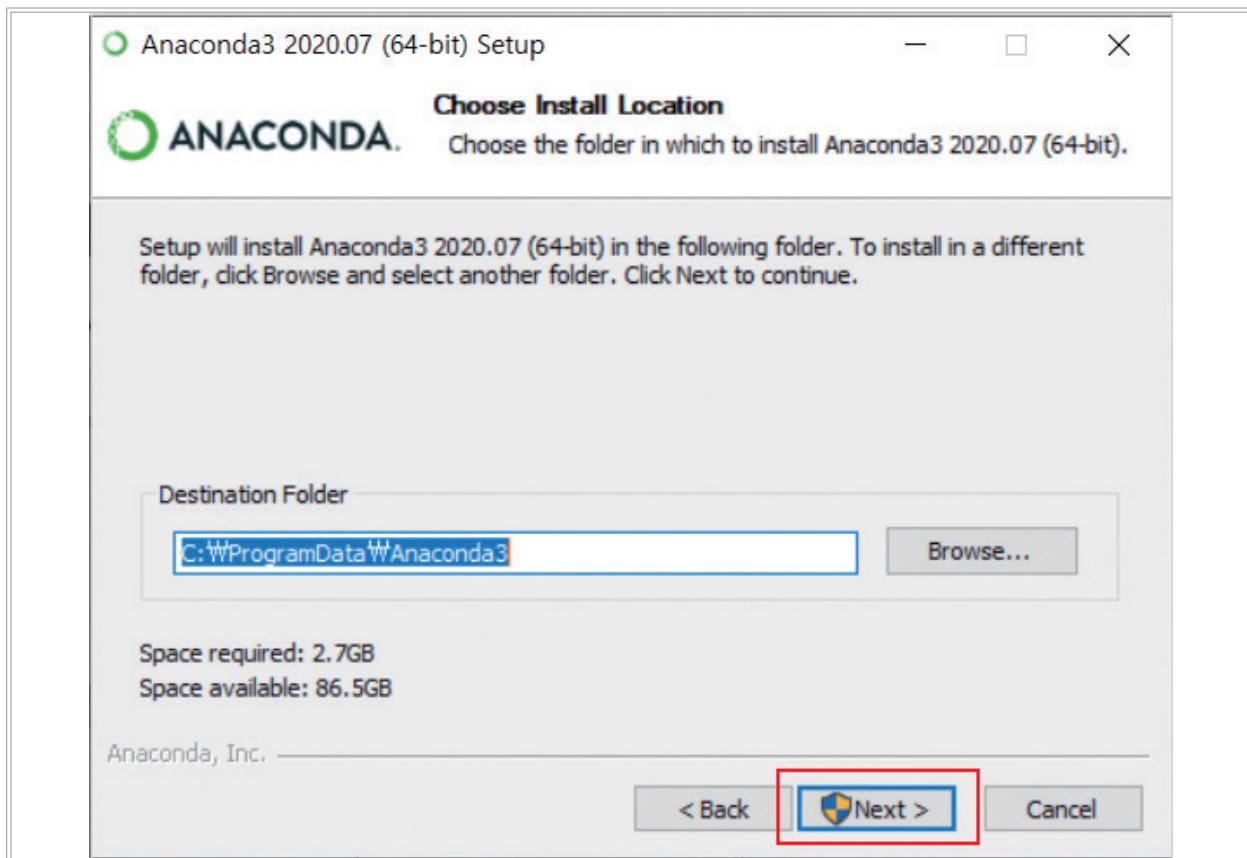
⑤ 다음 창이 나타나면 'I agree'를 선택



