Wrap-up Report

김종헌

최종 submission – **AUC 0.8640**, **5위**

개요

[문제 해결 전략] 파트에서는 이 문제를 보다 큰 그림에서 어떻게 해석할 수 있을지, 그리고 그 중에서 최적의 방법은 무엇이 될 수 있을지에 대한 저만의 고민 과정을 썼고, [피쳐 선택 전략] 파트에서는 [문제 해결 전략] 파트에서 정의 내린 문제를 풀기 위해 어떤 피처를 취하고, 어떤 피처를 버리고, 어떤 피처를 추가적으로 추출할지에 대해 생각해본 내용을 담았습니다. 대회 기간동안 생각한 순서대로 기록하였습니다.

문제 해결 전략

1. 분류인가? 회귀인가?

먼저 이 문제를 처음 마주했을 때 들었던 생각은, 이 문제가 회귀 문제이냐 분류 문제이냐를 확실히 짚고 넘어가야겠다는 점이었습니다. 물론 최종적으로 풀어야하는 문제는 분류 문제이지만 최종 결정을 모델이 바로 판단하는 것보다는, 직접적으로 구매 금액이 얼마일지 예측하는 회귀 문제를 먼저 푼 이후 회귀의 결과로 나온 금액이 300을 넘었냐 안넘었냐를 다시 판단하는 것이 좀 더 성능에 도움이 되지 않을까 라는 생각을 했습니다.

하지만 이 문제를 회귀 문제로 풀기에는 소비량의 스펙트럼 자체가 max값이 $100,000에 이를 정도로 너무 넓어 결국 분류로 풀기로 결정하였습니다. 이렇게 큰 범위에서 회귀를 하면 $300의 주변에 애매하게 걸친 값들에 대한 처리가 제대로 안될 것 같다는 생각이 들었기 때문입니다.

2. 시계열 데이터에 통상적으로 쓰이는 모델은 활용할 수 없다.

정말 오래 고민했던 부분입니다. 이 문제는 시계열 데이터를 예측하는 문제입니다. 따라서 이를 풀기 위해 유명한 시계열 회귀 모델인 ARIMA, Prophet를 활용할 수 있을 것이고, 더 나아가 딥러닝(RNN, LSTM, Transformer)을 활용할 수도 있었을 것입니다.

하지만 결국 위에서 나열한 시계열에 특화된 예측모델은 활용하지 않았습니다. 이러한 결정을

내리게 된 가장 결정적인 이유는 데이터 부족 때문이었습니다. 일단 딥러닝 모델은 일반적으로 데이터의 양이 어느정도 많아야 성능이 보장된다는 특성이 있습니다. customer 수는 총 5914명인데 이정도 수로는 딥러닝 모델을 학습시키기에 충분하지 않다고 판단했습니다.

하지만 딥러닝 기반 모델이 아닌 ARIMA와 같은 모델을 사용한다면 문제가 덜하지 않을까 싶었는데요, 사실 더욱 큰 문제는 데이터가 sparse하다는 점입니다. 보통 앞서 언급한 모델들은 주가 등의 여러 경제적 지표를 예측하기 위해 사용하며, 그런 지표들은 시간의 흐름에 따라 끊기지 않습니다. 하지만 주어진 데이터는 오히려 데이터가 빈 부분이 훨씬 많으며 이렇게 sparse한 데이터는 정밀한 시계열 예측에 적합하지 않다고 생각했습니다.

한편, 후술하겠지만 EDA를 어느정도 마친 후 저는 구매 데이터를 월별로 집계해서 사용해도 문제가 없다고 판단했고 오히려 이쪽이 코드를 짜기에도 더 편하다고 생각했습니다. 데이터를 월별로 집계해서 활용하면 앞에서 언급한 문제들이 더 심각해집니다. 결국 고객 한 명당 각 24개 (2009-12 ~ 2011-11) 데이터를 갖게 되고 역시 데이터가 턱없이 부족해집니다.

시계열 문제를 일반적인 정형 데이터 분류 문제로 풀게 되었다는 점이 못내 아쉽고 좀 걸리긴 했지만 그래도 성능이 어느정도 보장될 수 있는 트리 기반 모델들을 활용하기로 하였습니다.

3. train set/valid set을 어떻게 나누어야하는가?

