

✓ 결측값-이상치 처리

✓ 데이터 정제(Data Cleansing)란?

- 데이터 분석 목적에 부합하는 데이터 품질을 확보하기 위한 일체의 데이터 작업
 - 데이터 분석 작업의 약 80% 수준이 데이터 수집 및 전처리 작업임
-

✓ 데이터 전처리 과정

- 주어진 데이터에 어떠한 문제가 있는지 확인
 - 데이터 분석 결과에 직접적인 성능 차이를 야기하므로 반드시 수행 필요
 - 잘못 측정되고 불필요한 데이터를 제거할 수 있어야 함
- data cleansing의 주요 5단계
 - 1. 데이터 베이스 정의
 - 2. 불량 데이터의 원인 찾기
 - 3. 우선순위 매기기

- 4. 데이터 베이스에 들어오는 불량 데이터 막기
- 5. 불량 데이터 제거
- 데이터 오류 확인
 - 결측치(Missing Value)
 - 노이즈(Noise)
 - 이상값(Outlier)
 - 노이즈와 이상값은 무작정 지우고 진행할 것이 아닌, 적합시켜야 하는 귀중한 데이터

✓ 결측 값(Missing value)

- 값이 기록되지 않았거나 관측되지 않은 경우, 데이터에 저장되는 값

- NA : Not Available(유효하지 않은)
- NaN : Not a Number(숫자가 아닌)
- Null : 아무것도 존재하지 않음을 의미

```
1 #결측값 확인 - isnull(), isnull().sum()
2 import pandas as pd
3 df = pd.DataFrame({"v1": [None, 200, None, 400], "v2": [None, 200, 100, 250], "v3": [40, 60, 500, None]})
4 df
```



	v1	v2	v3
0	NaN	NaN	40.0
1	200.0	200.0	60.0
2	NaN	100.0	500.0
3	400.0	250.0	NaN

```
1 df.isnull()
```



	v1	v2	v3
0	True	True	False
1	False	False	False
2	True	False	False
3	False	False	True

더블클릭 또는 Enter 키를 눌러 수정

```
1 df.isnull().sum()
```



	0
v1	2
v2	1
v3	1

dtype: int64

```
1 df[df.v1.isnull()]
```



	v1	v2	v3
0	NaN	NaN	40.0
2	NaN	100.0	500.0

✓ 결측 값이 분석 결과에 미치는 영향

- 1. 표본의 규모가 감소되어 검정력이 감소된다.
- 2. 표본의 대표성이 낮아져 분석 결과에 편향(Bias)을 가져온다.
- 3. 결측으로 인한 실제 문제의 발생을 식별하기 어렵다.

✓ 결측치 종류

- MCAR(완전 무작위 결측)

- 결측치가 완전히 random으로 발생
- 특정 변수와 연관성이 없음
- 변수의 관측치와 결측치에 모두 독립
- 결측치 대체 전후의 분포 변화 없음
- 관측치가 많다면 지우는 것도 방법임

- 예) 센서 오류, TCP통신 중 데이터 누락

- MAR(무작위 결측)

- 결측치는 random으로 발생
- 특정 변수값에 따른 조건부 발생 가능
- 다양한 결측치 대체 기법으로 추정 가능

- 예) 해시계(밤 되면 측정 불가) -> 원인이 분명히 존재!

- MNAR(비 무작위 결측)

- 결측치는 임의로 발생하지 않음
- 관측값과 결측값 모두에 영향을 받음
- 결측치 원인을 특정 짓기 어렵고, 단순 결측치 대체법 만으로는 해결하기 어려움
- 분명한 결측 사유가 존재하지만, 비정상적이며 그 원인을 찾기가 쉽지 않은 경우

- 예) 응답자들이 고의로 자신을 감추고 사실과 다른 응답을 하는 경우

✓ 결측치 대체 방법


- 단순 대체법((Imputation))

- 결측치를 대체하는 값으로 기존 데이터를 수정하는 방법. 대체법은 여러 하위 유형으로 나눌 수 있다

- 평균 대체법: 결측치를 해당 변수의 평균 값으로 대체
- 중앙값 대체법: 결측치를 해당 변수의 중앙값으로 대체
- 최빈값 대체법: 결측치를 해당 변수의 최빈값으로 대체

