

# 빅 데이터 분석을 위한 알고리즘 활용

빅 데이터분석을 위한 알고리즘 소개 및 Python을 활용한 실습



## 2013.03. – 2015.01. 인하대학교 데이터 마이닝 연구소

- . 최적 용매 물질 분석을 위한 데이터 분석 및 처리 (인하대학교)
- . 지역별 알레르기 환자 특성 비교 프로젝트 (인하대 병원)
- . 수요예측을 위한 최적 시계열 모형 도출 알고리즘 개발 (인하대학교)
- . 한국 공항공사 데이터 분석 (사회 책임 윤리 경영 연구소)
- . 마케팅을 위한 고객 특성에 따른 군집화 알고리즘 생성 (SK Ubcare)

## 2015.01. – 2017.04. 통계 분석 전문 회사 (Begas)

- . 빅데이터 기반 추계모형 사전 컨설팅 (근로복지공단)
- . 분석 솔루션 개발 (베가스)
- . 제조 공정 질량분석 응용 시스템 개발을 위한 예측 모형 설계 및 개발 (중국 Baosteel)
- . 스마트 팩토리 솔루션 개발 (포스코 ICT)
- . 빅데이터 기반 범죄 분석 프로그램 개발 및 플랫폼 구축 (경찰청)

## 2017.06. – 2021.02. 인하대학교 고급 통계학 연구소

- . 가슴기살균제 건강피해 인정 및 판정기준 개선연구 (환경 독성 보건학회)

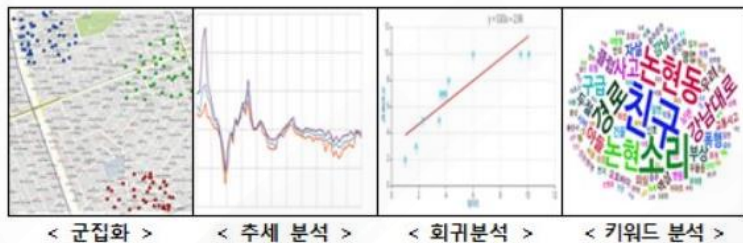
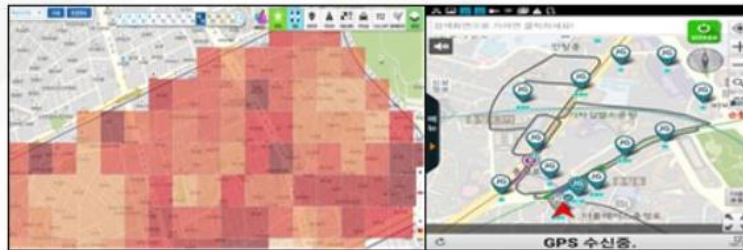
## 2021.01. – 2022.06. LG CNS

- . 분석 전문가를 위한 교육 프로그램 개발 (LG CNS)
- . 요금제 불만고객탐지 모델 개발 (LG U+)
- . 온라인 행동기반 기업고객 프로파일DB 구축 및 가입상담신청 예측모델개발(LG U+)
- . 영화 콘텐츠 평가 지표 고도화를 위한 지표 및 활용 방안 개발 (LG U+)
- . 불량 분석 솔루션 개발 (LG 에너지 솔루션)



# 데이터 분석 사례

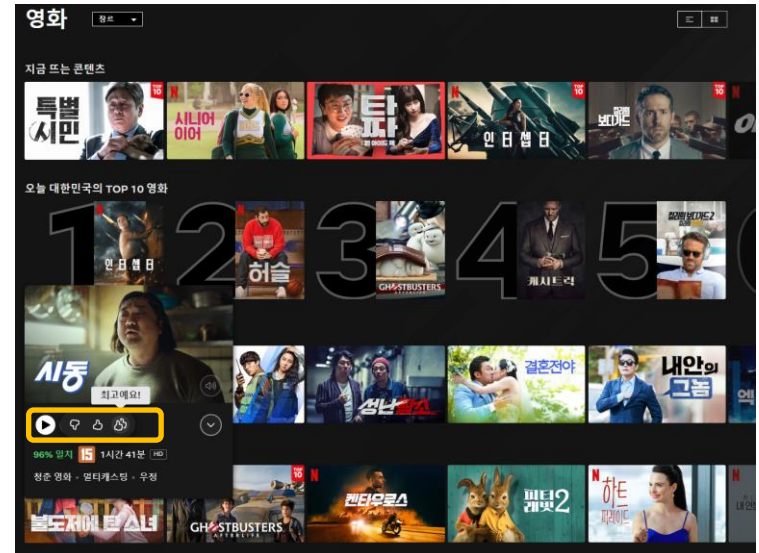
## 범죄 예측을 위한 데이터 분석



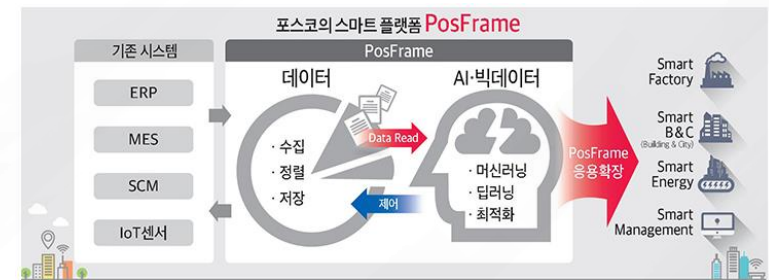
'범죄위험도 예측분석 시스템' 화면 [자료=경찰청]

\* 자료 출처  
<http://www.theviewers.co.kr/View.aspx?No=I520602>

## 넷플릭스 영화 추천 알고리즘 구축



## 제조업에서의 스마트 팩토리, 스마트 인더스트리



▲ 사진설명: 포스프레임(PosFrame)을 스마트인더스트리로 응용확장하는 과정

\* 자료 출처  
<https://newsroom.posco.com/kr/포스코-포스프레임posframe으로-스마트인더스트리-주도/>

# 데이터 분석 프로세스

데이터 분석 프로세스에는 1. 데이터 수집 2. 데이터 탐색 3. 모델링 4. 모형평가 단계로 나뉘집니다.

각 단계에서 가장 많은 시간이 소요되는 단계는 “데이터 탐색”으로 단순히 분석 알고리즘을 적용하는 것만큼 매우 중요한 단계 입니다.

01



## 데이터 수집

- 분석에 활용할 데이터 수집 및 추출

02



## 데이터 탐색

- 데이터 정제(전처리)
- 데이터 시각화
- 데이터 변환

03



## 모델링

- 분석 알고리즘 적용

04



## 모형 평가

- 알고리즘 성능 비교
- 알고리즘 성능 평가

# DATA Analysis

— 2022  
빅 데이터 분석을 위한 알고리즘 활용

## CONTENTS

01

### 데이터 수집 및 시각화

데이터 수집, 전처리, 시각화 과정에 대한 필요성을  
소개하고 간단한 실습을 통해 학습함

02

### 분석 알고리즘 I

선형 모형 중 선형 회귀 모형과 로지스틱 회귀 모형에  
대해 소개하고 간단한 실습을 통해 학습함

03

### 분석 알고리즘 II

의사결정나무 (Decision Tree)와 이를 기반으로 한 랜덤  
포레스트 모형에 대해 소개하고 간단한 실습을 통해 학습함

04

### 모형 평가

다양한 분석 알고리즘들에 대한 비교와 개발한 분석  
모형들에 대한 평가를 위한 방법에 대해 학습함

05

### 기타 알고리즘

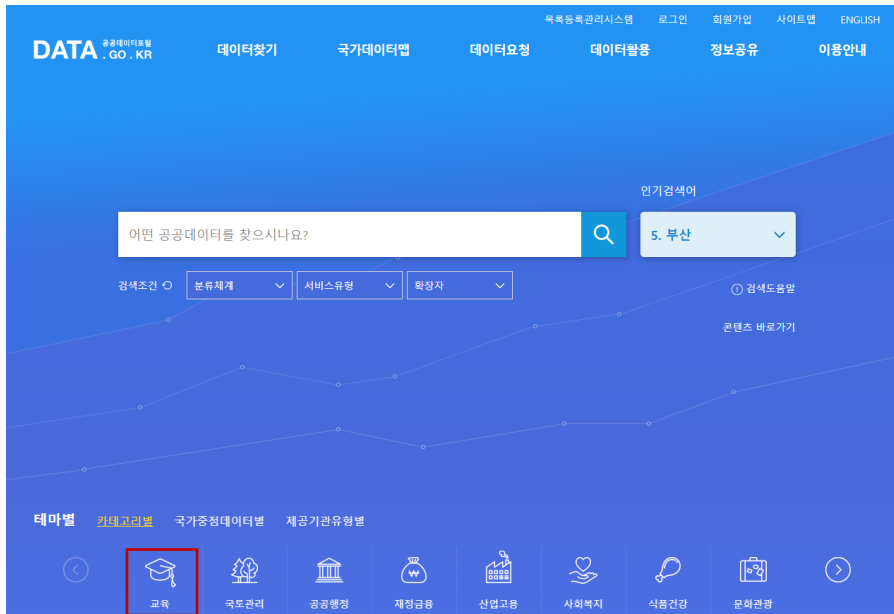
기본 분석 알고리즘 외 추천 시스템과 텍스트 마이닝에서  
사용되고 있는 알고리즘들에 대해 소개함



# 1.1 데이터 수집

## 01. 데이터 수집 및 시각화

- 활용해 볼 수 있는 데이터로 쉽게 수집할 수 있는 방법은 “**공공 데이터 포털**”을 활용하는 것 입니다. 이를 활용하여 데이터 분석 연습 뿐만 아니라 실제 분석 결과를 통한 활용도 가능합니다.
- 공공 데이터 포털 뿐만 아니라 행동 데이터를 의미하는 로그 데이터 혹은 웹사이트에 있는 영화 리뷰 데이터 등과 같은 정보도 크롤링을 통해 수집이 가능합니다.



• <https://www.data.go.kr/>





# 1.1 데이터 수집

## 01. 데이터 수집 및 시각화

- 활용해 볼 수 있는 데이터로 쉽게 수집할 수 있는 방법은 “공공 데이터 포털”을 활용하는 것 입니다. 이를 활용하여 데이터 분석 연습 뿐만 아니라 실제 분석 결과를 통한 활용도 가능합니다.
- 공공 데이터 포털 뿐만 아니라 행동 데이터를 의미하는 로그 데이터 혹은 웹사이트에 있는 영화 리뷰 데이터 등과 같은 정보도 크롤링을 통해 수집이 가능합니다.

**NAVER 영화**
로그인

영화홈  
상영작 · 예정작  
영화평점  
예매  
시사회 · 이벤트  
**평점 · 리뷰**  
네타즌 평점  
네타즌 리뷰  
올해의  
다운로드  
인더극장

**네타즌 평점 · 140자평**

현재 상영작 평점 · 140자평 보기
개봉 예정작 평점 · 140자평 보기

전체 리스트
총 10140589개의 평점 · 140자평이 있습니다

번호	평점	140자평	글쓴이 · 날짜
13452743	★★★★★ 1	미숙 영화가 이렇거면 팬티라도 보여주든가!! 신고	lmsk**** 17.12.03
13452742	★★★★★ 9	오리엔트 특급 살인 오빠는 지루하다 했지만 난 재밌었을 법인이 누군지 알게됨 올때 슬펐다 잔잔한 영화 신고	jade**** 17.12.03
13452741	★★★★★ 10	특시요전사 얼마만 보고왔습시다 둘다 너무너무 재밌게봤고 배우들이 너무 연기를 잘해서 정말 저때의 실존이물같았어요 한번 더	gta_**** 17.12.03

**kimi\*\*\*\* 님의 평점 리스트**
총 22개의 평점 · 140자평이 있습니다

번호	평점	140자평	글쓴이 · 날짜
13408277	★★★★★ 5	저스티스 리그 디자인을 위한 희생양일뿐, 신고	kimi**** 17.11.19
12782392	★★★★★ 7	육자 뻔한 시나리오, 평론가들의 평점이 어이없다 신고	kimi**** 17.06.30
11603663	★★★★★ 8	터널 영화 베리드를 가지고 한국재난 영화로 더 잘 풀어낸것같 다.하정우라는 배우가 아니었으면 너무 지루했을듯 신고	kimi**** 16.08.11
11586013	★★★★★ 6	수어사이드 스쿼드 초반의 개성있는 캐릭터와 분위기는 좋았는데 스토리가 최 악이다, 마녀로 설정한부분이 너무 과했고 개성을 내려고 노력하는데 마블따라하기밖에 안보인다. 차라리 할리퀸과 조커의 애기로만 영화 만들었어도 재밌었을듯 신고	kimi**** 16.08.08

A	B	C	D	F	G
id	nickname	date	movieID	point	movie
13369433	wnsr****	2014-02-08	100654	10	또 하나의 약속(100654)
13369432	cjfw****	2017-05-12	58085	1	거룩한 계보(58085)
13369424	roma****	2017-08-28	137696	9	장산범(137696)
13369423	yuns****	2014-09-09	72522	8	감기(72522)
13369419	saty****	2017-06-28	88253	1	늑대소년(88253)
13369416	kso7****	2015-11-24	137991	10	프리덤(137991)
13369413	jsp6****	2017-08-21	65540	8	바르게 살자(65540)
13369413	jsp6****	2017-08-21	50598	9	A-특공대(50598)
13369413	jsp6****	2017-07-21	72408	9	악마를 보았다(72408)
13369413	jsp6****	2017-07-21	85640	9	파파로티(85640)
13369412	dlsd****	2017-07-18	155256	2	악녀(155256)
13369412	dlsd****	2015-03-21	122457	4	소셜포비아(122457)
13369412	dlsd****	2015-02-17	31801	10	봄날은 간다(31801)
13369412	dlsd****	2015-02-05	45290	10	인터스텔라(45290)
13369412	dlsd****	2015-01-22	34227	10	아이 엠 샘(34227)
13369412	dlsd****	2014-12-18	93756	10	명량(93756)
13369409	love****	2016-08-17	144968	8	제이슨 본(144968)
13369409	love****	2016-07-25	123519	7	아가씨(123519)
13369409	love****	2015-10-07	115977	10	베테랑(115977)
13369409	love****	2015-10-05	121922	10	사도(121922)
13369409	love****	2015-08-29	95541	10	미션 임파서블: 로그네이션(95541)
13369400	brai****	2016-01-26	126389	10	무서운 집(126389)
13369400	brai****	2015-11-27	129050	9	뷰티 인사이드(129050)
13369400	brai****	2014-06-22	17116	9	워터월드(17116)
13369400	brai****	2014-05-16	11354	10	복수혈전(11354)
13369400	brai****	2014-02-02	73372	10	세 열간이(73372)
13369400	brai****	2013-12-23	101901	8	변호인(101901)
13369400	brai****	2013-10-30	70773	10	바람(70773)
13369400	brai****	2013-10-20	41705	8	한반도(41705)
13369400	brai****	2013-10-20	98420	9	악의교전(98420)
13369396	rlag****	2016-06-21	122133	7	남과 여(122133)

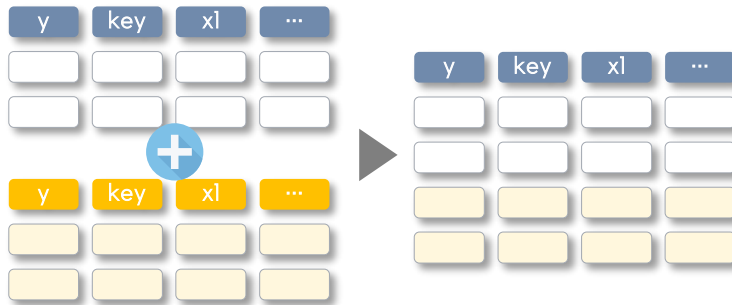


- 데이터 전처리에 사용되는 여러가지 활동 중 여러 “데이터를 결합”하는 방법과 “불필요한 데이터를 제거”하는 방법에 대해 파이썬을 활용한 실습을 통해 알아보도록 하겠습니다.
- 데이터를 결합하는 방법에는 “Union”과 “Left Join”에 대해 알아보도록 하겠습니다.

### 데이터 결합 (Union)

#### ■ 데이터 결합 (Union)

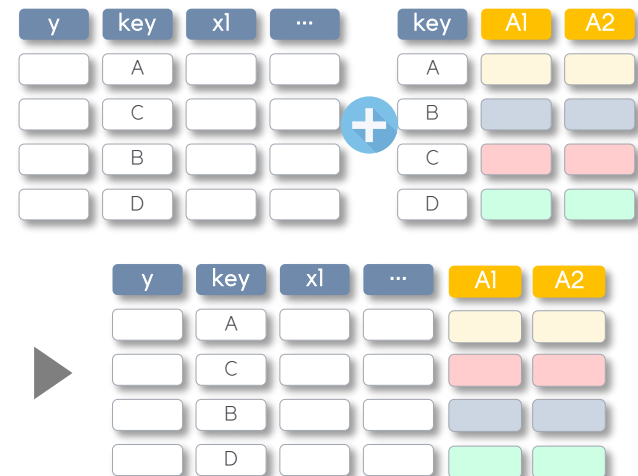
- Union은 데이터 결합의 방법 중 하나로 2개 이상의 데이터를 가로 방향으로 기존 데이터 아래 결합시키는 방법을 의미함
- 기존 데이터 아래로 결합시키는 경우 동일한 컬럼을 기준으로 결합시킴



### 데이터 결합 (Left Join)

#### ■ 데이터 결합 (Left Join)

- Left Join은 데이터 결합의 방법 중 하나로 왼쪽 데이터를 기준으로 오른쪽 데이터를 옆으로 결합시키는 방법을 의미함
- 왼쪽 데이터에 오른쪽 데이터를 결합시키는 경우 “key” 컬럼 이라고 불러주는 컬럼을 기준으로 결합시킴




- 데이터 전처리에 사용되는 여러가지 활동 중 여러 “데이터를 결합”하는 방법과 “불필요한 데이터를 제거”하는 방법에 대해 파이썬을 활용한 실습을 통해 알아보도록 하겠습니다.
- 데이터에 불필요한 데이터를 제거 하는 방법에 대해 알아보도록 하겠습니다.

