Project2 Wrap up Report

Title	Type	Date	Name
온라인 상점 고객 구매 예측	개인 프로젝트	2021.04.12 ~ 2021.04.22	T1138 윤준호

Result

- Leaderboard
  - AUC: **0.8552**
  - Rank: **44**/99

Task

- Data
  - 온라인 상점의 거래 log 데이터 (2009.12 ~ 2011.11)
- 고객별 구매액 300 초과 여부 예측 (2011.12)

Model & Hyperparameters

- Model
  - Light Gradient Boosting Machine
- Hyperparameters
  - metric: AUC
  - feature fraction: 0.8
  - bagging fraction: 0.8
  - bagging frequency: 1
  - n estimators: 10000
  - early stopping rounds: 100
  - train-validation set split: stratified 10-fold

Approaches

Model selection & ensemble

- Random Forest
  1. Validation AUC가 test AUC보다 높았고, 클래스 분포가 치우쳐 있었기 때문에 overfit 문제를 해결하는 것이 중요하다고 판단함. 예측이 틀린 데이터를 이용해 다음 트리를 생성하는 gradient boosting machine과는 다르게, 매번 랜덤 샘플링을 통해 트리를 생성하는 random forest 방식이 일반화 성능이 우월해 overfit 문제를 다루는데 적합하다고 판단해 모델 변경을 시도
  2. 내가 사용한 sklearn RandomForest의 경우, 결과값 형태가 기존 방식과 같은 probability가 아닌 decision (0 또는 1)이었음. 따라서 fold별 예측을 반영하지 못하고 극단적으로 이진 판단만 고려했기 때문에 성능 감소. (0.7289 cf. baseline 0.8100)
  3. predict()가 아닌 predict_proba() 함수를 사용해 결과값을 확률값 형태로 바꿔주고, 하이퍼파라미터 탐색을 통해 성능 향상 (0.8177)
     - n_estimator: 1000 (200~5000 실험)
     - max_depth: 16 (10~30 실험)
     - max_features: sqrt (sqrt/0.1/0.2/log2 실험)
  4. 하지만 feature engineering을 통해 새로운 특성 생성 후 기존 모델과 비교했을 때 성능이 더 낮았고, ensemble 결과도 기존 모델보다 낮았기 때문에 최종 제출은 기존 모델로 함
     - lgbm: 0.8552
     - random forest: 0.8430
     - ensemble (two above): 8515 (soft-voting)

EDA & Feature engineering

- Aggregation functions
  - customer를 기준으로 aggregation 시, 단순 월별 총 구매액뿐만 아니라 다양한 feature 변화 함수들을 적용 후 성능 향상
    - mean, max, min, sum, count, std, skew
  - 몇몇 feature에 대해서는 특수한 함수를 적용하기 위해 aggregation dict를 생성 후 key-value 지정
    - ‘month’: mode_func + agg_func (month 특성의 경우, mean, max, … 등의 agg 함수말고도 최빈값 mode 함수도 적용해 총 8개 feature 생성)
- Accumulation features
  - 단순 월별 총 구매액이 아니라, 시간이 지남에 따라 누적되는 정보를 담을 수 있는 accumulation feature를 생성해 성능 향상
    - customer id, product id, order id를 각각 기준으로 groupby 후 세 항목을 cumsum한 feature 생성
      - total, quantity, price
- nunique
  - order_id, product_id 각각의 unique 값 수를 feature로 생성 후 성능 향상
- Time series features
  - customer별 최대 구매한 year, month, 그리고 해당 month의 총 구매 가격을 feature로 생성 후 성능 향상
  - diff() 함수를 이용해 고객별 총 구매 가격, product price, quantity, 구매한 month 각각의 (직전 기록과의) 차이를 feature로 생성 후 성능 향상
- Description Word2Vec
  - 상품 상세 설명 text를 word2vec으로 수치화하여 feature로 생성해봤지만, 생성되는 feature 수를 줄이는 등의 실험에도 성능이 모두 하락해 제외함
- Average spend
  - 최초 구매 이후 매월 평균 사용 금액을 feature로 생성해봤지만 성능이 하락해 제외함
- Over 300 count
  - 클래스 1에 해당하는 month 수를 count해 feature로 생성해봤지만 성능이 하락해 제외함

ETC

- Threshold selection
  - 클래스 0 (300 미만)에 비해 클래스 1(300 이상)의 비율이 매우 적었기 때문에, 클래스 0에 학습이 overfit 되는 경향을 보일 수 있다고 의심해 학습과 예측에 대한 threshold를 300에서 290으로 낮춰 성능 향상
    - threshold: 290 (280/290/300 실험)
- Refund match drop
  - 구매한 month와 환불한 month가 서로 다른 경우에, 월별 학습과 예측에 있어 적지 않은 데이터 왜곡을 가져올 수 있다고 판단해, 같은 customer_id의 같은 product_id가 다른 month에서 구매되고 환불된 경우를 모두 찾아 둘 다 제거하려고 시도
  - 구매보다 환불이 먼저 발생한 경우 등 상식적으로 이해할 수 없는 케이스들이 존재해 더 알아보니, 구매 내역 log가 매일 24시간 모두 제공되는 것이 아니라 일부 시간에만 (7:30 AM ~ 17:30 AM) 잘려진 데이터였음. 따라서 데이터에 나타나지 않은 기록들이 존재할 수 있고, 1:1 매칭이 원천적으로 불가능하다고 판단해 포기함

Lessons

Ideas might have been helpful

- n개월 연속 구매 총합 0인 고객 제외, 최근 구매 날짜 feature
- 생필품 많이 사는 고객은 주기적으로 구매 확률 높음
- feature scaling: MinMaxScaler, StandardScaler
- distribution transformer: Quantile Transformer
- feature들을 PCA로 차원축소한 결과를 새로운 feature로 활용
- Time Series Nested Cross-Validation
- TEST001, TEST002 등의 의미 없는 값 제거

Thoughts

> 가장 아쉬운 점과 가장 뿌듯한 점이 동일한 프로젝트였다. 새로운 모델 (random forest)로 변경해 시도해본 것, 환불-구매 1:1 매칭을 찾아 제거하려고 시도해본 것 - 이 두 시도 모두 내가 스스로 밑바닥부터 가설을 세우고 구현해 검증까지 마쳤지만, 결과적으로 성능 향상을 이뤄낼 수 없었기 때문이다.
>
> 주 관심 분야는 Vision이지만, 역시 AI의 본토는 데이터분석이며 여전히 데이터를 이해하는 것이 얼마나 중요한 것인지 새삼 몸소 느껴볼 수 있는 경험이었다. 다양한 방식의 EDA와 Feature engineering을 실험으로 구성하고 결과를 확인하는 과정을 통해, 그동안 내가 데이터 자체보다 모델 구성에 너무 가중치를 두어 학습을 진행해왔다는 걸 알 수 있었다. 모델 학습을 잠시 멈추고, 나 자신의 학습에 대한 weight update가 필요한 시점인 것 같다.