실무 데이터 분석
(군집화 분석)

인공지능 기반 스마트 설계
컴퓨터 AI공학부
천세진

DATA SCIENCE LABS

사전준비

데이터

	A	B	C	D	E	F	G
1	I_FIELD	C_ITEM	D_BUY	GEN	CPU	MEM	VGA
2	A	OPC0030182	00:00.0	I3	I3	4G	온보드
3	A	OPC0030205	00:00.0	I3	I3	4G	9400GT
4	A	OPC0030280	00:00.0	I3	I3	2G	온보드
5	A	OPC0030337	00:00.0	I3	I3	4G	온보드
6	B	OPC0030327	00:00.0	I3	I3	2G	온보드
7	B	OPC0030454	00:00.0	I3	I3	4G	내장
8	G	OPC0030179	00:00.0	I3	I3	2G	온보드
9	K	OPC0030196	00:00.0	I3	I3	2G	온보드
10	K	OPC0030384	00:00.0	I3	I3	4G	온보드
11	K	OPC0030389	00:00.0	i3	i3 2100	8G	온보드
12	K	OPC0030284	00:00.0	I3	I3 2100	4G	온보드
13	K	OPC0030143	00:00.0	I3	I3 2100	4G	GT220
14	B	OPC0030397	00:00.0	i3	i3 2100	4G	GT650
15	B	OPC0030355	00:00.0	I3	I3 2100	8G	온보드
16	B	OPC0030295	00:00.0	I3	I3 2100	4G	GT210
17	B	OPC0030305	00:00.0	i3	i3 2100	8G	온보드
18	B	OPC0030192	00:00.0	i3	i3 2100	4G	온보드
19	B	OPC0030021	00:00.0	i3	i3 2100	8G	GT710
20	B	OPC0030041	00:00.0	I3	I3 2100	4G	온보드
21	A	OPC0030279	00:00.0	i3	i3 2100	4G	내장형
22	A	OPC0030329	00:00.0	i3	i3 2100	4G	온보드
23	A	OPC0030282	00:00.0	i3	i3 2100	6G	온보드
24	A	OPC0030376	00:00.0	i3	i3 2100	4G	온보드
25	A	OPC0030236	00:00.0	i3	i3 2100	4G	9400GT

데이터 처리 단계

1. 데이터 이해
(목적, 구성, 특징)

2. 데이터 전처리
(결측값, 이상치, 중복값)

3. 데이터 탐색
(데이터의 분포, 상관관계, 이상치 탐색)

4. 통계적인 분석
(Aggregation/Summarization)

5. 시각화

6. 결론 도출

1. 데이터 이해

- 목적
  - CPU, MEM, VGA간의 관계가 있고 이들간의 관계가 있을 것 같다
  - N개의 구매 그룹으로 나누고 싶다
  - -> 클러스터링/커뮤니티 알고리즘

	A	B	C	D	E	F	G
1	I_FIELD	C_ITEM	D_BUY	GEN	CPU	MEM	VGA
2	A	OPC0030182	00:00.0	I3	I3	4G	온보드
3	A	OPC0030205	00:00.0	I3	I3	4G	9400GT
4	A	OPC0030280	00:00.0	I3	I3	2G	온보드
5	A	OPC0030337	00:00.0	I3	I3	4G	온보드
6	B	OPC0030327	00:00.0	I3	I3	2G	온보드
7	B	OPC0030454	00:00.0	I3	I3	4G	내장
8	G	OPC0030179	00:00.0	I3	I3	2G	온보드
9	K	OPC0030196	00:00.0	I3	I3	2G	온보드
10	K	OPC0030384	00:00.0	I3	I3	4G	온보드
11	K	OPC0030389	00:00.0	i3	i3 2100	8G	온보드
12	K	OPC0030284	00:00.0	I3	I3 2100	4G	온보드
13	K	OPC0030143	00:00.0	I3	I3 2100	4G	GT220
14	B	OPC0030397	00:00.0	i3	i3 2100	4G	GT650
15	B	OPC0030355	00:00.0	I3	I3 2100	8G	온보드
16	B	OPC0030295	00:00.0	I3	I3 2100	4G	GT210
17	B	OPC0030305	00:00.0	i3	i3 2100	8G	온보드
18	B	OPC0030192	00:00.0	i3	i3 2100	4G	온보드
19	B	OPC0030021	00:00.0	i3	i3 2100	8G	GT710
20	B	OPC0030041	00:00.0	I3	I3 2100	4G	온보드
21	A	OPC0030279	00:00.0	i3	i3 2100	4G	내장형
22	A	OPC0030329	00:00.0	i3	i3 2100	4G	온보드
23	A	OPC0030282	00:00.0	i3	i3 2100	6G	온보드
24	A	OPC0030376	00:00.0	i3	i3 2100	4G	온보드
25	A	OPC0030236	00:00.0	i3	i3 2100	4G	9400GT

1. 데이터 이해

- 구성
- I_FIELD: 구매 ID
- C_ITEM: 상품 ID
- D_BUY: 구매 시간
- GEN: CPU 제조사
- CPU: CPU 모델명
- MEM: 메모리 용량
- VGA: 그래픽 카드 모델명

	A	B	C	D	E	F	G
1	I_FIELD	C_ITEM	D_BUY	GEN	CPU	MEM	VGA
2	A	OPC0030182	00:00.0	I3	I3	4G	온보드
3	A	OPC0030205	00:00.0	I3	I3	4G	9400GT
4	A	OPC0030280	00:00.0	I3	I3	2G	온보드
5	A	OPC0030337	00:00.0	I3	I3	4G	온보드
6	B	OPC0030327	00:00.0	I3	I3	2G	온보드
7	B	OPC0030454	00:00.0	I3	I3	4G	내장
8	G	OPC0030179	00:00.0	I3	I3	2G	온보드
9	K	OPC0030196	00:00.0	I3	I3	2G	온보드
10	K	OPC0030384	00:00.0	I3	I3	4G	온보드
11	K	OPC0030389	00:00.0	i3	i3 2100	8G	온보드
12	K	OPC0030284	00:00.0	I3	I3 2100	4G	온보드
13	K	OPC0030143	00:00.0	I3	I3 2100	4G	GT220
14	B	OPC0030397	00:00.0	i3	i3 2100	4G	GT650
15	B	OPC0030355	00:00.0	I3	I3 2100	8G	온보드
16	B	OPC0030295	00:00.0	I3	I3 2100	4G	GT210
17	B	OPC0030305	00:00.0	i3	i3 2100	8G	온보드
18	B	OPC0030192	00:00.0	i3	i3 2100	4G	온보드
19	B	OPC0030021	00:00.0	i3	i3 2100	8G	GT710
20	B	OPC0030041	00:00.0	I3	I3 2100	4G	온보드
21	A	OPC0030279	00:00.0	i3	i3 2100	4G	내장형
22	A	OPC0030329	00:00.0	i3	i3 2100	4G	온보드
23	A	OPC0030282	00:00.0	i3	i3 2100	6G	온보드
24	A	OPC0030376	00:00.0	i3	i3 2100	4G	온보드
25	A	OPC0030236	00:00.0	i3	i3 2100	4G	9400GT

