딥러닝이란

01 인공지능(AI), 머신러닝, 딥러닝(Deep Learning)소개

첫번째 만나볼 내용?

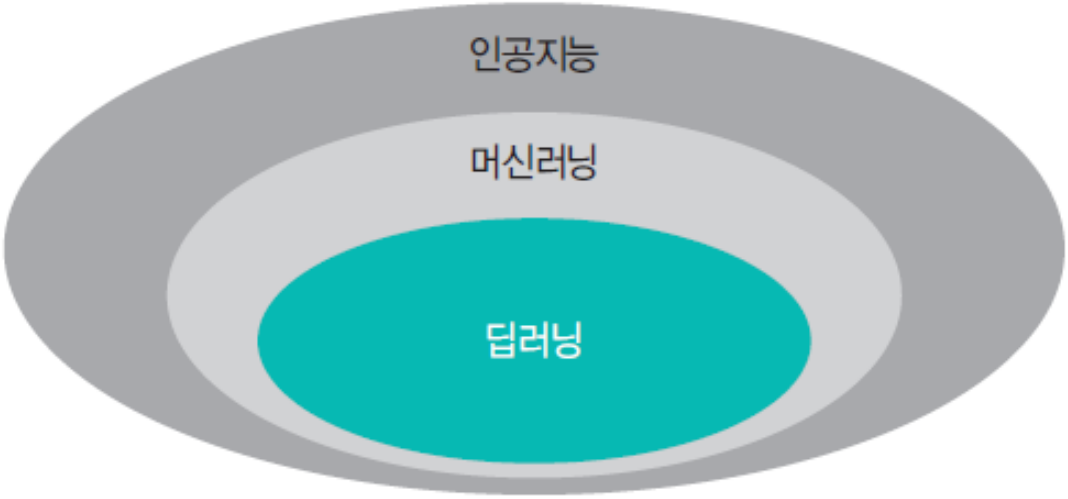
- 인공지능은 더 이상 특정 집단의 기술이 아님. 이미 우리 일상속에 침투해서 편리한 기능을 다양하게 제공 중
  - 스마트폰에 내장된 신경망 기술
  - 유튜브 동영상 추천, 스마트 티비, 스마트 홈 등

- 제공된 편리한 도구를 이용해서 우리도 직접 기능을 만들어보고, 적용해보고, 서비스를 제공할 수 있음

- 이를 위한 "기초 준비"를 시작
  - 케라스란?
  - 케라스 설치
  - 구글 코랩과 캐글 노트북(무료 클라우드)

시작하며

- 인공지능 ⇔ 머신러닝 ⇔ 딥러닝. 서로의 관계는?



[그림 1-1] 세 가지 용어의 관계

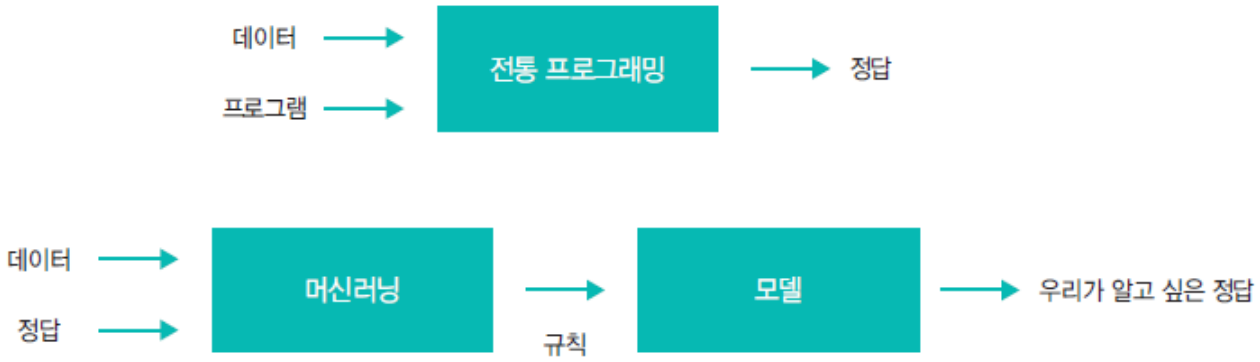
- 용어의 해석
  - **인공지능**: 학습, 인식, 추적 등 사람이 할 수 있는 작업과 할 수 없는 작업을 컴퓨터가 할 수 있도록 하는 것. 이 외에도 여러 가지 의미로 직접 정의할 수 있음
  - **머신러닝**: 기계가 학습하는 것
  - **딥러닝**: 깊은 신경망

머신 러닝 (ML; Machine Learning)

- 기계 학습 또는 머신 러닝은 경험을 통해 자동으로 개선하는 컴퓨터 알고리즘의 연구
- 컴퓨터가 학습할 수 있도록 하는 알고리즘과 기술을 개발하는 분야

- 머신 러닝은 **"우리가 어떤 것을 작동시키기 위해 '어떻게 명령할 지 알고 있는 것' 이상을 컴퓨터가 처리한 것이 가능한가?" 특정 작업을 수행하는 법을 스스로 학습할 수 있는가?** 질문에서 시작

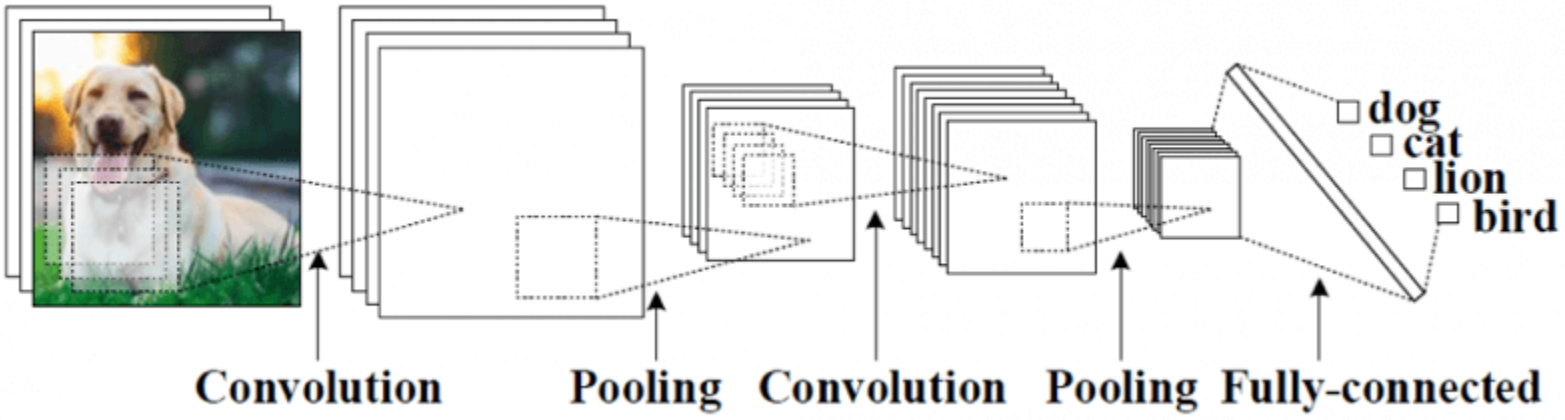
- 머신 러닝 시스템은 명시적으로 프로그램되는 것이 아니라 **훈련(training)**
- 1990년대 들어와서 각광 받기 시작 → 고성능 하드웨어와 대량의 데이터셋이 가능해지면서 AI에서 가장 인기 있고 성공적인 분야가 됨



[그림 1-2] 전통 프로그래밍과 머신러닝

딥 러닝 (Deep Learning)

- 머신 러닝 알고리즘 – 샘플과 기댓값이 주어졌을 때 데이터 처리 작업을 위한 실행 규칙을 찾는 것
  - 머신 러닝의 학습(Learning)이란 더 나은 표현을 찾는 자동화된 과정
- 딥 러닝 알고리즘 – 머신 러닝의 측정한 한 분야로서 연속된 층(layer)에서 점진적으로 의미 있는 표현을 배우는데 강점 → **데이터로부터 표현을 학습하는 새로운 방식**
  - 기본 층을 겹겹이 쌓아 올려 구성한 신경망이라는 모델을 사용하여 표현층을 학습
  - **딥 러닝의 학습이란** 주어진 입력을 정확한 타깃에 매핑하기 위해 신경망의 **모든 층에 있는 가중치 값을 찾는 것을 의미**