### 불필요한 데이터 제거

#### ■ 불필요한 데이터 제거

- 데이터 내에서 분석에 포함되면 안되는 데이터가 섞여있는 경우 해당 데이터로 인해 분석 결과가 잘못되는 경우가 발생하거나, 분석을 수행할 수 없는 경우가 발생함
- 이런 경우 데이터 내에서 불필요한 데이터가 있는지 확인하고, 해당하는 데이터를 제거해야 함

y	key	x1	...
		1	
		1.4	
		1.2	
		9999	
		0.98	



y	key	x1	...
		1	
		1.4	
		1.2	
		0.98	

### 실습 데이터

#### ■ 피자 체인 데이터

- m\_area.csv : 지역 마스터 데이터 (시/도/군/구 정보)
- m\_store.csv : 매장 마스터 데이터 (매장 이름 등)
- tb\_order\_202104 : 피자 체인의 2021년 4월 주문 데이터
- tb\_order\_202105 : 피자 체인의 2021년 5월 주문 데이터

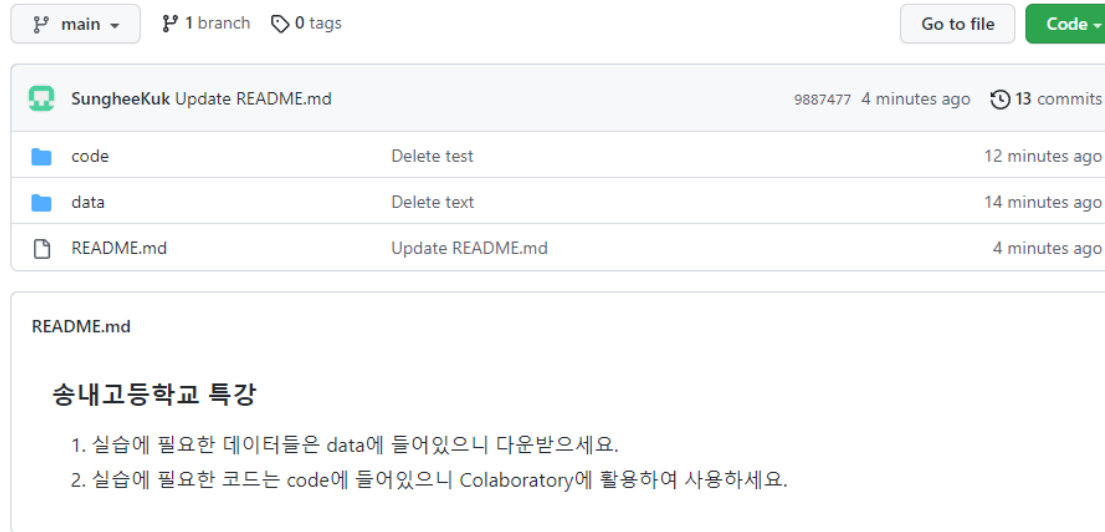


해당 데이터는 파이썬 머신러닝 실무테크닉 100 에서 발췌하였음

# 1.3 실습 (실습 데이터 다운로드)

## 01. 데이터 수집 및 시각화

- <https://github.com/SungheeKuk/songnae> 접속해서 data에 들어가 데이터를 다운받습니다.



SungheeKuk Update README.md 9887477 4 minutes ago 13 commits

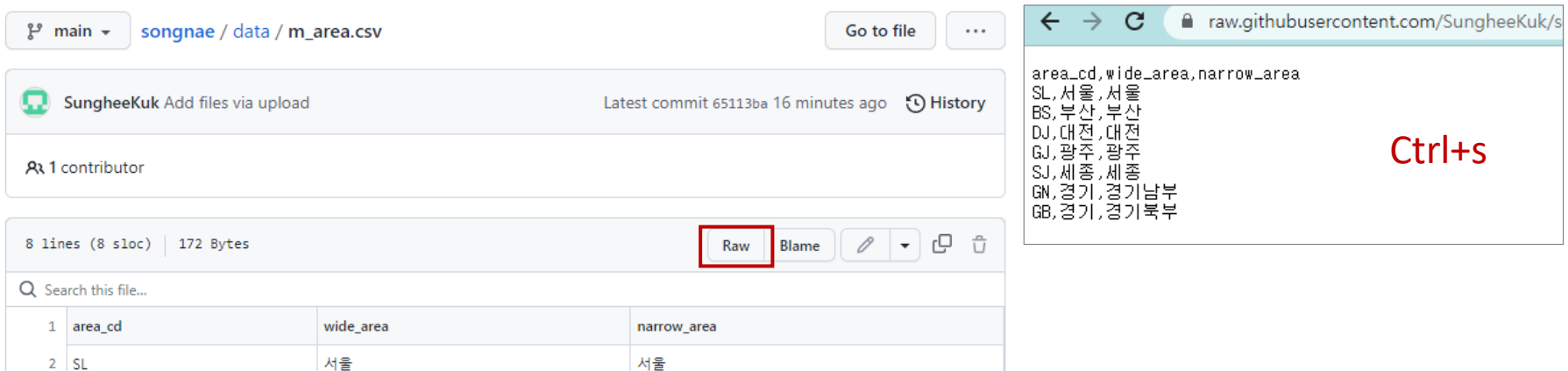
File	Commit Message	Time
code	Delete test	12 minutes ago
data	Delete text	14 minutes ago
README.md	Update README.md	4 minutes ago

README.md

### 송내고등학교 특강

1. 실습에 필요한 데이터들은 data에 들어있으니 다운받으세요.
2. 실습에 필요한 코드는 code에 들어있으니 Colaboratory에 활용하여 사용하세요.

- data에 들어가 csv file을 클릭한 후, Raw를 클릭합니다. 그 후, 데이터가 보이면 Ctrl+s 를 클릭하여 저장합니다.



SungheeKuk Add files via upload Latest commit 65113ba 16 minutes ago History

1 contributor

8 lines (8 sloc) | 172 Bytes

Raw Blame

Line	area_cd	wide_area	narrow_area
1	area_cd	wide_area	narrow_area
2	SL	서울	서울

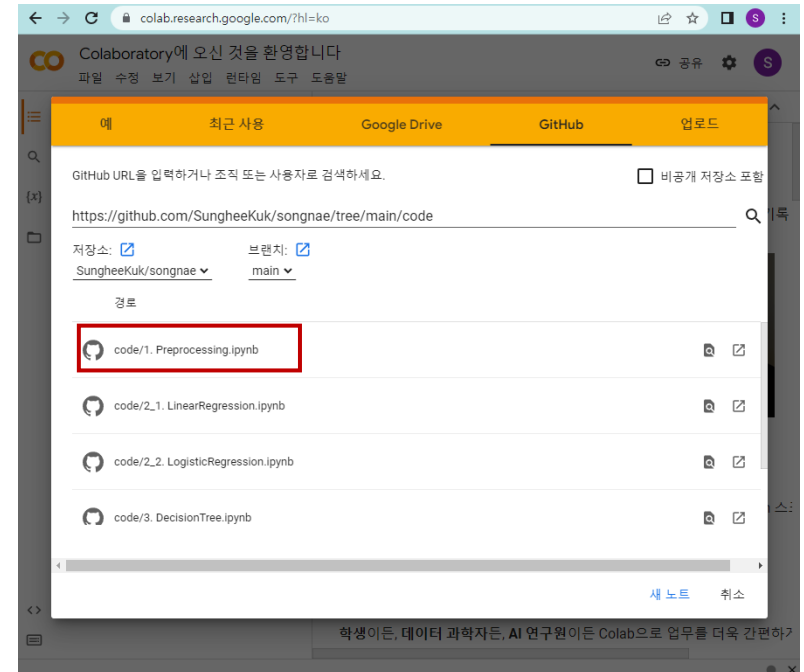
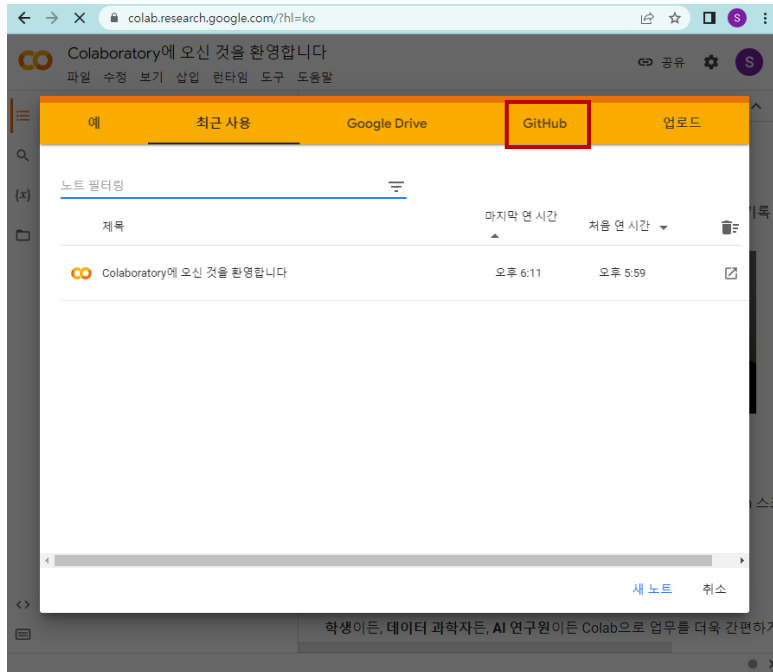
area\_cd,wide\_area,narrow\_area  
SL,서울,서울  
BS,부산,부산  
DJ,대전,대전  
GJ,광주,광주  
SJ,세종,세종  
GN,경기,경기남부  
GB,경기,경기북부

Ctrl+s

# 1.3 실습 (Colaboratory 사용)

## 01. 데이터 수집 및 시각화

- 구글에서 코랩을 검색하거나 <https://colab.research.google.com>에 직접 접속합니다.
- GitHub를 클릭합니다.
- <https://github.com/SungheeKuk/songnae/tree/main/code>를 입력합니다.
- 수업 순서에 맞는 실습 코드 파일을 선택하여 노트를 엽니다.



- 각 셀들을 실행시키고 싶은 경우 셀을 클릭 후, Ctrl+Enter 혹은 Shift+Enter를 누릅니다.

```
import pandas as pd
import numpy as np
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
```

# 1.3 실습 (Colaboratory 사용)

## 01. 데이터 수집 및 시각화

- 다운 받은 데이터를 코랩에 넣기 위해 폴더 모양을 클릭한 후, 다운받은 폴더에서 드래그로 넣어줍니다.

The image is a composite of three screenshots from a Google Colaboratory interface, illustrating the process of uploading data files.

**Top Left Screenshot:** Shows the Colaboratory file explorer. A folder icon in the left sidebar is highlighted with a red box. A blue arrow points from this icon to the bottom right screenshot.

**Top Right Screenshot:** Shows a file selection dialog box. The 'DAT' folder is selected, and several CSV files are listed: `m_area.csv`, `m_store.csv`, `tb_order_202104.csv`, and `tb_order_202105.csv`.

**Bottom Screenshot:** Shows the Colaboratory code editor. The file explorer on the left shows the `sample_data` folder. A blue box with the number '4' and an '이동' (Move) button is overlaid on the file explorer. The code editor contains the following code:

```
#!sudo apt-get install -y fonts-nanum
#!sudo fc-cache -fv
#!rm ~/.cache/matplotlib-rf

[ ] import pandas as pd
import numpy as np
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"

# 필요한 데이터 읽기
tb_order_04 = pd.read_csv('tb_order_202105.csv')
tb_order_05 = pd.read_csv('tb_order_202104.csv')
tb_order_04.head(10)
tb_order_04.shape
tb_order_05.head(10)
tb_order_05.shape
```

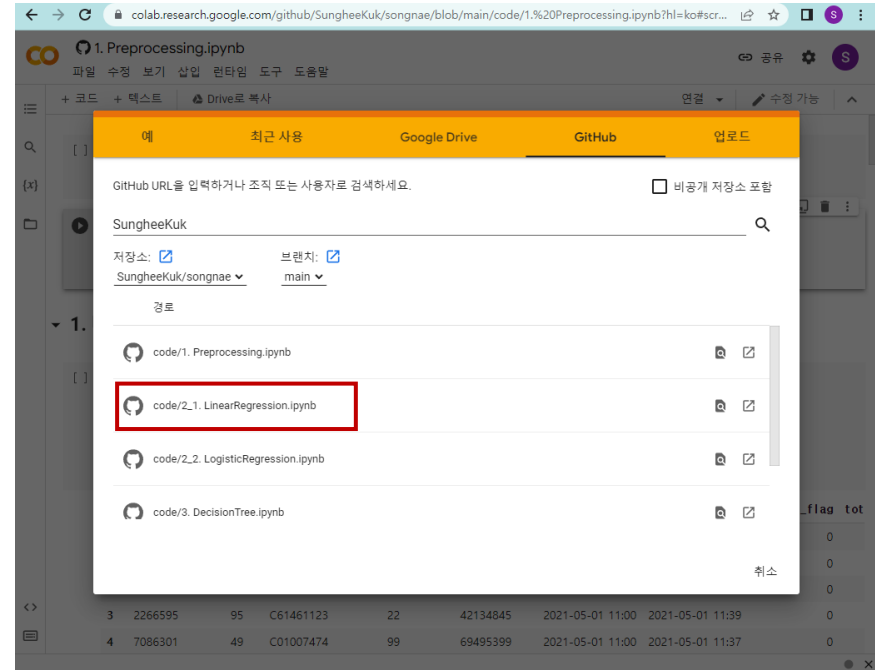
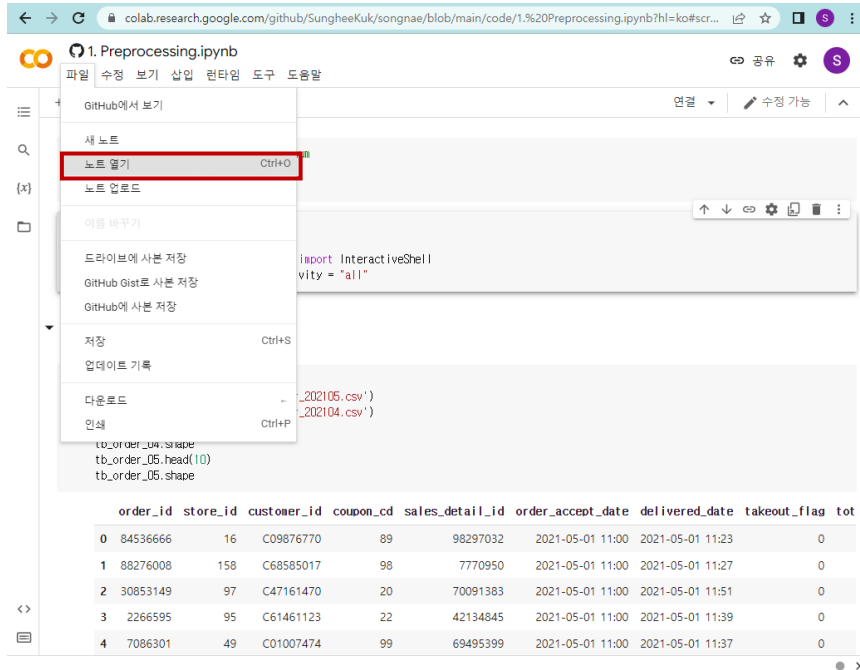
Below the code, a table displays the data:

	order_id	store_id	customer_id	coupon_cd	sales_detail_id	order_accept_id
0	84536666	16	C09876770	89	98297032	2021-05-01 11
1	88276008	158	C68585017	98	7770950	2021-05-01 11
2	30853149	97	C47161470	20	70091383	2021-05-01 11
3	2266595	95	C61461123	22	42134845	2021-05-01 11
4	7086301	19	C01007171	99	69495399	2021-05-01 11

# 1.3 실습 (Colaboratory 사용)

## 01. 데이터 수집 및 시각화

- 새로운 노트를 실행하고 싶은 경우 노트 열기를 클릭하여 동일하게 GitHub를 선택하면 이 전에 연결했던 Github내 코드 리스트가 있으니 동일하게 사용합니다.



- 새로운 노트를 사용합니다.

