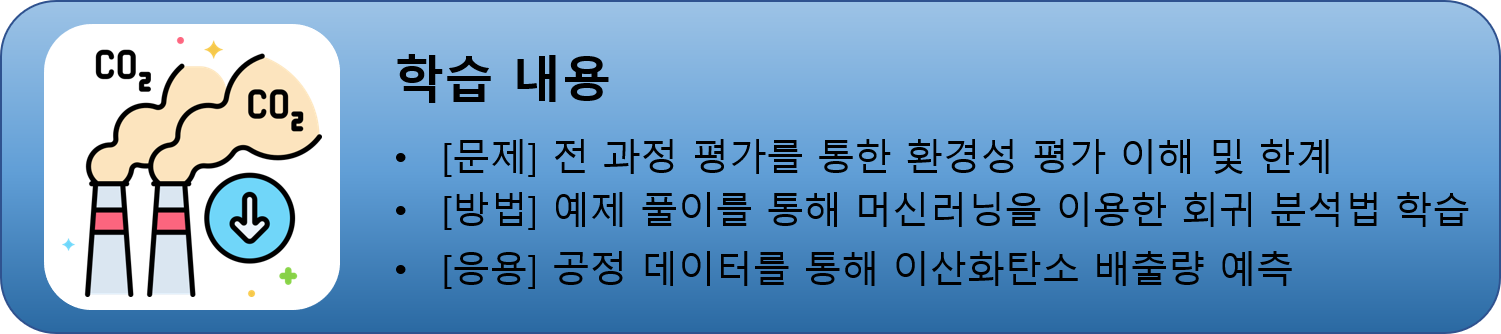
1. 화공산업에서의 인공지능

# **인공지능 기반 공정 설계 및 최적화**

### **바이오공정 전 과정 평가**

|  |  |
| --- | --- |
| **학습 내용** | |
| [문제] | 전 과정 평가를 통한 환경성 분석 이해 |
| [방법] | 머신러닝을 이용한 전 과정 평가 |
| [응용] | 공정 데이터를 통한 이산화탄소 배출량 예측 |
| [요약] | * 전과정 평가의 방법론 및 중요성 이해 * 전과정 평가의 머신러닝 접목의 필요성 * 머신러닝을 이용해 제지 공장의 이산화탄소 배출량 예측 |



### **[이론] 전 과정 평가**

전 과정 평가(life-cycle assessment)는 제품의 생산, 사용, 폐기, 전 과정에 걸쳐 수지(투입물질, 에너지, 배출물질)와 해당 과정에서 미치는 환경 영향을 정량화하는 환경경영기법을 말한다. 전 과정 평가는 공정의 환경성을 진단함으로써 환경라벨링 인증 획득이나 친환경제품 개발 등에서 의사결정을 하는 근거로 활용된다. 특히 화공산업은 대기, 수질, 토양 등 다양한 분야에 넓게 연계되어 있으므로, 지속가능한 산업 발전을 위해서도 환경성 평가는 매우 중요하다.

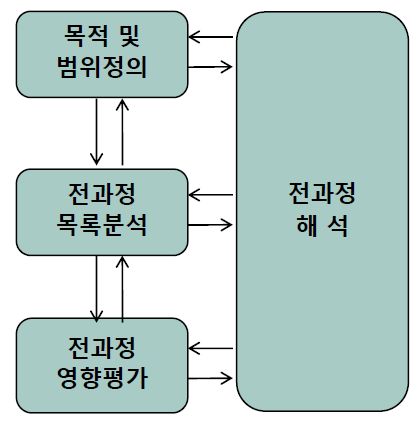


그림 1. 전 과정 평가 방법론 개요 (국제 표준화 기구 제공)

그림 1에 보인 바와 같이, 전 과정 평가는 목적 및 범위 정의, 전 과정 목록 분석, 전 과정 영향 평가, 전 과정 해석의 4단계로 구성된다.

### **목적 및 범위 정의(goal and scope definition)**

목적 및 범위 정의 단계에서는 전 과정 평가를 수행하고자 하는 목적과 대상 (제품 또는 서비스), 데이터 수집범위와 수집방법, 결과형태 및 활용방법을 결정한다. 이 단계에서 전 과정 평가의 기준이 되는 ‘기능적 단위(functional unit)’와 전 과정 평가 수행의 범위는 ‘시스템 경계(system boundary)’를 설정한다.

### **전 과정 목록 분석(life-cycle inventory analysis)**

전 과정 목록 분석은 목적 및 범위 정의 단계에서 설정한 시스템 경계에 대하여 물질수지 데이터를 수집하고, 기능적 단위를 기준으로 정량화하는 단계이다.

### **전 과정 영향 평가(life-cycle impact assessment)**

전 과정 영향 평가는 전 과정 목록 분석을 바탕으로 환경에 미치는 영향을 종합적으로산출하는 단계이다. 각 평가 항목을 산출하는 방법론에는 IPCC, ReCiPe 등이 존재한다.

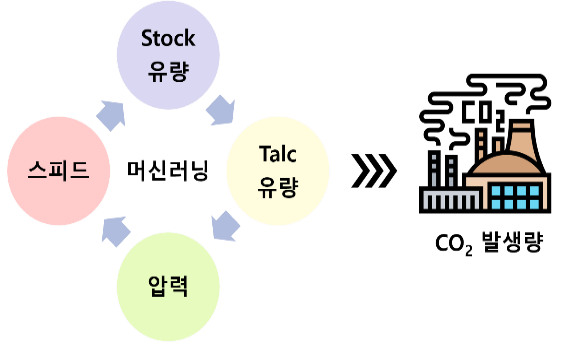
### **전 과정 해석(life-cycle interpretation)**

전 과정 해석은 전 과정 영향 평가 결과를 바탕으로 각각의 평가 항목에 중요한 영향을 미치는 환경인자를 분석하는 단계이다.

본 장에서는 머신 러닝 기반의 회귀분석을 전 과정 영향 평가에 적용하여, 이산화탄소 배출량 예측 모델을 제작함으로써, 공정 변수의 변화에 따른 정확한 온실가스 배출량 산정을 목적으로 한다.

