인공지능_자연어처리 13기

AI 활용한 이상탐지

Meter Anomaly Detection

2020.12

작성자 : 정현일

email : chohi@korea.com

개요

AD 개요

빅데이터 분석 환경에서 수집되는 계기정보의 검침데이터를 기반으로 이상 데이터 식별하기 위한 이상탐지 AI 데이터 모델 개발

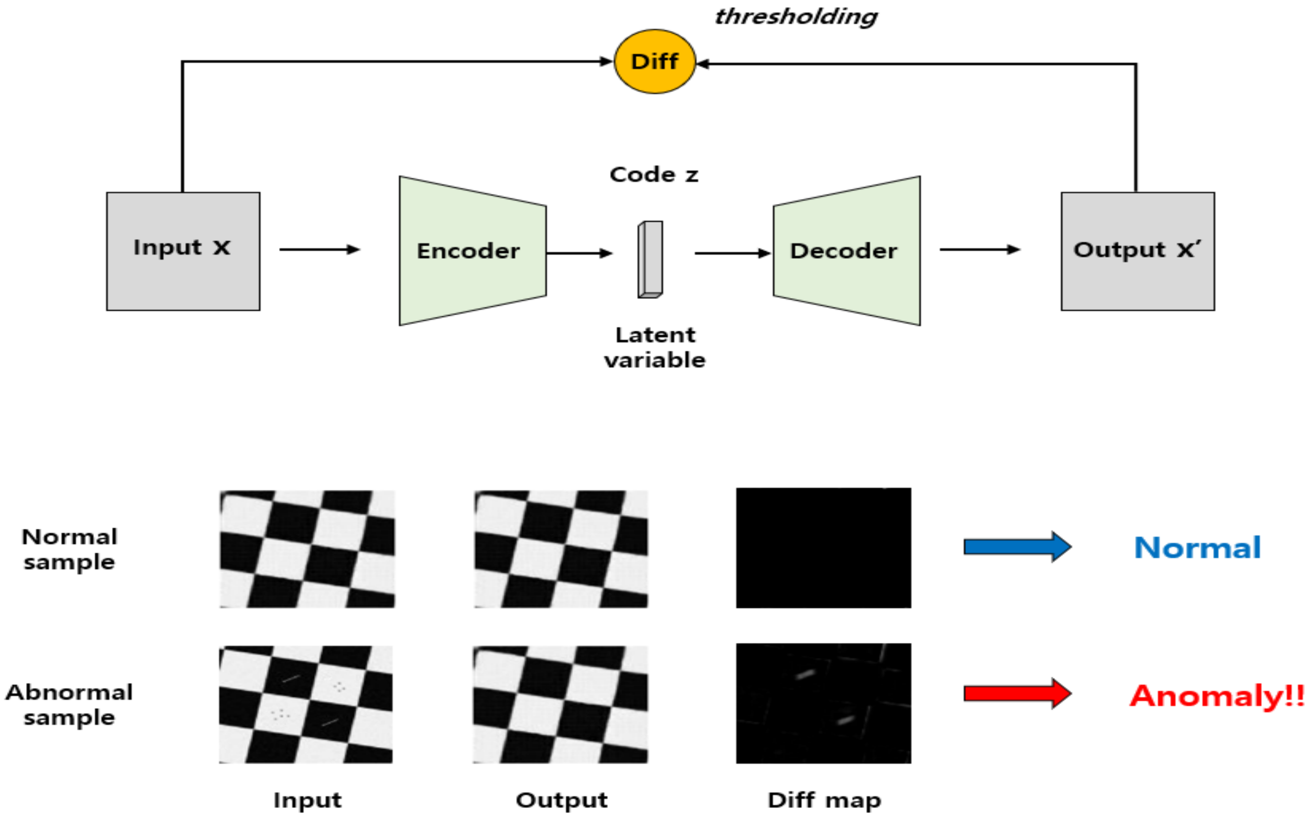


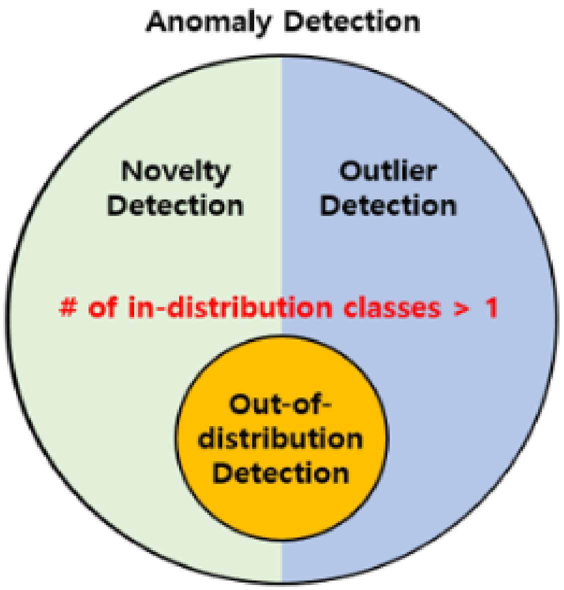
그림 출처: Improving Unsupervised Defect Segmentation by Applying Structural Similarity To Autoencoders, 2019 arXiv

Anomaly Detection 용어 분류(3가지)

1. 학습시 비정상 sample의 사용 여부 및 label 유무에 따른 분류

용어	정상 sample	비정상 sample
