

# 데이터 분석 들어가기

황도영



## 강사 약력

- (현) LG AI Research Material Intelligence Lab
- (전) AITRICS
- 고려대학교 컴퓨터전파통신공학과 석사 졸
- 고려대학교 통계학과 학사 졸
- Publication
  - Allele-conditional attention mechanism for HLA-peptide complex binding affinity prediction (Neurips 2022 MLSB)
  - Hit and lead discovery with explorative rl and fragment-based molecule generation (Neurips 2021)
  - A benchmark study on reliable molecular supervised learning via Bayesian learning (ICML 2020 UDL)
  - Comprehensive study on molecular supervised learning with graph neural networks (JCIM 2020)

# 데이터 분석 들어가기

---

1. 데이터 분석을 해야하는 이유
2. 데이터 분석 프로세스
3. 데이터 분석 툴(Google colab) 소개
4. 정규화와 데이터 스케일링

# 데이터 분석을 해야하는 이유

# 데이터 분석을 해야하는 이유

Top 10 NASDAQ: 2000 vs 2020

Company	MCap US\$bn	Company	MCap US\$bn
Microsoft	473	Apple	1,533
Cisco	451	Microsoft	1,489
Intel	387	Amazon	1,343
Oracle	202	Alphabet	866
Sun Micro	171	Facebook	520
Worldcom	130	Intel	243
Dell	110	Nvidia	225
Qualcomm	98	Adobe	205
Applied Mats	71	Netflix	195
Amgen	66	Paypal	201

Source: Refinitiv, THR.

- 2000년 VS 2020년 NASDAQ 시총 상위 기업을 보면  
'제조업' 기업 → '서비스' 기업
- FAANG 중 애플을 제외한 모든 기업은 제조업이 아닌  
서비스 기업
- 이들은 도대체 무엇을 판매하여서 돈을 벌었을까?  
→ "데이터를 이용한 가치"
- 아마존: '데이터 분석'을 이용한 예측배송 서비스  
구글, 페이스북: '데이터 분석'을 이용한 온라인 광고  
서비스  
넷플릭스: '데이터 분석'을 이용한 콘텐츠 추천  
서비스
- 데이터 분석을 기반한 사업 / 의사결정은  
이미 수많은 성공 사례를 보여주고 있습니다

# 데이터 분석을 해야하는 이유

---

So what is 데이터 분석?

- 데이터를 “정리, 변환, 조작, 검사”하여 “인사이트”를 만들어내는 작업
- 데이터 분석으로 무엇을 할 수 있는가? 왜 해야하는가?

수요 조사? 전략 수립? 성공 확률 측정?

→ 의사 결정의 판단 기준이 ‘주관적인 직감’에서 ‘객관적인 데이터’로

# 데이터 분석을 해야하는 이유

---

But...

- 데이터 분석을 꼭 해야될까?
  - 데이터가 없는 경우에는?
  - 데이터가 주어진 문제와 관련이 없는 경우에는?
  - 데이터가 주어진 문제를 해결하는데 결국 도움을 주지 못한다면?
- “주어진 데이터로 문제를 해결할 수 있을 지 없을지 가늠 하는 것도 데이터 분석이다”
  - 단순한 분석보다는 어떻게 문제를 해결할지에 대한 고민이 중요

# 데이터 분석 프로세스



# 데이터 분석 프로세스

---

1. 문제 정의
2. 데이터 수집
3. 데이터 전처리
4. 데이터 분석
5. 리포팅 / 피드백

## 1) 문제 정의

- 풀고자 하는 문제가 명확하지 않으면 데이터 분석은 무용지물이 됩니다
- a) 큰 문제(main objective)를 작은 단위의 문제들(sub objectives)로 나눈 후
- b) 각 작은 문제들에 대해서 여러 가설들을 세우고
- c) 데이터 분석을 통해 가설을 검증하고 결론을 도출하거나 피드백을 반영합니다

## 1) 문제 정의

- 궁극적으로 해결하고자 하는 문제(달성하고자 하는 목표)가 무엇인가요?
- 해당 문제를 일으키는 원인이 무엇인가요?
- 상황을 판단하는 지표나 기준이 무엇인가요?

## 1) 문제 정의

e.g. 인구 문제를 예시로 들면,

- 인구 감소(문제) ← 저출산 및 인구 유출(원인)
- 노동 인구 부족(문제) ← 인구 감소(원인)
- 내수시장 구매력 부족(문제) ← 인구 감소(원인)

→ '인구감소'라는 동일한 상황이 문제가 될 수도 있고, 문제의 원인이 될 수도 있습니다

→ 문제 정의에 따라서 풀어가는 방향이 완전히 바뀔 수 있습니다

→ 메타 인지 관점에서 문제 정의에 대한 충분한 고민이 필요합니다

e.g.) 인플레이션 ← 공급자 비용 증가 ← 노동시장 비용 증가 ← 노동 인구 부족 ← 인구 감소 ← 저출산 및 인구 유출 (원인?)

