1. 화공산업에서의 인공지능

# **인공지능 기반 물질 개발 및 거동 분석**

* 1. **인공지능 기반 물질 개발 및 거동 분석**

### **수성 가스 전이 반응 촉매 분석**

최근 수소가 환경 친화적인 차세대 에너지 전달체로 부각됨에 따라, 화공 산업에서도 다양한 수소 생산 기술에 관한 연구가 진행되고 있다. 특히, 수소 생산을 위한 대표적인 반응인 수성 가스 전이 반응 (water gas shift reaction, WGSR)은 고순도 수소를 생산하는 기술로, 수소 대량 생산 기술로써 그 역할이 매우 중요하다. WGSR은 합성가스 내의 일산화탄소와 수증기가 반응하여 수소와 이산화탄소를 생성하는 반응으로, 아래와 같은 반응식을 따른다.

고효율 WGSR 공정의 핵심은 상기 평형 반응을 열역학적 평형 전환율의 한계까지 끌어올리기 위한 고성능 WGSR 촉매 개발이다. 이때, 촉매의 성능은 활성 (전환율, 선택도) 및 안정성 (기계적 내구성, 열적 안정성)으로 대표된다. 현재 촉매 성능 개선을 위한 연구가 활발히 진행되고 있으나, 촉매 재료와 반응 운전 조건 간의 복잡한 상호관계로 인해, 실험을 통한 시행착오법으로 최적의 촉매를 도출하는데 어려움이 있다. 만약, 수치적으로, 촉매 재료와 운전 조건의 변화에 따른 촉매의 성능 변화를 정확히 예측 가능하다면, 기술개발에 필요한 시간을 효과적으로 단축할 수 있을 것이다. 인공신경망은 고차원 데이터를 처리할 수 있는 알고리즘으로, 복잡계 해석을 위한 유용한 방법론이다. 본 실습에서는 인공신경망 알고리즘을 이용하여 고성능 WGSR 촉매의 성능을 예측하고 분석하는 방법을 익힌다.

### **[문제]**

**촉매 물질 조건 (금속 함량, 가공 환경)과 운전 조건 (반응물 조성, 온도, 압력 등)을 이용하여 촉매 성능 예측 모델을 개발하고 모델의 정확도를 평가하라.**

**- “코드 및 데이터/4-1. WGSR.csv” 데이터를 활용하여라.**

**-학습 및 평가 데이터는 7:3 비율로 분리하여라.**

**- 95% 이상의 정확도를 갖는 모델을 개발하라.**

### **[방법] 인공신경망 예제**

#### 인공신경망을 활용하기 위한 라이브러리를 조사하라.

1. 인공신경망 학습을 위한 다양한 라이브러리가 존재한다.

|  |
| --- |
| Library(‘neuralnet’) |

|  |
| --- |
| Library(‘H2O’) |

|  |
| --- |
| Library(‘keras’) |

#### 인공신경망 학습에는 다양한 하이퍼파라미터가 선행적으로 결정되야 한다. 이들 중 모델 정확도와 직접적으로 연관 있는 하이퍼파라미터 중 활성화 함수에 대해 설명하라.

1. Activation function은 인공신경망의 node를 활성화시키는 활성화 함수로, input된 가중치 합을 출력 신호로 변환하는 함수이다. 인공 신경망에서 이전 레이어에 대한 가중 합의 크기에 따라 활성 여부가 결정된다. 목적 및 역할에 따라 선택적으로 사용된다.

### **[응용] 인공 신경망 기반 촉매 반응 예측**

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.

|  |
| --- |
| WGSR <- read.csv(file.choose(), header = T) |

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

|  |
| --- |
| View(WGSR) |

‘view()’ 함수를 사용하여 데이터를 확인할 수 있다.

|  |
| --- |
| WGSR\_re <- WGSR |

‘WGSR’ 데이터의 손상을 막기위해 ‘WGSR\_re’ 데이터를 생성하여 사용한다.

#### 인공지능 학습을 위해서 데이터의 단위를 무시한 상대적 영향력을 파악할 필요가 있다. 촉매 반응 데이터를 표준화하라. 또한, 데이터의 결측치를 제거하라.

1. 다음과 같은 절차를 통해 표준화 및 결측치 제거가 가능하다.

|  |
| --- |
| WGSR\_scale <- scale(WGSR\_re) |

‘scale()’ 함수를 사용하여 ‘WGSR\_re’ 데이터를 전처리한다.

|  |
| --- |
| WGSR\_scale <- data.frame(replace(WGSR\_scale, is.na(WGSR\_scale), 0) |

‘replace()’ 함수를 사용하여 ‘WGSR\_scale’ 데이터의 공백값 (‘NA’)를 제거하도록 한다. 이때, ‘is.na’ arg를 활용해 해당 데이터의 공백값만을 인식시킬 수 있다.

#### 인공신경망 기반 예측 모델의 정확도를 검증하기 위해 데이터를 모델 학습 데이터와 모델 검증 데이터로 분할하라.

1. 다음과 같은 절차를 통해 데이터의 분할이 가능하다.

|  |
| --- |
| smp\_size <- floor(0.7 \* nrow(WGSR\_scale))  train\_ind <- sample(seq\_len(nrow(WGSR\_scale)), 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.

|  |
| --- |
| neural\_train <- WGSR\_scale[train\_ind,]  neural\_test <- WGSR\_scale[-train\_ind,] |

임의로 추출된 행 번호 (‘train\_ind’)를 사용해서 ‘WGSR\_scale’ 데이터의 행들을 추출한다. 70%가 학습에 사용할 데이터로 ‘neural\_train’ 으로 명명한다. ‘train\_ind’ 행 번호를 제외한 데이터는 검증에 사용될 데이터이므로 ‘neural\_test’로 명명한다.

|  |
| --- |
| name\_col <- c(colnames(WGSR\_scale))  name\_col <- name\_col[1:38] |

인공신경망 학습을 위해 변수 이름을 지정하는 과정이 필요하다. ‘WGSR\_scale’ 데이터의 열 이름을 추출하여 ‘name\_col’로 명명한다. 해당 ‘name\_col’의 38번까지가 독립변수이므로 해당 부분을 추출하여 ‘name\_col’을 재 정의한다.

