LSTM Auto-Encoder를 통한 불량률 발생 요건 사전 예측

망구리 조
조장 김지호
강창민

목 차

프로젝트 목표
핵심 개념

프로젝트 목표 및 핵심 개념

프로젝트 목표

Austempering 열처리 공정 중 불량품이 발생하는 케이스를 예측하는 모델을 구현하여, 사전에 불량률을 예측. 공정환경을 조절함으로써 소모비용과 생산량을 증가시킨다.

핵심 개념

1) Austempering 열처리 공정

Austemping 열처리 공정은 일종의 담금질 기법으로, 열처리를 가한 austenite 상태의 제품을 공기중에서 급랭시키는 것이 아닌, 300℃ ~ 350℃로 가열된 염욕시설에서 천천히 냉각하며 제품의 변형을 막고 제품의 신축성, 단면수축률, 충격치 등이 우수한 제품을 얻을수 있는 기법으로, 공정 전체과정의 온도 및 유지, 냉각시간과 냉각 온도등이 불량을 결정하는 주요 온도로 볼 수 있다. 이 프로젝트에서는 공정과정의 온도 및 유지의 이상치가 제품 불량률에 영향을 줄 것이라고 판단하여 공정구간 온도의 이상치를 분석하였다.

프로젝트 목표 및 핵심 개념

핵심 개념

2) LSTM Auto-Encoder

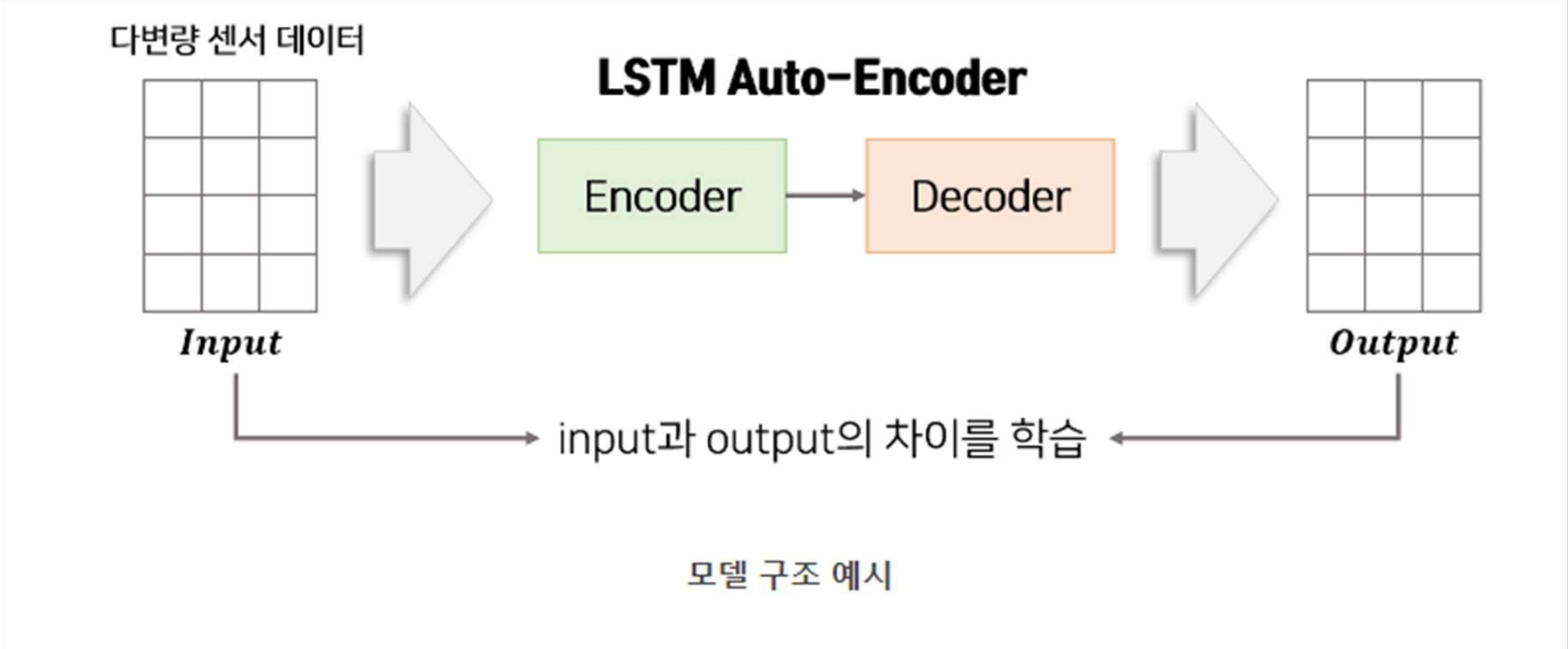
LSTM Auto-Encoder 모델은 RNN 모델중 하나인 LSTM을 활용하여 데이터를 학습, 압축하고 이를 재구성하는 Encoder-Decoder 아키텍처의 일종으로 데이터의 압축된 표현을 학습하는 비지도 학습 모델이다.

