

# Big data 분석이란?

---

컴퓨터공학부  
천세진

# 빅데이터란?

---

- 빅데이터 = 데이터마이닝, 예측 분석, 데이터 과학



# 빅데이터란?

데이터의 모델과 요약의 **일반화**  
(Generalizations)

메인 메모리를 위해 매우  
큰 데이터를 분석하는 방법

많은 수의 관찰(observations)와  
특징(features)과 함께 가능성을  
분석

## 데이터의 모델과 요약의 일반화 (Generalizations)

Data frameworks

Algorithms and Analyses

## 데이터의 모델과 요약의 일반화 (Generalizations)

Data frameworks

Hadoop File System  
Spark  
Streaming  
MapReduce  
Tensorflow

Algorithms and Analyses

## 데이터의 모델과 요약의 일반화 (Generalizations)

Data frameworks

Hadoop File System  
Spark  
Streaming  
MapReduce  
Tensorflow

Algorithms and Analyses

Similarity Search  
Linear Modeling  
Recommendation Systems  
Graph Analysis  
Deep Learning

## 데이터의 모델과 요약의 일반화 (Generalizations)

### ■ Descriptive analytics

- 데이터 자체에 대해서 (일반화하여) 표현하는 방법

### ■ Predictive analytics

- 새로운 데이터에 대한 일반화시키는 무엇인가를 생성하는 것

## 데이터의 모델과 요약의 일반화 (Generalizations)

- **Google's PageRank:** 하나의 수를 사용하여 web pages를 요약
- **Twitter financial market predictions:** 트위터 내 감정에 대한 동향에 따라 주식시장을 모델링
- **의학 이미지 내 조직 종류 식별:** 수백만 픽셀을 몇 개의 클러스터(Clusters)로 요약
- **소셜미디어 내 건강을 진단:** 언어학적 패턴의 요약과 함께 진단 상태를 모델링
- **같이 구매하는 좋은 상품 추천:** 수백만 건의 구매 기록을 함께, 빈도가 높은 함께 구매되는 상품을 파악



# 그럼 여러분은? 다음 이슈 해결을 위해 어떤 데이터들이 필요할까요?



치안 체감안전도 예측



도시센서(cctv 등) 최적위치 선정



재활용품 수거기 최적위치 선정



공공자전거 스테이션 위치선정



주택시장 특성 분석



전기차 충전소 최적의 입지 선정

# Big data analytics - 본 수업에서 배우는 내용

- To analyze different types of data
  - High dimensional
  - Graphs
  - Infinite/never-ending
  - Labeled
- To use different models of computation:
  - MapReduce
  - Streams and online algorithms
  - Single machine in-memory
  - Spark, Tensorflow(조금)



# High dimensional data

6 features

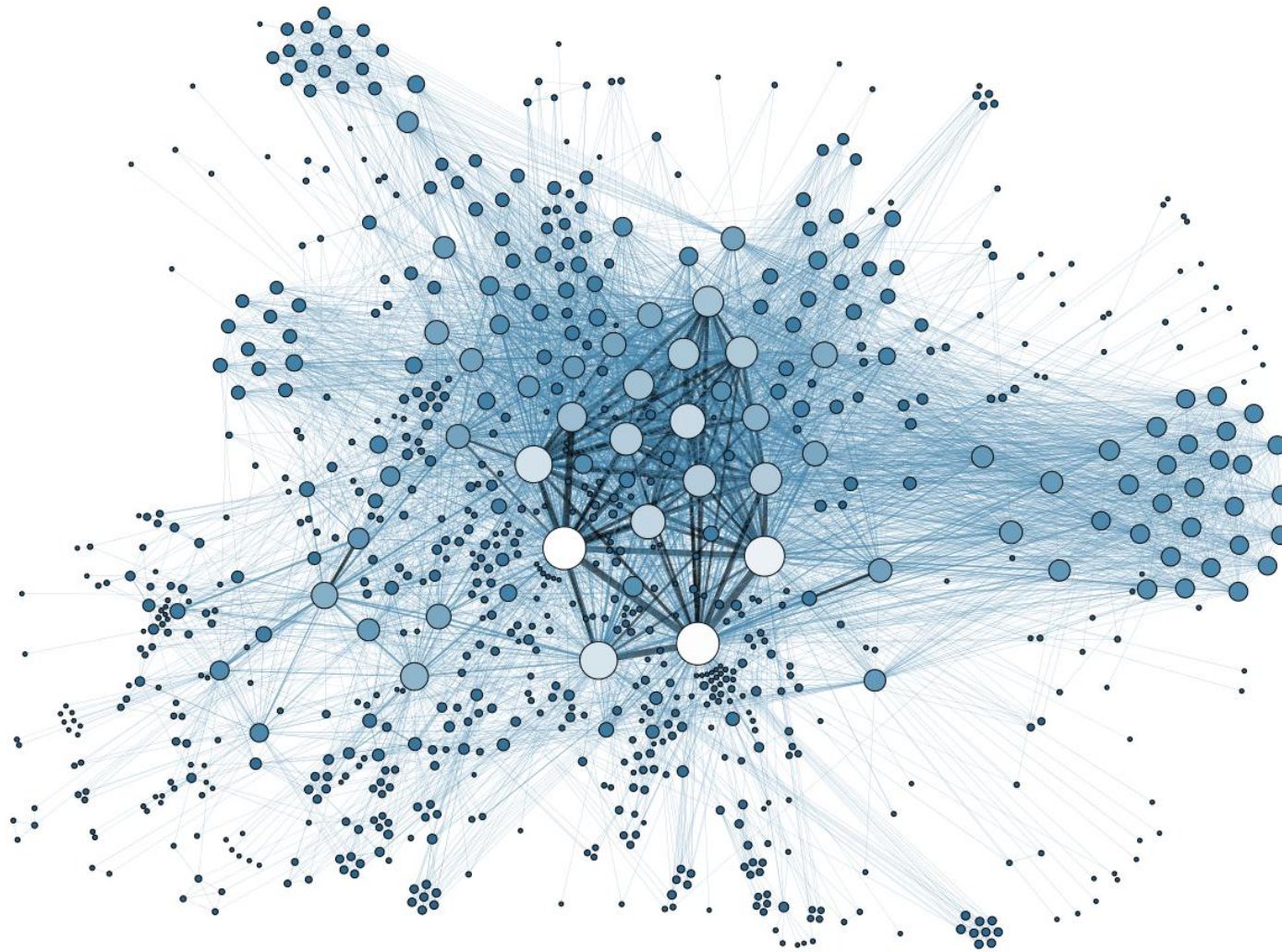
3 observations

	$p_1$	$p_2$	$p_3$	$p_4$	$p_5$	$p_6$
$n_1$						
$n_2$						
$n_3$						

<https://www.statology.org>



# Graphs





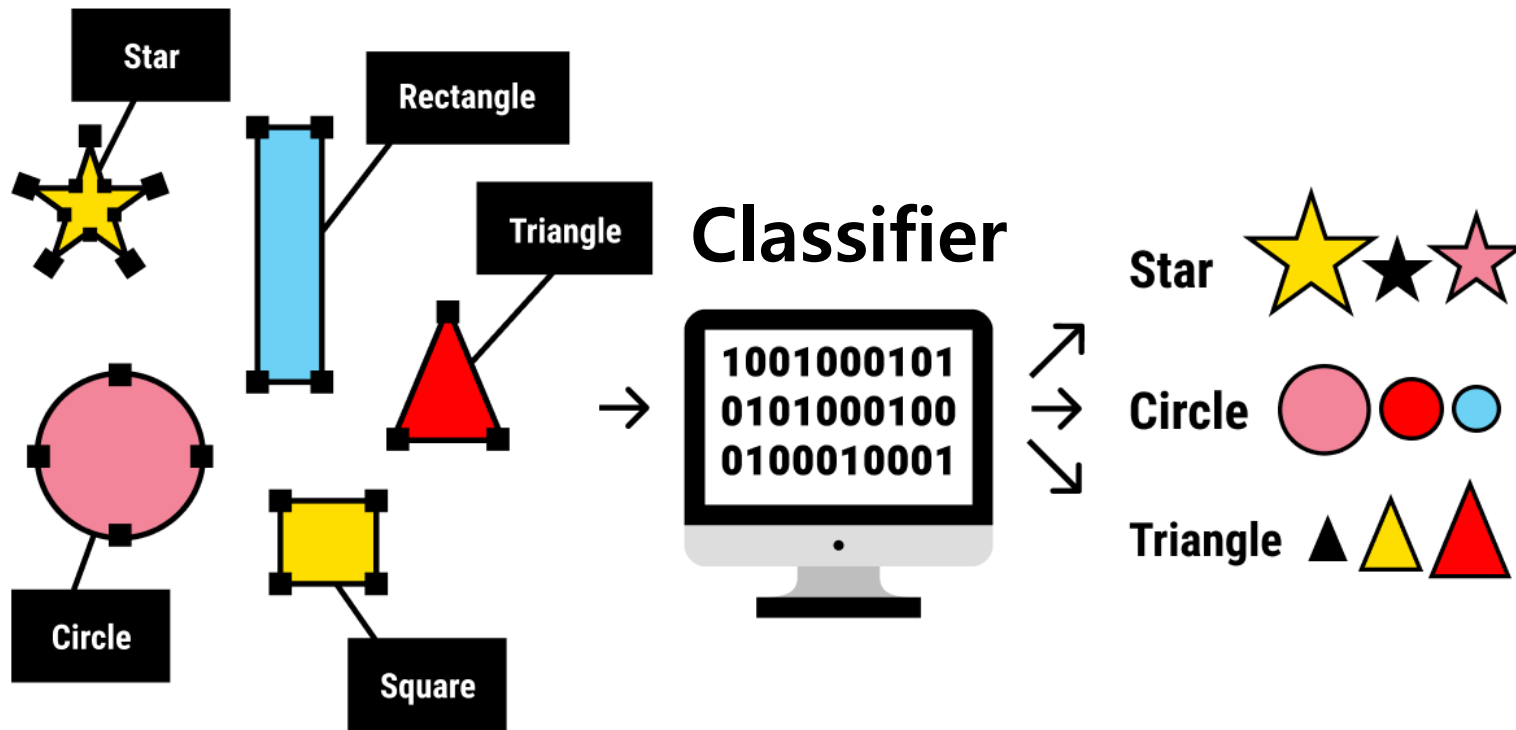
# Infinite data



<https://www.onaudience.com>



# Labeled Data

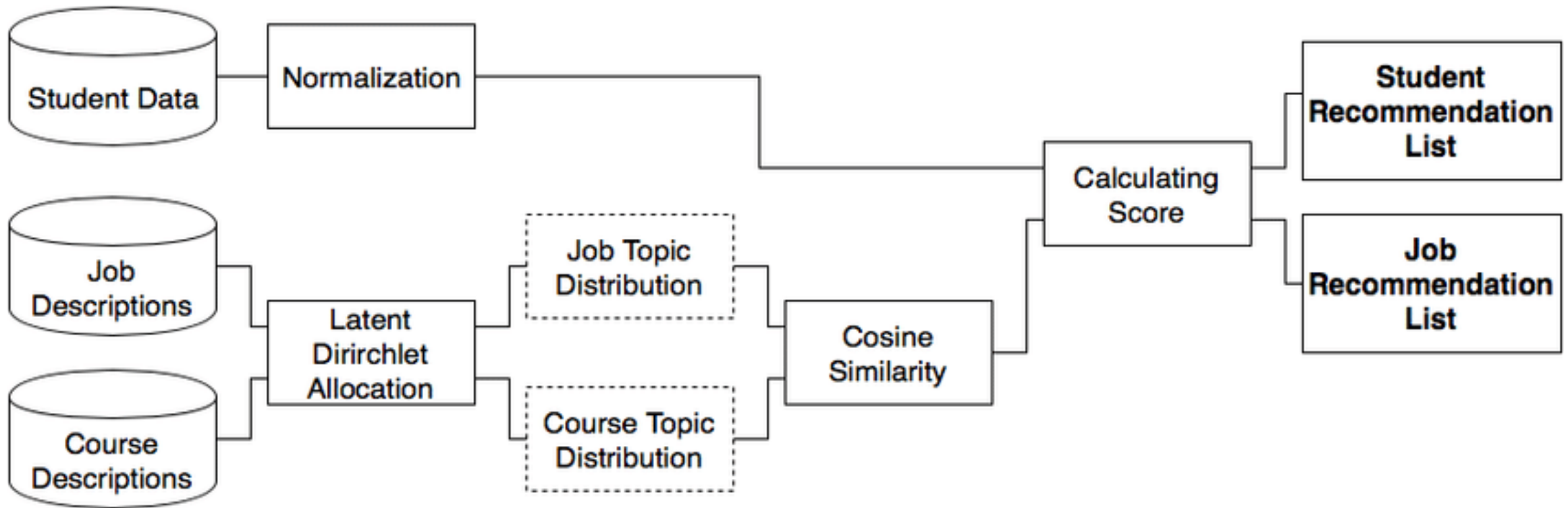


# Big data analytics - 본 수업에서 배우는 내용

- To solve real-world problems
  - Recommendation systems
  - Market-basket analysis
  - Spam and duplicate document detection
  - Geo-coding data
- Uses of various 'tools'
  - Linear algebra
  - Optimization
  - Dynamic programming
  - Hashing
  - Functional programming
  - tensorflow



