Categorical Data Analysis

[3주차 클린업 교안]

👏 범주형 자료분석팀 클린업이 또 돌아왔답니다! 👏

벌써 마지막,,? 한 번 더 (짝) 한 번 더 (짝)

오늘은 1주차에 배운 분할표와 유사한 **혼동행렬**과 이를 활용한 **분류평가지표,**

적합한 cut-off point를 찾아주는 **ROC곡선**, 마지막으로 **샘플링**, **인코딩**에 대해 배워보아요!

3주동안 패키지 클린업 세미나로 지친 여러분 너무 고생 많았어요😎

고지가 눈 앞이에요 빨리 끝내버립시다!

임지훈 010-3499-4655

목차

임지훈 010-3499-4655

Ⅰ. 혼동행렬 (Confusion Matrix)

1. 혼동행렬이란?

혼동행렬이란 모델의 성능을 평가하기 위한 지표로, 예측한 값($\hat{Y}$)과 실제값(Y)이 얼마나 정확히 일치하는지를 보여주는 행렬이다. 모델이 예측한 값과 데이터의 실제 값의 발생 빈도를 나열하는데, 이진 분류의 경우 실제값과 예측값을 참과 거짓으로 분류하여 다음 표와 같이 나타낸다.

		관측값(Y)	
		$Y=1$	$Y=0$
예측값($\hat{Y}$)	$\hat{Y}=1$	TP	FP
	$\hat{Y}=0$	FN	TN

혼동행렬에서는 총 4가지의 지표가 도출된다. 예측값이 관측값과 일치하였는지 여부에 따라 T(True)/F(False)로 구분되고, 예측값이 긍정 혹은 부정이었는지에 따라 P(Positive)/N(Negative)로 구분된다. 통계학원론에서 배웠던 내용을 적용해보면 FP는 1종 오류 (틀린 것을 맞다고 분류) , FN을 2종 오류 (맞은 것을 틀렸다고 분류) 라고 표현할 수 도 있다. 결국 4가지의 지표를 요약해보면 다음과 같다.

- ✓ TP (True Positive) : 긍정 ($\hat{Y}=1$)으로 예측하였고 실제 관측값도 긍정 (Y=1)인 경우
- ✓ TN (True Negative) : 부정 ($\hat{Y}=0$)으로 예측하였고 실제 관측값도 부정 (Y=0)인 경우
- ✓ FP (False Positive) : 긍정 ($\hat{Y}=1$)으로 예측하였으나 실제 관측값은 부정 (Y=0)인 경우
- ✓ FN (False Negative) : 부정 ($\hat{Y}=0$)으로 예측하였으나 실제 관측값은 긍정 (Y=1)인 경우

1) 혼동행렬의 한계점

① 정보의 손실

로지스틱 회귀모형은 예측 값($\hat{\pi}$)을 0과 1 사이의 연속적인 형태의 확률로 반환하지만, 이를 최종적으로 분류하기 위해 임의의 cut-off point에 따라 이항변수에 맞게 범주화 시켜야 한다. 결국 연속인 값($\hat{\pi}$)을 이항의 값($\hat{Y}$)으로 변환시키는 과정에서 숫자가 갖는 정보를 잃게 된다.

② 임의적인 cut-off point 설정

대부분의 경우 cut-off point를 0.5로 설정하여 분류를 진행하지만, cut-off point를 임의적으로 설정하는 것은 분석의 객관성을 떨어트린다. 어떤 값을 기준으로 분류하는지에 따라 혼동행렬이 크게 달라질 수 있는데, 특히 클래스 간 불균형이 심한 경우에 더 잘 드러난다. (이러한 문제를 해결하기 위한 방법으로 차후에 설명될 ROC 곡선이 등장한다!)

예를 들어, 예측 값($\hat{\pi}$)이 0.65인 경우 cut-off point를 0.5로 설정했을 때는 $\hat{Y}$=1로 분류되지만 0.7로 설정했을 때는 $\hat{Y}$=0으로 분류된다. 이처럼 cut-off point에 따라 어떤 값으로 분류되는지가 달라질 수 있으며 0.65라는 숫자 정보가 사라지는 문제점이 발생한다.

2. 분류평가지표

혼동행렬은 위와 같은 한계점을 가지고 있음에도 여전히 큰 해석력을 갖고 있다. 혼동행렬을 통해 구할 수 있는 다양한 분류평가지표에 대해 알아보고, 각 지표가 어떤 상황에서 적절한 쓰임새를 갖는지 알아보자.

1) 정확도 (Accuracy/ACC/정분류율)

정확도는 전체 혼동행렬 값에서 예측값과 실제값이 일치하는 비율이다.

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN}$$

		관측값(Y)	
		$Y = 1$	$Y = 0$
예측값($\hat{Y}$)	$\hat{Y} = 1$	**TP**	FP
	$\hat{Y} = 0$	FN	**TN**

두꺼운 테두리로 둘러싸인 부분이 분모, 색칠된 부분이 분자이다! 정확도는 직관적인 지표로 매우 자주 활용된다. 전체에서 정확히 예측한 비율을 나타내기 때문에 1에 가까울수록 성능이 좋은 모델이라고 판단한다. 하지만 관측치의 값이 한 클래스에 과하게 집중되어 있는 불균형 자료(imbalanced data)의 성능을 평가할 때는 한 클래스에 지나치게 의존하는 경향성을 띄게 되어 정확도가 큰 설명력을 갖지 못한다. (자세한 내용은 다음 샘플링 챕터에서!)

2) 정밀도 (Precision/PPV/Positive Predictive Value)

정밀도는 긍정 ($\hat{Y} = 1$)으로 예측한 값들 중에서 실제 관측치 역시 긍정 (Y=1)인 비율이다. 정확도와 마찬가지로 정밀도 역시 1에 가까울수록 성능이 좋은 모델이라고 판단한다.

$$Precision = \frac{TP}{TP + FP}$$

		관측값(Y)	
		$Y = 1$	$Y = 0$
예측값($\hat{Y}$)	$\hat{Y} = 1$	**TP**	FP
	$\hat{Y} = 0$	FN	TN

정밀도는 FP (False Positive, 실제로는 부정이지만 긍정으로 예측한 경우) 가 중요할 때 주로 사용한다. 예를 들어 재판의 경우를 생각해보자. 실제로 유죄인 사람($Y = 1$)을 무죄라고 선고하는 것($\hat{Y} = 0$), 즉 False Negative의 경우는 위험하지만 빈번히 발생하나, 실제로 무죄인 사람($Y = 0$)을 유죄로 선고($\hat{Y} = 1$)하는 것(FP)은 큰 문제를 야기한다. 이처럼 False Positive에 더 민감한 경우에 정밀도 지표를 활용한다.

3) 민감도 (=재현율)

민감도(Sensitivity/TPR/True Positive Rate) 또는 재현율(Recall)은 실제로 긍정($Y = 1$)인 관측값을 긍정으로 올바르게 예측($\hat{Y} = 1$)한 비율이다. 민감도 역시 올바르게 예측한 비율을 측정하는 척도이므로 1에 가까울수록 성능이 좋은 모델이라고 판단한다. 참고로 민감도는 차후 소개할 ROC 곡선의 Y축에 해당한다.

$$Sensitivity\ (Recall) = \frac{TP}{TP + FN}$$

		관측값(Y)	
		$Y = 1$	$Y = 0$
예측값($\hat{Y}$)	$\hat{Y} = 1$	**TP**	FP
	$\hat{Y} = 0$	FN	TN

민감도는 FN (False Negative, 실제로는 긍정이지만 부정이라고 예측한 경우)가 중요할 때 주로 사용된다. 예를 들어 코로나 검사를 진행하였을 때, 음성($Y = 0$)인 사람에게 양성으로 통보($\hat{Y} = 1$)하는 것, 즉 False Positive의 경우는 재검사를 통해 충분히 해결할 수 있지만, 확진자($Y = 1$)에게 음성으로 통보($\hat{Y} = 0$)하는 것(FN)은 매우 치명적이다. 이처럼 False Negative에 민감한 경우에 민감도 (=재현율) 지표를 활용한다.

4) 특이도 (Specificity/TNR/True Negative Rate)

특이도는 실제로 부정($Y = 0$)인 관측값을 부정으로 올바르게 예측($\hat{Y} = 0$)한 비율이다. 특이도 역시 올바르게 예측한 비율을 측정하는 척도이므로 1에 가까울수록 성능이 좋은 모델이라고 판단한다.

$$Specificity = \frac{TN}{TN + FP}$$

		관측값(Y)	
		$Y = 1$	$Y = 0$
예측값($\hat{Y}$)	$\hat{Y} = 1$	TP	FP
	$\hat{Y} = 0$	FN	**TN**

즉 특이도는 부정인 것을 잘 맞춘 정도를 나타내는 지표이다. 이와 반대로 실제로 부정인 관측값을 긍정으로 잘못 예측한 비율을 FPR (False Positive Rate)이라고 부른다. 식은 아래와 같다.

$$FPR = \frac{FP}{TN + FP} = 1 - Specificity$$

즉 FPR = 1-특이도로 계산 가능하다. 특이도는 잘못 예측한 비율을 나타내므로 위의 지표들과는 다르게 0에 가까울수록 성능이 좋은 모델이라고 판단한다. 또한 FPR은 ROC 곡선의 X축에 해당한다. (ROC 곡선의 X축 : FPR, Y축 : 민감도)

5) F1-Score

F1-Score는 정밀도(Precision)와 재현율(Recall)의 조화평균이다. 조화평균이란 주어진 수들의 역수의 산술평균의 역수를 의미한다 (*조화평균* $= \frac{2}{\frac{1}{a}+\frac{1}{b}} = \frac{2ab}{a+b}$). 조화평균을 활용하여 구한 F1-Score의 식은 아래와 같다.

$$F1\ score = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} = 2 \times \frac{Precision \times Recall}{Precision + Recall} = \frac{2TP}{2TP + FN + FP}$$

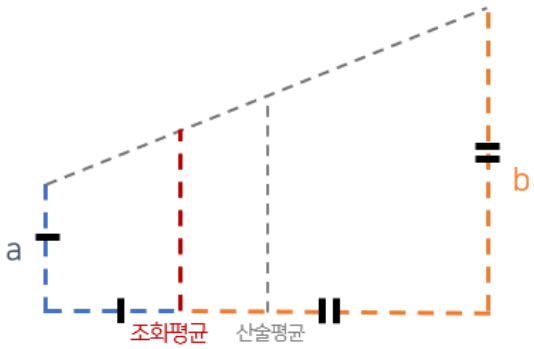
앞서 불균형한 데이터 (Imbalanced Data)인 경우 정확도 (Accuracy)가 적절하지 못하다고 설명했다. 이러한 단점을 보완하기 위한 지표로 정밀도와 재현율을 소개했는데, 이 둘의 조화평균을 통해 구한 F1-Score 역

시 데이터의 클래스 간 불균형을 보완할 수 있는 지표로 활용할 수 있다.

