노트북: 첫 번째 노트북
만든 날짜: 2021-01-03 오후 7:42
URL: https://continuous-development.tistory.com/171?category=736685

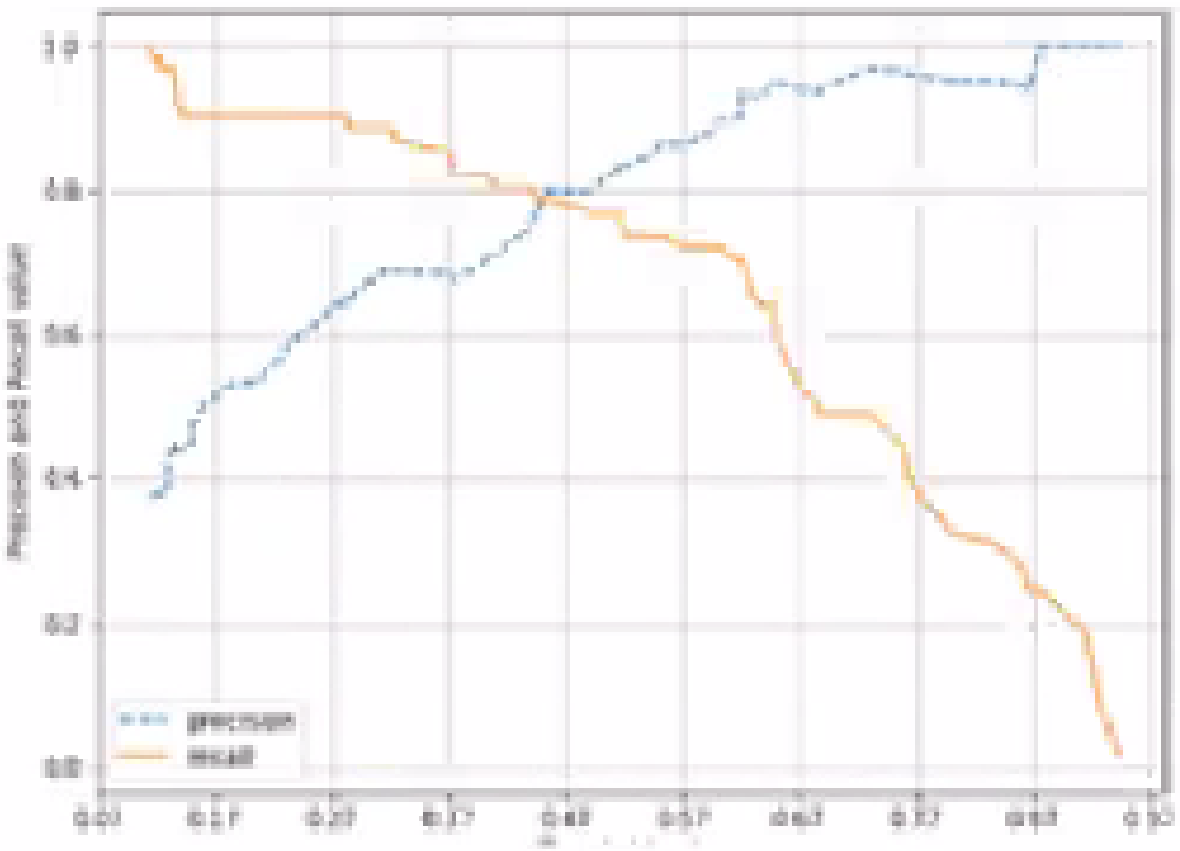
ML,DL

[ML/DL] 정밀도와 재현율의 트레이드 오프 정의와 구현

2020. 11. 2. 08:47 수정 삭제 공개

정밀도 / 재현율 트레이드오프

분류하려는 업무의 특성상 정밀도 또는 재현율이 특별히 강조돼야 할 경우 분류의 결정 임계값을 조정해 정밀도 또는 재현율의 수치를 높일 수 있다.



두 개는 상호 보완적인 평가지표여서 하나가 오르면 다른 하나가 떨어지기 쉽다. 이걸 트레이드오프라고 한다.