■ 마지막 관측값 대체법: 결측치를 직전의 관측값으로 대체

```
1 #Missing Value Imputation Methods with Implementation using Python
2 import numpy as np
3 import pandas as pd
4
5 # Creating a sample data
6 data = {'Score': [25, np.nan, 30, np.nan, 29, 27, 32, 31]}
7 df = pd.DataFrame(data)
8
9 # Mean Imputation
10 df['Score_Mean'] = df['Score'].fillna(df['Score'].mean())
11
12 # Median Imputation
13 df['Score_Median'] = df['Score'].fillna(df['Score'].median())
14
15 # Mode Imputation
16 df['Score_Mode'] = df['Score'].fillna(df['Score'].mode()[0])
17
18 print(df)
```




	Score	Score_Mean	Score_Median	Score_Mode
0	25.0	25.0	25.0	25.0
1	NaN	29.0	29.5	25.0
2	30.0	30.0	30.0	30.0
3	NaN	29.0	29.5	25.0
4	29.0	29.0	29.0	29.0
5	27.0	27.0	27.0	27.0
6	32.0	32.0	32.0	32.0
7	31.0	31.0	31.0	31.0

- 평균 대체법(Average Imputation)

- 관측 변수의 평균값으로 결측값을 대체하는 방법
- 평균, 중앙값, 최빈값 모두 사용 가능
- 분석이 간편하고 결측치 포함 관측치를 지우지 않아 좋으나 통계량의 변량을 과소 추정하는 문제 발생 -> 데이터가 왜곡 될 수 있음

```
1 # 평균 대체법 예시 코드
2 import pandas as pd
3
4 # 결측치가 있는 데이터프레임 예시
5 df = pd.DataFrame({'A': [1, 2, None, 4],
6                       'B': [5, None, 7, 8]})
7
8 # 열의 평균값 계산
9 mean_A = df['A'].mean()
10 mean_B = df['B'].mean()
11
12 # 결측치를 평균값으로 대체
13 df['A'].fillna(mean_A, inplace=True)
14 df['B'].fillna(mean_B, inplace=True)
15 df
```

 <ipython-input-6-d4c13f723aa3>:13: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment. The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values a

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method

```
df['A'].fillna(mean_A, inplace=True)
```

<ipython-input-6-d4c13f723aa3>:14: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment. The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values a

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method

```
df['B'].fillna(mean_B, inplace=True)
```

	A	B
0	1.000000	5.000000
1	2.000000	6.666667
2	2.333333	7.000000
3	4.000000	8.000000

• [완전 분석법\(Complete Analysis\)](#)

- [결측치가 포함된 특정 자료를 모두 무시하고 완전하게 관측된 자료만으로 데이터 분석에 필요한 데이터 셋을 구성하는 방법](#)
- 분석이 간편하지만, 관측치 부족 시에는 분석 기법에 대한 근거 미약


```

1 # 완전 분석법 예시 코드
2 import pandas as pd
3
4 # 결측치가 있는 데이터프레임 예시
5 df = pd.DataFrame({'A': [1, 2, None, 4],
6                        'B': [5, None, 7, 8]})
7
8 # 결측치가 있는 행 제거
9 df.dropna(inplace=True)
10
11 # 결측치가 있는 열 제거
12 df.dropna(axis=1, inplace=True)
13 df

```



	A	B
0	1.0	5.0
3	4.0	8.0

- 단순 확률 대체법(Single Stochastic Imputation)

- 평균 대체법에서 추정량 표준오차의 과소 추정문제를 보완하고자 고안된 방법 알고 있는 정보나 주어진 데이터를 기반으로 사건에 대한 확률 추정, 값을 부여한 후 대체하는 방법
- 단순 삭제나 평균 대체법보다 정확도가 높으나, 모든 상황에 적합하다고 볼 순 없음

```

1 # 단순 확률 대체법 예시 코드
2 import pandas as pd
3 import numpy as np
4
5 # 결측치가 있는 데이터프레임 예시

