5(1)-ML_report

1. 데이터 로드 및 기본 탐색

- (1) df.head()
  - 컬럼: Time, V1~V28, Amount, Class
  - 한 행이 하나의 카드 거래임.
  - Class 가 label임.

	Time	V1	V2	V3	V4	V5	V6	V7	V8
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533

5 rows × 31 columns

- (2) df.info()
  - 데이터 약 28만 건. null 없음으로 바로 모델링 가능함.
  - 모든 컬럼이 float 또는 int 임.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284807 entries, 0 to 284806
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   Time    284807 non-null  float64
 1   V1      284807 non-null  float64
 2   V2      284807 non-null  float64
 3   V3      284807 non-null  float64
 4   V4      284807 non-null  float64
 5   V5      284807 non-null  float64
 6   V6      284807 non-null  float64
 7   V7      284807 non-null  float64
 8   V8      284807 non-null  float64
 9   V9      284807 non-null  float64
 10  V10     284807 non-null  float64
 11  V11     284807 non-null  float64
 12  V12     284807 non-null  float64
 13  V13     284807 non-null  float64
 14  V14     284807 non-null  float64
 15  V15     284807 non-null  float64
 16  V16     284807 non-null  float64
 17  V17     284807 non-null  float64
 18  V18     284807 non-null  float64
 19  V19     284807 non-null  float64
...
 29  Amount  284807 non-null  float64
 30  Class   284807 non-null  int64  
dtypes: float64(30), int64(1)
memory usage: 67.4 MB
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

- (3) df.describe()
  - 분포를 확인함.

	Time	V1	V2	V3	V4	V5
count	284807.000000	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05
mean	94813.859575	1.175161e-15	3.384974e-16	-1.379537e-15	2.094852e-15	1.021879e-15
std	47488.145955	1.958696e+00	1.651309e+00	1.516255e+00	1.415869e+00	1.380247e+00
min	0.000000	-5.640751e+01	-7.271573e+01	-4.832559e+01	-5.683171e+00	-1.137433e+02
25%	54201.500000	-9.203734e-01	-5.985499e-01	-8.903648e-01	-8.486401e-01	-6.915971e-01
50%	84692.000000	1.810880e-02	6.548556e-02	1.798463e-01	-1.984653e-02	-5.433583e-02
75%	139320.500000	1.315642e+00	8.037239e-01	1.027196e+00	7.433413e-01	6.119264e-01
max	172792.000000	2.454930e+00	2.205773e+01	9.382558e+00	1.687534e+01	3.480167e+01

8 rows × 31 columns

- (4) df["Class"].value_counts()
  - Class 건수를 확인함.
  - 정상 거래(Class=0): **약 28만 건**
  - 사기 거래(Class=1): **약 500건**

  불균형 데이터임.

```
Class
0    284315
1       492
Name: count, dtype: int64
```

- (5) df["Class"].value_counts(normalize=True)
  - 비율을 확인함.
  - 정상 거래(0)은 약 99% 수준이고,

    사기 거래(1)는 약 0.17% 수준으로 매우 적음.

    데이터셋은 극단적인 클래스 불균형 문제를 지님.

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

2. 샘플링

- (1) 정상(0), 사기(1) 데이터를 분리함.
  - 사기(1) 건수: 492
  - 정상(0) 건수: 284315

```
492
284315
```

- (2) 사기는 전부 유지, 정상은 10,000건만 무작위 샘플링함.
  - 사기(1) 건수: 492
  - 정상(0) 건수: 284315 → 10000

- (3) 두 데이터를 합쳐서 데이터프레임 만들고 섞음.
  - 최종 샘플링 데이터 크기: (10492, 31)

- (4) 샘플링 후 Class 비율 다시 출력함.
  - 정상 건수: 약 0.99 → 약 0.95
  - 사기 건수: 약 0.0017 → 약 0.04

```
Class
0    0.953107
1    0.046893
Name: proportion, dtype: float64
```

3. 데이터 전처리

- (1) Amount만 표준화해서 Amount_Scaled로 대체함.
  - 값 범위(스케일)가 커서 모델 학습에 영향을 줄 수 있어 **표준화**함.
- (2) 원본 Amount 제거함.
- (3) X (입력 변수), y (정답 라벨) 분리함.
  - X shape: (10492, 30)
  - y shape: (10492,)

4. 학습 데이터와 테스트 데이터 분할

학습셋 : 테스트셋 비율을 8:2로 나눔.