Supervised Anomaly Detection	학습에 사용	학습에 사용
Semi-Supervised (One-Class) Anomaly Detection	학습에 사용	학습에 사용 X
Unsupervised Anomaly Detection	모름.(label이 없음) 학습에 사용하는 데이터의 대다수가 정상 sample일 것이라고 가정.	

3. 정상 sample의 class 개수에 따른 분류

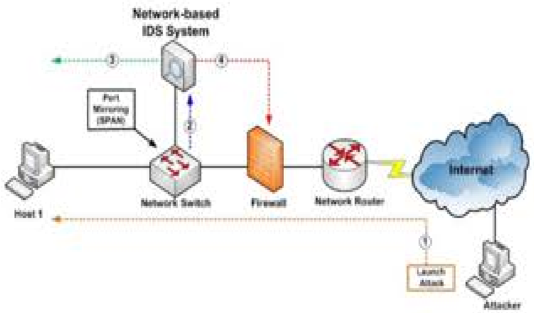


2. 비정상 sample정의에 따른 분류

용어	정상 sample	비정상 sample
Supervised Anomaly Detection	학습에 사용	학습에 사용
Semi-Supervised (One-Class) Anomaly Detection	학습에 사용	학습에 사용 X
Unsupervised Anomaly Detection	모름.(label이 없음) 학습에 사용하는 데이터의 대다수가 정상 sample일 것이라고 가정.	