이 의문에 대해서는 첫날부터 끝없이 고민했지만 결국 마지막까지도 해결하지 못했습니다. 토론 게시판에도 올라왔지만 시계열 문제의 특성상 단순히 stratified KFold로 CV를 할 경우 validation set의 신뢰도가 일정 수준부터는 급격히 떨어지게 됩니다. 실제로 문제를 풀 때 AUC 0.80정도까지는 validation set에서 나오는 AUC score가 의미가 있었으나, AUC 0.80부터 마지막 제출의 AUC 0.86이 될 때까지 validation set의 score는 거의 고정적이었고 따라서 이걸 통해 제 모델의 성능을 평가하기에는 어려움이 많이 따랐습니다. 결국 이것 때문에 해볼 수 있는 것들이 많이 줄어들었습니다. GridSearchCV, Optuna도 validation 기반으로 최적 파라미터를 산출해내기 때문에 이런 기법을 정상적으로 활용할 수 없었고, permutation importance 기법도 마찬가지 이유로 사용하기 애매한 부분이 있었습니다. 물론 이 문제를 해결하기 위한 방법으로 nested CV라는게 있다는 말을 듣기는 했으나 적용하지는 않았습니다. 제 생각으로는 2011-12와 시기가 비슷한 2011-11을 target으로 하는 validation이 가장 의미가 있다고 생각했고, 그 외 시기를 target date로 할 경우 모델이 좀 다른 방향으로 학습될 수도 있다는 우려가 있었기 때문입니다. 그래서 validation set을 어떻게 두어야할지는 아직도 의문으로 남습니다.

좀 다른 이야기지만, 이 문제를 해결하지 못했기 때문에 하루 5번 플랫폼에 직접 제출해보는거 말고는 제가 새롭게 세운 전략들을 검증할 방법이 없었습니다. 그래서 피처를 선택할 때 좀 더 논리적으로 생각하려고 노력하였고 결과적으로 지난 번 Stage1때보다 다양한 방법을 시도하지는 못했지만 좀 더 의미 있는 생각들을 많이 해볼 수 있었던 것 같습니다.

4. 싱글 모델 튜닝시에는 CatBoost를 활용하기로 한다.

여러 모델들에 대해 완벽히 이해하고 있었다면 좋았겠지만, 주어진 시간도 짧고 모델의 깊은 의미까지 이해하기에는 어려움이 있었기 때문에 일단 각 모델에서 hyperparameter의 의미와 기본적인 동작방식까지만 이해한 채로 모델 튜닝에 뛰어들었습니다.

대표적인 모델들인 XGBoost, LightGBM, CatBoost, TabNet 등을 모두 실험해보았습니다. hyperparameter가 모델의 성능에 영향을 주기 때문에 완벽히 같은 환경에서 실험했다고는 할 수 없지만 아무튼 실험의 결과 일반적으로 CatBoost의 성능이 가장 좋았습니다. 그리고 AutoML framework인 pycaret도 활용해보았었는데 역시 CatBoost가 가장 높은 성능을 보여주었기 때문에 feature를 찾을 때나 single model tuning시에는 CatBoost를 중점적으로 활용하기로 하였습니다. 아래는 AutoML을 돌려본 결과입니다.

```
ridge     0.3844  0.2832  0.3291     0.077
                                Model  Accuracy     AUC  Recall   Prec.
catboost             CatBoost Classifier    0.7938  0.7939  0.3684  0.6710
gbc         Gradient Boosting Classifier    0.7883  0.7867  0.3408  0.6579
lightgbm  Light Gradient Boosting Machine    0.7837  0.7727  0.3719  0.6223
rf              Random Forest Classifier    0.7776  0.7679  0.3676  0.6015
ada                 Ada Boost Classifier    0.7852  0.7646  0.3468  0.6418
xgboost        Extreme Gradient Boosting    0.7754  0.7632  0.3875  0.5859
et                Extra Trees Classifier    0.7649  0.7569  0.3564  0.5556
lda         Linear Discriminant Analysis    0.7835  0.7233  0.2967  0.6586
lr                  Logistic Regression    0.7822  0.7204  0.2768  0.6656
nb                          Naive Bayes    0.7551  0.7102  0.1003  0.5991
qda       Quadratic Discriminant Analysis    0.7656  0.7072  0.2491  0.5878
knn               K Neighbors Classifier    0.7516  0.7004  0.3511  0.5141
dt              Decision Tree Classifier    0.7114  0.6205  0.4524  0.4321
svm                 SVM - Linear Kernel    0.6717  0.0000  0.4176  0.4355
ridge                  Ridge Classifier    0.7841  0.0000  0.2681  0.6841
```

AutoML은 그냥 있다길래 돌려본거라 어떤 방식으로 돌아가는지 정확히 알지는 못하지만 그래도 CatBoost를 선택할 수 있는 근거/단서 정도는 될 수 있었던 것 같습니다.

