## 유기용매 막분리 소재 분석

### [공정 설명]

유기용매 나노여과분리막 (Organic solvent nanofiltration, OSN)은 용질의 선택적 분리 기능과 효율적인 물질 투과 능력을 갖고있어 난분해성 폐수의 분리에 탁월한 성능을 보인다. 이로 인해 유기용매 나노여과분리막은 화학, 환경, 의료, 제약, 식품공업 등 다양한 응용이 가능한 까닭에 산업적 수요 기대가 높아지고 있다.

유기용매 나노여과 분리막은 친환경적이고 효율적인 분리 기술이므로, 분리막의 성능 향상을 통해 분리 공정 효율을 개선시키려는 시도가 많이 진행된다. 하지만 분리막 성능은 분리막의 특성 (Property)과 사용 환경 (Operating condition)에 따라 큰 차이가 발생하여 예측이 상당히 어려운 실정이다.

따라서, 고성능 분리막 설계를 위해서는 반복적인 실험에 의존하는 것이 보편적이다. 하지만, 시행착오법은 많은 예산과 시간을 소모하여 다양한 조건에서의 분리막 설계를 지연시키는 한계점을 갖는다. 따라서 새로운 분리막 개발을 통한 분리 공정 개선은 상당히 소극적이게 이루어지며 낮은 성공률을 갖는다. 결국, 분리막 R&D의 효율을 증가시키기 위해서는 분리막 성능 예측이 필수적이다. 특히, 분리막의 주요 변수를 규명하여 정형화된 분리막 성능의 화학적 차원 (Chemical dimension)을 정의한다면, 분리막 성능 예측 모델 개발의 초석이 될 수 있다.

본 실습에서는 차원 축소 방법론 기반 분리막 성능의 경향 파악과 주요 변수 규명을 목적으로한다.

### [문제] ## 추가 필요

### [방법] PCA 예제

#### Q1. PCA을 활용하기 위한 라이브러리를 조사하라.

**A1.** PCA를 활용하기 위한 라이브러리는 대표적이게 sklearn의 sk learn. decomposition이 있다. PCA와 함께 다양한 차원축소 관련 알고리즘을 제공한다.

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

#### Q2. 화학적 차원에서 정보 차원으로의 투영에는 데이터 손실이 필연적으로 발생한다. 데이터 손실의 허용 수준을 조사하라.

**A2.** 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 채택해야한다.

### [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/

#### Q3. 데이터를 Jupyter환경으로부터 불러오고 데이터를 확인하라.

**A3.** 다음과 같은 code를 사용하여 불러올 수 있다.

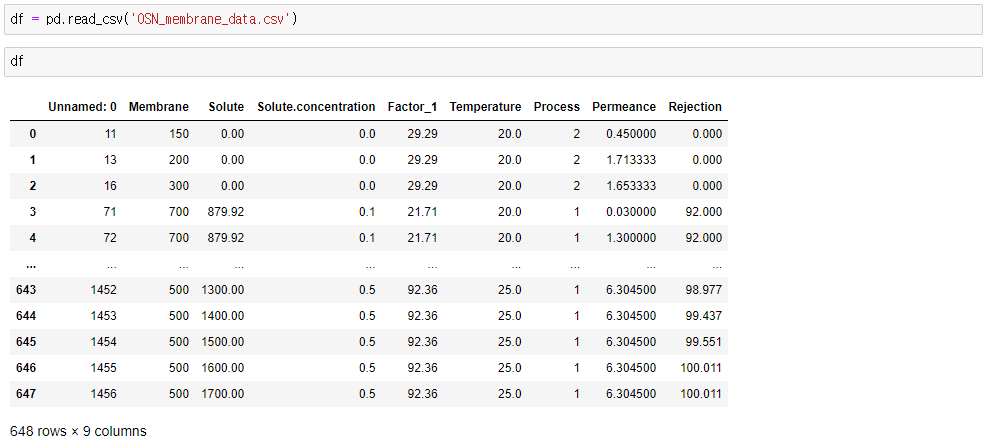
|  |
| --- |
| import pandas as pd  import numpy as np  import matplotlib.pyplot as plt |

‘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.