#### 인공 신경망 학습을 시도하고 정확도를 검증하라.

1. 다음과 같은 절차를 통해 인공 신경망 학습을 시도할 수 있다.

|  |
| --- |
| vari\_name <- as.formula(paste(‘CO.Conversion ~’,  paste(name\_col,  collapse = ‘+’))) |

‘CO.conversion’은 데이터의 종속변수에 해당한다. ‘paste()’함수를 사용해서 ‘name\_col’의 변수들을 ‘+’ 기호로 연결된 일련의 문장으로 만든다. 그리고 ‘CO.conversion~’과 결합한다. 해당 문자열을 ‘as.formula’ 함수를 통해 수식으로 인식시키고, ‘vari\_name’으로 명명한다.

|  |
| --- |
| set.seed(7)  WGSR\_scale\_neural <- neuralnet(vari\_name, neural\_train,  hidden = c(34, 10, 8, 8, 1)) |

‘set.seed()’를 ‘7’로 지정하여 계산의 무작위성을 고정한다. ‘neuralnet()’ 함수를 사용해서 인공신경망 모델을 구축한다. Arg로 ‘vari\_name’을 사용해 데이터의 독립 변수 및 종속 변수 관계를 지정하고, ‘neural\_train’ 데이터를 사용한다는 것을 명시한다. ‘hidden’ arg를 통해 인공신경망의 구조를 설정한다. 구조는 다섯개의 은닉층과 각 층에 해당하는 node인 34-10-8-8-1로 구성된다. 해당 인공신경망을 ‘WGSR\_scale\_neural’로 명명한다.

|  |
| --- |
| WGSR\_scale\_neural$result.matrix |

‘$result.matrix’를 사용하면 ‘WGSR\_scale\_neural’의 weight, bias, 그리고 학습 결과를 확인할 수 있다.

|  |
| --- |
| pred\_result <- data.frame(compute(WGSR\_scale\_neural, neural\_test[,1:38]))  pred\_result\_train <- data.frame(compute(  WGSR\_scale\_neural, neural\_train[,1:38]) |

‘compute’ 함수를 통해 ‘WGSR\_scale\_neural’모델을 사용할 수 있다.‘neural\_test’ 데이터의 독립변수 부분을 지정하여 예측해보도록 한다.마찬가지로 ‘neural\_train’도 예측하여 학습 정확도를 확인해보도록 한다.

|  |
| --- |
| Pred\_train\_unscaled <- data.frame(mean(WGSR$CO.Conversion) +  pred\_result\_train$net.result \* sd(WGSR$CO.Conversion))  neural\_train\_unscaled <- data.frame(mean(WGSR$CO.Conversion) +  neural\_train$net.result \* sd(WGSR$CO.Conversion)) |
| Pred\_result\_unscaled <- data.frame(mean(WGSR$CO.Conversion) +  pred\_result$net.result \* sd(WGSR$CO.Conversion))  neural\_test\_unscaled <- data.frame(mean(WGSR$CO.Conversion) +  neural\_test$net.result \* sd(WGSR$CO.Conversion)) |

예측된 데이터는 scale된 값이므로 unscale할 필요가 있다. 따라서, 수식을 역전시켜 예측된 학습 데이터와 검증 데이터의 원본 값을 추출하도록 한다.

|  |
| --- |
| rsq(neural\_train\_unscaled[,1], pred\_train\_unscaled[,1])  Metrics::mse(neural\_train\_unscaled[,1], pred\_train\_unscaled[,1])  Metrics::rmse(neural\_train\_unscaled[,1], pred\_train\_unscaled[,1]) |
| rsq(neural\_test\_unscaled[,1], pred\_ test\_unscaled[,1])  Metrics::mse(neural\_ test\_unscaled[,1], pred\_ test\_unscaled[,1])  Metrics::rmse(neural\_ test\_unscaled[,1], pred\_ test\_unscaled[,1]) |

정확도 파악을 위해 각 unscale된 데이터의 R2, MSE, RMSE 값을 확인한다.

### **[결론]**

본 장에서는 인공신경망 기법에 기반한 WGSR 촉매 성능 예측 모델을 개발하고, 정확도를 평가하였다. 또한, 모델 개발을 위한 데이터 전처리 과정으로 데이터 표준화를 진행하였으며, 인공신경망의 하이퍼파라미터를 조절하여 모델 정확도 개선 방법을 학습하였다. 이를 통해, 높은 모델 정확도를 확보하기 위한 과정으로, 데이터 전처리와 모델 하이퍼파라미터 튜닝의 중요성을 이해하였다.

### **[학습 결과]**

* 학습 내용

고성능 수성화 전이 반응 촉매 개발을 위한 인공 신경망 기반 예측 방법론 익히기

* 학습 결과 확인하기

인공 신경망 알고리즘의 활용 방법 및 예측 모델 학습을 위한 데이터 구조화 익히기

* 학습 결과 응용하기

본 장의 학습내용에 기반해 촉매 성능의 예측과 예측 알고리즘의 정확도 향상시키기

* 1. **인공지능 기반 물질 개발 및 거동 분석**

### **이온성 액체의 무한 희석 활성도 계수 추정**

### 이온성 액체 (Ionic liquids)는 그 성분과 구조를 사용목적에 따라 선택적으로 합성 및 사용할 수 있는 설계 물질 (designer material)로, 흔히 탄화수소 추출 공정에 이용된다. 이온성 액체는 독특한 물리/화학적 특성(전기화학적 안정성, 높은 전도성 등)으로 인해, 기존의 유기 용매 기반의 추출 공정을 대체할 친환경적인 대안으로 주목을 받고 있다.

### 이온성 액체의 거동 원리는 보통 묽은 용액에서의 활동도 계수 (Infinite diluent activity coefficient, IDAC)에 기반하여 해석된다. 이를 계산하기 위한 방법으로 UNIFAC, COSMO-RS, 혹은 Abraham과 같은 mechanistic model들이 사용됐다. 하지만, 활동도 계수 계산을 위한 mechanistic model들은 낮은 정확도와 한정된 데이터를 기반으로 작동한다는 단점이 명확하다. 또한, 수많은 용매와 용질 사이의 활동도 계수 실험은 상당한 시간과 예산을 필요로 하는 작업으로, 현재의 이온성 액체 개발과 적용은 비효율적인 상태에 머물러있다.

### 인공신경망 (artificial neural networks)은 데이터 기반 학습을 통한 복잡계 현상의 예측 모델을 생성하는 인공지능 알고리즘이다. 인공신경망은 고차원의 데이터 학습에 높은 성능을 보여주고 있고, 오늘날 다양한 물질 개발 및 공정 개선 연구에 활용되고 있다.

### 본 실습에서는 이온성 액체의 물리/화학적 특성 데이터 베이스를 활용하여, 인공신경망을 통한 활동도 계수 예측모델을 개발한다.

### **[문제]**

**이온성 액체의 물리/화학적 특성 (밀도, 임계 온도 등)을 이용하여 이온성 액체의 열역학성 특성 예측 모델을 개발하고 모델의 정확도를 평가하라.**

**- “코드 및 데이터/4-2. IL\_data.csv” 데이터를 활용하여라.**

**-학습 및 평가 데이터는 7:3 비율로 분리하여라.**

**- 95% 이상의 정확도를 갖는 모델을 개발하라.**

### **[방법] 인공신경망 예제**

#### 인공신경망 하이퍼파라미터 최적화를 위한 방법론을 조사하라.

1. 하이퍼파라미터 최적화를 위해 전통적으로는 manual search, grid search, random search 방법이 사용되었으며, 최근에는 통계적인 혹은 기계학습 방법으로 genetic algorithm, Bayesian optimization도 널리 사용되고 있다.