LSTM Auto-Encoder 모델은 LSTM-Encoder와 LSTM-Decoder로 구성되어 있으며 LSTM-Encoder에서는 다변량 데이터를 압축하여 feature로 변환하는 역할을 하고 LSTM-Decoder에서는 해당 feature를 다변량 데이터로 재구성 하는 방식으로 input된 다변량 데이터와 output된 다변량 데이터의 차이를 줄이도록 학습된다.

프로젝트 목표 및 핵심 개념

핵심 개념

2) LSTM Auto-Encoder



위와같이 Encoding 이전 데이터와 Decoder 이후 데이터의 차이를 학습하는 방식으로, 시계열데이터 및 시퀀스 데이터 내에서 이상치를 제거하거나, 이상치를 탐지하는데 우수한 모델이다.

데이터 분석 및 정제

데이터 분석 및 탐색

#	Column	Dtype
0	TAG_MIN	object
1	배정번호	int64
2	건조 1존 OP	float64
3	건조 2존 OP	float64
4	건조로 온도 1 Zone	float64
5	건조로 온도 2 Zone	float64
6	세정기	float64
7	소입1존 OP	float64
8	소입2존 OP	float64
9	소입3존 OP	float64
10	소입4존 OP	float64
11	소입로 CP 값	float64
12	소입로 CP 모니터 값	float64
13	소입로 온도 1 Zone	float64
14	소입로 온도 2 Zone	float64
15	소입로 온도 3 Zone	float64
16	소입로 온도 4 Zone	float64
17	솔트 컨베이어 온도 1 Zone	float64
18	솔트 컨베이어 온도 2 Zone	float64
19	솔트조 온도 1 Zone	float64
20	솔트조 온도 2 Zone	float64

프로젝트에서 사용된 data.csv 파일에 대한 분석

nlp.csv는 document 및 label 두가지의 컬럼으로 구성됨

nlp.csv 파일 내 레이블 별 데이터의 갯수는 0 : 7067개, 1:7067개로 동일

특수문자등 한글이 아닌 텍스트가 확인됨

데이터 분석 및 탐색

#	Column	Dtype
0	TAG_MIN	object
1	배정번호	int64
2	건조 1존 OP	float64
3	건조 2존 OP	float64
4	건조로 온도 1 Zone	float64
5	건조로 온도 2 Zone	float64
6	세정기	float64
7	소입1존 OP	float64
8	소입2존 OP	float64
9	소입3존 OP	float64
10	소입4존 OP	float64
11	소입로 CP 값	float64
12	소입로 CP 모니터 값	float64
13	소입로 온도 1 Zone	float64
14	소입로 온도 2 Zone	float64
15	소입로 온도 3 Zone	float64
16	소입로 온도 4 Zone	float64
17	솔트 컨베이어 온도 1 Zone	float64
18	솔트 컨베이어 온도 2 Zone	float64
19	솔트조 온도 1 Zone	float64
20	솔트조 온도 2 Zone	float64