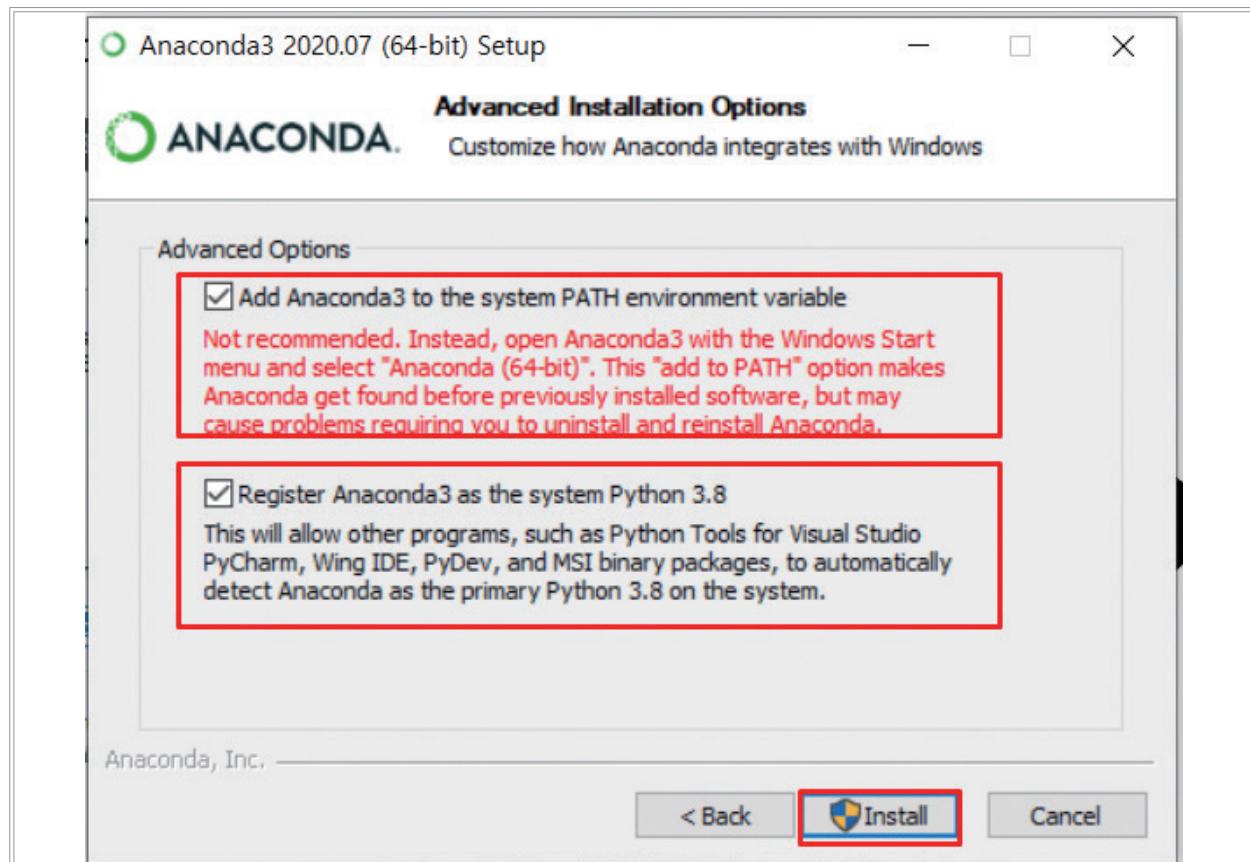
⑥ 셋팅 창이 뜨면 화면의 ‘All Users’를 선택 후 아래의 ‘Next’ 클릭