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from sklearn.datasets import load_boston
%matplotlib inline
```

# 1.3 실습 (데이터 전처리)

## 01. 데이터 수집 및 시각화

```
[22] import pandas as pd
import numpy as np
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
```

```
[23] # 필요한 데이터 읽기
tb_order_04 = pd.read_csv('tb_order_202105.csv')
tb_order_05 = pd.read_csv('tb_order_202104.csv')
tb_order_04.head(10)
tb_order_04.shape
tb_order_05.head(10)
tb_order_05.shape
```

	order_id	store_id	customer_id	coupon_cd	sales_detail_id	order_accept_date	delivered_date	takeout_flag	total_amount	status
0	84536666	16	C09876770	89	98297032	2021-05-01 11:00	2021-05-01 11:23	0	32340	2
1	88276008	158	C68585017	98	7770950	2021-05-01 11:00	2021-05-01 11:27	0	38650	2
2	30853149	97	C47161470	20	70091383	2021-05-01 11:00	2021-05-01 11:51	0	28270	2
3	2266595	95	C61461123	22	42134845	2021-05-01 11:00	2021-05-01 11:39	0	23160	2
4	7086301	49	C01007474	99	69495399	2021-05-01 11:00	2021-05-01 11:37	0	19000	9
5	92005874	104	C00574154	80	58592863	2021-05-01 11:00	2021-05-01 11:59	0	46240	2
6	8243996	171	C79994479	28	16376941	2021-05-01 11:00	2021-05-01 11:43	0	20640	2
7	62015909	191	C41716522	69	77974780	2021-05-01 11:00	2021-05-01 11:54	0	27320	2
8	23354107	108	C04237650	26	18054006	2021-05-01 11:00	2021-05-01 11:50	0	29870	2
9	82121011	123	C95413238	22	11628886	2021-05-01 11:00	2021-05-01 11:10	0	27910	2

(241142, 10)

	order_id	store_id	customer_id	coupon_cd	sales_detail_id	order_accept_date	delivered_date	takeout_flag	total_amount	status
0	34104383	11	C65806632	57	61573513	2021-04-01 11:00	2021-04-01 11:39	1	28270	1
1	70652318	59	C09760173	37	54068709	2021-04-01 11:00	2021-04-01 11:34	0	28270	2
2	71640388	195	C61227084	17	93678366	2021-04-01 11:00	2021-04-01 11:54	0	26470	9
3	75673365	127	C64119972	17	5287952	2021-04-01 11:00	2021-04-01 11:17	0	23080	2
4	9077529	174	C10231192	18	18248867	2021-04-01 11:00	2021-04-01 11:35	0	46920	2
5	86102793	167	C06298599	21	70395221	2021-04-01 11:00	2021-04-01 11:59	1	37420	1
6	49394078	187	C99985130	97	31854449	2021-04-01 11:00	2021-04-01 11:31	0	38650	9
7	10290387	30	C17281363	75	81945254	2021-04-01 11:00	2021-04-01 11:16	1	22380	1
8	54821099	9	C16681192	90	9773815	2021-04-01 11:00	2021-04-01 11:35	0	23080	2
9	42415936	156	C00944252	84	82103008	2021-04-01 11:00	2021-04-01 11:30	1	23120	1

(233262, 10)



# 1.3 실습 (데이터 전처리)

## 01. 데이터 수집 및 시각화

### 1.1. 데이터 결합 (유니온) 하기

```
[24] # 데이터 결합하기
tb_order_all = pd.concat([tb_order_04, tb_order_05])
tb_order_all.head()
tb_order_all.tail()
tb_order_all.shape
```

	order_id	store_id	customer_id	coupon_cd	sales_detail_id	order_accept_date	delivered_date	takeout_flag	total_amount	status
0	84536666	16	C09876770	89	98297032	2021-05-01 11:00	2021-05-01 11:23	0	32340	2
1	88276008	158	C68585017	98	7770950	2021-05-01 11:00	2021-05-01 11:27	0	38650	2
2	30853149	97	C47161470	20	70091383	2021-05-01 11:00	2021-05-01 11:51	0	28270	2
3	2266595	95	C61461123	22	42134845	2021-05-01 11:00	2021-05-01 11:39	0	23160	2
4	7086301	49	C01007474	99	69495399	2021-05-01 11:00	2021-05-01 11:37	0	19000	9
	order_id	store_id	customer_id	coupon_cd	sales_detail_id	order_accept_date	delivered_date	takeout_flag	total_amount	status
233257	61665702	103	C51797758	48	77989403	2021-04-30 21:58	2021-04-30 22:36	0	35300	2
233258	31151690	119	C04883863	78	16130893	2021-04-30 21:58	2021-04-30 22:35	0	32340	2
233259	15955692	175	C87594637	96	11457934	2021-04-30 21:58	2021-04-30 22:32	0	36170	9
233260	73251339	145	C93839111	92	64743537	2021-04-30 21:58	2021-04-30 22:46	0	36170	2
233261	42442108	117	C44570215	54	2304995	2021-04-30 21:58	2021-04-30 22:37	1	18990	1

(474404, 10)

```
[25] # 중복 제거 하기
tb_order_all.drop_duplicates(inplace = True)
```

```
# Index 초기화 하기
tb_order_all.reset_index(drop = True, inplace = True)
tb_order_all.head()
tb_order_all.tail()
```

	order_id	store_id	customer_id	coupon_cd	sales_detail_id	order_accept_date	delivered_date	takeout_flag	total_amount	status
0	84536666	16	C09876770	89	98297032	2021-05-01 11:00	2021-05-01 11:23	0	32340	2
1	88276008	158	C68585017	98	7770950	2021-05-01 11:00	2021-05-01 11:27	0	38650	2
2	30853149	97	C47161470	20	70091383	2021-05-01 11:00	2021-05-01 11:51	0	28270	2
3	2266595	95	C61461123	22	42134845	2021-05-01 11:00	2021-05-01 11:39	0	23160	2
4	7086301	49	C01007474	99	69495399	2021-05-01 11:00	2021-05-01 11:37	0	19000	9
	order_id	store_id	customer_id	coupon_cd	sales_detail_id	order_accept_date	delivered_date	takeout_flag	total_amount	status
474399	61665702	103	C51797758	48	77989403	2021-04-30 21:58	2021-04-30 22:36	0	35300	2
474400	31151690	119	C04883863	78	16130893	2021-04-30 21:58	2021-04-30 22:35	0	32340	2
474401	15955692	175	C87594637	96	11457934	2021-04-30 21:58	2021-04-30 22:32	0	36170	9
474402	73251339	145	C93839111	92	64743537	2021-04-30 21:58	2021-04-30 22:46	0	36170	2
474403	42442108	117	C44570215	54	2304995	2021-04-30 21:58	2021-04-30 22:37	1	18990	1

# 1.3 실습 (데이터 전처리)

## 01. 데이터 수집 및 시각화

### 1.2. 데이터 결합(Left Join)하기

```
[16] # 결합시킬 데이터 불러오기
m_store = pd.read_csv('m_store.csv')
m_store.head()

# 왼쪽에 기준이 될 데이터
tb_order_all.head()
```

	store_id	store_name	area_cd
0	1	삼일대로점	SL
1	2	세종대로점	SL
2	3	무교로점	SL
3	4	덕수궁길점	SL
4	5	서소문로점	SL

	order_id	store_id	customer_id	coupon_cd	sales_detail_id	order_accept_date	delivered_date	takeout_flag	total_amount	status
0	84536666	16	C09876770	89	98297032	2021-05-01 11:00	2021-05-01 11:23	0	32340	2
1	88276008	158	C68585017	98	7770950	2021-05-01 11:00	2021-05-01 11:27	0	38650	2
2	30853149	97	C47161470	20	70091383	2021-05-01 11:00	2021-05-01 11:51	0	28270	2
3	2266595	95	C61461123	22	42134845	2021-05-01 11:00	2021-05-01 11:39	0	23160	2
4	7086301	49	C01007474	99	69495399	2021-05-01 11:00	2021-05-01 11:37	0	19000	9

```
# 데이터 결합 (Left Join)
# key 컬럼 : store_id
order_df = pd.merge(tb_order_all, m_store, on='store_id', how='left')
order_df.head()
```

	order_id	store_id	customer_id	coupon_cd	sales_detail_id	order_accept_date	delivered_date	takeout_flag	total_amount	status	store_name	area_cd
0	84536666	16	C09876770	89	98297032	2021-05-01 11:00	2021-05-01 11:23	0	32340	2	삼릉로점	SL
1	88276008	158	C68585017	98	7770950	2021-05-01 11:00	2021-05-01 11:27	0	38650	2	광산로점	GJ
2	30853149	97	C47161470	20	70091383	2021-05-01 11:00	2021-05-01 11:51	0	28270	2	연재로점	BS
3	2266595	95	C61461123	22	42134845	2021-05-01 11:00	2021-05-01 11:39	0	23160	2	중앙대로2점	BS
4	7086301	49	C01007474	99	69495399	2021-05-01 11:00	2021-05-01 11:37	0	19000	9	당산로2점	SL

# 1.3 실습 (데이터 전처리)

## 01. 데이터 수집 및 시각화

### 1.3. 불필요한 데이터 제거하기

```
[20] # store_id가 999로 이상한 관측값이 포함되어있음  
order_df.loc[order_df['store_id'] == 999]
```

	order_id	store_id	customer_id	coupon_cd	sales_detail_id	order_accept_date	delivered_date	takeout_flag	total_amount	status	store_name	area_cd
90	50767209	999	C42395124	54	38116855	2021-05-01 11:07	2021-05-01 11:19	0	39010	2	보수담당	SL
175	26751004	999	C04230773	54	87066848	2021-05-01 11:16	2021-05-01 11:37	1	36170	1	보수담당	SL
316	18731635	999	C15370333	81	27190641	2021-05-01 11:29	2021-05-01 11:40	0	22380	2	보수담당	SL
600	98869612	999	C25277075	57	63594481	2021-05-01 11:52	2021-05-01 12:08	0	19000	2	보수담당	SL
660	43767543	999	C59856612	34	1557883	2021-05-01 11:56	2021-05-01 12:20	0	21220	2	보수담당	SL
...	...	...	...	...	...	...	...	...	...	...	...	...
473027	52648993	999	C32819157	11	83445699	2021-04-30 20:03	2021-04-30 20:24	1	21120	9	보수담당	SL
473128	42384749	999	C32393381	57	86751718	2021-04-30 20:11	2021-04-30 20:28	0	39010	2	보수담당	SL
473660	40214811	999	C63457263	53	36022036	2021-04-30 20:55	2021-04-30 21:29	0	41440	2	보수담당	SL
473756	45683537	999	C94176361	98	38755489	2021-04-30 21:05	2021-04-30 21:29	0	27500	9	보수담당	SL
474255	68604130	999	C07040059	41	85238148	2021-04-30 21:47	2021-04-30 21:59	0	26470	2	보수담당	SL

2555 rows × 12 columns

```
[21] # store_id가 999인 관측값 제거  
order_df = order_df.loc[order_df['store_id'] != 999]  
order_df.loc[order_df['store_id'] == 999]
```

