CHAPTER 08

# Logistic Regression Basics

01 로지스틱 회귀란?

02 분류 문제의 성능지표

03 로지스틱 회귀 구현하기

# 01
# 로지스틱 회귀란?

# 01 로지스틱 회귀란?

- 분류 문제 : 몇 가지 이산적 값 중 하나를 선택하는 모델. '분류 모델'이라고 부름

표 9-1 GRE와 GPA 정보를 활용하여 합격 여부를 나타내는 데이터

| Number | Admit | GRE | GPA | Number | Admit | GRE | GPA |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 380 | 3.61 | 16 | 0 | 660 | 3.34 |
| 2 | 1 | 660 | 3.67 | 17 | 1 | 740 | 4.00 |
| 3 | 1 | 800 | 4.00 | 18 | 0 | 560 | 3.19 |
| 4 | 1 | 640 | 3.19 | 19 | 0 | 380 | 2.94 |
| 5 | 0 | 520 | 2.93 | 20 | 0 | 400 | 3.65 |
| 6 | 1 | 760 | 3.00 | 21 | 0 | 600 | 2.82 |
| 7 | 1 | 560 | 2.98 | 22 | 1 | 620 | 3.18 |
| 8 | 0 | 400 | 3.08 | 23 | 0 | 560 | 3.32 |
| 9 | 1 | 540 | 3.39 | 24 | 0 | 640 | 3.67 |
| 10 | 0 | 700 | 3.92 | 25 | 1 | 680 | 3.85 |
| 11 | 0 | 800 | 4.00 | 26 | 0 | 580 | 4.00 |
| 12 | 0 | 440 | 3.22 | 27 | 0 | 600 | 3.59 |
| 13 | 1 | 760 | 4.00 | 28 | 0 | 740 | 3.62 |
| 14 | 0 | 700 | 3.08 | 29 | 0 | 620 | 3.30 |
| 15 | 1 | 700 | 4.00 | 30 | 0 | 580 | 3.69 |

# 01 로지스틱 회귀란?

- [표 9-1]은 GRE와 GPA 데이터를 통해 대학의 합격 여부(Admit 열)를 나타냄
- GRE와 GPA 정보를 산점도로 표현
    - 합격자는 파란색, 불합격자는 빨간색으로 나타냄

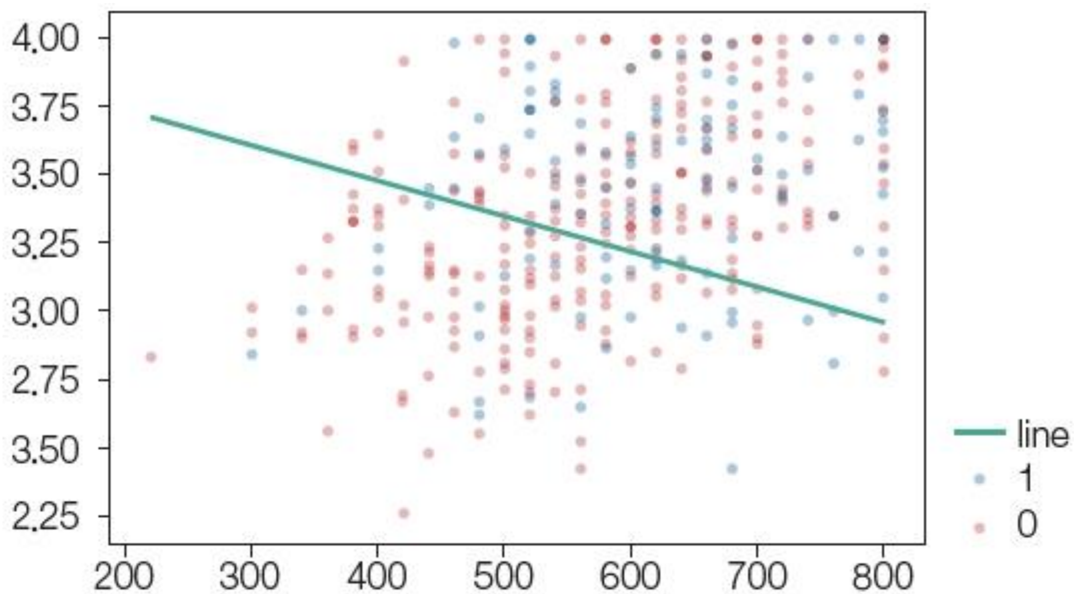



그림 9-1 산점도 위에 선형회귀 모델 표현

# 01 로지스틱 회귀란?

- 초록색 선을 추가해 선 상단은 합격, 선 하단은 불합격
  - 아래 수식으로 기존 선형회귀 모델을 적용

$$f(x) = 4 - 0.0013 \times GRE - GPA$$

$$Admit = \begin{cases} 1 & \text{if } f(x) \geq 0 \\ 0 & otherwise \end{cases}$$

