

# Contents

<b>1 분석요약</b>	04
<b>2 분석 실습</b>	
<b>1. 분석 개요</b>	05
<b>1. 분석 배경</b>	05
1) 공정(설비) 개요	
2) 이슈사항(Pain point)	
<b>2. 분석 목표</b>	07
1) 분석 목표	
2) 데이터 정의 및 소개	
3) 데이터 분석 기대효과	
4) 시사점(implication) 요약기술	
<b>2. 분석실습</b>	08
<b>1. 제조데이터 소개</b>	08
1) 데이터 수집 방법	
2) 데이터 유형/구조	
3) 데이터 (품질) 전처리	
<b>2. 분석 모델 소개</b>	14
1) 유전 알고리즘 선정 이유	
2) 유전 알고리즘 방법론	
3) 유전 알고리즘 구축 절차	
<b>3. 분석 체험</b>	23
1) 필요 SW, 패키지 설치 방법 및 절차 가이드	
2) 분석 단계별 프로세스	
[단계 ①] 라이브러리/데이터 불러오기	
[단계 ②] 데이터 유효성 검증	
[단계 ③] 데이터 종류 및 개수 확인	
[단계 ④] 데이터 정제 (전처리)	
[단계 ⑤] 변수 생성	
[단계 ⑥] 유전 알고리즘 구축	
[단계 ⑦] 유전 알고리즘 실행	
[단계 ⑧] 결과 분석 및 해석	
<b>3. 유사 타현장의 「생산계획 최적화 AI 데이터셋」 분석 적용</b>	48
<b>부 록</b>	
<b>분석환경 구축을 위한 설치 가이드</b>	49
<b>데이터 품질 전처리 [실습 코드]</b>	61

# 「생산계획 최적화 AI 데이터셋」 분석실습 가이드북

- 필요 SW : Python, Anaconda – Jupyter Notebook
- 필요 패키지 : pandas, Matplotlib
- 분석 환경 : [운영체제] Window, Ubuntu 14.0 이상, [CPU] AMD Ryzen 3700x 2.3 GHz, [RAM] 16GB
- 필요 데이터 : machine\_info.csv & order\_info.csv

## 1 분석요약

No	구 분		내 용
1	분석 목적 (현장 이슈, 목적)		<ul style="list-style-type: none"> <li>- 제조업에서 생산되는 부품들은 다양한 인도 납기 및 서로 다른 NC설비가 요구된다.</li> <li>- 따라서, 부품마다 서로 다른 NC설비 요구조건을 충족시키는 가운데 인도 납기 지연 최소화를 달성하는 생산계획을 수립하는 것이 본 가이드북의 목표이다.</li> <li>- 이를 위해서, 먼저 제조업 생산 공정 데이터 전처리 및 납기 지연을 일으키는 주요 변수 간 상관관계 분석한다.</li> <li>- 고객사 인도 납기를 고려한 제조업 NC설비 공정 생산계획 최적화 전략을 수립할 것이다.</li> <li>- 높은 계산 복잡도를 가진 최적화 모델링을 효율적으로 풀기 위해, 인공지능 기반의 휴리스틱 알고리즘 구현을 통한 솔루션 도출할 것이다.</li> </ul>
2	데이터셋 형태 및 수집 방법		<p>1) 분석에 사용된 변수 : item(중산도면), time(시간), machine(기계), cycle time(사이클타임), demand(수량), unit cost(단가), prepaid(선금)</p> <p>2) 데이터 수집 방법 : 생산 NC설비 및 생산 미결 리스트 데이터</p> <p>3) 데이터셋 파일 확장자 : CSV</p>
3	데이터 개수 데이터셋 총량		<ul style="list-style-type: none"> <li>- 데이터 개수 : 총 150,403개</li> <li>- 데이터셋 총량 : 1.4MB</li> </ul>
4	분석적용 알고리즘	알고리즘	유전 알고리즘 (Genetic algorithm)
		알고리즘 간 락 소개	<p>유전 알고리즘은 변이과정과 적자생존으로 최적의 개체를 유전하는 자연현상을 최적화에 적용한 알고리즘이다. 해당 알고리즘은 임의의 조합해 집합 (population)을 생성, 최적도 (fitness)가 높은 조합 해를 선택(selection), 재조합(crossover), 변이(mutation)하는 일련의 과정(generation)을 반복하며 더 나은 조합해를 찾아가는 과정이다. 이 알고리즘의 장점은 목적함수의 수치해석을 이용하지 않 기 때문에, 수치 해석적 최적 해를 찾기 힘든 목적함수나 제약조건 (예 : machine마다 item 생산 능력, 정수 조건 등을 고려 등)을 가지는 문제의 솔루션 도출에 효과적이다.</p>
5	분석결과 및 시사점		<ul style="list-style-type: none"> <li>- NC설비 공정의 작업 우선순위 설정 개선 및 납기 지연 20% 이상 감소를 기대한다.</li> <li>- 제조업 생산 계획을 최적화하는 인공지능 기반 알고리즘 개발할 것이다.</li> <li>- 제조기업의 생산 계획 최적화를 통해 다양한 제조 공정 현장에 널리 적용될 것으로 기대한다.</li> </ul>

## 2 분석 실습

### 1. 분석 개요

#### 1. 분석 배경

##### 1) 공정(설비) 정의 및 특징

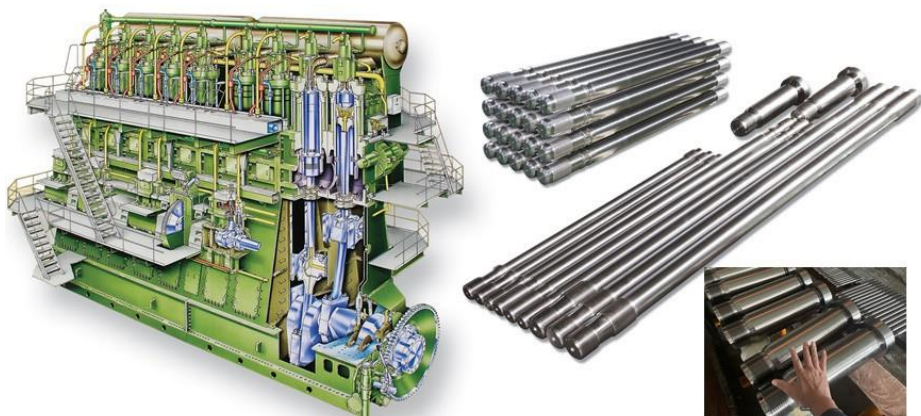
##### • 선박용 볼트 및 너트 제조공정

- 해당 기업((주)유피시앤에스)은 선박 엔진 기자재 중 볼트와 너트 제품을 생산한다. 선박 엔진에 들어가는 볼트와 너트의 제조공정은 환봉(둥글고 긴 막대 모양의 쇠로 만든 창 혹은 쇠막대기)을 가지고 도면에 의한 소재 크기만큼 쇠톱으로 절단하여 목적의 경도를 높이기 위해 열처리 과정을 거친다. 이에 선반으로 가공공정을 거쳐 나사산을 만드는 전조 과정을 통해 최종제품이 완성된다.



[그림 1] 선박용 볼트, 너트 제조공정 프로세스 도식화

##### • 전조 나사의 특성



[그림 2] 선박 엔진에 들어가는 볼트, 너트 예시

- ① 금속은 입자 간 상호 연결된 선을 가지며 이를 단류선 (grain flow)라고 한다.
- ② 이웃하는 금속 입자 간의 결속에 중요한 단류선은 중요한 요소이며 완제품 특성에 영향을 미친다.
- ③ 전조 나사는 단류선이 끊어지지 않고 강도가 매우 높고 치밀한 형상을 한다.
- ④ 열처리 전 전조의 경우 전조 금형의 수명이 길어지고 생산이 쉬워 많은 업체가 선호한다.
- ⑤ 전조로 인해 생산된 나사부의 치밀한 조직이 열처리 과정에서 재배열을 하게 되어 같은 등급의 고장력 볼트일지라도 열처리 후 전조 방식으로 생산된 볼트가 더 큰 힘을 견딘다.
- ⑥ 이로 인해 극한의 조건에서 사용되는 볼트일 경우 주문 전 열처리 후 전조 방식을 업체에 특별히 주문하는 것이 좋다.

## 2) 이슈사항(Pain point)

### • 공정(설비) 개요

- 제조공정에서는 각종 문제가 발생하며 이를 해결하기 위해서는 우선 현상 자체를 물리·화학적으로 잘 이해할 필요가 있다. 하지만 개개의 경우에 대해서 신속히 그 원인을 규명해서 적절한 대책을 세우는 것은 쉬운 일이 아니다. 원인이 되는 요인이 몇 가지가 복잡하게 동시에 얹혀서 나오기도 하고, 한 가지 불량현상의 대책을 세우는 것에 의해 다른 불량현상을 해결하는 때도 있을 것이다. 불량을 발생시키는 중요한 요인에 대한 이해는 필요한 부분이다. 따라서 이하에 제조공정 및 납기의 지연 현상과 그 직 접적인 원인에 관해서 설명해 그들 원인에 직결하는 요인을 대략적으로 정리하였으며 실제 공정에서의 이슈 사항들을 함께 나열하였다.
- 선박용 볼트 또는 너트는 선박 엔진 조립 부품의 다양한 공정이 들어간다.
- 고객으로부터 제공되는 공급망 시스템을 통해 견적 및 수주 확정을 통보받게 된다. 고객으로부터 인도 납기를 확정받게 되면, 생산 리드 타임을 고려하여 제조공정 라우팅 별 생산계획을 수립하게 된다.
- 수주 10건 중 2건(20%)은 납기 지연이 발생하며 그 2건 중 1건(10%)은 긴급으로 인한 지연이고, 나머지 1건(10%)은 생산계획의 비효율로 인한 지연이다.
- 제조공장의 공정별 생산 능력은 설비 및 품질검사 일정에 따라 그 기간이 유동적이며 효율적인 생산계획 작성은 많은 시간이 소요된다.

### • 문제해결 장애 요인

- 조선산업의 특성상 모기업의 엔진 조립 일정에 맞추어 부품을 여유롭지 않게(조립일 약 3일 전) 인도받기 때문에 협력 업체의 생산계획 일정은 생산 아이টে를 각각 선급품과 비선급품으로 나누어 생산해야 한다. 선급품은 완제품의 선급검사(외부기관검사)를 진행해야 하고 선급검사를 받는 비용 또한 무시할 수 없으므로 생산 일정은 유동적으로 관리되어야 한다.
- 관리되는 납기는 고객납기, 영업납기, 원재료입고일, 절단작업일, 열처리 작업일, 외주 가공일, 선급검사일, CNC 1,2,3차 가공일, 전조공정일, 포장완료일 등 설비 및 작업자의 능력에 따라 매우 유동적이다.

### • 극복방안

- 아래에 기술하는 생산계획 및 납기 지연에 영향을 미치는 변수를 수집하고 이를 통해 최적화된 생산 일정을 예측할 수 있다. 납기 지연 예측이 발생하면 이에 해당하는 패 널티를 표출하여 현장 작업자가 빠른 조치를 할 수 있게 되어 이를 공정 제어에 이용 할 수 있다. 각 작업을 어떤 장비에서 어떤 순서에 의해 생산하는지에 따라 납기 지연 의 정도 및 공정의 효율성은 극대화될 것이다.

## 2 분석 목표

### 1) 분석 목표

- 선박 엔진 기자재 중 볼트와 너트를 생산하고 제품별 인도 납기 지연이 발생하는 문제와 이를 고려한 생산조건 데이터들의 분석을 통하여 데이터 간의 상관관계를 찾고자 한다. 주요 문제별 원인을 분석하여 생산 계획 최적화를 도출할 수 있는 모델을 설계하고 기계 학습 알고리즘을 활용 및 구현하고자 한다.

### 2) 데이터 정의 및 소개

공정 변수 조건		내 용
독립변수	시간 관련	영업납기(time)
	할당 관련	중산도면(item), NC설비번호(machine)
	가격 관련	단가(cost)
	상태 관련	선급(prepaid), 긴급(urgent)
	수량 관련	수량(qty), 설비일일용량(CYCLETIME, capacity)
종속변수	페널티 여부	amount of item(실제 생산량), penalty(페널티)

### 3) 데이터 분석 기대효과

- 납기 지연이 발생하는 상황에 기존 수집된 데이터들을 분석하여 데이터 사이언스의 관점에서 물류 최적화에 도달할 수 있는 생산계획 최적화 모델 및 알고리즘을 제공한다.

### 4) 시사점(implication) 요약기술

- 현재 선박 부품을 생성하는 공정에서 긴급수주 및 사람의 경험에 의존한 판단으로 납기 지연 및 생산 계획의 효율성을 확보하기 힘들었다. 하여, NC 공정에서 제품별 인도 납기 및 생산 NC 설비를 고려한 생산계획 최적화 모델을 수립 및 분석한다. 또한 인공지능 기법을 활용한 생산계획 솔루션을 도출하는 분석을 수행하고자 한다.
- 본 분석은, NC 공정에서의 제품별 납기 및 가용한 데이터를 수집 후, 가공/전처리, AI 모델 개발과 제조공정의 적용 및 검증을 통해 중소기업에 빅데이터 및 AI를 적용하여 실질적인 생산계획 최적화 및 비용 절감 개선에 이바지할 수 있다는 점에서 시사하는 바가 크다고 판단된다.

## 2. 분석실습

### 1. 제조데이터 소개

#### 1) 데이터 수집 방법

- **제조 분야**: 선박 엔진용 볼트, 너트
- **제조 공정명**: 선박용 볼트, 너트 NC 공정
- **수집 장비**: 공장 내 24개 NC설비 Data, 생산 주문 리스트 Data
- **수집 기간**: 2021/02/10 ~ 2021/10/31 (7개월)
- **수집 주기**: 생산 주문 리스트 도착할 시 수집

#### 2) 데이터 유형/구조

- **데이터셋 구조**: 테이블 형식(tabular)
- **데이터 개수**: NC공정생산이력.xlsx 원본데이터 중  
(CYCLETIME) column 수 7개, row 수 21,151개, 데이터수 148,057개  
(생산미결리스트) column 수 17개, row 수 138개, 데이터수 2,346개
- **AI 데이터셋 주요 변수(속성) 정의**
  - 비식별화 원본 데이터 : NC공정생산이력.xlsx
    - 공정 과정에서 수집된 NC공정 생산 이력 데이터에서 품질 전처리를 수행한 데이터셋이다. 제공된 원본 데이터는 물품 각각에 대한 NC설비 작업 실적 데이터, 물품 각각에 대한 생산 주문 리스트 데이터가xlsx(엑셀파일)의 형태로 제공된다.
    - 분석에 사용한 모델은 데이터를 바탕으로 학습하는 모델이 아니며, 단지 기업에서 해결하고자 하는 기간에 해당하는 데이터셋만 사용하므로 2021년 데이터만 추출하였다.



JSDWG	MCNO	AVG_CT	MAX_CT	MIN_CT	JSDWG	MCNO	AVG_CT	MAX_CT	MIN_CT
ITEM	NC설비번호	CYCLETIME (평균,개당,분)	*참조값	*참조값	ITEM	NC설비번호	CYCLETIME (평균,개당,분)	*참조값	*참조값
050060	0433	1.08	1.42	0.73	056250	0440	8.91	10	7.82
050093	0404	7.13	16.5	4	056251	0408	3.02	3.02	3.02
050093	0408	4.67	5.58	4.22	056251	0410	3.17	3.17	3.17
050093	0410	4.5	4.5	4.5	056251	0424	2.97	2.97	2.97
050093	0416	3.92	3.92	3.92	056251	0426	3.22	3.22	3.22
050093	0417	4.17	4.17	4.17	056251	0440	3.45	3.45	3.45
050093	0424	4.2	4.22	4.17	056253	0404	7	7	7
050093	0433	4.15	4.17	4.12	056253	0410	9.75	10.17	9.33
050095	0433	4.68	5.17	3.7	056253	0425	12	12	12
050095	0434	1.51	1.67	1.35	056253	0426	6.88	7	6.75
050105	0426	2	2	2	056253	0434	5.87	5.87	5.87
050110	0427	2	2	2	056254	0404	23.33	23.33	23.33
050110	0435	1.82	1.82	1.82	056254	0426	17.17	17.17	17.17
050110	0436	2.02	2.02	2.02	056255	0404	5.5	5.5	5.5
050153	0407	5.46	5.67	4.87	056255	0409	1	1	1
050153	0408	4.43	5.4	1.2	056255	0416	5.12	5.12	5.12
050153	0409	5.12	5.8	4.75	056255	0422	9.17	9.17	9.17
050153	0410	1.11	1.17	1.08	056255	0426	1.27	1.27	1.27
050153	0416	4.5	5.5	2.15	056255	0436	2.58	5.5	1.12
050153	0424	3.7	7	1.15	056255	0440	10.1	10.1	10.1
050153	0425	4.52	9	1.1	056256	0439	2.4	2.4	2.4
050153	0426	3.33	5	1.47	056260	0408	12.61	30	2.82
050153	0433	1.46	1.5	1.33	056260	0409	3.22	4.83	2
050153	0434	1.25	1.33	1.22	056260	0424	3.92	5.5	2.33
050153	0435	2.33	4.92	1.07	056260	0426	5.34	7.67	3
050153	0436	3.6	5.32	1.12	056264	0407	4.58	4.58	4.57

[그림 3] 원본 데이터(NC공정생산이력.xlsx-CYCLETIME) 예시

- NC공정생산이력 원본 데이터의 CYCLETIME 데이터는 각 NC설비에 대한 능력에 따라 생산 가능한 아이템 및 개당 생산 가능한 평균 시간(단위: 분)을 설명한다.

생산 미결 리스트 - NC대기(L/T 3日)															
작성일자 : 2021-05-07															
№	발주번호	호선	중산도	용명	재질	소재	길이	중량	영업일	선급	현재진행공정	지연	비고	진도	
1	A21030400ET	7H35DF KBA0072	057387	STUD FOR CON-ROD SHAFT	SCM439	Φ35	246	84	158	2021-05-07	LR/W	CNC2차대기-202105	1	다에스티	진급
2	A21042000H9		K01086	STUD BOLT	SCM440	Φ35	409	4	12	2021-05-10		CNC1차대기-202105	1	중산	진급
3	A210409000R	KBW1001211/HOT	057387	STUD FOR CON-ROD SHAFT	SCM439	Φ35	246	80	150	2021-05-14	BV/W	CNC1차대기-202104	-17	다에스티	진급
4	A210405000Q	KBW1001203/HOT	057387	STUD FOR CON-ROD SHAFT	SCM439	Φ35	246	320	601	2021-05-14	BV/W	CNC1차대기-202104	-24	다에스티	진급
5	D191024006V-04	APOL5467	S00341	HEXAGON NUT	S45C	Φ70	35	500	574	2021-02-10		CNC1차(0)	-553		
6	A2012090023	KMAR2178	052996	STUD	SCM440	Φ140	1,140	2	50	2021-02-18		CNC2차대기-202101	-117		
7	D191024006V-05	APOL5467	S00341	HEXAGON NUT	S45C	Φ70	35	500	574	2021-03-10		CNC1차(0)	-553		
8	D191024006V-06	APOL5467	S00341	HEXAGON NUT	S45C	Φ70	35	500	574	2021-04-10		CNC1차(0)	-553		
9	A2104230094	S163	057387	STUD FOR CON-ROD SHAFT	SCM439	Φ35	246	4	8	2021-04-28	LR-W	CNC1차대기-202104	-3	현대중공업	
10	A21040900A4	CME53097/CME53	072016	STUD BOLT	SCM440 QT	Φ24	360	24	31	2021-05-04		CNC2차대기-202104	-11	HSD엔진(주)	
11	A210414009M	YASA5062	072320	HOLD/G DOWN BOLT	SCM440	Φ45	533	62	415	2021-05-07	NK/W	CNC1차(19)	2	HSD엔진(주)	
12	A21033100BZ	PANT2414/NAT 2	069354	FITTED STUD	SCM439	Φ60	444	16	159	2021-05-10		CNC1차대기-202105	3	HSD엔진(주)	
13	A2104280015	2021년6월오선엔	057481	PISTON ROD STUD BOLT	SCM439	Φ40	473	3	14	2021-05-10		CNC1차대기-202105	3	중산	
14	A210413007O	KAS1070045	072345	PISTON ROD BOLT	SCM440	Φ38	160	2	3	2021-05-10		CNC1차대기-202104	-11	현대글로벌서비스	
15	A210413007N	KAS1070045	072344	PISTON CROWN BOLT	SCM439	Φ38	157	2	3	2021-05-10		CNC1차대기-202105	1	현대글로벌서비스	
16	D191024006V-07	APOL5467	S00341	HEXAGON NUT	S45C	Φ70	35	398	457	2021-05-10		CNC1차(0)	-553		
17	A210303007B	CME53097/KMAR	072099	CHEESE HEAD FITTED BOLT	SCM440	Φ75	295	36	372	2021-05-11		CNC2차(16)	0	HSD엔진(주)	
18	A210407002T	3공강-3247호선	067667	STUD	SCM440	Φ35	439	40	133	2021-05-11		CNC1차대기-202105	3	현대중공업	
19	D21030900EM	KBA007101/KBA00	060039	STUD FOR CON ROD BIG END	SCM439	Φ32	334	106	225	2021-05-11	검사품	CNC1차(7)	-41	현대중공업	
20	D210318002P	MINE2332P1/AET	065810	ROUND NUT	SCM440	Φ130	91	26	255	2021-05-12		CNC1차대기-202105	4	HSD엔진(주)	
21	A21040500E5	EP21-0062(R.1)-7	072209	UNDERCUT BOLT (TCA66)	SCM439	Φ32	273	256	446	2021-05-13	검사품	CNC2차(120)	-14	ENSYS (수출)	
22	A21040500E6	EP21-0062(R.1)-9	072210	UNDERCUT BOLT (TCA77)	SCM439	Φ40	327	256	833	2021-05-13	검사품	CNC2차(99)	-1	ENSYS (수출)	
23	A210422008D	DUBHE_1404_LNY	K02528	STUD-FUEL VALVE	SCM435	Φ26	243	2	2	2021-05-13		CNC2차대기-202105	2	HSD엔진(주)	
24	D210422004P	DUBHE_1404_LNY	K01749	STUD. FUEL VALVE	SCM435	Φ22	243	2	1	2021-05-13		CNC2차대기-202105	2	HSD엔진(주)	

[그림 4] 비식별화된 원본 데이터(NC공정생산이력.xlsx-생산미결리스트) 예시

- NC공정생산이력 원본 데이터의 생산미결리스트는 생산계획 최적화를 해야 할 생산주문 리스트이다.

- 1차 가공 데이터 (NC공정생산이력\_1차 가공 데이터.xlsx)

- NC공정생산이력\_1차 가공 데이터 중 생산미결리스트(1차 가공) 데이터는 NC공정생산이력 원본데이터에 단가 열을 추가한 것이다.

№	발주번호	후선	중산도면	품명	재질	소구	길이	중량	영업날짜	선급	원재탄형공정	지명	비고	권도	단가
1	A21032400AM	KBA006830	K04033	MAIN BEARING STUD	SNCM439	Φ40	525	318	1637	2021-05-13	기시물	CNC2기(82)	-2 연내종류	기공	29,810
2	D21033000E3	KBA007332/K3A00	K04031	SIDE BOLT	SNCM439	Φ40	329	383	1254	2021-05-24	기시물	CNC2기(269)	1 연내종류	기공	16,225
3	D210525305G		009622	SCREW,hexagon head	SNCM439	Φ30	200	20	36	2021-05-30		CNC2시작기-202105	2 생산	기공	5,409
4	D210525300E		051718	SCREW	SCM435	Φ30	235	19	70	2021-05-30		CNC' 시작기-202105	1 생산	기공	8,333
5	A210513009C	A200336255-10	059984	STUD,REDUCED SHANK	SNCM439	Φ35	373	4	11	2021-06-03		CNC' 시작기-202105	4 MAN Energy Solutions-(수출)	기공	36,533
6	A210323001U	KBD000816/KBA00	057791	STUD FOR CYL HEAD	SNCM439 QT	Φ40	787	196	1527	2021-06-18	기시물	CNC2작나기-202105	-1 연내종류	기공	45,500
7	A21042200HM	KBA007409/K3A00	K03948	CYLINDER HEAD STUD	SNCM439	Φ45	885	85	942	2021-06-18	기시물	CNC2기(0)	0 연내종류	기공	37,641
8	D191024006V-04	APD_5467	S03341	HEXAGON NUT	S45C	Φ70	35	500	574	2021-02-10		CNC' 시작(0)	-577		2,990
9	A2012090023	KV/AR2178	052996	STUD	SCM440	Φ140	1,140	2	50	2021-02-18		CNC2시작기-202101	-141		20,000
10	D191024006V-05	APD_5467	S03341	HEXAGON NUT	S45C	Φ70	35	500	574	2021-03-10		CNC' 시작(0)	-577		2,990
11	D191024006V-06	APD_5467	S03341	HEXAGON NUT	S45C	Φ70	35	500	574	2021-04-10		CNC' 시작(0)	-577		2,990
12	D191024006V-07	APD_5467	S03341	HEXAGON NUT	S45C	Φ70	35	398	457	2021-05-10		CNC' 시작(0)	-577		2,990
13	D210324009C	KBA007323	068157	SIDE STUD	SNCM439	Φ40	322	208	667	2021-05-13	기시물	CNC2작나기-202105	-7 연내종류		18,500
14	A210504007C	E001001	072412	TENSION SCREW	SNCM439	Φ85	273	5	61	2021-05-30		CNC' 시작기-202105	-5 STX렌치		96,950
15	A210525005G		K04101	STUD FOR T/V DAMPER	SNCM439	Φ35	468	15	53	2021-05-30		CNC' 시작기-202105	0 생산		15,640
16	A201209002K	H3342	052996	ELASTIC BOLT(P.K.)	SNCM439	Φ120	1,139	2	50	2021-05-30		CNC2시작기-202101	-141		20,000
17	A210518007P	MINE2332P /AV N	S02271	HEXAGON BOLT-Ls	SCM440	Φ30	165	150	140	2021-05-31		CNC' 시작기-202105	4 HSD렌치		4,000
18	A21030400DI	202102020034-II	057386	STUD FOR CON-ROD BIG END	SNCM439	Φ35	461	144	504	2021-06-01	DNVCLW	CNC2작나기-202105	-2 역침시작		22,750
19	A19121000EE	WMAR0693	064421	END CHOCK BOLT	SCM440	Φ65	488	2	26	2021-06-01		CNC' 시작기-202002	-470		386,680
20	A191113005I	WMAR0693	067693	HOLD'S DOWN BOLT	SCM435	Φ45	521	56	366	2021-06-01		CNC' 시작기-202002	-470		46,280
21	A210520005B	MSC 0265	062138	FITTED STUD	SNCM439 QT	Φ60	210	20	96	2021-06-03		CNC2시작기-202105	2 HSD렌치(수)		18,500
22	A210525008D	BURDONAV_H-1894	059984	STUD,REDUCED SHANK	SNCM439	Φ35	373	4	11	2021-06-03	LR	CNC' 시작기-202105	4 HSD렌치(수)		14,835
23	A210510009C	D200335913-90	072430	HOLD'S DOWN BOLT	SCM435	Φ38	468	30	128	2021-06-03	LR	CNC' 시작(0)	5 MAN Energy Solutions-(수출)		28,687
24	A210204002E	KBA007311/K3A00	057791	STUD FOR CYL HEAD	SNCM439 QT	Φ40	787	216	1682	2021-06-03	기시물	CNC2작나기-202105	7 연내종류		42,810
25	A210406004M	PA/VT2413/KAA00	069559	EXHAUST VALVE STUD	S30CRNIMOC	Φ85	964	28	1205	2021-06-04	기시물	CNC' 시작기-202105	4 HSD렌치(수)		242,250
26	A210513009W	12P64357730	058039	STUD,AXIAL V RR DAMPER	SNCM439	Φ30	605	64	216	2021-06-05		CNC' 시작기-202105	1 유압공급(수)		23,740
27	A210513008X	HMD2799/H-MD25	061088	STUD,DAMPER	SCM440	Φ26	534	48	107	2021-06-05		CNC' 시작기-202105	1 유압공급(수)		11,500
28	D210422005Y	MINE2332P /AE	058799	NUT FOR CYL COVER STUD	SCM440	Φ140	81	93	944	2021-06-07		CNC' 시작기-202105	2 연내종류		28,111
29	A21033100DF	PLEI5066	059046	STUD	SCM440	Φ30	263	5	7	2021-06-09		CNC' 시작기-202104	-43 HSD렌치(수)		7,752

[그림 5] 원본 데이터에서 단가 열이 추가된 1차 가공 데이터

- NC공정생산이력\_1차 가공 데이터 중 CYCLETIME은 원본 데이터와 같다.

