< Gaussian density estimation(GDE) 가우시안 밀도 추정 >

**정상 데이터가 정규 분포를 따른다고 가정**

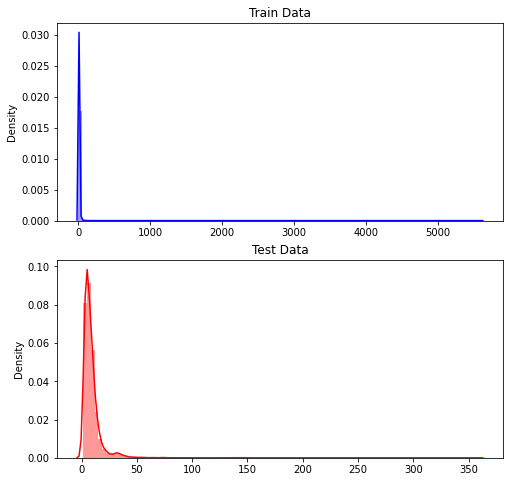
from scipy.stats import multivariate\_normal 이용

1. train 데이터의 가우시안 분포를 추정하기 위해 mu(평균 벡터)와 sigma(공분산 행렬)를 계산
2. 추정된 분포로 train & test 데이터의 -logpdf(-log likelihood) 계산

확률값이 높게 되면 -logpdf 값이 굉장히 낮아지고,

확률값이 낮아지면 -logpdf 값이 굉장히 높아집니다.(양수) : 이상치

시각화 하면(distplot)



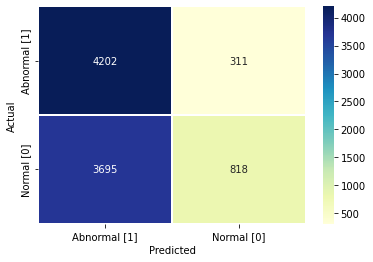
1. -log likelihood의 최솟값과 최댓값 사이에 1000개의 임의의 데이터를 생성하여 threshold 라고 놓고 각각의 샘플의 -log likelihood 가 threshold보다 클 때 f1 score를 구하여 f1 score 가 최대가 될 때의 Best threshold, Best F1 Score 도출하기
2. Best threshold를 기반으로 -log likelihood 가 크면 이상으로(1),

작으면 정상(0) 으로 구분

1. 결과

test 데이터에 대한 평가

FRR: 0.8187 | FAR: 0.0689 | F1 Score: 0.6772

****

* train 데이터도 확인해본 결과 train 데이터가 모두 정상(0) 임에도 불구하고

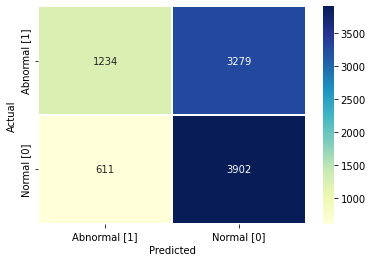
FRR: 0.8201 임을 확인

1. threshold 변경

4번에서 이상이라고 레이블링 한 개수가 실제 이상 개수 4513 개와 비슷할 때의 threshold를 구해봅니다.

* gauss\_best\_threshold = 11
* 이상치 비율 15%
* test 데이터에 대한 평가

FRR: 0.1354 | FAR: 0.7266 | F1 Score: 0.3882



<Mixture of Gaussian Density Estimation(MOG) 혼합 가우시안 밀도 추정

또는 Gaussian Mixture Model(GMM) 가우시안 혼합 모델>

from sklearn.mixture import GaussianMixture이용

GaussianMixture

* n\_components: 가우시안 분포 개수. 디폴트 1
* covariance\_type: 공분산 유형 ('full', 'tied', 'diag', 'spherical'). 디폴트 'full'
* max\_iter: EM 알고리즘 수행 횟수. 디폴트 100

