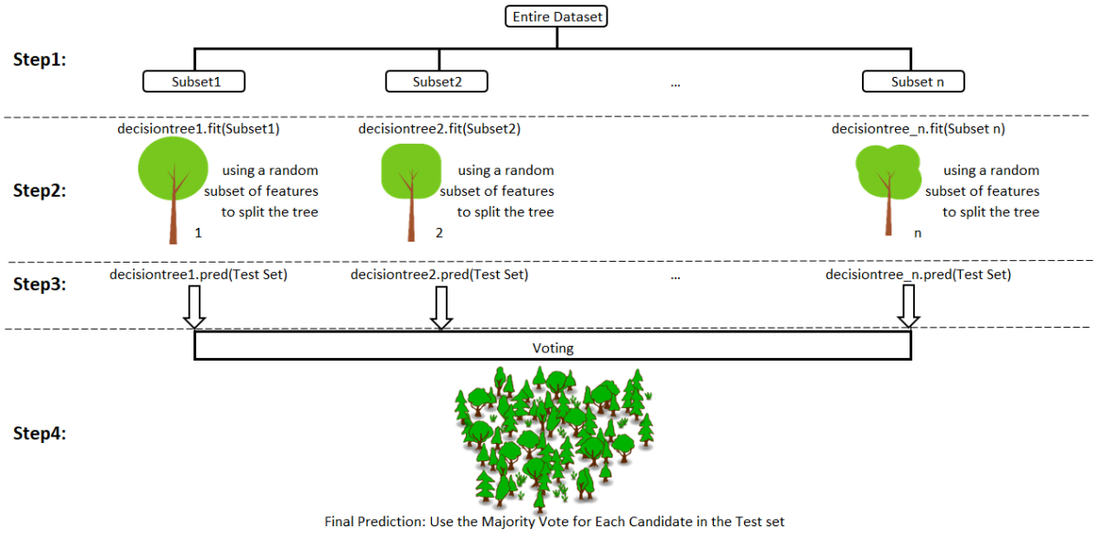
Bagging(Random Forest)

- **하나의 알고리즘**을 사용하지만 **학습 데이터셋을 랜덤하게 추출**하여 모델(분류기)을 각각 다르게 학습시키는 방법이다.

- Data Sampling : Bootstrapping
- 학습객체 : Decision Tree
- aggregate : voting



Decision Forest Model

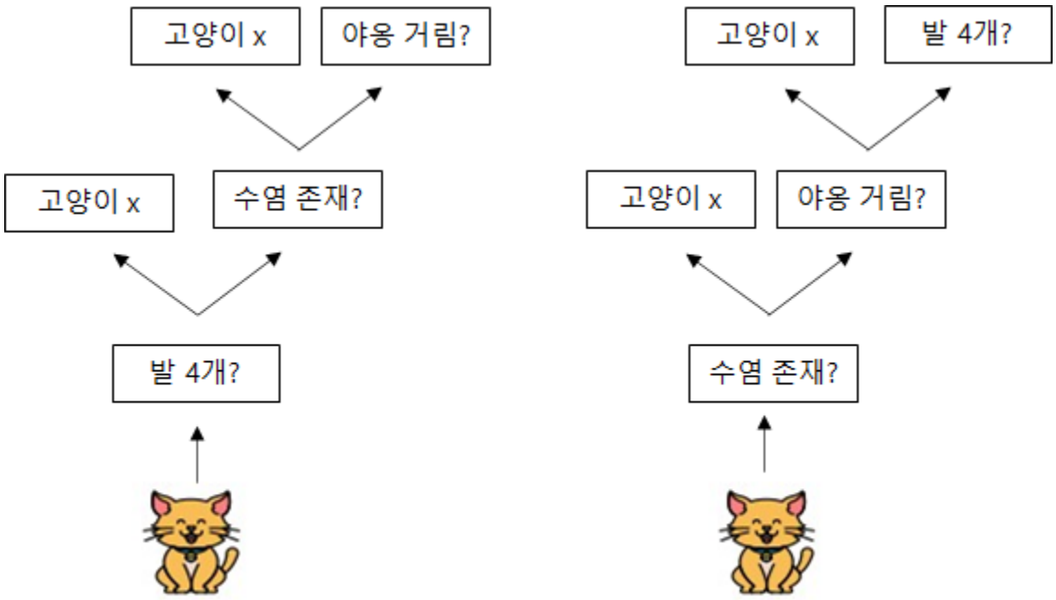
- **다양성**과 **무작위성**을 통해 Decision forest model 의 문제점 해결