# Pipeline of Recommendation system

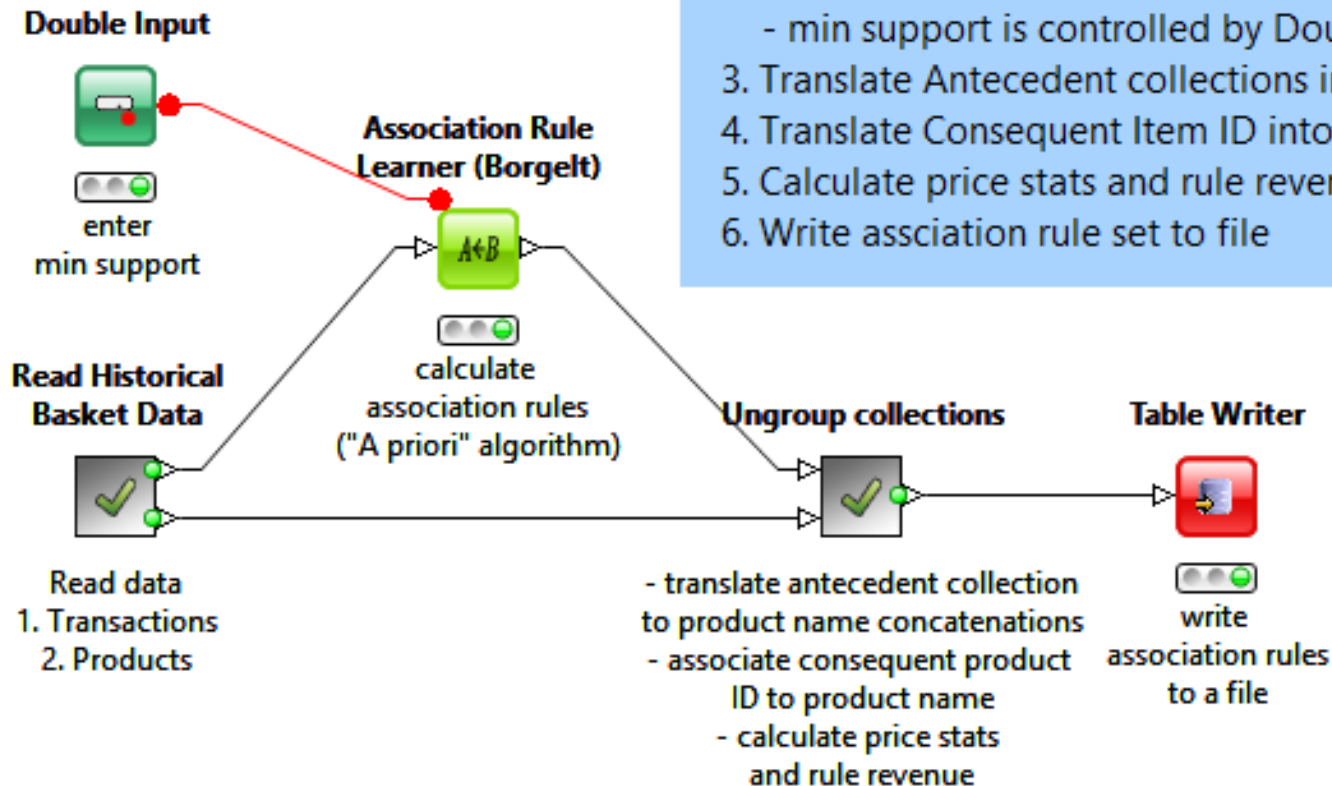




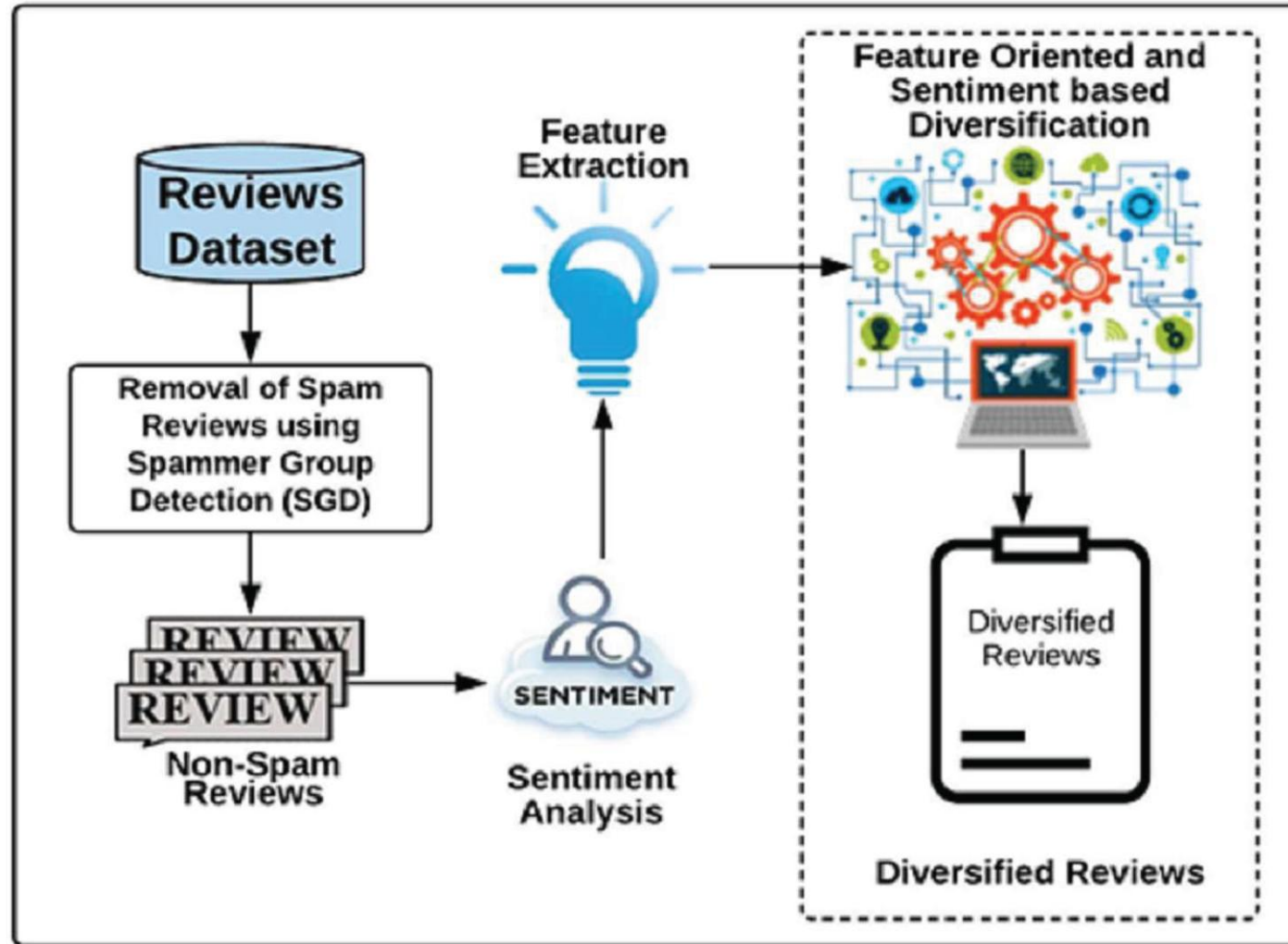
# Pipeline of Market-basket analysis

## 1. Market Basket Analysis: Build Association Rules

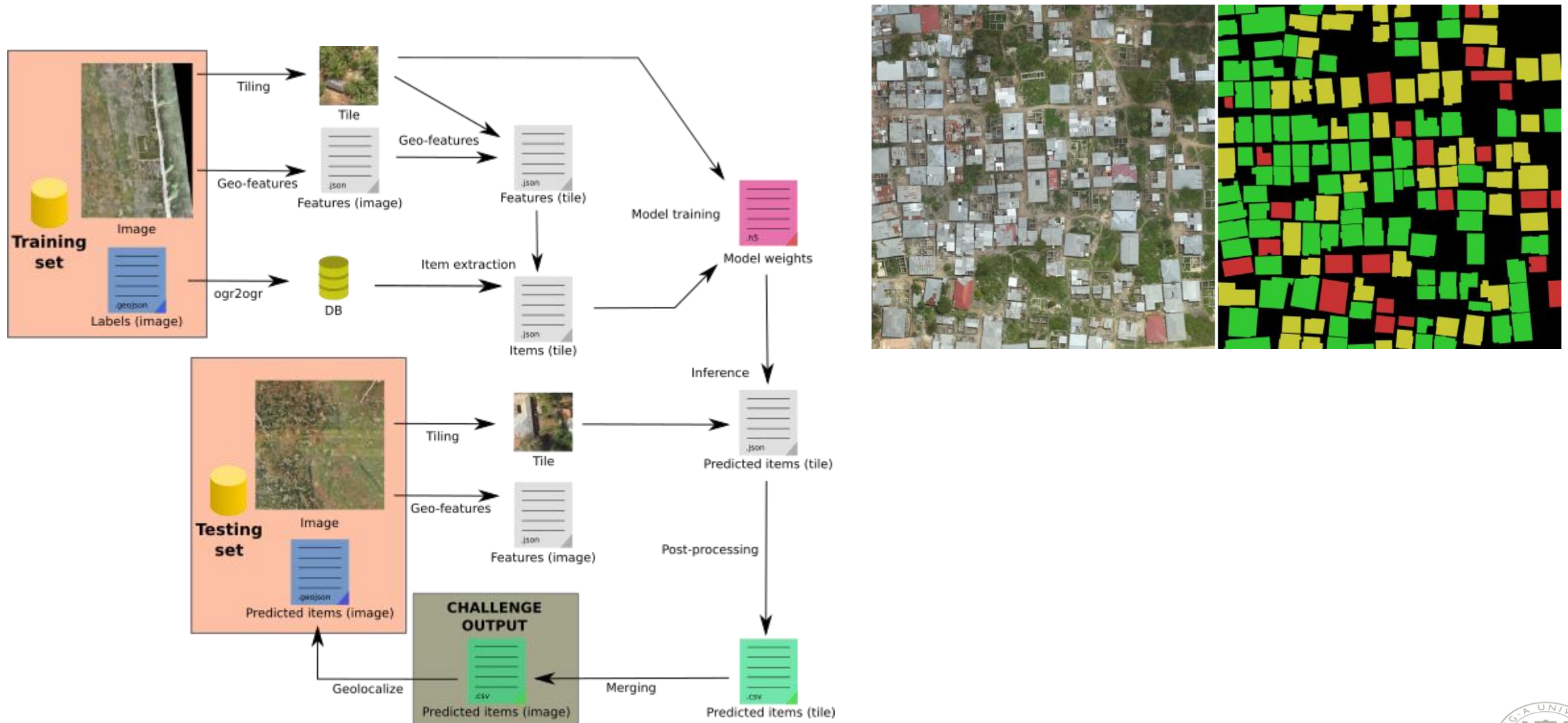
1. Read Transaction/Basket data and Product data
2. Using "A priori" algorithm, build association rule set
  - min. set size = 1
  - min rule confidence = 10%
  - min support is controlled by Double Input Quickform node in %
3. Translate Antecedent collections into product name concatenations
4. Translate Consequent Item ID into Consequent Product Name
5. Calculate price stats and rule revenue
6. Write association rule set to file