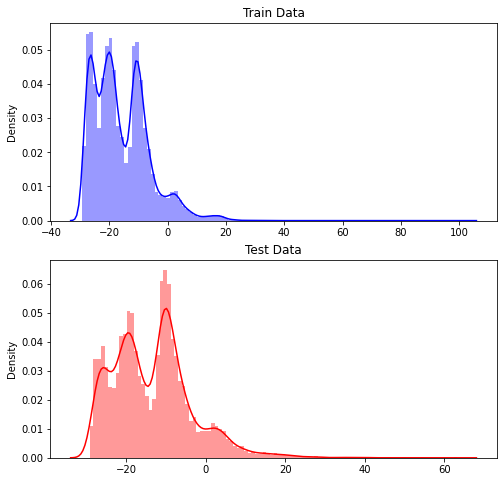
1. 하이퍼파라미터인 n\_components와 covariance\_type을 구하고 모델에 적합시키기

* covariance : full

n\_components : 6

1. 추정된 분포로 train & test 데이터의 -logpdf(-log likelihood) 계산

시각화 하면(distplot)



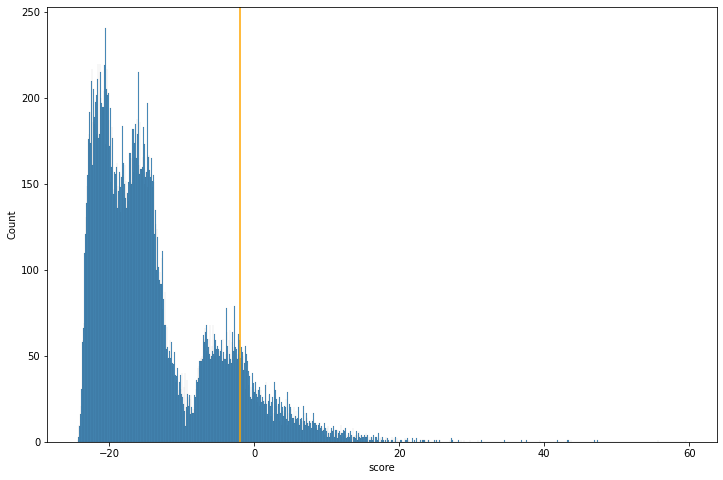
1. threshold 설정(퍼센트) -> 이상치 15% 이상 값

=> -2.02

-log likelihood 보다 작으면 정상으로(0),

크면 이상(1) 으로 구분

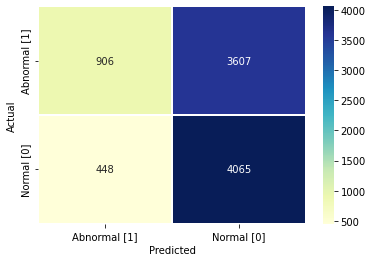
threshold(노란색 수직선)와 novelty score 시각화하면(histplot)



1. 결과

test 데이터에 대한 평가

FRR: 0.0993 | FAR: 0.7992 | F1 Score: 0.3088



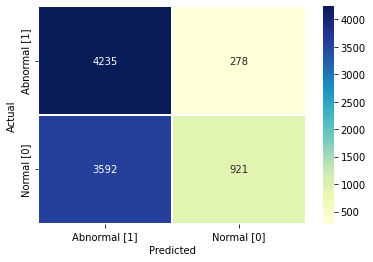
1. -log likelihood의 최솟값과 최댓값 사이에 1000개의 임의의 데이터를 생성하여 threshold 라고 놓고 각각의 샘플의 -log likelihood 가 threshold보다 클 때 f1 score를 구하여 f1 score 가 최대가 될 때의 Best threshold, Best F1 Score 도출하기
2. Best threshold를 기반으로 -log likelihood 가 크면 이상으로(1),

작으면 정상(0) 으로 구분

1. 결과

test 데이터에 대한 평가

FRR: 0.7959 | FAR: 0.0616 | F1 Score: 0.6864



1. threshold 변경

6번에서 이상이라고 레이블링 한 개수가 실제 이상 개수 4513 개와 비슷할 때의 threshold를 구해봅니다.

* gmm\_best\_threshold = -3
* test 데이터에 대한 평가
* FRR: 0.117 | FAR: 0.7682 | F1 Score: 0.3437

<Local outlier factor(LOF)> - 모델 기반이라 threshold 변경 못 했음.

