Bitamin 6주차 세션

part.2

10기 4조 노지예 부도현 임청수 한세림

로지스틱 회귀분석

다 중 분 류

이진 분류와 다중 분류

이진 분류

- 클래스가 2개인 분류 문제
- 주로 Sigmoid 함수를 이용하여 z값을 확률로 변환.
- ex. 도미/빙어 중 하나로 분류 (class 2개: 도미-1, 빙어-0)
- ex. 유방암 환자 여부 진단 (class 2개: 암 환자-1, 암 환자 X-0)

다중 분류

- 클래스가 2개 이상인 분류 문제
- 주로 Softmax 함수를 이용하여 z값을 확률로 변환.
- ex. 도미, 빙어, 청돔, 농어, 송어 중 하나로 분류 (class 5개)
- ex. 붓꽃의 종류 분류 (class 3개: Setosa-0, Versicolor-1, Virginica-2)

소프트맥스 함수

$$\text{softmax}(z_j) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \text{ for j} = 1,\ldots,\text{K}$$

Softmax Function

: 여러 개의 선형 방정식의 출력값(z)을 0과 1 사이로 압축하고, **전체 합이 1이 되게** 만드는 함수이다.

z_j: softmax 함수의 입력 요소. 로지스틱 회귀분석의 경우, 선형 방정식의 출력값으로 $z \in (-\infty, \infty)$.

K: 클래스 수

e^z: 표준 지수 함수에 z값을 입력하여 $(-\infty, \infty)$를 $[0, \infty)$ 범위의 값으로 바꿔준다.

$\sum_{k=1}^{K} e^{z^k}$: 정규화 항으로, 함수의 모든 출력값의 합 1로 만들어주고, 각 출력값을 [0, 1] 범위의 값으로 변환한다.

소프트맥스 함수

$$\text{softmax}(z_j) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \text{ for j} = 1,\ldots,\text{K}$$

- **다중 분류**에서 주로 사용한다.
- '**정규화된 지수함수**'라고도 한다.

(ex) k=3일 때,

$$softmax(z^1) = \frac{e^{z^1}}{e^{z^1} + e^{z^2} + e^{z^3}}$$

$$softmax(z^2) = \frac{e^{z^2}}{e^{z^1} + e^{z^2} + e^{z^3}}$$

$$softmax(z^3) = \frac{e^{z^3}}{e^{z^1} + e^{z^2} + e^{z^3}}$$

소프트맥스 함수

이진 분류 (클래스 2종) - 시그모이드

$$sigmoid(z_j) = \frac{1}{1 + \exp(-z_j)}$$

- 하나의 선형 방정식의 출력값(z)을 0~1 사이로 압축한다.
- 주로 두 개의 클래스 중 **타깃값 하나**에 시그모이드 함수를 사용한다.

다중 분류 (클래스 2종 이상) - 소프트맥스

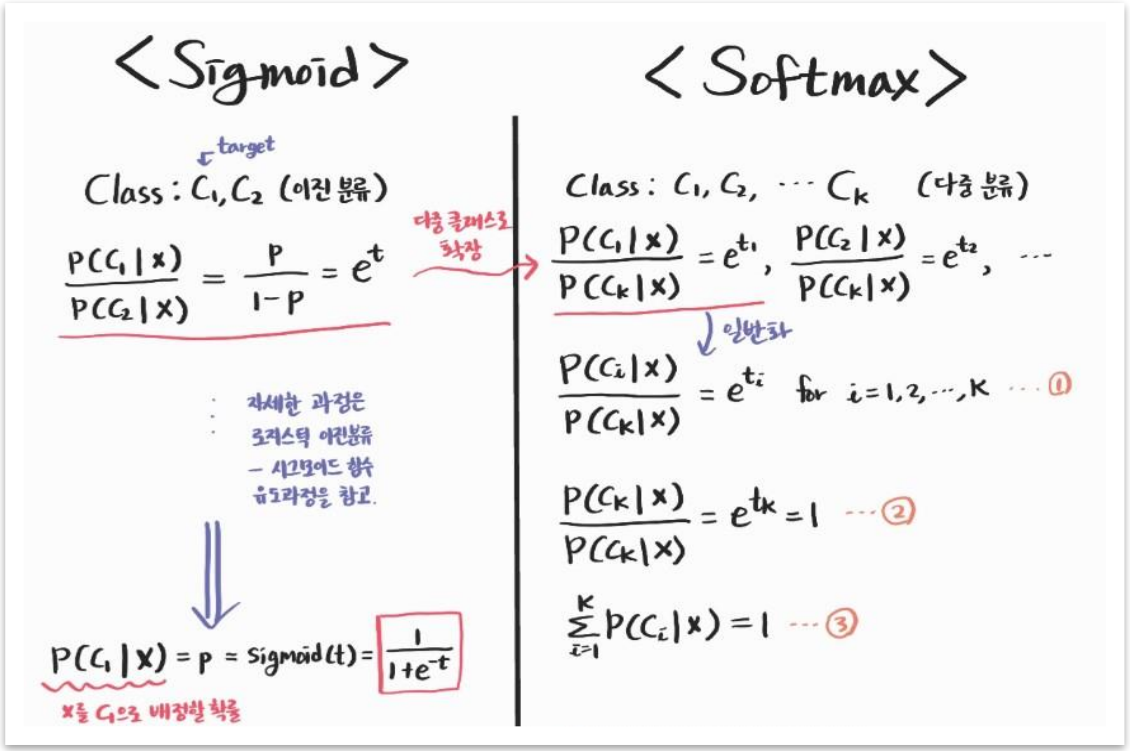
$$\text{softmax}(z_j) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \text{ for j} = 1,\ldots,\text{K}$$