```
order_id store_id customer_id coupon_cd sales_detail_id order_accept_date delivered_date takeout_flag total_amount status store_name area_cd
```

- 데이터 시각화는 기본적으로 데이터에서 어떤 정보를 얻을 수 있을지를 탐색하기 위한 단계로 관련 전공이 따로 있을 정도로 매우 중요한 단계이며 시각화를 위한 방법들 또한 매우 무궁무진 합니다.
- 데이터 분석을 위한 기본적인 그래프 중 “막대 그래프/히스토그램”과 “산점도” 그래프를 소개하고 “상관성”에 대한 간략한 내용을 배워봅니다.

### 분석 데이터 구조

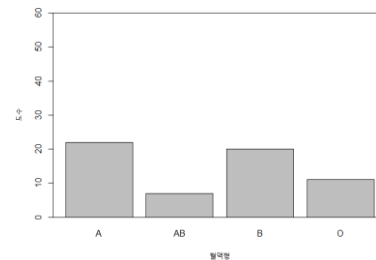
#### ■ 분석 데이터 구조

- 데이터의 구조는 한 단위(ex. 사람)의 정보들을 포함하는 row값들과 동일한 정보를 가지고 있는 column값들로 구분됨
- 데이터 내에는 기본적으로 “학번”과 같이 정보들을 이어주는 “key” 컬럼과, 실제 분석가가 예측을 하고자 하는 타겟 컬럼, 그리고 타겟을 예측하기 위한 컬럼들로 구성 되어있음
- 보통 컬럼의 단위를 분석에서는 “변수”라고 명칭 하며, 타겟을 예측하기 위한 컬럼들을 설명 변수, 혹은 feature라고 명칭함

Key	Target	등급	성별	과제	결석	과거성적	...
학번	성적						
20312	80	A	여	0	0	85	...
20421	95	A+	남	0	1	90	...
21125	72	B+	남	1	0	80	...
21217	78	A	여	1	1	75	...

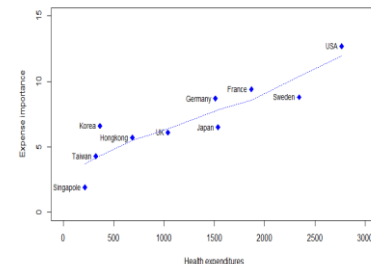
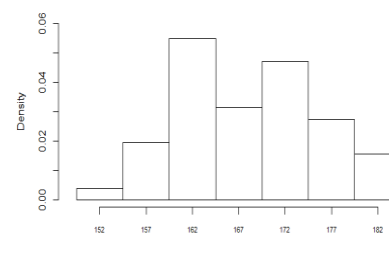
### 자료의 형태에 따른 그래프

#### ■ 범주형 변수



범주	도수	상대 도수
A	22	22/60
B	20	20/60
AB	7	7/60
O	11	11/60
합	60	1

#### ■ 수치형 변수



- 데이터 내 수치형 변수들 사이의 관계를 파악하기 위한 가장 대표적인 방법 중 하나는 “**상관성**”의 정도를 의미하는 “**상관계수**”와 함께 산점도를 확인하는 것입니다.
- 상관계수라는 통계량에서 중요한 것은 “**선형성**”을 의미하는 수치를 의미한다는 것이 유념해야 할 부분이기 때문에 그 외의 관계를 확인하기 위해 산점도를 함께 확인하는 것이 좋습니다.

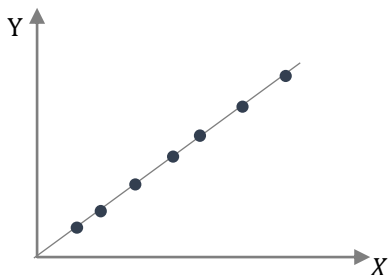
### 상관계수

#### ■ 상관성

- 두 수치형 변수 사이의 점의 선형성의 강도를 통해서 변수들끼리의 관계를 파악하고자 하는 특성

#### ■ 상관계수

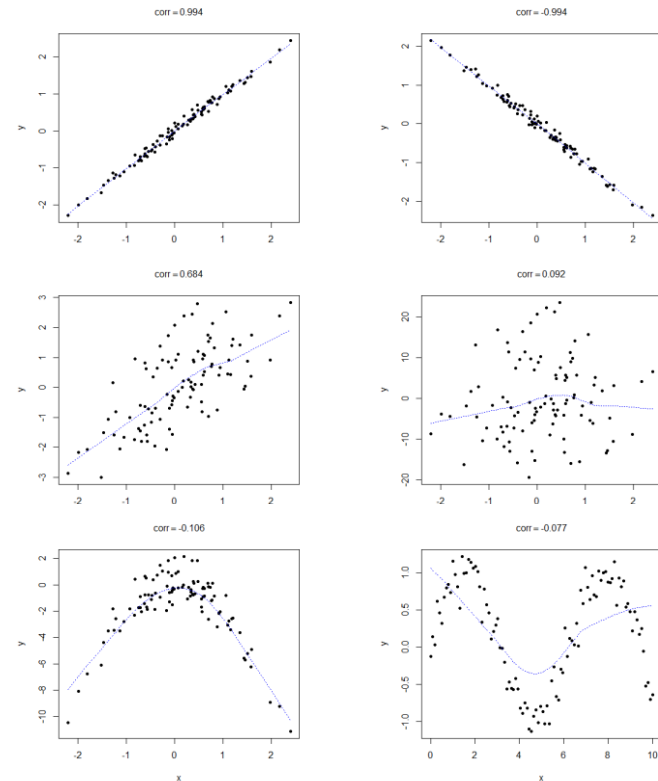
- 상관계수는 항상 **-1**과 **1**사이에 있음
- 상관계수의 **절댓값의 크기**는 **선형(직선)관계**에 가까운 정도를 나타내고 상관계수의 **부호**는 선형(직선)관계의 **방향**을 나타냄
- 상관계수의 단위는 없기 때문에 단위가 다른 여러 쌍의 변수의 **직선관계정도**를 비교할 수 있음



(2,2) (3,3) (5,5) (7,7)  
(8,8) (11,11) (13,13)

$$r = \frac{S_{xy}}{\sqrt{S_{xx}} \cdot \sqrt{S_{yy}}}$$

### 상관계수 형태



# 1.5 실습 (데이터 시각화)

## 01. 데이터 수집 및 시각화

DataPrep Report Overview Variables Interactions Correlations Missing Values

### Overview

#### Dataset Statistics

Number of Variables	13
Number of Rows	471849
Missing Cells	0
Missing Cells (%)	0.0%
Duplicate Rows	0
Duplicate Rows (%)	0.0%
Total Size in Memory	284.7 MB
Average Row Size in Memory	632.8 B
Variable Types	Numerical: 4 Categorical: 8 GeoGraphy: 1

#### Dataset Insights

<code>order_id</code> and <code>sales_detail_id</code> have similar distributions	<a href="#">Similar Distributions</a>
<code>customer_id</code> has a high cardinality: 41321 distinct values	<a href="#">High Cardinality</a>
<code>order_accept_date</code> has a high cardinality: 39588 distinct values	<a href="#">High Cardinality</a>
<code>delivered_date</code> has a high cardinality: 42939 distinct values	<a href="#">High Cardinality</a>
<code>store_name</code> has a high cardinality: 193 distinct values	<a href="#">High Cardinality</a>
<code>customer_id</code> has constant length 9	<a href="#">Constant Length</a>
<code>order_accept_date</code> has constant length 16	<a href="#">Constant Length</a>
<code>delivered_date</code> has constant length 16	<a href="#">Constant Length</a>
<code>takeout_flag</code> has constant length 1	<a href="#">Constant Length</a>
<code>status</code> has constant length 1	<a href="#">Constant Length</a>

1 2

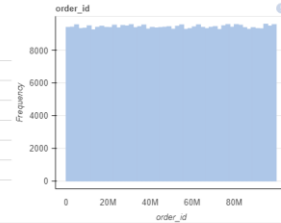
### Variables

Sort by Feature order ☐ Reverse order

#### order\_id

Show Details

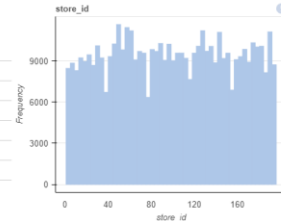
Approximate Distinct Count	470735	Mean	5.0019e+10 <sup>27</sup>
Approximate Unique (%)	99.8%	Minimum	70
Missing	0	Maximum	99999274
Missing (%)	0.0%	Zeros	0
Infinite	0	Zeros (%)	0.0%
Infinite (%)	0.0%	Negatives	0
Memory Size	7.2 MB	Negatives (%)	0.0%



#### store\_id

Show Details

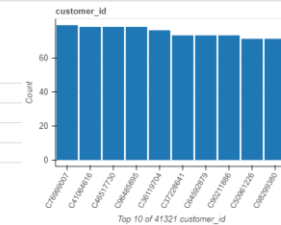
Approximate Distinct Count	196	Mean	99.0894
Approximate Unique (%)	0.0%	Minimum	1
Missing	0	Maximum	196
Missing (%)	0.0%	Zeros	0
Infinite	0	Zeros (%)	0.0%
Infinite (%)	0.0%	Negatives	0
Memory Size	7.2 MB	Negatives (%)	0.0%



#### customer\_id

Show Details

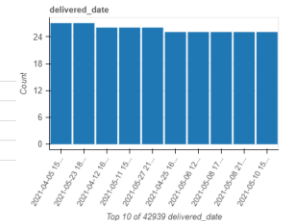
Approximate Distinct Count	41321
Approximate Unique (%)	8.8%
Missing	0
Missing (%)	0.0%
Memory Size	33.3 MB



#### delivered\_date

Show Details

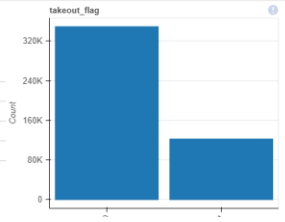
Approximate Distinct Count	42939
Approximate Unique (%)	9.1%
Missing	0
Missing (%)	0.0%
Memory Size	36.4 MB



#### takeout\_flag

Show Details

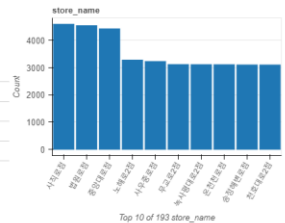
Approximate Distinct Count	2
Approximate Unique (%)	0.0%
Missing	0
Missing (%)	0.0%
Memory Size	29.7 MB



#### store\_name

Show Details

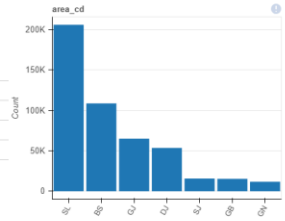
Approximate Distinct Count	193
Approximate Unique (%)	0.0%
Missing	0
Missing (%)	0.0%
Memory Size	45.0 MB



#### area\_cd

Show Details

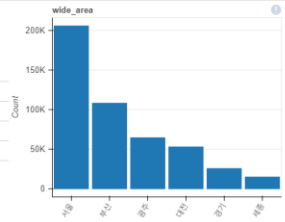
Approximate Distinct Count	7
Approximate Unique (%)	0.0%
Missing	0
Missing (%)	0.0%
Memory Size	30.1 MB



#### wide\_area

Show Details

Approximate Distinct Count	6
Approximate Unique (%)	0.0%
Missing	0
Missing (%)	0.0%
Memory Size	50.8 MB



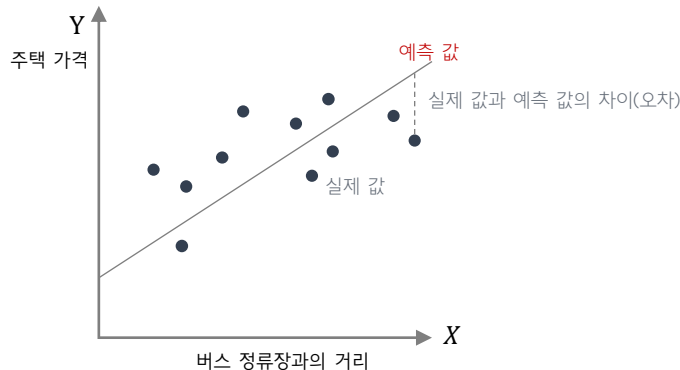
- 선형 회귀 모형(linear regression model)은 연속형 타겟 변수(Y)와 설명 변수(X)를 다음과 같은 선형적 함수 관계로 모형화 하는 방법으로 통계학 및 데이터 분석에서는 가장 기본으로 사용되는 주요 알고리즘 중 하나입니다.
- 머신 러닝 회귀 예측의 핵심은 데이터 기반으로 학습을 통한 최적의 회귀 계수를 찾아내는 것입니다.

## 선형 회귀 모형(linear regression model)

- 설명 변수(X)가 주어졌을 때 타겟 변수(Y)가 어떤 관계를 나타내는지를 다음과 같은 형태의 함수로 모델링하고 예측하는 통계적 방법

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p + \varepsilon \quad \varepsilon \sim N(0, \sigma^2)$$

- 일반적인 선형 회귀 모형(linear regression model)은 타겟 변수(Y)가 연속형일 때 사용됨  
ex. 주택 가격, 상품 가격, 상품 판매 수, ...



## 선형 회귀 모형의 종류

- 선형 회귀 모형은 설명 변수의 개수에 따라 단순 선형 회귀 모형(Simple linear regression)과 다중 선형 회귀 모형(Multiple linear regression)으로 구분 됨

### ■ 단순 선형 회귀(Simple linear regression)

- 설명 변수(X)와 타겟 변수(Y)가 하나 씩인 선형 회귀 모형을 단순 선형 회귀 모형이라고 함

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i \quad i = 1, \dots, n$$

### ■ 다중 선형 회귀(Multiple linear regression)

- 설명 변수(X)가 둘 이상인 선형 회귀 모형을 다중 선형 회귀 모형이라고 함

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_{i1} + \hat{\beta}_2 x_{i2} + \dots + \hat{\beta}_p x_{ip} \quad i = 1, \dots, n$$



- 선형 회귀 모형(linear regression model)은 해석이 직관적이라는 장점을 가지고 있으나, 데이터의 수와 이상치에 영향을 많이 받고 실제 데이터에서 만족하기 어려운 가정들을 가지고 있는 단점을 가지고 있습니다.
- 이상치는 데이터에서 발생하는 일반적인 결과와 극단적으로 다른 데이터 값을 의미합니다.

## 장점과 단점

### ■ 장점

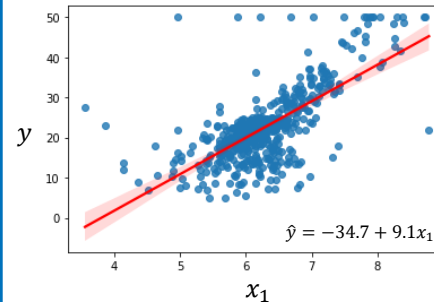
- 타겟 변수(Y)의 예측 값이 왜 이런 값을 갖게 되었는지에 대한 해석이 직관적임
- 설명 변수(X)들로부터 Y로의 영향력의 크기 비교가 가능하고 의미 있는 인자들을 선택할 수 있음

\* 인자 : 설명 변수로 사용한 각각의 데이터 특성을 의미함  
ex. 버스 정류장과의 거리, 성별, ..

### ■ 단점

- 데이터 수가 많아질수록 연산 속도가 기하급수적으로 느려짐 (데이터가 적을 때는 가장 빠름)
- 이상치의 영향을 매우 많이 받을 수 있음
- 좋은 모형을 얻기 위한 비현실적인 가정이 많음 (정규성, 독립성, 등분산성, X들 간의 독립성.. 등)
- 다양한 비현실적 가정을 맞추기 위해 수작업과 주관적 판단이 많이 필요됨

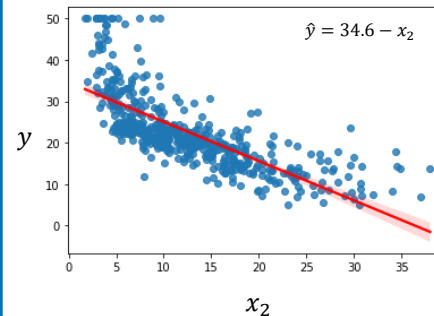
## 결과 예시 및 해석



y와  $x_1$ 은 양의 상관 관계(선형 관계)가 있어 보임

단순 회귀 모형으로부터 절편 값은 -34.7, 회귀 계수는 9.1로 추론됨

$x_1$ 이 한 단위 증가 할 때, y값은 9.1배 증가함



y와  $x_2$ 는 음의 상관 관계(선형 관계)가 있어 보임

단순 회귀 모형으로부터 절편 값은 34.6, 회귀 계수는 -1로 추론됨

$x_2$ 이 한 단위 증가 할 때, y값은 -1배 증가함

- 타겟 변수(Y)가 정규 분포인 경우인 기존의 선형 회귀 모형에서 다양한 타겟 변수(Y)의 분포들을 고려하여 설명 변수(X)들과의 선형 관계로 확장시킨 모형을 일반화 선형 모형(generalized linear model)이라고 합니다.
- 일반화 선형 모형에는 타겟 변수(Y)의 분포에 따라 다양한 모형이 존재 합니다.

### 일반화 선형 모형(generalized linear model)

- 타겟 변수(Y)와 설명 변수(X)들의 선형 관계를 고려하기 위하여 선형 예측치와 타겟 변수(Y)를 연결하기 위한 연결 함수(link function)을 사용함
- 예를 들어, 타겟 변수(Y)가 **이항 분포**인 경우 선형 예측치와 E(Y)의 범위가 일치하지 않는 문제가 생기기 때문에 연결 함수(link function)을 고려하여 모델링 함

$$g(E(Y)) = \beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip} \quad g() : \text{link function}$$

서로 간의 범위가 다르기 때문에 연결될 수가 없기 때문에,  
연결 함수 (g())을 통해 일치 시켜 줌

- 타겟 변수(Y)의 분포에 따라 다른 연결 함수를 사용하며, 분포에 따라 사용하는 모형이 다름

### 일반화 선형 회귀 모형의 종류

- 타겟 변수(Y)의 분포에 따라 선형 회귀 모형의 종류가 분류되며 자주 사용되는 분포는 다음과 같음

#### ■ 이항 분포 / 베르누이 분포

- 성공 또는 실패처럼 두 가지 결과로만 생성된 데이터  
ex. 생산된 제품이 양품 / 불량
- 타겟 변수가 이항 분포에 해당하는 경우 **“로지스틱 회귀 모형(logistic regression model)”**을 활용할 수 있음

#### ■ 포아송 분포 (포아송 회귀 분석)

- 어떤 사건이 발생하는 건수와 같은 결과로 생성된 데이터  
ex. 한 야구경기에서 실책이 발생한 수  
국내에 발생하는 진도 4이상의 지진의 횟수  
특정 영화관에서 영화를 관람한 사람의 수
- 타겟 변수가 포아송 분포에 해당하는 경우 **“포아송 회귀 모형(Poisson regression model)”**을 활용함

- 성공 또는 실패처럼 두 가지 결과를 분류하기 위한 일반화 선형 모형은 타겟 변수의 분포를 이항 분포로 가정하고 사용 합니다.
- 이 때 사용하는 연결 함수(link function)로 logit을 사용하는 모형을 **로지스틱 회귀 모형(logistic regression model)**이라고 합니다.

### 장점과 단점

#### ■ 장점

- 이항 분포의 Y에 대한 합리적인 예측 값을 구할 수 있음
- X들로부터 Y로의 영향력의 크기 비교가 가능하고 유의한 X인자들을 선택할 수 있음
- 이진 분류 뿐만 아니라 희소한 영역을 분류 하는 곳에서도 뛰어난 예측 성능을 보임

#### ■ 단점

- 데이터 수가 많아질수록 연산 속도가 기하급수적으로 느려짐 (데이터가 적을 때는 가장 빠름)
- 이상치의 영향을 매우 많이 받을 수 있음
- 좋은 모형을 얻기 위한 비현실적인 가정이 많음 (정규성, 독립성, 등분산성, X들 간의 독립성.. 등)
- 다양한 비현실적 가정을 맞추기 위해 수작업과 주관적 판단이 많이 필요됨
- 회귀 계수에 대한 해석을 비 전공자가 이해하기 어려울 수 있음  
ex. Odds ratio

### 결과 예시 및 해석

Optimization terminated successfully.

Current function value: 0.193036

Iterations 9

Logit Regression Results

Dep. Variable:	DIAG RES	No. Observations:	569		
Model:	Logit	Df Residuals:	565		
Method:	MLE	Df Model:	3		
Date:	Wed, 22 Jun 2022	Pseudo R-squ.:	0.7077		
Time:	12:55:11	Log-Likelihood:	-109.84		
converged:	True	LL-Null:	-375.72		
Covariance Type:	nonrobust	LLR p-value:	6.227e-115		
	coef	std err	z	P> z	[0.025 0.975]
$x_1$	-2.7594	1.613	-1.710	0.087	-5.922 0.403
$x_2$	1.1749	0.194	6.053	0.000	0.794 1.555
$x_3$	8.2305	1.811	4.545	0.000	4.682 11.780
$x_4$	1.5213	0.215	7.078	0.000	1.100 1.943

$x_1$ 은 한 단위 증가할 수록  $y$ 가 1일 확률이 감소하고,  
 $x_2 \sim x_4$ 은 한 단위 증가할 수록  $y$ 가 1일 확률이 증가함

각각의 회귀 계수들은 선형 회귀 계수와는 다르게 "odds ratio" 라는 값으로 해석해야 함

- 선형 회귀 모형과 로지스틱 회귀 모형을 실습하기 위한 데이터로 “보스턴 주택 데이터”와 “위스콘신 유방암 데이터”를 사용합니다.
- Python을 활용하여 각 데이터에 대한 예측 모형을 생성할 수 있습니다.

### 보스턴 주택 데이터

#### ■ 데이터 설명

- 보스턴 시의 주택 가격에 대한 데이터로 주택의 여러가지 요건들과 주택의 가격 정보가 포함되어 있음
- 보스턴 인근의 주택 가격의 중앙값을 기준으로 하고 있으며 여러 개의 측정 지표들을 포함하고 있음
- 총 14개의 column과 506개의 row를 가지고 있음
- **CRIM**: 지역별 범죄 발생률
- **ZN**: 25,000평방피트를 초과하는 거주 지역의 비율
- **INDUS**: 비상업 지역 넓이 비율
- **CHAS**: 찰스강의 경계에 위치 여부
- **NOX**: 일산화질소 농도
- **RM**: 거주할 수 있는 방 개수
- **AGE**: 1940년 이전에 건축된 소유 주택의 비율
- **DIS**: 5개 주요 고용센터까지의 가장 거리
- **RAD**: 고속도로 접근 용이도
- **TAX**: 10,000달러당 재산세율
- **PTRATIO**: 지역의 교사와 학생 수 비율
- **B**: 지역의 흑인 거주 비율
- **LSTAT**: 하위 계층의 비율
- **PRICE**: 본인 소유의 주택 가격 (중앙값)

#### ■ 데이터 형태

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	PRICE
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33	36.2

### 위스콘신 유방암 데이터

#### ■ 데이터 설명

- 위스콘신 유방암 데이터는 종양의 크기, 모양 등의 다양한 속성 값을 기반으로 해당 종양이 악성인지 양성인지를 분류한 데이터임
- 총 31개의 column과 569개의 row를 가지고 있으나 대표적인 4개의 설명변수만 사용

- **mean radius**: 중심에서 외벽까지 거리들의 평균값
- **mean texture**: 평균 질감
- **mean area**: 평균 면적
- **mean symmetry**: 대칭 정도
- **DIAG RES**: 양성 여부 (1. 양성 : 0. 악성)

	mean radius	mean texture	mean area	mean symmetry	DIAG RES
0	17.99	10.38	1001.0	0.2419	0
1	20.57	17.77	1326.0	0.1812	0
2	19.69	21.25	1203.0	0.2069	0
3	11.42	20.38	386.1	0.2597	0
4	20.29	14.34	1297.0	0.1809	0
5	12.45	15.70	477.1	0.2087	0
6	18.25	19.98	1040.0	0.1794	0
7	13.71	20.83	577.9	0.2196	0
8	13.00	21.82	519.8	0.2350	0
9	12.46	24.04	475.9	0.2030	0

## 2.3 실습 (선형 회귀 모형)

### 02. 분석 알고리즘 I

```
[1] import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from scipy import stats
from sklearn.datasets import load_boston
matplotlib inline
```

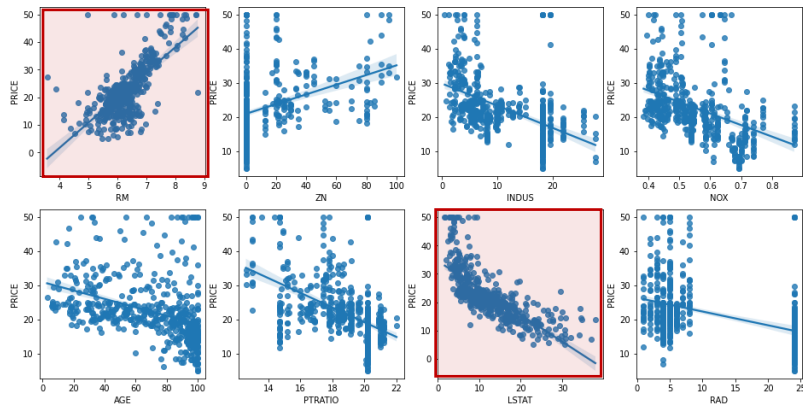
```
[2] # boston 데이터 로드
boston = load_boston()

