colab

HAI DATASET

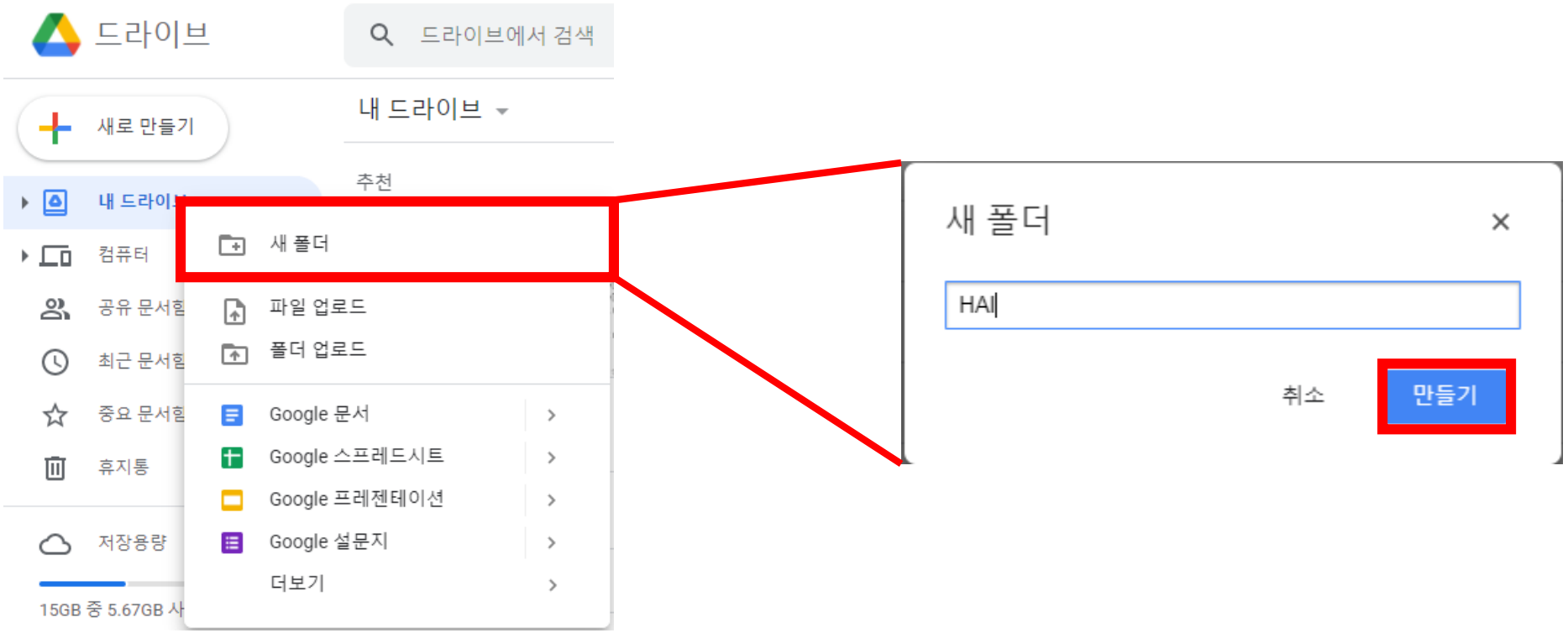
HIL-BASED AUGMENTED ICS (HAI) DATASET WAS COLLECTED FROM A REALISTIC ICS TESTBED AUGMENTED WITH A HARDWARE-IN-THE-LOOP SIMULATOR THAT EMULATES STEAM-TURBINE POWER GENERATION AND PUMPED-STORAGE HYDROPOWER GENERATION

Colab 사용 방법 설명

- Colab
  - ➢클라우드 기반의 Jupyter Notebook 개발 환경
  - ➢인터넷 브라우저로 언제 어디서든 접속 및 수정 가능
  - ➢구글 클라우드 서버에서 CPU, GPU 자원을 무료로 사용
    - 하드웨어의 투자 및 번거로운 설정과정을 하나하나 하지 않아도 되기 때문에 사람들이 딥러닝을 학습하는데 많이 사용
  - ➢무료 사용시 세션 유지는 최대 12시간 가능

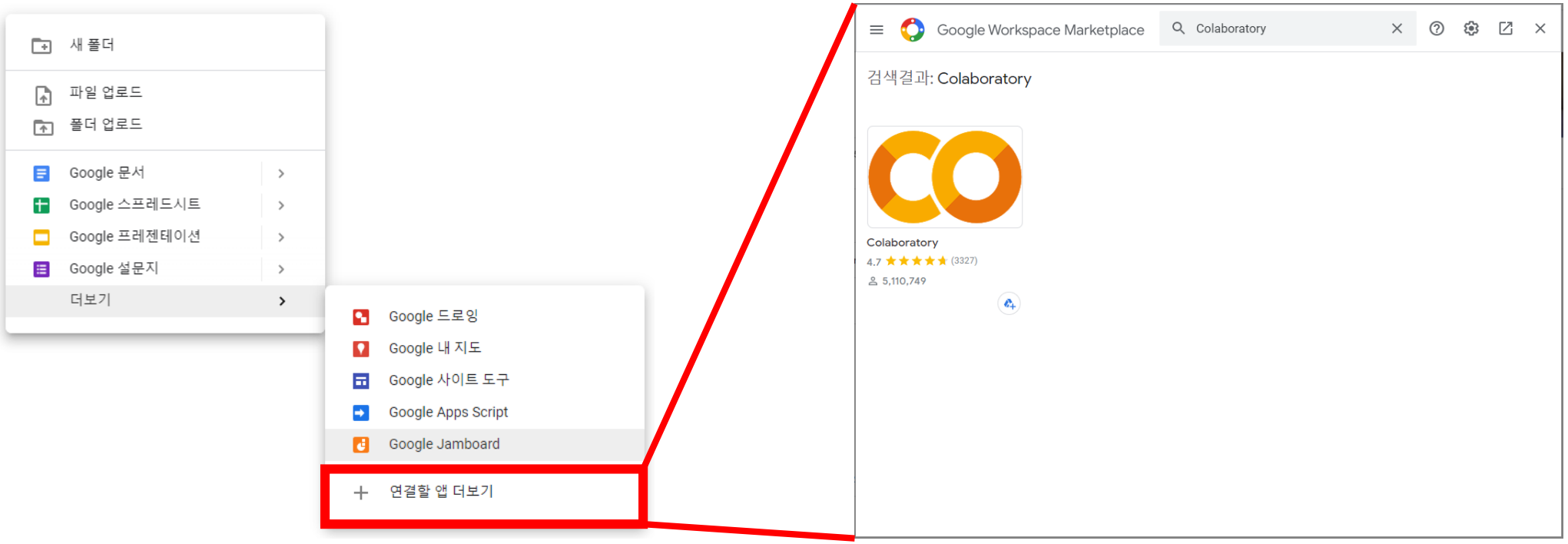
Colab 사용 방법 설명

- Colab 사용 방법
  - ➢구글 계정으로 로그인 후 구글 드라이브로 이동
  - ➢내드라이브 우클릭 -> '새폴더' 선택
  - ➢'HAI' 폴더 생성



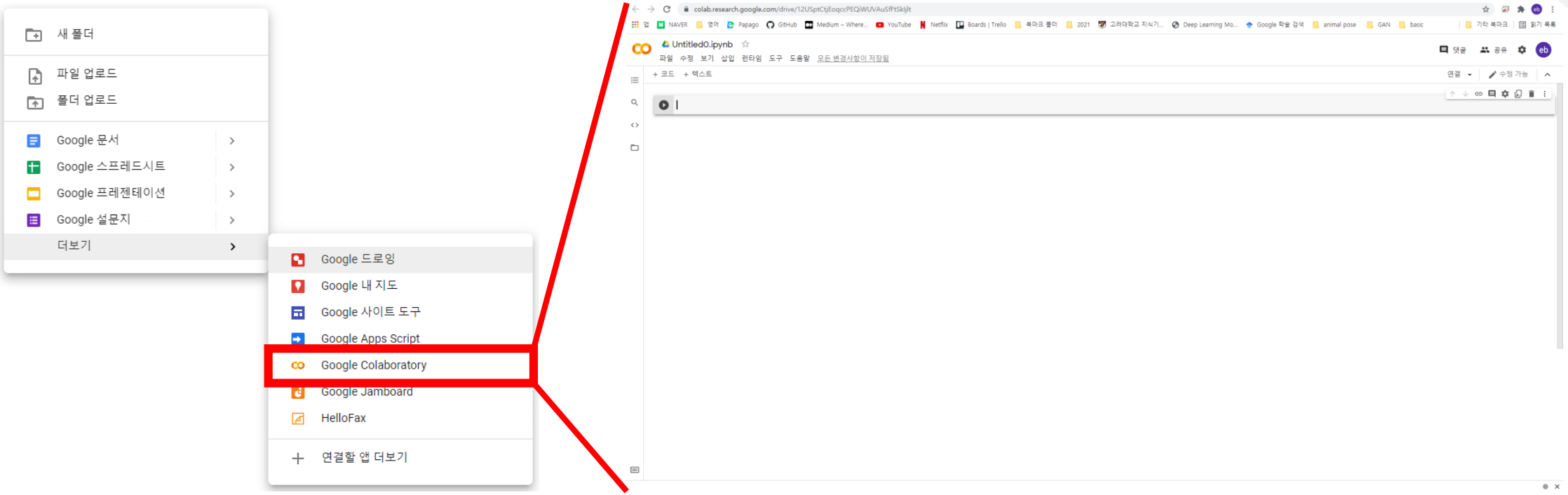
Colab 사용 방법 설명

- Colab 사용 방법
  - ➢'HAI'폴더로 이동
  - ➢우클릭 -> '더보기' -> 'Google Colaboratory' 실행
  - ➢('Google Colaboratory'가 없는 경우) 우클릭 -> '더보기' -> '연결할 앱 더 보기' 선택 -> apps 검색 창에 Colaboratory 검색 -> 구글 드라이브 연동



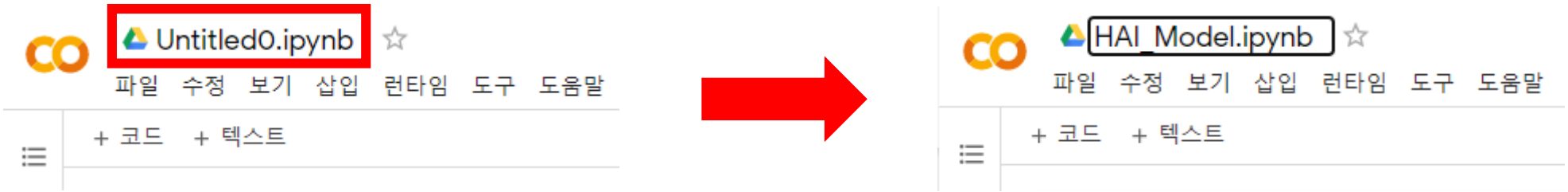
Colab 사용 방법 설명

- Colab 사용 방법
  - ➢'HAI'폴더로 이동
  - ➢우클릭 -> '더보기' -> 'Google Colaboratory' 실행
  - ➢('Google Colaboratory'가 없는 경우) 우클릭 -> '더보기' -> '연결할 앱 더 보기' 선택 -> apps 검색 창에 Colaboratory 검색 -> 구글 드라이브 연동



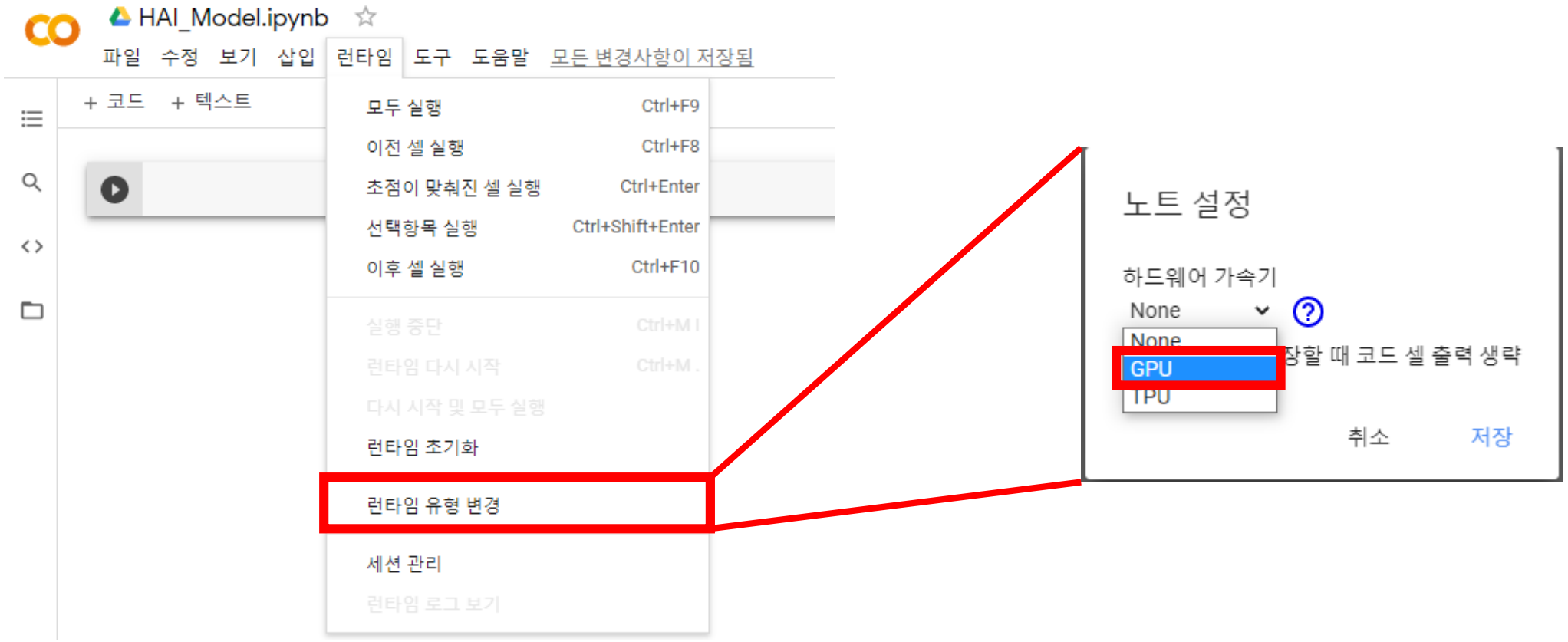
Colab 사용 방법 설명

- Colab 사용 방법
  - ➢‘Untitled0.ipynb’를 클릭 후 원하는 ‘HAI_Model.ipynb’로 변경



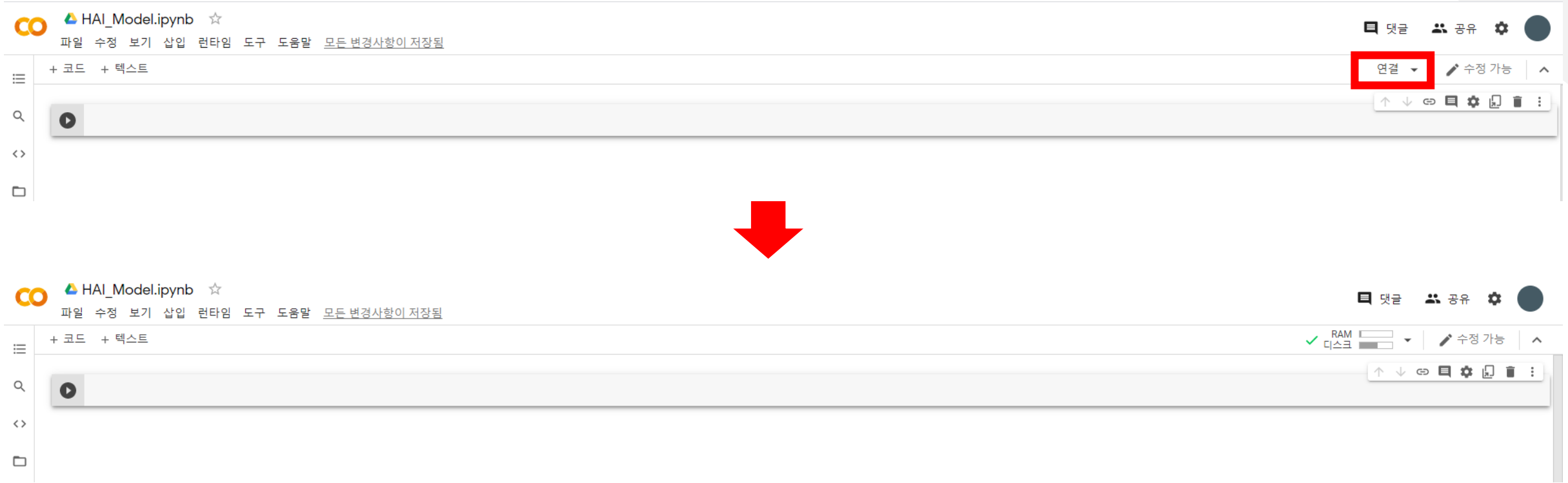
Colab 사용 방법 설명

- Colab 사용 방법
  - ➢런타임 유형 변경
    - None, GPU, TPU 3가지 유형 중에 GPU 유형 선택



Colab 사용 방법 설명

- Colab 사용 방법
  - ➢런타임 유형을 설정하면 자동으로 원격 서버에 연결됨
  - ➢자동 연결이 되지 않았을 경우 연결 버튼을 수동으로 눌려 서버에 연결

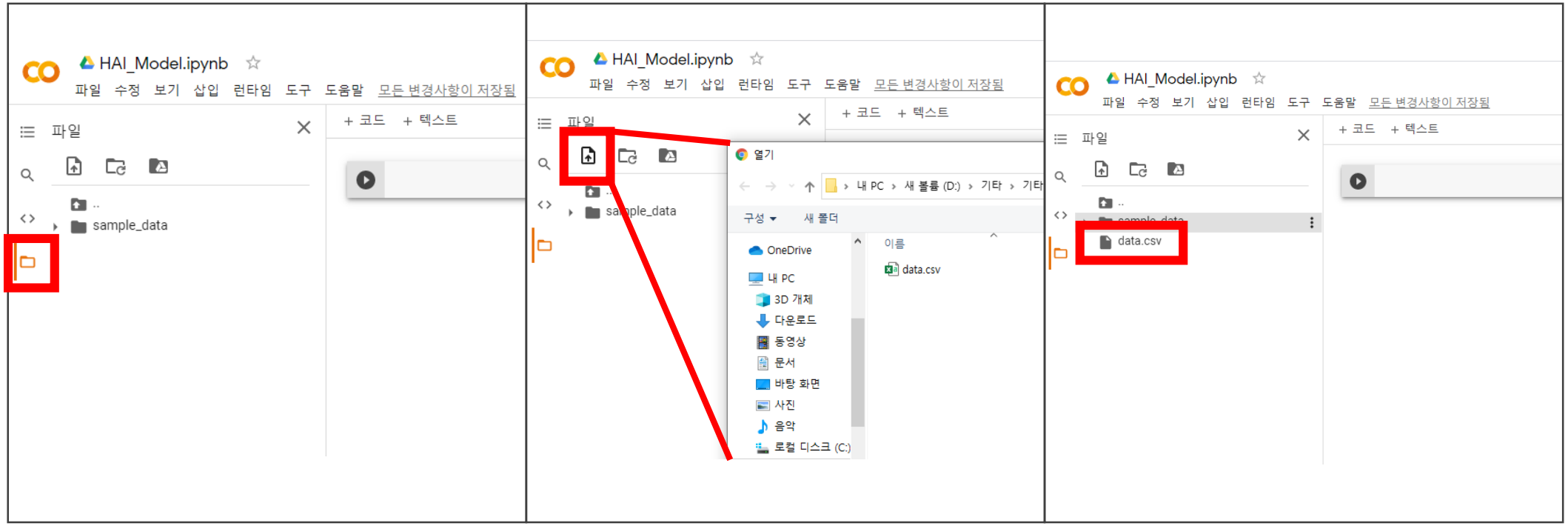


Colab 사용 방법 설명

- Colab 사용 방법
  - ➢Colab에서 외부 데이터를 사용하는 방법은 크게 2가지가 있음
    - 1. Local 폴더에서 데이터를 불러온 후 사용하는 방법
    - 2. 구글 드라이브와 연결하여 데이터를 사용하는 방법