1. 데이터 이해

```
[1]  1 import pandas as pd
```

```
1 df = pd.read_excel("/content/PC 보유 현황.xlsx")
2 df.head(5)
```

	I_FIELD	C_ITEM	D_BUY	GEN	CPU	MEM	VGA
0	A	OPC0030182	2010-01-01	I3	I3	4G	온보드
1	A	OPC0030205	2010-01-01	I3	I3	4G	9400GT
2	A	OPC0030280	2010-01-01	I3	I3	2G	온보드
3	A	OPC0030337	2010-01-01	I3	I3	4G	온보드
4	B	OPC0030327	2010-01-01	I3	I3	2G	온보드

1. 데이터 이해

```
1 df.sample(10)
```

	I_FIELD	C_ITEM	D_BUY	GEN	CPU	MEM	VGA
366	A	OPC0030541	2019-08-01	I7	I7-8700	16G	GTX1650
39	A	OPC0030378	2010-01-01	I3	I3 4130	8G	온보드
176	A	OPC0030379	2014-03-12	I5	I5 4670	8G	GT650
265	A	OPC0030390	2010-01-01	I5	I5-9600K	8G	온보드
30	A	OPC0030455	2010-01-01	i3	i3 3240	8G	460 SE
369	A	OPC0030545	2019-08-12	I7	I7-8700	16G	GTX1650
274	D	OPC0030509	2010-01-01	i7	i7 7700	16G	GTX1050
48	B	OPC0030437	2010-01-01	i3	i3 4130	4G	온보드
137	A	OPC0030475	2010-01-01	I3	I3-4170	4G	내장그래픽
297	A	OPC0030597	2021-04-19	I7	I7-10700	16G	온보드

1. 데이터 이해

- 특징
  - 데이터의 수, 데이터의 유형

데이터의 특징

```
1 df.shape
```

0s

```
(388, 7)
```

```
1 df.dtypes
```

```
I_FIELD              object
C_ITEM               object
D_BUY        datetime64[ns]
GEN                  object
CPU                  object
MEM                  object
VGA                  object
dtype: object
```

1. 데이터 이해

- 특징
  - 기본적인 데이터 통계, 결측값 확인

```
1 df.describe()
```

```
<ipython-input-8-ea8415b8a3ee>:1: FutureWarning: Treating datetime data as categorical rath
  df.describe()
```

	I_FIELD	C_ITEM	D_BUY	GEN	CPU	MEM	VGA
count	388	388	388	388	388	388	364
unique	7	388	75	6	79	11	49
top	A	OPC0030182	2010-01-01 00:00:00	I7	I7-8700	8G	온보드
freq	231	1	236	108	34	129	154
first	NaN	NaN	2005-01-10 00:00:00	NaN	NaN	NaN	NaN
last	NaN	NaN	2023-09-11 00:00:00	NaN	NaN	NaN	NaN

1. 데이터 이해

- 특징
  - 고유값 확인

```
1 df['CPU'].unique()
```

```
array(['I3', 'i3 2100', 'I3 2100', 'i3 3240', 'i3 3250', 'i3 4130',
       'I3 4130', 'i3 4160', 'i3 4170', 'I3 540', 'i3 540', 'i3 7100',
       'i3 7320', 'I3 7320', 'I3 8100', 'i3 8100', 'I3-10100', 'I3-10105',
       'I3-2100', 'i3-2100', 'I3-4130', 'i3-4130', 'I3-4160', 'I3-4170',
       'I3-7320', 'I3-9100', 'I5', 'i5', 'i5 2500', 'I5 2500', 'I5 3570',
       'i5 3570', 'I5 4570', 'i5 4570', 'i5 4670', 'I5 4670', 'i5 4690',
       'I5 4690', 'i5 7400', 'I5 7400', 'i5 760', 'I5 760', 'i5 7600',
       'I5 7600', 'I5 G10', 'I5-10400', 'I5-10500', 'I5-11500',
       'I5-12400', 'I5-12500', 'i5-2500', 'I5-4590', 'I5-4670', 'i5-4670',
       'I5-4690', 'I5-760', 'I5-8500', 'I5-9400', 'I5-9600K', 'i7 2700',
       'i7 4770', 'I7 7700', 'i7 7700', 'I7 8700', 'i7 8700', 'i7 9700',
       'I7-10700', 'I7-10700F', 'i7-10700f', 'i7-10700F', 'i7-11700',
       'I7-11700', 'i7-12700', 'I7-12700', 'I7-13700', 'I7-870',
       'I7-8700', 'I7-9700', 'I7-9700F'], dtype=object)
```

