



19.12.26  
정호영

# 시작하기에 앞서 유의사항

---

1. Download plz!!! ([github.com/hotorch/pnustat\\_lec\\_201912](https://github.com/hotorch/pnustat_lec_201912))
2. 짧은 시간 내에 많은 내용들이 들어가 있기 때문에 핵심만 다룰 예정입니다.  
집중해서 잘 들어주세요!
3. 여유가 되면 실습할 시간을 따로 줄 생각입니다.
4. git repo에 'book' 부분 예습해오시면 좋아요~
5. 실습 때 다루는 코드가 쉬운 수준이기 때문에 따로 공부하시면서 응용 권장
6. 막히거나 에러뜨거나 질문 → 손

# 저는 대략 이런 사람입니다

Birth : 1993.02.03

Company : AgileSoDA

Email : [ghdud1519@naver.com](mailto:ghdud1519@naver.com)

Github : <https://github.com/hotorch>

Kaggle : <https://www.kaggle.com/heuingttiang>

Tools : R, Python, SQL

Interest : NLP, NLG, NLU, Statistics, ML, DL

Paper : [Comparision of Term Weighting Schemes for Document Classification](#) (2018.06)

일대기(19.11.16 홈커밍강연)를 보고 싶다면 : <https://www.notion.so/dsghdud/PNU-11-16-27c29c865c1c4bd18492bdfe2a47d9ad> -> 제 이야기 부분 참고



Choi Yosich님이 링크를 공유했습니다.

인기 게시물 작성 멤버 - 4시간

애자일소다, 60억 규모 시리즈 A 투자 유치

기술 및 사업 협력을 전제로 한 전략적 투자 형태로 각 기업과 개별 접촉한 결과로 BNK증권, KB증권, 한화증권 등 총 6곳의 투자사 참여



AITIMES.KR

애자일소다, 60억 규모 시리즈 A 투자 유치 - 인공지능신문

AI 소프트웨어 기업인 애자일소다(대표 최대우)는 60억 규모의 시리즈 A 투...

# 저는 대략 이런 사람입니다

---



거제도



부산대학교  
PUSAN NATIONAL UNIVERSITY

통계학과(11학번)



부산대학교  
PUSAN NATIONAL UNIVERSITY

통계학  
(석사, 18년 졸)

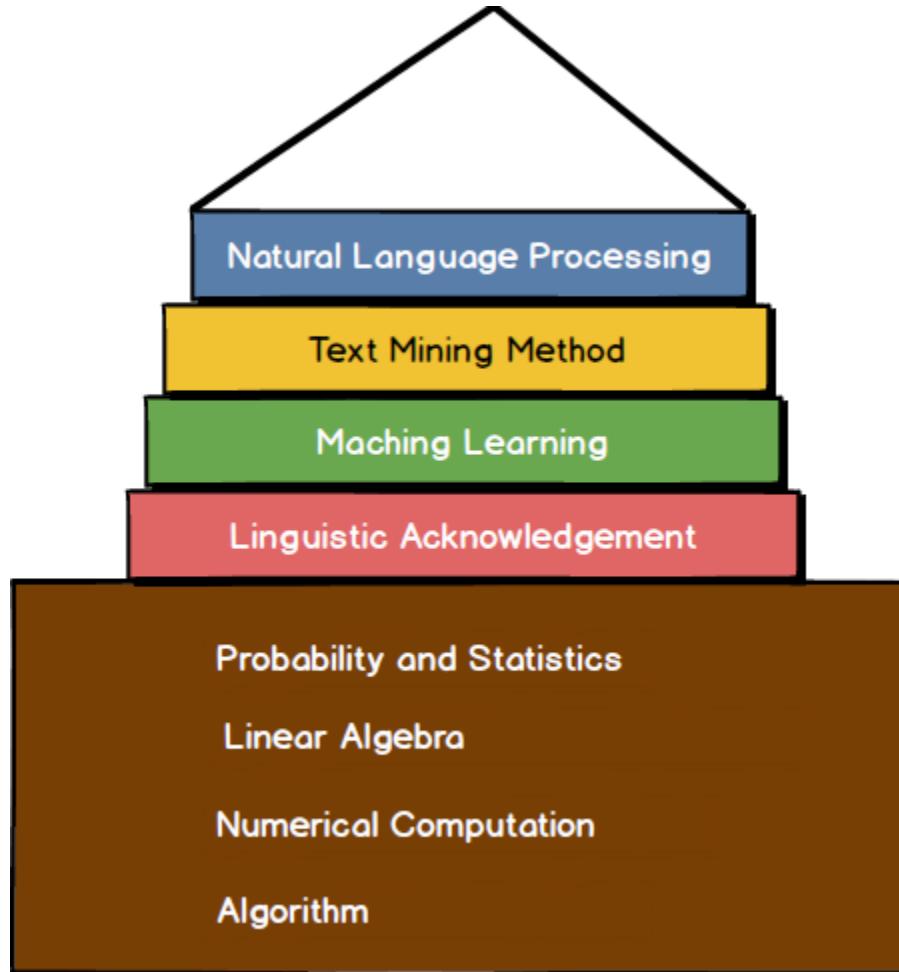


프로젝트

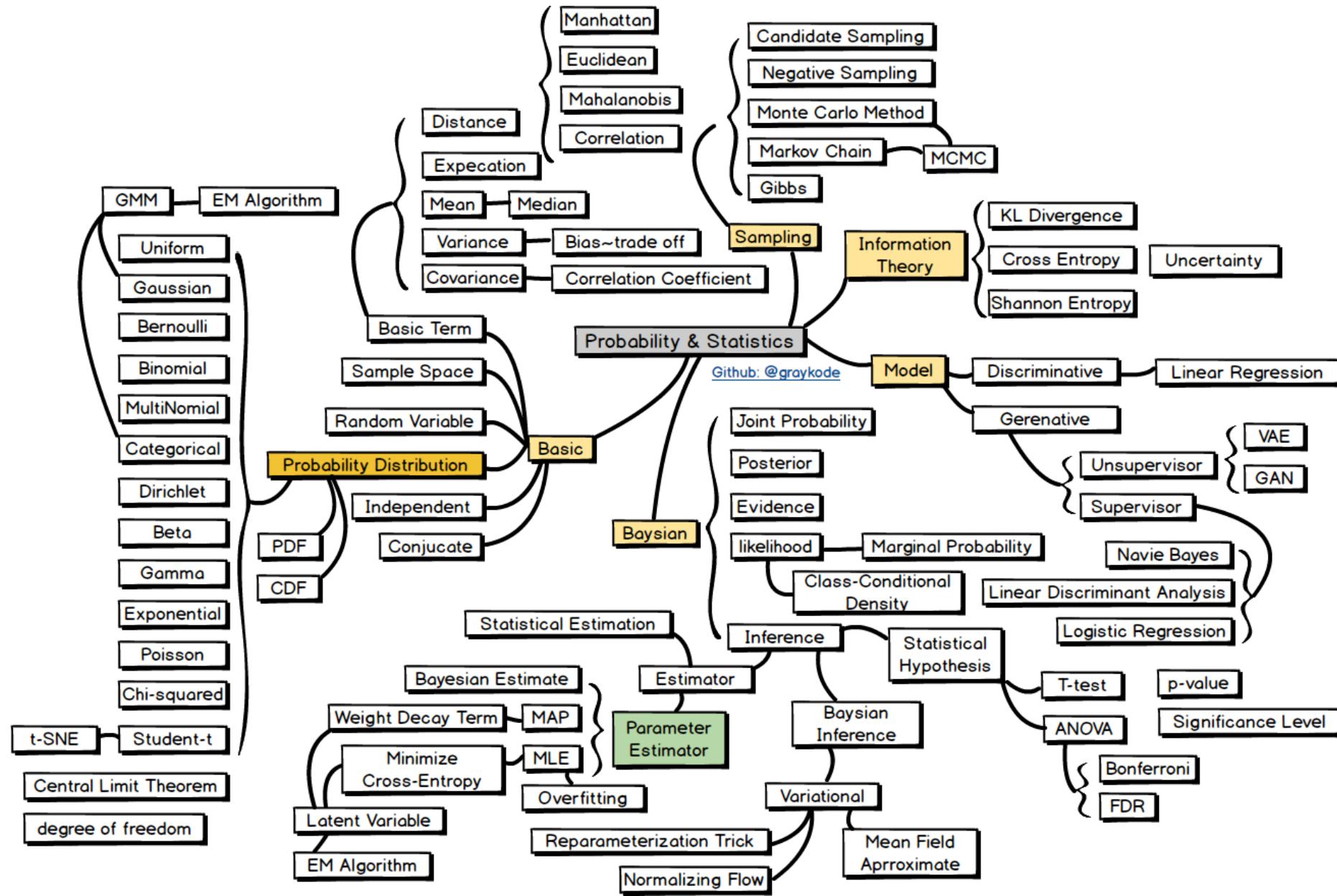
IBK기업은행  
현대해상손해보험  
우리은행

# 공부한 순서

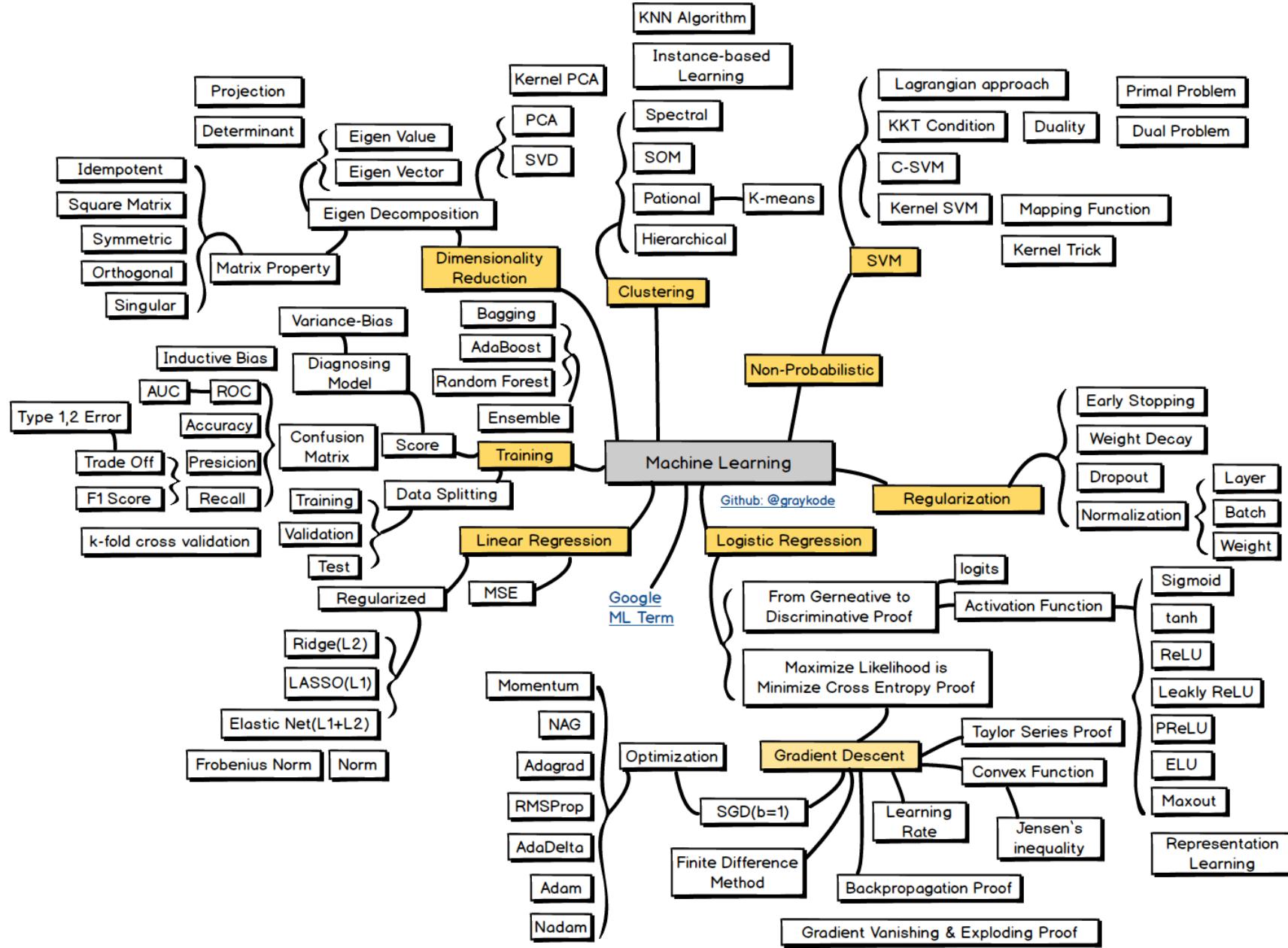
---



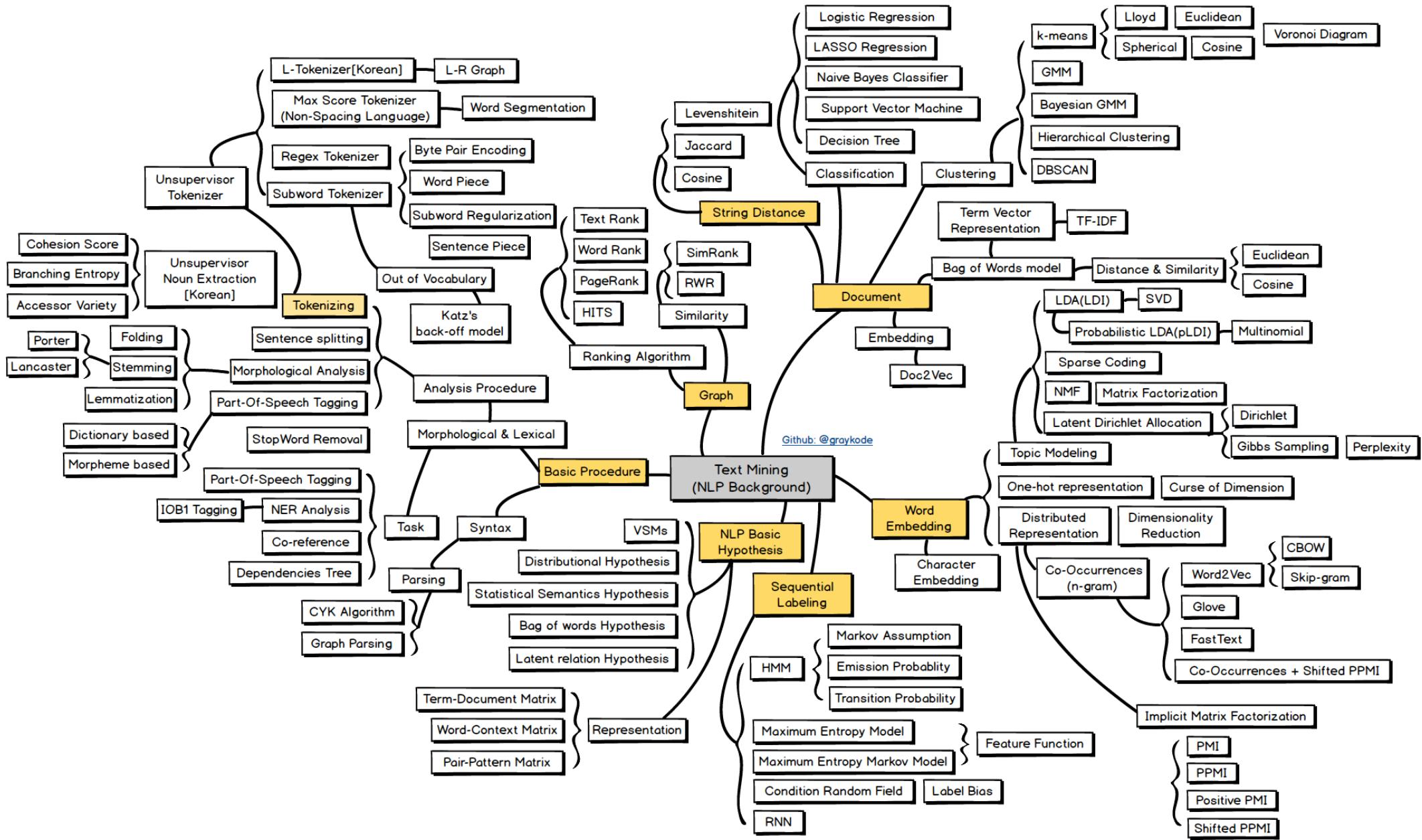
# 공부한 순서 - Prob & Statistics(학부)



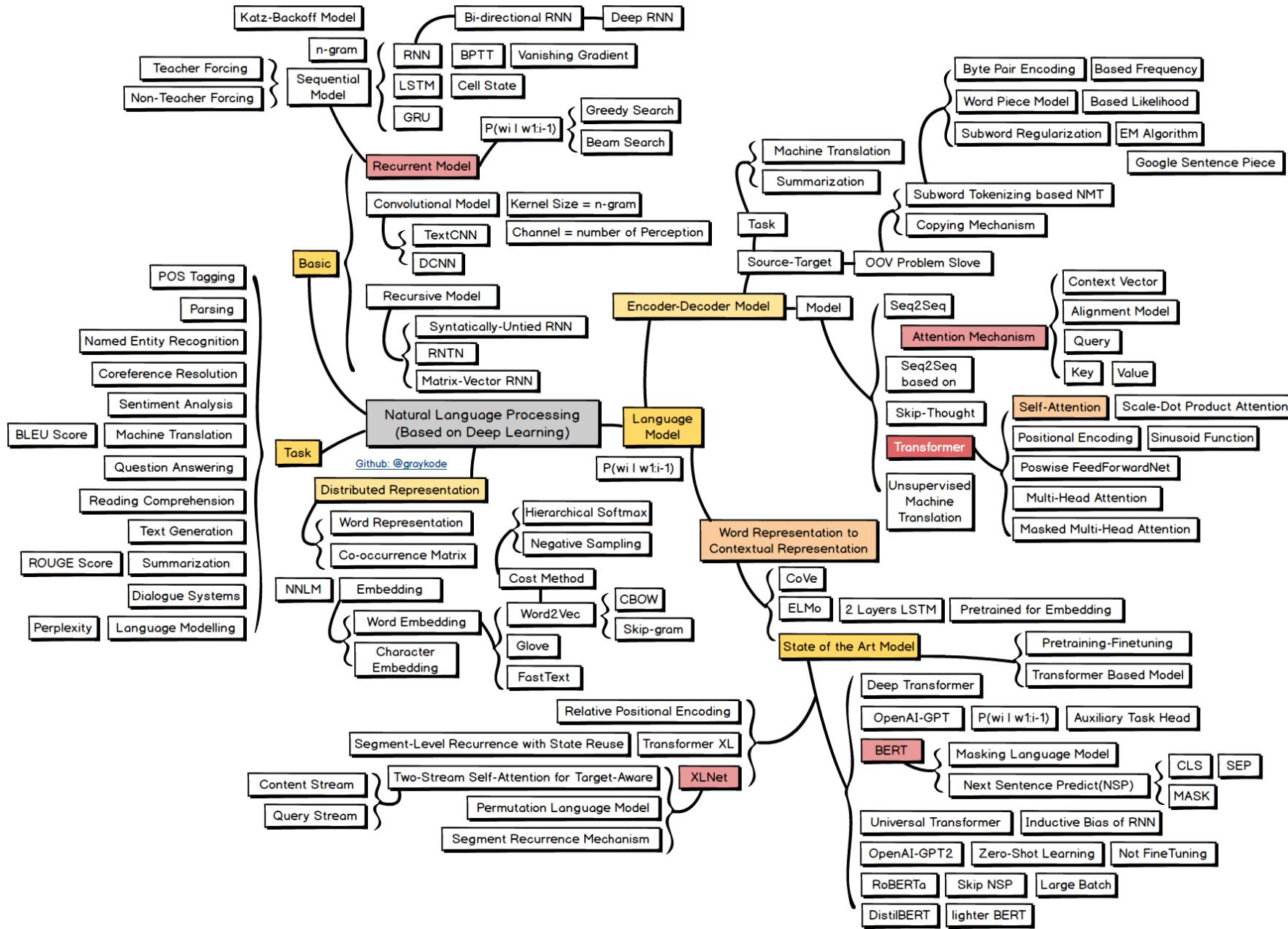
# 공부한 순서 - Prob & Statistics(3학년 ~ 대학원)



# 공부한 순서 - Text Mining(석사 때 독학)



# 공부한 순서 - NLP(졸업 이후 계속)



# Contents

---

## 1st session

- Preprocessing - dplyr, data.table

## 2<sup>nd</sup> session

- ML & Tree model

## 3<sup>rd</sup> session

- h2o tutorial

## 4<sup>th</sup> session

- XAI : Local Interpretable Model-agnostic Explanations

# Contents

---

## **1st session**

- **Preprocessing - dplyr, data.table**

## 2<sup>nd</sup> session

- Tree model

## 3<sup>rd</sup> session

- h2o tutorial

## 4<sup>th</sup> session

- XAI : Local Interpretable Model-agnostic Explanations

# 전처리?

여러분이 분석 시간의 7~80%를 투자하는 시간입니다.

