

CSE3013-02 / AIE3091-02

소프트웨어 개발도구 및 환경실습

2025년 1학기, 2주차 3/18 화요일
하기 전에 1주차 실습 리뷰

조교(강의자) 소개 및 연락처

- 박한나(메인 조교)
 - 인공지능학과 석사과정
 - 채점 문의: crescent.check@gmail.com
- 이강호
 - 컴퓨터공학과 석사과정
 - 출결 문의: lee7roho@sogang.ac.kr

>예비보고서/결과보고서/과제.. 관련 질문은 ‘사이버캠퍼스(eclass.sogang.ac.kr) - 질의응답’<

[1주차 실습 리뷰]

Python환경의 Google Colaboratory (a.k.a. Google Colab)에서
scikit-learn 라이브러리를 이용하여
기본적인 머신러닝 모델 제작

1주차 실습 과정

1. 데이터와 라이브러리 불러오기
 - Pandas에서 제공하는 데이터 형식으로 변환 필요
2. 살펴보기(원본 데이터, 기본적인 정보 확인) -> Q1, Q2
3. 데이터셋을 train / test로 나누기
 - 데이터를 나누기 위해서 `feature_vector`, `target_value` 나누기
4. train 데이터로 LinearRegression 모델 학습 진행
5. test 데이터도 사용해 모델 성능 평가

1. 데이터와 라이브러리 불러오기

- 일단 실행시키면 됩니다

- [NumPy](#)
- [pandas 2.2.3 documentation](#)
- [matplotlib.pyplot — Matplotlib 3.5.3 documentation](#)
- [Seaborn](#)
- [GitHub - tqdm/tqdm](#)
- [Scikit-learn](#)

```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 import seaborn as sns
5
6 import os
7 import sys
8 import gc
9
10 from tqdm.auto import tqdm
11
12 import sklearn
```

Q1. sklearn 라이브러리의 `fetch_california_housing` 함수를 사용하여 DataFrame 형태의 원본 데이터와, 데이터에 대한 설명을 출력하세요.

[fetch_california_housing — scikit-learn 1.6.1 documentation](https://scikit-learn.org/stable/modules/generated/sklearn.datasets.fetch_california_housing.html)

🏠 > API Reference > sklearn.datasets > fetch_california_housing

fetch_california_housing

```
sklearn.datasets.fetch_california_housing(*, data_home=None,  
download_if_missing=True, return_X_y=False, as_frame=False, n_retries=3,  
delay=1.0)
```


[\[source\]](#)

Load the California housing dataset (regression).

Samples total	20640
Dimensionality	8
Features	real
Target	real 0.15 - 5.

[정리] 다양한 import 방식


1. import ~: `sklearn.model_selection` 전체 다 가져온다 & 패키지명도 붙여야 함



```
import sklearn.model_selection

train, test = sklearn.model_selection.train_test_split(data, ...)
```

2. from ~ import ~: `sklearn.model_selection`에서 `train_test_split`만 가져온다



```
from sklearn.model_selection import train_test_split

train, test = train_test_split(data, ...)
```

Q1. sklearn 라이브러리의 `fetch_california_housing` 함수를 사용하여 DataFrame 형태의 원본 데이터와, 데이터에 대한 설명을 출력하세요.

[fetch_california_housing — scikit-learn 1.6.1 documentation](https://scikit-learn.org/stable/modules/generated/sklearn.datasets.fetch_california_housing.html)

🏠 > API Reference > sklearn.datasets > fetch_california_housing

fetch_california_housing

```
sklearn.datasets.fetch_california_housing(*, data_home=None,  
download_if_missing=True, return_X_y=False, as_frame=False, n_retries=3,  
delay=1.0)
```

[\[source\]](#)



```
# 1st method  
import sklearn  
data = sklearn.datasets.fetch_california_housing()  
  
# 2nd method  
from sklearn.datasets import fetch_california_housing  
data = fetch_california_housing()
```

loading dataset (regression):

	20640
	8
	real
	real 0.15 - 5.

Q1

A. 데이터를 불러올 때

[fetch_california_housing\(\)](#) 함수 안에 어떤 인자를 넣는다면 Pandas의 DataFrame 형태로 다룰 수 있습니다.

B. [fetch_california_housing\(\)](#) 함수가 뭘 반환하는지 문서 등에서 살펴보면 좋습니다.



	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	MedHouseVal
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23	4.526
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22	3.585
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24	3.521
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25	3.413
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25	3.422

...
20635	1.5603	25.0	5.045455	1.133333	845.0	2.560606	39.48	-121.09	0.781
20636	2.5568	18.0	6.114035	1.315789	356.0	3.122807	39.49	-121.21	0.771
20637	1.7000	17.0	5.205543	1.120092	1007.0	2.325635	39.43	-121.22	0.923
20638	1.8672	18.0	5.329513	1.171920	741.0	2.123209	39.43	-121.32	0.847
20639	2.3886	16.0	5.254717	1.162264	1387.0	2.616981	39.37	-121.24	0.894

20640 rows × 9 columns

```
.. _california_housing_dataset:
```

California Housing dataset

****Data Set Characteristics:****

:Number of Instances: 20640

:Number of Attributes: 8 numeric, predictive attributes and the target

:Attribute Information:

- MedInc median income in block group
- HouseAge median house age in block group
- AveRooms average number of rooms per household
- AveBedrms average number of bedrooms per household
- Population block group population
- AveOccup average number of household members
- Latitude block group latitude
- Longitude block group longitude

:Missing Attribute Values: None

This dataset was obtained from the StatLib repository.
https://www.dcc.fc.up.pt/~ltorgo/Regression/cal_housing.html

The target variable is the median house value for California districts, expressed in hundreds of thousands of dollars (\$100,000).

