Anomaly Detection

1. 모델 비교

(1) 모델 선택 기준

- ML/DL 선택 뿐 아니라, 비지도/지도 학습까지 굉장히 다양한 모델이 존재

- 아주 좋은 layer를 깔지 않는 이상, 비슷한 부류의 모델(예)PCA&AE)는 비슷한 성능 보임

- 모델 선택에 대한 논거가 필요

- 라벨링 유무 또한 중요하게 작용(라벨링 되어있지 않다면 지도학습은 힘들고, 센서데이터는 시계열로 끊임없이 들어오는데, 이를 라벨링 할 수 있을까에 대한 의문)

- 모델 예시:

1) 예측 기반 접근 방식:

- 예측 모델을 사용하여 예측 수행하고 신뢰 구간 벗어날 시 이상치로 판단

2) 신경망 기반 접근 방식:

- Auto-encoder: 비지도형 신경망으로, 주로 특징 추출 및 차원 축소에 사용, 이상치 탐지 분야에 좋은 성능 보임

- 인코딩 부분: 데이터의 패턴을 나타내는 주요 특징 추출
- 디코딩 부분: 각 샘플 재구성(reconstruction)
- reconstruction error가 높을수록 이상치일 가능성 높음

- SOM(Self-Organizing Maps): 다른 신경망 모델에 비해 작동 원리가 간단

3) 클러스터링 기반 접근 방식

아이디어: 이상치가 어떤 클러스터에도 속하지 않거나 자체적인 클러스터 가지고 있다.

k-means는 가장 잘 알려진 클러스터링 알고리즘 중 하나이며 구현하기 쉬움.

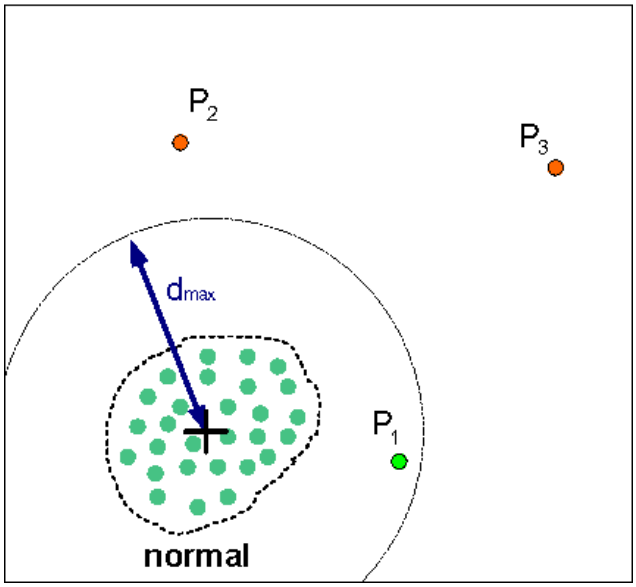
그러나, 적절한 k값을 선택하는 것과 같은 제한 있음

- GMM(Gaussian Mixture Model):

  - 위에서 언급한 k-평균의 약점에 초점을 맞추어 확률적 접근 방식 제시. 데이터 세트 내에서 유한한 가우시안 분포의 혼합을 찾고자 함

- DBSCAN:

  - 밀도 기반 클러스터링 알고리즘
  - 자체적으로 클러스터 번호 결정하고 이상값 샘플은 -1로 할당하여 이상치 탐지
  - 대규모 데이터 셋의 경우 성능 문제 발생



4) 근접 기반 접근 방식

- KNN

- LOF(Local Outlier Factor)

5) 트리 기반 접근 방식

- isolation Forest

6) 차원 축소 기반 접근 방식

- PCA(Principal Component Analysis):

- 고차원 데이터의 차원 축소 방법
- 기본적으로 가장 큰 eigen value를 가진 eigen vector를 추출하여 더 작은 차원의 데이터 분산 대부분을 처리하는데 도움
- 매우 작은 차원으로 데이터에 있는 대부분의 정보 유지할 수 있음

