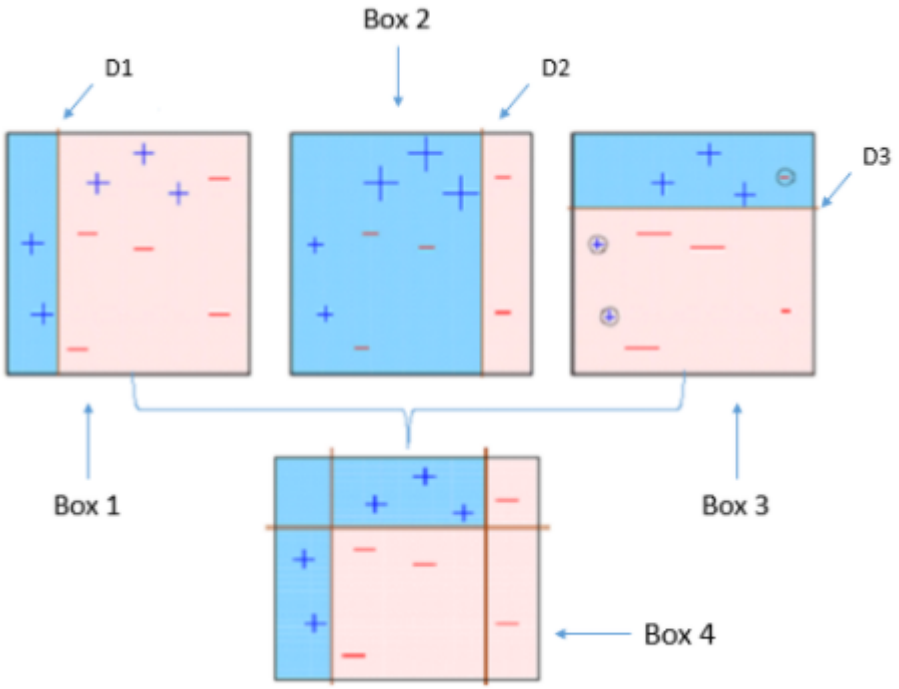
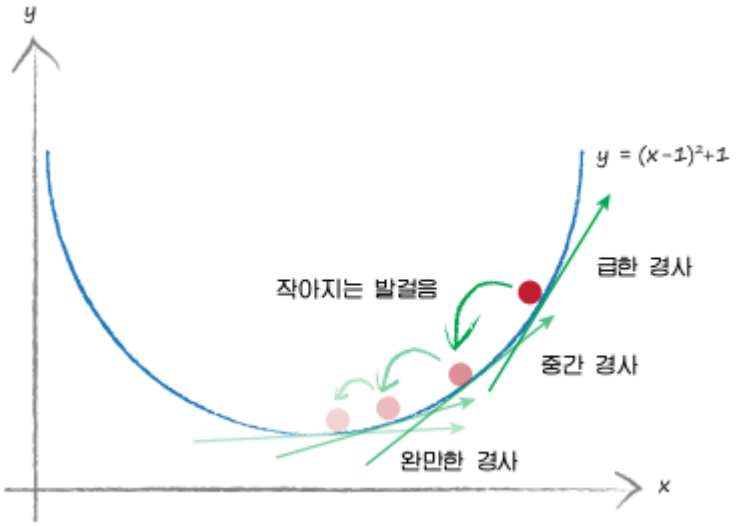
4.5 GBM(Gradient Boosting Model)

1. GBM의 개요 및 실습
    a. 부스팅 알고리즘: 여러개의 약한 학습기를 순차적으로 학습-예측하면서 잘못 예측한 데이터에 가중치를 부여를 통해 오류를 개선해 나가면서 학습하는 방식
    b. 부스팅의 대표모델
        i. 에이다 부스트: 오류에 데이터에 가중치를 부여하면서 부스팅을 수행하는 알고리즘
            1. 과정: 여러번의 약한 학습기로 분류하고, 각각의 오류 데이터에 대해서 가중치를 부여. 이러한 지표들을 모두 결합하여 미지의 데이터에 대해 예측 수행



            a. 개별 약한 학습기보다 결과가 훨씬 좋음
        ii. GBM: 에이다 부스트와 유사하나, 가중치 업데이트를 경사 하강법으로 이용하는 것이 큰 차이
            1. 오류값: 실제 값- 예측값