### **[문제]**

**엑셀파일 paperdata.xlsx는 어느 공장의 샘플 운전데이터를 나타낸다. 해당 파일의 각 열은 차례로 stock 유량, talc 유량, 압력, 속도, 이산화탄소 발생량을 나타낸다. stock 유량, talc 유량, 압력, 속도는 독립변수이자 예측변수이며, 이산화탄소 발생량은 출력변수이다. 운전 데이터의 40%를 홀드아웃 검증에 사용한다고 할 때, SVM 회귀모델을 통해 새로운 운전데이터(newdata.xlsx)의 이산화탄소 발생량을 예측하여라.**

****

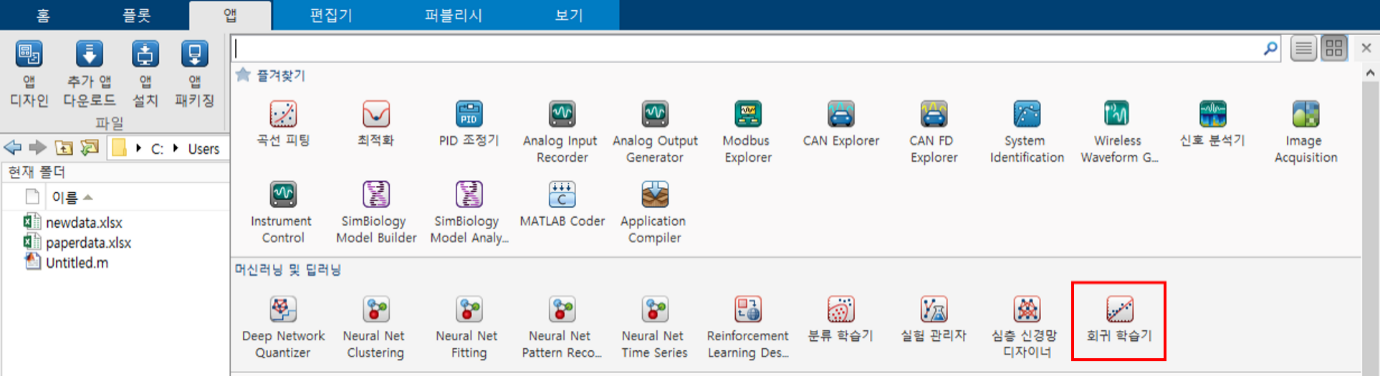
**[방법] 머신 러닝을 통한 회귀분석법**

#### 매트랩에서 SVM회귀분석을 하기 위한 명령어는 무엇이 있으며, 이들의 차이는 무엇인가?

1. SVM 회귀분석을 위한 명령어는 ‘fitrsvm’과 fitrlinear’가 있다. ‘fitsvm’은 비교적 크기가 작은 차원의 데이터 집합들에 대한 SVM 회귀모델을 생성하는 반면, ‘fitrlinear’는 비교적 크기가 큰 차원의 데이터 집합들에 대한 SVM 선형회귀 모델을 형성한다.

#### 명령어를 직접 입력하지 않고, 매트랩 내부의 앱을 통해 회귀분석을 하기 위해서는 어떻게 해야 하는가?

1. A그림 5와 같이 리본 메뉴의 ‘앱’ 탭에서 앱들 중 [머신러닝 및 딥러닝]에서 ‘회귀 학습기’를 선택한다.

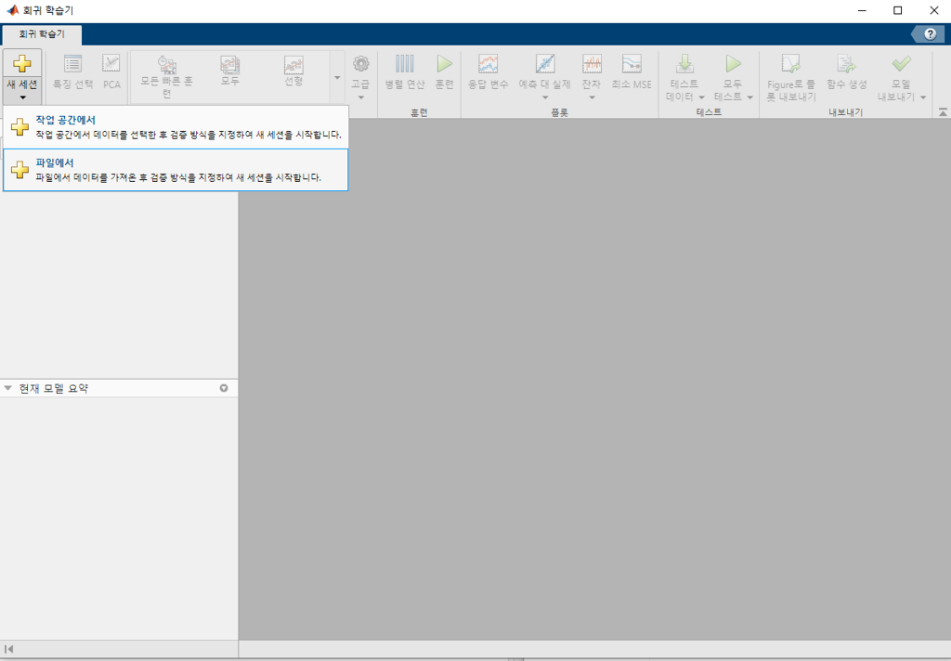


**그림 5. 회귀 분석을 위한 회귀 학습기**

**[응용] 머신 러닝을 통한 이산화탄소 배출량 예측**

#### 매트랩의 회귀 학습기를 사용해 샘플 운전데이터(paperdata.xlsx)를 입력하여라.

1. 그림 6~8과 같은 절차를 통해 샘플 운전데이터를 입력할 수 있다.



**그림 6. 샘플 운전데이터 불러오기**

상단 리본 메뉴에서 ‘새 세션’을 클릭한 후 ‘파일에서’를 선택한다. 그리고 ‘paperdata.xlsx’를 선택한다.

테이블이(가) 표시된 사진

자동 생성된 설명

**그림 7. 불러온 샘플 운전데이터의 항목 설정**

상단 리본 메뉴의 ‘선택 범위/내역’에 필요한 데이터가 모두 선택되었는지 확인한 후, 리본 메뉴 우측의 ‘선택 항목 가져오기’를 클릭한다.

