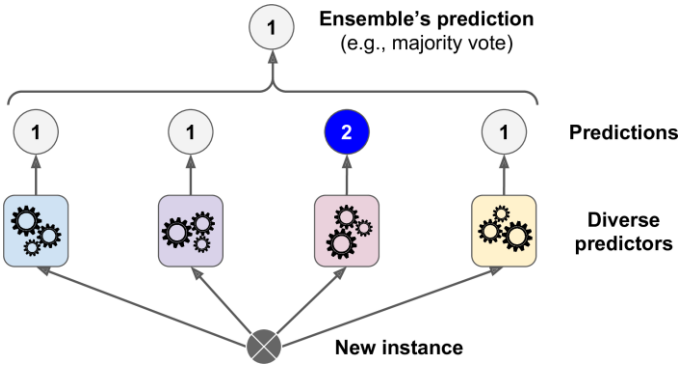
chapter 7
앙상블 학습과 랜덤 포레스트

앙상블 학습 ensemble learning : 일련의 예측기로부터 예측을 수집하여 가장 좋은 모델 하나보다 더 좋은 예측을 얻음.

7.1. 투표 기반 분류기



분류기 여러 개를 훈련한다. → 각 분류기의 예측을 모아서 가장 많이 선택된 클래스를 예측

직접 투표 hard voting 분류기 : 다수결 투표로 정해지는 분류기

각 분류기가 **약한 학습기 weak learner** 일지라도 충분하게 많고 다양하다면 앙상블은 **강한 학습기 strong learner**가 될 수 있다.

같은 데이터로 훈련시킬 경우 분류기들이 같은 종류의 오차를 만들기 쉽기 때문에 잘못된 클래스가 다수인 경우가 많고 앙상블의 정확도가 낮아진다.

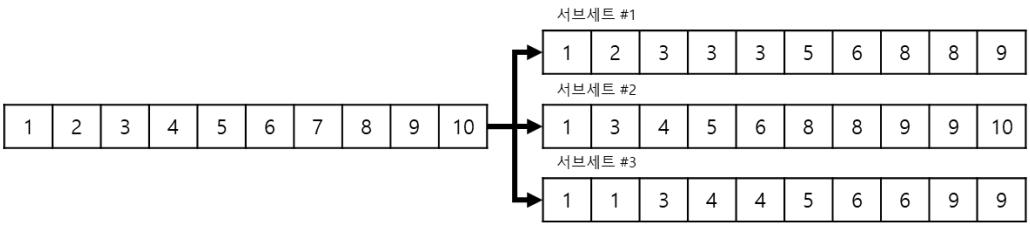
앙상블 방법은 예측기가 가능한한 서로 독립적일 때 최고의 성능 발휘.
각기 다른 알고리즘으로 학습하여 다양한 분류기를 얻으면 다른 오차를 만들 가능성이 높기 때문에 정확도 향상
→ 분류기가 독립적이고 오차에 상관관계가 없어야 한다.

간접 투표 soft voting 분류기 : 개별 분류기의 예측(확률)을 평균 내어 확률이 가장 높은 클래스를 예측
→ 모든 분류기가 클래스의 확률을 예측할 수 있어야 한다, 보통 더 많이 사용

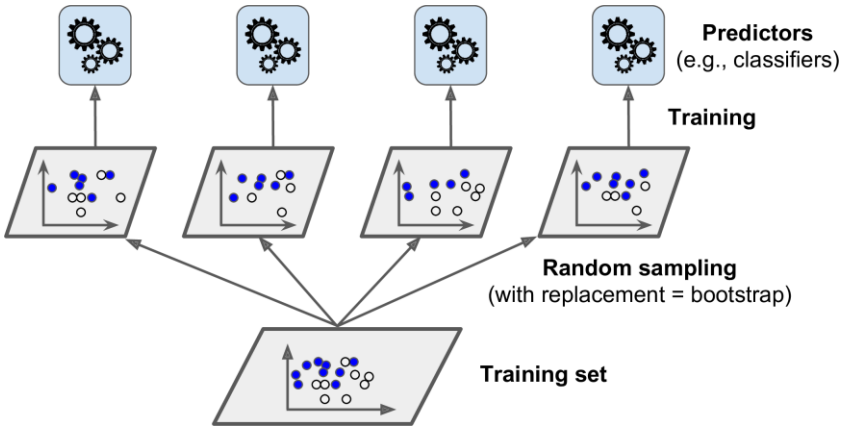
7.2. 배깅과 페이스팅

같은 알고리즘을 사용하고 훈련 세트의 서브셋을 무작위로 구성하여 분류기를 각기 다르게 학습시킴.
배깅 bagging (bootstrap aggregating) : 훈련 세트에서 중복을 허용하여 샘플링
→ 한 예측기를 위해 같은 훈련 샘플을 여러 번 샘플링할 수 있다.

<부트스트래핑 샘플링 방식>



페이스팅 pasting : 중복을 허용하지 않고 샘플링 하는 방식



병렬로 학습, 예측 가능 → 인기가 높다

무작위 샘플링후에 모든 예측기가 훈련을 마치면 앙상블은 모든 예측기의 예측을 모아서 새로운 샘플에 대한 예측을 만듬.