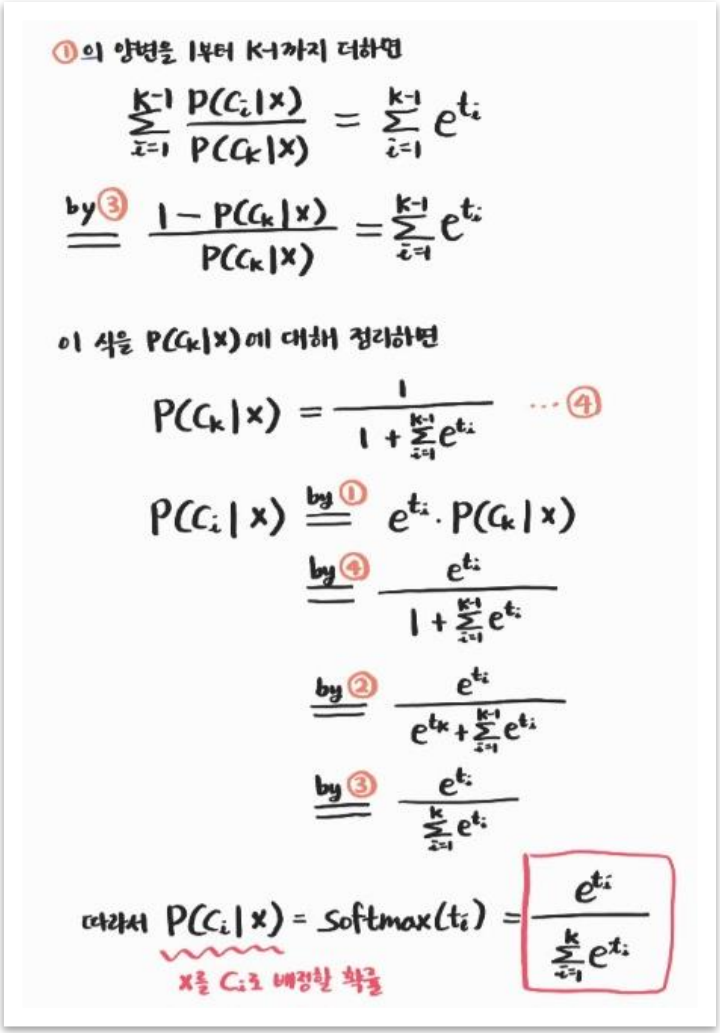
- 여러 개의 선형 방정식의 출력값(z_j)을 0~1 사이로 압축하고, 전체 합이 1이 되도록 만든다.
- **클래스 개수 만큼** 소프트맥스 함수를 사용한다.

소프트맥스 함수

유도 과정

- 오즈비로부터 시그모이드 함수를 유도한 방법과 동일하게, Class를 K개로 확장하여 소프트맥스 함수를 유도할 수 있음.





소프트맥스 함수

$$\text{Sigmoid}$$
$$S(x) = S(3) = \frac{1}{1+e^{-3}} = \frac{1}{1+e^{-3}} = 0.953$$

$$\text{Softmax}$$
$$\sigma(\vec{z})_1 = \sigma(3) = \frac{e^3}{e^3+e^0} = 0.953$$
$$\sigma(\vec{z})_2 = \sigma(0) = \frac{e^0}{e^3+e^0} = 0.0474$$

- z값을 하나만 알아도 되는 시그모이드 함수와 달리 **소프트맥스 함수를 사용하기 위해서는 z값을 모두 알아야 하기 때문에**, 이진 분류에서는 시그모이드를 더 자주 사용한다.

소프트맥스 함수

이진 분류 (ex. 생선 2종류)

z값: -6.03

Sigmoid: 0.002

생선을 Bream, Smelt 중 하나로 분류하는 경우
(sigmoid 함수 1개 필요)

타깃 Smelt에 대한 z값: -6.03
Sigmoid 함수값: 0.002
따라서 Smelt일 확률 0.002, Bream일 확률 0.998
→ Bream으로 분류

다중 분류 (ex. 생선 7종류)

	Bream	Parkki	Perch	Pike	Roach	Smelt	Whitefish
z값:	[-6.5,	1.03,	5.16,	-2.73,	3.34,	0.33,	-0.63]

Sigmoid: [0.002, 0.737, 0.994, 0.061, 0.966, 0.582, 0.348]
Softmax: [0.000, 0.014, 0.841, 0.000, 0.136, 0.007, 0.003]

생선을 7개의 클래스 중 하나로 분류하는 경우
(softmax 함수 7개 필요)

각 타깃에 대한 Sigmoid 함수값은 전체 합이 1이 되지 않음.
전체 합이 1이 되도록 정규화하는 Softmax 함수를 이용한다.
Perch일 확률 0.841 → Perch로 분류

