5장 예측성능 평가

1. 서론

데이터 마이닝 방법들의 예측 성능을 어떻게 평가할 것인가?
↳ 과적합 되었는가? 성능척도 등.

- 예측을 위한 척도 : 평균오차, MAPE, RMSE 등
- 분류를 위한 척도 : 전체 정확도, 특이도, 민감도, 오분류 비용 등.

- 컷오프 (cut-off) 값의 선택.
- 컷오프값들에 대해 성능을 평가하는 ROC곡선

2. 예측성능의 평가.

예측정확도는 적합도와 똑같지 않다는 것에 유의. → 목적이 예측이기 때문!
↳ (goodness of fit; 모델이 학습데이터에 얼마나 적합한가?)

예측성능 평가에 사용되는 척도들은 검증데이터에 기반.
↳ 앞으로 예측할 데이터와 유사.

• 나이브 벤치마크 : 평균.

- 단순한 평균 이용.
- 학습데이터의 레코드 평균. → 최소한 이것보다는 예측성능이 우수해야한다.

◦ 예측 정확성 측도들.

레코드 i의 예측오차 = $e_i = y_i - \hat{y}_i$ ← 예측된 출력값.
↑ 실제 출력값

- MAE (절대평균 오차/편차) = $1/n \sum_{i=1}^{n} |e_i|$: 평균절대오차의 크기.
- 평균오차 = $1/n \sum_{i=1}^{n} e_i$: 음의 오차가 동일한 크기의 양의 오차를 상쇄시킬 수 있음. → 예측이 평균적으로 반응의 예측을 넘거는지 나타냄.
- MPE (평균 백분율 오차) = $100 \times 1/n \sum_{i=1}^{n} e_i/y_i$: 오차의 방향을 고려하여 예측이 실제 값에서 평균적으로 몇 퍼센트 정도를 벗어나는가.
- MAPE (절대평균 백분율 오차) = $100 \times 1/n \sum_{i=1}^{n} |e_i/y_i|$: 방향 없이 벗어나는 정도만 표시.
- RMSE (근의 제곱평균오차) = $\sqrt{1/n \sum_{i=1}^{n} e_i^2}$: 선형 회귀분석에서 추정값의 표준오차와 유사. → 학습데이터 대신 검증데이터 사용. 출력변수와 동일단위.

이러한 척도들은 이상치 (아웃라이어등)의 영향을 받음.
↳ 영향을 줄이기 위해 중간값 기반 척도들을 계산하거나 히스토그램/박스플롯 이용.

박스플롯/히스토그램 등의 분포도는 매우 유용.

◦ 학습과 검증 성능의 비교.

- 학습데이터에 기반한 오차 ⇒ 모델이 얼마나 적합한가.
- 검증데이터에 기반한 오차 ⇒ 모델이 얼마나 예측을 잘할 것인가

↳ 당연히 학습오차가 더 작을 수 밖에 없음. 둘 사이의 격차가 유의미하게 크다면

과적합 의심.

⇒ 학습과 검증 세트 사이의 오차 비교 필요 (수치 / 도식)

↳ 어디에서 기인한 차이인지를 확인 (검증셋의 이상치 / 과적합 등)

• 향상 차트 (Lift Chart)

새로운 레코드의 집합 중 누적 예측값이 가장 높은 부분집합을 찾는 것이 목적일 경우 사용.

모델의 예측성능을 예측변수들을 사용하지않는 기준모델과 비교.

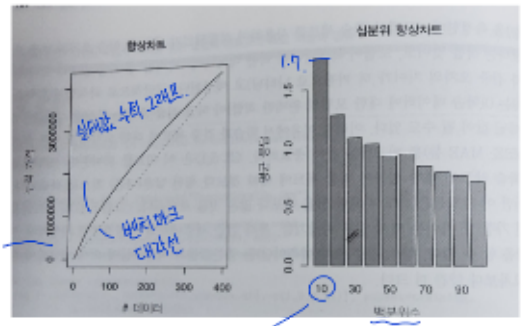
연속 반응값에 대한 향상차트는 누적 예측값이 가장 큰 레코드 집합을 찾을 때만 적절.

