

# 혼자 공부하는 머신러닝+딥러닝 (개정판)



# 시작하기 전에

지은이: 박해선

기계공학을 전공했으나 졸업 후엔 줄곧 코드를 읽고 쓰는 일을 했다. 머신러닝과 딥러닝에 관한 책을 집필하고 번역하면서 소프트웨어와 과학의 경계를 흥미롭게 탐험하고 있다.

『핸즈온 머신러닝 2판』(한빛미디어, 2020)을 포함해서 여러 권의 머신러닝, 딥러닝 책을 우리말로 옮겼고 『Do it! 딥러닝 입문』(이지스퍼블리싱, 2019)을 집필했다.

- 교재의 모든 코드는 웹 브라우저에서 파이썬 코드를 실행할 수 있는 구글 코랩(Colab)을 사용하여 작성했습니다.
- 사용할 실습 환경은 네트워크에 연결된 컴퓨터와 구글 계정입니다.

# 학습 로드맵



## 머신러닝편

01~06장

딥러닝만 먼저 배우고  
싶다면 01~04장을 읽은 후  
07장으로 건너뛰어도 좋습니다.

START

01

나의 첫 머신러닝



02

데이터 다루기



03

회귀 알고리즘과 모델 규제



2번 보기

## 딥러닝편

07~10장

07장을 읽은 후 08장과 09장은  
순서대로 읽지 않아도 괜찮습니다. 10  
장을 읽기 전에 07장과 09장을  
읽는 것이 좋습니다.

난이도

06

비지도 학습



05

트리 알고리즘



04

다양한 분류 알고리즘



07

딥러닝을 시작합니다



08

이미지를 위한 인공 신경망



09

텍스트를 위한 인공 신경망

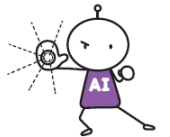


10

언어 모델을 위한 신경망



GOAL



# 이 책의 학습 목표

- **CHAPTER 01: 나의 첫 머신러닝**
  - 인공지능, 머신러닝, 딥러닝의 차이점을 이해합니다.
  - 구글 코랩 사용법을 배웁니다.
  - 첫 번째 머신러닝 프로그램을 만들고 머신러닝의 기본 작동 원리를 이해합니다.
- **CHAPTER 02: 데이터 다루기**
  - 머신러닝 알고리즘에 주입할 데이터를 준비하는 방법을 배웁니다.
  - 데이터 형태가 알고리즘에 미치는 영향을 이해합니다.
- **CHAPTER 03: 회귀 알고리즘과 모델 규제**
  - 지도 학습 알고리즘의 한 종류인 회귀 알고리즘에 대해 배웁니다.
  - 다양한 선형 회귀 알고리즘의 장단점을 이해합니다.
- **CHAPTER 04: 다양한 분류 알고리즘**
  - 로지스틱 회귀, 확률적 경사 하강법과 같은 분류 알고리즘을 배웁니다.
  - 이진 분류와 다중 분류의 차이를 이해하고 클래스별 확률을 예측합니다.
- **CHAPTER 05: 트리 알고리즘**
  - 성능이 좋고 이해하기 쉬운 트리 알고리즘에 대해 배웁니다.
  - 알고리즘의 성능을 최대화하기 위한 하이퍼파라미터 튜닝을 실습합니다.
  - 여러 트리를 합쳐 일반화 성능을 높일 수 있는 앙상블 모델을 배웁니다.

# 이 책의 학습 목표

- **CHAPTER 06: 비지도 학습**

- 타깃이 없는 데이터를 사용하는 비지도 학습과 대표적인 알고리즘을 소개합니다.
- 대표적인 군집 알고리즘인 k-평균과 DBSCAN을 배웁니다.
- 대표적인 차원 축소 알고리즘인 주성분 분석(PCA)을 배웁니다.

- **CHAPTER 07: 딥러닝을 시작합니다**

- 딥러닝의 핵심 알고리즘인 인공 신경망을 배웁니다.
- 대표적인 인공 신경망 라이브러리인 텐서플로와 케라스를 소개합니다.
- 인공 신경망 모델의 훈련을 돕는 도구를 익힙니다.

- **CHAPTER 08: 이미지를 위한 인공 신경망**

- 이미지 분류 문제에 뛰어난 성능을 발휘하는 합성곱 신경망의 개념과 구성 요소에 대해 배웁니다.
- 케라스 API로 합성곱 신경망을 만들어 패션 MNIST 데이터에서 성능을 평가해 봅니다.
- 합성곱 층의 필터와 활성화 출력을 시각화하여 합성곱 신경망이 학습한 내용을 고찰해 봅니다.

- **CHAPTER 09: 텍스트를 위한 인공 신경망**

- 텍스트와 시계열 데이터 같은 순차 데이터에 잘 맞는 순환 신경망의 개념과 구성 요소에 대해 배웁니다.
- 케라스 API로 기본적인 순환 신경망에서 고급 순환 신경망을 만들어 영화 감상평을 분류하는 작업에 적용해 봅니다.
- 순환 신경망에서 발생하는 문제점과 이를 극복하기 위한 해결책을 살펴봅니다.

## CHAPTER 01 나의 첫 머신러닝

SECTION 1-1      인공지능과 머신러닝, 딥러닝

SECTION 1-2      코랩과 주피터 노트북

SECTION 1-3      마켓과 머신러닝



# CHAPTER 01 나의 첫 머신러닝

이 생선의 이름은 무엇인가요?

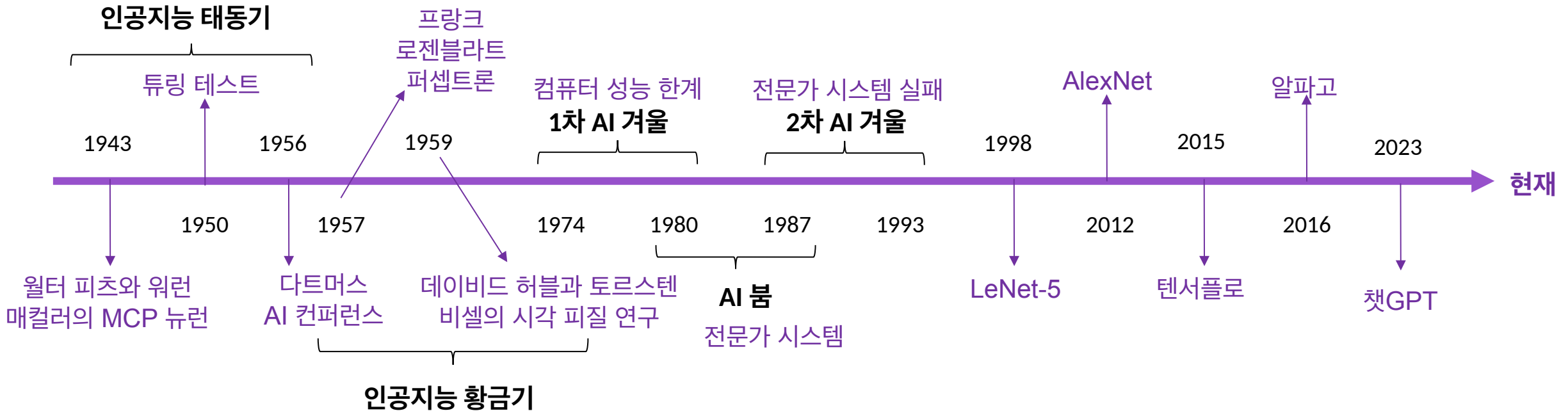
## 학습목표

- 인공지능, 머신러닝, 딥러닝의 차이점을 이해합니다.
- 구글 코랩 사용법을 배웁니다.
- 첫 번째 머신러닝 프로그램을 만들고 머신러닝의 기본 작동 원리를 이해합니다.