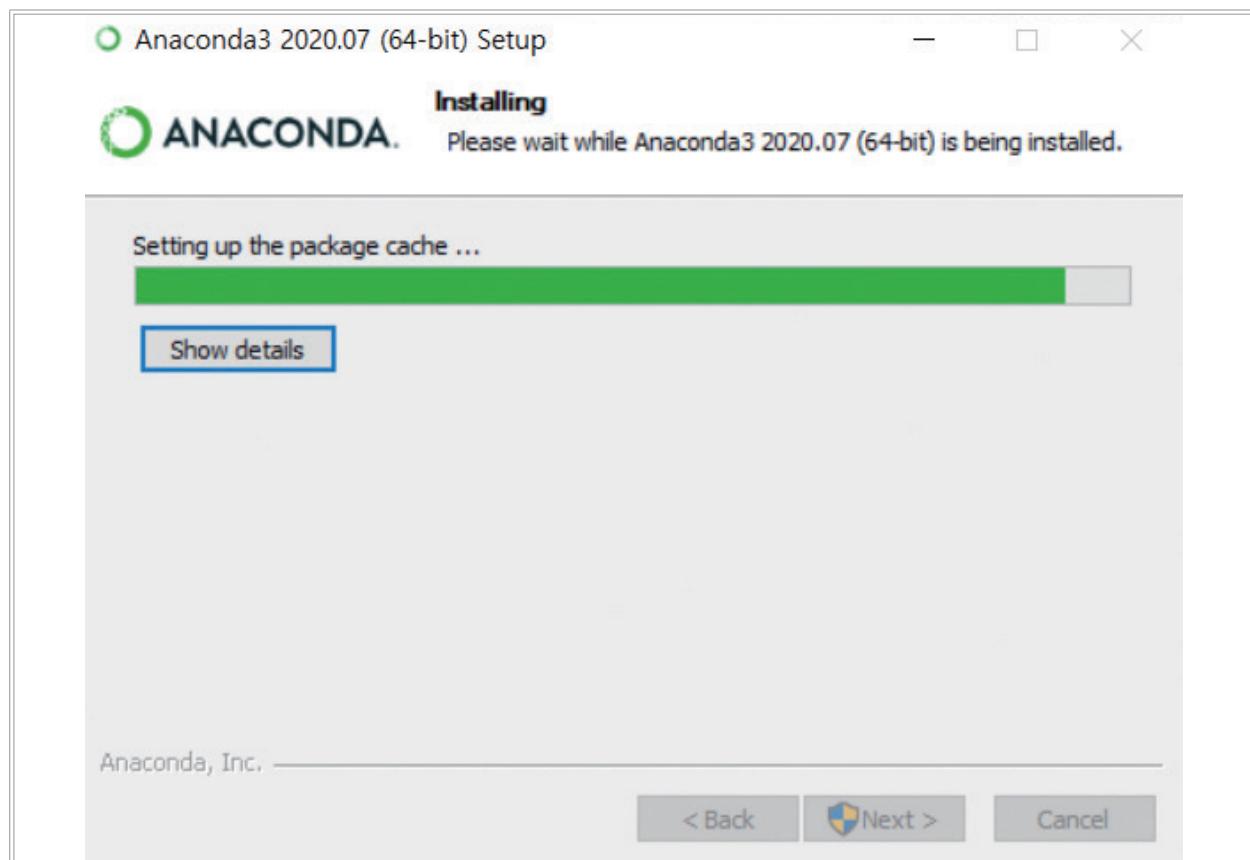
⑦ 다운로드 받을 경로를 물어보는 창이 뜨면, 아래의 ‘Next’ 클릭



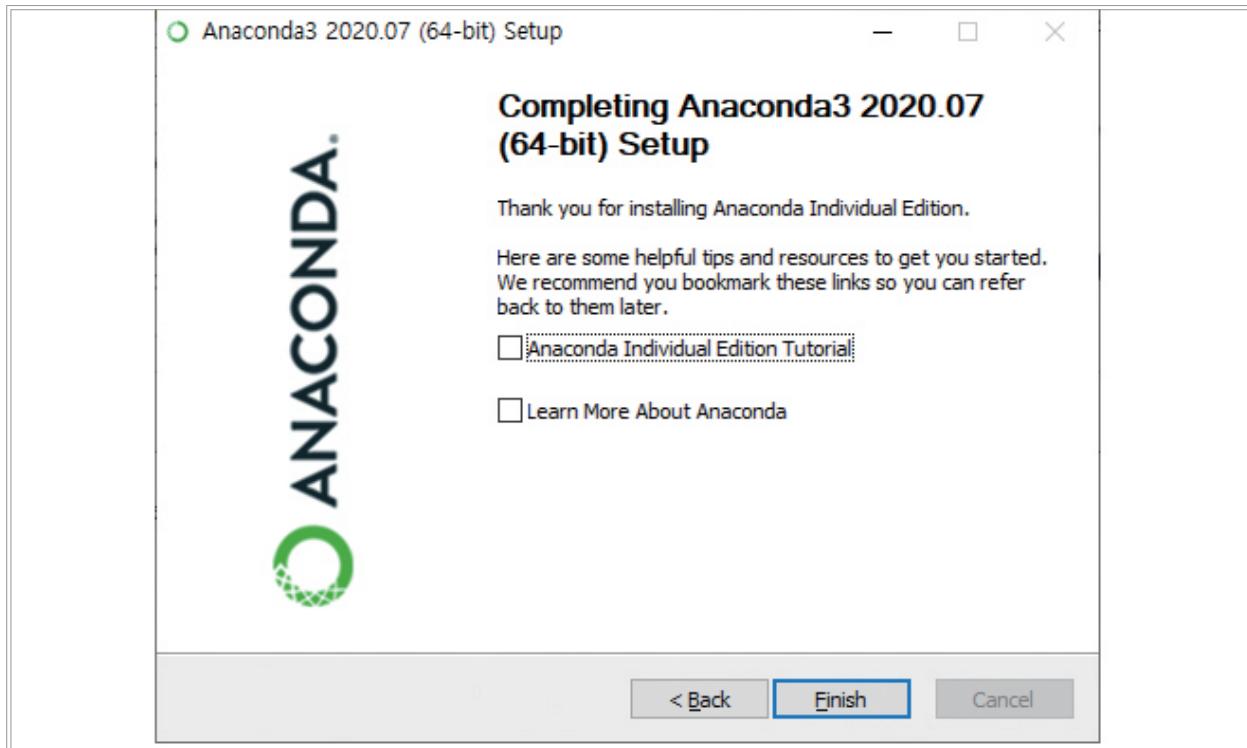
⑧ 고급 옵션 선택창이 뜨면, 아래와 같이 모두 선택 후, ‘Install’ 클릭