This dataset was derived from the 1990 U.S. census, using one row per census block group. A block group is the smallest geographical unit for which the U.S.

Q1. sklearn 라이브러리의 `fetch_california_housing` 함수를 사용하여 DataFrame 형태의 원본 데이터와, 데이터에 대한 설명을 출력하세요.

[fetch_california_housing — scikit-learn 1.6.1 documentation](https://scikit-learn.org/stable/modules/generated/sklearn.datasets.fetch_california_housing.html)

1. 함수 `fetch_california_housing()` 에 어떤 argument를 넣어야 하는지
2. 넣어서 나온 return value에서 우리가 원하는 값을 구하기 위해서는 어떤 값을 써야하는지

문제가 요구하는 것:

- A. DataFrame 형태의 원본 데이터
- B. 데이터에 대한 설명

Q1. sklearn 라이브러리의 `fetch_california_housing` 함수를 사용하여 DataFrame 형태의 원본 데이터와, 데이터에 대한 설명을 출력하세요.

[fetch_california_housing — scikit-learn 1.6.1 documentation](https://scikit-learn.org/stable/modules/generated/sklearn.datasets.fetch_california_housing.html)

1. 함수 `fetch_california_housing()` 에 어떤 argument를 넣어야 하는지
2. 넣어서 나온 return value에서 우리가 원하는 값을 구하기 위해서는 어떤 값을 써야하는지

문제가 요구하는 것:

- A. **DataFrame 형태의 원본 데이터**
- B. 데이터에 대한 설명

fetch_california_housing()'s return value

1. dataset으로 반환
 - a. Dictionary처럼 되어 있음

Bunch 자료형(바로가기 존재)

Dictionary로 된 return value의 key는

data, target, feature_names,
DESCR, frame ... 으로 되어 있다.

Returns:

dataset : *Bunch*

Dictionary-like object, with the following attributes.

data : *ndarray, shape (20640, 8)*

Each row corresponding to the 8 feature values in order. If `as_frame` is True, `data` is a pandas object.

target : *numpy array of shape (20640,)*

Each value corresponds to the average house value in units of 100,000. If `as_frame` is True, `target` is a pandas object.

feature_names : *list of length 8*

Array of ordered feature names used in the dataset.

DESCR : *str*

Description of the California housing dataset.

frame : *pandas DataFrame*

Only present when `as_frame=True`. DataFrame with `data` and `target`.

! Added in version 0.23.

fetch_california_housing()'s return value

Dictionary로 된 return value의 key는

data, target, feature_names,
DESCR, frame ... 으로 되어 있다.

```
from sklearn.datasets import fetch_california_housing
data = fetch_california_housing()
data
```

```
{'data': array([[ 8.3252, 41., 6.98412698, ..., 2.55555556,
                  37.88, -122.23, ],
                 [ 8.3014, 21., 6.23813708, ..., 2.10984183,
                  37.86, -122.22, ],
                 [ 7.2574, 52., 8.28813559, ..., 2.80225989,
                  37.85, -122.24, ],
                 ...,
                 [ 1.7, 17., 5.20554273, ..., 2.3256351,
                  39.43, -121.22, ],
                 [ 1.8672, 18., 5.32951289, ..., 2.12320917,
                  39.43, -121.32, ],
                 [ 2.3886, 16., 5.25471698, ..., 2.61698113,
                  39.37, -121.24, ]]),
 'target': array([4.526, 3.585, 3.521, ..., 0.923, 0.847, 0.894]),
 'frame': None,
 'target_names': ['MedHouseVal'],
 'feature_names': ['MedInc',
                   'HouseAge',
                   'AveRooms',
                   'AveBedrms',
                   'Population',
                   'AveOccup',
                   'Latitude',
                   'Longitude'],
 'DESCR': '.. _california_housing_dataset:\n\nCalifornia Housing dataset\n-----\n20640\n\nNumber of Attributes: 8 numeric, predictive attributes and the target\nHouseAge      median house age in block group\n- AveRooms     average number
```

sklearn.utils.Bunch (참고)

코랩은 친절하다

type(A): A의 type이 뭔지 반환하는 함수

```
from sklearn.datasets import fetch_california_housing
data = fetch_california_housing()
type(data)
```

```
sklearn.utils._bunch.Bunch
def __init__(**kwargs)
```

/usr/local/lib/python3.11/dist-packages/sklearn/utils/_bunch.py

Container object exposing keys as attributes.

