

데이터 전처리 고급

1. 피쳐 엔지니어링 소개

[1] 피쳐 엔지니어링 정의

- 피쳐 엔지니어링은 데이터를 분석하거나 모델을 학습시키기 전에 데이터를 변형하고 처리하는 과정을 말한다.
- 머신러닝 모델의 성능을 향상시키기 위해 매우 중요한 단계이다.
- 피쳐 엔지니어링은 원시 데이터를 더욱 유용한 형식으로 변환하는 것을 목표로 한다.
- 이를 통해 모델이 데이터를 더 잘 이해하고 예측할 수 있게 된다.
- 다양한 기법이 있으며, 도메인 지식을 활용하여 새로운 피쳐를 생성하거나 기존 피쳐를 변형하는 것이 포함된다.

[2] 데이터의 품질 및 표현 방식

- 데이터의 품질과 표현 방식이 머신러닝 모델의 성능에 직접적인 영향을 미친다.
- 예를 들어, 원시 데이터는 결측값, 잡음 또는 불균형이 있을 수 있다.
- 이러한 문제를 해결하기 위해 데이터 정제 과정을 거쳐야 한다.
- 결측값을 처리하거나, 이상치를 제거하거나, 데이터 정규화를 수행할 수 있다.

[3] 데이터 변형 및 새로운 정보 추출

- 피쳐 엔지니어링은 데이터를 변형하여 새로운 정보를 추출하는 것을 포함한다.
- 예를 들어, 시간 데이터가 주어졌을 때, 요일, 주, 월 등의 추가 피쳐를 생성할 수 있다.
- 이를 통해 모델이 시간적 패턴을 더 잘 이해할 수 있게 된다.

[4] 범주형 데이터 처리

- 범주형 데이터를 처리하는 방법도 피쳐 엔지니어링의 중요한 부분이다.
- 범주형 데이터는 레이블 인코딩이나 원-핫 인코딩을 통해 수치형 데이터로 변환할 수 있다.
- 이를 통해 모델이 범주형 데이터의 관계를 학습할 수 있게 된다.

[5] 데이터 스케일 맞추기

- 데이터의 스케일을 맞추는 것도 중요하다.
- 데이터의 각 피쳐가 서로 다른 단위를 가지거나 범위가 크게 다를 경우, 모델이 이를 제대로 학습하지 못할 수 있다.
- 이 때 표준화나 정규화를 통해 데이터의 스케일을 맞추어 줄 수 있다.

[6] 도메인 지식과 창의성

- 피쳐 엔지니어링은 도메인 지식과 창의성을 필요로 한다.
- 예를 들어, 금융 데이터에서는 이자율, 대출 기간, 대출 금액 등 다양한 피쳐를 생성하여 모델의 성능을 향상시킬 수 있다.

.

[7] 피쳐 엔지니어링의 최종 목표

- 피쳐 엔지니어링의 최종 목표는 모델이 데이터를 더 잘 이해하고, 이를 기반으로 더 정확한 예측을 할 수 있도록 돕는 것이다.
- 이를 통해 데이터의 가치가 극대화될 수 있다.

2. 레이블 인코딩과 로그변환

[1] 레이블 인코딩

- 레이블 인코딩은 범주형 데이터를 수치형 데이터로 변환하는 기법이다.
- 이는 머신러닝 모델이 범주형 데이터를 직접 처리할 수 없기 때문에 필요하다.
- 각 범주를 고유한 숫자로 변환하여 모델이 이를 학습할 수 있게 한다. 수 있기 때문이다.

2. 레이블 인코딩과 로그변환

- 예를 들어, 'red', 'green', 'blue'라는 범주형 데이터를 각각 0, 1, 2로 변환할 수 있다.
- 이렇게 변환된 숫자는 모델이 범주형 데이터의 특성을 이해하고 분석하는 데 도움을 준다.
- 레이블 인코딩은 간단하면서도 효과적인 변환 방법이지만, 범주의 순서가 없는 경우에는 주의가 필요하다.
- 왜냐하면 모델이 이 숫자들을 순서가 있는 값으로 오해할 수 있기 때문이다.
- 이러한 문제를 해결하기 위해 원-핫 인코딩 같은 다른 기법을 사용할 수도 있다.

2. 레이블 인코딩과 로그변환



레이블인코딩예제데이터.csv
스프레드시트

데이터 설명해줘(한글로)



ChatGPT는 실수를 할 수 있습니다. 중요한 정보를 확인하세요.



이 데이터는 다양한 특성과 레이블로 구성된 예제 데이터입니다. 각 열의 설명은 다음과 같습니다:

1. **feature_1**: 숫자로 된 특징 값입니다. (예: 103, 436, 861 등)
2. **feature_2**: 크기를 나타내는 범주형 데이터입니다. (예: medium, small, huge 등)
3. **feature_3**: 실수로 표현된 특징 값입니다. (예: 0.493, 0.041, 0.157 등)
4. **feature_4**: 문자를 사용한 범주형 데이터입니다. (예: C, A, E 등)
5. **label**: 데이터의 레이블을 나타내는 범주형 데이터입니다. (예: cat, mouse, dog 등)

이 데이터는 레이블 인코딩 예제를 위해 사용될 수 있는 자료입니다. 추가적인 분석이나 처리가 필요한 경우 알려주세요. [↗]



	feature_1	feature_2	feature_3	feature_4	label
0	103	medium	0.493	C	cat
1	436	small	0.041	A	mouse
2	861	small	0.157	E	dog
3	271	huge	0.015	B	mouse
4	107	medium	0.813	C	mouse
5	72	tiny	0.065	A	dog
6	701	medium	0.857	B	cat
7	21	tiny	0.839	B	cat
8	615	huge	0.521	D	dog
9	122	big	0.344	E	cat
10	467	small	0.129	C	dog
11	215	huge	0.648	A	dog
12	331	small	0.472	D	mouse
13	459	big	0.063	E	dog
14	88	big	0.139	D	mouse
15	373	huge	0.499	E	cat
16	100	big	0.593	E	cat
17	872	tiny	0.392	C	cat
18	664	big	0.675	E	cat
19	131	small	0.419	D	mouse
20	662	small	0.289	E	cat

2. 레이블 인코딩과 로그변환

범주형 컬럼을 레이블 인코딩 처리해줘.

Label Encoded Data				
	feature_1	feature_2	feature_3	feature_4
1	103	2	0.493	2
2	436	3	0.041	0
3	861	3	0.157	4
4	271	1	0.015	1
5	107	2	0.813	2
6	72	4	0.065	0

범주형 컬럼이 레이블 인코딩 처리된 데이터셋을 확인하실 수 있습니다. 각 범주형 컬럼의 값이 숫자로 변환되었습니다. 추가적으로 필요한 작업이나 질문이 있으시면 알려주세요. [\[~\]](#)

2. 레이블 인코딩과 로그변환

레이블 인코딩된 문자와 숫자 매핑값을 각 컬럼별로 알려줘.



각 컬럼별로 레이블 인코딩된 문자와 숫자 매핑값은 다음과 같습니다:

- **feature_2:**

- big: 0
- huge: 1
- medium: 2
- small: 3
- tiny: 4

- **feature_4:**

- A: 0
- B: 1
- C: 2
- D: 3
- E: 4

- **label:**

- cat: 0
- dog: 1
- mouse: 2

이 매핑값을 통해 범주형 데이터가 어떻게 숫자로 변환되었는지 확인할 수 있습니다. 추가로 궁금한 점이 있으면 말씀해주세요. [\[>-\]](#)



[2] 원-핫 인코딩

(1) 원-핫 인코딩 정의

1. 원-핫 인코딩은 범주형 데이터를 이진 벡터로 변환하는 기법이다.
2. 이는 범주형 데이터의 각 범주를 고유한 이진 벡터로 표현하여, 모델이 범주형 데이터를 수치형 데이터로 이해할 수 있도록 한다.

(2) 변환 방법

- 원-핫 인코딩은 범주형 데이터의 각 범주에 대해 새로운 피쳐(열)를 생성한다.
- 예를 들어, 'red', 'green', 'blue'라는 범주형 데이터가 있을 때, 이를 원-핫 인코딩으로 변환하면 세 개의 새로운 피쳐가 생성된다.
- 'red'는 $[1, 0, 0]$, 'green'은 $[0, 1, 0]$, 'blue'는 $[0, 0, 1]$ 로 변환된다.
- 이처럼 각 범주는 하나의 1과 나머지 0으로 이루어진 벡터로 표현된다.

