

Big data 분석이란?

컴퓨터AI공학부
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

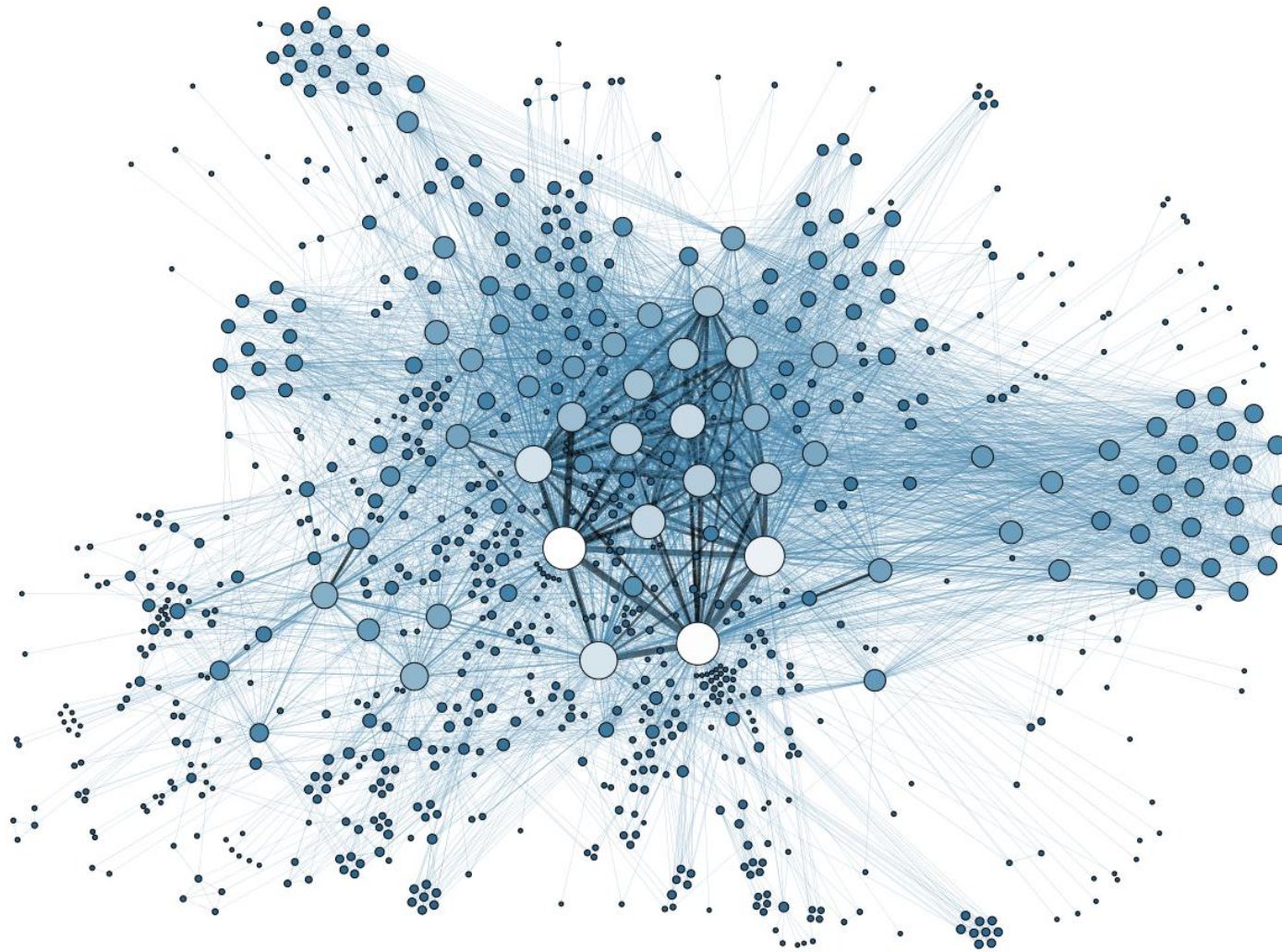
3 observations

	p_1	p_2	p_3	p_4	p_5	p_6
n_1						
n_2						
n_3						

<https://www.statology.org>



Graphs



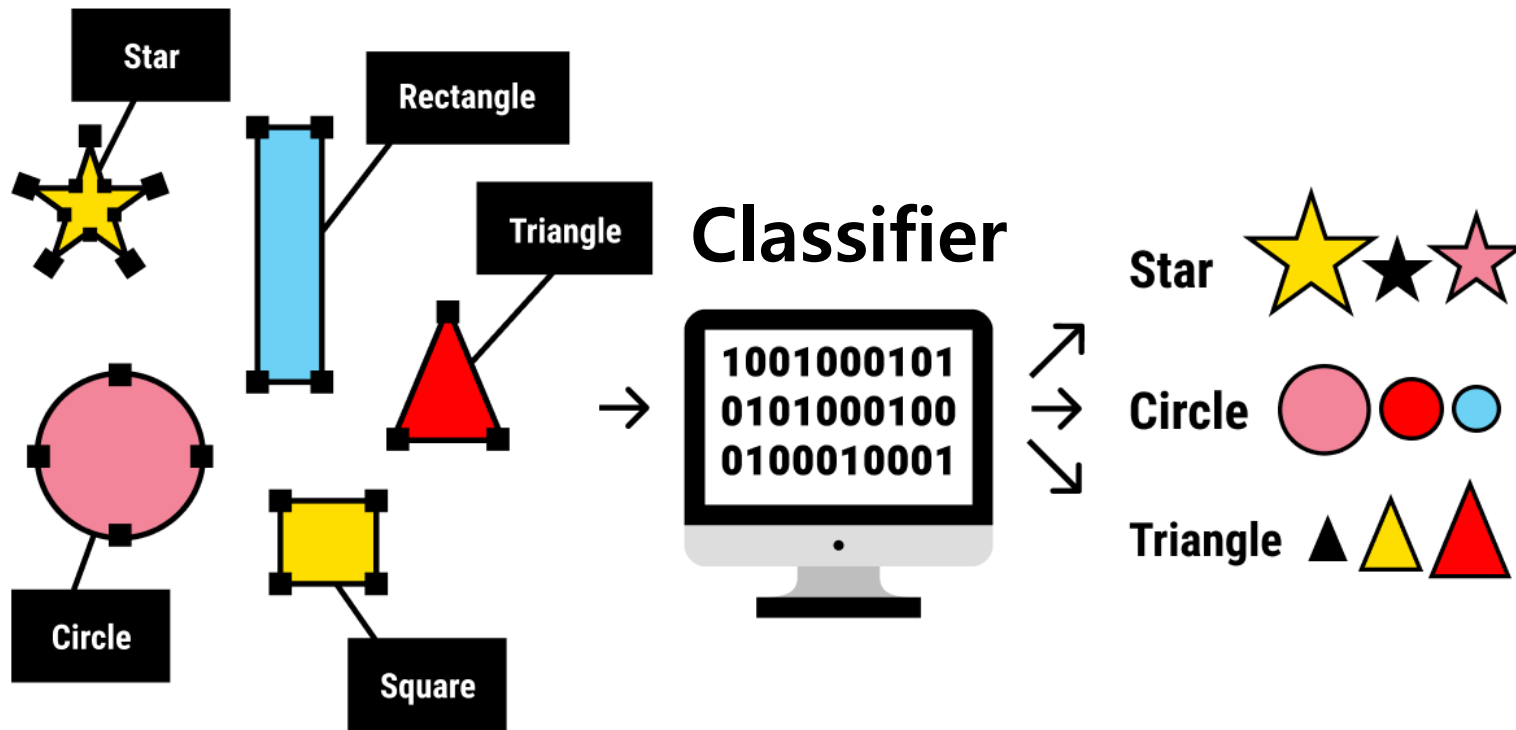
Infinite data



<https://www.onaudience.com>