# boston 데이터 DataFrame으로 변환
bostonDF = pd.DataFrame(boston.data, columns = boston.feature_names)
```

```
[3] # boston 데이터의 타겟 변수와 정의
bostonDF['PRICE'] = boston.target
bostonDF.shape
bostonDF.head()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	PRICE
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33	36.2

```
[4] # 타겟 변수와 설명 변수간의 상관 관계 확인을 위한 시각화
# 2개의 행과 4개의 열을 가진 multi-plot
fig, axs = plt.subplots(figsize = (16,8), ncols = 4, nrows = 2)
lm_features = ['RM', 'ZN', 'INDUS', 'NOX', 'AGE', 'PTRATIO', 'LSTAT', 'RAD']
for i, feature in enumerate(lm_features):
    row = int(i/4)
    col = i%4
    sns.regplot(x=feature, y='PRICE', data = bostonDF, ax=axs[row][col])
```



```
[5] from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
# 타겟 변수와 설명 변수 데이터를 구분
y_target = bostonDF['PRICE']
X_data = bostonDF.drop(['PRICE'], axis = 1, inplace = False)

# 선형 회귀 모형 생성
LM = LinearRegression()
LM.fit(X_data, y_target)

# 예측값 생성
y_preds = LM.predict(X_data)

# 예측값 비교
pd.DataFrame({'y_target': y_target, 'y_preds': y_preds})
```

	y_target	y_preds
0	24.0	30.003843
1	21.6	25.025562
2	34.7	30.567597
3	33.4	28.607036
4	36.2	27.943524
...	...	...
501	22.4	23.533341
502	20.6	22.375719
503	23.9	27.627426
504	22.0	26.127967
505	11.9	22.344212

```
[6] print('intercept:', np.round(LM.intercept_, 1))
print('coefficients:', np.round(LM.coef_, 1))
```

```
intercept: 36.5
coefficients: [-0.1  0.  0.  2.7 -17.8  3.8  0. -1.5  0.3 -0. -1.  0. -0.5]
```

```
coeff = pd.Series(data=np.round(LM.coef_, 1), index = X_data.columns)
coeff
```

```
CRIM    -0.1
ZN       0.0
INDUS    0.0
CHAS     2.7
NOX    -17.8
RM       3.8
AGE       0.0
DIS     -1.5
RAD       0.3
TAX      -0.0
PTRATIO -1.0
B         0.0
LSTAT   -0.5
dtype: float64
```

NOX의 회귀 계수 값이  
다른 값에 비해 - 방향으로 너무 큼

$$\widehat{PRICE} = 36.5 - 0.1CRIM + 2.7CHAS - 17.8NOX + 3.8RM - 1.5DIS + 0.3RAD - 1PTRATIO - 0.5LSTAT$$

다른 모든 값이 동일한 경우,  
RM(거주할 수 있는 방 개수)가 1개 증가할 수록 주택 가격이 3.8만큼 증가함

## 2.3 실습 (로지스틱 회귀 모형)

### 02. 분석 알고리즘 I

```
[1] import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.datasets import load_breast_cancer
%matplotlib inline

[2] # 유방암 데이터 로드
cancer = load_breast_cancer()

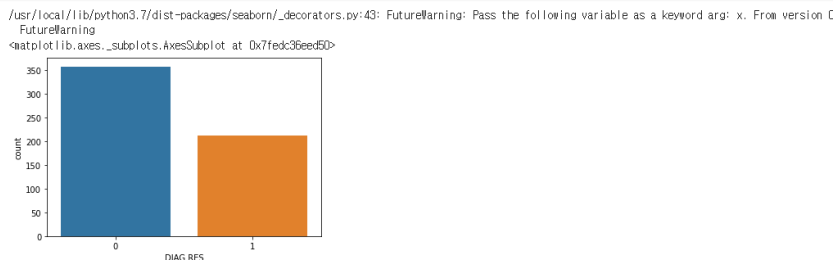
# 유방암 데이터 DataFrame으로 변환
cancerDF = pd.DataFrame(cancer.data, columns = cancer.feature_names)

[3] # 일부 변수만 사용
cancerDF = cancerDF[['mean radius', 'mean texture', 'mean area', 'mean symmetry']]

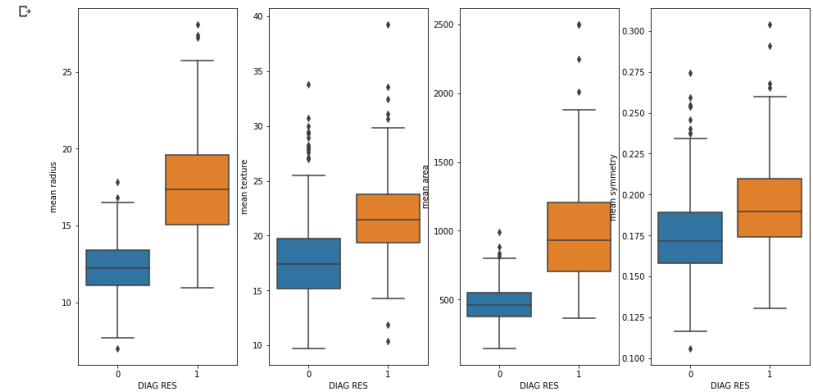
# 유방암 데이터의 타겟 변수와 정의 (1이면 양성 종양, 0이면 악성 종양)
cancerDF['DIAG RES'] = cancer.target
# 보통 1을 주요 타겟이 되는 범주로 정의함
cancerDF['DIAG RES'] = np.where(cancerDF['DIAG RES'] == 0, 1, 0)
cancerDF.shape
cancerDF.head(10)
```

	mean radius	mean texture	mean area	mean symmetry	DIAG RES
0	17.99	10.38	1001.0	0.2419	1
1	20.57	17.77	1326.0	0.1812	1
2	19.69	21.25	1203.0	0.2069	1
3	11.42	20.38	386.1	0.2597	1
4	20.29	14.34	1297.0	0.1809	1
5	12.45	15.70	477.1	0.2087	1
6	18.25	19.98	1040.0	0.1794	1
7	13.71	20.83	577.9	0.2196	1
8	13.00	21.82	519.8	0.2350	1
9	12.46	24.04	475.9	0.2030	1

```
[4] # 진단 결과에 대한 분포 확인
sns.countplot(cancerDF['DIAG RES'])
```



```
# 양성 / 음성별 설명 변수의 분포 확인
fig, axes = plt.subplots(figsize = (16,8), ncols = 4, nrows = 1)
for i, feature in enumerate(['mean radius', 'mean texture', 'mean area', 'mean symmetry']):
    sns.boxplot(x='DIAG RES', y = feature, data = cancerDF, ax = axes[i*4])
```



```
[6] from sklearn.linear_model import LogisticRegression

# 타겟 변수와 설명 변수 데이터를 구분
y_target = cancerDF['DIAG RES']
X_data = cancerDF.drop(['DIAG RES'], axis = 1, inplace = False)

# 표준화 작업
scaler = StandardScaler()
scaler.fit(X_data)
X_data = pd.DataFrame(scaler.transform(X_data), columns = X_data.columns)

# 로지스틱 회귀 모형 생성
Logistic = LogisticRegression()
Logistic.fit(X_data, y_target)

LogisticRegression()
```

```
[7] # 회귀 계수 확인
column_name = ['const'] + X_data.columns.tolist()
```

```
[8] # 베타 값 추출
beta = np.concatenate([Logistic.intercept_, Logistic.coef_.reshape(-1)]).round(2)
# exp(베타) 값 추출
odds = np.exp(beta).round(2)
# 베타 값 비교
beta_analysis = pd.DataFrame(np.c_[beta, odds], index = column_name, columns = ['beta', 'exp(beta)'])
beta_analysis
```

	beta	exp(beta)
const	-0.57	0.57
mean radius	2.05	7.77
mean texture	1.08	2.94
mean area	2.08	8.00
mean symmetry	1.37	3.94

## 2.3 실습 (로지스틱 회귀 모형)

### 02. 분석 알고리즘 I

```
[9] # 예측 확률
probs = Logistic.predict_proba(X_data)[: ,1]

# 타겟 변수 예측
y_preds = np.where(probs.reshape(-1) >= 0.5, 1, 0)

# 예측값 비교
타겟 변수가 1일 확률이 0.5보다 크거나 같을 경우 1로 분류하고, 작은 경우 0으로 분류함
pd.DataFrame({'y_target' : y_target, 'y_preds' : y_preds})
```

y_target	y_preds
0	1
1	1
2	1
3	1
4	1
...	...
564	1
565	1
566	1
567	1
568	0

569 rows x 2 columns

```
[10] from sklearn.metrics import confusion_matrix
pd.DataFrame(confusion_matrix(y_target, y_preds))
```

예측 값	0	1	
실제 값	0	343	14
1	30	182	

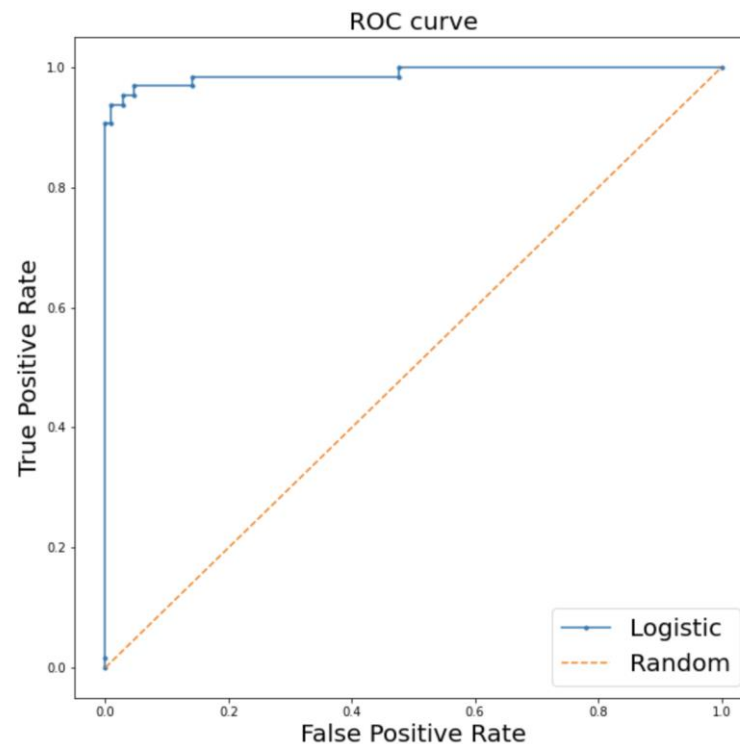
```
[11] # 타겟 변수 예측
y_preds = np.where(probs.reshape(-1) >= 0.8, 1, 0)

# 타겟 변수가 1일 확률이 0.8보다 크거나 같을 경우 1로 분류하고, 작은 경우 0으로 분류함
# 예측값 비교
pd.DataFrame({'y_target' : y_target, 'y_preds' : y_preds})

# confusion matrix
pd.DataFrame(confusion_matrix(y_target, y_preds))
```

예측 값	0	1
0	352	5
1	54	158

어떤 결과가 좋은 결과 인가?



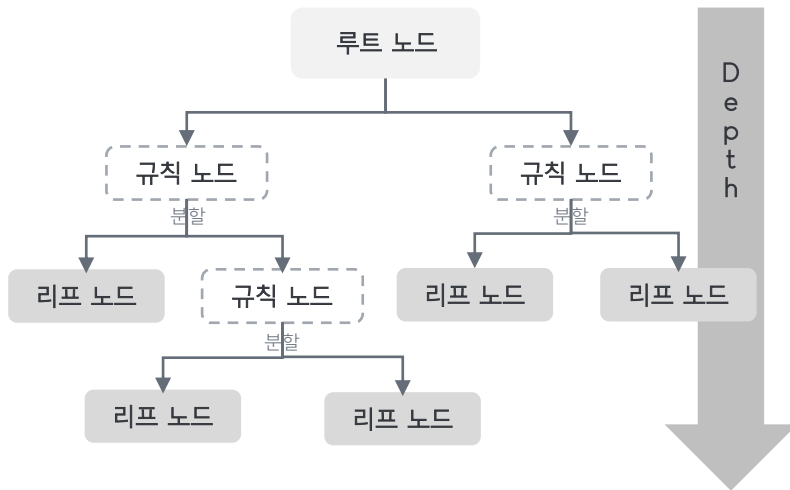


- 의사 결정 나무(Decision Tree)는 머신 러닝 알고리즘 중 직관적으로 이해하기 쉬운 알고리즘이며, 예측을 위한 알고리즘 뿐만 아니라 정상적인 값들에 비하여 분리되어 있는 이상 값들을 탐지하는 등을 위한 방법으로도 많이 사용되는 알고리즘 입니다.
- 의사 결정 나무는 예측력이 좋은 것으로 알려져 있는 랜덤 포레스트(Random Forest)등에 응용되어 사용되는 기본 알고리즘 중 하나 입니다.

## 의사 결정 나무(Decision Tree)

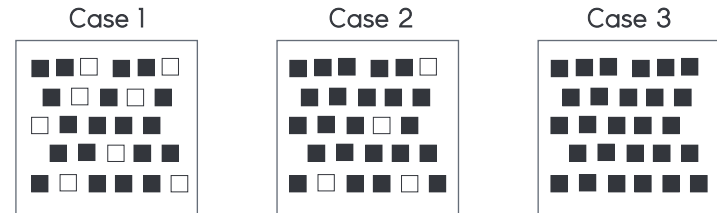
- 의사 결정 나무는 데이터에 있는 규칙을 학습으로 통해 찾아내어 나무(Tree) 기반의 분류 규칙을 만들어내는 알고리즘임
- 의사 결정 나무는 기본적으로 규칙 노드(Decision Node)와 리프 노드(Leaf Node)로 나누어서 생성 됨

\* 규칙 노드(Decision Node) : 규칙 조건을 표기하는 노드  
리프 노드(Leaf Node) : 규칙 조건에 의해 결정된 결과 값 노드



## 분할 기준 종류

- 의사 결정 나무는 정보의 불순도가 낮은 데이터로 분할 시킴



어떤 Case의 데이터가 불순도가 낮다고 할 수 있겠는가 ?

- 불순도를 측정하는 대표적인 방법으로는 엔트로피를 이용한 정보 이득(Information Gain)지수, 지니 계수(Gini index)가 있음
- C4.5는 엔트로피를 이용한 의사결정나무 알고리즘이며, CART는 지니 계수를 이용하는 의사결정나무 알고리즘이다.

- 의사 결정 나무에서 가치를 분할하기 위한 기준으로는 주로 “엔트로피 지수”를 이용한 정보 이득 지수와 “지니 계수”가 대표적입니다.
- 의사 결정 나무는 쉽게 생성이 가능하며 해석이 쉽다는 장점이 있으나 과적합의 문제가 발생할 수 있다는 단점을 가지고 있습니다.

## 분할 기준 종류

### ■ 엔트로피 (Entropy)

- 주어진 데이터 집합의 혼잡도를 의미하는 지수로 낮을 수록 불순도가 낮음

$$Entropy(A) = -\sum_{k=1}^n p_k \log_2(p_k) \quad p_k = \frac{\text{주어진 데이터 중 } C_k \text{에 속하는 수}}{\text{주어진 데이터 수}}$$

### ■ 정보 이득 (Information Gain) 지수

- 이전 엔트로피 지수에서 현재 엔트로피 지수를 뺀 값

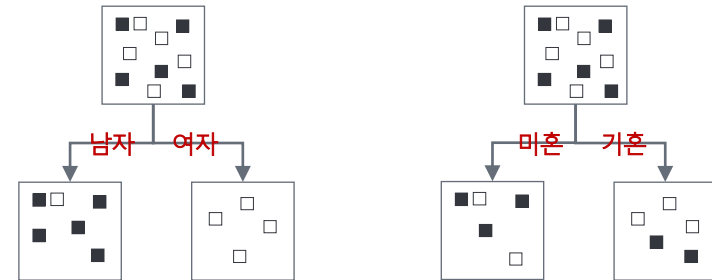
### ■ 지니 계수 (Gini index)

- 주어진 데이터 집합의 혼잡도를 의미하는 지수로 0에 가까울 수록 불순도가 낮고 1에 가까울수록 불순도가 높음

$$G(A) = 1 - \sum_{k=1}^n p_k^2 \quad p_k = \frac{\text{주어진 데이터 중 } C_k \text{에 속하는 수}}{\text{주어진 데이터 수}}$$

## 분할 기준 예시 및 장단점

### ■ 분할 기준 지수 예시



- A게임을 탈퇴할 것으로 예상되는 고객을 예측하여 고객 관리를 할 계획일때, 성별과 결혼 유무 중 어떤 조건을 사용해야 하는가

### ■ 장점

- 구성하기 쉽고, 직관적이기 때문에 해석이 쉬움
- 이상치에 대한 영향이 크지 않음

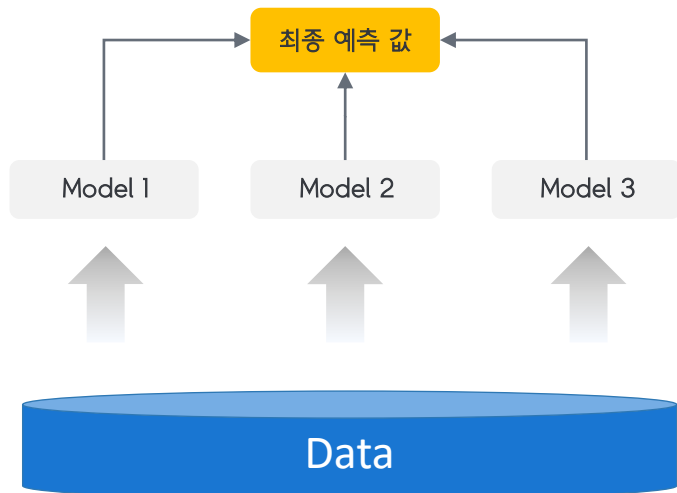
### ■ 단점

- 과적합의 문제가 발생할 수 있으므로 depth를 잘 조절 해야함

- 랜덤 포레스트(Random Forest)는 1개의 의사 결정 나무(Decision tree)이 아닌 보다 더 좋은 예측 알고리즘을 생성하기 위해 여러 개의 의사 결정 나무를 사용하는 앙상블(Ensemble) 알고리즘입니다.
- 랜덤 포레스트는 예측 문제에서 매우 높은 활용도를 자랑하는 대표적인 알고리즘입니다.