# SECTION 1-1 인공지능과 머신러닝, 딥러닝(1)

## ○ 인공지능이란

- 사람처럼 학습하고 추론할 수 있는 지능을 가진 컴퓨터 시스템을 만드는 기술
- 지능을 가진 로봇을 다룬 최초의 소설은 150년 전





# SECTION 1-1 인공지능과 머신러닝, 딥러닝(2)

## ○ 인공지능이란

- 인공일반지능(Artificial general Intelligence) 혹은 강인공지능(Strong AI)
  - 영화에 흔히 등장하는 인공지능
  - 영화 <그녀>의 사만다나 지금도 가장 사악한 인공지능으로 불리는 <터미네이터>의 스카이넷처럼 사람과 구분하기 어려운 지능을 가진 컴퓨터 시스템
- 약인공지능(Weak AI)
  - 현실에서 우리가 마주하고 있는 인공지능
  - 아직까지는 특정 분야에서 사람의 일을 도와주는 보조 역할만 가능  
(음성 비서, 자율 주행 자동차, 음악 추천, 기계 번역 등)
  - 예: 이세돌과 바둑 시합을 한 알파고

# SECTION 1-1 인공지능과 머신러닝, 딥러닝(3)

## 머신러닝이란

- 규칙을 일일이 프로그래밍하지 않아도 자동으로 데이터에서 규칙을 학습하는 알고리즘을 연구하는 분야
- 인공지능의 하위 분야 중에서 지능을 구현하기 위한 소프트웨어를 담당하는 핵심 분야
- 머신러닝은 통계학과 깊은 관련이 있음
  - 통계학에서 유래된 머신러닝 알고리즘이 많으며, 통계학과 컴퓨터 과학 분야가 상호 작용하면서 발전
  - 대표적인 오픈소스 통계 소프트웨어인 R에는 다양한 머신러닝 알고리즘이 구현되어 있음
- 최근 머신러닝의 발전은 통계나 수학 이론보다 경험을 바탕으로 발전하는 경우도 많음
  - 컴퓨터 과학 분야가 이런 발전을 주도, 컴퓨터 과학 분야의 대표적인 머신러닝 라이브러리는 사이킷런
  - 사이킷런이 있기 전까지 머신러닝 기술은 대부분 폐쇄적인 코드와 라이브러리로 통용되어서 해당 분야에 대한 전문 교육을 이수하거나 비싼 비용을 지불하고 구매를 해야 했음
  - 하지만 사이킷런과 같은 오픈소스 라이브러리의 발전 덕분에 머신러닝 분야는 말 그대로 폭발적으로 성장
  - 파이썬 코드를 다룰 수 있다면 누구나 머신러닝 알고리즘을 무료로 손쉽게 제품에 활용할 수 있음

# SECTION 1-1 인공지능과 머신러닝, 딥러닝(4)

## ◦ 딥러닝이란

- 딥러닝(deep learning): 많은 머신러닝 알고리즘 중에 인공 신경망(artificial neural network)을 기반으로 한 방법들의 통칭
- 대부분 인공 신경망과 딥러닝을 크게 구분하지 않고 사용
- 최초의 합성곱 신경망: 1998년 얀 르쿤(Yann Lecun)이 신경망 모델 LeNet-5을 만들어 손글씨 숫자를 인식하는 데 성공
- 합성곱 신경망이 이미지 분류 작업에 널리 사용되기 시작: 2012년에 제프리 힌턴(Geoffrey Hinton)의 팀이 AlexNet 모델로 이미지 분류 대회인 ImageNet에서 기존의 머신러닝 방법을 누르고 우승
- 국내에서는 2016년 이세돌과 알파고의 대국으로 인해 딥러닝에 대한 관심이 크게 상승
- 2022년에는 챗GPT가 출시되면서 일반 대중이 인공지능 기술을 직접 활용하는 시대를 열었습니다.

# SECTION 1-1 인공지능과 머신러닝, 딥러닝(5)

- 인공 신경망 성능 발전의 원동력 세 가지
  - 1) 복잡한 알고리즘을 훈련할 수 있는 풍부한 데이터
  - 2) 컴퓨터 성능의 향상
  - 3) 혁신적인 알고리즘 개발
- 2015년 구글은 딥러닝 라이브러리인 텐서플로(TensorFlow)를 오픈소스로 공개
- 2018년 페이스북도 파이토치(PyTorch) 딥러닝 라이브러리를 오픈소스로 발표
- 이 라이브러리들의 공통점은 인공 신경망 알고리즘을 전문으로 다루고 있다는 것과 모두 사용하기 쉬운 파이썬 API를 제공한다는 점



## SECTION 1-1 마무리

- 인공지능은 사람처럼 학습하고 추론할 수 있는 지능을 가진 시스템을 만드는 기술
  - 인공지능은 강인공지능과 약인공지능으로 구분
- 머신러닝은 규칙을 프로그래밍하지 않아도 자동으로 데이터에서 규칙을 학습하는 알고리즘을 연구하는 분야
  - 사이킷런이 대표적인 라이브러리
- 딥러닝은 인공 신경망이라고도 하며, 텐서플로와 파이토치가 대표적인 라이브러리

# SECTION 1-2 코랩과 주피터 노트북(1)

## ◦ 구글 코랩

- 구글 코랩(Colab)은 웹 브라우저에서 무료로 파이썬 프로그램을 테스트하고 저장할 수 있는 서비스
- 머신러닝 프로그램 제작도 가능한 클라우드 기반의 주피터 노트북 개발 환경
- 머신러닝은 컴퓨터 사양이 중요한데, 구글 코랩을 사용하면 컴퓨터 성능과 상관없이 프로그램을 실습해 볼 수 있음
- 구글 계정만 있다면 누구나 무료로 코랩을 사용할 수 있음
- 코랩 웹사이트(<https://colab.research.google.com>)에 접속하고 로그인
  - 구글 계정으로 로그인하지 않아도 코랩에 접속할 수 있지만, 코드를 실행할 수는 없음
- 코랩은 웹 브라우저에서 텍스트와 프로그램 코드를 자유롭게 작성할 수 있는 온라인 에디터
  - 이런 코랩 파일을 노트북(Notebook) 혹은 코랩 노트북이라고 칭함
  - 노트북에서 셀(cell)은 코드 또는 텍스트의 덩어리
  - 노트북은 보통 여러 개의 코드 셀과 텍스트 셀로 구성