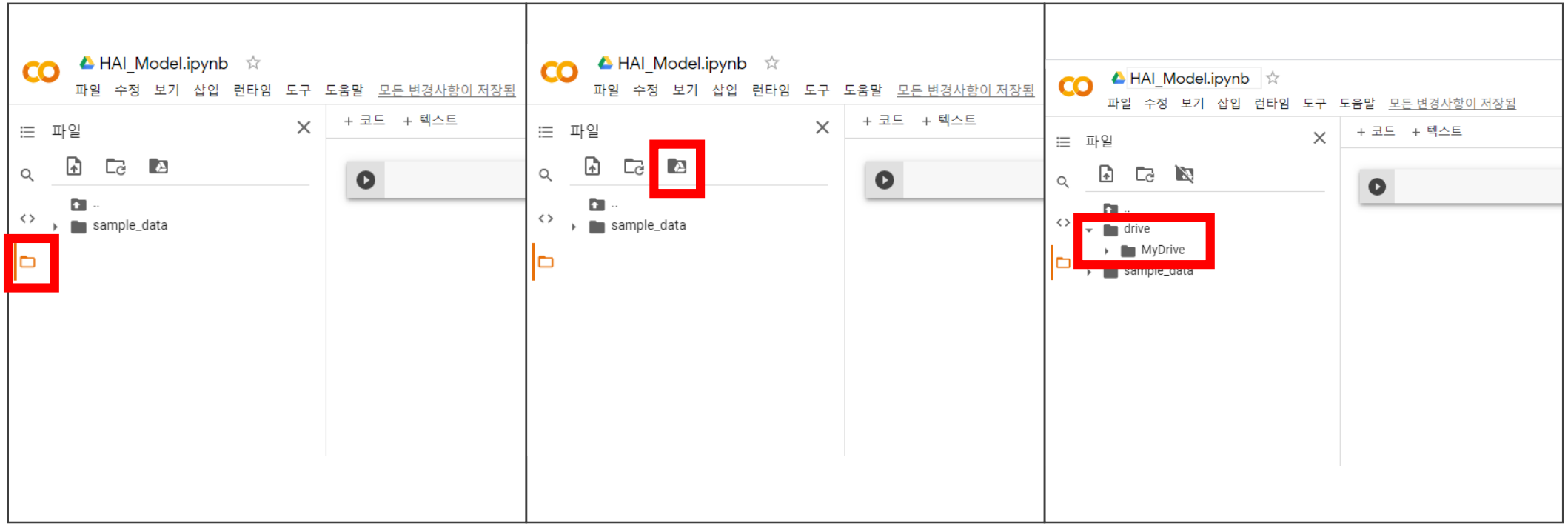
Colab 사용 방법 설명

- Colab 사용 방법
  - ➢Local 폴더에서 데이터를 불러온 후 사용하는 방법
    - 왼쪽 탭에서 폴더 아이콘 클릭
    - 로컬 드라이브 아이콘 선택 후 파일 선택
    - 선택된 파일은 현재 사용중인 세션에서 접근이 가능해짐



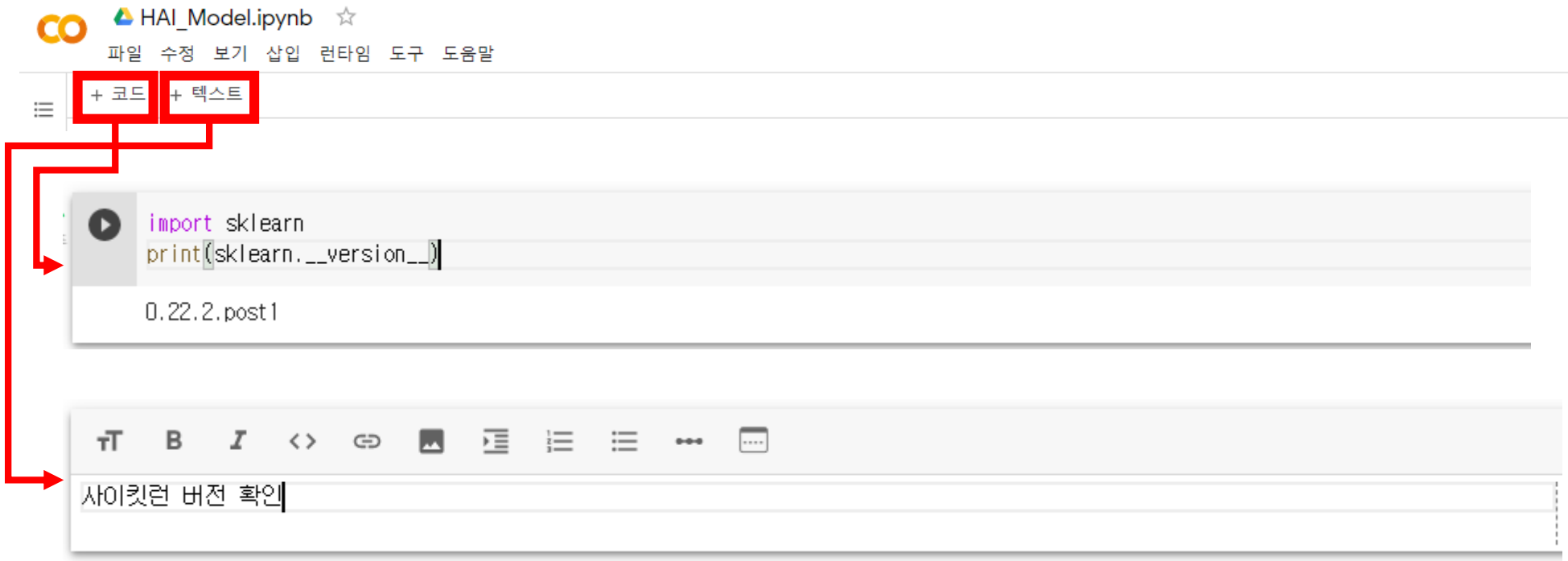
Colab 사용 방법 설명

- Colab 사용 방법
  - ➢구글 드라이브와 연결하여 데이터를 사용하는 방법
    - 왼쪽 탭에서 폴더 아이콘 클릭
    - 구글 드라이브 아이콘 선택
    - 본인 계정의 구글 드라이브에 들어있는 데이터 사용 가능



Colab 사용 방법 설명

- Colab 사용 방법
  - ➢'+ 코드'탭을 클릭하면 리눅스 명령어, Python 등 코드를 작성할 수 있는 셀이 생성됨
  - ➢실행 버튼 클릭 시 코드 셀의 하단에 실행 결과가 출력됨
  - ➢'+ 텍스트'탭을 클릭하면 Markdown 언어 작성 셀이 생성됨



Colab 사용 방법 설명

- Colab 사용 방법
  - ➢Colab을 통하여 사용하고 있는 원격 서버의 스펙을 확인하는 명령어들
    - 운영체제 사양 확인 : !cat /etc/issue.net
    - 메모리 사양 확인 : !cat /proc/meminfo
    - CPU 사양 확인 : !cat /proc/cpuinfo
    - 파이썬 버전 확인 : !python –version
    - Etc.

```
!cat /etc/issue.net
```

```
Ubuntu 18.04.5 LTS
```

```
[6]  !cat /proc/meminfo
```

```
MemTotal:       13305360 kB
MemFree:        10714216 kB
MemAvailable:   12530376 kB
Buffers:           79232 kB
Cached:          1870292 kB
SwapCached:            0 kB
Active:           950492 kB
Inactive:        1406892 kB
Active(anon):     369508 kB
```

```
!cat /proc/cpuinfo
```

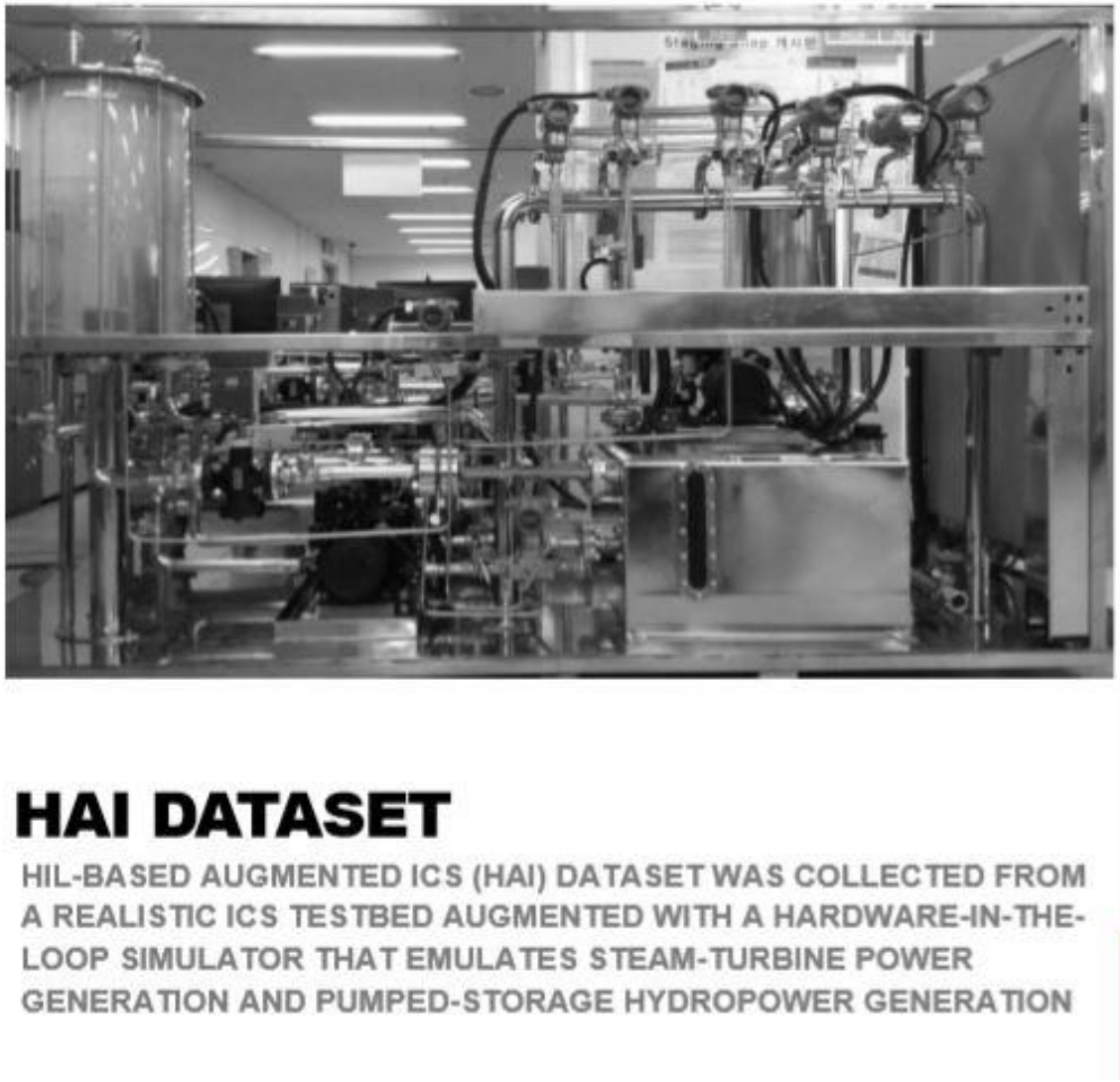
```
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 79
model name      : Intel(R) Xeon(R) CPU @ 2.20GHz
stepping        : 0
microcode       : 0x1
cpu MHz         : 2199.998
cache size      : 56320 KB
```

```
[6]  !python --version
```

```
Python 3.7.10
```

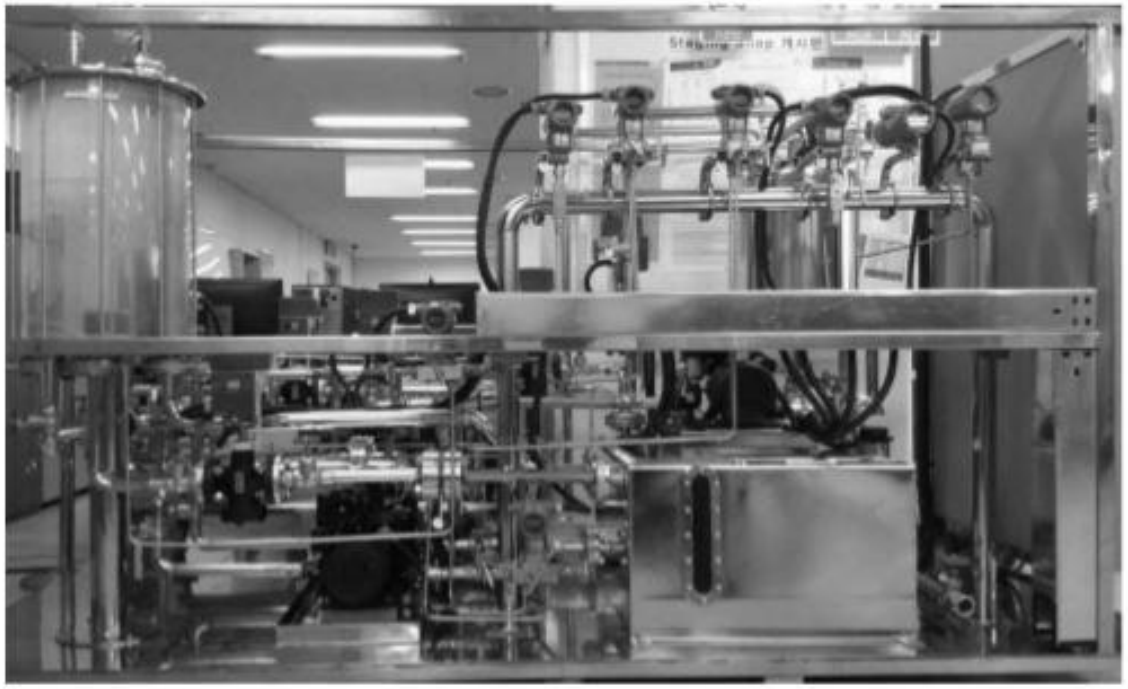
HAI Dataset 설명

- HAI (HIL-based Augmented ICS) Dataset
  - Hardware-In-the-Loop (HIL) 시뮬레이터를 통하여 보강된 실제 산업 제어 시스템 (Industrial Control System, ICS) 테스트베드에서 수집된 데이터
  - 테스트베드에서는 증기-터빈 발전 (steam-turbine power generation)과 양수-저장 수력 발전 (pumped-storage hydropower generation)을 모방



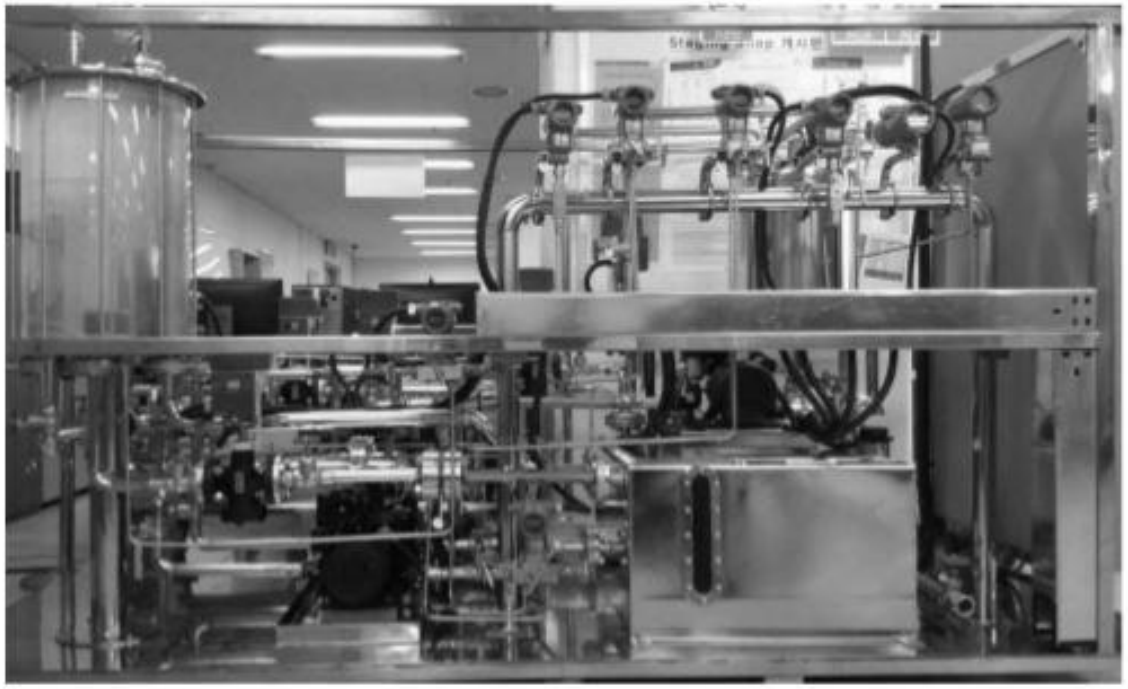
HAI Dataset 설명

- HAI (HIL-based Augmented ICS) Dataset
  - ➢국가보안기술연구소가 공개한 보안 데이터셋으로 산업제어시스템 보안기술 개발에 적합
  - ➢수작업에 의존한 기존 데이터셋의 데이터 레이블링 한계점을 극복해 데이터 신뢰성을 보장하고, 공격 수준별 데이터 생성이 가능하며 정밀한 성능평가가 가능함
  - ➢현재 세계적으로 가장 많이 사용되고 있는 싱가폴 SUTD 대학의 SWaT 데이터셋과 비슷한 크기



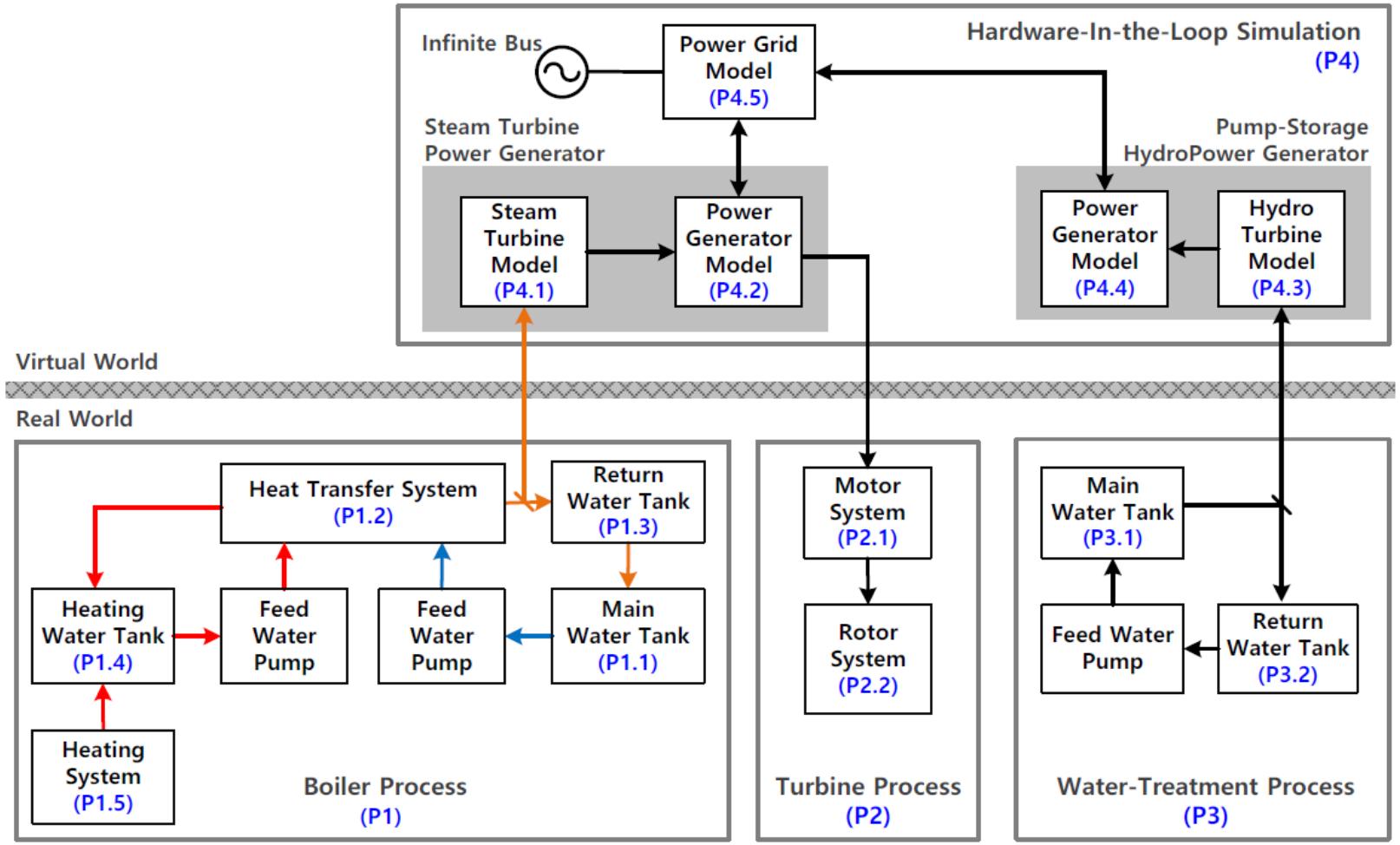
HAI Dataset 설명

- HAI (HIL-based Augmented ICS) Dataset
  - ➢HAI 데이터셋은 v1과 v2가 있음
  - ➢v1은 59개의 특징들로 이루어져 있으며, v2는 78개의 특징들로 이루어져 있음
  - ➢v1에서는 데이터 설명이 있어서 각 특징이 의미하는 바, 데이터들 간의 관계, 단위 등을 알 수 있지만, v2는 알 수 없음