  - 원본 이미지와는 점점 더 다른 표현으로 이미지 변환 → 다단계 정보추출 작업
  - 데이터 표현을 학습하기 위한 다단계 처리 방식

딥러닝 (Deep Learning) 성과

- 딥러닝은 머신 러닝의 오래된 하위 분야이지만 2010년 초가 되어 유명해짐
- 사람에게서는 자연스럽고 직관적으로 보이지만 기계로는 오랫동안 해결하기 어려웠던 시각, 청각 같은 지각의 문제에서 괄목할 만한 성과를 이룸
  - 사람과 비슷한 수준의 이미지 분류
  - 사람과 비슷한 수준의 음성 인식
  - 사람과 비슷한 수준의 필기 인식
  - 향상된 기계 번역
  - 향상된 TTS(Text-To-Speech) 변환
  - 구글 나우(Now)와 아마존 알렉사(Alexa) 같은 디지털 비서
  - 사람과 비슷한 수준의 자율 주행 능력
  - 구글, 바이두 등에서 사용하는 향상된 광고 타기팅(targeting)
  - 자연어 질문에 대답하는 능력
  - 사람을 능가하는 바둑 실력
- 딥러닝이 과학, 소프트웨어 개발 등에서 사람을 보조하게되는 시대로 발전

AI 전망

- 1960년대 심볼릭 AI 기대
  - 1967년 이번 세대 안에 인공지능을 만드는 문제는 거의 해결
  - 3년 후에 1970년 3년 ~8년 이내에 평균적은 사람 수준의 일반 지능을 가진 기계 나올 것
  - **1970년** 성과 이루지 못해 연구자들과 정부 자의 투자가 줄었음 **첫번째 AI 겨울 (AI winter)**

- 1980년대 심볼릭 AI 새로운 버전의 전문자 시스템 큰 기업들 사이에 인기 끌기 시작
  - 전문가 시스템 구축을 위해 내부에 AI 부서 꾸리기 시작
  - 1985년 연간 10억 달러 이상 사용
  - **1990년** 초기에 시스템 유지 비용이 비싸고 확장하기 어려우며 제한된 범위를 가진 다는 것이 증명되어 **두번 째 AI 겨울**

- **2012년 이후** 현재 세 번째 싸이클을 목격하고 있을지 모름
  - BUT 아직까지 매우 낙관적인 단계
  - 기술적 측면을 잘 모르는 사람들에게 딥러닝이 할 수 있는 것과 할 수 없는 것에 대해 명확하게 이해시키는 것이 좋음

왜 딥러닝일까? 왜 지금일까?

- 컴퓨터 비전에 대한 딥러닝 두 가지 합성곱 신경망, 역전파는 이미 1989년 소개
- 시계열을 위한 딥러닝 기본인 LSTM(Long Short-Term Memory) 알고리즘 1997년 개발
- 1990년대와 2000년대에 걸친 진짜 병목은 데이터와 하드웨어 → 2012년 이후 딥러닝이 부상
- **하드웨어**
  - CPU 1990년-2010년 사이 거의 5,000배 빨라짐
  - NVIDIA, AMD 그래픽 성능 높이기 위해 GPU 개발하는데 수십억 달러 투자
  - 대형 CPU 클러스터가 소량의 GPU로 대체되기 시작 → CUDA를 사용한 신경망 구현
- **데이터셋과 벤치마크**
  - 무어의 법칙 저장 장치의 급격한 발전
  - 데이터 셋을 수집하고 배포 할 수 있는 인터넷 성장이 시장의 판도를 바꿈
- **알고리즘 향상**
  - 신경망의 층에 더 잘 맞는 활성화 함수
  - 가중치 초기화, 최적화 방법으로 10개 이상의 층을 가진 모델 훈련 가능
- **새로운 투자 바람**
- 딥러닝의 대중화 → **"딥러닝" 마케팅 용어 → AI 분야가 많은 투자에 큰 영향**

* 무어의 법칙(Moore's law)은 반도체 집적회로의 성능이 24개월마다 2배로 증가한다는 법칙

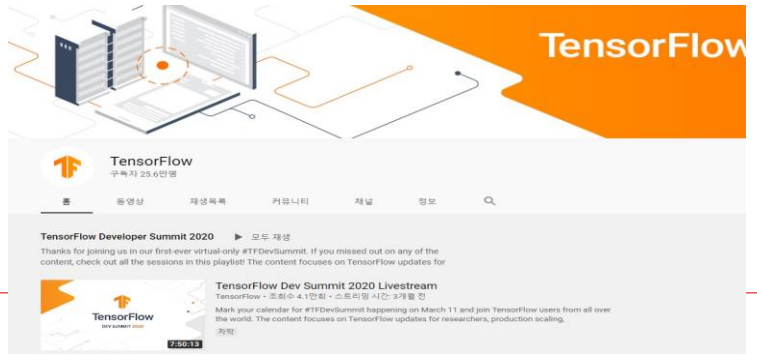
텐서플로우(TensorFlow)

- 가장 많은 사용자를 보유하고 있는 **머신러닝 오픈소스 플랫폼**

- 텐서플로우 2.x 버전으로 넘어오면서,
  - 사용자의 접근성과 편의를 고려한 설계
  - 설계부터 배포까지, 전과정을 고려한 기능 제공
  - 서비스뿐만 아니라 연구자도 고려, But 연구는 아직 PyTorch가...

- 원래 케라스와 텐서플로우는 다른 도구로 인식되어왔음
  - 케라스의 사용성을 인정받으면서 텐서플로우의 고수준 API로 채택
  - **입문자에게 케라스 사용을 권장**

- 그 외 텐서플로우의 다양한 기능 업그레이드는 텐서플로우 유튜브 채널에 자세히, 쉽게 설명하고 있음
  - 텐서플로우 내부는 어떻게 동작하는지
  - 어떤 변화가 일어나는지는 Dev Summit 2020 등을 참조



공부 방법?

- 다른 학문 분야도 마찬가지지만, 특히 우리가 공부할 이 분야는 책 한권으로 절대 해결되지 않으며, 초급 수준을 벗어나기 어려움. 따라서, **다양한 문서를 곁에두고 공부**하기를 강력하게 추천
  - 백견불여일타 딥러닝 입문서
  - 수학적 이론 및 배경을 다루는 이론서
  - 텐서플로우 및 케라스 공식 API

- **AI 민주화**의 의미를 떠올리면서 공유와 소통을 실천

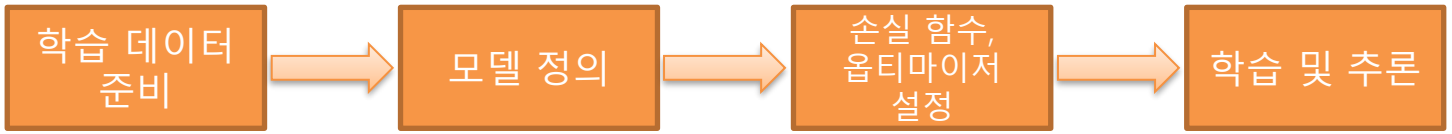
- 기술 습득을 위한 문제 해결이 아닌 **문제 해결을 위한 기술 습득**에 초점을 맞추어 공부할 것
  - 문제 해결에 초점을 맞춘다는 것은 매우 어려운 방법이므로 꾸준한 노력을 요구

케라스란?

