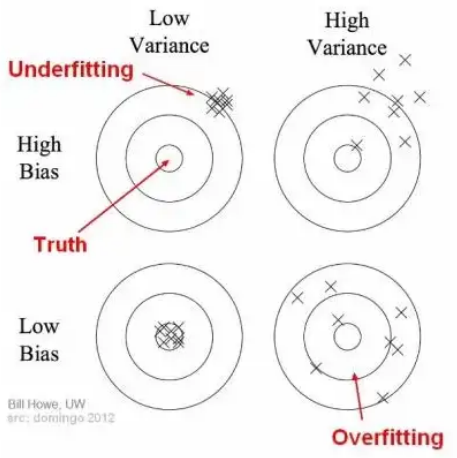
Training (2)

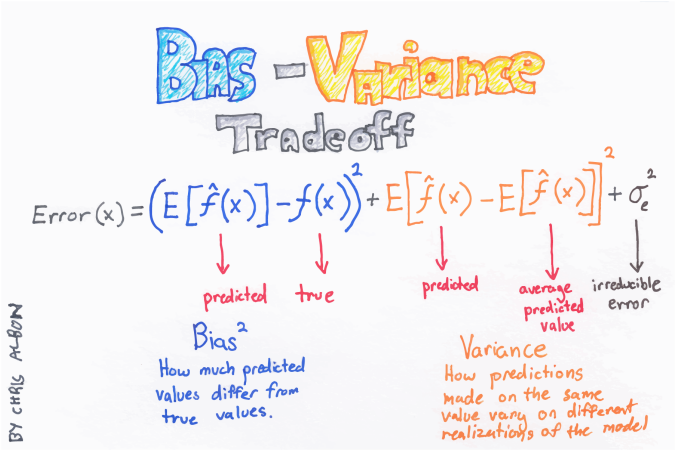
Variance-Bias

Bias = 예측값과 실제 정답과의 차이의 평균. 편향이 크면 모델이 지나치게 단순해짐

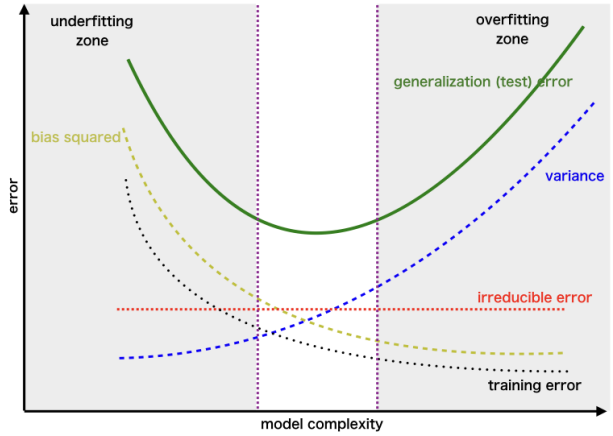
Variance = 다양한 데이터셋에 대하여 예측값이 얼만큼 변화할 수 있는지에 대한 양. 분산이 높으면 일반화가 잘 이루어지지 않음



