ML 과제 report

1. 데이터 탐색 및 전처리

1.1 데이터 구조 및 불균형 확인

```
[8 rows x 31 columns]
Class
0    284315
1       492
Name: count, dtype: int64
Class
0    0.998273
1    0.001727
Name: proportion, dtype: float64
```

원본 데이터의 구조를 확인한 결과, 총 284,807개의 거래 데이터([V1~V28 특성, amount, class 특성]가 존재하였으며 그 분포는 다음과 같습니다.

정상 거래(0): 284,315건 (99.83%)

사기 거래(1): 492건 (0.17%)

1.2 샘플링 및 스케일링

과제에서 수행하고자 하는 목적에 맞게 언더샘플링(Under-sampling)을 진행, 정상 거래(Class 0) 중 10,000건을 무작위 추출하고, 사기 거래(Class 1)는 모두 유지하여 데이터를 합쳤습니다.

스케일링(Scaling): Amount 변수의 분포 편차를 줄이기 위해 StandardScaler를 적용하여 Amount_Scaled 변수를 생성했습니다.

2. 학습 데이터 및 테스트 데이터 분할

2.1 데이터 분할

```
...
    ### Train/Test Data Distribution ###
    Train Class counts:
    Class
    0    7999
    1     394
    Name: count, dtype: int64
    Test Class counts:
    Class
    0    2001
    1      98
    Name: count, dtype: int64
```

위에서 추출한 데이터들을 8:2(학습:테스트)로 나눈 후 데이터의 수는 다음과 같다.

3. SMOTE 적용

3.1 SMOTE 설명 및 적용 이유

Train dataset에만 **SMOTE(Synthetic Minority Over-sampling Technique)**를 적용하여 사기 거래 데이터를 오버샘플링하였습니다. 단순 복제(Random Oversampling)가 아닌 SMOTE를 선택한 이유는 다음과 같습니다.

단순 복제 방식은 소수 클래스 데이터를 똑같이 늘리기 때문에 모델이 해당 데이터를 과하게 암기하는 과적합(Overfitting) 위험이 있습니다. 반면, SMOTE는 데이터 사이의 특성을 분석하여 새로운 데이터를 생성하므로, 모델의 결정 경계를 더 일반화하여 새로운 유형의 사기 패턴에도 강하게 대응할 수 있기 때문입니다.

3.2 적용 결과

```
### Before SMOTE ###
Class
0    7999
1     394
Name: count, dtype: int64

### After SMOTE ###
Class
0    7999
1    7999
Name: count, dtype: int64
```

적용 전: 정상 7,999건 vs 사기 394건

적용 후: 정상 7,999건 vs 사기 7,999건

학습 데이터의 클래스 비율을 1:1로 맞춰 모델이 사기 거래 패턴을 충분히 학습할 수 있도록 조정했습니다.

4. 모델 학습 및 초기 성능 평가

4.1 모델 선정

모델**: Random Forest Classifier**

선정 이유: 데이터 내 이상치(Outlier)에 강건하며, 복잡한 비선형 결정 경계를 잘 학습하여 불균형 데이터에서도 우수한 성능을 보이기 때문입니다.

4.2 기본 성능 **(Threshold 0.5)**

모델 학습 후 테스트 셋(Test set)에 대한 평가 결과는 다음과 같습니다.

```
...
    ### Classification Report (Default Threshold) ###
                  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 Score: 0.9537
```

PR-AUC Score: `0.9537`

5. Threshold 조정 및 최종 성능

5.1 최적 임계값 탐색

기본 임계값(0.5)에서는 Precision은 높으나 Recall이 다소 낮을 수 있어, Recall을 극대화하면서도 F1-score가 무너지지 않는 최적의 Threshold를 탐색했습니다.

설정 목표: Recall ≥ 0.80, F1 ≥ 0.88, PR-AUC ≥ 0.90

기본 임계값(0.5)에서도 기준을 만족합니다.

```
...
    ### Classification Report (Default Threshold) ###
                  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 Score: 0.9537
```

하지만 정확도가 더 높은 Threshold를 찾고자 P-R곡선 상의 점들을 탐색하며 찾아본 결과는 다음과 같습니다.

탐색된 Best Threshold: `0.6400`

5.2 최종 성능 평가 **(Final Report)**

조정된 Threshold`0.6400`를 적용한 최종 성적표입니다.

```
### Final Classification Report (Tuned 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

Final PR-AUC: 0.9537
>>> 목표 성능을 달성하였습니다!
```

최종 결과: Recall (Class 1): `0.8673`, F1-score (Class 1): `0.9239`, PR-AUC: `0.9537`