프로젝트에서 사용된 data.csv 파일에 대한 분석

nlp.csv는 document 및 label 두가지의 컬럼으로 구성됨

nlp.csv 파일 내 레이블 별 데이터의 갯수는 0 : 7067개, 1:7067개로 동일

특수문자등 한글이 아닌 텍스트가 확인됨

데이터 분석 및 탐색

순번	컬 럼 명	데이터 타입	최댓값	최솟값	평균	표준편차	결측치
1	TAG_MIN	object	-	-	-	-	-
2	배정번호	int64	148069	102410	-	-	-
3	건조 1존 OP	float64	87.2995	47.2532	69.8940	4.0148	1
4	건조 2존 OP	float64	47.5395	0.0001	20.4471	5.2171	1
5	건조로 온도 1 Zone	float64	102.4690	97.3421	100.0061	0.4360	116
6	건조로 온도 2 Zone	float64	101.8430	97.8706	100.0198	0.3624	148
7	세정기	float64	71.4901	60.6244	67.7186	1.6308	91
8	소입1존 OP	float64	100.0000	0.0009	75.6437	25.1608	4288
9	소입2존 OP	float64	77.2709	8.6200	54.8624	4.4291	-
10	소입3존 OP	float64	66.0150	0.0437	53.8603	2.6643	2
11	소입4존 OP	float64	87.3907	0.0062	71.0893	2.5570	3
12	소입로 CP 값	float64	0.9091	0.0051	0.4489	0.0189	1
13	소입로 CP 모니터 값	float64	0.0000	0.0000	0.0000	0.0000	147
14	소입로 온도 1 Zone	float64	877.2280	840.2980	859.2077	3.6477	130
15	소입로 온도 2 Zone	float64	866.0340	855.9290	860.0021	0.5578	128
16	소입로 온도 3 Zone	float64	870.1190	858.2800	860.0029	0.3518	157
17	소입로 온도 4 Zone	float64	882.1480	857.9920	860.0062	0.4552	170
18	솔트 컨베이어 온도 1 Zone	float64	298.5300	266.2300	283.9963	9.5128	106
19	솔트 컨베이어 온도 2 Zone	float64	291.6960	266.4260	279.9293	6.6116	142
20	솔트조 온도 1 Zone	float64	332.7170	328.1610	331.8062	0.7827	209
21	솔트조 온도 2 Zone	float64	333.1790	328.0730	332.1773	0.8733	203

프로젝트에서 사용된 data.csv에 대한 info()와 describe() 를 분석한 내용이다. 결측치가 적지 않으며 TAG_MIN 컬럼이 object로 datetime으로 변경해 주어야할 필요성이 확인된다.

데이터 분석 및 탐색

순번	컬 럼 명	데이터 타입	최댓값	최솟값	평균	표준편차	결측치
1	배정번호	int64	-	-	-	-	-
2	작업일	datetime64[ns]	-	-	-	-	-
3	공정명	object	-	-	-	-	-
4	설비명	object	-	-	-	-	-
5	양품수량	int64	-	-	-	-	-
6	불량수량	int64	-	-	-	-	-
7	총수량	int64	-	-	-	-	-

프로젝트에서 사용된 quality.xlsx에 대한 info()와 describe() 를 분석한 내용이다. 결측치가 없으며 배정번호별로 작업일, 공정명, 설비명, 양품수량, 불량수량, 총 수량등이 기재되어있다.

