


머신러닝 - 알고리즘



- scikit-learn은 2007년 구글 썸머 코드에서 처음 구현됐으며 현재 파이썬으로 구현된 가장 유명한 기계 학습 오픈 소스 라이브러리다. scikit-learn의 장점은 라이브러리 외적으로는 scikit 스택을 사용하고 있기 때문에 다른 라이브러리와의 호환성이 좋다. 내적으로는 통일된 인터페이스를 가지고 있기 때문에 매우 간단하게 여러 기법을 적용할 수 있어 쉽고 빠르게 최상의 결과를 얻을 수 있다.
- 라이브러리의 구성은 크게 지도 학습, 비지도 학습, 모델 선택 및 평가, 데이터 변환으로 나눌 수 있다(scikit-learn 사용자 가이드 참조, http://scikit-learn.org/stable/user_guide.html). 지도 학습에는 서포트 벡터 머신, 나이브 베이즈(Naïve Bayes), 결정 트리(Decision Tree) 등이 있으며 비지도 학습에는 군집화, 이상치 검출 등이 있다. 모델 선택 및 평가에는 교차 검증(cross-validation), 파이프라인(pipeline)등 있으며 마지막으로 데이터 변환에는 속성 추출(Feature Extraction), 전처리(Preprocessing)등이 있다.

<https://scikit-learn.org/stable/>

[Install](#) [User Guide](#) [API](#) [Examples](#) [More](#)

scikit-learn

Machine Learning in Python

[Getting Started](#) [Release Highlights for 0.23](#) [GitHub](#)

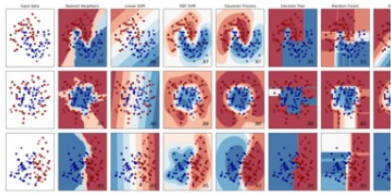
- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying which category an object belongs to.

Applications: Spam detection, image recognition.

Algorithms: SVM, nearest neighbors, random forest, and more...



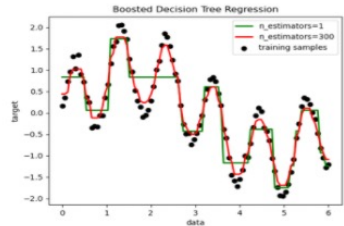
Examples

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.

Algorithms: SVR, nearest neighbors, random forest, and more...




Examples

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes

Algorithms: k-Means, spectral clustering, mean-shift, and more...



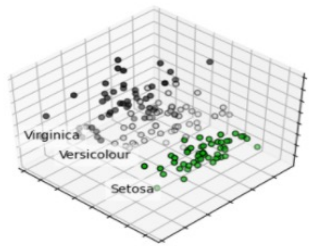
Examples

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency

Algorithms: k-Means, feature selection, non-negative matrix factorization, and more...



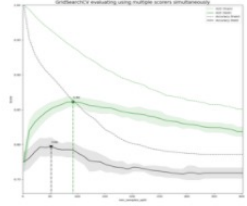
Examples

Model selection

Comparing, validating and choosing parameters and models.

Applications: Improved accuracy via parameter tuning

Algorithms: grid search, cross validation, metrics, and more...



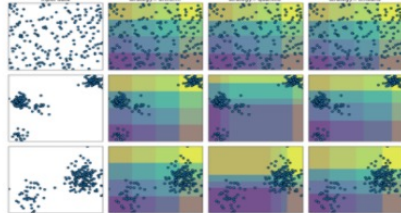
Examples

Preprocessing

Feature extraction and normalization.

Applications: Transforming input data such as text for use with machine learning algorithms.

Algorithms: preprocessing, feature extraction, and more...



Examples

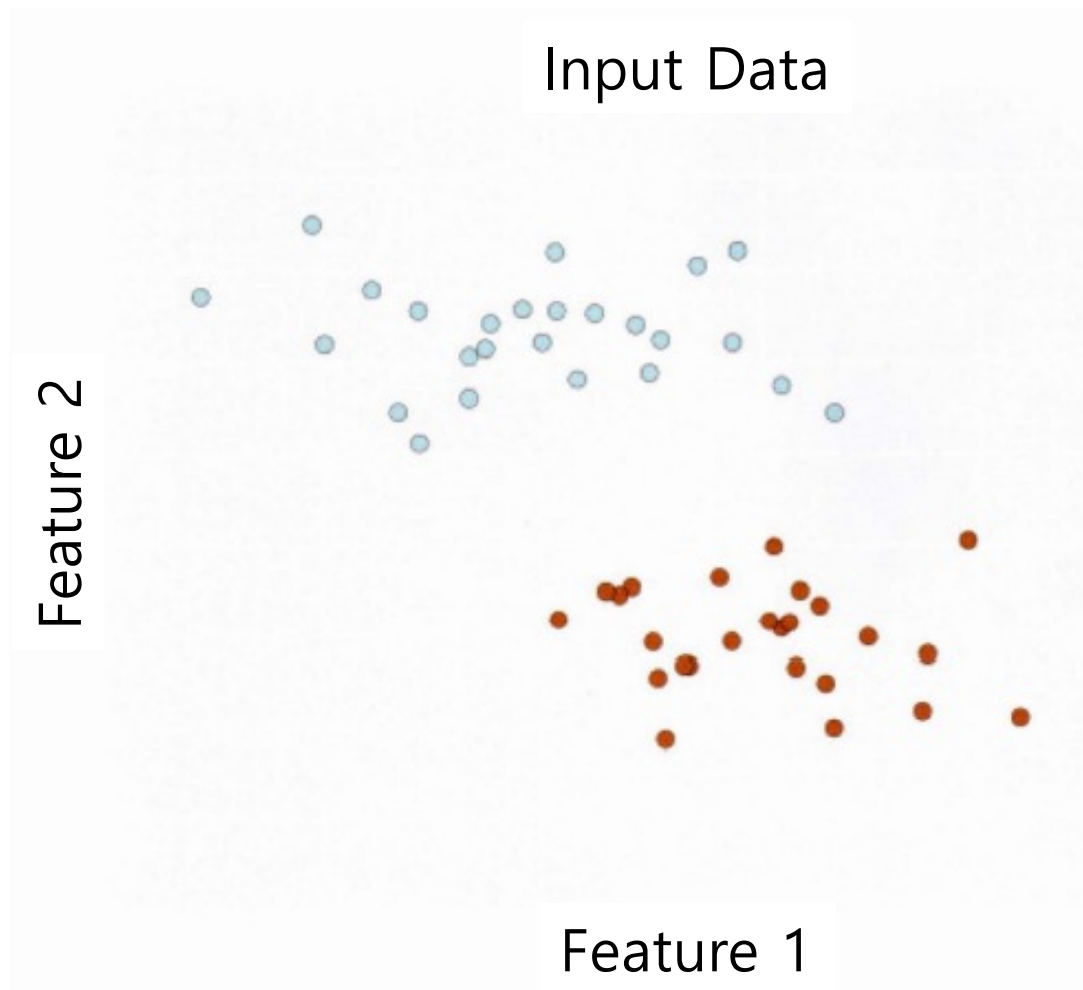
머신러닝의 일반적 유형

분류 : 이산적인 레이블 예측하기

레이블이 있는 점의 집합이 주어지고,
그것들을 사용해 레이블이 없는 점을
분류하는 작업

왼쪽의 예제는 2차원 데이터를 사용한다.
즉, 각 점이 평면상 점의 위치 (x, y)로 표시
되는 두 개의 특징 (feature)을 가진다.
이 밖에도 각 점은 두 개의 클래스 레이블
중 하나를 가지고 있는데, 여기서는 점의
색상으로 표시된다.

