**Summary**

2021-05-30

1. 데이터 전처리

우선 DAYS\_BIRTH 변수가 일 기준으로 되어 있어 나이를 의미하는 변수로 변환시켰다. 마찬가지로 DAYS\_EMPLOYED는 연차 개념으로 변환하였다. 일 단위를 사용하는 수치에 연도의 개념을 추가해줌으로써 분류의 정확도를 조금 더 높여줄 것이라 예상하였다.

DAYS\_EMPLOYED 변수에서 음수는 고용된 기간, 양수는 고용되지 않았음을 의미함에도 불구하고 극도로 높은 숫자가 default로 잡혀있었다(약 3만). 따라서 0으로 조정해주었다. 이후에 결측치 대체 작업과 5번 과정을 위해서 수입을 사분위 수로 구간화하여 income\_c 변수를 만들었다. 이를 이용하여 수입 1분위 여자, 수입 1분위 남자, 수입 2분위 여자… 등 더 다양하게 두 변수간의 교호작용을 파악하고 분류의 정확도를 높여보려 했다. (실제로 증가는 있었으나 눈에 띄는 정도는 아니었다.)

이후 occyp\_type 변수는 대부분(60%)이 결측치인 것을 확인하였다. Occyp\_type 변수를 삭제한다고 하면 그만큼 정보가 손실되는 것이기에 최대한 기존에 있는 정보들을 이용하여 예측해보기로 하였다. 이에 직업에 영향을 줄 것이라 예측되는 변수들을 추려내었다. 이 경우는 소득 분류(income\_type), 성별(gender), 그리고 수입(income\_c)이다. 기존에 결측되지 않은 값들을 바탕으로 피벗 테이블을 생성하여 각 구간별로 최빈도의 값을 대체할 값으로 정하였다. (실제로 유의미한 성능 향상을 이끌어냈다!)

가족 수(family\_size)와 자녀의 수(child\_num)이 있어 어른의 수 역시 어떻게 파악해볼 수 있지 않을까라는 생각에 adult\_num 변수를 만들어 보았다. 단, 자녀가 꼭 동거 중이라는(독립했을 수도 있다) 것이 보장되지 않음을 확인하여 가입자가 특정 나이에 도달할 시 자녀가 독립할 것이라는 가정하에 greed search를 시행하였다. 그 결과 54세 정도에 독립한다고 가정했을 때 가장 높은 성능을 보였다.

가족 타입(family\_type)의 경우 싱글/미혼으로 분류됐음에도 불구하고 아이가 존재하는 경우가 있었다. 이런 경우는 간단하게 ‘이혼’한 경우로 가정하여 새로운 범주를 만들었다. 또한 아이의 수가 5명 초과로 비정상적으로 많은 경우(핵가족 시대임을 고려하여) 신용카드 가입자가 어떤 개인이 아닌 보호기관 단위로 가입한 것이 아닌가 생각되어 이 역시 새로운 범주를 만들었다.

DAYS\_BIRTH, DAYS\_EMPLOYED, before\_EMPLOYED의 경우 날짜 정보를 포함하고 있기 때문에 월, 주, 요일 정보를 최대한 뽑아내어 사용하였다.

Income\_total은 연봉을 기준으로 책정되었기 때문에 사실 지금까지 얼마나 부를 축적했는가에 대한 지표는 되지 못한다. 따라서 우선 monthly\_pay변수로 월급을 표현하고, 거기에 고용 기간을 월 단위로 변환하여 곱하여 지금까지 번 총 금액을 의미하는 변수(asset)를 만들었다.

이렇게 만들어진 변수들 중 범주형 변수들은 모두 원핫 인코딩하였다.

2. 모델링

사실 이번 대회의 관건 중 하나는 희소성을 어떻게 해결하는가 였다. 카테고리 변수가 대부분이기 때문에 이를 원핫인코딩하게 되면(라벨 인코딩은 일반적으로 성능이 잘 나오지 않았다.) 희소성의 저주에 의해 분류 성능이 저하될 것이 분명하기 때문이다. 단, edu\_type은 예외적으로 학력의 높고 낮음이 존재하기 때문에 그 정보를 최대한 보존하기 위해서 따로 매핑을 해주었다.

