📌 4.1 분류의 개요

1) 분류

- **정의**: 지도학습의 한 유형으로 입력 데이터의 피처(feature)와 레이블(label)을 학습하여 새로운 데이터의 레이블을 예측하는 방법.
- **목적**: 기존 데이터의 패턴을 학습 → 새로운 데이터가 어떤 레이블에 속하는지 판단.

대표적인 분류 알고리즘

- **나이브 베이즈**: 통계적 확률 모델 기반
- **로지스틱 회귀**: 선형 관계를 활용한 분류
- **결정 트리**: 규칙 기반의 트리 구조
- **서포트 벡터 머신**: 클래스 간 최대 분리 마진 탐색
- **최근접 이웃**: 거리 기반 분류
- **신경망**: 심층 연결 구조 기반 학습
- **앙상블**: 여러 모델을 결합한 방법

---

2) 앙상블

- **정의**: 서로 다른(또는 같은) 알고리즘을 결합해 예측 성능을 향상시키는 방법.
- **특징**: 단일 모델보다 더 높은 성능 제공 → 이미지, 음성, NLP 등 다양한 분야에서 활용.
- **종류**:
  - **배깅**: 병렬적으로 여러 모델 학습 (예: Random Forest)
  - **부스팅**: 약한 학습기를 순차적으로 학습하여 강력한 모델 생성
  - **스태킹**: 여러 알고리즘의 예측을 메타 모델이 종합

---

3) 주요 앙상블 기법

- **랜덤 포레스트**: 배깅의 대표 기법. 빠른 학습과 높은 예측 성능 제공
- **그레디언트 부스팅**: 순차적으로 학습하여 높은 정확도 달성, 하지만 시간이 오래 걸림
- **XGBoost / LightGBM**: 기존 부스팅을 개선해 속도와 성능을 모두 강화한 최신 기법
- **스태킹**: 서로 다른 모델을 결합해 최종 예측 수행

---

4) 앙상블의 장단점

- 장점

- 단일 모델보다 높은 예측 성능
- 다양한 알고리즘의 강점을 결합 가능
- 복잡한 데이터 분류 문제에 강함

- 단점

- 계산 비용 증가, 학습 시간 오래 걸림
- 모델 구조가 복잡해 해석이 어려움
- 과적합 가능성 존재

📌 4.2 결정트리

1) 기본 개념과 직관

- **정의**: 데이터에 숨어 있는 **규칙**을 자동으로 찾아 **트리 형태의 분류 규칙**을 만드는 지도학습 알고리즘.
- **아이디어**: 모델이 가장 잘 구분되는 규칙 조건을 순차적으로 찾아가며 트리를 만든다.
- **구성 요소**
  - **루트 노드**: 첫 분기 지점(가장 중요한 질문).
  - **결정 노드(규칙 노드)**: 어떤 조건으로 데이터를 **분할** 하는 지점.
  - **리프 노드**: 더 이상 분할하지 않고 **최종 클래스**를 내리는 지점.
  - **서브트리**: 특정 규칙 조건을 만족하는 데이터 하위 집합으로 형성된 트리 조각.

깊이: 루트부터 리프까지의 최대 길이. 깊을수록 더 복잡한 규칙/높은 분해능을 갖지만 **과적합** 위험이 커짐.

---

2) "좋은 분할"이란? : 균일도

- 트리는 **각 분기에서 클래스가 최대한 한쪽으로 몰리도록** 데이터를 나누어야 예측력이 높아진다.
- 예시 직관
  - 세 집합 A/B/C 중 "C가 가장 균일(거의 한 색), 다음이 B, 마지막이 A(혼합 많음)"이라고 하면 트리는 매 분기에서 C처럼 **균일도가 높아지도록** 조건을 찾는다.

핵심 원리: **정보가 가장 많이 늘어나고(혼잡도 ↓), 자식 노드가 더 순수**해지는 조건을 고르는 것이 좋은 분할.

---

3) 분할 기준(불순도/정보량 측도)

트리는 매 단계에서 "어떤 속성/기준으로 나눌지"를 **지표**로 평가한다.

