「공급망 최적화 AI 데이터셋」 분석실습 가이드북

중소벤처기업부

KOSMO 스마트제조혁신추진단

KAIST

「공급망 최적화 AI 데이터셋」 분석실습 가이드북

중소벤처기업부

KOSMO 스마트제조혁신추진단

KAIST

Contents

「공급망 최적화 AI 데이터셋」 분석실습 가이드북

- **필 요 S W :** Python, Anaconda – Jupyter Notebook
- **필요 패키지 :** Pandas, NumPy, Matplotlib, scikit-learn, TensorFlow
- **분 석 환 경 :** [CPU] Intel i5 3.1 GHz, [RAM] 40GB
- **필요 데이터 :** data.xls

1 분석요약

No	구 분		내 용
1	**분석 목적** (현장 이슈, 목적)		– 모기업에 부품을 납품하는 협력기업은 납품된 부품의 사용량을 알 수 없기 때문에 경험을 기반으로 생산 계획을 세우고 있다. – 이 때문에 협력기업의 생산 계획 및 재고관리는 경험이 많은 특정 소수의 인력에 의존하게 되며, 발주 수량 예측이 맞지 않았을 때 재고량이 크게 늘어나거나 부족한 비효율적 생산 체계를 가지게 된다. – 이러한 경우 모기업의 생산량 및 발주 수량의 잦은 변동에 대한 대처가 어렵고, 이는 협력사에 2차, 3차 재고 부담을 발생시켜 경영 악화를 야기하게 된다. – 따라서 생산성 향상, 납기일 준수 및 재고관리를 위해 발주 및 계획 발주 수량 데이터를 분석하여 발주 수량을 미리 예측할 수 있는 AI 기반의 수량 예측 시스템이 필요하다.
2	**데이터셋 형태 및 수집방법** (csv, json, image 파일 등)		**1) 분석에 사용된 변수명 :** Part Number, CRET_TIME, 발주 수량, D+X 계획 발주 수량 외 15개 **2) 수집 방법 :** 생산업체의 Internet Data Center 내의 발주 및 계획 발주 수량 데이터 **3) 데이터셋 파일 확장자 :** .xls
3	**데이터 개수** **데이터셋 총량**		– **데이터 개수 :** Row 수 (17,364개), Column 수 (84개) – **데이터셋 총량 :** 9.7 MB
4	**분석적용 알고리즘**	**알고리즘**	장단기 메모리(Long-Short Term Memory, LSTM)
		알고리즘 간략소개	장단기 메모리는 인공 신경망의 한 부류인 순환 신경망(Recurrent Neural Network, RNN)의 일종으로서, 시계열적 특성을 지닌 데이터를 활용하여 예측 혹은 분류 문제를 해결하는 데에 사용된다. 일반적인 다중 퍼셉트론과 같은 여러 층의 완전연결층을 가지지 않고, 동일한 신경망 층이 루프 형태로 배치되어 시간 순으로 데이터를 순환하며, 모델 내의 은닉상태(메모리)를 반복적으로 갱신하며 학습을 진행한다. 특히 입력, 출력, 망각 게이트를 활용하여 기울기 소실/폭발 문제를 해결하기 때문에 시계열 데이터의 장기간 패턴을 학습하는 데에 적합하다.
5	**분석결과 및 시사점**		– 장단기 메모리를 활용하여 분석 및 예측을 실시한 결과 미래 발주량에 대한 장기적인 예측이 가능함을 확인하였다. – 실제 발주 수량 데이터 및 각 시점별 계획 발주 수량을 통합적으로 고려하여 미래 발주 수량을 예측했을 때 장단기 메모리 알고리즘이 높은 성능을 보임을 확인하였다. – 또한, 단일 부품 데이터뿐만 아니라 특정 부품과 연관성을 가지는 다른 부품의 다변량 시계열 데이터를 동시에 활용하여 예측 수행 시, 해당 알고리즘이 보다 높은 표현력을 가지는 학습을 진행함을 확인하였다.

2 분석 실습

1. 분석 개요

1.1 분석 배경

1) 공정(설비) 개요

• Vendor Management Inventory (VMI)

- VMI는 납품업자(공급사, 협력기업)가 모기업을 대신하여 사전에 합의한 최소 및 최대 재고 수준과 수요 예측에 기초하여 재고를 모니터링하고 계획하며 관리하는 공급망 관리 방법이다.

• VMI의 활용

- VMI는 다음과 같은 산업 및 제품 제조 공정에 효과적으로 사용된다.

① 실수에 민감한 산업

② 경쟁이 심하고 이윤이 적은 산업

③ 상하기 쉬운 특성의 제품

④ 수요를 예측하기 어려운 상품

⑤ 많은 매장 혹은 빠르게 움직이는 소비자 제품

• VMI의 장점

- 공급망 관점

  전체 공급망 수준 대비 낮은 수준의 재고 유지가 가능하다. 이로 인해 간접비가 감소하고 판매가 증가할 수 있다. 또한 작업자의 자료 입력에 따른 오류가 감소한다.

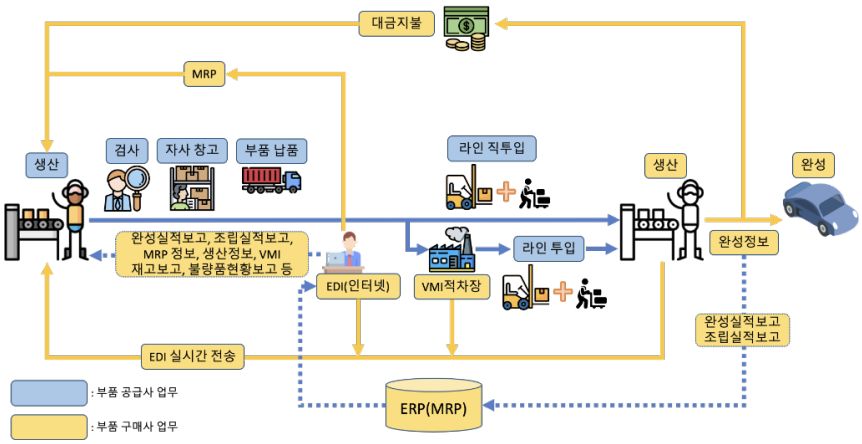
- 공급자 관점

  VMI는 고객 요구(needs)에 대한 더 나은 통찰력을 부여하며, 고객과의 직접적인 소통 및 시장 분석을 개선하는 데에 도움이 된다. 제품 항목(카테고리) 관리 및 부가서비스 제공 기회 및 재고 보충 시간의 감소 등의 장점으로 인해 전략적, 강한 공급망 관계 구축이 가능하다.

• VMI의 한계 및 단점

- 고객 요구의 불확실성으로 인해 협력기업(공급사)의 재고 부담이 가중될 수 있으며, 수시로 변경되는 계획으로 인해 공급사의 물류비가 증대될 수 있다.

• VMI 업무 워크 플로우



[그림 1] VMI 흐름도

2) 이슈사항(Pain point)

• **공정(설비)상의 문제 현황**

- 모기업(판매사, 고객사)은 협력기업(공급사)에 일자별로 소요될 품목별 발주 수량을 매일 전달함과 동시에 향후 특정 기간(예 : 2주)의 계획 발주 예상량을 전달한다. 매일 전달되는 당일의 발주 수량은 신뢰할 수 있고 이를 고려한 생산 계획을 수립할 수 있으나, 며칠 이후 발주된 수량에 대해서는 협력기업의 경험을 기반으로 생산 계획을 수립하여 제품을 생산하고 있다.
- VMI의 중요한 요구조건으로는 모기업(판매사, 고객사)이 언제든 고객에게 제품을 판매할 수 있도록 협력기업이 재고를 관리하는 것이 필요하다는 점이다. 모기업에 부품을 납품하기 전까지는 재고 관리의 책임과 비용 부담이 모두 협력기업에게 있다.

• **문제해결 장애요인**

- 모기업은 자사의 정보를 협력기업에 제공함으로써 유기적인 재고 관리가 될 수 있도록 한다. 하지만, 모기업이 협력기업에게 정확한 발주 수량을 제공하지 않을 경우 협력기업은 과잉 생산으로 발생하는 재고 부담에 어려움을 겪을 수 있다.
- 또한, 반대로 갑작스런 수요 혹은 발주량 변화로 인하여 협력기업이 충분한 물량을 준비하지 못하여 재고 부족 상태가 발생할 수 있다.
- 이상적으로 협력기업은 모기업과의 절대적인 신뢰관계를 구축하여 모기업의 수요 정보를 기반으로 재고를 관리해야 한다. 하지만 실제 산업 현장에서는 사전 수요 예측 및 발주 수량에 대한 예측 정보의 정확도가 낮아 재고 소진이 원활하게 이루어지지 못하는 실정이다.

• **극복 방안**

- 전통적으로 재고관리 혹은 생산량 예측에는 ARIMA (AutoRegressive Integrated Moving Average)와 같은 시계열 분석법과 베이지안 선형 회귀 같은 회귀분석법을 사용하였다. 하지만, 최근에는 의류 제조사, 카드사 등의 업체에서 인공신경망(Artificial Neural Network, ANN)을 활용한 수요 예측 기법을 이용하여 예측의 성능을 크게 향상시킬 수 있었다. 이처럼 AI 기반 분석모델을 모기업의 발주 및 계획 발주 수량 데이터에 적용할 경우 VMI 및 재고 관리 최적화에 큰 도움이 될 것으로 기대된다.

1.2 분석 목표

1) 분석 목표

- 본 분석에서는 모기업의 당일 발주량 뿐만 아니라 계획 발주량을 동시에 고려하여 정량적으로 미래 시점의 발주 수량을 예측할 수 있는 AI 분석모델을 개발하고자 한다. 정확한 AI 예측 모델을 개발하여 부품별로 발주 수량을 예측함으로써 합리적인 재고 및 인적 자원 관리에 도움을 주고자 한다.

2) 데이터 정의 및 소개

- 본 분석에서 활용되는 제조 데이터는 VMI 재고 관리 방식을 사용하는 협력기업과 모기업의 발주 수량 정보 등을 일자 및 시간별로 수집한 데이터이다. 일자별로 특정 시간(오전, 오후)마다 변경된 발주량 및 계획 발주 수량 정보가 기록되며, 협력기업에서 생산하는 여러 부품 별로 인스턴스(instance)화 되어있다. 일자별 당일 발주 수량 외에도 1일 이후, 2일 이후 등 향후 계획 발주 수량 정보도 데이터에 포함되어 있다.

3) 데이터 분석 기대효과

- 본 분석은 다양한 형태로 존재하는 다변량 시계열 형태의 제조 데이터에 AI 분석모델을 적용함으로써 부품별 발주 수량 예측이 가능한 모델을 제공한다. 또한, AI 분석모델 구축 방법 및 학습용 데이터로의 변환 등의 실제 산업 현장에서 쉽게 응용가능한 시계열 데이터 기반 분석 방법을 제공한다.

4) 시사점(implication) 요약기술

- 현재 모기업의 불규칙한 발주 수량 및 변동적인 계획 발주 수량은 물품을 납품하는 협력기업의 생산 계획 수립 및 재고 관리에 차질을 주고 있다. 이를 개선하기 위해 과거의 발주 수량 시계열 데이터를 활용하여 AI 분석모델을 사용한 발주 수량 예측을 수행하는 분석을 진행한다.

- 본 분석에서는 실제 발주 수량 및 계획 발주 수량을 고려하여 실제 산업현장에서 적용될 수 있는 수요 예측 모델을 개발한다. 이를 바탕으로 협력기업들의 개선된 공급망에서는 생산성 향상, 품질 향상, 납기일 준수, 재고 관리 및 수급이 최적의 수준으로 유지되도록 한다.
- 협력기업들의 공급망을 개선하여 효율적인 생산 시스템을 기대할 수 있으며, 이는 협력기업 뿐만 아니라 모기업의 생산량에도 긍정적인 영향을 미칠 것으로 판단된다. 본 분석은 실제 산업 현장에서 적용 가능한 수량 예측분석을 함으로써 산업의 문제를 해결하는 데에 기여했다는 점에서 시사점이 크다고 판단된다.

2. 분석실습

2.1 제조데이터 소개

1) 데이터 수집 방법

- 제조 분야 : 공급망 및 발주 관리
- 제조 공정명 : VMI 발주
- 수집 장비 : Enterprise Resource Planning (ERP) 시스템
- 수집 기간 : 2021년 9월 13일 ~ 2021년 11월 1일
- 수집 주기 : 평균 10시간

2) 데이터 유형/구조

- 데이터셋 구조 : 테이블 형식(tabular)
- 데이터 개수 : column 수 84개, row 수 17,364개, 데이터수 1,458,576개
- AI 데이터셋 주요 변수(속성) 정의
  데이터셋에는 일자, 시간 및 부품 식별자(비식별화됨)와 다양한 발주 수량 정보가 포함되어 있으며 주요 변수는 아래 표와 같다. 발주 수량 관련 변수를 독립 변수로, 당일 발주 수량을 종속 변수로 활용할 것이다.

속성(column)	설명	비고
Part Number	협력기업에서 모기업에 납품해야하는 부품의 식별자(ID)	비식별화됨 데이터형 : string
CRET_TIME	특정부품의 발주 수량이 기록된 해당 시각 정보	연도, 월, 일, 시각(24시 기준) 데이터형 : datetime
D일 X~XH투입 계획 수량	당일(D일)의 시각별 발주 및 계획 발주량	단위 : 개수 데이터형 : int
D일 투입 예정 수량	당일(D일)의 통합 발주 및 계획 발주량	단위 : 개수 데이터형 : int
D+X일 투입 예정 수량	당일(D일)의 기록된 X일 이후의 계획 발주량	단위 : 개수 데이터형 : int

- 주요 변수 기술 통계

속성(column)	데이터형	개수	평균	표준편차	최소값	중앙값	최대값
D일 투입예정 수량(D일계획)	int	17,364	101.311	168.014	0.000	30.000	1131.000
D일06~08(08)H 투입계획(발주) 수량	int	17,364	27.647	49.082	0.000	8.000	404.000
D일13~15(15)H 투입계획(발주) 수량	int	17,364	11.155	18.823	0.000	3.000	123.000
D일23~01(02)H 투입계획(발주) 수량	int	17,364	6.340	11.088	0.000	1.000	65.000
D+1일 투입예정 수량(Total)	int	17,364	67.076	131.268	0.000	5.000	883.000
D+2일 투입예정 수량(Total)	int	17,364	48.898	115.360	0.000	0.000	885.000
D+3일 투입예정 수량(Total)	int	17,364	50.639	114.235	0.000	0.000	872.000
D+4일 투입예정 수량(Total)	int	17,364	59.657	122.522	0.000	1.000	865.000
D+7일 투입예정 수량	int	17,364	89.968	146.443	0.000	29.000	912.000
D+12일 투입예정 수량	int	17,364	71.767	135.339	0.000	5.000	794.000
D+31~D+45일 투입예정 수량	int	17,364	0.576	10.607	0.000	0.000	320.000

3) 데이터 (품질) 전처리

• 데이터 품질 전처리 목적

- 실제 공정에서 발생하는 데이터는 의미 없는 값을 가지거나 누락 및 오타의 발생으로 데이터 품질이 떨어지는 경우가 빈번하다.
- 품질이 낮은 데이터를 이용하면 좋은 분석 방법론을 이용하더라도 좋은 결과를 도출하기 어렵기 때문에, 데이터 품질 전처리는 데이터 분석에서 필수적인 단계이다.
- 따라서 데이터의 6가지 품질 지수를 파악하고, 데이터 전처리를 통해 품질 지수를 향상하는 것이 데이터 전처리의 목적이라고 할 수 있다.

• 데이터 품질 지수

- 제조 현장에서 수집되는 데이터 및 AI 분석모델을 적용하기 위한 데이터를 가공하기 전 품질 확인이 우선시 되어야 한다. 다량의 데이터를 확보하는 것도 중요하지만, 데이터 사이언스 분야에서 쓰이는 표현인 'garbage in, garbage out'이 의미하는 것처럼 결함이 있는 입력 데이터를 이용하면 좋지 않은 결과를 도출할 수 있다. 그러므로 데이터의 품질이 전제되었을 때 AI 분석모델의 우수한 성능을 기대할 수 있다.
- 발주 데이터의 경우 결측치 제거 등의 데이터 전처리를 통해 데이터의 품질을 제고할

수 있다. 본 연구에서는 데이터 품질 확인을 위해 Gartner에서 제시한 아래와 같은 6종의 데이터 품질 평가지표를 고려하였다.

① 완전성(Completeness) : 필수항목에 누락이 없어야 한다.

② 유일성(Uniqueness) : 데이터 항목은 유일해야 하며 중복되어서는 안 된다.

③ 유효성(Validity) : 데이터 항목은 정해진 데이터 유효 범위 및 도메인을 충족해야 한다.

④ 일관성(Consistency) : 데이터가 지켜야 할 구조, 값, 표현되는 형태가 일관되게 정의되고, 동일한 데이터 간에 불일치가 발생하지 않아야 한다.

⑤ 정확성(Accuracy) : 실제 존재하는 객체의 표현 값이 정확히 반영되어야 한다.

⑥ 무결성(Integrity) : 데이터 처리의 선후 관계가 명확하게 준수되고 있어야 한다.

• **데이터 품질 지수 세부 설명**

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

① 데이터의 결측치를 확인하기 위해 pandas 패키지의 'isna()' 와 'sum()' 메서드를 이용하여 총 결측치 개수를 구한다.

② 결측치의 개수를 이용하여 완전성 품질 지수를 계산한다.

- 유일성 품질 지수 = (1-(중복 데이터 수 / 전체 데이터 수)) X 100본 실습의 데이터는 'CRET_TIME' 열을 기준으로 유일한 값을 가지는지 판단한다.

- 유효성 품질 지수

① 데이터가 유효 범위 내에 있는가?

② 데이터가 올바른 형식으로 구성되어 있는가?

③ 수집된 날짜 안에 들어가 있는가?

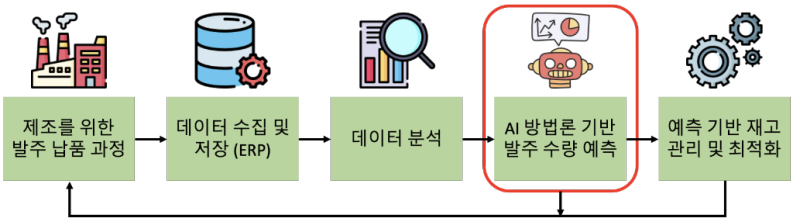
- 일관성 품질 지수 = (일관성 만족 데이터 수 / 전체 데이터 수) X 100

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

- 무결성 품질 지수 = (1-(유일성, 유효성, 일관성 지수 중 100%가 아닌 지수 개수 / 3)) X 100

2.2 분석모델 소개

1) 데이터 흐름 및 AI 분석모델 적용 흐름도



[그림 2] 제조 발주 데이터를 활용한 AI 분석모델 적용 흐름도(프레임워크)

2) AI 분석모델

• 제조 발주 데이터 분석 방법

- 시계열 예측이란?

  시계열(time-series) 데이터는 일정한 기간 동안 특정 시간 간격으로 관측(수집)된 시간적 순서를 가진 데이터를 말한다. 수집된 시계열 데이터를 활용하여 미래에 발생할 사건 혹은 값 등을 예측하는 것을 시계열 예측이라 한다. 데이터가 특정한 시간 단위(time step) 마다 하나의 수집된 값을 가지면 단변량(univariate) 시계열, 여러 값을 가지면 다변량(multivariate) 시계열로 정의한다.

- 시계열 특성을 가진 제조 발주 데이터의 분석

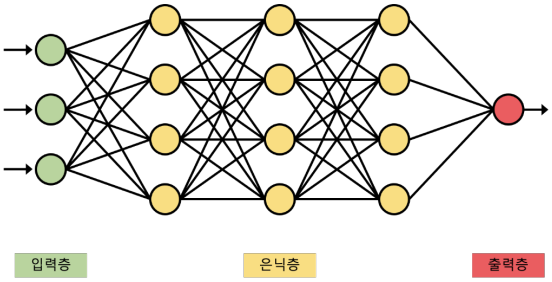
  발주 데이터의 특성상 당일 발주 수량뿐만 아니라 향후 1일부터 30일까지 이내의 계획 발주 수량의 정보를 포함하고 있다. 또한 부품별로 데이터를 분석할 때, 일자별로 다양한 발주 정보가 순차적으로 존재하는 다변량 시계열 데이터 형태로 볼 수 있다. 미래, 즉 예측 시점으로부터 특정 일자 이후의 발주 수량을 예측하는 것이 본 분석의 목적이므로 과거의 발주 다변량 시계열 데이터를 활용하는 AI 분석모델을 구축한다.

- AI 모델을 활용한 분석 목적

  단변량 시계열 및 다변량 시계열 분석 시 다수의 통계 기반 방법론은 인위적인(handcrafted) 특성 추출 및 변환 단계를 요구한다. 반면 인공신경망(Artificial Neural Network, ANN) 기반 방법론은 이러한 특성 추출을 자동적으로 수행할 수 있으며, 특히 본 분석에서 사용될 장단기 메모리의 경우 시계열 특성뿐만 아니라 장기적인 의존성 또한 학습할 수 있다. 따라서 AI 분석모델을 활용하여 발주 수량을 예측할 경우, 발주 관련 다양한 특성을 통합적으로 고려하는 분석을 실시할 수 있다.

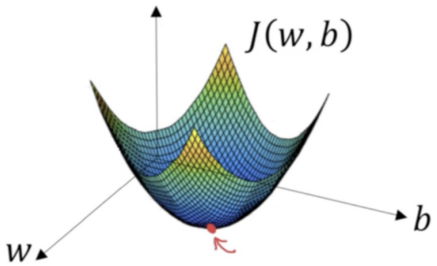
• 인공신경망 기반 알고리즘

- 인공신경망은 신경망이라 불리는 구조를 연쇄적으로 쌓은 후, 데이터를 활용하여 신경망을 구성하는 모수(parameter)를 학습시키는 모델이다. 일반적으로 신경망은 [그림 3]과 같이 입력층(input), 은닉층(hidden), 출력층(output)으로 구성되며 각 층은 여러 노드(node)를 가진다. 은닉층은 입력층으로부터 데이터를 받아 연속적인 연산(행렬곱 등)을 수행하고 출력층을 거쳐 최종 출력 값을 배출한다. 신경망의 층 사이에는 활성화 함수(activation function)를 사용하여 선형 및 비선형 변환을 적용할 수 있으며, 이러한 점 덕분에 인공신경망을 심층적으로 구성 시 데이터에 내재된 복잡한 특성 등을 모델링할 수 있다.



[그림 3] 인공신경망의 예시

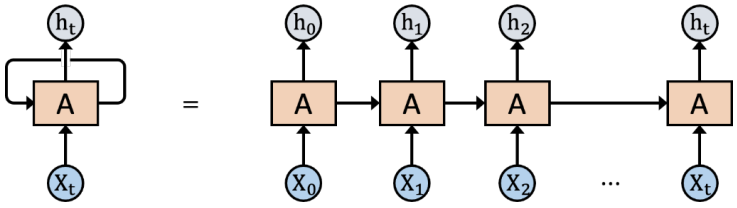
- 데이터를 사용하여 인공신경망을 학습시킨다는 것은 데이터를 모델의 입력과 출력으로 사용하여 인공신경망으로 하여금 입력과 출력 사이의 관계를 모방하는 함수를 학습하도록 하는 것이다. 학습 시에는 인공신경망 내 여러 모수들의 최적값을 찾게 되는데, 이 과정에서 일반적으로 경사 하강법(gradient descent)을 사용한다. 신경망의 예측값과 데이터의 실제값의 차이는 손실 함수(loss function)를 사용하여 계산되며 이를 학습 지표로도 볼 수 있다. 경사 하강법은 손실 함수의 편미분 값을 사용하여 모수의 값을 손실 함수 값이 감소하는 방향으로 반복적으로 변화시키는 방법이다. 편미분 값을 구하는 과정에서 오류 역전파(error backpropagation)를 사용하는데, 연쇄 법칙(chain rule)에 따라 신경망 내의 연결되어 있는 모든 모수에 대한 편미분 값을 계산하게 된다. 이러한 과정을 통해 신경망의 모수는 점차 최적의 값에 가까워지게 되고, 이를 신경망 학습이라 한다.