클래스 불균형 데이터에서 train, test 분할 시 클래스 비율이 깨지지 않도록 함.

- 학습셋 크기, 테스트셋 크기

```
(8393, 30) (8393,)
(2099, 30) (2099,)
```

- 학습셋 클래스 분포 및 비율 확인

```
Class
0    7999
1     394
Name: count, dtype: int64
Class
0    0.953056
1    0.046944
Name: proportion, dtype: float64
```

- 테스트셋 클래스 분포 및 비율 확인

```
Class
0    2001
1      98
Name: count, dtype: int64
Class
0    0.953311
1    0.046689
Name: proportion, dtype: float64
```

5. SMOTE 적용

사기(1) 데이터가 너무 적어서 모델이 사기 패턴을 학습하기 어려움.

따라서 SMOTE를 사용해 소수 클래스(사기(1))를 학습 데이터에서 오버샘플링하여 클래스 불균형을 완화하고 모델의 사기 탐지 성능을 높이고자 함.

- (1) SMOTE 적용 전, y_train의 사기 거래 건수 확인함.
  - 0(정상)이 훨씬 많고
  - 1(사기)은 매우 적게 나옴.

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

- (2) X_train에 SMOTE 적용함.

- (3) SMOTE 적용 후, 사기 거래 건수 확인함.
  - y_train 클래스의 사기 건수: 394 → 7999

```
Class
0    7999
1    7999
```

  - smote 적용 후, X_train 건수: 8393 → 15998

```
(8393, 30)
(15998, 30)
```

6. 모델 학습

로지스틱 회귀 모델을 학습한 결과, 사기 거래(1)에 대해 Recall은 약 0.87로 목표치(0.80)를 달성했고, PR-AUC 는 약 0.9157 로 목표치(0.90)를 달성했음.

하지만, F1-score는 0.84 수준으로 목표치(0.88)에는 미치지 못하였음. 이는 threshold 조정이나 추가적인 모델 튜닝을 통해 개선해야 함.

- **(1) 모델 선택: Logistic Regression**

- **(2) 테스트셋 예측값(predict), 예측확률(predict_proba) 출력함.**
  - 앞 20개 확인

```
predict 결과(앞 20개): [0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0]
predict_proba 결과(앞 20개): [2.29647960e-04 4.15315247e-04 8.58278577e-02 4.51860203e-02
 1.10343582e-02 2.06319651e-02 1.00000000e+00 2.33570816e-03
 1.04835123e-03 6.47552444e-03 2.40847127e-03 1.37173837e-02
 1.33003355e-01 4.59882288e-03 2.07577633e-01 1.00000000e+00
 9.47092102e-03 2.27977442e-02 1.22782237e-14 4.25275557e-03]
```

- (3) Recall, F1-score 확인함.

```
              precision    recall  f1-score   support

           0     0.9935    0.9900    0.9917      2001
           1     0.8095    0.8673    0.8374        98

    accuracy                         0.9843      2099
   macro avg     0.9015    0.9287    0.9146      2099
weighted avg     0.9849    0.9843    0.9845      2099
```

  - 정상 거래(0)
    - Recall= 0.99 (0.80 달성)
    - F1-score= 0.9917 (0.88 달성)
    - 모든 지표가 높게 나타남.
  - 사기거래(1)
    - Recall= 0.8673 (0.80 달성)
    - F1-score = 0.8374 (0.88 미달)
    - 사기 98건 (support= 98) 중에, Recall=86.73% 를 잡음.

      즉, 98건 중 약 85~86건을 사기로 맞춤. 사기라고 예측한 것 중 80.9%가 진짜 사기임.

      현재 모델은 사기 거래 탐지에서 Recall은 충분하지만 Precision과의 균형(F1)을 더 개선해야 함.

- (4) PR-AUC 확인함.
  - 0.9157021609832336 (0.90 달성)
  - 불균형 데이터 환경에서도 모델이 사기 거래를 효과적으로 구분하고 있음을 의미함.

