

시간이 아깝지 않은

데이터 분석

강사 이영배

강사 이력

1. 2023년 8월 ~ 2023년 12월 「2023년 하반기 산학연 미래유망기술 R&D기획지원」 수요조사 자문위원
(전남대학교 병원, 한림대학교 의대, 광주 대학교, 쌍용 소프트웨어)
2. 2018년 ~ 2022년 10월 : 데이터분석, 머신 러닝, 딥러닝 개발자
3. 2021년 2월 대학원 졸업
4. 개발 경력
 - ① Na 배설량 분석 및 배설량 회귀 방정식 도출(분당 서울대병원)
 - ② 교육부 학생 건강검사 표본통계 분석을 통한 어린이 비만 지수 예측 인공지능 모델 개발(중소벤처기업부 과제)
 - ③ 휴대폰 전파 세기와 기상청 기상 정보 분석을 통한 강수량 예측 인공지능 모델 개발(중소벤처기업부 과제)
 - ④ 전통식품 발효 과정 예측 인공 지능 모델 개발(2022년도 NIA 데이터 구축 사업)
 - ⑤ 신속 항원 진단 키트 검사 결과 분석 AI 모델 개발(자사 솔루션)
 - ⑥ 한글 특허, 영문 특허의 특허 요약 언어 모델 개발(과학기술정보통신부 연구산업육성사업)
 - ⑦ 한글 문서 요약 언어 모델 개발(자사 솔루션, 공공 기관 납품용)
 - ⑧ 딥러닝 기반 의류 가상 피팅 애플리케이션 개발(2020년 중소벤처기업부 예비창업패키지 인공지능분야 선정)
5. KCI 논문 등재(2021년 10월)
: 음표 임베딩과 마디 임베딩을 이용한 곡의 생성 및 정량적 평가 방법(정보처리학회논문지:소프트웨어 및 데이터공학)
6. 강의 경력
 - ① 2023, 24년 : 삼성멀티캠퍼스 KDT(금융&마케팅 데이터 분석) 주강사,
휴먼교육센터 KDT(데이터 분석 기반 인공지능 시스템 개발자 양성 과정) 주강사
 - ② 2022년 11월 ~ 현재 : 메가스터디 IT아카데미

수업 자료 공유

1. 카카오톡 오픈채팅 검색 : 메가IT 데이터분석
2. password : 1234
3. 기본 프로필로만 입장 가능



01. Introduction

커리큘럼

1. python 라이브러리1

- 1) numpy
- 2) pandas

2. 정형 데이터 분석 + python 라이브러리2(시각화)

- 1) 기본 개념
 - ① 데이터 수집
 - ② 데이터 전처리
 - ③ 데이터 탐색
 - ④ 모델링
- 2) 머신 러닝
 - ① 지도 학습을 이용한 분류(classification)
 - 붓꽃 데이터 분석
 - 당뇨병 건강 지표 데이터 분석
 - ② 지도 학습을 이용한 회귀(regression)
 - 보스턴 집값 데이터 분석

3. 비정형 텍스트 데이터 분석

- 1) 기본 개념
- 2) 토큰화(tokenization)
 - ① 문장 토큰화
 - ② 단어 토큰화
 - ③ 한글 형태소 단위 토큰화
- 3) 품사 태깅(tagging)
- 4) 단어 빈도수 분석
- 5) 특성 추출 : 임베딩(Embedding)
 - BoW(Bag of Words)
 - SentenceBERT
- 6) 텍스트 유사도 분석
- 7) 문장 유사도를 이용한 영화 추천 프로그램 개발

4. 개인 프로젝트(11주차~12주차)

- 1) 개인 프로젝트 진행
- 2) 1:1 코칭

데이터 분석 개요

- 개념 : 유용한 정보를 발견하기 위하여 데이터를 정리, 변환, 모델링하는 프로세스
- 목적 : 예측(분류 또는 회귀)을 통한 의사 결정(decision making) 지원
- 분석 과정
 - 1) 문제 정의 : 분석의 목적 설정
 - 2) 데이터 수집 : 분석에 필요한 데이터 확보
 - 3) 데이터 전처리(preprocessing) : 데이터를 분석에 적합한 형태로 만들어서 신뢰도를 높이는 작업
 - 4) 데이터 탐색 : 그래프나 통계적인 방법을 이용, 데이터의 특징과 패턴을 파악
 - 5) 모델링
 - : 분석 목적에 부합하고 수집된 데이터의 특성을 고려하여 적합한 데이터 분석 모형 선정, 분석 실행
 - 6) 분석 모형 평가

데이터 분석 개요

- 데이터의 종류

1) 정형 데이터(Structured Data)

- 구조화된 데이터, 미리 정해진 구조에 따라 저장된 데이터
- 예시) 표 안에서 행과 열에 의해 지정된 각 칸에 데이터를 저장하는 엑셀의 스프레드 시트, 관계형 데이터베이스의 테이블

	A	B	C	D	E	F	G	H
1	일자	요일	시간대	업종	시도	시군구	읍면동	통화건수
2	20180601	금		0 음식점-죽	서울특별시	강남구	논현동	5
3	20180601	금		0 음식점-죽	서울특별시	강동구	길동	5
4	20180601	금		0 음식점-죽	서울특별시	강서구	내발산동	5
5	20180601	금		0 음식점-죽	서울특별시	동대문구	제기동	5
6	20180601	금		0 음식점-죽	서울특별시	서대문구	창천동	7
7	20180601	금		0 음식점-죽	서울특별시	서초구	양재동	5
8	20180601	금		0 음식점-죽	서울특별시	성동구	성수동2가	5
9	20180601	금		0 음식점-죽	서울특별시	성북구	동선동2가	5
10	20180601	금		0 음식점-죽	서울특별시	송파구	송파동	5
11	20180601	금		0 음식점-죽	서울특별시	영등포구	문래동3가	5

2) 비정형 데이터

- 정해진 구조가 없이 저장된 데이터
- 예시) 텍스트, 이미지, 영상 파일

데이터 분석 개요

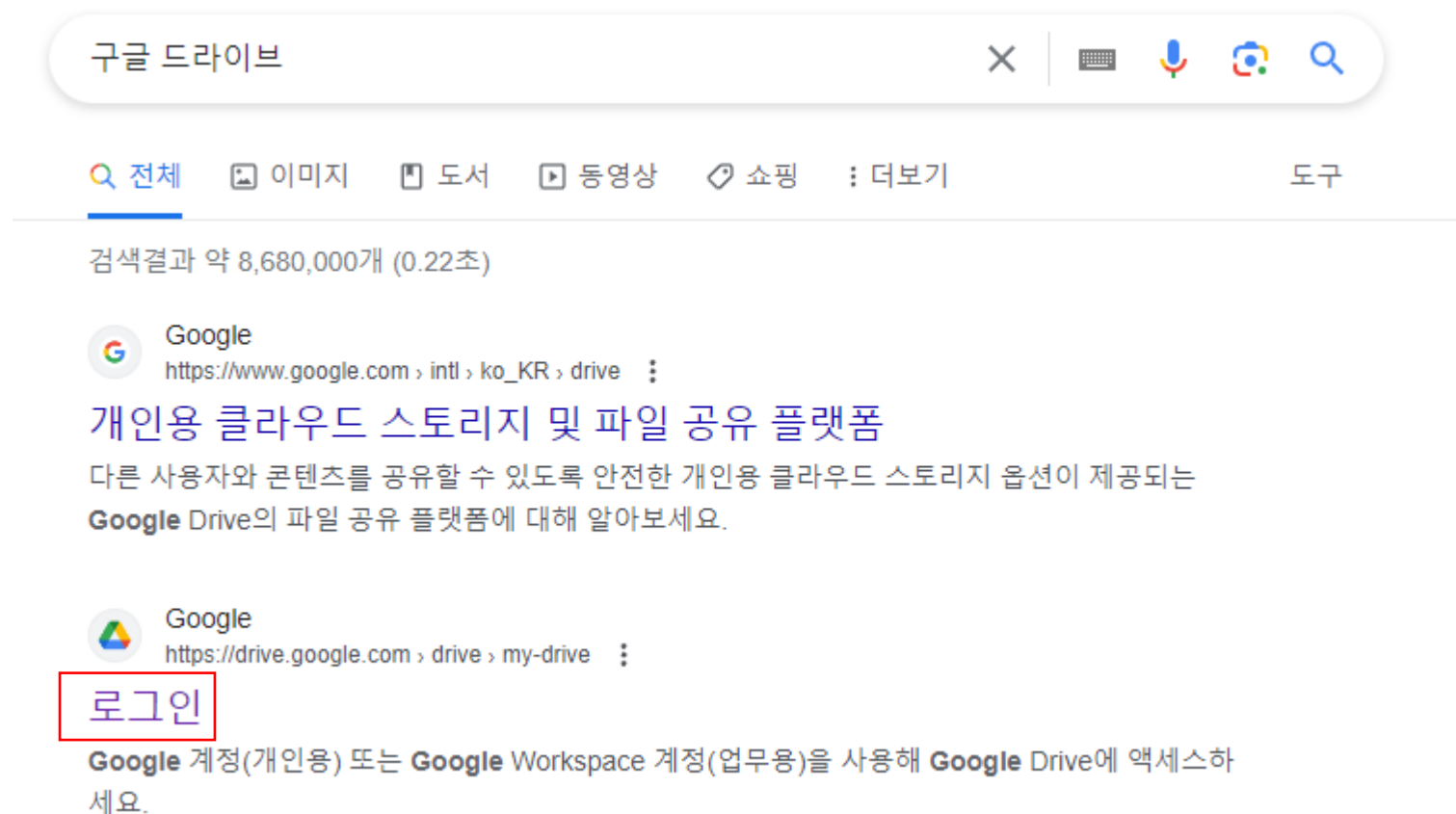
- 필요한 지식

- 1) pandas : 데이터 조작 및 분석을 위한 python 라이브러리
- 2) numpy : 수학적 연산을 지원하는 python 라이브러리
- 3) matplotlib, seaborn : 시각화 라이브러리, 시각화를 통한 데이터에 대한 이해(특징과 패턴 파악)
- 4) 머신 러닝 : 데이터의 특성과 패턴을 학습하여 그 결과를 바탕으로 미지의 데이터에 대해서 값 또는 분포를 예측하는 프로그램
- 5) 딥러닝 : 인공 신경망을 이용해서 데이터의 특성과 패턴을 학습하며 그 결과를 바탕으로 미지의 데이터에 대해서 값 또는 분포를 예측하는 프로그램

02. 실습 환경

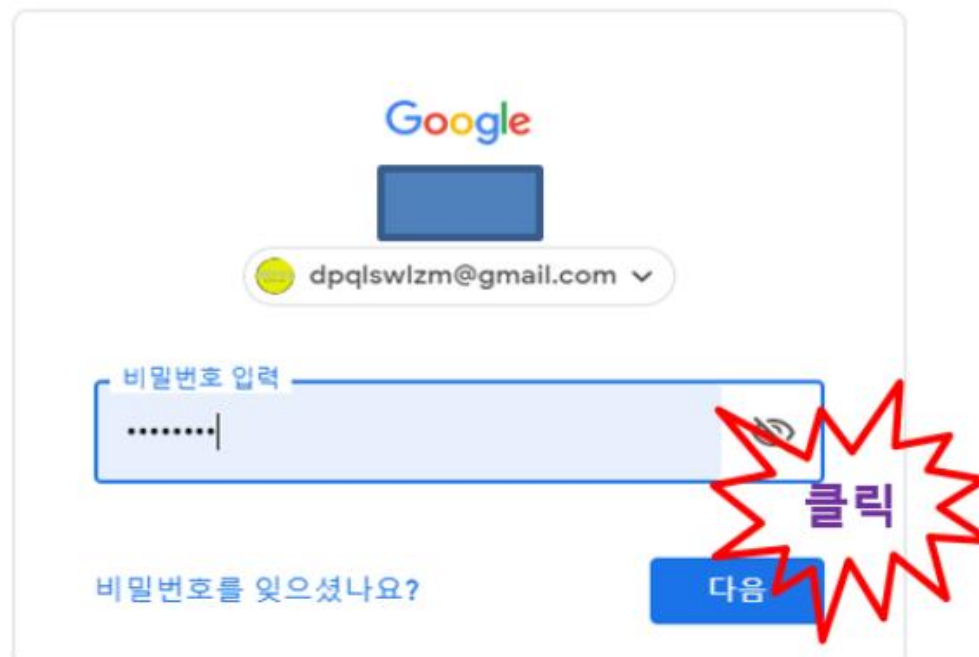
colab 설치 방법

- 구글 가입 / 구글 드라이브 검색 및 로그인



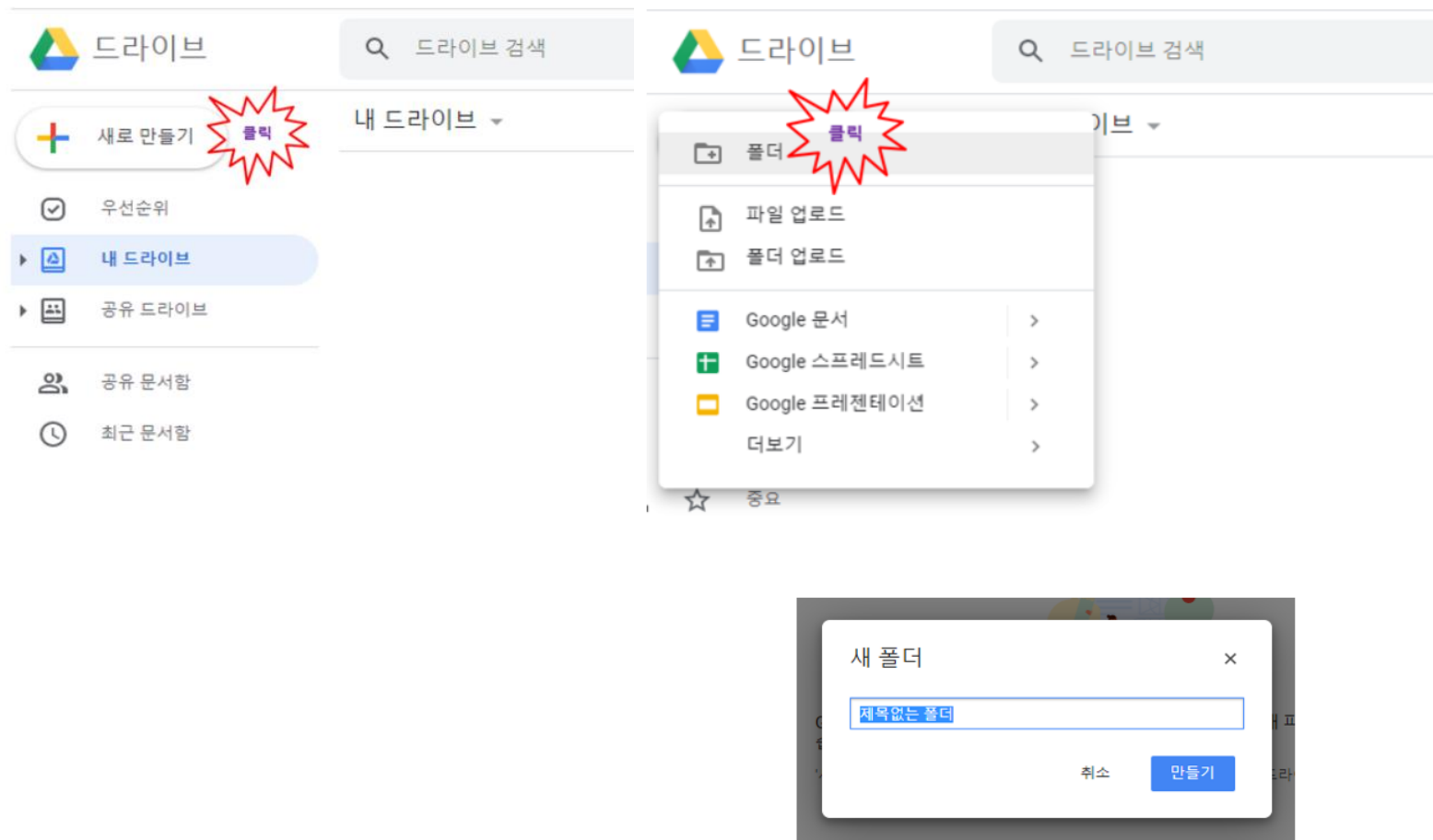
colab 설치 방법

- 본인 계정 선택 후 비밀번호 입력



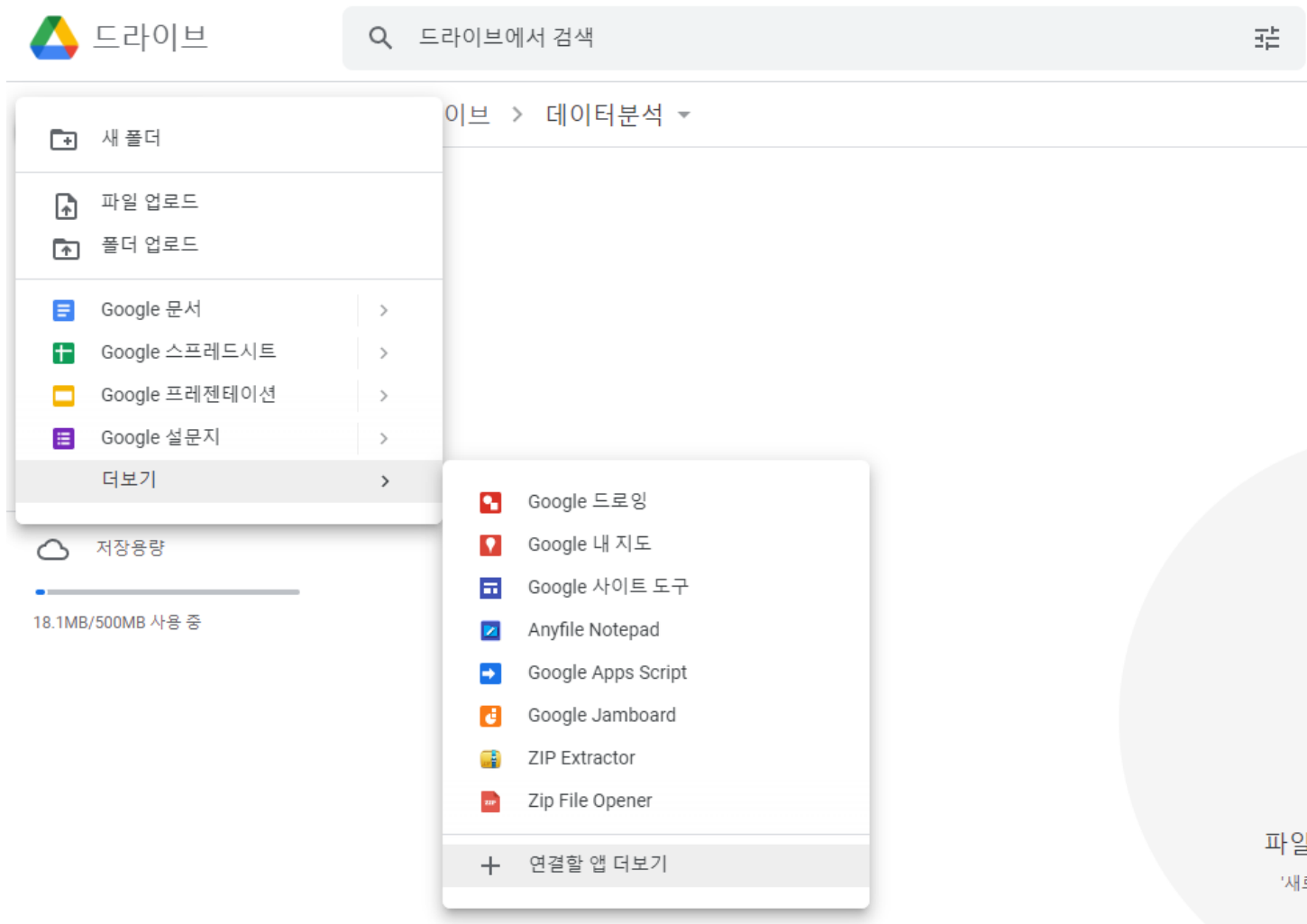
실습 환경 구축하기 - colab 설치 방법

- 구글 드라이브에 새 폴더 만들기 : 새로 만들기 클릭 → 폴더 → 새 폴더 : 데이터분석 입력



실습 환경 구축하기 - colab 설치 방법

- 데이터분석 폴더 클릭 → 새로 만들기 클릭 → 더보기 → 연결할 앱 더보기



실습 환경 구축하기 - colab 설치 방법

- Google Workspace Marketplace → Colaboratory 검색 → Colaboratory 아이콘 클릭

The screenshot shows the Google Workspace Marketplace interface. At the top, there's a search bar with 'Colaboratory' entered. Below the search bar, there are filters for '모든 필터' (All filters), '호환 기기' (Compatible devices), '가격' (Price), and '내부 앱' (Internal apps). The search results are titled '검색결과: Colaboratory' (Search results: Colaboratory). Below the title, there's a note: 'Google에서는 리뷰나 평점을 확인하지 않습니다. [리뷰 및 결과에 대해 자세히 알아보기](#)' (Google does not check reviews or ratings. [Learn more about reviews and results](#)). The results list four apps: 1. Colaboratory by Colaboratory team, described as a tool to open and create files in Google Drive. 2. MAIL MERGE for SHEETS by yamm, described as a mailing tool for Gmail. 3. Form Publisher by Talarian, described as a document generator or document merge solution. 4. Awesome Table - Data ... by Talarian, described as a tool to export data from many sources to Google Sheets. Each app card shows a star rating and a download count of 10,000,000+.

Google Workspace Marketplace

Colaboratory

모든 필터 호환 기기: 가격 내부 앱

검색결과: Colaboratory

Google에서는 리뷰나 평점을 확인하지 않습니다. [리뷰 및 결과에 대해 자세히 알아보기](#)

Colaboratory
Colaboratory team
This allows Google Colaboratory to open and create files in Google Drive. It i...
★ 4.7 * 10,000,000+

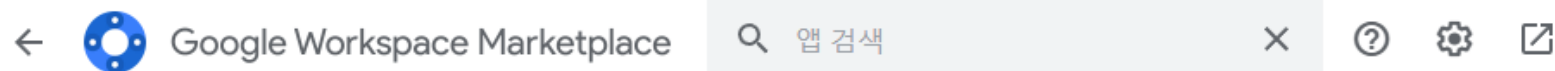
MAIL MERGE for SHEETS
yamm
YAMM - Best mailing tool
Talarian
The best and easiest mail merge tool for Gmail. Send mass emails with high deliverability...
★ 4.7 * 10,000,000+

Form Publisher
Generate Excels
Form Approvals and Do...
Talarian
Form Publisher is a document generator or document merge solution: generate PDF, Googl...
★ 4.6 * 10,000,000+

GET DATA INTO SHEETS
Awesome Table - Data ...
Talarian
Export data from many sources to Google Sheets™. No technical skill required.
★ 4.5 * 10,000,000+

실습 환경 구축하기 - colab 설치 방법

- Colaboratory 설치 버튼 클릭 → 설치 진행



Colaboratory

This allows Google Colaboratory to open and create files in Google Drive. It is automatically installed on first use; uninstalling this will not prevent access to Colaboratory.

개발자: [Colaboratory team](#)
정보 업데이트: 2022년 6월 17일

설치

호환 기기:

★★★★★ 3,483 ⓘ ⬇ 10,000,000+

개요

사용 권한

리뷰



CO 설치 가능

을(를) Colaboratory 설치하려면 권한이 필요합니다.