2. 경사 하강법: 오류식을 최소화하는 방향성을 가지고 반복적으로 가중치 값을 업데이트 하는 것



2. GBM 하이퍼 파라미터 및 튜닝
    a. loss: 경사 하강법에서 사용할 비용 함수
        i. 기본값: deviance
    b. learning_rate: 학습을 진행할때마다 적용하는 학습률, 기본값은 0.1. 만약 너무 적은 값이면, 업데이트되는 값이 작아서 최소 오류 값을 찾아 예측 성능이 높아질 가능성이 있고, 많은 값이면, 반복이 필요해서 수행시간이 오래 걸림 그렇기에 n_estimators와 상호 보완적으로 조합해 사용
    c. n_estimators: weak learner(약한 학습기)의 개수
    d. subsample: 약한 학습기가 학습에 사용하는 데이터의 샘플링 비율. 기본값은 1, 전체 학습 데이터를 기반으로 학습. 과적합 염려시 subsample을 1보다 작은 값으로 설정
    e. 과적합에도 강한 뛰어난 예측 성능을 자랑함. 그렇지만 수행시간이 너무 오래걸린다는 단점이 존재
        i. 병렬학습이 되지 않기 때문->아무리 성능이 좋아도 하나씩만 돌아가서 시간 낭비가 큼
    f. 이 모델을 기반으로 한 ML 패키지는 XGBoost와 LightGBM

4.6 XGBoost(eXtra Gradient Boost)

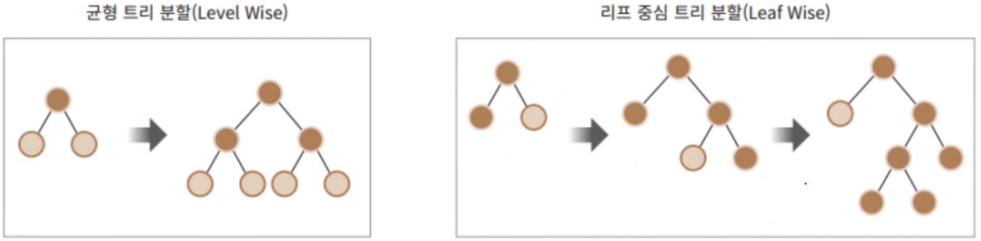
1. XGBoost 개요
    a. GBM에 기반하지만, GBM의 단점인 느린 수행 시간 및 과적합 규제 부재 등의 문제를 해결해서 매우 각광받고 있음
        i. 단, GBM에 비해 빠르지만, 다른 ML 패키지 보다 빠르다는 것은 아님
    b. 병렬 CPU 환경에서 병렬 학습이 가능
    c. 자체 과적합 규제 기능 존재
    d. 나무 가지치기 (Tree Pruning):더 이상 긍정 이득이 없는 분할을 가지치기 해서 분할 수를 더 줄이는 추가적인 장점을 가지고 있음

e. 자체 내장된 교차 검증: 반복 수행시마다 내부적으로 학습 데이터 세트와 평가 데이터 세트에 대한 교차 검증을 수행 + 최족화시 중간에 멈출 수 있는 조기 중단 기능 존재

