CHAPTER 1

인공지능 개요

인공지능 · Lecture Note

대전대학교 컴퓨터공학과
조교수 박상돈

목차 (Table of Contents)

학습 로드맵

교재는 크게 머신러닝편(제1장~제6장)과 딥러닝편(제7장~제10장)으로 나뉘어 있으며, 본 수업에서는 머신러닝편(CHAPTER 01~06)을 다룹니다.

장	제목	주요 내용
01	인공지능 개요	AI/ML/DL 개념, 코랜 세팅, 첫 번째 ML 프로그램 (k-최근접 이웃)
02	데이터 다루기	훈련 세트/테스트 세트 분리, 데이터 전처리
03	회귀 알고리즘과 모델 규제	선형 회귀, 릿지, 라쏘
04	다양한 분류 알고리즘	로지스틱 회귀, 확률적 경사 하강법
05	트리 알고리즘	결정 트리, 랜덤 포레스트, 그래디언트 부스팅
06	비지도 학습	군집, 차원 축소

딥러닝편(CHAPTER 07~10)은 본 수업 범위에 포함되지 않습니다. 딥러닝을 더 공부하고 싶은 분은 교재를 활용해 독학하거나, 상위 과목을 수강하시기 바랍니다.

> **✔ 학습 팁**
> 로드맵에서 중요한 점은, 순서대로 쌓아 올라간다는 것입니다. 01장을 제대로 이해해야 02장이 쉬고, 02장이 되어야 03장이 가능합니다. 초반에 기초를 잘 잡아두는 것이 핵심입니다.

SECTION 1-1 인공지능과 머신러닝, 딥러닝

1.1.1 인공지능이란

인공지능(Artificial Intelligence, AI)은 사람처럼 학습하고 추론할 수 있는 지능을 가진 컴퓨터 시스템을 만드는 기술입니다. 지능을 가진 로봇을 다룬 최초의 소설이 쓴 것은 약 150년 전의 일이며, 인공지능이라는 용어가 공식적으로 사용된 것은 1956년부터입니다.

1.1.2 인공지능의 역사 (AI 타임라인)

인공지능의 역사는 긴 기간에 걸쳐 "봄"과 "겨울"이 반복되어 왔습니다. 아래에서 주요 사건들을 살펴보겠습니다.

연도	사건	설명
1943	MCP 뉴런	월터 피츠와 워렌 매컬러가 인간 뇌의 뉴런을 수학적으로 모델링한 MCP 뉴런 모델을 제안. 입력 신호를 받아 일정 임계값을 넘으면 출력을 내보내는 구조로, 신경망의 출발점.
1950	튜링 테스트	앨런 튜링이 '기계가 생각할 수 있는가?' 논문을 발표. 사람이 기계와 대화했을 때 상대가 기계인지 구분할 수 없다면 그 기계는 지능이 있다고 볼 수 있다는 개념.
1956	다트머스 AI 컨퍼런스	미국 다트머스 대학에서 'Artificial Intelligence'라는 용어가 공식적으로 처음 사용된 컨퍼런스. AI의 공식적인 출발점.
1957	퍼셉트론	프랭크 로젠블랫트가 퍼셉트론을 발명. MCP 뉴런을 실제로 학습할 수 있게 만든 최초의 인공 신경망 모델. 1969년 마빈 민스키가 XOR 문제를 풀 수 없다는 한계를 증명.
1974-80	1차 AI 겨울	컴퓨터 성능 한계와 이론적 한계로 인해 연구 자금이 급격히 줄어든 시기.
1980-87	전문가 시스템	if-then 규칙으로 전문가의 지식을 프로그래밍한 시스템이 유행. 의사 진단, 법률 판단 등에 활용.
1987-93	2차 AI 겨울	전문가 시스템의 한계(규칙을 사람이 일일이 만들어야 하는 문제)로 인해 다시 침체기.
1998	LeNet-5	얀 르쿤(Yann LeCun)이 합성곱 신경망(CNN) LeNet-5를 만들어 손글씨 숫자 인식에 성공. 딥러닝 혁명의 씨앗.
2012	AlexNet	제프리 힌턴 팀이 AlexNet으로 ImageNet 대회에서 압도적 우승. 오류율을 26%→16%로 대폭 감소. 딥러닝 붐 시작.
2015	텐서플로	구글이 딥러닝 라이브러리 텐서플로(TensorFlow)를 오픈소스로 공개. 누구나 무료로 딥러닝 가능.
2016	알파고	이세돌 9단과 알파고의 대국. 딥러닝이 바둑이라는 인간의 영역마저 넘어섰. 한국 AI 붐의 기폭제.