따라서 여기서는 이를 해결하기 위해서 오토인코더를 이용한 임베딩을 사용하였다. 카테고리 변수를 따로 떼어내어 오코인코더를 학습하고, Encoder 단의 출력을 학습용으로 사용하는 전략이다. 참고적으로 VAE 인코더(변분적 인코더)와 PCA, 그리고 심플한 FC 인코더를 모두 사용해보았는데 VAE인코더를 사용하여 35차원으로 축소하는 것이 가장 안정적이고 좋은 성능을 보였다.

모델 선정에 있어서는 정형 데이터이기 때문에 선택지가 그렇게 많지는 않았다. 우선 가장 좋은 선택은 의사결정나무 기반 알고리즘을 사용하는 것이었다. 그 장점은 1) EDA 과정에서 일반적이지 않은 데이터들이 종종 보였는데, 이상치에 민감하지 않은 일반화된 결론을 도출해준다. 2) LGBM 라이브러리를 사용하면 그래디언트 부스팅 기법을 빠르게 적용할 수 있다. 3) 변수 생성 과정에서 쓸모없는 정보가 있었다 한들 변수 선택의 효과가 있다. 따라서 LGBM을 사용하기로 결정하였다.

참고적으로 XGBoost, Random Forest, TabNet, 2-NN LGBM 스태킹 앙상블 등 다양한 기법을 사용해보았다. 그 중엔 CatBoost도 있었는데, 나중에 확인해보니 상위권은 모두 CatBoost를 사용하고 있었다. 튜닝과정에서 문제가 있기도 했고, 너무 처음에 정해놓은 틀(LGBM을 사용하는 것)에 얽매여 진행한 것이 아닌가 반성해본다.

여하튼 Light Gradient Boosting Classifier를 100-fold Stratified K-fold한 데이터에 대해서 각각 학습시켜 결과를 평균내었다. K-fold 또한 Stratified 기법이 가장 성능이 높음을 경험적으로 확인했다.

이런 방식으로 약 10위 정도의 성능을 낼 수 있었는데, 여기서 성능을 더 뽑아내기 위해서 같은 방식으로 XGB 50-fold, RF 10-fold하여 기존 결과에 평균을 내어 사용했다. 이 결과 약 8~9위 정도 성능을 낼 수 있었다(일종의 Soft Voting). 이후엔 Public Score를 더 낮추기 위해서 Soft Voting 가중치를 변경해가며 제출하였다.

3. 평가

사실 개인적으로는 아쉬움이 많이 남을 수밖에 없는 것이 Private Score의 존재를 몰랐다는 점이다. Public Score를 최적화하는 것에만 눈이 멀어 가중치를 조정해가면서 반복해서 제출했고, 대회가 끝난 뒤에는 입상이라고 좋아하고 있었다. 일반화가 아닌 Public Score에만 초점을 맞춘 파라미터 튜닝은 결과적으로 테스트 셋에 대한 과적합이라는 결과를 안겨주었다.

또한 CatBoost에 대해서도 안타까움이 남는다. 우선 오토인코더를 매번 돌리며 적절한 스텝 사이즈와 차원을 찾는 것이 정말 번거로운 일이었는데(심지어 변수 하나가 추가될 때마다 다시 하였다.) CatBoost같은 경우는 카테고리 변수의 인덱스만 입력하면 자동으로 처리해주었다.

제일 많이 시간을 투자한 파라미터 튜닝의 경우에는 Optuna라는 라이브러리를 대회 종료 2일 전에 발견하면서 결국 고통만 남은 대회로 끝나고 말았다….

그럼에도 불구하고 이번 대회를 진행하면서 많은 것을 배웠고 앞으로는 더 잘할 자신이 분명하게 생겼다. 그리고 정말 큰 교훈도 얻었는데 그것은.. “과적합은 안 좋다”.