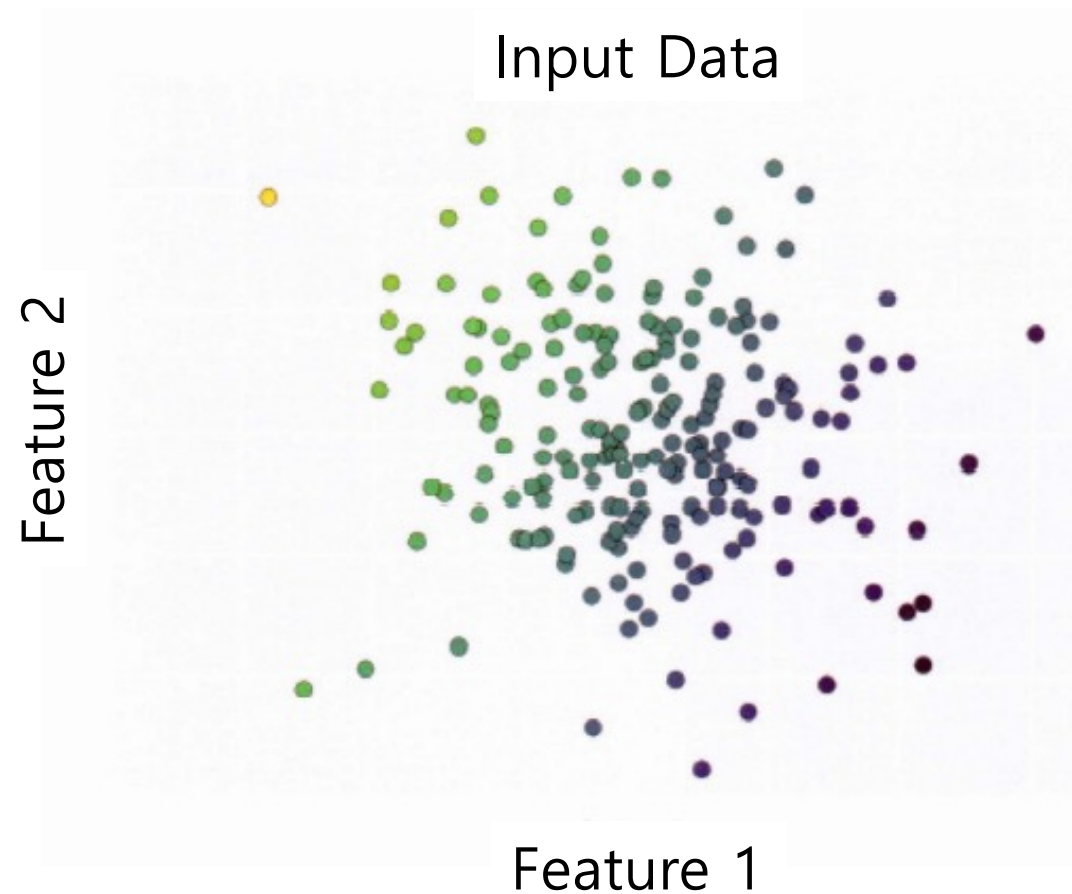
이 특징과 레이블로부터 새 점이 '파란색'
이나 '빨간색' 중 어느 레이블을 가져야
하는지 결정하는 모델을 만들고자 한다.



머신러닝의 일반적 유형

회귀 : 연속적인 레이블 예측하기

레이블이 연속적인 수량인 간단한 회귀 작업을 살펴보자.

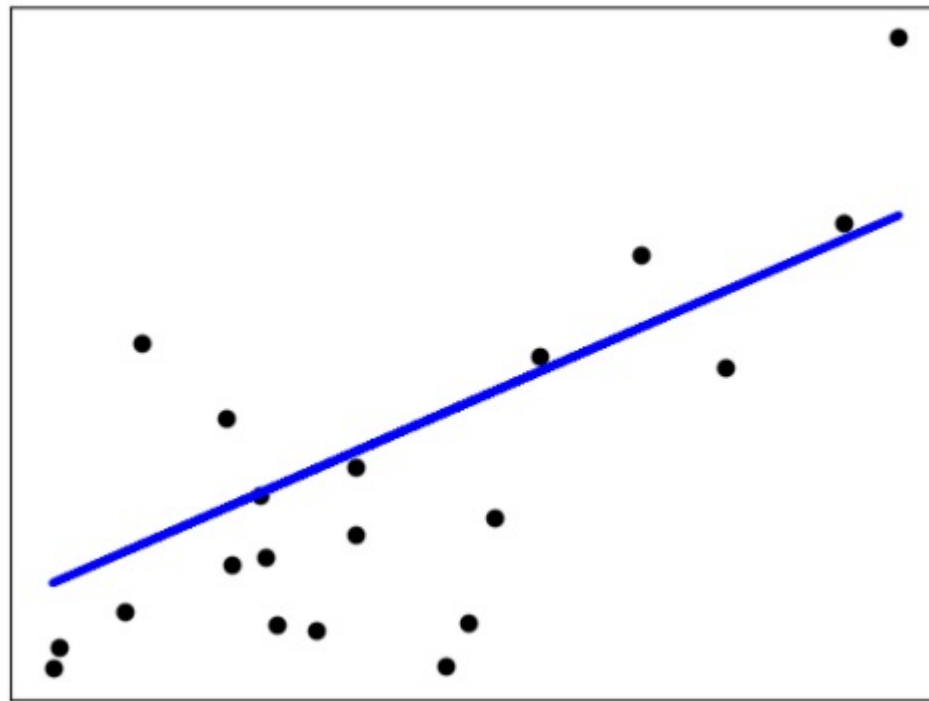


머신러닝의 주요 지도학습 알고리즘

- k-최근접 이웃 k-nearest neighbors
- 선형 회귀 linear regression
- 로지스틱 회귀 logistic regression
- 서포트 벡터 머신 support vector machine (SVM)
- 결정 트리 decision tree와 랜덤 포레스트 random forest
- 신경망 neural networks⁶

Linear Regression

Linear Regression은 단일 변수나 다변수 x 에 대해 w_0, w_1, \dots, w_p 값을 찾아서 \hat{y} 를 정확하게 예측하는 문제이다.



$$\hat{y}(w, x) = w_0 + w_1 x_1 + \dots + w_p x_p$$

$w = (w_1, \dots, w_p)$ as `coef_` and w_0

Linear Regression

- X와 Y데이터를 생성 후,
- 모델 선언 / 학습 / 예측의 순서로 진행한다.

```
# 모델 선언  
model = LinearRegression()
```

```
# 학습  
model.fit(x, y)
```

```
# 예측  
prediction = model.predict([[10.0]])
```

