

데이터 마이닝 소개

학습 목표

- 데이터 마이닝 과정을 소개하고 데이터 마이닝이 무엇인지에 대해 살펴본다.

주요 내용

1. 강의 계획 소개
2. Python 실습 방식
3. 데이터 마이닝
4. 데이터 마이닝 예제

2021년 3월 4일

윤성진



1. 강의 계획



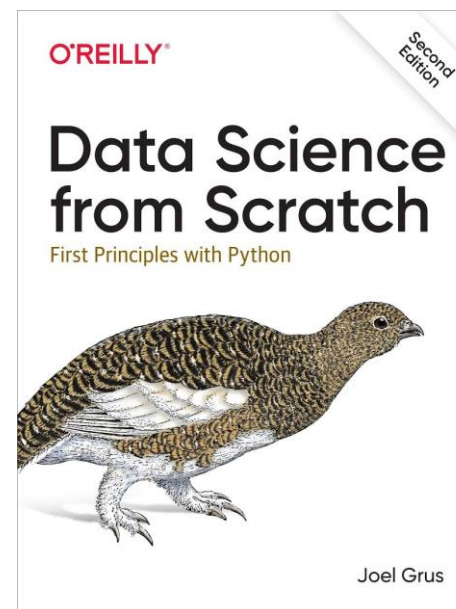
교재

주교재

- **밑바닥부터 시작하는 데이터 과학: 데이터 분석을 위한 파이썬 프로그래밍과 수학·통계 기초 2판,**
 - 조엘 그루스 저/김한결, 하성주, 박은정 역 | 인사이트(insight), 2020년 03월 12일.
- **Data Science from Scratch: First Principles with Python, 2nd Edition,** Joel Grus (2019)
 - 소스 코드 : <https://github.com/joelgrus/data-science-from-scratch>

참고 교재

- **Numpy User Guide**
 - <https://docs.scipy.org/doc/numpy-1.11.0/numpy-user-1.11.0.pdf>
- **Pandas User Guide**
 - <https://pandas.pydata.org/pandas-docs/stable/pandas.pdf>



진도 계획

* 통계적 추론, 자연어처리, 딥러닝, 맵리듀스 제외

No.	날짜	기초 이론 및 라이브러리	머신러닝 알고리즘
1	3/4	데이터 마이닝 소개	
2	3/11	파이썬 (Python)	
3	3/18	데이터 시각화 (Visualization) 선형대수 (Linear Algebra), 넘파이 (Numpy)	
4	3/25	머신러닝 (Machine Learning)	k-최근접 이웃 (k-NN)
5	4/1	확률 (Probability)	나이브 베이즈 (Naive Bayes)
6	4/8	통계 (Statistics)	단순 회귀 (Simple Regression)
7	4/15	경사 하강법 (Gradient Descent)	다중 회귀 (Multiple Regression)
8	4/22	중간고사	
9	4/29		로지스틱 회귀 (Logistics Regression)
10	5/6	데이터 수집 (Getting Data), 판다스 (Pandas)	군집화 (Clustering)
11	5/13	데이터 다루기 (Working with Data)	차원 축소 (Dimension Reduction)
12	5/20		결정 트리 (Decision Tree)
13	5/27		네트워크 분석 (Network Analysis)
14	6/3		추천 시스템 (Recommender Systems)
15	6/17	기말고사	

선수 지식

- **프로그래밍** : 파이썬, 자료구조 그래프 너비우선 탐색 (BFS)
- **확률** : 조건부 확률, 베이즈 정리, 확률밀도함수/누적밀도함수, 베르누이 분포/정규분포
- **정보이론** : 엔트로피, 크로스 엔트로피, 정보획득
- **통계** : 공분산, 상관관계, 최대우도추정
- **선형대수** : 고윳값, 고유벡터, 내적, 투영, 선형 변환, 행렬 분해, 다변수 함수
- **미적분** : 그래디언트, 연쇄법칙, 로그함수/지수함수 미분, 행렬 미분 등

수업 운영 방식

- 강의는 슬라이드 발표 및 Jupyter Notebook 파일로 실습하는 형식으로 진행
 - 이론 설명은 교재 외 내용
 - 실습 코드는 교재 내용
- 실습 환경
 - 언어는 Python
 - 수업 시간에 실습할 수 있도록 Local PC에 환경 구성
 - 교재에서 제공하는 실습 코드는 .py 파일이지만
수업 시간에는 Jupyter Notebook 형태로 진행할 예정

과제 및 평가

- 중간고사 30%, 기말고사 30%
 - eClass 온라인 방식으로 시험
- 출석 10%
 - 전자 출석 방식으로 출석 체크 (2교시 시작할 때 출석 체크 예정)
 - 수업 중 불시 호명 시 불출석한 경우 감점 처리
- 과제물 30%
 - 과제는 5~7회 예정 (알고리즘 별 1회)

2. Python 실습 방식

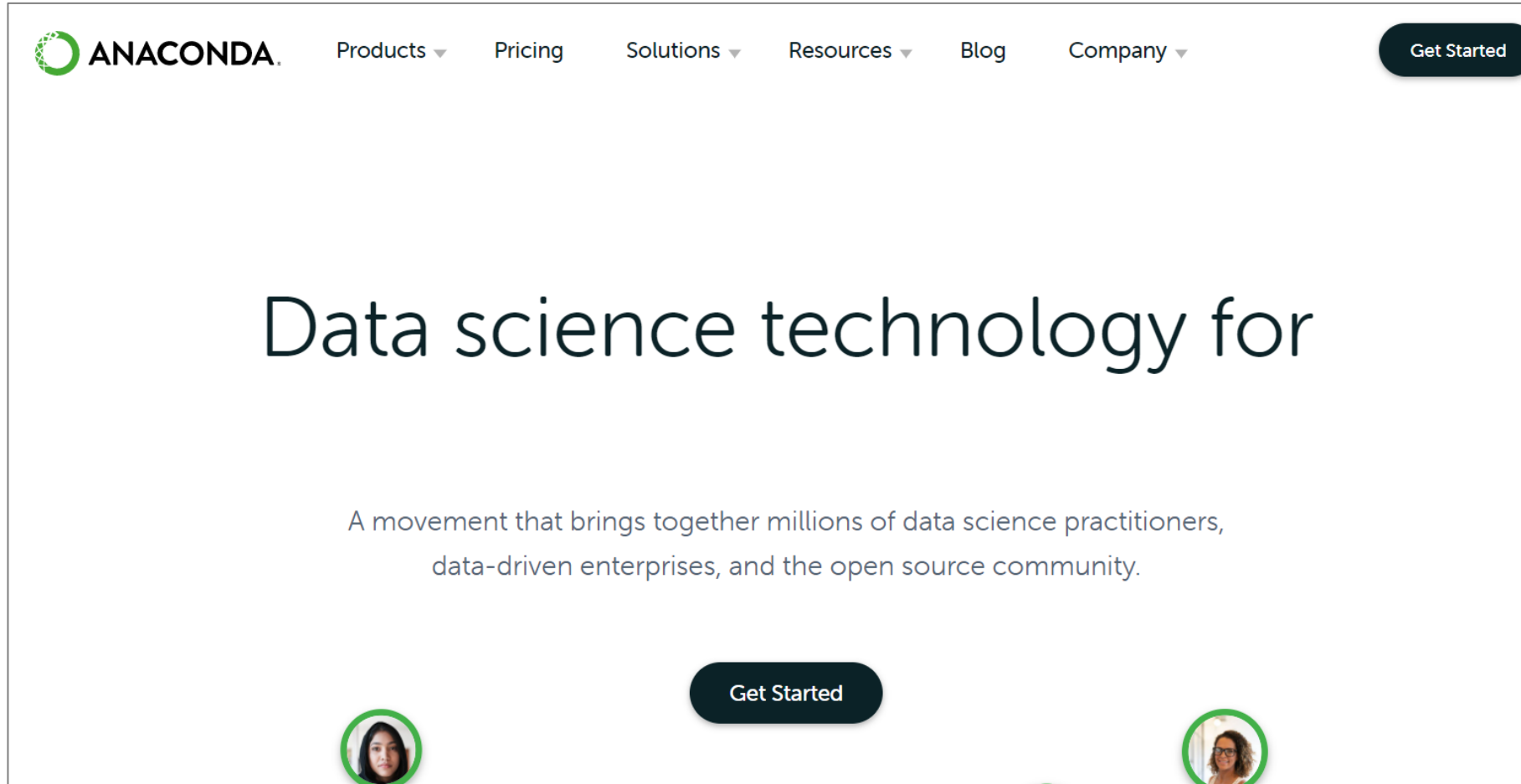


Python 개발 환경



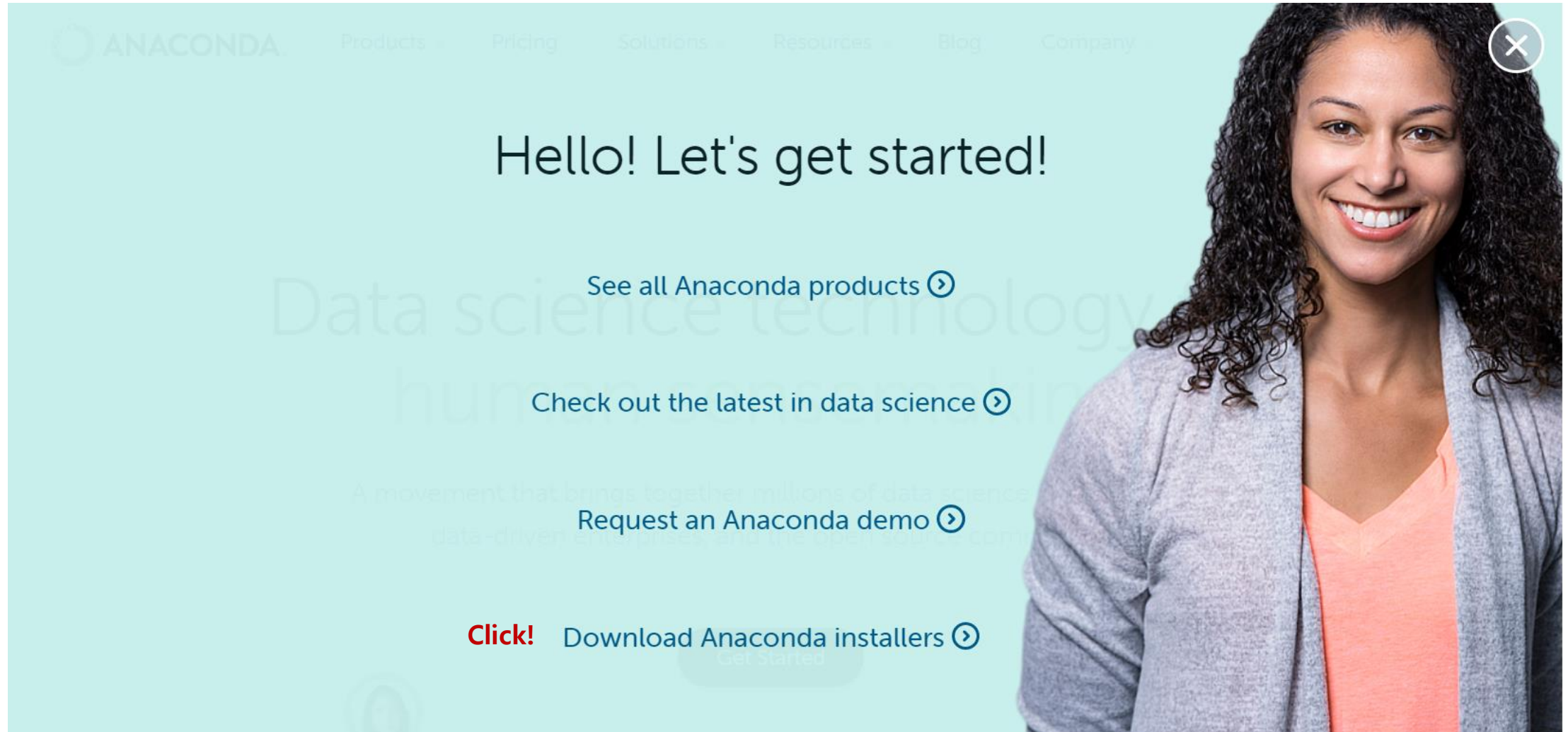
Anaconda 설치

<https://www.anaconda.com/>



Click!




