

비즈니스를 위한 데이터마이닝

가톨릭대학교 경영학과

이홍주

왜 평가해야 하는가?

- 분류나 예측을 위한 다양한 방법이 존재한다
- 각 방법마다 여러 설정 선택지가 있다
- 최적의 모델을 선택하기 위해서는 각 **모델의 성능**을 평가해야 한다

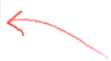
어떤 알고리즘으로
어떤 데이터로 학습해야 성능이 가장
좋은지 모름 (돌리기 전까지)
→ 여러 모형 실험

성능 평가

- 예측 성능 평가
 - 검증/테스트 데이터를 기반으로 모든 케이스에 대해 예측 정확도를 평가해 학습 데이터보다 좀 더 객관적인 근거 제공
- 지도 학습에서 관심 결과
 - 수치 값: 결과 변수가 수치형일 때(주택 가격)
 - 클래스 소속도 (범주 일치도): 결과 변수가 범주형일 때(구매자/비구매자)
→ 얼마나 정확?
 - 경향: 결과 변수가 범주형일 때(채무 불이행 경향) 클래스 소속도의 확률


수치 예측 정확성 판단 척도

- 우리가 알고자 하는 것은 모델이 학습한 데이터에 얼마나 잘 맞는지가 아니라, 새로운 데이터를 얼마나 잘 예측하는가이다
- 대부분의 측정 지표의 핵심 요소는 실제 y 와 예측된 \hat{y} 의 차이(“오차”, “손실”)이나

- 평균 절대 오차(Mean Absolute Error) $MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$ 
오차의 부호 반영 (음수 일수도)

- 평균 절대 백분율 오차(Mean Absolute Percentage Error)

$$MAPE = \frac{100}{n} \sum_{i=1}^n \left| \frac{y_i - \hat{y}_i}{y_i} \right|$$

- 근의 평균 제곱 오차(Root Mean Squared Error) $RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$ 

수치 예측 정확성 예제

회귀분석을 위해 데이터 불러오기

```
car_df = pd.read_csv('../data/ToyotaCorolla.csv')
```

독립변수와 종속변수 선정

```
excludeColumns = ('Price', 'Id', 'Model', 'Fuel_Type', 'Color')
```

```
predictors = [s for s in car_df.columns if s not in excludeColumns]
```

```
outcome = 'Price'
```

학습/검증 집합 분할

```
X = car_df[predictors]
```

```
y = car_df[outcome]
```

```
train_X, valid_X, train_y, valid_y = train_test_split(X, y, test_size=0.4, random_state=1)
```

회귀분석 모형 학습

```
reg = LinearRegression()
```

```
reg.fit(train_X, train_y)
```

성과 평가

학습 데이터

```
regressionSummary(train_y, reg.predict(train_X))
```

검증 데이터

```
regressionSummary(valid_y, reg.predict(valid_X))
```

수치 예측 정확성 예제

Regression statistics

학습데이터에 대한 성능 GOOD

Mean Error (ME) : 0.0000
Root Mean Squared Error (RMSE) : 1121.0606
Mean Absolute Error (MAE) : 811.6770
Mean Percentage Error (MPE) : -0.8630
Mean Absolute Percentage Error (MAPE) : 8.0054 %

Regression statistics

검증데이터에 대한 성능

Mean Error (ME) : 97.1891
Root Mean Squared Error (RMSE) : 1382.0352
Mean Absolute Error (MAE) : 880.1396
Mean Percentage Error (MPE) : 0.0138
Mean Absolute Percentage Error (MAPE) : 8.8744 %

학습과 검증 데이터의 성과 비교

- 학습 데이터셋에서 오차는 모델이 얼마나 잘 적합되었는지 알려줌
- 반면, 검증 데이터셋에서 오차(예측 오차)는 모델이 새로운 데이터를 예측하는 성능(예측 성능)을 측정
- 모델은 학습 데이터셋을 사용해 적합되었기 때문에 학습 오차가 검증 오차보다 작고, 복잡한 모델일수록 학습 데이터에 과적합될 가능성이 더 큼