1. 데이터 이해

- 특징
  - 값의 개수 확인

```
1 df['CPU'].value_counts()
```

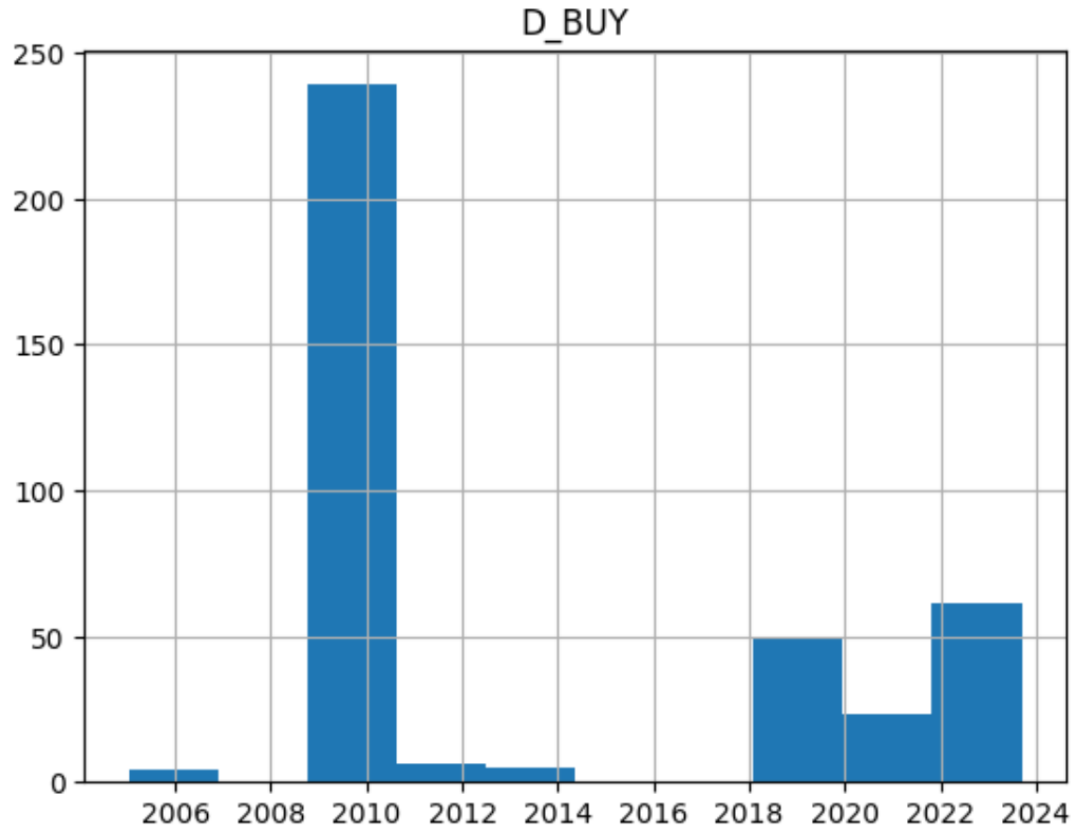
```
I7-8700      34
i5 4670      25
I7-10700     23
i3 4130      19
I3-10105     17
             ..
I5-4670       1
I5-4590       1
i5-2500       1
I5-12500      1
I3-10100      1
Name: CPU, Length: 79, dtype: int64
```

2. 결측값 이상치 확인

- 특징
  - 값의 분포

```
1 df.hist()
```

```
array([[<Axes: title={'center': 'D_BUY'}>]], dtype=object)
```



DATA SCIENCE LABS

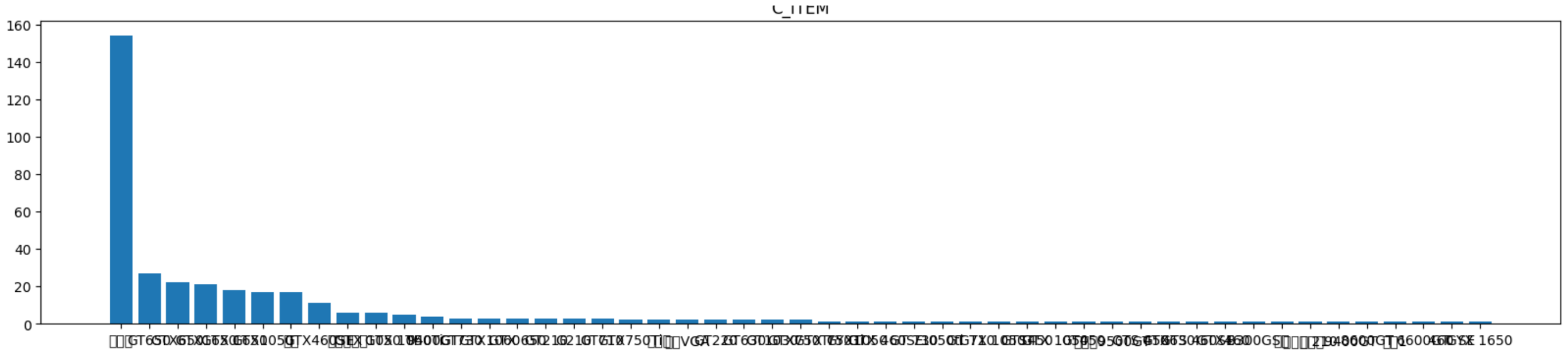
2. 결측값 이상치 확인 + 3. 데이터 탐색

- 특징
  - 값의 분포

```python
# 각 열의 고유한 값을 추출합니다.
for col in df.columns:
    unique_values = df[col].unique()

# 각 값의 개수를 계산합니다.
for col in df.columns:
    counts = df[col].value_counts()

# 값별 개수를 막대 그래프로 출력합니다.
for col in df.columns:
    plt.bar(counts.index, counts)
    plt.title(col)
    plt.rcParams["figure.figsize"] = (20, 4)
    plt.show()
```

