Credit Card Fraud Detection

2025150101 임예찬 | School of Computing

Email: yechan2025@yonsei.ac.kr | Date: January 25, 2026

This week at a glance

Topic	Credit Card Fraud Detection, Class Imbalance, Binary Classification
Process	Undersampling (Normal: 10k), Scaling (Amount), SMOTE (Over-sampling), Logistic Regression (`max_iter` tuning)
Result	Threshold Adjustment (0.5 → 0.75) → **Recall 0.9082**, **F1-score 0.9175**, **PR-AUC 0.9547** (All Goals Achieved)

1 데이터 전처리 및 샘플링

Problem 1.1 : 데이터 구조 확인 및 샘플링

원본 데이터의 클래스 비율을 확인하고, 정상 거래(Class 0) 10,000건과 사기 거래(Class 1) 전체를 추출하여 분석용 데이터셋을 생성한다. 또한 Amount 변수를 표준화한다.

Solution. 원본 데이터의 `Class` 비율을 확인한 결과, 정상 거래가 약 99.83%, 사기 거래가 0.17%로 극심한 불균형을 보였다. 이를 해결하기 위해 다음과 같이 전처리를 수행하였다.

1. **Undersampling**: 정상 거래 중 10,000건을 무작위 추출(`random_state=42`)하고, 사기 거래 492건과 병합하였다. 샘플링 후 사기 거래 비율은 약 4.7%로 상승하였다.
2. **Scaling**: `Amount` 변수의 분포 차이를 줄이기 위해 `StandardScaler`를 적용하여 `Amount_Scaled`로 변환하였다.

2 SMOTE 적용

학습 데이터(X_train)에 대하여 오버샘플링을 진행하였다.

Definition 2.1 : SMOTE (Synthetic Minority Over-sampling Technique)

소수 클래스(사기 거래) 데이터 사이의 벡터를 보간(Interpolation)하여 인위적으로 새로운 데이터를 합성해 내는 기법이다. 단순히 데이터를 복제하는 방식보다 과적합(Overfitting) 위험이 적다.

Solution. 현재 데이터는 정상 거래가 압도적으로 많고 사기 거래는 매우 적은 **클래스 불균형(Class Imbalance)** 문제가 존재한다. 모델이 모든 데이터를 '정상'으로만 예측해도 높은 정확도(Accuracy)를 달성할 수 있어, 소수 클래스(사기)를 학습하지 못하는 문제가 발생한다. 따라서 SMOTE를 사용하여 데이터 균형(5:5)을 맞춰줌으로써, 모델이 사기 거래의 특징(Decision Boundary)을 명확히 학습하도록 유도하였다.

Code 2.1 : SMOTE Implementation Code

```python
from imblearn.over_sampling import SMOTE
from collections import Counter

# Apply SMOTE only to the training set (prevent data leakage)
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X_train, y_train)

# Check the class distribution
print("Before SMOTE:", Counter(y_train))
# Output: {0: 7999, 1: 394}
print("After SMOTE:", Counter(y_resampled))
# Output: {0: 7999, 1: 7999}
```

3 모델 학습 및 성능 평가

3.1 모델 선정

이진 분류에 적합하고 결과 해석이 용이한 **Logistic Regression**을 선정하였다. 초기 학습 시 수렴 경고(ConvergenceWarning)가 발생하여, max_iter 파라미터를 10,000으로 상향 조정하여 학습을 완료하였다.

3.2 Threshold 조정 및 최종 성능

기본 임계값(0.5)에서는 Recall은 높았으나 Precision이 낮아 F1-score가 목표치(0.88)에 미달하였다. 이에 임계값을 0.5에서 0.95까지 조정하며 최적의 성능 지표를 탐색하였다.

Code 3.1 : Threshold Tuning Results

```python
import numpy as np

# Check thresholds from 0.5 to 0.95
thresholds = np.arange(0.5, 0.95, 0.05)

# ... (omitted code) ...

# 0.75       0.9175     0.9082     0.9271
```

Theorem 3.1 : 최종 결과

실험 결과, **Threshold를 0.75로 설정**했을 때 가장 우수한 성능을 보였다. 최종 성능 평가는 다음과 같다.

- **Class 1 (사기 거래)**:
  - **Recall**: **0.9082** (목표 $\geq$ 0.80 달성)
  - **F1-score**: **0.9175** (목표 $\geq$ 0.88 달성)
- **Class 0 (정상 거래)**:
  - **Recall**: **0.9990** (정상 거래를 정상으로 완벽히 분류)
  - **F1-score**: **0.9995**
- **전체 지표**:
  - **PR-AUC**: **0.9547** (목표 $\geq$ 0.90 달성)

4 결론

이번 프로젝트에서는 SMOTE를 통해 클래스 불균형 문제를 해소하고, Logistic Regression 모델의 임계값(Threshold)을 0.75로 조정하는 전략을 취했다. 그 결과 False Positive(오탐지)를 줄이면서도 높은 재현율을 유지할 수 있었으며, 모든 평가 지표에서 목표치를 초과 달성하는 성과를 거두었다.