조건부 확률과 베이즈 정리

조건부확률

정의

어떤 사건 A가 일어날 확률은 다음과 같이 계산할 수 있다.

$$\mathrm{P(A)} = \frac{\text{사건 } A\text{가 일어나는 경우의 수}}{\text{일어날 수 있는 모든 경우의 수}}$$

A, B라는 사건이 있을 때, 두 사건이 동시에 일어날 확률은 P(A∩B)로 표현한다. 가령 감기에 걸리고 열이 날 확률은 P(감기∩열)이 되는 것이다.

이 때, 열이 날 때 이 열이 감기에 걸려서 나는 것인지에 대해 질문할 수 있다. 즉, 어떤 조건이 주어졌을 때, 다른 사건이 일어났을 확률을 묻는 것이다. 이는 조건부확률을 이용해 대답할 수 있다. B라는 사건이 일어난 것을 알 때, 즉 B가 조건으로 주어질 때 A라는 사건이 발생할 확률은 다음과 같이 계산한다.

$$\mathrm{P(A|B)} = \frac{P(A \cap B)}{P(B)}$$

이 식을 변형하여 아래와 같은 정리를 얻을 수 있다.

$$\mathrm{P(A \cap B) = P(A)P(B|A) = P(B)P(A|B)}$$

의미

조건부확률을 통해 관측된 결과나 현상이 어떤 원인에 의해 발생했는지를 확률로 나타낼 수 있다. 원인을 H, 결과를 D라고 할 때 이와 같은 P(H|D)를 사후확률이라고 한다. 반대로 P(H)는 사전확률이라고 한다. 이는 D라는 조건을 알기 전의 일반적인 확률이라 사전확률이라고 하며, P(H|D)는 D라는 조건을 알고 난 후 즉 D라는 사건이 발생한 이후이므로 사후확률이라고 한다. 한편, P(D|H)를 우도(Likelihood)라고 하며, 원인을 알 때 결과가 나타날 확률을 말한다. 가령 감기를 원인, 열을 결과라 하면 감기 환자가 열이 날 확률이 우도인 것이다.

베이즈 정리

정의

앞의 식을 적절히 잘 사용하여 다음과 같은 정리를 얻을 수 있다.

$$\mathrm{P(H|D)} = \frac{P(D|H)P(H)}{P(D)}$$

배가 아플 때, 왜 배가 아픈지에 대해 역학 조사를 한다고 하자. 가능한 원인은 여럿이 있겠지만 우선 점심 때 먹은 식당을 용의자로 본다고 하자. 그러면 우리가 구하고 싶은 것은 P(식당|복통)이며, 이 값이 일정 수준 이상인지를 판단해 따지러 갈지 안 갈지를 결정할 것이다. 하지만 이 확률을 직접 계산하는 것은 어렵다. 베이즈 정리를 사용하면 대신 비교적 알기 쉬운 확률을 통해 계산할 수 있다는 장점이 있다. 즉, 일반적으로 배가 아플 확률, 일반적으로 식당이 잘못일 확률, 그리고 식당이 잘못일 때 배가 아플 확률을 알면 된다. 일반적인 확률은 데이터에서 직접 계산할 수 있다. 또한 특정한 결과에 대해 역으로 원인을 찾는 것은 어렵지만, 원인을 알 때 특정한 결과가 나타날 확률은 구하기 쉽다.

따라서 베이즈 정리는 특정한 사건의 원인을 추정할 때, 직접 구하기는 어려운 확률을 간접적으로 우리가 잘 알고 있거나 구하기 쉬운 확률을 통해 계산할 수 있게 한다는데 의의가 있다.