- 1차 가공 데이터 속성정의 표

- NC공정생산이력\_1차 가공 데이터 중 CYCLETIME 속성정의 표

[표 1] NC공정생산이력\_1차 가공 데이터 중 CYCLETIME 속성정의 표

속성(column)	설명	비고
ITEM	생산 제품 품목에 대한 코드 부여	string
NC설비번호	각 제품을 생산하는 NC설비	string
CYCLETIME	각 제품을 생산하는 NC설비(j)의 일일 용량	float



- NC공정생산이력\_1차 가공 데이터 중 생산미결리스트(1차 가공) 속성정의 표

[표 2] NC공정생산이력\_1차 가공 데이터 중 생산미결리스트(1차 가공) 속성정의 표

속성(column)	설명	비고
No.	생산 순서 번호	int
발주번호	생산 주문 목록	string
호선	생산을 지시한 NC설비의 상세 호선	string
중산도면	생산을 지시한 제품 품목에 대한 코드 부여	string
품명	제품의 모델 코드	string
재질	제품의 재료가 가지는 성질	string
소재	생산되는 제품의 바탕이 되는 재료	string
길이	생산 제품의 외형적 길이	float
수량	생산 결과로 약속된 물건의 양	int
중량	생산 제품의 무게	float
영업납기	생산 제품이 고객에게 인도되기까지 약속된 날짜	datetime
선급	생산 제품이 고객에게 인도되기까지 약속된 날짜	string
현재진행공정	제품 생산의 전체 공정에서 현재의 공정 상태	string
지연	생산 제품이 고객에게 인도되기까지 약속된 날짜보다 초과된 일수	float
비고	각 생산 제품 주문 업체	string
긴급	제품 생산시 고객으로부터 기존 영업납기보다 빠른 요구를 받은 상태	string
단가	제품 생산시 발생하는 제품 단위당 가격	int

- 2차 가공 데이터: machine\_info.csv, order\_info.csv

- machine\_info.csv는 1차 가공데이터에서 속성 이름을 수정하였다.
- order\_info.csv는 1차 가공데이터에서 분석모델에서 필요한 속성만 추출하였다.

- 2차 가공 데이터 속성정의 표

- machine\_info

[표 3] machine\_info.csv의 속성정의 표

속성(column)	설명	비고
item	생산 제품 품목에 대한 코드 부여	string
machine	각 제품을 생산하는 NC설비	string
capacity	각 제품을 생산하는 NC설비(j)의 일일 용량	float

- order\_info

[표 4] order\_info.csv의 속성정의 표

속성(column)	설명	비고
영업납기(time)	해당 제품을 생산하는 시간	datetime
중산도면(item)	생산 제품 품목에 대한 코드 부여	string
수량 (qty)	생산 결과로 약속된 제품의 양	int
단가 (cost)	정해진 수량을 만들지 못했을 때 부여되는 패널티	int
선급 (urgent)	제품 생산시 고객으로부터 기존 영업납기보다 빠른 요구를 받은 상태	bool

## • 주요 변수 기술 통계

- machine\_info.csv

[표 5] machine\_info.csv의 주요 변수 기술 통계

속성(column)	데이터형	개수	평균	표준편차	최소값	중앙값	최대값
item	string	21151	-	-	-	-	-
machine	string	21151	-	-	-	-	-
capacity	float	21151	10.44	18.30	0.05	5.50	930.00

- order\_info.csv

[표 6] order\_info.csv의 주요 변수 기술 통계

속성(column)	데이터형	개수	평균	표준편차	최소값	중앙값	최대값
영업납기(time)	datetime	127	-	-	-	-	-
중산도면(item)	string	127	-	-	-	-	-
수량 (qty)	float	127	114.11	140.70	2	48	716
단가 (cost)	int	127	45204.13	81277.08	2990	21160	481190
선급 (urgent)	bool	127	-	-	-	-	-

## • 독립변수/ 종속변수 정의

- 독립변수(Independent Variable)란, 다른 변수에 영향을 받지 않는 변수로 입력 값이나 원인을 나타낸다. 독립변수는 입력자에 의해 조절이 가능하며 원인변수 (Explanatory Variable), 예측변수(Predictor Variable)라고도 부른다.
- 종속변수(Dependent Variable)란, 독립변수의 변화에 따라 어떻게 변화하는지를 알고 싶어 하는 변수로 결과물이나 효과를 나타낸다. 이로 인해 종속변수를 반응변수 (Response Variable), 결과변수(Outcome Variable)라고도 부른다.
- 따라서 독립변수와 종속변수는 원인과 결과의 관계를 맺는다.

### 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 6가지 품질 지수를 파악하고, 데이터 전처리를 통해 품질 지수를 향상한다.

#### • 데이터 품질 지수

- 완전성(Completeness) : 필수항목에 누락이 없어야 한다.
- 유일성(Uniqueness) : 데이터 항목은 유일해야 하며 중복되어서는 안된다.
- 유효성(Validity) : 데이터 항목은 정해진 데이터 유효범위 및 도메인을 충족해야 한다.
- 일관성(Consistency) : 데이터가 지켜야 할 구조, 값, 표현되는 형태가 일관되게 정의되고, 서로 일치해야 한다.
- 정확성(Accuracy) : 실제 존재하는 객체의 표현 값이 정확히 반영되어야 한다.
- 무결성(Integrity) : 데이터베이스 자료의 오류 없이 변화에 영향을 받지 않고 데이터의 정확성과 일관성, 유효성이 보호되어야 한다.

#### • 데이터 품질 지수 세부 설명

- 완전성 품질 지수 =  $(1 - (\text{결측치} / \text{전체 데이터 수})) * 100$ 
  - ① Null 값이 30% 이상인 데이터들은 데이터의 완전성이 떨어지기 때문에 열별 Null 값의 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 =  $(1 - (\text{중복데이터 수} / \text{전체 데이터 수})) * 100$ 
  - ① 이 데이터는 유일한 값을 가지는 열이 존재하지 않으므로 유일성을 판단하지 않는다.
  - ② 유일성을 판단하고 싶다면 idx와 workingtime 두 열을 이용하여 합성키를 만든 뒤 기본키 key 로 설정하여 중복을 판단하면 된다.

- 유효성 품질 지수

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

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

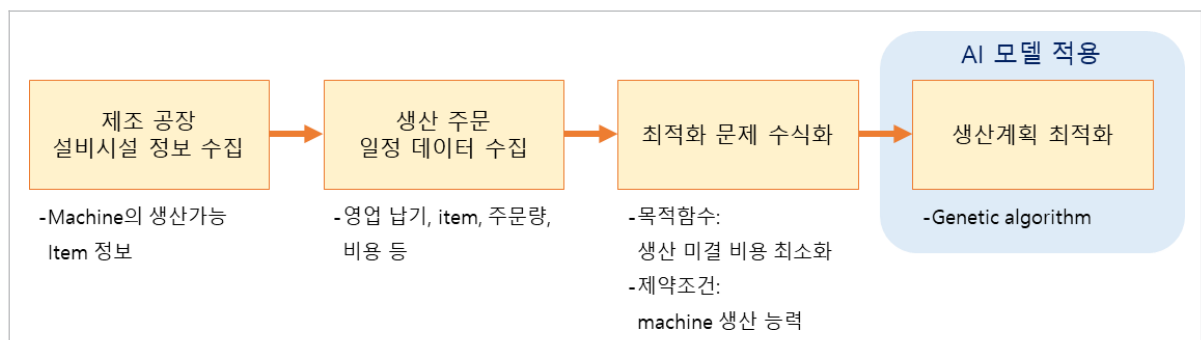
- ① 데이터 테이블 간 종속관계 확인한다.
- ② 관계가 있다면, 자식 테이블(종속되는 테이블, 본 분석에서는 'item\_info' 데이터)의 key 정보가 부모 테이블 (종속하는 테이블, 'machine\_info' 데이터) key 정보에 모두 종속이 되는지 'isin()' 함수를 활용하여 확인한다.

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

본 분석에서 사용되는 데이터는 모두 독립적이기 때문에 정확성을 판단하지 않는다.

- 무결성 품질 지수 = (유일성, 유효성, 일관성 지수중 100%가 되는 지수 개수 / 3)\*100:
- 무결성 품질지수는 고려되는 모든 데이터가 '유일성', '유효성', '일관성' 지수를 만족하는지 확인하는 지수이다. 본 가이드 북에서는 세 가지 지수중 100% 품질을 달성하는 지수의 비율을 무결성 품질지수로 정의한다.

## 2.2 분석 모델 소개

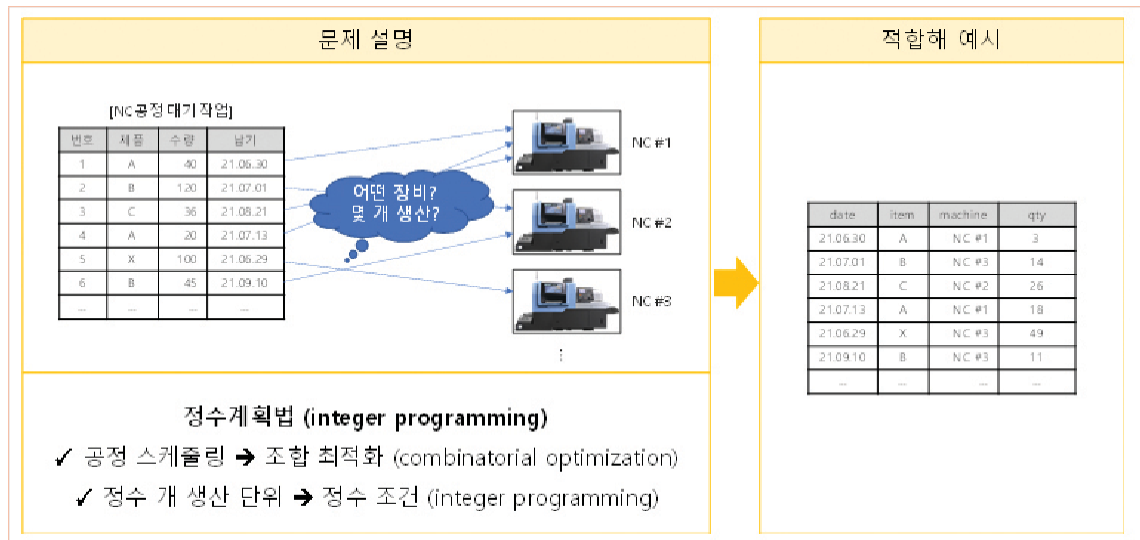


[그림 6] 생산공정에서 데이터 흐름 및 AI 모델 적용 단계 도식화

### 1) 유전 알고리즘 선정 이유

- 해당 문제는 주어진 납부기한에 맞춘 부품 제조공정 스케줄링을 위한 조합 최적화 (combinatorial optimization) 문제이다. 또한, 생산공정 스케줄링은 생산된 물건이 단

위이므로 정수 조건(integer constraint)<sup>1)</sup>이 필요하다. 정수 조건을 고려한 조합 최적화 문제는 정수계획법(integer programming)이며 이는 다항 시간 내에 풀 수 없는 np-hard 문제이다. [그림 7]은 본 프로젝트 문제 설명 및 문제 유형을 도식화한 것이다.



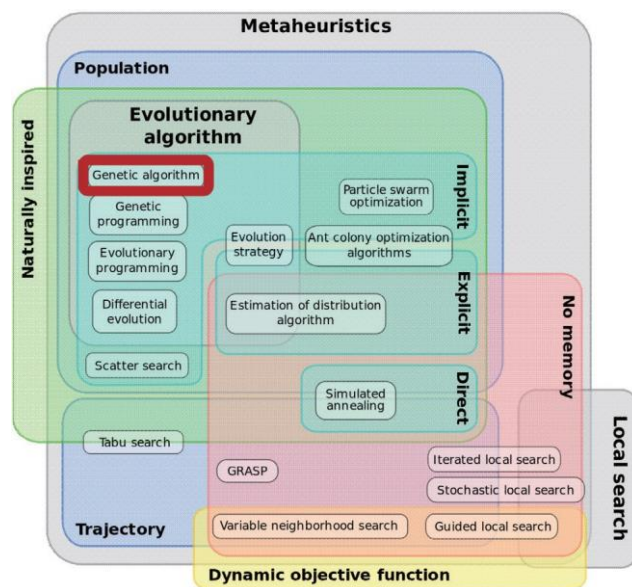
[그림 7] 문제 유형 도식화

- 정수계획법 문제를 해결하기 위한 대표적인 방법론에는 수치 해석적 방법론 (mathematical programming)과 휴리스틱한 방법론(heuristic programming) 이 있다. 수치해석적 방법론은 해를 구할 수 있다면 전사적 최적해(global optimal solution)가 보장된다는 장점이 있지만, 문제 구조에 따라, 혹은 크기에 따라 어떠한 해 도 도출하지 못할 수 있다. 휴리스틱한 방법론은 전사적 최적 해를 보장하지는 않지만, 문제 크기나 구조에 상관없이 최적 해에 근접한 해를 효과적으로 도출할 수 있다. 또한, 목적함수의 수치해석을 이용하지 않기 때문에, 수치 해석적 최적 해를 찾기 힘든 목적 함수나 제약조건(예: machine마다 item 생산 능력, 정수 조건 등을 고려 등)을 가지는 문제의 솔루션 도출에 효과적이다. 덧붙여, 도메인(domain)에 상관없이 적용할 수 있는 휴리스틱 방법론을 메타 휴리스틱 방법론(meta-heuristics)이라 한다. 본 가이드북에서는 생산공정 스케줄링뿐만 아니라 다양한 정수 조합 최적화 문제에 적용할 수 있는 확장성을 위해 메타 휴리스틱 알고리즘 중 하나를 채택하였다.
- [그림 8]은<sup>2)</sup> 다양한 메타 휴리스틱 방법론의 예시이다. 이 중 타부 서치(tabu search)와 담금질 기법(simulated annealing), 유전 알고리즘(genetic algorithm)이 대표적이다.

1) 정수 조건 (integer constraint) : 모델의 해가 정수이어야 하는 제약 조건

2) Tian, Zhonghuan, and Simon Fong. "Survey of meta-heuristic algorithms for deep learning training." Optimization algorithms—methods and applications (2016).

타부 서치는 해가 될 수 없는 금지(tabu) 조건을 사전에 정의한다는 특징이 있다. 이 방법은 임의의 해를 구성한 후 금지 조건에 해당하는 해를 따로 기억(memory), 이를 금지 조건과 기억에 저장된 해를 제외하면서 가장 좋은 해를 선택한다. 담금질 기법은 금속 온도를 높인 후 냉각시키는 과정을 반복하며 쇠의 결정을 안정화하는 일련의 담금질 과정을 이용한 알고리즘이다. 이 알고리즘은 임의의 해를 고려하여 적합도를 찾는 반복과정 중 일정 구간마다 현재 임의의 해에서 이웃하지 않는 임의의 해를 고려하며, 이 과정으로 전역적 근사해를 찾는다. 유전 알고리즘은 변이과정과 적자생존으로 최적의 개체를 유전하는 자연현상을 최적화에 적용한 알고리즘이다. 해당 알고리즘은 임의의 조합해 집합(population)을 생성, 최적도(fitness)가 높은 조합해를 선택(selection), 재조합(crossover), 변이(mutation)하는 일련의 과정(generation)을 반복하며 더 나은 조합해를 찾아가는 과정이다. 세 알고리즘의 특징과 한계를 [표 7]에 정리했다.



[그림 8] 다양한 메타휴리스틱 방법론 예시

[표 7] 대표적인 메타휴리스틱 비교

	타부서치 (Tabusearch)	담금질 기법 (Simulated annealing)	유전 알고리즘 (Genetic algorithm)
기원	기억 프로세스 (memory process)	금속 담금질	Natural evolution
통제 변수	- 금지 조건 (Tabucriteria) - 기억 사이즈 (Tabulist size)	- 볼츠만 상수 (Boltzmann constant) - 냉각온도 비율 (cooling controlled rate)	- 한 세대를 구성할 염색체 수 - 총 세대 수 등
특징	- 전역적 근사해를 찾을 수 있음	- 전역적 근사해를 찾을 수 있음	- 전역적 근사해를 찾을 수 있음 - 계산 간편성
한계점	- 문제에 맞는 금지조건 설정이 어려움	- 문제에 맞는 파라미터 설정이 어려움	- 목적함수 복잡도에 따라 계산 시간이 다소 소요될 수 있음

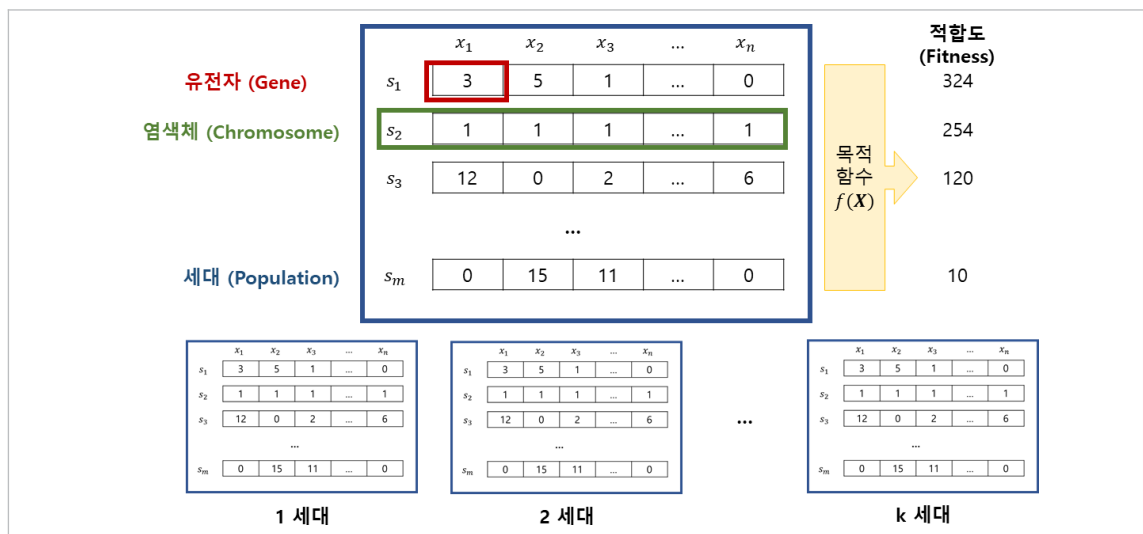


- 본 가이드북에서는 계산의 간편성과 전역적 근사해를 찾기 위해 유전 알고리즘을 채택하였다.

## 2) 유전 알고리즘 방법론

### • 요약

- 1975년 존 홀랜드(John Holland)가 유전 알고리즘의 이론적 기반을 다졌다. 유전 알고리즘은 자연생태계에서 관찰된 진화 방법 및 유전학 이론을 이용한 확률적 탐색 방법론이다.
- 유전 알고리즘은 생물학적 표현을 빌리며, 관련 용어는 [그림 9]에 표현하였다. '유전 자(gene)'는 변수(variable) 하나에 해당하는 임의의 해이다. '염색체(chromosome)'는 변수들로 이루어진 임의의 조합해이며, 임의의 조합해 집합을 세대(population)라 한다. '적합도(fitness)'는 임의의 조합해 인 염색체가 목적함수와 얼마나 적합한지를 나타내는 지표이다. 이 알고리즘의 한 세대는 대부분  $n$ 개로 정해진 조합해  $m$ 로 이루어진다.  $k$  세대를 거듭하며 적합도가 가장 높은 유전자 조합인 염색체를 찾는 것이 목적이다.



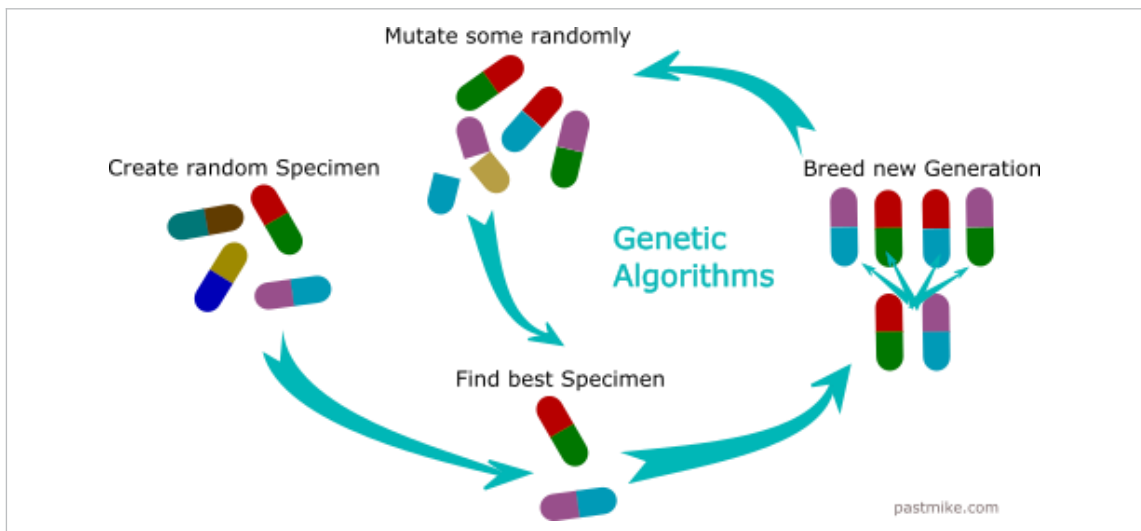
[그림 9] 유전 알고리즘 요소

### • 유전 알고리즘 과정

- 이 알고리즘은 초기해 집합인 초기 세대(initial population)에서 적합도를 바탕으로 우수한 염색체를 선택(selection) 한다. 선택 방법에는 룰렛 선택법, 기대치 선택법, 순위 선택법, 토너먼트 선택법 등이 있다.
- 선택된 염색체 중 두 개체를 뽑아, 미리 설정된 교배율에 따라 재구성되며, 이 과정을 재조합(crossover)이라 한다. 이때, 서로 결합하여 새로운 개체를 만드는 과정에서 교

배 위치는 다양할 수 있다. 교배율은 두 개체가 서로 유전자를 주고받을 비율이다. 교배율이 너무 높으면 새로운 개체가 개체군에서 빨리 나타나고 성능이 좋은 개체가 무시될 수 있다. 반면 교배율이 너무 낮으면 탐색이 부진할 수 있다. 따라서 적절한 값을 설정해야 한다.

- 변이(mutation)는 설정된 돌연변이율에 따라 개체 내 임의의 부분을 선택, 변환시키는 것이다. 이때 확률값이 너무 작으면 개체의 다양성이 줄어들어 극소 최적값으로 수렴할 가능성이 있다. 반면 너무 크게 되면 임의적 탐색이 되므로 적절하게 설정해야 한다.
- 유전 알고리즘은 앞서 설명한 선택과 재구성, 변이를 반복(generation)하며 해를 탐색한다. [그림 10]은 유전 알고리즘을 도식화한 것이다. 이 알고리즘은 단일 해가 아닌 잠재적 임의의 해 집합 (즉, 세대)을 이용한 확률적 탐색기법을 사용하므로 전역적 최적해 근사를 도출할 수 있다. 또한, 최적화 함수는 정확도, 편차 등 수치 해석적 의미를 내포하고 있지 않기 때문에 성능 지표 또는 평가 함수를 설계자 의도에 따라 쉽게 정의할 수 있는 장점이 있다.



[그림 10] 유전 알고리즘 도식화

### 3) 유전 알고리즘 구축 절차

#### • 목적함수와 제약조건 정립 및 수식화

- 유전 알고리즘을 적용하기 위해 문제에 적절한 목적함수와 제약조건을 수식화해야 한다. 본 가이드북에서 다룰 예제는 item 생산 요청에 따른 장비 할당 및 생산량 최적화 문제이며, 변수는 다음과 같다.

