

DAY 1 / CLASS 3

AI-based Anomaly Detection for Digital Twin



"정상(Normal)을 학습하여 비정상(Abnormal)을 찾아내다"



TIME

13:00 ~ 15:00 (120분)

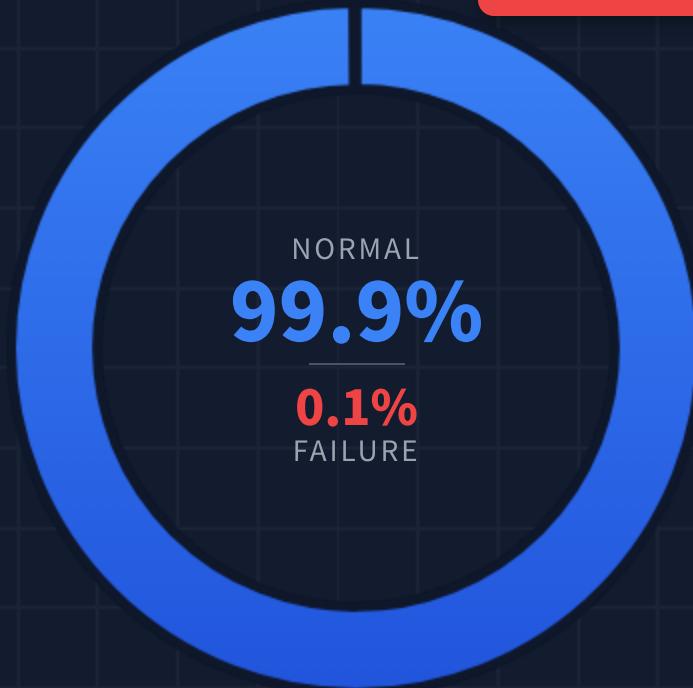


SESSION

PPT(30분) + 실습(90분)

"불량 데이터가 없어요!"

공장은 불량이 나면 안 되는 곳입니다. 그래서 데이터가 없습니다.



⌚ 1년치 데이터 중 고장 데이터는 단 몇 건뿐

🔴 현실의 문제 (Reality)

데이터의 대부분은 '정상'이고, 고장 데이터는 극도로 희귀합니다.
심지어 어떤 고장은 **한 번도 발생한 적이 없습니다**.



❓ 핵심 질문 (Question)

"고장 데이터(오답)가 거의 없는데,
어떻게 AI에게 무엇이 고장인지 가르칠 수 있을까요?"



💡 해결책: 비지도 학습 (Unsupervised)

정답(Label)이 필요 없습니다.
"정상 패턴"만 완벽하게 학습시키고,
거기서 벗어나는 모든 것을 '이상(Anomaly)'으로 간주합니다.

지도학습 vs 비지도학습

"정답이 있는 공부(객관식) vs 정답이 없는 공부(논술형)"



지도학습

Supervised Learning (Classification)

ANALOGY

1
2
3

객관식 시험

"이건 고장(A)이야, 이건 정상(B)이야"



비지도학습

Unsupervised Learning (Anomaly Detection)

VS

ANALOGY



논술형 시험

"답은 없지만, 평소랑 논리가 다른데?"