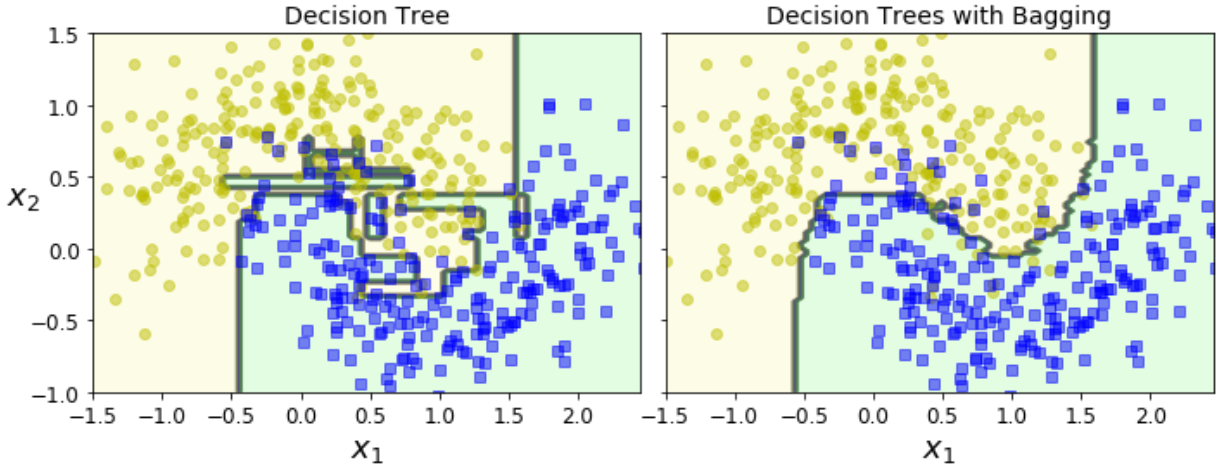
수집함수는 전형적으로 분류일 때 → **통계적 최빈값** statistical mode

회귀일 때 → 평균을 계산

개별 예측기는 원본 훈련 세트로 훈련시킨 것보다 훨씬 크게 편향되어 있지만

→ 수집함수를 통과하면 편향과 분산이 모두 감소(↓)

일반적으로 앙상블의 결과 → (원본 데이터셋 & 하나의 예측기에 비해) 편향은 비슷, 분산은 감소(↓)



앙상블은 비슷한 편향에서 더 작은 분산을 만든다. (훈련 세트의 오차 수가 비슷, 결정 경계는 덜 불규칙)

부트스트래핑 : 각 예측기가 학습하는 서브셋에 다양성을 증가시킴 → 배깅이 페이스팅보다 편향이 조금더 높다.

다양성을 추가한다는 것은 상관관계를 줄이므로 앙상블의 분산을 감소시킨다.

전반적으로 배깅이 더 나은 모델을 만들기 때문에 더 선호한다.

(여유가 있다면 모두 평가해서 더 나은쪽을 선택하는 것이 좋다.)

보팅 Voting 과 배깅 Bagging 의 차이점

1. 보팅 : 일반적으로 서로 다른 알고리즘을 가진 분류기를 결합
2. 배깅 : 각각의 분류기가 모두 같은 유형의 알고리즘 기반이지만, 데이터 샘플링을 서로 다르게 가져가면서 학습을 수행해 보팅을 수행하는 것.

7.2.2 oob 평가

배깅을 사용하면 어떤 샘플은 한 예측기를 위해 여러 번 샘플링되고 어떤 것은 전혀 선택되지 않을 수 있다.
선택되지 않은 훈련 샘플을 **oob** **out-of-bag** 샘플이라 한다. (평균 37%)
훈련에 oob 샘플을 사용하지 않으므로 oob 샘플을 사용해 평가한다 (각 예측기의 oob 평가를 평균함)

7.3. 랜덤 패치와 랜덤 서브스페이스

(샘플이 아닌)특성 샘플링 조절 → 각 예측기는 무작위로 선택한 입력 특성의 일부분으로 훈련됨
→ 더 다양한 예측기를 만들며 편향을 늘리는 대신(↑), 분산을 낮춘다(↓)
매우 고차원의 데이터셋을 다룰 때 유용 ex) 이미지

랜덤 패치 방식 **random patches method** : 훈련 특성과 샘플을 모두 샘플링하는 것
랜덤 서브스페이스 방식 **random subspace** : 훈련 샘플을 모두 사용하고 특성을 샘플링 하는 것

7.4 랜덤 포레스트 random forest

랜덤 포레스트 : 일반적으로 배깅 방법(또는 페이스팅)을 적용한 결정 트리의 앙상블

랜덤 포레스트 알고리즘은 트리의 노드를 분할할 때 전체 특성 중에서 최선의 특성을 찾는 대신 무작위로 선택한 특성 후보 중에서 최적의 특성을 찾는 식으로 무작위성을 더 주입함.
→ 트리를 더욱 다양하게 만들고, (다시 한번) 편향을 손해보는 대신 분산을 낮추어 더 훌륭한 모델을 만듬

7.4.1 엑스트라 트리 extra-trees

랜덤 포레스트에서 트리를 만들 때 각 노드는 무작위로 특성의 서브셋을 만들어 분할에 사용한다.
트리를 더 무작위하게 만들기 위해 (보통의 결정 트리처럼) 최적의 임곗값을 찾는 대신 후보 특성을 사용해 무작위로 분할한 다음 그중에서 최상의 분할을 선택한다.

