

# 7장 앙상블 학습과 랜덤 포레스트 1부

# 주요 내용

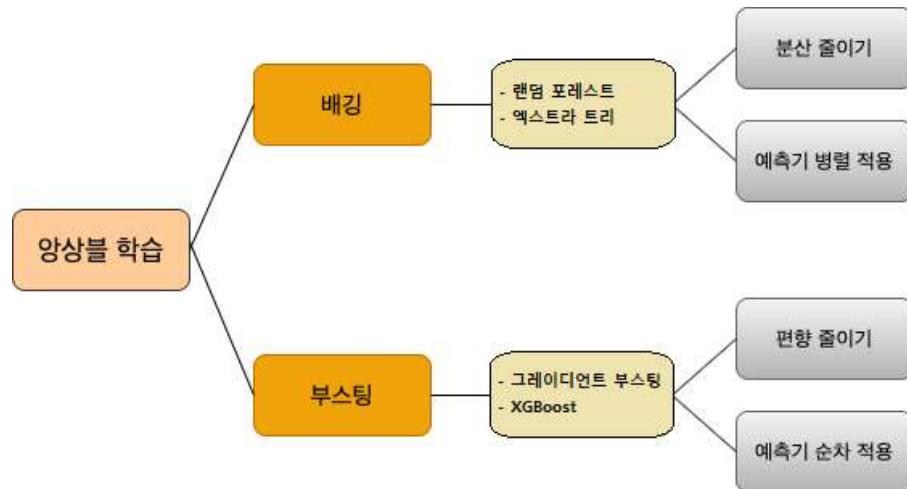
- 앙상블 학습
- 배깅
  - 배깅과 페이스팅
  - 램덤포레스트
- 부스팅
  - 그레이디언트 부스팅
  - XGBoost

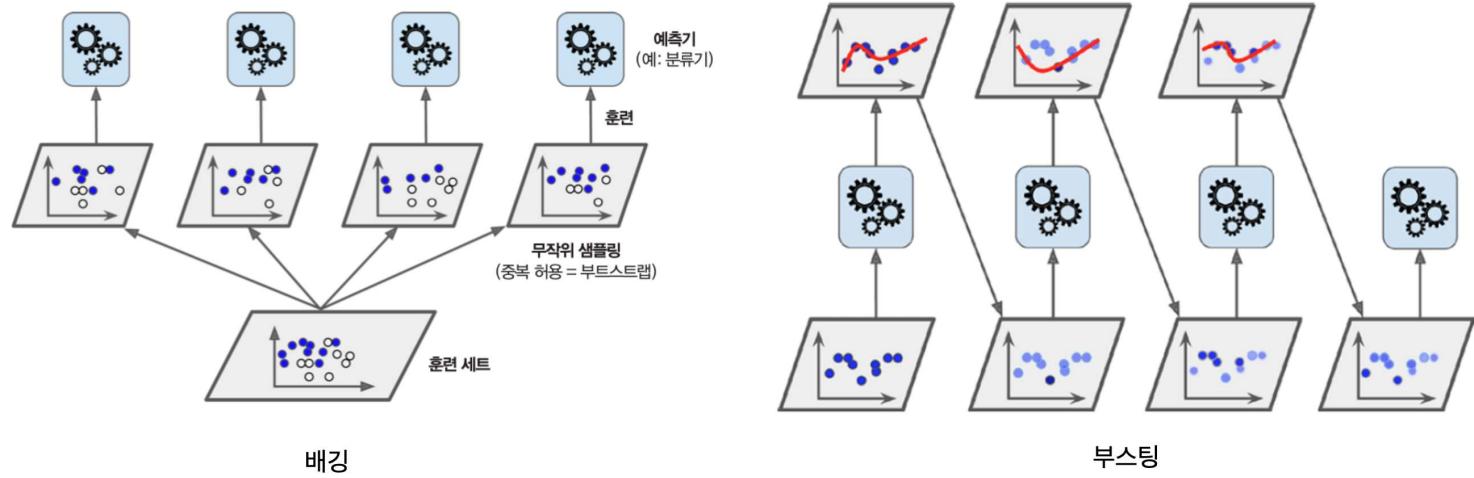
양상을 학습

## 앙상블 학습이란?

- 여러 개의 모델을 훈련시킨 결과를 이용하여 기법
- 배깅 bagging 기법과 부스팅 boosting 기법이
- 배깅 기법: 여러 개의 예측기를 (가능한한) 독립적으로 학습시킨 후 모든 예측기들의 예측값들의 평균값을 최종 모델의 예측값으로 사용한다. 분산이 보다 줄어든 모델을 구현한다.
- 부스팅 기법: 여러 개의 예측기를 순차적으로 훈련시킨 결과를 예측값으로 사용한다. 보다 적은 편향을 갖는 모델을 구현한다.