모델에 넣어 좋은 결과를 얻고 싶으시다면 많은 시간을 할애하셔야 합니다.(쓰레기를 넣으면 쓰레기가 나오듯이!)

이론의 탄탄함도 중요하지만 분석 Tool을 다루는데 있어서 경쟁력은 전처리 실력이라고 생각합니다.

## dplyr

pipe operator %>% 는 이런 느낌이에요(in dplyr)

```
g(f(y)) == y %>% f() %>% g()  
g(f(x,y,z)) == y %>% f(x, ., z) %>% g
```

장점은 코드를 보았을 때 직관적인 해석이 용이함

## SQL vs dplyr

sql 구문 작동 순서(출처)

- ⑤ SELECT 컬럼명
- ① FROM 테이블명
- ② WHERE 조건식
- ③ GROUP BY 컬럼이나 (표현식)
- ④ HAVING 조건식(집계함수 조건)
- ⑥ ORDER BY 컬럼명(표현식)

### 순서 설명

1. 조회 대상 테이블을 확인 (FROM 절)
2. 조회하고자 하는 데이터를 추출하는 조건 확인 (WHERE 절)
3. 행들을 명시된 컬럼을 기준으로 그룹화 (GROUP BY 절)
4. 그룹화 된 데이터 중 조건에 맞는 데이터만 출력 (HAVING 절)
5. 명시된 데이터만 화면에 출력 (SELECT 절)
6. 원하는 순서대로 정렬 (ORDER BY 절)

dplyr 직관적인 해석 순서(출처)

```
filter()
  summarise(
    select(
      group_by(chflights, Year, Month, DayofMonth),
      Year:DayofMonth, ArrDelay, DepDelay
    ),
    arr = mean(ArrDelay, na.rm = TRUE),
    dep = mean(DepDelay, na.rm = TRUE)
  ),
  arr > 30 | dep > 30
)
```

## dplyr의 주요 함수

- **filter** : 행 조건을 줘서 불러옴, sql에서 where
- **select** : 필요한 열만 선택
- **mutate** : 새로운 컬럼을 계산해서 생성, 파생변수를 만듬
- **arrange** : 조건에 따라 재정렬
- **group\_by** : 그룹을 조건으로 사용
- **summarise** : 요약형 계산을 진행, 위와 같이 병행해서 사용함
- **left join**

code 연습!!

# tidyr(따로 공부 해보세요!)

## ▼ spread : long 폼 → wide 폼

spread () Function : spread(data, key, value)

- data : data frame
- key : wide하게 변수명들이 될 column name
- value : key값들의 value

## ▼ gather : wide 폼 → long 폼

gather() Function : gather(data, key, value, ...)

- data : data frame
- key : column name들의 새로운 variable을 생성
- value : variable value값들의 variable을 생성
- ... : 변환할 column 지정

The diagram illustrates the transformation of a wide data frame into a long data frame using the `spread` function. It shows two data frames: 'messy' (wide) and 'tidier' (long).

**messy** Data Frame:

id	trt	work.T1	home.T1	work.T2	home.T2
1	treatment	0.08513597	0.6158293	0.1135090	0.05190332
2	control	0.22543662	0.4296715	0.5959253	0.26417767
3	treatment	0.27453052	0.6516557	0.3580500	0.39879073
4	control	0.27230507	0.5677378	0.4288094	0.83613414

**tidier** Data Frame:

id	trt	key	time
1	treatment	work.T1	0.08513597
2	control	work.T1	0.22543662
3	treatment	work.T1	0.27453052
4	control	work.T1	0.27230507
1	treatment	home.T1	0.61582931
2	control	home.T1	0.42967153
3	treatment	home.T1	0.65165567
4	control	home.T1	0.56773775
1	treatment	work.T2	0.1135098
2	control	work.T2	0.59592531
3	treatment	work.T2	0.35804998
4	control	work.T2	0.42880942
1	treatment	home.T2	0.05190332
2	control	home.T2	0.26417767
3	treatment	home.T2	0.39879073
4	control	home.T2	0.83613414

A blue arrow points from the 'work.T1' column in the 'messy' frame to the 'key' column in the 'tidier' frame. A red arrow points from the 'time' column in the 'tidier' frame back to the 'home.T2' column in the 'messy' frame.

▼ separate : 한 열에서 데이터를 지정 조건으로 분리

**separate()** Function : separate(data, col, into, sep = " ")

- data : data frame
- col : separate 하려는 변수
- into : 새롭게 넣게되는 변수이름
- sep : separate의 구분자(char, num, or symbol)

▼ unite : 여러 열의 데이터를 한 열로 합침

**unite()** Function : unite(data, col, ..., sep = " ")

- data : data frame
- col : Merge 한 변수이름
- ... : Merge하려는 변수들
- sep : Merge의 구분자(char, num, or symbol)
- extract : 데이터를 분리하는 품을 지정하여 분리

# data.table

C++ 기반의 데이터 처리 패키지

## Basic

- `dt[i, j, by = 'colname', ...]`
  - `dt` : `data.table` 클래스를 가지는 데이터
  - `i, j` : `i`는 행을 선택하는 문법, `j`에는 열을 선택하는 문법
  - `by` : `group_by`의 대상이 되는 변수 이름의 list or vector

## Index

- `data.table`의 형식은 [행, 표현식, 옵션], `data.frame`은 [행, 열, 옵션]으로 되어있기 때문에 서로 조금씩 다른 인덱싱방법을 사용함
- 코드를 보자

# Fast Grouping & Fast Selecting

## data.table의 by에 의한 Grouping

- 코드를 보자

## data.table의 J()에 의한 selecting

- 조건으로 데이터 선택
  - 반드시 setkey를 이용하여 KEY변수가 지정된 상태여야함
  - DT[J('제약조건')]의 형식으로 작성함

---

## 처리 속도차이

- data.frame보다 20배 빠름, pandas 보다 빠름
- 그 이유는 key로 index 지정하고 연산하기 때문

## 조건을 이용한 데이터 선택

- `data.table`에서는 `J`표현식이 존재하며, `J`를 사용하기 위해서는 `key`설정이 반드시 필요
    - `key`는 행을 `sort`해주기 위한 기준 변수
  - `setkeyv`의 경우는 `key`를 두개이상 설정할 수 있음
  - `tables()`를 통해 `key` 지정된 것을 확인할 수 있음
-

## Grouping 연산

- DT[, 연산식, by = 'variable']
- 여기서, reserved keyword(종류가 엄청 많음)는
  - .N
    - by에 의해 grouping이 될 땐 매칭된 변수의 행의 개수를 나타냄
    - 매칭이 되지 않는다면 NA나 0으로 출력
  - .SD
    - Subset of x's Data for each group, excluding any columns used in by(or keyby)
    - 예를 들어, nrow(.SD)는 J와 by로 분류된 변수를 포함하는 행의 개수 단,(예를 들어) by로 나눈 "male"과 "female"는 분리되어 계산
  - 여기서, .SDcols는 .SD의 특정한 변수이름을 지정하여 해당 변수로만 이루어진 새 .SD를 구성함

## Merge 연산

- data.table merge연산이 빛을 보게 되는데 data.frame보다 400배 이상 속도를 보인다.

## Data 수정 및 삭제

- DT[i, 파생변수명 := '값']
- DT[i, `:=` (파생변수명 = '값')]

## 여러분들한테 당부의 말씀

- dplyr에서 알려준 함수 이외에 응용 함수가 되게 많습니다. 찾아서 공부하는 것 추천!(\_all, \_if, \_each, ... etc)
- data-wrangling-cheatsheet 참고하는 것 추천([R studio cheat sheets 모음](#))
- dplyr 보다 data.table이 빠르지만 직관적이진 못합니다. 연습을 특히 많이 하시면서 찾아서 공부하는 것 추천합니다.
- 전처리가 60%이기 때문에 사람들 실력은 여기서 많이 갈립니다! 화이팅~

## dtplyr : Data Table Back-End for dplyr

dplyr의 직관적인 풍부한 표현력 + data.table의 속도 반영

최근에 나온거니까 코드만 점검해봐요

# Contents

---

## 1st session

- Preprocessing - dplyr, data.table

## 2<sup>nd</sup> session

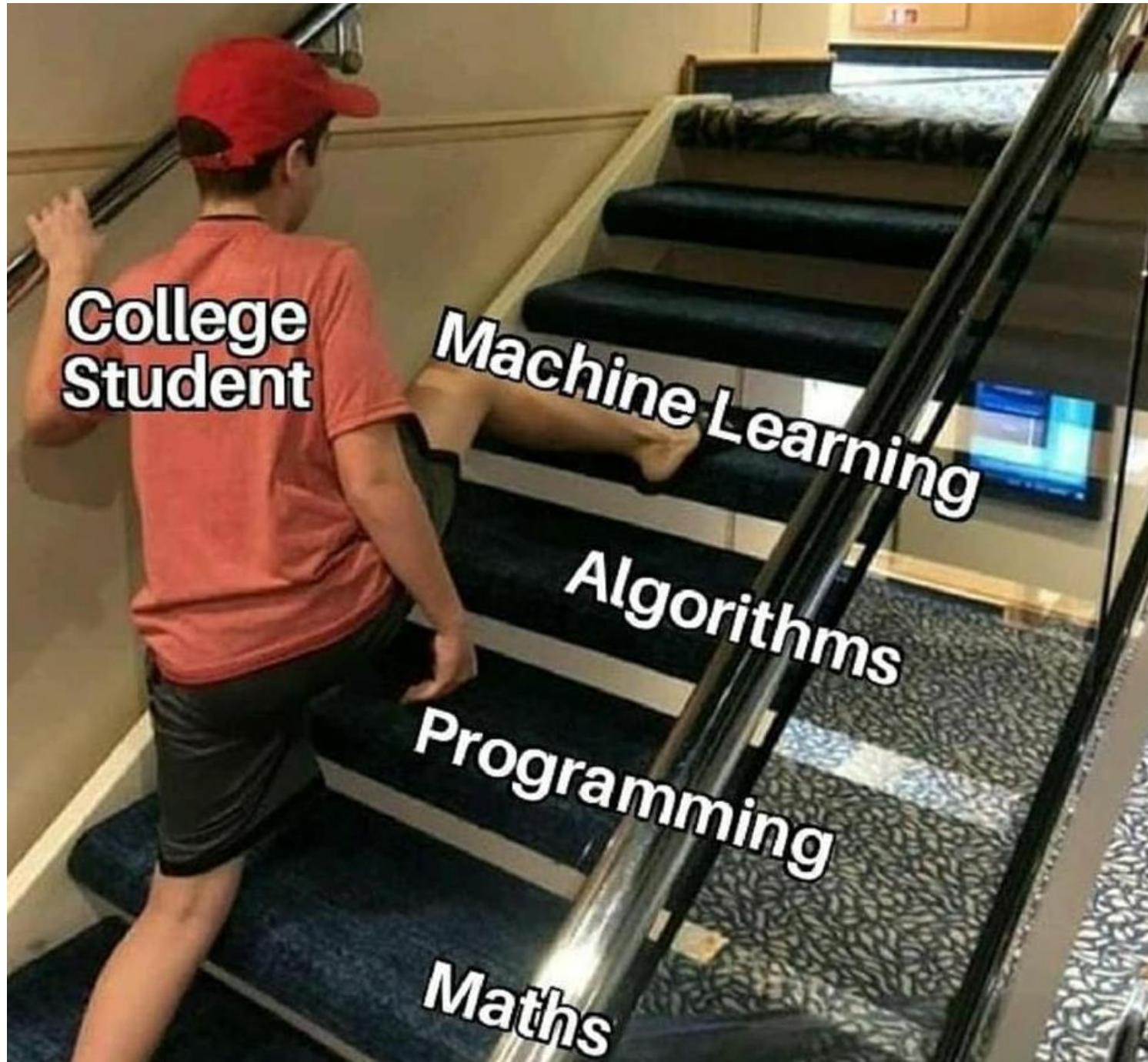
- **ML & Tree model**

## 3<sup>rd</sup> session

- h2o tutorial

## 4<sup>th</sup> session

- XAI : Local Interpretable Model-agnostic Explanations



# 머신러닝, 딥러닝의 최종목표??



## 이안 굿펠로우 <

미국 연구원

출생: 미국

논문: Deep Learning of Representations and its Application to Computer Vision (2014)

분야: 컴퓨터 과학

저서: Deep Learning Adaptive Computation and Machine Learning

학력: 스탠포드 대학, 몬트리올 대학교

지도 교수: 요슈아 벤지오

Regularization도 중요한 개념이지만!

"Regularization is any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error." [Ian Goodfellow et al., 2016]

Generalization : Having good prediction on unseen data.



$$\begin{aligned}
\text{MSE}(\hat{\theta}) &= \mathbb{E} \left[ (\hat{\theta} - \theta)^2 \right] \\
&= \mathbb{E} \left[ \left( \hat{\theta} - \mathbb{E}[\hat{\theta}] + \mathbb{E}[\hat{\theta}] - \theta \right)^2 \right] \\
&= \mathbb{E} \left[ \left( \hat{\theta} - \mathbb{E}[\hat{\theta}] \right)^2 + 2 \left( \hat{\theta} - \mathbb{E}[\hat{\theta}] \right) \left( \mathbb{E}[\hat{\theta}] - \theta \right) + \left( \mathbb{E}[\hat{\theta}] - \theta \right)^2 \right] \\
&= \mathbb{E} \left[ \left( \hat{\theta} - \mathbb{E}[\hat{\theta}] \right)^2 \right] + \mathbb{E} \left[ 2 \left( \hat{\theta} - \mathbb{E}[\hat{\theta}] \right) \left( \mathbb{E}[\hat{\theta}] - \theta \right) \right] + \mathbb{E} \left[ \left( \mathbb{E}[\hat{\theta}] - \theta \right)^2 \right] \\
&= \mathbb{E} \left[ \left( \hat{\theta} - \mathbb{E}[\hat{\theta}] \right)^2 \right] + 2 \left( \mathbb{E}[\hat{\theta}] - \theta \right) \mathbb{E} \left[ \hat{\theta} - \mathbb{E}[\hat{\theta}] \right] + \left( \mathbb{E}[\hat{\theta}] - \theta \right)^2 \quad \mathbb{E}[\hat{\theta}] - \theta = \text{const.} \\
&= \mathbb{E} \left[ \left( \hat{\theta} - \mathbb{E}[\hat{\theta}] \right)^2 \right] + 2 \left( \mathbb{E}[\hat{\theta}] - \theta \right) \left( \mathbb{E}[\hat{\theta}] - \mathbb{E}[\hat{\theta}] \right) + \left( \mathbb{E}[\hat{\theta}] - \theta \right)^2 \quad \mathbb{E}[\hat{\theta}] = \text{const.} \\
&= \mathbb{E} \left[ \left( \hat{\theta} - \mathbb{E}[\hat{\theta}] \right)^2 \right] + \left( \mathbb{E}[\hat{\theta}] - \theta \right)^2 \\
&= \text{Var}(\hat{\theta}) + \text{Bias}(\hat{\theta}, \theta)^2
\end{aligned}$$

linearity: [https://en.wikipedia.org/wiki/Expected\\_value](https://en.wikipedia.org/wiki/Expected_value)

누가 TM 클을 잘 받을 것인가?



누가 이탈할 가능성이 높은가?



질병고지에 따른 할증률을  
어떻게 운영할 것인가?



어떤 계약을 인수 또는 거절할 것인가?



머신러닝은 컴퓨터가 경험에서부터 자동으로 몰랐던 사실을 찾아낸다

컴퓨터가 데이터에서부터 자동으로 몰랐던 사실을 찾아낸다

컴퓨터가 데이터에서부터 학습에 의해 몰랐던 사실을 찾아낸다

컴퓨터가 데이터에서부터 학습에 의해 내재된 규칙을 찾아낸다

사람이 이해하기 어려운

사람이 학습하기 어려운

사람이 추출하기 어려운

## 머신러닝을 돌렸는데 새로운 사실이 안나와요...

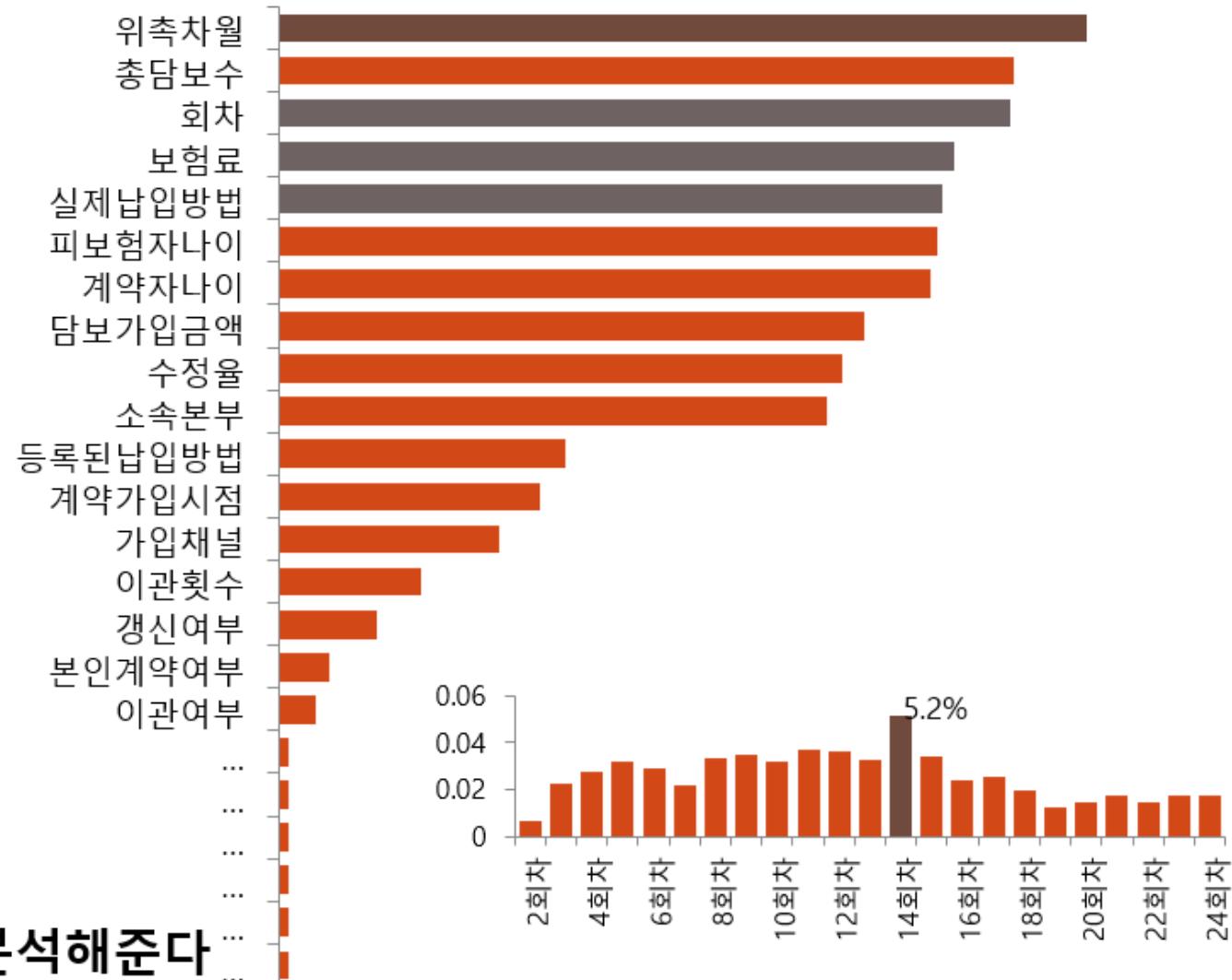
누가 이탈(탈락)할 가능성이 높은가?