(3) 원-핫 인코딩의 장점

- 범주형 데이터의 순서를 고려하지 않아도 되기 때문에, 데이터가 순서를 가지지 않는 경우에도 사용할 수 있다.
- 이는 레이블 인코딩의 단점을 보완하는 역할을 한다.
- 모델이 각 범주를 독립적으로 처리할 수 있게 되어 범주 간의 관계를 정확하게 학습할 수 있다.

(4) 원-핫 인코딩의 단점

- 범주의 수가 많을 경우, 데이터의 차원이 급격히 증가할 수 있다.
- 예를 들어, 천 개의 범주가 있는 경우 천 개의 새로운 피쳐가 생성되어 메모리 사용량과 계산 비용이 증가한다.
- 이러한 문제를 해결하기 위해 고차원 데이터를 다루는 효율적인 알고리즘이나 차원 축소 기법을 사용할 수 있다.

(5) 사용 예시

- 원-핫 인코딩은 텍스트 데이터의 단어를 벡터로 표현할 때 자주 사용된다.
- 예를 들어, 'cat', 'dog', 'fish'라는 단어를 원-핫 인코딩하면 각각 $[1, 0, 0]$, $[0, 1, 0]$, $[0, 0, 1]$ 로 변환된다.
- 이를 통해 모델이 각 단어를 개별적으로 인식하고 학습할 수 있게 된다.
- 원-핫 인코딩은 범주형 데이터를 수치형 데이터로 변환하는 효과적인 방법이며, 이를 통해 머신러닝 모델의 성능을 향상시킬 수 있다.

2. 레이블 인코딩과 로그변환



원핫인코딩예제데이터.csv
스프레드시트

데이터 설명해줘(한글로)

ChatGPT는 실수를 할 수 있습니다.

feature_1	
0	D_category
1	E_category
2	C_category
3	E_category
4	E_category
5	B_category
6	C_category
7	C_category
8	C_category
9	E_category
10	D_category



업로드된 데이터셋은 다음과 같이 설명할 수 있습니다:

- 1. 컬럼 이름: `feature_1`
- 2. 데이터 형태: 카테고리 데이터
- 3. 행의 개수: 50개
- 4. 카테고리 종류:
 - ``A_category``
 - ``B_category``
 - ``C_category``
 - ``D_category``
 - ``E_category``

각 행은 `feature_1` 컬럼에 하나의 카테고리 값을 포함하고 있습니다. 이 데이터는 원-핫 인코딩을 통해 각 카테고리 값을 별도의 열로 변환할 수 있습니다. 예를 들어, ``A_category``가 있는 행은 ``A_category`` 열에서 1의 값을 가지며, 나머지 카테고리 열에서는 0의 값을 갖게 됩니다.

더 궁금한 점이나 추가 요청이 있으시면 말씀해 주세요.



2. 레이블 인코딩과 로그변환

원-핫 인코딩 처리해줘.



One-Hot Encoded Dataset

	Category_A_category	Category_B_category	Category_C_category	Category_D_category
1	0	0	0	1
2	0	0	0	0
3	0	0	1	0
4	0	0	0	0
5	0	0	0	0
6	0	1	0	0

업로드된 데이터셋에 대해 원-핫 인코딩을 완료했습니다. 각 행은 카테고리 값에 따라 별도의 열로 변환되었으며, 해당 카테고리에 속하면 1, 그렇지 않으면 0으로 표시됩니다.

예를 들어, 원본 데이터셋의 첫 번째 행에 `D_category`가 포함되어 있었다면, 원-핫 인코딩된 데이터셋의 첫 번째 행은 `Category_D_category` 열에서 1의 값을 가지게 됩니다.

추가적인 분석이나 다른 요청이 있으시면 말씀해 주세요. [\[>\]](#)

2. 레이블 인코딩과 로그변환

[원-핫 인코딩 처리된 데이터]

	A	B	C	D	E	F
1	Category_	Category_	Category_	Category_	Category_	E_category
2	0	0	0	1	0	
3	0	0	0	0	1	
4	0	0	1	0	0	
5	0	0	0	0	1	
6	0	0	0	0	1	
7	0	1	0	0	0	
8	0	0	1	0	0	
9	0	0	1	0	0	
10	0	0	1	0	0	
11	0	0	0	0	1	
12	0	0	0	1	0	
13	0	0	1	0	0	
14	0	0	0	0	1	
15	0	1	0	0	0	
16	0	0	0	1	0	
17	0	1	0	0	0	
18	0	0	0	1	0	
19	0	0	0	0	1	
20	1	0	0	0	0	

[원본 데이터]

	A
1	feature_1
2	D_category
3	E_category
4	C_category
5	E_category
6	E_category
7	B_category
8	C_category
9	C_category
10	C_category
11	E_category
12	D_category
13	C_category
14	E_category
15	B_category
16	D_category
17	B_category
18	D_category
19	E_category
20	A_category

[3] 로그변환

- 로그변환은 데이터의 분포를 정규화하기 위해 사용하는 기법이다.
- 주로 데이터가 치우쳐 있는 경우에 사용되며, 로그함수를 적용하여 데이터의 분포를 더 대칭적으로 만든다.
- 이를 통해 모델의 성능을 향상시킬 수 있다.
- 예를 들어, 데이터가 매우 큰 값과 작은 값으로 극단적으로 나뉘어 있는 경우 로그변환을 통해 이러한 차이를 완화할 수 있다.

2. 레이블 인코딩과 로그변환

- 로그변환은 데이터의 스케일을 조정하여 모델이 더 안정적으로 학습할 수 있도록 돕는다.
- 예를 들어, 소득 데이터나 주가 데이터와 같이 큰 범위의 값을 가지는 데이터에 효과적이다.
- 로그변환을 적용하면 데이터의 분포가 정규분포에 더 가까워지기 때문에 통계적 분석 및 모델링에 유리하다.
- 로그변환은 데이터의 분산을 줄이고, 이상치의 영향을 완화하는데도 유용하다.
- 이를 통해 모델이 데이터의 전반적인 패턴을 더 잘 학습할 수 있다.

2. 레이블 인코딩과 로그변환



gpu_transcount.csv

스프레드시트

📎 데이터셋 설명해줘(한글로)



ChatGPT는 실수를 할 수 있습니다. 중요한 정보를 확인하세요.



주어진 데이터셋 `gpu_transcount.csv`는 GPU 트랜잭션 수와 해당 연도를 포함하고 있습니다. 데이터의 구조는 다음과 같습니다:

- `gpu_trans_count`: 특정 연도의 GPU 트랜잭션 수를 나타내는 숫자 값입니다.
- `year`: 트랜잭션 수가 기록된 연도를 나타내는 숫자 값입니다.

데이터셋의 예시 행은 다음과 같습니다:

gpu_trans_count	year
3,500,000	1997
8,000,000	1998
15,000,000	1999
17,500,000	1999
20,000,000	2000

2. 레이블 인코딩과 로그변환

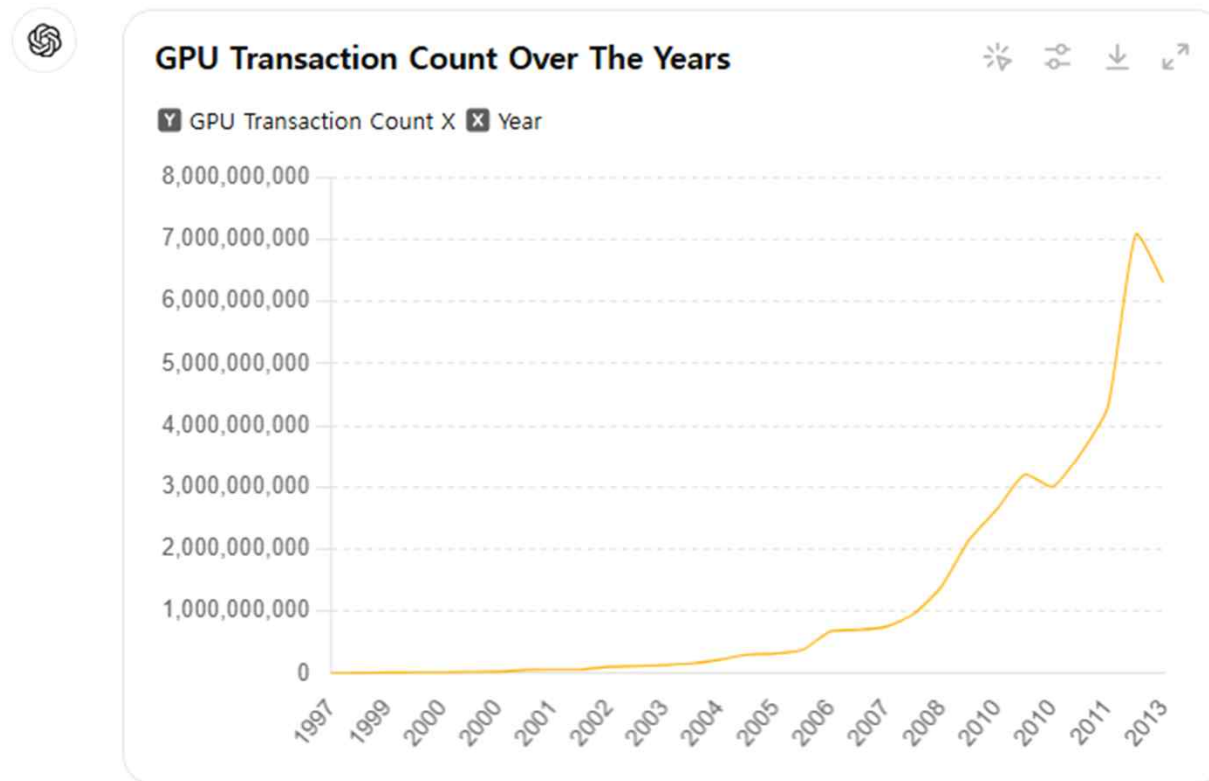
이 데이터셋은 특정 연도에 발생한 GPU 트랜잭션 수를 보여줍니다. 이를 통해 시간에 따른 GPU 트랜잭션의 변화를 분석할 수 있습니다. 예를 들어, 연도별 GPU 트랜잭션 수의 증가 또는 감소 추이를 파악하여 GPU 시장의 성장 또는 축소를 이해할 수 있습니다.