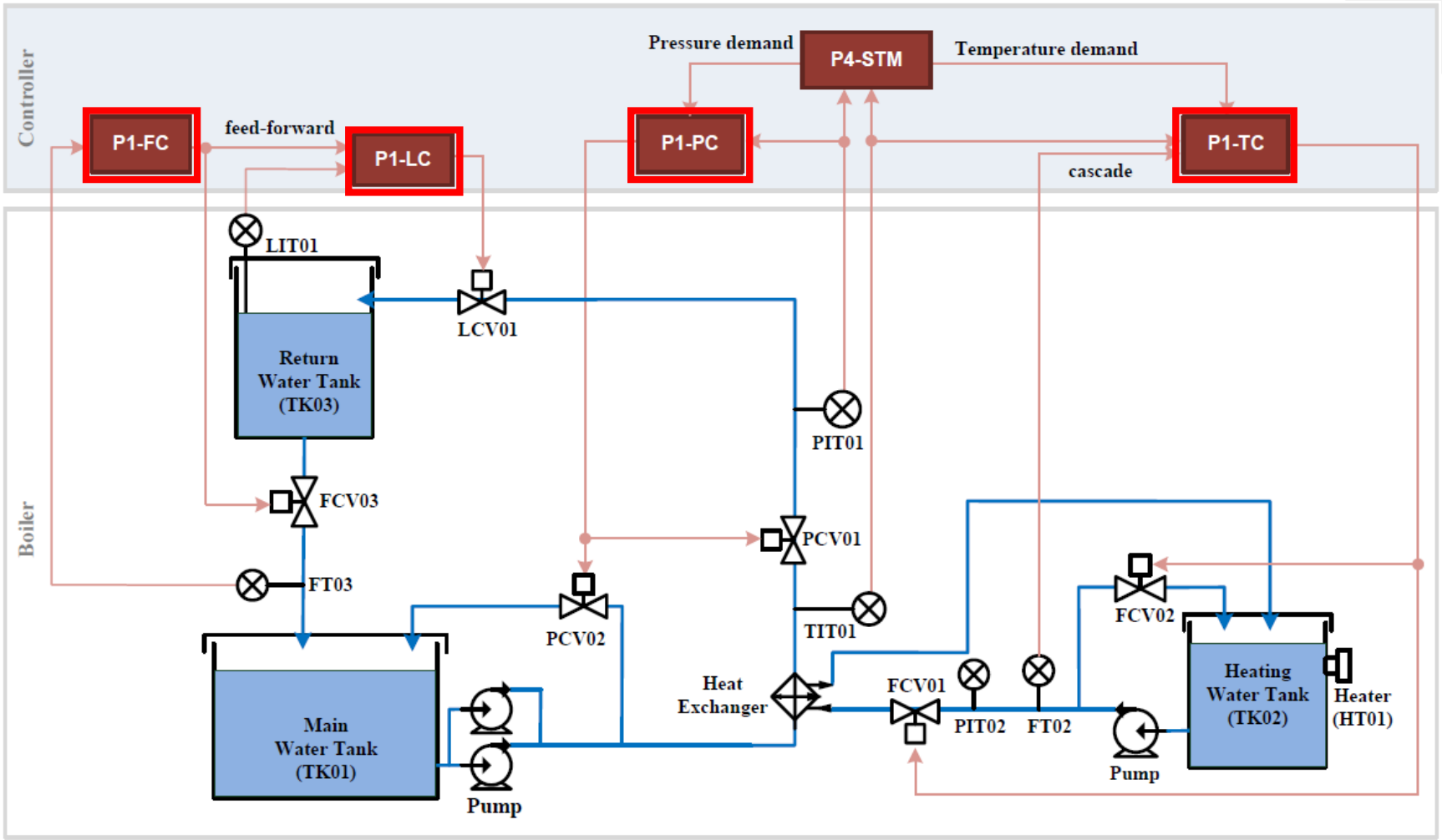
HAI Dataset 설명

- HAI Dataset
  - ➢4가지 공정(보일러, 터빈, 정수 처리, HIL 시뮬레이션)을 포함함



HAI Dataset 설명

- 보일러 조절



HAI Dataset 설명

- 보일러 조절
  - ➢관련 데이터

No	Point Name	Range	Unit	Description
1	P1_B2004	0 ~ 10	Bar	Heat-exchanger outlet pressure setpoint
2	P1_B2016	0 ~ 10	bar	Pressure demand to follow P1_B2004 and electrical load from steam turbine model
3	P1_B3004	0 ~ 720	mm	Water level setpoint in return water tank
4	P1_B3005	0 ~ 2,500	L/H	Water outflow rate setpoint from return water tank
5	P1_B4002	0 ~ 100	°C	Heat-exchanger outlet temperature setpoint
6	P1_B4005	0 ~ 100	-	Temperature cascade control (On: 1, Off: 0)
7	P1_B400B	0 ~ 3,000	L/H	Water outflow rate setpoint from heating water tank
8	P1_B4022	0 ~ 100	°C	Temperature demand to follow P1. B4005 and electrical load from steam-turbine model
9	P1_FCV01D	0 ~ 100	%	Position command for FCV01 valve
10	P1_FCV01Z	0 ~ 100	%	Current position of FCV01 valve
11	P1_FCV02D	0 ~ 100	%	Position command for FCV02 valve
12	P1_FCV02Z	0 ~ 100	%	Current position of FCV02 valve
13	P1_FCV03D	0 ~ 100	%	Position command for FCV03 valve
14	P1_FCV03Z	0 ~ 100	%	Current position of FCV03 valve
15	P1_FT01	0 ~ 2500	mmH_2O	Digital value of FT01 flow transmitter

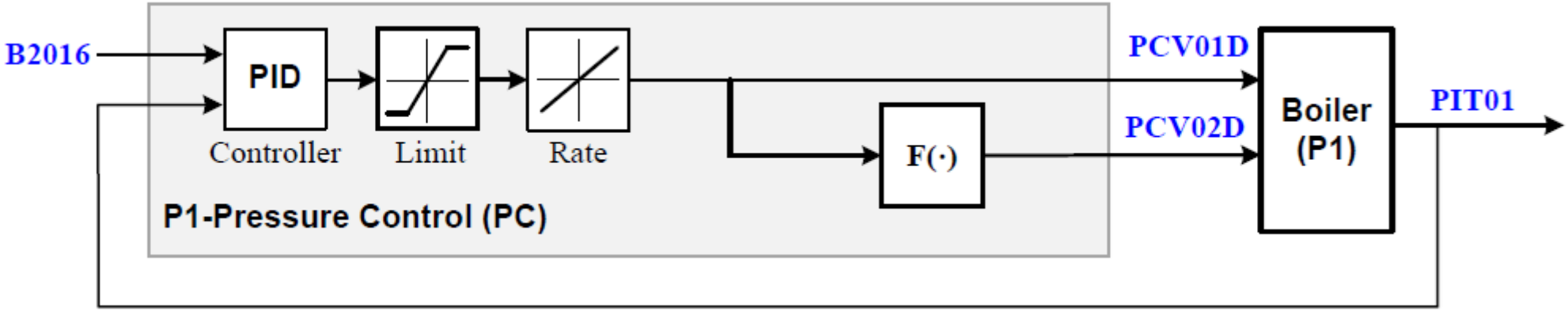
HAI Dataset 설명

- 보일러 조절
  - ➢관련 데이터

No	Point Name	Range	Unit	Description
16	P1_FT01Z	0 ~ 3190	L/H	Water inflow rate into return water tank
17	P1_FT02	0 ~ 2500	mmH_2O	Digital value of FT02 flow transmitter
18	P1_FT02Z	0 ~ 3190	L/H	Conversion from P1_FT02 to outflow rate at heating water tank
19	P1_FT03	0 ~ 2500	mmH_2O	Digital value of FT03 flow transmitter
20	P1_FT03Z	0 ~ 3190	L/H	Conversion from P1_FT03 to outflow rate at return water tank
21	P1_LCV01D	0 ~ 100	%	Position command for LCV01 valve
22	P1_LCV01Z	0 ~ 100	%	Current position of LCV01 valve
23	P1_LIT01	0 ~ 720		Water level of return water tank
24	P1_PCV01D	0 ~ 100	%	Position command for PCV01 valve
25	P1_PCV01Z	0 ~ 100	%	Current position of PCV01 valve
26	P1_PCV02D	0 ~ 100	%	Position command for PCV2 valve
27	P1_PCV02Z	0 ~ 100	%	Current position of PCV02 valve
28	P1_PIT01	0 ~ 10	bar	Heat-exchanger outlet pressure
29	P1_PIT02	0 ~ 10	bar	Water supply pressure of heating water pump
30	P1_TIT01	-10 ~ 100	℃	Heat-exchanger outlet temperature
31	P1_TIT02	-10 ~ 100	℃	Temperature of heating water tank

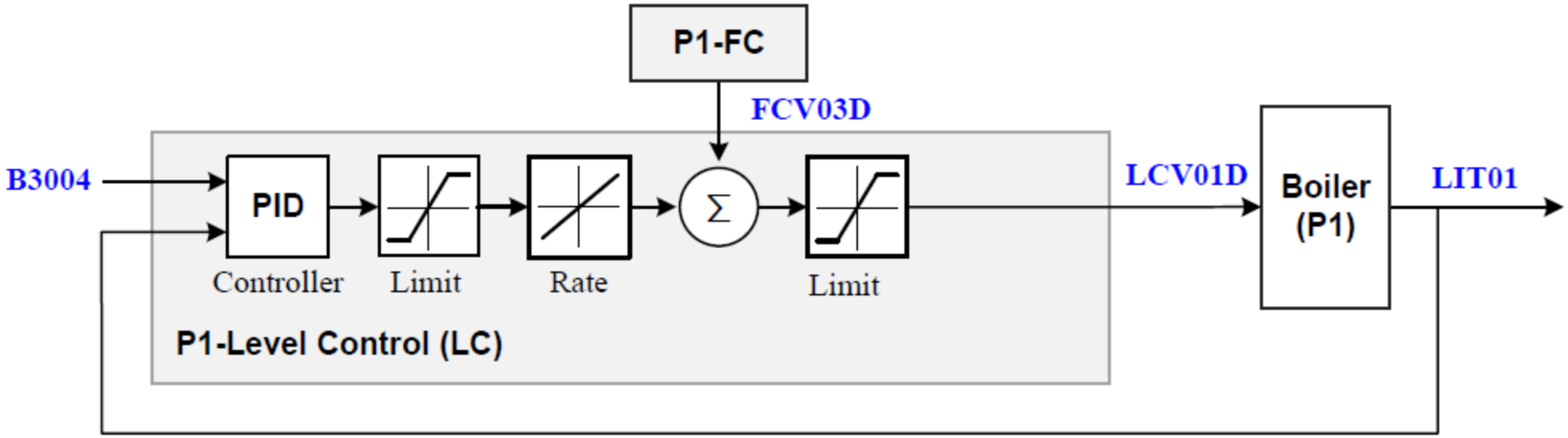
HAI Dataset 설명

- 보일러 조절
  - ➢압력 조절
    - 두개의 압력 제어 밸브를 제어하여 설정에 따라 메인 물 탱크와 회수 물 탱크 사이의 압력을 유지하도록 제어



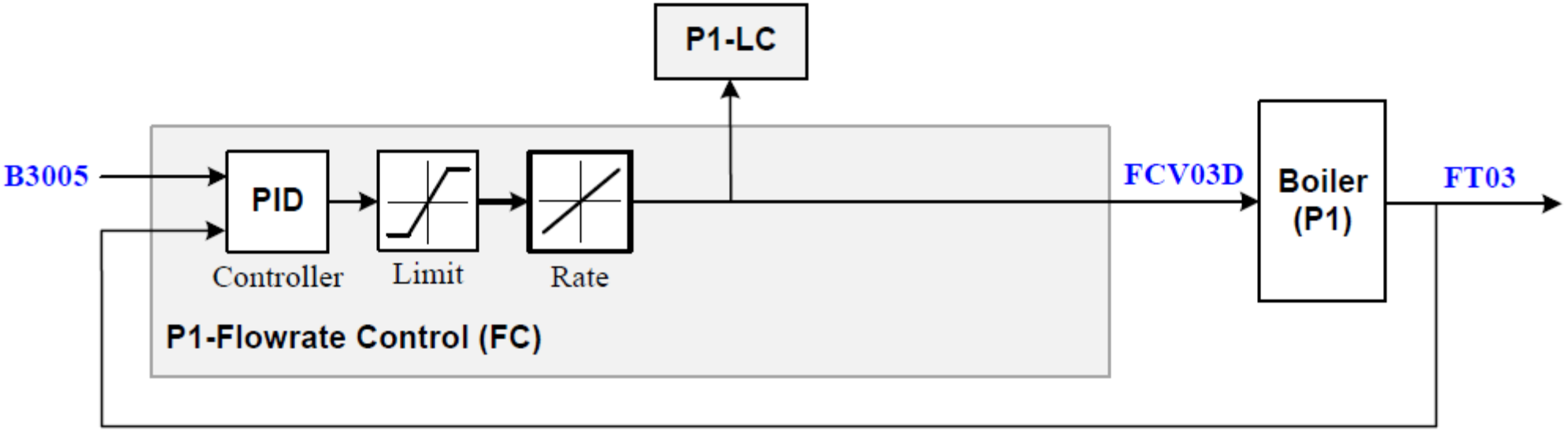
HAI Dataset 설명

- 보일러 조절
  - ➢수위 조절
    - 설정에 따라 회수 물 탱크 사이의 수위를 유지하기 위하여 밸브를 제어



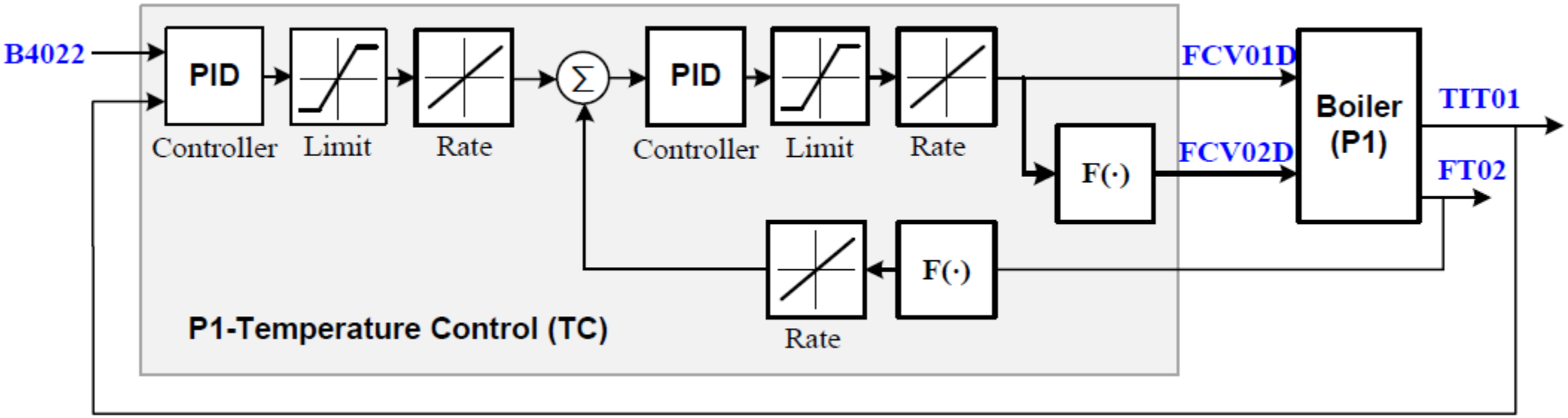
HAI Dataset 설명

- 보일러 조절
  - ➢유속 조절
    - 밸브를 제어하여 설정에 따라 회수 물탱크의 유출량을 유지하도록 제어



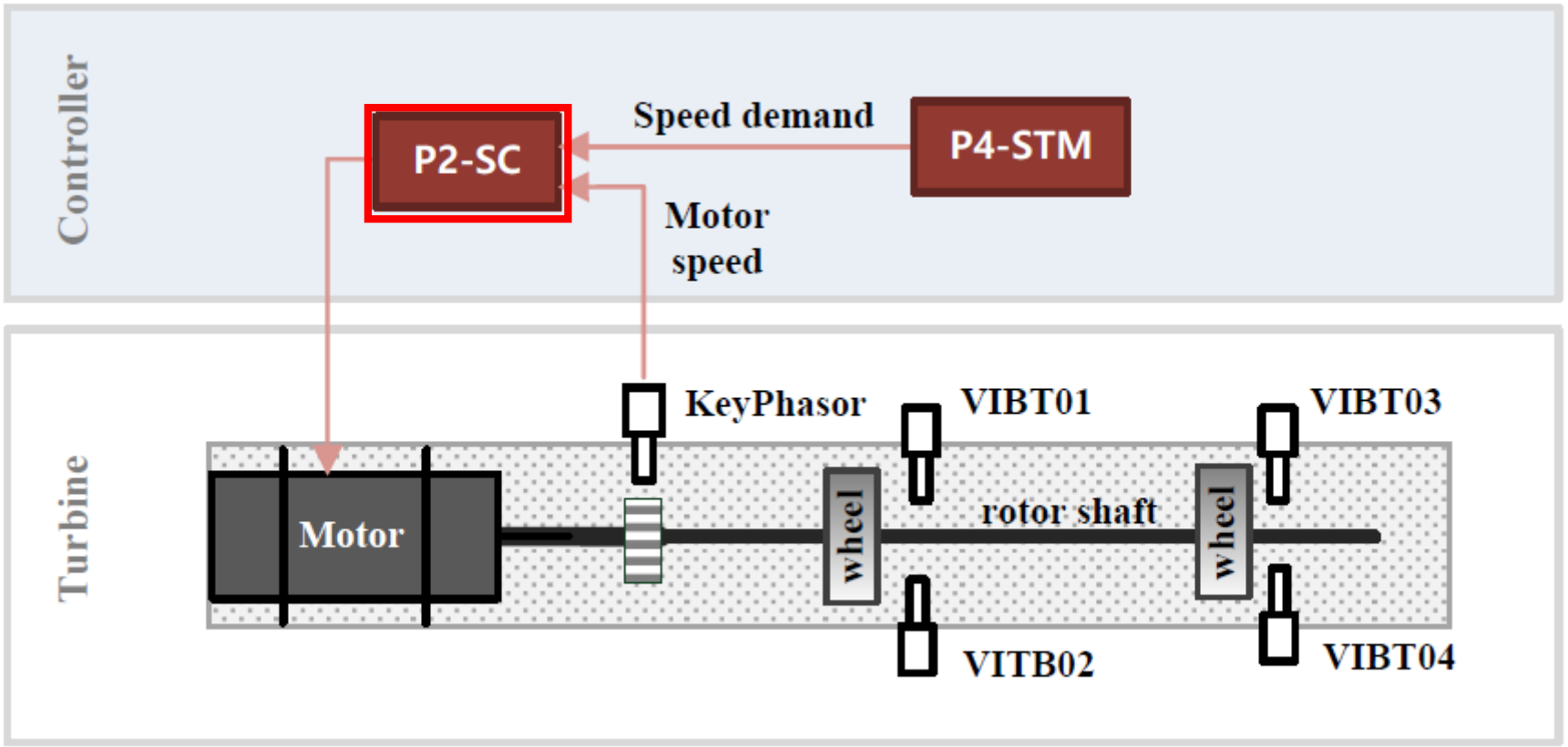
HAI Dataset 설명

- 보일러 조절
  - ➢온도 조절
    - 두개의 밸브를 제어하여 설정에 따라 회수 메인 용기의 온도를 유지



HAI Dataset 설명

- 터빈 조절



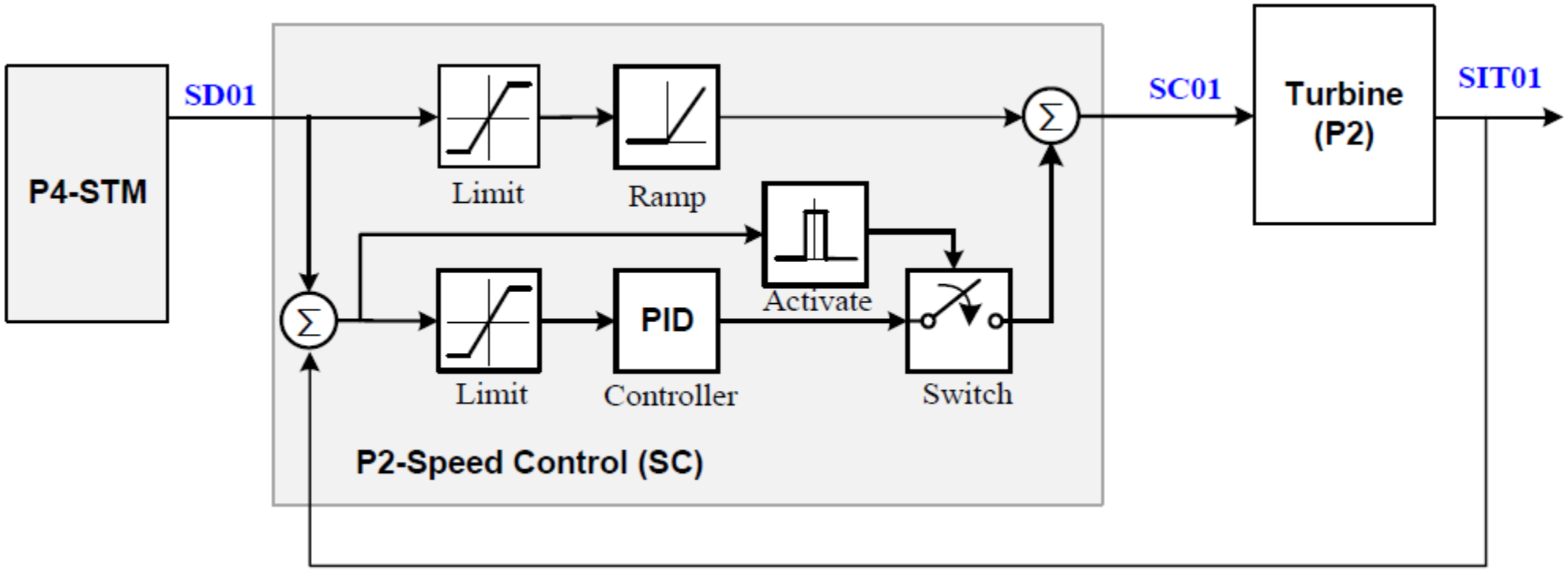
HAI Dataset 설명

- 터빈 조절
  - ➢관련 데이터

No	Point Name	Range	Unit	Description
32	P2_SIT01	0 ~ 3600	RPM	Current motor speed
33	P2_SD01	0 ~ 3600	RPM	User speed demand
34	P2_VT01	0 ~ 15	V	Phase lag signal of key phasor probe near motor
35	P2_VYT02	-10 ~ 10	μm	Shaft-vibration-related Y-axis displacement near the first mass wheel
36	P2_VXT02	-10 ~ 10	μm	Shaft-vibration-related X-axis displacement near the first mass wheel
37	P2_VYT03	-10 ~ 10	μm	Shaft-vibration-related Y-axis displacement near the second mass wheel
38	P2_VXT03	-10 ~ 10	μm	Shaft-vibration-related X-axis displacement near the second mass wheel
39	P2_24Vdc	0 ~ 30	V	DCS power supply
40	P2_Auto	0 or 1	-	System auto/manual mode
41	P2_Emgy	0 or 1	-	Emergency-stop input
42	P2_On	0 or 1	-	System on/off input
43	P2_TripEx	0 or 1	-	Trip exit input