↳ 새로운 레코드 각각에 대한 결과를 예측하는 데는 적절하지 않음.

관심있는 레코드의 집합 (보통 검증데이터)을 예측값에 대해서 높은 것부터 낮은 순으로 배열.

그 후 실제값들을 누적시키고 누적된 레코드의 수 (x축)에 대한 함수로서 y축 상에 누적값을 그림.

이 곡선은 각 레코드에 대한 단순한 기준예측 (y)을 배정하고 이러한 평균값을 누적시켜서 대각선이 되는 직선과 비교.



서로 멀리 떨어질수록 낮은값과 높은 값을 더 잘 분리하는 모델.

예측된 판매를 가장 높게 하는 상위 10%를 선택하는 것이 임의의 10%보다 1.7배 더 높은 수익성.

400대의 10% ⇒ 40대.

임의의 40대 (400대를 모두 누적한 후 ÷ 10) 과 향상곡선에서 x=40 의 값 비교를 해도 알 수 있음.

3. 분류기 성능의 판정.

성능척도가 필요한 이유 : 분류기와 예측방법들이 다양하게 선택될 수 있기 때문.

↳ 여러 개의 다양한 방법들, 같은 방법이라도 선택에 따라 다양한 결과.

기본적인 기준: 오분류 오차 – 다른 클래스에 속하는 걸 잘못 분류한 경우.

◦ 벤치마크 : 나이브 규칙.

예측 변수 정보를 무시하고 가장 지배적인 클래스에 속한다고 분류.

나이브 규칙보다 더 좋은 성능을 내야함.

↳ 나이브 규칙과 비교해서 얼마나 더 좋은 성능을 내는지 측정하는 척도 다양: 다음 R's

• 클래스 분리.

예측변수들이 잘 선별되지 않으면 큰 데이터 셋이 있어도 큰 도움이 되지 않음.

↳ 분류를 할 수 있는 정보가 충분하지 않으면.

예측변수 정보가 분류를 잘해낸다면 적은 데이터 셋으로도 좋은 분류기를 찾을 수 있음.

· 정오(분류) 행렬.

정오행렬 (Confusion Matrix) - 분류기가 특정 데이터셋에 대해 산출하는 정확한 분류와 부정확한 분류를 요약.

Confusion Matrix의 행과 열은 각각 예측클래스와 실제 클래스를 나타냄.

표 5.2 3000개 레코드와 두 클래스에 기반한 정오행렬

		실제 클래스	
		0	1
예측된 클래스	0	2689	85
	1	25	201

85, 25 → 오분류의 수.

2689, 201 → 올바른 분류의 수

정오행렬은 진분류의 추정값과 오분류율을 보여줌.

· 검증 데이터의 사용.

분류오차의 정확한 추정값을 얻기 위해 검증 데이터로 부터 정오행렬을 구함.

↳ 학습데이터로부터 구하면 과적합의 위험성 때문에 정확한 추정값을 얻기에는 무리.

학습데이터의 정오행렬과 검증데이터의 정오행렬을 비교하여(성능차이) 과적합의 가능성을 알아낼 수 있음.

· 정확성 측도.

정오행렬 각 셀의 의미 (클래스 C_1, C_2 2개)

표 5.3 정오행렬: 각 셀의 의미

	실제 클래스	
예측된 클래스	C_1	C_2
C_1	$n_{1,1}$ = 올바르게 분류된 C_1 레코드의 수	$n_{2,1}$ = C_1로 잘못 분류된 C_2 레코드의 수
C_2	$n_{1,2}$ = C_2로 잘못 분류된 C_1 레코드의 수	$n_{2,2}$ = 올바르게 분류된 C_2 레코드의 수

추정된 오분류율 (전체 오차율) $err = \frac{n_{1,2} + n_{2,1}}{n}$

정확도 (전체 정확도) $Accuracy = 1 - err = \frac{n_{1,1} + n_{2,2}}{n}$

· 분류의 경향과 컷오프.