테이블이(가) 표시된 사진

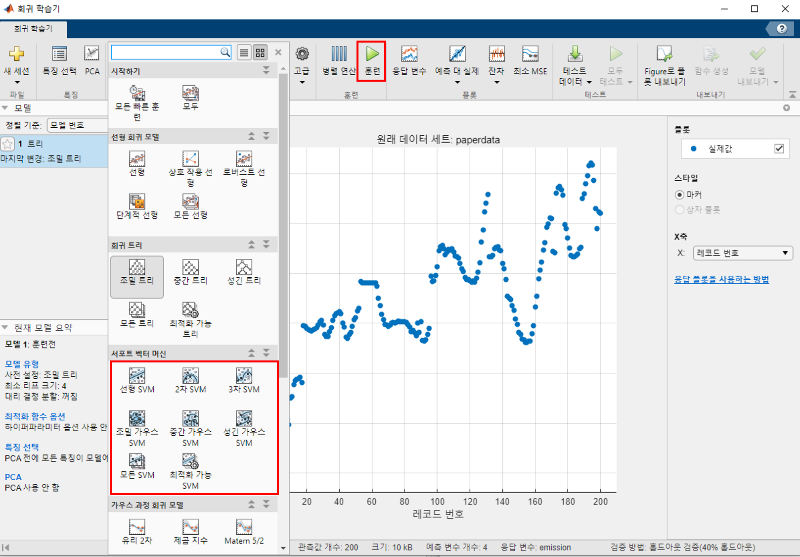
자동 생성된 설명

**그림 8. 응답변수 및 검증 방법 설정**

응답변수에 이산화탄소 발생량이 설정된 것을 확인한다. 그 후 검증 방법에서 ‘홀드아웃 검증’을 선택하고, ‘홀드아웃 비율’에는 40을 입력한 후 ‘세션 시작’을 클릭한다.

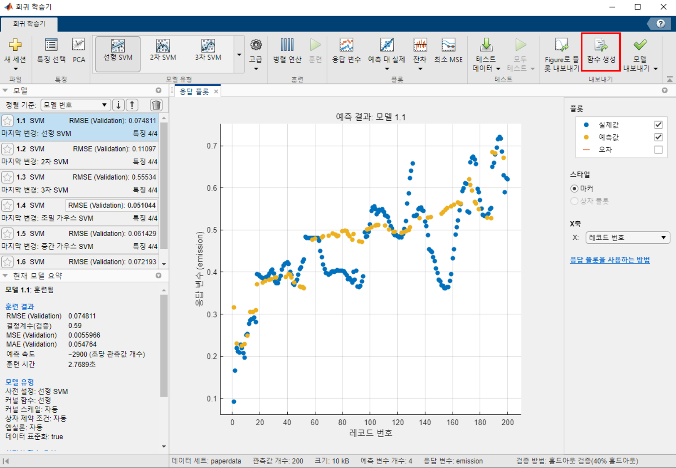
#### 입력한 샘플 운전데이터를 이용해 SVM 회귀 모델을 학습하여 이산화탄소 발생량 예측 함수를 생성하여라.

1. SVM 회귀 모델을 학습 및 이산화탄소 발생량 예측 함수를 생성은 아래의 그림9와 그림 10과 같은 절차를 통해 가능하다.



**그림 9. SVM 회귀 모델 설정 및 훈련**

모델 유형에서 SVM 모델을 선택한다. 특정 SVM 모델을 선택할 수도 있고, ‘모든 SVM’을 통해 모든 종류의 SVM을 동시에 학습시킬 수 있다. (이번 예제에서는 ‘모든 SVM’을 선택하였다.) 모델을 선택한 후, 상단의 훈련 버튼을 클릭하면 SVM 회귀 분석 모델이 생성된다.

 텍스트이(가) 표시된 사진

자동 생성된 설명

**그림 10. 훈련된 모델의 함수 생성**

리본 메뉴에서 ‘함수 생성’을 클릭하면 오른쪽 그림과 같이 코드가 생성된 것을 확인할 수 있다.

#### 생성한 SVM 회귀 함수를 이용해 새로운 운전데이터(newdata.xlsx)의 이산화탄소 발생량을 예측하여라.

1. 그림 11~13과 같은 절차를 통해 새로운 운전데이터의 이산화탄소 발생량을 예측할 수 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명

**그림 11. 샘플 운전데이터 지정 및 범위 입력**

샘플 운전데이터를 불러와 데이터 범위를 지정하여 trainingData로 입력한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

**그림 12. 생성된 모델을 이용한 새로운 운전데이터의 이산화탄소 발생량 예측**

샘플 운전데이터와 마찬가지로 새로운 운전데이터를 불러와 범위를 지정한다(T로 설정). 이를 통해, 샘플 운전데이터로 학습한 모델을 통해 새로운 운전데이터의 이산화탄소 발생량을 예측한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

**그림 13. 모델을 통해 예측한 새로운 운전데이터의 이산화탄소 발생량**

### **[결론]**

전 과정 평가는 공정의 환경성을 판단할 수 있는 분석 방법 중 하나로, 기후 위기가 심각해짐에 따라 그 중요성이 더욱 강조되고 있다. 본 장에서는 전 과정 평가와 머신 러닝을 접목함으로써 효과적으로 이산화탄소의 발생량을 예측하였다. 주어진 예제에서는 계산의 용이성을 위해, 예측에 200개의 샘플을 이용하였으나, 더 많은 샘플을 이용할 경우 정확도를 크게 향상시킬 수 있다. 또한, 전 과정 평가에서 이산화탄소 배출량 뿐만 아니라 연료 소모량, 오존 발생량 등 다양한 환경 지표를 함께 산정하였다.

### **[학습 결과]**

* 학습 내용

환경성 평가를 위한 전 과정 평가의 방법론 및 이해

* 학습 결과 확인하기

머신러닝을 이용한 회귀 분석 절차 및 방법을 익히기

* 학습 결과 응용하기

본 장의 학습내용에 기반해 회귀분석함으로써 새로운 공정에서 발생하는 이산화탄소 배출량을 예측

### **탈 실험 단원자 증착 공정 설계**

### **단원자 증착법**

증착 공정 (Deposition process)은 반도체 소자 재조를 위한 단위공정들 중 하나로 반도체 산업에 핵심적인 기술이다. 보통 증착 공정은 단결정의 규소기판에 반도체, 절연체, 도체 등의 재료를 박막의 형태로 증착 시킴으로써, 집적회로의 미세구조 형성에 사용된다. 일반적으로 증착 공정은 기판 위 회로 규모에 가장 큰 영향을 주므로, 소자의 집적도에 가장 큰 기여를 한다. 따라서 고집적 박막 형성을 위한 효율적이고 섬세한 증착 공정 개발이 차세대 반도체 개발을 위해 핵심적이다.