임계값이 낮아질수록 positive로 예측할 확률이 높아짐 - 재현율 증가

predict_probal() 메서드는 분류 결정 예측 확률을 반환한다.

precision / recall tradeOff

- predict_proba(): 예측 레이블의 확률을 반환해 주는 함수

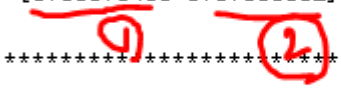
```
pred_pro_result = lr_model.predict_proba(X_test)

pred_pro_result
```

```
array([[0.88292346, 0.11707654],
       [0.84599256, 0.15400744],
       [0.86140115, 0.13859885],
       [0.08019498, 0.91980502],
       [0.13863637, 0.86136363],
       [0.84276519, 0.15723481],
       [0.85386641, 0.14613359],
       [0.82675835, 0.17324165],
       [0.87588497, 0.12411503],
       [0.85099024, 0.14900976],
       [0.75132045, 0.24867955],
       [0.14873202, 0.85126798],
       [0.36861588, 0.63138412],
       [0.58115351, 0.41884649],
       [0.76858664, 0.23141336],
       [0.7785993 , 0.2214007 ],
       [0.9577885 , 0.0422115 ],
       [0.94309   , 0.05691   ],
       [0.43288502, 0.56711498],
```

분류를 하는 데 있어서 0인지 1인지를 결정하는 비율을 나타낸다. 두 값 중 높은 쪽의 값을 1로 정의한다.

```
print('shape',pred_pro_result.shape)

print('result\n', pred_pro_result[:4])


print()

print("*"*50)

print()

y_pred = lr_model.predict(X_test)

print(y_pred)
```

```
shape (179, 2)
result
 [[0.88292346 0.11707654]
 [0.84599256 0.15400744]
 [0.86140115 0.13859885]
 [0.08019498 0.91980502]]
```



```
**********************************************

[0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 1 0 0 0 1 0 0 0 0 1 0 1 1 0 1 0 1 0 1
 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 1 1 0 0 1 1 1 0 0 0 0 0 1 0 0 0 0
 1 0 0 0 0 0 0 0 0 1 1 0 1 1 0 1 0 0 0 0 1 1 1 0 1 0 0 0 0 0 1 1 0 0 1 1 1
 0 1 0 0 0 0 1 1 0 1 0 1 1 0 1 1 0 0 0 1 0 0 1 1 1 0 1 1 0 1 0 0 0 0 0 0 0
 0 0 0 0 1 0 0 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 1]
```

1번 쪽이 negative

2번 쪽이 positive이다.

```
result = np.concatenate([pred_pro_result, y_pred.reshape(-1,1)],axis=1)

print('확률에 따른 예측 결과\n',result[:5])
```

```
확률에 따른 예측 결과
 [[0.88292346 0.11707654 0.        ]
 [0.84599256 0.15400744 0.        ]
 [0.86140115 0.13859885 0.        ]
 [0.08019498 0.91980502 1.        ]
 [0.13863637 0.86136363 1.        ]]
```

확률과 그에 따른 예측 결과를 0,1로 나타낸다. 왼쪽이 높으면 0 오른쪽이 높으면 1을 나타낸다.

- Binarizer 클래스 fit_transform()

Binarizer는 이항 변수 변환으로 연속형 변수를 기준으로 0과 1을 결정하는 값을 가지는 변수로 만든다.

```
user_threshold = 0.5

pred_pro_result[:,1].reshape(-1,1)
```

```
array([[0.11707654],
       [0.15400744],
       [0.13859885],
       [0.91980502],
       [0.86136363],
       [0.15723481],
       [0.14613359],
       [0.17324165],
       [0.12411503],
       [0.14900976],
       [0.24867955],
       [0.85126798],
       [0.63138412],
       [0.41884649],
       [0.23141336],
       [0.2214007 ],
       [0.0422115 ],
       [0.05691   ],
```

```
from sklarn.preprocessing import Binarizer


- threshold를 낮추면 재현율은 올라가고, 정밀도는 떨어진다.


user_threshold = 0.5

positive_pred_proba = pred_pro_result[:,1].reshape(-1,1)

user_predict = Binarizer(threshold=user_threshold).fit(positive_pred_proba).transform(positive_pred_proba)

display_eval(y_test, user_predict)
```

여기서 보면 Binarizer함수를 사용해서 prositive_pred_proba를 fit 하고 있다.