계속을 클릭하면 이 애플리케이션의 [서비스 약관](#) 및 [개인정보 처리방침](#)에 따라 내 정보가 사용되는 것에 동의하게 됩니다.

취소

계속

Google 계정으로 로그인

hansung.ac.kr 계정 선택

[Google Colaboratory](#)(으)로 이동



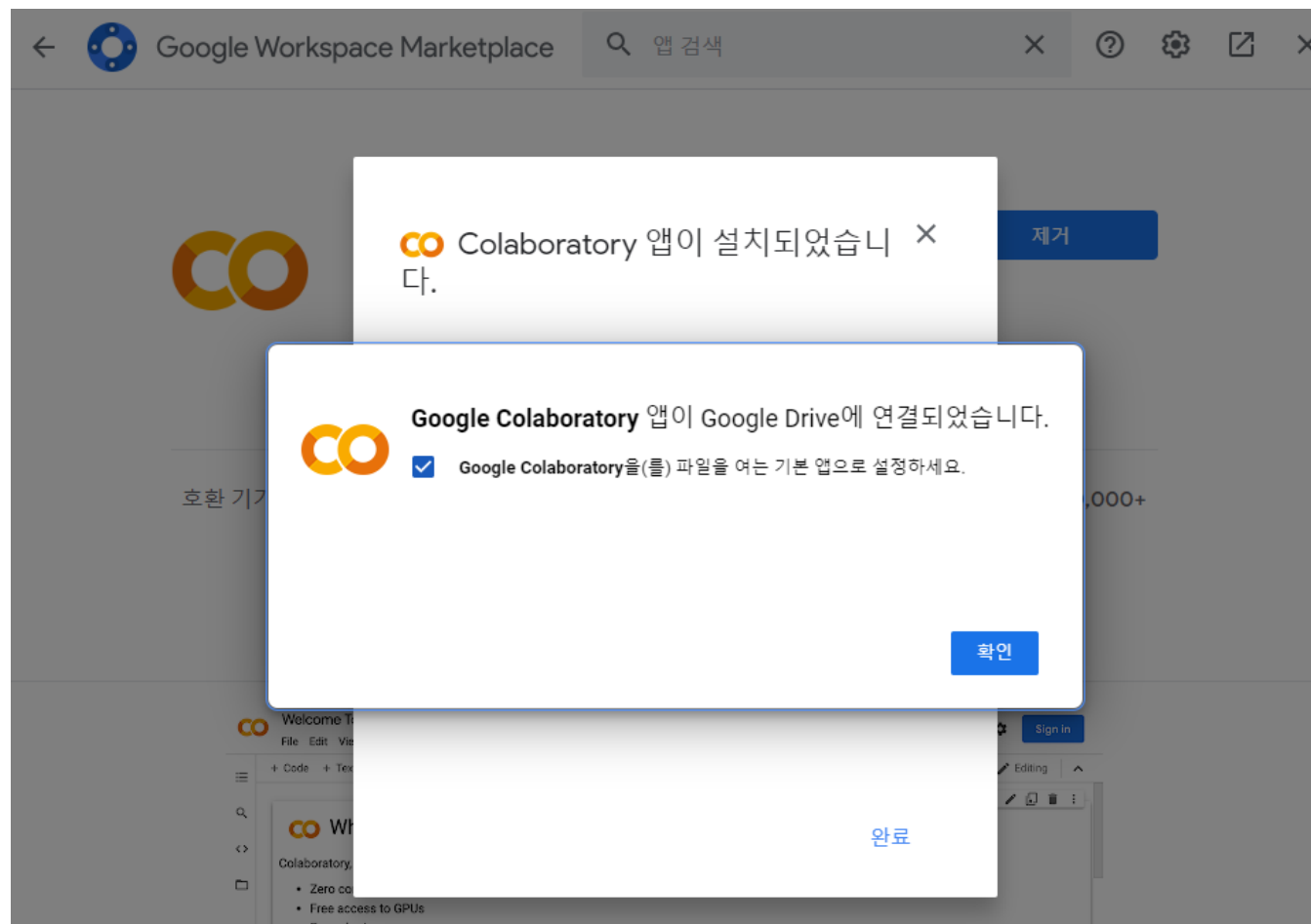
ai.artdirector@hansung.ac.kr



다른 계정 사용

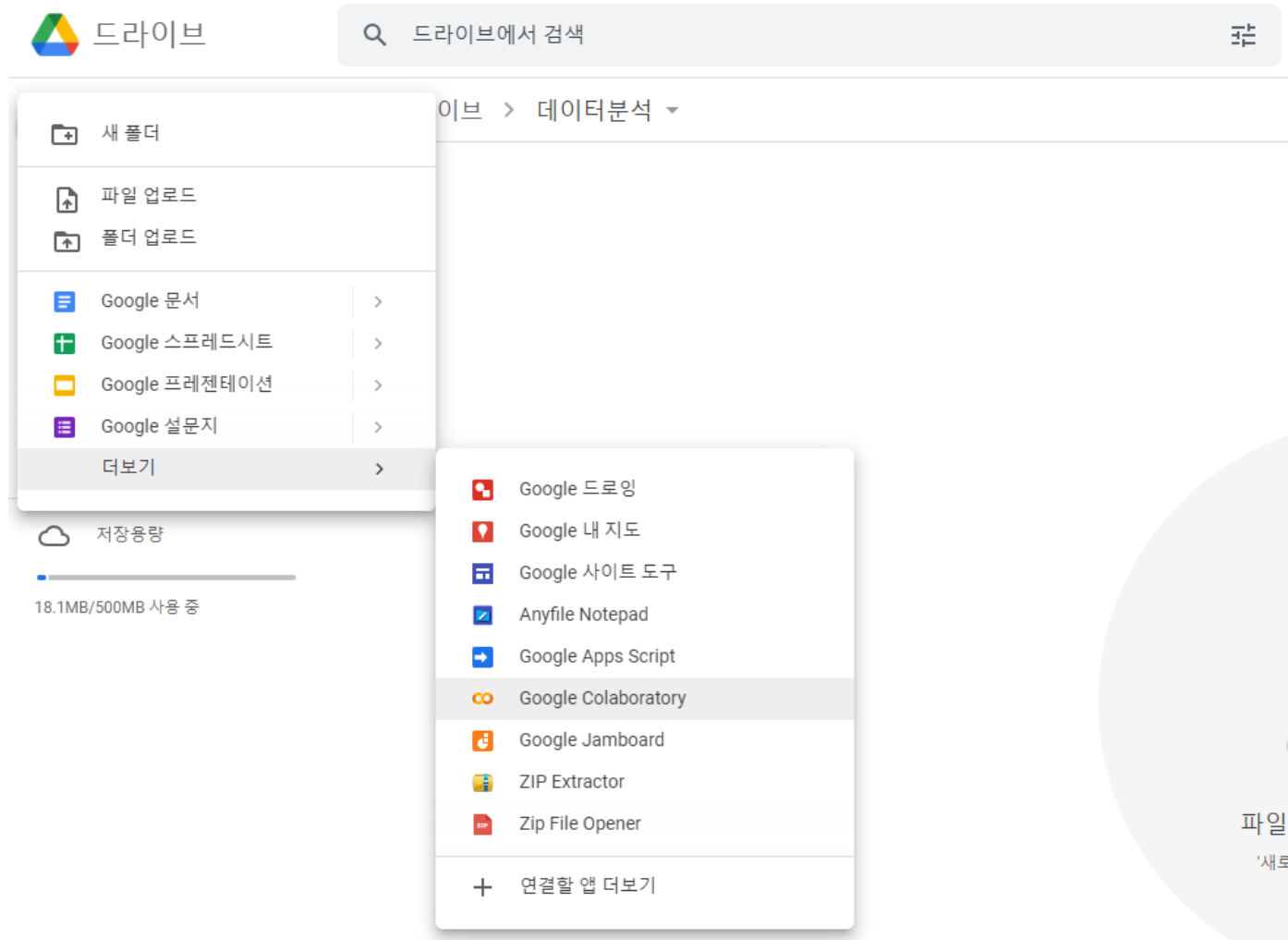
실습 환경 구축하기 - colab 설치 방법

- Colaboratory 설치 완료



실습 환경 구축하기 - colab 설치 방법

- 데이터분석 폴더 : 새로 만들기 클릭 → 더보기 → Google Colaboratory 실행



03. Python 핵심 라이브러리

- numpy

Numpy

- 고성능 수학 계산을 위한 python 라이브러리
 - 데이터 : 정수, 실수, 문자열, 불리언
 - 자료형 : 다차원 배열(ndarray)
- 주요 기능
 - python list 자료형 → numpy 다차원 배열 자료형으로 변환

python list	numpy ndarray
연산 불가능	연산 가능
list1 + list2	arr1 + arr2

- 효율적인 연산이 가능한 수학 함수 (sum(), sqrt(), mean(), ...) 제공

numpy 사용하기

- `import numpy as np`
- numpy 라이브러리를 import하고 앞으로 np라는 이름으로 부른다.

array 생성 하기 : 1차원

- 변수 = np.array(data)
- data = 1차원

```
list1 = [1,2,3,4,5]
```

```
list1
```

```
[1, 2, 3, 4, 5]
```

```
arr = np.array(list1)
```

```
arr
```

```
array([1, 2, 3, 4, 5])
```

or

```
arr1 = np.array([1,2,3,4,5])
```

```
arr1
```

```
array([1, 2, 3, 4, 5])
```

array 생성 하기 : 1차원

- **np.arange()**
- np.arange(시작점(생략 시 0), 끝점(미포함), step size(생략 시 1))
- 예시

```
np.arange(10)
```

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
np.arange(1, 15, 2)
```

```
array([ 1,  3,  5,  7,  9, 11, 13])
```

연습 문제 : 1차원 array 생성 하기

np.arange()를 사용하여,

1부터 50까지의 정수가 담긴 1차원 배열 생성하기

array 생성 하기 : 2차원

- 변수 = np.array(data)
- data = 2차원

```
1 arr2 = np.array([[1,2,3],[4,5,6]])  
2 arr2  
  
array([[1, 2, 3],  
       [4, 5, 6]])
```


array 생성 하기 : 2차원

- `np.arange().reshape()` 사용
 - 1) 기능 : 현재 배열의 차원을 변경하여 사용자가 원하는 새로운 차원으로 변환
 - 2) 매개변수 : `newshape = (행, 열)`, 튜플 형식
- `np.arange(start, stop, step).reshape((행, 열))`

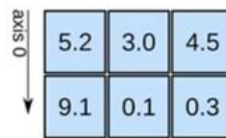
배열의 차원과 모양 확인하기

1D array



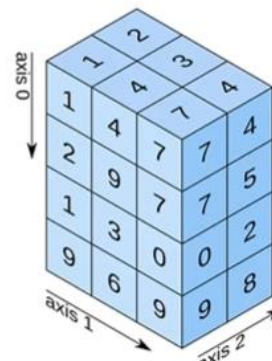
shape: (4,)

2D array



shape: (2, 3)

3D array



shape: (4, 3, 2)

배열의 모양 확인하기

- `array.shape`

```
arr1
```

```
array([1, 2, 3, 4, 5])
```

```
arr1.shape
```

```
(5,)
```

```
arr2
```

```
array([[1, 2, 3],  
       [4, 5, 6]])
```

```
arr2.shape
```

```
(2, 3)
```

배열의 전체 원소 개수 확인하기

- `array.size`

```
print(arr1)  
print(arr1.size)
```

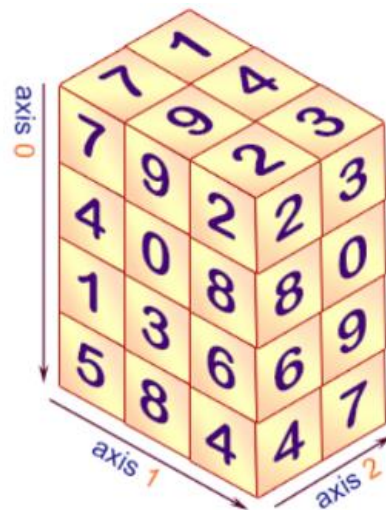
```
[1 2 3 4 5]  
5
```

```
print(arr2)  
print(arr2.size)
```

```
[[1 2 3]  
 [4 5 6]]  
6
```

array 생성 하기 : 3차원

- `np.arange().reshape()` 사용
- `np.arange()` : 1차원 배열 생성
- `np.arange().reshape(newshape=(행, 열, 깊이))`



shape : (4, 3, 2)

array 인덱싱

```
1 arr = np.array([[1,2,3],[4,5,6]])  
2 arr
```

```
array([[1, 2, 3],  
       [4, 5, 6]])
```

```
1 print(arr[0])
```

```
[1 2 3]
```

```
1 print(arr[0][0])
```

```
1
```

```
1 print(arr[0,0])
```

```
1
```

1차원 array 슬라이싱

```
arr1 = np.arange(10)
arr1
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
arr1[3:8]
```

```
array([3, 4, 5, 6, 7])
```

```
arr1[3:8] = 12
```

```
arr1
array([ 0,  1,  2, 12, 12, 12, 12, 12,  8,  9])
```

2차원 array 슬라이싱

0부터 49까지 50개의 정수가 담긴 (5, 10) 모양의 2차원 배열 생성하기

```
[[ 0  1  2  3  4  5  6  7  8  9]
 [10 11 12 13 14 15 16 17 18 19]
 [20 21 22 23 24 25 26 27 28 29]
 [30 31 32 33 34 35 36 37 38 39]
 [40 41 42 43 44 45 46 47 48 49]]
```



```
arr2[:2,:]
```

```
array([[ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14, 15, 16, 17, 18, 19]])
```

```
arr2[:,0]
```

```
array([ 0, 10, 20, 30, 40])
```


04. Python 핵심 라이브러리

- pandas

Pandas

1. 데이터 조작 및 분석을 위한 파이썬 라이브러리
 - 데이터 : 행과 열로 구성된 표 형식의 데이터(예: excel file)
2. 자료형(type)
 - 1) Series : 인덱스(index) + 값(column 1개)
 - 2) DataFrame : DataFrame → 인덱스(index) + 값(column 여러 개)

	A	B	C	D	E
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					

Series

	A	B	C	D	E
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					

DataFrame

Pandas 사용하기

- `import pandas as pd`
- Pandas 라이브러리를 import하고 앞으로 pd라는 이름으로 부른다

Series 생성(1)

- 변수 = `pd.Series(data)`
- data = 리스트 자료형
- `pop_data = [9904312, 3448737, 2890451, 2466052]`
- `population = pd.Series(data)`

```
0      9904312
1      3448737
2      2890451
3      2466052
dtype: int64
```

Series 생성(1)

문자열 인덱스 추가

- `pop_data = [9904312, 3448737, 2890451, 2466052]`
- `pop_index=['서울','부산','인천','대구']`
- `population = pd.Series(data=pop_data, index=pop_index)`

```
서울      9904312  
부산      3448737  
인천      2890451  
대구      2466052  
dtype: int64
```

Series 생성(2)

- 변수 = `pd.Series(data)`
- data = dict 자료형
 - {key : value}, key : Series의 index, value : Series의 value
 - data = {"서울": 9904312, "부산": 3448737, "인천": 2890451, "대구": 2466052}
 - population = `pd.Series(data)`

```
서울      9904312
부산      3448737
인천      2890451
대구      2466052
dtype: int64
```

Series 정보 확인

1. Series 값

- `population.values`

```
[9904312 3448737 2890451 2466052]
```

2. Series 인덱스

- `population.index`

```
Index(['서울', '부산', '인천', '대구'], dtype='object')
```

3. Series 타입

- `population.dtype`

```
int64
```

Series 인덱싱(indexing)

- 개념 : 인덱스를 이용하여 Series로부터 특정한 데이터를 추출하는 것
- 방법
 - series변수.iloc[정수 인덱스]
 - series변수.loc[문자 인덱스]

Series 인덱싱(indexing)

- `population.iloc[1], population.loc["부산"]`
- `population.iloc[3], population.loc["대구"]`
- `population.iloc[[0,3,1]]`
- `population.loc[["서울", "대구", "부산"]]`

```
(3448737, 3448737)
```

```
(2466052, 2466052)
```

```
도시
서울    9904312
대구    2466052
부산    3448737
Name: 인구, dtype: int64
```

Series 슬라이싱

- 개념 : Series로부터 인덱스의 범위(from:to)를 지정하여 데이터를 추출하는 것
- 방법
 - series변수.iloc[시작 정수 인덱스 : 종료 정수 인덱스]
 - series변수.loc[시작 문자 인덱스 : 종료 문자 인덱스]

Series 슬라이싱

➤ `population.iloc[1 : 3]`

```
도시  
부산      3448737  
인천      2890451  
Name: 인구, dtype: int64
```

➤ `population.loc["부산" : "대구"]`

```
도시  
부산      3448737  
인천      2890451  
대구      2466052  
Name: 인구, dtype: int64
```

DataFrame 생성(1)

- `df = pd.DataFrame(data)`
- `data = dict` 자료형
 - `{key : value}`, `key` : column의 name, `value` : column의 value
 - `data = {"2022": [9436836, 3317812, 2964820, 2365619]`
`"2015": [9904312, 3448737, 2890451, 2466052],`
`"2010": [9631482, 3393191, 2632035, 2431774]}`
 - `df = pd.DataFrame(data)`

	2022	2015	2010
0	9436836	9904312	9631482
1	3317812	3448737	3393191
2	2964820	2890451	2632035
3	2365619	2466052	2431774

DataFrame 정보 확인

1. DataFrame 컬럼

➤ df.columns

```
Index(['2022', '2015', '2010'], dtype='object')
```

2. DataFrame 값

➤ df.values

```
[[9436836 9904312 9631482]  
 [3317812 3448737 3393191]  
 [2964820 2890451 2632035]  
 [2365619 2466052 2431774]]
```

3. DataFrame 인덱스

➤ df.index

```
RangeIndex(start=0, stop=4, step=1)
```

DataFrame 생성(1)

DataFrame 인덱스 수정

➤ `df.index = ["서울", "부산", "인천", "대구"]`

	2022	2015	2010
서울	9436836	9904312	9631482
부산	3317812	3448737	3393191
인천	2964820	2890451	2632035
대구	2365619	2466052	2431774

인덱싱(indexing)

- 개념 : 인덱스(index)를 이용하여 데이터프레임으로부터 특정한 데이터를 추출하는 것
- 특정 컬럼 인덱싱
 - `df변수.loc[:, "컬럼 이름"]` \Rightarrow `df.loc[:, '2005']`
 - `df변수.iloc[:, 정수 인덱스]` \Rightarrow `df.iloc[:, 2]`

인덱싱(indexing)

- 개념 : 인덱스(index)를 이용하여 데이터프레임으로부터 특정한 데이터를 추출하는 것
- 특정 행 인덱싱
 - `df변수.loc["행 이름", :]` \Rightarrow `df.loc['부산', :]`
 - `df변수.iloc[정수 인덱스, :]` \Rightarrow `df.iloc[1, :]`

인덱싱(indexing)

- 개념 : 인덱스(index)를 이용하여 데이터프레임으로부터 특정한 데이터를 추출하는 것
- fancy 인덱싱 : 행 / 컬럼 → 리스트 형태 → 여러 개의 행 / 컬럼을 동시에 추출
 - `df변수.loc[:, ['컬럼 이름1', '컬럼 이름2',...]] ⇒ df.loc[:, [' 2015 ' , ' 2005 ']]`
 - `df변수.iloc[:, [정수 인덱스1, 정수 인덱스2,...]] ⇒ df.iloc[:, [0, 2]]`
 - `df변수.loc [['행 이름1','행 이름2',...], :] ⇒ df.loc[['서울','인천'], :]`
 - `df변수.iloc[[정수 인덱스1, 정수 인덱스2,...], :] ⇒ df.iloc[[0, 2], :]`

슬라이싱(slicing)

- 개념 : 데이터프레임으로부터 인덱스의 범위(from:to)를 지정하여 데이터를 추출하는 것
- 컬럼 슬라이싱
 - `df변수.loc[:, "시작 컬럼 이름:종료 컬럼 이름"]` \Rightarrow `df.loc[:, '2015':'2010']`
 - `df변수.iloc[:, 시작 정수 인덱스: 종료 정수 인덱스]` \Rightarrow `df.iloc[:, 0:2]`
- 행 슬라이싱
 - `df변수.loc["시작 행 이름:종료 행 이름", :]` \Rightarrow `df.loc['서울':'인천', :]`
 - `df변수.iloc[시작 정수 인덱스: 종료 정수 인덱스, :]` \Rightarrow `df.iloc[0:3, :]`

정리 : df.loc[행, 열]

- 문자열 인덱스를 이용하여 자유롭게 인덱싱 / 슬라이싱 할 수 있는 기능

1) df.loc["서울" , "2022":"2010"]

2022	9436836
2015	9904312
2010	9631482

Name: 서울, dtype: int64

2) df.loc["서울":"부산 " , "2015":"2010"]

	2015	2010
서울	9904312	9631482
부산	3448737	3393191

정리 : df.loc[행, 열]

- 문자열 인덱스를 이용하여 자유롭게 인덱싱 / 슬라이싱 할 수 있는 기능

4) 행 : 특정 값, 열: 전체

: df.loc['부산':'인천', **열(전체)**] → df.loc['부산':'인천', :]

	2022	2015	2010
부산	9436836	3448737	3393191
인천	2964820	2890451	2632035

5) 행 : 전체, 열 : 특정 컬럼

: df.loc[**행(전체)**, '2015'] → df.loc[:, '2015']

서울	9904312
부산	3448737
인천	2890451
대구	2466052
Name: 2015, dtype: int64	

정리 : df.iloc[행, 열]

- 정수(integer) 인덱스를 이용하여 자유롭게 인덱싱 / 슬라이싱 할 수 있는 기능
 - 행 : 첫번째부터 두번째 행까지, 열 : 두번째부터 끝까지 가져오기

df.iloc[0:2, 1:]

	2015	2010
서울	9904312	9631482
부산	3448737	3393191

새로운 컬럼 생성

“2005”라는 컬럼명으로 2005년 인구수 삽입

➤ `df.loc[:, "2005"] = [9762546, 3512547, 2517680, 2456016]`

	2022	2015	2010	2005
서울	9436836	9904312	9631482	9762546
부산	3317812	3448737	3393191	3512547
인천	2964820	2890451	2632035	2517680
대구	2365619	2466052	2431774	2456016

Boolean 인덱싱

- 개념 : 비교 연산자 또는 논리 연산자의 결과를 이용하여, 조건을 만족하는 데이터를 동시에 추출할 수 있는 기능
- 비교 연산자
 - 1) 두 변수의 크기를 비교할 수 있는 도구
 - 2) 결과 값 : 참(True), 거짓(False)

종류	설명
>	왼쪽 값이 크다
<	왼쪽 값이 작다
>=	왼쪽 값이 크거나 같다
<=	왼쪽 값이 작거나 같다
==	값이 같다
!=	값이 같지 않다

Boolean 인덱싱

- 논리 연산자

- 1) 두 가지 이상의 조건(비교 연산)을 동시에 처리할 때 사용하는 도구
- 2) 종류 : &(논리곱) , |(논리합), ~(논리부정)

논리 연산자	문법	설명
and	a and b	AND(논리곱), 양쪽 모두 참일 때 참
or	a or b	OR((논리합), 양쪽 중 한쪽만 참이라도 참
not	not x	NOT(논리부정), 참과 거짓을 뒤집음

Boolean 인덱싱

- 비교 연산자를 이용하여 컬럼(Series)에 대한 boolean indexing

- `df.loc[:, col].loc[비교 연산자] → df.loc[:, '2015'].loc[df.loc[:, '2015'] >= 3000000]`

Seoul	9904312
Busan	3448737

- 논리 연산자를 이용하여 전체 데이터프레임에 대한 boolean indexing

- `df.loc[논리 연산자, 열]`

→ `df.loc[(df.loc[:, '2015'] >= 3000000) & (df.loc[:, '2010'] >= 4000000), :]`

	2022	2015	2010	2005
Seoul	9436836	9904312	9631482	9762546

DataFrame 생성(2)

- `df = pd.DataFrame(data)`
- `data = list 자료형`
- `pop_list = [[9436836, 3317812, 2964820, 2365619],
[9904312, 3448737, 2890451, 2466052],
[9631482, 3393191, 2632035, 2431774]]`
- `pop_index = ["2022", "2015", "2010"]`
- `pop_columns = ["서울", "부산", "인천", "대구"]`
- `df = pd.DataFrame(data=pop_list, index=pop_index, columns=pop_columns)`

	서울	부산	인천	대구
2022	9436836	3317812	2964820	2365619
2015	9904312	3448737	2890451	2455052
2010	9632482	3393191	2632035	2431774

read_csv() : csv 파일 처리