# 배깅과 부스팅





## 캐글(Kaggle)과 양상블 학습

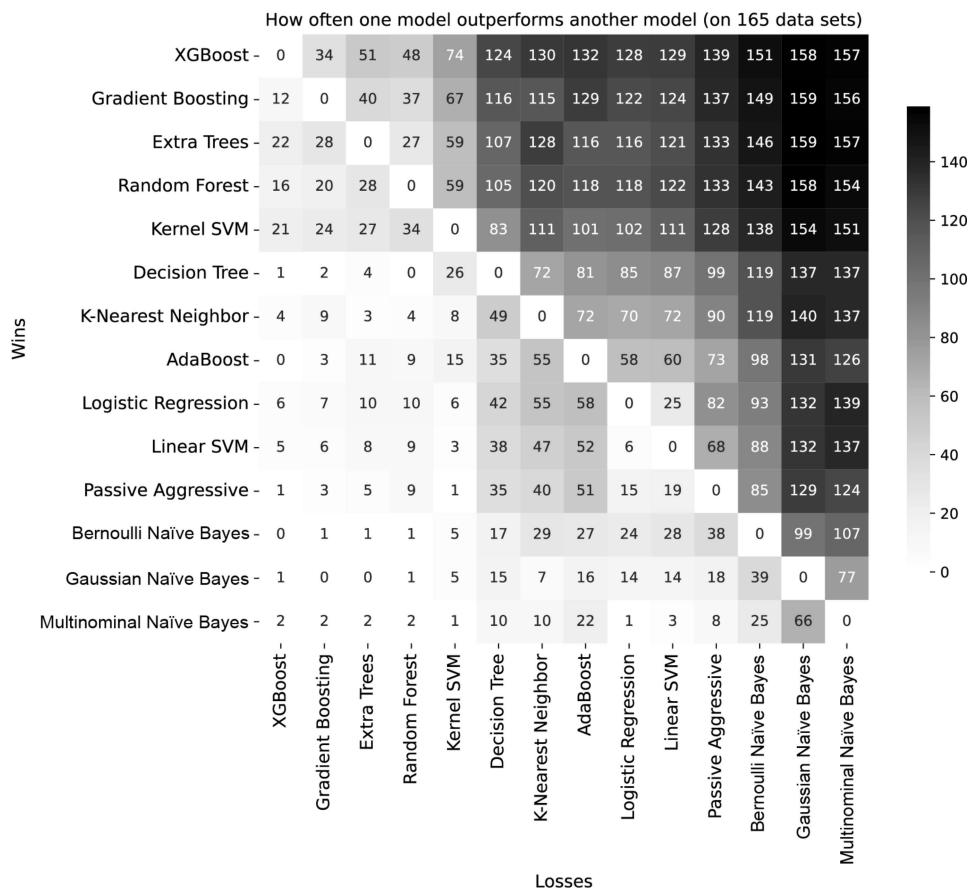
캐글 Kaggle 경진대회에서 가장 좋은 성능을 내는 3 개의 모델은 다음과 같이 모두 양상블 학습 모델이다.

- XGBoost
- 랜덤 포레스트
- 그레이디언트 부스팅

양상블 학습 모델은 특히 엑셀의 표 table 형식으로 저장될 수 있는 정형 데이터 structured data의 분석에 유용한다.

반면에 이미지, 오디오, 동영상, 자연어 등 비정형 데이터 unstructured data에 대한 분석은 지금은 딥러닝 기법이 훨씬 좋은 성능을 보인다. 그럼에도 불구하고 양상블 학습 기법을 딥러닝 모델에 적용하여 모델의 성능 최대한 끌어 올리기도 한다.

# 앙상블 학습 모델 비교



## 편향과 분산

- 앙상블 학습의 핵심: 편향과 분산 줄이기
- 편향: 예측값과 정답이 떨어져 있는 정도. 정답에 대한 잘못된 가정으로 발생하며, 편향이 크면 과소적합 발생.
- 분산: 샘플의 작은 변동에 반응하는 정도. 정답에 대한 너무 복잡한 모델을 설정하는 경우 발생할 수 있으며, 분산이 크면 과대적합 발생.