그렇다면 왜 산술평균이 아닌 조화평균으로 계산할까? 이는 정밀도와 재현율 간의 상충관계를 반영하기 위함으로, 다음의 그림을 통해 기하학적으로 접근해보자.

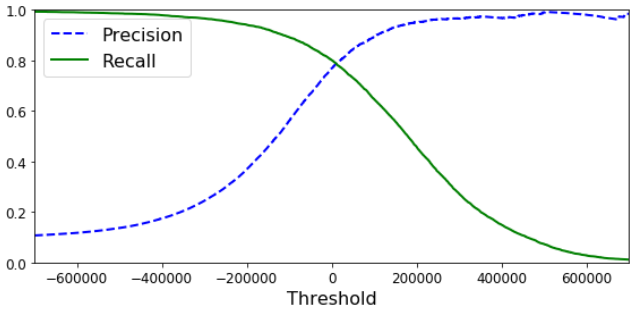
오른쪽 그림은 밑변이 a와 b인 사다리꼴이다. 이 때, 산술평균은 한 직선의 정확히 가운데 지점을 나누는 반면 조화평균은 a와 b 간의 크기의 차이를 반영하여 직선을 나눈다.

즉, 각 밑변의 길이와 동일한 거리에 떨어진 지점에서 빗변으로 그은 직선이 조화평균이 되는 것이다.



이처럼 조화평균은 더 큰 값에 페널티를 주어 작은 값에 가까운 평균을 구하게 되는 것이다. 따라서 불균형 데이터 (Imbalanced Data)가 주어졌을 때 더 많은 값을 갖고 있는 클래스에 페널티를 부과하여 두 클래스를 균형 있게 반영할 수 있다. 그래서 클래스 간 불균형이 존재할 때 정확도(Accuracy)의 한계를 보완하는 역할을 한다.

또한 앞서 F1-Score를 조화평균으로 계산하는 이유를 정밀도와 재현율 간의 상충관계를 반영하기 위함이라고 설명하였다. 상충관계(Trade-off)란 일정한 값을 양자가 나누어 가지는 관계로, 한쪽이 가진 몫이 커질수록 상대방의 몫이 줄어들게 된다. 정밀도와 재현율 모두 1에 가까운 값을 가질수록 모델이 더 좋은 성능을 보이지만 둘은 동시에 높은 값을 지닐 수 없다. 즉, 정밀도가 높아질수록 재현율은 낮아지고, 재현율이 높아질수록 정밀도는 낮아지는 양상을 띈다. 둘 간의 관계를 시각화하면 다음과 같다.



임계값(Threshold, cutoff-point)이 낮아질수록 예측값이 긍정으로 판별되기 더 쉬워진다. 예를 들어, 임계값이 0.6일 경우, 0.6 이상의 값들만 긍정으로 판별($\hat{Y}=1$)되지만, 임계값이 0.4로 낮아지면 0.4 이상의 값들이 긍정으로 판별($\hat{Y}=1$)되어 더 많은 값들이 긍정으로 판별될 것이다. 그에 따라서 재현율의 분모를 구성

하는 FN의 값이 줄어들어 재현도의 값이 커지게 될 것이다. 반면 FN의 값이 줄어듦에 따라 정밀도의 분모를 구성하는 FP의 값이 증가하여 정밀도의 값은 작아지는 결과가 발생한다.

위와 같이 정밀도나 재현율 중 한 지표만을 이용하여 성능을 평가하기보다 둘을 모두 고려한 F1-Score를 확인함으로써 더 좋은 모델 성능 지표를 찾고자 하는 것이다. F1-Score 역시 1에 가까울수록 모델의 성능이 좋다고 판단한다.

하지만 이런 F1-Score에도 한계점이 존재한다. 정밀도와 재현율만을 활용하기 때문에 TN (True Negative) 수치를 전혀 반영하지 못한다. 만약 실제로 부정($Y=0$)인 관측치를 올바르게 부정으로 예측($\hat{Y}=0$)한 값이 증감하더라도 F1-Score를 통해서는 이를 해석할 수 없다. 아래 예시를 살펴보자.

		관측값(Y)	
		$Y=1$	$Y=0$
예측값($\hat{Y}$)	$\hat{Y}=1$	26	27
	$\hat{Y}=0$	24	22

		관측값(Y)	
		$Y=1$	$Y=0$
예측값($\hat{Y}$)	$\hat{Y}=1$	26	27
	$\hat{Y}=0$	24	72

$$F1\ score = \frac{2*26}{2*26+27+24} = 0.505$$

위의 두 도표는 TN (True Negative) 수치에서 큰 차이를 보였지만 같은 F1-Score 값을 가진다. 이러한 한계점을 보완하기 위한 지표로 MCC를 이용할 수 있다.

6) MCC (Matthews Correlation Coefficient/매튜 상관계수/파이계수)

MCC는 혼동행렬의 모든 구성요소를 활용하여 값을 계산한다. 따라서 평가 지표들 중에서 가장 균형 잡힌 척도로 여겨진다. 식은 아래와 같다.

$$MCC = \frac{(TP \times TN) - (FP \times FN)}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$$

비율을 계산한 위의 지표들과는 달리 MCC는 상관계수 값을 의미하므로 -1 ~ 1 사이 범위 내의 값을 가진다. 1에 가까울수록 완전예측, 0에 가까울수록 랜덤예측 그리고 -1에 가까울수록 거꾸로 예측(역예측)을 의미한다.

MCC 역시 불균형 데이터 (Imbalanced Data)가 주어졌을 때도 높은 해석력을 가진다. 아래의 두 예시의 F1-Score와 MCC를 계산하여 결과를 비교해보자.

		관측값(Y)	
		$Y=1$	$Y=0$
예측값($\hat{Y}$)	$\hat{Y}=1$	92	4
	$\hat{Y}=0$	3	1

		관측값(Y)	
		$Y=1$	$Y=0$
예측값($\hat{Y}$)	$\hat{Y}=1$	1	3
	$\hat{Y}=0$	4	92

혼동행렬	F1-Score	MCC
왼쪽	$\frac{2\times 92}{2\times 92+4+3}=0.96$	$\frac{(92\times 1)-(4\times 3)}{\sqrt{(92+4)(92+3)(1+4)(1+3)}}=0.18$
오른쪽	$\frac{2\times 1}{2\times 1+3+4}=0.22$	$\frac{(1\times 92)-(3\times 4)}{\sqrt{(1+3)(1+4)(92+3)(92+4)}}=0.18$
비교	상이 (≠)	상동 (=)

양쪽 도표 모두 100개의 데이터를 기반으로 만들어진 혼동행렬이다. 왼쪽 도표는 TP (True Positive) 관측치가 매우 큰 불균형 데이터 (Imbalanced Data)이고, 오른쪽 도표는 TN (True Negative) 관측치가 매우 큰 불균형 데이터이다. 두 도표의 F1-Score는 약 0.74의 큰 차이를 보인 반면 MCC는 불균형과 무관하게 같은 수치의 MCC 값이 도출되었다. 모든 구성요소를 반영하는 MCC와 달리 F1-Score는 TN 값을 활용하지 않기 때문에 TN 값의 차이가 클수록 F1-Score도 더 큰 차이를 보인다. 그러므로 F1-Score 하나만을 활용하여 모델의 성능을 판단하는 것은 매우 위험하다!

그렇지만 MCC가 반드시 F1-Score보다 좋은 지표인 것은 아니다. 특정 클래스에 집중하는 경우가 아닌 모든 클래스에 대한 균형적인 평가가 목적이라면 MCC를 활용하는 것이 효과적이다. 하지만 대다수 실제 분류 예측의 경우 FN으로 예측했을 때와 FP로 예측했을 때의 Cost가 다르다. 따라서 비대칭 데이터를 더 효과적으로 평가하기 위해, 중요하지만 관측치가 적은 항목을 Positive로 설정하여 TP 수치를 중점적으로 F1-Score를 계산한다.

지금까지 총 6가지의 분류 평가지표에 대해 알아보았다. 각 평가지표가 어떻게 계산되는지를 이해하는 것도 중요하지만, 각 지표가 어떤 장단점을 갖고 어떤 데이터에서 더 유용하게 활용될 수 있는지를 정확히 파악하고 적재적소에 알맞은 지표를 선택할 수 있는 것이 더 중요하다!