[그림 4] 경사 하강법(gradient descent)
〈출처:https://builtin.com/data-science/gradient-descent〉

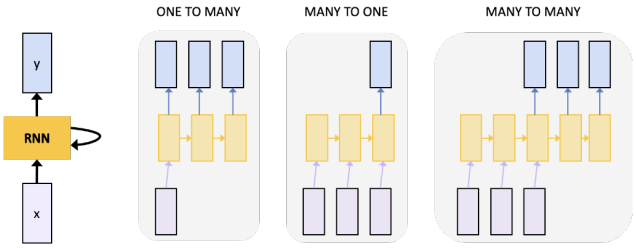
- 순환 신경망(Recurrent Neural Network, RNN)
순환 신경망은 본 분석에서 이용하고자 하는 시계열 데이터와 같이 순서가 있는 데이터 즉, 시퀀스(sequence)를 처리하는 인공신경망 기반 심층학습(deep learning) 방법론 중의 하나이다. 순환 신경망은 일반적인 인공신경망 구조와 동일하게 입력층, 은닉층, 출력층으로 구성되어 있다. 그러나 각 신경망 층이 연쇄적으로 완전연결되어 있지 않고, 동일한 신경망 층이 루프 형태로 배치되어 시간 순으로 데이터를 순환하는

구조를 가진다. [그림 5]의 왼쪽 형태와 같이 순환 신경망에서는 기준 시점(t)에서 계산된 출력값이 다시 다음 시점(t+1)의 입력으로 사용되는데, 이러한 순환 경로를 가짐으로써 시간의 흐름에 따라 데이터의 특성을 학습할 수 있다. [그림 5]의 오른쪽 그림과 같이 순환 신경망을 펼쳐보면, 각 시간 단계마다 한 시점(t)의 입력(x_t)과 이전 시점(t-1)의 은닉층(h_{t-1})을 입력값으로 사용하고, 이후 출력은 다시 다음 시간 단계에서 입력으로 사용되는 순환적인 구조를 볼 수 있다. 이러한 구조는 이전의 정보를 은닉층을 통해 저장하고 사용할 수 있게 한다. 모델 학습 과정에서는 일종의 메모리 역할을 하는 은닉층의 상태가 반복적으로 갱신되면서 신경망이 시계열의 주요한 패턴을 학습하게 된다.



[그림 5] 순환 신경망(RNN)

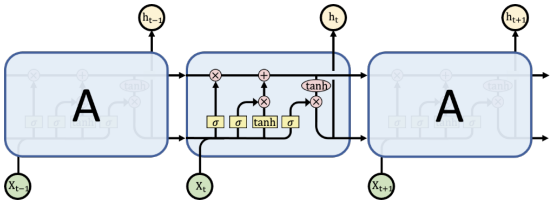
- 순환 신경망은 경사 하강법과 시간 펼침 역전파(BackPropagation Through Time, BPTT)를 활용하여 학습을 진행한다. 시간 펼침 역전파란 출력에서 입력 방향으로, 즉 시간 축을 거꾸로 올라가는 방향으로 오류 역전파를 계산하는 경사 하강법 기반의 방법이다. 이는 [그림 5]의 우측 부분과 같이 임의로 순환 신경망의 구조를 시간 순으로 펼친 뒤 일반적인 신경망에서의 오류 역전파를 적용하는 것과 동일하다. 다른 점은 순환 신경망이 모수를 공유하는 단일 신경망 층을 사용하기 때문에 동일한 모수들에 대한 여러 시간 단계에서의 오차를 역전파하게 된다는 것이다. 순환 신경망은 [그림 6]과 같이 다양한 입출력의 형태를 활용할 수 있으므로, 시계열 데이터 예측뿐만 아니라 사진 설명 생성, 감성 분석, 번역, 영상 분류 등의 다양한 분야에서 사용되고 있다. 하지만, 입력과 출력 지점 거리가 멀 경우에 시간 펼침 역전파 과정에서 기울기가 사라지거나 폭발하여 학습 및 예측 능력이 저하될 수 있다. 즉, 데이터의 시퀀스가 길어질수록 데이터 내 거리가 먼 과거의 정보를 학습 과정에서 반영하지 못하는 장기 의존성(long-term dependency) 문제가 발생할 수 있다. 이를 해결하기 위한 대표적인 방법으로 장단기 메모리(Long-Short Term Memory, LSTM)와 게이트 순환 유닛(Gated Recurrent Unit, GRU)과 같은 변형된 순환 신경망이 있으며, 장기 의존성을 잘 학습할 수 있도록 고안되었다.



[그림 6] 순환 신경망(RNN)의 다양한 활용 형태

- 장단기 메모리(Long-Short Term Memory, LSTM)

앞서 설명한 것처럼 장단기 메모리는 순환 신경망의 장기 의존성 문제를 해결하기 위해 만들어진 대표적인 순환 신경망의 변형 알고리즘이다. 장단기 메모리는 크게 메모리 셀(memory cell)과 3개의 게이트(gate)로 구성되어 있다. 순환 신경망과 구분되는 장단기 메모리의 핵심은 메모리 셀이라고 할 수 있으며, 이전 시간 단계의 정보가 메모리 셀에 저장된다. 이후 입력, 망각, 출력의 정도를 조절하는 총 3개의 게이트를 통해 정보가 선택적으로 메모리 셀에 업데이트되면서, 기울기 소실 문제를 방지함과 동시에 장기 의존성을 학습한다. 장단기 메모리는 [그림 7]과 같은 구조를 가지고 있다. 장단기 메모리의 내부 구조에 대한 자세한 설명은 다음 장에서 기술한다.



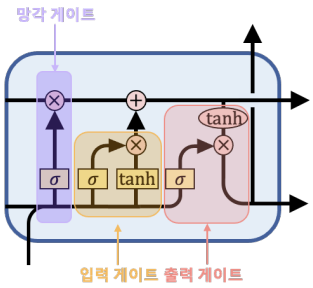
[그림 7] 장단기 메모리(LSTM)의 구조

• 해당 AI 방법론(알고리즘) 선정 이유 기술

- 제조 데이터 분석 알고리즘 : **장단기 메모리(LSTM)**

본 분석에서는 장단기 메모리 알고리즘을 기반으로 다변량 시계열 데이터를 활용한 발주 수량 예측 모델을 구축한다. 앞서 설명한 대로, 장단기 메모리는 순환 신경망의 일종으로서, 시간 정보가 담긴 데이터를 이용한 예측 혹은 분류 문제를 해결하는 데에 활용된다. 또한, 메모리 셀, 입력, 출력 및 망각 게이트로 구성된 구조로 인해 기존 순환 신경망의 기울기 소실/폭발 문제를 해결하며 긴 시퀀스를 성공적으로 학습에 활용할 수 있다. 따라서, 생산성 향상, 납기일 준수 및 최적의 재고 관리를 위해 시계열 특성을 가진 제조 데이터를 활용한 AI 기반의 발주 수량 예측 시스템에 장단기 메모리 알고리즘을 적용하는 것이 적절하다고 사료된다.

• **적용하고자 하는 AI 분석 방법론(알고리즘)의 구체적 소개**



[그림 8] 장단기 메모리(LSTM)의 구조

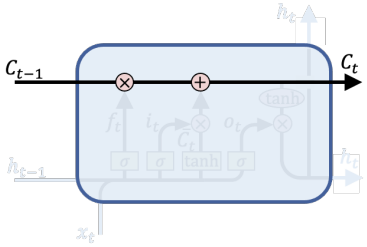
- 장단기 메모리는 [그림 7], [그림 8]과 같이 망각 게이트, 입력 게이트, 출력 게이트, 그리고 은닉 상태(h_t)와 메모리 셀(C_t)로 구성된다. 은닉 상태(h_t)는 단기 상태라고도 부르며, 메모리 셀(C_t)은 장기 기억을 저장한다는 점에서 장기 상태라고 부르기도 한다. 3개의 게이트 구조는 장단기 메모리 내 정보의 흐름을 제어하는 역할을 담당하는데, 각 게이트의 역할은 다음과 같다. 아래 3개의 게이트는 0부터 1까지의 값을 취하며, 밸브와 같은 역할을 한다고 볼 수 있다.

  ① 망각 게이트(forget gate, f_t) : 이전 단계의 메모리 셀의 과거 정보를 얼마나 삭제할지 결정한다.

  ② 입력 게이트(input gate, i_t) : 현재 시간 단계에서의 새로운 정보를 추가하는 정도를 조절한다.

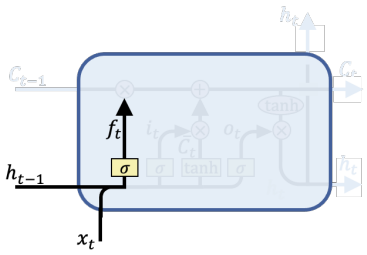
  ③ 출력 게이트(output gate, o_t) : 계산된 메모리 셀의 정보를 활용하여 최종 출력인 은닉 상태를 제어하는 역할을 한다.

- [그림 9]는 이전 단계(t-1)의 메모리 셀(C_{t-1})에서 새로운 단계(t)의 메모리 셀 (C_t)로 업데이트 되는 과정을 보여준다. 망각 게이트와 입력 게이트를 통해 삭제할 정보, 추가할 정보를 결정하여 메모리 셀을 업데이트한다. 동시에, 새로운 단계(t)의 메모리 셀(C_t)의 정보를 은닉 상태(h_t)로 얼마나 출력할지 출력 게이트를 통하여 결정한다. 즉, 메모리 셀은 컨베이어 벨트와 같이 3개의 게이트(f_t, i_t, o_t)를 통해 보존할 정보, 삭제할 정보, 출력할 정보를 업데이트한다. 다음은 장단기 메모리의 동작 단계에 대한 설명이다.

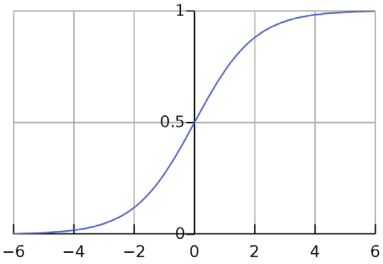


[그림 9] 메모리 셀

① 망각 단계 : 메모리 셀에 저장된 과거의 정보를 제거할지 혹은 유지할지를 결정하는 단계다. 이전 단계의 은닉 상태(h_{t-1})와 현 단계의 입력(x_t)을 아핀 변환(affine transformation) 후 시그모이드 함수를 취한다. 시그모이드 함수의 출력 범위는 [그림 11]과 같이 0에서 1사이로 제한된다. 출력값이 0에 가까우면 이전 정보를 대부분 제거하고, 1에 가까우면 이전 정보를 유지하게 된다. 기존 메모리 셀과 아다마르 곱(Hadamard product) 연산을 수행하게 된다.

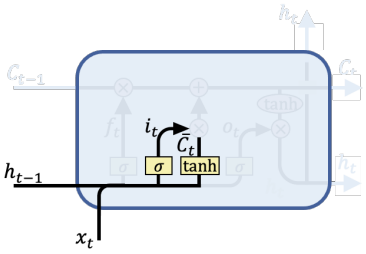


[그림 10] 망각 단계



[그림 11] 시그모이드 함수

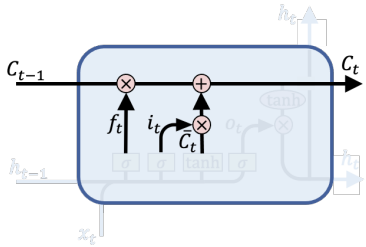
② 입력 단계 : 메모리 셀에 어떤 새로운 정보를 저장할지 결정하는 단계이다. 입력 단계에서는 이전 단계의 은닉 상태(h_{t-1})와 현 단계의 입력(x_t)을 아핀 변환 후 시그모이드를 함수를 취한다. 추가 정보로서 가치가 크다고 판단되면 시그모이드 값이 1에 가깝게, 적다고 판단되면 0에 가깝게 된다. 현재 시간 단계에서 추가할 새로운 정보를 후보 메모리 셀(candidate memory cell, $\tilde{C}_t$)이라 하는데, 입력 게이트와 같이 이전 단계의 은닉 상태(h_{t-1})와 현 단계의 입력(x_t)을 아핀 변환(affine transformation) 하되, 이후 하이퍼볼릭 탄젠트(tanh) 함수를 취한다. 이렇게 구한 후보 메모리 셀과 입력 게이트의 출력값의 아다마르 곱을 취한 결과를 이후 메모리 셀 업데이트 시에 사용한다.



[그림 12] 입력 단계

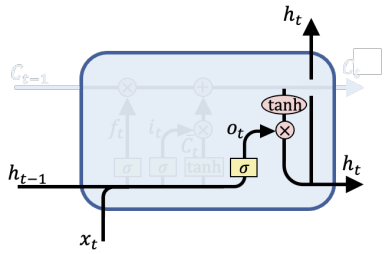
③ 메모리 셀 업데이트 단계 : 이전 단계의 메모리 셀(C_{t-1})을 새로운 메모리 셀(C_t)로 업데이트하는 단계이다. 메모리 셀 업데이트 단계는 [그림 13]에 나타나 있다. 먼저, 이전의 메모리 셀(C_{t-1})과 망각 단계의 결과값(f_t)의 아다마르 곱을 계산한다. 그리고 입력 단계에서 계산한 입력 게이트 결과값과 후보 메모리 셀의 아다마르 연산값을 더하여 새로운 메모리 셀(C_t) 값으로 결정한다. 즉, 메모리 셀은 $(f_t \odot C_{t-1})+(i_t \odot \tilde{C}_t)$의 값으로

업데이트 된다. 이는 과거의 정보와 새로 계산된 정보의 가중합으로 볼 수 있다.



[그림 13] 메모리 셀 업데이트 단계

④ 출력 단계 : 어떤 값을 출력할지 결정하는 단계이다. [그림 14]와 같이 이전 단계의 은닉 상태(h_{t-1})와 현 단계의 입력(x_t)을 아핀 변환 후 각 원소에 시그모이드 함수를 취한다. 그 후, 계산된 현시점의 메모리 셀과의 아다마르 곱을 통해 현재의 은닉 상태(h_t)를 계산하고 출력한다.



[그림 14] 출력 단계

- 이러한 단계들을 통해 과거의 정보를 선택적으로 업데이트하고, 새로운 정보를 자동적으로 선택하여 메모리 셀을 통해 저장함으로써 기울기 소실 문제 및 장기 의존성 문제 해결을 가능하게 한다.

• AI 분석 방법론(알고리즘) 구축 절차 설명

- 발주 수량 데이터를 활용하여 AI 분석모델을 적용하기 위해서는 적합한 형태로 데이터셋을 변환하는 것이 필요하다. 전처리 과정을 통하여 수집된 데이터에 대해 정제 및 필터링을 진행한다. 이후 데이터셋을 학습-검증-평가 데이터로 분리하여 일련의 학습 과정 내에서 목적에 부합하게 사용한다. 본격적인 AI 분석 방법론 구축으로는, 선택한 알고리즘(장단기 메모리)을 정의하는 과정이 요구된다. 인공신경망 기반 모델의 경우, 조정할 수 있는 초매개변수(hyper-parameter)가 존재한다. 예를 들어, 장단기 메모리의 경우 은닉층의 노드 개수, 활성화 함수, 규제항 및 손실 함수의 종류 등을 사용자의 선택에 따라 다양하게 조정할 수 있다. 따라서 먼저 적합한 초매개변수를 설정함으로써 모델을 정의한다. 이후 준비된 데이터셋을 활용하여 학습을 진행함과 동시에 모델 성능을 검증한다. 필요시에는 모델 성능 제고를 위해 모델을 재정의하거나 변경할 수 있다. 최종적으로 학습이 종료된 모델을 저장하고 평가용 데이터셋을 사용하여 성능을 평가한다.

2.3 분석 체험

1) 필요 SW, 패키지 설치 방법 및 절차 가이드

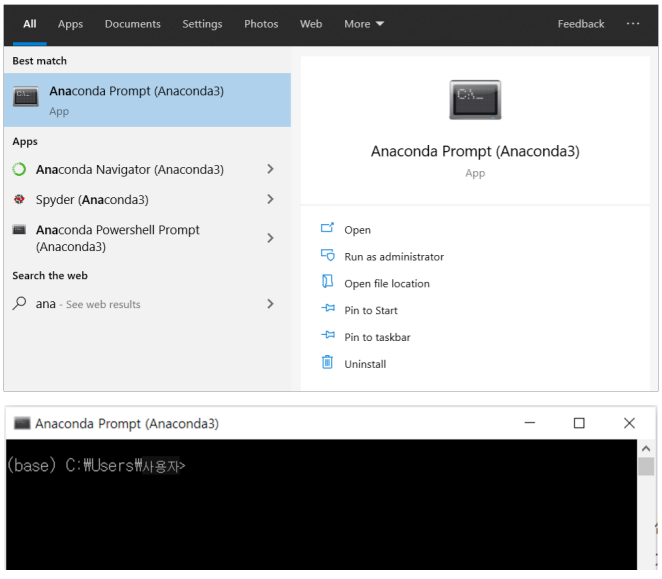
▶ 필요 SW 설치

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

▶ 가상환경 구축 가이드

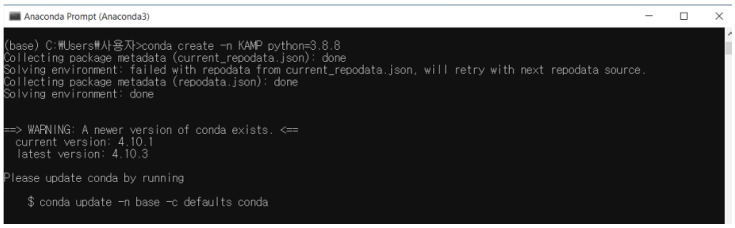
- 3. 부록 [분석 환경 구축을 위한 설치 가이드]에서 설치한 아나콘다(Anaconda)에서 파이썬의 독립적인 가상의 작업 환경을 구축한다. 파이썬의 모듈은 다양한 버전이 존재하기 때문에 충돌이 일어나기 쉬우며, 이러한 충돌을 방지하기 위해 실습마다 독립적인 가상 환경 구축을 권장한다. 가상 환경을 사용하기 위해서는 환경을 구축하고 실행(활성화)해야하고, 실습이 끝난 이후에는 환경을 비활성화 시켜야 한다. 주피터 노트북에서 가상 환경을 실행하는 구체적인 순서 및 실행 방법은 아래와 같다.

- 아나콘다 프롬프트 실행



- 가상 환경 생성

```
# conda create -n 가상환경이름 python=버전

conda create -n KAMP python=3.8.8
```



...

```
Proceed ([y]/n)? y
```

해당 문구가 뜨면 'y' 를 입력한다.

...

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

(base) C:₩Users₩사용자>
```

- 가상 환경 실행

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

```
# conda activate 가상환경이름
conda activate KAMP
```

```
(base) C:₩Users₩사용자>conda activate KAMP

(KAMP) C:₩Users₩사용자>
```

- 분석이 끝난 후에는 가상 환경을 종료한다.

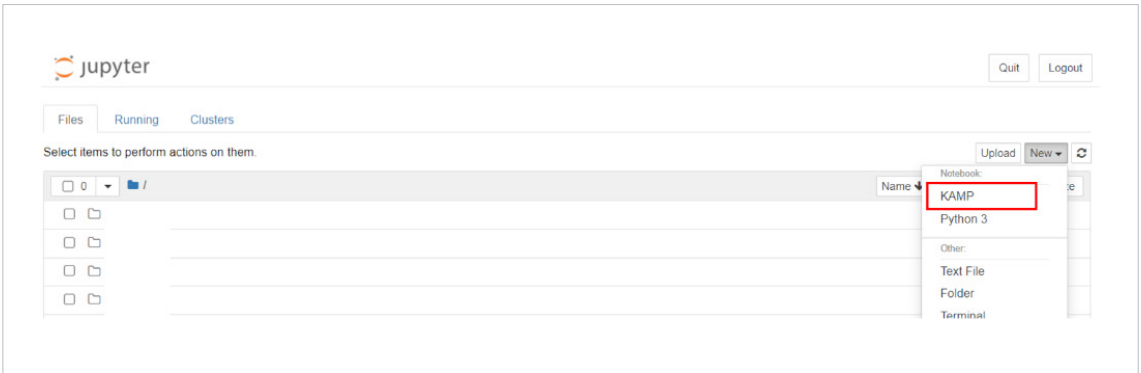
```
conda deactivate
```

```
(KAMP) C:₩Users₩사용자>conda deactivate

(base) C:₩Users₩사용자>
```

- 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.

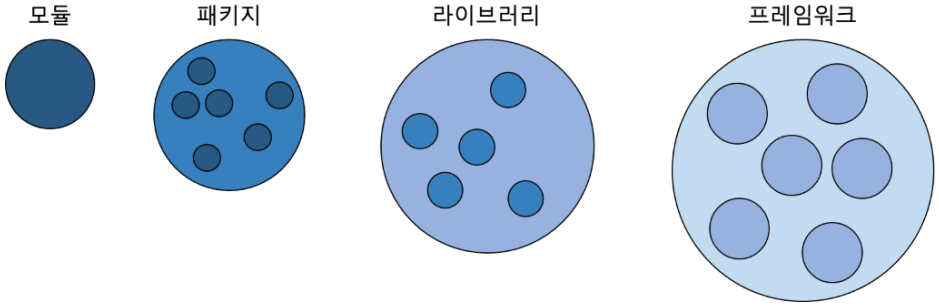
```
pip install ipykernel

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

# jupyter notebook 실행
jupyter notebook
```



▶ 패키지 설치 가이드

- 패키지 설치 전 파이썬 관련 용어 정리

모듈 (module)	• 함수나 변수 또는 클래스를 모아 놓은 파일 • 파이썬 확장자(.py) 파일에 특정 함수, 변수, 클래스 등이 저장된 코드이다.
패키지 (package)	• 여러 모듈의 집합 • 하나의 디렉토리(경로)에 놓인 모듈들의 집합을 의미하며, 일반적으로 __init__.py라는 패키지 초기화 파일이 존재한다. • 예) NumPy : 수치해석 패키지, pandas : 데이터 분석 패키지
라이브러리 (library)	• 여러 패키지의 집합 • 파이썬을 설치할 때 기본적으로 설치되는 라이브러리를 표준라이브러리라고 한다. 파이썬 공식이 아닌 외부에서 개발한 모듈과 패키지를 묶어 외부 라이브러리라고 한다. • 예) matplotlib : 데이터 시각화를 생성하기 위한 표준 라이브러리
프레임워크 (framework)	• 라이브러리의 집합 • 목적에 맞게 코드를 작성할 수 있는 기본 아키텍처이며, 필수 구성 요소를 제공한다.



[그림 15] 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 matplotlib scikit-learn tensorflow
```

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

```
Requirement already satisfied: pandas in /home/ubuntu/anaconda3/lib/pytho
n3.6/site-packages (0.23.0)
Requirement already satisfied: python-dateutil>=2.5.0 in /home/ubuntu/ana
conda3/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/li
b/python3.6/site-packages (from pandas) (1.19.5)
Requirement already satisfied: six>=1.5 in /home/ubuntu/anaconda3/lib/pyt
hon3.6/site-packages (from python-dateutil>=2.5.0->pandas) (1.15.0)
```

- 설치된 패키지들의 전체 목록을 확인할 수 있다.

```
!pip list
```

```
Package                           Version
--------------------------------- -------------------
absl-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 - Flow Chart

[그림 16] AI 분석모델 활용 흐름도

[단계 ①] 라이브러리/데이터 불러오기

①-1. 패키지 불러오기

- import 〈패키지 이름〉, 혹은 from 〈패키지 이름〉 import 〈하위 모듈 이름〉 을 사용하여 전체 패키지를 불러오거나 특정 패키지 안의 모듈만 불러오기가 가능하다. 또한 import 〈패키지 이름〉 as 〈약자〉 를 사용하여 추후 패키지 사용 시 약자(alias)로 이름을 대신할 수 있다.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

[코드 1] 필요 패키지 불러오기

①-2. 데이터 불러오기

- 핵심 데이터인 'data.xls' 를 분석 코드와 같은 폴더 내에 저장 후 사용한다. pandas 패키지 내의 'read_excel()' 함수를 활용하여 엑셀(.xls) 형식의 데이터를 불러올 수 있다. 'head()' 메서드는 불러온 데이터의 상단에 위치한 5개의 행을 보여준다.

```
data = pd.read_excel('data.xls')
data.head()
```

	Part Number	D일 06~08(08)H 투입계획(발주) 수량	D일 08~10(10)H 투입계획(발주) 수량	D일 10~12(13)H 투입계획(발주) 수량	D일 13~15(15)H 투입계획(발주) 수량	D일 15~17(18)H 투입계획(발주) 수량	D일 18~20(21)H 투입계획(발주) 수량	D일 21~23(23)H 투입계획(발주) 수량	D일 23~01(02)H 투입계획(발주) 수량	D일 02~04H 투입계획 수량	...	D+23일 투입예정 수량
0	Part 0	0	0	0	0	0	0	0	0	0	...	0
1	Part 1	0	0	0	0	0	0	0	0	0	...	0
2	Part 2	139	60	80	76	68	81	68	44	0	...	0
3	Part 3	15	11	17	13	12	16	12	8	0	...	0
4	Part 4	40	27	29	17	18	29	18	9	0	...	0

[코드 2] 데이터 불러오기 및 상단 5개 데이터 확인

①-3. 분석하고자 하는 제품 데이터 선택하기

- 본 분석에서는 전체 부품 117개 중에서 단일 제품의 제조 공정에 함께 사용되는 총 2개 부품의 데이터를 활용하여 발주 수량 예측을 실시한다. 불러온 데이터에서 분석하고자 하는 제품으로 선정한(다른 부품도 가능) 부품 94와 부품 95에 해당하는 데이터를 선택하여 data_94와 data_95에 각각 저장한다. 각 부품의 데이터는 부품 식별자(Part Number)가 'Part 94'와 'Part 95'에 해당한다. 추가로, pandas 패키지의 'reset_index()' 메서드를 사용하여 분리된 데이터 각각의 인덱스를 초기화한다.

