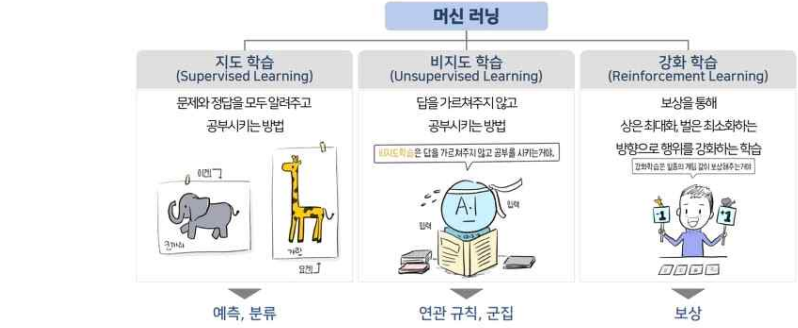
머신러닝(Machine Learning)

- 기계학습이라고도 불리는 머신러닝은 컴퓨터를 인간처럼 학습시킴으로써 인간의 도움 없이 컴퓨터가 스스로 새로운 규칙을 생성할 수 있지 않을까 하는 발상으로부터 시작
- 기계 학습(機械學習) 또는 머신 러닝(영어: machine learning)은 인공 지능의 한 분야로, 컴퓨터가 학습할 수 있도록 하는 알고리즘과 기술을 개발하는 분야



머신러닝(Machine Learning)

▶ 지도 학습(Supervised Learning)

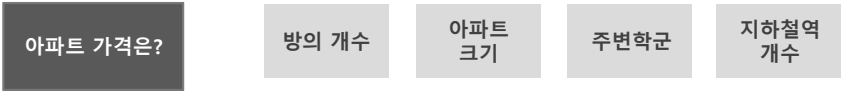
- 입력(Input, Feature)과 출력(Target)이 쌍으로 주어진 훈련 데이터(Training data)를 이용한 학습
- 레이블(Label)이 있는 데이터를 기반으로 입력 변수와 출력 변수를 매핑하는 함수를 찾은 후, 새로운 입력에 대한 예측을 수행
- **분류(Classification)/회귀(Regression)**

▶ 비지도 학습(Unsupervised Learning)

- 출력값 없이 입력값만으로 학습하는 방식
- 사람의 개입 없이 컴퓨터가 알아서 입력 데이터를 학습한 뒤, 패턴과 상관관계를 인식
- **군집 또는 클러스터링(Clustering/차원 축소(Dimensionality Reduction)**

회귀(regression)

▶ 회귀는 여러 독립변수와 한 개의 종속변수 간의 상관관계를 모델링하는 기법을 통칭한다.



$$Y = W_1*X_1 + W_2*X_2 + W_3*X_3 + \ldots + W_n*X_n$$

Y : 종속변수, 즉 아파트 가격

$X_1, X_2, X_3, \ldots X_n$: 방의개수, 아파트 크기, 주변 학군 등의 독립변수

$W_1, W_2, W_3, \ldots W_n$: 독립변수의 값에 영향을 미치는 회귀 계수(Regression Coefficients)

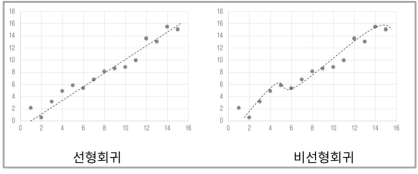
머신러닝 회귀 예측의 핵심은 주어진 피처와 결정값 데이터 기반에서 학습을 통해 최적의 회귀 계수를 찾아내는 것이다.

회귀의 유형

- 회귀는 회귀계수의 선형/비선형 여부, 독립변수의 개수, 종속변수의 개수에 따라 여러가지 유형으로 나눌 수 있다.
- 회귀에서 가장 중요한 것은 바로 회귀 계수이다. 이 회귀 계수가 '선형이다, 아니다 ' 에 따라 선형 회귀와 비선형 회귀로 나누어 진다.
- 독립변수의 개수가 한 개인지 여러 개인지에 따라 단일 회귀, 다중 회귀로 나뉜다.

독립변수 개수	회귀 계수의 결합
1개: 단일 회귀	선형: 선형 회귀
여러 개: 다중 회귀	비선형: 비선형 회귀

〈회귀 유형 구분〉



선형회귀가 비선형회귀 보다 성능이 좋음

분류(Classification)와 회귀(regression)

선형 회귀의 종류

▶ **독립변수의 개수와 회귀 계수의 선형/비선형 여부로 크게 4가지로 구분**

- **일반 선형 회귀** : 예측값과 실제값의 RSS(Residual Sum of Squares)를 최소화 하도록 회귀계수를 최적화, 규제(Regularization)를 적용하지 않음

- **릿지(Ridge)회귀** : 선형회귀에 L2 규제를 추가, L2 규제는 상대적으로 큰 회귀 계수 값의 예측 영향도를 감소시키기 위해 회귀 계수 값을 더 작게 만드는 모델

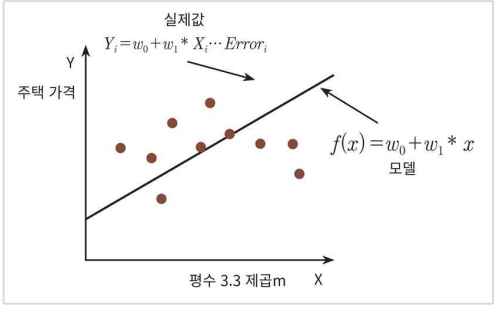
- **라쏘(Lasso)회귀** : 선형회귀에 L1 규제를 추가, L1 규제는 예측 영향력이 작은 피처의 회귀 계수를 0으로 만들어 예측 시 피처가 선택되지 않게 하는 모델(L1 규제는 피처 선택 기능으로도 불림)

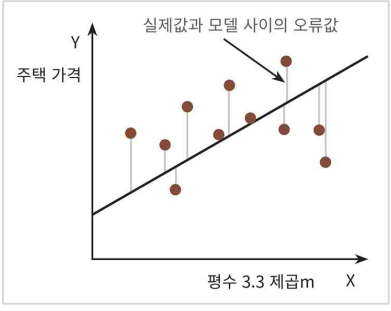
- **엘라스틱넷(ElasticNet)** : L2, L1 규제를 함께 결합한 모델. 주로 피처가 많은 데이터 세트에 적용, L1 규제로 피처를 줄이고 L2규제로 계수 값의 크기를 조정

- **로지스틱 회귀(Logstic Regression)** : 회귀라는 이름이 쓰이지만 분류에 사용되는 선형 모델. 이진분류, 희소 영역 분류(텍스트 분류) 등에 뛰어난 예측 성능을 보임

단순 선형 회귀를 통한 회귀의 이해

주택 가격이 주택의 크기만으로만 결정되는 단순 선형 회귀로 가정하면 다음과 같이 주택 가격은 주택 크기에 대해 선형(직선형태)의 관계로 표현할 수 있다.





