

2022

스마트교통 빅데이터 분석

신산업 대응 데이터 및 인공지능 역량강화 교육



2022.9.21.

이홍석 (hsyi@kisti.re.kr)



DAY 1

❖ 성명 : 이홍석

❖ 소속

- ✓ KISTI 데이터기반문제해결연구단 책임연구원
- ✓ 과학기술연합대학원(UST-KISTI) 응용AI전공 교수

❖ 연구내용

- ✓ 딥러닝기반 도심지 교통혼잡 예측 (2018~2021)
- ✓ 지능형 인프라 기술 연구 (2018~2019)
- ✓ 도심지 교통예측을 위한 트랜스포머 모델 연구(2022~2025)

❖ 컴퓨팅 기술

- ✓ GPU 기반 가속컴퓨팅 기술 : CUDA, OpenCL
- ✓ 슈퍼컴퓨팅 병렬처리 기술 : MPI, OpenMP, ManyCore Computing
- ✓ Deep Learning 기술 : Tensorflow, PyTorch, Keras, Theano

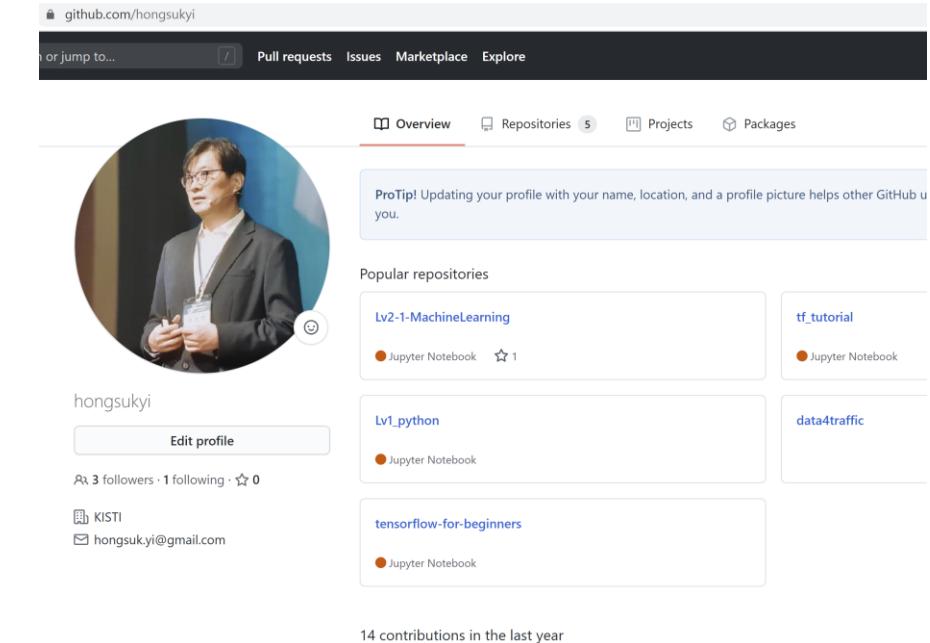


❖ 교육 경험: 머신러닝, 딥러닝, 인공지능, 빅데이터, MPI/OpenMP, CUDA, OpenCL 등

- ✓ 딥러닝 워크숍 개최 (2015)
- ✓ 한국교통학회 텐서플로우 튜토리얼 강좌 (2016~2018)
- ✓ 교통학회/KIRD/KISTI 과학데이터스쿨 강의 진행 (2018~2022)
- ✓ 교통 빅데이터 교육 (2022)
- ✓ UST 인공지능 특론 (전공필수) (2022)
- ✓ UST 인공지능 이해 (교양) (2021~2022)
- ✓ UST 기계학습의 이해, 인공지능 기초 (2019~2022)

❖ 참고자료

- ✓ 인공지능, 딥러닝 강의 자료 모음
(github.com/hongsuk.yi)



❖ 수업 운영

- ✓ 이론
- ✓ 실습(예제)

❖ 강의자료

- ✓ https://github.com/hongsukyi/Traffic_Data_ML

❖ 실습관련

- ✓ Google Colab 활용
 - 브라우저에서 파이션을 작성하고 실행할 수 있음
 - 실행환경 구성 필요 없음
 - GPU 무료 액세스
 - 간편한 공유
- ✓ 실습 시 개인별 Google 계정 필요

강의 1

인공지능 소개

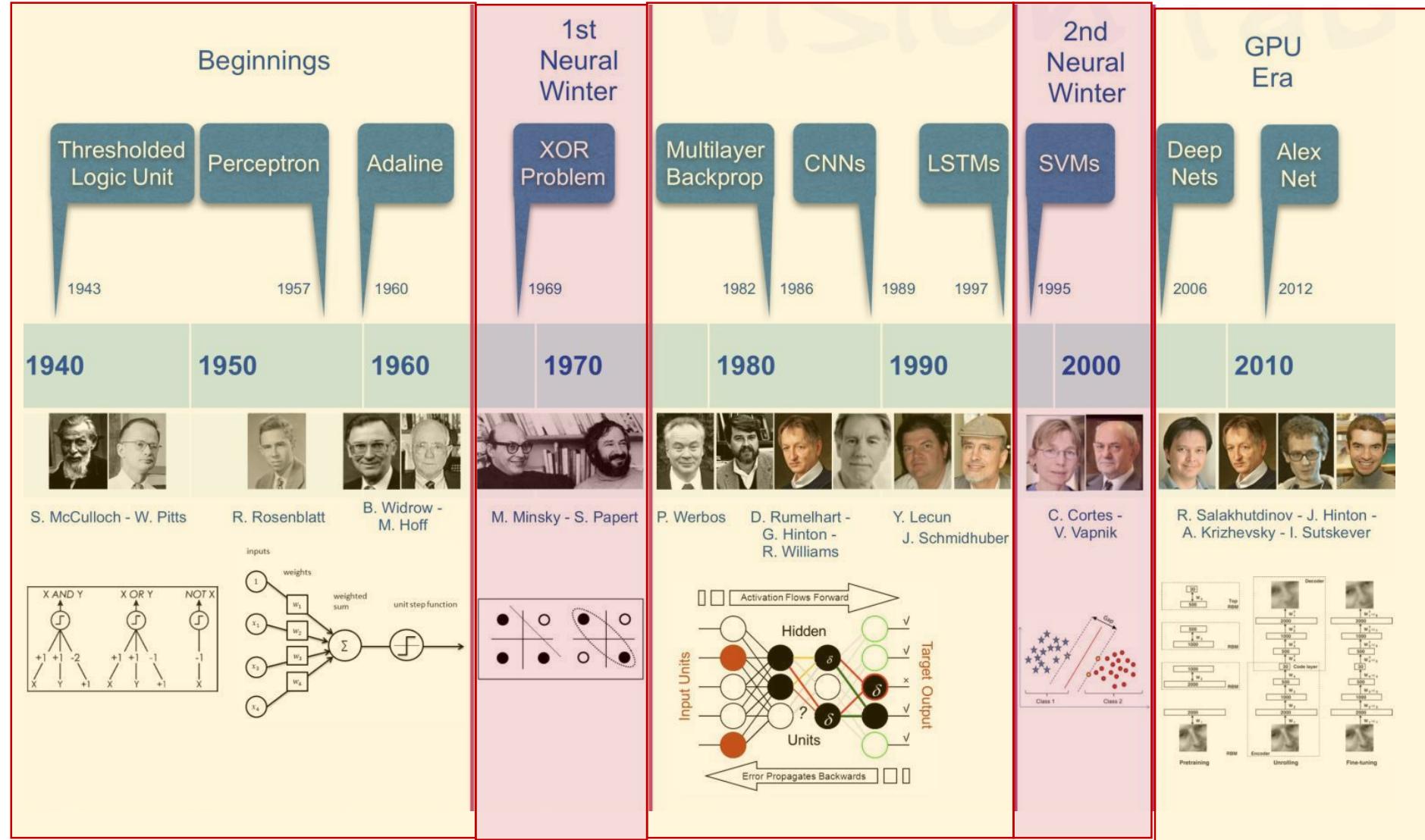
❖ 인공지능의 역사에 대하여 이해한다.

- ✓ 딥러닝과 머신러닝의 관계
- ✓ 지도학습, 비지도학습, 강화학습 구분
- ✓ 지도학습을 위한 데이터 특성과 라벨의 중요성을 이해한다.

❖ 인공지능을 구현하기 위한 SW 프레임워크

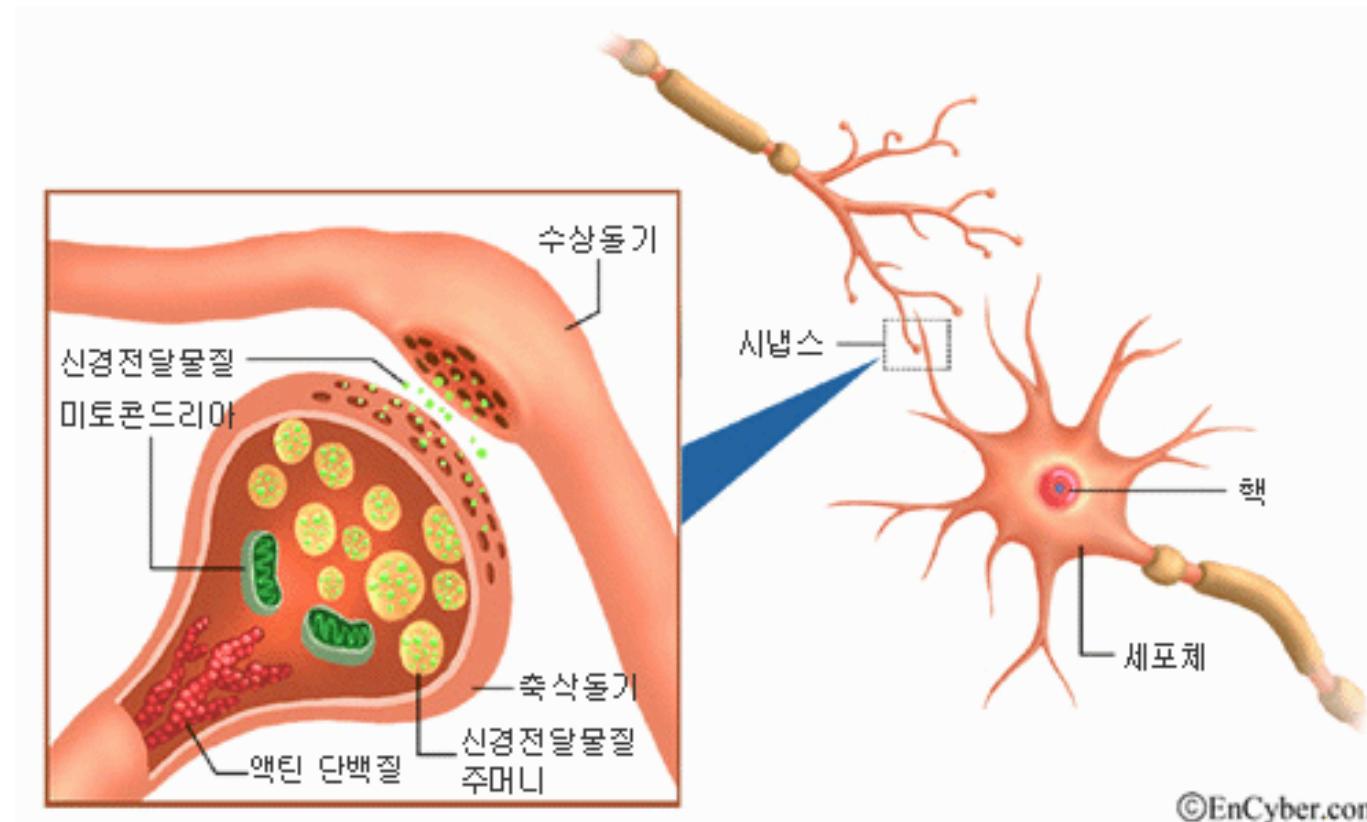
- ✓ Sklearn
- ✓ Tensorflow
- ✓ Pytorch
- ✓ Keras
- ✓ 등

History of Artificial Intelligence



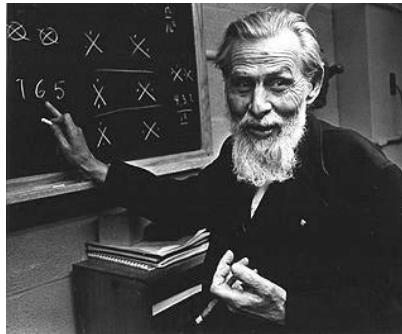
시냅스 (Synapse): 자극의 전달

뉴런의 집합체로 한 뉴런의 축삭돌기 말단과 다음 뉴런의 수상돌기 사이의 연접 부위



❖ 인공신경망 개념 최초 제안 : 맥컬록-피츠 신경망 모델 (1943)

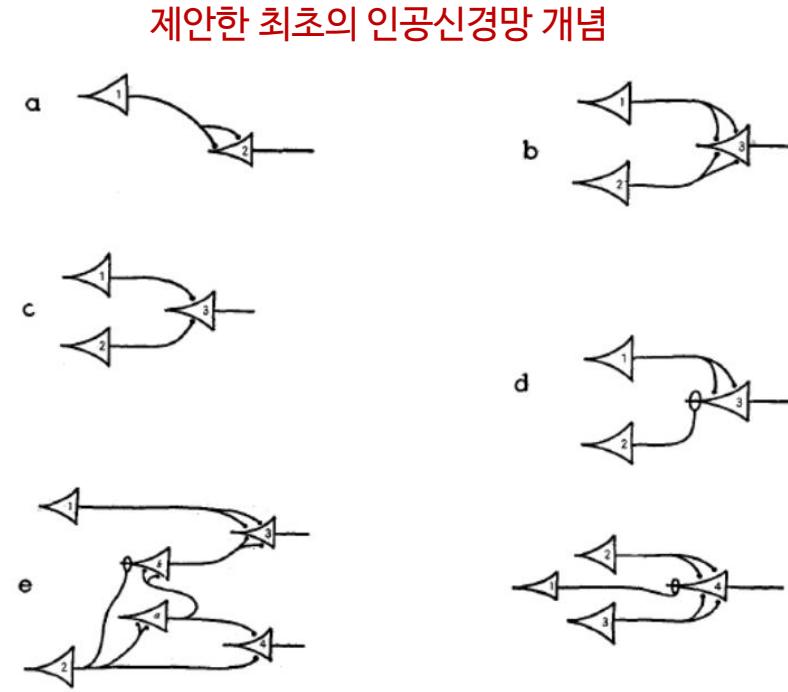
- ✓ McCulloch와 Pitts가 1943년에 ANN 최초 논문이 발표
 - “A logical calculus of the ideas immanent in nervous activity”
- ✓ 인간의 신경 구조를 복잡한 스위치들이 연결된 네트워크로 표현할 수 있다



맥컬록

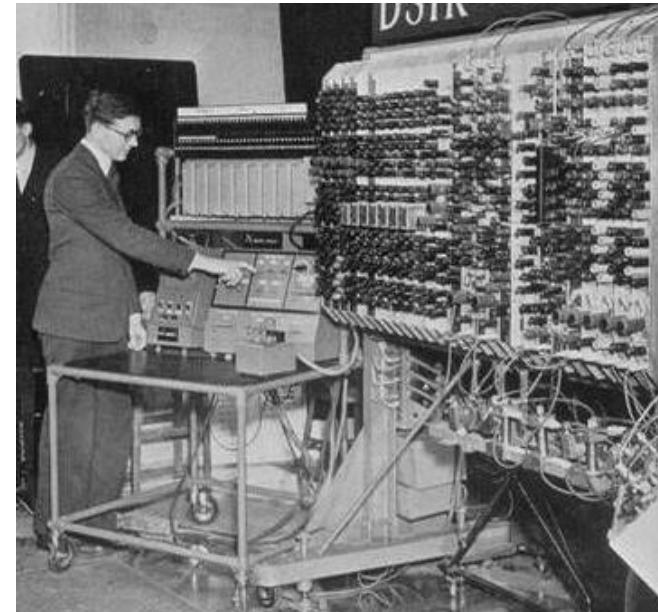
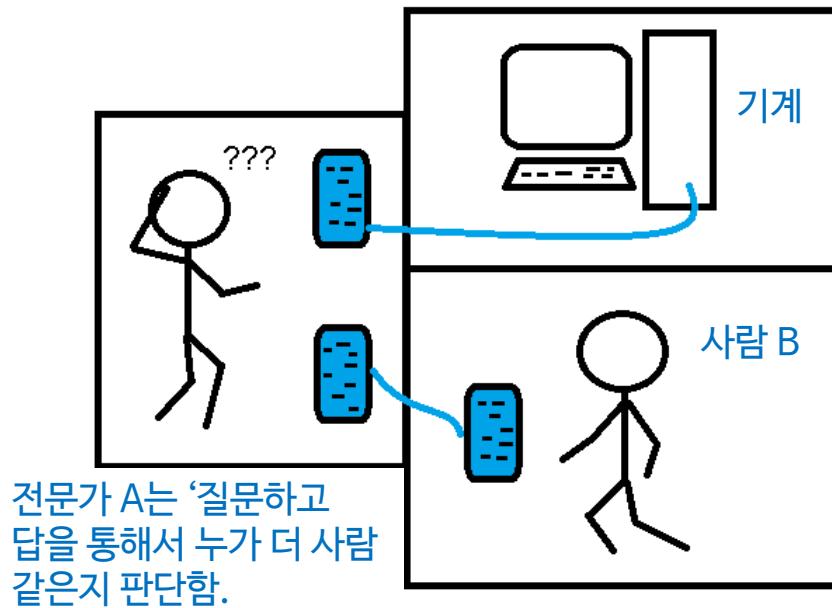


피츠



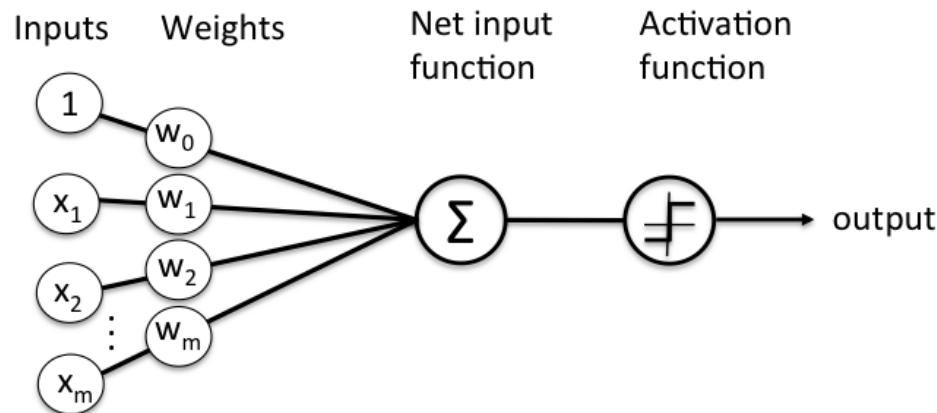
❖ 튜링 테스트 (1950)

- ✓ 앨런 튜링(Turing)은 컴퓨터 과학의 아버지로 불리움.
- ✓ 기계가 사람처럼 생각할 수 있다는 것을 아래 그림처럼 테스트 함.
- ✓ 사람 A가 상태에서 2명과 대화를 했을 때, 기계(Z)가 더 자연스러움.

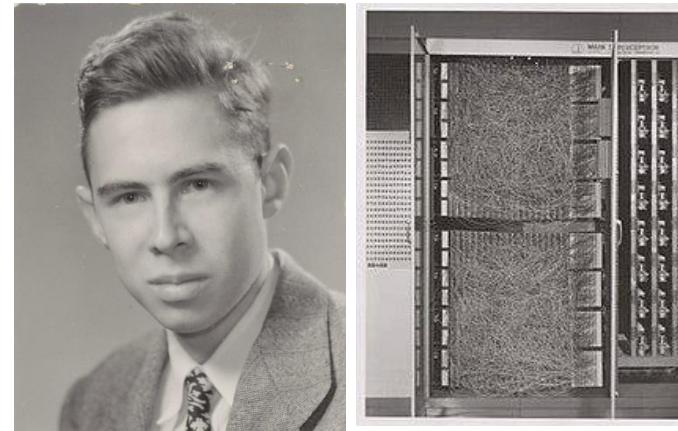


❖ ANN은 퍼셉트론 신경망 설명 : 프랑크 로젠블라트, 1958)

- ✓ Frank Rosenblatt는 퍼셉트론(Perceptron)라는 선형분류 피드포워드 신경망
 - “The perceptron: A probabilistic model for information storage and organization in the brain.” 논문에서 제시
 - 입력과 가중치(weight)들의 곱을 모두 더한 뒤 활성함수(계단함수)로 선형 분류기
- ✓ (문제점) 1개 퍼셉트론으로 XOR 선형 분류를 설명할 수 없음.



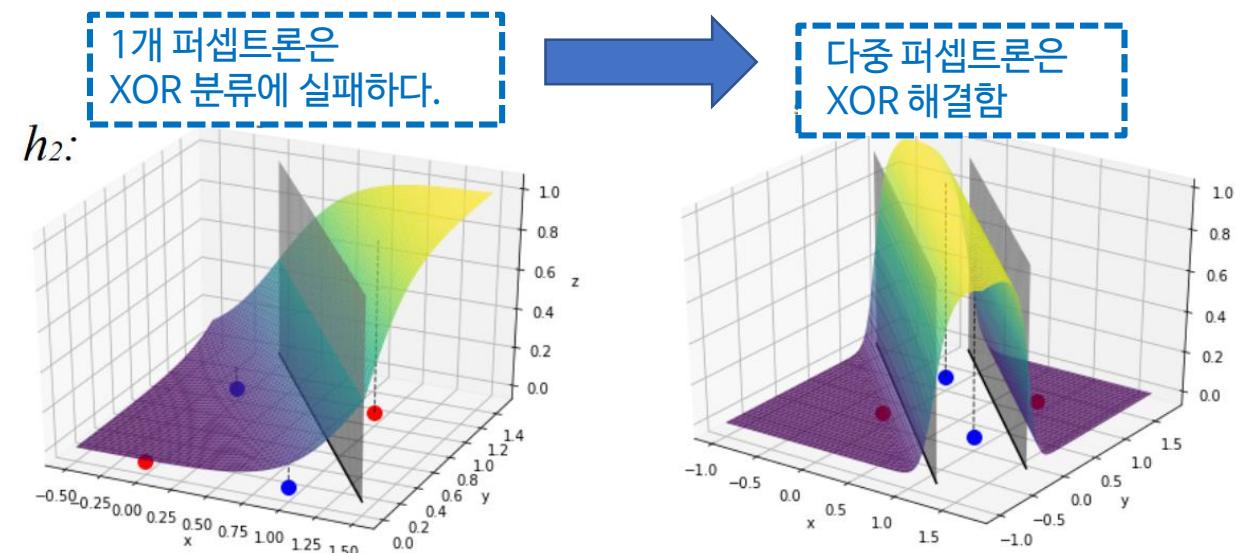
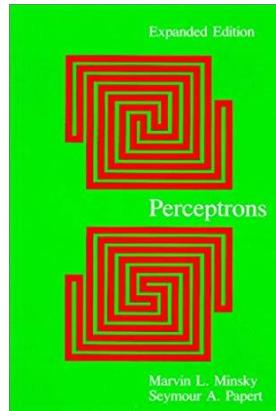
퍼셉트론은 현대의 딥러닝의 기초이다. 그 당시, 퍼셉트론을 통해서 진짜 인간과 같은 인공지능을 만들 수 있다는 기대가 매우 컸다.



로젠블라트 IBM 퍼셉트론 계산기

❖ 퍼셉트론의 무용론 등장으로 1차 인공지능 겨울이 시작된다.

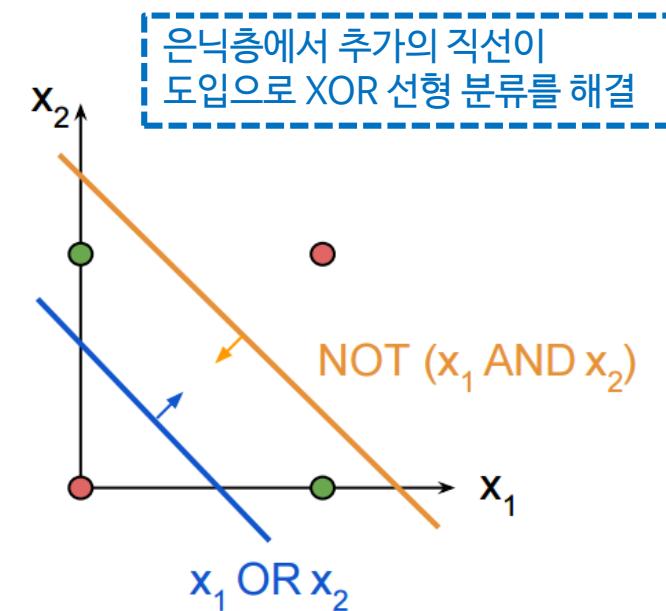
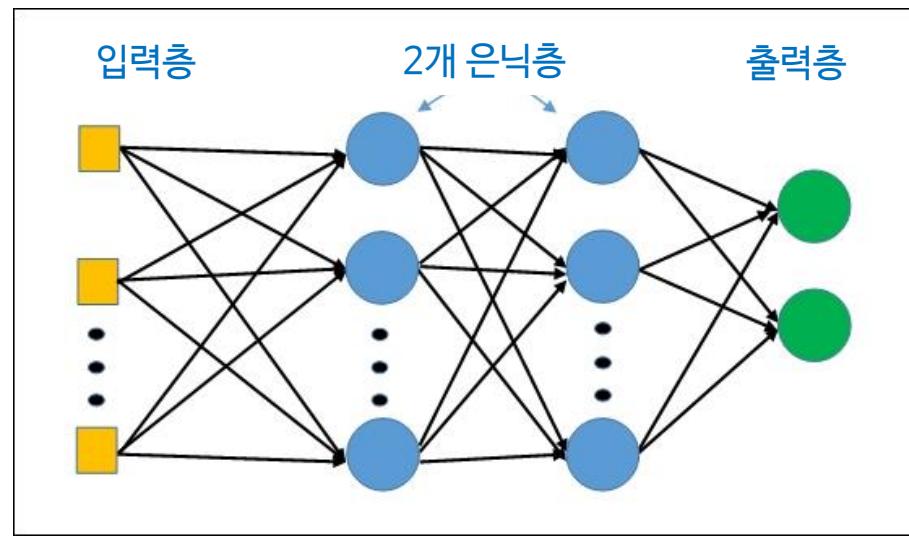
- ✓ 1969년 마빈 민스키는 '퍼셉트론'은 단순 선형 분류기이며, 'XOR' 분류도 할 수 없는 미미한 선형분류기라는 것을 수학적으로 증명함
- ✓ 퍼셉트론 인기가 사그라 들면서 인공지능 1차 암흑기가 도래한다.



1개의 퍼셉트론은 XOR 문제에서 빨강과 파랑색을 구분하는 초평면을 만들수 없다.

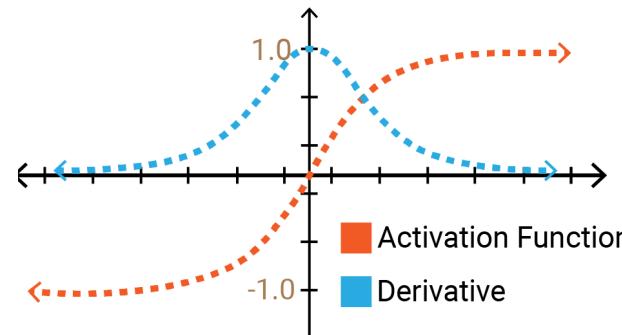
❖ 다층 퍼셉트론과 역전파 알고리즘 등장

- ✓ 다층 퍼셉트론은 전방향(feed-forward) 신경망으로 중간에 은닉층을 추가함
- ✓ 기존의 퍼셉트론이 선형 분류기라는 한계에 의해 XOR 문제를 해결할 수 없었다면,
- ✓ 다층 퍼셉트론은 은닉층(hidden layer)라는 중간 레이어를 추가로 XOR 문제를 해결
- ✓ (문제점) 다층 퍼셉트론은 은닉층의 추가로 신경망을 훈련에 많은 어려움이 있다.

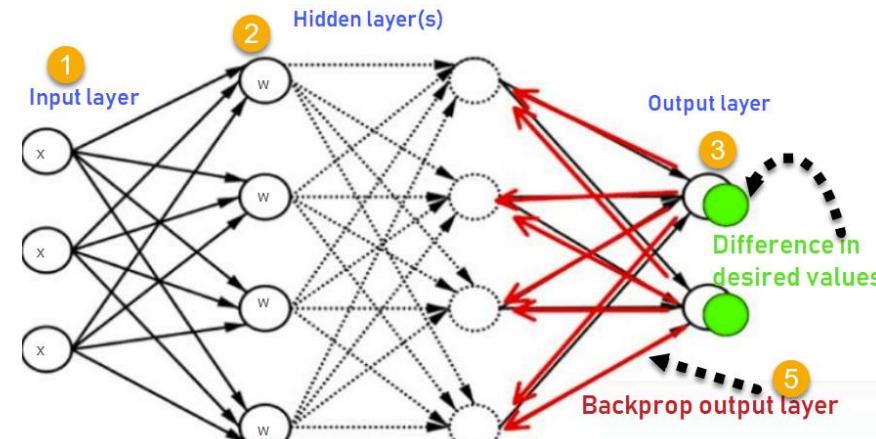


❖ 다층 퍼셉트론과 역전파 알고리즘 등장

- ✓ 1986년 McClelland, James L., David E. Rumelhart, and Geoffrey E. Hinton은 Backpropagation Algorithm을 제안해서 이 문제를 해결
- ✓ 오류 역전파 알고리즘은 Feedforward 연산 이후, 오차를 후방(Backward)으로 다시 보내 줌으로써, 많은 노드를 가진 MLP라도 최적의 가중치와 Bias를 학습할 수 있다.

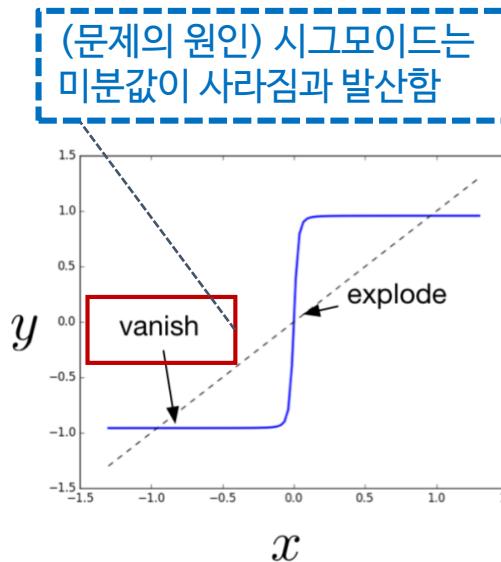
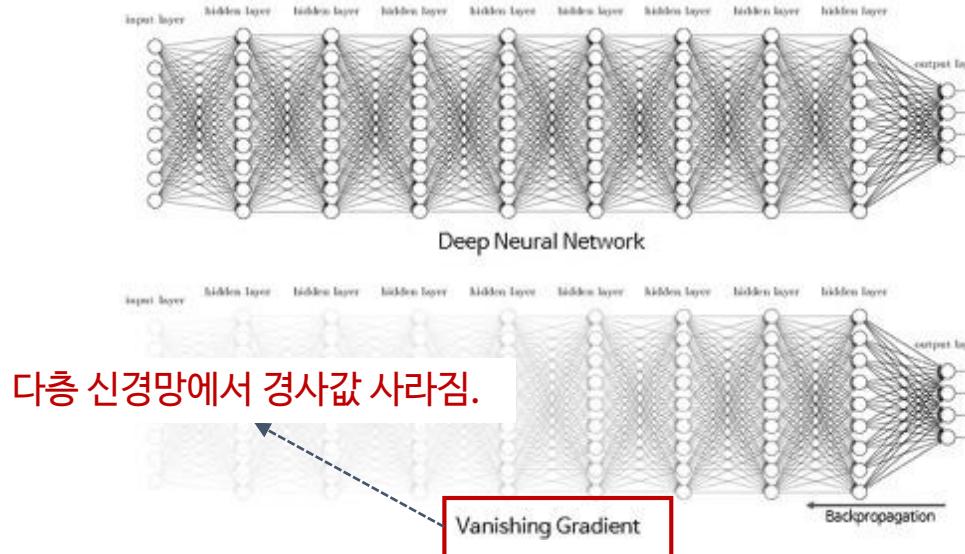


활성함수로 Sigmoid 함수와 미분 가능함.



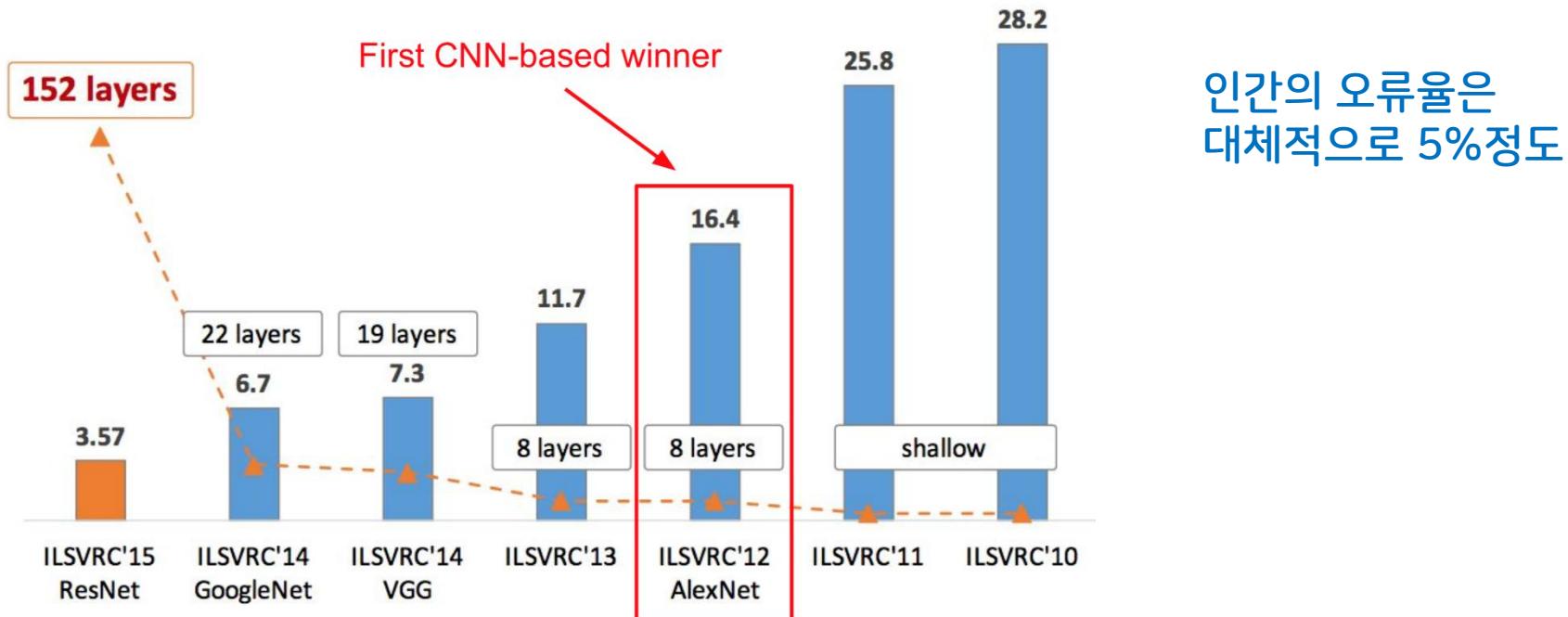
❖ 다층 퍼셉트론의 경사 발산과 소멸 문제 등장(1997)

- ✓ RNN에서 VGP(Vanishing Gradient Problem) 문제 (1993)
- ✓ LSTM(Long Short Term Memory)로 VGP 해결(1997), Hochreiter
- ✓ 이시기는 기계학습 SVM, Random Forest 등이 큰 인기가 있음.