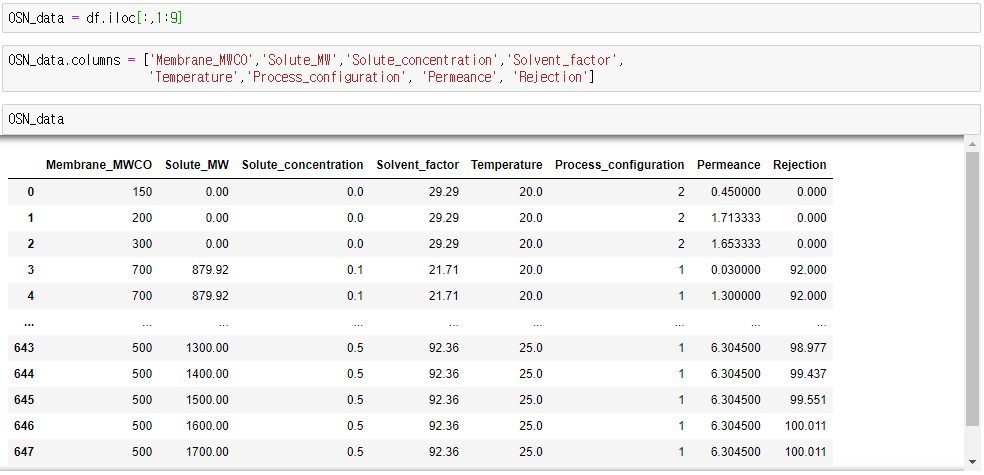
|  |
| --- |
| df = pd.read\_csv('OSN\_membrane\_data.csv')  df |



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 |



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.

#### Q4. 주성분 분석은 데이터의 스케일을 무시한 영향력을 확인이 가능하다. 주성분 분석을 위해 데이터를 표준화하라.

**A4.** 다음과 같은 절차를 통해 표준화한다.

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

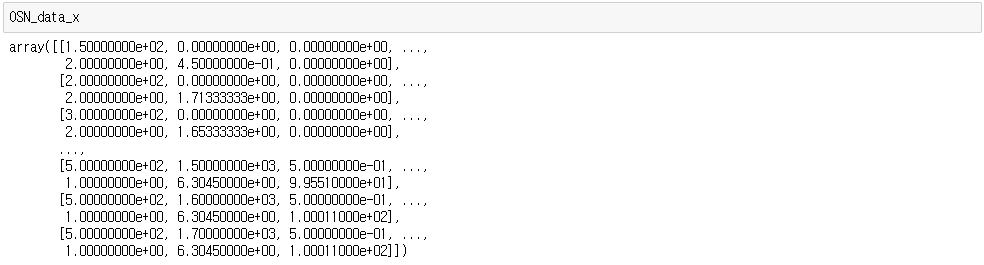
‘sklearn’ package는 계산을 위한 유용한 함수를 다수 내장하고 있다. ‘sklearn.preprocessing’의 ‘StandardScaler’ 함수를 import하도록 한다.

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



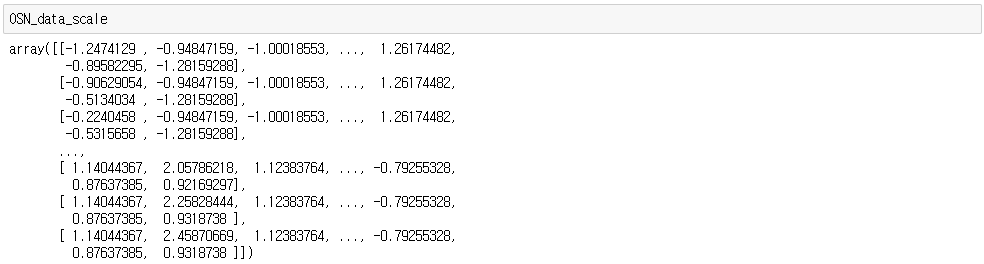
‘OSN\_data’의 값만을 ‘OSN\_data\_x’로 가져온다. 이때 ‘.loc’ 함수를 사용하여 데이터를 지정하고, ‘.value’함수를 사용하여 값을 가져온다.

