Week5 _ Ensemble . 2071 황인정

- 분류 (Classification)
  - 지도학습의 대표적 유형
    ↳ 명시적인 정답이 있는 데이터가 주어진 상태에서 학습하는 머신러닝 방식

⇒ 학습 데이터로 주어진 데이터의 피처와 레이블값 (결정값, 클래스값) 을 머신러닝 알고리즘으로 학습해 모델을 생성하고, 이렇게 생성된 모델에 새로운 데이터값이 주어졌을 때 미지의 레이블 값을 예측하는 것

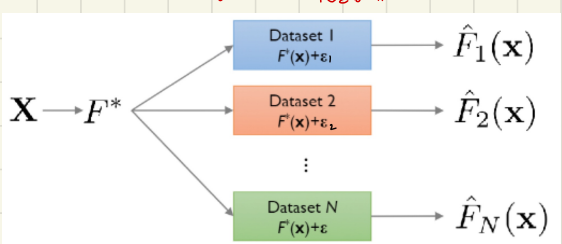
(ex) 나이브 베이즈, 퍼셉트론, Decision Tree, SVM, KNN, 앙상블

- 앙상블 (Ensemble) : 서로 다른 (또는 같은) 머신러닝 알고리즘을 결합한 것
  - 무조건적으로 최고의 성능인 알고리즘은 없음
    → 각 모델의 장점을 합쳐서 예측해보자 !

[배깅 (Bagging) (ex) 랜덤포레스트
[부스팅 (Boosting) (ex) GB, XGBoost

- Bias & Variance Decomposition

added model : $y = \underbrace{F^*(x)}_{\text{target func}} + \underbrace{\varepsilon}_{\text{자연발생 에러}}$, $\varepsilon \sim N(0, \sigma^2)$



$\bar{F}(x) = E[\hat{F}_D(x)]$

: 각 에러값에 따른 다른 예측값들의 평균

= N개의 예측값의 평균

- 특정 값 (x_0) 에서의 MSE

$Err(x_0) = E[\underbrace{y}_{\text{실제값}} - \underbrace{\hat{F}(x)}_{\text{예측값}} \mid x = x_0]^2$

↳ $y = F^*(x) + \varepsilon$

$= E[\hat{F}^*(x_0) + \varepsilon - \hat{F}(x_0)]^2$

$= E[\hat{F}^*(x_0) - \hat{F}(x_0)]^2 + \sigma^2$ ($\varepsilon \sim N(0, \sigma^2)$)

$= E[\hat{F}^*(x_0) - \bar{F}(x_0) + \bar{F}(x_0) - \hat{F}(x_0)]^2 + \sigma^2$

$= E[\hat{F}^*(x_0) - \bar{F}(x_0)]^2 + E[\bar{F}(x_0) - \hat{F}(x_0)]^2 + \sigma^2$

($\hat{F}^*(x_0)$: 정답값, $\bar{F}(x_0)$: 평균값 → 편향)

($\bar{F}(x_0)$: 평균값, $\hat{F}(x_0)$: 예측값 → 분산)

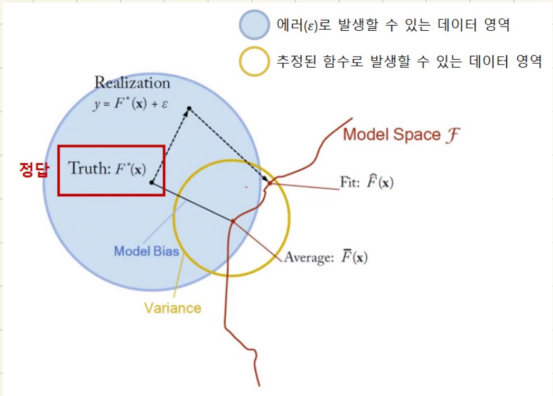
$= Bias^2(\hat{F}(x_0)) + Var(\hat{F}(x_0)) + \sigma^2$

