와빅 ML 과제

1. 데이터 로드 및 기본 탐색/샘플링/전처리

전체 데이터 중 사기 거래(Class 1)의 비율이 0.17%로 매우 적다.

```
Class
0    0.998273
1    0.001727
Name: proportion, dtype: float64
```

데이터의 불균형을 완화하고 연산 효율성을 높이기 위해, 사기 거래의 수는 유지하고 정상 거래(Class 0)는 10,000건만 무작위 추출하였다. 샘플링 후 데이터셋을 재구성하여 분석에 사용하였다.

```
=== 샘플링 후 Class 비율 ===
Class
0    0.953107
1    0.046893
Name: proportion, dtype: float64

=== 샘플링 후 데이터 건수 ===
Class
0    10000
1      492
Name: count, dtype: int64
```

`Amount` 변수의 값 범위가 다른 변수들에 비해 매우 크므로, **StandardScaler**를 적용하여 평균 0, 분산 1로 변환한 `Amount_Scaled` 변수를 생성하였다. 또한, 이후 분석에서 같은 정보를 가진 변수가 두 개일 경우 다중공선성과 같은 문제가 발생할 수 있으므로 원본 `Amount` 변수는 제거하였다. 이후 모델 학습을 위해 Feature(X)와 Target(y)으로 분리하였다.

```
=== 1. 전처리 후 X 데이터 상위 5개 (Amount_Scaled 확인) ===
       Time        V1        V2        V3        V4        V5        V6  \
541   406.0 -2.312227  1.951992 -1.609851  3.997906 -0.522188 -1.426545
623   472.0 -3.043541 -3.157307  1.088463  2.288644  1.359805 -1.06482
```

```
3
4920  4462.0 -2.303350  1.759247 -0.359745  2.330243 -0.821628 -0.075
788
6108  6986.0 -4.397974  1.358367 -2.592844  2.679787 -1.128131 -1.70653
6
6329  7519.0  1.234235  3.019740 -4.304597  4.732795  3.624201 -1.35774
6

          V7        V8        V9  ...       V20       V21       V22  \
541  -2.537387  1.391657 -2.770089  ...  0.126911  0.517232 -0.035049
623   0.325574 -0.067794 -0.270953  ...  2.102339  0.661696  0.435477
4920  0.562320 -0.399147 -0.238253  ... -0.430022 -0.294166 -0.932391
6108 -3.496197 -0.248778 -0.247768  ... -0.171608  0.573574  0.176968
6329  1.713445 -0.496358 -1.282858  ...  0.009061 -0.379068 -0.704181

          V23       V24       V25       V26       V27       V28  \
541  -0.465211  0.320198  0.044519  0.177840  0.261145 -0.143276
623   1.375966 -0.293803  0.279798 -0.145362 -0.252773  0.035764
4920  0.172726 -0.087330 -0.156114 -0.542628  0.039566 -0.153029
6108 -0.436207 -0.053502  0.252405 -0.657488 -0.827136  0.849573
6329 -0.656805 -1.632653  1.488901  0.566797 -0.010016  0.146793

      Amount_Scaled
541       -0.394650
623        1.976514
4920       0.680800
6108      -0.130191
6329       -0.390168

[5 rows x 30 columns]

=== 2. X, y 데이터 크기(Shape) 확인 ===
X shape: (10492, 30)
y shape: (10492,)
```

2. 데이터 분할/SMOTE 적용/모델 학습

학습용(Train)과 테스트용(Test)을 8:2 비율로 분할하였다. 또한, `stratify=y` 옵션을 사용하여, 학습셋과 테스트셋 모두에서 정상/사기 거래의 비율이 원본과 동일하게 유지되도록 설정하였다. 만약 이 옵션을 설정하지 않는 경우, 현재 데이터에서 사기 거래가 매우 적기 때문에, 우연히 학습 데이터에는 사기 거래가 하나도 들어가지 않을 수 있다. 결과에서 학습 데이터와 테스트 데이터 모두에서 정상/사기 거래의 비율이 유지되고 있음을 확인할 수 있다.

```
학습 데이터(X_train) 크기: (8393, 30)
테스트 데이터(X_test) 크기: (2099, 30)

=== 학습 데이터(y_train) Class 비율 ===
Class
0    0.953056
1    0.046944
Name: proportion, dtype: float64

=== 테스트 데이터(y_test) Class 비율 ===
Class
0    0.953311
1    0.046689
Name: proportion, dtype: float64
```