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



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

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



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

#### Q5. PCA를 통해 화학적차원에서 정보차원으로 데이터를 투영하라.

**A5.** 다음과 같은 절차를 통해 데이터를 정보차원에 투영할 수 있다.

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

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

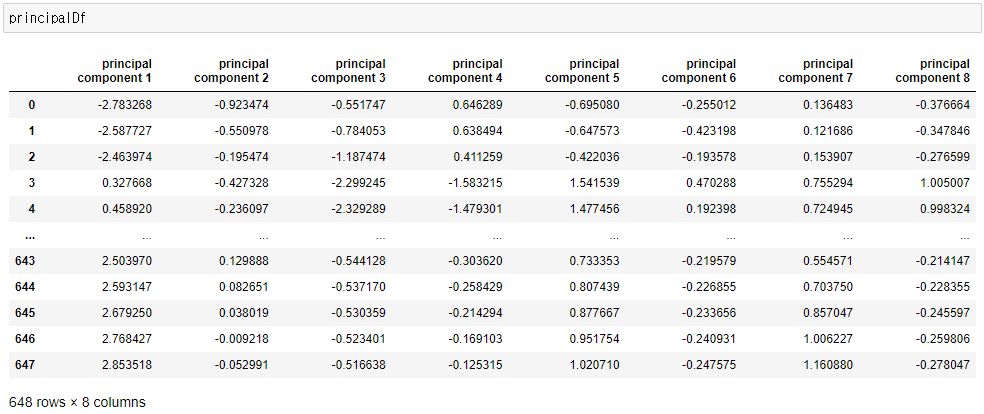
|  |
| --- |
| pca = PCA(n\_components=8) |

‘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 |



‘pd’ 패키지의 ‘DataFrame’ 함수를 사용하여 ‘principalComponents’ 데이터을 ‘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.

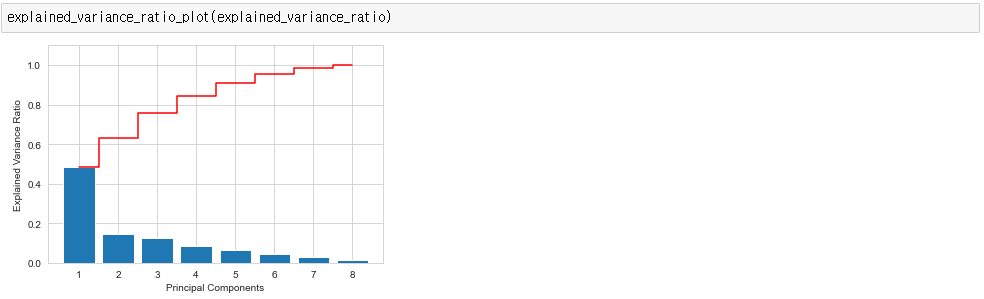
앞에서부터 첫번째 주성분 (PC1)이며 혼자서 전체 데이터의 48.3% 정도 설명할 수 있는 것을 확인한다.

전체 주성분 (PC1~8)의 데이터 설명도 총합은 100%이다.