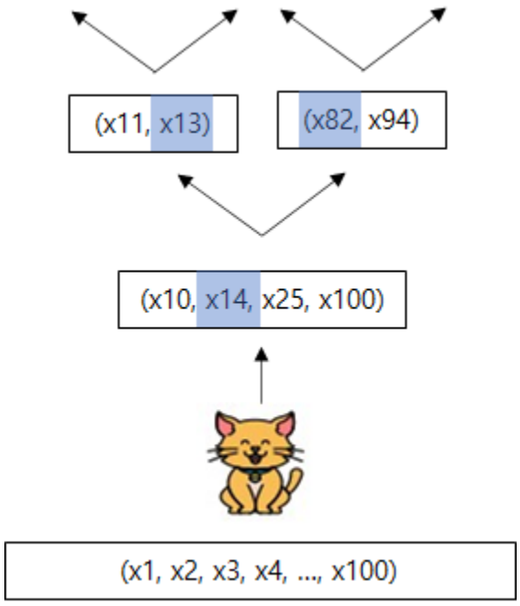
    - 여러개의 Training data 생성을 통한 다양성 확보 → **Bagging** (bootstrapping + aggregating)
    - Decision Tree model 구축시 무작위 변수선택을 이용한 무작위성 확보 → Random subspace
- Bagging 과 Random SubSpace 를 통해서 각 Tree 간의 상관관계를 줄여준다.
    - 개별적 Tree model생성

- Bagging 만 사용시 각 Tree model 간의 상관관계가 높아지게 된다.
    - model 을 학습시키는 Data를 Random 하게 선택해도 결국 그놈이 그놈
    - $var(X+Y) = var(X) + var(y) + corr(X,Y)$ 이고, Data 가 같은이상 $corr(X,Y) \neq 0$이다.

Random subspace

- 변수 : [발의 갯수, 울음소리, 수염 존재, 꼬리 존재, ...] 다양한 변수가 존재함
- Decision Tree Model 에서는 **모든 Feature 를 고려**하여 분기점을 생성
- Random Forest 에서는 모든 Feature 중에서 Model 에 사용될 **일부 Featue 를 무작위로 선택**한다.
    1. 분기점 생성시 모델 생성을 위한 입력변수를 무작위 생성
    2. 무작위로 선택된 변수중 분할 변수 선택(전체 n , classfication$\sqrt{n}$, regression $\frac{n}{3}$)
    3. Terminal Node가 될때까지 반복
- 개별모델의 성능은 Decision Tree(all Featue) 보다 낮다. but 다양한 Feature에 최적화된 모델이기 때문에 다양한 모델을 이용해 최적의 결과를 도출 해낼수 있다.
    - 원맨팀 vs 훈련이 잘된팀