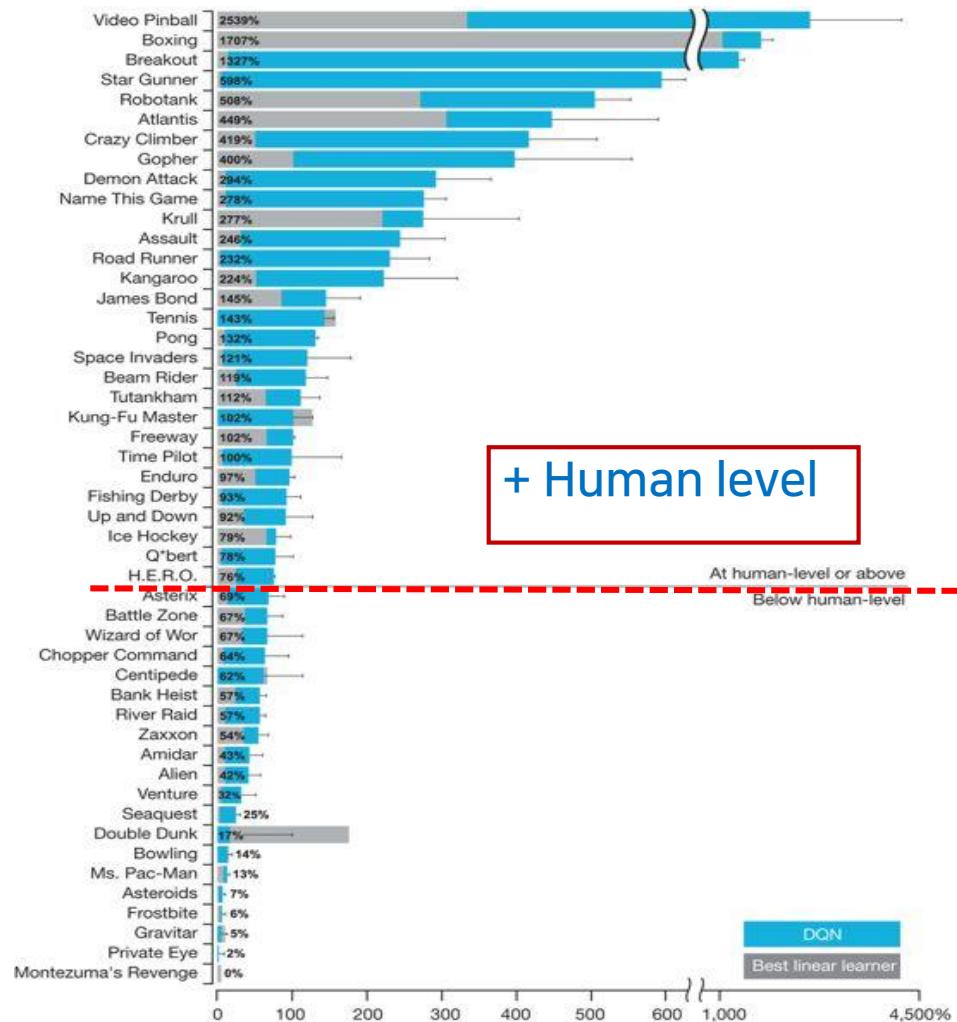
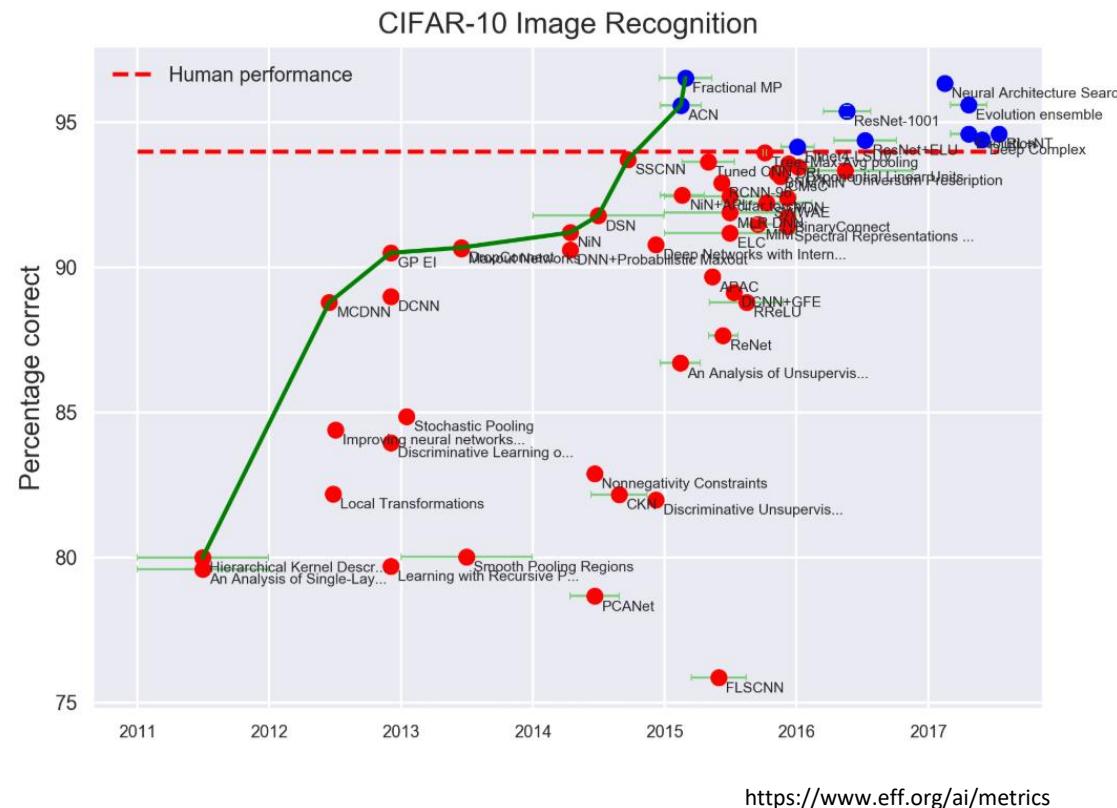
❖ 신경망 대신 딥러닝으로 인공지능 분야의 부활 (2006~)

- ✓ 드롭아웃 층(dropout layer) 도입으로 과적합 문제를 해결
- ✓ ReLU(Rectified Linear Unit) 활성함수 도입으로 기울기 사라짐 문제 해결
- ✓ GPU 컴퓨팅과 고속 최적화 알고리즘 등장
- ✓ ILSVRC에서 오류율을 15%로 획기적으로 낮춤



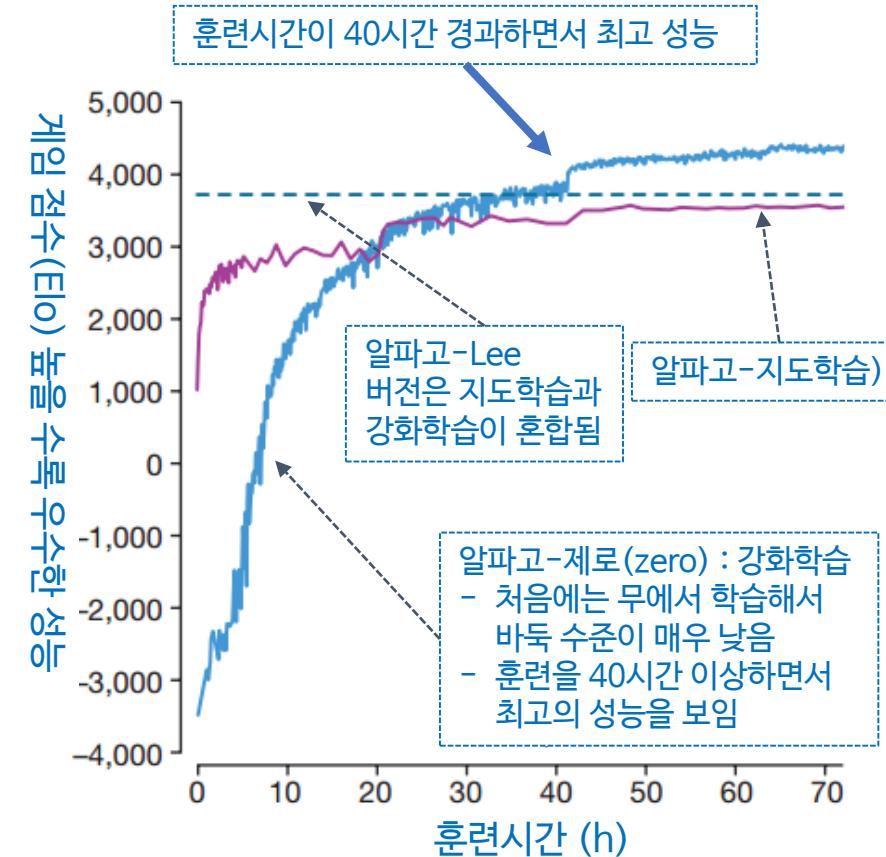
ANN의 역사 (12): 이미지 및 비전

구글 딥마인드, Nature 2015



알파고(AlphaGo)는 3개 버전

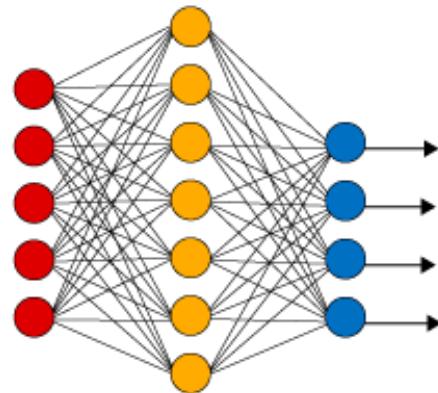
- ① 알파고-지도학습(바둑 기보)
- ② 알파고-리 (이세돌 9단)
- ③ 알파고-제로 (강화학습)



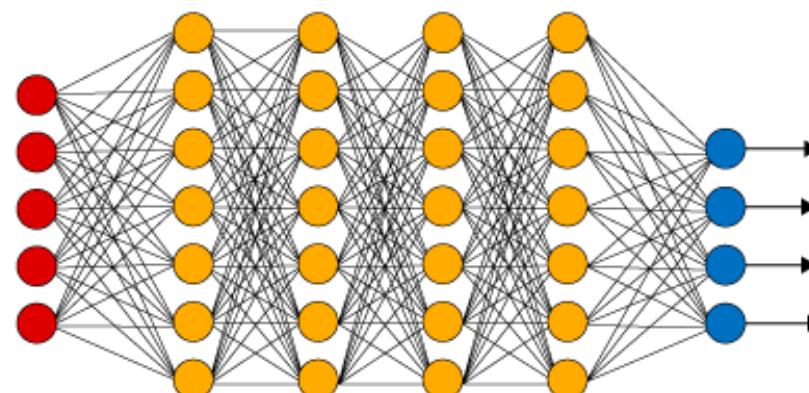
자료: Mastering the game of Go without human knowledge , David Silver, et al. Nature(2017)

딥러닝은 은닉층 2개 이상으로 구성된 인공신경망으로 정의함.

A. 단순 인공신경망(Shallow)
1개의 은닉층과 7개의 뉴런으로 구성



B. 딥러닝 구조
4개의 은닉층과 4x7(28)개의 뉴런으로 구성



● Input Layer

● Hidden Layer

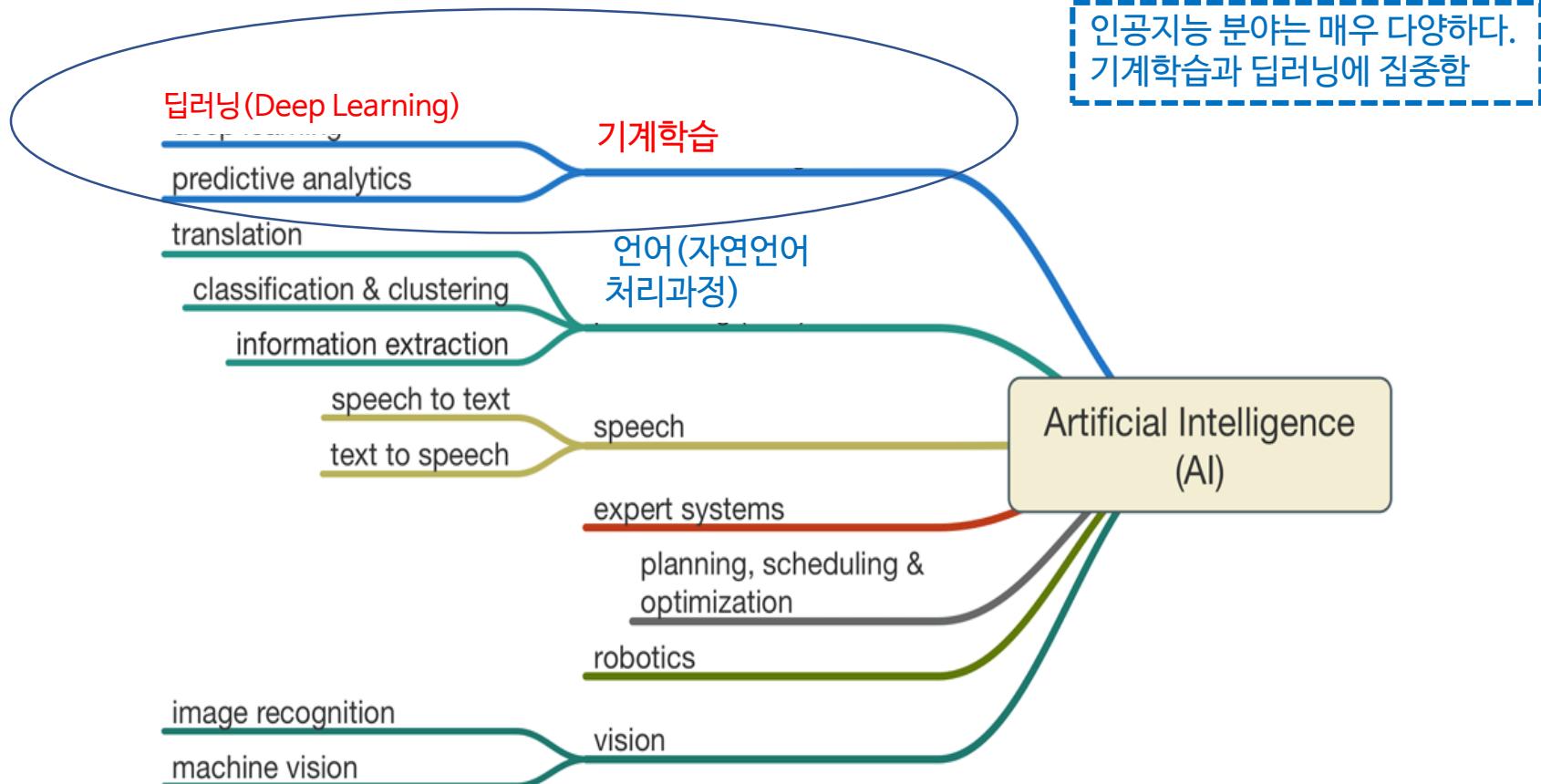
● Output Layer

자료: <https://www.xenonstack.com/blog/data-science/log-analytics-deep-machine-learning-ai/>

(퀴즈) 단순 인공신경망 II(Shallow)

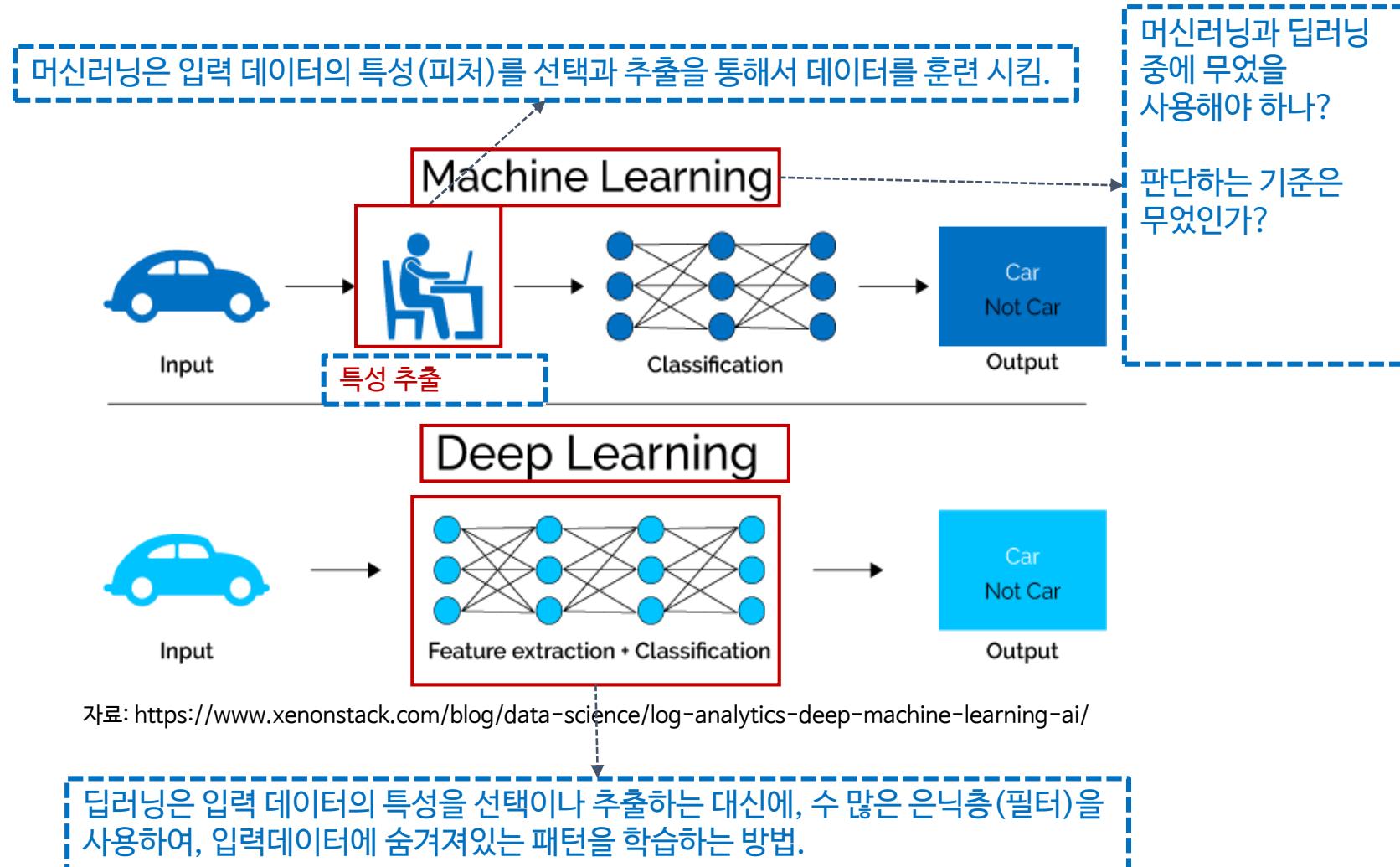
만일, 1개의 은닉층과 28개의 뉴런으로 구성되었다면, B와 비교하면 무엇이 다를까?

인공지능의 분류 (1)



자료: <https://www.eyerys.com/articles/paving-roads-artificial-intelligence-its-either-us-or-them>

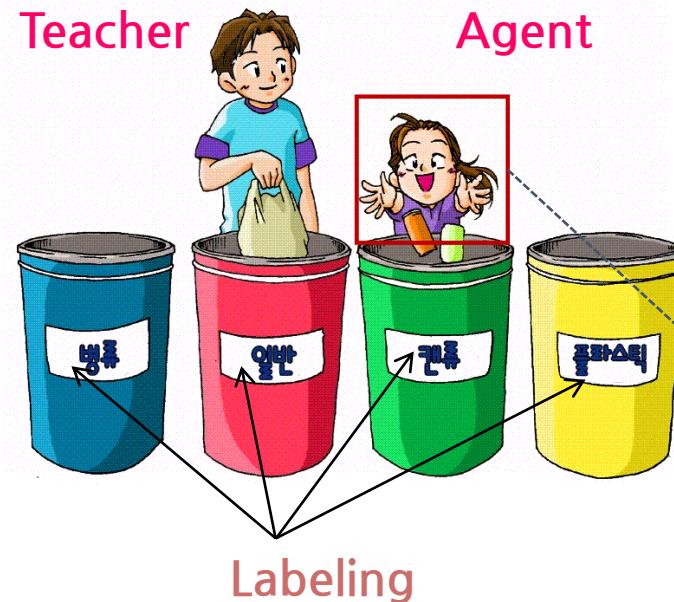
인공지능의 분류 (2)



Supervised Learning (지도학습)

- ① 입력(x), 출력(y)
- ② Labeling
- ③ Prediction

빅데이터 : 원본 데이터
훈련데이터+테스트 데이터로 나눔
비율은 7:3정도가 일반적임.



훈련데이터와 테스트 데이터

기존 훈련 데이터로 학습을 완료한 이후에,
새로운 테스트 데이터(쓰레기)를 누구의
도움도 없이 혼자서 분류를 한다면, 분류의
정확도는 얼마나 될까요?

지도학습과 레이블(Label) (2)

대표적인 딥러닝(머신러닝) 기반 지도학습을 위한 기초 데이터 제공함.
MNIST 데이터는 훈련용으로 60,000장 이미지와, 테스트용으로 10,000장

label = 5



label = 0



label = 4



label = 1



label = 9



label = 2



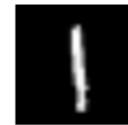
label = 1



label = 3



label = 1



label = 4



label = 3



label = 5



label = 3



label = 6



label = 1



label = 7



label = 2



label = 8



label = 6



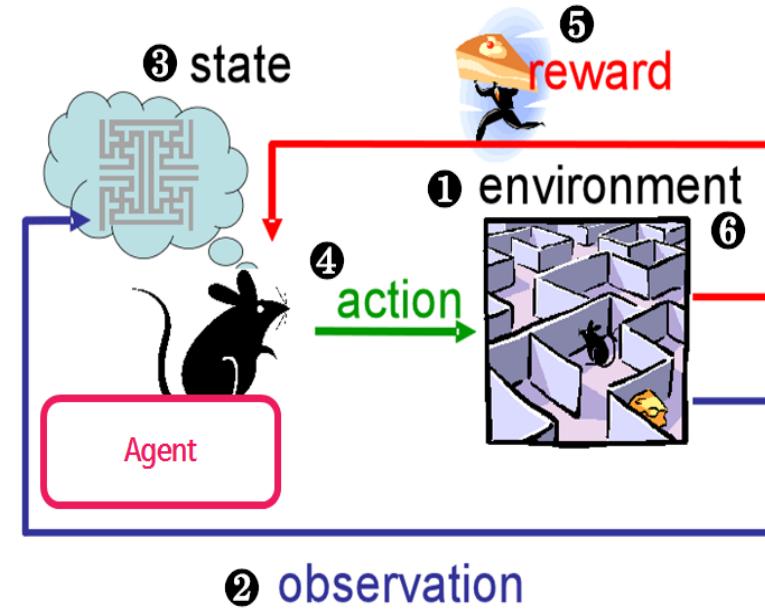
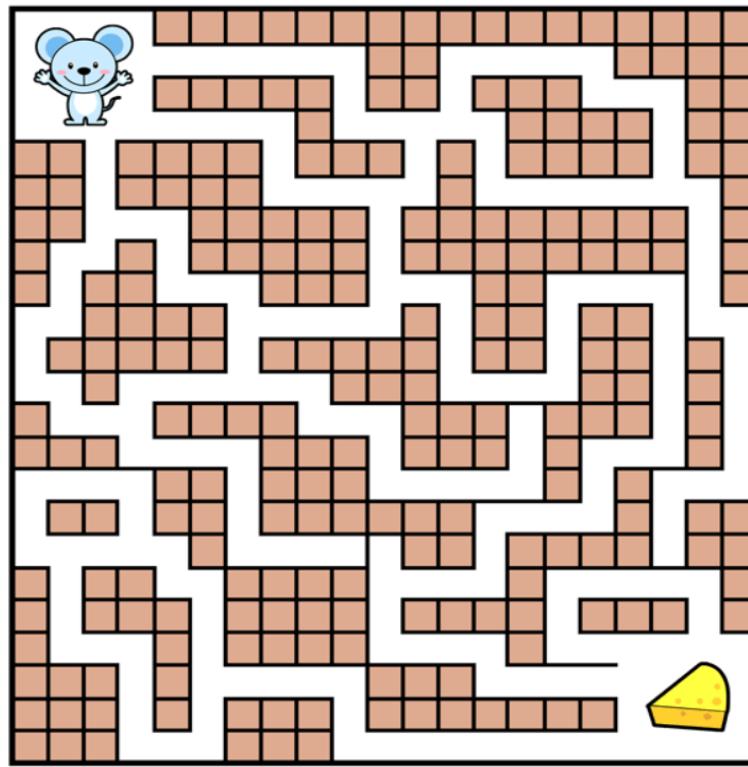
label = 9



모든 손글씨 이미지
마다 라벨링이 되어
있음.
이미지 9와 라벨 '9'는
항상 쌍으로 저장됨

강화학습 : 쥐가 치즈를 먹을 수 있을까?

마르코프 결정 프로세스 (MDP) : 지금 이순간 최적의 행동을 선택한다



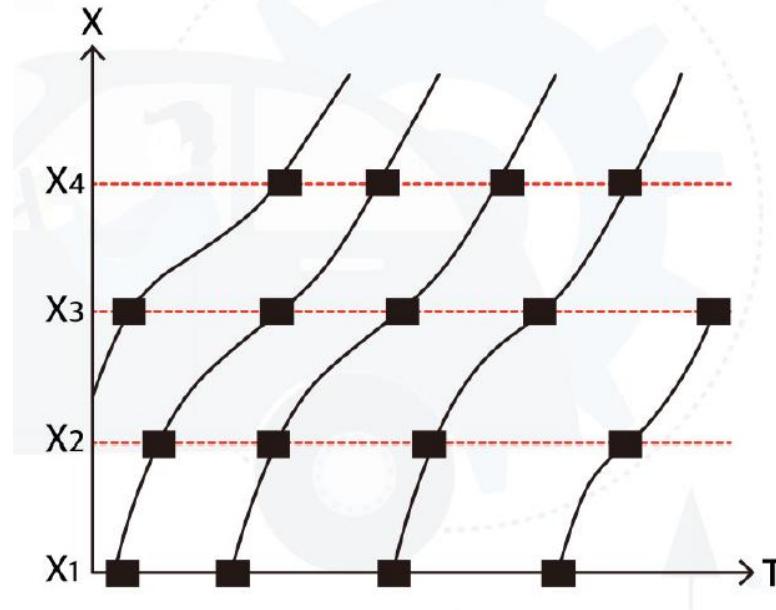
강의 2 교통 데이터 소개

❖ 고정위치 관측

- ✓ 도로상의 고정된 위치에서 지나가는 차량들을 관측
- ✓ 고정된 시각에 일정 구간의 교통 상태를 관측
- ✓ 항공 사진 촬영

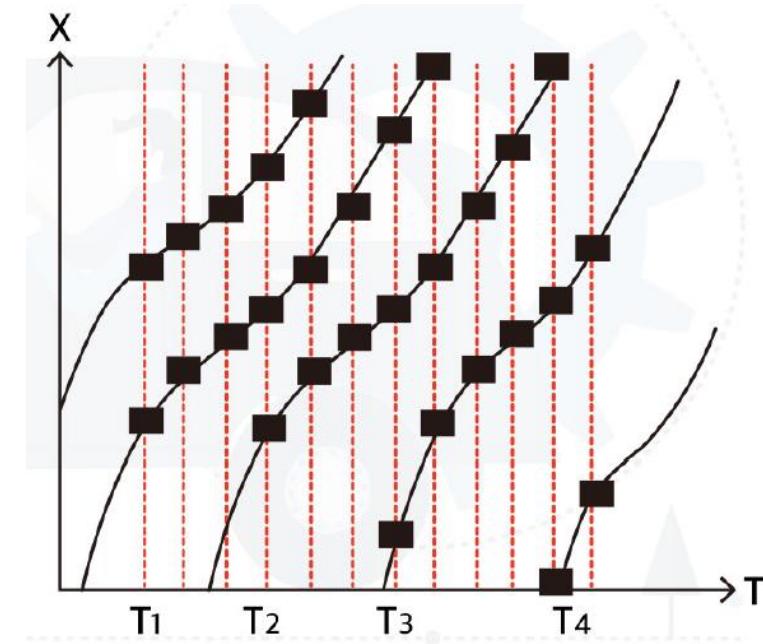
❖ 관측가능한 데이터는

- ✓ 통과 차량수, 차량 사이의 시간 간격, 차량 속도



• 연속 시간 관측

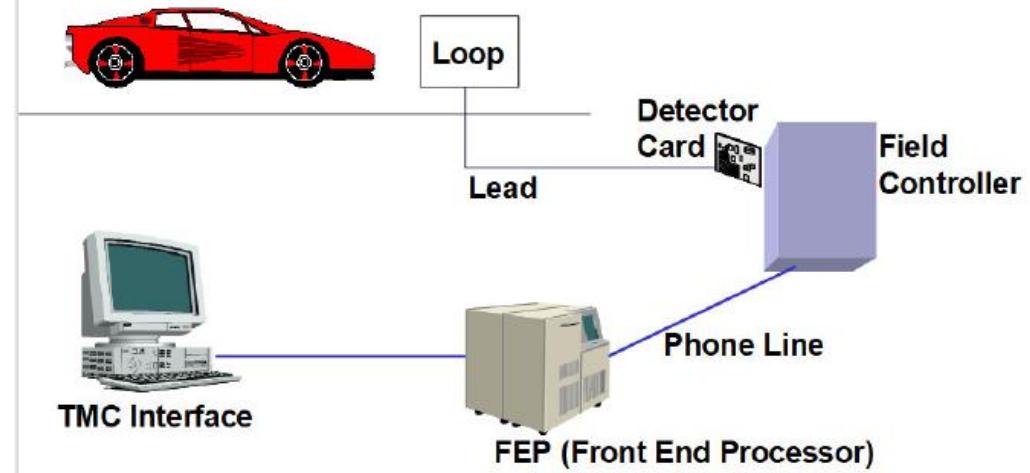
- ✓ 일정 구간내의 교통상태를 연속된 시간동안 관측
- ✓ 교통 카메라
- ✓ 관측데이터



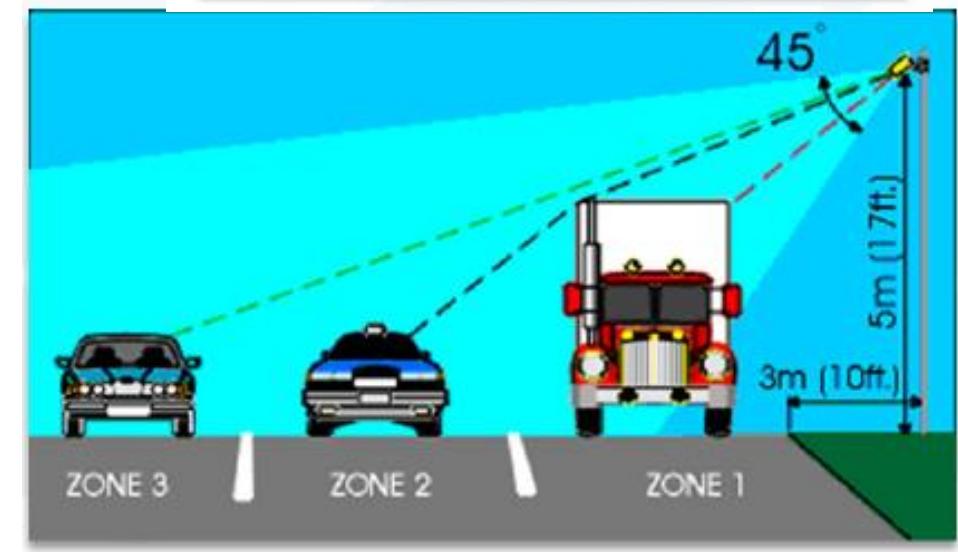
루프 검지기



- 교통량 산정
- 도로 용량 분석
- 차량 통과 속도 감시



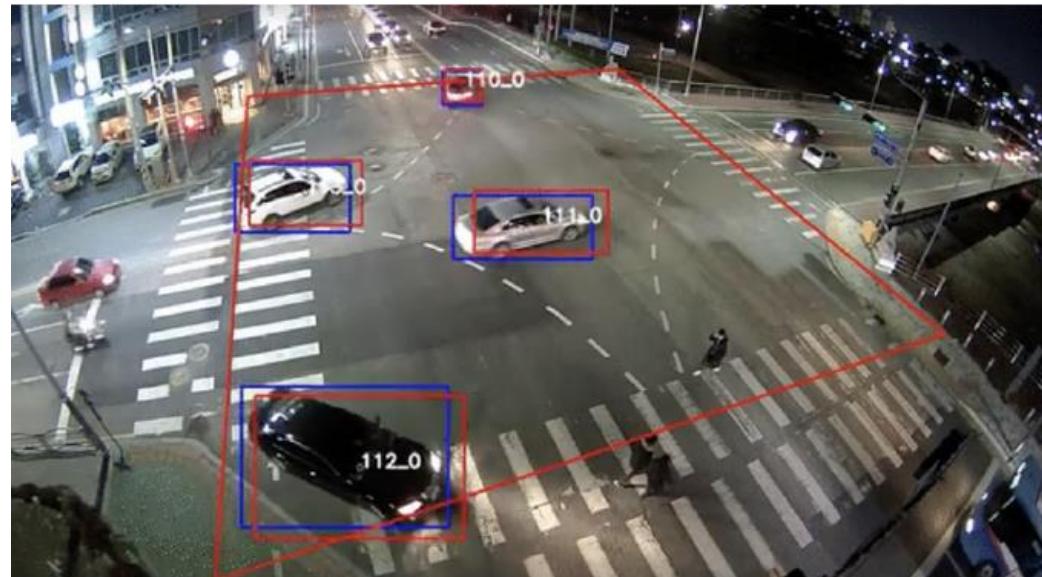
교통 레이다



영상 검지기

RSU(Road Side Unit)