Bunch objects are sometimes used as an output for functions and methods. They extend dictionaries by enabling values to be accessed by key, `bunch["value_key"]`, or by an attribute, `bunch.value_key`.

sklearn.utils.Bunch (참고)

[Bunch — scikit-learn 1.6.1 documentation](#)

- 파이썬 구문에서는 `sample_dict["key1"]`으로 해야하지만
- Bunch로 되어있는 자료형에서는 `sample_bunch.key1`도 동작

Returns:

dataset : `Bunch`

Dictionary-like object, the same as a `dict`, but also has attributes. `sklearn.utils.Bunch` has the following attributes:

data : `ndarray, shape (20640,`

0초



```
sample_dict = {'key1': 'value1', 'key2': 'value2'}  
sample_dict.key1
```



```
-----  
AttributeError                                Traceback (most recent call last)  
<ipython-input-7-cbe1673953d8> in <cell line: 0>()  
      1 sample_dict = {'key1': 'value1', 'key2': 'value2'}  
----> 2 sample_dict.key1  
  
AttributeError: 'dict' object has no attribute 'key1'
```

✓
0초



```
[10] from sklearn.utils import Bunch  
      sample_bunch = Bunch(key1='value1', key2='value2')  
      sample_bunch.key1, sample_bunch['key1']
```



```
('value1', 'value1')
```

sklearn.utils.Bunch (참고)

아무튼 그렇습니다



```
from sklearn.datasets import fetch_california_housing
data = fetch_california_housing()
print(data["feature_names"])
print(data.feature_names)
```



```
['MedInc', 'HouseAge', 'AveRooms', 'AveBedrms', 'Population', 'AveOccup', 'Latitude', 'Longitude']
['MedInc', 'HouseAge', 'AveRooms', 'AveBedrms', 'Population', 'AveOccup', 'Latitude', 'Longitude']
```


Q1. DataFrame 형태의 원본 데이터 구하기

원본 데이터: Feature value(8개 요소, data) + Target value(1개 요소, target) = 총 9개 요소



	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	MedHouseVal
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23	4.526
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22	3.585
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24	3.521
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25	3.413
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25	3.422
...
20635	1.5603	25.0	5.045455	1.133333	845.0	2.560606	39.48	-121.09	0.781
20636	2.5568	18.0	6.114035	1.315789	356.0	3.122807	39.49	-121.21	0.771
20637	1.7000	17.0	5.205543	1.120092	1007.0	2.325635	39.43	-121.22	0.923
20638	1.8672	18.0	5.329513	1.171920	741.0	2.123209	39.43	-121.32	0.847
20639	2.3886	16.0	5.254717	1.162264	1387.0	2.616981	39.37	-121.24	0.894

20640 rows × 9 columns



Q1. DataFrame 형태의 원본 데이터 구하기

원본 데이터: Feature value(8개 요소, data) + Target value(1개 요소, target) = 총 9개 요소

B번 방법: data랑 target을 구해서 합친다 -> Pandas에서 쓰는 DataFrame으로 변환

- data: ndarray(numpy에서 사용하는 type), (20640, 8)
- target: 같음

Returns:

dataset : *Bunch*

Dictionary-like object, with the following attributes.

data : *ndarray, shape (20640, 8)*

Each row corresponding to the 8 feature values in order. If `as_frame` is True, `data` is a pandas object.

target : *numpy array of shape (20640,)*

Each value corresponds to the average house value in units of 100,000. If `as_frame` is True, `target` is a pandas object.

Q1. DataFrame 형태의 원본 데이터 구하기

원본 데이터: Feature value(8개 요소, data) + Target value(1개 요소, target) = 총 9개 요소

A번 방법(의도): return value 중 frame을 이용 -> data와 target을 동시에 갖고 있는 DataFrame

- 잘 읽어보자
 - 함수 안에 `as_frame=True`를 넣어야 return value에 frame 요소가 생긴다

frame : *pandas DataFrame*

Only present when `as_frame=True`. DataFrame with `data` and `target`.

! *Added in version 0.23.*

Q1. DataFrame 형태의 원본 데이터 구하기

```
] from sklearn.datasets import fetch_california_housing
data = fetch_california_housing()
print(data.frame)
```

- None

```
from sklearn.datasets import fetch_california_housing
data = fetch_california_housing(as_frame=True)
print(data.frame)
```

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85
							.
							16
							17
							15
							19
							11

frame : *pandas DataFrame*

Only present when `as_frame=True`. DataFrame with `data` and `target`.

! Added in version 0.23.

Q1. sklearn 라이브러리의 `fetch_california_housing` 함수를 사용하여 **DataFrame** 형태의 원본 데이터와, 데이터에 대한 설명을 출력하세요.

```
# 함수 불러오기
from sklearn.datasets import fetch_california_housing

data = fetch_california_housing(as_frame=True)['frame'] # or .frame
display(data)
# description =
# print(description)
```



	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	MedHouseVal
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23	4.526
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22	3.585
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24	3.521
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25	3.413
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25	3.422
...
20635	1.5603	25.0	5.045455	1.133333	845.0	2.560606	39.48	-121.09	0.781
20636	2.5568	18.0	6.114035	1.315789	356.0	3.122807	39.49	-121.21	0.771
20637	1.7000	17.0	5.205543	1.120092	1007.0	2.325635	39.43	-121.22	0.923
20638	1.8672	18.0	5.329513	1.171920	741.0	2.123209	39.43	-121.32	0.847
20639	2.3886	16.0	5.254717	1.162264	1387.0	2.616981	39.37	-121.24	0.894

20640 rows × 9 columns



Q1. sklearn 라이브러리의 `fetch_california_housing` 함수를 사용하여 DataFrame 형태의 원본 데이터와, 데이터에 대한 설명을 출력하세요.