기존의 반도체 공정에서의 증착법은 전통적으로 물리기상증착법 (Physical vapor deposition, PVD)과 화학기상증착법 (Chemical vapor deposition, CVD)에 의존한다. PVD는 도포에 사용할 물질들을 높은 에너지나 열에 노출시켜 입자를 기판 표면에 방사하여 증착하는 방식이다. 이 기술은 높은 생산성을 갖고 준수한 박막 두께 조절력을 갖는다. CVD는 기질 표면에서 반응기체를 증착시키는 방법으로, 박막 두께 조절에 우수하고 적절한 생산성을 갖는다. 하지만 CVD와 PVD는 모두 200 Å 이하의 박막 증착에는 사용할 수 없는데, 반도체 박막 증착 두께가 높은 집적도의 소자 개발에 필수적임으로 새로운 기술이 요구되는 실정이다. 단원자 증착법 (Atomic layer deposition, ALD)은 소자 미세화와 집적도 향상을 위한 가장 이상적인 무기 및 금속 박막 증착 기술이다. ALD은 원자 단위에서 단원자층을 증착하는 기술로써 200 Å 이하의 박막 형성이 가능하여 고용량 반도체 소자 개발에 핵심적이다. 또한, 3차원 구조의 반도체 제조 공정에 응용할 수 있어 반도체 생산에 큰 각광을 받고 있다.

ALD은 기판과의 전구체 물질 (Precursor) 사이의 선택적 표면 화학반응을 통한 반응물 분자의 증착을 유도한다. 이를 위해 밀도범함수 이론 (Density functional theory, DFT), kinetic Monte Carlo 방법 등이 기판-전구체 사이의 반응 예측과 해석에 이용된다. 하지만, 수많은 전구체 물질, 반응물, 그리고 기판 사이의 현상 실험은 비용과 시간의 한계점이 명확히 존재한다. 따라서 비효율적인 단원자 증착법을 탈피해 실용화를 위한 고성능 물질 탐색을 위해 혁신이 요구된다.

서포트 벡터 회귀 (Support vector regression)은 고차원 데이터 식별을 위한 서포트 벡터 머신 (Support vector machine) 으로부터 발달한 인공지능 알고리즘이다. 서포트 벡터 회귀는 단순한 하이퍼파라미터 구조를 통해 신속한 모델 생성이 가능한 특징을 지녔다. 오늘날 서포트 벡터 회귀는 다양한 모델과 결합된 hybrid 형태로 화학공정의 많은 복잡계 문제를 해결하고 있다.

본 실습에서는 탈실험화된 단원자 증착 공정 설계를 위해 서포트벡터머신 기반의 전구체 성능 예측 모델을 개발한다.

### **[문제]**

**고성능 반도체 소재 개발에 핵심적인 단원자 증착 과정 제어를 위해 서포트 벡터 머신을 활용해 증착 두께 예측 모델을 개발하라.**

**- “코드 및 데이터/4-3. ALD.csv” 데이터를 활용하여라.**

**- 서포트 벡터 머신의 하이퍼파라미터를 이해하고 정확도 개선을 위한 하이퍼파라미터 튜닝을 시도하라.**

### **[방법] SVM 예제**

#### SVM을 활용하기 위한 라이브러리를 조사하라. SVM을 regression 형태로 사용하기위한 명령어는 무엇인가?

1. SVM 활용을 위한 라이브러리로 ‘e1071’이 있다. SVM은 input데이터 형식에 따라 자동으로 classification/regression을 설정한다. 따라서 ‘svm()’함수를 불러오고, 데이터를 독립/종속변수를 설정한다.

|  |
| --- |
| Library(‘e1071’)  Model\_SVM <- e1071::svm(Data[,:1:x], Data[,x], gamma = , cost = ) |

#### SVM의 학습은 하이퍼파라미터에 따라 다른 성능의 예측모델을 생성한다. SVM의 하이퍼파라미터는 무엇인가? 하이퍼파라미터가 의미하는 것은 무엇인가?

1. SVM의 하이퍼파라미터에는 ‘gmma’와 ‘cost’가 있다. ‘gamma’는 데이터 샘플의 영향력 행사 거리 (분산)을 가정한다. ‘gamma’ 가 클수록 데이터가 다른 데이터에 더 큰 영향을 줄 수 있다고 판단한다. ‘cost’는 데이터 샘플의 최대/최소 허용 클래스 수를 가정한다. ‘cost’가 크면 이상치의 가능성을 매우 작게 판단한다.

### **[응용] SVM 기반 고성능 전구체 물질 예측 모델 개발**

예제는 R 4. 0. 2 프로그래밍 언어를 기준으로 Rstudio 개발환경에서 작성되었다. 예제 실습을 위해 Anaconda 프로그램을 아래 URL로부터 다운로드할 수 있다.

https://www.anaconda.com/

#### 데이터를 Jupyter환경으로부터 불러오고 데이터를 확인하라.

1. 다음과 같은 code를 사용하여 불러올 수 있다.

|  |
| --- |
| DFT\_data <- read.csv(file.choose(), header = T) |

‘read.csv’ 함수를 사용하여 데이터 파일이 저장된 장소를 직접 찾아 ‘DFT\_data’ 이름으로 데이터를 불러온다. 해당 데이터는 column 이름이 이미 존재하는 데이터로 ‘header = T’ arg를 통해 이를 밝힌다.

|  |
| --- |
| str(DFT\_data) |

‘str()’ 함수를 사용하여 데이터의 형태를 확인할 수 있다.

|  |
| --- |
| DFT\_data |

테이블이(가) 표시된 사진

자동 생성된 설명

불러온 데이터를 확인해본다.

