December 10, 2025

▣ **강의명:** CS109A: 데이터 과학 입문

▣ **주차:** Lecture 18

▣ **교수명:** Pavlos Protopapas, Kevin Rader, Chris Gumb

▣ **목적:** Lecture 18의 핵심 개념 학습

□ 핵심 요약

이 문서는 CS109A 강의의 **결정 트리(Decision Tree)** 모델을 다룹니다. 결정 트리는 데이터를 분류하기 위해 "예/아니오" 질문을 반복하는, 마치 스무고개와 같은 직관적인 모델입니다.

- **왜 필요한가?**: 로지스틱 회귀 등 선형 모델이 잘 구분하지 못하는 복잡하고 비선형적인 데이터 경계를 쉽게 처리할 수 있습니다.
- **어떻게 작동하는가?**: "키가 6.5보다 큰가?" 같은 질문(규칙)을 통해 데이터를 반복적으로 분할하여, 각 영역이 하나의 클래스(예: 오렌지, 레몬)로만 구성되도록 합니다.
- **어떻게 학습하는가?**: 각 단계에서 데이터를 가장 '순수하게'(즉, 한 종류로만) 나누는 최적의 질문(분할 기준)을 **탐욕적(Greedy)**으로 찾습니다.
- **주요 과제**: 트리가 너무 복잡해져 훈련 데이터에만 과적합(Overfitting)되는 것을 방지하기 위해 **중지 조건**(예: 트리의 최대 깊이 제한)이 반드시 필요합니다.

Contents

1 개요: 왜 로지스틱 회귀가 아닌 결정 트리인가?

우리가 배운 로지스틱 회귀는 강력한 분류 모델이지만, 한계가 명확합니다. 로지스틱 회귀는 기본적으로 데이터 공간을 **하나의 선(또는 초평면)**으로 나누려고 시도합니다.

- **로지스틱 회귀가 잘하는 것**: 데이터가 선형적으로 잘 분리될 때 (예: 위도/경도 데이터에서 '농경지'와 '건조지'가 직선으로 명확히 나뉠 때)
- **로지스틱 회귀가 못하는 것**: 데이터의 경계가 복잡한 비선형(non-linear)일 때.

예를 들어, 위 (우) 이미지처럼 농경지(초록 점)가 중앙에 모여 있고 건조지(흰 점)가 그 주위를 둘러싸고 있는 경우, 로지스틱 회귀는 직선 하나로 이 두 클래스를 제대로 분리할 수 없습니다.

물론 다항 회귀(Polynomial Regression)를 사용해 $x_1^2 + x_2^2 - 0.25 = 0$ 과 같은 원형 경계를 만들 수도 있지만, 데이터의 경계가 더 복잡해지면 (예: 4사분면에 흩어져 있는 경우) 이를 설명할 방정식을 찾는 것은 거의 불가능에 가깝습니다.

새로운 모델의 요구사항 (Wish List) 우리는 다음과 같은 특징을 가진 새로운 모델이 필요합니다.

1. **복잡한 결정 경계**를 만들 수 있어야 합니다.
2. 모델의 결정 과정을 **이해하고 해석하기 쉬워야** 합니다. (Interpretability)
3. **계산이 효율적**이고 빨라야 합니다.

이 모든 것을 만족하는 모델이 바로 **결정 트리(Decision Tree)**입니다.

2 용어 정리

결정 트리를 이해하기 위해 사용되는 주요 용어들입니다.

Table 1: 결정 트리 핵심 용어

용어	쉬운 설명	원어	비고
결정 트리	데이터를 분류하기 위해 스무고개처럼 질문을 나열한 나무 구조의 모델	Decision Tree	
루트 노드	트리가 시작되는 첫 번째 질문 노드 (나무의 뿌리)	Root Node	트리에 단 하나만 존재
내부 노드	루트와 리프 사이의 모든 중간 질문 노드 (나무의 가지)	Internal Nodes	
리프 노드	트리의 가장 마지막에 위치한 노드로, 최종 결정을 의미 (나뭇잎)	Leaf Nodes	'터미널 노드'라고도 함
분할	하나의 노드(데이터 영역)를 질문을 통해 2개 이상의 자식 노드로 나누는 과정	Split	
순회	새로운 데이터가 주어졌을 때, 루트부터 리프까지 질문을 따라 내려가는 과정	Traversing the Tree	이 과정을 통해 예측 수행
순수도	특정 노드(영역)에 하나의 클래스만 존재하는 정도. (100% 순수 = 모두 같은 클래스)	Purity	불순도(Impurity)의 반대
불순도	특정 노드에 여러 클래스가 섞여 있는 정도. (예: 50:50으로 섞인 상태)	Impurity	Gini, Entropy로 측정
과적합	트리가 너무 복잡해져서 훈련 데이터의 '노이즈'까지 모두 암기한 상태	Overfitting	새 데이터에 대한 성능 저하
중지 조건	과적합을 막기 위해 트리 성장을 멈추는 규칙 (예: 최대 깊이 제한)	Stopping Conditions	하이퍼파라미터