그 속에서 threshold에 따라 0과 1을 결정하는 **이항 변수화 기준선(threshold)을** 사용해 transform을 하는 것을 볼 수 있다.

```
accuracy 0.8324022346368715
precision 0.7758620689655172
recall 0.7258064516129032
```

threshold 값에 바뀌는 것을 볼 수 있다.

```
user_threshold = 0.2

positive_pred_proba = pred_pro_result[:,1].reshape(-1,1)

user_predict = Binarizer(threshold=user_threshold).fit(positive_pred_proba).transform(positive_pred_proba)

display_eval(y_test, user_predict)
```

```
accuracy 0.7318435754189944
precision 0.5729166666666666
recall 0.8870967741935484
```

threshold를 낮추면 재현율은 올라가고, 정밀도는 떨어진다.

- precision_recall_curve(정답, 예측 확률 값)

=> 정밀도, 재현율 값을 리턴 시켜준다.

```
from sklearn.metrics import precision_recall_curve


# 레이블 값이 1일때의 예측확률을 추출
pred_positive_label = lr_model.predict_proba(X_test)[:,1]
# print(pred_positive_label)
precisons, recalls,thresholds = precision_recall_curve(y_test,pred_positive_label)
print('precisons\n:', precisons)
print('recalls:\n', recalls)
print('thresholds:\n', thresholds)
```

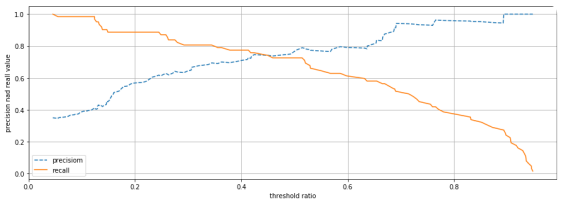
```
precisons
: [0.35028249 0.34659091 0.34857143 0.35057471 0.35260116 0.35465116
 0.35672515 0.35882353 0.36094675 0.36309524 0.36526946 0.36746988
 0.36969697 0.37195122 0.37423313 0.37654321 0.37888199 0.38125
 0.3836478  0.38607595 0.38853503 0.39102564 0.39354839 0.3961039
```

```
recalls:
 [1.         0.98387097 0.98387097 0.98387097 0.98387097 0.98387097
 0.98387097 0.98387097 0.98387097 0.98387097 0.98387097 0.98387097
 0.98387097 0.98387097 0.98387097 0.98387097 0.98387097 0.98387097
 0.98387097 0.98387097 0.98387097 0.98387097 0.98387097 0.98387097
```

```
thresholds:
 [0.04551763 0.05506945 0.05533097 0.05691    0.05922296 0.06616305
 0.07084287 0.07395733 0.07535051 0.07606252 0.07730531 0.07893212
 0.08634934 0.08720876 0.09332729 0.09439687 0.09530762 0.09573983
 0.09742478 0.09912378 0.10017056 0.10277881 0.1107047  0.11201496
 0.11707654 0.11967311 0.12034529 0.12180015 0.1225947  0.12411503
```

- 시각화 (정밀도, 재현율이 임계값 변화에 따른 시각화)

```
import matplotlib.pyplot as plt
%matplotlib inline



precisions, recalls, thresholds = precision_recall_curve(y_test, pred_positive_label)


plt.figure(figsize=(15,5))


plt.plot(thresholds,precisions[0:thresholds.shape[0]],linestyle='--', label='precisiom')
plt.plot(thresholds,recalls[0:thresholds.shape[0]],label='recall')
plt.xlabel('threshold ratio')
plt.ylabel('precision nad reall value')
plt.legend()
plt.grid()
plt.show()
```



'ML,DL' 카테고리의 다른 글□

[ML/DL] DecisionTree 구현 및 hyper parameter 설정□

[ML/DL] python 으로 구현하는 ROC곡선과 AUC□

[ML/DL] 정밀도와 재현율의 트레이드 오프 정의와 구현□

[ML/DL] python 을 통한 분류(classification) 성능평가지표 사용법(Accuracy,Pre...

[ML/DL] python 을 통한 교차검증 (k -Fold , stratifiedkFold)□

[ML/DL] python 을 통한 결측값 확인 및 결측치 처리 방법□

이항분포 기준선 정밀도와 재현율의 트레이드오프 트레이드 오프

나아무늘보

혼자 끄적끄적하는 블로그 입니다.