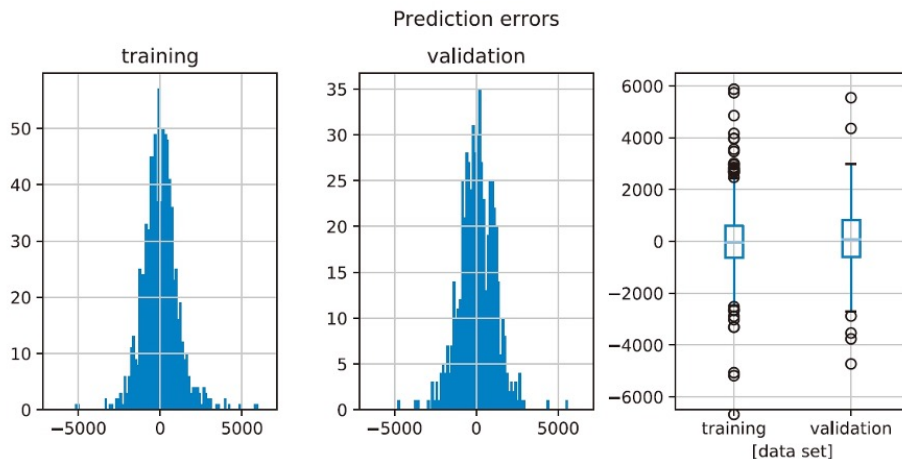


그림 5-1 학습 데이터셋과 검증 데이터셋에 대한 도요타 가격 예측 오차의 히스토그램과 박스 플롯

누적 이득 차트 Gains Chart

사실 잘 안그림

- 수치 예측 문제에서 관심은 “예측값이 높은 데이터가 실제로도 높은 값을 가지는가?” 임
- 따라서 예측값 \hat{y} ^{what (예측값~)} 기준으로 데이터를 정렬한 뒤, 구간(decile, quantile 등) ^{10개 4개} 별로 실제값 y 를 누적 합하거나 전체 합의 누적 비율을 계산해서 곡선을 그림
- 교재 예제) 실제 중고차 데이터와 예측 가격이 있다고 합시다.
- 예측 자동차 가격이 높은 순서대로 정렬 (x축)
- 실제 자동차 값의 누적 합을 계산하여 y축에 표시
- 기준선은 무작위 선택 시 기대되는 누적 합 (대각선) 임.

누적 이득 차트 Gains Chart

- 데이터 예제

	실제값 y	예측값 \hat{y}
1	100	95
2	80	70
3	60	65
4	40	30
5	20	25
6	10	15
7	5	10
8	2	5
9	1	3
10	0	1

누적 이득 차트 Gains Chart

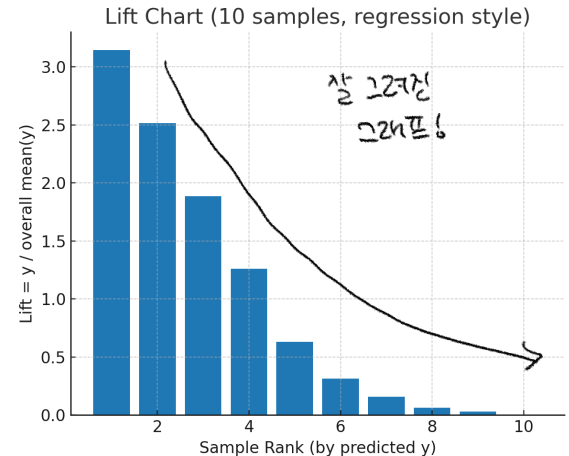
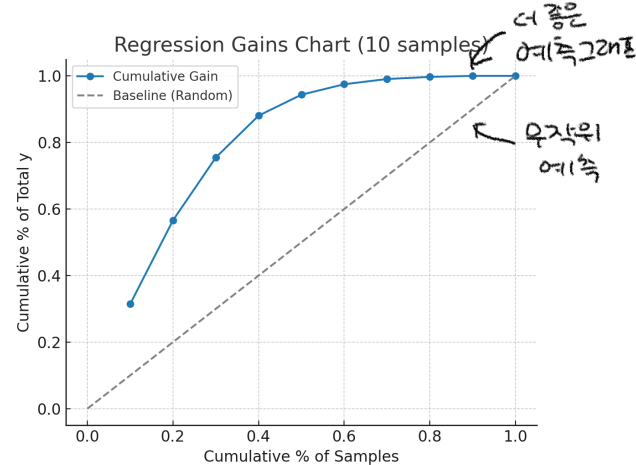
- cum_y : 누적된 실제값 합
- $\text{cum_y_pct} = \text{cum_y} / \text{전체 } y \text{ 합}$
- $\text{cum_n_pct} = \text{샘플 개수} / \text{전체 샘플 수}$