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

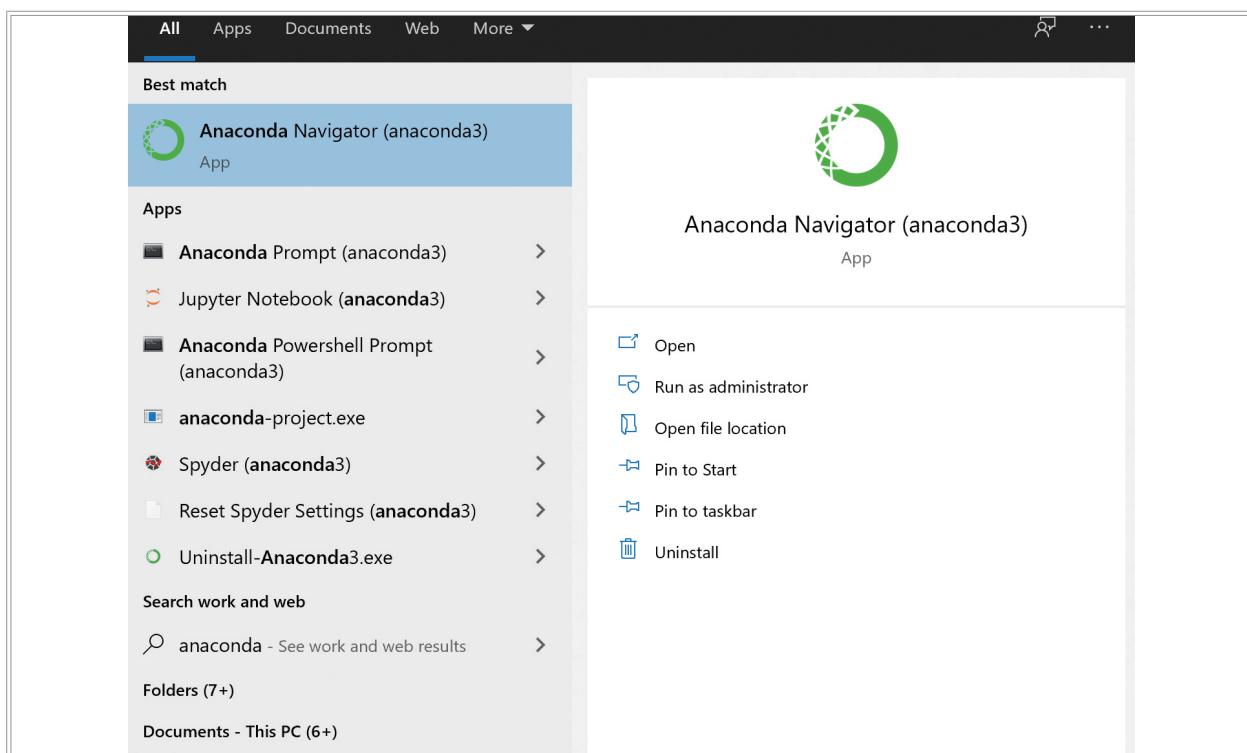


⑩ 마지막 화면에서, 모두 체크 해제한 후, ‘Finish’ 눌러 설치 완료



⑪ 설치 확인하기

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



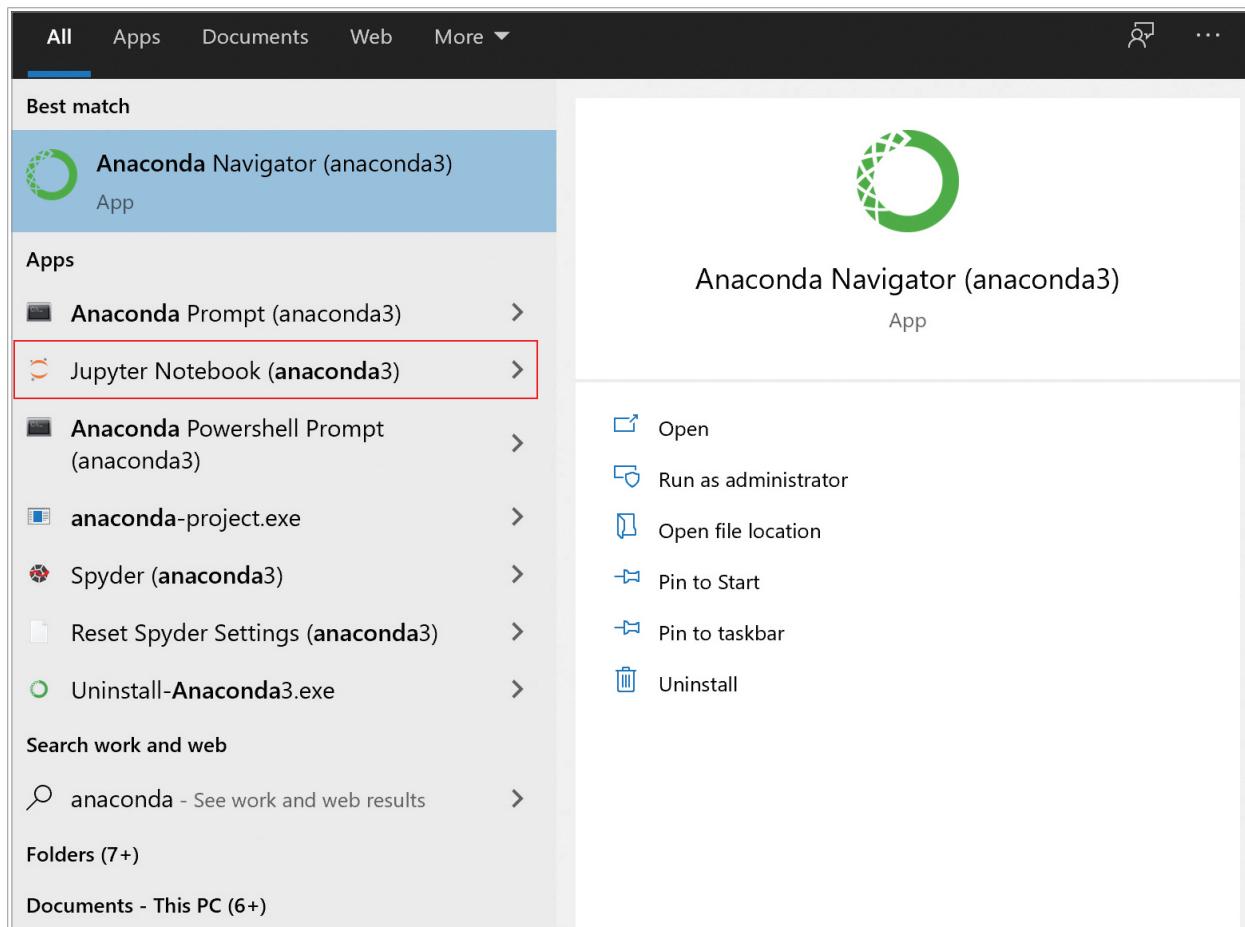
- Anaconda Navigator, Anaconda Prompt, Jupyter Notebook 등 다른 응용 프로그램들도 잘 깔려있는지 확인이 된다면, 설치 완료

