AI 기반 스마트 팩토리 프로세스 구축

: 열처리 예지보전의 GRU 모델링 및 품질보증의 알고리즘 최적화를 통한 클라우드 서비스 구현

Machine Learning & Cloud Platform

[FutureTech]

김주성, 이길연, 최선재

AI 기반 스마트 팩토리 프로세스 구축

INDEX

AI 기반 스마트 팩토리 프로세스 구축

프로젝트 조직 (구성원 및 역할)

김주성 (팀장)

데이터 전처리, 통계 분석, 머신 러닝

블로그: https://jay-s1.tistory.com/

깃허브: https://github.com/gaf21asd

이길연

데이터 전처리, 통계 분석, 머신 러닝

블로그: https://ls-alt.tistory.com/

깃허브: https://github.com/Gil-Yeon

최선재

데이터 전처리, 시각화, Web 연동

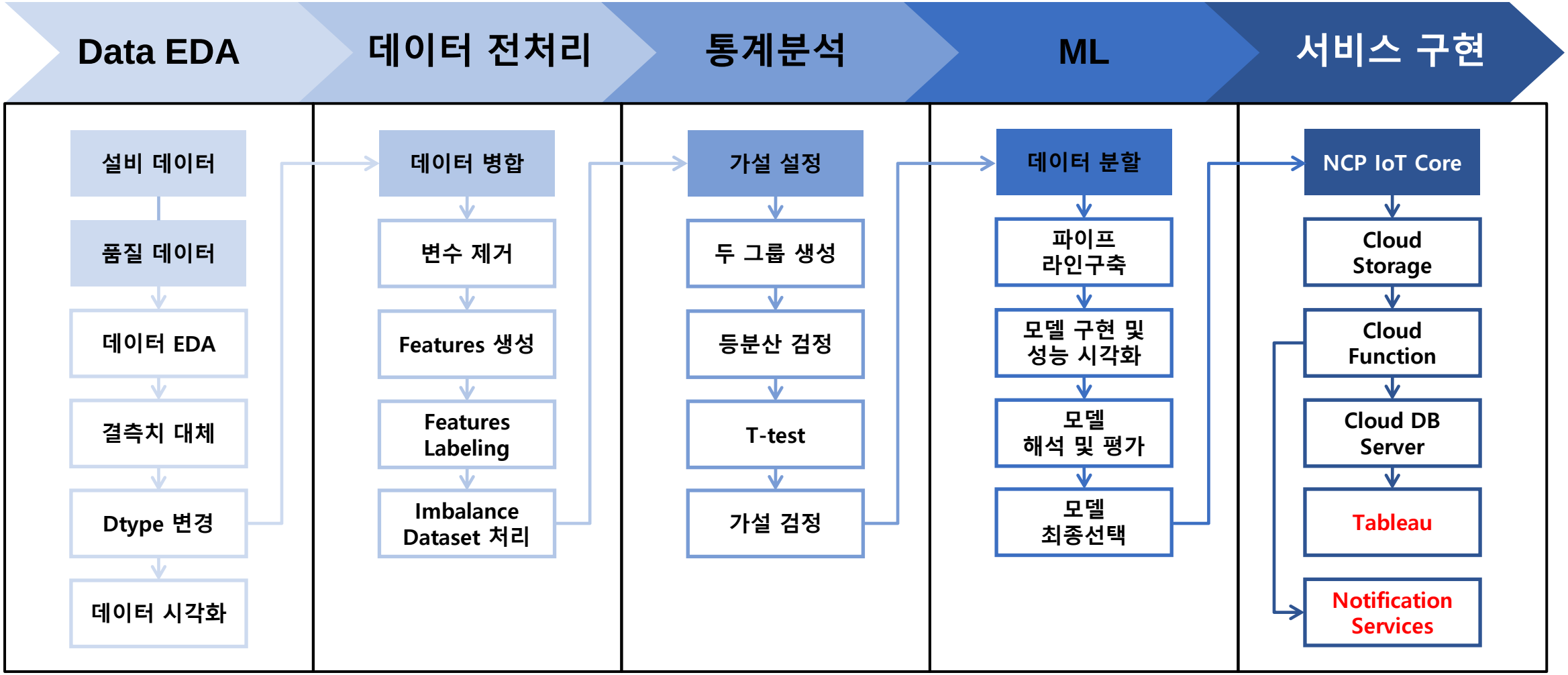
블로그: https://seonjaechoi.tistory.com/

깃허브: https://github.com/seonjaechoi0307

분석 목표

구분	예지보전	품질보증
문제점	설비 문제에 대한 명확한 원인 분석의 애로사항	일련의 생산 과정에서 불량품 확인의 어려움
	설비 고장 시 상당한 비용 및 손실 발생	불량 원인 파악의 부재
해결책	각 설비별로 고장 시점 예측 및 예지보전 실시	실시간 불량품 판별 및 원인 설비 파악
기대효과	생산 효율성 제고 및 품질 향상 기대	불량 리스크(비용 및 신뢰도) 관리 가능

순서도(Flow Chart)



Frameworks

Preprocessing

pandas

Mean

Shap

SHAP

Imbalanced learn

Oversampling

Machine Learning

scikit learn

Logistic Regression Random Forest

Gradient Boosting Extra Trees

XGBoost LightGBM

PYCARET

Deep Learning

TensorFlow

Keras

LSTM GRU

WBS(Work-Breakdown Structure)

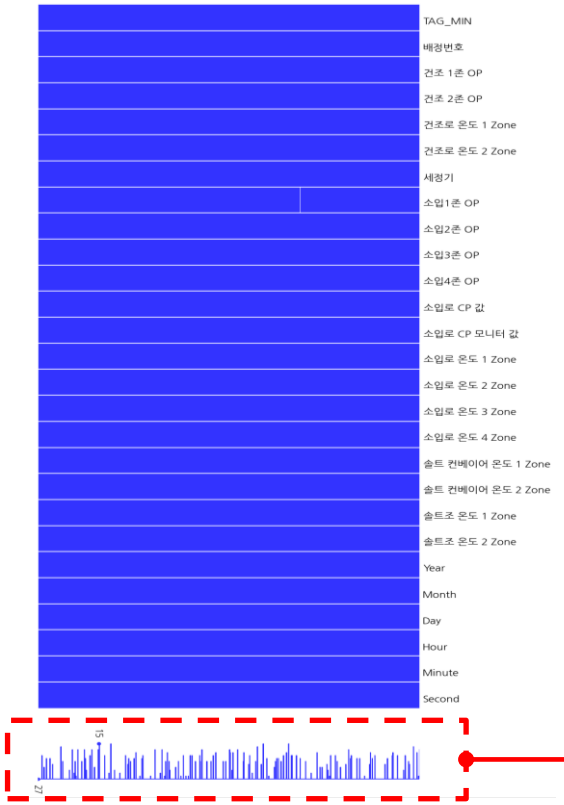
구분		1주차 (11. 01 ~ 11. 07)			2주차 (11. 08 ~ 11. 14)			3주차 (11. 15 ~ 11. 21)			4주차 (11. 22 ~ 11. 28)			5주차 (11. 29 ~ 12. 05)		
프로젝트 기획	데이터 탐색적 분석															
	문제정의 및 가설 설정															
데이터 전처리 가설 검정	결측치 처리 및 결합 가설 검정															
	모델 학습 전 파이프 라인 구축															
모델 생성 및 평가	각 Part 모델 구현 및 평가															
클라우드 서비스	Big Query & Tableau BI 대시보드 구축															
	카카오톡 챗봇 알림 서비스															
프로젝트 PPT 작성 및 발표																

AI 기반 스마트 팩토리 프로세스 구축

INDEX

결측치(Missing Values)