#### 인공신경망 학습을 위해 데이터의 정규화/표준화가 필요한 이유를 설명하고, 일반적인 정규화/표준화 이외의 데이터 전처리 방법을 조사하라.

1. 인공신경망은 데이터의 변위 및 크기를 인식하는 과정이 가중치 설정에 영향을 준다. 따라서 데이터의 정확한 상대적 영향력 파악을 위해 데이터의 정규화/표준화가 필요하다.

아래에 일반적인 정규화/표준화 이외의 데이터 전처리 방법을 보였다.

* L1 (Manhattan distance) normalization
* L2 (Euclidean distance) normalization
* L infinity normalization

### **[응용] 인공 신경망 기반 이온성 액체의 열역학적 특성 예측**

예제는 Python 3.5 프로그래밍 언어를 기준으로 Jupyter 개발환경에서 작성되었다. 예제 실습을 위해 Anaconda 프로그램을 아래 URL로부터 다운로드할 수 있다.

https://www.anaconda.com/

#### 데이터를 Jupyter환경으로부터 불러오고 확인하라.

1. 다음과 같은 code를 사용하여 불러올 수 있다.

|  |
| --- |
| import numpy as np  import pandas as pd  import matplotlib.pyplot as plt |

‘pandas’ package는 데이터 편집을 위한 도구로, python에서 매우 유용하게 사용된다. 본 실습에서는 OSN 데이터 불러오기 및 데이터 편집 (컬럼 및 이름 설정)을 위해 사용한다.

‘numpy’ package는 수학적 기능이 탑재된 도구로, 난수 발생과 대수적 계산을 위해 사용된다. 본 실습에서는 데이터의 형태 변환 및 벡터 계산을 위해 사용한다.

‘matplotlib.pyplot’ package는 시각화를 위한 그래프 도구로, 다양한 그래프를 그리기 위해 사용된다.

‘pandas’ package를 import하고 ‘pd’로 축약해 사용한다.

‘numpy’ package를 import하고 ‘np’로 축약해 사용한다.

‘matplotlib.pyplot’ package를 import하고 ‘plt’로 축약해 사용한다.

|  |
| --- |
| IDAC = pd.read\_csv('IL\_data.csv')  df = pd.DataFrame(data = IDAC)  df\_copy = df.copy()  df\_copy |

테이블이(가) 표시된 사진

자동 생성된 설명

‘pd’를 사용하여 Jupyter 환경에 있는 ‘IL\_data.csv’ 데이터를 Jupyter script 환경에 불러온다.

‘IL\_data.csv’는 ‘pd’를 활용, 데이터 프레임 형태인 df로 정의한다.

‘.copy’ 함수를 사용하여 ‘df’ 데이터의 복사본인 ‘df\_copy’를 만든다. 이 과정은 원본 데이터의 훼손을 피하기위해 실행된다.

#### 재현성 있는 모델 구축을 위해 계산의 무작위성을 고정하고 데이터를 정규화하라.

1. 다음과 같은 절차를 통해 무작위성 고정과 데이터 정규화가 가능하다.

|  |
| --- |
| import random as rn |

‘random’ module은 무작위 숫자 생성 등 무작위 관련 함수를 제공한다. 본 실습에서 훈련 및 검증 데이터셋 생성을 위해 사용한다.

‘random’ module을 import하고 ‘rn’으로 축약해 사용한다.

|  |
| --- |
| import sklearn  from sklearn import preprocessing  from sklearn.preprocessing import MinMaxScaler |

‘sklearn’ package는 계산을 위한 유용한 함수를 다수 내장하고 있다. ‘sklearn.preprocessing’의 ‘MinMaxScaler’ 함수를 import한다.

|  |
| --- |
| from sklearn.model\_selection import train\_test\_split  from sklearn.metrics import mean\_squared\_error, r2\_score |

‘sklearn.model\_selection’의 ‘train\_test\_split’ 함수를 import하도록 한다. ‘train\_test\_split’ 함수는 모델 학습 및 검증을 위한 모델 학습 데이터와 검증 데이터를 분할하기 위해 사용한다.

‘sklearn.metrics’의 ‘mean\_square\_error’ 함수와 ‘r2\_score’ 함수를 import하도록 한다. ‘mean\_square\_error’와 ‘r2\_score’는 모델 검증을 위한 정확도 지표를 내장하고 있는 함수이다.

|  |
| --- |
| from keras import Sequential, optimizers, metrics  from tensorflow.keras import backend as K  from keras.layers import Dense, BatchNormalization, Dropout, Activation  from keras.optimizers import Adam, SGD  from keras.initializers import lecun\_normal |

‘keras’는 ‘TensorFlow’와 같은 머신러닝 백앤드 엔진 플랫폼을 지원하는 라이브러리 이자 API (Application programming interface)이다.

‘keras’로부터 ‘Sequential’, ‘optimizer’, ‘metrics’ 함수들을 import한다.

‘tensorflow.keras’ 함수로부터 ‘backend’ import하고 ‘K’로 축약한다.

‘keras.layers’로부터 ‘Dense’, ‘BatchNormalization’, ‘Dropout’, ‘Activation’ 함수들을 import한다.

‘keras.optimizers’로부터 ‘Adam’ 함수를 import한다.

‘keras.initializers’로부터 ‘lecun\_normal’ 함수를 import한다.

|  |
| --- |
| sc = MinMaxScaler() |

‘MinMaxScaler()’ 함수를 ‘sc’로 명명하여 사용.

|  |
| --- |
| standardized\_copy = df\_copy  standardized\_copy.iloc[:,2:] = sc.fit\_transform(df\_copy.iloc[:,2:])  standardized\_copy = pd.DataFrame(standardized\_copy,  columns = df\_copy.columns)  df\_copy = standardized\_copy |

‘sc.fit\_transform’ 함수를 사용하여 ‘df\_copy’ 데이터의 numerical value만 (3열부터 17열)을 정규화한다.

정규화한 데이터는 ‘standardized\_copy’로 명명하고 데이터 프레임 형태로 변환한다.

‘standardized\_copy’ 데이터를 ‘df\_copy’ 로 명명한다.

|  |
| --- |
| X = df\_copy.iloc[:,2:-1]  y = df\_copy[['ln IDAC']] |

‘df\_copy’ 데이터의 독립변수 부분인 2열부터 16열까지를 추출하여 ‘X’로 명명한다.

‘df\_copy’ 데이터의 종속변수 부분인 ‘in IDAC’을 추출하여 ‘Y’로 명명한다.