최적의 회귀 모델을 만든다는 것을 바로 전체 데이터의 잔차(오류 값)의 합이 최소가 되는 모델을 만드다는 의미
동시에 오류 값 합이 최소가 될 수 있는 최적의 회귀 계수를 찾는 의미 이기도 함

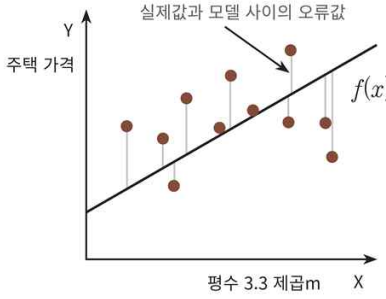
RSS 기반의 회귀 오류 측정

- 오류값은 +나 -가 될 수 있다. 따라서 (**오차의 합을 구하기 위해 단순히 더한다면 오류의 합이 작아질 수 있다.**)
- 이를 해결하기 위해서: (**절대값을 취해 더하**)거나, (**오류 값에 제곱을 구해서 합을 구**)한다.

RSS

RSS(Residual Sum of Square)

일반적으로 오류의 값을 제곱해서 더하는 방식으로, 미분 등의 계산을 편리하게 하기 위해서 RSS 방식으로 오류 합을 구한다. (**Error^2 = RSS**)



$$f(x) = w_0 + w_1 * x$$

RSS = (#1 주택 가격 – $(w_0+w_1$ #1 주택크기$)^2$

+ (#2 주택 가격 – $(w_0+w_1$ #2 주택크기$)^2$

+ (#3 주택 가격 – $(w_0+w_1$ #3 주택크기$)^2$

+ … (모든 학습 데이터에 대해 RSS 수행)

RSS(Residual Sum of Square) 이해

- RSS는 이제 변수가 **w0, w1**인 식으로 표현 할 수 있으며 RSS를 최소로 하는 **w0,w1** 즉 회귀 계수를 학습을 통해서 찾는 것이 머신러닝 기반 회귀의 핵심 사항이다.
- RSS는 **회귀식의 독립변수 X, 종속변수 Y가 중심 변수가 아니라 w 변수(회귀계수)가 중심 변수임을 인지하는 것이 매우 중요**(학습 데이터로 입력되는 **독립변수와 종속변수**는 RSS에서 **모두 상수로 간주**한다.)
- 일반적으로 RSS는 학습 데이터의 건수로 나누어서 다음과 같이 정규화된 식으로 표현

$$RSS(w_0, w_1) = \frac{1}{N}\sum_{i=1}^{N}(y_i - (w_0 + w_1 * x_i))^2$$

(*i*는 1부터 학습 데이터의 총 건수 *N*까지)

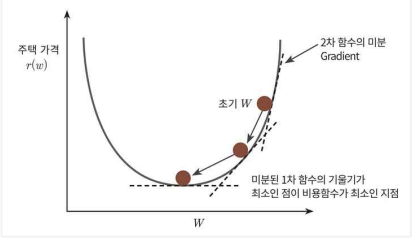
- 회귀에서 이 RSS는 비용(Cost)이며, w 변수(회귀 계수)로 구성되는 RSS를 **비용 함수(Cost function)** 라고 합니다.
- 머신러닝 회귀 알고리즘은 계속해서 학습하면서 이 비용 함수가 반환하는 오류 값을 지속해서 감소시키고 최종적으로는 더 이상 감소하지 않는 최소의 오류 값을 구하는 것입니다.

  (*비용 함수를* ***손실 함수(loss function)****라고도 함.)*

경사 하강법 (Gradient Descent)-비용(로스) 최소화

- W 파라미터의 개수가 적다면 고차원 방정식으로 비용 함수가 최소가 되는 W 변수 값을 도출할 수 있겠지만 W 파라미터가 많으면 고차원 방정식을 동원하더라도 해결하기가 어렵다.
- 경사 하강법은 이러한 고차원 방정식에 대한 문제를 해결해 주며서 비용 함수 RSS를 최소화하는 방법을 직관적으로 제공하는 뛰어난 방식이다.

많은 W 파라미터가 있는 경우에 경사 하강법은 보다 간단하고 직관적인 비용함수 최소화 솔루션 제공



경사 하강법의 사전적 의미: "점진적인 하강"이라는 뜻에서도 알 수 있듯

"점진적으로" 반복적인 계산을 통해 W 파라미터 값을 업데이트하면서 오류 값이 최소가 되는 W 파라미터를 구하는 방식

경사 하강법 (Gradient Descent)-비용(로스) 최소화

- 경사 하강법은 반복적으로 비용 함수의 반환 값, 즉 예측값과 실제값의 차이가 작아지는 **방향성**을 가지고 W 파라미터를 지속해서 보정해 나간다.
- 최초 오류 값이 100이었다면 두 번째 오류 값은 100보다 작은 90, 세 번째는 80과 같은 방식으로 지속해서 오류를 감소시키는 방향으로 W 값을 계속 업데이트해 나간다.
- 그리고 오류 값이 더 이상 작아지지 않으면 그 오류 값을 최소 비용으로 판단하고 그때의 W 값을 **최적 파라미터로 반환**한다.

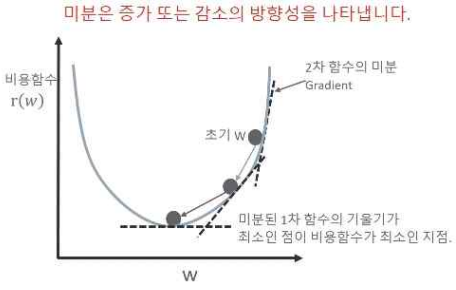
경사 하강법의 핵심: "어떻게 하면 오류가 작아지는 방향으로 W 값을 보정할 수 있을까?"

미분을 통한 로스 함수의 최소값 찾기

경사 하강법의 핵심: "어떻게 하면 오류가 작아지는 방향으로 W 값을 보정할 수 있을까?"

- 비용 함수가 그림과 같은 포물선 형태의 2차 함수라면
  경사 하강법은 최초 W에서부터 미분을 적용한 뒤 이 미분
  값이 계속 감속하는 방향으로 순차적으로 W를 업데이트 한다.

- 더 이상 미분된 1차 함수의 기울기가 감소하지 않는 지점을
  비용함수가 최소인 지점으로 간주하고 W를 반환 한다.