Anaconda 설치



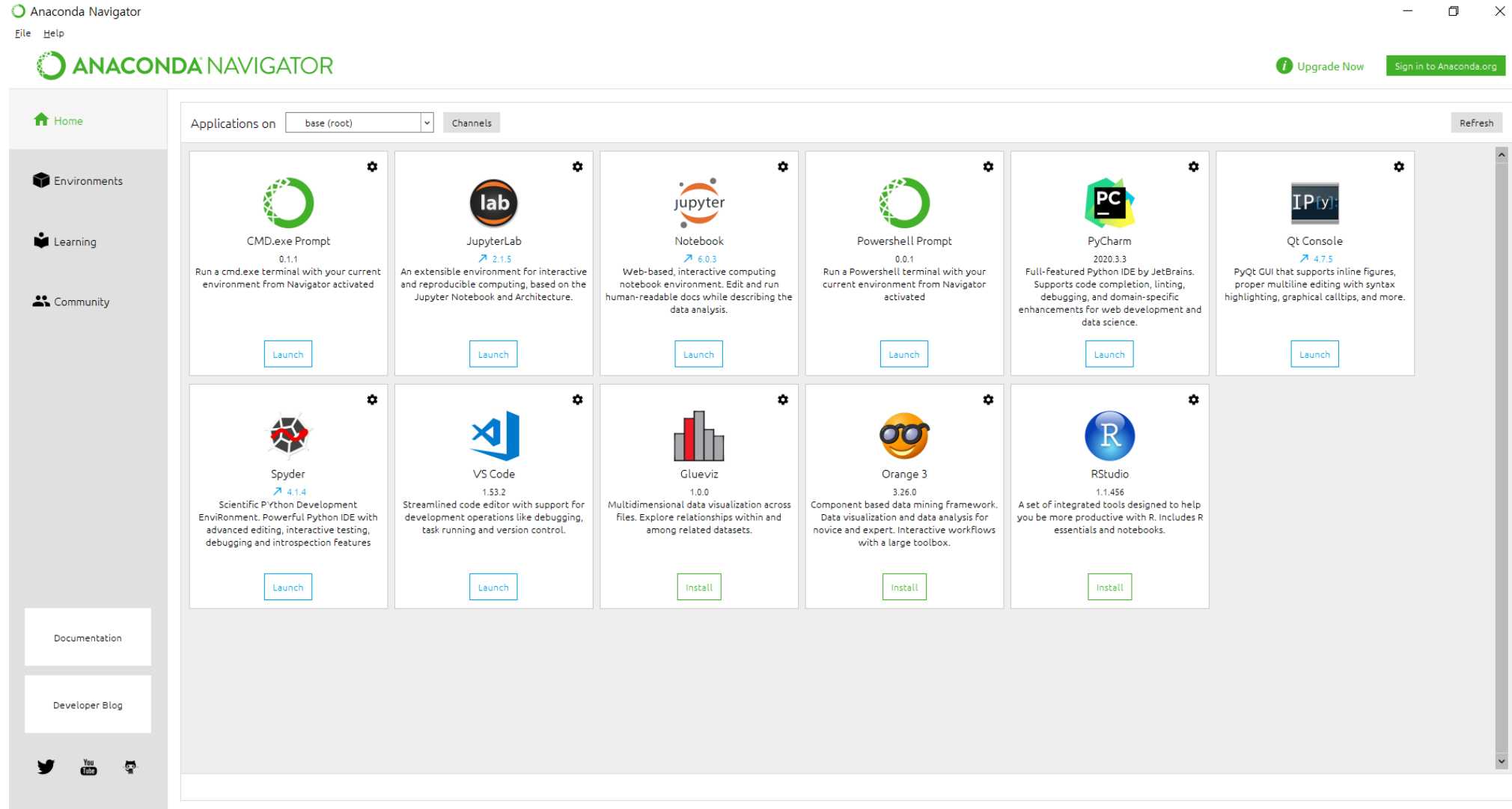
Anaconda 설치

OS에 따라 선택

Anaconda Installers

Windows 	MacOS 	Linux 
Python 3.8	Python 3.8	Python 3.8
Click! 64-Bit Graphical Installer (457 MB)	64-Bit Graphical Installer (435 MB)	64-Bit (x86) Installer (529 MB)
32-Bit Graphical Installer (403 MB)	64-Bit Command Line Installer (428 MB)	64-Bit (Power8 and Power9) Installer (279 MB)

Anaconda Navigator 실행



가상 환경 추가

Anaconda Navigator

File Help

ANACONDA NAVIGATOR

Upgrade Now

Sign in to Anaconda.org

2. 환경 이름 입력 및 Create 버튼 클릭

1. 환경 추가 버튼 Click

Package	Description	Version
enabling anaconda-bundled jupyter extensions		0.1.0
able sphinx theme.		0.7.12
d deployment of anaconda		2020.07
library		1.7.2
d reproducing data science projects		0.8.4
		0.26.2
asn1crypto	Python asn.1 library with a focus on performance and a pythonic api	1.3.0
astroid	A abstract syntax tree for python with inference support.	2.4.2
astropy	Community-developed python library for astronomy	4.0.1.po...
atomicwrites	Atomic file writes.	1.4.0
attrs	Attrs is the python package that will bring back the joy of writing classes by relieving you from the drudgery of implementing object protocols (aka dunder methods).	19.3.0
autopep8	A tool that automatically formats python code to conform to the pep 8 style guide	1.5.3

331 packages available

가상 환경 추가 – Console

환경 생성

```
$ conda create -n data_mining python=3.8
```

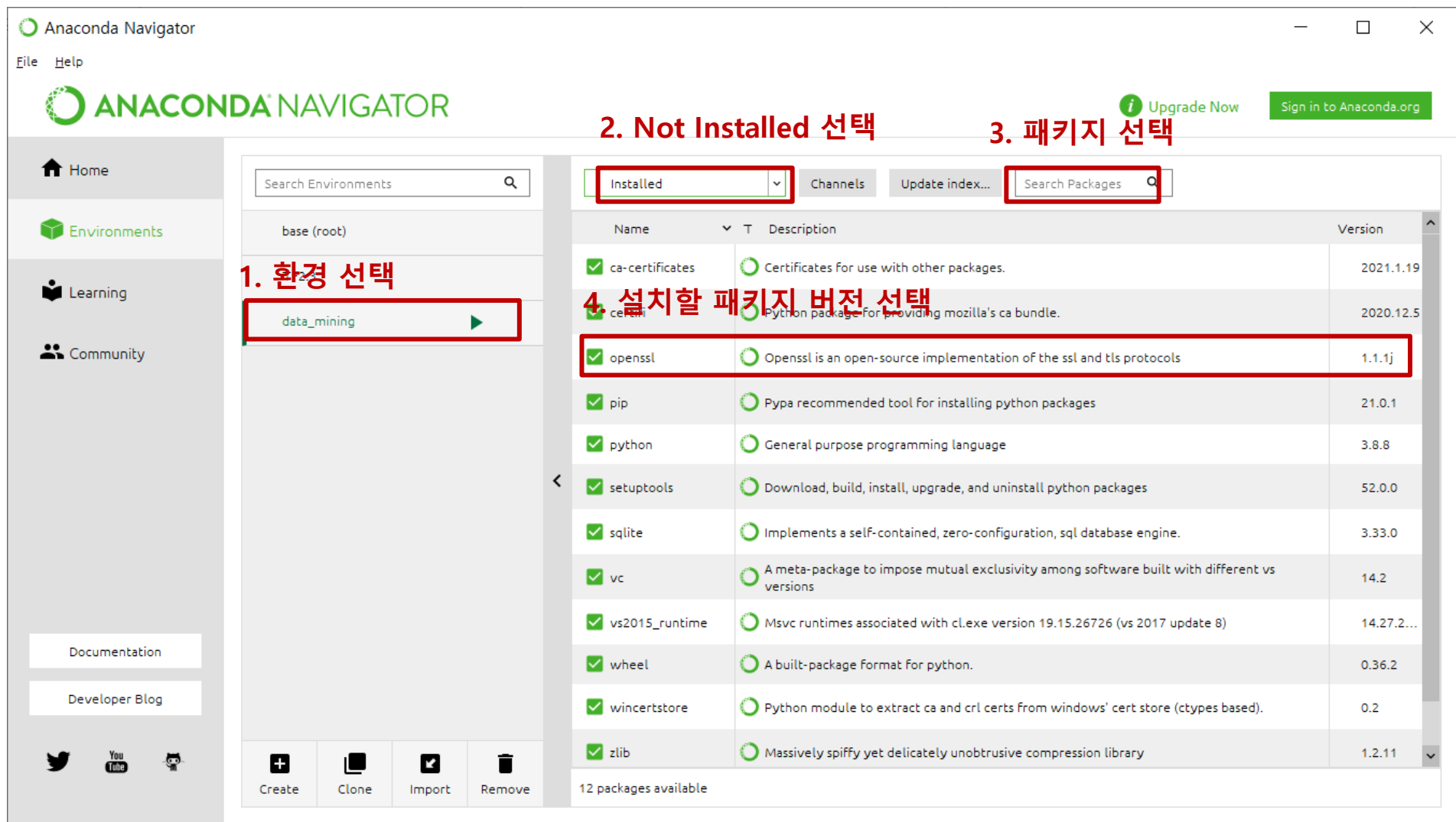
환경 리스트

```
$ conda env list
```

환경 삭제

```
$ conda env remove -n data_mining
```

패키지 설치



1. 환경 선택

2. Not Installed 선택

3. 패키지 선택

4. 설치할 패키지 버전 선택

Name	Description	Version
ca-certificates	Certificates for use with other packages.	2021.1.19
certifi	Python package for providing mozilla's ca bundle.	2020.12.5
openssl	Openssl is an open-source implementation of the ssl and tls protocols	1.1.1j
pip	Pypa recommended tool for installing python packages	21.0.1
python	General purpose programming language	3.8.8
setuptools	Download, build, install, upgrade, and uninstall python packages	52.0.0
sqlite	Implements a self-contained, zero-configuration, sql database engine.	3.33.0
vc	A meta-package to impose mutual exclusivity among software built with different vs versions	14.2
vs2015_runtime	Msvc runtimes associated with cl.exe version 19.15.26726 (vs 2017 update 8)	14.27.2...
wheel	A built-package format for python.	0.36.2
wincertstore	Python module to extract ca and crt certs from windows' cert store (ctypes based).	0.2
zlib	Massively spiffy yet delicately unobtrusive compression library	1.2.11

12 packages available

패키지 설치 – Console

환경 활성화

```
$ activate data_mining
```

패키지 설치

```
pip install numpy
```

```
pip install pandas
```

```
pip install matplotlib
```

pip 대신 conda install을 사용해도 됨

환경 비활성화

```
$ deactivate
```

패키지 설치 확인 – Console

패키지 import 및 버전 확인

```
$ python
>>> import matplotlib as mpl
>>> import numpy as np
>>> import pandas as pd
>>> import sklearn
>>> import sys
>>>
>>> print("python", sys.version)
>>> print(tf.__version__)
2.1.0
>>> for module in mpl, np, pd, sklearn:
>>>     print(module.__name__, module.__version__)
```

Python Console에서 확인

최종 버전으로 업그레이드

```
pip install --upgrade numpy
```

Jupyter Notebook 가상환경 Kernel 설정

가상환경 Kernel 라이브러리 설치

```
$ conda install ipykernel
```

Kernel 확인



가상환경 Kernel 추가

```
$ python -m ipykernel install --user --name [virtualEnv] --display-name "[displayKernelName]"
```

가상환경 Kernel 삭제

```
$ sudo jupyter kernelspec uninstall {kernelname}
```

Jupyter Notebook

\$ jupyter notebook

1. 파일 선택

2. 파일을 만들려면 New 버튼 선택

3. 커널 선택

The screenshot shows the Jupyter Notebook web interface. At the top, there's a header with the Jupyter logo, 'Quit', and 'Logout' buttons. Below the header, there are tabs for 'Files', 'Running', and 'Clusters'. A message says 'Select items to perform actions on them.' Below this is a file browser showing a list of files and folders. The file '1. Introduction.ipynb' is highlighted with a red box. To the right of the file list, there's a 'New' button with a dropdown menu open, showing various kernel options. The 'Data Mining' option is highlighted with a red box. The dropdown menu also shows 'M2Det', 'Python 3', 'TF2.1', 'TF2.3', 'TensorFlow CPU', and 'Other: Text File', 'Folder', 'Terminal'.