단순히 소수 클래스를 복제하는 오버샘플링은 모델이 동일한 데이터를 반복 학습하게 하여 과적합을 유발할 수 있다. 현재 데이터는 정상 거래가 압도적으로 많기 때문에, 모델이 정상이라고 찍기만 해도 높은 정확도가 나타난다. 이런 경우, 사기 거래를 전혀 학습하지 못하게 되므로 SMOTE 기법을 사용하여 이를 보완하고자 했다. SMOTE는 단순히 데이터를 복사하는 것이 아닌, 사기 거래 데이터 사이의 특성을 분석해 가상의 새로운 데이터를 생성하는 기법이다. 이를 통해 과적합 위험을 줄이면서 결정 경계를 보다 일반화할 수 있어 SMOTE를 적용하였다. 테스트 데이터는 미래의 실제 상황을 대변해야 하므로 건드리지 않고, 오직 학습 데이터에만 SMOTE를 적용하였다. 그 결과 학습 데이터 내 사기 거래 개수가 정상 거래 개수와 동일해짐을 확인할 수 있다.

```
=== SMOTE 적용 전 학습 데이터 클래스 분포 ===
Class
0    7999
1     394
Name: count, dtype: int64

=== SMOTE 적용 후 학습 데이터 클래스 분포 ===
```

```
Class
0    7999
1    7999
Name: count, dtype: int64

기존 데이터 개수: 8393
증강된 데이터 개수: 15998
```

다양한 ML 모델들 중에서 Random Forest 모델을 사용하였다. 수많은 결정 트리를 만들어 투표로 결과를 정하는 모델로 과적합에 강해 기본 성능이 뛰어나 선택하였다. 기본 임계값인 0.5를 적용하여 돌린 결과는 아래와 같다.

- **Precision (정밀도):** 0.95
- **Recall (재현율):** 0.89
- **F1-score:** 0.92
- **PR-AUC:** 0.9537

과제에서 제시한 기준을 충족하고 있음을 확인해볼 수 있다.

```
모델 학습 중... (잠시만 기다려주세요)
학습 완료!

=== Classification Report (Precision, Recall, F1-score) ===
              precision    recall  f1-score   support

           0       0.99      1.00      1.00      2001
           1       0.95      0.89      0.92        98

    accuracy                           0.99      2099
   macro avg       0.97      0.94      0.96      2099
weighted avg       0.99      0.99      0.99      2099

=== PR-AUC (Average Precision Score) ===
PR-AUC: 0.9537
```

Class 0 (정상 거래)

- precision (0.99): 모델이 정상으로 판단한 것 중 99%가 진짜 정답이었다.
- recall (1.00): 실제 정상 거래들 중에서 모델이 "정상"이라고 정확히 맞춘 비율이 100%였다. 즉, 정상인 사람을 사기꾼으로 오해한 경우가 거의 없다.

- support (2001): 테스트 데이터(시험 문제) 속에 정상 거래가 총 2,001건 존재했음.

Class 1 (사기 거래)

- precision (0.95): 모델이 사기라고 판단한 것 중 95%가 실제 사기였다.
- recall (0.89): 실제 일어난 사기 사건 100건 중에서 모델이 89건을 잡아냈고, 11건은 놓쳤다.
- f1-score (0.92): 정밀도와 재현율의 조화 평균. 0.92면 우수하다고 판단할 수 있음.
- support (98): 테스트 데이터(시험 문제) 속에 사기 거래가 총 98건 존재했음

---

accuracy (0.99)

- 의미: 전체 문제(2,099개) 중에서 정답을 맞힌 비율
- 주의: 정상 거래가 워낙 많기 때문에, 사기를 하나도 못 잡고 무조건 "정상"이라고만 찍어도 95점은 나옴

macro avg (0.97 / 0.94 / 0.96)

- 의미: 정상(0)과 사기(1)의 점수의 단순 평균
- 해석: 데이터 개수와 상관없이 "두 클래스 다 공평하게 잘 맞췄는지" 보는 지표. 점수가 높으므로, 소수 클래스(사기)도 무시하지 않고 잘 공부했음을 알 수 있습니다.

weighted avg (0.99 / 0.99 / 0.99)

- 의미: 데이터 개수(Support)를 반영한 가중 평균.
- 해석: 정상 거래(2001건)가 압도적으로 많아서 정상 거래 점수와 거의 비슷하게 나옴. 여기서는 큰 의미가 없다.