데이터 품질지수 측정

```
def data_quality_idx(df):

    def completeness_quality_idx(df):
        null_data = df.isnull().sum().sum()
        total_data = df.size
        completenss_idx = (1-(null_data/total_data))
        return completenss_idx

    def uniqueness_quality(df):
        unique_data = df.groupby('TAG_MIN').size().count()
        total_data = len(df['TAG_MIN'])
        uniqueness_idx = (unique_data/total_data)
        return uniqueness_idx

    def uniqueness_quality2(df):
        unique_data = df.groupby('배정번호').size().count()
        total_data = len(df['배정번호'])
        uniqueness_idx = (unique_data/total_data)
        return uniqueness_idx

    def validity_quality(df):
        validity_range = [df.min(), df.max()]
        valid_data = ((df >= validity_range[0])
                      & (df <= validity_range[1])).sum().sum()
        total_data = df.size
        validity_idx = (valid_data / total_data)
        return validity_idx

    def accuracy_quality(df):
        defective = df['불량수량'].sum()
        normal = df['양품수량'].sum()
        total = df['총수량'].sum()
        accuracy_idx = 1 - (total-(normal+defective))/total
        return accuracy_idx

    def consistency_quality(df):
        consistency_count=0
        for i in range(len(df.columns)):
            if df.dtypes[i] == df[df.columns[i]].dtypes:
                consistency_count += 1
        consistency_idx = consistency_count/len(df.columns)
        return consistency_idx

    def intergrity_quality(uniqueness_idx, validity_idx, consistency_idx):
        lst = [uniqueness_idx, validity_idx, consistency_idx]
        count = 0
        for i in lst:
            if i != 1:
                count += 1
        if count == 0:
            intergrity_idx = 1
        elif count == 1:
            intergrity_idx = (1 - (sorted(lst)[0])) / 100
        elif count == 2:
            intergrity_idx = (1 - (sorted(lst)[0] + sorted(lst)[1])/2)/100
        else:
            intergrity_idx = (1 - (sum(lst)/3))/100
        return intergrity_idx
```

- * 데이터 품질지수는 일괄적으로 함수에 묶어서 리스트로 출력되게끔 구현하였다.

  * 전체적으로 일반적인 품질지수를 구하는 공식이나 data.csv의 PK는 TAG_MIN 컬럼 이고 quality.xlsx의 PK는 배정번호 컬럼이므로 함수를 분리하여 별도 계산하게끔 하였다.

- * 일관성 품질지수의 경우 data.csv 데이터셋은 수량에 대한 정보가 없어 1로 처리하였다.

데이터 품질지수 측정

```
#완전성
completeness_idx = completeness_quality_idx(df)
#유일성
if '불량수량' in df.columns:
    uniqueness_idx = uniqueness_quality2(df)
else:
    uniqueness_idx = uniqueness_quality(df)
# 유효성
validity_idx = validity_quality(df)
# 일관성
if '불량수량' in df.columns:
    accuracy_idx = accuracy_quality(df)
else:
    accuracy_idx = 1
#정확성
consistency_idx = consistency_quality(df)
#무결성
intergrity_idx = intergrity_quality(uniqueness_idx, validity_idx,
                                    consistency_idx)

six_data_quality = ['완전성품질지수','유일성품질지수','유효성품질지수',
                    '일관성품질지수','정확성품질지수','무결성품질지수']
data_quality_list = [completeness_idx, uniqueness_idx, validity_idx,
                     consistency_idx, accuracy_idx, intergrity_idx]

for i in range(0,6):
    print(six_data_quality[i] + ' : ' + str(data_quality_list[i]))

return data_quality_list
```

```
data.csv 품질지수
완전성품질지수 : 1.0
유일성품질지수 : 1.0
유효성품질지수 : 1.0
일관성품질지수 : 1.0
정확성품질지수 : 1
무결성품질지수 : 1

quality.xlsx 품질지수
완전성품질지수 : 1.0
유일성품질지수 : 1.0
유효성품질지수 : 1.0
일관성품질지수 : 1.0
정확성품질지수 : 1.0
무결성품질지수 : 1
[1.0, 1.0, 1.0, 1.0, 1.0, 1]
```

- * 함수의 최종부에서 데이터셋의 각 품질지수를 구하며 이를 리스트로 반환해 준다. 품질지수는 위와 같다.