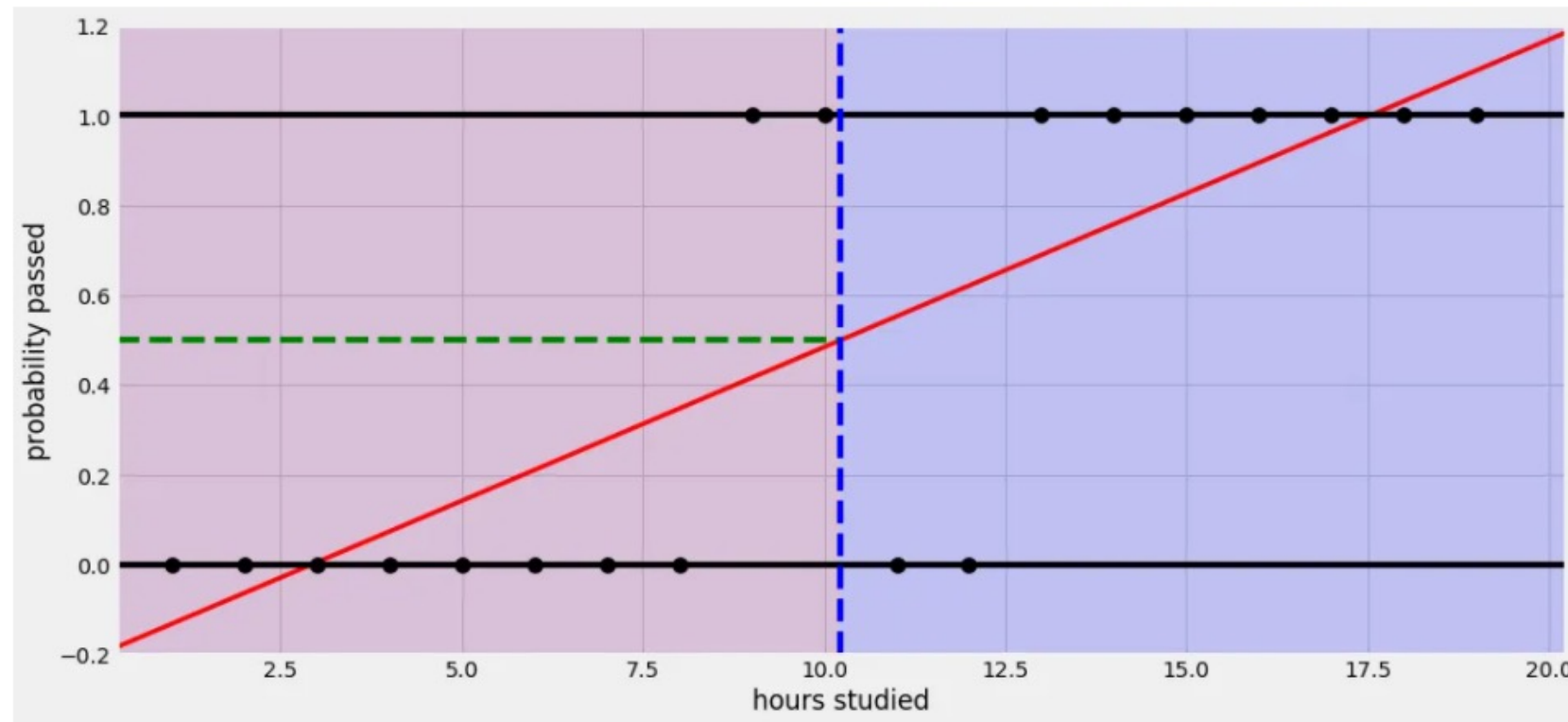
[ex1_sklearn_start.ipynb](#) 파일 참조

Logistic Regression

- 로지스틱 회귀는 회귀를 사용하여 데이터가 어떤 범주에 속할 확률을 0에서 1 사이 (Logistic Regression)의 값으로 예측하고 그 확률에 따라 가능성이 더 높은 범주에 속하는 것으로 분류해주는 **지도 학습** 알고리즘이다.
- 스팸 메일 분류기 같은 예시를 생각하면 쉽다. 어떤 메일을 받았을 때 그것이 스팸일 확률이 0.5 이상이면 spam으로 분류하고, 확률이 0.5보다 작은 경우 ham으로 분류하는 거다. 이렇게 데이터가 2개의 범주 중 하나에 속하도록 결정하는 것을 **2진 분류(binary classification)**라고 한다.

Logistic Regression

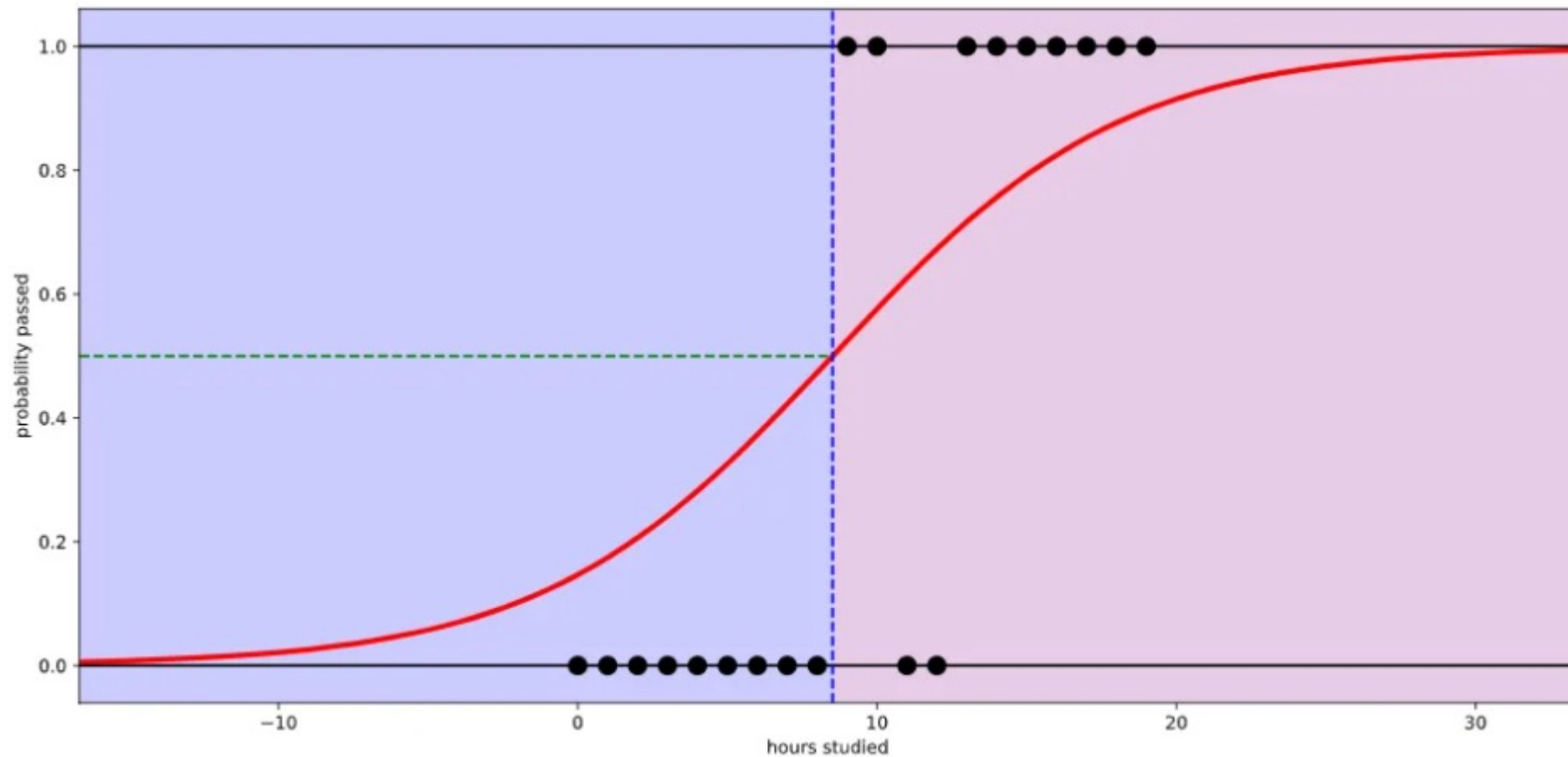
- 예를 들어 어떤 학생이 공부하는 시간에 따라 시험에 합격할 확률이 달라진다고 해보자. 선형 회귀를 사용하면 아래와 같은 그림으로 나타낼 수 있다.