① bias : 노이즈를 바꿔가며 모델링 했을 때 추정값들의 중심이 실제 데이터의 중심과 얼마나 떨어져있는가

⇒ 학습 알고리즘에 대한 잘못된 가정에서 비롯된 오류

② Variance : 노이즈를 바꿔가며 모델링했을 때 개별적인 모델링이 평균과 얼마나 차이가 나는가

③ σ^2 : 자연발생 노이즈 (감소 불가)



◯ : 실제 데이터의 영역 . 중심 = 정답

◯ : 우리가 추정한 함수가 뱉어내는 값들의 영역
중심 = 추정하는 값들의 기댓값

- 분산이 작음 = 노이즈가 변한다고해서 함수의 추정값이 크게 바뀌지 않음
- 편향이 작음 = 여러 데이터 셋을 바탕으로 반복적으로 추정하는 과정을 통해 전체적인 오차를 줄일 수 있음

- 모델의 복잡도 큼 → 작은 편향 & 높은 분산
  〃 작음 → 큰 편향 & 작은 분산

- 앙상블의 목적

① 분산 줄이기 : Bagging, RandomForest

② 편향 줄이기 : Adaboost

- 앙상블 사용 이유

: 앙상블 모델의 평균 에러

< 여러 개별 모델 사용 후 에러 평균

- 배깅 (Bagging)

: 각각의 분류기가 모두 같은 유형의 알고리즘 기반이지만, 데이터 샘플링을 서로 다르게 가져가면서 학습을 수행

- 목적 : 데이터의 다양성 확보

- 서브셋의 경우 중복을 허용해 무작위로 추출 (복원추출)
  = bootstrap

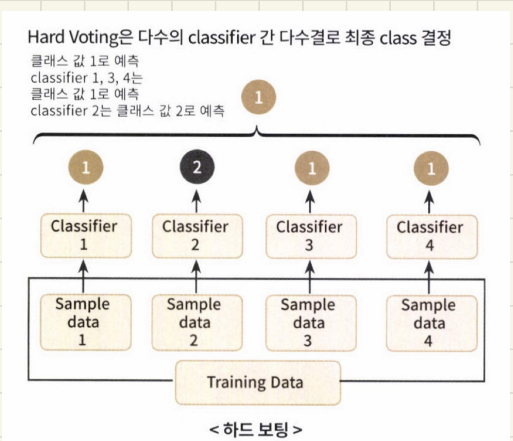
  → 모든 bootstrap에 포함되지 않는 데이터가 약 37% 발생함 = OOB 샘플

- OOB 샘플은 검증 데이터로 사용됨

- Aggregating

① Majority voting (hard voting)

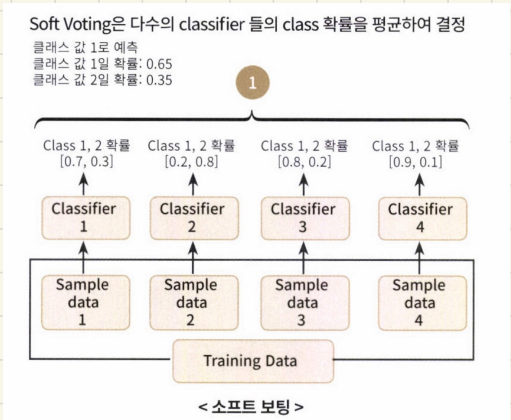
: 예측한 결과값들 중 다수의 분류기가 결정한 예측값을 최종 보팅 결과값으로 선정하는 것



< 하드 보팅 >

② Weighted voting (soft voting)

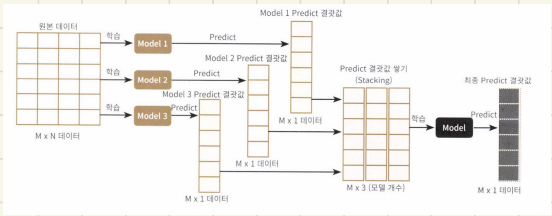
· 분류기의 레이블 값 결정 확률을 모두 더하고 이를 평균내서 이들 중 확률이 가장 높은 레이블 값을 최종 보팅 결과값으로 선정하는 것