# SECTION 1-2 코랩과 주피터 노트북(2)












## ◦ 텍스트 셀

- 셀(cell): 코랩에서 실행할 수 있는 최소 단위
  - 셀 안에 있는 내용을 한 번에 실행하고 그 결과를 노트북에 나타냄
  - 텍스트 셀은 코드처럼 실행되는 것이 아니기 때문에 자유롭게 사용해도 괜찮음
  - 셀 하나에 아주 긴 글을 써도 되고, 여러 셀에 나누어 작성해도 무방함
- 텍스트 셀 수정: 원하는 셀로 이동한 후 Enter 키를 누르거나 마우스를 더블 클릭하여 편집 화면으로 들어감
- 텍스트 셀에서는 HTML과 마크다운(Markdown)을 혼용해서 사용 가능
  - 왼쪽 창에서 텍스트를 수정하면 오른쪽 미리 보기 창에서 수정된 결과를 바로 볼 수 있음
  - <h1> 태그 아래에 임의의 텍스트를 추가하고 미리 보기 창에 나타나는 결과를 확인해보기
- 텍스트 셀의 수정을 끝내려면 ESC 키를 누름








# SECTION 1-2 코랩과 주피터 노트북(3)

## ◦ 텍스트 셀

### - 기본 메뉴

1		현재 라인을 제목으로 바꾸기
2		선택한 글자를 굵은 글자로 바꾸기
3		선택한 글자를 이탤릭체로 바꾸기
4		코드 형식으로 바꾸기
5		선택한 글자를 링크로 만들기
6		현재 커서 위치에 이미지를 추가
7		현재 커서 위치에 들여 쓴 블록을 추가
8		현재 커서 위치에 번호 매기기 목록을 추가
9		현재 커서 위치에 글머리 기호 목록을 추가
10		현재 커서 위치에 가로줄을 추가
11		미리 보기 창의 위치를 오른쪽에서 아래로 또는 아래에서 오른쪽으로 바꾸기

### 현재 셀에 적용할 수 있는 기능

-  위로 이동
-  아래로 이동
-  링크 만들기
-  수정
-  탭에서 셀 미러링
-  셀 삭제
-  셀 작업 더보기



## SECTION 1-2 코랩과 주피터 노트북(4)

- 코드 셀

- 'Colaboratory에 오신 것을 환영합니다' 노트북의 세 번째 셀이 코드 셀
- 코드 셀로 이동하면 코드와 결과가 함께 선택됨

```
seconds_in_a_day = 24 * 60 * 60  
seconds_in_a_day
```

← 코드



86400

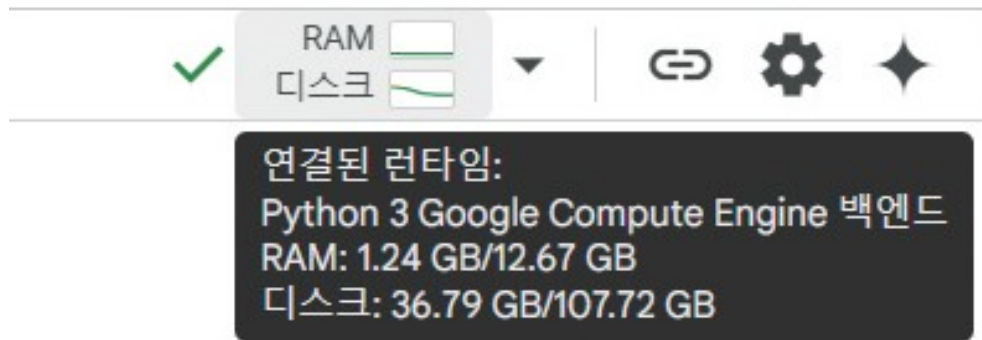


결과

## SECTION 1-2 코랩과 주피터 노트북(5)

### ◦ 노트북

- 코랩은 구글이 대화식 프로그래밍 환경인 주피터(Jupyter)를 커스터마이징한 것
- 파이썬 지원으로 시작한 주피터 프로젝트는 최근에는 다른 언어도 지원
- 주피터 프로젝트의 대표 제품이 바로 노트북(Notebook), 흔히 주피터 노트북이라고 부름
- 코랩 노트북은 구글 클라우드의 가상 서버(Virtual Machine)를 사용
- 화면 오른쪽 상단에 있는 RAM, 디스크 아이콘에 마우스를 올리면 상세 정보를 알 수 있음  
코드를 실행하기 전이나 연결이 끊어진 상태에서는 아이콘 대신에 [연결] 버튼이 활성화됨



# SECTION 1-2 코랩과 주피터 노트북(6)

## ◦ 노트북

- 노트북은 구글 클라우드의 컴퓨트 엔진(Compute Engine)에 연결되어 있음
    - 서버의 메모리는 약 12기가이고, 디스크 공간은 약 225기가
    - 구글 계정만 있으면 코랩 노트북을 사용해 무료로 가상 서버를 활용할 수 있음
    - 무료 제한 사항으로 코랩 노트북으로 동시에 사용할 수 있는 구글 클라우드의 가상 서버는 최대 5개
    - 5개 이상의 노트북을 열어야 한다면 이미 실행 중인 노트북을 저장한 다음 구글 클라우드와 연결을 끊어야 함
    - 1개의 노트북을 12시간 이상 실행할 수 없음
- 구글은 2020년 2월 더 많은 메모리와 컴퓨팅 파워를 제공하는 코랩 프로(Colab Pro) 유료 서비스 시작
  - 코랩 프로는 한 번에 최대 24 시간 동안 프로그램을 실행할 수 있음
  - 현재는 미국과 캐나다에서만 서비스 가입이 가능

## SECTION 1-2 코랩과 주피터 노트북(7)

### ◦ 새 노트북 만들기

- 01 [파일]-[새 노트]를 클릭해서 새로운 노트북 시작
- 02 새 노트북은 Untitled[숫자].ipynb 이름으로 만들어지고, 노트북에는 빈 코드의 셀 하나가 나타남
- 03 코드 셀에 'Hello World'를 출력하는 `print( )` 코드를 작성하고, 이 파일의 이름을 'Hello World'로 저장하기
  - 빈 코드 셀을 마우스로 선택하고 다음과 같이 입력한 다음 셀을 실행
  - 코드 셀을 실행하려면 Ctrl + Enter 키(macOS는 cmd + Enter 키)를 누르거나 왼쪽에 있는 플레이 아이콘을 클릭
- 04 노트북은 자동으로 구글 드라이브의 [내 드라이브]-[Colab Notebooks] 폴더 아래에 저장됨
  - 웹 브라우저에서 구글 드라이브(<https://drive.google.com>)로 접속해서 확인
- 05 노트북의 이름 변경
  - 제목을 마우스로 클릭하면 수정할 수 있도록 바뀜
  - 이 파일의 제목을 'Hello World'로 바꿔 보기