편미분(Partial Derivative): https://tinyurl.com/y2dqckps

RSS의 편미분

- R(w)는 변수가 w 파라미터로 이루어진 함수: $R(w) = \frac{1}{N}\sum_{i=1}^{N}(y_i - (w_0 + w_1 * x_i))^2$
  잔차(모집단이 아닌 표본 집단의 오차) 제곱 합
- 각각 w0,w1을 기준으로 한 편미분 값을 구할 수 있다.
- 각 편미분 결과가 0이 되는 지점, (n은 데이터의 개수),이 바로 RSS(w0), RSS(w1), 손실함수의 최적값이다.
  그렇기 때문에 경사 하강법을 이용한 선형 회귀란, 최적의 w0과 w1를 찾는 과정을 학습이라 할 수 있겠다

  - w0, w1로 편미분

$$\frac{\partial R(w)}{\partial w_1} = \frac{2}{N}\sum_{i=1}^{N} -x_t * (y_i - (w_0 + w_1 x_i) = -\frac{2}{N}\sum_{i=1}^{N} x_i * (\text{실제값}_i - \text{예측값}_i)$$

$$\frac{\partial R(w)}{\partial w_0} = \frac{2}{N}\sum_{i=1}^{N} -(y_i - (w_0 + w_1 x_i) = -\frac{2}{N}\sum_{i=1}^{N} (\text{실제값}_i - \text{예측값}_i)$$

특성이 여러개 일 경우

▶ **상수항 결합:** 회귀분석모형 수식을 간단하게 만들기 위해 다음과 같이 상수항을 독립변수 데이터에 추가하는 것

$$\hat{Y} \qquad X_{mat}$$

$$\begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_n \end{bmatrix} = \begin{bmatrix} \overset{\text{Feature 1}}{x_{11}} & \overset{\text{Feature 2}}{x_{12}} & \cdots & \overset{\text{Feature M}}{x_{1m}} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ x_{n1} & x_{n2} & \cdots & x_{nm} \end{bmatrix} * \begin{bmatrix} w_1 & w_2 & \cdots & w_m \end{bmatrix}^T + w_0$$

내적

$$f(x) = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D = \begin{bmatrix} 1 & x_1 & x_2 & \cdots & x_D \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_D \end{bmatrix} = x_a^T w_a = w_a^T x_a$$

경사 하강법 수행 프로세스

- Step 1: w_1, w_0를 임의의 값으로 설정하고 첫 비용 함수의 값을 계산합니다.
- Step 2: w_1을 $w_1 - \eta \frac{2}{N} \sum_{i=1}^{N} x_i * (\text{실제값}_i - \text{예측값}_i)$, w_0을 $w_0 - \eta \frac{2}{N} \sum_{i=1}^{N} (\text{실제값}_i - \text{예측값}_i)$으로 업데이트한 후 다시 비용 함수의 값을 계산합니다.
- Step 3: 비용 함수의 값이 감소했으면 다시 Step 2를 반복합니다. 더 이상 비용 함수의 값이 감소하지 않으면 그때의 w_1, w_0를 구하고 반복을 중지합니다.

주피터에서 실습해보세요.

사이킷런 **LinearRegression** 클래스

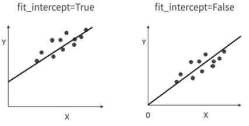
▶ LinearRegression 클래스

예측값과 실제 값의 RSS(Residual Sum of Squares)를 최소화해 OLS(Ordinary Least Squares) 추정 방식으로 구현
- fit() 메서드로 X, y 배열을 입력 받으면 회귀 계수 W를 coef_ 속성에 저장

```
class sklearn.linear_model.LinearRegression(fit_intercept=True, normalize=False, copy_X=True, n_jobs=1)
```

1. fit_intercept
   Default=True, y절편 값을 계산할지 말지 결정
2. normalize
   Default=False, fit_intercept = False일 때는 무시,
   True일 때는 데이터 세트 정규화 결정(정렬)
3. 속성값(coef_)
   fit()을 수행했을 때 회귀 계수가 배열형태로 저장하는 속성, (Target 값 개수, 피처 개수)
4. 속성값(intercept_) → 절편값

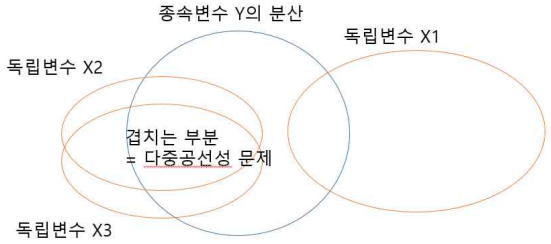
   # fit() : 학습하다.

입력 파라미터	**fit_intercept**: 불린 값으로, 디폴트는 True입니다. Intercept(절편) 값을 계산할 것인지 말지를 지정합니다. 만일 False로 지정하면 intercept가 사용되지 않고 0으로 지정됩니다.  **normalize**: 불린 값으로 디폴트는 False입니다. fit_intercept가 False인 경우에는 이 파라미터가 무시됩니다. 만일 True이면 회귀를 수행하기 전에 입력 데이터 세트를 정규화합니다.
속성	**coef_**: fit() 메서드를 수행했을 때 회귀 계수가 배열 형태로 저장하는 속성. Shape는 (Target 값 개수, 피처 개수). **intercept_**: intercept 값

선형 회귀의 다중 공선성 문제

▶ 다중공선성(Multicollinearity)

- 상관관계가 매우 높은 독립변수들이 동시에 모델에 포함될 때 발생
- 만약 두 변수가 완벽하게 다중공선성에 걸려있으면, 같은 변수를 두 번 넣은 것이며 최소제곱법을 계산이 어렵다.
- 완벽한 다중공선성이 아니더라도 다중공선성이 높다면 **회귀계수의 표준오차가 비정상적으로 커지게**된다.
- 일반적으로 상관관계가 높은 피처가 많은 경우 독립적인 주요한 피처만 남기고 제거하거나 규제를 적용한다.