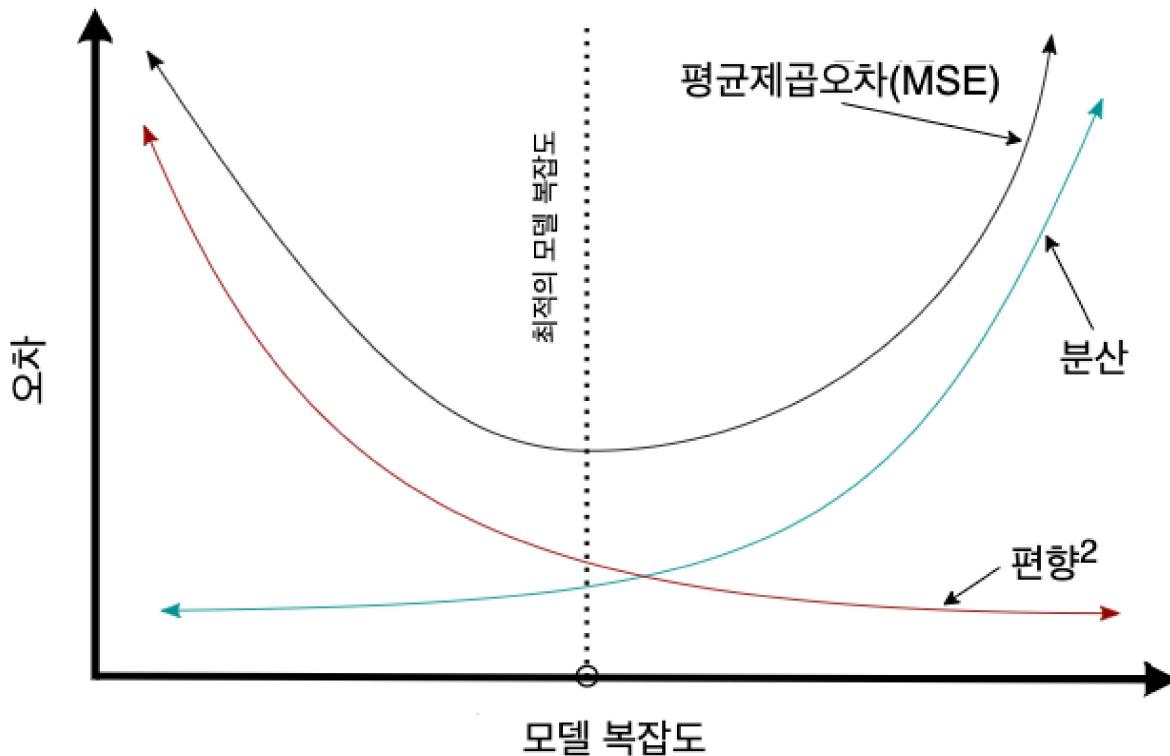
## 편향과 분산의 트레이드오프

- 편향과 분산의 트레이드오프: 편향과 분산을 동시에 좋아지게 할 수는 없음.
- 예제: 훈련셋 크기
  - 훈련셋 작게: 편향은 커지고, 분산은 작아짐.
  - 훈련셋 크게: 편향은 작아지고, 분산은 커짐.
- 예제: 특성 개수
  - 특성 개수 작게: 편향은 커지고, 분산은 작아짐.
  - 특성 개수 크게: 편향은 작아지고, 분산은 커짐.

## 모델 복잡도, 편향, 분산의 관계

- 회귀모델의 평균제곱오차는 편향의 제곱과 분산의 합으로 근사됨.

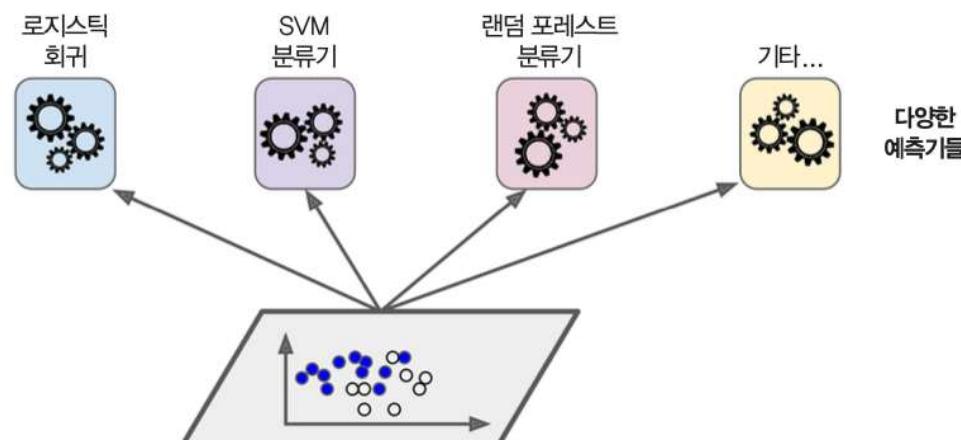
$$\text{평균제곱오차} \approx \text{편향}^2 + \text{분산}$$