#### 인공지능 학습을 위해서 데이터의 단위를 무시한 상대적 영향력을 파악할 필요가 있다. DFT데이터를 정규화하라.

1. 다음과 같은 절차를 통해 정규화한다.

|  |
| --- |
| DFT\_wo\_NA <- data.frame(DFT\_data[,1:2], DFT\_data[,4:28]) |

데이터의 3번째 열은 전구체 물질의 이름이므로 이를 제거한다. 추출된 데이터는 ‘DFT\_wo\_NA’로 명명한다.

|  |
| --- |
| DFT\_wo\_NA\_symbol <- data.frame(DFT\_wo\_NA)  DFT\_wo\_NA\_symbol\_scale <- data.frame(scale(DFT\_wo\_NA\_symbol)) |

데이터를 ‘DFT\_wo\_NA’로 명명하고 ‘scale()’ 함수를 사용해서 정규화한다. 정규화된 데이터는 ‘DFT\_wo\_NA\_symbol\_scale’로 명명한다

#### 9개의 종속변수 각각 예측하는 SVM 모델을 생성할 것이다. 데이터를 이에 맞춰 나누어라.

1. 다음과 같은 절차를 통해 데이터를 나눌 수 있다.

|  |
| --- |
| DFT\_EadH <- data.frame(DFT\_wo\_NA\_symbol\_scale[,1:9], DFT\_wo\_NA\_symbol\_scale[,10])  DFT\_EadOH <- data.frame(DFT\_wo\_NA\_symbol\_scale[,1:9], DFT\_wo\_NA\_symbol\_scale[,11])  DFT\_EadNH2 <- data.frame(DFT\_wo\_NA\_symbol\_scale[,1:9], DFT\_wo\_NA\_symbol\_scale[,12]) |

데이터의 종속변수인 3가지 표면에서의 adsorption 에너지 (Ead\_H, Ead\_OH, Ead\_NH2)에 대해 각각 독립변수-종속변수 관계를 규명한 데이터를 생성한다

|  |
| --- |
| DFT\_EaH <- data.frame(DFT\_wo\_NA\_symbol\_scale[,1:9], DFT\_wo\_NA\_symbol\_scale[,13])  DFT\_EaOH <- data.frame(DFT\_wo\_NA\_symbol\_scale[,1:9], DFT\_wo\_NA\_symbol\_scale[,14])  DFT\_EaNH2 <- data.frame(DFT\_wo\_NA\_symbol\_scale[,1:9], DFT\_wo\_NA\_symbol\_scale[,15]) |

데이터의 종속변수인 3가지 표면에서의 active barrier 에너지 (Ea\_H, Ea\_OH, Ea\_NH2)에 대해 각각 독립변수-종속변수 관계를 규명한 데이터를 생성한다

|  |
| --- |
| DFT\_dEH <- data.frame(DFT\_wo\_NA\_symbol\_scale[,1:9], DFT\_wo\_NA\_symbol\_scale[,16])  DFT\_dEOH <- data.frame(DFT\_wo\_NA\_symbol\_scale[,1:9], DFT\_wo\_NA\_symbol\_scale[,17])  DFT\_dENH2 <- data.frame(DFT\_wo\_NA\_symbol\_scale[,1:9], DFT\_wo\_NA\_symbol\_scale[,18]) |

데이터의 종속변수인 3가지 표면에서의 dE (dE\_H, dE\_OH, dE\_NH2)에 대해 각각 독립변수-종속변수 관계를 규명한 데이터를 생성한다

#### SVM 학습을 위해 데이터를 학습/검증 데이터로 나누어야 한다. 학습/검증 데이터를 각각 70%와 30%로 나누어라.

1. 다음과 같은 절차를 통해 데이터를 나눌 수 있다.

|  |
| --- |
| smp\_size <- floor(0.7 \* nrow(DFT\_EadH))  set.seed(1)  train\_ind <- sample(seq\_len(nrow(DFT\_wo\_NA)), size = smp\_size) |

‘floor()’ 함수를 사용하여 ‘WGSR\_scale’ 데이터의 행 개수의 70%에 해당되는 값을 생성한다. 이 값은 ‘smp\_size’로 명명하여, 후에 무작위 데이터 추출에 사용한다.

‘sample()’ 함수를 사용해서 ‘WGSR\_scale’ 데이터로부터 행 번호를 무작위 추출을 진행한다. ‘size’ arg를 ‘smp\_size’로 설정하여 총 70% 데이터를 추출한다. ‘seq\_len()’ 함수를 통해 1부터 임의의 지정된 숫자까지 순차 데이터를 생성할 수 있다.

|  |
| --- |
| SVM\_train <- DFT\_EadH[train\_ind,]  SVM\_test <- DFT\_EadH[-train\_ind,]  SVM\_train <- DFT\_EadOH[train\_ind,]  SVM\_test <- DFT\_EadOH[-train\_ind,]  SVM\_train <- DFT\_EadNH2[train\_ind,]  SVM\_test <- DFT\_EadNH2[-train\_ind,] |

SVR의 학습과 검증을 위해 세가지 표면에서의 adsorption 에너지 대한 데이터를 생성한다.

|  |
| --- |
| SVM\_train <- DFT\_EaH[train\_ind,]  SVM\_test <- DFT\_EaH[-train\_ind,]  SVM\_train <- DFT\_EaOH[train\_ind,]  SVM\_test <- DFT\_EaOH[-train\_ind,]  SVM\_train <- DFT\_EaNH2[train\_ind,]  SVM\_test <- DFT\_EaNH2[-train\_ind,] |

SVR의 학습과 검증을 위해 세가지 표면에서의 active barrier 에너지 대한 데이터를 생성한다.

|  |
| --- |
| SVM\_train <- DFT\_dEH[train\_ind,]  SVM\_test <- DFT\_dEH[-train\_ind,]  SVM\_train <- DFT\_dEOH[train\_ind,]  SVM\_test <- DFT\_dEOH[-train\_ind,]  SVM\_train <- DFT\_dENH2[train\_ind,]  SVM\_test <- DFT\_dENH2[-train\_ind,] |

SVR의 학습과 검증을 위해 세가지 표면에서의 dE 대한 데이터를 생성한다.

#### 위에서 만든 데이터를 사용하여 SVM 모델을 생성할 것이다. Gamma = 0.1, cost = 10 환경에서 SVM 모델을 생성하라.

1. 다음과 같은 절차를 통해 SVM을 생성할 수 있다.

|  |
| --- |
| library(e1071)  library(Metrics) |

SVR 모델을 내장하고 있는 ‘e1071’ package를 import한다.

SVR 모델의 검증을 예측 정확도 평가 지표를 내장하고 있는 ‘Metrics’ package를 import 한다.

|  |
| --- |
| model\_SVM <- e1071::svm(SVM\_train[,1:9], SVM\_train[,10],  gamma = 0.001, cost = 60) |

SVR의 학습을 진행한다. Gamma와 cost는 ‘svm()’ 함수의 ‘arg’로, 하이퍼파라미터에 해당한다. 이는 여러 번의 시행착오를 통해 얻도록 한다.

|  |
| --- |
| summary(model\_SVM) |

텍스트이(가) 표시된 사진

자동 생성된 설명

‘summary()’ 함수를 통해 모델의 정보를 얻을 수 있다.