익스트림 랜덤 트리 앙상블 **extremely randomized trees ensemble** (또는 줄여서 **엑스트라 트리** **extra-trees**)
위와 같이 극단적으로 무작위한 트리의 랜덤포레스트
편향이 늘어나지만(↑) 대신 분산을 낮춤(↓)
모든 노드에서 특성마다 가장 최적의 임곗값을 찾는 것이 트리 알고리즘에서 가장 시간이 많이 소요되는 작업 중 하나 → 일반적 랜덤포레스트보다 엑스트라 트리가 훨씬 빠름(↑)

7.4.2 특성 중요도

랜덤 포레스트는 특성의 상대적 중요도를 측정하기가 쉽다.
DecisionTreeClassifier의 특성 중요도 : 일부 특성을 완전히 배제
무작위성이 주입된 RandomForestClassifier : 거의 모든 특성에 대해 평가할 기회를 가짐

사이킷런은 어떤 특성을 사용한 노드가 (랜덤 포레스트에 있는 모든 트리에 걸쳐서) 평균적으로 불순도를 얼마나 감소시키는지 확인하여 특성의 중요도를 측정한다.
더 정확히 말하면 가중치 평균이며 각 노드의 가중치는 연관된 훈련 샘플 수와 같다.

훈련이 끝난 뒤 특성마다 자동으로 이 점수를 계산하고 중요도의 전체 합이 1이 되도록 결과값을 정규화

→feature_importances_ 변수에 저장
특성을 선택해야 할 때 어떤 특성이 중요한지 빠르게 확인할 수 있어 매우 편리함.

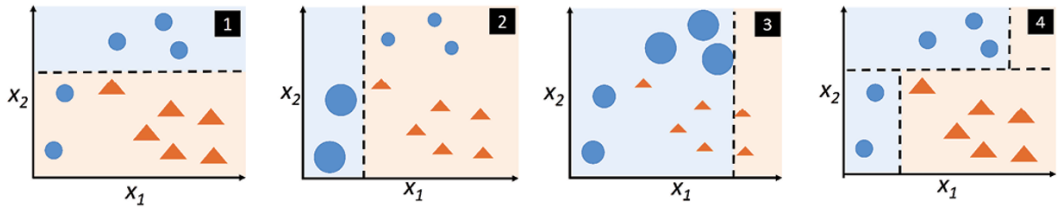
결정 트리의 특성 중요도
노드에 사용된 특성별로
(현재 노드의 샘플 비율 × 불순도) – (왼쪽 자식 노드의 샘플 비율 × 불순도) – (오른쪽 자식 노드의 샘플 비율 × 불순도)
와 같이 계산하여 더하고, 특성 중요도 합이 1이 되도록 전체 합으로 나누어 정규화한다.
샘플 비율 : 트리 전체 샘플 수에 대한 비율
랜덤 포레스트의 특성 중요도는 각 결정 트리의 특성 중요도를 모두 계산하여 더한 후 트리수로 나눈 것.

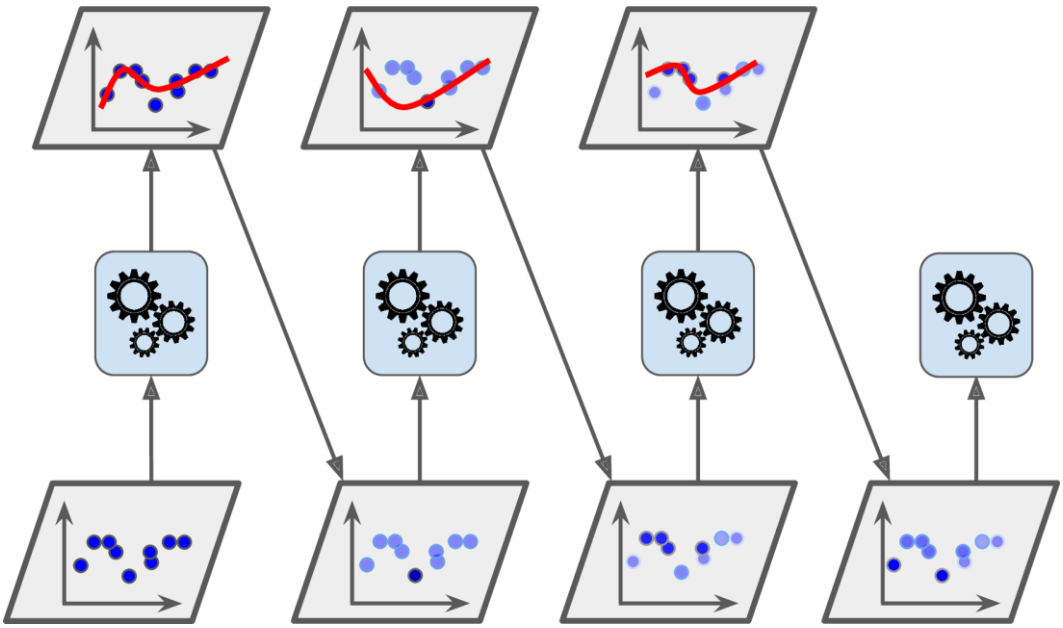
7.5. 부스팅

부스팅 boosting : 약한 학습기를 여러 개 연결하여 강한 학습기를 만드는 앙상블 방법
아이디어 : 앞의 모델을 보완해나가면서 일련의 예측기를 학습시키는 것