```
data_94 = data[(data["Part Number"]=='Part 94')]
data_95 = data[(data["Part Number"]=='Part 95')]

data_94.reset_index(drop=True, inplace=True)
data_95.reset_index(drop=True, inplace=True)
```

[코드 3] 분석하고자 하는 제품 데이터를 선택 및 저장

[단계 ②] 데이터 탐색

②-1. 기초 통계 확인

- 분석에 먼저 사용할 부품 94 데이터의 기초 통계를 확인한다. pandas 패키지의 'describe()' 메서드를 사용하면 데이터의 행 개수, 평균, 표준편차 등의 통계치(statistics)를 확인할 수 있다.

```
data_94.describe()
```

	D일 06~08(08)H 투입계획(발주) 수량	D일 08~10(10)H 투입계획(발주) 수량	D일 10~12(13)H 투입계획(발주) 수량	D일 13~15(15)H 투입계획(발주) 수량	D일 15~17(18)H 투입계획(발주) 수량	D일 18~20(21)H 투입계획(발주) 수량	D일 21~23(23)H 투입계획(발주) 수량	D일 23~01(02)H 투입계획(발주) 수량	D일 02~04H 투입계획 수량	D일 04~06H 투입계획 수량	...	D+23일 투입예정 수량	D+24일 투입예정 수량	D+25일 투입예정 수
count	153.000000	153.000000	153.000000	153.000000	153.000000	153.000000	153.000000	153.000000	153.0	153.0	...	153.000000	153.000000	153.00000
mean	12.686275	3.372549	3.960784	3.254902	3.254902	3.797386	3.444444	2.411765	0.0	0.0	...	3.738562	3.477124	3.71241
std	6.156373	1.609521	1.669708	1.519736	1.519736	1.836614	1.681513	1.388531	0.0	0.0	...	10.032035	9.946624	10.16810
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0.0	...	0.000000	0.000000	0.00000
25%	9.000000	3.000000	4.000000	4.000000	4.000000	4.000000	3.000000	2.000000	0.0	0.0	...	0.000000	0.000000	0.00000
50%	14.000000	4.000000	5.000000	4.000000	4.000000	5.000000	4.000000	3.000000	0.0	0.0	...	0.000000	0.000000	0.00000
75%	18.000000	4.000000	5.000000	4.000000	4.000000	5.000000	4.000000	3.000000	0.0	0.0	...	0.000000	0.000000	0.00000
max	21.000000	8.000000	5.000000	4.000000	4.000000	5.000000	9.000000	8.000000	0.0	0.0	...	34.000000	34.000000	34.00000

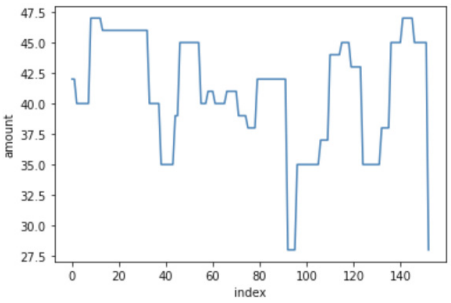
8 rows × 83 columns

[코드 4] 데이터의 통계치 확인

②-2. 데이터 시각화

- matplotlib 패키지를 사용하여 부품 94 데이터의 발주량을 시각화한다. 시각화하려는 변수를 선택하고 'plot()' 함수를 활용하여 그래프를 그릴 수 있다. 예시로는 당일 발주량인 'D일 투입예정 수량(D일계획)' 변수를 선택하였다.

```
plt.plot(data_94[['D일 투입예정 수량(D일계획)']])
plt.xlabel('index')
plt.ylabel('amount')
```



[코드 5] 데이터 시각화

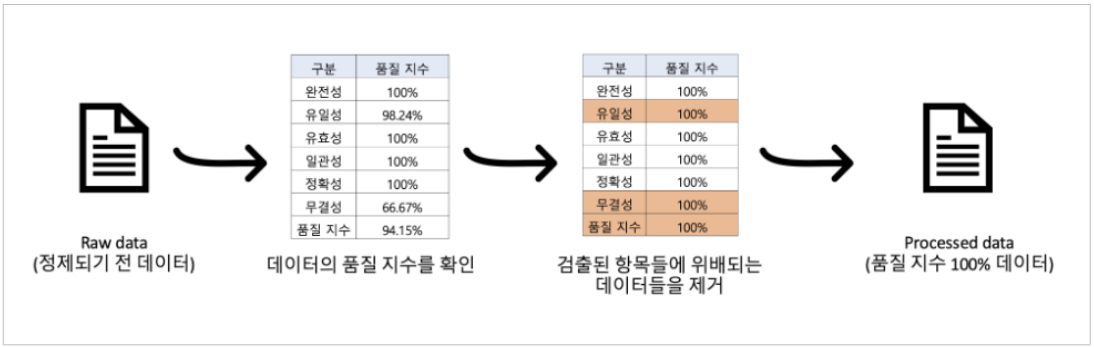
[단계 ③] 데이터 정제(전처리)

③-1. 데이터 전처리

• 데이터 전처리 방법

- 개발 언어 : Python (파이썬)

- 개발 환경 : Jupyter Notebook (주피터 노트북)



[그림 17] 데이터 전처리 과정 도식화

③-2. 데이터 정제

- 데이터 내부에 'NaN(Not-a-Number)'으로 표기된 누락 데이터가 있는지 확인한다. pandas 패키지의 'isna()' 메서드를 통해 데이터 내에 누락값 여부를 True 혹은 False로 확인할 수 있다.

```
data_94.isna()
```

	Part Number	D일 06~08(08)H 투입계획(발주) 수량	D일 08~10(10)H 투입계획(발주) 수량	D일 10~12(13)H 투입계획(발주) 수량	D일 13~15(15)H 투입계획(발주) 수량	D일 15~17(18)H 투입계획(발주) 수량	D일 18~20(21)H 투입계획(발주) 수량	D일 21~23(23)H 투입계획(발주) 수량	D일 23~01(02)H 투입계획(발주) 수량	D일 02~04H 투입계획 수량	...	D+23일 투입예정수량	D+24일 투입예정수량	D+25일 투입예정수량	D+26일 투입예정수량	D+27일 투입예정수량
0	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False
5	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False
6	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False
7	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False

[코드 6] 데이터 누락값 여부 확인

- ‘isna().sum()’ 메서드를 사용하여 변수별 누락 데이터의 개수를 확인할 수 있다.

```
data_94.isna().sum()
```

```
Part Number                0
D일06~08(08)H 투입계획(발주) 수량    0
D일08~10(10)H 투입계획(발주) 수량    0
…
…
…
D+30일 투입예정 수량              0
D+31~D+45일 투입예정 수량         0
CRET_TIME                  0
Length: 84, dtype: int64
```

[코드 7] 누락 데이터 개수 확인

③-3. 불필요한 열 제거 (필요한 열만 남기기)

- pandas 패키지의 속성(attribute)인 ‘columns’를 통해 데이터에 존재하는 변수의 종류를 확인한다.

```
data_94.columns # 혹은 data.info()
```

```
Index(['Part Number', 'D일06~08(08)H 투입계획(발주) 수량', 'D일08~10(10)H 투입계획(발주) 수량', 'D일
10~12(13)H 투입계획(발주) 수량', 'D일13~15(15)H 투입계획(발주) 수량',
…
…
…
'D+3일 투입예정 수량(과부족수량).5', 'D+3일 투입예정 수량(과부족수량).6', 'D+28일 투입예정 수량', 'D+29일 투입예
정 수량', 'D+30일 투입예정 수량', 'D+31~D+45일 투입예정 수량', 'CRET_TIME'], dtype='object')
```

[코드 8] 데이터의 변수 확인

- pandas 패키지의 ‘loc[]’ 메서드로 데이터 변수들 중 필요한 변수만 추출한다.

```
data_94 = data_94.loc[:, ['Part Number', 'D일 투입예정 수량(D일계획)', 'D+1일 투입예정 수량(Total)', 'D+2일 투입예정 수량
(Total)', 'D+3일 투입예정 수량(Total)', 'D+4일 투입예정 수량(Total)', 'D+5일 투입예정 수량', 'D+6일 투입예정 수량', 'D+7일 투입예
정 수량', 'D+8일 투입예정 수량', 'D+9일 투입예정 수량', 'D+10일 투입예정 수량', 'D+11일 투입예정 수량', 'D+12일 투입예정 수량',
'CRET_TIME']]


data_94.columns
```

```
Index(['Part Number', 'D일 투입예정 수량(D일계획)', 'D+1일 투입예정 수량(Total)',
       'D+2일 투입예정 수량(Total)', 'D+3일 투입예정 수량(Total)', 'D+4일 투입예정 수량(Total)',
       'D+5일 투입예정 수량', 'D+6일 투입예정 수량', 'D+7일 투입예정 수량', 'D+8일 투입예정 수량',
       'D+9일 투입예정 수량', 'D+10일 투입예정 수량', 'D+11일 투입예정 수량', 'D+12일 투입예정 수량',
       'CRET_TIME'],
      dtype='object')
```

[코드 9] 필요한 변수만 추출하여 데이터 저장 및 변수 확인

- 데이터의 수집 시간 정보인 ‘CRET_TIME’ 변수를 날짜 시간형으로 변환한다.

pandas 패키지의 'to_datetime()' 메서드 내에서 수집된 데이터의 형태와 동일하게 연도(Y)-월(m)-일(d)-시(H)-분(M)의 순서로 내용을 정리하기 위해 형태(format)를 지정한다.

```
data_94['CRET_TIME'] = pd.to_datetime(data_94['CRET_TIME'], format="%Y%m%d%H%M")
```

[코드 10] 날짜 시간형 변수의 형태 지정

- 'CRET_TIME' 변수에 존재하는 값을 확인한다. 'unique()' 메서드는 특정 열에 대한 유일한 값(unique value)을 찾기 때문에, 이를 사용하여 존재하는 값의 종류를 확인할 수 있다.

```
# CRET_TIME 열에서 존재하는 값의 종류 확인
data['CRET_TIME'].unique()
```

```
array(['2021-09-13T18:30:00.000000000', '2021-09-14T06:05:00.000000000',
       ···, dtype='datetime64[ns]')
```

[코드 11] 특정 변수('CRET_TIME')에 존재하는 값 확인

③-4. 불필요한 행 제거

- 하루에 평균적으로 3~4회 당일 발주량에 대한 로그가 수집된다. (예 : 2021년 9월 14일에 4차례 당일 주문 계획에 대한 기록이 누적되어 있다). 실제로는 매일 가장 마지막 시간의 발주 로그가 해당일의 전체 발주 정보를 담고 있으므로, 'groupby()' 메서드를 사용하여 일자를 기준으로 그룹화하고, 'last()' 메서드를 사용해 가장 늦게 기록된 로그만을 남긴다. 불필요한 열을 제거하였으므로 'reset_index()' 메서드로 인덱스를 초기화한다.

```
data_94 = data_94.groupby(by=[data_94['CRET_TIME'].dt.year,
                              data_94['CRET_TIME'].dt.month,
                              data_94['CRET_TIME'].dt.day]).last()
data_94.reset_index(drop=True, inplace=True)
```

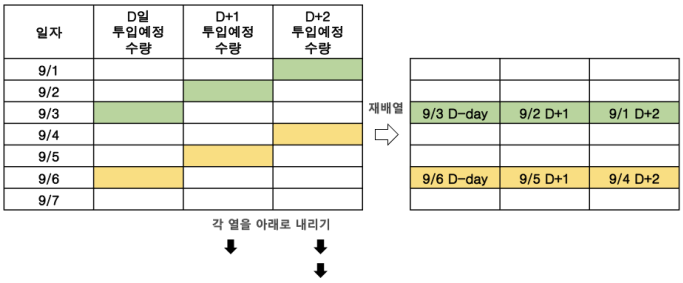
[코드 12] 불필요한 행 제거

[단계 ④] 주요 변수 선택

④-1. 변수별 상관관계 파악을 위한 데이터 재배열

- 전체 변수 중 유의미한 특성을 가진 주요 변수를 선정하기 위해서는 변수간 상관관계를 파악해야 한다. 당일 주문량에 해당되는 'D일 투입 예정 수량' 열(변수)을 기준으로 다른 열과의 상관관계를 계산하면 당일 주문량과 연관성이 있는 열을 파악할 수 있다. 이러한 변수간 상관관계 파악을 위해서는 주어진 데이터를 재배열할 필요가 있다. 예를 들어, 당일 발주 수량과 1, 2일 전의 계획 발주 수량 간의 상관관계를 확인하는 것이 목적일 경우 동일한 일자에 대한 계획으로 데이터를 맞춰주는 재배열을 시행해야 한다. [그림 18]처럼 9월 3일에 대한 발주 수량은 1) 9월 3일의 당일 발

주 수량, 2) 9월 2일의 하루 뒤 계획 발주 수량(D+1 투입 예정 수량), 3) 9월 1일의 이틀 뒤 계획 발주 수량(D+2 투입 예정 수량) 등과 연관된다고 볼 수 있다.



[그림 18] 변수별 상관관계 파악을 위한 재배열

- 본 실습에서는 당일 투입 예정 수량과 12일 전까지의 투입 예정 수량(D+1~D+12 투입 예정 수량)에 대한 상관관계를 확인하고자 하였다. pandas 패키지의 'iloc[]' 메서드를 이용하여 변수별로 추출할 행과 열을 다르게 설정함으로써 [그림 18]과 같이 재배열을 실시하였다. 'reset_index()' 를 사용하여 원하는 열만 인덱스 없이 추출하였다.

```
corr_data_94 = pd.concat([
        data_94.iloc[12:, :]['D일 투입예정 수량(D일계획)'].reset_index(drop=True),
        data_94.iloc[11:-1, :]['D+1일 투입예정 수량(Total)'].reset_index(drop=True),
        data_94.iloc[10:-2, :]['D+2일 투입예정 수량(Total)'].reset_index(drop=True),
        data_94.iloc[9:-3, :]['D+3일 투입예정 수량(Total)'].reset_index(drop=True),
        data_94.iloc[8:-4, :]['D+4일 투입예정 수량(Total)'].reset_index(drop=True),
        data_94.iloc[7:-5, :]['D+5일 투입예정 수량'].reset_index(drop=True),
        data_94.iloc[6:-6, :]['D+6일 투입예정 수량'].reset_index(drop=True),
        data_94.iloc[5:-7, :]['D+7일 투입예정 수량'].reset_index(drop=True),
        data_94.iloc[4:-8, :]['D+8일 투입예정 수량'].reset_index(drop=True),
        data_94.iloc[3:-9, :]['D+9일 투입예정 수량'].reset_index(drop=True),
        data_94.iloc[2:-10, :]['D+10일 투입예정 수량'].reset_index(drop=True),
        data_94.iloc[1:-11, :]['D+11일 투입예정 수량'].reset_index(drop=True),
        data_94.iloc[:-12, :]['D+12일 투입예정 수량'].reset_index(drop=True)
        ],
        axis=1)

corr_data_94.head()
```

	D일 투입예정 수량(D일계획)	D+1일 투입예정 수량(Total)	D+2일 투입예정 수량(Total)	D+3일 투입예정 수량(Total)	D+4일 투입예정 수량(Total)	D+5일 투입예정 수량	D+6일 투입예정 수량	D+7일 투입예정 수량	D+8일 투입예정 수량	D+9일 투입예정 수량	D+10일 투입예정 수량	D+11일 투입예정 수량	D+12일 투입예정 수량
0	46	30	30	0	0	0	0	30	30	30	30	0	0
1	46	30	30	0	0	0	0	30	30	0	0	0	0
2	40	30	30	0	0	0	0	30	30	0	30	30	30
3	35	30	30	0	0	0	0	30	30	0	30	30	30
4	39	30	30	0	0	0	0	30	30	0	30	30	30

[코드 13] 데이터 재배열

④-2. 변수별 상관관계 파악 및 현장의 도메인 지식(domain knowledge)을 기반으로 한 유의미한 변수 선택

- 선정한 13개의 특성 간의 표준 상관계수를 'corr()' 메서드를 이용해 자동으로 계산한다.

```
corr_data_94.corr()
```

	D일 투입예정 수량(D일계획)	D+1일 투입예정 수량 (Total)	D+2일 투입예정 수량 (Total)	D+3일 투입예정 수량 (Total)	D+4일 투입예정 수량 (Total)	D+5일 투입예정 수량	D+6일 투입예정 수량	D+7일 투입예정 수량	D+8일 투입예정 수량	D+9일 투입예정 수량	D+10일 투입예정 수량	D+11일 투입예정 수량	D+12일 투입예정 수량
D일 투입예정 수량(D일계획)	1.000000	0.651721	0.408578	0.393761	0.136667	0.388251	0.018768	0.127412	0.246102	0.024242	0.215673	0.103277	0.081598
D+1일 투입예정 수량 (Total)	0.651721	1.000000	0.567057	0.007026	-0.052853	-0.074226	-0.028867	0.114627	0.345470	0.079831	0.286655	0.184163	0.178313
D+2일 투입예정 수량 (Total)	0.408578	0.567057	1.000000	0.042953	-0.181014	-0.270177	0.043475	0.270970	0.398243	0.102379	0.376835	0.275252	0.280119
D+3일 투입예정 수량 (Total)	0.393761	0.007026	0.042953	1.000000	0.210142	0.562022	-0.003860	-0.002500	0.073848	0.098664	0.107248	-0.012449	-0.036512

[코드 14] 데이터 변수 간 상관관계 확인

④-3. 주요 변수 선택 및 분석을 위한 데이터 선정

- 당일 발주 수량과 가장 유의미한 관계가 있는 변수로 D+3, D+4, D+5일의 투입 예정 수량을 최종 선택하였고, 이를 기반으로 3~5일 전 발주 및 계획 발주 수량을 활용하여 재고관리 최적화를 위한 발주량 예측 모델을 설계한다. 즉, 해당 변수를 활용하여 학습된 예측 모델은 부품의 발주 수량을 3일 전에 예측할 수 있게 된다.

```
data_94 = data_94.loc[:, ['D일 투입예정 수량(D일계획)',
            'D+3일 투입예정 수량(Total)',
            'D+4일 투입예정 수량(Total)',
            'D+5일 투입예정 수량'
            ]].reset_index(drop=True)
data_94.head()
```

	D일 투입예정 수량(D일계획)	D+3일 투입예정 수량 (Total)	D+4일 투입예정 수량 (Total)	D+5일 투입예정 수량
0	42	30	30	38
1	40	30	50	45
2	40	47	43	51
3	47	53	42	39
4	46	44	37	51

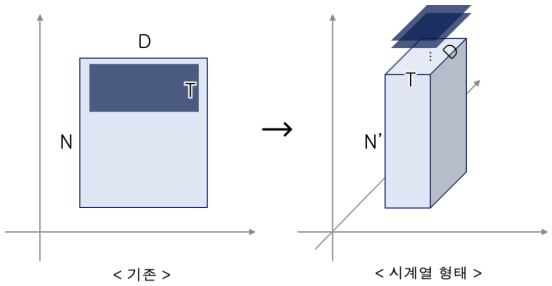
[코드 15] 주요 변수 선택 및 저장

[단계 ⑤] 모델의 입력 데이터 형태 설정

⑤-1. 시계열 형태로 데이터 변환

- 수집된 발주 및 계획 발주 수량은 본질적으로 시계열 데이터이지만, AI 분석모델을 활용 시에 모델의 입력으로 사용되기 위해서는 추가적인 처리가 필요하다. 본 분석에서는 과거 사흘 치에 해당하는 수량 데이터((D+3~5일 투입 예정 수량)를 활용하여 3일 후의 발주 수량을 예측하므로, 전체 시계열 데이터를 여러 인스턴스로 분리하는 슬라이싱(slicing) 작업을 진행한다. 이후 사용할 장단기 메모리(LSTM) 모델은

3차원 시계열 데이터를 입력값으로 취하므로, [그림 19]처럼 데이터를 2D 텐서에서 3D 텐서 형태로 변환한다.



[그림 19] 시계열 형태 데이터 변환

- [코드 16]에서는 사용자 정의 함수인 'to_timeseries_data'를 만들어 기존 데이터 입력 시 시계열 형태의 데이터로 변환한다. [그림 20]처럼, 예측 대상(target, y)과 예측에 사용될 데이터(input, X)가 모델 학습을 위해 짝지어지고, 해당 과정이 전체 데이터셋에 대해 순차적으로 진행되어 총 N' 번 실행된다.

[그림 20] 사용자 정의 함수 'to_timeseries_data'의 데이터 추출 및 작동 방식

```
# 사용자 정의 함수 만들기
def to_timeseries_data(data, lookback=3, delay=3):
    # data는 원본 tabular 데이터
    # lookback: 입력으로 사용하기 위해 거슬러 올라갈 시간단위의 개수=3일전
    # delay: target으로 사용할 미래의 시점=3일후

    output_len = len(data)-(lookback+delay)+1 # N=total_length-(3+3)+1
    n_feature = data.shape[-1] # =4

    inputs = np.zeros((output_len, lookback, n_feature)) # (N,3,4)
    targets = np.zeros((output_len,)) # (N,)

    for i in range(output_len):
        inputs[i] = data.iloc[i:i+lookback, :]
        targets[i] = data.iloc[i+lookback+delay-1, 0]

    return inputs, targets
# 사용자 정의 함수 적용
X_94, y_94 = to_timeseries_data(data_94)

print("X의 형태: ", X_94.shape)
print("y의 형태: ", y_94.shape)
```

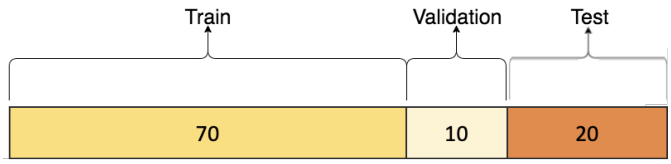
```
X의 형태:  (44, 3, 4)
y의 형태:  (44,)
```

[코드 16] 분석에 맞게 사용자 정의 함수 만들기 및 적용

[단계 ⑥] 학습, 검증, 평가 데이터셋 분리 및 스케일링(scaling)

⑥-1. 학습, 검증, 평가용 데이터셋 분리

- 좋은 모델은 학습 시에 접한 적 없는 새로운 데이터에 대해 일반화 가능한 성능을 보이는 모델이라고 할 수 있다. 따라서 평가 데이터를 사용하여 AI 분석모델의 일반화 성능을 확인하는 것이 보편적이다. 데이터셋을 용도에 맞게 총 3가지의 학습(train), 검증(validation), 평가(test) 데이터로 나눌 필요가 있다. 학습 데이터셋은 AI 분석 모델 학습 시에 사용하고, 검증 데이터셋은 학습 데이터셋으로 훈련된 모델의 성능 측정 시에 사용된다. 모델의 성능이 검증 데이터셋을 통해 일차적으로 검증이 되면, 마지막으로 평가 데이터셋을 사용하여 모델의 성능을 최종적으로 판단한다.
- 모델의 성능 판단을 위해 데이터셋을 검증용과 평가용으로 분리하는 이유는 모델이 학습 데이터에 과적합(overfitting)되지 않았는지 확인하기 위한 목적이다. 과적합된 모델은 학습 데이터에 대해서는 잘 작동하지만, 실제 산업 현장에서 사용될 본 적 없는 데이터(unseen data)에 대해서는 낮은 성능을 보인다. 학습 중간에 검증 데이터를 사용하여 모델의 성능을 검증하고, 이를 바탕으로 모델의 초매개변수를 조정함으로써 학습된 모델이 평가 데이터셋에도 잘 작동하는 우수한 일반화 성능(generalization power)을 확보할 수 있다.
- 본 분석에서는 전체 데이터셋의 70%, 10%, 20%를 각각 학습, 검증, 평가 데이터로 활용한다. 특히 전처리 이후의 시계열 데이터가 시간 순으로 존재하므로, 학습-검증-평가 데이터를 시간 순으로 분할한다. NumPy 패키지에서 제공하는 'split()' 함수로 데이터가 분리되는 구간을 지정하여 데이터를 분할한다. (70%와 80% 위치에서 데이터 분할 실시)



[그림 21] 학습, 검증, 평가용 데이터셋 분리

```
# 데이터셋 분리, train:validation:test = 7:1:2
X_train_94, X_val_94, X_test_94 = np.split(X_94, [int(0.7*len(X_94)), int(0.8*len(X_94))])
y_train_94, y_val_94, y_test_94 = np.split(y_94, [int(0.7*len(y_94)), int(0.8*len(y_94))])


# 분리 이후 데이터 형태
print("X 학습: {}, X 검증: {}, X 평가: {}".format(X_train_94.shape,X_val_94.shape,X_test_94.shape))
print("y 학습: {}, y 검증: {}, y 평가: {}".format(y_train_94.shape,y_val_94.shape,y_test_94.shape))
```

```
X 학습: (30, 3, 4), X 검증: (5, 3, 4), X 평가: (9, 3, 4)
y 학습: (30,), y 검증: (5,), y 평가: (9,)
```

[코드 17] 데이터셋 분리 및 형태 확인

⑥-2. 데이터 스케일링 (Data Scaling)

- AI 분석모델의 입력 데이터로 활용하기 위해서는 데이터 변수의 크기를 적절히 조정할 필요가 있다. 값의 범위가 다른 변수들이 있는 경우, 모델 학습 시 변수별로 공평하게 기여하지 못하거나 편향이 발생할 수 있다. 이를 방지하기 위해 전체 데이터를 표준화(standardization)함으로써 데이터가 평균이 0, 분산이 1인 정규 분포를 따르도록 변환한다. 각 데이터에서 평균을 빼고 표준편차로 나눔으로써 아래 [그림 22]의 식과 같이 값을 변환할 수 있다. 데이터 정규화는 각 변수별로 실시한다.

$$Z = \frac{X - \mu}{\sigma}$$

[그림 22] 표준화 공식

- scikit-learn (sklearn) 에서 제공하는 표준화 스케일러(StandardScaler)를 사용하여 학습 데이터를 기준으로 평균과 표준편차를 계산한 후 학습, 검증, 평가 데이터셋에 정규화를 실시한다. 학습 데이터의 경우 'fit_transform()' 메서드를 활용하여 평균 및 표준편차 계산하기(fit)와 정규화 적용(transform)을 동시에 실행하고, 검증 및 평가 데이터에는 'transform()' 메서드를 통해 정규화만 실행한다.