→ 해당 원인이 정말 문제에 대한 '결정적인' 원인일까요?

e.g.) 인플레이션의 결정적인 원인이 노동시장이 아닌 원자재 시장 비용 증가일지도?

## 2) 데이터 수집

- 검증해보고자 하는 가설을 해결해줄 데이터를 수집
- a) 가설 검증에 필요한 데이터가 존재하는가?  
→ 데이터가 가설 검증에 부적절하거나 없을 수도 있어요
- b) 어떤 종류의 데이터가 필요한가?  
→ 데이터는 산재되어있고 너무 많습니다  
데이터로부터 얻고자 하는 정보가 무엇인지 명확하게 해야 필요한 데이터만 모을 수 있습니다
- c) 얻고자 하는 데이터의 지표가 명확한가?  
→ 적절해 보이는 데이터라도 지표가 부적절하면 가설 검증 및 결론 도출시 오류를 범할 수 있습니다

## 3) 데이터 전처리

- 데이터 추출, 필터링, 그룹핑, 조인 등 (SQL 및 DB):  
데이터 분석을 위한 기본적인 테이블을 만드는 단계  
테이블과 칼럼의 명칭, 처리/집계 기준, 조인시 데이터 증식 방지
- 이상치 제거, 분포 변환, 표준화, 카테고리화, 차원 축소 등 (Python/R)  
수집한 데이터를 데이터 분석에 용이한 형태로 만드는 과정

## 4) 데이터 분석

- 탐색적 데이터 분석(EDA)
  - 그룹별 평균, 합 등 기술적 통계치 확인
  - 분포 확인
  - 변수 간 관계 및 영향력 파악
  - 데이터 시각화
- 모델링(머신러닝, 딥러닝)
  - Classification (categorical label)
  - Regression (numerical label)
  - 클러스터링 (비지도학습)

## 5) 리포팅 / 피드백

- 내용의 초점은 데이터 분석가가 아닌 상대방
  - 상대가 이해할 수 있는 언어 사용
  - 목적을 수시로 상기하고 재확인
- 적절한 시각화 방법 활용
  - 항목간 비교시 원 그래프는 지양하고 막대 그래프 위주, x,y축 및 단위 주의
  - 시계열은 라인이나 실선으로 표현
  - 분포는 히스토그램이나 박스플롯
  - 변수간 관계는 산점도