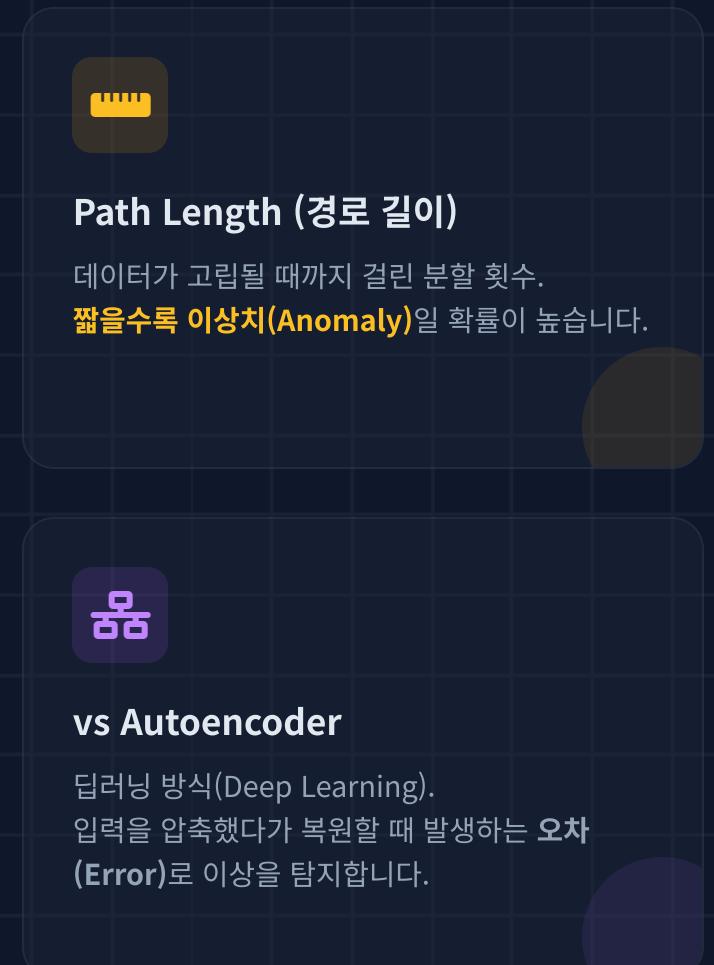
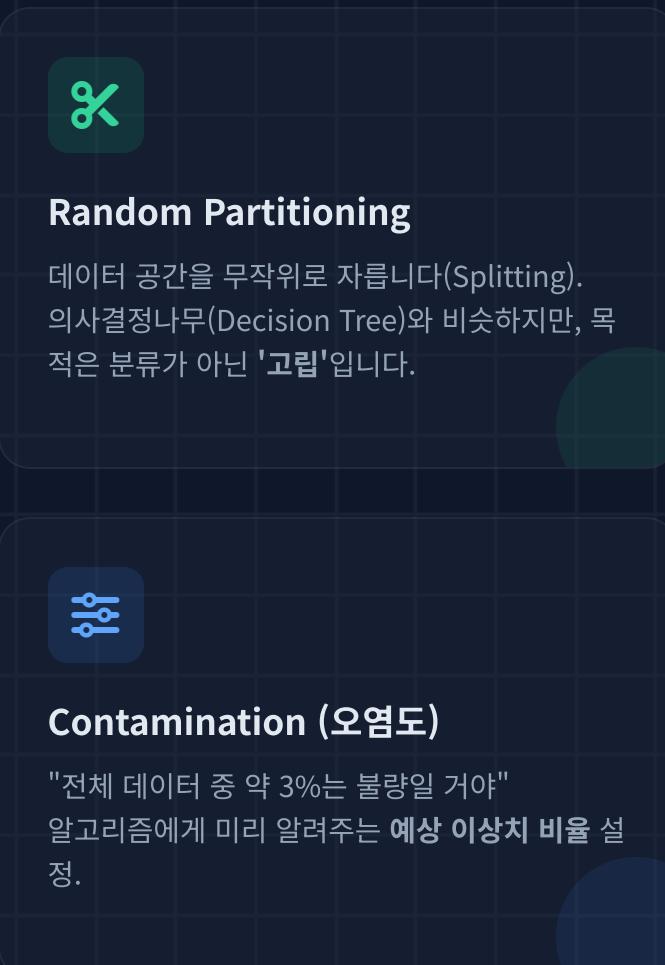
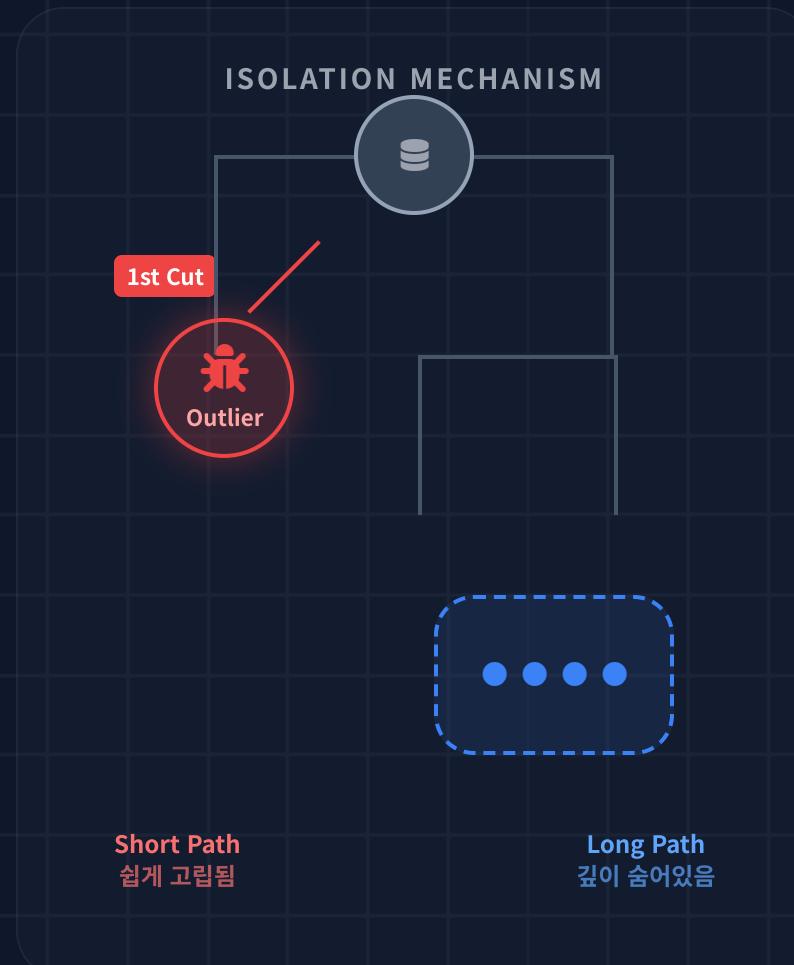
- ✓ **답안지(Label) 필수:** 데이터 하나하나에 '정답'이 있어야 학습 가능함.
- ✓ **암기식 학습:** "내가 본 고장 유형은 기가 막히게 맞춘다."
- ✗ **치명적 단점:** 처음 보는 새로운 유형의 고장(New Fault)은 탐지 못함.