Labeled Data

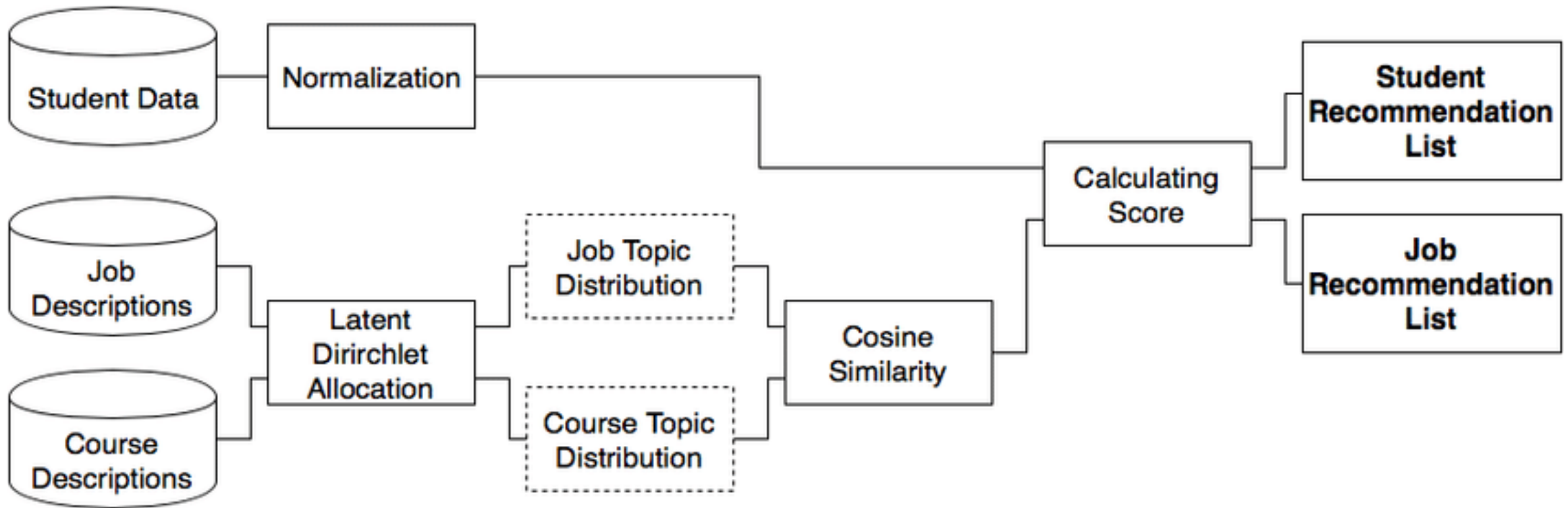


Big data analytics - 본 수업에서 배우는 내용

- 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



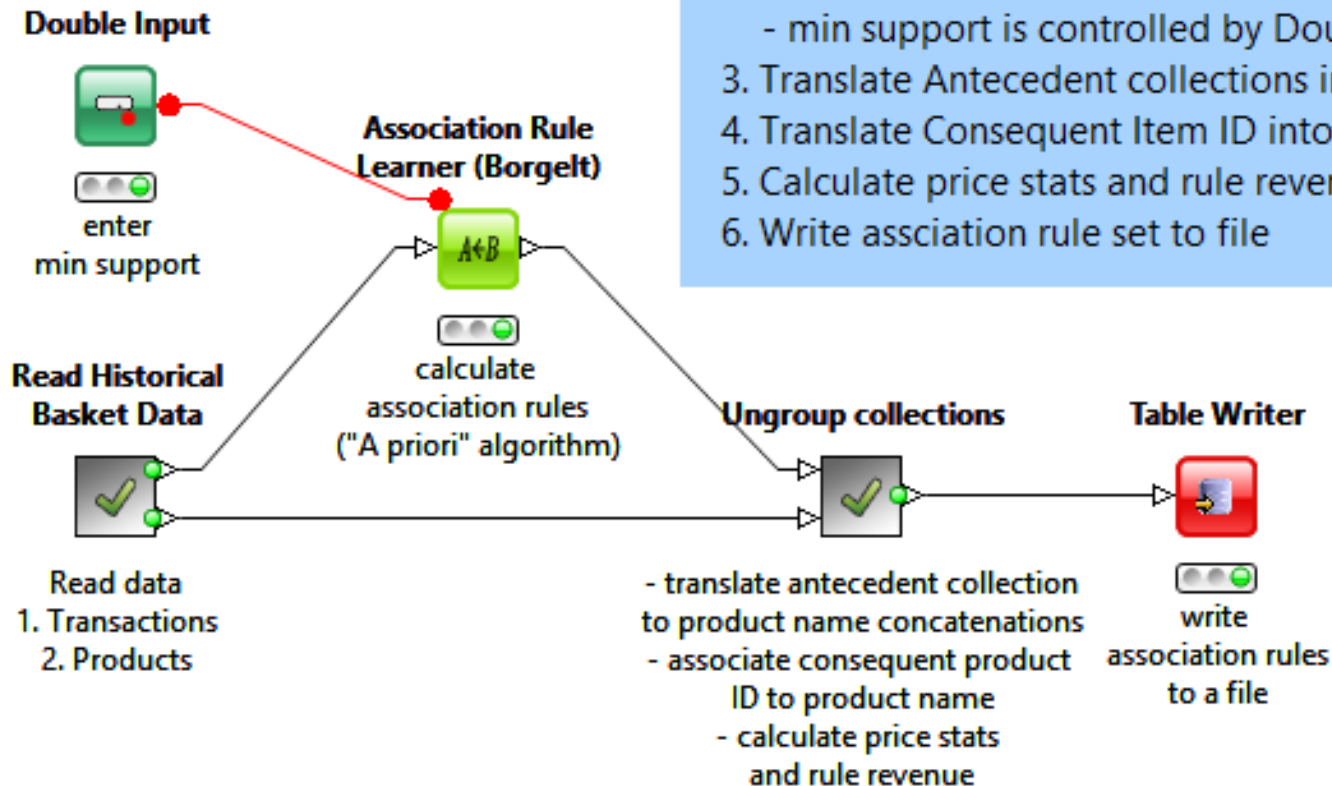
Pipeline of Recommendation system



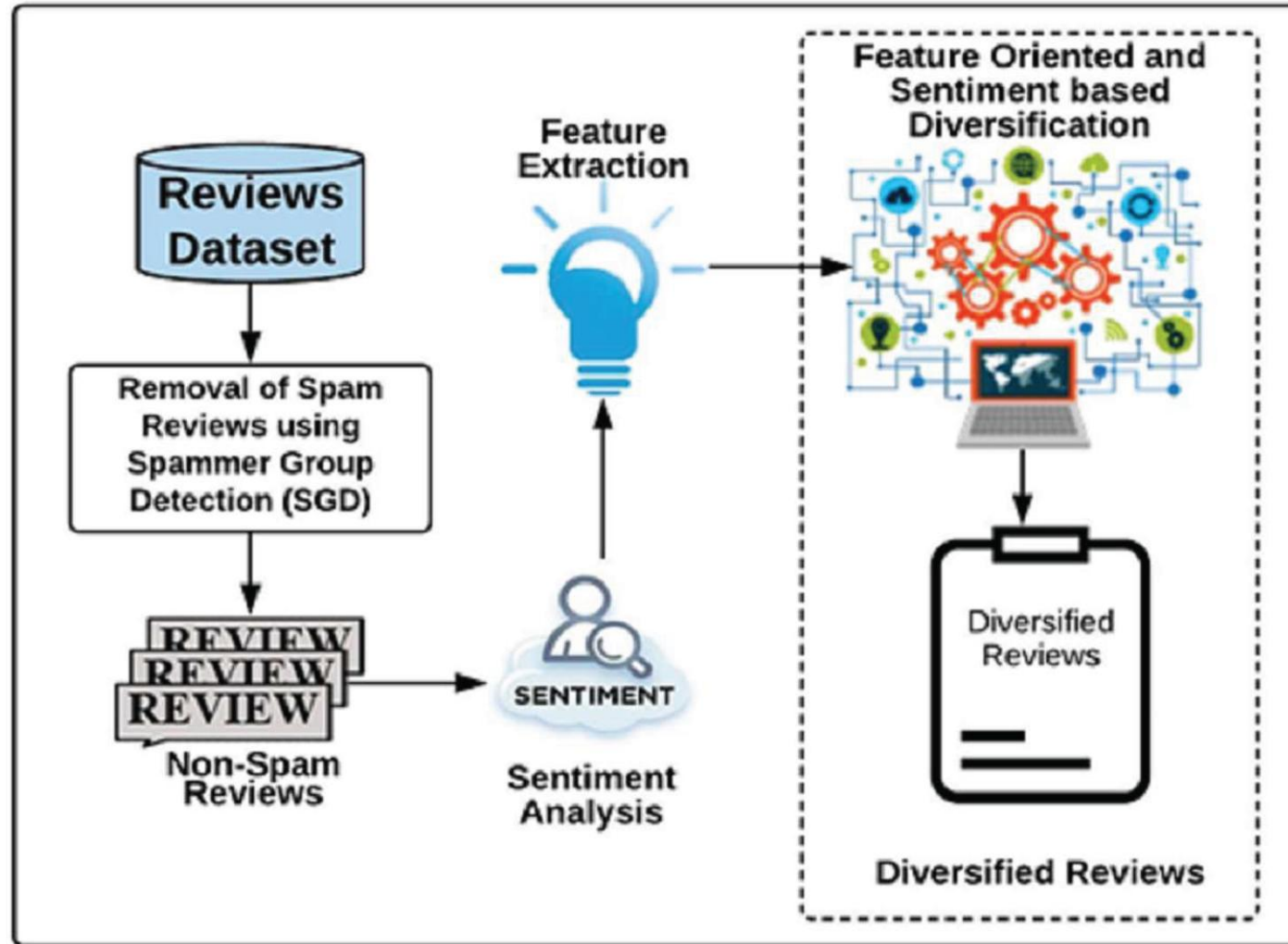
Pipeline of Market-basket analysis