도로를 운행하는 차량에 설치된 단말기와 WAVE 무선통신을 수행 차량 단말기에서 전송하는 각종 정보를 수집 저장하여 센터로 전송하는 기능

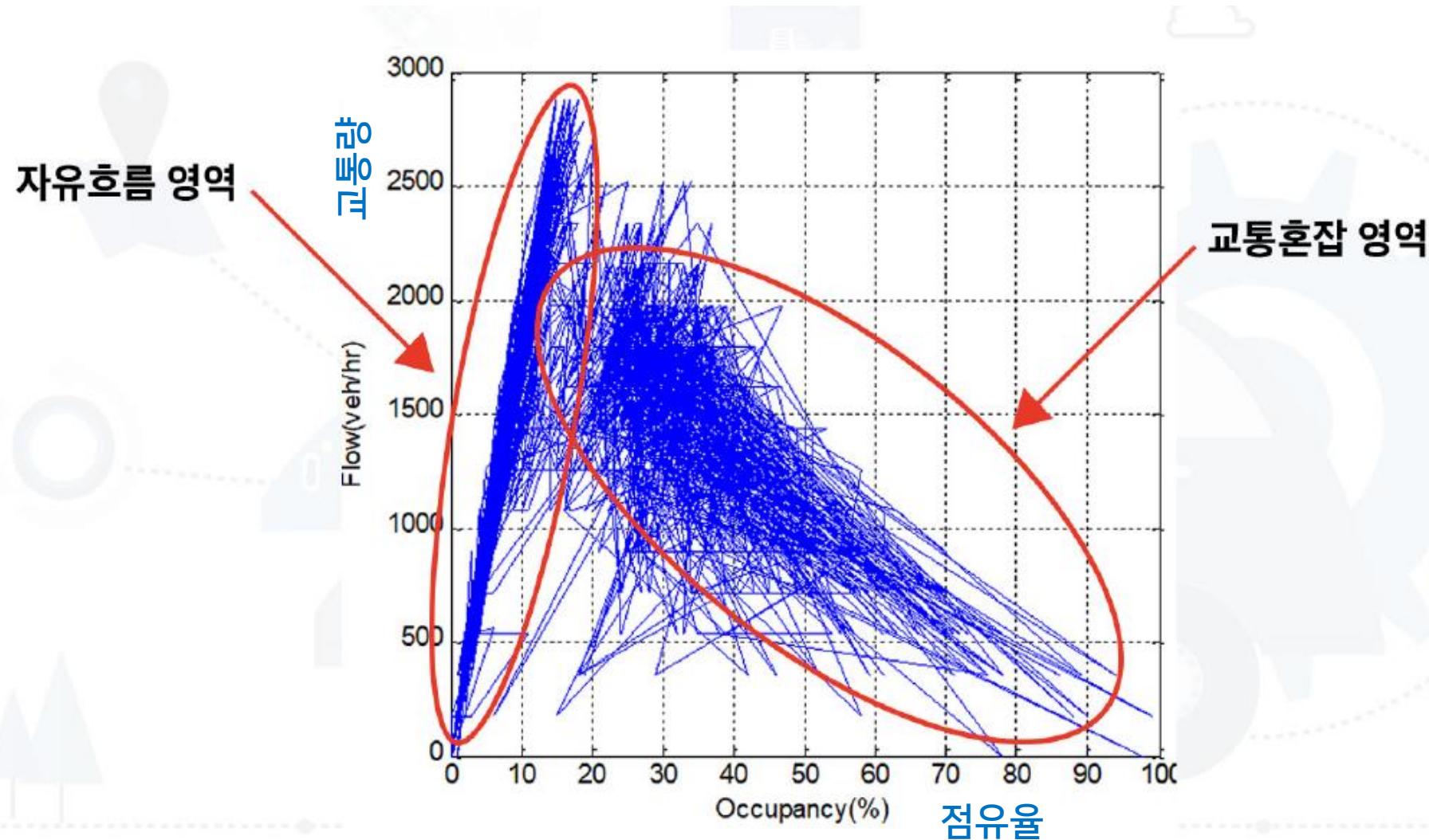


VDS 차량검지기



속도
차량별 교통량
점유률

교통 데이터의 해석



❖ 도시 교통 문제의 원인

- ✓ 교통혼잡은 온실가스, 미세먼지 원인
- ✓ 도시문제는 지속 가능성에 대한 글로벌한 도전

❖ 도시문제 해결을 위한 핵심 키워드는 스마트 시티(Smart city)



❖ 빅데이터 및 활용 과정

- ✓ 분석, 해석의 자동화 수단으로 빅데이터의 축적에 따라 주목받고 투자가 집중
- ✓ 데이터 수집과 저장, 분석, 시각화 단계를 거쳐 최종 제품이나 서비스에 적용



❖ 인공지능

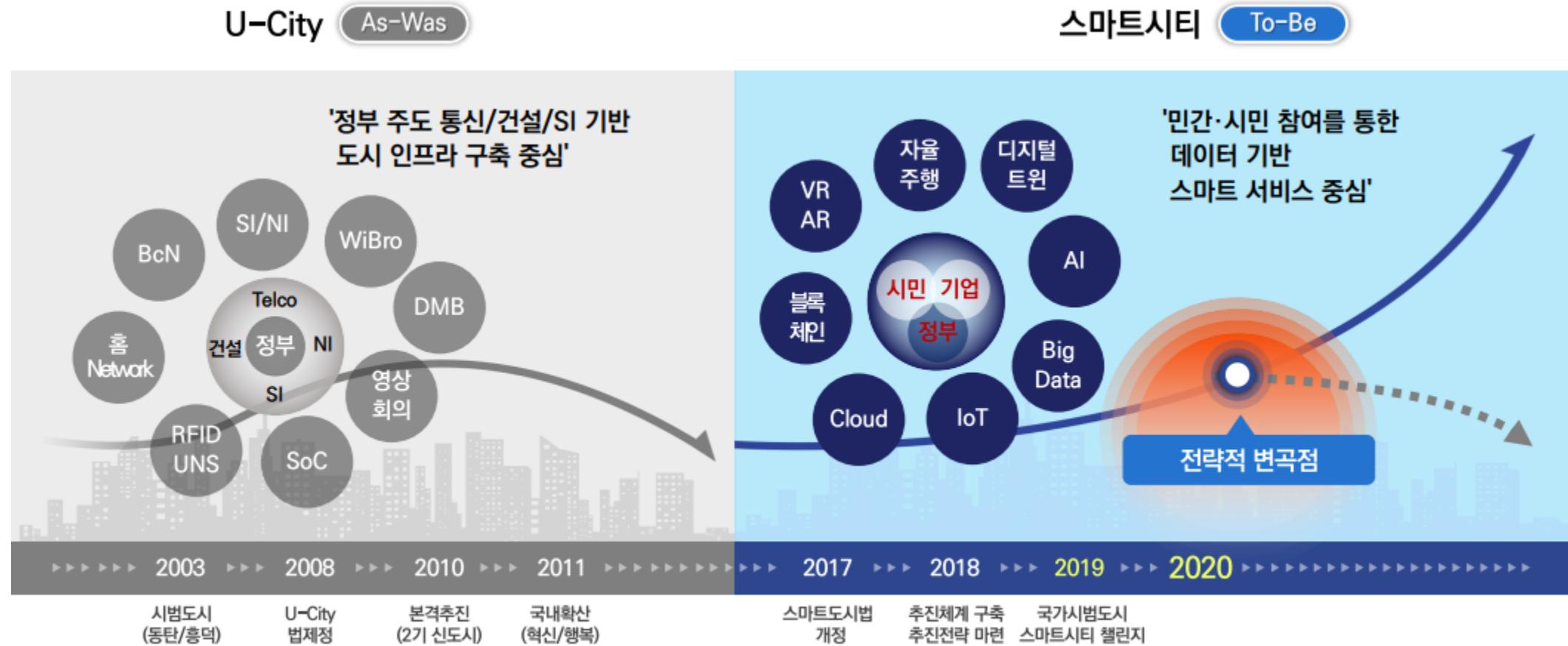
- ✓ AI (학습을 통한 빅데이터 분석)는 인간의 인지, 예측, 분석을 보조하고
- ✓ 장기적으로 스스로 지식, 아이디어의 창출 활동을 함으로써 대리인 역할을 수행

도시 교통분야의 인공지능 활용

- ❖ 기존 CCTV 영상을 분석하여 교통정보 생성 및 정지차량, 역주행차량, 보행자 등 돌발상황을 검지하여 교통상황실 운영자에게 알림



구현지속 가능한 도시운영효율 및 도시 경쟁력 강화를 목표



스마트 시티 실증도시



부산 Eco Delta City

“로봇 등 산업육성으로 혁신생태계가 조성되는 미래 수변도시”

4차산업혁명에 대응하고 산업육성을 위한 5대 클러스터 조성

(서비스) 로봇활용, 배움-일-놀이, 도시관리 지능화, 스마트워터, 제로에너지, 스마트교육&리빙, 헬스, 모빌리티, 안전, 스마트공원



세종 5-1 생활권

“인공지능(AI)기반 도시로 시민의 일상을 바꾸는 스마트시티”

소유차 제한구역 설정, BRT 중심으로 직주 근접(용도혼합) 등

(서비스) 모빌리티, 헬스케어, 교육, 에너지-환경, 거버넌스, 문화-쇼핑, 일자리

2021년 국토부 스마트시티 실증 도시 1단계 예산 지원 사업

- SPC 설립 후 이를 통한 사업 발주 진행될 예정이고 부산 EDC는 한화에너지컨소시엄이, 세종5-1은 LG 컨소시엄이 주관이 되어 SPC 설립 후 진행 예정
- AI 데이터센터
- 디지털트윈
- 스마트IOT
- 사이버보안
- 스마트혁신단지
- 스마트교통
- 스마트에너지
- 스마트헬스케어
- 스마트안전

도시데이터플랫폼은 '오픈 데이터 허브'로 구축



강의 3 (실습) 교통데이터 다운로드

교통 데이터 다운로드

❖ 국가교통데이터 오픈마켓 (로그인 필수로 하세요)

(예제) 도로/대전시청을 선택

<https://www.bigdata-transportation.kr/>

카테고리	도로	63
<input type="checkbox"/> 버스	0	
<input type="checkbox"/> 철도	0	
<input type="checkbox"/> 시설·안전	0	
<input type="checkbox"/> 자율주행	0	
<input type="checkbox"/> 공간	0	
<input type="checkbox"/> 유통	0	
<input type="checkbox"/> 일반·기타	0	

판매업체	대전시청	63
<input type="checkbox"/> 한국도로공사	0	
<input type="checkbox"/> 한국교통연구원	0	
<input type="checkbox"/> 한국철도공사	0	
<input type="checkbox"/> 울산정보산업진흥원	0	
<input type="checkbox"/> 포항테크노파크	0	
<input type="checkbox"/> 진주시청	0	
<input type="checkbox"/> 성남시청	0	
<input checked="" type="checkbox"/> 대전시청	63	
<input type="checkbox"/> 서울특별시	0	
<input type="checkbox"/> 한국부동산원	0	

대전 교차로 데이터
대전 시내 교차로 데이터로서 교차로 ID, 교차로명, 교차로타입, 경위도좌표 데이터
대전시청 | 2022-05-18 ↓ 8 ❤ 0 ⚡ 295 무료

대전시 가변정보표지판(VMS) 소통 패턴 5분전 비교 데이터
대전 시내 가변정보표지판 표출정보 비교 데이터로서 요일, 일자, 시간, VMS, 링크ID, 정보표출범위, 수집건수, 구간통행속도, 패턴통행속도, 속도오차 데이터
대전시청 | 2022-05-18 ↓ 73 ❤ 0 ⚡ 255 무료

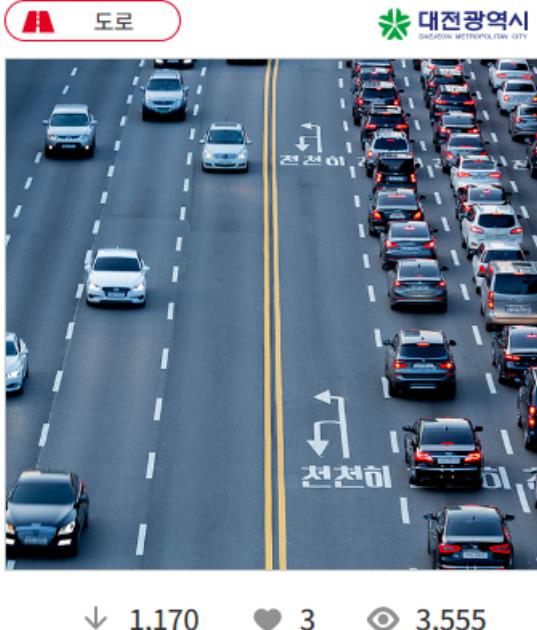
대전시 가변정보표지판(VMS) 소통 패턴 비교 데이터
대전 시내 가변정보표지판 표출정보 비교 데이터로서 요일, 일자, 시간, VMS, 링크ID, 정보표출범위, 수집건수, 구간통행속도, 패턴통행속도, 속도오차 데이터
대전시청 | 2022-05-18 ↓ 133 ❤ 0 ⚡ 248 무료

대전시 VKTVHT 데이터(교통수요예측)
대전 신내 노선별 교통 수요 예측 데이터로서 요일, 일자, 시간, 노선방향, 구간연장, 통행시간, 교통량, VKT, VHT 데이터
대전시청 | 2022-05-18 ↓ 5 ❤ 0 ⚡ 255 무료

대전시 통행 시간 지표 데이터
대전 신내 노선별 통행 시간 지표 데이터로서 요일, 일자, 시간, 자유유속도, 통행시간, 자유유동시간, TTI, PTI, BI 데이터
대전시청 | 2022-05-18 ↓ 66 ❤ 0 ⚡ 280 무료

데이터 상품 상세

[홈](#) > [데이터 상품](#) > [데이터 상품 상세](#)



대전시 VDS 운행 원시 이력 데이터

VDS(Vehicle Detection System) : 차량검지기

도로 상에 약 1km 간격으로 설치되어 실시간으로 교통량, 점유율, 속도, 대기행렬길이, 차량길이 등의 정보를 검지하여 소통 및 돌발상황 등을 감시하는 장치로 도로 환경적 특성에 따라 설치하며 종류는 루프식, 영상식, 레이더식 등이 있다.

대전시 VDS 운행 원시 이력 데이터 (등록일자, 검지기 ID, VDS ID, VDS 구간 ID, 요일 구분, 교통량(소/중/대형), 속도, 점유율, 차두 길이, 차두 시간 등)

기본 이용료

무료

카테고리

도로

제공 기관

대전시청

상품 키워드

#차량검지시스템 #쌍루프원형검지기 #원시이력 #검지기 #대전시
#교통량 #속도 #매팅 #영상검지 #VDS #루프검지기 #레이더

3

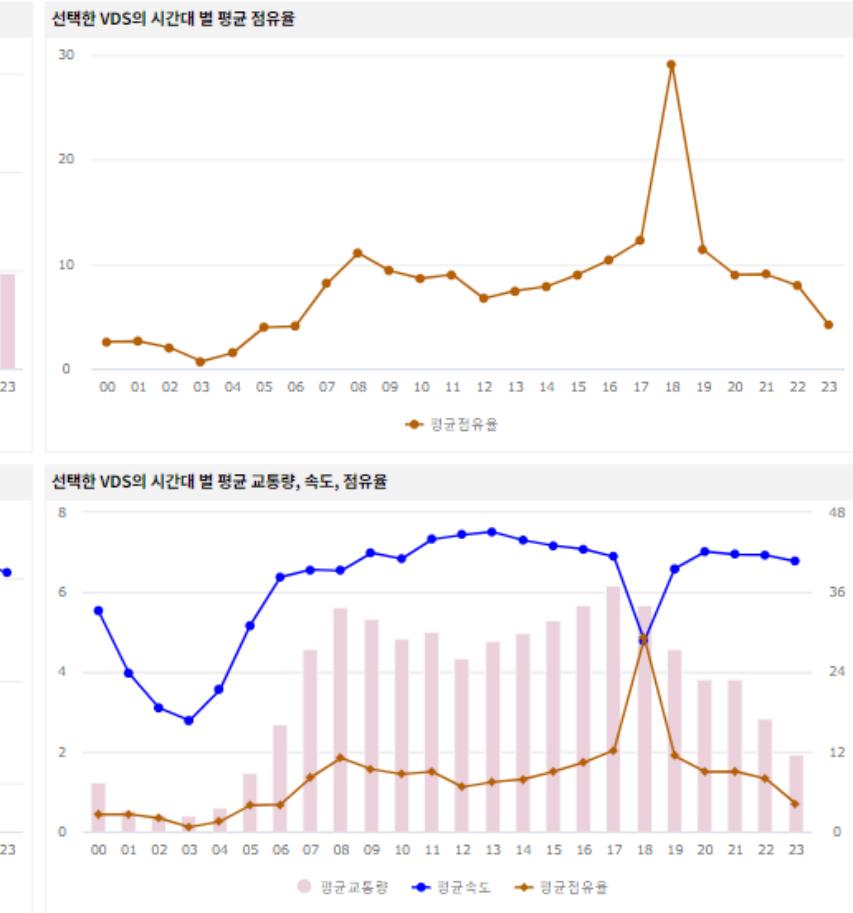
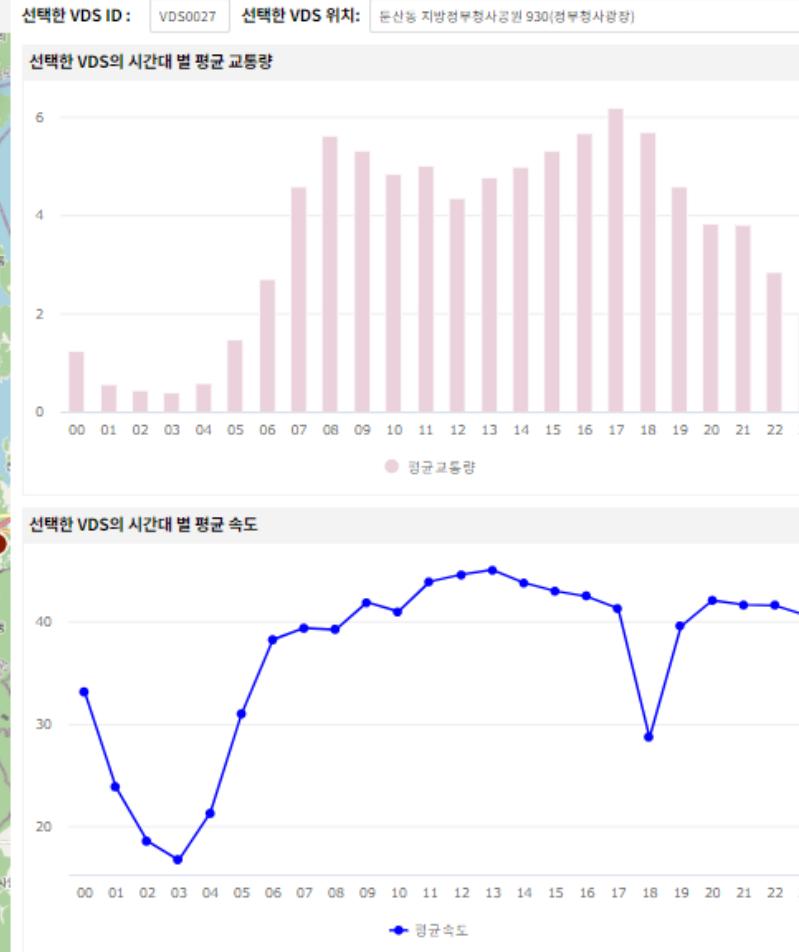
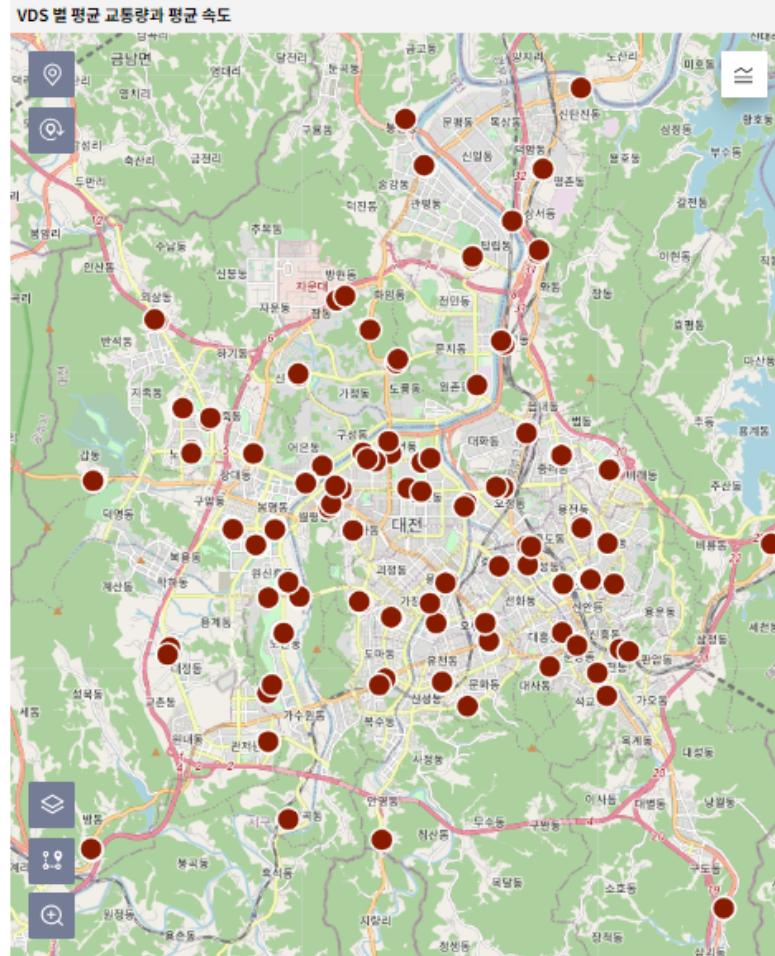
장바구니

데이터 구매하기



대전시 VDS 운행 원시 이력 데이터

대전시 VDS 운행 원시 이력 정보 2020년 9월 8일 기준



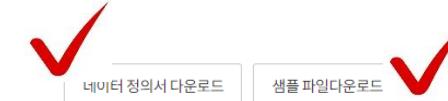
(샘플 데이터) 대전시 VDS 운행 원시 이력 데이터

샘플 데이터

TRVL_SPD	OCCUPY_RATE	AVG_CAR_LEN	AVG_CAR_TM	ETL_TYPE	ETL_DATE
0.0	0.00	0.0	30.0	0602	2020-09-29 03:27:46
0.0	0.00	0.0	30.0	0602	2020-09-29 03:27:46
0.0	100.00	0.0	30.0	0602	2020-09-29 03:27:46
72.0	1.00	70.0	0.0	0602	2020-09-29 03:27:46
0.0	0.00	0.0	0.0	0602	2020-09-29 03:27:46

네이버 정의서 다운로드

샘플 파일다운로드



코드	내용	비고
1501	일요일	
1502	월요일	
1503	화요일	
1504	수요일	
1505	목요일	
1506	금요일	
1507	토요일	
1508	특수일	
1509	전체요일	
1599	미분류	

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
REG_YMDHMS DETR_ID VDS_ID VDS_SECTN_ID DAY_CLS DETR_FAIL_YN TR_VOL SM_TR_VOL MD_TR_VOL LG_TR_VOL TRVL_SPD OCCUPY_RATE AVG_CAR_LEN AVG_CAR_TM ETL_TYPE ETL_DATE																	
2020-09-28 01:53:04 DET005503 VDS0055 VL1850000060 2 0 0 0 0 0.0 0.0 30.0 0602 2020-09-29 03:27:46																	
2020-09-28 01:53:04 DET005505 VDS0055 VL1850000060 2 0 0 0 0 0.0 0.0 30.0 0602 2020-09-29 03:27:46																	
2020-09-28 01:53:04 DET005702 VDS0057 VL1830000063 2 0 0 0 0 0.0 100.00 0.0 30.0 0602 2020-09-29 03:27:46																	
2020-09-28 01:53:04 DET006401 VDS0064 VL1880000070 2 0 1 1 0 0 72.0 1.00 70.0 0.0 0602 2020-09-29 03:27:46																	
2020-09-28 01:53:04 DET006403 VDS0064 VL1880000070 2 0 0 0 0 0.0 0.0 0.0 0.0 0602 2020-09-29 03:27:46																	
2020-09-28 01:53:04 DET007802 VDS0078 VL1880000087 2 0 1 1 0 0 42.0 1.44 46.0 30.0 0602 2020-09-29 03:27:46																	
2020-09-28 01:53:04 DET003401 VDS0034 VL1850000037 2 0 0 0 0 0.0 0.0 0.0 0.0 30.0 0602 2020-09-29 03:27:46																	
2020-09-28 01:53:04 DET003403 VDS0034 VL1850000037 2 0 0 0 0 0.0 0.0 0.0 0.0 30.0 0602 2020-09-29 03:27:46																	
2020-09-28 01:53:04 DET004002 VDS0040 VL1830000043 2 0 0 0 0 0.0 0.0 0.0 0.0 30.0 0602 2020-09-29 03:27:46																	
2020-09-28 01:53:04 DET004004 VDS0040 VL1830000044 2 0 0 0 0 0.0 0.0 0.0 0.0 30.0 0602 2020-09-29 03:27:46																	

실습 데이터 예제: 대전시 VDS 운행 원시 이력 데이터

네이버셋		대전시 VDS 운행 원시 이력 데이터		상품코드
데이터 상품 정보	대전시 VDS 운행 원시 이력 데이터		대전시청	연락처
관리기관	대전시청		이메일	
홈페이지	https://www.daejeon.go.kr/		차량이동	접근권한
카테고리	차량이동			
데이터 상품 정보	대전시 VDS 운행 원시 이력 데이터 (등록일자, 검지기 ID, VDS ID, VDS 구간 ID, 요일 구분, 교통량 (소/중/대형), 속도, 점유율, 차두 길이, 차두 시간 등)		원시이력, 대전시, VDS, 교통량, 속도, 검지기, 차량감지시스템, 루프검지기, 쌍루프원형검지기, 영상감지	
키워드	원시이력, 대전시, VDS, 교통량, 속도, 검지기, 차량감지시스템, 루프검지기, 쌍루프원형검지기, 영상감지			
생성주기	1월		데이터 기본 이용료	
테이블		VDS 운행 원시자료 이력		테이블명(영문)
테이블명(한글)	컬럼명(한글)	컬럼명(영문)	컬럼 설명	데이터 타입
등록 일자	등록 일자	REG_YMDHMS	등록 일자	DATE
검지기 ID	검지기 ID	DETR_ID	검지기 ID	CHAR(9)
VDS ID	VDS ID	VDS_ID	VDS ID	CHAR(7)
VDS 구간 ID	VDS 구간 ID	VDS_SECTN_ID	VDS 구간 ID	CHAR(12)
요일_구분	요일_구분	DAY_CLS	요일_구분	CHAR(1)
검지 장애	검지 장애	DETR_FAIL_YN	검지 장애	CHAR(1)
교통량	교통량	TR_VOL	교통량	NUMBER(5)
소형 교통량	소형 교통량	SM_TR_VOL	소형 교통량	NUMBER(5)
중형 교통량	중형 교통량	MD_TR_VOL	중형 교통량	NUMBER(5)
대형 교통량	대형 교통량	LG_TR_VOL	대형 교통량	NUMBER(5)
속도	속도	TRVL_SPD	속도	NUMBER(4)
점유율	점유율	OCCUPY_RATE	점유율	NUMBER(5)
차두 길이	차두 길이	AVG_CAR_LEN	차두 길이	NUMBER(4)
차두 시간	차두 시간	AVG_CAR_TM	차두 시간	NUMBER(4)
ETL_유형	ETL_유형	ETL_TYPE	ETL_유형	VARCHAR2(8)
ETL_DATE	ETL_DATE	ETL_DATE	ETL_DATE	DATE

1 페이지

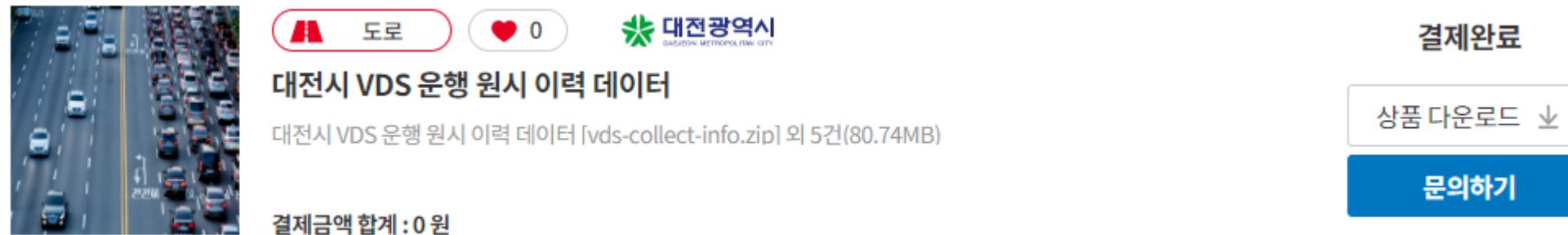
❖ 대전시 VDS 운행 원시 이력 데이터 :

- ✓ 등록일자, 검지기 ID, VDS ID, VDS 구간 ID,
- ✓ 요일 구분,
- ✓ 교통량 (소/중/대형),
- ✓ 속도,
- ✓ 점유율,
- ✓ 차두 길이, 차두 시간 등

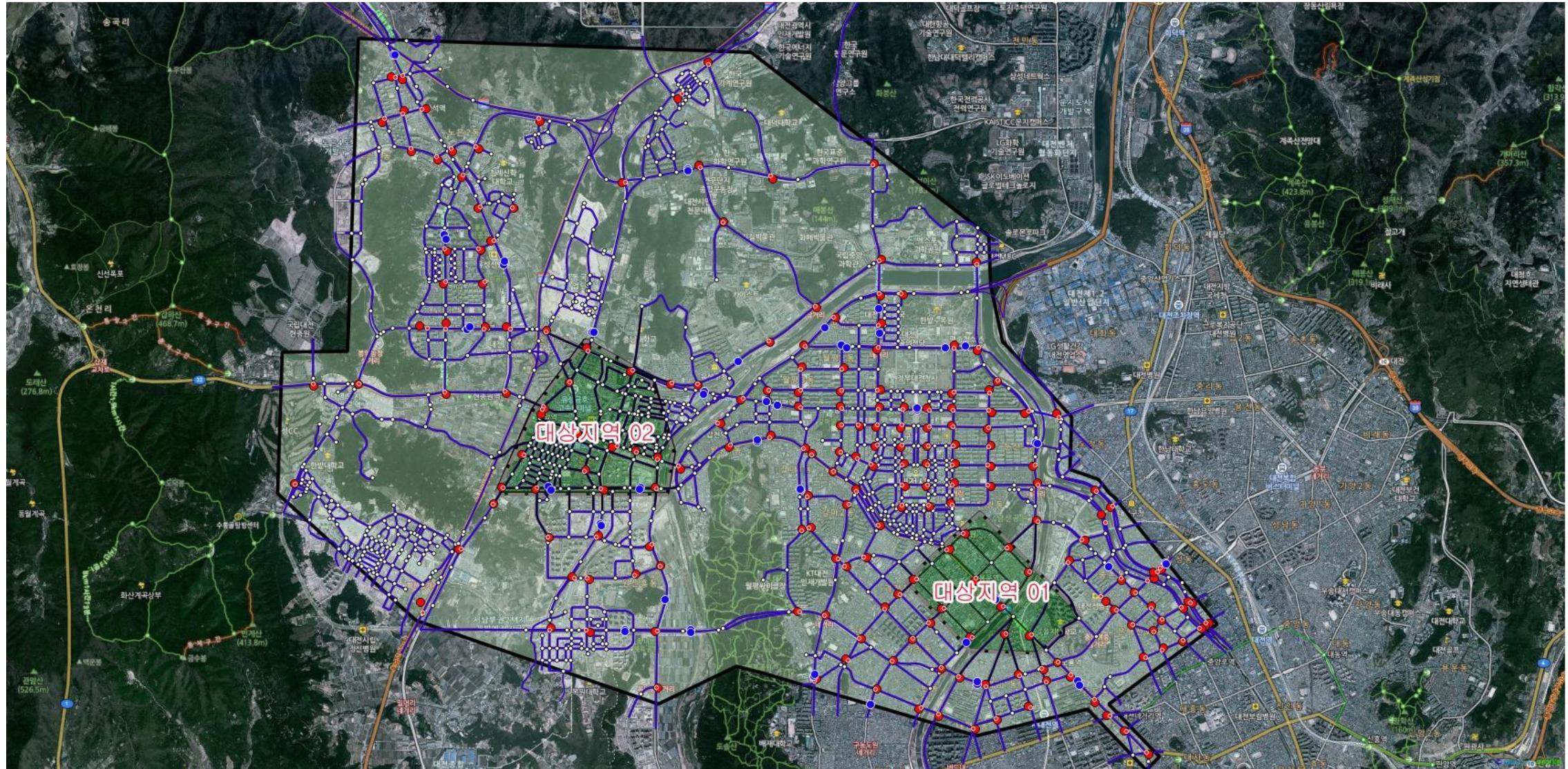
A	B	C	D	E	F	G	H
Date	ToVol	SmVol	MeVol	LaVol	Speed	Occ.Rate	
2017-04-02 0:00	43	34	9	0	50.3	1.9	
2017-04-02 0:05	45	32	13	0	58.9	1.84	
2017-04-02 0:10	46	34	12	0	50.6	1.87	
2017-04-02 0:15	45	36	9	0	50.9	1.72	
2017-04-02 0:20	27	13	13	1	62.2	1.12	
2017-04-02 0:25	30	15	15	0	63.5	1.17	
2017-04-02 0:30	27	14	12	1	73.6	0.99	
2017-04-02 0:35	40	25	15	0	52.7	1.76	