물론 CatBoost만 쓸게 아니라, 나중에는 다른 모델들과도 ensemble도 해보고 싶었지만 성능이 잘 나오는 피처를 막바지에 발견하게 되었고, 제출 기회가 부족했기 때문에 결국 이전에 만들어 온 CatBoost 모델들끼리 앙상블한 결과를 최종 제출하게 되었습니다. ㅠㅠ

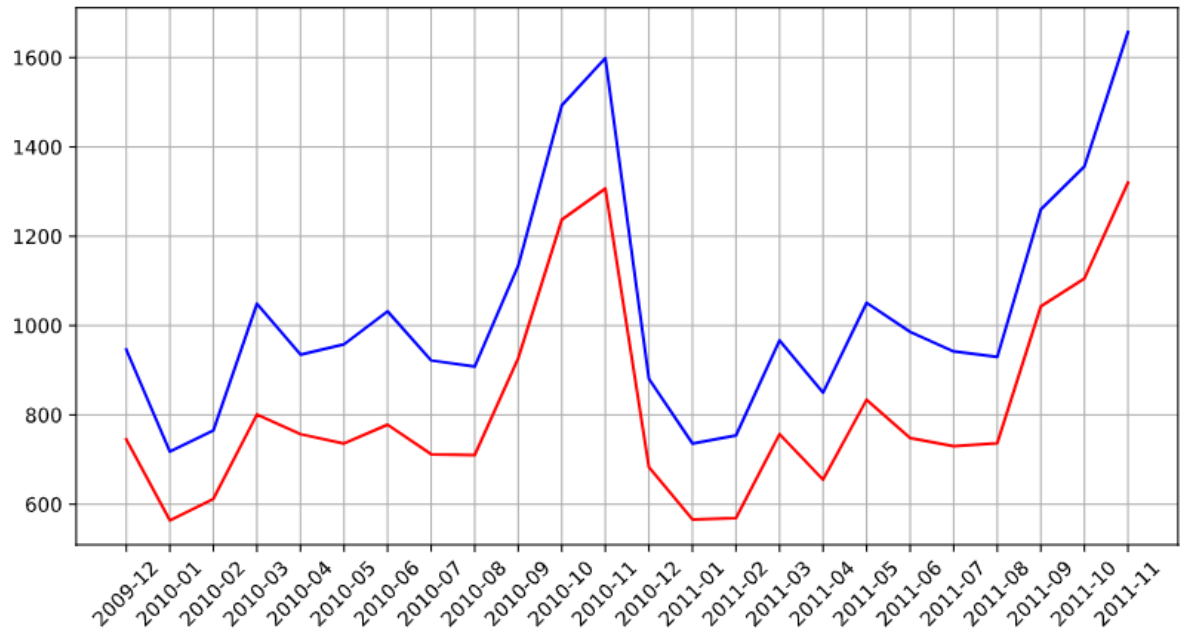
5. 구매데이터는 고객별/월별로 집계해서 사용하며 이를 중심으로 활용

70만개의 데이터가 있지만 고객은 고작 5900명 남짓 됩니다. 또한 각자의 구매 시점도 제각기입니다. 저는 이 상황에서 하루하루 day까지 고려하는 것은 너무 복잡한 작업이 될 것으로 예상했습니다. 따라서 앞서 언급했듯이 저는 월별로 집계된 데이터를 활용하기로 했습니다.