규제

L2 규제

- 로지스틱 회귀분석은 기본적으로 **L2 규제**(ex. Ridge)를 사용한다.
- Ridge 회귀 - 매개변수 α (α가 커질수록 규제가 커짐)
- 로지스틱 회귀 - **매개변수 C** (C가 커질수록 규제가 작아짐)

규제

손코딩

```
lr = LogisticRegression(C=20, max_iter=1000)
lr.fit(train_scaled, train_target)
print(lr.score(train_scaled, train_target))
print(lr.score(test_scaled, test_target))
```

```
0.9327731092436975
0.925
```

C: L2 규제에서의 매개변수

- C의 기본값은 1이다.
- 코드에서는 C를 20으로 늘려 규제를 완화했다.

max_iter: 모델 훈련의 반복 횟수

- max_iter의 기본값은 100이다.
- 반복 횟수가 부족하면 경고가 뜰 수 있다. 코드에서는 max_iter를 1000으로 늘렸다.

Iris 데이터를 이용한 실습

로지스틱 회귀분석 실습

- Scikit-learn의 iris dataset을 이용한다.
- 붓꽃의 꽃받침과 꽃잎 각각의 길이와 폭을 이용하여 붓꽃을 세 종류로 분류한다.

```
#Iris 데이터셋 불러오기
from sklearn.datasets import load_iris
iris = load_iris()
X = pd.DataFrame(iris.data, columns=iris.feature_names)
y = pd.DataFrame(iris.target, columns=['class'])
```

Iris 데이터를 이용한 실습

X

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

150 rows × 4 columns

y

	class
0	0
1	0
2	0
3	0
4	0
...	...
145	2
146	2
147	2
148	2
149	2

150 rows × 1 columns

4개의 변수 (X)

- Sepal Length(cm) - 꽃받침 길이
- Sepal Width(cm) - 꽃받침 폭
- Petal Length(cm) - 꽃잎 길이
- Petal Width(cm) - 꽃잎 폭

클래스 (y)

- 붓꽃의 종류

(setosa=0, versicolor=1, virginica=2)

Iris 데이터를 이용한 실습

```
#훈련 셋과 테스트 셋으로 나누기
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=16)


#로지스틱 회귀분석 수행 (이진 분류, 다중 분류 상관 없이 모두 LogisticRegression으로 계산)
from sklearn.linear_model import LogisticRegressionCV
LR=make_pipeline(StandardScaler(),LogisticRegressionCV(max_iter=1000))
LR.fit(X_train,y_train)


#훈련 셋의 스코어와 테스트 셋의 스코어 출력
train_score = LR.score(X_train, y_train)
test_score = LR.score(X_test, y_test)
print(train_score)                                    0.9821428571428571
print(test_score)                                     0.8947368421052632
```

Iris 데이터를 이용한 실습

```
#X_train의 맨 위 5개 데이터의 클래스 예측 결과
LR.predict(X_train[:5])
```

```
array([2, 0, 2, 2, 0])
```

```
#실제 클래스
y_train[:5]
```

	Class
107	2
35	0
112	2
131	2
49	0

```
#맨 위 5개 데이터가 각 클래스에 해당할 확률
LR.predict_proba(X_train[:5])
```

```
array([[9.16846742e-07, 3.46108844e-02, 9.65388199e-01],
       [9.85107156e-01, 1.48928404e-02, 3.85594858e-09],
       [4.71265346e-06, 2.22460404e-02, 9.77749247e-01],
       [2.08794387e-06, 1.26070342e-02, 9.87390878e-01],
       [9.87024708e-01, 1.29752876e-02, 4.24029563e-09]])
```

Iris 데이터를 이용한 실습

```
#회귀 모델이 학습한 방정식 출력하기
coef = LR.named_steps['logisticregressioncv'].coef_
interc = LR.named_steps['logisticregressioncv'].intercept_

for i in range(len(coef)):
  print(f'z{i} = {interc[i]:.3} + {coef[i][0]:.3}*(sepal length) + {coef[i][1]:.3}*(
sepal width) + {coef[i][2]:.3}*(petal length) + {coef[i][3]:.3}*(petal width)')
```

z0 = -0.394 + -1.29*(sepal length) + 1.23*(sepal width) + -2.38*(petal length) + -2.27*(petal width)

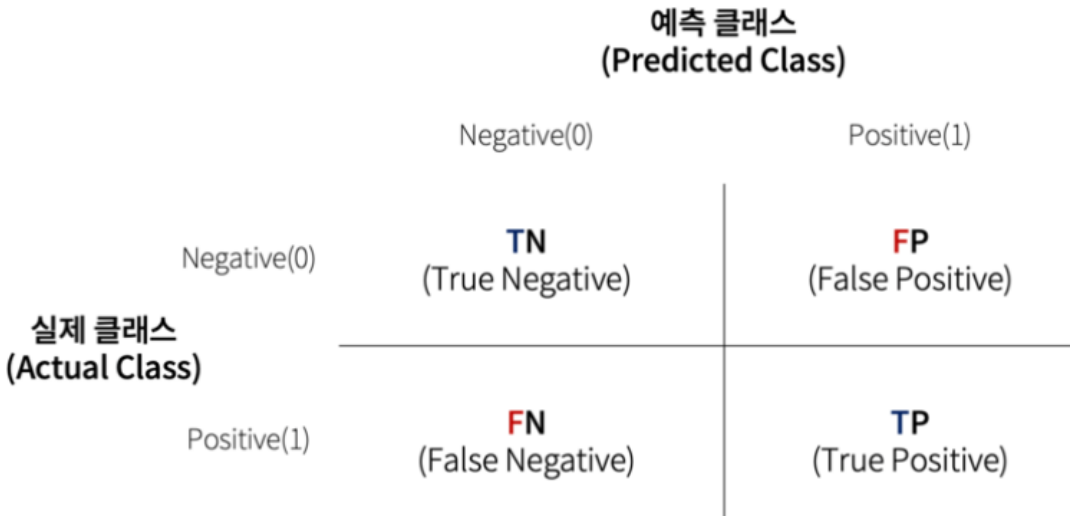
z1 = 2.48 + 0.502*(sepal length) + -0.478*(sepal width) + -0.13*(petal length) + -1.24*(petal width)