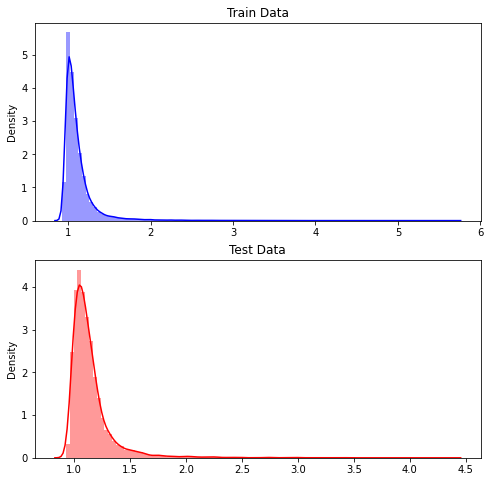
이 모델이 이상 비율을 5%로 잡음

**특정 분포 가정 X**

from sklearn.neighbors import LocalOutlierFactor이용

| LocalOutlierFactor   * n\_neighbors: neighbors 개수. 디폴트 20 * algorithm: nearest neighbors 계산 방법 ('auto', 'ball\_tree', 'kd\_tree', 'brute'}. 디폴트 'auto' * metric: distance 계산 방법. 디폴트 'minkowski' * novelty: True (for novelty detection - no oulier in train data), False (for outlier detection - ouliers in train data). 디폴트 False |
| --- |

* 1. 모델 적합 n\_neighbors=15
  2. 추정된 분포로 train & test 데이터 -log likelihood 계산
  3. train/test 데이터의 -log likelihood 분포 시각화



* 1. 모델 예측
* 모델이 예측을 하면 -1,1 값으로 딱 정상, 이상 나와서 threshold 변경 못 했음.
* 이 모델이 이상 비율을 4%로 잡음

정상=0, 이상=1 각각의 개수

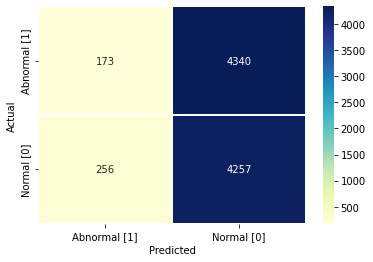
0 20954

1 927

로 이상 비율이 0.042365522599515565

* 1. 결과

FRR: 0.0567 | FAR: 0.9617 | F1 Score: 0.07

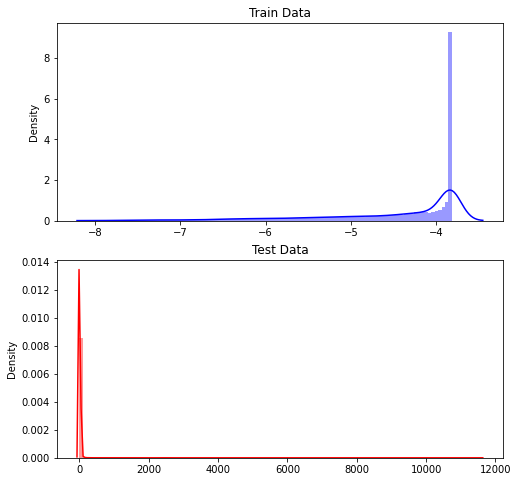


<Parzen Window(PW)>

from sklearn.neighbors import KernelDensity 이용

| KernelDensity   * kernel: 사용할 커널 종류 ('gaussian', 'tophat', 'epanechnikov', 'exponential', 'linear', 'cosine'). 디폴트 'gaussian' |
| --- |

* 1. 모델 적합 n\_neighbors=15
  2. 추정된 분포로 train & test 데이터 -log likelihood 계산
  3. train/test 데이터의 -log likelihood 분포 시각화