또한, 로그 변환된 GPU 트랜잭션 수를 사용하여 연도별 성장 패턴을 더 명확하게 시각화할 수 있으며, 이는 기하급수적인 증가 또는 감소 추세를 쉽게 파악하는 데 도움이 됩니다.

🔊 📄 ↺ 🔒 ☆✓

2. 레이블 인코딩과 로그변환

그래프로 출력해줘.

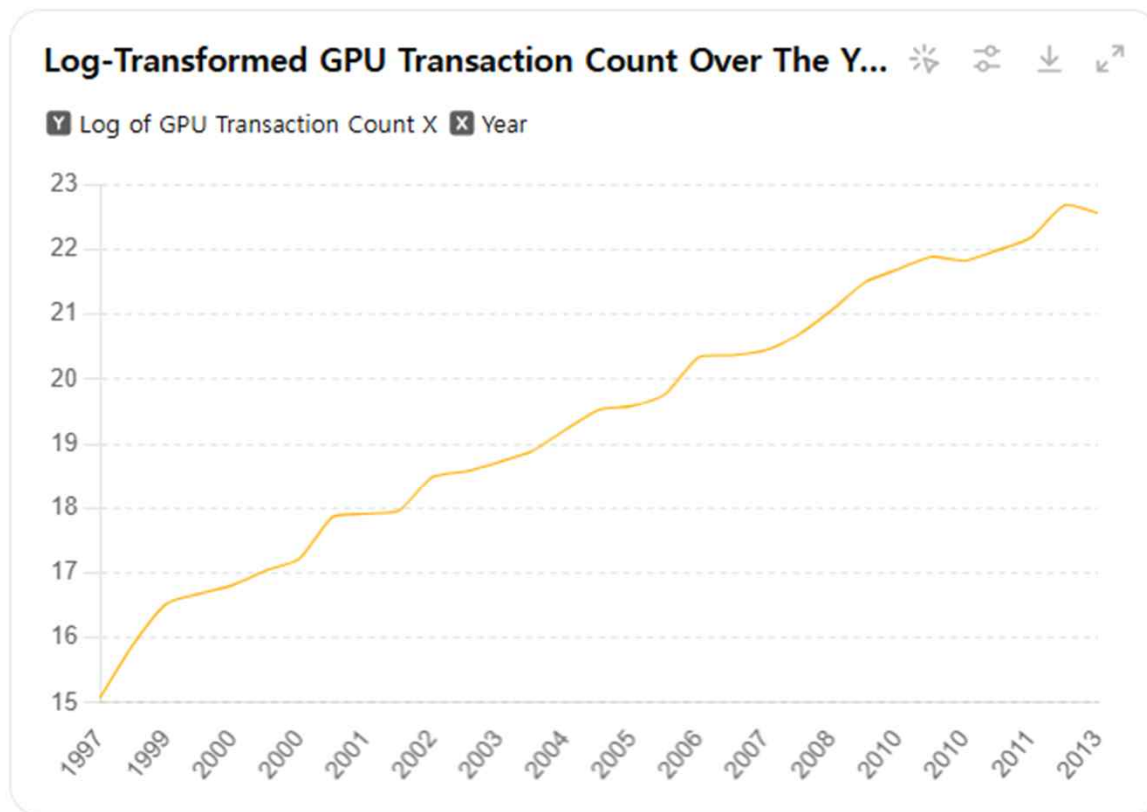


1997년~2005년 사이의
변화가 분명치 않음

Here is the graph showing the GPU transaction count over the years. If you need any specific modifications or additional information included in the graph, please let me know! [↗]

2. 레이블 인코딩과 로그변환

gpu_trans_count를 로그 변환한 뒤 그래프로 출력해줘.



1997년~2005년 사이의
변화가 분명하게 보임

Here is the graph showing the log-transformed GPU transaction count over the years. If you need any further modifications or additional analysis, feel free to let me know! [\[↗\]](#)

[4] 오버샘플링 기법

데이터 분석과 머신러닝에서 불균형한 데이터 세트를 다룰 때 사용되는 기법이다. 불균형한 데이터 세트는 특정 클래스의 데이터 포인트가 다른 클래스에 비해 현저히 적을 때 발생하며, 이는 모델이 소수 클래스에 대해 잘 학습하지 못하게 한다는 문제를 초래할 수 있다.

오버샘플링 기법에는 여러 가지가 있다. 대표적인 기법으로는 **SMOTE(Synthetic Minority Over-sampling Technique)**, **ADASYN(Adaptive Synthetic Sampling)**, 그리고 단순 복제 오버샘플링 등이 있다.

2. 레이블 인코딩과 로그변환

(1) SMOTE: SMOTE는 소수 클래스의 데이터 포인트 사이에서 새로운 데이터 포인트를 합성하여 데이터를 증강하는 방법이다. 이는 두 소수 클래스 데이터 포인트 사이의 벡터를 따라 새로운 포인트를 생성하여 데이터의 다양성을 증가시킨다.

(2) ADASYN: ADASYN은 SMOTE와 유사하지만, 소수 클래스의 샘플링이 어려운 영역에 더 많은 가중치를 두어 새로운 샘플을 생성하는 방법이다. 이로 인해 모델이 더 어려운 학습 예제를 잘 다룰 수 있도록 도와준다.

(3) 단순 복제 오버샘플링: 이 기법은 소수 클래스의 데이터를 단순히 복제하여 데이터의 양을 늘리는 방법이다. 가장 간단한 방법이지만, 데이터의 다양성이 증가하지 않기 때문에 과적합(overfitting)의 위험이 있다.

2. 레이블 인코딩과 로그변환

오버샘플링 기법을 사용할 때는 원래 데이터의 특성을 유지하면서 새로운 데이터를 생성하는 것이 중요하다. 또한, 오버샘플링으로 인해 발생할 수 있는 과적합 문제를 피하기 위해 모델의 성능을 주의 깊게 모니터링해야 한다.

이와 같은 오버샘플링 기법들은 불균형한 데이터 세트에서 모델의 성능을 개선하고 소수 클래스의 예측 정확도를 높이는 데 큰 도움이 된다.