분류의 첫 단계 - 레코드가 각 클래스에 속할 확률을 구하는 것.

↳ 경향 (propensities)라고도 함.

경향은 보통 예측된 클래스 소속도(분류)를 생성하거나 관심있는 클래스에 속할 확률에 의해서 레코드들을 순위-배열 하기 위한 중간단계로 사용.

관심있는 클래스에 속할 확률 > 컷오프 (cut-off) 값 ⇒ 그 클래스에 배정.

클래스가 2개인 2-클래스의 경우 컷오프의 기본값은 0.5.

↳ 더 크거나 작은 컷오프값도 사용가능 ⇒ 2가지 경우 모두 오분류율은 증가.

컷오프값의 함수에 따라 오분류율이 어떻게 변화하는지 알아볼수 도 있음.

⇒ 오분류의 비용에 따라, 분류의 기준에 따라 컷오프값은 변할 수 있음.
- 비용이 적은 경우 더 높은 오분류를 허용.
- 특정 항목에 대한 분류가 더 중요할 때.

⇒ 이런 조정은 모델선택 이후에 이루어짐. 모델변경X!

◦ 클래스 중요성이 불균등한 경우의 성능.

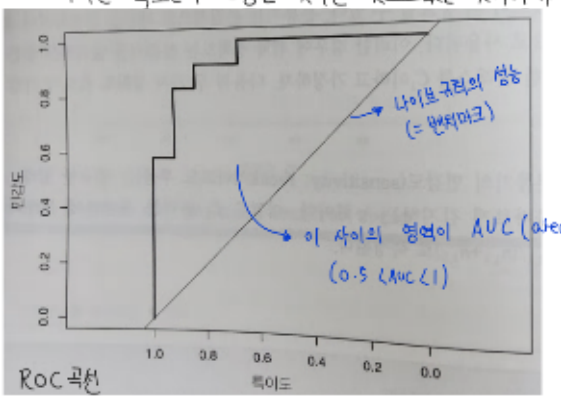
C_1, C_2 2개의 클래스가 있을 때, C_1에 속하는 것을 분류하는 일이 C_2보다 중요할 때.

2가지의 정확성 측도.

민감도 (Sensitivity recall) : C_1에 속하는 레코드를 정확하게 분류하는 비율 $(n_{1,1}/(n_{1,1}+n_{1,2}))$

특이도 (specificity) : C_2 레코드를 올바르게 제외하는 능력 $(n_{2,2}/(n_{2,1}+n_{2,2}))$

↳ 이러한 척도들의 균형을 맞추는 컷오프값을 찾기위해서 측도들 대 컷오프값을 그려보면 유용.



ROC곡선

◦ 비대칭적 오분류 비용.

다른 클래스보다 한 클래스에 속하는 레코드를 오분류하는 오차가 더 중요할 때.

ex) 위험한 상황인데 아니라고 분류하는 경우. (비용이 다르다.)

⇒ 올바른 분류를 내리는 비용은 0. 결국 오분류에 대한 비용만 생각. (비용을 생각하는 것이 중요하다)

비용을 포함한 일반적인 성능척도 – 평균 오분류 비용.

C_1의 레코드를 C_2로 오분류했을때의 비용: q_1

C_2의 레코드를 C_1로 오분류했을때의 비용: q_2

⇒ $\frac{q_1 n_{1,2} + q_2 n_{2,1}}{n}$ → 이 값을 최소화하는 분류기.

$$= \frac{q_1 n_{1,2} + q_2 n_{2,1}}{n} = \frac{n_{1,2}}{n_{1,2}+n_{1,1}} \times \frac{n_{1,1}+n_{1,2}}{n} q_1 + \frac{n_{2,1}}{n_{2,1}+n_{2,2}} \times \frac{n_{2,1}+n_{1,2}}{n} q_2$$

q_1으로 나누면 각 값들에 상관없이 q_2/q_1에 의존한다는 것을 알 수 있음.