Name	Size	Last Modified
0		
backup		
code		
ml-100k		
scratch		
spam_data		
1. Introduction.ipynb		
10. Working_with_Data.ipynb		
13. NaiveBayes.ipynb		
17. Decision Tree.ipynb	14.1 kB	6일 전
20. Clustering.ipynb	215 kB	5일 전
22. Network Analysis.ipynb	17.1 kB	4일 전
23. Recommender Systems.ipynb	49.1 kB	18시간 전
8. GradientDescent.ipynb	22.9 kB	13일 전

파일 별 커널 설정

The screenshot displays the JupyterLab web interface. At the top, the title bar shows 'jupyter 1. Introduction' and 'Last Checkpoint: 11시간 전 (autosaved)'. On the right, there is a 'Logout' button and a 'Python 3' kernel status indicator with a circular refresh icon. Below the title bar is a menu bar with 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', 'Widgets', and 'Help'. The 'Kernel' menu is open, showing options: 'Interrupt', 'Restart', 'Restart & Clear Output', 'Restart & Run All', 'Reconnect', 'Shutdown', and 'Change kernel'. The 'Change kernel' option is highlighted with a red box, and a sub-menu is open showing available kernels: 'Data Mining', 'M2Det', 'Python 3', 'TF2.1', 'TF2.3', and 'TensorFlow CPU'. A red box also highlights the 'Python 3' kernel status indicator in the top right. Red text annotations are present: '실행 중인 커널' (Running kernel) next to the 'Python 3' status, and '커널을 변경하려면 Kernel 메뉴 클릭' (Click the Kernel menu to change the kernel) next to the 'Change kernel' option in the menu.

1장. 데이터

예제 1. 핵심 인

사용자 목록 (그래프 노드)

```
In [2]: users = [
    { "id": 0, "name": "Hero" },
    { "id": 1, "name": "Dunn" },
    { "id": 2, "name": "Sue" },
    { "id": 3, "name": "Chi" },
    { "id": 4, "name": "Thor" },
    { "id": 5, "name": "Clive" },
    { "id": 6, "name": "Hicks" },
    { "id": 7, "name": "Devin" },
    { "id": 8, "name": "Kate" },
    { "id": 9, "name": "Klein" }
]
```

3. 데이터 마이닝



Data is the New Oil



Clive Humby,
a British mathematician and data
science entrepreneur

"Data is the New Oil"

"Data is the new oil. It's valuable, but if unrefined it cannot really be used. It has to be changed into gas, plastic, chemicals, etc to create a valuable entity that drives profitable activity; so must data be broken down, analyzed for it to have value."

- Clive Humby, 2006 -

<https://towardsdatascience.com/is-data-really-the-new-oil-in-the-21st-century-17d014811b88>

AI Is the New Electricity



Andrew Ng,
A Founder of DeepLearning.AI, General
Partner at AI Fund, Chairman and Co-
Founder of Coursera, and an Adjunct
Professor at Stanford University.

"AI Is the New Electricity"

"Just as electricity transformed almost everything 100 years ago, today I actually have a hard time thinking of an industry that I don't think AI will transform in the next several years," Ng says.

- Andrew Ng, 2016 -

<https://www.gsb.stanford.edu/insights/andrew-ng-why-ai-new-electricity>

기계 학습 (Machine Learning)

Artificial Intelligence

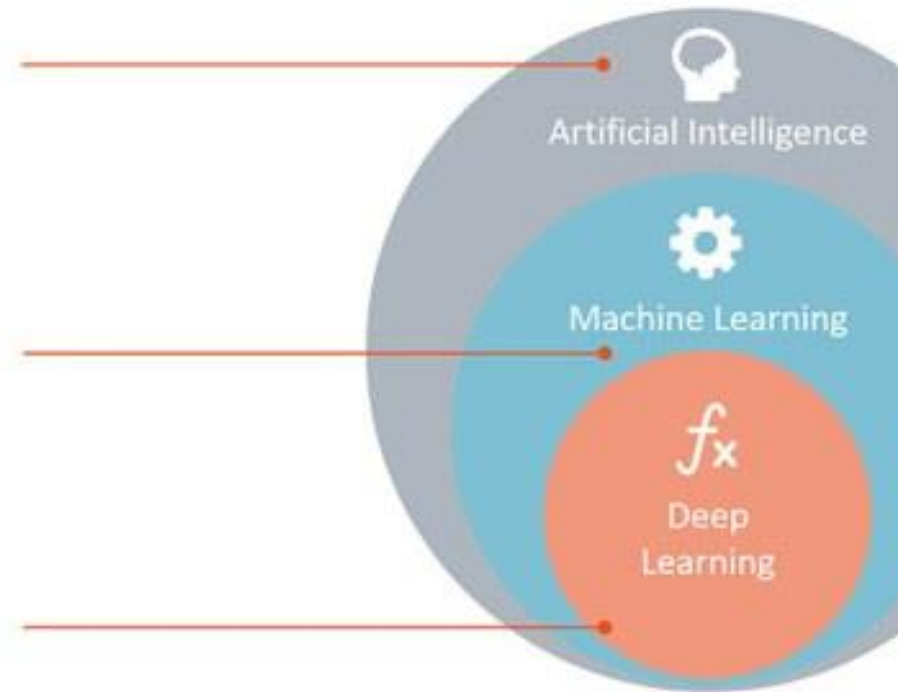
Any technique which enables computers to mimic human behavior.

Machine Learning

Subset of AI techniques which use statistical methods to enable machines to improve with experiences.

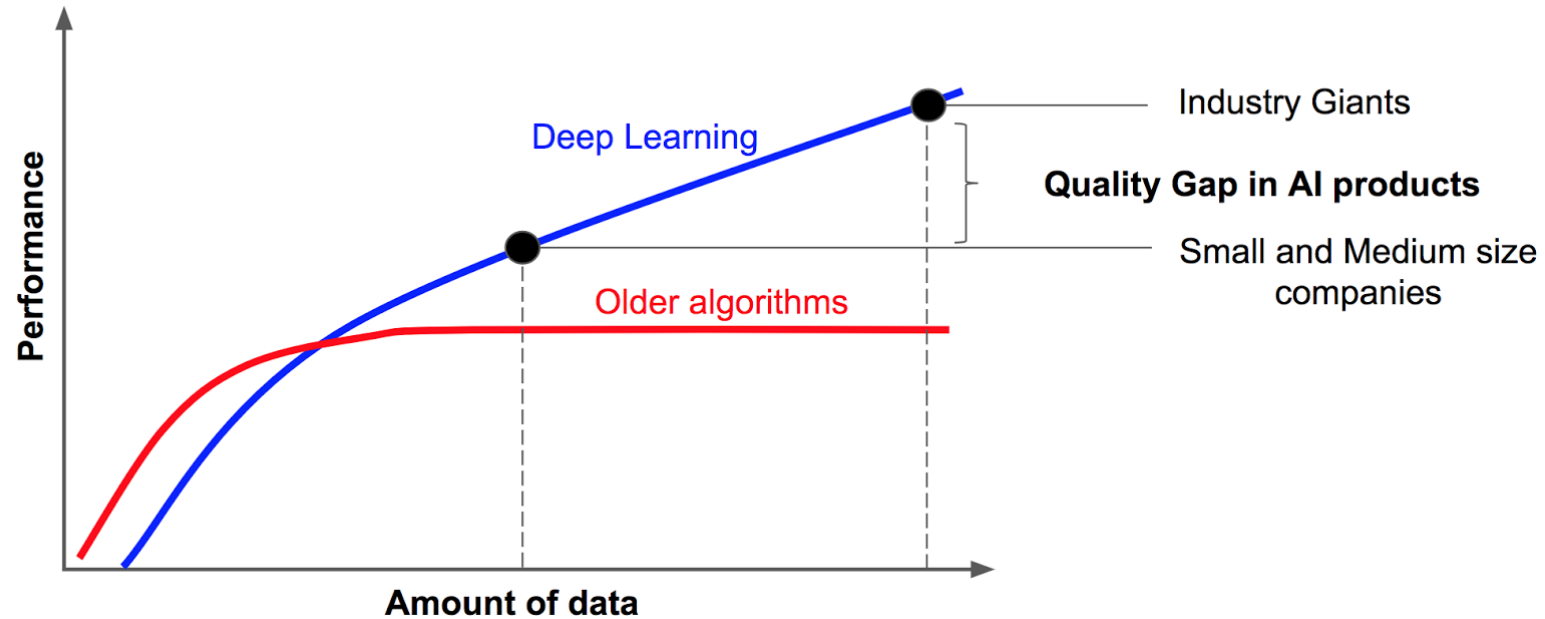
Deep Learning

Subset of ML which make the computation of multi-layer neural networks feasible.



<https://www.kdnuggets.com/2017/07/rapidminer-ai-machine-learning-deep-learning.html>

데이터양과 기업 경쟁력



Conclusion 0: AI products need data.

Conclusion 1: the more data we have — the smarter AI will be.

Conclusion 2: industry giants have much more data than others.

Conclusion 3: quality gap in AI products is defined by amount of data.

<https://hackernoon.com/%EF%B8%8F-big-challenge-in-deep-learning-training-data-31a88b97b282>

빅데이터는 어디에서 생성되는가?

소셜 미디어

(Social data)

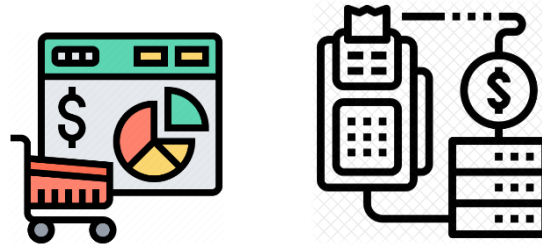


소셜 미디어 데이터

- 소셜 미디어의 데이터
- 좋아요, 트윗 및 리트윗, 댓글, 사진/동영상 데이터
- 고객 행동분석, 감성 분석, 마켓분석에 사용

거래 데이터

(Transaction data)

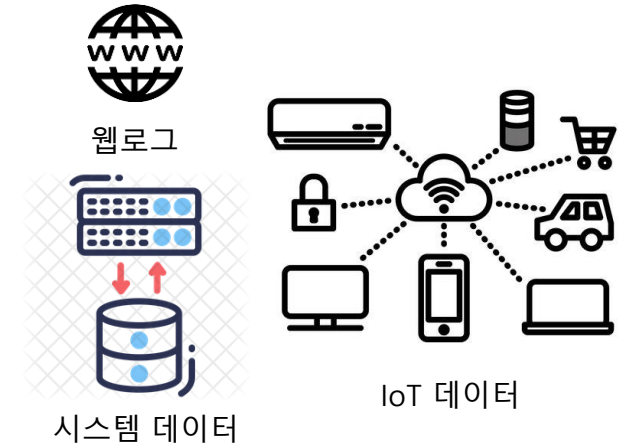


오프라인 거래 데이터 온라인 거래 데이터

- 온라인 오프라인에서 발생하는 일일 거래 데이터
- 건적, 주문, 결제, 재고 상태, 배송, 영수증

기계 데이터

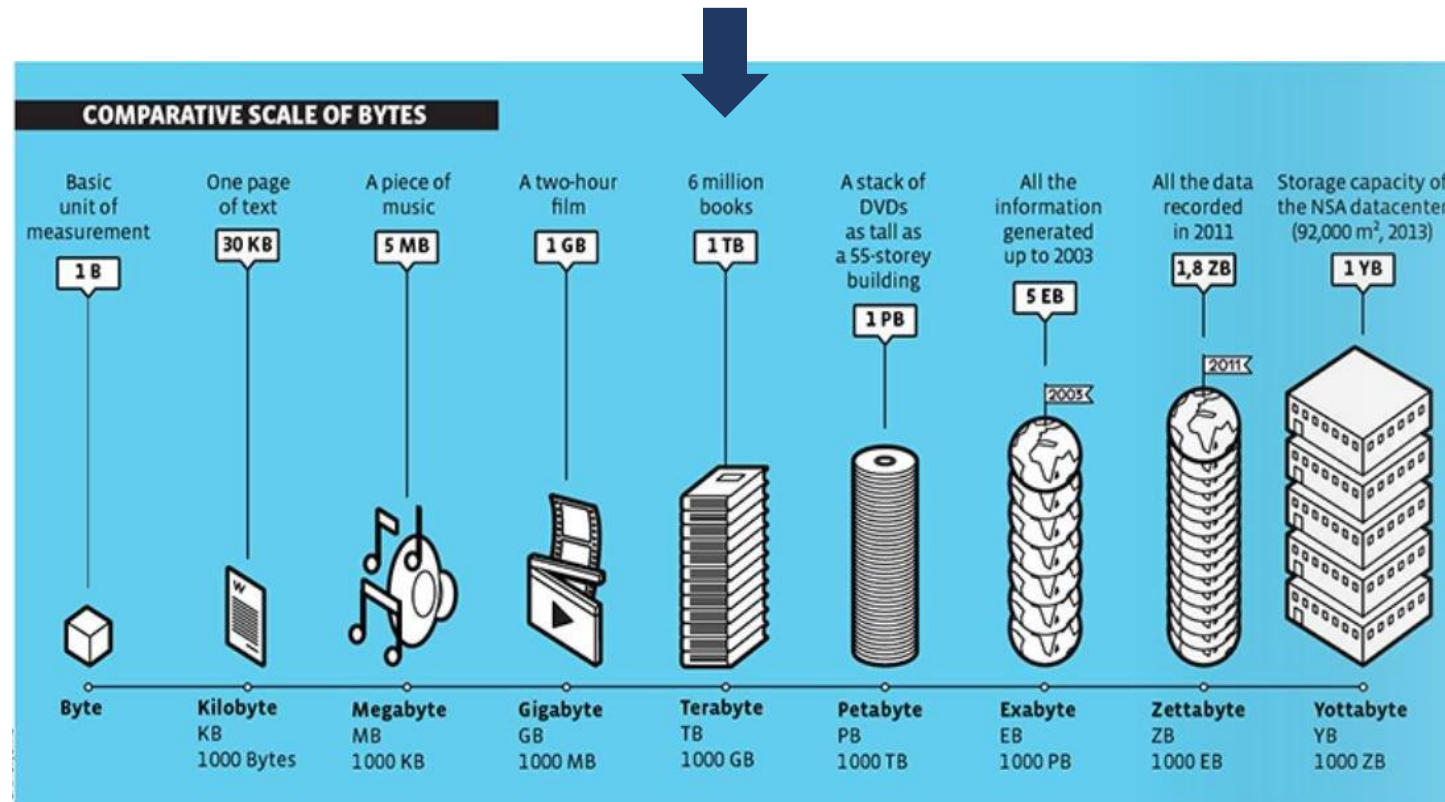
(Machine data)