z2 = -2.09 + 0.783*(sepal length) + -0.753*(sepal width) + 2.51*(petal length) + 3.51*(petal width)

분류 모델의 평가 지표

오 차 행 렬 을 중 심 으 로

오차 행렬 (Confusion Matrix)



오차행렬

- 클래스가 1(positive, 양성)과 0(negative, 음성)뿐인 이진 분류의 결과표.
- 예측 클래스 = 실제 클래스이면 True, 다르면 False.

True Positive = 진양성

: 양성(1)으로 맞게 예측했고, 실제로도 양성(1).

True Negative = 진음성

: 음성(0)으로 맞게 예측했고, 실제로도 음성(0).

False Positive = 위양성

: 양성(1)으로 잘못 예측했고, 실제로는 음성(0).

False Negative = 위음성

: 음성(0)으로 잘못 예측했고, 실제로는 양성(1).

분류 모델의 평가 지표

오차 행렬 (Confusion Matrix)

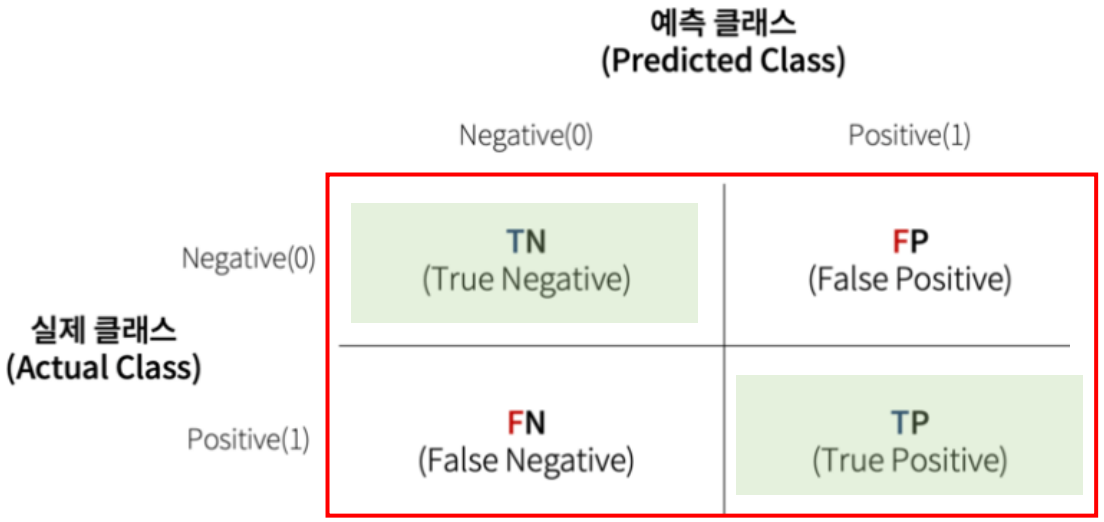
	예측 클래스 0	예측 클래스 1	예측 클래스 2
정답 클래스 0	TP	FP/FN	FP/FN
정답 클래스 1	FP/FN	TP	FP/FN
정답 클래스 2	FP/FN	FP/FN	TP

(참고) 클래스가 3개일 때 오차행렬

어떤 클래스를 target으로 설정하는지에 따라 Positive와 Negative가 달라질 수 있다.

분류 모델의 평가 지표

정확도 (Accuracy)