# SECTION 1-2 코랩과 주피터 노트북(8)

## ○ 새 노트북 만들기

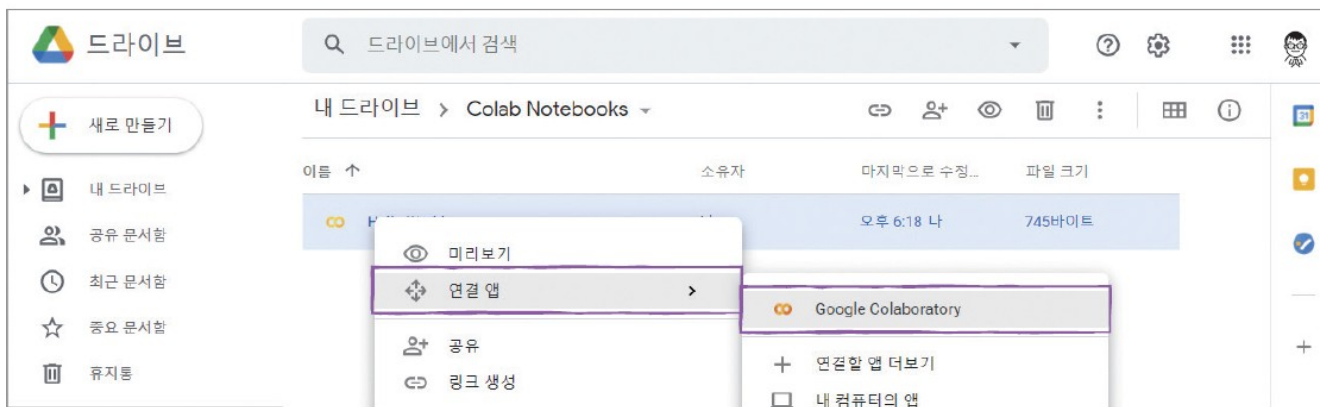
06        노트북 이름을 수정하고 저장하면 잠시 후 구글 드라이브에 있는 이름이 변경됨

07        이렇게 저장된 노트북을 코랩으로 불러오기

- 코랩 노트북 화면에서 [파일]-[노트 열기]를 선택

- 팝업 창에서 [Google 드라이브]를 선택하면 [Colab Notebooks]에 들어간 노트북을 코랩에서 열거나 구글 드라이브에서 코랩 노트북을 선택하고 마우스 오른쪽 버튼을 클릭하여 팝업 메뉴를 띄운 다음 [연결 앱]-[Google Colaboratory]를 선택하면 노트북을 코랩에서 열기

- 만약 [연결 앱] 목록에 [Google Colaboratory]가 없다면, [연결할 앱 더보기]를 클릭한 후 Colab을 검색해 설치해 주세요.



# SECTION 1-2 마무리(1)

## ◦ 키워드로 끝내는 핵심 포인트

- 코랩은 구글 계정이 있으면 누구나 사용할 수 있는 웹 브라우저 기반의 파이썬 코드 실행 환경
- 노트북은 코랩의 프로그램 작성 단위이며, 일반 프로그램 파일과 달리 대화식으로 프로그램을 만들 수 있기 때문에 데이터 분석이나 교육에 매우 적합  
    노트북에는 코드, 코드의 실행 결과, 문서를 모두 저장하여 보관할 수 있음
- 구글 드라이브는 구글이 제공하는 클라우드 파일 저장 서비스. 코랩에서 만든 노트북은 자동으로 구글 클라우드의 'Colab Notebooks' 폴더에 저장되고 필요할 때 다시 코랩에서 열 수 있음

## ◦ 표로 정리하는 툴바와 마크다운

	제목 전환	<b>B</b>	굵게	<i>I</i>	기울임 꼴
	코드로 형식 지정		링크 삽입		이미지 삽입
	인용구 추가		번호 매기기 목록 추가		글머리 기호 목록 추가
	가로줄 추가		레이텍		이모티콘 삽입
	마크다운 미리 보기 위치 변경				

# SECTION 1-2 마무리(2)

- 텍스트 셀에 사용할 수 있는 마크다운

마크다운 형식	설명	예제
# 제목1	<h1> 태그와 동일	제목1
## 제목2	<h2> 태그와 동일	제목2
### 제목3	<h3> 태그와 동일	제목3
#### 제목4	<h4> 태그와 동일	제목4
##### 제목5	<h5> 태그와 동일	제목5
**혼공머신**	굵게 쓰기	혼공신
*혼공머신* 또는 _혼공머신_	기울임 꼴	혼공신/
~~혼공머신~~	취소선을 추가	혼공신
`print("Hello World!")`	백틱 기호를 사용한 코드 서체	Print ("Hello World")
> 혼공머신	들여쓰기(여러 단계를 들여쓸 수 있음)	혼공신
* 혼공머신 또는 - 혼공머신	글머리 기호 목록	• 혼공신
[한빛미디어](http://www.hanbit.co.kr/)	링크 생성	<a href="http://www.hanbit.co.kr/">한빛미디어</a>
! [한빛미디어](http://www.hanbit.co.kr/images/common/logo_hanbit.png)	이미지를 추가	
\$ y = x \times z \$	레이텍을 추가	$Y = X \times Z$

※ 레이텍(LaTeX)은 수식, 그래프, 다이어그램 등을 그리는 데 유용한 문서 저작도구로 보통 논문 작성에 많이 사용

## SECTION 1-2 확인 문제(1)

1 구글에서 제공하는 웹 브라우저 기반의 파이썬 실행 환경은 무엇인가?

- ① 주피터 노트북                      ② 코랩
- ③ 크롬                                  ④ 아나콘다

2 코랩 노트북에서 쓸 수 있는 마크다운 중에서 다음 중 기울임 꼴로 쓰는 것은?

- ① **\*\*혼공머신\*\***                      ② *~~혼공머신~~*
- ③ ``혼공머신``                      ④ 혼공머신

3 코랩 노트북은 어디에서 실행되는가?

- ① 내 컴퓨터                      ② 구글 드라이브
- ③ 구글 클라우드                      ④ 아마존 웹서비스