- ❖ **대전교통정보센터** <http://traffic.daejeon.go.kr/mainFront/atmsMain.do>
- ❖ **VDS(Vehicle Detection System) : 차량검지기**

- ✓ 도로 상에 약 1km 간격으로 설치되어 실시간으로 교통량, 점유율, 속도, 대기행렬길이, 차량길이 등의 정보를 검지하여 소통 및 돌발상황 등을 감시하는 장치로 도로 환경적 특성에 따라 설치하며 종류는 루프식, 영상식, 레이더식 등이 있다
- ✓ 대전시 VDS 운행 원시 이력 데이터 (등록일자, 검지기 ID, VDS ID, VDS 구간 ID, 요일 구분, 교통량 (소/중/대형), 속도, 점유율, 차두 길이, 차두 시간 등)



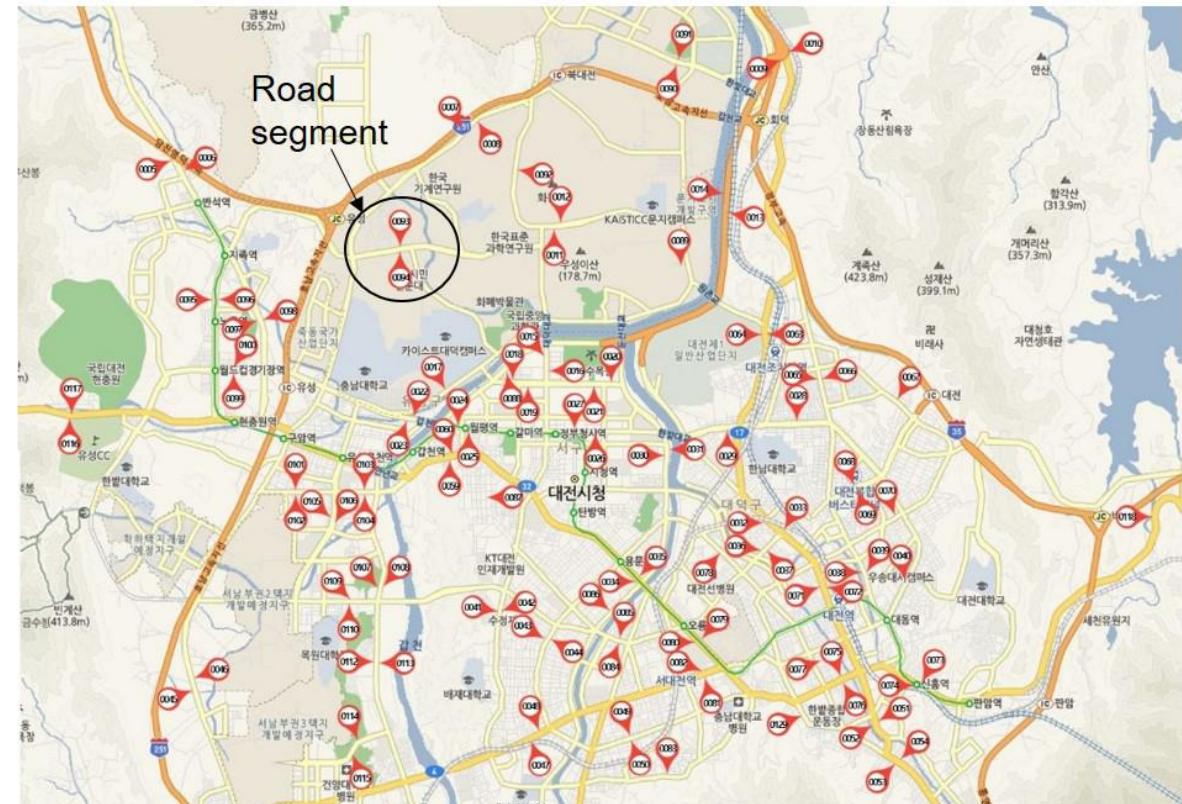
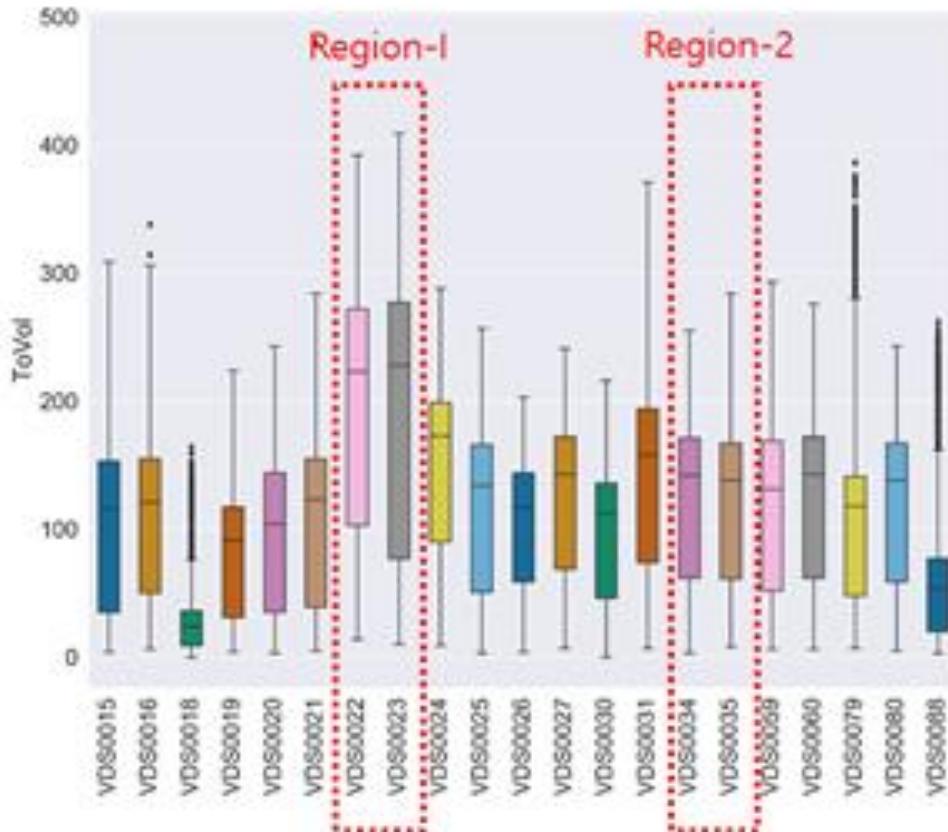
VDS 데이터를 활용한 교통 흐름 분석 사례(대전시 테스트베드)



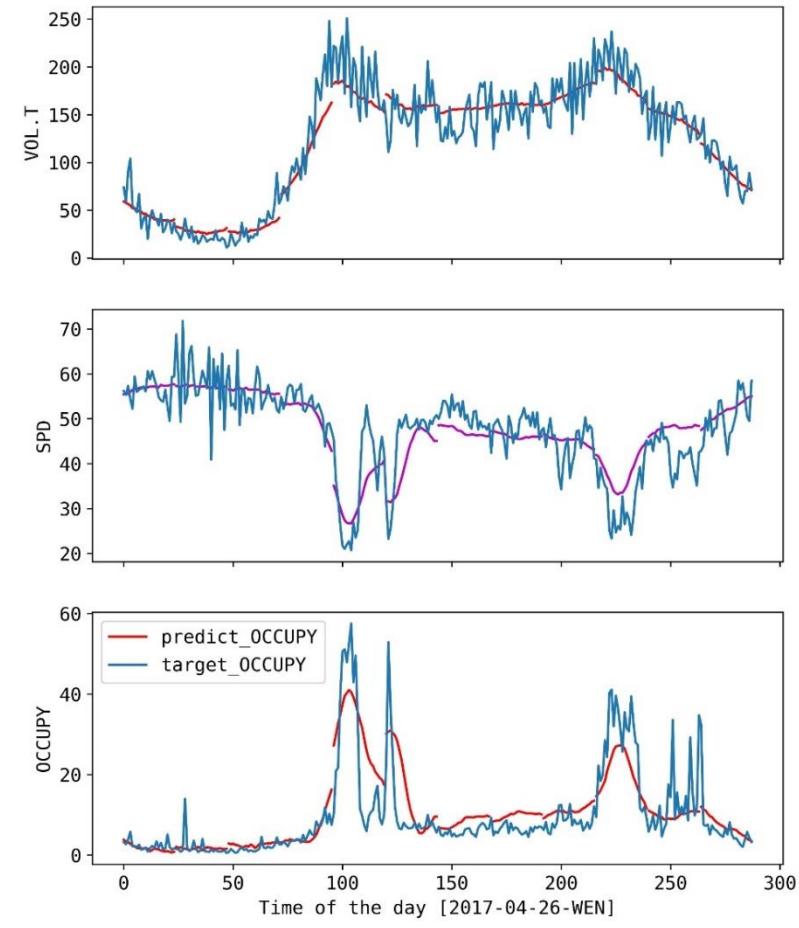
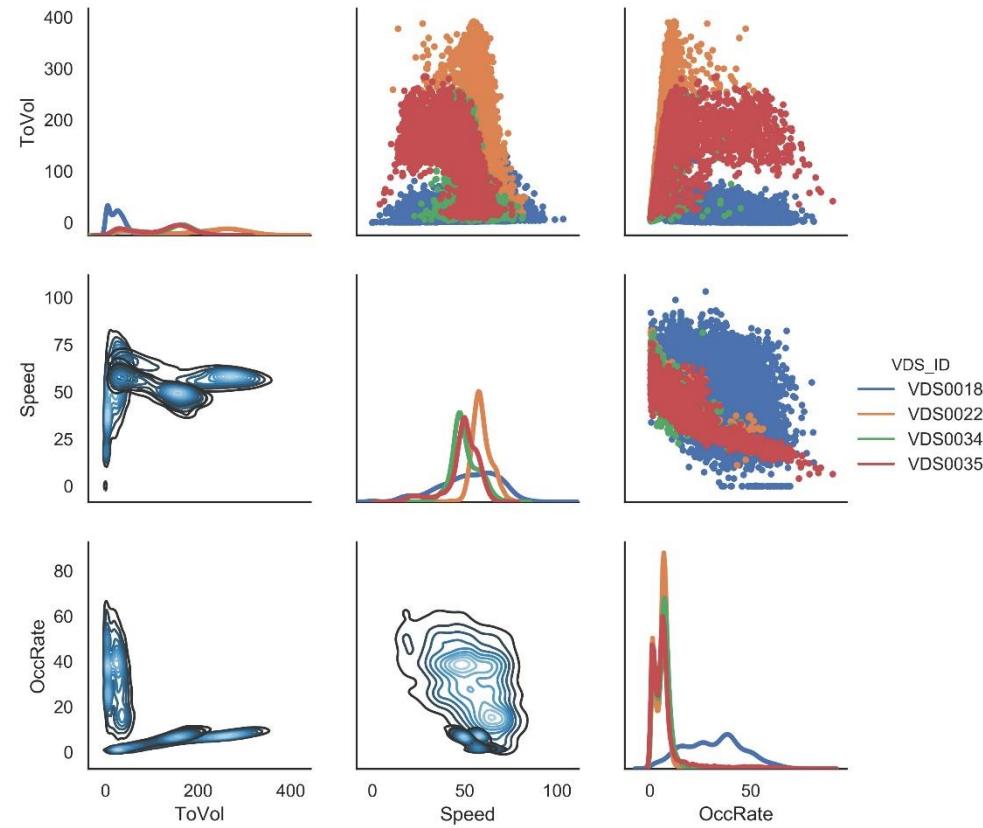
교통 장기 예측

- ❖ 대전시 VDS 교통데이터를 이용한 교통혼잡 지역에서 교통흐름 예측

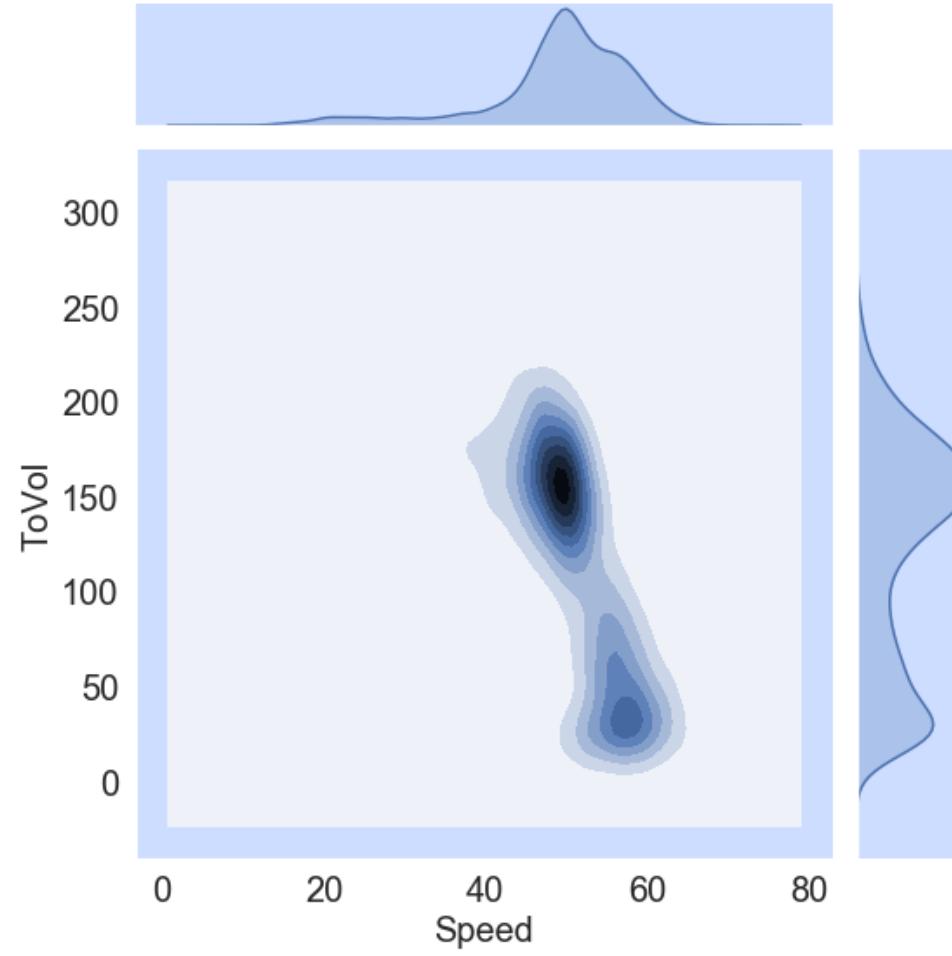
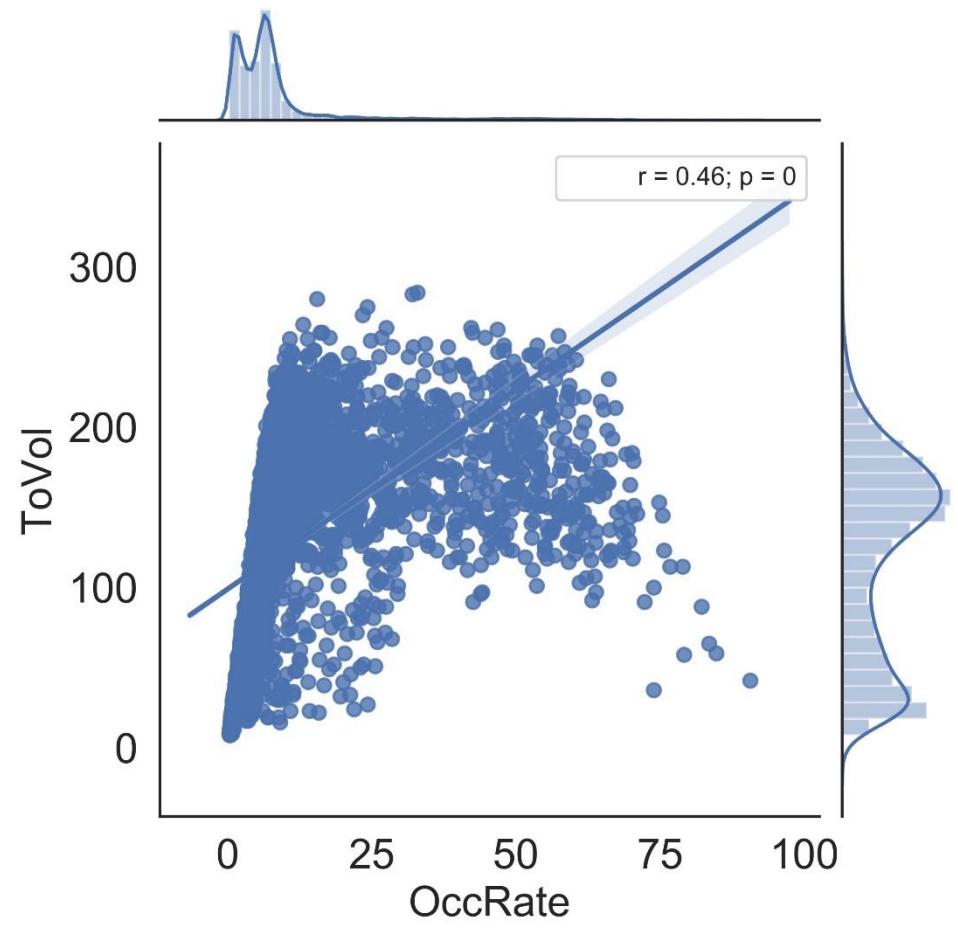
- ✓ VDS(속도, 교통량, 점유율)을 장단기메모리(LSTM)을 이용한 예측 모델



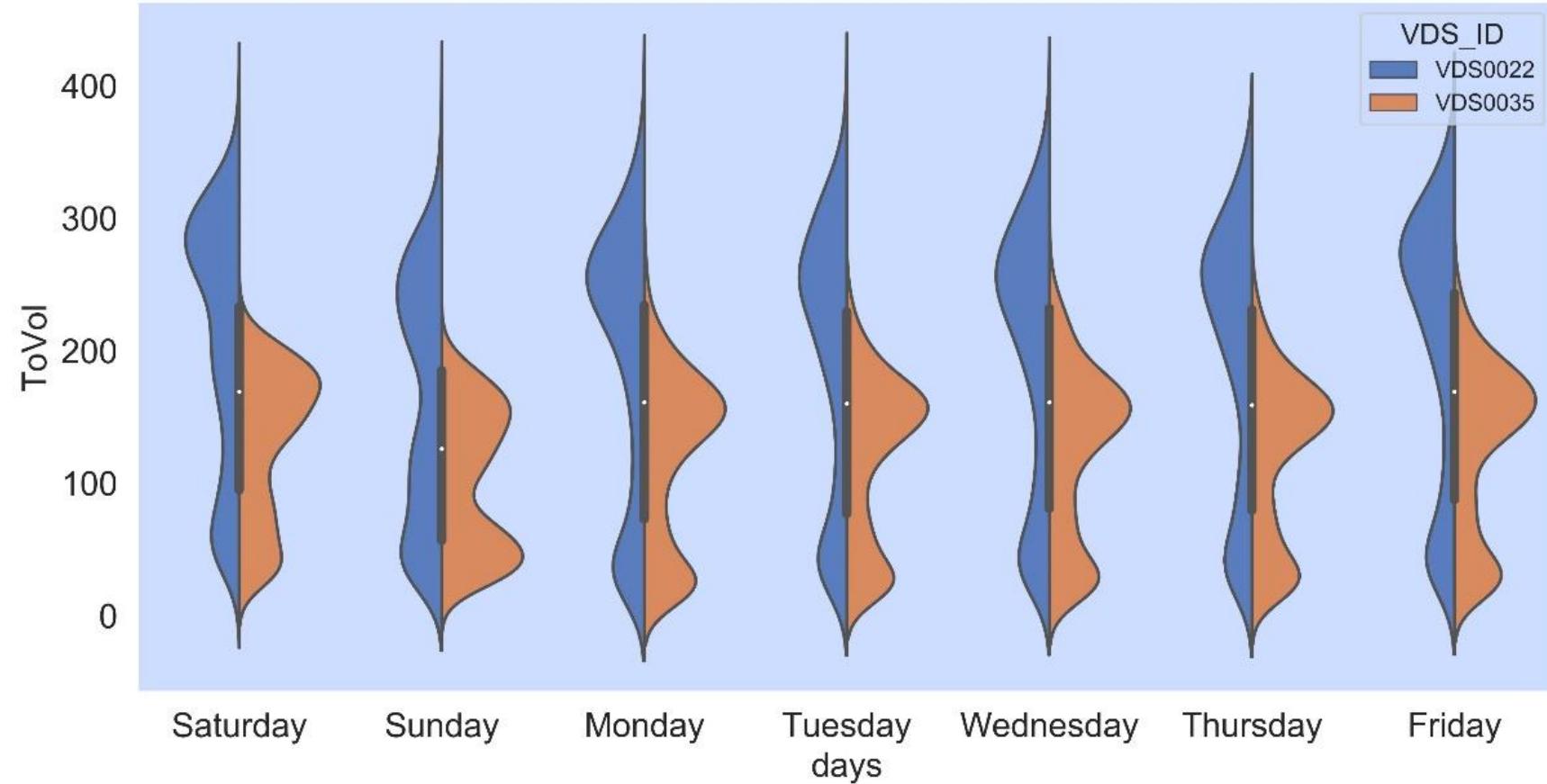
상관관계 및 RNN 예측



점유률과 교통량 상관관계 구하기



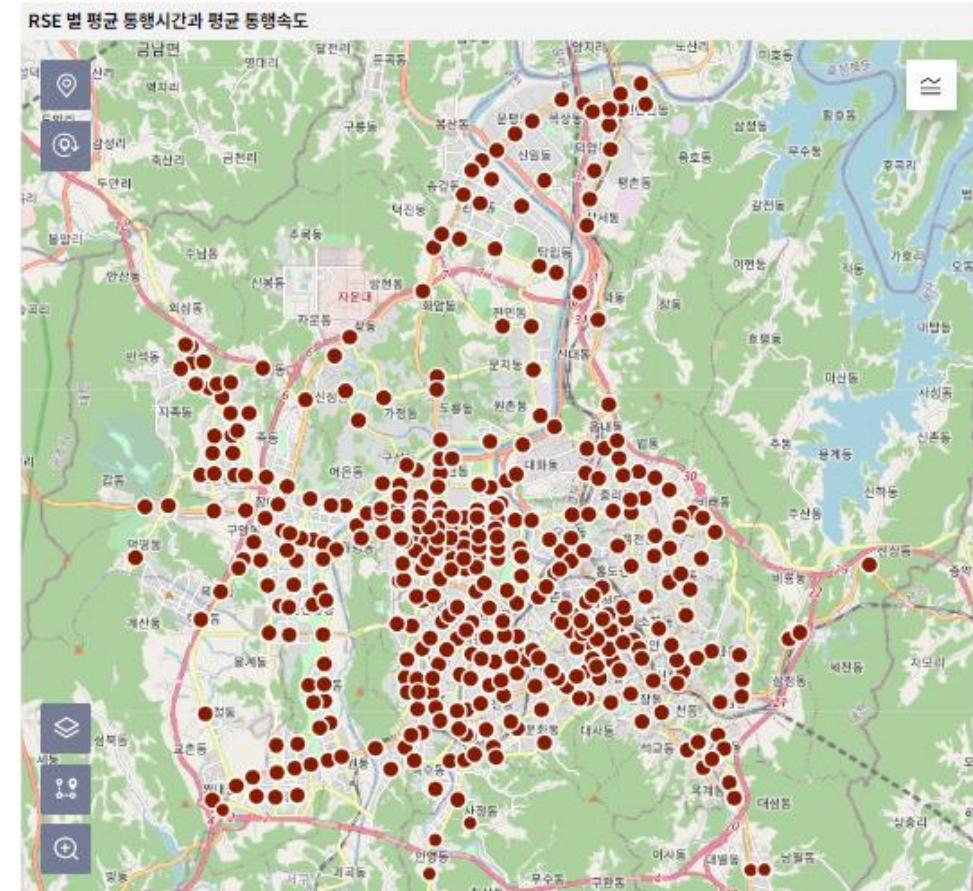
요일별 교통량 분석



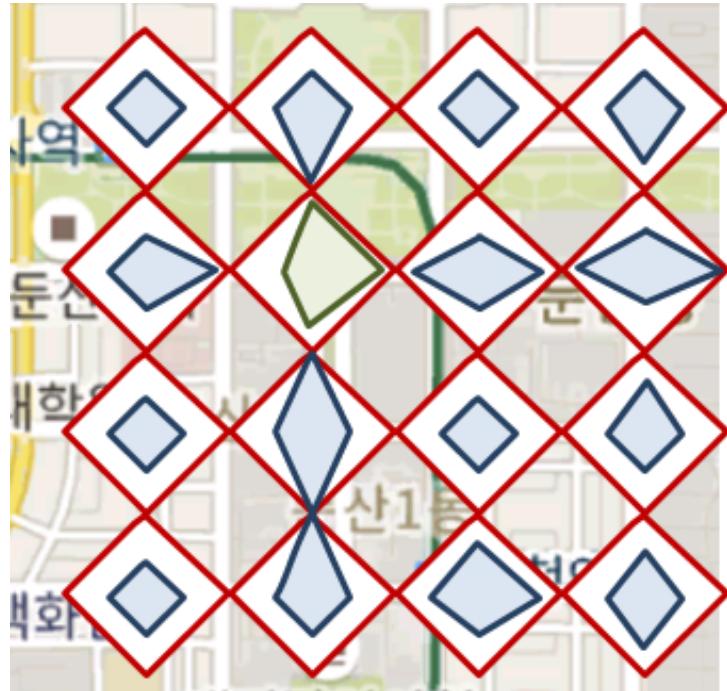
❖ 첨단 교통량 관리 시스템

- ✓ (ATMS, Advanced Transportation Management System)
 - 실시간 교통상황 정보를 토대로 도로와 같은 교통시설의 이용률을 극대화하기 위한 교통 관제체계
- ✓ RSE란
 - 교차로나 차로변에 위치한 CCTV 지주 등에 설치되어 차량의 운행정보를 수집하는 기기
- ✓ 대전시 노변장치(RSE) 정보 5분 단위 데이터
 - 일자 ID, 시 ID, 분 ID, 링크 ID, RSE 통행 속도, RSE 통행 시간 등
- ✓ RSE 위치 및 각 RSE 별 평균 통행시간/통행 속도

대전시 ATMS RSE 소통 이력 정보 2020년 9월 8일 기준

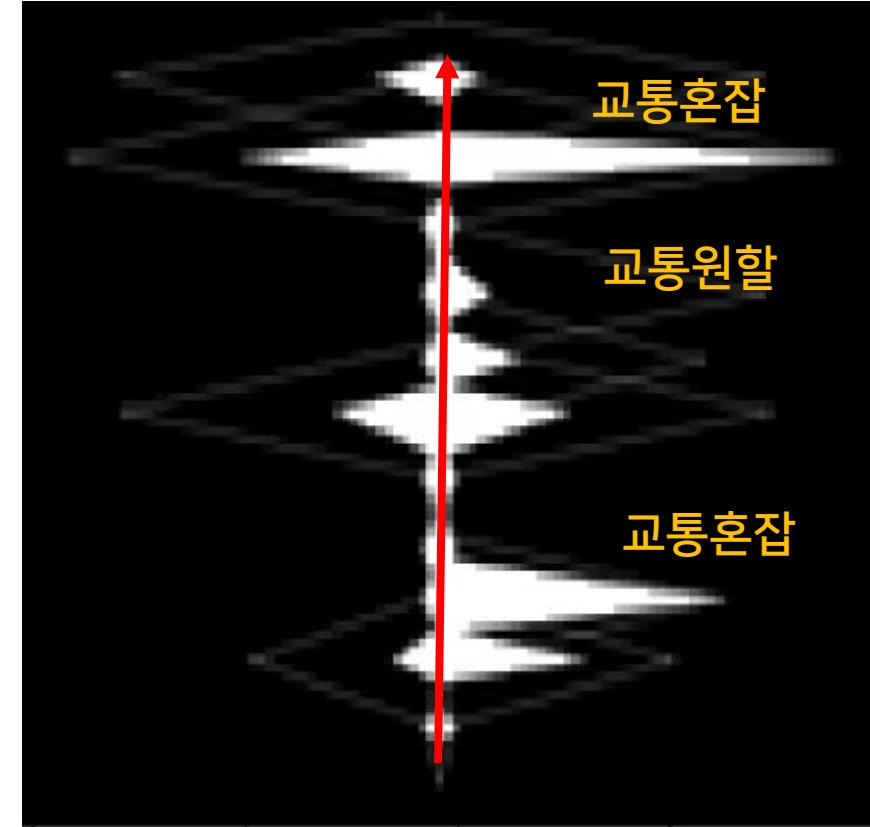


- ❖ 대전시 대덕대로(갤러리아 백화점~과학관) RSE 데이터 활용
- ❖ RSE 데이터 기반 교통 혼잡 데이터 분석으로 AI CNN 모데러 적용 가능함



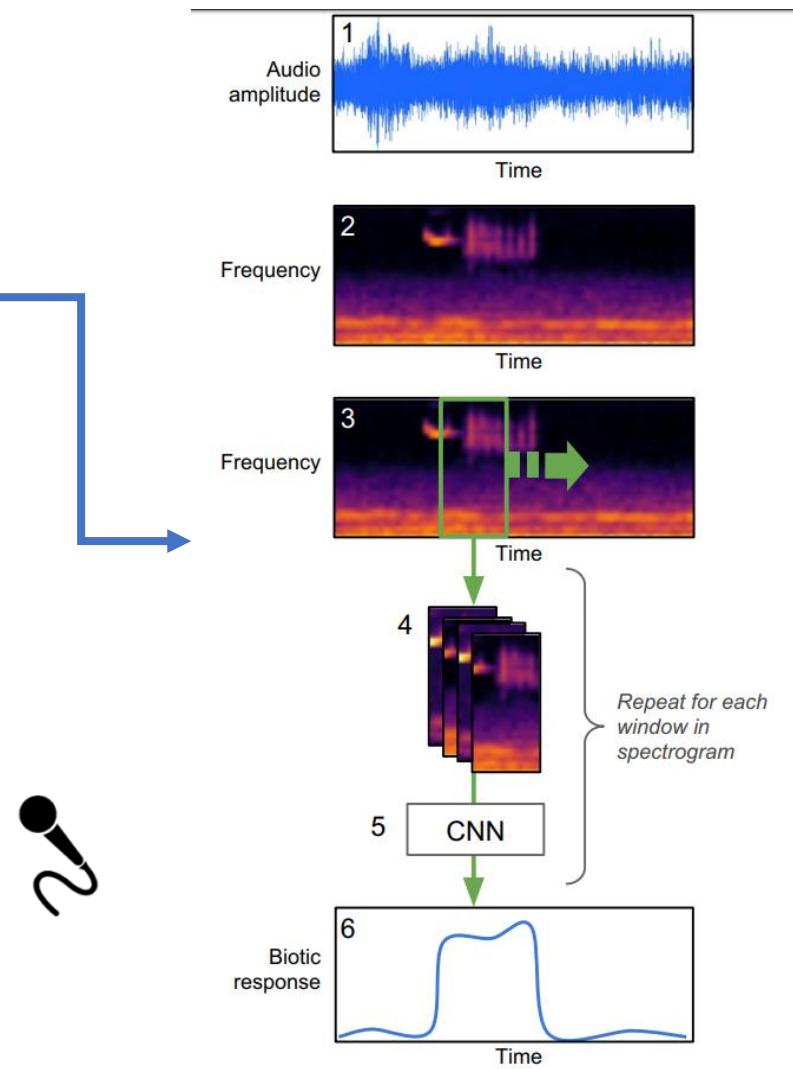
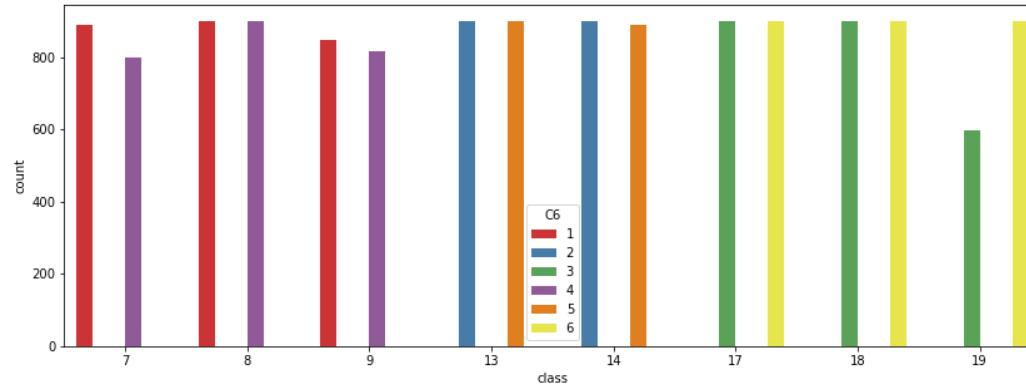
대전시 중앙과학관 방면