1. Market Basket Analysis: Build Association Rules

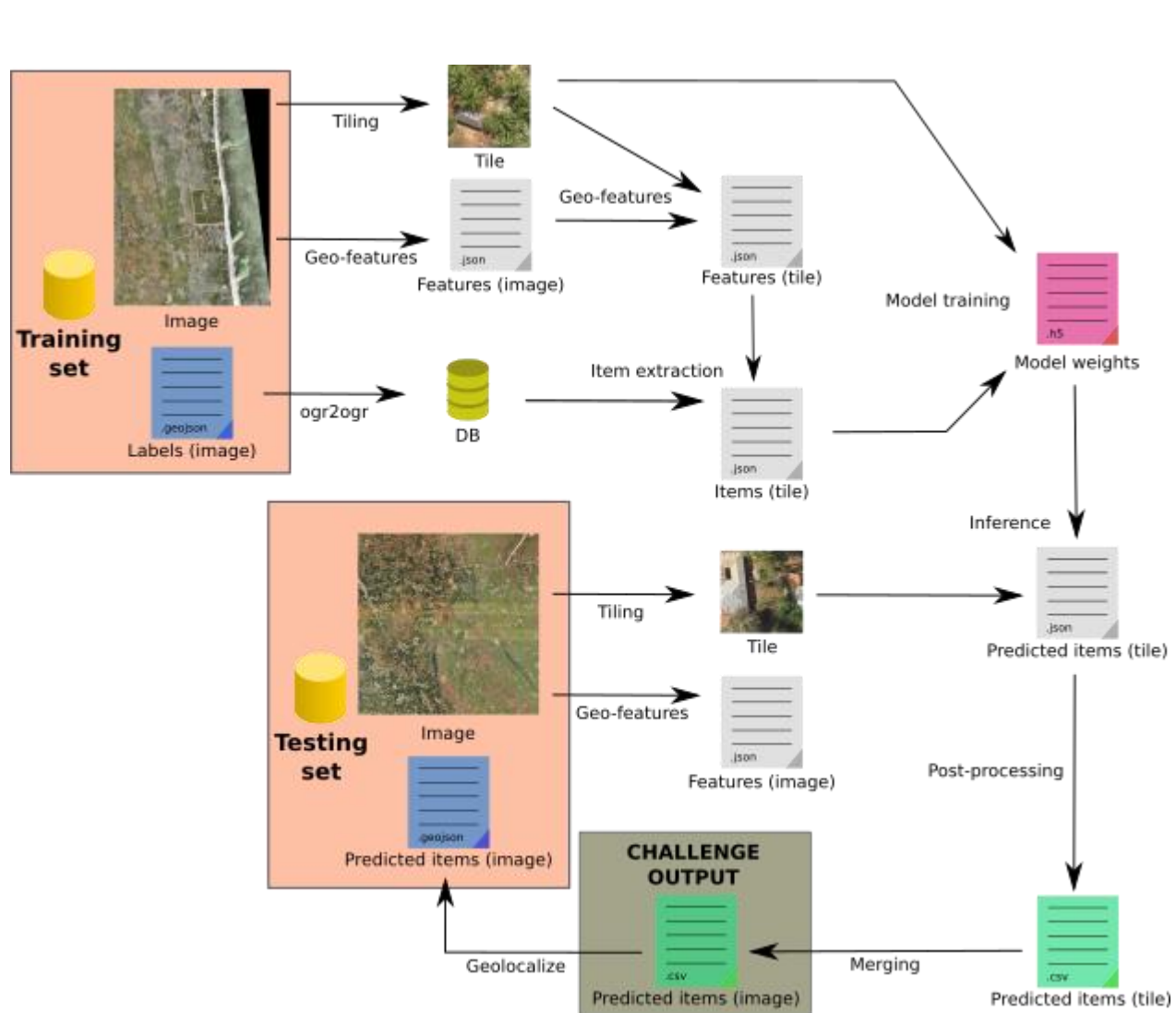
1. Read Transaction/Basket data and Product data
2. Using "A priori" algorithm, build association rule set
 - min. set size = 1
 - min rule confidence = 10%
 - min support is controlled by Double Input Quickform node in %