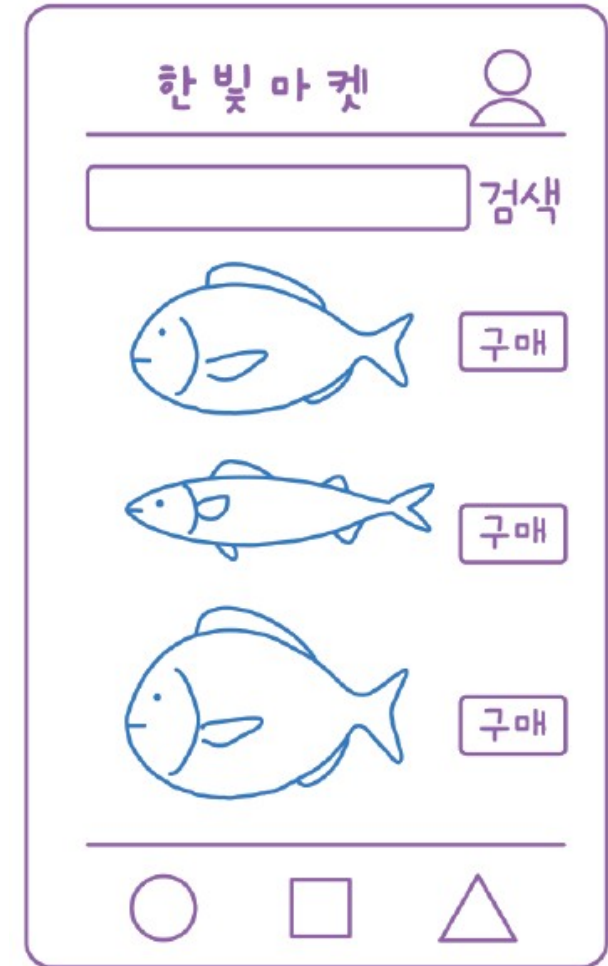


## SECTION 1-3 마켓과 머신러닝(1)

- k-최근접 이웃을 사용하여 2개의 종류를 분류하는 머신러닝 모델을 훈련해보기

- 학습의 가상 조건 설정

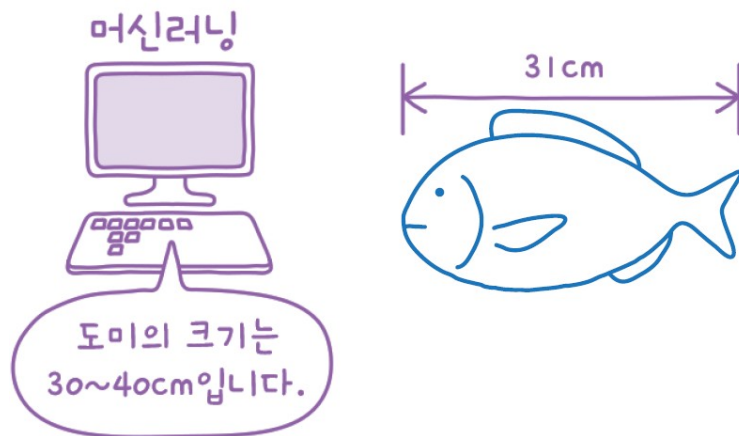
- 한빛 마켓은 앱 마켓 최초로 살아 있는 생선을 판매하기 시작
- 고객이 온라인으로 주문하면 가장 빠른 물류 센터에서 신선한 생선을 곧바로 배송
- 문제 발생
  - 물류 센터에서 생선을 고르는 직원이 도통 생선 이름을 외우지 못함
  - 항상 주위 사람에게 “이 생선 이름이 뭐예요?”라고 물어봐 배송이 지연되기 일쑤
- 문제 해결
  - 훈공머신에게 첫 번째 임무로 생선 이름을 자동으로 알려주는 머신러닝을 만들기



## SECTION 1-3 마켓과 머신러닝(2)

### ◦ 생선 분류 문제

- 한빛 마켓에서 팔기 시작한 생선은 ‘도미’, ‘곤들매기’, ‘농어’, ‘강꼬치고기’, ‘로치’, ‘빙어’, ‘송어’
- 이 생선들은 물류 센터에 많이 준비되어 있고, 이 생선들을 프로그램으로 분류한다고 가정하여 적합한 프로그램을 만들기
- 사용할 생선 데이터는 캐글에 공개된 데이터셋
  - <https://www.kaggle.com/aungpyaeap/fish-market>
  - 캐글(kaggle.com)은 2010년에 설립된 전 세계에서 가장 큰 머신러닝 경연 대회 사이트로, 대회 정보뿐만 아니라 많은 데이터와 참고 자료를 제공



# SECTION 1-3 마켓과 머신러닝(3)

## ◦ 생선 분류 문제

- 생선을 분류하는 일이니 생선의 특징을 알면 쉽게 구분할 수 있을 것으로 예상
  - 도미 특성1: 생선 길이가 30cm 이상이면 도미
  - 특성을 바탕으로 혼공머신이 만든 파이썬 프로그램
  - 문제 발생
    - 30cm보다 큰 생선이 무조건 도미? 또 도미의 크기가 모두 같을 리도 없음
  - 한빛 마켓에서 판매하는 생선은 이렇게 절대 바뀌지 않을 기준을 정하기 어려움
  - 이 문제를 머신러닝으로 어떻게 해결할 수 있을까?
    - 보통 프로그램은 ‘누군가 정해진 기준대로 일’ 을 수행
    - 반대로 머신러닝은 누구도 알려주지 않는 기준을 찾아서 일을 수행
- 즉, 누가 말해 주지 않아도 머신러닝은 “30~40cm 길이의 생선은 도미이다” 라는 기준을 찾음
- 머신러닝은 기준을 찾을 뿐만 아니라 이 기준을 이용해 생선이 도미인지 아닌지 판별

```
if fish_length >= 30:  
    print("도미")
```

## SECTION 1-3 마켓과 머신러닝(4)

- 도미 데이터 준비하기
  - 머신러닝은 어떻게 이런 기준을 스스로 찾을 수 있을까?
    - 머신러닝은 여러 개의 도미 생선을 보면 스스로 어떤 생선이 도미인지를 구분할 기준을 찾음
    - 그렇다면 도미 생선을 많이 준비해야 함
- 무게와 길이를 함께 재어 주는 저울을 이용하여 데이터 수집
  - 먼저 혼공머신은 머신러닝을 사용해 도미와 빙어를 구분
  - 도미와 빙어를 준비해서 저울에 올려놓고 무게와 길이를 측정
- 길이와 무게를 입력하기 전에 코랩에서 새 노트를 하나 생성
  - 코랩 메뉴에서 [파일]-[새노트]를 클릭해서 노트를 생성한 다음에 제목은 ‘BreamAndSmelt’라 수정하고, 이후 과정을 입력

# SECTION 1-3 마켓과 머신러닝(5)

## ◦ 도미 데이터 준비하기

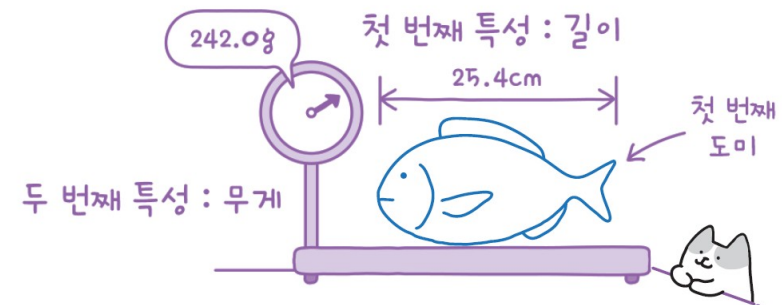
- 35마리의 도미를 준비하고 저울로 잰 도미의 길이(cm)와 무게(g)를 파이썬 리스트로 준비  
(데이터 소스: [http://bit.ly/bream\\_list](http://bit.ly/bream_list))

```
bream_length = [ 25.4, 26.3, 26.5, 29.0, 29.0, 29.7, 29.7, 30.0, 30.0, 30.7,  
                31.0, 31.0, 31.5, 32.0, 32.0, 32.0, 33.0, 33.0, 33.5, 33.5,  
                34.0, 34.0, 34.5, 35.0, 35.0, 35.0, 35.0, 36.0, 36.0, 37.0,  
                38.5, 38.5, 39.5, 41.0, 41.0]  
  
bream_weight = [ 242.0, 290.0, 340.0, 363.0, 430.0, 450.0, 500.0, 390.0,  
                450.0, 500.0, 475.0, 500.0, 500.0, 340.0, 600.0, 600.0,  
                700.0, 700.0, 610.0, 650.0, 575.0, 685.0, 620.0, 680.0,  
                700.0, 725.0, 720.0, 714.0, 850.0, 1000.0, 920.0, 955.0,  
                925.0, 975.0, 950.0]
```

생선의 길이

생선의 무게

- 리스트에서 첫 번째 도미의 길이는 25.4cm, 무게는 242.0g이고  
두 번째 도미의 길이는 26.3cm, 무게는 290.0g
- 특성(feature): 각 도미를 길이와 무게로 표현

