Deep Learning for Anomaly Detection

(from Supervised to Unsupervised approach)

2020. 3. 20.

Data Scientist

Kwang Myung Yu

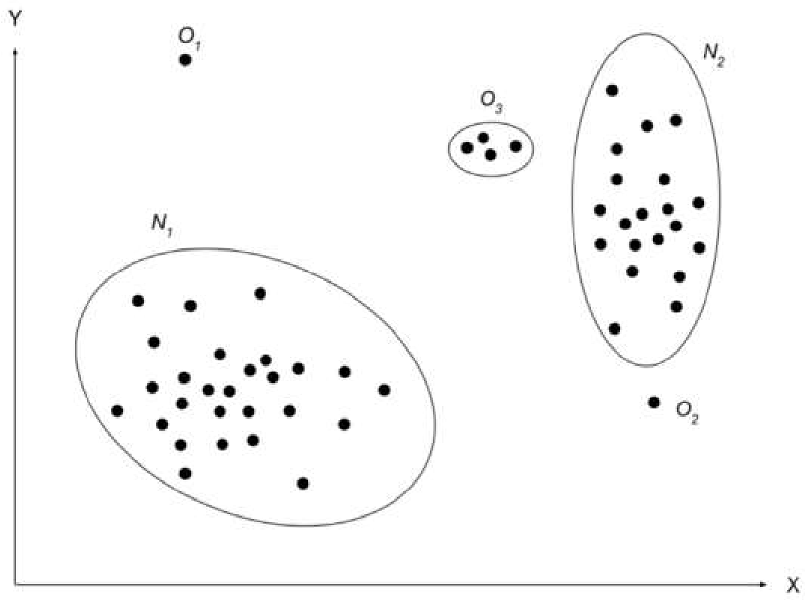
Table of Contents

1. Backgrounds

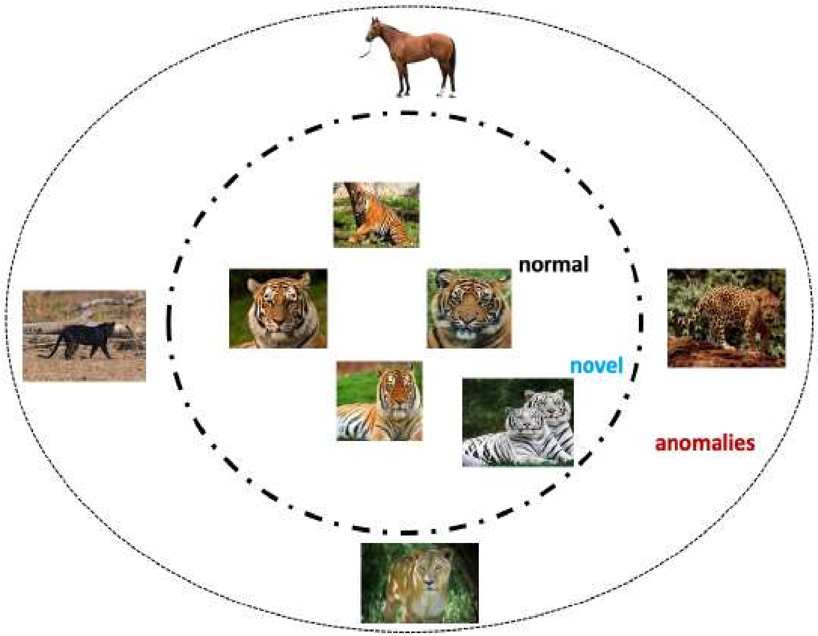
1.1 정 의

✓ Anomaly Detection(이상 탐지)

- 정상(Normal) 샘플과 비정상, 이상치, 특이치(Abnormal) 샘플을 구별해 내는 문제
- 유사 용어 : Novelty Detection, Outlier Detection ** Novelty : new, unseen*



〈2D dataset case〉



〈Image dataset case〉

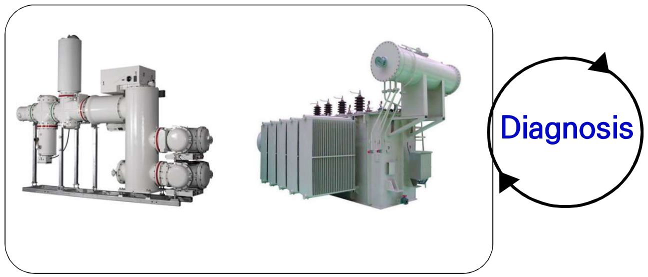
〈출처 : Deep learning for anomaly detection [13]〉

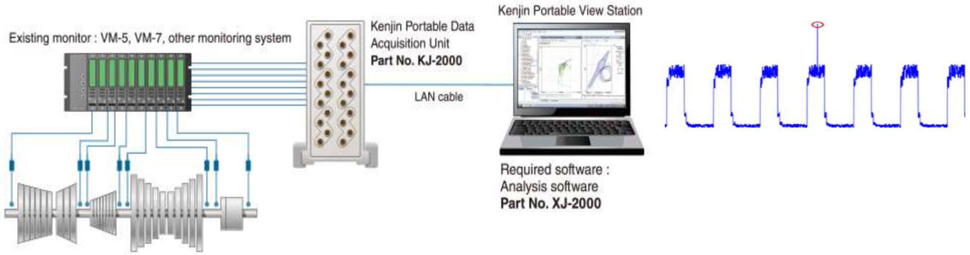
1. Backgrounds

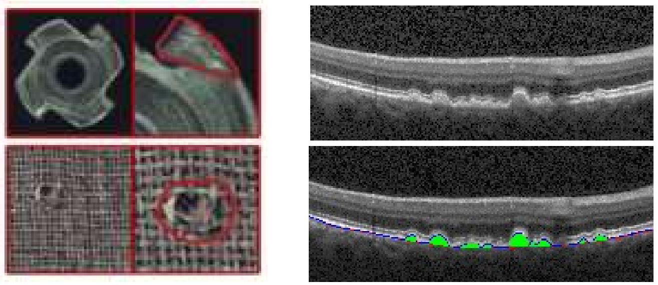
1.2 활용 분야

✓ Anomaly Detection 활용 분야

- Fault(Damage) Detection and Diagnostics(FDD), Fraud detection : 정형 데이터
- Sensor networks, IoT data anomaly detection : 정형(주로 시계열) 데이터
- Video surveillance, Medical image, data-log anomaly detection : 비정형 데이터





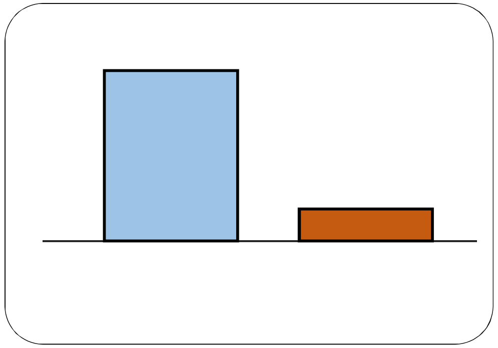


1. Backgrounds

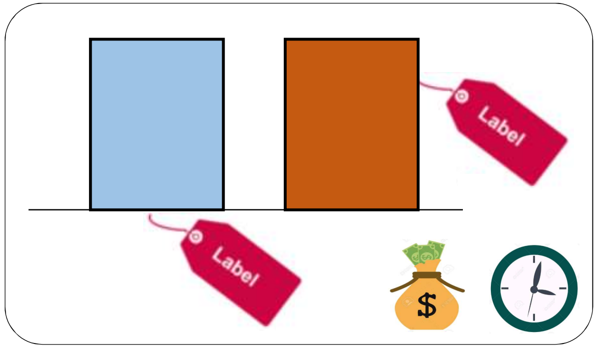
1.3 주요 이슈

✓ 데이터 특징

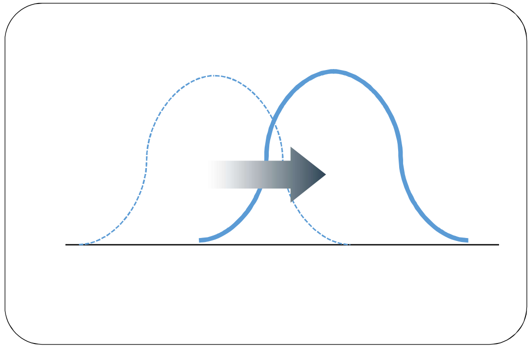
- Data Imbalance : 정상 샘플보다 비정상 샘플의 비중이 상대적으로 적다.
- Labeling : 비정상 데이터를 수집하고, 데이터를 레이블링(Labeling)하는데 많은 비용과 시간이 소요된다.
- Availability of Labels : 운영 환경의 변화, 설비 노후 등에 의한 데이터 분포, 이상(Abnormality) 기준이 달라진다.



- Data Augmentation
- Resampling



- Semi-Supervised learning
- Unsupervised learning



- Unsupervised learning

→ 데이터 및 타겟 시스템의 특징에 따라 다른 접근법이 적용되어야 한다.

1.4 연구 동향

✓ 딥러닝 기반 이상탐지 연구동향

- Supervised → Unsupervised
- Statistical ML(OCSVM, LOF, Isolation Forest, ~'05) → Autoencoders(~ '15) → GAN (최근)
- 접근방법 비교

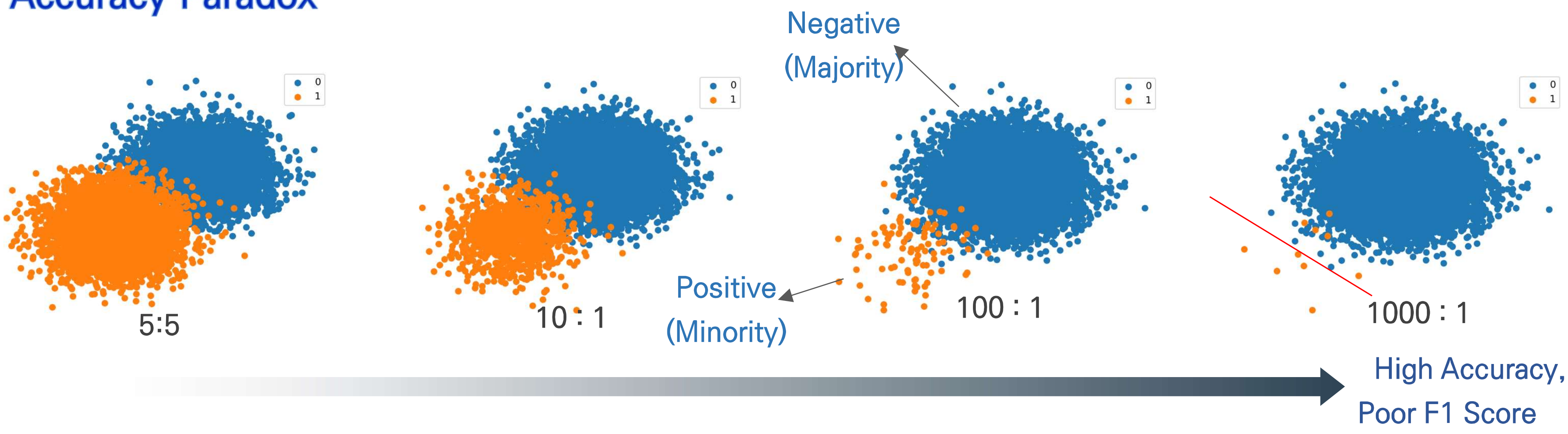
Supervised	Semi-supervised	Hybrid	Unsupervised
· Resampling – Over/ Under-sampling · Data Augmentation – GAN	· One-Class SVM · SVDD	· NN + SVM (Deep SVDD) · Autoencoder + Classifier (One-Class NN))	· Statistical ML – Isolation Forest, LOF · Autoencoders · GAN – anoGAN, GANomaly

高성능, 전처리 의존성 上 → 低성능, 전처리 의존성 下

1. Backgrounds

1.5 평가 방법 – 1

✓ Accuracy Paradox



✓ Confusion matrix

	Predicted 0	Predicted 1
Actual 0	True Negative (TN)	False Positive (FP)
Actual 1	False Negative (FN)	True Positive (TP)

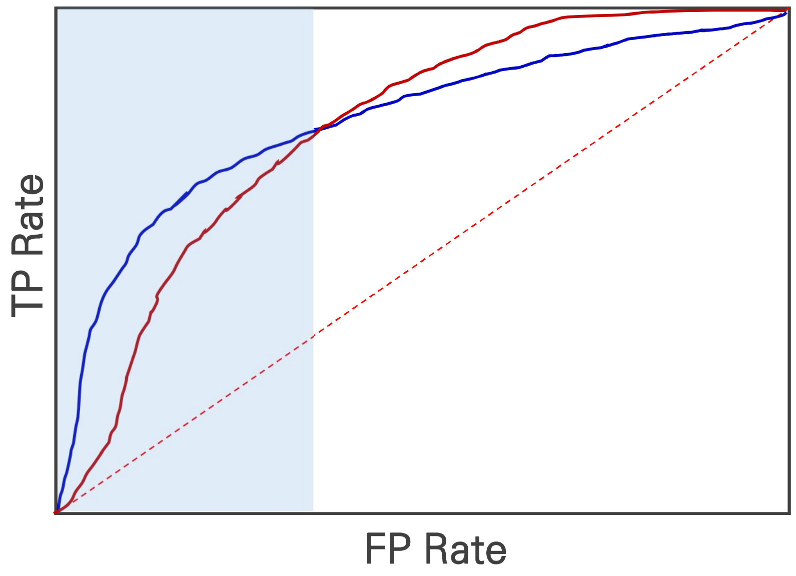
- Accuracy = (TN + TP) /(TN + TP + FN + FP)
- Precision = TP / (TP+ FP) ： FP 최소화에 관심
- Recall = TP / (TP+ FN) : FN 최소화에 관심
- F1 = (2*Precision*Recall)/(Precision + Recall)