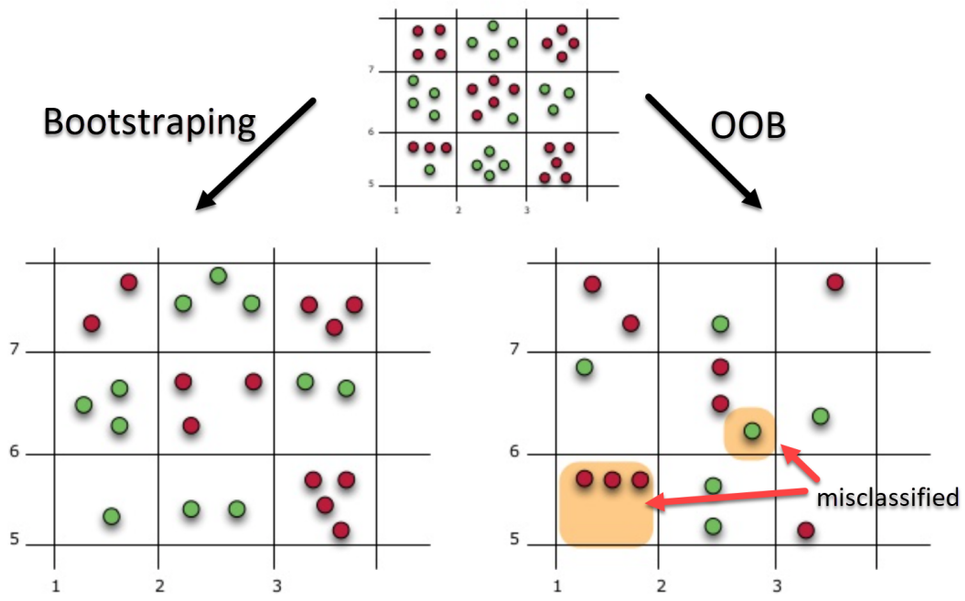


주요 Feature 선택

OOB(Out-of-Bag) 평가

- 배깅은 중복을 허용하는 리샘플링(resampling) 즉, 부트스트래핑(bootstraping) 방식이기 때문에 전체 학습 데이터셋에서 **어떠한 데이터 샘플은 여러번 샘플링 되고, 또 어떠한 샘플은 전혀 샘플링 되지 않을 수가 있다**.
- 평균적으로 학습 단계에서 전체 학습 데이터셋 중 63% 정도만 샘플링 되며(자세한 내용은 여기 참고), 샘플링 되지 않은 나머지 37% 데이터 샘플들을 **oob(out-of-bag) 샘플**이라고 한다.
- 앙상블(배깅) 모델의 학습 단계에서는 oob 샘플이 사용되지 않기 때문에, 이러한 oob 샘플을 검증셋(validation set)이나 교차검증(cross validation)에 사용할 수 있다.
- Scikit-Learn에서는 `BaggingClassifier`의 인자인 `oob_score=True`로 설정하면 학습이 끝난 후 자동으로 oob 평가를 할 수 있다.

- DataSet에서 Bootstraping 을 통해 추출될 확률 63.2%
- 나머지 36.8% 을 통해서 tree의 성능을 평가 할수 있다.



- OOBE = $\frac{11}{15} * 100 = 73.33\%$

- 앙상블의 평가는 각 예측기의 oob 평가를 평균
- 평가 점수 결과는 oob_score_ 변수에 저장되어 있다.

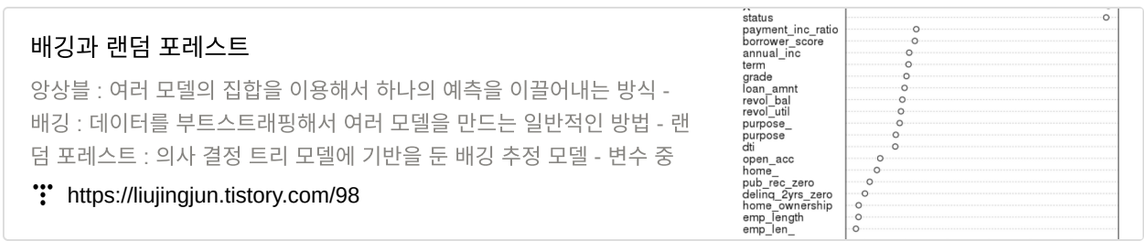
```
>>> bag_clf = BaggingClassifier(
...     DecisionTreeClassifier(), n_estimators=500,
...     bootstrap=True, n_jobs=-1, oob_score=True)
...
>>> bag_clf.fit(X_train, y_train)
>>> bag_clf.oob_score_
0.90133333333333332
```

Feature Selection

- Feature 를 섞어도 OOBE가 동일하다면 중요한 Feature 가 아니다.
- Feature 를 섞을시 OOBE가 변경된다면 중요한 Feature다.
- model 의 feature_importances_ 변수에 저장된다.