컴퓨터가 데이터에서부터 ...

→ 우리가 넣은 변수에 대한 결과만을 분석해준다

### 변수중요도

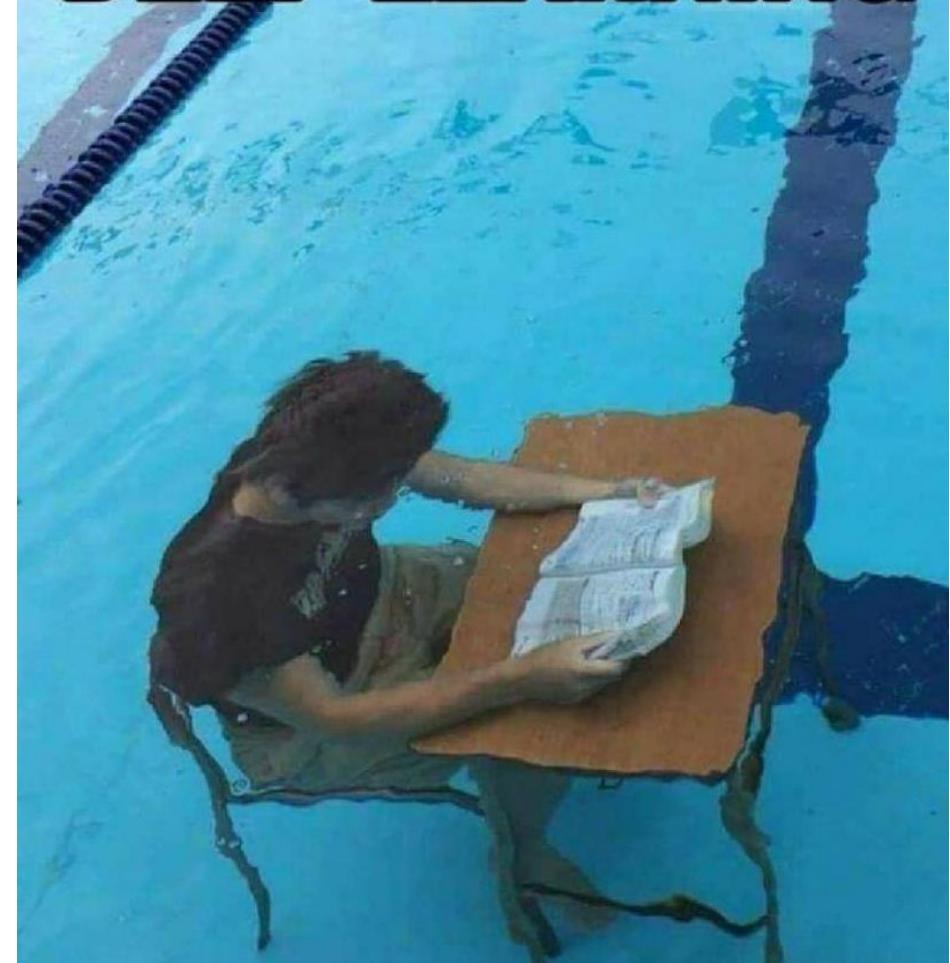


# 왜 많고 많은 모델 중 Tree Model?

1. 정형 데이터에서 기가 막히게 잘 맞추는 모델이 Tree Model!!, 해석하기도 좋고 baseline 모델로 접근 했을 때에도 성능이 괜찮음
2. 머신러닝 모델들은 기본적으로 독립변수들을 서로 독립적이라고 가정하고 접근
3. 어떠한 특별한 가정을 세운다 → 가정을 기반으로 여러 변수를 만든다 → 그 모델이 성능에 많은 기여를 한다 → 해석하기에 용이
4. 딥러닝 딥러닝 노래를 부르는 시대!(확실히 부흥기) 하지만, 캐글이나 머신러닝 대회에서는 boosting 계열 알고리즘이 석권!
5. 딥러닝 모델은 층도 많고, 오버피팅도 심하고 학습 시간 오래걸리고 장비도 좋아야하고.... (가난한 학생인데...)



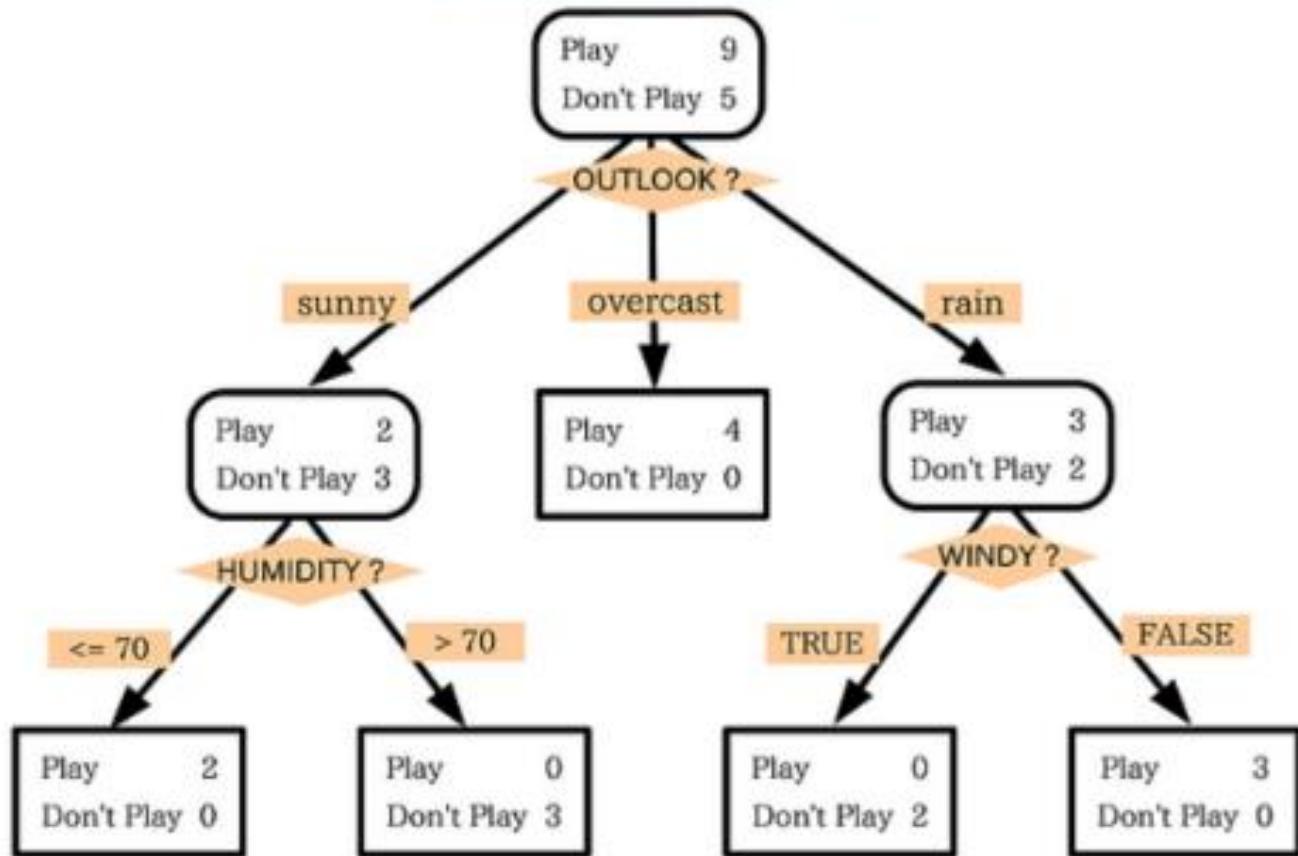
# DEEP LEARNING



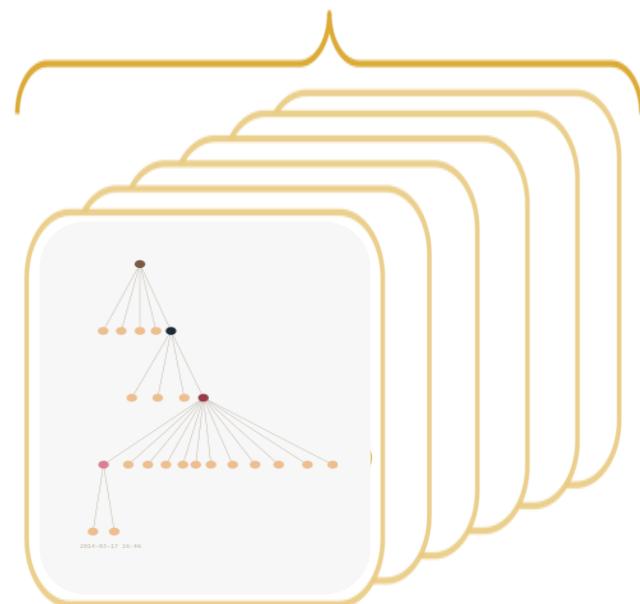
# 의사결정나무 Remind

- [https://ratsgo.github.io/machine learning/2017/03/26/tree/](https://ratsgo.github.io/machine-learning/2017/03/26/tree/)
- 주요 특징
  1. 변수들로 기준("수 많은 변수 중에 ??에 가장 영향을 주는 변수가 무엇일까?)을 만들고, 이것을 통하여 샘플을 분류하고 분류하고 집단의 성질을 통해 추정하는 모형
  2. 의사결정나무는 규칙들을 표현 → 규칙은 문장 형태로 표현될 수 있다.(and 조건을 이어나가)
  3. 스무고개 같음(예/아니오의 연속 질문 → 처음 질문의 답에 따라서 다음 질문이 달라짐)

Dependent variable: PLAY



Decision Forest



- 장단점
  - 장점 1 : 신경망, 판별분석, 회귀분석 보다 이해하기 쉽고(해석하기 쉽고) 설명력이 높다, 직관적
  - 장점 2 : 자료 가공이 조금 상대적으로 적음
  - 장점 3 : 비선형적인 관계를 설명 가능(교호작용 등이 반영)
  - 장점 4 : Target이 연속형, 범주형 모두 설명 가능
  - 단점 : 변동성이 너무 큼, 샘플에 민감

- 구성요소는 다 아시죠? → <https://dreamlog.tistory.com/576>
- 어떻게 Fitting? (비용복잡도 Cost Complexity를 기준으로 가지치기)
  - 주요 알고리즘 및 지표

- 주요 알고리즘 및 지표
    - Gini impurity(지니 불순도)

- CART 알고리즘에서 활용
    - 특정 그룹에 이질적인 것이 얼마나 섞였는지를 계산하는 지표
      - 0에 가까울 수록 순도가 높음

$$I_G(p) = \sum_{i=1}^J p_i \sum_{k \neq i} p_k = \sum_{i=1}^J p_i (1 - p_i) = \sum_{i=1}^J (p_i - p_i^2) = \sum_{i=1}^J p_i - \sum_{i=1}^J p_i^2 = 1 - \sum_{i=1}^J p_i^2$$

- Entropy, Information Gain(의사결정나무에서의 특정 노드 이전과 이후의 엔트로피 차이)

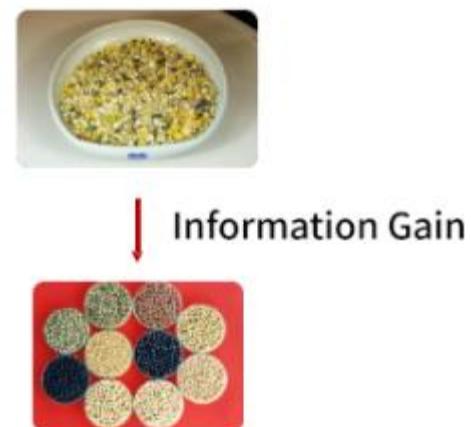
- 엔트로피

- 고등학교 과학시간에 '무질서의 정도', 골고루 섞여 있으면 높다.
- 정보이론에서는 정보량의 크기(해가 서쪽에서 뜬다)

$$H(T) = I_E(p_1, p_2, \dots, p_J) = - \sum_{i=1}^J p_i \log_2 p_i$$

- Information Gain(정보 획득량) : 특정 그룹의 엔트로피와 분할된 하위 그룹의 엔트로피 차이를 계산

- IG 값이 클 수록 '변별력이 좋다'(지정된 속성이 얼마나 잘 sample간을 구분하는 가에 대한 수치)
- 특정 변수를 기준으로 sample을 구분할 때 '감소되는 엔트로피의 양'

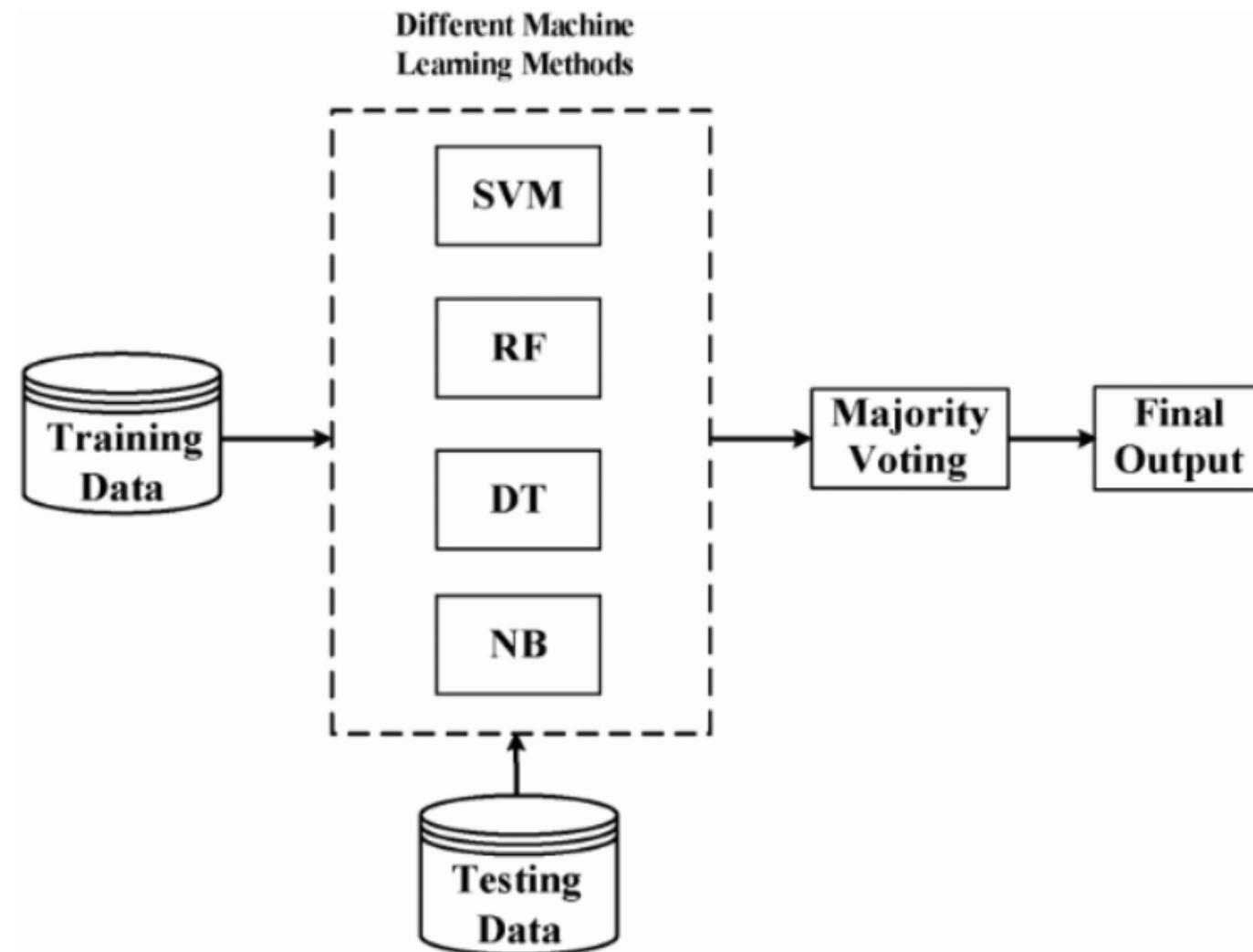


$$\overbrace{IG(T, a)} = \overbrace{H(T)} - \overbrace{H(T|a)}$$

$$= - \sum_{i=1}^J p_i \log_2 p_i - \sum_a p(a) \sum_{i=1}^J -Pr(i|a) \log_2 Pr(i|a)$$