BEFORE



결측치 처리과정

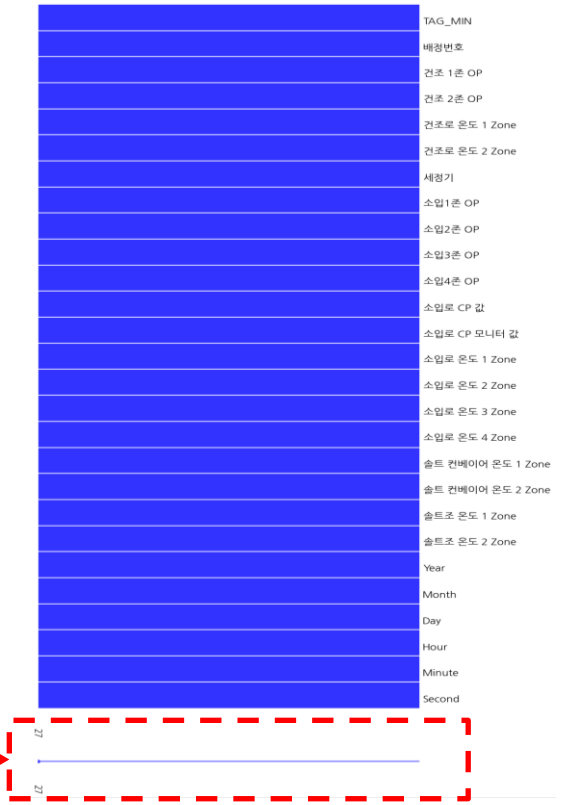
결측치 대체방법

- **Mean**
- Median
- Back Fill
- Impyute Mice

평균값 대체 채택 사유

평균값 대체가 다른 방식에 비해 **모델 성능 향상에 가장 크게 기여**하였다.

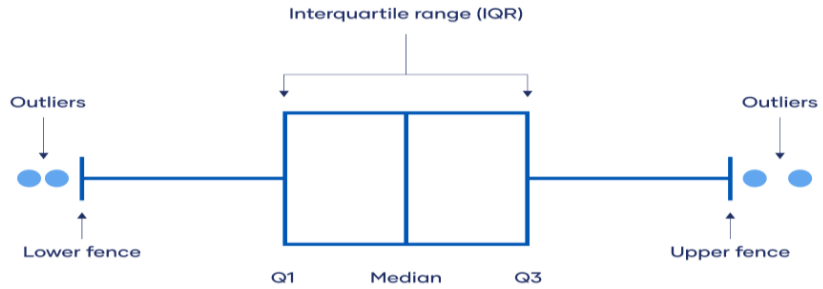
AFTER



예지보전 Part. 01

1. 파생변수 생성

이상치(Outlier) : 이상치 판별을 위해 IQR의 Fence 활용



2. 데이터 그룹화

배정번호별 시간 단위로 설비값은
평균값, 이상치는 총합으로 그룹화
TAG_MIN, 분, 초 변수 제거

3. Target 생성

설비 이상 신호: "설비_Signal"

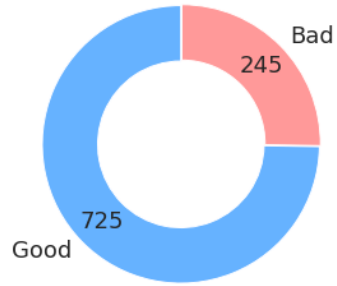
- 0 = 이상치 총합이 3사분위 보다 작으면 정상
- 1 = 이상치 총합이 3사분위 보다 크거나 같으면 비정상

```
# 이상치기준으로 Q3정의
Q3 = np.percentile(df['Outlier'], 75)

# 'Outlier'컬럼 값이 Q3보다 작으면 0, 크거나 같으면 1 할당
df[f'{column}_Signal'] = np.where(df['Outlier'] < Q3, 0, 1)
```

```
df['DZ2_OP_Signal'].value_counts()
```

```
0    725
1    245
Name: DZ2_OP_Signal, dtype: int64
```



예지보전 Part. 02

4. 데이터 정규화

Min-Max Scaler

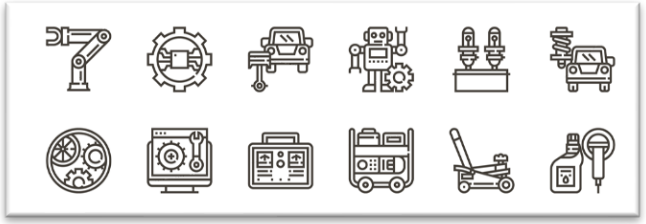
- 최댓값과 최솟값으로 0과 1 사이로 변환

DZ1_OP	DZ2_OP	DZ1_TEMP	DZ2_TEMP	CLEAN	HDZ1_OP	HDZ2_OP	HDZ3_OP
73.2501	24.8670	99.8905	100.1810	69.6934	62.2119	59.0551	50.5300
72.4101	20.4676	99.9949	99.9767	69.6177	83.6938	59.5025	50.4305
70.5052	18.3643	99.9043	100.0814	69.4428	77.1296	59.3487	51.4435
72.8980	21.4173	99.9567	100.4232	70.8956	80.5230	60.4992	51.5050
72.5923	18.1548	99.9917	100.0415	69.2022	76.0577	58.9792	51.7318

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

DZ1_OP	DZ2_OP	DZ1_TEMP	DZ2_TEMP	CLEAN	HDZ1_OP	HDZ2_OP	HDZ3_OP
0.8736	0.7363	0.4722	0.6560	0.8562	0.6239	0.8266	0.6611
0.8329	0.6027	0.5034	0.5826	0.8478	0.8518	0.8459	0.6571
0.7407	0.5388	0.4763	0.6203	0.8286	0.7822	0.8393	0.6987
0.8565	0.6315	0.4920	0.7430	0.9883	0.8182	0.8889	0.7013
0.8417	0.5324	0.5025	0.6059	0.8022	0.7708	0.8233	0.7106

5. 전처리 함수화



각각의 설비(column)에 예지보전 필요

각 설비별 데이터 전처리를 위한 함수 활용

```
def get_pp_df(column):
    df = data.copy()

    # 데이터 컬럼에 이상치 값 0으로
    df['Outlier'] = 0


df = get_pp_df('DZ2_OP')
df.info()
```

```
#   Column      Non-Null Count  Dtype
---  ------      --------------  -----
0   AN          970 non-null    int64
1   Year        970 non-null    int64
2   Month       970 non-null    int64
3   Day         970 non-null    int64
4   Hour        970 non-null    int64
5   DZ1_OP      970 non-null    float64
6   DZ2_OP      970 non-null    float64
7   DZ1_TEMP    970 non-null    float64
```

품질보증 Part. 01

1. 데이터 그룹화

설비 데이터의 배정번호별
Mean과 Std 데이터 생성
TAG_MIN Column 제거

+

품질 데이터의 배정번호별
품질 수량 데이터 불량 단계
외 변수 전부 제거

2. 파생변수 생성

불량률(BQ Rate)

- 불량률 = (불량품수 / 총생산량) * 100%

```
Standard_Total['BQ Rate'] = round(Standard_Total['BQ'] / Standard_Total['TQ'] * 100, 3)
Standard_Total.head(5)
```

불량단계(DS)

- 안전 = 불량률이 3사분위 보다 작으면
- 위험 = 불량률이 3사분위 보다 크거나 같으면

```
# Defective Stage: 불량 단계
# 'BQ Rate' 컬럼의 값이 3사분위값인 0.46보다 크거나 같으면 1, 작으면 0을 'DS' 컬럼에 할당
Standard_Total['DS'] = np.where(Standard_Total['BQ Rate'] >= 0.046, '위험', '안전')

# 불량단계 안전 0은 101개, 위험 1은 35개 배정번호에서 일어났다.
Standard_Total['DS'].value_counts()

# 필요없는 컬럼 삭제
Standard_Total.drop(['AN', 'GQ', 'BQ', 'TQ', 'BQ Rate'], axis=1, inplace=True)
```

품질보증 Part. 02

3. 데이터 라벨링

Label Encoder(종속변수 분류)