2023	ChatGPT	2022년 말 출시된 ChatGPT로 일반 대중이 AI 기술을 직접 활용하는 시대 개막.

> ✔ **핵심 포인트**
> 인공지능의 역사를 보면 '겨울'과 '봄'이 반복됩니다. 현재는 AI의 전성기(여름)이지만, 역사는 과도한 기대 뒤에 실망이 올 수 있음을 보여줍니다. AI의 원리를 제대로 이해하면 유행에 흔들리지 않고 올바른 판단을 할 수 있습니다.

1.1.3 강인공지능 vs 약인공지능

강인공지능 (Strong AI) = 인공일반지능 (AGI)

영화에서 흔히 등장하는 인공지능으로, 사람과 구분하기 어려운 수준의 지능을 가진 컴퓨터 시스템입니다.

- 영화 <그녀(Her)>의 AI '사만다': 감정을 이해하고, 농담도 하고, 사랑까지 할 수 있는 AI
- 영화 <터미네이터>의 '스카이넷': 인류를 멸망시키려는 사악한 AI

핵심은 범용적이라는 점입니다. 한 가지만 잘하는 것이 아니라, 어떤 상황에서든 사람처럼 사고하고 판단할 수 있는 지능입니다. 현재 AGI는 아직 존재하지 않습니다.

약인공지능 (Weak AI)

현실에서 우리가 실제로 마주하고 있는 인공지능으로, 특정 분야에서 사람의 일을 도와주는 보조 역할을 합니다.

- 음성 비서: 시리, 빅스비, 구글 어시스턴트
- 자율 주행 자동차: 테슬라 오토파일럿
- 음악 추천: 스포티파이, 유튜브 뮤직
- 기계 번역: 구글 번역, 파파고

이세돌과 대국한 알파고도 약인공지능입니다. 바둑은 세계 최고 수준으로 두지만, '오늘 저녁 뭐 먹을까?' 질문에는 답하지 못합니다. 본 수업에서 배우는 머신러닝도 약인공지능의 영역입니다.

구분	강인공지능 (Strong AI)	약인공지능 (Weak AI)
능력 범위	범용적 (모든 분야)	특정 분야 한정
현재 존재 여부	아직 존재하지 않음	현재 우리가 사용 중
예시	영화 속 AI	알파고, ChatGPT, 시리

1.1.4 머신러닝이란

머신러닝(Machine Learning)은 **규칙을 일일이 프로그래밍하지 않아도 자동으로 데이터에서 규칙을 학습하는 알고리즘을 연구하는 분야**입니다.

인공지능의 하위 분야 중에서 지능을 구현하기 위한 소프트웨어를 담당하는 핵심 분야입니다.

전통적 프로그래밍 vs 머신러닝

전통적 프로그래밍: 프로그래머가 규칙을 하나하나 코드로 집니다.

```
if '무료' in 이메일 and '당첨' in 이메일:
    스팸
```

새로운 패턴이 나올 때마다 규칙을 추가해야 하며, 확장성이 없습니다.

머신러닝: 데이터를 주고 '니가 알아서 규칙을 찾아라'고 합니다.
알고리즘이 데이터를 분석해서 스스로 패턴을 발견합니다. 새로운 데이터만 추가하면 되므로 규칙을 수정할 필요가 없습니다.

머신러닝과 통계학

머신러닝은 통계학과 깊은 관련이 있습니다. 선형 회귀, 로지스틱 회귀 같은 알고리즘은 원래 통계학 기법에서 유래했습니다. R 프로그래밍 언어에도 많은 머신러닝 알고리즘이 구현되어 있습니다. 최근에는 통계나 수학 이론보다 경험적으로(empirically) 발전하는 경우도 많습니다.

사이킷런 (scikit-learn)

본 수업에서 가장 많이 사용할 파이썬 기반의 오픈소스 머신러닝 라이브러리입니다. 사이킷런이 나오기 전에는 머신러닝 기술이 대부분 폐쇄적이어서 전문 교육을 받거나 비싼 소프트웨어를 구매해야 했습니다. 사이킷런과 같은 오픈소스 라이브러리의 발전 덕분에 파이썬 코드를 다룰 수 있다면 누구나 머신러닝 알고리즘을 무료로 손쉽게 활용할 수 있게 되었습니다.