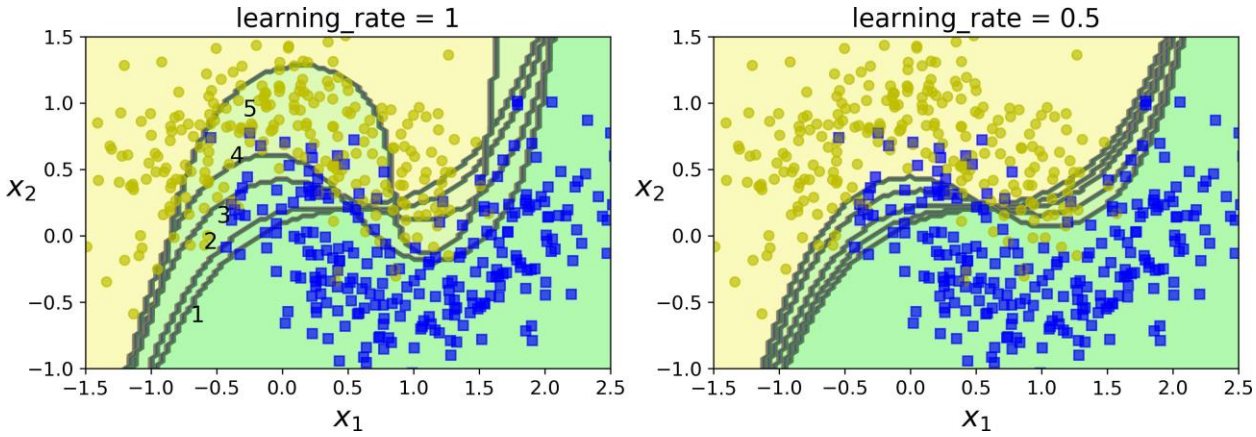
7.5.1. 에이다부스트 AdaBoost (Adaptive Boosting)

이전 모델이 과소적합했던 훈련 샘플의 가중치를 더 높이는 것.
→ 학습하기 어려운 샘플에 점점 더 맞춰지게 된다.





경사 하강법은 비용 함수를 최소화하기 위해 한 예측기의 모델 파라미터를 조정해가는 반면 에이다부스트는 점차 더 좋아지도록 앙상블에 예측기를 추가한다.



오른쪽 : 학습률을 반으로 낮춤 → 잘못 분류된 샘플의 가중치는 반복마다 절반 정도만 높아진다.

