

Project 1 : 제조 불량 원인 분석

LS Bigdata School 3rd

.

Team 4

오서연·이재준·장일준·정아영·최지원

문제 정의

“
...”

저희 회사는 데이터를 통해서 생산 공정에서 불량품을 판정
하는 기준을 더 명확하게 세우고, 불량을 야기하는 원인을
찾고 싶습니다.

”
...”

✓ 문제 정의

1. 불량을 야기하는 원인 확인
2. 불량율을 낮추기 위한 분석

목표 설정

- ✓ 목표 설정: 제공받은 데이터에서 불량품을 야기하는 중요 변수와 불량을 판정하는 기준을 찾습니다.

변수 확인 및 EDA > 샘플링 > 샘플링에 따른 모델별 성능 확인 > 변수 중요도 확인 > 결론 도출

데이터 확인하기

✓ 타겟 변수 : 불량여부를 나타내는 이진(0, 1) 데이터

✓ 설명 변수 : 변수에 대한 설명이 없는 비식별화된 20개의 변수

X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	...	X12	X13	X14	X15	X16	X17	X18	X19	X20	Y
0.202296	0.003560	0.411261	0.015348	0.693147	0.036360	0.343512	0.028087	0.681974	0.295769	...	0.682731	0.249262	0.681974	0.016330	0.242926	0.557461	0.028087	0.682731	0.036360	0
0.006836	0.007688	0.597304	0.015348	0.686838	0.067407	0.347869	0.028640	0.497403	0.297943	...	0.680891	0.249262	0.497403	0.670040	0.245876	0.561336	0.028640	0.680891	0.067407	0
0.609621	0.001461	0.466186	0.015348	0.693090	0.018944	0.345032	0.024502	0.647685	0.298255	...	0.685525	0.249262	0.647685	0.648486	0.243602	0.559431	0.024502	0.685525	0.018944	0
0.074007	0.003072	0.535876	0.015348	0.693147	0.031475	0.347502	0.025393	0.587787	0.296057	...	0.677980	0.249262	0.575364	0.632563	0.246078	0.556761	0.025393	0.677980	0.031475	0
0.198356	0.007020	0.416238	0.015348	0.686838	0.061888	0.347441	0.028450	0.537143	0.298167	...	0.673286	0.249262	0.537143	0.671124	0.245505	0.561695	0.028450	0.673286	0.061888	0
...	
0.612937	0.001173	0.401400	0.015348	0.693147	0.015237	0.346218	0.020338	0.470004	0.298167	...	0.679403	0.249262	0.483797	0.664080	0.243377	0.561398	0.020338	0.679403	0.015237	1
0.512265	0.002282	0.395505	0.015348	0.688810	0.021745	0.342540	0.024307	0.681974	0.297209	...	0.687581	0.249262	0.670674	0.658006	0.240500	0.555487	0.024307	0.687581	0.021745	1
0.390334	0.008615	0.415327	0.015348	0.693147	0.068450	0.451511	0.028167	0.470004	0.599760	...	0.680428	0.249262	0.456017	0.682601	0.693147	0.693040	0.028167	0.680428	0.068450	1
0.512265	0.002267	0.394744	0.015348	0.688810	0.020826	0.341926	0.024307	0.693147	0.296889	...	0.689263	0.249262	0.681974	0.658006	0.240500	0.557215	0.024307	0.689263	0.020826	1
0.118755	0.003229	0.398373	0.015348	0.510595	0.028349	0.343154	0.025920	0.647685	0.297209	...	0.683904	0.249262	0.635989	0.627492	0.241516	0.559431	0.025920	0.683904	0.028349	1