3. Translate Antecedent collections into product name concatenations
4. Translate 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

■ 대립가설(Alternative: H1), 혹은 대안가설

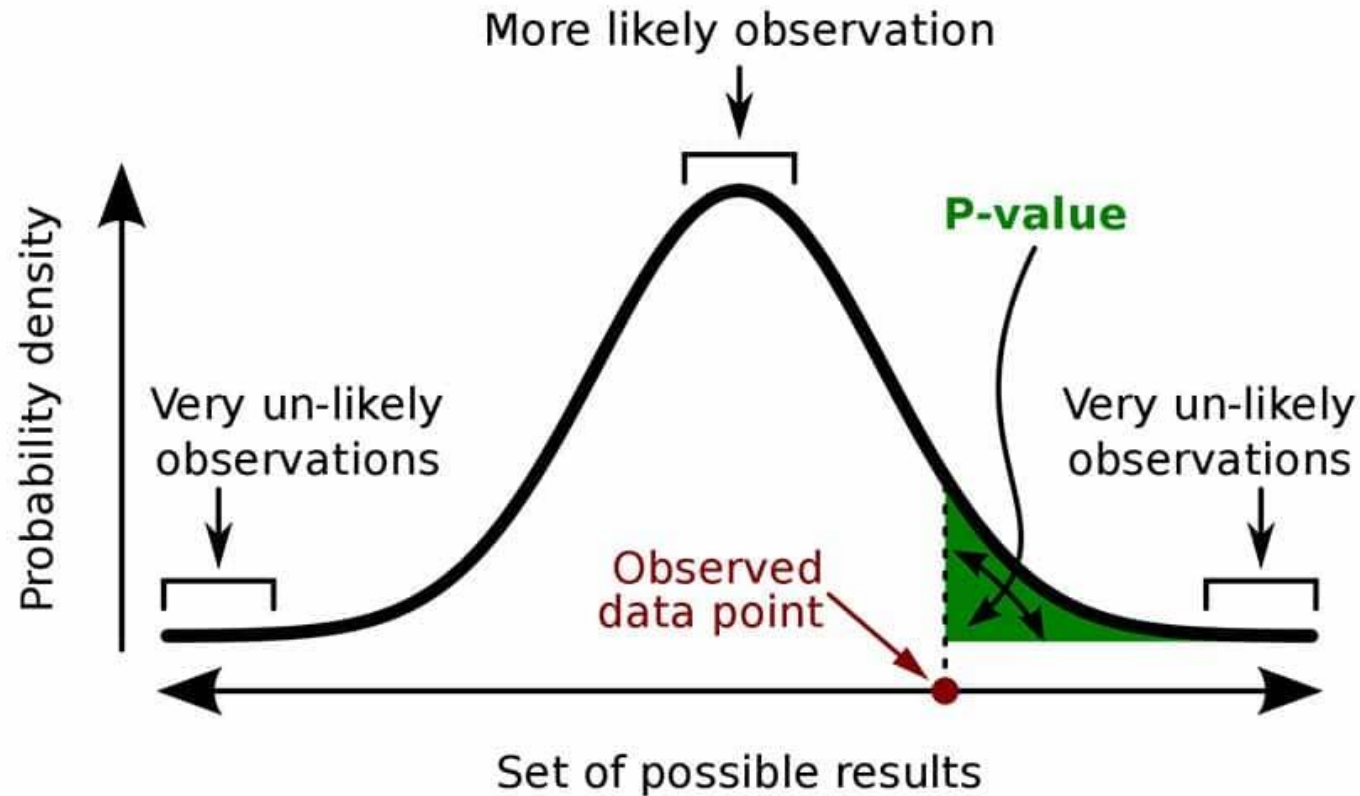
- 귀무가설이 거짓이라면 대안적으로 참이 되는 가설
- ~와 차이가 있다. ~의 효과는 있다. ~와 다르다



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.