물론 단순 aggregation을 취한다면 월별로 집계하든 일별로 집계하든 상관이 없을 것입니다. 하지만 저는 baseline에서 제공된 형태처럼 전체에 대한 aggregation을 구하는 것이 넓은 관점에서 봤을 때는 학습에 많은 도움이 되는지도 의문이었고 무엇보다 제 스스로가 확실하게 이해할 수 있는 feature를 가져가는 것이 이번 대회에서 제가 추구하고자 하는 방향이었기 때문에 이후 다른 피처들을 뽑아낼 때도 모두 월별 집계 데이터에서 작업을 진행하였습니다.

6. 문제를 '구매 여부 예측'과 '구매량 예측'이라는 두 개의 문제로 분리하기

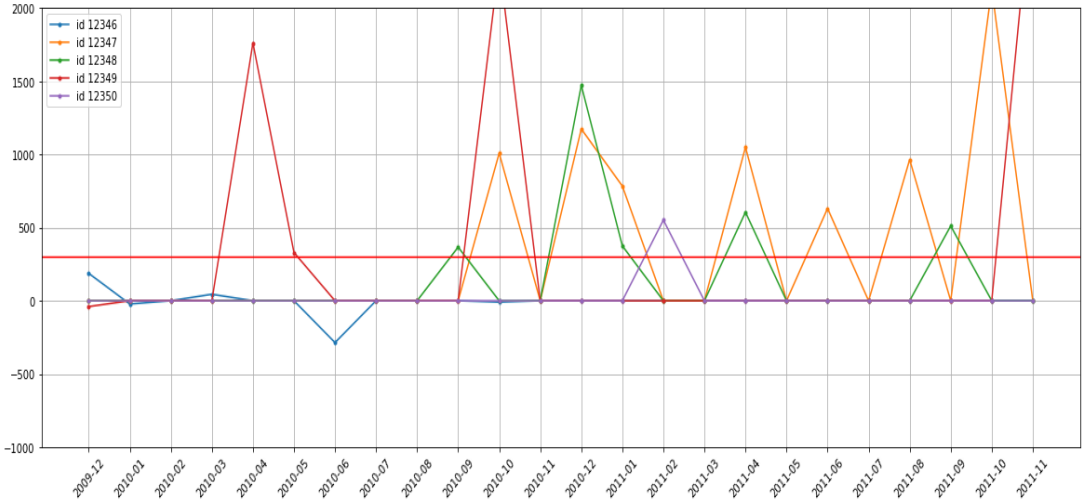
피처 선택을 제외하고는 score에 가장 큰 영향을 준 전략입니다. 고객의 구매 데이터를 전체적으로 살펴보면 대체로 $300 이상 구매하는 사람은 계속해서 $300 이상 구매하는 경향성이 있습니다. 반대로 $300 이하로 구매하는 사람은 거의 항상 $300 이하로 구매합니다. 물론 구매량이 $300 주변 어딘가에 걸쳐있는 애매한 사람도 있겠지만 사실 중요한건 구매량이 아니라 **구매가 존재하느냐**입니다. 매달 구매내역이 존재하는 사람이 몇 명이나 있는지 살펴보면 그 비율이 대략 4(구매X):1(구매O) 정도의 비율로 나옵니다. 즉 구매를 하지 않는 사람이 훨씬 많다는 것입니다. 그러면 이 분포를 보고 어떤 판단을 할 수 있을까요? 일단은 여러 관점에서 라벨의 분포를 더 보면 좋을 것 같습니다. 아래는 구매를 한 사람 중 $300 이상 구매한 사람 수를 나타냅니다.



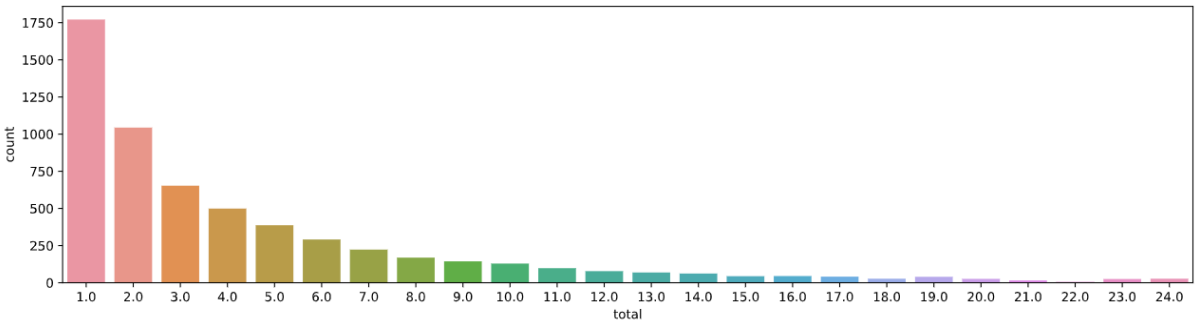
위 그래프에서 빨간 선은 $300이상 구매한 사람 수, 파란 선은 뭐라도 구매한 사람 수($0 이상)를 나타냅니다. 추세가 매우 유사하다는 점을 먼저 알 수 있고, **구매를 했다는 사실만 알 수 있다면** $300 이상 구매를 했는지 구별하는 문제는 보다 쉬울 것으로 예상됩니다.