# Ensemble

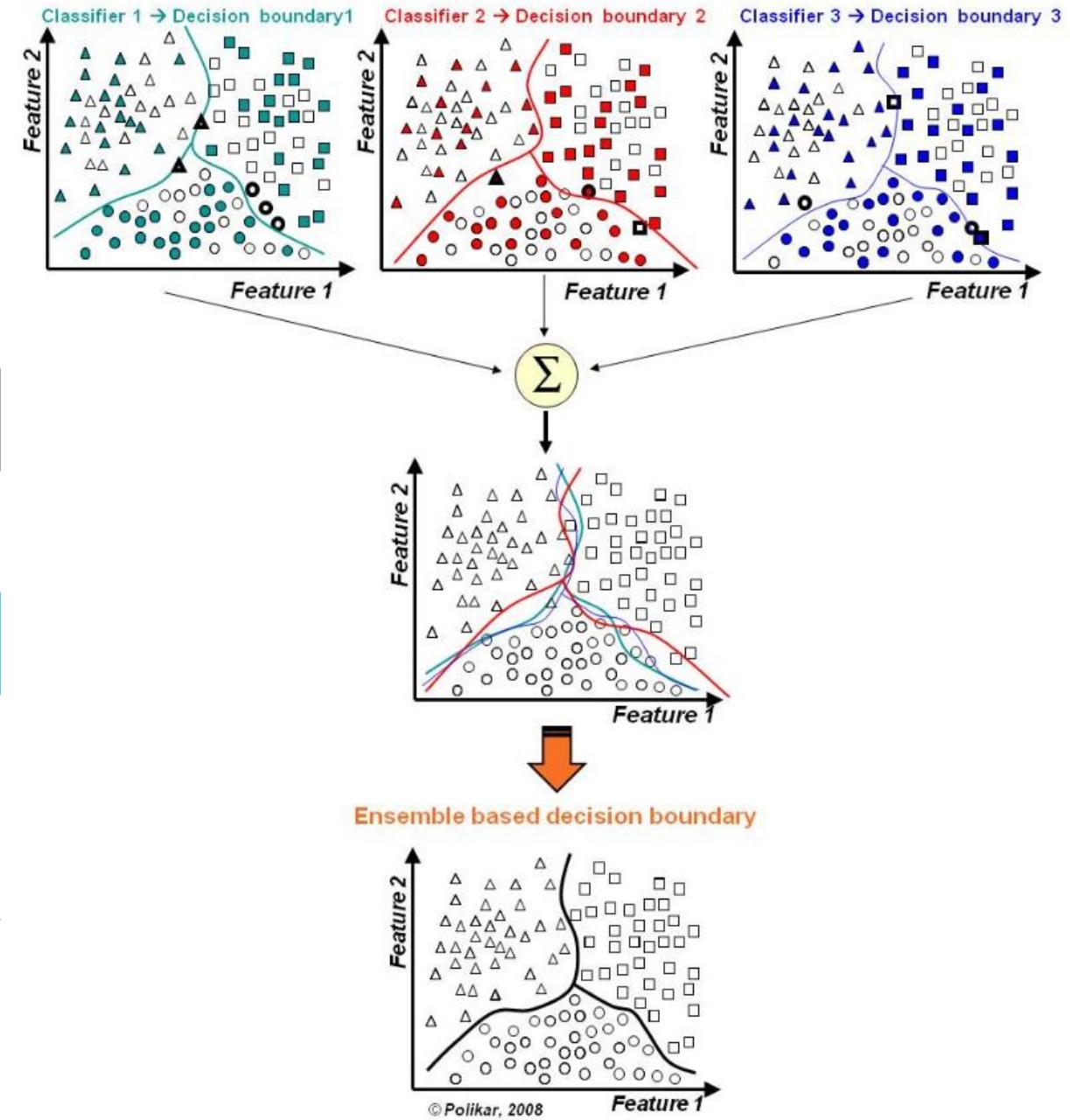
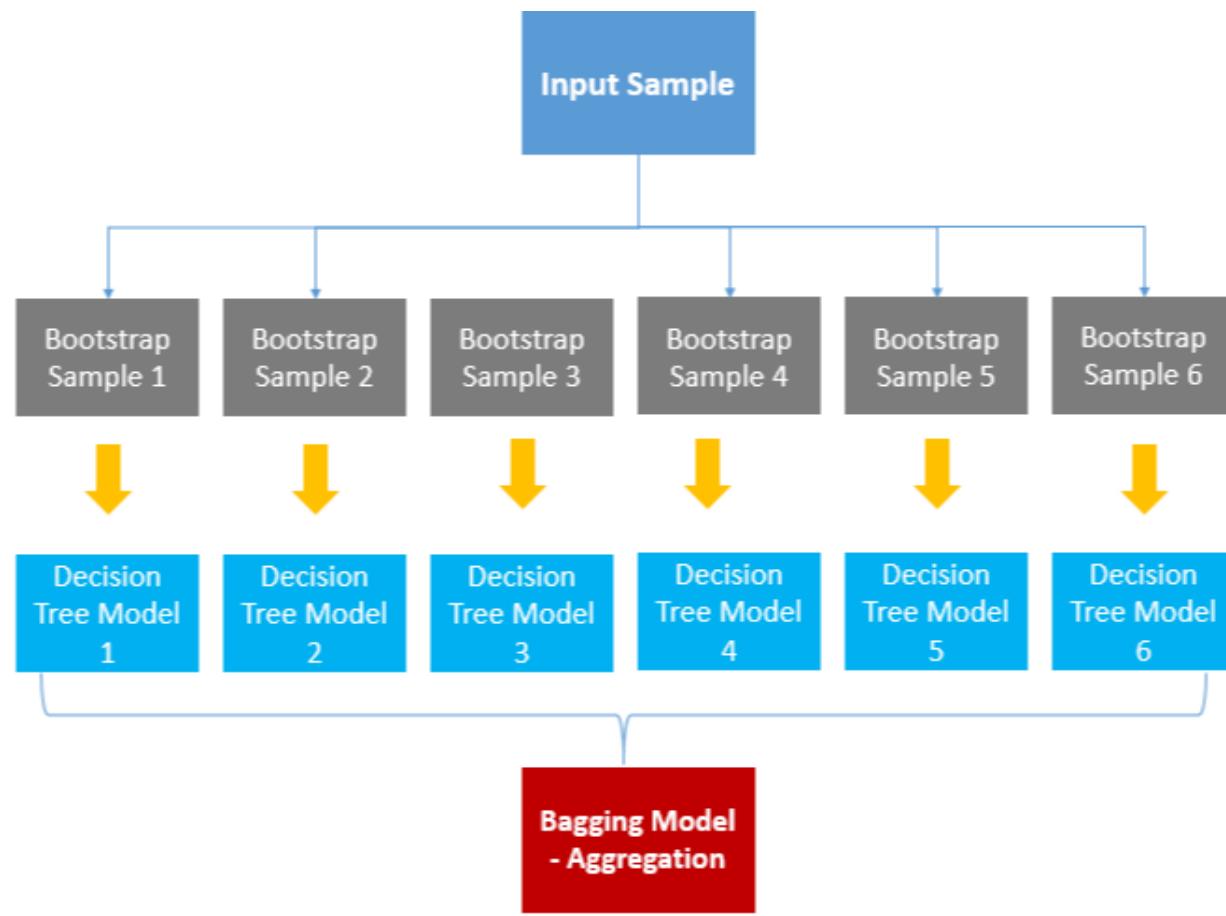
- Ensemble : '조화'라는 의미
- Ensemble Learning : 여러개의 기본 모델(weak learner, classifier 등)을 활용하여 하나의 새로운 모델을 만들어 내는 개념



- Ensemble Learning 종류
  - Bagging : 모델을 다양하게 만들기 위해 데이터를 재구성(모든 변수 사용, 샘플링)
  - RandomForest : 배깅이랑 산출방식은 같은데 변수도 샘플링함
  - Boosting : 맞추기 어려운 데이터에 대해 좀더 가중치를 두어 학습하는 개념(오답노트)
    - 경험상 : LightGBM > Catboost > XgBoost > GBM > AdaBoost
  - Stacking : 모델의 output값을 새로운 독립변수로 사용(PCA loading, SVM, KNN Etc)

## Bagging(Bootstrap AGGREGATING)

- **복원추출**로 데이터 추출하여 여러 분류기를 만들어 결과값을 평균으로 산출
- 깊이 들어간 의사결정나무의 **단점** : 분산 증가, 편향 감소
- **Bagging 장점** : Tree들의 편향 유지, 분산 감소
- **Bagging 단점**(모든 양상을 모형의 단점) : 모형 해석 어려움



## RandomForest

- Bagging Model의 분산은 각각 트리들의 분산과 트리들의 공분산으로 이루어 져 있음
- 전체 데이터에서 복원 추출했으나, 각각의 트리들은 중복되는 데이터를 다수 가지고 있기 때문에 독립이라는 보장이 없다  $\rightarrow 2\text{Cov}(X, Y) \neq 0$

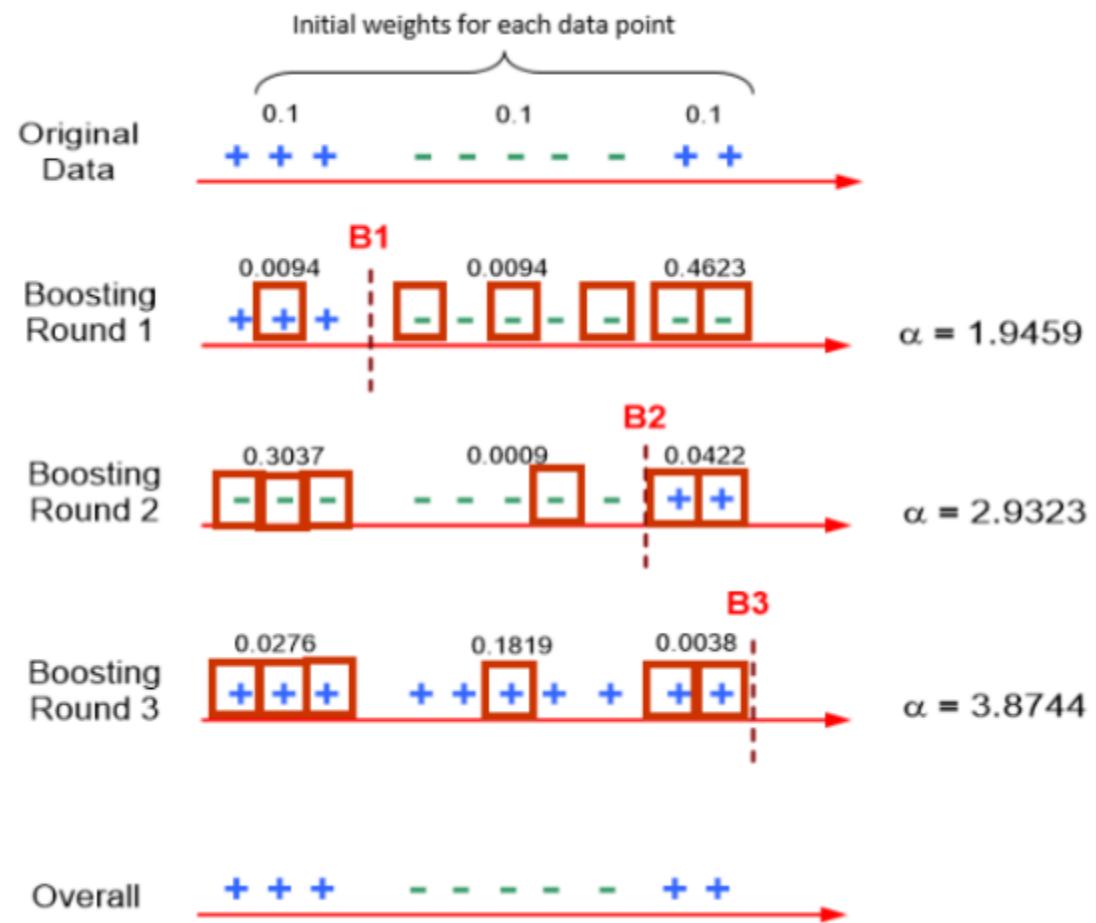
$$\text{Var}(X + Y) = \text{Var}(X) + \text{Var}(Y) + 2\text{Cov}(X, Y)$$

- Tree가 증가함에 따라 오히려 모델 전체의 분산이 증가 할 수 있다.  $\rightarrow$  각 분류기간 공분산을 줄일 방법이 필요
- RF의 핵심은 데이터 샘플링 + 변수도 Random하게( $\text{sqrt}(p)$ 개) 추출해 분류기를 생성함  $\rightarrow$  모델간의 공분산 감소시키는 것이 컨셉
- 일반적으로 뽑을 변수의 수는  $\text{sqrt}(p)$  사용  $\rightarrow$  모델의 분산을 줄여 일반적으로 Bagging보다 성능이 좋음

## Boosting - Adaboost(Adaptive Boost)

- (공통) 오분류된 데이터에 초점을 맞추어 더 많은 가중치를 주는 방식
  - 공통이라고 하면 AdaBoost, GBM, LogitBoost, TotalBoost, LPBoost, MadaBoost .... ETC
- 초기에는 모든 데이터가 동일한 가중치 → Round 종료 후 가중치와 중요도 계산
- 복원추출시에 데이터 가중치 분포 고려
- 오분류된 데이터에 Weight를 크게, 정분류된 데이터에는 Weight를 작게 설정 → Weight별로 샘플을 재수집하며 계속 Weight를 업데이트(Round별로)!

- ① 모든 데이터에 대해 가중치를 동일하게 0.1로 설정
- ② Round 1에서 빨간색 네모의 데이터가 수집되며 이를 기반으로 분류 기준값인 B1을 설정(B1보다 작은 경우 +, 큰 경우 -로 분류)
- ③ 데이터  $i$ 에 대해  $m$ 번째 round에서의 가중치를 업데이트 (오분류된 데이터에 가중치를 크게, 정분류된 데이터에 가중치를 작게 설정)
- ④ 업데이트한 가중치의 확률로 샘플을 재수집
- ⑤ 4번의 결과로 Round 2의 빨간색 네모의 데이터가 수집
- ⑥ 수집된 데이터로 모형을 학습한 결과 B2가 분류 기준값으로 도출됨
- ⑦ 가중치를 업데이트
- ⑧ 이와 같은 방법을 설정한 반복 횟수만큼 반복
- ⑨ 예시에서 Round 3까지의 결과를 종합하면 Original Data 와 동일한 결과가 도출됨



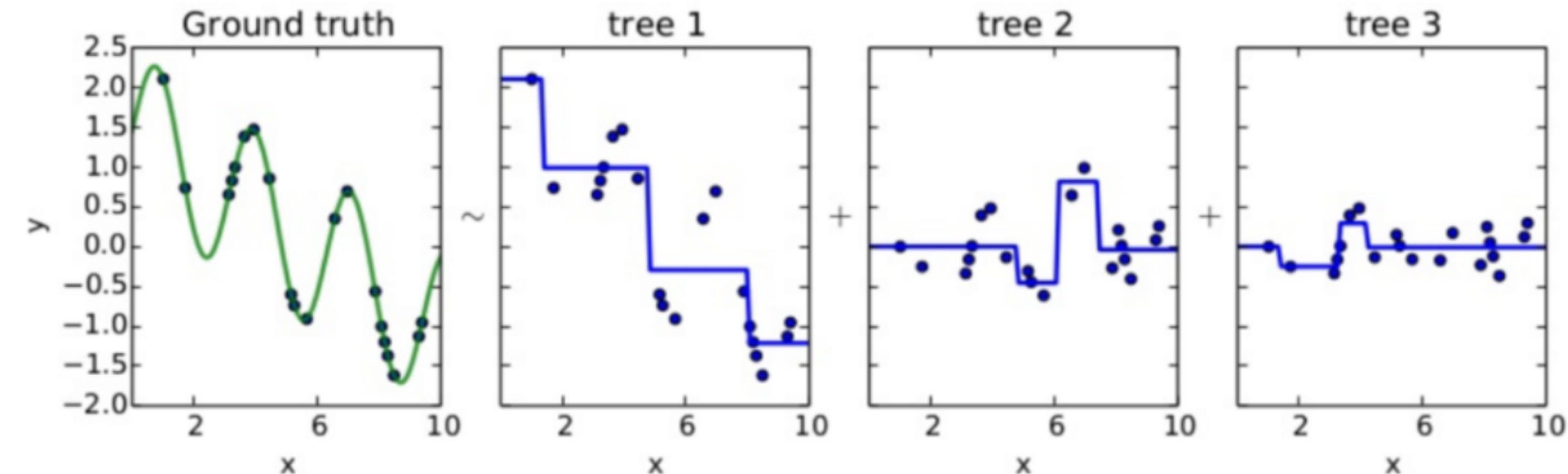
- 최종 모델 결정 방법

$$H(x) = \text{sign}\left\{\sum_{m=1}^m \alpha_m h_m(x)\right\}$$

- $H(x)$  : 최종 분류기(final classifier)
- $h_m$  : m라운드에서 생성된 약한 분류기
- $\alpha_m$  : m라운드에서 생성된 약한 분류기에 대한 가중치(크면  $\epsilon_m$ 이 작다는 의미 → 분류기 m이 좋은 성능을 보임)

## Boosting - GBM(Gradient Boosting Machine)

- Boosting 기법들끼리 큰 차이점 : **오분류된 데이터를 다음 Round에 무슨 짓을 할까???** (AdaBoost는 오분류된 데이터들에 더 큰 가중치를 주어서 샘플링에 힘을 주었음)
- residual에 대해 계속 학습해 나가면서 error를 최소화(<https://3months.tistory.com/368>)



$x$ 를 입력받아  $y$ 를 예측하는 모델  $h_0$ 가 있다고 하면,

$$y = h_0(x) + \text{error}$$

$$\text{error} = h_1(x) + \text{error2}$$

$$\text{error2} = h_2(x) + \text{error3}$$

$$\text{error3} = h_3(x) + \text{error4}$$

(고등학교 수열시간에 많이 보신 형태!!)

$$y = h_0(x) + h_1(x) + h_2(x) + \cdots + \text{small error}$$

- Error를 많이 많이 줄여보자~

## What is gradient?

- Define loss func

$$L(y_i, f(x_i)) = 0.5 * (y_i - f(x_i))^2$$

- Loss func의 gradient

$$\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)} = \frac{\partial [\frac{1}{2}(y_i - f(x_i))^2]}{\partial f(x_i)} = f(x_i) - y_i$$

- Negative gradient = Residual

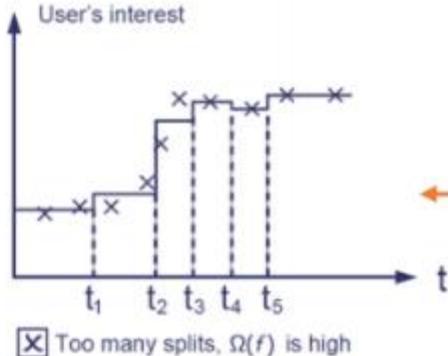
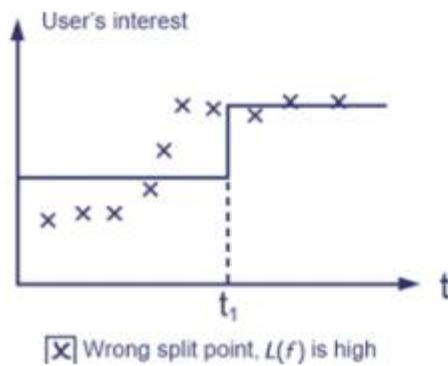
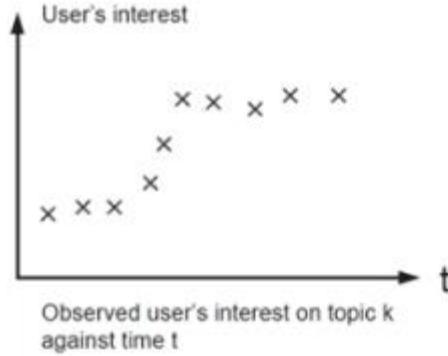
$$-(f(x_i) - y_i) = y_i - f(x_i)$$

- Negative gradient를 최소화 시키면서 학습 시키기 때문에 gradient boosting이라 부름

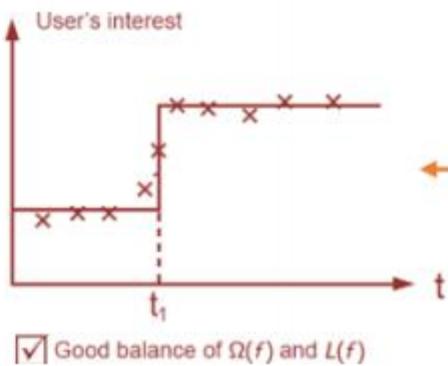
## Boosting - XgBoost(eXtreme Boosting)

- 기존 GBM이랑 큰 차이 없음. 단지 목적식에 Regularization Term이 추가됨(Ridge, Lasso와 같은)  
→ 오버피팅을 잡아주자!!!
- 이 알고리즘 때문에 옛날 캐글에서 꽤나 우수한 성적이 나옴
- 왼쪽 Term : Loss Function
- 오른쪽 Term : Regularization

$$obj^{(t)} = \sum_{i=1}^n l(Y_i, \hat{Y}_i^{(t)}) + \sum_{i=1}^t \Omega(f_i)$$

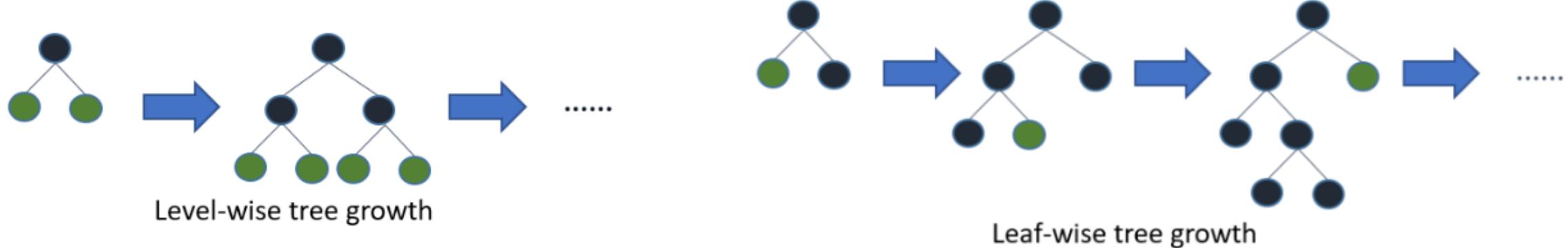


**Gradient Boosting**



**XGBoost**

## Boosting - LightGBM



- 넘 좋고 달달~!! **동일한** hyperparameter에서 Xgboost보다 약 2배 이상 빠름
- 이전 boosting기법은 **균형 트리 분할**(level wise) → LGBM은 **리프 중심 트리 분할**(leaf wise)가 큰 컨셉
- 공식 문서에서 데이터 **최소 10000개 이상 권장**(오버피팅이 끝내주게 잘 일어남), 학습데이터 많을 수록 좋다.
- gpu애매하면 cpu로 돌리는 거 권장(경험상)
- h2o에서 xgboost에서 옵션 조금만 바꿔서 lgbm을 사용. (`tree_method`, `grow_policy`)

```
# In h2o, there is no specific module for LightGBM yet.  
# Instead, using "tree_method='hist'" and "grow_policy='lossguide'" in h2o.xgboost() offer "LightGBM algorithm" to us  
# you can find detail information in h2o website => http://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/xgboost.html  
# parameter = default  
start.time <- Sys.time()  
ml_LGB <- h2o.xgboost( training_frame = train,  
                        validation_frame = valid,  
                        x=1:19,  
                        y=20,  
                        tree_method="hist",  
                        grow_policy="lossguide",  
                        seed = 1234)  
  
Sys.time() - start.time # check time  
  
# check model performance by using test data set  
# AUC = 0.9234419  
# LogLoss = 0.1694313  
h2o.performance(ml_LGB, newdata = test)
```

## Boosting - Catboost(unbiased boosting with categorical features, 2017)

- 잔차 추정의 분산을 최소로 하면서 bias도 피하는 기법
- 범주형 변수가 많으면 꽤나 좋은 효과를 발휘함(캐글에서 종종보임)
- 관측치를 포함한 채로 boosting하지말고, 관측치를 뺀채로 학습 → 그 관측치에 대한 unbiased residual을 구하고 학습하는 컨셉
- 범주형 변수를 One-hot으로 바꾸는 방식이 아닌, Numeric하게 변환하는 방법 제안!
- 논문(<https://arxiv.org/abs/1706.09516>, Cat > LGBM > Xgboost)

Table 8: Comparison with baselines: logloss / zero-one loss, relative increase is presented in the brackets.