사례: 질병 검사

T와 F를 각각 어떤 질병에 걸린, 걸리지 않은 사건이라고 하자. 그리고 D를 질병 검사가 양성인 사건이라고 하자. 그러면 질병검사를 받고 양성 판정을 받은 사람이 실제로 질병에 걸렸을 확률은 다음과 같다. P(D|Y)=0.98, P(D|N)=0.05, P(Y)=0.03, P(N)=0.97이라 주어질 때 해당 확률은 어떤 값이 나오는가?

$$\mathrm{P(Y|D)} = \frac{P(D|Y)P(Y)}{P(D)} = \frac{P(D|Y)P(Y)}{P(D|Y)P(Y) + P(D|N)P(N)}$$

나이브 베이즈 분류(Naïve Bayes Classification)

베이즈 분류는 특정한 데이터의 패턴에 대해 해당 패턴의 원인을 파악하는 기법으로, 보통 스팸 메일의 분류에 많이 사용된다. 어떤 단어들의 나열을 보고, 이 단어들이 과연 스팸 메일이기 때문에 나타난 것인지, 스팸 메일이 아닌 것인지 파악하는 것이다. 즉 P(스팸|단어들)와 P(정상|단어들)을 비교해보는 것이다.

독립과 나이브

어떤 두 사건이 독립이라는 것은, 각각의 사건 발생 유무가 다른 사건의 발생 확률에 영향을 미치지 않는다는 것을 의미한다. 가령 동전과 주사위를 던질 때 앞면과 2가 나올 확률은 서로 영향이 없기에 독립이다. 독립인 경우 두 사건이 같이 일어날 확률은 단순히 각자의 확률을 곱해서 구할 수 있다.

단어1과 단어2가 나타나는 사건이 서로 독립일 때, 두 단어가 나타났을 때 스팸 메일일 확률은 위의 성질을 이용하여 다음과 같이 나타낼 수 있다.

$$P\left(\text{스팸}\middle|\text{단어1, 단어2}\right)=P\left(\text{스팸}\middle|\text{단어1}\right)P\left(\text{스팸}\middle|\text{단어2}\right)$$

이는 단어1, 2뿐만 아니라 3, 4, 5, 등 독립이기만 한다면 계속해서 적용된다. 즉, 복잡한 결과에 대한 원인을 추정할 때도 단순히 한 결과에 대한 원인의 확률만 계산할 수 있으면 되는 것이다.

물론 이는 상당히 나이브한 가정인데, 왜냐하면 각각의 단어가 나타나는 사건이 실제로 독립인 경우는 거의 없기 때문이다. “바다”와 “이야기”라는 단어가 서로 독립이라고 보긴 힘들다. 또한 “경마”라는 단어는 그와 함께 “신규가입” 혹은 “안전” 같은 단어가 나오면 스팸일 확률이 더더욱 증가하지만, “성심”, “반대”, “서명” 등의 단어가 같이 나오면 스팸일 확률은 더더욱 감소한다.

나이브 베이즈 각 사건을 모두 독립이라고 가정하기 때문에 나이브하며, 한계가 존재한다. 그럼에도 불구하고 복잡한 문제를 간소화하고 나름대로 좋은 결과를 얻을 수 있다는 점에서 활용도가 높다. 특히, 확률 추정이 정확하지 않아도 분류 자체는 적당히 잘 하면 되기에 실용적으로는 더 높은 정확도를 보여줄 수 있다.

마르코프 연쇄(Markov Chain)

오늘의 날씨를 알 때, 내일, 모레, 3일 후의 날씨에 대해 예측한다고 하자. 미래의 날씨 예측은 당연히 이전의 날씨가 어떠했는가에 따라 결정된다. 이와 같이 어떤 시스템의 상태가 시간에 따라 확률적으로 변해가는 과정을 확률과정이라 하며, 이 중 그 확률이 오직 바로 전의 상태에만, 즉 내일의 날씨가 오늘의 날씨에만 영향을 받고 어제나 엊그제의 날씨에는 영향을 받지 않는 경우를 마르코프 과정이라고 한다.

마르코프 과정을 통해 오늘로부터 특정일 이후의 날씨에 대한 확률을 쉽게 계산할 수 있다. 가령 다음과 같은 표가 있다고 하자.

오늘 / 내일	맑음	비
맑음	0.9	0.1
비	0.4	0.6

이 때, 오늘이 맑았을 때 모레가 맑을 확률을 구해보자. 전체 경우의 수를 먼저 구하면 (1) 맑음-맑음-맑음, (2) 맑음-비-맑음이 될 것이다. (1)은 0.9*0.9*0.9 = 0.729이고, (2)는 0.081로 더하면 0.81이 나온다. 반대로 오늘 맑을 때 모레가 비가 올 확률은 1-0.81=0.19가 될 것이다.