	실제값 y	예측값 $y \text{ hat}$	cum_y	cum_y_pct	cum_n_pct
1	100	95	100	0.314 <small>$(100 + 318)$</small>	0.1
2	80	70	180	0.566	0.2
3	60	65	240	0.755	0.3
4	40	30	280	0.881	0.4
5	20	25	300	0.943	0.5
6	10	15	310	0.975	0.6
7	5	10	315	0.991	0.7
8	2	5	317	0.997	0.8
9	1	3	318	1	0.9
10	0	1	318	1	1.0

누적 이득 차트 Gains Chart

- 대각선(기준선)은 무작위 선택일 때, 10% 샘플에서 10%의 실제값, 50% 샘플에서 50%의 실제값을 얻는 경우.
- **Gains Chart**: 예측값이 큰 순서로 정렬했을 때, 누적된 실제값의 비율(파란 곡선)이 무작위 기준선(점선)보다 훨씬 위에 있음
- **Lift Chart**: 각 샘플별 실제값을 전체 평균값과 비교한 비율을 막대로 표시. 상위 랭크일수록 Lift가 높아 모델이 “가치가 큰 항목”을 앞쪽에 배치했음을 확인할 수 있음
- 예제 데이터의 y 평균 31.8 ($318/10$)

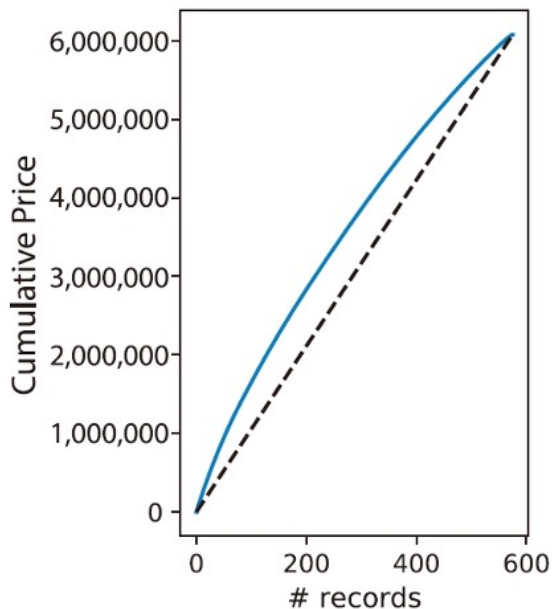
#1 data는 1000이네
나눠주면 31.8



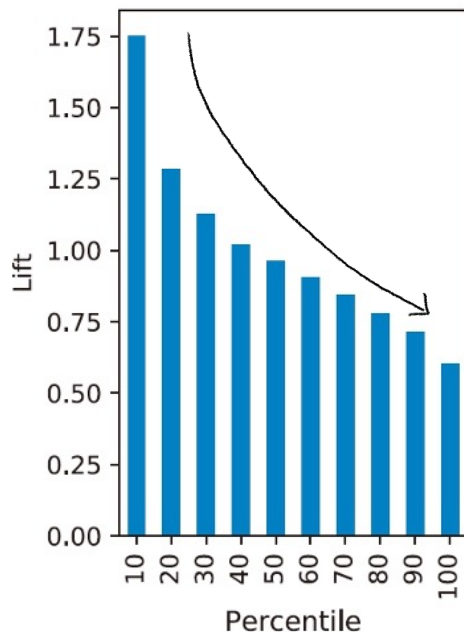
누적 이득 차트 및 십분위 리프트 차트

이 연관 규칙 - '하향상도' Lift

하나 / 여러 개 넣어도



(a) 누적 이득 차트



(b) 십분위 리프트 차트

그림 5-2 연속 결과 변수(도요타 자동차 판매량)에 대한 차트

경향성은 보여줌! - 보조용!
신체 지능은 어렵음

분류 성능 판단

예시. 고객이탈

- 단순 규칙(Naïve Rule)

A 60% 40% B
↓
그냥 100% A로 분류
(60%는 맞췄네..)