회귀 평가 지표

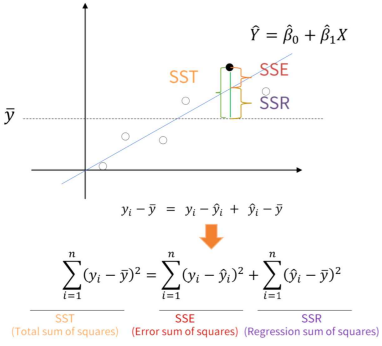
평가 지표	설명	수식
MAE	Mean Absolute Error(MAE)이며 실제 값과 예측값의 차이를 절댓값으로 변환해 평균한 것입니다.	$MAE = \frac{1}{n}\sum_{i=1}^{n}\lvert Yi - \hat{Y}i\rvert$
MSE	Mean Squared Error(MSE)이며 실제 값과 예측값의 차이를 제곱해 평균한 것입니다.	$MSE = \frac{1}{n}\sum_{i=1}^{n}(Yi - \hat{Y}i)^2$
RMSE	MSE 값은 오류의 제곱을 구하므로 실제 오류 평균보다 더 커지는 특성이 있으므로 MSE에 루트를 씌운 것이 RMSE(Root Mean Squared Error)입니다.	$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Yi - \hat{Y}i)^2}$
R^2	분산 기반으로 예측 성능을 평가합니다. 실제 값의 분산 대비 예측값의 분산 비율을 지표로 하며, 1에 가까울수록 예측 정확도가 높습니다.	$R^2 = \frac{\text{예측값 } Variance}{\text{실제값 } Variance}$

선형 회귀의 정확도 평가

- 선형회귀는 잔차의 제곱합(SSE : Error sum of squares)를 최소화 하는 방법으로 회귀 계수를 추정
- 즉, SSE가 작으면 작을수록 좋은 모델이라고 볼 수 있음
- MSE(Mean Squared Error)는 SSE를 표준화한 개념

$$SSE = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

$$MSE = \frac{1}{n-2}SSE$$



선형 회귀의 정확도 평가

$$\underbrace{\sum_{i=1}^{n}(y_i-\bar{y})^2}_{\text{SST}}=\underbrace{\sum_{i=1}^{n}(y_i-\hat{y}_i)^2}_{\text{SSE}}+\underbrace{\sum_{i=1}^{n}(\hat{y}_i-\bar{y})^2}_{\text{SSR}}$$

- Y의 총 변동은 회귀직선으로 설명 불가능 한 변동과 회귀직선으로 설명 가능한 변동으로 이루어져 있음
- R^2는 RSE의 단점을 보완한 평가지표로 0~1의 범위를 가짐
- R^2은 설명력으로 입력 변수인 X로 설명할 수 있는 Y의 변동을 의미
- R^2이 1에 가까울 수록 선형회귀 모형의 설명력이 높다는 것을 뜻함

입력 변수로 설명할 수 없는 변동 비율

$$R^2=\frac{SST-SSR}{SST}=1-\frac{SSE}{SST}=\frac{SSR}{SST} \qquad where\ SST=\sum_{i=1}^{n}(y_i-\bar{y})^2$$

사이킷런 회귀 평가 **API**

- 사이킷런은 아쉽게도 RMSE를 제공하지 않는다. RMSE를 구하기 위해서는 **MSE에 제곱근을 씌워서 계산하는 함수**를 직접 만들어야 한다.
- 다음은 각 평가 방법에 대한 사이킷런의 API 및 cross_val_score나 GridSearchCV에서 평가 시 사용되는 scoring 파라미터의 적용 값이다

평가 방법	사이킷런 평가 지표 API	Scoring 함수 적용 값
MAE	metrics.mean_absolute_error	'neg_mean_absolute_error'
MSE	metrics.mean_squared_error	'neg_mean_squared_error'
R^2	metrics.r2_score	'r2'

Scoring(cross_val_score, GridSearchCV) 함수에 회귀 평가 적용시 유의 사항

negative를 쓰는 이유.

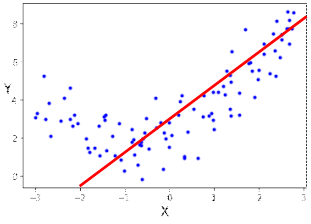
classification은 정확도, f1 score 등 클수록 좋게끔 셋팅 되어 있다. 즉 높을수록 좋은 것으로 이미 세팅

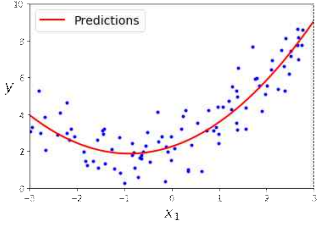
하지만 회귀에서는 loss값이 작아지는게 좋은 거라서 -1곱해서 사용. 특히 RMSE 직접 만들어 쓸 때 주의.

다항회귀(Polynomial Regression) 개요

- **항이 여러 개인 가설 함수로 결과를 예측하는 회귀 분석 방법**이다.
- 일반적인 단순 선형 회귀는 y = w0 + w1*x1 + w2*x2 + w3*x3^2, ... + wn * xn 과 같이 독립변수(feature)와 종속변수(target)의 관계가 **일차방정식(단항식)이 아닌, 2차, 3차 등 다항식으로 표현되는 회귀**

데이터 세트에 대하여 픽처 X에 대해 Target Y 값의 관계를 단순 선형회귀 직선으로 표현한 것보다 다항 회귀 곡선형으로 표현한 것이 더 예측 성능이 높다.





선형회귀와 비선형 회귀의 구분

선형회귀

Y = w0 + w1*x1 + w2*x2 + w3*x1*x2 + w4*x1^2 +w5*x2^2

새로운 변수인 Z를 z =[x1, x2, x1*x2, x1^2, x2^2]로 한다면

- 다항 회귀는 선형 회귀임(비선형 회귀가 아님)
- 회귀에서 선형/비선형을 나누는 기준은 회귀 계수가 선형/비선형인지에 따른 것으로, 독립변수의 선형/비선형 여부와는 무관함

비선형 회귀

Y = w1*cos(X+w4)+w2*cos(2*X+w4)+w3

사이키런에서의 다항 회귀

- sklearn은 다항 회귀를 위한 클래스를 제공하지 않음
- 대신 **PolynomialFeatures 클래스** 활용해 **원본 다항 피처**들을 Polynomial(다항식) 피처로 변환한 데이터 세트에 **LinearRegression 객체를 적용하여 다항회귀 기능을 제공한다.**

PolynomialFeatures 원본 피처 데이터세트를 기반으로 degree 차수에 따른 다항식을 적용하여 새로운 피처들을 생성하는 클래스(피처 엔지니어링의 기법중의 하나임)

단항 피처 $[x_1,\ x_2]$ 를 Degree = 2, 즉 2차 다항 피처로 변환한다면?
$(x_1 + x_2)^2$ 의 식 전개에 대응되는 $[1,\ x_1,\ x_2,\ x_1x_2,\ x_1^2,\ x_2^2]$의
다항 피처들로 변환

1차 단항 피처들의 값이 $[x_1,\ x_2]$ = [0 1] 일 경우
2차 다항 피처들의 값은 $[1, x_1 = 0, x_2 = 1, x_1x_2 = 0,\ x_1^2 = 0, x_2^2 = 1]$
형태인 [1, 0, 1, 0, 0, 1] 로 변환

단항 피처 $[x_1,\ x_2]$ 를 Degree = 3, 즉 3차 다항 피처로 변환한다면?
$(x_1 + x_2)^3$ 의 식 전개에 대응되는
$[1,\ x_1,\ x_2,\ x_1x_2,\ x_1^2,\ x_2^2,\ x_1^3,\ x_1^2x_2,\ x_1x_2^2,\ x_2^3]$의 다항 피처들로 변환

사이키런에서의 다항 회귀

사이킷런은 다항회귀를 바로 API로 제공하지 않습니다.

