7장 앙상블 학습과 랜덤 포레스트 1부

감사의 글

자료를 공개한 저자 오렐리앙 제롱과 강의자료를 지원한 한빛아카데미에게 진심어린 감사를 전합니다.

주요 내용

- 1부
    - 앙상블 학습이란?
    - 배깅
        - 페이스팅
        - 램덤 패치 / 랜덤 서브스페이스
        - 램덤포레스트
- 2부
    - 부스팅
    - 스태킹

앙상블 학습이란?

- 여러 개 모델의 예측값을 조합하여 일반화 성능이 좋은 보다 강력한 모델을 구현하며, 대표적으로 평균화 기법과 부스팅 기법이 사용된다.
- 평균화 기법: 독립적으로 학습된 예측기 여러 개의 예측값들의 평균값을 예측값으로 사용하여 분산이 줄어든 모델을 구현한다.
    - 예제: 배깅(Bagging) 기법, 랜덤 포레스트(Random Forest)등
- 부스팅 기법: 예측기 여러 개를 순차적으로 쌓아 예측값의 편향를 줄인 모델을 구현한다.
    - 예제: 에이다부스트(AdaBoost), 그레이디언트 부스팅(Gradient Tree Boosting) 등

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.

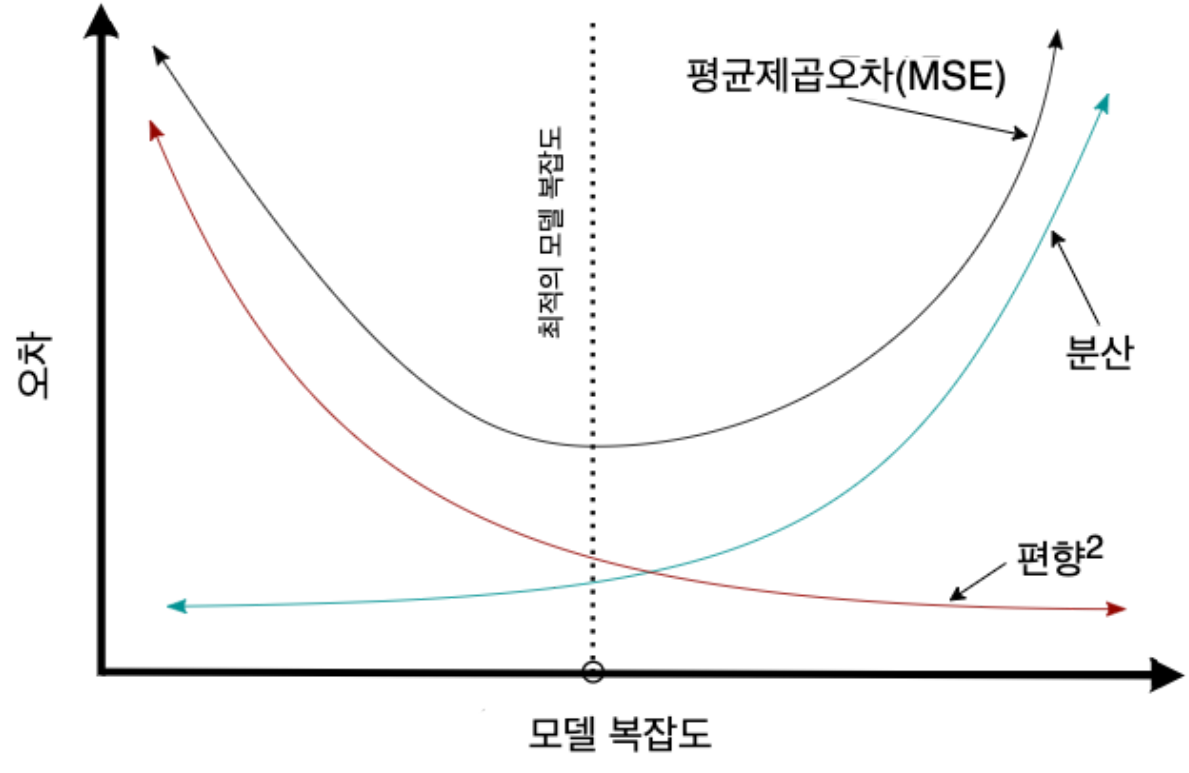
편향과 분산의 트레이드오프

- 편향과 분산의 트레이드오프: 편향과 분산을 동시에 좋아지게 할 수는 없음.
- 예제: 훈련 세트 크기
    - 훈련 세트 작게: 편향은 커지고, 분산은 작아짐.
    - 훈련 세트 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.

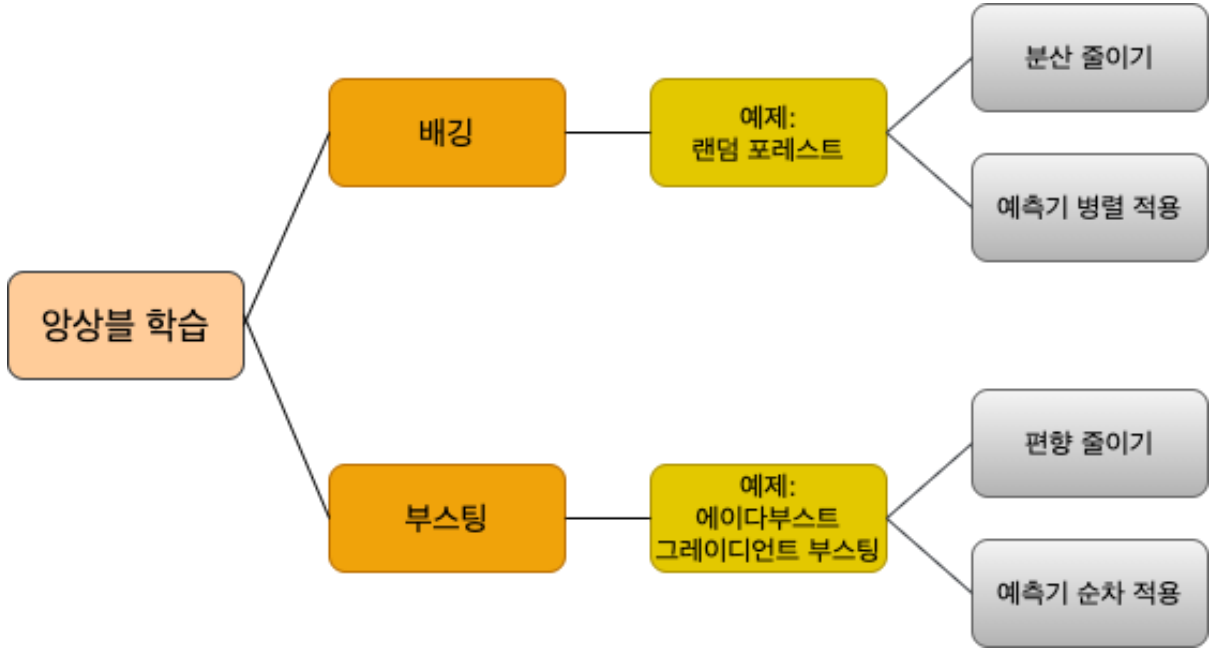
$$\text{평균제곱오차} \approx \text{편향}^2 + \text{분산}$$