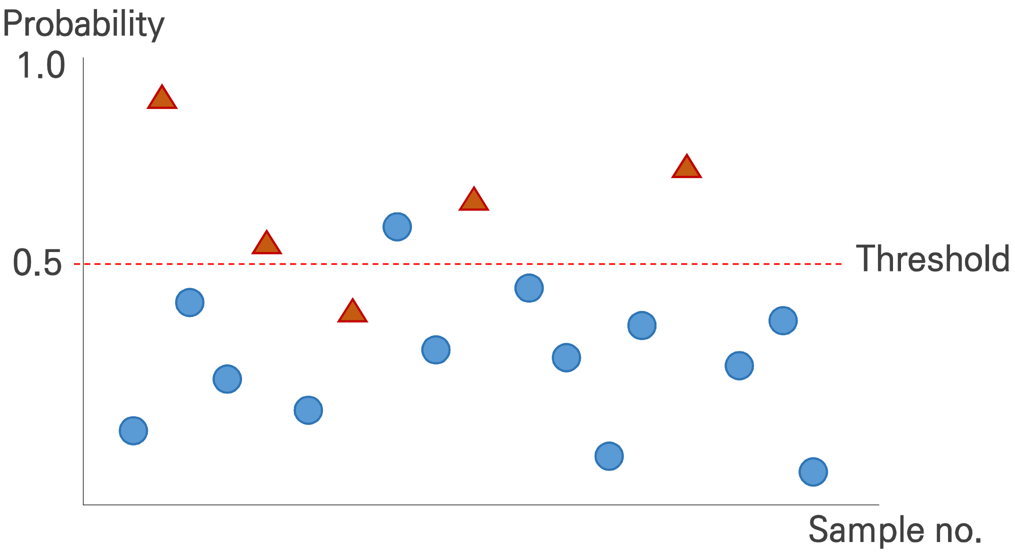
  ☞ Precision과 Recall의 Harmonic mean

1. Backgrounds

1.5 평가 방법 – 2

✓ **ROC(Receiver Operating Characteristic) :** False Positive Rate (x) vs. True Positive Rate (y)





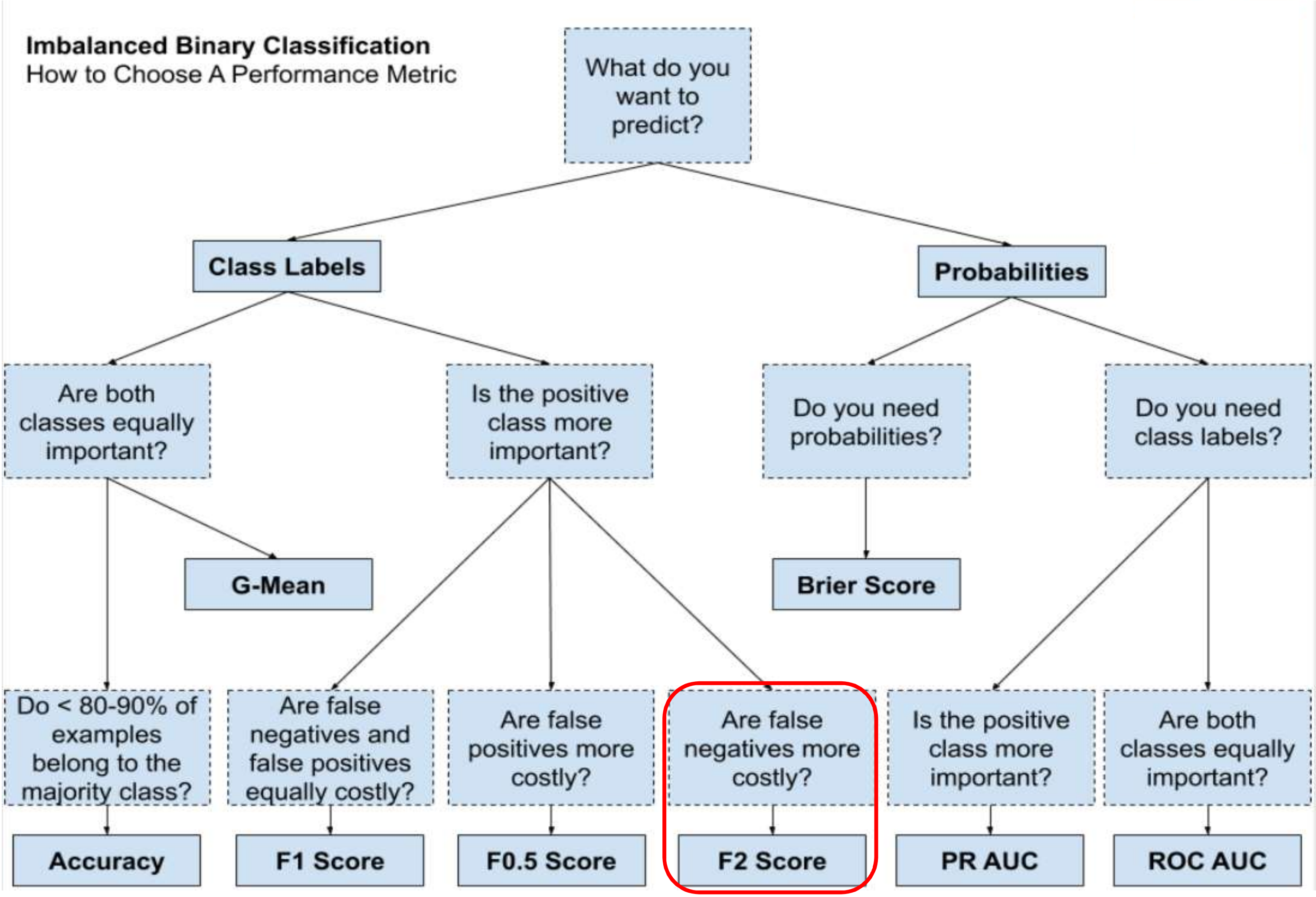
- False Positive Rate = $\dfrac{FP}{FP + TN}$

- True Positive Rate = Sensitivity = Recall = $\dfrac{TP}{TP + FN}$

1.6 참 고

✓ 성능지표 선정방법

$$F_\beta = (1+\beta^2)\frac{precision \cdot recall}{(\beta^2 \cdot precision) + recall}$$



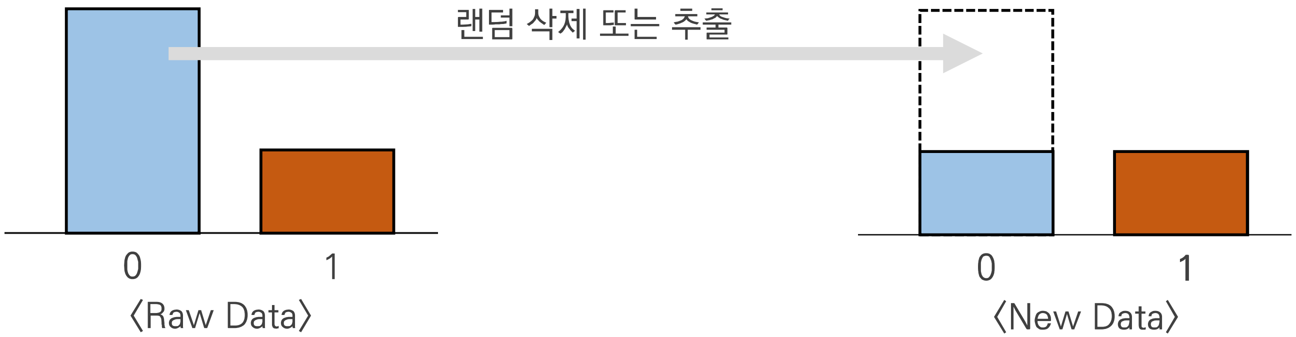
〈출처 : https://machinelearningmastery.com/tour-of-evaluation-metrics-for-imbalanced-classification/〉

2. Supervised learning

2.1 Data resampling

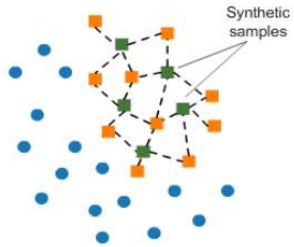
✓ Undersampling

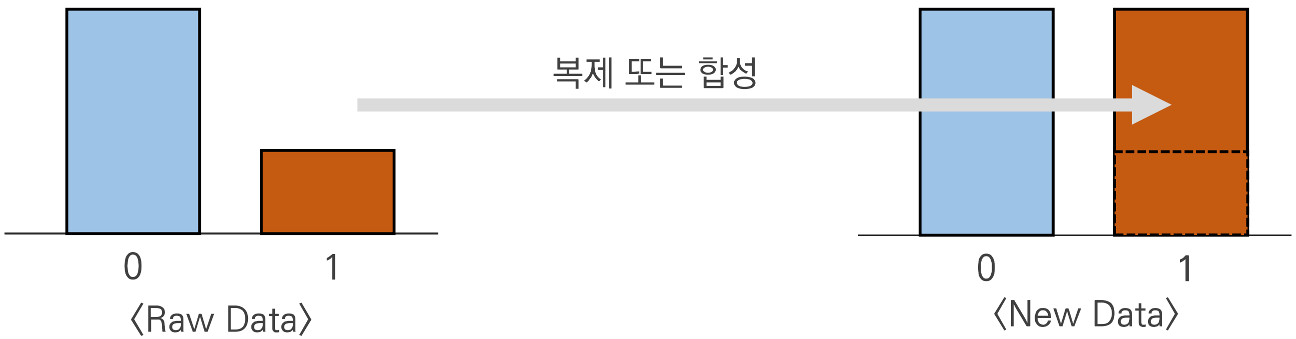
- Majority Class 데이터 일부를 삭제하거나, 랜덤 샘플링하여 imbalance 개선
  - 데이터 셋의 샘플(observation) 수가 많을 때 효과적임/ 중요정보 손실 문제가 발생



✓ Oversampling

- Minority Class를 복제(Duplicate)하거나, 합성하여 imbalance 개선
  - 정보 손실 문제 해결 가능/ Overfitting 발생 가능성
  - 예) SMOTE(Synthetic Minority Over-sampling Technique), Random OverSampler