- 시스템/네트워크 로그 데이터, 센서/IoT 데이터, 사용자 행동을 추적하는 웹로그 등의 데이터
- 의료 기기, 스마트 미터, 도로 카메라, 위성, 게임 및 사물 인터넷과 같은 센서 데이터

데이터 크기

Wal-Mart: 20 million transactions per day in 2003 – 10TB DB



<https://www.haikudeck.com/how-is-data-measured--uncategorized-presentation-vA114nosVT>

데이터 마이닝 (Data Mining)

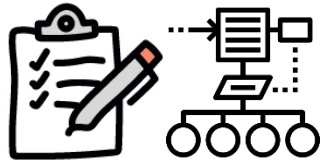
데이터 마이닝 (Data Mining)

대용량 데이터에 있는 데이터의 **관계, 패턴, 규칙** 등을 탐색하고
모델로 만들어서 유용한 지식을 추출하는 과정

데이터베이스 기술 (Database Technology)



이론, 알고리즘 (Theory/Algorithm)



머신 러닝 (Machine Learning)



시각화 (Visualization)

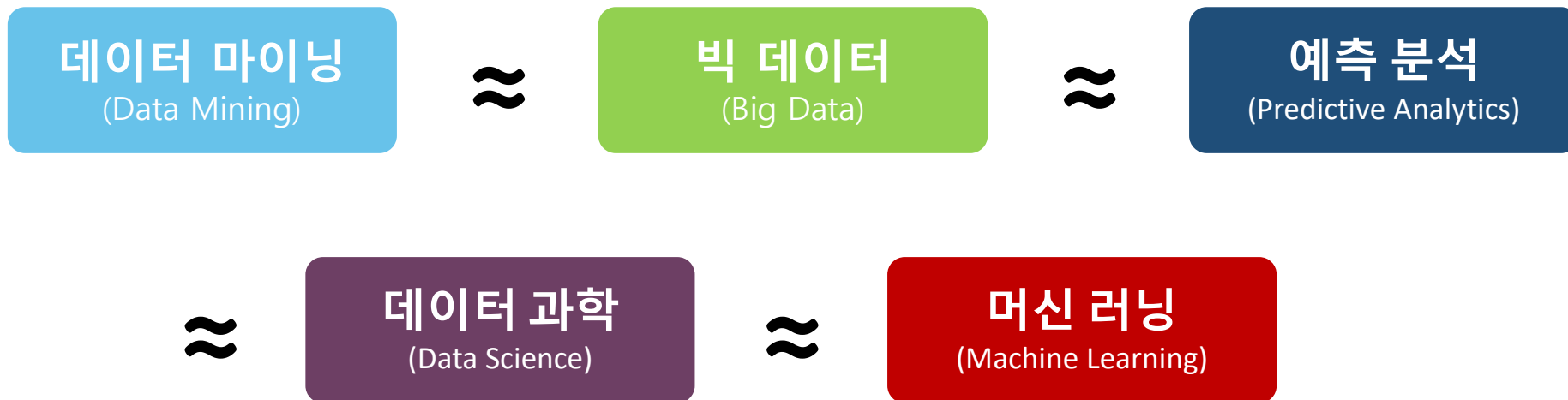


통계 (Statistics)



데이터 마이닝 (Data Mining)

점점 영역 간에 경계가 모호해지고 있음



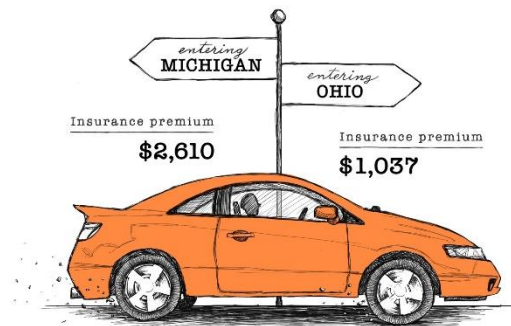
데이터 마이닝 ≠ 머신 러닝

1. 데이터 마이닝의 목표를 명확히 기술할 수 있다면 머신 러닝은 경쟁력이 없어진다.



- WhizBand! Labs라는 스타트업에서는 머신 러닝을 이용해서 웹에 올려진 이력서를 찾아서 데이터 베이스를 구축하고자 함
- 하지만 머신 러닝을 이용한 알고리즘은 사람이 직접 제작한 알고리즘보다 더 좋은 성능을 보이지 못하였고 회사는 사라지게 되었다.
- 왜냐하면, 사람들은 이력서에 어떤 내용이 있는지 잘 알고 있기 때문에 머신 러닝을 이용할 이점을 찾지 못했기 때문이다.

2. 모델이 정확하더라도 결과를 설명할 수 없다면 사용하기 어려운 경우가 있다.



- 자동차 보험 회사에서 운전자의 위험도에 따라 보험료를 계산하는 모델을 만들려고 한다면
- 운전자의 위험도가 어떻게 산정되었고 그에 따라 보험료가 어떻게 책정되었는지 설명할 수 있어야 하는데, 딥러닝과 같은 머신 러닝 방법들은 모델이 왜 그런 예측을 했는지에 대한 설명을 하기가 매우 어렵다.

데이터 마이닝 단계

목표 정의



- 비즈니스 목표 설정

데이터 소스



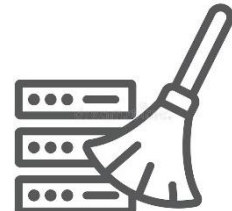
- CRM, ERP 등 내부 시스템
- 소셜 미디어, 거래 데이터 등

데이터 수집 및 저장



- 대용량 데이터베이스
- 데이터 파일

데이터 처리



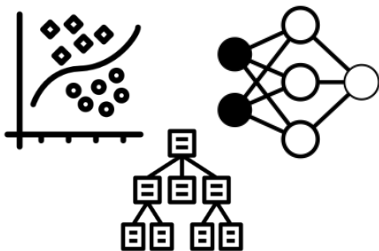
- 데이터 정제 & 통합

데이터 탐색



- 데이터에 대한 이해와 직관
- 개별 변수 분포 및 요약 통계량
- 변수 간 상관성 및 규칙

데이터 마이닝



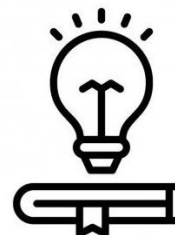
- 모델 선정 및 구축
- 모델 실행 및 성능 평가

결과 보고 및 해석



- 결과에 대한 분석

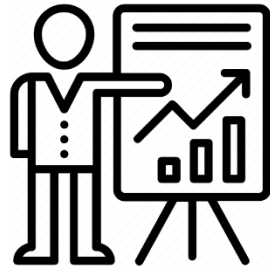
의사 결정



- 의사 결정에 따른 실행

데이터 마이닝 방법

기술 방법 (Descriptive methods)



- 사람이 해석할 수 있는 방식으로 과거 또는 현재의 상황을 파악할 수 있도록 데이터를 요약하거나 패턴을 식별하는 방식
- 요약 (Summarization), 특징 추출 (Feature Extraction) 군집화 (Clustering), 연관 규칙 (Association Rule), 시각화 (Visualization)

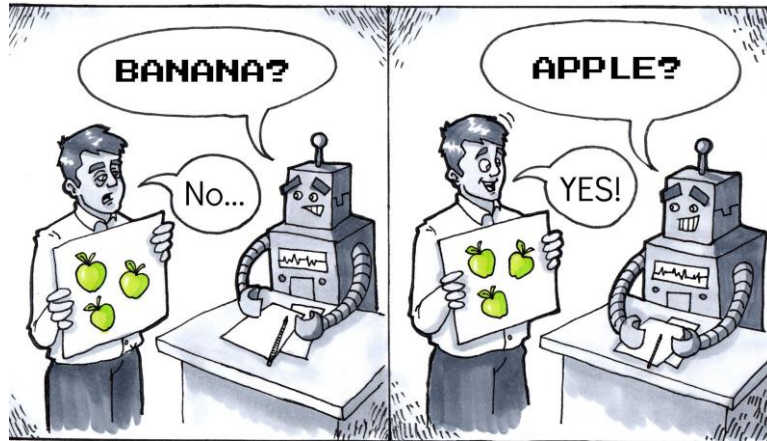
예측 방법 (Predictive methods)



- 미래 또는 알지 못하는 변수 값을 예측하는 방식
- 분류 (Classification), 회귀 (Regression)

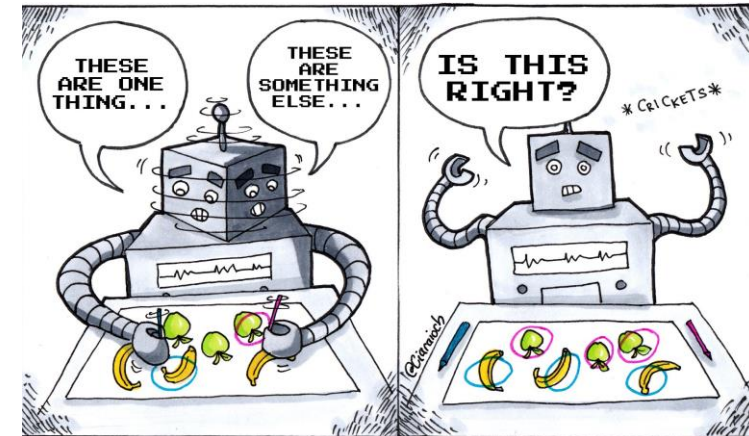
머신 러닝 방법

지도 학습 (Supervised Learning)



