범주형 자료분석 3주차 교안

신나는 행복 범주 클린업 3주차 시간이 돌아왔습니다~(짝짝짝)

지난시간에는 GLM에 대해서 배웠었죠?

이번주는 혼동행렬, 분류지표, 샘플링, 인코딩을 배워볼 예정입니다!

이번주는 어려운 내용은 없으니 부담 없이 가벼운 마음으로 임해보아요~ㅎㅎ

벌써 3주차가 되었네요!!

마지막 클린업까지 오신 여러분 고생 많으셨고

마지막까지 최선을 다해봅시다 ㅎㅎ

목차

1. 혼동 행렬 (Confusion Matrix)

■ 혼동행렬이란?

혼동행렬(Confusion Matrix)이란 예측값($\hat{Y}$)과 실제값(Y)을 비교하기 위한 행렬(표)로, 분류 모델의 성능을 평가하는 지표라고 할 수 있다. 혼동행렬을 통해 모델이 얼마나 실제값을 잘 예측했는지를 확인할 수 있다. 혼동행렬의 형태는 다음과 같다. 우리는 반응변수 Y가 이항변수인 경우만 볼 예정이다. 만약 다항변수라면 3X3 등의 혼동행렬이 만들어진다.

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

각 칸을 보면 TP,FP 이런 것들이 적혀 있는데 **T(True)와 F(False)**는 실제와 예측이 같은지 혹은 다른지를 나타내고, 뒤의 **P(Positive)와 N(Negative)**는 예측을 긍정 혹은 부정이라 했는지 여부를 나타낸다. 쉽게 말해 T/F는 내 예측이 맞았는지 틀렸는지를 의미하고, P/N은 내가 뭐라고 예측했는지를 의미한다. 가령 FN은 아니라고 했는데 틀린 경우를 의미한다. 통계학원론에서 배운 내용을 적용하면 FP는 1종오류(음성을 양성으로 판정), FN은 2종오류(양성을 음성으로 판정)라고도 표현할 수 있다. 코로나 검사로 예시를 들어보자.

- TP(True Positive)는 코로나 양성이라고 예측된 환자가 코로나 양성인 경우 (맞춤)

- TN(True Negative)는 코로나가 음성이라고 예측된 환자가 코로나 음성인 경우 (맞춤)

- FP(False Positive)는 코로나가 양성이라고 예측된 환자가 코로나 음성인 경우 (틀림)

- FN(False Negative)는 코로나가 음성이라고 예측된 환자가 코로나 양성인 경우 (틀림)

하지만 혼동행렬은 분류모델의 성능을 평가하기에 한계점이 존재한다. 로지스틱 회귀 모형은 예측 확률 $\hat{\pi}$을 연속적인 값으로 반환하는데 반해, 예측은 그 확률을 보고 cut-off point를 기준으로 예측값 $\hat{Y}$를 0(아님) 또는 1(맞음)로 범주화해야 하기 때문이다. 즉, 연속인 값($\hat{\pi}$)을 이항값($\hat{Y}$)으로 묶어버리다 보니 ① 정보의 손실이 발생하는 것이다.

또 다른 문제는 ②cut-off point의 선택이 임의적이라는 점이다. 분석가가 cut-off point를 무엇으로 정하느냐에 따라 혼동행렬의 값은 달라진다! 특히 클래스 불균형이 심한 경우에는 cut-off point에 따라서 혼동행렬이 크게 바뀐다.

예를 들면 예측확률 $\hat{\pi}=0.66$라 하면, cut-off point가 0.5인 경우에는 0.66>0.5니까 $\hat{Y}=1$이고, cut-off point가 0.7인 경우에는 $\hat{Y}=0$이 된다. 예측값이 다르니 혼동행렬도 달라질 것이다. 또한, 0.66이라는 정보는 사라지게 된다.

혼동행렬은 이런 한계점이 존재하긴 하지만 한계점을 보완할 수 있는 방법도 있고,(이건 이따가 ROC에서 배울 예정ㅎㅎ) 모델의 성능을 평가할 다양한 평가지표가 있기에 널리 사용된다. 어떤 평가지표들이 있는지 확인해보자.

■ 분류 평가지표

혼동행렬을 통해 다양한 평가지표를 이용할 수 있다. 상황에 따라 사용해야 하는 평가지표가 달라질 수 있기 때문에, 적절한 평가지표를 사용하는 것이 중요하다. 그럼 어떤 종류가 있는지 보자!

1) 정확도 (Accuracy)

정확도(accuracy/ACC/정분류율)는 전체 경우에서 실제값과 예측값이 같은 경우의 비율이다. 즉, 예측이 실제값과 얼마나 정확히 일치하는지를 나타내는 지표이다.

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN\ (\textit{전체})}$$

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

굵은 테두리가 분모, 파란 색칠한 부분이 분자라고 생각하면 된다! 얼마나 정확한지를 보는 지표이니 정확도가 1에 가까울수록 좋은 모형이라고 할 수 있다. 정확도는 직관적으로 이해하기도 쉽고 단순해서 가장 많이 쓰인다. 하지만 특정 범주에 데이터가 쏠려 있는 **unbalanced data**일 때 해당 범주에 지나치게 의존적이게 되기 때문에 모형을 평가하는 경우 문제가 생긴다. 따라서 다른 지표도 고려해봐야 한다.

2) 정밀도 (Precision)

정밀도(Precision/PPV/Positive Predictive Value)는 맞다고 예측한 것 중 실제로 맞는 것이 얼마인지를 비율로 나타내는 지표이다. 즉, $\hat{Y} = 1$이라고 말했던 것 중에서 실제로 $Y = 1$인 경우를 비율로 나타낸 것이다. 앞선 정확도(accuracy)와는 다르게 unbalanced data일 때 특정 범주에 대한 의존성을 줄어드는 장점이 있다.

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

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

정밀도는 FP가 더 critical한 경우에 사용된다. 예를 들어, 재판을 생각해보자. 죄 있는 사람을 무죄라고 하는 것(FN)은 문제가 있긴 하지만 흔히 발생한다. 하지만 결백한 사람을 유죄로 판결하는 경우(FP) 상당한 문제가 발생한다. 이처럼 실제로는 아닌데 맞다고 판단하는 FP가 더 치명적일 때 정밀도를 사용하여 모형을 평가한다.

3) 민감도 (=재현율)

민감도(Sensitivity/TPR/True Positive Rate) 또는 **재현율(Recall)**은 실제로 맞는 것 중에서 맞다고 예측한 것이 얼마인지를 비율로 나타낸 지표이다. 즉, 실제로 $Y = 1$인 것 중에서 $\hat{Y} = 1$이라고 예측한 것의 비율을 나타낸 지표이다. 민감도는 높으면 맞는 것을 잘 맞췄다는 뜻이니까 1에 가까울수록 좋다. 민감도도 정밀도와 마찬가지로 정확도(accuracy)와는 다르게 unbalanced data일 때 특정 범주에 대한 의존성을 줄어드는 장점이 있다. 참고로 민감도는 ROC곡선의 Y축이다.(이따 배울예정!)

$$Sensitivity = \frac{TP}{TP + FN}$$

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

민감도는 FN이 더 critical한 경우에 사용된다. 예를 들어, 코로나 예시를 다시 생각해보자. 코로나에 안 걸린 사람을 걸렸다고 하는 것(FP)은 다시 검사를 해보면 되지만, 코로나에 걸린 사람을 안 걸렸다고 하는 것(FN)은 방역에 큰 악영향을 미친다. 이처럼 실제로는 맞는데 아니라고 판단하는 FN이 더 치명적일 때 민감도를 사용하여 모형을 평가한다.

4) 특이도 (Specificity)

특이도(Specificity/TNR/True Negative Rate)는 실제로 아닌 것 중 아니라고 예측한 것이 얼마인지를 비율로 나타낸 지표이다. 즉, 실제로 $Y = 0$인 것 중에서 $\hat{Y} = 0$이라고 예측한 것의 비율을 나타낸 지표이다. 특이도 역시 높다면 아닌 것을 잘 맞췄다는 뜻이니까 특이도는 1에 가까울수록 좋다.

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

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

예를 들면, 코로나에 안 걸린 사람을 음성이라고 하는 확률 $P(\text{음성} \mid \text{코로나 } X)$을 특이도라고 할 수 있다.

특이도의 반대 경우를 **FPR(False Negative Rate)**이라고 하는데, 실제로 아닌 것 중 맞다고 예측한 것을 비율로 나타낸 지표이다.