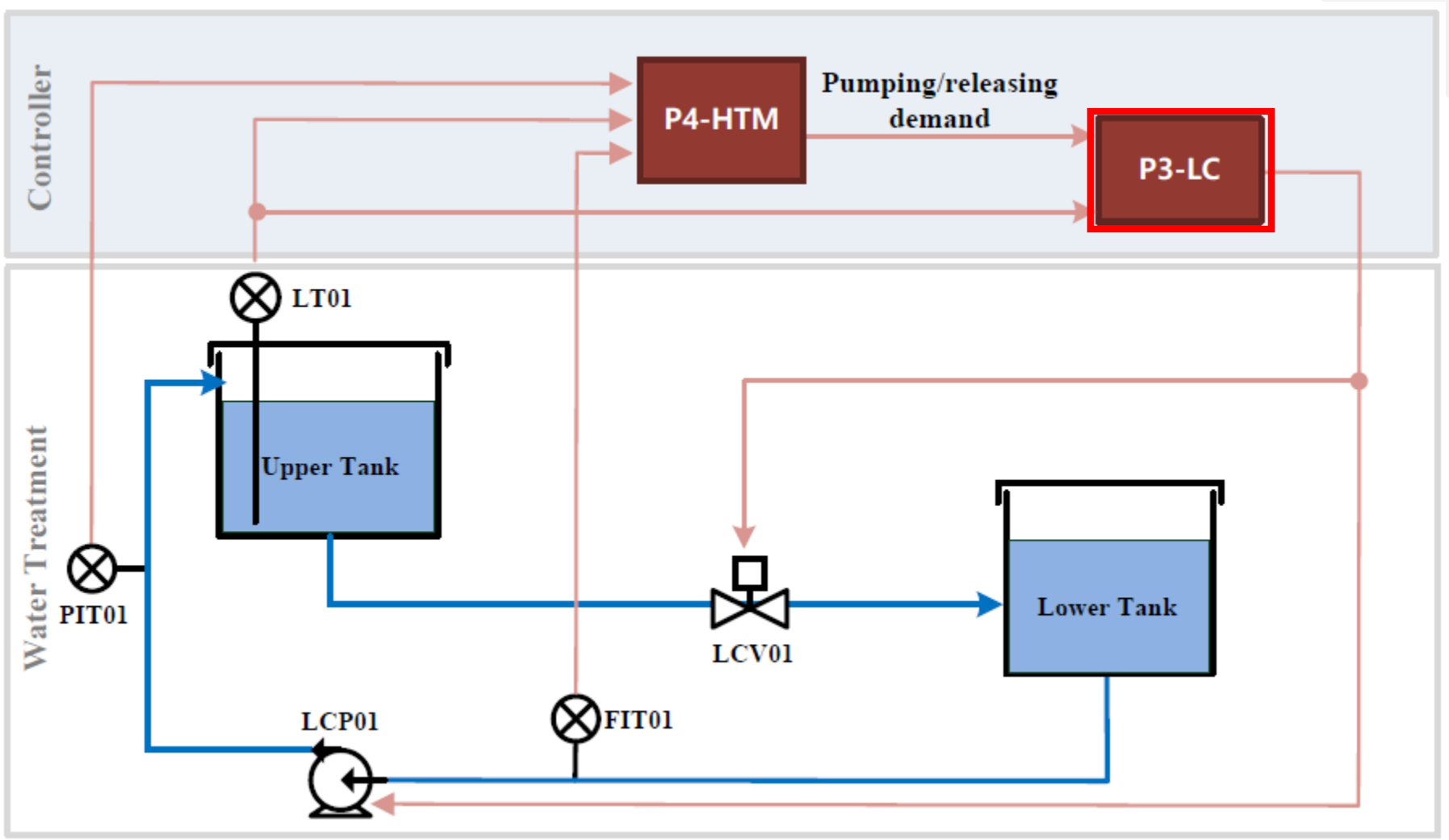
HAI Dataset 설명

- 터빈 조절
  - ➢속도 조절
    - 모터를 최소 제어 속도로 움직이게 만든 다음, PID 컨트롤러와의 결합 제어를 통해 모터의 속도를 설정한 값에 가까워지게 조절



HAI Dataset 설명

- 정수처리 조절



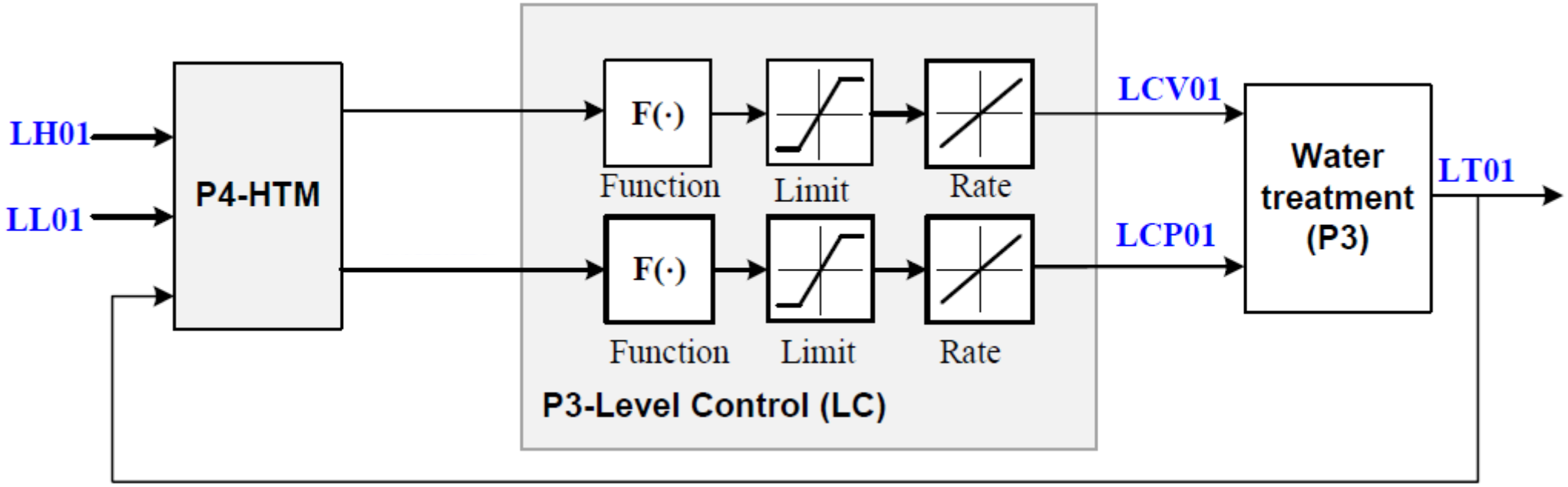
HAI Dataset 설명

- 정수처리 조절
  - ➢관련 데이터

No	Point Name	Range	Unit	Description
44	P3_LT01	0 ~ 100	%	Water level in upper tank
45	P3_LH01	0 ~ 100	%	High water level setpoint
46	P3_LL01	0 ~ 100	%	Low water level setpoint
47	P3_LCP01D	0 ~ 27648	-	Speed command for feed water pump
48	P3_LCV01D	0 ~ 27648	-	Position command for LCV01 valve

HAI Dataset 설명

- 정수처리 조절
  - ➢수위 조절
    - HIL 시뮬레이터의 배출 및 펌핑 요구량을 조정하여 밸브와 펌프를 조절



HAI Dataset 설명

- HIL 시뮬레이션 조절
  - ➢관련 데이터

No	Point Name	Range	Unit	Description
49	P4_LD	0 ~ 600	MW	Total electrical load demand
50	P4_ST_FD	-0.1 ~ 0.1	mHz	Frequency deviation of steam-turbine model
51	P4_ST_PO	0 ~ 500	MW	Output power of steam-turbine model
52	P4_ST_PT01	0 ~ 27648	-	Digital value of steam pressure in steam-turbine model
53	P4_ST_TT01	0 ~ 27648	-	Digital value of steam temperature in steam-turbine model
54	P4_ST_LD	0 ~ 500	MW	Electrical load demand for steam-turbine model
55	P4_ST_PS	0 ~ 500	MW	Scheduled power demand of steam-turbine model
56	P4_HT_FD	-0.1 ~ 0.1	mHz	Frequency deviation of hydropower-turbine model
57	P4_HT_PO	0 ~ 100	MW	Output power of hydropower-turbine model
58	P4_HT_LD	0 ~ 100	MW	Electrical load demand for steam-turbine model
59	P4_HT_PS	0 ~ 100	MW	Scheduled power demand of hydropower-turbine model

평가 지표

- 분류(Classification) 문제에 사용되는 평가 지표
  - ➢혼동 행렬(Confusion Matrix)을 기반으로 측정하는 정확도(Accuracy), 정밀도(Precision), 재현율(Recall), F1-Score 등이 있음
    - 혼동 행렬이란 모델의 분류 결과와 실제 정답을 행렬 형태로 아래와 같이 나타낸 것

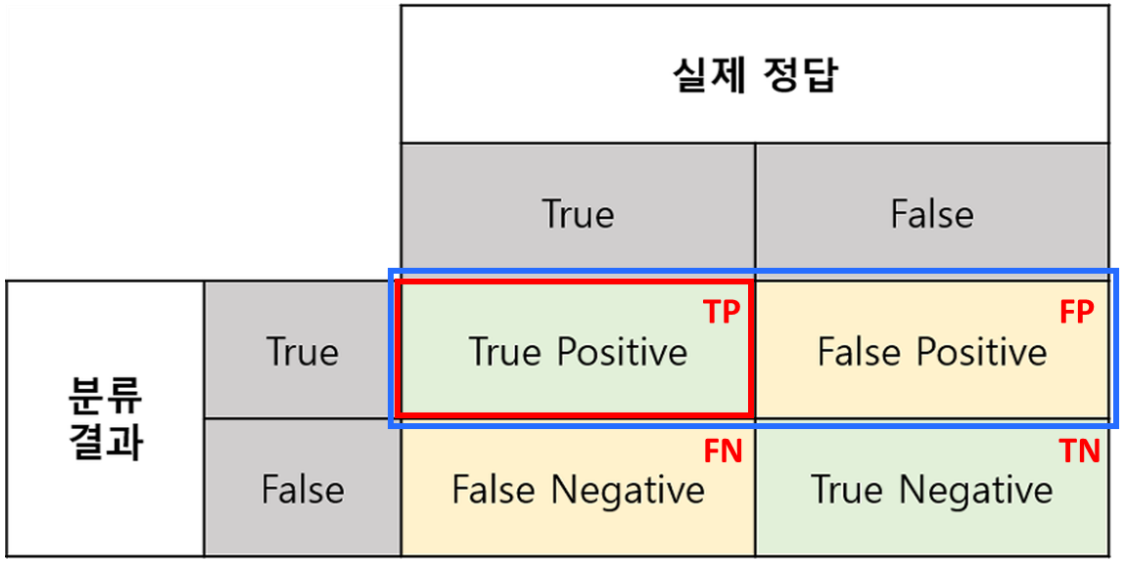
		실제 정답	
		True	False
분류 결과	True	True Positive	False Positive
	False	False Negative	True Negative

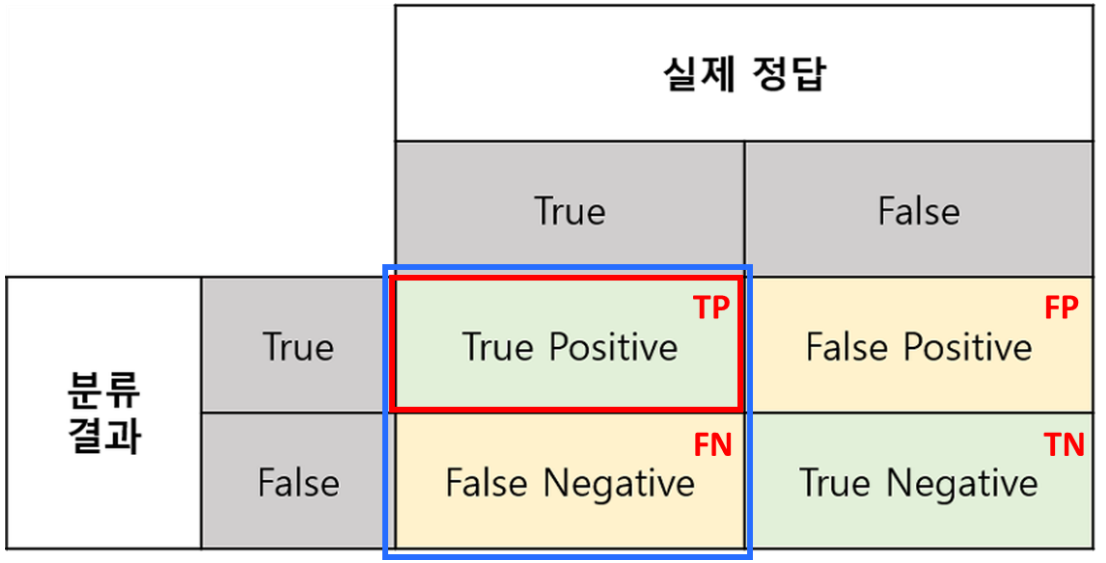
평가 지표

- 분류(Classification) 문제에 사용되는 평가 지표
  - ➢다양한 평가지표 중 정밀도와 재현율이 널리 사용됨
    - 정밀도 : 예측을 Positive로 한 대상 중 예측과 실제 값이 일치하는 비율
    - 재현율 : 실제 Positive인 대상 중에서 예측과 실제 값이 일치하는 비율

$$\text{Precision} = \frac{TP}{FP+TP}$$

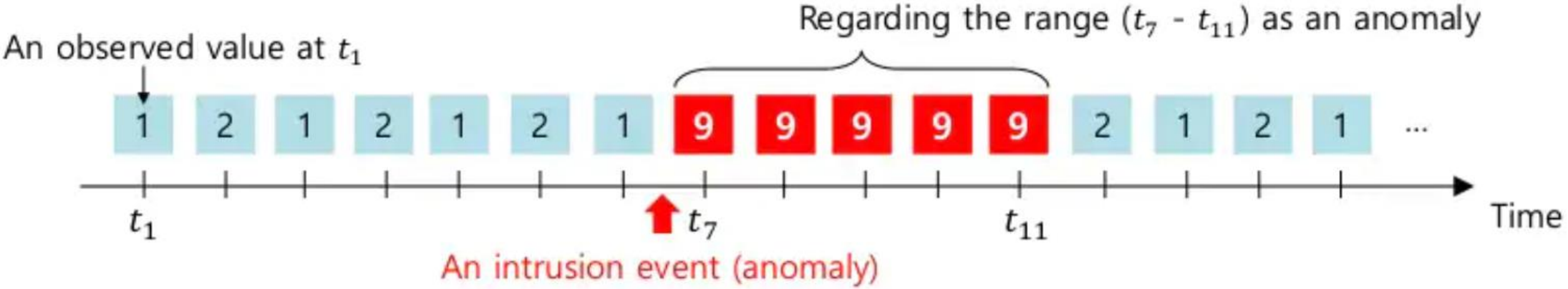
$$\text{Recall} = \frac{TP}{FN+TP}$$





평가 지표

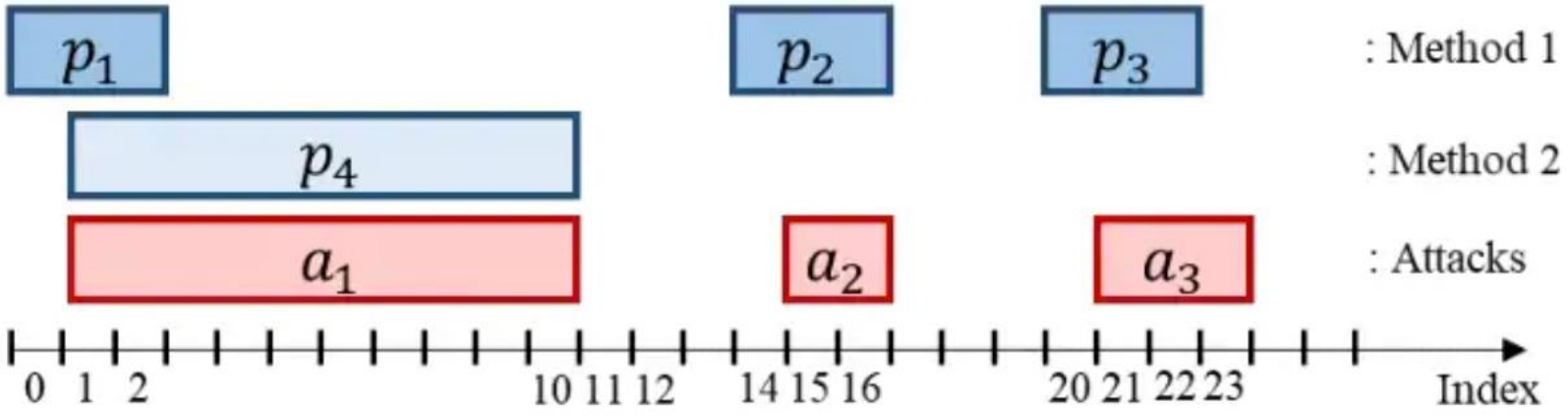
- 시계열 데이터에서의 이상 탐지를 위한 평가지표
  - ➢시계열 데이터가 아닌 경우에는 단순히 데이터에 이상이 있는지 없는지 만 판단하면 됨
  - ➢하지만, 시계열 데이터의 경우 범위도 굉장히 중요한 의미를 가지고 있음



평가 지표

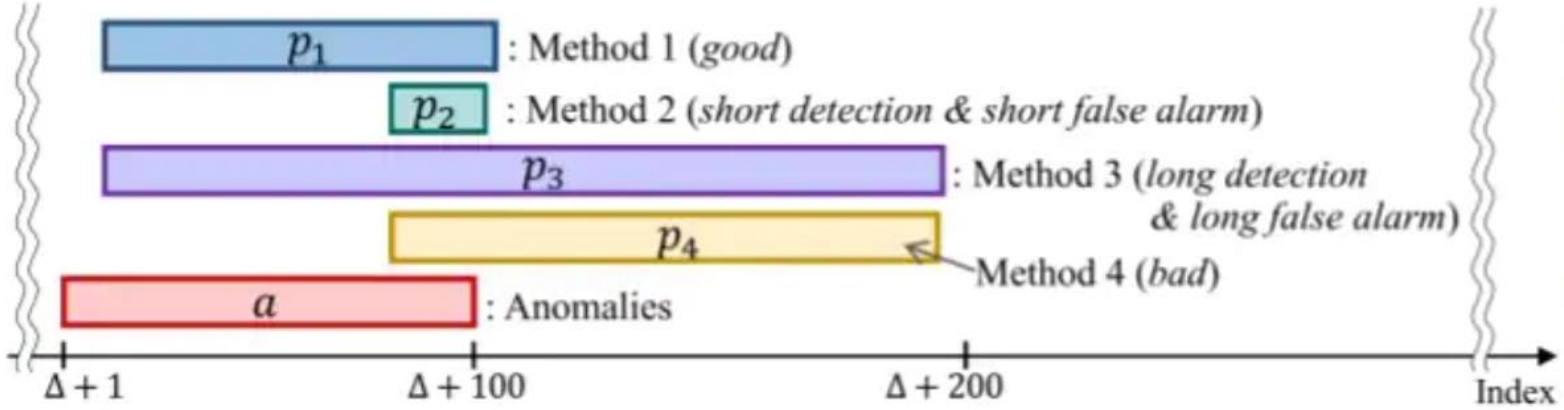
- 시계열 데이터에서의 이상 탐지를 위한 평가지표
  - ➢또한, 정밀도와 재현율은 다양한 원인으로 인하여 발생한 이상을 탐지하였는지를 평가하기에는 부적합
    - 아래 그림에서 방법2는 a1만을 탐지하였으나 높은 점수를 받는 것을 확인할 수 있음