통계적인 한계

■ Bonferroni's Principle



■ 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(α)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

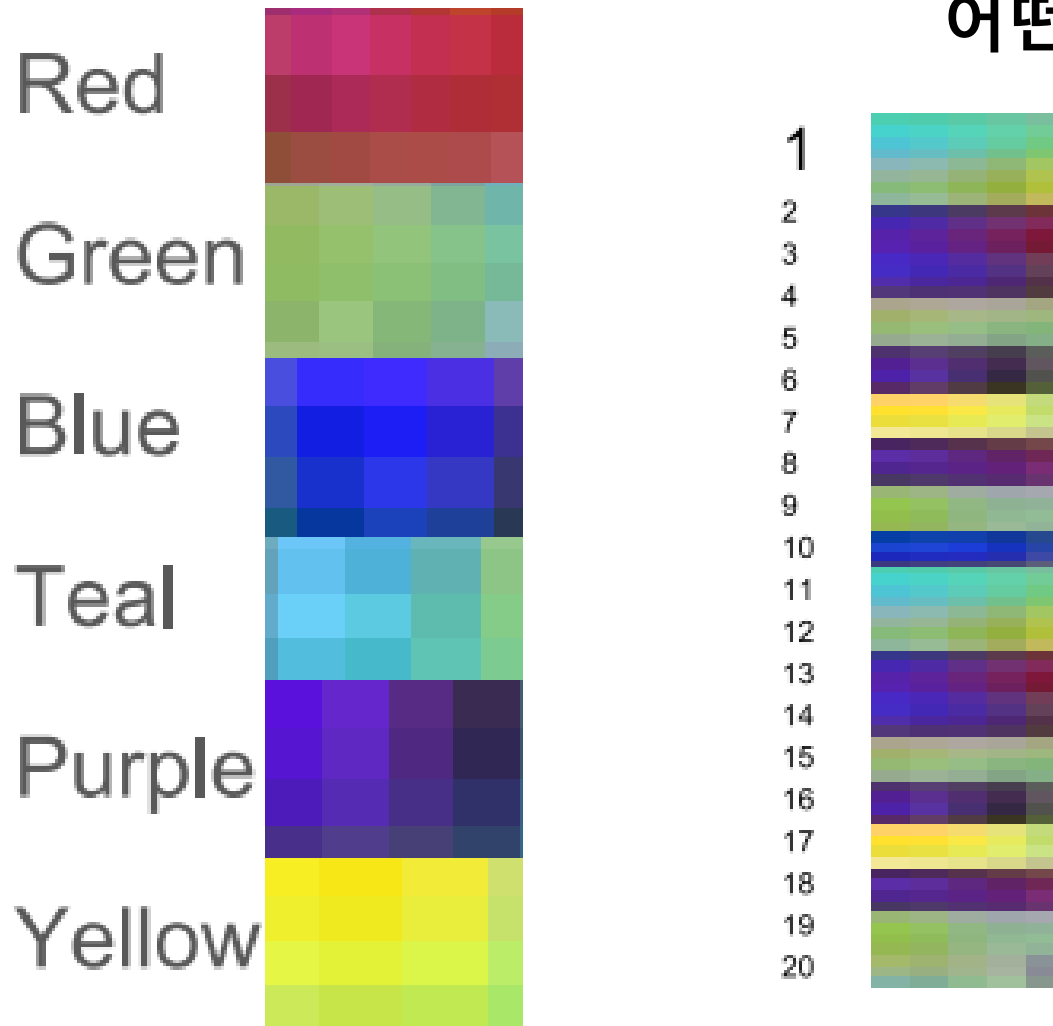
■ Bonferroni's Principle



어떤 아이폰 케이스가 가장 인기가 없을까요?

통계적인 한계

■ Bonferroni's Principle



어떤 아이폰 케이스가 가장 인기가 없을까요?