- 공부한 시간이 적으면 시험에 통과 못하고, 공부한 시간이 많으면 시험에 통과한다는 식으로 설명할 수 있다. 그런데 이 회귀선을 자세히 살펴보면 **확률이 음과 양의 방향으로 무한대까지 뻗어 간다**. 말 그대로 '선'이라서. 그래서 공부를 2시간도 안 하면 시험에 통과할 확률이 0이 안 된다. 이걸 말이 안 된다.

Logistic Regression

- 로지스틱 회귀를 사용하면 아래와 같이 나타난다.
- 시험에 합격할 확률이 0과 1사이의 값으로 그려진다.



Logistic Regression

로지스틱 모형 식은 독립 변수가 $[-\infty, \infty]$ 의 어느 숫자이든 상관 없이 종속 변수 또는 결과 값이 항상 범위 $[0, 1]$ 사이에 있도록 한다. 이는 오즈(odds)를 로짓(logit) 변환을 수행함으로써 얻어진다.

1. odds

성공 확률이 실패 확률에 비해 몇 배 더 높은가를 나타내며 그 식은 아래와 같다.

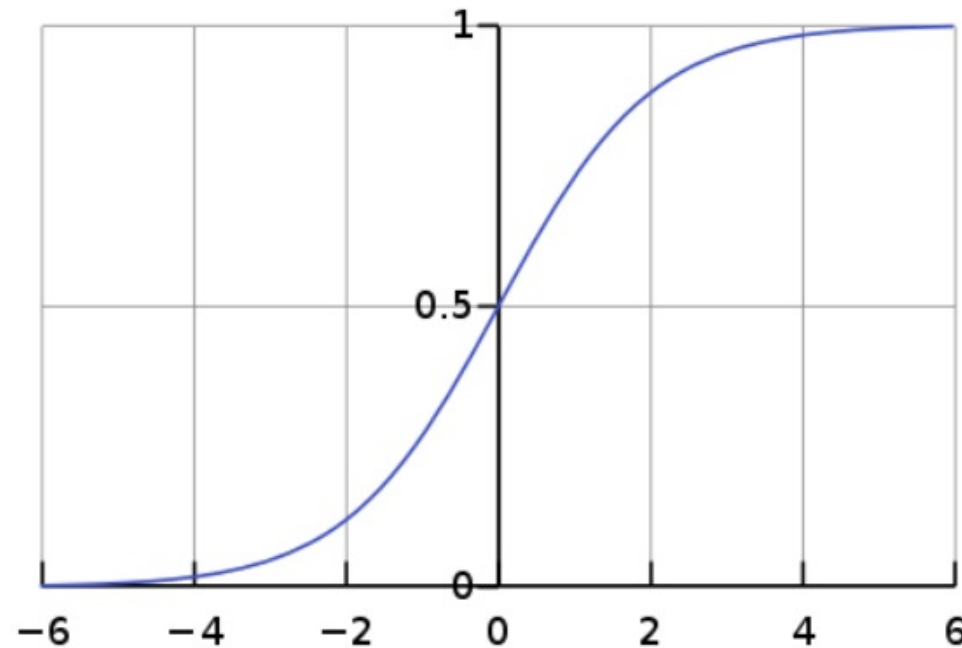
$$\text{odds} = \frac{p(y = 1|x)}{1 - p(y = 1|x)}$$

2. 로짓변환

오즈에 로그를 취한 함수로서 입력 값의 범위가 $[0, 1]$ 일때 출력 값의 범위를 $[-\infty, \infty]$ 로 조정한다.

$$\text{logit}(p) = \log \frac{p}{1 - p}$$

Logistic Regression



- 로지스틱 함수 (logistic function)

로지스틱 함수의 그래프는 Figure 1과 같고 이는 독립 변수 x 가 주어졌을 때 종속 변수가 1의 범주에 속할 확률을 의미한다. 즉, $p(y = 1|x)$ 를 의미한다.

로지스틱 함수는 로짓 변환을 통해 만들어지고, 그 형태는 다음과 같다.

$$\text{logistic function} = \frac{e^{\beta \cdot X_i}}{1 + e^{\beta \cdot X_i}}$$

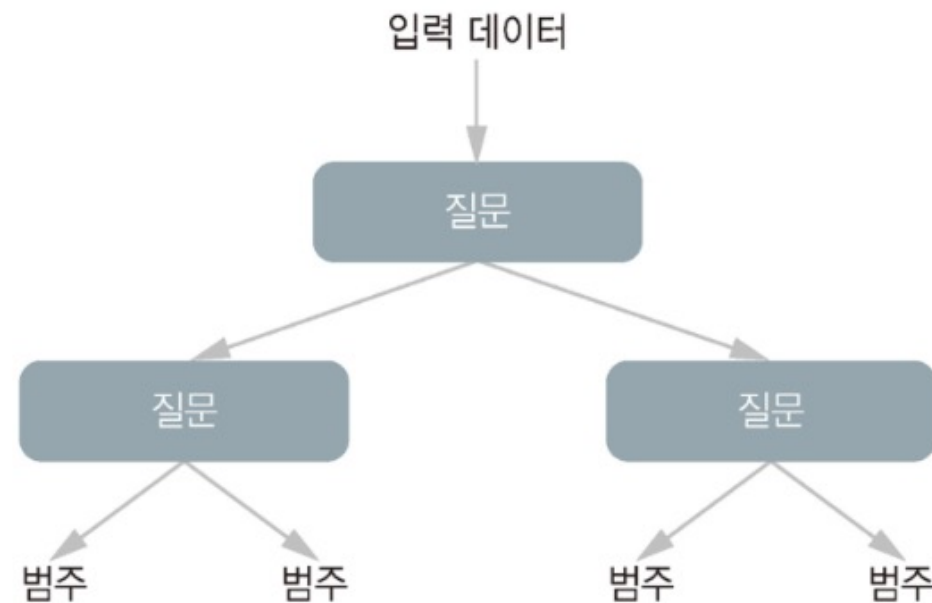
Logistic Regression

참조 사이트

- https://ko.wikipedia.org/wiki/%EB%A1%9C%EC%A7%80%EC%8A%A4%ED%8B%B1_%ED%9A%8C%EA%B7%80
- <http://hleecaster.com/ml-logistic-regression-concept/>

결정 트리 (Decision tree)

- 플로차트와 같은 구조를 가지며 입력 데이터 포인트를 분류하거나 주어진 입력에 대해 출력 값을 예측합니다. 결정 트리는 시각화하고 이해하기 쉽습니다.