## 7.1 투표식 분류기

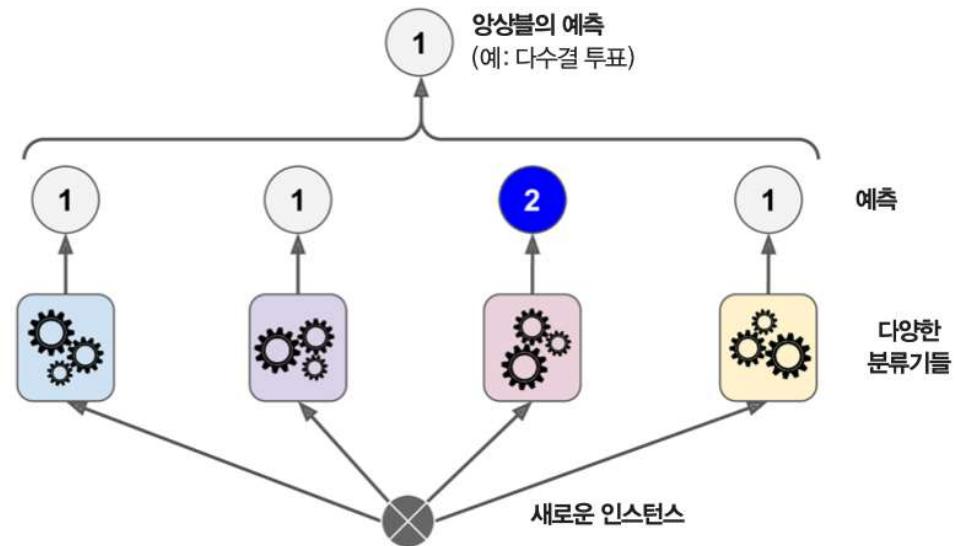
### 정의

- 동일한 훈련셋에 대해 여러 종류의 분류기 이용한 앙상블 학습 적용 후 직접 또는 간접 투표를 통해 예측값 결정.



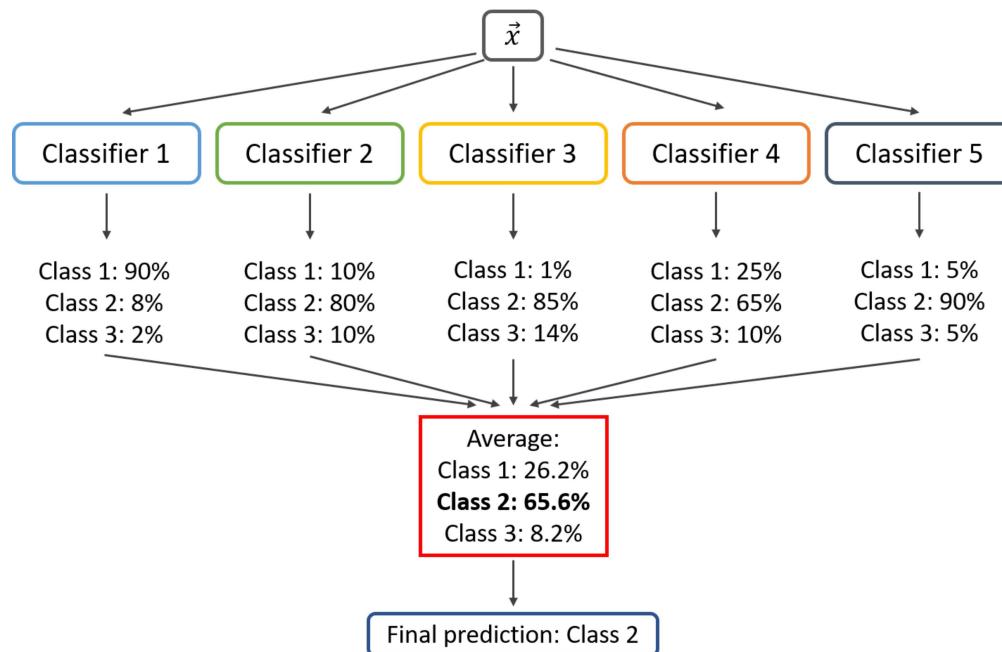
## 직접투표

- 양상블에 포함된 예측기들의 예측값들의 다수로 결정



# 간접투표

- 양상블에 포함된 예측기들의 예측한 확률값들의 평균값으로 예측값 결정
- 전제: 모든 예측기가 `predict_proba()` 메서드와 같은 확률 예측 기능을 지원해야 함.
- 높은 확률에 보다 비중을 두기 때문에 직접투표 방식보다 성능 좀 더 좋음.