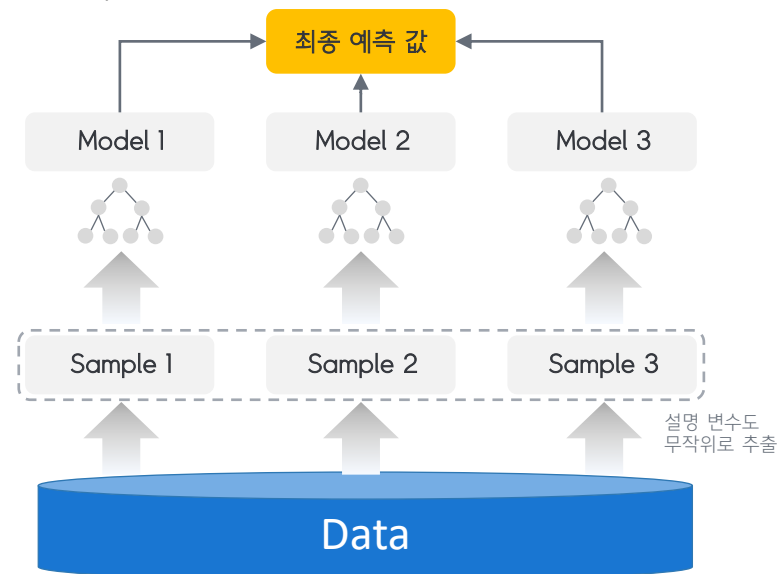
### 앙상블(Ensemble)

- 앙상블 학습(Ensemble Learning)은 여러 개의 알고리즘을 생성하여 그 예측결과들을 결합함으로써 더 정확한 최종 결과를 도출하는 방법을 의미함
- 앙상블 학습방법은 사용 알고리즘의 종류나, 데이터 샘플링 여부에 따라 구분 될 수 있음



### 랜덤 포레스트 (Random Forest)

- 랜덤 포레스트(Random Forest)는 의사결정나무(Decision Tree)기반의 알고리즘으로 데이터를 샘플링하여 여러 개의 의사결정나무를 학습한 뒤, 최종적으로 예측 값을 결정하는 알고리즘임



- 랜덤 포레스트(Random Forest)는 의사 결정 나무(Decision Tree)에서의 단점인 과적합(overfitting)문제를 보완하는 방법으로 높은 예측력을 보인다는 장점을 가지고 있습니다.
- 그러나 의사 결정 나무에 비해 여러 개의 나무를 사용하기 때문에 해석의 어려움이 있습니다.

## 장점과 단점

### ■ 장점

- 과적합(Overfitting)의 위험이 있는 의사결정 나무에 비해 과적합 문제에 강함
- 기본적으로 분류 및 예측 문제에서 비교적 높은 예측 정확도를 보임
- 이상치에 대한 영향이 크지 않음

### ■ 단점

- 개별 적인 의사 결정 나무에 대한 분석이 어렵고 복잡한 구조이기 때문에 해석의 어려움이 있음
- 고차원의 데이터나 Sparse한 데이터에는 적절한 학습이 어려움
- 선형 모델에 비해 많은 메모리를 사용하기 때문에 알고리즘을 생성하는데 속도가 느려질 가능성이 있음
- 메모리 사용에 제약에 발생할 수 있음

## 실습 데이터

### ■ 데이터 설명

- 위스콘신 유방암 데이터는 종양의 크기, 모양 등의 다양한 속성 값을 기반으로 해당 종양이 악성인지 양성인지를 분류한 데이터임
- 총 32개의 column과 569개의 row를 가지고 있음

- **radius** : 중심에서 외벽까지 거리
- **texture** : 질감
- **area** : 면적
- **perimeter** : 둘레
- **smoothness** : 매끄러운 정도
- **compactness** : 조그만 정도
- **concavity** : 오목함
- **points** : 오목한 점의 수
- **symmetry** : 대칭 정도
- **dimension** : 프랙탈 차원

평균값,  
표준 오차값,  
제일 큰 3개의 값을 평균낸 값

- **diagnosis** : 양성 여부 (1. 양성, 0. 악성)

	mean radius	mean texture	mean area	mean symmetry	DIAG RES
0	17.99	10.38	1001.0	0.2419	0
1	20.57	17.77	1326.0	0.1812	0
2	19.69	21.25	1203.0	0.2069	0
3	11.42	20.38	386.1	0.2597	0
4	20.29	14.34	1297.0	0.1809	0

## 3.3 실습 (Decision Tree)

### 03. 분석 알고리즘 II

```
[1] import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from sklearn.datasets import load_breast_cancer
from os import system # Tree 시각화를 위한
system("pip install graphviz")
import graphviz
```

```
[2] # 유방암 데이터 로드
cancer = load_breast_cancer()

# 유방암 데이터 DataFrame으로 변환
cancerDF = pd.DataFrame(cancer.data, columns = cancer.feature_names)
```

```
[3] # 유방암 데이터의 타겟 변수와 정의 (1이면 양성 종양, 0이면 악성 종양)
cancerDF['diagnosis'] = cancer.target
# 보통 1을 주요 타겟이 되는 범주로 정의함
cancerDF['diagnosis'] = np.where(cancerDF['diagnosis'] == 0, 1, 0)
cancerDF.shape
cancerDF.head(10)
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness
0	17.99	10.38	122.80	1001.0	0.11840	0.27760
1	20.57	17.77	132.90	1326.0	0.08474	0.07864

```
# 진단 결과에 대한 분포 확인
sns.countplot(cancerDF['diagnosis'])
```

```
# 양성 / 음성별 설명 변수의 분포 확인
plt.figure(figsize=[20,15])
```

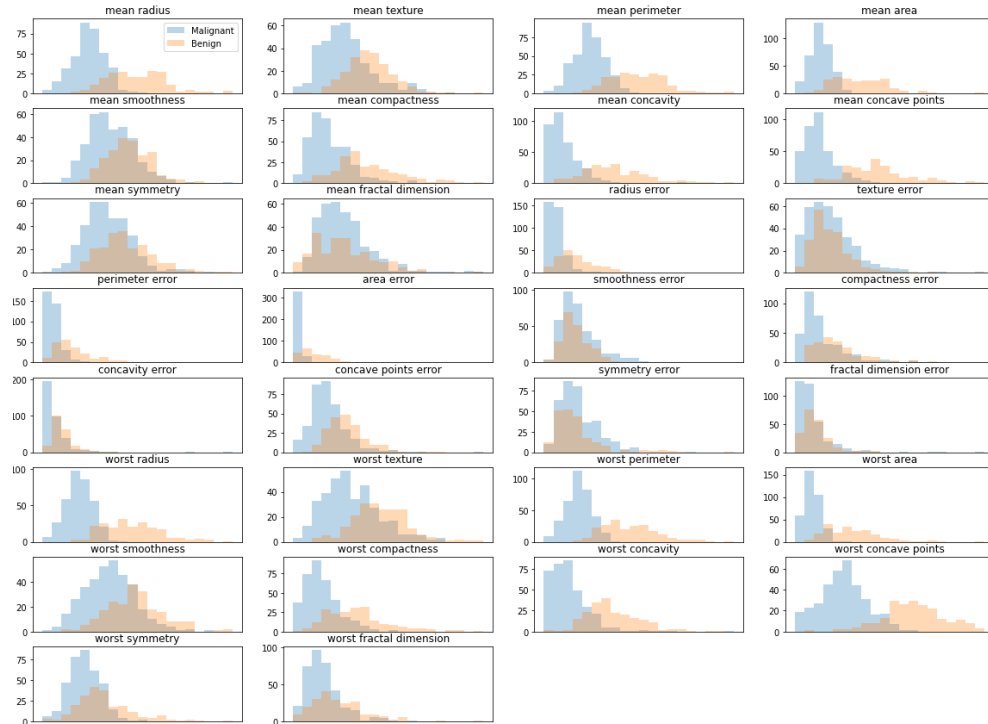
```
malignant = cancerDF[cancerDF.diagnosis==0]
benign = cancerDF[cancerDF.diagnosis==1]
```

```
# 설명 변수 데이터 set
X_data = cancerDF.drop("diagnosis", axis = 1, inplace = False)
```

```
# 히스토그램을 활용한 모든 설명 변수에 대한 분포 확인
for col in range(30):
    plt.subplot(8,4,col+1)
    _, bins=np.histogram(X_data.iloc[:,col], bins=20)

    plt.hist(malignant.iloc[:,col],bins=bins, alpha=0.3)
    plt.hist(benign.iloc[:,col], bins=bins ,alpha=0.3)
    plt.title(X_data.columns[col])
    if col==0: plt.legend(['Malignant', 'Benign'])
    plt.xticks([])
```

### 시각화 결과



### 의사결정 모형 생성

```
[6] from sklearn import tree

# 타겟 변수
y_target = cancerDF['diagnosis']

# 의사결정나무(Decision Tree) 모형 생성
DTree = tree.DecisionTreeClassifier(random_state=300)
DTree.fit(X_data, y_target)
```

## 3.3 실습 (Decision Tree)

### 03. 분석 알고리즘 II

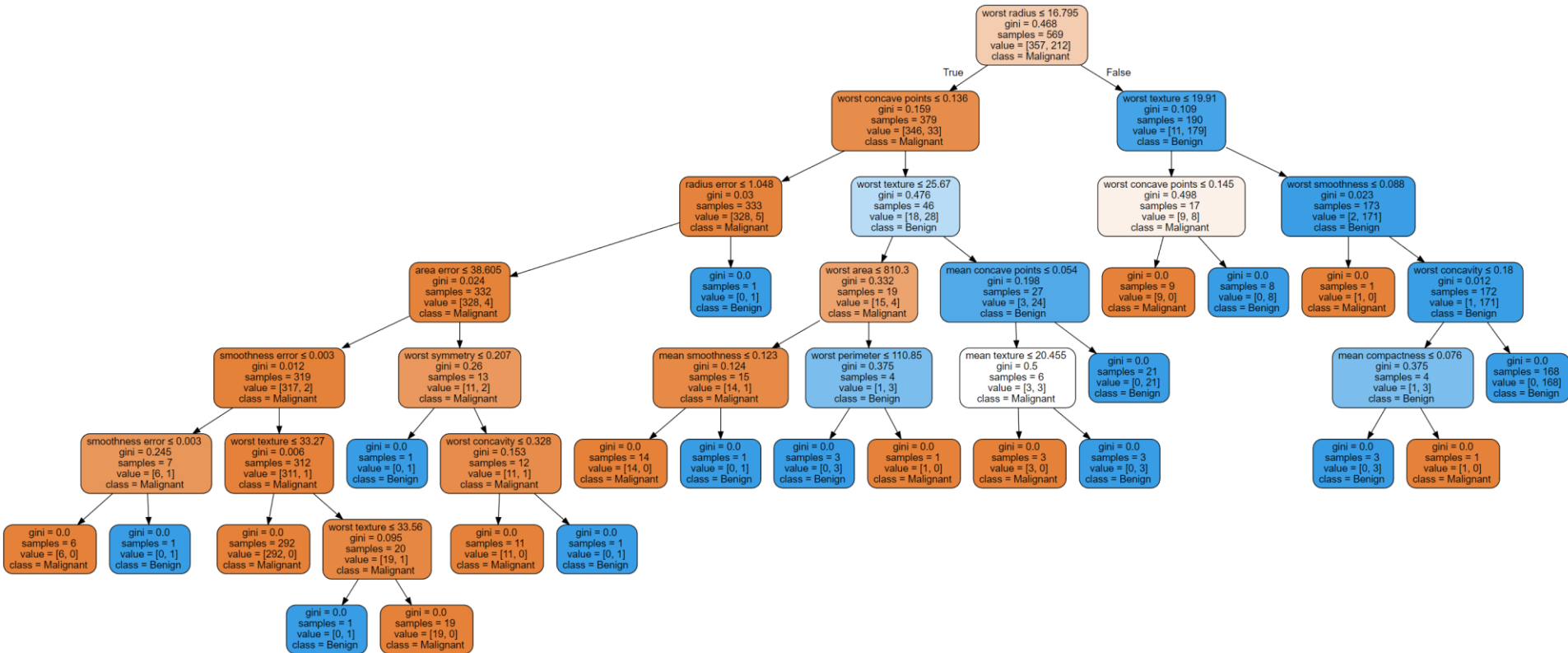


# Decision Tree 그래프로 확인

```
dot_data = tree.export_graphviz(DTree,  
    out_file = None,  
    feature_names = X_data.columns,  
    class_names = ['Malignant', 'Benign'],  
    filled = True,  
    rounded = True,  
    special_characters = True)  
# file로 변환하지 않음  
# 변수명  
# 타겟 종류  
# 색상 채움  
# 반올림 함  
# 특수 문자 사용함
```

```
graph = graphviz.Source(dot_data)
```

```
graph
```



## 3.3 실습 (Decision Tree)

### 03. 분석 알고리즘 II

# 의사결정나무(Decision Tree) 모형 생성 (depth 3으로 조절)

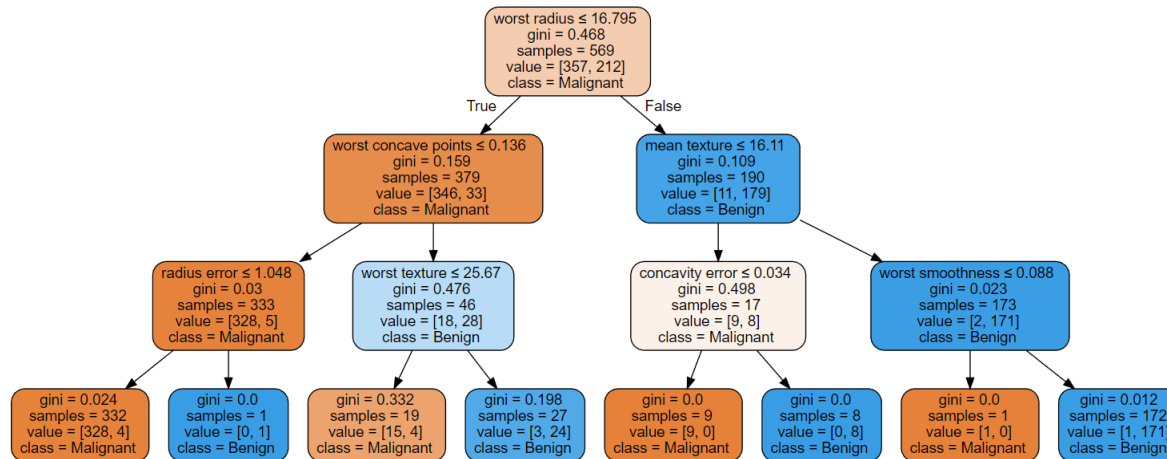
```
DTree_depth3 = tree.DecisionTreeClassifier(random_state=300, max_depth = 3)
DTree_depth3.fit(X_data, y_target)
```

# Decision Tree 그래프로 확인 (depth 3으로 조절)

```
dot_data3 = tree.export_graphviz(DTree_depth3,
    out_file = None,
    feature_names = X_data.columns,
    class_names = ['Malignant', 'Benign'],
    filled = True,
    rounded = True,
    special_characters = True)

# file로 변환하지 않음
# 변수명
# 타겟 종류
# 색상 채움
# 반올림 함
# 특수 문자 사용함
```

```
graph3 = graphviz.Source(dot_data3)
graph3
```



# 타겟 변수 예측 (depth를 주지 않은 것과, 준 것 비교)

```
y_preds = DTree.predict(X_data)
y_preds3 = DTree_depth3.predict(X_data)
```

# 예측값 비교

```
pd.DataFrame({'y_target': y_target, 'y_preds': y_preds, 'y_preds3': y_preds3})
```

```
[10] from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
```

```
from sklearn.metrics import confusion_matrix
print("Depth 제한 없는 의사결정 나무 결과")
pd.DataFrame(confusion_matrix(y_target, y_preds))
print("\n")
print("Depth를 3으로 제한한 의사결정 나무 결과")
pd.DataFrame(confusion_matrix(y_target, y_preds3))
```

### Depth 조절 별 결과 비교

Depth 제한 없는 의사결정 나무 결과

예측 값	0	1
실제 값 0	0	357
실제 값 1	1	0

Depth를 3으로 제한한 의사결정 나무 결과

예측 값	0	1
실제 값 0	0	353
실제 값 1	1	8



## 랜덤 포레스트(Random Forest) 모형 생성

```
[12] from sklearn.ensemble import RandomForestClassifier

# 100개의 decision tree를 사용한 랜덤 포레스트 (max_features는 sqrt개수를 사용하는 것이 디폴트임)
RF = RandomForestClassifier(n_estimators=100, random_state=0)
RF.fit(X_data, y_target)

# 예측값 생성
y_preds_RF = RF.predict(X_data)

# 예측값 비교
pd.DataFrame({'y_target' : y_target, 'y_preds' : y_preds, 'y_preds3' : y_preds3, 'y_preds_RF' : y_preds_RF})
```

RandomForestClassifier(random\_state=0)

	y_target	y_preds	y_preds3	y_preds_RF
0	1	1	1	1
1	1	1	1	1
2	1	1	1	1
3	1	1	1	1
4	1	1	1	1
...	...	...	...	...
564	1	1	1	1
565	1	1	1	1
566	1	1	1	1
567	1	1	1	1
568	0	0	0	0

569 rows x 4 columns