- 지도가 들어가는 학습 방식
- 입력 값에 대한 예측 값의 정답을 알려주어 입력과 출력의 관계를 학습하도록 하는 방식

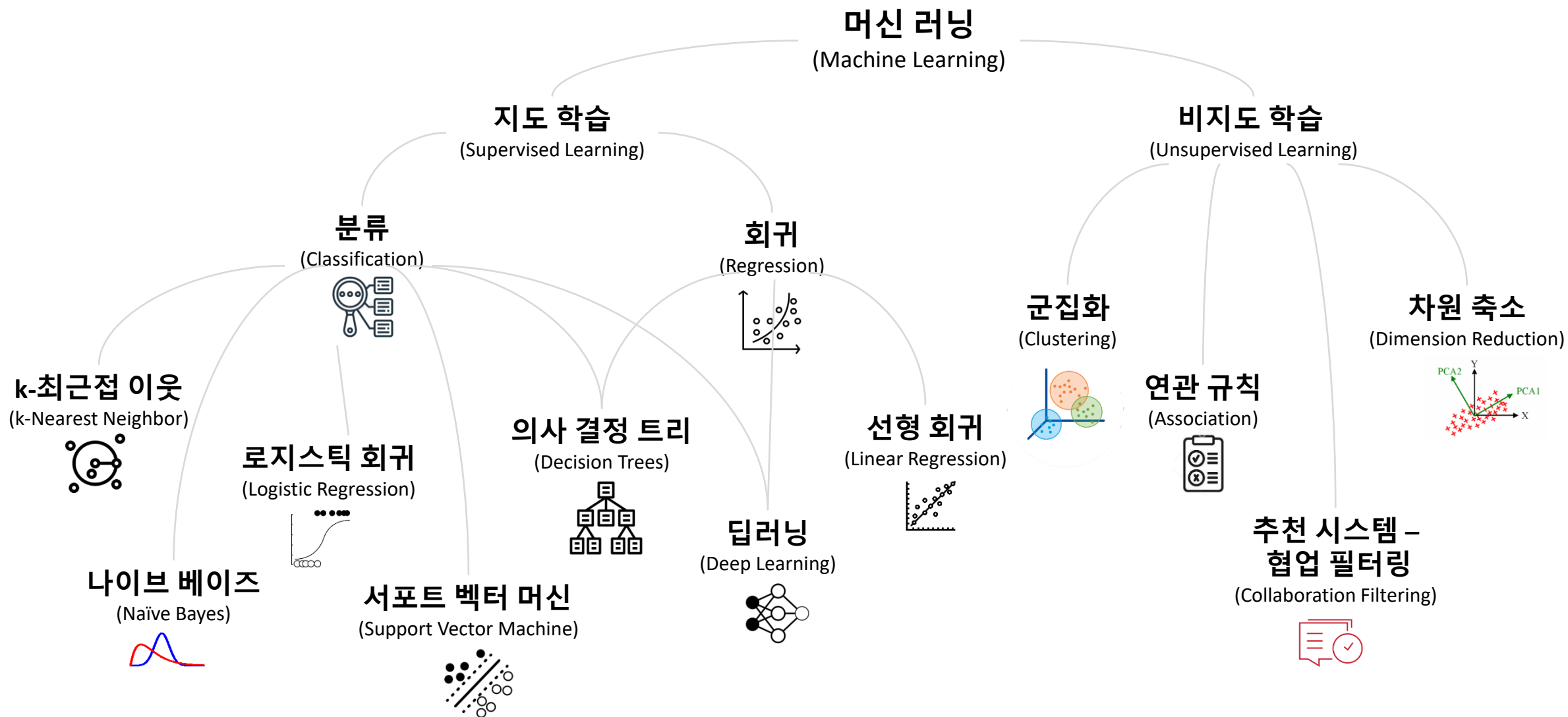
비지도 학습 (Unsupervised Learning)



- 순수하게 데이터만으로 내재적인 정보와 규칙을 학습하는 방식

그림 : https://twitter.com/athena_schools/status/1063013435779223553/photo/1

머신 러닝 방법



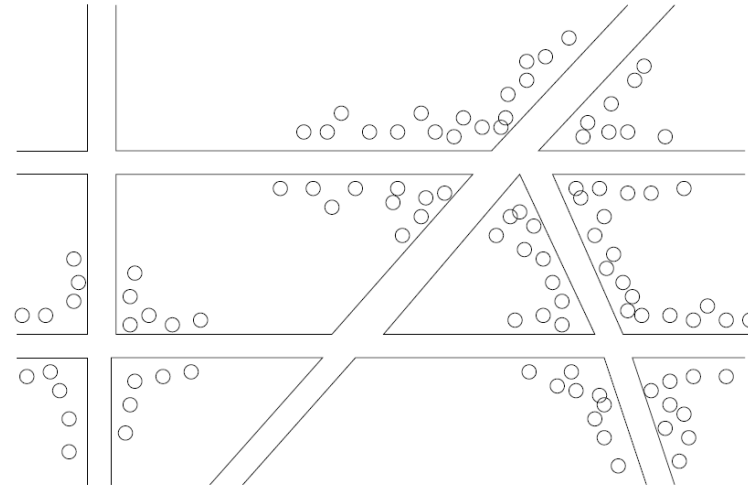
런던 콜레라 대유행

1854 Broad Street cholera outbreak



A COURT FOR KING CHOLERA.

도시 지도에 콜레라 환자의 위치를 추적



- 1854년 영국에서 콜레라 대유행이 있을 때, John Snow라는 의사가 **도시 지도에 콜레라 감염자를 나타내는 지도를 그려 봄**
- 교차로에 주변에 콜레라 감염자가 많은 것을 확인하고 원인이 근처에 있는 물 펌프가 콜레라에 오염된 것으로 추정하게 됨. 반면, 교차로에서 먼 물 펌프의 물을 먹는 사람들은 콜레라에 걸리지 않음을 확인
- 물 펌프를 사용하지 못하도록 만들자 콜레라 환자가 줄어드는 것을 확인
- 당시 공기를 통해 콜레라가 전파된다고 믿고 있었는데 **군집화를 통해 오염된 물이 전염원임을 규명한 사례**

https://www.wikiwand.com/en/1854_Broad_Street_cholera_outbreak

월마트 구매 패턴 분석

- 미국의 할인점 월마트(Wall Mart)에서 매장내의 상품들과 **고객들의 구매패턴의 연관성**을 발견하기 위하여 연관성 분석 알고리즘을 사용
- 젊은 아빠들이 수요일 저녁에 기저귀를 사면서 맥주를 같이 사는 경향이 있음이 밝혀짐
- 기저귀와 맥주를 가까이 배치하여 매출이 증가



그림 : <http://www.munhwa.com/news/view.html?no=20100319010312240460020>

2012년 Obama 대통령 선거



<https://m.blog.naver.com/businessinsight/221073735549>

- **마이크로 타겟팅**(microtargeting)을 선거 전략에 활용해서 2012년 대통령 재선에서 승리
- 유권자 별로 **설득가능점수**(Persuadability Score)를 모형화해서 후보의 지지율 확보에 활용
- 유권자의 **페이스북**과 **인터넷 사용기록** 추적
 - **페이스북**에서 좋아요를 누르면 1) 어느 대학을 나왔는지 2) 어떤 성향의 언론에 우호적인지 3) 어느 분야에 종사하며 4) 어떤 이슈에 민감하게 반응하는지 추적 5) 친구관계에 있는 회원 정보도 수집
 - 이메일 수신자 그룹에 가입한 유권자가 **홍보 웹페이지**에 접속하면 마우스를 클릭하는 모든 기록을 저장해서 1) 어떤 정책에 관심을 보이는지 2) 정책에 대한 맞춤형 페이지를 통해 세세한 정보 취득
- 800개 메시지와 1500개의 **이메일을 변형해서 빈번하게 메일을 발송**하고 설득가능점수를 평가해서 **다른 접근 전략을 수립**

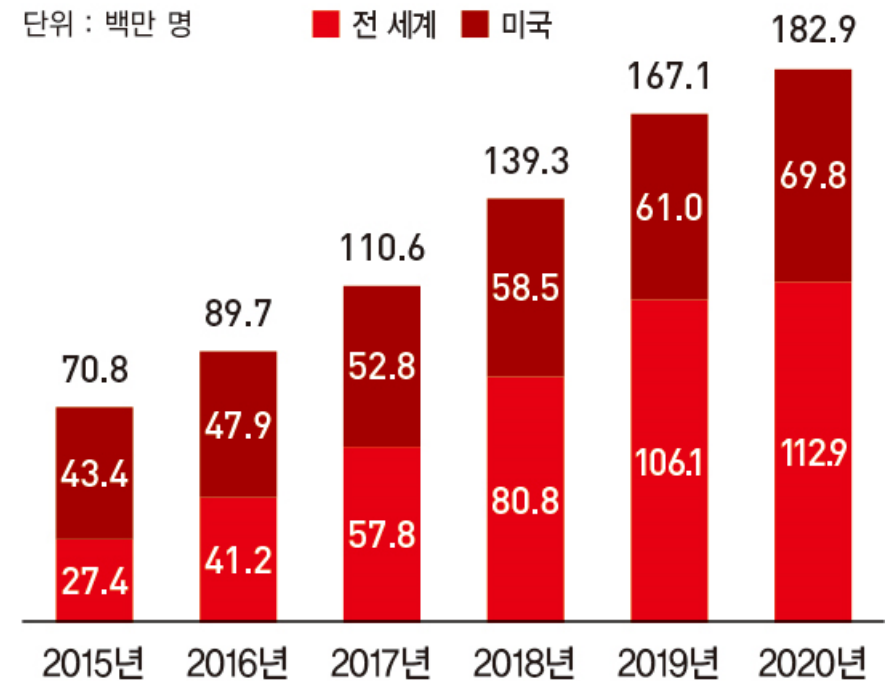
Netflix 추천시스템

- ‘넷플릭스 프라이즈(Netflix Prize)’는 넷플릭스 사용자들의 영화 **별점 데이터**를 가지고 2006년 10월부터 2009년 7월까지 약 3년에 걸쳐 이어진 영화 평가 데이터 예측 대회
- 이 대회를 통해 대표적인 **추천 알고리즘인 ‘협업 필터링 (collaborative filtering)’**이 발전하였고 특히 우승팀의 SVD를 활용한 **SVD++**는 이후 굉장히 많은 분야에서 활용됨
- 당시 대회의 우승 상금은 US \$1,00만 달러 (약11억)
- 현재 추천 알고리즘을 담당하는 개발자만 800명이 넘음
- **고객들의 ‘시청 패턴’에 대한 세세한 분석 및 그룹화**
 - 같은 그룹 내의 고객들이 1) 어떤 콘텐츠를 시청했는지, 2) 이 콘텐츠는 어떤 인물들이 등장하며 3) 어떤 스토리를 갖고 있는지 등을 분석한 결과가 **콘텐츠 추천**의 밑바탕이 된다.
 - 넷플릭스는 1) 시청 콘텐츠의 유사성은 물론 2) **재생 중 되돌리거나 빨리 감기 한 지점까지 매우 세분화해** 그룹을 만드는 것으로 알려져 있음
 - **모든 콘텐츠를 ‘태그화’**

넷플릭스 구독자 수 증가

단위 : 백만 명

■ 전 세계 ■ 미국



자료 : 넷플릭스 · 스탯ISTA

<https://magazine.hankyung.com/business/article/202005206677b>

4. 데이터 마이닝 예제

예제 1. 핵심 인물 찾아라



예제 1. 핵심 인물 찾아라

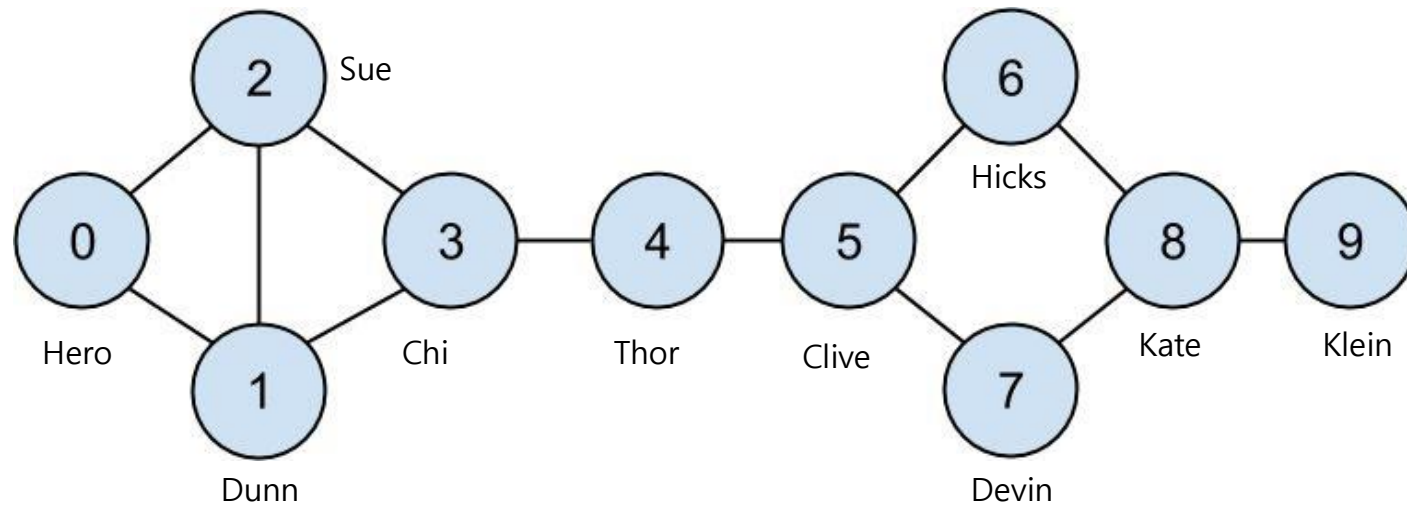
조직의 핵심 인물을 찾아라!



예제 1. 핵심 인물 찾아라

10명의 데이터 사이언티스트가 있고 이들의 친구 관계는 다음과 같다.

데이터 사이언티스트 중 핵심 인물은 누구인가?



핵심 인물이란 연결이 가장 많은 사람일까?