Anomaly Detection 적용 사례

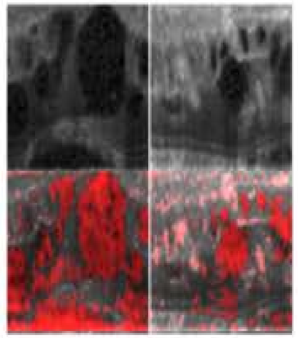
Cyber-intrusion



Fraud

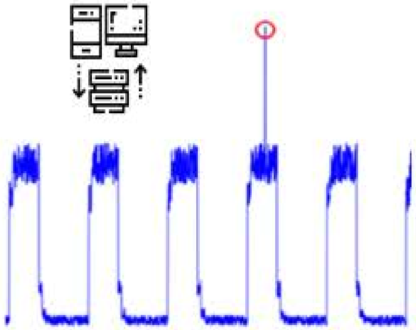
Malware

Medical

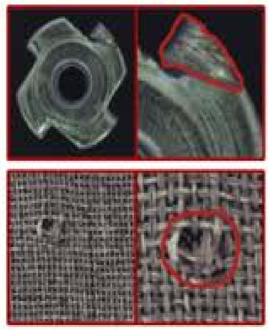


Social Network

Log file



IoT Big-Data



Industrial

Video Surveillance

이상탐지 구성

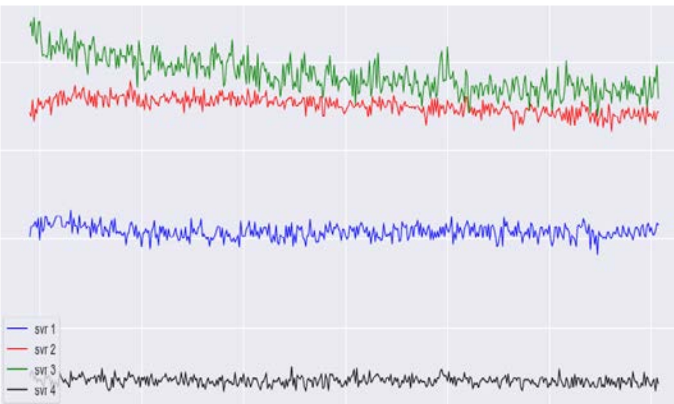
AD 데이터 처리 절차

과거의 정상 데이터 → 데이터 전처리 → AI 이상탐지 모델 (Test Set)

실시간 변동 데이터 → 데이터 전처리 → AI 이상탐지 모델 (Test Set) → 이상탐지 데이터

AD 목표

관찰 불가능할 만큼 많은 변수

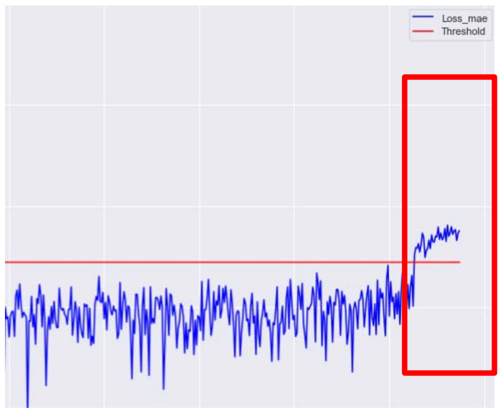


이상탐지

(Anomaly Detection)

AI 모델

탐지된 이상징후 측정값들



데이터 처리

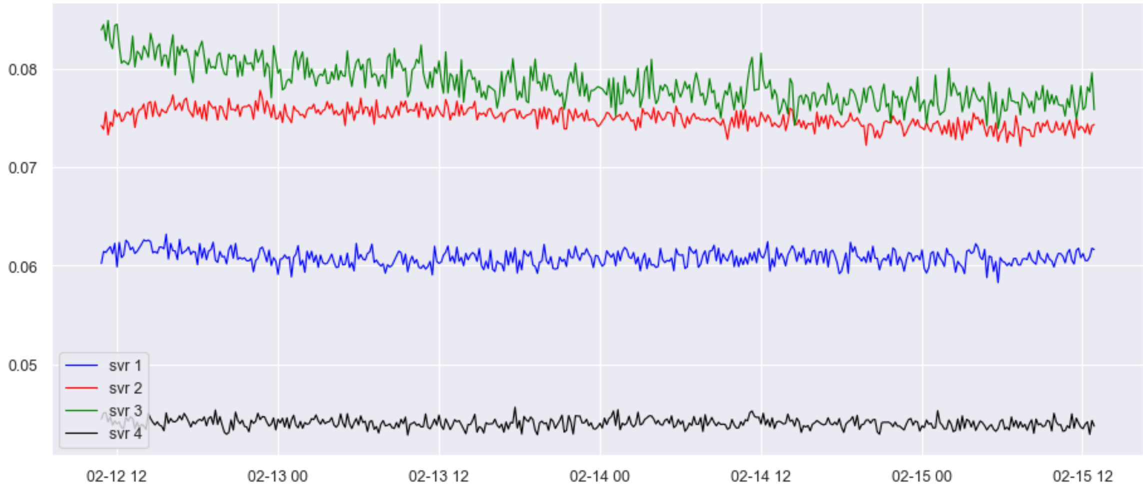
계기의 검침데이터 사용

4개의 계기에서 10분마다 수집되는 검침값 중 약 94시간 데이터를 사용함

```
# 검침 계기의 시계열 데이터
merged_data.index = pd.to_datetime(merged_data.index, format='%Y.%m.%d.%H.%M.%S')
merged_data = merged_data.sort_index()
merged_data.to_csv('Averaged_svr_Dataset.csv')
print("Dataset shape:", merged_data.shape)
merged_data.head()
```

```
Dataset shape: (565, 4)
```