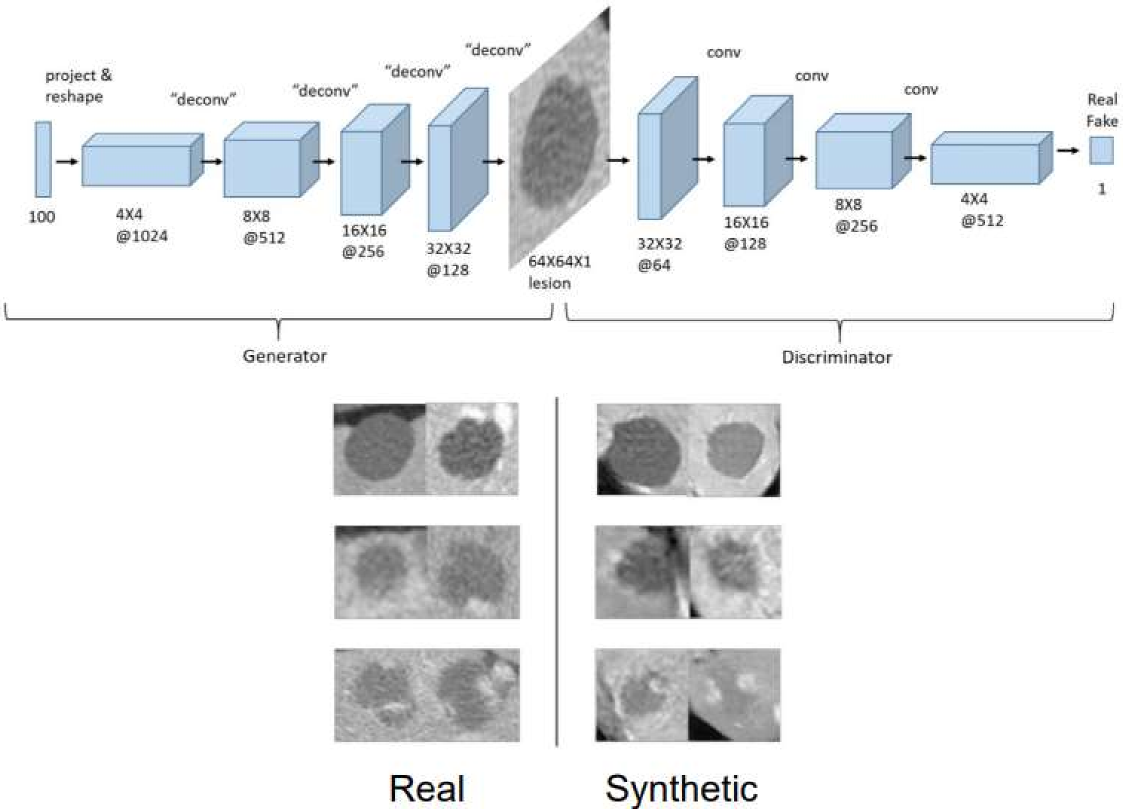


2. Supervised learning

2.1 Data resampling

✓ Data Augmentation

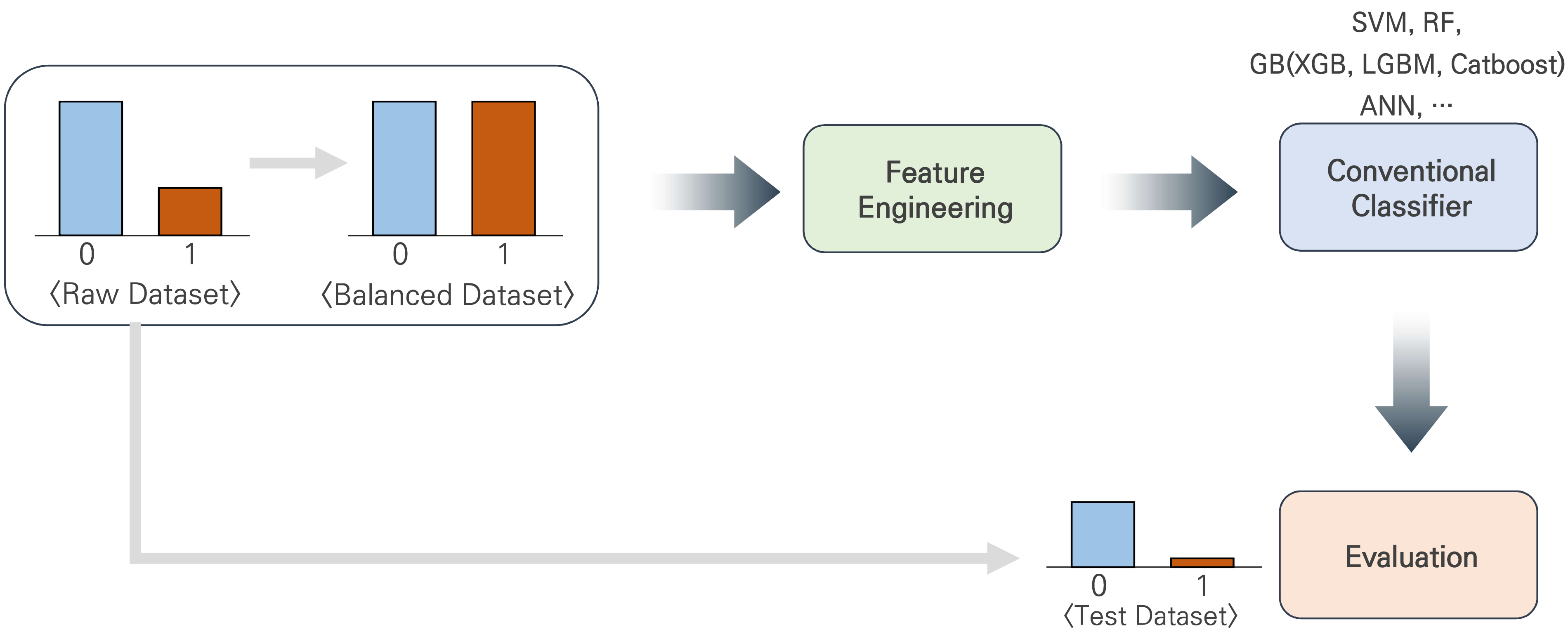
- 생성모델(Generative model), GAN 등을 활용한 데이터 복제
  – 이미지 분야에 연구 진행(예: 의료) / Tabular 데이터에 효과가 적음



〈출처 : Synthetic Data Augmentation using GAN for Improved Liver Lesion Classification, Maayan Frid-Ada et. al, ISBI 2018〉

2.2 지도학습 기반 이상탐지

✓ **분류 문제로 전환**

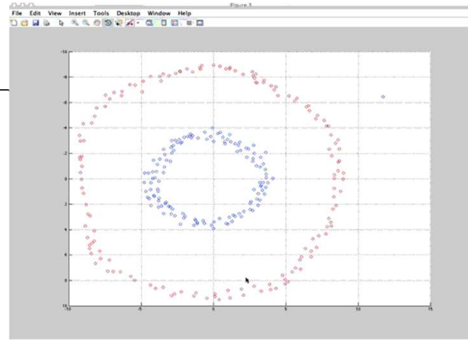


2.3 장단점

- Pros
  - (이상탐지) 성능이 우수하다.
  - 기존 머신러닝, 딥러닝 모델을 그대로 사용할 수 있다.

- Cons
  - 비정상 샘플이 충분히 확보된 경우에만 사용 가능하다.
    (대부분의 경우 비정상 샘플을 취득하거나 레이블링을 하는데 시간과 비용이 많이 소요된다.)

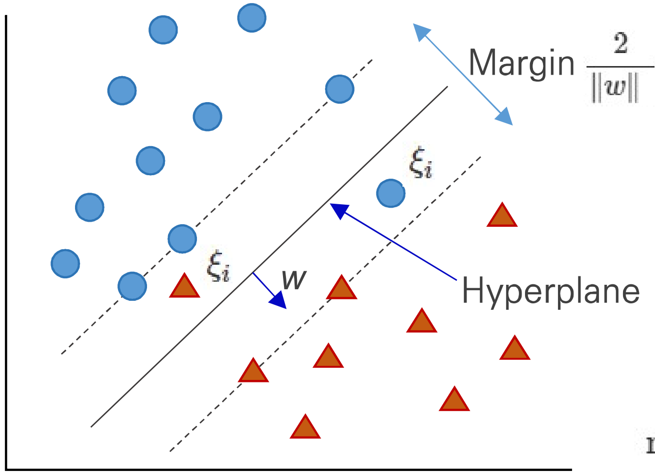
3. Semi-supervised learning



3.1 SVM based approach

✓ One-Class Support Vector Machine(Schölkopf et al., 2000)

〈SVM〉



$$max \frac{2}{\|w\|_2}$$

$$\min \frac{1}{2}\|w\|_2^2$$

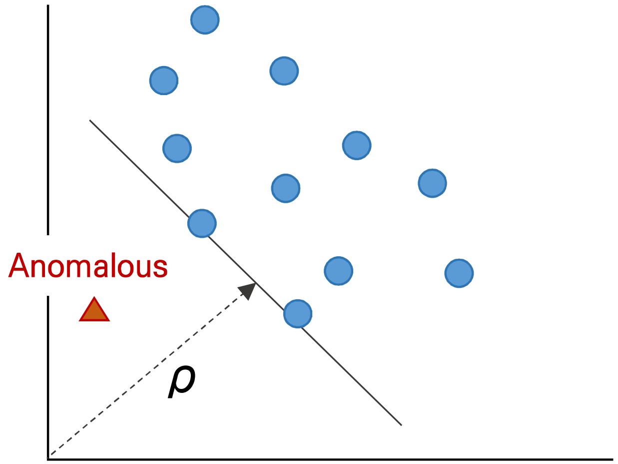
$$\min_{w,\,b,\,\xi_i} \frac{\|w\|^2}{2} + C\sum_{i=1}^{n}\xi_i$$

subject to:

$$y_i(w^T\phi(x_i)+b) \geq 1-\xi_i \quad \text{for all } i=1,\ldots,n$$
$$\xi_i \geq 0 \quad \text{for all } i=1,\ldots,n$$

- Decision boundary(Hyperplane)의 마진이 최대가 되도록 학습

〈One-Class SVM〉



$$\min_{w,\,\xi_i,\,\rho} \frac{1}{2}\|w\|^2 + \frac{1}{\nu n}\sum_{i=1}^{n}\xi_i - \rho$$

subject to:

$$(w\cdot\phi(x_i)) \geq \rho-\xi_i \quad \text{for all } i=1,\ldots,n$$
$$\xi_i \geq 0 \quad \text{for all } i=1,\ldots,n$$

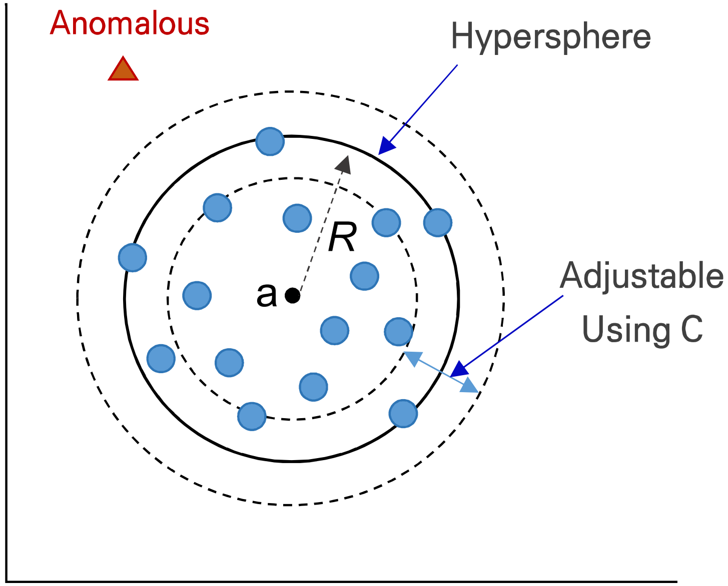
- 원점에서 Decision boundary의 거리(ρ)가 최대가 되도록 학습
- 정상(Normal) 데이터만으로 학습
- 예측 값 =1 일 때 정상, 예측 값이 −1일 때 이상

3. Semi-supervised learning

3.1 SVM based approach

✓ SVDD(Support Vector Data Description, Tax and Duin, 2004)

〈SVDD〉



$$\min_{R,\,\mathbf{a}} R^2 + C\sum_{i=1}^{n}\xi_i$$

subject to:

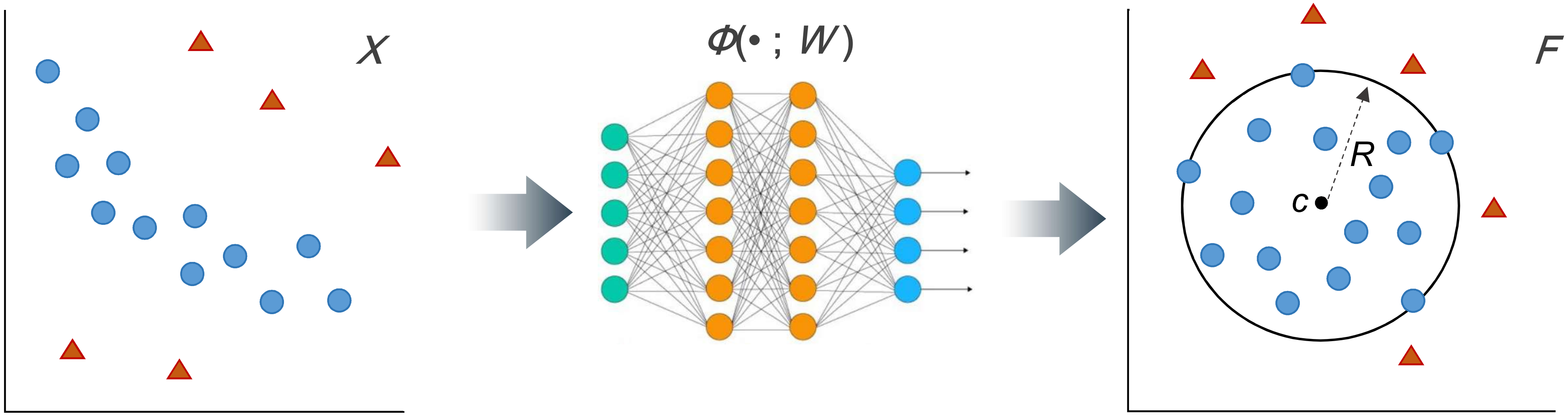
$$\|x_i - \mathbf{a}\|^2 \le R^2 + \xi_i \qquad \text{for all } i = 1,\ldots,n$$

$$\xi_i \ge 0 \qquad \text{for all } i = 1,\ldots,n$$

- Decision boundary의 반경(R)이 최소화 되도록 학습
- 정상(Normal) 데이터만으로 학습
- a에서 측정 값의 거리가 〈 R 일 때 정상, 〉 R 일 때 이상

3.2 Neural Net + SVM

✓ Deep SVDD(Ruff et al., 2018)



- 신경망을 커널(Kernel) 함수로 사용
- Decision boundary의 반경(R)이 최소화 되도록 학습
- 정상(Normal) 데이터만으로 학습
- c 에서 측정 값의 거리가 〈 R 일 때 정상, 〉 R 일 때 이상

$$\min_{R,\mathcal{W}} \quad R^2 + \frac{1}{\nu n}\sum_{i=1}^{n}\max\{0, \|\phi(\boldsymbol{x}_i;\mathcal{W}) - \mathbf{c}\|^2 - R^2\} + \frac{\lambda}{2}\sum_{\ell=1}^{L}\|\boldsymbol{W}^\ell\|_F^2.$$

Anomaly Score

$$s(\boldsymbol{x}) = \|\phi(\boldsymbol{x};\mathcal{W}^*) - \mathbf{c}\|^2$$

3. Semi-supervised learning : Hybrid approach

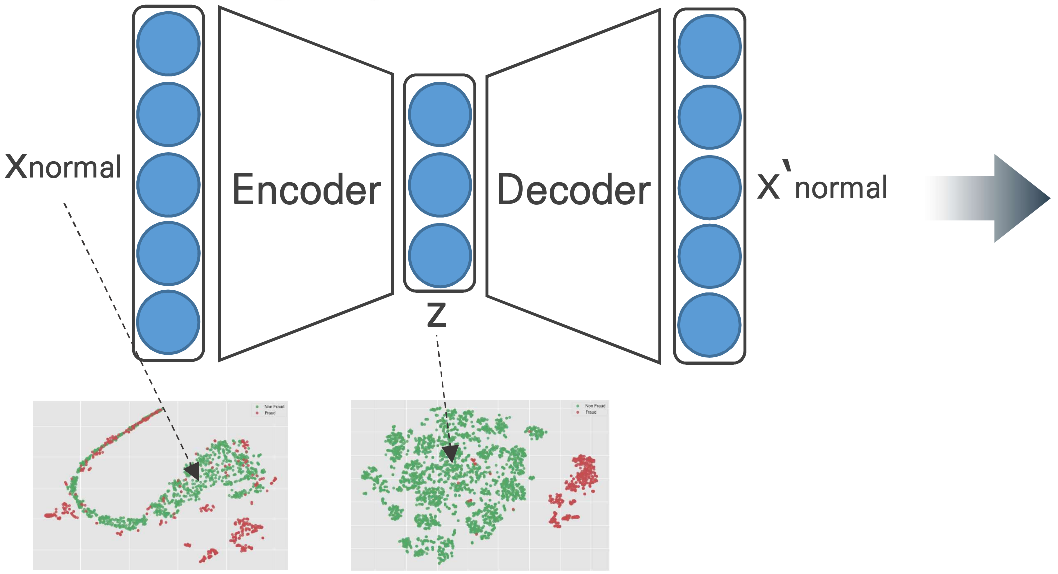
3.3 Autoencoder + Classifier

✓ One-Class Neural Networks(Chalapathy et al., 2019)