2. 레이블 인코딩과 로그변환

언더 샘플링과 오버 샘플링 비교

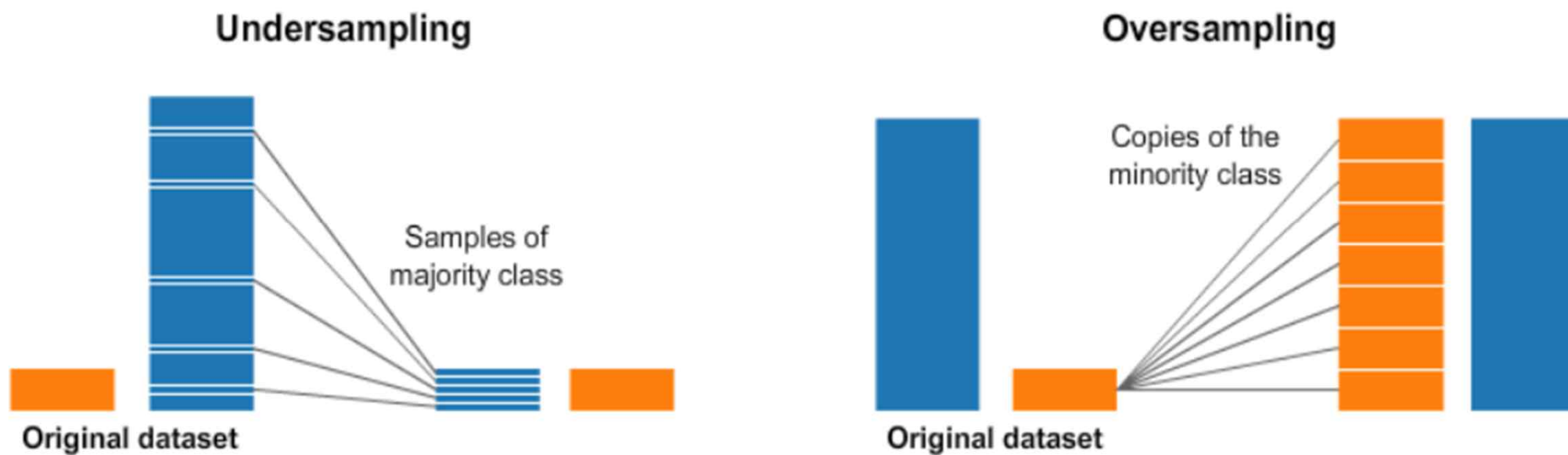


그림 출처 : <https://incodom.kr/SMOTE>

2. 레이블 인코딩과 로그변환

SMOTE(Synthetic Minority Over-sampling Technique)는 불균형한 데이터 세트에서 소수 클래스의 데이터 포인트를 증가시켜 데이터의 균형을 맞추기 위해 사용되는 기법이다. SMOTE는 단순히 소수 클래스의 데이터 포인트를 복제하는 것이 아니라, 기존 데이터 포인트를 기반으로 새로운 데이터 포인트를 합성하여 데이터의 다양성을 증가시킨다.

다음은 SMOTE의 작동 원리를 단계별로 설명한 것이다

(1) K-최근접 이웃(K-Nearest Neighbors) 선택:

각 소수 클래스 데이터 포인트에 대해 k 개의 가장 가까운 이웃을 선택한다. 일반적으로 유클리드 거리(Euclidean Distance)를 사용하여 거리를 계산한다.

2. 레이블 인코딩과 로그변환

(2) 새로운 데이터 포인트 생성:

- 선택된 k개의 이웃 중 무작위로 하나의 이웃을 선택한다.
- 선택된 이웃과 원래 데이터 포인트 사이의 벡터를 계산한다.
- 이 벡터를 따라 무작위로 임의의 점을 선택하여 새로운 데이터 포인트를 생성한다. 이는 다음과 같은 공식을 사용한다:

$$x_{new} = x_i + \delta \cdot (x_{nn} - x_i)$$

- 여기서 x_{new} 는 새로운 데이터 포인트, x_i 는 원래 데이터 포인트, x_{nn} 는 선택된 이웃, δ 는 0과 1 사이의 임의의 값이다.

2. 레이블 인코딩과 로그변환

(3) 반복:

- 이 과정을 소수 클래스의 각 데이터 포인트에 대해 반복하여 원하는 수만큼의 새로운 데이터 포인트를 생성한다.

이러한 과정을 통해 생성된 새로운 데이터 포인트는 원래 데이터 포인트와 선택된 이웃 사이의 임의의 위치에 존재하게 된다. 이는 소수 클래스 데이터의 분포를 더 잘 대표하게 만들어 데이터의 다양성을 증가시키고, 모델이 소수 클래스에 대해 더 잘 학습할 수 있도록 돕는다.

2. 레이블 인코딩과 로그변환

SMOTE의 장점은 데이터의 다양성을 증가시켜 과적합을 방지하고 모델의 일반화 성능을 향상시킬 수 있다는 점이다. 그러나 SMOTE를 사용할 때 주의해야 할 점도 있다. 예를 들어, 너무 많은 새로운 데이터 포인트를 생성하면 모델이 과적합될 수 있으며, 노이즈 데이터나 이상치(outlier)가 포함된 경우 부정적인 영향을 미칠 수 있다.

이러한 문제를 해결하기 위해 SMOTE와 다른 기법(예: ADASYN)을 함께 사용하거나, 데이터 전처리 과정에서 노이즈 데이터를 제거하는 등의 방법을 고려할 수 있다.

2. 레이블 인코딩과 로그변환

SMOTE 처리의 예시 그래프

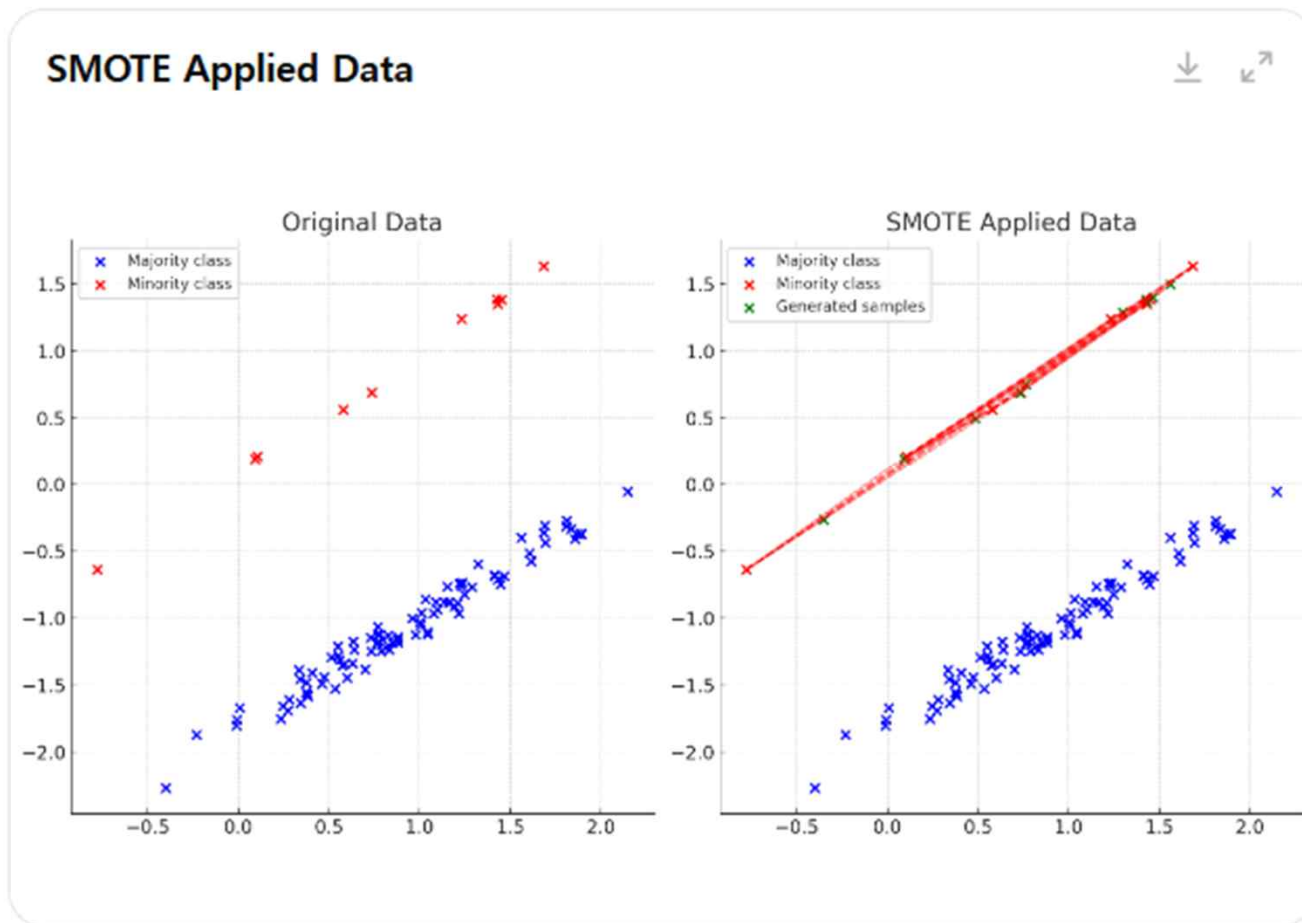


그림 설명

원본 데이터 (왼쪽 그래프)

- 파란색 점: 다수 클래스 샘플
- 빨간색 점: 소수 클래스 샘플

SMOTE 적용 후 (오른쪽 그래프)

- 파란색 점: 다수 클래스 샘플
- 빨간색 점: 원래 소수 클래스 샘플
- 초록색 점: 생성된 새로운 소수 클래스 샘플
- 빨간색 실선: 소수 클래스 샘플 간의 연결
- 빨간색 점선: 새로운 샘플 생성 경로 (소수 클래스 샘플 간의 연결을 기반으로 무작위 위치에 생성)

이 그래프는 SMOTE 기법을 통해 소수 클래스 샘플이 어떻게 생성되고, 전체 데이터셋의 클래스 균형이 맞춰지는지 시각적으로 보여줍니다. [2]

2. 레이블 인코딩과 로그변환



Kaggle신용카드사용데이터_SMOTE용....
스프레드시트

📎 데이터셋 설명해줘(한글로)



ChatGPT는 실수를 할 수 있습니다. 중요한 정보를 확인하세요.