# Pipeline of Spammer group detection



# Pipeline of Geo-coding data



# Preliminaries(사전 지식)

---

# 아래 아이디어와 방법론은 강의 내내 자주 사용될 것입니다

---

- Bonferroni's Principle (본페로니 교정)
- Normalization (TF.IDF) (정규화)
- Power Laws (멱법칙)
- Hash functions
- IO Bounded (Secondary Storage)
- Unstructured data



# 통계적인 한계

## ■ 가설(Hypothesis): 지구는 둥글다

- 어떤 모집단의 모수(예: 평균, 분산 등)에 대한 잠정적인 주장

## ■ 귀무가설(Null)

- 20세 이상의 성인 남자의 평균 키는 170과 같다(또는 차이가 없다)
- ~차이가 없다, ~의 효과는 없다, ~와 같다
- 현상은 특별하지 않고, 우연이나 기존 지식으로 설명 가능하다

# 통계적인 한계

## ■ 가설(Hypothesis): 지구는 둥글다

- 어떤 모집단의 모수(예: 평균, 분산 등)에 대한 잠정적인 주장

## ■ 귀무가설(Null)

## ■ 대립가설(Alternative: $H_1$ ), 혹은 대안가설

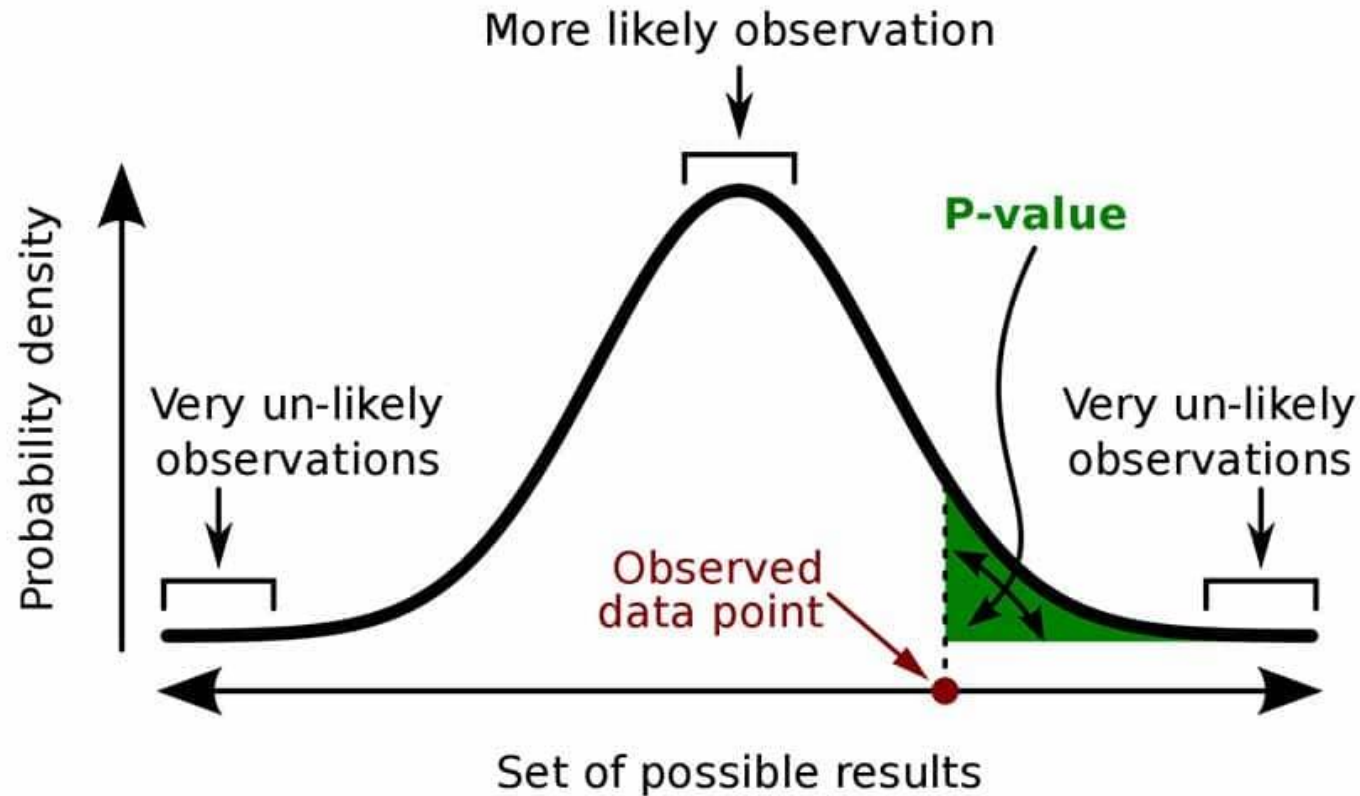
- 귀무가설이 거짓이라면 대안적으로 참이 되는 가설
- ~와 차이가 있다. ~의 효과는 있다. ~와 다르다
- 특별한 효과나 차이가 존재한다

# p-value(유의확률)

- p-값(유의확률, p-value)
  - 귀무 가설(null hypothesis)이 맞다는 전제 하
  - 표본에서 실제로 관측된 통계치와 '같거나 더 극단적인' 통계치가 관측될 확률이다
- p-값이 작을수록 그 정도가 약하다고 보며, 특정 값 (대개 0.05나 0.01 등) 보다 작을 경우 귀무가설을 기각하고, p-값이 클수록 귀무가설을 채택한다.



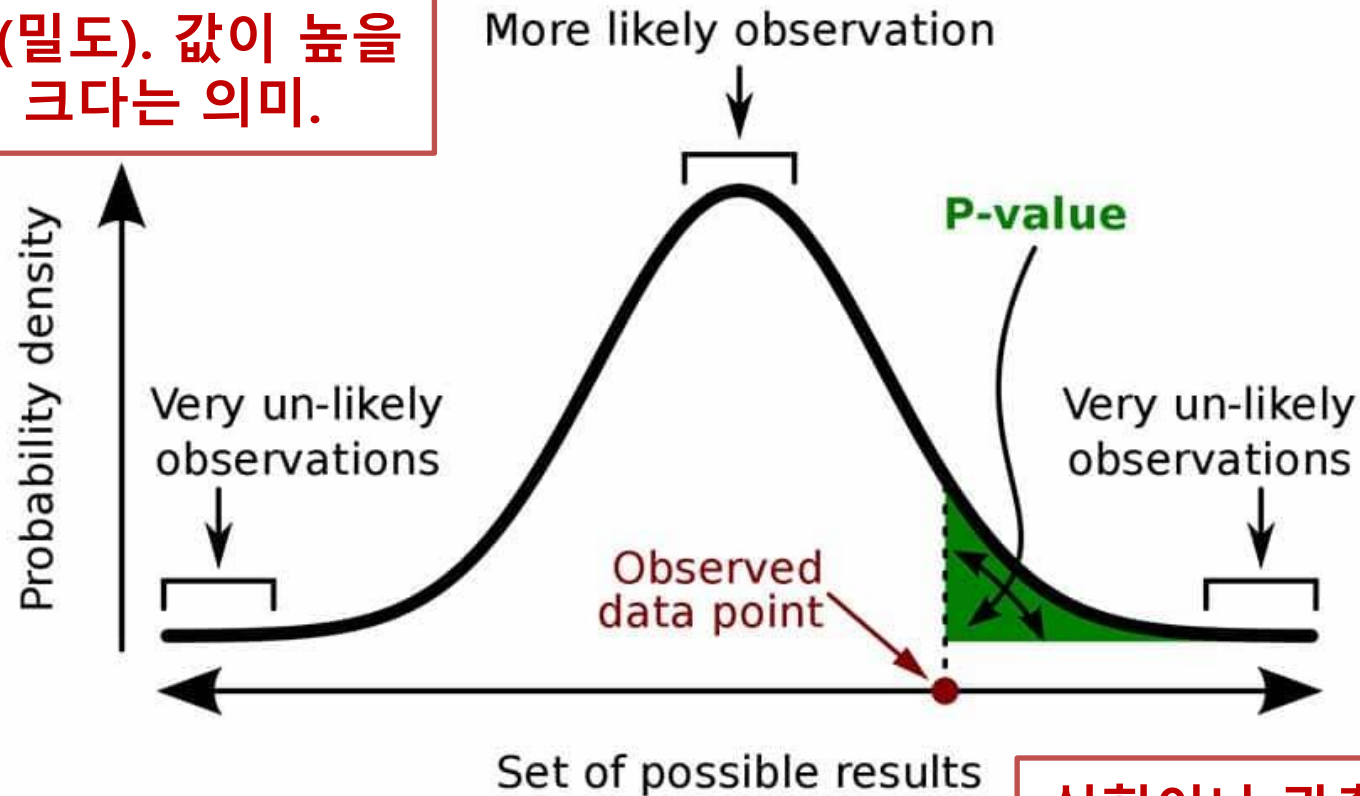
# p-value(유의확률)



A **p-value** (shaded green area) is the probability of an observed (or more extreme) result assuming that the null hypothesis is true.

# p-value(유의확률)

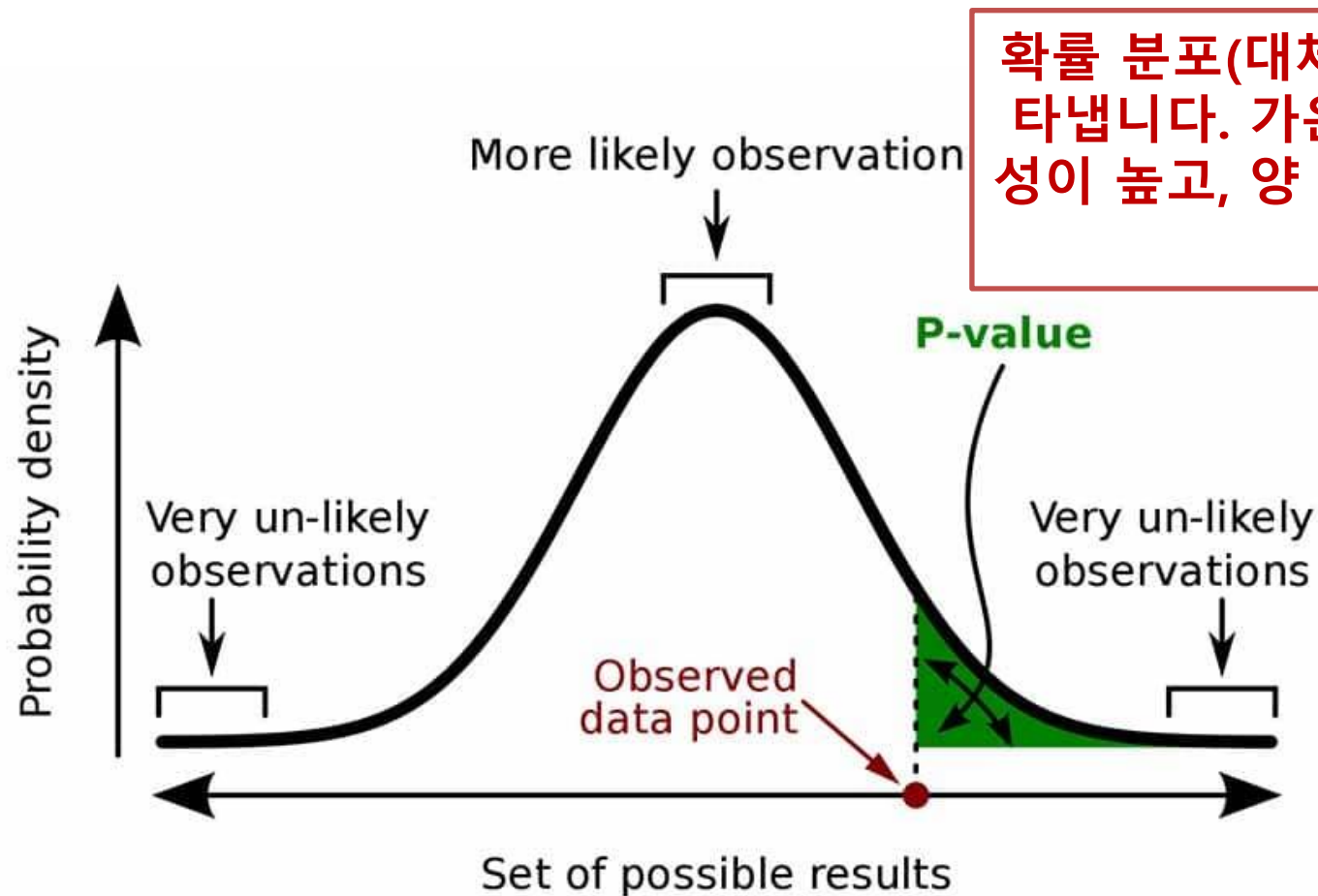
해당 값이 나올 확률(밀도). 값이 높을수록 더 가능성이 크다는 의미.