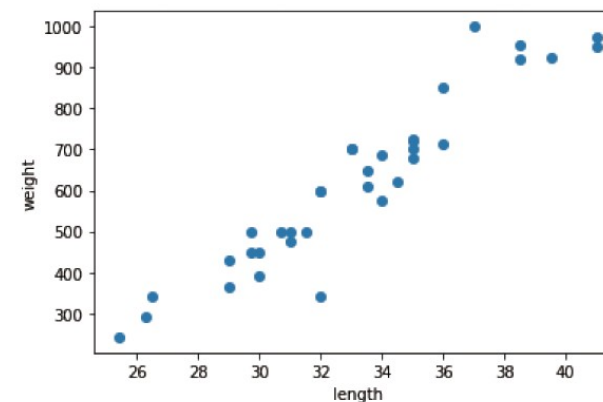


## SECTION 1-3 마켓과 머신러닝(6)

- 산점도(scatter plot)
  - 두 특성을 숫자에서 그래프로 표현
  - 길이를 x축, 무게를 y축
  - 각 도미를 이 그래프에 점으로 표시
- 파이썬에서 과학계산용 그래프를 그리는 대표적인 패키지는 맷플롯립(matplotlib)
  - 이 패키지를 임포트하고 산점도를 그리는 scatter( ) 함수 사용

```
import matplotlib.pyplot as plt # matplotlib의 pyplot 함수를 plt로 줄여서 사용

plt.scatter(bream_length, bream_weight)
plt.xlabel('length') # x축은 길이
plt.ylabel('weight') # y축은 무게
plt.show()
```



- 생선의 길이가 길수록 무게가 많이 나간다고 생각하면 이 그래프 모습은 매우 자연스러움
- 이렇게 산점도 그래프가 일직선에 가까운 형태로 나타나는 경우를 선형(linear)적이라고 표현

## SECTION 1-3 마켓과 머신러닝(7)

- 빙어 데이터 준비하기

- 빙어 14마리의 데이터를 앞에서와 같이 파이썬 리스트로 만들기

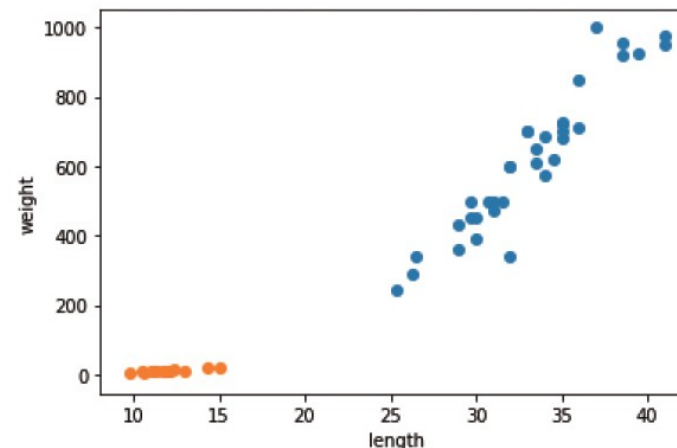
(데이터 소스: [http://bit.ly/smelt\\_list](http://bit.ly/smelt_list))

```
smelt_length = [ 9.8, 10.5, 10.6, 11.0, 11.2, 11.3, 11.8, 11.8, 12.0, 12.2,  
                12.4, 13.0, 14.3, 15.0]  
smelt_weight = [ 6.7, 7.5, 7.0, 9.7, 9.8, 8.7, 10.0, 9.9, 9.8, 12.2, 13.4,  
                12.2, 19.7, 19.9]
```

- 빙어의 산점도 그래프

- scatter( ) 함수를 연달아 사용하여 2개의 산점도를 한 그래프로 그리기

```
plt.scatter(bream_length, bream_weight)  
plt.scatter(smelt_length, smelt_weight)  
plt.xlabel('length')  
plt.ylabel('weight')  
plt.show()
```



## SECTION 1-3 마켓과 머신러닝(8)

### ◦ 첫 번째 머신러닝 프로그램

- k-최근접 이웃(k-Nearest Neighbors) 알고리즘을 사용해 도미와 빙어 데이터를 구분
  - k-최근접 이웃 알고리즘을 사용하기 전에 앞에서 준비했던 도미와 빙어 데이터를 하나의 데이터로 합침
  - 파이썬에서는 다음처럼 두 리스트를 더하면 하나의 리스트로 만들어 줌

```
length = bream_length + smelt_length  
weight = bream_weight + smelt_weight
```

도미 35개의 길이      빙어 14개의 길이

length = [25.4, 26.3, ... , 41.0, 9.8, ... , 15.0]

도미 35개의 무게      빙어 14개의 무게

weight = [242.0, 290.0, ... , 950.0, 6.7, ... , 19.9]



## SECTION 1-3 마켓과 머신러닝(9)

### ◦ 첫 번째 머신러닝 프로그램

- 사이킷런(scikit-learn) 패키지를 사용하려면 오른쪽처럼 각 특성의 리스트를 세로 방향으로 늘어뜨린 2차원 리스트를 만들어야 함

49개의 생선

길이	무게
25.4	242.0
26.3	290.0
.	.
.	.
.	.
15.0	19.9

- 파이썬의 zip( ) 함수와 리스트 내포(list comprehension) 구문을 사용
  - zip( ) 함수는 나열된 리스트 각각에서 하나씩 원소를 꺼내 반환
  - zip( ) 함수와 리스트 내포 구문을 사용해 length와 weight 리스트를 2차원 리스트로 만들기

```
fish_data = [[l, w] for l, w in zip(length, weight)]
```

- for 문은 zip( ) 함수로 length와 weight 리스트에서 원소를 하나씩 꺼내어 l과 w에 할당하면 [l, w]가 하나의 원소로 구성된 리스트가 만들어짐
- 예상대로 fish\_data가 만들어졌는지 출력해서 확인

## SECTION 1-3 마켓과 머신러닝(10)

## 첫 번째 머신러닝 프로그램

- 첫 번째 생선의 길이 25.4cm와 무게 242.0g이 하나의 리스트를 구성하고, 이런 리스트가 모여 전체 리스트를 만들었음. 이런 리스트를 2차원 리스트 혹은 리스트의 리스트라고 부름
- 정답 데이터 준비
  - 첫 번째 생선은 도미이고, 두 번째 생선도 도미라는 식으로 각각 어떤 생선인지 답을 만드는 작업
  - 도미와 빙어를 숫자 1과 0으로 표현
    - 예를 들어 첫 번째 생선은 도미이므로 1이고, 마지막 생선은 빙어이므로 0
  - 앞서 도미와 빙어를 순서대로 나열했기 때문에 정답 리스트는 1이 35번 등장하고 0이 14번 등장하면 됨
  - 곱셈 연산자를 사용하면 파이썬 리스트를 간단하게 반복시킬 수 있음

```
fish_target = [1] * 35 + [0] * 14
print(fish_target)
```



[1,  
1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

# SECTION 1-3 마켓과 머신러닝(11)

## ◦ 첫 번째 머신러닝 프로그램

- 사이킷런 패키지에서 k-최근접 이웃 알고리즘을 구현한 클래스인 KNeighborsClassifier를 импорт

```
from sklearn.neighbors import KNeighborsClassifier
```