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

이는 ROC곡선의 X축이 된다. 또한, FPR은 (1-특이도)와 같다. 따라서 FPR은 0에 가까울수록 좋다.

5) F1-Score

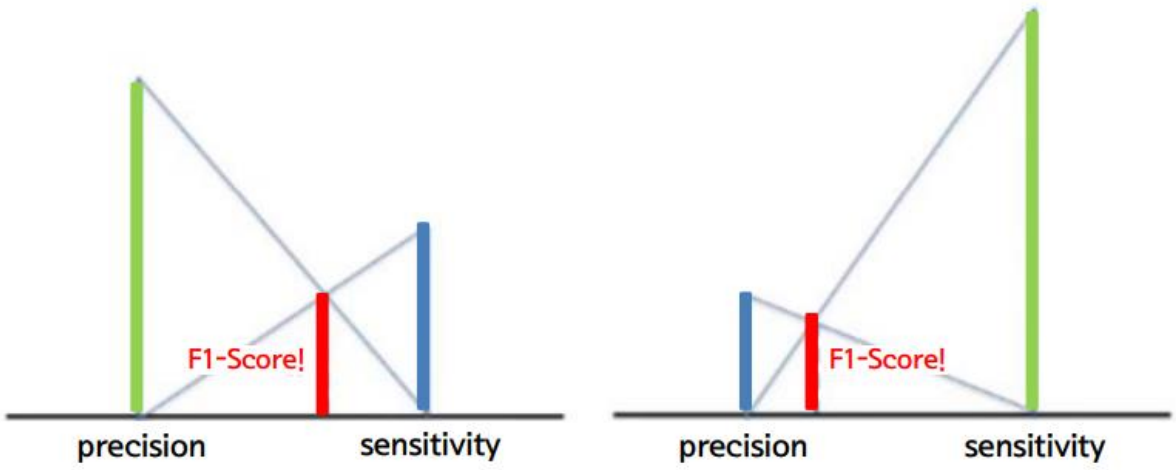
F1-Score는 정밀도(Precision)와 민감도(=재현율,Recall, Sensitivity)의 조화평균이다. 조화평균은 역수의 산술평균의 역수이다. (조화평균 $= \frac{1}{\frac{\frac{1}{a}+\frac{1}{b}}{2}} = \frac{2}{\frac{1}{a}+\frac{1}{b}} = \frac{2ab}{a+b}$) F1-Score 역시 1에 가까울수록 좋은 모델을 의미한다.

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

앞서 정밀도(Precision)와 민감도(Recall) 모두 unbalanced data일 때 정확도(Accuracy)가 가지는 단점을 보완하는 지표였다. 만약에 이 둘을 동시에 잘 이용하면 unbalanced data일 때 좀 더 모형을 잘 평가하는 지표를 만들 수 있지 않을까? 따라서 F1-Score는 이 두 지표를 모두 사용해서 이 두 지표의 조화평균을 고려한다.

그렇다면 왜 조화평균일까? 흔히 쓰는 산술평균을 안 쓰고 조화평균을 쓰는 이유는 정밀도와 민감도의 trade-off를 고려해서 두 지표 모두 균형 있게 반영하기 위함이다.

Trade-off란 상충관계를 의미한다. 즉 정밀도(Precision)와 민감도(Recall)는 동시에 최대값을 가질 수 없다. 정밀도가 커지면 민감도는 작아진다. 두 명이서 라면 하나를 먹는 경우를 생각하면 된다. 한 명이 많이 먹으면 다른 한 명은 조금 먹는 그런 관계가 바로 Trade-off이다. 정밀도(Precision)와 민감도(Recall)는 둘 다 클수록(1에 가까울수록) 좋은 모델을 의미했다. 하지만 Trade-off로 인해 동시에 큰 값을 가질 수 없기 때문에 이를 적절히 고려해서 F1-Score 값을 구해야 한다. 이를 위해 조화평균을 사용하는 것이다. 조화평균은 큰 값을 갖는 쪽에 페널티를 주어 작은 값에 가까운 평균을 구한다. 이런 원리로 unbalanced data에서 큰 값을 가지는 클래스에 대해 페널티를 줄 수 있기에 두 지표의 밸런스를 고려해 줄 수 있다. 그림으로 그려보면 이런 느낌이다.



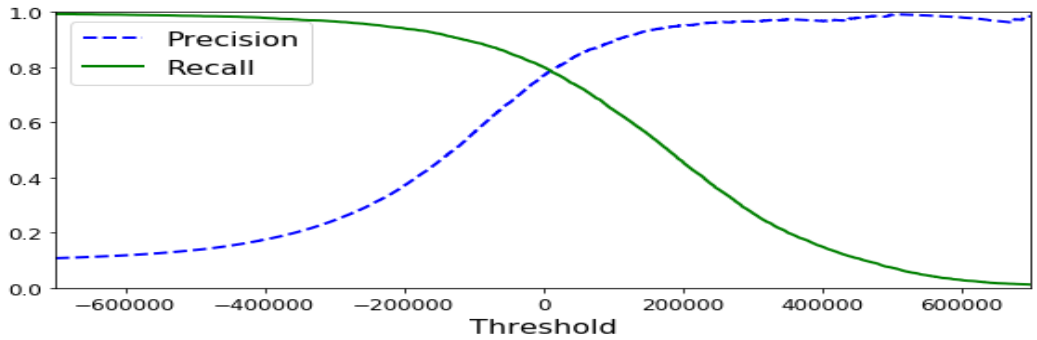
예를 들어 만약 민감도가 1 이고 정밀도가 0.01로 측정된 모델이 있다고 할 때, 이 둘의 산술평균과 조화평균을 따져보자.

$$산술평균 = \frac{1 + 0.01}{2} = 0.505$$

$$조화평균(F1\ Score) = 2 \times \frac{1 \times 0.01}{1 + 0.01} = 0.019$$

민감도가 1로 높지만 정밀도가 0.01로 무척 낮기 때문에 이 모델은 문제가 있다고 판단된다. 하지만 산술평균으로 구하게 되면 이 모델의 성능이 0.505로 꽤 높게 나온다. 하지만 조화평균으로 계산하면, F1 score가 0.019로 매우 낮게 나온다. 이런 이유로 조화평균을 사용하는 것이다~

아래는 정밀도와 민감도의 Trade-off 관계를 그래프로 나 그림이다.



하지만, F1 score는 TN을 고려하지 않는다는 단점이 있어서 F1-Score만을 가지고 모델을 평가하기에는 몇 가지 문제가 발생한다. 예시를 들어보자.

		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

$$F1\ Score = \frac{2 \times 26}{2 \times 26 + 24 + 27} = 0.505$$

왼쪽과 오른쪽 혼동행렬에서 TN의 값은 다르지만, 둘 다 F1-Score값 0.505로 같음을 알 수 있다. 즉 F1-Score를 사용하면 이 두 경우를 구분하지 못하게 된다.

6) MCC

이런 F1-Score의 단점을 보완할 수 있는 지표로는 **MCC** (Matthews correlation coefficient,매튜 상관계수/파이계수)가 있다.

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

이름에서 알 수 있듯이 이 지표는 상관계수이다. 따라서 우리가 아는 피어슨 상관계수와 동일하게 -1~1사이의 값을 갖고, 1에 가까울수록 완전 예측, -1에 가까울수록 완전 거꾸로 예측, 0에 가까울수록 찍는것과 다름없는 랜덤예측이라고 할 수 있다.

MCC는 F1-Score와는 다르게 혼동행렬의 모든 부분을 사용하여 만들어지기 때문에 F1-Score의 단점을 보완할 수 있는 지표라고 할 수 있는 것이다. 실제로 어떤 식으로 작용하는지 unbalanced data 예시를 살펴보자.

		관측값(Y)	
		$Y=1$	$Y=0$
예측값($\hat{Y}$)	$\hat{Y}=1$	90	4
	$\hat{Y}=0$	5	1

		관측값(Y)	
		$Y=1$	$Y=0$
예측값($\hat{Y}$)	$\hat{Y}=1$	1	5
	$\hat{Y}=0$	4	90

왼쪽 혼동행렬에서 $F1\ Score = \frac{2\times90}{2\times90+4+5} = 0.95,\ MCC = \frac{(90\times1)-(5\times4)}{\sqrt{(90+5)(90+4)(1+5)(1+4)}} = 0.14$로 계산된다.

오른쪽 혼동행렬에서 $F1\ Score = \frac{2\times1}{2\times1+5+4} = 0.18,\ MCC = \frac{(90\times1)-(5\times4)}{\sqrt{(90+5)(90+4)(1+5)(1+4)}} = 0.14$로 계산된다.