```
df = pd.read_csv(file_path, encoding = 'cp949', index_col=0)
```

	sentence	label
0	본 발명은 2종 이상의 생약 추출물을 유효성분으로 포함하는, 면역력 증진용 조성물 ...	1
1	본 발명은 타히보 추출물 및 팔각회향을 유효성분으로 포함하는 면역 증진용 조성물에 ...	1
2	본 출원은 신규한 터키 겨우살이의 추출물, 렉틴 및 상기 추출물 또는 렉틴의 제조방...	1
3	아로니아(Aronia) 열매 추출물, 탕덩이나무(Lonicera caerulea) ...	1
4	본 발명의 청각 추출물 및 홍삼 추출물을 포함하는 선천성 면역 증진용 조성물에 관한...	1
...
145	본 발명은 엉겅퀴(Cirsium japonicum var. ussuriense) 잎...	1
146	본 발명은 어성초 초임계 추출물을 유효성분으로 함유하는 관절염 질환의 예방 및 치료...	1
147	본 발명은 항염증 활성과 파골세포 억제 효과를 갖는 두충 추출물, 상기 추출물을 유...	1
148	본 발명은 유효성분으로서 다음과 같은 진세노사이드 함량을 나타내는 홍삼 발효액을 포...	1
149	본 발명은 전통 한의약 이론에 근거하여 한방의 복합 생약추출물에 발효기술을 이용하여...	1

150 rows × 2 columns

DataFrame.info()

- 데이터에 대한 전반적인 정보 표시
- df를 구성하는 행과 열의 크기, 컬럼명, 컬럼을 구성하는 값의 자료형, 누락 데이터 정보 제공

```
df = pd.read_csv(file_path, encoding = 'cp949', index_col = 0)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 150 entries, 0 to 149
Data columns (total 2 columns):
#   Column      Non-Null Count  Dtype
---  -
0   sentence    147 non-null    object
1   label       150 non-null    int64
dtypes: int64(1), object(1)
memory usage: 3.5+ KB
```

누락 데이터 확인

	sentence	label
10	본 발명은 동과씨 추출물을 유효성분으로 포함하는 면역 증진용 식품 조성물에 관한 것이다.	1
11	본 발명은 전처리한 우렁헝이(<i>Halocynthia roretzi</i>) 껍질에 물 및 단...	1
12	NaN	1
13	본 발명은 2종 이상의 생약 추출물을 유효성분으로 포함하는, 면역력 증진용 조성물 ...	1
14	본 발명은 비박피 양파를 숙성 및 발효하여 섭취자의 면역력을 증강시키면서도 양파 섭...	1
15	본 발명은 동과씨 추출물을 유효성분으로 포함하는 면역 증진용 식품 조성물에 관한 것이다.	1

누락 데이터 확인

- 사용 함수 : `df.sentence.isnull().sum()`
- 데이터 프레임으로부터 특정 컬럼 인덱싱하는 방법
 - ① `df.loc[:, '컬럼이름']`
 - ② `df.컬럼이름`
- `isnull()` : 결측치를 확인해서 bool 형식으로 전달하는 메서드
- `sum()` : 값의 총합을 구하는 메서드

누락 데이터 처리(1) - 데이터를 삭제해도 되는 경우

- `df.dropna(inplace)`
 - ① 행 또는 열에 누락 데이터(missing value, NaN)가 1개만 있어도 해당 행 제거
 - ② `inplace=True` : 현재 데이터프레임 수정

누락 데이터 처리(1) - 데이터를 삭제해도 되는 경우

- `reset_index()` : 누락 데이터 제거 후 남은 데이터의 행 인덱스 재정렬
- `df.reset_index(drop, inplace)`

누락 데이터 처리(2) - 데이터를 삭제하면 안되는 경우

- 내용이 없는 데이터(NaN) 대신에 적당한 데이터 삽입
- `df_cleaned = df.fillna("내용 없음")`

누락 데이터 처리 확인

df_cleaned.info()

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 150 entries, 0 to 149
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype
---  -
 0   sentence    150 non-null    object
 1   label       150 non-null    int64
dtypes: int64(1), object(1)
memory usage: 3.5+ KB
```

df_cleaned.iloc[11:13, :]

	sentence	label
11	본 발명은 전처리한 우렁쟁이(<i>Halocynthia roretzi</i>) 껍질에 물 및 단...	1
12	내용 없음	1

value_counts()

- 컬럼(column) : 데이터 → 항목별 개수를 알려준다
- label 컬럼의 항목별 개수 구하기

```
df_cleaned.label.value_counts()
```

concat()

- 데이터프레임들을 병합한다

concat 데이터 생성

```
df1 = pd.DataFrame({'A': ['A0', 'A1', 'A2', 'A3'],  
                    'B': ['B0', 'B1', 'B2', 'B3'],  
                    'C': ['C0', 'C1', 'C2', 'C3'],  
                    'D': ['D0', 'D1', 'D2', 'D3']},  
                    index=[0, 1, 2, 3])  
  
df2 = pd.DataFrame({'A': ['A4', 'A5', 'A6', 'A7'],  
                    'B': ['B4', 'B5', 'B6', 'B7'],  
                    'C': ['C4', 'C5', 'C6', 'C7'],  
                    'D': ['D4', 'D5', 'D6', 'D7']},  
                    index=[4, 5, 6, 7])  
  
df3 = pd.DataFrame({'A': ['A8', 'A9', 'A10', 'A11'],  
                    'B': ['B8', 'B9', 'B10', 'B11'],  
                    'C': ['C8', 'C9', 'C10', 'C11'],  
                    'D': ['D8', 'D9', 'D10', 'D11']},  
                    index=[8, 9, 10, 11])
```



df1				
	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3

df2				
	A	B	C	D
4	A4	B4	C4	D4
5	A5	B5	C5	D5
6	A6	B6	C6	D6
7	A7	B7	C7	D7

df3				
	A	B	C	D
8	A8	B8	C8	D8
9	A9	B9	C9	D9
10	A10	B10	C10	D10
11	A11	B11	C11	D11

concat 메서드 사용 1, axis=0번 방향

```
result1 = pd.concat([df1, df2, df3], axis=0)  
result1|
```

axis=0



“공통된 컬럼을 기준으로,
y축 방향으로 병합”



	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3
4	A4	B4	C4	D4
5	A5	B5	C5	D5
6	A6	B6	C6	D6
7	A7	B7	C7	D7
8	A8	B8	C8	D8
9	A9	B9	C9	D9
10	A10	B10	C10	D10
11	A11	B11	C11	D11

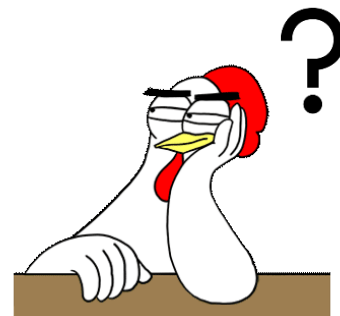
concat 메서드 사용 2, axis=1 번 방향

```
result2 = pd.concat([df1, df2, df3], axis=1)  
result2
```

axis=1



“서로 다른 컬럼들을 x축 방향으로 병합”



05. 정형 데이터 분석 및 시각화

- 기본 개념

데이터 수집

- 분석에 필요한 데이터 수집 방법

1) 공공 데이터

① 공공 기관 사이트 : 공공데이터포털, AI Hub , 도서관 정보나루 등

② 데이터 형태 : CSV file, 오픈 API를 통한 JSON file, XML file

전체(77,593건)

파일데이터(57,542건)

오픈 API(10,841건)

표준데이터셋169개(9,210건)

수정일자순

10개씩

정렬

파일데이터 (57,542건)

공공행정

자치행정기관

미리보기

CSV

서울특별시 강남구_코로나19 확진자 현황

서울특별시 강남구의 2020-02-26 부터 2022-12-31 까지 코로나 확진자 현황데이터 입니다. 기타 자세한 사항은 서울특별시 강남구 질병관리과(02-3423-7110)으로 주시면 자세히 안내...

제공기관 서울특별시 강남구 수정일 2023-01-30 조회수 2 다운로드 0 키워드 코로나,확진자,현황

다운로드

전체(77,593건)

파일데이터(57,542건)

오픈 API(10,841건)

표준데이터셋169개(9,210건)

수정일자순

10개씩

정렬

오픈 API (10,841건)

교통운류

자치행정기관

미리보기

JSON

경상남도 남해군_횡단보도 조회 서비스

경상남도 남해군 관내 횡단보도의 횡단보도관리번호, 횡단보도종류, 자전거횡단도검용여부, 고원식적용여부, 위도, 경도, 차로수, 횡단보도폭, 횡단보도연장, 보행자신호등유무, 보행자작...

제공기관 경상남도 남해군 수정일 2023-01-16 조회수 45 활용신청 2 키워드 도로,교통,횡단보도

활용신청

2) 기업 데이터 : DB, excel file

3) 크롤링(스크래핑)을 통한 웹 데이터 수집 : 무단으로 수집함으로써 문제가 발생할 수 있음

CSV file 형식의 데이터 수집

- 공공 기관 또는 각종 사이트(kaggle)에서 제공하는 csv file을 다운로드해서 사용하면 됨

도서관 정보나루
LIBRARY BIGDATA

데이터 제공 데이터 신청 데이터 활용 도서관 빅데이터 소개 알림소통

장서/대출데이터 도서관별로 공개된 장서/대출 데이터를 보실 수 있습니다. 데이터 제공 > 장서/대출데이터

참여 도서관 목록 **장서/대출데이터** 인기대출도서 도서별 이용분석 대출 급상승 도서 지역별 비교분석 이달의 키워드

지역 **전체** ▼ 세부지역 **전체** ▼ 도서관유형 **전체** ▼ 도서관명 **도서관 선택** ▼ 데이터 검색 🔍

도서관명	데이터 유형	공개범위	최근 데이터 제공일
2.28도서관 장서/대출 데이터	Text Excel API	장서/대출	2023-02-01
U보라작은도서관 장서/대출 데이터	Text Excel API	장서/대출	2023-02-01
가락물도서관 장서/대출 데이터	Text Excel API	장서/대출	2023-02-01
가수원도서관 장서/대출 데이터	Text Excel API	장서/대출	2023-02-01
가슴따뜻한작은도서관 장서/대출 데이터	Text Excel API	장서/대출	2023-02-01

Data Dictionary

[View more](#)

gender_submission.csv (3.26 kB)

[Detail](#) [Compact](#) [Column](#)

2 of 2 columns ▼

About this file

An example of what a submission file should look like.

These predictions assume only female passengers survive.

PassengerId	# Survived
892	0
893	1

Data Explorer

93.08 kB

- [gender_submission.csv](#)
- [test.csv](#)
- [train.csv](#)

CSV file 형식의 데이터 수집 실습

- csv file을 pandas dataframe 자료형으로 변환하기

▶ pd.read_csv() 함수 사용

```
# import pandas as pd
```

```
# df = pd.read_csv(file_path)
```

데이터 전처리 (preprocessing)

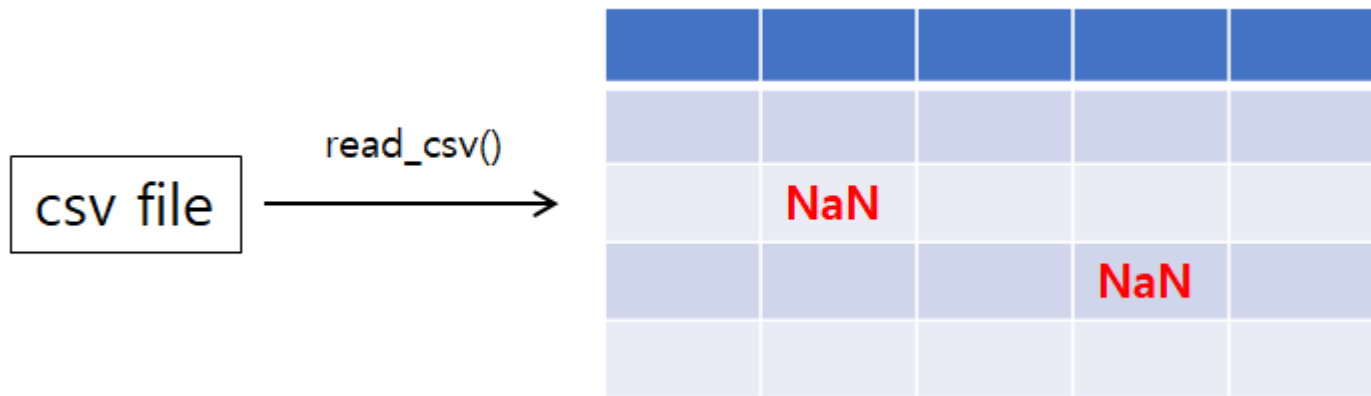
1. 특정 행 추출하기

- ▶ 도서관 정보나루 남산도서관 대출 데이터 활용
- ▶ loc 속성 변수와 불리언 배열 이용

데이터 전처리 (preprocessing)

2. 누락 데이터 처리하기

- ▶ 누락 데이터(missing value) : 값이 누락된 데이터



데이터 전처리 (preprocessing)

2. 누락 데이터 처리하기

- ▶ 도서관 정보나루 남산도서관 대출 데이터 활용
- ▶ 주요 내용
 - 누락된 데이터 개수 확인 : `isnull().sum()`
 - 누락된 데이터 삭제하기 : `drop()` , `dropna()`
 - 누락된 데이터 바꾸기 : `fillna()`, `replace()`

데이터 전처리 (preprocessing)

2. 누락 데이터 처리하기

▶ 누락된 데이터 개수 확인

- isnull()

- ① dataframe 자료형이나 series 자료형에 누락 데이터가 존재하는지 확인

- ② 결과값 : True or False

- isnull().sum() : isnull() 함수의 결과값에서 True 개수를 합산, 데이터에 존재하는 누락 데이터의 총 수를 확인

데이터 전처리 (preprocessing)

2. 누락 데이터 처리하기

▶ 누락된 데이터 삭제하기

- `df.col.drop(index, inplace)`

- ① 컬럼 → 누락된 행의 인덱스 추출
- ② `index`=누락된 행의 인덱스
- ③ `inplace=True` : 자동 저장 기능 활성화

데이터 전처리 (preprocessing)

2. 누락 데이터 처리하기

▶ 누락된 데이터 삭제하기

- `dropna(axis, subset, inplace)`

- ① `axis=0` : 모든 컬럼에 대해서 누락된 데이터(missing value)가 1개만 존재해도 해당 행 제거
- ② `subset=[col1, ...]` : 특정 컬럼에 대해서 누락된 데이터(missing value)가 존재하는 행 제거
- ③ `axis=1` : 누락된 데이터(missing value)가 1개만 존재해도 해당 컬럼 제거
- ④ `inplace=True` : 자동 저장 기능 활성화

데이터 전처리 (preprocessing)

2. 누락 데이터 처리하기

▶ 누락된 데이터 변경하기

- fillna()
- replace()

데이터 전처리 (preprocessing)

2. 누락 데이터 처리하기

▶ fillna(value, inplace)

- value : 입력하길 원하는 값, df.fillna(0), df.fillna("없음"), df.fillna({'col1':'A','col2':df.col2.mean()})
- inplace=True : 자동 저장 기능 활성화

데이터 전처리 (preprocessing)

2. 누락 데이터 처리하기

▶ `replace(to_replace, value, inplace)`

- NaN은 물론, 어떤 값도 바꿀 수 있는 편리한 함수
- `to_replace` : 원래 값(old)
- `value` : 새로운 값(new), `df.replace(np.nan, "없음")`
- 바꾸려는 데이터가 여러 개일 경우 : `df.replace({np.nan:'없음', '2021':'21'})`
- 컬럼마다 다른 값으로 바꾸는 경우 : `df.replace({열 이름 : {원래 값1:새로운 값1}, ...})`

데이터 전처리 (preprocessing)

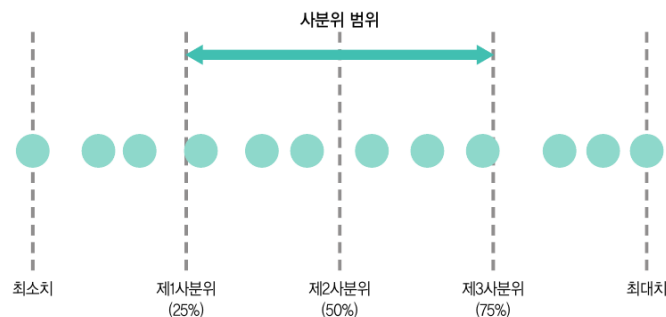
3. 이상치(outlier) 처리하기

1) 이상치 : 수집된 데이터 중에서 아주 작은 값이나 매우 큰 값

2) 이상치 측정 : IQR 이용

① IQR(Interquartile range) : 사분위수의 75% 지점의 값과 25% 지점의 값 차이($Q3 - Q1$)

② 사분위수 : 순서대로 정렬된 데이터를 4구간으로 균등하게 분할 할 때 기준이 되는 값(수)



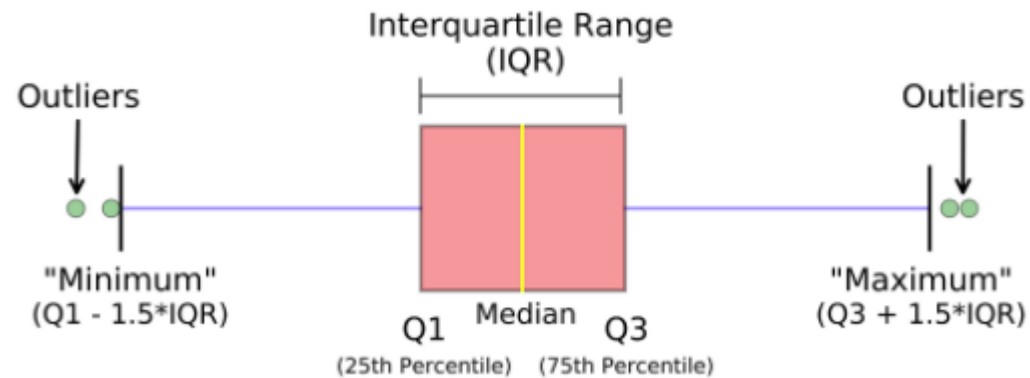
③ 정상 범위 데이터의 최소값(min) = $Q1 - (1.5 * IQR)$

④ 정상 범위 데이터의 최대값(max) = $Q3 + (1.5 * IQR)$

데이터 전처리 (preprocessing)

3. 이상치(outlier) 처리하기

⑤ 이상치 : 정상 범위 데이터의 min, max를 벗어난 데이터



데이터 전처리 (preprocessing)

3. 이상치(outlier) 처리하기

⑥ 요약 통계량

- 데이터 분포의 특징을 나타내는 통계량
- describe() 함수 : DataFrame 또는 Series 자료형 → 요약 통계량 제공
- pd.DataFrame.describe()

	temp	hum	atm	speed
count	100000.000000	100000.000000	100000.000000	100000.000000
mean	16.909460	78.159290	1005.071590	3.315666
std	3.840548	21.021064	5.657643	1.915367
min	9.000000	17.000000	993.000000	0.500000
25%	15.000000	66.000000	1002.000000	1.500000
50%	16.000000	87.000000	1006.000000	3.100000
75%	19.000000	93.000000	1007.000000	4.600000
max	27.000000	100.000000	1028.000000	10.300000

- i) count : 컬럼 별 데이터의 총수
- ii) mean / std : 컬럼 별 데이터의 평균 / 컬럼 별 데이터의 표준편차
- iii) min / max : 컬럼 별 데이터의 최소값 / 최대값
- iv) 25% / 50% / 75% : 사분위수의 각 지점에 해당하는 값

데이터 전처리 (preprocessing)

3. 이상치(outlier) 처리하기

3) 이상치 처리 방법

- ① 제거(Deletion) : IQR(Interquartile range)을 이용, 이상치 정의 → 이상치 제거
- ② 대체(Imputation) : 이상치를 특정 값(중앙값, 평균 등)으로 대체하는 것

데이터 전처리 (preprocessing)

3. 공공 데이터를 이용하여 이상치(outlier) 처리하기

- ▶ 기상청 날씨 데이터 활용
- ▶ 각 컬럼 별 이상치 제거

데이터 전처리 (preprocessing)

4. Feature Engineering(특성 공학)

1) 개념

- ① 분석 모델의 정확도를 높이기 위해서 feature들을 만드는 과정
- ② 해당 데이터에 대한 도메인 지식을 사용하여 feature들을 변형하는 과정

데이터 전처리 (preprocessing)

4. Feature Engineering

1) Encoding

- ① 개념 : 범주형 문자열 데이터를 수치형 데이터로 변환하는 작업
- ② 방법
 - Label Encoding
 - One-Hot Encoding

데이터 전처리 (preprocessing)

4. Feature Engineering


1-1) Label Encoding

① 개념 : 각 컬럼의 범주형 문자열 데이터를 0부터 n-1까지의 연속적 수치 데이터로 표현하는 것

② 구현 방법

- `df.replace({'문자열1':숫자1, 문자열2':숫자2, ...})`

예시) `df = df.replace({'no': 0, 'yes': 1})`



	age	job	marital	education	default	balance	housing	loan
0	58	management	married	tertiary	no	2143	yes	no
1	36	technician	single	secondary	no	265	yes	yes
2	25	blue-collar	married	secondary	no	-7	yes	no
3	53	technician	married	secondary	no	-3	no	no
4	24	technician	single	secondary	no	-103	yes	yes

	age	job	marital	education	default	balance	housing	loan
0	58	management	married	tertiary	0	2143	1	0
1	36	technician	single	secondary	0	265	1	1
2	25	blue-collar	married	secondary	0	-7	1	0
3	53	technician	married	secondary	0	-3	0	0
4	24	technician	single	secondary	0	-103	1	1

데이터 전처리 (preprocessing)

4. Feature Engineering

1-2) One-Hot Encoding

① 개념

- 범주형 컬럼의 문자열의 고유한 값 (n개) → 새로운 컬럼 생성 (n개)
- 기존 범주형 컬럼의 각 문자열 → 새로운 컬럼 중 해당 문자열 컬럼에 1, 나머지 컬럼에 0을 할당

id	day
0	월
1	월
2	화
3	수
4	화



id	day_월	day_화	day_수
0	1	0	0
1	1	0	0
2	0	1	0
3	0	0	1
4	0	1	0

데이터 전처리 (preprocessing)