3-1. 엔트로피와 **정보 이득**

- **엔트로피**: 혼란/무질서의 정도. 클래스 분포가 섞일수록 큼.
- **정보 이득**: 분할 전후의 엔트로피 감소량(얼마나 더 뚜렷해졌는가).
- 전략: **IG가 가장 큰 속성/문턱값**으로 분할.

3-2. 지니 불순도

- **지니 계수(지니 불순도)**: 무작위로 뽑은 두 표본이 서로 다른 클래스로 뽑힐 확률. 낮을수록 순도↑.
- 실무에서는 계산이 빠르고 성능이 좋아 "CART 구현(sklearn 기본)"에서 많이 사용.
- 전략: **지니가 가장 작아지도록** 분할.

요약: **엔트로피→정보이득 최대화**, **지니→불순도 최소화**. 목적은 동일: **자식 노드의 순도 최대화.**

---

4) 학습 절차(알고리즘 흐름)

1. **현재 노드의 데이터가 모두 같은 클래스**인지 확인 → 같다면 **리프 노드**로 결정.
2. 아니라면, **가장 좋은 분할 기준**을 찾는다(정보이득 최대 혹은 지니 최소).
   - **연속형 변수**는 여러 후보 임계값을 시험해 최적 문턱을 고른다.
   - **범주형 변수**는 가능한 분할(집합)을 시험한다.

선택된 기준으로 데이터를 **분할**하여 **브랜치/자식 노드** 생성.

각 자식 노드에 대해 1-3을 **재귀적으로** 반복.
다음 **정지 조건** 중 하나를 만족하면 더 이상 분할하지 않음:

- 최대 깊이 도달,
- 노드 내 표본 수가 너무 적음,
- 더 이상 순도 개선이 의미 없음(정보이득≈0),
- 리프 최소 표본 수 미만 등.

---

5) 결정 트리의 특징, 장단점

장점

- **직관적·해석 용이**: 규칙이 if/else로 읽혀 **설명 가능성**이 높음.
- **비선형/복잡한 경계**도 잘 표현(충분히 깊으면 거의 어떤 결정 경계도 근사).
- **피처 스케일링/정규화 불필요**: 분기 기준이 순서/문턱 기반이라 스케일 영향이 작음.
- **결측·범주형 처리 용이**(원리상 가능; 구현마다 차이).

단점

- **과적합**에 매우 취약(깊고 복잡한 트리).
  → 사전 제한(프리프루닝) 또는 사후 가지치기(pruning)가 필요.
- 데이터 작은 변화에도 트리 구조가 크게 달라질 수 있어 **불안정**할 수 있음.
- 단일 트리는 SOTA 성능이 제한적 → **랜덤포레스트/부스팅** 등 앙상블이 자주 사용됨.

---

6) scikit-learn의 결정 트리(CART)와 주요 하이퍼파라미터

- 구현: DecisionTreeClassifier(분류) / DecisionTreeRegressor(회귀)
  기본 아이디어는 **CART(Classification And Regression Trees)**.

자주 쓰는 파라미터

- **min_samples_split**
  - 한 노드를 **분할할 최소 표본 수**.
  - 너무 작으면 분기를 남발(과적합↑), 너무 크면 분할이 멈춰 **과소적합** 가능.

min_samples_leaf

- **리프 노드에 필요한 최소 표본 수**.
- 값을 키우면 **각 리프가 충분히 크도록** 강제되어 과적합 방지, 경계가 매끄러워짐.
- 클래스 불균형 시 **소수 클래스 리프가 사라질 수 있음**에 유의.

max_features

- 각 분기에서 고려할 **피처의 최대 개수/비율**.
- 값이 작을수록 무작위성↑(편향↑, 분산↓) → 앙상블(랜덤포레스트)에서 효과적.
- 지정 방식: 정수(개수), 실수(비율), "sqrt", "log2", None(전부) 등.

max_depth

- 트리의 **최대 깊이**. 과적합을 억제하는 대표 파라미터.
- 너무 작으면 과소적합, 너무 크면 과적합.

max_leaf_nodes

- **리프 개수 상한**. 리프 수로 모델 복잡도를 직접 제한.