- 누구나 쉽게 사용하고 접할 수 있도록 파이썬으로 설계된 머신러닝 라이브러리이며, 구글 엔지니어 **프랑소와 숄레**가 창조
  - 더 많은 이야기: https://brunch.co.kr/@hvnpoet/93

- 케라스 특징
  - 단순성 및 간결성
    - 케라스의 표현은 짧고 간결. Input, Model, Layer 등과 같이 이름만으로 기능을 추측
  - 유연성
    - 텐서플로우와의 호환 → 향상된 성능
  - 모듈화
    - 독립적으로 문제 정의 가능하며, 적합한 모델 구성 가능
  - 파이썬 기반

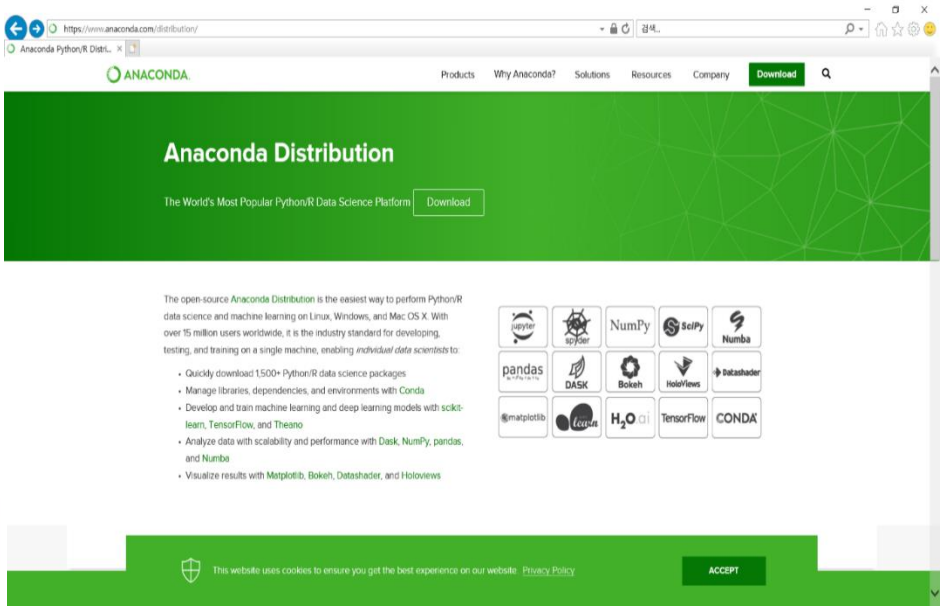
- 케라스의 핵심은 **"모델(Model)"**이다.
  - 단순하면서도 매우 강력함



케라스 학습 과정

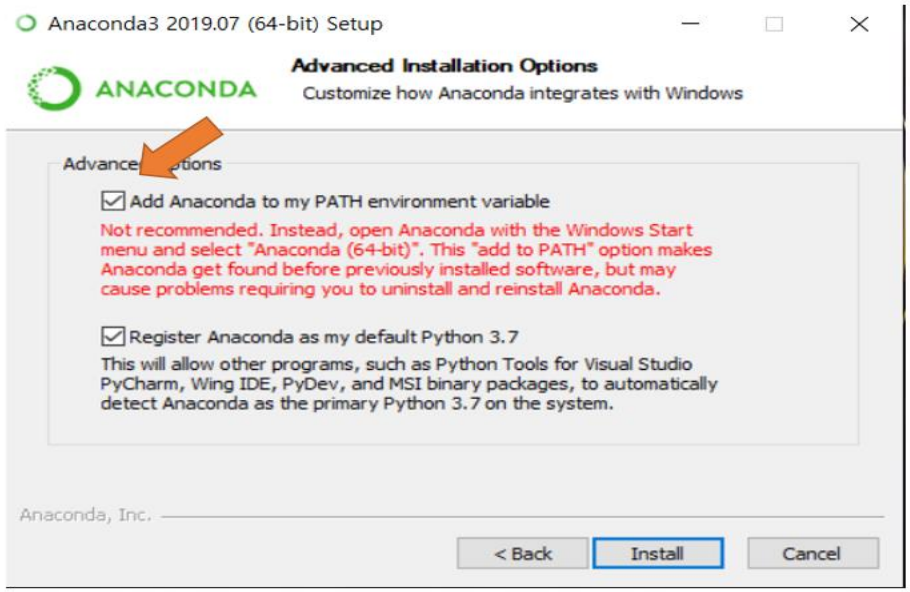
케라스 준비하기

- 케라스 사용을 위해 설치할 것
  - 아나콘다(Anaconda)
  - 텐서플로우 CPU
  - **텐서플로우 GPU**
    - CUDA(텐서플로우 버전에 맞게)
    - Cudnn(CUDA 버전에 맞게)



[그림 1-3] Anaconda 설치

https://www.anaconda.com/products/individual



[그림 1-4] Anaconda PATH 자동 생성

무료 클라우드 사용하기

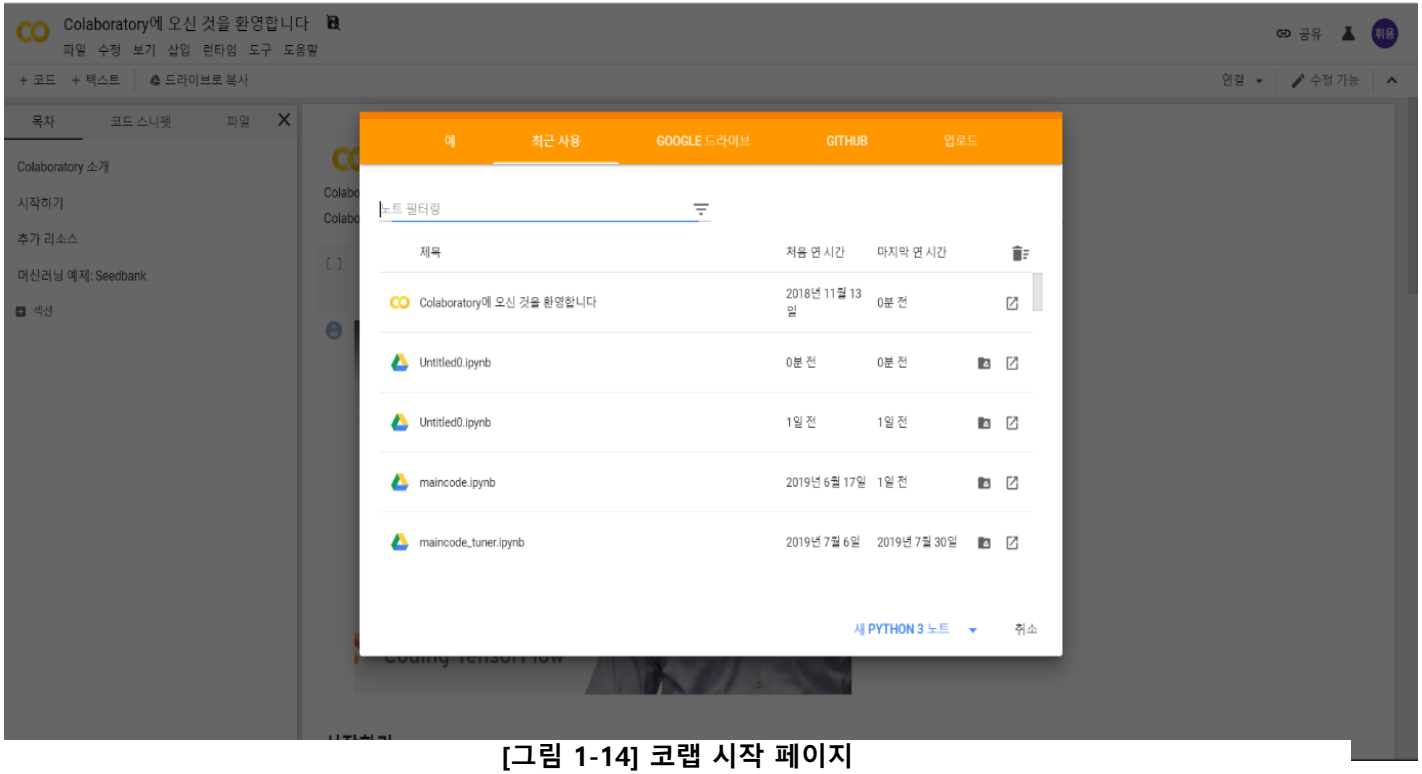
- 딥러닝은 개발환경이 좋을수록 성능이 좋아질 가능성이 높음
  - **돈이 없으면 실험도 못해**본다는 말은 과언이 아님
  - 많은 사람이 신경망 모델 구축을 위해서는 GPU가 필수라고 함