# 데이터 분석 툴(Google Colab) 소개

# 데이터 분석 툴(Google Colab) 소개

---

1. Colab 소개
2. Colab 시작하기
3. Colab 기본 사용법

# Colab 소개

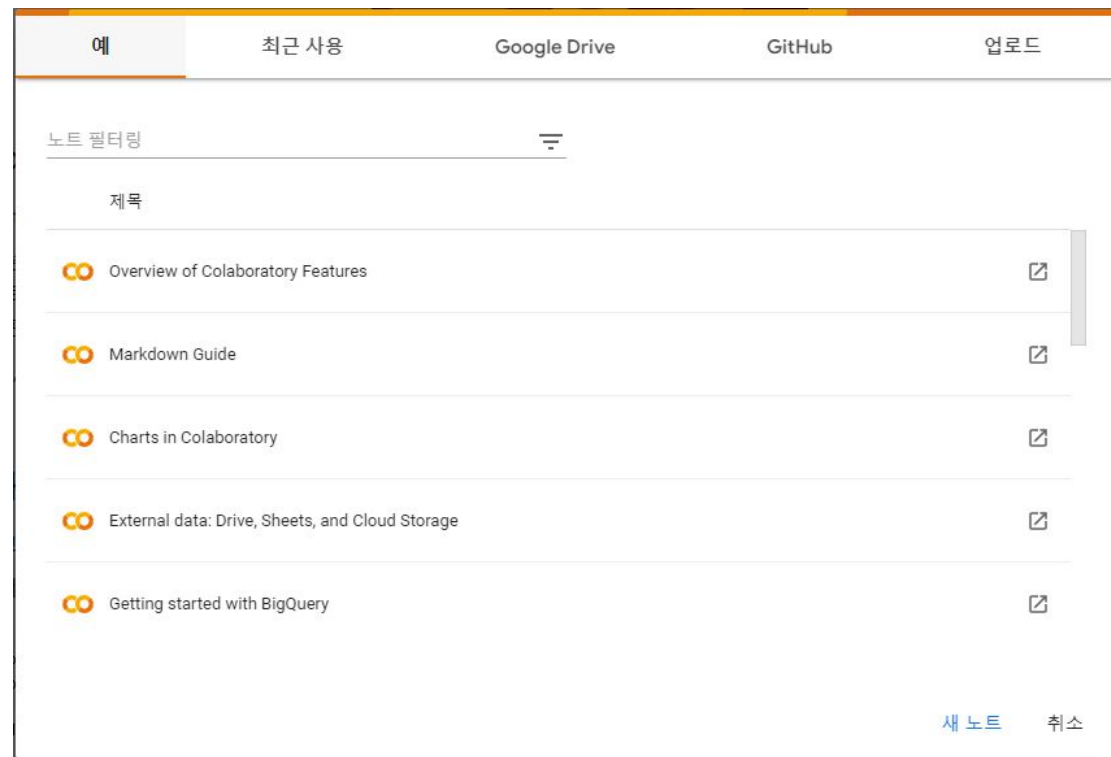
---

Colab은 클라우드 기반의 Jupyter 노트북 개발 환경입니다

- 웹 브라우저에서 텍스트와 프로그램 코드를 자유롭게 작성할 수 있는 일종의 온라인 텍스트 에디터입니다
- CPU와 램을 제공해주기 때문에 컴퓨터 성능과 상관없이 프로그램을 실행할 수 있습니다
- Colab 파일을 노트북(.ipynb)이라고 부르며, 코드를 입력하는 곳이 코드 셀이라고 합니다
- Colab에서 사용할 수 있는 프로그래밍 언어는 '파이썬'입니다

# Colab 시작하기

- 구글 계정만 존재하면 바로 사용이 가능합니다
- <https://colab.research.google.com/> 으로 접속하여 사용 가능합니다
- 새 노트 를 통해 새로운 파일(노트북) 생성이 가능합니다
- 기본적으로는 본인의 구글드라이브에 저장됩니다 (내 드라이브 > Colab Notebooks)




내 드라이브 > Colab Notebooks ▾

이름 ↑

Untitled0.ipynb

# Colab 기본 사용법

셀(Cell): 코드를 실행시키거나 설명 텍스트를 작성하는 블록입니다.

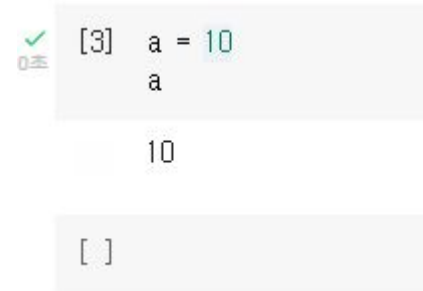
- 셀 위 혹은 아래에 마우스 커서를 갖다대면  와 같은 버튼이 생깁니다  
이 버튼을 눌러 새로운 셀을 생성할 수 있습니다

코드 셀: 작성된 코드를 실행하는 셀입니다

- Ctrl/Shift/Alt + Enter 를 통해서 실행 가능합니다  
Ctrl + Enter: 실행 후 키보드 커서가 셀 내 그대로  
Shift + Enter: 실행 후 키보드 커서가 다음 셀로  
Alt + Enter: 실행 후 새 코드셀 생성,  
키보드 커서가 다음 셀로

텍스트 셀: 설명에 필요한 텍스트 입력이 가능합니다

- 입력 문법은 markdown 문법을 따릅니다
- [https://colab.research.google.com/notebooks/markdown\\_guide.ipynb](https://colab.research.google.com/notebooks/markdown_guide.ipynb)



- 녹색 체크는 실행이 완료된 셀임을 의미하며, 실행 시간도 확인 가능합니다

[] 는 실행이 아직 되지 않은 코드 셀을 의미하며, [3]과 같이 숫자를 통해 셀들이 실행된 순서를 확인할 수 있습니다

- 코드 셀 마지막에 변수가 존재할 시 변수를 자동적으로 print 해줍니다.