### [변수집합]

$T$  = 생산해야 하는 기한의 날짜 집합

$I$  = 생산해야 하는 item 종류

$J$  = 사용 가능한 NC machine 종류

### [변수]

$x_{i,j,t}$  = 날짜  $t$ 에, item  $i$ 를 NC machine  $j$ 가 생산 요구량 중 생산해야 하는 백분율  
(범위: 0~1)

$c_{i,t}$  = 날짜  $t$ 에, item  $i$ 가 미생산 될 때 발생하는 비용

$p_{i,t}$  = 날짜  $t$ 에, item  $i$ 가 긴급 생산이 필요한 경우 1, 그렇지 않으면 0

$d_{i,t}$  = 날짜  $t$ 에 item  $i$ 마다 생산되어야 하는 요구량

$m_{i,j,t}$  = 날짜  $t$ 에 NC machine  $j$ 가 item  $i$ 를 생산할 수 있는 능력

- 목적함수와 제약조건은 다음과 같다.

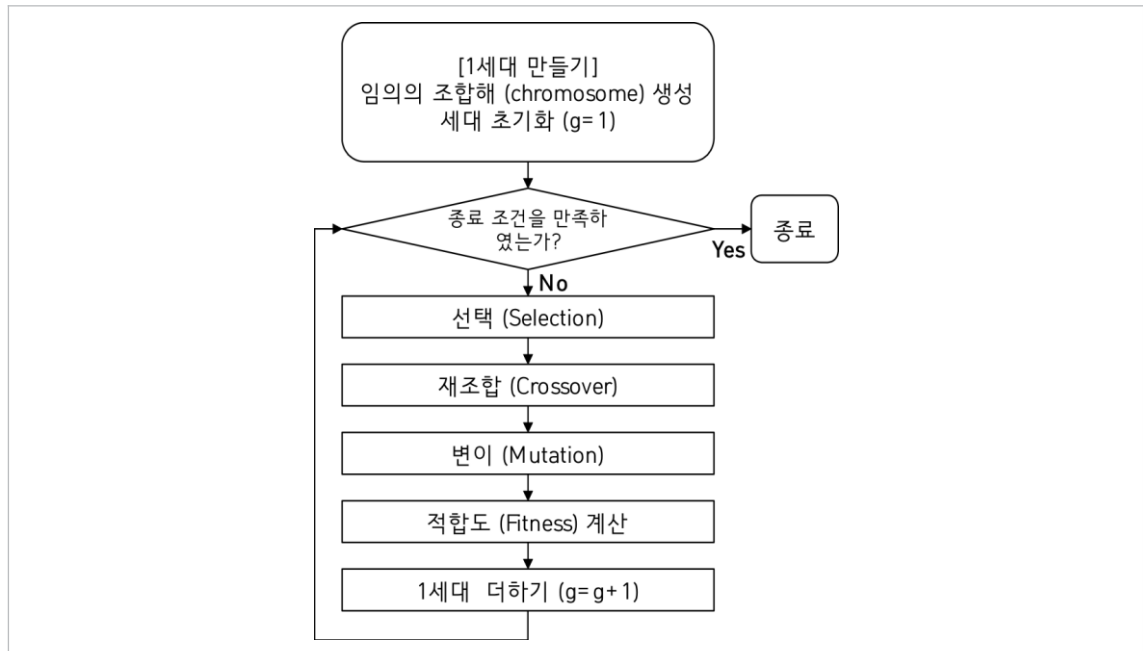
$$\min \sum_{t=T} \sum_{i=I} (d_{i,t} - \sum_{j=J} \lfloor x_{i,j,t} d_{i,t} \rfloor) c_{i,t} p_{i,t} \quad \dots (1)$$

$$s.t. \quad 600(\min) - \sum_{i=I} m_{i,j,t} x_{i,j,t} \geq 0, \text{ for all } j \in J, t \in T \quad \dots (2)$$

- (1) 식은 기한 내에 만들지 못해 발생한 비용에 대한 페널티를 최소화하는 목적함수이다.
- (2) 식은 기계가 만들 수 있는 최대 용량에 대한 제약조건이다. 본 가이드북에서는 한 기계는 하루 최대 600분만 작동이 가능하다고 가정하였다.

## • 유전 알고리즘 순서도

- 해당 유전 알고리즘은 [그림 11]과 같이 진행된다.



[그림 11] 유전알고리즘 순서도

## • Step 0: 하이퍼파라미터 할당

- 유전 알고리즘에 사용되는 하이퍼파라미터는 [표 8]과 같다.

[표 8] 필요한 하이퍼파라미터 종류

하이퍼파라미터	의 미
n_pop	한 세대를 구성하는 염색체 수
n_iter	반복할 세대 수, 유전 알고리즘을 반복할 수
r_cross	교배율, 부모 염색체로부터 교배할 유전자 비율
r_mut	변이율, 새로 생긴 염색체에서 임의로 변경할 유전자 비율

## • Step 1: 초기 임의의 조합해 생성

- 유전 알고리즘을 위한 가장 첫 번째 단계는 초기 해를 만드는 것이다. 유전 알고리즘 임의의 해는 실수코딩으로 만들어진다. [그림 9]는 이를 그림으로 표현한 것이다. 초기 해는  $n$ 개 유전자로 이루어진  $m$ 개 염색체로 구성됨을 가정하였을 때, 다음과 같이 표현된다.

$$\begin{aligned}
 s_1(1) &= \mathbf{x}^T(1) = (x_{11}(1) \ x_{12}(1) \ \cdots \ x_{1j}(1) \ \cdots \ x_{1n}(1)) \\
 s_2(1) &= \mathbf{x}^T(1) = (x_{21}(1) \ x_{22}(1) \ \cdots \ x_{2j}(1) \ \cdots \ x_{2n}(1)) \\
 &\vdots \\
 s_m(1) &= \mathbf{x}^T(1) = (x_{m1}(1) \ x_{m2}(1) \ \cdots \ x_{mj}(1) \ \cdots \ x_{mn}(1))
 \end{aligned}$$

- k 세대의 구성은 다음과 같다.

$$\begin{aligned} s_1(k) &= \mathbf{x}^T(k) = (x_{11}(k) \ x_{12}(k) \cdots x_{1j}(k) \cdots x_{1n}(k)) \\ s_2(k) &= \mathbf{x}^T(k) = (x_{21}(k) \ x_{22}(k) \cdots x_{2j}(k) \cdots x_{2n}(k)) \\ &\vdots \\ s_m(k) &= \mathbf{x}^T(k) = (x_{m1}(k) \ x_{m2}(k) \cdots x_{mj}(k) \cdots x_{mn}(k)) \end{aligned}$$

- k 세대의 집단은 다음과 같이 정의된다.

$$P(k) = \{s_1(k) s_2(k) \cdots s_i(k) \cdots s_m(k)\}$$

- 여기서  $s_i(k) = (x_{i1}(k) \ x_{i2}(k) \cdots x_{ij}(k) \cdots x_{in}(k))$ 는 i번째 염색체이며,  $x_{ij}(k)$ 는 j번째 염색체의 j번째 요소이다. n은 염색체 벡터의 차원이다. m은 한 세대 집단의 크기이며, 세대와는 상관없이 고정된다. 또한, i번째 염색체의 j번째 요소  $x_{ij}(k)$ 는  $[x_j^L, x_j^U]$ 내에서 임의로 발생하는 실숫값으로 초기화된다. 이때  $x_j^L$ 은 하한 범위 값이며,  $x_j^U$ 는 상한 범위 값이다. 본 가이드북에서는 0이  $x_j^L$ 이며, 1이  $x_j^U$ 이다.

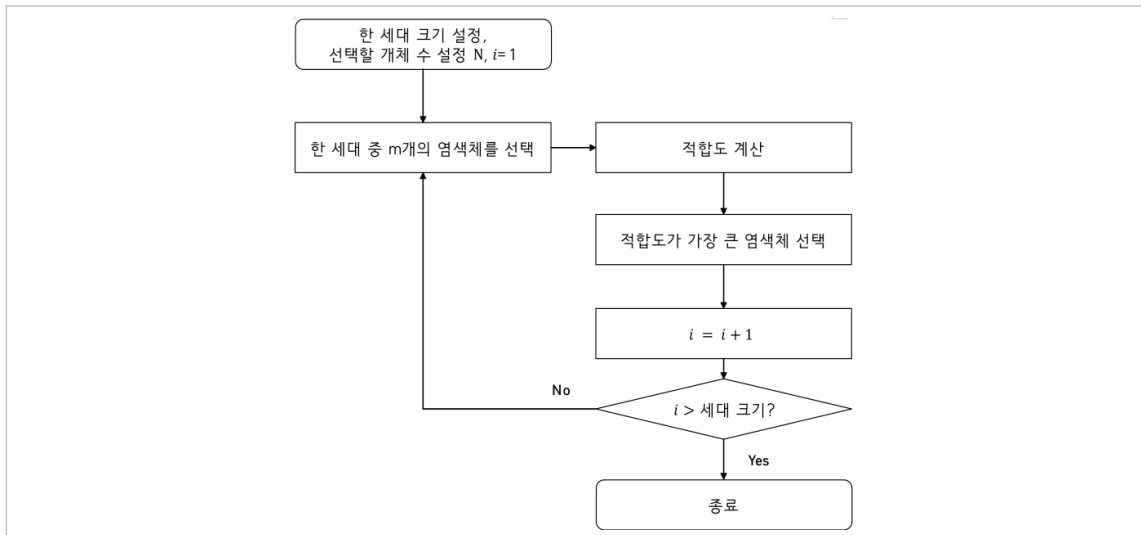
#### • Step 2: 초기 임의의 조합해 적합도 계산 (fitness)

- 다음 세대로 유전자를 전달하기 위해 초기 해 집합인  $P(0)$ 의 모든 염색체의 적합도를 계산한다.

$$\text{Fitness}(k) = \{\text{fit}_1(k) \ \text{fit}_2(k) \cdots \text{fit}_{\text{npop}}(k)\}$$

#### • Step 3: 적합도 기준 조합해 선택 (select)

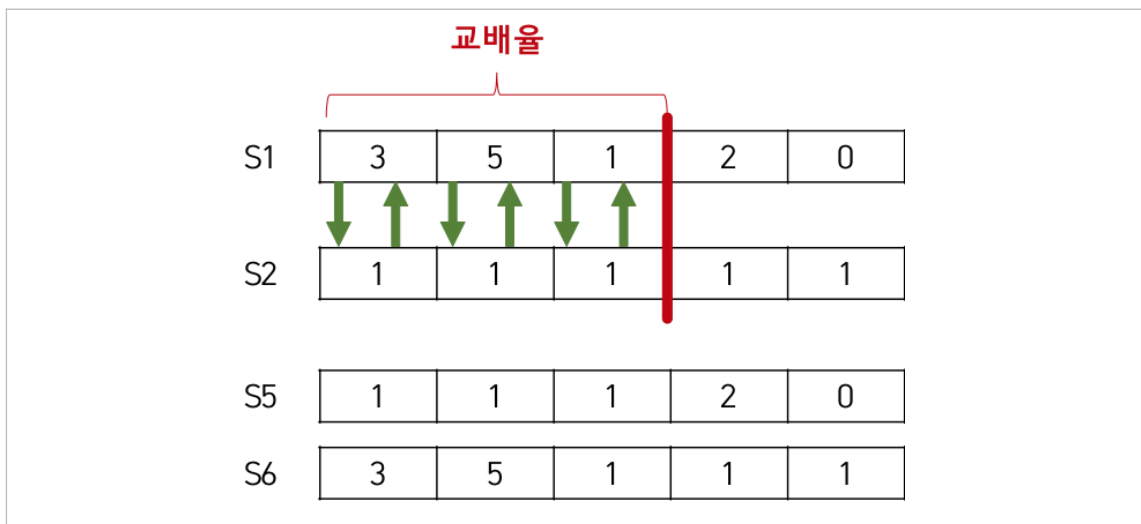
- 선택 단계는 적응도에 따라 생존할 염색체를 선택하는 과정이다. 선택하는 기준은 다양하며, 본 가이드북에서는 토너먼트 선택법을 사용하였다. 토너먼트 선택법은 염색체 집합 중 무작위로 선택, 적합도가 가장 높은 개체를 선택하는 방법이다. [그림 12]는 토너먼트 선택법 알고리즘이다.



[그림 12] 토너먼트 선택법

#### • Step 4: 선택된 조합해의 자손해 생성 (재조합, crossover)

- 다음 세대로 넘길 때, 해의 다양성 생성을 위해 일정 부분 유전자를 조합하여 자손을 생산한다. Step 3 에서 선택된 염색체 중 임의로 두 염색체를 선택한 후 미리 정해둔 교배율만큼 교차시킨다. [그림 13]은 재조합 과정을 도식화한 것이다.



[그림 13] 재조합 과정

#### • Step 5: 돌연변이 연산 (mutation)

- 이 과정 또한 다양한 해를 고려하기 위한 과정이다. 재조합 과정 이후 변이율만큼 임의의 유전자를 변경한다.



- **Step 6: 생성된 자손의 적합도 계산 (fitness)**

- 재조합 과정과 돌연변이 연산 이후 구성된 다음 세대의 모든 염색체의 적합도를 계산한다. 이후 Step 3 ~ Step 6을 반복한다.

- **Step 7: 종료 조건 판별**

- 미리 정해둔 세대 수에 도달하였거나, 특정 종료 조건이 만족 되면 종료한다.

### 3. 분석 체험

#### 1) 필요 SW, 패키지 설치 방법 및 절차 가이드

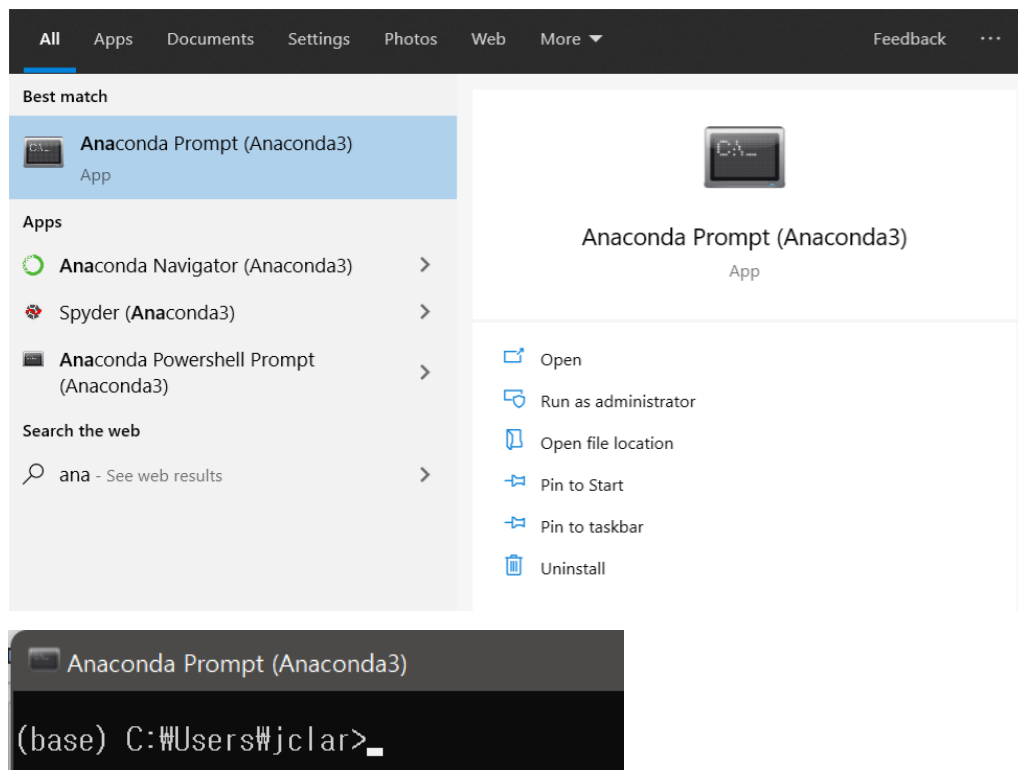
##### ▶ 필요 SW 설치

- 프로그래밍 언어인 파이썬(Python)을 사용하며 개발 환경으로는 주피터 노트북 (Jupyter Notebook)을 활용한다. 관련 SW 설치는 3. 분석 [분석 환경 구축을 위한 설치 가이드]를 참고하면 된다.

##### ▶ 가상환경 구축 가이드

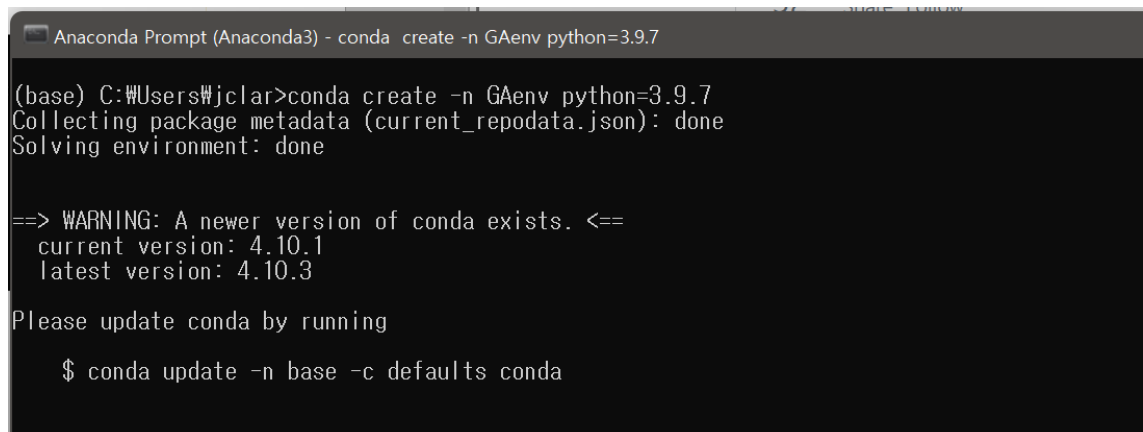
- 3. 부록 [분석 환경 구축을 위한 설치 가이드]에서 설치한 아나콘다(Anaconda)에서 파이썬의 독립적인 가상의 작업 환경을 구축한다. 파이썬의 모듈은 다양한 버전이 존재하기 때문에 충돌이 일어나기 쉬우며, 이러한 충돌을 방지하기 위해 실습마다 독립적인 가상 환경 구축을 권장한다. 가상 환경을 사용하기 위해서는 환경을 구축하고 실행(활성화)해야하고, 실습이 끝난 이후에는 환경을 비활성화 시켜야 한다. 주피터 노트북에서 가상 환경을 실행하는 구체적인 순서 및 실행 방법은 아래와 같다.

## - 아나콘다 프롬프트 실행



## - 가상 환경 생성

```
# conda create -n 가상환경이름 python=버  
전 conda create -n GAenv python=3.9.7
```



```
...  
Proceed ([y]/n)? y # 해당 문구가 뜨면 'y' 를 입력한다.  
...
```

```
done
#
# To activate this environment, use
#
#     $ conda activate GAenv
#
# To deactivate an active environment, use
#
#     $ conda deactivate

(base) C:\Users\jclar>
```

## - 가상 환경 실행

가상 환경이 활성화되면 아래 그림처럼 명령 프롬프트 앞에 (가상 환경 이름)이 표시된다.

```
# conda activate 가상환경이름
conda activate GAenv
```

```
(base) C:\Users\jclar>conda activate GAenv
(GAenv) C:\Users\jclar>
```

## - 분석이 끝난 후에는 가상 환경을 종료한다.

```
conda deactivate
```

```
(GAenv) C:\Users\jclar>conda deactivate
(base) C:\Users\jclar>
```

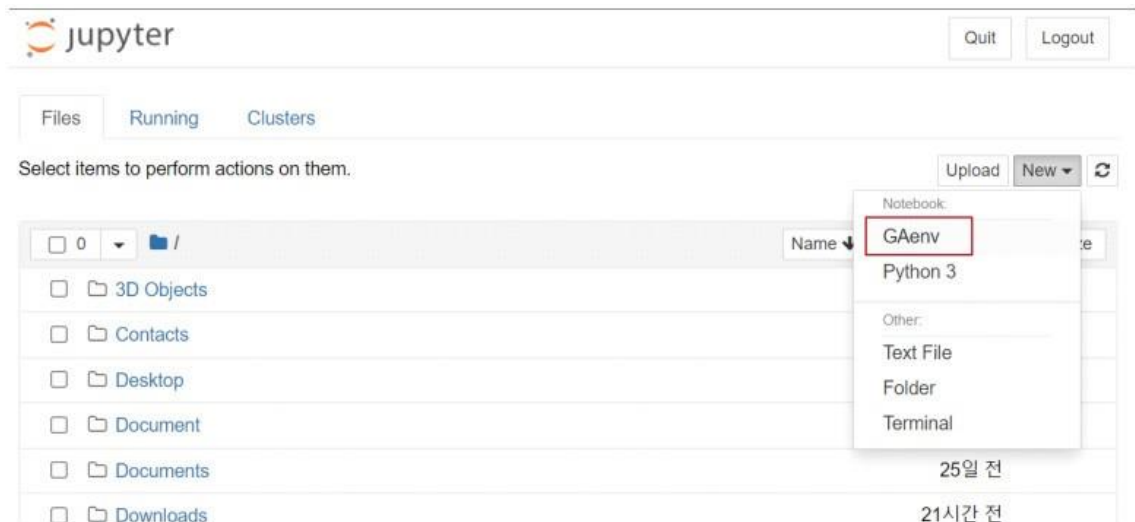
## - 주피터 노트북에서 가상 환경 실행

주피터 노트북에서 가상 환경을 실행하기 위해 가상 환경 비활성화(종료) 상태에서 아래의 코드를 작성한다. 주피터 노트북 실행창에서 빨간 박스처럼 생성한 가상 환경이 있는 것을 확인할 수 있다. 해당 가상 환경을 선택 후 실습을 진행한다.

```
pip install ipykernel

# python -m ipykernel install --user --name 가상환경이름 --display-name "가상환경이름"
python -m ipykernel install --user --name GAenv --display-name "GAenv"

# jupyter notebook 실행
jupyter notebook
```



## ▶ 패키지 설치 가이드

- 패키지 설치 전 파이썬 관련 용어 정리

모듈 (module)	<ul style="list-style-type: none"> <li>- 함수나 변수 또는 클래스를 모아 놓은 파일</li> <li>- 파이썬 확장자(.py) 파일에 특정 함수, 변수, 클래스 등이 저장된 코드이다.</li> </ul>
패키지 (package)	<ul style="list-style-type: none"> <li>- 여러 모듈의 집합</li> <li>- 하나의 디렉토리에 놓여진 모듈들의 집합을 의미하며, 일반적으로 <code>__init__.py</code>라는 패키지 초기화 파일이 존재한다.</li> <li>- 예) NumPy: 수치해석 패키지, pandas: 데이터 분석 패키지</li> </ul>
라이브러리 (library)	<ul style="list-style-type: none"> <li>- 여러 패키지의 집합</li> <li>- 파이썬을 설치할때 기본적으로 설치되는 라이브러리를 표준라이브러리라고 한다. 파이썬 공식이 아닌 외부에서 개발한 모듈과 패키지를 묶어 외부 라이브러리라고 한다.</li> <li>- 예) matplotlib: 데이터 시각화를 생성하기 위한 표준 라이브러리</li> </ul>
프레임워크 (framework)	<ul style="list-style-type: none"> <li>- 라이브러리의 집합</li> <li>- 목적에 맞게 코드를 작성할 수 있는 기본 아키텍처이며, 필수 구성 요소를 제공한다.</li> </ul>

[그림 14] 파이썬 관련 개념도

## ▶ 필요 패키지 설치 방법

- 분석 환경인 주피터 노트북 내에서 데이터 분석 및 AI 분석모델 적용을 위한 패키지를 설치한다. `pip install <패키지 이름>` 을 입력하여 원하는 라이브러리(혹은 패키지)를 설치할 수 있다.

```
!pip install pandas
!pip install numpy
!pip install tqdm
!pip install matplotlib
```

- 이미 설치되어 있는 패키지 외에 신규 설치가 필요한 패키지들이 자동으로 설치된다.

```
Requirement already satisfied: pandas in /home/ubuntu/anaconda3/lib/python3.6/site-packages (0.23.0)
Requirement already satisfied: python-dateutil>=2.5.0 in /home/ubuntu/anaconda3/lib/python3.6/site-packages (from pandas) (2.7.3)
Requirement already satisfied: pytz>=2011k in /home/ubuntu/anaconda3/lib/python3.6/site-packages (from pandas) (2018.4)
Requirement already satisfied: numpy>=1.9.0 in /home/ubuntu/anaconda3/lib/python3.6/site-packages (from pandas) (1.19.5)
Requirement already satisfied: six>=1.5 in /home/ubuntu/anaconda3/lib/python3.6/site-packages (from python-dateutil>=2.5.0->pandas) (1.15.0)
```

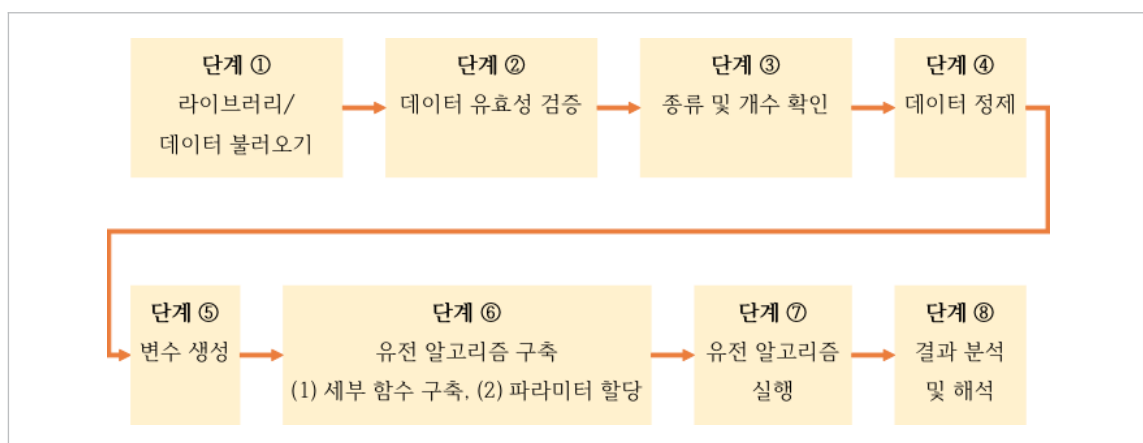
- 설치된 패키지들의 전체 목록을 확인할 수 있다.

```
!pip list
```

Package	Version
abs1-py	0.14.1
alabaster	0.7.10
alembic	1.4.1
anaconda-client	1.6.14
anaconda-navigator	1.8.7
anaconda-project	0.8.2

## 2) 분석 단계별 프로세스

- 크게 아래의 흐름으로 데이터셋을 준비하고 모델을 구현할 수 있다.