이를 일반화하여 행렬로 나타낼 수 있다.

$$\mathbf{P^2}=\begin{bmatrix}0.9 & 0.1\\ 0.4 & 0.6\end{bmatrix}\begin{bmatrix}0.9 & 0.1\\ 0.4 & 0.6\end{bmatrix}=\begin{bmatrix}0.81 & 0.19\\ 0.6 & 0.4\end{bmatrix}$$

따라서 오늘 비가 왔을 때 모레가 맑을 확률은 0.6이라고 구할 수 있다.

미래의 상태가 과거가 아닌 오직 현재의 상태에만 영향을 받는다고 가정한다는 점에서 마르코프 연쇄도 나이브 베이즈 분류처럼 한계가 있으나, 그와 마찬가지로 복잡한 예측과 확률 계산을 간단한 과정을 통해 수행할 수 있고, 그럼에도 불구하고 적당히 괜찮은 결과를 내준다는 점에서 활용도가 높다.

분류

분류란

회귀분석과 마찬가지로 분류도 독립변수로부터 종속변수를 예측하는 기법이다. 그러나 회귀분석은 종속변수가 수치적인 값을 가지는 반면, 분류는 종속변수가 범주 데이터라는 것이 다르다. 성별, 직업, 나이 등으로부터 소득을 예측할 때, 연봉을 1000만, 2000만과 같은 수치로 예측값을 낼 경우 회귀분석이지만 2000만 이상, 2000만 이하와 같이 범주로 구분할 경우 분류가 된다.

장점

회귀분석과 달리 범주 데이터를 처리할 수 있으며, 독립변수들의 값을 통해 해당 데이터가 어떤 범주에 속할지를 판단할 수 있다. 데이터 분석 시나리오의 관점에서 볼 때 실제 기업이나 사회의 의사결정에서 필요한 것은 정확한 수치가 아닌 "좋은가, 나쁜가", "A? B? C? 어디에 속하지?"와 같은 질문에 대한 답이기 때문에 많은 서비스가 분류를 활용하고 있다.

단점

분류 기법을 적용해야 할 상황은 대개의 경우 독립변수의 수가 더 많고, 다양성이 더 크다. 따라서 데이터 전처리 과정이 더 필요하고, 모형이 복잡해질 가능성이 크다. 특히 독립변수의 종류에 비해 활용할 수 있는 데이터의 양이 상대적으로 적은 경우도 있어 모형 설계 및 학습에 이를 유념할 필요가 있다.

과정

시나리오 목표 달성을 위한 종속변수를 정의한 뒤, 관련된 독립변수를 찾는다. 회귀분석과 마찬가지로 가능한 모든 변수들에서 중요한 변수를 찾아 최적화하는 작업이 필요하다. 그러나 텍스트 분류, 영화 추천 등에서의 작업에서는 독립변수의 수가 매우 늘어날 수 있다. (각 단어의 출현 여부나 영화 관람, 만족 여부 등) 이후 데이터의 수와 독립변수의 수를 고려하여 모형을 정하고, 훈련한 뒤 모델을 평가하여 가장 좋은 것을 선택한다.

분류 기법

로지스틱 회귀

일반적인 선형회귀는 Y = aX + b와 같은 식으로 종속변수 Y의 값을 예측한다. 이때 X의 값의 변화에 따라 Y값은 (−∞, ∞)의 범위를 가진다. 그런데 분류의 경우, Y는 어떤 범주에 해당하는가(Y = 1) 해당하지 않는가 (Y = 0)이 되기 때문에 [0, 1]의 범위가 된다. 선형회귀를 사용할 경우 Y가 이 범위에 들어가지 않기 때문에 예측을 제대로 할 수 없다.

따라서 이를 해결하기 위해 적절한 변환을 적용할 필요가 있다. 우선 등장하는 것이 오즈비(Odds Ratio)의 개념으로, 어떤 사건의 성공확률이 실패확률에 비해 상대적으로 얼마나 큰지를 의미하며 다음과 같이 계산한다.