II. ROC 곡선 (Receiver Operating Characteristic Curve)

1. ROC 곡선

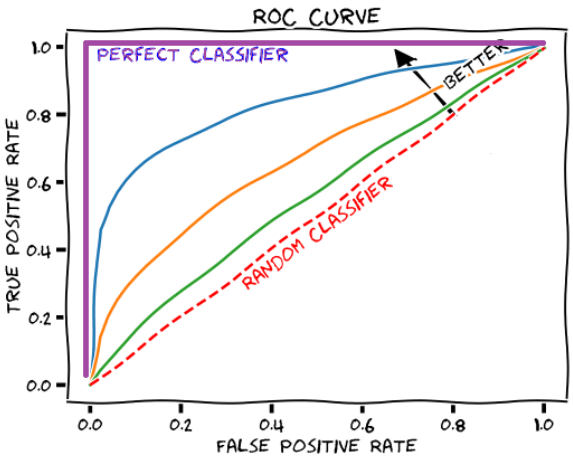
1) ROC 곡선이란?

지금까지 알아보았던 혼동행렬은 특정 cut-off point를 기준으로 관측값과 예측값을 분류하여 나열한 도표이다. 따라서 혼동행렬은 cut-off point가 변화함에 따라 검정력이 어떻게 변화하는지 파악하기 어렵다.

ROC 곡선은 위와 같은 혼동행렬의 한계점을 보완한다. ROC 곡선은 0~1 범위의 모든 cut-off point에 대하여 **재현율**과 **1-특이도** (FPR)의 함수로 나타낸 곡선 (혹은 직선) 그래프이다. 즉, 모든 cut-off point에 대하여 혼동행렬을 구하고 각 혼동행렬의 재현율과 FPR(1-특이도)를 2차원 상의 점으로 찍어 연결한 형태이다.

cut-off point에 지나치게 의존적이었던 혼동행렬의 한계점을 보완한 ROC 곡선은 ①혼동 행렬보다 더 많은 정보를 가지며 ②주어진 모형에서 가장 적합한 cut-off point를 찾을 수 있다는 이점을 지닌다.

2) ROC 곡선의 형태

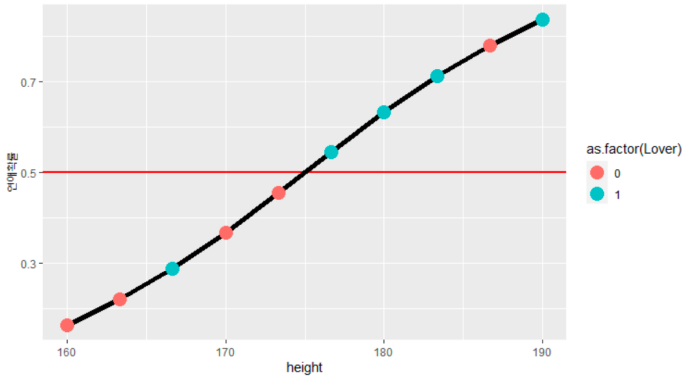


위의 그림은 다양한 형태의 ROC 곡선 (혹은 직선)의 형태를 나타낸다. ROC 곡선은 1-특이도 (FPR)을 X축, 재현율 (TPR)을 Y축으로 삼는다. 양 축 모두 0 ~ 1 사이의 값을 갖고, 그래프는 (0,0) 과 (1,1)을 잇는 우상향 그래프의 형태를 띈다. 그렇다면 왜 ROC 곡선은 (0,0) 과 (1,1)을 잇는 우상향 그래프의 형태를 띄는 것일까?

- ✓ Cut-off point가 0에 가까워짐 → 대부분을 $\hat{Y} = 1$로 예측! → TP & FP ↑ → TN & FN ↓ (한정된 숫자 안에서 긍정으로 예측한 값이 늘어나면 그만큼 부정으로 예측한 값이 줄어들게 되는 것이다!) → TPR & FPR 모두 1에 가까워짐!
- ✓ Cut-off point가 1에 가까워짐 → 대부분을 $\hat{Y} = 0$으로 예측! → TP & FP ↓ → TN & FN ↑ → TPR & FPR 모두 0에 가까워짐!

즉, Cut-off point가 0에 가까울수록 ROC 곡선이 (1,1)에, Cut-off point가 1에 가까울수록 ROC 곡선이 (0,0)에 가까워지기 때문에 이를 이은 우상향 곡선의 형태를 띄게 되는 것이다!

3) 최적의 Cut-off point 찾기



(설명변수 : 키 (cm) / 반응변수 : 연애여부 / 붉은 원, 푸른 원 : 관측치 / Y축 : 연애 확률)

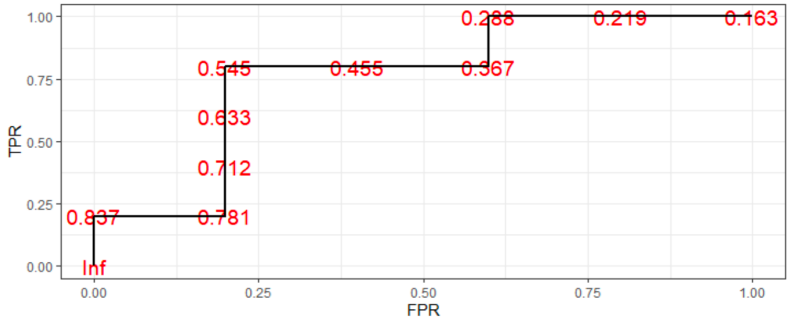
위의 그림은 키에 따른 연애 여부를 나타낸 로지스틱 회귀모형이다. 로지스틱 회귀모형의 식이 $\log\left[\frac{\pi(x)}{1-\pi(x)}\right] = -19 + 0.1x$ 일 때, 푸른 원은 실제로 연애중인 관측치, 붉은 원은 실제로 연애를 하지 않고 있는 관측치를 시각화한 것이다. 이 때 회귀식에 cut-off point를 0.5로 설정하여 0.5보다 큰 값을 $\hat{Y} = 1$, 작은 값을 $\hat{Y} = 0$ 로 예측한다면, 혼동행렬은 아래와 같고, 그에 따른 TPR, FPR 값을 구해보면,

		관측값(Y)	
		$Y = 1$	$Y = 0$
예측값($\hat{Y}$)	$\hat{Y} = 1$	4	1
	$\hat{Y} = 0$	1	4

$$\text{TPR} = \frac{TP}{TP+FN} = \frac{4}{4+1} = 0.8$$

$$\text{FPR} = \frac{FP}{FP+TN} = \frac{1}{1+4} = 0.2$$

위와 같이 모든 cut-off point에 대한 혼동행렬을 구하여 계산한 TPR, FPR 값을 바탕으로 ROC 곡선을 그려보면 아래와 같은 그림을 생성할 수 있다.



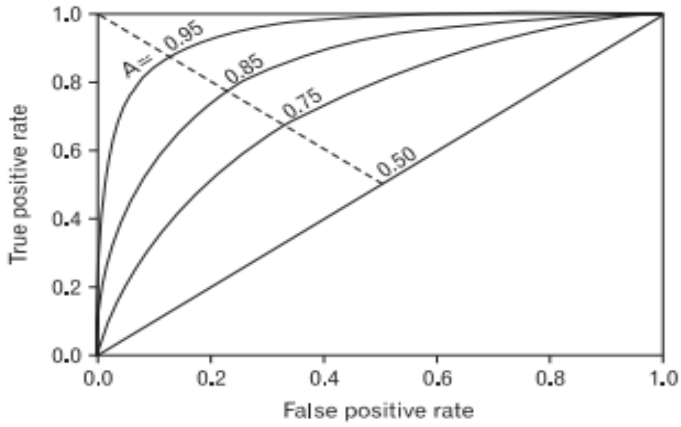
붉은 글씨는 TPR, FPR에 따른 cut-off point를 나타낸다. 위와 같이 cut-off point가 1에 가까울수록 (0,0)에, 0에 가까울수록 (1,1)에 향하는 것을 알 수 있다. 그렇다면 이 많은 cut-off point 중 무엇을 최적의 cut-off point로 선정하는 것이 합리적인가? 우리는 TPR이 1에 가까울수록 좋고 (실제 긍정 관측치를 긍정으로 예측했을 비율) FPR이 0에 가까울수록 좋다 (실제 부정 관측치를 긍정으로 잘못 예측했을 비율)는 것을 알고 있다. 이 사실을 근거로, 아래의 내용을 고려하여 최적의 cut-off point를 정하면 된다!

- ✓ Y값 (TPR) 이 같을 때, X값 (FPR) 이 더 작을수록,
- ✓ X값 (FPR) 이 같을 때, Y값 (TPR) 이 더 클수록 좋은 cut-off point이다!

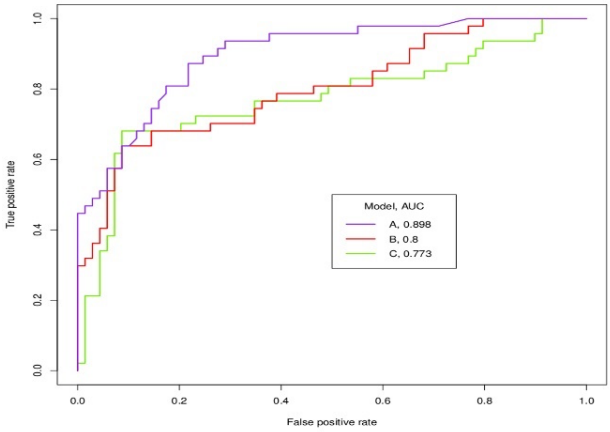
2. AUC 곡선

1) AUC란?

AUC (Area Under Curve)는 의미 그대로 ROC 곡선 아래의 면적을 의미한다. 0 ~1 범위의 X축과 Y축 내부의 면적을 구하는 것이므로 AUC 역시 0~1 사이의 값을 가진다.