- импорт한 KNeighborsClassifier 클래스의 객체 만들기

```
kn = KNeighborsClassifier()
```

- 훈련(training)

- 이 객체에 fish\_data와 fish\_target을 전달하여 도미를 찾기 위한 기준을 학습시킴
- 이런 과정을 머신러닝에서는 훈련이라고 부름
- 사이킷런에서는 fit( ) 메서드가 이런 역할을 함
- 이 메서드에 fish\_data와 fish\_target을 순서대로 전달

```
kn.fit(fish_data, fish_target)
```

- 객체(또는 모델) kn이 얼마나 잘 훈련되었는지 평가
  - 사이킷런에서 모델을 평가하는 메서드는 score( ) 메서드이고, 0에서 1 사이의 값을 반환 (1은 모든 데이터를 정확히 맞췄다는 것을 나타냄. 예를 들어 0.5라면 절반만 맞췄다는 의미)

```
kn.score(fish_data, fish_target)
```



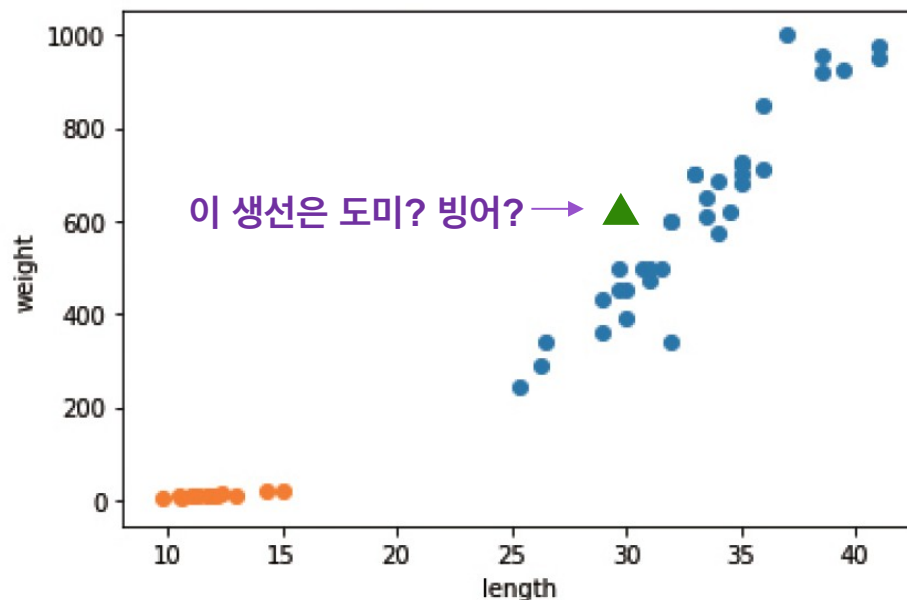
1.0

# SECTION 1-3 마켓과 머신러닝(12)

## ◦ 첫 번째 머신러닝 프로그램

### - k-최근접 이웃 알고리즘

- 앞에서 첫 번째 머신러닝 프로그램을 성공적으로 만들었는데, 여기서 사용한 알고리즘은 k-최근접 이웃
- 어떤 데이터에 대한 답을 구할 때 주위의 다른 데이터를 보고 다수를 차지하는 것을 정답으로 사용  
마치 근목자혹과 같이 주위의 데이터로 현재 데이터를 판단
- 다음 그림에 삼각형으로 표시된 새로운 데이터가 있다고 가정하면, 이 삼각형은 도미와 빙어 중 어디에 속할까?



# SECTION 1-3 마켓과 머신러닝(13)

- 첫 번째 머신러닝 프로그램

- k-최근접 이웃 알고리즘

- predict( ) 메서드는 새로운 데이터의 정답을 예측
    - 이 메서드도 앞서 fit( ) 메서드와 마찬가지로 리스트의 리스트를 전달해야 함  
그래서 삼각형 포인트를 리스트로 2번 감싸고, 반환되는 값은 1
    - 우리는 앞서 도미는 1, 빙어는 0으로 가정. 따라서 삼각형( )은 도미▲

```
kn.predict([[30, 600]])
```



```
array([1])
```

- k-최근접 이웃 알고리즘을 위해 준비해야 할 일은 데이터를 모두 가지고 있는 게 전부
    - 새로운 데이터에 대해 예측할 때는 가장 가까운 직선거리에 어떤 데이터가 있는지를 살피기만 하면 됨
    - 단점은 k-최근접 이웃 알고리즘의 이런 특징 때문에 데이터가 아주 많은 경우 사용하기 어려움  
데이터가 크기 때문에 메모리가 많이 필요하고, 직선거리를 계산하는 데도 많은 시간이 필요
  - 사이킷런의 KNeighborsClassifier 클래스도 마찬가지
    - 이 클래스는 \_fit\_X 속성에 우리가 전달한 fish\_data를 모두 가지고 있으며, \_y 속성에 fish\_target을 가짐

# SECTION 1-3 마켓과 머신러닝(14)

## ◦ 첫 번째 머신러닝 프로그램

### - k-최근접 이웃 알고리즘

- 실제로 k-최근접 이웃 알고리즘은 무언가 훈련되는 게 없는 것인가?
- `fit()` 메서드에 전달한 데이터를 모두 저장하고 있다가 새로운 데이터가 등장하면 가장 가까운 데이터를 참고하여 도미인지 빙어인지를 구분
- 가까운 몇 개의 데이터를 참고할지는 정하기 나름이지만, `KNeighborsClassifier` 클래스의 기본값은 5임  
이 기준은 `n_neighbors` 매개변수로 바꿀 수 있음

- 다음과 같이 하면 어떤 결과가 나올까?

```
kn49 = KNeighborsClassifier(n_neighbors=49) # 참고 데이터를 49개로 한 kn49 모델
```

- 가장 가까운 데이터 49개를 사용하는 k-최근접 이웃 모델에 `fish_data`를 적용하면 `fish_data`에 있는 모든 생선을 사용하여 예측
  - `fish_data`의 데이터 49개 중에 도미가 35개로 다수를 차지하므로 어떤 데이터를 넣어도 무조건 도미로 예측

```
kn49.fit(fish_data, fish_target)  
kn49.score(fish_data, fish_target)
```



0.7142857142857143

# SECTION 1-3 마켓과 머신러닝(14)

- 첫 번째 머신러닝 프로그램

- k-최근접 이웃 알고리즘

- fish\_data에 있는 생선 중에 도미가 35개이고 빙어가 14개
    - kn49 모델은 도미만 올바르게 맞히기 때문에 다음과 같이 정확도를 계산하면 score( ) 메서드와 같은 값을 얻을 수 있음

```
print(35/49)
```



0.7142857142857143

- n\_neighbors 매개변수를 49로 두는 것은 좋지 않음
    - 기본값을 5로 하여 도미를 완벽하게 분류한 모델을 사용



결괏값은 왜 한 번만 출력될까?