#### 인공신경망 기반 예측 모델의 정확도를 검증하기 위해 데이터를 모델 학습 데이터와 모델 검증 데이터로 분할하라.

1. 다음과 같은 절차를 통해 데이터의 분할이 가능하다.

|  |
| --- |
| tf.random.set\_seed(34)  randomstate=8 |

‘tf’의 ‘.random.set\_seed’ 함수를 사용하여 데이터 추출의 무작위성을 고정시킨다.

마찬가지로 인공신경망 학습의 무작위성을 고정시키기 위해 ‘randomstate’를 생성하고 차후 학습에 사용한다.

|  |
| --- |
| X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y, test\_size = .3,  random\_state = randomstate) |

‘train\_test\_split’ 함수를 사용하여 학습데이터와 검증데이터를 생성한다.

‘X\_train’과 ‘Y\_train’은 각각 학습데이터의 독립변수와 종속변수 부분이고 마찬가지로 ‘X\_test’와 ‘Y\_test’는 각각 검증데이터의 독립변수와 종속변수 부분이다.

본 실습에서는 학습에 전체 데이터의 70%를 할당할 것이므로, ‘test\_size’ arg를 ‘.3’으로 설정한다. 모델 학습의 재현성을 위해 데이터 추출의 무작위성을 고정한다 (‘random\_state’ arg).

#### 인공신경망 기반 예측 모델을 구축하라. 모델 학습을 위해 활성화함수로 ‘swish’, optimizer로 ‘adam’, 그리고 손실함수는 ‘mean\_square\_error’를 이용하라.

1. 다음과 같은 절차를 통해 인공신경망 학습이 가능하다.

|  |
| --- |
| model = Sequential()  model.add(Dense(units=50, kernel\_initializer=lecun\_normal(seed=**None**),  activation='swish', input\_dim=X.shape[1]))  model.add(Dense(units=100, kernel\_initializer=lecun\_normal(seed=**None**),  activation = 'swish'))  model.add(Dense(units=150, kernel\_initializer=lecun\_normal(seed=**None**),  activation = 'swish'))  model.add(Dense(units=200, kernel\_initializer=lecun\_normal(seed=**None**),  activation = 'swish'))  model.add(Dense(units=250, kernel\_initializer='normal', activation = 'swish'))  model.add(Dense(units=150, kernel\_initializer='normal', activation = 'swish'))  model.add(Dense(units=100, kernel\_initializer='normal', activation = 'swish'))  model.add(Dense(units=50, kernel\_initializer='normal', activation = 'swish'))  model.add(Dense(units=1, kernel\_initializer='normal'))  model.compile(loss='mean\_squared\_error',  optimizer=Adam(lr=0.001, beta\_1=0.9, beta\_2=0.999,  epsilon=1e-8), metrics=[metrics.mse]) |

‘sequential()’ 함수를 사용하여 인공신경망 모델을 생성한다. 모델은 ‘model’로 명명하고 인공신경망 층은 ‘.add’를 통해 추가할 수 있다.

일반적인 신경망 모델을 사용하기 위해 ‘Dense’ 기능을 넣고, 각 노드를 지정한다.

Activation argument는 인공신경망의 활성화 함수에 해당하며, Input shape argument는 데이터의 column 차원이다.

본 실습에선 ‘swish’를 활성화 함수로 사용한다.

‘.compile’ 함수를 통해 학습의 optimizer와 loss function을 설정할 수 있다.

본 실습에서 사용된 Optimizer는 adam optimizer이고 loss function으로 MSE (Mean square error)를 사용한다.

|  |
| --- |
| model.fit(X\_train, y\_train, batch\_size=10, epochs=1000, verbose=0) |

‘.fit’ 함수를 사용하여 모델 학습을 진행한다.

모델 학습에 ‘X\_train’와 ‘y\_train’ 데이터를 사용한다. ‘batch\_size’ arg를 통해 한번에 10개 데이터셋을 고려하도록 한다.

또한, ‘epochs’ arg를 통해 반복 학습의 주기를 1000으로 지정한다.

‘verbose’ arg는 모델 학습의 진행상황을 표현하는 것으로, 0으로 설정하여 진행상황을 표시하지 않는 것으로 지정한다.

#### 인공신경망 기반 예측 모델의 정확도를 파악하라.

1. 다음과 같은 절차를 통해 정확도 검증이 가능하다.

|  |
| --- |
| y\_train\_pred = model.predict(X\_train)  train\_mse = mean\_squared\_error(y\_train, y\_train\_pred)  train\_r2 = r2\_score(y\_train, y\_train\_pred) |

학습이 끝나면, ‘.predict’ 함수를 사용해서 ‘X\_train’ 데이터를 예측한다. 예측된 결과값은 ‘y\_train\_pred’로 명명한다.

‘mean\_squared\_error’와 ‘r2\_score’함수를 사용해서 ‘y\_train’과 ‘y\_train\_pred’의 정확도를 확인한다.

|  |
| --- |
| train\_mse |



|  |
| --- |
| train\_r2 |



학습의 정확도를 확인한다.

|  |
| --- |
| y\_pred = model.predict(X\_test)  test\_mse = mean\_squared\_error(y\_test, y\_pred)  test\_r2 = r2\_score(y\_test, y\_pred) |

마찬가지로 ‘mean\_squared\_error’와 ‘r2\_score’함수를 사용해서 ‘y\_test’과 ‘y\_test\_pred’의 정확도를 확인한다.