[그림 15] 유전 알고리즘 적용을 위한 모델 구현 과정

## [단계 ①] 라이브러리/데이터 불러오기

### ①-1. 라이브러리 불러오기 가이드

```
import random
from numpy.random import randint
from numpy.random import rand
import pandas as pd
import datetime
import numpy as np
from tqdm import tqdm
import matplotlib.pyplot as plt
```

[코드 1] 필요 라이브러리 불러오기

### ①-2. 데이터 불러오기 가이드

- 레이블 된 데이터를 불러온다. csv 파일을 불러올 때에는 pandas의 'read\_csv('파일 경로')' 기능을 사용한다.

```
machine_info = pd.read_csv('machine_info.csv')
order_info = pd.read_csv('order_info.csv')
```

[코드 2] 클래스 변수가 존재하는 데이터 불러오기

- 추후 분석 용이성을 위해 'rename' 기능을 사용하여 열 이름을 변경한다.

```
# 전처리 (1) column name 변경
order_info = order_info.rename(columns={
    '영업납기':'time',
    '중산도면':'item',
    '단가':'cost',
    '수량':'qty',
    '선급':'urgent'
})

machine_info = machine_info.rename(columns={
    'JSDWG':'item',
    'MCNO':'machine',
    'AVG_CT':'capacity'
})
```

[코드 3] 변수명 수정하기

- urgent 변수 특성상, 실제 긴급 생산이 필요한 경우 자세한 사항이 기재 돼 있고, 그렇지 않은 경우는 비어있다(NaN). 분석 용이성을 위해 긴급 생산 요건이 있는 경우 1, 그렇지 않은 경우는 0으로 채운다. 'fillna()' 함수는 Nan 값에 괄호 안 특정 값을 넣을 수 있으며, 'fillna(0)'로 NaN 값에 0을 채운다. 그다음 0이 아닌 곳은 1로 치환한다.
- time 변수는 'apply()' 함수로 datetime type을 부여한다.



```
# 전처리 (2) 분석을 위한 데이터 처리
order_info['urgent']=order_info['urgent'].fillna(0)
for i in range(len(order_info)):
    if order_info.loc[i,'urgent']!=0:
        order_info.loc[i,'urgent']=1

# 전처리 (3) time type 부여
order_info['time'] = order_info['time'].apply(lambda x: datetime.datetime.strptime(x, '%Y-%m-%d'))
```

[코드 4] urgent와 time 변수 전처리

## [단계 ②] 데이터 유효성 검증

- 데이터 유효성 검사는 [부록 4]에 수록함
- 최종 결과는 [표 9]와 같으며, 데이터 유효성 검증이 부족하다면 이 부분이 만족 될 때까지 데이터 자체를 개선해야 한다.

구 분	품질지수	가중치	가중치 지수	오류율
완정성 (누락)	99.99%	60%	59.99%	0.01%
유일성 (중복)	99.16%	10%	9.92%	0.08%
유효성 (유효)	100%	10%	10%	0%
일관성 (표현)	100%	10%	10%	0%
정확성 (실제)	-	-	-	-
무결성	83.35%	10%	8.34%	1.66%
품질지수		100%	98.25%	1.75%

[표 9] 데이터 유효성 검증 결과

## [단계 ③] 데이터 종류 및 개수 확인

- ③-1. 데이터 종류에 따른 데이터 예시 확인
  - '.head()' 함수는 데이터 프레임의 상위 5개를 확인할 수 있으며, 이를 이용하여 데이터 형태를 확인한다.
  - 이 과정에서 각 열의 데이터 타입을 파악한다.

```
order_info.head()
```

	time	item	cost	qty	urgent
0	2021-05-13	K04033	25870	318	1
1	2021-05-24	K04031	16229	383	1
2	2021-05-30	051718	8333	19	0
3	2021-06-03	056984	36533	4	0
4	2021-06-18	057791	45500	196	1

[코드 5] order\_info의 유일성 상세 확인

```
machine_info.head()
```

	item	machine	capacity
0	050060	433.0	1.08
1	050093	404.0	7.13
2	050093	408.0	4.67
3	050093	410.0	4.50
4	050093	416.0	3.92

[코드 6] machine\_info의 유일성 상세 확인

### ③-2. 데이터 종류에 따른 데이터 개수 확인

- '.value\_counts()' 함수는 데이터 행의 개수를 확인할 수 있다. 마지막 열은 해당 행 데이터가 몇 개 있는지를 의미하며, 결과는 내림차순으로 정렬된다. 마지막 열이 1이 아닌 경우 중복임을 알 수 있다.

```
order_info.value_counts()
```

```

: time      item      cost      qty      urgent
2021-02-10  S00341    2990      500      0          1
2021-07-01  057387   16100     144      1          1
2021-07-07  K03894   60000      46      1          1
2021-07-06  K03899   22005     396      1          1
                059988  166250     24      1          1
...
2021-06-17  053695   33984      6       0          1
2021-06-15  Z00807   10450     300      0          1
                K03115   26719      30      0          1
                28       0          1
2021-09-15  Z00807   10450     325      0          1
Length: 127, dtype: int64

```

[코드 7] order\_info의 value\_counts() 확인

```
machine_info.value_counts()
```

```

item      machine      capacity
050060    0433         1.08          1
K00347    0404        11.05          1
K00337    0438        40.00          1
                0437         7.13          1
                0410         8.77          1
...
058398    0408         4.61          1
                0404         4.17          1
058396    0438         7.08          1
                0434         6.30          1
Z01415    0409         9.50          1
Length: 21149, dtype: int64

```

[코드 8] machine\_info의 value\_counts() 확인

### ③-3. 데이터 특성 파악

- '.describe()' 함수는 데이터구조 및 변수 모양을 확인할 수 있다.

```
order_info.describe()
```

	cost	qty
count	127.000000	127.000000
mean	45204.133858	114.110236
std	81277.077503	140.706450
min	2990.000000	2.000000
25%	11551.000000	20.000000
50%	21160.000000	48.000000
75%	35816.500000	169.000000
max	481190.000000	716.000000

[코드 9] order\_info의 describe() 확인

```
machine_info.describe()
```

	machine	capacity
count	21149.000000	21151.000000
mean	420.686132	10.441903
std	12.071250	18.304675
min	401.000000	0.050000
25%	409.000000	3.200000
50%	424.000000	5.500000
75%	433.000000	10.500000
max	442.000000	930.000000

[코드 10] machine\_info의 describe() 확인

- order\_info 데이터 경우, datetime 관련 열이 있으므로 월별 생산 주문을 확인하였다.

```

date_start = [ '2021-01-01',
                '2021-02-01',
                '2021-03-01',
                '2021-04-01',
                '2021-05-01',
                '2021-06-01',
                '2021-07-01',
                '2021-08-01',
                '2021-09-01',
              ]
for i in range(len(date_start)-1):
    check_df = order_info[ (order_info['time'] > date_start[i] & (order_info['time'] < date_start[i+1])) ]
    print(f'{date_start[i]} ~ {date_start[i+1]} 생산 주문 건 수: {len(check_df)}')

```

```

2021-01-01 ~ 2021-02-01 생산 주문 건 수: 0
2021-02-01 ~ 2021-03-01 생산 주문 건 수: 2
2021-03-01 ~ 2021-04-01 생산 주문 건 수: 1
2021-04-01 ~ 2021-05-01 생산 주문 건 수: 1
2021-05-01 ~ 2021-06-01 생산 주문 건 수: 8
2021-06-01 ~ 2021-07-01 생산 주문 건 수: 63
2021-07-01 ~ 2021-08-01 생산 주문 건 수: 40
2021-08-01 ~ 2021-09-01 생산 주문 건 수: 4

```

[코드 11] order\_info의 생산 일정 확인

## [단계 ④] 데이터 정제 (전처리)

### ④-1. 불필요한 데이터 제거 가이드

- '.dropna()' 함수는 값이 없는 행 (axis=0) 또는 열 (axis=1)을 제거해 주는 함수이다. 'in place=True' 옵션은 따로 데이터프레임을 지정해 저장하지 않고 바로 적용된다. 본 분석에서는 모든 데이터 (machine\_info.csv, order\_info.csv)에 대해 값이 없는 행을 모두 제거하였다.

```

# nan값 처리
n_before_preprocess = len(machine_info)
machine_info.dropna(axis=0, inplace=True)
n_after_preprocess = len(machine_info)
print(f'machine_info : nan 값 제거 ({n_before_preprocess} row --> {n_after_preprocess} row, {round(((n_before_preprocess-n_after_preprocess)/n_before_preprocess*100),3)}% 삭제)')

n_before_preprocess = len(order_info)
order_info.dropna(axis=0, inplace=True)
n_after_preprocess = len(order_info)
print(f'order_info : nan 값 제거 ({n_before_preprocess} row --> {n_after_preprocess} row, {round(((n_before_preprocess-n_after_preprocess)/n_before_preprocess*100),3)}% 삭제)')

```

```

machine_info : nan 값 제거 (21151 row --> 21149 row, 0.009% 삭제)
order_info : nan 값 제거 (127 row --> 127 row, 0.0% 삭제)

```

[코드 12] order\_info의 유일성 상세 확인

#### ④-2. 데이터 오류 처리 : unique key 정보의 유일성 보장을 위한 수정

- machine\_info 데이터는 item과 machine 열이 unique key 값이다. unique key에 대해 '.value\_counts()' 함수로 확인했다. 이 경우 모든 key 값이 1이므로 수정할 사항이 없다.

```
# unique 처리 machine_info[['item','machine']].value_counts().head()
```

```
item machine
050060 0433      1
K00347 0404      1
K00337 0438      1
        0437      1
        0410      1
dtype: int64
```

[코드 13] machine\_info의 unique key 값의 유일성 확인

- order\_info 데이터는 time, item, cost, urgent 열이 unique key 값이다. 유일성을 확인했을 때, 1이 아닌 값을 발견하였다.

```
order_info[['time','item','cost', 'urgent']].value_counts().head()
```

```
time item cost urgent
2021-07-09 057386 24418 1 2
           057387 17794 1 2
2021-06-15 K03115 26719 0 2
2021-02-10 S00341 2990 0 1
2021-07-07 K03894 60000 1 1
dtype: int64
```

[코드 14] order\_info의 unique key 값의 유일성 확인

- 자세한 정보는 다음과 같다. 같은 날 같은 item을 중복으로 주문했고, 합쳐지지 않았다.

```
order_info[(order_info['time']=='2021-06-15')&(order_info['item']=='K03115')]
```

	time	item	cost	qty	urgent
32	2021-06-15	K03115	26719	30	0
34	2021-06-15	K03115	26719	28	0

[코드 15] order\_info의 유일성 상세 확인 (1)

- '.groupby().sum()' 기능은 특정 열로 그룹을 묶은 후 나머지 열은 모두 합을 하는 기능이다. 즉, 'time', 'item', 'cost', 'urgent'의 값이 모두 같은 행의 경우 중복 값으로 판단, 해당 행들은 합치고 order\_info에 저장한다.

```
order_info = order_info.groupby(['time', 'item', 'cost', 'urgent']).sum().reset_index()
```

[코드 16] order\_info 전처리 (1)

- 문제가 된 아이템을 확인해보았을 때, 중복으로 주문된 값이 옳게 합쳐짐을 확인하였다.

```
order_info[(order_info['time']=='2021-06-15') & (order_info['item']=='K03115')]
```

	time	item	cost	urgent	qty
34	2021-06-15	K03115	26719	0	58

[코드 17] order\_info의 유일성 상세 확인 (2)

#### ④-3. 모델 적용을 위한 데이터 선택

- 본 가이드북에서는 2021년 상반기 주문데이터에 대해 생산계획 최적화를 할 것이다. 2021-01-01 이후부터 2021-05-31까지 데이터를 filtered\_order\_info로 변수를 할당했다.

```
# 1월 6월 주문 고려 filtered_order_info
er_info = order_info[
    (order_info['time'] > '2021-01-01')
    & (order_info['time'] < '2021-06-01')
]
filtered_order_info
```

	time	item	cost	urgent	qty
0	2021-02-10	S00341	2990	0	500
1	2021-02-18	052996	20000	0	2
2	2021-03-10	S00341	2990	0	500
3	2021-04-10	S00341	2990	0	500
4	2021-05-10	S00341	2990	0	398
5	2021-05-13	066157	18500	1	208
6	2021-05-13	K04033	25870	1	318
7	2021-05-24	K04031	16229	1	383
8	2021-05-30	051718	8333	0	19
9	2021-05-30	052996	20000	0	2
10	2021-05-30	K04101	15640	0	15
11	2021-05-31	S00271	4000	0	150

[코드 18] order\_info의 unique key 값의 유일성 확인



- 'merge()' 함수는 pandas 내장 함수로, 특정 열로 두 데이터 프레임을 연결해준다. 최종 분석데이터는 filtered\_order\_info와 machine\_info 데이터를 item 열을 기준으로 연결한 결과값이다.

- 'to\_csv()' 함수는 pandas 내장 함수로, 데이터 프레임을 CSV로 저장한다.

```
dataset = pd.merge(filtered_order_info,machine_info,on='item',how='inner')
dataset.to_csv('dataset.csv',index=False)
```

[코드 19] 분석을 위한 데이터셋 저장

## [단계 ⑤] 변수 생성

- 2.2.3) 유전 알고리즘 구축 절차에서 정의한 변수 집합과 변수를 코드화한다.
- 'set()' 메소드(method)는 list의 unique 한 객체를 집합 형태로 다루기 쉽게 처리하기 위해 만든 자료형이다. 본 가이드북에서는 원본 데이터셋에서 필요한 변수 집합을 추출한다.

### [변수집합]

$T$  = 생산해야 하는 기한의 날짜 집합  
 $I$  = 생산해야 하는 item 종류  
 $J$  = 사용 가능한 NC machine 종류

```
dataset = pd.read_csv('dataset.csv')
# variables information
T = list(set(dataset['time']))
I = list(set(dataset['item']))
J = list(set(dataset['machine']))
```

T	:	I	J
['2021-05-24',	:	['052996',	[404.0,
'2021-05-13',	:	'K04101',	405.0,
'2021-02-10',	:	'K04033',	407.0,
'2021-05-30',	:	'051718',	408.0,
'2021-05-10',	:	'066157',	409.0,
'2021-04-10',	:	'S00341',	410.0,
'2021-02-18',	:	'S00271',	412.0,
'2021-03-10',	:	'K04031',	416.0,
'2021-05-31']	:		421.0,
			422.0,
			424.0,
			425.0,
			426.0,
			433.0,
			434.0,
			435.0,
			436.0,
			438.0,
			439.0,
			440.0]

[코드 20] 변수 형태 예시

- 본 가이드북에서는 특정  $i, j, t$ 에 해당하는 값이 필요하므로 변수는 dictionary 자료형으로 저장한다.
- 고려하고 있는 모든 변수 조합에 대해 변수를 할당한다. 데이터셋에서 특정 변수 조합에 해당하는 행을 temp\_dataset로 할당한다. 만약 temp\_dataset의 행의 개수가 0개라면 해당 조합의 변수값은 0으로 할당한다. 그렇지 않으면 데이터셋의 값을 할당한다.

## [변수]

cit = 날짜  $t$ 에, item  $i$ 가 미생산 될 때 발생하는 비용 (cost,  $c_{i,t}$ )

pit = 날짜  $t$ 에, item  $i$ 가 긴급 생산이 필요한 경우 1, 그렇지 않으면 0 (urgent,  $p_{i,t}$ )

dit = 날짜  $t$ 에 item  $i$ 마다 생산되어야 하는 요구량 (qty,  $d_{i,t}$ )

mijt = 날짜  $t$ 에 NC machine  $j$ 가 item  $i$ 를 생산할 수 있는 능력 (capacity,  $m_{i,j,t}$ )

```
cit = dict() #c_i,t
for i in I:
    for t in T:
        temp_dataset = dataset[
            (dataset['item']==i) & (
                dataset['time']==t)]
        if len(temp_dataset) !=0:
            value = list(set(temp_dataset['cost']))[0]
            cit[i, t] = value
        else:
            cit[i, t] =0
pit = dict() #p_i,t
for i in I:
    for t in T:
        temp_dataset = dataset[
            (dataset['item']==i) & (
                dataset['time']==t)]
        if len(temp_dataset) !=0:
            value = list(set(temp_dataset['urgent']))[0]
            pit[i, t] = value
        else:
            pit[i, t] =0
dit = dict() #d_i,t
for i in I:
    for t in T:
        temp_dataset = dataset[
            (dataset['item']==i) & (
                dataset['time']==t)]
        if len(temp_dataset) !=0:
            value = list(set(temp_dataset['qty']))[0]
            dit[i, t] = value
        else:
            dit[i, t] =0
mijt = dict() #m_i,j,t
for i in I:
    for j in J:
        temp_dataset = dataset[
            (dataset['item']==i)
```

```

&(dataset['machine']==j)]

if len(temp_dataset) !=0:
    value = list(set(temp_dataset['capacity']))[0]
    for t in T:
        mijt[i, j, t] = value
else:
    for t in T: mijt[i
    , j, t] =0

```

cit		pit	
{('052996', '2021-05-10'):	0,	{('052996', '2021-05-10'):	0,
('052996', '2021-05-30'):	20000,	('052996', '2021-05-30'):	0,
('052996', '2021-04-10'):	0,	('052996', '2021-04-10'):	0,
('052996', '2021-02-10'):	0,	('052996', '2021-02-10'):	0,
('052996', '2021-05-13'):	0,	('052996', '2021-05-13'):	0,
('052996', '2021-05-24'):	0,	('052996', '2021-05-24'):	0,
('052996', '2021-02-18'):	20000,	('052996', '2021-02-18'):	0,
('052996', '2021-05-31'):	0,	('052996', '2021-05-31'):	0,
('052996', '2021-03-10'):	0,	('052996', '2021-03-10'):	0,
('K04101', '2021-05-10'):	0,	('052996', '2021-03-10'):	0,
('K04101', '2021-05-30'):	15640,	('K04101', '2021-05-10'):	0,
('K04101', '2021-04-10'):	0,	('K04101', '2021-05-30'):	0,
('K04101', '2021-02-10'):	0,	('K04101', '2021-04-10'):	0,
('K04101', '2021-05-13'):	0,	('K04101', '2021-02-10'):	0,
('K04101', '2021-05-24'):	0,	('K04101', '2021-05-13'):	0,
('K04101', '2021-02-18'):	0,	('K04101', '2021-05-24'):	0,
('K04101', '2021-05-31'):	0,	('K04101', '2021-02-18'):	0,
('K04101', '2021-03-10'):	0,	('K04101', '2021-05-31'):	0,
('K04033', '2021-05-10'):	0,	('K04101', '2021-03-10'):	0,

dit		mijt	
{('052996', '2021-05-10'):	0,	{('052996', 404.0, '2021-05-10'):	0,
('052996', '2021-05-30'):	2,	('052996', 404.0, '2021-05-30'):	0,
('052996', '2021-04-10'):	0,	('052996', 404.0, '2021-04-10'):	0,
('052996', '2021-02-10'):	0,	('052996', 404.0, '2021-02-10'):	0,
('052996', '2021-05-13'):	0,	('052996', 404.0, '2021-05-13'):	0,
('052996', '2021-05-24'):	0,	('052996', 404.0, '2021-05-24'):	0,
('052996', '2021-02-18'):	2,	('052996', 404.0, '2021-02-18'):	0,
('052996', '2021-05-31'):	0,	('052996', 404.0, '2021-05-31'):	0,
('052996', '2021-03-10'):	0,	('052996', 404.0, '2021-03-10'):	0,
('K04101', '2021-05-10'):	0,	('052996', 405.0, '2021-05-10'):	0,
('K04101', '2021-05-30'):	15,	('052996', 405.0, '2021-05-30'):	0,
('K04101', '2021-04-10'):	0,	('052996', 405.0, '2021-04-10'):	0,
('K04101', '2021-02-10'):	0,	('052996', 405.0, '2021-02-10'):	0,
('K04101', '2021-05-13'):	0,	('052996', 405.0, '2021-05-13'):	0,
('K04101', '2021-05-24'):	0,	('052996', 405.0, '2021-05-24'):	0,
('K04101', '2021-02-18'):	0,	('052996', 405.0, '2021-02-18'):	0,
('K04101', '2021-05-31'):	0,	('052996', 405.0, '2021-05-31'):	0,
('K04101', '2021-03-10'):	0,	('052996', 405.0, '2021-03-10'):	0,

[코드 21] 변수 형태 예시

## [단계 ⑥] 유전 알고리즘 구축

### ⑥-1. 세부함수 구축 가이드

- GA 특성상, 모든 위치에서 같은 범위의 값을 가지는 것이 좋다. 본 가이드북에서는 특정 item이 시간 t에 요구되는 demand 값이 모두 다르다. 모델 특성을 고려하여 요구되는 demand의 백분율로 정규화하였다. 본 가이드북에서는 demand의 백분율을  $x_{ijt}$ 라 할당하였으며, 유전 알고리즘은  $x_{ijt}$ 의 최적 백분율 조합을 찾는다.
- $x_{ijt}$ 는 날짜 t에, item i를 NC machine j가 생산 요구량 중 생산해야 하는 백분율 (범위: 0~1)이다.

- 즉, 날짜 t에 item i를 NC machine j가 생산해야 하는 생산량은  $x_{ijt} \cdot d_{it}$ 이다.
- 'generation\_xijt()' 함수는 목적함수를 최적화하기 위한  $x_{ijt}$  조합을 임의로 생성하는 함수이다. 'random.uniform()' 함수는 특정 값 사이 실수를 임의로 추출하는 함수이다.  $x_{ijt}$ 는 0~1 사이 임의의 실숫값을 할당해야 하므로 'random.uniform(0,1)'을 사용하여 할당하였다. demand가 없는 변수에는 0을 할당하였다.

```
def generation_xijt():
    xijt = {}
    for i in I:
        for j in J:
            for t in T:
                if d_it[i, t] > 0:
                    xijt[i, j, t] = random.uniform(0, 1)
                else:
                    xijt[i, j, t] = 0
    xijt = decode(mijt, xijt)
    return xijt
```

[코드 22] 유전 알고리즘 함수 코드 (1)

- 다음은 빠른 계산을 위한 utility 함수이다. 변수는 해석하기 쉬운 dictionary 자료형으로 생성했지만, crossover 또는 mutation과 같이 index를 이용하는 경우를 위해 dictionary 자료형에서 list 자료형으로 변경하는 함수 'dict2list()'와 list 자료형에서 dictionary 자료형으로 변경하는 함수 'list2dict()'를 작성하였다.

```
def dict2list(xijt):
    return list(xijt.values())

def list2dict(bitstring, type='xijt'):
    if type == 'xijt':
        _keys = xijt_keys
    elif type == 'mijt':
        _keys = mijt_keys
    for idx, value in enumerate(bitstring):
        xijt[_keys[idx]] = value
    return xijt
```

[코드 23] 유전 알고리즘 함수 코드 (2)

- 'decode()' 함수는 임의의 해인  $x_{ijt}$  조합 중 범위 밖의 값을 0으로 조정해주는 함수이다.  $m_{ijt}$ 가 0인 것은 시간 t에, item i를 machine j로 만들 수 없다는 의미이므로  $x_{ijt}$  값을 0으로 할당 한다. decode() 함수는 이를 보정 해준다.

```
def decode(mijt, xijt):
    for j in J:
        for t in T:
            for i in I:
                if mijt[i, j, t] == 0:
                    xijt[i, j, t] = 0
    return xijt
```

[코드 24] 유전 알고리즘 함수 코드 (3)

- 'objective()' 함수는 임의의 해 조합인 xijt의 objective 값을 계산한다. 양의 u는 주어진 demand보다 적게 생산한 양이다. 음의 u는 주어진 demand보다 초과 생산한 양이며, 이 경우 좀 더 큰 페널티를 주었다.
- 다음은 목적함수 식이다. 우리의 목적은 페널티를 포함한 비용을 최소화하는 것이다.

$$\min \sum_{t=T} \sum_{i=I} (d_{i,t} - \sum_{j=J} \lfloor x_{i,j,t} d_{i,t} \rfloor) c_{i,t} p_{i,t}$$

```
def objective(xijt):
    uit = {}
    xijt = constraint_check(xijt) #[코드 26]에서 정의한 함수
    for i in I:
        for t in T:
            u = dit[i, t] - sum(round(xijt[i, j, t]*dit[i, t]) for j in J) if
            u >= 0:
                uit[i, t] = u
            else:
                uit[i, t] = abs(u)*10000000 objective
    = sum(uit[i, t]*cit[i, t]*pit[i, t] for i in I for t in T) return
    objective
```

[코드 25] 유전 알고리즘 함수 코드 (4)

- 'constraint\_check()'은 제약조건 관련 함수이다. 기계가 하루동안 총 생산할 수 있는 시간은 600분이며, 한 기계에 할당된 생산량이 600분 안에 만들 수 없다면 해당 아이템을 생산하지 않게 할당하는 함수이다.
- 다음은 제약조건 식이다.

$$600(\min) - \sum_{i=I} m_{i,j,t} \lfloor x_{i,j,t} d_{i,t} \rfloor \geq 0, \text{ for all } j \in J, t \in T$$

```
def constraint_check(xijt):
    for j in J:
        for t in T:
            check_value = sum(mijt[i, j, t]*round(xijt[i, j, t]*dit[i, t]) for i in I) <= 600
            if check_value == False:
                for i in I:
                    xijt[i, j, t] = 0
    return xijt
```