4. Feature Engineering

1-2) One-Hot Encoding

② 구현 방법

- `pd.get_dummies(df)`
- `pd.get_dummies(df, columns=['col1', 'col2', ..])`

	name	color
0	apple	red
1	banana	yellow
2	cherry	red
3	durian	green
4	NaN	NaN

`pd.get_dummies(fruit)` *#dataframe 전체를 했을 때*

	name_apple	name_banana	name_cherry	name_durian	color_green	color_red	color_yellow
0	1	0	0	0	0	1	0
1	0	1	0	0	0	0	1
2	0	0	1	0	0	1	0
3	0	0	0	1	1	0	0
4	0	0	0	0	0	0	0

`pd.get_dummies(fruit, columns=['name'])` *#dataframe에서 특정 열만 인코딩*

	color	name_apple	name_banana	name_cherry	name_durian
0	red	1	0	0	0
1	yellow	0	1	0	0
2	red	0	0	1	0
3	green	0	0	0	1
4	NaN	0	0	0	0

데이터 전처리(preprocessing)

4. Feature Engineering

2) 구간화(Binning)

① 개념 : 연속형 데이터를 특정 구간으로 나누어 범주형 데이터로 변환하는 작업

② 구현 방법

- `pd.cut(x, bins, labels)`

- x : data

- bins : 구간의 크기

- labels: 범주의 라벨

데이터 탐색

1. 상관관계 분석(Correlation Analysis)

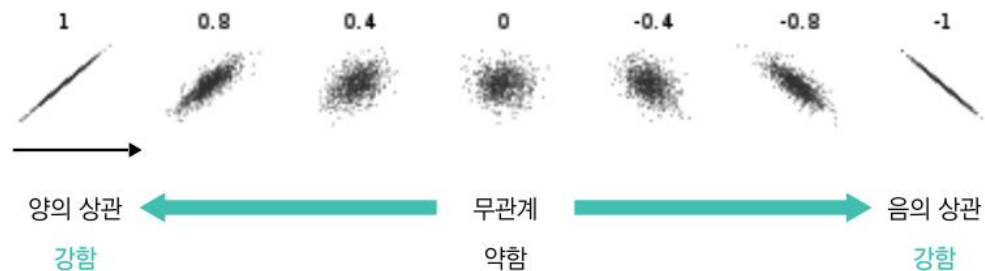
1) 개념 : 두 변수 사이에 연관성의 존재 여부와 연관성의 강도를 측정하여 분석하는 방법

예) 기업에서 광고비 지출이 매출액의 증가에 어느 정도의 영향이 있는 지를 파악

2) 상관관계의 종류

종류	설명
양(+의) 상관관계	<ul style="list-style-type: none">한 변수의 값이 증가할 때 다른 변수의 값도 증가한다강도에 따라 강한 양의 상관관계, 약한 양의 상관관계가 있다
음(-의) 상관관계	<ul style="list-style-type: none">한 변수의 값이 증가할 때 다른 변수의 값은 감소한다강도에 따라 강한 음의 상관관계, 약한 음의 상관관계가 있다
상관관계가 없음	한 변수의 값의 변화에 무관하게 다른 변수의 값이 변화한다

3) 상관계수(피어슨 상관계수) : 두 변수 간에 존재하는 선형 관계의 정도를 수량화, +1 ~ -1



데이터 탐색

1. 상관관계 분석 (Correlation Analysis)

4) 구현 방법

① df.corr() : 변수들 간의 상관 계수를 표현한 데이터프레임 생성 → 상관 행렬

	survived	pclass	age	sibsp	parch	fare
survived	1.000000	-0.338481	-0.077221	-0.035322	0.081629	0.257307
pclass	-0.338481	1.000000	-0.369226	0.083081	0.018443	-0.549500
age	-0.077221	-0.369226	1.000000	-0.308247	-0.189119	0.096067
sibsp	-0.035322	0.083081	-0.308247	1.000000	0.414838	0.159651
parch	0.081629	0.018443	-0.189119	0.414838	1.000000	0.216225
fare	0.257307	-0.549500	0.096067	0.159651	0.216225	1.000000



② 데이터의 형태에 따른 상관관계 분석 방법

Y변수 \ X변수	연속형	범주형	순서형
연속형	피어슨(Pearson)		
범주형	Point biserial Biserial Polyserial	파이계수(Phi) 크래머V(Cramer'V)	
순서형	스피어만(Spearman)	-	켄달(Kendall) 스피어만(Spearman)

데이터 탐색

2. 통계로 요약하기

1) 요약 통계량 구하기

- ▶ 요약 통계량 : 데이터 분포의 특징을 나타내는 통계량
- ▶ 주요 내용
 - describe() : dataframe 자료형에서 요약 통계량 제공

데이터 탐색

2. 통계로 요약하기

1) 요약 통계량 구하기

▶ describe()

- 수치형 컬럼 : `pd.DataFrame.describe()`

	age	balance	day	duration	campaign	pdays	previous
count	7234.000000	7234.000000	7234.000000	7234.000000	7234.000000	7234.000000	7234.000000
mean	40.834808	1374.912911	15.623860	262.875311	2.713989	40.277716	0.565939
std	10.706442	3033.882933	8.307826	268.921065	2.983740	99.188008	1.825100
min	2.000000	-3313.000000	1.000000	0.000000	1.000000	-1.000000	0.000000
25%	33.000000	74.000000	8.000000	103.000000	1.000000	-1.000000	0.000000
50%	39.000000	453.500000	16.000000	183.000000	2.000000	-1.000000	0.000000
75%	48.000000	1470.750000	21.000000	321.750000	3.000000	-1.000000	0.000000
max	157.000000	81204.000000	31.000000	3366.000000	44.000000	850.000000	40.000000

- ① count : 컬럼 별 데이터의 총수
- ② mean / std : 컬럼 별 데이터의 평균 / 컬럼 별 데이터의 표준편차
- ③ min / max : 컬럼 별 데이터의 최소값 / 최대값
- ④ 25% / 50% / 75% : 사분위수의 각 지점에 해당하는 값

데이터 탐색

2. 통계로 요약하기

2) 평균 구하기

▶ 주요 내용

- `df.mean()`
- `series.mean()`

데이터 탐색

2. 통계로 요약하기

3) 중앙값 구하기

- ▶ 중앙값(median) : 데이터를 오름차순으로 나열하였을 때 가운데(50%) 있는 값

1 2 4 7 9

- ▶ 주요 내용

- `df.median()`
- `series.median()`
- 중복 데이터 제거하고 중앙값 구하기

데이터 탐색

2. 통계로 요약하기

4) 최솟값, 최대값 구하기

▶ 주요 내용

- 최솟값 : `df.min()` / `series.min()`
- 최대값 : `df.max()` / `series.max()`

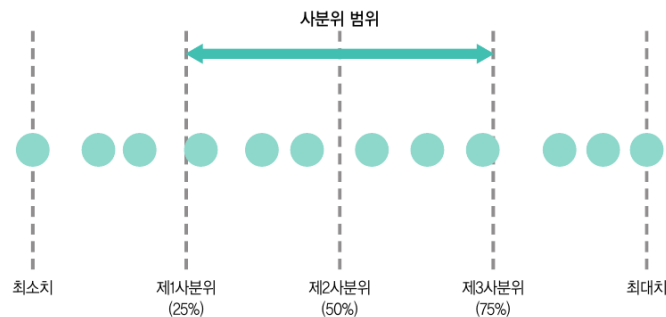
데이터 탐색

2. 통계로 요약하기

5) 분위수(quantile) 구하기

▶ 분위수(quantile) : 순서대로 정렬된 데이터를 균등한 간격으로 분할 할 때 기준이 되는 값(수)

- 사분위수 : 데이터를 4등분하는 경우 기준이 되는 수 $\rightarrow 0.25, 0.5, 0.75$
- 백분위수 : 데이터를 100등분하는 경우 기준이 되는 수 $\rightarrow 0.01 \sim 0.99$



▶ 주요 내용

- `df.quantile()`
- `df.quantile()`

데이터 탐색

2. 통계로 요약하기

5) 분위수(quantile) 구하기

▶ quantile(q)

- q : 원하는 백분위수 설정

```
# Series 자료형 데이터 생성
s = pd.Series([1,2,3,4,5])
print(s)

print('-'*80)

# 1/4 분위수 구하기
q1 = s.quantile(q=0.25)
print(q1)

print('-'*80)

# 2/4 분위수 구하기
q2 = s.quantile(q=0.5)
print(q2)

print('-'*80)

# 3/4 분위수 구하기
q3 = s.quantile(q=0.75)
print(q3)
```



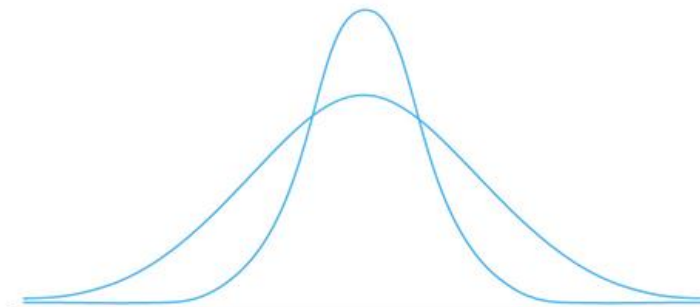
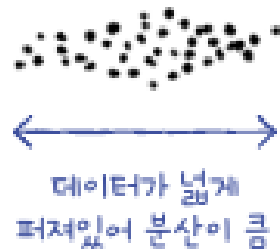
```
0    1
1    2
2    3
3    4
4    5
dtype: int64
-----
2.0
-----
3.0
-----
4.0
```


데이터 탐색

2. 통계로 요약하기

6) 분산(variance) 구하기

▶ 분산 : 평균으로부터 데이터가 얼마나 퍼져 있는지를 나타내는 통계량



▶ 주요 내용

- `df.var()`
- `df.var()`

데이터 탐색

2. 통계로 요약하기

7) 표준편차(standard deviation) 구하기

- ▶ 표준편차 : 분산에 제곱근을 한 것
- ▶ 주요 내용
 - `df.std()`
 - `series.std()`

데이터 탐색

2. 통계로 요약하기

8) 최빈값(mode) 구하기

- ▶ 최빈값 : 데이터에서 가장 많이 등장하는 값
- ▶ 주요 내용
 - `df.mode()`
 - `series.mode()`

데이터 탐색

3. 탐색적 데이터 분석(EDA, Exploratory Data Analysis)

1) 개념 : 시각화를 통한 데이터에 대한 이해(특징과 패턴 파악)

2) 시각화 (visualization)

- ① 방대한 양의 자료를 한눈에 볼 수 있도록 도표나 차트 등으로 정리하는 것
- ② 시각화를 통해 데이터의 특징을 쉽게 파악 할 수 있다
- ③ 분석 결과를 상대방에게 효과적으로 전달 가능하다

데이터 탐색

3. 탐색적 데이터 분석(EDA, Exploratory Data Analysis)

3) python 시각화 라이브러리

- ① matplotlib.pyplot
- ② seaborn

4) 주로 사용되는 시각화 도구

- ① 막대 그래프(Bar chart)
- ② 히스토그램(Histogram)
- ③ 박스 플롯(Box plot)
- ④ 산점도(Scatter plot)
- ⑤ 히트맵(heatmap)
- ⑥ Implot

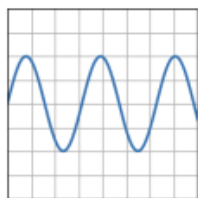
데이터 탐색

3. 탐색적 데이터 분석(EDA, Exploratory Data Analysis)

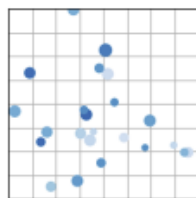
▶ python 시각화 라이브러리 : matplotlib.pyplot

- Python에서 시각화 할 수 있는 대표적인 라이브러리
- matplotlib 사용하기

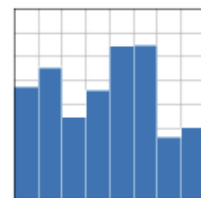
```
# import matplotlib.pyplot as plt
```



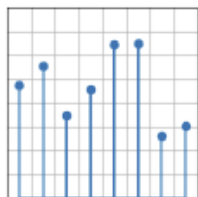
plot(x, y)



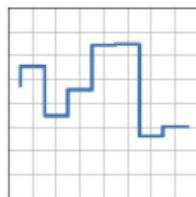
scatter(x, y)



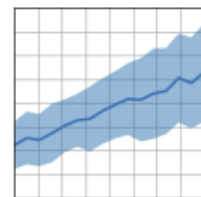
bar(x, height)



stem(x, y)



step(x, y)



fill_between(x, y1, y2)

데이터 탐색

3. 탐색적 데이터 분석(EDA, Exploratory Data Analysis)

▶ python 시각화 라이브러리 : matplotlib.pyplot

1) Line plot(선 그리기)

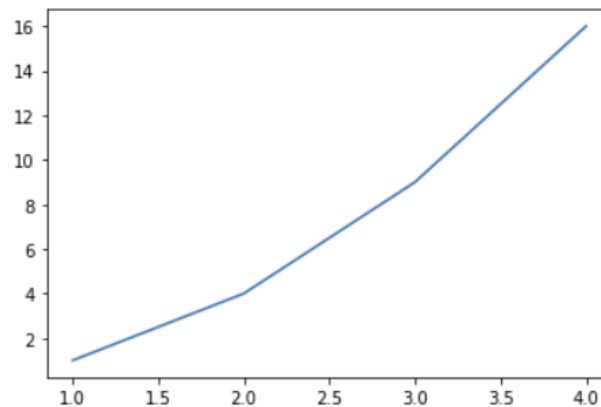
① 데이터 설정하기

`x = [1, 2, 3, 4],`

`y = [1, 4, 9, 16]`

② 그려주라!! → `plt.plot(x, y)`

③ 보여주라!! → `plt.show()`



데이터 탐색

3. 탐색적 데이터 분석(EDA, Exploratory Data Analysis)

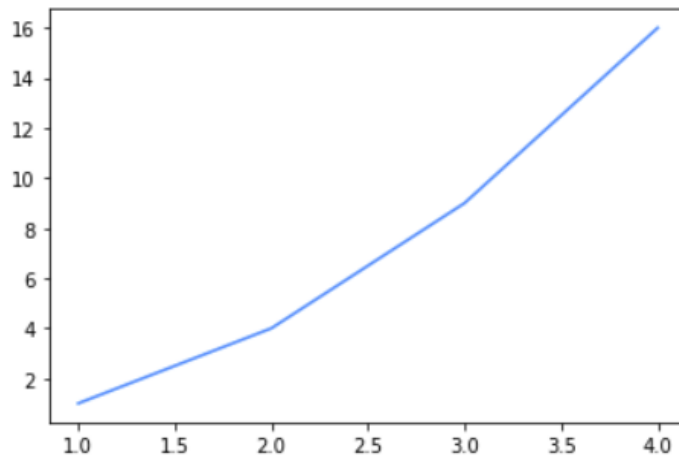
▶ python 시각화 라이브러리 : matplotlib.pyplot

1) Line plot(선 그리기) : 선의 색 지정

① 매개 변수 : color

② plt.plot(x, y, color='dodgerblue')

③ plt.show()



데이터 탐색

3. 탐색적 데이터 분석(EDA, Exploratory Data Analysis)

▶ python 시각화 라이브러리 : matplotlib.pyplot

1) Line plot(선 그리기) : 선의 색 정리

Character	Color	Character	Color
‘ b ’	Blue	‘ m ’	Magenta
‘ g ’	Green	‘ y ’	yellow
‘ r ’	Red	‘ k ’	Black
‘ c ’	cyan	‘ w ’	white

데이터 탐색

3. 탐색적 데이터 분석(EDA, Exploratory Data Analysis)

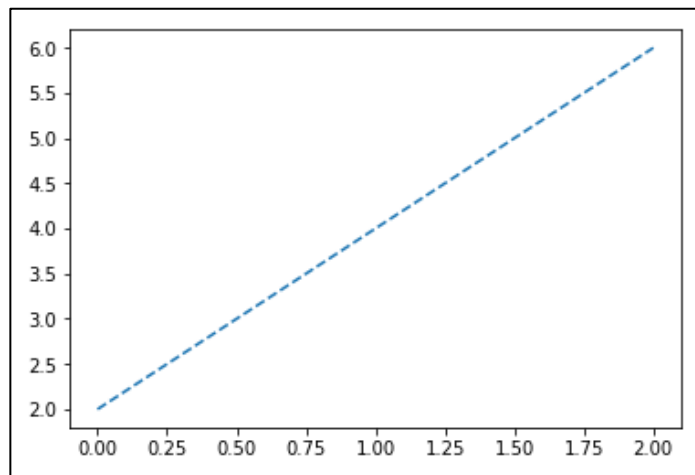
▶ python 시각화 라이브러리 : matplotlib.pyplot

1) Line plot(선 그리기) : 선의 모양 지정

① 매개 변수 : linestyle

② `plt.plot(x, y, linestyle = '--')`

③ `plt.show()`



데이터 탐색

3. 탐색적 데이터 분석(EDA, Exploratory Data Analysis)

▶ python 시각화 라이브러리 : matplotlib.pyplot

1) Line plot(선 그리기) : 선의 모양 정리

Character	Description
' _ '	Solid line style
' _ _ '	Dashed line style
' _ . '	Dash-dot lint style
' : '	Dotted line style

데이터 탐색

3. 탐색적 데이터 분석(EDA, Exploratory Data Analysis)

▶ python 시각화 라이브러리 : matplotlib.pyplot

1) Line plot(선 그리기) : 축 레이블 설정하기

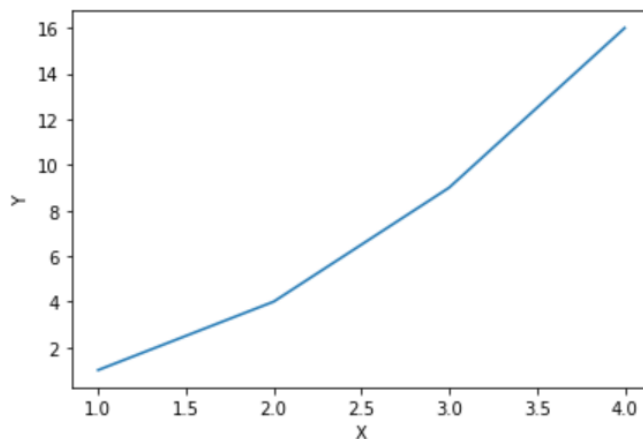
① matplotlib.pyplot 모듈의 **xlabel()**, **ylabel()** 함수 사용 → 그래프의 x, y 축에 대한 레이블 표시

② `plt.plot([1, 2, 3, 4], [1, 4, 9, 16])`

③ `plt.xlabel('X')`

④ `plt.ylabel('Y')`

⑤ `plt.show()`



데이터 탐색

3. 탐색적 데이터 분석(EDA, Exploratory Data Analysis)

▶ python 시각화 라이브러리 : matplotlib.pyplot

1) Line plot(선 그리기) : 타이틀 설정하기

① **title()** 함수 이용 → 그래프의 타이틀 (title) 설정

② plt.title('Sample graph')

③ plt.show()

데이터 탐색

3. 탐색적 데이터 분석(EDA, Exploratory Data Analysis)

▶ python 시각화 라이브러리

1) matplotlib.pyplot

- Bar Chart(막대 그래프) 그리기

① 범주가 있는 데이터에 대해서 범주 별 빈도수의 크기를 직사각형의 막대로 표현하는 그래프(예 : 혈액형)

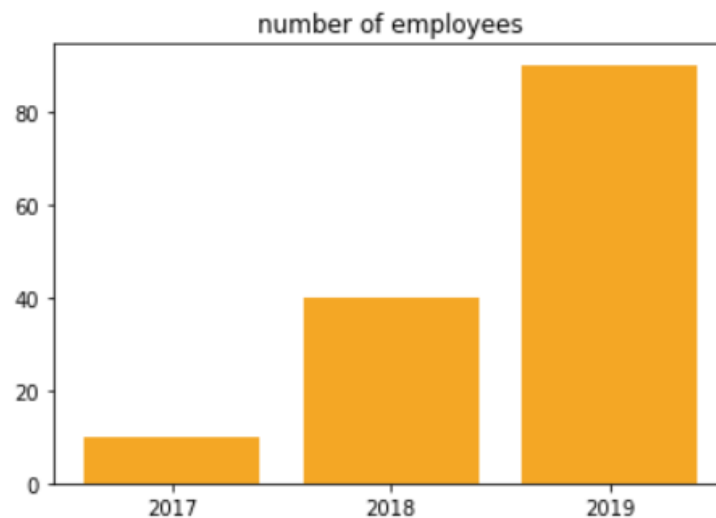
② 데이터 설정하기

```
x = ['2017', '2018', '2019']
```

```
height = [10, 40, 90]
```

③ plt.bar(x, height)

④ plt.show()



데이터 탐색

3. 탐색적 데이터 분석(EDA, Exploratory Data Analysis)

▶ python 시각화 라이브러리

1) matplotlib.pyplot

- Histogram(히스토그램) 그리기

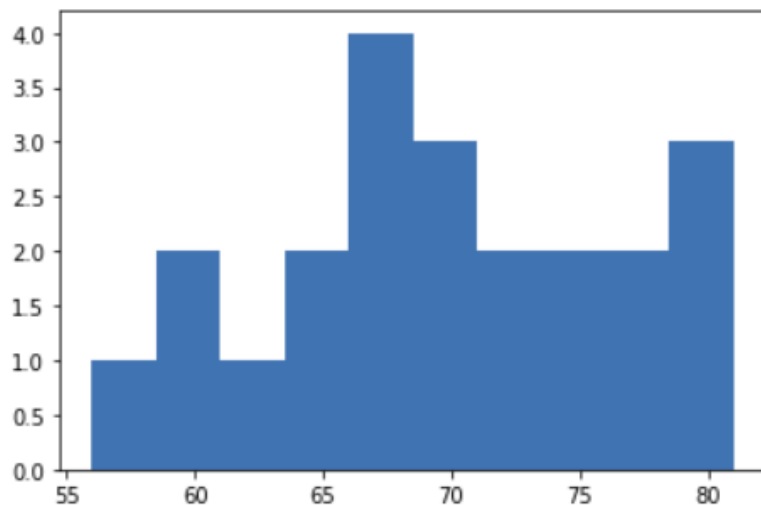
① 연속형 자료에 대해서 빈도수의 크기를 직사각형의 막대로 표현하는 그래프

② 데이터 생성하기

```
weight = [68, 81, 64, 56, 78, 74, 61, 77, 66, 68, 59, 71, 80, 59, 67, 81, 69, 73, 69, 74, 70, 65]
```

③ plt.hist(x=weight)

④ plt.show()