대신 PolynomialFeatures 클래스로 원본 단항 피처들을 다항 피처들로 변환한 데이터 세트에 LinearRegression 객체를 적용하여 다항회귀 기능을 제공합니다.

단항 피처 $[x_1, x_2]$를 2차 다항 피처 $[1, x_1, x_2, x_1x_2, x_1^2, x_2^2]$ 로 변경
(degree=2로 가정)

PolynomalFeatures 로 변환된 X 피처들을 LinearRegression 객체로 학습

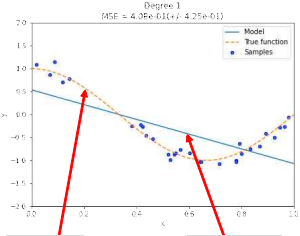
사이킷런에서는 일반적으로 Pipeline 클래스를 이용하여
PolynomialFeatuers 변환과 LinearRegression 학습/예측을 결합하여 다항 회귀를 구현합니다.

▶ 머신러닝 파이프라인(ML Pipeline)

데이터 수집부터 전처리, 학습 모델 배포, 예측까지 전과정을 순차적으로 처리하도록 설계된 머신러닝 아키텍처

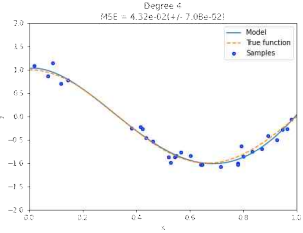
다항 회귀를 이용한 과소적합 및 과대적합 이해

실선 = 회귀, 점선 = 실제 데이터 세트의 곡선, scatter = 데이터 샘플

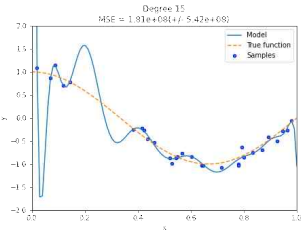


Cosine 곡선

1차직선 (회귀선)



Cosine 곡선과 동일함

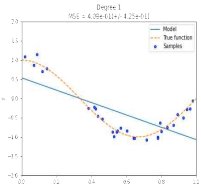


학습 데이터를 대부분 예측함

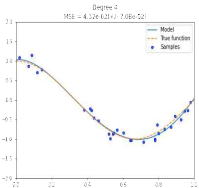
원본

다항 회귀를 이용한 과소적합 및 과대적합 이해

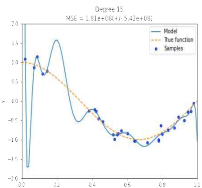
과소적합



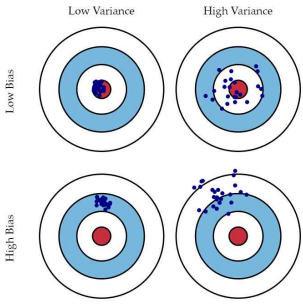
Bias



과대적합



Varianc
e



편향(Bias):
- 편향은 지나치게 단순한 모델로 인한 error입니다.
- 편향이 크면 과소 적합(under-fitting)을 야기합니다.
- 모델에 편향이 크다는 것은 그 모델이 뭔가 중요한 요소를 놓치고 있다는 뜻입니다.

분산(Variance)
- 분산은 지나치게 복잡한 모델로 인한 error입니다.
- 훈련 데이터에 지나치게 적합시키려는 모델입니다.
- 분산이 크면 과대 적합(Over-fitting)을 야기합니다.
- 분산이 큰 모델은 훈련 데이터에 지나치게 적합을 시켜 일반화가 되지 않은 모델입니다.

규제(Regularization) 선형 회귀 개요

- 이전까지의 선형 모델 비용 함수는 RSS를 최소화하는(실제값과 예측값의 차이를 최소화하는) 것만 고려
- Degree가 1인 경우 예측곡선이 단순하여 과소적합을 하게 되고, Degree가 15인 경우 학습 데이터 샘플에 지나치게 적합한 곡선을 그려 과적합을 유발한다.
- **규제(Regularization)**는 모델이 과대 적합을 피하여 일반화 성능을 잃지 않도록 가중치를 제한하는 방법으로, *과대 적합을 완화하기 위한 대표적인 방법*

$$\text{비용 함수 목표} = \text{Min}(\text{RSS(W)} + \text{alpha} * \|W\|_2^2)$$

alpha는 학습 데이터 적합 정도와 회귀 계수 값을 크기 제어를 수행하는 튜닝 파라미터

노름(Norm) 이해하기

- 라쏘와 릿지와 같은 규제를 알기전에 우선 노름이 무엇인지에 대해 설명하겠습니다.
- 노름이란 벡터의 크기를 측정하는 방법입니다.
- 벡터의 크기는 다음과 같은 방정식을 통해 나타낼 수 있습니다.

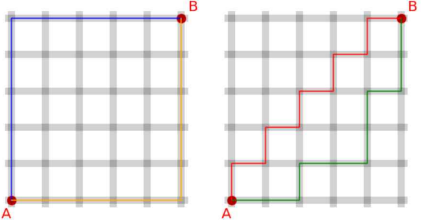
$$\|x\|_p := \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p}$$

p는 차수, n은 차원을 나타내고,
차수가 1인 노름을 **L1 노름**,
차수가 2인 노름을 **L2 노름**이라고 합니다.

- L1 노름은 **맨해튼 노름(Manhattan norm)** 또는 **택시 노름(Taxicab norm)**이라고도 불립니다.

$$\|x\|_1 := \sum_{i=1}^{n} |x_i|$$

아래 그림처럼 A에서 B까지의 거리는 맨해튼 거리에서 택시를 타고 이동한 거리와 같습니다.

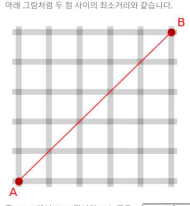


즉, A(0,0)에서 B(5,5)까지의 L1 노름은 |5-0| + |5-0| = 10 입니다.