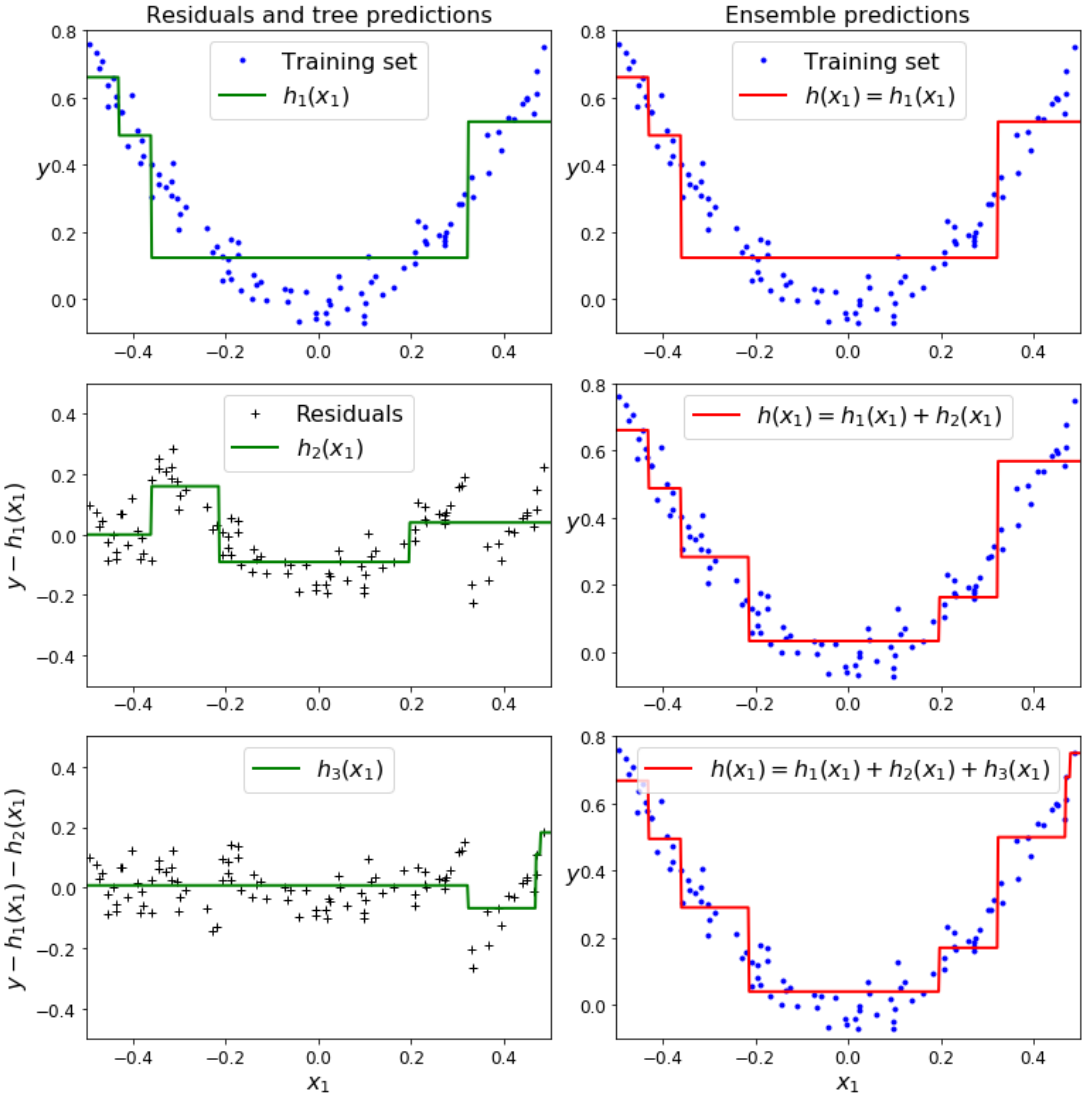
7.5.2. 그레이디언트 부스팅 gradient boosting

앙상블에 이전까지의 오차를 보정하도록 예측기를 순차적으로 추가한다.

→ 에이다부스트처럼 반복마다 샘플의 가중치를 수정하는 대신 이전 예측기가 만든 **잔여 오차** residual error에 새로운 예측기를 학습시킨다.

그레이디언트 트리 부스팅 gradient tree boosting 또는 **그레이디언트 부스티드 회귀 트리** gradient boosted regression tree (GBRT)

→결정 트리를 기반 예측기로 사용하는 회귀



왼쪽 열 : 트리의 예측, 오른쪽 열 : 앙상블의 예측

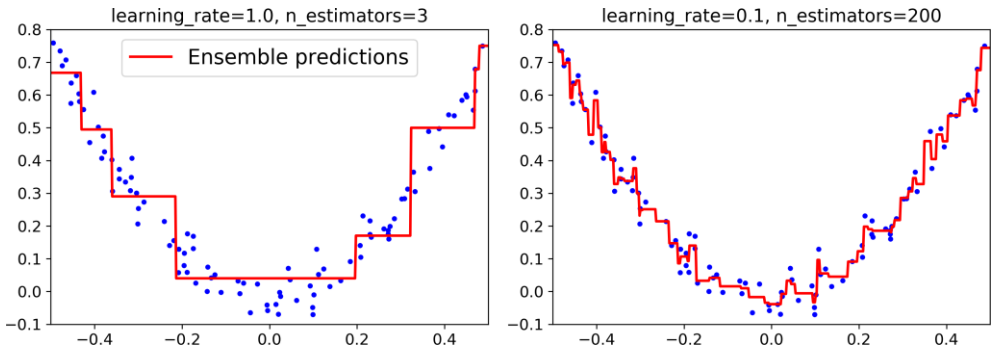
새로운 트리가 전 트리의 잔여오차에 대해 학습되면서, 트리가 앙상블에 추가될수록 앙상블의 예측이 점차 좋아

진다.

GradientBoostingRegressor

learning_rate 매개변수 : 각 트리의 기여 정도를 조절한다.

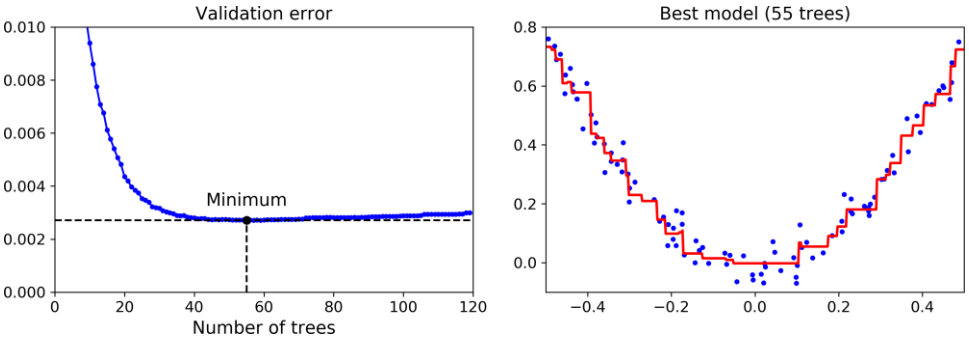
→ 낮게 설정(=축소 shrinkage) : 앙상블을 훈련 세트에 학습시키기 위해 많은 트리가 필요, 예측의 성능은 좋아짐



왼쪽 : 훈련 세트를 학습하기에는 트리가 충분하지 않다.

오른쪽 : 트리가 너무 많아 훈련 세트에 과대적합되었다.

최적의 트리 수 찾기 → 조기 종료 기법을 사용할 수 있다.



GradientBoostingRegressor는 subsample 매개변수를 통해 훈련시 사용할 훈련 샘플의 비율을 지정할 수 있다.

예를 들어 subsample=0.25라고 하면 각 트리는 무작위로 선택된 25%의 훈련샘플로 학습된다.

편향이 높아진다(↑), 분산이 낮아지다(↓), 훈련 속도를 높인다(↑)

→ 이를 **확률적 그레이디언트 부스팅** **stochastic gradient boosting**이라고 한다.