A **p-value** (shaded green area) is the probability of a result (or more extreme) result assuming that the null hypothesis is true.

실험이나 관측에서 나올 수 있는 모든 가능한 값(데이터 점, 관측 결과).

# p-value(유의확률)

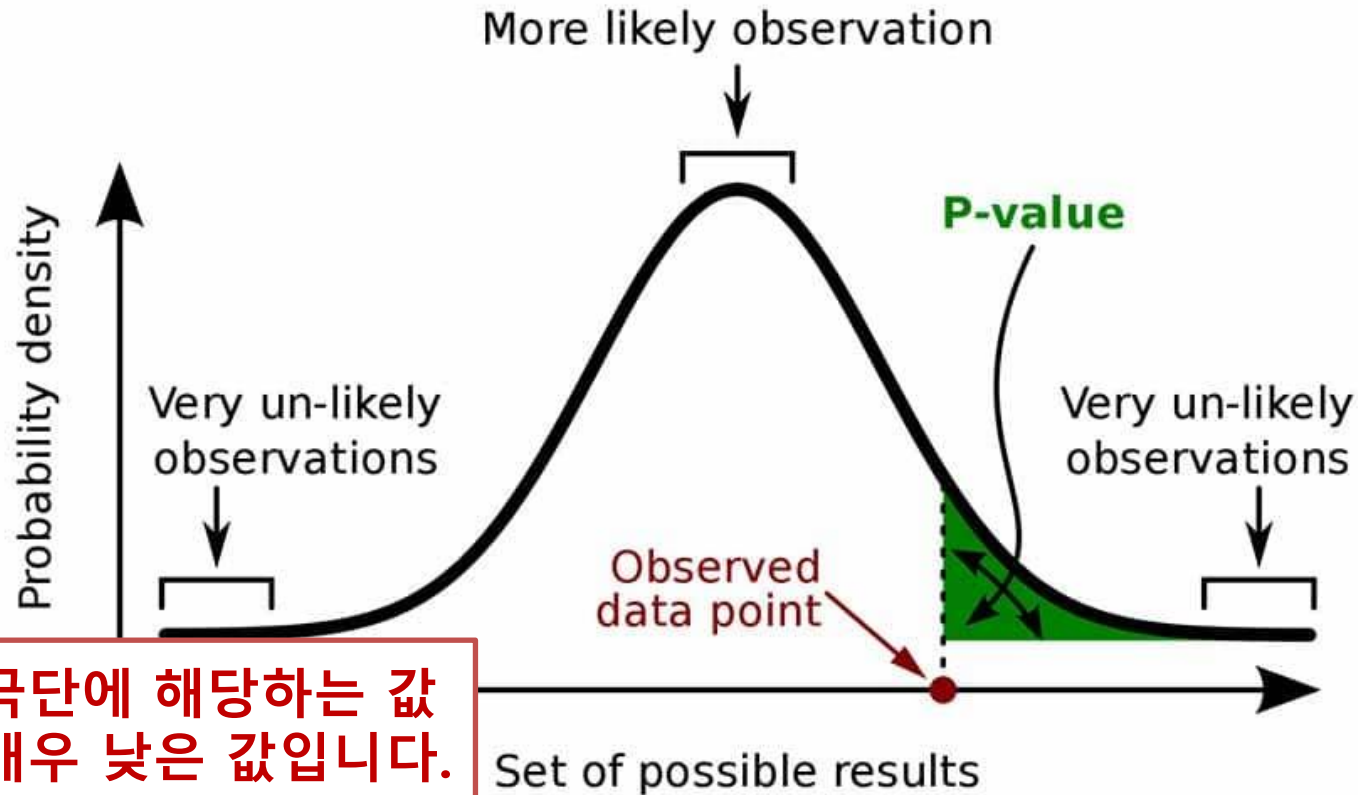


확률 분포(대체로 정규분포 가정)를 나타냅니다. 가운데 값일수록 관측 가능성이 높고, 양 극단은 발생 가능성이 낮습니다.

A **p-value** (shaded green area) is the probability of an observed (or more extreme) result assuming that the null hypothesis is true.

# p-value(유의확률)

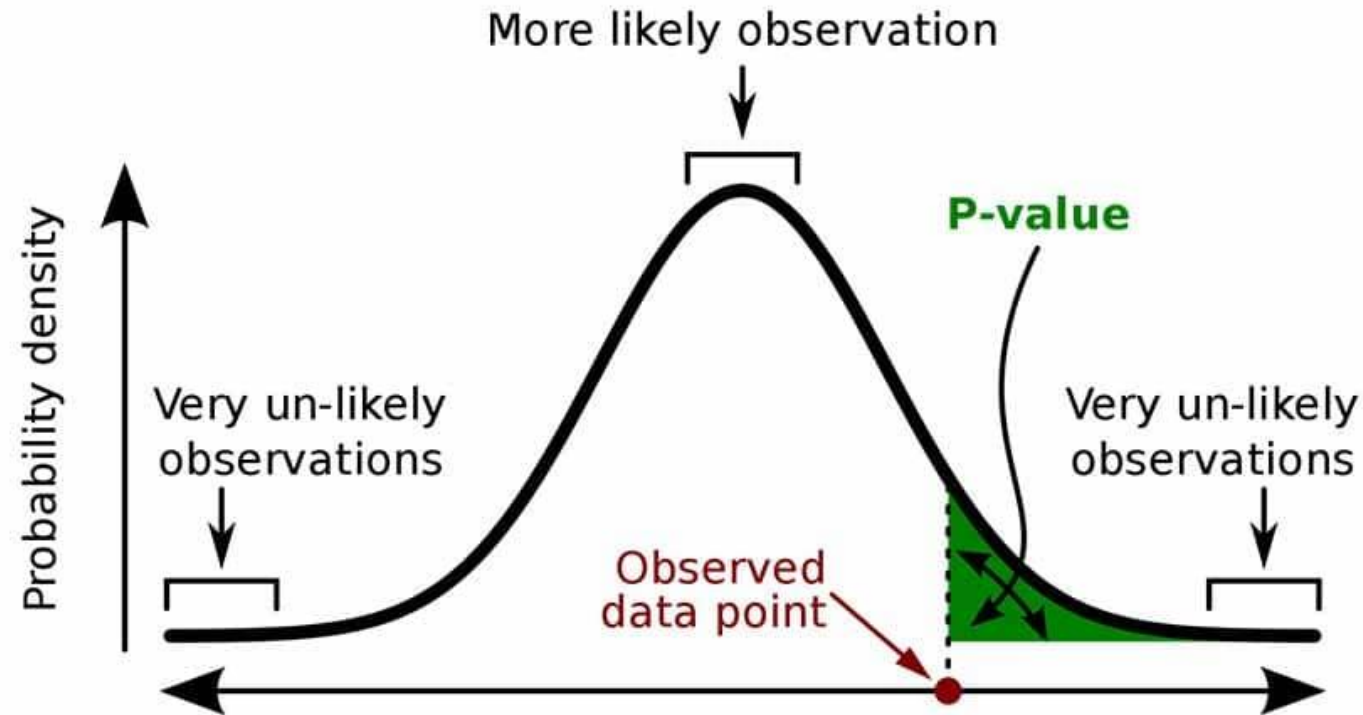
평균에 가까운 값들. 자주 일어나는, "자연스럽다"고 볼 수 있는 결과입니다



확률 분포에서 양 극단에 해당하는 값들. 발생 가능성이 매우 낮은 값입니다. 통계학에서 보통 이런 영역에 들어가면 "이상하다(unlikely)"고 보고 귀무가설( $H_0$ )을 의심하게 됩니다.

(green area) is the probability of an observed result assuming that the null hypothesis is true.

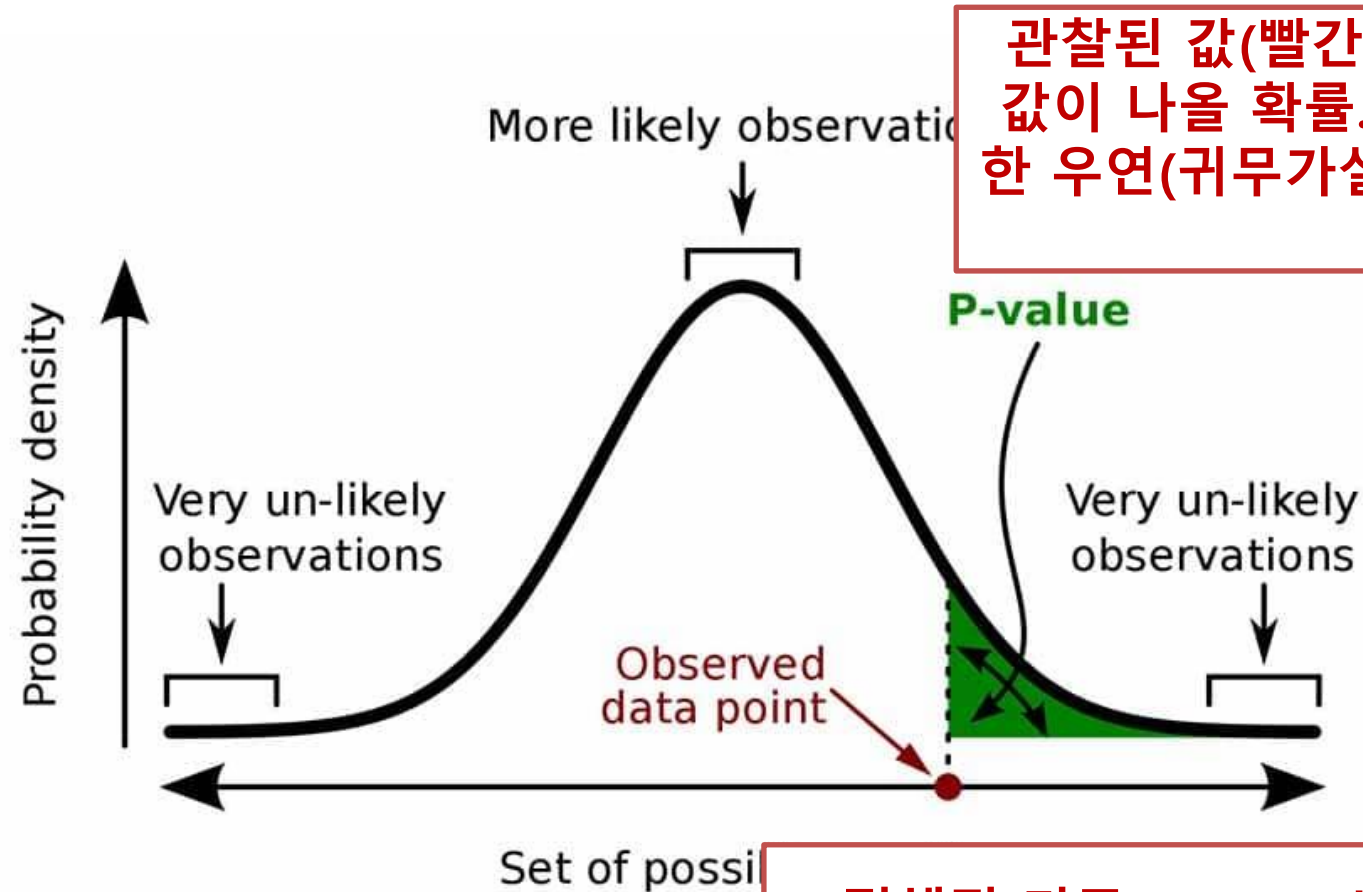
# p-value(유의확률)



A **p-value** (shaded green) (or more extreme) result as

실제로 관찰된 데이터 값. 그림에서는 오른쪽 꼬리 부분, 즉 "확률적으로 드물게 일어나는 구간"에 위치해 있습니다.

# p-value(유의확률)



관찰된 값(빨간 점) 이상으로 극단적인 값이 나올 확률. 내가 얻은 결과가 단순한 우연(귀무가설이 참일 때)으로 나타날 확률

정해진 기준( $\alpha=0.05$ )보다 작으므로, 귀무가설을 기각

A **p-value** (shaded green area) is the probability of a result (or more extreme) result assuming that the null hypothesis is true.