2. 파이썬 래퍼 XGBoost 하이퍼파라미터
   a. 기존 GBM에 조기중간(early stopping), 과적합 규제를 위한 하이퍼 파라미터 등이 추가
      i. 일반 파라미터: 일반적으로 실행 시 스레드의 개수나 silent 모드 등의 선택을 위한 파라미터로서 디폴트 파라미터 값을 바꾸는 경우는 없음
      ii. 부스터 파라미터: 트리 최적화, 부스팅, 규제 등 관련 파라미터 지정
      iii. 학습 태스크 파라미터: 학습 수행시의 객체 함수, 평가를 위한 지표를 설정하는 파라미터
         1. 대부분의 하이퍼 파라미터는 부스터 파라미터에 속함
   b. 조기 중단 기능: 일정 횟수를 반복하는 동안 학습 오류가 감소하지 않으면, 더 이상 부스팅을 진행하지 않고 종료
      i. 예) early_stoppings이 50이면 50회를 반복하는 동안 더이상 학습 오류가 감소하지 않으면 더이상 부스팅을 진행하지 않고 종료
3. 파이썬 래퍼 XGBoost 적용 - 위스콘신 유방암 예측
   a. XGBoost 하이퍼파라미터는 주로 딕셔너리 형태
   b. 조기 중단을 수행하기 위해서는?
      i. eval_set(평가용 데이터 세트),eval_metric(성능평가방법)이 설정되어야 함
   c. plot_importance()를 통해 바로 피처 중요도를 시각화 할 수 있읍

4.7 LightGBM

1. LightGBM 설치
   a. XGBoost의 약점을 극복하기 위해 등장 -> 학습 시간이 훨씬 적음
   b. 일반 GBM 계열의 트리 분할 방법과 다르게, 리프 중심 트리 분할 방식을 사용하고 있음
      i. 기존의 대부분 트리 기반 알고리즘은 트리의 깊이를 효과적으로 줄이기 위한 균형트리 분할 방식 시용. 즉, 최대한 균형 잡힌 트리를 유지하면서 분할하기 때문에 트리의 깊이가 최소화 될 수 있음 -> 과적합에 더 강한 구조를 가질 수 있다고 알려져 있음
      ii. 트리 중심의 단점은, 균형을 맞추기 위한 시간이 필요
      iii. 리프 중심 방식은 트리의 균형을 맞추지 않고, 트리의 깊이가 깊어지고 비대칭적인 규칙 트리가 생성 -> 학습을 반복할 수 록 균형 트리 분할 방식보다 예측 오류 손실을 최소화 할 수 있음



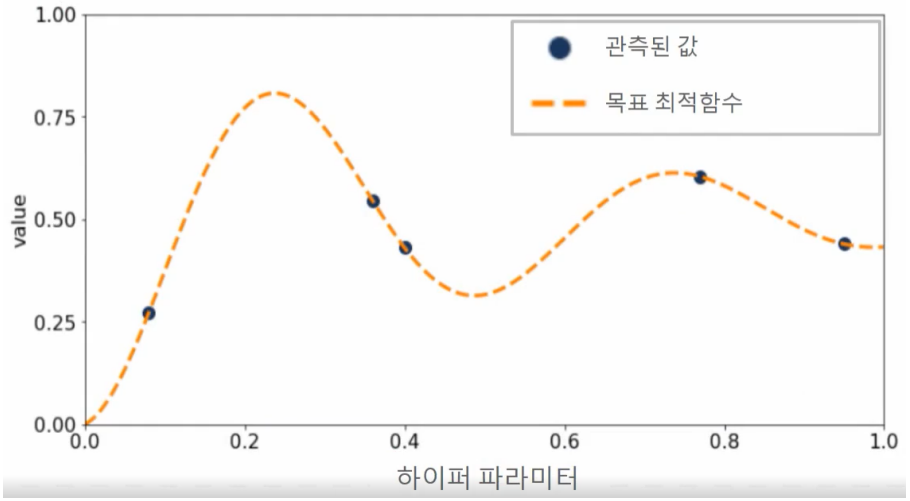
      iv.