공식 사이트: scikit-learn.org

1.1.5 딥러닝이란

딥러닝(Deep Learning)은 **많은 머신러닝 알고리즘 중에 인공 신경망(Artificial Neural Network)을 기반으로 한 방법들의 통칭**입니다.

모든 딥러닝은 머신러닝이지만, 모든 머신러닝이 딥러닝은 아닙니다. '딥(Deep)'이라는 이름은 신경망의 층(layer)이 깊게 쌓여 있기 때문입니다. 보통 수십 층 이상이면 딥러닝이라고 부릅니다.

딥러닝의 주요 역사

- 1998 년, LeNet-5: 얀 르쿤이 만든 합성곱 신경망(CNN). 손글씨 숫자 인식에 성공.
- 2012 년, AlexNet: 제프리 힌턴 팀이 ImageNet 대회에서 압도적 우승. 딥러닝 연구 폭발적 증가의 계기.
- 2016 년, 알파고: 국내 AI 붐의 기폭제.
- 2022 년, ChatGPT: 대규모 언어 모델(LLM)이라는 새로운 패러다임을 개척.

본 수업에서는 딥러닝을 직접 다루지는 않지만, 머신러닝의 기초를 잘 잡아놓으면 딥러닝을 배울 때 훨씬 수월합니다. 데이터 준비, 훈련, 평가 같은 기본 과정은 동일하기 때문입니다.

1.1.6 인공 신경망 성능 발전의 원동력 세 가지

1. 풍부한 데이터: 인터넷과 스마트폰 시대가 열리면서 엄청난 양의 데이터(빅데이터)가 쌓였습니다. 딥러닝은 데이터가 많을수록 성능이 좋아지는 특성이 있습니다.
2. 컴퓨터 성능의 향상: 특히 GPU 의 발전이 결정적이었습니다. 게임 그래픽을 위해 만들어진 GPU 가 신경망의 행렬 연산을 매우 빠르게 처리할 수 있었습니다.
3. 혁신적인 알고리즘 개발: 드롭아웃(Dropout), 배치 정규화(Batch Normalization), 어텐션(Attention) 메커니즘 등 새로운 기법들이 지속적으로 개발되었습니다.

주요 라이브러리 정리

분야	대표 라이브러리	공개 연도	비고
머신러닝	사이킷런 (scikit-learn)	-	본 수업에서 사용
딥러닝	텐서플로 (TensorFlow)	2015	구글 오픈소스
딥러닝	파이토치 (PyTorch)	2018	메타(구 페이스북) 공개

이 라이브러리들의 공통점은 모두 파이썬 API를 제공한다는 점입니다.

1.1.7 Section 1-1 정리

인공지능 ⊃ 머신러닝 ⊃ 딥러닝

- 인공지능(AI): 사람처럼 학습하고 추론할 수 있는 지능을 가진 시스템을 만드는 기술
- 머신러닝(ML): 데이터에서 자동으로 규칙을 학습하는 알고리즘. 대표 라이브러리: 사이킷런
- 딥러닝(DL): 인공 신경망 기반 머신러닝. 대표 라이브러리: 텐서플로, 파이토치

SECTION 1-2 코랩과 주피터 노트북

1.2.1 구글 코랩이란

구글 코랩(Google Colab)은 **웹 브라우저에서 무료로 파이썬 프로그램을 테스트하고 저장할 수 있는 클라우드 기반의 주피터 노트북 개발 환경**입니다.

코랩을 사용하는 이유

4. 설치가 필요 없음: 모든 환경이 클라우드에 세팅되어 있어 웹 브라우저만 열면 바로 코딩 가능.
5. 핵심 라이브러리 기본 탑재: 사이킷런, 넘파이, 판다스, 맷플롯립 등이 이미 설치되어 있음.
6. 컴퓨터 사양에 상관없이 사용 가능: 구글 서버에서 코드를 실행하므로 노트북 성능에 영향받지 않음. GPU 도 무료로 사용 가능.
7. 구글 계정만 있으면 됨: Gmail 사용자는 이미 구글 계정이 있음.

코랩 접속 방법

크롬 브라우저에서 colab.research.google.com 접속 → 구글 계정으로 로그인

코랩 무료 사용 제한사항

항목	내용
서버 메모리	약 12GB
디스크 공간	약 225GB
동시 실행 노트북	최대 5개
연속 실행 시간	최대 12시간
유휴 시 자동 연결 끊김	약 90분

✔ **주의사항**
작업 중 중간중간 저장(Ctrl+S)하는 습관을 들이세요. 12시간 제한이나 유휴 시 자동 연결 끊김으로 인해 저장하지 않은 작업이 사라질 수 있습니다.