두 혼동행렬은 모두 MCC값이 0.14로 모델의 성능이 좋다고 할 수 없다. 하지만, 왼쪽 혼동행렬의 경우 F1 Score의 값이 0.95로 매우 높게 나오는 모순이 발생한다. 이 때문에 앞서 F1 Score 하나만 보고 모델을 평가하는 데는 문제가 있다고 한 것이다!

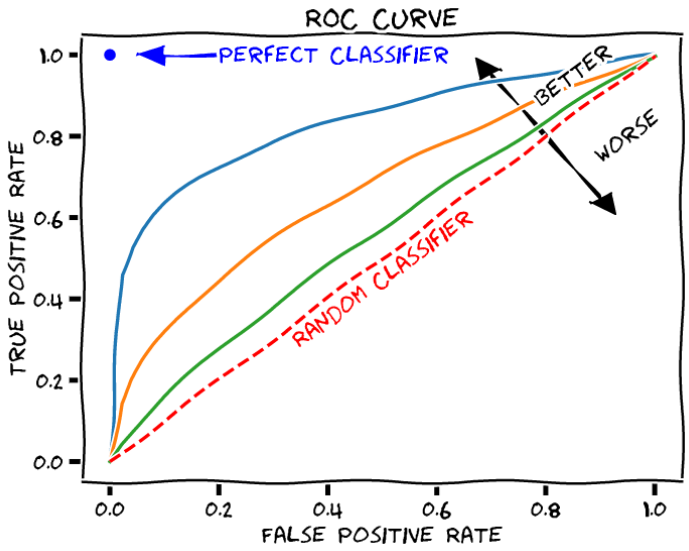
이처럼 TN을 사용하지 않는 F1 Score은 값이 엄청난 차이를 보이는 반면, MCC는 모든 성분을 사용하기 때문에 상당히 균형 잡힌 척도라고 할 수 있다. 사실 F1 Score는 위와 같은 문제가 발생하지 않고 비대칭 데이터를 효과적으로 평가하기 위해 적은 범주를 positive로 두고 계산한다고 한다.

2. ROC 곡선

■ ROC 곡선이란?

ROC 곡선 (Receiver Operating Characteristic curve)이란 가능한 모든 cut-off point에 대해 민감도(Sensitivity, =재현율,Recall) 또는 1-특이도(Specificity)를 나타낸 곡선(또는 직선)이다. 쉽게 말해 cut-off point가 0, 0.023, 0.1, 0.37, 0.5,… , 1까지 모든 경우를 다 따져서 무수히 많은 혼동행렬을 만들고 각각의 민감도와 1-특이도를 계산해서 그래프로 그려준 게 바로 ROC곡선이다.

우리는 앞서 혼동행렬은 cut-off point를 무엇으로 정하는지에 따라 그 값이 달라지고 정보의 손실이 일어난다는 한계점이 있다고 배웠다. ROC 곡선은 이 한계점에 대한 해결책을 제공한다. ROC 곡선은 모든 cut-off point에 대하여 예측 검정력을 구하기 때문에 ①혼동행렬보다 더 많은 정보를 갖고, ②가장 적합한 cut-off point를 찾을 수 있도록 해준다.



ROC 곡선의 형태는 위와 같다. ROC 곡선의 X축은 FPR (1-특이도)이고, Y축은 TPR(민감도,재현율)이다.

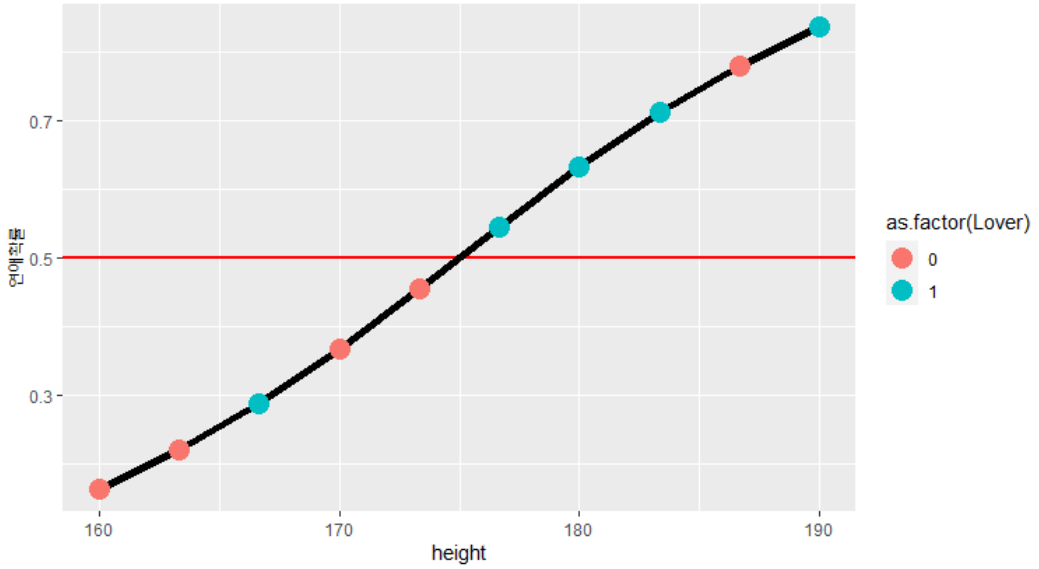
X축과 Y축 모두 0~1의 범위를 갖고, ROC 곡선은 (0,0)과 (1,1)을 잇는 우상향하는 위로 볼록한 곡선의 형태를 갖는다. 각각의 그래프는 다양한 모델의 ROC 곡선이다. 곡선 위의 각 점들은 cut-off point에 따른 FPR, TPR값이다. 이 점들을 이으면 ROC곡선이 되는 것이다.

ROC곡선 위의 점들은 cut-off point가 작을수록 (0,0)에 가까워지고, 클수록 (1,1)에 가까워진다. 왜 그런지 알고리즘을 정리해보았다.

- Cut-off point가 0에 가까워지면 -> (기준이 널널해졌으므로) 대부분 $\hat{Y} = 1$로 예측! -> TP&FP 증가 / TN&FN 감소 -> FPR과 TPR이 1에 가까워진다!
- Cut-off point가 1에 가까워지면 -> (기준이 까다로워졌으므로) 대부분 $\hat{Y} = 0$으로 예측! -> TP&FP 감소 / TN&FN 증가 -> FPR과 TPR이 0에 가까워진다!

예를 들어, 똑같은 확률 0.3에 대해서 cut-off point가 0.01이면 $\hat{Y} = 1$로 예측하고, cut-off point가 높아져서 0.7이 되면 $\hat{Y} = 0$로 예측한다고 생각하면 된다. $\hat{Y} = 1$인 값이 커지면 한정된 수에서 P라고 한 부분이 많아지니까 TP와 FP가 증가 & TN과 FN이 감소하는 것이고, $Sensitivity = TPR = \frac{TP}{TP+FN}$, $FPR = \frac{FP}{FP+TN}$에서 TP와 FP가 증가했으므로 TPR과 FPR이 증가하는 것이다.

2주차 때 자주 들었던 키와 연애 예시를 들어서 ROC 곡선 개념을 명확히 해보자!



위의 로지스틱 회귀모형 식이 $\log\left[\frac{\pi(x)}{1-\pi(x)}\right] = -19 + 0.1x$이고 $Y = 1$(연애성공), $Y = 0$(실패), x는 키(cm)를 의미한다고 하자. Y축은 연애할 확률을 의미하고, 각 점은 관측값이다. 가령 키가 190cm인 사람은 연애할 확률이 0.9이다. 이 로지스틱 회귀를 기반으로 혼동행렬을 만든다. 혼동행렬은 cut-off point를 기준으로 만들기 때문에, 연구자가 임의로 지정해 주어야 한다. 위의 빨간 선처럼 cut-off point를 0.5라고 하면 빨간 선을 기준으로 위는 $\hat{Y} = 1$, 아래는 $\hat{Y} = 0$으로 아래와 같은 혼동행렬이 만들어진다.