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

이 오즈비는 그러면 (0, ∞)의 범위를 가진다. 그러나 아직 음수의 경우가 해결이 되지 않았는데, 여기에 로그를 취하는 로짓(logit) 변화를 수행한다.

$$logit = log(\frac{p}{1-p}) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots$$

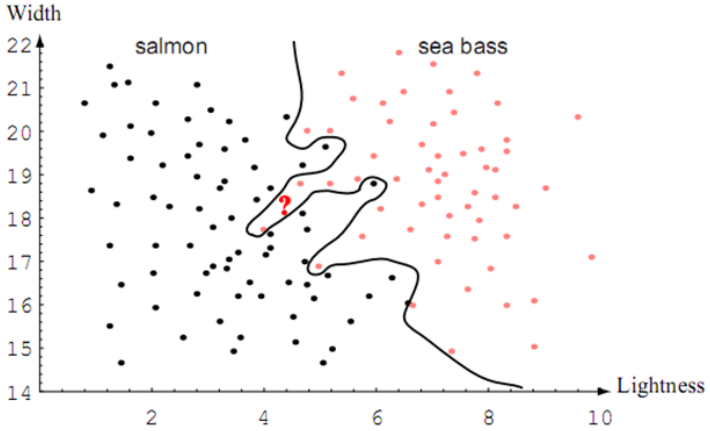
로그는 (−∞, ∞)의 범위를 가지기 때문에, 이제 좌변과 우변의 범위가 같아졌다. 이후 p, 즉 해당 x에 대해 y가 1에 해당할 확률을 구하려고 정리하면

$$\frac{e^{\beta_0+\beta_1 x_1+\beta_2 x_2+\cdots}}{1+e^{\beta_0+\beta_1 x_1+\beta_2 x_2+\cdots}}$$

이 된다. 이 함수는 0부터 1까지의 값을 가지며, 그래프 상에서는 S자 형태로 나타난다. 이러한 특징의 함수를 Sigmoid function이라고도 한다. 여기에 오차함수를 정의하고, 최적화를 통해 계수들을 구한다.

계수를 구하고 난 뒤, 실제 분류에서는 0.5와 같은 특정한 값을 기준으로, 함수에 독립변수들을 넣고 계산한 값이 그것보다 크면 Y = 1, 작으면 Y = 0으로 간주하여 분류를 진행한다. 이 기준치는 보통 0.5를 사용하지만, 상황에 따라 다른 값을 쓸 수도 있다.

오버피팅과 정규화



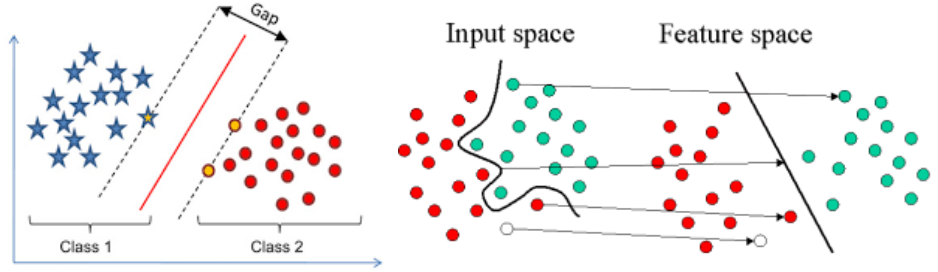
분류기의 학습을 많이 진행할수록 보통 좋은 결과가 나타나지만, 이를 실제로 적용했을 때 이전보다 더 좋지 않은 결과가 나타날 수도 있다. 이는 대부분의 머신러닝, 딥러닝 알고리즘에서 흔히 나타나는 오버피팅(over fitting)현상 때문이다. 훈련 데이터 중에서 일반적인 기준에서 벗어나는 이상치(아웃라이어)에 규칙을 맞추려다 보니 위의 그림처럼 오히려 전반적인 성능 측면에서는 더 떨어지게 되는 것이다.