Method	Metric	
	Precision	Recall
1	0.67	0.40
2	1.00	0.67



평가 지표

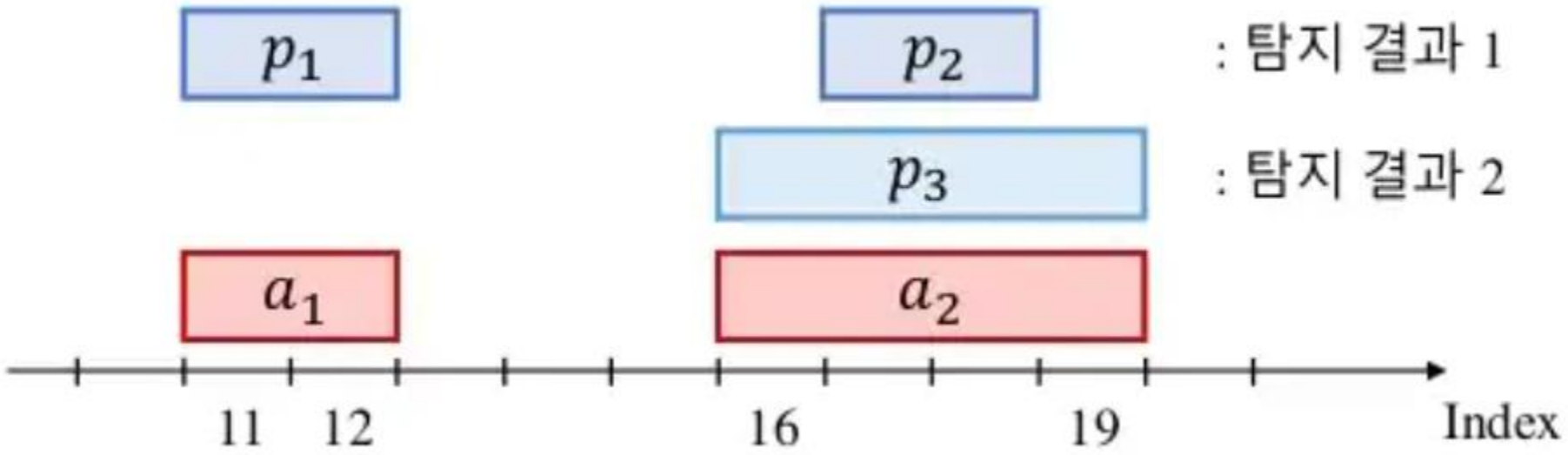
- 시계열 데이터에서의 이상 탐지를 위한 평가지표
  - ➢데이터에 이상이 있는지에 대한 여부와 더불어 범위까지 측정할 수 있는 평가지표를 사용해야 함
  - ➢TaPR은 위의 조건을 만족하는 평가를 제공
    - TaPR : Time-series aware Precision and Recall for anomaly detection
    - TaR과 TaP로 구성이 되어 있음
    - TaP은 예측 결과가 오탐 없이 이상징후를 찾아내는 지를 나타내고, TaR은 얼마나 다양한 공격 범위를 찾아내는지를 나타냄



예측 결과	TaP	TaR
p_1	High	High
p_2	High	Low
p_3	Low	High
p_4	Low	Low

평가 지표

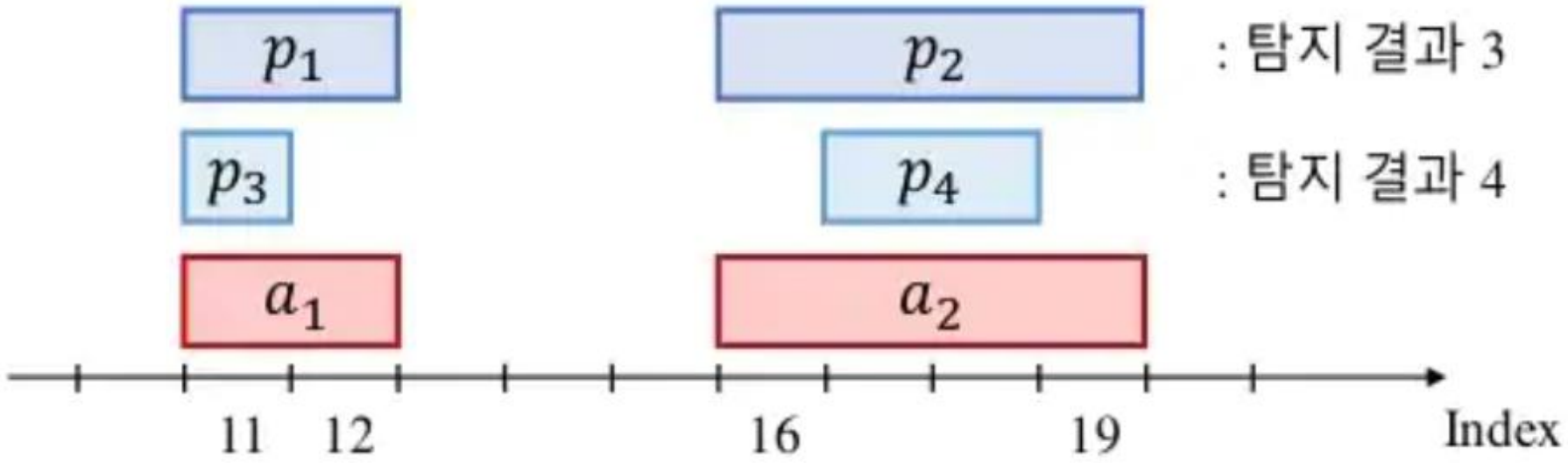
- TaPR의 평가 목표
  - **목표 1. 탐지된 공격의 다양성 평가**
  - 목표 2. 탐지의 정확성
  - 목표 3. 낮은 오탐



결과 1이 더 높게 평가 됨

평가 지표

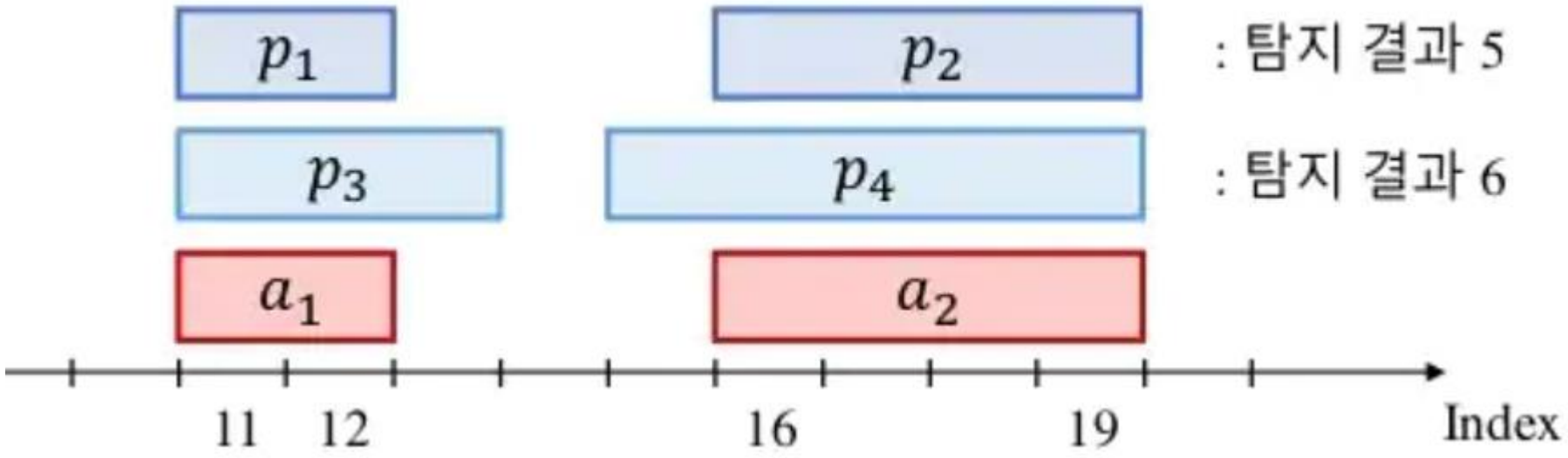
- TaPR의 평가 목표
  - 목표 1. 탐지된 공격의 다양성 평가
  - **목표 2. 탐지의 정확성**
  - 목표 3. 낮은 오탐



결과 3이 더 높게 평가 됨

평가 지표

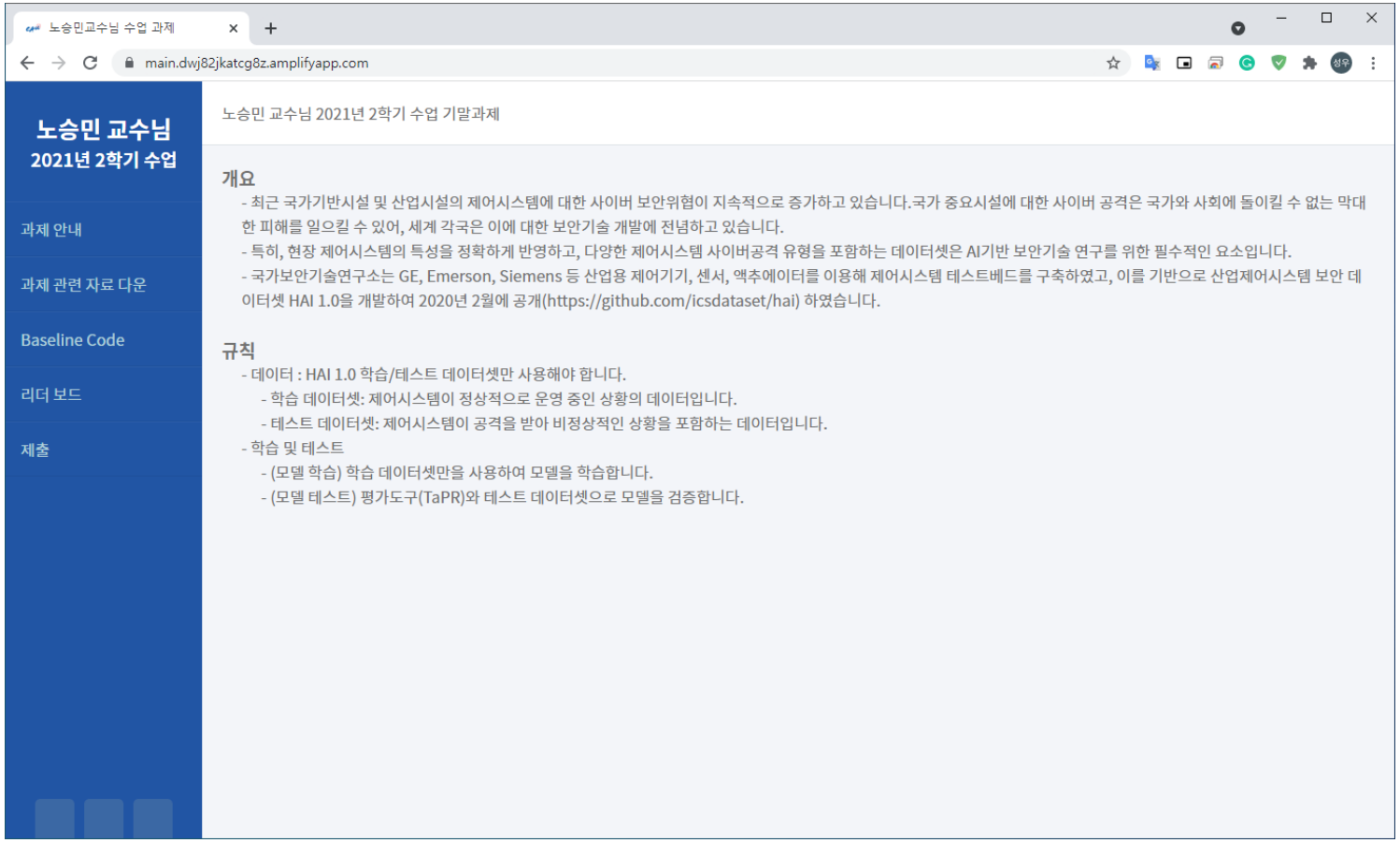
- TaPR의 평가 목표
  - 목표 1. 탐지된 공격의 다양성 평가
  - 목표 2. 탐지의 정확성
  - **목표 3. 낮은 오탐**



결과 5가 더 높게 평가 됨

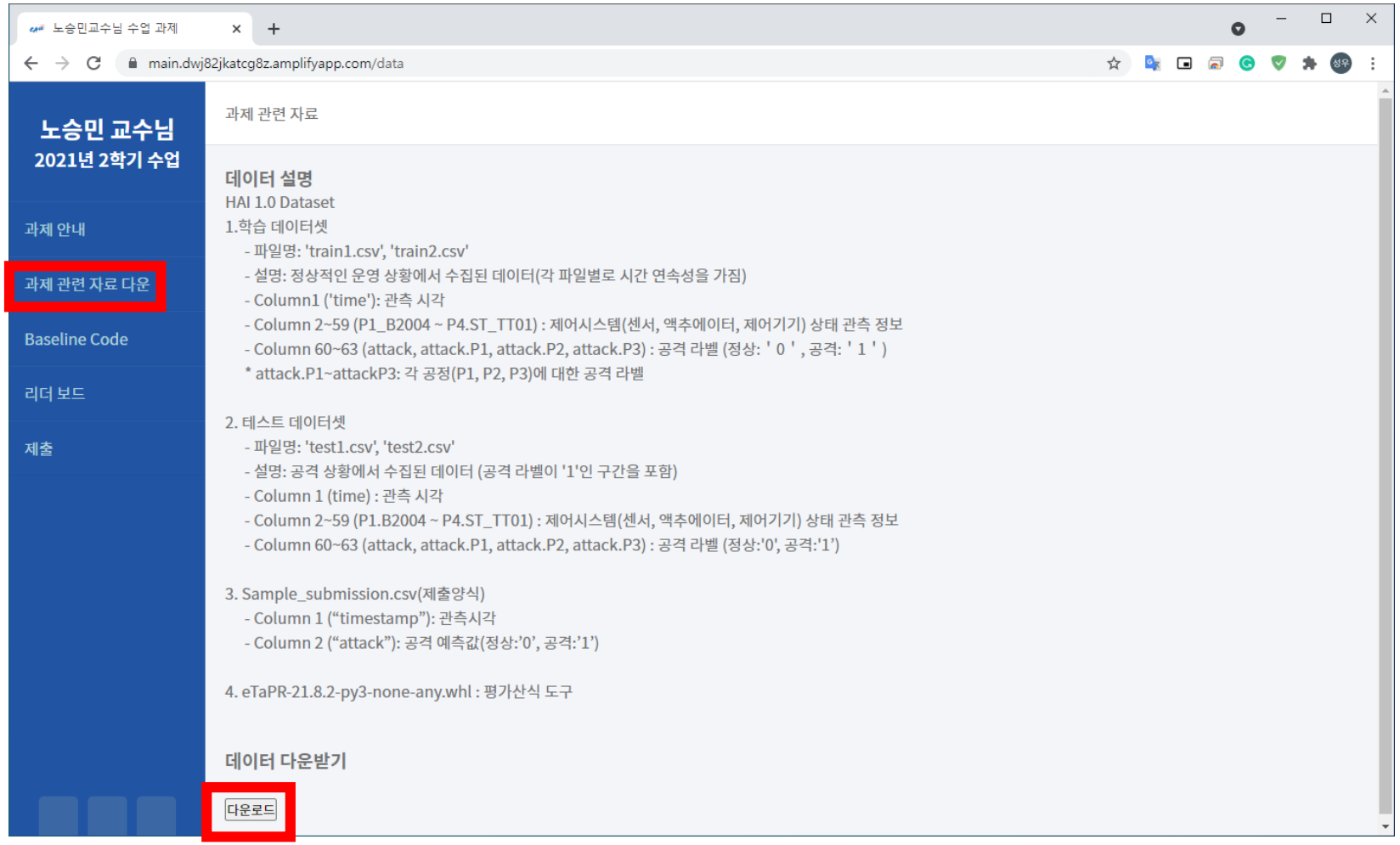
Basic Code

- HAI 데이터셋 다운로드
  - ‘https://main.dwj82jkatcg8z.amplifyapp.com/’에서 HAI 데이터셋 다운 가능



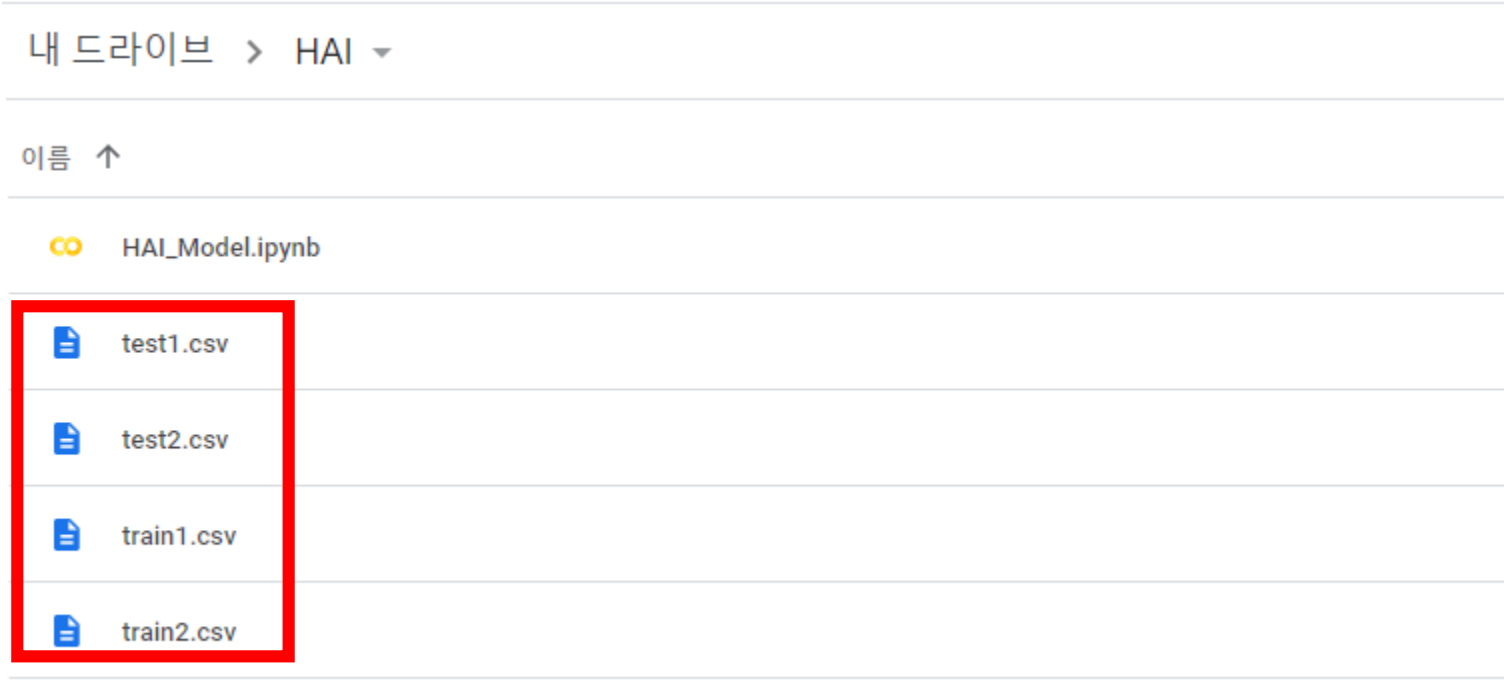
Basic Code

- HAI 데이터셋 다운로드
  - ➢과제 관련 자료 다운 탭에서 데이터셋 및 평가지표 라이브러리 다운 가능



Basic Code

- HAI 데이터셋을 구글 드라이브에 옮기기
  - ➢train1, train2, test1, test2 파일들을 구글 드라이브의 HAI 폴더로 드래그&드롭



Basic Code

- HAI 데이터셋을 Colab에서 불러오기
  - ➢구글 드라이브와 연결하여 데이터를 사용하는 방법의 과정을 통하여 Colab과 구글 드라이브를 연동
  - ➢"/content/drive/MyDrive/HAI/" 경로를 사용하여 데이터 접근 가능

```
import pandas as pd

train_data1 = pd.read_csv("/content/drive/MyDrive/HAI/train1.csv", sep=";", engine="python")
train_data2 = pd.read_csv("/content/drive/MyDrive/HAI/train2.csv", sep=";", engine="python")
```

```
print(train_data1)
```

```
                     time  P1_B2004  ...  attack_P2  attack_P3
0       2019-09-11 20:00:00    0.0983  ...          0          0
1       2019-09-11 20:00:01    0.0983  ...          0          0
2       2019-09-11 20:00:02    0.0983  ...          0          0
3       2019-09-11 20:00:03    0.0983  ...          0          0
4       2019-09-11 20:00:04    0.0983  ...          0          0
...                     ...       ...  ...        ...        ...
309595  2019-09-15 09:59:55    0.0304  ...          0          0
309596  2019-09-15 09:59:56    0.0304  ...          0          0
309597  2019-09-15 09:59:57    0.0304  ...          0          0
309598  2019-09-15 09:59:58    0.0304  ...          0          0
309599  2019-09-15 09:59:59    0.0304  ...          0          0

[309600 rows x 64 columns]
```

Basic Code

- Train 데이터들을 하나의 데이터 프레임으로 만들기
  - ➢"/content/drive/MyDrive/HAI/" 경로를 사용하여 데이터 접근 가능
  - ➢read_csv() 함수를 통해 csv 파일 불러오기

```
import pandas as pd

train_data1 = pd.read_csv("/content/drive/MyDrive/HAI/train1.csv", sep=";", engine="python")
train_data2 = pd.read_csv("/content/drive/MyDrive/HAI/train2.csv", sep=";", engine="python")
```

```
print(train_data1)
```

```
                     time  P1_B2004  ...  attack_P2  attack_P3
0       2019-09-11 20:00:00    0.0983  ...          0          0
1       2019-09-11 20:00:01    0.0983  ...          0          0
2       2019-09-11 20:00:02    0.0983  ...          0          0
3       2019-09-11 20:00:03    0.0983  ...          0          0
4       2019-09-11 20:00:04    0.0983  ...          0          0
...                     ...       ...  ...        ...        ...
309595  2019-09-15 09:59:55    0.0304  ...          0          0
309596  2019-09-15 09:59:56    0.0304  ...          0          0
309597  2019-09-15 09:59:57    0.0304  ...          0          0
309598  2019-09-15 09:59:58    0.0304  ...          0          0
309599  2019-09-15 09:59:59    0.0304  ...          0          0

[309600 rows x 64 columns]
```