[fetch_california_housing — scikit-learn 1.6.1 documentation](https://scikit-learn.org/stable/modules/generated/sklearn.datasets.fetch_california_housing.html)

1. 함수 `fetch_california_housing()` 에 어떤 argument를 넣어야 하는지
2. 넣어서 나온 return value에서 우리가 원하는 값을 구하기 위해서는 어떤 값을 써야하는지

문제가 요구하는 것:

- A. DataFrame 형태의 원본 데이터
- B. 데이터에 대한 설명**

Q1. sklearn 라이브러리의 `fetch_california_housing` 함수를 사용하여 DataFrame 형태의 원본 데이터와, 데이터에 대한 설명을 출력하세요.

DESCR: str

-> Description of the California housing dataset.

Returns:

dataset : *Bunch*

Dictionary-like object, with the following attributes.

data : *ndarray, shape (20640, 8)*

Each row corresponding to the 8 feature values in order. If `as_frame` is True, `data` is a pandas object.

target : *numpy array of shape (20640,)*

Each value corresponds to the average house value in units of 100,000. If `as_frame` is True, `target` is a pandas object.

feature_names : *list of length 8*

Array of ordered feature names used in the dataset.

DESCR : *str*

Description of the California housing dataset.

frame : *pandas DataFrame*

Only present when `as_frame=True`. DataFrame with `data` and `target`.



Added in version 0.23.



```
# 함수 불러오기
from sklearn.datasets import fetch_california_housing

data = fetch_california_housing(as_frame=True) ['frame']
display(data)
description = fetch_california_housing(as_frame=True) ['DESCR']
print(description)
```



20636	2.5568	18.0	6.114035	1.315789	356.0	3.122807	39.49	-121.21	0.771
20637	1.7000	17.0	5.205543	1.120092	1007.0	2.325635	39.43	-121.22	0.923
20638	1.8672	18.0	5.329513	1.171920	741.0	2.123209	39.43	-121.32	0.847
20639	2.3886	16.0	5.254717	1.162264	1387.0	2.616981	39.37	-121.24	0.894

20640 rows × 9 columns

.. _california_housing_dataset:

California Housing dataset

****Data Set Characteristics:****

:Number of Instances: 20640

Q2

1. 위에서 구한 data의 크기를 출력

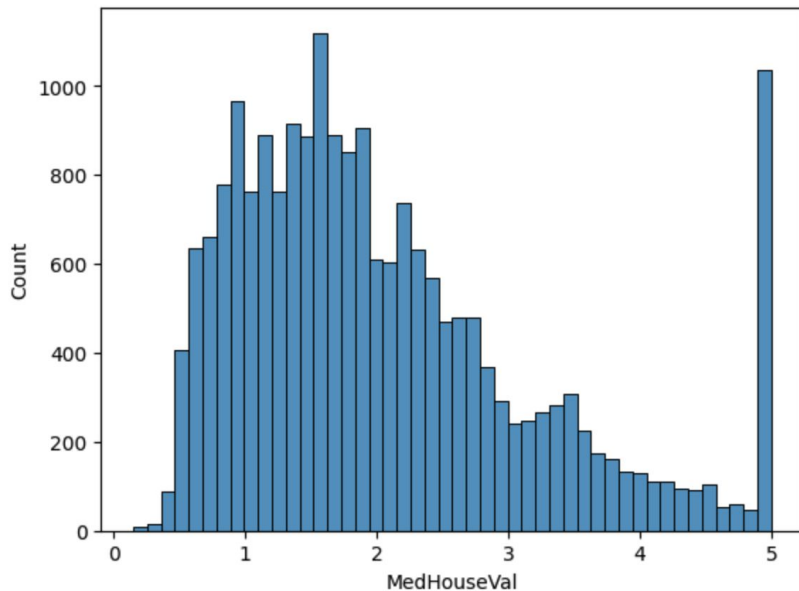
data의 타입은 'pandas.core.frame.DataFrame' 형식으로 되어있습니다.

2. 그래프 그리는 라이브러리를 사용하여

MedHouseVal의 histogram을 출력해보세요.

관련 라이브러리는 이미 import 되어있습니다

(20640, 9)
<Axes: xlabel='MedHouseVal', ylabel='Count'>



Q2-1. 1. 위에서 구한 data의 크기를 출력
data의 타입은 'pandas.core.frame.DataFrame'형식으로 되어있습니다.

shape of pandas dataframe



전체 이미지 동영상 쇼핑 짧은 동영상 뉴스 웹 : 더보기



Pandas

<https://pandas.pydata.org/docs/reference/api/pand...>

pandas.DataFrame.shape — pandas 2.2.3 documentation

pandas.DataFrame.shape# Return a tuple representing the dimensionality of the **DataFrame**. See also Examples

Shape

pandas.DataFrame.shape# ... Return a tuple representing ...

[pandas.DataFrame.shape — pandas 2.2.3 documentation](#)

[🏠](#) > [API reference](#) > [DataFrame](#) > [pandas.DataFrame...](#)

pandas.DataFrame.shape

property DataFrame.**shape**

[\[source\]](#)

Return a tuple representing the dimensionality of the DataFrame.