실전 팁

- 먼저 max_depth/min_samples_leaf로 **복잡도를 제어**하고, 성능이 부족하면 min_samples_split/max_features를 탐색.
- 교차검증으로 적정 복잡도(바이어스-분산 균형)를 찾는 게 핵심.

7) 과적합 방지(프루닝/사전 제한)

- **사전 제한**: 학습 중 미리 제동
  - max_depth, min_samples_leaf, min_samples_split, max_leaf_nodes, max_features.

사후 가지치기: 완성된 트리를 잘라 단순화
  - 비용복잡도 가지치기 등.

효과: **일반화 성능 향상**, 해석성 개선, 과적합 완화.

8) 시각화(트리 해석)

- 트리는 규칙이 명확하므로 **시각화**하면 이해가 빠름.
- 방법
  - export_graphviz로 Graphviz 포맷을 내보내 **.dot → 이미지** 렌더링.
  - 혹은 plot_tree/export_text로 파이썬 내부에서 간단히 확인.

Graphviz는 원래 C/C++ 기반 그래프 툴로, 시스템에 설치 후 파이썬에서 래퍼를 통해 사용 가능.
설치 후 export_graphviz(model, out_file=..., feature_names=..., class_names=...) 형태로 내보내면 규칙이 시각적으로 표현됨.

9) 데이터 구성과 분할의 질

- **균일도가 높은(한 클래스가 우세한) 자식 노드**를 만드는 분할이 좋다.
- 불균형/혼합이 심한 노드는 정보가 부족하여 다음 분할에서 **추가 질문(깊이)을 더 요구**한다.
- 따라서 매 스텝에서 **정보량을 가장 많이 늘리는 질문**을 선택하는 것이 트리의 핵심 전략.

📌 4.3 앙상블 학습

1) 앙상블 학습 개요

- 앙상블 학습은 여러 개의 분류기를 만들어 그 결과를 **결합**하여 더 정확한 최종 예측을 얻는 방법.
- 여러 전문가의 의견을 종합해 더 신뢰도 높은 결론을 내리는 것과 유사.
- 단일 분류기보다 **높은 예측 성능**을 보이는 것이 목적.

---

2) 앙상블 학습의 활용

- 이미지, 영상, 음성 같은 비정형 데이터 분류에서 높은 성능.
- 정형 데이터에서도 앙상블은 단일 모델보다 성능이 좋음.
- 대표적 기법:
  - **랜덤 포레스트**: 배깅 방식
  - **그레디언트 부스팅**: 부스팅 방식
  - **XGBoost, LightGBM**: 최신 부스팅 계열, 빠르고 성능 뛰어남
  - **스태킹**: 여러 모델의 결과를 종합하여 최종 모델 생성

---

3) 앙상블 학습의 유형

1. **보팅(Voting)**
   - 여러 분류기의 예측을 투표하여 최종 결정을 내림.
   - 서로 다른 알고리즘의 분류기를 조합할 수 있음.
   - 데이터 샘플은 같지만 학습기는 다름.

배깅(Bagging)

- 같은 분류기를 사용하지만 데이터 샘플을 **중복 추출(부트스트랩)** 하여 학습.
- 대표적 기법: **랜덤 포레스트**.

부스팅(Boosting)

- 약한 학습기를 순차적으로 학습시켜 점점 성능을 높이는 방식.
- 오차가 큰 데이터에 가중치를 주어 학습.
- 대표적 기법: **그레디언트 부스팅, XGBoost, LightGBM**.

---

4) 보팅 방식의 종류

- **하드 보팅**
  - 각 분류기가 낸 최종 예측값 중 **다수결 원칙**으로 결정.

소프트 보팅

- 분류기들이 출력한 **확률**을 평균 내어 가장 확률이 높은 클래스를 선택.
- 일반적으로 소프트 보팅이 더 좋은 성능을 보임.