  - 문제점:

    ❶ $f(x)$의 값이 1 이상이나 0 이하로 나올 수 있음

    ❷ 각 피쳐들이 Y에 영향을 주는 것을 해석하는 문제

    ❸ 사건의 발생 여부는 이산적인데 실제 $f(x)$수식은 연속적

# 01 로지스틱 회귀란?

## 1. 로지스틱 회귀의 개념

- 이진 분류(binary classification) 문제를 확률로 표현
- 어떤 사건이 일어날 확률을 P(X)로 나타내고 일어나지 않을 확률을 1 - P(x)로 나타냄 ($0 \le P(X) \le 1$)
- 오즈비(odds ratio) : 어떤 사건이 일어날 확률과 일어나지 않을 확률의 비율

$$\frac{P(X)}{1-P(X)}$$

# 01 로지스틱 회귀란?

- 확률이 올라갈수록 오즈비도 급속히 상승

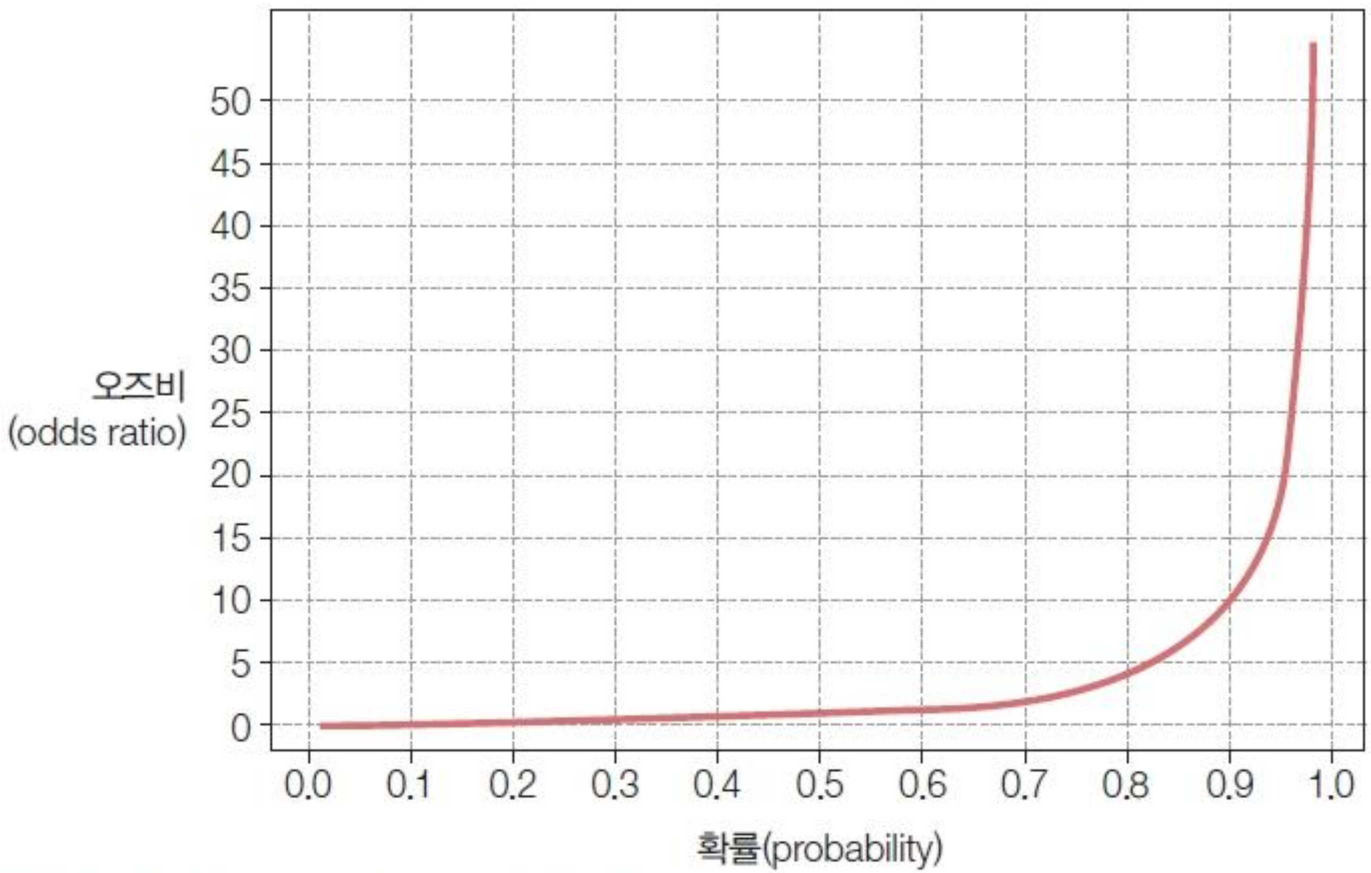



그림 9-2 확률이 올라가면서 오즈비가 상승하는 그래프

# 01 로지스틱 회귀란?

- 로짓(logit) 함수 : 오즈비에 상용로그를 붙인 수식

$$
\begin{aligned}
logit(p(y=1|x)) &= \log_e\left(\frac{p}{1-p}\right) \\