최적화 그레이디언트 부스팅 구현으로 XGBoost extreme gradient boosting 이 유명하다

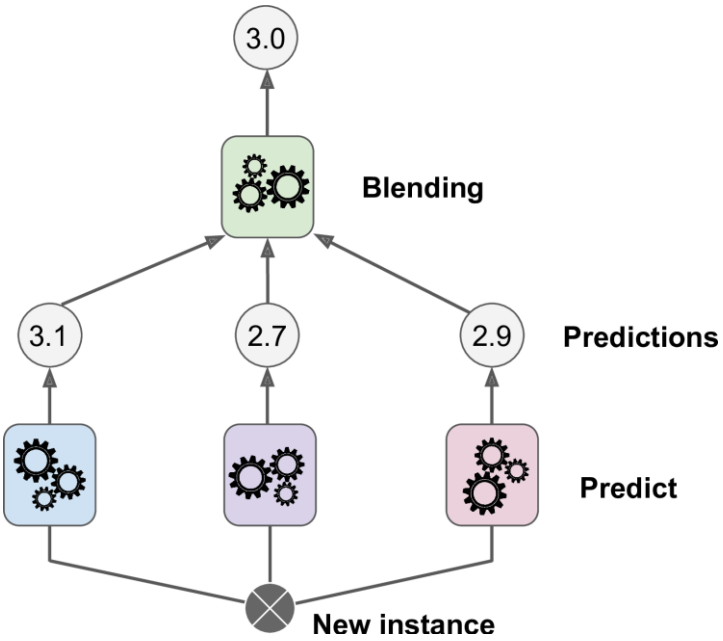
7.6. 스태킹 stacking (stacked generalization)

스태킹(Stacking), 배깅(Bagging), 부스팅(Boosting) 공통점: 개별적인 여러 알고리즘을 서로 결합해 예측 결과를 도출
그러나 부스팅은 개별 알고리즘으로 예측한 데이터를 기반으로 다시 예측을 수행한다는 것이 다르다.

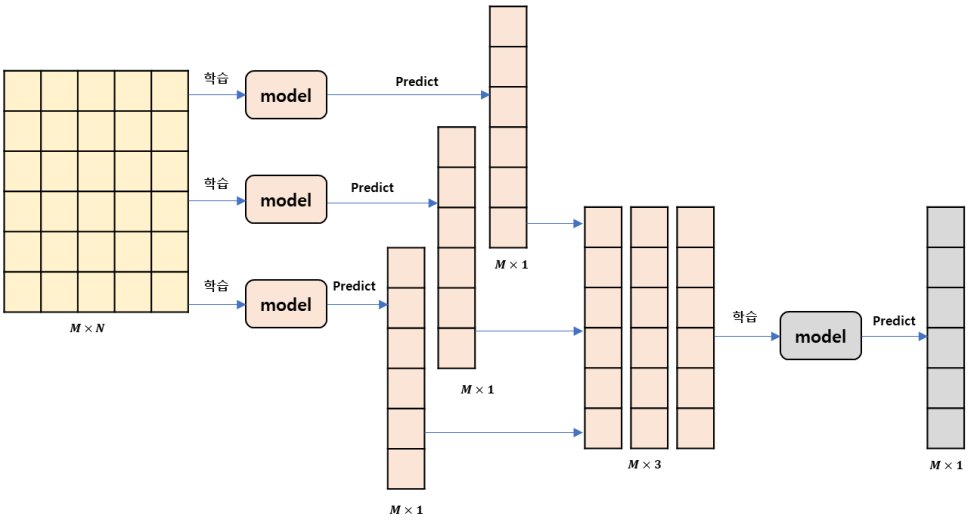
즉 개별 알고리즘의 예측 결과 데이터 세트를 최종적인 메타 데이터 세트로 만들어 별도의 ML 알고리즘으로 최종 학습을 수행하고 테스트 데이터를 기반으로 다시 최종 예측을 수행하는 방식.

(이렇게 개별 모델의 예측된 데이터 세트를 다시 기반으로 하여 학습하고 예측하는 방식을 메타모델이라 한다.)

스태킹 모델은 두 종류의 모델이 필요하다
1. 개별적인 기반 모델
2. 이 개별 기반 모델의 예측 데이터를 학습 데이터로 만들어서 학습하는 최종 메타 모델
→ 스태킹 모델의 핵심 : 여러 개별 모델의 예측 데이터를 각각 스태킹 형태로 결합해 최종 메타 모델의 학습용 피처 데이터 세트와 테스트용 피쳐 데이터 세트를 만드는 것이다.