Cut-off = 0.5		관측값(Y)	
		$Y = 1$	$Y = 0$
예측값($\hat{Y}$)	$\hat{Y} = 1$	4	1
	$\hat{Y} = 0$	1	4

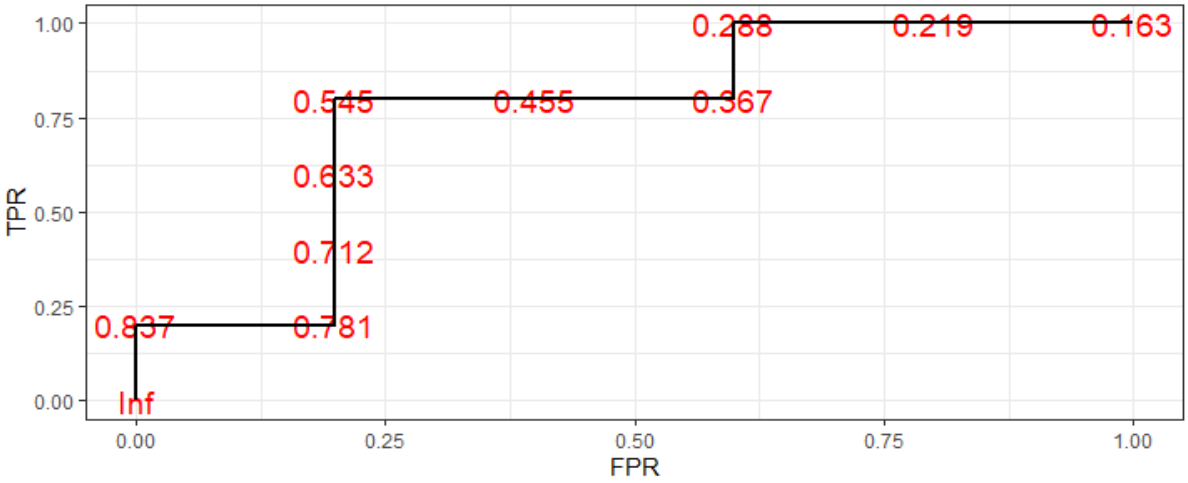
이때 $TPR = \frac{TP}{TP+FN} = \frac{4}{4+1} = 0.8$, $FPR = \frac{FP}{FP+TN} = \frac{1}{1+4} = 0.2$로 계산된다.

같은 모델에서 cut-off point가 0.7인 경우를 살펴보자.

Cut-off = 0.7		관측값(Y)	
		$Y=1$	$Y=0$
예측값($\hat{Y}$)	$\hat{Y}=1$	2	1
	$\hat{Y}=0$	3	4

이때 $TPR = \frac{TP}{TP+FN} = \frac{2}{2+3} = 0.4$, $FPR = \frac{FP}{FP+TN} = \frac{1}{1+4} = 0.2$로 계산된다.

이런 식으로 cut-off point에 따라 무수히 많은 혼동행렬이 만들어지고 TPR과 FPR이 계산된다. 이를 그림으로 그리면, 다음과 같은 ROC 곡선이 만들어진다!



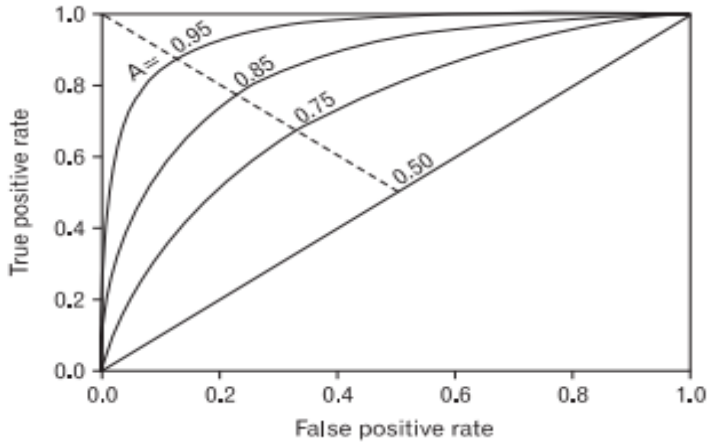
위의 빨간 글씨는 cut-off point를 의미하고, ROC 곡선의 각 점은 cut-off point에 따른 TPR과 FPR의 좌표를 나타낸다. cut-off point가 클수록 (0,0)에 가까워지고, cut-off point가 작을수록 (1,1)에 가까워짐을 확인할 수 있다.

TPR은 예측과 실제가 같은 경우를 나타낸 지표이므로 1에 가까울수록 좋고, FPR은 예측과 실제가 다른 경우를 나타낸 지표이므로 0에 가까울수록 좋다는 것을 배웠다. 이를 참고해서 최적의 cut-off point를 구하면 된다!

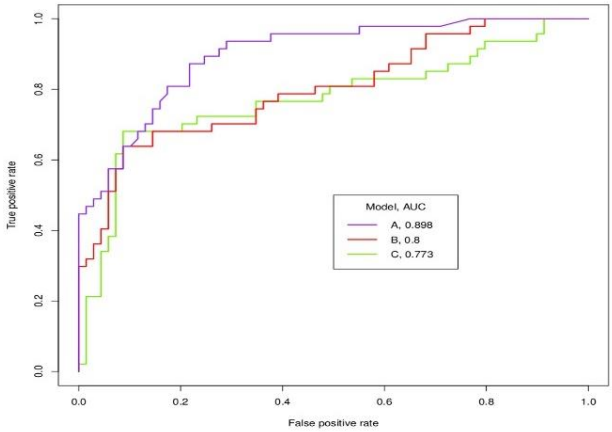
- 같은 Y값(TPR)일 때, X값(FPR)이 더 작을수록 좋은 cut-off point이고,
- 같은 X값(FPR)일 때, Y값(TPR)이 더 클수록 좋은 cut-off point인 것이다!

■ AUC란?

AUC (Area Under Curve)란 말그대로 곡선 아래의 넓이, 즉 ROC 곡선 아래의 면적을 의미한다. AUC는 넓이이기 때문에 0에서 1 사이의 값을 갖는다.



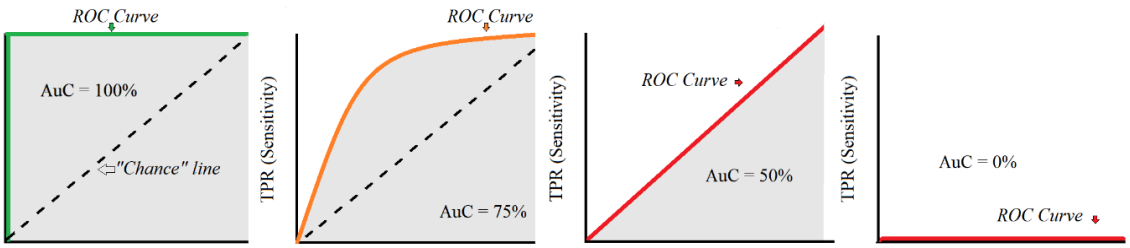
AUC 값이 클수록 더 좋은 예측 성능을 가진 모델이라고 해석할 수 있다. 앞에서 X값(FPR)이 고정되었을 때 Y값(TPR)이 클수록 더 좋은 모델임을 배웠다. 따라서 ROC 곡선이 위로 볼록할수록 성능이 더 좋은 모델이고, AUC 값이 커지게 된다.



이런 식으로 다양한 모델에 대한 ROC곡선이 그려지면, 각각의 AUC값을 구해서 모델 간의 성능을 비교할 수 있게 된다. 위의 경우는 AUC가 0.898인 보라색 ROC 곡선을 그리는 모형이 가장 좋은 성능을 갖는다.

ROC 곡선은 모든 cut-off point를 반영한 그래프이므로, AUC를 사용하면 cut-off point와 상관없이 모델의 성능을 측정할 수 있다.

아래는 다양한 ROC 커브에 대한 AUC를 나타낸 그림이다.



- AUC=1인 경우 : 100% 완벽하게 예측한 경우이다. 이런 경우는 모델이 과적합(overfitting)되었을 가능성을 고려해보아야 한다.

- AUC=0.75인 경우 : 75% 맞춘 예측으로, 뭐 적당한 모델이라고 할 수 있다. 보통 0.8 정도 넘어가면 좋은 모델이라고 할 수 있다.
- AUC=0.5인 경우 : 50%만 맞춘 예측으로, 랜덤하게 예측한 결과와 다름없는 성능이다. (찍은거나 다름없다는 소리다.) AUC는 보통 0.5 이상의 값을 가져야 정상이다. 그럼 만약 AUC가 0.5보다 낮은 값을 갖는 경우는 무슨 의미일까? 분류군을 반대로 처리한 경우라고 생각하면 된다. 즉, 예측을 거꾸로 했다는 소리다.
- AUC=0인 경우 : 100% 반대로 예측한 경우이다. 완벽하게 반대로 예측한 경우니까 거꾸로 해주면 완벽하게 맞춘 경우가 된다. (시험문제를 완벽하게 틀리는 것은 다 맞는 것 만큼 어렵지,,ㅎ)

3. 샘플링 (Sampling)

■ 클래스 불균형

범주형 자료에서 각 클래스(=수준)가 갖고 있는 데이터의 양에 큰 차이가 있는 경우 **클래스 불균형**이 있다고 말한다. 위에서 여러 번 언급했던 unbalanced data와 같은 의미이다. 현실 데이터에서는 클래스 불균형이 존재하는 경우가 많다.