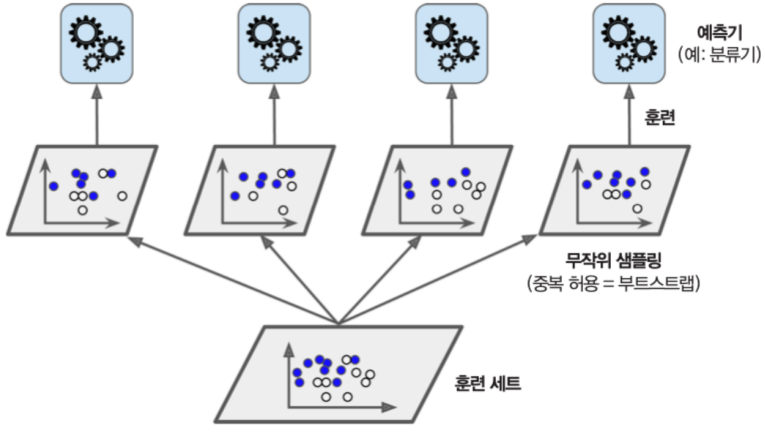


<위키백과: 편향-분산 트레이드오프>

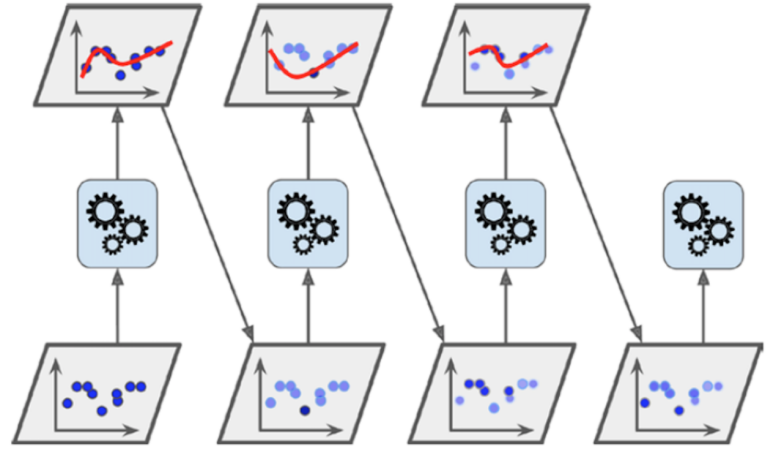
배깅 vs. 부스팅

- 편향 또는 분산을 줄이기 위한 앙상블 학습의 주요 두 기법: 배깅과 부스팅





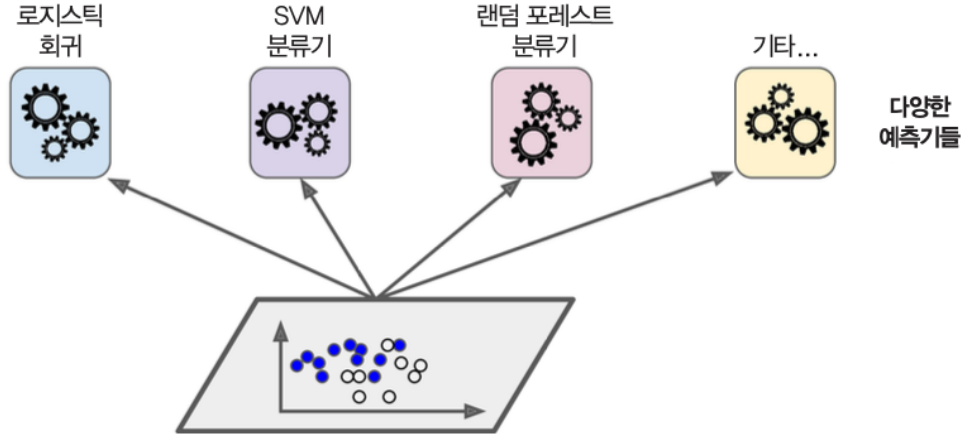
배깅



부스팅

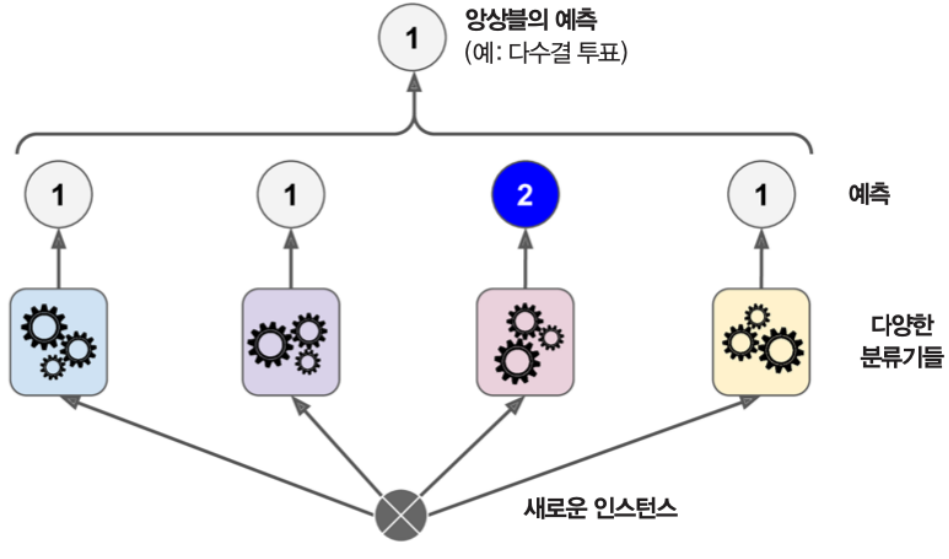