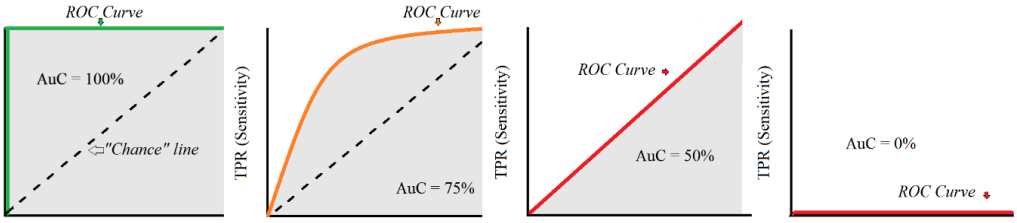
X값 (FPR)이 동일할 때, Y값 (TPR)이 클수록 더 좋은 ROC 곡선임을 배웠다. 따라서 위의 그림과 같이 ROC 곡선이 더 볼록할수록 좋은 성능을 의미한다. ROC 곡선이 볼록해지면 동시에 곡선 아래의 면적인 AUC 역시 더 큰 값을 가진다. 따라서 AUC 값이 클수록 모델 성능이 더 좋다고 판단한다. AUC는 모든 cut-off point를 고려하므로 특정 cut-off point와 상관없이 모델의 성능을 측정할 수 있다.



위의 그림은 총 3개 모델의 ROC 곡선을 시각화하여 나타낸다. 각 ROC 곡선 아래의 면적, 즉 AUC를 계산해 보았을 때, A 모델의 AUC가 0.898로 가장 높았다. 이는 A 모델의 성능이 비교적 가장 높다는 것을 의미한다.

2) AUC의 해석

이제 서로 다른 AUC 곡선의 의미에 대해 해석해보자.



① AUC = 1 : 모델이 100% 완벽하게 관측치를 예측했다는 의미이다. 이 경우에는 모델이 혹시 과적합(overfitting)된 것은 아닌지 확인해 보아야 한다.

② AUC = 0.75 : 모델이 75% 수준으로 관측치를 예측했다는 의미이다. 일반적으로 모델 AUC가 **0.8 이상** 일 때 성능이 좋다고 말한다.

③ AUC = 0.5 : 모델이 절반의 관측치를 예측했다는 의미이다. 이는 무작위로 예측한 것과 다름이 없다. 따라서 보통 AUC는 0.5 이상의 값을 보여야 정상이다.

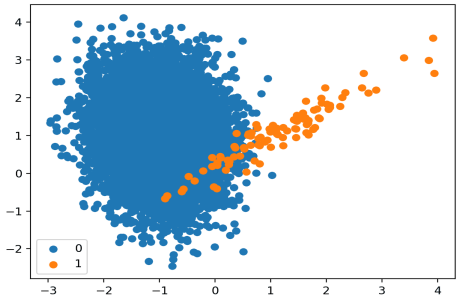
④ AUC = 0 : 모델이 관측치를 100% 반대로 예측했다는 의미이다.

AUC가 0.5보다 낮다는 것은 예측을 반대로 했다는 의미이다. 즉, Y=1과 Y=0을 거꾸로 예측한 것이다. 따라서 AUC가 0.5보다 낮을 때는 다시 확인해볼 필요가 있다. (시험문제를 완벽하게 다 틀리는 것은 전부 다 맞는 것 만큼 어렵기 때문…!)

III. 샘플링 (Sampling)

1. 클래스 불균형

데이터를 다루다 보면, 아래 그림과 같이 관측치의 개수가 크게 차이나는 경우를 접할 수 있다. 이를 클래스 불균형이라고 하는데, 범주형 자료에서 각 수준(클래스)의 관측치의 차이가 큰 경우를 의미한다. 실생활에서 클래스 불균형을 자주 접할 수 있는데, 예를 들어 혈액형을 RH-형과 RH+형 두 클래스로 구분할 때, 두 클래스 간 관측치에는 클래스 불균형이 발생할 것이다.



2. 샘플링 (Sampling)

1) 샘플링의 필요성

그렇다면 클래스 간 불균형을 조정해야 하는 이유가 무엇일까? 반응변수 Y의 클래스 간 차이가 클 경우에 모델의 성능을 정확히 파악하기 어렵기 때문이다. 예를 들어 (Y=1)인 관측치가 95개, (Y=0)인 관측치가 5개 있다고 하자. 설명변수들을 반영하여 모델이 예측한 값이 100개 전부 ($\hat{Y}$=1)을 도출했다. 이 경우 정확도를 계산하면 95%로 매우 높은 성능을 의미한다. 하지만 이 사실이 신뢰할 만 한가? (Y=1)인 관측치만을 고려하면 예측을 잘 한 것이 사실이지만 (Y=0)을 고려하면 전혀 예측을 해내지 못한 것이다. (이게 바로 정확도 지표의 한계점!) 아래의 예시를 통해 더 자세히 알아보자!

		Y	
		$Y=1$	$Y=0$
$\hat{Y}$	$\hat{Y}=1$	**60**	5
	$\hat{Y}=0$	40	**5**

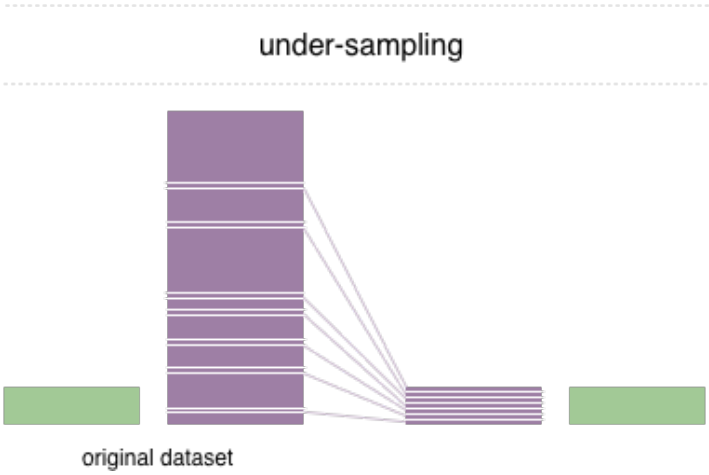
		Y	
		$Y=1$	$Y=0$
$\hat{Y}$	$\hat{Y}=1$	**50**	4
	$\hat{Y}=0$	50	**6**

위의 혼동행렬에서 왼쪽 혼동행렬은 Y=1 수준의 정확도가 0.6($\frac{60}{100}$), Y=0 수준의 정확도가 0.5($\frac{5}{10}$)이다. 반대로 오른쪽 혼동행렬은 Y=1 수준의 정확도가 0.5($\frac{50}{100}$), Y=0 수준의 정확도가 0.6($\frac{6}{10}$)이다. 이 때 각 혼동행렬의 전체 정확도를 계산해보면 왼쪽 혼동행렬은 0.59, 오른쪽 혼동행렬은 0.509가 나온다. 즉 두 혼동행렬 모두 각 수준의 정확도가 0.5, 0.6으로 동일함에도 불구하고 더 많은 데이터 값을 가진 클래스 정확도가 높은 혼동행렬이 더 높은 전체 정확도를 보이는 것이다. 이처럼 정확도 지표는 데이터 간 불균형의 정도에 큰 영향을 받는다.

따라서 소수의 클래스에 특별히 더 관심이 있는 경우에 샘플링을 통한 클래스 불균형 해소는 매우 필요하다!

2) 언더 샘플링 (Under Sampling)

언더 샘플링이란, 소수의 클래스는 변형하지 않고, 다수의 클래스를 소수의 클래스에 맞추어 관측치를 감소시키는 방법이다.



언더 샘플링은 데이터의 사이즈가 줄어들어 메모리 사용이나 처리 속도 측면에서 유리하다는 장점이 있지만 관측치의 손실이 일어나기 때문에 정보가 누락되는 문제가 발생한다.

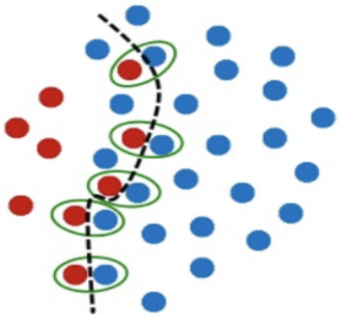
① 종류

다양한 언더 샘플링 기법 중 대표적인 Random Under Sampling 과 Tomek Links에 대해 알아보자.

✓ Random Under Sampling

이름 그대로 임의적으로 다수의 클래스의 데이터를 제거하여 관측치의 수를 줄이는 방법이다. 하지만 임의적으로 제거한 데이터의 정보가 누락되며, 추출된 샘플들이 기존 데이터에 대한 대표성을 띄지 못하면 부정확한 결과를 야기할 수 있다.

✓ Tomek Links



우선 임의로 서로 다른 클래스의 데이터를 두 점을 선택하여 이를 연결한다. 해당 거리가 주위에 있는 다른 데이터들과 연결한 거리보다 짧다면, 두 점 간 Tomek Link가 있다고 말한다. 즉, Tomek Link가 있는 점들을 위에선 초록색 동그라미로 강조해 놓은 것이다.

Tomek Link Method란 그림에서 초록색 동그라미로 묶인 데이터 쌍에서 다수의 클래스에 속한 데이터(파란색 점)들을 삭제함으로써 데이터들의 크기를 줄이는 것이다. 즉, 두 클래스 간 경계에 있는 노이즈 데이터를 제거하는 방식으로 언더 샘플링이 이루어지는 것이다.

Tomek link는 분포가 높은 클래스의 중심분포는 어느정도 유지하면서 경계선을 조정하기 때문에 무작위로 삭제하는 샘플링보다 정보의 유실을 크게 방지할 수 있지만 토멕링크로 묶이는 값이 한정적이기 때문에 큰 언더 샘플링의 효과를 얻을 수 없다는 단점이 있다.