하지만 우리가 소수의 클래스에 특별히 더 관심이 있는 경우에 클래스 균형을 맞출 필요가 있다. 반응변수 Y의 클래스의 데이터 양이 너무 많이 차이 나게 되면 단순히 양이 많은 클래스를 택하는 모형의 정확도가 높게 나타나기 때문에 모델의 성능을 판별하기 어렵게 된다.

		관측값(Y)	
		$Y=1$	$Y=0$
예측값($\hat{Y}$)	$\hat{Y}=1$	93	4
	$\hat{Y}=0$	2	1

		관측값(Y)	
		$Y=1$	$Y=0$
예측값($\hat{Y}$)	$\hat{Y}=1$	60	2
	$\hat{Y}=0$	35	3

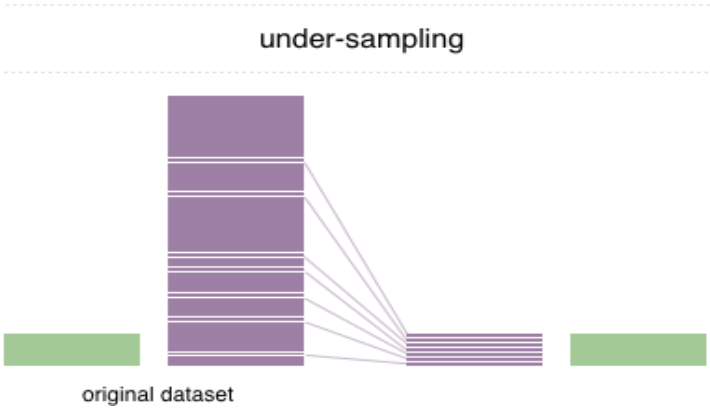
예를 들어 $Y=1$이 95개, $Y=0$이 5개로 클래스 불균형을 갖는 반응변수 Y가 있다고 할 때, 이를 예측하는 두 모델이 있다고 하자. 왼쪽은 무조건 데이터 양이 더 많은 클래스인 $Y=1$로 예측하는($\hat{Y}=1$) 모형이고 오른쪽은 골고루 선택했다고 하자. 이 경우 정확도(accuracy)를 따져보면 왼쪽 $Accuracy = \frac{93+2}{93+4+2+1} = 0.95$, 오른쪽 $Accuracy = \frac{60+3}{60+2+35+3} = 0.63$으로 차이가 난다. 즉 클래스 불균형인 경우 정확도는 양이 더 많은 클래스에 더 의존적이게 되기 때문에 모델의 성능을 판별하기 어렵게 된다. 무시하고 모델을 적합한다고 하더라도 과적합 될 가능성이 높다. 앞서 분류 지표를 알아볼 때 정확도의 단점이 바로 이 경우에 해당한다.

GIGO(Garbage In, Garbage Out)라는 말이 있다. 좋은 모델을 만들려면 좋은 train set이 필요한 법이다. 많은 문제를 낳는 클래스 불균형은 샘플링을 통해 해결할 수 있다. 다양한 샘플링 방법들을 알아보자.

■ 샘플링

1) 언더 샘플링 (Under Sampling)

언더 샘플링은 소수 클래스는 그대로 두고, 다수 클래스의 데이터를 소수 클래스에 맞추어 감소시키는 방법이다.



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

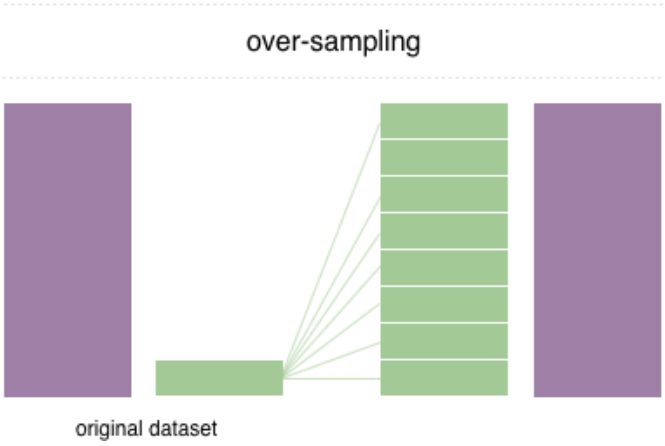
언더 샘플링은 Random Under Sampling, Condensed Nearest Neighbors, Tomek Links, Edited Nearest Neighbors, One-sided selection, Neighborhood Cleaning 등 다양한 방법이 있다. 간략하게 알아보자. (자세한 내용이 궁금하다면 따로 물어봐주세용~)

종류	특징
Random Under Sampling	랜덤으로 다수 클래스 데이터 제거
Condensed Nearest Neighbors (CNN)	- KNN 알고리즘 적용 - 다수 클래스 데이터가 밀집된 곳을 위주로 제거 - 어떤 데이터를 지울지는 아직 랜덤
Tomek Links	- 클래스 경계의 노이즈 데이터만 제거 - 주로 다른 기법과 결합해서 사용
Edited Nearest Neighbors (ENN)	
One-sided selection (OSS)	Tomek Links + CNN
Neighborhood Cleaning Rule (NCR)	ENN + CNN

각각 원리는 조금씩 다르지만, 다수 클래스의 데이터를 줄이는 방법이다. 다수 클래스 데이터의 경우, 샘플 수를 줄임으로써 메모리 사용 등의 문제를 해결할 수 있지만, 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.

2) 오버 샘플링 (Over Sampling)

오버 샘플링은 소수 클래스의 데이터를 다수 클래스에 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.

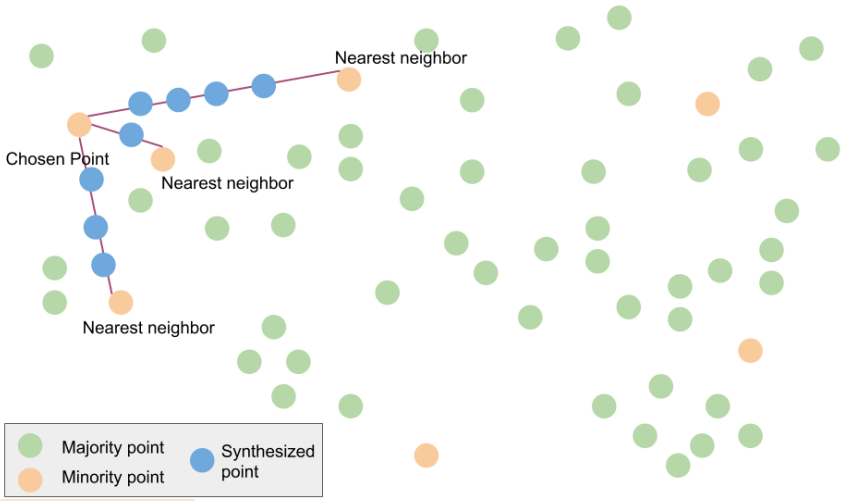
오버 샘플링 역시 다양한 방법이 있는데 우리는 Random Over Sampling과 SMOTE를 알아볼 예정이다.

- **Random Over Sampling**

  Random Over Sampling은 가장 기본적인 오버 샘플링이다. 무작위로 소수 클래스의 데이터를 복사함으로써 소수 클래스 데이터를 늘리는 방법이다. 정보의 손실은 없지만, 임의로 데이터를 복제하기 때문에 과적합(overfitting)될 가능성이 있다.

- **SMOTE** (Synthetic Minority Over Sampling Technique)

SMOTE는 실제로 많이 사용되는 방법으로, 데이터를 늘리는 알고리즘은 다음과 같다.



① 소수 클래스의 데이터 중 랜덤으로 하나를 선택한다.

② 선택한 데이터와 가장 가까운 k개의 소수 클래스의 데이터를 선택한다.

③ 처음에 선택한 데이터와 무작위로 선택한 k개의 데이터 사이의 직선을 그리고, 그 직선 상에 가상의 소수 클래스 데이터를 생성한다.