### 3. 주피터 노트북 (Jupyter notebook) 실행



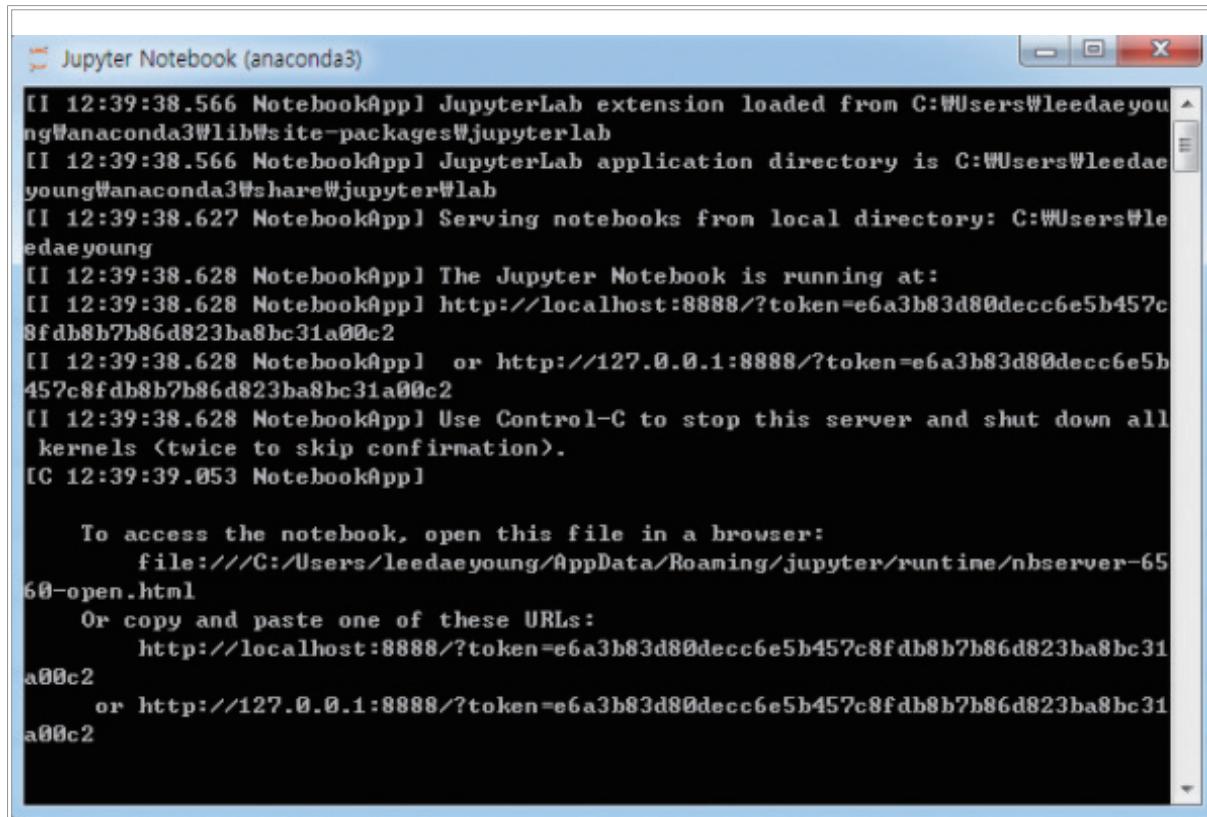
주피터 노트북이란, 실제로 코딩(분석 문장 작성)을 할 수 있는 도구이다. 쉽게 얘기하자면, 문서 도구로 마이크로소프트 사의 ‘Word’ 파일이나, 국내에서는 ‘한글’ 파일 등과 같은 도구라고 생각할 수 있다. 데이터 분석에 다양한 입력, 실행 도구가 있지만, 본 가이드북에서는 주피터 노트북을 활용하는 방법을 공유하기로 한다. [부록2]를 참고하여, Anaconda를 설치하였다면, 주피터 노트북(Jupyter notebook) 설치도 확인한다.