Basic Code

- Train 데이터들을 하나의 데이터 프레임으로 만들기
  - ➢2개로 나뉘어져 있는 train 데이터를 하나로 통합
    - concat() 함수를 통하여 2개의 데이터 프레임을 1개로 통합
    - reset_index() 함수를 통해 데이터 프레임의 index를 다시 설정

```
total_train_data = pd.concat([train_data1, train_data2], axis=0).reset_index(drop=True)
```

```
print(total_train_data)
```

```
                      time  P1_B2004  ...  attack_P2  attack_P3
0       2019-09-11 20:00:00    0.0983  ...          0          0
1       2019-09-11 20:00:01    0.0983  ...          0          0
2       2019-09-11 20:00:02    0.0983  ...          0          0
3       2019-09-11 20:00:03    0.0983  ...          0          0
4       2019-09-11 20:00:04    0.0983  ...          0          0
...                     ...       ...  ...        ...        ...
550795  2019-11-04 14:59:55    0.1001  ...          0          0
550796  2019-11-04 14:59:56    0.1001  ...          0          0
550797  2019-11-04 14:59:57    0.1001  ...          0          0
550798  2019-11-04 14:59:58    0.1001  ...          0          0
550799  2019-11-04 14:59:59    0.1001  ...          0          0

[550800 rows x 64 columns]
```

Basic Code

- Test 데이터들을 하나의 데이터 프레임으로 만들기
  - ➢Train 데이터와 동일한 과정을 거쳐 하나의 데이터 프레임으로 만들기

```
test_data1 = pd.read_csv("/content/drive/MyDrive/HAI/test1.csv", sep=";", engine="python")
test_data2 = pd.read_csv("/content/drive/MyDrive/HAI/test2.csv", sep=";", engine="python")
```

```
total_test_data = pd.concat([test_data1, test_data2], axis=0).reset_index(drop=True)
```

```
print(total_test_data)
```

```
                     time  P1_B2004  ...  attack_P2  attack_P3
0       2019-10-29 11:00:00    0.0982  ...          0          0
1       2019-10-29 11:00:01    0.0982  ...          0          0
2       2019-10-29 11:00:02    0.0982  ...          0          0
3       2019-10-29 11:00:03    0.0982  ...          0          0
4       2019-10-29 11:00:04    0.0982  ...          0          0
...                     ...       ...  ...        ...        ...
444595  2019-11-06 09:29:55    0.0304  ...          0          0
444596  2019-11-06 09:29:56    0.0304  ...          0          0
444597  2019-11-06 09:29:57    0.0304  ...          0          0
444598  2019-11-06 09:29:58    0.0304  ...          0          0
444599  2019-11-06 09:29:59    0.0304  ...          0          0

[444600 rows x 64 columns]
```

Basic Code

- Train 데이터를 특징 데이터와 라벨 데이터로 분리
  - ➢시간 데이터인 time과, 라벨과 관련된 attack, attack_P1, attack_P2, attack_P3 변수를 빼고 특징 데이터 구성
    - drop() 함수를 통하여 총 변수 리스트에서 특정 변수 제거

```
LABEL_COLUMN = "attack"
FEATURE_COLUMN = total_train_data.columns.drop(["time", "attack", "attack_P1", "attack_P2", "attack_P3"])
```

```
train_x = total_train_data.loc[:,FEATURE_COLUMN]
train_y = total_train_data.loc[:,[LABEL_COLUMN]]
print(train_x)
print(train_y)
```

```
        P1_B2004  P1_B2016  P1_B3004  ...  P4_ST_PS  P4_ST_PT01  P4_ST_TT01
0         0.0983    1.0702   399.2321  ...   50.9871      9973.0     27629.0
1         0.0983    1.0699   399.2321  ...   50.9871      9973.0     27629.0
2         0.0983    1.0703   399.2321  ...   50.9871      9973.0     27629.0
3         0.0983    1.0719   399.2321  ...   50.9871      9973.0     27629.0
4         0.0983    1.0710   399.2321  ...   50.9871      9973.0     27629.0
...          ...       ...        ...  ...       ...         ...        ...
550795    0.1001    1.5194   395.3508  ...    0.0000     10026.0     27548.0
550796    0.1001    1.5220   395.3508  ...    0.0000     10026.0     27549.0
550797    0.1001    1.5235   395.3508  ...    0.0000     10026.0     27547.0
550798    0.1001    1.5279   395.3508  ...    0.0000     10026.0     27551.0
550799    0.1001    1.5281   395.3508  ...    0.0000     10026.0     27552.0

[550800 rows x 59 columns]
```

```
        attack
0            0
1            0
2            0
3            0
4            0
...        ...
550795       0
550796       0
550797       0
550798       0
550799       0

[550800 rows x 1 columns]
```

Basic Code

- Test 데이터를 특징 데이터와 라벨 데이터로 분리
  - ➢Train 데이터와 동일한 과정을 거쳐 특징 데이터와 라벨 데이터로 분리

```
test_x = total_test_data.loc[:,FEATURE_COLUMN]
test_y = total_test_data.loc[:,[LABEL_COLUMN]]
print(test_x)
print(test_y)
```

```
        P1_B2004  P1_B2016  P1_B3004  ...  P4_ST_PS  P4_ST_PT01  P4_ST_TT01
0         0.0982    1.4610   461.9883  ...       0.0     10053.0     27629.0
1         0.0982    1.4578   461.9883  ...       0.0     10053.0     27629.0
2         0.0982    1.4666   461.9883  ...       0.0     10053.0     27629.0
3         0.0982    1.4688   461.9883  ...       0.0     10053.0     27629.0
4         0.0982    1.4717   461.9883  ...       0.0     10053.0     27629.0
...          ...       ...        ...  ...       ...         ...         ...
444595    0.0304    1.2162   399.8923  ...       0.0     10013.0     27622.0
444596    0.0304    1.2118   399.8923  ...       0.0     10012.0     27626.0
444597    0.0304    1.2170   399.8923  ...       0.0     10012.0     27625.0
444598    0.0304    1.2193   399.8923  ...       0.0     10012.0     27625.0
444599    0.0304    1.2249   399.8923  ...       0.0     10013.0     27623.0

[444600 rows x 59 columns]
```

```
        attack
0            0
1            0
2            0
3            0
4            0
...        ...
444595       0
444596       0
444597       0
444598       0
444599       0

[444600 rows x 1 columns]
```

Basic Code

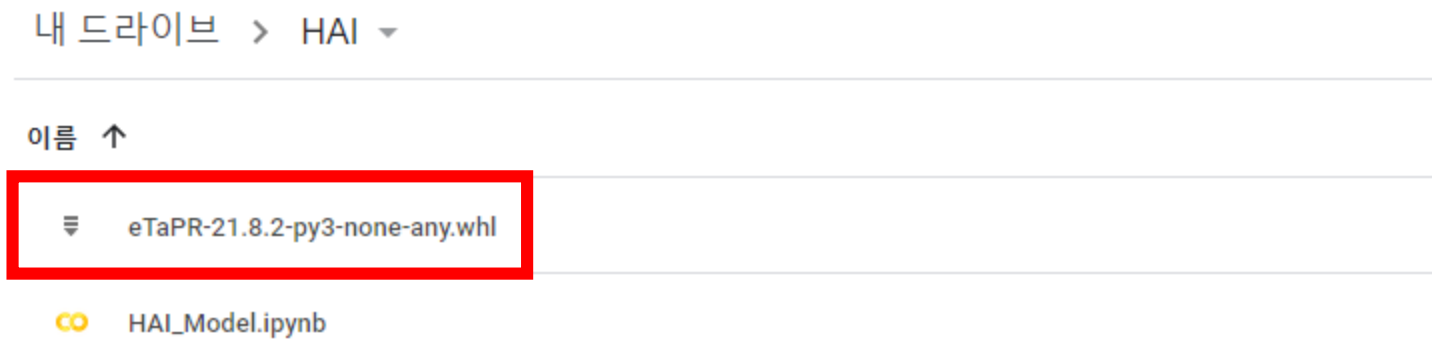
- Train 데이터를 사용하여 Random Forest 분류 모델 학습
  - ➢Scikit-learn에서 제공하는 Random Forest 모델을 사용
    - RandomForestClassifier() 함수를 통하여 모델 선언
    - fit() 함수를 통하여 train 데이터를 이용해 모델 학습
    - predict() 함수를 통하여 test 데이터의 입력변수를 넣어 결과 도출

```
from sklearn.ensemble import RandomForestClassifier

RF_Model = RandomForestClassifier(n_estimators=1024)
RF_Model.fit(train_x, train_y)
rf_result = RF_Model.predict(test_x)
```

Basic Code

- eTaPR을 사용하여 예측 성능 평가
  - ➢드라이브의 HAI 폴더에 rTaPR-21.8.2-py3-none-any.whl 을 넣은 후 pip 명령어를 통하여 해당 라이브러리 설치




```
!pip install /content/drive/MyDrive/HAI/eTaPR-21.8.2-py3-none-any.whl

Processing ./drive/MyDrive/HAI/eTaPR-21.8.2-py3-none-any.whl
Installing collected packages: eTaPR
Successfully installed eTaPR-21.8.2
```

Basic Code

- eTaPR을 사용하여 예측 성능 평가
  - ➢라이브러리를 사용하여 성능을 출력
    - evaluate_haicon() 함수를 통하여 성능 도출
    - 측정된 성능은 f1-score의 관점, 정확도의 관점, 재현율의 관점에서 측정되어 있음
    - 각각의 관점에서의 성능을 출력

```
from TaPR_pkg import etapr

TaPR = etapr.evaluate_haicon(anomalies=test_y, predictions=rf_result)
print(f"F1: {TaPR['f1']:.3f} (TaP: {TaPR['TaP']:.3f}, TaR: {TaPR['TaR']:.3f})")
print(f"# of detected anomalies: {len(TaPR['Detected_Anomalies'])}")
print(f"Detected anomalies: {TaPR['Detected_Anomalies']}")
```

```
F1: 0.035 (TaP: 0.169, TaR: 0.020)
# of detected anomalies: 2
Detected anomalies: [<TaPR_pkg.DataManage.Range.Range object at 0x00000214A21BBF08>, <TaPR_pkg.DataManage
.Range.Range object at 0x000002149DF62B88>]
```

Advanced Code 1

- Basic Code의 특징 데이터와 라벨 데이터 분리 이전부터 이어서 시작

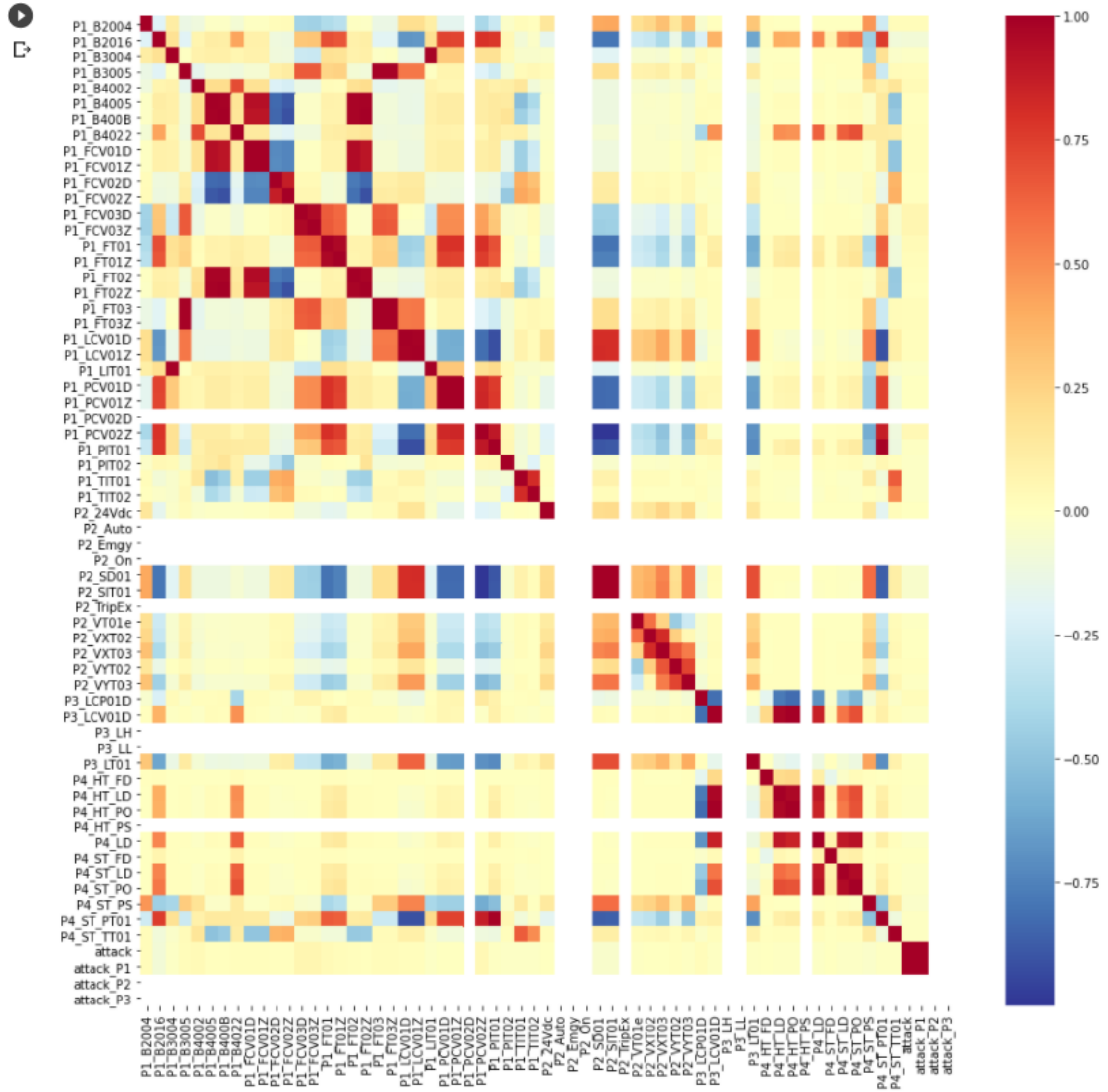
- Train 데이터에서 각 column들 간의 상관관계 확인하기
  - ➢데이터들 간의 상관관계를 구한 후 시각화
    - corr() 함수를 통하여 전체 column들 간의 상관관계를 도출
    - heatmap() 함수를 통해 상관관계를 히트맵 형식으로 시각화

```
import matplotlib.pyplot as plt
import seaborn as sns

fig, ax = plt.subplots( figsize=(15,15) )
sns.heatmap(total_train_data.corr(), cmap = 'RdYlBu_r')
```

Advanced Code 1

- Train 데이터에서 각 column들 간의 상관관계 확인하기



Advanced Code 1

- Train 데이터에서 각 column 별 데이터 분포 확인하기
  - ➢데이터를 통계적 분석할 수 있는 사용자 함수 선언
    - mean() 함수를 통해 평균
    - std() 함수를 통해 표준편차를 도출
    - median() 함수를 통해 중앙값을 도출
    - 평균, 표준편차, 중앙값을 사용하여 비대칭도(skew)를 계산
      - 데이터가 동일한 값으로만 이루어진 경우 skew 값은 nan으로 나옴

```
import numpy as np

def statistical_analysis(data):
  mean = data.mean().round(3)
  std = data.std().round(3)
  skew = (3*(mean - np.median(data))/data.std()).round(3)
  return mean, std, skew
```

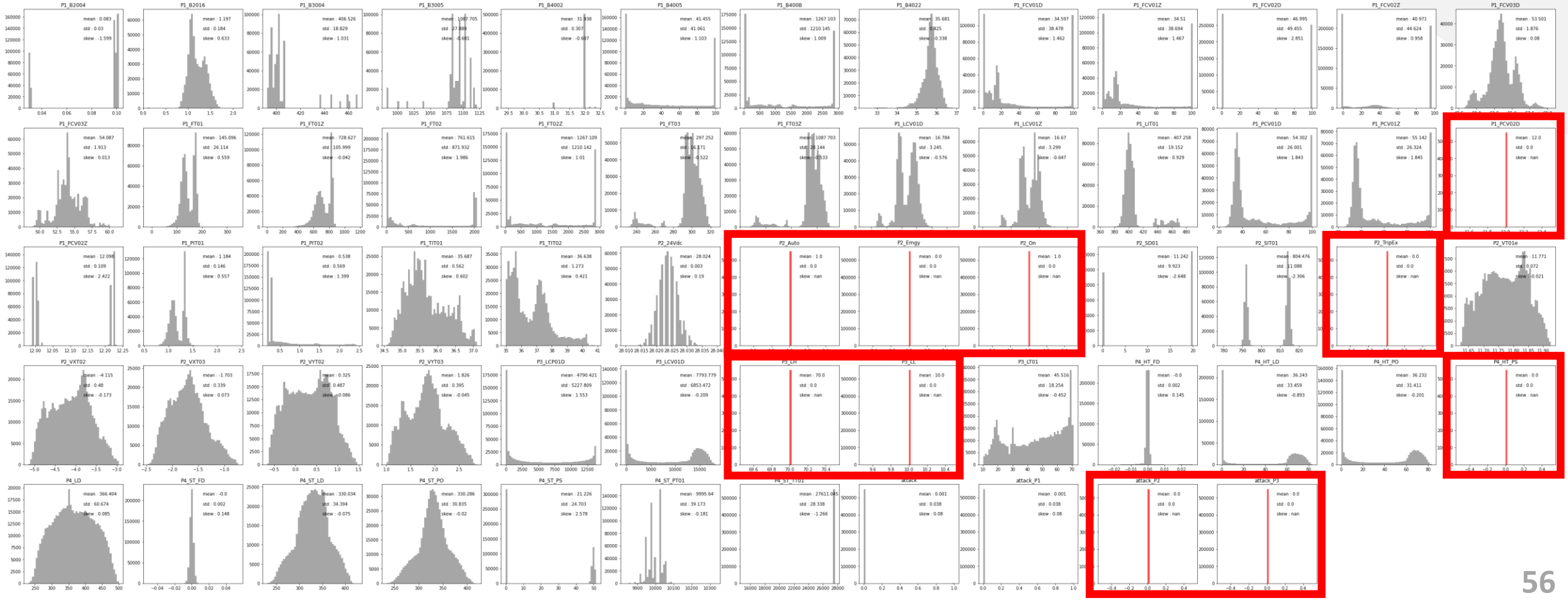
Advanced Code 1

- Train 데이터에서 각 column 별 데이터 분포 확인하기
  - ➢데이터 분포도 시각화
    - skew 값이 nan으로 나올 경우 그래프를 빨간색으로 그리게끔 설정

```
fig = plt.figure(figsize = (65, 25))
for num in range(1,64):
    ax = plt.subplot(5, 13, num)
    current_column_data = total_train_data.iloc[:,num].values
    mean, std, skew = statistical_analysis(current_column_data)
    if np.isnan(skew):
        plt.hist(current_column_data, alpha = 0.7, bins = 50, color = 'red')
    else:
        plt.hist(current_column_data, alpha = 0.7, bins = 50, color = 'gray')
    plt.title(total_train_data.columns[num])
    plt.text(0.6, 0.9, f'mean : {mean}',  ha='left', va='center', transform=ax.transAxes)
    plt.text(0.6, 0.8, f'std : {std}',  ha='left', va='center', transform=ax.transAxes)
    plt.text(0.6, 0.7, f'skew : {skew}',  ha='left', va='center', transform=ax.transAxes)
```

Advanced Code 1

- Train 데이터에서 각 column 별 데이터 분포 확인하기
  - ➢데이터 분포도 시각화
    - skew 값이 nan으로 나올 경우 그래프를 빨간색으로 그리게끔 설정



Advanced Code 1

- 특정 column 제거하기
  - ➢데이터가 동일한 값으로만 이루어진 column 리스트 도출
    - value_counts() 함수를 통하여 데이터 분포를 구함
    - len() 함수를 통하여 데이터가 1개의 값으로만 이루어지는 지를 검사
    - append() 함수를 통해 제거할 column명을 리스트에 추가

```
column_list = total_train_data.columns.to_list()
useless_column_list = []
for column in column_list:
  if len(total_train_data[column].value_counts()) == 1:
    useless_column_list.append(column)
```

```
print(useless_column_list)
```

```
['P1_PCV02D', 'P2_Auto', 'P2_Emgy', 'P2_On', 'P2_TripEx', 'P3_LH', 'P3_LL', 'P4_HT_PS', 'attack_P2', 'attack_P3']
```

Advanced Code 1

- 특정 column 제거하기
  - ➢시간 데이터인 time과, 라벨과 관련된 attack, attack_P1 변수, 데이터가 동일한 값으로만 이루어진 변수들을 빼고 특징 데이터 구성

```
LABEL_COLUMN = "attack"
FEATURE_COLUMN = total_train_data.columns.drop(["time", "attack", "attack_P1"]+ useless_column_list)
```

```
train_x = total_train_data.loc[:,FEATURE_COLUMN]
train_y = total_train_data.loc[:,[LABEL_COLUMN]]
print(train_x)
print(train_y)
```

```
        P1_B2004  P1_B2016  P1_B3004  ...  P4_ST_PS  P4_ST_PT01  P4_ST_TT01
0         0.0983    1.0702   399.2321  ...   50.9871      9973.0     27629.0
1         0.0983    1.0699   399.2321  ...   50.9871      9973.0     27629.0
2         0.0983    1.0703   399.2321  ...   50.9871      9973.0     27629.0
3         0.0983    1.0719   399.2321  ...   50.9871      9973.0     27629.0
4         0.0983    1.0710   399.2321  ...   50.9871      9973.0     27629.0
...          ...       ...        ...  ...       ...         ...        ...
550795    0.1001    1.5194   395.3508  ...    0.0000     10026.0     27548.0
550796    0.1001    1.5220   395.3508  ...    0.0000     10026.0     27549.0
550797    0.1001    1.5235   395.3508  ...    0.0000     10026.0     27547.0
550798    0.1001    1.5279   395.3508  ...    0.0000     10026.0     27551.0
550799    0.1001    1.5281   395.3508  ...    0.0000     10026.0     27552.0

[550800 rows x 51 columns]
```

```
        attack
0            0
1            0
2            0
3            0
4            0
...        ...
550795       0
550796       0
550797       0
550798       0
550799       0

[550800 rows x 1 columns]
```