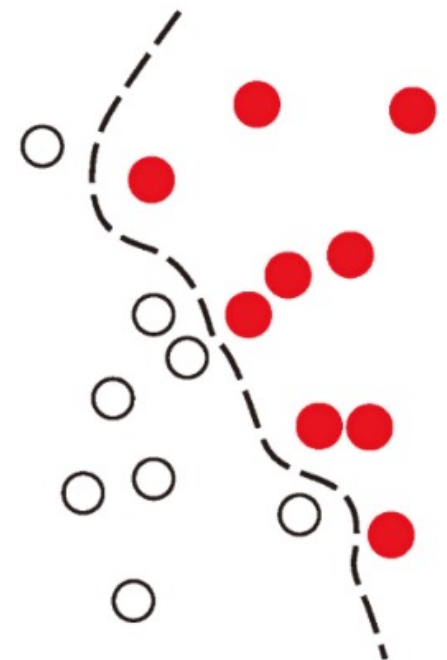


학습된 파라미터는 데이터에 관한 질문으로,
“데이터에 있는 두번째 특성이 3.5보다 큰가?”
같은 질문이 될 수 있다.

SVM

현대적인 SVM의 공식은 1995년에 벨 연구소에서 공개되었습니다. SVM은 분류 문제를 해결하기 위해 2개의 다른 범주에 속한 데이터 포인트 그룹 사이에 좋은 결정 경계를 찾습니다. 결정 경계는 훈련 데이터를 2개의 범주에 대응하는 영역으로 나누는 직선이나 표면으로 생각할 수 있습니다. 새로운 데이터 포인트를 분류하려면 결정 경계 어느 쪽에 속하는지를 확인하면 됩니다.

결정 경계

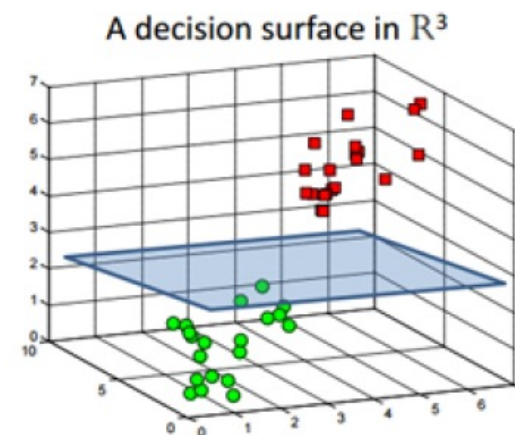
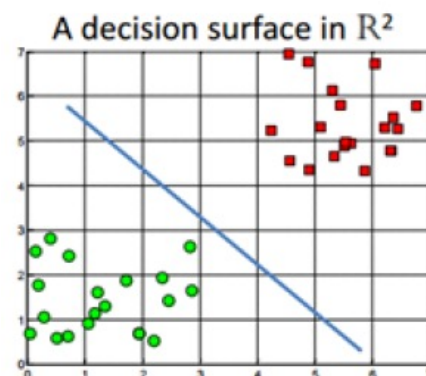


SVM이 결정 경계를 찾는 과정은 두 단계입니다.

1. 결정 경계가 하나의 초평면(hyperplane)으로 표현될 수 있는 새로운 고차원 표현으로 데이터를 매핑합니다(그림 1-10과 같은 2차원 데이터라면 초평면은 직선이 됩니다).
2. 초평면과 각 클래스의 가장 가까운 데이터 포인트 사이의 거리가 최대가 되는 최선의 결정 경계(하나의 분할 초평면)를 찾습니다. 이 단계를 **마진 최대화**(maximizing the margin)라고 부릅니다. 이렇게 함으로써 결정 경계가 훈련 데이터셋 이외의 새로운 샘플에 잘 일반화되도록 도와줍니다.

초평면(Hyperplane)이란?

SVM은 보통 범주 값을 유사한 데이터들로 그룹짓기 위해 초평면(hyperplane)이라는 경계를 사용하는데 초평면은 $n-1$ 차원의 subspace를 의미하는 것이며, 3차원의 경우 초평면은 2차원의 면이 되고, 2차원의 경우 초평면은 1차원의 선이 된다.



랜덤 포레스트

의사결정트리 학습법은 좋은 방법이지만, 주어진 데이터에 따라 생성되는 의사결정트리가 매우 달라져서 일반화하여 사용하기 어렵고, 의사결정트리를 이용한 학습 결과 역시 성능과 변동의 폭이 크다는 단점이 있습니다.

이에 단점을 보완한 방법이 랜덤 포레스트 입니다.

1. 주어진 트레이닝 데이터에서 무작위로 중복을 허용해서 n 개를 선택합니다.
2. 선택한 n 개의 데이터 샘플에서 데이터 특성값을 중복 허용없이 d 개 선택합니다.
3. 1~2단계를 k 번 반복합니다.
4. 1~3단계를 통해 생성된 k 개의 의사결정트리를 이용해 예측하고, 예측된 결과의 평균이나 가장 많이 등장한 예측 결과를 선택하여 최종 예측값으로 결정합니다.

1단계에서 무작위로 중복을 허용해서 선택한 n 개의 데이터를 선택하는 과정을 부트스트랩이라고 부르며, 부트스트랩으로 추출된 n 개의 데이터를 부트스트랩 샘플이라 부릅니다. scikit-learn이 제공하는 랜덤 포레스트 API는 부트스트랩 샘플의 크기 n 의 값으로 원래 트레이닝 데이터 전체 개수와 동일한 수를 할당합니다.

랜덤 포레스트

4단계에서 여러 개의 의사 결정트리로부터 나온 예측 결과들의 평균이나 다수의 예측 결과를 이용하여 방법을 앙상블기법이라고 합니다. 다수의 예측 결과를 선택하는 것은 다수 결의 원칙과 비슷합니다.

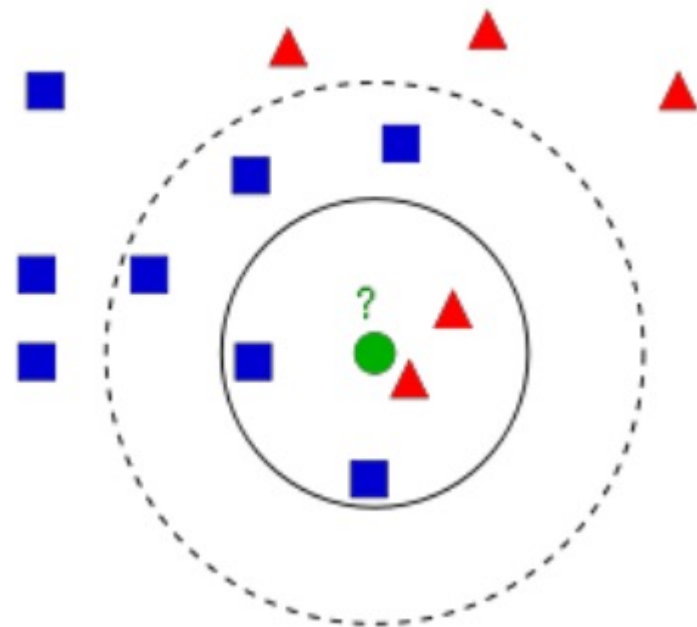
부트스트랩을 이용해 무작위 의사 결정트리의 집합인 랜덤 포레스트를 구성하는 것처럼 부트스트랩으로 다양한 분류기에 대해 앙상블 기법을 활용하여 특징적인 하나의 분류기로 구성하는 방법을 배깅(Bagging)이라고 합니다.