2. 결측값 이상치 확인 + 3. 데이터 탐색

- 특징
  - 값의 분포



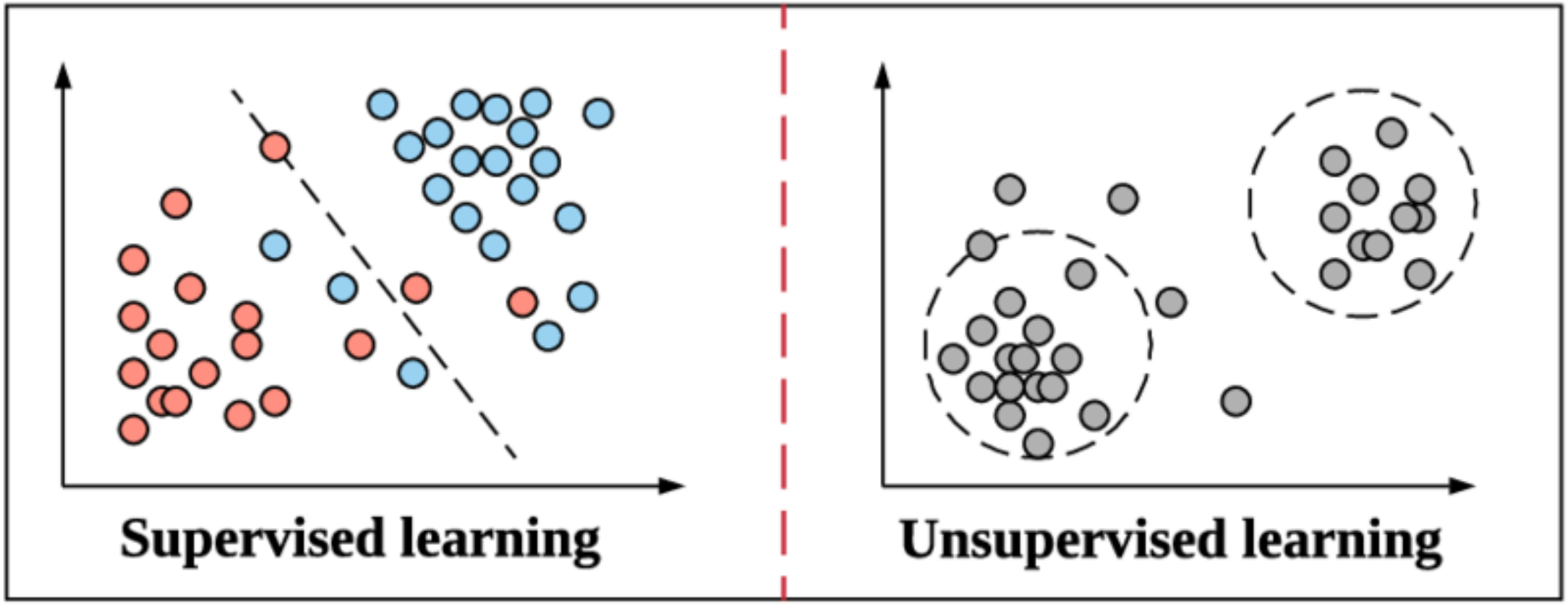
2. 결측값 이상치 확인 + 3. 데이터 탐색

- 특징
  - 값의 분포

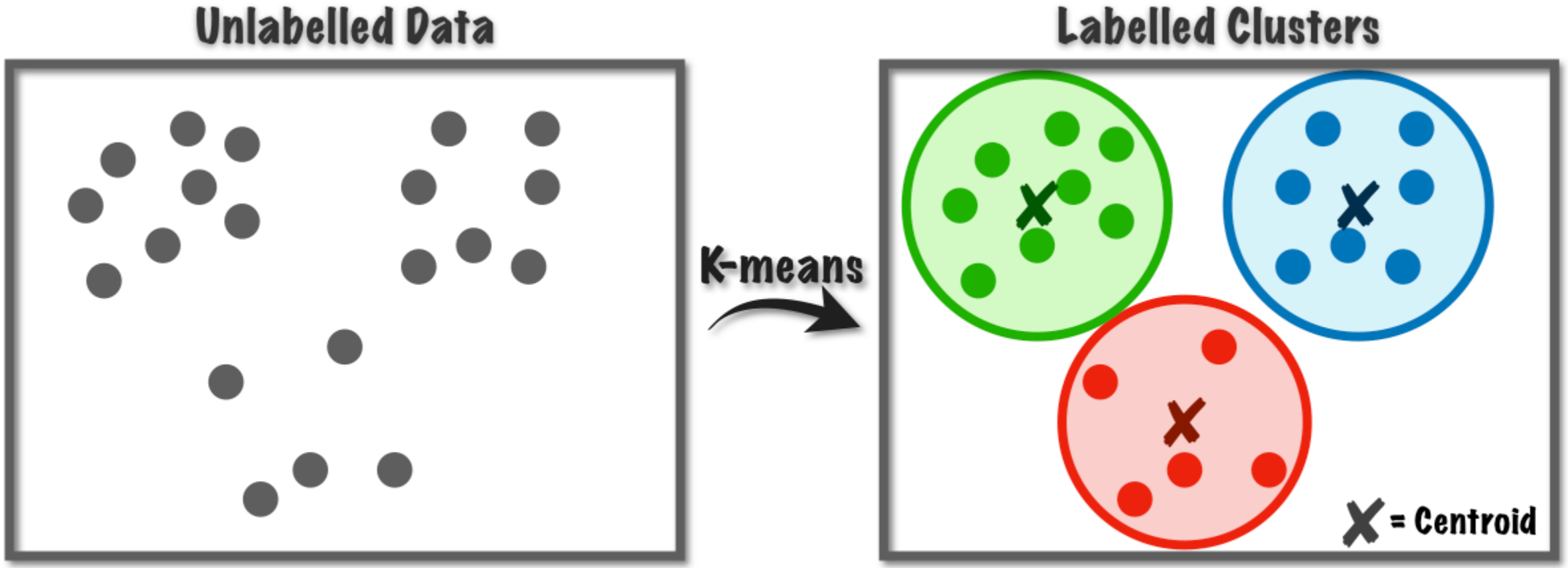
```
[27]  1 counts
```

```
온보드          154
GT650            27
GTX650           22
GTX1650          21
GTX 1650         18
GTX1050          17
내장              17
GTX460SE         11
내장그래픽          6
GTX 1050 TI       6
GTX 1050Ti        5
9400GT            4
GT730             3
GTX1060           3
GTX 650           3
GT210             3
G210              3
GT710             3
GTX750Ti          2
내장형               2
내장VGA             2
GT220             2
GT630             2
GT1030            2
GTX750TI          2
```