	svr 1	svr 2	svr 3	svr 4
2004-02-12 10:52:39	0.060236	0.074227	0.083926	0.044443
2004-02-12 11:02:39	0.061455	0.073844	0.084457	0.045081
2004-02-12 11:12:39	0.061361	0.075609	0.082837	0.045118
2004-02-12 11:22:39	0.061665	0.073279	0.084879	0.044172
2004-02-12 11:32:39	0.061944	0.074593	0.082626	0.044659



<Training Data>

학습데이터, 검증 데이터 (90:10)비율로 학습

```
# define the autoencoder network model
def autoencoder_model(X):
    inputs = Input(shape=(X.shape[1], X.shape[2]))
    L1 = LSTM(16, activation='relu', return_sequences=True,
              kernel_regularizer=regularizers.l2(0.00))(inputs)
    L2 = LSTM(4, activation='relu', return_sequences=False)(L1)
    L3 = RepeatVector(X.shape[1])(L2)
    L4 = LSTM(4, activation='relu', return_sequences=True)(L3)
    L5 = LSTM(16, activation='relu', return_sequences=True)(L4)
    output = TimeDistributed(Dense(X.shape[2]))(L5)
    model = Model(inputs=inputs, outputs=output)
    return model
```

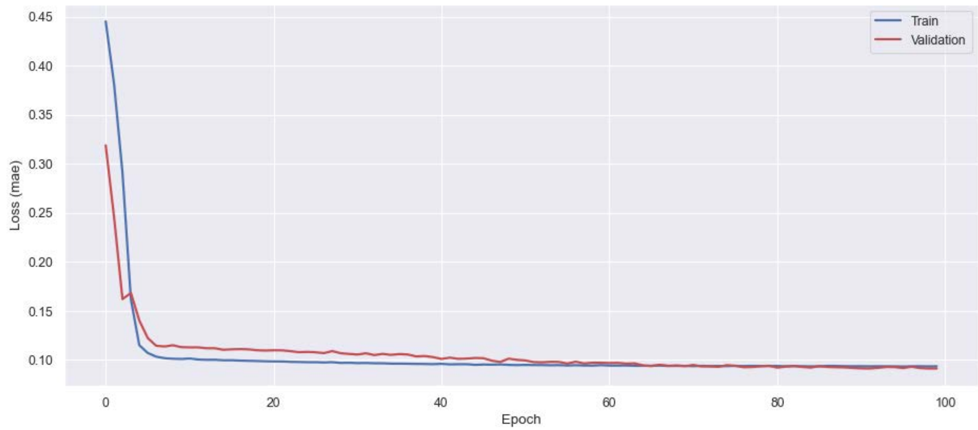
```
# create the autoencoder model
model = autoencoder_model(X_train)
model.compile(optimizer='adam', loss='mae')
model.summary()
```

```
Model: "functional_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_4 (InputLayer)         [(None, 1, 4)]            0
_________________________________________________________________
lstm_12 (LSTM)               (None, 1, 16)             1344
_________________________________________________________________
lstm_13 (LSTM)               (None, 4)                 336
_________________________________________________________________
repeat_vector_3 (RepeatVecto (None, 1, 4)              0
_________________________________________________________________
lstm_14 (LSTM)               (None, 1, 4)              144
_________________________________________________________________
lstm_15 (LSTM)               (None, 1, 16)             1344
_________________________________________________________________
time_distributed_3 (TimeDist (None, 1, 4)              68
=================================================================
Total params: 3,236
```