---

5) 보팅 학습 흐름

- 여러 분류기를 동시에 학습시켜 예측을 수행한 후, 최종 결과를 **투표 방식**으로 결정.
- 하드 보팅은 다수결로, 소프트 보팅은 예측 확률을 평균하여 결과를 냄.
- 예시:
  - 하드 보팅: 4개 분류기 중 3개가 “1번 클래스” 예측 → 최종 결과는 1번.
  - 소프트 보팅: 각 분류기가 낸 확률 평균값을 계산 → 가장 확률 높은 클래스로 최종 결정.

6) 앙상블 학습의 장단점

장점

- 단일 모델보다 **높은 성능**.
- 다양한 알고리즘을 조합 가능.
- 복잡한 데이터에도 강력한 성능 발휘.

단점

- 계산량 증가, 학습 시간이 길어짐.
- 모델이 복잡해 해석이 어려움.

📌 4.4 랜덤 포레스트 정리

1) 랜덤 포레스트의 개요 및 원리

- **랜덤 포레스트**는 배깅 방식을 사용하는 대표적인 앙상블 알고리즘이다.
- 개별 결정 트리를 여러 개 학습시켜 보팅으로 최종 결과를 결정한다.
- 개별 트리의 단점(과적합)을 줄이고 높은 성능을 얻을 수 있다.
- 각 트리는 데이터 전체가 아닌 **부트스트래핑(중복을 허용한 샘플링)**을 통해 생성된 데이터로 학습한다.

2) 랜덤 포레스트의 구조

1. 전체 학습 데이터에서 여러 개의 **부분 집합**(샘플 데이터)을 생성한다.
2. 각 부분 집합을 이용해 **개별 결정 트리**를 학습한다.
3. 학습된 여러 개의 트리 예측을 **보팅(다수결 또는 평균)**을 통해 최종 예측 결과를 도출한다.

3) 랜덤 포레스트의 장점

- 단일 결정 트리에 비해 **과적합 위험이 줄어듦**.
- 다양한 데이터 샘플을 이용하기 때문에 **일반화 성능이 뛰어남**.
- 계산 속도가 빠르고, 다양한 분야에서 활용 가능.

4) 랜덤 포레스트 하이퍼 파라미터

- **트리 개수 (n_estimators)**
  - 학습에 사용할 트리의 개수. 많을수록 성능은 좋아질 수 있으나 시간이 오래 걸림.

최대 특성 수 (max_features)

- 분할 시 고려할 최대 특성 수. 트리마다 다른 특성을 사용하여 다양성을 확보한다.

최대 깊이 (max_depth)

- 트리의 최대 깊이. 깊어질수록 복잡해지고 과적합 위험이 커짐.

노드 분할 최소 샘플 수 (min_samples_split)

- 노드가 분할되기 위한 최소 샘플 수.

리프 노드 최소 샘플 수 (min_samples_leaf)

- 리프 노드가 되기 위한 최소 샘플 수. 값이 커지면 모델이 단순해져 과적합이 줄어든다.

5) 하이퍼 파라미터 튜닝

- 그리드 탐색 기법을 통해 다양한 파라미터 조합을 실험해 최적 값을 찾는다.
- 멀티 코어 환경에서는 병렬 처리를 통해 빠르게 수행 가능하다.

6) 성능 평가

- 랜덤 포레스트는 일반적으로 높은 정확도를 보이며, 단일 결정 트리보다 안정적인 성능을 낸다.
- 또한 학습 후 **특성 중요도(feature importance)**를 제공하여 어떤 변수가 예측에 중요한 역할을 했는지 알 수 있다.

📌 5 서포트 벡터 머신 정리

0) 개요

- 서포트 벡터 머신은 **강력한 분류·회귀·이상치 탐지**에 쓰이는 지도학습 방법이야.
- 핵심 아이디어: 두 집단을 **가장 넓은 여유폭(마진)** 으로 가르는 **결정 경계**를 찾는 것.
- 경계를 실제로 **결정하는 데이터**는 경계에 딱 붙어 있는 소수의 점들인데, 이를 **서포트 벡터(지지점)** 라고 해.
- 데이터가 복잡해도 비교적 적은 표본·중간 규모의 특성 수에서 튼튼하게 작동하고, 특히 **분류 경계가 뚜렷한 문제**에 강해.