#### 모델의 예측 정확도를 파악하라. 예측 정확도는 R2, MSE, RMSE, 그리고 MAE 지표를 사용해 판단하라.

1. 다음과 같은 절차를 통해 정확도를 평가할 수 있다.

|  |
| --- |
| pred\_train <- predict(model\_SVM, SVM\_train[,1:9])  pred\_test <- predict(model\_SVM, SVM\_test[,1:9]) |

SVR의 모델을 통해 학습 데이터와 검증 데이터를 예측한다.

|  |
| --- |
| rsq <- function (x, y) cor(x, y) ^ 2 |

예측 정확도 평가를 위한 R2 계산식을 만든다. 해당 함수는 ‘rsq()’로 명명한다.

|  |
| --- |
| rsq(pred\_train,SVM\_train[,10])  Metrics::mse(pred\_train,SVM\_train[,10])  Metrics::rmse(pred\_train,SVM\_train[,10])  Metrics::mae(pred\_train,SVM\_train[,10])  rsq(pred\_test,SVM\_test[,10])  Metrics::mse(pred\_test,SVM\_test[,10])  Metrics::rmse(pred\_test,SVM\_test[,10])  Metrics::mae(pred\_test,SVM\_test[,10]) |

학습 데이터 예측에 대한 정확도와 검증 데이터 예측에 대한 정확도를 계산한다.

|  |
| --- |
| plot(pred\_train, SVM\_train[,10])  plot(pred\_test, SVM\_test[,10]) |

예측 값과 실제 값을 시각화 한다.

### **[결론]**

본 장에서는 증착 두께 제어가능한 단원자 증착법을 위해 서포트 벡터 머신을 활용하여 증착 두께 예측 모델을 개발하였다. 이를 위해 데이터를 표준화하고 예측 모델 학습을 위한 데이터 구조를 설계하였다. 예측 정확도 평가를 위해 통계적 정확도 지표를 사용과 정확도 개선을 위한 하이퍼파라미터 조사를 학습하였다. 결과로, 서포트 벡터 머신을 활용한 증착 두께 예측 모델 개발 방법을 학습할 수 있었다.

### **[학습 결과]**

* 학습 내용

반도체 소재 개발을 위한 SVM 방법론 기반 단원자 증착 설계 익히기

* 학습 결과 확인하기

SVM 알고리즘의 활용 방법 및 예측 결과 해석 방법 익히기

* 학습 결과 응용하기

본 장의 학습내용에 기반해 단원자 증착을 위한 전구체 물질 탐색 및 효과적 표면 화학반응 설계를 위한 전구체 물질 성능 예측하기

### **접착용 에폭시 고분자 개발**

|  |  |
| --- | --- |
| **학습 내용** | |
| [문제] | 에폭시 최적 생산조건 탐색 |
| [방법] | 접착용 에폭시 생산 모사 함수 구축 |
| [응용] | 베이지안 최적화 기반 최적 생산 조건 도출 |
| [요약] | * 접착용 에폭시 생산 반응 이해 * 생산 모사 함수 구축 * 베이지안 최적화를 사용하여 최적의 생산 조건 도출 |

### **[이론] - 능동학습을 사용한 고분자 생산조건 최적화.**

2010년대에 들어서부터 능동학습 프레임워크를 활용한 기능성 신물질 탐색이 점점 각광을 받고 있다. 기계학습을 통한 능동학습(active learning)의 프레임 워크의 하에서의 의사결정 과정의 장점이 기존의 의사결정 과정들과 비교해서 여러 장점이 있기 때문이다. 인간의 의사결정보다는 의사결정과정이 더욱 정교하기 때문에 탐험-탐사(exploration-exploitation)사이의 균형을 고차원의 의사결정에서도 잘 맞출 수 있고, 지도학습 기반 의사결정 보다는 필요한 초기 데이터의 수가 훨씬 적고 또한, 데이터 생성이 시간이 많이 걸리고 비용이 드는 신물질 개발에서 능동학습은 점점 중요도가 올라가고 있다.

능동학습 기반 의사과정은 다음과 같은 세가지 과정들을 거친다. 첫째로는, 현재 존재하는 소수의 데이터를 바탕으로 모델을 구축하여 input-output 사이의 상관관계를 기계가 학습을 한다 (modeling). 둘째로는 기계가 직접 다음 실험 지점을 추천한다 (proposal). 이때 추천이 되는 점은 모델에서의 이해를 바탕으로 우리가 원하는 물성이 최대화가 될 수 있는 지점이면서, 또한 탐색되지 않은 지점의 정보를 얻어낼 수 있는, 즉 탐험-탐사 간의 균형이 맞은 지점이다. 셋째로는 추천된 실험 후보를 사람이 직접 실험을 하여 얻어진 결과를 확인하는 것 이다. 이때 얻어진 결과가 목적에 부합하는지 여부에 따라서 실험이 종결되거나 지속된다.

탐험-탐사간의 균형이 맞은 의사결정을 내리는 방법론중에서 가장 대표적인 방법은 베이지안 최적화이다. 따라서 베이지안 최적화를 접착용 에폭시 개발 문제에 직접 적용하는 것이 이번 실습의 목적이다. 베이지안 최적화의 이론 관한 자세한 설명은 챕터 8, 기계학습과 최적화에 서술되어있다.

베이지안 최적화를 실제 실험 적용해 보면 가장 좋겠지만, 교과서적인 한계로 인해 고분자를 실제로 합성해 나가며 물질 개발을 할 수 없기에, 접착용 에폭시 고분자의 input – output의 상관관계를 Sum-squares function이라는 기준 함수(benchmark function)으로 정의한다. 이때, 함수를 모른다고 가정한 뒤, 초기의 특정한 실험 데이터 (spot data)만을 시작으로 베이지안 최적화 기반의 의사결정을 해 나가면서 의사결정이 효율적으로 이루어짐을 확인한다. 효율성을 보이기 위해, 베이지안 최적화를 진행한 결과와, 불확실성에 대한 고려 없는 greedy-policy (대체 모델의 평균값이 최대화 되는 지점을 실험. 즉, 지도학습 기반 의사결정) 기반 최적화를 진행한 퍼포먼스를 비교한다.

### **[문제] 새로운 기능성 물질을 효율적으로 개발하기 위한 베이지안 최적화 기반의 의사결정 방법론을 접착용 에폭시 고분자 개발이라는 예시에 적용하여, 블랙박스 최적화에서의 효율적인 의사결정을 가이드 해 보자.**

- “part3\_접작용 에폭시 고분자 개발.py 코드를 활용하여라.