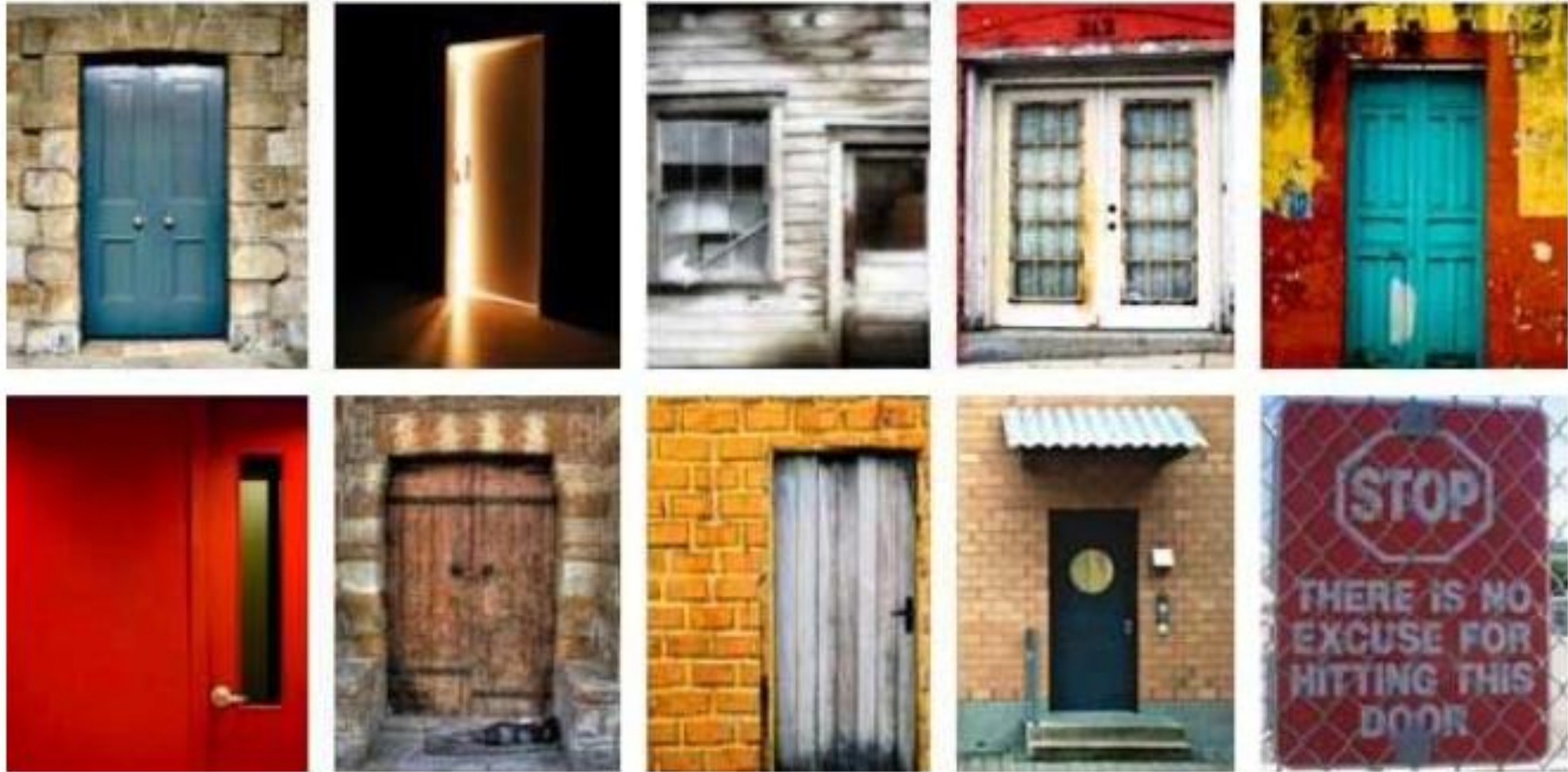
# 통계적인 한계

## ■ Bonferroni's Principle



# 통계적인 한계 (Bonferroni' s Principle )

- 어떤 문이 비밀의 문인지 (비밀의 문일 확률이 5%( $\alpha=0.05$ ))





# Bonferroni 교정 전

- 최소 1개 문을 "잘못 비밀의 문으로 판정"할 확률
  - $1 - (0.95)^{10} \approx 40\%$



# Bonferroni 교정 후

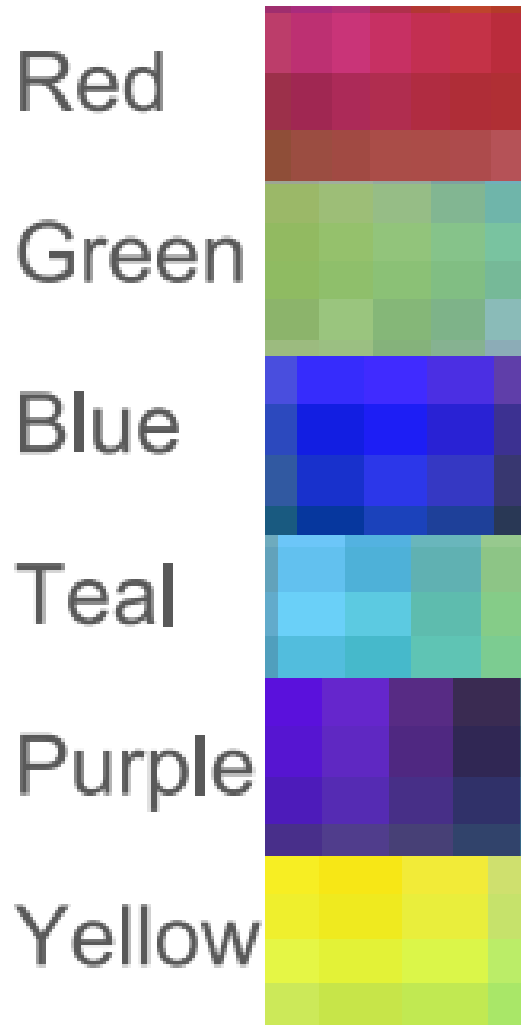
- 전체 오류 확률을 여전히 5%로 유지하려면,
  - 각 문에 대한 검정에서의 유의수준을  $0.05 \div 10 = 0.005$  (0.5%)로 낮춤.
    - 각 검정에서 틀릴 확률 = 0.005
  - 전체에서 단 한 번도 틀리지 않을 확률 =  $(1 - 0.005)^{10} = (0.995)^{10} \approx 0.951$
  - 따라서, 최소 한 번은 틀릴 확률 =  $1 - 0.951 \approx 0.049$  (약 4.9%)
- 개별 검정의 기준을 더 엄격하게 만들자

## ■ Bonferroni's Principle

- 통계적 가설검정에서 여러 번의 검정을 수행할 때, 우연에 의해 잘못된 결론(즉, 1종 오류, false positive)이 발생할 가능성을 통제하는 원칙
- 하나의 가설검정에서 유의수준( $\alpha$ )을 0.05로 설정하면, 잘못된 결론을 내릴 확률이 5%임. 그러나 서로 독립적인 검정을 여러 번 하면, 전체적으로 잘못된 결론을 낼 확률은 훨씬 커짐

# 통계적인 한계

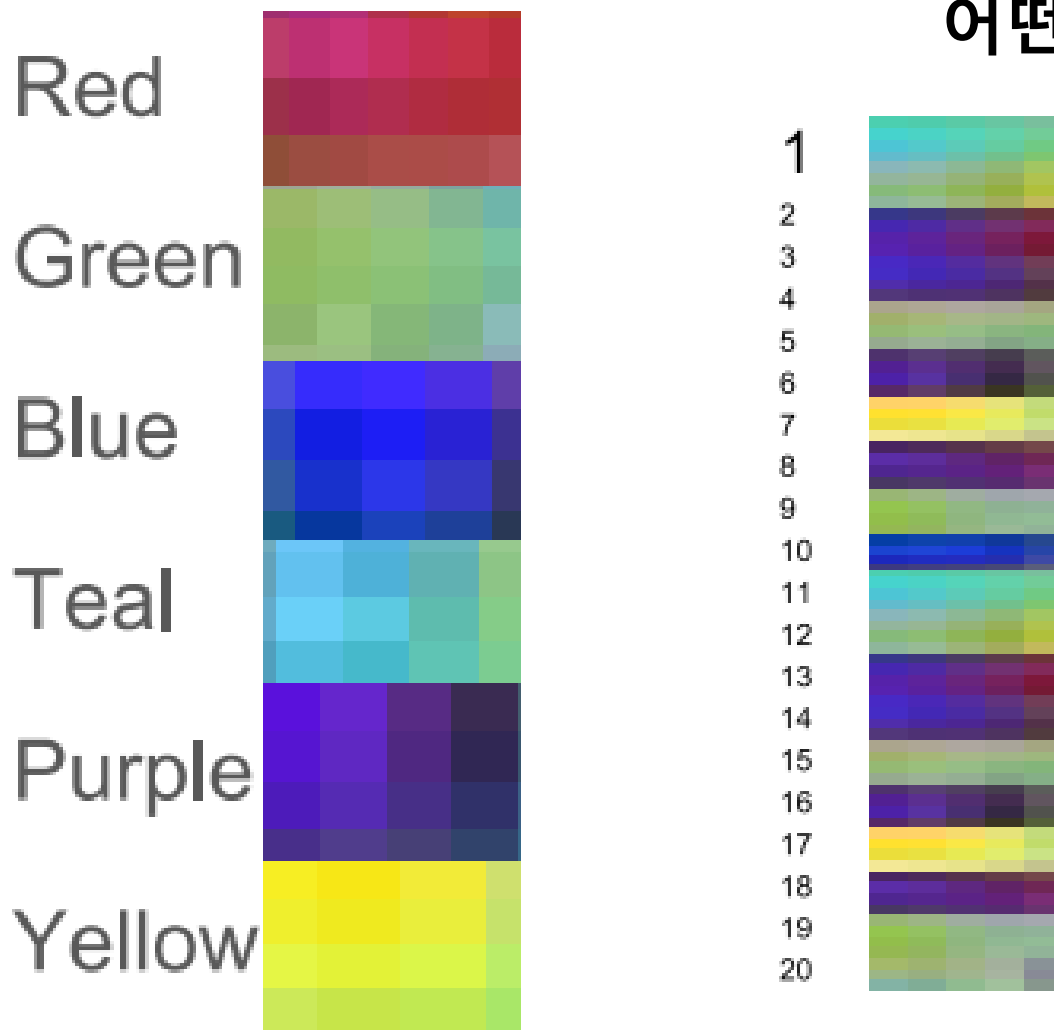
## ■ Bonferroni's Principle



어떤 아이폰 케이스가 가장 인기가 없을까요?

# 통계적인 한계

## ■ Bonferroni's Principle



어떤 아이폰 케이스가 가장 인기가 없을까요?

상위 20개의 판매량이 주어졌을때,  
여러분은 어떠한 결론을 내릴 수 있을 가요?