KNN(K-Nearest Neighbors)

지도학습에 활용되는 가장 단순한 종류의 알고리즘입니다.

KNN은 트레이닝 데이터로부터 예측을 위한 함수를 학습하는 것이 아니라 트레이닝 데이터 자체를 기억하는 것이 핵심입니다.

새로운 데이터와 가장 가깝게 위치하는 데이터가 속해 있는 그룹으로 분류합니다. 그러나 이는 오류를 발생할 수 있는 여지가 있습니다. 따라서 통계적인 분포를 보고 판단하게 되면 더 정확한 판단을 할 수 있습니다. 주변에 있는 데이터를 몇개를 보고 판단하느냐가 k 입니다. 작은 원의 경우 $k=4$ 인 경우이고, 점선의 경우 $k=7$ 입니다.



예측의 정확도 판단

Accuracy
Recall
Precision
F1 Score



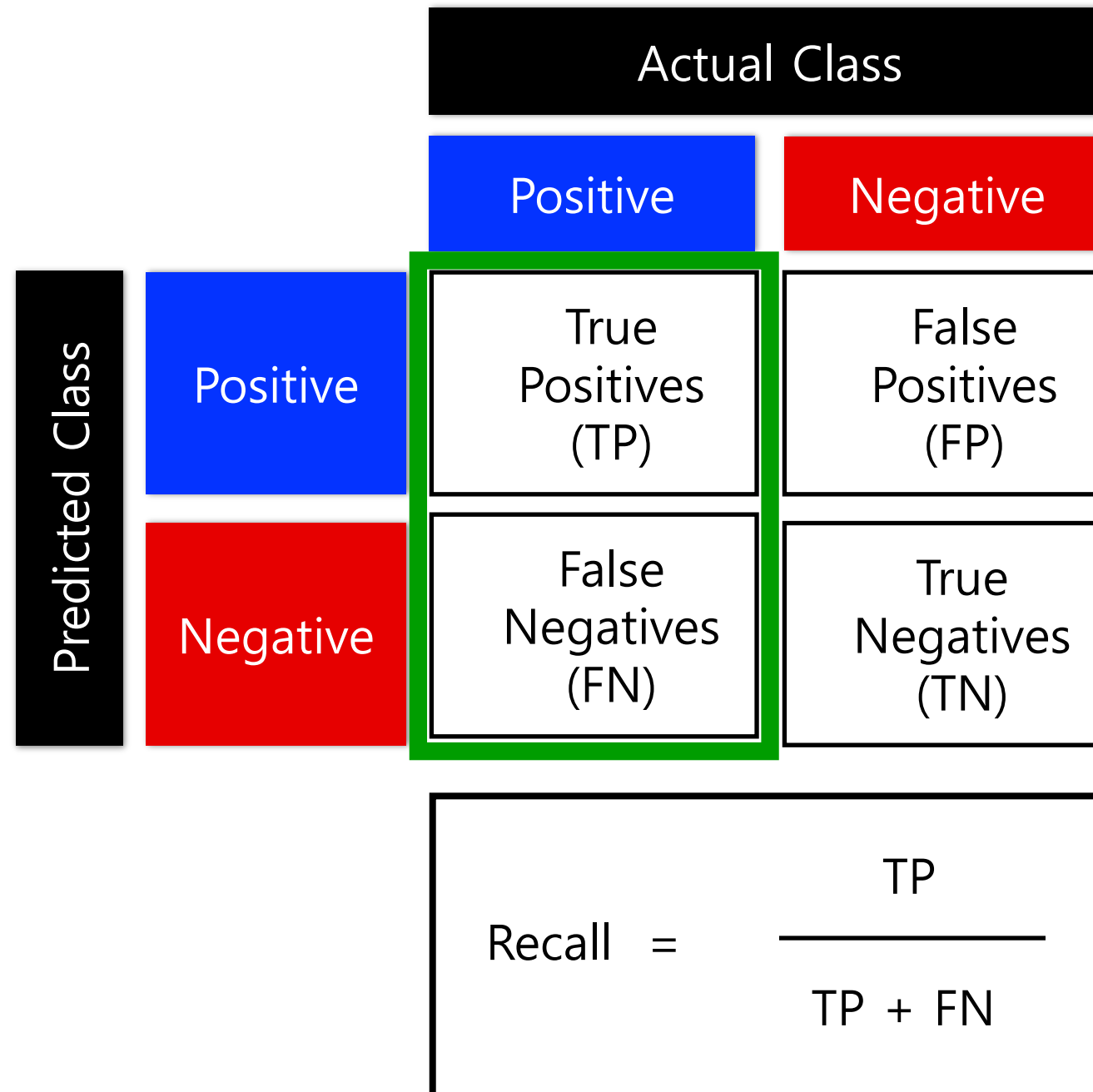
Accuracy

		Actual Class	
		Positive	Negative
Predicted Class	Positive	True Positives (TP)	False Positives (FP)
	Negative	False Negatives (FN)	True Negatives (TN)

$$\text{Accuracy} = \frac{\text{정확한 예측수}}{\text{총 예측수}} = \frac{TP + TN}{TP + TN + FP + FN}$$

검출 결과가 얼마나 정확한지를 알려준다.

Recall



검출 결과가 얼마나 정확한지를 알려준다.

Precision

		Actual Class	
		Positive	Negative
Predicted Class	Positive	True Positives (TP)	False Negatives (FN)
	Negative	False Positives (FP)	True Negatives (TN)

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

얼마나 잘 검출했는지를 알려준다.

양성 종양 VS 악성종양

종양은 일반적으로 임상 경과에 따라 크게 '양성종양(benign tumor)'과 '악성종양(malignant tumor)'으로 구분한다. 양성종양은 한정된 범위 내에서 성장하다가 중단되고, 피막에 싸여 침윤과 전이가 없어 예후가 좋지만, 악성종양은 그 반대로 암(cancer)이라 불린다. 'cancer'라는 단어 자체는 라틴어에서 유래한 것으로 '게(crab)'를 뜻하고, 혈관이 발달한 종양의 모습이 마치 '게 등딱지'처럼 침습하는 것처럼 보여 붙여진 이름이다.