데이터셋 설명

이 데이터셋은 신용카드 사용 데이터를 포함하고 있으며, 총 30개의 열(column)과 85443개의 행(row)으로 구성되어 있습니다. 각 열은 다양한 특성을 나타내며, 이 데이터셋은 특히 불균형한 클래스 문제를 해결하기 위해 SMOTE 기법을 적용하기에 적합합니다.

데이터셋 구조

- **특성(Feature) 열**
 - V1 ~ V28: 다양한 금융 트랜잭션의 피쳐들. 대부분의 피쳐는 주성분 분석(PCA)으로 변환된 값들입니다.
 - Amount: 거래 금액.
- **레이블(Label) 열**
 - label: 거래가 정상인지 부정인지 나타내는 레이블. (0: 정상 거래, 1: 부정 거래)

2. 레이블 인코딩과 로그변환

데이터 샘플

plaintext 코드 복사

	V1	V2	V3	V4	V5	V6	V7	\	
0	0.101337	0.665808	-0.514168	-0.675752	1.087999	-0.681790	0.877323		
1	-1.678212	1.244794	0.947874	-2.101099	-0.811015	-1.037444	0.144664		
2	-1.407998	-0.094725	1.218361	-1.884626	0.639017	-1.127801	0.537467		
3	2.025163	-1.001783	-0.855068	-0.506527	-0.793876	-0.197866	-0.779789		
4	0.477071	-1.295023	0.586617	1.599474	-1.215659	0.114858	-0.178181		
	V8	V9	V10	...	V21	V22	V23	V24	\
0	-0.096356	0.265313	-1.297842	...	0.114197	0.500593	-0.066669	0.558720	
1	0.465090	-1.137320	0.847260	...	-0.383548	-0.581462	0.033139	0.483823	
2	-0.295973	-1.670308	0.022252	...	0.295294	0.836930	-0.367286	0.157457	
3	0.032557	-0.357041	1.015046	...	-0.298900	-0.427423	0.298094	0.747735	
4	0.090062	0.944582	-0.207109	...	0.299761	0.114989	-0.461395	-0.122341	
	V25	V26	V27	V28	Amount	label			
0	-0.518068	0.367070	0.079707	0.131678	12.50	0			
1	-0.039289	0.078483	0.486383	0.291222	10.00	0			
2	0.840789	-0.160280	-0.210861	0.103780	29.95	0			
3	-0.426508	0.500501	-0.051892	-0.048338	50.27	0			
4	0.381628	-0.245921	-0.013775	0.092648	380.00	0			

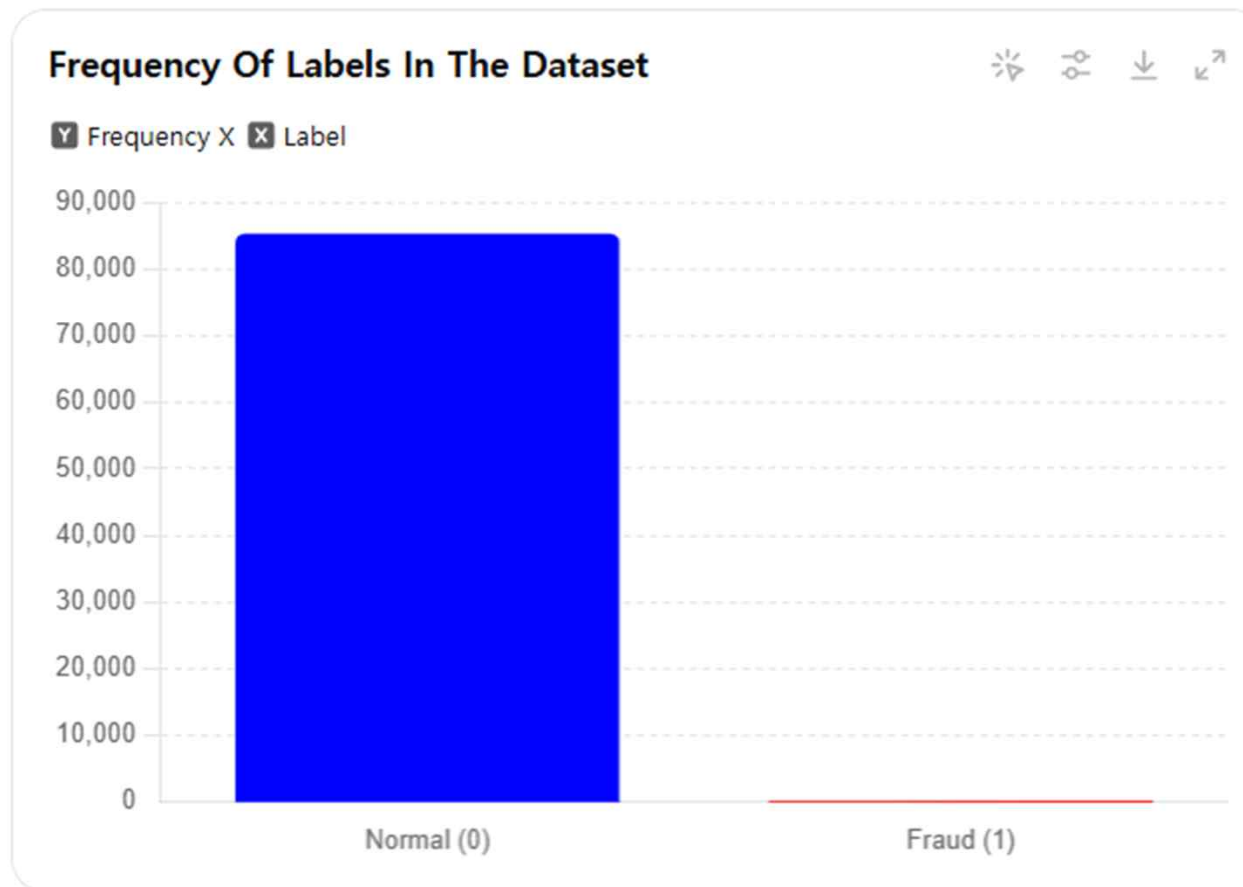
이 데이터셋을 사용하여 SMOTE 기법을 적용하면, 부정 거래(소수 클래스) 데이터를 증가시켜 학습 데이터의 균형을 맞출 수 있습니다. [↗]