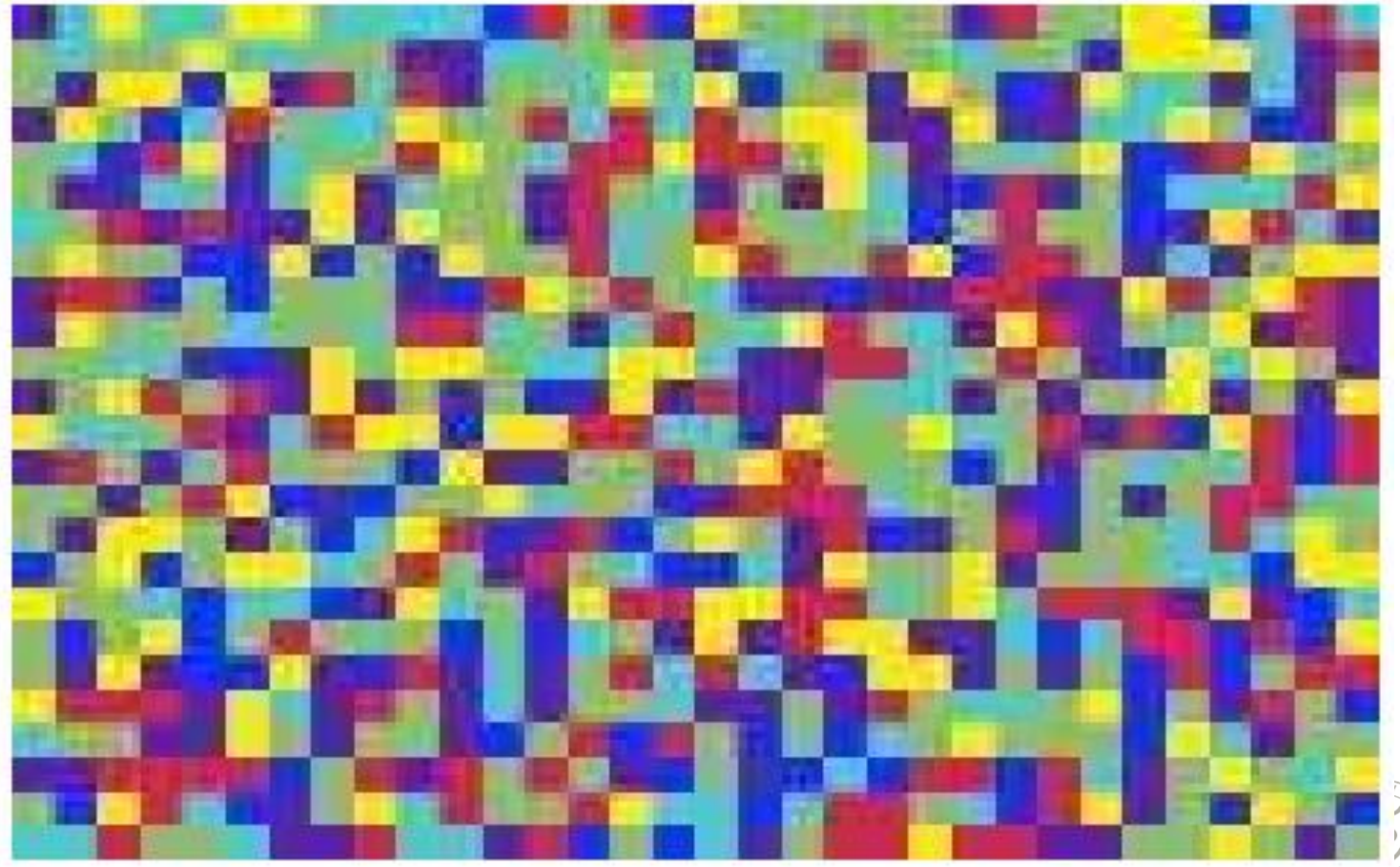
상위 20개의 판매량이 주어졌을때,
여러분은 어떠한 결론을 내릴 수 있을 가요?

통계적인 한계

■ Bonferroni's Principle



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

α / 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:

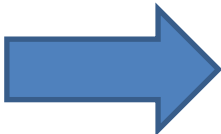
$$idf_i = \log_2\left(\frac{docs_*}{docs_i}\right) \propto \frac{1}{\frac{docs_i}{docs_*}}$$

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



	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

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

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	0.301	0	0	0	0	0.151	0	0
2	0	0.0417	0	0	0	0	0.0417	0.201
3	0	0.0417	0	0	0	0.100	0.0417	0
4	0	0.0209	0.100	0.100	0.100	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

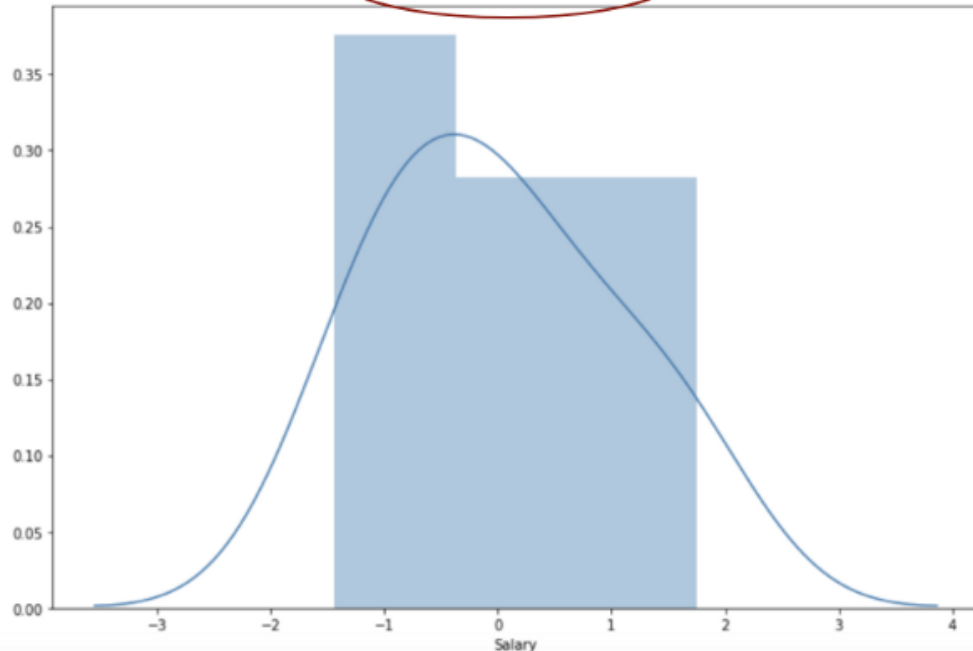
Example: Standardization vs Max-min normalization