각각 원리는 조금씩 다르지만, 결국 다수 클래스의 데이터를 줄이는 방법이다. 이 때 샘플링해서 얻은 임의의 표본이 대표성을 띠지 못하면, 부정확한 결과를 초래할 수 있게 된다. 언더 샘플링은 관측치를 삭제함으로써 정보를 누락시키는 방법이기 때문에 보통은 오버 샘플링을 사용한다.

3) 오버 샘플링 (Over Sampling)

오버 샘플링이란, 소수의 클래스의 데이터들을 다수의 클래스의 관측치 수에 맞추어 증가시키는 방법이다.

오버 샘플링은 정보의 손실이 없기 때문에 언더 샘플링에 비해 성능이 좋다. 반면 관측치 수가 늘어나기 때문에 메모리 사용이나 처리속도 측면에서 상대적으로 불리하다.

① 종류

다양한 오버 샘플링 기법 중 대표적인 Random Over Sampling 과 SMOTE에 대해 알아보자.

✓ Random Over Sampling

이름 그대로 무작위로 소수 클래스의 데이터를 복제하여 관측치의 수를 늘리는 것이다. 하지만, 기존의 데이터를 그대로 복제하는 방식이기 때문에 동일한 데이터의 수가 늘어나 과적합(overfitting)될 가능성이 크다. (즉, 같은 자리에 점을 계속해서 찍는 방식인 것이다.)

✓ SMOTE (Synethetic Minority Over-sampling Method)

그림에서 초록색 원은 Minority Point, 파란색 네모는 Majority Point,
빨간색 원은 SMOTE를 통해 새롭게 생성된 Synthesized Point(Sample) 이다.

SMOTE 기법은 소수 범주의 데이터를 단순 복제하는 Resampling 기법과는 다르게 소수 범주의 데이터를 가상으로 만들어낸다. 위 그림의 좌측과 같은 원시 Dataset이 있다고 할 때, SMOTE는 다음과 같은 절차로 이루어진다.

- 소수 범주의 데이터 중 무작위로 하나를 선택한다. 선택한 데이터를 X라고 하자.
- 무작위로 선택된 데이터(X)를 기준으로 k개의 가장 가까운 데이터를 선택한다. (이 때 k의 개수는 1보다 큰 값, KNN 알고리즘을 활용한다)
- 선택된 K-nearest neighbors 데이터 k들 중 하나를 무작위로 선정하고 데이터 X와 연결한 직선 상의 임의의 위치에 가상의 데이터를 생성한다.
- 위의 과정을 소수 범주 내 모든 데이터에 대하여 반복한다.

데이터를 단순하게 복사하는 Random Over Sampling과 달리 알고리즘에 기반하여 새로운 데이터를 생성하는 SMOTE는 과적합의 위협이 낮다. 하지만 SMOTE는 소수 범주의 데이터들 간의 거리만을 고려하여 새로운 데이터를 생성하기 때문에 다른 클래스의 데이터와 겹치거나 노이즈가 발생할 위험이 있다. 따라서 고차원의 데이터에서는 SMOTE 기법이 효율적이지 못하다.

위에 소개된 언더 샘플링, 오버 샘플링 기법들 외에도 다양한 기법들이 존재하는데, 이들에 대해서는 [부록] 부분에서 매~~우 간단하게만 소개해두겠다… (서비스야 주는 사람 맴이지…^^)

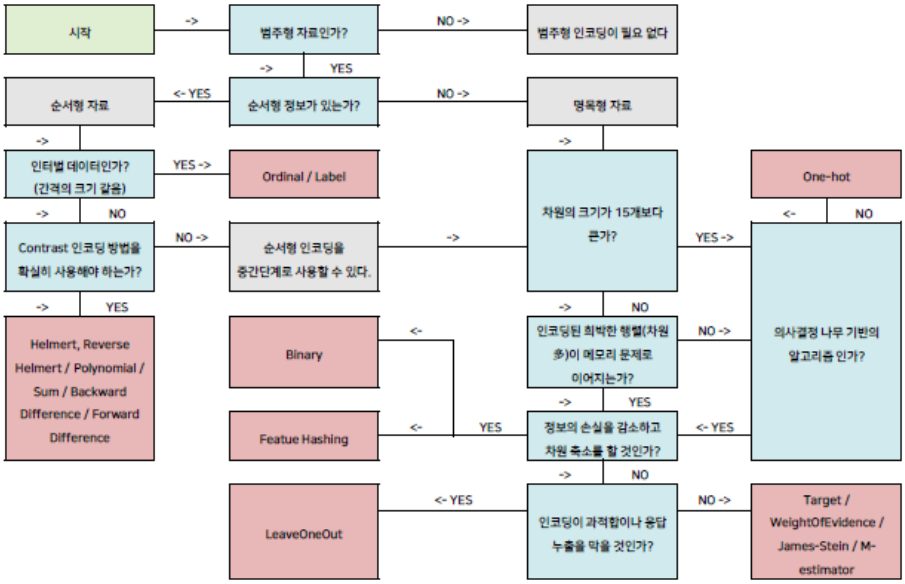
IV. 인코딩 (Encoding)

1. 인코딩 (Encoding)

데이터 분석에서 인코딩이란 사용자가 입력한 문자나 기호들을 컴퓨터가 이용할 수 있는 신호로 변환시키는 과정을 뜻한다. 특히 범주형 자료들을 분석할 때 대부분의 범주형 자료들은 문자나 기호들로 기록되어 있는 경우가 빈번하여 모델에 해당 자료들을 학습시키기 위해서는 인코딩의 과정이 필수적이다.

실제로 정형 데이터에서 수치형 변수만큼이나 범주형 변수들을 많이 마주치기에 인코딩의 다양한 방법에 대해 익히는 것이 중요하다. 인코딩 과정을 통해 수치형 변수들만을 설명 변수로 갖는 다양한 분석기법을 적용할 수 있다. (ex. 회귀분석 모델)

아래의 그림은 다양한 경우에 따라 알맞은 인코딩 기법들을 분류해둔 순서도이다.



2. 인코딩의 종류

아래의 표와 같이 인코딩 기법은 매우 다양하다. 이 중에서 우리는 가장 자주 활용되고 중요한 총 6가지 기법들에 대해서만 알아보도록 하자!

Classic	Contrast	Bayesian	기타
Ordinal	Simple	**Mean (Target)**	Frequency
One-Hot	Sum	**Leave One Out**	
Label	Helmert	Weight of Evidence	
Binary	Reverse Helmert	Probability Ratio	
BaseN	Forward Difference	James Stein	
Hashing	Backward Difference	M-estimator	
	Orthogonal Polynomial	**Ordered Target**	

1) Ordinal Encoding

Ordinal Encoding은 **순서형 자료**를 인코딩하는 기법으로, 1을 기준으로 순서에 따라 차등적인 점수를 부여하는 방식이다. 이 경우 각 수준에 부여된 점수들 간에 순서와 연관성이 존재한다.

범주형자료분석팀 클린업 만족도를 아래의 도표와 같이 Ordinal Encoding 해보면,

만족도	점수
매우 별로	1
별로	2
보통	3
좋음	4
매우 좋음	5

다음과 같이 1부터 시작하여 각 수준에 점수를 차등 부여할 수 있다. 이 때 각 점수는 순서와 연관성의 의미를 가진다. Ordinal Encoding 기법은 기존의 항목을 해당하는 숫자로 변환시키는 과정이므로 해당 차원 외에 추가적인 차원의 증가가 발생하지 않아 모델이 빠르게 데이터를 학습할 수 있다는 장점을 갖는다.

하지만 Ordinal Encoding에서 할당한 점수가 각 수준(매우 별로, 별로, 보통, 좋음, 매우 좋음) 간의 정확한 간격의 차이를 반영하기 어렵다는 한계점이 존재한다. 예를 들어 "매우 별로"와 "별로" 항목 간의 차이보다 "좋음"과 "매우 좋음" 간의 차이가 더 극명한 경우에도 Ordinal Encoding의 결과에 따르면 두 간격이 모두 1로 동일하게 나타나는 것이다. 따라서 이런 차이를 정확히 반영하기 위해선 해당 과제에 대한 도메인 지식의 중요성이 크다.

(Python의 Pandas를 사용할 때 사용자가 주어진 범주형 변수를 보고 직접 변수 값들 간의 순서(order)를 dictionary 형태로 정의해주어야 한다. 따라서 직관적이지만 추가적인 코딩을 해야 하는 수고가 필요하다…)

2) One-Hot Encoding

앞선 Ordinal Encoding이 순서형 변수를 처리하는 기법이었다면, One-Hot Encoding은 **명목형 변수**를 처리하는 기법이다. One-Hot Encoding은 가변수(Dummy Variable)을 생성함으로써 Encoding을 진행한다. (회귀분석입문 수업을 들었던 학생이라면 회귀분석 모델에 질적변수를 적합 시키기 위한 방법으로 접해봤을 것이다.) 아래 예시를 통해 자세히 알아보자.

뉴진스
하니
민지
다니엘
해린
혜인

→

하니	민지	다니엘	해린	혜인
1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1

왼쪽의 원시 데이터는 뉴진스 멤버들이 각 데이터의 값으로 입력되어 있었지만, 각 멤버들이 새로운 변수, 즉 가변수, 의 명칭으로 생성되었음을 알 수 있다. 열과 일치하는 값에는 1, 일치하지 않는 값에는 0을 부여하였다. 이진분류와 같이 설명변수 X 값들을 이용하여 각 관측치의 클래스를 분류하는 **분류 모델**의 경우에는 모든 정보를 이용하여 분류를 진행하기 때문에 J개의 가변수를 문제 없이 그대로 사용 가능하다.