- ✓ **답안지 불필요:** 데이터의 패턴과 분포만 보고 스스로 학습함.
- ✓ **느낌적 학습:** "정상 데이터들은 뭉쳐있는데, 째만 따로 노네?"
- ★ **핵심 장점:** 우리가 모르는 미지의 고장 (**'Unknown Unknowns'**) 탐지 가능.

“ 고장 데이터가 거의 없는 현실 제조 현장에서는 **비지도 학습 (Anomaly Detection)**이 더 실용적입니다. ”

오늘의 무기: Isolation Forest (고립 숲)

“ 평범한 데이터는 뭉쳐 있어서 격리하기 힘들지만, 아싸(Outlier)는 몇 번만 칼질해도 금방 격리된다. ”



디지털 트윈 대시보드란?

▣ "단순한 모니터링을 넘어, 설비의 '건강'을 진단하다"

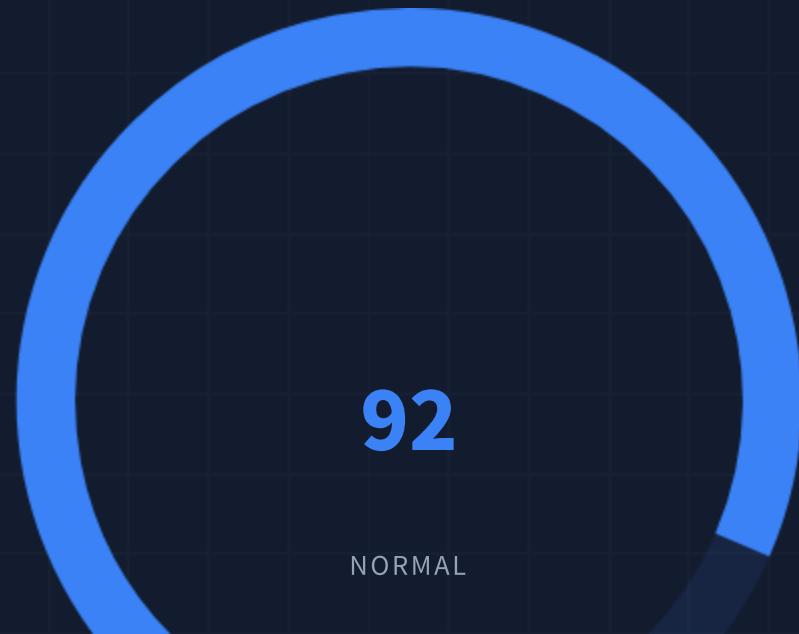


Health Index (건전성 지수)

Conceptual Definition

복잡한 센서 데이터(진동, 전류 등)를 보는 것만으로는 상태를 알 수 없습니다.