```
# 전처리시 필요한 패키지 불러오기
from sklearn.preprocessing import StandardScaler

Xscaler_94 = StandardScaler()
X_train_94 = Xscaler_94.fit_transform(X_train_94.reshape(-1, X_train_94.shape[-1])).reshape(X_train_94.shape)
X_val_94 = Xscaler_94.transform(X_val_94.reshape(-1, X_val_94.shape[-1])).reshape(X_val_94.shape)
X_test_94 = Xscaler_94.transform(X_test_94.reshape(-1, X_test_94.shape[-1])).reshape(X_test_94.shape)

yscaler_94 = StandardScaler()
y_train_94 = yscaler_94.fit_transform(y_train_94.reshape(-1,1))
y_val_94 = yscaler_94.transform(y_val_94.reshape(-1,1))
y_test_94 = yscaler_94.transform(y_test_94.reshape(-1,1))
```

[코드 18] 데이터 스케일링

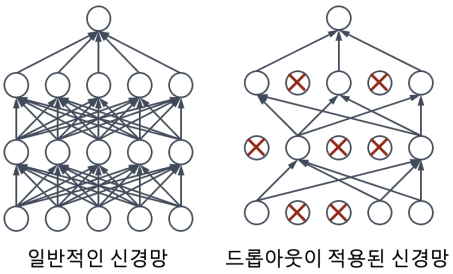
[단계 ⑦] 모델 구축 및 학습

⑦-1. 분석모델 구축

- TensorFlow 라이브러리 내 Keras 패키지의 Sequential 모델을 사용하여 순서대로 연결된 층을 일렬로 쌓아서 구성한다. 훈련에 필요한 계산과 모델을 표현하는 구조를 구성하는 과정에서 사용자가 직접 초매개변수를 설정한다. 사용할 분석모델의 경우 입력층, 은닉층 및 출력층의 크기와 학습 속도 및 드롭아웃(Dropout) 비율 등을 직접 설정할 수 있다. 첫 번째 신경망 층으로 LSTM 은닉층을 사용하고 내부 매개변수로는 (1) 드롭아웃 비율을 0.2로, (2) 활성화 함수(activation function)를 ReLU

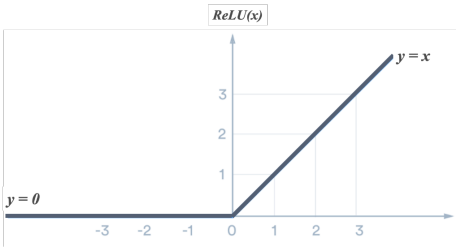
로, (3) input_shape=[time_step, features]은 3D 텐서로 변환시킨 데이터 형태에 맞게 (3,4) 크기로 설정한다.

(1) 드롭아웃이란 [그림 23]에서 볼 수 있듯, 신경망 연산에 사용되는 몇몇 뉴런들을 확률적으로 누락(drop out)시킨다는 개념에서 비롯되었다. 은닉층의 드롭아웃을 확률 p로 적용하는 경우 확률 p만큼의 은닉 유닛을 사용하지 않게 되고, 역전파를 수행할 때 제거된 유닛들에 대한 기울기 정보가 이용되지 않는다. 또한, 모델의 학습 과정에서 매 연산마다 은닉 유닛 중 어느 하나에 전적으로 의존하지 않게 되므로 과적합 문제를 해결할 수 있다. 학습이 완료된 모델의 평가 시에는 드롭아웃을 사용하지 않는 것이 일반적이다.



[그림 23] 드롭아웃 설명

(2) 활성화 함수란 입력된 데이터의 가중 합을 출력 신호로 변환하는 함수이다. 활성화 함수 Rectified Linear Unit (ReLU)는 [그림 24]와 같이 0보다 작은 입력은 0으로 만들고, 0 이상의 입력은 그대로 출력함으로써 빠른 학습을 가능하게 한다. ReLU는 시그모이드나 하이퍼볼릭탄젠트(tanh)와 같은 다른 활성화 함수보다 간단한 연산을 사용하므로 빠른 학습이 가능하며, 기울기 소실 등의 문제를 해결하는 장점이 있어 최근 가장 많이 사용되는 활성화 함수이다.



[그림 24] ReLU 활성화 함수

- 마지막으로 뉴런을 1개를 가진 완전연결층인 Dense 신경망 층을 추가하여 출력값이 1개(3일 후의 발주량)가 될 수 있도록 설정한다.

```
# 필요한 패키지 불러오기
import tensorflow as tf
tf.random.set_seed(42)
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM
```

[코드 19] 분석에 필요한 필요 패키지 불러오기

```
model = Sequential()
model.add(LSTM(8, dropout=0.2, activation='relu', input_shape=(3,4), return_sequences=True))
model.add(LSTM(8, dropout=0.2, activation='relu'))
model.add(Dense(1, activation='linear'))

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 3, 8)              416
_________________________________________________________________
lstm_1 (LSTM)                (None, 8)                 544
_________________________________________________________________
dense (Dense)                (None, 1)                 9
=================================================================
Total params: 969
Trainable params: 969
Non-trainable params: 0
_________________________________________________________________
```

[코드 20] 분석모델 구축

⑦-2. 모델 컴파일 (compile)

- 모델을 훈련시키기 전에 훈련 과정에 필요한 매개변수를 정의해야 한다. 훈련 매개변수는 'compile()' 메서드에 전달하며, 메서드를 실행할 때 필요한 매개변수는 다음과 같다.

(1) 손실 함수(loss) : 실제 값과 예측값의 차이를 계산하는 함수이며, 값이 작을수록 오차가 적음을 의미한다. 본 실습에서는 연속형 변수인 발주 수량을 예측하는 문제이므로 이에 적합한 목적식인 평균 절대 오차(mean absolute error, MAE)를 사용한다.

$$\mathbf{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}$$

[그림 25] 평균 절대 오차(MAE)

(2) 옵티마이저(optimizer) : 앞서 정의한 손실값을 최소화하는 신경망의 모수를 찾을 수 있도록 하게 하는 최적화 알고리즘이다. 그 예시로 경사 하강법이 있다. 본

실습에서는 다양한 인공신경망 모델 구조에서 좋은 성능을 보이는 'Adaptive Moment Estimation (ADAM)' 을 옵티마이저로 사용한다.

```
model.compile(optimizer='adam', loss='mae')
```

[코드 21] 옵티마이저, 손실함수 정의

⑦-3. 모델 학습 및 조건

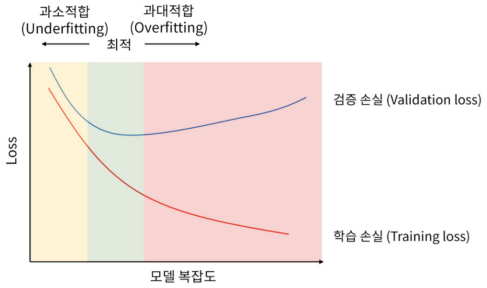
- 딥러닝 혹은 인공신경망 기반 모델 학습 시에는 CPU 혹은 GPU를 사용할 수 있다. 일반적으로 사진, 동영상과 같이 용량이 큰 데이터셋 사용 시 병렬처리를 통한 학습 속도 향상을 위해 GPU를 사용한다. 하지만 이런 경우에는 GPU를 추가로 구매하고 설치해야 하므로, 작은 데이터셋을 사용하여 분석할 경우에는 CPU만을 사용하여 AI 분석모델을 학습하기도 한다. 본 분석에서 사용되는 학습 데이터셋은 용량이 크지 않으므로, 본 분석에서는 학습 과정의 편의를 위해 CPU를 사용한다.
- 최종적으로 모델 훈련을 위해 'fit()' 메서드를 사용하며, 필요한 매개변수는 다음과 같다.

  (1) 입력 및 출력 데이터 : 훈련 데이터의 X와 y가 각각 입력과 출력 데이터이며, 모델을 학습시키는데 활용된다.

  (2) 에폭(epochs) : 모델이 전체 데이터를 순회하며 학습한 횟수를 의미하며, 에폭이 클수록 학습 정도는 증가하지만 횟수가 너무 클 경우 과적합 현상이 발생하므로 적절한 훈련 횟수를 설정하는 것이 중요하다.

  (3) 배치 사이즈(batch_size) : 한 번의 신경망 모수 업데이트시 사용하는 데이터의 개수를 의미한다. 배치의 크기는 인공신경망 학습 및 성능에 영향을 미친다. 전체 훈련 데이터를 한 번에 학습시키면 학습 시간이 오래 걸리거나 일반화 성능이 떨어지는 것이 관측되기도 하며, 너무 적은 양으로 쪼개어 학습시키면 각 실행 결과의 변동이 생겨 전체 결과값이 불안정해질 수 있으므로 적절한 배치의 크기를 찾아야 한다.

  (4) 검증용 데이터(validation_data) : 앞서 전체 데이터에서 분리해둔 검증 데이터를 사용한다. 검증 데이터를 이용하면 모델의 과적합 여부와 같은 학습 정도를 파악할 수 있다. 모델의 과적합 여부를 판단하는 방법은 [그림 26]처럼 학습 데이터에 대한 손실은 지속적으로 떨어지지만 검증 데이터에 대한 손실은 상승하는 것을 확인하는 것이다. 이는 학습 데이터에 지나치게 학습되어 오차가 감소하지만, 실제 평가 데이터와 유사할 것으로 가정한 검증 데이터에 대해서는 예측 오차가 증가하는 것을 의미한다. 과적합을 방지하기 위해서는 학습 조기 종료(early stopping) 방법을 이용하여 과적합이 일어나기 전의 최적의 구간에서 모델의 학습을 중지해야 한다.



[그림 26] 과적합 판단 기준

(5) 콜백(callback) : 콜백은 모델의 'fit()' 메서드가 호출될 때 전달되는 객체이다. 훈련 동안 모델은 여러 지점에서 콜백을 호출하여 모델의 상태와 성능에 대한 정보에 접근하여, 훈련을 중지하거나 모델을 저장하고 가중치를 적재하거나 모델 상태를 변경할 수 있다.

- 본 실습에서는 Keras 패키지 내 Callback (콜백) 함수의 학습 조기 종료(early stopping) 기능을 사용하여 신경망의 과적합을 방지하였다. 학습 중 관찰하는(monitor) 수치에 대하여 모델에 악영향을 끼치고 있다고 판단될 시, 모델의 학습을 전체 에폭만큼 훈련되기 전에 조기 종료하도록 학습을 계획하였다. patience는 예측 결과의 개선이 없음에도 즉시 학습을 종료하지 않고 개선이 없는 에폭을 얼마나 기다릴지를 설정하는 값이다. 본 실습에서는 검증용 데이터셋의 손실 값(val_loss)에 대하여 15회 이상 개선이 관찰되지 않으면 학습을 조기 종료하도록 설정하였다.
- 또한, 모델을 저장할 때 사용하는 콜백 함수인 모델 체크포인트(ModelCheckpoint)로 최적의 모델을 저장하였다. 관찰하는 대상인 검증용 데이터셋의 손실값(monitor=val_loss)이 가장 우수할 때(save_best_only=True), 즉, 가장 낮을 때의 에폭에서 모델을 저장함으로써 개선된 모델을 선택하여 추후에 사용할 수 있도록 하였다. 인공신경망 기반 분석모델은 매번 같은 학습 결과를 도출하지는 않기 때문에, 아래의 [코드 22]의 학습 결과와 다른 결과가 도출될 수 있다.

```
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint

model_path = './model/{epoch:02d}-{val_loss:.4f}.hdf5'
callbacks = [EarlyStopping(monitor='val_loss', patience=15),
             ModelCheckpoint(filepath=model_path, monitor='val_loss', verbose=0, save_best_only=True)]

history = model.fit(X_train_94, y_train_94, epochs=100, batch_size=4, validation_data=(X_val_94, y_val_94),
callbacks=callbacks)
```

```
Epoch 1/100
8/8 [==============================] - 3s 68ms/step - loss: 0.7913 - val_loss: 0.8230
Epoch 2/100
8/8 [==============================] - 0s 20ms/step - loss: 0.7912 - val_loss: 0.8240
...
...
...
Epoch 15/100
8/8 [==============================] - 0s 8ms/step - loss: 0.7054 - val_loss: 0.8411
Epoch 16/100
8/8 [==============================] - 0s 8ms/step - loss: 0.6882 - val_loss: 0.8427
```

[코드 22] 모델 학습 및 저장

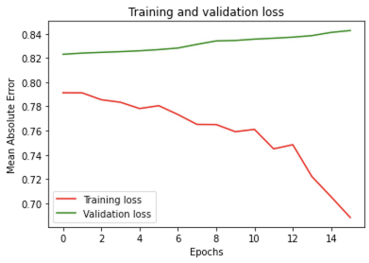
- 혹여나 모델이 저장되는 파일 경로인 model 폴더가 자동으로 생성되지 않는 경우 [그림 27]처럼 직접 실습 코드와 동일한 폴더 내에 model 폴더를 직접 만든 후 [코드 22]를 사용하여 모델 학습 및 저장을 실시한다.

[그림 27] 실습 코드와 동일한 경로 내 model 폴더 생성

⑦-4. 학습 결과 확인

- matplotlib 패키지를 활용하여 에폭별 학습 진행 상황을 확인할 수 있다. 'plot()' 함수로 학습 손실값과 검증 손실값을 그래프로 나타내고, 'title()' 과 'legend()' 함수로 제목과 범례를 만들고 난 후, 'show()' 함수로 해당 그래프를 시각화한다.

```
loss = history.history['loss']
val_loss = history.history['val_loss']
epoch = history.epoch

plt.figure()
plt.plot(epoch, loss, 'r', label='Training loss')
plt.plot(epoch, val_loss, 'g', label='Validation loss')

plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Mean Absolute Error')
plt.legend()
plt.show()
```



[코드 23] 학습 결과 시각화

[단계 ⑧] 모델 성능 확인

⑧-1. 평가용 데이터셋에 대한 예측 성능 확인

- '.hdf5' 확장자로 저장된 학습 모델 중 가장 낮은 val_loss 값을 가진 좋은 모델을 불러온다. 콜백 함수의 모델 체크포인트를 이용하여 val_loss가 최저일 때마다 모델을 저장하였으므로, 모델 저장 경로 내의 파일들 중 가장 마지막에 저장된 모델을 선택한다.

```
from tensorflow.keras.models import load_model
from os import listdir
path = listdir('./model/')[-1]
best_model = load_model('./model/'+path)
```

[코드 24] 학습 모델 불러오기

- Keras 패키지의 Sequential 인스턴스의 'evaluate()' 메서드를 사용하여 평가 데이터셋에 대한 성능을 확인한다.

```
best_model.evaluate(X_test_94, y_test_94) # MAE
```

```
1/1 [==============================] - 0s 476ms/step - loss: 1.2037

1.203702449798584
```

[코드 25] 평가 데이터셋에 대한 성능 확인

⑧-2. 예측값 확인

- 마찬가지로 Sequential 인스턴스의 'predict()' 메서드를 사용하여 모델의 예측 결과를 확인할 수 있다.

```
y_pred_94 = best_model.predict(X_test_94)
y_pred_94
```

```
array([[ 0.02630846],
       [ 0.00877363],
       ...
       ...
       ...
       [ 0.02330384],
       [-0.0150137 ]], dtype=float32)
```

[코드 26] 예측 및 예측 결과 확인

- 최종 출력 형태를 확인하기 위해 [단계 ⑥-2]에서 표준화한 값을 'inverse_transform()' 메서드를 통해 기존의 값 범위로 역변환(복원)한다. 학습 데이터 중 y에 대한 평균과 표준편차 값이 저장되어 있는 변수 'yscaler_94'에 대해 역변환을 실시한다.

```
y_pred_94_inv = yscaler_94.inverse_transform(y_pred_94)
y_test_94_inv = yscaler_94.inverse_transform(y_test_94)
```

[코드 27] 예측값을 기존의 값 범위로 복원

⑧-3. 최종 성능 확인

- 최종 성능 출력을 위해 손실 함수로 쓰였던 평균 절대 오차(MAE)를 모델의 성능 지표로 사용한다. [단계 ⑦-2]의 수식을 자동으로 계산해주는 scikit-learn의 'mean_absolute_error()' 함수로 성능을 계산한다.

```
from sklearn.metrics import mean_absolute_error
mean_absolute_error(y_test_94_inv, y_pred_94_inv)
```

```
5.449846903483073
```

[코드 28] 최종 성능 확인

[단계 ⑨] 추가 정보를 포함한 모델 구축, 학습 및 평가

- 이전 단계의 분석에서는 하나의 부품 데이터(부품 94)의 발주 데이터를 활용하여 3일 후 미래의 발주량을 예측하였다. 본 단계에서는 분석모델의 성능을 제고하기 위해 동일한 공정에 사용되는 다른 부품(부품 95)의 과거 발주량 데이터를 추가적으로 활용하여 동일한 부품(부품 94)의 발주량을 예측해 본다. 더 많은 정보를 가진 데이터를 활용할 때 AI 분석모델의 성능이 향상될 수 있음을 확인한다.

⑨-1. 추가 부품의 발주 데이터를 활용한 발주 수량 예측 모델 구축

- [단계 ①-3]에서 이미 부품 95의 데이터를 불러왔었다. 이후 부품 94의 데이터를 활용한 분석과 동일한 단계([단계 ⑥]까지)를 부품 95의 데이터에 대해 진행한다. AI 모델 학습에 필요한 데이터로 변환하는 작업들에 대해서만 진행하며, 예측하고자 하는 것은 동일한 부품 94의 미래 발주량이므로 부품 95의 y 데이터는 사용하지 않는다.

```
data_95 = data_95.loc[:,['Part Number','D일 투입예정 수량(D일계획)','D+1일 투입예정 수량(Total)',
            'D+2일 투입예정 수량(Total)','D+3일 투입예정 수량(Total)','D+4일 투입예정 수량(Total)',
            'D+5일 투입예정 수량', 'D+6일 투입예정 수량','D+7일 투입예정 수량','D+8일 투입예정 수량',
            'D+9일 투입예정 수량','D+10일 투입예정 수량','D+11일 투입예정 수량','D+12일 투입예정 수량',
            'CRET_TIME']]

data_95['CRET_TIME'] = pd.to_datetime(data_95['CRET_TIME'], format="%Y%m%d%H%M")
data_95 = data_95.groupby(by=[data_95['CRET_TIME'].dt.year,
            data_95['CRET_TIME'].dt.month,
            data_95['CRET_TIME'].dt.day]).last()
data_95.reset_index(drop=True, inplace=True)

data_95 = data_95.loc[:, ['D일 투입예정 수량(D일계획)',
            'D+3일 투입예정 수량(Total)',
            'D+4일 투입예정 수량(Total)',
            'D+5일 투입예정 수량'
            ]].reset_index(drop=True)

X_95, y_95 = to_timeseries_data(data_95)
X_train_95, X_val_95, X_test_95 = np.split(X_95, [int(0.7*len(X_95)), int(0.8*len(X_95))])
Xscaler_95 = StandardScaler()
X_train_95 = Xscaler_95.fit_transform(X_train_95.reshape(-1, X_train_95.shape[-1])).reshape(X_train_95.shape)
X_val_95 = Xscaler_95.transform(X_val_95.reshape(-1, X_val_95.shape[-1])).reshape(X_val_95.shape)
X_test_95 = Xscaler_95.transform(X_test_95.reshape(-1, X_test_95.shape[-1])).reshape(X_test_95.shape)
```

[코드 29] [단계 ⑥]까지 추가 부품 데이터 처리

- 새롭게 변환한 부품 95의 데이터가 이전에 사용한 부품 94의 학습, 검증, 평가 데이터와 형태가 동일한지 확인한다.

```
# 부품 95
print("X 학습: {}, X 검증: {}, X 평가: {}".format(X_train_95.shape, X_val_95.shape, X_test_95.shape))
```

```
X 학습: (30, 3, 4), X 검증: (5, 3, 4), X 평가: (9, 3, 4)
```

[코드 30] 추가 부품 형태 확인

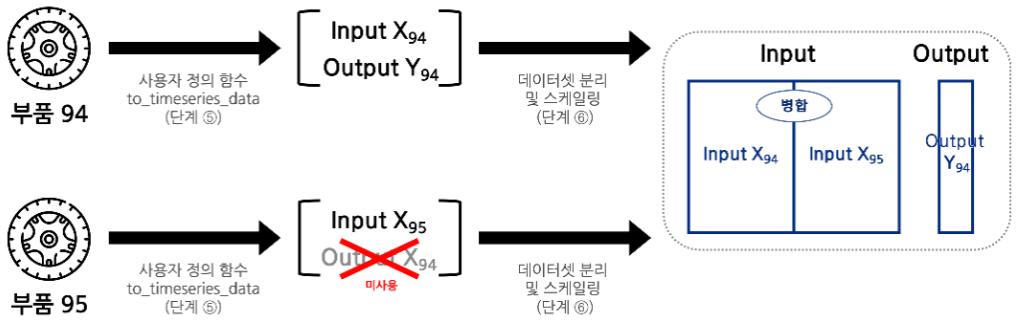
- 예측값인 y 데이터는 부품 94의 것으로 동일하므로, X 데이터의 마지막 변수 차원을 기준으로 부품 94와 부품 95의 데이터를 병합한다. NumPy 패키지의 'concatenate()' 함수를 사용하며, 병합하려는 축(차원)을 기준으로 매개변수인 axis를 2로 설정한다(0, 1, 2의 순서로 병합하고자 하는 변수의 차원을 설정할 수 있다). 이로써 본래 (데이터 개수, 3, 4) 이던 각 부품 데이터의 형태를 (데이터 개수, 3, 8) 로 변형시키게 된다.

```
X_train_both = np.concatenate([X_train_94, X_train_95], axis=2)
X_val_both = np.concatenate([X_val_94, X_val_95], axis=2)
X_test_both = np.concatenate([X_test_94, X_test_95], axis=2)
# 부품 94 + 부품 95
print("X 학습: {}, X 검증: {}, X 평가: {}".format(X_train_both.shape, X_val_both.shape, X_test_both.shape))
```

```
X 학습: (30, 3, 8), X 검증: (5, 3, 8), X 평가: (9, 3, 8)
```

[코드 31] 추가 부품에 대한 데이터 셋 저장 및 분리

- [그림 28]은 [코드 31]까지의 데이터 분석 흐름도이다.



[그림 28] 단계 ⑨ 데이터 분석 흐름도

- [단계 ⑦]과 동일한 LSTM 모델을 사용하되, 모델의 초매개변수(노드 수 등)를 수정하고 새로 병합된 부품 94와 부품 95의 데이터를 사용하여 부품 94의 미래 발주량을 예측하는 모델을 구축한다.