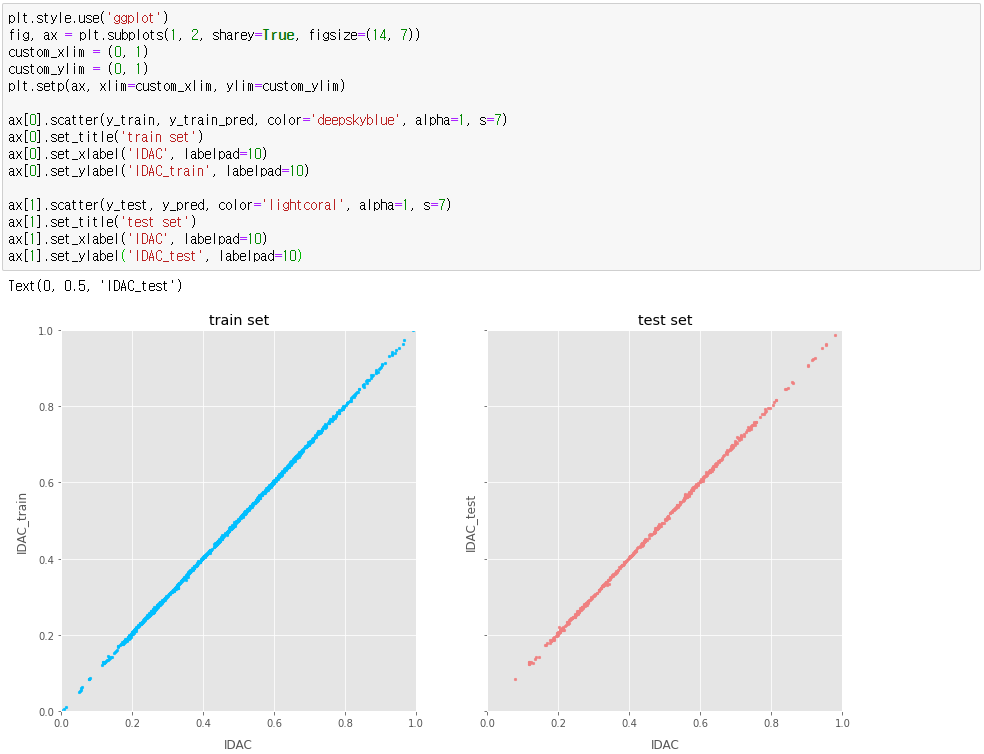
|  |
| --- |
| test\_mse |



|  |
| --- |
| test\_r2 |



|  |
| --- |
| plt.style.use('ggplot')  fig, ax = plt.subplots(1, 2, sharey=**True**, figsize=(14, 7))  custom\_xlim = (0, 1)  custom\_ylim = (0, 1)  plt.setp(ax, xlim=custom\_xlim, ylim=custom\_ylim)  ax[0].scatter(y\_train, y\_train\_pred, color='deepskyblue', alpha=1, s=7)  ax[0].set\_title('train set')  ax[0].set\_xlabel('IDAC', labelpad=10)  ax[0].set\_ylabel('IDAC\_train', labelpad=10)  ax[1].scatter(y\_test, y\_pred, color='lightcoral', alpha=1, s=7)  ax[1].set\_title('test set')  ax[1].set\_xlabel('IDAC', labelpad=10)  ax[1].set\_ylabel('IDAC\_test', labelpad=10) |



‘plt’를 사용하여 정확도를 시각화한다.

### **[결론]**

본 장에서는 이온성 액체의 열역학적 특성(활동 계수)을 예측하기위해 인공신경망을 이용하였다. 데이터 정규화를 통해 데이터 전처리를 진행하였다. 또한, 모델 개발을 위해 인공신경망의 하이퍼파라미터를 조정하여 모델 정확도 개선 방법을 학습하였다. 결과로, 열역학적 특성 예측을 위한 모델 개발 방법을 학습하였다.

### **[학습 결과]**

* 학습 내용

추출 공정의 열역학 특성 분석을 위한 인공 신경망 기반 예측 방법론 익히기

* 학습 결과 확인하기

인공 신경망 알고리즘의 활용 방법 및 예측 모델 학습을 위한 데이터 구조화 익히기

* 학습 결과 응용하기

본 장의 학습내용에 기반해 추출 공정 설계를 위한 인공 신경망 적용 방법 조사하기

* 1. **인공지능 기반 물질 개발 및 거동 분석**

### **유기용매 막분리 소재 분석**

유기용매 나노여과 분리막(Organic solvent nanofiltration, OSN)은 용질의 선택적 분리 기능과 효율적인 물질 투과 능력으로 인해, 난분해성 폐수의 분리에 탁월한 성능을 보인다. 이로 인해 유기용매 나노여과 분리막은 화학, 환경, 의료, 제약, 식품공업 등 다양한 산업분야로 응용이 가능하다.

유기용매 나노여과 분리막은 친환경적이고 효율적인 분리 기술로서, 추가적인 분리 성능 향상을 위해 분리막 개선에 관한 많은 연구가 시도되고 있다. 하지만 사용 환경에 따라 그 성능이 크게 변화하는 분리막의 특성 상, 변수 변화에 따른 분리막 성능 예측이 어려운 실정이다. 이에 따라, 고성능 분리막 설계 과정에서, 반복적인 실험에 의존하는 것이 보편적이다. 하지만, 시행착오법은 많은 예산과 시간을 소모하여 다양한 조건에서의 분리막 설계를 지연시키는 한계점을 갖는다. 그 결과 새로운 분리막 개발을 통한 분리 공정 개선은 상당히 소극적으로 이루어지며 낮은 성공률을 갖는다. 분리막의 기술개발 효과를 증가시키기 위해서는 분리막 성능 예측이 필수적이며, 특히, 분리막의 주요 변수를 규명하여 정형화된 분리막 성능의 화학적 차원(Chemical dimension)을 정의한다면, 좋은 성능 예측 모델의 초석이 될 수 있다.

본 실습에서는 차원 축소 방법론 기반 분리막 성능 예측과 주요 변수 규명을 목적으로 한다.

### **[문제]**

**유기용매 막분리 소재의 화학적 차원 조사를 위해 주성분 분석 (principal component analysis; PCA) 방법론을 이용하고, 막분리 소재 성능 결정의 주요 변수를 규명하라.**

**- “코드 및 데이터/4-3. OSN\_membrane\_data.csv” 데이터를 활용하여라.**

**- 데이터의 80~90%를 설명할 수 있는 성분 수를 파악하라.**

**- 가장 높은 데이터 설명력을 갖는 두가지 성분에 기반해 데이터를 정보 차원에서 시각화하고 두 성분에 대한 변수 기여도를 파악하라.**

### **[방법] PCA 예제**

#### PCA을 활용하기 위한 라이브러리를 조사하라.

1. PCA를 활용하기 위한 라이브러리는 대표적이게 sklearn의 ‘sklearn.decomposition’이 있다. PCA와 함께 다양한 차원축소 관련 알고리즘을 제공한다.

|  |
| --- |
| From sklearn.decomposition import PCA |

#### 화학적 차원에서 정보 차원으로의 투영에는 데이터 손실이 필연적으로 발생한다. 데이터 손실의 허용 수준을 조사하라.

1. PCA는 변수의 수 (n) 만큼 정보 차원의 축을 생성한다. 축은 principal component (PC) 라고 부르며, 가장 데이터를 잘 설명하는 차원 (PC1) 부터 가장 설명을 못하는 차원 (PCn) 순으로 분류된다. 여기서 데이터 설명력이 80~90% 되는 수준의 PC 수를 채택해야 한다.

### **[응용] PCA 기반 여과 공정 소재 특성 분석**

예제는 Python 3.5 프로그래밍 언어를 기준으로 Jupyter 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.