✓ data 크기 : 527000 × 21

✓ dataframe 이름 : defect

✓ defect.isnull().sum() >> 0 # null 없음

✓ defect.info() >>

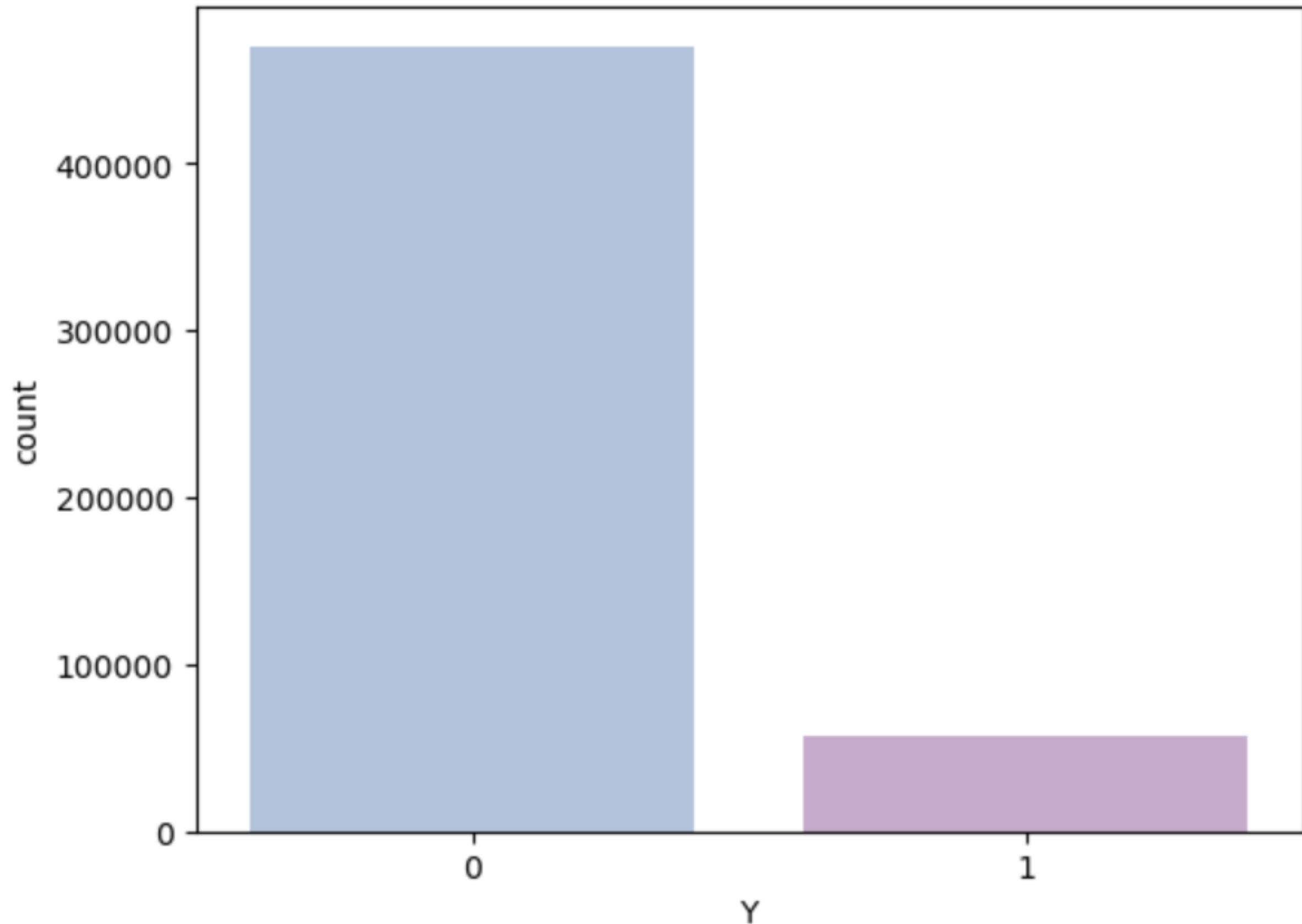
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 527000 entries, 0 to 526999
Data columns (total 21 columns):
 #   Column   Non-Null Count  Dtype  
--- 
 0   X1        527000 non-null   float64
 1   X2        527000 non-null   float64
 2   X3        527000 non-null   float64
 3   X4        527000 non-null   float64
 4   X5        527000 non-null   float64
 5   X6        527000 non-null   float64
 6   X7        527000 non-null   float64
 7   X8        527000 non-null   float64
 8   X9        527000 non-null   float64
 9   X10       527000 non-null   float64
 10  X11       527000 non-null   float64
 11  X12       527000 non-null   float64
 12  X13       527000 non-null   float64
 13  X14       527000 non-null   float64
 14  X15       527000 non-null   float64
 15  X16       527000 non-null   float64
 16  X17       527000 non-null   float64
 17  X18       527000 non-null   float64
 18  X19       527000 non-null   float64
 19  X20       527000 non-null   float64
 20  Y         527000 non-null   int64
dtypes: float64(20), int64(1)
memory usage: 84.4 MB
```

종속 변수 : 20개의 수치형 데이터

✓ defect.describe() >>

	X1	X2	X3	X4
count	527000.000000	527000.000000	527000.000000	5.270000e+05
mean	0.375129	0.003963	0.455679	1.534800e-02
std	0.200043	0.004316	0.105150	8.274639e-14
min	0.000000	0.000000	0.000000	1.534800e-02
25%	0.204692	0.002366	0.415990	1.534800e-02
50%	0.399744	0.003523	0.435739	1.534800e-02
75%	0.545196	0.005094	0.530429	1.534800e-02
max	0.693147	0.693147	1.534800e-02	

EDA

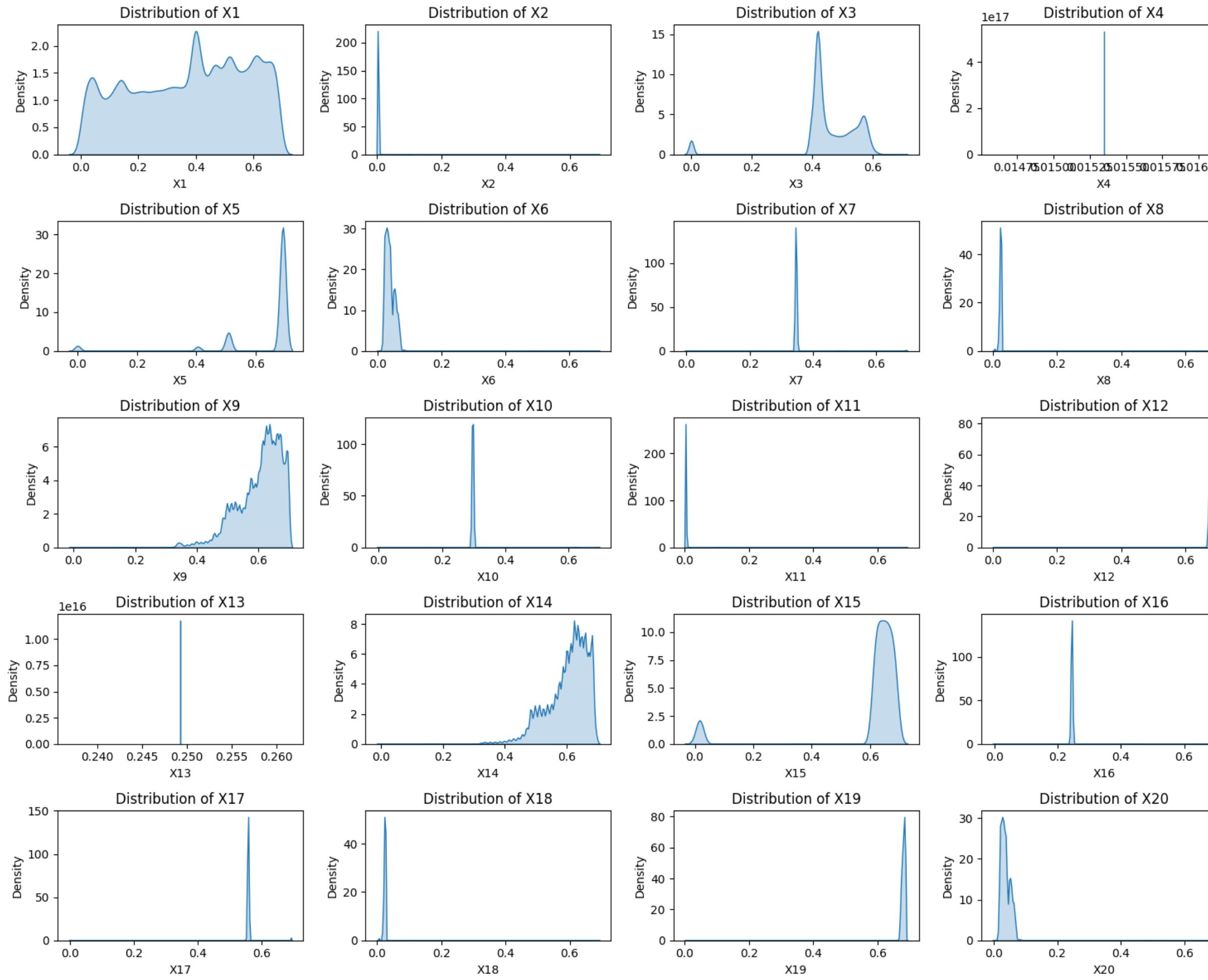


✓ `sns.countplot(x = 'Y', data = defect)`

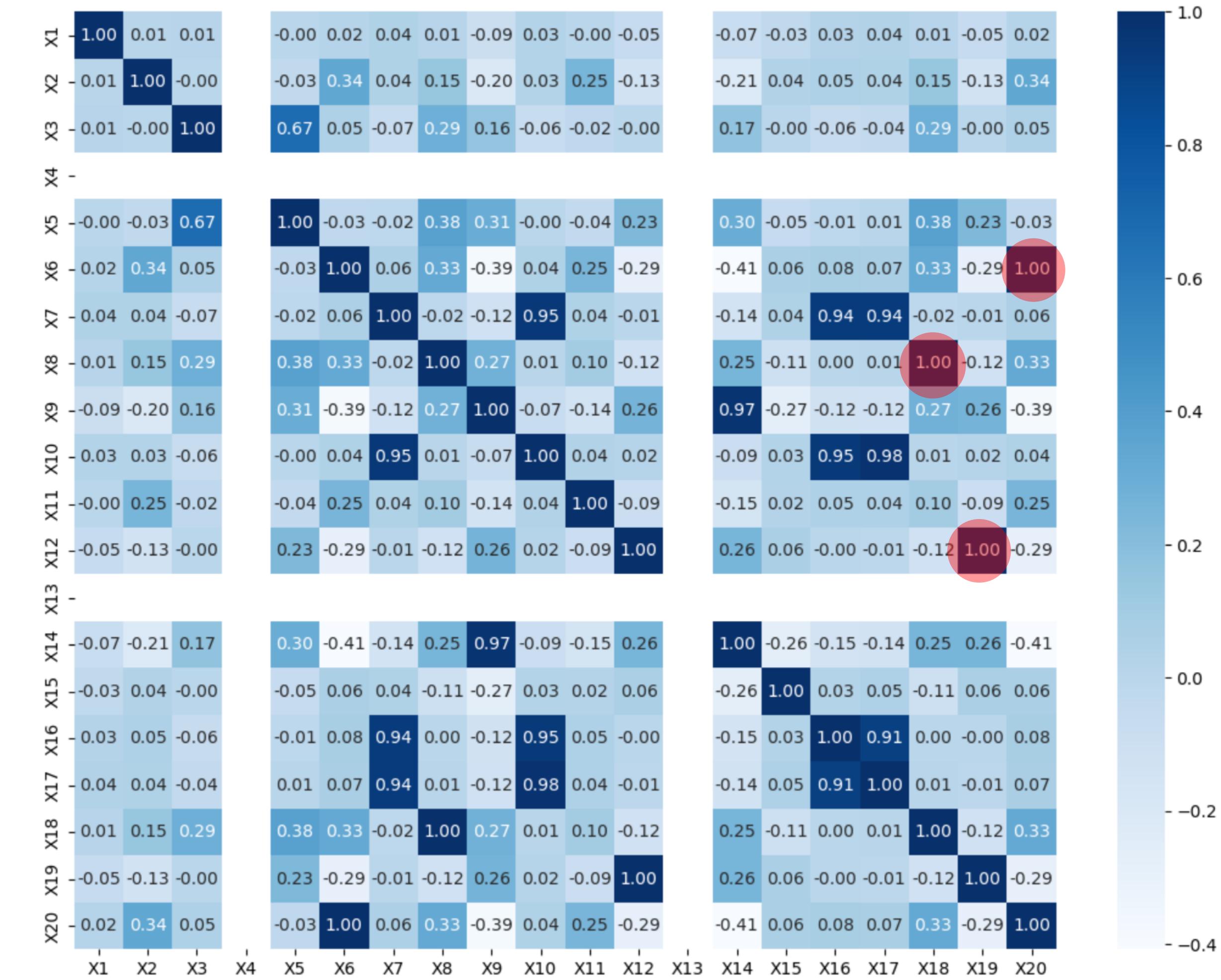
“1” 데이터 비율이 10.82% 정도로, 이는 굉장히 불균형 데이터임을 알 수 있습니다.

EDA

✓ KDE plot을 통해 각 변수의 분포 시각화



✓ sns.heatmap(defect.corr()) 흰색 빈칸(Nan), 상관계수 1 발견



EDA

- ✓ 상관계수 1인 데이터 확인

서로 뺀 결과가 0, 즉 서로 동일한 값이 들어간 열
중복된 데이터를 제거함

- ✗ X18, X19, X20 제거

```
print(defect_X["X6"] - defect_X["X20"].unique())
print(defect_X["X8"] - defect_X["X18"].unique())
print(defect_X["X12"] - defect_X["X19"].unique())
✓ 0.0s
```

[0.]
[0.]
[0.]

- ✓ heatmap 흰색 빈칸 데이터 확인

열 안의 값이 모두 동일, 단일값을 가짐
분류에 영향을 미치지 않을 것으로 판단됨

- ✗ X4, X13 제거

```
print(defect_X['X4'].unique())
print(defect_X['X13'].unique())
✓ 0.0s
```

[0.015348]
[0.2492619]

EDA

- ✓ 다중공선성 확인

대부분 10 이상의 높은 수치

	VIF Factor	features
0	4.608706	X1
1	2.183903	X2
2	38.570471	X3
3	48.205489	X5
4	9.985000	X6
5	1835.454304	X7
6	73.145095	X8
7	1489.955096	X9
8	2305.708857	X10
9	1.845529	X11
10	13143.749063	X12
11	1741.138450	X14
12	15.746152	X15
13	828.369567	X16
14	20745.209270	X17

- ✓ train, test 데이터 → 층화 임의 추출을 통해 분리 데이터 분포 유지

```
# 결과 출력 (훈련 데이터에서 클래스 비율 확인)
print("훈련 데이터에서 클래스 분포:\n", y_train.value_counts(normalize=True))
print("테스트 데이터에서 클래스 분포:\n", y_test.value_counts(normalize=True))
```

✓ 0.1s

훈련 데이터에서 클래스 분포:

```
Y
0    0.891841
1    0.108159
Name: proportion, dtype: float64
```

테스트 데이터에서 클래스 분포:

```
Y
0    0.891841
1    0.108159
Name: proportion, dtype: float64
```

샘플링

Under Sampling

Under Sampling

- Random Under Sampling
- Tomek Links

Over Sampling

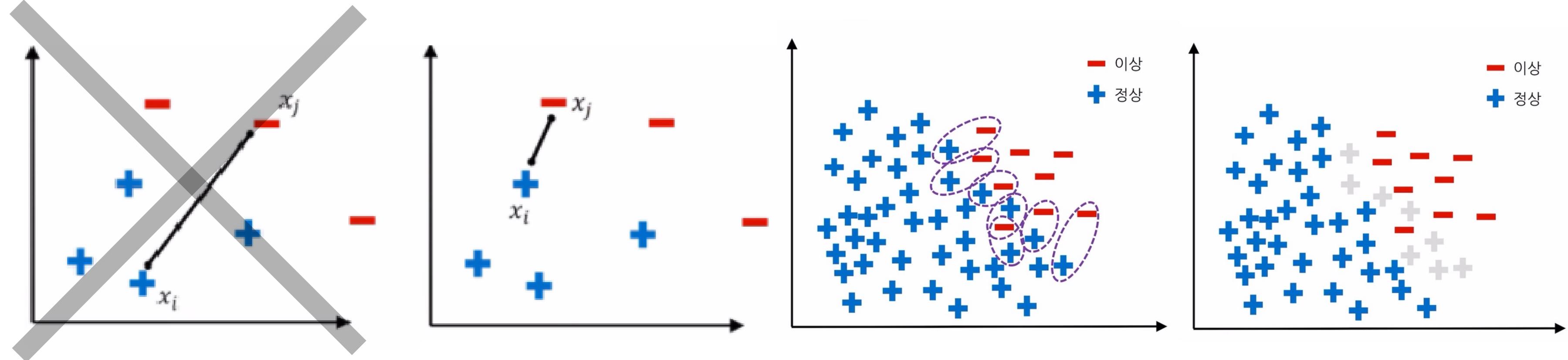
- Random Over Sampling
- SMOTE

Both Sampling

- SMOTETomek

✓ **Random Under Sampling** : 다수 범주에서 무작위로 샘플링하는 방법

✓ **Tomek Links** : 두 범주 사이를 탐지하고 정리를 통해 부정확한 분류경계선을 방지하는 방법



- 두 범주에서 하나씩 데이터 추출.
- $d(x_i, x_k) < d(x_i, x_j)$ 또는 $d(x_i, x_k) < d(x_i, x_i)$ 가 되는 관측치 x_k 가 없는 경우
- 선택한 두 데이터 간의 링크를 Tomek Link라 부름. Tomek Links를 형성한 후, 다수 범주에 속한 관측치를 제거.

샘플링

Over Sampling

Under Sampling

- Random Under Sampling
- Tomek Links

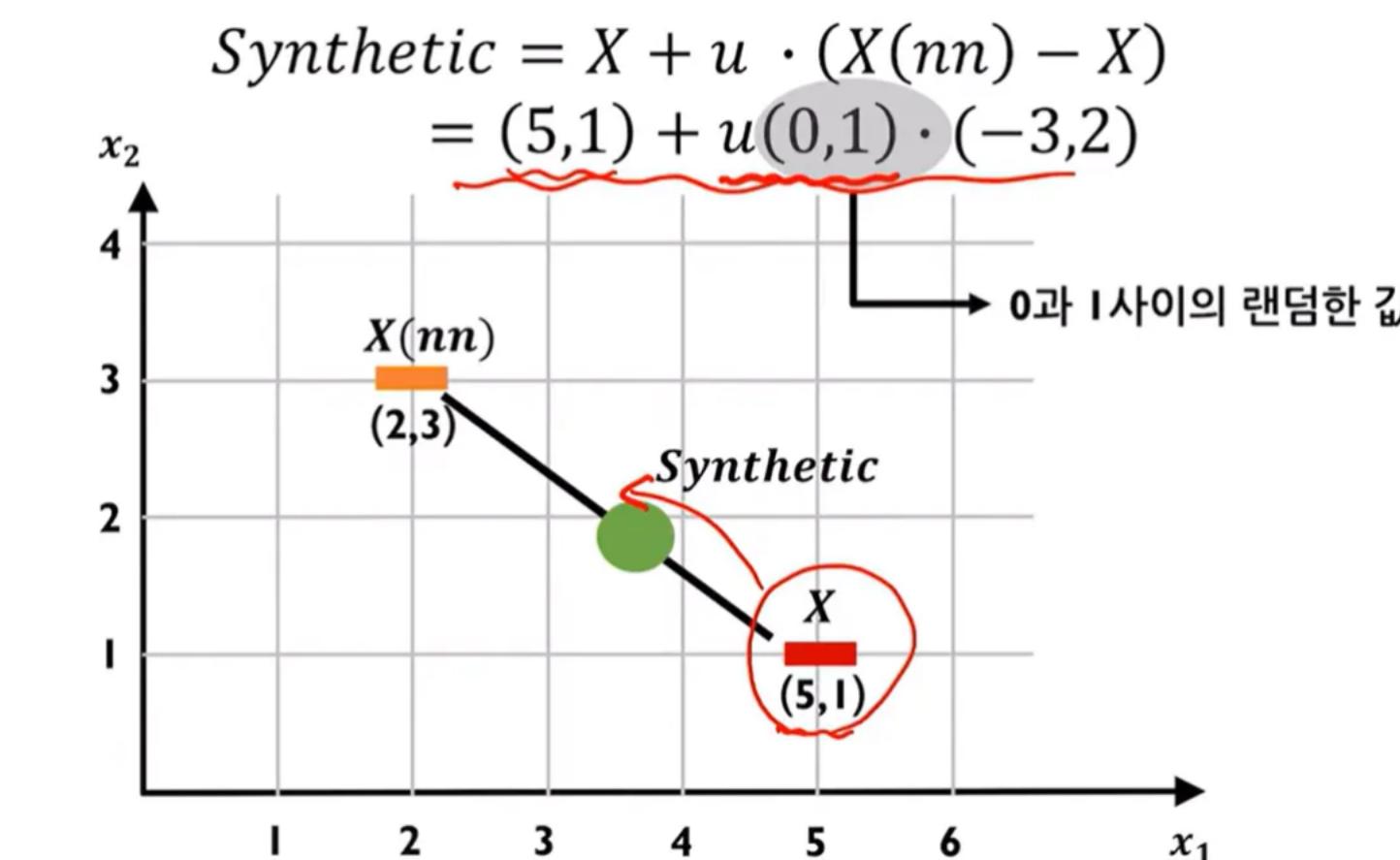
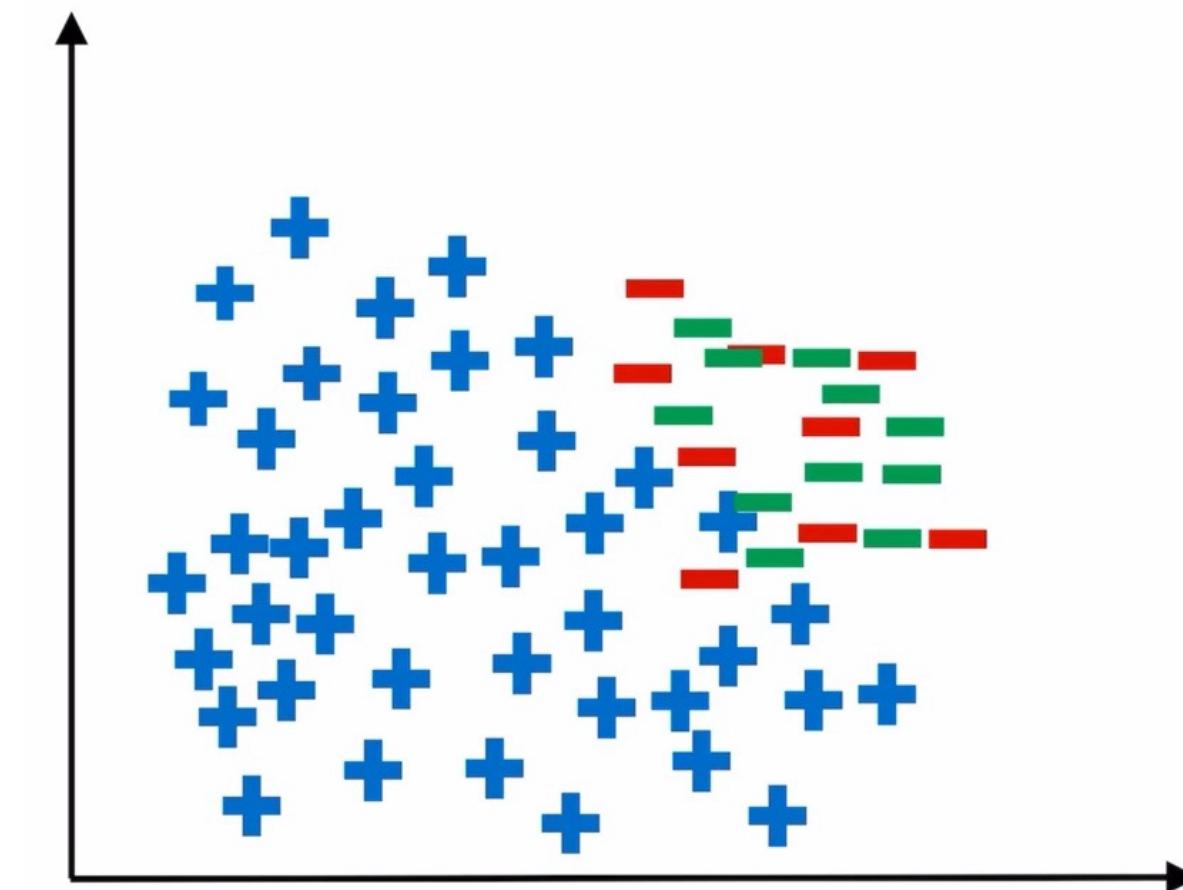