출처 : https://health.chosun.com/healthyLife/column_view.jsp?idx=9768

특징	양성종양	악성종양 (암)
성장속도	느림	빠름
모양	경계가 분명	경계가 불분명
전이 여부	없음	흔함
대칭성	대칭	비대칭
경계	규칙적	불규칙적
색조	단일	다양함

종양의 양성과 악성 분류 예제

양성으로 분류된 종양과 악성으로 분류된 종양 모델 100개의 정확성 계산

참 양성(TP):

- 실제: 악성
- ML 모델 예측: 악성
- 참 양성 결과수: 1

허위 양성(FP):

- 실제: 양성
- ML 모델 예측: 악성
- 허위 양성 결과수: 1

허위 음성(FN):

- 실제: 악성
- ML 모델 예측: 양성
- 허위 음성 결과수: 8

참 음성(TN):

- 실제: 양성
- ML 모델 예측: 양성
- 참 음성 결과수: 90

$$\text{정확성} = \frac{TP + TN}{TP + TN + FP + FN} = \frac{1 + 90}{1 + 90 + 1 + 8} = 0.91$$

정확성은 0.91 또는 91%(총 100개의 예제 중 정확한 예측 91개)로 나타난다.

이는 종양 분류자가 악성 종양을 제대로 식별했음을 의미합니다.

종양의 악성과 양성 분류 예제

종양 예제 100개 중,
91개는 **양성**(TN : 90개, FP: 1개)이고,
9개는 **악성**(TP : 1개, FN : 8개)이다.

모델은 양성 종양 91개 중 90개를 양성으로 정확히
식별한다. 뛰어난 예측 능력처럼 보이지만, 악성 종양

9개 가운데 1개만 악성으로 식별한다. 악성 종양 9개 중 8개가 미확진 상태로 남았다는 것은 형
편없는 예측 결과이다.

언뜻 보기에는 91% 정확성이 좋아 보일 수 있지만 이 예제에서 **항상 양성으로 예측하는 다른
종양 분류자** 모델도 동일한 정확성(91/100의 정확한 예측)을 달성할 것이다. 다시 말해 이 모델
은 악성 종양과 양성 종양을 구분하는 예측 능력이 0인 모델과 비교해서 전혀 나을 바가 없다는
것이다.

이와 같이 **클래스 불균형 데이터 세트**를 사용하면 양성 라벨수와 음성 라벨수가 상당히 다르므
로 정확성만으로는 모든 것을 평가할 수 없다.

Predicted Class	Actual Class	
	양성 (Positive)	악성 (Negative)
양성 (Positive)	TP : 1개 실제 : 양성 예측 : 양성	FP : 1개 실제 : 양성 예측 : 악성
악성 (Negative)	FN : 8개 실제 : 악성 예측 : 양성	TN : 90개 실제 : 악성 예측 : 악성

종양의 악성과 양성 분류 예제

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{1}{1 + 1} = 0.5$$

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{1}{1 + 8} = 0.11$$

Predicted Class	Actual Class	
	악성 (Malignant)	양성 (Benign)
악성 (Malignant)	TP : 1개 실제 : 악성 예측 : 악성	FP : 1개 실제 : 양성 예측 : 악성
양성 (Benign)	FN : 8개 실제 : 악성 예측 : 양성	TN : 90개 실제 : 양성 예측 : 양성

클래스 불균형 데이터 세트

두 클래스의 레이블이 서로 크게 다른 빈도를 보이는 분류 문제에서 나타날 수 있다. 예를 들어 질병 데이터 세트에서 0.0001의 예가 긍정 라벨을 가지고 0.9999의 예가 부정 라벨을 가진다면 클래스 불균형 문제에 해당 한다.

Precision **VS** Recall

recall과 precision을 동시에 고려한 성능 지표

$$F_1 = \left(\frac{\text{recall}^{-1} + \text{precision}^{-1}}{2} \right)^{-1} = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$



Test Sample



Our Prediction

- 체리의 경우 정상을 위험물질로 과 탐지
- 톱, 칼의 경우 위험 물질인데 미 탐지

	Classified as	
	Positive	Negative
Positive	2	2
Negative	1	0

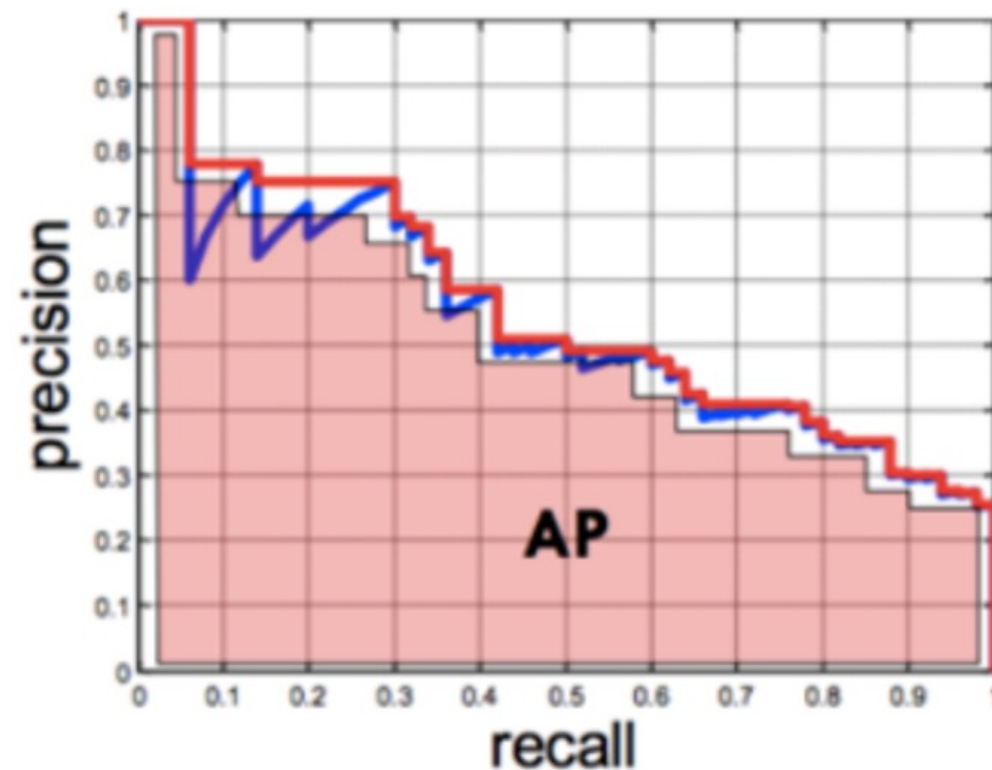
$$\text{Recall} = \frac{2}{(2 + 2)} = 0.5$$