3 핵심 원리 1: 결정 트리는 어떻게 작동하는가?

3.1 직관: 공학용 순서도 (Engineering Flowchart)

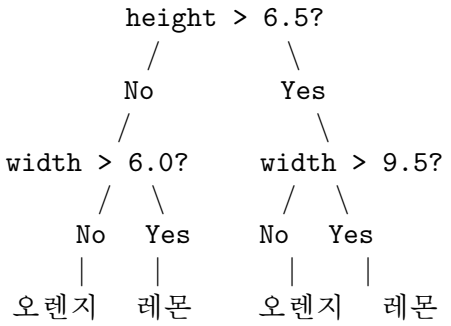
결정 트리는 우리가 일상에서 문제를 해결하는 방식과 매우 유사합니다. "공학용 순서도" 예시를 살펴봅시다.

이 순서도는 일련의 **이진(binary) 질문**을 통해 최종 결론(WD-40, 덕트 테이프, 문제 없음)에 도달합니다. 결정 트리는 이 아이디어를 데이터 분류에 그대로 적용합니다.

3.2 예측 (Prediction): 트리 순회 (Traversing the Tree)

결정 트리가 이미 "학습"되었다고 가정해봅시다. 예를 들어, 과일의 `height`(높이)와 `width`(너비)를 보고 '레몬'과 '오렌지'를 분류하는 트리입니다.

Figure 1: 결정 트리(좌)와 그로 인한 특징 공간 분할(우)



[2D 특징 공간이 4개 영역으로 분할된 그림]

좌측의 트리는 우측의 2D 특징 공간(Feature Space)을 **축에 평행한(axis-parallel) 직선**들로 분할하는 것과 같습니다. 각 리프 노드(최종 결정)는 특징 공간의 사각형 영역 하나에 해당합니다.

□ 예제:

새로운 과일 예측하기
새로운 과일이 들어왔습니다: (`height = 5.9, width = 5.8`)
이 과일은 레몬일까요, 오렌지일까요?
이 예측 과정을 **"트리 순회(Traversing the Tree)"**라고 부릅니다.

1. **루트 노드**: `height > 6.5`인가?
   - 5.9는 6.5보다 크지 않습니다. → **No** 브랜치로 이동합니다.
2. **내부 노드**: `width > 6.0`인가?
   - 5.8은 6.0보다 크지 않습니다. → **No** 브랜치로 이동합니다.
3. **리프 노드**: 최종 결정 노드에 도달했습니다.
   - 이 노드의 레이블은 **"오렌지"**입니다.

결론: 이 과일은 '오렌지'로 예측됩니다.

4 핵심 원리 2: 트리는 어떻게 "학습"하는가?

예측은 간단합니다. 하지만 `height > 6.5`나 `width > 6.0` 같은 "최적의 질문"은 어떻게 찾아내는 것일까요? 이것이 바로 결정 트리의 "학습" 과정입니다.

4.1 학습 목표: 영역 순수도 (Region Purity)

트리 학습의 목표는 각 분할(Split)을 통해 생성된 자식 노드(영역)가 최대한 **순수(Pure)**해지도록 하는 것입니다.

- **순수한(Pure) 노드**: 노드 안의 모든 데이터가 동일한 클래스에 속합니다. (예: 100% 오렌지) → 불확실성 낮음
- **불순한(Impure) 노드**: 여러 클래스가 섞여 있습니다. (예: 오렌지 50%, 레몬 50%) → 불확실성 높음

학습 알고리즘은 현재 노드를 분할할 수 있는 모든 가능한 질문(모든 특징, 모든 가능한 분할 값)을 테스트해보고, 그 결과로 만들어지는 두 자식 노드의 **평균 불순도를 가장 많이 낮추는** 질문을 선택합니다.