이를 종합하여 **0~100점의 직관적인 점수**로 환산해야 합니다.



Key Metrics (핵심 지표)

Practical Application

실전 대시보드는 3가지 핵심 요소로 구성됩니다.

◎ Anomaly Score

현재 얼마나 이상한가?

↗ Trend (경향성)

상태가 나빠지고 있는가?

🔔 Threshold (임계치)

언제 알람을 울릴 것인가?



실습 1 환경 설정 및 데이터 로드

분석에 필요한 라이브러리를 준비하고 UCI AI4I 2020 데이터셋을 불러옵니다.



01_setup_data.py

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split

# 1. 한글 폰트 설정 (Windows/Mac 환경에 맞게)
plt.rc('font', family='NanumBarunGothic')
plt.rc('axes', unicode_minus=False)

# 2. UCI AI4I 2020 데이터셋 로드
url = "https://archive.ics.uci.edu/ml..."
df = pd.read_csv(url)

# 3. 데이터 확인
print(f"전체 데이터 크기: {df.shape}")
display(df.head())
```



Dataset: UCI AI4I 2020

실제 밀링 머신(Milling Machine)의 센서 데이터와 고장 여부를 기록한 예측 정비용 표준 데이터셋입니다.

Column	Desc	Example
Type	제품 등급	L (Low)
Air Temp	공기 온도	298.1 K
Torque	토크(Nm)	42.8 Nm
Failure	고장 여부	0 (Normal)



Practice Goals



Comparative Study

지도학습(Random Forest) vs 비지도학습(Isolation Forest) 성능 비교

실습 2 데이터 준비 (Feature Selection)

제조 설비 상태를 나타내는 핵심 센서 데이터를 선정하고 학습셋과 테스트셋으로 분리합니다.



02_feature_split.py

1. 학습에 사용할 핵심 특성(Feature) 선택

```
features = [  
    'Air temperature [K]', 'Process temperature [K]',  
    'Rotational speed [rpm]', 'Torque [Nm]',  
    'Tool wear [min]'  
]
```

X = df[features]

y = df['Machine failure']

2. 학습/테스트 데이터 분리 (8:2 비율)

```
# random_state=42 : 언제 실행해도 똑같이 섞이도록 고정  
X_train, X_test, y_train, y_test = train_test_split(  
    X, y, test_size=0.2, random_state=42  
)
```

print(f"학습 데이터(Train): {X_train.shape}")

print(f"테스트 데이터(Test): {X_test.shape}")



Feature Selection (X)

기계의 물리적 상태와 직접 연관된 5가지 핵심 센서 데이터를 입력 변수로 사용합니다.

Air Temp

Process Temp

RPM

Torque

Tool Wear



Train / Test Split

전체 데이터를 80%는 AI 학습용으로, 나머지 20%는 성능 검증용으로 분리합니다.

Train Set (80%)

Test (20%)

WHY RANDOM STATE?

- ★ 실험 결과의 재현성(Reproducibility) 보장
- ★ 강사 화면과 수강생 화면의 결과 일치

실습 3 지도학습 - Random Forest

전통적인 지도학습(Supervised Learning) 방식을 적용하여 고장 여부를 예측해 봅니다.



03_supervised_learning.py

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report

# 1. 모델 생성 (정답 y_train을 사용함!)
rf_model = RandomForestClassifier(
    n_estimators=100,
    random_state=42
)

# 2. 학습 (Training with Labels)
rf_model.fit(X_train, y_train)

# 3. 예측 (Prediction)
y_pred_rf = rf_model.predict(X_test)

# 4. 결과 리포트
print("== 지도학습(Random Forest) 성능 평가 ==")
print(classification_report(y_test, y_pred_rf))
```



Supervised Strategy

이 모델은 "답안지(Label)"를 보고 공부했습니다.