## 직접 투표 대 간접 투표

분류기 다섯개의 예측확률이 아래와 같은 경우 직접 투표 방식과 간접 투표 방식의 결과가 다르다.

분류기	클래스1 예측 확률	클래스2 예측 확률	클래스3 예측 확률
분류기1	90%	8%	2%
분류기2	40%	7%	53%
분류기3	45%	9%	46%
분류기4	30%	20%	50%
분류기5	44%	16%	40%
합	249%	60%	191%

- 직접 투표: 클래스 3으로 예측
- 간접 투표: 클래스 1로 예측

## 투표식 분류기의 확률적 근거

이항분포의 누적분포함수를 이용하여 앙상블 학습의 성능이 향상되는 이유를 설명할 수 있음.

- $p$ : 예측기 하나의 성능
- $n$ : 예측기 개수
- 반환값: 다수결을 따를 때 성공할 확률, 즉 다수결 의견이 보다 정확할 확률. 이항 분포의 누적분포함수 활용.

```
In [1]: from scipy.stats import binom

def ensemble_win_proba(n, p):
    """
    p: 예측기 하나의 성능
    n: 앙상블 크기, 즉 예측기 개수
    반환값: 다수결을 따를 때 성공할 확률. 이항 분포의 누적분포함수 반환값.
    """
    return 1 - binom.cdf(int(n*0.4999), n, p)
```

적중률 51% 모델 1,000개의 다수결을 따르면 74.7% 정도의 적중률 나옴.

```
In [2]: ensemble_win_proba(1000, 0.51)
```

```
Out[2]: 0.7467502275563249
```

적중률 51% 모델 10,000개의 다수결을 따르면 97.8% 정도의 적중률 나옴.

```
In [3]: ensemble_win_proba(10000, 0.51)
```

```
Out[3]: 0.9777976478701103
```

적중률 80% 모델 10개의 다수결을 따르면 100%에 가까운 성능이 가능함.

```
In [4]: ensemble_win_proba(10, 0.8)
```

```
Out[4]: 0.9936306176
```

- **주의사항:** 양상별 학습에 포함된 각각의 모델이 서로 독립인 것을 전제로한 결과임.
- 동일한 데이터를 사용할 경우 독립성이 보장되지 않으며, 경우에 따라 성능이 하락 할 수 있음.
- 독립성을 높이기 위해 매우 다른 알고리즘을 사용하는 여러 모델을 사용해야 함.

## 투표식 분류기 예제

- voting='hard' 또는 voting='soft' : 직접 또는 간접 투표 방식 지정 하이퍼파라미터. 기본값은 'hard'.
- 주의: SVC 모델 지정할 때 probability=True 사용해야 predict\_proba() 메서드 지원됨.

```
voting_clf = VotingClassifier(  
    estimators=[  
        ('lr', LogisticRegression(random_state=42)),  
        ('rf', RandomForestClassifier(random_state=42)),  
        ('svc', SVC(random_state=42))  
    ]  
)
```