[코드 26] 유전 알고리즘 함수 코드 (5)

- 'selection()' 함수는 임의의 해 조합을 crossover 하기 위해 한 세대 조합해 중 가장 좋은 성능을 가진 조합을 찾는 과정이다. 'k'는 임의의 해에서 crossover로 넘길 변수 개수이며, 전체 임의의 해 조합(pop) 중 50%이다. 본 가이드북에서는 토너먼트 방법을 사용하였으며, 이는 임의로 뽑은 임의의 해 중 가장 좋은 해만을 다음으로 넘기는 것이다.

```
# tournament selection
def selection(pop, scores): #n_pop은 한 세대를 구성하는 염색체 수이며, 하이퍼파라미터 값
    k = round(n_pop*0.5)
    selection_ix = randint(len(pop))
    for ix in randint(0, len(pop), k-1):
        if scores[ix] < scores[selection_ix]:
            selection_ix = ix
    return pop[selection_ix]
```

[코드 27] 유전 알고리즘 함수 코드 (6)

- 'crossover()'는 'select()' 된 함수에서 두 쌍을 선택, 특정 비율 (r\_cross)만큼을 교차시켜 재조합하여 새로운 임의의 해를 만드는 함수이다.
- 본 가이드북 문제는 특정 i, j, t 만이 정수를 해를 가질 수 있으므로 (임의의 item i는 특정 날짜 t, 특정 machine j로만 생산 가능) 임의로 교차를 할 수 없다. 따라서 해를 가질 수 있는 index를 valid\_index 변수에 저장하고 (crossover\_(1)), 이 중 특정 비율만큼 index를 선택하여 교차시킨다. (crossover\_(2))

```
def crossover(p1, p2, r_cross):
    p1 = dict2list(p1)
    p2 = dict2list(p2)
    c1, c2 = p1.copy(), p2.copy()
    if rand() < r_cross:
        base_bitstring = dict2list(mjit)
        # crossover_(1)    valid_index
        = []
        for i in range(len(base_bitstring)):
            if base_bitstring[i]>0:
                valid_index.append(i)

        n = round(r_cross*len(valid_index))
        pt_0 = random.sample(valid_index, n)
        pt_1 = list(set(valid_index)-set(pt_0)) #
        crossover_(2)
        for i in pt_0:
            c1[i] = p2[i]
            c2[i] = p1[i]
        for i in pt_1:
            c1[i] = p2[i]
            c2[i] = p1[i]
    return [c1, c2]
```

[코드 28] 유전 알고리즘 함수 코드 (7)

- 'mutation()' 함수는 'crossover()' 이후 특정 비율 (r\_mut) 만큼 새로운 임의의 해 (mutation\_(1))로 교체(mutation\_(2))하는 함수로 다양한 임의의 해를 생성하는 역할을 한다.

```

def mutation(bitstring, r_mut):
    for i in range(len(bitstring)):
        if rand() < r_mut:
            base_bitstring = dict2list(mijt)
            valid_index = []
            for i in range(len(base_bitstring)):
                if base_bitstring[i]>0:
                    valid_index.append(i)

            n = round(r_mut*len(valid_index))
            # mutation_(1)
            pt_0 = random.sample(valid_index, n)
            # mutation_(2)
            for i in pt_0:
                bitstring[i] = random.uniform(0, 1)

```

[코드 29] 유전 알고리즘 함수 코드 (8)

- 'genetic\_algorithm()' 함수는 유전 알고리즘이며, 앞서 정의한 함수를 활용한다. 본 알고리즘은 100세대 이상 개선이 없다면 종료한다. (stop\_rule)

```

def genetic_algorithm(bounds, n_iter, n_pop, r_cross, r_mut):

    # utility
    log = []
    log_detail = []
    best_gen = 0
    # GA algorithm
    pop = [generation_xijt() for _ in range(n_pop)]
    best, best_eval = decode(mijt, pop[0]), objective(decode(mijt, pop[0]))
    print(best_eval)
    for gen in tqdm(range(n_iter)):
        decoded = [decode(mijt, p) for p in pop]
        # 임의의 해 집합에서 목적함수 값 계산
        하기 scores = [objective(d) for d in decoded] # 가장 좋은 해 선택
        for i in range(n_pop):
            if scores[i] < best_eval:
                best, best_eval = pop[i], scores[i]
                print(f'>best! {gen}, {scores[i]}')
                best_gen = gen
            else:
                # stop_rule
                if gen - best_gen > 100:
                    print('stop')
                    return [best, best_eval, log, log_detail]
        # 부모 해 집합 선택
        selected = [selection(pop, scores) for _ in range(n_pop)]
        log_detail.append((gen, objective(selected[0])))
        children = list()
        for i in range(0, n_pop, 2):
            p1, p2 = selected[i], selected[i+1]
            for c in crossover(p1, p2, r_cross):
                mutation(c, r_mut)
                children.append(list2dict(c))
        pop = children
        log.append((gen, best_eval))
    return [best, best_eval, log, log_detail]

```

[코드 30] 유전 알고리즘 함수 코드 (9)

## ⑥-2. 파라미터 할당

- 유전 알고리즘은 [표 8]에서 설명한 하이퍼파라미터를 설정해야 하며, 문제 유형에 따라, 계산 환경에 따라 사용자가 설정해야 한다. 또한, 파라미터 조합에 따라 최종 성능이 달라질 수 있어 주의해야 한다. 다음은 하이퍼파라미터 추천 범위이다.

하이퍼파라미터	추천 범위
n_iter	100 ~ ∞ (정수)
n_pop	10 ~ ∞ (짝수 정수)
r_cross	0 ~ 0.5 (실수)
r_mut	0 ~ 1 (실수)

- n\_iter는 최대 반복 계산 가능한 세대 수이다. 세대 수를 크게 하면 더 좋은 해를 얻을 수 있지만, 세대를 거듭할수록 개선되는 값이 계산 시간보다 의미 있지 않다면 적당한 종료 조건으로 종료를 해주는 것이 모델의 효율성을 높일 수 있다.
- n\_pop은 한 세대에서 고려하는 염색체 수이다. 본 가이드북 알고리즘에서는 짝수 정수 범위면 모두 가능하다. n\_pop의 크기가 작으면, 한 세대 계산이 빠르지만 다양한 해를 만들 수 없어 최적 해를 찾기 위한 반복되는 세대가 다소 요구될 수 있다. 반대로 n\_pop의 크기가 크면 한 세대 계산이 다소 요구되지만, 다양한 임의 해를 고려하므로 최적 해를 더 적은 세대에서 발견할 수 있다.
- r\_cross는 교차율이다. 부모로 선택된 두 임의의 해 조합의 교차율 비율만큼 교차하여 다양한 후보 해 집합을 만든다. 따라서 교차율이 크면 부모 해와 차이가 큰 자식 해를 후보로 얻을 수 있다. 최대로 교환 가능한 범위는 절반인 0.5이다. r\_cross값은 한 세대 계산 시간에 영향을 끼치지는 않는다. 다만 교차율 크기가 작다면, 더 좋은 임의 해를 발견할 가능성이 적어, 더 많은 세대 수를 요구할 수 있다.
- r\_mut는 변이율이다. 이 비율은 crossover 과정 이후 생성된 새로운 자식 해에 임의의 해로 대체하는 비율이다. 이 비율 또한, r\_cross 파라미터와 같이 값이 너무 작으면 다양성을 얻을 수 없어 개선되기가 힘들고, 값이 너무 크면 너무 임의성에 기대게 된다. 세대 계산에 영향을 끼치지는 않지만 다양한 후보 해를 생성에 영향을 끼치므로 적당한 값을 할당해 주어야 한다.
- 하이퍼파라미터 실험을 위한 n\_iter, n\_pop, r\_cross, r\_mut 값조합을 hyper\_parameters 데이터프레임에 저장한다. 본 가이드북에서는 총 8개 하이퍼파라미터 조합을 실험하였다.



```
hyper_parameters = pd.DataFrame({'index': ['index_1', 'index_2', 'index_3', 'index_4', 'index_5', 'index_6', 'index_7', 'index_8'],
                                'n_iter': [200, 200, 200, 200, 200, 200, 200, 200],
                                'n_pop': [10, 20, 40, 20, 20, 20, 20, 20], 'r_cr
                                'oss': [0.4, 0.4, 0.4, 0.1, 0.2, 0.3, 0.4, 0.4],
                                'r_mut': [0.4, 0.4, 0.4, 0.4, 0.4, 0.4, 0.1, 0.6]
                                })
hyper_parameters['objective'] = np.nan
hyper_parameters['time'] = np.nan
hyper_parameters
```

	index	n_iter	n_pop	r_cross	r_mut	objective	time
	index_1	200	10	0.4	0.4	NaN	NaN
	index_2	200	20	0.4	0.4	NaN	NaN
	index_3	200	40	0.4	0.4	NaN	NaN
	index_4	200	20	0.1	0.4	NaN	NaN
	index_5	200	20	0.2	0.4	NaN	NaN
	index_6	200	20	0.3	0.4	NaN	NaN
	index_7	200	20	0.4	0.1	NaN	NaN
	index_8	200	20	0.4	0.6	NaN	NaN

[코드 31] 유전 알고리즘 하이퍼파라미터 실험을 위한 데이터프레임

- 모든 하이퍼파라미터 조합을 실험하여 가장 좋은 조합을 선택한다. 실험을 위한 코드는 다음과 같다. 'len(hyper\_parameters)'은 hyper\_parameters 데이터프레임의 row 수이며, 모든 경우에 대해 반복 실험한다.
- GA는 임의성이 바탕이 되므로 매번 결과는 달라질 수 있다.

```

log_list = []
for i in range(len(hyper_parameters)):
    parameter = hyper_parameters.iloc[i]
    index_nm = parameter['index']
    print(f'{index_nm}')
    start = datetime.datetime.now()
    xijt = generation_xijt()
    xijt_keys = list(xijt.keys())
    mijt_keys = list(mijt.keys())
    # number of generations
    n_iter = parameter['n_iter']
    # define the population size
    n_pop = parameter['n_pop']
    # crossover rate
    r_cross = parameter['r_cross']
    # mutation rate
    r_mut = parameter['r_mut']
    best, score, log, log_detail = genetic_algorithm(mijt, n_iter, n_pop, r_cross, r_mut)
    time = datetime.datetime.now() - start
    hyper_parameters.loc[i, 'time'] = time
    hyper_parameters.loc[i, 'objective'] = score
    log_list.append(log)
hyper_parameters

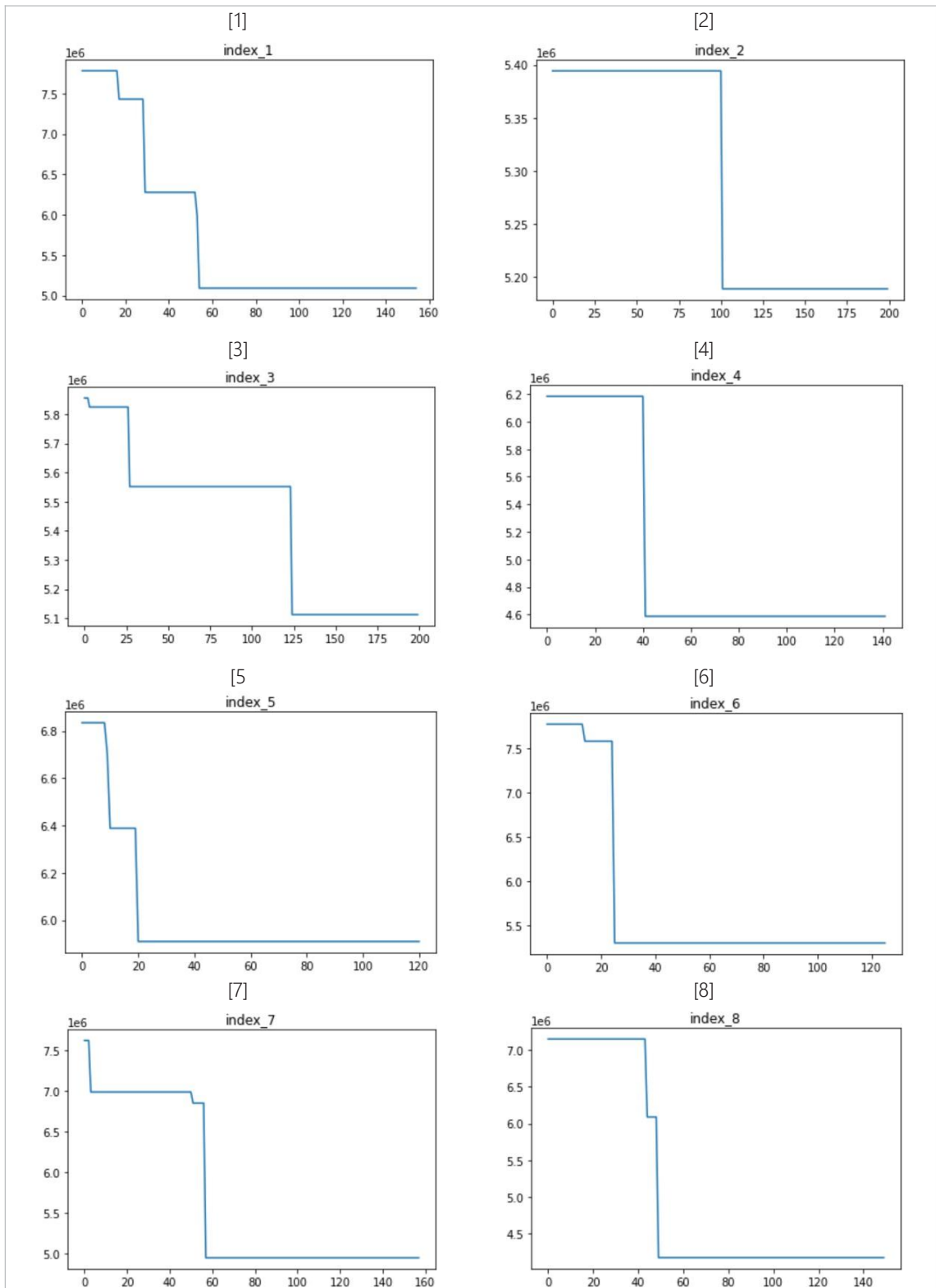
```

	index	n_iter	n_pop	r_cross	r_mut	objective	time (시간:분:초)
	index_1	200	10	0.4	0.4	5815951.0	0:02:32
	index_2	200	20	0.4	0.4	3938689.0	0:06:23
	index_3	200	40	0.4	0.4	3699132.0	0:09:52
	index_4	200	20	0.1	0.4	4061548.0	0:07:45
	index_5	200	20	0.2	0.4	4351699.0	0:09:50
	index_6	200	20	0.3	0.4	5621300.0	0:04:57
	index_7	200	20	0.4	0.1	4643242.0	0:01:19
	index_8	200	20	0.4	0.6	5117076.0	0:09:21

[코드 32] 유전 알고리즘 하이퍼파라미터 실험 코드

- 다음은 각 실험에 대한 시각화 코드와 결과이다.

```
for i in range(len(log_list)): plt.plot(list(pd.
    DataFrame(log_list[i])[1])) plt.title(f'index
    {i+1}')
plt.show()
```



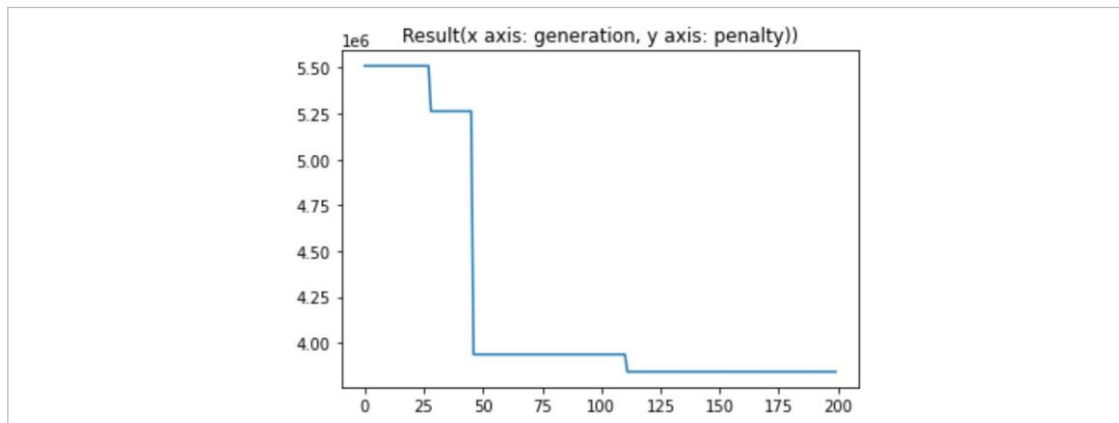
[코드 33] 유전 알고리즘 실험 시각화 코드

## [단계 7] 유전 알고리즘 실행

- 유전 알고리즘은 임의성이 바탕이며, 이로 인해 매번 결과는 달라진다. 따라서 [단계 6]에서 가장 적합한 하이퍼파라미터 조합은 달라질 수 있다. 'objective' 값이 가장 작은 조합을 선택한다.
- [단계 6] 실험을 통해 가장 적합한 하이퍼파라미터 조합은 index 3번이다. 따라서 최종 유전 알고리즘의 하이퍼파라미터는  $n\_iter = 200$ ,  $n\_pop = 40$ ,  $r\_cross = 0.4$ ,  $r\_mut = 0.4$  으로 할당한다.

```
xijt = generation_xijt() xijt_
keys = list(xijt.keys()) mijt_
keys = list(mijt.keys())
# 반복할 세대 수, 유전 알고리즘을 반복할 수
n_iter = 200
# 한 세대를 구성하는 염색체 수
n_pop = 40
# 교배율, 부모 염색체로부터 교배할 유전자 비율
r_cross = 0.4
# 변이율, 새로 생긴 염색체에서 임의로 변경할 유전자 비율
r_mut = 0.4
best, score, log, log_detail = genetic_algorithm(mijt, n_iter, n_pop, r_cross, r_mut)
print('Done!')

plt.plot(list(pd.DataFrame(log)[1]))
plt.title(f'Result(x axis: generation, y axis: penalty)')
```



[코드 34] 유전 알고리즘 실험 결과

## [단계 8] 결과 분석 및 해석

### ⑧-1. 분석결과 도출 가이드

- 유전 알고리즘으로 도출된 모든 해 확인은 다음과 같다.
- 'to\_csv()'함수는 데이터 프레임 자료형 데이터를 CSV로 저장한다.
- 유전 알고리즘의 최적해 결과는 매번 달라질 수 있다.

```

solution_ = dict()
xijt = decode(mijt, xijt)
for i in I:
    for j in J:
        for t in T:
            solution_[i, j, t] = round(best[i, j, t]*dit[i, t])
sol = pd.DataFrame.from_dict(solution_, orient='index').reset_index()
sol.columns = [('item, machine, time'), 'qty'] sol.to_csv('GA_solution.csv', index=False)
sol = pd.read_csv('GA_solution.csv')
sol[sol['qty']>0]

```

	(item, machine, time)	qty
46	('052996', 410.0, '2021-05-30')	1
51	('052996', 410.0, '2021-02-18')	1
73	('052996', 421.0, '2021-05-30')	1
78	('052996', 421.0, '2021-02-18')	2
208	('K04101', 408.0, '2021-05-30')	10
217	('K04101', 409.0, '2021-05-30')	13
244	('K04101', 416.0, '2021-05-30')	14
271	('K04101', 424.0, '2021-05-30')	10
280	('K04101', 425.0, '2021-05-30')	12
289	('K04101', 426.0, '2021-05-30')	7
445	('K04033', 422.0, '2021-05-13')	117
454	('K04033', 424.0, '2021-05-13')	30
490	('K04033', 434.0, '2021-05-13')	2
535	('K04033', 440.0, '2021-05-13')	62
568	('051718', 408.0, '2021-05-30')	6
577	('051718', 409.0, '2021-05-30')	4
586	('051718', 410.0, '2021-05-30')	7
595	('051718', 412.0, '2021-05-30')	5
604	('051718', 416.0, '2021-05-30')	4
631	('051718', 424.0, '2021-05-30')	4
640	('051718', 425.0, '2021-05-30')	4
649	('051718', 426.0, '2021-05-30')	15
658	('051718', 433.0, '2021-05-30')	6
667	('051718', 434.0, '2021-05-30')	3

[코드 35] 유전 알고리즘 최적해 예

## ⑧-2. 분석결과에 대한 해석 가이드

- 유전 알고리즘을 이용한 최적해는 정 해(exact solution)를 알 수 없으므로 정확성을 평가할 수 없다. 따라서 도출된 해를 전문가 지식을 바탕으로 평가해야 한다. 현재 모델의 최종 해에는 요구된 아이템 개수보다 많은 수를 생산하라는 결과를 도출

하기도 하는데, 이는 목적함수 값의 페널티(penalty)를 수정하여 개선할 수 있다. 개선 방향은 모델을 만드는 전문가가 페널티를 변경하며 경험적으로 알아내야 한다. 또한, 더 좋은 최적 해를 찾기 위해서는 한 세대를 구성하는 구성이 많을수록, 거듭 되는 세대가 많을수록 좋다. 하지만 최적 해와 계산 비용은 서로 절충(trade-off) 관계를 맺고 있으므로 전문가는 개발환경을 고려해 하이퍼파라미터를 결정하는 것도 중요하다.

### 3. 유사 타현장의 「생산계획 최적화 AI 데이터셋」 분석 적용

#### 1. 본 분석이 적용 가능한 제조현장 소개

- NC기계를 사용하여 생산품을 제조하는 대부분의 중소기업 현장에서는 주문받은 제품을 생산하기 위하여 생산 일정계획 및 납기 산정을 중요한 이슈로 다루고 있다. 더욱더 효율적으로 생산 일정계획 및 납부기한을 맞추어야 하지만 여전히 현장 숙련자의 경험과 노하우에 의존하여 상황에 맞게 설비운동을 변경할 수밖에 없는 상황이다. 따라서 추후 발생 가능한 긴급 주문건 등을 고려하여 문제의 복잡도가 올라갔을 때의 동적인 주문에 대응 가능한 연구가 필요할 것이다. 특히 여러 기계가 동시에 작업이 가능한 공정에서 각 기계가 가지는 사이클 타임을 고려하여 주문의 투입 시점과 제조 납기를 결정하고자 한다.

#### 2. 본 「생산계획 최적화 AI 데이터셋」 분석을 원용하여 타 제조현장 적용 시, 주요고려사항

- 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.
- 공정 과정을 수행하는 NC 기계는 작업 할당 시 먼저 실시되는 작업이 종료된 이후 다음 작업을 시행하며 동시에 다양한 작업을 하는 것이 불가능하다는 제약내용이 반영되어야 한다

### 3 부록 (분석환경 구축을 위한 설치 가이드)

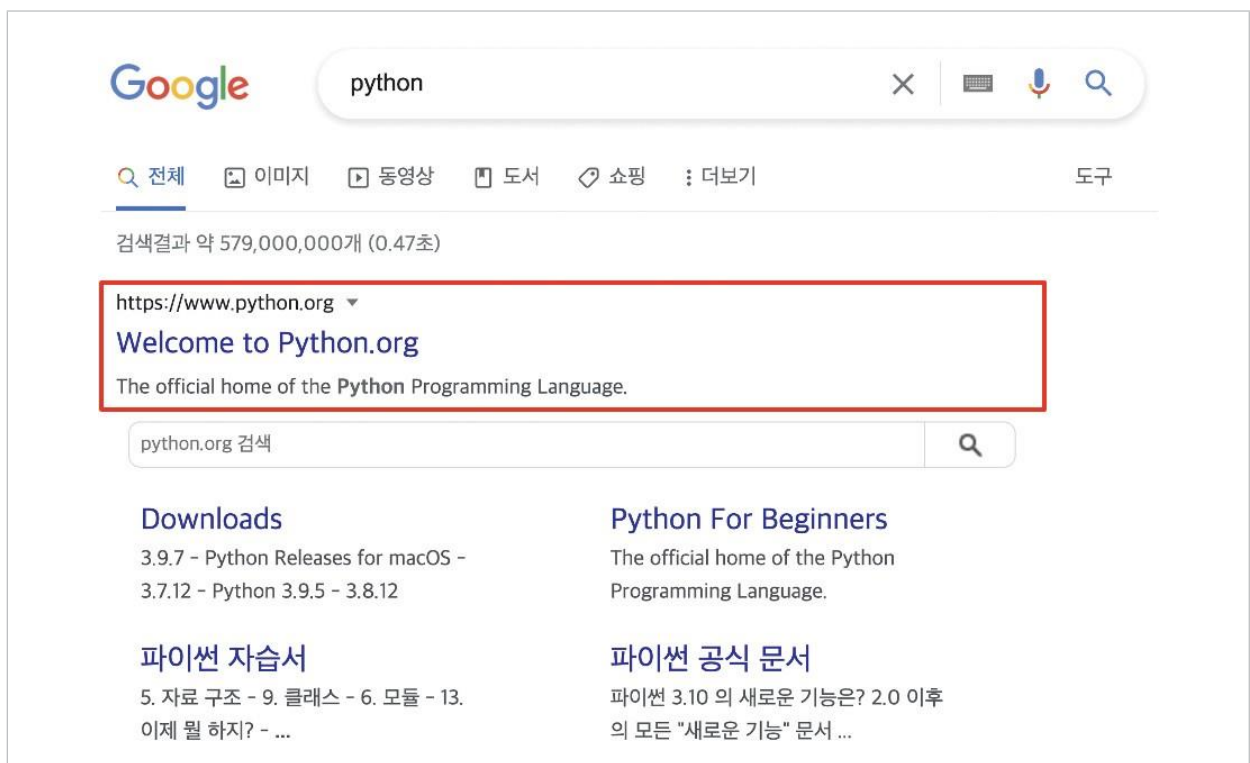
#### 1. 파이썬(Python) 설치

파이썬은 컴퓨터 언어로서 최근 데이터 분석 및 AI 분석모델 개발 등에 널리 사용되는 도구이다. 다운로드 및 설치가 간편하고 활용도가 높은 파이썬을 로컬 컴퓨터에 설치하고 적용하는 방법을 안내한다.