&= \log_e(p) - \log_e(1-p) \\
&= -\log_e\left(\frac{1}{p}-1\right)
\end{aligned}
$$

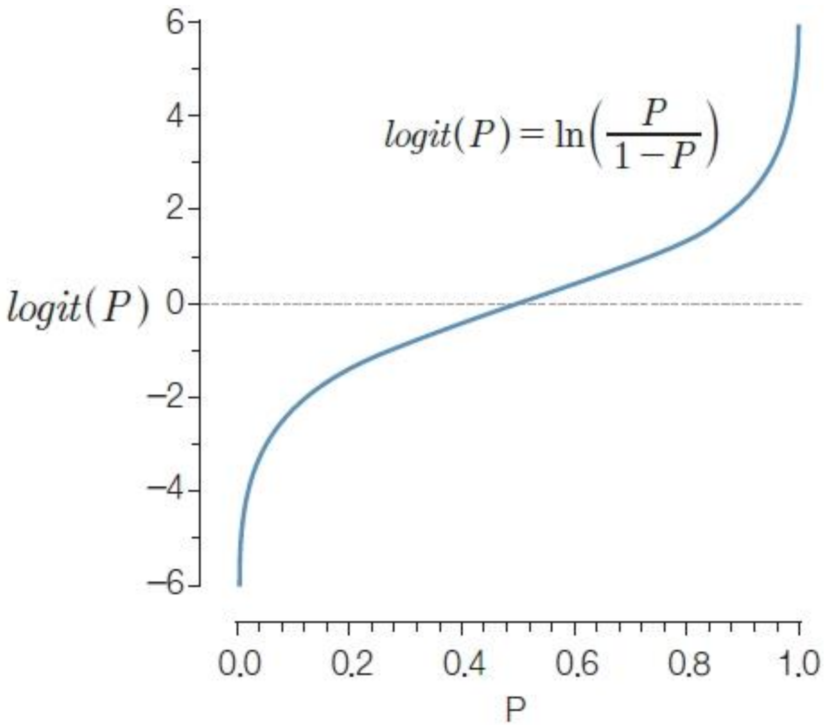



그림 9-3 로짓 함수 그래프

# 01 로지스틱 회귀란?

- X 값으로 확률을 넣으면 logit(P) 꼴로 나타남
- 확률을 구하려면 기존 함수의 역함수를 취하여 연산

$$f(z) = y = -\log_e\left(\frac{1}{z} - 1\right)$$

$$z = -\log_e\left(\frac{1}{y} - 1\right)$$

$$e^{-z} = \frac{1 - y}{y}$$

$$y \times e^{-z} + y = 1$$

$$y(e^{-z} + 1) = 1$$

$$y = \frac{1}{1 + e^{-z}}$$

$$f(X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}}$$

# 01 로지스틱 회귀란?

- 로지스틱 함수(logistic function) : 로짓 함수의 역함수
  - 그래프가 s자 커브 형태인 시그모이드 함수(sigmoid function)