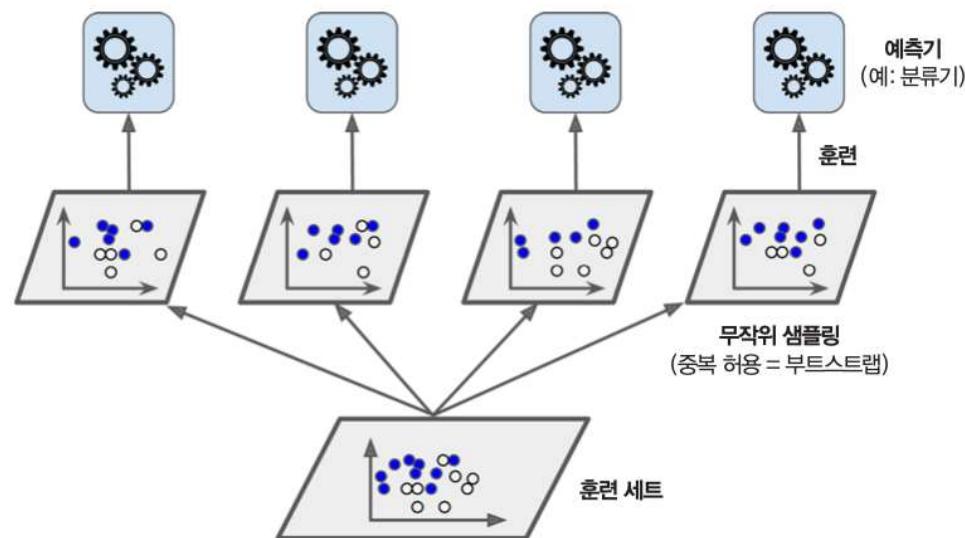
## 7.2 배경과 페이스팅

## 정의

- 여러 개의 동일 모델을 하나의 훈련셋의 다양한 부분집합을 대상으로 학습시키는 방식
- 부분집합을 임의로 선택할 때 중복 허용 여부에 따라 양상을 학습 방식이 달라짐
  - **배깅**: 중복 허용 샘플링
  - **페이스팅**: 중복 미허용 샘플링

# 배깅

- 배깅(bagging): bootstrap aggregation의 줄임말
- **부트스트래핑**: 통계에서 중복허용 리샘플링을 가리킴



## 배깅/페이스팅의 예측값

- 분류 모델: 직접 투표 방식 사용. 즉, 수집된 예측값들 중에서 최빈값(mode) 선택
- 회귀 모델: 수집된 예측값들의 평균값 선택

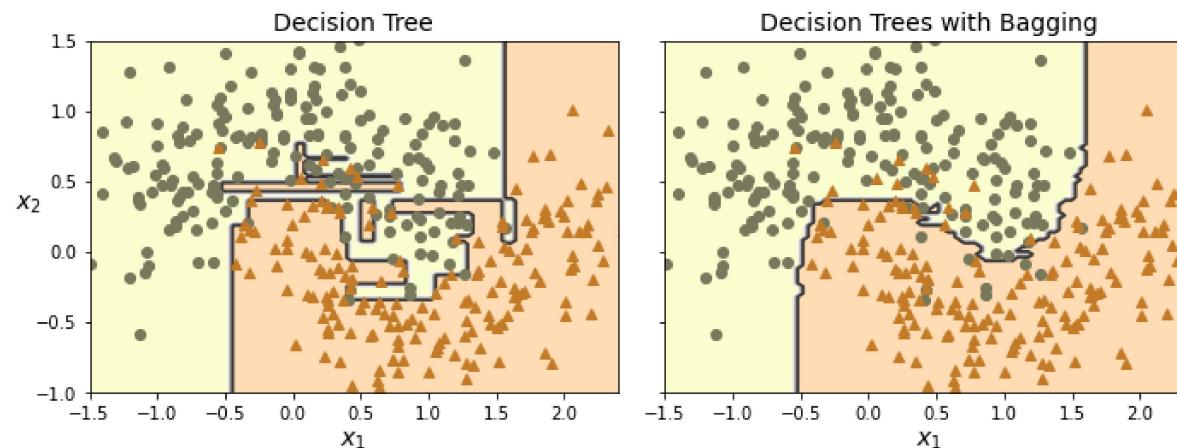
## 배깅/페이스팅 방식으로 훈련된 모델의 편향과 분산

- 개별 예측기의 경우에 비해 편향은 조금 커지거나 거의 비슷하지만 분산은 줄어듦.
  - 배깅이 표본 샘플링의 다양성을 보다 많이 추가하기 때문임.
  - 배깅이 과대적합의 위험성일 보다 줄어주며, 따라서 배깅 방식이 기본으로 사용됨.
- 개별 예측기: 배깅/페이스팅 방식으로 학습하면 전체 훈련셋을 대상으로 학습한 경우에 비해 편향이 커짐. 따라서 과소적합 위험성 커짐.
- 참고: [Single estimator versus bagging: bias-variance decomposition](#)

## 예제: 사이킷런의 배깅/페이스팅

- 왼쪽 그림: 규제 없는 결정 트리 모델. 훈련셋에 과대적합됨.
- 오른쪽 그림:
  - 규제 `max_samples=100` 를 사용하는 결정트리 500개
  - 배깅방식

```
BaggingClassifier(DecisionTreeClassifier(),
                   n_estimators=500,
                   max_samples=100, random_state=42)
```



## oob 평가

- oob(out-of-bag) 샘플: 배깅 모델에 포함된 예측기로부터 선택되지 않은 훈련 샘플.  
평균적으로 훈련셋의 약 37% 정도.
- oob 평가: 각각의 샘플에 대해 해당 샘플을 훈련에 사용하지 않은 모델들의 예측값  
을 이용하여 앙상블 학습 모델을 검증하는 기법