- sklearn, scipy 라이브러리를 설치 하여라

- 챕터 8을 읽고 베이지안 최적화의 이론을 이해하여라.

### **[방법] 접착용 에폭시 생산 모사 함수 구축**

2019년에 science and technology of advanced materials 출간된 Sirawit P.의 논문에서는 접착용 에폭시 고분자를 DGEBA 레진, propylene glycol(경화제)의 두가지 재료를 사용하여 접작용 에폭시 고분자를 생성한다. 이러한 시스템에서 조작변인은 4가지로써, 레진과 글리콜의 비율, 레진의 분자량, 경화제의 분자량, 그리고 반응 온도이다. 논문에서는 이러한 상황 하에서 어떠한 비율, 분자량, 반응 온도를 설정해야 가장 접착력이 높은 에폭시 폴리머를 만들 수 있는지를 베이지안 최적화를 사용하여 찾아낸다(실험 가능한 조합의 수: 256). 이를 간단하게 모사하기 위하여 본 실습에서는 2차원의 input을 받아들이는 Real\_epoxy 함수를 정의하여 베이지안 최적화를 진행한다. 와 같은 Sum-squares 함수를 이번 실습의 에폭시 폴리머의 input-output 상관관계라고 정의한다. 이때 최적해 (global maximum)는 (0,0)에서 0의 꼴로 존재하며, 과의 가능한 실험 후보는 -30 ~ 30까지 0.1의 단위로 존재한다 (실험 가능한 조합의 수: ).

#### 관련 라이브러리를 설치하자.

1. 베이지안 최적화를 진행하기 위해선 numpy, sklearn, scipy등의 라이브러리와 GaussianProcessRegressor, WhiteKernel, RBF 등의 함수가 요구되기에, 이를 불러온다.

|  |
| --- |
| import numpy as np from sklearn.gaussian\_process import GaussianProcessRegressor from sklearn.gaussian\_process.kernels import WhiteKernel, RBF import itertools from scipy.stats import norm |

#### 문제 조건대로 에폭시 접착제의 생산 모사 class를 정의해보자.

1. 문제 상황에서 정의한대로의 코드를 구성해보면 다음과 같이 구성할 수가 있다. 이때 이상적인 상황을 가정하여 실험 결과가 특정 input을 넣었을 때 같은 output이 매번 나온다면 .real의 함수를 사용할 수 있고, 현실적인 상황을 가정하여 실험 결과가 noise에 의해 오염된 상황을 실험해보고 싶다면 .noisy 함수를 사용할 수 있다.

|  |
| --- |
| class Real\_epoxy:  def \_\_init\_\_(self):  pass   def real(self, x):  x1 = x[:,0]  x2 = x[:,1]  return -x1\*\*2 - 2\*x2\*\*2   def noisy(self, x, noise\_var):  x1 = x[0]  x2 = x[1]  return -x1\*\*2 - 2\*x2\*\*2 + np.sqrt(noise\_var) \* np.random.standard\_normal()  test\_func = Real\_epoxy() noise\_var = 0.01 GP\_lengthscale = np.sqrt(0.1) search\_space\_org = np.array(list(itertools.product(np.arange(-30,30,0.1), np.arange(-30,30,0.1)))) y\_max\_real = 0 action\_min = np.array([-30, -30]) action\_max = np.array([30, 30]) |

### **[응용] 베이지안 최적화 기반 최적 생산조건 도출**

모사된 에폭시 input-output 함수를 바탕으로 초기 데이터 확보, 데이터 전처리, 획득함수 정의, 그리고 베이지안 최적화를 진행해보자

#### 의사 결정을 위한 초기 데이터셋을 구축해보자.

1. 초기 데이터셋은 전체 가능한 실험 조합인 360,000개의 점들 중에서 20개를 알고 있다고 가정을 한다. 이때, 어떠한 초기 데이터를 안다고 가정 하는지에 따라 베이지안 최적화의 성능이 많이 다를 수 있다. 따라서 베이지안 최적화의 평균적인 성능을 보기 위하여 랜덤하게 20 실험 포인트의 결과를 알고있는 상태(num\_initial\_data)에서 20번 최적화를 진행을 한 뒤(num\_experiments), 이러한 과정을 초기 데이터를 바꾸어가며 10번 진행을 하게끔 세팅을 한다(num\_runs). 이때 매 의사결정마다 최적해로부터 얼마나 떨어진 지점을 탐색했는지 성과를 수치화 하기 위해 regret\_store라는 빈 list를 만들어 둔다.

|  |
| --- |
| np.random.seed(1) num\_runs = 10 num\_experiments = 20 num\_initial\_data = 20 regret\_store = [] initial\_indices\_store = [] for i in range(num\_runs):  random\_indices = np.random.choice(search\_space\_org.shape[0], size=num\_initial\_data, replace=False)  initial\_indices\_store.append(random\_indices) |

#### 데이터 전처리를 위한 class를 설정해보자.

1. 기계학습을 위하여 input-output 로우 데이터(raw data)를 사용하는 것 보다 데이터를 표준화하여 스케일 된 데이터를 사용하면 기계학습의 성능이 더 나아진다는 것은 널리 알려진 사실이다. 따라서 에폭시 모사 시스템의 대체모델과 획득함수를 정의 하기에 앞서 로우 데이터를 표준화된 데이터로 스케일링 할 수 있는 함수와 표준화된 형태로 나타내어진 데이터를 우리에게 친숙한 로우 데이터의 형태로 다시 바꾸는 함수를 정의 하여야 한다.

|  |
| --- |
| class Scaler:  def \_\_init\_\_(self):  pass   def fit(self, x):  self.mu = np.mean(x,0)  self.std = np.std(x,0)   def transform(self, x):  return (x-self.mu)/self.std   def inverse\_transform\_mean(self, x):  return x\*self.std + self.mu   def inverse\_transform\_std(self, x):  return x\*self.std |

#### Expected improvement 기반 획득함수 class를 설정해보자.

1. 챕터 8에서 소개된 expected improvement의 식은 와 같고, 이를 예측 평균치(와 표준편차(를 활용하여 표현하면 다음과 같이 나타낼 수 있다: . 이때 는 기존에 관찰된 데이터 중에서 가장 성능이 좋았던 값이고, 는 표준 정규 누적 분포 함수이고, 는 표준 정규 분포 함수이다. 이를 코드로 구현하면 다음과 같다.