Advanced Code 1

- 특정 column 제거하기
  - ➢Test 데이터도 train 데이터와 동일한 과정을 거쳐 특징 데이터와 라벨 데이터로 분리

```
test_x = total_test_data.loc[:,FEATURE_COLUMN]
test_y = total_test_data.loc[:,[LABEL_COLUMN]]
print(test_x)
print(test_y)
```

```
        P1_B2004  P1_B2016  P1_B3004  ...  P4_ST_PS  P4_ST_PT01  P4_ST_TT01
0         0.0982    1.4610  461.9883  ...       0.0     10053.0     27629.0
1         0.0982    1.4578  461.9883  ...       0.0     10053.0     27629.0
2         0.0982    1.4666  461.9883  ...       0.0     10053.0     27629.0
3         0.0982    1.4688  461.9883  ...       0.0     10053.0     27629.0
4         0.0982    1.4717  461.9883  ...       0.0     10053.0     27629.0
...          ...       ...       ...  ...       ...         ...         ...
444595    0.0304    1.2162  399.8923  ...       0.0     10013.0     27622.0
444596    0.0304    1.2118  399.8923  ...       0.0     10012.0     27626.0
444597    0.0304    1.2170  399.8923  ...       0.0     10012.0     27625.0
444598    0.0304    1.2193  399.8923  ...       0.0     10012.0     27625.0
444599    0.0304    1.2249  399.8923  ...       0.0     10013.0     27623.0

[444600 rows x 51 columns]
```

```
        attack
0            0
1            0
2            0
3            0
4            0
...        ...
444595       0
444596       0
444597       0
444598       0
444599       0

[444600 rows x 1 columns]
```

Advanced Code 1

- 데이터 정규화
  - ➢데이터에 Min-Max Normalization을 적용하여 0~1사이의 값으로 변환
  - ➢데이터를 정규화 할 수 있는 사용자 함수 선언
    - min() 함수를 통하여 각 column 데이터에서의 최소값 도출
    - max() 함수를 통하여 각 column 데이터에서의 최대값 도출
    - Train 데이터에서 최소 최대값을 도출한 후 train, test 데이터셋에 적용

```
def min_max_normalization(train_data, test_data):
  columns_min = train_data.min()
  columns_max = train_data.max()

  scaled_train_data = train_data.copy()
  for c in train_data.columns:
    scaled_train_data[c] = (train_data[c] - columns_min[c]) / (columns_max[c] - columns_min[c])

  scaled_test_data = test_data.copy()
  for c in test_data.columns:
    scaled_test_data[c] = (test_data[c] - columns_min[c]) / (columns_max[c] - columns_min[c])

  return scaled_train_data, scaled_test_data
```

Advanced Code 1

- 데이터 정규화
  - ➢데이터에 Min-Max Normalization을 적용하여 0~1 사이의 값으로 변환
  - ➢데이터를 정규화 할 수 있는 사용자 함수 선언
    - min() 함수를 통하여 각 column 데이터에서의 최소값 도출
    - max() 함수를 통하여 각 column 데이터에서의 최대값 도출

```
scaled_train_x, scaled_test_x = min_max_normalization(train_x, test_x)
print(scaled_train_x)
print(scaled_test_x)
```

```
        P1_B2004  P1_B2016  P1_B3004  ...  P4_ST_PS  P4_ST_PT01  P4_ST_TT01
0       0.950413  0.482115  0.092349  ...  0.990676     0.28169    0.999920
1       0.950413  0.481966  0.092349  ...  0.990676     0.28169    0.999920
2       0.950413  0.482164  0.092349  ...  0.990676     0.28169    0.999920
3       0.950413  0.482960  0.092349  ...  0.990676     0.28169    0.999920
4       0.950413  0.482512  0.092349  ...  0.990676     0.28169    0.999920
...          ...       ...       ...  ...       ...         ...         ...
550795  0.975207  0.705374  0.040591  ...  0.000000     0.38833    0.993401
550796  0.975207  0.706666  0.040591  ...  0.000000     0.38833    0.993482
550797  0.975207  0.707411  0.040591  ...  0.000000     0.38833    0.993321
550798  0.975207  0.709598  0.040591  ...  0.000000     0.38833    0.993643
550799  0.975207  0.709698  0.040591  ...  0.000000     0.38833    0.993723

[550800 rows x 51 columns]
```

```
        P1_B2004  P1_B2016  P1_B3004  ...  P4_ST_PS  P4_ST_PT01  P4_ST_TT01
0       0.949036  0.676348  0.929214  ...       0.0    0.442656    0.999920
1       0.949036  0.674758  0.929214  ...       0.0    0.442656    0.999920
2       0.949036  0.679131  0.929214  ...       0.0    0.442656    0.999920
3       0.949036  0.680225  0.929214  ...       0.0    0.442656    0.999920
4       0.949036  0.681666  0.929214  ...       0.0    0.442656    0.999920
...          ...       ...       ...  ...       ...         ...         ...
444595  0.015152  0.554679  0.101153  ...       0.0    0.362173    0.999356
444596  0.015152  0.552492  0.101153  ...       0.0    0.360161    0.999678
444597  0.015152  0.555076  0.101153  ...       0.0    0.360161    0.999598
444598  0.015152  0.556220  0.101153  ...       0.0    0.360161    0.999598
444599  0.015152  0.559003  0.101153  ...       0.0    0.362173    0.999437

[444600 rows x 51 columns]
```

Advanced Code 1

- Train 데이터를 사용하여 Random Forest 분류 모델 학습
  - ➢Scikit-learn에서 제공하는 Random Forest 모델을 사용

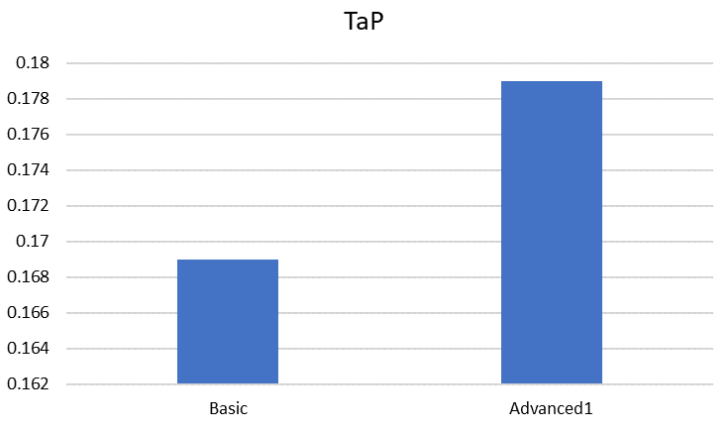
```
RF_Model = RandomForestClassifier(n_estimators=1024)
RF_Model.fit(scaled_train_x, train_y)
rf_result = RF_Model.predict(scaled_test_x)
```

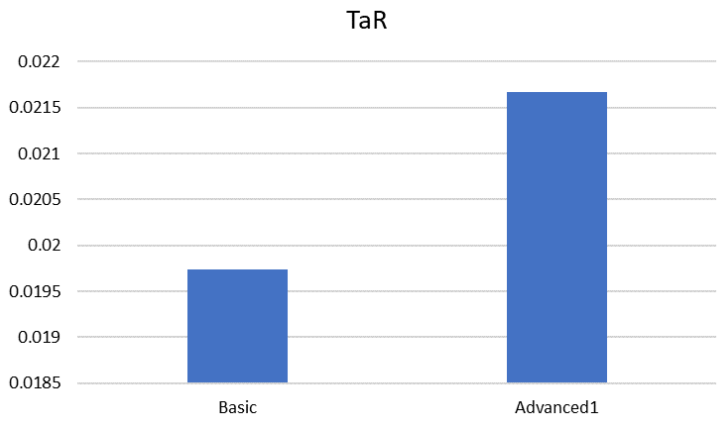
Advanced Code 1

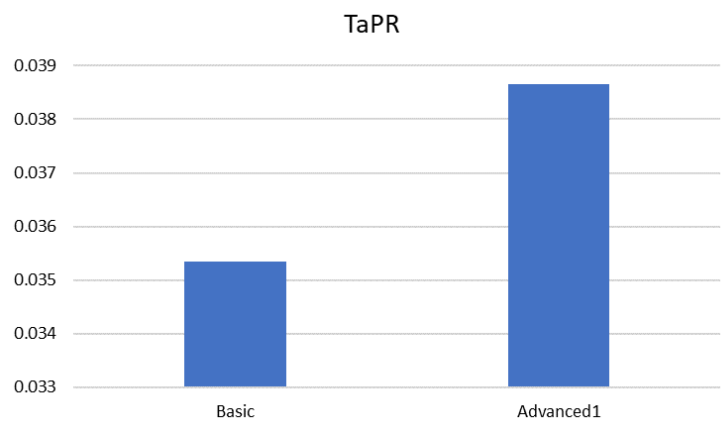
- eTaPR을 사용하여 예측 성능 평가
  - ➢라이브러리를 사용하여 성능을 출력

```
TaPR = etapr.evaluate_haicon(anomalies=test_y, predictions=rf_result)
print(f"F1: {TaPR['f1']:.3f} (TaP: {TaPR['TaP']:.3f}, TaR: {TaPR['TaR']:.3f})")
print(f"# of detected anomalies: {len(TaPR['Detected_Anomalies'])}")
print(f"Detected anomalies: {TaPR['Detected_Anomalies']}")
```

```
F1: 0.039 (TaP: 0.179, TaR: 0.022)
# of detected anomalies: 2
Detected anomalies: [<TaPR_pkg.DataManage.Range.Range object at 0x00000214A1DE0348>, <TaPR_pkg.DataManage
 .Range.Range object at 0x00000214A4AB0588>]
```







Advanced Code 2

- Advanced Code1의 불필요한 column 삭제 이후부터 이어서 시작
- 데이터 정규화
  - ➢데이터에 Standardization을 적용하여 데이터를 정규분포로 변환
  - ➢데이터를 정규화 할 수 있는 사용자 함수 선언
    - mean() 함수를 통하여 각 column 데이터에서의 평균값 도출
    - std() 함수를 통하여 각 column 데이터에서의 표준편차값 도출
    - Train 데이터에서 평균 표준편차값을 도출한 후 train, test 데이터셋에 적용

```
def standardization(train_data, test_data):
  columns_mean = train_data.mean()
  columns_std = train_data.std()

  scaled_train_data = train_data.copy()
  for c in train_data.columns:
    scaled_train_data[c] = (train_data[c] - columns_mean[c]) / columns_std[c]

  scaled_test_data = test_data.copy()
  for c in test_data.columns:
    scaled_test_data[c] = (test_data[c] - columns_mean[c]) / columns_std[c]

  return scaled_train_data, scaled_test_data
```

Advanced Code 2

- 데이터 정규화
  - ➢데이터에 Standardization을 적용하여 데이터를 정규분포로 변환
  - ➢데이터를 정규화 할 수 있는 사용자 함수 선언
    - mean() 함수를 통하여 각 column 데이터에서의 평균값 도출
    - std() 함수를 통하여 각 column 데이터에서의 표준편차값 도출

```
scaled_train_x, scaled_test_x = standardization(train_x, test_x)
print(scaled_train_x)
print(scaled_test_x)
```

```
        P1_B2004  P1_B2016  P1_B3004  ...  P4_ST_PS  P4_ST_PT01  P4_ST_TT01
0       0.516495 -0.689052 -0.387361  ...  1.204747   -0.577959    0.633590
1       0.516495 -0.690686 -0.387361  ...  1.204747   -0.577959    0.633590
2       0.516495 -0.688507 -0.387361  ...  1.204747   -0.577959    0.633590
3       0.516495 -0.679788 -0.387361  ...  1.204747   -0.577959    0.633590
4       0.516495 -0.684692 -0.387361  ...  1.204747   -0.577959    0.633590
...          ...       ...       ...  ...       ...         ...         ...
550795  0.576836  1.758602 -0.593499  ... -0.859228    0.775029   -2.224768
550796  0.576836  1.772769 -0.593499  ... -0.859228    0.775029   -2.189480
550797  0.576836  1.780942 -0.593499  ... -0.859228    0.775029   -2.260057
550798  0.576836  1.804917 -0.593499  ... -0.859228    0.775029   -2.118903
550799  0.576836  1.806007 -0.593499  ... -0.859228    0.775029   -2.083615

[550800 rows x 51 columns]
```

```
        P1_B2004  P1_B2016  P1_B3004  ...  P4_ST_PS  P4_ST_PT01  P4_ST_TT01
0       0.513142  1.440385  2.945650  ... -0.859228    1.464287    0.633590
1       0.513142  1.422948  2.945650  ... -0.859228    1.464287    0.633590
2       0.513142  1.470899  2.945650  ... -0.859228    1.464287    0.633590
3       0.513142  1.482886  2.945650  ... -0.859228    1.464287    0.633590
4       0.513142  1.498688  2.945650  ... -0.859228    1.464287    0.633590
...          ...       ...       ...  ...       ...         ...         ...
444595 -1.759714  0.106490 -0.352298  ... -0.859228    0.443164    0.386571
444596 -1.759714  0.082515 -0.352298  ... -0.859228    0.417636    0.527724
444597 -1.759714  0.110849 -0.352298  ... -0.859228    0.417636    0.492436
444598 -1.759714  0.123382 -0.352298  ... -0.859228    0.417636    0.492436
444599 -1.759714  0.153896 -0.352298  ... -0.859228    0.443164    0.421859

[444600 rows x 51 columns]
```

Advanced Code 2

- PCA를 통한 차원 축소
  - ➢Scikit-learn에서 제공하는 PCA 모델을 사용

```
from sklearn.decomposition import PCA

pca = PCA(random_state=13)
pca.fit(scaled_train_x)
print(pd.Series(np.cumsum(pca.explained_variance_ratio_)))
```

```
0     0.247863
1     0.400593
2     0.524072
3     0.624317
4     0.670581
5     0.710993
6     0.749853
7     0.782713
8     0.812317
9     0.838777
10    0.861180
11    0.882331
12    0.900767
13    0.918662
14    0.933403
15    0.943363
16    0.951914
17    0.959863
18    0.967299
19    0.972274
20    0.976948
21    0.981324
22    0.984598
23    0.987527
24    0.989723
25    0.991293
26    0.992691
27    0.993862
28    0.994902
29    0.995932
30    0.996628
31    0.997229
32    0.997799
33    0.998263
34    0.998676
35    0.999052
36    0.999291
37    0.999467
38    0.999581
39    0.999683
40    0.999779
41    0.999849
42    0.999916
43    0.999956
44    0.999967
45    0.999976
46    0.999983
47    0.999989
48    0.999995
49    0.999998
50    1.000000
dtype: float64
```

Advanced Code 2

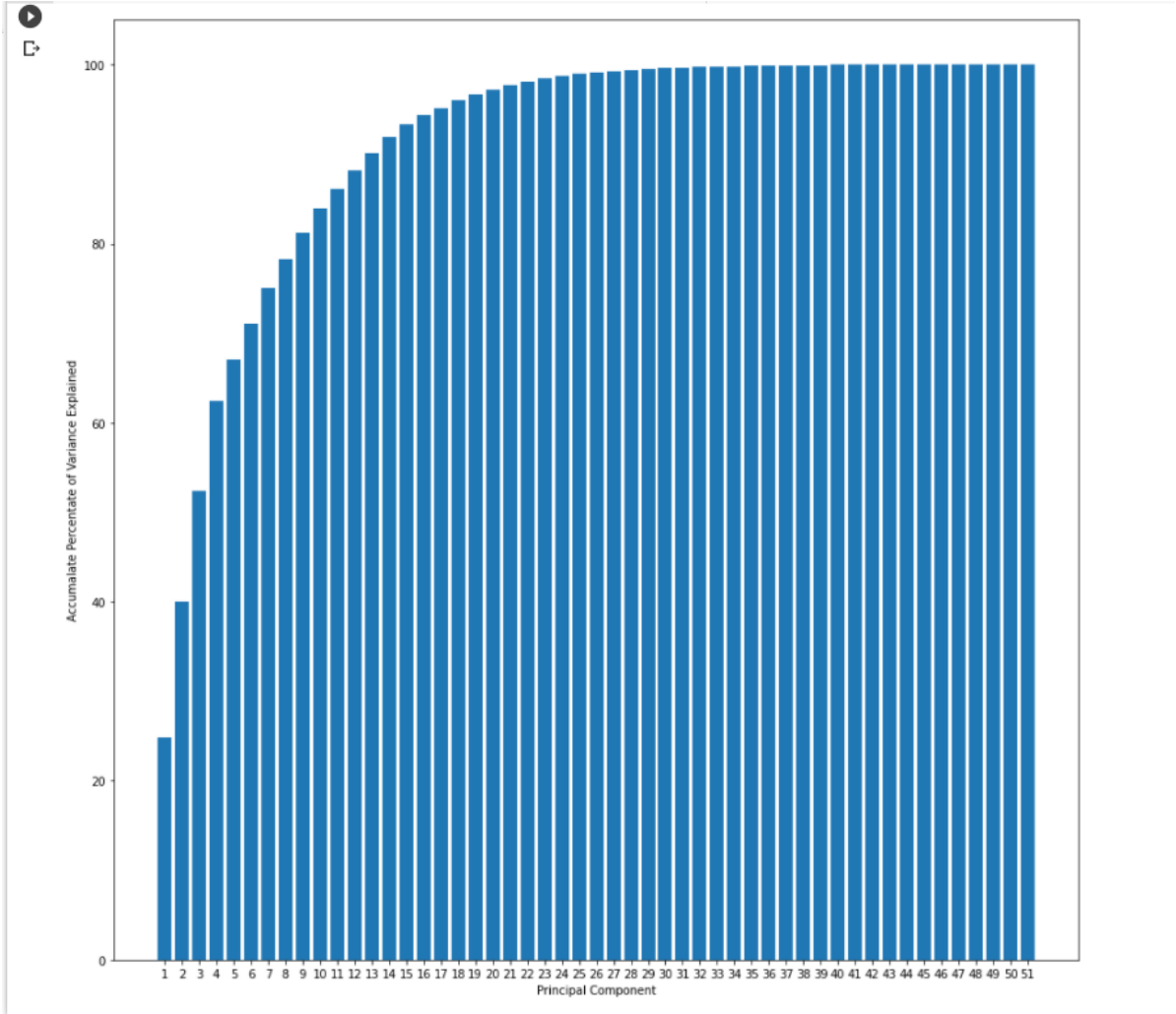
- PCA를 통한 차원 축소
  - ➢최적의 차원수를 찾기 위하여 시각화 1

```
percent_variance = np.round(np.cumsum(pca.explained_variance_ratio_) * 100, decimals=2)
columns = []
for i in range(len(percent_variance)):
    columns.append(f'{i + 1}')

fig = plt.figure(figsize = (15, 15))
ax = plt.bar(x=range(len(percent_variance)), height=percent_variance, tick_label=columns)
plt.ylabel('Accumalate Percentate of Variance Explained')
plt.xlabel('Principal Component')
plt.show()
```

Advanced Code 2

- PCA를 통한 차원 축소
  - ➢최적의 차원수를 찾기 위하여 시각화 1



Advanced Code 2

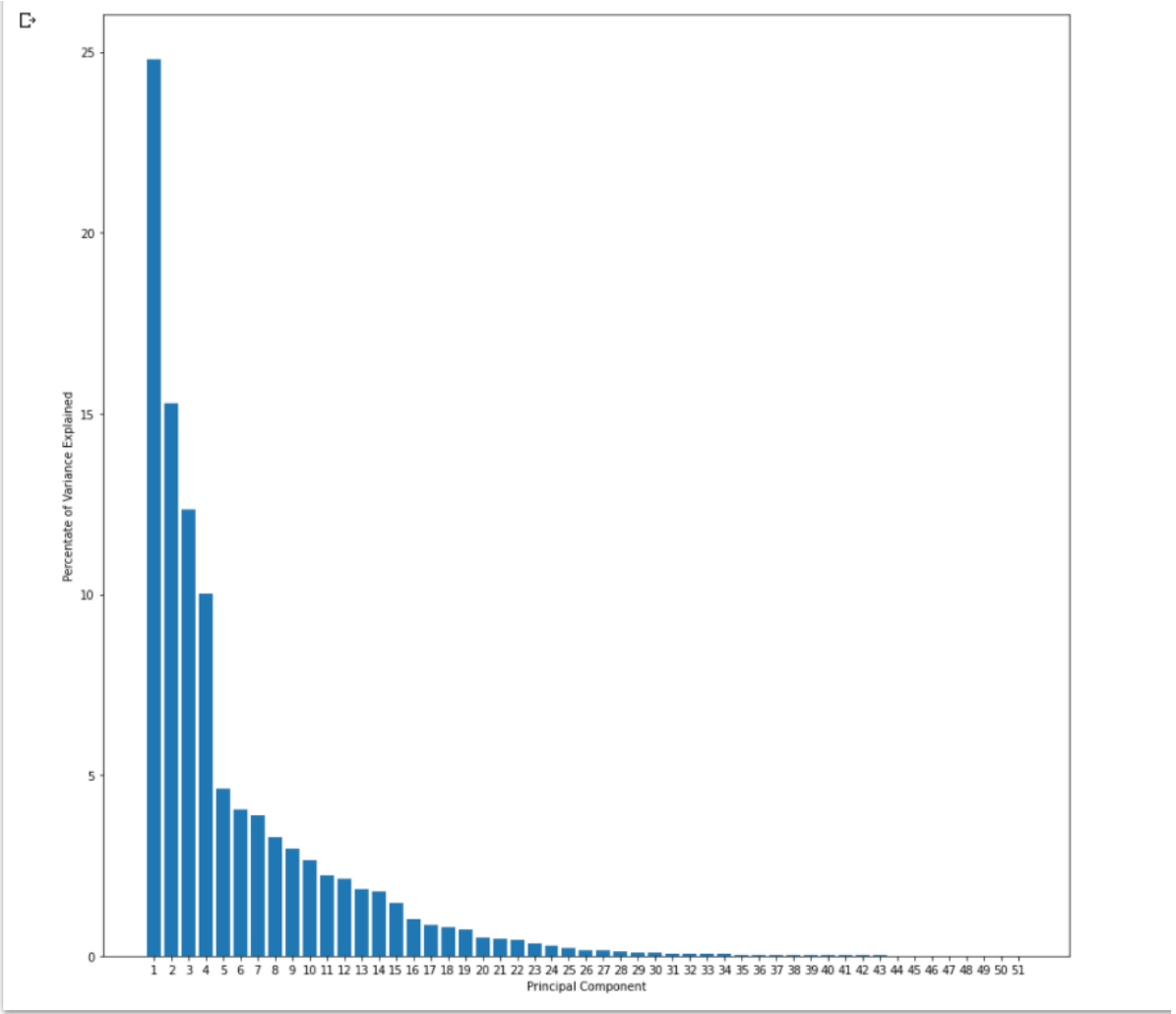
- PCA를 통한 차원 축소
  - ➢최적의 차원수를 찾기 위하여 시각화 2

```
percent_variance = np.round(pca.explained_variance_ratio_ * 100, decimals=2)
columns = []
for i in range(len(percent_variance)):
    columns.append(f'{i + 1}')

fig = plt.figure(figsize = (15, 15))
ax = plt.bar(x=range(len(percent_variance)), height=percent_variance, tick_label=columns)
plt.ylabel('Percentate of Variance Explained')
plt.xlabel('Principal Component')
plt.show()
```