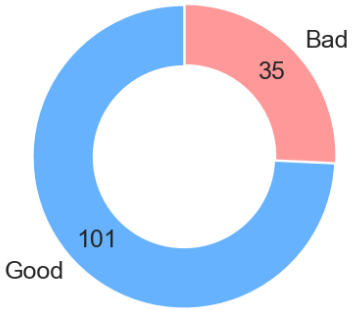
- **불량단계 Binary [안전: 0 / 위험: 1]**

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

# 데이터셋 로드
y = LabelEncoder().fit_transform(Standard_Total['DS'])
```

```
print(pd.Series(y).value_counts())
```

```
0    101
1     35
Name: count, dtype: int64
```



4. 데이터 정규화

Z2_OP_AVG	DZ2_OP_Std	DZ1_TEMP_AVG	DZ1_TEMP_Std	DZ2_TEMP_AVG	DZ2_TEMP_Std	CLEAN_AVG	CLEAN_Std	...	HDZ
21.3545	4.3489	99.9435	0.5939	100.0619	0.4835	69.6026	0.8454	...	
18.6026	2.8597	99.9874	0.5154	100.0650	0.3561	69.5912	1.0642	...	
20.9119	2.5821	99.9956	0.4727	100.0216	0.3430	69.5295	1.0979	...	
22.2502	2.4028	100.0051	0.3314	100.0097	0.2518	69.5369	1.0643	...	
21.8652	3.6228	99.9835	0.6553	100.0437	0.4707	69.3210	0.9917	...	

$$X_{\text{scale}} = \frac{x_i - x_{\text{med}}}{x_{75} - x_{25}}$$

Z2_OP_Std	DZ1_TEMP_AVG	DZ1_TEMP_Std	DZ2_TEMP_AVG	DZ2_TEMP_Std	CLEAN_AVG	CLEAN_Std	...	HDZ4_TEMP_AVG	H
0.9400	-2.8135	1.0709	1.3305	0.7202	0.6605	0.9403	...	0.8475	
-0.3781	-0.7224	0.5080	1.4138	-0.0860	0.6563	1.5501	...	-0.8555	
-0.6238	-0.3341	0.2016	0.2657	-0.1691	0.6330	1.6441	...	0.5667	
-0.7825	0.1161	-0.8124	-0.0509	-0.7469	0.6358	1.5504	...	0.2044	
0.2973	-0.9093	1.5118	0.8497	0.6398	0.5546	1.3480	...	0.5409	

통계분석 Part. 01 - 가설검정

1. 가설 설정

불량률의 차이가 큰 두 그룹간의 설비데이터에 대해 T-test를 진행한다.

만약, 통계적으로 유의하다면 불량률에 설비데이터가 영향을 끼칠 가능성이 높다고 가정한다.

T-test

귀무가설(H0) : 두 그룹의 설비 데이터는 서로 같다.

대립가설(H1) : 두 그룹의 설비 데이터는 서로 다르다.

2. 두 그룹 생성

두 배정번호 색출

AN	BQ_Rate	DZ1_OP	DZ2_OP	DZ1_TEMP	DZ2_TEMP	CLEAN
104126	0.000000	73.328862	22.591392	100.047576	100.060726	69.898156
128795	0.368509	68.056119	20.104963	99.973817	100.007704	66.168607

설비데이터에서 두 배정번호를 기준으로 두 그룹 생성

AN	DZ1_OP	DZ2_OP	DZ1_TEMP	DZ2_TEMP	CLEAN	HDZ1_OP	HDZ2_OP
104126	69.4803	26.0150	100.002	99.8174	69.4575	73.2032	46.8250
104126	71.8280	23.9977	100.685	99.8174	69.4575	73.6174	46.8453

AN	DZ1_OP	DZ2_OP	DZ1_TEMP	DZ2_TEMP	CLEAN	HDZ1_OP	HDZ2_OP
128795	63.3972	22.0008	100.002	100.0190	66.0195	89.3817	52.6439
128795	63.2470	22.1017	100.783	100.0190	65.9582	82.3504	52.6312

통계분석 Part. 02 – 가설검정 방법론

3. T-test

정규성 검정

표본의 수가 크므로 중심극한정리에 의해 정규성을 만족

$$n \geq 30 \quad \Rightarrow \quad \text{중심극한정리}$$

등분산성 검정 및 T-test

두 그룹간의 등분산성을 검정하고 등분산이냐 이분산이냐에 따라 T-test를 다르게 진행

등분산인 경우 : Students's T-test 진행

이분산인 경우 : Welch's T-test 진행

4. 가설 검증

두 그룹간의 설비별 T-test결과는 아래와 같다.

```
DZ1_OP설비
두 배정번호 간의 DZ1_OP설비는 등분산성을 만족하지 않는다.
두 배정번호 간의 DZ1_OP설비데이터는 유의하게 차이가 있다
pvalue:0.0

DZ2_OP설비
두 배정번호 간의 DZ2_OP설비는 등분산성을 만족한다.
두 배정번호 간의 DZ2_OP설비데이터는 유의하게 차이가 있다
pvalue:0.0
```

⋮

전체 설비 중 HDZ4_TEMP설비를 제외한 모든 설비가 대립가설을 만족하므로 불량률 차이가 큰 두 그룹간에는 설비 데이터의 차이가 존재한다고 해석 가능하다.

즉, 제품 불량률에 설비데이터가 영향을 미칠 가능성이 높다고 볼 수 있다.

통계분석 Part. 03 – 통계량

Cohen's d

Cohen's d는 **두 집단의 평균차이를 계산하는 효과크기(d)**
T-test는 표본이 충분히 크면 대부분 통계적으로 유의하다고 하므로 한계점을 보완하기 위해 Cohen's d도 함께 제시한다.

$$Cohen's\ d = \frac{① \overline{X_1} - \overline{X_2}}{② \sqrt{SD_P^2}}$$

$$SD_P^2 = \frac{(n_1 - 1)SD_1^2 + (n_2 - 1)SD_2^2}{(n_1 - 1) + (n_2 - 1)}$$

① 두 표본 집단의 평균 차이 ② 추정된 표준편차

효과크기(d)값이 클수록 두 집단이 겹치는 부분이 작아 **두 집단이 서로 다르다고 볼 수 있다.** 일반적으로 0.8이상이어야 큰 효과라고 간주한다.

BF10

BF10은 귀무가설과 대립가설을 비교하여 **대립가설이 데이터를 얼마나 잘 예측하는지 정량화한 수치**이다.

$$BF_{10} = \frac{①\ p(D|H_1)}{②\ p(D|H_0)}$$

① 데이터가 주어졌을 때 대립가설이 참일 확률
② 데이터가 주어졌을 때 귀무가설이 참일 확률

BF10의 값이 클수록 대립가설을 채택하기 쉬워진다.
1의 BF값은 데이터가 두 가설 하에서 동일하게 발생한다는 것을 의미한다.

통계분석 Part. 04 – 통계지표

설비번호	설비	T 통계량	자유도	alternative	P value	95% 신뢰구간	cohen-d	BF10	power
0	DZ1_OP	90.16345	12467.8	two-sided	0	[5.16 5.39]	1.412957	inf	1
1	DZ2_OP	53.69672	14127.17	two-sided	0	[2.4 2.58]	0.806229	inf	1
2	DZ1_TEMP	8.818917	9501.728	two-sided	1.36E-18	[0.06 0.09]	0.152867	1.17E+15	1
3	DZ2_TEMP	8.749648	9924.689	two-sided	2.49E-18	[0.04 0.06]	0.14905	6.39E+14	1
4	CLEAN	636.3704	13148.22	two-sided	0	[3.72 3.74]	9.791977	inf	1
5	HDZ1_OP	-35.8723	11171.51	two-sided	7.88E-267	[-14.36 -12.87]	0.584422	4.193E+268	1
6	HDZ2_OP	-179.778	13989.6	two-sided	0	[-9.4 -9.2]	2.708309	inf	1
7	HDZ3_OP	-155.223	18558.58	two-sided	0	[-5.74 -5.6]	2.092628	inf	1
8	HDZ4_OP	-17.1427	16762.38	two-sided	2.57E-65	[-0.55 -0.44]	0.242122	3.26E+61	1
9	HDZ_CP	-84.7984	12625.6	two-sided	0	[-0.11 -0.11]	0.942821	inf	1
10	HDZ_CPM	7.176744	10520.13	two-sided	7.62E-13	[0. 0.]	0.119547	2.44E+09	1
11	HDZ1_TEMP	11.02264	8076.371	two-sided	4.71E-28	[0.59 0.85]	0.205189	3.22E+24	1
12	HDZ2_TEMP	5.332404	8694.709	two-sided	9.94E-08	[0.03 0.08]	0.095968	2.47E+04	0.999996
13	HDZ3_TEMP	7.37063	10512.44	two-sided	1.83E-13	[0.03 0.06]	0.122811	9.96E+09	1
14	HDZ4_TEMP	1.277287	14960.89	two-sided	0.201521	[-0. 0.02]	0.018803	0.038	0.241036
15	SCZ1_TEMP	-8.1857	13682.71	two-sided	2.95E-16	[-1.47 -0.9]	0.124253	5.51E+12	1
16	SCZ2_TEMP	-13.7261	12440.99	two-sided	1.45E-42	[-1.62 -1.21]	0.215264	8.13E+38	1
17	STZ1_TEMP	-1079.23	16231.22	two-sided	0	[-3.06 -3.05]	12.62676	inf	1
18	STZ2_TEMP	-1026.7	15577.26	two-sided	0	[-3.67 -3.65]	11.89546	inf	1

예지보전 Part. 01 : 모델 비교

분석 도구

TensorFlow Keras

LSTM

3개의 Features 선정
(DZ1_OP, DZ2_OP, DZ1_TEMP)