예제 1. 핵심 인물 찾아라

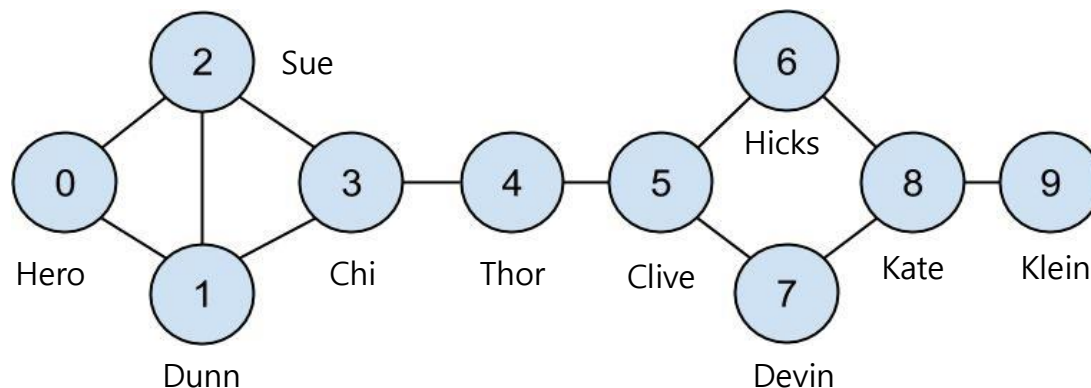
가장 친구가 많은 사람을 찾자!

사용자 리스트

```
users = [  
  { "id": 0, "name": "Hero" },  
  { "id": 1, "name": "Dunn" },  
  { "id": 2, "name": "Sue" },  
  { "id": 3, "name": "Chi" },  
  { "id": 4, "name": "Thor" },  
  { "id": 5, "name": "Clive" },  
  { "id": 6, "name": "Hicks" },  
  { "id": 7, "name": "Devin" },  
  { "id": 8, "name": "Kate" },  
  { "id": 9, "name": "Klein" }  
]
```

- 각 사용자 별로 { "id" : 아이디, "name" : 이름 }
 딕셔너리로 구성

주어진 데이터는 사용자 목록과 친구 관계



친구 관계 (에지 리스트)

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

예제 1. 핵심 인물 찾아라

그래프 자료 구조를 에지 리스트에서 인접 리스트 형태로 변경

친구 관계 (인접 리스트)

```
# Initialize the dict with an empty list for each user id:
friendships = {user["id"]: [] for user in users}

# And loop over the friendship pairs to populate it:
for i, j in friendship_pairs:
    friendships[i].append(j) # Add j as a friend of user i
    friendships[j].append(i) # Add i as a friend of user j
```

- friendships : 각 노드에 연결된 에지의 반대편 노드를 리스트로 저장
- 각 에지 (i,j)에 대해 노드 i에는 j가 추가되고, 노드 j에는 i가 추가됨

예제 1. 핵심 인물 찾아라

먼저, 평균 친구 수를 계산해보자.

User의 친구 수 세기

```
def number_of_friends(user):  
    """How many friends does _user_ have?"""  
    user_id = user["id"]  
    friend_ids = friendships[user_id]  
    return len(friend_ids)
```

- user : { "id" : 아이디, "name" : 이름 } 딕셔너리
- 사용자 id를 기준으로 친구 리스트를 가져옴
- 친구 리스트의 길이를 반환

모든 사람의 친구 수 개수 세기

```
total_connections = sum(number_of_friends(user)  
                        for user in users)      # 24  
  
assert total_connections == 24
```

- 각 노드 별로 연결된 에지 수를 세서 합산함

예제 1. 핵심 인물 찾아라

평균 친구 수 세기

```
num_users = len(users) # length of the users list
avg_connections = total_connections / num_users # 24 / 10 == 2.4

assert num_users == 10
assert avg_connections == 2.4
```

- 전체 에지 수를 사용자 수로 나눠서 평균 에지 수를 셈

사용자 수는 10명이고 평균 에지 수는 2.4

예제 1. 핵심 인물 찾아라

가장 많은 친구를 가진 사람을 찾아보자.

사용자 별로 친구 수를 세서 리스트로 만듦

```
# Create a list (user_id, number_of_friends).
num_friends_by_id = [(user["id"], number_of_friends(user))
                      for user in users]
```

- (사용자 id, 친구 수) 리스트 생성

친구가 많은 순서대로 정렬

```
num_friends_by_id.sort(
    key=lambda id_and_friends: id_and_friends[1],  # Sort the list
    reverse=True)                                  # by num_friends
                                                    # largest to smallest
```

- 친구 수 순서로 (사용자 id, 친구 수) 리스트를 정렬

예제 1. 핵심 인물 찾아라

결과 확인

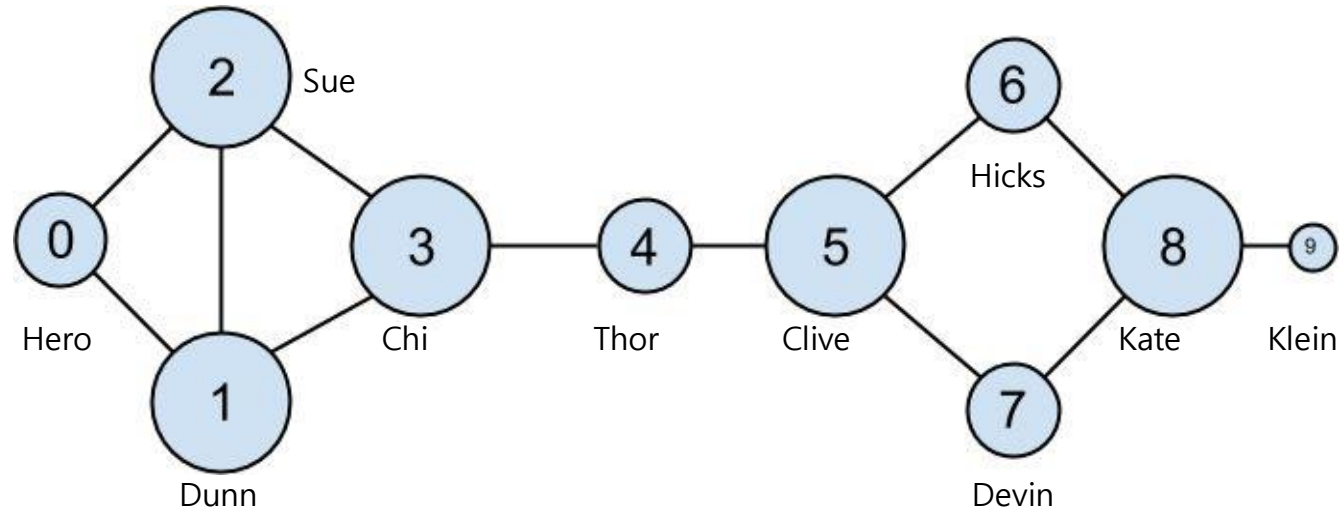
```
# Each pair is (user_id, num_friends):  
# [(1, 3), (2, 3), (3, 3), (5, 3), (8, 3),  
#  (0, 2), (4, 2), (6, 2), (7, 2), (9, 1)]  
  
assert num_friends_by_id[0][1] == 3      # several people have 3 friends  
assert num_friends_by_id[-1] == (9, 1)  # user 9 has only 1 friend
```

- 친구가 가장 많은 사용자는 3명의 친구를 가짐
- 친구가 가장 적은 사용자는 사용자 9로 친구가 한 명임

친구가 많은 사람은 3명까지 있고 가장 적은 사람은 1명이다.

예제 1. 핵심 인물 찾아라

시각화를 해서 보면 확인하기가 좋다!



조직의 핵심 인물은 친구가 3명인 Dunn, Sue, Chi, Clive, Kate!

그래프의 중요한 노드를 연결 중심성(degree centrality)으로 해석한 방법

Thor가 더 중심적인 역할을 할 것 같은데 과연 그럴지는 네트워크 분석에서 확인해보자!

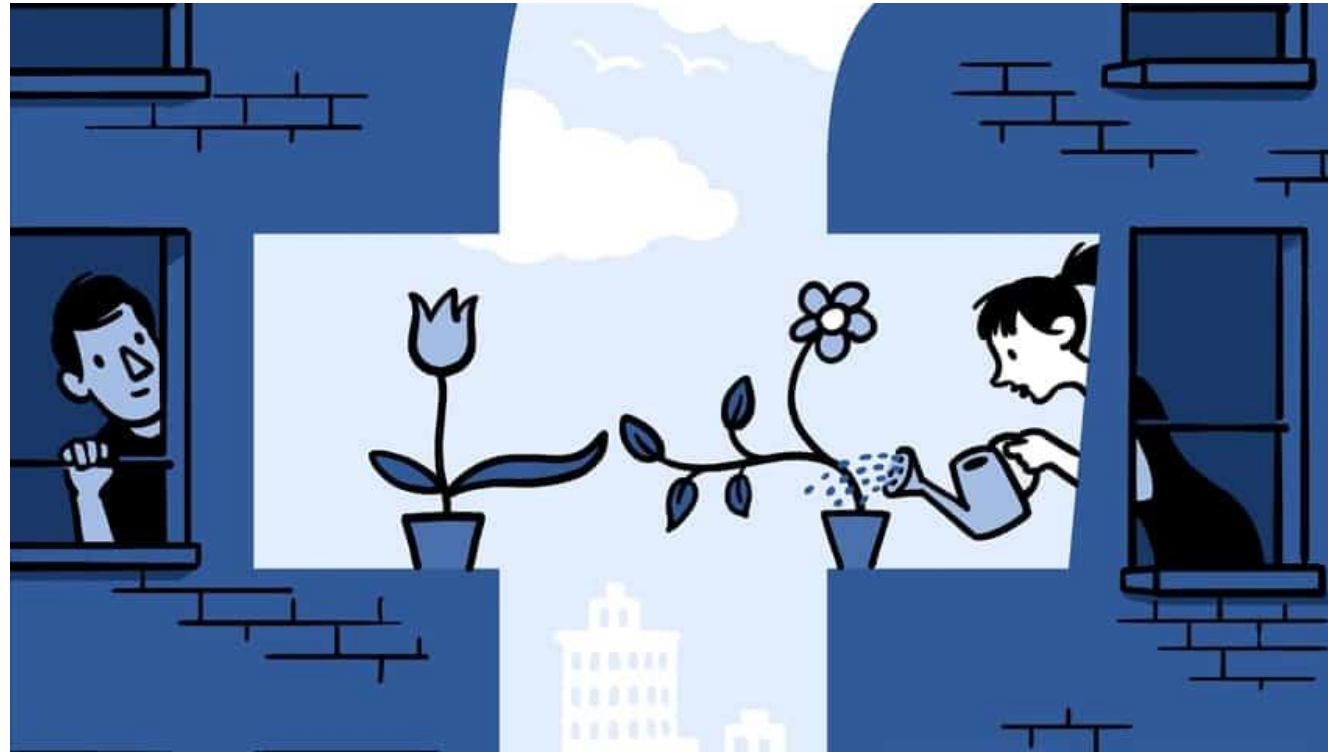
4. 데이터 마이닝 예제

예제2. 친구 추천하기



예제2. 친구 추천하기

소셜 네트워크에서 어떤 사람을 친구 추천 목록에 보여줄까?