하지만 **회귀 모형**의 경우 1개의 변수를 삭제하여 총 J-1개의 가변수를 생성해야 함을 유의하자. 이는 회귀분석에서 중요한 다중공선성 문제를 해결하기 위함으로, J-1개의 가변수들 만으로 J개의 수준을 모두 표현할 수 있기 때문이다(자유도를 반영한다고 표현한다). 즉, 새롭게 생성된 가변수들 중 1개의 변수를 삭제해도 데이터가 가진 자료는 그대로 유지된다. 예를 들어, "하니" 변수가 사라져도, 남아있는 모든 가변수가 0일 때 이것이 사라진 변수 (여기에서 "하니" 변수)를 의미한다고 표현 가능하기 때문이다. 즉, 회귀 모형에서 총 J개의 수준을 갖는 범주형 변수를 One-Hot Encoding 할 때는 J-1개의 가변수로 Encoding 해야 한다. (이렇게 한 개의 가변수를 줄여주는 One-Hot Encoding을 Dummy Encoding이라고 부른다.)

One-Hot Encoding은 아래와 같은 장점을 지닌다.

① 해석이 용이하다. 왜냐하면 기준 범주에 대한 정보가 intercept로 존재하기 때문에 기준 범주를 기준으로

모델을 해석할 수 있는 것이다.

② 명목형 변수 값을 가장 잘 반영하는 방법이다.

③ 해당 수준에 속하는 경우만 1, 나머지는 0으로써 표현되기 때문에 한 변수가 다른 변수들로 설명되는 다중공선성 문제를 해결할 수 있다. (다중공선성(Multicollinearity)는 회귀팀 클린업 참고!)

이러한 이점들로 인하여 One-Hot Encoding은 실제 데이터 분석에서 활발히 활용된다. 하지만 데이터 안에 범주형 변수가 많거나 변수의 수준이 지나치게 많은 경우, One-Hot Encoding은 너무 많은 가변수를 생성시켜 데이터의 차원이 과다해지는 문제점을 초래할 수 있다. 이는 모델의 학습 속도를 늦출 뿐 아니라 많은 Computing Power를 요구한다.

3) Label Encoding

Label Encoding은 **명목형 변수**에 사용하는 또 다른 기법으로 각 수준에 점수를 할당하는 방법이다. 앞에서 살펴보았던 Ordinal Encoding과 같은 원리지만, 그와 달리 Label Encoding에서 각 수준에 부여한 숫자들 사이에는 어떠한 의미나 연관성이 존재하지 않는다.

마찬가지로 뉴진스 예시를 통해 알아보자.

뉴진스	점수
하니	1
민지	2
다니엘	3
해린	4
혜인	5

각 멤버에 부여된 점수 (1~5)는 분석자가 임의로 부여한 점수로, Ordinal Encoding과 달리 1부터 시작할 필요가 없으며, 각 점수 간 간격이 일정하지 않아도 된다. Label Encoding은 명목형 변수를 Encoding하는 것으로 항목들 간에 구분만 되면 충분하다. One-Hot Encoding과 달리 가변수를 생성하지 않아 차원이 증가하지 않아 처리 속도가 빠른 장점이 있지만 모델 학습 과정에서 Ordinal Encoding으로 인식하여 할당한 점수들 간에 순서나 연관성이 있다고 잘못 판단할 위험이 있다.

지금까지 살펴본 3가지의 Encoding 기법들은 범주형 변수의 수준들을 구분하는 것에 초점을 맞춘 방법으로 할당된 값이 특별한 의미를 지니진 않는다. 하지만 다음으로 소개할 3가지 기법들은 범주형 변수의 수준들을 구별할 뿐 아니라 해당 변수와 반응변수 간의 수치적인 관계를 반영하여 숫자를 할당하는 Encoding 기법들이다.

4) Mean Encoding (Target Encoding)

Mean Encoding은 범주형 변수의 각 수준에서 도출된 반응변수의 평균을 모든 수준에 동일하게 할당하는 방식이다.

아래 예시를 통해 살펴보자. 범주형 설명변수 (학과) 와 반응변수 (키) 간의 관계를 나타낸 도표이다. 이 때 Mean Encoding 기법을 사용한다면 각 학과의 수준에 어떤 숫자를 부여해야 할까?

[Y] 키(cm)	[X] 학과	[X] Mean Encoding
168	경영	172
180	경영	172
168	경영	172
174	통계	166
156	통계	166
163	통계	166
171	통계	166
165	경제	171.66
180	경제	171.66
170	경제	171.66

학과 변수에는 총 3개의 수준 (경영, 통계, 경제)가 있다. 이 때 각 학과별 반응변수들의 평균을 계산해보면 경영학과는 172, 통계학과는 166, 경제학과는 약 171.66이 구해진다. 이 값이 기존 데이터의 문자를 대체하는 값으로 사용된다.

(1) 장점

① One-Hot Encoding과 달리 차원이 증가하지 않아 학습 속도가 빠르다.

② 앞선 Encoding 기법들과 달리 해당변수와 반응변수 간의 관계를 고려하여 점수를 할당하였다는 점에서 당위성을 갖는다.

(2) 한계점

① 이상치에 취약하다. 지나치게 크거나 작은 값이 존재할 때, 이 숫자 역시 반영하여 평균을 계산하기 때문에 정보의 왜곡이 발생할 수 있다.

② 설명변수를 처리하기 위해 반응변수 값을 활용하기 때문에 모델 학습 시 과적합이 발생할 위험이 있다.

③ Train set에 없던 수준이 Test set에 등장하면 활용할 수 없다. 예를 들어 문헌정보학과라는 새로운 X의 수준이 Test set에 있다면 점수 할당이 어렵다.

④ 관측치 값이 적은 범주의 경우 모델링에 부정확한 결과가 도출될 가능성이 있다. (ex. 학습 데이터에서 5개만으로 진행한 Encoding 값이 테스트 데이터에서 관측된 50개의 데이터를 대표한다고 하기 어렵다.)

이런 문제들을 해결하기 위해 Smoothing, CV loop, Expanding Mean과 같은 기법들이 등장하여 문제점을 해결할 수 있게 되었다. 이 부분 설명은 생략할게요. 궁금하신 분들은 구글링 혹은 저를 찾아오세요!

5) Leave-One-Out Encoding (LOO Encoding)

Leave-One-Out Encoding은 이상치에 취약한 Mean Encoding의 한계점을 개선한 Encoding 기법이다. 이는 Mean Encoding과 마찬가지로 수준별 반응변수의 평균을 계산하지만, 현재 행을 제외하고 나머지 행들의 평균을 계산한다는 점에서 차이가 있다. 만약 해당 값이 이상치에 해당했다면, 이 값을 제외하고 평균을 계산하기 때문에 영향력을 감소시킬 수 있는 것이다. 아래 예시를 살펴보자.

[Y] 키(cm)	[X] 학과	[X] LOO Encoding
168	경영	174
180	경영	168
168	경영	174
174	통계	163.33
156	통계	169.33
163	통계	167
171	통계	164.33
165	경제	175
180	경제	167.5
170	경제	172.5

경영학과의 3개 행으로 계산해보자. 첫 번째 행은 스스로를 제외한 나머지 두개 행의 반응변수 값의 평균($\frac{180+168}{2} = 174$)을 할당한다. 두 번째 행도 마찬가지로 스스로를 제외한 평균($\frac{168+168}{2} = 168$)을 할당한다.

마지막 행도 스스로를 제외한 평균($\frac{168+180}{2} = 174$)을 할당한다. 이처럼 Mean Encoding과 달리 Leave-One-Out Encoding에서는 같은 수준(여기서는 학과)이더라도 서로 다른 값이 할당될 수 있다.

(1) 장점

① 이상치의 영향을 덜 받는다.

② 과적합의 위험성이 Mean Encoding보다 낮다. 스스로를 포함한 모든 행의 반응변수 값을 반영하여 계산하는 Mean Encoding과 달리 LOO Encoding은 스스로의 반응변수 값은 제외하기 때문이다.

(2) 한계점

① Mean Encoding과 동일 (위 내용 참고)

6) Ordered Target Encoding (CatBoost Encoding)

Ordered Target Encoding은 범주형 변수가 많은 데이터 처리에 유용한 CatBoost에서 사용하는 방식으로 CatBoost Encoding이라고도 불린다. Ordered Target Encoding은 같은 수준에 속하는 행들 중 현재 행의 이전 행들 값의 평균을 할당하는 방식이다. 아래 예시를 통해 계산해보자. 이번에는 Ordered Target Encoding의 원리를 정확히 전달하기 위해 행들을 수준에 무관하게 배열하였다.

[Y] 키	[X] 학과	Mean Encoding	Ordered Target Encoding
168	경영	172	169.5
174	통계	166	169.5
165	경제	171.66	169.5
156	통계	166	174
180	경영	172	168
163	통계	166	165
180	경제	171.66	165
170	경제	171.66	172.5
168	경영	172	174
171	통계	166	164.33

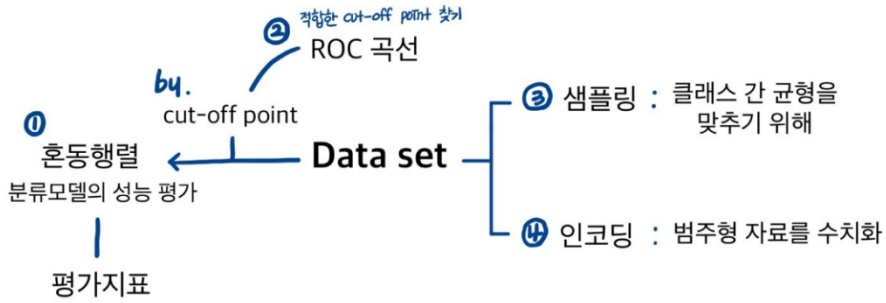
이번에도 경영학과로 논리를 확인해보자. 경영학과에 해당하는 행은 전체 데이터의 0번째, 4번째, 8번째 행이다. 이 때 각 수준의 첫 번째 등장하는 행은 같은 수준에 해당하는 이전 값이 없으므로 전체 데이터의 반응변수의 평균을 사용한다. 따라서 첫 번째 경영학과 행은 전체 데이터 평균(모든 수준 통합 평균)인 169.5을 할

당한다. 두 번째 경영학과 행은 유일한 이전 행인 첫 번째 경영학과 반응변수 값 168을 할당한다. 세 번째 경영학과 행은 이전 행이 두개 존재하므로 앞선 두 경영학과 행의 반응변수 값의 평균인 $\frac{168+180}{2} = 174$ 값을 할당한다.