- ① 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.

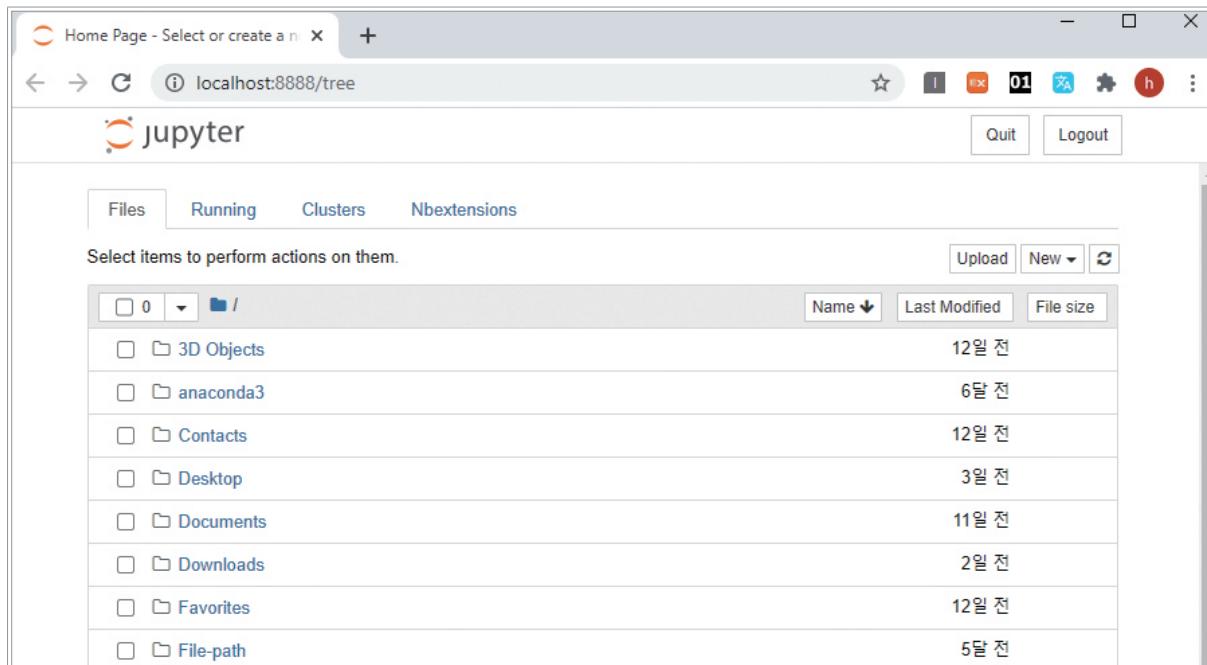
(1) 검은색 배경의 화면은, 주피터 노트북이 실행되는 환경에 대한 상태를 나타내주는 ‘상태 표시창’ 같은 곳. **분석을 하는 동안 종료하면 안된다.**



```
[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
```