- GPU가 없으면?
  - **구글 코랩(Google Colaboratory)**
  - **캐글 노트북(Kaggle Notebooks)**

- 무료로 GPU를 사용하게 해주기 때문에 GPU가 없다면,
  위 두가지를 적극 사용하기를 권장함

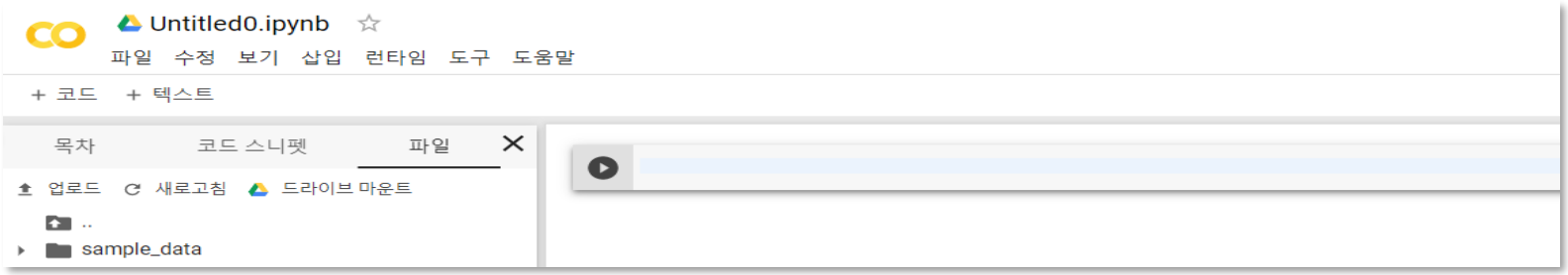
구글 코랩

- 구글 코랩은
  - GPU 무료 사용(**하루 12시간**), 계정 당 2개의 GPU 할당
  - 구글 드라이브와 연동 가능
  - http://colab.research.google.com