오버피팅을 막기 위해서는 우선 데이터를 용도에 따라 구분할 필요가 있다. 모든 머신러닝 시나리오에서는 훈련용(train) 데이터와 최종 성능 확인을 위한 테스트(test) 데이터를 구분한다. 훈련용 데이터를 가지고 성능을 측정하면 오버피팅을 하든 무엇을 하든 훈련만 완벽하게 하면 점수가 높게 나오기 때문이다. 마치 시험문제를 미리 다 풀어보고 시험을 치는 것과 같다. 따라서 공부할 때 쓸 문제와 시험문제를 구분하는 것이다. 그런데 이렇게 할 경우 훈련 중에 모형을 조정할 때 성능을 확인하기가 힘들다. 이를 위하 검증용(validation) 데이터를 구분한다. 즉, 문제집(train) -> 모의고사(validation) -> 수능(test)와 같은 방식이 된다. 오버피팅이 발생하면 train에서는 점수가 올라가지만 validation에서는 점수가 떨어지게 되어, test 이전에 문제를 알고 모형을 수정하거나, 아니면 충분한 점수가 확보됐지만 아직 오버피팅이 나타나기 전에 훈련을 멈출 수도 있다.

오버피팅을 막는 다른 방법은 정규화(Regularization)이 있다. 이는 계수들의 값 분포를 일정하게 만들어주는 방식이다. 본래 오차는 예측된 Y값과 실제 Y값의 차이만 반영하여 정해진 함수를 통해 계산하지만, 정규화의 경우 여기에 계수들의 상대적인 분포차이를 반영하는 항을 추가한다. 따라서 설령 예측을 잘 하더라도 계수들의 값이 들쭉날쭉하면 오차가 크게 나타나게 되는 것이다.

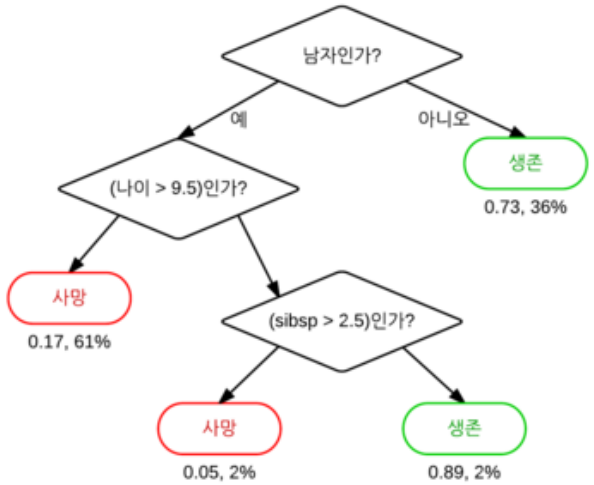
정규화에는 크게 L1과 L2가 있는데, L1은 각 계수들의 절대값의 합, L2는 각 계수들을 제곱한 것의 합이 된다. L2는 계산이 쉽고 정확한 해를 찾을 수 있다는 점에서 강점이 있다. 그러나 L1은 L2에 비해 계산은 어려우나 큰 장점이 있는데, 가령 100개의 변수가 있을 경우, L2는 변수들 사이의 계수가 골고루 퍼지게 할 수는 있어도 어떤 계수가 0이 되도록 (=그 변수를 아예 식에서 뺄 수 있게) 만들지는 않지만 L1은 가장 중요한 변수들 몇 외에 다른 변수는 0이 되도록 만드는 경향이 있어 유용하다.

SVM (Support Vector Machine)



좌측 그림처럼, 다른 부류의 데이터 사이를 가르는 어떤 경계선(벡터)를 그릴 수 있을 것이다. 이 때, 각 데이터로부터 가장 갭이 커지도록 하는 선을 찾을 수 있을 텐데 이를 Support Vector라고 하며, SVM은 이를 이용해 분류를 진행한다. 본래 SVM은 이처럼 직선으로 구분할 수 있는 경우에만 적용이 가능하나, 데이터의 좌표를 변환시키는 함수인 Kernel을 통해 우측 그림처럼 선형으로 구분이 안 되는 데이터를 이동시켜 적용할 수 있어 활용도를 높일 수 있다.