Bias-Variance Tradeoff

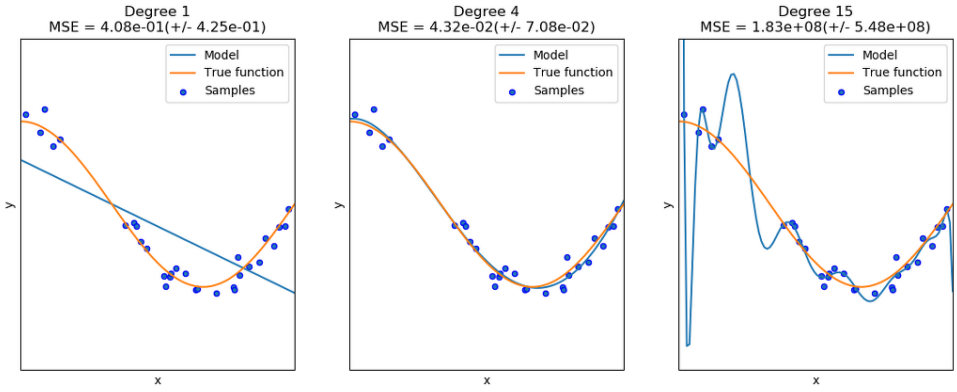


- irreducible error = 근본적으로 줄일 수 없는 에러. 모델이 항상 완벽할 수는 없기 때문에 추가된 항



Diagnosing Model

: Bias와 variance는 모델에 있어서 가장 핵심적으로 적용함. learning curves (학습 곡선) 활용해서 모델 진단 ⇒ optimal fit, underfit (bias가 높음), overfit (variance가 높음)



underfit/optimal fit/overfit

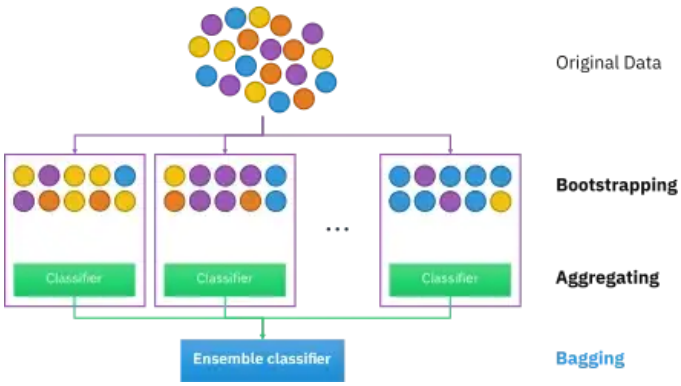
Ensemble

: 여러 모델 (”weak learner”) 을 합쳐 더 나은 결과를 얻고자 하는 머신러닝 기법

대표적으로 bagging, boosting, stacking 기법이 있음

1) Bagging (bootstrap aggregation)

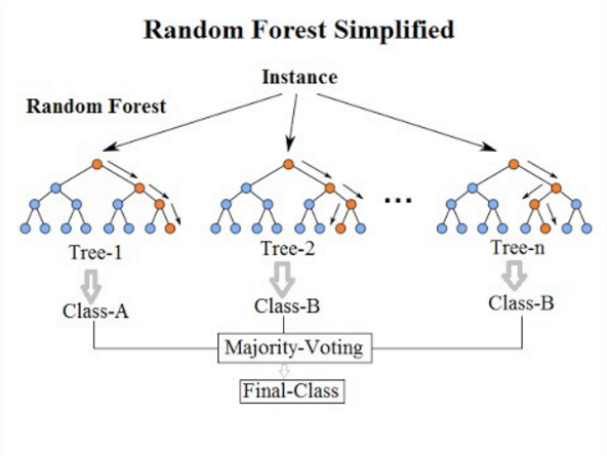
: 샘플을 여러번 뽑아서 각 모델을 독립적으로 학습한 뒤 결과를 집계하는 것.



- 카테고리 데이터는 투표 방식으로 집계함 Ex) 6개의 결정 트리 모델 중 4개는 A로 예측, 2개는 B로 예측한다면 A를 최종 결과로.
- 연속형 데이터는 평균으로 집계함
- 부트스트랩 = 랜덤으로 샘플링을 하는 것. raw data의 분포를 추정할 때 사용할 수 있음 (머신러닝에서는 훈련데이터를 늘릴 수 있음)
- 부트스트랩을 집계함으로써 학습 데이터가 충분하지 않더라도 충분한 학습효과를 주어, underfitting문제나 overfitting 문제 해결에 도움을 줌