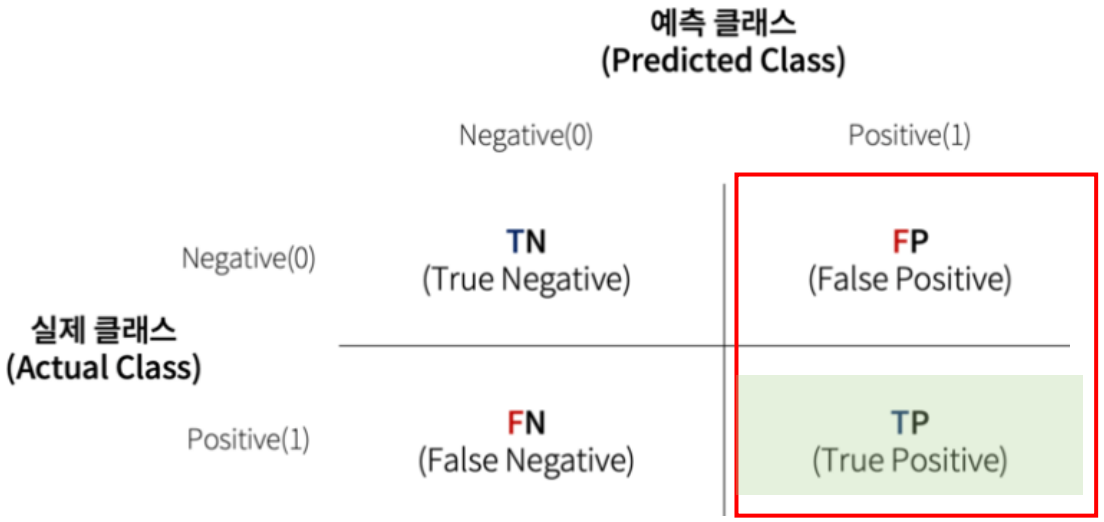


$$accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

- 전체 샘플 중 맞게 예측한 샘플의 비율.
- 높을수록 좋은 모형이다.
- 클래스 간의 샘플 수가 균일하지 못할 경우, 정확도(accuracy)로 모델의 성능을 판단하기 어렵다.

분류 모델의 평가 지표

정밀도 (Precision)

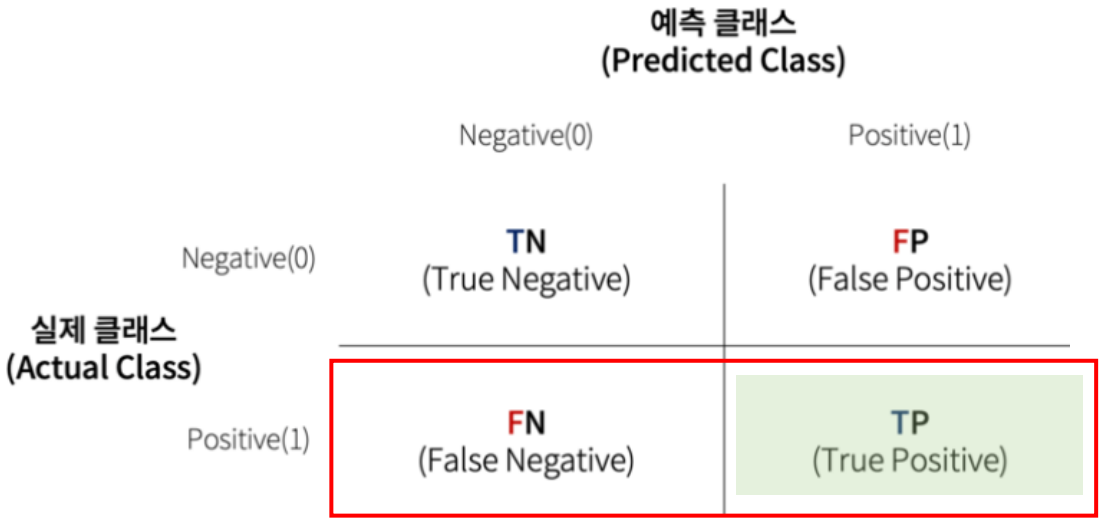


$$precision = \frac{TP}{TP+FP}$$

- 양성 클래스로 판단한 샘플 중 실제로 양성 클래스인 샘플의 비율.
- 높을수록 좋은 모형이다.
- ex. 암 환자로 판단한 사람 중 실제로 암 환자인 사람의 비율.

분류 모델의 평가 지표

재현율 (Recall)

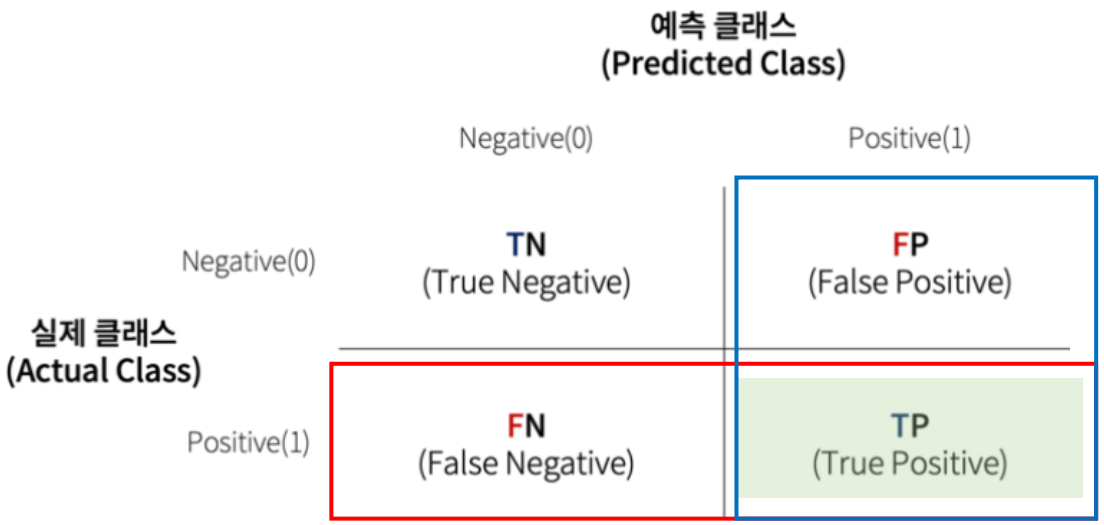


$$recall = \frac{TP}{TP+FN}$$

- 실제 양성 클래스인 샘플 중 양성 클래스로 판단한 샘플의 비율.
- 높을수록 좋은 모형이다.
- **TPR(True Positive Rate)**라고도 한다.
- ex. 실제로 암 환자인 사람 중 암 환자로 맞게 판단한 사람의 비율.