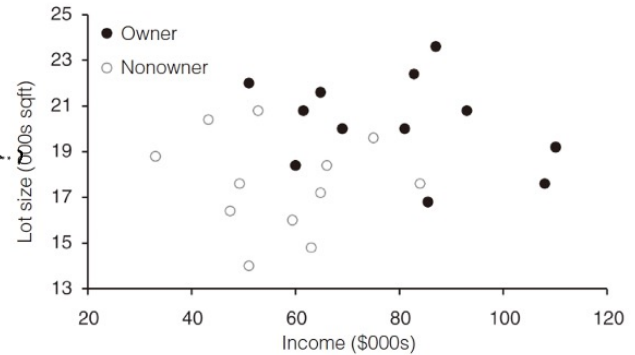
- 해당 레코드를 다수 클래스의 멤버로 분류

- 종종 벤치마크로 사용됨

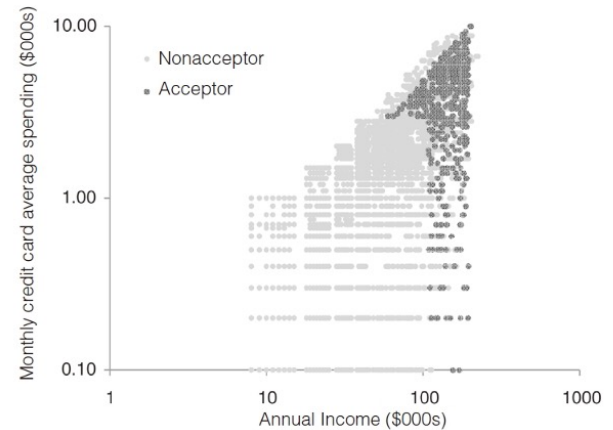
- 범주 분리

- 고수준 분리는 예측 변수를 사용해 오류가 낮음을 의미

- 저수준 분리는 단순 규칙서 크게 개선되지 않음을 의미



(a) 고수준의 분리



(b) 저수준의 분리

분류 성능 판단

- 정오 행렬(Confusion matrix, 혼동 행렬)
 - 특정 데이터셋에 대해 산출하는 정확한 분류와 부정확한 분류를 요약
 - 정오 행렬의 행과 열은 각각 예측 클래스와 진(실제)클래스에 대응

표 5-2 3,000개 레코드와 클래스에 기반한 정오 행렬

		예측된 클래스 0	예측된 클래스 1
2714	= 실제 클래스 0	2689	25
286	= 실제 클래스 1	85	201

바뀌어도 0
그러나
정확함

가로: 실제값
세로: 예측값

분류 성능 판단

- 정확성 척도

표 5-3 정오 행렬: 각 셀의 의미

		예측된 클래스	
		C_1	C_2
실제 클래스	C_1	$n_{1,1}$ = 정확히 분류된 C_1 레코드의 수	$n_{1,2}$ = C_2 로 잘못 분류된 C_1 레코드의 수
	C_2	$n_{2,1}$ = C_1 로 잘못 분류된 C_2 레코드의 수	$n_{2,2}$ = 정확히 분류된 C_2 레코드의 수

- 전체 오차율 = $(25+85)/3000 = 3.67\%$
- 정확도 accuracy = $1 - \text{오류} = (201+2689)/3000 = 96.33\%$

$$\text{정확성} = 1 - \text{err} = \frac{n_{1,1} + n_{2,2}}{n}$$

분류 성능 판단

- 분류의 경향과 컷오프 cut off
- 경향 propensities 은 레코드가 각 클래스(범주)에 속할 확률
- 예측된 클래스 소속도를 생성하거나 레코드들을 관심 클래스에 속할 확률로 순위를 정하기 위해 사용
- 2-클래스 분류기의 기본 컷오프 값은 0.50(다른 컷오프 값 사용 가능)
 - ≥ 0.50 이면 C1 클래스로 분류
 - < 0.50 이면 C2 클래스로 분류

보통 0.5임 0.5 이하

2진분리에서는

하나의 클래스 값만 구분

분류 성능 판단

Predicted Probability of Being an Owner

예측 확률				
ACTUAL	Pred. Prob.	ACTUAL	Pred. Prob.	
owner	0.9959	owner	0.5055	if cutoff is <u>0.5</u> , then 11 records classified as "owner"
owner	0.9875	nonowner	0.4713	
owner	0.9844	nonowner	0.3371	
owner	0.9804	owner	0.2179	
owner	0.9481	nonowner	0.1992	
owner	0.8892	nonowner	0.1494	
owner	0.8476	nonowner	0.0479	
nonowner	0.7628	nonowner	0.0383	
owner	0.7069	nonowner	0.0248	
owner	0.6807	nonowner	0.0218	
owner	0.6563	nonowner	0.0161	
nonowner	0.6224	nonowner	0.0031	