쉬운 문제인만큼 $300 이상 구매 했는지를 파악하기 위해 별도의 모델을 두어야 하는지도 곰곰이 생각해볼 사안입니다. 모델을 둔다고 이 부분에 대한 분류가 더 잘 될까요? 저는 처음에는

모델을 두려고 하였으나 앞서 언급했듯이 **$300 이상 구매자는 구매 내역만 존재한다면 그때는 거의 항상 $300 이상 구매하는 경향**이 있습니다.



위 그래프에서 가운데 빨간선이 $300의 경계를 나타냅니다. 다소 애매할 때도 있지만 앞서 언급한 경향성에서 크게 벗어나지 않습니다. 약 500명의 고객을 직접 눈으로 확인해보았는데 거의 비슷한 경향세를 보입니다. 따라서 저는 **'구매를 했느냐?'에 집중하기로 했고,** '얼마나 구매했느냐'는 **평균값을 활용하기로 했습니다.** 평균을 취하는 것이 단순한 방법이기는 하지만 이렇게 길이가 **짧은 시계열 데이터에서는** 오히려 모델을 통한 예측보다도 강력한 방법이라고 생각했습니다.



위 그래프는 월별 집계 데이터에서 구매내역이 존재하는 달 수를 나타낸 것입니다. 즉, **구매 내역이 존재하는 달이 하나뿐인 사람이 1750명에 달한다는 것**이죠. 월별로 평균을 취할 때는 구매가 존재하지 않는 달은 제외하고 구하기 때문에 그 수가 대부분 6을 넘어가지 못합니다. 이러한 상황에서 평균을 통한 예측은 간단하면서도 좋은 방법이라고 생각했고 이를 채택했습니다.

정리하자면, 우리는 먼저 모델을 통해 구매 여부를 예측한 후, 평균 구매량이 $300 이상인 사람은 이 확률에 1을 곱해주고 $300 이하인 사람은 이 확률에 0을 곱해주어 최종 logit을 내게 됩니다. 그런데 이렇게 하면 평균 구매량이 $200~$300에 걸쳐있는 애매한 사람들까지 무조건 확률이 0이 되어버리는 문제가 발생하기 때문에 적당한 경계값을 정해 이 경계값 이상의 구매내역을 가진 사람 정도들은 구매 여부 확률에 0.5를 곱해주기로 하였습니다. 예를 들어 경계값이 $200이면

$200 미만의 평균 구매량을 가진 사람은 0.0, $200 이상 $300 미만의 평균 구매량을 가진 사람은 0.5, $300 이상의 평균 구매량을 가진 사람은 1.0을 곱해주게 됩니다.

경계값은 최종적으로 $210으로 설정하였습니다. 추가적으로 1.0을 곱해주는 조건(threshold)은 $300가 아닌 $270으로 설정하였습니다. 앞서 언급했듯이 직접 제출하는 것 말고는 실험에 대한 성능 평가를 정상적으로 진행할 환경이 안되었기 때문에 이 값은 몇 번 해봐서 잘되는 것으로 설정하였습니다. 확실한 것은 $300로 strict하게 조건을 주는 것보다는 $270로 조건을 좀 loose하게 주는 것이 성능에 긍정적 영향을 주었던 것 같습니다.

7. 유실된 데이터는 train set에서 그대로 가져온다.