- Step 1 : 정상 데이터로 Autoencoder를 학습
- Step 2 : 전체(정상+비정상) 데이터로 분류기를 학습(Encoder의 가중치는 고정)

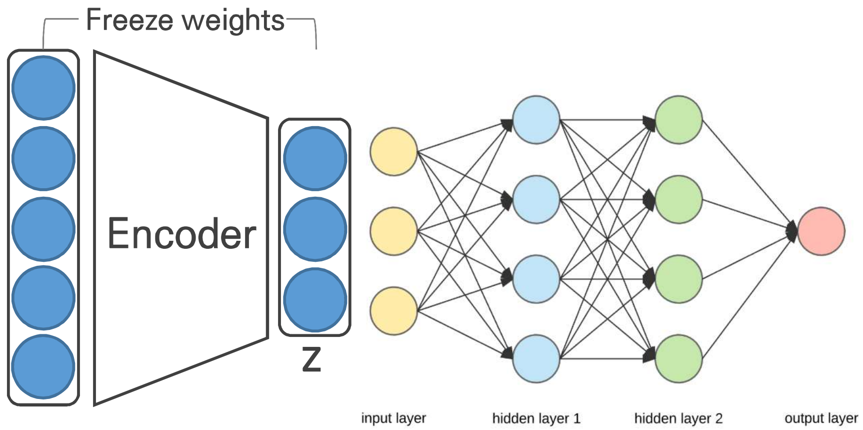
〈Step 1: Representation learning〉

$$L(x, x') = ||\, x - x' ||^2$$



〈t-SNE chart〉

〈Step 2: One-Class NN〉



3.4 장단점

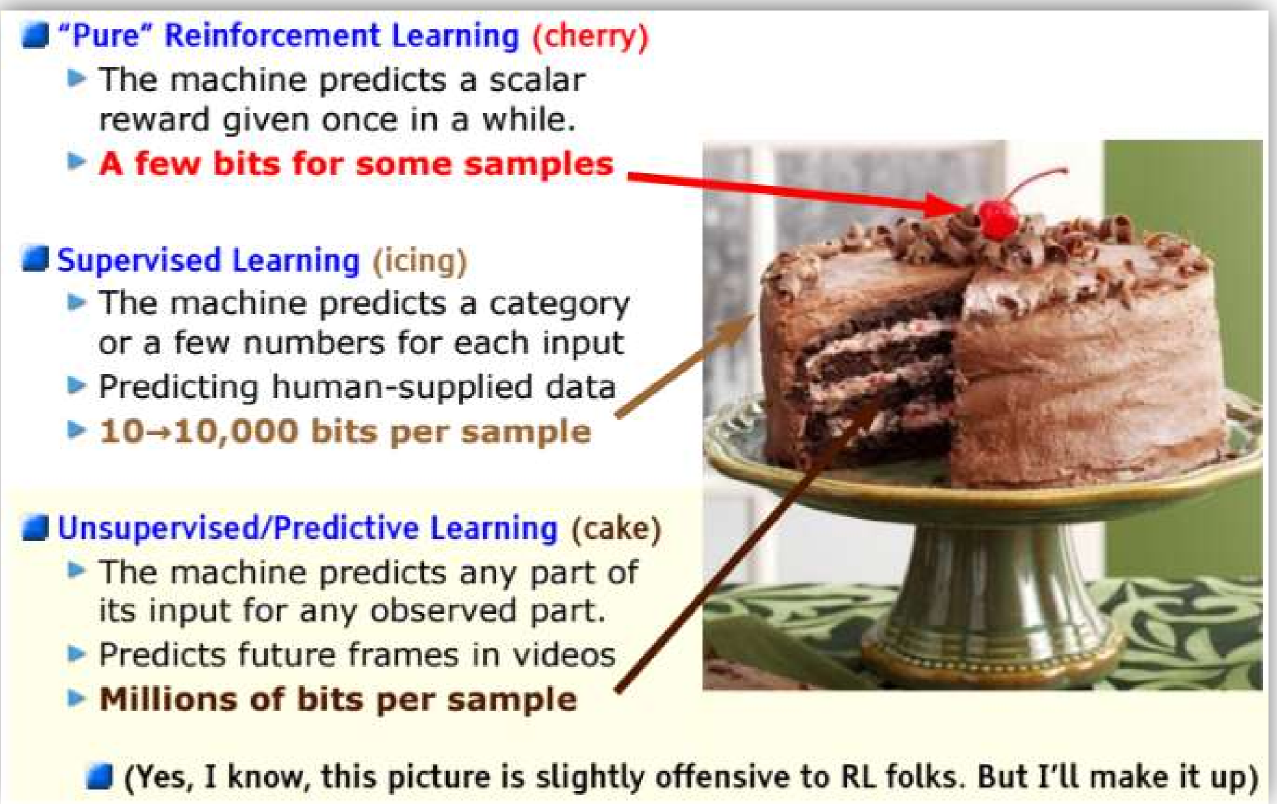
- Pros
  - 정상 데이터 만으로 모델 학습이 가능하다.
  - 비정상 데이터가 부족해도 모델 구현이 가능하다.

- Cons
  - 지도학습 방법과 비교했을 때 검출 정확도가 낮다.

4. Unsupervised learning

4.0 Why unsupervised?

✓ Yann LeCun's cake analogy (at NIPS 2016)

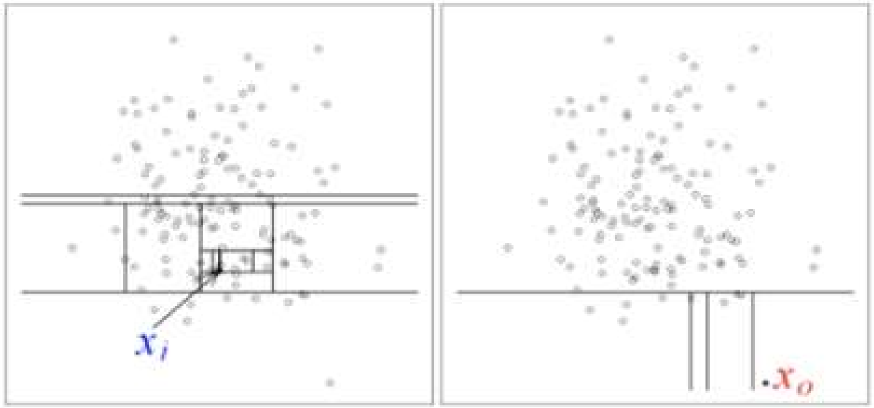


"If intelligence is a cake, the bulk of the cake is unsupervised learning,
the icing on the cake is supervised learning,
and the cherry on the cake is reinforcement learning (RL)."

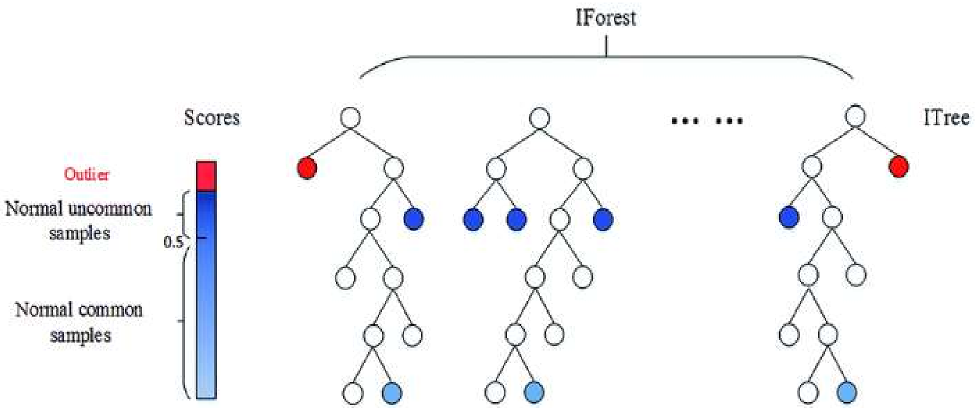
4.1 Statistical ML based

✓ Isolation Forest (F. T. Liu et al., 2008)

- Tree 기반의 split로 데이터를 고립(Isolation 시키는 방법)
  - 비정상 데이터가 고립되려면 낮은 depth를 가짐
  - 정상 데이터가 고립되려면 깊은 depth를 가지는 원리에 착안



〈Isolating normal sample〉〈Isolating abnormal sample〉

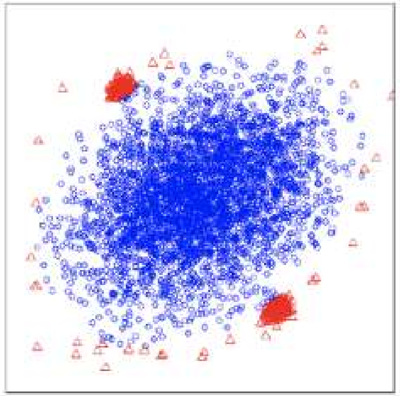


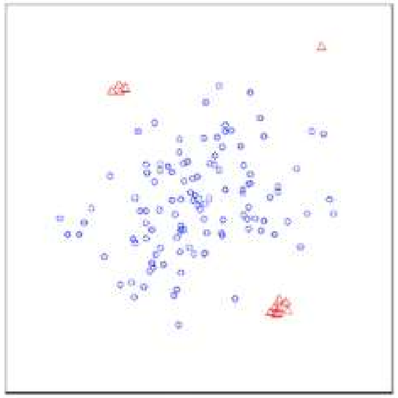
〈출 처 : https://donghwa-kim.github.io/iforest.html〉

4.1 Statistical ML based

✓ Isolation Forest (F. T. Liu et al., 2008)

- 학습순서
  - 데이터 부분 샘플링 → Tree split → Outlier score 계산(학습)





h(x) : 해당 샘플의 경로길이
E(h(x)) : 각 tree에서 해당 샘플의 평균 경로 길이
c(n) : tree n개의 평균경로 길이를 정규화 하기 위한 상수
n : tree 개수 , s : outlier score

If E(h(x)) == c(n), s = 0.5
If E(h(x)) == 0, s = 1
If E(h(x)) == n−1, s = 0

$$s(x, n) = 2^{-\frac{E(h(x))}{c(n)}}$$

〈출 처 : https://donghwa-kim.github.io/iforest.html〉

- 오픈소스 라이브러리 : sklearn.ensemble.IsolationForest
  - https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html

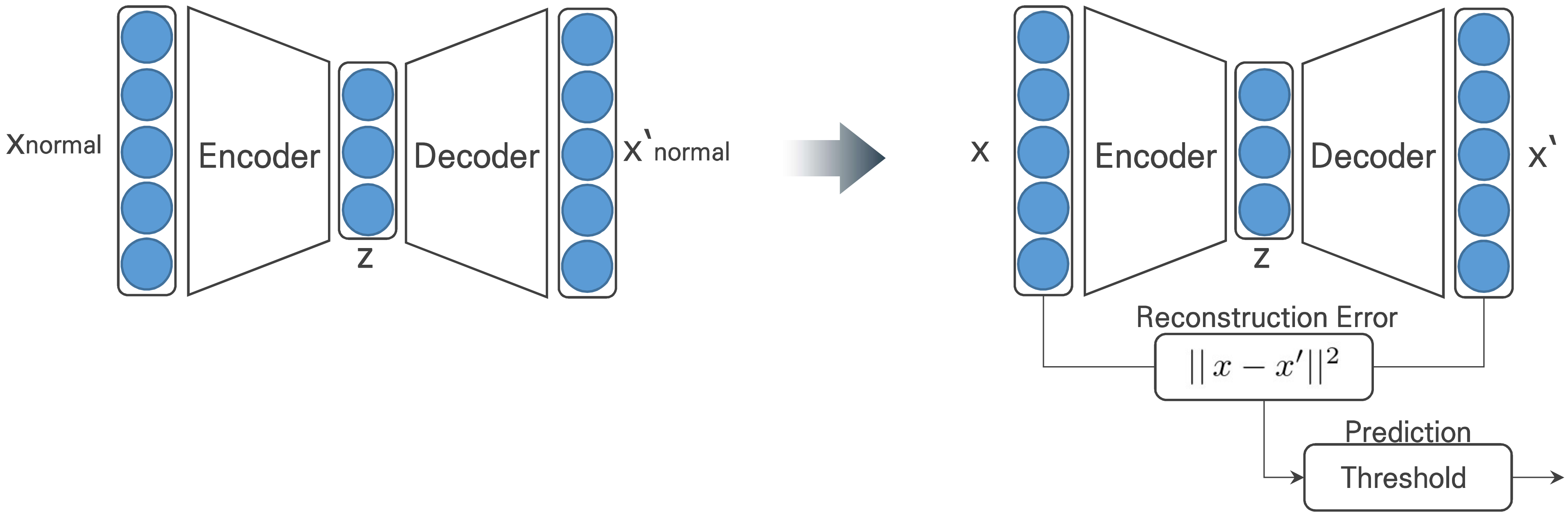
```
from sklearn.ensemble import IsolationForest
model = IsolationForest(n_estimators=100, max_samples=0.25,
                        contamination=0.15,random_state=7, verbose=0)
```