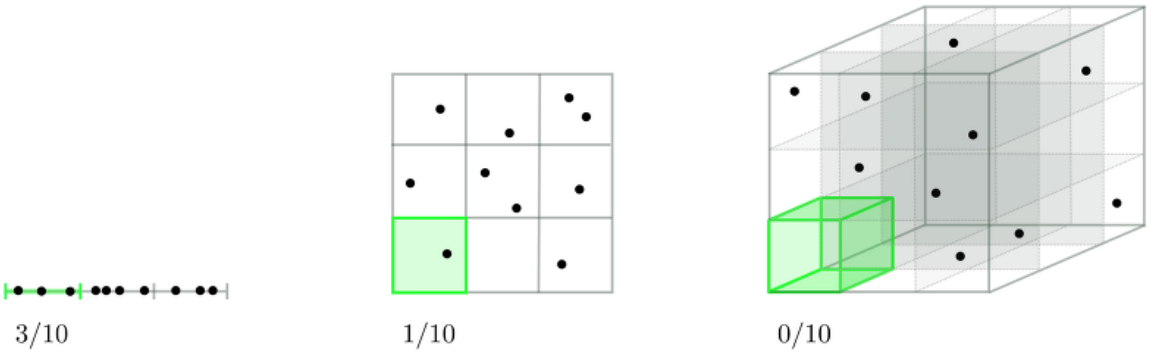
2. PCA (Principal component Analysis)

- 실무에서 분석하는 데이터는 매우 많은 feature 가지고 있음

- 이 데이터가지고 ML시 학습속도도 느리고 성능도 안 좋을 가능성 큼

- 따라서 데이터 차원을 축소시키는 PCA 기법 필요

1) 차원



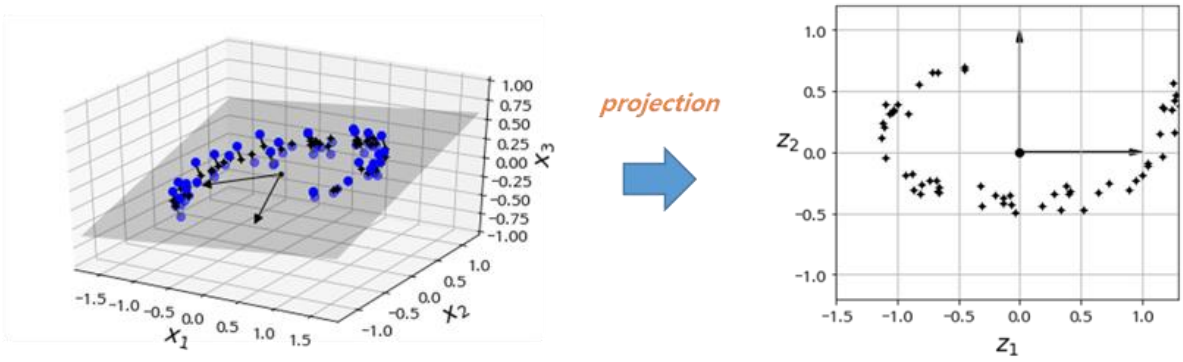
- feature 증가 시, dimension도 증가
- 부피 기하급수적으로 증가
- 이 데이터 사용하여 ML 시, 모델이 복잡해지고 결국 overfitting 위험
- 해결 방법: 데이터의 밀도가 높아질 때까지 학습데이터셋의 크기를 늘리기 -> 매우 힘듬

2) 차원 축소를 위한 접근 방법

- 두가지 방법: 투영(projection)과 매니폴드 학습(manifold learning)

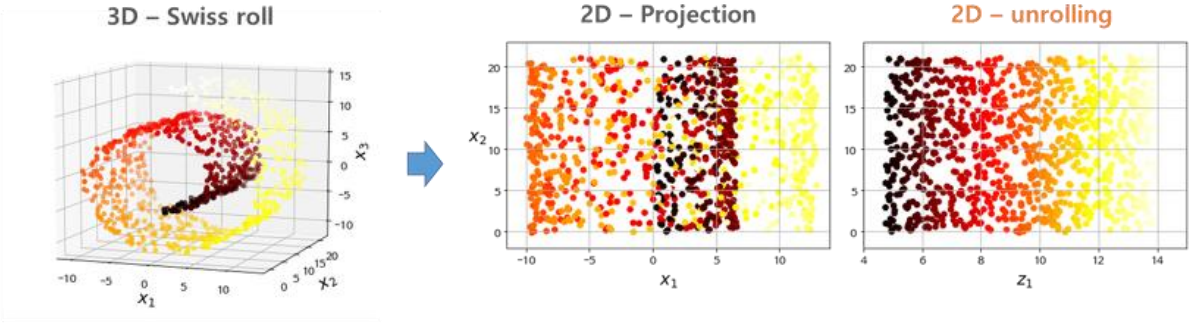
- 투영(Projection):