---

3. PR-AUC (Average Precision Score)

- 값: 0.9537
- 의미: Precision-Recall 곡선 아래의 면적. (1.0 만점)
- 해석: 단순히 "맞다/틀리다"가 아니라, 모델이 내놓은 "사기일 확률(%)"의 순위가 얼마나 정확한지를 평가

3. 성능 개선 방안

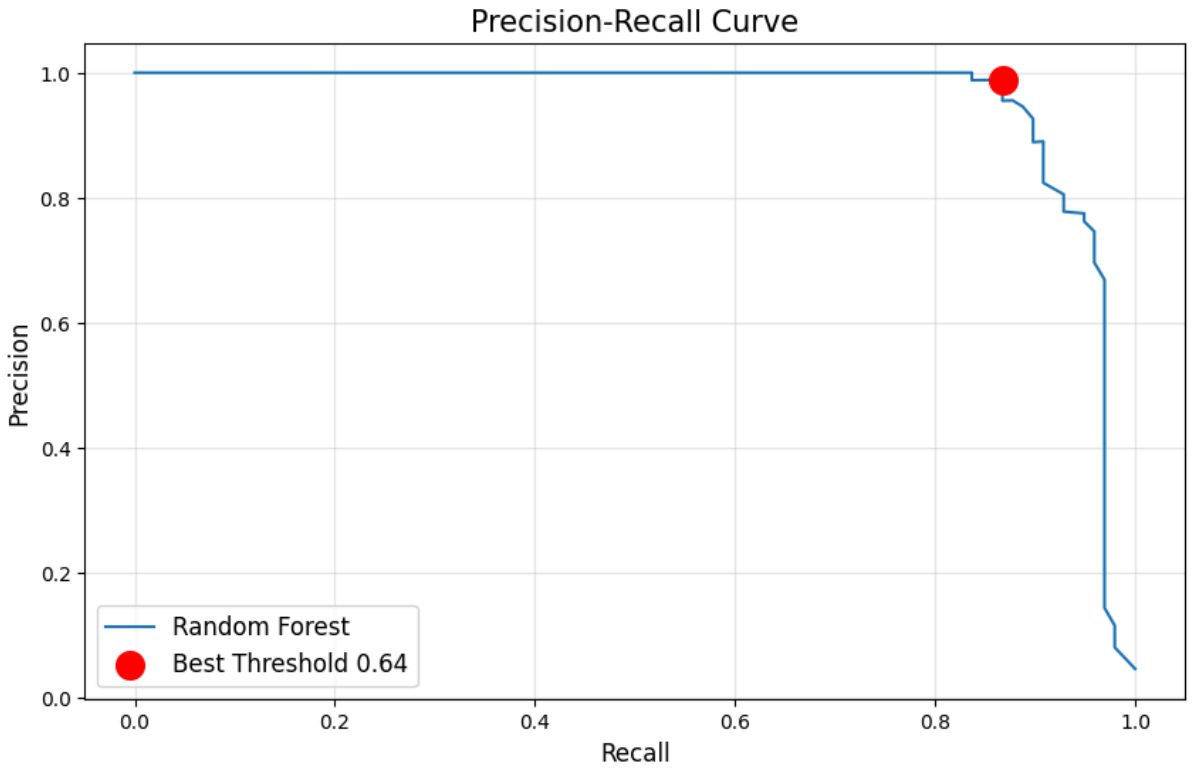
모델은 과제에서 제시한 기준을 충족하고 있지만, 사기 탐지 시스템의 특성상, 실제 사기를 놓치는 것이 가장 치명적이므로, 모델의 예측 확률(`predict_proba`)을 활용하여 성능을 극대화 할 수 있는 최적의 임계값(Threshold)을 탐색하였다. Precision-Recall Curve 상에서 F1-score가 최대가 되는 지점을 찾고자 했다. 모델이 뱉어낸 확률값(`y_pred_proba`)을 기준으로, 가능한 모든 임계값을 찾았다. 이후 구한 모든 임계점들에 대해 F1-score를 계산하여 최고의 순간을 찾았다. 그 위치에 해당하는 임계값을 기준으로 기존의 모델에 재적용하였고 그 결과는 다음과 같다.

```
=== 최적의 Threshold 탐색 결과 ===
Best Threshold (임계값): 0.6400
Best F1-Score: 0.9239
Recall (재현율): 0.8673
Precision (정밀도): 0.9884

=== 최종 Classification Report (Optimized Threshold) ===
              precision    recall  f1-score   support

           0       0.99      1.00      1.00      2001
           1       0.99      0.87      0.92        98

    accuracy                           0.99      2099
   macro avg       0.99      0.93      0.96      2099
weighted avg       0.99      0.99      0.99      2099
```



이번 과제에서는는 불균형한 거래 데이터를 SMOTE로 증강하고 Random Forest 모델로 학습하여 사기 탐지 모델을 구축하였다. 단순 예측(0/1)에 그치지 않고 예측 확률 기반의 Threshold 최적화를 통해 모델의 잠재 성능을 최대한 이끌어내고자 하였다. 최종적으로 Recall, F1-score, PR-AUC 모든 지표에서 목표치를 달성하였다.