```

```
6 df = pd.DataFrame({'A': [1, 2, np.nan, 4], #파이썬의 NumPy 모듈에서 사용되는 결측값을 나타내는 기호로, Not a Number의 약자
7                      'B': [5, np.nan, 7, 8],
8                      'C': [np.nan, 2, 6, 8]})
9
10 # 각 열의 결측치 개수 확인
11 missing_counts = df.isnull().sum()
12 print(missing_counts)
13
14 # 각 열의 결측치를 해당 열의 랜덤 데이터로 대체
15 for col in df.columns:
16     if missing_counts[col] > 0:
17         col_data = df[col].dropna() # 결측치가 아닌 데이터 추출
18         fill_values = np.random.choice(col_data, size=missing_counts[col]) # 랜덤으로 대체값 선택
19
20 #레포트!!!!!!
21 df[col].fillna(pd.Series(fill_values), inplace=True) #fillna 메서드는 DataFrame에서 결측값을 원하는 값으로 변경하는 메서드
22 # df[col].fillna(pd.Series(fill_values, index=df.index), inplace=True) #fillna 메서드는 DataFrame에서 결측값을 원하는 값으로 변경하는 [
23 #https://stackoverflow.com/questions/42382263/valueerror-length-of-values-does-not-match-length-of-index-pandas-dataframe-u
24 #https://rfriend.tistory.com/634
25
26 df
```

```

→ A    1
   B    1
   C    1
dtype: int64
<ipython-input-8-38142d166c3d>:21: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values a

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method

```

```
df[col].fillna(pd.Series(fill_values), inplace=True) #fillna 메서드는 DataFrame에서 결측값을 원하는 값으로 변경하는 메서드
```

	A	B	C
0	1.0	5.0	6.0
1	2.0	NaN	2.0
2	NaN	7.0	6.0
3	4.0	8.0	8.0

• 다중 대체법

- N번의 단순 대체법을 반복 수행 - N개의 독립 데이터 셋 만들어 활용
- bias가 높고 정확도 성능 부족한 단순 대체법의 한계 극복
- 대체 적용 > 분석 > 결과 병합 단계 수행
- 반복적인 예측 과정을 통해 결측치를 추정하므로 결과는 매번 달라질 수 있음

```

1 # 다중 대체법 예시 코드
2 import pandas as pd
3 # explicitly require this experimental feature
4 from sklearn.experimental import enable_iterative_imputer # noqa
5 # now you can import normally from sklearn.impute
6 from sklearn.impute import IterativeImputer

```

```
7
8 from sklearn.impute import SimpleImputer
9
10 # 결측치가 있는 데이터프레임 예시
11 df = pd.DataFrame({'A': [1, 2, None, 4],
12                      'B': [5, None, 7, 8],
13                      'C': [2, 4, 6, None]})
14
15 # 다중 대체법 모델 생성
16 imputer = SimpleImputer()
17
18 # 결측치 대체
19 df_imputed = pd.DataFrame(imputer.fit_transform(df), columns=df.columns)
20 df
```



	A	B	C
0	1.0	5.0	2.0
1	2.0	NaN	4.0
2	NaN	7.0	6.0
3	4.0	8.0	NaN

✓ 이상치(Outlier)