2. 레이블 인코딩과 로그변환

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD
1	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15	V16	V17	V18	V19	V20	V21	V22	V23	V24	V25	V26	V27	V28	Amount	label
2	0.10134	0.66581	-0.5142	-0.6758	1.088	-0.6818	0.87732	-0.0964	0.26531	-1.2978	-1.0657	-0.0726	-0.091	-1.4738	-1.2434	0.21906	0.53565	0.04459	-0.8875	-0.1777	0.1142	0.50059	-0.0667	0.55872	-0.5181	0.36707	0.07971	0.13168	12.5	0
3	-1.6782	1.24479	0.94787	-2.1011	-0.811	-1.0374	0.14466	0.46509	-1.1373	0.84726	0.95105	0.6873	0.24206	0.16246	-1.0974	-0.5143	-0.8211	1.33428	-2.4073	-0.2464	-0.3835	-0.5815	0.03314	0.48382	-0.0393	0.07848	0.48638	0.29122	10	0
4	-1.408	-0.0947	1.21836	-1.8846	0.63902	-1.1278	0.53747	-0.296	-1.6703	0.02225	-0.1037	-0.369	0.84487	-0.3258	-0.1158	0.68573	0.27047	-2.0523	-0.517	-0.127	0.29529	0.83693	-0.3673	0.15746	0.84079	-0.1603	-0.2109	0.10378	29.95	0
5	2.02516	-1.0018	-0.8551	-0.5065	-0.7939	-0.1979	-0.7798	0.03256	-0.357	1.01505	0.54055	0.36855	-0.3192	0.21216	-0.434	-1.1238	-0.3687	1.56091	-0.5641	-0.519	-0.2989	-0.4274	0.29809	0.74774	-0.4265	0.5005	-0.0519	-0.0483	50.27	0
6	0.47707	-1.295	0.58662	1.59947	-1.2157	0.11486	-0.1782	0.09006	0.94458	-0.2071	-1.4199	-0.6525	-1.3226	0.10575	0.83718	0.55167	-0.501	0.63231	-0.4069	0.54349	0.29976	0.11499	-0.4614	-0.1223	0.38163	-0.2459	-0.0138	0.09265	380	0
7	1.90241	-0.0192	-2.6276	0.47311	1.02974	-0.2177	0.3132	-0.1203	0.63075	-0.9075	0.21951	0.59434	0.27048	-1.9074	-0.7806	0.54941	0.86699	1.21579	0.62735	0.04695	-0.1477	-0.3178	-0.1862	-1.5982	0.30294	-0.2332	0.00038	-0.0182	91.7	0
8	-2.892	-1.3903	-1.0614	-0.2802	0.72199	-1.7413	0.82133	-0.2183	-0.1454	-0.2065	-0.9363	0.35997	0.83258	0.41059	0.16737	0.47875	-0.5907	-0.5297	-0.2803	-0.5548	-0.0415	0.1375	-0.6874	0.07492	-0.9338	0.728	0.10421	0.19407	211.57	0
9	1.93979	-0.0546	-1.7793	1.24216	0.46873	-0.6619	0.47391	-0.1709	0.19963	0.45746	0.16371	0.06596	-1.8635	1.03301	-0.1074	-0.3685	-0.4055	0.14294	0.1891	-0.3097	0.0832	0.27834	-0.0669	-0.4314	0.40544	-0.485	-0.0359	-0.0728	43.77	0
10	1.0595	0.18837	0.34041	1.23448	-0.0866	-0.3475	0.23918	-0.1581	-0.238	-0.0516	0.04293	1.00354	1.4366	0.092	1.0435	-0.1998	-0.2726	-0.6768	-0.8646	0.05627	0.15554	0.44615	-0.1566	0.1325	0.62338	-0.2634	0.03094	0.03366	72.8	0
11	0.12715	0.81394	-0.5423	-1.1746	1.70592	-1.2908	2.00408	-0.9218	-0.1284	-0.1296	-0.8336	-0.4827	-0.5294	0.33471	-0.0425	-0.7843	-0.6392	-0.361	-0.0561	-0.0919	0.19241	0.91072	-0.3174	0.78564	-0.4247	-0.031	-0.304	-0.2414	3.9	0
12	-1.1069	1.19384	0.70769	-0.0975	-0.0687	0.11121	0.13464	0.01805	0.02969	1.25954	1.59889	0.84468	0.71641	0.00093	1.427	-0.5629	0.11965	-0.1333	1.7809	0.45898	-0.0139	0.1865	-0.0197	-0.2138	-0.5251	1.14487	-0.0531	0.30076	15.37	0
13	-2.4604	-2.1724	4.18781	5.82027	2.18205	0.1613	-2.7943	0.58198	1.22349	0.95848	-0.9713	-2.5432	2.04327	0.03249	-1.6882	-0.6218	1.3927	0.74927	1.39577	1.02837	0.15004	0.47416	0.14961	0.03049	0.18949	0.58796	-0.0958	-0.0595	0.06	0
14	-0.2484	1.09869	0.67251	-0.0595	1.03749	-0.2806	1.04363	-0.1308	-0.7965	-0.6993	1.60486	0.00115	-0.5305	-1.0563	0.36458	0.28552	0.65728	0.47892	-0.0964	0.02147	-0.0567	0.01004	-0.3137	-0.4142	0.00524	0.33249	-0.1344	-0.1634	0.76	0
15	2.0408	-0.4158	-0.5874	0.21709	-0.5975	-0.5388	-0.5973	0.05307	1.68448	-0.1628	-1.6323	-0.8137	-1.8976	0.45261	1.33968	0.50016	-0.7628	0.47063	0.1013	-0.3608	-0.1835	-0.4861	0.30213	-0.7012	-0.3551	-0.8806	0.04494	-0.0377	1	0
16	-0.7433	-0.8116	-0.1135	-2.2544	0.5221	-0.9827	0.51772	-0.0699	-1.2617	0.31109	0.37132	-0.5718	-0.3271	0.15354	-1.0279	1.37686	-0.521	-0.5113	-0.0006	0.051635	0.68312	1.57774	0.12746	-0.4432	-0.2627	-0.1248	0.33963	0.26722	138.65	0
17	-0.4121	1.13397	1.58572	-0.0443	0.02841	-0.8604	0.79837	-0.1364	-0.5804	-0.35	-0.1958	0.59447	1.24774	-0.1	0.71494	0.14323	-0.4645	-0.4994	0.0067	0.17849	-0.2134	-0.4958	-0.0329	0.39378	-0.1517	0.06736	0.27265	0.12363	4.99	0
18	2.26707	-0.7683	-1.8087	-1.1431	-0.0454	-0.5984	-0.3866	-0.2731	-0.4384	0.87992	-1.3752	-1.1067	-0.0183	0.04619	0.46677	1.14526	-0.1261	-0.9755	0.71638	0.01913	0.38017	1.04282	-0.0753	0.08267	0.30625	0.07892	-0.0495	-0.065	25	0
19	1.32368	-0.7348	0.1104	-0.7649	-0.7896	-0.3427	-0.6212	-0.0007	-0.567	0.60932	-0.6972	-1.362	-1.0424	0.11629	1.15725	1.02287	0.427	-1.5021	0.34222	0.05638	0.15114	0.24485	-0.0983	-0.44	0.43223	-0.1469	0.0038	0.0171	52.49	0
20	2.09692	-1.0297	-1.7729	-1.1692	-0.1595	-0.433	-0.364	-0.2295	-0.7051	1.00579	0.123	-0.2256	0.44328	0.12721	-0.2638	1.43434	-0.5164	-0.4106	1.0695	0.18699	0.47881	1.19738	-0.2328	-1.0283	0.27747	0.11381	-0.0555	-0.0711	95.75	0
21	0.82757	-2.0092	-0.7364	-1.2842	0.63031	4.0324	-1.3034	0.97142	-0.04	0.30324	-0.568	-0.3723	0.10614	-0.7178	-0.4606	1.10163	0.19762	-1.0438	1.1924	0.77551	0.12798	-0.2727	-0.2891	1.04956	0.43637	-0.2606	0.00103	0.0699	304.45	0
22	1.99875	-0.4542	-2.2467	-0.9179	2.16164	3.32836	-0.6436	0.38068	0.02288	0.2583	0.36229	-0.0366	0.5348	1.27007	0.35887	-0.6579	-0.9295	-0.1112	-0.082	-0.4495	-1.4703	0.57612	0.65124	-0.6918	0.0394	-0.052	-0.0513	23.04	0	
23	1.21373	0.39932	0.30462	0.63115	-0.0694	-0.5721	0.09714	-0.1389	-0.1675	-0.3649	0.37602	1.00936	1.5021	-0.517	1.20372	0.12535	0.199	-0.907	-0.5655	-0.0291	-0.245	-0.6095	0.14313	0.06941	0.18706	0.11976	0.00185	0.03023	1.98	0
24	1.98141	-0.0657	-0.3894	1.17327	-0.17	0.03145	-0.4978	0.01813	0.7774	0.1883	-0.8714	0.61904	0.97697	-0.1794	0.78592	0.42676	-0.8511	0.23167	-0.7775	-0.182	0.22191	0.82038	0.16038	0.60822	-0.038	-0.5869	0.06158	-0.0239	0	0
25	-4.789	1.29963	-0.8924	-0.5808	-1.959	2.07153	-3.6557	-7.9016	-1.247	-0.9987	0.00257	2.06013	0.62788	1.21007	-0.5781	-1.0619	0.73431	1.43042	0.40028	-1.4398	-0.628	-2.0569	-2.668	-0.3549	-0.4549	-0.8142	0.98144	-0.2512	568.81	0
26	0.11061	0.77439	0.34901	-0.8149	0.82205	-0.5533	1.18755	-0.3114	-0.2709	-0.0612	0.72654	1.09581	0.63487	-0.0931	-1.1463	0.07359	-0.9299	-0.2816	0.32242	0.12337	-0.2679	-0.4198	-0.0176	-0.3194	-0.471	0.10905	0.08304	-0.1368	9.88	0
27	1.2384	0.25856	0.08011	1.18542	0.32742	0.30544	0.08771	2.678849	0.24634	-0.1399	-1.1698	0.61561	0.69492	-0.1214	-0.1091	-0.5121	-0.0798	-0.7905	-0.0542	-0.1351	-0.1535	-0.15	-0.1679	-0.7402	0.8005	-0.2617	0.04472	0.00758	1	0
28	0.30749	0.60928	0.71024	0.55427	0.19813	-0.0316	0.35092	-0.0469	0.19913	-0.2667	-1.6039	0.30613	0.76285	-0.4172	-0.4407	-0.4872	-0.087	-0.121	1.1691	-0.0034	0.04026	0.46306	-0.0487	-0.3638	-0.9052	0.35125	0.19359	0.16084	1	0
29	1.21928	-2.3637	-0.7983	0.01428	-1.6974	-0.4712	-0.3293	-0.2026	0.03636	0.59523	-1.1931	-0.0692	0.25393	-0.1081	0.48019	-0.9458	-0.2792	1.09487	-0.9272	0.34102	-0.3086	-1.3753	0.0784	-0.1426	-0.9095	0.23343	-0.0987	0.04131	475	0
30	1.05507	-0.3563	0.92947	0.6722	-0.6036	0.8249	-0.7847	0.47926	0.69768	-0.1043	1.22192	0.75071	-0.1064	0.11852	-0.0148	-0.3629	0.25398	-0.4703	-0.4943	-0.2065	0.16162	0.66654	-0.0512	-0.2442	0.27865	0.55456	0.02649	0.00117	15.52	0
31	1.21367	0.23745	0.18747	0.73624	-0.4154	-1.3569	0.36599	-0.3412	-0.14	-0.032	-0.0073	0.34408	0.05505	0.42572	0.76225	-0.0954	-0.172	-0.5971	-0.2434	-0.068	-0.0208	-0.0784	-0.0416	0.75068	0.51937	0.34457	-0.0555	0.01361	28.61	0
32	1.99879	-1.0008	-1.4346	-0.9354	-0.1113	0.96954	-1.3288	0.5203	0.07667	0.1631	1.1598	-0.8687	-1.4217	-1.551	0.61301	1.24605	1.81232	-0.9018	-0.3304	-0.1208	0.31602	0.98901	0.11091	-1.7722	-0.3658	0.02874	0.06272	-0.0409	24.99	0
33	1.18546	-0.3891	0.99019	-0.5545	-1.043	-0.3238	-0.7943	0.07717	3.24784	-1.4712	0.56116	-2.1951	0.61681	1.50841	0.89426	-0.7159	0.91933	0.13455	-0.0005	-0.2403	-0.1401	0.0567	0.00697	0.00563	0.36833	-0.6931	0.07503	0.02523	11.85	0
34	-0.9678	0.51654	0.21534	-1.168	-0.1971	-1.5243	0.60745	0.2984	0.6786	-1.3314	-1.1624	-0.2559	-1.7349	0.78077	-0.1257	-0.5627	0.02745	0.00082	-0.0938	-0.1028	0.15535	0.36542	-0.0166	0.37088	-0.1228	-0.7064	0.28636	0.13308	49.77	0
35	1.28685	1.20623	-2.058	1.25926	1.44623	-0.8435	0.63587	-0.1543	-0.5982	-1.3742	1.44462	-0.2767	-0.2058	-3.2146	0.47635	1.58108	2.04329	2.23215	-0.1073	0.00032	-0.2376	-0.623	-0.3629	-1.1674	0.97684	-0.2387	0.03339	0.07455	1	0
36	1.19967	-0.443	0.96708	0.62588	-1.2066	-0.2968	-0.7357	0.06226	-0.7238	0.80496	-0.2621	-0.2551	-0.2631	0.13835	1.35529	-1.4157	0.07794	1.06223	-1.8705	-0.565	-0.1817	-0.0								