```
model = Sequential()
model.add(LSTM(32, dropout=0.2, activation='relu', input_shape=(3,8), return_sequences=True))
#(batch_size, time_steps, features)
model.add(LSTM(16, dropout=0.2, activation='relu'))
model.add(Dense(1, activation='linear'))

model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_2 (LSTM)                (None, 3, 32)             5248
_________________________________________________________________
lstm_3 (LSTM)                (None, 16)                3136
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 17
=================================================================
Total params: 8,401
Trainable params: 8,401
Non-trainable params: 0
_________________________________________________________________
```

[코드 32] 분석모델 구축

- 모델 학습에 필요한 옵티마이저(Adam)와 손실함수(MAE)를 설정하고, 'compile()' 메서드를 사용하여 모델에 전달한다.

```
model.compile(optimizer='adam', loss='mae')
```

[코드 33] 옵티마이저, 손실함수 정의

- 콜백 함수의 학습 조기 종료를 사용하여 모델을 학습시키고, 모델 체크포인트를 통해 학습 중 검증 데이터에 대하여 가장 좋은 예측력을 보이는 모델을 저장하는 경로

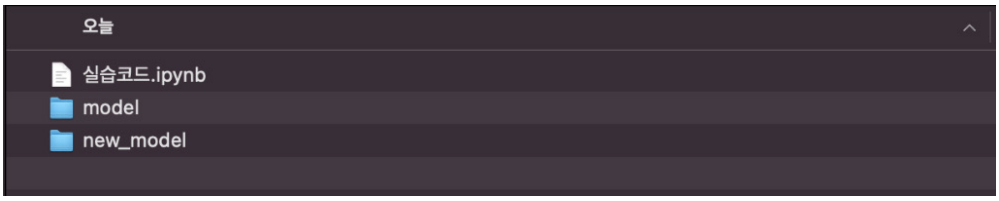
를 지정한다. [단계 ⑦-3]에서 설명한 것처럼, 인공신경망 기반 분석모델은 매번 동일한 학습 결과를 도출하지는 않기 때문에 아래 [코드 34]의 학습 결과와 다른 결과가 출력될 수 있다.

```
model_path = './new_model/{epoch:02d}-{val_loss:.4f}.hdf5'
callbacks = [EarlyStopping(monitor='val_loss', patience=30),
        ModelCheckpoint(filepath=model_path, monitor='val_loss',
                verbose=0, save_best_only=True)]

history = model.fit(X_train_both, y_train_94, epochs=100, batch_size=4,
                validation_data=(X_val_both, y_val_94),
                callbacks=callbacks)
```

```
Epoch 1/100
8/8 [==============================] - 3s 58ms/step - loss: 0.7841 - val_loss: 0.8396
Epoch 2/100
8/8 [==============================] - 0s 10ms/step - loss: 0.7745 - val_loss: 0.8415
...
...
...
Epoch 57/100
8/8 [==============================] - 0s 33ms/step - loss: 0.3312 - val_loss: 0.7135
Epoch 58/100
8/8 [==============================] - 0s 33ms/step - loss: 0.2425 - val_loss: 0.7181
```

[코드 34] 모델 학습 및 저장

- 혹여나 모델이 저장되는 파일 경로인 new_model 폴더가 자동으로 생성되지 않는 경우 [그림 29]처럼 직접 실습 코드와 동일한 폴더 내에 new_model 폴더를 직접 만든 후 [코드 34]를 사용하여 모델 학습 및 저장을 실시한다.



[그림 29] 실습 코드와 동일한 경로 내 model 폴더 생성

- [단계 ⑦-3]과 같이 학습 및 검증 손실값을 시각화한다.

```
loss = history.history['loss']
val_loss = history.history['val_loss']
epoch = history.epoch

plt.figure()
plt.plot(epoch, loss, 'r', label='Training loss')
plt.plot(epoch, val_loss, 'g', label='Validation loss')

plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Mean Absolute Error')
plt.legend()
plt.show()
```

[코드 35] 학습 및 검증 손실값 시각화

⑨-2. 추가 부품의 발주 데이터를 활용한 예측 모델의 예측값 도출 및 최종 성능 확인

- '.hdf5' 확장자로 저장된 학습 모델 중 가장 낮은 val_loss 값을 가진 좋은 모델을 불러온다. 가장 마지막에 저장된 모델의 val_loss가 낮으므로 모델이 저장되는 경로의 파일들 중 가장 마지막의 것을 선택한다.

```
from tensorflow.keras.models import load_model
from os import listdir
path = listdir('./new_model/')[-1]
best_model = load_model('./new_model/'+path)
```

[코드 36] 학습 모델 불러오기

- 앞서 생성해 둔 Keras 패키지의 Sequential 인스턴스의 'evaluate()' 메서드를 사용하여 평가 데이터셋에 대한 성능을 확인한다.

```
best_model.evaluate(X_test_both, y_test_94) # MAE
```

```
1/1 [==============================] - 1s 733ms/step - loss: 1.0879

1.087890863418579
```

[코드 37] 평가 데이터셋에 대한 성능 확인

- 마찬가지로 Sequential 인스턴스의 'predict()' 메서드를 사용하여 모델의 예측 결과를 확인할 수 있다.

```
y_pred_94 = best_model.predict(X_test_both)
y_pred_94
```

```
array([[-0.13641329],
       [-0.09419459],
       ...
       ...
       ...
       [ 0.5600838 ],
       [ 0.62408286]], dtype=float32)
```

[코드 38] 예측 및 예측 결과 확인

- 최종 출력 형태를 확인하기 위해 이전 단계에서 표준화한 값을 'inverse_transform()' 메서드를 통해 기존의 값 범위로 역변환(복원)한다.

```
y_pred_94_inv = yscaler_94.inverse_transform(y_pred_94)
y_test_94_inv = yscaler_94.inverse_transform(y_test_94)
```

[코드 39] 예측값을 기존의 값 범위로 복원

- 결과 분석을 위해 역변환된 예측값 및 평가 데이터의 값을 확인한다.

```
y_pred_94_inv
```

```
array([[40.41571263],
       [40.60686075],
       [41.26618218],
       [41.36151949],
       [42.52771563],
       [42.53347275],
       [42.8262922 ],
       [43.56915187],
       [43.85891208]])
```

[코드 40] 기존의 값 범위로 복원된 예측값 확인

```
y_test_94_inv
```

```
array([[35.],
       [35.],
       [38.],
       [45.],
       [45.],
       [47.],
       [45.],
       [45.],
       [28.]])
```

[코드 41] 기존의 값 범위로 복원된 평가 데이터 값 확인

- 최종 성능 출력을 위해 손실 함수로 쓰였던 평균 절대 오차(MAE)를 성능지표로 사용한다. scikit-learn 패키지의 'mean_absolute_error()' 함수로 예측 성능을 계산한다.

```
from sklearn.metrics import mean_absolute_error
mean_absolute_error(y_test_94_inv, y_pred_94_inv)
```

```
4.925501929389106
```

[코드 42] 예측 성능 계산

[단계 ⑩] 분석결과에 대한 논의 및 해석

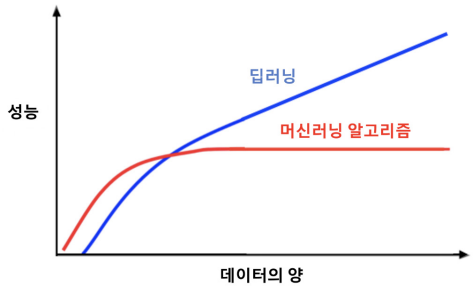
⑩-1. 분석 결과 해석 가이드

- [단계 ⑧]에서는 하나의 부품(부품 94)에 대한 D+3, D+4, D+5일 계획 발주량을 기반으로 미래 발주 수량 예측 모델을 구축하였고, [단계 ⑨]에서는 연관성이 있는 두 개의 부품(부품 94 및 부품 95)에 대한 D+3, D+4, D+5일 계획 발주량을 활용하여 단일 부품에 대한 미래 발주 수량 예측 모델을 구축하였다. 두 모델의 예측 성능을 비교한 [표 1]은 [단계 ⑨]에서 실시한 두 가지 부품을 모두 활용한 예측 모델의 성능이 우수하다는 것을 보여준다.

[표 1] 부품 1개와 2개를 이용한 두 모델의 발주 수량 예측 성능 비교

예측 성능	부품 94의 발주데이터만 사용	부품 94와 부품 95의 발주데이터 모두 사용
MAE Loss	1.2037	**1.0878**
MAE (inverse scaling)	5.4498	**4.9255**

- 동일한 제품 생산에 사용된 부품들이 서로 발주 수량에 연관이 있을 것이라는 가정('corr()' 메서드 등을 활용하여 수치적으로 부품간의 상호연관성을 파악할 수 있다) 하에, AI 분석모델 구축에 해당 부품들의 데이터를 모두 활용하여 재고량을 예측하는 것이 더 좋은 결과를 도출할 수 있을 것으로 예상된다. 이를 통해 유의미한 정보가 데이터에 추가되면 분석모델의 성능이 향상될 수 있음을 확인하였다.
- 수집된 데이터가 많을수록 정확한 예측 모델을 구축할 수 있는 것으로 알려져 있다. 본 실습에서는 학습 데이터로 30일간 수집한 발주 및 계획 발주 수량 데이터로 발주량 예측 모델을 구축하였다. [그림 30]처럼, 일반적으로 LSTM과 같은 딥러닝 알고리즘은 양질의 데이터를 충분히 확보했을 때 성능이 향상되는 경향이 있다. 다량의 데이터를 추가로 수집하여 모델 학습에 활용한다면 본 실습에서 도출한 성능보다 향상된 결과를 기대할 수 있다. 뿐만 아니라, 발주 수량 데이터 외에 제품, 프로모션 등 발주에 영향을 미치는 다양한 변수 데이터를 함께 활용한다면 보다 정교하고 정확한 예측 모델 구축이 가능할 것이라 예상한다.

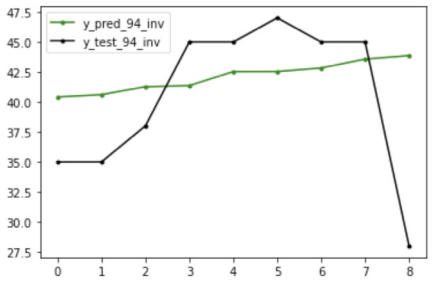


[그림 30] 데이터양에 따른 모델의 성능

⑩-2. 분석 결과 활용 가이드

- 제품 생산을 위한 라인 투입이 될 수 있도록 VMI 적차장에 부품별 재고가 필요량만큼 적재되어 있어야 한다. 부품 공급사의 자사 창고보다 VMI 적차장에서 보유 가능한 부품의 수가 훨씬 적기 때문에, 효율적인 VMI 적차장 내 재고 관리가 이루어지는 것이 필수적이다. 본 해석 단계에서는 부품 94와 부품 95의 과거 발주량 데이터를 동시에 활용하여 부품 94의 발주량을 예측한 [단계 ⑨]의 결과를 기반으로 적절한 재고 관리를 위한 솔루션을 제공한다.
- [단계 ⑨]에서 도출한 예측값(y_pred_94_inv)과 실제값(y_test_94_inv)을 시각화하였다. 최종 예측 성능은 [코드 43]에 따르면 MAE 기준 약 5이므로, 예측 수량보다 5개 이상의 재고를 VMI 적차장에서 보유하고 있는 것이 적절하다는 것을 알 수 있다.