데이터 전처리

```
TAG_MIN                  0
배정번호                     0
건조 1존 OP                 1
건조 2존 OP                 1
건조로 온도 1 Zone          116
건조로 온도 2 Zone          148
세정기                     91
소입1존 OP                4288
소입2존 OP                   0
소입3존 OP                   2
소입4존 OP                   3
소입로 CP 값                 1
소입로 CP 모니터 값          147
소입로 온도 1 Zone          130
소입로 온도 2 Zone          128
소입로 온도 3 Zone          157
소입로 온도 4 Zone          170
솔트 컨베이어 온도 1 Zone     106
솔트 컨베이어 온도 2 Zone     142
솔트조 온도 1 Zone          209
솔트조 온도 2 Zone          203
dtype: int64
```

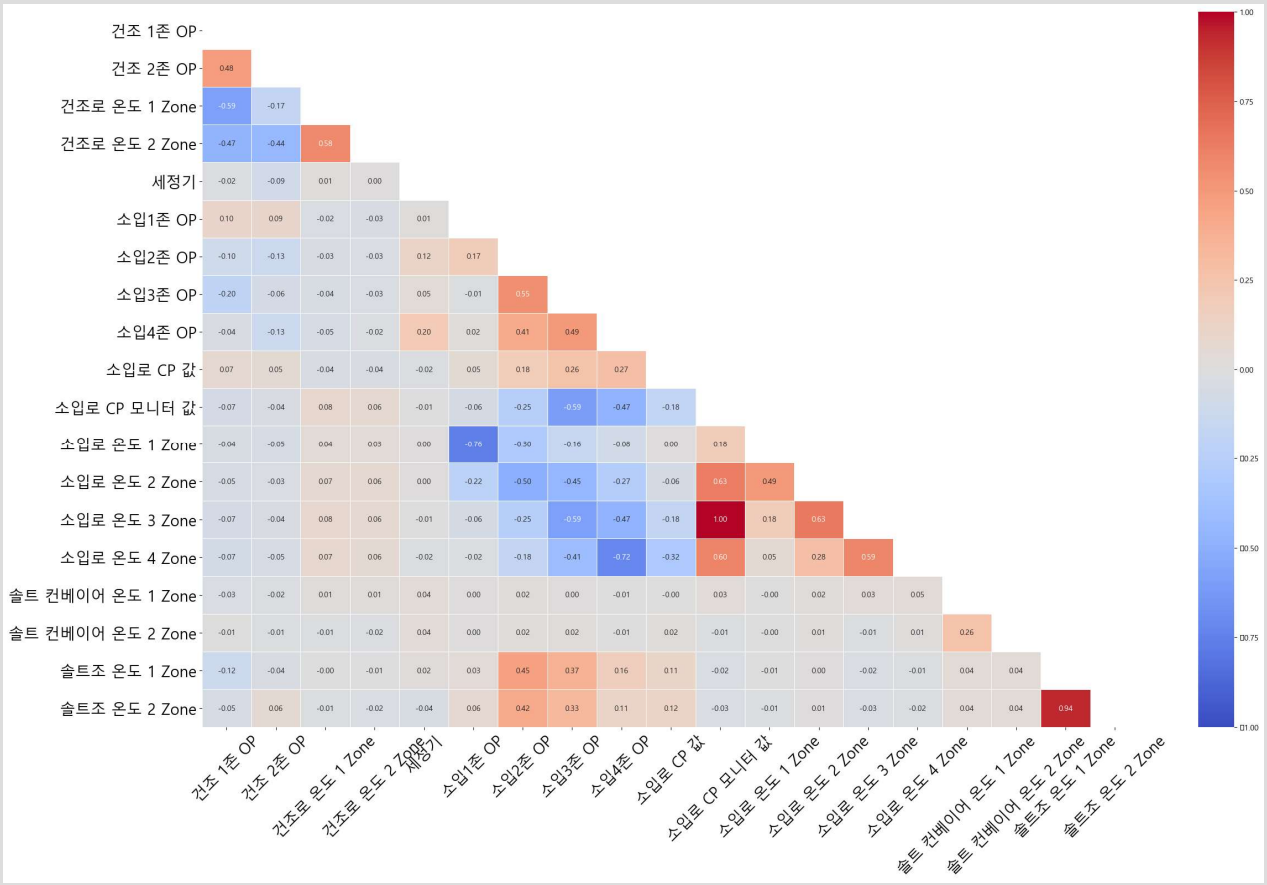
```
dataset.fillna(method='bfill', inplace=True)
dataset.fillna(method='ffill', inplace=True)

dataset['TAG_MIN'] =pd.to_datetime(dataset.
                                   TAG_MIN,
                                   format='%Y-%m-%d %H:%M:%S')
```