- 학습 데이터셋은 고차원 공간에서 저차원 부분 공간(subspace)에 위치하게 됨
- 즉, 고차원의 데이터의 특성 중 일부 특성으로 데이터 표현 가능

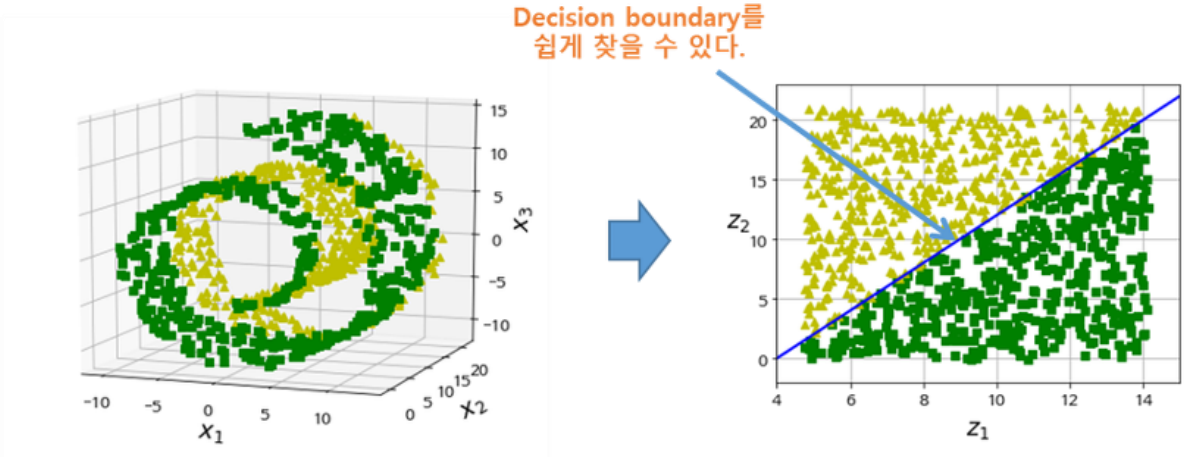


- 매니폴드 학습(Manifold Learning)