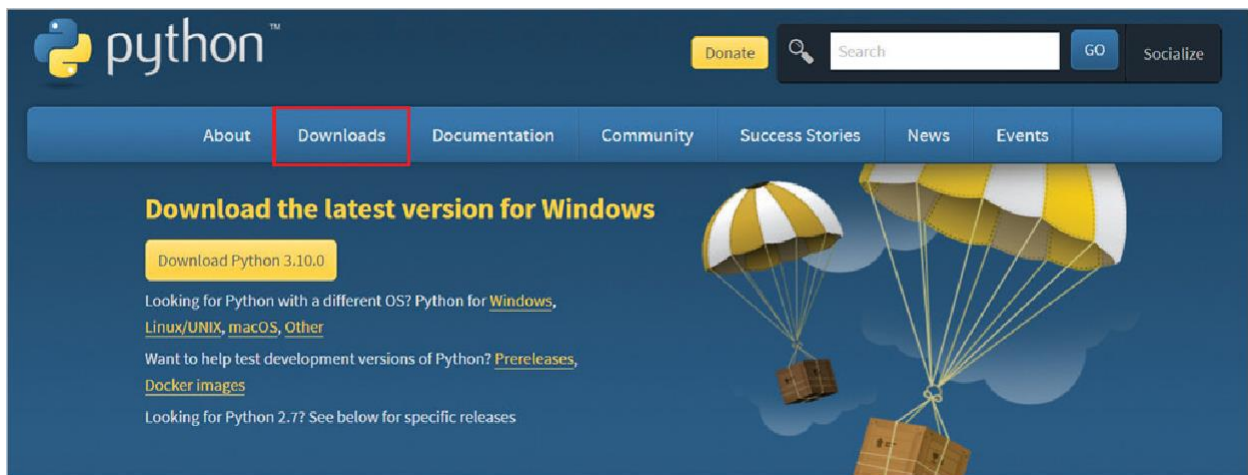
① google.com 등의 검색 엔진에 'python'을 검색한다.



② 제일 처음에 보이는 'Welcome to Python.org'를 클릭한다.



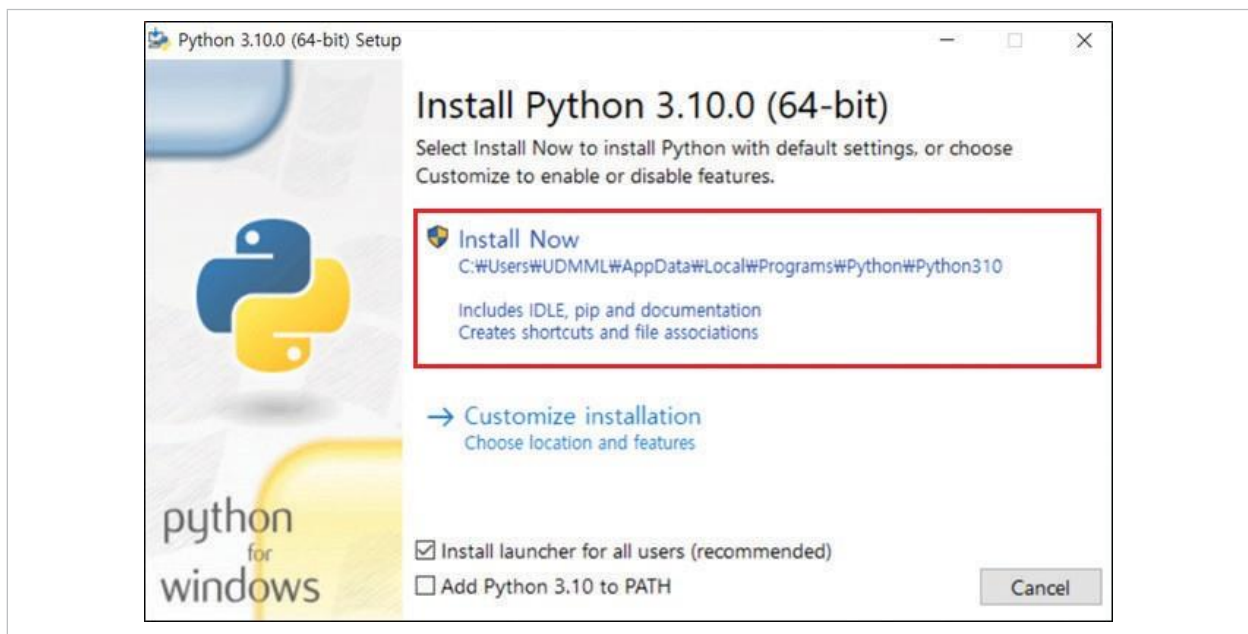
③ 클릭후 보이는 페이지의 정면에서 상단 좌측 2번째 'Downloads'를 클릭한다.



④ 컴퓨터의 운영체제에 따라 상단에서 세번째(macOS의 경우 네번째) 탭을 선택한 후, Python 3.10.0을 다운로드한다. (Python 3.10.0 버전은 업데이트 될 수 있다)

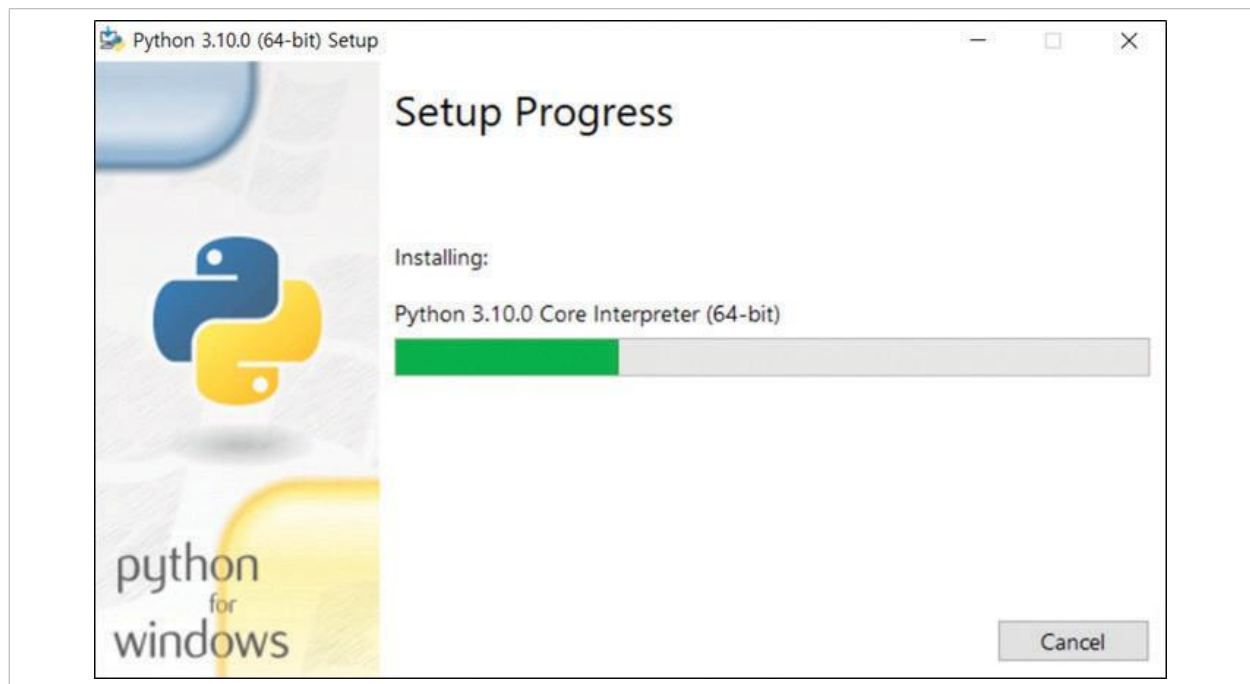


⑤ 아래와 같은 설치창이 뜨면, 'Install Now'를 클릭한다.

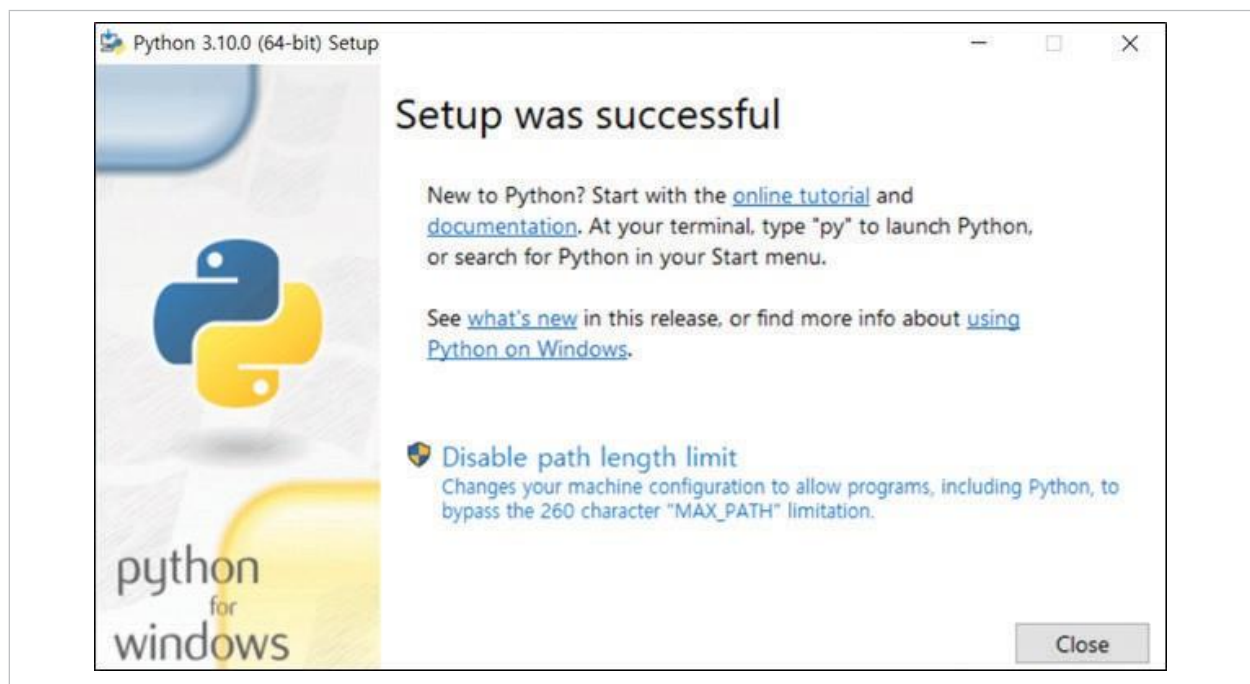




⑥ 아래와 같은 설치 진행창이 완료가 될 때까지 유지한다.



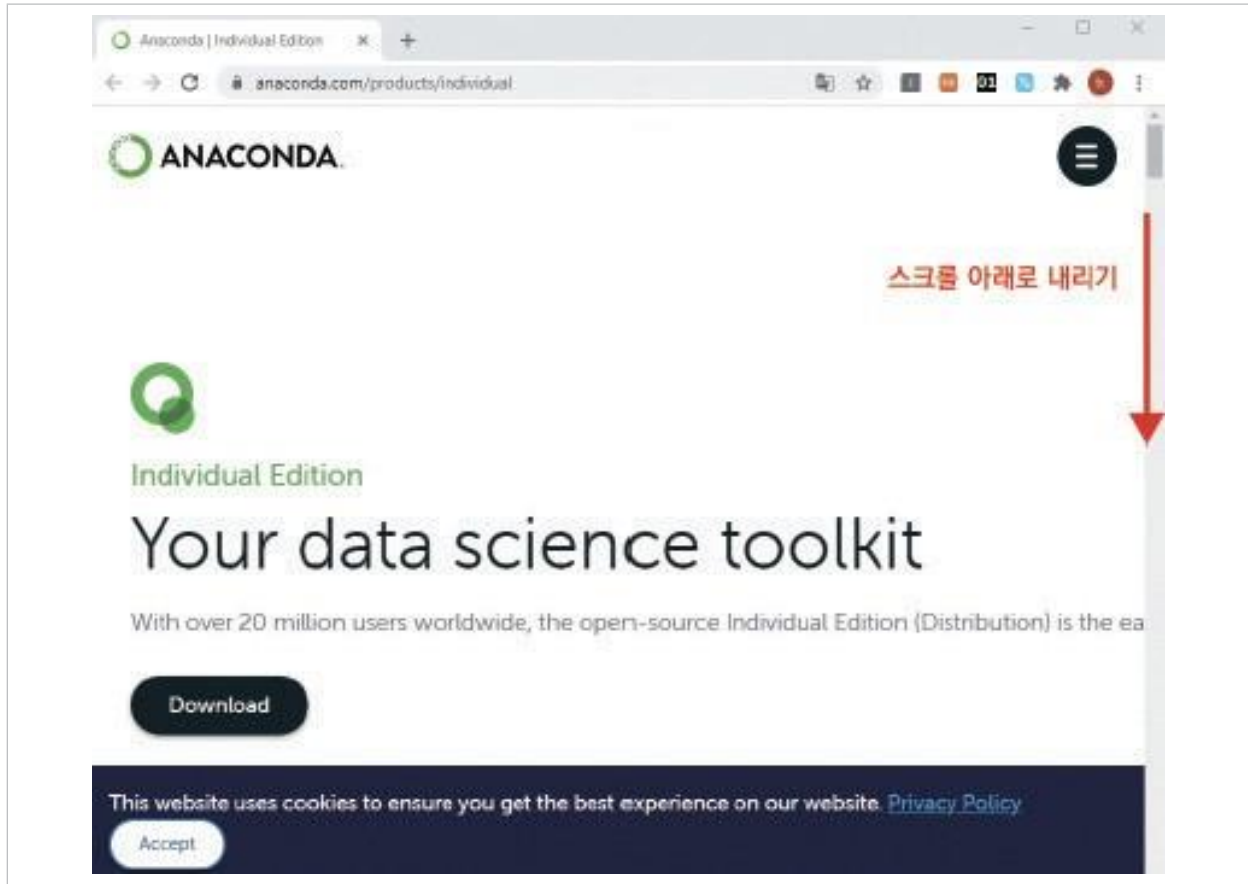
⑦ 완료가 되면 아래와 같은 창이 뜨는 것을 확인 후 종료한다. [설치완료]



## 2. 아나콘다(Anaconda) 설치

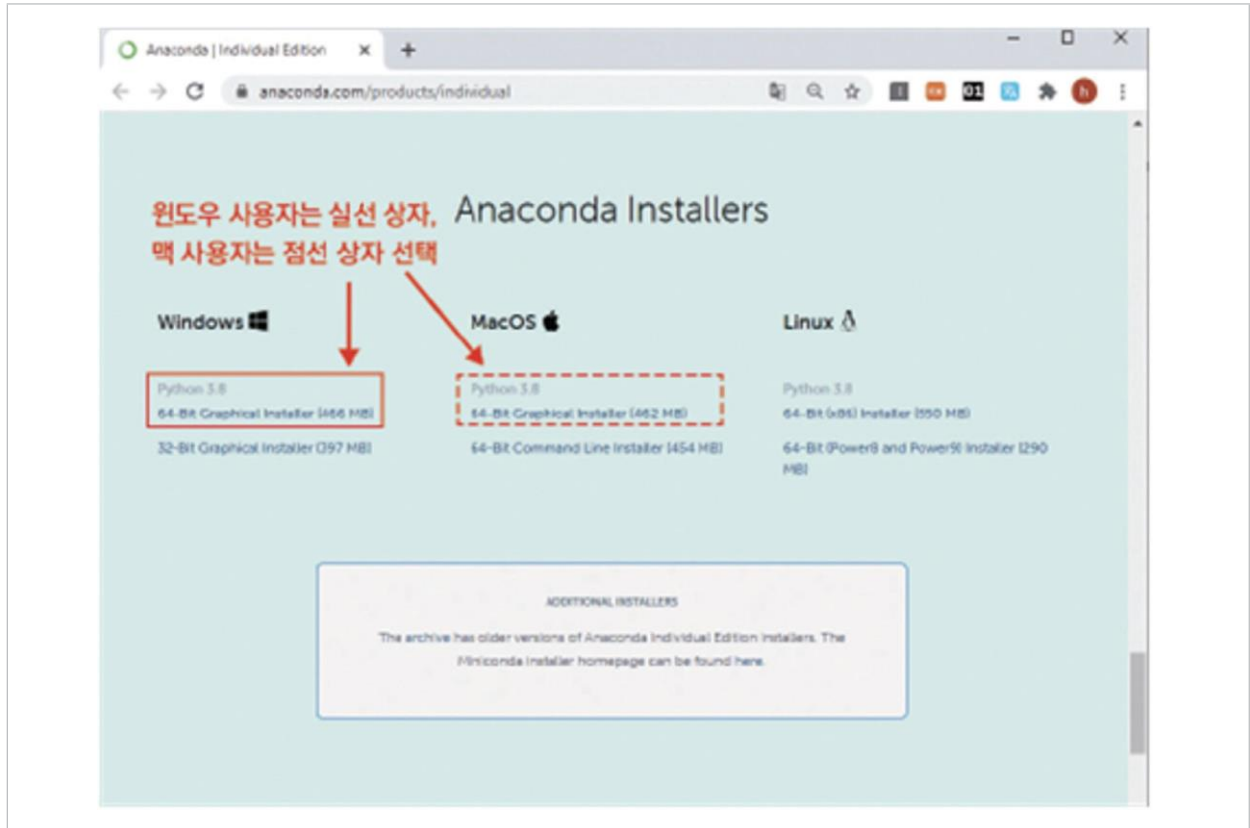
아나콘다란 파이썬과 같은 분석 도구를 사용할 때 필요한 고급 기능 및 분석을 보조하는 도구이다. 아나콘다를 설치함으로써 다양한 기능들을 바로 사용할 수 있고, 결과물을 쉽게 확인할 수 있다. 아나콘다를 설치하고 분석을 실시할 수 있는 환경을 구축하는 방법에 대해 안내한다.

① <https://www.anaconda.com/distribution/> 로 접속한다.



② 다음과 같은 화면이 나올 때 까지 스크롤하여 아래로 내린 후, 로컬 컴퓨터 사용 환경에 맞는 파일을 다운받는다.

- ▶ Windows : 64-Bit Graphical Installer
- ▶ MacOS : 64-Bit Graphical Installer

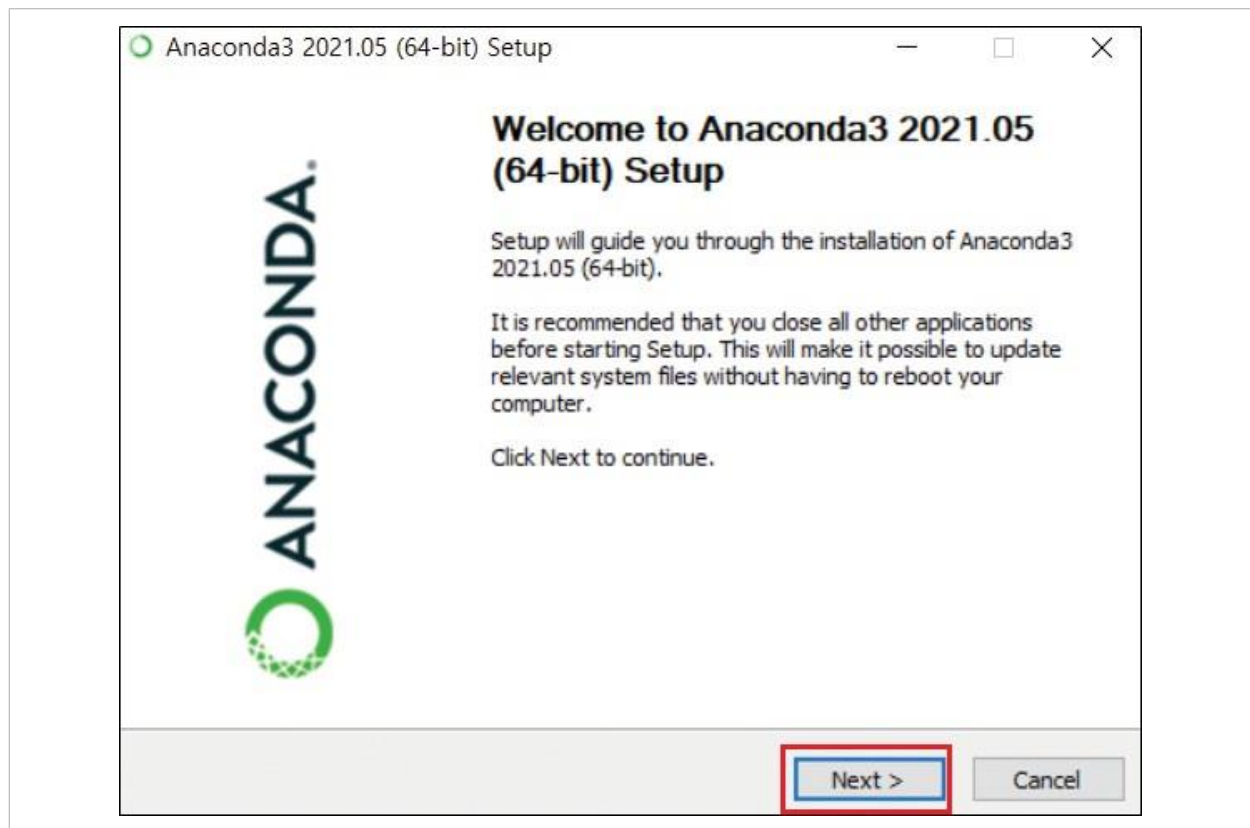


③ 다운로드 받은 파일로 이동하여, 설치된 아나콘다 파일 위에서 마우스 오른쪽을 클릭한 후, 방패모양의 '관리자 권한으로 실행'을 클릭한다.

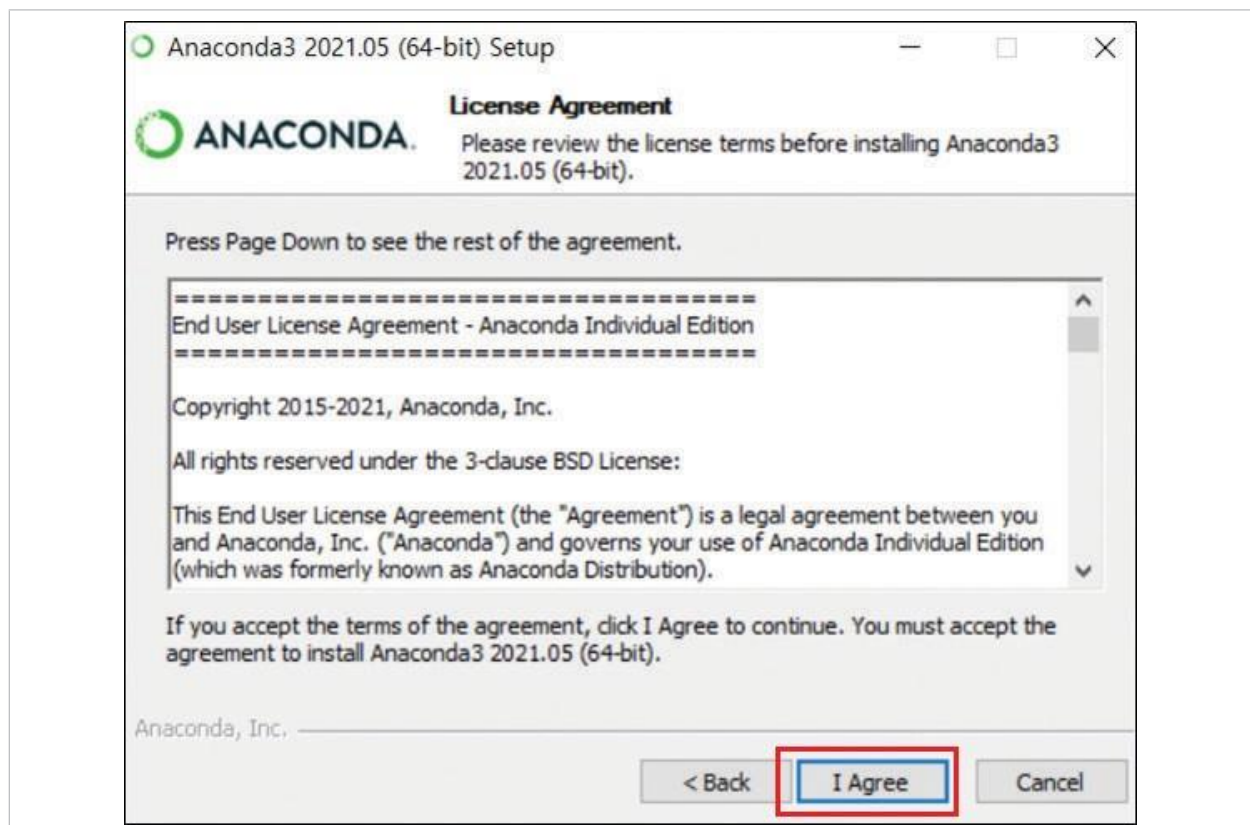
- ▶ (예) '다운로드' 폴더로 아나콘다를 다운 받은 경우



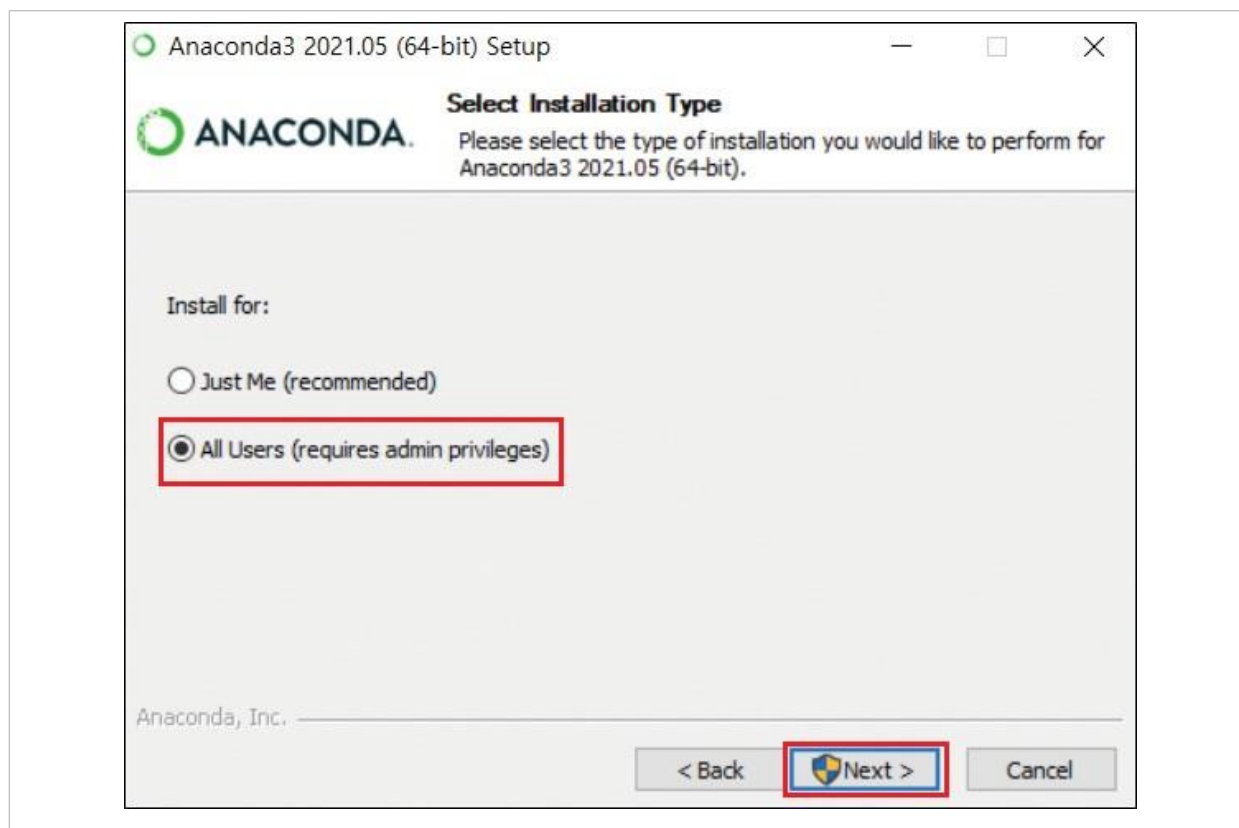
④ 파일을 실행 한 후, 'Next' 버튼을 클릭한다.