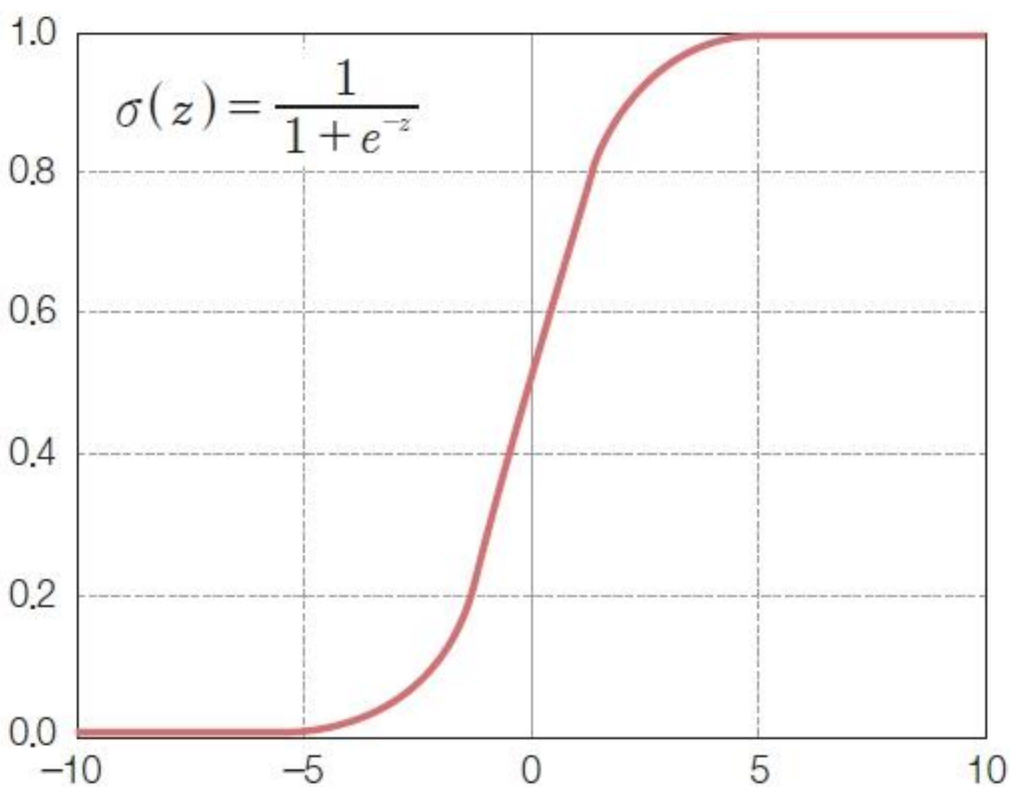



그림 9-4 시그모이드 함수

# 01 로지스틱 회귀란?

- 로지스틱 회귀(Logistic Regression) : 종속변수가 이분형일 때 수행할 수 있는, 예측 분석을 위한 회귀분석 기법
- 시그모이드 함수 수식
  - y 값을 확률 p로 표현
  - z 값은  선형회귀와 같이 가중치와 피쳐의 선형 결합(linear combination)으로 표현 가능

$$p = \sigma(z) = \frac{1}{1+e^{-z}}, \quad \frac{p}{1-p} = \frac{\frac{1}{1+e^{-z}}}{\frac{e^{-z}}{1+e^{-z}}} = \frac{1}{e^{-z}} = e^{z}$$

$$\log_e \frac{p}{1-p} = z$$

$$\log_e \frac{p}{1-p} = z = w_0 x_0 + w_1 x_1 + \cdots + w_n x_n$$

## 2. 로지스틱 회귀의 기본 함수

### 2.1 가설함수

- 가설함수(hypothesis function)

$$h_\theta(x) = g(z) = \frac{1}{1+e^{-z}}$$

  - z는 가중치 값과 피쳐 값의 선형 결합
  - 가중치 값을 찾는 학습을 위해 경사하강법 알고리즘 사용

$$z = w_0x_0 + w_1x_1 + \cdots + w_nx_n = \theta^T X$$

# 01 로지스틱 회귀란?

## 2.2 비용함수

- 먼저 비용함수를 정의하고 예측값과 실제값 간의 차이를 최소화하는 방향으로 학습
- 실제값이 1일 때와 실제값이 0일 때 각각 다르게 비용함수를 정의

$$Cost(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

# 01 로지스틱 회귀란?

- (a)는 y = 1일 때, (b)는 y = 0일 때 비용함수 그래프
  (0 ≤ h ≤ 1)