7.1 투표식 분류기

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



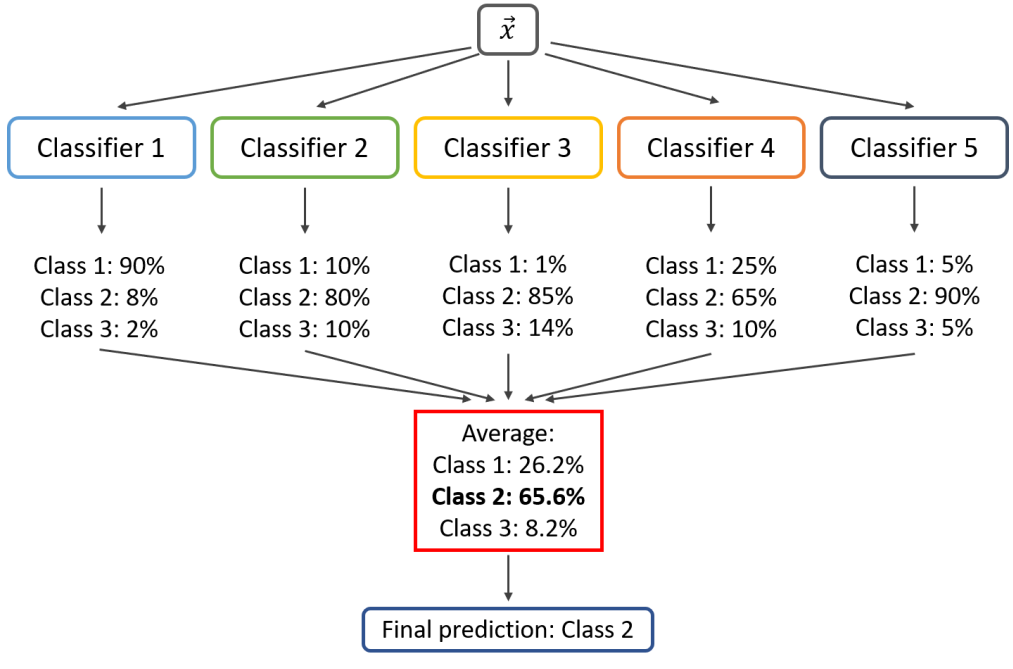
직접투표

- 앙상블에 포함된 예측기들의 예측값들의 다수로 결정



간접투표

- 앙상블에 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.