	CatBoost	LightGBM	XGBoost
Adult	<b>0.2695 / 0.1267</b>	0.2760 (+2.4%) / 0.1291 (+1.9%)	0.2754 (+2.2%) / 0.1280 (+1.0%)
Amazon	<b>0.1394 / 0.0442</b>	0.1636 (+17%) / 0.0533 (+21%)	0.1633 (+17%) / 0.0532 (+21%)
Click	<b>0.3917 / 0.1561</b>	0.3963 (+1.2%) / 0.1580 (+1.2%)	0.3962 (+1.2%) / 0.1581 (+1.2%)
Epsilon	<b>0.2647 / 0.1086</b>	0.2703 (+1.5%) / 0.114 (+4.1%)	0.2993 (+11%) / 0.1276 (+12%)
Appetency	<b>0.0715 / 0.01768</b>	0.0718 (+0.4%) / 0.01772 (+0.2%)	0.0718 (+0.4%) / 0.01780 (+0.7%)
Churn	<b>0.2319 / 0.0719</b>	0.2320 (+0.1%) / 0.0723 (+0.6%)	0.2331 (+0.5%) / 0.0730 (+1.6%)
Internet	<b>0.2089 / 0.0937</b>	0.2231 (+6.8%) / 0.1017 (+8.6%)	0.2253 (+7.9%) / 0.1012 (+8.0%)
Upselling	<b>0.1662 / 0.0490</b>	0.1668 (+0.3%) / 0.0491 (+0.1%)	0.1663 (+0.04%) / 0.0492 (+0.3%)
Kick	<b>0.2855 / 0.0949</b>	0.2957 (+3.5%) / 0.0991 (+4.4%)	0.2946 (+3.2%) / 0.0988 (+4.1%)

Table 9: Comparison with baselines: logloss / zero-one loss (relative increase for baselines).

	Raw setting of CatBoost	LightGBM	XGBoost
Adult	0.2800 / 0.1288	-1.4% / +0.2%	-1.7% / -0.6%
Amazon	0.1631 / 0.0533	+0.3% / 0%	+0.1% / -0.2%
Click	0.3961 / 0.1581	+0.1% / -0.1%	0% / 0%
Appetency	0.0724 / 0.0179	-0.8% / -1.0%	-0.8% / -0.4%
Churn	0.2316 / 0.0718	+0.2% / +0.7%	+0.6% / +1.6%
Internet	0.2223 / 0.0993	+0.4% / +2.4%	+1.4% / +1.9%
Upselling	0.1679 / 0.0493	-0.7% / -0.4%	-1.0% / -0.2%
Kick	0.2955 / 0.0993	+0.1% / -0.4%	-0.3% / -0.2%
Average		-0.2% / +0.2%	-0.2% / +0.2%

## 여러분들한테 당부의 말씀

- 공모전(정형데이터)을 참가하실텐데 **오늘 소개해준 모델로 baseline** 잡아서 하시는 것을 추천합니다!
- 저는 많이 쓰이는 모델들을 소개해주었습니다. → 상을 펴서 **수저만 차린 수준** → 알아서 **공식문서** 보면서 해보는 것을 추천합니다. 공식문서랑 친해지세요!!!
- 방법론보다는 더 중요한 것은 **전처리 전처리 전처리**가 중요합니다. 실제로 여러분들이 하는 프로젝트에서 모델링은 5~10%입니다.
- 공모전(데이콘, 캐글, 카카오아레나 등) **상위권 코드**를 보면서 여러명 모여서 하향식 공부법을 추천합니다.
- 혹시나 (머신러닝)논문에 쓰시는 데이터가 너무 안좋다면 한번 이걸로 실험해볼 수는 있겠습니다.
- Stacking이라고 또 다른 기법이 있는데 시간이 기가 막하게 걸리기 때문에 컴퓨터 사양이 자신 있는 분들은 따로 공부해보시는 거 추천합니다.

# Contents

---

## 1st session

- Preprocessing - dplyr, data.table

## 2<sup>nd</sup> session

- Tree model

## 3<sup>rd</sup> session

- **h2o tutorial**

## 4<sup>th</sup> session

- XAI : Local Interpretable Model-agnostic Explanations

# 대부분 이렇게 합니다!



## Data Load

- Data Load
- Data Integration (Merge)



## Exploratory Data Analysis

- Basic Statistics
- Initial Data Pre-processing
- Initial Modeling
- EDA using Caret
- Uni-variate
- Relation to Y



## Data Pre-Processing

- Scaling
- Imputation
- Feature Transformation
- Dimension Reduction
- Feature Selection
- Sampling



## Model Development

- Single Model
- Grid Search
- Parallel Computing



## Predict & Performance Check

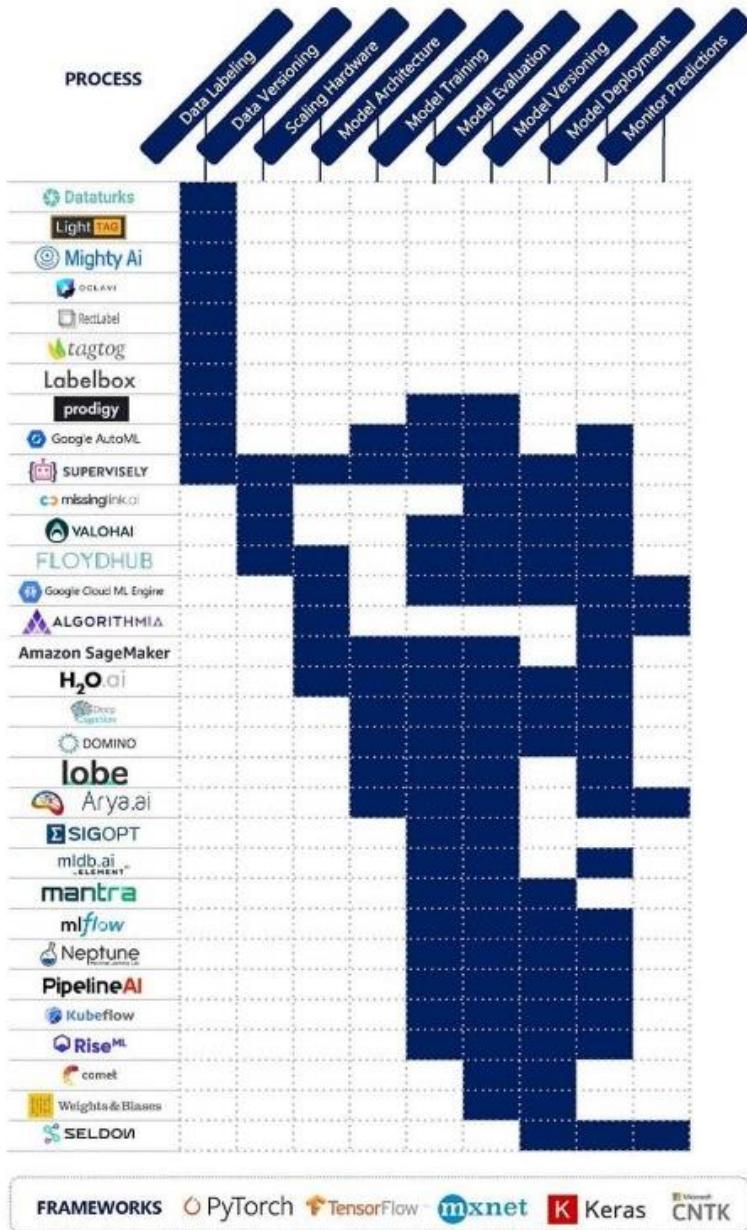
- Predict
- Performance Metric
  - ROC, Lift
- Business Performance



## Deploy

- API Development
- Mojo / Pojo (H2O)

## DEEP LEARNING TOOLS



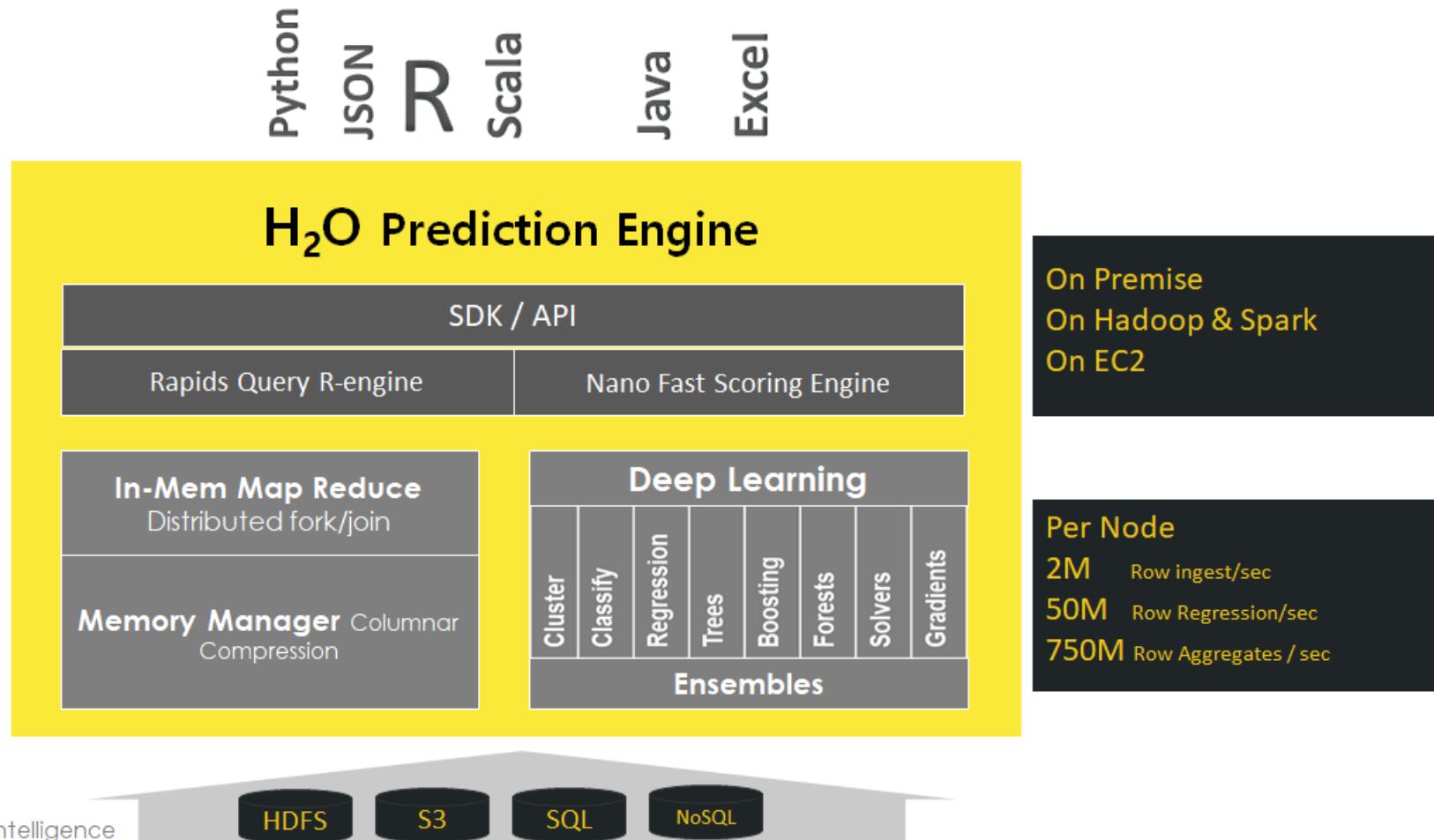
Tool이 과하게 많은 시대!!

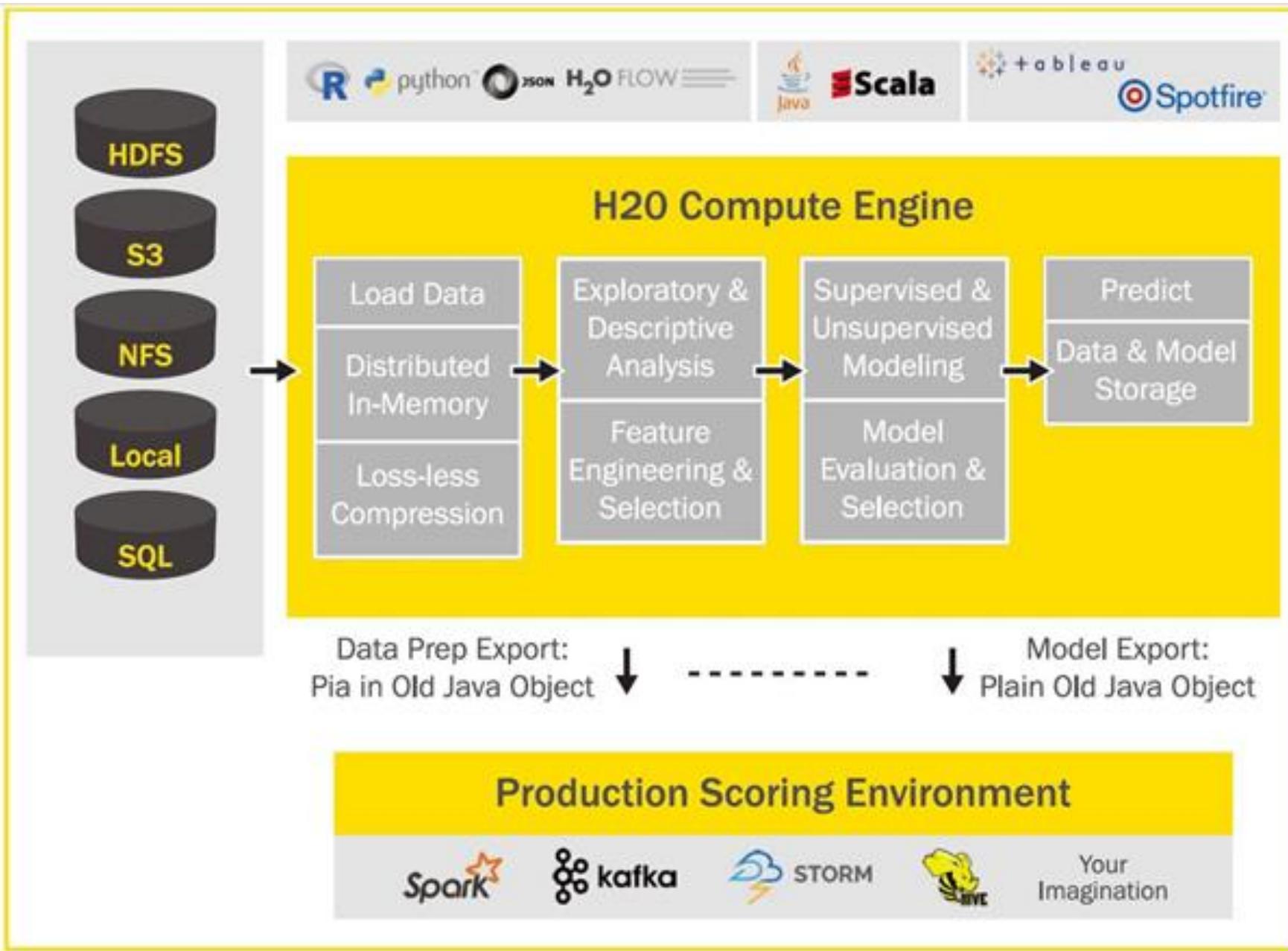
## h2o.ai Overview

- **Founded:** 2011 venture-backed, debuted in 2012
- **Product:** H2O open source in-memory prediction engine
- **Team:** 30
- **HQ:** Mountain View, CA
- **SriSatish Ambati** – CEO & Co-founder (Founder Platfora, DataStax; Azul)
- **Cliff Click** – CTO & Co-founder (Creator Hotspot, Azul, Sun, Motorola, HP)
- **Tom Kraljevic** – VP of Engineering (CTO & Founder Luminix, Azul, Chromatic)

# h2o architecture

- 머신러닝 방법론 중심의 알고리즘 framework
- Spark 활용 분산처리, GPU 병렬처리 등의 다양하게 지원
- DAI(Driverless AI) 등 비즈니스 수요 창출





# h2o를 이제 다루어보아요

## h2o 설치하려면

- java 1.7~1.8 필요함(java 기반의 어플리케이션으로 R과 별도로 구동합니다)
- 오라클 <https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

## 데이터의 형태를 바꾸어줘야합니다

- as.h2o()를 활용해서 object를 h2o 형태로 변환해주어야 합니다

## 모형 어떻게 fit?

- h2o로 변환된 데이터를 활용하여 모형을 적합합니다
  - h2o.randomForest()
  - h2o.gbm()
  - h2o.xgboost

## 모형 최적화 및 성능 평가

- 옵션을 활용한 각 모형 모수 변경 가능
  - h2o.grid()
- 주요 모수는 코드 보면서 이야기해요!

## 새로운 관측치에 대한 예측값 계산

- h2o.predict()

# Model 만들었으니 평가

분석의 목표치와 measurement를 정해야합니다.

performance measurement는 학교에서 잘 가르쳐 줄 내용들입니다.

- Regression

- MAE
- MAPE
- MSE
- RMSE

- Classification

- Logloss
- accuracy
- auc
- precision
- recall
- f1score

# 모형이 없을 경우, 대비 모델을 사용할 때 얼마나 더 개선 되는가?

## 문제 정의

100만명 의 고객 중



모형이  
없다면

매달 2만명이 이탈한다면  
전체 이탈률은 2%

이 고객들을 Care하기 위해  
매달 랜덤하게 10% (10만명)에게  
Contact 하고자 한다.