5.1 선형 분류 (직선/평면으로 나눌 수 있는 경우)

5.1-1. 최대 마진 원리

- 많은 분류기 중에서 **여유폭이 가장 넓은 경계**를 고르면, 새로운 데이터에도 **일반화**가 잘 되는 경향이 있어.
- 경계에 멀리 떨어진 학습 표본을 몇 개 더 추가하더라도 **경계는 거의 바뀌지 않고,**
  경계에 닿아 있는 소수의 표본(서포트 벡터) 가 경계를 좌우해.

5.1-2. 특성 스케일(단위)에 민감

- 각 특성의 범위가 크게 다르면, 특정 축 방향으로 **도로(마진) 폭**이 왜곡되어 **경계가 불리하게 기울어질 수 있어**.
- 그래서 **표준화/정규화 같은 스케일 맞춤**을 해 주면 경계 품질이 확 좋아진다.

5.1.1 완전 분리 vs 위반 허용

하드 마진(완전 분리)

- 모든 학습 표본이 **경계의 바깥쪽**에 있어야 함(위반 허용 안 함).
- **이상치에 매우 민감**하고, 완벽히 직선으로 나눌 수 있을 때만 가능 → 실제 데이터엔 잘 맞지 않는 경우가 많아.

소프트 마진(위반 허용)

- 일부 표본이 경계 안쪽으로 들어오는 **위반**을 허용하고, 대신 **벌점**을 부여해 균형을 맞춘다.
- **규제 강도 C**

- **큰 값**: 위반을 강하게 벌함 → 경계가 **딱 맞추는 쪽(마진 좁음)** 으로 가서 **과적합** 위험 ↑
- **작은 값**: 위반을 더 허용 → 경계가 **완만(마진 넓음)** 해져 **과소적합** 위험 ↑

실제 문제에선 **소프트 마진**이 기본 선택지야.

5.2 비선형 분류 (직선/평면으로는 안 나뉘는 경우)

비선형 문제를 다루는 대표 방법 두 가지:

방법 A) 특성 추가(다항식 등)

- 원래 특성으로는 직선 분리가 안 되더라도, **새로운 특성(예: 제곱, 곱)** 을 추가하면
  확장된 공간에서는 직선으로 깔끔히 나뉠 수 있어.
- 하지만 차수를 높일수록 차원이 급격히 커져 **계산량·과적합** 위험이 함께 커진다.

방법 B) 커널 기법(핵 함수)

- 실제로 특성을 만들지 않고도, **특성 공간에서의 내적**을 **함수로 바로 계산**하여
  확장된 공간에서 선형 분리를 흉내 내는 방법.
- 장점: **많은 특성을 실제로 생성하지 않아도** 복잡한 경계를 만들 수 있음.

5.2.1 다항식 커널

- 원 공간의 직선을 **확장 공간의 곡선 경계**로 바꿔 주는 효과.
- **차수**가 높을수록 훨씬 복잡한 경계를 만들 수 있으나, **과적합** 위험도 함께 증가.
- 데이터가 단순하면 낮은 차수, 복잡하면 적절히 높은 차수로 **균형**을 잡아야 해.

5.2.2 유사도 특성의 관점

- 또 다른 시각: 각 표본이 특정 **기준점(랜드마크)** 과 **얼마나 가까운지**를 나타내는 값을 **유사도 특성**으로 쓰는 것.
- 모든 학습 표본을 기준점으로 삼으면, 차원이 매우 커지지만 **복잡한 경계를 표현**할 수 있다.
- 대표적인 유사도 함수가 **가우시안 방사 기저 함수**(종 모양).
  - **감마(γ)** 는 종의 **폭**을 조절하는 값.
    - **감마 큼**: 영향 범위가 매우 좁아져 **경계가 울퉁불퉁(세밀)** → **과적합** 위험 ↑
    - **감마 작음**: 영향 범위가 넓어져 **경계가 완만(매끈)** → **과소적합** 위험 ↑

요약: 비선형 문제에선 **가우시안 방사 기저 함수**가 가장 널리 쓰이고,
감마(경계의 세밀함)와 **C**(위반 허용 정도)가 **함께 균형**을 이뤄야 성능이 잘 나온다.