```
# Print Classification Report
print(classification_report(y_test, y_pred_classes))
```

✓ 25.5s

	precision	recall	f1-score	support
0	0.91	1.00	0.95	116
1	1.00	0.75	0.86	48
accuracy			0.93	164
macro avg	0.95	0.88	0.90	164
weighted avg	0.93	0.93	0.92	164

모델 채택 사유

아래는 DZ2_OP 예시

1. 25.5s vs. 18.5s

2. Similar Performance

GRU

3개의 Features 선정
(DZ1_OP, DZ2_OP, DZ1_TEMP)

```
# Print Classification Report
print(classification_report(y_test, y_pred_classes))
```

✓ 18.5s

	precision	recall	f1-score	support
0	0.92	0.98	0.95	116
1	0.95	0.79	0.86	48
accuracy			0.93	164
macro avg	0.93	0.89	0.91	164
weighted avg	0.93	0.93	0.92	164

예지보전 Part. 02 : 모델 성능 테스트

GRU

```
# Define the number of time_steps
time_steps = 30
```

time_steps = 30 설정

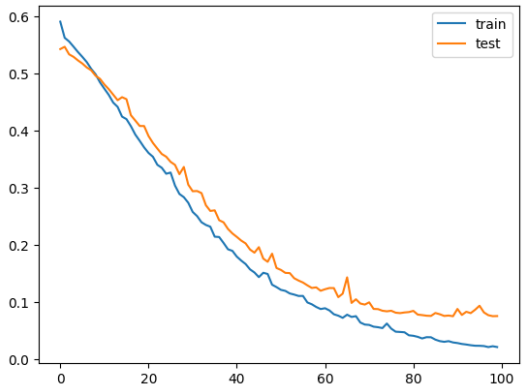
Year	Month	Day	Hour
2022	1	3	11
2022	1	3	12
2022	1	3	13
2022	1	3	22
2022	1	3	23
2022	1	4	0
2022	1	4	1
2022	1	4	2
2022	1	4	22
2022	1	4	23

대략적으로 3~5일 간격

모델 채택 사유

```
# Set early stopping
early_stopping = EarlyStopping(monitor='val_loss', patience=10)
```

patience = 10 설정



Overfitting 방지

특정 설비 이상신호 감지

전체 공정 중에서 구체적으로 특정 설비에 대한 예지보전을 실시

예지보전 Part. 03 : 모델 성능 시각화

GRU

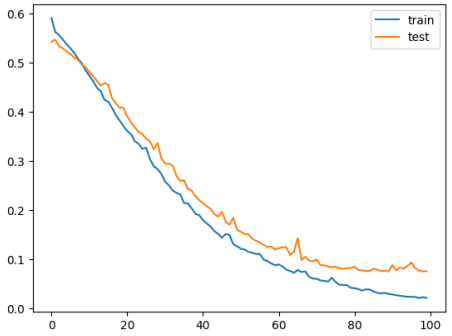
총 18개의 설비 데이터

각각의 설비 데이터를 전부 GRU 알고리즘과 접목하여 예지보전을 시행

3개의 Features 선정 (DZ1_OP, DZ2_OP, DZ1_TEMP)

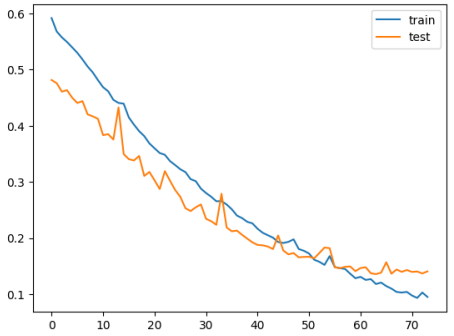
각 Target별 성능 및 시각화

```
F1 Score: 0.9512195121951219
Accuracy: 0.975609756097561
Recall: 0.9512195121951219
Precision: 0.9512195121951219
```



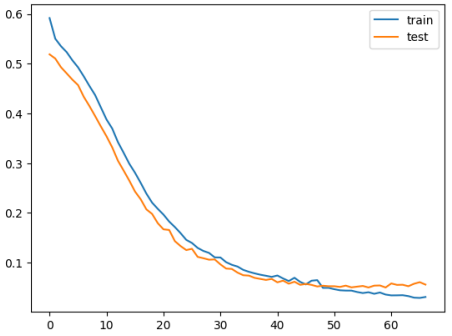
DZ1_OP

```
F1 Score: 0.8636363636363635
Accuracy: 0.926829268292683
Recall: 0.7916666666666666
Precision: 0.95
```



DZ2_OP

```
F1 Score: 0.9375
Accuracy: 0.9634146341463414
Recall: 0.9375
Precision: 0.9375
```



DZ1_TEMP

품질보증 Part. 01 : 다양한 모델, 오버샘플링 기법 적용

Over Sampling

Imbalanced learn

↓

Data Imbalance 방지

↓

4가지 Oversampling Methods 도입

모델 채택 사유

e.g.) LightGBM + Random Oversampling

- 1. Random Oversampling
- 2. SMOTE
- 3. SVM SMOTE
- 4. Borderline SMOTE

특정 설비 이상신호 감지

LightGBM + 1. Random Oversampling

```
Accuracy: 64.29%
F1 Score: 0.64
Recall: 0.64
Precision: 0.64
Confusion Matrix:
 [[15  5]
 [ 5  3]]
ROC AUC: 0.56
Classification Report:
              precision    recall  f1-score   support