|  |
| --- |
| class Base\_EI:  def \_\_init\_\_(self):  self.scaler\_EI = Scaler()   def evaluate(self, x, model, y\_max, y\_train):  self.scaler\_EI.fit(y\_train)   mean, std = model.predict(x, return\_std=True)  mean = self.scaler\_EI.inverse\_transform\_mean(mean)  std = self.scaler\_EI.inverse\_transform\_std(std)   a = (mean - y\_max).reshape(-1,)  z = a / std  ei = a \* norm.cdf(z) + std \* norm.pdf(z)   index\_next = np.argmax(ei)  X\_next = x[index\_next]  return X\_next, index\_next |

#### 초기 데이터를 사용한 베이지안 최적화를 통해 의사결정을 진행해보고 의사결정들의 평균 퍼포먼스를 print해보자.

1. 사용하는 획득함수를 Q5에서 정의한 expected improvement로 정의한다. 이후, 실험이 진행됨에 따라 초기 데이터에 가우시안 프로세스 모델을 사용하여 대체모델을 구축한다. 이를 바탕으로 획득함수가 최대화 되는 점을 찾고, real\_epoxy 함수에 넣어서 결과값을 얻고, 얻어진 결과값을 최적해와 비교한 성과지표(regret)를 기록하는것을 하나의 에피소드가 끝날 때까지 20번 반복을 하고(num\_experiments), 초기 데이터를 바꾸어 가며 총 10번의 에피소드(num\_run)을 반복하여 베이지안 최적화를 진행한다.

|  |
| --- |
| acq\_func = Base\_EI() kernel = 1.0 \* RBF(length\_scale=GP\_lengthscale) + WhiteKernel(noise\_level=noise\_var) gpr = GaussianProcessRegressor(kernel=kernel, alpha=0.0, optimizer=None, random\_state=0) scaler = Scaler() for i in range(num\_runs):  print(**"run: "** + str(i))  regrets = []  random\_indices = initial\_indices\_store[i]  print(random\_indices)  X\_train = search\_space\_org[random\_indices,:]  y\_train = test\_func.real(X\_train).reshape(-1,1)  search\_space = np.delete(search\_space\_org, random\_indices, axis=0)   y\_max = np.max(y\_train)  regret = y\_max\_real - y\_max  regrets.append(regret)  for j in range(num\_experiments):  print(**"experiment: "** + str(j))  scaler.fit(y\_train)  y\_train\_scaled = scaler.transform(y\_train)  gpr.fit(X\_train, y\_train\_scaled)   X\_next, index\_next = acq\_func.evaluate(search\_space, gpr, y\_max, y\_train)  print(X\_next)  y\_next = test\_func.noisy(X\_next, 0)  search\_space = np.delete(search\_space, index\_next, axis=0)   X\_train = np.vstack((X\_train, X\_next))  y\_train = np.vstack((y\_train, y\_next))   y\_max = np.max(y\_train)  regret = y\_max\_real - y\_max  print(regret)  regrets.append(regret)   regret\_store.append(regrets)  regret\_store = np.array(regret\_store) avg\_regret = np.mean(regret\_store, axis=0) print(avg\_regret) |

이때 print(avg\_regret)의 결과로는 다음과 같은 list가 프린트 된다.

[101.293 99.311 98.47 98.47 97.717 97.717 96.724 96.125 95.136 92.865 92.254 90.46 88.852 87.793 86.667 85.133 84.711 82.068 81.889 80.708 79.627]

즉, 평균적으로 매 time-step마다 베이지안 최적화 기반의 의사결정이 최적해에 근접한 값들을 제시해 나갔다는 것을 알 수 있다.

#### 초기 데이터를 사용한 greedy-policy기반 최적화를 통해 의사결정을 진행해보고 의사결정의 효율성을 베이지안 기반 최적화와 비교해보자.

1. Q6의 성능 지표가 다른 의사결정 방법론보다 좋은지 확인할 대조군 제작 위해, 대체 모델의 불확실성에 대한 고려 없이 예측값의 최댓값만을 실험해 나가는 greedy policy 기반의 의사결정을 진행한 뒤(supervised learning기반 의사결정), 그 결과(avg\_regret)를 베이지안 최적화의 결과와 비교하여 베이지안 최적화의 의사결정 효율성을 인식한다.

|  |
| --- |
| class Base\_EI:  def \_\_init\_\_(self):  self.scaler\_EI = Scaler()   def evaluate(self, x, model, y\_max, y\_train):  self.scaler\_EI.fit(y\_train)   mean, std = model.predict(x, return\_std=True)  mean = self.scaler\_EI.inverse\_transform\_mean(mean)  std = self.scaler\_EI.inverse\_transform\_std(std)   a = (mean - y\_max).reshape(-1,)  z = a / std  ei = a \* norm.cdf(z) + std \* norm.pdf(z)   index\_next = np.argmax(mean)  X\_next = x[index\_next]  return X\_next, index\_next |

코드상으론 기존의 베이지안 최적화 기반 코드와 대부분이 똑같지만, 획득함수를 정의함에 있어서 index\_next = np.argmax(ei)가 아닌, index\_next = np.argmax(mean)로 바꾸어주면 예측값의 최대치를 추천하는 greedy policy 기반의 의사결정 실험이 된다. 이때 결과를 print해보면 다음과 같은 list가 출력된다.

[101.293 100.358 99.959 99.959 98.678 98.678 98.678 97.699 97.348 96.92 96.236 95.887 95.27 94.678 94.312 93.779 93.037 92.675 92.164 91.426 91.083]

Q6의 결과와 비교하여 불확실성을 고려한 베이지안 최적화 기반 의사결정의 데이터 효율성을 입증할 수 있다.

### **[결론]**

본 장에서는 베이지안 최적화 기반 의사결정의 효율성을 모사된 접착용 에폭시 폴리머 생산 공정에 적용하여 능동학습을 실제로 진행해보았고, 이러한 의사결정을 탐욕정책(greedy policy)기반 의사결정과정과 성능을 비교하여 불확실성을 고려한 의사결정과정의 효율성을 몸소 체험해보는 공부를 하였다.

### **학습 결과**

* 학습 내용

접착용 에폭시 폴리머 생산 공정을 모사해보기.

* 학습 결과 확인하기

베이지안 최적화 기반 의사결정을 접착용 에폭시 폴리머 공정에 적용해보기.

* 학습 결과 응용하기

베이지안 최적화 기반 의사결정과 탐욕정책 기반 의사결정의 데이터 효율성 비교해보기