실제 q_1, q_2 비용을 측정하는 것은 어렵지만 비율을 추정하기는 쉬움.

미래의 실제 비율 $P(C_1), P(C_2)$ 반영하면 샘플이 왜곡되어도 평균 오분류 비용 추정 가능.

⇒ $\frac{n_{1,2}}{n_{1,1}+n_{1,2}} P(C_1) q_1 + \frac{n_{2,1}}{n_{2,1}+n_{2,2}} P(C_2) q_2$

◦ 두 개이상 클래스로의 일반화.

클래스가 m개일 때, 정오행렬은 m개의 행, 열을 가짐.
↳ 오분류 행렬을 구하는 것이 힘들어질뿐 나머지는 동일.

4. 랭킹 성능의 평가.

레코드 집합 중 관심있는 클래스에 속할 가능성이 가장 큰 것을 탐지하는 예측과제

- 이진데이터에 대한 향상차트.

  클래스 소속도의 점함에 따른 랭크순서를 얻는 것이 중요.

  - 경향에 따른 정렬.

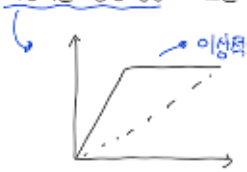
    레코드의 집합을 경향에 따라 내림차순으로 정렬. (관심있는 클래스에 속할 경향에 따라)

    그 후 실제 관심클래스에 속하는 누적수를 계산.

    향상차트 생성.

  - 향상차트의 해석.

    이상적인 랭킹 성능 - 모든 1 (관심클래스에 실제로 속하는 것)을 앞쪽에 위치. 모든 0을 뒤에.

  - 십분위 향상차트.

    항상 10개의 정보를 등으로 잡게.

    그 후 차트에 임의의 10개보다 얼마나 배정을 잘하는지 보여줌.

- 2-클래스를 넘어서.

  향상차트는 2-클래스에만 사용가능 → 하나의 중요클래스와 중요하지 않은 클래스로 축소필요.

  - 비용과 이득을 포함한 향상차트.

    올바른 분류와 오분류의 이득/비용이 알려지거나 추정가능하면 향상 차트는 유용한 (시각화 도구 / 점수)

    각 레코드에 특정 클래스에 속할 경향을 배정하는 분류기가 필요.
    그 후
    ① 레코드를 성공의 예측확률순으로 정렬 (관심있는 클래스에 속할 확률순)

    ② 각 레코드에 대해서 실제 출력과 연관된 비용(이득)을 기록.

    ③ 경향이 가장 큰 (첫 번째) 레코드에 대해서 x축 값은 1이고, y값은 비용(이득) ← 여기에서 계산된

    ④ 다음 레코드에 대해서 실제 출력과 연관된 비용(이득)을 다시 계산.
    그리고 그 값을 이전 레코드의 비용에다가 더함. → y값. x값은 2.

⑤ 모든 레코드에 대해 반복.

⑥ 이것에 대한 참조선(= 벤치마크)는 원점에서 X=0, Y=1을 이득을 잇는 직선.

- 컷오프 값의 함수로서의 함상

목적에 따라 레코드 수 / 컷오프 값 에 대한 함상 선택.

레코드 수: 특정 기준에 근거한 상위 레코드를 선택하는 정도

컷오프 값: 두 클래스를 잘 구별하는 컷오프값의 선택.

5. 확대 샘플링.

클래스가 매우 불균등한 비율일 경우 ⇒ 단순 랜덤 샘플링을 하면 드문 클래스가 너무 적게 선택되어 유용한 정보를 얻지못함.

↳ 층화 샘플링을 통해 희귀클래스로 부터 확대 샘플링 가능 ⇒ 분류기 성능 향상.

확대 샘플링 가중치를 통해 혼동행렬 (Confusion Matrix) 조정 필요.