SMOTE는 단순 랜덤이 아닌 KNN 알고리즘을 사용해 가상의 데이터를 생성함으로써 소수 클래스의 데이터를 늘리기 때문에, Random Over Sampling보다 overfitting이 발생할 가능성이 줄어드는 효율적인 방법이다. 이는 SMOTE로 만든 데이터가 단순히 랜덤으로 생성한 데이터보다 좀더 소수 클래스의 공간에 가깝기 때문이다.

반면, SMOTE는 새로운 소수의 가상 데이터를 생성하는 과정에서 인접한 "다수 데이터의 위치"는 고려하지 않기 때문에 서로 다른 클래스의 데이터가 겹치거나 노이즈가 생성될 수 있다. 따라서 고차원 데이터에서는 효율적이지 않다는 단점이 있다. 이 문제를 해결하기 위해 SMOTE를 확장 및 수정해서 Borderline-SMOTE, SVMSMOTE, ADASYN 등의 방법들이 있다. 각각 원리는 조금씩 다르지만 기존 SMOTE와는 달리 노이즈를 고려한다는 점은 동일한 방법들이다. 간략하게만 살펴보자. (자세한 내용이 궁금하다면 언제든지 물어보세요!)

종류	특징
Borderline-SMOTE	- 클래스 경계 근처에만 소수클래스 데이터 생성 - 경계 설정에 KNN 알고리즘 사용
Borderline-SMOTE SVM (SVMSMOTE)	- 클래스 경계 근처에만 소수클래스 데이터 생성 - 경계 설정에 SVM 사용
ADASYN	- 소수클래스의 밀도분포를 고려해서 데이터 생성 - 밀도에 가중치를 두어 밀도가 낮으면 많이, 높으면 조금 생성

4. 인코딩 (Encoding)

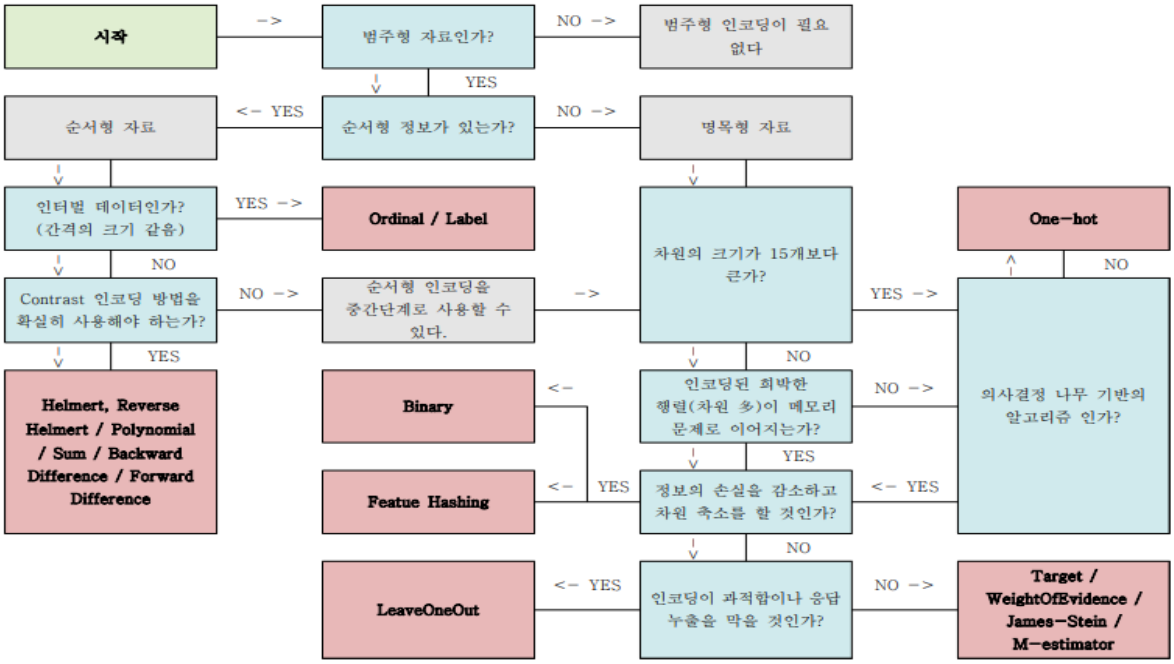
■ 인코딩이란?

데이터 분석에서 인코딩은 범주형 자료를 수치화 하는 것을 의미한다. 데이터 분석에서 무언가 모델화하고 학습을 시키기 위해서는 data를 모두 수치화해야 한다. 범주형 자료 역시 문자열 상태 그대로 모델에 넣을 수 없기 때문에 자료를 수치적인 값으로 인코딩해주어야 한다. 때문에 범주형 자료를 잘 다루기 위해서 인코딩이 필요한 것이다. 그렇다면 인코딩의 종류는 무엇이 있는지 알아보자.

■ 인코딩의 종류

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	

인코딩의 종류는 이처럼 상당히 많다. 항상 답이 정해진 것은 아니지만, 아래의 순서도와 같이 상황에 따라서 적절한 인코딩 방법을 선택해서 사용하면 된다. 이중에서 우리는 자주 쓰이는 인코딩 몇 가지만 배워볼 예정이다.



1) One-Hot Encoding (Dummy Encoding)

One-Hot Encoding은 **가변수**(Dummy variable)를 만들어주는 인코딩 방법이다. 따라서 Dummy Encoding으로 불리기도 한다. 회귀분석입문 수업을 들었다면, 질적변수가 있을 때 가변수를 만들어서 회귀모형을 만드는 법을 배웠을 것이다. 그때와 동일한 방법이 바로 One-Hot Encoding이다. 가장 흔하게 쓰이는 One-Hot Encoding이 어떻게 이루어지는지 아래 예시를 통해 알아보자. 기존의 변수는 "최애 블랙핑크 멤버"로, 제니,로제,리사,지수의 수준을 갖는다.

범주	제니(기준범주)	로제	리사	지수
제니	1	0	0	0
로제	0	1	0	0
리사	0	0	1	0
지수	0	0	0	1

범주 \ 가변수	로제	리사	지수
제니	0	0	0
로제	1	0	0
리사	0	1	0
지수	0	0	1

One-Hot Encoding에서는 가변수라는 새로운 변수들을 만들어 데이터에 추가한다. 가변수를 만들 때 자기 자신 범주에는 1, 그 외에는 0을 입력한 뒤 기준이 되는 범주의 열을 삭제한다. 이렇게 되면 J-1개의 가변수로 J개의 수준을 갖는 변수를 설명할 수 있게 된다. 모든 가변수가 0이면 기준 범주일 경우를 뜻하기 때문이다. 위의 예시로 보면, 어떤 관측치의 로제,리사,지수 변수 값이 (0,1,0)이면 최애 멤버가 리사이고 (0,0,0)이

면 제니를 의미한다는 뜻이다. 이처럼 3개의 변수를 만들어 4개의 범주를 표현할 수 있게 된다.

One-Hot Encoding은 여러 장점이 있어서 자주 쓰이는 인코딩 방법이다. 어떤 장점이 있는지 알아보자.

① 해석이 용이하다. 기준 범주의 정보가 intercept로 들어가 있기 때문에(모든 가변수가 0일 때 기준 범주이니까) 기준 범주를 기준으로 해석하면 쉽다.

② 명목형 변수 값들을 가장 잘 반영한다.

③ 다중공선성을 해결한다. 해당 범주만 1이고 나머지는 0으로 표현되기 때문에, 한 변수가 다른 변수들로 설명되는 것을 방지해주기 때문이다.

범주 \ 가변수	정연	모모	사나	지효	미나	다현	채영	쯔위
나연	0	0	0	0	0	0	0	0
정연	1	0	0	0	0	0	0	0
모모	0	1	0	0	0	0	0	0
사나	0	0	1	0	0	0	0	0
지효	0	0	0	1	0	0	0	0
미나	0	0	0	0	1	0	0	0
다현	0	0	0	0	0	1	0	0
채영	0	0	0	0	0	0	1	0
쯔위	0	0	0	0	0	0	0	1

하지만 위의 트와이스 예시처럼 범주형 변수의 Level이 너무 많거나 범주형 변수가 너무 많은 데이터의 경우 수많은 가변수가 생기기 때문에 차원이 늘어나는 문제가 발생한다. 이로 인해 학습속도가 느려지고 상당한 computer power가 요구되는 문제가 발생한다. (위처럼 광기의 가변수가 생겨나는 것이다…)

One-Hot Encoding은 회귀(Regression)와 분류(Classification) 문제 둘 다에 사용된다. 회귀문제에서는 J-1의 가변수를 사용하지만, 분류문제에서는 기준 범주를 지우지 않고 J개의 가변수를 그대로 사용한다.