4.2 불순도 측정 기준 (Splitting Criteria)

불순도(Impurity)를 측정하는 방법에는 여러 가지가 있습니다.

4.3 기준 1: 분류 오류 (Classification Error)

가장 직관적인 방법입니다. 해당 노드에서 가장 많은 클래스를 정답으로 택했을 때, 틀리는 데이터의 비율입니다.

$$\text{분류 오류} = 1 - \max_k(\Psi(k|R))$$

여기서 $\Psi(k|R)$는 영역 R에서 k번째 클래스가 차지하는 비율입니다.

□ 예제:

영역 R_2에 **파란 원 5개, 주황 삼각형 8개**가 있습니다. (총 13개)

- 파란 원의 비율: $\Psi(\text{파랑}|R_2) = 5/13$
- 주황 삼각형의 비율: $\Psi(\text{주황}|R_2) = 8/13$

이 영역의 다수 클래스는 '주황 삼각형'입니다. 이 클래스로 예측하면 5개가 틀립니다.

분류 오류 $= 1 - \max(5/13, 8/13) = 1 - 8/13 = 5/13 \approx 0.38$

주의사항

분할 평가는 반드시 '가중 평균'으로 해야 합니다.

어떤 분할이 영역 R_1 (샘플 1개, 오류 0)과 R_2 (샘플 17개, 오류 0.18)를 만들었다고 가정합시다.

단순히 R_1의 오류가 0이라고 해서 이 분할이 좋은 것은 아닙니다. 샘플이 1개뿐인 영역은 의미가 없습니다.

따라서, 분할의 좋고 나쁨은 생성된 자식 노드들의 불순도를 **샘플 수로 가중 평균**하여 평가해야 합니다.

$$\text{가중 평균 오류} = \left(\frac{N_1}{N}\right) \times \text{Error}(R_1) + \left(\frac{N_2}{N}\right) \times \text{Error}(R_2)$$

(여기서 N은 총 샘플 수, N_1, N_2는 각 자식 노드의 샘플 수)

4.4 기준 2: 지니 불순도 (Gini Impurity)

결정 트리에서 가장 널리 사용되는 기준 중 하나입니다.

$$\text{Gini} = 1 - \sum_k \Psi(k|R)^2$$

이 식은 해당 영역에서 랜덤하게 2개의 샘플을 뽑았을 때, 두 샘플이 **서로 다른 클래스일 확률**을 의미합니다.

- **완전 순수** (예: 100% 주황): $1 - (1.0^2 + 0^2) = 0$
- **완전 불순** (예: 50% 파랑, 50% 주황): $1 - (0.5^2 + 0.5^2) = 1 - (0.25 + 0.25) = 0.5$

□ **예제:**

위와 동일하게 **파란 원 5개, 주황 삼각형 8개** (총 13개)가 있는 영역:
$\text{Gini} = 1 - \left[(5/13)^2 + (8/13)^2\right] = 1 - [25/169 + 64/169] = 1 - 89/169 \approx 0.47$

4.5 기준 3: 엔트로피 (Entropy)

정보 이론(Information Theory)에서 유래한 개념으로, 데이터의 **불확실성(무질서도)**을 측정합니다.

$$\text{Entropy} = -\sum_k \Psi(k|R) \log_2(\Psi(k|R))$$

- **완전 순수** (예: 100% 주황): $-[1.0 \log_2(1.0) + 0 \log_2(0)] = 0$
- **완전 불순** (예: 50% 파랑, 50% 주황): $-[0.5 \log_2(0.5) + 0.5 \log_2(0.5)] = 1$

엔트로피를 기준으로 분할을 평가할 때는 **정보 획득(Information Gain)**이라는 개념을 사용하며, 이는 "분할 전 엔트로피 - 분할 후 가중 평균 엔트로피" 입니다. 즉, 엔트로피를 가장 많이 줄이는 분할을 선택합니다.