<그림출처: kaggle>

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 누적분포 함수 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 활용.
    """
    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]:
```
0.7467502275561786
```

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

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

Out[3]:
```
0.9777976478701533
```

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

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

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

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

투표식 분류기 예제

- 사이킷런의 투표식 분류기
- `VotingClassifier` : 투표식 분류기 모델 제공
    - `voting='hard'` : 직접 투표 방식 지정 하이퍼 파라미터
    - `voting='soft'` : 간접 투표 방식 지정 하이퍼 파라미터
    - 주의: `SVC` 모델 지정할 때 `probability=True` 사용해야 `predict_proba()` 메서드 지원됨.

```
log_clf = LogisticRegression(solver="lbfgs", random_state=42)
rnd_clf = RandomForestClassifier(n_estimators=100, random_state=42)
svm_clf = SVC(gamma="scale", random_state=42)

# 투표식 분류기: 직접 투표
voting_clf = VotingClassifier(
    estimators=[('lr', log_clf), ('rf', rnd_clf), ('svc', svm_clf)],
    voting='hard')
```

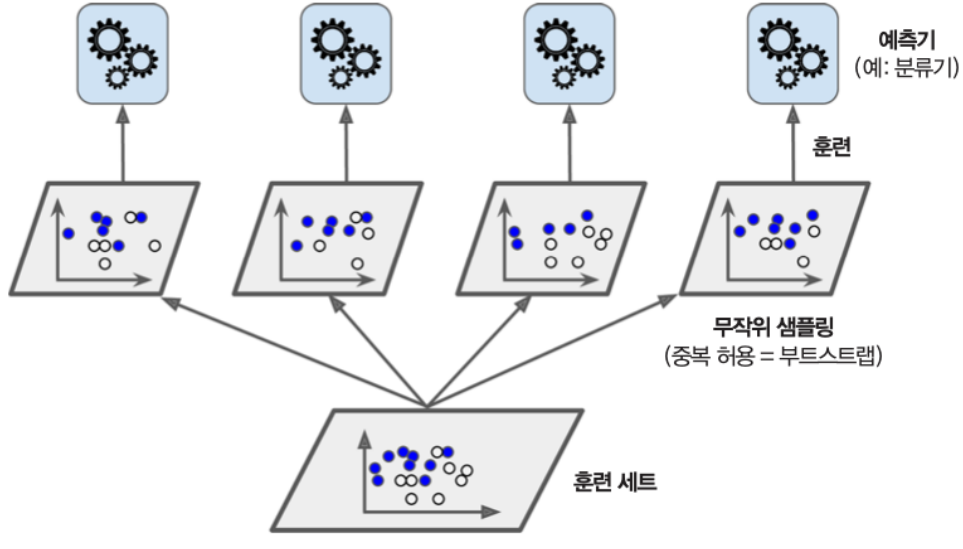
7.2 배깅/페이스팅

정의

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

배깅

- 배깅(bagging): bootstrap aggregation의 줄임말
- 통계 분야에서 **부트스트래핑**, 즉, 중복허용 리샘플링으로 불림