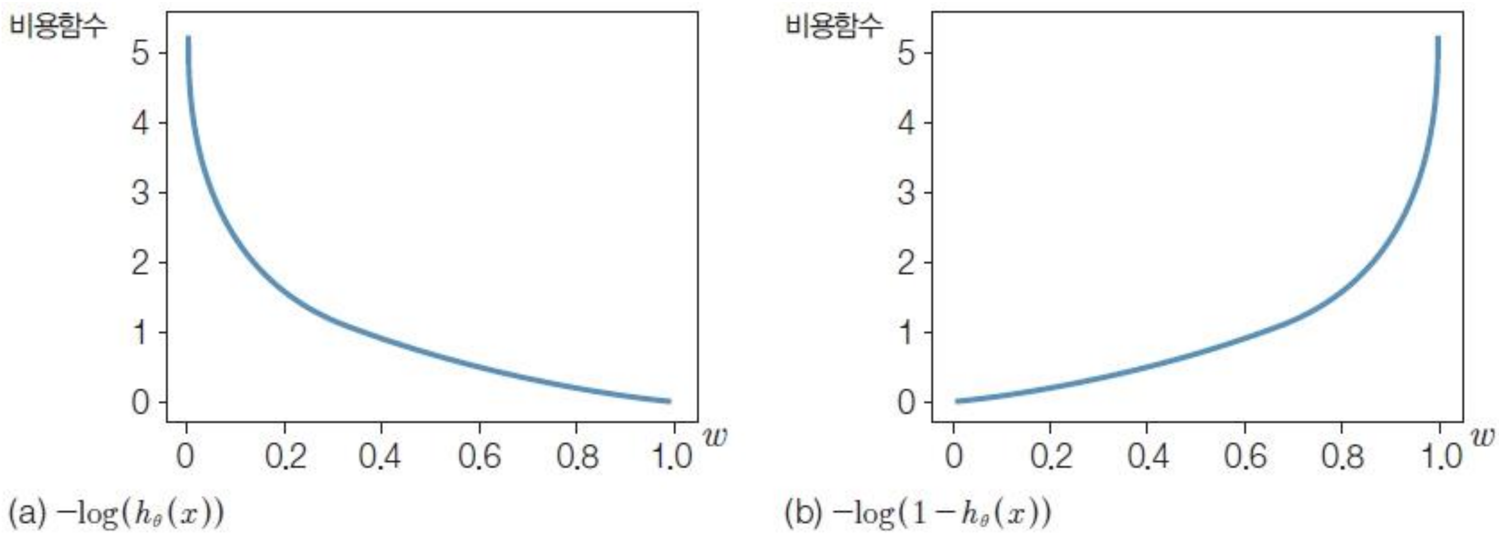



그림 9-5 비용함수 그래프

- (a)에서 h 값이 1에 가까워질수록 비용함수가 0에 가까워짐
  (b)에서 h 값이 0에 가까워질수록 비용함수가 0에 가까워짐

# 01 로지스틱 회귀란?

- 두 경우의 비용함수를 하나로 통합

$$
\begin{aligned}
J(\theta) &= \frac{1}{m}\sum_{i=1}^{m} Cost(h_\theta(x^{(i)}), y^{(i)}) \\
&= -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log h_\theta(x^{(i)}) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right]
\end{aligned}
$$

# 02
# 분류 문제의 성능지표

# 02 분류 문제의 성능지표

## 1. 분류 문제에 있어서 성능지표의 필요성

- 모델을 평가하는 성능지표들
  - 회귀(regression) : MAE, MSE, RMSE, SSE
  - 분류(classification) : 정확도, 정밀도, 민감도, F1 스코어, ROC 커브, 리프트 차트
  - 클러스터링(clustering) : DBI, 엘보우 메서드, 실루엣계수

# 02 분류 문제의 성능지표

**[하나 더 알기] 머신러닝에서 다양한 추가 성능지표를 사용하는 예**

앞에서 소개한 기초 성능지표 외에도 실제 머신러닝을 적용한 시스템에서 사용가능한 다양한 추가 성능지표들이 있다. 이때 다음과 같은 여러 가지 상황을 고려하여 모델의 성능지표를 선택해야 한다.