4.2 Representation learning

✓ Autoencoders

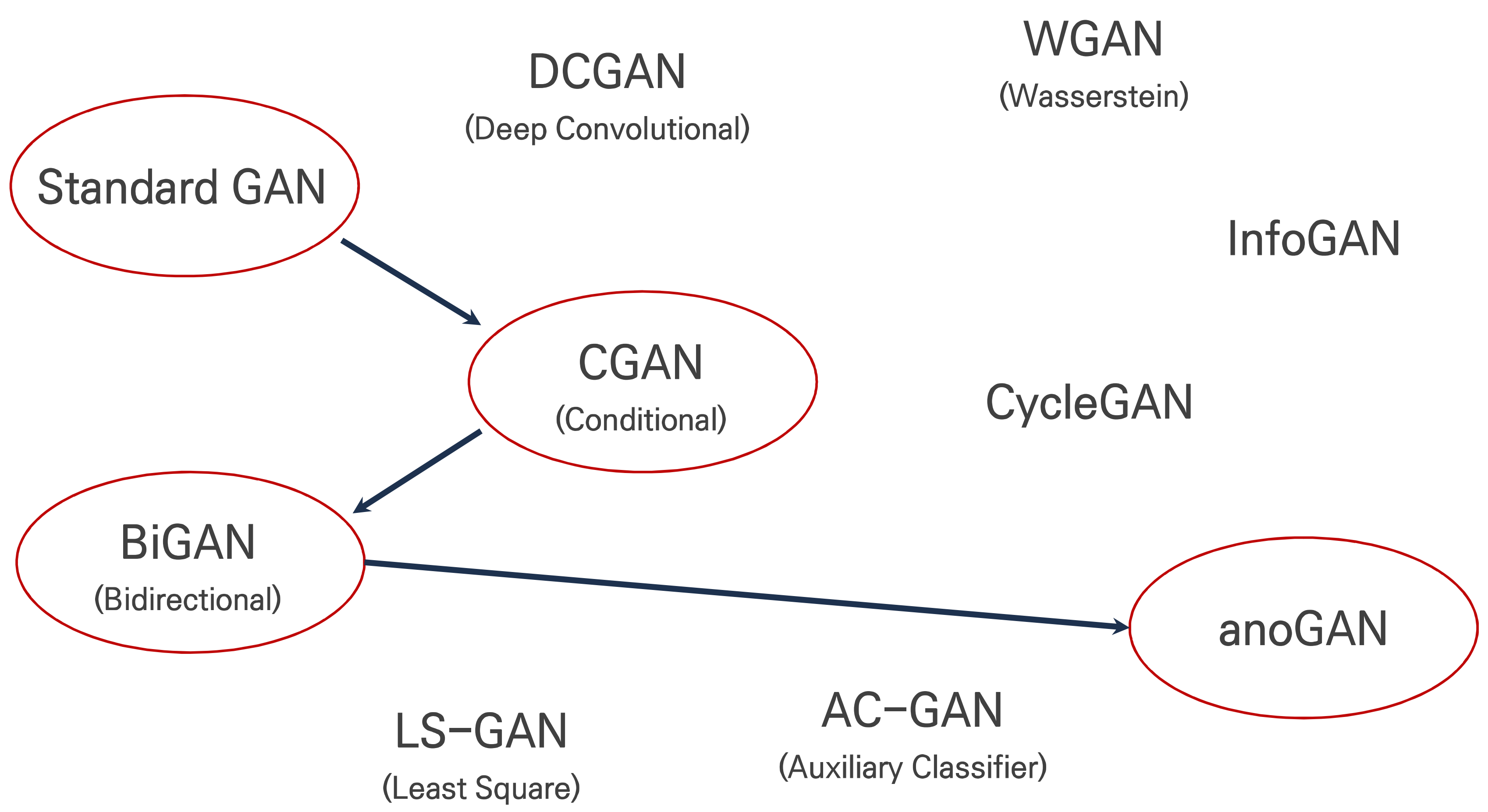
- (가급적) 정상 데이터로 Autoencoder를 학습
- Threshold를 설정하고 모델 예측 결과(Reconstruction Error, ε)가 Threshold 초과 시 이상으로 판정
  - ε 〉 Threshold 이면, 이상
  - ε 〈 Threshold 이면, 정상

〈Step 1: Training using normal data〉 〈Step 2: Prediction〉



4.3 GAN(Generative Adversarial Network) based

✓ GANs for anomaly detection

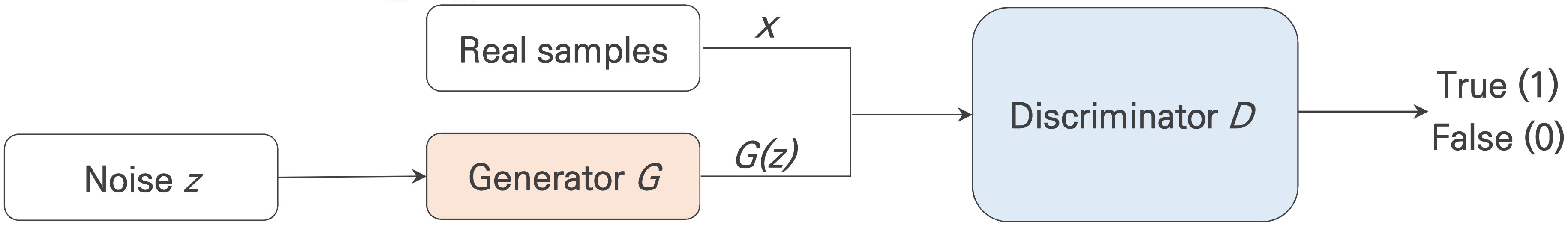


4.3 GAN(Generative Adversarial Network) based

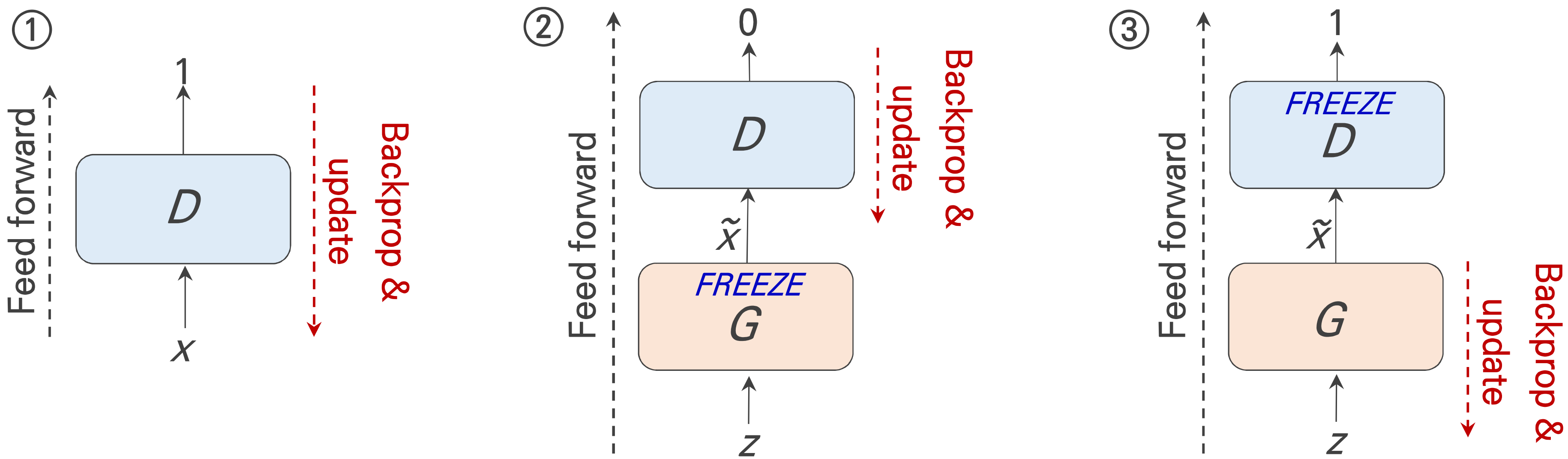
✓ Standard GAN

- 기본구조

$$\min_G \max_D V(D,G) = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))]$$



- 학습방법



4.3 GAN(Generative Adversarial Network) based

✓ Standard GAN

- 구현(Keras)

```
def creat_generator():
    generator = Sequential()

    generator.add(Dense(units = 256, input_dim = noise_dim))
    generator.add(LeakyReLU(0.2))

    generator.add(Dense(512))
    generator.add(LeakyReLU(0.2))

    generator.add(Dense(1024))
    generator.add(LeakyReLU(0.2))

    generator.add(Dense(img_rows*img_rows*channels,
                        activation = 'tanh'))

    generator.compile(loss='binary_crossentropy',
                      optimizer=optimizer)
    return generator
```

```
def create_descriminator():
    discriminator = Sequential()

    discriminator.add(Dense(1024,
                  input_dim=img_rows*img_cols*channels))

    discriminator.add(LeakyReLU(0.2))

    discriminator.add(Dense(512))
    discriminator.add(LeakyReLU(0.2))

    discriminator.add(Dense(256))
    discriminator.add(LeakyReLU(0.2))

    discriminator.add(Dense(1, activation='sigmoid'))

    discriminator.compile(loss='binary_crossentropy',
                          optimizer=optimizer)
    return discriminator
```

4.3 GAN(Generative Adversarial Network) based

✓ Standard GAN

- 구현(Keras)

```
discriminator = create_descriminator()
generator = creat_generator()

discriminator.trainable = False
# generator 학습할 때 discriminator는 학습하지 않도록 설정

gan_input = Input(shape=(noise_dim,))
fake_image = generator(gan_input)

gan_output = discriminator(fake_image)

gan = Model(gan_input, gan_output)
gan.compile(loss='binary_crossentropy', optimizer=optimizer)
```

```
for epoch in range(epochs):
    for batch in range(steps_per_epoch):
        noise = np.random.normal(0, 1, size = (batch_size,
                                               noise_dim))

        fake_x = generator.predict(noise)
        real_x = X_train[np.random.randint(0, X_train.shape[0],
                         size = batch_size)]
        x = np.concatenate((real_x, fake_x))

        disc_y = np.zeros(2*batch_size)
        disc_y[:batch_size] = 0.9

        d_loss = discriminator.train_on_batch(x, disc_y)
        # 먼저 discriminator를 학습

        y_gen = np.ones(batch_size)
        g_loss = gan.train_on_batch(noise, y_gen)
```

4.3 GAN(Generative Adversarial Network) based

✓ Standard GAN

- GAN 학습 속도 개선을 위한 방법
  - Normalize the inputs between -1 and 1
  - Use soft labels(ex: 0.9)
  - Avoid sparse gradients : ReLU, MaxPooling
  - Use ADAM Optimizer

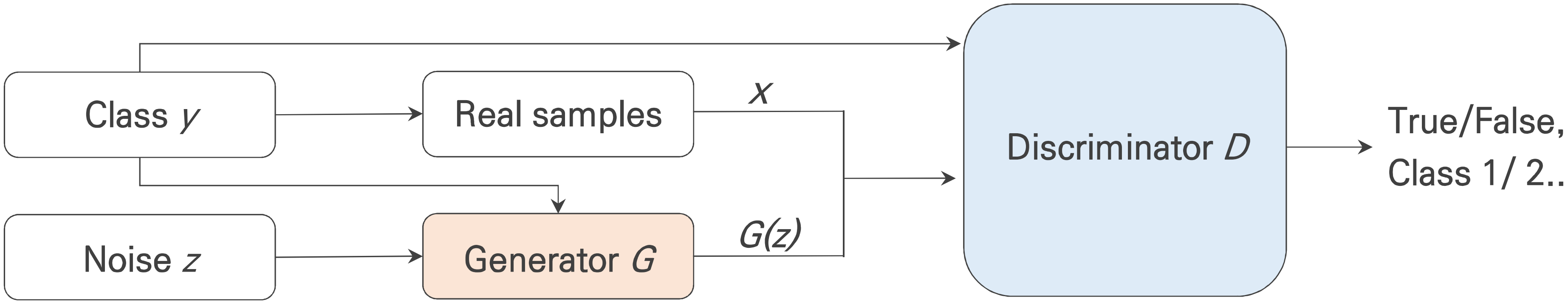
〈출처 : How to Train a GAN? (at NIPS 2016), https://github.com/soumith/ganhacks〉

4. Unsupervised learning

4.3 GAN(Generative Adversarial Network) based

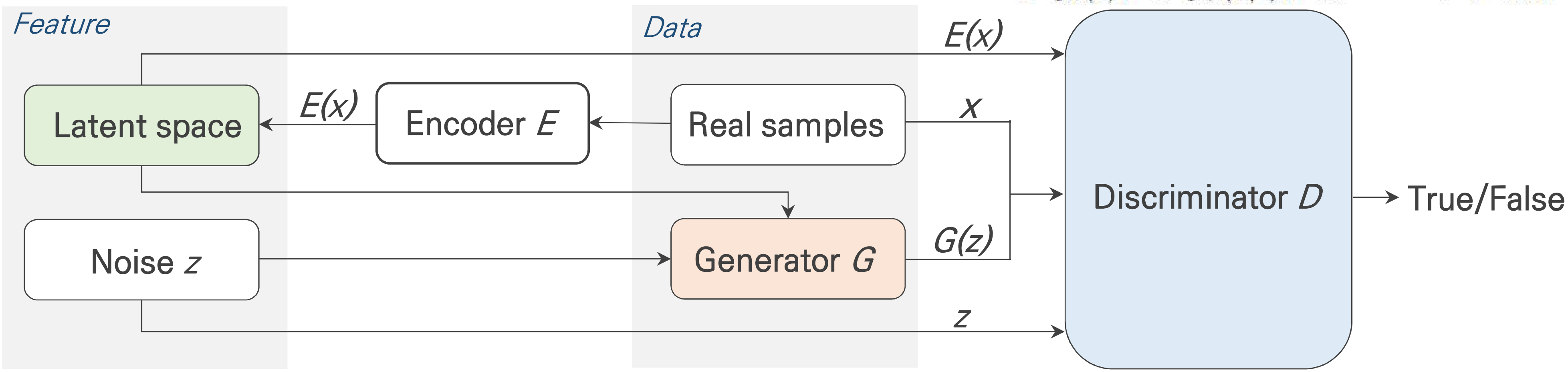
✓ Conditional GAN (CGAN)

$$\min_{G}\max_{D} V(D,G) = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x}|y)}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})}[\log(1 - D(G(\mathbf{z}|y)))].$$



✓ Bidirectional GAN (BiGAN)

$$\min_{G,E}\max_{D} V(D,G,E) = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\mathbb{E}_{\mathbf{z}\sim p_E(\mathbf{z}|\mathbf{x})}[\log D(\mathbf{x},\mathbf{z})]] + \mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})}[\mathbb{E}_{\mathbf{x}\sim p_G(\mathbf{x}|\mathbf{z})}[\log(1 - D(\mathbf{x},\mathbf{z}))]]$$