Random Forest

: 대표적인 bagging 알고리즘. 여러개의 결정트리 기반으로



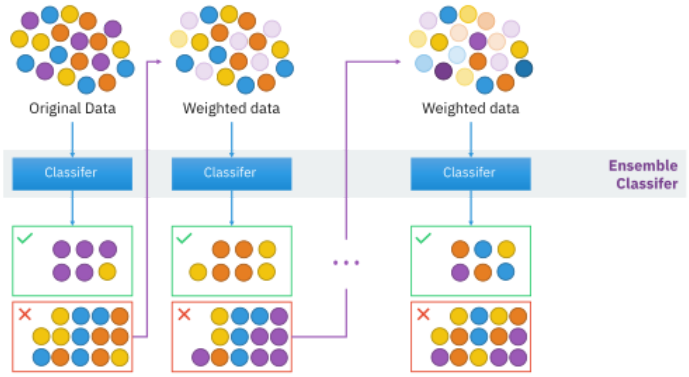

```
# Classifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import make_classification
X, y = make_classification(n_samples=1000, n_features=4,
...                            n_informative=2, n_redundant=0,
...                            random_state=0, shuffle=False)
clf = RandomForestClassifier(max_depth=2, random_state=0)
clf.fit(X, y)
RandomForestClassifier(...)
print(clf.predict([[0, 0, 0, 0]])) #[1]

# Regressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
model = RandomForestRegressor(n_estimators = 10, random_state = 0)
model.fit(X_train, y_train)
model.predict(X_test)
```

- 장점: overfitting 줄임, 정확도 개선. 카테고리/연속형 상관 없이 좋은 성능을 냄, 변수에 대한 중요도를 추출할 수 있어 feature selection에 유용함
- 단점: 파라미터/random seed 조절 외에는 크게 건드릴 수 있는 부분이 없고 결과에 대한 설명력이 떨어져 블랙박스 모델로 여겨지기도 함, 학습 오래걸림, CPU 할당량 높음

2) Boosting

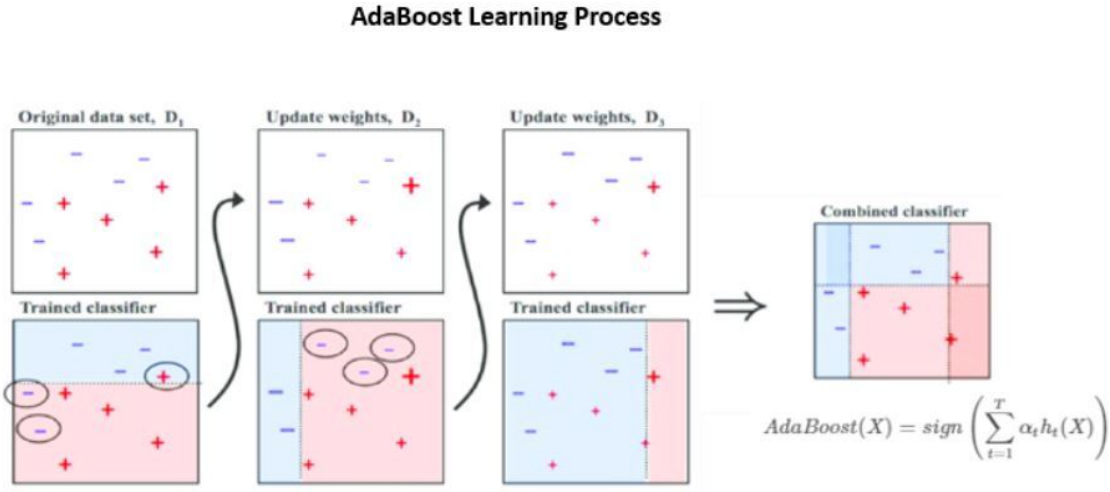
: 순차적으로, 복원추출로 가중치를 주는 것. Bagging과 비슷하지만, 순차적으로 학습이 진행됨. 즉 이전 분류기의 학습 결과를 토대로 다음 분류기의 학습 데이터의 샘플 가중치를 조정해서 학습을 진행하는 방법.