2) Label Encoding

Label Encoding은 각 범주를 나누기 위해 단순히 점수를 할당하는 인코딩 방법이다. 명목형 자료에 많이 사용하는 방법으로, 말그대로 라벨링이기 때문에, 할당된 점수의 숫자에는 어떠한 의미나 순서, 연관성도 없다. 점수 할당은 임의대로 하면 되고, 꼭 1~N으로 부여할 필요는 없다.

Label Encoding은 One-Hot Encoding과는 다르게 차원이 늘어나지 않는 장점이 있지만, 할당된 숫자가 순서나 연관성이 있다고 학습이 일어나서 정보의 왜곡이 생길 수도 있다는 단점이 있다. 아래는 Label Encoding의 예시이다.

에스파	점수
카리나	1
윈터	2
지젤	3
닝닝	4

기존의 변수가 "에스파"이고 카리나, 윈터, 지젤, 닝닝의 수준을 가졌다면, Label Encoding을 통해 "에스파"라는 변수는 이제 1,2,3,4의 수치형 값을 가지는 것이다. 다만 이때 숫자 간의 순서나 연관성은 없다.

3) Ordinal Encoding

Ordinal Encoding은 순서형 변수에 대응하는 점수를 할당하는 인코딩 방법이다. 순서형 자료에 사용하는 방법으로, 점수를 할당할 때는 1부터 시작해서 순서가 있도록 부여한다. Label Encoding과는 달리 할당된 점수들은 순서나 연관성을 갖는다.

빡침 정도	점수
소	1
중	2
대	3
극대	4

기존 변수가 "빡침 정도"이고 소,중,대,극대의 수준을 가졌다면, Ordinal Encoding을 통해 "빡침정도" 변수는 이제 1,2,3,4의 수치형 값을 가진다. 대신 이때 숫자 간에는 순서와 연관성이 존재한다.

Ordinal Encoding은 Label Encoding처럼 차원이 늘어나지 않는다는 장점이 있지만, 범주 간의 순서를 표현할 때 정확한 간격을 반영하기가 쉽지 않다. 예를 들면 위 표에서 빡침 정도는 수치상 1이 차이가 나지만 그게 어느 정도의 차이인지는 애매하다는 것이다. "소에서 중"과 "대에서 극대"는 둘 다 1의 차이가 나더라도 전자는 3배 차이, 후자는 1000배 차이로 그 정도가 다를 수도 있다. 따라서 분석과제에 대한 도메인 지식이 중요해진다.

4) Mean Encoding (Target Encoding)

위의 세 encoding 방법에서는 Encoding된 숫자 자체는 별 의미가 없었다. 그저 임의로 숫자를 부여하여 "구분"하는 것 자체가 목적이었다. 하지만 **Mean Encoding**은 단순 구분을 넘어 반응변수 Y와 설명변수 X 간의 어떤 수치적인 관계를 반영하는 Encoding이다. 따라서 각 수준에 대하여 반응변수 Y(target)의 평균으로 점수를 할당하는 방식으로 Encoding을 한다. 앞으로 Target Encoding이라고 함은 반응변수를 고려하는 인코딩이란 뜻이다.

올림픽 양궁으로 예시를 들어보자. 반응변수 Y가 획득 점수이고 설명변수 X가 국가일때의 예시를 살펴보자. 범주형 변수인 국가에 대하여 Mean Encoding을 진행하여 각 국가별 양궁 점수 평균을 할당하였다.

[Y] 양궁 획득 점수	[X] 국가	[X] 국가 (Mean Encoding)
30	대한민국	30
30	대한민국	30
30	대한민국	30
28	러시아	28.333
29	러시아	28.333
28	러시아	28.333
28	이탈리아	27.333
27	이탈리아	27.333
27	이탈리아	27.333

즉 제일 첫번째 관측치는 학과 변수의 "대한민국"이라는 값 대신 30라는 대한민국의 평균으로 인코딩된 값으로 대체되는 것이다. 이런 식으로 설명변수 X의 문자열 값이 인코딩 되어 수치형으로 변하게 된다.

Mean Encoding은 위의 인코딩 방법들과는 다르게 할당된 점수가 임의의 숫자가 아닌 관계를 고려한 점수이므로 당위성이 존재한다는 장점이 있다. 또한 차원이 늘어나지 않는다는 장점도 존재한다.

평균은 이상치(Outlier)에 영향을 많이 받는다는 한계를 갖는다. 따라서 Mean Encoding 역시 이 한계점을 그대로 갖는다. 이상치가 있을 경우 인코딩을 할 때 반응변수의 관계가 제대로 반영되기 어렵다는 단점이 생긴다. 또한, 반응변수 Y에 대한 정보가 설명변수 X에도 들어가기 때문에 모델 학습 시 과적합될 가능성이 크다는 단점도 갖는다. 또한 Training set에 없던 새로운 범주가 Test set에 등장하면 점수를 할당할 수 없게 된다는 문제가 생긴다.

5) Leave One Out Encoding (LOO Encoding)

Leave One Out Encoding은 이상치의 영향을 줄이기 위한 인코딩으로, 현재 행을 제외하고(leave one out) 평균을 구한 뒤 이를 점수로 할당하는 인코딩 방식이다. 즉, 각 수준에 대하여 현재 행을 제외하고 반응변수의 평균을 구하는 방법이다. 평균을 구한다는 점에서 Mean Encoding과 유사하다.

아래 LOO Encoding의 올림픽 양궁 예시를 Mean Encoding과 비교하며 살펴보자.

[Y] 양궁 획득 점수	[X] 국가	[X] 국가 (Mean Encoding)	[X] 국가 (LOO Encoding)
30	대한민국	30	30
30	대한민국	30	30
30	대한민국	30	30
28	러시아	28.333	28.5
29	러시아	28.333	28
28	러시아	28.333	28.5
28	이탈리아	27.333	27
27	이탈리아	27.333	27.5
27	이탈리아	27.333	27.5

러시아의 경우를 보면, Mean Encoding에서는 전부 28.333으로 같았지만, LOO Encoding에서는 다른 값을 갖는다. 러시아의 Y값인 28,29,28 중에서 첫번째 관측치의 값인 28을 빼고 남은 29,28의 평균을 내서 28.5라는 LOO Encoding 결과가 나오게 된 것이다. LOO Encoding은 이런 원리로 이루어진다!

LOO Encoding은 Mean Encoding과 유사하기 때문에 장점도 동일하다. 이에 더해 이상치의 영향을 덜 받는다는 장점까지 갖는다. LOO는 Mean Encoding과 같이 반응변수 값을 사용해서 인코딩하는 target Encoding 방식이기 때문에 과적합이 발생할 가능성이 있지만, 하나를 빼고 평균을 구하는 방법 덕분에 반응변수의 정보가 완전 다 들어가지는 않아서 과적합의 정도나 가능성은 Mean Encoding보다는 적다. 과적합 가능성이 상대적으로 적다는 것과 이상치의 영향을 덜 받는다는 점 외에는 Mean Encoding과 단점도 동일하다.

6) Ordered Target Encoding (CatBoost Encoding)

Ordered Target Encoding은 현재 행 이전의 값들을 사용하여 평균을 구하고 이를 점수로 할당하는 인코딩 방식이다. 범주형 변수가 많을 때 사용하기 좋은 부스팅 모델인(ㄹㅇ 범주깡패,,,)CatBoost에서 사용되는 인코딩 방식이라 CatBoost Encoding이라고 불리기도 한다. 양궁 예시를 통해 Ordered Target Encoding이 어떤 원리로 이루어지는지 알아보자. 이번엔 이해를 위해 순서를 좀 섞고 추가를 해 보았다.

[Y] 양궁 획득 점수	[X] 국가	[X] 국가 (Mean Encoding)	[X] 국가 (CatBoost Encoding)
30	대한민국	30	28.2
28	러시아	27.5	28.2
28	이탈리아	27.333	28.2
30	대한민국	30	30
27	이탈리아	27.333	28
30	대한민국	30	30
29	러시아	27.5	28
28	러시아	27.5	28.5
27	이탈리아	27.333	27.5
25	러시아	27.5	28.333

앞서 말했듯이 Ordered Target Encoding은 현재 행 이전의 값들을 사용하여 평균을 구한다. 맨 아래 행인 러시아 관측치를 보면, 그 관측치보다 앞에 있는 "러시아"의 Y값 평균을 구한다. 즉 $\frac{28+29+28}{3} = 28.333$이 된다. 그 위의 이탈리아 관측치를 보면 자기보다 앞에 있는 "이탈리아"의 Y값 평균을 구한다. 즉 $\frac{28+27}{2} = 27.5$가 된다. 이런식으로 자기 앞에 있는 동포들의 양궁점수 평균을 구하면 되는 것이다! 만약 자기 앞에 아무도 없다면 전체 데이터의 평균을 사용하면 된다. 따라서 각 국가의 맨 처음 값은 전체 데이터의 평균인 $\frac{30+28+28+30+27+30+29+28+27+25}{10} = 28.2$가 된다.