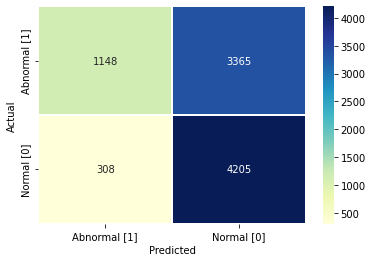


* 1. best f1 score 방식
  2. threshold 변경
     + kde\_best\_threshold = 0

이상치 비율 0.15155142847898534

* 1. 결과

FRR: 0.0682 | FAR: 0.7456 | F1 Score: 0.3847



< Autoencoder>

1. from sklearn.neural\_network import MLPRegressor이용

| * hidden\_layer\_sizes: 뉴런넷 hidden layer의 사이즈를 tuple로 받으며, 순차적으로 연결됨 * activation: non-linear activation 사용 * learninig\_rate\_init: 초기 학습률 * max\_iter: 학습 반복수 * tol: stop 기준 * verbose: 학습 log 출력 |
| --- |

* 1. train 데이터는 독립변수가 20개
* encoder structure로

encoder1\_dim = 400

encoder2\_dim = 400

* hidden layer로

latent\_dim = 40

* decoder structure로

decoder1\_dim = 400

decoder2\_dim = 400

* input과 output이 같도록 모델 적합

ae = MLPRegressor(hidden\_layer\_sizes = (encoder1\_dim, encoder2\_dim, latent\_dim, decoder1\_dim, decoder2\_dim),

                   activation = 'tanh',

                   solver = 'adam',

                   learning\_rate\_init = 0.01,

                   max\_iter = 100,

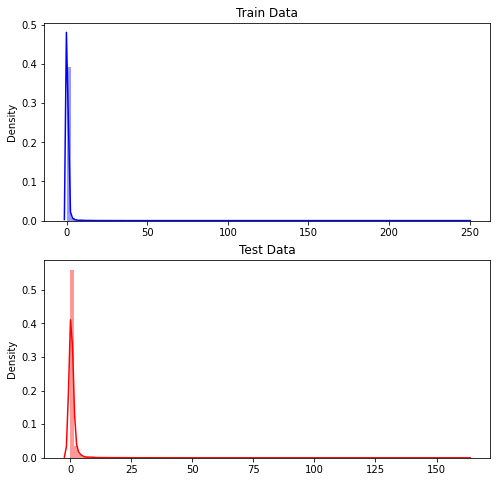
                   tol = 0.0001, # 학습 개선도가 이만큼 되지 않으면 중단한다는 뜻

                   verbose = True)

ae.fit(X\_train, X\_train)

* 1. 로 novelty score 생성

train 데이터, test 데이터 novelty score 시각화 하면



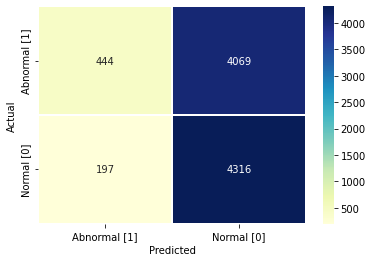
* 1. train 데이터의 실제 값과 예측 값 사이의 차이인 MSE값 상위 5%를 임계치로 설정
  2. test 데이터의 실제 값과 예측 값 사이의 차이인 MSE값을 reconstruction error로 정의
  3. 3 번의 임계치를 기반으로 novelty score 가 크면 이상으로(1)

작으면 정상(0) 으로 구분

* 1. 결과

test 데이터에 대한 평가 임계치:4.034694105996352

FRR: 0.0437 | FAR: 0.9016 | F1 Score: 0.1723



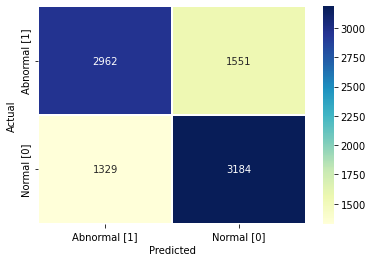
* train 데이터에 대해서도 평가를 해보니

FRR: 0.05 으로 거의 다 맞춤을 확인

* 1. threshold 변경

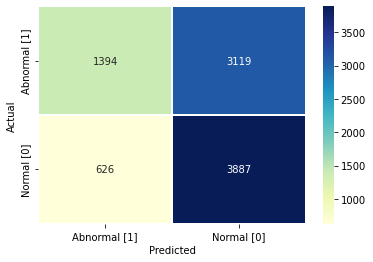
train 데이터의 실제 값과 예측 값 사이의 차이인 MSE값 상위 30%를 임계치로 설정0.32275006582533067