대전시 갤러기아 백화점

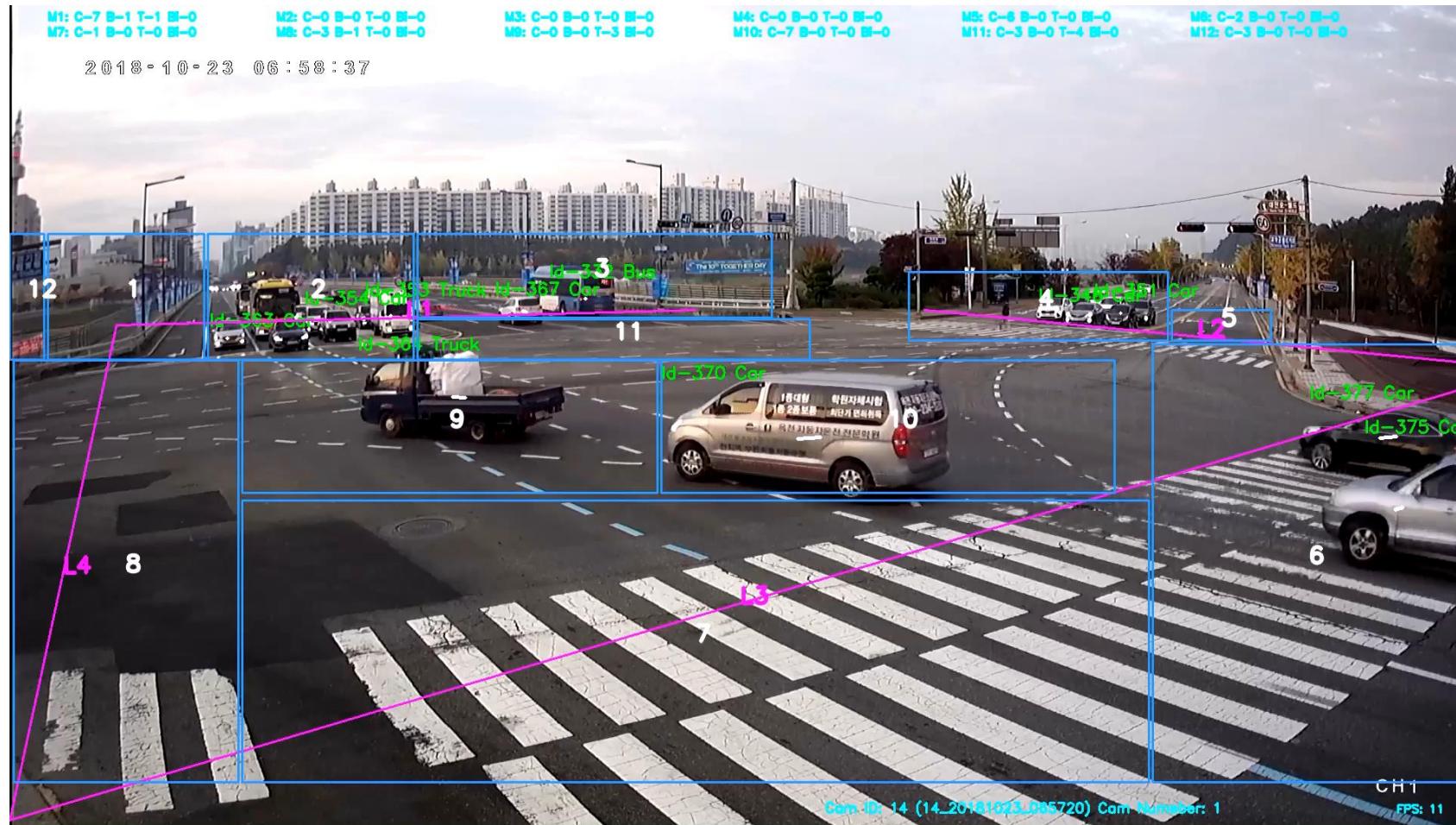


(교통 소음 데이터 구축) 대전시 둔산대로 소음 영상

- 소음 빅데이터 자체 수집 : DJ_TrafficSound14K (대덕대로)



❖ 세계 최고 AI 컨퍼런스인 CVPR20의 'AI City Challenge' 경진대회 참가 및 (7위, KISTI)



강의 4 (실습: 해보기) VDS 데이터 전처리

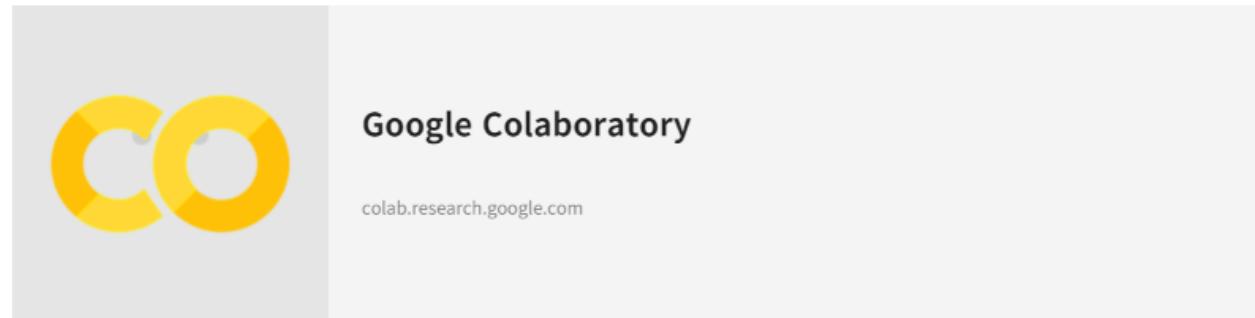
(lab-01-vds-preprocessing.ipynb)

❖ Colaboratory는 완전히 클라우드에서 실행되는 무료 Jupyter 노트 환경

- ✓ 즉, 브라우저에서 클라우드 환경을 이용해 Python 코드를 실행할 수 있으며,
- ✓ 무료로 GPU와 TPU 등의 컴퓨팅 자원을 함께 사용할 수 있다.
- ✓ Colaboratory를 사용하면 브라우저를 통해 무료로 코드를 작성 및 실행하고
- ✓ 분석을 저장 및 공유하며, 강력한 컴퓨팅 리소스를 이용할 수 있습니다.

❖ Google Colaboratory 접속하기

<https://colab.research.google.com/>

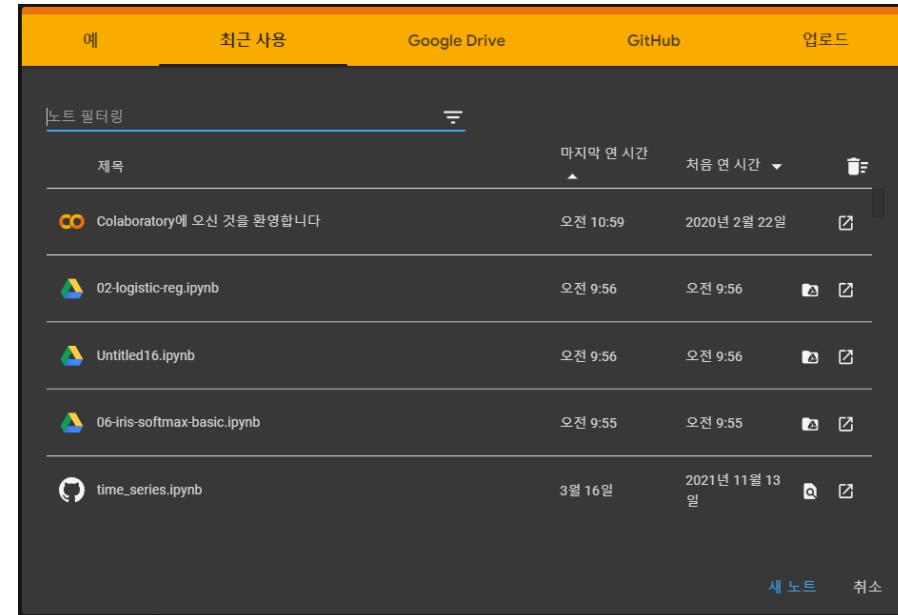


❖ Google Colab 스펙

- ✓ CPU : 제온
- ✓ Memory : 13GB
- ✓ HDD : 320GB
- ✓ GPU : NVIDIA Tesla K80

❖ 파일 생성/접근 방법

- ✓ 개인 구글 개정으로 접근
- ✓ <https://colab.research.google.com> 접속
- ✓ GOOGLE 새드라이브 PYTHON 노트탭 선택
- ✓ 실습은python3로 진행할 예정.



파일 이름 변경



Code cell, Text cell

- .ipynb 파일은 code cell과 text cell로 구성
- 각 셀 하단에 마우스를 대거나, 화면 좌상단 버튼으로 셀 추가 가능
- 셀 선택(마우스) 후 셀 우상단 삭제버튼으로 셀 삭제 가능

Code cell

- 일반적인 파이썬 코딩 방식과 동일
- 각 셀은 한번에 실행할 단위를 뜻함
- 실행 이후에도 메모리는 유지되어 다른 셀 실행 시 영향을 줌
 - 런타임 다시 시작 시 초기화



```
[1] # Code Cell!
a = 1
b = 2
print(a+b)

# Ctrl+Enter 로 해당 코드 셀 실행

3

# 각 셀은 한번에 실행할 단위를 뜻함
# 실행 이후에도 메모리는 그대로 유지되어 다른 셀의 실행에 영향을 줌
a += 3
b -= 1
print(a+b)

5
```

- 상단 메뉴의 런타임
 - 실행 중인 셀 중단
 - 런타임 다시 시작



■ Text cell

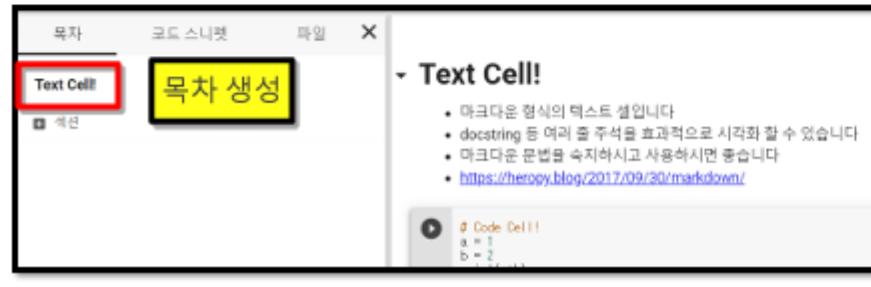
- 여러 줄 주석의 효과적인 시각화
- 마크다운(Markdown) 문법
- 자동 목차 생성



The screenshot shows two views of a Jupyter Notebook cell. On the left, the code cell contains the following Markdown:

```
# Text Cell!
• 마크다운 형식의 텍스트 셀입니다
• docstring 등 여러 줄 주석을 효과적으로 시각화 할 수 있습니다
• 마크다운 문법을 숙지하시고 사용하시면 좋습니다
• https://heropy.blog/2017/09/30/markdown/
```

On the right, the rendered output shows the same content as a Text Cell, with the title "Text Cell!" and the same bulleted list.



The screenshot shows the Jupyter Notebook interface with the TOC sidebar open. The sidebar has tabs for "목차", "코드 스니펫", and "파일". The "목차" tab is selected and highlighted with a red box. A yellow box highlights the "목차 생성" button. Below the sidebar, a code cell titled "Text Cell!" is visible, containing the same Markdown content as the previous screenshot.

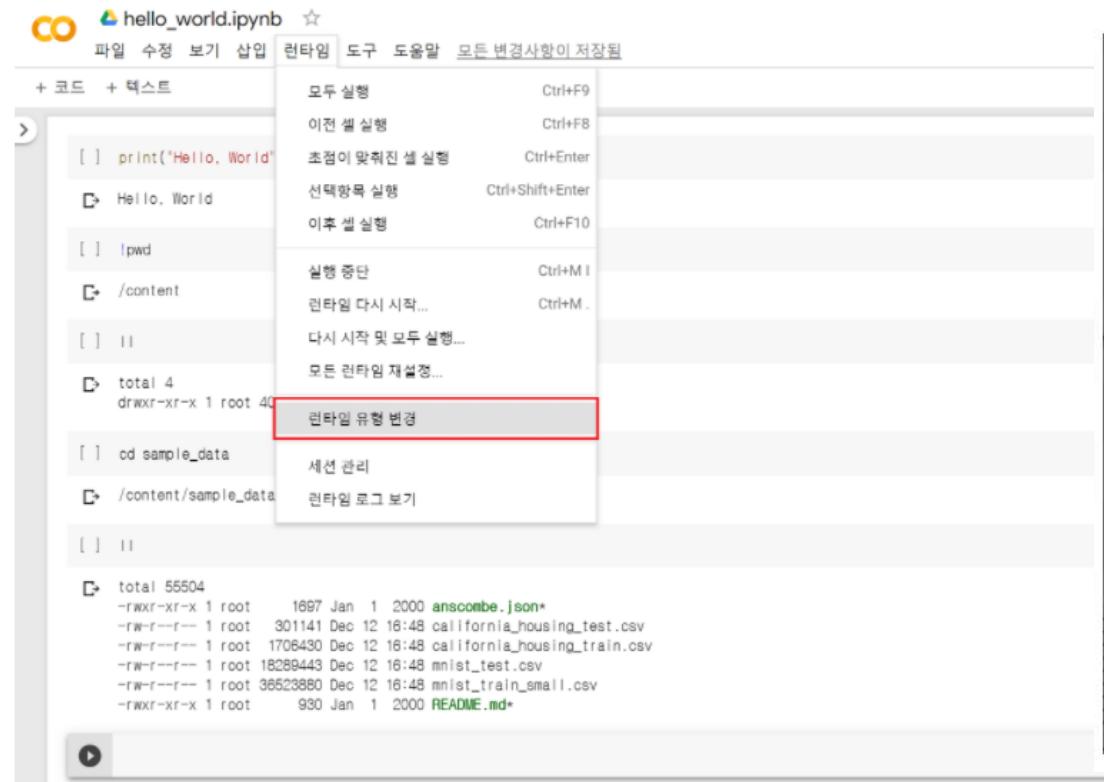
11

런타임 유형 GPU, TPU 설정하기

무료답지 않게 엄청난 서비스가 무료입니다.

GPU, TPU 등을 지원하기 때문에 요즘 핫한 Tensorflow등을 동작할 수 있습니다.

Colab의 GPU 정보 확인 방법:
!nvidia-smi



hello_world.ipynb

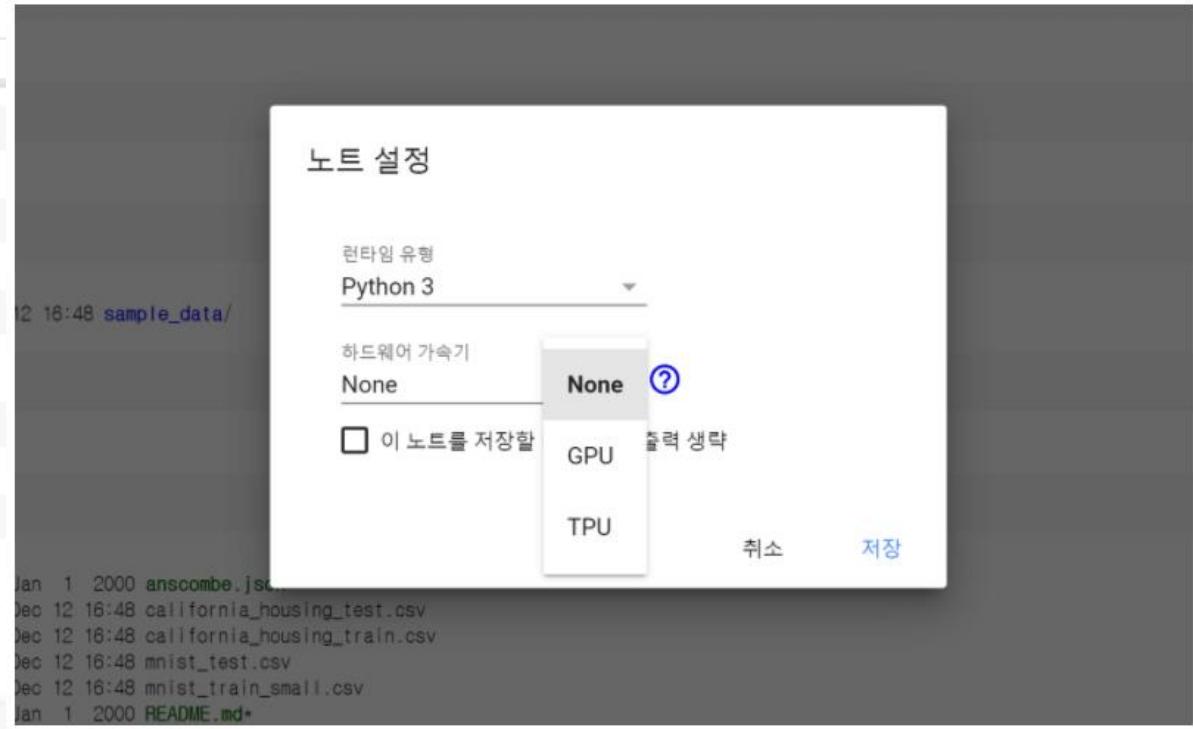
파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨

+ 코드 + 텍스트

```
[ ] print('Hello, World')
Hello, World
[ ] pwd
/content
[ ] !ls
Hello, World
[ ] total 4
drwxr-xr-x 1 root 4096 Dec 12 16:48 sample_data
[ ] cd sample_data
[ ] ls
```

런타임 유형 변경

모두 실행 Ctrl+F9
이전 셀 실행 Ctrl+F8
초점이 맞춰진 셀 실행 Ctrl+Enter
선택항목 실행 Ctrl+Shift+Enter
이후 셀 실행 Ctrl+F10
실행 중단 Ctrl+M I
런타임 다시 시작... Ctrl+M .
다시 시작 및 모두 실행... Ctrl+M ..
모든 런타임 재설정...



❖ VDS 데이터 전처리 과정을 이해한다.

- ✓ VDS 데이터 특성 파악
 - Pandas를 이용하여 데이터를 읽어본다.
 - 나중에 DataFrame 기능을 사용해서 데이터를 다룬다.
- ✓ 속도, 교통량, 점유율, 일시, 요일 등

❖ VDS 데이터 특성(Feature)를 이해한다.

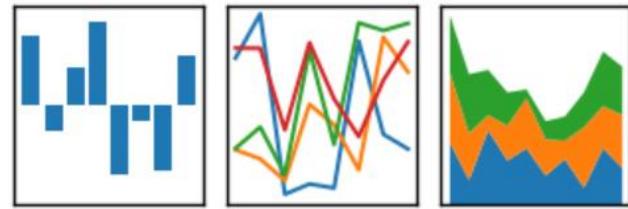
- ✓ 시간대별, 요일별 특성을 파악한다.
 - Matplotlib을 통해서 가시화 한다.
- ✓ 특성 사이의 상관 관계를 고려해 본다.
 - Seaborn을 통해서 가시화 한다.

❖ 지도학습 문제 중에서 분류(Classification)을 다룬다.

- ✓ 라벨링을 시도해 보자.

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



Python Pandas



DataFrame Basics

❖ 판다스 데이터 읽기

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

```
In [2]: df = pd.read_csv('./daejeon_vds16.csv')
```

```
In [3]: df.head()
```

Out[3]:

	Date	ToVol	SmVol	MeVol	LaVol	Speed	Occ.Rate
0	2017-04-02 0:00	43	34	9	0	50.3	1.90
1	2017-04-02 0:05	45	32	13	0	58.9	1.84
2	2017-04-02 0:10	46	34	12	0	50.6	1.87
3	2017-04-02 0:15	45	36	9	0	50.9	1.72
4	2017-04-02 0:20	27	13	13	1	62.2	1.12

```
In [4]: df.info()
```

판다스로 데이터 전처리

```
In [5]: df.set_index('Date', inplace=True)
```

```
In [6]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 8064 entries, 2017-04-02 0:00 to 2017-04-29 23:55
Data columns (total 6 columns):
 #   Column   Non-Null Count  Dtype  
--- 
 0   ToVol    8064 non-null   int64  
 1   SmVol    8064 non-null   int64  
 2   MeVol    8064 non-null   int64  
 3   LaVol    8064 non-null   int64  
 4   Speed    8064 non-null   float64 
 5   Occ.Rate 8064 non-null   float64 
dtypes: float64(2), int64(4)
memory usage: 441.0+ KB
```

```
In [14]: df.head(2)
```

```
Out[14]:
```

	Date	ToVol	SmVol	MeVol	LaVol	Speed	Occ.Rate
2017-04-02 0:00		43	34	9	0	50.3	1.90
2017-04-02 0:05		45	32	13	0	58.9	1.84

```
In [8]: df.describe()
```

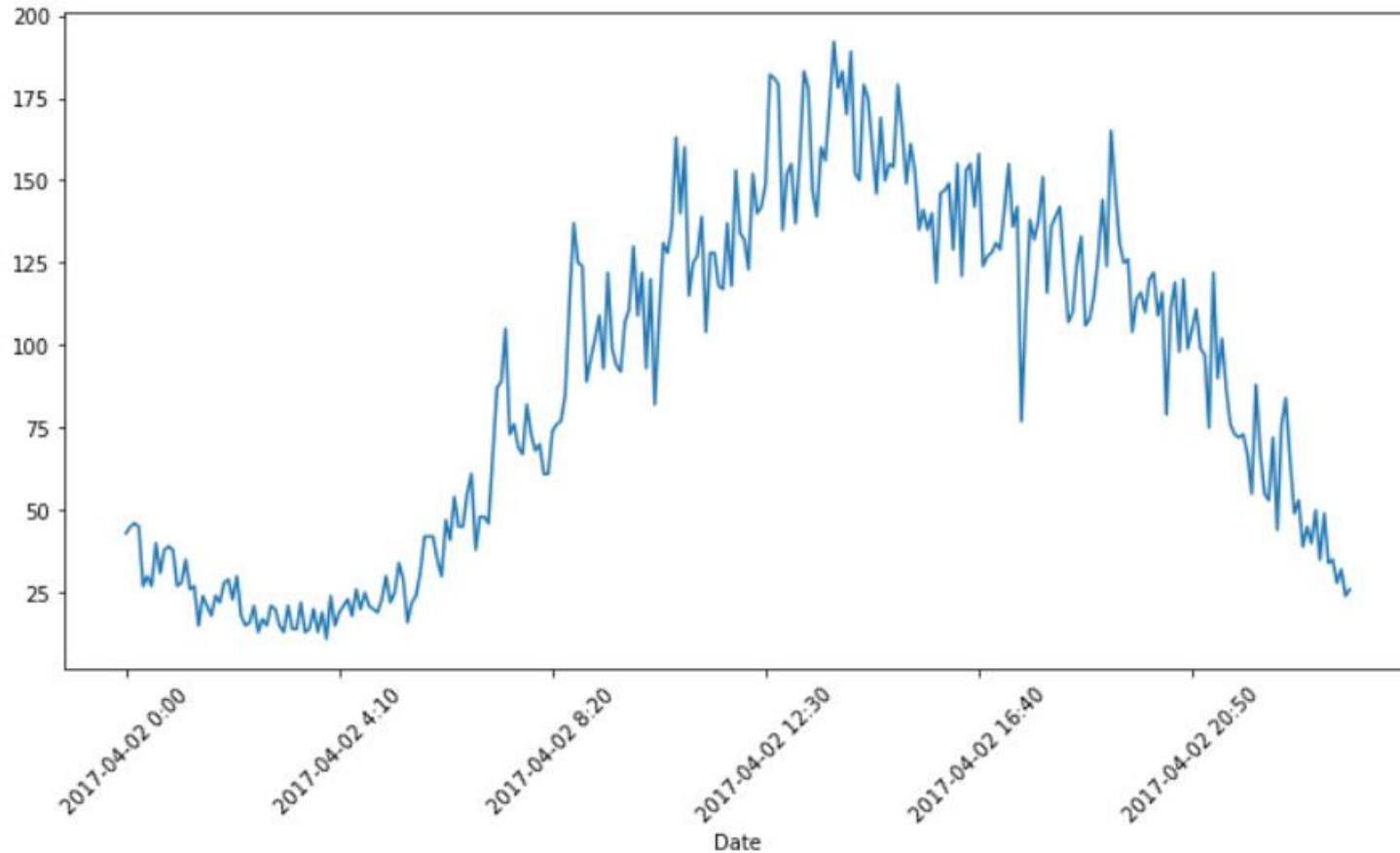
```
Out[8]:
```

	ToVol	SmVol	MeVol	LaVol	Speed	Occ.Rate
count	8064.000000	8064.000000	8064.000000	8064.000000	8064.000000	8064.000000
mean	110.459945	79.353299	29.948537	1.158110	49.327431	6.166941
std	63.954451	46.802106	19.081136	1.530192	7.921856	6.739946
min	6.000000	2.000000	0.000000	0.000000	9.100000	0.230000
25%	50.000000	35.000000	13.000000	0.000000	44.900000	2.140000
50%	122.000000	87.000000	29.000000	1.000000	48.500000	5.550000
75%	155.000000	111.000000	44.000000	2.000000	54.200000	7.290000
max	338.000000	250.000000	145.000000	16.000000	87.800000	82.100000

판다스로 데이터 전처리

```
In [16]: df['ToVol'][:288].plot(rot=45,figsize=(12,6))
```

```
Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x2994144df08>
```

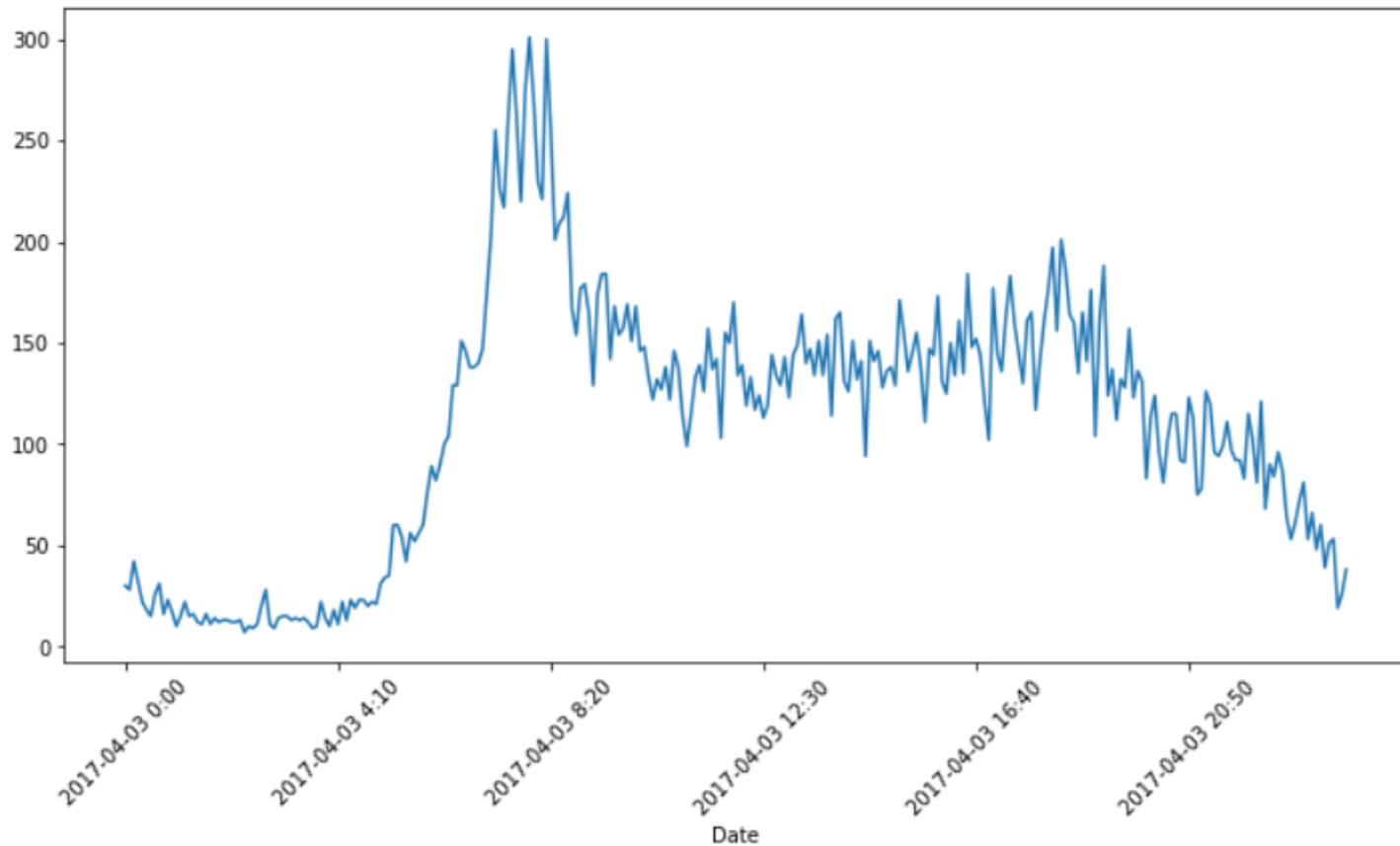


판다스로 데이터 전처리

```
In [17]: ## 2017년 4월 3일은 월요일.
```

```
df['ToVol'][288:576].plot(rot=45,figsize=(12,6))
```

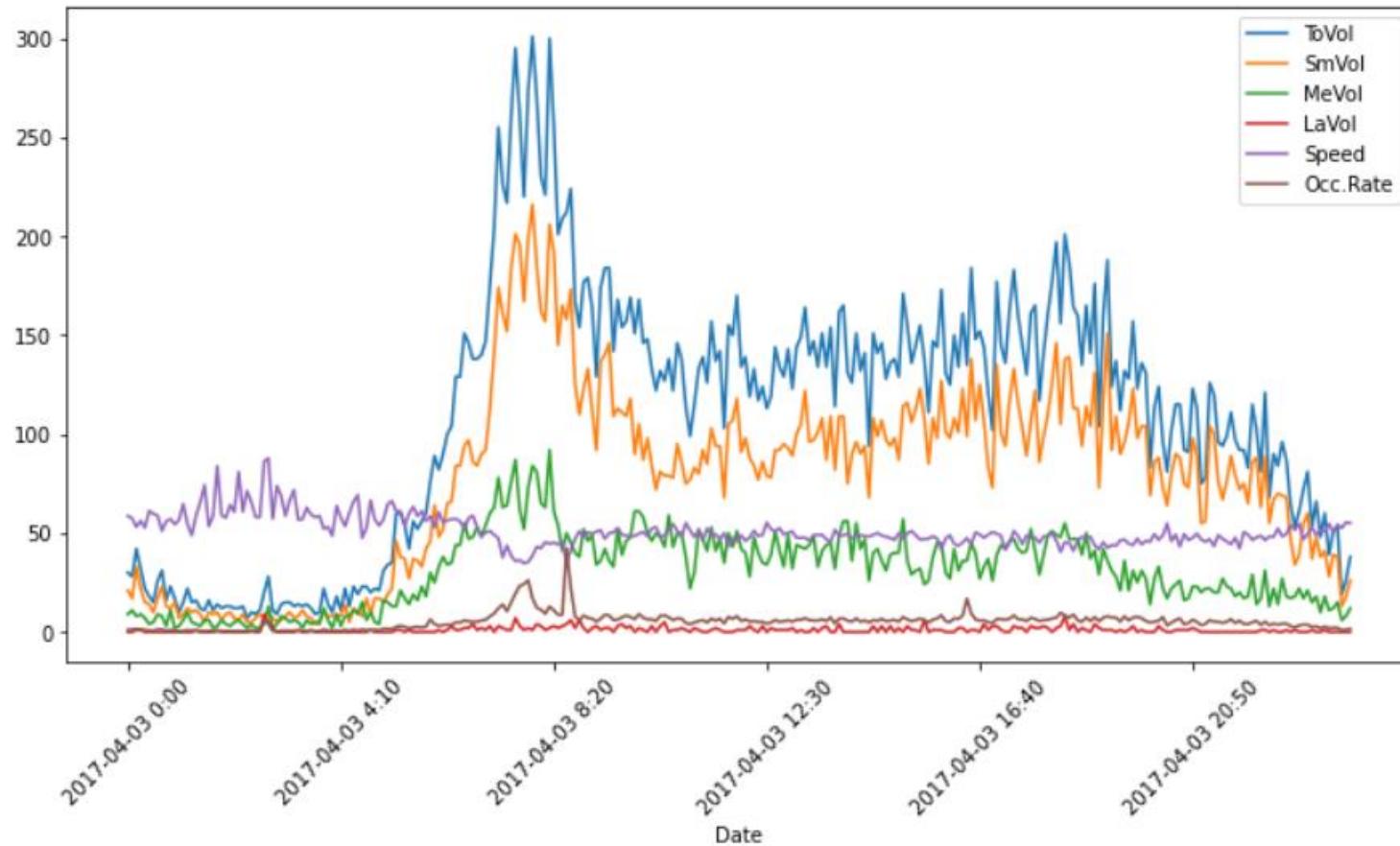
```
Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0x299414bb9c8>
```



연습문제: 점유률 데이터

```
In [32]: df[288:576].plot(rot=45,figsize=(12,6))
```

```
Out[32]: <matplotlib.axes._subplots.AxesSubplot at 0x29943e10908>
```

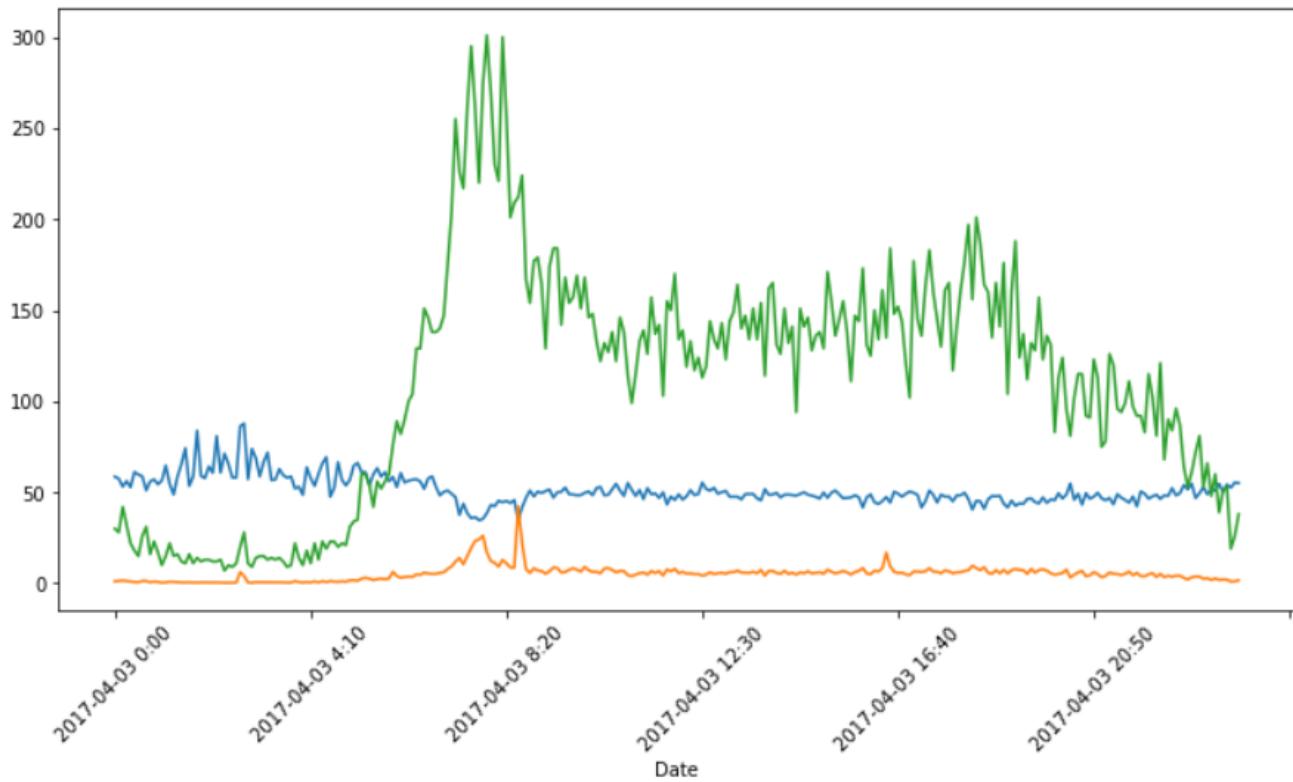


연습문제: 속도 데이터

In [33]: `## 2017년 4월 3일은 월요일.`

```
df['Speed'][288:576].plot(rot=45,figsize=(12,6))
df['Occ.Rate'][288:576].plot(rot=45,figsize=(12,6))
df['ToVol'][288:576].plot(rot=45,figsize=(12,6))
```

Out[33]: <matplotlib.axes._subplots.AxesSubplot at 0x29943e02748>



❖ 라벨을 정하기

- ✓ 지도학습을 위해서 데이터를 Feature와 Label로 나누자.
- ✓ 무엇이 Feature이며 무엇이 Label인가?