코드 셀 중간의 변수를 확인하고 싶을때는 print(a) 와 같이 따로 print 작업을 입력해야 합니다

# 정규화와 데이터 스케일링

## 정규화가 필요한 이유

- 데이터에서 하나의 instance(sample)는 그것이 가진 여러 속성값들을 이용해서 표현이 가능합니다  
이 속성값들을 feature라고 합니다  
(e.g. 엑셀 테이블에서 row는 sample을, column은 sample들이 가지는 속성값을 의미)

- feature들간의 크기 및 단위가 들쭉날쭉 하거나 가지는 값의 범위가 크게 다른 경우,  
혹은 이상치(outlier)문제가 심각한 경우  
데이터 분석이 어려워지거나 머신러닝, 딥러닝 방법을 적용하기 어려워지는 경우가 있습니다

연봉	나이	보유 주택 수	연간 라면 소비수
3000천만~3억	20~60	0~2	12~100

- 특히, 머신 러닝 모델에 feature 값들을 input으로 사용하는 경우  
feature의 스케일이 들쭉날쭉하다면 모델이 데이터를 이상하게 해석할 우려가 있습니다  
위와 같은 경우, 스케일이 큰 '연봉'만을 중요한 feature로 모델이 해석하게 될 수도 있는 것입니다
- 이때 정규화와 스케일링을 통해 feature들이 가지는 값의 범위를 일정하게 맞춰주는 과정이 필요합니다

# Normalization(정규화)

---

- Normalization(정규화): 여러가지 값(feature)들이 가지는 범위의 차이를 왜곡하지 않으면서 범위를 맞추는 것
  1. Min-max normalization
  2. Z-score normalization(Standardization)
  3. Log scaling



# Min-max normalization

---

- 모든 feature값이 [0,1] 사이에 위치하도록 scaling하는 기법입니다
- 분모는 feature가 가질 수 있는 maximum값과 minimum값의 차이로 두고, 분자는 해당 feature 값과 minimum값의 차이로 둡니다

$$\frac{x - x_{\min}}{x_{\max} - x_{\min}}$$

- feature들간의 variance 패턴은 그대로 유지한 채로 feature가 scaling되게 됩니다
- 다르게 이야기 하면, 특정 feature만 variance가 매우 큰 경우, 특히 이상치(outlier)가 존재하는 경우  
여전히 feature간의 scaling이 데이터 분석에 적절하지 않을 수 있습니다

## Z-score normalization(standardization, 표준화)

---

- Feature 값들이  $\mu$ (평균)=0,  $\sigma$ (표준편차)=1 값을 가지는 정규분포를 따르도록 스케일링합니다

$$z = \frac{x - \mu}{\sigma}$$

즉, Feature값을 평균값으로 뺀 후 표준편차값으로 나눈 값을 사용합니다  
이때 z 값을 표준점수(z-score) 라고 합니다

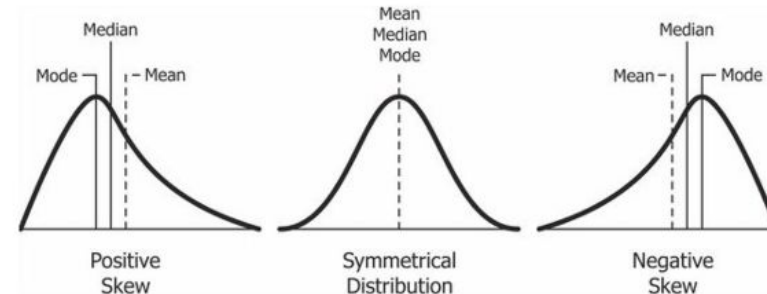
- Outlier 문제에 상대적으로 robust한 스케일링 방법입니다
- 다만, min-max normalization처럼 feature값이 가지는 최소값-최대값 범위가 정해지지 않는 단점이 있습니다
- 대부분의 머신러닝 기법(선형회귀, 로지스틱 회귀, SVM, Neural Networks)들을 활용하는 경우 input에 standardization를 적용해야하는 경우가 많습니다  
특히, Gradient descent를 활용한 학습 과정을 안정시켜주고 빠른 수렴을 가능케 합니다
- z-score가  $\pm 1.5\sigma$ ,  $\pm 2\sigma$ 를 벗어나는 경우 해당 데이터를 이상치로 간주하고 제거할 수 있습니다

# Log 스케일링

- Feature 값들이 exponential 한 분포(positive skewed)를 가지는 경우 feature 값들에 log 연산을 취하여 스케일링할 수 있습니다.  
e.g. 대한민국 국민 연봉의 분포

$$x' = \log(x)$$

- 비슷하게 square root 연산을 취하거나 반대의 분포를 가지는 경우 power / exponential 연산을 통해 스케일링해볼 수 있습니다.
- 다양한 스케일링을 통해 데이터가 좀더 정규분포에 가까워지도록 스케일링하며 outlier 문제에도 좀 더 적극적으로 대응 가능합니다.



```
power_scale_data = np.power(np.random.randn(300), 2)
df_power = pd.DataFrame(columns=['x'])
df_power['x'] = power_scale_data
```

```
df_power.hist(bins=30)
```

```
array([[<Axes: title='{center': 'x'}>]], dtype=object)
```