분류 성능 판단

Predicted Probability of Being an Owner

	ACTUAL	Pred. Prob.	ACTUAL	Pred. Prob.
	owner	0.9959	owner	0.5055
	owner	0.9875	nonowner	0.4713
	owner	0.9844	nonowner	0.3371
	owner	0.9804	owner	0.2179
	owner	0.9481	nonowner	0.1992
	owner	0.8892	nonowner	0.1494
	owner	0.8476	nonowner	0.0479
cut-off ~	nonowner	0.7628	nonowner	0.0383
if cutoff is 0.8, then 7	owner	0.7069	nonowner	0.0248
records classified as	owner	0.6807	nonowner	0.0218
"owner"	owner	0.6563	nonowner	0.0161
	nonowner	0.6224	nonowner	0.0031

분류 성능 판단

- 전체 범위의 컷오프 값들과 오분류율이나 정확성이 컷오프 값의 함수로 어떻게 변하는지 보기 위해 컷오프 값에 따라 관심 있는 성능 척도를 그려볼 수 있음

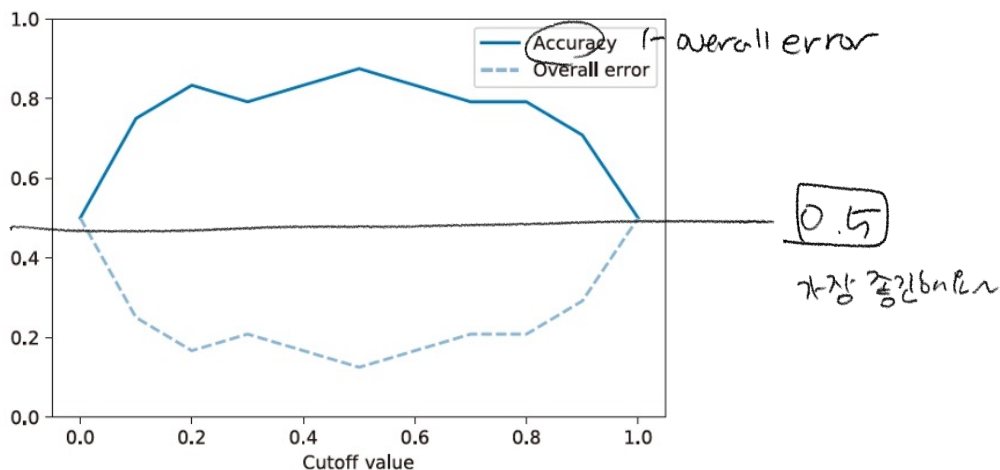


그림 5-4 컷오프 값의 함수로 그려진 정확성과 전체 오차(잔디깎이 기계 예)

분류 성능 판단

- 클래스의 중요성이 불균등한 경우의 성능
- 많은 경우에 한 클래스에 속하는 지를 판단하는 것이 더 중요한 경우가 있음
 - 탈세
 - 신용 부도
 - 프로모션에 대한 응답
 - 해킹 탐지
 - 항공기 지연 여부
- 전체 데이터에 대한 오차가 높더라도 중요한 클래스에 대한 판단이 바른 경우를 선호

Precision과 Recall

accuracy는 갖는게 중요X

- 클래스의 중요성이 불균등한 경우의 성능

- 예측된 클래스 C_2 이 양성 범주인 경우 (대출 가능 등)

- 정밀도 Precision: 양성 예측의 정확도 $\text{Precision} = \frac{TP}{TP + FP}$ ↑ 수를 crowd

- 재현률 Recall: 실제 양성인 데이터 중에서 예측이 양성이라고 된 데이터의 비율 $\text{Recall} = \frac{TP}{TP + FN}$