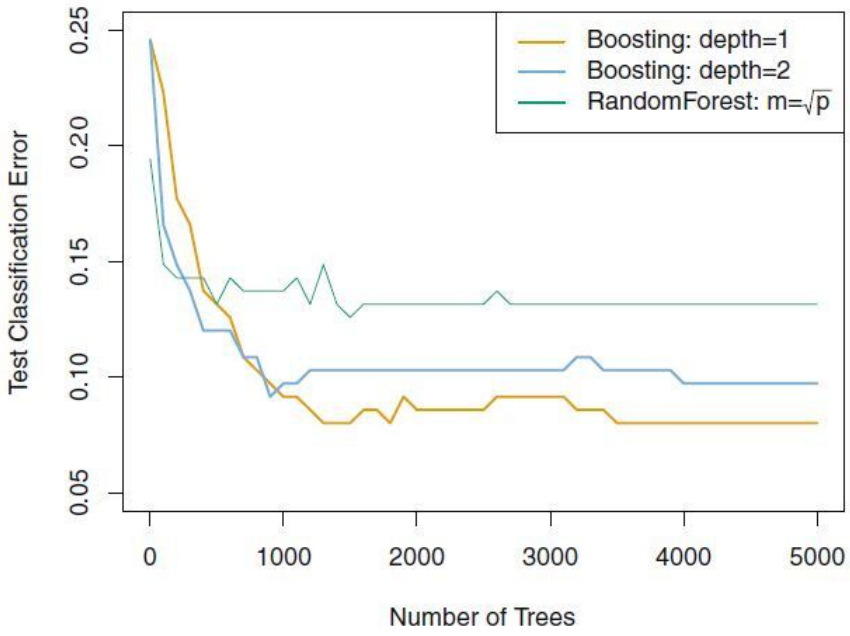
- 먼저 생성된 모델을 개선해 나가는 방향으로 학습이 진행됨
- 정확도가 높게 나오지만 outlier에 취약할 수 있음

AdaBoost (Adaptive Boosting)

: 대표적인 boosting 알고리즘



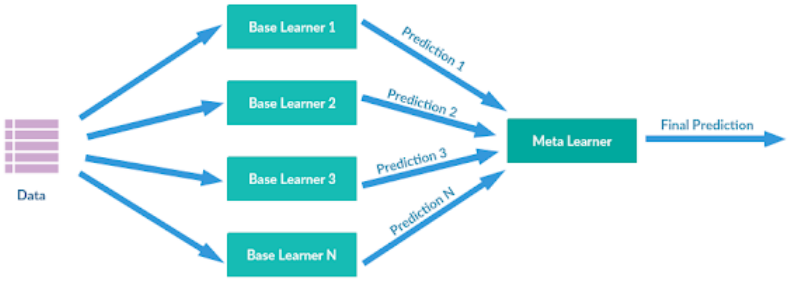

```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.datasets import make_classification
X, y = make_classification(n_samples=1000, n_features=4,
...                            n_informative=2, n_redundant=0,
...                            random_state=0, shuffle=False)
clf = AdaBoostClassifier(n_estimators=100, random_state=0)
clf.fit(X, y)
AdaBoostClassifier(n_estimators=100, random_state=0)
clf.predict([[0, 0, 0, 0]]) #array([1])
clf.score(X, y) #0.983
```



- 그 외: Gradient Boosting

3) Stacking

: cross validation 기반으로 서로 상이한 모델들을 조합함. 개별 모델이 예측한 데이터를 다시 meta data 형태로 사용해서 학습함



- “base learner”=개별 모델들, “meta learner”=최종모델
- 모든 base learner 모델들은 원본 데이터를 그대로 가지고 학습했기 때문에 overfitting 문제 발생 가능 (bagging과 boosting에서는 부트스트랩을 사용하기 때문에 overfitting 방지) ⇒ CV를 적용해 방지

참고

https://gaussian37.github.io/machine-learning-concept-bias_and_variance/

https://towardsdatascience.com/understanding-the-bias-variance-tradeoff-165e6942b229

https://medium.com/geekculture/random-forest-ensemble-method-860aaf4fcd16

https://data-analysis-science.tistory.com/61

https://scikit-learn.org