우리가 Random하게 선택한  
10만명 집단의 이탈률은 2%

2000명

Lift = 7배  
(향상률)

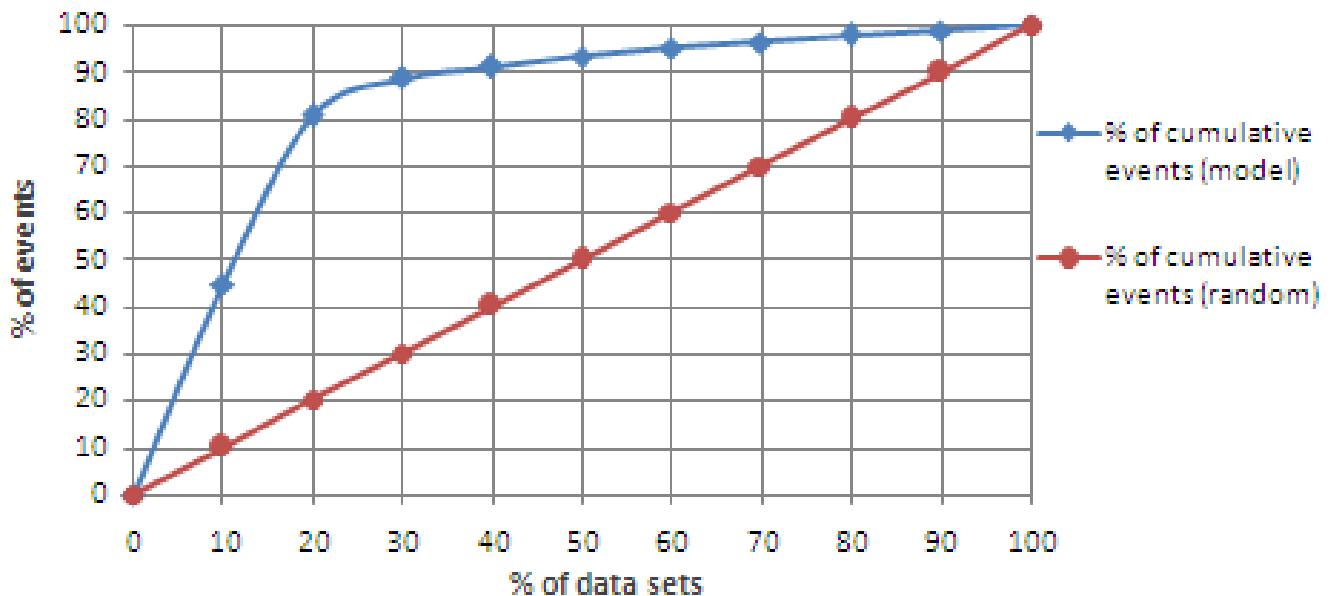
머신러닝  
모형을  
활용하면

머신러닝 모형을 위해 이탈 가능성  
높은 10만명을 선택했을 때  
그 집단내의 이탈률은 14%

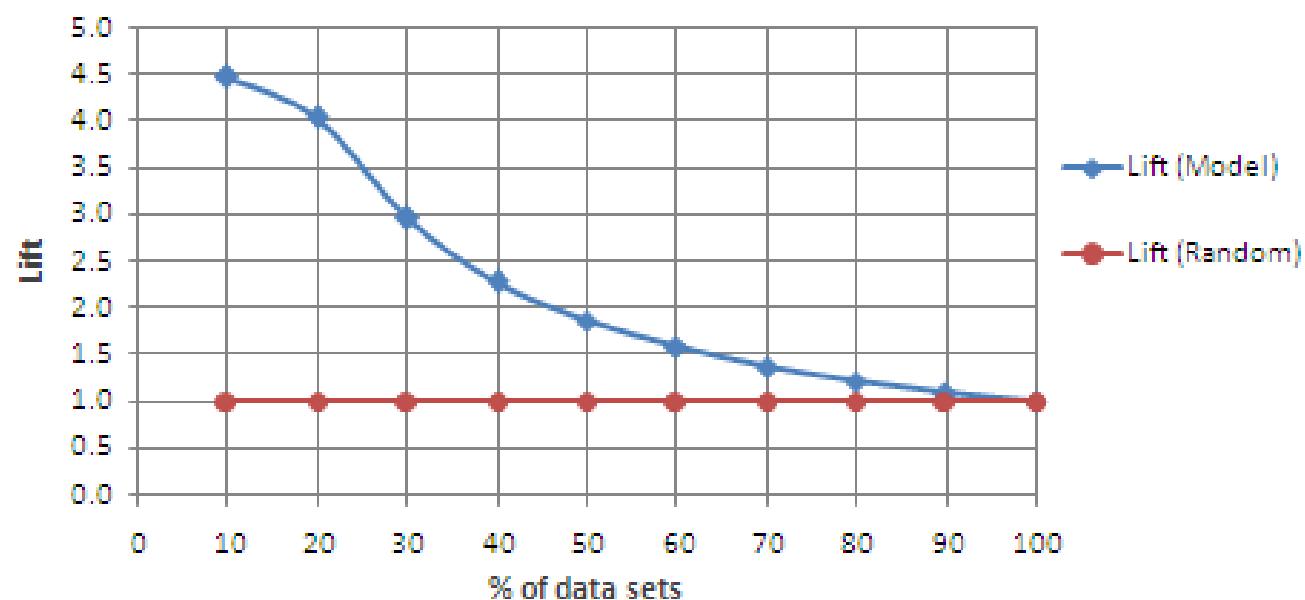
1.4만명

Decile	n	res	survRatio	random	lift	cumRes	cumRandom	cumLift
<fct>	<int>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	26	26	1	10.1	2.58	26	10.1	2.58
2	25	23	0.92	9.70	2.37	49	19.8	2.48
3	30	24	0.8	11.6	2.06	73	31.4	2.32
4	26	8	0.308	10.1	0.793	81	41.5	1.95
5	27	7	0.259	10.5	0.668	88	52.0	1.69
6	27	2	0.0741	10.5	0.191	90	62.5	1.44
7	26	3	0.115	10.1	0.297	93	72.6	1.28
8	26	6	0.231	10.1	0.595	99	82.7	1.20
9	28	2	0.0714	10.9	0.184	101	93.5	1.08
10	27	3	0.111	10.5	0.286	104	104.	1.00

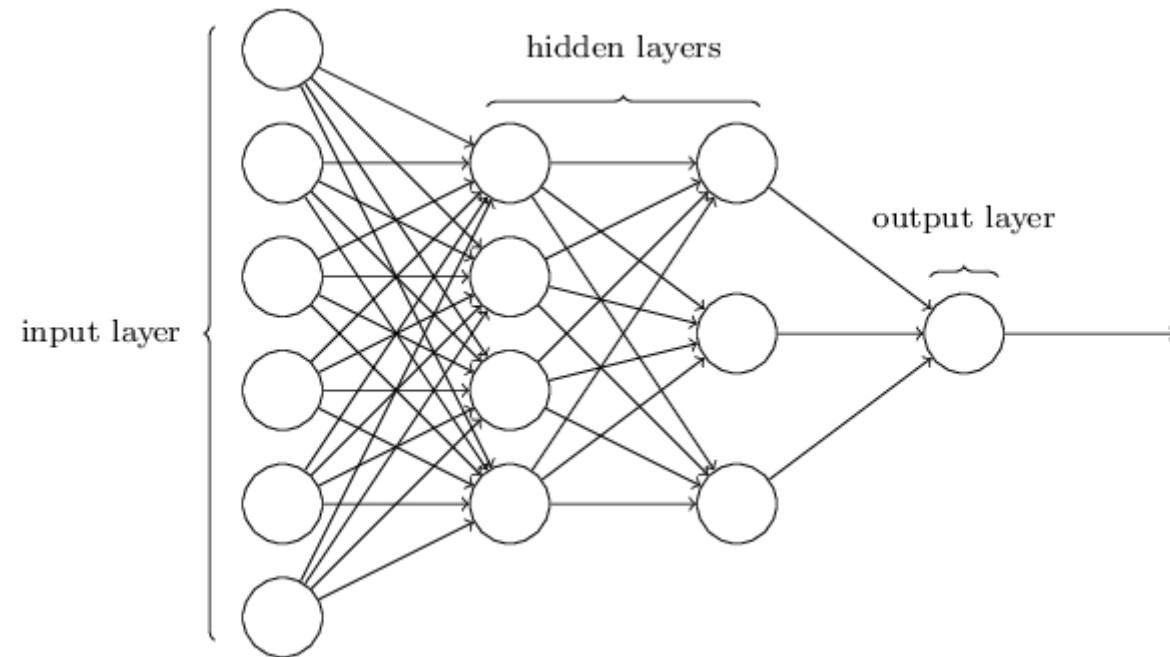
## Gain Chart



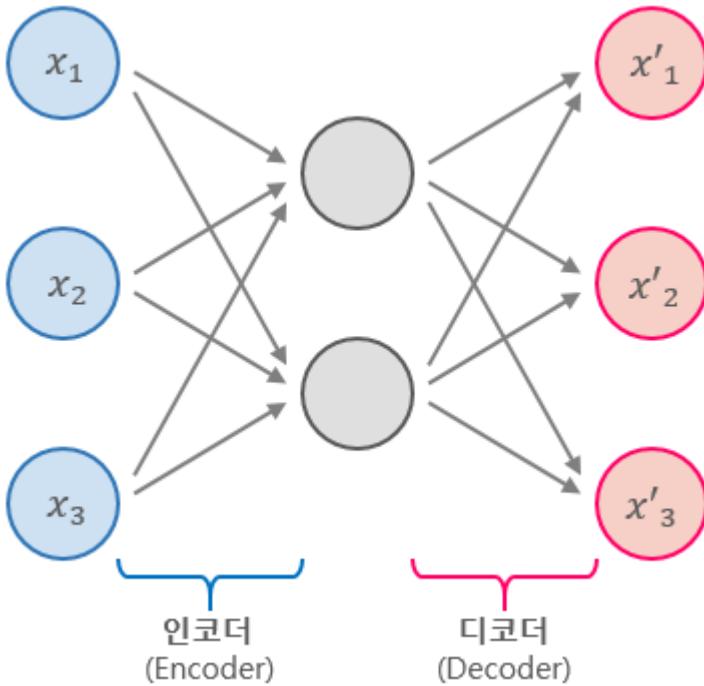
## Lift Chart



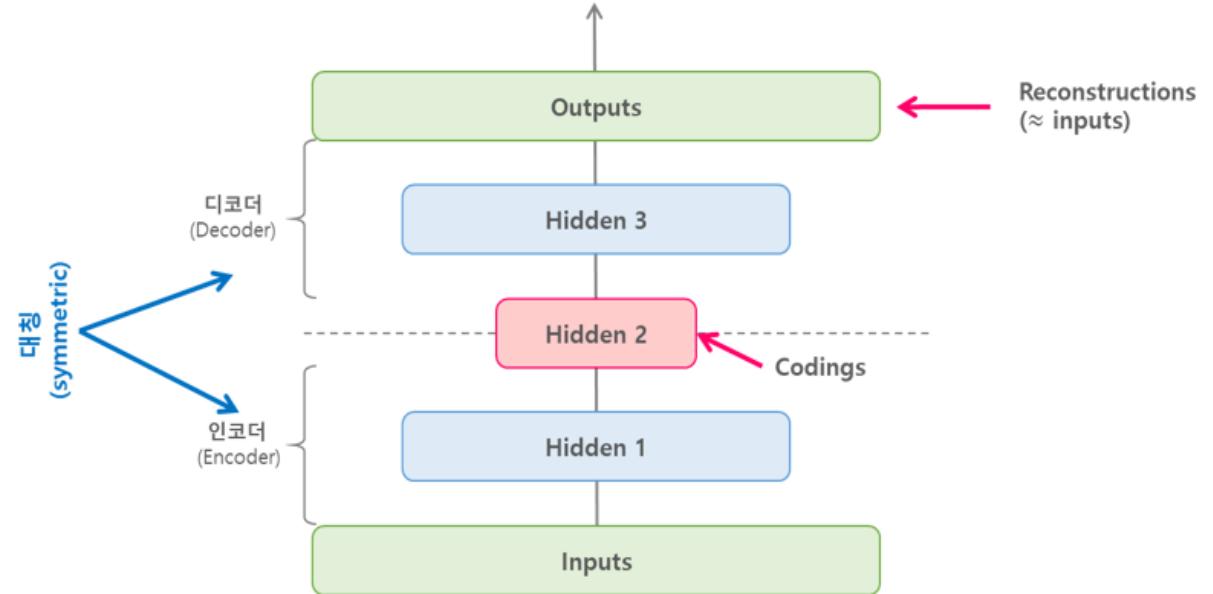
# Multi Layer Perceptron (코드 미리 돌리지 마세요!!)



## 오토인코더

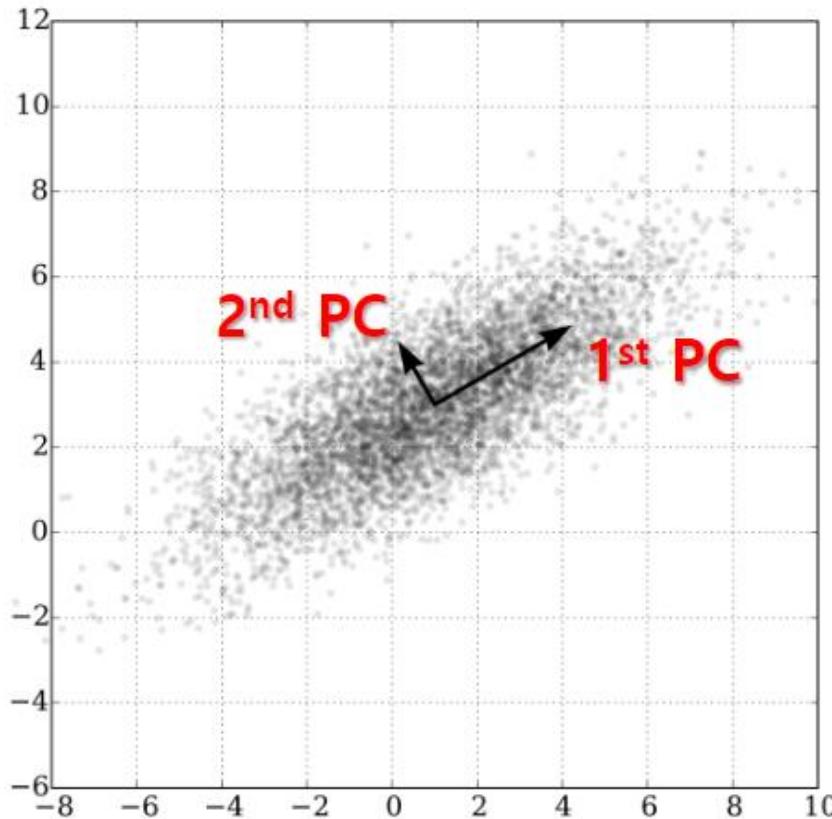


## Stacked 오토인코더

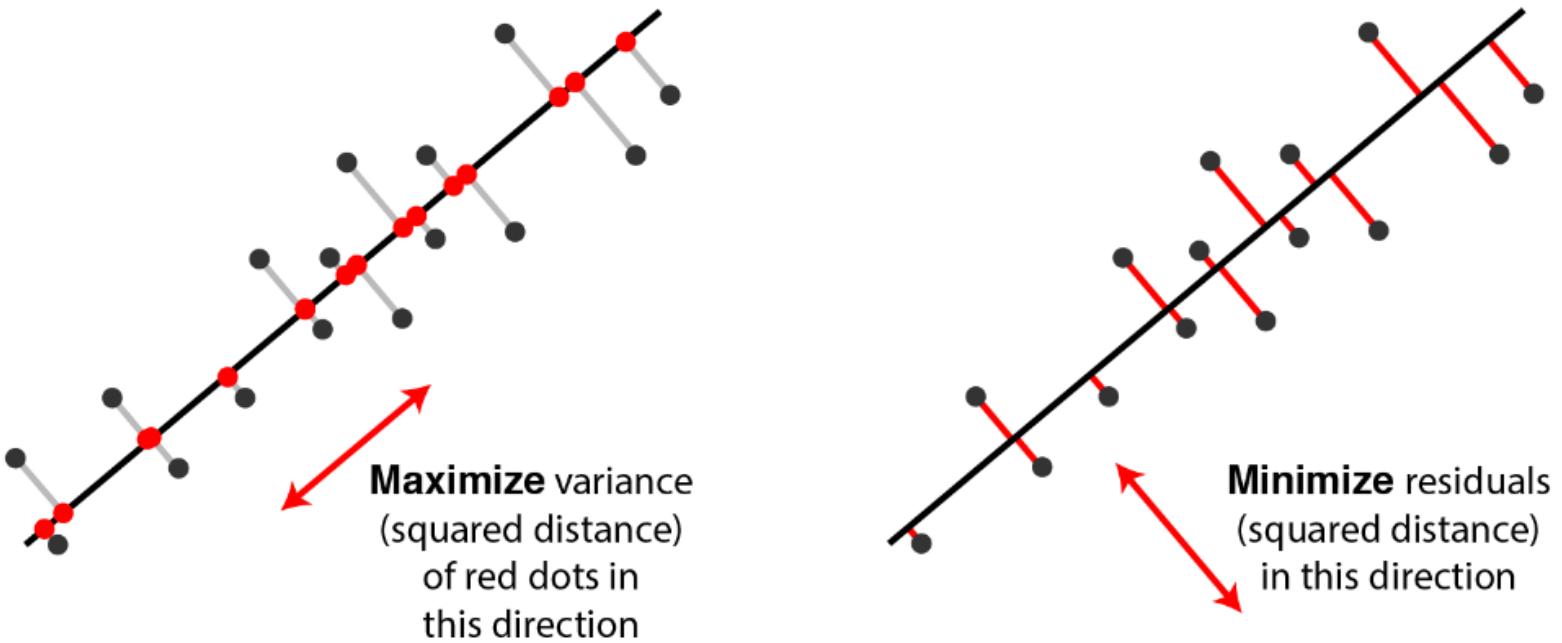


오토인코더(Autoencoder)는 그림과 같이 단순히 입력을 출력으로 복사하는 신경망이다. 어떻게 보면 간단한 신경망처럼 보이지만 네트워크에 여러가지 방법으로 제약을 줌으로써 어려운 신경망으로 만든다. 예를 들어 아래 그림처럼 hidden layer의 뉴런 수를 input layer(입력층) 보다 작게해서 데이터를 압축(차원을 축소)하거나, 입력 데이터에 노이즈(noise)를 추가한 후 원본 입력을 복원할 수 있도록 네트워크를 학습시키는 등 다양한 오토인코더가 있다. 이러한 제약들은 오토인코더가 단순히 입력을 바로 출력으로 복사하지 못하도록 방지하며, 데이터를 효율적으로 표현(representation)하는 방법을 학습하도록 제어한다.