노름(Norm) 이해하기

▶ L2 노름은 **유클리드 노름(Euclidean norm)**이라고도 불립니다.

$$\|\mathbf{x}\|_2 := \sqrt{\sum_{i=1}^{n} x_i^2}$$



정규화(Regularization)

▶ L1 규제를 통한 정규화를 라쏘(least absolute shrinkage and selection operator, LASSO),
L2 규제를 통한 정규화를 릿지(Ridge)라고 합니다.

$$cost(\theta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda(\sum_{i=1}^{n}|\theta_i|^p)^{1/p}$$

참조: https://www.robotstory.co.kr/raspberry/?mode=view&board_pid=64

규제(Regularization) 선형 회귀 개요

$$\text{비용 함수 목표} = \text{Min}(\text{RSS}(\text{W}) + \text{alpha} * \| W \|_2^2)$$

- 위 수식의 최소값을 만족하는 W 벡터를 찾기 위해 alpha 값이 어떤 역할을 하는지 확인보면
- **alpha 값이 0 또는 0에 수렴**하는 매우 작은 값이라면 위 수식은 다음과 같이 변합니다.

$$Min(RSS(W) + 0)$$

- 반면 **alpha 값이 무한대에 가까운** 매우 큰 값이라면 RSS(W)에 비해 아래의 값이 비정상적으로 커지게 됩니다.

$$alpha * ||W||_2^2$$

- 따라서 **alpha 값을 크게 가져간다면 회귀 계수 W를 작게 가져가는 것으로 과적합을 개선**할 수 있으며 **alpha 값을 작게 가져가는 경우 회귀 계수 W의 값이 커져도 어느 정도 상쇄**가 가능하여 데이터 적합을 개선할 수 있게 됩니다.

규제(Regularization) 선형 회귀 개요

- 이처럼 alhpa 값을 0에서부터 지속적으로 증가시켜 회귀 계수 값의 크기를 감소시키는 것이 가능하며 비용 함수에 **alhpa 값으로 패널티를 부여해 회귀 계수 값의 크기를 감소**시켜 과적합을 개선하는 방법을 **규제(Regularization)** 이라고 합니다.

규제에는 L1 규제와 L2 규제로 나누어짐

▶ L1 규제는 아래의 식과 같이 W의 절댓값에 대해 패널티를 부여하는 규제 방법으로 영향력이 크지 않은 회귀 계수 값을 0으로 변환하며 **L1 규제를 적용한 회귀를 라쏘(Lasso) 회귀라 합니다.**

$$alpha * ||W||_1$$

▶ L2 규제는 아래의 식과 같이 W의 제곱에 대해 패널티를 부여하는 방법을 사용하며 **L2 규제를 적용한 회귀를 릿지(Rldge) 회귀**라 부릅니다.

$$alpha * ||W||_2^2$$

릿지 회귀 (Ridge Regression)

- 릿지 회귀는 L2 규제를 이용한 회귀로 Ridge 클래스의 주요 생성 파라미터는 alpha입니다.
- 따라서 **릿지 회귀는 alpha 값이 커질수록 회귀 계수 값을 작게 만든다는 특징**이 있습니다.
- 사이킷런은 Ridge 클래스를 통해 릿지 회귀를 구현합니다.

주피터에서 보스턴 주택 가격을 Ridge 클래스로 예측하고,
cross_val_score()로 평가하기

라쏘 회귀 (Lasso Regression)

- L1규제는 불필요한 회귀 계수를 급격하게 감소시켜 0으로 만들어 제거
- L1규제는 적절한 피처만 회귀에 포함시키는 피처 선택의 특성을 가지고 있다.
- L2 규제와 다르게 불필요한 **회귀 계수는 0으로 만들고 제거**, 즉 적절한 피처만 선택
- L1 규제는 alpha∗||W||1 를 의미
- 라쏘 회귀 비용함수 목표는 RSS(W) + a l p h a ∗||W||1 식을 최소화하는 W를 찾는 것
- sklearn의 Lasso 클래스 제공

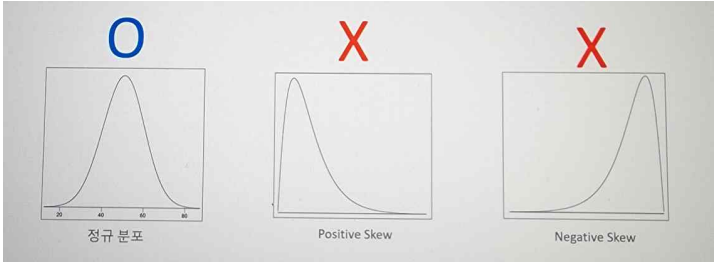
엘라스틱넷(ElasticNet)

- 엘라스틱넷(Elastic Net) 회귀: L2 규제와 L1 규제를 결합한 회귀
- L1규제의 상관관계가 높은 피처를 제외한 나머지는 0으로 바꾸고, 회귀 계수의 급변을 방지하기 위해 L2규제를 적용
- 유일한 단점은 수행시간이 길다는 것
- 엘라스틱넷 회귀 비용함수 목표: RSS(W)+alpha2∗||W||2 2 + alpha1 ∗||W||1 식을 최소화하는 W를 찾는 것
- sklearn의 ElasticNet(주요 파라미터: alpha = a + b, l1_ratio = a / (a + b))
- ElasticNet의 규제는 a*L1 + b*L2로 정의(a: L1의 alpha, b: L2의 alpha)

선형 회귀 모델을 위한 데이터 변환

선형 회귀 모델과 같은 선형 모델은 일반적으로 피처와 타겟 간에 선형의 관계가 있다 가정하고, 이러한 최적의 선형함수를 찾아내 결과를 예측합니다.

또한 선형 회귀 모델은 **피처값과 타겟값의 분포가 정규 분포(즉 평균을 중심으로 종 모양으로 데이터 값이 분포된 형태) 형태를 매우 선호**하며 타겟값의 경우 정규 분포 형태가 아니라 특정값의 **분포가 치우친 왜곡된 형태의 분포도**일 경우 예측 성능에 **부정적인 영향을 미칠 가능성이 높으며** 피처값 역시 결정값보다는 덜하지만 **왜곡된 분포도로 인해 예측 성능에 부정적인 영향을** 미칠 수 있습니다.



선형 회귀 모델을 위한 데이터 변환

1. StandardScaler 클래스를 이용해 평균이 0, 분산이 1인 표준 정규 분포를 가진 데이터 셋으로 변환하거나 MinMaxScaler 클래스를 이용해 최소값이 0이고 최대값이 1인 값으로 정규화를 수행.

2. 스케일링/정규화를 수행한 데이터 셋에 다시 다항 특성을 적용하여 변환하는 방법.
   보통 1번 방법을 통해 예측 성능에 향상이 없을 경우 이와 같은 방법을 적용.