Over Sampling

- Random Over Sampling
- SMOTE

Both Sampling

- SMOTETomek

- ✓ **Random Over Sampling** : 소수 범주의 데이터 수를 다수 범주의 데이터 수와 비슷해지도록 증가시키는 방법, 무작위로 복제
- ✓ **SMOTE** : Synthetic Minority Oversampling Technique, 소수 범주에서 가상의 데이터를 생성하는 방법

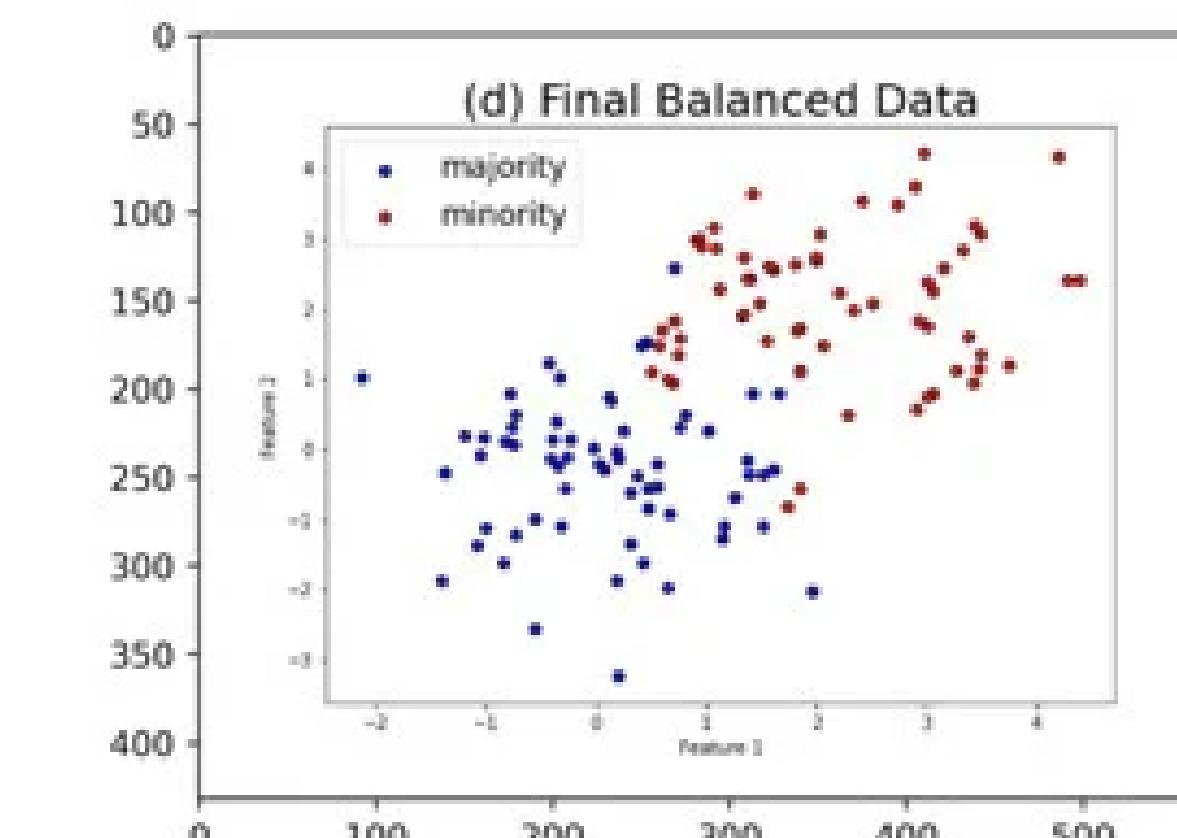
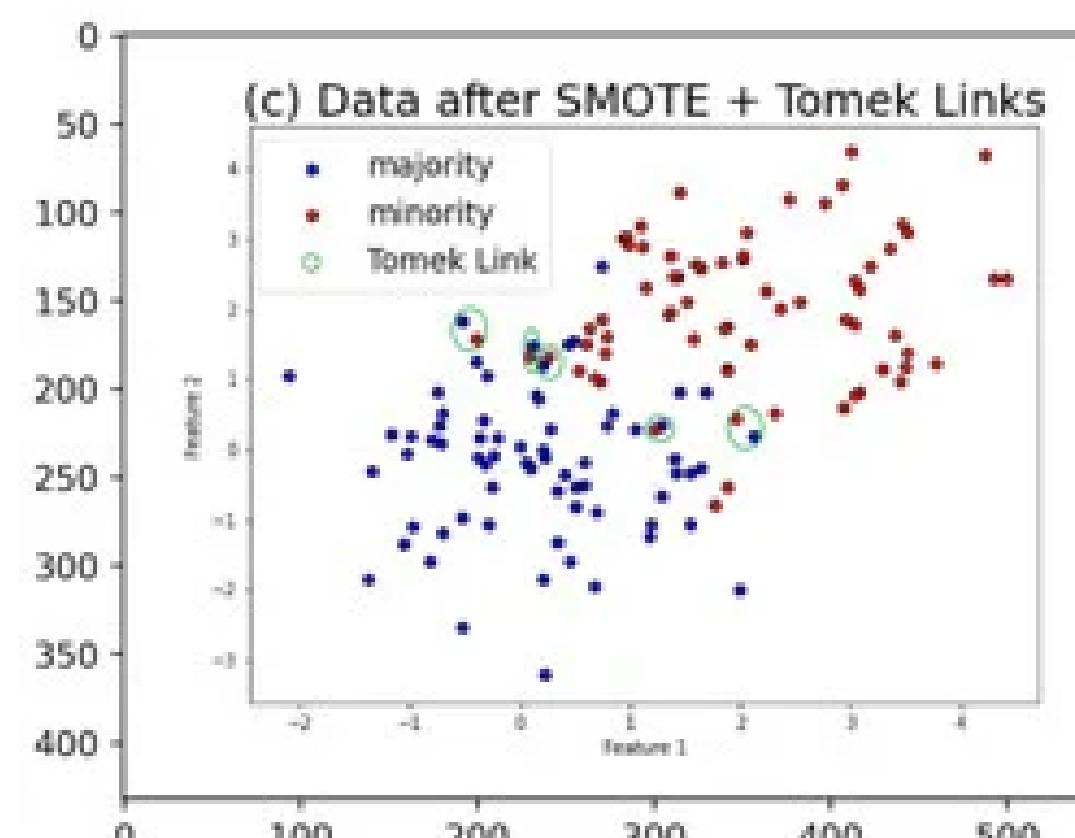
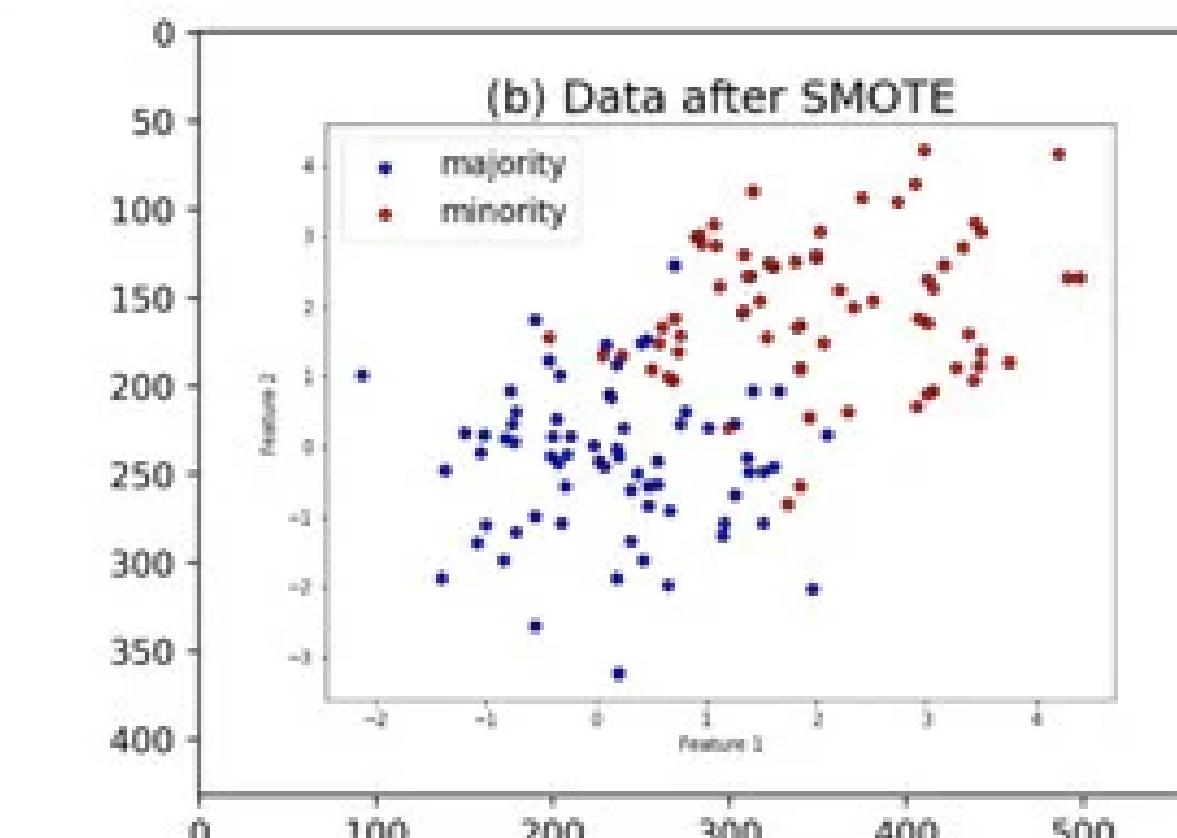
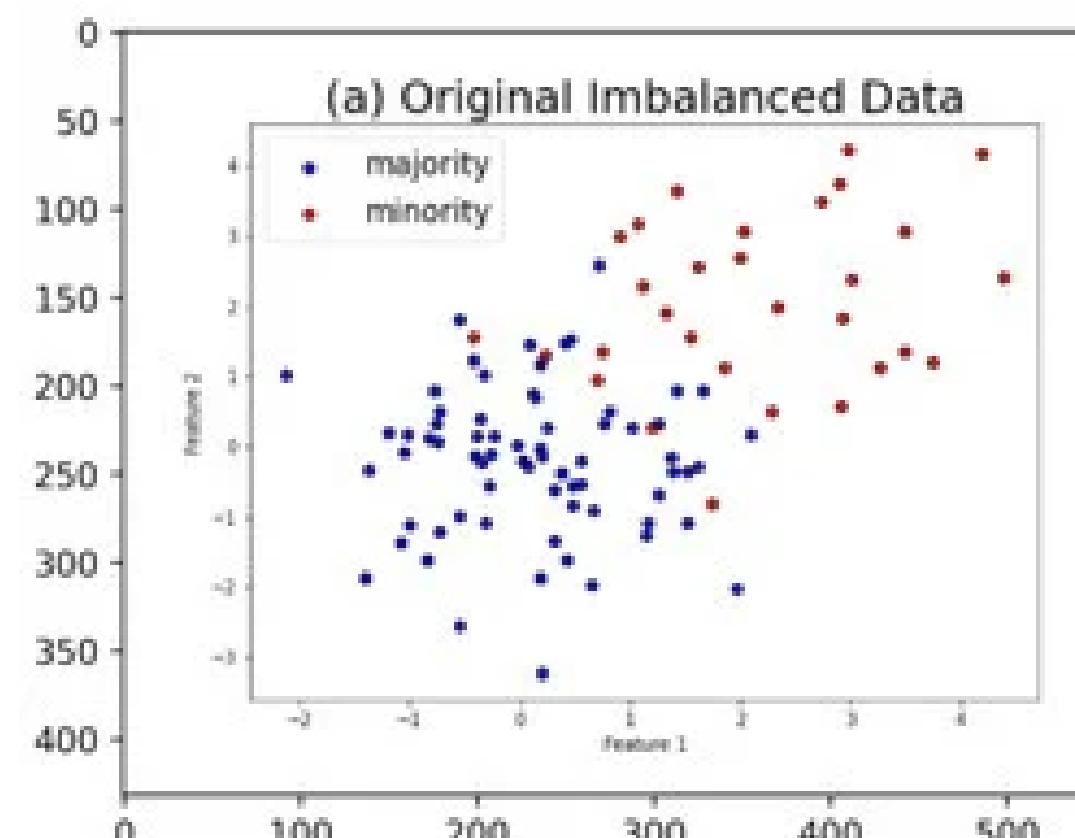


- k 값을 정한 후 소수 클래스 중 임의의 소수 데이터를 선정
- 근처의 k개의 소수 데이터 중 랜덤하게 하나의 관측치를 선택
- 선택된 두 관측치에 대해 다음 식을 통해 가상의 관측치를 생성
- 원하는 비율의 소수 클래스가 생성될 때까지 반복

샘플링

Both Sampling

- ✓ **SMOTE-Tomek**: SMOTE와 Tomek를 결합한 방식
 - 우선 SMOTE를 진행한 다음에 Tomek을 진행



Under Sampling

- Random Under Sampling
- Tomek Links

Over Sampling