FRR: 0.2945 | FAR: 0.3437 | F1 Score: 0.6729



train 데이터의 실제 값과 예측 값 사이의 차이인 MSE값 상위 15%를 임계치로 설정 1.2114339029982377

FRR: 0.1387 | FAR: 0.6911 | F1 Score: 0.4268



1. import tensorflow as tf 이용
   1. train 기반 모델 적합

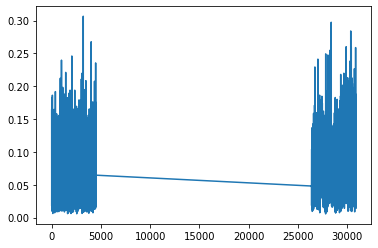
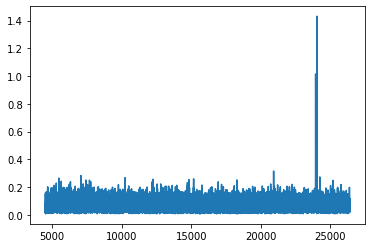
| input\_dim = X\_train.shape[1]  AE = tf.keras.models.Sequential([        # encode      tf.keras.layers.Dense(input\_dim, activation='elu', input\_shape=(input\_dim, )),      tf.keras.layers.Dense(15, activation='elu'),      tf.keras.layers.Dense(5, activation='elu'),        # decode      tf.keras.layers.Dense(15, activation='elu'),      tf.keras.layers.Dense(input\_dim, activation='elu')    ])  AE.compile(optimizer="adam", loss="mse") |
| --- |

* input과 output이 같도록 모델 적합

| history = AE.fit(      X\_train,      X\_train,      epochs=100,      batch\_size=32,      validation\_split=0.3  ) |
| --- |

* 1. 로 novelty score 생성

train 데이터, test 데이터 novelty score 시각화 하면



* 1. train 데이터의 실제 값과 예측 값 사이의 차이인 MSE값 상위 5%를 임계치로 설정
  2. test 데이터의 실제 값과 예측 값 사이의 차이인 MSE값을 reconstruction error로 정의
  3. 3 번의 임계치를 기반으로 novelty score 가 크면 이상으로(1)

작으면 정상(0) 으로 구분

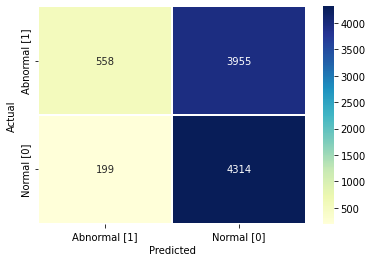
* 1. 결과

임계치

0.23711762220364335에 대해서

test 데이터에 대한 평가

FRR: 0.0441 | FAR: 0.8764 | F1 Score: 0.2118

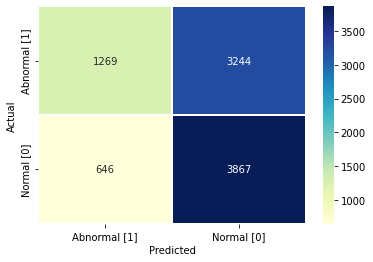


* 1. threshold 변경

train 데이터의 실제 값과 예측 값 사이의 차이인 MSE값 상위 15%를 임계치로 설정

임계치 0.12328756440921747에 대해서

FRR: 0.1431 | FAR: 0.7188 | F1 Score: 0.3948



train 데이터의 실제 값과 예측 값 사이의 차이인 MSE값 상위 30%를 임계치로 설정

임계치 0.08833904900057317에 대해서

FRR: 0.302 | FAR: 0.5349 | F1 Score: 0.5264