See also

[ndarray.shape](#)

Tuple of array dimensions.

Examples

```
>>> df = pd.DataFrame({'col1': [1, 2], 'col2': [3, 4]})
>>> df.shape
(2, 2)
```

```
>>> df = pd.DataFrame({'col1': [1, 2], 'col2': [3, 4],
...                    'col3': [5, 6]})
>>> df.shape
(2, 3)
```

```
] # data 크기 출력
print(data.shape)
```

```
# histogram 출력
```

```
> (20640, 9)
```

2. MedHouseVal에 대한 histogram을 출력합니다.

seaborn: 데이터 시각화 라이브러리인 matplotlib의 고급 버전

- 그래프를 그리는데 너무 큰 힘을 안들여도 됨

```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 import seaborn as sns
5
6 import os
7 import sys
8 import gc
9
10 from tqdm.auto import tqdm
11
12 import sklearn
```

data에서 MedHouseVal에 선택

pandas select column



전체 이미지 동영상 쇼핑 짧은 동영상 뉴스 웹 : 더보기



Pandas

<https://pandas.pydata.org> > docs > 03_subset_data



How do I select a subset of a DataFrame? - Pandas - PyData |

To select a single column, **use square brackets [] with the column name of the column of interest.**
Each column in a DataFrame is a Series.

[How do I create plots in pandas?](#)

[Dev](#)

[1.0](#)

[1.4](#)

Pandas DataFrame 요소 선택

[How do I select a subset of a DataFrame? — pandas 2.2.3 documentation](#)

How do I select specific columns from a
DataFrame?



I'm interested in the age of the Titanic passengers.

```
In [4]: ages = titanic["Age"]
```

```
In [5]: ages.head()
```

```
Out[5]:
```

```
0    22.0
```

```
1    38.0
```

```
2    26.0
```

```
3    35.0
```

```
4    35.0
```

```
Name: Age, dtype: float64
```

matplotlib.pyplot.hist

matplotlib histogram

전체 이미지 쇼핑 동영상 짧은 동영상 뉴스 웹 : 더보기



Matplotlib

<https://matplotlib.org> > stable > api > _as_gen > matplotlibli... ⋮

matplotlib.pyplot.hist — Matplotlib 3.10.1 documentation

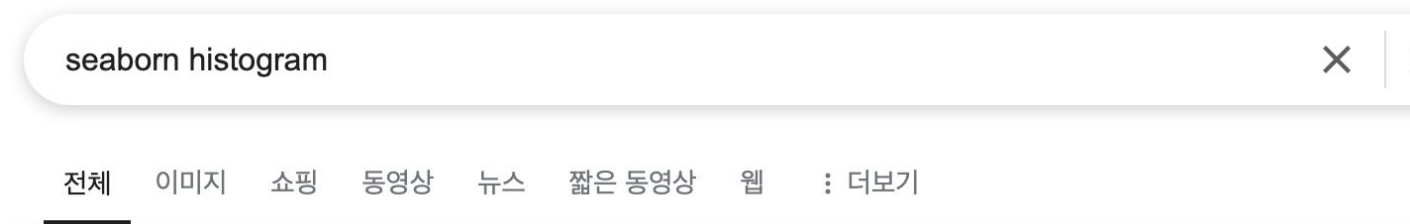
Histograms

Histogram bins, density, and...

Demo of the histogram...

Hist2d

seaborn.histplot — seaborn 0.13.2 documentation



도움말: 결과를 **한국어**로 표시합니다. 언어별 필터링에 대해서도 자세히 알아보세요.