4. 클러스터 알고리즘



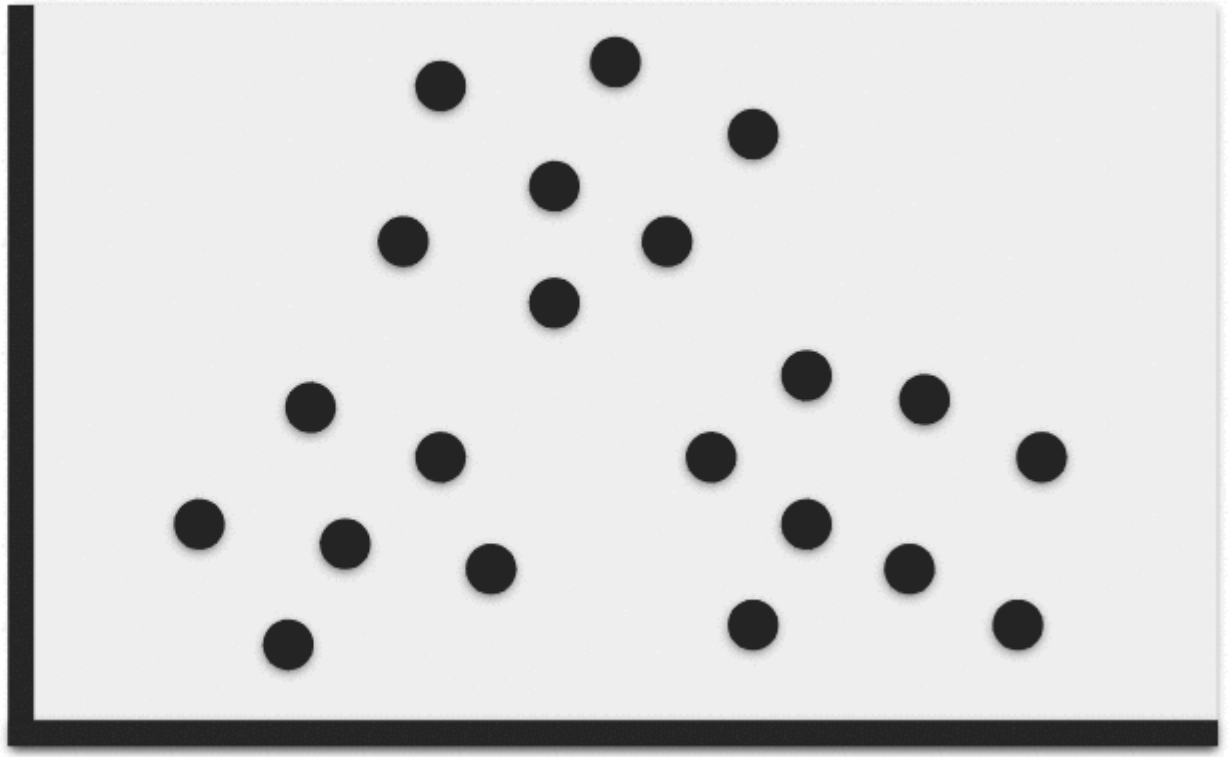
4. 클러스터 알고리즘: K-MEAN



https://towardsdatascience.com/k-means-a-complete-introduction-1702af9cd8c

4. 클러스터 알고리즘: K-MEAN

1. Initialise random centroids
2. Until convergence:
   - Assign step
   - Update step
3. End

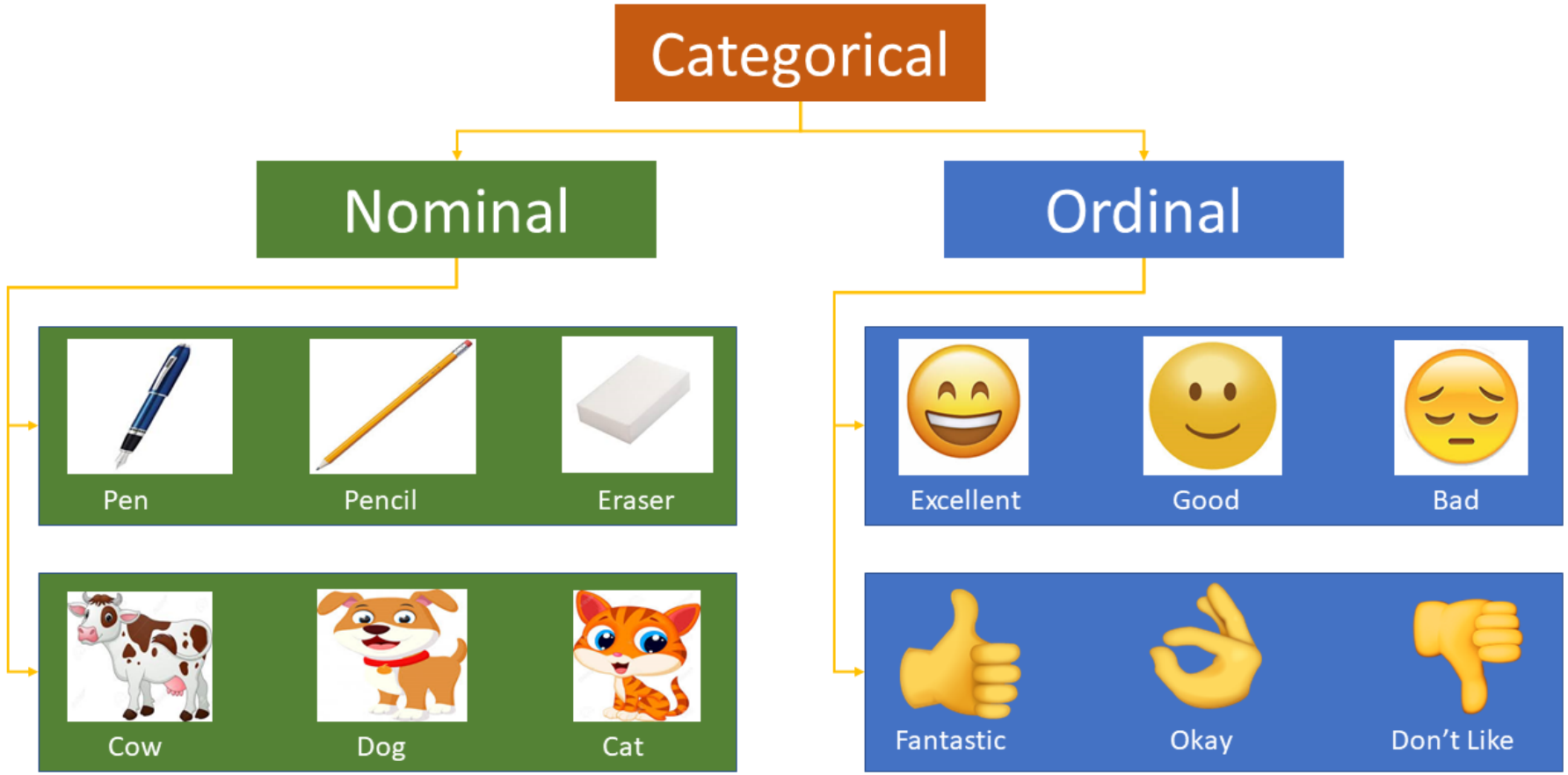


카테고리 값 간 비교

- I3 vs I3 2100
- 4G vs 8G
- 예로,
  - A 구역과 B 구역간 차이

	A	B	C	D	E	F	G
1	I_FIELD	C_ITEM	D_BUY	GEN	CPU	MEM	VGA
2	A	OPC0030182	00:00.0	I3	I3	4G	온보드
3	A	OPC0030205	00:00.0	I3	I3	4G	9400GT
4	A	OPC0030280	00:00.0	I3	I3	2G	온보드
5	A	OPC0030337	00:00.0	I3	I3	4G	온보드
6	B	OPC0030327	00:00.0	I3	I3	2G	온보드
7	B	OPC0030454	00:00.0	I3	I3	4G	내장
8	G	OPC0030179	00:00.0	I3	I3	2G	온보드
9	K	OPC0030196	00:00.0	I3	I3	2G	온보드
10	K	OPC0030384	00:00.0	I3	I3	4G	온보드
11	K	OPC0030389	00:00.0	i3	i3 2100	8G	온보드
12	K	OPC0030284	00:00.0	I3	I3 2100	4G	온보드
13	K	OPC0030143	00:00.0	I3	I3 2100	4G	GT220
14	B	OPC0030397	00:00.0	i3	i3 2100	4G	GT650
15	B	OPC0030355	00:00.0	I3	I3 2100	8G	온보드
16	B	OPC0030295	00:00.0	I3	I3 2100	4G	GT210
17	B	OPC0030305	00:00.0	i3	i3 2100	8G	온보드
18	B	OPC0030192	00:00.0	i3	i3 2100	4G	온보드
19	B	OPC0030021	00:00.0	i3	i3 2100	8G	GT710
20	B	OPC0030041	00:00.0	I3	I3 2100	4G	온보드
21	A	OPC0030279	00:00.0	i3	i3 2100	4G	내장형
22	A	OPC0030329	00:00.0	i3	i3 2100	4G	온보드
23	A	OPC0030282	00:00.0	i3	i3 2100	6G	온보드
24	A	OPC0030376	00:00.0	i3	i3 2100	4G	온보드
25	A	OPC0030236	00:00.0	i3	i3 2100	4G	9400GT