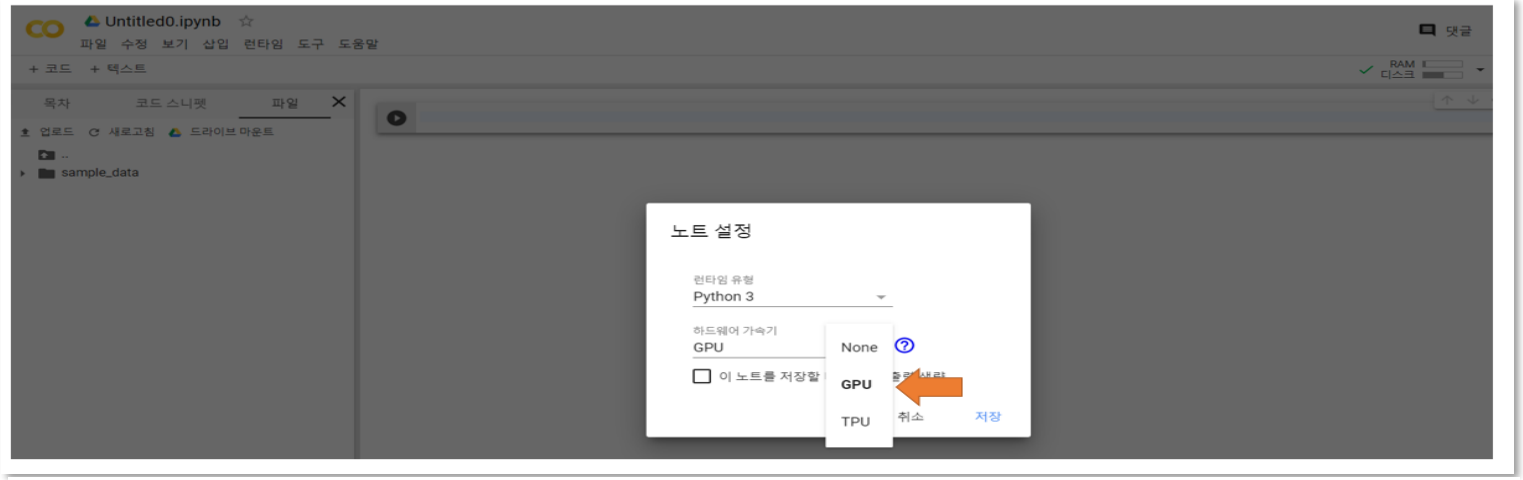


[그림 1-14] 코랩 시작 페이지

구글 코랩



[그림 1-15] Google Colaboratory

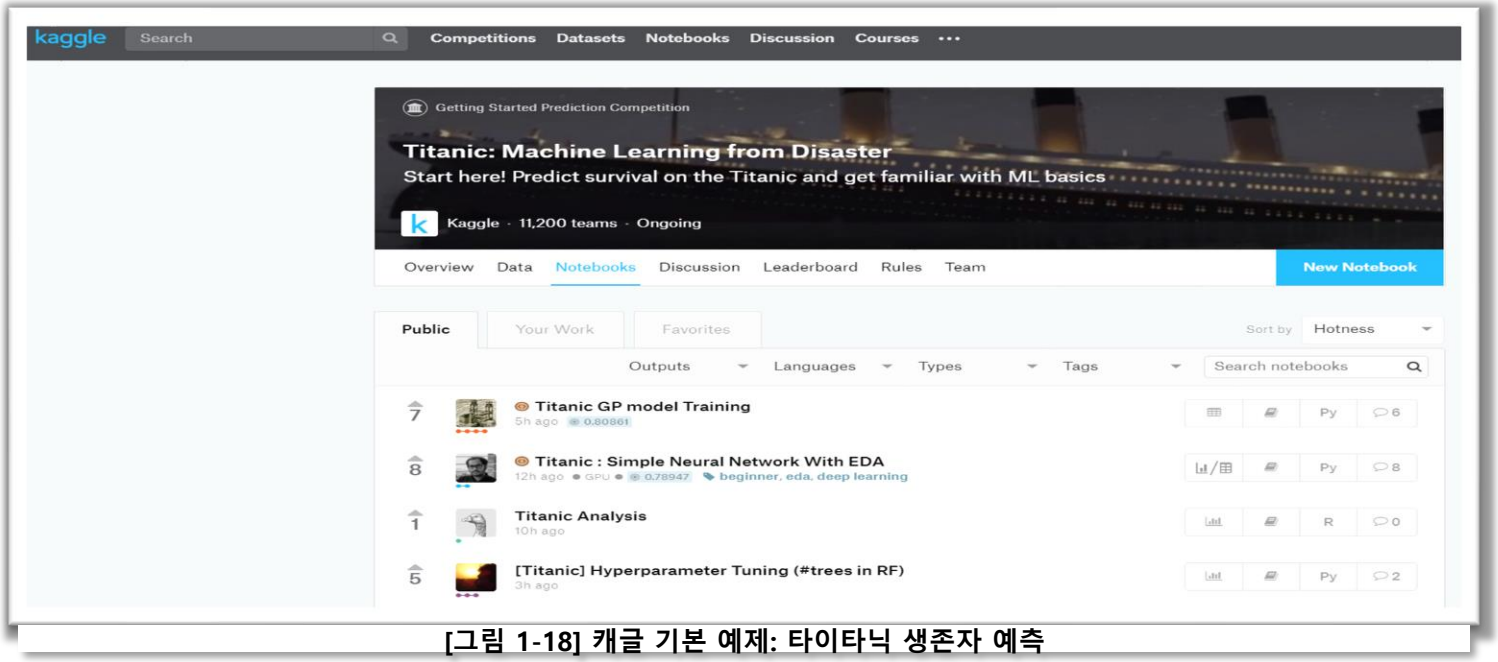


[그림 1-16] 코랩에서의 GPU 설정하기

캐글 노트북

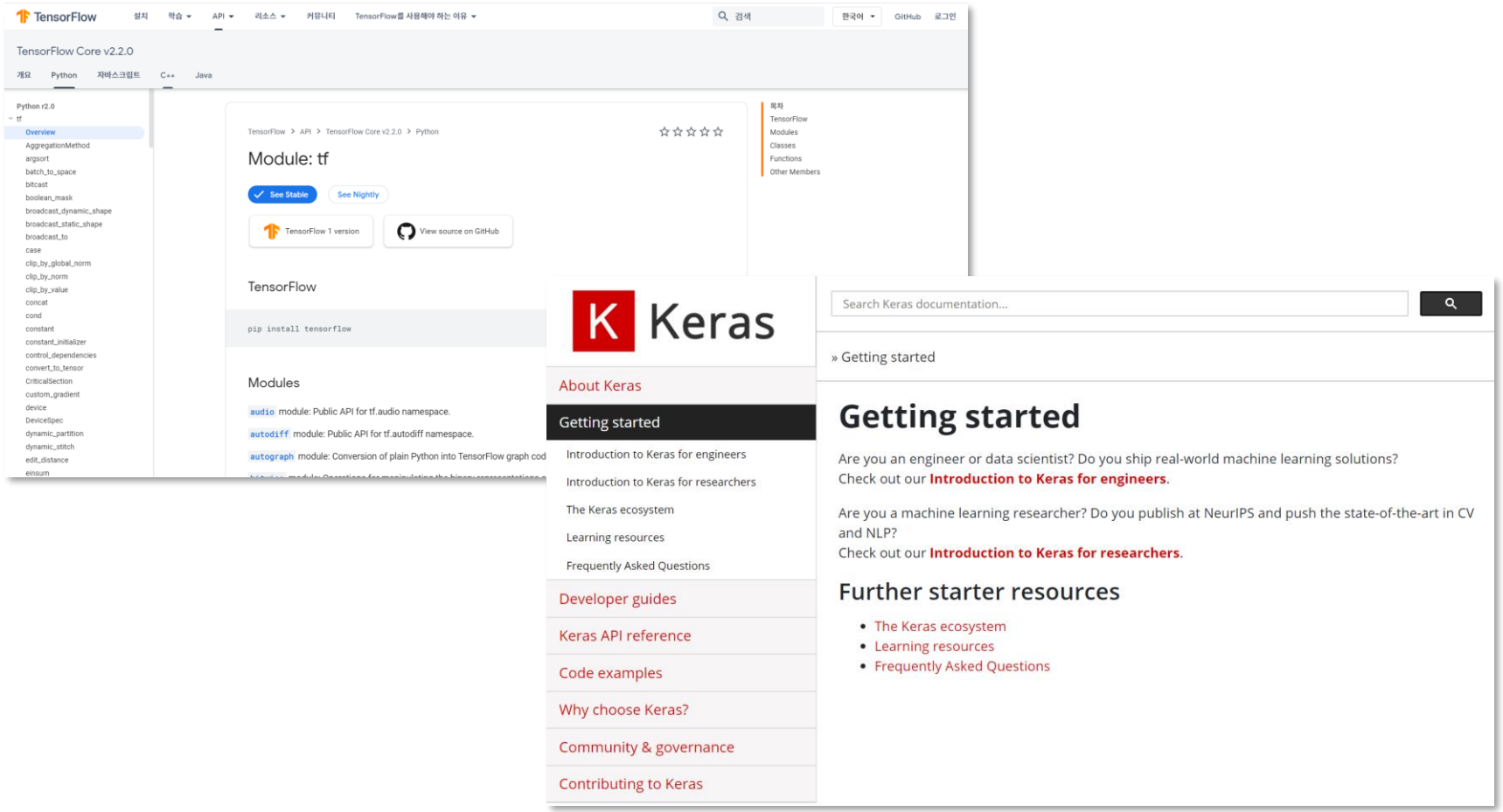
- 캐글(Kaggle): 데이터 과학자들을 위한 놀이터
  - **상금이 걸린 대회**가 **다양한 데이터**를 통해 개최
  - 타이타닉 생존자 예측, 보스턴 집값 예측 등 **다양한 튜토리얼 예제** 마련
  - 캐글 노트북을 통해 **고수들의 분석 기술을 무료로 살펴볼 수 있음**

- 20GB의 데이터 저장 공간과 무료 GPU를 제공



[그림 1-18] 캐글 기본 예제: 타이타닉 생존자 예측

API 문서 활용하기



텐서플로우 및 케라스 공식 문서

정리해봅시다

1. 인공지능 ← 머신러닝 ← 딥러닝
2. 텐서플로우의 케라스는 매우 단순하며 강력합니다.
3. 신경망 모델 사용을 위해서 GPU 보유 여부는 매우 중요합니다.
4. GPU를 보유하고 있지 않더라도 우리에게는 **구글 코랩과 캐글 노트북**이 존재합니다.
5. 무료 GPU는 사용 시간이 제한되어 있으니, 주의해서 사용해야 합니다.
6. 캐글의 최대 강점인 **캐글 노트북은 전문가들의 노하우를 볼 수 있는 최고의 기술서**입니다.
7. 공식 홈페이지를 보는 습관은 실력 향상을 위한 좋은 방법입니다.

머신러닝 프로세스

01 머신러닝 프로세스
02 다양한 용어 살펴보기

두번째 만나볼 내용?

- 스포츠 경기에서 이기려면?
  - 선수 능력 파악 및 적재적소 배치와 경기 흐름을 파악
  - 우리 팀 선수뿐만 아니라 적 팀의 선수 능력을 정확하게 파악해야 함

- 동일하게, 신경망 개념도 중요하지만 그 외의 다양한 요소를 고민할 수 있어야 함

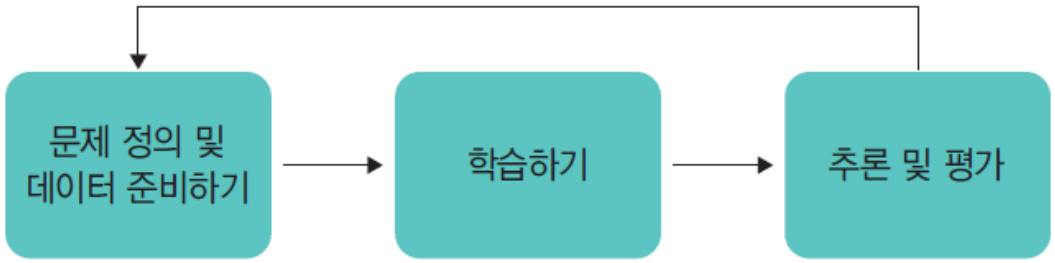
- 이번 장에에서는 다음 내용을 다뤄봅니다.
  - **머신러닝 프로세스**
  - 다양한 **용어** 살펴보기
    - 데이터 준비하기
    - 학습하기
    - 평가하기
  - **데이터셋 및 커뮤니티** 살펴보기

용어를 자세히 아는 것도 중요하지만,
많은 용어를 아는 것도 굉장히 중요합니다.

머신러닝 프로세스 간략히 살펴보기

- ML Process
  - 문제 정의 및 데이터 준비하기 → 학습하기 → 추론 및 평가 → …

데이터 준비로
학습하기로
etc.



[그림 2-1] 머신러닝 프로세스(간략)

- 잘 구축된 프로세스가 결과적으로 성능이 우수한 모델을 만듦
  - 어떻게? **'잘'** 해야함.
  - 직접 문제를 정의해보고, 해결해보는 과정을 **경험**하자!

- 이 책에서 다루는 대부분의 프로세스는 위 그림의 과정을 따름

문제 정의 및 데이터 준비하기

- 문제를 어떻게 정의하느냐에 따라 우리가 가야할 길이 달라짐
  - 올바른 방향으로 나아가기 위해 탐색적 데이터 분석(EDA)를 시작
  - 가장 중요한 것은 **명확한 문제 정의**

    무엇이 문제인가?
    난 문제를 어떻게 풀고 싶은가?
    사용자는 누구인가?

  - 숫자? 문자? 이미지? ...
  - 이진 분류? 다중 분류? 회귀? 생성?

- 모델을 선택하기 전에, **데이터를 아~주 자세히 들여다보자!**
  - 변형해야 할까, 어떤 전처리 방법을 선택할까
  - 데이터가 속한 도메인에는 어떤 처리 방법이 존재할까

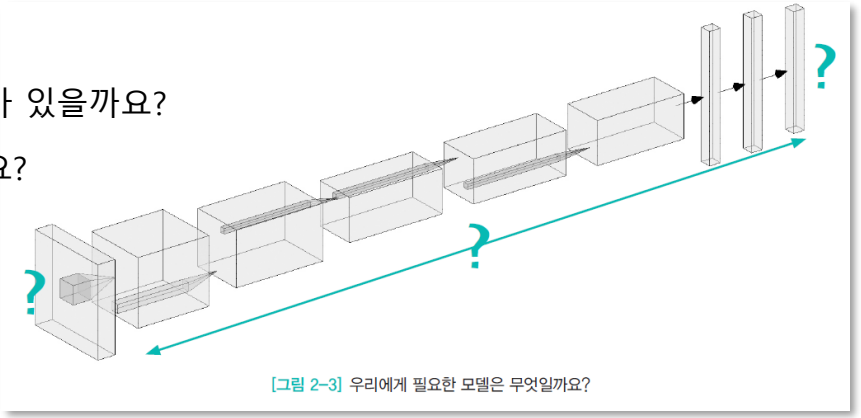
  캐글을 활용하자!