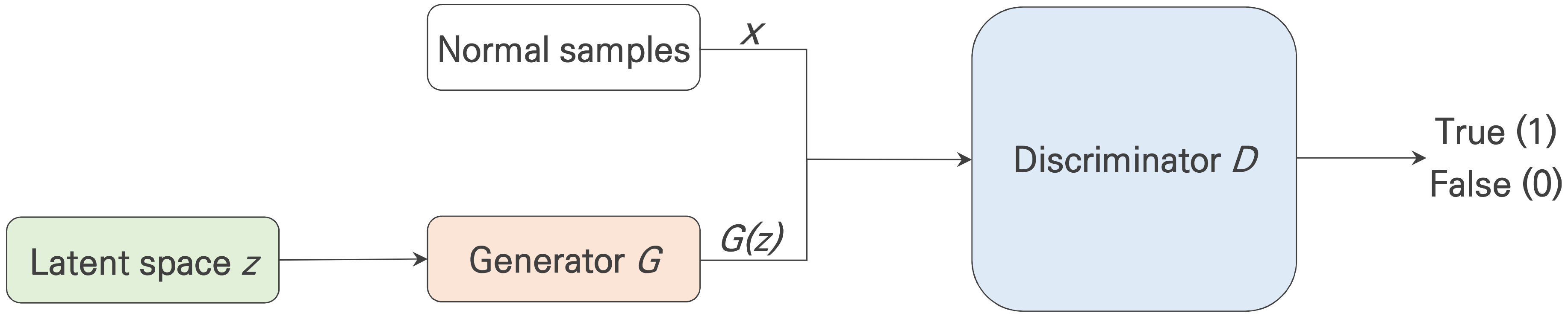


4.3 GAN(Generative Adversarial Network) based

✓ anoGAN (T. Schlegl et al., 2017)

- 기본구조

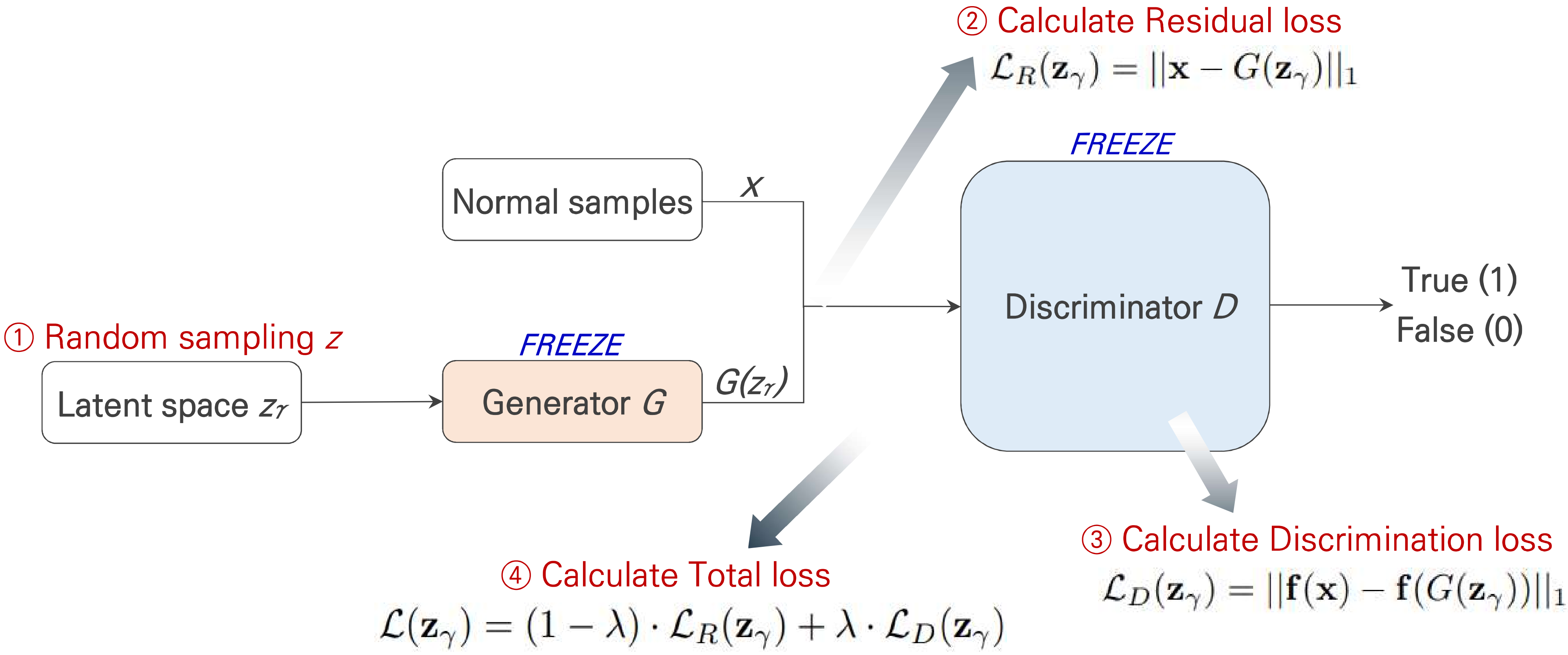
$$\min_{G}\max_{D} V(D,G) = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{z}(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))]$$



4.3 GAN(Generative Adversarial Network) based

✓ anoGAN (T. Schlegl et al., 2017)

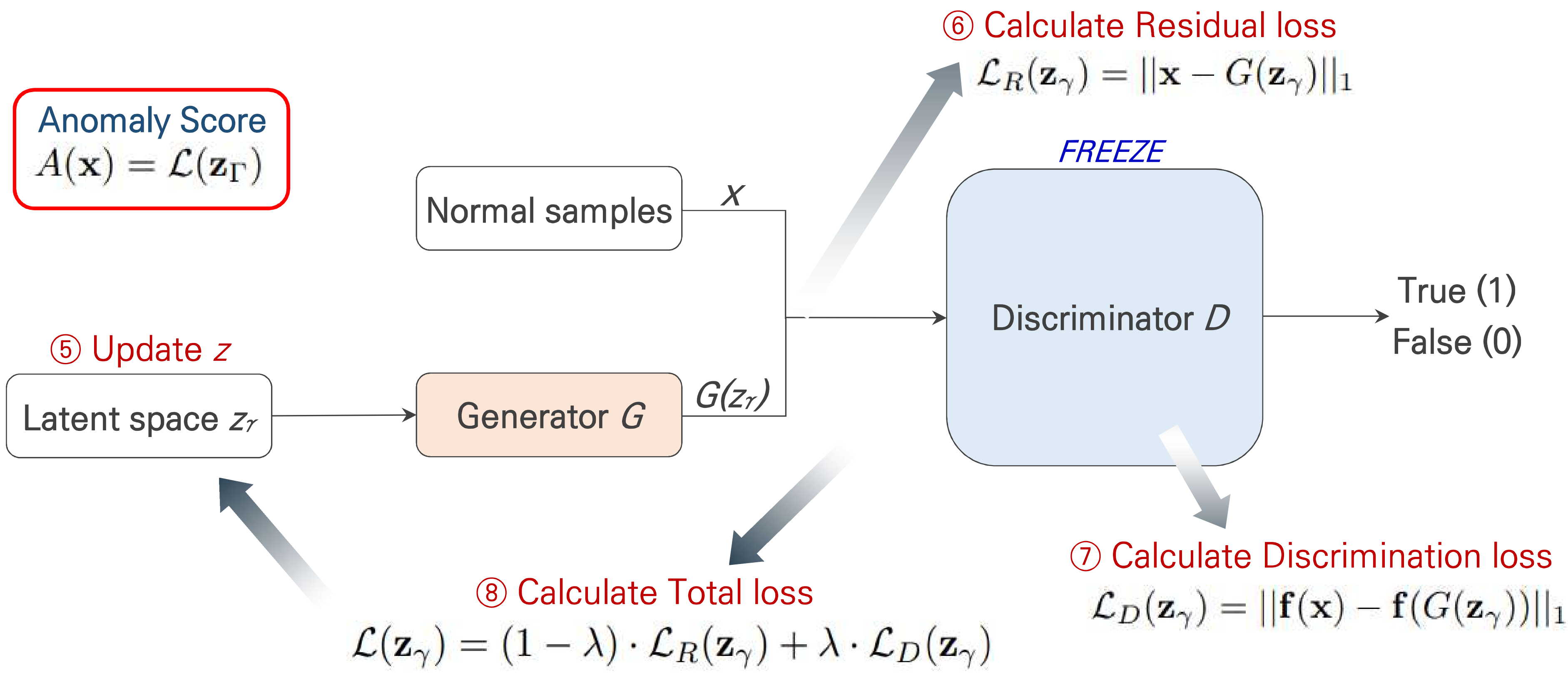
- 학습순서 : GAN Training → Loss Calculation → Latent space update
  - Loss Calculation : Residual loss + Discrimination loss



4.3 GAN(Generative Adversarial Network) based

✓ anoGAN (T. Schlegl et al., 2017)

- 학습순서 : GAN Training → Loss Calculation → Latent space update
  - Latent space update : Total loss를 사용하여 z_γ을 업데이트



4.3 GAN(Generative Adversarial Network) based

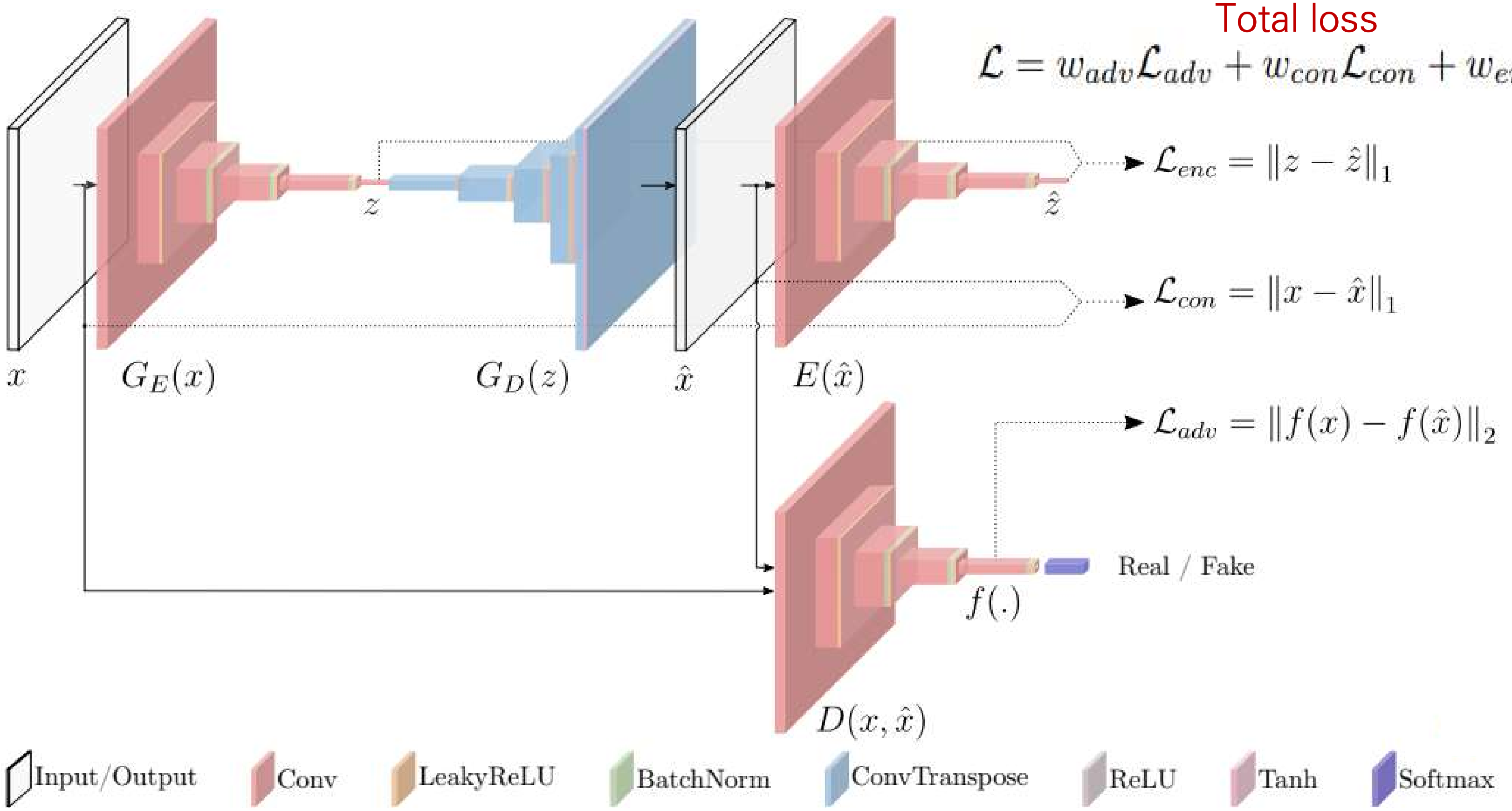
✓ GANomaly(S. Akcay et al., 2018)

- 기본구조

$$s'_i = \frac{s_i + \min(S)}{\max(S) - \min(S)} \qquad \mathcal{A}(\mathbf{x}) = ||G_E(\mathbf{x}) - E(G(\mathbf{x}))||_2 \qquad \mathcal{S} = \left\{s_i : \mathcal{A}(\hat{x}_i), \hat{x}_i \in \hat{D}\right\}$$

Total loss

$$\mathcal{L} = w_{adv}\mathcal{L}_{adv} + w_{con}\mathcal{L}_{con} + w_{enc}\mathcal{L}_{enc}$$



4.4 장단점

- Pros
  - 레이블링 과정이 필요없다.
  - 관련 연구가 활발하게 진행되고 있다.