- kn49.score() 다음에 바로 print 명령을 사용하면 결괏값이 2번 출력되어야 할 것 같지만 그렇지 않음
    - 코드 셀은 마지막 실행 코드의 반환값만을 자동 출력
    - 모든 코드를 한 셀에 넣으면 중간의 반환값은 출력하지 않음
    - 따라서 두 값을 모두 출력하려면 각각 print 명령을 사용하거나 여러 개의 코드 셀로 나누어 작성해야 함

# SECTION 1-3 마켓과 머신러닝(15)

- 도미와 빙어 분류 (문제해결 과정)

- 분류 준비 과정

- 도미와 빙어를 구분하기 위한 첫 번째 머신러닝 프로그램을 작성
- 먼저 도미 35마리와 빙어 14마리의 길이와 무게를 측정해서 파이썬 리스트 생성
- 도미와 빙어 데이터를 합쳐 리스트의 리스트로 데이터를 준비

- 첫 번째 머신러닝 프로그램

- k-최근접 이웃 알고리즘
- 사이킷런의 k-최근접 이웃 알고리즘은 주변에서 가장 가까운 5개의 데이터를 보고 다수결의 원칙에 따라 데이터를 예측
- 이 모델은 혼공머신이 준비한 도미와 빙어 데이터를 모두 완벽하게 맞힘

- 사용 메서드

- KNeighborsClassifier 클래스의 fit ( ), score ( ), predict ( ) 메서드를 사용



# SECTION 1-3 마무리(1)

- 키워드로 끝내는 핵심 포인트
  - 특성은 데이터를 표현하는 하나의 성질. 이 절에서 생선 데이터 각각을 길이와 무게 특성으로 표현
  - 머신러닝 알고리즘이 데이터에서 규칙을 찾는 과정이 훈련
    - 사이킷런에서는 `fit()` 메서드가 하는 역할
  - k-최근접 이웃 알고리즘은 가장 간단한 머신러닝 알고리즘 중 하나
    - 사실 어떤 규칙을 찾기보다는 전체 데이터를 메모리에 가지고 있는 것이 전부
  - 머신러닝 프로그램에서 알고리즘이 구현된 객체를 모델이라 함
    - 종종 알고리즘 자체를 모델이라고 부르기도 함
  - 정확도는 정확한 답을 몇 개 맞혔는지를 백분율로 나타낸 값
    - 사이킷런에서는 0 ~1 사이의 값으로 출력
    - $\text{정확도} = (\text{정확히 맞힌 개수}) / (\text{전체 데이터 개수})$

# SECTION 1-3 마무리(2)

- 핵심 패키지와 함수

- matplotlib

- scatter( )는 산점도를 그리는 맷플롯립 함수
    - 처음 2개의 매개변수로 x축 값과 y축 값을 전달
    - 이 값은 파이썬 리스트 또는 넘파이 배열
    - c 매개변수로 색깔을 지정
    - 지정하지 않을 경우 10개의 기본 색깔을 사용해 그래프를 그림
    - 색깔은 [https://bit.ly/matplotlib\\_prop\\_cycle](https://bit.ly/matplotlib_prop_cycle)을 참고
    - marker 매개변수로 마커 스타일을 지정
    - marker의 기본값은 o(circle, 원)
    - 지정할 수 있는 마커 종류는 [https://bit.ly/matplotlib\\_marker](https://bit.ly/matplotlib_marker)를 참고

## SECTION 1-3 마무리(3)

- 핵심 패키지와 함수

- scikit-learn

- KNeighborsClassifier( )는 k-최근접 이웃 분류 모델을 만드는 사이킷런 클래스
    - n\_neighbors 매개변수로 이웃의 개수를 지정. 기본값은 5
    - p 매개변수로 거리를 재는 방법을 지정. 1일 경우 맨해튼 거리([https://bit.ly/man\\_distance](https://bit.ly/man_distance))를 사용하고, 2일 경우 유클리디안 거리([https://bit.ly/euc\\_distance](https://bit.ly/euc_distance))를 사용. 기본값은 2
    - n\_jobs 매개변수로 사용할 CPU 코어를 지정 가능. -1로 설정하면 모든 CPU 코어를 사용  
이웃 간의 거리 계산 속도를 높일 수 있지만 fit ( ) 메서드에는 영향이 없음. 기본값은 1
    - fit( )은 사이킷런 모델을 훈련할 때 사용하는 메서드. 처음 두 매개변수로 훈련에 사용할 특성과  
정답 데이터를 전달
    - predict( )는 사이킷런 모델을 훈련하고 예측할 때 사용하는 메서드. 특성 데이터 하나만 매개변수로 받음
    - score( )는 훈련된 사이킷런 모델의 성능을 측정. 처음 두 매개변수로 특성과 정답 데이터를 전달. 이 메서드는 먼저 predict ( )  
메서드로 예측을 수행한 다음 분류 모델일 경우 정답과 비교하여 올바르게 예측한 개수의  
비율을 반환

## SECTION 1-3 확인 문제(1)

- 1 데이터를 표현하는 하나의 성질로써, 예를 들어 국가 데이터의 경우 인구 수, GDP, 면적 등이 하나의 국가를 나타냄. 머신러닝에서 이런 성질을 무엇이라고 하나?
  - ① 특성                      ② 특질
  - ③ 개성                      ④ 요소
  
- 2 가장 가까운 이웃을 참고하여 정답을 예측하는 알고리즘이 구현된 사이킷런 클래스는?
  - ① SGDClassifier                      ② LinearRegression
  - ③ RandomForestClassifier                      ④ KNeighborsClassifier

## SECTION 1-3 확인 문제(1)

3 사이킷런 모델을 훈련할 때 사용하는 메서드는 어떤 것인가?

- ① predict( ) ② fit( )
- ③ score( ) ④ transform( )

4. 다음 중 모델의 정확도를 계산하는 올바른 방법은?

- ① (틀린 개수) / (전체 데이터 개수)
- ② (전체 데이터 개수) / (틀린 개수)
- ③ (정확히 맞힌 개수) / (전체 데이터 개수)
- ④ (전체 데이터 개수) / (정확히 맞힌 개수)

## SECTION 1-3 확인 문제(2)

5. 교재 56쪽에서 `n_neighbors`를 49로 설정했을 때 점수가 1.0보다 작았음. 즉 정확도가 100%가 아님. 그럼 `n_neighbors`의 기본값인 5부터 49까지 바꾸어 가며 점수가 1.0 아래로 내려가기 시작하는 이웃의 개수를 찾아보자.

이 문제를 위해 `KNeighborsClassifier` 클래스 객체를 매번 다시 만들 필요는 없음. 심지어 `fit()` 메서드로 훈련을 다시 할 필요도 없음. `k`-최근접 이웃 알고리즘의 훈련은 데이터를 저장하는 것이 전부이기 때문임.

`KNeighborsClassifier` 클래스의 이웃 개수는 모델 객체의 `n_neighbors` 속성으로 바꿀 수 있으며, 이웃 개수를 바꾼 후 `score()` 메서드로 다시 계산하기만 하면 됨

```
kn = KNeighborsClassifier()
kn.fit(fish_data, fish_target)
for n in range(5, 50):
    # k-최근접 이웃 개수 설정
    kn.n_neighbors =                      # 이 라인을 완성해 보세요
    # 점수 계산
    score = kn.score(                    ,                     ) # 이 라인을 완성해 보세요
    # 100% 정확도에 미치지 못하는 이웃 개수 출력
    if score < 1:
        print(n, score)
        Break
```