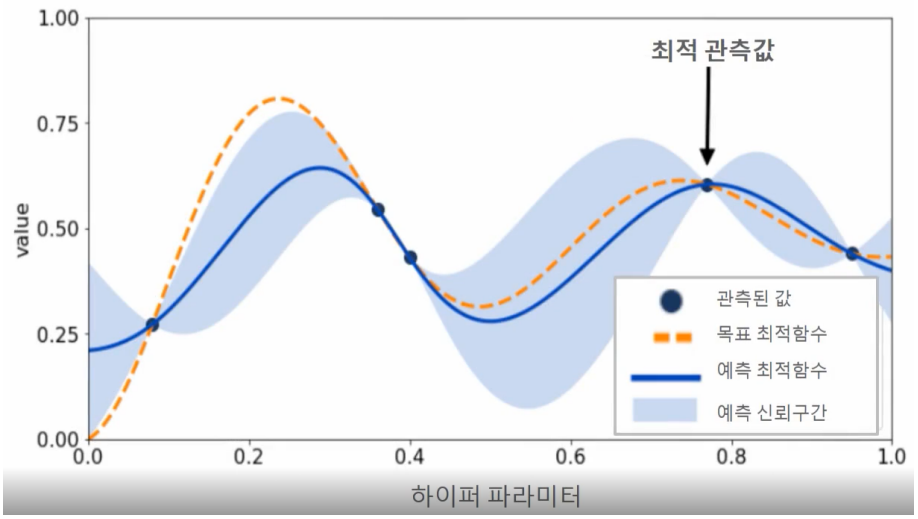
c. XGBoost와 마찬가지로 plot_importance()를 통해 바로 피처 중요도를 시각화할 수 있음
d. XGBoost 대비 LightGBM의 장점
    i. 더 빠른 학습, 예측 시간
    ii. 더 적은 메모리 사용량
    iii. 카테고리형 피처의 자동 변환과 최적 분할(원-핫 인코딩을 사용하지 않고도 카테고리형 피처를 최적으로 변환하고, 이에 따른 노드 분할 수행)

4.8 베이지안 최적화 기반의 HyperOpt를 이용한 하이퍼 파라미터 튜닝

1. 하이퍼 파라미터 튜닝 수행 방법
    a. Grid Search
    b. Random Search
    c. Bayesian
    d. Optimixation
    e. 수동 튜닝
    f. 등
2. 하이퍼 파라미터 튜닝의 주요 이슈
    a. Gradient Boosting 기반 알고리즘은 튜닝해야 할 하이퍼 파라미터 개수가 많고 범위가 넓어서 가능한 개별 경우의 수가 너무 많음
        i. 경우의 수가 많을 경우 데이터가 크면 하이퍼 파라미터 튜닝에 굉장이 오랜 시간이 투입되어야 함
    b. Grid Search CV는 수행 시간이 너무 오래 걸림
        i. 개별 하이퍼 파라미터들은 Grid 형태로 지정하는 것은 한계가 존재
            1. 데이터 세트가 작을 떄 유리
    c. Ramdomized Search는 Grid Seacrch CV를 랜던하게 수행하면서 수행 시간은 줄여주지만, Random한 선택으로 최적 하이퍼 파라미터 검출에 태생적 제약(데이터 세트가 클 때 유리)
    d. 두가지 방법 모두 iteraion 중에 어느정도 최적화 된 하이퍼파라미터들을 활용하면서 최적화를 수행할 수 없음
3. Baysian 최적화가 필요한 순간
    a. 가능한 최소의 시도로 최적의 답을 찾아야 할 경우
    b. 개별 시도가 너무 많은 시간/자원이 필요할 때
4. Baysian 최적화 개요
    a. 베이지안 최적화는 알려지지 않은 목적함수(블랙 박스 형태의 함수)에서, 최대 또는 최소의 함수 반환값을 만드는 최적해(최적 입력값)를 짧은 반복을 통해 찾아내는 방식
    b. 베이지안 확률에 기반을 두고 있는 최적화 기법
        i. 베이지안 확률이 새로운 데이터를 기반으로 사후 확률을 개선해 나가듯이, 베이지안 최적화는 새로운 데이터를 입력받았을 때 최적 함수를 예측하는 사후 모델을 개선해 나가면서 최적 함수 모델을 만들어 낸다.