- 매니폴드란 유클리드 공간과 닮은 위상 공간
- 고차원(3차원) 공간에서 휘거나 말린 2D 모양



- 대부분 차원 축소 알고리즘이 이러한 매니폴드를 모델링하는 방식으로 동작



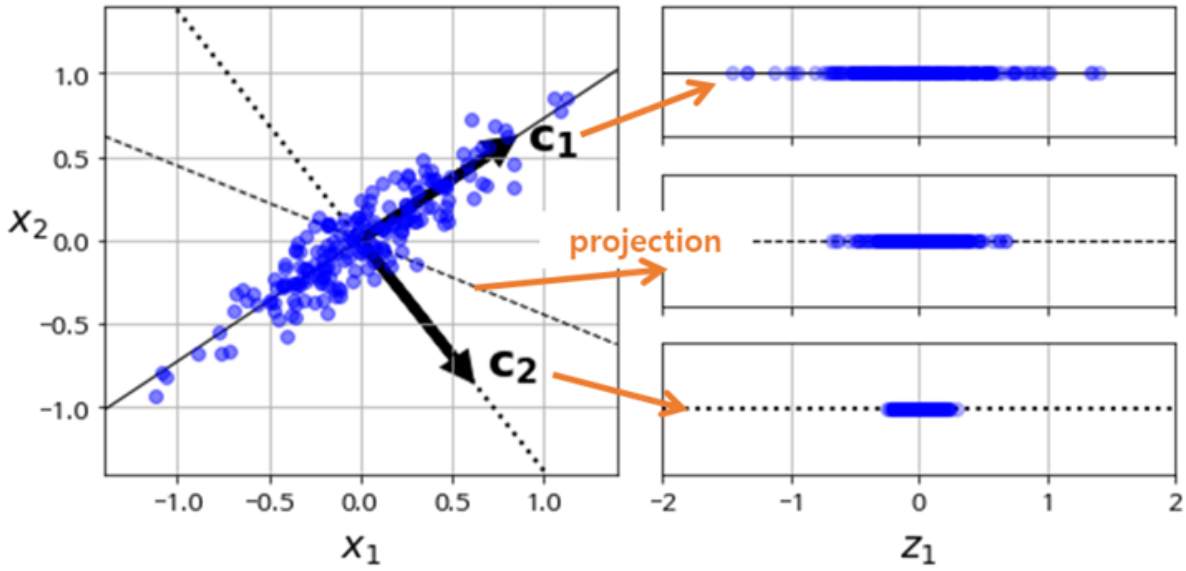
3) PCA

- 대표적인 차원 축소 알고리즘

- 먼저 데이터에 가장 가까운 초평면을 구한 뒤, 데이터를 이 초평면에 투영시킴

3-1) 분산 보존:

- 적절한 초평면 선택
- 원본 데이터셋과 투영된 데이터셋 간의 평균제곱거리 최소화하는 축 찾음



3-2) 주성분

- 학습데이터셋에서 분산이 최대인 축 찾기
- 찾은 첫번째 축과 직교하면서 분산이 최대인 두 번째 축 찾기
- 첫 번째 축과 두 번째 축에 직교하고 분산을 최대한 보존하는 세 번째 축 찾기
- 반복하며 데이터셋의 차원만큼 축 찾기

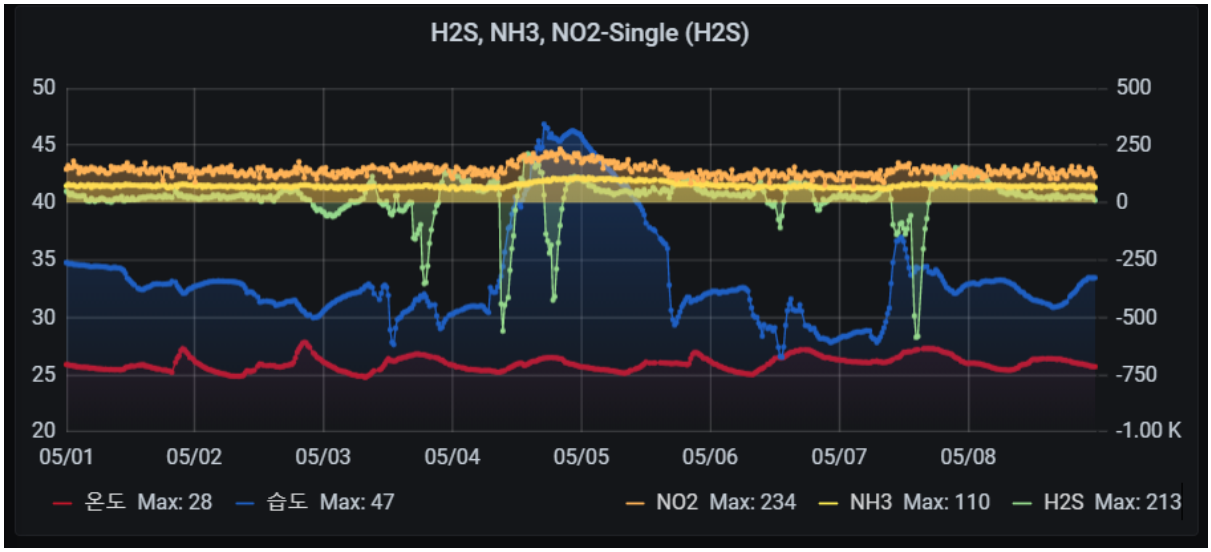
3-3) PCA 구하기(계산 과정은 생략하겠음)

- 공분산 구하기
- PCA 계산
- Scikit-Learn에서 PCA 계산

4) 센서 데이터 셋 적용 (온도, 습도, H2S, NO2, NH3)

- 2021.05.01 00:00:00 ~ 2021.05.08 23:50:00 시간으로 잡음

-



- Grafana에서 csv 추출

- Colab에 csv import

```
dataset = pd.read_csv('/content/drive/MyDrive/Sensor/H2S, NH3, NO2-Single (H2S)-data-as-seriestocolumns-2021-05-17
merged_data = pd.DataFrame()
merged_data=merged_data.append(dataset)
```

```
dataset
#merged_data
```