카테고리 값



X 선정

- 숫자로 변경이 필요

- 4G -> 0
- 5G -> 1

```
[41]  1 data = df.drop(["D_BUY","I_FIELD","C_ITEM"], axis=1)
```

```
[43]  1 data = data[['GEN','CPU','MEM','VGA']].astype("category")
      2 data
```

	GEN	CPU	MEM	VGA
0	I3	I3	4G	온보드
1	I3	I3	4G	9400GT
2	I3	I3	2G	온보드
3	I3	I3	4G	온보드
4	I3	I3	2G	온보드
...	...	...	...	...
383	I7	I7-9700F	16G	GTX 1650
384	I7	I7-9700F	16G	GTX 1650
385	I7	I7-9700F	16G	GTX 1650
386	I7	I7-9700F	16G	GTX 1650
387	I7	I7-9700F	16G	GYX 1650

388 rows × 4 columns

X 선정

- 숫자로 변경이 필요

- 4G -> 0
- 5G -> 1

```
1 data['CPU'].cat.categories
```

```
Index(['I3', 'I3 2100', 'I3 4130', 'I3 540', 'I3 7320', 'I3 8100', 'I3-10100',
       'I3-10105', 'I3-2100', 'I3-4130', 'I3-4160', 'I3-4170', 'I3-7320',
       'I3-9100', 'I5', 'I5 2500', 'I5 3570', 'I5 4570', 'I5 4670', 'I5 4690',
       'I5 7400', 'I5 760', 'I5 7600', 'I5 G10', 'I5-10400', 'I5-10500',
       'I5-11500', 'I5-12400', 'I5-12500', 'I5-4590', 'I5-4670', 'I5-4690',
       'I5-760', 'I5-8500', 'I5-9400', 'I5-9600K', 'I7 7700', 'I7 8700',
       'I7-10700', 'I7-10700F', 'I7-11700', 'I7-12700', 'I7-13700', 'I7-870',
       'I7-8700', 'I7-9700', 'I7-9700F', 'i3 2100', 'i3 3240', 'i3 3250',
       'i3 4130', 'i3 4160', 'i3 4170', 'i3 540', 'i3 7100', 'i3 7320',
       'i3 8100', 'i3-2100', 'i3-4130', 'i5', 'i5 2500', 'i5 3570', 'i5 4570',
       'i5 4670', 'i5 4690', 'i5 7400', 'i5 760', 'i5 7600', 'i5-2500',
       'i5-4670', 'i7 2700', 'i7 4770', 'i7 7700', 'i7 8700', 'i7 9700',
       'i7-10700F', 'i7-10700f', 'i7-11700', 'i7-12700'],
      dtype='object')
```

DATA SCIENCE LABS

X 선정

- 숫자로 변경이 필요

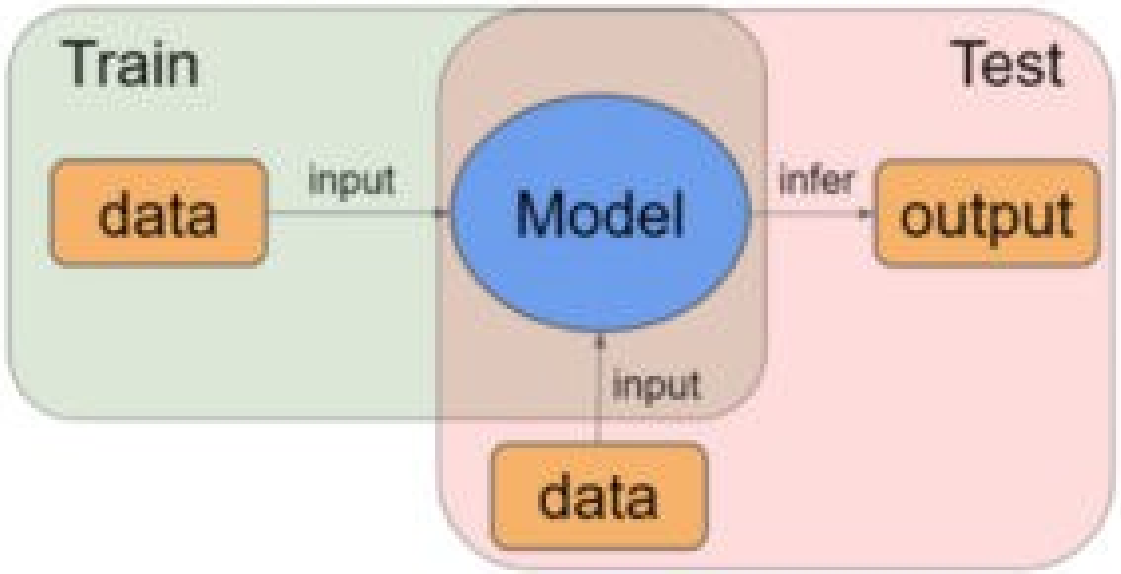
- 4G -> 0
- 8G -> 1

```
[54]  1 def to_codes(df):
      2     for col in df.columns:
      3         if df[col].dtype == "category":
      4             df[col] = df[col].cat.codes
      5     return df
```

```
[57]  1 data = to_codes(data)
```

훈련데이터 완성

- train 데이터는 모두 숫자로 구성




```
[64]  1 train = to_codes(data)
```

```
1 train
```

	GEN	CPU	MEM	VGA
0	0	0	6	44
1	0	0	6	5
2	0	0	3	44
3	0	0	6	44
4	0	0	3	44
...	...	...	...	...
383	2	46	1	24
384	2	46	1	24
385	2	46	1	24
386	2	46	1	24
387	2	46	1	37

388 rows × 4 columns

데이터 표준화

- BDI

- 키와 몸무게 는 서로 비교가능한가?

```
1 # 각 열에 대해 최소값과 최대값을 계산합니다.
2 mins = train.min(axis=0)
3 maxs = train.max(axis=0)
4
5 # 각 값을 표준화합니다.
6 df_std = (train - mins) / (maxs - mins)
7
8 # 표준화된 테이블을 출력합니다.
9 print(df_std)
```