"이 패턴은 정상(0)이고, 저 패턴은 고장(1)이야"라고 명확히 알려주었기 때문에 성능이 매우 높게 나옵니다.

Metric	Score
Accuracy	0.99
Precision (1)	0.94
Recall (1)	0.82

⚠ The Reality Gap

실험실에서는 완벽해 보이지만, 실제 현장에서는 "고장 데이터(Label 1)"를 거의 구할 수 없습니다.

고장이 난 적 없는 새 기계라면 이 방식은 사용할 수 없습니다.

실습 4 비지도학습 - Isolation Forest

정상 데이터의 패턴만을 학습하여, 그 패턴에서 벗어나는 이상치를 탐지합니다.



04_unsupervised_learning.py

```
from sklearn.ensemble import IsolationForest

# 1. 모델 생성 (contamination=0.03 : 전체의 3%를 이상치로 가정)
iso_model = IsolationForest(contamination=0.03, random_state=42)

# 2. 학습 (주의: y_train 없이 X_train만으로 학습!)
iso_model.fit(X_train)

# 3. 예측 (-1: 이상, 1: 정상)
y_pred_iso_raw = iso_model.predict(X_test)

# 4. 변환: -1(이상) -> 1(고장), 1(정상) -> 0(정상)
y_pred_iso = [1 if x == -1 else 0 for x in y_pred_iso_raw]

# 결과 확인
print(f"탐지된 이상치 개수: {sum(y_pred_iso)}")
```



Unsupervised Learning

Label-Free: 정답(y)을 보여주지 않고, 데이터(X)의 분포 특성만을 학습합니다. 정상 데이터의 밀도가 높은 영역을 파악합니다.



Key Parameter

contamination=0.03

전체 데이터셋 중 "약 3%가 불량일 것이다"라는 사전 가정을 설정합니다. 이 값을 조절하여 민감도를 튜닝합니다.



Output Mapping

Isolation Forest의 원시 출력값을 우리가 사용하는 0/1 라벨로 변환해야 합니다.

Raw Output



Meaning

Normal

Mapped



실습 5 디지털 트윈 상태 모니터링 시작화

AI 분석 결과를 직관적인 대시보드 그래프로 구현하여 설비의 건강 상태를 한눈에 파악합니다.

```
import matplotlib.pyplot as plt

# 1. 캔버스 설정
plt.figure(figsize=(12, 6))

# 2. 산점도 그리기 (AI 예측 결과: 정상 vs 이상)
# c=y_pred_iso: 0(파랑/정상), 1(빨강/이상)
scatter = plt.scatter(
    X_test['Rotational speed [rpm]'],
    X_test['Torque [Nm]'],
    c=y_pred_iso,
    cmap='coolwarm',
    alpha=0.6,
    label='AI Prediction'
)

# 3. 실제 고장 데이터 표시 (검증용: 노란 별)
actual_faults = X_test[y_test == 1]
plt.scatter(
    actual_faults['Rotational speed [rpm]'],
    actual_faults['Torque [Nm]'],
    color='yellow',
    label='Actual Failure'
)
```

Analysis Strategy

RPM vs Torque 상관관계 분석을 통해 설비의 운전 영역을 시각화합니다. 정상 운전 범위를 벗어난 데이터 포인트들의 군집을 확인합니다.

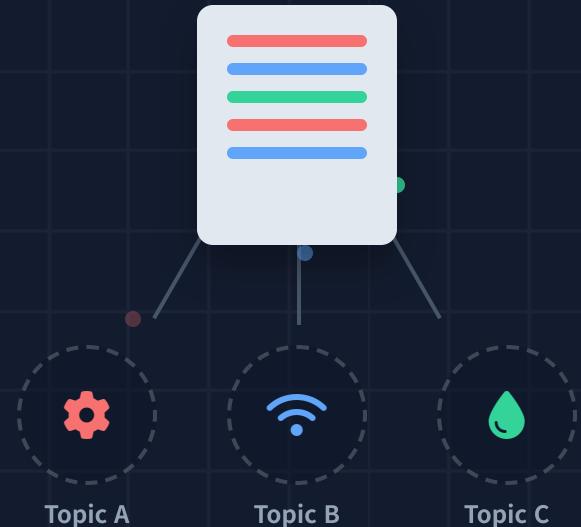
☰ Dashboard Legend

- **AI Normal:** 정상 패턴으로 예측된 데이터
- **AI Anomaly:** 이상 징후가 포착된 데이터 (경보)
- ★ **Actual Failure:** 실제 고장이 발생한 지점