(Ordered Target Encoding도 고려하는 반응변수의 수를 줄였으므로 과적합의 위험이 Mean Encoding 보다 낮아졌겠죠?)

Ordered Target Encoding 역시 Mean Encoding과 다르게 같은 수준에 해당하는 행이라도 다른 값이 할당될 수 있다.

[3주차 흐름 정리]



[실습코드]

이번 실습 파일은 R이 아닌 파이썬을 통해 실습을 진행하고자 합니다!

지난 2주차 때 실습을 진행하지 않았던 만큼, 이번에는 더 자세하고 다양한 코드들을 포함시켜 실습 파일을 만들어 보았습니다! 패키지와 세미나 준비로 바쁘실 것을 백 번 이해하기에! 과제 형식이 아니라 전체 코드를 보내 드리고 직접 실행해 보며 원리에 대해 익혀 보셨으면 좋겠습니다.

3주차 내용은 실제로 데이터를 분석하는 과정에서 유용하게 사용할 수 있는 기법들에 대해 주로 담았기 때문에 이번 주차 실습은 내용 복습에 특히 더 도움이 되실 것이라고 기대합니다!

실습 파일에서 궁금한 내용이 생기셨다면 주저 말고 저에게 연락 해주세요!😃

[맺음말]

이렇게 3주간의 클린업이 모두 끝났습니다! 작년 2학기가 끝나고 범주팀 팀장을 맡는 것이 정해진 이후로 클린업이 끝나는 날이 올까 싶었을 정도로 막막했는데 정말 이 날이 오는군요..! 어떻게 하면 더 많은 범주형자료분석에 대한 정보와 지식을 여러분에게 정확히 전달할 수 있을지에 대해 끊임없이 고민했던 것 같아요. 비록 부족한 팀장의 클린업을 매주 들어주신 여러분들이 훨씬 더 고생하시고 힘든 한 주 한 주를 보내셨으리라 생각하지만 여러분들에게 도움이 되고자 했던 저의 노력은 진심이었답니다~!😂

방학때는 혼자 공부하는 시간이 많아서 막막하고 지루하기도 했는데, 팀을 꾸리고 여러분들과 만나서 클린업을 진행할수록 더 많은 욕심과 의욕이 생겼던 것 같아요. 여러분들이 계셔서 제가 정말 큰 힘을 많이 얻었습니다. 항상 바쁘다는 핑계로 여러분들과 함께하는 시간을 많이 만들지 못한 것 같아 너무 아쉽지만 범주팀은 이제 1라운드가 끝난 것일 뿐. 이제 진짜 본 라운드 주제분석이 다가오고, 함께할 시간은 훨~~씬 많을 것이기 때문에 ^^ 그때 주제분석 준비도 열심히 하고 같이 놀기도 열심히 하는 걸로 해요!

정말 진심으로! 범주팀 모두 고생 너무 많았고 이제 시험기간 동안 피셋 활동은 잠시 미뤄두고 시험까지 완벽히 챙기는 완벽범주 보여줍시다! 휴학생들은 남 부럽지 않게 푹 쉬면서 체력 보충하는 걸로 해요! 지금까지 너무 고마웠고 남은 기간도 잘 부탁드려요 범주 최고다~~~

임지훈 010-3499-4655

[부록]

여기서 설명되는 부분은 정말x100 Only For 참고… 클린업 내용보다 심화되는 내용이지만 궁금하실 멋진 범주러들을 위해!

I. Multi-Class F1-Score

앞선 클린업 교안에서는 이진분류 모델에 대한 혼동행렬에 대해서만 살펴보았지만, 2개 이상인 n개의 수준으로 분류하는 모델도 물론 존재하며, 이 때 혼동행렬은 n*n 형태를 띌 것이다.

이 때, Multi-Class 상의 F1-Score 계산은 One vs All 방식을 따르며, 해당 클래스를 긍정, 나머지 클래스를 부정으로 간주하여 계산한다. 즉, 우리가 배운 혼동행렬과 같은 모양으로 클래스의 개수만큼 여러 개 생성하는 것이다. 예시는 편의를 위해 3개 클래스 (한국, 중국, 일본)가 있다고 가정한다. (각 혼동행렬을 1번,2번,3번으로 한다)

		관측값(Y)	
		Y = *한국*	$Y \neq$ *한국*
예측값 ($\hat{Y}$)	$\hat{Y}$ = *한국*	3	1
	$\hat{Y} \neq$ *한국*	2	6

		관측값(Y)	
		Y = *중국*	$Y \neq$ *중국*
예측값 ($\hat{Y}$)	$\hat{Y}$ = *중국*	2	2
	$\hat{Y} \neq$ *중국*	2	6

		관측값(Y)	
		Y = *일본*	$Y \neq$ *일본*
예측값 ($\hat{Y}$)	$\hat{Y}$ = *일본*	1	3
	$\hat{Y} \neq$ *일본*	2	6

이렇듯, 3개의 클래스가 있는 경우, 각 클래스를 긍정, 나머지 클래스들을 부정으로 간주하여 혼동행렬을 계산하면 클래스의 개수만큼 혼동행렬이 생성된다. 이 경우 F1-Score를 계산하는 방법은 총 4가지다.

(1) Micro F1-Score

혼동행렬의 TP, FP, FN을 모두 합하여 통일된 F1-Score를 계산한다.

위의 예시의 경우, TP = 6 (3+2+1), FP = 6 (1+2+3), FN = 6 (2+2+2)다. 이 숫자들을 바탕으로 Micro F1-Score를 계산하면, $\frac{6*2}{6*2+6+6} = 0.5$ 가 도출된다.

(2) Macro F1-Score

각 클래스별 F1-Score를 계산하여 산술평균을 구한다.

$$Macro\ F1-Score = \frac{1}{N}\sum F1-Score \ \text{(N : 클래스 개수)}$$

위의 예시의 경우, 1번 혼동행렬의 F1-Score는 $\frac{2*3}{2*3+1+2}$ = 약 0.67, 2번 혼동행렬의 F1-Score는 $\frac{2*2}{2*2+2+2}$ = 0.5, 3번 혼동행렬의 F1-Score는 $\frac{2*1}{2*1+3+2}$ = 약 0.29가 나온다. 따라서 세 F1-Score의 산술평균인 Macro F1-Score는 $\frac{0.67+0.5+0.29}{3}$ = 약 0.487이 도출된다.

(3) Weighted Average F1-Score

각 클래스별 F1-Score를 계산하여 실제 데이터의 빈도수에 따라 가중평균을 구한다.

위의 예시의 경우, 1번 혼동행렬의 F1-Score는 약 0.67, 2번 혼동행렬의 F1-Score는 0.5, 3번 혼동행렬의 F1-Score는 약 0.29였다. 여기에 실제 데이터 개수는 {한국 : 5, 중국 : 4, 일본 :3}이므로 이에 따른 빈도수의 가중평균 값을 반영한다. 따라서 $0.67 * \frac{5}{12} + 0.5 * \frac{4}{12} + 0.29 * \frac{3}{12}$ = 약 0.518이 구해진다.

(4) None

위의 세 방법처럼 최종 F1-Score 한 값을 도출하지 않고 3개의 F1-Score를 각각 표시한다.

4가지 F1-Score 계산 방법 중 정답은 없다. 데이터 분포나 특징에 따라 가장 알맞은 방식을 선택하여 활용하는 것이 바람직하다!

II. 다양한 Sampling 기법

클린업에서 소개한 언더 샘플링, 오버 샘플링 기법들 외에도 다양한 기법들이 있다. 각각의 방법들에 대해 간단히만 알아보자.

1. 언더 샘플링

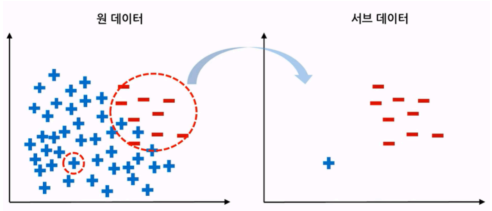
종류	특징
Condensed Nearest Neighbors (CNN)	- KNN 알고리즘 사용 (K=1이어야 함) - 다수의 클래스끼리 밀집된 곳을 제거 - 가장 처음에 선택하는 기준 데이터는 랜덤 선택
Edited Nearest Neighbors (ENN)	- Tomek Links 기법처럼 클래스 경계에 있는 노이즈 데이터들을 제거하는 방식 - 주로 다른 기법들과 결합하여 사용된다.
One-sided selection (OSS)	Tomek Links + CNN
Neighborhood Cleaning Rule (NCR)	ENN + CNN

Random Under Sampling & Tomek Links 외에도 위와 같은 방법들이 존재한다. 그림을 통해 살펴보자.

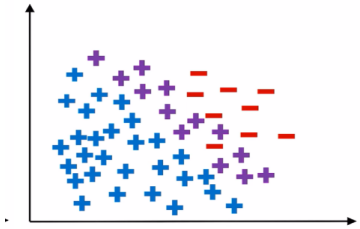
(1) Condensed Nearest Neighbors (CNN)

CNN은 KNN기법을 사용하여 1-NN(밑에서 설명) 모형으로 분류되지 않는 데이터만 남기는 방법이다. 과정은 아래와 같다.