< 소프트 보팅 >

③ Stacking

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



- 랜덤포레스트 (RandomForest)

: 여러 개의 결정 트리 분류기가 전체 데이터에서 배깅 방식으로 각자의 데이터를 샘플링해 개별적으로 학습을 수행 후 최종적으로 모든 분류기가 보팅을 통해 예측 결정을 함

⇒ 데이터가 중첩된 개별 데이터 세트에 결정 트리 분류기를 각각 적용하는 것

- 배깅 중 한 종류 + decision tree
- 변수 랜덤하게 선택해 사용
- 변수 중요도 산출 가능
  ↳ 변수가 모델의 예측에 영향을 주는 정도
  원래 데이터 셋의 OOB 에러 (e_m) 와 특정 변수 x_m 가 바뀌어진 데이터 셋의 OOB 에러 (p_m) 를 구해 그 차이를 중요도로 계산 (permutation)
  ↳ $d_m^m = p_m^m - e_m^m$
- 변수 중요도가 높다
  → p_m 와 e_m 의 값 차이가 크다
  → 이 값의 차이에 대한 편차가 크지 않아야 한다
- 특정변수 x_m 가 트리 split에 사용 X ⇒ $p_m = e_m$
  // O ⇒ $p_m > e_m$

• Boosting

: 여러 개의 분류기가 순차적으로 학습을 수행하되, 앞에서 학습한 분류기가 예측이 틀린 데이터에 대해서는 올바르게 예측할 수 있도록 다음 분류기에게 가중치를 부여하며 학습과 예측을 진행하는 것

- 특정 데이터셋에 대한 모델링이 끝나면, misclassified 된 데이터 찾은 후 이 데이터가 다음 학습에서 뽑힐 확률이 높아지도록 sampling

ex) AdaBoost : 오류데이터에 가중치 부여
GBM : 가중치 업데이트에 경사하강법 이용

• GBM

- 경사하강법

: 오류식 $h(x) = y - F(x)$ 를 최소화하는 방향성을 가지고 반복적으로 가중치 값을 업데이트하는 것

- GBM 과정

① target y의 평균을 초기 추정치로 계산 후 잔차 계산
② feature 와 잔차를 대상으로 의사결정나무 생성 } 약한 학습기
③ feature 를 추정해 초기 추정치 업데이트
& 새로운 잔차 다시 계산
④ 새로운 잔차로 또 트리 생성
↳ 위 과정을 잔차가 더이상 줄지 않을 때까지 반복함

• XGBoost

: GBM의 최적화 버전
↳ GBM의 느린 수행 시간 및 과적합 규제부재 등의 단점을 해결

- 조기 종단 (Early stopping) 가능

: n_estimators 에 지정한 부스팅 반복 횟수에 도달하지 않더라도 예측 오류가 더 이상 개선되지 않으면 반복을 끝까지 수행하지 않고 종지 ⇒ 수행 시간 개선 가능

- 핵심적인 알고리즘

① Approximate algorithm

: 결정트리의 어떤 노드에 있는 데이터를 Bucket으로 나눈 뒤 각 Bucket 에서 가장 최적의 split point를 찾아내는 알고리즘

- 기존의 Basic exact greedy algorithm 과 달리 분산 처리가 가능하고, 대부분 최적의 분할을 찾진 못하지만 다양성 높이기 가능

② Sparsity-aware Split Finding

. 결측값이거나 One-hot 인코딩으로 인해 데이터의 어떤 값이 희소하게 존재하는 경우 해당 값을 갖는 데이터는 모두 한쪽으로 split 되도록 하는 알고리즘

- 배깅과 부스팅 차이점

① 배깅은 병렬 처리 , 부스팅은 순차적으로 처리

② 배깅은 OOB 데이터가 생길 수 있지만, 부스팅은 데이터를 전부 활용