## 예제

- 6 개의 훈련 샘플로 구성된 훈련셋 대해 5개의 결정트리 모델을 배깅 기법으로 적용
- 표에 사용된 정수는 중복으로 뽑힌 횟수
- 각 샘플은 위치 인덱스로 구분

훈련 샘플(총 6개)		OOB 평가 샘플
결정트리1	1, 1, 0, 2, 1, 1	2번
결정트리2	3, 0, 1, 0, 2, 0	1번, 3번, 5번
결정트리3	0, 1, 3, 1, 0, 1	0번, 4번
결정트리4	0, 0, 2, 0, 2, 2	0번, 1번, 3번
결정트리5	2, 0, 0, 1, 3, 0	1번, 2번, 5번

그러면 각 샘플을 이용한 양상을 학습에 사용된 모델은 다음과 같다.

- 0번 샘플: 결정트리3, 결정트리4
- 1번 샘플: 결정트리2, 결정트리4, 결정트리5
- 2번 샘플: 결정트리1, 결정트리5
- 3번 샘플: 결정트리2, 결정트리4
- 4번 샘플: 결정트리3
- 5번 샘플: 결정트리2, 결정트리5

## 예제: BaggingClassifier 를 이용한 oob 평가

- BaggingClassifier 의 oob\_score=True 옵션
  - 훈련 종료 후 oob 평가 자동 실행
  - 평가점수는 oob\_score\_ 속성에 저장됨.
  - 테스트세트에 대한 정확도와 비슷한 결과가 나옴.

```
BaggingClassifier(DecisionTreeClassifier(),
                   n_estimators=500,
                   oob_score=True, random_state=42)
```

## 7.3 랜덤 패치와 랜덤 서브스페이스

- `BaggingClassifier` 는 특성에 대한 샘플링 기능도 지원: `max_features` 와 `bootstrap_features`
- 이미지 등 매우 높은 차원의 데이터셋을 다룰 때 유용
- 더 다양한 예측기를 만들며, 편향이 커지지만 분산은 낮아짐

## max\_features

- 학습에 사용할 특성 수 지정
- 특성 선택은 무작위
  - 정수인 경우: 지정된 수만큼 특성 선택
  - 부동소수점( $\in [0, 1]$ )인 경우: 지정된 비율만큼 특성 선택
- max\_samples와 유사 기능 수행

## bootstrap\_features

- 학습에 사용할 특성을 선택할 때 중복 허용 여부 지정
- 기본값은 False. 즉, 중복 허용하지 않음.
- bootstrap과 유사 가능 수행

## 랜덤 패치 기법

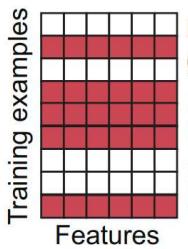
- 훈련 샘플과 훈련 특성 모두를 대상으로 중복을 허용하며 임의의 샘플 수와 임의의 특성 수만큼을 샘플링해서 학습하는 기법

```
BaggingClassifier(DecisionTreeClassifier(), n_estimators=500,  
                  max_samples=0.75, bootstrap=True,  
                  max_features=0.5, bootstrap_features=True,  
                  random_state=42)
```

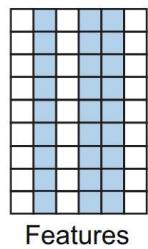
## 랜덤 서브스페이스 기법

- 전체 훈련 세트를 학습 대상으로 삼지만 훈련 특성은 임의의 특성 수만큼 샘플링해서 학습하는 기법
  - 샘플에 대해: `bootstrap=False`이고 `max_samples=1.0`
  - 특성에 대해: `bootstrap_features=True` 또는 `max_features`는 1.0 보다 작게.

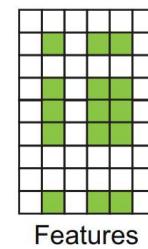
```
BaggingClassifier(DecisionTreeClassifier(), n_estimators=500,  
                  max_samples=1.0, bootstrap=False,  
                  max_features=0.5, bootstrap_features=True,  
                  random_state=42)
```



**Bagging  
(sample instances)**  
max\_samples = 0.75  
bootstrap = True  
max\_features = 1.0  
bootstrap\_features = False