배깅/페이스팅 예측 방식

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

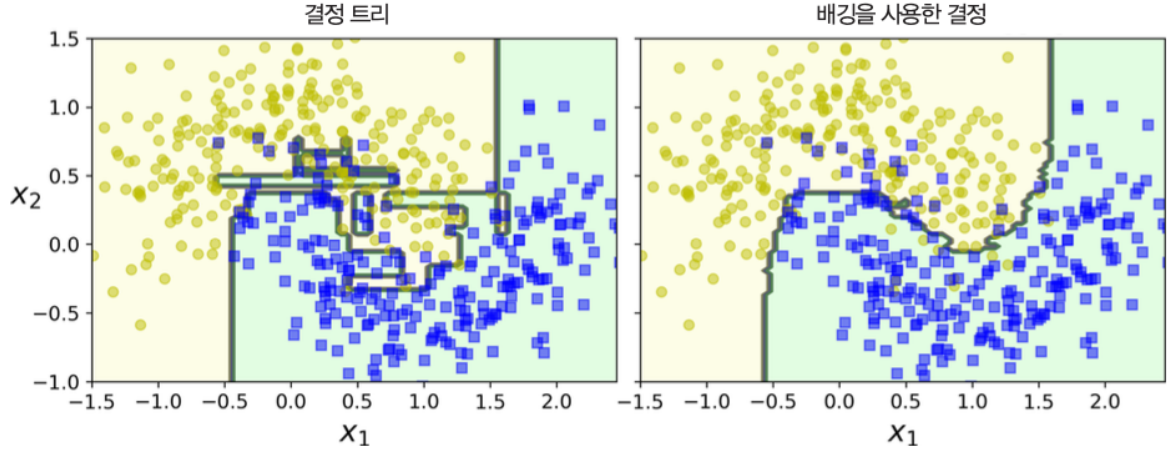
앙상블 학습의 편향과 분산

- 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 훈련 세트를 대상으로 학습한 경우에 비해 편향이 커짐. 따라서 과소적합 위험성 커짐.
- **참고:** Single estimator versus bagging: bias-variance decomposition

예제: 사이킷런의 배깅/페이스팅

- 훈련세트(`X_train`) 크기: 500
- `n_estimators=500` : 결정트리 500개 사용
- `max_samples=100` : 각 예측기가 100개 샘플 사용. 기본값은 1.0, 즉 전체 훈련 샘플 선택
- `bootstrap=True` : 배깅 방식 사용(기본값). `False` 면 페이스팅 방식 사용.
- `n_jobs=-1` : 모든 사용가능한 cpu 사용하여 훈련을 **병렬처리**함. 양의 정수일 경우 정해진 수의 cpu 사용.

```
bag_clf = BaggingClassifier(DecisionTreeClassifier(random_state=42),
                            n_estimators=500, max_samples=100,
                            bootstrap=True, n_jobs=-1, random_state=42)
```



oob 평가

- oob(out-of-bag) 샘플: 배깅 모델에 포함된 예측기로부터 선택되지 않은 훈련 샘플. 평균적으로 훈련 세트의 약 37% 정도.
- 각 예측기가 oob 샘플을 성능 검증에 활용.
- 앙상블 모델 자체의 성능 검증은 각 예측기의 oob 검증결과의 평균값 활용

앙상블 모델의 검증과 테스트

- `BaggingClassifier` 의 `oob_score=True` 옵션
  - 훈련 종료 후 oob 평가 자동 실행
  - 평가점수는 `oob_score_` 속성에 저정됨.
  - 테스트세트에 대한 정확도와 비슷한 결과가 나옴.

```
bag_clf = BaggingClassifier(DecisionTreeClassifier(random_state=42),
                            n_estimators=500, max_samples=100,
                            bootstrap=True, n_jobs=-1,
                            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.
- botostrap과 유사 기능 수행

랜덤 패치 기법

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

랜덤 서브스페이스 기법

- 전체 훈련 세트를 학습 대상으로 삼지만 훈련 특성은 임의의 특성 수만큼 샘플링해서 학습하는 기법

7.4 랜덤 포레스트

- 배깅/페이스팅 방법을 적용한 결정트리의 앙상블을 최적화한 모델
    - 분류 용도: `RandomForestClassifier`
    - 회귀 용도: `RandomForestRegressor`
- 아래 두 모델은 기본적으로 동일한 모델임.

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

bag_clf = BaggingClassifier(DecisionTreeClassifier(splitter="random",
                                                   max_leaf_nodes=16,
                                                   random_state=42),
                            n_estimators=500, max_samples=1.0,
                            bootstrap=True, random_state=42)
```

랜덤 포레스트 하이퍼파라미터

- `BaggingClassifier`와 `DecisionTreeClassifier`의 옵션을 거의 모두 가짐. 예외는 다음과 같음.
    - `DecisitionClassifier`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) 앙상블** 이라고도 불림.
- 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
- 참고: ExtraTreeClassifier 클래스 정의

예제

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

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.