분류 모델의 평가 지표

정밀도 vs 재현율



$$precision = \frac{TP}{TP+FP} \quad , \quad recall = \frac{TP}{TP+FN}$$

- 정밀도는 FP를 줄이는데 초점을 맞추고, 재현율은 FN을 줄이는데 초점을 맞춘다.
- 스팸 메일 판별 시, 실제로 일반 메일(0)인데 스팸 메일(1)로 판단하면 치명적이다. 이럴 경우 **FP에 초점을 맞추기 위해 precision을** 평가지표로 사용한다.
- 암 환자 판별 시, 실제로 암 환자(1)인데 암 환자가 아니라고(0) 판단하면 치명적이다. 이럴 경우 **FN에 초점을 맞추기 위해 recall을** 평가 지표로 사용한다.

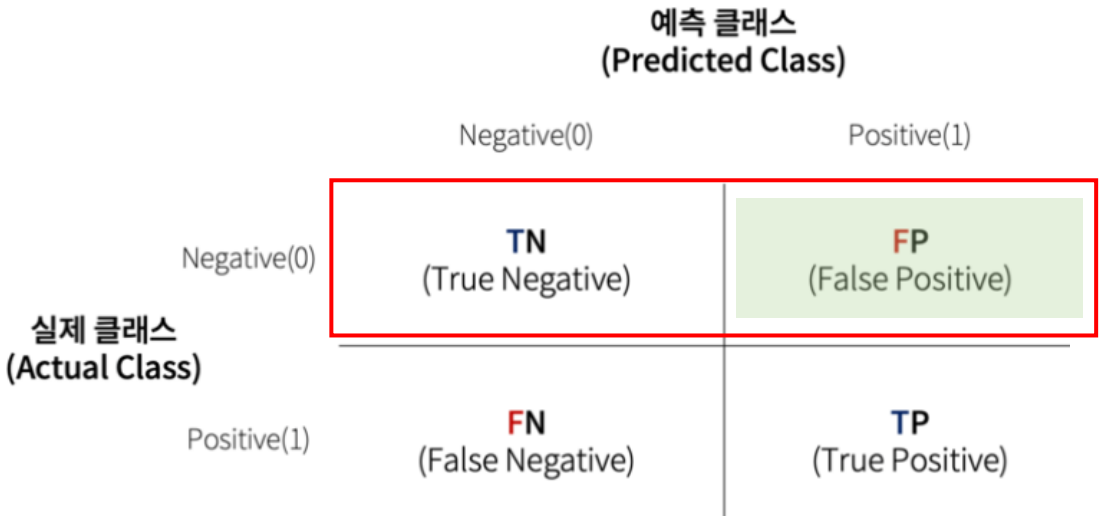
F1 score

$$F1\ Score = 2 \times \frac{precision\ \times\ recall}{precision\ +\ recall}$$

- **정밀도(precision)과 재현율(recall)의 조화 평균**
- 높을수록 좋은 모델이다.

분류 모델의 평가 지표

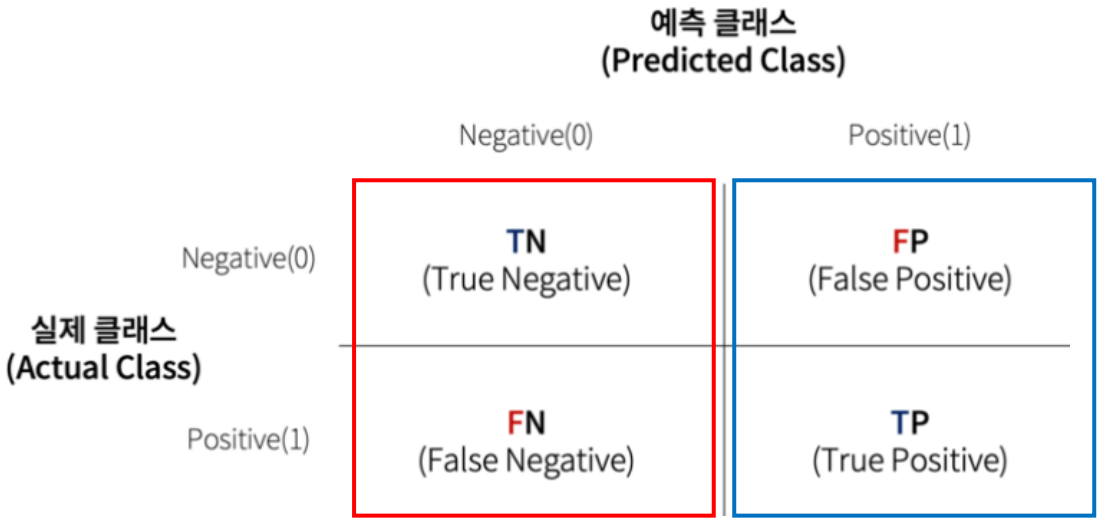
위양성률 (Fall-out)