Column: Salary

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

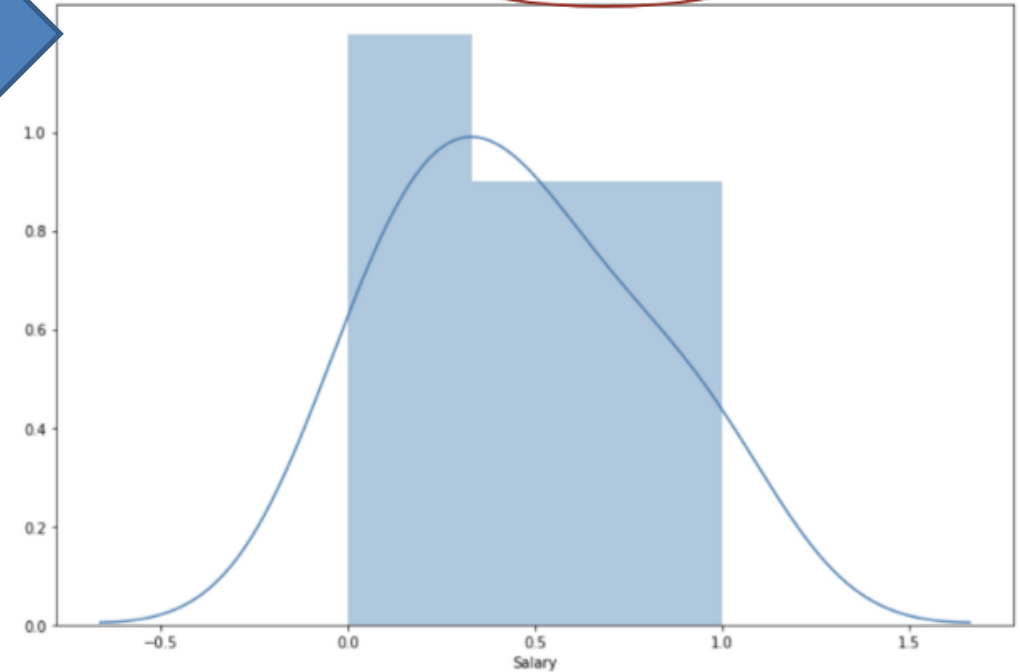
Standardisation

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

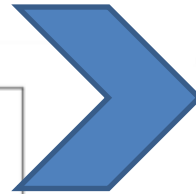
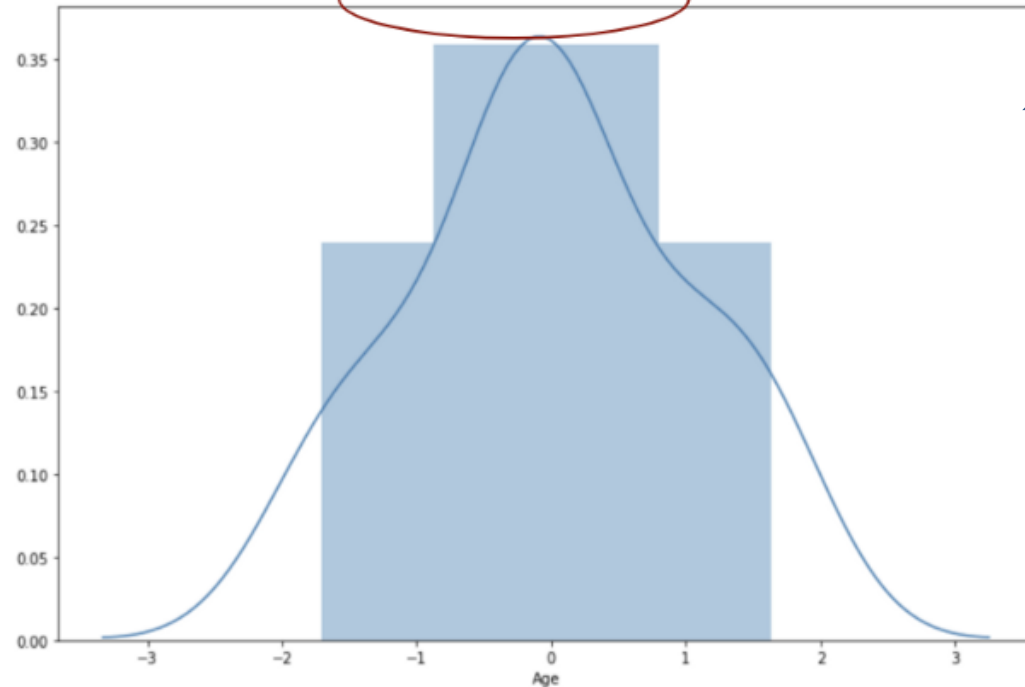
Standard Deviation (Age):
Max-Min Normalization (0.315) < Standardisation (1.05)

Standardisation

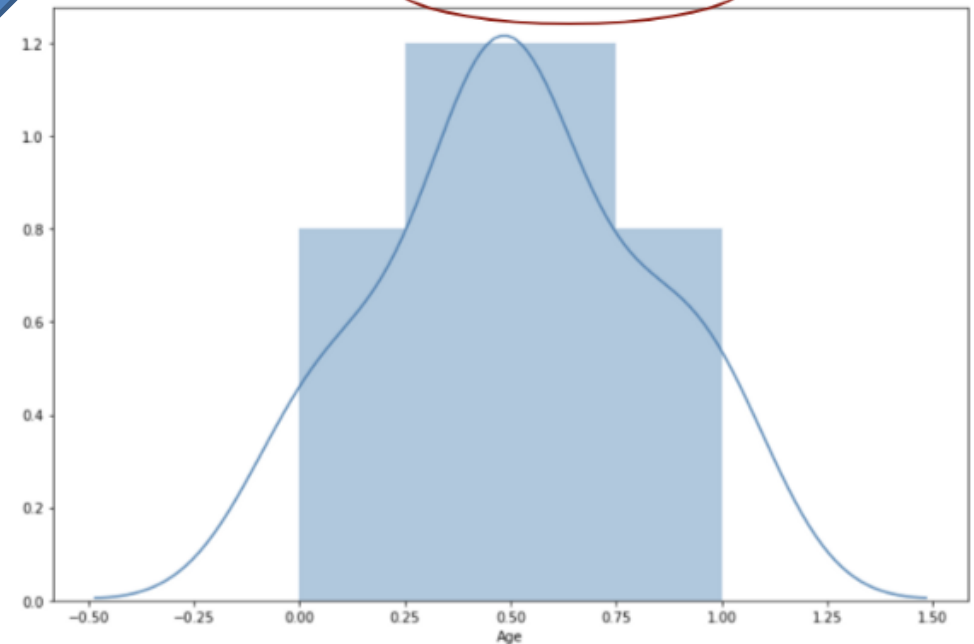
표준 편차가 작다는 것은
데이터가 밀집되어 있다

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)