Seaborn

<https://seaborn.pydata.org> > generated > seaborn.histplo... ⋮

seaborn.histplot — seaborn 0.13.2 documentation - PyData |

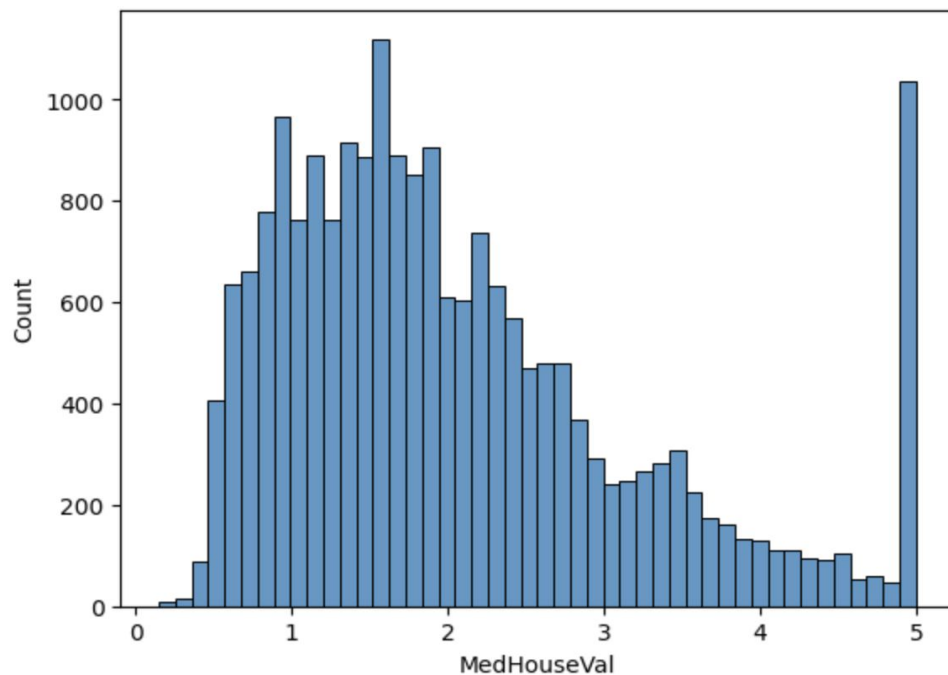
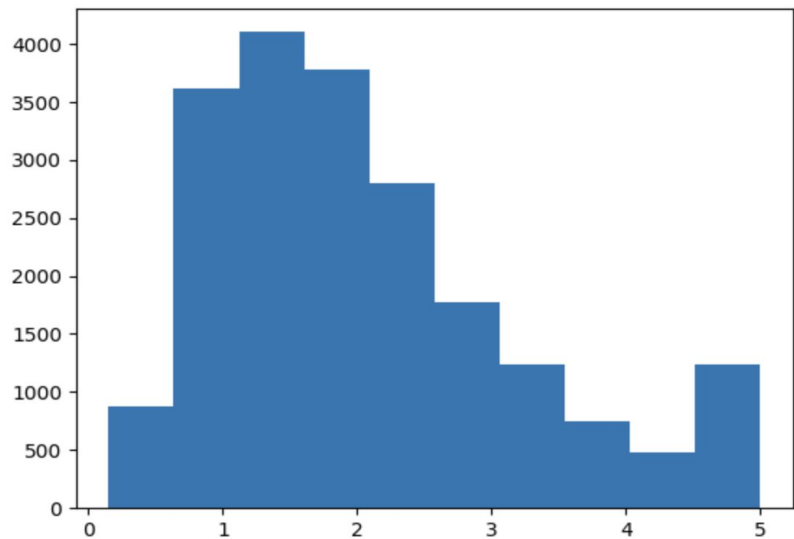
A histogram is a classic visualization tool that **represents the distribution of one or more variables** by counting the number of observations that fall within ...


```
# histogram 출력
sns.histplot(data["MedHouseVal"])
```

```
<Axes: xlabel='MedHouseVal', ylabel='Count'>
```

```
plt.hist(data["MedHouseVal"])
```

```
(array([ 877., 3612., 4099., 3771., 2799., 1769., 1239., 752., 479.,
        1243.]),
 array([0.14999, 0.634992, 1.119994, 1.604996, 2.089998, 2.575,
        3.060002, 3.545004, 4.030006, 4.515008, 5.00001 ]),
 <BarContainer object of 10 artists>)
```



Q3

Q3. 학습을 위해서 데이터를 train, test 데이터로 분할합니다.

- 데이터 분할을 위한 feature_vector와 target value를 정의하고,
- train_test_split 함수를 사용하여 데이터를 분할하세요.

(단, train : test 데이터의 비율은 8 : 2로 합니다.)

```
[ ] # 데이터 분할에 사용할 데이터 정의

X = # feature vector
y = # target value

from sklearn.model_selection import train_test_split
# train_test_split 함수 사용

print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

Q3

A. DataFrame형식의 data에서 feature vector와 target value을 찾아야 합니다.

B. train_test_split() 함수의 return값으로 다음의 sanity check가 진행됩니다

```
10 # sanity check: (16512, 8) (4128, 8) (16512,) (4128,)
11 print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

⇒ (16512, 8) (4128, 8) (16512,) (4128,)

Q3.

1. 위에 코드에서 data는 Pandas의 DataFrame 형식
2. 뭐가 Feature Vector이고 뭐가 Target Value일까?

Returns:

dataset : *Bunch*

Dictionary-like object, with the following attributes.

data : *ndarray, shape (20640, 8)*

Each row corresponding to the 8 feature values in order. If `as_frame` is True, `data` is a pandas object.

target : *numpy array of shape (20640,)*

Each value corresponds to the average house value in units of 100,000. If `as_frame` is True, `target` is a pandas object.

feature_names : *list of length 8*

Array of ordered feature names used in the dataset.

DESCR : *str*

Description of the California housing dataset.

frame : *pandas.DataFrame*

Q3.

위의 코드에서

`fetch_california_housing()` 함수
이용해 return value 중 frame을 이용

-> frame은 data(feature_values)와
target(target vector)를 가지고 있음



```
# 함수 불러오기
from sklearn.datasets import fetch_california_housing

data = fetch_california_housing(as_frame=True) ['frame']
display(data)
```

Returns:

dataset : *Bunch*

Dictionary-like object, with the following attributes.

data : *ndarray, shape (20640, 8)*

Each row corresponding to the 8 feature values in order. If `as_frame` is True, `data` is a pandas object.

target : *numpy array of shape (20640,)*

Each value corresponds to the average house value in units of 100,000. If `as_frame` is True, `target` is a pandas object.

feature_names : *list of length 8*

Array of ordered feature names used in the dataset.

DESCR : *str*

Description of the California housing dataset.

frame : *pandas DataFrame*

Only present when `as_frame=True`. DataFrame with `data` and `target`.