- 모델이 다른 모델보다 경제적으로 나은가?
- 모델이 사용하는 데이터가 많은가? 또는 적은가?
- 모델이 용량이 작은 컴퓨터에서도 작동하는가?
- 모델의 손해가 얼마나 나는가

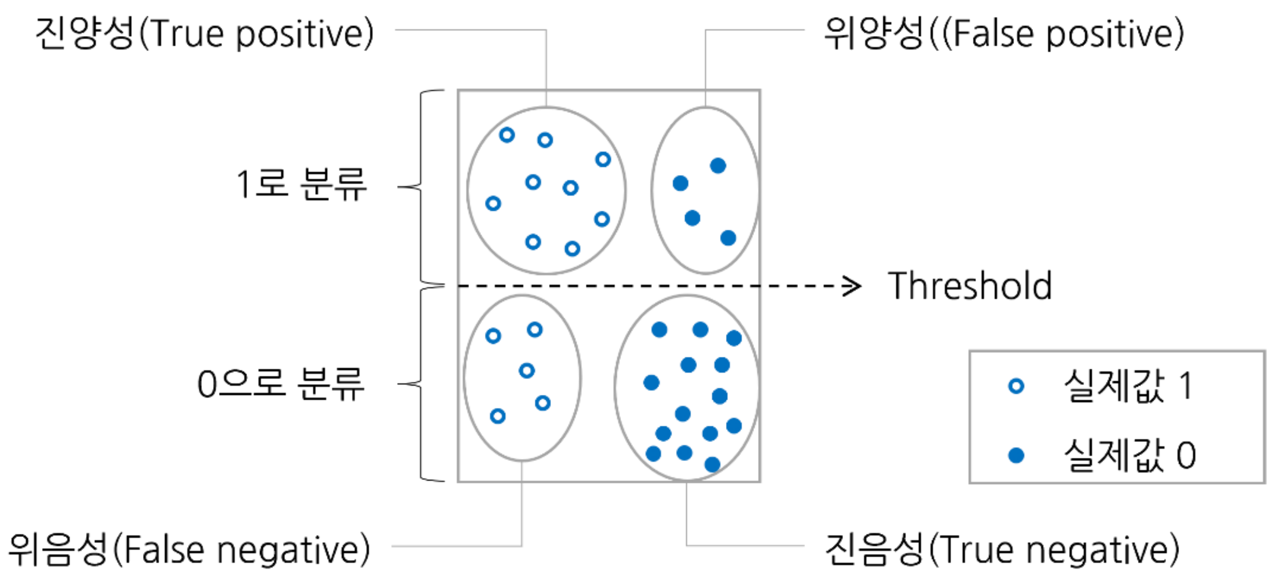
## 2. 혼동행렬

- 혼동행렬(confusion matrix): 예측값이 실제값 대비 얼마나 잘 맞는지 2×2 행렬로 표현
  - 일반적으로 질문의 '예'에 해당하는 값은 True 또는 1, '아니오'에 해당하는 값은 False 또는 0

| | | 예측값(prediction) | |
|---|---|---|---|
| | | 1 | 0 |
| 실제값 (actual class) | 1 | True Positive | False Negative |
| | 0 | False Positive | True Negative |

그림 9-6 혼동행렬

# 02 분류 문제의 성능지표




- **진양성(TP, True positive):** 모델은 1(Positive)로 예측하고, 실젯값도 동일(True)
- **위양성(FP, False positive):** 모델은 1(Positive)로 예측하고, 실젯값은 상이(False)
- **진음성(TN, True negative):** 모델은 0(Negative)로 예측하고, 실젯값도 동일(True)
- **위음성(FN, False negative):** 모델은 0(Negative)로 예측하고, 실젯값은 상이(false)

# 02 분류 문제의 성능지표

- 널리 쓰이는 성능 측정 기준
  - 정확률$_{\text{accuracy}}$
    - 부류가 불균형일 때 성능을 제대로 반영하지 못함

$$정확률 = \frac{맞힌\ 샘플\ 수}{전체\ 샘플\ 수} = \frac{대각선\ 샘플\ 수}{전체\ 샘플\ 수}$$

  - 특이도$_{\text{specificity}}$와 민감도$_{\text{sensitivity}}$ (의료에서 주로 사용)

$$특이도 = \frac{TN}{TN+FP},\quad 민감도 = \frac{TP}{TP+FN}$$

  - 정밀도$_{\text{precision}}$와 재현률$_{\text{recall}}$ (정보검색에서 주로 사용)

$$정밀도 = \frac{TP}{TP+FP},\quad 재현율 = \frac{TP}{TP+FN}$$

| | | 그라운드 트루스 | |
|---|---|---|---|
| | | 긍정 | 부정 |
| 예측값 | 긍정 | TP | FP |
| | 부정 | FN | TN |