예제2. 친구 추천하기

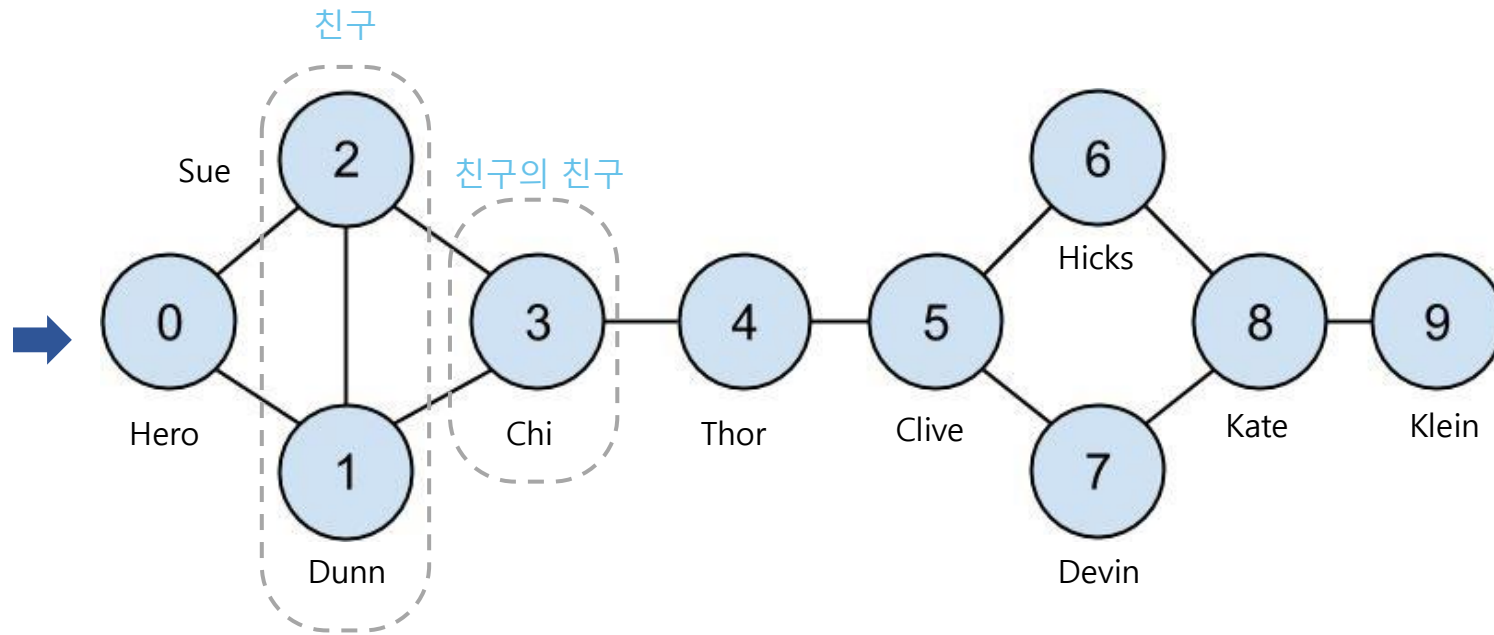


페이스북 친구 추천 방식

1. 개인 정보 (학교, 출신지, 혈액형, 거주지)와 비슷한 정보를 입력한 친구
2. 내 핸드폰에 저장되어 있는 친구
3. 친구의 친구
4. 검색해서 자주 들어가본 친구
5. 나를 몰래 엿탐하는 친구
6. 이메일을 주고받은 친구

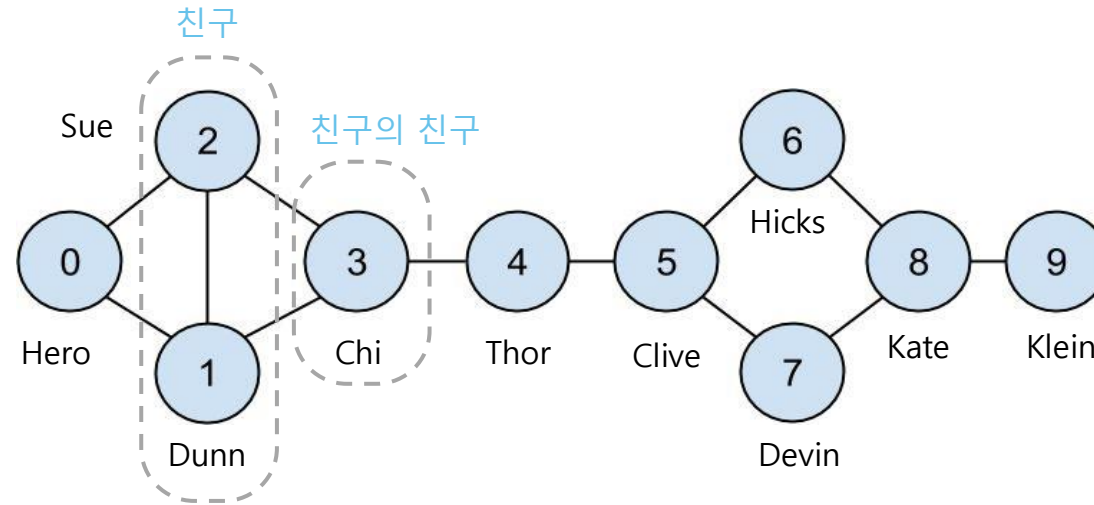
예제2. 친구 추천하기

친구의 친구를 추천하는 기능을 만들어보자



Hero에게 Chi를 추천해주자.

예제2. 친구 추천하기

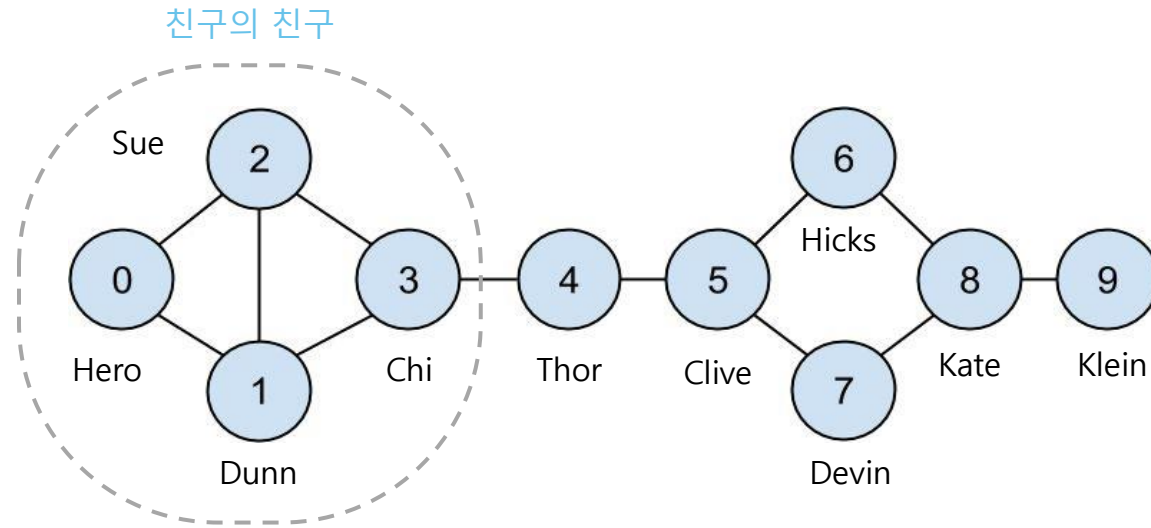


친구의 친구 목록 만들기 (않좋은 버전)

```
def foaf_ids_bad(user):  
    """foaf is short for "friend of a friend" """  
    return [foaf_id  
            for friend_id in friendships[user["id"]]  
            for foaf_id in friendships[friend_id]]
```

- user의 친구를 friendships[user["id"]] 찾고
- 다시 친구의 친구를 찾는 friendships[friend_id]로 찾는다.

예제2. 친구 추천하기



친구의 친구에 나와 친구가 포함!

1. user를 제외해야 한다
2. user의 친구를 제외해야 한다.

친구의 친구 목록 확인

```
assert foaf_ids_bad(users[0]) == [0, 2, 3, 0, 1, 3]

assert friendships[0] == [1, 2]
assert friendships[1] == [0, 2, 3]
assert friendships[2] == [0, 1, 3]
```

예제2. 친구 추천하기

Counter 객체 import

```
from collections import Counter # not loaded by default
```

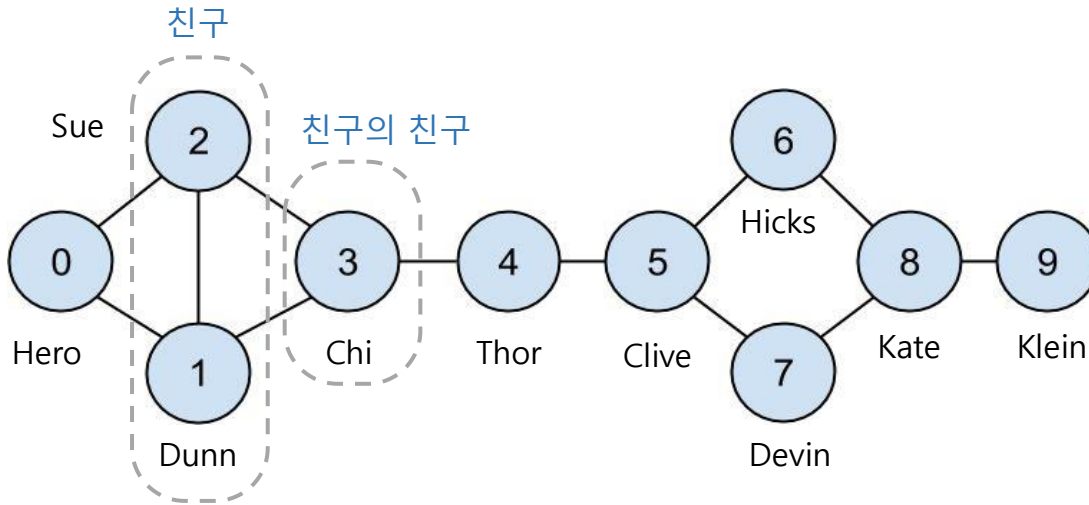
- 같은 값을 가진 항목의 개수를 세어주는 객체

친구의 친구 목록 만들기

```
def friends_of_friends(user):  
    user_id = user["id"]  
    return Counter(  
        foaf_id  
        for friend_id in friendships[user_id] # For each of my friends,  
        for foaf_id in friendships[friend_id] # find their friends  
        if foaf_id != user_id # who aren't me  
        and foaf_id not in friendships[user_id] # and aren't my friends.  
    )
```

- 나와 나의 친구가 아닌 사람만 목록에 나타나도록 조건을 추가함

예제2. 친구 추천하기



Hero에게 추천해줄 친구는 Chi이다.

친구의 친구 목록 확인

```
print(friends_of_friends(users[0]))
```

```
# Counter({3: 2})
```

예제2. 친구 추천하기2

관심사가 비슷한 친구를 추천해보자!



<https://www.pinterest.cl/pin/708613322601784212/>

예제2. 친구 추천하기2

데이터 사이언티스트 중에 관심사가 비슷한 친구를 추천해보자!

사용자 별 관심 목록

사용자 id

관심사

```
interests = [  
    (0, "Hadoop"), (0, "Big Data"), (0, "HBase"), (0, "Java"),  
    (0, "Spark"), (0, "Storm"), (0, "Cassandra"),  
    (1, "NoSQL"), (1, "MongoDB"), (1, "Cassandra"), (1, "HBase"),  
    (1, "Postgres"), (2, "Python"), (2, "scikit-learn"), (2, "scipy"),  
    (2, "numpy"), (2, "statsmodels"), (2, "pandas"), (3, "R"), (3, "Python"),  
    (3, "statistics"), (3, "regression"), (3, "probability"),  
    (4, "machine learning"), (4, "regression"), (4, "decision trees"),  
    (4, "libsvm"), (5, "Python"), (5, "R"), (5, "Java"), (5, "C++"),  
    (5, "Haskell"), (5, "programming languages"), (6, "statistics"),  
    (6, "probability"), (6, "mathematics"), (6, "theory"),  
    (7, "machine learning"), (7, "scikit-learn"), (7, "Mahout"),  
    (7, "neural networks"), (8, "neural networks"), (8, "deep learning"),  
    (8, "Big Data"), (8, "artificial intelligence"), (9, "Hadoop"),  
    (9, "Java"), (9, "MapReduce"), (9, "Big Data")  
]
```

- (사용자 id, "관심사") 튜플 리스트로 정의됨

예제2. 친구 추천하기2

같은 것에 관심을 갖는 사용자 목록

```
def data_scientists_who_like(target_interest):  
    """Find the ids of all users who like the target interest."""  
    return [user_id  
            for user_id, user_interest in interests  
            if user_interest == target_interest]
```

- Target_interest와 같은 관심사를 갖는 사용자들의 목록을 만듦

예제2. 친구 추천하기2

“관심사가 비슷한 친구” = “관심사가 가장 많이 겹치는 친구”

관심 항목

	Item1	Item2	Item3	Item4	Item5
User1	O	O		O	O
User2		O		O	
User3		O	O	O	O
User4	O		O		O

사용자

1 관심 항목이 비슷한 사람들의 목록을 만들기

- 사용자1의 관심 목록인 {item1, item2, item4, item5}와 비슷한 관심을 갖는 사용자 목록을 만들기

2 사용자 별로 공통 관심 항목 횟수 세기



공통 관심항목 2개 3개 2개

User3이 가장 많이 나타나므로 친구로 추천!

예제2. 친구 추천하기2

Defaultdict 딕셔너리

```
from collections import defaultdict
```

- defaultdict를 사용하면 연산 시에 새로운 키에 대해 초기화를 할 필요 없다.