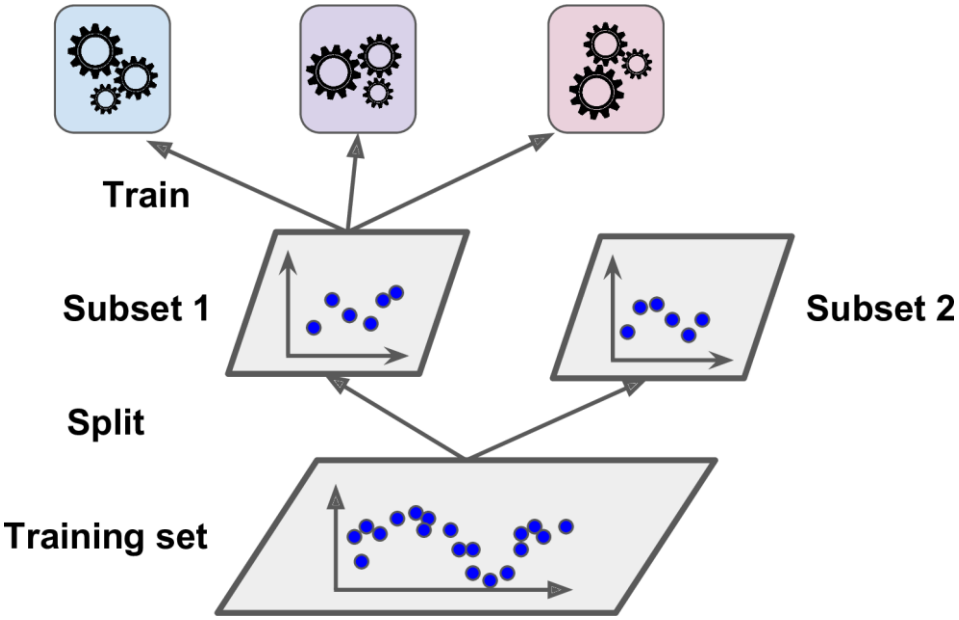


새로운 샘플에 회귀작업을 수행하는 앙상블
세 예측기가 각각 다른 값(3.1, 2.7, 2.9)을 예측
→마지막 예측기(**블렌더 blender** 또는 **메타 학습기 meta learner**)가 이 예측을 입력으로 받아 최종 예측(3.0)을 만듦

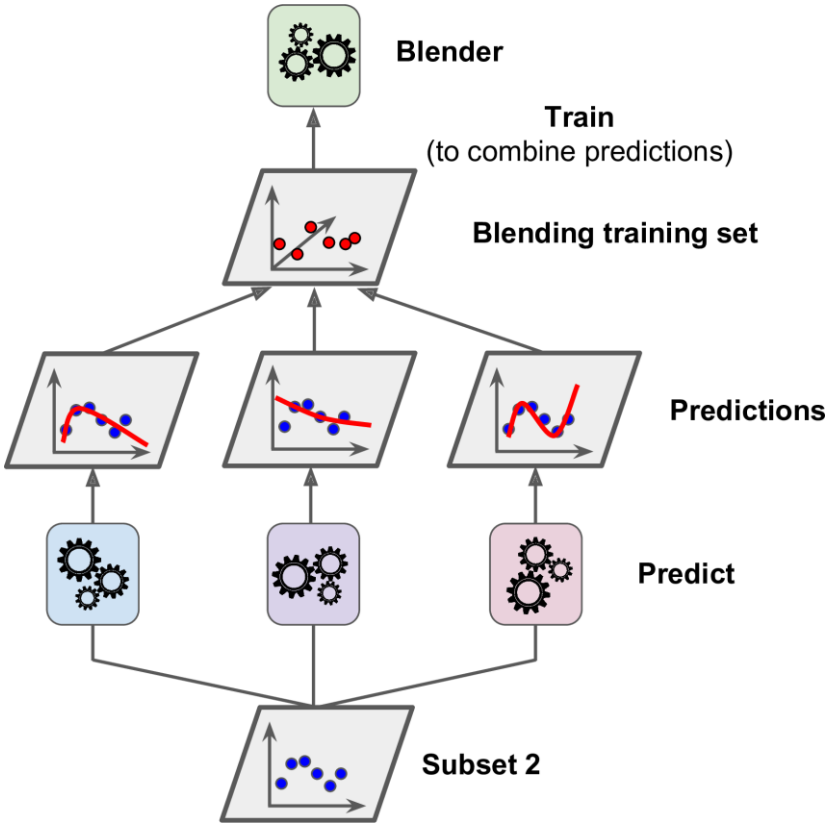


블렌더의 학습

일반적인 방법은 **홀드 아웃** hold-out 세트를 사용하는 것



1. 훈련 세트를 두 개의 서브셋으로 나눈다.
첫 번째 서브셋은 첫 번째 레이어의 예측을 훈련시키기 위해 사용된다.