- * 우선적으로 data.csv의 nan값을 확인하였으며 fillna 메서드를 통하여 bfill, ffill 순서로 적용 첫번째 row에 있을 nan 값도 채워준다

- * data.csv의 'TAG_MIN' 컬럼의 속성을 pd.to_datetime을 이용하여 datetime[ns] 속성으로 변경해 주는 과정을 거친다.

- * quality.xlsx의 경우 배정번호별 수량결과정도만 기재된 데이터로 nan값이 없고 별도로 처리할 필요성이 없다고 판단. 사용하지 않을 컬럼들만 제거하는 과정을 거쳤다.

데이터 전처리 - 상관관계분석



- * .corr() 메서드와 히트맵을 활용하여 각 컬럼들의 상관관계를 분석해본다. 상관관계가 1인 컬럼[소입로 CP 모니터 값]이 존재하므로 삭제한다.

데이터 전처리

```
aa = list(df_02.배정번호.unique())
bb = aa[:round(len(aa)*.8)]

for i in range(len(df_01)):
    if df_02['배정번호'][i] in bb:
        pass
    else:
        print(i)
        break
```

```
40650
```

```
train = df_02[:40899]
test = df_02[40899:]
```

```
train.drop(columns=['배정번호','TAG_MIN'],
           inplace=True)
test.drop(columns=['배정번호','TAG_MIN'],
          inplace=True)
```

- * 시계열 데이터의 특징상, 특정시점이나 랜덤한 위치에서 데이터셋을 분리하는 것이 아닌 배정번호기준으로 LSTM Auto-Encoder의 train test 데이터셋을 분리하는 것이 필요함
- * 데이터셋의 약 80%위치에 존재하는 컬럼을 찾아 train과 test로 나누었다.
- * 이후 인덱스별로 시간의 흐름을 가지고 있는 데이터셋 이기에 분석에 불필요한 '배정번호', 'TAG_MIN' 컬럼을 삭제해 주는 것으로 전처리를 마무리 한다.

LSTM Auto-Encoder
모델구축 및 학습

LSTM Auto-Encoder 모델 구축

```
def get_model_lstmae():
    lr = 0.0001
    encoder_decoder = Sequential()
    encoder_decoder.add(L.LSTM(x_train.shape[1],
                               activation='relu',
                               input_shape=(x_train.shape[1],x_train.shape[2]),
                               return_sequences=True))
    encoder_decoder.add(L.LSTM(6, activation='relu', return_sequences=True))
    encoder_decoder.add(L.LSTM(1, activation='relu'))
    encoder_decoder.add(L.RepeatVector(n=x_train.shape[1]))
    encoder_decoder.add(L.LSTM(x_train.shape[1], activation='relu', return_sequences=True))
    encoder_decoder.add(L.LSTM(6, activation='relu', return_sequences=True))
    encoder_decoder.add(L.TimeDistributed(L.Dense(units=x_train.shape[2])))

    adam = optimizers.Adam(lr)
    encoder_decoder.compile(loss='mse', optimizer=adam)

    return encoder_decoder
```

