# [5회차] ML 과제 Report

**28기 남궁현종**

1. 데이터 로드 및 기본 탐색

데이터로 사용한 creditcard.csv 파일은 신용카드 거래 기록이 사기인지 여부를 탐지하기 위한 데이터이다. 데이터를 df 변수로 불러온 뒤, 분석을 진행하였다(원본 데이터 파일은 용량으로 인해 깃허브 업로드가 불가능하여 코드 파일만 업로드하였다. 코드 재현을 위해서는 YBIGTA_newbie_assignment 폴더 안에 csv 파일을 넣고 진행하면 된다.)

우선 df.head()로 변수의 개수와 첫 다섯 행을 출력해보았다. 총 31열이 출력되었으며, 독립변수로 Time, V1 ~ V28, Amount가, 종속변수로 Class가 있음을 알게 되었다. 또한 첫 다섯 행을 통해 데이터의 범위가 Time은 정수, V1~ V28은 대략 -1 ~ 1 부근의 실수 값, Amount는 다소 스케일이 큰 실수 값, Class는 0만 확인이 가능했지만 사전 지식을 통해 0또는 1의 범주형 변수일 것으로 예측할 수 있었고, 이후 분류 모델 사용 시 독립변수들에 대해 스케일링을 진행해야 스케일에 민감한 모델의 분류 성능을 저하시키지 않을 수 있을 것이라 생각하였다.

이후 df.info()를 통해 데이터의 개수, 결측치 개수, 데이터 형식을 확인해보았다. 데이터 개수는 총 284807개로, 결측치는 관측되지 않았다. 또한 독립변수는 모두 float 형식, 종속변수는 int 형식으로 확인되었다. 또한 df.describe()를 통해 데이터의 기초통계값을 출력해보았다. 그 결과 Time의 평균값이 가장 크게 출력되어, 변수 간 스케일 차이가 크다는 것을 다시 한번 확인할 수 있었다. 종속 변수의 경우, 최솟값이 0, 최댓값이 1인 int 데이터인 것으로 보아, class가 0, 1로 인코딩 된 범주형 변수임을 확인할 수 있었다.

분류 문제이기 때문에 종속변수의 class imbalance 여부를 확인하는 것이 매우 중요하다. 앞서 종속 변수가 0, 1로 이루어진 범주형 변수임을 확인했으니, Class 변수에서 0과 1 데이터 수의 비율을 출력해보았다. 그 결과 사기 거래로 분류된 1의 비율이 정상 거래로 분류된 0에 비해 현저히 낮다는 것을 확인하였고, class imbalance 해결이 모델 성능에 큰 영향을 미칠 것이라 판단하였다.

2. 샘플링

Class imbalance 문제를 해결하기 위해서는 under sampling, over sampling 등 다양한 방법들이 있지만, over sampling을 적용하기엔 소수 class의 수가 너무 낮아서 over sampling 적용 시 실제 데이터가 희석되어 사실상 가짜 데이터만으로 분류를 진행하게 되는 문제가 생긴다. 따라서 정상 거래 중 우선 10,000건만 무작위 샘플링하여 정상 거래와 사기 거래의 class 비율 차이를 개선하고자 하였다. 그 결과, 샘플링을 통해 사기

거래의 비율이 0.173%에서 4.689%로 대폭 상승하였다.

3. 데이터 전처리

데이터간 스케일 차이를 해소하기 위해 Amount 변수에 대해 z-score scaling을 진행하였고, 기존의 Amount 변수를 스케일링한 변수인 Amout_Scaled로 대체하였다. 이후 스케일링된 데이터를 모델 학습을 위해 독립변수는 X로, 종속변수는 y로 분리하였다.

4. 학습 데이터와 테스트 데이터 분할

모델 학습을 위해 X와 y 데이터를 학습 데이터인 X_train, y_train, 테스트 데이터인 X_test, y_test로 분할하였다. 이때 train:test 비율은 일반적으로 많이 사용하는 8:2로 분할하였고, 아직 class imbalance가 존재하므로 데이터 분할 시 stratify=y 옵션을 통해 종속변수의 class 비율을 유지하면서 분할하였다.

5. SMOTE 적용

앞서 말했듯, class imbalance 해결을 위해서는 under sampling 또는 over sampling을 시도해볼 수 있다. 그러나 이 데이터 셋의 경우, 소수 클래스인 사기 거래의 수가 394건(학습 데이터 기준)으로 만약 under sampling을 통해 정상 거래의 수를 줄여 class 비율을 1:1로 만들게 되면 학습을 위한 데이터 수가 너무 적어지게 되어, 모델이 학습 데이터에 과적합되는 등의 분류 성능 저하 위험이 존재한다. 따라서 over sampling 기법인 SMOTE를 통해 사기 거래 데이터와 유사한 가짜 데이터들을 만들어, 정상 거래와의 비율이 1:1이 되도록 맞춰주었다. 추가적으로 만약 학습 데이터와 테스트 데이터에 대해 함께 over sampling을 진행한다면, data leakage의 위험이 존재하므로 학습 데이터에 대해서만 SMOTE를 적용하였다.

6. 모델 학습

모델의 경우, 앙상블 모델들 중 분류 모델로 대표적인 Bagging 기법의 RandomForestClassifier, Boosting 기법의 XGBClassifier, LGBMClassifier로 우선 학습을 진행한 뒤, 테스트 데이터에 대해 가장 성능 지표가 좋은 모델을 선정하고자 하였다. 하이퍼파라미터의 경우 보편적인 값으로 임의 지정 후 학습을 진행하였고, Recall, F1-Score, PR-AUC 점수를 확인하였다. 그 결과, Recall의 경우 세 모델 모두 동일한 점수가 나타났고, F1-Score의 경우 RandomForest가, PR-AUC의 경우 LightGBM이 가장 높은 점수를 받았다. Recall과 F1-Score의 경우, Threshold에 의존적인 지표이기 때문에, 모델의 전반적인 성능을 평가하기엔 한계가 있다. 그러나 PR-AUC의 경우 모든 Threshold에서의 Precision과 Recall의 tradeoff 관계를 종합적으로 나타낸다. 따라서 정상 거래보다 사기 거래에 대한 탐지 능력을 키우는 것이 중요한 분류 문제에는 PR-AUC가 성능 지표로 가장 적합하다고 생각하여 LightGBM을 최종 모델로 선정하였다.

7. 최종 성능 평가

최종 선정된 모델인 LightGBM에 대해서 하이퍼파라미터 튜닝과 Threshold 조정을 통해 성능을 향상시키고자 하였다. 우선 하이퍼파라미터 튜닝을 진행하였다. Cross Validation을 통해 최적의 하이퍼파라미터 값을 탐색하였고, 이때 기준 점수는 F1-Score를 사용하였다. 그 결과, {'colsample_bytree': np.float64(0.8281323365878769), 'learning_rate': 0.1, 'max_depth': 11, 'min_child_samples': 22, 'n_estimators': 300, 'num_leaves': 25, 'subsample': np.float64(0.8532241907732696)}에서 사기 거래에 대한 Precision이 0.94, Recall이 0.89, F1-Score가 0.91, PR-AUC가 0.9571로 목표했던 기준치를 달성하였다. 또한 정상 거래의 경우에도 기준치를 넘긴 것을 확인할 수 있었다.

이후 Threshold도 튜닝을 진행하였는데, F1-Score를 기준 점수로, 0.3부터 0.65까지에서 성능 지표를 확인해본 결과, 약 0.5에서 성능 점수가 가장 높았던 것을 확인할 수 있었다.