- Cons
  - 지도학습/ 반지도학습 방법과 비교하면 검출 정확도가 낮다.
  - Hyperparameter(Latent space size, Thresholds, anomaly score weight 등)에 민감하다.

1. Backgrounds

2. Approach 1 : Supervised learning

3. Approach 2 : Semi-supervised/ Hybrid learning

4. Approach 3 : Unsupervised learning

5. Case study : Credit card fraud detection

5. Case study : Credit card fraud detection

5.1 데이터 개요

✓ Credit Card Fraud Detection Dataset

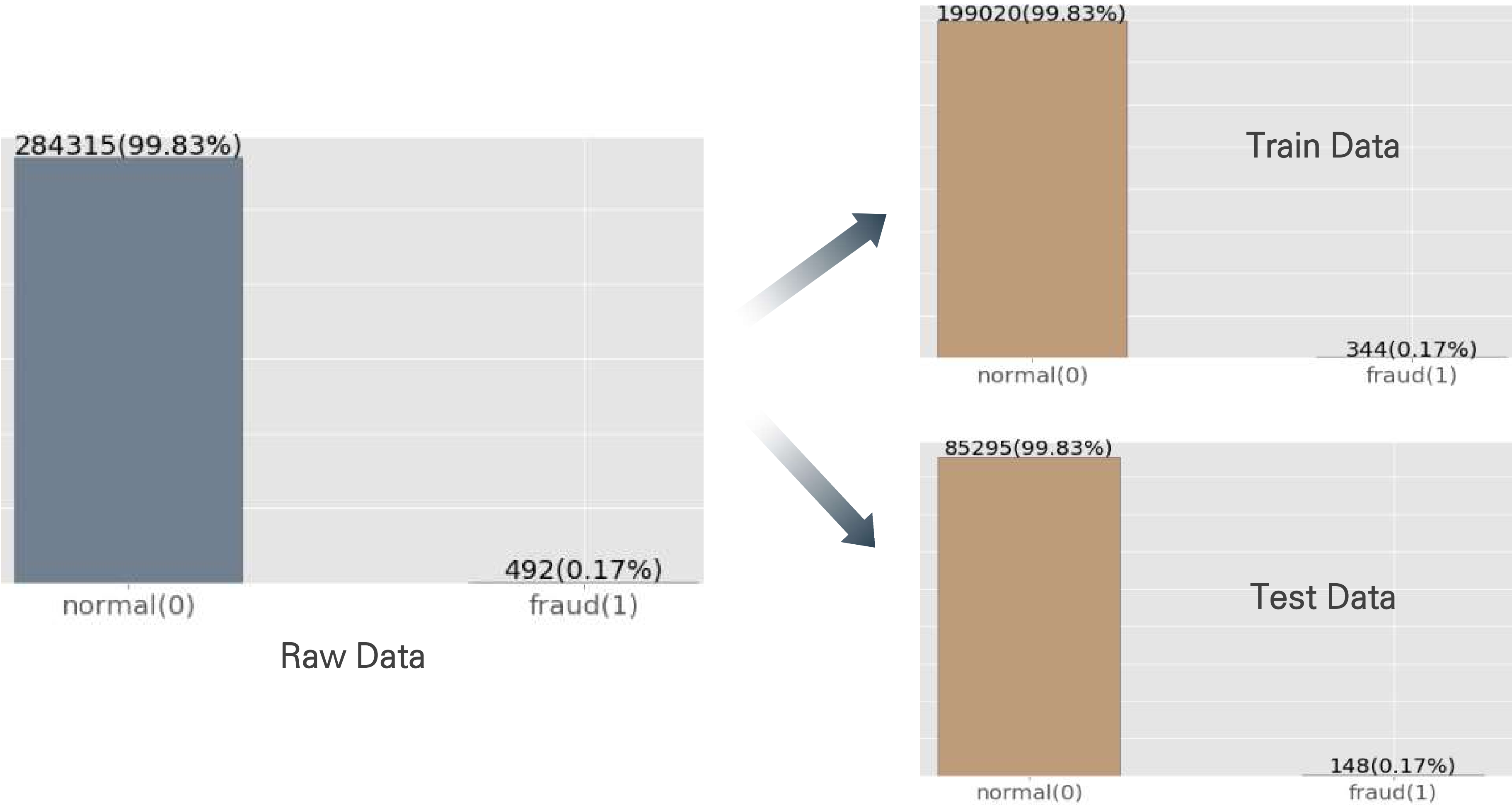
- ULB(Université Libre de Bruxelles) Machine Learning Group에 의해 수집된 '13년 9월의 유럽지역 신용카드 거래정보
- 보안을 위해 원본데이터를 PCA를 통해 비식별화 처리함(피처정보 확인불가)
- **피처(Features)** : Time(거래 진행시간), Amount(거래금액), V1 ~ 20
- **타겟(Target)** : 'Class' 컬럼(1: Fraud, 0: Normal)
- **데이터 shape** : (284,807, 31)

	Time	V1	V2	V3	V4	…	V26	V27	V28	Amount	Class
0	0.0	-1.359807	-0.072781	2.536347	1.378155		-0.189115	0.133558	-0.021053	149.62	0
1	0.0	1.191857	0.266151	0.166480	0.448154		0.125895	-0.008983	0.014724	2.69	0
2	1.0	-1.358354	-1.340163	1.773209	0.379780		-0.139097	-0.055353	-0.059752	378.66	0
3	1.0	-0.966272	-0.185226	1.792993	-0.863291		-0.221929	0.062723	0.061458	123.50	0
4	2.0	-1.158233	0.877737	1.548718	0.403034		0.502292	0.219422	0.215153	69.99	0

5.2 전처리

✓ Preprocessing & Feature engineering

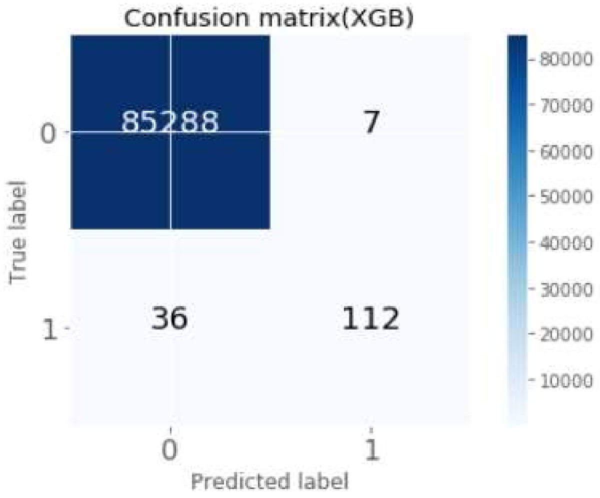
- 탐색적 분석(EDA) → 스케일링(Log, MinMax scaling) → 데이터 분리(X_train, y_train, X_test, y_test)

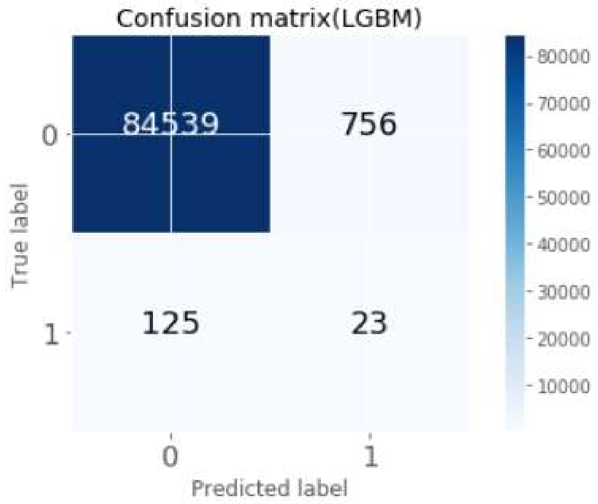


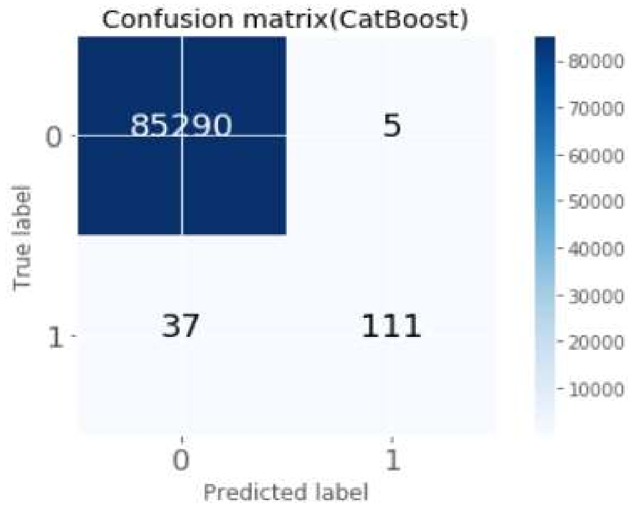
5. Case study : Credit card fraud detection

5.3 모델 평가

✓ Baseline model





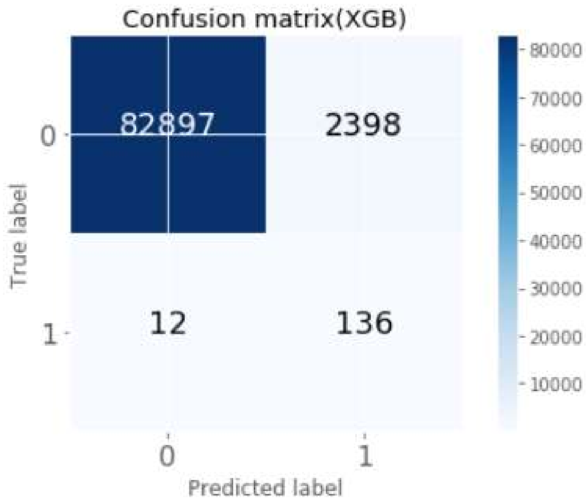


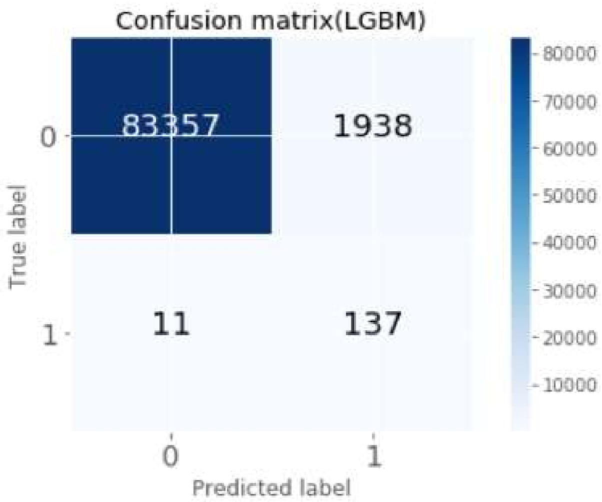
- Xgboost
  - Accuracy : 99.95%
  - Recall : 75.68%
  - Precision : 94.12%
- LightGBM
  - Accuracy : 98.97%
  - Recall : 15.54%
  - Precision : 2.95%
- CatBoost
  - Accuracy : 99.96%
  - Recall : 75%
  - Precision : 95.69%

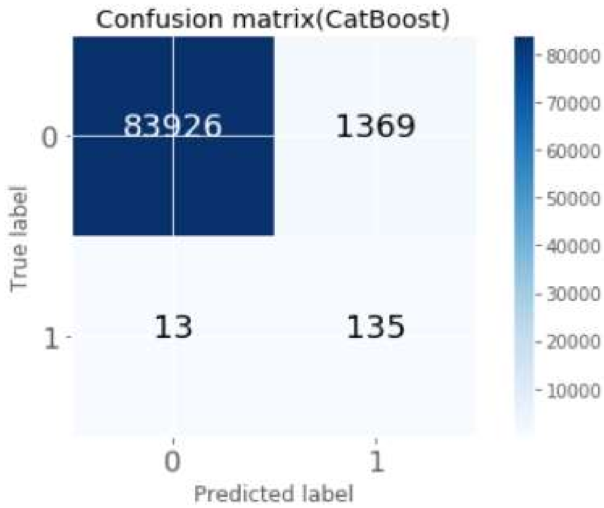
5. Case study : Credit card fraud detection

5.3 모델 평가

✓ Supervised learning : Undersampling

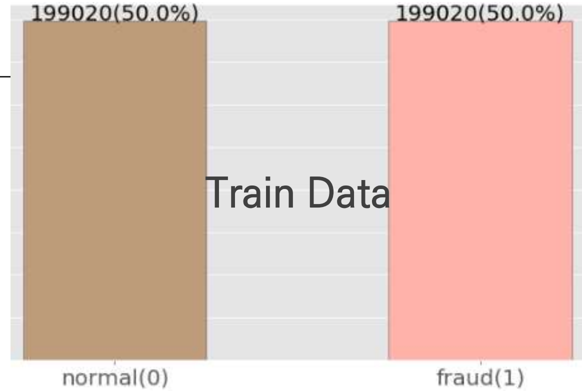






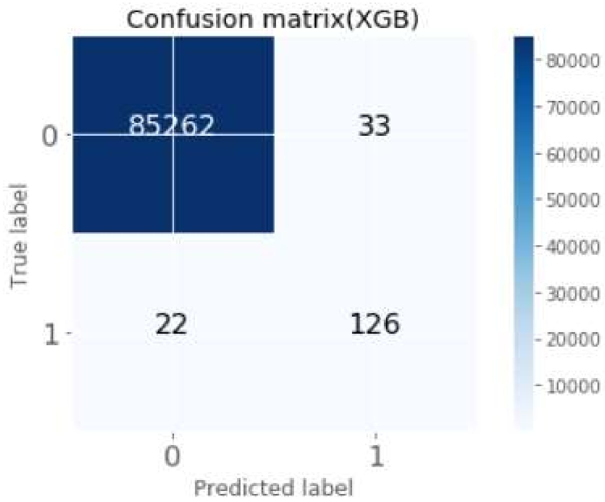
- Xgboost
  - Accuracy : 97.18%
  - Recall : 91.89%
  - Precision : 5.37%
- LightGBM
  - Accuracy : 97.71%
  - Recall : 92.57%
  - Precision : 6.6%
- CatBoost
  - Accuracy : 98.38%
  - Recall : 91.22%
  - Precision : 8.98%