ExtraTree(Extremely Randomized trees)

- 극단적으로 무작위한 Tree의 Random Forest
- 편향은 늘어나지만 분산은 낮다.
- ?????



특성 중요도 (Feature Importance)

- 랜덤 포레스트의 장점은 특성(feature)의 상대적인 중요도를 측정하기 쉽다
- Scikit-Learn에서는 어떠한 특성을 사용한 노드가 불순도(impurity)를 얼마나 감소시키는지를 계산하여 각 특성마다 상대적 중요도를 측정한다. 불순도에 대해서는 **여기**를 참고하면 된다.
- Scikit-Learn의 `RandomForestClassifier`에서 `feature_importances_` 변수를 통해 해당 특성의 중요도를 확인할 수 있다.

7.2 배깅(bagging, bootstrap aggregating)과 페이스팅(pasting)

- 같은 알고리즘을 사용하고 훈련 세트의 서브셋을 무작위로 구성하여 분류기를 각기 다르게 학습
- 사이킷런의 배깅과 페이스팅
  - 사이킷런 API: BaggingClassifier(회귀의 경우에는BaggingRegressor)

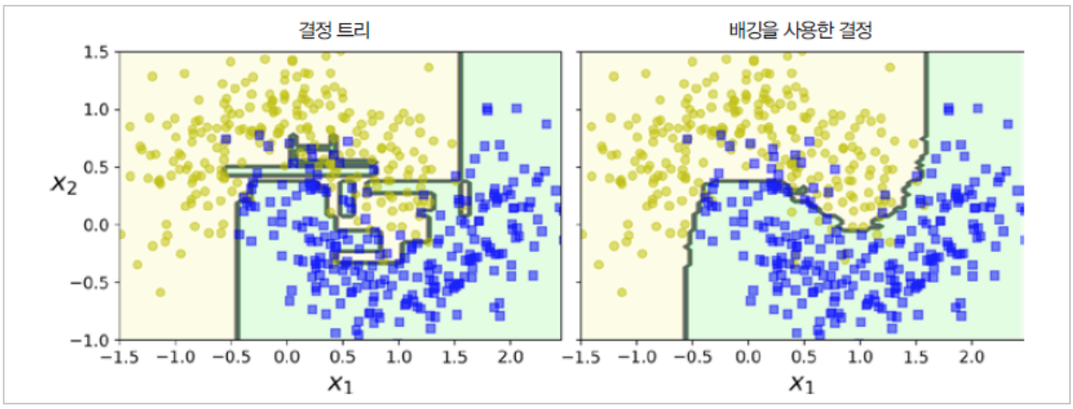


그림 7-5 단일 결정 트리(왼쪽)와 500개 트리로 만든 배깅 앙상블(오른쪽) 비교

7.3 랜덤 패치와 랜덤 서브스페이스

BaggingClassifier는 특성 샘플링도 지원

- 작동 방식은 max_samples, bootstrap과 동일하지만 샘플이 아니고 특성에 대한 샘플링
  - 샘플링은 max_features, bootstrap_features 두 매개변수로 조절
  - 각 예측기는 무작위로 선택한 입력 특성의 일부분으로 훈련
- (이미지와 같은) 매우 고차원의 데이터셋을 다룰 때 유용
- 랜덤 패치 방식
  - 훈련 특성과 샘플을 모두 샘플링
- 랜덤 서브스페이스 방식
  - 훈련 샘플을 모두 사용하고(bootstrap=False이고 max_samples=1.0로 설정) 특성은 샘플링(bootstrap_features=True 그리고/또는 max_features는 1.0보다 작게 설정)
- 특성 샘플링은 더 다양한 예측기를 만들며 편향을 늘리는 대신 분산을 낮춤

코드

- (최대 16개의 리프 노드를 갖는) 500개 트리로 이뤄진 랜덤 포레스트 분류기를 여러 CPU 코어에서 훈련시키는 코드

```
from sklearn.ensemble import RandomForestClassifier

rnd_clf = RandomForestClassifier(n_estimators=500, max_leaf_nodes=16, n_jobs=-1)
rnd_clf.fit(X_train, y_train)

y_pred_rf = rnd_clf.predict(X_test)
```