저는 제공된 baseline과 비슷하게 train/valid는 2009-12~2011-10의 데이터를 활용하였으며 (2011-11의 구매 데이터 예측) test는 2010-01~2011-11 데이터를 활용하였습니다. 그러면 자연스레 2009-12에만 구매 내역이 존재하는 customer들은 test data에서 손실됩니다. 그래서 유실된 row들을 가져오는 전략 역시 필요합니다. 저는 이 데이터를 그냥 train에서 그대로 가져왔습니다.

처음에는 그냥 손실된 데이터들을 모두 label 0으로 처리했습니다. 구매내역이 쭉 없다가 2년만에 갑작스레 $300 이상 구매할 확률은 없다고 생각했기 때문입니다. 그런데 실제로는 모두 0으로 처리하는 것보다는 train에서 row들을 그대로 따서 가져오는 것이 더 성능이 좋았기 때문에 이러한 imputing 전략을 사용하였습니다. 그런데 2009-12에만 구매 데이터가 존재한다면 2011-11에 대한 label 값도 0이라 어차피 해당 데이터들은 0에 가깝게 예측이 되도록 학습될텐데 이게 더 잘 되었던 이유는 아직 잘 모르겠습니다..

피쳐 선택 전략

1. country, order_id, product_id, description은 제외하기로 한다.

겉보기에 큰 영향이 없어보이는 데이터들은 제외하기로 하였습니다. 정형 데이터를 처음 다뤄보고 있는데 이런 부분들까지 고려하기에 2주는 너무 짧은 시간이기도 했고, 토론 게시판에서 많은 분들이 분석 글들을 올려주셨으나 여전히 큰 의미를 찾을 수 없었습니다. 주어진 피처가 그리 많지도 않은데 그중 벌써 4개를 제외해야한다는 점이 너무나 슬펐지만ㅠㅠ 다른 피처들에서 잠재 의미를 찾아내는 것이 더 의미가 있다고 판단했습니다.

2. Aggregation은 sum과 skew만을 적용한다.

이것 역시 앞서 언급했듯이 고도화된 실험을 해보지는 못했으나 제가 주목한 것은 (1) 피처가 많다고 성능이 잘 나오는 것은 확실히 아니었다는 점, (2) 그렇다면 피처를 필요한 것만 남겨야하

므로 각각에 어떤 의미가 있는지 파악해야 한다는 점이었습니다. 일단 강의에서 알려주신 aggregation 기법 외에는 자세히 알아보지 못해서 그 baseline에 있는 aggregation function들을 위주로 생각해보았습니다.

일단 첫번째로 짚고 넘어갈 점은 앞서 task를 두 개로 분리하였기 때문에 사용할 feature는 **구매 여부 분류를 도울 수 있는 것**이어야 합니다. 먼저 max, min, mean은 $300 이상 구매했는지 판단할 때는 영향이 있을 수 있겠지만, 구매 여부와 직접적 연관은 없다고 생각했기 때문에 이를 제외하였습니다. 반면 **구매의 빈도를 파악하는 수단**으로써 sum이 유의미하다고 생각했습니다. 물론 사람마다 sum이 크게 차이날 수 있으나 sum이 일정 수준 이상이면 구매 빈도가 어느정도 있다고 판단할 수 있다고 생각했기 때문입니다. (다만 추후 sum 대신 cumsum을 적용하게 되었습니다. cumsum의 마지막 column은 어차피 sum의 값과 일치하기 때문에 sum이 어차피 포함됩니다.) 그리고 skew도 적용하였는데, skew는 time series data의 추세선 역할을 할 수 있다고 생각했기 때문입니다. 실제로 baseline에서 feature importance를 출력해보면 skew는 거의 모든 모델에서 중요도가 높게 출력되었기 때문에 skew 채택은 큰 고민 없이 할 수 있었던 것 같습니다.

3. price feature도 없는 것이 낫다.

price라는 feature도 경험적으로 제외하게 되었습니다. 일단 가장 큰 이유는 없앤 쪽의 성능이 더 나았기 때문입니다. 현재 주어진 수치형 데이터가 price, quantity, 그리고 total입니다. 처음부터 total은 반드시 포함되어야 한다고 생각했고 quantity와 price 중 하나는 없어도 상관이 없지 않을까 싶었는데(quantity * price = total 이라는 상관관계가 있었기 때문에 꼭 세 개가 다 필요하지는 않을 것 같다는 생각을 했습니다) 마침 성능이 이렇게 나오다보니 price feature는 빼게 되었습니다.