4.6 불순도 지표 비교

Q: Gini와 Entropy는 왜 '분류 오류'보다 더 선호되나요? **A: 불순도 변화에 더 민감하기 때문입니다.**
'분류 오류'는 약간의 불순도 변화에 둔감합니다. 예를 들어, (50%:50%)인 노드를 (60%:40%)으로 개선해도 분류 오류는 여전히 0.4로, (70%:30%)로 개선해야 비로소 0.3으로 떨어집니다.
반면, **Gini와 Entropy**는 곡선 형태(Convex)이기 때문에, (50%:50%)에서 약간만 벗어나도 불순도 값이 민감하게 감소합니다.
결론: Gini와 Entropy는 노드를 "더 순수하게" 만들려는 동기가 '분류 오류'보다 강하기 때문에, 더 좋은 트리를 만드는 경향이 있습니다. 특히 **Entropy**가 불순도에 가장 민감하게(가장 가파르게) 반응합니다. (실무에서는 계산이 조금 더 빠른 Gini가 자주 쓰입니다.)

4.7 학습 알고리즘: 탐욕적 알고리즘 (Greedy Algorithm)

모든 가능한 트리 구조를 탐색하여 "전역 최적(Global Optimum)" 트리를 찾는 것은 **NP-complete 문제**로, 사실상 계산이 불가능합니다.

대신, 결정 트리는 **탐욕적 알고리즘(Greedy Algorithm)**을 사용합니다. "미래를 보지 않고, **지금 당장** 최선의 선택을 한다"는 의미입니다.

1. **시작**: 모든 데이터가 루트 노드 R에 있습니다.
2. **최적 분할 탐색**:
   - 모든 특징 p (예: height, width)에 대해,
   - 모든 가능한 분할 값 t (예: 6.0, 6.1, 6.5...)를 시도해봅니다.
   - 이 분할(p, t)이 만드는 두 자식 노드(R_1, R_2)의 **가중 평균 불순도**를 계산합니다.
3. **분할 수행**: 계산된 가중 평균 불순도를 **가장 많이 감소시키는** 최적의 특징(p^*)과 분할 값(t^*)을 선택하여 노드를 분할합니다.
4. **재귀 (Recurse)**:
   - 생성된 자식 노드 R_1과 R_2에 대해 2 3단계를 **재귀적으로 반복**합니다.
5. **중지**: 특정 **중지 조건**이 만족되면 분할을 멈추고 해당 노드를 리프 노드로 선언합니다.

5 핵심 원리 3: 과적합 방지 (Overfitting)

5.1 문제점: 멈추지 않는 트리

만약 트리가 멈추지 않고 계속 분할하도록 내버려 두면 어떻게 될까요? 트리는 훈련 데이터의 **모든 노이즈(noise)까지 암기**하기 시작합니다.

결국, 각 리프 노드에 **단 하나의 훈련 데이터 포인트**만 남을 때까지 트리가 성장합니다. 이 트리는 훈련 데이터에 대해서는 100%의 정확도를 보이지만, 실제 데이터(Test Data)에서는 노이즈까지 반영했기 때문에 성능이 매우 나빠집니다. 이를 **과적합(Overfitting)**이라고 부릅니다.

5.2 해결책: 중지 조건 (Stopping Conditions)

과적합을 방지하기 위해, 우리는 트리의 성장을 의도적으로 제한해야 합니다. 이를 "사전 가지치기(Pre-pruning)"라고도 하며, 다양한 **중지 조건 (하이퍼파라미터)**을 사용합니다.

주요 중지 조건 (Hyperparameters)

- **`max_depth` (최대 깊이)**: 트리가 몇 단계까지 내려갈 수 있는지 제한합니다. (예: '$\max_d epth = 3$') .
- **`min_samples_leaf` (리프 노드 최소 샘플 수)**: 리프 노드가 되기 위해 필요한 최소한의 샘플 수를 지정합니다. (예: '$\min_s amples_l eaf = 4$') , .
- **`max_leaf_nodes` (최대 리프 노드 수)**: 트리 전체의 리프 노드 개수를 제한합니다. (예: '$\max_l eaf_n odes = 5$')
- **`min_impurity_decrease` (최소 불순도 감소량)**: 분할을 통해 얻는 불순도 감소(정보 획득)가 이 임계값보다 커야만 분할을 수행합니다.
- **순수 노드**: 분할하려는 노드가 이미 100% 순수하다면(Gini=0) 더 이상 분할할 이유가 없으므로 중지합니다.

5.3 트리 성장 전략: Level-Order vs. Best-First

트리가 성장하는 방식에는 두 가지가 있습니다.