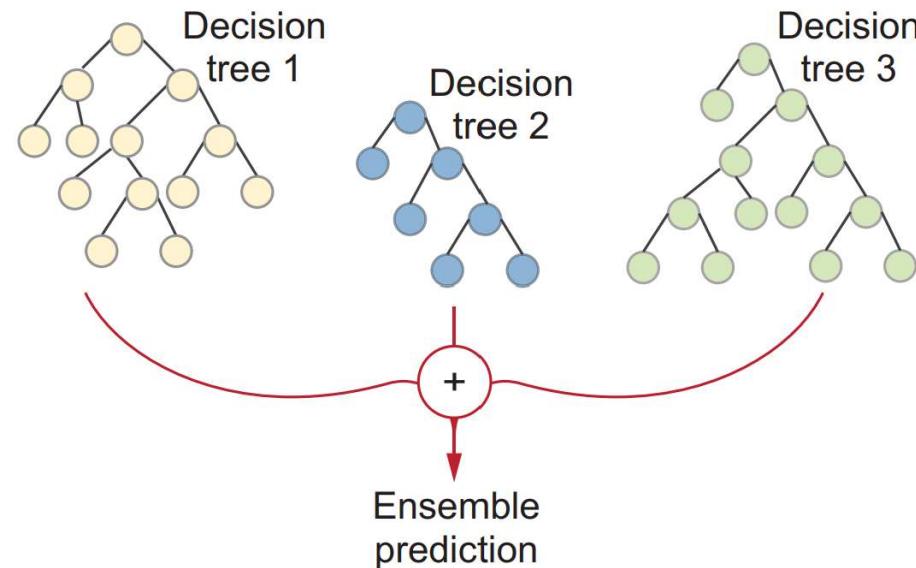
**Random subspaces  
(sample features)**  
max\_samples = 1.0  
bootstrap = False  
max\_features = 0.5  
bootstrap\_features = True



**Random patches  
(sample both)**  
max\_samples = 0.75  
bootstrap = True  
max\_features = 0.5  
bootstrap\_features = True

## 7.4 랜덤 포레스트

- 배깅/페이스팅 방법을 적용한 결정트리의 앙상블을 최적화한 모델
  - 분류 용도: RandomForestClassifier
  - 회귀 용도: RandomForestRegressor



- 아래 두 모델은 기본적으로 동일한 모델임.

```
RandomForestClassifier(n_estimators=500, max_leaf_nodes=16,  
                      n_jobs=-1, random_state=42)
```

```
BaggingClassifier(DecisionTreeClassifier(max_features="sqrt",  
                                         max_leaf_nodes=16),  
                  n_estimators=500,  
                  n_jobs=-1, random_state=42)
```

## 랜덤 포레스트 하이퍼파라미터

- BaggingClassifier 와 DecisionTreeClassifier 의 옵션을 거의 모두 가짐. 예외는 다음과 같음.
  - DecisionClassifier 의 옵션 중: splitter='random', presort=False, max\_samples=1.0
  - BaggingClassifier 의 옵션 중:  
base\_estimator=DecisionClassifier(...)
- splitter='random' 옵션: 특성 일부를 무작위적으로 선택한 후 최적의 임곗값 선택
- max\_features='auto' 가 RandomForestClassifier 의 기본값임. 따라서 특성 선택에 무작위성 사용됨.
  - 선택되는 특성 수: 약  $\sqrt{\text{전체 특성 수}}$
- 결정트리에 비해 편향은 크게, 분산은 낮게.

## 엑스트라 트리

- 익스트림 랜덤 트리(**extremely randomized tree**) 양상을 이라고도 불림.

```
extra_clf = ExtraTreesClassifier(n_estimators=500, max_leaf_nodes=16,  
                                 n_jobs=-1, random_state=42)
```

- 무작위로 선택된 일부 특성에 대해 특성 임곗값도 무작위로 몇 개 선택한 후 그중에서 최적 선택
- 일반적인 램덤포레스트보다 속도가 훨씬 빠름
- 이 방식을 사용하면 편향은 늘고, 분산은 줄어듦

## 특성 중요도

- 특성 중요도: 해당 특성을 사용한 마디가 평균적으로 불순도를 얼마나 감소시키는지를 측정
  - 즉, 불순도를 많이 줄이면 그만큼 중요도가 커짐
- 사이킷런의 `RandomForestClassifier`
  - 특성별 상대적 중요도를 측정해서 중요도의 전체 합이 1이 되도록 함.
  - `feature_importances_` 속성에 저장됨.

예제: 붓꽃 데이터셋

특성	중요도(%)
꽃잎 길이	44.1
꽃잎 너비	42.3
꽃받침 길이	11.3
꽃받침 너비	2.3

예제: MNIST

아래 이미지는 각 픽셀의 중요도를 보여준다.