- 즉, 실제 양성인 데이터를 얼마나 찾아내었는 지를 뜻함

표 5-3 정오 행렬: 각 셀의 의미

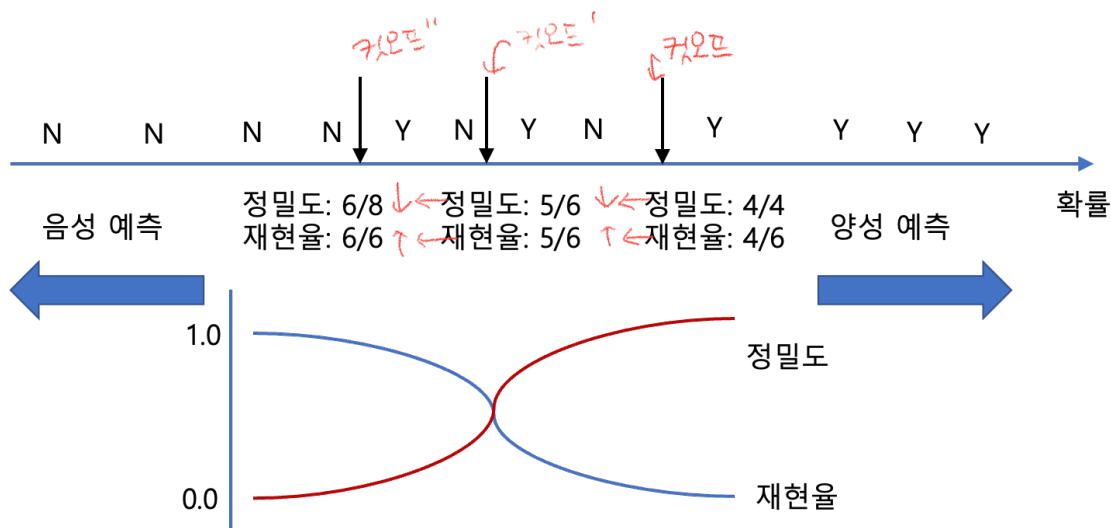
		예측된 클래스	
		C_1	C_2
실제 클래스	C_1	$n_{1,1}$ = 정확히 분류된 C_1 레코드의 수 TN	$n_{1,2}$ = C_2 로 잘못 분류된 C_1 레코드의 수 FP
	C_2	$n_{2,1}$ = C_1 로 잘못 분류된 C_2 레코드의 수 FN	$n_{2,2}$ = 정확히 분류된 C_2 레코드의 수 TP

Recall (Yellow circle around FN and TP)
 Precision (Green circle around FP and TP)

False Negative (under FN)
 False positive (under FP)
 True positive (under TP)

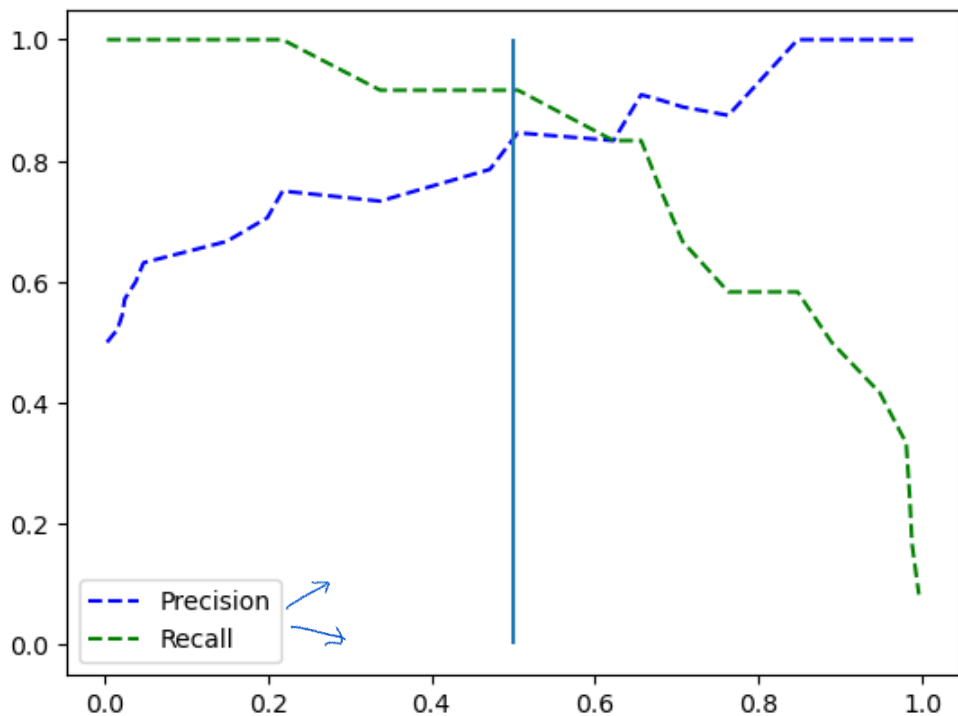
Precision과 Recall

- 정밀도와 재현율의 상충관계(trade off)
- 아래와 같이 대출승인에 대한 확률값에 따라 데이터를 정렬해 놓았다. 여러 가지 임계값 기준에 따라 양성, 음성을 분류하는 경우에 정밀도와 재현율 값이 변화하며, 서로 상충관계에 있음