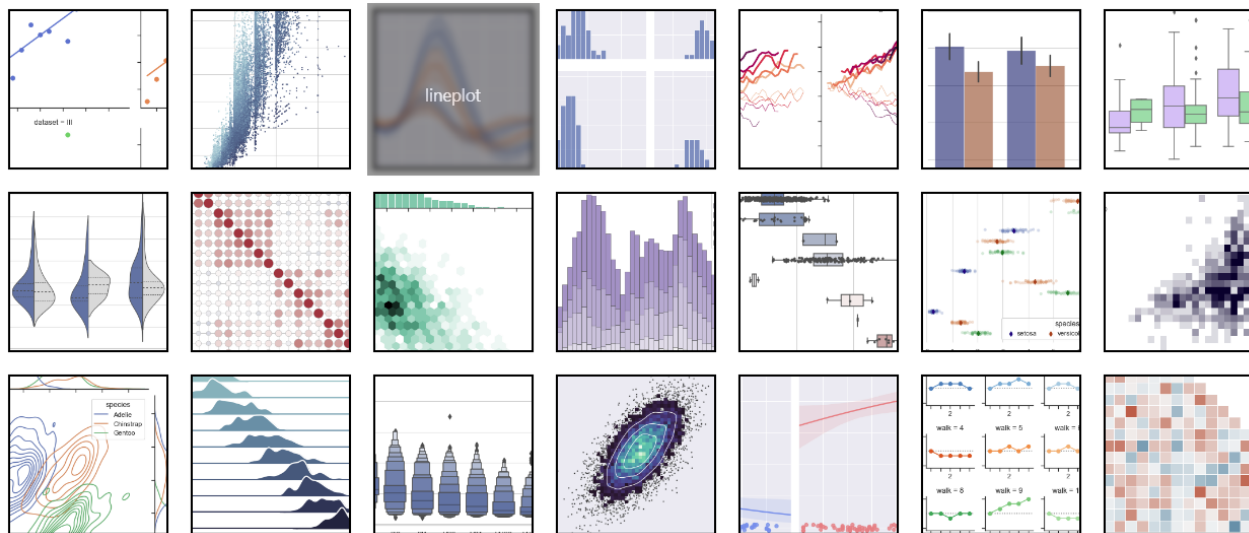
데이터 탐색

3. 탐색적 데이터 분석(EDA, Exploratory Data Analysis)

▶ python 시각화 라이브러리 : seaborn

- matplotlib을 기반, 통계 데이터 시각화 라이브러리
- seaborn 사용하기

```
# import seaborn as sns
```



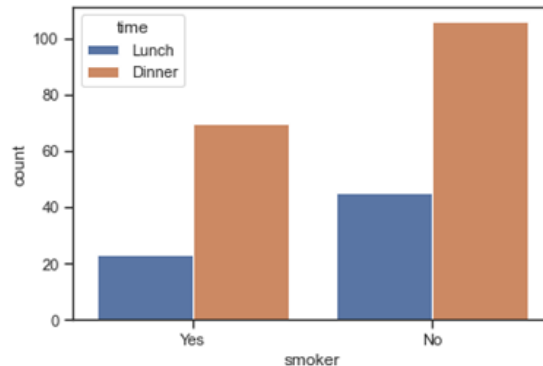
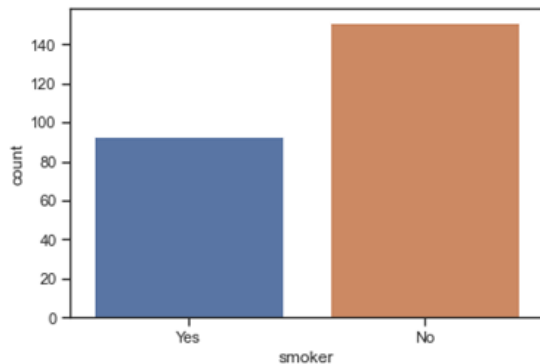
데이터 탐색

3. 탐색적 데이터 분석(EDA, Exploratory Data Analysis)

▶ python 시각화 라이브러리 : seaborn

1) 막대형 그래프(Bar chart)

- ① 개념 : 범주가 있는 데이터에 대해서, 범주 별 빈도수의 크기를 직사각형의 막대로 표현하는 그래프
- ② 구현 방법 : `sns.countplot(data , x, y, hue)`
- ③ 하이퍼파라미터
 - data : 데이터프레임
 - x, y : 시각화 할 컬럼(x 사용 : 수직 막대형 그래프, y 사용 : 수평 막대형 그래프)
 - hue : 새로운 변수를 추가, 새로운 의미 부여 가능



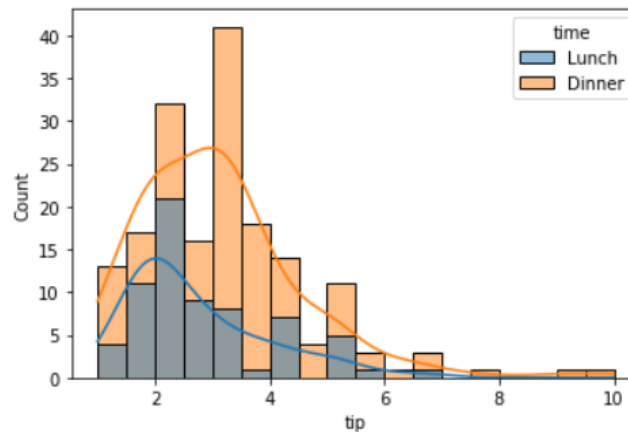
데이터 탐색

3. 탐색적 데이터 분석(EDA, Exploratory Data Analysis)

▶ python 시각화 라이브러리 : seaborn

2) 히스토그램(Histogram)

- ① 개념 : 연속형 자료에 대해서 빈도수의 크기를 직사각형의 막대로 표현하는 그래프
- ② 구현 방법 : `sns.histplot(data , x, y, hue, kde)`
- ③ 하이퍼파라미터
 - data : 데이터프레임
 - x, y : 시각화 할 컬럼(x 사용 : 수직 히스토그램, y 사용 : 수평 히스토그램)
 - kde 밀도 함수 곡선 추가
 - hue : 새로운 변수를 추가, 새로운 의미 부여 가능



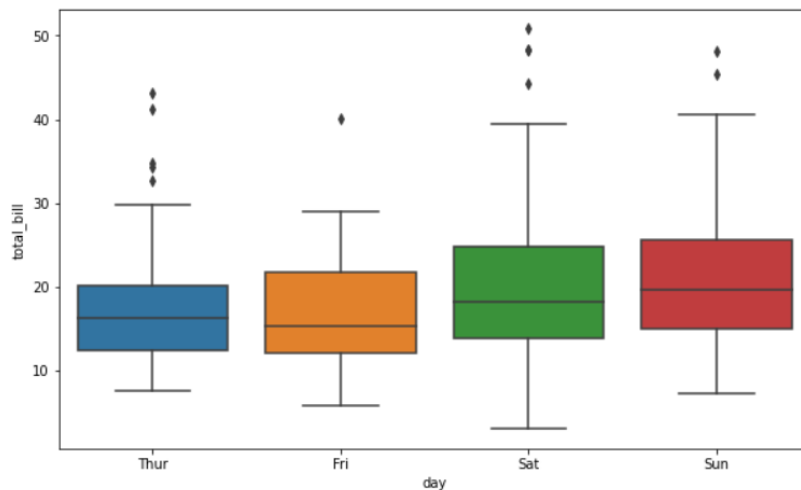
데이터 탐색

3. 탐색적 데이터 분석(EDA, Exploratory Data Analysis)

▶ python 시각화 라이브러리 : seaborn

3) 박스 플롯(Box plot)

- ① 개념 : 제 1사분위수(Q1), 중앙값(Q2, median), 제 3사분위수(Q3), 최소값(min), 최대값(max)을 이용하여 수치형 데이터를 표현하는 방법
- ② 구현 방법 : `sns.boxplot(data , x, y)`
- ③ 하이퍼파라미터
 - data : 데이터프레임
 - x, y : 시각화 할 컬럼



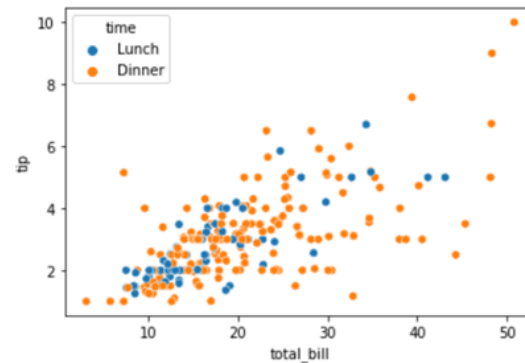
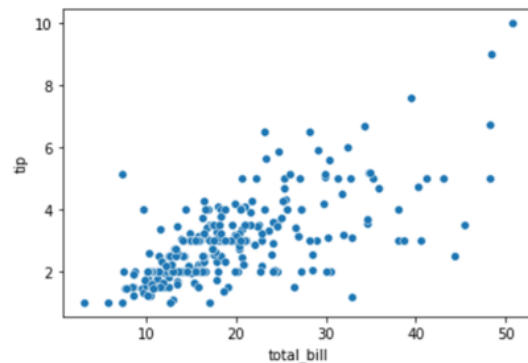
데이터 탐색

3. 탐색적 데이터 분석(EDA, Exploratory Data Analysis)

▶ python 시각화 라이브러리 : seaborn

4) 산점도 그래프(Scatter plot)

- ① 개념 : 두 변수의 관계를 직교 좌표계의 평면에 점으로 표현하는 그래프
- ② 구현 방법 : `sns.scatterplot(data, x, y, hue)` 함수를 이용
- ③ 하이퍼파라미터
 - data : 데이터프레임
 - x, y : 시각화 할 컬럼
 - hue : 새로운 변수를 추가, 새로운 의미 부여 가능



데이터 탐색

3. 탐색적 데이터 분석(EDA, Exploratory Data Analysis)

▶ python 시각화 라이브러리 : seaborn

5) heatmap

- ① 개념 : 변수들 간의 상관 계수 크기를 색상으로 표현
- ② 구현 방법 : `sns.heatmap(data, annot, fmt, linewidths)` 함수를 이용
- ③ 하이퍼파라미터
 - data : 상관 행렬, `df.corr()`
 - annot : 상관계수 표시 여부
 - fmt : 상관계수의 소수점 자리 수를 지정, 예시) `'%.2f'`
 - linewidths : cell 사이의 간격을 조정

	survived	pclass	age	sibsp	parch	fare
survived	1.000000	-0.338481	-0.077221	-0.035322	0.081629	0.257307
pclass	-0.338481	1.000000	-0.369226	0.083081	0.018443	-0.549500
age	-0.077221	-0.369226	1.000000	-0.308247	-0.189119	0.096067
sibsp	-0.035322	0.083081	-0.308247	1.000000	0.414838	0.159651
parch	0.081629	0.018443	-0.189119	0.414838	1.000000	0.216225
fare	0.257307	-0.549500	0.096067	0.159651	0.216225	1.000000



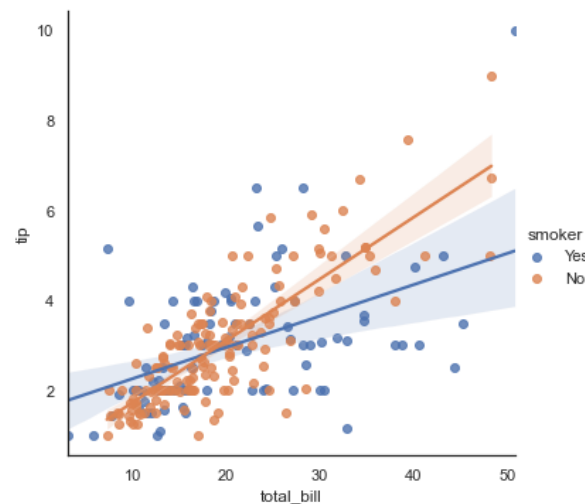
데이터 탐색

3. 탐색적 데이터 분석(EDA, Exploratory Data Analysis)

▶ python 시각화 라이브러리 : seaborn

6) Implot

- ① 개념 : 산점도, 선형회귀직선을 동시에 표현
- ② 구현 방법 : `sns.lmplot(data, x, y, hue)` 함수를 이용
- ③ 하이퍼파라미터
 - data : 데이터프레임
 - x, y : 시각화 할 컬럼
 - hue : 새로운 변수를 추가, 새로운 의미 부여 가능



데이터 탐색

3. 탐색적 데이터 분석(EDA, Exploratory Data Analysis)

▶ python 시각화 라이브러리 : pandas

1) 막대 그래프 그리기

① 데이터 생성하기

```
data_list = [10, 40, 90]
```

```
index_list = ['2017', '2018', '2019']
```

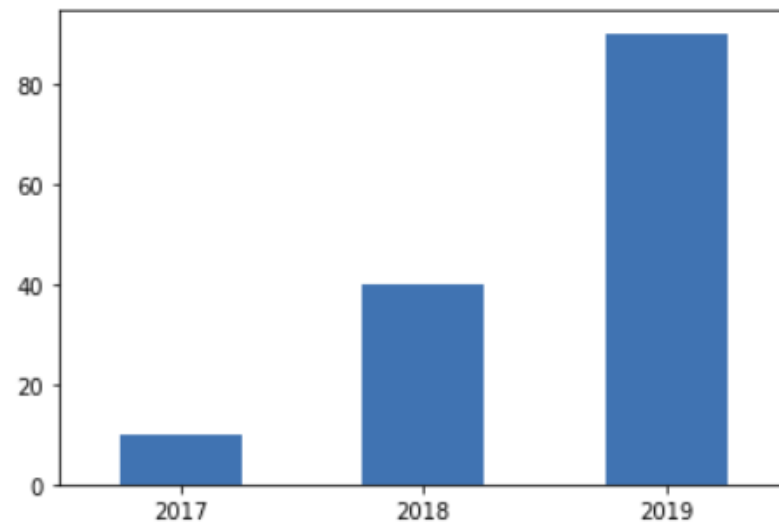
② series 자료형 생성하기

```
series = pd.Series(data=data_list, index=index_list)
```

③ 시각화

```
# index를 기준으로 값을 시각화 : x축 → index
```

```
# series.plot(kind='bar', rot=0)
```



데이터 탐색

3. 탐색적 데이터 분석(EDA, Exploratory Data Analysis)

▶ python 시각화 라이브러리 : pandas

1) 막대 그래프 그리기

① 데이터 생성하기

```
data = {'2015':[9904312,3448737,2890451,2466052],  
        '2010':[9631482,3393191,2632035,2431774]}
```

```
index = ['seoul', 'busan', 'inchon', 'daegu']
```

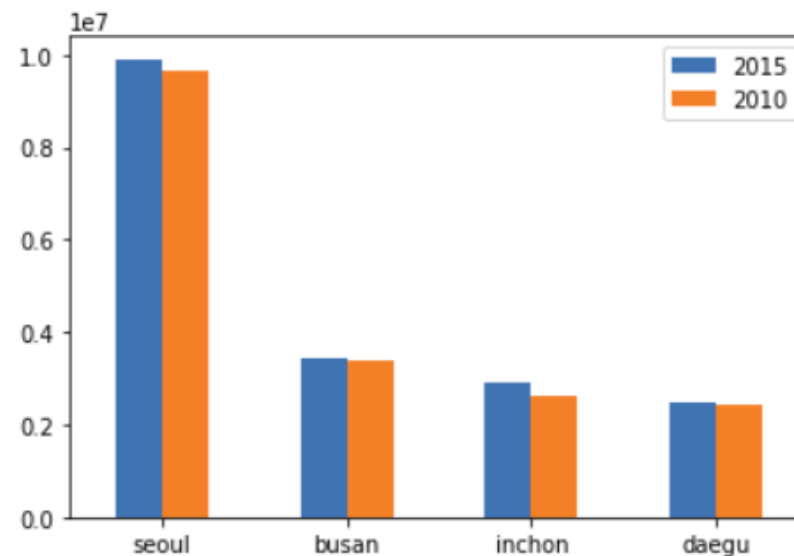
② dataframe 자료형 생성하기

```
df = pd.DataFrame(data, index)
```

③ 시각화

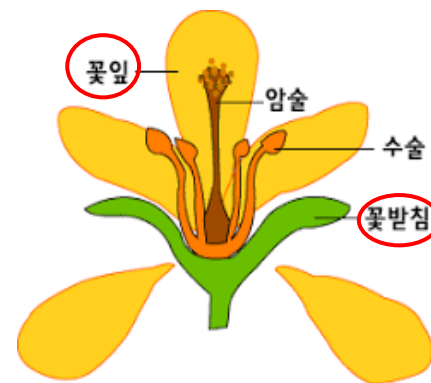
```
# index를 기준으로 값을 시각화 : x축 → index
```

```
# df.plot(kind='bar', rot=0)
```



iris 데이터 분석

- 150개의 붓꽃 데이터
- 컬럼 정보 : 4개의 특성 + 품종 레이블
 - sepal length (cm) : 꽃받침의 길이
 - sepal width (cm) : 꽃받침의 너비
 - petal length (cm) : 꽃잎의 길이
 - petal width (cm) : 꽃잎의 너비
 - label : 품종 레이블



setosa



virginica



versicolor

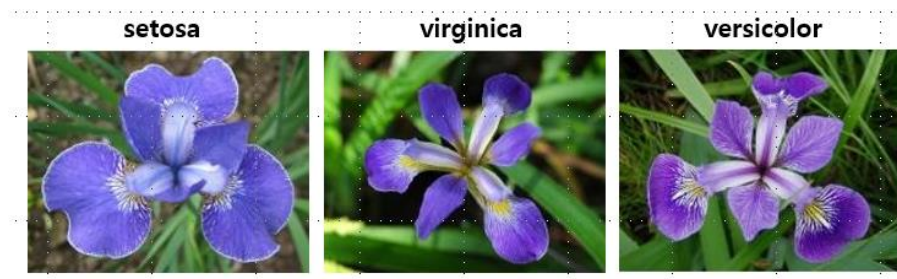


- 분석 목표 : 붓꽃의 품종을 분류할 수 있는 특성 분석 및 시각화
 - ⇒ 붓꽃의 품종을 분류하는 머신 러닝 모델의 원리 이해

모델링

- 개념 : 분석 목적에 부합하고 수집된 데이터의 특성을 고려하여 적합한 데이터 분석 모형 선정, 분석 실행
- 분석 목적 : 분류 또는 회귀
 - 1) 분류 : 기존에 존재하는 데이터의 범주(category)를 파악하고, 새롭게 관측된 데이터의 범주를 판별하는 과정
예) 품종 분류, 종양 분류(양성, 악성)

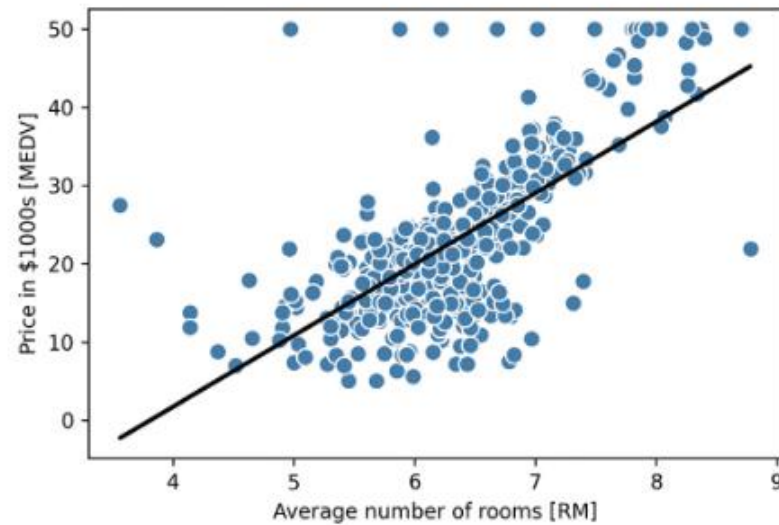
sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	label
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
7.0	3.2	4.7	1.4	versicolor
6.4	3.2	4.5	1.5	versicolor
6.9	3.1	4.9	1.5	versicolor
6.3	3.3	6.0	2.5	virginica
5.8	2.7	5.1	1.9	virginica
7.1	3.0	5.9	2.1	virginica



모델링

2) 회귀 : 연속적인 값을 예측

예) 주가 예측, 가격 예측



- 분석 모형

1) 통계기반 모형

2) 머신 러닝 기반 모형

06. 머신 러닝

- 기본 개념

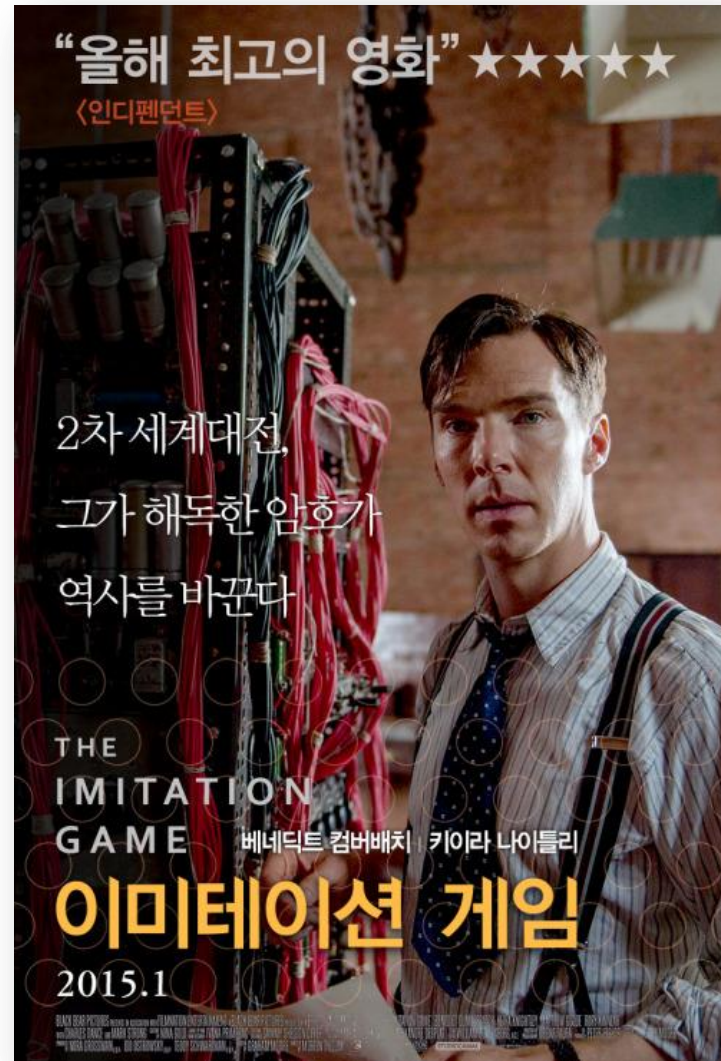
인공지능의 역사



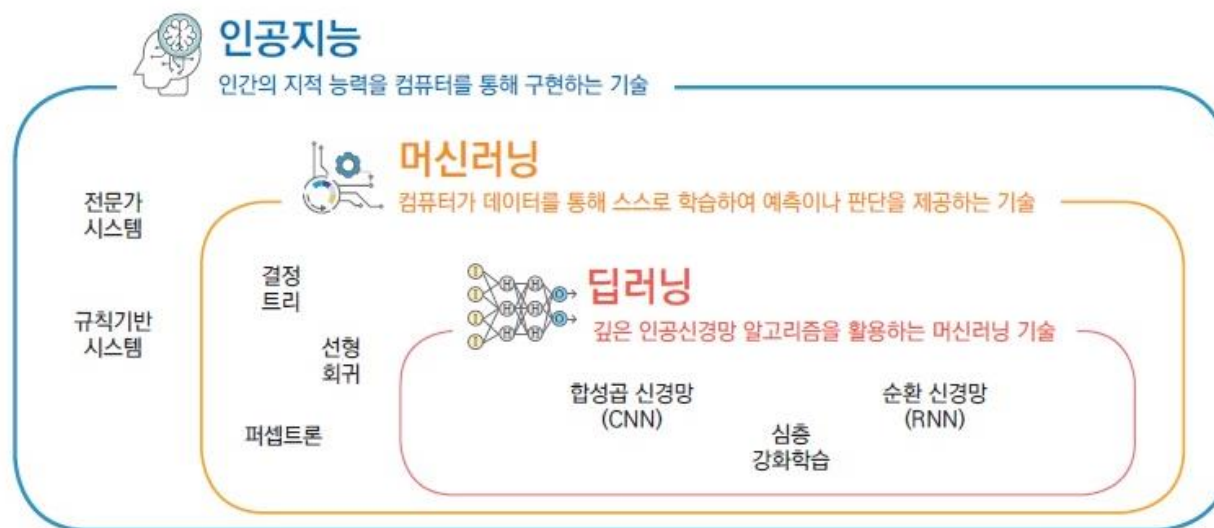
Alan Mathison Turing(앨런 매티슨 튜링)
(1912.6.23 ~ 1954.6.7)