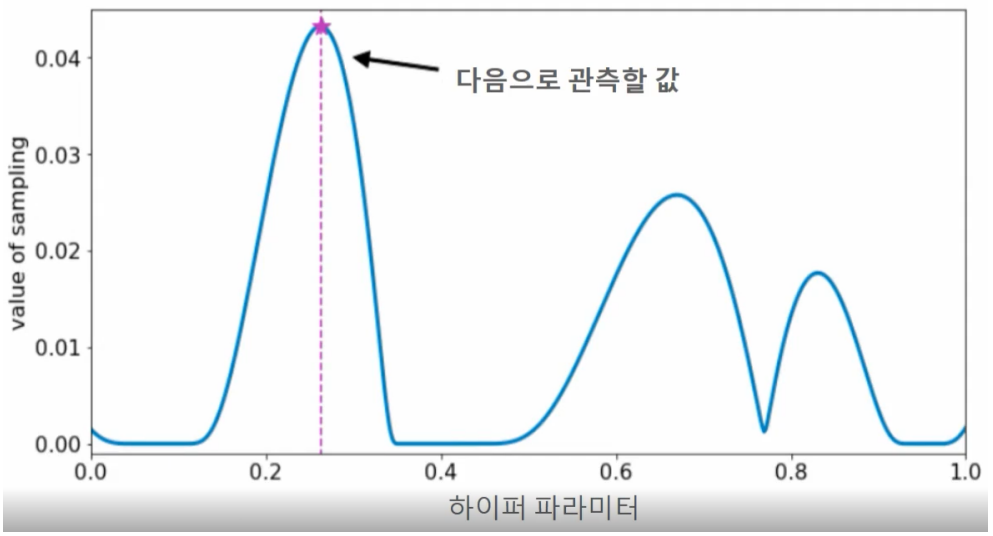
c. 베이지안 최적화의 주요 요소ㅓ
    i. 대체 모델(surrogate Model)
    ii. 획득 함수(Aquisition Function)
        1. 대체 모델은 획득 함수로부터 최적 함수를 예측할 수 있는 입력값을 추천 받은 뒤 이를 기반으로 최적함수 모델을 개선해 나가며, 획득 함수는 개선된 대체 모델을 기반으로 최적 입력값을 계산하는 방식
        2. 이때 입력 값은 하이퍼 파라이미터이다. 즉, 대체 모델은 획득 함수가 계산한 하이퍼파라미터를 입력받으면서 점차적으로 모델 개선을 수행, 획득 함수는 개선된 대체 모델을 기반으로 더 정확한 하이퍼 파라미터를 계산

5. 베이지안 최적화 프로세스
    a. 최초에는 랜덥하게 하이퍼 파라미터들을 샘플링하고 성능 결과를 관측
        i. 

        ii. 검은색 원은 특정 하이퍼 파라미터가 입력되었을 때 관측된 성능 지표 결과값을 뜻하며 주황색 사선은 찾아야 할 목표 최적함수 이다
    b. 관측된 값을 기반으로 대체 모델은 최적 함수를 추정
        i. 