# 통계적인 한계

## ■ Bonferroni's Principle

Red

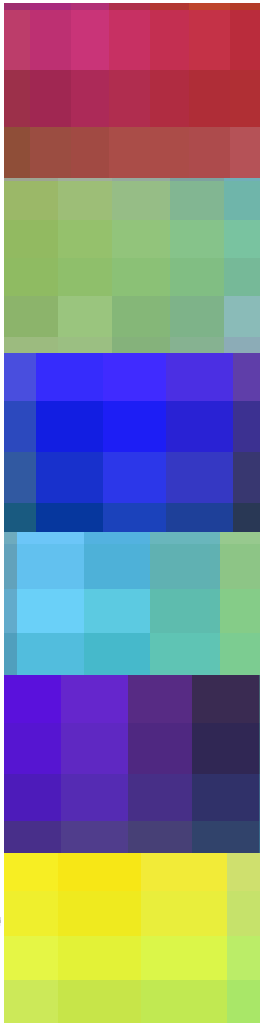
Green

Blue

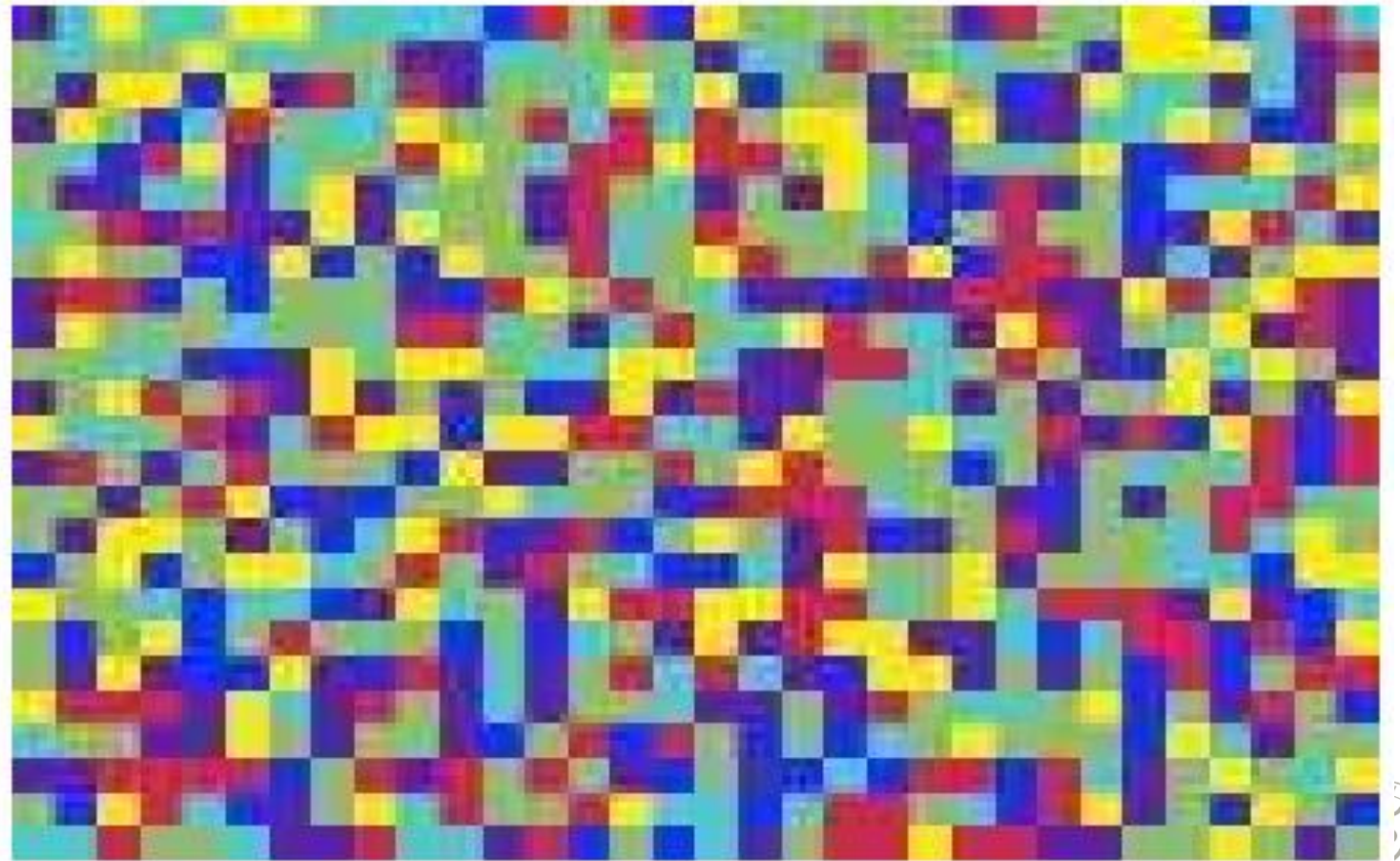
Teal

Purple

Yellow



어떤 아이폰 케이스가 가장 인기가 없을까요?



## ■ Bonferroni's Principle

사실 이외  $N$  발견 중의 어떠한 확률 을 계산하는 것은  
1개의 발견을 위한 테스트로서 확률을  $N$ 번을 곱한 것을 요구한다

요약하면, 우리는 데이터 내 많은 패턴(특징을) 관찰할 수 있다.  
누군가 우연하게 무엇인가를 찾기 전까지  
일반화되지 않은 무엇을 발견하는 것

# Bonferroni Correction

- 교정된 값은

$\alpha$  / 10번의 비교

$$0.05 / 10 = 0.005$$



# Bonferroni Correction

---

- 장점: 계산이 단순하고, 보수적으로 오류를 통제 가능
- 한계: 매우 보수적이어서, 검정력이 낮아질 수 있음(특히  $m$  이 클 경우)

# Normalizing

---

- 데이터는 자주 정규화가 필요하다. 동등한 범위로 숫자들을 놓아준다
- 전형적인 예제가: TF.IDF
- 예로,
  - 1-100 범위의 수집된 데이터와
  - 1-500 범위의 수집된 데이터와의 차이

# Normalizing

- 데이터는 자주 정규화가 필요하다. 동등한 범위로 숫자들을 놓아준다

특정한 단어가 문서 내에 얼마나 자주 등장하는지 **Term Frequency** (TF) 한 단어가 문서 집합 전체에서 얼마나 공통적으로 나타나 있는지를 나타내는 값 **Inverse Document Frequency** (IDF)

**Term Frequency:**

$$tf_{ij} = \frac{count_{ij}}{\max_k count_{kj}}$$

$$tf.idf_{ij} = tf_{ij} \times idf_i$$

**Inverse Document Frequency:**

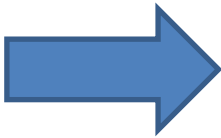
$$idf_i = \log_2\left(\frac{docs_*}{docs_i}\right) \propto \frac{1}{\frac{docs_i}{docs_*}}$$

where docs is the number of documents containing word  $i$ .

- $d_1$ : "The sky is blue."
- $d_2$ : "The sun is bright today."
- $d_3$ : "The sun in the sky is bright."
- $d_4$ : "We can see the shining sun, the bright sun."

문서들의 집합

## ■ STEP1: Stopwords을 filter out

- $d_1$ : "The sky is blue."
  - $d_2$ : "The sun is bright today."
  - $d_3$ : "The sun in the sky is bright."
  - $d_4$ : "We can see the shining sun, the bright sun."
- 
- $d_1$ : "sky blue"
  - $d_2$ : "sun bright today"
  - $d_3$ : "sun sky bright"
  - $d_4$ : "can see shining sun bright sun"

# TF-IDF

## STEP2: TF를 계산

- $d_1$ : "sky blue"
- $d_2$ : "sun bright today"
- $d_3$ : "sun sky bright"
- $d_4$ : "can see shining sun bright sun"

$f_{t,d}$

	blue	bright	can	see	shining	sky	sun	today
1	1	0	0	0	0	1	0	0
2	0	1	0	0	0	0	1	1
3	0	1	0	0	0	1	1	0
4	0	1	1	1	1	0	2	0



$$\text{tf}(t, d) = \frac{f_{t,d}}{\sum_{t'} f_{t',d}}$$

	blue	bright	can	see	shining	sky	sun	today
1	1/2	0	0	0	0	1/2	0	0
2	0	1/3	0	0	0	0	1/3	1/3
3	0	1/3	0	0	0	1/3	1/3	0
4	0	1/6	1/6	1/6	1/6	0	1/3	0

# TF-IDF

## STEP3: IDF를 계산

- $d_1$ : "sky blue"
- $d_2$ : "sun bright today"
- $d_3$ : "sun sky bright"
- $d_4$ : "can see shining sun bright sun"

$f_{t,d}$

	blue	bright	can	see	shining	sky	sun	today
1	1	0	0	0	0	1	0	0
2	0	1	0	0	0	0	1	1
3	0	1	0	0	0	1	1	0
4	0	1	1	1	1	0	2	0
n_t	1	3	1	1	1	2	3	1



$N = 4$

$$\text{idf}(t, D) = \log_{10} \frac{N}{n_t}$$

blue	bright	can	see	shining	sky	sun	today
0.602	0.125	0.602	0.602	0.602	0.301	0.125	0.602

$\log_{10} \frac{4}{1} = 0.602$

$\log_{10} \frac{4}{3} = 0.125$



# TF-IDF

## STEP4: TF-IDF를 계산

$tf(t, d)$

	blue	bright	can	see	shining	sky	sun	today
1	1/2	0	0	0	0	1/2	0	0
2	0	1/3	0	0	0	0	1/3	1/3
3	0	1/3	0	0	0	1/3	1/3	0
4	0	1/6	1/6	1/6	1/6	0	1/3	0

- TF-IDF: Multiply TF and IDF scores, use to rank importance of words within documents
- Most important word for each document is highlighted

**x**

$idf(t, D)$

	blue	bright	can	see	shining	sky	sun	today
	0.602	0.125	0.602	0.602	0.602	0.301	0.125	0.602

$$tfidf(t, d, D) = tf(t, d) \cdot idf(t, D)$$



	blue	bright	can	see	shining	sky	sun	today
1	<b>0.301</b>	0	0	0	0	0.151	0	0
2	0	0.0417	0	0	0	0	0.0417	<b>0.201</b>
3	0	0.0417	0	0	0	<b>0.100</b>	0.0417	0
4	0	0.0209	<b>0.100</b>	<b>0.100</b>	<b>0.100</b>	0	0.0417	0

# Normalizing

■ Standardize(표준화): 중앙값에 동등한 범위 위에 데이터의 다른 셋을 배치시키는 것

- Subtract the mean (중앙값)
- 표준 편차로 나눈다

$$z_i = \frac{x_i - \bar{x}}{s_x}$$

Standardization

$$x_{\text{norm}} = \frac{x - \min(x)}{\max(x) - \min(x)}$$

Max-min normalization



# Example: Standardization vs Max-min normalization

	Country	Age	Salary	Purchased
1	France	44	72000	No
2	Spain	27	48000	Yes
3	Germany	30	54000	No
4	Spain	38	61000	No
5	Germany	40		Yes
6	France	35	58000	Yes
7	Spain		52000	No
8	France	48	79000	Yes
9	Germany	50	83000	No
10	France	37	67000	Yes

The range of Age: 27 - 50

The range of Salary: 48,000 - 83,000

```
dataset['Age'].min()
```

27.0

```
dataset['Salary'].min()
```

48000.0

```
dataset['Age'].max()
```

50.0

```
dataset['Salary'].max()
```

83000.0

# Example: Standardization vs Max-min normalization

## Standardisation

	Age	Salary
0	0.758874	7.494733e-01
1	-1.711504	-1.438178e+00
2	-1.275555	-8.912655e-01
3	-0.113024	-2.532004e-01
4	0.177609	6.632192e-16
5	-0.548973	-5.266569e-01
6	0.000000	-1.073570e+00
7	1.340140	1.387538e+00
8	1.630773	1.752147e+00
9	-0.258340	2.937125e-01

## Max-Min Normalization

	Age	Salary
0	0.739130	0.685714
1	0.000000	0.000000
2	0.130435	0.171429
3	0.478261	0.371429
4	0.565217	0.450794
5	0.347826	0.285714
6	0.512077	0.114286
7	0.913043	0.885714
8	1.000000	1.000000
9	0.434783	0.542857

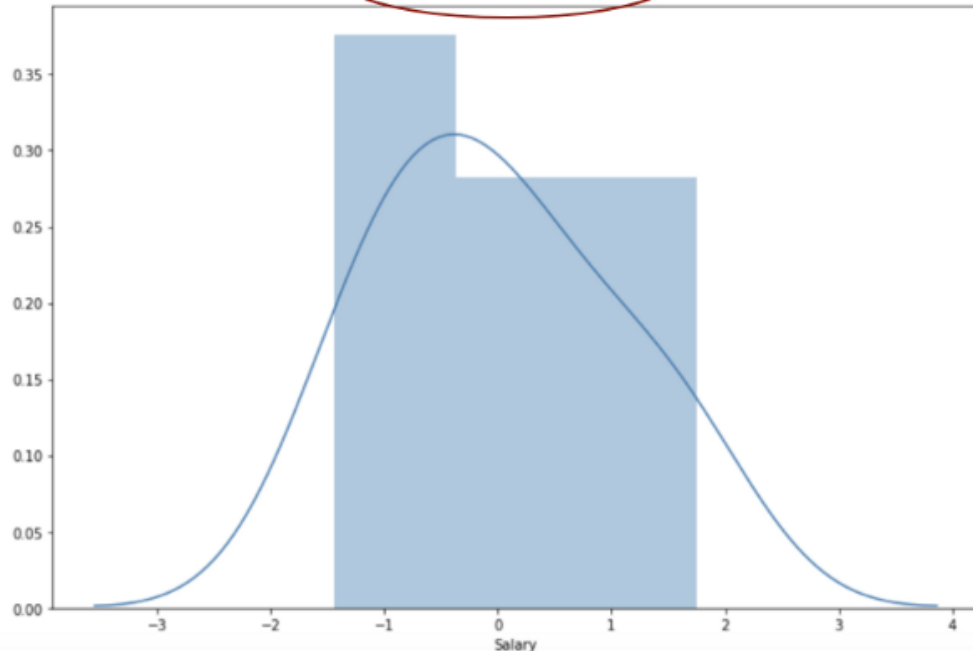
# Example: Standardization vs Max-min normalization