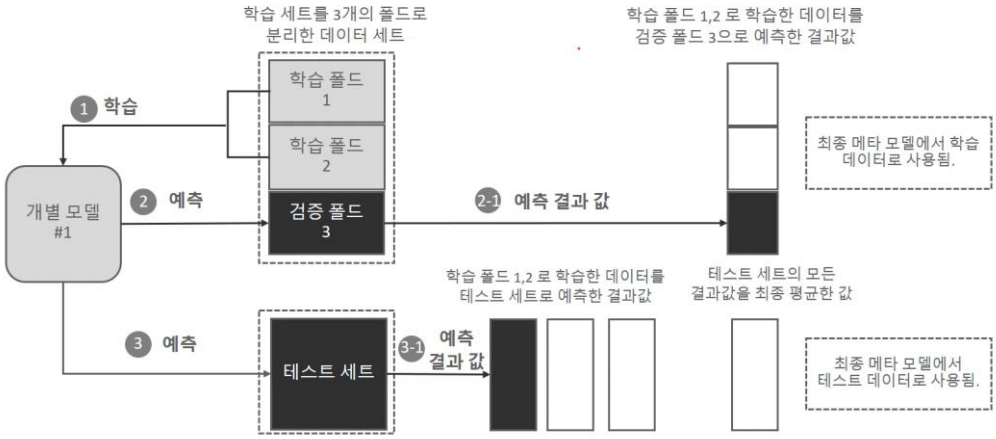


2. 첫번째 레이어의 예측기를 사용해 두 번째 (홀드 아웃) 세트에 대한 예측을 만든다.
예측기들이 훈련하는 동안 이 샘플들을 전혀 보지 못했기 때문에 이때 만들어진 예측은 완전히 새로운 것.
홀드 아웃 세트의 각 샘플에 대해 세 개의 예측값이 있다.
타깃값은 그대로 쓰고 앞에서 예측한 값을 입력 특성으로 사용하는 새로운 훈련 세트를 만들 수 있다. (3차원)
블렌더가 새 훈련 세트로 훈련된다. 즉, 첫 번째 레이어의 예측을 가지고 타깃값을 예측하도록 학습된다.

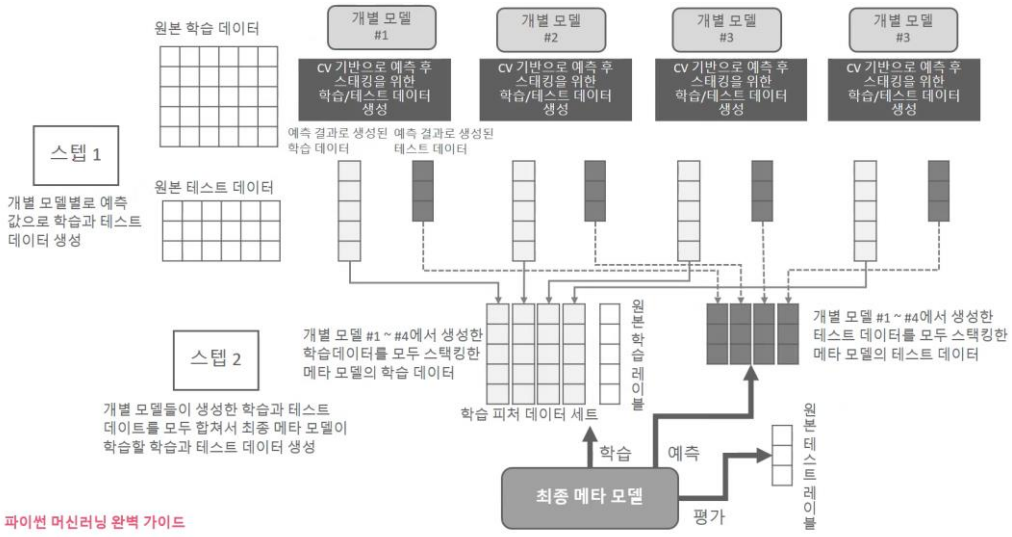
CV 세트 기반의 스태킹

과적합을 개선하기 위해 개별 모델들이 각각 교차 검증으로 메타 모델을 위한 학습용 스태킹 데이터 생성과 예측을 위한 테스트용 스태킹 데이터를 생성한 뒤 이를 기반으로 메타 모델이 학습과 예측을 수행한다.

[Step 1] : 각 모델별로 원본 학습/테스트 데이터를 예측한 결과 값을 기반으로 메타 모델을 위한 학습용 / 테스트용 데이터를 생성한다.
[Step 2] : [Step 1]에서 개별 모델들이 생성한 학습용(테스트용) 데이터를 모두 스태킹 형태로 합쳐서 메타 모델이 학습할(예측할) 최종 학습용(테스트) 데이터를 생성한다. 메타 모델은 최종적으로 생성된 학습 데이터 세트와 원본 학습 데이터의 레이블 데이터를 기반으로 학습한 뒤, 최종적으로 생성된 테스트 데이터 세트를 예측하고, 원본 테스트 데이터의 레이블 데이터를 기반으로 평가한다.



1. 학습용 데이터를 N개의 폴드(Fold)로 나눈다. (예시: N=3)
2. 2개의 폴드는 학습을 위한 데이터 폴드, 나머지 1개의 폴드는 검증을 위한 데이터 폴드로 나눈다.
3. 2개의 학습 폴드를 기반으로 개별 모델을 학습시킨다.
4. 학습된 개별 모델은 검증폴드 1개 데이터로 예측하고 그 결과를 저장한다.
5. 이 과정을 3번 반복하여 학습, 검증 데이터 세트를 변경해가면서 학습 후 예측 결과를 별도로 저장
6. 5번에서 저장된 예측 데이터는 메타 모델을 학습시키는 학습 데이터로 사용
7. 2개의 학습폴드 데이터로 학습된 개별 모델은 원본 테스트 데이터를 예측하여 예측값을 생성
8. 이러한 로직을 3번 반복하면서 이 예측값의 평균으로 최종 결괏값을 생성하고 이를 메타 모델을 위한 테스트 데이터로 사용



9. 메타 모델이 사용할 최종 학습 데이터와 원본 데이터의 레이블 데이터를 합쳐서 메타 모델을 학습한 후에 최종 테스트 데이터로 예측을 수행한 뒤, 최종 예측 결과를 원본 테스트 데이터의 레이블 데이터와 비교해 평가ㄴ