# 02 분류 문제의 성능지표

- 사이킷런으로 혼동행렬표 나타내기

| | |
|---|---|
| In [1]: | `from sklearn.metrics import confusion_matrix`<br>`y_true = [1, 0, 1, 1, 0, 1]`<br>`y_pred = [0, 0, 1, 1, 0, 1]`<br>`confusion_matrix(y_true, y_pred)` |
| Out [1]: | `array([[2, 0],`<br>`       [1, 3]], dtype=int64)` |
| In [2]: | `tn, fp, fn, tp = confusion_matrix(y_true,`<br>`y_pred).ravel()`<br>`(tn, fp, fn, tp)` |
| Out [2]: | `(2, 0, 1, 3)` |

# 02 분류 문제의 성능지표

## 3. 혼동행렬표를 사용한 지표

### 3.1 정확도

- 정확도(accuracy) : 전체 데이터 개수 대비 정답을 맞춘 데이터의 개수

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

| | | 예측값(prediction) | |
|---|---|---|---|
| | | 1 | 0 |
| 실제값 (actual class) | 1 | True Positive | False Negative |
| | 0 | False Positive | True Negative |

그림 9-7 정확도 측정

# 02 분류 문제의 성능지표

- 사이킷런으로 정확도 구하기

| | |
|---|---|
| In [3]: | import numpy as np<br>from sklearn.metrics import accuracy_score<br>y_pred = np.array([0, 1, 1, 0])<br>y_true = np.array([0, 1, 0, 0])<br>sum(y_true == y_pred) / len(y_true) |
| Out [3]: | 0.75 |
| In [4]: | accuracy_score(y_true, y_pred) |
| Out [4]: | 0.75 |

[TIP] 사실상 정확도와 동일하나 오답을 맞춘 개수를 구하는 오차율(error rate)이라는 지표도 존재한다. 오차율 지표는 'ERR=1-ACC'로 '1-정확도'로 이해할 수 있다.

# 02 분류 문제의 성능지표

## 3.2 정밀도, 민감도, F1 스코어

- 정밀도와 민감도는 불균일한 데이터셋을 다룰 때 유용
  - 데이터에서 1과 0의 비율이 7:3 또는 3:7 이상 차이나는 상태
- 정밀도(precision) : 모델이 1이라고 예측했을 때 얼마나 잘 맞을지에 대한 비율

$$PRECISION(PPV) = \frac{TP}{TP + FP}$$

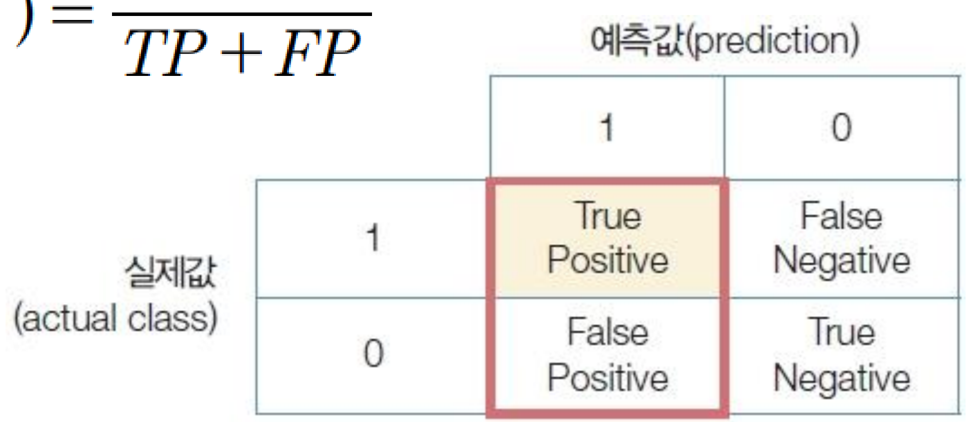



그림 9-8 정밀도 측정

# 02 분류 문제의 성능지표

- 민감도(recall) : 실제 1인 값을 가진 데이터를 모델이 얼마나 1 이라고 잘 예측했는지에 대한 비율
  - 반환율 또는 재현율이라고도 부름

$$RECALL(TPR) = \frac{TP}{TP + FN}$$

| | | 예측값(prediction) | |
|---|---|---|---|
| | | 1 | 0 |
| 실제값 (actual class) | 1 | True Positive | False Negative |
| | 0 | False Positive | True Negative |