- 데이터에 관한 고민은 끊임없이 반복될 것

[그림 2-2] 내가 데이터를 만들 때

학습하기

- 모델 선택에는 어떤 질문이 필요할까요
  - 선택한 모델이 주로 어떤 데이터에 적용되었나요? 사례가 있을까요?
  - 모델이 얼마나 깊어야 하나요? 어떤 환경에서 사용될까요?
  - 옵티마이저는요? 손실 함수는요?
  - 실험 환경에 적합한 모델인가요?



[그림 2-3] 우리에게 필요한 모델은 무엇일까요?

- 깊은 고민도 좋지만, **SOTA(state-of-the-art;최고의 성능) 모델**을 활용해보자
  - 이미지? VGG, ResNet, Inception etc.
  - 자연어 처리? ELMO, BERT, GPT etc.

- 모델을 선택했다면, **하이퍼파라미터를 조정**하는 단계는 필수!
  - 배치 크기, 학습률, 층 파라미터 등

- 모델 선택은 **내부 요소와 외부 요소를 복합적으로 고려**해야 함
  - 하이퍼파라미터와 모델 크기(실험 환경에 적합한지)

추론 및 평가

- 추론(Inference)
  - 학습된 모델로부터 정답이 없는 데이터에 대해 정답을 만드는 행위

- 추론을 통해 얻은 정답을 어떻게 평가? 햄버거 가게를 생각해보자.

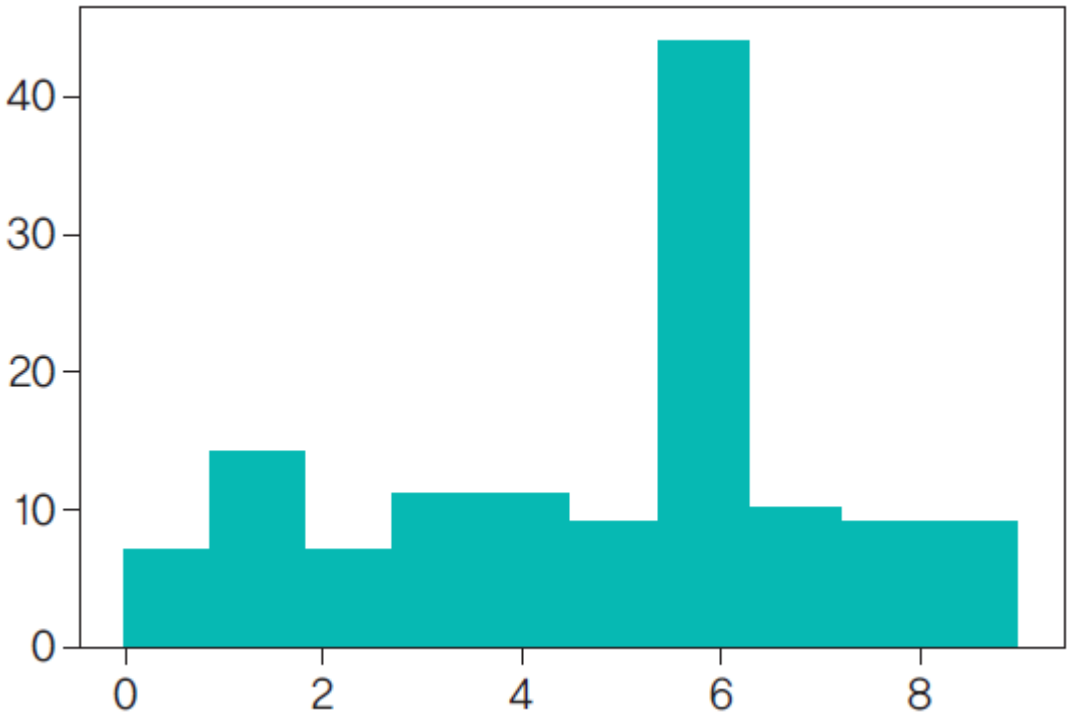
- 역할: 매니저(나), 직원 A, B
- 오늘 햄버거 만드는 실력을 보고 A, B 중 한 명을 승진시키겠다.

- A 직원: 10개를 다 만들었지만, 5개만 완벽
- B 직원: 다 만들지 못했지만, 6개가 완벽

누굴 선택?

용어 살펴보기

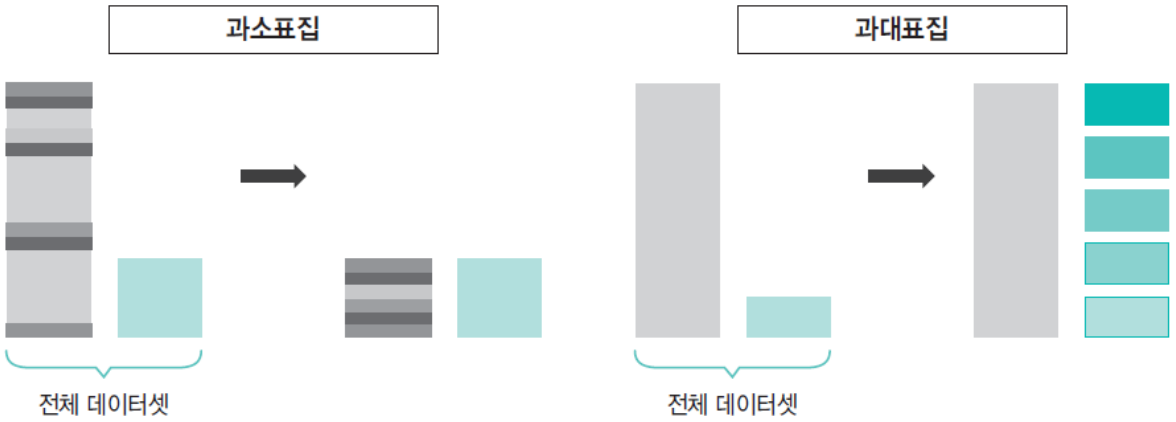
- 어떤 용어가 떠오르나요?



용어: 데이터 준비하기

- 클래스 불균형(Class Imbalance)
  - 클래스가 불균형하게 분포되어 있음
  - 은행 거래 사기, 희귀 질병, 기계 불량음 등의 사례
  - 이상 탐지(Anomaly Detection)

- 과소표집(UnderSampling)과 과대표집(OverSampling)
  - 과소표집은 다른 클래스에 비해 상대적으로 많이 나타나 있는 클래스의 개수를 줄이는 것
  - 과대표집은 개수가 적은 클래스를 복제하는 것 (SMOTE 등 기법들을 활용)



https://john-analyst.medium.com/smote%EB%A1%9C-%EB%8D%B0%EC%9D%B4%ED%84%B0-%EB%B6%88%EA%B7%A0%ED%98%95-%ED%95%B4%EA%B2%B0%ED%95%98%EA%B8%B0-5ab674ef0b32

용어: 데이터 준비하기

- 회귀(Regression)
  - 여러 개의 특징을 통해 연속적인 숫자로 이루어진 정답을 예측
  - 햄버거 가격, 영화 관객 수, 축구 선수 연봉, 주식 가격 등
  - 0과 1을 예측하는 로지스틱 회귀(Logistic Regression)

빵(g)	고기(g)	치즈(g)	가격(₩)
300	500	100	5500
200	1000	50	10500
500	500	500	8000
150	300	150	4000
200	200	200	?

[그림 2-6] 회귀의 예 : 햄버거 가격 예측

- 분류(Classification)
  - 미리 정의된 여러 클래스 중 하나를 예측
  - 햄버거 종류, 숫자 판별, 얼굴인식 또는 종류 구분 등
  - 이진분류(Binary Classification)
  - 다중분류(Multi-class Classification)
  - 다중 레이블 분류(Multi-label Classification)

• 햄버거를 선택한다면?

햄버거	음료
1	0

• 불고기버거를 선택한다면?

불고기버거	치킨버거	치즈버거
1	0	0

• 불고기버거 세트를 선택했다면?