Added in version 0.23.

Q3-A. DataFrame형식의 data에서 feature vector와 target value을 찾아야 합니다.

frame은 data(feature_values) + target(target vector, MedHouseVal)

X = data에서 target을 뺀 것

y = data에서 target -> [이전의 슬라이드에 설명](#)

[] # 데이터 분할에 사용할 데이터 정의

X = # feature vector

y = # target value



	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	MedHouseVal
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23	4.526
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22	3.585
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24	3.521
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25	3.413
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25	3.422
...
20635	1.5603	25.0	5.045455	1.133333	845.0	2.560606	39.48	-121.09	0.781
20636	2.5568	18.0	6.114035	1.315789	356.0	3.122807	39.49	-121.21	0.771
20637	1.7000	17.0	5.205543	1.120092	1007.0	2.325635	39.43	-121.22	0.923
20638	1.8672	18.0	5.329513	1.171920	741.0	2.123209	39.43	-121.32	0.847
20639	2.3886	16.0	5.254717	1.162264	1387.0	2.616981	39.37	-121.24	0.894

20640 rows × 9 columns

Q3-A

X = data에서 target을 뺀 것 -> [pandas.DataFrame.drop — pandas 2.2.3 documentation](https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.drop.html)

pandas dataframe column remove



Pandas

<https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.drop.html>

pandas.DataFrame.drop — pandas 2.2.3 documentation

Remove rows or columns by specifying label names and corresponding axis, or by directly specifying index or column names. When using a multi-index, labels on ...

Dev

Pandas - Drop

pandas.Series.drop

Dropna

Q3-2. train_test_split 함수를 사용하여 데이터를 분할하세요.

train: test = 8:2 / random_state = 42

(단, train : test 데이터의 비율은 8 : 2로 합니다.)

```
[ ] # 데이터 분할에 사용할 데이터 정의

X = # feature vector
y = # target value

from sklearn.model_selection import train_test_split
# train_test_split 함수 사용

print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```


train_test_split — scikit-learn 1.6.1 documentation

```
>>> X_train, X_test, y_train, y_test = train_test_split(
...     X, y, test_size=0.33, random_state=42)
...
>>> X_train
array([[4, 5],
       [0, 1],
       [6, 7]])
>>> y_train
[2, 0, 3]
>>> X_test
array([[2, 3],
       [8, 9]])
>>> y_test
[1, 4]
```

test_size : float or int, default=None

If float, should be between 0.0 and 1.0 and represent the proportion of the dataset to include in the test split. If int, represents the absolute number of test samples. If None, the value is set to the complement of the train size. If `train_size` is also None, it will be set to 0.25.

train_size : float or int, default=None

If float, should be between 0.0 and 1.0 and represent the proportion of the dataset to include in the train split. If int, represents the absolute number of train samples. If None, the value is automatically set to the complement of the test size.

Q3

```
# 데이터 분할에 사용할 데이터 정의

X = data.drop(columns='MedHouseVal') # feature vector
y = data.MedHouseVal # target value

from sklearn.model_selection import train_test_split

# train_test_split 함수 사용
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
                                                    random_state=42)
# sanity check: (16512, 8) (4128, 8) (16512,) (4128,)
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)

(16512, 8) (4128, 8) (16512,) (4128,)
```

Q4. Q3에서 만든 학습 데이터를 사용하여, LinearRegression 모델을 학습합니다.

LinearRegression 모델을 불러옵니다.

import된 라이브러리에 속한 해당 모델은 여러가지 함수도 포함한 class로 되어있습니다.



▼ LinearRegression



LinearRegression()

LinearRegression – scikit-learn 1.6.1 documentation

Examples

```
>>> import numpy as np
>>> from sklearn.linear_model import LinearRegression
>>> X = np.array([[1, 1], [1, 2], [2, 2], [2, 3]])
>>> #  $y = 1 * x_0 + 2 * x_1 + 3$ 
>>> y = np.dot(X, np.array([1, 2])) + 3
>>> reg = LinearRegression().fit(X, y)
>>> reg.score(X, y)
```

fit(X, y, sample_weight=None)

Fit linear model.

Parameters:

X : {array-like, sparse matrix} of shape (n_samples, n_features)

Training data.

y : array-like of shape (n_samples,) or (n_samples, n_targets)

Target values. Will be cast to X's dtype if necessary.

모델 불러오기

```
from sklearn.linear_model import LinearRegression
```

모델 선언하기

```
reg = LinearRegression()
```

모델 학습하기

```
reg.fit(X_train, y_train)
```

▼ LinearRegression ⓘ ?

LinearRegression()

Why `reg.fit(X_train, y_train)`??

X = Feature_value = 문제 / y = Target_vector = 답

학습용(train) / 평가용(test)로 나눈 다음에 LinearRegression 모델에 train 데이터셋만 넣음
-> 나중에 잘 학습되었는지 판단하기 위해서 test 데이터셋을 활용함 (Q5)

```
# train_test_split 함수 사용
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
# sanity check: (16512, 8) (4128, 8) (16512,) (4128,)
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

```
(16512, 8) (4128, 8) (16512,) (4128,)
```

Q5

Q5. Q4에서 학습한 모델의 예측값과 실제 정답을 비교하여 모델 성능을 평가합니다. 성능 평가는 학습 데이터와 테스트 데이터 둘다 평가합니다.