이처럼 Ordered Target Encoding을 사용하면 같은 범주(국가)라고 해도 전혀 다른 인코딩 값을 가지게 된다. 따라서 앞의 두 Target Encoding과는 다르게 반응변수 Y에 대한 정보가 설명변수 X에도 들어가서 모델 학습 시 과적합될 가능성이 크다는 단점을 해결할 수 있다.

클린업 끝~~~~~~~

<이번주 실습과제>

주분 때 있을 패키지 대비 겸 주제분석 대비 겸 이번주 실습은 파이썬으로 해볼 예정입니다!

최대한 주석 많이 달아놨으니 파이썬이 낯설더라도 열심히 해보시길 바랍니다..ㅎㅎ

1. 인코딩 및 평가지표 실습

첫번째 실습 데이터는 그 유명한 캐글의 타이타닉 데이터입니다! 실습에서의 흐름 그대로 과제를 해오시면 됩니다!

0) 데이터 불러오기 및 전처리

- 변수 제거, 인코딩, 데이터셋 분할 (실습 때 했던 코드 그대로 사용!)

1) LGBM 모델로 train하기

-아래 코드 실행 후 학습하면 됨!

!pip install lightgbm (설치 안 돼있는 경우)

from lightgbm import LGBMClassifier

분류기이름 = LGBMClassifier(learning_rate=어쩌구, n_estimators=저쩌구)

- 성능을 좋게 만들어보고 싶다면 하이퍼 파라미터 수정가능!

(learning_rate와 n_estimators만 조정하기)

2) 혼동행렬 만들고 배웠던 평가지표 중 3개 골라서 계산해보기

3) ROC 그래프 그리고 AUC 값 계산하기

2. 샘플링 실습

두번째 실습 데이터는 지난 방학세미나 때 시용했던 데이터를 조금 수정한 데이터입니다! 이거는 그냥 실행해보면서 살펴만 보고 넘어가면 됩니다~

3. 인코딩 실습

마지막 실습 데이터는 지난 학기 주분 1주차 패키지 데이터를 조금 수정한 데이터입니다! 이것도 그냥 실행해보면서 살펴만 보고 넘어가면 됩니다~

클린업 진짜 끝~~~~

여러분 안녕하세요ㅎ 범주형 자료분석팀 팀장입니다. 클린업을 마치면서 간단한 소감과 느낀점을 작성해보려고 하는데요,,ㅎ 긴 글이지만 읽어주시면 감사하겠습니다(꾸벅)

무엇보다도 3주동안 잘 따라와준 우리 팀원들 너무 고맙습니다ㅜㅠㅜ 범주가 다소 생소하고 쉽지만은 않은 내용이었는데 개떡같이 설명해도 찰떡같이 이해해주는 팀원들을 보면서 너무 고마웠어요ㅠㅠ 저도 지난 학기 범주팀이어서 스터디를 들었지만 전 범주팀장 현 부회장 지연이의 명강의에도 불구하고 저의 딸리는 이해력으로 인해 고생을 좀 했었는데, 여러분들은 그렇지 않은 것 같아서 참 대단한 사람들이라고 느꼈습니다ㅎㅎ 이해도 빠른데 열정은 어찌나 넘치던지, 모르는 부분 있으면 질문도 열심히 하고 실습도 늦거나 빼먹는 사람 없이 열심히 하는 모습을 보면서 감동 받았습니다ㅠㅠ 실습한 것 보면 이해를 했는지 못했는지 보이는데 다들 완벽하게 이해한 것이 보여서 너무 좋았습니다. 이런 여러분들의 명석한 두뇌와 뜨거운 열정이라면 무엇이든지 쉽게 해낼 수 있을 거에요,,,ㅎㅎ 클린업 발표도 어찌나 잘하던지..! (아직 못해본 팀원도 있긴 하지만 분명 잘할 듯ㅋㅋ) 완벽한 내용 숙지를 바탕으로 간결하면서도 핵심 내용은 다 전달하고, 약간의 긴장감이 있으면서도 여유롭고, 차분하면서도 속사포 랩을 쏟아내는 여러분들을 보면서 감탄을 금치 못했었습니다,,, 이렇게 완벽한 팀원들을 만난 저는 참 행운아라고 생각하고, 이렇게 열정적인 여러분들을 보면서 저 역시 나태해지지 말고 더 열심히 해야겠다는 동기부여도 되었습니다!

3주동안의 클린업을 돌아보면 1주차에는 분할표, 연관성 측도 등 범주의 기본적인 이론들에 대해서 배웠고, 2주차 때는 앞서 배운 이론들을 적용해볼 만한 GLM 모형을, 3주차 때는 실제 분석에 써먹을 수 있는 실용적인 방법들을 배웠어요. 바이오통계입문이나 범주형자료분석 수업을 들으신 분들은 수월하거나 익숙했겠고, 듣지 않았다면 헷갈리고 낯설었을 거에요! 둘 중 어느 경우가 됐든 잘 따라와준 여러분 정말 고맙습니다 ㅎㅎ

팀장이 돼서 팀을 구상할 때 두가지 목표가 있었는데 첫번째는 서로서로 도우면서 으쌰으쌰해서 다같이 성장하는 그런 팀을 만들고 싶었어요. 누구 한 명 낙오되거나 소외되는 사람 없이 모두가 똘똘 뭉치는 그런 팀이요! 교학상장(敎學相長)이라는 사자성어가 있어요. 가르치고 배우면서 성장한다는 뜻인데, 팀장·팀원, 기존·신입 할 것 없이 모두가 서로에게 가르칠 수 있고 배울 수 있다고 생각해요. 그래서 우리가 서로 도우면서 성장하면 결국에는 모두가 훌륭한 엘리트로 발돋움하지 않을까?라는 망상이 있는데, 지금까지 여러분의 모습을 보면 저의 목표는 이미 따 놓은 당상이 아닌가라는 생각이 듭니닿ㅎㅎㅎ 우리 서로서로 도와가면서 최고의 팀을 만들어보자구요!

팀을 구상할 때 가졌던 두번째 목표는 귀여움입니다ㅋㅋㅋㅋㅋ 귀여운 사람들끼리 모여서 영차영차 서로서로 도우면 얼마나 귀여울까라는 생각이 들었었어요ㅋㅋㅋㅋ 칙칙한 팀이 되면 어쩌나 걱정했었는데 다행히 우리 팀원들은 한 명도 빠짐없이 귀여워서 너무 기분이 좋습니다 ㅎㅎㅎ 매번 세미나때마다 우리의 귀여움을 한껏 자랑하고 싶었어요. 우리 팀이 드러나는 유일한 시간이 발표시간이다 보니 여러분에게 귀여운 PPT를 강요하지 않았나 싶습니다,,ㅎㅎ 이런 억지 강요에 반발하거나 기분 나빴을 수도 있는데 이 부분은 미안하다고 말하고 싶네요,, ㅎㅎ 그래도 너무 잘 받아줘서 정말 고마웠습니다ㅠㅠ 앞으로도 우리의 귀여움 잔뜩 어필해보자구요! ㅎㅎ

주제분석부터는 우리의 귀여움 외에 능력에 초점을 좀더 두어서 보여줄 시간이에요! 그동안은 워밍업이었고 본격적인 프로젝트가 진행될 텐데, 주제 선정부터 분석까지 하나도 쉽지 않을거에요,, 중간 끝나고 과제가 쏟아지는 기간이라 더 바쁠거구요ㅠ 주제는 미리미리 생각해 놓는게 좋을거고 본인이 재밌다고 느끼는 분야를 주제로 생각해오는게 좋아요! (물론 선정이 안 될 수도 있지만ㅠㅠ) 힘든 과정이기 때문에 그나마 재미를 느껴야 할만해지거든요,,ㅎㅎ 앞으로도 서로서로 도와가면서 주제분석도 잘해봅시다! 저도 열심히 공부하고 배워서 여러분들 도와 주제분석 재밌게 해보려구요 ㅎㅎ 그럼 중간고사 잘 보시고 주제분석 때 보자구요 우리~~ 사랑합니다 범주팀💔