           0       0.75      0.75      0.75        20
           1       0.38      0.38      0.38         8

    accuracy                           0.64        28
   macro avg       0.56      0.56      0.56        28
weighted avg       0.64      0.64      0.64        28
```

총 4개의 Oversampling 기법과 7개의 알고리즘을 비교하여 모델을 선정

품질보증 Part. 02 : 각 모델, 오버샘플링별 성능 지표

예측 모델별 성능 지표

No	Models	Oversampling	AUC	F1-Score
1	XGBoost	RandomOverSampler	0.68	0.59
2	XGBoost	SMOTE	0.82	0.77
3	XGBoost	SVMSMOTE	0.61	0.50
4	XGBoost	BorderlineSMOTE	0.61	0.54
1	RandomForest	RadomOverSampler	0.71	0.51
2	RandomForest	SMOTE	0.75	0.60
3	RandomForest	SVMSMOTE	0.71	0.51
4	RandomForest	BorderlineSMOTE	0.68	0.49

예측 모델별 성능 지표

No	Models	Oversampling	AUC	F1-Score
1	Extra Trees	RandomOverSampler	0.56	0.56
2	Extra Trees	SMOTE	0.42	0.43
3	Extra Trees	SVMSMOTE	0.51	0.49
4	Extra Trees	BorderlineSMOTE	0.53	0.50
1	Gradient Boosting	RadomOverSampler	0.54	0.51
2	Gradient Boosting	SMOTE	0.50	0.50
3	Gradient Boosting	SVMSMOTE	0.46	0.45
4	Gradient Boosting	BorderlineSMOTE	0.50	0.50

선정되지 않은 알고리즘 모델

4 Oversampling Methods + 6 Algorithm Models

예측 모델별 성능 지표

No	Models	Oversampling	AUC	F1-Score
1	LogisticRegression	RandomOverSampler	0.68	0.64
2	LogisticRegression	SMOTE	0.57	0.55
3	LogisticRegression	SVMSMOTE	0.71	0.65
4	LogisticRegression	BorderlineSMOTE	0.68	0.66
1	LightGBM	RadomOverSampler	0.56	0.56
2	LightGBM	SMOTE	0.61	0.62
3	LightGBM	SVMSMOTE	0.47	0.47
4	LightGBM	BorderlineSMOTE	0.47	0.47

품질보증 Part. 03 : 모델 과적합 방지

Ensemble + SMOTE

No	Models	Oversampling	AUC	F1-Score
2	Ensemble(et+qda+gbc)	SMOTE	0.94	0.93

↓

상대적으로 높은 성능 발휘

↓

Overfitting 방지 필요

→ 새로운 설비 데이터에 적용 시 신뢰 여부 →

5가지 적용

1. train_test_split
2. K-Fold Cross Validation
3. Oversampling - SMOTE
4. Regularization
5. Model Ensemble

↓

1. test_size = 0.2 설정
2. 모델을 신뢰성 있게 평가
3. 모델의 복잡성 제어
4. 단일 모델 대비 방지 효과
5. Imbalance Class 방지

품질보증 Part. 04 : 최종 모델 선정

4 Oversampling Methods + Ensemble Algorithm Models

최종선정

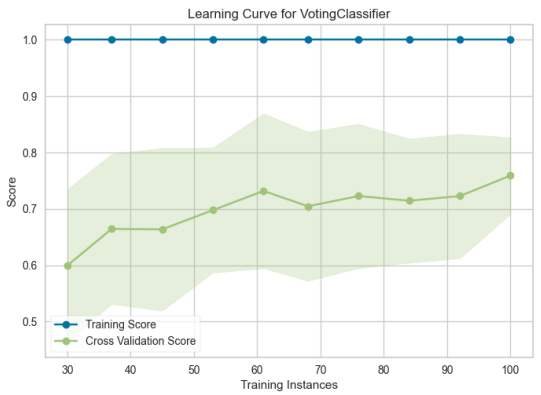
예측 모델별 성능 지표

No	Models	Oversampling	AUC	F1-Score
1	Ensemble(gbc+cat+xgb)	RandomOverSampler	0.97	0.94
2	Ensemble(et+qda+gbc)	SMOTE	0.94	0.93
3	Ensemble(gbc+et+cat)	SVMSMOTE	0.96	0.88
4	Ensemble(qda+et+gbc)	BorderlineSMOTE	0.94	0.93

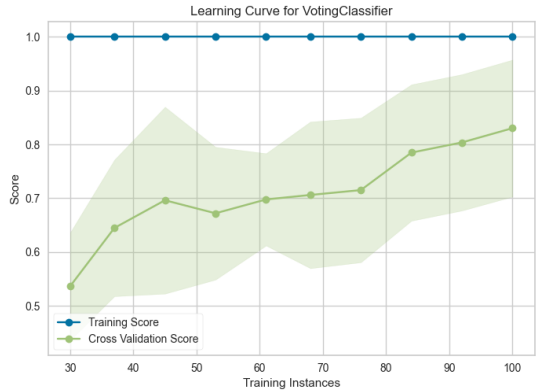
선정 근거 1: 안정된 AUC & F1-Score

No	Models	Oversampling	AUC	F1-Score
2	Ensemble(et+qda+gbc)	SMOTE	0.94	0.93

선정 근거 2: CV Score

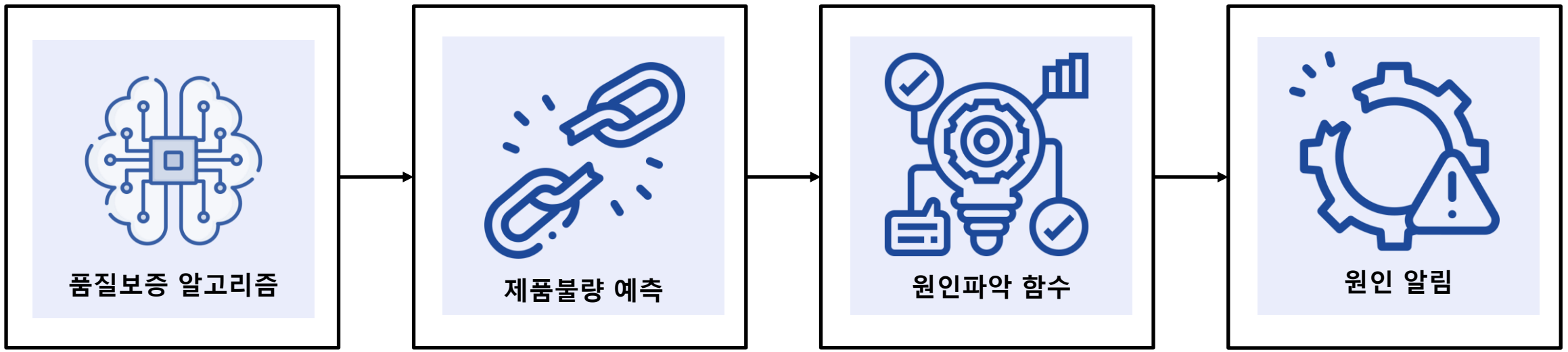


Random Oversampling



SMOTE

품질보증 Part. 05 : 알고리즘 활용한 불량원인 도출



품질보증 Part. 06 : 불량원인의 도출

```python
def Bad_Cause(df):
    outlier_count = pd.DataFrame()
    outlier_label_dict = {}

    for column in df.columns:
        q1 = df[column].quantile(0.25)
        q3 = df[column].quantile(0.75)
        iqr = 1.5 * (q3 - q1)

        outlier_mask = (df[column] < (q1 - iqr)) | (df[column] > (q3 + iqr))
        outlier_count[column] = outlier_mask.astype(int)

    for column in outlier_count.columns:
        outlier_percentage = outlier_count[column].sum() / len(outlier_count)
        if outlier_percentage >= 0.4:
            outlier_label_dict[column] = 1
        else:
            outlier_label_dict[column] = 0

    outlier_label = pd.DataFrame(outlier_label_dict, index=[0])

    for column in outlier_label.columns:
        if outlier_label[column][0] != 0:
            print(f'{column}설비에 이상이 감지되었습니다.')

    return outlier_label
```

특정 기간에 어떤 설비에 이상이 발생했는지 도출해주는 함수를 정의

실제 활용시 : 품질보증 알고리즘으로 불량이 발생한 구간 도출

이 구간의 데이터를 함수에 집어넣어 아래의 예시처럼 어떤 설비가 원인인지 파악이 가능

```
Bad_Cause(df1.iloc[0:60])
```

	DZ1_OP	DZ2_OP	DZ1_TEMP	DZ2_TEMP	CLEAN	HDZ1_OP	HDZ2_OP	HDZ3_OP	HDZ4_OP	HDZ_CP	HDZ_CPM	HDZ1_TEMP	HDZ2_TEMP	HDZ3_TEMP	HDZ4_TEMP	SCZ1_TEMP	SCZ2_TEMP	STZ1_TEMP	STZ2_TEMP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