```
plt.plot(y_pred_94_inv, 'g.-',label="y_pred_94_inv")
plt.plot(y_test_94_inv, 'k.-',label="y_test_94_inv")
plt.legend()
plt.show()
```



[코드 43] 결과 시각화

3. 유사 타 현장의 『공급망 최적화 AI 데이터셋』 분석 적용

3.1 본 분석이 적용 가능한 제조현장 소개

- 본 실습에서는 VMI 적차장 내 최적의 재고 관리를 위해 공급망 최적화를 위한 AI 기반의 발주량 예측 모델을 구축하였다. 공급망 최적화 기술은 발주량 예측 뿐만 아니라 다양한 목적으로 적용될 수 있는데, 공급망을 구성하는 변동적인 요소들의 예측에도 활용 가능하다. 변수 간의 상관관계를 파악하고, 독립 변수를 활용하여 종속 변수의 값을 예측하는 분석을 통해 정확도가 높은 물류량 예측, 소비자 수요 예측 등이 가능할 것으로 예상된다. 특히, AI 분석모델을 활용하여 공급망 내의 다양한 예측을 실시할 때, 관리자의 직접적인 관여 없이도 재고 관리 및 물류 이동이 최적화될 수 있으며, 고객사 등의 니즈의 변화와 공급망 움직임에 대한 계획 수립 및 실행 전략이 유연해질 수 있다. 더 나아가, 급격하게 변하는 시장 수

요 변화에 대한 대응성 역시 기대할 수 있을 것이다.

- 전통적으로 자동차 산업 및 자동차 부품 공급망 관리에서는 기업 간 통상적인 발주량을 평균적으로 산출하여 재고관리를 하는 방식으로 진행되어 왔다. 자동차 산업에 들어가는 수많은 부품 수에 정확히 부합하는 발주량을 예측하여 대비하는 것은 불가능한 일로 여겨지고 있었으나, 본 분석을 통해 구축한 AI 분석모델의 예측 결과를 바탕으로 재고 적정성을 확보할 수 있을 것으로 기대된다.

3.2 본 『공급망 최적화 AI 데이터셋』 분석을 원용하여 타 제조현장 적용 시, 주요 고려사항

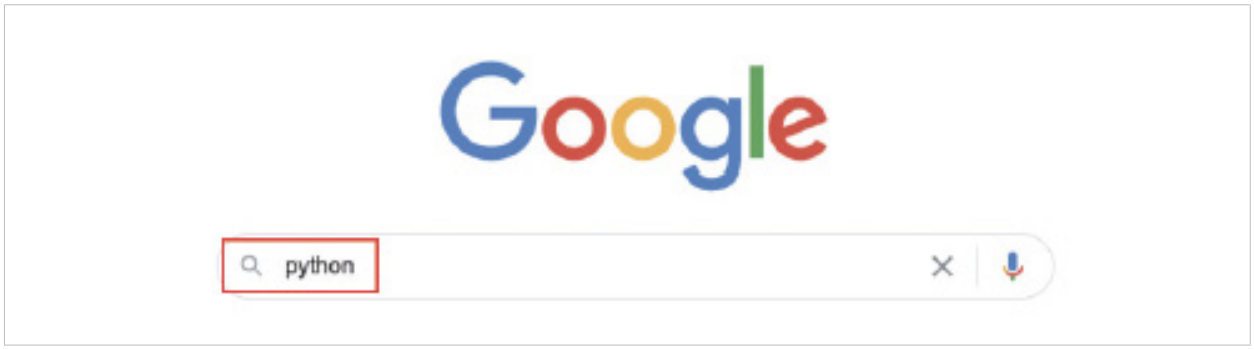
- 예측하고자 하는 값과 연관 있는 변수들을 사용하거나, 필요시 양질의 데이터를 새로 수집하는 과정이 요구된다.
- 당일 발주 수량 및 계획 발주 수량이 누락 없이 정확하게 기입되고 있는지 확인하는 것이 필요하다. 정확하지 않은 데이터를 분석하고 AI 모델을 학습시키는 것은 무의미하기 때문이다.
- 과거와 현재의 데이터 간에 큰 변화가 있는 경우, 최근 시점의 변화에 대한 가중치를 두어 예측할 필요가 있다. 같은 맥락에서, 불필요한 과거의 데이터를 제거하여 양질의 데이터만으로 예측 정확도를 높이는 등의 학습 유효성을 고려하는 것이 중요하다. 예를 들어, 코로나-19와 같이 예측하지 못한 돌발 사건으로 인한 큰 변화가 있을 때 기존의 분석모델을 활용한 예측 결과는 무의미하게 된다. 새로운 예측 결과를 도출하기 위해서는 새로운 데이터를 수집하거나 정제하고, 변수의 종류와 가중치를 모두 새롭게 구성하는 과정이 필요하다.
- 수집하는 데이터의 형태에 대한 논의 및 관리가 필수적이다. 데이터를 수집하는 현장 관리자와 데이터 분석 및 모델 구축을 담당하는 분석 전문가의 충분한 소통을 통해, 분석에 필요한 데이터의 형태를 논의하여 가공 가능한 형태로 데이터를 수집하고 저장함으로써 AI 분석모델 적용의 토대를 구축하는 것이 중요하다.

3 부록_분석환경 구축을 위한 설치 가이드

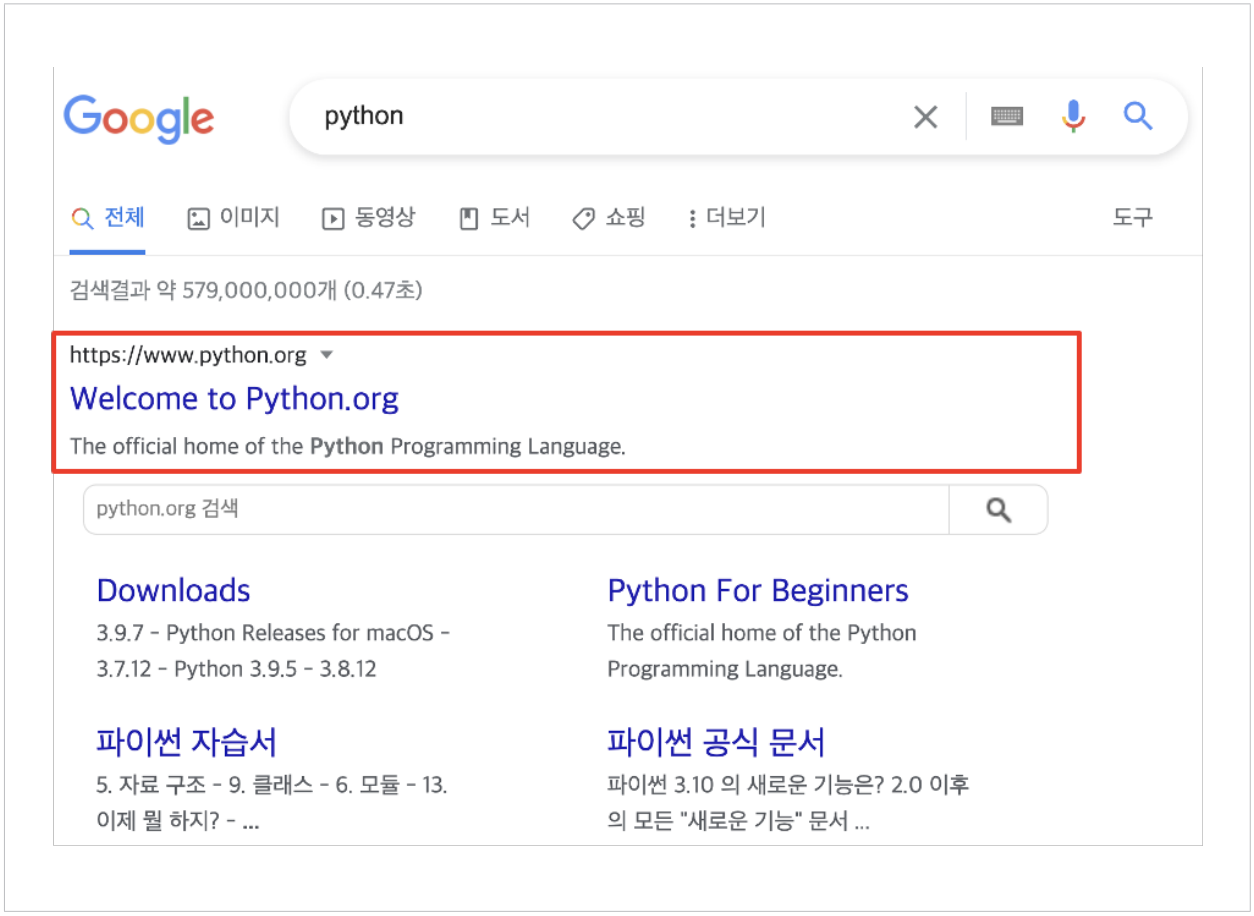
1. 파이썬(Python) 설치

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.



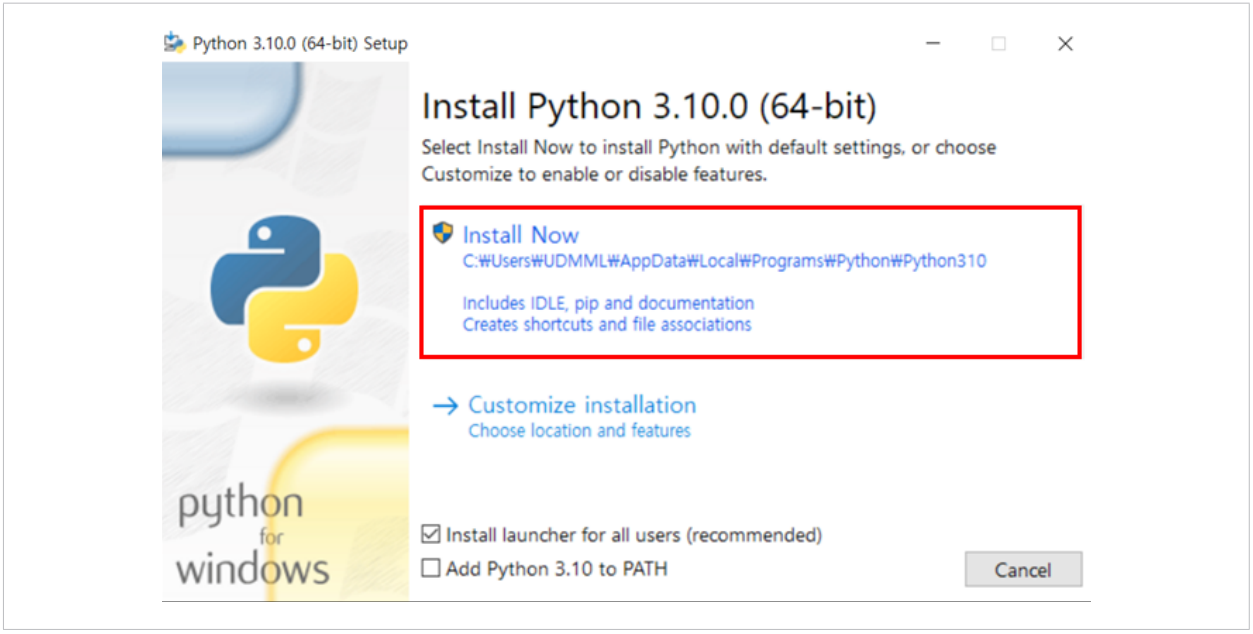
② 제일 처음에 보이는 'Welcome to Python.org'를 클릭한다.



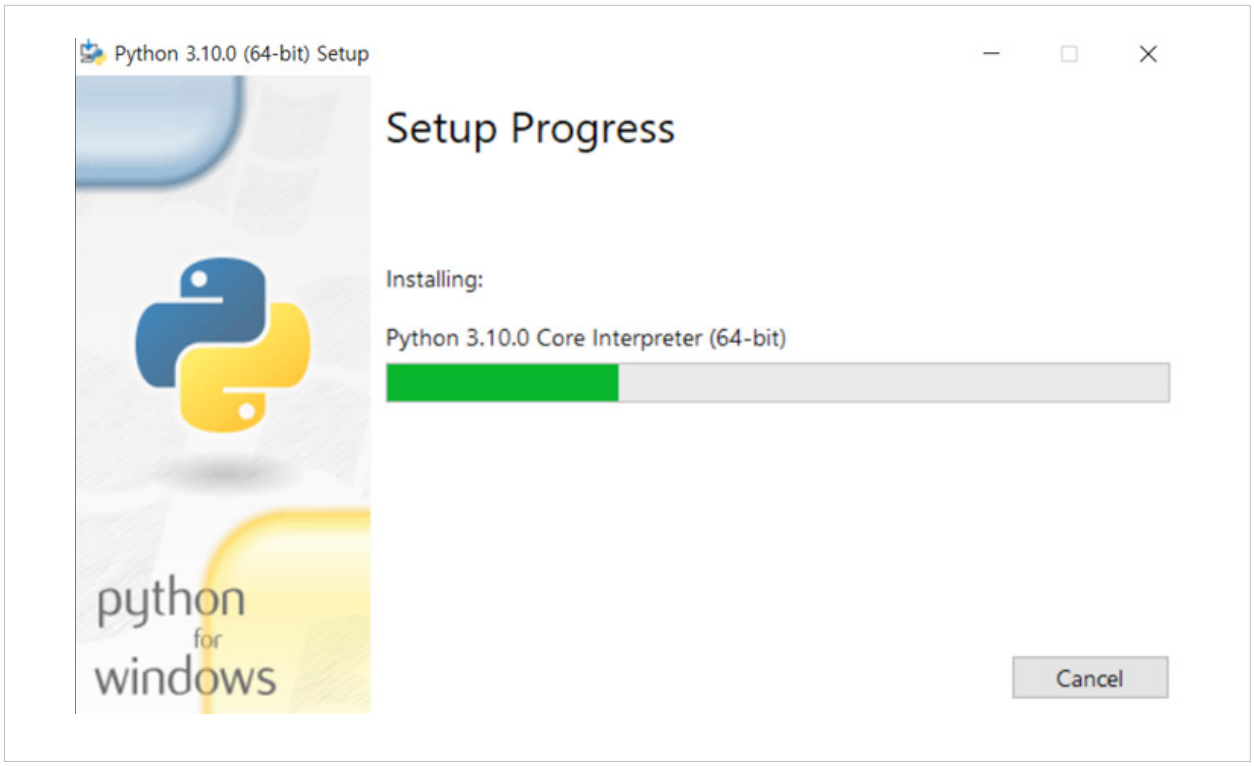
③ 클릭후 보이는 페이지의 정면에서 상단 좌측 2번째 'Downloads'를 클릭한다.

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)

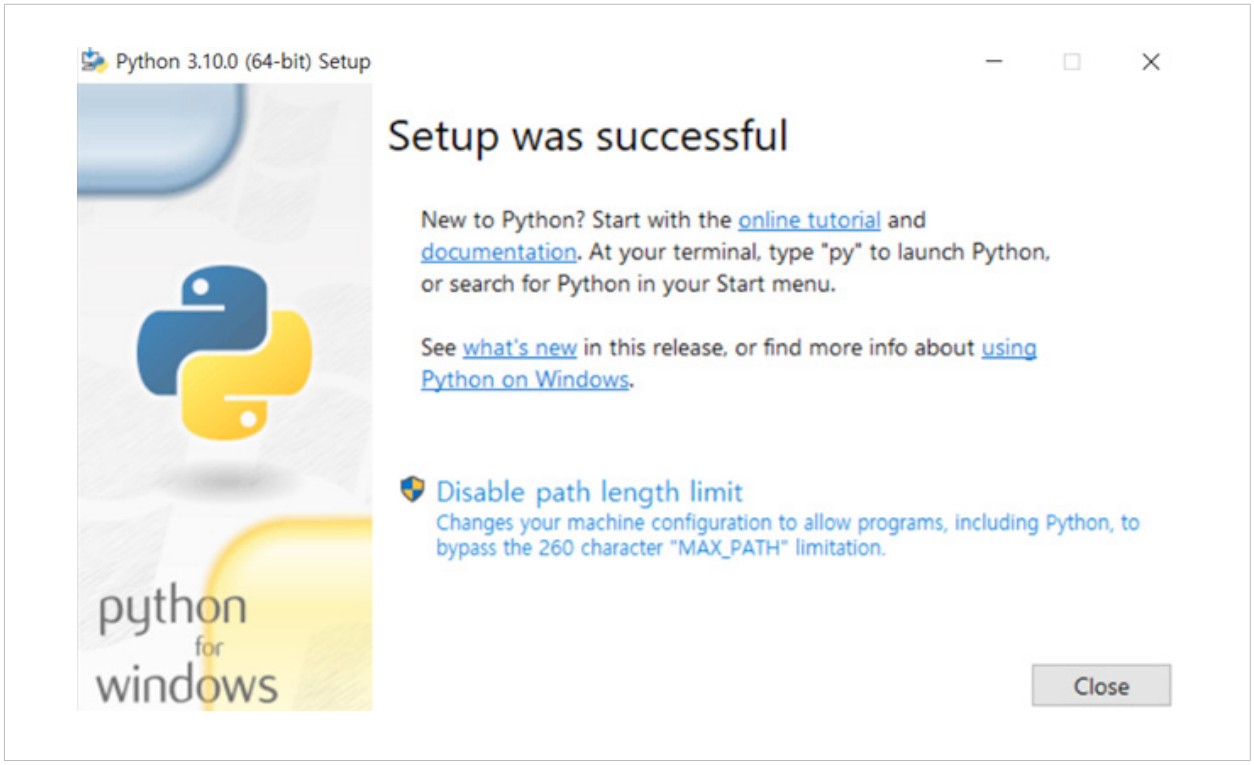
⑤ 아래와 같은 설치창이 뜨면, 'Install Now'를 클릭한다.



⑥ 아래와 같은 설치 진행창이 완료가 될 때까지 유지한다.



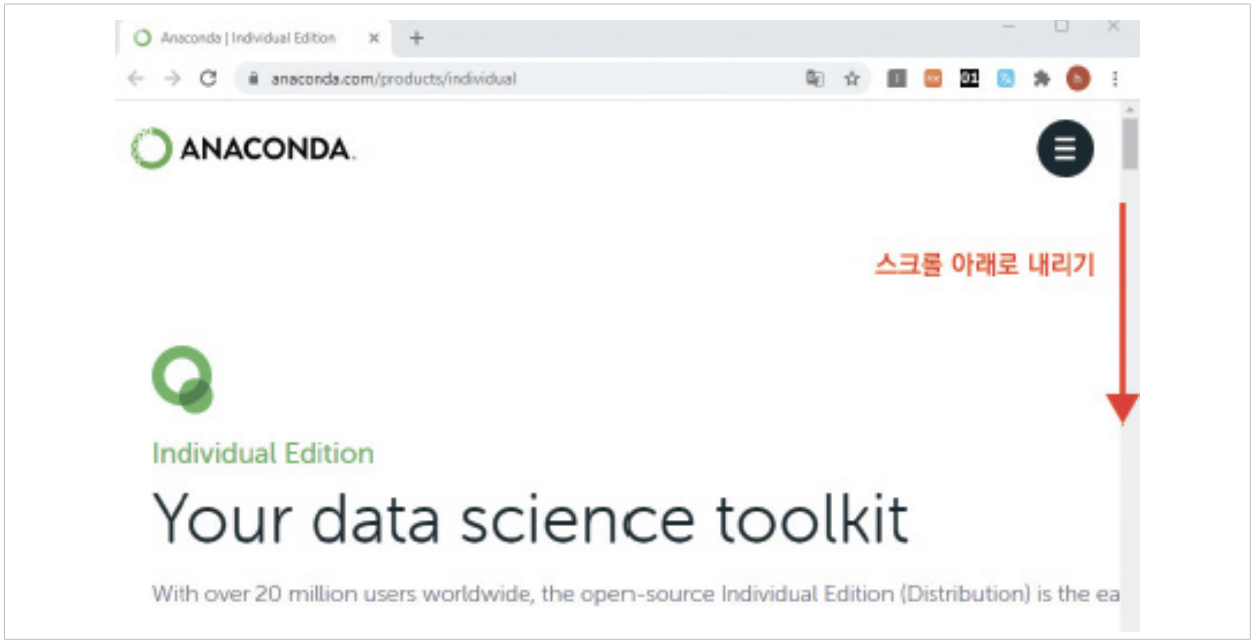
⑦ 완료가 되면 아래와 같은 창이 뜨는 것을 확인 후 종료한다. [설치완료]



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.

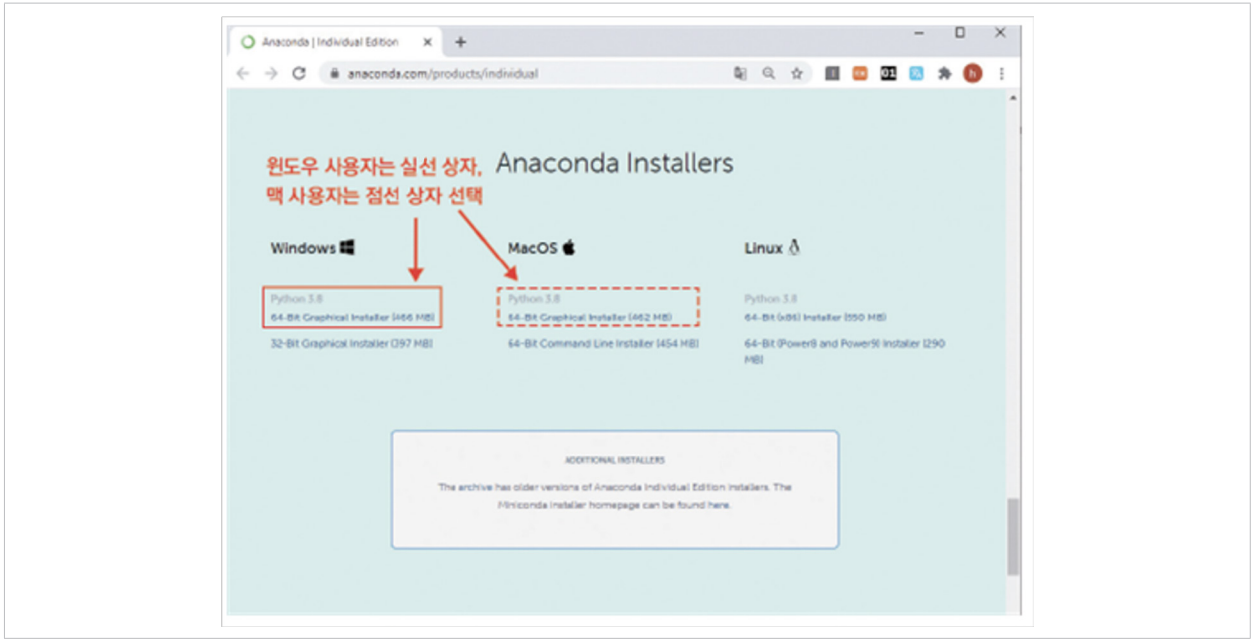
① https://www.anaconda.com/distribution/ 로 접속한다.



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

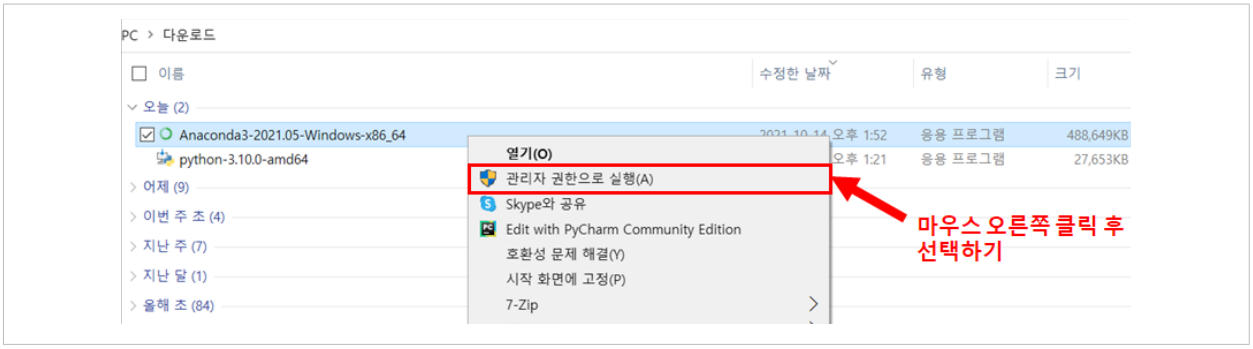
▶ Windows : 64-Bit Graphical Installer

▶ MacOS : 64-Bit Graphical Installer

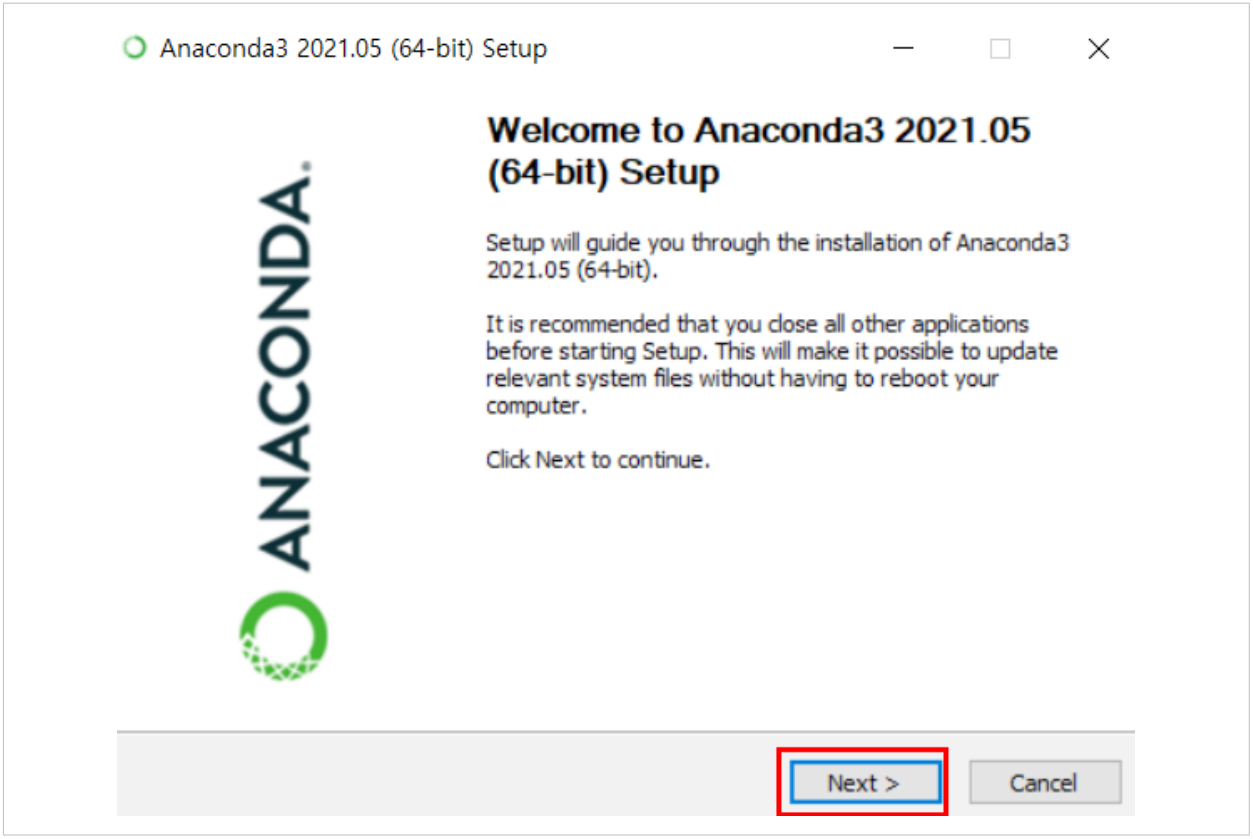


③ 다운로드 받은 파일로 이동하여, 설치된 아나콘다 파일 위에서 마우스 오른쪽을 클릭한 후, 방패모양의 '관리자 권한으로 실행'을 클릭한다.

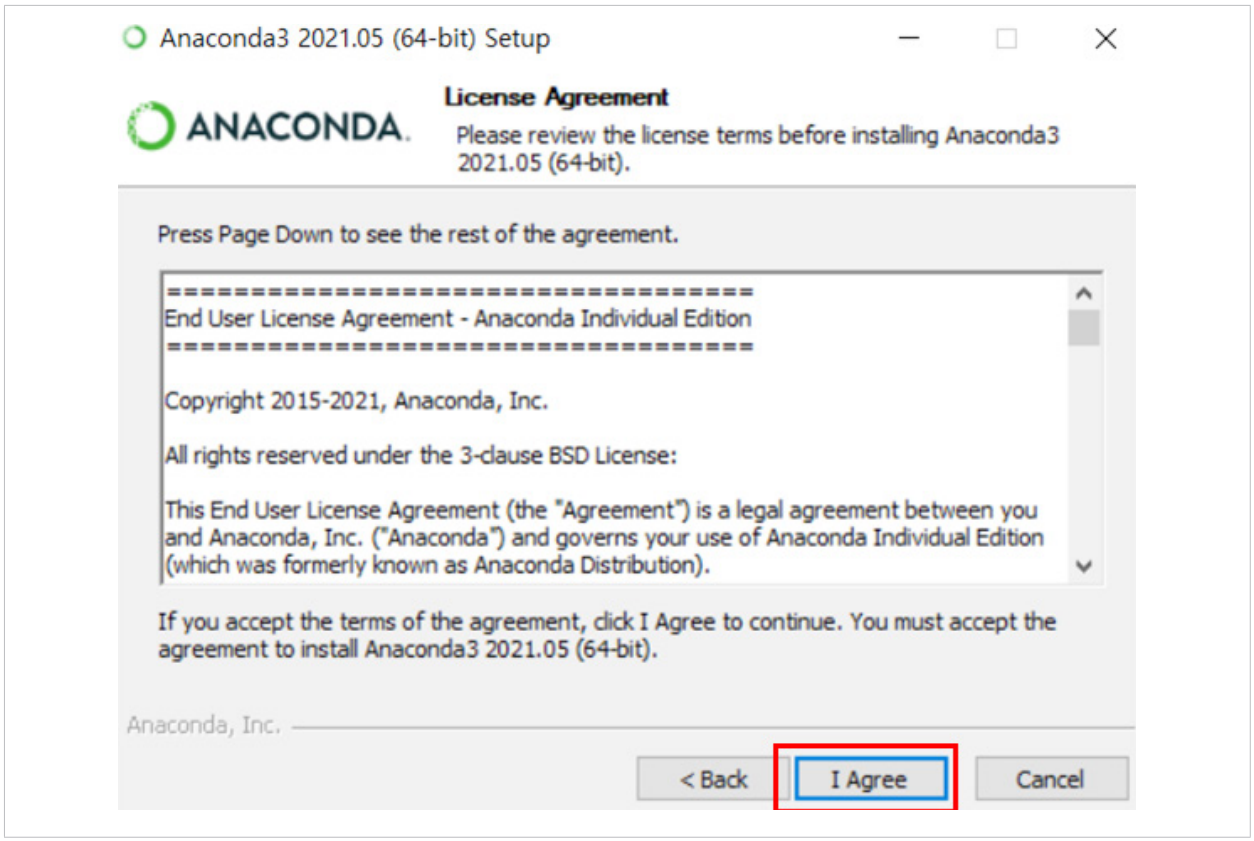
▶ (예) '다운로드' 폴더로 아나콘다를 다운 받은 경우



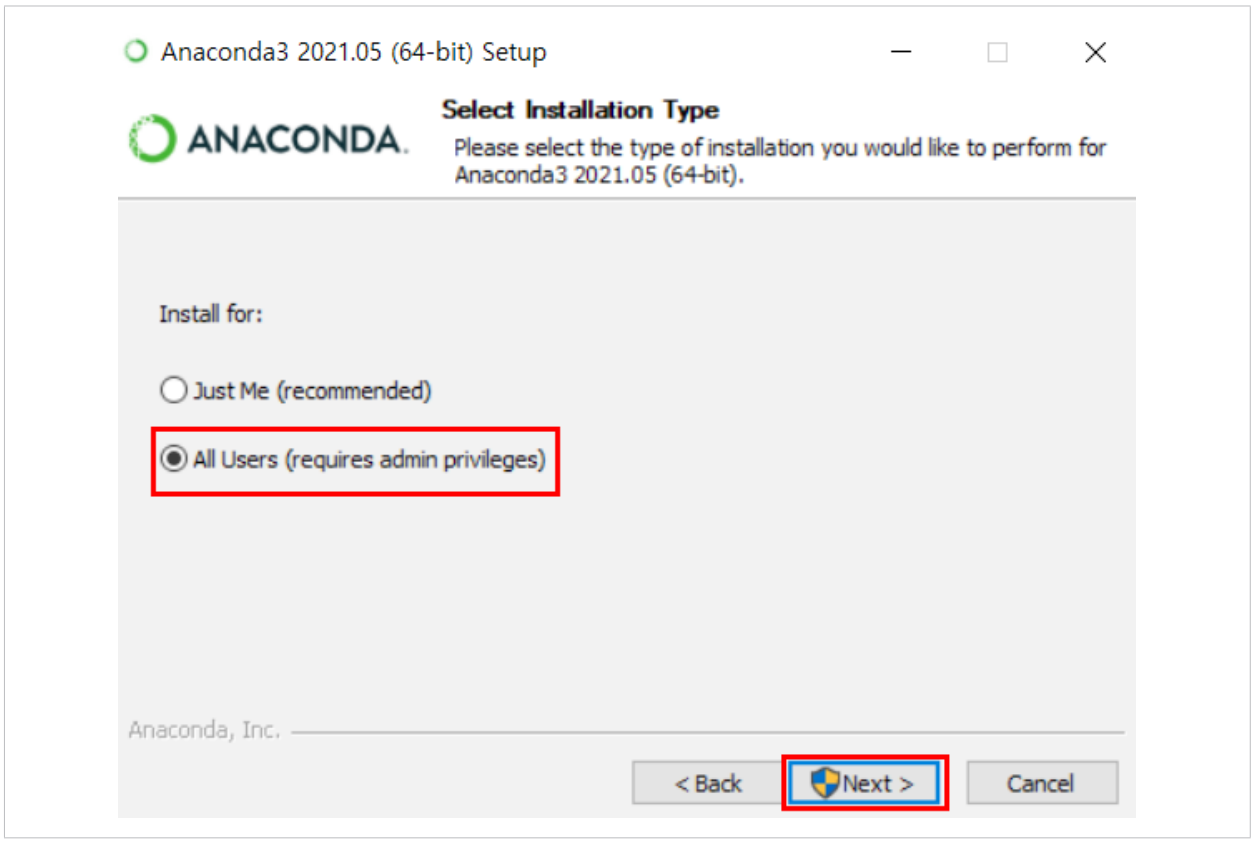
④ 파일을 실행 한 후, 'Next' 버튼을 클릭한다.



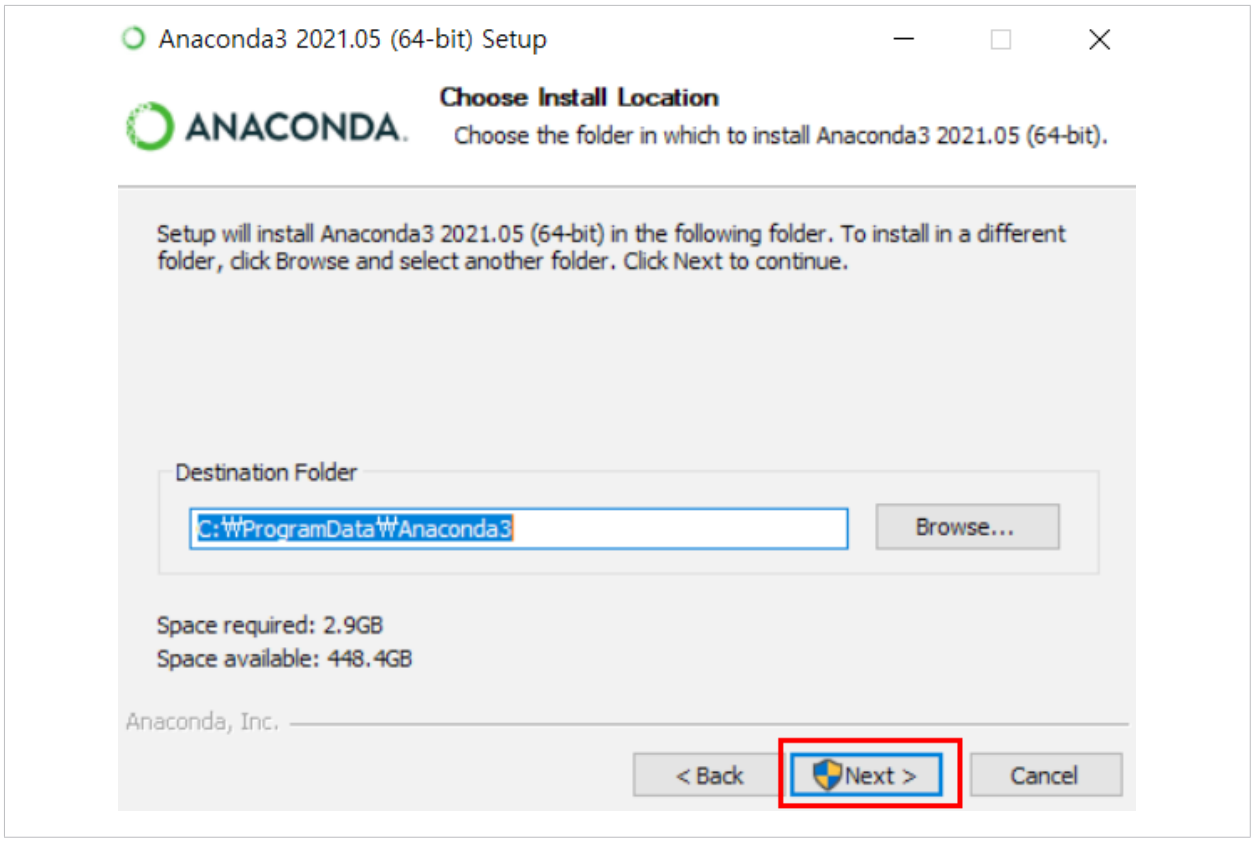
⑤ 다음 창이 나타나면 'I Agree'를 선택한다.



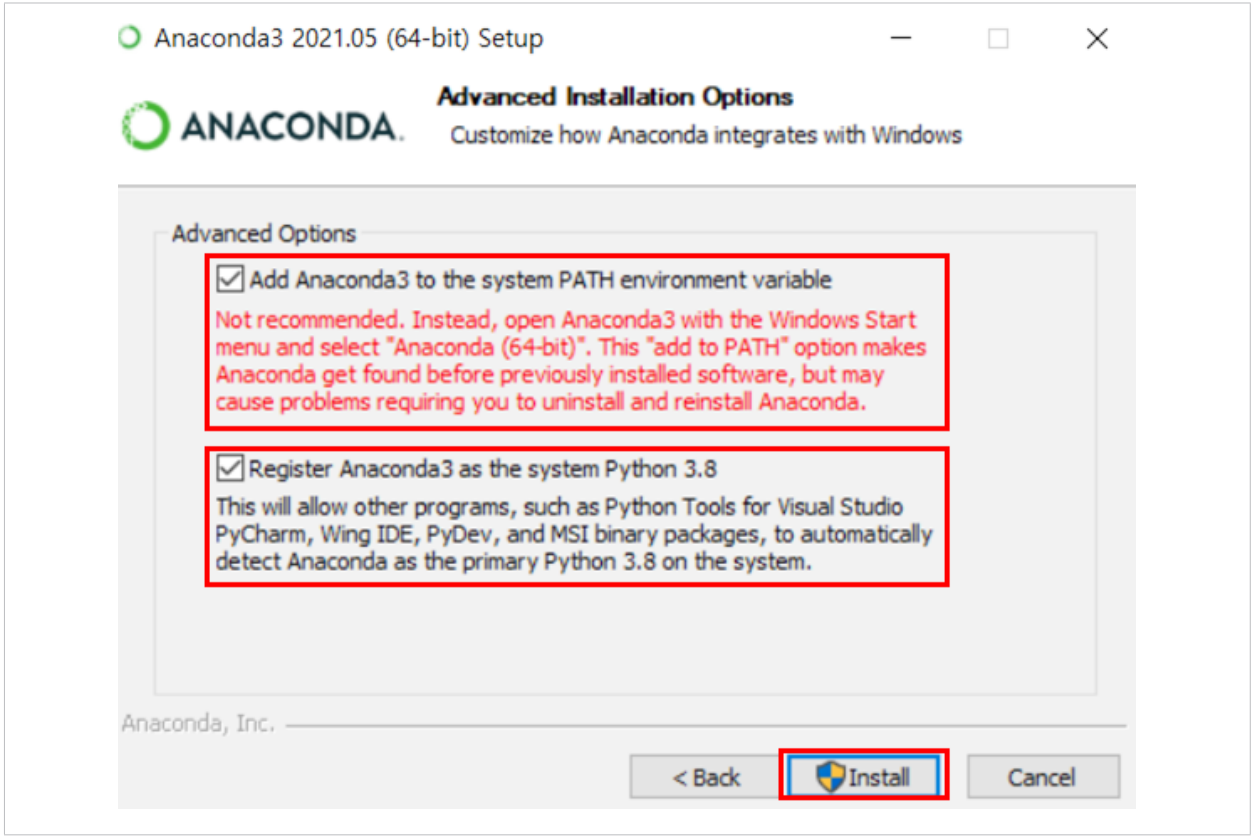
⑥ 셋팅 창이 뜨면 화면의 'All Users'를 선택 후 아래의 'Next'를 클릭한다.



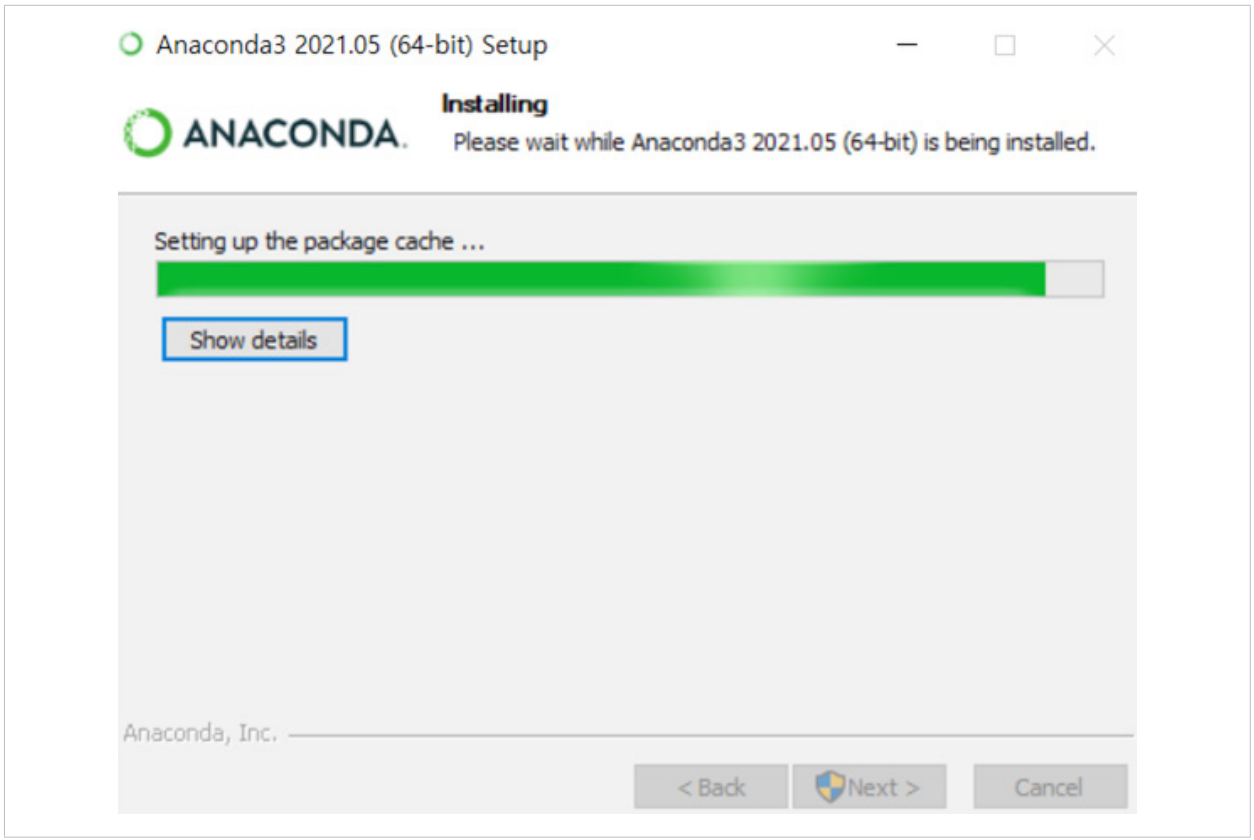
⑦ 다운로드 받을 경로를 물어보는 창이 뜨면, 아래의 'Next'를 클릭한다.



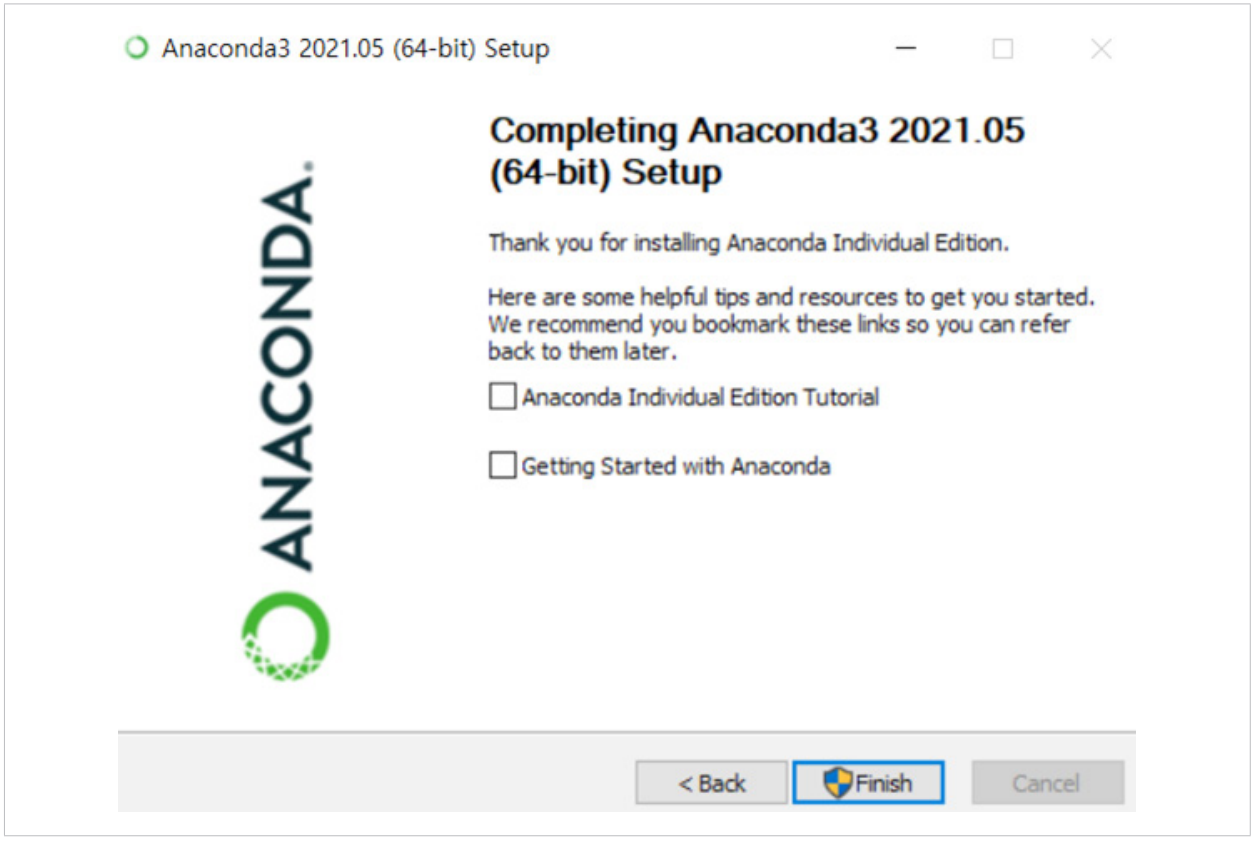
⑧ 고급 옵션 선택창이 뜨면, 아래와 같이 모두 선택 후, 'Install'을 클릭한다.



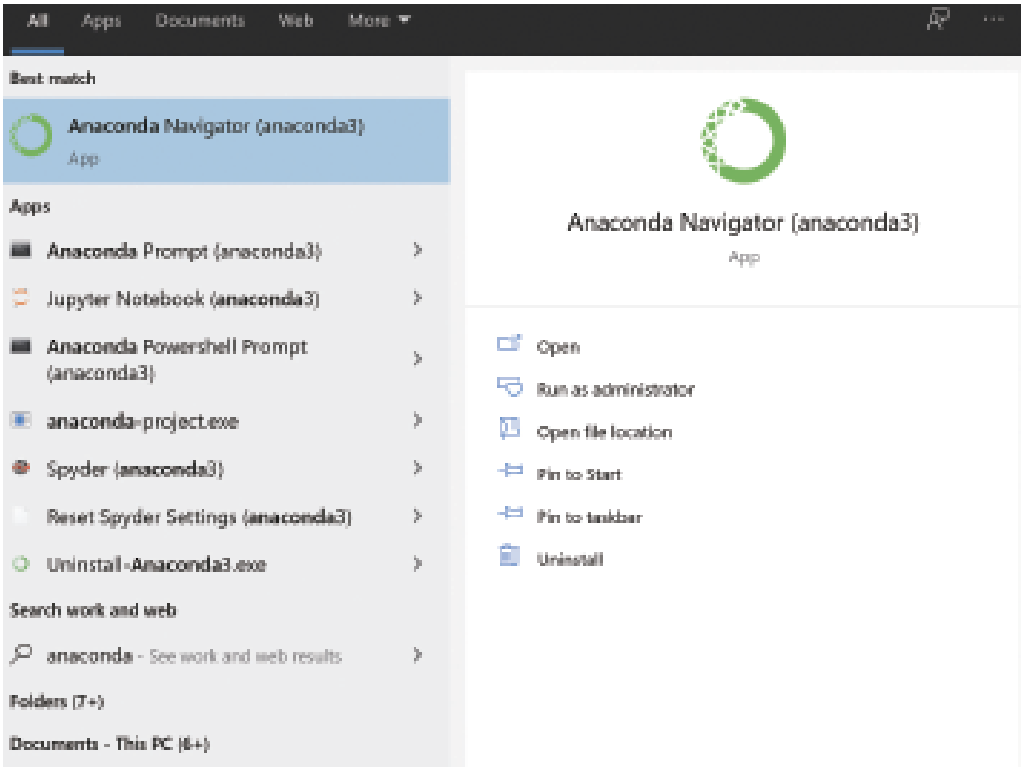
⑨ 다음과 같은 설치창이 뜨면 완료가 될 때까지 대기 (5분이상 소요)



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



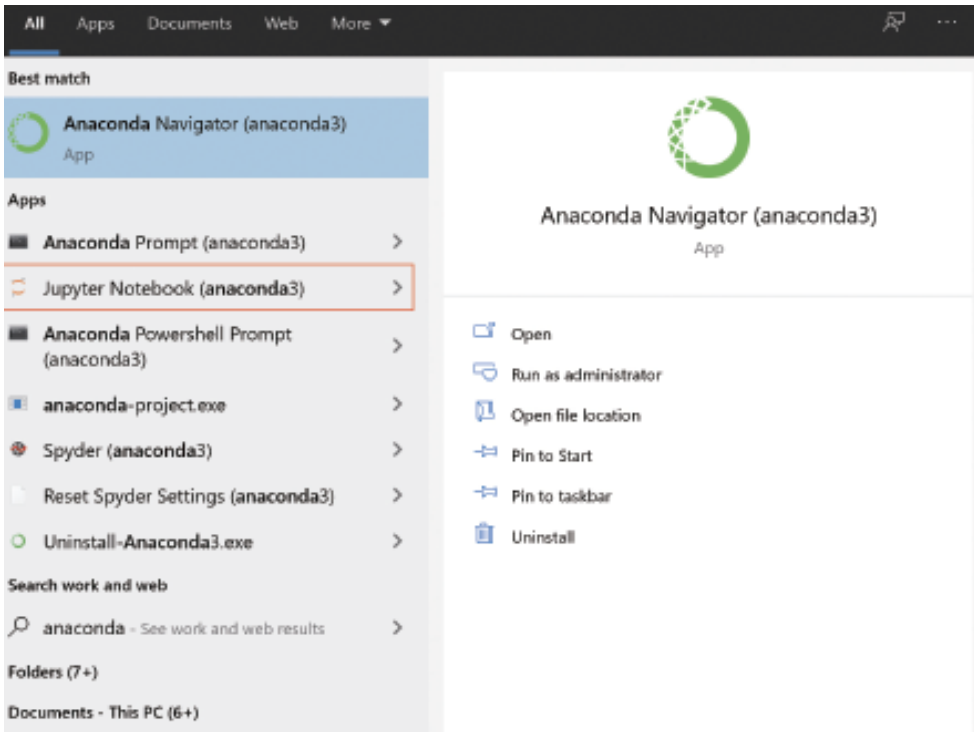
⑪ 화면상의 '홈( )'키를 눌러서 화면과 같이 anaconda prompt가 잘 설치 되었는지를 확인한다. Anaconda Navigator, Anaconda Prompt, Jupyter Notebook 등의 다른 응용 프로그램들도 함께 확인이 된다면 설치가 완료된 것이다.



3. 주피터 노트북 (Jupyter Notebook) 실행

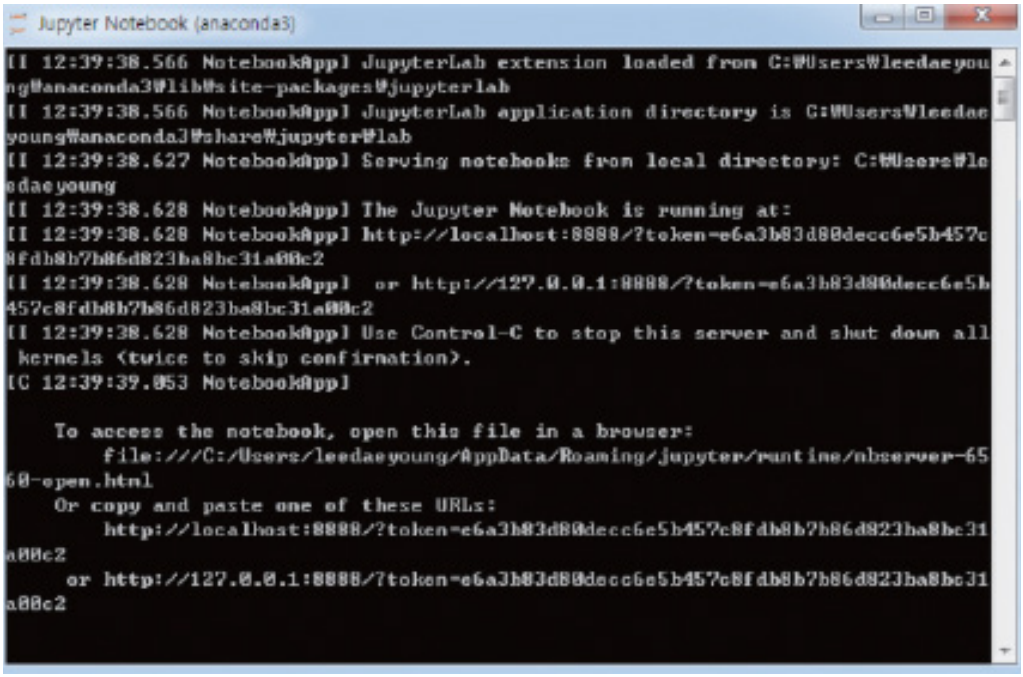
주피터 노트북은 실제로 사용자가 코딩(분석 문장 작성)을 할 수 있는 도구이다. 쉽게 비교하자면, 문서 도구로 마이크로소프트 사의 'Word' 프로그램이나, 한컴소프트 사의 '한글' 프로그램 등과 같은 도구라고 생각할 수 있다. 데이터 분석에 다양한 입력, 실행 도구가 있지만, 본 가이드북에서는 주피터 노트북을 활용하는 방법을 안내하기로 한다. Anaconda를 설치한 이후 주피터노트북(Jupyter Notebook) 설치 방법을 확인하면 된다.

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

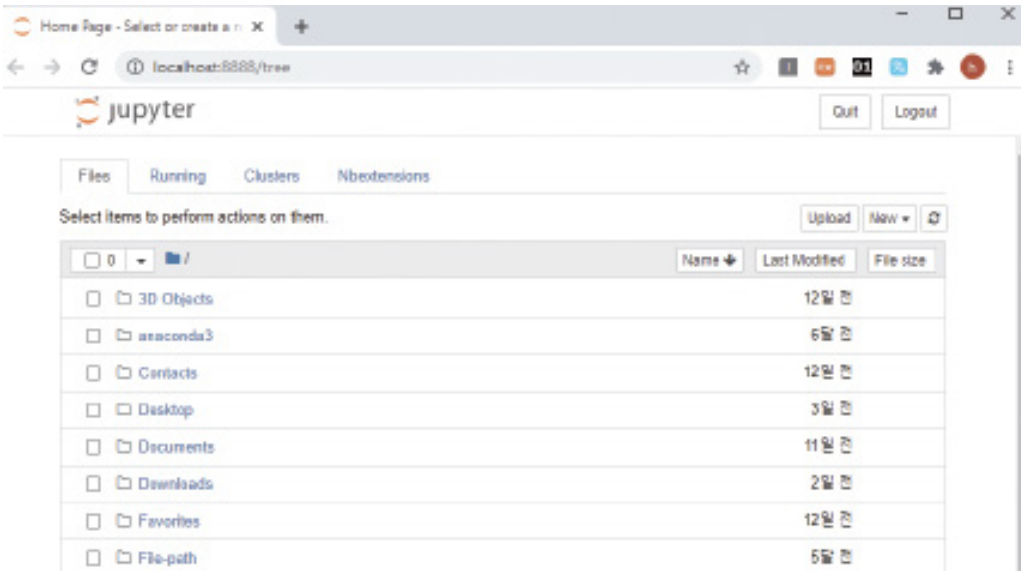


② Jupyter Notebook을 클릭시 아래처럼 2개의 윈도우가 실행된다.

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



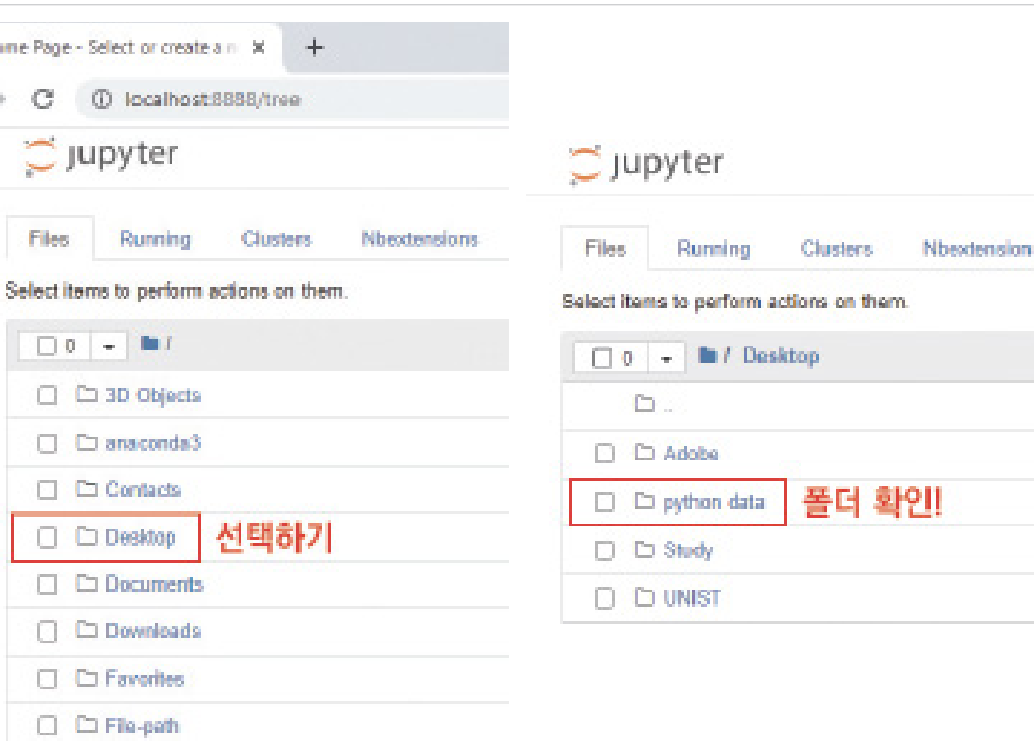
(2) 사용하는 인터넷 프로그램(크롬, 인터넷 익스플로러)에 주피터 노트북이 열린다. (다른 확장 프로그램 사용하고 싶다면 기본 브라우저를 변경해 주어야 한다.)



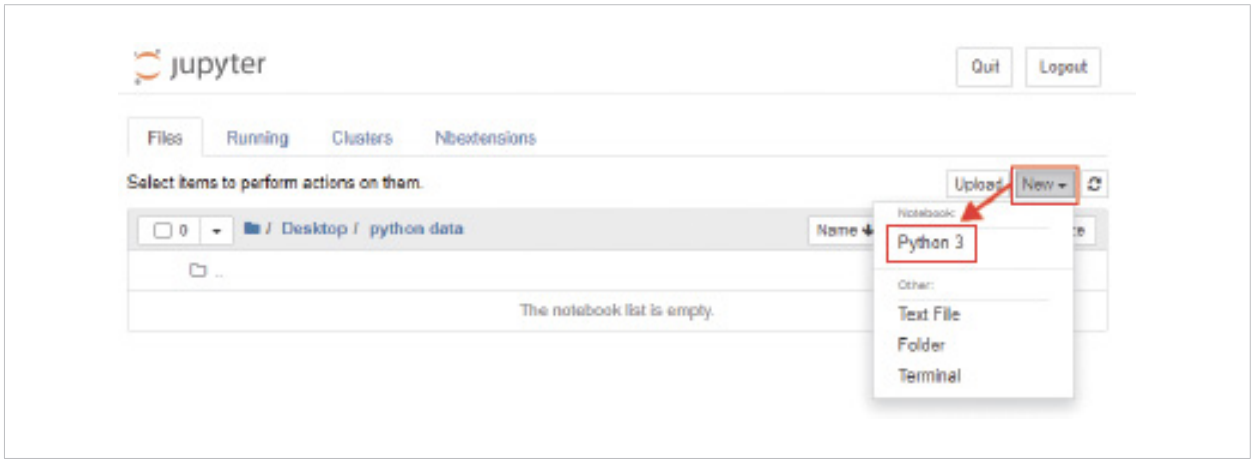
③ 데이터를 저장하고, 불러오고, 분석할 경로의 폴더를 하나 생성한다.

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

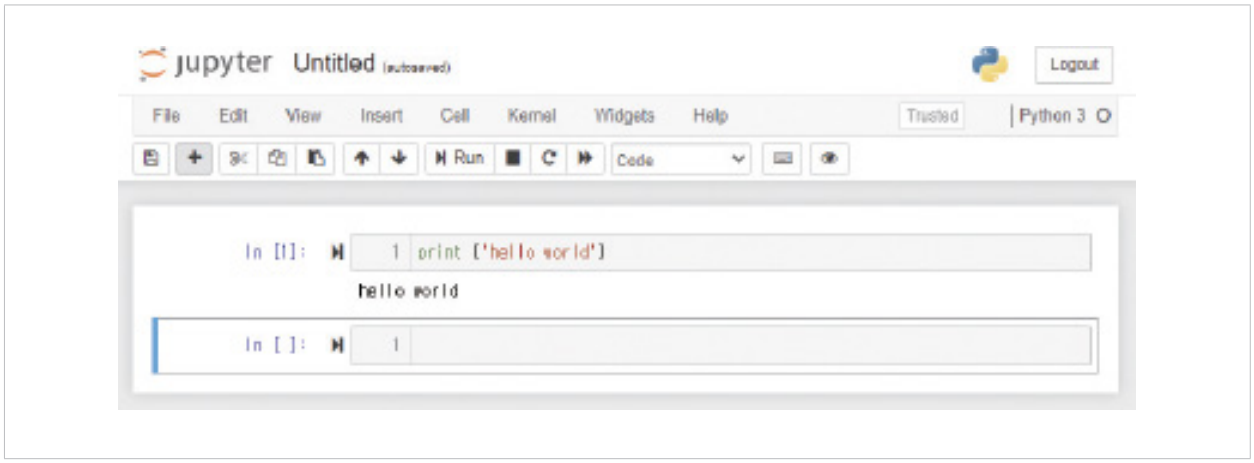
(1) 주피터 노트북에서 'python data' 폴더를 확인한다.



(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 ▶| Run 버튼 혹은 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, Gartner에서 제시한 6종의 데이터 품질 평가지표에 대하여 논의하였다. (완전성, 유일성, 유효성, 일관성, 정확성, 무결성)

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

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