❖ 라벨은 무엇으로 정하는게 좋은가?

- ✓ 속도

- 혼잡(Jam) : 0~25km
- 서행(Slow): 25~50km
- 원활(Fast): 50~

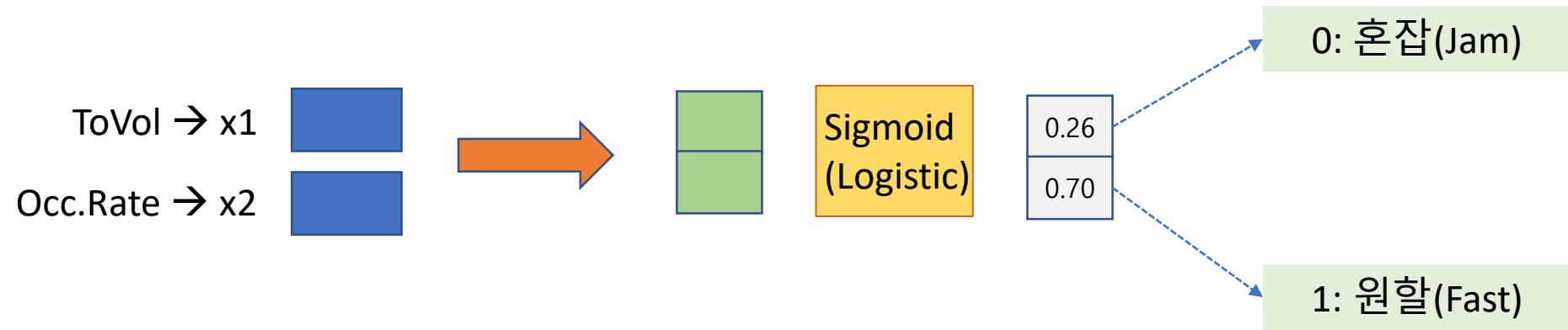
- ✓ 점유율

- ✓ 교통량 (대형, 중형, 소형, 통합)

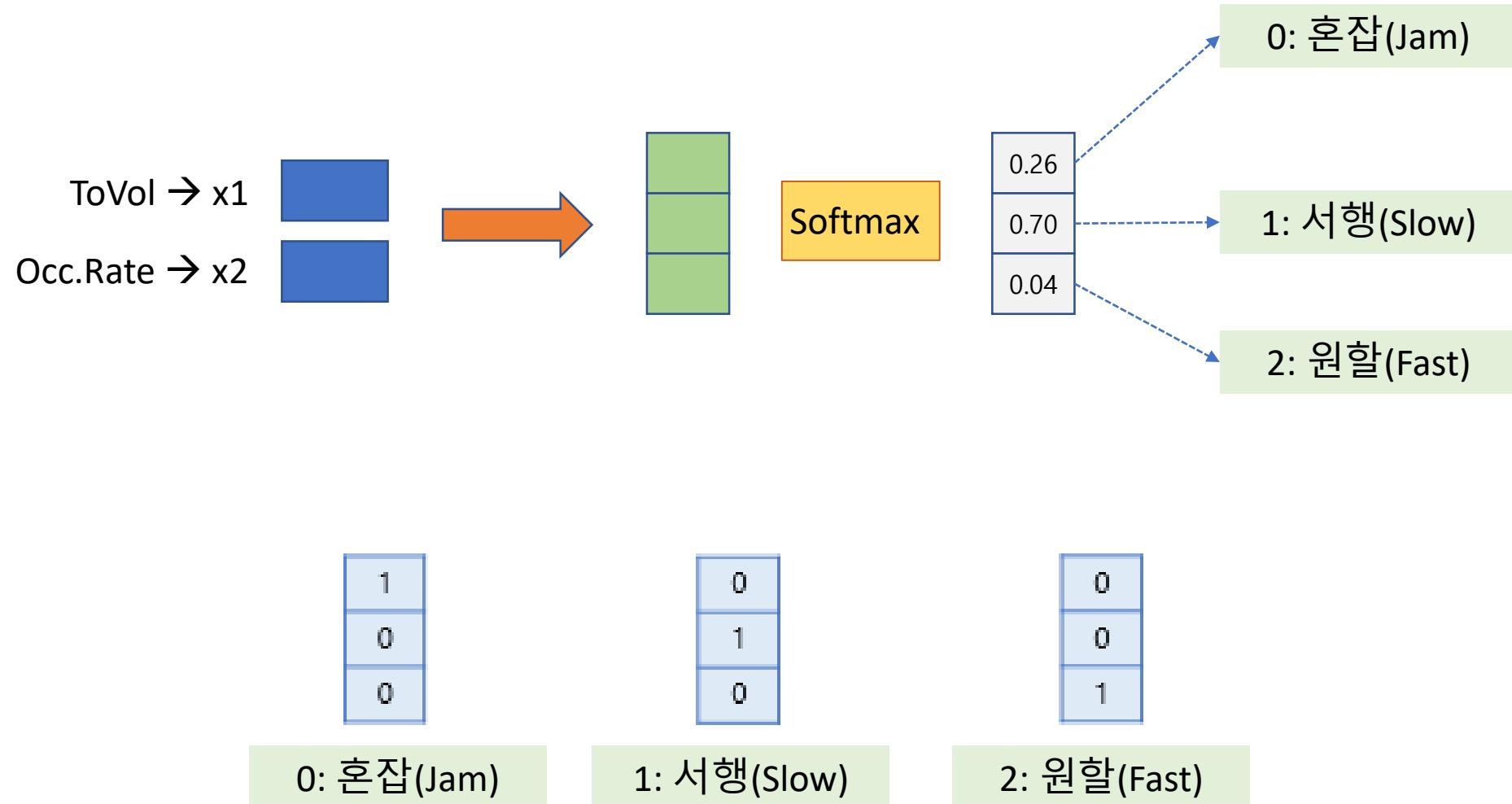
Traffic-jam Level	Speed Performance Index	Traffic State	Traffic Context
1	[0, 0.25]	Traffic jam	The average speed is the lowest; the road traffic state is very poor.
2	(0.25, 0.50]	Slow	The average speed is low; the road traffic state is poor.
3	(0.5, 0.75]	Moderate	The average speed is moderate; the road traffic state is a little congested.
4	(0.75, 1.0]	Fast	The average speed is high; the road traffic state is good.

https://www.researchgate.net/figure/The-criterion-of-traffic-jam-level-and-the-speed-performance-index_tbl1_322834098

라벨이 2개 일경우



라벨이 3개 일경우



데이터 가시화

In [17]:

```
def get_score(v):
    if v < 30:
        score = 'Jam'
    elif v < 50:
        score = 'Slow'
    else :
        score = 'Fast'
    return score
```

In [18]:

```
df["label_speed"] = df["Speed"].apply(lambda v: get_score(v))
df.head(4)
```

Out[18]:

	ToVol	SmVol	MeVol	LaVol	Speed	Occ.Rate	label_speed
Date							
2017-04-02 0:00	43	34	9	0	50.3	1.90	Fast
2017-04-02 0:05	45	32	13	0	58.9	1.84	Fast
2017-04-02 0:10	46	34	12	0	50.6	1.87	Fast
2017-04-02 0:15	45	36	9	0	50.9	1.72	Fast

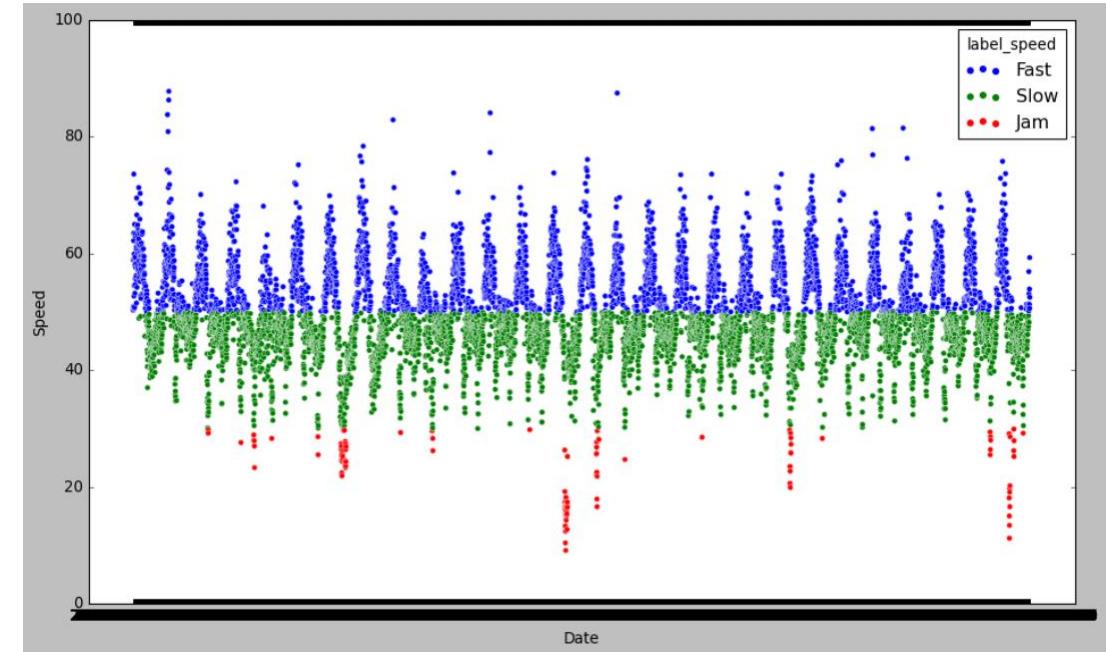
❖ Matplotlib과 Seaborn을 이용한 가시화

3. 데이터 가시화

```
In [20]: import matplotlib.pyplot as plt  
import seaborn as sns  
plt.style.use("classic")
```

```
In [*]: plt.figure(figsize=(14,8))  
sns.scatterplot(data=df, x = 'Date', y = 'Speed', hue='label_speed')
```

```
Out[21]: <matplotlib.axes._subplots.AxesSubplot at 0x25ac9ab1148>
```

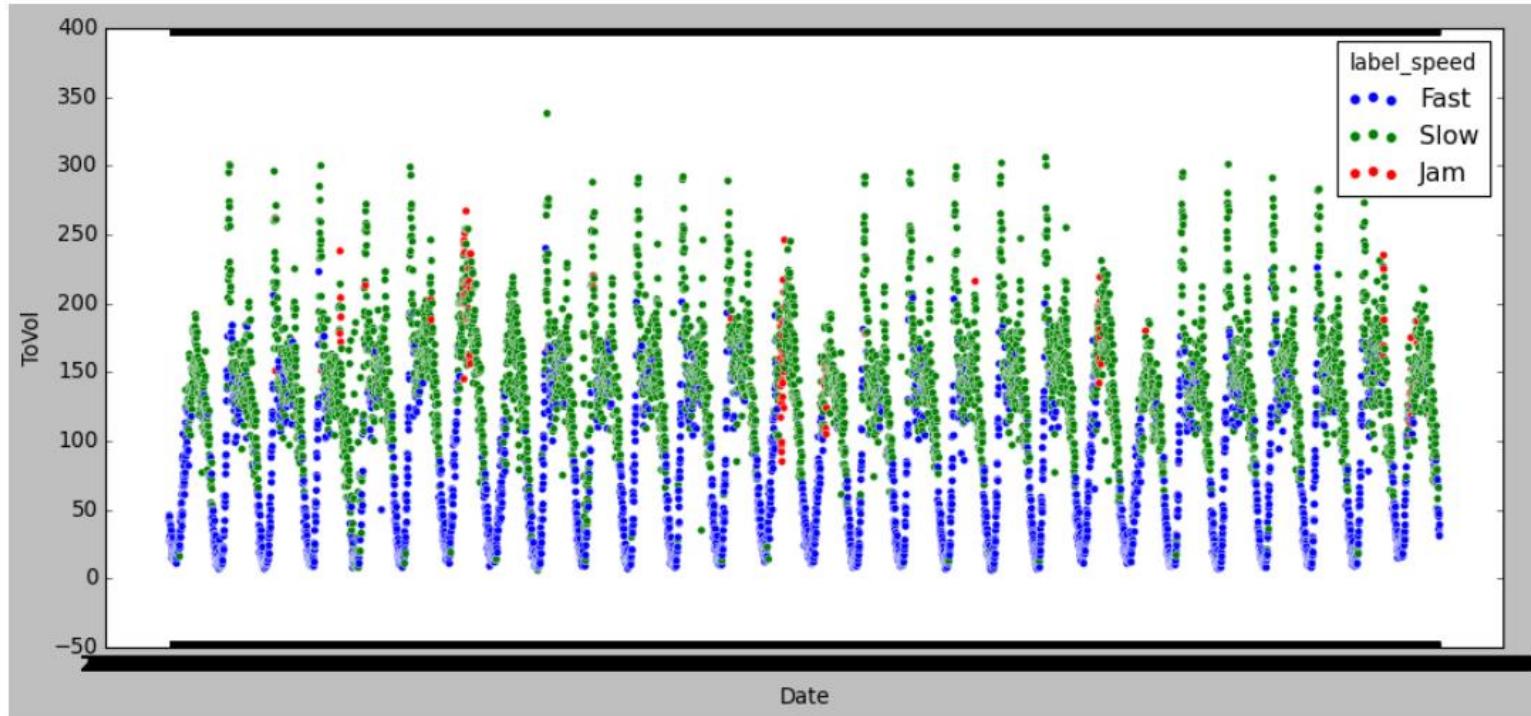


라벨링을 가시화 해보자: 교통량과 점유률



```
plt.figure(figsize=(14,6))
sns.scatterplot(data=df, x = 'Date', y = 'ToVol', hue='label_speed')
```

: <matplotlib.axes._subplots.AxesSubplot at 0x25ad6401108>

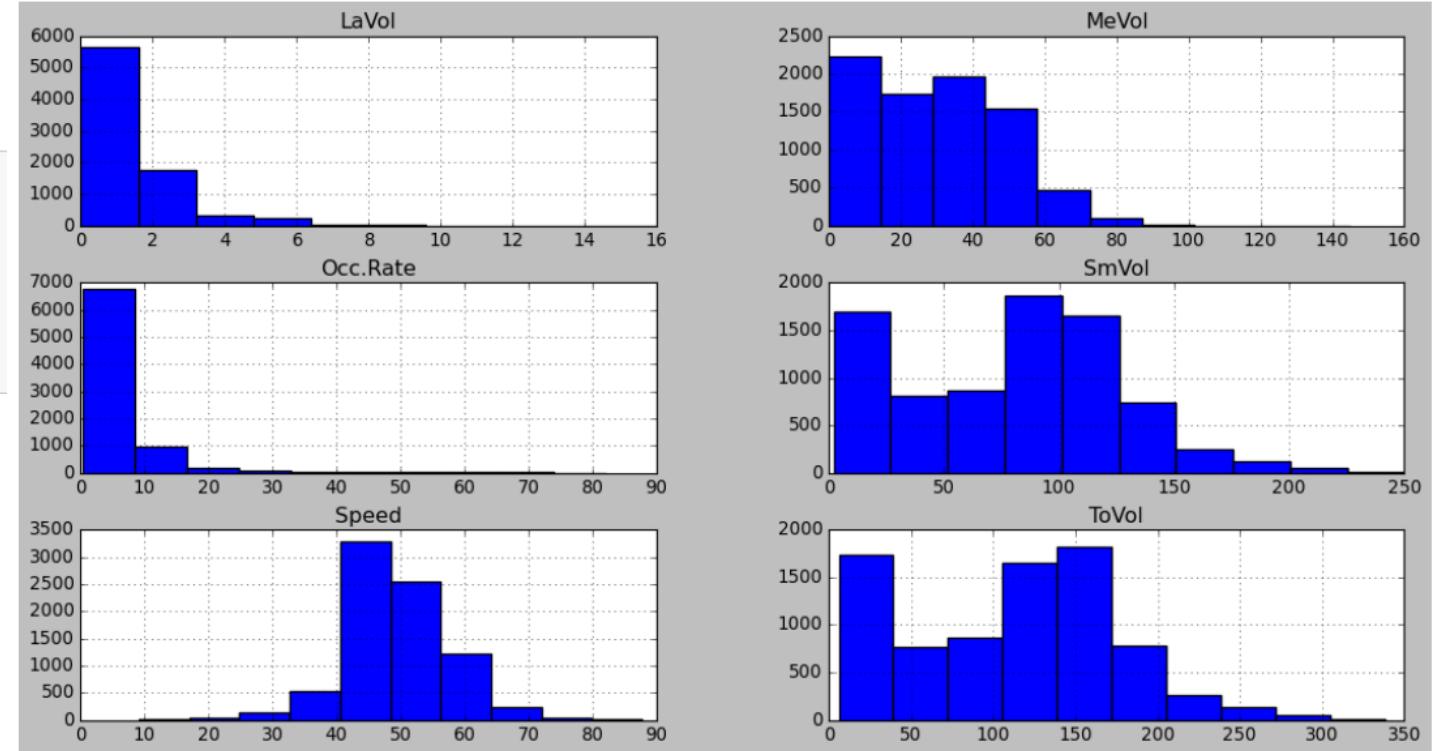


```
In [*]: plt.figure(figsize=(14,6))
sns.scatterplot(data=df, x = 'Date', y = 'Occ.Rate', hue='label_speed')
```

히스토그램으로 확인하기



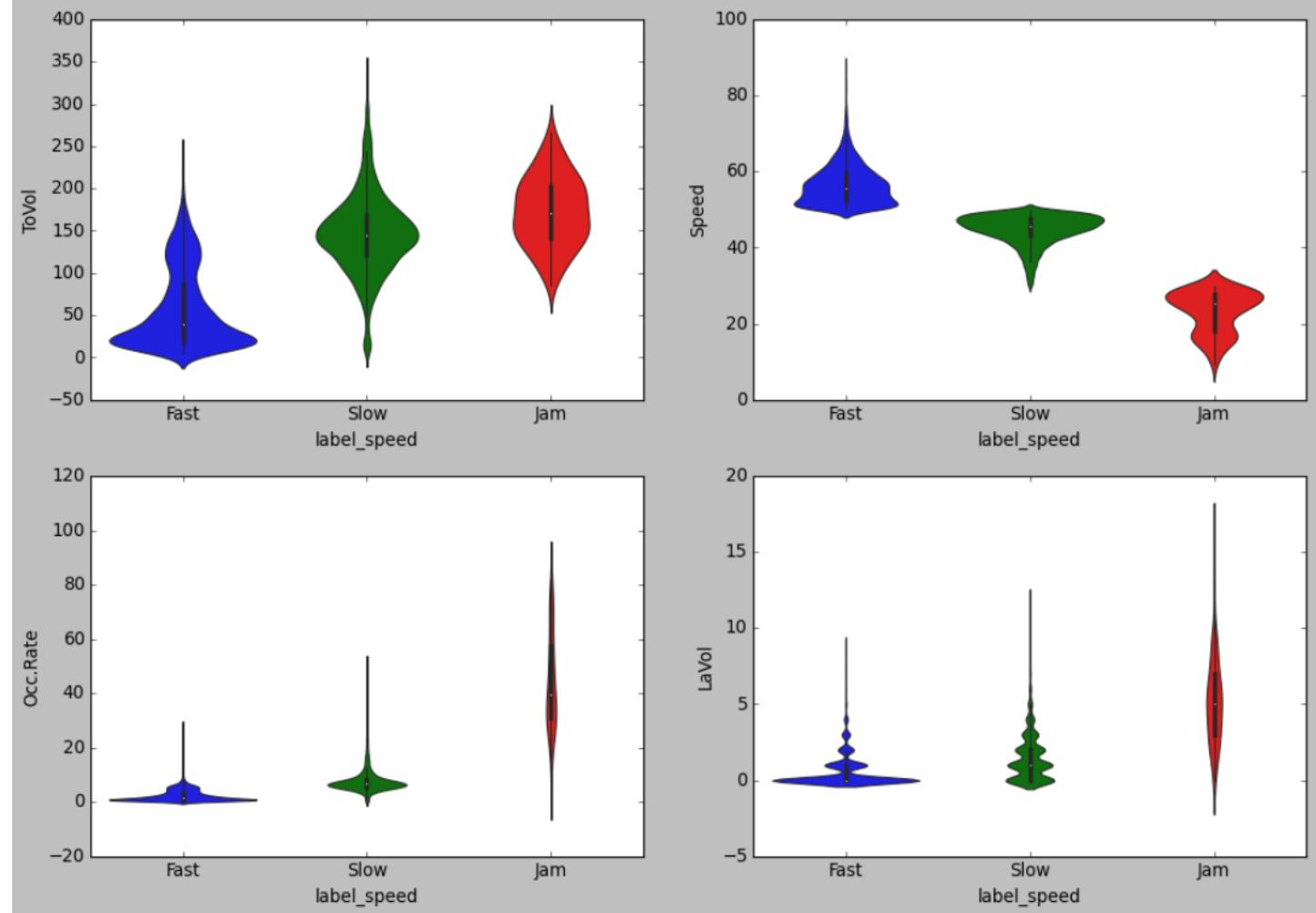
```
In [24]: df.hist(edgecolor='black', linewidth=1.2)  
fig=plt.gcf()  
fig.set_size_inches(16,8)  
plt.show()
```



Seaborn으로 가시화

```
plt.figure(figsize=(15,10))

plt.subplot(2,2,1)
sns.violinplot(x='label_speed',y='ToVol',data=df)
plt.subplot(2,2,2)
sns.violinplot(x='label_speed',y='Speed',data=df)
plt.subplot(2,2,3)
sns.violinplot(x='label_speed',y='Occ.Rate',data=df)
plt.subplot(2,2,4)
sns.violinplot(x='label_speed',y='LaVol',data=df)
```



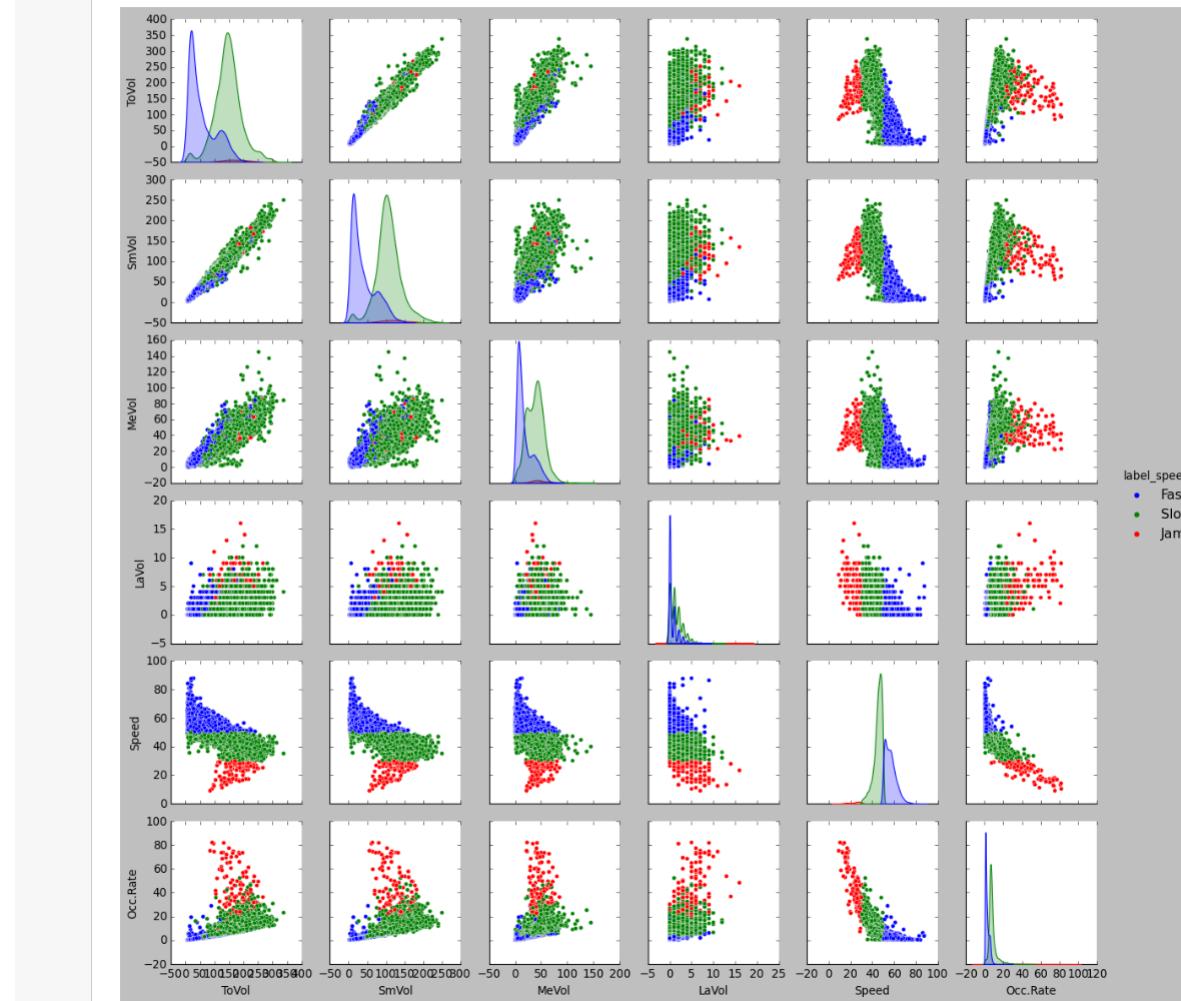
Pair correlation

In [26]:

```
sns.pairplot(df,hue='label_speed')
```

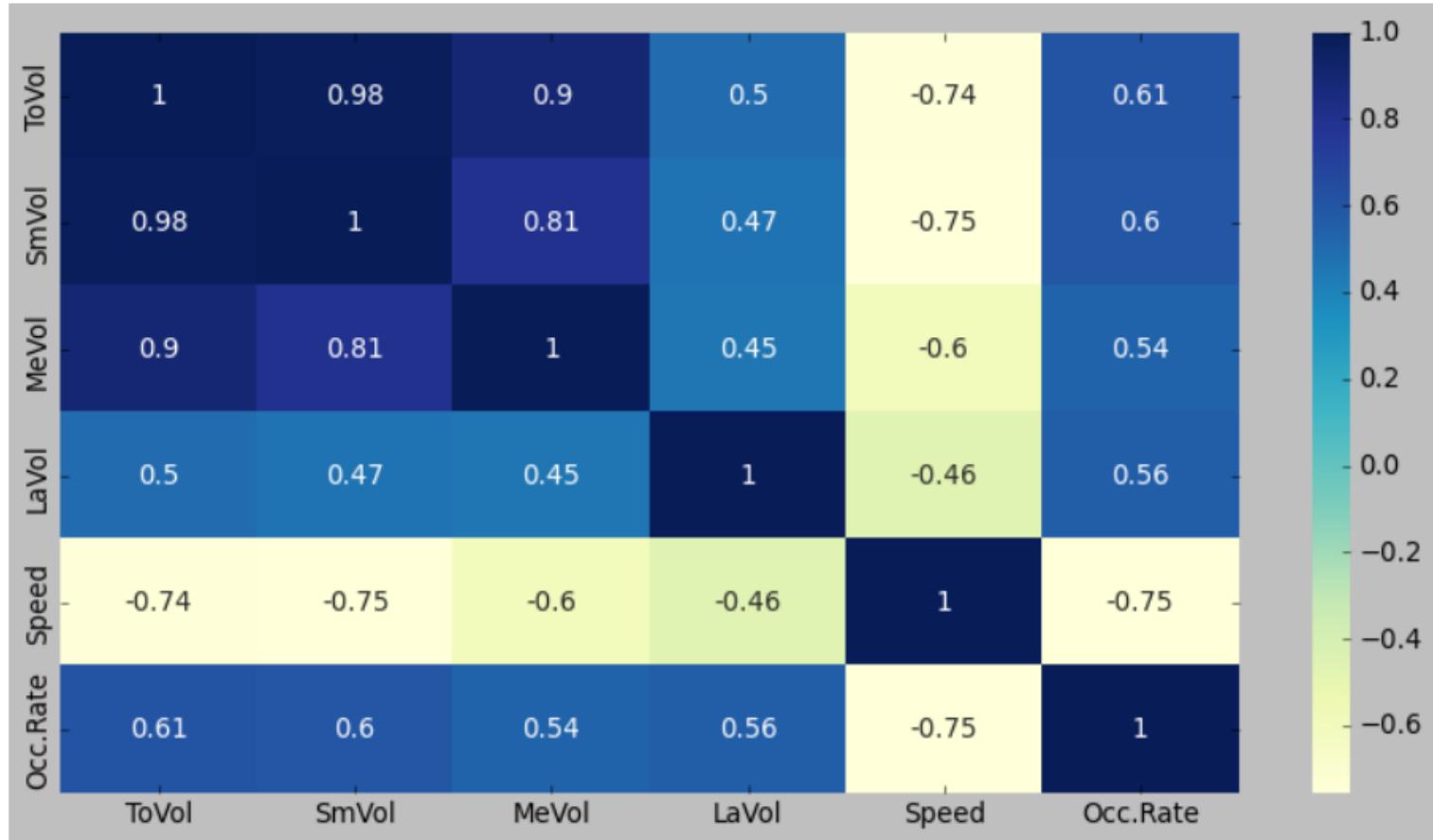
Out[26]:

```
<seaborn.axisgrid.PairGrid at 0x25adf51e7c8>
```



HeatMap 가시화

```
plt.figure(figsize=(12,6))
sns.heatmap(df.corr(), annot=True, cmap='YlGnBu')
plt.show()
```

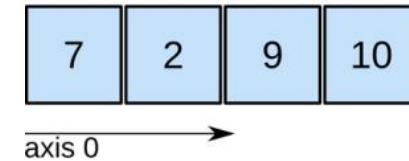


강의 5

스마트교통을 위한 머신러닝 기본 개념 소개

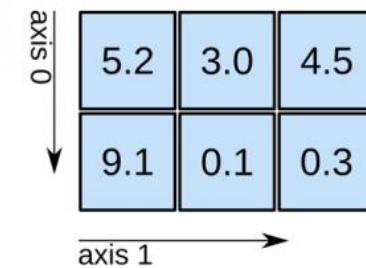


1D array



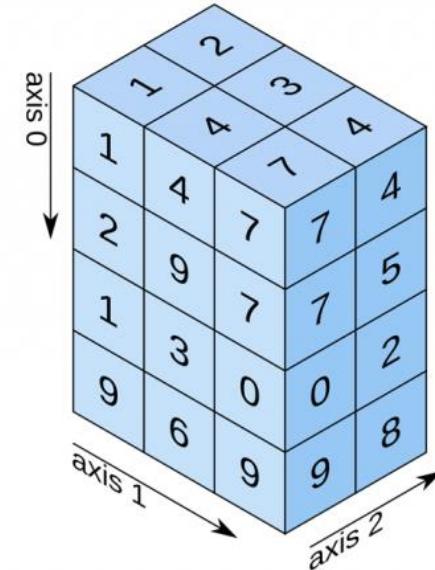
shape: (4,)