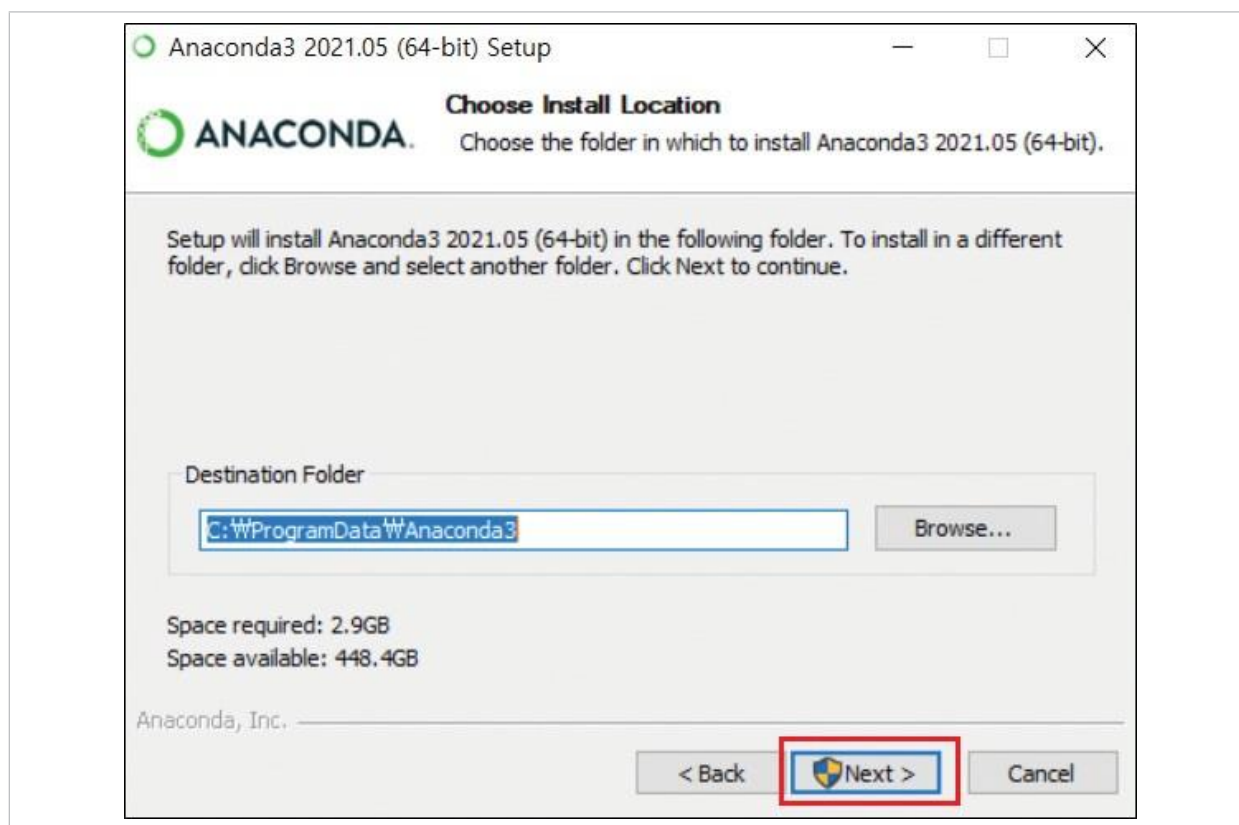
⑤ 다음 창이 나타나면 'I Agree'를 선택한다.



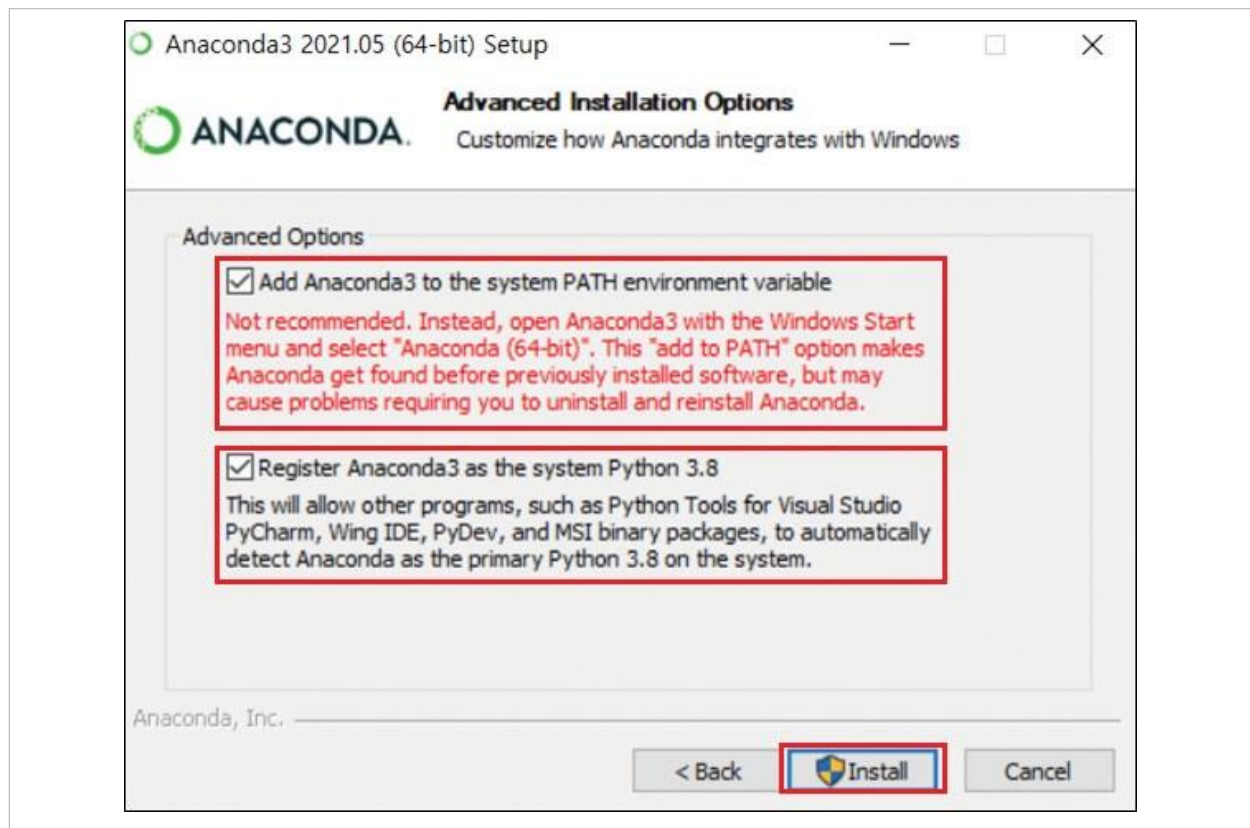
⑥ 셋팅 창이 뜨면 화면의 'All Users'를 선택 후 아래의 'Next'를 클릭한다.



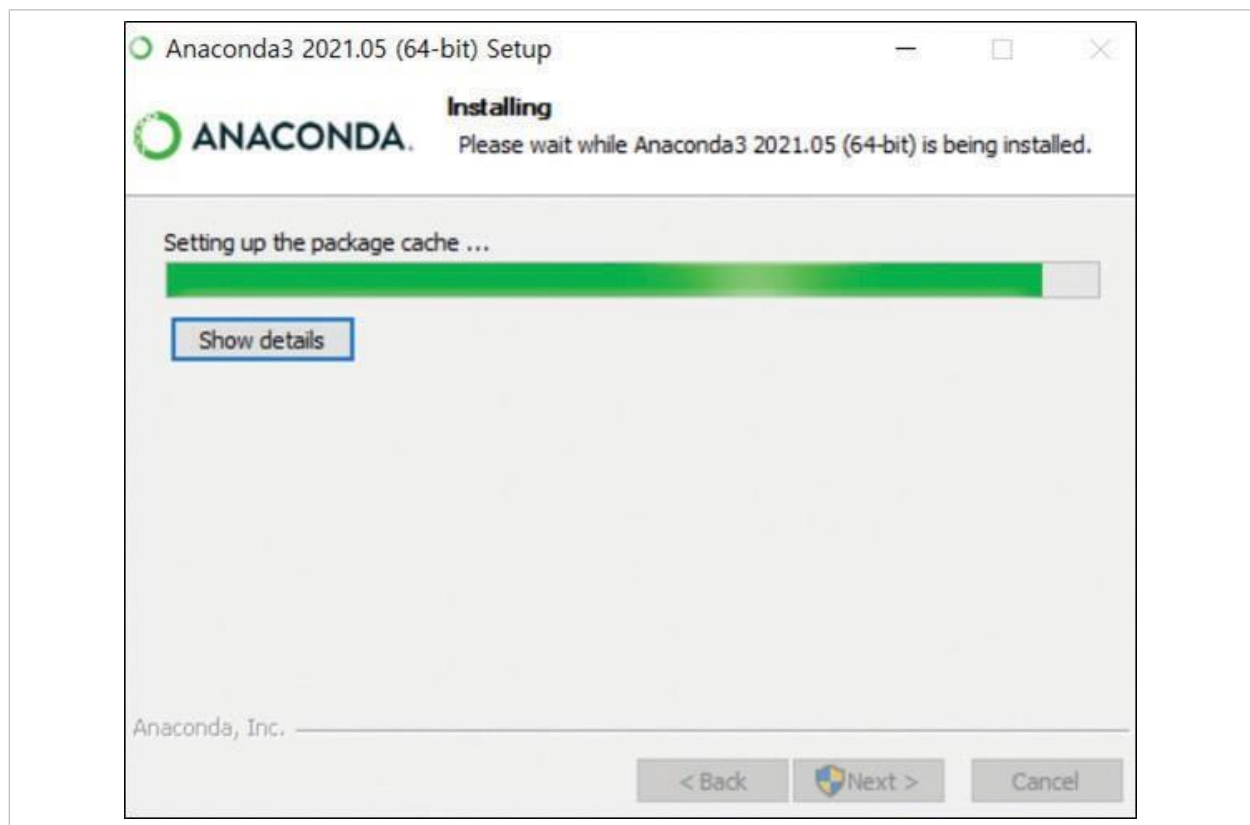
⑦ 다운로드 받을 경로를 물어보는 창이 뜨면, 아래의 'Next'를 클릭한다.



⑧ 고급 옵션 선택창이 뜨면, 아래와 같이 모두 선택 후, 'Install'을 클릭한다.

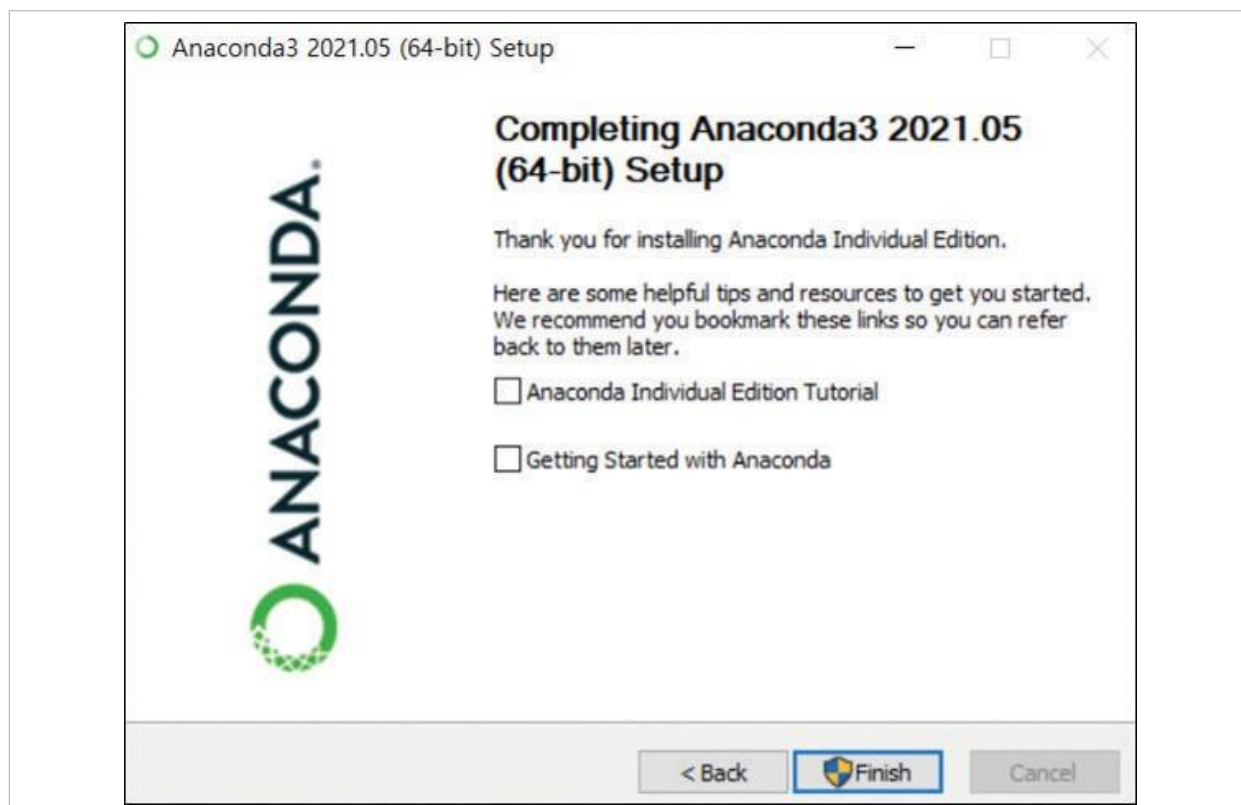


⑨ 다음과 같은 설치창이 뜨면 완료가 될 때까지 대기 (5분이상 소요)

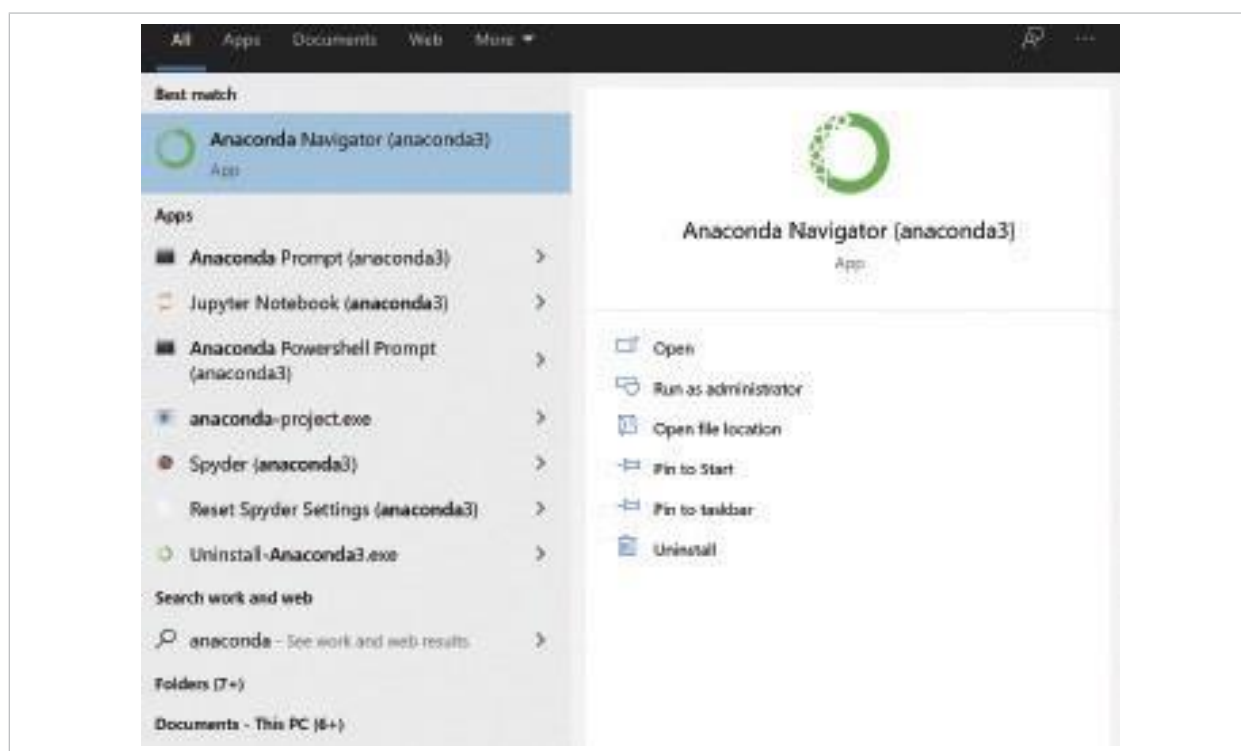




⑩ 마지막 화면에서, 모두 체크 해제한 후, 'Finish'를 눌러 설치를 완료한다.




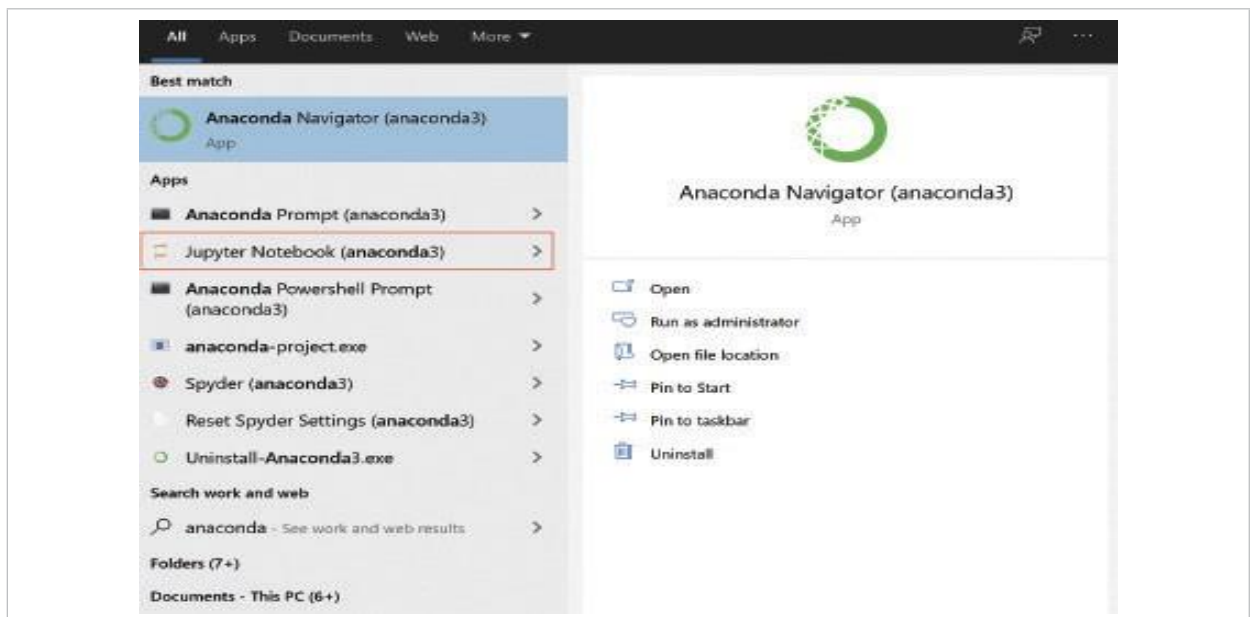
⑪ 화면상의 '홈' 버튼을 눌러서 화면과 같이 anaconda prompt가 잘 설치 되었는지를 확인 한다. Anaconda Navigator, Anaconda Prompt, Jupyter Notebook 등의 다른 응용 프로그램들도 함께 확인이 된다면 설치가 완료된 것이다.



### 3. 주피터 노트북 (Jupyter Notebook) 실행

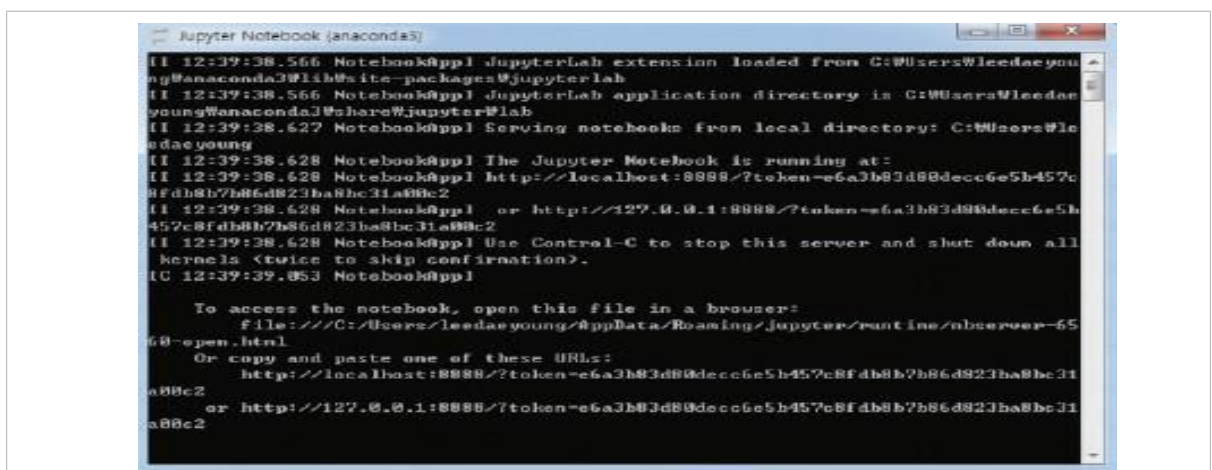
주피터 노트북은 실제로 사용자가 코딩(분석 문장 작성)을 할 수 있는 도구이다. 쉽게 비교하자면, 문서 도구로 마이크로소프트 사의 'Word' 프로그램이나, 한컴소프트 사의 '한글' 프로그램 등과 같은 도구라고 생각할 수 있다. 데이터 분석에 다양한 입력, 실행 도구가 있지만, 본 가이드 북에서는 주피터 노트북을 활용하는 방법을 안내하기로 한다. Anaconda를 설치한 이후 주피터 노트북(Jupyter Notebook) 설치 방법을 확인하면 된다.

- ① 화면상의 '홈(  )'키를 눌러서 화면과 같이 'anaconda' 검색 및 폴더 리스트 중 'Jupyter Notebook (anaconda3)'을 실행한다.



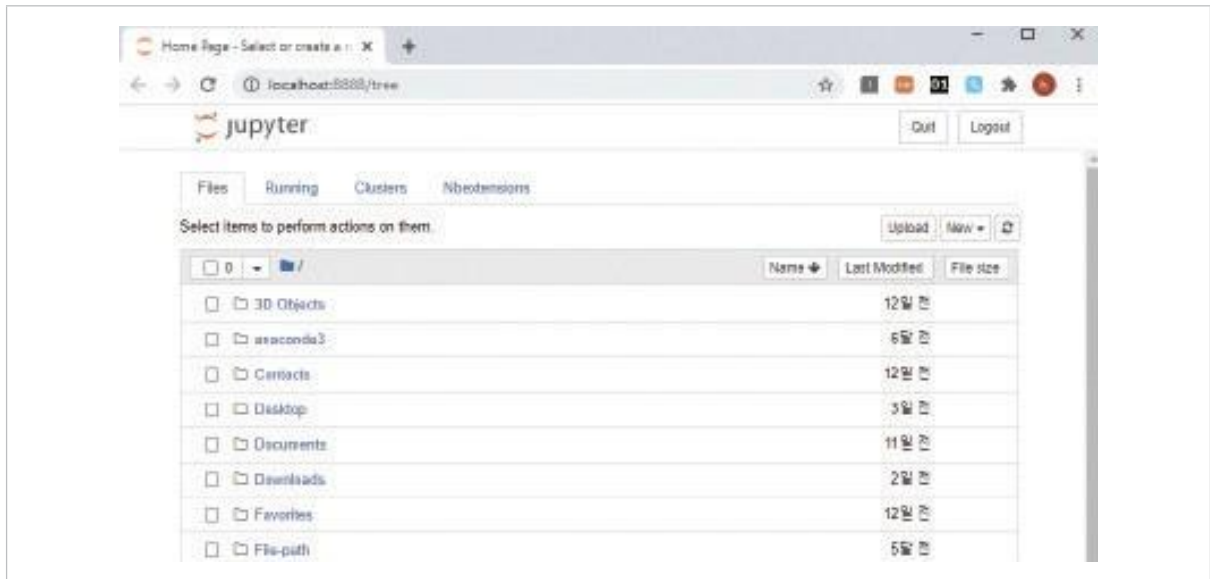
- ② Jupyter Notebook을 클릭시 아래처럼 2개의 윈도우가 실행된다.