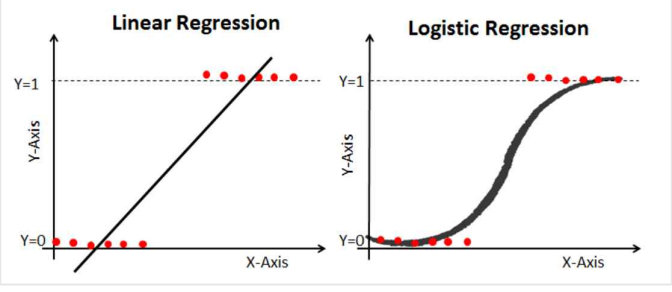
3. 원래 값에 log 함수를 적용하면 보다 정규 분포에 가까운 형태로 값이 분포. 이런 변환을 로그 변환이라함.
   로그 변환은 매우 유용한 변환이며, 실제로 선형 회귀에서는 앞서 소개한 1,2번 방법보다 로그 변환이 훨씬 많이 사용되는 변환방법.
   그 이유는 1번 방법의 경우 예측 성능 향상을 크게 기대하기 어려운 경우가 많으며, 2번 방법의 경우 피처의 개수가 매우 많을 경우에는 다항 변환으로 생성되는 피처의 개수가 기하급수로 늘어나서 과적합의 이슈가 발생할 수 있기 때문.

로지스틱 회귀

- 로지스틱 회귀: 선형 회귀 방식을 분류에 적용한 알고리즘 → '분류'에 사용
- 선형 회귀 계열이나, 선형 회귀와 다른 점은 학습을 통해 선형 함수의 회귀 최적선을 찾지 않고 시그모이드(Sigmoid) 함수 최적선을 찾고 시그모이드 함수 반환 값을 확률로 간주하여 확률에 따라 분류를 결정하는 것
- 로지스틱 회귀는 가볍고 빠르며, 이진 분류 예측 성능까지 뛰어남
  - 이진 분류의 기본 모델로 사용하는 경우가 많음
  - 예) 합격/불합격, 높음/낮음, 정답/오답 등
- 로지스틱 회귀는 희소한 데이터 세트 분류에서도 뛰어난 성능을 보임
  - 텍스트 분류에서도 자주 사용

로지스틱 회귀

- 회귀에서 분류 문제에 적용하기
- 종양의 크기에 따라 악성 종양인지(Yes = 1), 아닌지(No = 0)를 회귀를 이용하여 1과 0 값으로 예측
- 종양 크기에 따라 악성될 확률이 높다고 하면 아래 왼쪽 그림과 같이 분포하며 선형 회귀 선을 그릴 수 있으나, 해당 회귀 라인은 0과 1을 제대로 분류하지 못
- 아래의 오른쪽 그림처럼 시그모이드 함수를 이용하면 조금 더 정확하게 0과 1을 분류할 수 있음



로지스틱 회귀

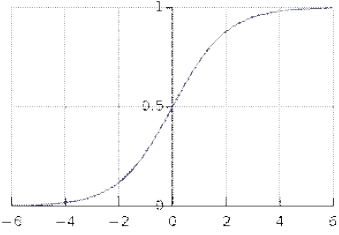
1) 시그모이드 함수 (Sigmoid Function)

- 선형회귀만으로 해결이 안되던 문제, 어떤 값을 넣든 결과가 0~1 을 출력해야 하는 문제

  > 기존의 가설의 결과를 0~1 범위로 나타낼 수 있는 시그모이드 함수 발견

$$S(x) = \frac{1}{1+e^{-x}} = \frac{e^{x}}{e^{x}+1}$$

<= 시그모이드 함수 (로지스틱 함수의 일종)



오즈비에 로그를 취한 로짓 그래프 : S 모양

- 그래프처럼 0과 1 사이에 집약되어 있고 그 모양이 S 형태를 띄기 때문에 Sigmoid라는 이름이 붙는다.
- 합격/불합격을 분류한 데이터를 학습 > 시그모이드 함수 > x값을 넣으면 y 값은 0~1 사이의 확률로 예측됨

Sigmoid 함수의 손실함수

2) 오즈비 vs 로짓변환

오즈비(Odds ratio) : 0(실패)에 대한 1(성공)의 비율 (0 : no, 1 : yes)

- no인 상태와 비교하여 yes가 얼마나 높은지 낮은지 정량화한 것
- 오즈비 = p (success) / 1-p (fail)
- p : y=1이 나올 확률, 1-p : y=1의 여사건

로짓(logit) 변환

- 오즈비에 log 함수 적용한 것
- 로짓 = log(p / 1-p)

=> 로짓을 대상으로 회귀분석을 적용한 것이 로지스틱 회귀분석(Logistic Regression Analysis)

$$C(H(x), y) = \begin{cases} -log(H(X)) & : y = 1 \\ -log(1 - H(x)) & : y = 0 \end{cases}$$

참조: https://nittaku.tistory.com/478?category=765632

분류 모델 성능 평가 지표

1. 오차 행렬(Confusion Matrix)

- 분류의 예측 범주와 실제 데이터의 분류 범주를 교차 표(Cross Table) 형태로 정리한 행렬
- 이진 분류의 예측 오류가 얼마인지와 더불어 어떠한 유형의 예측 오류가 발생하고 있는지를 함께 나타낸다.

* 표 내부의 설명를 **환자다 / 환자가 아니다 로 설명**

- **Actual(Positive)** : 실제 환자
- **Actual(Negative)** : 실제 환자가 아님
- **Predict(Positive)** : 실제 환자로 예측
- **Predict(Negative)** : 환자가 아닌 것으로 예측

		예측 값	
		Predicted Positive	Predicted Negative
실제 값	Actual Positive	True Positive(TP)	False Negative(FN)
	Actual Negative	False Positive(FP)	True Negative(TN)

- TP(True Positive) : 긍정예측을 성공 즉, 환자라고 예측해서 실제 환자임을 맞춤
- TN(True Negative) : 부정예측을 성공 즉, 비환자라고 예측하여 실제 비환자임을 맞춤
- FP(False Positive) : 긍정예측을 실패 즉, 환자라고 예측했지만 비환자임
- FN(False Negative) : 부정예측을 실패 즉, 비환자라고 예측했지만 실제 환자임

분류 모델 성능 평가 지표

2. 정확도(Accuarcy)

- 실제 데이터에서 예측 데이터가 얼마나 같은지를 판단하는 지표이다.
- *전체 데이터중에, 정확하게 예측한 데이터의 수'라고* 할 수 있다.
- 정확도는 직관적으로 모델 예측 성능을 나타내는 평가 지표
- 정확도를 분류 모델의 평가 지표로 사용할 때는 주의해야 하는데, 특히 불균형한 데이터(imbalanced data)의 경우에 정확도는 적합한 평가지표가 아니다.

  (예) 100개의 데이터가 있고 이 중에 90개의 데이터 레이블이 0, 단 10개의 데이터 레이블이 1이라고 한다면 무조건 0으로 예측 결과를 반환하는 ML모델의 경우라도 정확도가 90%가 됩니다.