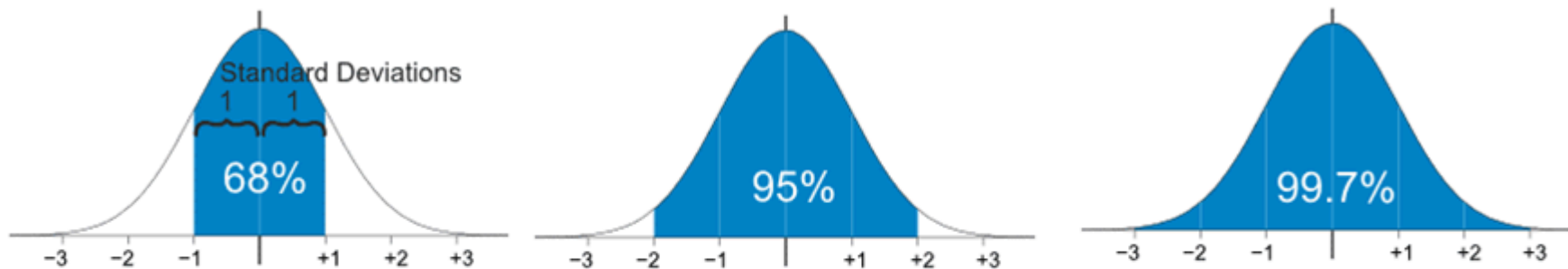
- 보통 관측된 데이터의 범위에서 많이 벗어난 아주 작은 값이나 큰 값

◦ 어떤 의사결정을 하는데 필요한 데이터를 분석 혹은 모델링할 경우, 이러한 이상치가 의사결정에 큰 영향을 미칠 수 있기 때문에 데이터 전처리 과정에서의 적절한 이상치 처리는 필수적임

- '데이터의 범위에서 많이 벗어난' 혹은 '아주 작은/큰'이라는 것은 정확히 어떤 기준으로 판단할 수 있을까?

✓ 이상치 탐색 방법 - Standard Deviation

- 데이터의 분포가 정규 분포를 이룰 때, 데이터의 표준 편차를 이용해 이상치를 탐지하는 방법



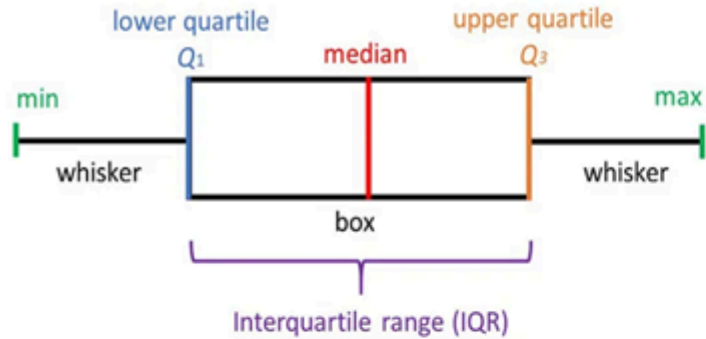
1 표준 편차, 2 표준 편차, 3 표준 편차를 사용했을 때 파란색 범위를 벗어나는 데이터는 이상치로 간주될 수 있음을 의미

$$Z = \frac{X - \mu}{\sigma} \quad \mu = \text{mean} \quad \sigma = \text{standard deviation}$$

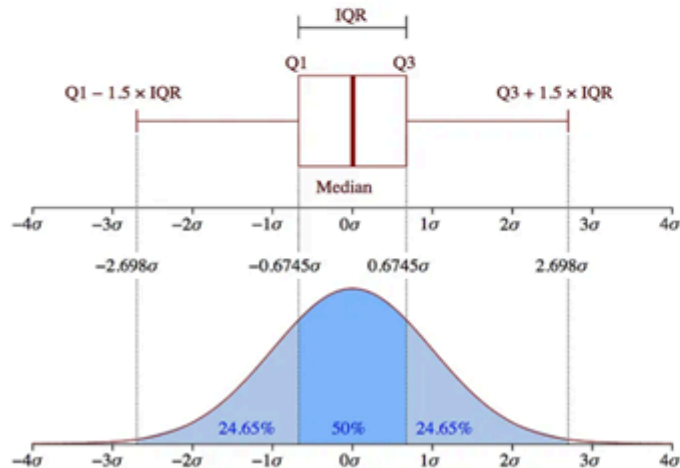
- ✓ 데이터의 **Z-score** 는 해당 데이터가 평균으로부터 얼마의 표준 편차만큼 벗어나 있는지를 의미함
- ✓ 예를 들어 3 표준 편차 만큼을 벗어나는 데이터를 이상치로 처리하는 것은 **Z-score**가 3 보다 크고 -3 보다 작은 데이터를 이상치로 처리하는 것과 같은 작업임

✓ 이상치 탐색 방법 - IQR(Interquartile Range) with Box plots 방법

- 데이터의 분포가 정규 분포를 이루지 않거나 한 쪽으로 skewed 한 경우, 데이터의 IQR 값을 이용해 이상치를 탐지하는 방법



- ✓ 박스 플롯은 위와 같이 데이터의 최소값, 최대값, 중간값, 첫번째 사분위(Q1) 및 세번째 사분위(Q3) 값에 대한 정보를 제공
- ✓ 사분위란, 데이터를 4등분 했을 때의 범위로 Q1 아래로는 최소값을 포함한 하위 25% 범위를, Q3 위로는 최대값을 포함한 상위 25% 범위를 포함
- ✓ 이때 IQR 값은 Q3에서 Q1을 뺀 값