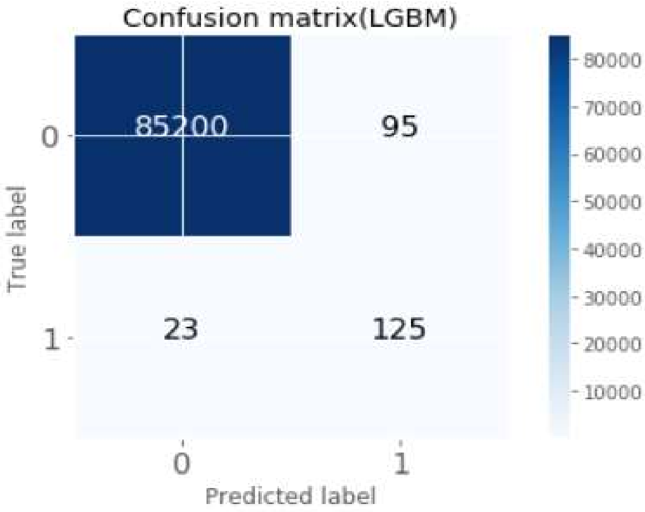
5. Case study : Credit card fraud detection

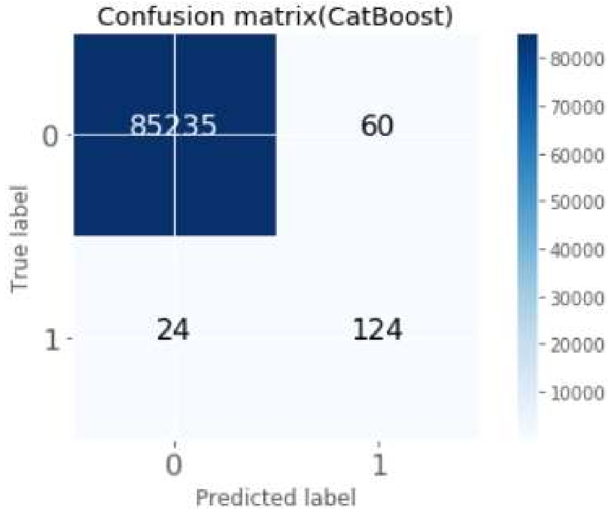


5.3 모델 평가

✓ Supervised learning : Oversampling

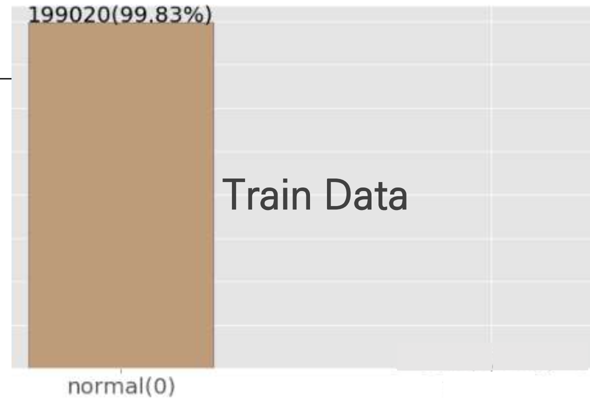






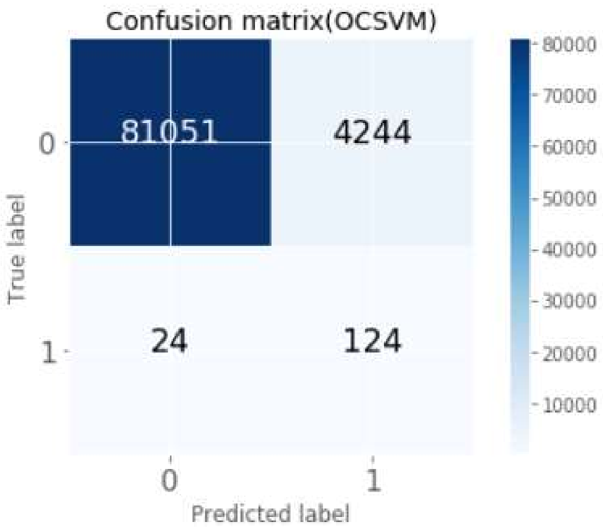
- Xgboost
  - Accuracy : 99.93%
  - Recall : 85.14%
  - Precision : 79.25%
- LightGBM
  - Accuracy : 99.86%
  - Recall : 84.46%
  - Precision : 56.82%
- CatBoost
  - Accuracy : 99.9%
  - Recall : 83.78%
  - Precision : 67.39%

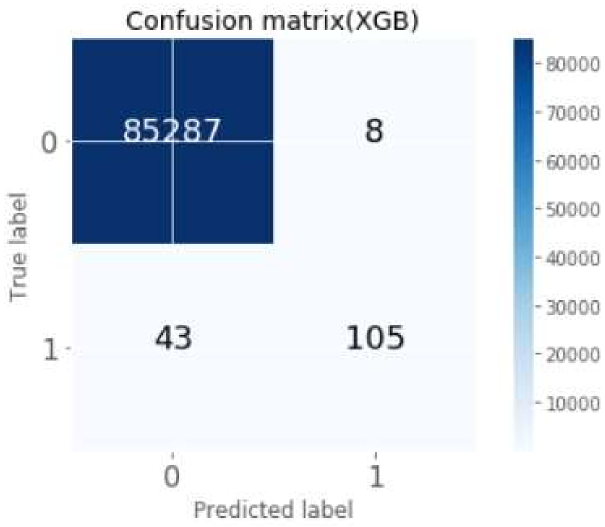
5. Case study : Credit card fraud detection

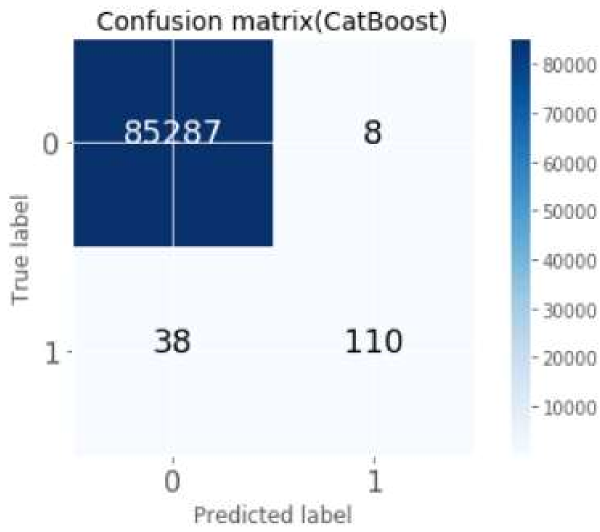


5.3 모델 평가

✓ Semi-supervised learning





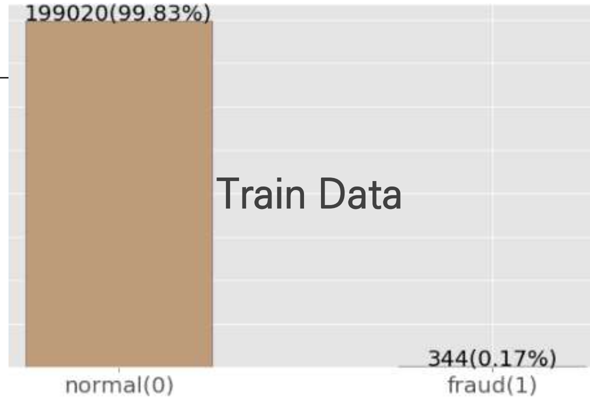


- One-Class SVM
  - Accuracy : 95%
  - Recall : 83.78%
  - Precision : 2.84%

- AE + Xgboost
  - Accuracy : 99.94%
  - Recall : 70.95%
  - Precision : 92.92%

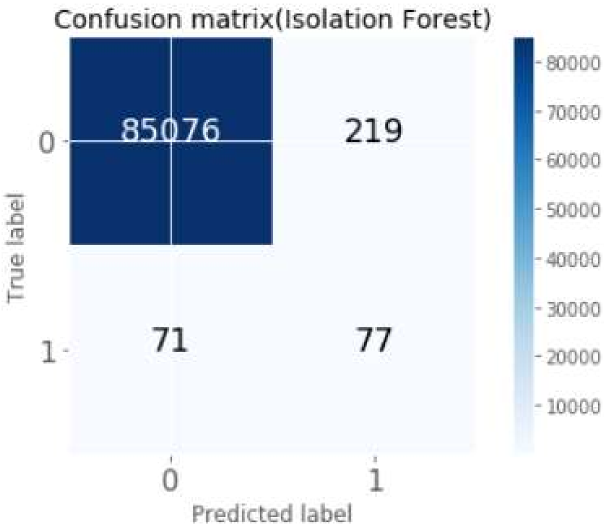
- AE + CatBoost
  - Accuracy : 99.95%
  - Recall : 74.32%
  - Precision : 93.22%

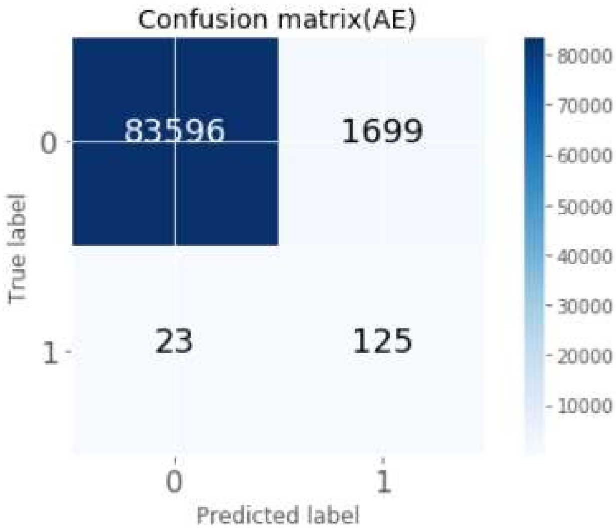
5. Case study : Credit card fraud detection

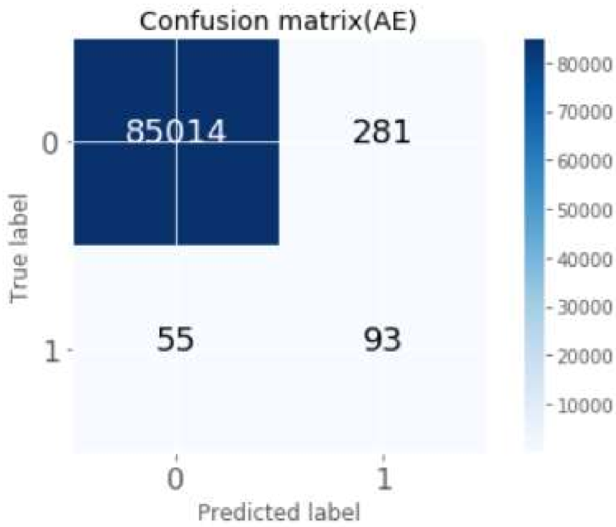


5.3 모델 평가

✓ Unsupervised learning







- Isolation Forest
  - Accuracy : 99.66%
  - Recall : 52.02%
  - Precision : 26.01%

- Autoencoder (th : 0.001)
  - Accuracy : 97.98%
  - Recall : 84.46%
  - Precision : 6.85%

- Autoencoder (th : 0.0035)
  - Accuracy : 99.61%
  - Recall : 62.84%
  - Precision : 24.87%

참고자료

[1] B. Schölkopf, J. Platt, A. Smola, Kernel method for percentile feature extraction, Tech. Rep. TR MSR 2000-22, Microsoft Research, Redmond, WA, 2000.

[2] Tax, D. M. J. and Duin, R. P. W. Support Vector Data Description. Machine learning, 54(1):45–66, 2004

[3] Ruff, L. , Vandermeulen, R., et al. Deep one-class classification, pp. 4393–4402, 2018.

[4] Liu, F. T., Ting, K. M., and Zhou, Z.-H. Isolation Forest. In ICDM, pp. 413–422, 2008.

[5] An, J. and Cho, S. Variational Autoencoder based Anomaly Detection using Reconstruction Probability. SNU Data Mining Center, Tech. Rep., 2015.

[6] Goodfellow, I., Bengio, Y., et al., Generative Adversarial Nets. pp. 2672–2680, 2014.

[7] Mirza, M. and Osindero, S. Conditional Generative Adversarial Nets. arxiv/1411.1784, 2014.

[8] Donahue, J., Krhenbhl, P., and Darrell, T. Adversarial Feature Learning. arxiv/1605.09782, 2016.

[9] Schlegl, T., Seebck, P., Waldstein, S. M., Schmidt-Erfurth, U., and Langs, G. Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery. arxiv/1703.05921, 2017.

[10] Zenati, H., Foo, C. S., Lecouat, B., Manek, G., et al. Efficient GAN-Based Anomaly Detection. arxiv/1802.06222, 2018.

[11] Akcay, S., Abarghouei, A. A., and Breckon, T. P. GANomaly: Semi-Supervised Anomaly Detection via Adversarial Training. arxiv/1805.06725, 2018.

[12] Mattia, F. D., Galeone, P., Simoni, M. D., Ghelfi, M., A Survey on GANs for Anomaly Detection. arxiv/ 1906.11632, 2019

[13] Chalapathy, R., Chawla, S., Deep Learning for Anomaly Detection. arxiv/ 1901.03407, 2019

[14] Machin Learning Mastery: Imbalanced classification, URL : https://machinelearningmastery.com/category/imbalanced-classification/

[15] Kaggle Dataset, Credit Card Fraud Detection, URL : https://www.kaggle.com/mlg-ulb/creditcardfraud

감사합니다.

Kwang Myung Yu

www.github.com/sguys99 sguys99@naver.com