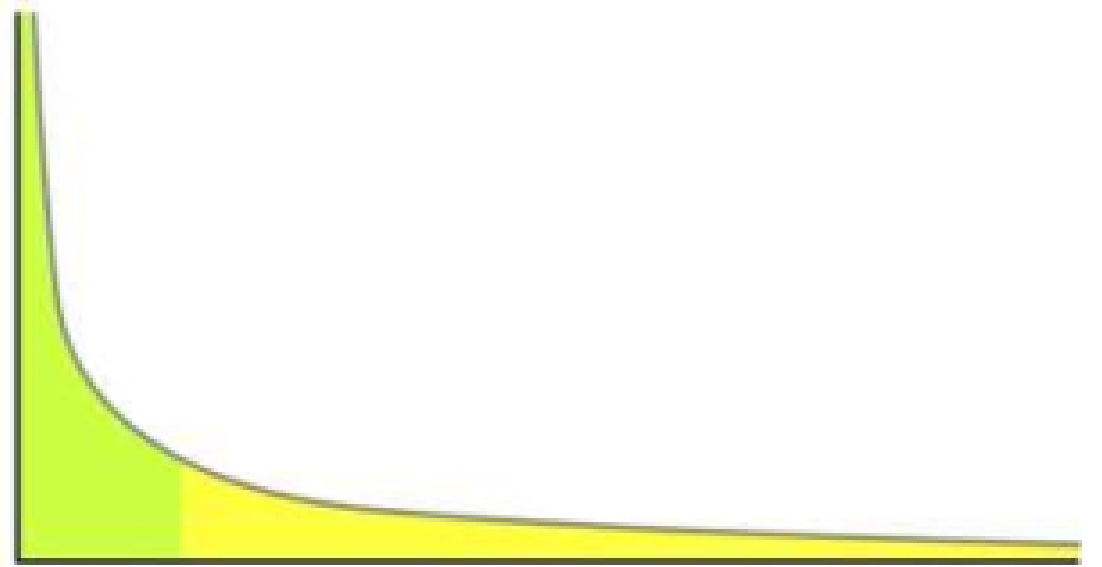
Power Law

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

raising to the natural log:

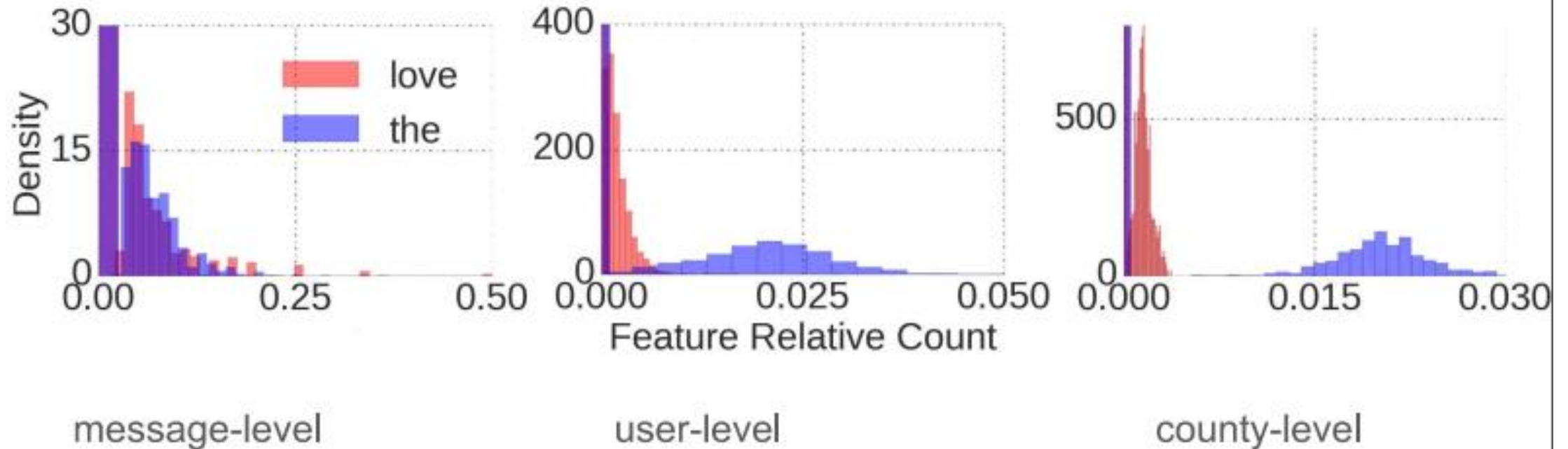
$$y = e^b e^{a \log x} = e^b x^a = cx^a$$

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 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

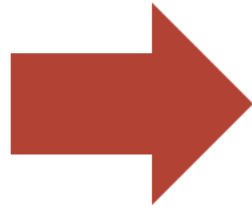
Database Indexes: Retrieve all records with a given *value*. (also review if unfamiliar / forgot)

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.

One hot encoding

Vocabulary:

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

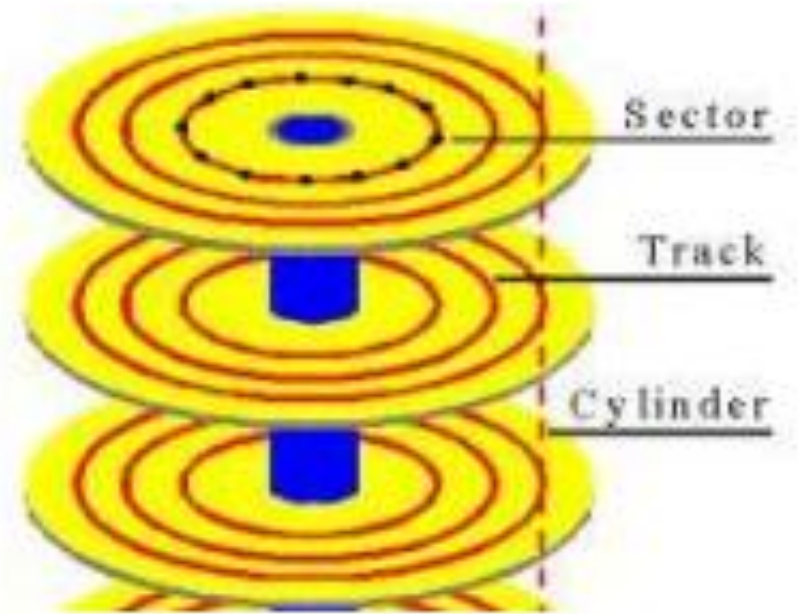


	1	2	3	4	5	6	7	8	9
man	1	0	0	0	0	0	0	0	0
woman	0	1	0	0	0	0	0	0	0
boy	0	0	1	0	0	0	0	0	0
girl	0	0	0	1	0	0	0	0	0
prince	0	0	0	0	1	0	0	0	0
princess	0	0	0	0	0	1	0	0	0
queen	0	0	0	0	0	0	1	0	0
king	0	0	0	0	0	0	0	1	0
monarch	0	0	0	0	0	0	0	0	1

Each word gets
a 1x9 vector
representation

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