Column: Salary

Standard Deviation (Salary):  
Max-Min Normalization (0.33) < Standardisation (1.05)

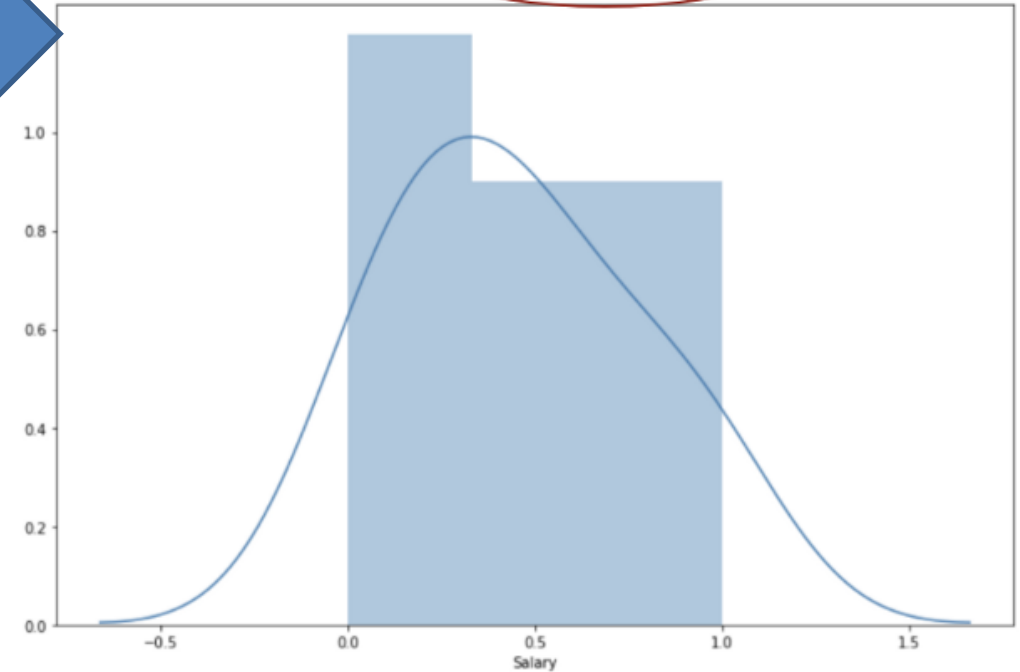
Standardisation

Standard Deviation of sc\_Salary is 1.0540925533894598



Max-Min Normalisation

Standard Deviation of df\_MinMax\_Salary is 0.33040284015892535



# Example: Standardization vs Max-min normalization

Column:Age

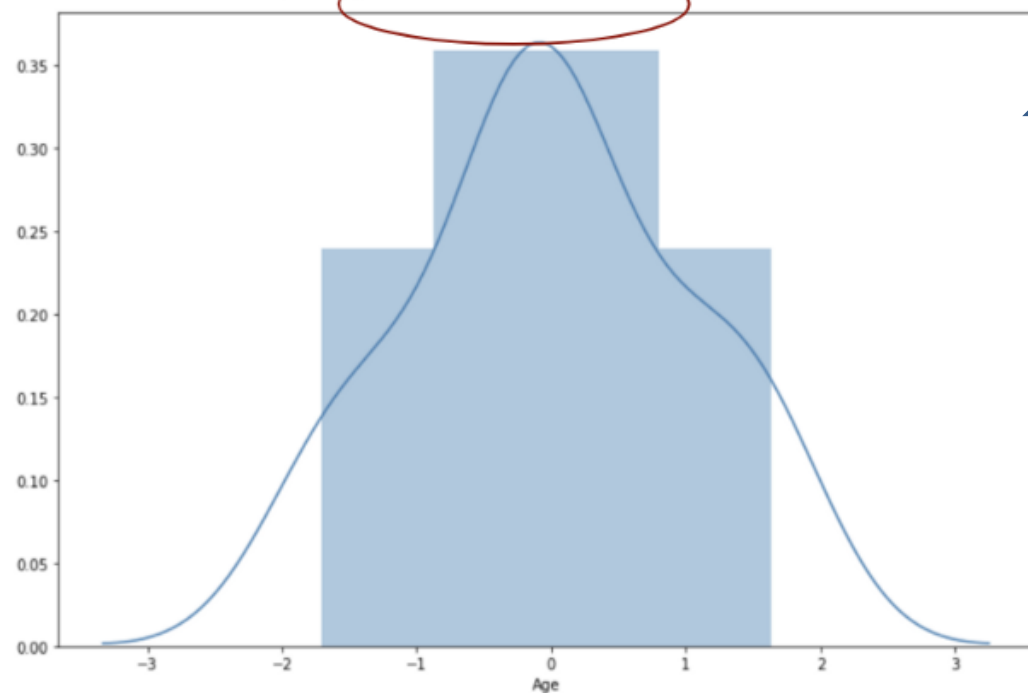
Standard Deviation (Age):  
Max-Min Normalization (0.315) < Standardisation (1.05)

Standardisation

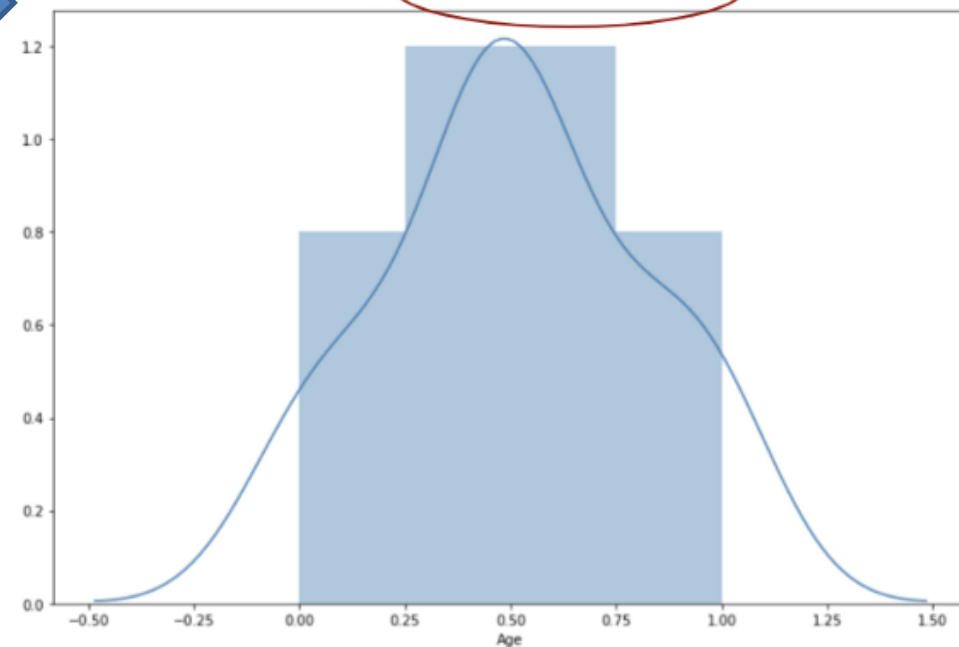
표준 편차가 작다는 것은  
데이터가 밀집되어 있다

Max-Min Normalisation

Standard Deviation of sc\_Salary is 1.0540925533894598



Standard Deviation of df\_MinMax\_Age is 0.3153816182405694



# Power Law

---

- 가장 많은 것부터 가장 적은 것까지 순서대로 정렬했을 때 빈번하게 출몰되는 패턴을 특징화 한 것
  - 인구밀도
  - 웹페이지의 링크
  - 상품의 판매량
  - 단어의 출현 빈도
- 인기(Popularity 기반의 통계)



# Power Law

---

- 어떤 변수  $y$ 가 다른 변수  $x$ 의 거듭제곱에 비례하는 관계

$$y = C \cdot x^{-\alpha}$$

- 큰 사건은 드물고, 작은 사건은 자주 발생한다

# Power Law의 특징

## ■ Scale-free (스케일 불변성)

- 축척을 달리해도 같은 형태를 유지합니다. (로그-로그 좌표에 그리면 직선 형태)

## ■ Heavy tail (두꺼운 꼬리 분포)

- 평균 근처에 값이 몰려 있는 정규분포와 달리, 극단적인 값도 꽤 자주 나타납니다.

## ■ 소수의 큰 값, 다수의 작은 값

- 예: 소수의 대기업이 대부분의 매출을 차지, 소수의 인기 영상이 대부분의 조회수를 차지

# Power Law의 예시

## ■ 사회현상

- 도시 인구 분포 (몇 개의 초거대 도시 + 수많은 중소도시)
- 부의 분포 (상위 1%가 전체 자산의 대부분을 소유)

## ■ 인터넷/미디어

- 웹사이트 트래픽 (몇몇 대형 사이트가 대부분의 방문자 차지)
- 유튜브 조회수 ((상위 1% 영상이 전체 조회수 대부분 차지)

## ■ 자연현상

- 지진 규모 분포 (작은 지진은 자주, 큰 지진은 드물지만 발생)
- 단어 사용 빈도 (Zipf's Law, 상위 몇 단어가 전체 언어 사용의 큰 비중 차지)

# Power Law을 왜 적용하는가?

- 실제 데이터가 멱법칙을 따르는 경우가 많음
  - 자연현상, 사회현상, 기술 네트워크에서 관측되는 데이터는 정규분포보다는 **극단값이 자주 발생**
  - 모델링할 때 **정규분포 대신 멱법칙을 적용해야 현실을 더 잘 설명**할 수 있습니다.
- 극단적 사건을 설명
- 예측과 의사결정에 유리
  - 리스크 관리
  - 마케팅/추천 시스템
  - 네트워크 보안

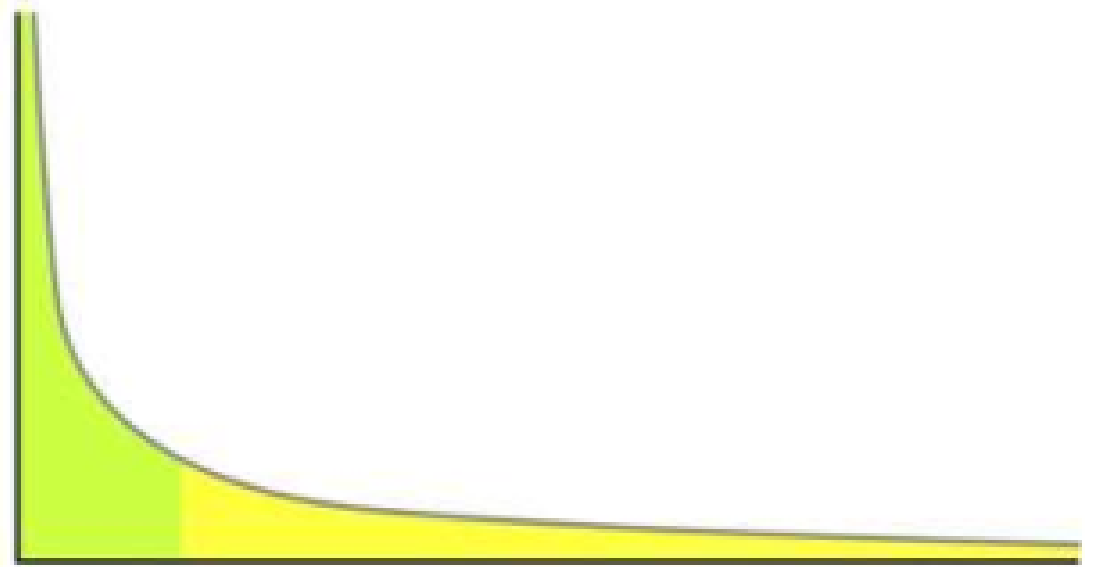
# Power Law

$$\log y = b + a \log x$$

raising to the natural log:

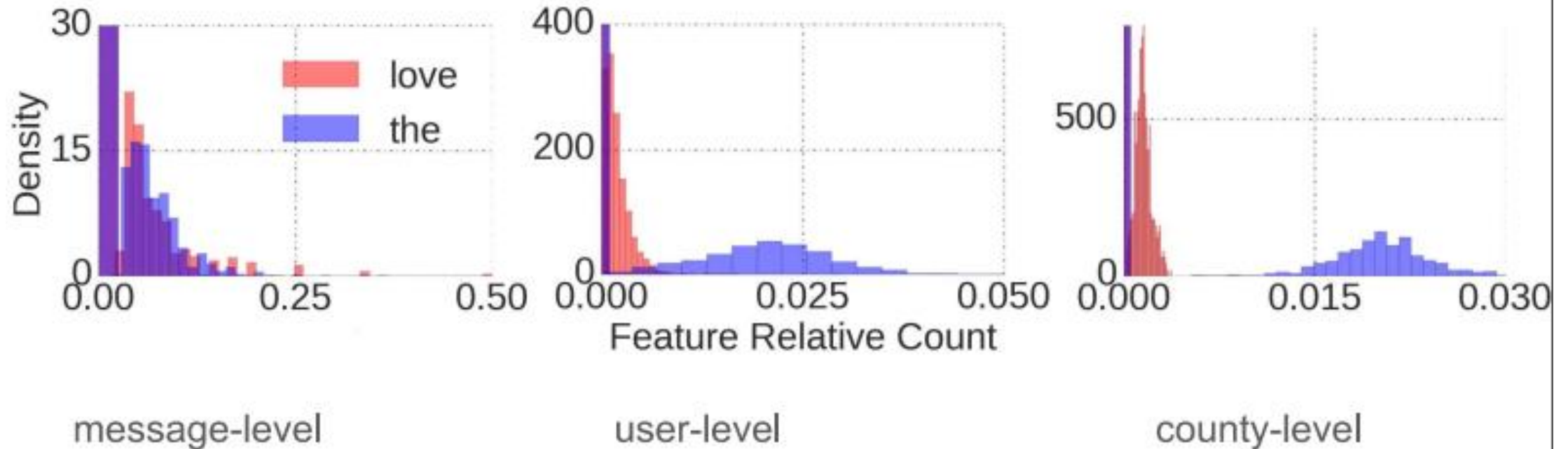
$$y = e^b e^{a \log x} = e^b x^a = cx^a$$

where  $c$  is just a constant



**Matthew Effect를 특징화함**  
**부자들은 더 부자가 된다**

# Power Law



Almodaresi, F., Ungar, L., Kulkarni, V., Zakeri, M., Giorgi, S., & Schwartz, H. A. (2017). On the Distribution of Lexical Features at Multiple Levels of Analysis. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics* (pp. 79-84).

# Hash Functions and Indexes

---

## ■ Review:

$h$ : hash-key  $\rightarrow$  bucket-number

목표: 버킷들에 hash-keys를 균등하게(uniformly) 배분하는 것  
예로: 단어의 개수를 저장



# Hash Function (해시 함수)

- 임의 길이의 입력값(Input)을 받아 고정 길이의 출력값(Hash Value, Digest) 을 반환하는 함수.
- 같은 입력은 항상 같은 출력으로 매핑되며, 작은 입력 변화에도 출력이 크게 달라짐.
- 특징
  - Deterministic: 동일 입력 → 동일 출력
  - Uniformity: 가능한 한 출력 공간에 균등하게 분포 → 충돌 최소화
  - Efficiency: 빠르게 계산 가능
  - Irreversibility: 일반적으로 해시값만으로 원래 입력을 복원 불가



# Hash Functions and Indexes

## ■ Review:

$h$ : hash-key  $\rightarrow$  bucket-number

목표: 버킷들에 hash-keys를 균등하게(uniformly) 배분하는 것  
예로: 단어의 개수를 저장

$$h(word) = \left( \sum_{char \in word} \text{ascii}(char) \right) \% \#buckets$$

# Hash Functions and Indexes

## ■ Review:

$h$ : hash-key  $\rightarrow$  bucket-number

목표: 버킷들에 hash-keys를 균등하게(uniformly) 배분하는 것  
예로: 단어의 개수를 저장

$$h(word) = \left( \sum_{char \in word} ascii(char) \right) \% \#buckets$$

Data structures utilizing hash-tables (i.e.  $O(1)$  lookup; dictionaries, sets in python) are a friend of big data algorithms! Review further if needed.

# Hash Functions and Indexes

## ■ Review:

$h$ : hash-key  $\rightarrow$  bucket-number

목표: 버킷들에 hash-keys를 균등하게(uniformly) 배분하는 것  
예로: 단어의 개수를 저장

**Database Indexes:** Retrieve all records with a given *value*. (also review if unfamiliar / forgot)

Data structures utilizing hash-tables (i.e.  $O(1)$  lookup; dictionaries, sets in python) are a friend of big data algorithms! Review further if needed.

# Hash Functions and Indexes

---

- `import hashlib`
- `# 문자열 입력`
- `text = "DataScienceLabs"`
- `# SHA-256 해시 함수 적용`
- `hash_value = hashlib.sha256(text.encode()).hexdigest()`
- `print("입력값 :", text)`
- `print("SHA-256 해시값 :", hash_value)`



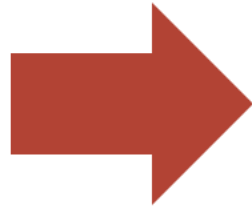
# One hot encoding

- One-Hot Encoding은 범주형 데이터(categorical data)를 벡터 형태의 수치 데이터로 변환하는 기법
- 각 범주(category)에 대해 고유한 벡터를 만들고, 해당 범주만 1, 나머지는 0으로 표시
- 특징
  - 범주형 데이터를 수학적 계산이 가능한 형태로 변환
  - 각 **카테고리 간 순서(ordinal relationship)**가 없음을 보장
  - 머신러닝, 딥러닝 모델에서 널리 활용 (특히 입력 feature로)

# One hot encoding

Vocabulary:

Man, woman, boy,  
girl, prince,  
princess, queen,  
king, monarch



	1	2	3	4	5	6	7	8	9
man	1	0	0	0	0	0	0	0	0
woman	0	1	0	0	0	0	0	0	0
boy	0	0	1	0	0	0	0	0	0
girl	0	0	0	1	0	0	0	0	0
prince	0	0	0	0	1	0	0	0	0
princess	0	0	0	0	0	1	0	0	0
queen	0	0	0	0	0	0	1	0	0
king	0	0	0	0	0	0	0	1	0
monarch	0	0	0	0	0	0	0	0	1

Each word gets  
a 1x9 vector  
representation

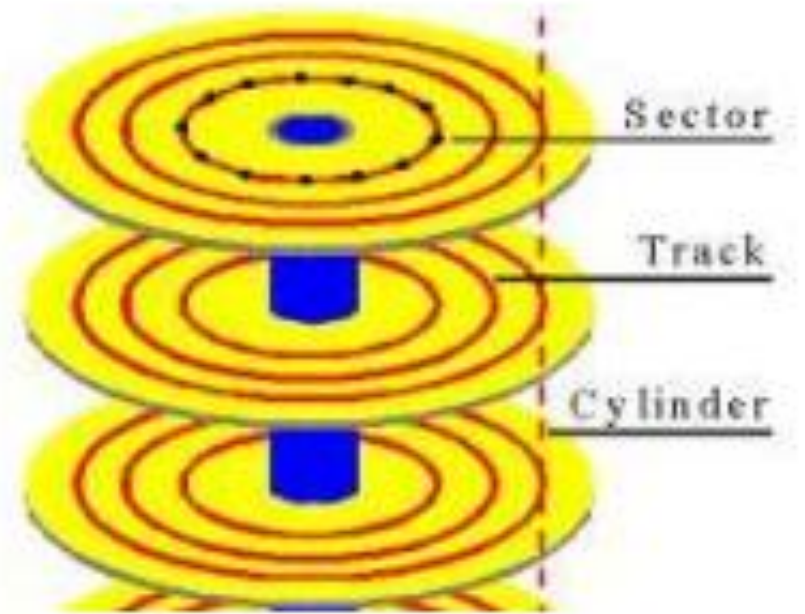


# One hot encoding

- `from sklearn.preprocessing import OneHotEncoder`
- `import numpy as np`
- `colors = np.array(["Red", "Green", "Blue", "Red"]).reshape(-1, 1)`
- `encoder = OneHotEncoder(sparse=False)`
- `one_hot = encoder.fit_transform(colors)`
  - # fit: 데이터의 고유한 카테고리 추출
  - # transform: 카테고리를 one-hot 벡터로 변환
- `print("Categories:", encoder.categories_)`
- `print("One-hot encoded:\n", one_hot)`

# IO Bounded

- $10^5$  차이 단어 읽기: 디스크로부터 vs 메인 메모리 부터
- IO Bound: 디스크에 읽기/쓰기를 진행하는 것은 가장 큰 성능 저하
- 100GB 시작한다면, 읽는데 10분이 소!



# Structured Data vs Unstructured Data

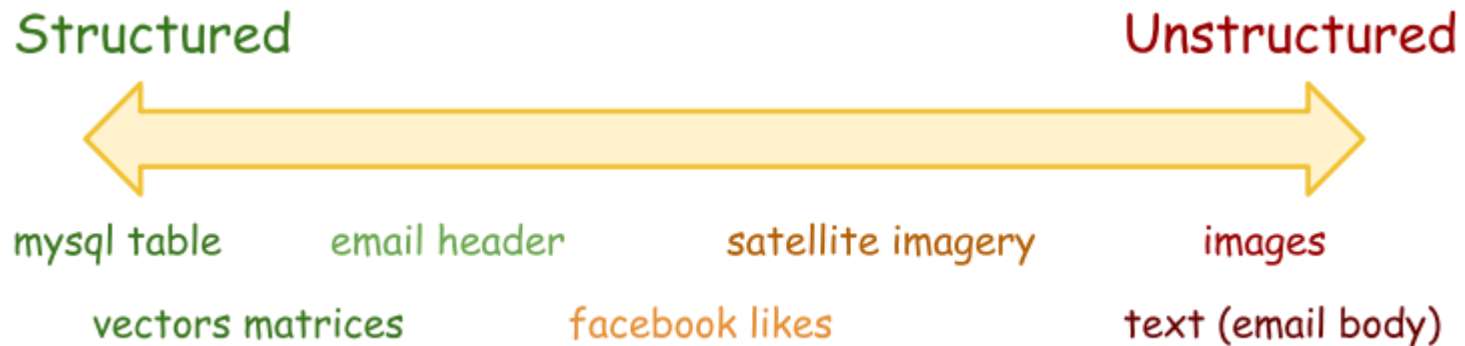
Structured

Unstructured



Unstructured ~ 흥미(interest)을 얻는데 처리가 요구된다  
Feature extraction은 unstructured 를 structured된 형태로 변환할 때 사용된다  
Unstructured 데이터 내에 특징 가지 수는 거의 무제한에 가깝다

# Data



Unstructured ~ 흥미(interest)을 얻는데 처리가 요구된다  
Feature extraction은 unstructured 를 structured된 형태로 변환할 때 사용된다  
Unstructured 데이터 내에 특징 가지 수는 거의 무제한에 가깝다

## ■ 빅데이터의 정의와 특징

- 방대한 양의 데이터를 다루며 **모델과 요약의 일반화**가 핵심

## ■ 통계적 가설 검정의 한계

- **가설(Hypothesis), 귀무가설( $H_0$ ), 대립가설( $H_1$ )** 개념 이해
- p-value: 귀무가설이 참일 때 현재보다 극단적인 결과가 나올 확률
- Bonferroni's Principle: 다중 검정 시 오류 누적 → 보수적 교정 필요

## ■ 데이터 처리 및 분석 기법

- **정규화(Normalization):** 데이터 범위를 통일 (예: TF-IDF, 표준화, min-max)
- **Power Law:** “작은 사건은 자주, 큰 사건은 드물지만 무시할 수 없음” → 극단값 설명 가능
- **Hash Functions & Indexes:** 데이터 검색 및 저장 최적화 (효율성, 충돌 최소화)
- **One-Hot Encoding:** 범주형 데이터를 수치화하여 머신러닝 입력 가능

## ■ 실제 데이터 분석의 도전과제

- I/O Bounded 문제: 대용량 데이터는 디스크 접근이 병목
- Structured vs Unstructured Data: 정형 데이터: 표, 관계형 DB에 잘 맞음  
비정형 데이터: 이미지, 텍스트, 영상 → 특징 추출 후 구조화 필요