Precision과 Recall

- 정밀도와 재현율의 상충관계(trade off)



F1 Score

산술평균 X

- 정밀도와 재현율의 조화평균(harmonic mean)을 통해 하나의 지표인 F1 점수 (F1 score) 고안

$$F_1 = \frac{2}{\frac{1}{\text{정밀도}} + \frac{1}{\text{재현율}}} = 2 \times \frac{\text{정밀도} \times \text{재현율}}{\text{정밀도} + \text{재현율}}$$

- Precision과 Recall은 상충(trade-off) 관계에 있기 때문에, 두 값의 균형을 잡아야 함.
- 단순 산술평균을 쓰면, 한쪽이 매우 낮더라도 다른 쪽이 높으면 평균이 꽤 높게 나오기에 균형을 잘 반영하지 못함.
- 조화평균은 두 값 중 하나라도 낮으면 전체 평균이 크게 낮아짐. 따라서 Precision과 Recall이 모두 높은 경우에만 F1 score가 높게 나옴

F1 Score

사이 매우 大

- Precision = 1.0, Recall = 0.1
- 산술평균: $(1.0 + 0.1) / 2 = 0.55$
- 조화평균(F1): $2 \times (1 \times 0.1) / (1 + 0.1) \approx 0.18$

F1 Score

- F1 점수 기준 모형 선정 ?

- 기본적으로는 정밀도와 재현율을 고려한 F1 점수 값이 높은 모형을 선택함. 정밀도와 재현율이 비슷한 모형의 F1 점수 값이 높게 나옴

- 정밀도가 중요한 경우

- 은행의 대출승인의 경우 대출을 갚을 고객을 찾는 것이 중요하다면, 재현율은 낮더라도 높은 정밀도로 대출을 갚을 고객을 찾는 것이 중요
- 대출을 갚을 고객 중 일부에게는 대출을 해주지 않더라도, 대출을 해준 고객은 대부분 대출을 갚게 됨

- 재현율이 중요한 경우

- 감시카메라로 좀도둑을 감지하는 분류모형을 만든다면 정확도는 아주 높지 않더라도 높은 재현율로 좀도둑을 분류하는 것이 중요함
- 경비원이 잘 못된 호출을 종종 받더라도 거의 모든 좀도둑을 잡을 수 있음

ROC (Receiver Operating Characteristics) 곡선

분류기가 얼마나 잘 분류하는지

- 클래스의 중요성이 불균등한 경우의 성능
- C1이 중요한 클래스인 경우
positive class
- 분류기의 민감도(Sensitivity, 또는 Recall)는 중요한 클래스의 멤버를 올바르게 알아내는 능력