```
perc=30
data.isna().sum()/len(data)*100
```

Part Number	0
D일06~08(08)H 투입계획(발주) 수량	0
D일08~10(10)H 투입계획(발주) 수량	0
D일10~12(13)H 투입계획(발주) 수량	0
D일13~15(15)H 투입계획(발주) 수량	0
...	...
D+28일 투입예정 수량	0
D+29일 투입예정 수량	0
D+30일 투입예정 수량	0
D+31~D+45일 투입예정 수량	0
CRET_TIME	0

```
perc=30
data.isna().sum()/len(data)*100>perc
```

Part Number	False
D일06~08(08)H 투입계획(발주) 수량	False
D일08~10(10)H 투입계획(발주) 수량	False
D일10~12(13)H 투입계획(발주) 수량	False
D일13~15(15)H 투입계획(발주) 수량	False
...	...
D+28일 투입예정 수량	False
D+29일 투입예정 수량	False
D+30일 투입예정 수량	False
D+31~D+45일 투입예정 수량	False
CRET_TIME	False

[코드 1] 분석 도구를 활용한 완전성 품질 지수 확인

② 데이터의 결측치를 확인하기 위해 'isna()' 메서드를 사용한 뒤 'sum()' 메서드를 이용하여 총 결측치 개수를 구한다.