	Time	온도	습도	H2S	NH3	NO2
0	2021-05-01 00:00:00	26	35	39	69	146
1	2021-05-01 00:10:00	26	35	49	73	143
2	2021-05-01 00:20:00	26	35	43	80	161
3	2021-05-01 00:30:00	26	35	27	71	149

```
df_date_idx = pd.read_csv('/content/drive/MyDrive/Sensor/H2S, NH3, NO2-Single (H2S)-data-as-seriestocolumns-2021-05-

                          sep=",",

                          names=['time','temp', 'moist','H2S','NH3','NO2'],

                          index_col=['time'], # or index_col=0

                          parse_dates=True,

                          dayfirst=True,

                          infer_datetime_format=True)
df_date_idx
```

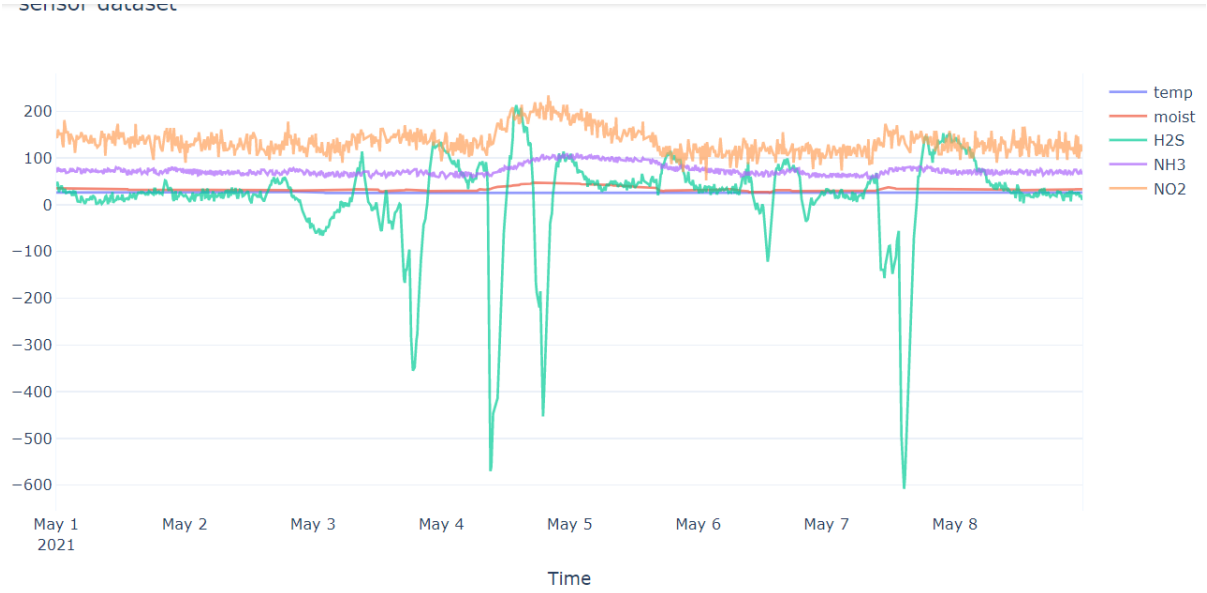
	temp	moist	H2S	NH3	NO2
time					
Time	온도	습도	H2S	NH3	NO2
2021-05-01 00:00:00	26	35	39	69	146
2021-05-01 00:10:00	26	35	49	73	143
2021-05-01 00:20:00	26	35	43	80	161
2021-05-01 00:30:00	26	35	27	71	149
...	...	...	...	...	...
2021-05-08 23:00:00	26	33	21	65	146
2021-05-08 23:10:00	26	33	27	65	99
2021-05-08 23:20:00	26	33	16	71	120
2021-05-08 23:30:00	26	33	21	73	129
2021-05-08 23:50:00	26	33	10	65	114