#### Q6. PCA 방법론은 데이터 손실을 최소화할 차원수가 선택되야한다. 데이터 해석을 위해 최소 몇 개의 차원이 필요한가?

**A6.** 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) |
|  |



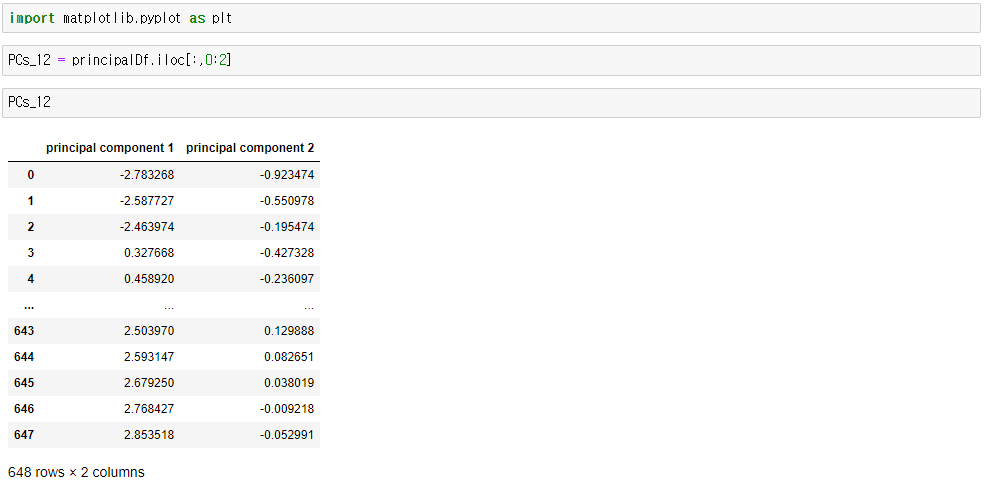
차원 축소는 전체 데이터의 80~90% 이상의 설명력을 갖는 주성분 개수를 선택하면 된다. 여기서 누적 그래프를 확인하면 최소 4개이상의 주성분이 선택되야한다는 것을 알 수 있다.

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

#### Q7. 가장 데이터 설명력이 높은 두개의 차원을 채택해 데이터를 시각화하라.

**A7.** 다음과 같은 절차를 통해 데이터 시각화가 가능하다.

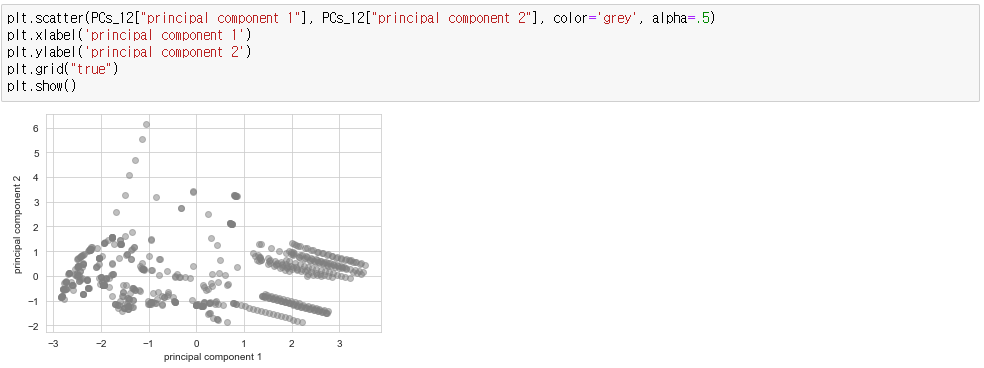
|  |
| --- |
| PCs\_12 = principalDf.iloc[:,0:2]  PCs\_12 |



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

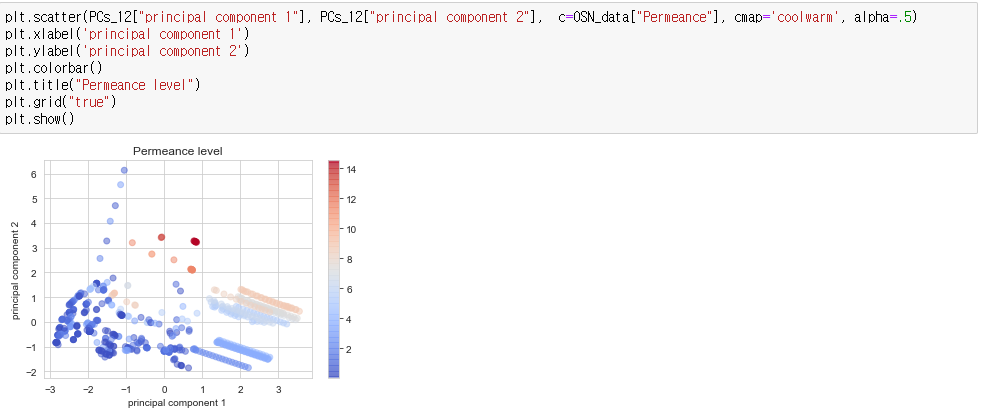
나노여과 분리막 정보 차원 시각화 (2D visualization)