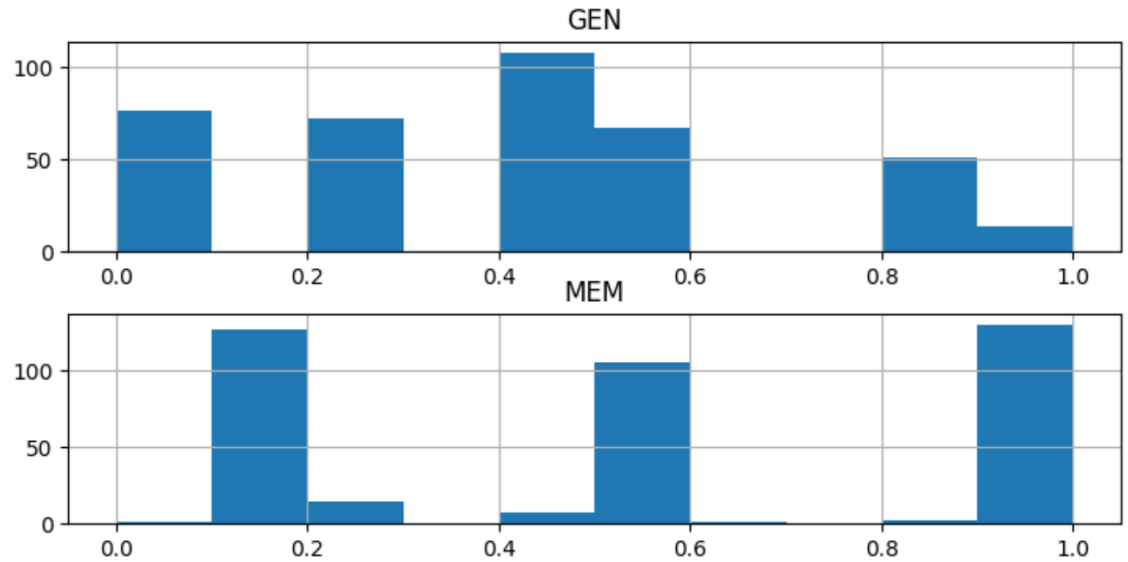
```
      GEN       CPU  MEM       VGA
0     0.0  0.000000  0.6  0.918367
1     0.0  0.000000  0.6  0.122449
2     0.0  0.000000  0.3  0.918367
3     0.0  0.000000  0.6  0.918367
4     0.0  0.000000  0.3  0.918367
..    ...       ...  ...       ...
383   0.4  0.589744  0.1  0.510204
384   0.4  0.589744  0.1  0.510204
385   0.4  0.589744  0.1  0.510204
386   0.4  0.589744  0.1  0.510204
387   0.4  0.589744  0.1  0.775510

[388 rows x 4 columns]
```

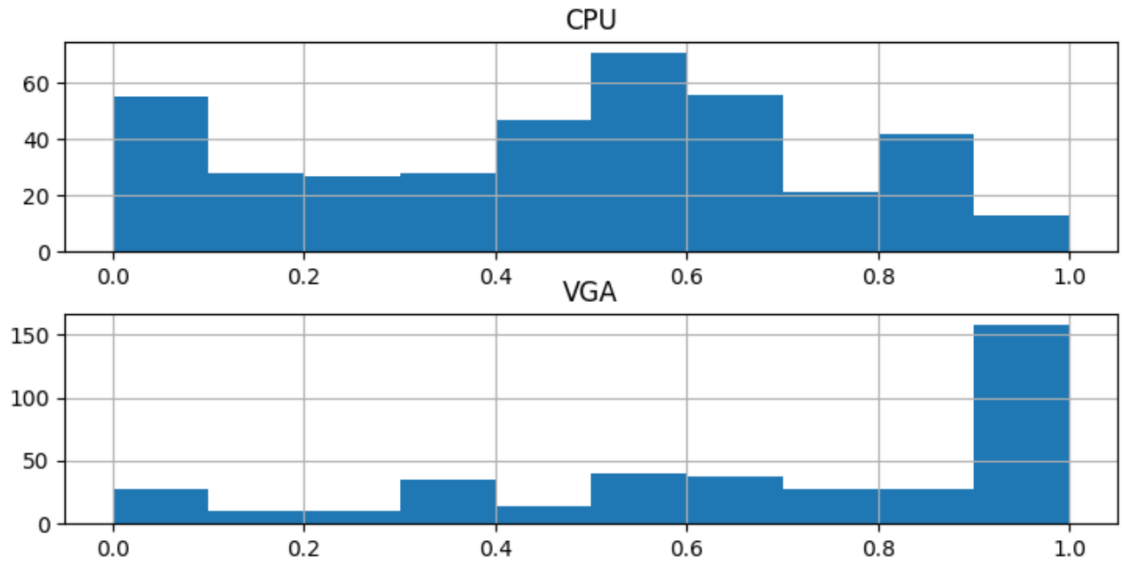
DATA SCIENCE

데이터 표준화

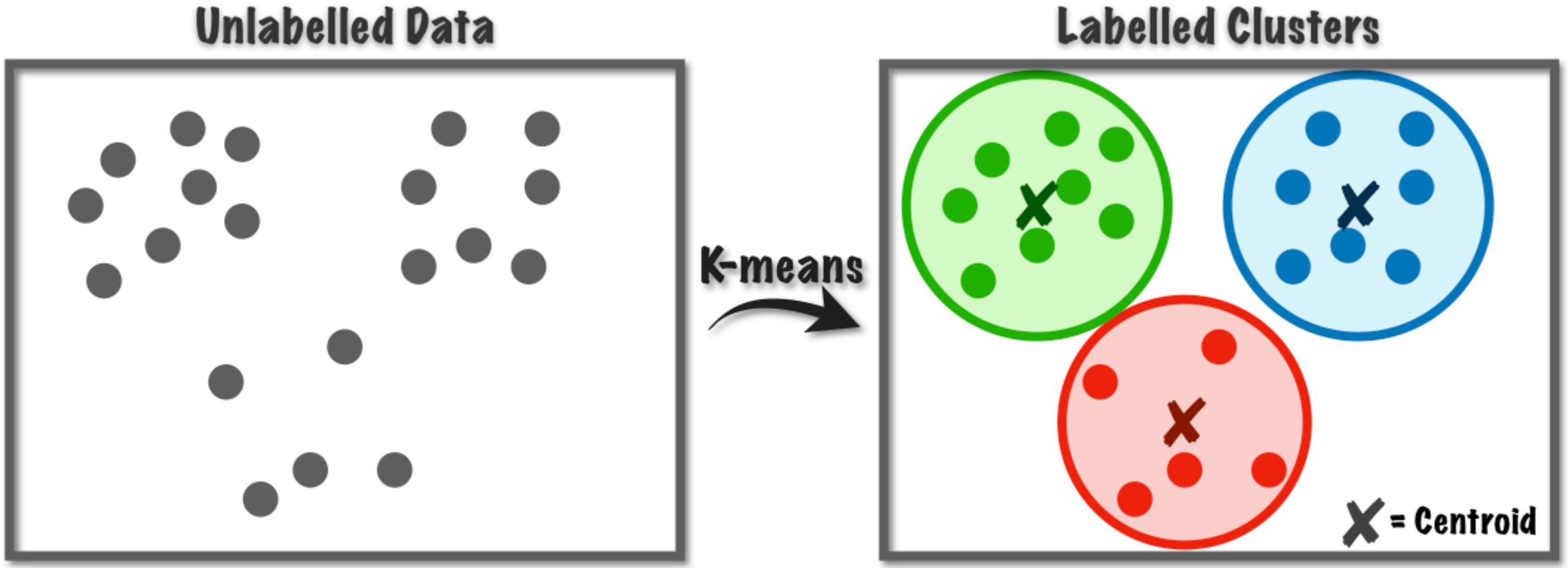
```
1 df_std.hist()
```

```
array([[<Axes: title={'center': 'GEN'}>, <Axes: title={'center': 'CPU'}>],
       [<Axes: title={'center': 'MEM'}>, <Axes: title={'center': 'VGA'}>]],
      dtype=object)
```