- ✓ $(Q1 - 1.5 * IQR)$ 보다 작거나 $(Q3 + 1.5 * IQR)$ 보다 큰 데이터는 이상치로 처리
- ✓ 1.5 보다 큰 3 혹은 그 이상의 값을 곱하기도 하며 값이 클수록 더욱 최극단의 이상치를 처리함을 알 수 있음

1 #이상치 감지 및 제거

2 #데이터 분석 단계에서 발생한 이상치를 감지해야 하며, 동일한 접근 방식을 목록 및 시리즈 유형 객체에도 사용할 수 있음

3 #이상치 탐지에 사용되는 데이터 세트 Importing


4

5 import sklearn

```

6 from sklearn.datasets import load_diabetes
7 import pandas as pd
8 import matplotlib.pyplot as plt
9
10 # Load the dataset
11 diabetics = load_diabetes()
12
13 # Create the dataframe
14 column_name = diabetics.feature_names
15 df_diabetics = pd.DataFrame(diabetics.data)
16 df_diabetics.columns = column_name
17 print(df_diabetics.head())

```



```

      age      sex      bmi      bp      s1      s2      s3  W
0  0.038076  0.050680  0.061696  0.021872 -0.044223 -0.034821 -0.043401
1 -0.001882 -0.044642 -0.051474 -0.026328 -0.008449 -0.019163  0.074412
2  0.085299  0.050680  0.044451 -0.005670 -0.045599 -0.034194 -0.032356
3 -0.089063 -0.044642 -0.011595 -0.036656  0.012191  0.024991 -0.036038
4  0.005383 -0.044642 -0.036385  0.021872  0.003935  0.015596  0.008142

      s4      s5      s6
0 -0.002592  0.019907 -0.017646
1 -0.039493 -0.068332 -0.092204
2 -0.002592  0.002861 -0.025930
3  0.034309  0.022688 -0.009362
4 -0.002592 -0.031988 -0.046641

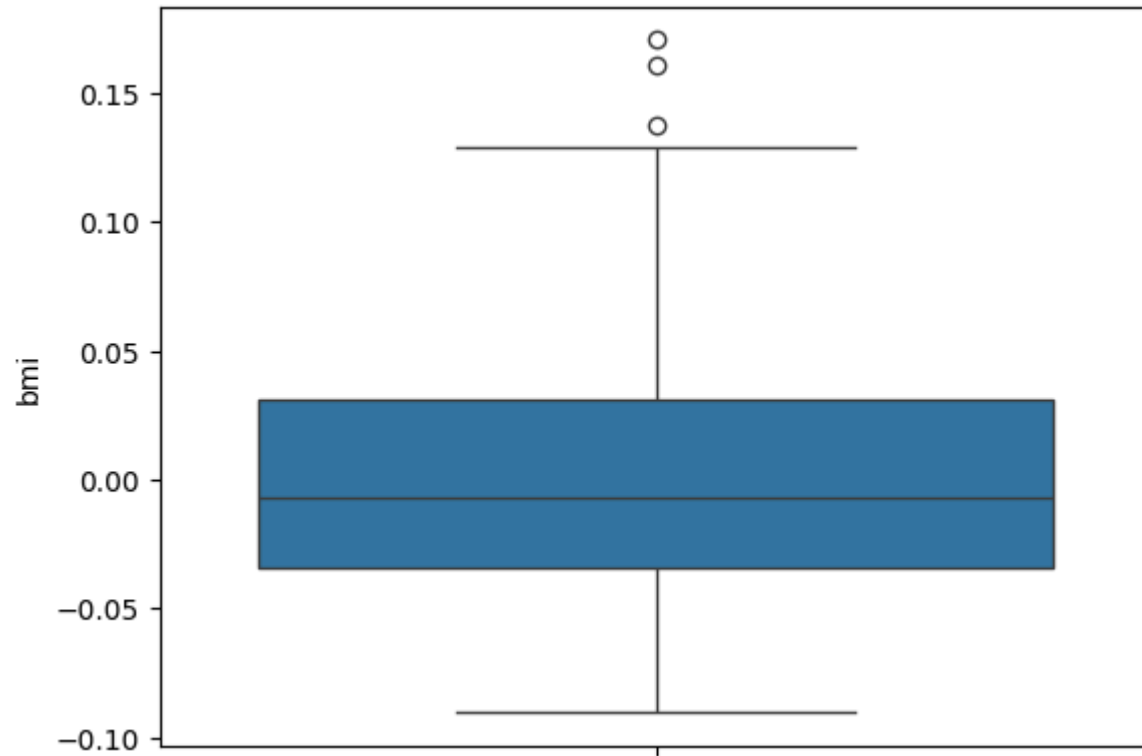
```

```

1 #상자 그림을 사용하여 이상치 시각화 및 제거
2 # Box Plot
3 import seaborn as sns
4 sns.boxplot(df_diabetics['bmi'])
5 #10보다 큰 값은 이상치로 작용하는 것을 분명히 볼 수 있음