Advanced Code 2

- PCA를 통한 차원 축소
  - ➢최적의 차원수를 찾기 위하여 시각화 2



Advanced Code 2

- PCA를 통한 차원 축소
  - ➢최적의 차원수를 도출 후 train 데이터와 test 데이터를 차원 축소

```
pca = PCA(n_components=16, random_state=13)
pca.fit(scaled_train_x)
scaled_train_x_p = pca.transform(scaled_train_x)
scaled_test_x_p = pca.transform(scaled_test_x)
```

Advanced Code 2

- Train 데이터를 사용하여 Random Forest 분류 모델 학습
  - ➢Scikit-learn에서 제공하는 Random Forest 모델을 사용

```
RF_Model = RandomForestClassifier(n_estimators=1024)
RF_Model.fit(scaled_train_x_p, train_y)
rf_result = RF_Model.predict(scaled_test_x_p)
```

Advanced Code 2

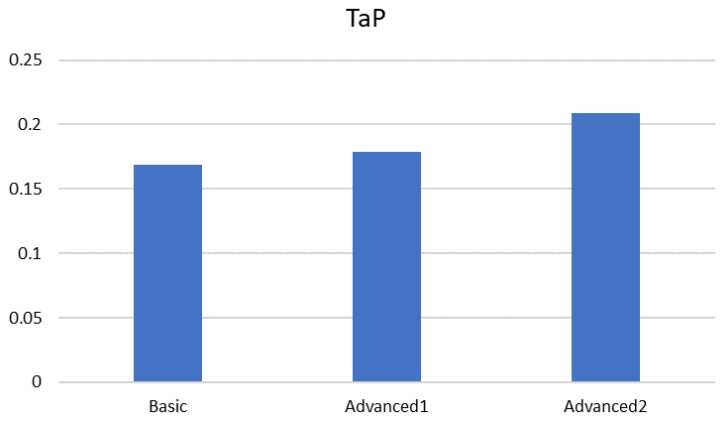
- eTaPR을 사용하여 예측 성능 평가
  - ➢라이브러리를 사용하여 성능을 출력

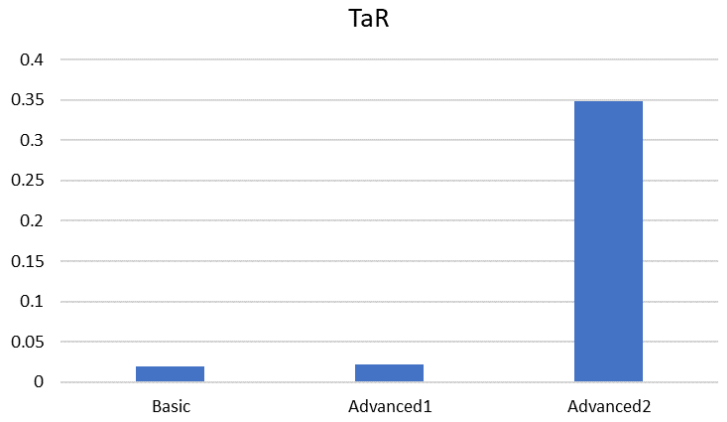
```
TaPR = etapr.evaluate_haicon(anomalies=test_y, predictions=rf_result)
print(f"F1: {TaPR['f1']:.3f} (TaP: {TaPR['TaP']:.3f}, TaR: {TaPR['TaR']:.3f})")
print(f"# of detected anomalies: {len(TaPR['Detected_Anomalies'])}")
print(f"Detected anomalies: {TaPR['Detected_Anomalies']}")
```

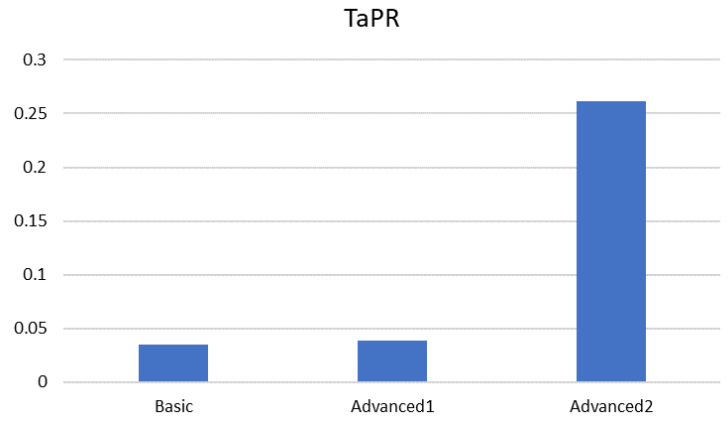
```
F1: 0.261 (TaP: 0.209, TaR: 0.348)
# of detected anomalies: 23
Detected anomalies: [<TaPR_pkg.DataManage.Range.Range object at 0x00000214A1E0FFC8>, <TaPR_pkg.DataManage
 .Range.Range object at 0x00000214A1E13808>, <TaPR_pkg.DataManage.Range.Range object at 0x00000214A1E13908>,
  <TaPR_pkg.DataManage.Range.Range object at 0x00000214A1DD8EC8>, <TaPR_pkg.DataManage.Range.Range object at
  0x00000214A44B20C8>, <TaPR_pkg.DataManage.Range.Range object at 0x00000214A44B2148>, <TaPR_pkg.DataManage
 .Range.Range object at 0x00000214A44B2448>, <TaPR_pkg.DataManage.Range.Range object at 0x00000214A44B2688>,
  <TaPR_pkg.DataManage.Range.Range object at 0x00000214A44B2708>, <TaPR_pkg.DataManage.Range.Range object at
  0x00000214A44B2808>, <TaPR_pkg.DataManage.Range.Range object at 0x00000214A44B2888>, <TaPR_pkg.DataManage
 .Range.Range object at 0x00000214A44B2988>, <TaPR_pkg.DataManage.Range.Range object at 0x00000214A44B2A08>,
  <TaPR_pkg.DataManage.Range.Range object at 0x00000214A44B2B08>, <TaPR_pkg.DataManage.Range.Range object at
  0x00000214A44B2B88>, <TaPR_pkg.DataManage.Range.Range object at 0x00000214A44B2C88>, <TaPR_pkg.DataManage
 .Range.Range object at 0x00000214A44B2D88>, <TaPR_pkg.DataManage.Range.Range object at 0x00000214A44B2E08>,
  <TaPR_pkg.DataManage.Range.Range object at 0x00000214A44B2E88>, <TaPR_pkg.DataManage.Range.Range object at
  0x00000214A44B2F08>, <TaPR_pkg.DataManage.Range.Range object at 0x00000214A44B2F88>, <TaPR_pkg.DataManage
 .Range.Range object at 0x00000214A1AAB208>, <TaPR_pkg.DataManage.Range.Range object at 0x000002149DDAEFC8>]
```

Advanced Code 2

- eTaPR을 사용하여 예측 성능 평가
  - ➢데이터와 모델에 맞는 전처리 방법을 적용하면 성능이 향상된다는 것을 확인할 수 있음



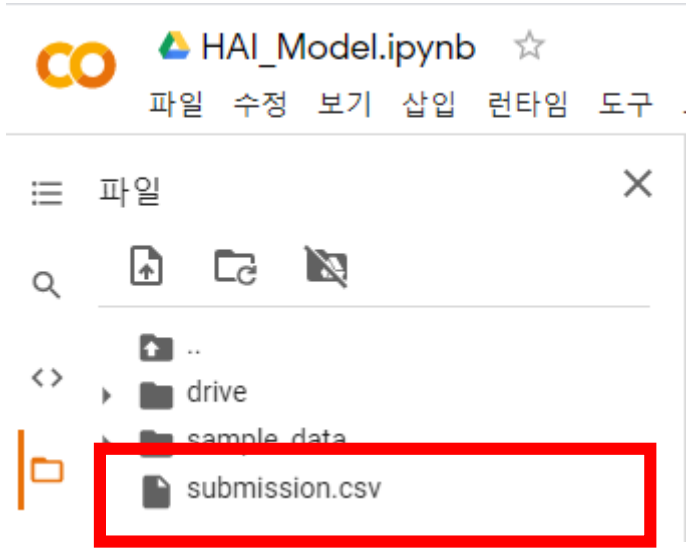




Advanced Code 2

- 제출 파일 생성
  - ➢to_csv() 함수를 통하여 데이터프레임을 파일로 저장하기

```
rf_result = pd.DataFrame(rf_result, columns=["attack"])
submission = pd.concat([total_test_data["time"], rf_result], axis=1)
submission.columns = ["timestamp", "attack"]
submission.to_csv("./submission.csv", index=False)
```

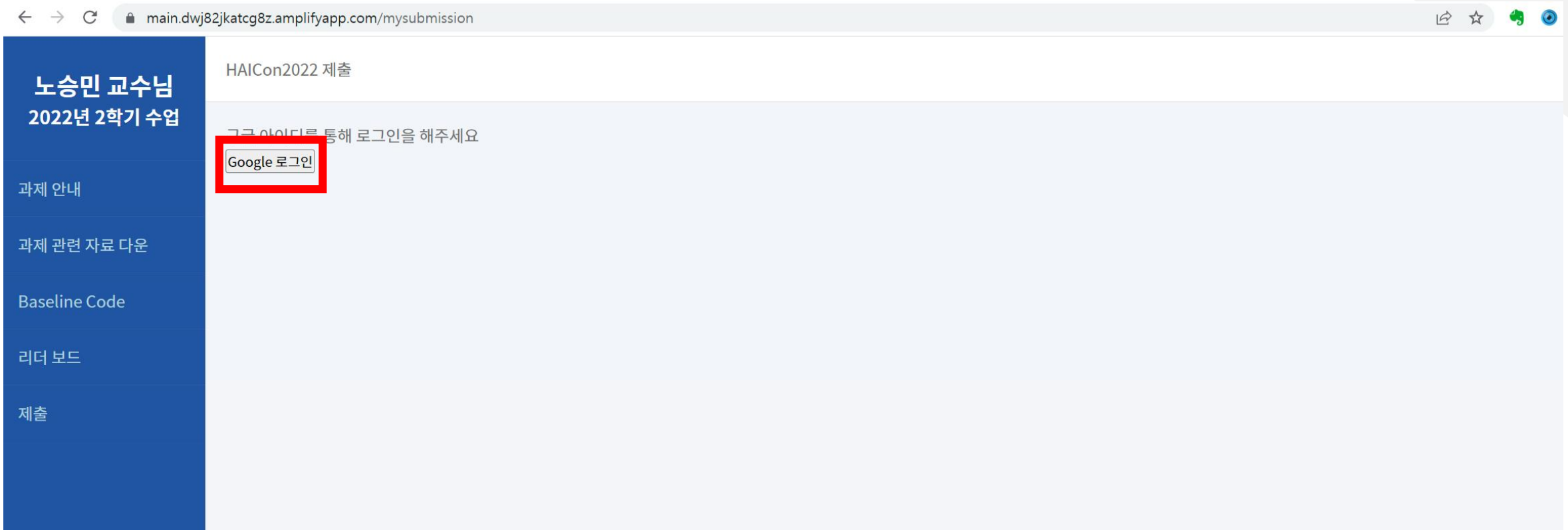


과제 제출 방법

- 과제 제출
  - ➢과제 제출용 사이트 https://main.dwj82jkatcg8z.amplifyapp.com/ 에 들어간 후 제출 탭을 클릭

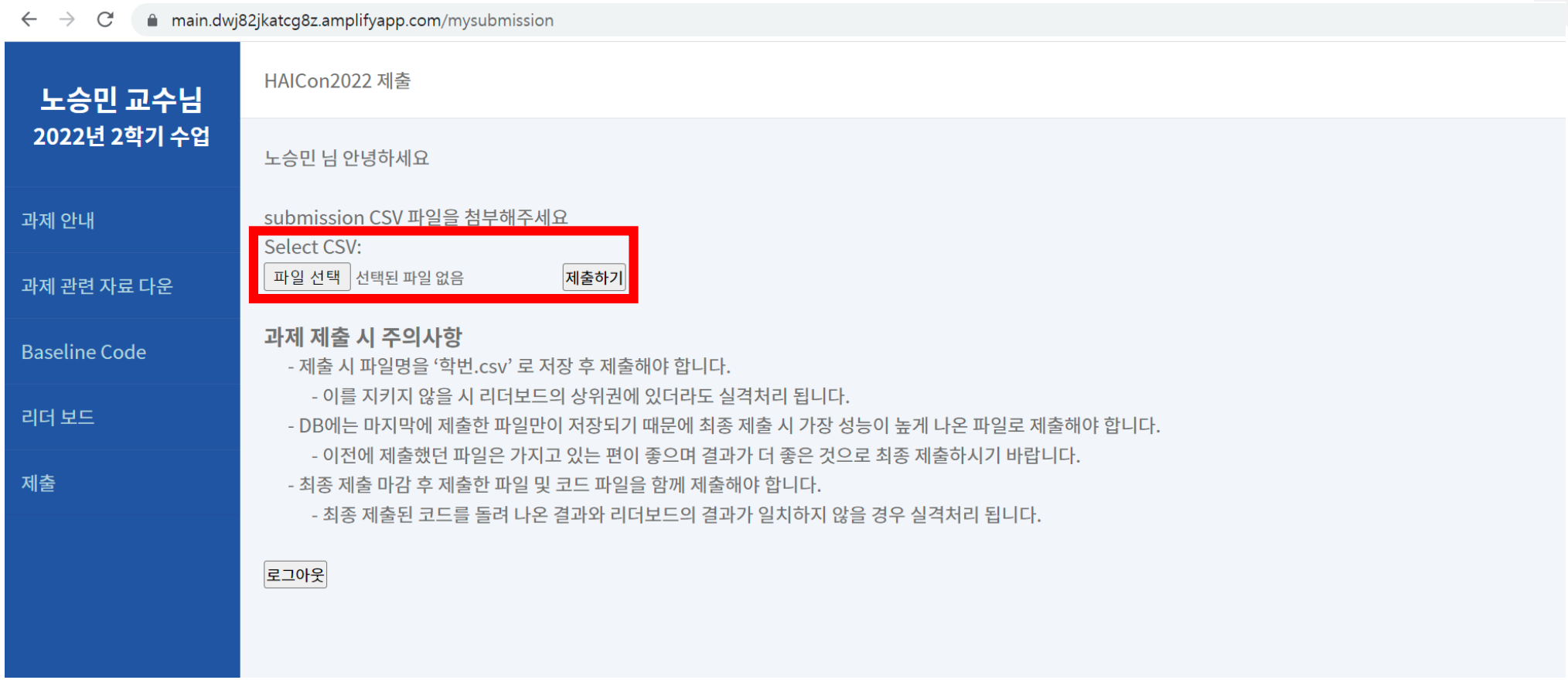
과제 제출 방법

- 과제 제출
  - ➢각자 로그인 후 제출해야 하기 때문에 Google 로그인 버튼을 클릭



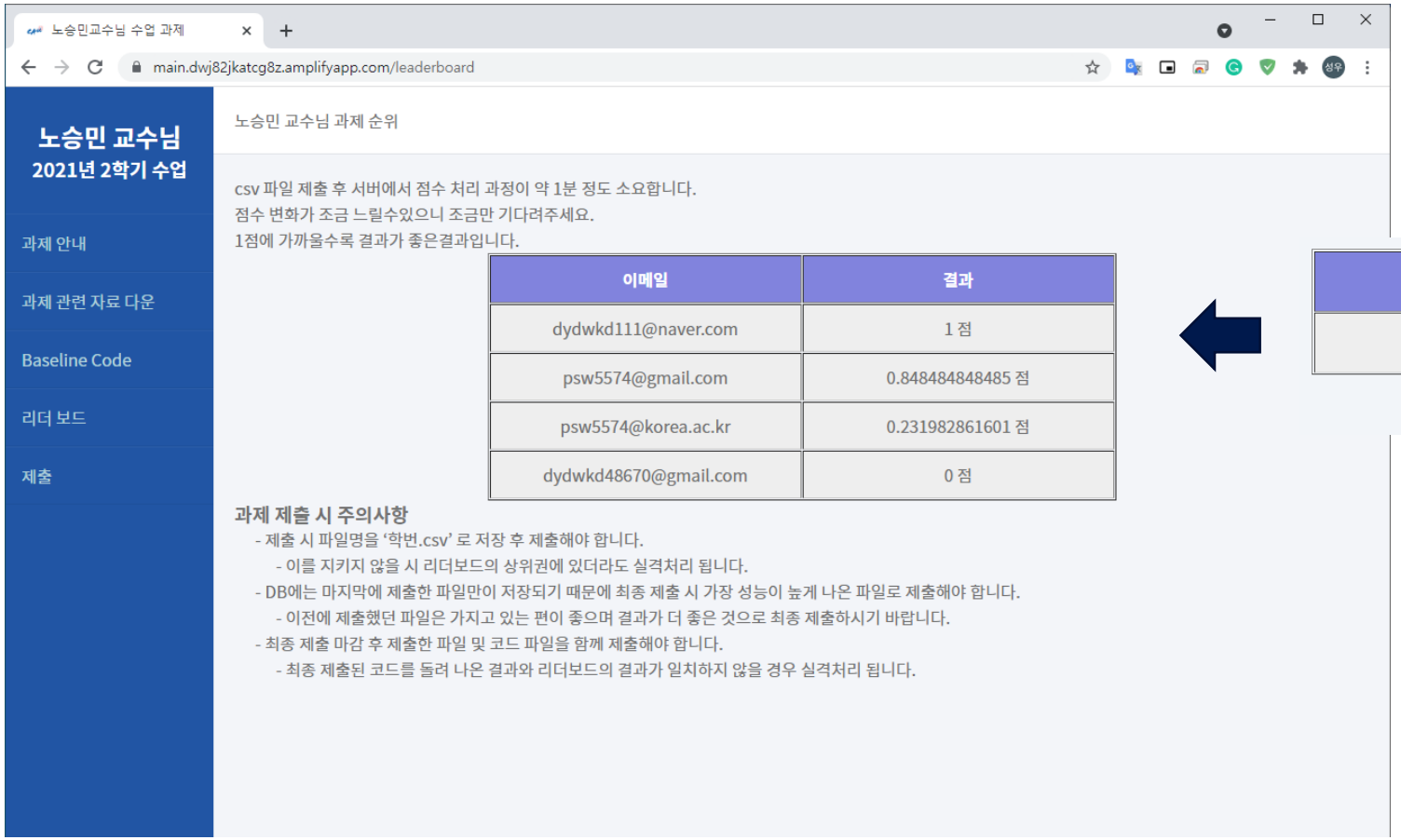
과제 제출 방법

- 과제 제출
  - ➢파일 선택 버튼을 클릭 후 결과 파일 선택
  - ➢파일 선택 후 제출하기 버튼 클릭



과제 제출 방법

- 과제 제출
  - ➢파일 제출하기 버튼 클릭 후 약 1분 대기 후 리더보드 탭에서 자신으 | 현재 순위 확인 가능



학번	결과
baseline	0.279567451403 점

과제 제출 방법

- 과제 제출 시 주의사항
  - ➢제출 시 파일명을 '학번.csv' 로 저장 후 제출해야 합니다.
    - 이를 지키지 않을 시 리더보드의 상위권에 있더라도 실격처리 됩니다.
  - ➢DB에는 마지막에 제출한 파일만이 저장되기 때문에 최종 제출 시 가장 성능이 높게 나온 파일로 제출해야 합니다.
    - 이전에 제출했던 파일은 가지고 있는 편이 좋으며 결과가 더 좋은 것으로 최종 제출하시기 바랍니다.
  - ➢최종 제출 마감 후 제출한 파일 및 코드 파일을 함께 제출해야 합니다.
    - 최종 제출된 코드를 돌려 나온 결과와 리더보드의 결과가 일치하지 않을 경우 실격처리 됩니다.