1.2.2 텍스트 셀과 코드 셀

코랩 노트북은 셀(cell)으로 구성됩니다. 셀은 코랩에서 실행할 수 있는 최소 단위입니다.

셀 종류	설명	용도
코드 셀	파이썬 코드를 작성하고 실행	프로그램 작성, 데이터 분석
텍스트 셀	설명이나 메모를 적는 공간 (마크다운 형식)	코드 설명, 문서화

코드 셀 실행 방법

단축키	동작
Ctrl + Enter	현재 셀 실행 (커서 그대로)
Shift + Enter	현재 셀 실행 후 다음 셀으로 이동
Alt + Enter	현재 셀 실행 후 새 셀 삽입

코드 셀의 특성: 마지막 줄의 값은 자동으로 출력됩니다(print() 없이도). 여러 줄을 한 셀에 넣으면 중간 반환값은 출력되지 않고 마지막 값만 출력됩니다.

1.2.3 코랜과 주피터의 관계

코랜은 구글이 주피터(Jupyter)를 커스터마이징한 것입니다. 주피터는 Julia + Python + R의 앞글자를 조합한 이름으로, 파이썬 지원으로 시작된 대화식 프로그래밍 환경입니다. 주피터 프로젝트의 대표 제품이 노트북(Notebook)이며, 코랜 노트북은 주피터 노트북의 구글 클라우드 버전입니다.

코랜 노트북은 구글 클라우드의 가상 서버(Virtual Machine)에서 실행되며, 노트북 파일의 확장자는 .ipynb (IPython Notebook)입니다.

1.2.4 새 노트북 만들기

8. [파일] → [새 노트]를 클릭하여 새로운 노트북을 생성합니다.
9. Untitled[number].ipynb 이름으로 생성되며, 빈 코드 셀이 하나 나타납니다.
10. 코드 셀에 print('Hello World')를 입력하고 Ctrl+Enter 로 실행합니다.
11. 노트북은 자동으로 구글 드라이브의 [Colab Notebooks] 폴더에 저장됩니다.
12. 제목을 클릭하여 노트북 이름을 변경할 수 있습니다.

저장된 노트북을 다시 열려면: 코랜에서 [파일] → [노트 열기] → [Google 드라이브] 탭을 선택하거나, 구글 드라이브에서 노트북 파일을 우클릭하여 [연결 앱] → [Google Colaboratory]를 선택합니다.

1.2.5 마크다운 문법

텍스트 셀에서는 간단한 기호를 사용해 텍스트에 서식을 적용하는 마크다운(Markdown)을 사용할 수 있습니다. HTML과 혼용도 가능합니다.

마크다운 형식	설명	결과
# 제목	<h1> 태그와 동일 (가장 큰 제목)	제목1
## 제목	<h2> 태그와 동일	제목2
### 제목	<h3> 태그와 동일	제목3
굵게	굵은 글씨	굵게
기울임골	기울임골	기울임골
~~취소선~~	취소선 추가	취소선
`코드`	백틱으로 감싼 코드 서체	print('hello')
> 들여쓰기	인용 블록	들여쓰기
* 목록 또는 - 목록	글머리 기호 목록	• 목록
[텍스트](URL)	하이퍼링크 생성	텍스트(링크)
$ 수식 $	LaTeX 수식 렌더링	y = ax + b

LaTeX(레이텍)은 수식, 그래프, 다이어그램 등을 그리는 데 유용한 문서 저작도구로, 논문 작성에 많이 사용됩니다. 달러 기호($)로 감싸면 수학 수식이 렌더링됩니다.

1.2.6 Section 1-2 확인 문제

Q1. 구글에서 제공하는 웹 브라우저 기반의 파이썬 실행 환경은?

→ **정답: ② 코랜**

Q2. 코랜 노트북에서 쓸 수 있는 마크다운 중 기울임골로 쓰는 것은?

→ **정답: ④ _혹공머신_ (underscoreundersccore 또는 *혹공머신*)**

Q3. 코랜 노트북은 어디에서 실행되는가?

→ **정답: ③ 구글 클라우드 (파일 저장은 구글 드라이브, 코드 실행은 구글 클라우드 컴퓨트 엔진)**

SECTION 1-3 마켓과 머신러닝

1.3.1 가상 시나리오와 문제 설정

교재에서는 '한빛 마켓'이라는 가상의 앱 마켓이 살아 있는 생선을 판매하는 시나리오를 설정합니다. 물류 센터 직원이 생선 이름을 외우지 못해 배송이 지연되는 문제를 머신러닝으로 해결합니다.