2D array

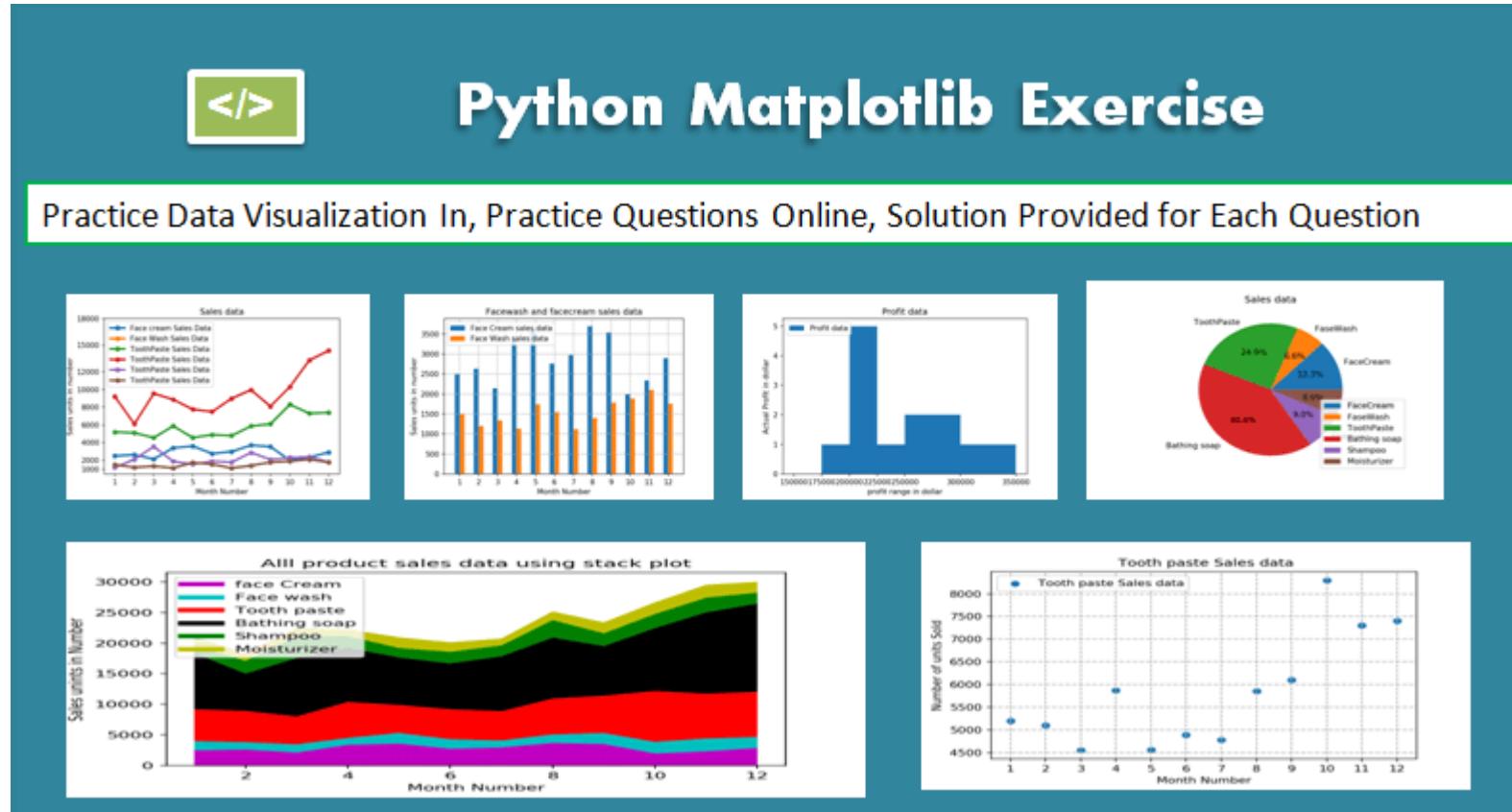


shape: (2, 3)

3D array

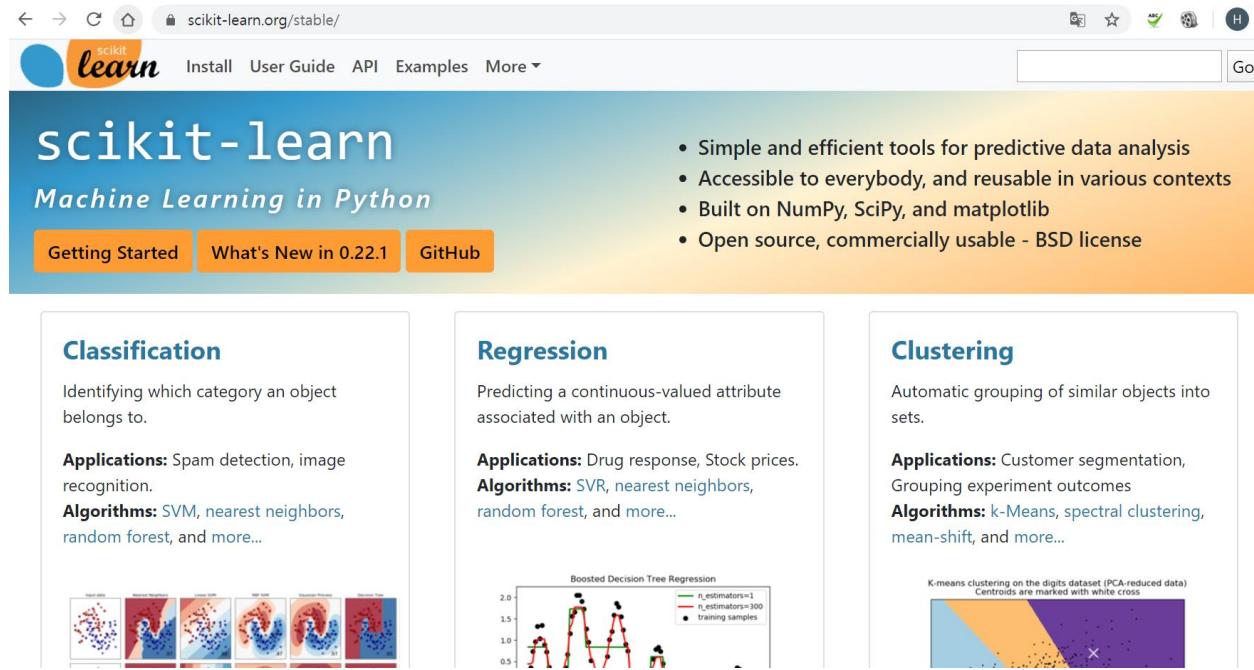


shape: (4, 3, 2)



❖ 파이썬 머신러닝 중에서 가장 많이 사용되는 라이브러리

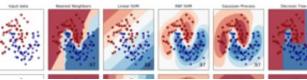
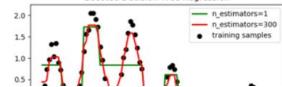
- ✓ 예측 데이터 분석을 위해 간단하고 효과적인 툴 제공
- ✓ Numpy, SciPy, Matplotlib을 기반으로 구성
- ✓ 아나콘다를 설치하면 기본적으로 사이킷런까지 설치가 완료 됨
 - \$ conda install scikit-learn



The screenshot shows the official website for scikit-learn at scikit-learn.org/stable/. The header includes the KISTI logo and the text "SINCE 1962". Below the header, there's a navigation bar with links for "Install", "User Guide", "API", "Examples", and "More". The main content area has a blue header with the text "scikit-learn" and "Machine Learning in Python". Below this, there are three orange buttons: "Getting Started", "What's New in 0.22.1", and "GitHub". To the right of the header, there's a list of bullet points:

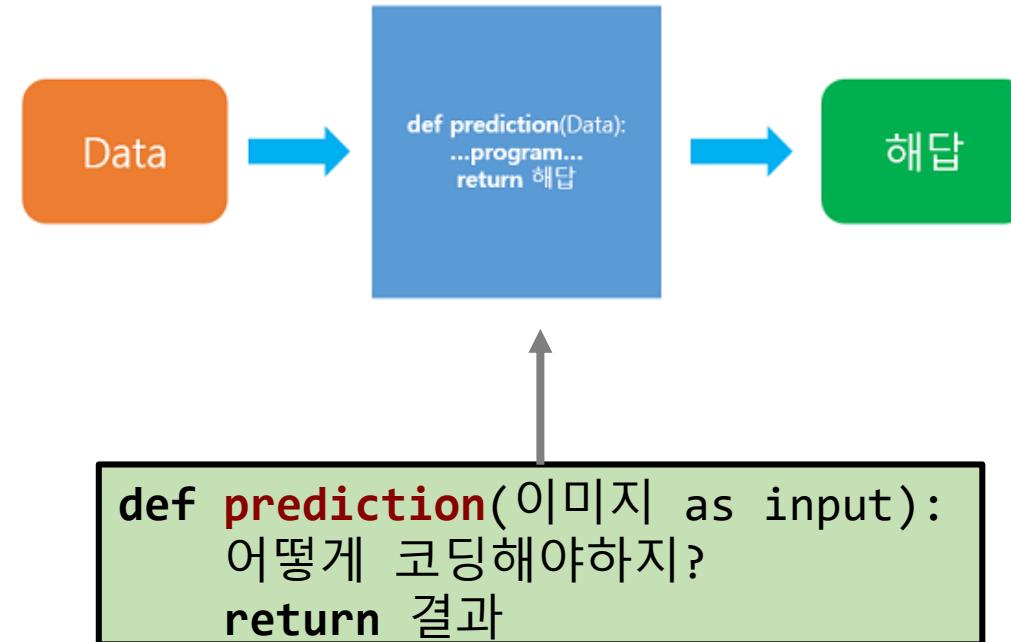
- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Below this, there are three main sections with sub-sections and images:

- Classification**:
Identifying which category an object belongs to.
Applications: Spam detection, image recognition.
Algorithms: SVM, nearest neighbors, random forest, and more...

- Regression**:
Predicting a continuous-valued attribute associated with an object.
Applications: Drug response, Stock prices.
Algorithms: SVR, nearest neighbors, random forest, and more...

- Clustering**:
Automatic grouping of similar objects into sets.
Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, spectral clustering, mean-shift, and more...


❖ 기존의 프로그래밍 접근 방법

- ✓ Ex) 주어진 사진으로부터 고양이 사진인지 강아지 사진인지 판별하는 일.

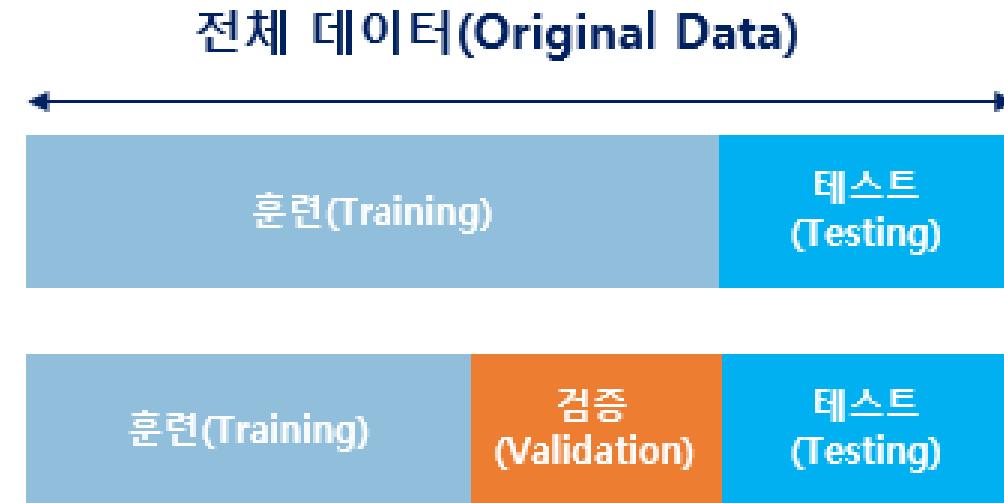


❖ 기존 프로그래밍의 한계에 대한 해결책: 학습(Train)

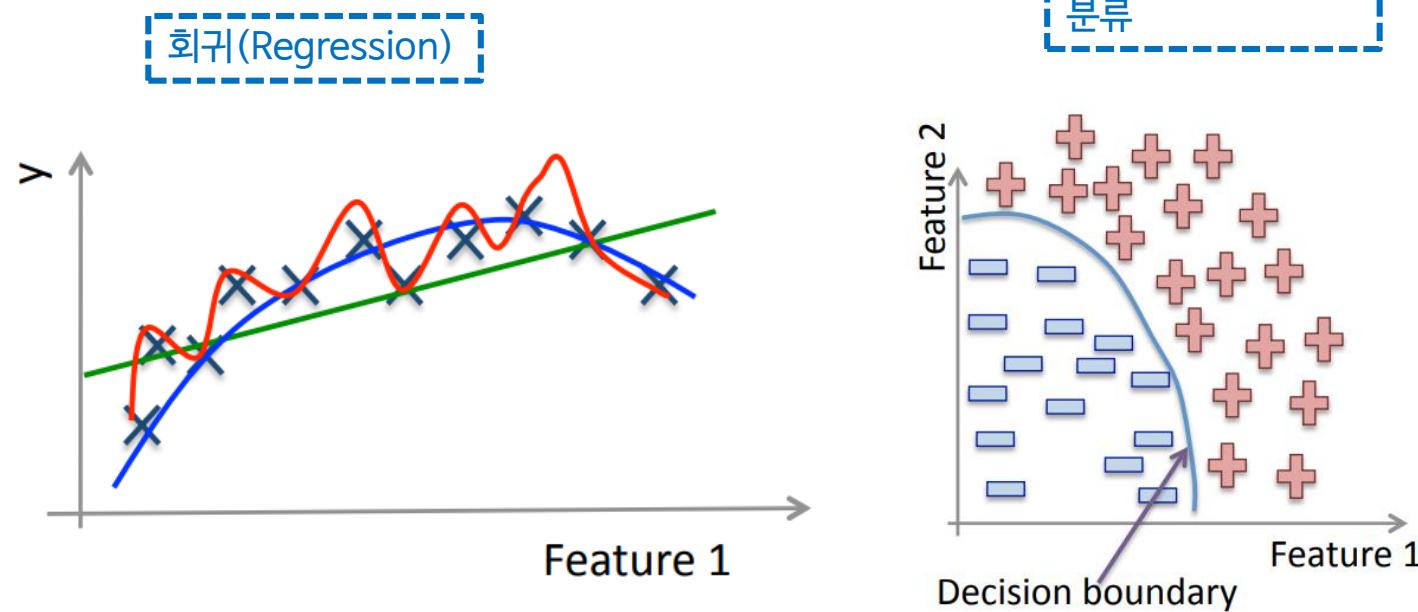
- ✓ 머신 러닝은 주어진 데이터로부터 규칙성 또는 패턴을 찾는 것에 초점이 맞추어져 있다.
- ✓ 주어진 데이터로부터 규칙성을 찾는 과정을 우리는 학습(training)이라고 합니다.
- ✓ 일단 규칙성을 발견해내면, 그 후에 들어오는 새로운 데이터에 대해서 발견한 규칙성을 기준으로 정답을 찾아내는데, 이는 기존의 프로그래밍 방식으로 접근하기 어려웠던 문제의 해결책이 됨



- ❖ 실제 모델 훈련 및 평가하기
- ❖ 데이터를 훈련용, 검증용, 테스트용 이렇게 세 가지로 분리
 - ✓ 테스트 데이터는 20%~30% 정도
 - ✓ 검증용 데이터 약 10%정도는 과적합을 판단하거나 하이퍼파라미터의 조정을 위한 용도



- ❖ 지도학습 알고리즘
- ❖ Support Vector Machines, neural networks,
- ❖ decision trees, K-nearest neighbors, naive Bayes
- ❖ 레이블이 있어야 함. 누가 레이블을 만드나?



❖ 특징

- ✓ 파이썬 머신러닝 중에서 가장 많이 사용되는 라이브러리
- ✓ 예측 데이터 분석을 위해 간단하고 효과적인 툴 제공
- ✓ Numpy, SciPy, Matplotlib을 기반으로 구성
- ✓ 공개 소프트웨어, 상업용을 사용 불가 - BSD 라이선스
- ✓ 최근 텐서플로우, 케라스, 파이토치 등 딥러닝 전문 라이브러리와 경쟁 중

❖ 설치

- ✓ 아나콘다를 설치하면 기본적으로 사이킷런까지 설치가 완료 됨
- ✓ 가능하면 conda로 설치를 권장
 - `$ conda install scikit-learn`
- ✓ 버전을 확인
 - `$ print(sklearn.__version__)`

머신러닝 모델 프로세스

1) 데이터의 전처리 과정

- ✓ 피처와 라벨을 결정함,
- ✓ 필요한 피처를 추출하고 가공함,
- ✓ 입력과 출력을 위한 피처의 정규화(표준화)

2) 머신러닝 모델을 선정

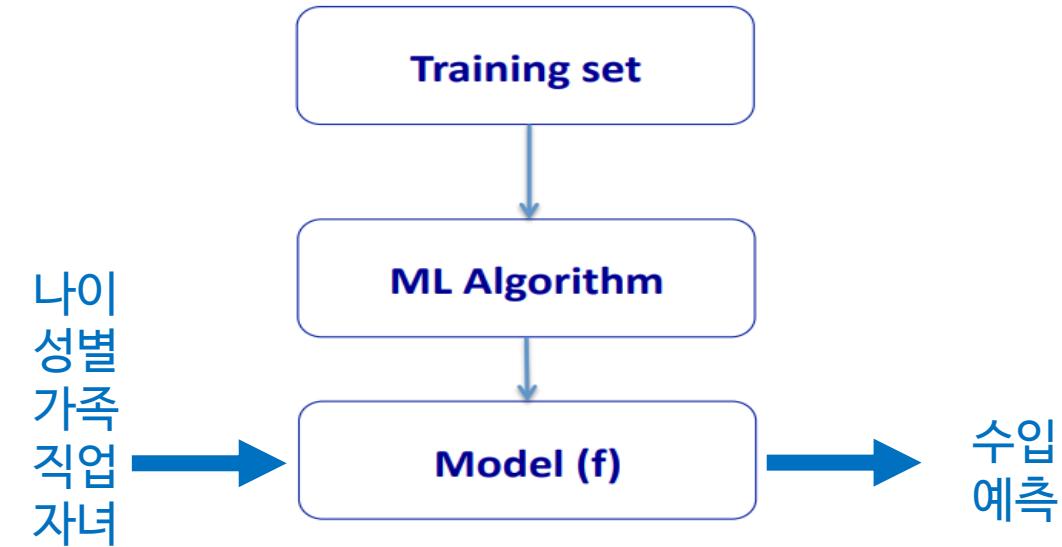
- ✓ 학습, 검증, 테스트 데이터로 나누자
- ✓ 적용할 머신러닝 알고리즘을 선정하자.
 - 분류 혹은 회귀 문제인지 먼저 정하자.

3) 모델 훈련

- ✓ 모델을 훈련시키자.
- ✓ 과적합 발생등 확인하자.

4) 모델 평가하자.

- ✓ 모델의 테스트로 평가하자.



❖ 지도학습의 분류와 회귀

- ✓ 모델 학습은 fit()으로 하고
- ✓ 모델 평가 후 예측은 predict()를 이용

❖ Estimator란?

- ✓ regressor + classifier 클래스로 지칭

❖ 비지도학습에서는

- ✓ 차원축소, 클러스터링, 피처추출
- ✓ fit()와 transform()을 적용
 - 여기서 fit()은 지도학습의 fit()과 다름.
 - 입력데이터 형태에 맞춰 데이터를 변환하기 위한 사전 구조를 맞추는 작업
 - 실제 fit()로 사전 구조를 맞추면 이후 입력데이터의 차원변환 등 transform()로 처리
- ✓ fit_transform()도 제공함.

❖ 피처 스케일링(feature scaling): 서로 다른 변수의 값 범위를 일정하게 맞추는 작업

✓ 표준화(Standardization)

- 데이터 피처 각각이 평균이 0이고 분산이 1인 가우시안 분포로 변환

$$x_i - new = \frac{x_i - mean(x)}{stddev(x)}$$

✓ 정규화(Normalization)

- 서로 다른 피처의 크기를 통일하기 위해 크기를 변환해주는 개념
- 값이 0과 1 사이로 변환하는 것

$$x_i - new = \frac{x_i - min(x)}{max(x) - min(x)}$$

- ❖ 분류문제는 회귀 분석과 달리 다양한 성능평가 기준이 필요하다.
- ❖ 사이킷런 패키지에서 지원하는 분류 성능평가 명령
 - ✓ `confusion_matrix(y_true, y_pred)`
 - ✓ `accuracy_score(y_true, y_pred)`
 - ✓ `precision_score(y_true, y_pred)`
 - ✓ `recall_score(y_true, y_pred)`
 - ✓ `f1_score(y_true, y_pred)`
 - ✓ `classification_report(y_true, y_pred)`
 - ✓ `roc_curve`
 - ✓ `auc`

Confusion Matrix 이해

❖ 임신테스트 예제로 임신이면 +(1), 아니면 -(0)

- ✓ TP (True Positive)
- ✓ TN (True Negative)
- ✓ FP (False Positive)
- ✓ FN (False Negative)

❖ 정밀도 = $TP/(FP+TP)$

- ✓ FP를 낮추는 데 초점

❖ 재현율 = $TP/(FN+TP)$

- ✓ 재현율은 FN을 낮추는데 초점

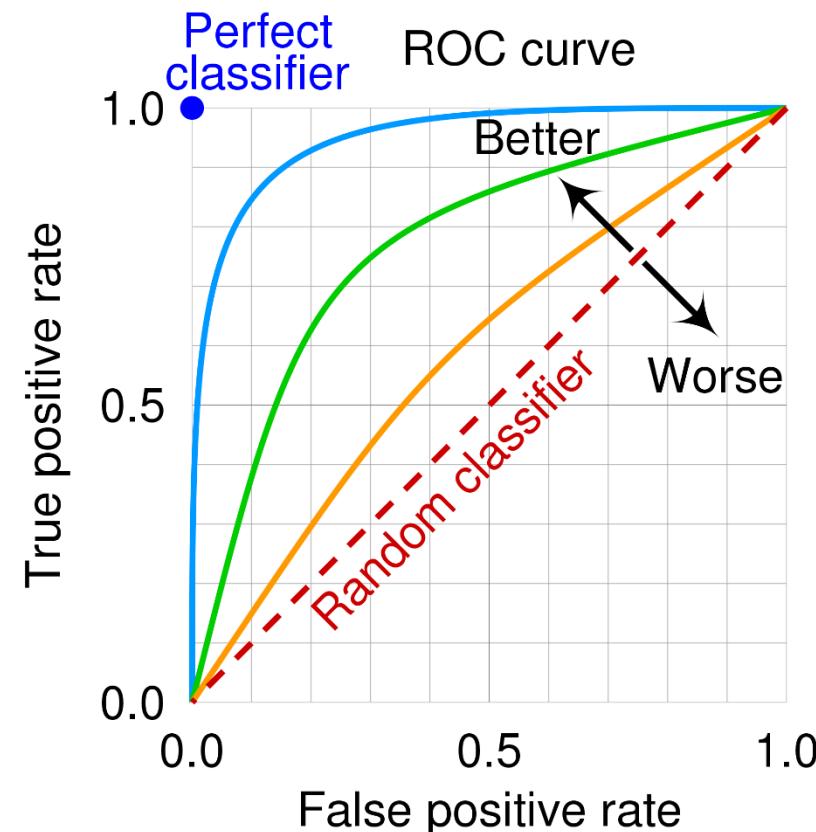


		Predicted		
		Negative (0)	Positive (1)	
Actual	Negative (0)	True Negative TN	False Positive FP (Type I error)	Specificity $= \frac{TN}{TN + FP}$
	Positive (1)	False Negative FN (Type II error)	True Positive TP	Recall, Sensitivity, True positive rate (TPR) $= \frac{TP}{TP + FN}$
Accuracy $= \frac{TP + TN}{TP + TN + FP + FN}$		Precision, Positive predictive value (PPV) $= \frac{TP}{TP + FP}$	F1-score $= 2 \times \frac{Recall \times Precision}{Recall + Precision}$	

<https://velog.io/tags/roc-auc>

❖ ROC (Receiver Operation Characteristic Curve)

- ✓ x축: FPR(False Positive Rate)
 - TNR(True Negative Rate)은 특이성 (Specificity)라고 부름
 - $FPR = FP / (FP + TN) = 1 - TNR = 1 - \text{특이성}$
- ✓ y 축: TPR(True Positive Rate)
 - TPR은 recall(재현율)이다, 민감도 (Sensitivity)라고 부름
- ✓ ROC 곡선이 직선일 경우 ($AUC=0.5$)
 - 모델 예측 성능은 떨어짐



https://en.wikipedia.org/wiki/Receiver_operating_characteristic#/media/File:Roc_curve.svg

- ❖ 장점

- ✓ 데이터의 전처리가 거의 필요없다.
- ✓ 특성 스케일을 맞추거나, 원점에 맞추는 작업은 필요가 없다.

- ❖ 노드의 속성은 얼마나 많은 훈련 샘플이 적용되었는지 헤아린 것이다.

- ✓ 100개의 훈련 샘플의 꽃의 길이가 2.45cm보다 길고(깊이1, 오른쪽)
 - 그 중에 54개 샘플이 1.75보다 짧고 (깊이2, 왼쪽)

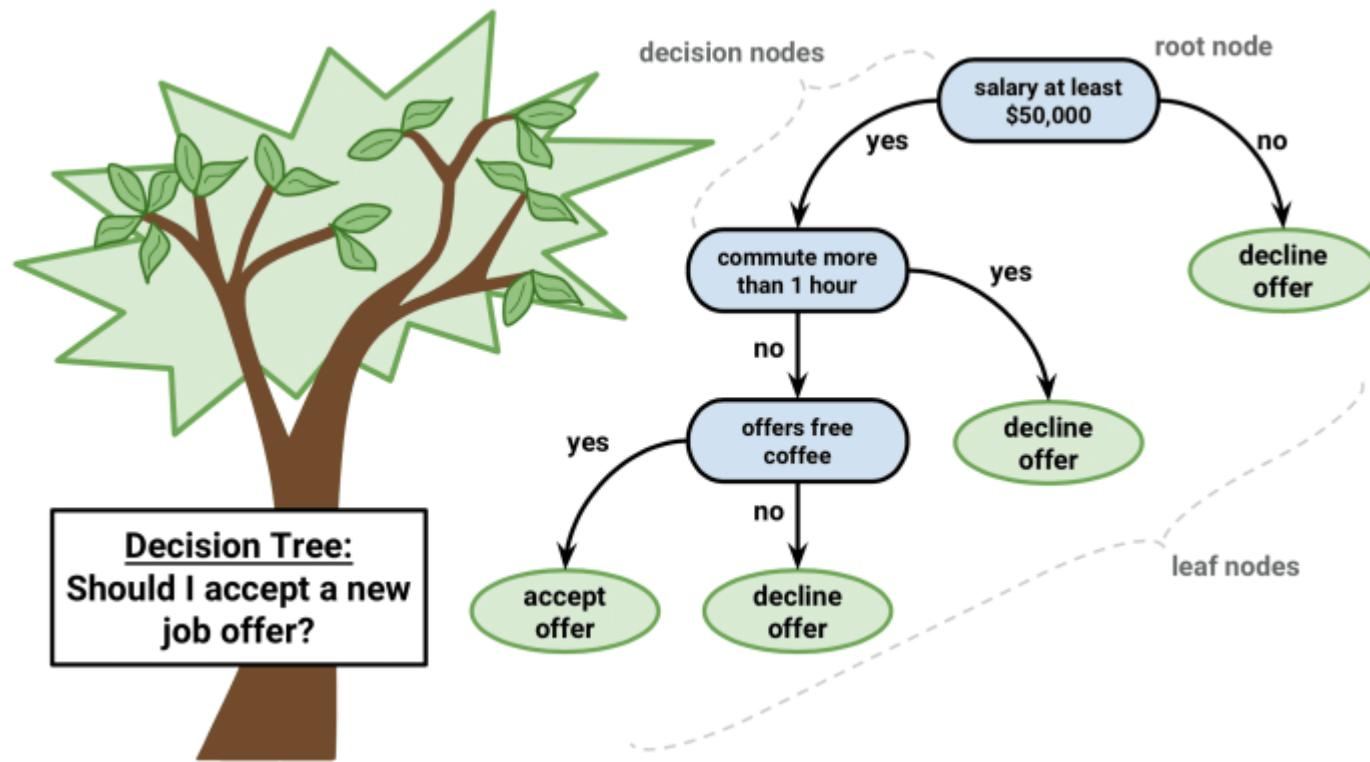
- ❖ 노드의 value 속성은 노드에서 각 클래스에 있는 훈련 샘플 개수다.

- ✓ 맨 오른쪽 아래 노드 Setosa=0, Versicolor=1, Virginica=45개

- ❖ 결정트리는 화이트 모델로 직관적이고 이해하기 쉽다.

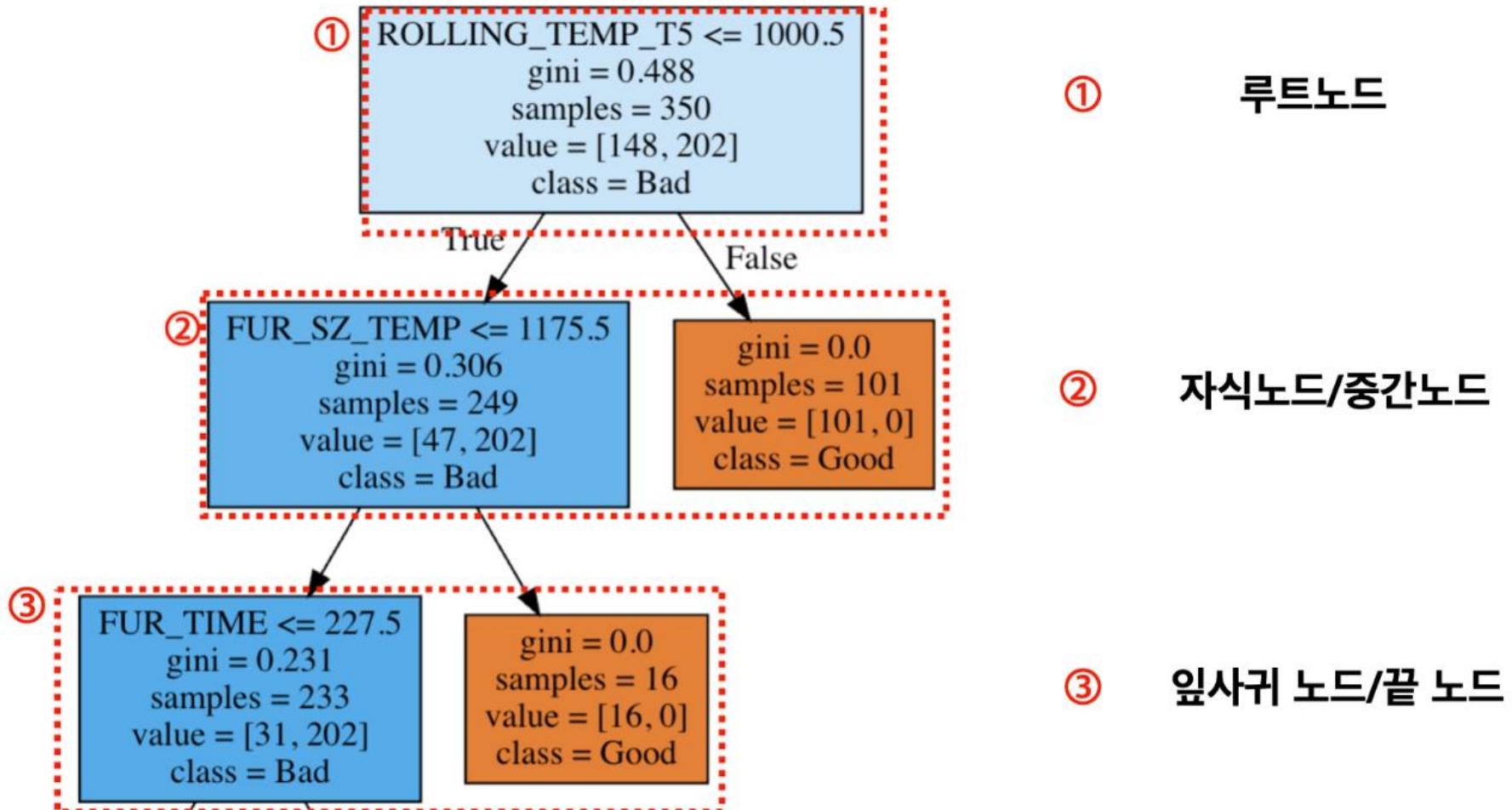
- ❖ 램덤 포레스트나 신경망은 블랙박스 모델이다.

Decision trees



<https://goldinlocks.github.io/Decision-trees-in-python/>

Building Blocks of a Decision-Tree



<https://heytech.tistory.com/>

의사결정나무 구조 및 용어

No	항목	설명
1	루트노드(Root Node)	나무가 시작되는 노드를 의미합니다. 그림 1 과 같이 의사결정나무를 시작화 했을 때 루트노드는 맨 위에 위치합니다.
2	자식노드(Child Node)	상위의 노드에서 분리된 하위 노드를 의미합니다.
3	부모노드(Parent Node)	자식 노드의 상위 노드를 의미합니다.
4	중간노드(Internal Node)	나무 중간에 위치한 노드로 루트노드 또는 최하위 노드가 아닌 모든 노드가 해당됩니다.
5	가지(Branch)	하나의 노드로부터 잎사귀 노드까지 연결된 일련의 노드들을 포함합니다.
6	잎사귀 노드(Leaf Node) 또는 끝 노드(Terminal Node)	각 가지 끝에 위치한 노드
7	순수노드(Pure Node)	해당 노드의 목표변수가 동일한 값이나 종류만 가지는 노드를 의미합니다.
8	깊이(Depth)	가지를 이루고 있는 노드의 분리 층수(그림 1 의 경우 depth = 3)

<https://heytech.tistory.com/145>

❖ 의사결정나무 기반 예측 (회귀) 모델링

✓ MSE(Mean Squared Error, 평균제곱오차)

- Mean Squared Error(MSE)는 평균 제곱오차라고 부르며 모델 예측값과 실젯값 간의 제곱오차의 평균을 의미
- 부모 노드의 평균제곱오차를 가장 많이 감소시키는 설명변수와 분리 값을 기준으로 자식 노드를 생성
- 따라서 MSE가 작을수록 오차가 적은 좋은 모델임.