불고기버거	치킨버거	치즈버거	콜라	환타	사이다
1	0	0	1	0	0

*선택: 1 미선택: 0

[그림 2-7] 이진 분류, 다중 분류, 다중 레이블 분류의 예

용어: 데이터 준비하기

- 원핫 인코딩(One-Hot Encoding)
  - 하나의 클래스만 1이고 나머지 클래스는 전부 0인 인코딩 (자연어;NLP 처리 분야에서 많이 언급)
  - 불고기버거: [1, 0, 0]
  - 치킨버거: [0, 1, 0]
  - 치즈버거: [0, 0, 1]

- 교차 검증(Cross-Validation)
  - 모델의 타당성을 검증 (과대적합 방지를 위해 사용)
    - 학습 데이터: 모델 학습에 사용
    - 검증 데이터: 모델의 검증을 위해 사용, 주로 학습 도중에 사용
    - 테스트 데이터: 모델의 최종 성능 평가에 사용
  - **테스트 데이터는 최종 평가 이전에는 절대로 사용하면 안됨**

[그림 2-8] 홀드아웃 교차 검증, 데이터 분리

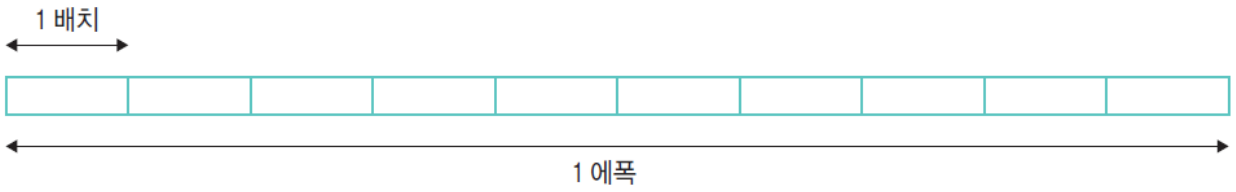
용어: 학습하기

- 하이퍼파라미터(Hyperparameter)
  - **경험에 의해 결정되는 요소**
  - 학습률, 배치 크기, 에폭 등
  - 적합한 값을 찾기 위해 반복적인 실험과 많은 시간 투자가 필수 → 하이퍼파라미터 튜닝

- 배치와 배치크기(Batch&Batch Size)
  - 데이터를 한 개만 사용하기엔 정확한 정답을 찾는 데 방해가 될 수 있고, 전부를 사용하기엔 시간이 너무 오래 걸리기 때문에 배치를 사용
  - 1,000개 데이터에서 배치가 10이면, 각 배치는 100개 데이터를 보유
  - 배치 크기는 기존 사례(주로 논문)를 참고하거나 주로 2 제곱수를 사용(16, 32, …)

용어: 학습하기

- 에폭과 스텝(Epoch&Step)
  - 에폭: 전체 데이터를 사용하여 학습하는 횟수
    - 전체 데이터를 10회 반복? 10 에폭
  - 스텝: 모델이 가진 가중치를 1회 업데이트하는 것



[그림 2-12] 배치와 에폭

햄버거 100개가 있고, 이를 전체 데이터라고 하겠습니다.
또한, 햄버거는 100개의 단위로 다시 제공받을 수 있으며, **100개를 전부 먹으면 1 에폭**이라고 정의하겠습니다.
햄버거 1,000개를 먹기 위해서는 **10회 반복(10 에폭)**해야 합니다. 이 과정에서 한번에 100개를 먹는 것은 너무 많은 것 같습니다. 20개씩 나눠서 먹겠습니다. 여기서 **배치는 5(100/20)**이며, **배치 크기는 20**이라고 할 수 있습니다.

용어: 학습하기

- 지도 학습(Supervised Learning)
  - **학습 데이터에 정답이 포함된 것**
  - 모델에게 햄버거 사진을 보여주면서 햄버거라고 알려줌
  - 대표적으로 회귀와 분류가 해당됨

- 비지도 학습(UnSupervised Learning)
  - **학습 데이터에 정답이 포함되어 있지 않은 것**
  - 모델에게 햄버거를 종류별로 여러 개 주고 같은 종류끼리 묶어보라고 하는 것 → 클러스터링(Clustering)

- 햄버거 사진을 주고, 모델에게 다시 햄버거 사진을 그려보라고 하는 것
  - 생성 모델(Generative Model)

- 에이전트가 주어진 환경에 대해 어떠한 행동을 결정하고, 이를 통해 얻는 보상으로 학습하는 것
  - 강화 학습(Reinforcement Learning)

용어: 학습하기

- 과대적합(Overfitting)
  - 모델이 학습 데이터에서는 좋은 성능을 보이지만, **새로운 데이터에 대해서는 좋은 성능을 보이지 못하는 결과**
  - 모델은 학습 데이터를 단순히 외웠다고 표현할 수 있으며, **모델이 문제를 일반화하지 못했음**

- 학습 데이터를 다양하게, 많이 수집합니다.
- 정규화(Regularization)를 사용합니다 → 규칙을 단순하게
- 트리플 치즈버거와 같은 이상치는 제거합니다.(데이터가 많다면 제거하는 방법은 좋지 않습니다)

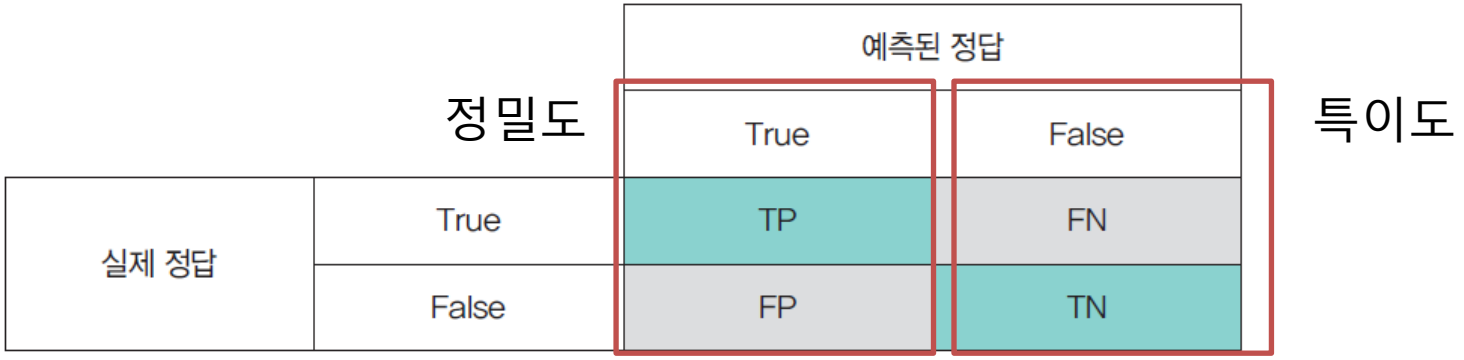
- 과소적합(Underfitting)
  - 모델이 학습 데이터를 충분히 학습하지 않아 모든 측면에서 좋지 않은 성능을 보여주는 결과
  - 모델은 아직 성능이 개선될 여지가 남아있는 상태

- 학습 데이터를 다양하게, 많이 수집합니다.
- 더 복잡한 모델을 사용합니다.
- 모델을 충분히 학습시킵니다.

- 두가지 문제를 동시에 해결할 수 있는 가장 최고의 방법 → **양질의 데이터 수집**

용어: 평가하기

- 혼동행렬(Confusion Matrix)
  - 모델의 성능 평가에 사용
  - **정답(True) 유통기한이 지난 햄버거, 오답(False) 정상 햄버거**



[그림 2-13] 혼동행렬

  - 유통기한이 지난 햄버거를 유통기한이 지난 햄버거로 분류: TP(True Positive)
  - 정상 햄버거를 정상 햄버거로 분류한 경우: TN(True Negative)
  - 유통기한이 지난 햄버거를 정상 햄버거로 잘못 분류한 경우: FN(False Negative)
  - 정상 햄버거를 유통기한이 지난 햄버거로 잘못 분류한 경우: FP(False Positive)

용어: 평가하기

- 정확도(Accuracy)
  - 전체 데이터 중에서 실제 데이터의 정답과 모델이 예측한 정답이 같은 비율

$$\mathrm{ACC} = \frac{TP + TN}{TP + TN + FP + FN}$$

  - 데이터가 불균형할 때 사용하는 경우, 잘못된 지표로써 사용될 수 있음

		예측된 정답	
		유통기한이 지난 햄버거	정상 햄버거
실제 정답	유통기한이 지난 햄버거	90	0
	정상 햄버거	10	0

[그림 2-14] 정말로 정확도가 90%일까요?