생선의 길이와 무게를 측정하면, 프로그램이 자동으로 '이건 도미입니다', '이건 빙어입니다' 하고 알려주는 것입니다. 이것이 바로 분류(classification) 문제이며, 머신러닝에서 가장 기본적인 문제 유형입니다.

사용할 데이터는 캐글(Kaggle)에 공개된 Fish Market 데이터셋이며, 도미와 빙어 2종류를 분류하는 이진 분류를 수행합니다.

1.3.2 전통적 프로그래밍 vs 머신러닝

전통적 프로그래밍 방식

프로그래머가 규칙을 직접 코드로 작성합니다.

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

문제점: 30cm보다 큰 생선이 모두 도미일까요? 아닙니다. 절대적인 기준을 정하기 어렵고, 새로운 생선 종류가 추가될 때마다 규칙을 계속 수정해야 합니다.

머신러닝 방식

보통 프로그램은 '누군가 정해준 기준대로 일'을 수행합니다. 반대로 머신러닝은 '누구도 알려주지 않는 기준을 찾아서 일'을 수행합니다.

도미 데이터와 빙어 데이터를 주면 머신러닝이 스스로 기준을 학습하고, 새로운 생선이 오면 해당 기준으로 도미인지 빙어인지 판별합니다.

1.3.3 데이터 준비하기

특성(Feature)의 개념

특성(feature)은 데이터를 표현하는 하나의 성질입니다. 이번 실습에서는 각 생선을 길이(cm)와 무게(g)라는 두 가지 특성으로 표현합니다.

도미 데이터 (35 마리)

```
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]
```

빙어 데이터 (14 마리)

```
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]
```

데이터 합치기

사이킷런을 사용하려면 도미와 빙어 데이터를 하나로 합쳐야 합니다.

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

2 차원 리스트 만들기

사이킷런은 각 생선의 [길이, 무게] 형태의 2차원 리스트를 입력으로 받습니다. zip() 함수와 리스트 내포(list comprehension) 구문을 사용하여 변환합니다.

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

결과: [[25.4, 242.0], [26.3, 290.0], [26.5, 340.0], ...]

정답(타겟) 데이터 준비

머신러닝이 학습하려면 정답도 함께 줘야 합니다. 이것을 지도 학습(supervised learning)이라고 합니다. 도미는 1, 빙어는 0으로 표현합니다.

```
fish_target = [1] * 35 + [0] * 14
```

결과: [1, 1, 1, ..., 1, 0, 0, 0, ..., 0] (1이 35개, 0이 14개)

1.3.4 산점도로 데이터 시각화

두 개의 특성(길이, 무게)을 각각 x축과 y축에 놓고, 각 생선을 점으로 표시하는 그래프를 산점도(scatter plot)라고 합니다. 파이썬에서는 맷플롯립(matplotlib) 패키지의 scatter() 함수를 사용합니다.

```
import matplotlib.pyplot as plt

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

결과 그래프에서 도미는 오른쪽 위(길고 무겁다), 빙어는 왼쪽 아래(짧고 가볍다)에 모여 있습니다. 도미의 산점도는 일직선에 가까운 형태로, 이런 경우를 선형(linear)적이라고 표현합니다.

1.3.5 첫 번째 머신러닝 프로그램

k-최근접 이웃(k-Nearest Neighbors) 알고리즘을 사용하여 도미와 빙어를 분류하는 모델을 만듭니다. 핵심 코드는 단 3줄입니다.

1 단계: 사이킷런에서 알고리즘 가져오기

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

2 단계: 모델 객체 생성

```
kn = KNeighborsClassifier()
```

3 단계: 훈련(Training)

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

fit() 메서드가 훈련을 담당합니다. '이 데이터(fish_data)와 이 정답(fish_target)으로 학습해라'고 명령하는 것입니다.

4 단계: 평가

```
kn.score(fish_data, fish_target)  # 결과: 1.0
```

score() 메서드로 모델 성능을 평가합니다. 1.0 = 49개 데이터를 모두 정확하게 분류함.

5 단계: 예측