5.2.3 가우시안 방사 기저 함수 커널의 실전 튜닝 감각

- **감마**는 "경계의 곡선 정도(지역성)"를, **C**는 "위반 허용"을 조절.
- 보통은 **감마(작음↔큼) × C(작음↔큼)** 격자처럼 여러 조합을 **교차검증**으로 비교해 최적을 찾는다.
- 일반적인 경향
  - 복잡한 경계 필요(데이터가 뒤섞임) → **감마↑, C도 어느 정도↑**
  - 과적합 기미(경계가 들쭉날쭉) → **감마↓** 또는 **C↓**

- 너무 둔함(경계가 너무 완만) → **감마↑** 또는 **C↑**

5.2.4 계산 복잡도(학습 시간·메모리 감각)

- **선형형(커널 미사용)**
  - 대략 **표본 수 × 특성 수**에 비례하는 규모로 학습(반복 횟수에 따라 선형적으로 증가).
  - **특성이 매우 많은 문제**에서 특히 유리하고, 데이터가 엄청 커도 비교적 빠르다.

커널형(다항식·가우시안 등)

  - 표본 수가 늘수록 **표본 수의 제곱~세제곱 수준**까지 학습 시간이 커질 수 있어,
    큰 데이터에서는 느리고 메모리도 많이 든다.
  - 반면 **복잡한 경계**를 훨씬 잘 표현한다.

7.2.1 배깅과 페이스팅

- **공통점**: 동일한 알고리즘을 여러 번 학습시켜 앙상블을 만드는 방식.
- **차이점**
  - **배깅**: 훈련 샘플을 **중복 허용**하여 무작위 추출.
  - **페이스팅**: 훈련 샘플을 **중복 없이** 무작위 추출.

효과:

  - 여러 예측기를 학습시킨 후 결과를 모아 최종 예측(분류 → **최빈값**, 회귀 → **평균값**).
  - 개별 예측기보다 편향은 비슷하지만, **분산 감소** 효과.
  - 병렬 학습 가능 → CPU/GPU 자원 활용 효율적.

7.2.1 사이킷런의 배깅/페이스팅

- **BaggingClassifier / BaggingRegressor** API 제공.
- 주요 매개변수:
  - bootstrap=True → 배깅, bootstrap=False → 페이스팅
  - max_samples: 각 예측기에 사용될 샘플 수 비율(0~1)
  - n_jobs: 병렬 처리 시 CPU 코어 개수 지정 (-1 → 모든 코어 사용)

실험 결과: 단일 결정트리보다 배깅 앙상블이 훨씬 일반화 잘됨(분산 작아짐).

7.2.2 OOB (Out-of-Bag) 평가

- 배깅 시 어떤 샘플은 훈련에 선택되지 않음(평균적으로 약 37%).
- 이 샘플들을 **oob 샘플**이라 부르고, 검증용으로 활용 가능.
- 별도의 검증 세트 없이 **일종의 교차 검증 역할** 수행.
- 사이킷런: oob_score=True 옵션으로 자동 평가 가능.

7.3 랜덤 패치와 랜덤 서브스페이스

- **랜덤 패치**: 샘플과 특성 모두를 무작위로 선택해 학습.
- **랜덤 서브스페이스**: **특성만** 무작위 선택해 학습.
- 관련 매개변수:
  - max_features (특성 수 제한),
  - bootstrap_features=True (특성 중복 허용 샘플링).

장점:

- 고차원 데이터(예: 이미지)에서 효과적.
- 예측기의 다양성 증가 → 편향 조금 늘지만, 분산 크게 감소 → 일반화 성능 향상