925 rows × 5 columns

- **Time 열을 index로 바꾸는 작업**
- **'time' 'temp' 'moist' 'H2S' 'NH3' 'NO2' 열로 이름 변환**

```
fig2 = go.Figure()
fig2.add_trace(go.Scatter(x=df_date_idx.index, y=df_date_idx['temp'], name='temp',opacity=0.7))
fig2.add_trace(go.Scatter(x=df_date_idx.index, y=df_date_idx['moist'], name='moist',opacity=0.7))
fig2.add_trace(go.Scatter(x=df_date_idx.index, y=df_date_idx['H2S'], name='H2S',opacity=0.7))
fig2.add_trace(go.Scatter(x=df_date_idx.index, y=df_date_idx['NH3'], name='NH3',opacity=0.7))
fig2.add_trace(go.Scatter(x=df_date_idx.index, y=df_date_idx['NO2'], name='NO2',opacity=0.7))

fig2.update_layout(title='sensor dataset',hovermode='x',template='plotly_white',xaxis=dict(mirror=
                   yaxis=dict(mirror=True,linewidth=0.5,linecolor='aliceblue',title=''))
fig2.show()
```



- **센서 데이터 plotting**
- **grafana와 거의 유사함 확인(온도, 습도 그래프만 grafana와 다름을 볼 수 있는데, 이는 그라파나에서는 온도, 습도 y축 값을 다른 요소들과 다르게 설정했기 때문)**

```
dataset_train = df_date_idx['2021-05-01 00:00:00': '2021-05-04 00:00:00']
dataset_test = df_date_idx['2021-05-04 00:00:00':]
#dataset_train.plot(figsize = (12,6))
```

- **Train/test 셋 구분**
- **임의로 시간 잡았고 시간 잡는 기준에 대해서는 더 연구해야 함**

```
from sklearn.decomposition import PCA
pca = PCA(n_components=2, svd_solver= 'full')
X_train_PCA = pca.fit_transform(X_train)
X_train_PCA = pd.DataFrame(X_train_PCA)
X_train_PCA.index = X_train.index

X_test_PCA = pca.transform(X_test)
X_test_PCA = pd.DataFrame(X_test_PCA)
X_test_PCA.index = X_test.index
```

- **PCA train/test에 각각 fitting**
- **여기서 components는 5개의 열을 몇 개로 줄일 건지 설정하는 것**

- 즉, 모델의 변수 개수 조정

```
def MahalanobisDist(inv_cov_matrix, mean_distr, data, verbose=False):
    inv_covariance_matrix = inv_cov_matrix
    vars_mean = mean_distr
    diff = data - vars_mean
    md = []
    for i in range(len(diff)):
        md.append(np.sqrt(diff[i].dot(inv_covariance_matrix).dot(diff[i])))
    return md
```

- Mahalanobis Distance 정의: 정상군과 비정상군의 통계적 상대 거리

```
[77] def MD_detectOutliers(dist, extreme=False, verbose=False):
        k = 4. if extreme else 2.
        threshold = np.mean(dist) * k
        outliers = []
        for i in range(len(dist)):
            if dist[i] >= threshold:
                outliers.append(i)  # index of the outlier
        return np.array(outliers)
```

```
[78] def MD_threshold(dist, extreme=False, verbose=False):
        k = 4. if extreme else 2.
        threshold = np.mean(dist) * k
        return threshold
```

- threshold 정의
- extremeness에 따라 k 조절
- k 조절하는 부분에 대해서는 더 공부해야 함

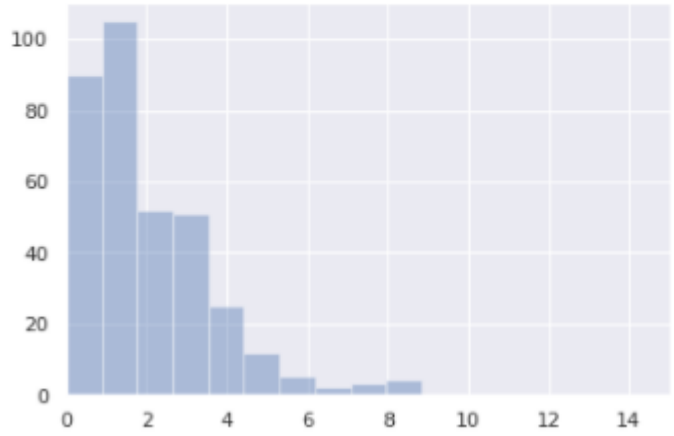
```
data_train = np.array(X_train_PCA.values)
data_test = np.array(X_test_PCA.values)
```

```
dist_test = MahalanobisDist(inv_cov_matrix, mean_distr, data_test, verbose=False)
dist_train = MahalanobisDist(inv_cov_matrix, mean_distr, data_train, verbose=False)
threshold = MD_threshold(dist_train, extreme = True)
```