```
kn.predict([[30, 600]])  # 결과: array([1]) → 도미
```

predict() 메서드로 새로운 데이터의 정답을 예측합니다. 입력은 반드시 2차원 리스트 형태여야 합니다.

✔ **핵심 패턴**
머신러닝의 기본 흐름: 모델 선택 → fit()()훈련) → score()(평가) → predict()(예측). 이 패턴은 사이킷런의 모든 알고리즘에 공통으로 적용됩니다.

1.3.6 k-최근접 이웃 알고리즘의 원리

핵심 원리: "주위의 다른 데이터를 보고, 다수를 차지하는 것을 정답으로 사용한다." (근묵자흑의 원리)

13. 새로운 데이터가 들어온다
14. 이 데이터와 기존 모든 데이터 사이의 거리를 계산한다
15. 가장 가까운 k 개의 데이터를 찾는다 (KNeighborsClassifier 의 기본값 k=5)
16. 그 k 개 중에서 다수결로 정답을 결정한다

특징과 단점

장점	단점
이해하기 쉽고 구현이 간단	데이터가 많으면 메모리 많이 필요
별도의 훈련 과정이 필요 없음	모든 데이터와의 거리 계산으로 시간 오래 걸림
데이터를 그대로 저장하는 것이 전부	데이터가 매우 많을 경우 사용하기 어려움

1.3.7 n_neighbors 매개변수와 확인 문제

k값을 기본값 5 대신 49(전체 데이터 수)로 설정하면 어떻게 될까요?

```
kn49 = KNeighborsClassifier(n_neighbors=49)
kn49.fit(fish_data, fish_target)
kn49.score(fish_data, fish_target)  # 결과: 0.714...
```

49개 전체를 참고하면 도미가 35개로 다수이므로 어떤 데이터를 넣어도 무조건 도미로 예측합니다. 정확도 = 35/49 ≈ 0.714. n_neighbors가 너무 크면 모델이 의미를 잃습니다.

도전 문제: 정확도가 떨어지는 이웃 개수 찾기

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

for n in range(5, 50):
    kn.n_neighbors = n
    score = kn.score(fish_data, fish_target)
    if score < 1:
```

```
        print(n, score)
        break
```

결과: n=18에서 처음으로 정확도가 1.0 아래로 떨어집니다.

확인 문제

Q1. 데이터를 표현하는 하나의 성질을 머신러닝에서 무엇이라 하는가?

→ **정답: ① 특성(feature)**

Q2. 가장 가까운 이웃을 참고하여 정답을 예측하는 사이킷런 클래스는?

→ **정답: ④ KNeighborsClassifier**

Q3. 사이킷런 모델을 훈련할 때 사용하는 메서드는?

→ **정답: ② fit()**

Q4. 모델의 정확도 계산 방법은?

→ **정답: ③ (정확히 맞힌 개수) / (전체 데이터 개수)**

핵심 용어 및 패키지 정리

핵심 용어

용어	영문	설명
특성	Feature	데이터를 표현하는 하나의 성질 (예: 길이, 무게)
훈련	Training	머신러닝 알고리즘이 데이터에서 규칙을 찾는 과정
모델	Model	알고리즘이 구현된 객체 (예: kn = KNeighborsClassifier())
정확도	Accuracy	정확히 맞힌 개수 / 전체 데이터 개수 (0~1)
타겟	Target	정답 레이블 (예: 도미=1, 빙어=0)
산점도	Scatter plot	두 특성을 x/y축에 놓고 각 데이터를 점으로 표시하는 그래프
분류	Classification	주어진 데이터의 카테고리를 나누는 문제
선형	Linear	데이터가 일직선에 가까운 형태를 따르는 관계

핵심 패키지와 함수

matplotlib

함수	설명	주요 매개변수
plt.scatter(x, y)	산점도 그리기	c: 색깔, marker: 마커 스타일
plt.xlabel(text)	x축 라벨 설정	
plt.ylabel(text)	y축 라벨 설정	
plt.show()	그래프 화면에 표시	

scikit-learn

클래스/메서드	설명	주요 매개변수
KNeighborsClassifier()	k-최근접 이웃 분류 모델	n_neighbors(기본:5), p(기본:2)
fit(특성, 정답)	모델 훈련	특성 데이터, 타겟 데이터
score(특성, 정답)	모델 성능 평가 (0~1)	특성 데이터, 타겟 데이터
predict(특성)	새 데이터의 정답 예측	특성 데이터만 (2차원)

fit()과 score()는 매개변수 2개(특성+정답), predict()는 1개(특성만)

이유: fit()은 정답을 보면서 학습하고, score()는 정답과 비교하지만, predict()는 정답을 모르는 상태에서 예측하기 때문.