7. 최종 성능 평가

Logistic regression 모델을 사용하여 SMOTE 적용 후 학습을 진행하였음.

목표는 Class 0,1 모두에 대하여 Recall ≥ 0.8, F1 ≥ 0.88, PR_AUC≥ 0.9 가 되는 것임.

- Recall 목표는 Class 0, 1 에서 모두 달성했음.
- F1 목표는 Class 0 은 달성, Class 1은 달성 하지 못했음. threshold 조정 및 C 튜닝에서도 실패했음.
- PR-AUC 목표는 달성했음.

추가적인 성능 개선을 위해 RandomForest 나 XGBoost 와 같은 비선형 모델을 적용해보아야 했음. Logistic Regression은 선형 경계로만 구분하는데, 사기 패턴이 비선형/복잡한 조합일 경우 선형 모델은 Precision-Recall 균형 (F1) 개선에 한계가 생길 수 있음.

트리 기반 모델은 변수 간 비선형 상호작용을 잡아내어, 오탐을 줄이면서 (Precision을 향상시키며) 사기도 놓치지 않는 (Recall 유지) 방향으로 F1 개선 가능성이 있음.

- (1) threshold= 0.4

	precision	recall	f1-score	support
0	0.9940	0.9875	0.9907	2001
1	0.7748	0.8776	0.8230	98
accuracy			0.9824	2099
macro avg	0.8844	0.9325	0.9068	2099
weighted avg	0.9837	0.9824	0.9829	2099

  - 정상거래(0)
    - Recall=0.9875 (0.80 달성)
    - F1=0.9907 (0.88 달성)
    - 추가 해설:

      정상 거래가 사기 거래에 비해 패턴이 비교적 명확하며, 모델이 정상 거래를 안정적으로 학습했음을 의미함.
  - 사기거래(1)
    - Recall= 0.8776 (0.80 달성)
    - F1= 0.8230 (0.88 미달)
    - 추가해설:

      recall이 충분하여 사기를 놓치는 경우는 적음.

      하지만 사기로 예측한 것 중에, 정상거래를 사기로 잘못 잡는 오탐이 존재하여 Precision이 떨어지고 F1 이 목표에 못 미침.

- PR-AUC 는 threshold와 무관하므로 약 0.92로 같음. (0.9 달성)

- (2) threshold= 0.3

```
              precision    recall  f1-score   support

           0     0.9944    0.9835    0.9889      2001
           1     0.7250    0.8878    0.7982        98

    accuracy                         0.9790      2099
   macro avg     0.8597    0.9356    0.8936      2099
weighted avg     0.9819    0.9790    0.9800      2099
```

  - 정상거래(0)
    - Recall= 0.9835 (0.8 달성)
    - F1= 0.9889 (0.88 달성)
  - 사기거래(1)
    - Recall= 0.8878 (0.8 달성)
    - F1= 0.7982 (0.88 미달)
    - 추가해설:

      threshold 를 낮추면 사기라고 더 쉽게 판단하여, recall은 소폭 상승 혹은 유지되지만,

      그만큼 오탐이 늘어 precision이 하락하고 F1이 더 떨어질 수 있음.
  - PR-AUC 는 threshold와 무관하므로 약 0.92로 같음. (0.9 달성)

- (3) 하이퍼파라미터 C =0.1, C=1, C=10 의 Recall, F1 같음.

```
              precision    recall  f1-score   support

           0     0.9935    0.9900    0.9917      2001
           1     0.8095    0.8673    0.8374        98

    accuracy                         0.9843      2099
   macro avg     0.9015    0.9287    0.9146      2099
weighted avg     0.9849    0.9843    0.9845      2099

PR-AUC: 0.9151
```

- 정상거래(0)
  - Recall= 0.99 (0.8 달성)
  - F1= 0.9917 (0.88 달성)
- 사기거래(1)
  - Recall= 0.8673 (0.8 달성)
  - F1= 0.8374 (0.88 미달)
- PR-AUC 는 C가 0.1, 1, 10 으로 커짐에 따라, 0.9151, 0.9157, 0.9163 으로 커짐. (0.9 달성)
- 추가 해설:

  분류 결과는 거의 동일하지만 PR-AUC는 소폭 개선됨.

  C 조정만으로는 F1-score 개선에 한계가 있음.