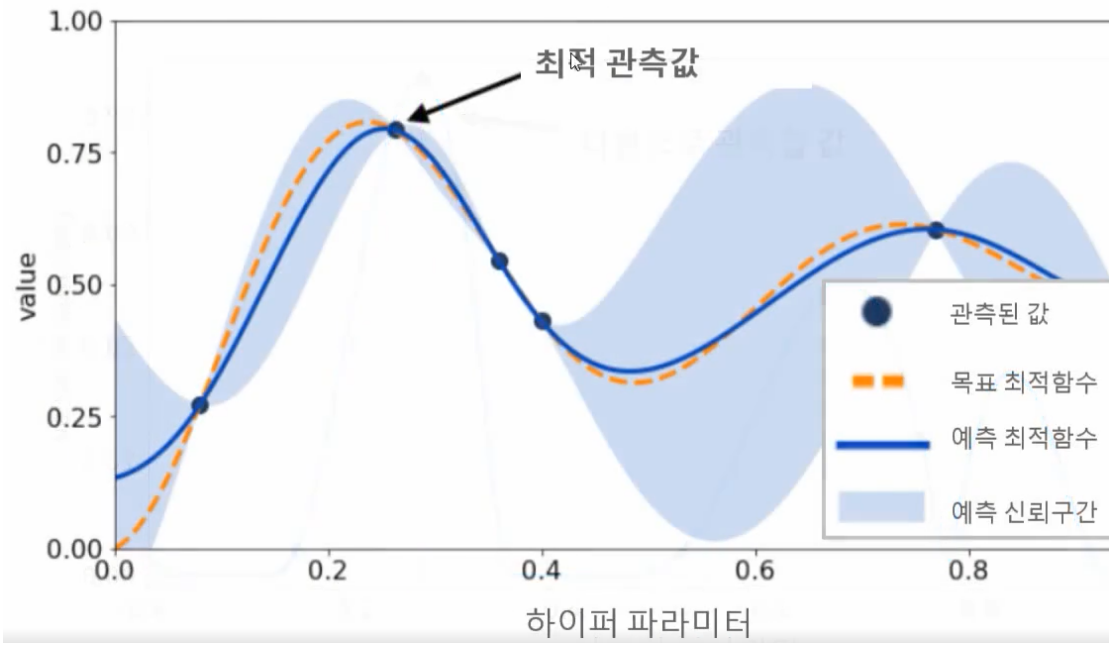
        ii. 파란색 실선은 대체 모델이 추정한 최적 함수이다. 하늘색 영역은 예측된 함수의 신뢰 구간으로, 추정된 함수의 결과값 오류 편차를 의미하며 추정 함수의 불확실성을 나타낸다
        iii. 최적 관측값은 y푹 value에서 가장 높은 값을 가질떄의 하이퍼 파라미이다

c. 추정된 최적 함수를 기반으로 획득 함수에서 다음으로 관측할 하이퍼 파라미터 추천

i. 

ii. 획득 함수는 이전의 최적 관측값보다 더 큰 최대값을 가질 가능성이 높은 지점을 찾아서 다음에 관측할 하이퍼 파라미터를 대체 모델에 전달한다

d. 획득 함수로부터 전달된 하이퍼 파라미터로 관측도니 값을 기반으로 대체 모델은 갱신되어 다시 최적함수 예측 추정

i. 

e. 이런 방식으로 c와 d의 과정을 특정 횟수만큼 반복하게 되면 대체 모델의 불확실성이 개선되고 점차 정확한 최적 함수 추정이 가능해진다

4.11 스태킹 앙상블

1. 스태킹: 개별적인 여러 알고리즘을 서롷 결합해 예측 결과를 도출한다는 점에서 앞에 소개한 배깅 및 부스팅과 공통점을 가지고 있음

2. 개별 알고리즘으로 예측한 데이터를 기반으로 다시 예측을 수행한다는 점에서 큰 차이가 존재
    a. ->개별 알고리즘의 예측결과 데이터 세트를 최적적인 메타 데이터 세트로 만들어 별도의ML 알고리즘으로 최종 학습 수행하고 테스트 데이터를 기반으로 다시 최종 예측을 수행하는 방식
    b. ->개별 모델의 예측도니 데이터 세트를 기반으로 하여 다시 학습하고 예측하는 바식을 메타 모델이라고 함
        i. 메타 모델이 사용할 학습 데이터 세트와 예측 데이터 세트를 개별 모델의 예측 값들을 스태킹 형태로 결합하는 것이 핵심
3. 스태킹에 필요한 모델
    a. 개별적인 기반 모델
    b. 최종 메타 모델
        i. 스태킹 모델의 핵심은 여러 개별 모델의 예측 데이터를 각각 스태킹 형태로 결합해 최종 메타 모델의 학습용 피처 데이터 세트와 테스트용 피처 데이터 세트를 만드는 것
    c. 스태킹 앙상블의 전개 과정
    d. CV 기반 스태킹 앙상블
        i. 과적합을 개선하기 위해 최종 메타 모델을 이한 데이터 세트를 만들 때 교차 검증 기반으로 예측된 결과 데이터 세트 이용