4. time series data에서만 뽑아낼 수 있는 피처를 활용

오피스 아워를 듣고 정형 데이터 문제에서 cumsum(cumulative sum)을 통해 생성한 feature를 많이 활용한다는 것을 알게 되었습니다. 이에 따라 월별 구매 데이터(23개 feature)에 cumsum이 적용된 23개 feature(column)를 더 추가하게 되었습니다. 더군다나 지금은 time series data라서 cumsum을 활용하는 것에 대한 의미가 더욱 크다고 생각했습니다.

한편, time series data에는 주기성이나 계절성(지속성)과 같은 좋은 특징이 존재합니다. 강의에서도 밝혀졌듯이 예를 들어 이전 해 12월 데이터는 유용한 지표로 활용될 수 있습니다. 이러한 점을 최대한 활용하기 위해 원래 월별 데이터와 cumsum이 적용된 월별 데이터 각각에 대하여 다음과 같은 피처를 추가로 추출하였습니다.

(1) 최근 10개월/7개월/4개월 구매 데이터에 대한 sum/skew – 최근에 구매를 많이 했다면 계속해

서 구매를 지속할 가능성이 큽니다.

(2) 최근 1년간 2개월/3개월 간격 구매 데이터에 대한 sum/skew, 1년 간격 구매 데이터에 대한 sum/skew – 데이터를 보면 일정 간격으로 구매를 하는 사람들이 존재합니다.

(3) 이전 해 비슷한 시기(ex. 2010년 11월~2011년 1월) 데이터들의 sum/skew – 딱 12월 데이터만이 아니라, 비슷한 시기의 데이터들도 도움이 될 수 있습니다.

이걸 처음에는 sum과 cumsum이 아니라 count와 이에 대한 cumsum에다가 적용했었습니다. 그때는 어차피 지금 보고있는 것은 $300 이상 이하가 중요한게 아니고 구매 여부가 중요한 것이니 그냥 구매를 했으면 1, 안했으면 0으로 나타내고 이에 대한 피처 엔지니어링을 하는 것이 모델 학습에 더 도움이 되지 않을까라는 생각을 했습니다. 하지만 이렇게 하니 오히려 모든 사람의 데이터가 거의 비슷비슷하게 변하게 되었고 model이 일부 데이터에 overfitting되는 현상이 일어나 성능이 급격히 감소하였습니다.

결국 당시 이 피처를 폐기하게 되었는데, 지표 그 자체(total, quantity)를 그대로 활용하니 큰 성능향상을 볼 수 있었습니다. 정수형 데이터로 바꿔 놓으면 모델이 고려할 사항이 줄어들어 더 좋은 학습을 할 수 있을 거라고 생각했는데 이렇게 하면 오히려 그 경우의 수가 너무 줄어들어 학습에 방해가 될 수도 있다는 것을 깨달을 수 있었습니다. 당시 count를 활용했으니 경우의수가 24개밖에 되지 않아(0-23) skew를 활용한다고 해도 고객 하나하나마다의 특성을 파악하기에는 부족했을 것이라고 생각됩니다.

한 가지 아쉬운 점은 이러한 피처는 추가하려면 계속해서 추가할 수 있을 것 같은데 기한이 얼마 남지 않았을 때 이 피처가 유의미하다는 점을 재발견해서 피처를 더 추가하지 못했다는 것입니다. 더 다양한 관점에서의 계절성과 주기성을 반영할 수 있었다면 성능이 조금은 더 향상되지 않았을까 합니다.

그 외 전략

1. $20 이상부터 구매 내역으로 간주

status_thres라는 변수를 설정하였습니다. 상술했던 모든 전처리에 있어 **status_thres를 넘는 데이터만을 환불이 아닌 유의미한 데이터라고 판단**했으며, status_thres를 넘지 못하면 모두 0으로 간주하였습니다. status_thres라는 것을 두게 된 유일한 근거는 실험적으로 성능이 괜찮았다는 점입니다. status_thres 값은 20으로 두었습니다. 0으로 두는게 가장 일반적일 수 있겠지만 $20 이하의 구매는 구매를 안 한 것으로 간주하는 게 더 좋을 것 같다고 생각했습니다.