- (1) 검은색 배경의 화면은 주피터 노트북이 실행되는 환경에 대한 상태를 나타내주는 상태 표시 창이다. 주피터 노트북을 사용하는 동안 종료하면 안된다.





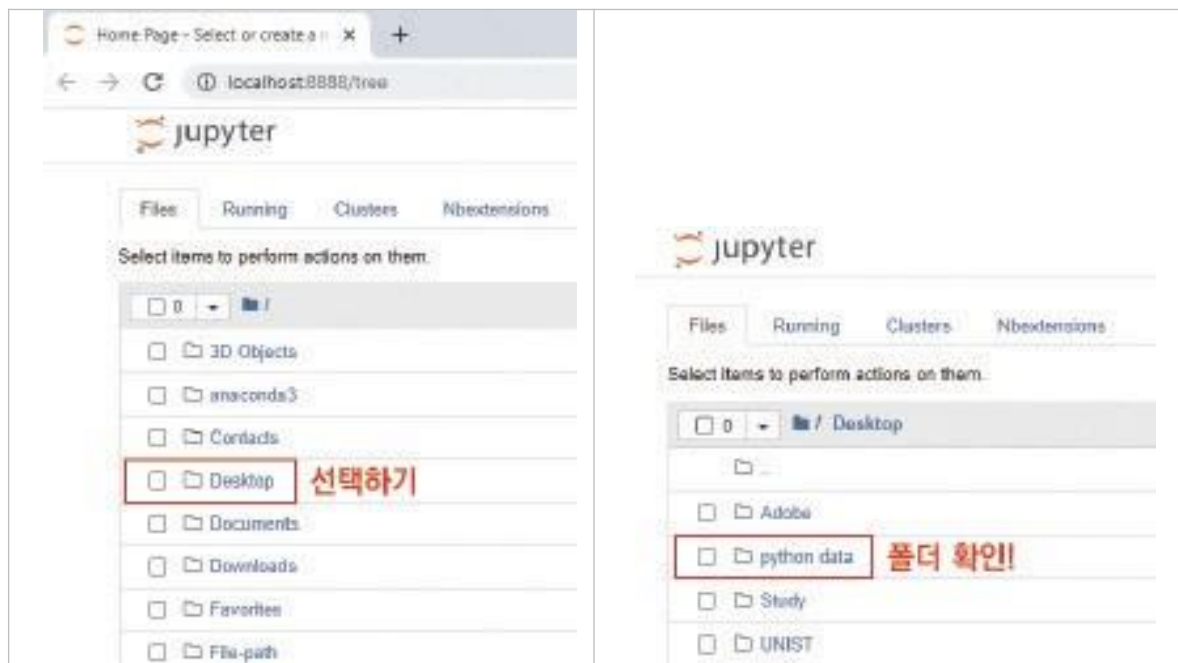
(2) 사용하는 인터넷 프로그램(크롬, 인터넷 익스플로러)에 주피터 노트북이 열린다. (다른 확장 프로그램 사용하고 싶다면 기본 브라우저를 변경해주어야한다)



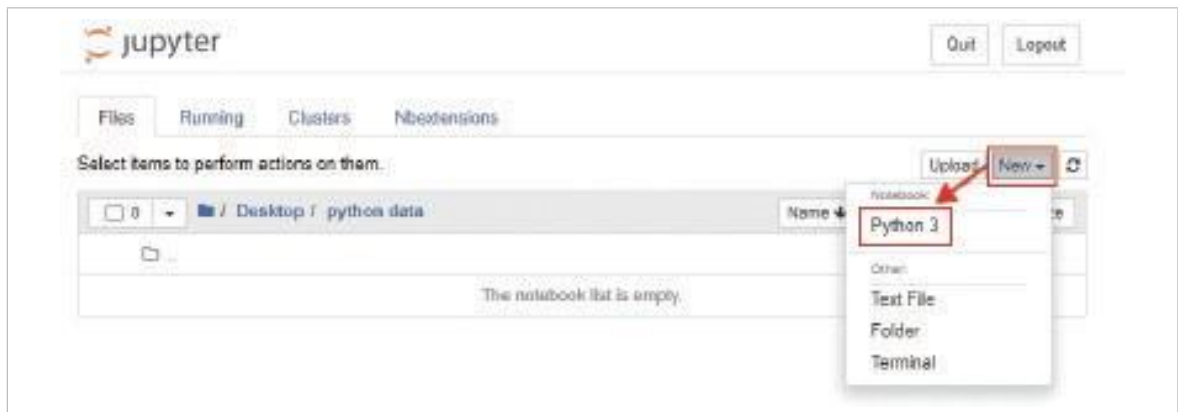
③ 데이터를 저장하고, 불러오고, 분석할 경로의 폴더를 하나 생성한다.


▶ (예) '바탕화면'에 'python data' 폴더를 생성 후 (미리 생성), 파이썬 코드 실행하고 저장한다.

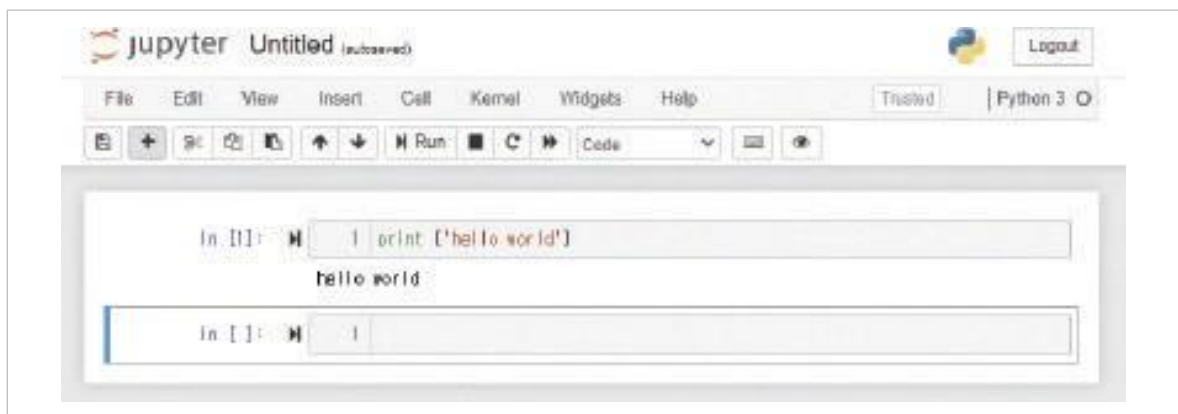
(1) 주피터 노트북에서 'python data' 폴더를 확인한다.



(2) python data에서 파이썬 파일을 생성한다. 오른쪽 상단의 'New'를 누른 후, 'Python 3'을 선택한다.



(3) 'hello world' 출력을 확인해본다. 보이는 In [ ] 우측 회색 창에 print('hello world') 입력 후, 상단의  또는 shift + Enter 키를 눌러서 실행한다. 코드 파일의 확장자는 '.ipynb'로 저장된다.



## 4 부록 [데이터 품질 전처리 [실습 코드]]

앞선 '2.분석 실습'에서 데이터 품질 전처리에 필요한 6가지 특성에 대하여 논의하였다. (완전성, 유일성, 유효성, 일관성, 정확성, 무결성)

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

- 다루는 Numerical 데이터는 order\_info와 machine\_info, 두 가지이다.

```
perc = 30
df_set = {'order_info':order_info,'machine_info':machine_info}
for df_name in list(df_set):
    print(f'DataFrame name: {df_name}')
    df = df_set[df_name]
    print('[step 1-1]')
    print(round(df.isnull().sum()/len(df)*100,2))
    print('[step 1-2]')
    print(df.isnull().sum()/len(df)*100>perc)
    print('[step 2-1]')
    print(df.isnull().head())
    print('[step 2-2]')
    print(df.isnull().sum())
    print('[step 2-3]')
    cmpt_len = df.isnull().sum().sum()
    print(cmpt_len)

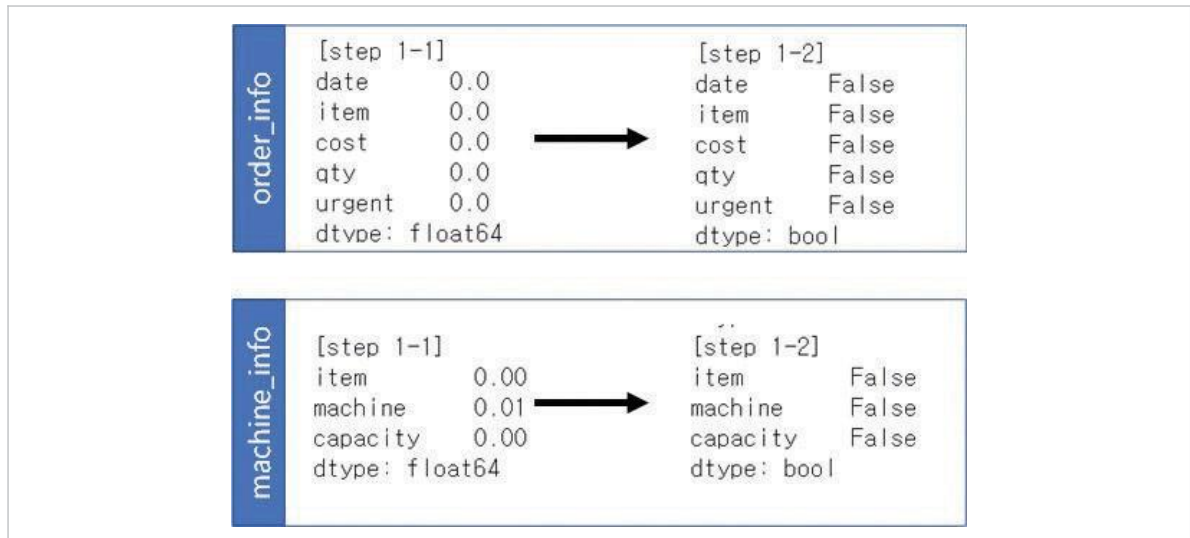
print("결측치 = %d개 %n완전성 지수 : %.2f%%"%(cmpt_len,(1-cmpt_len/len(df))*100))
```

[코드 36] 완전성 품질지수 전체 코드

① Null 값이 30% 이상인 데이터들은 데이터의 완전성이 떨어지기 때문에 열별 Null 값의 비율을 확인하여 삭제한다. 이 데이터는 30%를 넘는 열이 존재하지 않으므로 열을 제거하지 않는다.

```
print('[step 1-1]')
print(round(df.isnull().sum()/len(df)*100,2))
```

[코드 37] 완전성 품질지수 코드 (1)

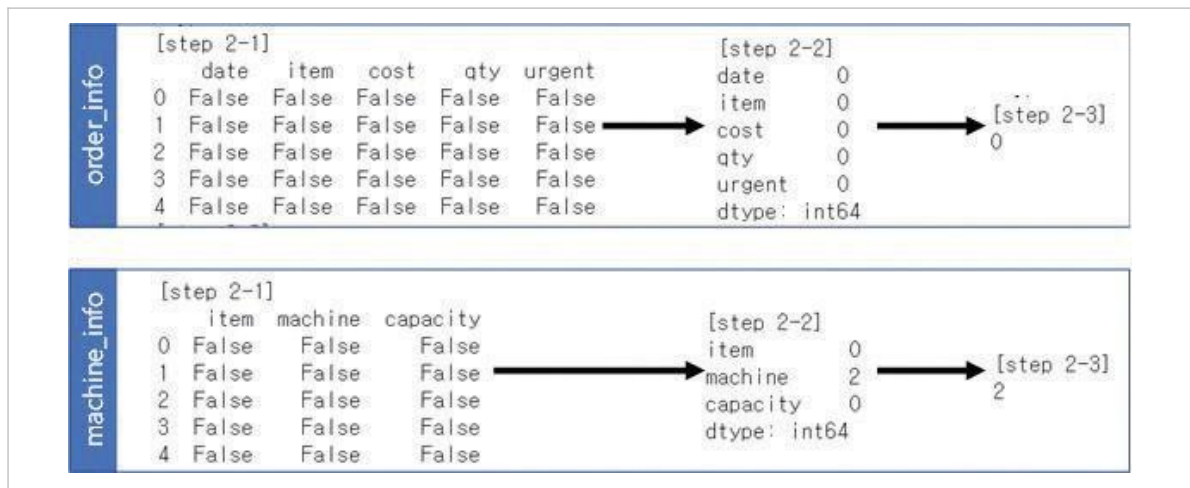


[그림 16] 분석 도구를 활용한 완전성 품질 지수 확인 (1)

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

```
print('step 2-1')
print(sum(df.isnull()))
```

[코드 38] 완전성 품질지수 코드 (2)



[그림 17] 분석 도구를 활용한 완전성 품질지수 확인 (2)

③ 구한 결측치의 개수를 이용하여 완전성 품질 지수를 구한다.

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

결측치 = 2개  
완전성 지수 : 99.99%

[코드 39] 분석 도구를 활용한 완전성 품질 지수 구하기

## ▶ 유일성 품질 지수 = ((유일한 데이터 수)/전체 데이터 수) \* 100

- order\_info 데이터의 경우 [date, item, cost, urgent] 열이 유일한 값을 가지는 기본 key이며, machine\_info 데이터는 [item, machine] 열이 기본 key이다.

### # 2. 유일성 (Uniqueness)

```
check_unique = order_info[
    ['time', 'item', 'urgent']
].value_counts().reset_index()
perc_check_unique_item_urgent_info = round(
    (len(check_unique) - len(check_unique[check_unique[0] > 1]))
    / len(check_unique) * 100, 2)

check_unique = machine_info[
    ['item', 'machine']
].value_counts().reset_index()
perc_check_unique_machine_info = round(
    (len(check_unique) - len(check_unique[check_unique[0] > 1]))
    / len(check_unique) * 100, 2)

print(f'The percentage of uniqueness for time<->item<->cost<->urgent : {perc_check_unique_item_urgent_info}')
print(f'The percentage of uniqueness for item<->machine : {perc_check_unique_machine_info}')
print(f'유일성 지수 : %.2f%%' % (
    (perc_check_unique_machine_info + perc_check_unique_item_urgent_info) / 2))
```

The percentage of uniqueness for time<->item<->cost<->urgent : 98.33

The percentage of uniqueness for item<->machine : 100.0

유일성 지수 : 99.16%

[코드 40] 유일성 품질 지수 코드

- ① 'value\_counts()' 함수를 이용하여 key 값들 개수를 확인한다. 개수가 큰 순서로 기본 정렬된다. 'reset\_index()' 함수는 데이터 프레임 인덱스를 재부여한다.

<pre>check_unique = order_info[     ['date', 'item', 'urgent'] ].value_counts().reset_index() check_unique</pre>					<pre>check_unique = machine_info[     ['item', 'machine'] ].value_counts().reset_index() check_unique</pre>			
	date	item	urgent	0		item	machine	0
0	2021-07-09	057386	1	2	0	050060	0433	1
1	2021-07-09	057387	1	2	1	K00347	0404	1
2	2021-06-15	K03115	0	2	2	K00337	0438	1
3	2021-02-10	S00341	0	1	3	K00337	0437	1
4	2021-07-07	K03894	1	1	4	K00337	0410	1
...	...	...	...	...	...	...	...	...
119	2021-06-17	060711	0	1	21144	058398	0408	1
120	2021-06-17	053695	0	1	21145	058398	0404	1
121	2021-06-15	Z00807	0	1	21146	058396	0438	1
122	2021-06-15	063414	0	1	21147	058396	0434	1
123	2021-09-15	Z00807	0	1	21148	Z01415	0409	1
124 rows × 4 columns					21149 rows × 3 columns			

[코드 41] 분석 도구를 활용한 유일성 품질 지수 값 확인하기

② key 값들 개수로 정렬했을 때, 1 이상인 행 개수 비율의 평균이 유일성 품질 지수다.

```
perc_check_unique_item_urgent_info = round(
    (len(check_unique)-len(check_unique[check_unique[0]>1]))
    /len(check_unique)*100,2)
print(f'The percentage of uniqueness for time<->item<->cost<->urgent: {perc_check_unique_item_urgent_info}')
print("유일성 지수: %.2f%%"%(
    (perc_check_unique_machine_info+perc_check_unique_item_urgent_info)/2))
```

The percentage of uniqueness for time<->item<->cost<->urgent: 98.33  
유일성 지수 : 99.16%

[코드 42] 분석 도구를 활용한 유일성 품질 지수 구하기

## ▶ 유효성 품질 지수 = (유효성 만족 데이터 수/전체 데이터 수)\*100

- 유효성 지수를 만족하는 데이터만 vald\_df에 저장하고 최종적으로 완성된 데이터를 이용하여 유효성 품질 지수를 구한다.

```
# 3. 유효성 (Validity)

print('order_info case')
df = order_info
c_lb = df['cost'] >= 0
c_ub = df['cost'] <= 1000000
q_lb = df['qty'] >= 0
q_ub = df['qty'] <= 1000000
u_lb = df['urgent'] >= 0
u_ub = df['urgent'] <= 1
b = df['urgent'] <= 1
vald_df = df[c_lb & c_ub & q_lb & q_ub & u_lb & u_ub]
print(f'[Step 1] 데이터 범위를 벗어난 데이터 수: {len(df)-len(vald_df)}')

d0 = pd.Timestamp(datetime.date(2021,1,31))
d1 = pd.Timestamp(datetime.date(2021,10,31))
con1 = vald_df['time'] >= d0
con2 = vald_df['time'] <= d1
ld_df = vald_df[con1 & con2]
print(f'[Step 2] 수집된 날짜를 벗어나는 데이터 수: {len(df)-len(vald_df)}')

vald_df['time'] = vald_df['time'].apply(lambda x: isinstance(x, datetime.datetime))
vald_df['item'] = vald_df['item'].apply(lambda x: isinstance(x, str))
vald_df['cost'] = vald_df['cost'].apply(lambda x: isinstance(x, int))
vald_df['qty'] = vald_df['qty'].apply(lambda x: isinstance(x, int))
vald_df['urgent'] = vald_df['urgent'].apply(lambda x: isinstance(x, int))
print(f'[Step 3] 데이터 형식을 벗어나는 데이터 수: {len(df)-len(vald_df)}')

vald_df[
    (vald_df['time'] == True) & (
        vald_df['item'] == True) & (
            vald_df['cost'] == True) & (
                vald_df['qty'] == True) & (
                    vald_df['urgent'] == True)
]
vald_len = len(vald_df)
item_vald = vald_len / len(df) * 100
print("order_info 유효성 지수: %.2f%%"%(item_vald))
```

```

print(machine_info case)
df = machine_info
cap_lb = df['capacity'] >= 0
cap_ub = df['capacity'] <= 1000000
vald_df = df[cap_ub & cap_lb]
print(f'[Step 1] 데이터 범위를 벗어난 데이터 수: {len(df)-len(vald_df)}')

vald_df['item'] = vald_df['item'].apply(lambda x: isinstance(x, str))
vald_df['machine'] = vald_df['machine'].apply(lambda x: isinstance(x, str))
vald_df['capacity'] = vald_df['capacity'].apply(lambda x: float(x) if isinstance(x, int) == True else x)
vald_df['capacity'] = vald_df['capacity'].apply(lambda x: isinstance(x, float))
print(f'[Step 3] 데이터 형식을 벗어나는 데이터 수: {len(df)-len(vald_df)}')

vald_df[
    (vald_df['item'] == True)
    & (vald_df['machine'] == True)
    & (vald_df['capacity'] == True)
]
vald_len = len(vald_df)
machine_vald = vald_len / len(df) * 100
print("machine_info 유효성 지수 : %.2f%%" % (machine_vald))

print("유효성 지수 : %.2f%%" % ((item_vald + machine_vald) / 2))

order_info case
[Step 1] 데이터 범위를 벗어난 데이터 수: 0
[Step 2] 수집된 날짜를 벗어나는 데이터 수: 0
[Step 3] 데이터 형식을 벗어나는 데이터 수: 0
order_info 유효성 지수 : 100.00%

machine_info case
[Step 1] 데이터 범위를 벗어난 데이터 수: 0
[Step 3] 데이터 형식을 벗어나는 데이터 수: 0
machine_info 유효성 지수 : 100.00%
유효성 지수 : 100.00%

```

[코드 43] 유효성 품질지수 전체 코드

### ① 데이터가 유효범위 내에 들어가 있는가?

- 데이터셋에 정의된 수집 범위를 이용하여 조건(유효범위)을 모두 충족하는 데이터들을 vald\_df에 저장한다. 다음은 order\_info에 대한 예시이다.

```

df = order_info
c_lb = df['cost'] >= 0
c_ub = df['cost'] <= 1000000
q_lb = df['qty'] >= 0
q_ub = df['qty'] <= 1000000
u_lb = df['urgent'] >= 0
u_ub = df['urgent'] <= 1
vald_df = df[c_lb & c_ub & q_lb & q_ub & u_lb & u_ub]
print(f'[Step 1] 데이터 범위를 벗어난 데이터 수: {len(df)-len(vald_df)}')
[Step 1] 데이터 범위를 벗어난 데이터 수: 0

```

[코드 44] 유효성 품질지수 확인 코드 (1)

## ② 수집된 날짜 안에 들어가 있는가?

- 데이터셋 중 날짜 데이터를 포함하는 데이터셋 만 확인한다.
- 2021.02.01 ~ 2021.10.31에 수집된 데이터이기 때문에, 데이터의 날짜가 이 기간을 벗어나지 않았는지 확인한다.

```
d0 = pd.Timestamp(datetime.date(2021,1,31))
d1 = pd.Timestamp(datetime.date(2021,10,31))
con1 = vald_df['time']>=d0
con2 = vald_df['time']<=d1
vald_df = vald_df[con1&con2]
print(f'[Step 2] 수집된 날짜를 벗어나는 데이터 수: {len(df)-len(vald_df)}')
```

[Step 2] 수집된 날짜를 벗어나는 데이터 수: 0

[코드 45] 유효성 품질 지수 확인 코드 (2)

## ③ 데이터가 형식에 맞는가?

- 데이터 열은 다음 형식이어야 한다.

order_info		machine_info	
item	str	item	str
time	datetime	machine	str
cost	int	capacity	int or float
qty	int		
urgent	int		

- 'isinstance()' 함수를 사용하여 데이터 형식을 만족하는지 확인한다. 다음은 order\_info 에 대한 예시이다.

```
vald_df['time']=vald_df['time'].apply(lambda x:isinstance(x,datetime.datetime))
vald_df['item']=vald_df['item'].apply(lambda x:isinstance(x,str))
vald_df['cost']=vald_df['cost'].apply(lambda x:isinstance(x,int))
vald_df['qty']=vald_df['qty'].apply(lambda x:isinstance(x,int))
vald_df['urgent']=vald_df['urgent'].apply(lambda x:isinstance(x,int))
print(f'[Step 3] 데이터 형식을 벗어나는 데이터 수: {len(df)-len(vald_df)}')
```

[Step 3] 데이터 형식을 벗어나는 데이터 수: 0

[코드 46] 유효성 품질 지수 확인 코드 (3)

## ④ 유효성 품질지수 값은 다음과 같다.

```
vald_df[
    (vald_df['time']==True)
    &(vald_df['item']==True)
    &(vald_df['cost']==True)
    &(vald_df['qty']==True)
    &(vald_df['urgent']==True)
]
vald_len = len(vald_df)
item_vald = vald_len/len(df)*100
print("order_info 유효성 지수 : %.2f%%"%(item_vald))
```

order\_info 유효성 지수 : 100.00%

[코드 47] 유효성 품질 지수 확인 코드 (4)



⑤ 최종 유효성 품질 지수는 모든 데이터 유효성 품질지수의 평균이다.

```
print("유효성 지수: %.2f%%" % ((item_vald + machine_vald)/2))
```

유효성 지수: 100.00%

[코드 48] 유효성 품질 지수 확인하기

#### ▶ 일관성 품질 지수 = (일관성 만족 데이터수/전체 데이터 수)\*100:

- order\_info는 자식 테이블이며, machine\_info는 부모 테이블이다. order\_info의 'item' 열은 machine\_info의 'item' 열에 종속된다. 즉, order\_info의 모든 'item'은 machine\_info 'item' 중 하나여야 한다.

```
# 4. 일관성 품질지
수 df = order_info
vald_df = df[['item']].copy()
vald_df['check'] = vald_df['item'].isin(set(machine_info['item'])) print("
일관성 지수: %.2f%%" % (sum(vald_df['check'])/len(df)*100))
vald_df
```

일관성 지수: 100.00%

[코드 49] 일관성 품질 지수 확인하기

#### ▶ 정확성 품질 지수 = (1-(정확성 위배 데이터 수/전체 데이터 수))\*100

- 모든 열값은 독립적이므로 정확성 지수를 확인하지 않음

#### ▶ 무결성 품질지수 = (1-(유일성, 유효성, 일관성 지수중 100%가 아닌 지수 개수 / 3))\*100

- order\_info 데이터는 세 가지 지수중 유일성 지수가 100%를 만족하지 못하므로 (99.16%) 무결성 품질 지수는 66.66%이다. machine\_info의 경우, 일관성 지수는 고려하지 않는 대 상이며, 유일성과 유효성 지수는 모두 100%이므로 무결성 품질 지수는 100%이다. order\_info와 machine\_info의 평균 무결성 품질 지수는 83.35%이다.

#### ▶ 가중치 지수 = 품질 지수 \* 가중치

#### ▶ 데이터 품질 지수 = ∑ 가중치 지수

구 분	품질지수	가중치	가중치 지수	오류율
완정성 (누락)	99.99%	60%	59.99%	0.01%
유일성 (중복)	99.16%	10%	9.92%	0.08%
유효성 (유효)	100%	10%	10%	0%
일관성 (표현)	100%	10%	10%	0%
정확성 (실제)	-	-	-	-
무결성	83.35%	10%	8.34%	1.66%
품질지수		100%	98.25%	1.75%

[표 9] 데이터 품질지수