- data_train/test 정의
- dist_test/train 정의

```
plt.figure()
sns.distplot(np.square(dist_train),
             bins = 10,
             kde= False);
plt.xlim([0.0,15])
```

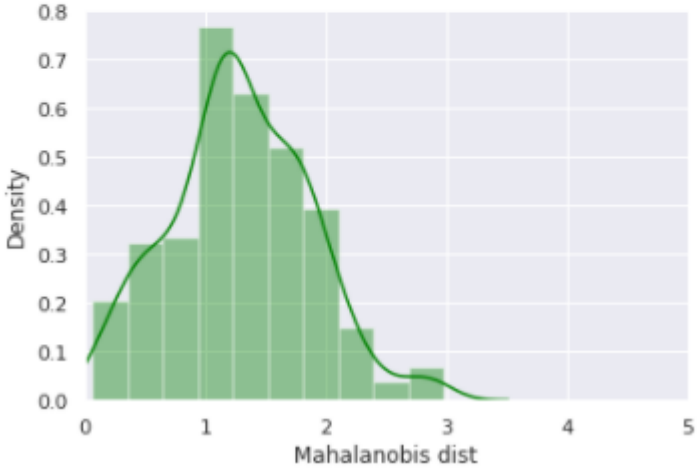
```
(0.0, 15.0)
```



- **dist_train 거리 plotting**

```
plt.figure()
sns.distplot(dist_train,
             bins = 10,
             kde= True,
            color = 'green');
plt.xlim([0.0,5])
plt.xlabel('Mahalanobis dist')
```

```
Text(0.5, 0, 'Mahalanobis dist')
```



- **dist_train Mahalanobis distance 시각화**

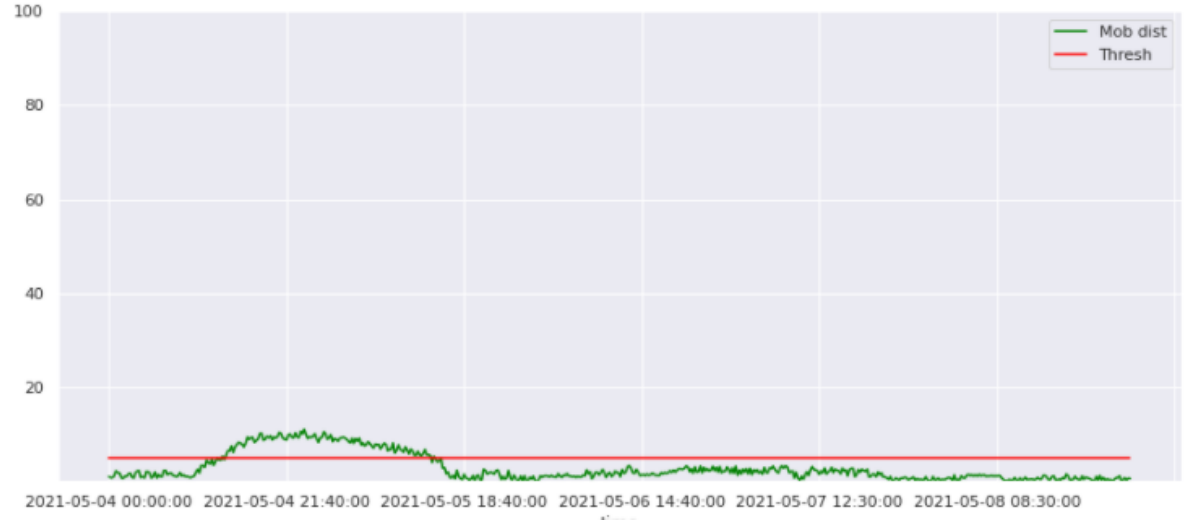
```
anomaly = pd.DataFrame()
anomaly['Mob dist']= dist_test
anomaly['Thresh'] = threshold
# If Mob dist above threshold: Flag as anomaly
anomaly['Anomaly'] = anomaly['Mob dist'] > anomaly['Thresh']
anomaly.index = X_test_PCA.index
anomaly.head()
```