의사결정나무 (Decision Tree)



위는 타이타닉호의 탑승객 데이터로부터, 탑승객의 생존 여부를 예측하기 위한 의사결정나무의 한 예다. sibsp는 같이 탑승한 가족의 수를 의미하며, 나무의 각 끝, 즉 잎사귀 아래의 숫자는 좌측은 생존 확률, 우측은 전체 탑승객 중에서 해당 잎에 해당하는 탑승객의 비율 (혹은 어떤 탑승객이 해당 잎에 들어갈 확률)을 의미한다.

의사결정나무를 만들 때, 무엇을 기준으로 나누기 시작해야 할지를 정해야 한다. 알고리즘에 따라 세부적인 기준은 달라지지만, 공통적으로 생각해 볼 수 있는 것은 나누기 이전과 이후에 분포가 얼마나 극적으로 달라지는가를 생각해볼 수 있다. 가령 타이타닉호의 생존자 예측을 할 때, 처음 데이터 분포가 반반, 즉 생존 확률과 사망 확률이 둘 다 50%인 경우를 생각해보자. 이 때 남자를 기준으로 나누면 위처럼 50:50에서 64:36으로 갈리게 되면서 분포가 극적으로 변하지만, 이름의 길이가 홀수인가?와 같은 기준으로 나누면 50:50에서 거의 변하는 것이 없을 것이다. 따라서 남자라는 변수는 판단에 영향을 끼치는 중요한 변수지만 이름의 길이는 그렇지 않은 것이다.

의사결정나무는 다른 기법에 비해 직관적으로 이해가 쉽고, 빠르게 결과가 나오지만 정확도가 낮고 바보 같은 결과물이 나오는 경우도 있다.

랜덤 포레스트 (Random Forest)

의사결정나무는 어떤 데이터에 대해서는 매우 높은 성능을 보이다가 데이터가 조금만 달라지면 아주 낮은 성능을 보이기도 하는 등 예측력이 널뛰기하는 한계가 있다. 그러나 비슷하지만 조금씩 다른 결정 트리를 여러 개 만들어 다수결로 결과를 예측한다면 그 한계를 극복할 수 있다. (트리가 여러 개 있어서 포레스트)

물론 같은 데이터를 가지고 트리를 계속해서 만들면 같은 트리가 또 나올 것이기 때문에, 랜덤 포레스트는 부트스트랩이란 방법을 사용한다. 원본 데이터가 100개가 있다면, 이 100개의 데이터에서 중복을 허락해서 100개의 데이터를 뽑아낸 뒤, 이 데이터로 하나의 트리를 만들고, 또 다시 100개를 뽑아 트리를 만들고 반복하는 식으로 포레스트를 구성한다. 각 트리는 랜덤하게 뽑혀진 데이터에 대해 오버피팅을 하는 경우가 많지만, 트리 사이에는 어떤 편향이 없이 임의로 각자 구성됐기 때문에 전체 포레스트는 오히려 오버피팅이나 이상값 등에 대해 좋은 성능을 보일 수 있게 된다. 일종의 집단지성이라고 볼 수 있다.

랜덤 포레스트는 사용이 무척 간단하고 계산이 빠르게 진행된다는 강점이 있다. 나무를 몇 개를 만들지, 나무의 깊이는 몇까지 허용할지 등을 정하기만 하면 되는 것이다. 그러나 의사결정나무와 달리 랜덤 포레스트는 해석이 불가능해진다. 물론 포레스트 안의 나무 하나하나는 직접 보면 이해할 수 있지만, 전체 포레스트는 수 천 개의 나무로 구성됐기 때문에 이를 이해하는 것은 불가능하다.

기법이 너무 많아요!

분류 한 번 하려고 하는데 알아야 할 것이 너무 많다고 생각할 수 있다. 제일 이상적인 것은 데이터의 특징과 목표에 따라 적절한 기법을 선택하여 사용하는 것이지만, 이는 많은 경험과 이론에 대한 깊이 있는 이해를 요구한다. 그리고 그런 경험과 능력을 갖추더라도 최적의 기법을 선택하는 것은 어려운 일이다.