```
Bad_Cause(df1.iloc[60:120])
```

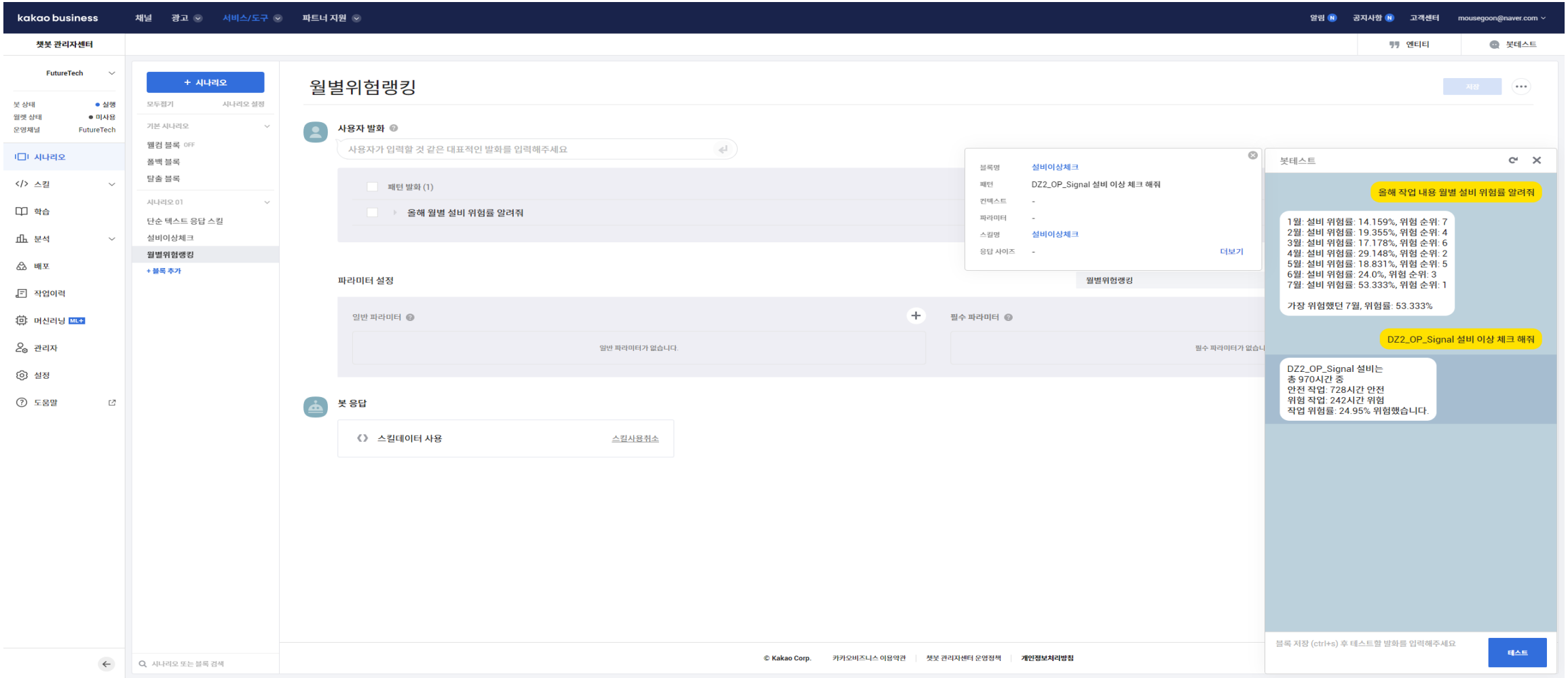
```
HDZ_CPM설비에 이상이 감지되었습니다.
```

	DZ1_OP	DZ2_OP	DZ1_TEMP	DZ2_TEMP	CLEAN	HDZ1_OP	HDZ2_OP	HDZ3_OP	HDZ4_OP	HDZ_CP	HDZ_CPM	HDZ1_TEMP	HDZ2_TEMP	HDZ3_TEMP	HDZ4_TEMP	SCZ1_TEMP	SCZ2_TEMP	STZ1_TEMP	STZ2_TEMP
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

AI 기반 스마트 팩토리 프로세스 구축

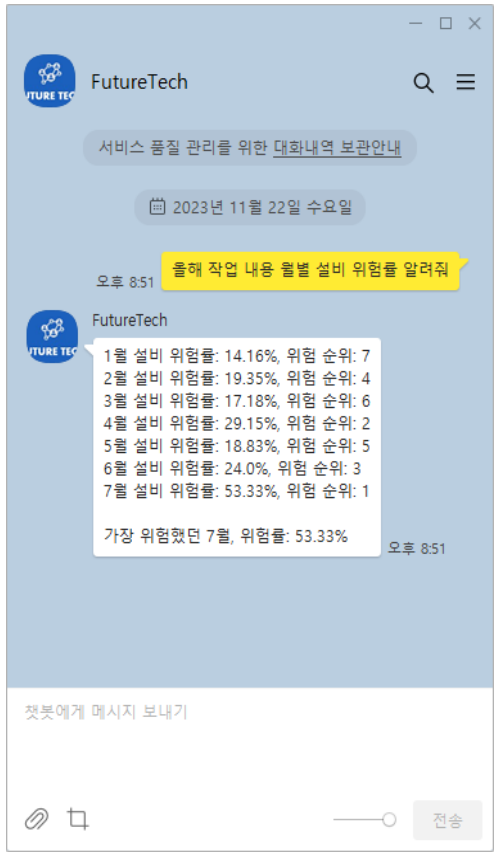
INDEX

알림 서비스 구현 – 카카오톡 챗봇 테스트 화면

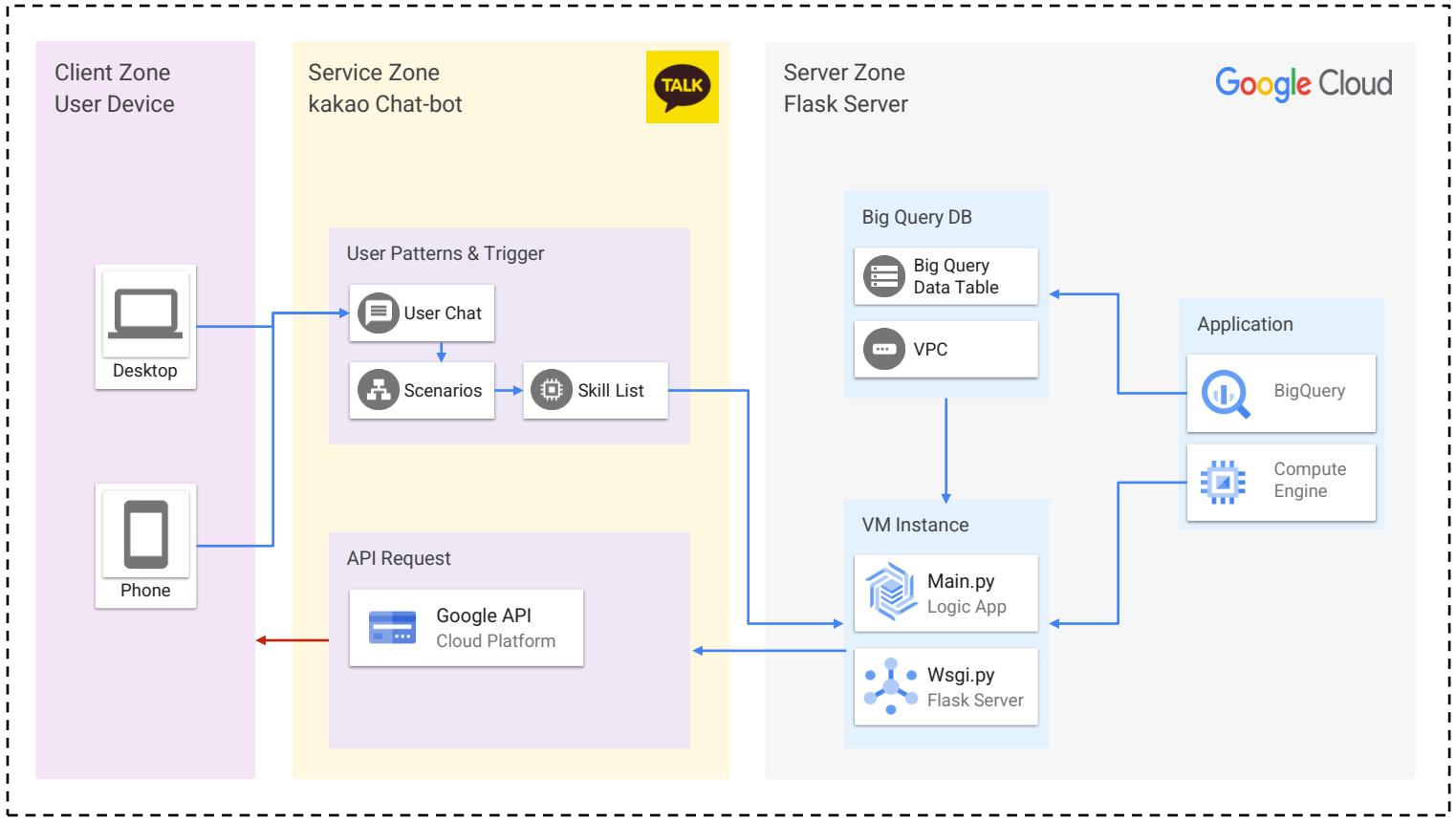


알림 서비스 구현 아키텍처

카카오톡 챗봇 로직 사용 예시

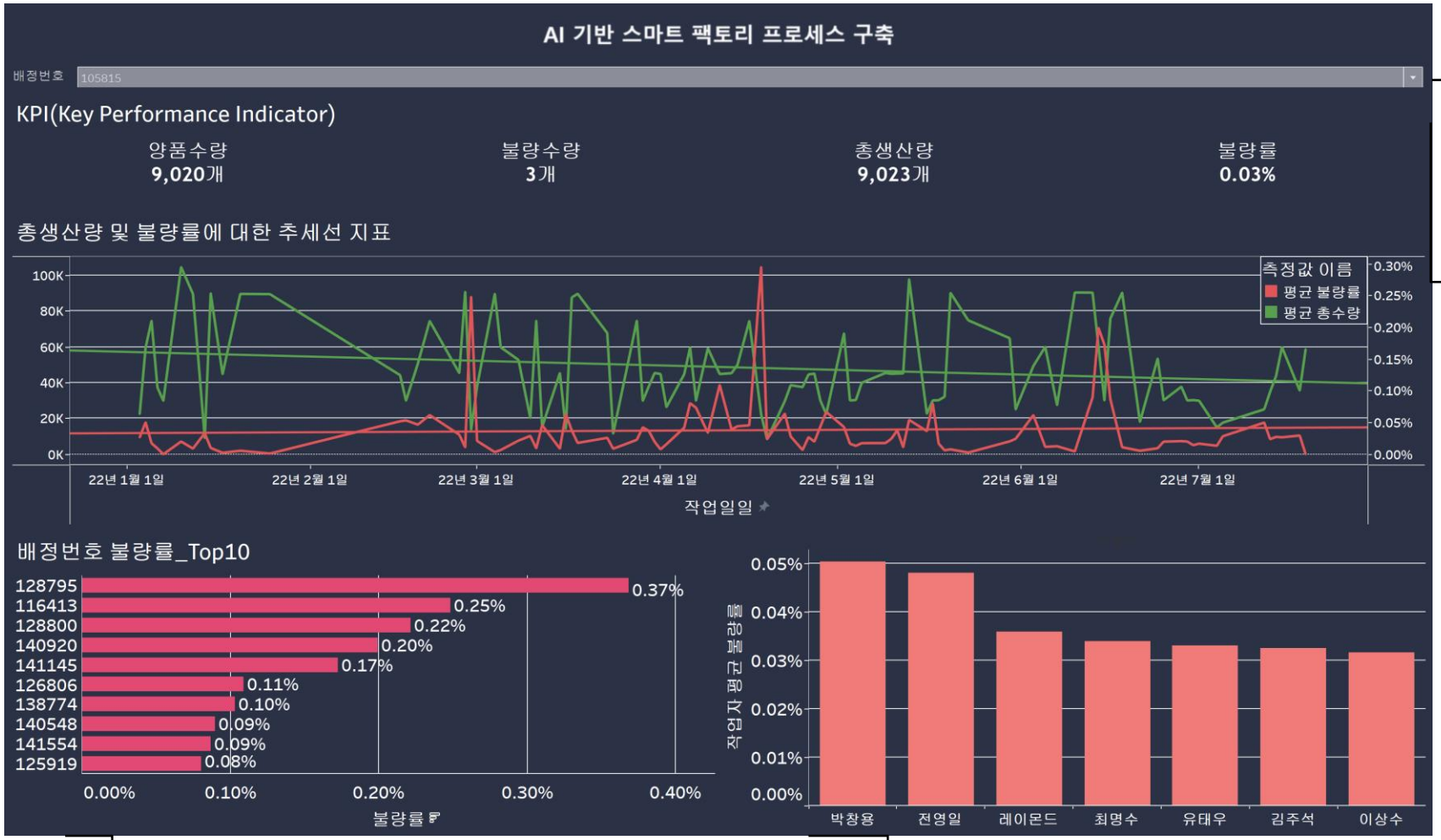


카카오톡 챗봇 아키텍처 설계



03. 프로젝트 수행 결과 알림 서비스 / 클라우드 서비스

multicampus



각 배정번호를 Dropdown에서 선택 후 **KPI** 확인

해당 Dataset의 시각화 및 불량률 분석

① **작업기간과 불량률 간의 상관성 ↓**

② **총 수량과 불량률 간의 상관성 ↓**

③ **평균 불량률에서 Spike가 발생한 일자에 점검 필요성**

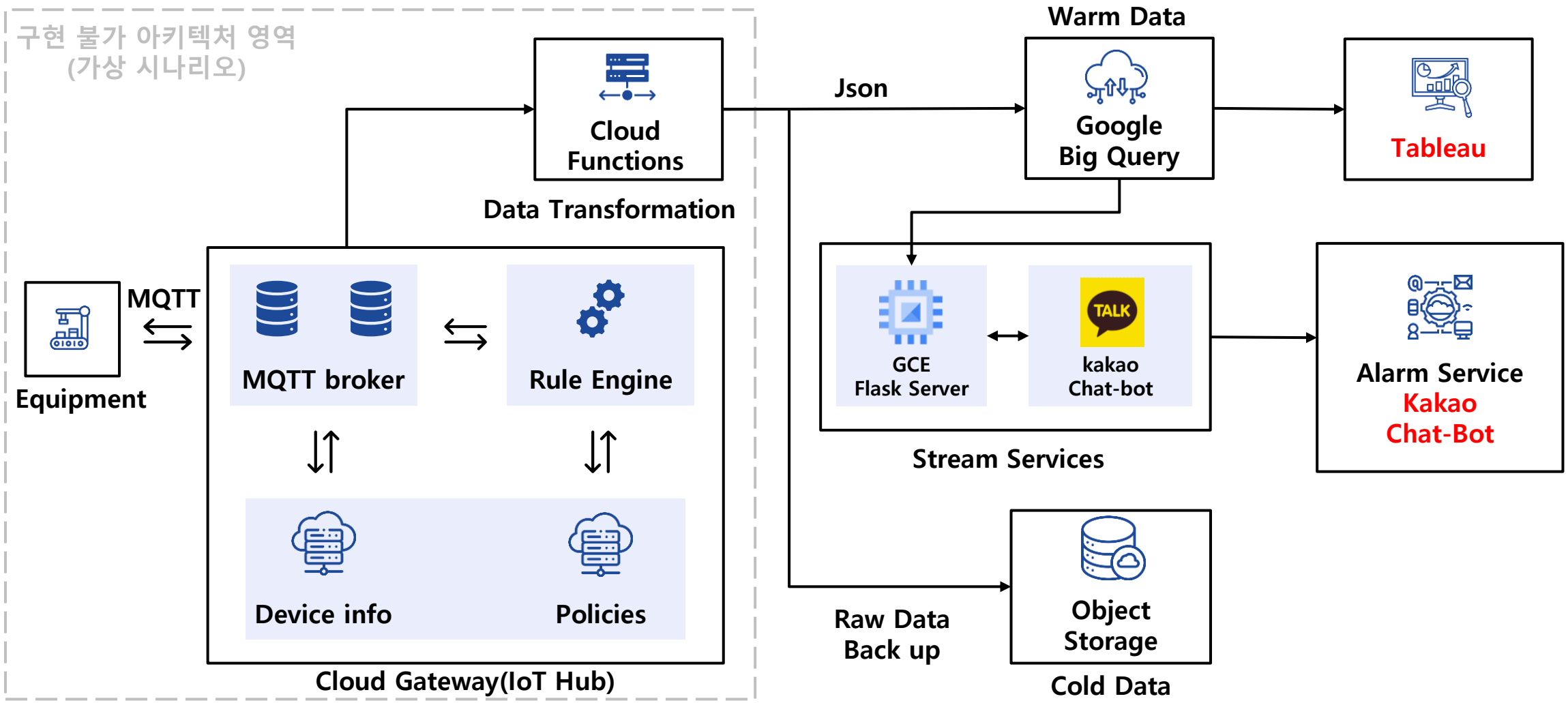
전체 배정번호들 중 **Top 10**의 불량률 순위를 나타낸 그래프

"박창용"과 "이상수"의 제품 불량률 비교 → 약 **60%** 차이 (숙련도 & 재교육)

tableau

대시보드 보러가기

클라우드 서비스 아키텍처 설계



AI 기반 스마트 팩토리 프로세스 구축

INDEX

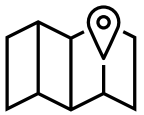
multicampus

분석 기대효과 및 한계점

기대 효과

작업자의 경험 의존적인 업무환경 및 제조공정 중 제품 퀄리티를 육안으로 확인하기 어려움
-> AI 기반 알고리즘 & Cloud Platform을 통해 데이터 기반의 산업 현장 문제 해결 기대

사전에 설비 문제를 파악하여 현장의 작업자에게 정보를 제공하여 예지보전을 달성



AI 모델링 작업을 거쳐서 대략적으로 전체 공정에 대한 문제를 파악하는 것을 넘어서 구체적으로 특정한 설비의 문제를 파악

한계점