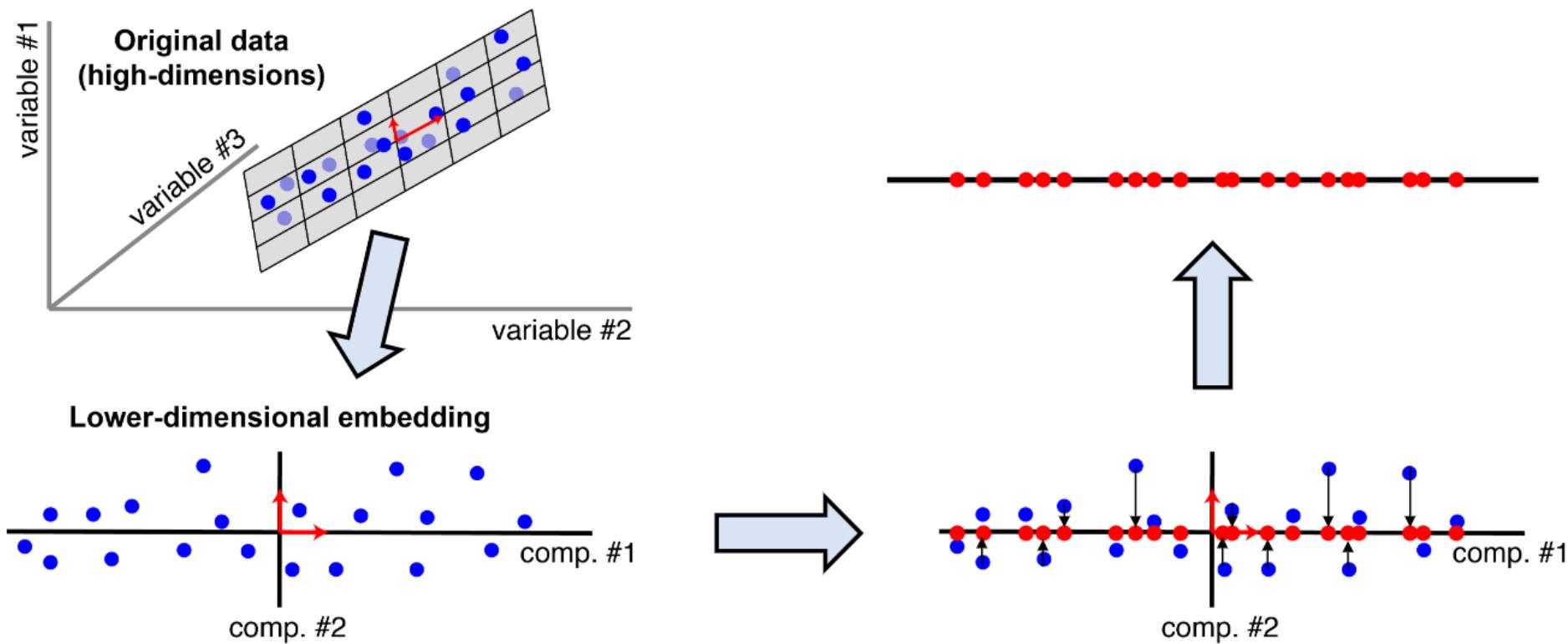
# PCA



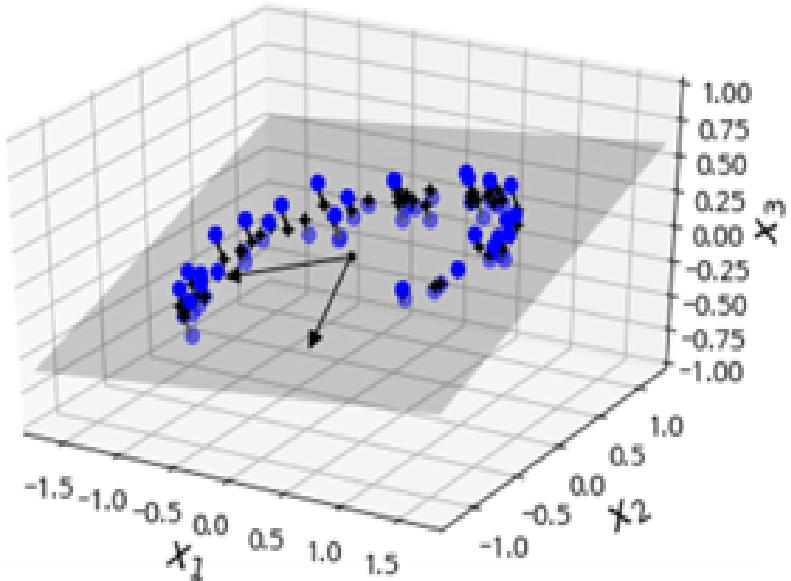
# PCA



# PCA

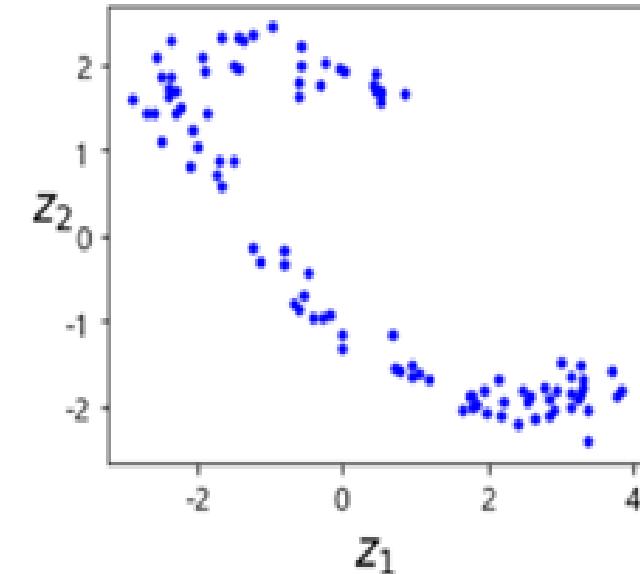


3D - Dataset



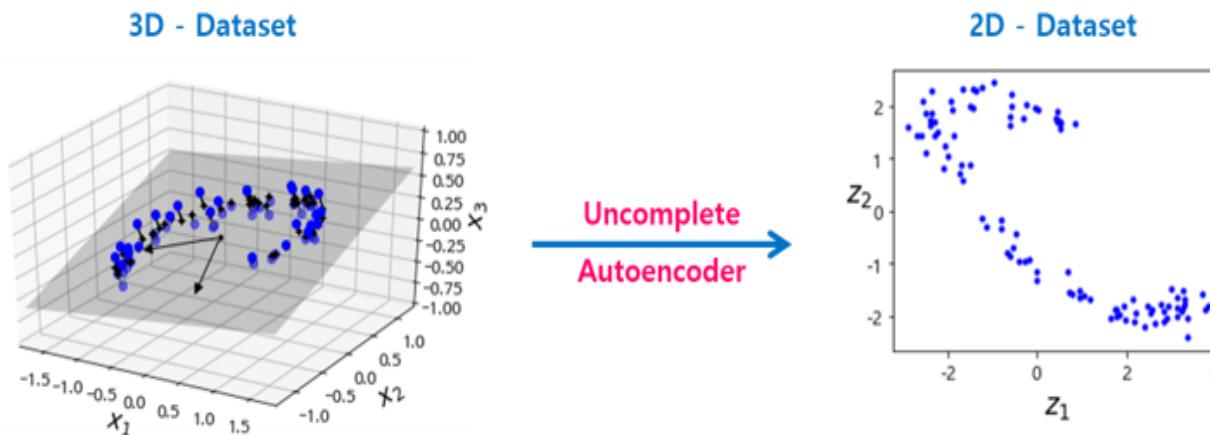
Uncomplete  
Autoencoder

2D - Dataset

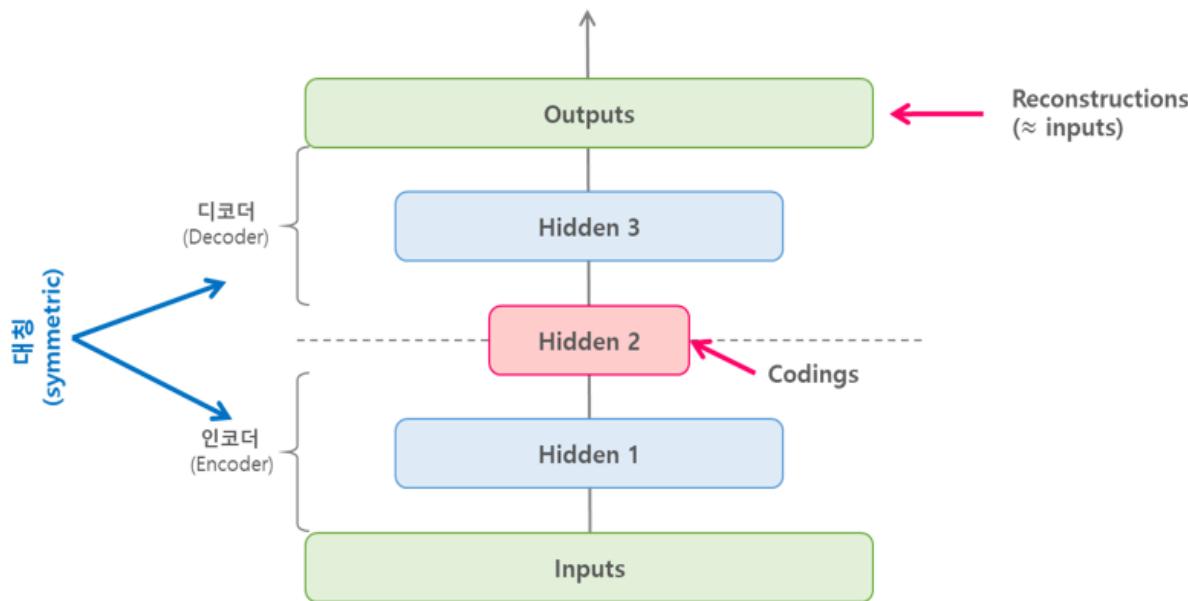


위에서 살펴본 Undercomplete 오토인코더에서 활성화 함수를 sigmoid, ReLU 같은 비선형(non-linear)함수가 아니라 선형(linear) 함수를 사용하고, 손실함수로 MSE(Mean Squared Error)를 사용할 경우에는 PCA라고 볼 수 있다.

DNN의 또 다른 적용은 차원 감소입니다.  
구체적으로는 형상 공간을 해당 공간의 비선형 변환으로 투영하는 것입니다.  
H2O에서 우리 `h2o.deepfeatures()`는 H2O 심층 학습 모델을 사용하여  
H2O 데이터 세트에서 비선형 피쳐를 추출 하는 함수를 사용할 수 있습니다.  
여기서는 첫 번째 숨겨진 레이어의 기능을 가져옵니다.

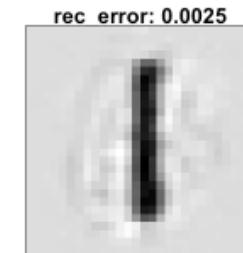
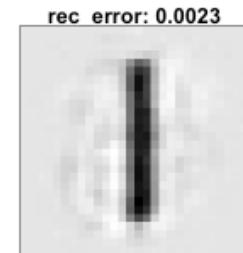
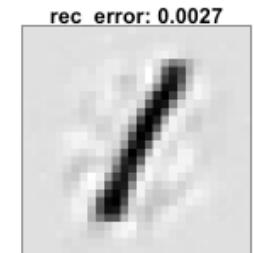
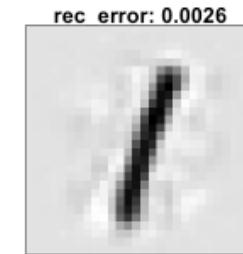
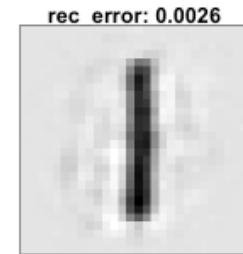
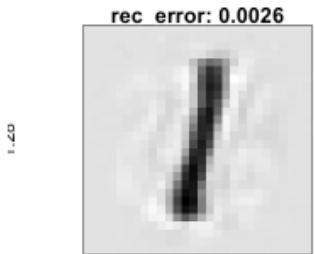


또한 심층 학습 자동 코드를 사용하여 데이터 세트의 특이점을 식별 할 수 있습니다.  
이 `h2o.anomaly()`함수는 테스트 데이터 세트에 대한 행당 재구성 오류를 계산합니다  
(자동 행 코드 모델을 통과하고 각 행에 대해 평균 제곱 오류 (MSE)를 계산합니다).



자기 자신을 적은수의  
Feature로 생성했을때  
Reconstruction Error를  
평가함

**6 digits with lowest reconstruction error**

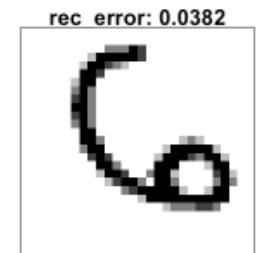
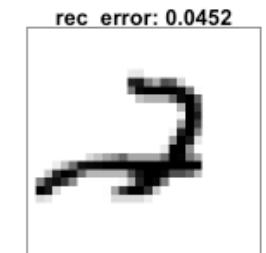
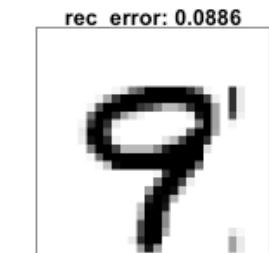
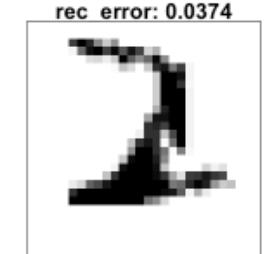
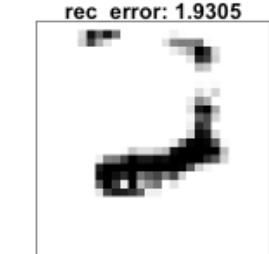


0.7.1

0.7.1

0.7.1

**6 digits with the highest reconstruction error**



0.7.1

0.7.1

0.7.1

0.7.1

0.7.1

0.7.1

0.7.1

0.7.1

0.7.1

# Contents

---

## 1st session

- Preprocessing - dplyr, data.table

## 2<sup>nd</sup> session

- Tree model

## 3<sup>rd</sup> session

- h2o tutorial

## 4<sup>th</sup> session

- XAI : Local Interpretable Model-agnostic Explanations**

# XAI



- <https://youtu.be/hUnRCxnydCc>
- <http://www.aitimes.kr/news/articleView.html?idxno=14859>

# 왜 모델 설명이 중요한가?

- ML을 실무에 적용하다 보면 만나게 되는 도전과제 → 모형이 내놓는 결과에 대한 해석!
- 성능이 좋아도 실제 액션을 취하려면 왜 이러한 예측과정에 대한 이해가 필수적!
- 모델을 만드는 분석가나 배포나 운영하는 엔지니어의 입장에서도 모형이 왜 이렇게 작동하는지 이해가 반드시 필요! → 결함을 사전에 인지하고 대응하기 가능!

# 논문 리뷰 - "Why Should I Trust You?" Explaining the Predictions of Any Classifier

- 2016 논문, 2133회 인용(19.12.07 시점)
  - Attention is all you need, 2017, 4866회 인용
  - BERT 2018, 2905회 인용
- 논문 출처



<https://www.kdd.org/kdd2016/papers/files/rfp0573-ribeiroA.pdf>

## ABSTRACT

- 다양한 분야에서 딥러닝과 같은 머신 러닝 기법들이 쓰임에도 불구하고, 여전히 많은 머신 러닝 기법들은 'black box'로 불림.
- 모델이 복잡해질수록 모델의 추론 과정을 인간이 이해하기는 난이도가 점점 상승. 하지만 모델을 이해하는 것은 굉장히 중요.(1, 2차함수 vs 다항함수)
- 모델을 이해할 수 있어야지만 그 모델을 trust 할 수 있기 때문
- 이 논문에서는 모델의 추론 과정을 이해하기 위해 LIME(Local Interpretable Model-agnostic Explanations) 알고리즘을 제안

## locally interpretable(LI)

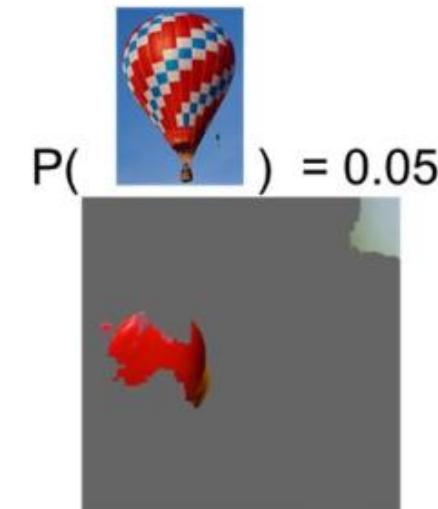
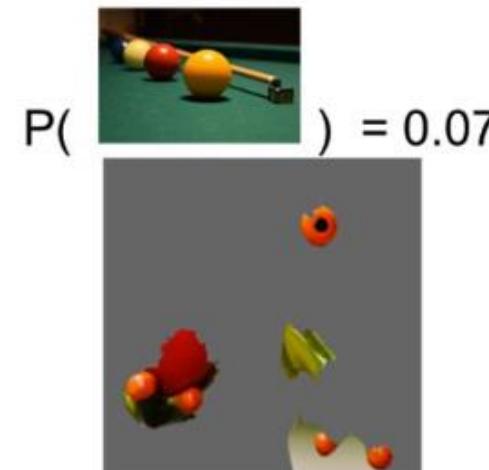
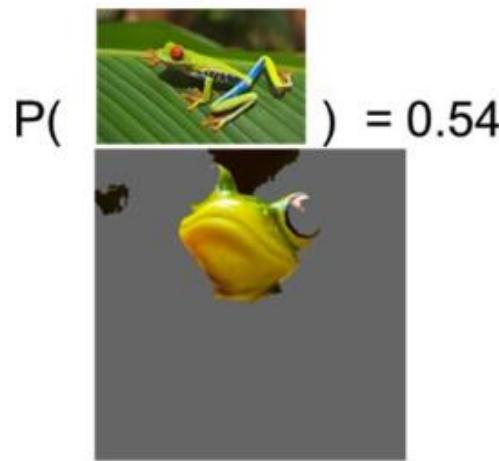
- 모형 자체가 어떠한 방식으로 작동하는지 이해하는 대신 Prediction 값 근방에 모형이 어떻게 작동하는지 설명
- 가정 : 작은 영역에서는 해석이 가능한 단순한 형태를 보일 것!!

## model-agnostic(ME)

- 어떠한 모델도 설명이 가능하다!!!
- 모형에 대한 가정을 하지 않기 때문에 어떠한 모델에도 적용이 가능한 접근법!

# Introduction

- 딥러닝 모델의 경우 사람이 레이어, 노드, 파라미터를 보고 모델의 추론 과정을 해석하기란 매우 어려움
- 하지만 모델의 복잡한 내부가 아닌 주어진 데이터의 "어떤 부분(local한 특성)"을 보고 개구리, 당구공, 열기구라고 예측했는지라도 알 수 있다면, 사용자는 모델을 trust 가능



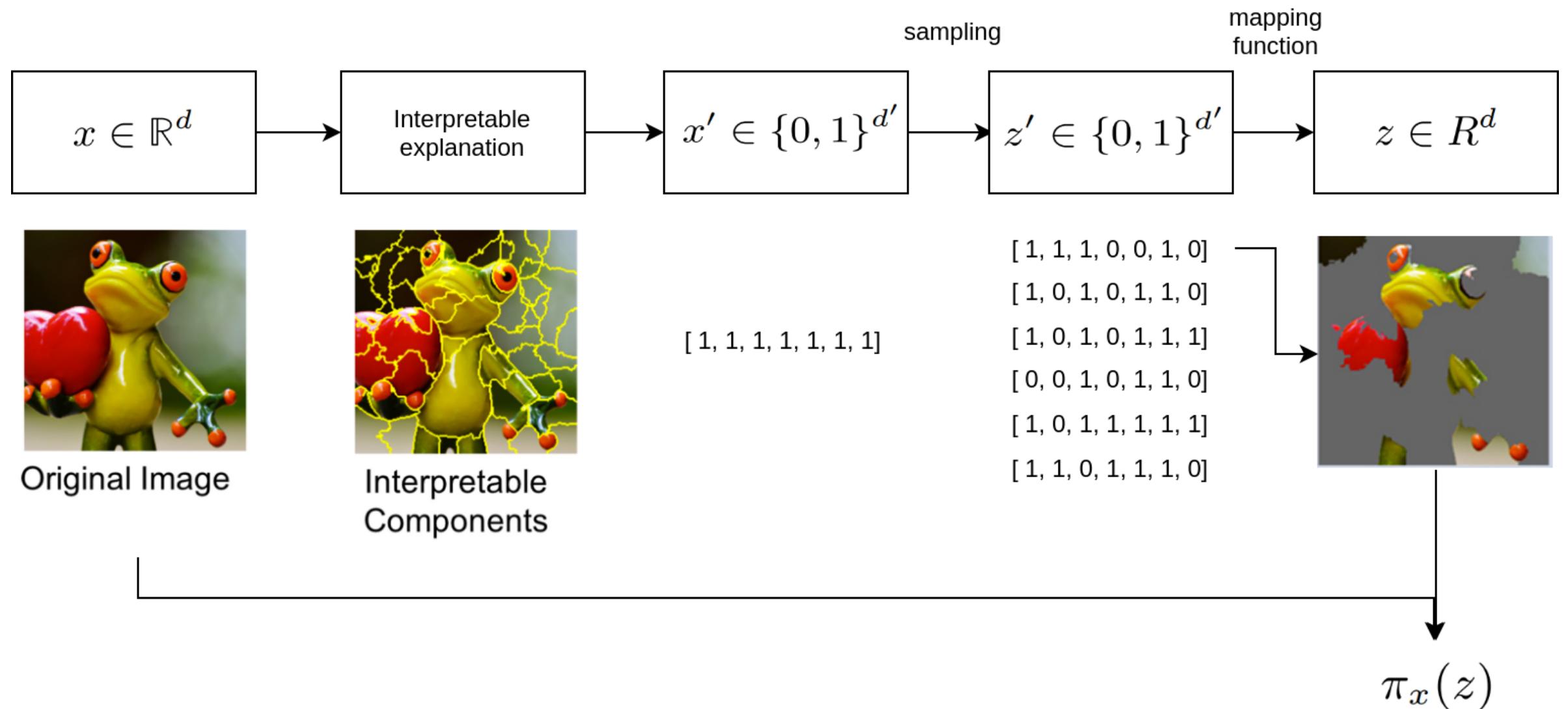
## LIME의 General Framework

- 논문에서는 예측에 중요한 부분을 찾기 위해 복잡한 모델(원래 모델)을 해석 가능한 '*interpretable representation*' + '*interpretable model*'로 표현하여 해석
  - interpretable representation : 실제 사용하는 데이터를 사람이 해석할 수 있는 데이터로 재구성한(자기 자신을 잘 표현하는) 벡터
  - **interpretable model** : linear models, decision trees 등과 같이 사람이 직관적으로 이해할 수 있는 모델
- 아무리 복잡한 모델이라도 이를 사람이 **해석 가능한 데이터와 모델로 근사** 할 수 있다면 → 데이터의 어떤 부분이 중요한 역할을 했는지 알아낼 수 있음