```
data.isna()
```

	Part Number	D일06~08(08)H 투입계획(발주) 수량	D일08~10(10)H 투입계획(발주) 수량	D일10~12(13)H 투입계획(발주) 수량	D일13~15(15)H 투입계획(발주) 수량
0	False	False	False	False	False
1	False	False	False	False	False
2	False	False	False	False	False
3	False	False	False	False	False
4	False	False	False	False	False
...	...	...	...	...	...
17359	False	False	False	False	False
17360	False	False	False	False	False
17361	False	False	False	False	False
17362	False	False	False	False	False
17363	False	False	False	False	False

[코드 2] 'isna()' 메서드를 활용한 결측치 확인

```
data.isna().sum()
```

```
Part Number                    0
D일06~08(08)H 투입계획(발주) 수량     0
D일08~10(10)H 투입계획(발주) 수량     0
D일10~12(13)H 투입계획(발주) 수량     0
D일13~15(15)H 투입계획(발주) 수량     0
                              ..
D+28일 투입예정 수량                  0
D+29일 투입예정 수량                  0
D+30일 투입예정 수량                  0
D+31~D+45일 투입예정 수량             0
CRET_TIME                      0
Length: 84, dtype: int64
```

[코드 3] 'sum()' 메서드를 이용하여 컬럼별 결측치 개수 확인

```
cmpt_len=data.isna().sum().sum()
print(cmpt_len)
```

```
0
```

[코드 4] 분석 도구를 활용한 총 결측치 개수 확인

③ 구한 결측치의 개수를 이용하여 완전성 품질 지수를 구한다.

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

```
결측치 = 0개
완전성 지수 : 100.00%
```

[코드 5] 분석 도구를 활용한 완전성 품질 지수 구하기

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

- 본 데이터는 'Part Number'별 'CRET_TIME' 열이 유일한 값을 가지는 기본키이다.

① 'groupby()' 메서드를 이용하여 'Part Number'별 'CRET_TIME' 열이 가지는 유일한 값들을 확인하고, 유일한 데이터 개수를 구한다.

```
data.groupby(['Part Number','CRET_TIME']).size()
```

```
Part Number  CRET_TIME
Part 0       2021-09-13 18:30:00    1
             2021-09-14 06:05:00    1
             2021-09-14 06:25:00    1
             2021-09-14 06:34:00    1
             2021-09-14 17:30:00    1
                                   ..
Part 99      2021-10-29 12:34:00    1
             2021-10-29 15:03:00    1
             2021-10-29 15:38:00    1
             2021-10-30 07:04:00    1
             2021-11-01 07:03:00    1
Length: 17058, dtype: int64
```

[코드 6] 분석 도구를 활용한 유일한 값들 확인하기

```
len(data.groupby(['Part Number','CRET_TIME']).size())
```

```
17058
```

[코드 7] 분석 도구를 활용한 유일한 값들의 개수 확인하기

② 구한 유일한 데이터 개수를 이용하여 유일성 품질지수를 구한다.

```
uniq_len=len(data.groupby(['Part Number','CRET_TIME']).size())
print("유일성 지수 : %.2f%% "%((uniq_len/len(data))*100)) #결과는 아래에서 확인 가능하다.
```

```
유일성 지수 : 98.24%
```

[코드 8] 분석 도구를 활용한 유일성 품질지수 구하기

③ 전처리 이후의 유일성 지수 값

- 본 데이터에는 모든 발주 주문 로그가 기록되어 있다. 즉, 최종 주문 로그뿐만 아니라 모든 발주 주문 로그 데이터들이 수집되어있다. 전처리과정에서 불필요한 행들을 제거하여 최종 일일 발주 주문 데이터만 저장하여 분석에 사용한다. 불필요한 행을 제거한 데이터의 유일성 지수는 아래와 같다.

```
# [단계 ③] 데이터 정제(전처리) 과정을 거친 데이터 valid_dt로 유일성 품질지수를 구한다
valid_dt = data[(data["Part Number"]=='Part 94')]
valid_dt.reset_index(drop=True, inplace=True)
valid_dt = valid_dt.loc[:,['Part Number','D일 투입예정 수량(D일계획)','D+1일 투입예정 수량(Total)',
                           'D+2일 투입예정 수량(Total)','D+3일 투입예정 수량(Total)','D+4일 투입예정 수량(Total)', 'D+5일 투입예정 수량', 'D+6일
투입예정 수량','D+7일 투입예정 수량','D+8일 투입예정 수량', 'D+9일 투입예정 수량','D+10일 투입예정 수량','D+11일 투입예정 수량','D+12
일 투입예정 수량', 'CRET_TIME']]
valid_dt['CRET_TIME'] = pd.to_datetime(valid_dt['CRET_TIME'], format="%Y%m%d%H%M")
valid_dt['CRET_TIME'] = pd.to_datetime(valid_dt['CRET_TIME'], format="%Y%m%d%H%M")
valid_dt = valid_dt.groupby(by=[valid_dt['CRET_TIME'].dt.year,
                                valid_dt['CRET_TIME'].dt.month,
                                valid_dt['CRET_TIME'].dt.day]).last()
valid_dt.reset_index(drop=True, inplace=True)

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

```
유일성 지수 : 100%
```

[코드 9] 분석 도구를 활용한 유일성 품질지수 구하기

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

① 데이터가 유효 범위 내에 있는가?

- 데이터셋에 정의된 수집 범위를 이용하여 조건(유효 범위)을 모두 충족하는 데이터들을 vald_dt에 저장한다. 본 데이터는 전체 부품 117개의 발주 주문 로그가 저장되어 있는 데이터다. 'nunique()' 메서드를 사용하여 고유한 'Part Number'의 개수가 117개인지 확인을 한다.

```
print(data['Part Number'].nunique())
```

```
117
```

[코드 10] 분석 도구를 활용하여 유효 범위 내에 있는지 확인하기

- 모든 데이터가 유효 범위 내에 있는 것을 알 수 있다.

```
vald_dt=data
```

[코드 11] 분석 도구를 활용하여 유효 범위내에 있는 데이터를 vald_dt에 저장하기

② 데이터가 형식에 맞는가?

- 'CRET_TIME' 열의 타입을 datetime으로 변경할 때 errors='coerce' 옵션을 추가한다. errors='coerce' 옵션은 올바르지 않은 날짜 형식을 Null값으로 채워주는 옵션이다.

```
vald_dt['CRET_TIME']=pd.to_datetime(vald_dt['CRET_TIME'],format='%Y%m%d%H%M',errors='coerce')
```

[코드 12] 분석 도구를 활용하여 데이터 형식 맞춰주기

- 'CRET_TIME'이 Null인 경우를 'isnull()' 메서드를 통해 모든 데이터의 형식이 올바르다는 것을 알 수 있다.

```
vald_dt[vald_dt['CRET_TIME'].isnull()]
```

'Part Number', 'D일06~08(08)H 투입계획(발주) 수량', 'D일08~10(10)H 투입계획(발주) 수량', 'D일10~12(13)H 투입계획(발주) 수량', 'D일13~15(15)H 투입계획(발주) 수량', 'D일15~17(18)H 투입계획(발주) 수량', 'D일18~20(21)H 투입계획(발주) 수량', 'D일21~23(23)H 투입계획(발주) 수량', 'D일23~01(02)H 투입계획(발주) 수량', 'D일 02~04H 투입계획 수량', 'D일 04~06H 투입계획 수량', 'D일 투입예정 수량(D일계획)', 'D+1일 투입예정 수량(D+1일)', 'D+1일 투입예정 수량(D+1일).1', 'D+1일 투입예정 수량(D+1일).2', 'D+1일 투입예정 수량(D+1일).3', 'D+1일 투입예정 수량(D+1일).4', 'D+1일 투입예정 수량(D+1일).5', 'D+1일 투입예정 수량(D+1일).6', 'D+1일 투입예정 수량(D+1일).7', 'D+1일 투입예정 수량(D+1일).8', 'D+1일 투입예정 수량(D+1일).9', 'D+1일 투입예정 수량(Total)', 'D+2일 투입예정 수량(과부족수량)', 'D+2일 투입예정 수량(과부족수량).1', 'D+2일 투입예정 수량(과부족수량).2', 'D+2일 투입예정 수량(과부족수량).3', 'D+2일 투입예정 수량(과부족수량).4', 'D+2일 투입예정 수량(과부족수량).5', 'D+2일 투입예정 수량(과부족수량).6', 'D+2일 투입예정 수량(과부족수량).7', 'D+2일 투입예정 수량(과부족수량).8', 'D+2일 투입예정 수량(과부족수량).9', 'D+2일 투입예정 수량(Total)', 'D+3일 투입예정 수량(과부족수량)', 'D+3일 투입예정 수량(과부족수량).1', 'D+3일 투입예정 수량(과부족수량).2', 'D+3일 투입예정 수량(과부족수량).3', 'D+3일 투입예정 수량(과부족수량).4', 'D+3일 투입예정 수량(과부족수량).5', 'D+3일 투입예정 수량(과부족수량).6', 'D+3일 투입예정 수량(과부족수량).7', 'D+3일 투입예정 수량(과부족수량).8', 'D+3일 투입예정 수량(과부족수량).9', 'D+3일 투입예정 수량(Total)', 'D+4일 투입예정 수량(과부족수량)', 'D+4일 투입예정 수량(과부족수량).1', 'D+4일 투입예정 수량(과부족수량).2', 'D+4일 투입예정 수량(과부족수량).3', 'D+4일 투입예정 수량(과부족수량).4', 'D+4일 투입예정 수량(과부족수량).5', 'D+4일 투입예정 수량(과부족수량).6', 'D+4일 투입예정 수량(과부족수량).7', 'D+4일 투입예정 수량(과부족수량).8', 'D+4일 투입예정 수량(과부족수량).9', 'D+4일 투입예정 수량(Total)', 'D+5일 투입예정 수량', 'D+6일 투입예정 수량', 'D+7일 투입예정 수량', 'D+8일 투입예정 수량', 'D+9일 투입예정 수량', 'D+10일 투입예정 수량', 'D+11일 투입예정 수량', 'D+12일 투입예정 수량', 'D+13일 투입예정 수량', 'D+14일 투입예정 수량', 'D+15일 투입예정 수량', 'D+16일 투입예정 수량', 'D+17일 투입예정 수량', 'D+18일 투입예정 수량', 'D+19일 투입예정 수량', 'D+20일 투입예정 수량', 'D+21일 투입예정 수량', 'D+22일 투입예정 수량', 'D+23일 투입예정 수량', 'D+24일 투입예정 수량', 'D+25일 투입예정 수량', 'D+26일 투입예정 수량', 'D+27일 투입예정 수량', 'D+28일 투입예정 수량', 'D+29일 투입예정 수량', 'D+30일 투입예정 수량', 'D+31~D+45일 투입예정 수량', 'CRET_TIME'

0 rows × 84 columns

[코드 13] 분석 도구를 활용한 데이터 형식 확인하기

- 'Part Number' 열의 데이터 형태가 'Part XX' 형태여야한다. 해당 열의 데이터에 유효하지 않은 형식이 들어있는지 'unique()' 메서드를 이용하여 확인한다. [코드 14]의 결과를 보면 유효성을 위배하는 데이터가 없는 것을 알 수 있다.

```
vald_dt['Part Number'].unique()
```

```
array(['Part 0', 'Part 1', 'Part 2', 'Part 3', 'Part 4', 'Part 5', 'Part 6', 'Part 7', 'Part 8', 'Part 9', 'Part 10', 'Part 11', 'Part 12',
'Part 13', 'Part 14', 'Part 15', 'Part 16', 'Part 17', 'Part 18', 'Part 19', 'Part 20', 'Part 21', 'Part 22', 'Part 23', 'Part 24',
'Part 25', 'Part 26', 'Part 27', 'Part 28', 'Part 29', 'Part 30', 'Part 31', 'Part 32', 'Part 33', 'Part 34', 'Part 35', 'Part 36',
'Part 37', 'Part 38', 'Part 39', 'Part 40', 'Part 41', 'Part 42', 'Part 43', 'Part 44', 'Part 45', 'Part 46', 'Part 47', 'Part 48',
'Part 49', 'Part 50', 'Part 51', 'Part 52', 'Part 53', 'Part 54', 'Part 55', 'Part 56', 'Part 57', 'Part 58', 'Part 59', 'Part 60',
'Part 61', 'Part 62', 'Part 63', 'Part 64', 'Part 65', 'Part 66', 'Part 67', 'Part 68', 'Part 69', 'Part 70', 'Part 71', 'Part 72',
'Part 73', 'Part 74', 'Part 75', 'Part 76', 'Part 77', 'Part 78', 'Part 79', 'Part 80', 'Part 81', 'Part 82', 'Part 83', 'Part 84',
'Part 85', 'Part 86', 'Part 87', 'Part 88', 'Part 89', 'Part 90', 'Part 91', 'Part 92', 'Part 93', 'Part 94', 'Part 95', 'Part 96',
'Part 97', 'Part 98', 'Part 99', 'Part 100', 'Part 101', 'Part 102', 'Part 103', 'Part 104', 'Part 105', 'Part 106', 'Part 107',
'Part 108', 'Part 109', 'Part 110', 'Part 111', 'Part 112', 'Part 113', 'Part 114', 'Part 115', 'Part 116'], dtype=object)
```

[코드 14] 분석 도구를 활용한 데이터 유효성 검사

③ 수집된 날짜 안에 들어가 있는가?

- 2021년 09월 13일부터 2021년 11월 01일에 수집된 데이터이기 때문에, 데이터가 해당 기간을 벗어나지 않았는지 확인한다.

```
import datetime
d0=pd.Timestamp(datetime.date(2021,9,13)) #데이터 수집 시작 날짜
d1=pd.Timestamp(datetime.date(2021,11,2)) #데이터 수집 종료 날짜+1
con1=vald_dt['CRET_TIME']>=d0
con2=vald_dt['CRET_TIME']<=d1
vald_dt=vald_dt[con1&con2]
```

[코드 15] 분석 도구를 활용하여 데이터 수집 기간 확인하기

- 따라서 3가지 경우를 모두 만족하는 데이터는 vald_dt에 저장하고, 최종적으로 완성된 데이터를 이용하여 유효성 품질 지수를 구한다.

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

```
유효성 지수 : 100%
```

[코드 16] 분석 도구를 활용하여 유효성 품질 지수 최종 확인하기

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

- 본 데이터가 구조, 값, 형태를 일관성 있게 지키고 있는지 확인한다. 'Cret_Time' 열의 데이터가 일관된 날짜 형태의 데이터를 가졌는지, 'Part Number' 열의 데이터는 'Part XX'의 형태를 가졌는지는 유효성을 확인하면서 확인하였다. 'dtypes()' 메서드를 이용하여 이외의 발주수량 데이터 값들의 형태가 int64인지 확인한다.

```
data.dtypes
```

```
Part Number                  object
D일06~08(08)H 투입계획(발주) 수량          int64
D일08~10(10)H 투입계획(발주) 수량          int64
D일10~12(13)H 투입계획(발주) 수량          int64
D일13~15(15)H 투입계획(발주) 수량          int64
                    ...
D+28일 투입예정 수량                int64
D+29일 투입예정 수량                int64
D+30일 투입예정 수량                int64
D+31~D+45일 투입예정 수량              int64
CRET_TIME              datetime64[ns]
Length: 84, dtype: object
```

[코드 16] 분석 도구를 활용하여 유효성 품질 지수 최종 확인하기

- 본 데이터는 일관성이 있다고 할 수 있다.

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

- 'Part Number', 'D일06~08(08)H 투입계획(발주) 수량', 'D일08~10(10)H 투입계획(발주) 수량', 'D일10~12(13)H 투입계획(발주) 수량', 'D일13~15(15)H 투입계획(발주) 수량', 'D일15~17(18)H 투입계획(발주) 수량', 'D일18~20(21)H 투입계획(발주) 수량', 'D일21~23(23)H 투입계획(발주) 수량', 'D일23~01(02)H 투입계획(발주) 수량', 'D일02~04H 투입계획 수량', 'D일 04~06H 투입계획 수량'의 합이 'D일 투입예정 수량(D일계획)'과 같아야 한다. 본 데이터는 하루에 평균적으로 3-4회 당일 발주량에 대한 로그가 수집된다. 실제로는 매일 가장 마지막 시간의 발주 로그가 해당일의 전체 발주 정보를 담고 있기 때문에, 정확성 품질 지수는 매일 마지막 시간의 발주 로그 데이터로 평가한다.

```
# 매일 마지막 시간의 발주 로그 데이터를 valid_dt에 저장한다.
valid_dt=data
valid_dt['CRET_TIME'] = pd.to_datetime(valid_dt['CRET_TIME'], format="%Y%m%d%H%M")
valid_dt['CRET_TIME'] = pd.to_datetime(valid_dt['CRET_TIME'], format="%Y%m%d%H%M")
valid_dt = valid_dt.groupby(by=[valid_dt['CRET_TIME'].dt.year,
                                valid_dt['CRET_TIME'].dt.month,
                                valid_dt['CRET_TIME'].dt.day]).last()
valid_dt.reset_index(drop=True, inplace=True)

data1=valid_dt.loc[:,['Part Number', 'D일06~08(08)H 투입계획(발주) 수량',
       'D일08~10(10)H 투입계획(발주) 수량', 'D일10~12(13)H 투입계획(발주) 수량', 'D일13~15(15)H 투입계획(발주) 수량', 'D일15~17(18)H 투입계획(발주) 수량', 'D일18~20(21)H 투입계획(발주) 수량', 'D일21~23(23)H 투입계획(발주) 수량', 'D일23~01(02)H 투입계획(발주) 수량', 'D일 02~04H 투입계획 수량', 'D일 04~06H 투입계획 수량' ]]
data1['sum']=data1.sum(axis=1)
(data1['sum']-valid_dt['D일 투입예정 수량(D일계획)']).sum()
```

```
0
```

[코드 18] 분석 도구를 활용한 정확성 품질 지수 확인하기

- 합해야 할 컬럼들을 data1에 저장을 한 뒤, sum(axis=1)을 이용하여 각 열의 합을 구하고, 본 데이터의 'D일 투입예정 수량(D일계획)'과의 차이를 구한다. 값의 차이의 합은 0으로, 정확성을 위배하는 데이터는 없다고 할 수 있다.

▶ 무결성 품질 지수 = (1-(유일성, 유효성, 일관성 지수 중 100%가 아닌 지수 개수/3))*100

- 본 데이터에서 유일성은 98.24%, 유효성과 일관성은 100%로, 세 가지 지수 중 유일성 지수가 100%를 만족하지 못하므로 무결성 품질 지수는 은 66.67%이다.

▶ 가중치지수 = 품질 지수 * 가중치

- 데이터의 특성에 따라 품질 지수의 중요도는 달라진다. 활용되는 데이터의 중요한 품질 지수가 무엇인지를 평가하고, 데이터의 특성에 따라 적절한 가중치를 부여한다.

▶ 데이터 품질 지수 = Σ 가중치지수

- 최종적으로 구한 6가지 데이터 품질 지수에 적절한 가중치를 부여하여 품질 지수를 구한다. 품질 지수 가중치는 활용되는 데이터의 특성에 따라 부여할 수 있다. 중요한 품질 지수가 무엇인지를 평가하고, 데이터의 특성에 따라 가중치를 부여한다. 본 데이터는 발주량에 대한 로그 데이터라는 특성상 하루에 평균적으로 3~4회 주문이 기록된다. 즉, 유일성에 위반하는 데이터가 존재할 수 있다는 특성이 있기에 유일성의 가중치와 품질 지수 산출 과정에 유일성을 고려하는 무결성의 가중치를 10%로 부여하여 상대적으로 적게 가중치를 둔다. 이 외에 완전성, 유효성, 일관성, 정확성은 동일하게 중요하다는 가정하에 가중치를 동일하게 각각 20%로 부여하였다.

- 제조 데이터 품질 지수 결과는 아래와 같다. 가중치를 부여하지 않고 6가지 품질 지수(완전성, 유일성, 유효성, 일관성, 정확성, 무결성)를 평균을 낸 품질 지수는 94.15%다. 가중치 지수는 각각의 품질 지수에 가중치를 적용한 값을 나타내며, 가중치를 적용한 품질 지수는 96.49%로, 가중치를 적용하지 않은 품질 지수 94.15%보다 2.34% 높은 값이 나왔다. 오류율 역시 5.85%에서 3.51%로 줄어들었다.

구분	품질 지수	가중치	가중치지수	오류율
완전성	100%	20%	20%	0%
유일성	98.24%	10%	9.82%	1.76%
유효성	100%	20%	20%	0%
일관성	100%	20%	20%	0%
정확성	100%	20%	20%	0%
무결성	66.67%	10%	6.67%	33.33%
품질 지수	94.15%	100%	96.49%	5.85%(3.51%)

※ 제출 시 유의사항

1) 제안서와 발표자료는 반드시 각각 1개의 단일파일(PDF)이어야 합니다.(용량이 크더라도 파일분할 금지)

2) 전체적으로 시각자료를 최대한 많이 활용하여 초보자(중소 제조기업/대학생)도 이해할 수 있는 가이드북 제작을 요망합니다.

3) 본 실습에 활용되는 데이터셋이 기업에 민감한 핵심 레시피 데이터인 경우, 꼭 비식별화하여야 합니다. 이에 대한 법적 책임은 AI 데이터셋 과제 수행기관에 있음을 명확히 합니다.

「공급망 최적화 AI 데이터셋」 분석실습 가이드북

중소벤처기업부 KOSMO 스마트제조혁신추진단 KAIST

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