영국의 수학자, 암호학자, 논리학자

인공지능의 역사



인공지능과 머신 러닝



sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	label
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
7.0	3.2	4.7	1.4	versicolor
6.4	3.2	4.5	1.5	versicolor
6.9	3.1	4.9	1.5	versicolor
6.3	3.3	6.0	2.5	virginica
5.8	2.7	5.1	1.9	virginica
7.1	3.0	5.9	2.1	virginica

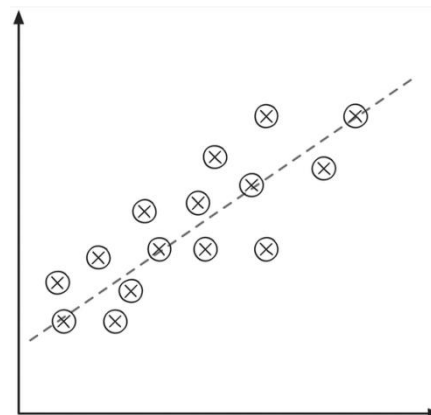
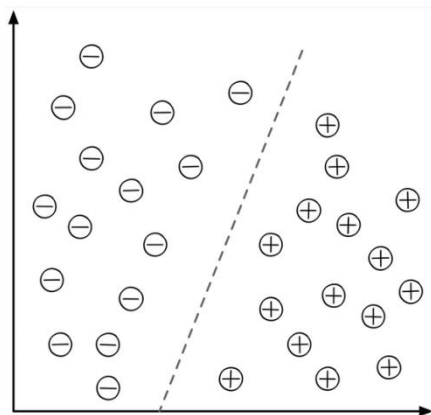


머신 러닝 종류(기준 : 학습 방식)

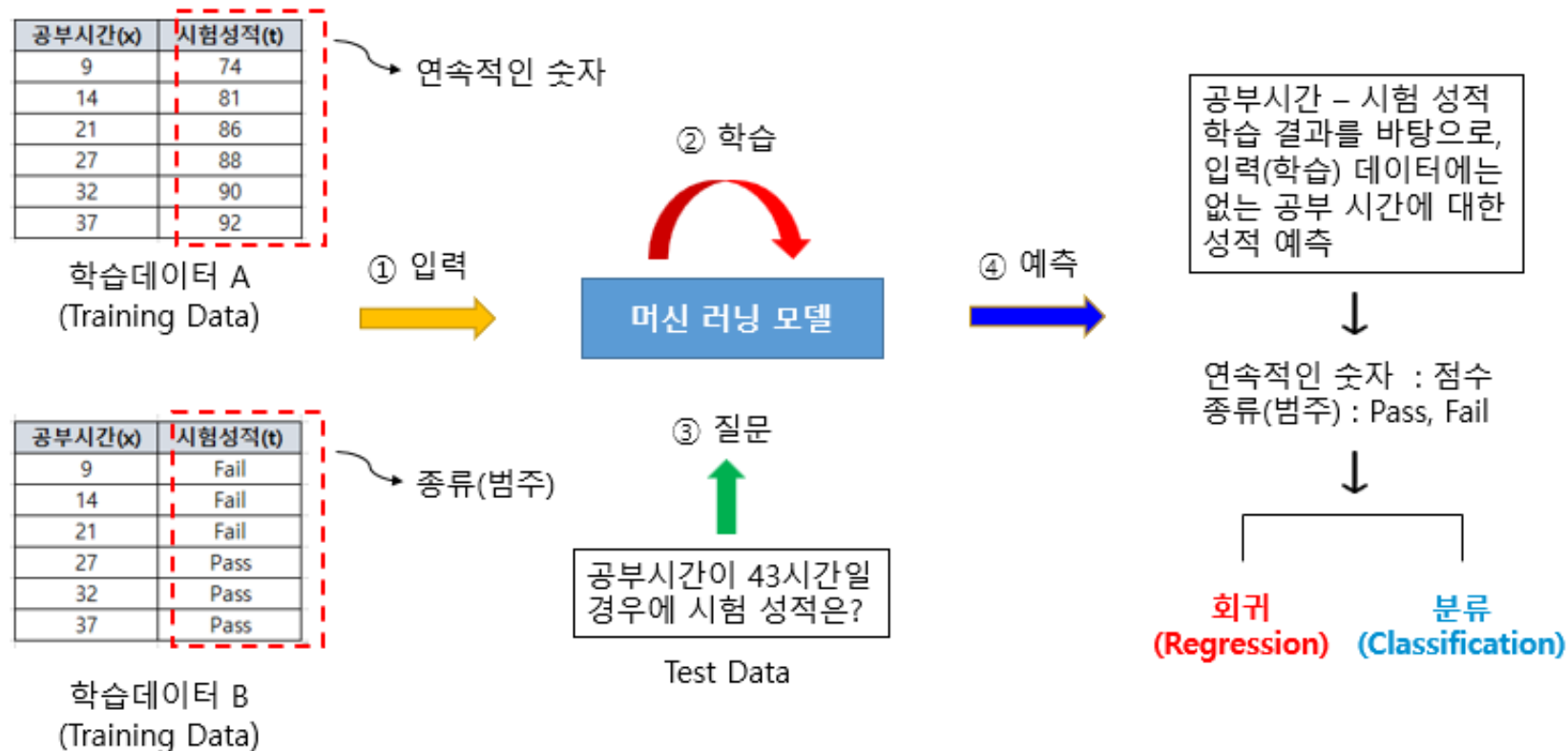
- 지도 학습(Supervised Learning)
- 비지도 학습(Unsupervised Learning)

지도 학습(Supervised Learning)

- 데이터에 대한 Label(명시적인 답)이 주어진 상태에서 머신 러닝 모델을 학습시키는 방법
- 분야(task)
 - 1) 분류 : 어떤 Label에 속할 지를 예측 예) 품종 분류, 종양 분류(양성, 악성)
 - 2) 회귀 : 연속적인 값을 예측 예) 주가 예측

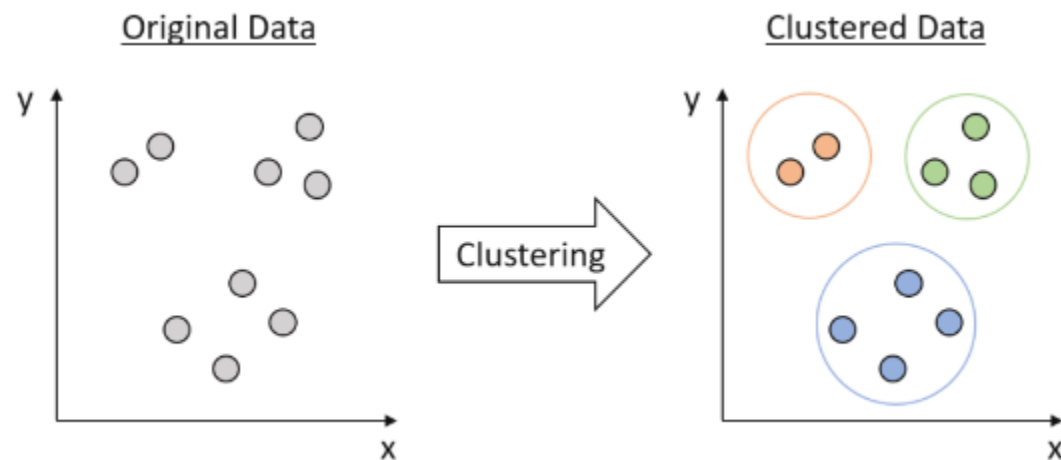


지도 학습(Supervised Learning)



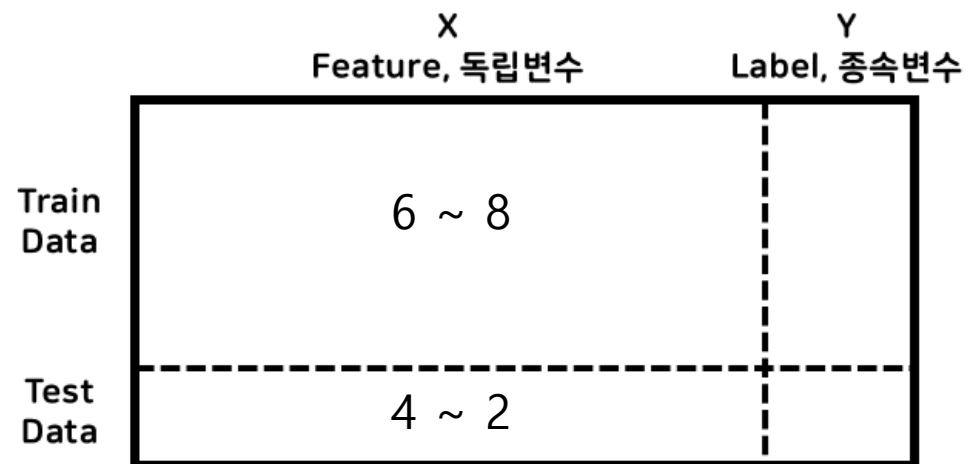
비지도 학습(Unsupervised Learning)

- 데이터에 대한 Label(명시적인 답)이 없는 상태에서 머신 러닝 모델을 학습시키는 방법
- 데이터를 비슷한 특성끼리 묶는 클러스터링(Clustering) 등이 있음



머신 러닝 학습 - 데이터의 분할

- 학습용 데이터와 평가용 데이터로 분할



- Hyper Parameter 조정을 통한 모델 성능 개선

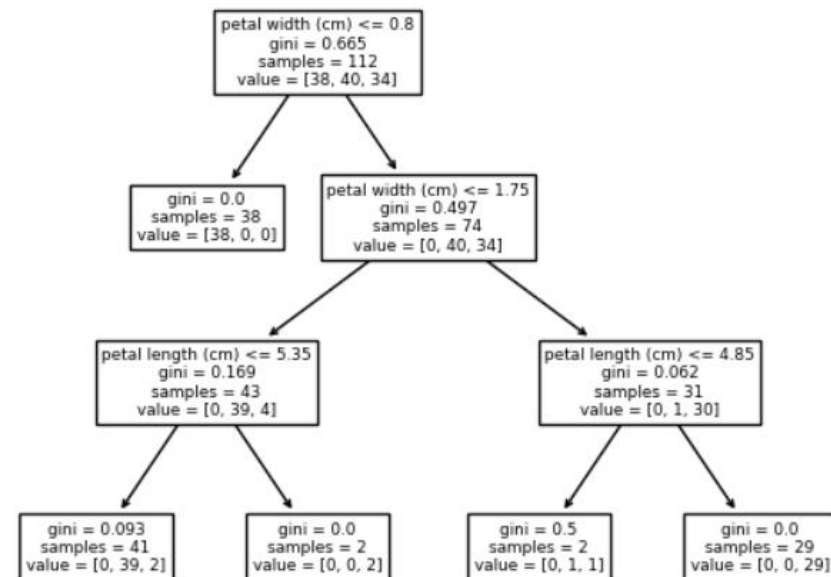
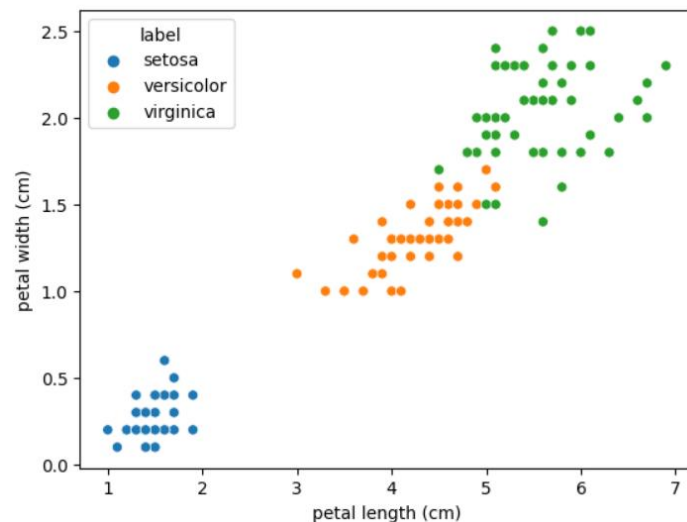
07. 머신 러닝

- 지도 학습을 이용한 분류

Decision Tree

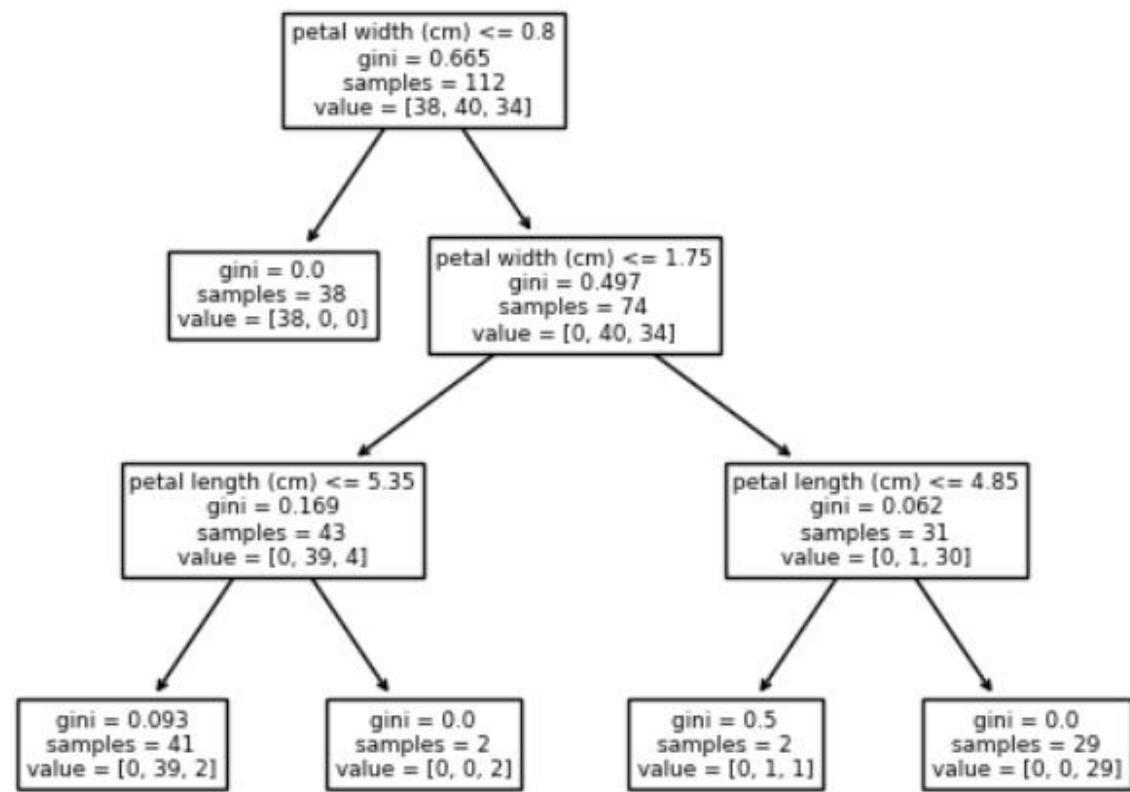
- 일련의 분류 규칙(비교 연산 반복)을 통해 데이터를 분류, 회귀하는 지도 학습 모델 중 하나, 지도 학습의 기본 모델
- 모델이 Tree 구조를 가지고 있기 때문에 Decision Tree라는 이름을 가지게 되었음

sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	label
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
7.0	3.2	4.7	1.4	versicolor
6.4	3.2	4.5	1.5	versicolor
6.9	3.1	4.9	1.5	versicolor
6.3	3.3	6.0	2.5	virginica
5.8	2.7	5.1	1.9	virginica
7.1	3.0	5.9	2.1	virginica



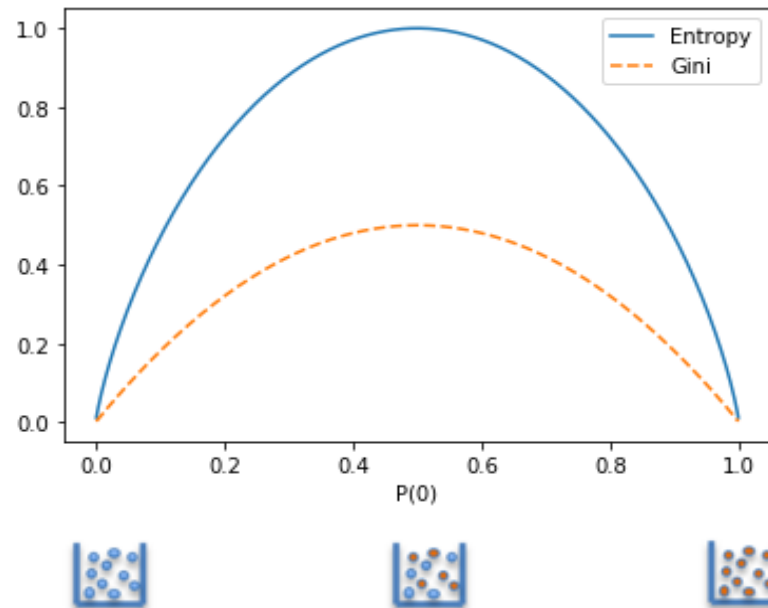
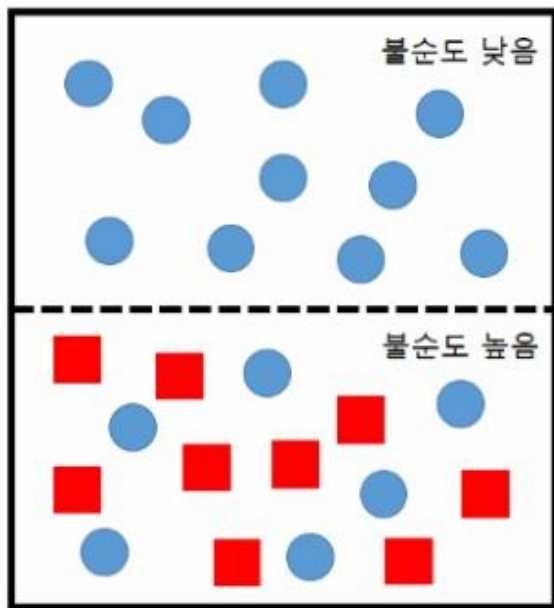
의사 결정 방향

- 불순도가 낮아지는 방향



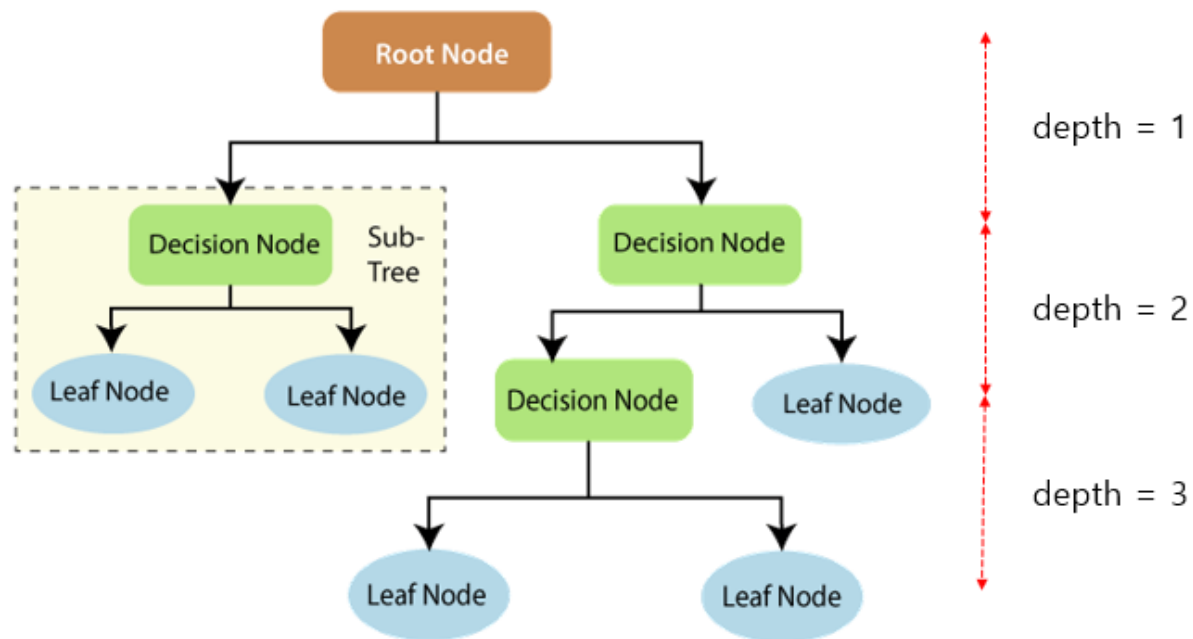
용어 정리 1. 불순도

- 서로 다른 데이터가 섞여 있는 정도
 - 다양한 개체들이 섞여 있을수록 불순도가 높아짐
- 불순도 함수 - 불순도를 수치적으로 나타내는 함수
 - 지니 지수(Gini) / 엔트로피 지수(Entropy)



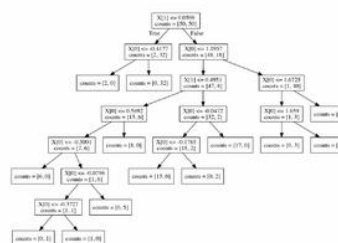
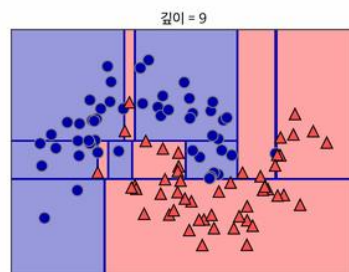
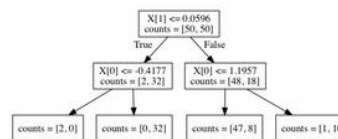
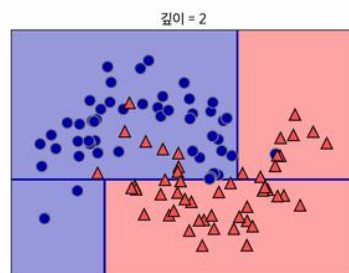
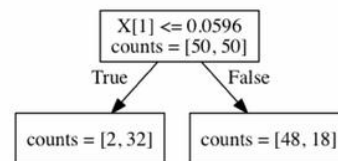
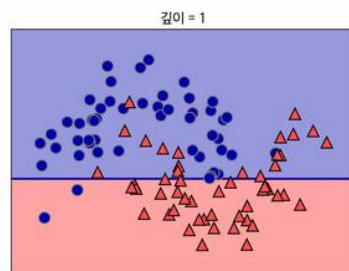
용어 정리 2. Node, depth, sub tree

- Node : 분류 기준 또는 분류의 결과
 - Root Node : 맨 처음 분류 기준
 - Intermediate Node(Decision Node) : 중간 분류 기준
 - Leaf Node : 분류가 끝난 결과
- depth : 분류의 단계