## interpretable representation - 해석 가능한 특성(e.g. vision)

- interpretable representation은 사람이 이해할 수 없는 original 데이터를 사람이 이해 할 수 있는 데이터의 존재 유/무로 표현한 binary vector(vision에서는 픽셀값, nlp에서는 단어)
  - 예를 들어 이미지를 분류하는 경우 original 데이터가 픽셀값
  - 각 픽셀값은 사람이 해석하기 어렵기 때문에 이를 사람이 해석할 수 있는 super-pixel(관련 링크)의 존재 유/무만 표현

1. 아래 그림에서 개구리 사진 픽셀(original 데이터)을 super-pixel 단위로 표현(사람이 인지 할 수 있도록 함)
2. 이후 super-pixel 단위의 존재 유무를 **binary vector  $x'$** (interpretable representation) 으로 표현(아래 경우 모든 super-pixel이 1로 표현,  $x'$ 의 정확한 의미는 하단의 암 환자 예시로 이해하는 것이 더 명확)
3. (여기서부터 LIME idea) 표현한  $x'$ 의 non-zero 부분에 대해서 부분집합을 하이퍼 파라미터 N개 만큼 sampling 하여  $z'$ 을 생성
4. 다음으로  $z'$ 을 사용자가 지정한 mapping 함수(e.g. gray)를 통해  $z$ 로 재변환
5. 마지막으로 원본 데이터  $x$  와 만들어진  $z$  간의 유사도를 측정 →  $\pi_x(z)$  에 저장 이후에 원본 데이터 와 가까운 sample에 가중치를 부여하기 위함(유사도는 이미지의 경우 L2 distance)



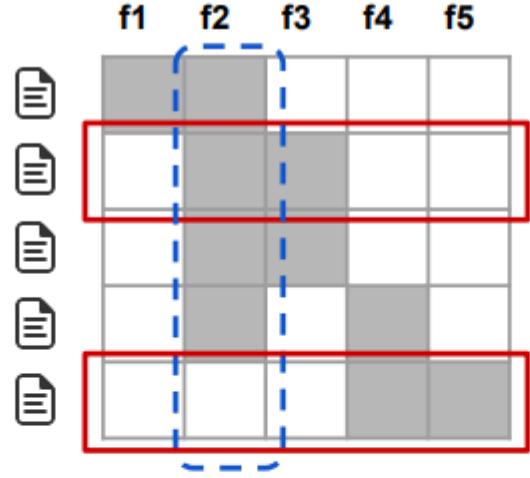
- Shape(dimension)

- **x\_d 차원** : original 데이터 (e.g. pixel)
- **interpretable components\_d' 차원** : 사람이 해석 가능한 데이터 단위
- **x'\_d' 차원** : interpretable representation, x를 interpretable components 존재 유무로 표현한 binary 벡터 (e.g. 1의 의미는 개구리 이미지 super-pixel의 존재를 뜻함)
- **z'\_d' 차원** :  $x'$  중 non-zero의 부분집합 sampling, 사용자가 sampling 개수를 정의
- **z\_d 차원** :  $z'$ 을 mapping 함수를 정의하여(e.g. zero super pixel을 gray로 채움)을 통해 다시 original 데이터의 형태로 변환한 데이터(origianl 데이터와 같은 dim)
- **$\pi_x(z)$ (스칼라)** : x와 z 유사도 (원본 데이터와 가장 가까운 sampling 데이터에 가중치를 주기 위함)

## interpretable representation - 해석 가능한 특성(e.g. Text)

interpretable components의 의미를 조금 더 이해해 보자면, interpretable components 주어진 데이터에 한정되지 않습니다. (즉,  $d > d'$  or  $d < d'$ )

- **original 데이터** : 주어진 문서 안에 있는 단어의 embedding vector(d는 주어진 문서 안에 있는 단어 수,  $d'$ 는 단어 사전의 수)
- **반면 interpretable components** : 주어진 문서의 단어 뿐 아니라 다른 단어들을 포함한 단어 사전
- **Task** : 암 환자의 진단서 분류를 예시



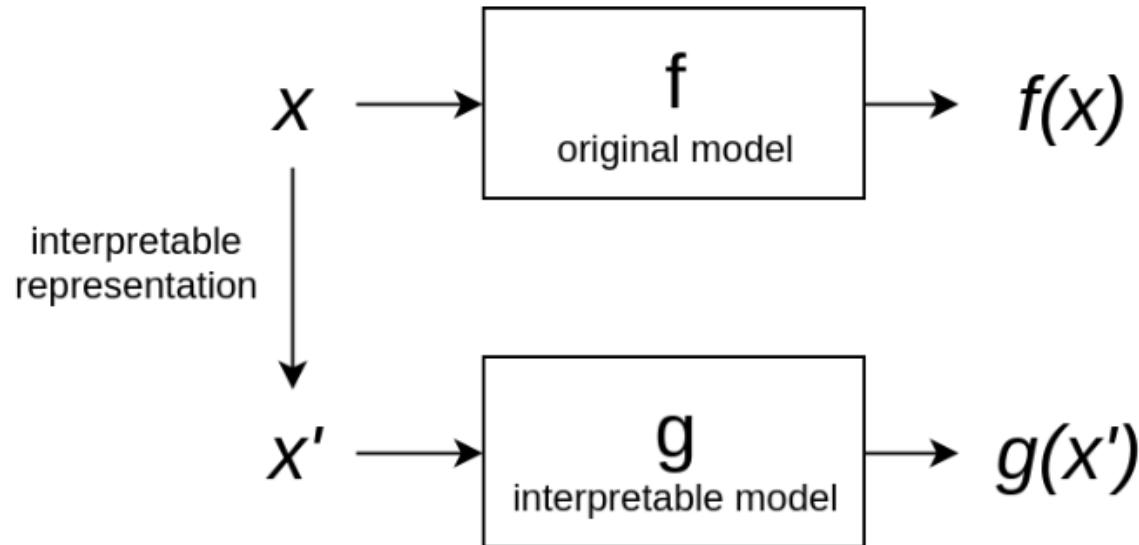
**Figure 5:** Toy example  $\mathcal{W}$ . Rows represent instances (documents) and columns represent features (words). Feature  $f_2$  (dotted blue) has the highest importance. Rows 2 and 5 (in red) would be selected by the pick procedure, covering all but feature  $f_1$ .

- 100개의 진단서가 있는 경우 각 진단서에는 서로 다른 여러개의 증상들이 나열되어 있음
- 이 때 첫번째 진단서를 doc1라고 한다면, doc1의 original 데이터  $x$ 는 doc1에 적힌 증상의 embedding vector이고 d차원은 doc1에 적힌 증상의 개수
- 반면 interpretable components은 100개의 진단서에 있는 모든 증상의 binary 벡터이고, d'차원은 100개 진단서에 있는 모든 증상의 개수
- d차원과 d'차원의 차이가 하단에 서술할 LIME에 쓰여지는 local fidelity의 핵심!

개구리 예시와 같은 이미지의 경우 super-pixel은 각 이미지마다 다르게 형성 됩니다. 따라서 텍스트 분류에서 단어 사전과 같이 interpretable representation을 명확히 정의 할 수 없습니다. 논문에서도 color histograms or other features of super-pixels 등과 같은 방법으로 정의를 해야하고, 이는 차후 연구 주제 넘긴다고 명시해놨습니다.

# interpretable model - 해석 가능한 모델

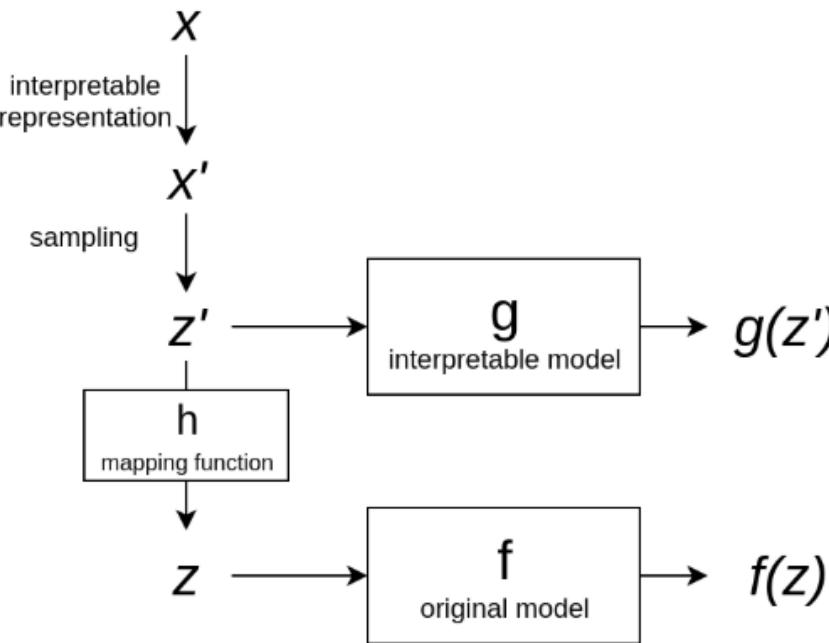
- 해석 가능한 모델(간단한 모델)의 대표적인 예시로, linear models, decision trees 이 존재.
  - linear models의 경우 각 변수들의 계수로 직관적 해석이 가능
  - decision trees의 경우 분리 기준으로 직관적 해석이 가능
  - 결국 복잡한 모델을 간단한 모델로 근사 시킬 수 있는 interpretable model을 찾는 것과 모델을 해석 할 수 있다는 것은 동일한 의미를 뜻합니다.
- interpretable model은 사람이 해석 가능한 간단한 모델
- 위에 서술한 interpretable representation을 사용하여 변수를 먼저 해석 가능하게 만듬
- 이후 original  $f(x)$ 값을 해석 가능한  $g(x')$ 으로 근사시키는 것이 목적
- 이 때  $f(x)$ ,  $g(x')$ 은 분류의 경우 확률값임



- $x$  : original 데이터
- $x'$  : interpretable representation of  $x$  ( $x$ 를 interpretable components로 매핑한 binary 벡터)
- $f$  : orginal model (복잡한 모델)
- $g$  : interpretable model (사람이 해석 가능한 간단한 모델)
- $f(x)$  : original 예측치
- $g(x')$  : 해석 가능한 변수와 해석 가능한 모델을 쓴 예측치

결국 복잡한 함수  $f$ 에 근사 할 수 있는 해석 가능한 함수  $g$ 을 찾는게 interpretable model의 목표!!

## LIME Algorithm



1. 주어진  $x$ 를 해석 가능한  $x'$  으로 변환(e.g. 개구리 이미지를 super-pixel 단위의 binary vector로 변환)
2.  $x'$ 의 non-zero fraction  $z'$ 을 N개 추출.(N은 하이퍼 파라미터)
3.  $z'$ 에서  $h(z')$  매팅 함수를 통해  $z$ 를 만듬
4. 원래 함수  $f$ 를 통해  $z$ 의 예측치  $f(z)$  (e.g. sampling 된  $z'$  을 변환한 이미지의 개구리 확률) 계산
5.  $f(z)$ 에 근사 할 수 있는 선형  $g(z')$ 을 탐색 → 선형  $g(z')$ 의 가중치를 통해 어떤 변수가 중요한 역할을 할 수 있는지 해석 (선형이 아닌 해석 가능한 다른 함수도 가능하지만 논문에선 선형을 사용)

$$g(z') = \phi_0 + \sum_{i=1}^M \phi_i z'_i$$

## Sparse Linear Explanations

- $g(z')$ 을  $f(z)$ 에 근사시키는 과정
- 먼저 Loss function 을 풀어 설명하면  $Z$  집합에 속하는  $z$ 와  $z'$  이 주어짐

$$\mathcal{L}(f, g, \pi_x) = \sum_{z, z' \in \mathcal{Z}} \pi_x(z) (f(z) - g(z'))^2$$

- $Z$ 는 사용자가 지정한 sampling 개수( $N$ ) 개 집합
- $f(z)$ 와  $g(z')$ 의 차에  $x$ 와  $z$  유사도  $\pi_x(z)$ 를 곱해주어 원본과 가까운 sample에 가중치를 부여

다음은 근사를 일반화 한 수식

$$\xi(x) = \operatorname{argmin}_{g \in G} \mathcal{L}(f, g, \pi_x) + \Omega(g)$$

- L : x에 대한 설명 모델의 손실(예를 들면, MSE)
- f : 설명하고자 하는 원래 모델
- g : Loss를 최소화 하는 모델(예를 들면, 선형회귀모델)
- G : 설명이 가능한 모든 모델, g의 상위 집합(예를 들면, 설명이 가능한 모든 선형 회귀 모델)
- Omega : 모델 복잡도, 낮게 유지하려고 함
- G 집합은 해석 가능한 모델 (linear, decision tree 등) 의 집합
- loss를 최소화 하는 동시에 g의 복잡도를 낮추는 페널티를 부여
- 논문에서는 Lasso-regression 을 이용

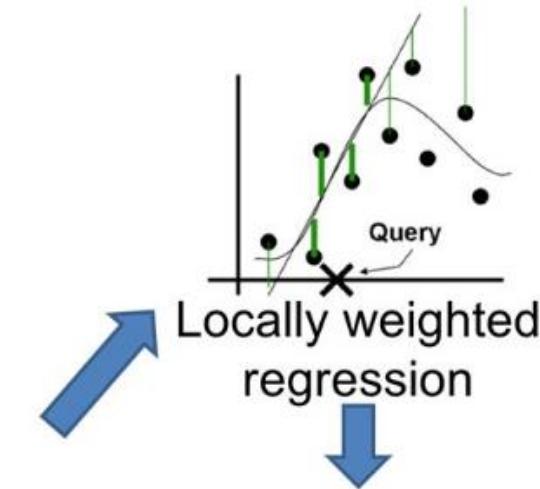
# 정리



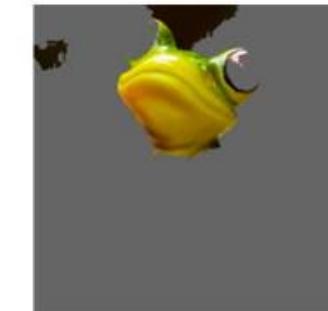
Original Image  
 $P(\text{tree frog}) = 0.54$



Perturbed Instances	$P(\text{tree frog})$
	0.85
	0.00001
	0.52



Locally weighted regression

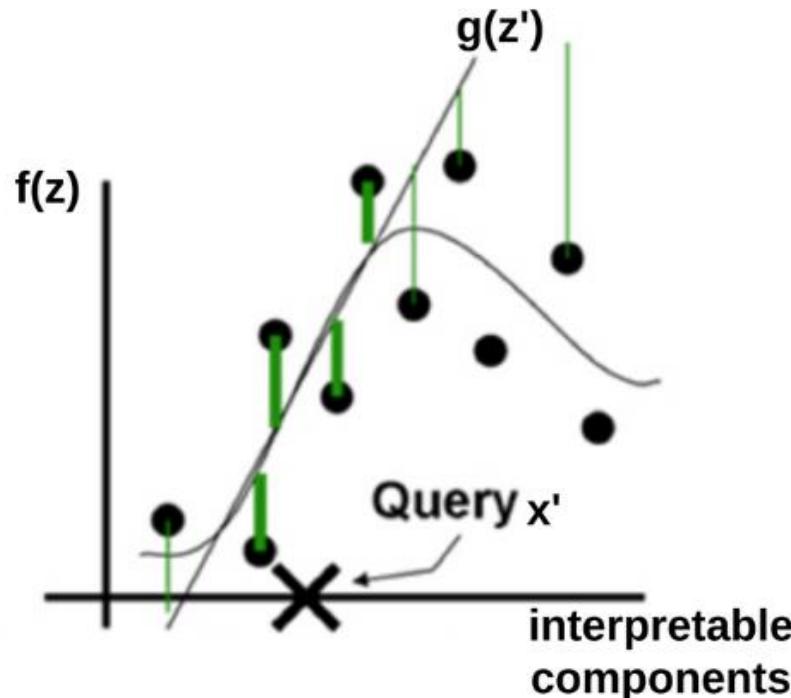


Explanation

|출처: Marco Túlio Ribeiro, Pixabay.

- 개구리 이미지( $x$ )가 들어오면 이는 interpretable representation( $x'$ )으로 변환
- 이는  $N$ 개의 non-zero fraction sampling  $z'|0$ 이 되고 mapping 함수를 통해 부분 이미지( $z$ )가 됨
- $x', z'$ 은 그림에서 생략되고 Perturbed Instance가  $z \rightarrow$  이를 통해  $f(z)$ 를 구함

- 이제 구해진  $f(z)$ 를 통해  $g(z')$  을 찾고 싶어 → 이때 Locally weighted regression이 사용
- 논문에서는  $x$ 와  $z$ 의 유사도 weight로 사용하고 Lasso를 통해  $g$ 를 찾음



(그림의 초록색 부분이 weighted loss  $\pi x(z)^*(f(z) - g(z'))^2$  를 나타내고 있음)

$$g(z') = \phi_0 + \sum_{i=1}^M \phi_i z'_i$$

최종적으로  $g(z')$ 을 찾을 수 있고 높은 가중치를 가진 부분을 찾아보니 개구리 얼굴이라는 것을 알아 낼 수 있음 → 이를 통해 예측에 중요한 역할을 데이터를 찾고 모델을 해석 할 수 있습니다.

## 여러분들한테 당부의 말씀

- LIME 말고 SHAP(SHapley Additive exPlanations)라는 방법도 있는데 많이 쓴다고 합니다. 한번 공부해보세요~
- <https://pair-code.github.io/what-if-tool/> 여기도 한번 가보세요 (<https://youtu.be/ZAS2FwlLTNE> 송호연님 유튜브)
- 여기는 정말 흥미로운 소재 + 트렌디한 소재이니 공부해보세요~!

# Reference

- Main Reference

LIME - why should i trust you

다양한 분야에서 딥러닝과 같은 머신 러닝 기법들이 쓰임에도 불구하고, 여전히 많은 머신 러닝 기법들은 'black box'로 불립니다. 모델이 복잡해질수록 모델의 추  
<https://www.notion.so/LIME-why-should-i-trust-you-caf64b5b4d5f41ba824c69...>

- A Unified Approach to Interpreting Model Predictions

<https://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf>

- 파이썬 구현 및 한글 설명

LIME (Local Interpretable Model-agnostic Explanation)

ML을 실무에 적용하다 보면 만나게 되는 도전과제 중 하나는 모형이 내놓는 결과에 대한 해석이다. 아무리 예측력이 좋은 ML 모형이라고 하더라도 실제 액션을 취  
☞ <https://nhlmary3.tistory.com/entry/LIME-Locally-Interpretable-Modelagno...>



머신러닝 모델의 블랙박스 속을 들여다보기 : LIME

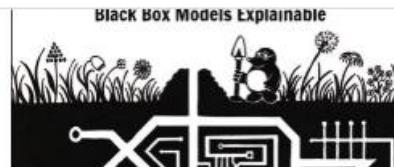
머신 러닝 모델에 대해서 예측의 이유를 설명하는 것은 어렵습니다. 모델이 복잡해질수록 예측의 정확도는 높아지지만, 결과의 해석은 어려워지죠. 그렇기 때문에  
☞ <https://dreamgonfly.github.io/2017/11/05/LIME.html>



- 교재가 따로 있을 정도([여기](#)를 눌러서 반드시 읽어보시는 것을 권장)

Interpretable Machine Learning

Machine learning has great potential for improving products, processes and research. But computers usually do not explain their predictions which is a  
<https://christophm.github.io/interpretable-ml-book/>



- 구현 공식 github

marcotcr/lime

This project is about explaining what machine learning classifiers (or models) are doing. At the moment, we support explaining individual predictions for text  
☞ <https://github.com/marcotcr/lime/tree/ce2db6f20f47c3330beb107bb17fd2...>



마치면서~

[https://github.com/hotorch/pnustat\\_lec\\_201912](https://github.com/hotorch/pnustat_lec_201912)

hotorch / pnustat\_lec\_201912

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Actions Projects 0 Wiki Security Insights Settings

2019/12/26 ~ 27일, 2 ~ 5pm R, ML, h2o 특강

Edit

Manage topics

↳ XAI(eXplainable AI)

- BOOK Local Interpretable Model-agnostic Explanations
- CODE Local Interpretable Model-agnostic Explanations Code 진행중

Feedback plz!!

피드백 부탁드립니다!