- * Encoder – RepeatVector – 디코더 순으로 구축되어 있는것을 확인할 수 있다.
- * Encoder : 입력데이터의 시퀀스의 길이 및 요소를 받아 feature로 변환하는 역할
- * RepeatVector : Encoder에서 출력된 feature를 Decoder에 입력하는 역할
- * Decoder : Encoder로부터 입력받은 feature를 다시 시퀀스 및 각 요소로 복원하여 출력하는 역할

LSTM Auto-Encoder 모델 구축

```
# 각 컬럼마다 lstm을 진행
for i in range(len(train.columns)):
    print(train.columns[i])

    train_ = pd.DataFrame(train[train.columns[i]])
    test_ = pd.DataFrame(test[test.columns[i]])

    scaler = StandardScaler()
    scaler = scaler.fit(train_)

    train_ = scaler.transform(train_)
    test_ = scaler.transform(test_)

    x_train,x_test,y_train,y_test = train_test_split(train_,
                                                     test_,
                                                     TIME_STEPS)

    model = get_model_lstmae()

    history = model.fit(x_train,y_train,
                        epochs =50 , # 시간 관계상 50 epoch로 설정
                        batch_size=128,
                        validation_split=0.1,
                        shuffle = False,
                        callbacks=early_stopping)

    x_train_pred = model.predict(x_train)
```

- * 본 프로젝트에선 반복문을 이용하여 각 컬럼(구간)별 학습을 진행하였으며, 데이터 정규화 및 train_test_split을 각 컬럼별로 진행하여 모델링하였다.
- * 좌측 코드에 따라 각 컬럼별로 LSTM Auto-Encoder를 이용하여 각자 모델링 된다.

LSTM Auto-Encoder 모델 학습

```
x_train_pred = model.predict(x_train)

# MSE
train_mse_loss=np.mean(np.abs
                       (x_train_pred,x_train)
                       ,axis=1)

x_test_pred = model.predict(x_test)

# MAE
test_mae_loss = np.mean(np.abs
                        (x_test_pred - x_test)
                        , axis=1)

# MAE의 95% 지점을 threshold로 지정
THRESHOLD = np.percentile(test_mae_loss, 95)


test_score_df = pd.DataFrame(index=test[TIME_STEPS:].index)
test_score_df[train.columns[i]+'_loss'] = test_mae_loss
test_score_df[train.columns[i]+'_threshold'] = THRESHOLD
test_score_df[train.columns[i]+'_anomaly'] =\
      test_score_df[train.columns[i]+'_loss']\
        > test_score_df[train.columns[i]+'_threshold']

tmp_df = pd.concat([tmp_df,test_score_df],axis=1)
```

- * THRESHOLD는 이상치 판정 구간으로 loss의 약 95% 구간으로 선정하였다.

- * 반복문에 포함된 좌측 코드가 작동되며 각 컬럼별 MAE_loss, THRESHOLD, 그리고 위 두 값의 크기를 비교하여 MAE_loss가 THRESHOLD보다 큰 값이 나온 이상치 여부를 출력한다.

프로그램 예측 평가

모델 학습 예측 결과

```
건조 1존 OP_loss                9239 non-null   float64
건조 1존 OP_threshold           9239 non-null   float64
건조 1존 OP_anomaly             9239 non-null   bool
건조 2존 OP_loss                9239 non-null   float64
건조 2존 OP_threshold           9239 non-null   float64
건조 2존 OP_anomaly             9239 non-null   bool
건조로 온도 1 Zone_loss          9239 non-null   float6
건조로 온도 1 Zone_threshold     9239 non-null   float6
건조로 온도 1 Zone_anomaly       9239 non-null   bool
건조로 온도 2 Zone_loss          9239 non-null   float6
건조로 온도 2 Zone_threshold     9239 non-null   float6
건조로 온도 2 Zone_anomaly       9239 non-null   bool
세정기_loss                     9239 non-null   float64
세정기_threshold                9239 non-null   float64
세정기_anomaly                  9239 non-null   bool
소입1존 OP_loss                 9239 non-null   float64
소입1존 OP_threshold            9239 non-null   float64
소입1존 OP_anomaly              9239 non-null   bool
소입2존 OP_loss                 9239 non-null   float64
소입2존 OP_threshold            9239 non-null   float64
소입2존 OP_anomaly              9239 non-null   bool
소입3존 OP_loss                 9239 non-null   float64
소입3존 OP_threshold            9239 non-null   float64
소입3존 OP_anomaly              9239 non-null   bool
소입4존 OP_loss                 9239 non-null   float64
소입4존 OP_threshold            9239 non-null   float64
소입4존 OP_anomaly              9239 non-null   bool
```