따라서 제일 좋은 방법은 그냥 다 해보는 것이다. 최근에는 컴퓨터의 연산 능력이 매우 발전했고, 개인도 클라우드를 통해 최고 성능의 컴퓨터를 싼 값에 이용할 수 있기 때문에, 괜찮아 보이는 기법을 다 집어넣고 한 번에 훈련시킨 다음 뛰어난 것을 골라 사용하면 된다.

ROC Curve

분류기 평가

회귀분석의 경우 특정한 오차함수를 지정하고, 오차의 합이나 분포 등을 통해 모형을 평가하지만, 분류는 기준치를 0.5가 아닌 다른 값을 설정하는 것 등을 통해 같은 모형이지만 다른 결과를 낼 수 있다. 이를 크게 4가지 지표의 변화를 통해 평가할 수 있다. 예측 값이 1이면 Positive, 0이면 Negative이고, 예측과 실제가 맞으면 True, 틀리면 False가 된다.

	실제 값 1	실제 값 0
예측 값 1	True Positive (TP)	False Positive (FP)
예측 값 0	False Negative (FN)	True Negative (TN)

정확도 (Accuracy)

가장 많이 쓰이지만, 데이터가 불균형(0과 1에 속하는 데이터의 양이 다름)하거나 특정한 그룹에 더 중요도가 클 경우 효과적이지 않다.

$$accuracy = \frac{TP + TN}{Positive + Negative}$$

정밀도 (Precision)

긍정적으로 평가된, 즉 어떤 사건에 해당한다고 분류한 데이터 중 실제로 그 사건이 일어난 경우를 평가한다. 가령 질병검사 결과가 양성으로 나온 사람들 중 실제로 병에 걸린 비율을 정밀도를 통해 알 수 있다.

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

재현률 (Recall)

재현률은 실제로 해당 사건이 일어난 경우 중에서, 얼마나 많은 비율을 사건이 일어났을 것이라 예측한 비율을 의미한다. 질병검사의 경우 병에 걸린 사람들 중에서 검사결과가 양성으로 나온 사람의 비율이 된다.

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

F1-Score

정확도와 재현률을 같이 고려하는 지표다.

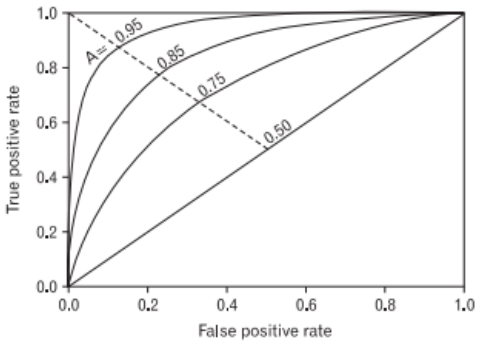
$$\mathrm{F1} = 2\frac{precision * recall}{precision + recall}$$

True Positive Rate / False Positive Rate

$$\mathrm{TPR} = \frac{TP}{postive}, \quad FPR = \frac{FP}{negative}$$

ROC Curve

ROC 곡선은 2차 세계대전에서 레이더 신호의 분석을 위해 고안됐다. ROC는 그림과 같이 가로축이 FPR, 세로축이 FPR인 그래프에서 기울기가 1인 직선보다 위에 위치한 곡선이 된다.



만약 기준치를 0.5보다 높게 잡을 경우, TPR은 올라가지만 FPR도 같이 올라가게 된다. 즉, 실제 병에 걸린 사람을 더 많이 찾지만 오진의 확률도 증가하는 것이다. 반대의 경우 오진의 확률은 줄지만 병에 걸린 사람을 찾아내지 못 하는 경우가 생길 것이다. 이처럼 TPR과 FPR은 함께 움직이게 되는데, 이 관계를 ROC 곡선을 통해 알 수 있다. 특히, ROC 곡선 아래의 면적을 AUC라고 하는데, 이 값은 0.5에서 1의 범위를 가지며, 1에 가까울수록 뛰어난 분류기라고 할 수 있다. ROC 곡선을 통해 분류기의 종합적인 성능 평과와 기준치 설정을 할 수 있다.