용어: 평가하기

- 정밀도와 재현율(Precision&Recall)
  - 정밀도: True라고 예측한 정답 중에서 실제로 True인 비율
    - 실제 False 음성인 데이터 예측을 True 양성으로 잘못 판단하게 되면 업무상 큰 영향이 발생하는 경우
      → **FP이 Critical한 경우** ex) 스팸메일 여부 판단 모형
  - 재현율: 실제 데이터가 True인 것 중에서 모델이 True라고 예측한 비율
    - 실제 True 양성 데이터를 False 로 잘못 판단하게 되면 업무상 큰 영향이 발생하는 경우
      → **FN이 Critical한 경우** ex) 암 판단 모형

		예측된 정답	
		True	False
실제 정답	True	TP	FN
	False	FP	TN

예측 true 중 실제 true ~

$$\text{정밀도} = \frac{TP}{TP + FP}$$

		예측된 정답	
		True	False
실제 정답	True	TP	FN
	False	FP	TN

$$\text{재현율} = \frac{TP}{TP + FN}$$

실제 true 중 예측 true ~

용어: 평가하기

- 정밀도와 재현율(Precision&Recall)
  - 재고 관리 직원과 고객 응대 직원, 어떤 기계를 제공해야 할까요?

A

		예측된 정답	
		유통기한이 지난 햄버거	정상 햄버거
실제 정답	유통기한이 지난 햄버거	30	10
	정상 햄버거	30	30

$$acc = \frac{60(30+30)}{100(30+10+30+30)} = 60\%$$

$$precision = \frac{30}{60(30+30)} = 50\%$$

$$recall = \frac{30}{40(30+10)} = 75\%$$

B

		예측된 정답	
		유통기한이 지난 햄버거	정상 햄버거
실제 정답	유통기한이 지난 햄버거	30	30
	정상 햄버거	10	30

$$acc = \frac{60(30+30)}{100(30+10+30+30)} = 60\%$$

$$precision = \frac{30}{40(30+10)} = 75\%$$

$$recall = \frac{30}{60(30+30)} = 50\%$$

[그림 2–16] 두 햄버거 기계의 성능

  - 재고 관리 직원에게는 정상 햄버거를 유통기한이 지난 햄버거라고 판별하여 버리지 않도록 정밀도를 고려한 기계를 제공
  - 고객 응대 직원에게는 유통기한이 지난 햄버거를 정상 햄버거라고 판별하여 고객에게 주지 않도록 재현율을 고려한 기계를 제공

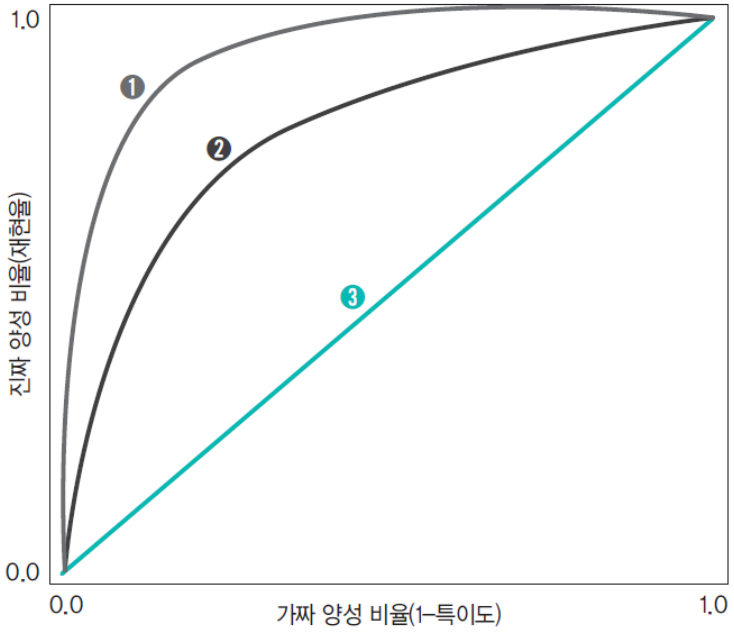
  - 정밀도와 재현율은 Trade-off 관계!

용어: 평가하기

- F1-Score
  - 정밀도와 재현율의 중요성이 같다고 가정하고, 두 지표의 조화평균으로 새로운 지표를 제공

$$F1 = 2 * \frac{precision * recall}{precision + recall}$$

- ROC 곡선
  - 코드로 실습해보자.
  - 코드 결과에서 어떤 모델을 선택하는 것이 좋을지 토론해보세요.
  - 책 내용이 정답은 아님!

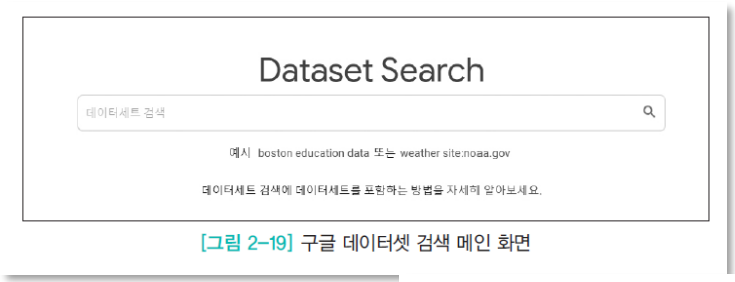


[그림 2-17] ROC 곡선

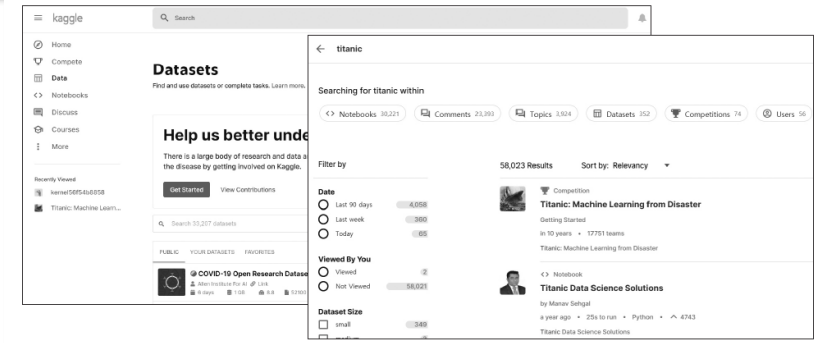
데이터셋 살펴보기

- 프로젝트를 수행하기 위해 데이터를 수집하는 것은 정말 어려운 일
  - 에이~ 그냥 있는거 다운받고, 크롤링 코드짜서 수집해보면 되잖아요.
  - ??? : 일단 해봐.

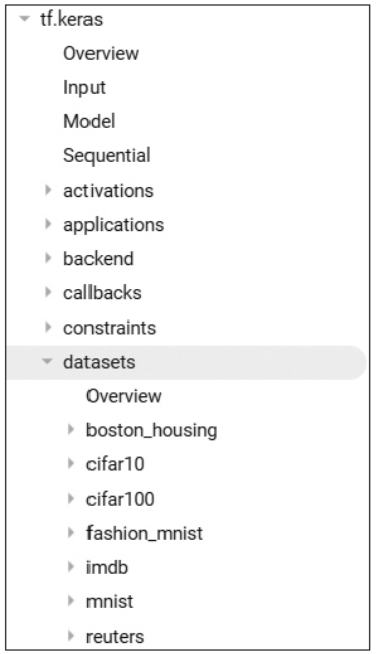
- 기존 사례에서 사용된 데이터셋을 먼저 적용해보는 것도 프로젝트를 성공으로 이끄는 지름길
  - 구글 데이터셋 검색
  - 캐글, AI hub, 공공 데이터 포털 등



[그림 2-19] 구글 데이터셋 검색 메인 화면



[그림 2-20] 캐글 데이터셋 활용



[그림 2-21] tf.keras.datasets

커뮤니티 살펴보기

- **공유와 소통을 할 수 없다면, 이제 살아남기 힘들 것**
  - 커뮤니티를 활용하여 공유와 소통을 실천하고, 적극적으로 질문하자
  - 커뮤니티뿐만 아니라 주변 사람과도 공유와 소통을!

No official relationship with Kaggle

캐글 코리아
Kaggle Korea
Non-Profit Facebook Group Community
함께 공부해서, 함께 나눕시다
Study Together, Share Together

Thank you for your attention