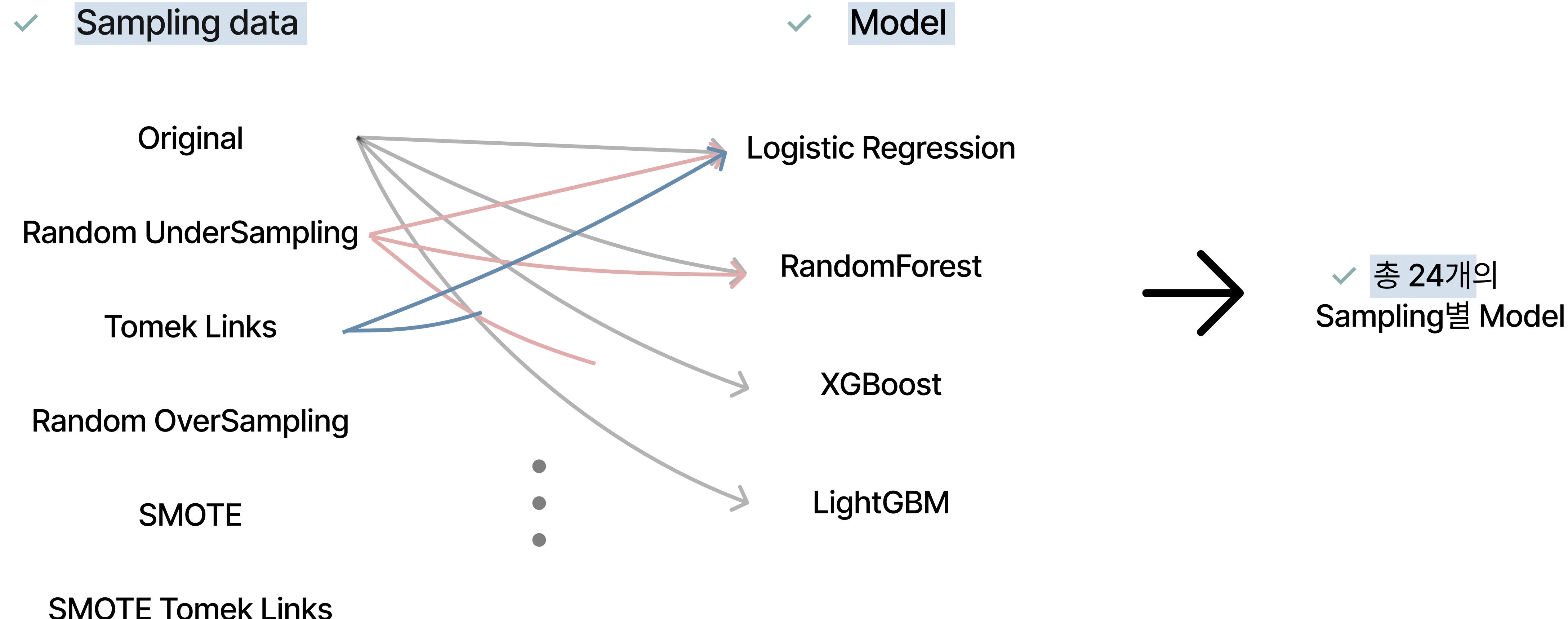
- Random Over Sampling
- SMOTE

Both Sampling

- SMOTETomek

샘플링

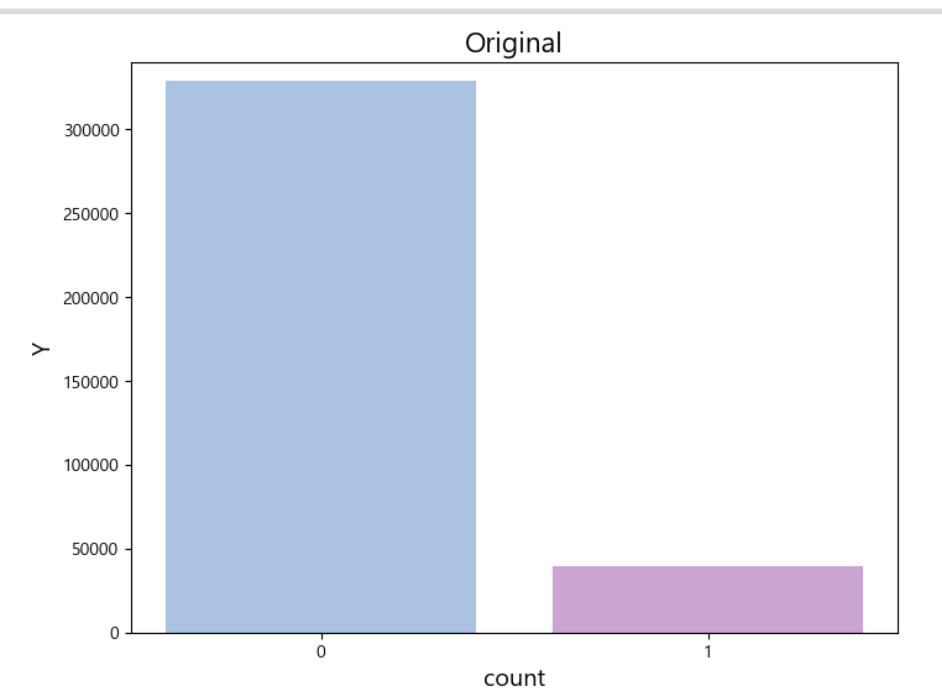
✓ Sampling data



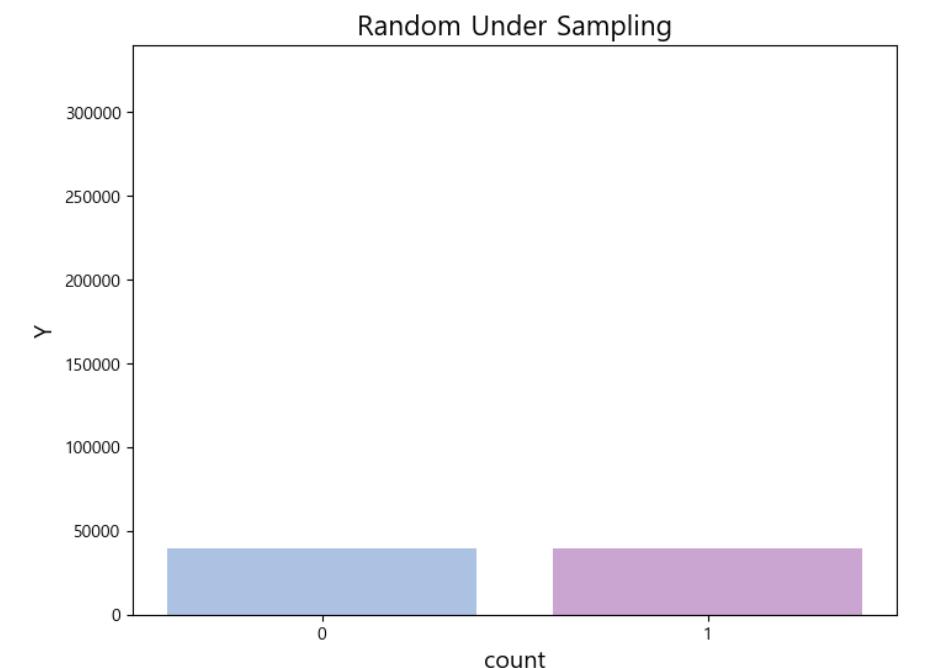
train과 test data를 계층적 분할한 후, 각 샘플링 방법을 사용해 불균형한 train_x, train_y 데이터를 샘플링 합니다. 이 샘플링한 train data를 모델에 적용 후 학습합니다.

샘플링에 따른 모델별 성능 확인

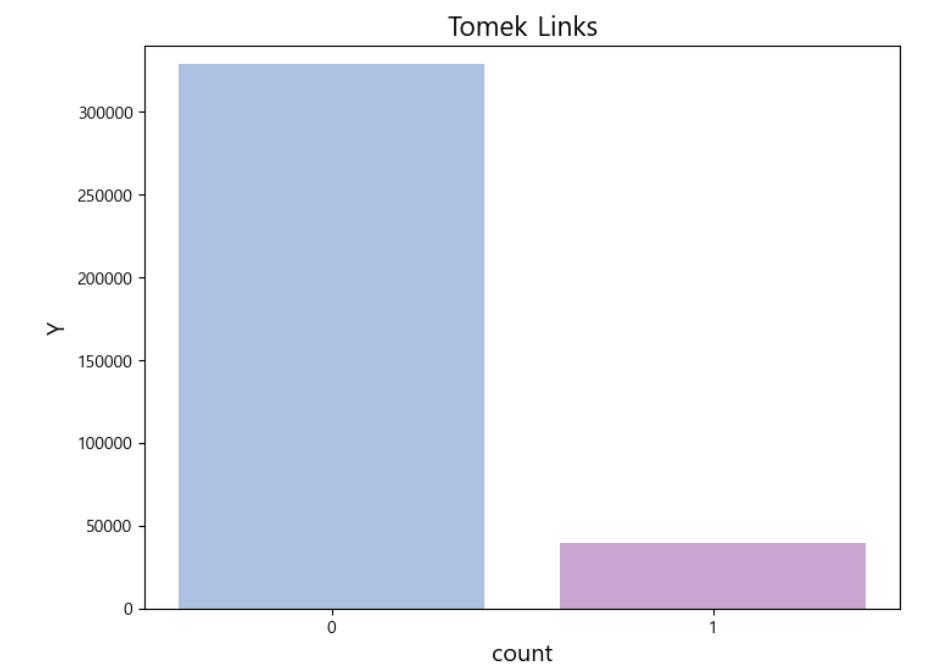
- ✓ Original
329000, 39900



- ✓ Random UnderSampling
39900, 39900



- ✓ Tomek Links
328971, 39900



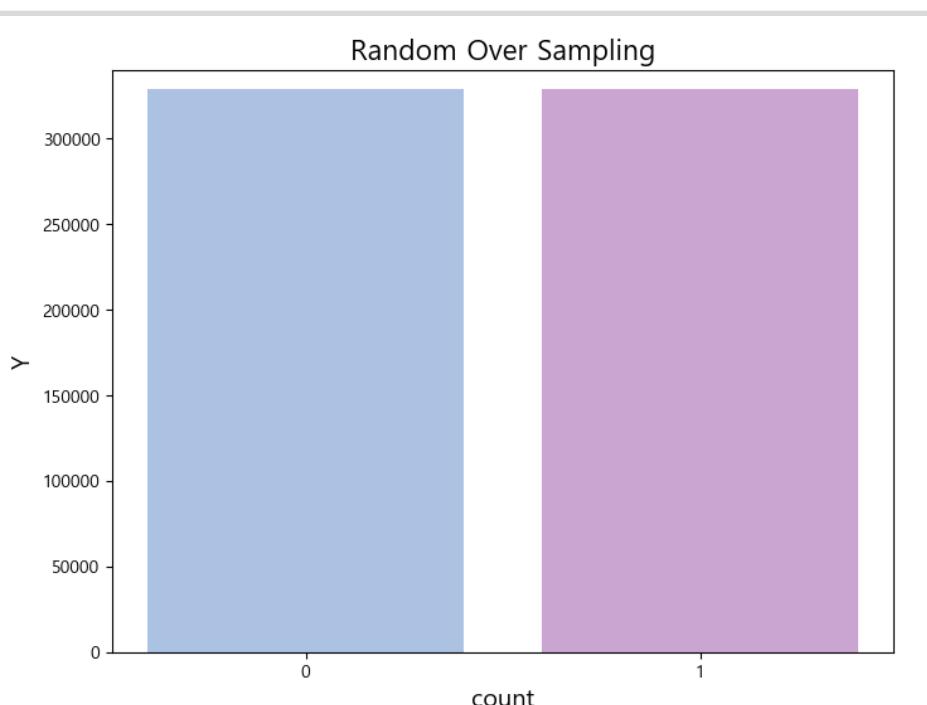
	model	F1-score	G-means	FN count	AUC score
0	Logistic Regression	0.944333	0.946625	1773	0.948037
1	Random Forest	0.998302	0.998406	54	0.998407
2	XGBoost	0.998565	0.998618	47	0.998619