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

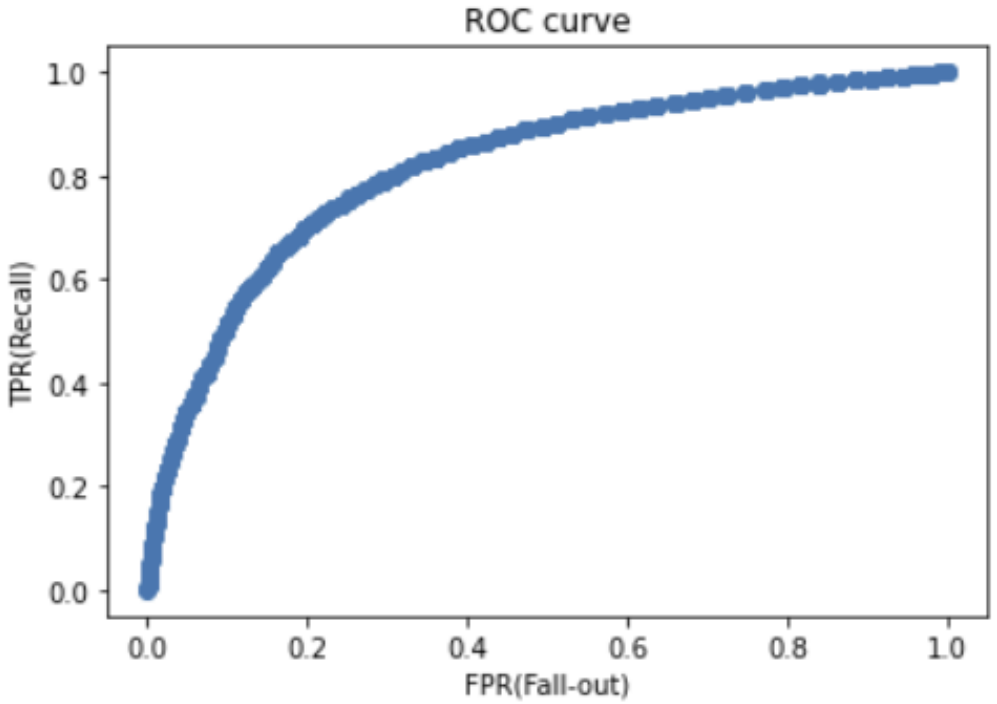
- 실제 양성 클래스에 속하지 않는 샘플 중 양성 클래스로 판단한 샘플의 비율.
- 낮을수록 좋은 모형이다.
- **FPR(False Positive Rate)**라고도 한다.
- ex. 실제로 암 환자가 아닌 사람 중 암 환자로 잘못 판단한 사람의 비율.

판별 기준 변화에 따른 평가지표의 변화



- **판별 기준을 1에 가깝게 할 경우**, 양성으로 예측되는 샘플이 늘어난다. 파란색으로 표현된 TP와 FP는 증가하고, 빨간색으로 표현된 TN과 FN은 감소한다.
- 반대로 **판별 기준을 0에 가깝게 할 경우**, 음성으로 예측되는 샘플이 늘어난다. TP와 FP는 감소하고, TN과 FN이 증가한다.
- 판별 기준을 변화시킬 때, 정확도와 정밀도는 상황에 따라 높아질 수도 낮아질 수도 있다.
- 판별 기준을 1에 가까워지도록 변화시킬 때, **재현율과 위양성률이 모두 증가**한다.