LDA (Latent Dirichlet Allocation)

텍스트 속 숨겨진 주제를 찾아내는 확률적 접근

PROBABILISTIC MODELING



"도서관 사서가 책을 읽지 않고도
키워드만 보고 분류하는 원리"



Concept (개념)

비지도 학습(Unsupervised)

정답(Label)이 없는 상태에서 데이터의 패턴
만으로 주제를 자동 추출합니다.



Assumption (가정)

"문서는 토픽의 혼합, 토픽은 단어의 혼합"
모든 문서는 확률적으로 생성되었다고 가정합
니다.



Structure (구조)

Document → Topics → Words

단어의 동시 등장(Co-occurrence) 패턴을 기
반으로 역추적하는 계층 모델.



Application (응용)

제조 로그 분류, 고객 리뷰 분석, 문서 자동 태
깅 등
대량의 문서를 빠르게 요약하고 분류하는 데
최적.

[보너스] 제조 로그 토픽 모델링 (LDA)

숫자 데이터뿐만 아니라 작업자의 수리 일지(Log)에서 자동으로 주요 고장 원인을 추출합니다.



05_text_lda.py

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.decomposition import LatentDirichletAllocation
```

1. 가상 제조 로그 데이터 (15건)

```
logs = [
    "Motor overheating and shutdown",
    "Software update failed error 404",
    "Oil leak in hydraulic pump system",
    "Bearing noise detected in joint",
    "Connection timeout with server"
    ... # (생략)
]
```

2. 텍스트 벡터화 (단어 빈도수 카운트)

```
vec = CountVectorizer(stop_words='english')
dtm = vec.fit_transform(logs)
```

3. LDA 모델 학습 (3개 토픽으로 분류)

```
lda = LatentDirichletAllocation(n_components=3, random_state=42)
lda.fit(dtm)
```



Structured vs Unstructured

센서 데이터(정형)는 숫자로 명확하지만, **유지보수 로그(비정형)**는 숨겨진 의미를 파악해야 합니다.



LDA Mechanism

"문서는 여러 주제(Topic)의 혼합이다"라는 가정하에, 단어의 동시 등장 패턴을 보고 주제를 확률적으로 추론합니다.



Expected Topics

Topic 1 기계/베어링 (bearing, vibration, noise)

Topic 2 SW/네트워크 (error, timeout, update)

Topic 3 펌프/유압 (pump, leak, pressure)

오늘의 성과와 다음 단계

AI 기반 이상 탐지 및 텍스트 분석 실습 완료

Application
Layer 4: AI Service

 DONE

Cyber
Layer 3: Data Analysis

 DONE

Connectivity
Layer 2: Network

 BASE

Physical
Layer 1: Asset

 BASE

Key Achievements

- 클래스 불균형 이해**
제조 현장의 99.9% 정상 데이터 특성과 **비지도 학습**의 필요성을 파악했습니다.
- Isolation Forest 구현**
데이터 패턴의 고립 정도를 이용해 이상치를 탐지하는 **Isolation Forest**를 실습했습니다.
- Comparative Study**
지도학습(Random Forest)과 비지도학습의 차이를 비교하고, 현실적 **적용 한계**를 이해했습니다.
- Health Index & Dashboard**
설비 건전성 지수(0~100) 개념을 정립하고, AI 예측 결과를 **시각화 대시보드**로 구현했습니다.
- Digital Twin Monitoring**
산점도(Scatter Plot)를 통해 정상/이상 군집을 확인하고 **실시간 모니터링**의 기초를 완성했습니다.
- BONUS Text Mining (LDA)**
제조 로그에서 토픽을 추출하며 **정형/비정형 데이터** 융합 분석 가능성을 확인했습니다.

NEXT LEVEL CHALLENGE (DAY 2)

 LLM Context

 Autoencoder

 Real-time Alert