그림 9-9 민감도 측정

# 02 분류 문제의 성능지표

- F1 스코어(F1 score) : 정밀도와 민감도의 조화평균 값

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall}$$

In [5]:
```
import numpy as np
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score

y_pred = np.array([0, 1, 1, 0, 1, 1, 1, 0])
y_true = np.array([0, 1, 0, 0, 0, 0, 1, 1])
```

In [6]:
```
precision_score(y_true, y_pred) # 정밀도
```

Out [6]:
```
0.4
```

In [7]:
```
recall_score(y_true, y_pred)    # 민감도
```

Out [7]:
```
0.6666666666666666
```

In [8]:
```
f1_score(y_true, y_pred)        # F1 스코어
```

Out [8]:
```
0.5
```

# 02 분류 문제의 성능지표

- [프로그램 3-5]는 모델 선택 제외
  - 08행: train_test_split 함수로 훈련 60%, 테스트 40%로 랜덤 분할
  - 12행: 훈련 집합 x_train, y_train을 fit 함수에 주어 학습 수행
  - 14행: 테스트 집합의 특징 벡터 x_test를 predict 함수에 주어 예측 수행
  - 17~20행: 테스트 집합의 레이블 y_test를 가지고 혼동 행렬 계산

| 프로그램 3-5 | 필기 숫자 인식 – 훈련 집합으로 학습하고 테스트 집합으로 성능 측정 |
|---|---|

```
01  from sklearn import datasets
02  from sklearn import svm
03  from sklearn.model_selection import train_test_split
04  import numpy as np
05
06  # 데이터셋을 읽고 훈련 집합과 테스트 집합으로 분할
07  digit=datasets.load_digits()
08  x_train,x_test,y_train,y_test=train_test_split(digit.data,digit.target,train_size=0.6)
09
```

# 02 분류 문제의 성능지표

예) 부류 3에 속하는 75개 샘플 중 73개를 3, 1개를 2, 1개를 7로 인식

```python
# svm의 분류 모델 SVC를 학습
s=svm.SVC(gamma=0.001)
s.fit(x_train,y_train)

res=s.predict(x_test)

# 혼동 행렬 구함
conf=np.zeros((10,10))
for i in range(len(res)):
    conf[res[i]][y_test[i]]+=1
print(conf)

# 정확률 측정하고 출력
no_correct=0
for i in range(10):
    no_correct+=conf[i][i]
accuracy=no_correct/len(res)
print("테스트 집합에 대한 정확률은", accuracy*100, "%입니다.")
```

```
[[76.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0. 78.  0.  0.  0.  0.  0.  0.  3.  0.]
 [ 0.  0. 66.  1.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0. 73.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0. 63.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0. 70.  0.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.  0. 77.  0.  0.  0.]
 [ 0.  0.  0.  1.  0.  0.  0. 77.  0.  1.]
 [ 0.  0.  0.  0.  0.  0.  0.  0. 74.  0.]
 [ 0.  0.  0.  0.  0.  1.  0.  0.  0. 56.]]

테스트 집합에 대한 정확률은 98.74826147426981%입니다.
```

# 03
# 로지스틱 회귀 구현하기

# 03 로지스틱 회귀 구현하기

```
df2 = pd.get_dummies(df, columns = ['HeartDisease','Smoking',
                                     'AlcoholDrinking','Stroke',
                                     'DiffWalking','Sex',
                                     'AgeCategory','Race',
                                     'Diabetic','PhysicalActivity',
                                     'GenHealth','Asthma',
                                     'KidneyDisease','SkinCancer']
                     ,drop_first=True
                    )

df2.head()
```

| | BMI | PhysicalHealth | MentalHealth | SleepTime | HeartDisease_Yes | Smoking_Yes | AlcoholDrinking_Yes |
|---|---|---|---|---|---|---|---|
| **0** | 16.60 | 3.0 | 30.0 | 5.0 | 0 | 1 | 0 |
| **1** | 20.34 | 0.0 | 0.0 | 7.0 | 0 | 0 | 0 |
| **2** | 26.58 | 20.0 | 30.0 | 8.0 | 0 | 1 | 0 |
| **3** | 24.21 | 0.0 | 0.0 | 6.0 | 0 | 0 | 0 |
| **4** | 23.71 | 28.0 | 0.0 | 8.0 | 0 | 0 | 0 |