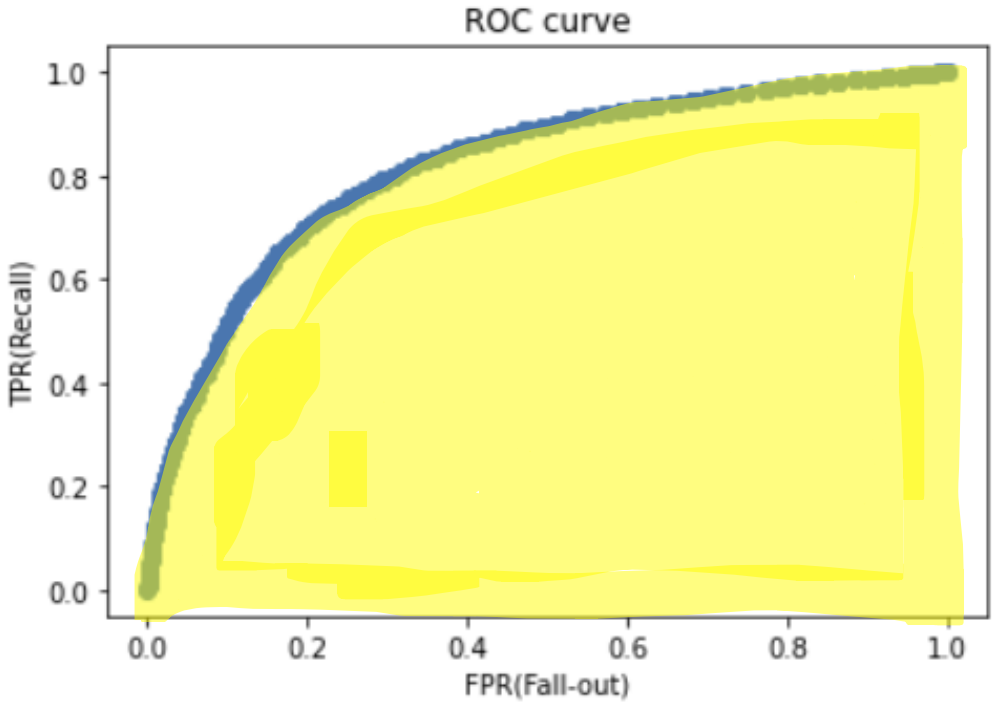
ROC curve와 AUC



ROC 곡선

- 판별 기준의 변화에 따른 **재현율(TPR)과 위양성률(FPR)의 변화**를 나타내는 그래프이다.
- TPR과 FPR은 일반적으로 양의 상관관계이다. 재현율을 높이기 위해 양성 클래스 판단 기준을 낮추면, FP가 늘어나 위양성률이 높아진다.
- **왼쪽 상단**에 가까운 점일수록 TPR이 커지고 FRP이 작아지므로 최적의 값이다.

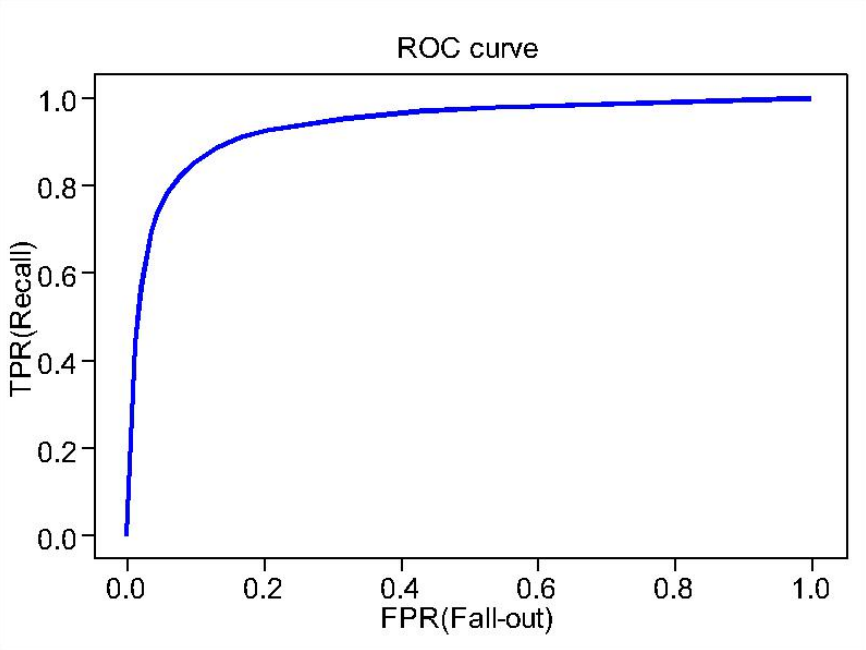
ROC curve와 AUC



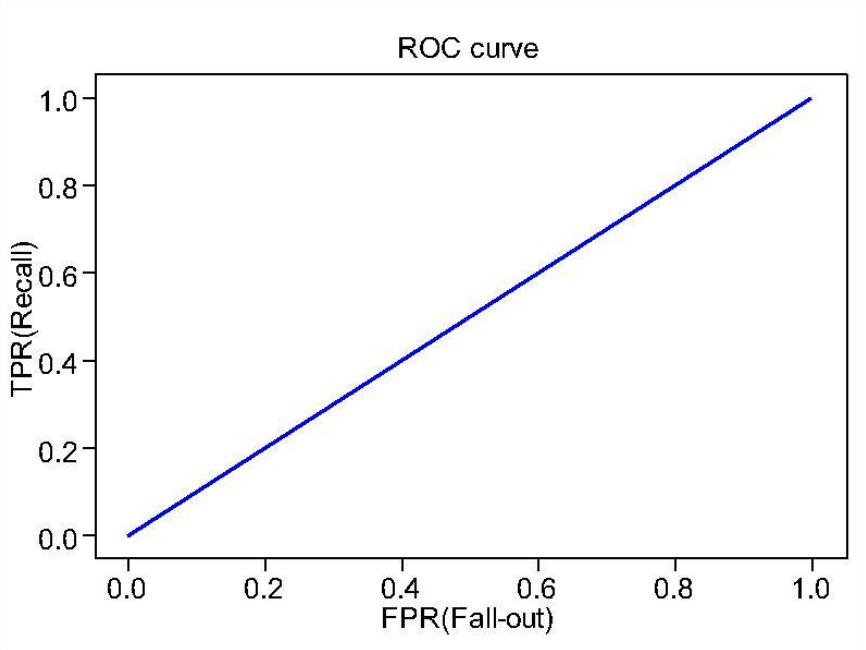
AUC (Area Under the Curve) 점수

- **ROC 곡선의 면적**이다.
- $0.5 \leq AUC \leq 1$
- ROC curve가 좌상단에 가까우면 AUC가 1에 가까워지고, 좋은 모델이다.
- ROC curve가 y=x에 가까우면 AUC가 0.5에 가까워지고, 좋지 않은 모델이다.

ROC curve와 AUC



AUC가 높은 경우 (about 1)



AUC가 낮은 경우 (about 0.5)