4. 클러스터 알고리즘: K-MEAN



https://towardsdatascience.com/k-means-a-complete-introduction-1702af9cd8c

4. 클러스터 알고리즘: K-MEAN

- K= 값을 자유롭게 설정

```
1 from sklearn.cluster import KMeans
2
3 model = KMeans(n_clusters=k)
4
5 labels = model.fit_predict(train)
```

4. 클러스터된 결과

```
1 labels
```

```
array([0, 2, 0, 0, 0, 0, 0, 0, 0, 1, 0, 2, 1, 0, 2, 1, 1, 1, 0, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 2, 1,
       1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 2, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 2, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 0, 2, 2, 1, 1, 2, 1, 0, 1, 0, 0, 0, 0, 2, 0, 2, 0, 0, 0, 0,
       2, 2, 2, 0, 2, 2, 2, 0, 0, 0, 2, 1, 2, 0, 0, 2, 0, 0, 1, 1, 0, 1,
       0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0,
       1, 0, 1, 0, 1, 2, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1,
       2, 1, 1, 2, 1, 0, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 2, 1, 2, 1, 2, 0,
       0, 1, 2, 2, 1, 2, 0, 2, 1, 1, 0, 0, 1, 1, 0, 2, 2, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 2, 1, 0, 0, 1, 1, 2, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 2, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], dtype=int32)
```

차트

[115]
```
# 클러스터별 데이터를 추출합니다.
cluster_0 = df[labels == 0]
cluster_1 = df[labels == 1]
cluster_2 = df[labels == 2]

# scatter plot을 생성합니다.
plt.scatter(cluster_0.GEN, cluster_0.CPU, label="클러스터 0")
plt.scatter(cluster_1.GEN, cluster_1.CPU, label="클러스터 1")
plt.scatter(cluster_2.GEN, cluster_2.CPU, label="클러스터 2")

# x축과 y축의 레이블을 설정합니다.
plt.xlabel("GEN")
plt.ylabel("CPU")

# 범례를 표시합니다.
plt.legend()

# 그래프를 출력합니다.
plt.show()
```

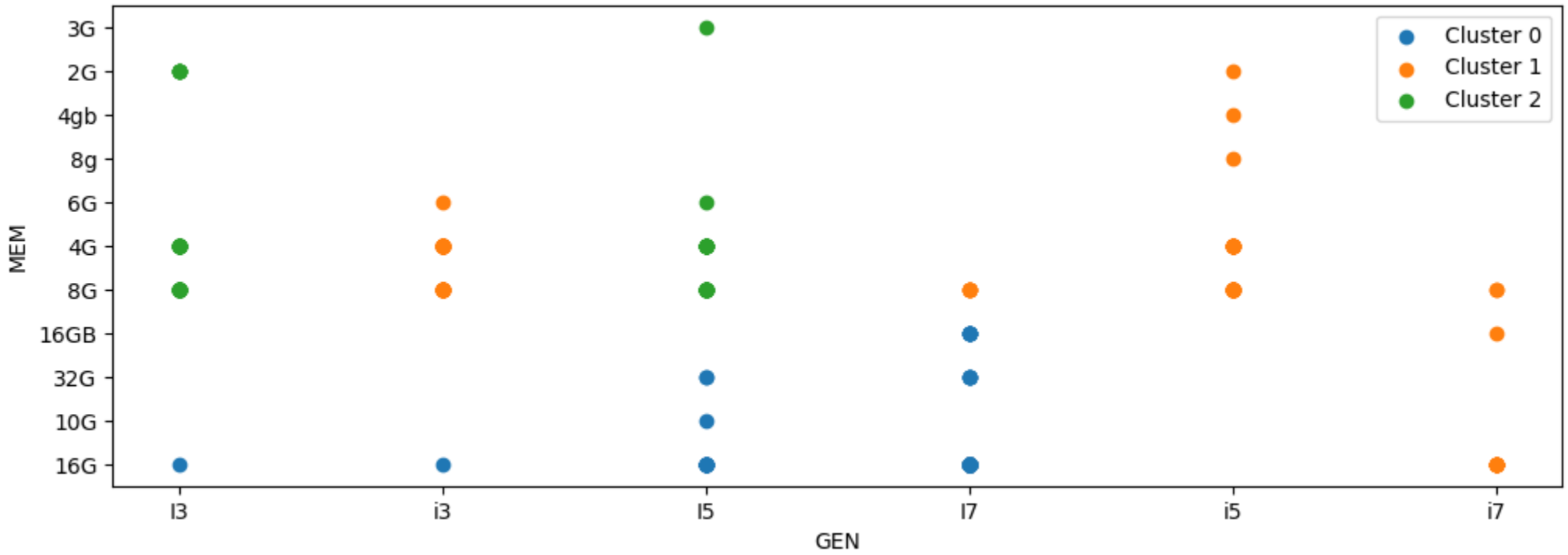
차트

- GEN, CPU 기준

```python
# 클러스터별 데이터를 추출합니다.
cluster_0 = df[labels == 0]
cluster_1 = df[labels == 1]
cluster_2 = df[labels == 2]

# scatter plot을 생성합니다.
plt.scatter(cluster_0.GEN, cluster_0.CPU, label="Cluster 0")
plt.scatter(cluster_1.GEN, cluster_1.CPU, label="Cluster 1")
plt.scatter(cluster_2.GEN, cluster_2.CPU, label="Cluster 2")

# x축과 y축의 레이블을 설정합니다.
plt.xlabel("GEN")
plt.ylabel("CPU")

# 범례를 표시합니다.
plt.legend()

# 그래프를 출력합니다.
plt.show()
```

차트

- GEN, MEM 기준



DATA SCIENCE LABS