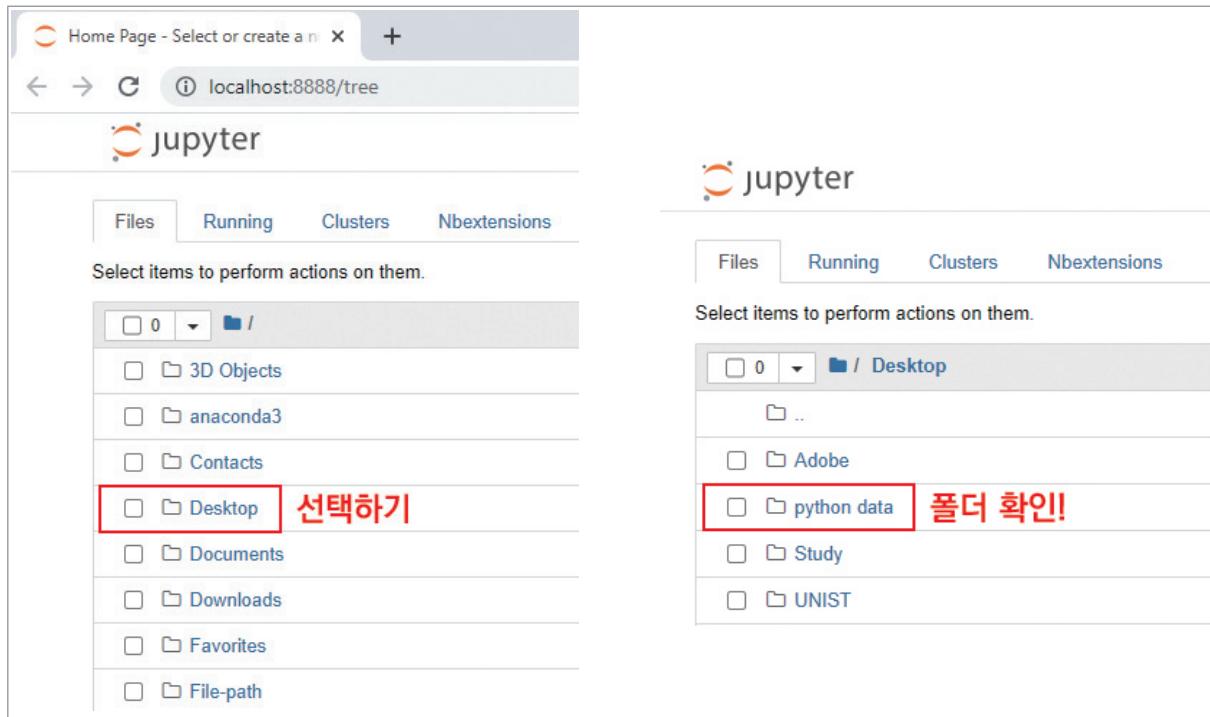
(2) 사용하는 인터넷 프로그램(크롬 / 인터넷 익스플로러)에 주피터 노트북이 열림. (다른 확장 프로그램 사용하고 싶다면, 기본 브라우저를 변경해주어야함). **이 창을 주로 사용.**