```

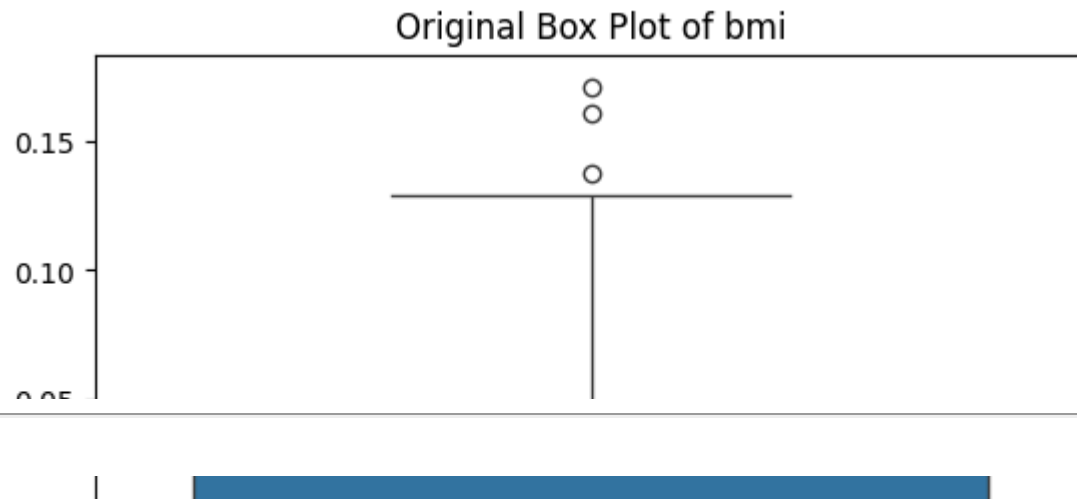
↩ <Axes: ylabel='bmi'>



```
1 #import seaborn as sns
2 #import matplotlib.pyplot as plt
3
4 import matplotlib.pyplot as plt
5
6 def removal_box_plot(df, column, threshold):
7     sns.boxplot(df[column])
8     plt.title(f'Original Box Plot of {column}')
9     plt.show()
10
11     removed_outliers = df[df[column] <= threshold]
12
13     sns.boxplot(removed_outliers[column])
14     plt.title(f'Box Plot without Outliers of {column}')
```



```
15     plt.show()
16     return removed_outliers
17 no_outliers = removal_box_plot(df_diabetics, 'bmi', 0.12) #threshold_value = 0.12
```



✓ <<<참조자료 사이트>>>

1. [데이터 유형: 숫자, 범주 및 순서 데이터의 이해](#)
2. [IQR\(사분위수 범위\)](#)
3. [통계분석-표준화\(Standardization\)란?](#)
4. [정규화 vs 표준화](#)
5. [데이터의 정규화\(normalization\) 또는 표준화\(standardization\)이 필요한 이유](#)
6. [교차 검증\(Cross Validation\)](#)
7. [표본추출방법 - 계통 추출, 층화 추출](#)