- 예측 진행
- RMSE 이용해서 예측한 값과 실제 값 차이 구하기

```
# 검증에 사용될 함수 불러오기
```

```
train_preds = # 모델의 학습 데이터 예측값  
test_preds = # 모델의 테스트 데이터 예측값
```

```
train_rmse = # 학습 데이터로 평가한 RMSE  
test_rmse = # 테스트 데이터로 평가한 RMSE
```

```
print("Train RMSE : %.4f" % train_rmse)  
print("Test RMSE : %.4f" % test_rmse)
```

LinearRegression — scikit-learn 1.6.1 documentation

Examples

```
>>> import numpy as np
>>> from sklearn.linear_model import LinearRegression
>>> X = np.array([[1, 1], [1, 2], [2, 2], [2, 3]])
>>> #  $y = 1 * x_0 + 2 * x_1 + 3$ 
>>> y = np.dot(X, np.array([1, 2])) + 3
>>> reg = LinearRegression().fit(X, y)
>>> reg.score(X, y)
1.0
>>> reg.coef_
array([1., 2.])
>>> reg.intercept_
np.float64(3.0...)
>>> reg.predict(np.array([[3, 5]]))
array([16.])
```

predict(x)

Predict using the linear model.

Parameters:

X : array-like or sparse matrix, shape (n_samples, n_features)

Samples.

Returns:

C : array, shape (n_samples,)

Returns predicted values.

```
train_preds = reg.predict(X_train) # 모델의 학습 데이터 예측값
test_preds = reg.predict(X_test) # 모델의 테스트 데이터 예측값
print(train_preds, test_preds)
```

```
[1.93725845 2.48910616 2.64735483 ... 2.03879912 2.84075139 2.27373156] [0.71
```

Root Mean Square Error

방법 A: MSE를 구하고 np.sqrt

방법 B: RMSE를 구하기 -> [root_mean_squared_error — scikit-learn 1.6.1 documentation](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.root_mean_squared_error.html)

sklearn rmse



전체 이미지 동영상 쇼핑 짧은 동영상 뉴스 도서 : 더보기



Scikit-learn

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.root_mean_squared_error.html

root_mean_squared_error

Root mean squared error regression loss. Read more in the User Guide. Added in version 1.4.
Defines aggregating of multiple output values.



Scikit-learn

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_squared_error.html

mean_squared_error — scikit-learn 1.6.1 documentation

Mean squared error regression loss. Read more in the User Guide. Defines aggregating of multiple output values. Array-like value defines weights used to ...

Q5

검증에 사용될 평가 함수 불러오기

```
from sklearn.metrics import mean_squared_error
```

```
train_preds = reg.predict(X_train) # 모델의 학습 데이터 예측값
```

```
test_preds = reg.predict(X_test) # 모델의 테스트 데이터 예측값
```

```
train_rmse = np.sqrt(mean_squared_error(y_train, train_preds)) # 학습 데이터로  
평가한 RMSE
```

```
test_rmse = np.sqrt(mean_squared_error(y_test, test_preds)) # 테스트 데이터로  
평가한 RMSE
```

```
print("Train RMSE : %.4f" % train_rmse)
```

```
print("Test RMSE : %.4f" % test_rmse)
```

```
Train RMSE : 0.7197
```

```
Test RMSE : 0.7456
```

```
from sklearn.metrics import root_mean_squared_error
```

```
new_train_rmse = root_mean_squared_error(y_train, train_preds)
```

```
new_test_rmse = root_mean_squared_error(y_test, test_preds)
```

```
print("Train NEW RMSE : %.4f" % new_train_rmse)
```

```
print("Test NEW RMSE : %.4f" % new_test_rmse)
```

```
Train NEW RMSE : 0.7197
```

```
Test NEW RMSE : 0.7456
```

학습용(train) / 평가용(test)로 나눈 다음에 LinearRegression 모델에 train 데이터셋만 넣음
-> 나중에 잘 학습되었는지 판단하기 위해서 test 데이터셋을 활용함 (Q5)

검증 하는 법

모델에 train 데이터셋 중 문제(X_train)을 넣는다 -> X_train에 대한 예측값 train_preds가 됨

- 예측한 값(train_preds)이랑 실제 값(답, y_train)과의 차이를 구하는 것이 검증.
 - 오차가 얼마나 많은지에 대해 다양하게 평가 함수를 사용하는데 RMSE를 사용해서 진행했을 뿐

test도 동일하게 진행하면 됨.

- 모델은 test 데이터셋을 보고 학습하지 않았기 때문에 모델 평가시 유용한 지표가 될 수 있음.

```
# 검증에 사용될 평가 함수 불러오기
from sklearn.metrics import root_mean_squared_error

train_preds = reg.predict(X_train) # 모델의 학습 데이터 예측값
test_preds = reg.predict(X_test) # 모델의 테스트 데이터 예측값

new_train_rmse = root_mean_squared_error(y_train, train_preds) # 학습 데이터로
평가한 RMSE
new_test_rmse = root_mean_squared_error(y_test, test_preds) # 테스트 데이터로 평가한
RMSE
```