Fit the Model

epochs = 100, batch_size = 10,

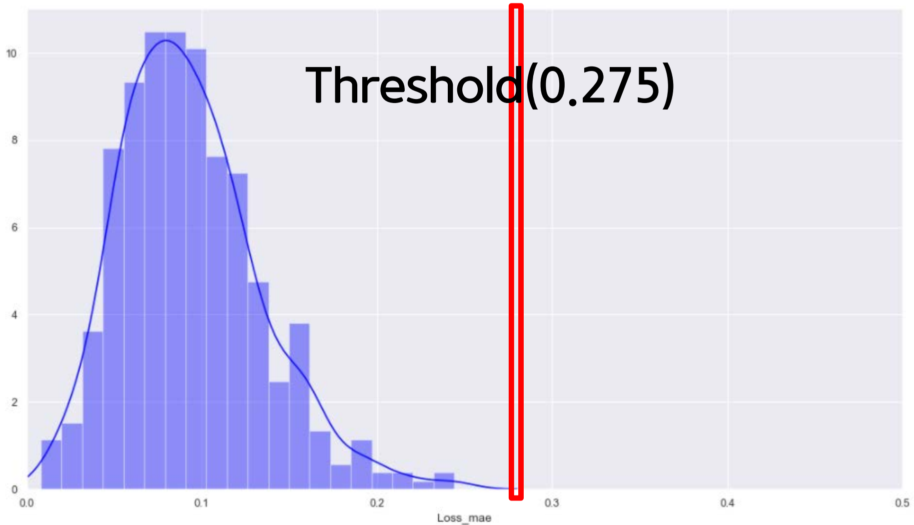
Epoch 1/100 : 1s 20ms/step - loss: 0.4116 - val_loss: 0.2901
Epoch 10/100 : 1s 11ms/step - loss: 0.0962 - val_loss: 0.1327
Epoch 20/100 : 1s 11ms/step - loss: 0.0936 - val_loss: 0.1229
Epoch 30/100 : 1s 11ms/step - loss: 0.0896 - val_loss: 0.1076
Epoch 40/100 : 1s 11ms/step - loss: 0.0891 - val_loss: 0.1041
Epoch 50/100 : 1s 11ms/step - loss: 0.0885 - val_loss: 0.1007
Epoch 60/100 : 1s 11ms/step - loss: 0.0878 - val_loss: 0.0993
Epoch 70/100 : 1s 12ms/step - loss: 0.0860 - val_loss: 0.0974
Epoch 80/100 : 1s 12ms/step - loss: 0.0696 - val_loss: 0.0761
Epoch 90/100 : 1s 11ms/step - loss: 0.0676 - val_loss: 0.0740
Epoch 100/100 : 1s 11ms/step - loss: 0.0656 - val_loss: 0.0740



손실함수 분포

```
X_pred = model.predict(X_train)
X_pred = X_pred.reshape(X_pred.shape[0], X_pred.shape[2])
X_pred = pd.DataFrame(X_pred, columns=train.columns)
X_pred.index = train.index

scored = pd.DataFrame(index=train.index)
Xtrain = X_train.reshape(X_train.shape[0], X_train.shape[2])
scored['Loss_mae'] = np.mean(np.abs(X_pred-Xtrain), axis = 1)
plt.figure(figsize=(16,9), dpi=80)
plt.title('Loss Distribution', fontsize=16)
sns.distplot(scored['Loss_mae'], bins = 20, kde= True, color =
'blue');
plt.xlim([0.0,.5])
```



손실계산(test Set)

```
X_pred = model.predict(X_test)
X_pred = X_pred.reshape(X_pred.shape[0], X_pred.shape[2])
X_pred = pd.DataFrame(X_pred, columns=test.columns)
X_pred.index = test.index

scored = pd.DataFrame(index=test.index)
Xtest = X_test.reshape(X_test.shape[0], X_test.shape[2])
scored['Loss_mae'] = np.mean(np.abs(X_pred-Xtest), axis = 1)
scored['Threshold'] = 0.275
scored['Anomaly'] = scored['Loss_mae'] > scored['Threshold']
scored.head()
```