결정 트리 모델의 프로세스

- 데이터를 가장 잘 구분할 수 있는 질문을 기준으로 나눔
- 나뉜 각 범주에서 또 다시 데이터를 가장 잘 구분할 수 있는 질문을 기준으로 나눔

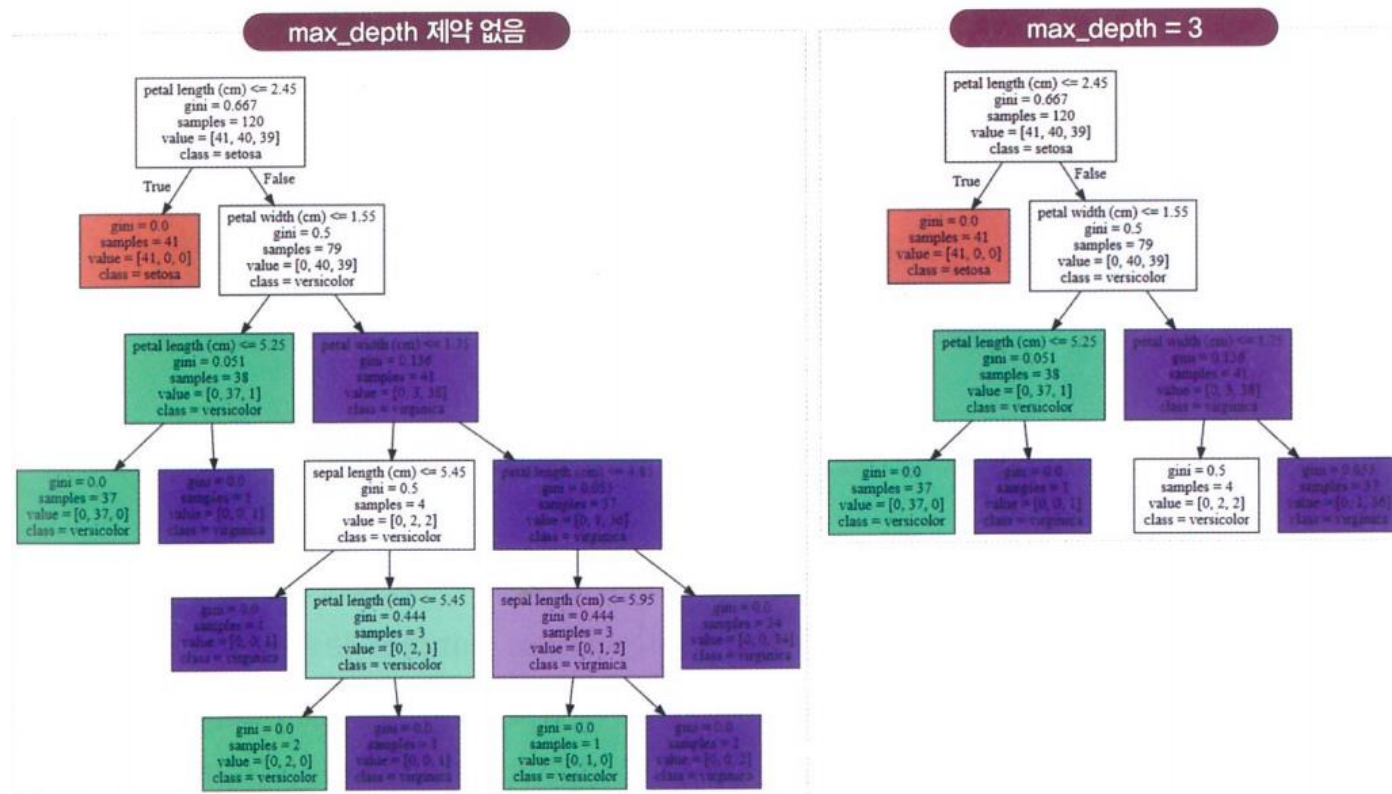


하이퍼파라미터와 과대 적합의 관계

- 과대 적합(Overfitting)
 - 모델을 학습할 때 지나치게 복잡하게 학습하여 학습 데이터셋에서는 모델 성능이 높게 나타나지만 새로운 데이터가 주어졌을 때 정확한 예측/분류를 수행하지 못하는 경우
 - 일반화 능력 부족
- **max_depth** : 값이 클수록 모델의 복잡도가 커지고 과대적합 가능성 증가

과대 적합 억제 – 사전 가지치기(pre-pruning)

- 결정 트리 모델 객체 생성 시 hyperparameter(max_depth) 값을 미리 지정
- 과대 적합 억제



모델의 장단점

장점	단점
<ul style="list-style-type: none">• 쉽다• 직관적이다	<ul style="list-style-type: none">• 과대적합이 발생하기 쉽다• 사전 가지치기와 같은 튜닝이 필요하다

Decision Tree 사용법

필요한 라이브러리 임포트

```
from sklearn.tree import DecisionTreeClassifier
```

결정 트리 분류 모델 생성 함수 호출, 결정 트리 분류 모델 객체 생성

```
dt = DecisionTreeClassifier(하이퍼파라미터)
```

주요 매개 변수(하이퍼파라미터)

- max_depth : 트리의 최대 깊이
- random_state : 재현성

Decision Tree 실습 개요

- 데이터를 학습용과 평가용으로 분할
 - `from sklearn.model_selection import train_test_split()`
 - `X_train, X_test, y_train, y_test = train_test_split(X, y, test_size, random_state)`
- 결정 트리 분류 모델 생성 함수 호출, 결정 트리 분류 모델 객체 생성
 - `from sklearn.tree import DecisionTreeClassifier`
 - `dt = DecisionTreeClassifier(random_state=0)`
- 분류 모델 학습 진행
 - `dt.fit(X_train, y_train)`

Decision Tree 실습 개요

- 평가용 데이터를 이용해서 예측
 - `y_pred = dt.predict(X_test)`
- 평가용 데이터에 대한 분류 모델의 성능(평균 정확도) 확인
 - `from sklearn.metrics import accuracy_score`
`accuracy = accuracy_score(y_test, y_pred)`
`print(accuracy)`

Decision Tree 실습1 – 붓꽃 품종 분류

- 전처리 : 정형 데이터 분석 및 시각화에서 완료
- 목표 : 학습용 데이터로 학습 후 평가용 데이터를 이용 품종 분류 / 모델 평가

Decision Tree 실습2 – 당뇨병 여부 분류

- 데이터 : Diabetes Health Indicators Dataset

- 미국 질병통제예방센터(CDC)에서 매년 수집하는 건강 관련 전화 설문조사 데이터
- 70692개의 데이터(컬럼 22개), 당뇨병 분류 결과 제공
- 실습 목적 : 컬럼 21개 → 탐색적 데이터 분석, 모델 학습 → 당뇨병 여부 분류(예측)

#	Column	Non-Null Count	Dtype
0	Diabetes_binary	70692 non-null	int64
1	HighBP	70692 non-null	int64
2	HighChol	70692 non-null	int64
3	CholCheck	70692 non-null	int64
4	BMI	70692 non-null	int64
5	Smoker	70692 non-null	int64
6	Stroke	70692 non-null	int64
7	HeartDiseaseorAttack	70692 non-null	int64
8	PhysActivity	70692 non-null	int64
9	Fruits	70692 non-null	int64
10	Veggies	70692 non-null	int64
11	HvyAlcoholConsump	70692 non-null	int64
12	AnyHealthcare	70692 non-null	int64
13	NoDocbcCost	70692 non-null	int64
14	GenHlth	70692 non-null	int64
15	MentHlth	70692 non-null	int64
16	PhysHlth	70692 non-null	int64
17	DiffWalk	70692 non-null	int64
18	Gender	70692 non-null	int64
19	Age	70692 non-null	int64
20	Education	70692 non-null	int64
21	Income	70692 non-null	int64

Decision Tree 실습2 – 당뇨병 여부 분류

- 컬럼 정보

- Diabetes_binary : 정답 레이블, 0 = 정상, 1 = 당뇨병 + 당뇨병 전단계
- HighBP : 고혈압 유무, 0 = 정상, 1 = 고혈압
- HighChol : 고지혈 유무, 0 = 정상, 1 = 고지혈
- CholCheck : 5년 동안에 고지혈 검사 여부, 0 = no, 1 = yes
- BMI : 체질량지수, 몸무게(kg)를 키의 제곱(m²)으로 나눈 값

비만 기준(세계보건기구)

BMI \geq 25 kg/m ² : 과체중
BMI \geq 30 kg/m ² : 비만

- Smoker : 최소 100개비 이상 흡연 여부, 0 = no, 1 = yes
- Stroke : 뇌졸중 여부, 0 = no, 1 = yes
- HeartDiseaseorAttack : 심근 경색과 같은 심장 질환 여부, 0 = no, 1 = yes
- PhysActivity : 지난 30일 동안 운동 여부, 0 = no, 1 = yes
- Fruits : 1일 1회 이상 과일 섭취 여부, 0 = no, 1 = yes

Decision Tree 실습2 – 당뇨병 여부 분류

- 컬럼 정보

- Veggies : 1일 1회 이상 야채 섭취 여부, 0 = no, 1 = yes
- HvyAlcoholConsump : 음주 여부, 0 = no, 1 = yes

```
### 음주의 기준  
# adult men >=14 drinks per week  
# adult women>=7 drinks per week
```

- AnyHealthcare : 건강 보장 보험 보유 여부, 0 = no, 1 = yes
- NoDocbcCost : 0 = no, 1 = yes
- GenHlth : 본인이 생각하는 자신의 전반적인 건강 상태 평가, 1 – 5단계
1 = excellent, 2 = very good, 3 = good, 4 = fair, 5 = poor
- MentHlth : 본인이 생각하는 자신의 정신 건강 상태 평가
질문 : 스트레스, 우울증을 포함하여 지난 30일 중 며칠 동안 정신 건강이 좋지 않았었나요?
답변 : 정신 건강이 좋지 않았던 일 수

Decision Tree 실습2 – 당뇨병 여부 분류

- 컬럼 정보

- PhysHlth : 본인이 생각하는 자신의 신체 건강 상태 평가
 - # 질문 : 신체적 질병과 부상을 포함하여 포함하여 지난 30일 중 며칠 동안 신체적 건강이 좋지 않았었나요?
 - # 답변 : 정신 건강이 좋지 않았던 일 수
- DiffWalk : 0 = no, 1 = yes
 - # 질문 : 당신은 걷거나 계단을 오르는 데 심각한 어려움이 있습니까?
 - # 답변 : 아니오(0), 예(1)
- Gender : 응답자의 성별, 0 = female, 1 = male
- Age : 응답자의 나이, 1 – 13 단계로 범주화
 - # 1단계 : 18 – 24 / 2단계 : 25 – 29 / 3단계 : 30 – 34 / 4단계 : 35 – 39 / 5단계 : 40 - 44
 - # 6단계 : 45 – 49 / 7단계 : 50 – 54 / 8단계 : 55 – 59 / 9단계 : 60 – 64 / 10단계 : 65 - 69
 - # 11단계 : 70 – 74 / 12단계 : 75 – 79 / 13단계 : 80 or older

Decision Tree 실습2 – 당뇨병 여부 분류

- 컬럼 정보

- Education : 응답자의 교육 수준. 1 - 6단계 범주화

1단계 : Never attended school / 2단계 Elementary / 3단계 Some high school

4단계 : High school graduate / 5단계 : Some college or technical school / 6단계 : College graduate

- Income : 응답자의 연간 가구 소득 수준, 1 - 8단계 범주화

1단계 : Less than \$10,000 / 2단계 : Less than \$15,000 / 3단계 : Less than \$20,000

4단계 : Less than \$25,000 / 5단계 : Less than \$35,000 / 6단계 : Less than \$50,000

7단계 : Less than \$75,000 / 8단계 : \$75,000 or more

모델 평가 방법의 개선

▶ 일반적인 경우

전체 데이터

Train
Data

(60% 또는 80%)

Test
Data

(40% 또는 20%)

모델 평가 방법의 개선

▶ K-fold cross-validation(교차 검증)



1. Train 데이터를 k개의 그룹으로 나눈다.
2. K-1개의 그룹을 학습에 사용한다.
3. 나머지 1개의 그룹을 이용해서 검증을 수행한다.
4. 2번, 3번 과정을 k번 반복한다.
5. 모든 결과의 평균을 구한다.

모델 평가 방법의 개선

▶ K-fold cross-validation(교차 검증)의 장/단점

- 장점

1. 모든 학습용 데이터를 검증에 활용할 수 있다.
 - 검증에 사용되는 데이터 편중을 막을 수 있다
2. 모든 학습용 데이터를 학습에 활용할 수 있다.
 - 정확도를 향상시킬 수 있다.
 - 데이터 부족으로 인한 underfitting을 방지할 수 있다
3. 데이터의 크기가 충분하지 않은 경우에도 유용하게 사용 가능하다.

- 단점

: 여러 번 학습하고 검증하는 과정을 거치기 때문에 모델 학습 시간이 오래 걸린다.

GridSearchCV 를 이용한 모델 성능 개선

- sklearn 라이브러리에서 제공해주는 모델 성능 개선 함수
- 모델(기본형) → hyperparameter들을 지정
- GridSearchCV 함수가 순차적으로 hyperparameter들을 변경해가면서 학습과 평가를 수행
- 가장 성능이 좋은 hyperparameter를 제시
- 모델 완성

GridSerchCV 를 이용한 모델 성능 개선

▶ GridSerchCV 사용법

```
# GridSearchCV 모델 생성에 필요한 함수 임포트
from sklearn.model_selection import GridSearchCV

# 최적화 할 기본 모델
dt = DecisionTreeClassifier(random_state=0)
params = {'max_depth':[3,4,5,6,7,8,9]}

# GridSearchCV 생성 함수 호출, 객체 생성
grid_dt = GridSearchCV(
    estimator=dt,
    param_grid=params,
    scoring='accuracy',
    cv=10
)

# 학습 및 평가
grid_dt.fit(X_train, y_train)
```

GridSerchCV를 이용한 모델 성능 개선

▶ 모델 완성

```
### 최적의 성능일 때의 정확도 확인  
print(grid_dt.best_score_)
```

```
0.7404380597210916
```

```
### 최적의 성능일때의 하이퍼파라미터 확인  
print(grid_dt.best_params_)
```

```
{'max_depth': 7}
```

```
### best 모델 생성  
best_dt = DecisionTreeClassifier(max_depth=7, random_state=0)
```

평가 방법 개선과 GirdSerchCV를 이용한 Decision Tree 실습

- ▶ Diabetes Health Indicators Dataset

앙상블(Ensemble)

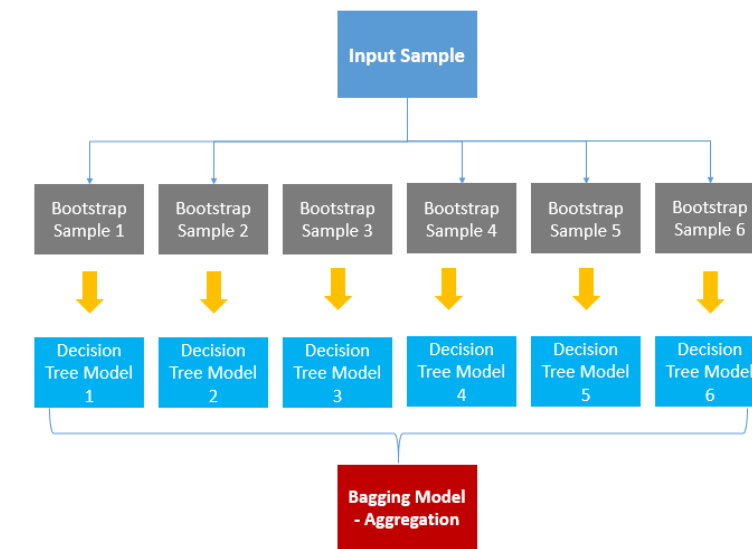
- 여러 개의 머신 러닝 모델을 연결하여 더 강력한 모델을 만드는 기법



- Bagging
- Boosting

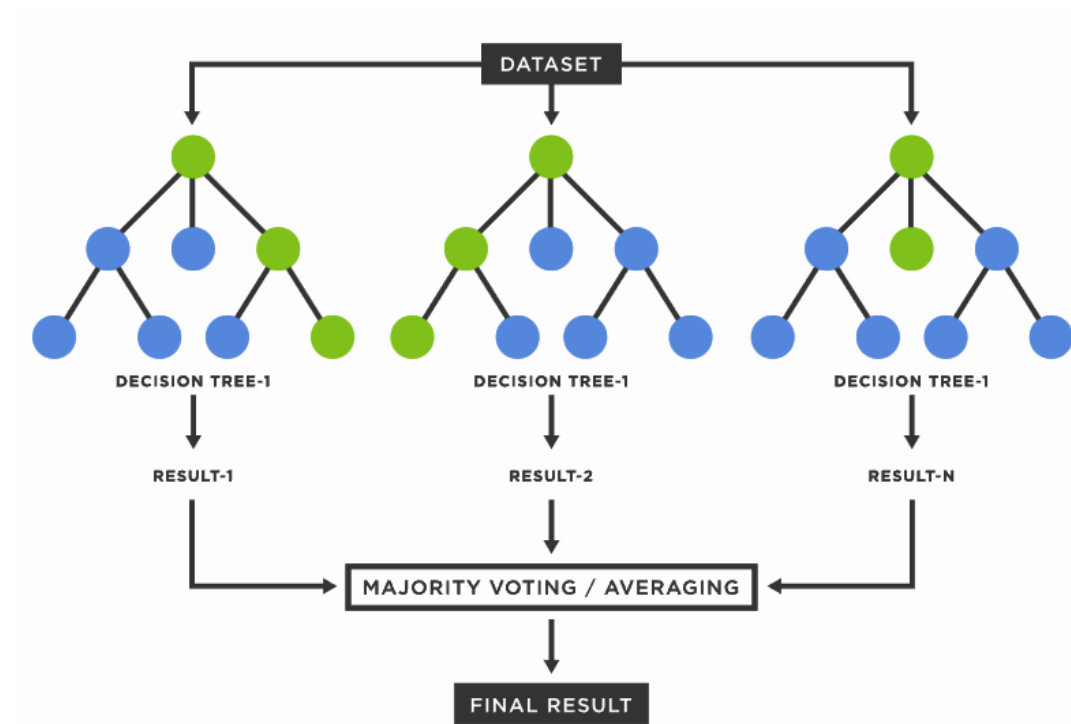
앙상블(Ensemble) - Bagging

- Bootstrap 방식의 데이터셋 생성 : 학습 데이터(Original Dataset)로부터 랜덤하게 '복원추출'하여 동일한 사이즈의 데이터셋 n 개 생성
- 다수결의 원칙 적용 : n 개의 Decision Tree 모델을 개별적으로 학습 후 각 모델의 예측 결과에 대해 가장 많은 값을 최종 예측 값으로 선정, 모델의 정확도를 향상



랜덤 포레스트(Random Forest)

- 여러 개의 Decision Tree 모델이 Bootstrap 방식으로 각자의 데이터를 샘플링, 개별적으로 학습 후 각 모델의 예측 결과에 대해 가장 많은 값을 최종 예측 값으로 선정(다수결의 원칙)
 - 다양한 모델을 만드는 두 가지 방법
 - 1) 데이터를 무작위로 선택(Bootstrap 방식)
 - 2) 분할의 기준이 되는 특성을 무작위로 선택($\sqrt{n_features}$)
 - 서로 다른 데이터를 가지고 다른 방향으로 학습된 모델들을 만들고 평균을 내어 일반화 시키는 모델
- 분류와 회귀가 모두 가능



랜덤 포레스트(Random Forest)

- ▶ 랜덤 포레스트(RandomForest) 분류 모델 사용법

필요한 라이브러리 임포트

from sklearn.ensemble **import** RandomForestClassifier

랜덤 포레스트 분류 모델 생성 함수 호출, 분류 모델 객체 생성

rf = **RandomForestClassifier**(hyperparameter)

랜덤 포레스트(Random Forest)

- ▶ 랜덤 포레스트(RandomForest) 분류 모델 + GridSearchCV() 함수

```
### GridSearchCV 함수 실행

# 최적화 할 대상 모델 생성
rf = RandomForestClassifier(random_state=0)

# 최적화 할 매개 변수 설정
params = {'n_estimators':[100, 200, 300, 400],
          'max_depth':[7,8,9]}

# GridSearchCV 함수 호출, 객체 생성
grid_rf = GridSearchCV(
    estimator=rf,
    param_grid=params,
    scoring='accuracy',
    cv=10
)

# 학습 및 평가
grid_rf.fit(X_train, y_train)
```

랜덤 포레스트(Random Forest)

▶ 모델 완성

```
### 최적의 성능 확인
```

```
print(grid_rf.best_score_)
```

```
0.749933611966729
```

```
### 최적의 하이퍼파라미터 조합 확인
```

```
print(grid_rf.best_params_)
```

```
{'max_depth': 9, 'n_estimators': 200}
```

```
### best 모델 생성
```

```
best_rf = RandomForestClassifier(n_estimators=200, max_depth=9, random_state=0)
```

랜덤 포레스트(Random Forest) 실습

- ▶ Diabetes Health Indicators Dataset

앙상블(Ensemble) – Gradient Boosting

- 여러 개의 모델을 순차적으로 학습하는 방식
- 잔차(residual, 오차, 손실, error) : 실제 정답과 예측 값의 차이
- 잔차(residual)를 최소화할 수 있도록 트리 모델을 순차적으로 학습
- 잔차(residual)를 순차적으로 개선하여 분류의 정확도를 향상

Gradient Boosting 알고리즘

- 1단계 : 초기 평균 값 \rightarrow 1차 예측 값으로 설정

Height (m)	Gender	Age	Weight (kg)
1.8	Male	27	88
1.7	Male	44	68
1.7	Male	58	76
1.5	Female	15	35
1.6	Female	25	54



평균 : 64.2

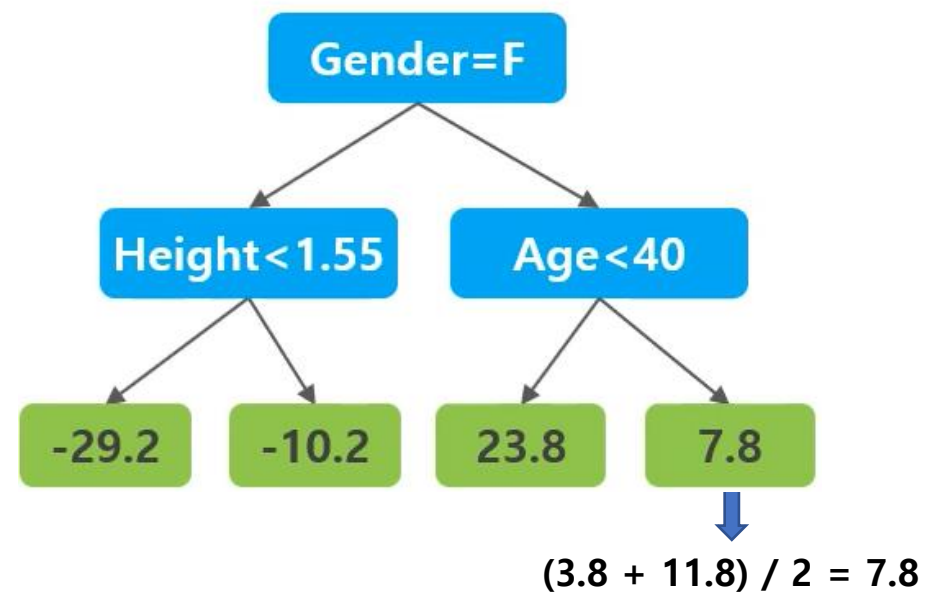
Height (m)	Gender	Age	Weight (kg)	Residual
1.8	Male	27	88	23.8
1.7	Male	44	68	3.8
1.7	Male	58	76	11.8
1.5	Female	15	35	-29.2
1.6	Female	25	54	-10.2

