LGBM(Light Gradient Boosting Machine) 모델

1. Gradient boosting : Gradient Descent + Boosting

부스팅(boosting)이란 단순한 학습기를 결합해서 보다 정확하고 강력한 학습기를 만드는 방식을 의미한다. 정확도가 낮더라도 일단 모델을 만들고, 드러난 예측 오류는 두번째 모델이 보완한다. 이 둘을 합치면 처음보다는 정확한 모델이 만들어지고, 그럼에도 여전히 남아 있는 문제는 다음 모델에서 보완하여 계속 더하는 과정을 반복하는 원리이다. Gradient Boosting을 알기에 앞서 손실함수는 예측 모델의 오류를 정량화해주는 함수이다. 학습이란 손실함수를 최소화하는 파라미터를 찾는 일이라고 할 수있다. Gradient Descent는 손실함수를 파라미터로 미분해서 기울기를 구하고, 값이 작아지는 방향으로 파라미터를 움직이다 보면 손실함수가 최소화 되는 지점에 도달한다. 그렇다면Gradient Boosting은 손실함수를 파라미터가 아니라 현재까지 학습된 모델 함수로 미분한다. 정리하면 Gradient Boosting에서는 Gradient가 현재까지 학습된 모델의 약점을 드러내는 역할을 하고, 다른 모델이 그걸 중점적으로 보완해서 성능을 Boosting한다.

1. LGBM(Light Gradient Boosting Machine) 이란?

Gradient boosting알고리즘을 사용하는 학습모델이며 기존의 GBM보다 더 빠르고 성능이 높은 학습모델이다.

1. 사용한 패키지

numpy: 다차원 배열 객체를 다루는데 유용한 함수를 제공하는 패키지이다.

pandas: 현실의 다량의 데이터를 상대적으로 라던지 이름을 부여하여 접근 한다던지 하여

직관적이고 쉽게 접근 처리할 수 있도록 도와준다.

lightgbm: gradient boosting을 사용하기 위한 패키지이다. 비슷한 예로는 xgboost, gbm이 있다.

1. 데이터의 전처리
   1. 제공받은 데이터에 있는 Feature들의 타입을 명시화한다.
   2. registration\_init\_time, expiration\_date를 각각 연월일로 세분화하여 feature에 추가 한다.
   3. 노래정보의 해당하는 isrc를 통해 해당 노래의 연도를 추출하여 feature에 추가한다.
   4. song\_length 중 에서 na인 원소를 평균값으로 채워준다.
   5. lyricist, composer, artist 중 에서 na인 원소를 각각 no\_lyricist, no\_composer,no\_artist로 바꾸어 준다.
   6. 각 노래당 작곡, 작사, 아티스트가 몇 명인지 세고 그 각각의 수를 feature에 추가한다.
   7. 해당 노래가 피쳐링되었는 지 안되었는 지도 새로운 feature로 추가한다.
   8. 아티스트와 작곡가가 같은 지도 artist\_composer라는 feature에 추가한다.
   9. 아티스트, 작사가, 작곡가가 같은 지도 artist\_composer\_lyricist라는 feature에 추가한다.
   10. 노래길이가 평균의 절반도 안될 경우 0, 평균 이하일 경우 1, 평균의 1.5배보다 작은 경우 2 나머지는 3으로 값을 같는 feature를 추가한다.
   11. 노래가 재생된 횟수를 feature에 추가한다.
   12. 해당 노래의 가수의 노래 재생 횟수를 feature에 추가한다.
2. LGBM의 파라미터
   1. Learning\_rate: 형식은 정수형. 각각의 반복마다 적용되는 축소비율.
   2. Boosting: 형식은 문자열. Boosting 타입.
   3. objective : 형식은 문자열. 학습할 과제를 규정한다. 예를 들면 'regression', 'binary(Classfication)', or 'lambdarank'.
   4. max\_depth: 트리 모델의 최대 깊이.
   5. num\_leaves: 한 트리에 있는 노드 수.
   6. metric: 최적화 할 측정 항목. 'l1'(절대 손실), 'l2'(제곱 손실), 'NDCG', 'AUC'(AUC), 'binary\_logloss'(로그 손실) 및 'binary\_error'(정확도) . 기본값은 'l2'.
   7. verbose : 형식은 bool. True로 두면 평가 셋이 사용되고 평가 진행률을 쓴다.
   8. baggig\_fraction: 형식은 double. 이것은 리샘플링없이 데이터의 일부를 무작위로 선택한다. 훈련 속도를 높이고 과다한 피팅을 처리하는 데 사용할 수 있다.
   9. bagging\_freq: 배깅에 대한 빈도 수. 0은 배깅을 비활성화 함을 의미한다. 예를들어 k라 설정하면 k 반복마다 bagging을 수행하는 것을 의미한다.
   10. bagging\_seed: 행 서브 샘플링의 무작위 시작 시드
   11. feature\_fraction: feature\_fraction이 1.0보다 작으면 각 반복에서 피쳐의 일부를 무작위로 선택한다. 예를 들어 0.8로 설정하면 각 트리를 학습하기 전에 80 % 의 특징을 선택한다.
   12. feature\_fraction\_seed: 열 서브 샘플링을위한 무작위 시작 시드
   13. Max\_bin: 특성 값이 버킷으로 묶일 수 있는 최대 bin수. bin 수가 적 으면 교육 정확도는 떨어지지만 일반 전원은 증가 할 수 있다. Overfitting을 막을 수 있다.
   14. Num\_rounds: 부스팅 반복횟수.
3. 과정

|  |  |  |
| --- | --- | --- |
| 경과 | 추가된 특징 | 정확도 |
| 1 | 제공받은 데이터에 있는 Feature로만 이용. | 62.0% |
| 2 | 제공받은 데이터에 있는 Feature들의 타입을 명시화. | 62.8% |
| 3 | 노래정보에 있는 노래의 연도를 Feature에 추가. | 64.2% |
| 4 | 유저들의 계정 등록일, 만료일을 년/월/일 각각의 Feature로 세분화. | 65.6% |
| 5 | 노래를 만든 아티스트, 작곡가, 작사의 숫자들을 각각의 Feature로 추가. | 66.8% |
| 6 | 노래 길이에 따라 부여한 점수를 Feature에 추가 . | 67.6% |
| 7 | 데이터 상에서 노래가 재생된 횟수를 Feature에 추가.. | 68.3% |
| 8 | 데이터 상에서 해당 가수의 노래가 재생된 횟수를 Feature에 추가. | 68.7% |

1. 결과

* Lgbm을 통해서 00%의 정확도를 기록하였다. 실제로 제출을 할 시에 68.71%로 정확도가 좀 더 떨어졌지만 전에 시도한 2가지 모델을 포함해서 제일 높은 정확도를 기록하였다. 처음에는 2가지 모델과 비슷한 정확도를 보여줬지만 유효한 Feature들을 추가하였을 때 정확도가 증가하였다. Learning rate은 0.2, 부스팅 횟수는 100일 때 실제 테스트 데이터에 대한 정확도가 제일 높았다. 이로써 3가지 모델 중에서 가장 유효한 모델은 Lgbm이 었다.