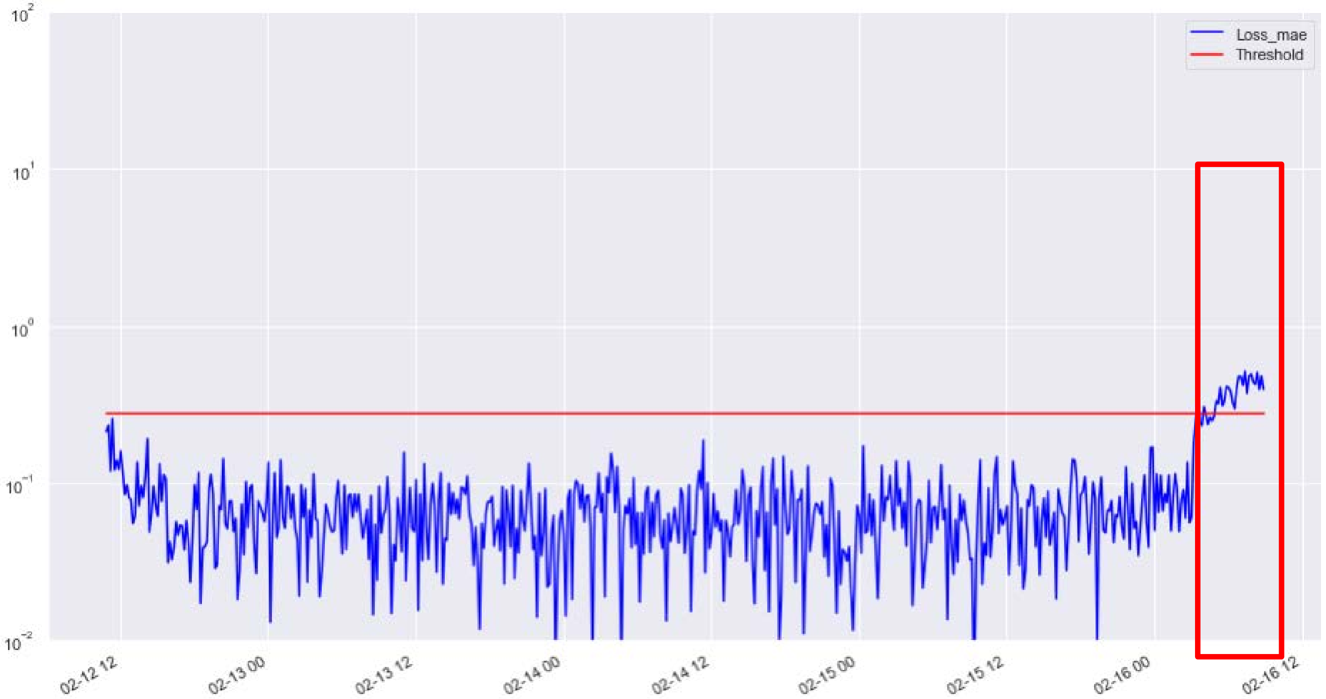
	Loss_mae	Threshold	Anomaly
2004-02-15 23:52:39	0.168483	0.275	False
2004-02-16 00:02:39	0.049893	0.275	False
2004-02-16 00:12:39	0.114139	0.275	False
2004-02-16 00:22:39	0.065233	0.275	False
2004-02-16 00:32:39	0.111165	0.275	False

이상탐지 결과

계기 검침데이터 이상 탐지 결과

```
X_pred_train = model.predict(X_train)
X_pred_train = X_pred_train.reshape(X_pred_train.shape[0], X_pred_train.shape[2])
X_pred_train = pd.DataFrame(X_pred_train, columns=train.columns)
X_pred_train.index = train.index

scored_train = pd.DataFrame(index=train.index)
scored_train['Loss_mae'] = np.mean(np.abs(X_pred_train-Xtrain), axis = 1)
scored_train['Threshold'] = 0.275
scored_train['Anomaly'] = scored_train['Loss_mae'] > scored_train['Threshold']
scored = pd.concat([scored_train, scored])
```



참고 문서 및 산출물

참고 문서

1. Anomaly Detection of Time Series
2. LSTM-Based System-Call Language Modeling and Robust Ensemble Method for Designing Host-Based Intrusion Detection Systems
3. Time-Series Anomaly Detection Service at Microsoft

참고 URL

https://github.com/hoya012/awesome-anomaly-detection
https://blog.naver.com/phj8498/222120169674

소스 코드

https://github.com/chohi22/Anomaly_Detection