2. hyperparameter 튜닝

앞서 valid set의 신뢰도가 낮아 하이퍼파라미터 튜닝도 어렵다는 말을 했었지만 그럼에도 일단은 하는게 낫지 않을까 싶어 GridSearchCV로 몇 개 돌려보았고 가장 최적으로 나온 hyperparameter들을 활용하기로 하였습니다. 앞선 이유로 여기에는 많은 시간을 쏟지 않았고 그냥 많이들 하길래 연습삼아..? 해봤습니다. 이게 성능에 어떤 영향을 줬을 지는 잘 모르겠습니다.

3. 최종 submission에서 0.1 이하의 probability는 0으로 간주

최종 submission에서 0.1 이하의 확률으로 나왔다면, 제 모델의 특성상 대부분 **구매조차 하지 않았을 확률이 높다는 점**을 생각했습니다. 그렇다면 0.1 이하의 확률 값을 가진 사람들은 그냥 0.0으로 간주하는 것이 더 좋다고 생각했고 실제로 소폭의 성능 향상이 있었습니다.

그렇다고 확률이 매우 높은, 예를 들어 확률 0.8 이상의 사람들을 1.0으로 간주하는 것은 오히려 성능 하락을 가져왔습니다. 사실 제출 기회가 좀 많이 남았을 때 이러한 실험을 몇 번 해봤었는데 0.1이 아래쪽 경계의 최적 값이었고 위쪽 경계는 그냥 안 두는게 더 나았던 것 같습니다. 물론 이쪽 실험도 그렇게 많이 한 건 아니라서.. 최적 값이 있을지도 모르긴 하겠습니다.

그럼 일반적으로 위쪽 경계를 두는게 안 좋았던 이유는 뭘까요? 생각해보면 앞서 언급했듯이 구매하는 사람과 구매 안하는 사람의 비율이 1:4로 크게 차이나기 때문인 것 같습니다. 즉 그냥 아무나 붙잡고 넌 구매 안할거야 라고 했을 때 맞을 확률이 80%이기 때문이라는 것이죠. 반면 구매를 할 사람은 구매를 하지 않을 사람에 비해 훨씬 적기 때문에 이 사람들이 확실히 구매를 한다 라고 못 박았을 때의 risk는 훨씬 더 클 것입니다. 그래서 확률이 0에 가까운 쪽은 신뢰도가 높은 반면 1에 가까운 쪽의 신뢰도는 비교적 낮을 수밖에 없었다고 생각합니다(이걸 신뢰도라고 표현하는게 맞는진 모르겠지만). 요약하자면 결국 분포의 차이로 인해 이러한 현상이 발생했다고 생각합니다.

최종 제출 요약

Model	CatBoost
Hyperparameter	depth: 3 iterations: 500 l2_leaf_reg: 3.16e-20 leaf_estimation_iterations: 10 learning_rate: 0.1
# of Feature	125 features
추가 활용 전략	문제를 2개로 분리(구매 여부, 구매 금액) 주기성/계절성/지속성 반영한 feature 생성
Ensemble	(1) [피처 선택 전략 - 4] 반영 X + GridSearchCV (AUC 0.8597) (2) (1)과 동일 + [문제 해결 전략 – 7] 반영 O (AUC 0.8603) (3) (1) + (2) Ensemble, weight 0.5/0.5 (AUC 0.8614) (4) 레포트에 기술한 전체 내용을 반영한 모델 (AUC 0.8613) (5) **(최종 제출)** (3) + (4) Ensemble, weight 0.5/0.5 (AUC 0.8640)

마주한 한계와 도전 숙제

- 제출 기회를 너무 막 써버린 감이 있는데, 다음부터는 제출 기회도 계획적으로 써야할 것 같다는 생각이 듭니다. 이번에는 2주동안 총 55번의 제출 기회가 있었는데 제가 총 54번의 제출 기회를 썼습니다. 하지만 그 중 의미 있는 시도가 반은 되는지.. 이건 생각해볼 문제인 것 같습니다.
- 방법에 열중하지 말고 근본적인 것에 열중해야한다는 생각이 듭니다. 어차피 지금은 연습 과정인데 그렇다면 속 알맹이를 단단하게 만들며 넘어가는게 중요할 것입니다. 현재는 그러지 못하고 있기 때문에 이건 반성할 부분인 것 같습니다.