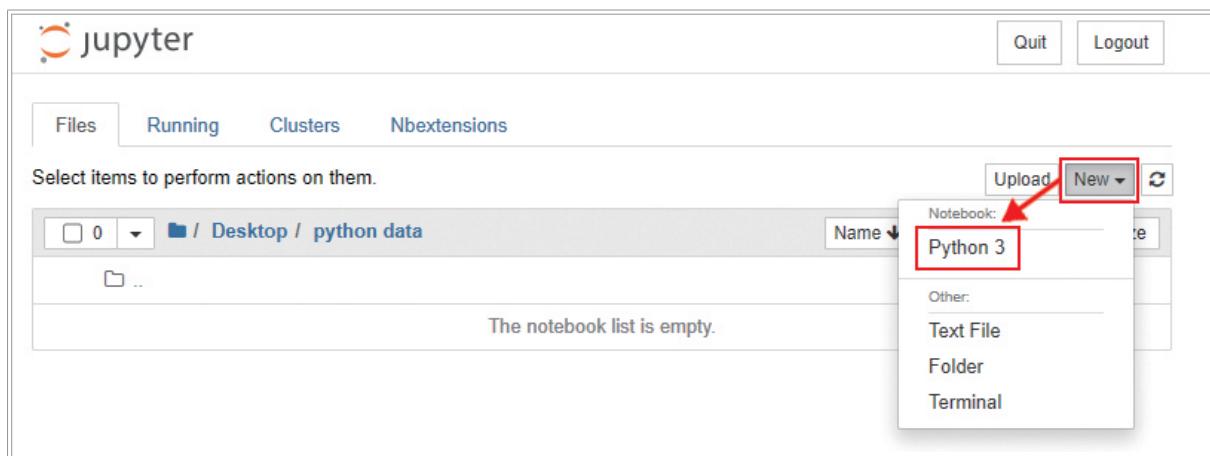
③ 데이터를 저장하고, 불러오고, 분석할 경로의 폴더를 하나 생성

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

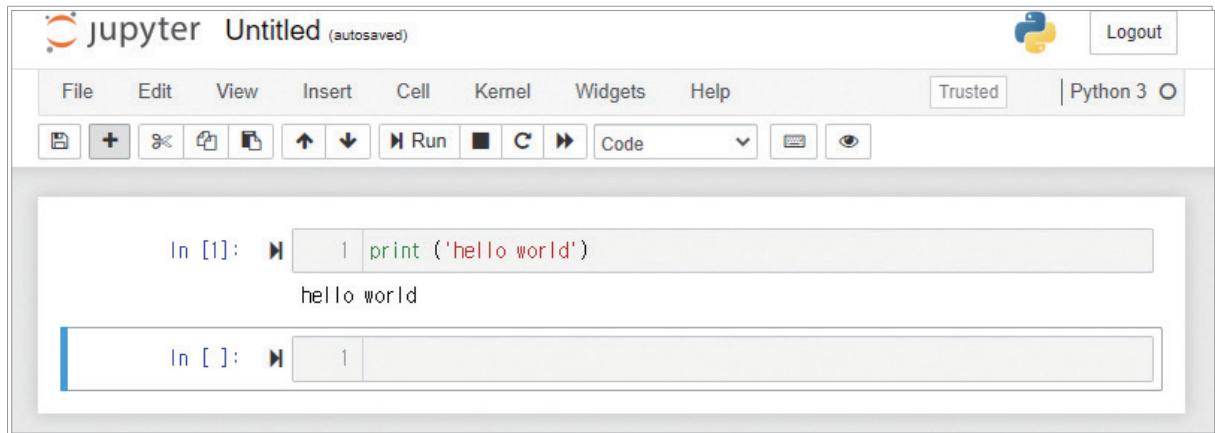
(1) 주피터에서 ‘python data’ 폴더 확인



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

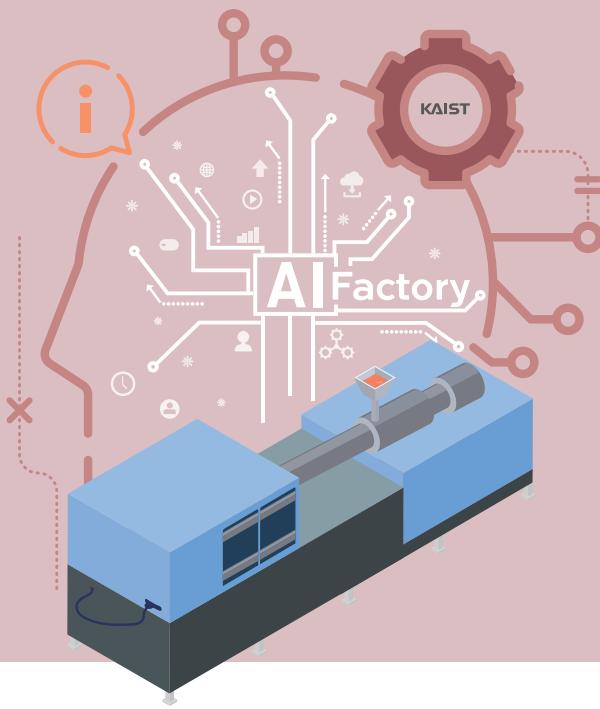


(3) ‘hello world’ 출력 확인해보기: 보이는 In[] 옆의 회색 창에 `print('hello world')` 입력 후, **shift + Enter** 키 눌러서 실행. : 자동으로 저장되며, 확장자는 ‘(파일이름).ipynb’로 저장



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, and Logout. Below the menu is a toolbar with icons for file operations like New, Open, Save, and Run, along with a Code dropdown. The main area contains two code cells. The first cell, labeled "In [1]", contains the Python code `print ('hello world')`. When the cell is run (indicated by the play icon), it outputs "hello world". The second cell, labeled "In [ ]", is currently empty.

# 『사출성형기 AI 데이터셋』 분석실습 가이드북



중소벤처기업부



스마트제조혁신추진단



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