```
pd.DataFrame(confusion_matrix(y_target, y_preds_RF))
```

	0	1
0	357	0
1	0	212

[14] # 의사 결정 나무와 랜덤 포레스트의 결과 비교

```
from sklearn.metrics import confusion_matrix
print("Depth 제한 없는 의사결정 나무 결과")
pd.DataFrame(confusion_matrix(y_target, y_preds))
print("\n")
print("Depth를 3으로 제한한 의사결정 나무 결과")
pd.DataFrame(confusion_matrix(y_target, y_preds3))
print("\n")
print("랜덤 포레스트 결과(100 개의 tree 사용)")
pd.DataFrame(confusion_matrix(y_target, y_preds3))
```

Depth 제한 없는 의사결정 나무 결과

	0	1
0	357	0
1	0	212

Depth를 3으로 제한한 의사결정 나무 결과

	0	1
0	353	4
1	8	204

랜덤 포레스트 결과(100 개의 tree 사용)

	0	1
0	353	4
1	8	204

- 여러가지의 예측 알고리즘을 생성한 후, 최종 모형을 선택하기 위해서는 생성한 모형이 얼마나 좋은 성능을 보이는지 수치적으로 비교할 수 있는 평가 지표가 필요합니다.
- 모형 평가를 위한 지표는 수치 예측 문제와 분류 예측 문제에 따라 다른 지표를 사용합니다.

## 수치 예측 문제인 경우

### ■ MSE (Mean Squared Error)

- 실제 값과 예측 값의 차이를 제곱한 값의 평균을 구한 지표
- 대표적으로 사용되는 지표로 계산 및 이해하기가 쉽다는 장점을 가지고 있으나 타겟의 스케일에 의존적임
- 에러를 제곱하면서 발생하는 왜곡을 보완하기 위해 루트를 사용한 RMSE(Root Mean Squared Error)도 많이 사용됨

$$MSE = \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{n}, \quad RMSE = \sqrt{\frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{n}}$$

### ■ MAPE (Mean Absolute Percentage Error)

- 실제 값 대비 실제 값과 예측 값의 차이에 대한 절대값 비율의 평균을 퍼센트로 표현한 지표
- 타겟의 스케일에 대한 문제를 보완하고 직관적이지만 타겟의 값이 매우 작은 경우 무한대에 가까운 값이 생성될 수 있음

$$MAPE = \frac{100}{n} \sum_{i=1}^n \left| \frac{y_i - \hat{y}_i}{y_i} \right|, \quad MPE = \frac{100}{n} \sum_{i=1}^n \frac{y_i - \hat{y}_i}{y_i}$$

## 분류 예측 문제인 경우

### ■ Accuracy / Precision / Recall

- Accuracy(정확도)는 실제 값에서 예측 값이 얼마나 같은지를 판단하는 지표로 전체 예측된 데이터의 건수에서 실제 값과 동일한 데이터 건수의 비율을 나타낸 지표임
- Precision(정밀도)와 Recall(재현율)은 예측 값 중에서도 대상이 되는 예측 값("1")에 초점을 둔 평가 지표임

		예측 값	
		0	1
실제 값	0	445	15
	1	35	5

Confusion matrix

정확도 : 450/500 = 0.90

정밀도 : 5/20 = 0.25

재현율 : 5/40 = 0.125

### ■ F1 Score

- Precision과 Recall을 결합한 지표

$$F1 = \frac{1}{\frac{1}{recall} + \frac{1}{precision}} = 2 \times \frac{precision \times recall}{precision + recall}$$

- 정밀도와 재현율은 서로 상호 보완적인 지표이기 때문에 양쪽 모두를 올릴 수는 없는데 이러한 관계를 Precision-Recall Trade-Off라고 합니다.
- AUC 스코어는 분류 문제에서 모형의 성능을 평가하는 지표들 중 하나로 ROC 곡선에 기반한 면적의 값을 의미합니다.

## Trade-Off

### Precision-Recall Trade-Off

- 분류 문제의 특성상 예측 결과에 대한 확률 값을 도출한 경우 분류를 하기 위해 특정 확률을 기준으로 1과 0을 나눠주는 임계값(Threshold)이 필요함
- Precision과 Recall은 이 임계값을 통해 조정될 수 있으나 두 지표는 서로 상호 보완적인 지표이기 때문에 한 쪽을 올리는 경우 다른 한 쪽이 줄어드는 현상이 발생되는데 이를 Precision-Recall Trade-Off라고 함

Threshold = 0.5

	예측 값 0	1	
실제 값	0	343	14
	1	30	182

정확도 :  $525/569 = 0.92$

정밀도 :  $182/196 = 0.93$

재현율 :  $182/212 = 0.86$

Threshold = 0.8

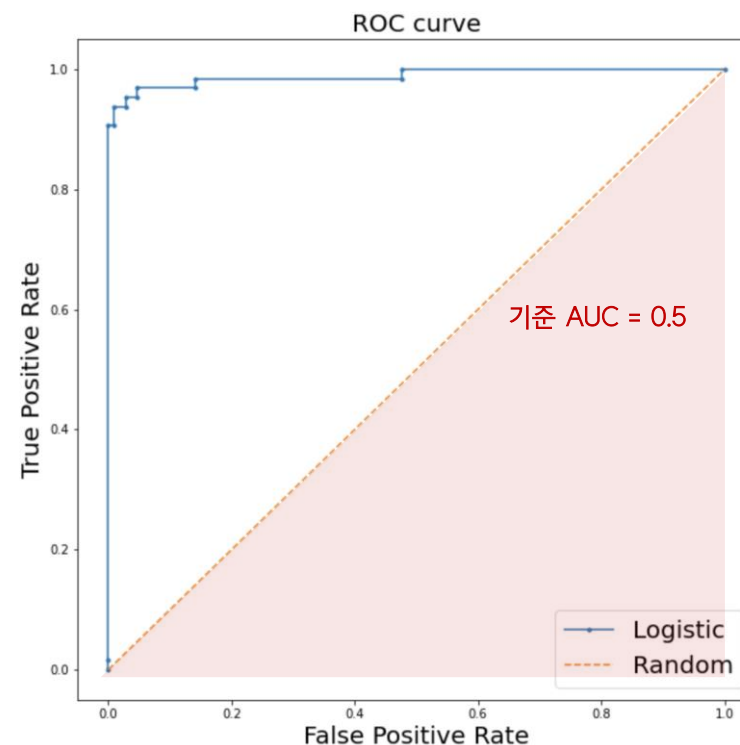
	예측 값 0	1	
실제 값	0	352	5
	1	54	158

정확도 :  $510/569 = 0.90$

정밀도 :  $158/163 = 0.97$

재현율 :  $158/212 = 0.75$

## AUC Score



- 생성한 분석 모형이 분석 데이터에 과도하게 맞춰져서 생성되어 모형이 굉장히 복잡하게 되었을 때 주로 나타나는 현상으로 분석 데이터 외의 새로운 데이터에서 예측 성능이 크게 저하되는 경우를 Overfitting 문제라고 합니다.
- Overfitting 문제를 방지하기 위해 모형을 생성하는 경우 모든 데이터를 사용하는 것이 아닌 모형 생성을 위한 훈련 데이터와 이를 검증하기 위한 테스트 데이터로 나누어서 사용되어야 합니다.

### Overfitting

#### ■ Overfitting (과대 적합)

- 분석 데이터에만 존재하는 특징에 과대하게 학습되어 작은 변화에도 과장된 결과를 초래하여 새로운 데이터에 대한 예측력이 떨어지는 문제를 의미함

#### ■ 발생 원인

- 모형을 생성할 때 모든 데이터를 사용한 경우
- 설명 변수가 너무 많이 존재하여 생성한 모형이 복잡한 경우



가능한 다양한 정보가 포함되어 있는 데이터로 모델링 해야함

### Overfitting 방지 방법

#### ■ 훈련 데이터 / 테스트 데이터

- 데이터를 모형 생성에 직접적으로 사용되는 훈련 데이터와 테스트 데이터로 나누어서 사용함으로써 모형을 생성하는 경우는 훈련 데이터만을 사용하고 해당 모형의 성능을 평가하는 경우 테스트 데이터를 활용하여 모형을 비교함
- 보통 훈련 데이터와 테스트 데이터의 비율을 7:3이나 8:2의 비율로 사용함



#### ■ Cross Validation

- 데이터를 분할하여 모형 생성 및 적용에 번갈아 사용함으로써 모형을 일반화 하여 평가하고자 하는 방법으로 보통 데이터를 10개의 등분으로 나누어서 사용하는 10-fold cross validation 방법이 대표적임

- 랜덤 포레스트(Random Forest)는 의사 결정 나무(Decision Tree)에서의 단점인 과적합(overfitting)문제를 보완하는 방법으로 높은 예측력을 보인다는 장점을 가지고 있습니다.
- 그러나 의사 결정 나무에 비해 여러 개의 나무를 사용하기 때문에 해석의 어려움이 있습니다.

## 실습 데이터

### ■ 데이터 설명

- 위스콘신 유방암 데이터는 종양의 크기, 모양 등의 다양한 속성 값을 기반으로 해당 종양이 악성인지 양성인지를 분류한 데이터임
- 총 32개의 column과 569개의 row를 가지고 있음

- **radius** : 중심에서 외벽까지 거리
- **texture** : 질감
- **area** : 면적
- **perimeter** : 둘레
- **smoothness** : 매끄러운 정도
- **compactness** : 조그만 정도
- **concavity** : 오목함
- **points** : 오목한 점의 수
- **symmetry** : 대칭 정도
- **dimension** : 프랙탈 차원

평균값,  
표준 오차값,  
제일 큰 3개의 값을 평균낸 값

- **diagnosis** : 양성 여부 (1. 양성 ; 0. 악성)

	mean radius	mean texture	mean area	mean symmetry	DIAG RES
0	17.99	10.38	1001.0	0.2419	0
1	20.57	17.77	1326.0	0.1812	0
2	19.69	21.25	1203.0	0.2069	0
3	11.42	20.38	386.1	0.2597	0
4	20.29	14.34	1297.0	0.1809	0

## 실습 단계 설명

### ■ 최종 결과 확인 데이터

- 569개의 데이터에서 처음 50개의 row를 결과 비교를 위해 최종 결과 확인용 데이터로 사용함

### ■ 분석 데이터

- 569개의 데이터에서 처음 50개의 row를 제외한 519개의 row를 활용하여 비교함

### ■ 비교 방법

- 전체 데이터 활용 : 이전 실습에서 해본 것 처럼 519개의 모든 row를 활용하여 분석 모형 생성하고 가장 좋은 결과를 가진 모형을 선택함
- Train / Test 활용 : Train data와 Test data를 나누어 훈련용 데이터를 통해 모형을 생성한 후 Test 데이터를 기준으로 가장 좋은 모형을 선택함 (8:2의 비율을 사용함)

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from sklearn.datasets import load_breast_cancer
from os import system # Tree 시각화를 위한
system("pip install graphviz")
import graphviz
```

```
0
```

```
# 유방암 데이터 로드
cancer = load_breast_cancer()
```

```
# 유방암 데이터 DataFrame으로 변환
cancerDF = pd.DataFrame(cancer.data, columns = cancer.feature_names)
```

```
# 유방암 데이터의 타겟 변수와 정의 (1이면 양성 종양, 0이면 악성 종양)
cancerDF['diagnosis'] = cancer.target
# 보통 1을 주요 타겟이 되는 범주로 정의함
cancerDF['diagnosis'] = np.where(cancerDF['diagnosis'] == 0, 1, 0)
cancerDF.shape
```

```
(569, 31)
```

```
# 최종 테스트 데이터로 사용할 50개의 관측값 분류
final_testDF = cancerDF.loc[0:49]
final_testDF.head()
```

```
cancerDF = cancerDF.loc[50:]
cancerDF.head()
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean c
0	17.99	10.38	122.80	1001.0	0.11840	
1	20.57	17.77	132.90	1326.0	0.08474	
2	19.69	21.25	130.00	1203.0	0.10960	
3	11.42	20.38	77.58	386.1	0.14250	
4	20.29	14.34	135.10	1297.0	0.10030	

5 rows × 31 columns

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean
50	11.76	21.60	74.72	427.9	0.08637	
51	13.64	16.34	87.21	571.8	0.07685	
52	11.94	18.24	75.71	437.6	0.08261	
53	18.22	18.70	120.30	1033.0	0.11480	
54	15.10	22.02	97.26	712.8	0.09056	

## 모든 데이터를 활용하여 모형을 생성

### 1.1. 로지스틱 회귀분석 (Logistic Regression)

```
[51] from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler

# 타겟 변수 설명 변수 데이터 정의
y_target = cancerDF['diagnosis']
X_data = cancerDF.drop('diagnosis', axis = 1)

# 표준화 작업
scaler = StandardScaler()
scaler.fit(X_data)
X_scale = pd.DataFrame(scaler.transform(X_data), columns = X_data.columns)

# 로지스틱 회귀 모형 생성
Logistic = LogisticRegression()
Logistic.fit(X_scale, y_target)

# 예측 확률
probs = Logistic.predict_proba(X_scale)[:,1]

# 타겟 변수 예측
pred_logistic = np.where(probs.reshape(-1) >= 0.5, 1, 0)

StandardScaler()LogisticRegression()
```

### 1.2. 의사 결정 나무(Decision Tree)

```
[52] from sklearn import tree

# 의사결정나무(Decision Tree) 모형 생성
DTree = tree.DecisionTreeClassifier(random_state=300)
DTree.fit(X_data, y_target)

# 의사결정나무(Decision Tree) 모형 생성 (depth를 3으로 준 것)
DTree3 = tree.DecisionTreeClassifier(random_state=300, max_depth = 3)
DTree3.fit(X_data, y_target)

DecisionTreeClassifier(random_state=300)DecisionTreeClassifier(max_depth=3, random_state=300)

[59] # 타겟 변수 예측 (depth를 주지 않은 것과, 준 것)
pred_Tree = DTree.predict(X_data)
pred_Tree3 = DTree3.predict(X_data)
```

### 1.3. 랜덤 포레스트 (Random Forest)

```
[45] from sklearn.ensemble import RandomForestClassifier

# 100개의 decision tree를 사용한 랜덤 포레스트 (max_features는 sqrt개수를 사용하는 것이 디폴트임)
RF = RandomForestClassifier(n_estimators=100, random_state=0)
RF.fit(X_data, y_target)

# 예측값 생성
pred_RF = RF.predict(X_data)
```

## 모든 데이터를 활용한 모형 평가

```
print("Logistic 결과")
pd.DataFrame(confusion_matrix(y_target, pred_logistic))

print("Logistic 모형 평가")
print(classification_report(y_target, pred_logistic, digits=4))
```

Logistic 결과

	0	1
0	350	0
1	5	164

Logistic 모형 평가

	precision	recall	f1-score	support
0	0.9859	1.0000	0.9929	350
1	1.0000	0.9704	0.9850	169
accuracy			0.9904	519
macro avg	0.9930	0.9852	0.9889	519
weighted avg	0.9905	0.9904	0.9903	519

```
[49] print("Depth를 3으로 제한한 의사결정 나무 결과")
pd.DataFrame(confusion_matrix(y_target, pred_Tree3))

print("Depth를 3으로 제한한 의사결정 나무 모형 평가")
print(classification_report(y_target, pred_Tree3, digits=4))
```

Depth를 3으로 제한한 의사결정 나무 결과

	0	1
0	349	1
1	18	151

Depth를 3으로 제한한 의사결정 나무 모형 평가

	precision	recall	f1-score	support
0	0.9510	0.9971	0.9735	350
1	0.9934	0.8935	0.9408	169
accuracy			0.9634	519
macro avg	0.9722	0.9453	0.9572	519
weighted avg	0.9648	0.9634	0.9629	519

```
[48] print("Depth 제한 없는 의사결정 나무 결과")
pd.DataFrame(confusion_matrix(y_target, pred_Tree))

print("Depth 제한 없는 의사결정 나무 모형 평가")
print(classification_report(y_target, pred_Tree, digits=4))
```

Depth 제한 없는 의사결정 나무 결과

	0	1
0	350	0
1	0	169

Depth 제한 없는 의사결정 나무 모형 평가

	precision	recall	f1-score	support
0	1.0000	1.0000	1.0000	350
1	1.0000	1.0000	1.0000	169
accuracy			1.0000	519
macro avg	1.0000	1.0000	1.0000	519
weighted avg	1.0000	1.0000	1.0000	519

```
[53] print("랜덤 포레스트 결과(100 개의 tree 사용)")
pd.DataFrame(confusion_matrix(y_target, pred_RF))

print("랜덤 포레스트 모형 평가")
print(classification_report(y_target, pred_RF, digits=4))
```

랜덤 포레스트 결과(100 개의 tree 사용)

	0	1
0	350	0
1	0	169

랜덤 포레스트 모형 평가

	precision	recall	f1-score	support
0	1.0000	1.0000	1.0000	350
1	1.0000	1.0000	1.0000	169
accuracy			1.0000	519
macro avg	1.0000	1.0000	1.0000	519
weighted avg	1.0000	1.0000	1.0000	519

의사결정 나무  
(Depth 제한 없는)

랜덤 포레스트

## 훈련 데이터와 테스트 데이터의 활용 (8:2 비율 활용)

```
[57] # 훈련 데이터와 테스트 데이터를 8:2의 비율로 나눔
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X_data, y_target, test_size = 0.2, random_state = 123)
X_train.shape
X_test.shape
```

(415, 30)(104, 30)

### 2.1. 로지스틱 회귀분석 (Logistic Regression)