2. 레이블 인코딩과 로그변환

label의 빈도를 그래프로 그려줘



2. 레이블 인코딩과 로그변환

위 그래프는 데이터셋의 레이블 빈도를 나타낸 것입니다.

- **파란색 막대:** 정상 거래 (Label = 0)
- **빨간색 막대:** 부정 거래 (Label = 1)

그래프에서 볼 수 있듯이, 정상 거래의 수가 부정 거래의 수에 비해 현저히 많아 불균형한 데이터셋임을 알 수 있습니다. 이 불균형을 해소하기 위해 SMOTE와 같은 오버샘플링 기법을 적용할 수 있습니다. [3-]

2. 레이블 인코딩과 로그변환

수치로 알려줘.



레이블 빈도 수치

- 정상 거래 (Label = 0): 85,295건
- 부정 거래 (Label = 1): 148건

이 데이터셋은 매우 불균형한 분포를 보이고 있으며, 정상 거래가 부정 거래에 비해 압도적으로 많은 것을 확인할 수 있습니다. [↗]

🔊 📄 ↺ 🔒 ⭐ ✓

2. 레이블 인코딩과 로그변환

label에 SMOTE를 적용한 새로운 데이터를 만들어줘.



It appears that the ``imblearn`` library is not installed in the current environment. Let me proceed with an alternative approach to balance the data using SMOTE.

Please hold on while I address this. [\[>-\]](#)

It seems the installation process took too long, and I had to stop it. I will use an alternative approach without requiring additional installations.

Let's manually handle the synthetic generation for the minority class to balance the dataset. Please hold on. [\[>-\]](#)

2. 레이블 인코딩과 로그변환

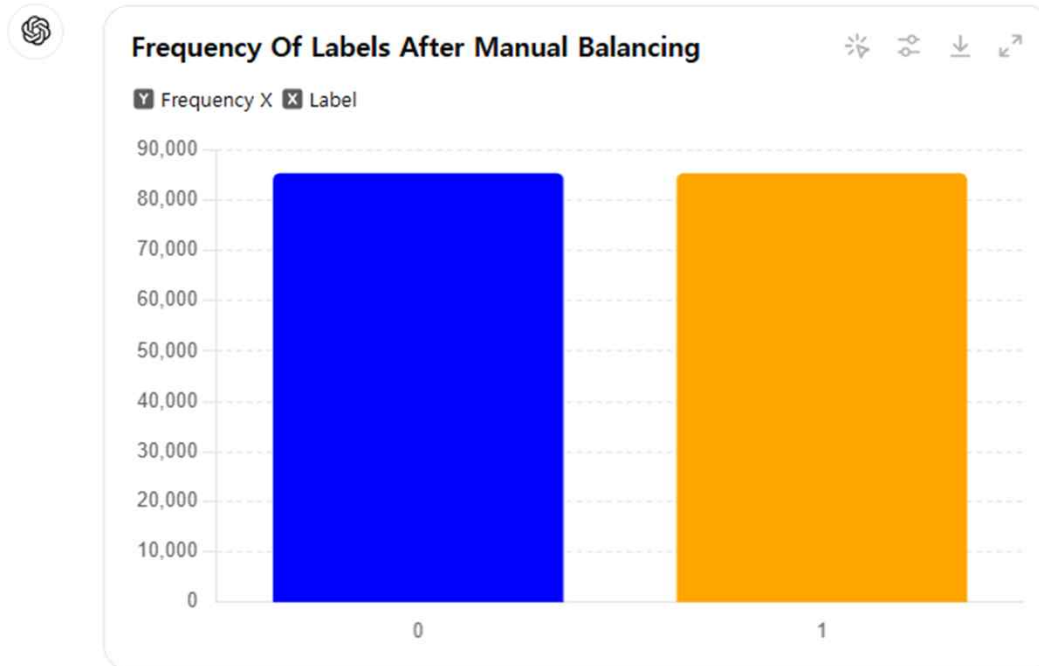
Manually Balanced Data With Synthetic Samples				
	V1	V2	V3	V4
7	-1.32697552264756	-1.01977101911383	-1.40601101079457	-1.0
8	0.27563902791278	0.65377089472757	0.80136107374961	0.00
	8	4	4	94
9	-1.16920341321234	1.86341353532436	-2.51513548140496	5.46
10	-3.84300903438536	3.37511020399082	-5.49289319668564	6.13
11	-0.05333867739605	-0.50441269435481	-0.75602148494091	-3.4

The data has been balanced by manually generating synthetic samples for the minority class.
You can download the new dataset from the link below:

[Download the balanced dataset \[↗\]](#)

2. 레이블 인코딩과 로그변환

SMOTE적용 된 label의 빈도를 수치로 알려주고 그래프로 그려줘.