실제로 설비의 문제가 아닌, 예기치 못한 요인이나 원재료 자체로부터 문제가 발생했을 경우의 한계

현재로선 작은 데이터 사이즈로 인한 모델의 성능 저하 및 Overfitting 가능성

INDEX

테이블 종류	database.data
데이터 정의	열처리 뿌리금형 설비 데이터

No	Column Name	Column Name(KOR)	Dtype	Etc
1	TAG_MIN	TAG_MIN	TIMESTAMP	IoT 수집 데이터(초 단위)
2	AN	배정번호	INTEGER	공정의 작업 배정번호
3	DZ1_OP	건조 1존 OP	FLOAT	건조 온도 유지를 위한 출력량(%)
4	DZ2_OP	건조 2존 OP	FLOAT	건조 온도 유지를 위한 출력량(%)
5	DZ1_TEMP	건조로 온도 1 Zone	FLOAT	각 건조로 Zone 온도
6	DZ2_TEMP	건조로 온도 2 Zone	FLOAT	각 건조로 Zone 온도
7	CLEAN	세정기	FLOAT	세정기 온도
8	HDZ1_OP	소입1존 OP	FLOAT	소입존 온도 유지를 위한 출력량(%)
9	HDZ2_OP	소입2존 OP	FLOAT	소입존 온도 유지를 위한 출력량(%)
10	HDZ3_OP	소입3존 OP	FLOAT	소입존 온도 유지를 위한 출력량(%)
11	HDZ4_OP	소입4존 OP	FLOAT	소입존 온도 유지를 위한 출력량(%)
12	HDZ_CP	소입로 CP 값	FLOAT	침탄 가스의 침탄 능력의 양(%)
13	HDZ_CPM	소입로 CP 모니터 값	FLOAT	침탄 가스의 침탄 능력 모니터링 값
14	HDZ1_TEMP	소입로 온도 1 Zone	FLOAT	솔트 온도 유지를 위한 출력량(%)
15	HDZ2_TEMP	소입로 온도 2 Zone	FLOAT	솔트 온도 유지를 위한 출력량(%)

테이블 종류	database.data
데이터 정의	열처리 뿌리금형 설비 데이터

No	Column Name	Column Name(KOR)	Dtype	Etc
16	HDZ3_TEMP	소입로 온도 3 Zone	FLOAT	솔트 온도 유지를 위한 출력량(%)
17	HDZ4_TEMP	소입로 온도 4 Zone	FLOAT	솔트 온도 유지를 위한 출력량(%)
18	SCZ1_TEMP	솔트 컨베이어 온도 1 Zone	FLOAT	솔트 컨베이어 Zone의 온도
19	SCZ2_TEMP	솔트 컨베이어 온도 2 Zone	FLOAT	솔트 컨베이어 Zone의 온도
20	STZ1_TEMP	솔트조 온도 1 Zone	FLOAT	솔트조 Zone의 온도
21	STZ2_TEMP	솔트조 온도 2 Zone	FLOAT	솔트조 Zone의 온도

테이블 종류	database.quality
데이터 정의	열처리 뿌리금형 품질 데이터