```
[61] # 표준화 작업
scaler = StandardScaler()
scaler.fit(X_train)
X_scale_tr = pd.DataFrame(scaler.transform(X_train), columns = X_train.columns)
X_scale_te = pd.DataFrame(scaler.transform(X_test), columns = X_test.columns)

# 로지스틱 회귀 모형 생성
Logistic = LogisticRegression()
Logistic.fit(X_scale_tr, y_train)

# 예측 확률 (TEST 데이터로 예측하여 평가)
probs_te = Logistic.predict_proba(X_scale_te)[:,-1]

# 타겟 변수 예측
pred_TEST_logistic = np.where(probs_te.reshape(-1) >= 0.5, 1, 0)
```

### 2.2. 의사 결정 나무(Decision Tree)

```
[62] # 의사결정나무(Decision Tree) 모형 생성
DTree_train = tree.DecisionTreeClassifier(random_state=300)
DTree_train.fit(X_train, y_train)

# 의사결정나무(Decision Tree) 모형 생성 (depth를 3으로 준 것)
DTree3_train = tree.DecisionTreeClassifier(random_state=300, max_depth = 3)
DTree3_train.fit(X_train, y_train)

# 타겟 변수 예측 (depth를 주지 않은 것과, 준 것)
pred_TEST_Tree = DTree_train.predict(X_test)
pred_TEST_Tree3 = DTree3_train.predict(X_test)
```

### 2.3. 랜덤 포레스트(Random Forest)

```
[63] # 100개의 decision tree를 사용한 랜덤 포레스트 (max_features는 sort개수를 사용하는 것이 디폴트임)
RF_train = RandomForestClassifier(n_estimators=100, random_state=0)
RF_train.fit(X_train, y_train)

# 예측값 생성
pred_TEST_RF = RF_train.predict(X_test)
```



8 : 2의 비율로 데이터를 랜덤하게 나눠서 활용함

519건 중 415건을 훈련 데이터로 활용  
519건 중 104건을 훈련 데이터로 활용



모형들을 비교하여 최종 모형을 선정하기 위해  
Test data를 활용하여 모형을 평가 함



## 테스트 데이터를 활용한 모형 평가

```
[69] print("Logistic 결과")
pd.DataFrame(confusion_matrix(y_test, pred_TEST_logisc))

print("Logistic 모형 평가")
print(classification_report(y_test, pred_TEST_logisc, digits=4))
```

Logistic 결과

	0	1
0	76	0
1	2	26

Logistic 모형 평가

	precision	recall	f1-score	support
0	0.9744	1.0000	0.9870	76
1	1.0000	0.9286	0.9630	28
accuracy			0.9808	104
macro avg	0.9872	0.9643	0.9750	104
weighted avg	0.9813	0.9808	0.9805	104

```
[70] print("Depth 제한 없는 의사결정 나무 결과")
pd.DataFrame(confusion_matrix(y_test, pred_TEST_Tree))

print("Depth 제한 없는 의사결정 나무 모형 평가")
print(classification_report(y_test, pred_TEST_Tree, digits=4))
```

Depth 제한 없는 의사결정 나무 결과

	0	1
0	73	3
1	4	24

Depth 제한 없는 의사결정 나무 모형 평가

	precision	recall	f1-score	support
0	0.9481	0.9605	0.9542	76
1	0.8889	0.8571	0.8727	28
accuracy			0.9327	104
macro avg	0.9185	0.9088	0.9135	104
weighted avg	0.9321	0.9327	0.9323	104

```
[71] print("Depth를 3으로 제한한 의사결정 나무 결과")
pd.DataFrame(confusion_matrix(y_test, pred_TEST_Tree3))

print("Depth를 3으로 제한한 의사결정 나무 모형 평가")
print(classification_report(y_test, pred_TEST_Tree3, digits=4))
```

Depth를 3으로 제한한 의사결정 나무 결과

	0	1
0	73	3
1	4	24

Depth를 3으로 제한한 의사결정 나무 모형 평가

	precision	recall	f1-score	support
0	0.9481	0.9605	0.9542	76
1	0.8889	0.8571	0.8727	28
accuracy			0.9327	104
macro avg	0.9185	0.9088	0.9135	104
weighted avg	0.9321	0.9327	0.9323	104

```
[72] print("랜덤 포레스트 결과(100 개의 tree 사용)")
pd.DataFrame(confusion_matrix(y_test, pred_TEST_RF))

print("랜덤 포레스트 모형 평가")
print(classification_report(y_test, pred_TEST_RF, digits=4))
```

랜덤 포레스트 결과(100 개의 tree 사용)

	0	1
0	75	1
1	4	24

랜덤 포레스트 모형 평가

	precision	recall	f1-score	support
0	0.9494	0.9868	0.9677	76
1	0.9600	0.8571	0.9057	28
accuracy			0.9519	104
macro avg	0.9547	0.9220	0.9367	104
weighted avg	0.9522	0.9519	0.9510	104

로지스틱 회귀모형

## 최종 결과 확인

```
[76] # 최종 데이터의 X 데이터 추출/ y 데이터 추출
y_fin = final_testDF[['diagnosis']]
X_fin = final_testDF.drop('diagnosis', axis = 1)

# 랜덤 포레스트와 의사결정나무 결과 확인
pred_FIN_RF = RF.predict(X_fin)
pred_FIN_Tree = DTree.predict(X_fin)
```

```
[79] # 스케일링
X_scale_fin = pd.DataFrame(scaler.transform(X_fin), columns = X_fin.columns)

# 예측 확률 (TEST 데이터로 예측하여 평가)
probs_fin = Logistic.predict_proba(X_scale_fin)[:,1]

# 타겟 변수 예측
pred_FIN_logistic = np.where(probs_fin.reshape(-1) >= 0.5, 1, 0)
```

## 랜덤 포레스트

```
print("랜덤 포레스트 결과(100 개의 tree 사용)")
pd.DataFrame(confusion_matrix(y_fin, pred_FIN_RF))

print("랜덤 포레스트 모형 평가")
print(classification_report(y_fin, pred_FIN_RF, digits=4))
```

랜덤 포레스트 결과(100 개의 tree 사용)

0 1

0 7 0

1 4 39

랜덤 포레스트 모형 평가

	precision	recall	f1-score	support
0	0.6364	1.0000	0.7778	7
1	1.0000	0.9070	0.9512	43
accuracy			0.9200	50
macro avg	0.8182	0.9535	0.8645	50
weighted avg	0.9491	0.9200	0.9269	50

의사결정 나무  
(Depth 제한 없는)

```
print("Depth 제한 없는 의사결정 나무 결과")
pd.DataFrame(confusion_matrix(y_fin, pred_FIN_Tree))

print("Depth 제한 없는 의사결정 나무 모형 평가")
print(classification_report(y_fin, pred_FIN_Tree, digits=4))
```

Depth 제한 없는 의사결정 나무 결과

0 1

0 7 0

1 3 40

Depth 제한 없는 의사결정 나무 모형 평가

	precision	recall	f1-score	support
0	0.7000	1.0000	0.8235	7
1	1.0000	0.9302	0.9639	43
accuracy			0.9400	50
macro avg	0.8500	0.9651	0.8937	50
weighted avg	0.9580	0.9400	0.9442	50

## 로지스틱 회귀모형

```
print("Logistic 결과")
pd.DataFrame(confusion_matrix(y_fin, pred_FIN_logistic))

print("Logistic 모형 평가")
print(classification_report(y_fin, pred_FIN_logistic, digits=4))
```

Logistic 결과

0 1

0 7 0

1 1 42

Logistic 모형 평가

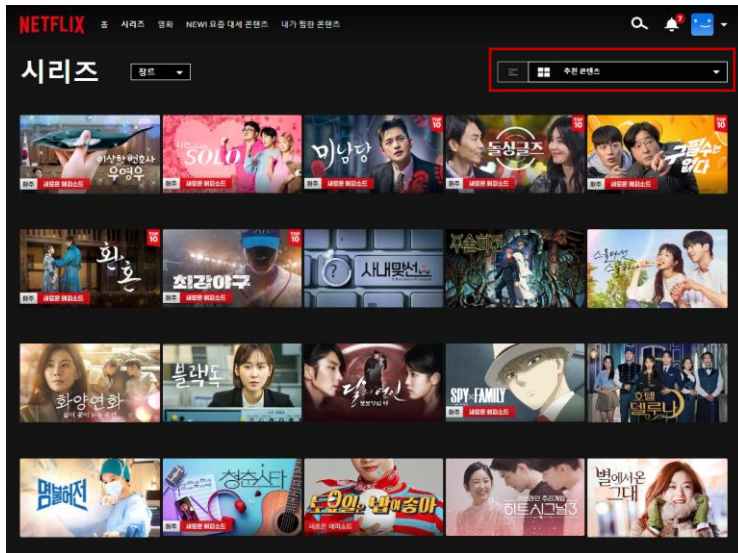
	precision	recall	f1-score	support
0	0.8750	1.0000	0.9333	7
1	1.0000	0.9767	0.9882	43
accuracy			0.9800	50
macro avg	0.9375	0.9884	0.9608	50
weighted avg	0.9825	0.9800	0.9805	50

최종적으로 결과를 확인해 볼 때 모든 데이터를 활용한 모형은 overfitting 문제로 비교적 예측성능이 떨어졌으나  
훈련 데이터와 테스트 데이터를 나눠 사용한 경우 여전히 성능이 높은 것을 알 수 있음

- 추천 시스템은 고객(사용자)들의 과거 구매 및 선택에 대한 데이터 분석을 통해 수 많은 선택지들 중 가장 선호할 수 있는 상품들을 예측하여 추천해주는 알고리즘입니다.

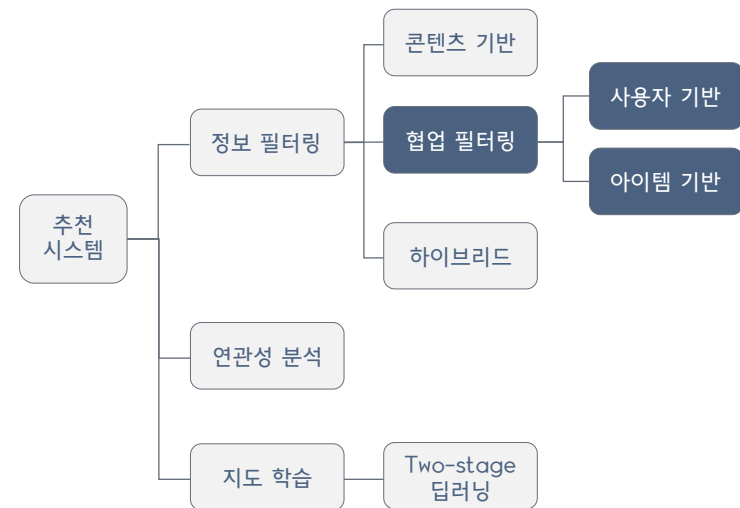
## 추천 시스템

- 추천 시스템은 고객의 과거 데이터를 통하여 선호도를 예측하고 적절한 상품을 추천해주는 알고리즘임
- 추천시스템을 사용하는 범위는 온라인 쇼핑이나 넷플릭스, 티빙 등과 같은 OTT서비스 등이 있음



## 대표 알고리즘

- 추천 시스템에는 다음과 같은 대표 알고리즘들이 있음
- 추천 시스템은 크게 정보 필터링 / 연관성 분석 / 지도 학습으로 나눌 수 있음
- 협업 필터링은 정보 필터링 알고리즘 방법의 하나로 협업 필터링에 대해 간략히 소개함



- 추천 시스템의 알고리즘 중 협업 필터링(Collaborative Filtering) 알고리즘은 사용자 기반(User-based)방법과, 아이템 기반(Item-based)방법의 알고리즘으로 나누어질 수 있습니다.

## 사용자 기반(User-based)

- 사용자 기반 협업 필터링(User-based collaborative Filtering)은 추천 대상 고객이 선정되는 경우, 구매 이력을 바탕으로 추천 대상 고객과 다른 사용자들 중 가장 유사한 사용자가 선호하는 아이템을 추천함



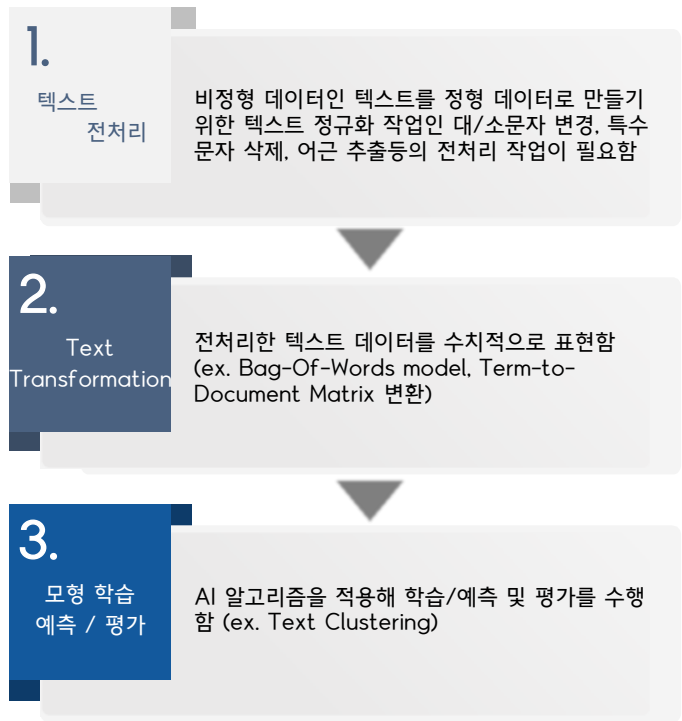
## 아이템 기반(Item-based)

- 아이템 기반 협업 필터링(Item-based collaborative Filtering)은 추천 대상 아이템을 기준으로 구매 이력 혹은 선호도를 바탕으로 유사도를 측정하여 가장 유사한 아이템을 선호한 사용자에게 추천 대상 아이템을 추천

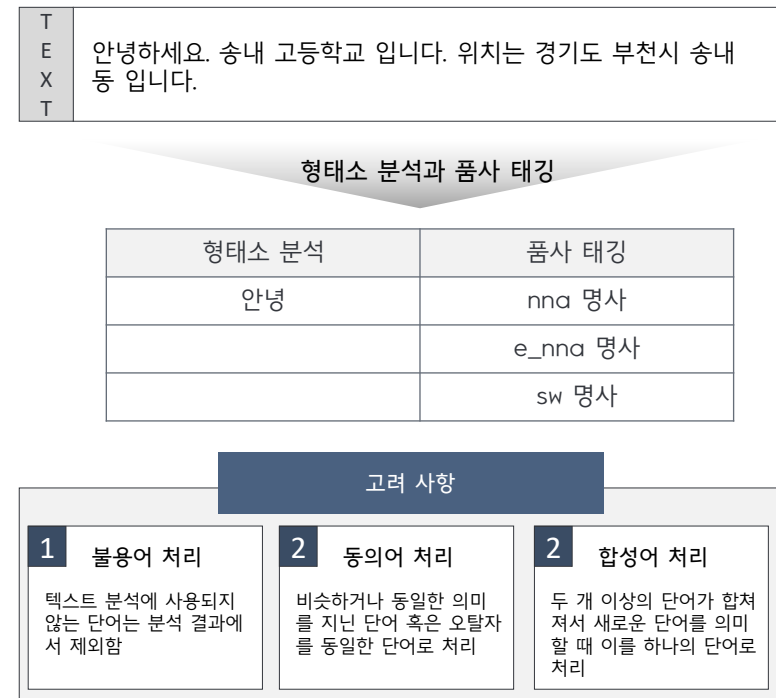


- 텍스트 마이닝(Text Mining)은 비정형 데이터인 텍스트에서 의미 있는 정보를 추출하는 과정을 의미합니다.
- 텍스트 마이닝을 수행하기 위한 기본 프로세스는 기본적인 데이터 분석 단계와 같이 텍스트의 전처리과정과 Transformation 과정이 있으며, 이를 통해 정형화된 데이터를 활용하여 모델을 개발하고 평가하는 순서를 가지고 있습니다.

## 텍스트 마이닝 수행 프로세스



## 전처리를 위한 NLP



- Bag-Of-Words model은 문서가 갖고 있는 모든 단어의 문맥이나 순서를 무시하고 일괄적으로 단어에 대해 빈도 값을 부여하여 데이터를 수치적으로 표현하는 모델입니다.

## Bag-Of-Words model



### ■ BOW 방식

- 카운트 기반의 벡터화
  - 카운트 값이 높을 수록 중요한 단어로 인식
  - 언어 특성상 문장에서 자주 사용되는 단어까지 높은 값을 부여
- TF-IDF(Term Frequency -Inverse Document Frequency)
  - 개별 문서에서 자주 나타나는 단어에 높은 가중치를 줌
  - 모든 문서에서 전반적으로 자주 나타나는 단어에 패널치를 부여

## Classification

### ■ 영화 리뷰에 대한 데이터

ID	Sentiment	review
1234	(긍정) 1	With all this stuff going down at the moment ...
2544	(긍정) 1	"The Classic War of the World" by Timothy ...
7492	(부정) 0	The film starts with a manager (Nicholas Bell...

this : 38492  
is : 24053  
terrible : 23152  
of : 39421



BOW  
model



AI 알고리즘  
(분류 모형)

- 텍스트 자료들을 정제 후, BOW 방식을 통해 데이터를 수치적으로 표현
- 영화 평가를 긍정과 부정으로 분류하기 위해 AI 알고리즘을 사용하여(ex. Logistic) 학습 및 예측
- 이러한 긍정/부정을 분류하는 분석을 세부적으로 감성 분석이라고도 함