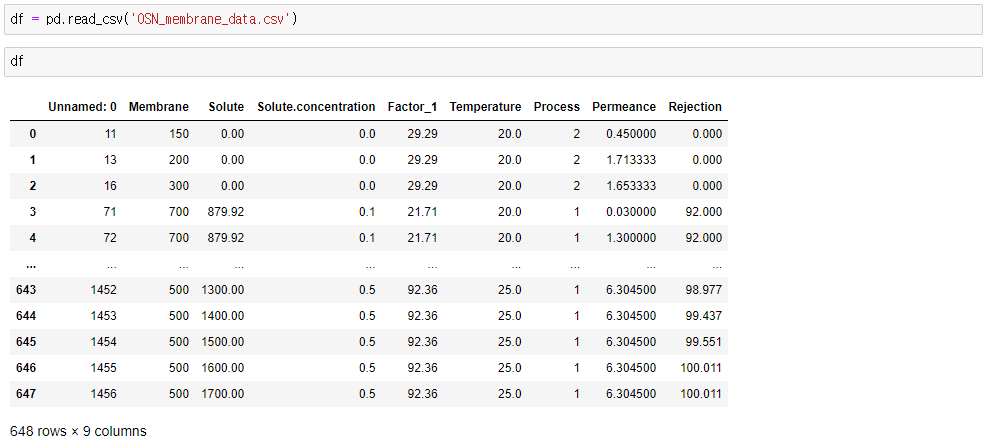
|  |
| --- |
| import pandas as pd  import numpy as np  import matplotlib.pyplot as plt |

‘pandas’ package는 데이터 편집을 위한 도구로, python 언어에서 매우 유용하게 사용된다. OSN 데이터 불러오기 및 데이터 편집 (컬럼 및 이름 설정)을 위해 본 실습에서 사용한다. ‘pandas’ package를 import하고 ‘pd’로 축약해 사용한다.

‘numpy’ package는 수학적 기능이 탑재된 도구로, 난수 발생과 대수적 계산을 위해 사용된다. 본 실습에서는 데이터의 형태 변환 및 벡터 계산에 사용한다. ‘numpy’ package를 import하고 ‘np’로 축약해 사용한다.

‘matplotlib.pyplot’ package는 시각화를 위한 그래프 도구로, 다양한 그래프를 그리기 위해 사용된다. ‘matplotlib.pyplot’ package를 import하고 ‘plt’로 축약해 사용한다.

|  |
| --- |
| df = pd.read\_csv('OSN\_membrane\_data.csv')  df |



데이터는 ‘OSN\_membrane\_data.csv’로 Jupyter 환경에 저장됐다. ‘pd’에 내장된 read.csv 함수를 사용하여 해당 데이터를 Jupyter script 환경에 불러온다.

해당 데이터는 Jupyter script 환경에서 df로 명명되었으며, df를 타이핑하여 데이터를 읽어온다.

|  |
| --- |
| OSN\_data = df.iloc[:,1:9]  OSN\_data.columns = ['Membrane\_MWCO','Solute\_MW','Solute\_concentration', 'Solvent\_factor', 'Temperature','Process\_configuration', 'Permeance', 'Rejection']  OSN\_data |

테이블이(가) 표시된 사진

자동 생성된 설명

행 번호 (row name)가 포함된 첫번째 열을 제거하고 사용한다. 해당 열은 첫번째 열이고, python에서는 0번째 열이다.

행 번호가 삭제된 데이터는 ‘OSN\_data’로 명명하여 사용한다. ‘.colums’ 함수를 사용하여 ‘OSN\_data’의 열 이름을 변수 이름으로 설정한다.

#### 주성분 분석은 데이터의 스케일을 무시한 영향력을 확인이 가능하다. 주성분 분석을 위해 데이터를 표준화하라.

1. 다음과 같은 절차를 통해 표준화한다.

|  |
| --- |
| from sklearn.preprocessing import StandardScaler |

‘sklearn’ package는 계산을 위한 유용한 함수를 다수 내장하고 있다. ‘sklearn.preprocessing’의 ‘StandardScaler’ 함수를 import하도록 한다.

|  |
| --- |
| OSN\_data\_x = OSN\_data.loc[:, :].values |



‘OSN\_data’의 값만을 ‘OSN\_data\_x’로 가져온다. 이때 ‘.loc’ 함수를 사용하여 데이터를 지정하고, ‘.value’함수를 사용하여 값을 가져온다.

|  |
| --- |
| OSN\_data\_x |

테이블이(가) 표시된 사진

자동 생성된 설명

‘OSN\_data\_x’를 확인한다. 데이터는 Array 형식으로 편집됐다.

|  |
| --- |
| OSN\_data\_scale = StandardScaler().fit\_transform(OSN\_data\_x)  OSN\_data\_scale |

테이블이(가) 표시된 사진

자동 생성된 설명

‘StandardScaler’ 함수를 사용하여 ‘OSN\_data\_x’ 데이터의 표준화를 진행한다.

#### PCA를 통해 화학적차원에서 정보차원으로 데이터를 투영하라.

1. 다음과 같은 절차를 통해 데이터를 정보차원에 투영할 수 있다.

|  |
| --- |
| from sklearn.decomposition import PCA |

‘sklearn.decomposition’의 ‘PCA’함수를 import하도록 한다.

|  |
| --- |
| pca = PCA(n\_components=8) |

‘PCA’함수를 사용하여 8개의 성분을 갖는 주성분을 생성한다.

|  |
| --- |
| principalComponents = pca.fit\_transform(OSN\_data\_scale) |

‘pca.fit\_transform’ 함수를 사용하여 정규화된 ‘OSN\_data\_scale’ 데이터를 정보차원으로 변환한다. 정보 차원으로 변환된 데이터는 ‘principalComponents’로 명명한다.

|  |
| --- |
| principalDf = pd.DataFrame(data = principalComponents  , columns = ['principal component 1', 'principal component 2',  'principal component 3', 'principal component 4',  'principal component 5', 'principal component 6',  'principal component 7', 'principal component 8'])  principalDf |

테이블이(가) 표시된 사진

자동 생성된 설명

‘pd’ 패키지의 ‘DataFrame’ 함수를 사용하여 ‘principalComponents’ 데이터를 ‘principalDf’로 명명하고 열 이름 (Column name)을 수정한다.

|  |
| --- |
| pca.explained\_variance\_ratio\_  sum(pca.explained\_variance\_ratio\_) |



‘.explained\_variance\_ratio’ 함수를 통해 미리 생성해둔 ‘pca’의 각 주성분의 데이터 설명도 (dimension coverage)를 확인할 수 있다.

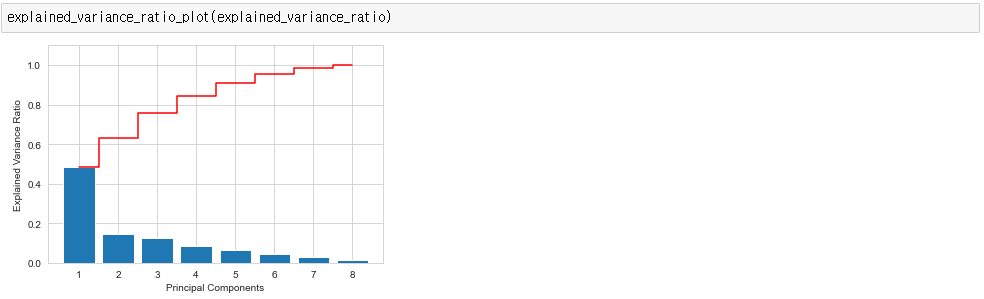
앞에서부터 첫번째 주성분 (PC1)이며 혼자서 전체 데이터의 48.3% 정도 설명할 수 있는 것을 확인한다.