관심 별 사용자 목록 구성

```
# Keys are interests, values are lists of user_ids with that interest
user_ids_by_interest = defaultdict(list)

for user_id, interest in interests:
    user_ids_by_interest[interest].append(user_id)
```

사용자 별 관심 목록 구성

```
# Keys are user_ids, values are lists of interests for that user_id.
interests_by_user_id = defaultdict(list)

for user_id, interest in interests:
    interests_by_user_id[user_id].append(interest)
```

예제2. 친구 추천하기2

같은 관심을 갖는 사람들 목록

```
def most_common_interests_with(user):  
    return Counter(  
        interested_user_id  
        for interest in interests_by_user_id[user["id"]]  
        for interested_user_id in user_ids_by_interest[interest]  
        if interested_user_id != user["id"]  
    )
```

- User와 같은 관심사를 갖고 있는 사용자 목록을 만들 (단, user 자신은 제외)
- 사용자 별로 몇 개의 관심사가 겹치는지 개수를 세서 반환

사용자 0의 추천 친구

```
print(most_common_interests_with(users[0]).most_common())
```

[(9, 3), (1, 2), (8, 1), (5, 1)] 친구 9를 추천

추천 시스템의 "협업 필터링 (Collaborative Filtering)"의 기본 아이디어

4. 데이터 마이닝 예제

예제3. 연봉과 경력과의 관계 찾기



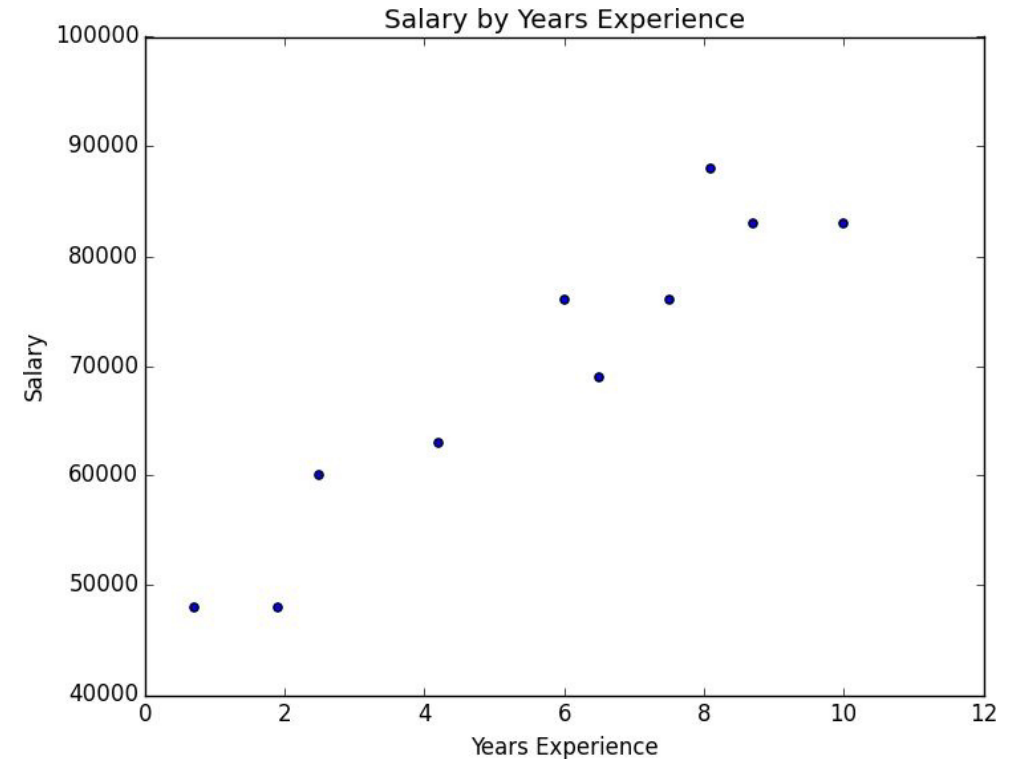
예제3. 연봉과 근속연수의 관계를 찾아라

근속연수에 따라 연봉의 관계를 찾아라

직원들의 연봉 및 근속연수 테이블

```
salaries_and_tenures = [(83000, 8.7), (88000, 8.1),  
                        (48000, 0.7), (76000, 6),  
                        (69000, 6.5), (76000, 7.5),  
                        (60000, 2.5), (83000, 10),  
                        (48000, 1.9), (63000, 4.2)]
```

- (연봉, 경력) 튜플 리스트로 정의됨



근속연수에 따라 평균 연봉을 계산해보자!

근속연수에 따라 연봉이 높아진다는 사실은 그래프로 파악이 됨

예제3. 연봉과 근속연수의 관계를 찾아라

근속연수 별로 연봉 리스트 구성

```
# Keys are years, values are lists of the salaries for each tenure.
salary_by_tenure = defaultdict(list)

for salary, tenure in salaries_and_tenures:
    salary_by_tenure[tenure].append(salary)
```

- 같은 근속연수를 갖는 사람들의 연봉 목록을 만듦

근속연수 별 평균 연봉 계산

```
# Keys are years, each value is average salary for that tenure.
average_salary_by_tenure = {
    tenure: sum(salaries) / len(salaries)
    for tenure, salaries in salary_by_tenure.items()
}
```

- 근속 연수 별로 연봉 목록을 이용해서 평균 연봉을 계산해서
근속 연수 별 평균 연봉 딕셔너리 생성

예제3. 연봉과 근속연수의 관계를 찾아라

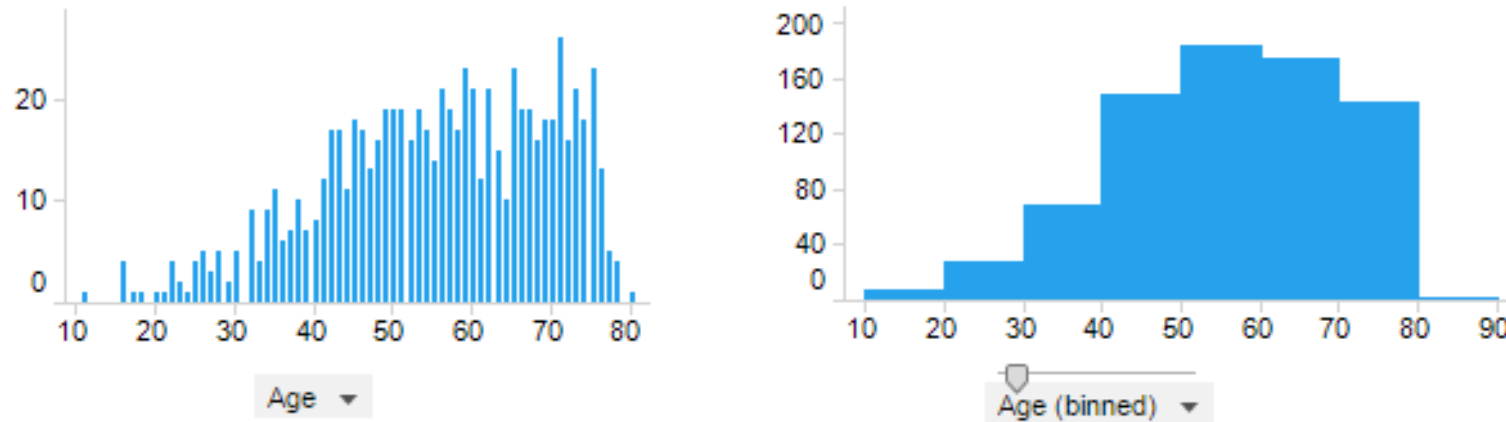
결과

```
assert average_salary_by_tenure == {  
    0.7: 48000.0,  
    1.9: 48000.0,  
    2.5: 60000.0,  
    4.2: 63000.0,  
    6: 76000.0,  
    6.5: 69000.0,  
    7.5: 76000.0,  
    8.1: 88000.0,  
    8.7: 83000.0,  
    10: 83000.0  
}
```

같은 근속연수가 없어서
요약 통계량이 의미가 없어짐

예제3. 연봉과 근속연수의 관계를 찾아라

관측 오차를 없애기 위해 숫자 데이터를 작은 구간의 값으로 대체하는 방법



비닝 (binning), 이산화 (discretization) 또는 양자화 (quantization)라고 부름

예제3. 연봉과 근속연수의 관계를 찾아라

근속연수 버킷 구성

```
def tenure_bucket(tenure):  
    if tenure < 2:  
        return "less than two"  
    elif tenure < 5:  
        return "between two and five"  
    else:  
        return "more than five"
```

- 근속연수를 {2년 이하, 2년에서 5년사이, 5년 이상}으로 분류

근속연수 버킷 별 연봉 리스트 구성

```
# Keys are tenure buckets, values are lists of salaries for that bucket.  
salary_by_tenure_bucket = defaultdict(list)  
  
for salary, tenure in salaries_and_tenures:  
    bucket = tenure_bucket(tenure)  
    salary_by_tenure_bucket[bucket].append(salary)
```

- 각 근속연수를 {2년 이하, 2년에서 5년사이, 5년 이상} 버킷 중 하나에 할당
- 각 버킷 별로 연봉 리스트를 생성

예제3. 연봉과 근속연수의 관계를 찾아라

버킷 별 평균 연봉 계산

```
# Keys are tenure buckets, values are average salary for that bucket
average_salary_by_bucket = {
    tenure_bucket: sum(salaries) / len(salaries)
    for tenure_bucket, salaries in salary_by_tenure_bucket.items()
}
```

결과

```
assert average_salary_by_bucket == {
    'between two and five': 61500.0,
    'less than two': 48000.0,
    'more than five': 79166.66666666667
}
```

5년 이상 경력을 가지면 2년 경력 미만의 데이터 과학자보다 65%정도 더 번다.

예제3. 연봉과 근속연수의 관계를 찾아라

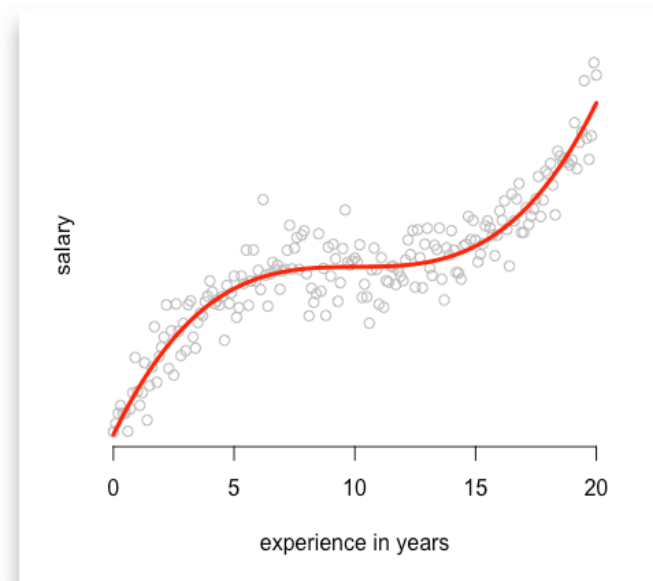
연봉을 예측하는 모델을 만들려면 "회귀 분석"을 알아보자.

선형 회귀 분석



- 선형 회귀 모델

비선형 회귀 분석



- 딥러닝

4. 데이터 마이닝 예제

예제4. 유료 계정 전환 대상자를 찾아라



예제4. 유료 계정 전환 대상자를 찾아라

어떤 사용자가 유료 계정으로 전환하는가?

서비스 이용 기간	유료 계정
0.7	paid
1.9	unpaid
2.5	paid
4.2	unpaid
6	unpaid
6.5	unpaid
7.5	unpaid
8.1	unpaid
8.7	paid
10	paid

유료 계정

무료 계정

유료 계정

서비스 이용 기간에 따라 유료 계정 사용 여부를 예측하는 모델을 생성

서비스 이용 기간에 따라 유료 계정 전환 예측 모델 정의

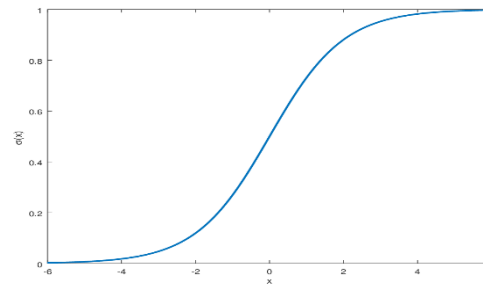
```
def predict_paid_or_unpaid(years_experience):  
    if years_experience < 3.0:  
        return "paid"  
    elif years_experience < 8.5:  
        return "unpaid"  
    else:  
        return "paid"
```

예제4. 유료 계정 전환 대상자를 찾아라

유료 계정으로 전환할 가능성을 예측하는 모델은 "로지스틱 회귀 분석"에서 알아보자.



로지스틱 회귀 모델은 확률을 출력하는 확률 모델



유료 계정일 확률

로지스틱 함수를 이용해서 선형 회귀 값을 확률로 변환하는 모델

Thank you!