3	Light GBM	0.996753	0.998038	61	0.998039

	model	F1-score	G-means	FN count	AUC score
0	Logistic Regression	0.889303	0.981351	141	0.981406
1	Random Forest	0.998244	0.998553	48	0.998554
2	XGBoost	0.998420	0.998832	38	0.998832
3	Light GBM	0.998068	0.998455	51	0.998456

	model	F1-score	G-means	FN count	AUC score
0	Logistic Regression	0.944463	0.946749	1769	0.948154
1	Random Forest	0.998360	0.998464	52	0.998465
2	XGBoost	0.998419	0.998497	51	0.998498
3	Light GBM	0.995298	0.997861	61	0.997862

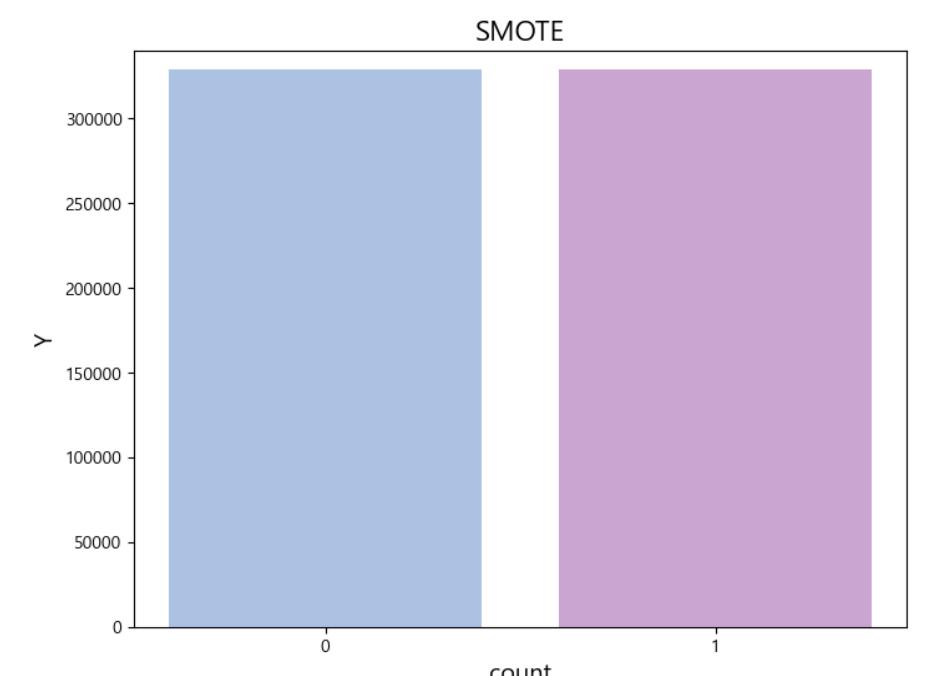
샘플링에 따른 모델별 성능 확인

- ✓ Random OverSampling
329000 , 329000



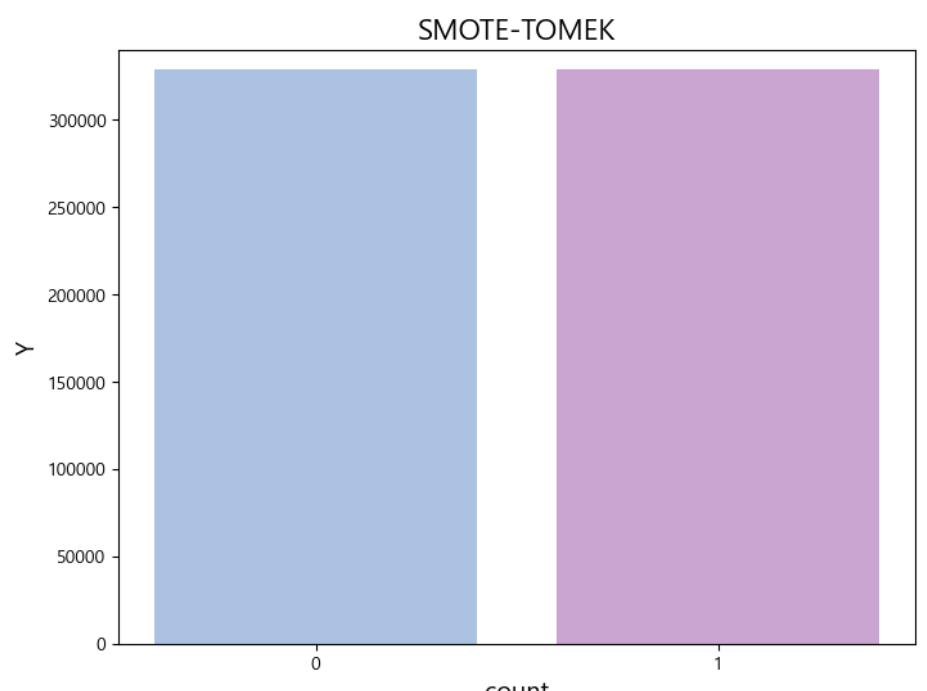
	model	F1-score	G-means	FN count	AUC score
0	Logistic Regression	0.980036	0.993976	139	0.993978