|  |
| --- |
| 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() |



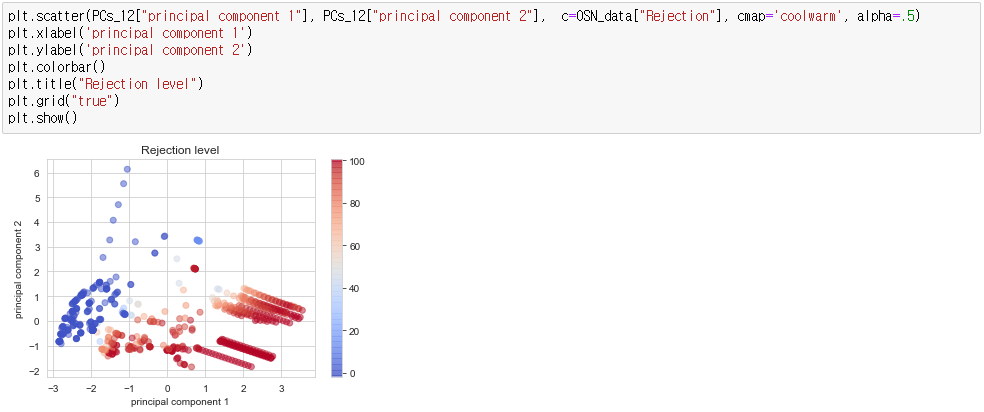
‘plt’ package의 ‘scatter’ 함수를 활용하여 시각화한다. ‘PC\_12’의 첫번째 열과 두번째 열을 좌표로 사용한다.

|  |
| --- |
| 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() |

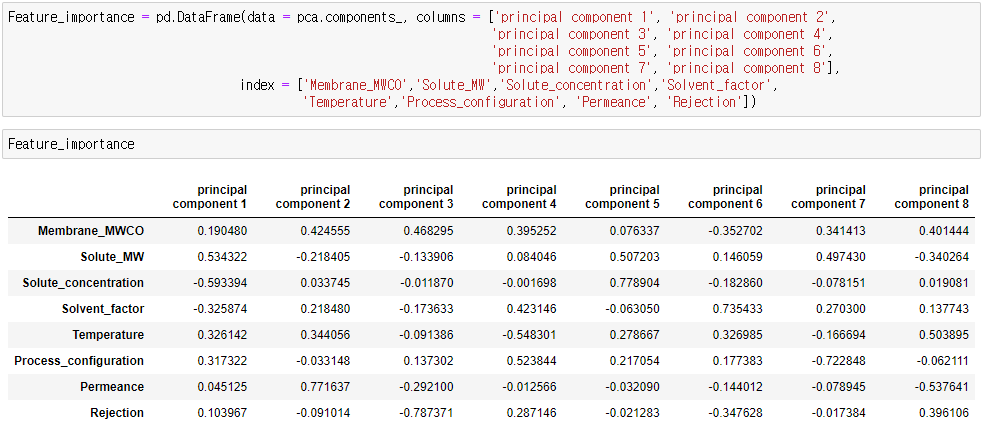


‘plt’ package의 ‘scatter’ 함수를 활용하여 시각화한다. ‘c‘ 함수 인자 (Argument)를 사용하여 ‘OSN\_data’의 Rejection 열을 색으로 표현할 수 있다.