No	Column Name	Column Name(KOR)	Dtype	Etc
1	AN	배정번호	INTEGER	
2	GQ	양품수량	INTEGER	
3	BQ	불량수량	INTEGER	
4	TQ	총수량	INTEGER	
5	BQ Rate	불량률	FLOAT	(불량수량 / 총수량) * 100
6	DS	불량단계	STRING	불량률 >= 3 IQR 위험, 이외 안전
7	작업일	작업일	Date	작업일자
8	공정명	공정명	STRING	
9	설비명	설비명	STRING	

참고문헌

- Zhuohan Li, Di He, Fei Tian, Wei Chen, Tao Qin, Liwei Wang, Tieyan Liu. (2018). Towards Binary-Valued Gates for Robust LSTM Training. Proceedings of Machine Learning Research.
- Pawan Whig, Ketan Gupta, Nasmin Jiwani, Hruthika Jupalle, Shama Kouser & Naved Alam. (2023). A novel method for diabetes classification and prediction with Pycaret. Springer.
- Rahul Dey; Fathi M. Salem. (2017). Gate-variants of Gated Recurrent Unit (GRU) neural networks. IEEE.
- JL Dossett, HE Boyer. (2006). Practical heat treating. Journal of Real Estate Analysis, 9(2), ASM International.

출처

- 무료 아이콘 제공 사이트 1 : https://www.flaticon.com/kr/
- 무료 아이콘 제공 사이트 2 : https://icons8.kr/icons
- 무료 아이콘 제공 사이트 3 : https://icon-icons.com/ko/
- 무료 아이콘 제공 사이트 4 : https://kr.freepik.com/icons
- NCP 서버 구축 가이드북 : https://guide.ncloud-docs.com/docs/ko/server-create-vpc
- NCP IoT Analysis Platform 아키텍처 도식화 : https://www.ncloud.com/intro/architecture/23
- Kakao Talk icon : https://mbolt.tistory.com/260
- Google Cloud Platform Icon : https://cloud.google.com/icons?hl=ko