- * 반복문을 활용하여 각 모델 학습이 종료되었을 경우 좌측과 같이 loss값, THRESHOLD 값 및 이상치 여부를 기록한 데이터 프레임이 생성 된다.
- * 이 프로젝트는 온도에 관련한 이상치를 찾는 학습법으로, 각 컬럼들 중 OP 컬럼은 공정의 온도가 하강하였을 시 다시 온도를 올리는 출력의 %를 나타낸 값으로 이후 추론엔 필요없어 제외한다.

모델 학습 예측 결과

```
check_TF_WO_OP['T_sum_3over'] = 0
for j in range(len(check_TF_WO_OP)):
    k = 0
    for i in range(len(check_TF_WO_OP.iloc[j])-2):
        if sum(check_TF_WO_OP.iloc[j][i:i+3]) >= 3:
            k +=1
        else :
            pass
    check_TF_WO_OP['T_sum_3over'][j] = k


check_TF_WO_OP[check_TF_WO_OP['T_sum_3over']>0]

print(len(check_TF_WO_OP[check_TF_WO_OP['T_sum_3over']>1])
      /len(check_TF_WO_OP))
print(len(check_TF_WO_OP[check_TF_WO_OP['T_sum_3over']>2])
      /len(check_TF_WO_OP))
print(len(check_TF_WO_OP[check_TF_WO_OP['T_sum_3over']>3])
      /len(check_TF_WO_OP))
```

```
0.010715445394523217
0.0006494209330014071
0.0
```

```
0.0003357431395151549
```

- * 연속된 3개의 공정에서 이상치인 _anomaly가 확인되었을 경우 1점, 연속된 4개의 공정 또는 3개의 공정 2구간에서 _anomaly가 확인되었을 경우 2점, 연속된 5개의 공정 또는 연속된 4개의 공정 2구간 또는 연속된 3개의 공정 3구간에서 _anomaly가 확인되었을 경우엔 3점을 부여하는 방식으로 점수를 기입

- * 각 로우별 2점 이상인 경우, 3점 이상인 경우, 4점이상인 경우에 대한 불량률을 계산해 보았다

- * 가장 하단부 수치는 전체 데이터셋의 불량률이다.

결 론

- * 이전 페이지에서의 점수체계를 토대로 불량률 계산값을 크기순으로 나열해보면,
- 3점 이상일경우의 불량률 < quality.xlsx 기준 전체 불량률 < 2점 이상일 경우의 불량률 순으로 불량률이 높아지는 것을 확인 할 수 있다.

- * 즉 3점 이상이 되는 경우, 상술한 연속된 5개의 공정 또는 연속된 4개의 공정 2구간 또는 연속된 3개의 공정 3구간에서 이상치가 확인되었을 경우 경고메시지, 알림 등을 통하여 해당 구간의 온도를 이상치 에서 정상구간으로 조절함으로써 불량률을 줄일 수 있을 것으로 기대된다.

- * 위와같은 방식은 공정의 시작 전 미리 예측하여 불량률은 감소시켜 별도의 추가설비 없이도 제조공정의 소모되는 비용은 감소시키고 생산량은 증가시킬 수 있는 장점이 있다.

- * 단순하게 위 공정뿐 아니라, 일정시간단위로 기록된 각 구간별 환경데이터가 있는 제조산업류에도 모두 적용 가능 할 것이라 기대된다.