#### Q8. 가장 데이터 설명력이 높은 두개의 차원에대한 변수 기여도를 확인하라.

**A8.** 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 |



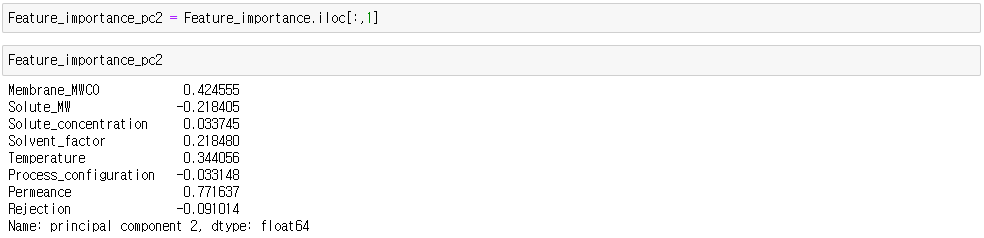
‘.components\_’ 함수를 사용하면 ‘pca’의 변수들의 주성분에 대한 기여도를 확인할 수 있다.

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

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



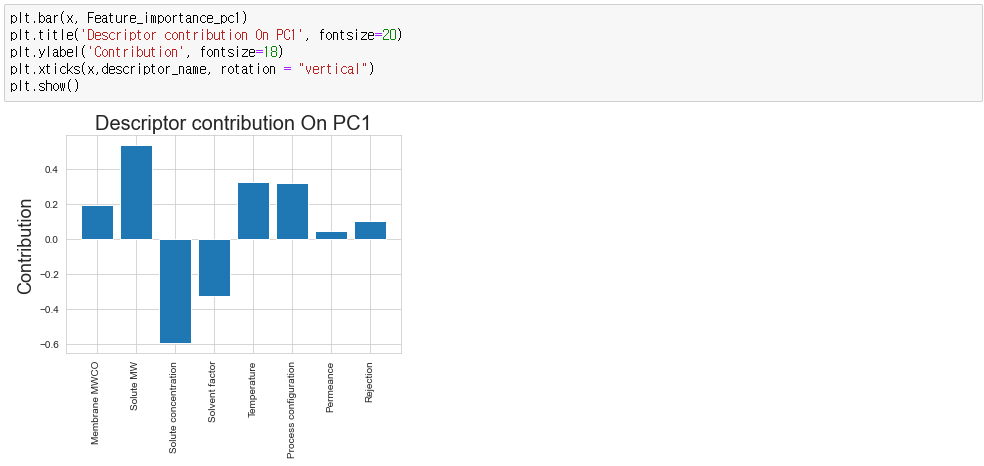
|  |
| --- |
| Feature\_importance\_pc2 = Feature\_importance.iloc[:,0]  Feature\_importance\_pc2 |



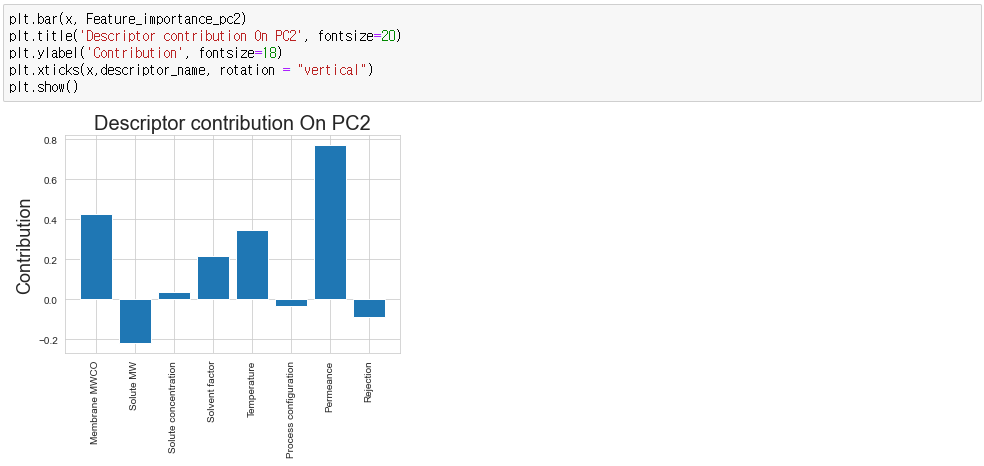
가장 데이터 설명력이 좋은 두 성분 (PC1, PC2)를 추출해 사용한다.

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.W와 Solute Concentration, PC2에서는 MWCO가 가장 주요한 변수로 식별되었다.

### [결론] ## 추가 필요

### [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.