1. **Level-Order (수준별 성장)**:
   - 우리가 흔히 생각하는 방식입니다. 너비 우선 탐색(BFS)과 유사합니다.
   - Depth 1의 모든 노드를 분할하고, 그다음 Depth 2의 모든 노드를 분할하는 식으로 트리를 층(level)별로 완성해 나갑니다.
   - `max_depth`와 같은 대부분의 중지 조건에서 기본으로 사용됩니다.
2. **Best-First (최선 우선 성장)**:
   - `max_leaf_nodes` 중지 조건이 설정될 때 주로 사용됩니다.
   - 이 방식은 "어떤 노드를 분할하는 것이 지금 당장 불순도를 가장 많이 줄일까?"를 트리의 **모든 리프 노드**에 대해 비교합니다.
   - 즉, Depth 2에 있든 Depth 4에 있든 상관없이, 불순도 감소량이 가장 큰 노드를 "골라서" 분할합니다.
   - **단점**: 모든 가능한 다음 분할을 비교해야 하므로 **계산 비용이 매우 비쌉니다.**

5.4 편향-분산 트레이드오프 (Bias-Variance Trade-off)

트리의 복잡도(예: `max_depth`)는 모델의 편향(Bias)과 분산(Variance)에 직접적인 영향을 줍니다.

Table 2: 트리 깊이에 따른 편향-분산 트레이드오프

특징	**얕은 트리 (Shallow Tree)** (예: `max_depth = 4`)	**깊은 트리 (Deep Tree)** (예: `max_de`
모델 복잡도	낮음 (단순함)	높음 (복잡함)
편향 (Bias)	**높음 (High Bias)**	**낮음 (Low Bias)**
(설명)	데이터를 단순하게만 봐서 진짜 패턴을 놓침.	데이터의 미세한 패턴(노이즈 포함)
분산 (Variance)	**낮음 (Low Variance)**	**높음 (High Variance)**
(설명)	훈련 데이터가 조금 바뀌어도 트리가 거의 변하지 않음.	훈련 데이터가 조금만 바뀌어도 트리
결과	**과소적합 (Underfitting)**	**과적합 (Overfitting)**

5.5 최적의 하이퍼파라미터 찾기

그렇다면 `max_depth`는 4, 6, 100 중 무엇으로 정해야 할까요? 정답은 **데이터마다 다르다**이며, 이 최적의 값을 찾는 방법이 바로 **교차 검증 (Cross-Validation)**입니다.

우리는 `max_depth`를 2, 3, 4, ..., 10 등으로 바꿔가며 교차 검증을 수행하고, 검증 세트(Validation Set)에서 가장 성능이 좋았던 `max_depth` 값을 최종 모델의 하이퍼파라미터로 선택합니다.

6 학습 체크리스트

결정 트리 학습 점검표

- **동기 (Motivation)**
  - ☐ 로지스틱 회귀가 어떤 종류의 데이터(경계)에서 실패하는지 설명할 수 있는가?
  - ☐ 결정 트리가 로지스틱 회귀 대비 가지는 장점(복잡성, 해석력)을 아는가?
- **작동 원리 (Prediction)**
  - ☐ 루트, 내부, 리프 노드의 차이점을 아는가?
  - ☐ 새로운 데이터가 주어졌을 때 '트리 순회'를 통해 예측하는 과정을 시연할 수 있는가?
  - ☐ 트리의 분할이 2D 특징 공간에서 어떻게 '사각형 영역'으로 표현되는지 이해하는가?
- **학습 원리 (Training)**
  - ☐ 트리 학습의 목표가 '영역 순수도'를 높이는 것임을 아는가?
  - ☐ '순수도'와 '불순도'가 무엇을 의미하는지 설명할 수 있는가?
  - ☐ 3가지 불순도 지표(분류 오류, Gini, Entropy)의 공식을 아는가?
  - ☐ Gini와 Entropy가 분류 오류보다 선호되는 이유(민감도)를 설명할 수 있는가?
  - ☐ 분할 평가 시 '가중 평균' 불순도를 사용해야 하는 이유를 아는가?
  - ☐ 결정 트리 학습이 왜 '탐욕적(Greedy)' 알고리즘인지 설명할 수 있는가?
- **과적합 (Overfitting)**
  - ☐ 트리를 멈추지 않으면 왜 과적합이 발생하는지(노이즈 암기) 설명할 수 있는가?
  - ☐ 주요 중지 조건(예: `max_depth`, `min_samples_leaf`)의 역할을 아는가?
  - ☐ 'Level-order'와 'Best-first' 성장 전략의 차이점과 계산 비용을 아는가?
  - ☐ 트리 깊이(복잡도)와 편향-분산 트레이드오프의 관계를 설명할 수 있는가?
  - ☐ 최적의 중지 조건(하이퍼파라미터)을 어떻게 찾는지 아는가? (정답: 교차 검증)