$$\text{Precision} = \frac{2}{(2 + 1)} = 0.66$$

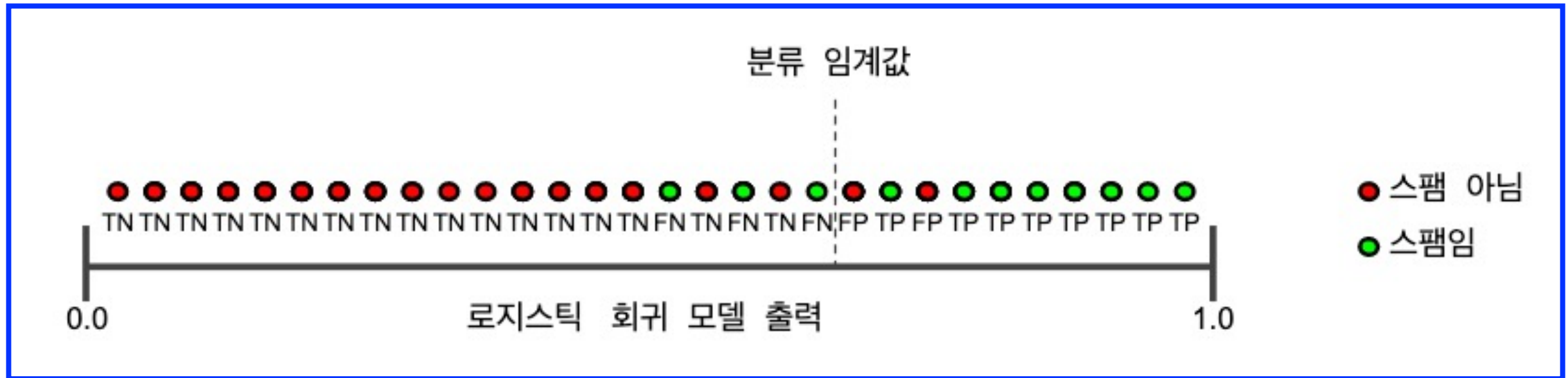
➡ **F1 score = 0.57**

Precision **VS** Recall

모델의 효과를 완전히 평가하려면 정밀도와 재현율을 모두 검사해야 한다.
그런데 정밀도와 재현율은 서로 상충하는 관계에 있는 경우가 많다.
즉, 정밀도가 향상되면 대개 재현율이 감소되고 반대의 경우도 마찬가지이다.



스팸메일 분류 예제

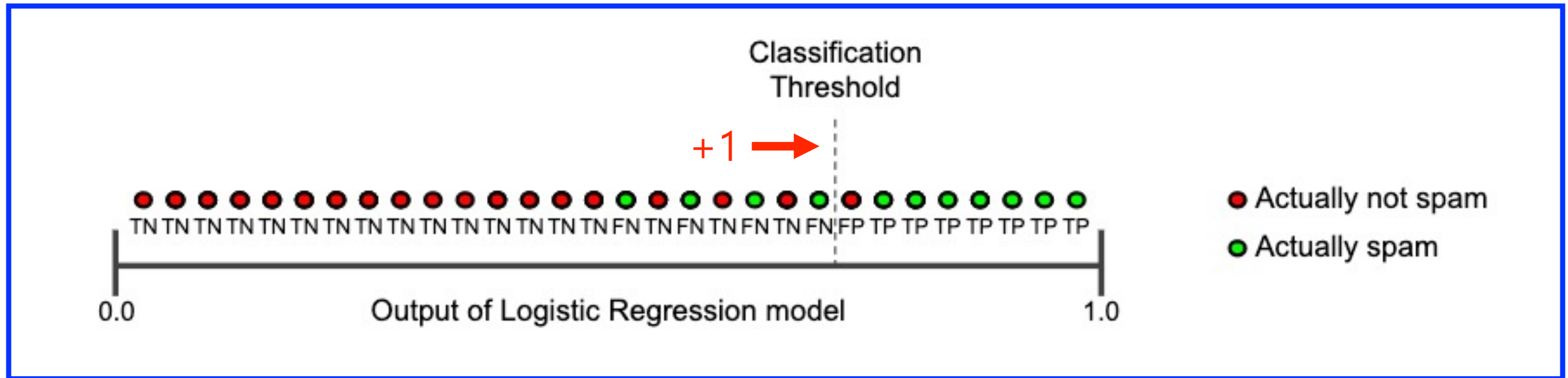


	Actual Class	
Predicted Class	TP : 8개 실제 : 스팸 예측 : 스팸	FP : 2개 실제 : 일반 예측 : 스팸
	FN : 3개 실제 : 스팸 예측 : 일반	TN : 17개 실제 : 일반 예측 : 일반

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{8}{8 + 2} = 0.8$$

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{8}{8 + 3} = 0.73$$

스팸메일 분류 예제

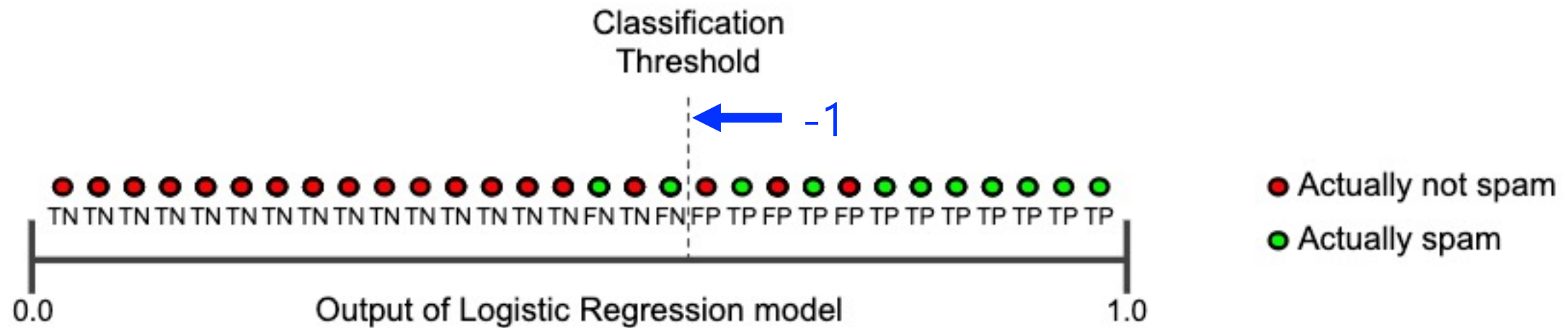


Predicted Class	Actual Class	
	TP : 7개 실제 : 스팸 예측 : 스팸	FP : 1개 실제 : 일반 예측 : 스팸
	FN : 4개 실제 : 스팸 예측 : 일반	TN : 18개 실제 : 일반 예측 : 일반

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{7}{7 + 1} = 0.88$$

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{7}{7 + 4} = 0.64$$

스팸메일 분류 예제



	Actual Class	
Predicted Class	TP : 9개 실제 : 스팸 예측 : 스팸	FP : 3개 실제 : 일반 예측 : 스팸
	FN : 2개 실제 : 스팸 예측 : 일반	TN : 16개 실제 : 일반 예측 : 일반

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{9}{9 + 3} = 0.75$$

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{9}{9 + 2} = 0.82$$

$$F_1 = \left(\frac{\text{recall}^{-1} + \text{precision}^{-1}}{2} \right)^{-1} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

F1 Score

: 정밀도와 재현율의 조화 평균

$$F_1 = \left(\frac{\text{recall}^{-1} + \text{precision}^{-1}}{2} \right)^{-1} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}.$$

앞에 내용을 함께 정리하면...

정확도(Accuracy) : 예측이 정답과 얼마나 정확한가

정밀도(Precision) : 예측한 것중에 정답의 비율

재현율(Recall) : 찾아야 할 것중에 실제로 찾은 비율

F1 Score : 정밀도와 재현율의 평균

F1 Score 값이 높으면 성능이 높다고 할 수 있다.