The frequency counts for the labels after applying manual balancing are as follows:

- Label 0: 85,295 instances
- Label 1: 85,295 instances

The bar chart above visualizes the balanced frequency of each label. [\[↗\]](#)

2. 레이블 인코딩과 로그변환

결과 파일 다운로드 : Kaggle신용카드사용데이터_ManuallyBalanced.csv

170570	2.01258	0.18473	-1.6433	0.76215	0.23839	-1.0116	0.11739	-0.2736	0.63529	-0.7098	-0.0978	0.71758	0.78892	-1.8922	-0.5178	-0.0123	1.29183	-0.031	-0.255	-0.0962	-0.0012	0.32002	0.10414	1.10449	0.09406	0.57629	-0.039	-0.0168	6.99	0
170571	-3.7657	5.89074	-10.202	10.259	-5.6114	-3.2354	-10.633	3.27272	-5.2689	-11.182	8.87948	-18.431	-0.2328	-15.022	0.14119	-12.186	-20.166	-7.0517	2.50083	1.19414	2.24561	0.54632	0.38185	0.38202	-0.821	0.39436	1.41296	0.78241	0.01	1
170572	1.99368	-0.0713	-3.0662	-0.3841	2.74129	3.06506	-0.4384	0.63742	1.72175	-0.8382	1.35864	-2.4453	1.44956	0.91572	0.5872	0.10661	1.04473	0.6615	-0.7687	-0.1168	0.05926	0.4114	0.00669	0.57722	0.25613	-0.122	-0.0153	-0.0394	31.1	0
170573	1.94397	-0.408	0.18078	0.37633	-0.729	0.06472	-1.0039	0.04551	2.2665	-0.2538	1.67005	-0.9963	2.89753	1.07104	-0.7675	0.87114	-0.33	0.76348	-0.1145	-0.1163	-0.0694	0.20973	0.261	-0.3396	-0.5748	0.4388	-0.0372	-0.054	15.95	0
170574	-3.0361	-3.1481	1.79514	-0.9024	0.46427	-0.5172	-0.2145	-0.4586	0.3447	1.03311	0.57436	-0.3432	-0.3974	-1.5058	-1.624	1.2156	-0.2821	-0.8856	1.12362	-1.1616	-0.386	0.78318	1.48489	0.02062	0.01005	-0.4626	-0.4348	0.17958	60.35	0
170575	2.02491	-0.0164	-1.8148	0.25652	0.50613	-0.3629	-0.0079	0.01693	0.42762	-0.186	0.76947	0.14725	-1.1683	-0.4185	-0.2996	0.57921	0.26191	0.32226	0.36456	-0.2048	-0.3317	-0.9527	0.31298	0.16471	-0.3148	0.17706	-0.0728	-0.0449	8.99	0
170576	0.04552	0.57285	-2.7367	-1.413	3.93155	2.64524	0.75109	0.58225	-0.517	-0.7257	0.0233	-0.3015	-0.5583	-0.5119	-0.2799	-0.3397	0.60072	-0.2338	-0.3677	-0.1853	0.16393	0.50261	-0.182	0.68435	-0.4219	0.55789	-0.0102	0.09369	11.4	0
170577	-0.1716	-0.4364	1.72585	-2.5473	-0.6149	-0.3623	-0.0881	-0.1434	-2.4972	1.13915	1.46901	-0.289	0.40799	-0.4498	-0.6035	-0.4943	0.21652	-0.0154	-0.9673	-0.3143	-0.0503	0.37057	-0.0467	0.19299	-0.5132	-0.458	-0.0026	-0.0842	10	0
170578	1.82515	-1.5055	-0.6183	-0.6799	-0.777	0.95018	-1.1577	0.20013	-0.0953	0.85042	-0.1719	0.97005	1.80811	-0.5322	-0.0663	-0.3471	-1.1049	2.10711	-0.4118	-0.1099	-0.2625	-0.5094	0.14763	-0.3308	-0.6001	0.45754	-0.0208	-0.0173	168	0
170579	-1.2768	-0.9036	1.4771	-2.5826	0.63598	-0.8665	1.13807	-0.9246	-0.2215	0.09665	0.9826	0.73727	0.52146	-0.6259	-0.5642	-2.4449	-0.5336	1.57735	0.10092	-0.4311	-0.4821	-0.1825	-0.6141	0.03765	0.79308	-0.7247	-0.6165	-0.486	110.4	0
170580	0.45785	1.37377	-0.4889	2.80535	1.77739	0.10049	1.29502	-0.1359	-1.6958	0.955	-1.7022	-0.2401	0.45648	0.13957	-1.891	0.37238	-0.7982	-0.6971	-1.4972	-0.2647	0.10559	0.37101	0.0511	0.40152	-0.7248	-0.2029	0.09212	0.09496	0	1
170581	-4.2212	2.87112	-5.8887	6.89095	-3.4049	-1.1544	-7.7399	2.85136	-2.5076	-5.1107	5.35089	-9.2998	2.79314	-6.1066	-2.1069	-6.2506	-13.566	-4.1928	0.51057	-0.2279	1.62059	1.56795	-0.578	-0.059	-1.8292	-0.0724	0.13673	-0.5998	7.59	1
170582	-0.4198	-1.156	-2.0925	2.78675	0.7363	-0.1673	1.60003	-0.1174	-0.797	-0.134	1.03303	-0.2844	-1.2083	-0.4292	0.24895	-0.9289	1.67999	1.31534	2.22616	1.27536	0.48064	0.53352	1.28465	0.51613	-0.6029	-0.305	-0.0214	0.1291	451.27	1
170583	-5.1925	3.16472	-5.0477	2.2466	-4.0118	-0.6389	-2.8735	1.57632	-2.862	-2.1205	1.8636	-3.6203	-1.4807	-1.5833	-1.2305	-1.2023	-6.1676	-2.6515	0.01359	-1.8505	1.16724	-1.0066	0.77456	0.0634	-0.3907	1.88474	-1.7426	-0.0822	247.86	1
170584	-1.6629	3.25389	-7.0405	2.26646	-4.1776	-0.7469	-0.2483	1.09116	-0.3071	-5.5679	2.71274	-5.9484	2.14473	-4.2118	-0.1372	-2.6394	-2.5324	-0.6671	0.62606	-0.8422	0.45038	0.52116	0.30833	-0.318	-1.2554	-0.692	0.26488	-0.1304	600.73	1
170585	-0.3598	1.39831	-1.2307	-1.1094	1.14949	-1.3812	1.35844	-0.037	-0.7208	-0.7318	-0.9555	0.4926	0.75819	0.83451	-0.2745	-0.5485	-0.3262	-0.4106	-0.1162	-0.0914	0.29193	0.92494	-0.1826	0.81179	-0.229	0.05392	0.27298	0.20877	1.54	0
170586	-3.843	3.37511	-5.4929	6.13638	2.7972	-2.6462	-1.6689	-2.6176	-3.9458	-4.5653	4.09722	-5.4509	-0.9657	-10.904	0.52695	-1.1398	0.83564	0.38977	-1.4396	0.0548	-1.2778	0.71965	0.45113	-0.2581	0.65613	0.55668	0.73938	-0.203	1	1
170587	-1.1692	1.86341	-2.5151	5.46368	-0.298	1.36492	0.75922	-0.1189	-2.2939	-0.4238	2.37588	-3.2448	-0.5566	-5.1525	0.05091	-1.022	-1.6465	0.12646	1.81901	-0.2994	-0.3931	-0.7087	0.47131	-0.0786	-0.5447	0.01478	-0.2409	-0.7811	324.59	1
170588	0.91914	4.19963	-7.5356	7.42694	1.11822	-2.8867	-1.341	0.36393	-2.2032	-4.1378	4.57011	-7.6292	1.73392	-9.4404	-0.0234	-1.234	1.63201	1.31573	-0.2872	0.53543	0.31609	0.05518	0.21069	-0.4179	-0.9112	0.46652	0.62739	0.15785	1	1
170589	-12.34	4.48827	-16.587	10.1073	-10.42	0.13067	-15.6	-1.1577	-5.3046	-12.939	8.80568	-13.556	1.16546	-9.8099	0.36999	-9.5052	-17.542	-6.7926	2.06938	-0.0855	-2.0896	1.74531	1.37682	-0.5543	-1.6107	0.15373	1.21248	-1.8693	188.78	1
170590	-16.917	9.6699	-23.736	11.825	-9.8305	-2.5148	-17.291	1.82041	-6.2649	-12.917	9.56711	-13.717	0.89954	-13.273	-0.4023	-7.7541	-11.645	-4.7413	0.58463	0.99675	-2.3361	0.97275	1.24187	-1.0511	0.03801	0.67232	2.10847	-1.4212	1	1
170591	-3.843	3.37511	-5.4929	6.13638	2.7972	-2.6462	-1.6689	-2.6176	-3.9458	-4.5653	4.09722	-5.4509	-0.9657	-10.904	0.52695	-1.1398	0.83564	0.38977	-1.4396	0.0548	-1.2778	0.71965	0.45113	-0.2581	0.65613	0.55668	0.73938	-0.203	1	1

감사합니다