✓ MAE(Mean Absolute Error, 평균절대오차)

- Mean Absolute Error(MAE)는 평균 절대오차로, 모델 예측값과 실젯값 간의 절대오차의 평균을 의미
- MSE와 마찬가지로 부모 노드의 평균 절대오차를 가장 많이 감소시키는 설명변수와 분리 값을 기준으로 자식 노드를 생성
- 따라서 MAE가 작을수록 오차가 적은 좋은 모델

❖ 지니 지수(Gini Index)

- ✓ 불순도 측정 지수로서 쉽게 말해 '얼마나 다양한 데이터가 잘 섞여있는지 정도'를 나타냄.
- ✓ 반대 개념인 순수도는 같은 클래스의 데이터가 얼마나 포함되었는지를 나타냄.

$$\text{지니 지수}: G = 1 - \sum_i^c p_i^2 , \quad 0 \leq G \leq 1/2$$

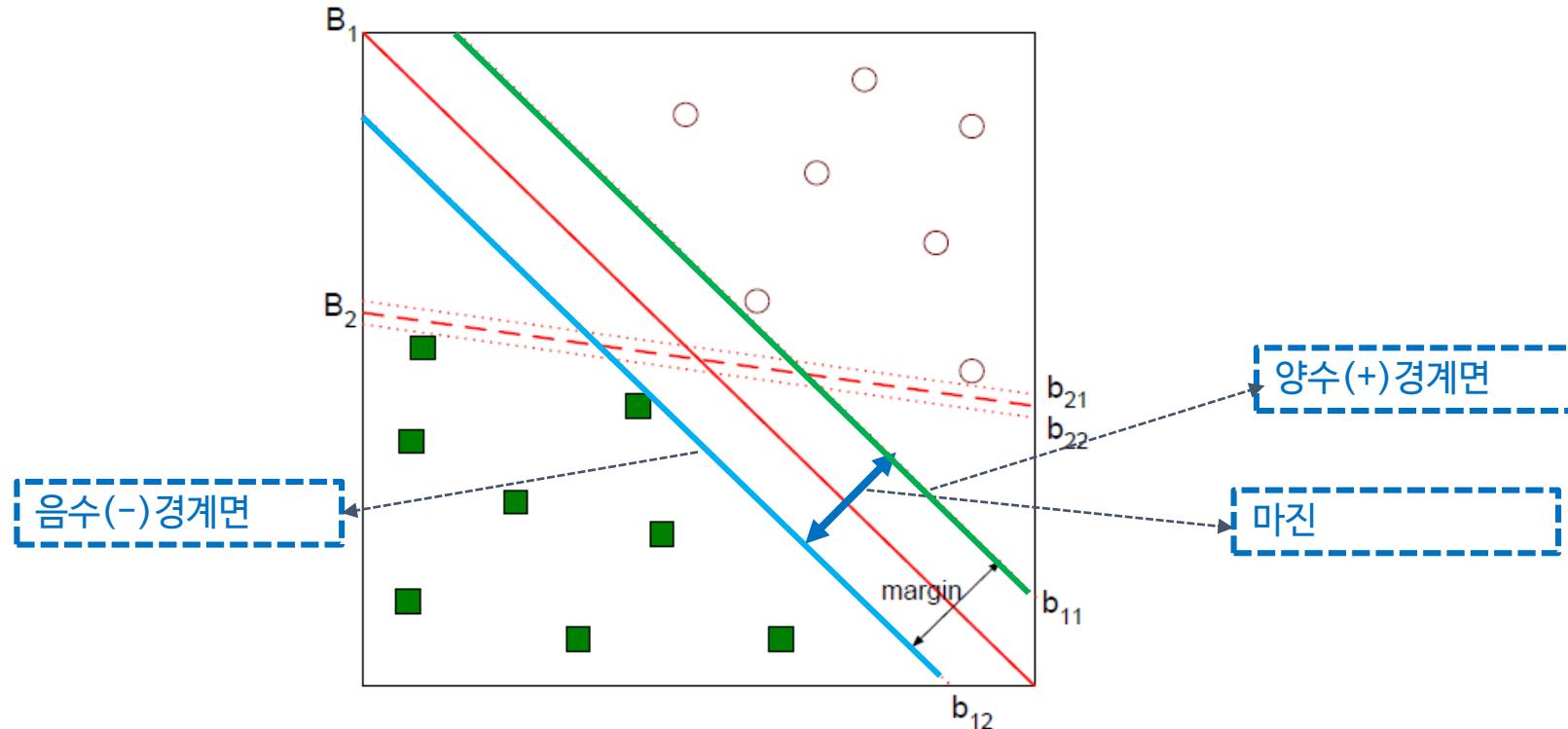
$$\text{엔트로피 지수}: E = - \sum_i^c p_i \log_2 p_i , \quad 0 \leq E \leq 1$$

❖ 엔트로피 지수(Entropy Index)

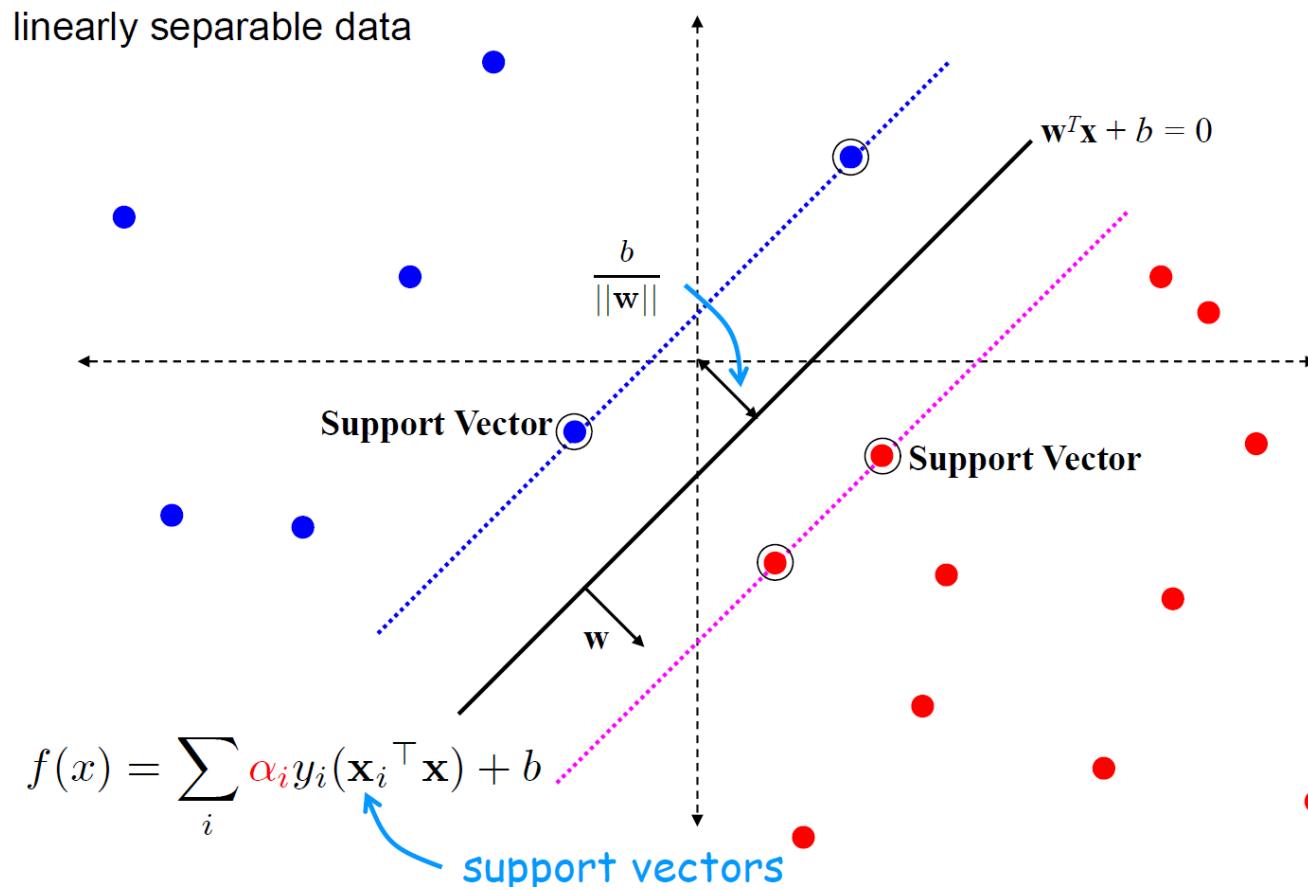
- ✓ '데이터가 섞여있는 정도'

마진(margin)이란 무엇인가?

- ❖ 결정 경계를 정하기 위해 마진(margin)을 도입하자.
 - ✓ 마진은 두 경계면 사이의 거리이며, 이 거리가 최대일때 좋은 결정 경계를 얻음.
- ❖ SVM의 목적은 마진을 최대화하는 경계면을 찾는 것



- ❖ 알파는 해당 벡터(x)가 경계선을 정하는 샘플이라는 뜻으로 “서포트 벡터 ”라고 부름

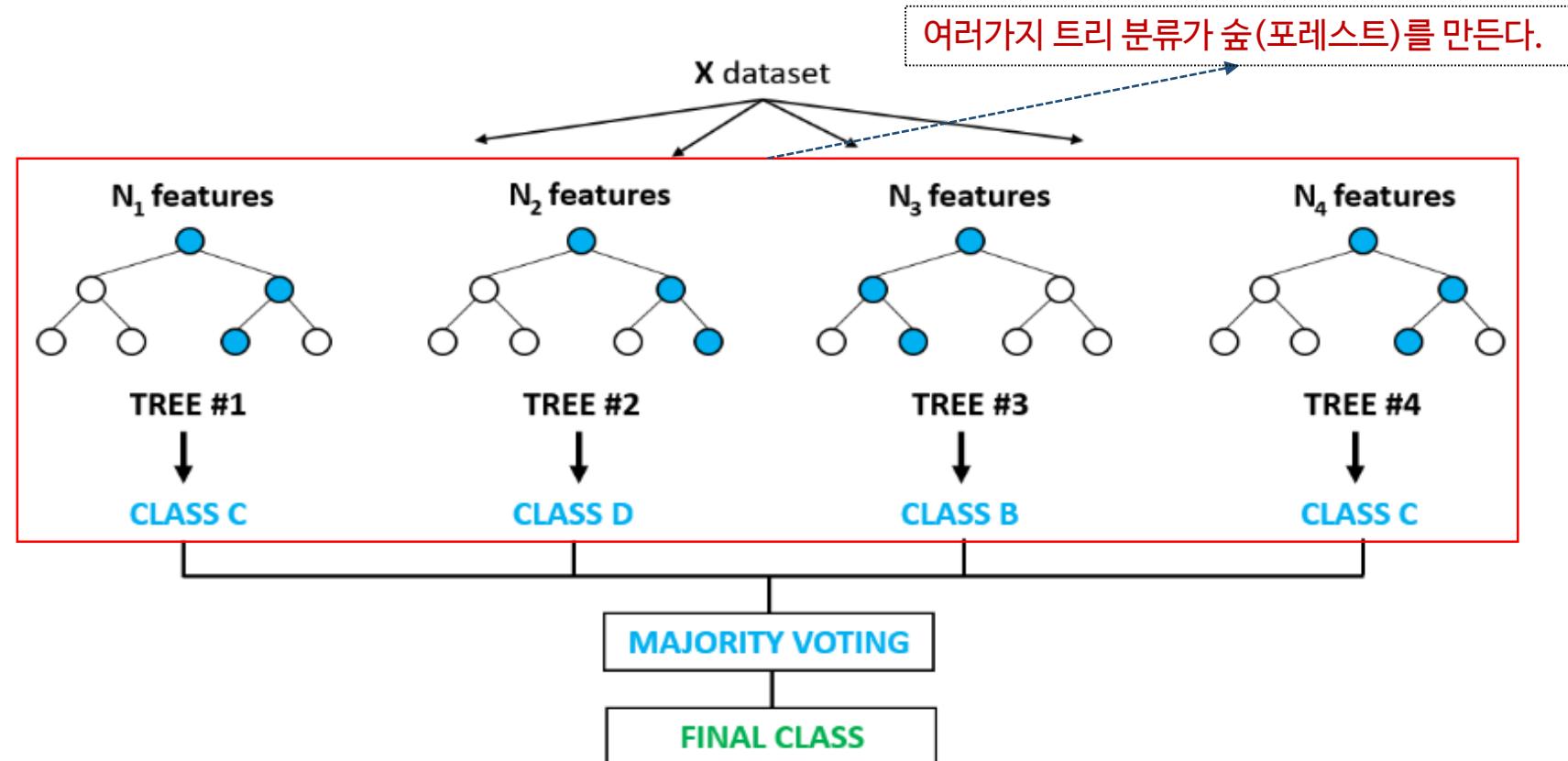


❖ 양상블 학습 유형

- ✓ 보팅(Voting) : 여러 개의 분류기가 투표를 통해 최종 예측
 - 서로 다른 알고리즘(knn, svm, kmeans)을 가진 분류기를 결합하는 방식
- ✓ 배깅(Bagging) : 여러 개의 분류기가 투표를 통해 최종 예측
 - 분류기는 같은 유형의 알고리즘이지만, 데이터 샘플링을 서로 다르게 학습
 - 대표적인 배깅 방식은 램덤 포레스트 알고리즘이다.
 - 부트스트래핑(Bootstrapping) 분할 방식을 사용, 즉, 원본 학습 데이터를 샘플링해서 추출하는 방식
 - 배깅은 교차 검증과 달리 데이터 셋트 간의 중첩을 허용하는 것이 차이점이다.
- ✓ 부스팅(Boosting)
 - 앞에서 학습한 분류기가 예측이 틀린 데이터에 대해서 올바른 예측을 할 수 있도록 다음 분류기에는 가중치 (weight)를 부여하면서 학습과 예측을 진행하는 것
- ✓ 스태킹(Stacking)
 - 여러 가지 다른 모델의 예측 결과 값을 다시 학습 데이터로 만들어서 다른 모델(메타모델)로 재학습시켜 결과를 예측하는 방법

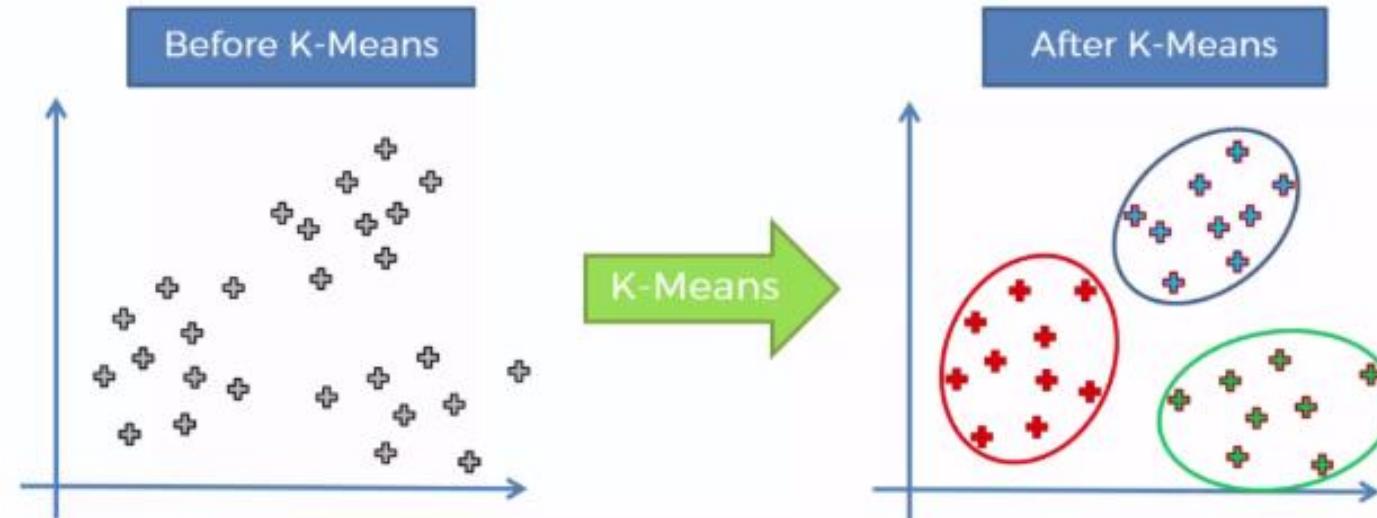
❖ 랜덤 포레스트 개요

- ✓ 배깅의 대표적 알고리즘은 랜덤 포레스트
- ✓ 여러 개의 결정 트리 분류기가 전체 데이터에서 배깅 방식으로 보팅 함

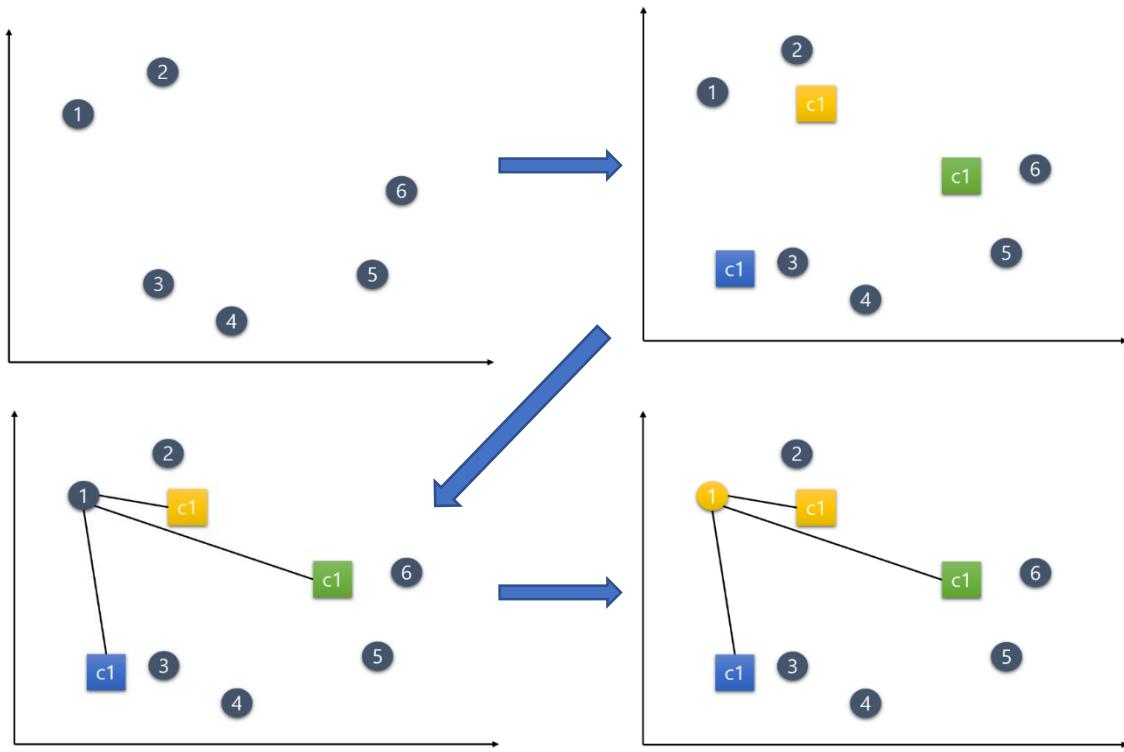


❖ K-Means는 비지도학습(Unsupervised Learning)

- ✓ 주어진 데이터를 k 개의 클러스터(군집)로 묶는 알고리즘
- ✓ “k”는 각 데이터 점들의 서로에 대한 유사성을 기초로 한 고정된 수(k)의 군집을 찾는다는 것을 의미함
- ✓ 각 클러스터간의 거리 차이의 분산을 최소화



K-Means 군집화 과정 (예제)



1번 데이터의 경우 c1 중심점과 가장 가까우므로 노란색으로 바뀐다.

강의 6 : 실습

ML로 VDS 데이터 분류하기

(lab-02-vds-DTC-ML.ipynb)

데이터 가져오기

```
In [1]: import pandas as pd  
import numpy as np
```

```
In [2]: df = pd.read_csv('./daejeon_vds16.csv')  
df.head()
```

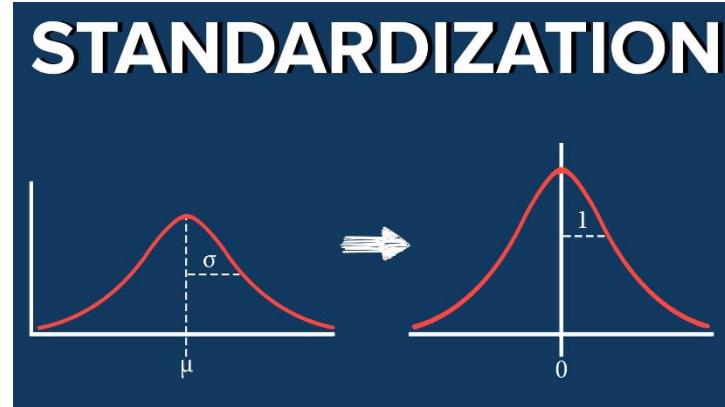
```
In [3]: def get_score(v):  
    if v < 25:  
        score = 'Jam'  
    elif v < 50:  
        score = 'Slow'  
    else :  
        score = 'Fast'  
    return score
```

```
In [4]: df["label_speed"] = df["Speed"].apply(lambda v: get_score(v))  
df.head(5)
```

	Date	ToVol	SmVol	MeVol	LaVol	Speed	Occ.Rate	label_speed
0	2017-04-02 0:00	43	34	9	0	50.3	1.90	Fast
1	2017-04-02 0:05	45	32	13	0	58.9	1.84	Fast
2	2017-04-02 0:10	46	34	12	0	50.6	1.87	Fast
3	2017-04-02 0:15	45	36	9	0	50.9	1.72	Fast
4	2017-04-02 0:20	27	13	13	1	62.2	1.12	Fast

❖ 표준화

- ✓ 피처를 리스케일링하여 피처의 평균이 0이고 분산이 1을 만들어 주는 과정임.
- ✓ 훈련(train) 데이터에는
 - `fit_transform`을 사용하여 정규화하고
- ✓ 테스트(test) 데이터에는
 - `transform`을 적용한다. 그 이유는 `fit_transform`에서 정규화한 조건과 같은 것을 사용하기 위해서이다.



```
In [5]: import matplotlib.pyplot as plt
```

```
In [6]: df['label_speed'].unique()
```

```
Out[6]: array(['Fast', 'Slow', 'Jam'], dtype=object)
```

```
In [7]: feature_cols = ['ToVol', 'Occ.Rate', 'LaVol', 'MeVol']
target_col = 'label_speed'
X = df[feature_cols]
y = df[target_col]
```

2) 출력용 라벨을 머신러닝

- 텍스트를 숫자로 바꾸자
- One-Hot Encoding
- 자연어 처리에는 Embedding을 사용함.
- Word2Vec

```
In [8]: class_dic = {'Jam':0, 'Slow':1, 'Fast':2}
y_ohc = y.apply(lambda z: class_dic[z])
```

```
In [9]: y_ohc.head()
```

```
Out[9]: 0    2
        1    2
        2    2
        3    2
        4    2
Name: label_speed, dtype: int64
```

데이터를 훈련과 테스트로 나누자

- (실전) 데이터를 validation을 포함해서 나눌 수 있다.
- (해보기) 전체 데이터를 train : validation : test = 0.6: 0.2: 0.2로 나누어라

```
In [10]: from sklearn.model_selection import train_test_split  
X_train, X_test, y_train, y_test = train_test_split(X, y_ohc, test_size=0.20, random_state=30)
```

```
In [11]: print(X_train.shape, y_train.shape)  
print(X_test.shape, y_test.shape)
```

```
(6451, 4) (6451,)  
(1613, 4) (1613,)
```

```
In [12]: from sklearn.preprocessing import StandardScaler  
scaler = StandardScaler()  
  
X_train = scaler.fit_transform(X_train)  
X_test = scaler.transform(X_test)
```

머신러닝의 러닝커브를 위한 함수를 정의하자

```
In [13]: from sklearn import metrics  
from sklearn.metrics import plot_roc_curve  
from sklearn.model_selection import ShuffleSplit, learning_curve
```

In [14]:

```
def plot_ml_curve(estimator, title, X, y, ylim=None, cv=None,
                  n_jobs=None, train_sizes=np.linspace(.1, 1.0, 20)):
    plt.figure()
    plt.title(title)
    plt.xlabel("Training examples")
    plt.ylabel("Score")
    train_sizes, train_scores, test_scores = learning_curve(
        estimator, X, y, cv=cv, n_jobs=n_jobs, train_sizes=train_sizes)
    train_mean = np.mean(train_scores, axis=1)
    train_std = np.std(train_scores, axis=1)
    test_mean = np.mean(test_scores, axis=1)
    test_std = np.std(test_scores, axis=1)
    plt.grid()
    plt.plot(train_sizes, train_mean, 'o-', color="r", label="Training score")
    plt.plot(train_sizes, test_mean, 'o-', color="g", label="Cross-validation score")
    plt.legend(loc='best')
    return plt
```

Decision Tree(의사결정나무) 분류를 적용

```
In [15]: from sklearn.tree import DecisionTreeClassifier
```

```
In [16]: model = DecisionTreeClassifier()
```

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

```
pred = model.predict(X_test)
```

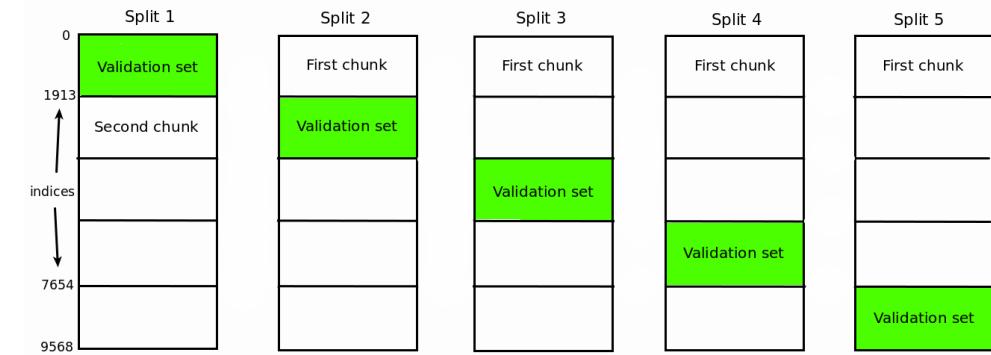
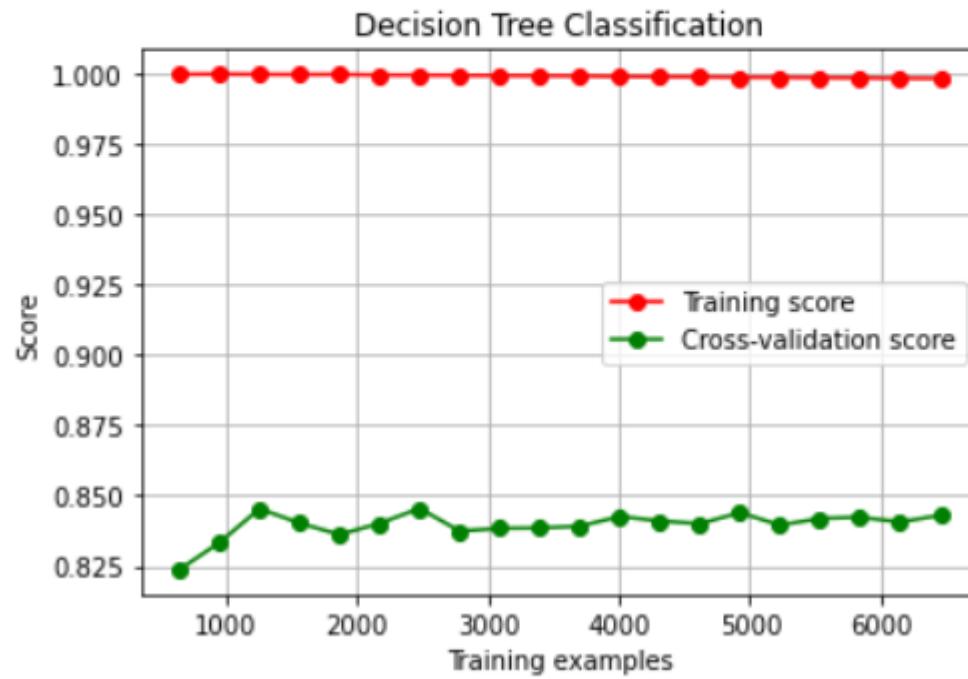
```
metrics.accuracy_score(pred, y_test)
```

```
Out[16]: 0.8245505269683819
```

Decision Tree(의사결정나무) 분류를 적용

```
In [18]: cv = ShuffleSplit(n_splits=5, test_size=0.2)  
plot_ml_curve(model, 'Decision Tree Classification', X, y, cv=cv)
```

Out[18]: <module 'matplotlib.pyplot' from 'C:\Python37\lib\site-packages\W



K-NN 해보기

```
In [19]: from sklearn.neighbors import KNeighborsClassifier
```

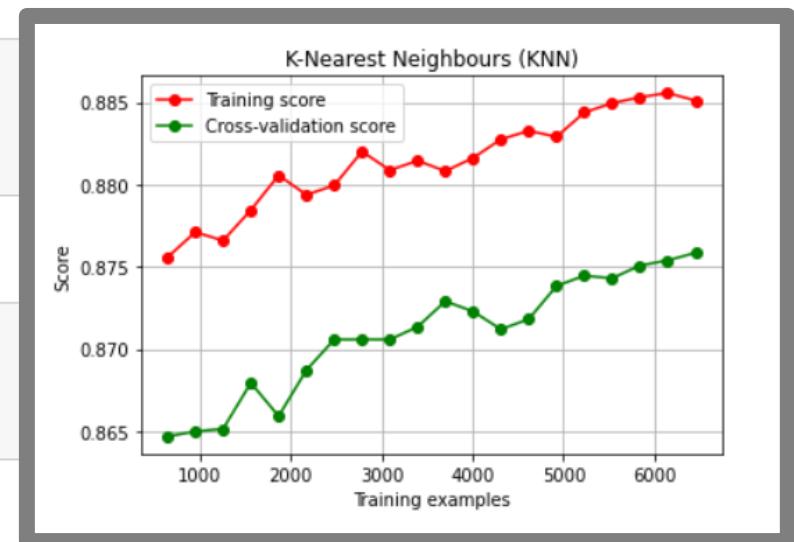
- class sklearn.neighbors.KNeighborsClassifier(n_neighbors=5, *, weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=None)
- n_neighbors : default=5

```
In [20]: model = KNeighborsClassifier(n_neighbors=9)  
model.fit(X_train,y_train)
```

```
Out[20]: KNeighborsClassifier(n_neighbors=9)
```

```
In [21]: pred = model.predict(X_test)  
metrics.accuracy_score(pred,y_test)
```

```
Out[21]: 0.8766274023558587
```



```
In [22]: title = "K-Nearest Neighbours (KNN)"  
cv = ShuffleSplit(n_splits=4, test_size=0.2, random_state=0)  
plot_ml_curve(model, title, X, y, cv=cv, n_jobs=4)
```

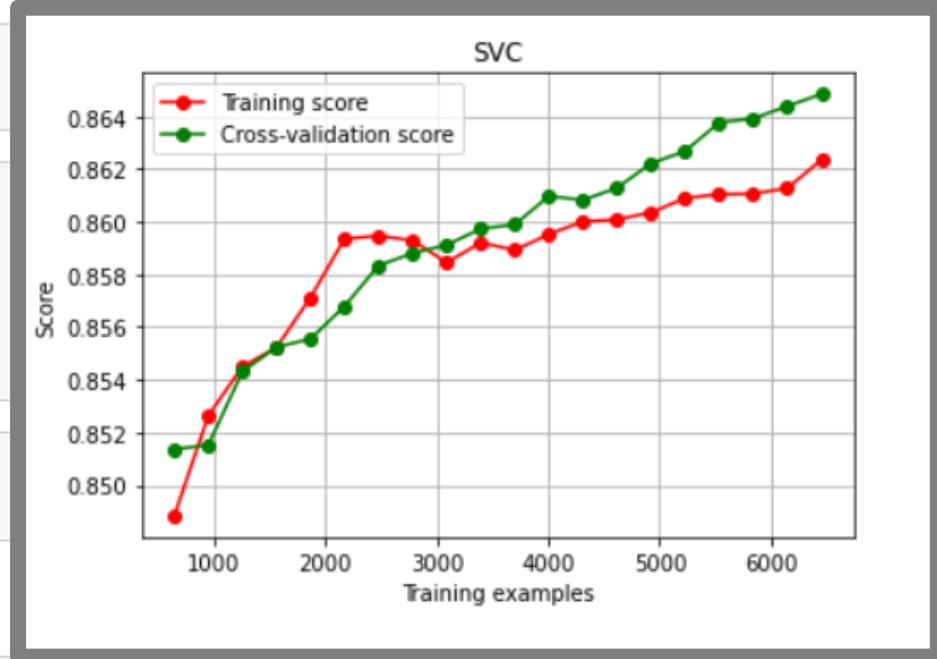
Support Vector Machine Classification 해보기

```
In [24]: from sklearn.svm import SVC
```

```
In [25]: model = SVC()  
model.fit(X_train, y_train)  
pred = model.predict(X_test)
```

```
In [26]: metrics.accuracy_score(pred, y_test)
```

```
Out[26]: 0.8772473651580905
```



```
In [*]: title='SVC'  
cv = ShuffleSplit(n_splits=4, test_size=0.2, random_state=0)  
plot_ml_curve(model, title, X, y, cv=cv)
```

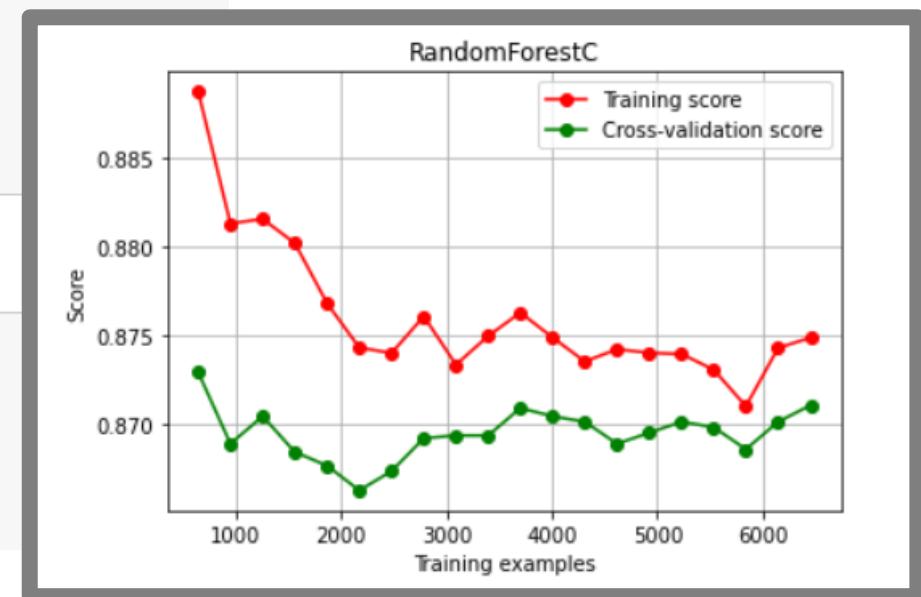
RandomForest

```
In [28]: from sklearn.ensemble import RandomForestClassifier
```

```
In [29]: model = RandomForestClassifier(n_estimators=100, max_depth=3)
model.fit(X_train, y_train)
pred = model.predict(X_test)
metrics.accuracy_score(pred, y_test)
```

```
Out[29]: 0.8605083694978302
```

```
In [30]: title='RandomForestC'
cv = ShuffleSplit(n_splits=4, test_size=0.2, random_state=0)
plot_ml_curve(model, title, X, y, cv=cv)
```



2022

Korea Institute of Science
and Technology Information

TRUST
KISTI