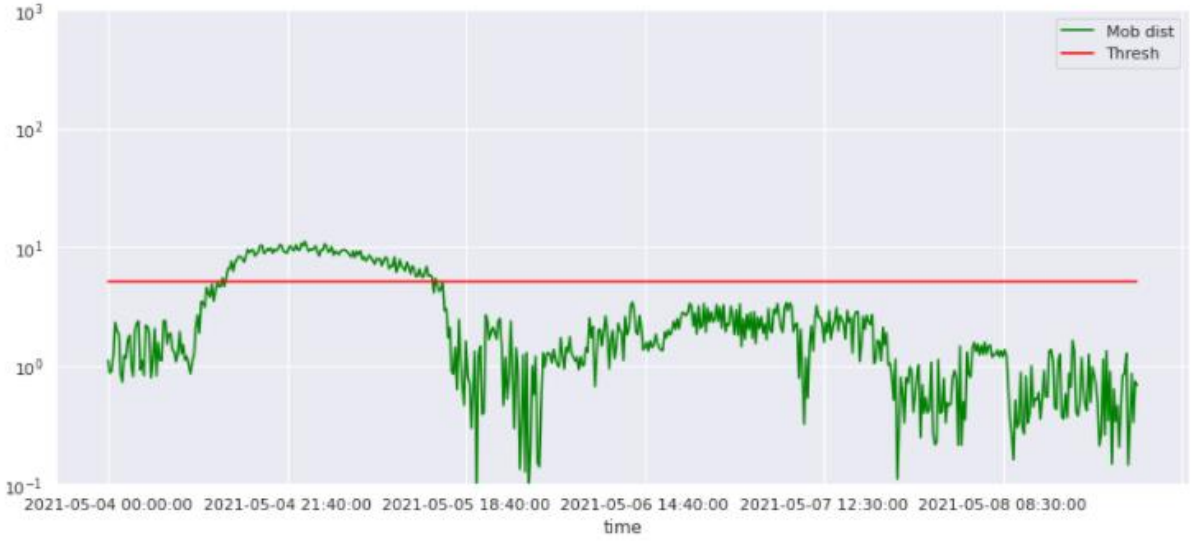
time	Mob dist	Thresh	Anomaly
2021-05-04 00:00:00	1.109411	5.152832	False
2021-05-04 00:10:00	0.866114	5.152832	False
2021-05-04 00:20:00	0.905503	5.152832	False
2021-05-04 00:30:00	1.224057	5.152832	False
2021-05-04 00:50:00	2.307125	5.152832	False

- **dist_test(Mahalanobis distance)와 Threshold 값 비교**
- **Threshold 보다 높으면 Abnormal data, 낮으면 정상**

```
#verifying PCA model in test data
anomaly_alldata.plot(figsize = (14,6), ylim = [1e-1,1e2], color = ['green','red'])
anomaly_alldata.plot(logy=True, figsize = (14,6), ylim = [1e-1,1e3], color = ['green','red'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5369b694d0>
```





- **PCA 모델 시각화**

- **빨간선: Threshold, 초록선: Mahalanobis distance**

```
anomaly_alldata = pd.concat([anomaly])
anomaly_alldata.to_csv('/content/drive/MyDrive/Sensor/Anomaly_distance.csv')
```

- **anomaly data csv로 추출**

2021-05-04 13:00	4.998257	5.152832	FALSE
2021-05-04 13:20	4.673592	5.152832	FALSE
2021-05-04 13:30	4.625418	5.152832	FALSE
2021-05-04 13:40	5.473338	5.152832	TRUE
2021-05-04 14:00	4.678669	5.152832	FALSE
2021-05-04 14:20	5.582728	5.152832	TRUE
2021-05-04 14:30	6.590327	5.152832	TRUE
2021-05-04 14:50	6.57587	5.152832	TRUE
2021-05-04 15:00	7.616899	5.152832	TRUE
2021-05-04 15:10	6.243442	5.152832	TRUE
2021-05-04 15:30	7.400259	5.152832	TRUE
2021-05-04 15:40	7.872987	5.152832	TRUE
2021-05-04 16:00	8.371664	5.152832	TRUE
2021-05-04 16:10	8.254558	5.152832	TRUE

참고: https://blog.naver.com/ppsy17/222207023137

https://excelsior-cjh.tistory.com/167

https://towardsdatascience.com/machine-learning-for-anomaly-detection-and-condition-monitoring-d4614e7de770

https://towardsdatascience.com/lstm-autoencoder-for-anomaly-detection-e1f4f2ee7ccf