7 FAQ (주요 질문 및 답변)

Q: Gini 불순도와 Entropy 중 무엇을 사용해야 하나요? A: 대부분의 경우 **결과는 비슷합니다.** Gini 불순도는 log_2 계산이 없는 단순 제곱 연산이라 **계산 속도가 조금 더 빠릅니다.** (이것이 Scikit-learn의 기본값인 이유입니다.) Entropy는 불순도에 이론적으로 더 민감하지만, 실제 성능 차이는 크지 않은 경우가 많습니다.

Q: 왜 '탐욕적(Greedy)' 알고리즘을 사용하나요? 모든 경우를 다 따져보면 더 좋지 않나요? A: 모든 경우의 수(모든 가능한 트리 구조)를 따져보는 것은 "전역 최적해"를 찾는 것을 의미하지만, 이 문제는 조합 폭발(combinatorial explosion)로 인해 **계산적으로 불가능(NP-complete)**합니다. 탐욕적 접근법은 비록 전역 최적해를 보장하지는 않지만, **매우 빠르고 현실적인 시간 안에 "충분히 좋은" 트리를 찾아냅니다.**

Q: `max_depth`와 `min_samples_leaf` 중 과적합 방지에 무엇이 더 효과적인가요? A: 둘 다 트리의 복잡도를 제어하는 중요한 하이퍼파라미터입니다.

- **`max_depth`**: 트리의 전체적인 '크기'와 '구조'를 직접적으로 제한합니다. 매우 강력한 규제 수단입니다.
- **`min_samples_leaf`**: 데이터의 '밀도'에 기반하여 분할을 제어합니다. 노이즈(이상치)가 리프 노드가 되는 것을 방지하여 모델을 더 안정적으로(robust) 만듭니다.

어느 것이 더 낫다고 말할 수 없으며, 두 하이퍼파라미터 모두 **교차 검증(Cross-Validation)**을 통해 데이터에 맞게 튜닝되어야 합니다.

8 1페이지 요약: 결정 트리 핵심

결정 트리(Decision Tree) 한눈에 보기

1. 모델 정의: 스무고개 (직관적)

- **아이디어**: "예/아니오" 질문(예: `height > 6.5`?)을 반복하여 데이터를 분류하는 나무 구조의 모델.
- **장점**:
  1. **해석 가능성 (White Box)**: 모델의 결정 과정을 사람이 쉽게 이해할 수 있음.
  2. **비선형 분류**: 로지스틱 회귀가 못하는 복잡한 결정 경계(원, 사각형 등)를 만들 수 있음.
- **예측**: '트리 순회(Traversing)' → 새로운 데이터가 루트부터 리프까지 질문을 따라 내려가며 최종 클래스 레이블을 할당받음.

2. 학습 원리: 순수도 찾기 (탐욕적)

- **목표**: 각 노드(영역)가 최대한 '순수하게'(한 클래스로만 구성) 되도록 분할.
- **학습 방식**: **탐욕적(Greedy)** 알고리즘 → 지금 당장 불순도를 가장 많이 줄이는 최적의 (특징, 분할 값)을 찾아 분할하고 재귀 반복.
- **불순도 측정 (Splitting Criteria)**:
  - **Gini Index** (기본값): $1 - \sum p_k^2$ (계산이 빠름)
  - **Entropy**: $-\sum p_k \log_2(p_k)$ (불순도에 가장 민감함)

3. 주요 과제: 과적합 방지 (규제)

- **문제**: 중지 조건이 없으면 트리가 무한정 성장하여 훈련 데이터의 '노이즈'까지 모두 암기함 (**과적합: High Variance, Low Bias**).
- **해결 (Stopping Conditions)**: 트리의 성장을 의도적으로 제한 (사전 가지치기).
  - `max_depth`: 트리의 최대 깊이 제한.
  - `min_samples_leaf`: 리프 노드가 가져야 할 최소 샘플 수.
- **튜닝**: 최적의 하이퍼파라미터(예: 최적의 '$\max_d epth$') **교차 검증(Cross-Validation)** .