전체 주성분 (PC1~8)의 데이터 설명도 총합은 100%이다.

#### PCA 방법론은 데이터 손실을 최소화하도록 차원수가 선택되어야 한다. 데이터 해석을 위해 최소 몇 개의 차원이 필요한가?

1. 다음과 같은 절차를 통해 각 차원의 데이터 설명력을 확인할 수 있다.

|  |
| --- |
| explained\_variance\_ratio = pca.explained\_variance\_ratio\_  **def** explained\_variance\_ratio\_plot(explained\_variance\_ratio):  x\_axis = range(1, len(explained\_variance\_ratio)+1)  plt.bar(x\_axis, explained\_variance\_ratio,  align = 'center', label = 'Individual Explained Variance Ratio')  plt.step(x\_axis, np.cumsum(explained\_variance\_ratio),  where = 'mid', color='red',  label='Cumulative Explained Variance Ratio')  plt.ylim(0, 1.1)  plt.xticks(x\_axis)  plt.xlabel('Principal Components')  plt.ylabel('Explained Variance Ratio')  plt.grid("true")  plt.show()  explained\_variance\_ratio\_plot(explained\_variance\_ratio) |
|  |



차원 축소는 전체 데이터의 80~90% 이상의 설명력을 갖는 주성분 개수를 선택하면 된다. 여기서 누적 그래프를 확인하면 최소 4개이상의 주성분이 선택되야 한다는 것을 알 수 있다.

시각화를 위해 함수를 정의하여 사용한다. ‘def’를 사용하여 함수를 정의할 수 있고, ‘plt’ package를 사용해 시각화 한다.

#### 가장 데이터 설명력이 높은 두개의 차원을 선택하여 데이터를 시각화하라.

1. 다음과 같은 절차를 통해 데이터 시각화가 가능하다.

|  |
| --- |
| PCs\_12 = principalDf.iloc[:,0:2]  PCs\_12 |

텍스트이(가) 표시된 사진

자동 생성된 설명

‘pd’ package의 ‘iloc’ 함수를 활용하여 ‘principalDf’ 데이터의 1~2번째 열을 추출하여 ‘PC\_12’로 명명한다.

* 나노여과 분리막 정보 차원 시각화 (2D visualization)

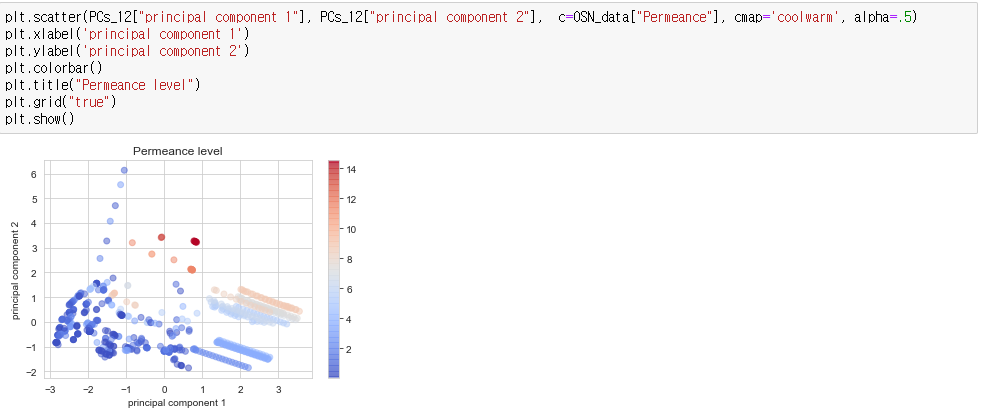
|  |
| --- |
| plt.scatter(PCs\_12["principal component 1"], PCs\_12["principal component 2"], color='grey', alpha=.5)  plt.xlabel('principal component 1')  plt.ylabel('principal component 2')  plt.grid("true")  plt.show() |

텍스트이(가) 표시된 사진

자동 생성된 설명

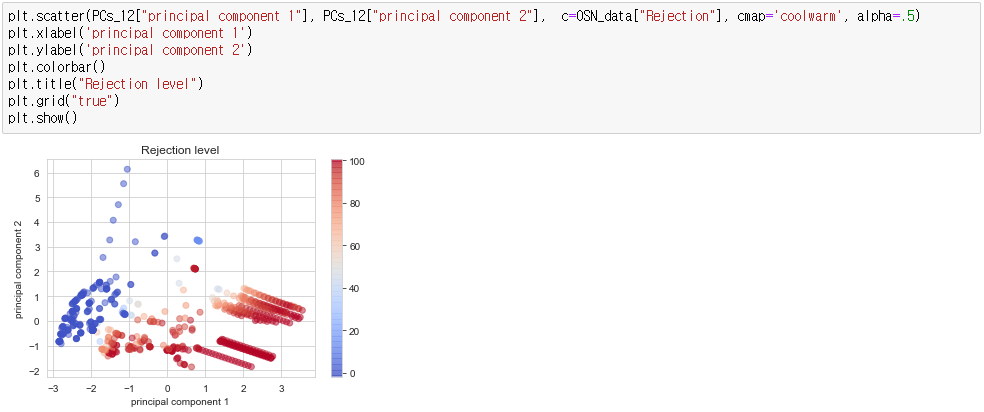
‘plt’ package의 ‘scatter’ 함수를 활용하여 시각화한다. ‘PC\_12’의 첫번째 열과 두번째 열을 좌표로 사용한다.

|  |
| --- |
| plt.scatter(PCs\_12["principal component 1"], PCs\_12["principal component 2"],  c=OSN\_data["Permeance"], cmap='coolwarm', alpha=.5)  plt.xlabel('principal component 1')  plt.ylabel('principal component 2')  plt.colorbar()  plt.title("Permeance level")  plt.grid("true")  plt.show() |



‘plt’ package의 ‘scatter’ 함수를 활용하여 시각화한다. ‘c‘ 함수 인자 (Argument)를 사용하여 ‘OSN\_data’의 Permeance 열을 색으로 표현할 수 있다.

|  |
| --- |
| plt.scatter(PCs\_12["principal component 1"], PCs\_12["principal component 2"],  c=OSN\_data["Permeance"], cmap='coolwarm', alpha=.5)  plt.xlabel('principal component 1')  plt.ylabel('principal component 2')  plt.colorbar()  plt.title("Rejection level")  plt.grid("true")  plt.show() |



‘plt’ package의 ‘scatter’ 함수를 활용하여 시각화한다. ‘c‘ 함수 인자 (Argument)를 사용하여 ‘OSN\_data’의 Rejection 열을 색으로 표현할 수 있다.