$76 - 64.2 = 11.8$

Gradient Boosting 알고리즘

- 2단계 : 1차 트리 생성

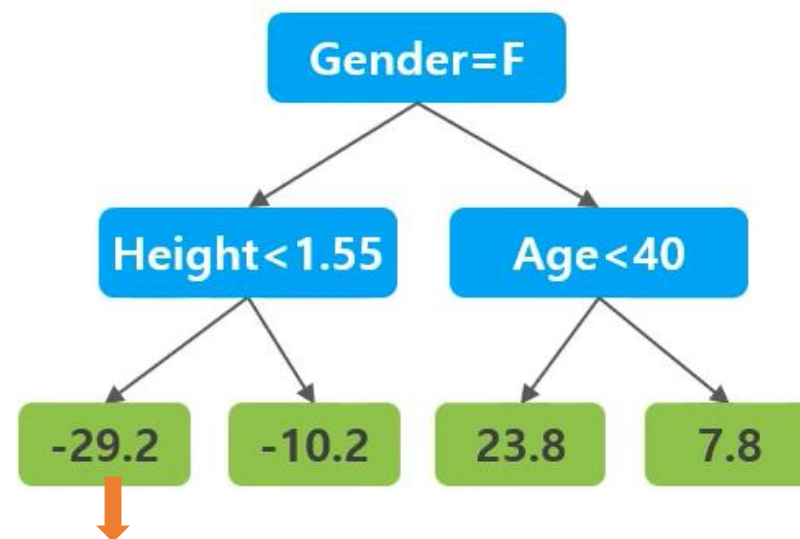
Height (m)	Gender	Age	Weight (kg)	Residual
1.8	Male	27	88	23.8
1.7	Male	44	68	3.8
1.7	Male	58	76	11.8
1.5	Female	15	35	-29.2
1.6	Female	25	54	-10.2



Gradient Boosting 알고리즘

- 3단계
 - 1차 잔차, learning_rate 이용 → 2차 예측 값 연산

Height (m)	Gender	Age	Weight (kg)	Residual
1.8	Male	27	88	23.8
1.7	Male	44	68	3.8
1.7	Male	58	76	11.8
1.5	Female	15	35	-29.2
1.6	Female	25	54	-10.2



predicted weight = 초기 평균값 + learning_rate X 1차 잔차
= 64.2 + (0.1 X -29.2)
= 61.28

Gradient Boosting 알고리즘

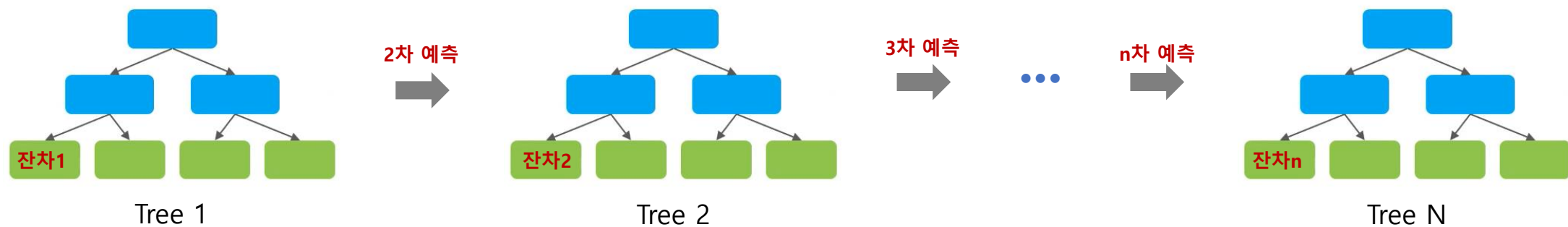
- 3단계
 - 1차 잔차 → 2차 잔차 연산

Height (m)	Gender	Age	Weight (kg)	Residual (1차)	Residual (2차)
1.8	Male	27	88	23.8	21.42
1.7	Male	44	68	3.8	$68 - (64.2 + (0.1 \times 7.8)) =$
1.7	Male	58	76	11.8	$76 - (64.2 + (0.1 \times 7.8)) =$
1.5	Female	15	35	-29.2	-26.28
1.6	Female	25	54	-10.2	-9.18

$$\begin{aligned}\text{2차 잔차} &= \text{Weight} - \text{predicted weight} \\ &= 35 - (64.2 + (0.1 \times -29.2)) \\ &= (35 - 64.2) - (0.1 \times -29.2) \\ &= \text{1차 잔차} - (0.1 \times \text{1차 잔차}) < \text{1차 잔차}\end{aligned}$$

Gradient Boosting 알고리즘

- 2차 트리 생성 → 3차 예측 값 & 잔차 계산 → ...
- 잔차를 최소화할 수 있도록 트리 모델을 순차적으로 학습



✓ 최종 예측 값 = 초기 평균값 + (learning_rate X 1차 잔차) + ... + (learning_rate X n차 잔차)

✓ 1차 잔차 > 2차 잔차 > ... > n차 잔차

LightGBM(Light Gradient Boosting Machine)

- MS
- 성능이 좋고, 학습에 걸리는 시간이 짧음
- 단점 : 작은 dataset을 사용할 경우 과대적합 발생 가능성이 큼 (일반적으로 10,000개 이하의 데이터를 적다고 함)

LightGBM(Light Gradient Boosting Machine)

▶ LightGBM 분류 모델 사용법

필요한 라이브러리 설치

```
!pip install lightgbm
```

필요한 라이브러리 임포트

```
from lightgbm import LGBMClassifier
```

분류 모델 생성 함수 호출, 분류 모델 객체 생성

```
lgbm_model = LGBMClassifier(hyperparameter)
```

LightGBM(Light Gradient Boosting Machine)

▶ 주요 매개변수(hyperparameter)

- n_estimators : default=100
- learning_rate : default=0.1
- max_depth : default = -1(가능한 최대)
- GridSerchCV 사용하여 튜닝

LightGBM(Light Gradient Boosting Machine) 실습

- ▶ Diabetes Health Indicators Dataset

08. 머신 러닝

- 지도 학습을 이용한 회귀

회귀(Regression)

- 개념

- 학습 데이터를 이용하여 데이터의 특성과 상관 관계 등을 파악하고, 그 결과를 바탕으로 학습 데이터에 없는 미지의 데이터가 주어졌을 경우에, 연속적인 숫자 값으로 예측하는 것
예) 공부 시간과 시험 성적의 관계

x(hour)	y(score)
9	90
8	80
4	40
2	20

시험성적 데이터

7시간 공부 할 경우
성적은 몇 점 일까?

회귀(Regression)

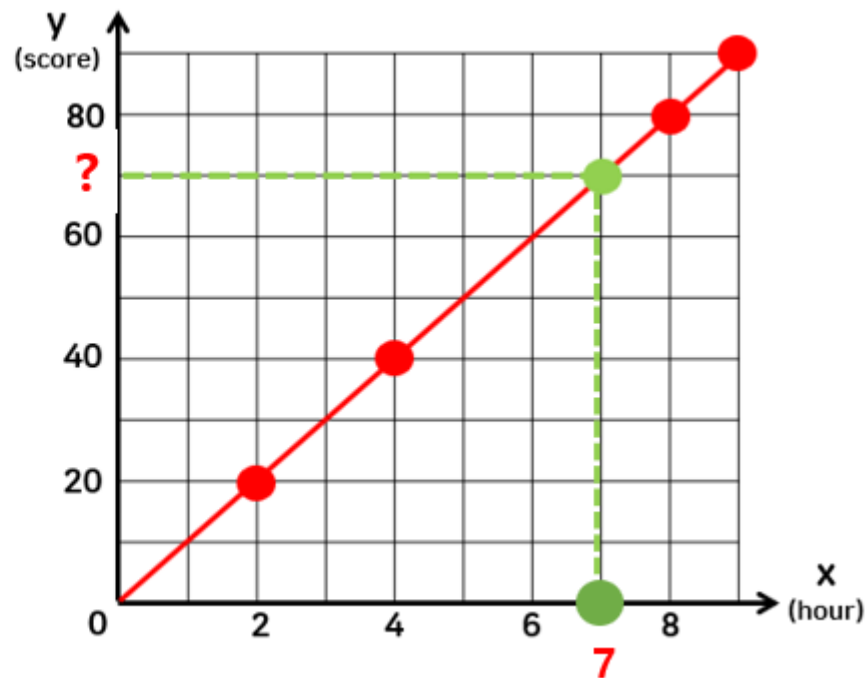
- 선형 회귀(Linear Regression)

- 개념 : 학습 데이터의 입력과 출력의 관계를 직선의 형태로 나타낼 수 있고, 미지의 데이터도 $y = Wx + b$ 의 값으로 예측할 수 있는 것

x(hour)	y(score)
9	90
8	80
4	40
2	20

시험성적 데이터

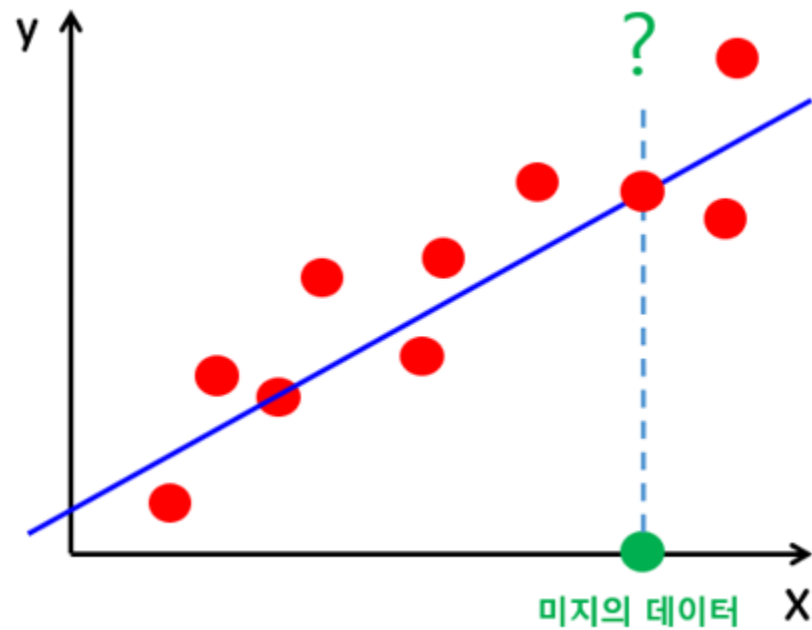
7시간 공부 시
성적 ?



회귀(Regression)

- 선형 회귀(Linear Regression)

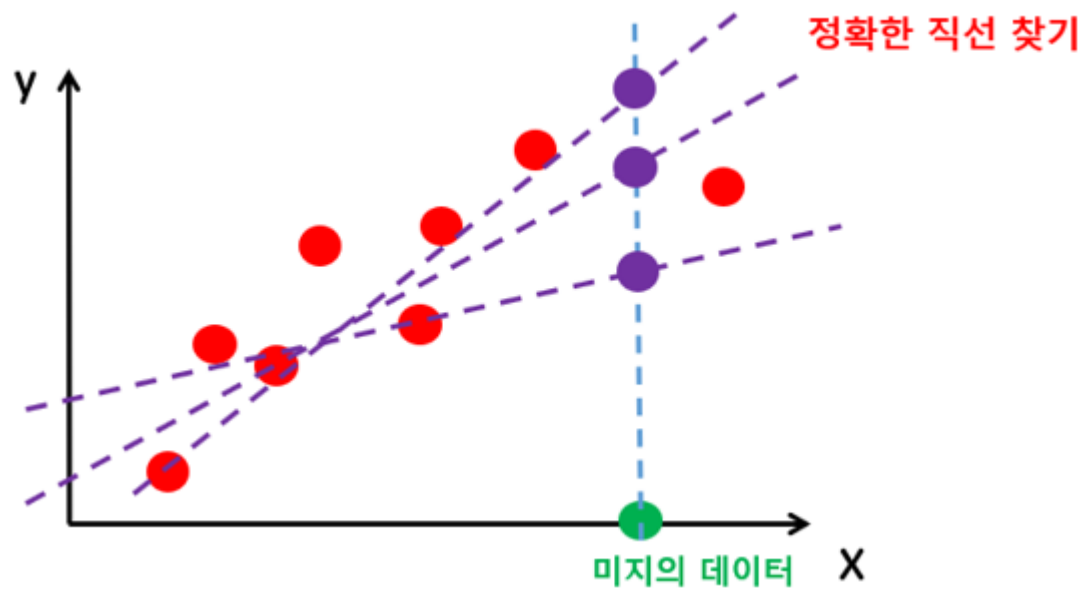
X_train	y_train
공부시간(x)	시험성적(t)
9	74
14	81
21	86
27	88
32	90
37	92



회귀(Regression)

- 선형 회귀(Linear Regression) 정리

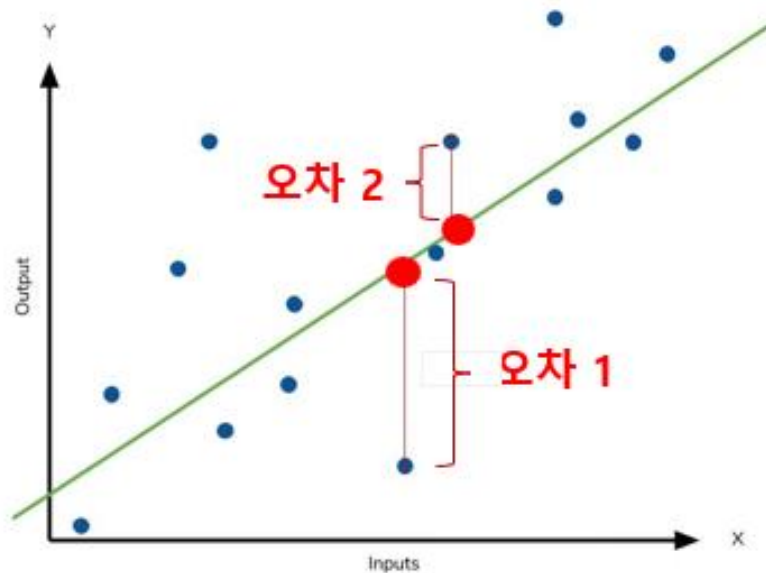
- 개념 : 학습 데이터에는 없는 미지의 데이터에 대한 값을 예측할 때, 데이터의 분포를 가장 잘 표현할 수 있는 직선을 그려서 값을 예측하는 방법



회귀(Regression)

- 선형 회귀 모델의 성능 평가 방법

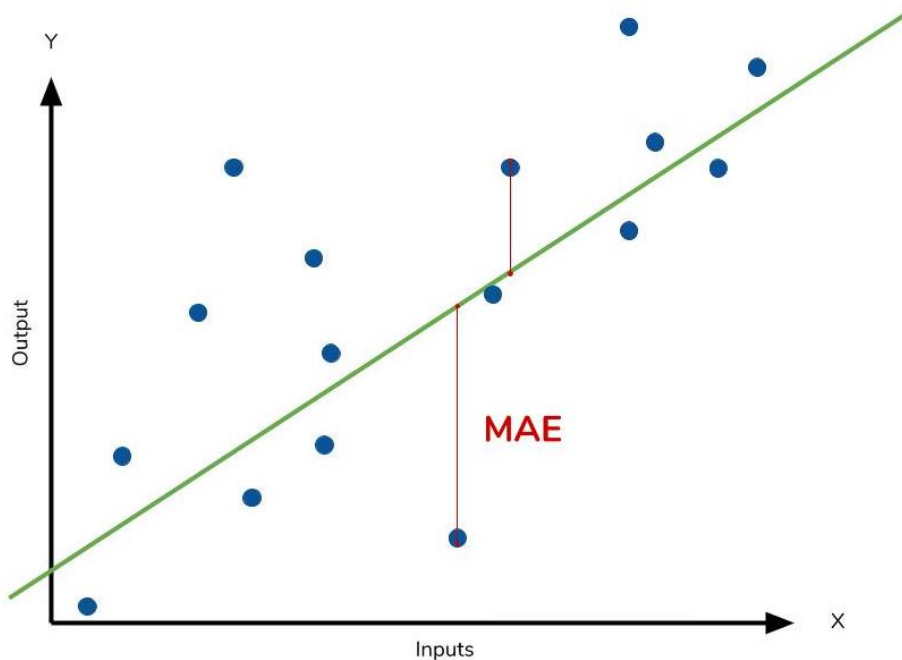
- 오차 또는 에러(Error) 또는 손실(loss)이 작을 수록 성능이 좋은 모델이다.



선형 회귀 모델의 성능 평가 방법

(1) 평균 절대값 오차(MAE : Mean Absolute Error)

- 개념 : 모델의 예측 값과 실제 값의 차이의 절대 값을 모두 더하여 평균을 구한다

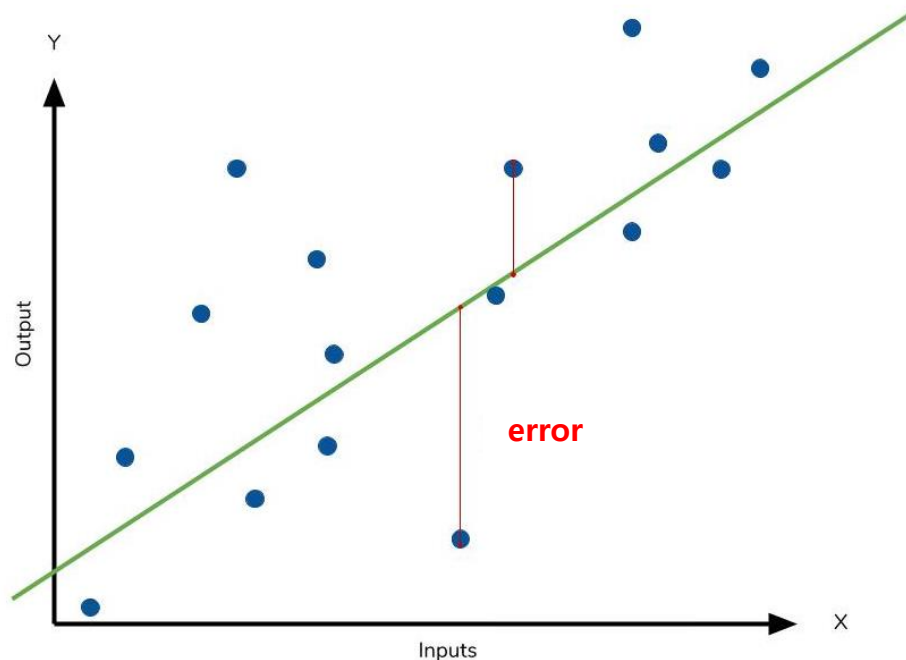


$$MAE = \frac{\sum |y - \hat{y}|}{n}$$

선형 회귀 모델의 성능 평가 방법

(2) 평균 제곱 오차(MSE : Mean Squared Error)

- 개념 : 모델의 예측 값과 실제 값의 차이의 제곱을 모두 더하여 평균을 구한다



$$MSE = \frac{\sum (y - \hat{y})^2}{n}$$

선형 회귀 모델 - LinearRegression

- 평균 제곱 오차(MSE)가 최소가 되는 직선을 구하여 값을 예측하는 선형 회귀 모델
- 사용법

```
# 필요한 라이브러리 임포트
```

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

```
# 선형 회귀 모델 생성 함수 호출, 회귀 모델 객체 생성
```

```
lr = LinearRegression()
```

```
# 선형 회귀 모델 학습, 예측
```

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

```
# 선형 회귀 모델 학습, 예측
```

```
y_pred = lr.predict(X_test)
```

보스턴 주택 가격 데이터 분석 실습

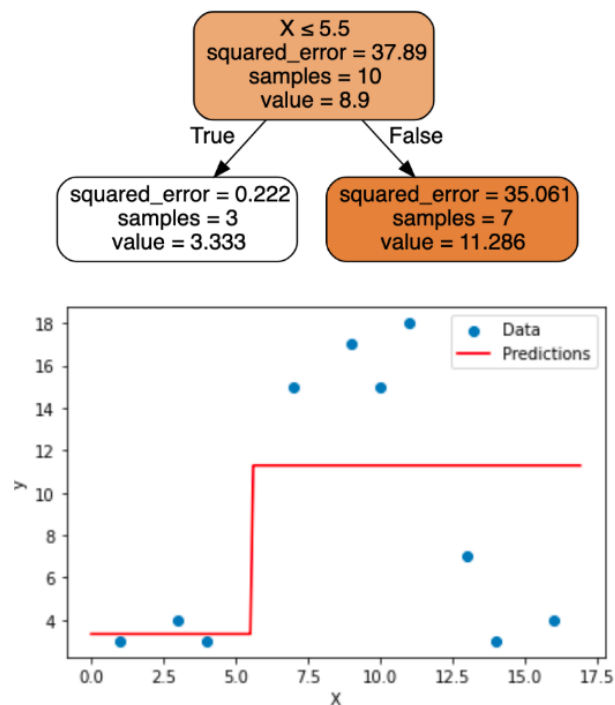
데이터의 구성 : 집 값에 영향을 주는 특성 13가지 + 정답 컬럼 1개, 506개 데이터

0	CRIM: 인구 1인당 범죄 발생 수
1	ZN: 25,000평방 피트 이상의 주거 구역 비중
2	INDUS: 소매업 외 상업이 차지하는 면적 비율
3	CHAS: 찰스강 위치 변수(1: 강 주변, 0: 이외)
4	NOX: 일산화질소 농도
5	RM: 집의 평균 방 수
6	AGE: 1940년 이전에 지어진 비율
7	DIS: 5가지 보스턴 시 고용 시설까지의 거리
8	RAD: 순환고속도로의 접근 용이성
9	TAX: \$10,000당 부동산 세율 총계
10	PTRATIO: 지역별 학생과 교사 비율
11	B: 지역별 흑인 비율
12	LSTAT: 급여가 낮은 직업에 종사하는 인구 비율(%)

분석 목표 : 주택 가격에 영향을 주는 특성 분석, 주택 가격을 예측하는 모델링, 모델 평가

회귀 트리(Regression Tree) 모델

- 오차를 가장 잘 줄일 수 있는 특성(변수)을 기준으로 트리를 만들어 값을 예측



- DecisionTree 회귀 모델
- RandomForest 회귀 모델
- LightGBM 회귀 모델

Decision Tree 회귀 모형

▶ 사용법

필요한 라이브러리 임포트

```
from sklearn.tree import DecisionTreeRegressor
```

결정 트리 회귀 모델 생성 함수 호출, 회귀 모델 객체 생성

```
dt = DecisionTreeRegressor(하이퍼파라미터, random_state)
```

결정 트리 회귀 모델 학습

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

결정 트리 회귀 모델 예측

```
dt.predict(X_test)
```

Random Forest 회귀 모형

▶ 사용법

```
# 필요한 라이브러리 임포트
```

```
from sklearn.ensemble import RandomForestRegressor
```

```
# 랜덤 포레스트 회귀 모델 생성 함수 호출, 회귀 모델 객체 생성
```

```
rf = RandomForestRegressor(하이퍼파라미터, random_state)
```

```
# 랜덤 포레스트 회귀 모델 학습
```

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

```
# 랜덤 포레스트 회귀 모델 예측
```

```
rf.predict(X_test)
```

LightGBM 회귀 모형

▶ 사용법

필요한 라이브러리 설치

```
!pip install lightgbm
```

필요한 라이브러리 임포트

```
from lightgbm import LGBMRegressor
```

LightGBM 회귀 모델 생성 함수 호출, 회귀 모델 객체 생성

```
lgbm = LGBMRegressor(하이퍼파라미터, random_state)
```

LightGBM 회귀 모델 학습

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

LightGBM 회귀 모델 예측

```
lgbm.predict(X_test)
```

회귀 트리(Regression Tree) 모델 실습

- ▶ 보스톤 주택 가격 예측