① 소수 범주 전체와 다수 범주에서 무작위로 하나의 관측치를 선택하여 서브데이터를 구성한다.

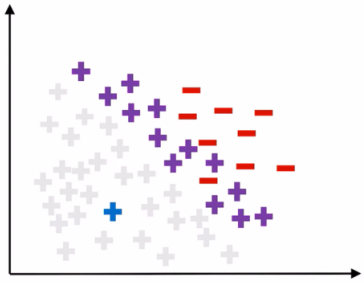


② 1-NN 알고리즘을 통해 다수 범주 관측치를 재분류한다. 쉽게 말해, 다수 범주 관측치들을 소수 범주 데이터들과 무작위로 선택된 다수 범주 데이터 중 어디에 더 가까운지에 따라 분류하는 것이다.



보라색 : 다수 범주 관측치 중 소수 범주 관측치들과 가까운 집합
파란색 : 다수 범주 관측치 중 임의의 다수 범주 관측치와 가까운 집합

③ 정상(푸른색) 분류된 다수 범주 관측치 제거



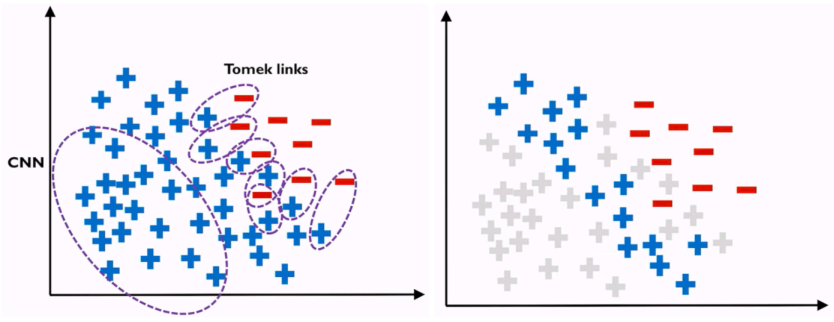
✓ 이 때, 1-NN 알고리즘에서 1보다 더 큰 값을 사용할 경우, 모든 다수 범주가 이상값 (소수 범주 데이터들과 가까운 집합)으로 분류되므로 반드시 k=1이어야 한다!

(2) One-Sided selection (OSS)

OSS는 Tomek Links와 CNN 기법을 합친 방법으로, 각각의 단점을 보완하는 효과를 가진다.

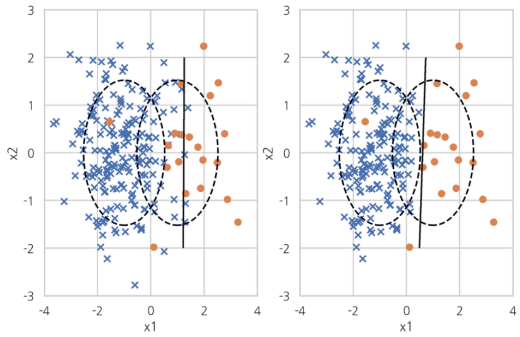
- Tomek Links는 borderline 관측치가 없어지는 경향
- CNN은 완전한 다수 범주 관측치가 없어지는 경향

위의 두 경향의 단점을 보완하여 Tomek Links, CNN에 해당하는 다수 범주 데이터를 모두 제거한다.



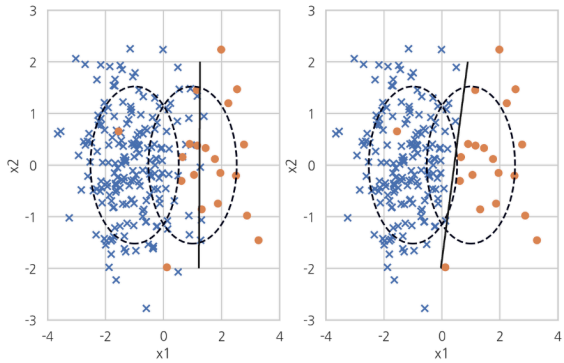
(3) Edited Nearest Neighbors (ENN)

CNN 기법이 소수 범주 데이터와 가까운 데이터를 남긴다면, ENN은 반대로 클래스 경계에 있는 데이터를 제거한다. 다수 클래스 데이터 중 가장 가까운 k개의 데이터를 구하여, 모든 또는 다수(분석자가 설정할 수 있다) k개의 데이터가 다수 클래스가 아니면 삭제하는 방법이다. 결과적으로 소수 클래스 주변의 다수 클래스 데이터는 사라진다. ENN은 Tomek Links Method 처럼 클래스를 구분하는 임계점을 다중 클래스 쪽으로 밀어낼 수 있지만 제거 효과가 크지 않은 것을 알 수 있다.



(4) Neighborhood Cleaning Rule (NCR)

NCR 기법은 위에 설명한 CNN과 ENN을 결합하여 해당하는 데이터를 모두 제거하는 방식이다. 분포가 큰 데이터에 대한 제거 효과가 크지 않지만 좀 더 직관적으로 두 클래스를 나눌 수 있는 장점이 있다. CNN과 ENN 방식을 설명했으므로 자세한 설명은 생략한다. (궁금하면 찾아와 주세요!)



2. 오버 샘플링

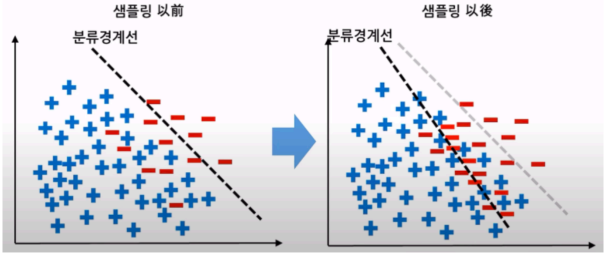
종류	특징
Borderline-SMOTE	- 각 minority point가 어떠한 유형의 관측치 인지 KNN을 통해 판단 - Danger 관측치라고 분류된 minority point에 대해 SMOTE 진행 (Danger 관측치: 하나의 minority point에서 주위에 있는 데이터 중 majority point의 비율이 50%이상인 경우)
Borderline – SMOTE SVM	- 클래스간 경계를 SVM을 통해 판단 - SVM을 통해 분별한 클래스 경계선 근처에서 minority point에 대한 SMOTE 진행
ADASYN	- 소수 클래스 주위에 있는 다수 클래스 관측치의 비율을 계산 - 계산된 수치만큼 각 minority point에서 SMOTE 진행 (즉 비율이 높으면 많이 생성, 비율이 적으면 적게 생성)

오버 샘플링에도 위와 같이 다양한 방법들이 존재한다. 앞서 SMOTE에선 새롭게 생성된 데이터가 다수의 클래스와 겹치거나 노이즈가 발생할 수 있다는 한계점을 지녔다. 아래 표에 있는 오버 샘플링 기법들은 SMOTE를 확장하고 변형하여 해당 한계를 보완하는 기법들이다. 원리는 약간씩 다르지만, SMOTE와는 달리 노이즈를 고려한다는 공통점을 지닌다. 그림을 통해 간단히 알아보자!

(1) Borderline-SMOTE

Borderline-SMOTE는 SMOTE에서 발전된 기법으로, “Danger”로 분류된 소수 범주 데이터에 대해서만 SMOTE를 적용한다. 소수 범주 데이터는 3가지 종류로 분류되는데, 각 데이터의 k개의 가장 가까운 데이터를 찾아 k 중에 50%보다 크고 100%보다 작은 (50% 이하는 “Safe”, 100%는 “Noise”로 분류) 개수가 다수 범주 데이터에 해당할 때 이 데이터를 “Danger”로 분류한다.

“Danger” 데이터에 대해서만 SMOTE를 진행한 결과 아래의 그림과 같이 Borderline 근방에서만 데이터가 새롭게 생성된 것을 알 수 있다. 따라서 Borderline-SMOTE 기법에서는 Danger 데이터를 정확히 파악하는 것이 매우 중요하다.

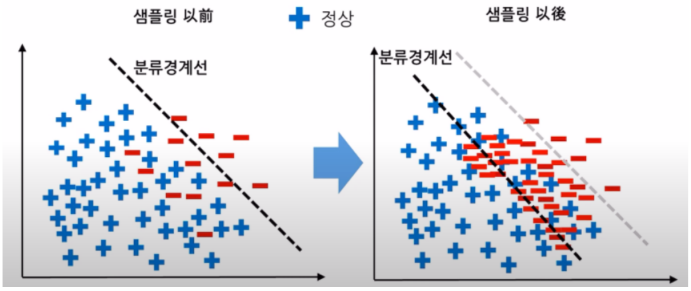


(2) ADASYN

Borderline-SMOTE는 경계선에 집중해서 Sampling을 진행한다면, ADASYN은 Sampling 개수를 위치에 따라 다르게 하는 방법이다.

① 새롭게 생성하고자 하는 샘플의 개수가 G (일반적으로 다수 범주 데이터 개수 – 소수 범주 데이터 개수)라면, 각 소수 범주 데이터마다 k개의 가까운 데이터 중 다수 범주 데이터에 해당하는 개수의 비율을 계산하여 Scaling을 진행한 후 G를 곱한 값으로 Sampling 개수를 차등 설정한다.

② 모든 소수 범주 데이터에 대해서 ①에서 계산된 횟수만큼 각각 SMOTE 알고리즘을 사용하여 오버샘플링을 진행한다.



위 그림과 같이, ADASYN을 통해 오버샘플링을 진행하면, 클래스 간 경계선 뿐만 아니라 다수클래스 근처의 소수클래스 증폭도 진행된다. (이것이 ADASYN의 장점) 하지만 비교적 긴 처리 시간이 요구된다는 단점이 있다.

진짜 끝!!!!