Voting, Bagging, Random Forest, Boosting, AdaBoost, Gradient Boost 요약 정리

- Voting

  : 여러 알고리즘이 도출해낸 결과물을 조합하여 최종 투표를 통해 결정하는 방식.

  단순 다수결을 따르는 Hard voting과 각 class별로 모델들이 예측한 probability를 합산해서 가장 높은 class를 따르는 soft voting으로 나뉘며, soft voting이 보다 합리적.

- Bagging (Bootstrap Aggregating)

  : 데이터셋에서 여러 번 복원추출한 서로 다른 샘플 데이터셋으로 모델을 학습, 이에 대한 투표로 결정하는 방식. Variance를 줄이고 overfitting 방지에 도움을 준다.

- Random Forest

  : 의사결정나무를 이용한 bagging 방식. tree간 상관성이 높아지는 것을 막기 위해(다양성을 확보하기 위해) 변수를 임의추출하여 사용한다.

- Boosting

  : 여러 분류기가 **순차적으로** 학습을 수행한다. 잘못 예측한 데이터에 가중치 부여하는 방식으로 오차를 보완하기에 예측하기 어려운 데이터의 정확한 예측이 가능하지만, 순차적이기에 병렬 처리에 어려움이 있고 다른 앙상블 대비 학습 시간이 길다.

- AdaBoost

  : 가중치를 부여한 약분류기를 모아 강분류기를 생성하는 기법. 특정분류기가 다른 분류기보다 더 중요한 것을 고려해 가중치를 부여한다.

- Gradient Boost

  : 약분류기들을 결합한 강분류기를 사용하는 것은 AdaBoost와 동일하지만 이와 달리 Stump 가 아닌, 하나의 Leaf Node 로 시작. 또한 잔차를 예측하는 모형을 만든다는 Residual Fitting 방식을 사용한다.

Ensemble Methods 중 XGBoost, LightGBM, CatBoost에 대하여

<XGBoost>

Gradient Boosting이 지닌 문제를 해결하고자 만들어진 것이 XGBoost이다. GBM은 속도가 느리고 overfitting될 수 있다는 문제를 가지고 있다. 이를 해결하기 위해 만든 알고리즘이니 비교적 빠르고 과적합 방지가 가능하다는 점이 특징이다. 분류, 회귀 문제에 모두 적용 가능하며 조기 종료를 설정할 수 있다. 이는 GBM을 기반으로 overfitting 방지를 위한 Regularization term, 규제를 포함한다.

다만 하이퍼 파라미터가 굉장히 많다. 모델의 성능이 고도화되며 모델 자체가 '무거워진' 것이다. 추가된 파라미터에는 앞서 언급한 규제에 관한 것들 등이 있다. 이 역시 grid search를 통해 튜닝이 가능하지만 이러한 이유로 시간이 오래 걸린다.

조기 종료라는 기능이 여전히 느리다는 특징을 보완하는데, 지정한 횟수만큼 training했음에도 모델의 성능이 좋아지지 않았다면 기존의 모델을 사용하기로 하고 일찍 학습을 종료하는 것이다. 이를 통해 그렇지 않아도 무거운 모델을 돌려 시간낭비하는 것을 방지한다. 물론 너무 적은 값을 선택하면 성능이 좋아지기 전에 training이 멈추니 early_stopping_rounds 값을 잘 지정해야 한다.

<LightGBM>

LightGBM은 GBM을 보완한 XGBoost의 단점에서 탄생한 알고리즘이다. 여전히 느리다는 점, 하이퍼 파라미터가 많아 튜닝이 어려운 점을 보완해, LightGBM은 대용량 데이터 처리를 더 빨리 더 적은 자원을 이용해 가능케 한다. 다만 데이터 수가 적으면 과적합 문제가 발생할 수 있다는 단점이 있다.

이는 트리의 분할 방식에서 이전까지 알고리즘과 가장 큰 차이를 갖는다. 기존 GBM이나 이를 바탕으로 둔 XGBoost는 트리를 level-wise하게 분할하는데, lightGBM은 leaf-wise하게 분할하는 것이다. tree의 depth는 줄이지만 균형을 잡고자 연산이 추가되어 손실이 생기는 이전까지의 방식과 달리, leaf-wise한 분할은 균형을 맞추지 않기에 비대칭적이고 깊은 트리를 생성하나 손실이 줄어든다.

LightGBM의 하이퍼 파라미터는 XGBoost와 거의 유사한 대신 leaf-wise 분할에 대한 파라미터가 추가되었다. 마찬가지로 조기 종료 설정이 가능하다.

<CatBoost>

CatBoost는 XGBoost와 같이 level-wise하게 트리를 분할한다. 또한 기존의 부스팅 모델과 잔차를 계산하는 방식에 있어 차이가 있는데, 기존 모델이 일괄적으로 모든 훈련 데이터를 대상으로 잔차 계산을 했다면, Catboost는 일부만 가지고 계산해 이를 토대로 모델을 만들고, 그 뒤에 데이터의 잔차는 이 모델로 예측한 값을 사용한다. 이때 x1의 잔차를 계산해 x2의 잔차를 예측하고, x1, x2를 계산해 x3, x4를 예측하기에 순서에 따라 모델을 만든다는 점에서 이를 Ordered Boosting이라 한다.

이러한 Ordered Boosting때문에 학습 데이터의 순서도 고려해야만 한다. 데이터 순서를 섞어주지 않으면 매번 같은 순서대로 잔차를 예측하는 모델을 만들 가능성이 있기 때문이다. 따라서 데이터를 셔플링해서 뽑아내며, 이때 전체 데이터가 아닌 일부 데이터도 가져오게 설정할 수 있다. 이를 포함해 다양한 오버피팅 방지를 위한 기법을 포함하고 있다.

CatBoost도 한계를 지니는데, 먼저 Sparse Matrix(희소행렬)는 처리하지 못한다고 한다. 또한 데이터 대부분이 수치형 변수인 경우, Light GBM 보다 학습 속도가 느리다. 즉 대부분이 범주형 변수인 경우에 사용하는 것이 좋다.