#### 가장 데이터 설명력이 높은 두개의 차원에대한 변수 기여도를 확인하라.

1. 다음과 같은 절차를 통해 각 차원에대한 변수 기여도를 확인할 수 있다.

|  |
| --- |
| Feature\_importance = pd.DataFrame(data = pca.components\_, columns =  ['principal component 1', 'principal component 2',  'principal component 3', 'principal component 4',  'principal component 5', 'principal component 6',  'principal component 7', 'principal component 8'],  Index = ['Membrane\_MWCO', 'Solute\_MW', 'Solute\_concentration',  'Solvent\_factor', 'Temperature', 'Process\_configuration',  'Permeance', 'Rejection'])  Feature\_importance |

테이블이(가) 표시된 사진

자동 생성된 설명

‘.components\_’ 함수를 사용하면 ‘pca’의 변수들의 주성분에 대한 기여도를 확인할 수 있다.

각 성분의 이름을 열 이름으로, 행은 변수 이름으로 설정하고 데이터를 ‘Feature\_importance’로 명명한다.

|  |
| --- |
| Feature\_importance\_pc1 = Feature\_importance.iloc[:,0]  Feature\_importance\_pc1 |

텍스트이(가) 표시된 사진

자동 생성된 설명

|  |
| --- |
| Feature\_importance\_pc2 = Feature\_importance.iloc[:,0]  Feature\_importance\_pc2 |

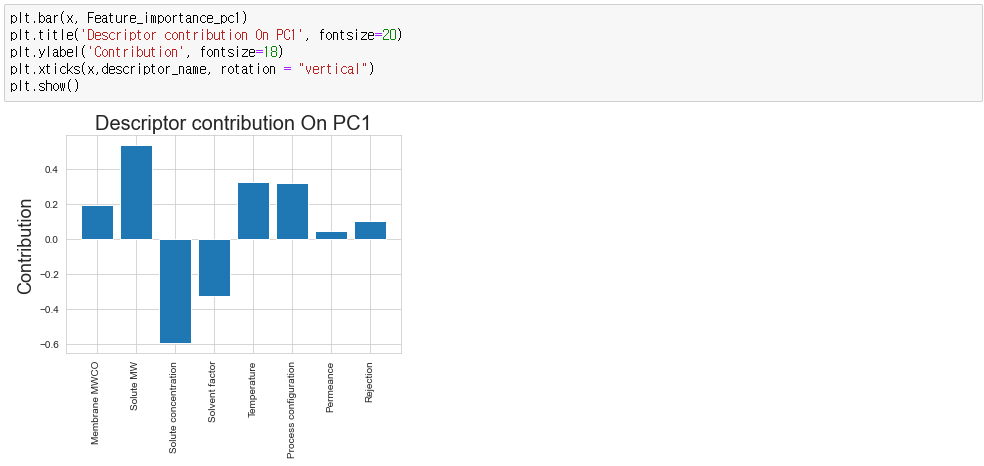
텍스트이(가) 표시된 사진

자동 생성된 설명

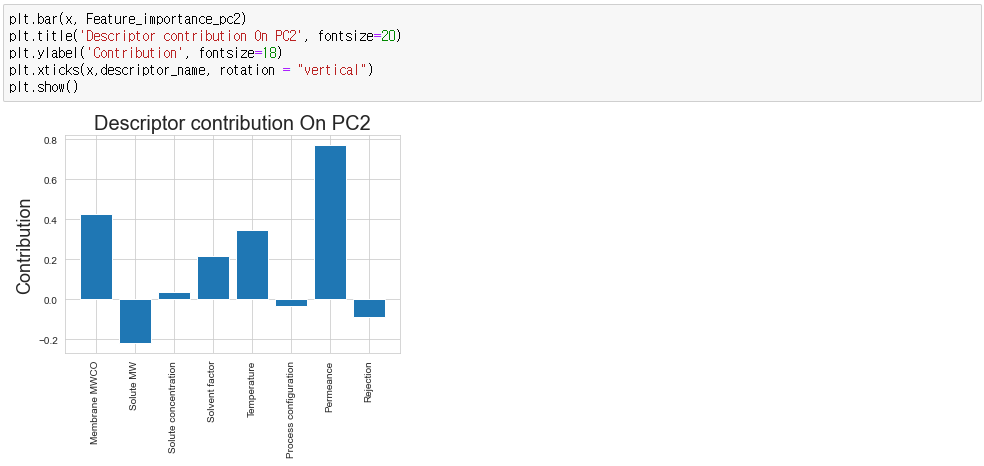
가장 데이터 설명력이 좋은 두 성분 (PC1, PC2)를 추출해 사용한다.

각 PC1과 PC2에대한 기여도 정보를 추출하여 ‘Feature\_importance\_pc1’과 ‘Feature\_importance\_pc2’로 명명한다.

|  |
| --- |
| x = np.arange(8)  descriptor\_name = ['Membrane MWCO','Solute MW','Solute concentration', 'Solvent factor', 'Temperature','Process configuration', 'Permeance', 'Rejection']  plt.bar(x, Feature\_importance\_pc1)  plt.title('Descriptor contribution On PC1', fontsize=20)  plt.ylabel('Contribution', fontsize=18)  plt.xticks(x, descriptor\_name, rotation = "vertical")  plt.show() |
|  |



|  |
| --- |
| plt.bar(x, Feature\_importance\_pc2)  plt.title('Descriptor contribution On PC2', fontsize=20)  plt.ylabel('Contribution', fontsize=18)  plt.xticks(x, descriptor\_name, rotation = "vertical")  plt.show() |



‘np’ package의 ‘array’ 함수를 사용하여 8개의 name space를 갖는 ‘x’ array를 생성한다.

‘plt’ package의 ‘bar’ 함수를 사용하여 ‘Feature\_importance\_pc1’과 ‘Feature\_imporatnce\_pc2’를 시각화한다.

결과로 PC1에서 Solute M.W.와 Solute Concentration, PC2에서는 MWCO가 가장 주요한 변수로 식별되었다.

### **[결론]**

본 장에서는 주성분 분석 방법론을 활용하여 나노여과 막분리 소재 성능의 화학적 차원 탐색을 수행하였다. 화학적 차원에서 정보 차원으로 데이터를 변환하였고, 데이터 설명력을 기반으로 데이터 손실과 주요 차원을 확인하였다. 이를 통해, 막분리 소재 성능에 가장 중요한 변수를 규명하기 위한 변수들의 주성분 기여도를 학습하였다.

### **[학습 결과]**

* 학습 내용

유기용매 나노여과 분리막 개발을 위한 주성분 분석 방법론 익히기

* 학습 결과 확인하기

주성분 분석 알고리즘의 활용 방법 및 변수 기여도 해석 방법 익히기

* 학습 결과 응용하기

본 장의 학습내용에 기반해 나노여과 분리막의 성능 향상을 위한 특성 이해와 주요 원인 규명하기