1	Random Forest	0.998595	0.998621	47	0.998622
2	XGBoost	0.998448	0.998604	47	0.998604
3	Light GBM	0.996842	0.998280	52	0.998281

- ✓ SMOTE
329000, 329000



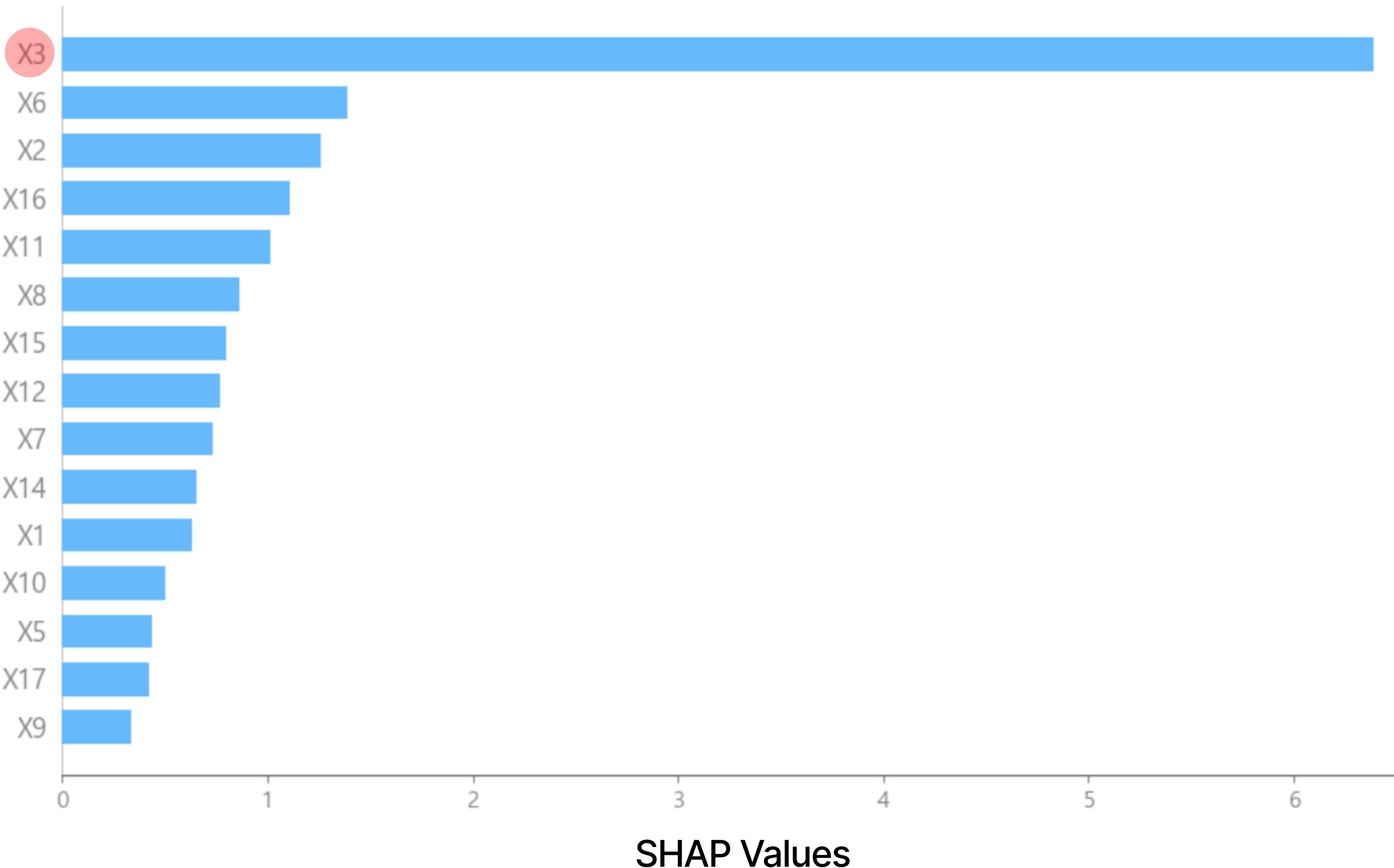
	model	F1-score	G-means	FN count	AUC score
0	Logistic Regression	0.979328	0.993862	140	0.993864
1	Random Forest	0.998829	0.998958	35	0.998959
2	XGBoost	0.998654	0.999040	31	0.999040
3	Light GBM	0.997660	0.998559	45	0.998560

- ✓ SMOTE Tomek
328953, 328953



	model	F1-score	G-means	FN count	AUC score
0	Logistic Regression	0.980516	0.993985	141	0.993987
1	Random Forest	0.998859	0.998936	36	0.998937
2	XGBoost	0.998771	0.999028	32	0.999029
3	Light GBM	0.998273	0.998582	47	0.998583

변수 중요도 확인



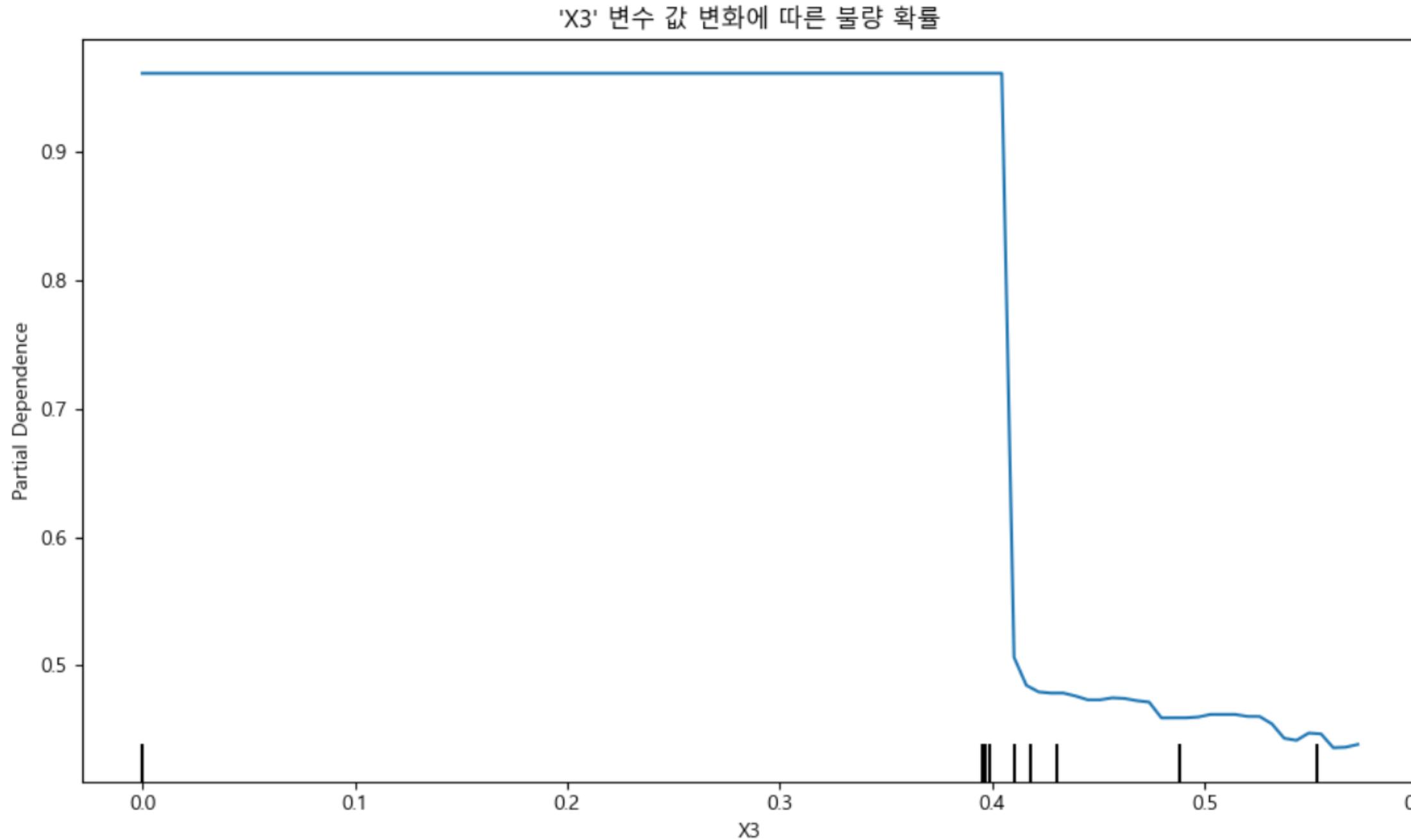