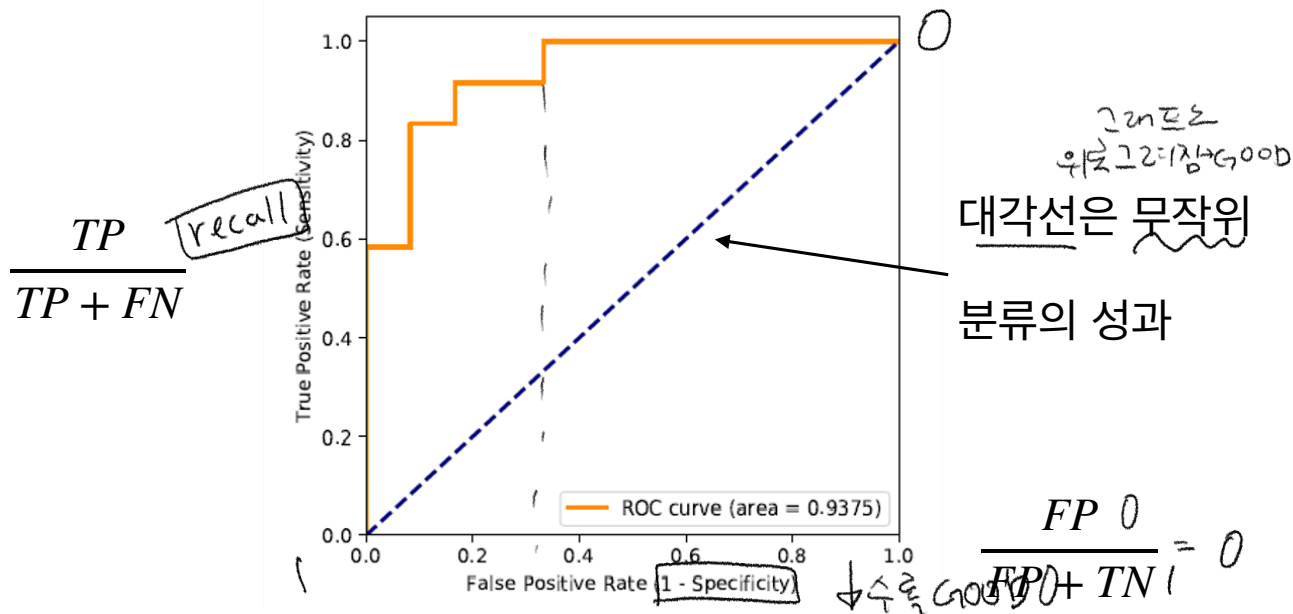
- 정확하게 분류된 C1 멤버의 비율인 $\frac{n_{1,1}}{n_{1,1} + n_{1,2}}$ 로 측정

- 분류기의 특이도(Specificity)는 C2 멤버를 정확하게 제외하는 능력

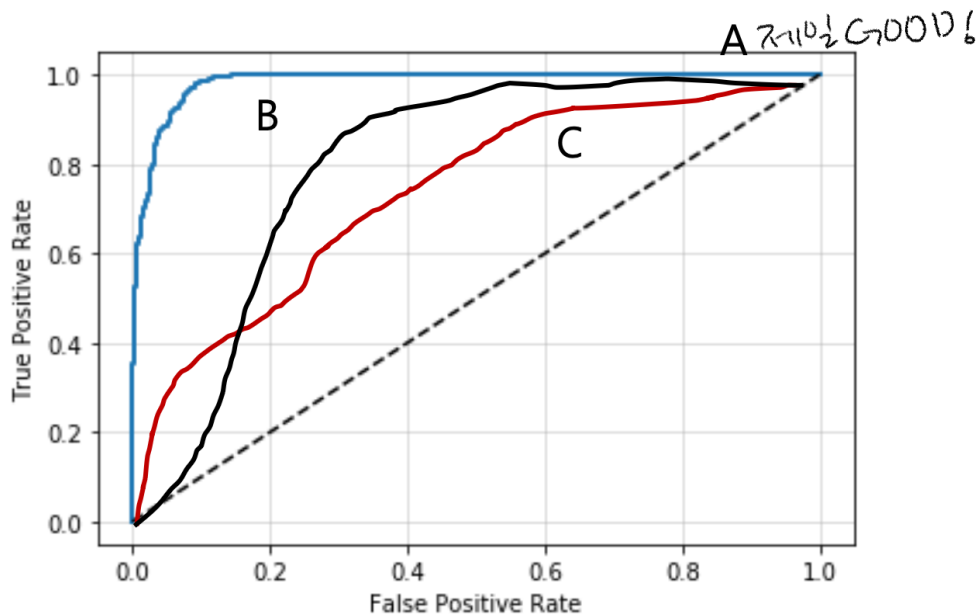
- 정확하게 분류된 C2 멤버의 비율인 $\frac{n_{2,2}}{n_{2,1} + n_{2,2}}$ 로 측정

ROC 곡선

- 척도들의 균형을 맞추는 컷오프 값을 찾기 위해 컷오프 값에 대한 이 척도들을 그려보는 게 유용
- ROC 곡선은 왼쪽 아래에서 시작해 컷오프 값을 1에서 0으로 줄이면서 {민감도, 1-특이도} 쌍을 그린 것



ROC 곡선



- ROC 곡선을 요약하는 척도는 '곡선 아래 영역(AUC)'인데, 1(클래스들 사이의 완벽한 구별)에서 0.5(임의 추측보다 좋지 않음) 사이의 값을 가짐

Area Under Curve

그래프상 해당 면적을
AUC로 비교 가능