✓ 특성 중요도 시각화

SMOTE 방식으로 오버샘플링한 XGBoost의 **변수 중요도** 시각화

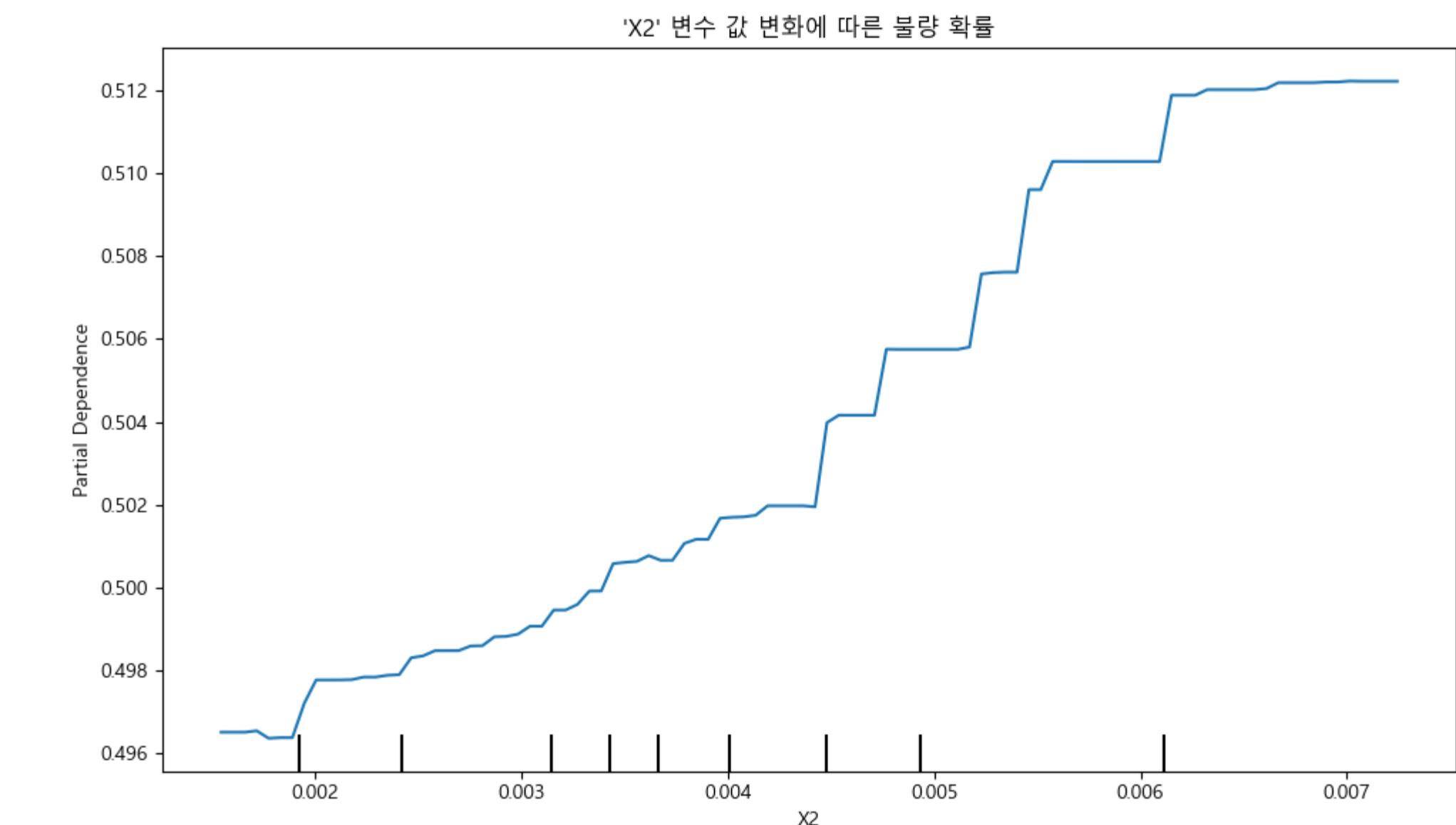
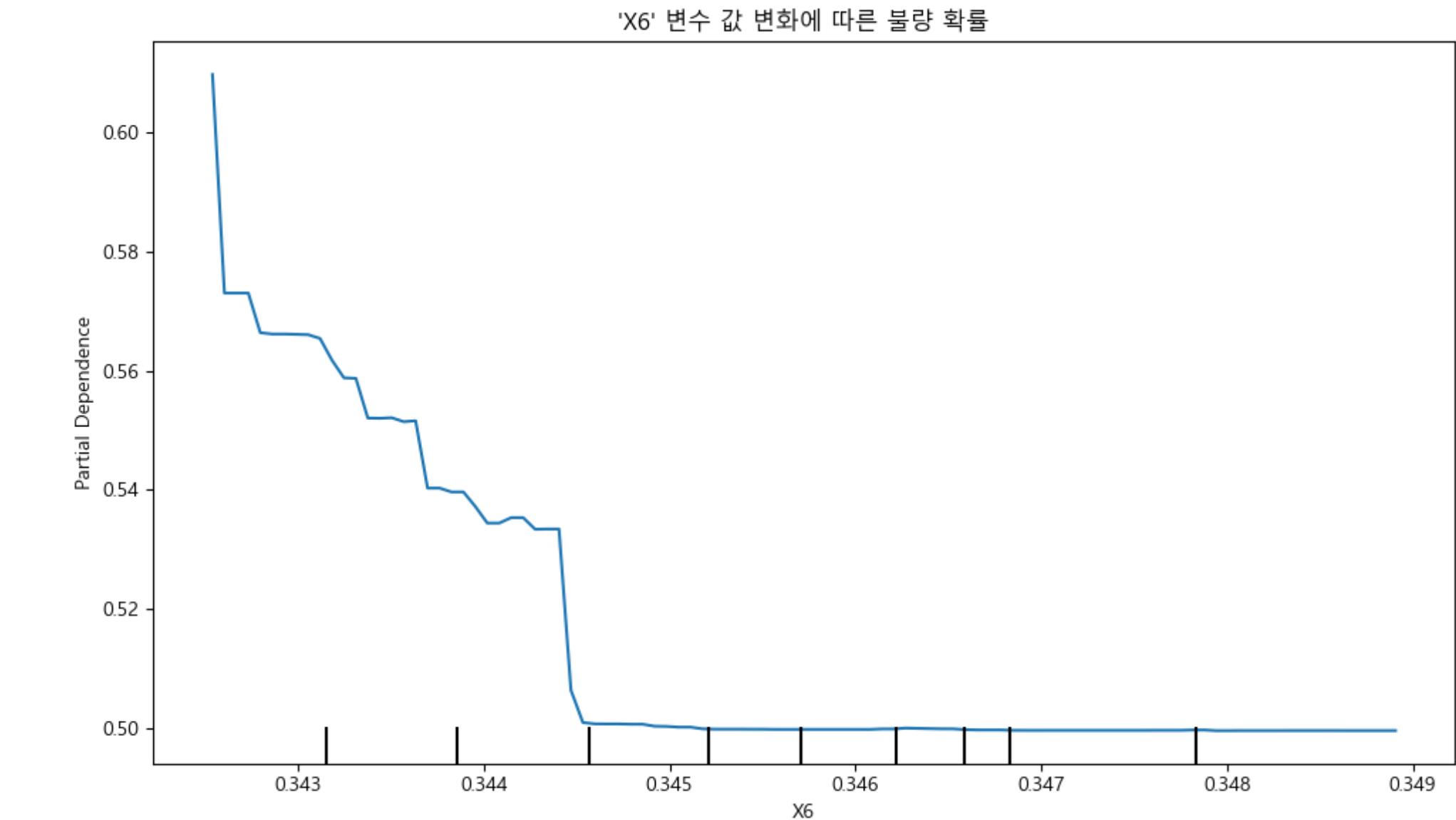
→**X3**: 불량품, 양품 분류에 가장 큰 역할

변수 중요도 확인

- ✓ 변수 중요도가 가장 높은 X3, X6, X2의 PDP 확인

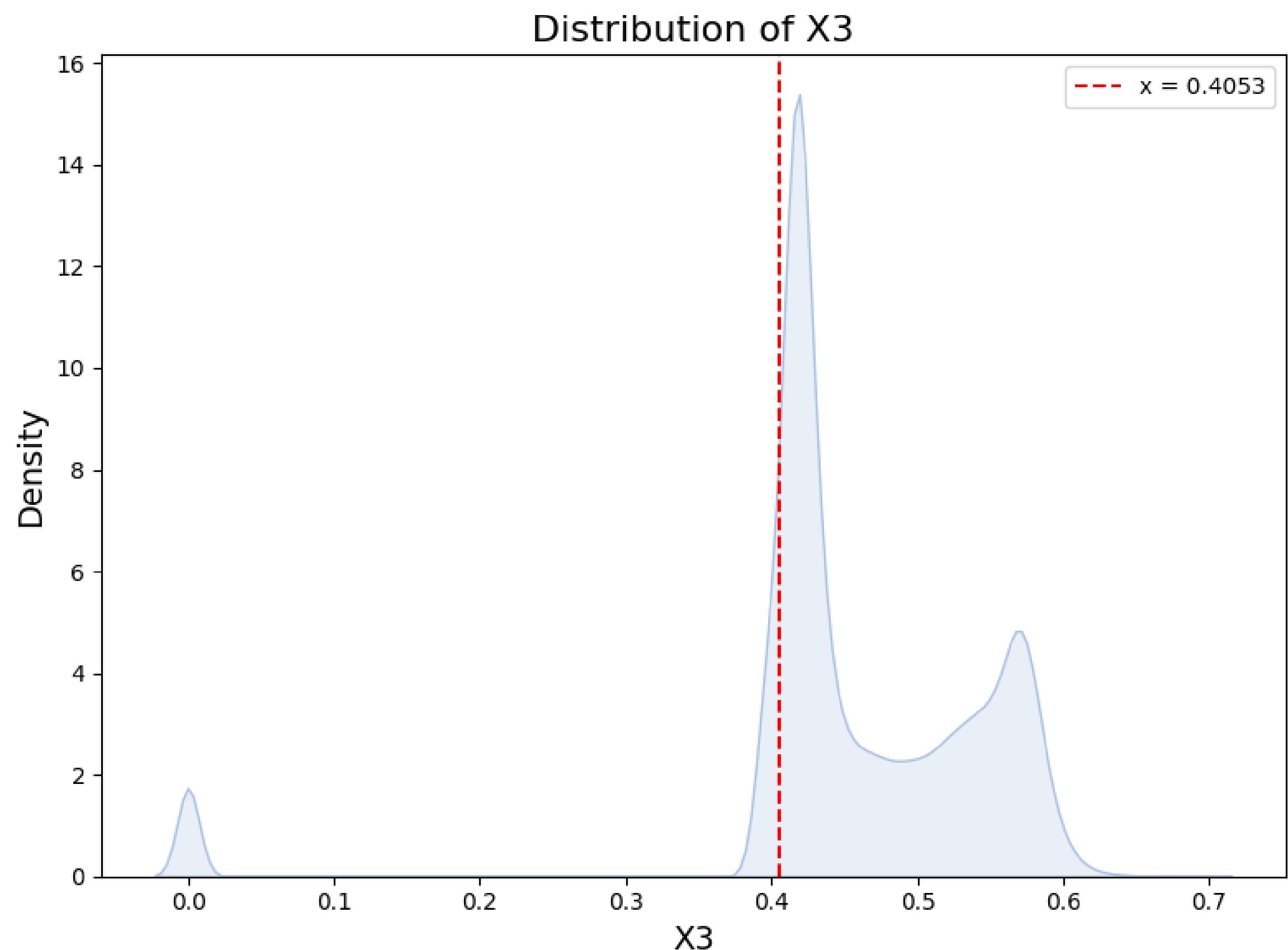


- X3은 기준 0.4보다 작은 구간에서 불량률이 0.9 이상으로 높다.
따라서, 0.4보다 높은 구간에서 불량률이 줄어든 것이 확인된다.
- X6의 경우, 증가할수록 불량률이 감소하는 형태를 보인다.
- X2는 증가할수록 불량률이 증가하는 형태를 보인다.



결론 도출

		X3 < 0.4053	X3 > = 0.4053
		Y=0	0
		Y=1	53352
			470000
			3648



- ✓ 불량을 야기하는 중요 변수는 **X3**입니다. 즉, 양품 및 불량품 선정 시 변수 **X3**에 큰 주의를 기울여야 할 것으로 보입니다.
특히 변수 **X3이 0.4053 이하인 경우 데이터의 전체가 불량품인것을 확인할 수 있습니다.**
따라서, **X3이 0.4053 미만으로 내려가지 않도록 유의하시기 바랍니다.**

Thanks

Team 4

오서연·이재준·장일준·정아영·최지원