정확도(Accuracy) = 예측 결과가 동일한 데이터 건수 / 전체 예측 데이터 건수

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

		Predict	
		Positive	Negative
Actual	Positive	TP	FN
	Negative	FP	TN

분류 모델 성능 평가 지표

3. 정밀도(Precision)

- *긍정으로 예측한 것 중 실제로 맞춘 비율*을 보여준다
- 열방향으로써 예측한 값들 중에서 얼마나 잘 예측했는지를 나타내는 척도이다
- 환자가 맞다고 예측하여 실제로 환자인 경우와 환자가 아닌 경우에 대비한 실제로 환자임을 맞춘 것을 나타냅니다

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

		Predict	
		Positive	Negative
Actual	Positive	TP	FN
	Negative	FP	TN

분류 모델 성능 평가 지표

4. 재현율(Recall) = 민감도(Sensitivity)

- *특이도로 실제 Negative를 얼마나 잘 예측했는지를 나타내는 지표*
- 즉, 환자가 아닌 사람을 환자가 아니다라고 얼마나 잘 예측하는지 보여주는 척도이다.
- 특이도를 통해서 False Positive Rate를 구할 수 있는데 이는 환자가 아닌데 환자라고 예측하는 경우로써 ***False Positive Rate = 1 - Specificity*** 로 나타낼 수 있다.

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

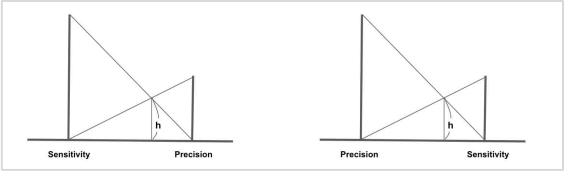
		Predict	
		Positive	Negative
Actual	Positive	TP	FN
	Negative	FP	TN

분류 모델 성능 평가 지표

5. F1 Score

- 보통 불균형 분류문제에서 평가척도로 주로 사용한다.
- 데이터가 불균형한 상태에서 Accuracy로 성능을 평가하기엔 데이터 편향성이 너무 크게 나타나 올바르게 성능을 측정하기 힘들게 된다. 그렇기 때문에 F1 Score를 평가척도로 사용한다.
- **F1 Score**는 **Sensitivity**와 **Precision**을 이용하여 **조화평균**을 구하여 평가 척도를 구성한다.

정밀도와 재현율이 어느 한쪽으로 치우치지 않는 수치를 나타낼 때 F1 Score는 높은 값을 가지게 된다.



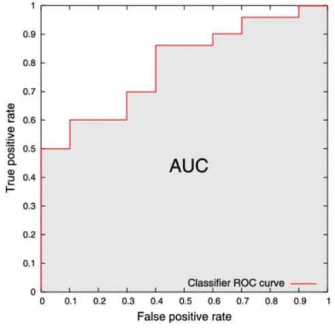
F1 Score = 2 * Precision * Recall / (Precision + Recall)

(*Sensitiviy(특이성) = Recall(재현율))

$$\Rightarrow \quad 2 * \frac{\text{Sensitivity} * \text{Precision}}{\text{Sensitivity} + \text{Precision}}$$

6. ROC-AUC 곡선

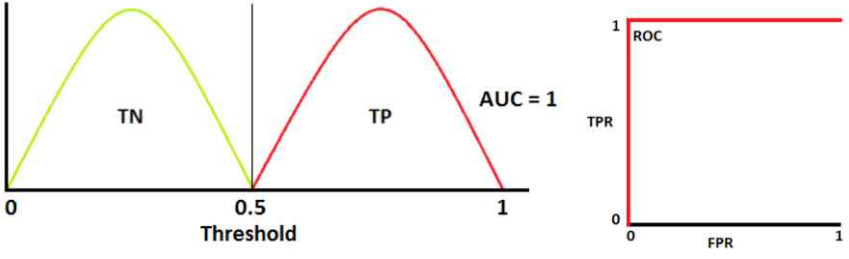
- AUC-ROC 곡선은 다양한 임계값에서 모델의 분류 성능에 대한 측정 그래프임
  * ROC(Receiver Operating Characteristic) = 모든 임계값에서 분류 모델의 성능을 보여주는 그래프
  * AUC(Area Under the Curve) = ROC 곡선 아래 영역
- AUC가 높다는 사실은 클래스를 구별하는 모델의 성능이 훌륭하다는 것을 의미
- 임상에서 AUC-ROC 곡선은 정상인 및 환자 클래스를 구분하는 모델(ex. 특정 유전자군)의 성능 평가로 흔하게 사용됨
- ROC 곡선은 TPR(=민감도)이 y축에 있고, FPR(=1-specificity)이 x축으로 그려짐



ROC-AUC 커브

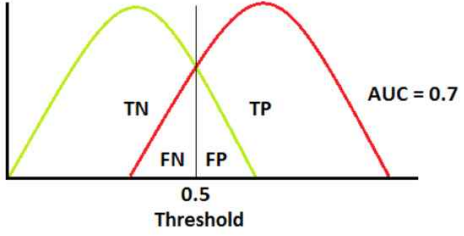
1) AUC = 1

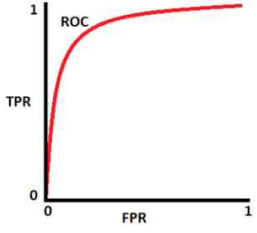
- 두 개의 곡선이 전혀 겹치지 않는 경우 모델은 이상적인 분류 성능을 보임
- 양성 클래스와 음성 클래스를 완벽하게 구별 할 수 있음



2) AUC = 0.7

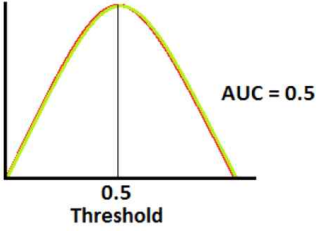
- 두 분포가 겹치면 'type 1 error' 및 'type 2 error'가 발생
- 설정한 threshold에 따라, 위에 오류값들을 최소화 또는 최대화 할 수 있음
- AUC 값이 0.7이면, 해당 분류 모델이 양성 클래스와 음성 클래스를 구별 할 수 있는 확률은 70%임을 의미

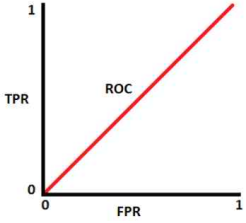




3) AUC = 5

- 분류 모델의 성능이 최악인 상황
- AUC가 0.5 정도인 경우, 해당 분류 모델은 양성 클래스와 음성 클래스를 구분할 수 있는 능력이 없음





GridSearch(그리드 서치)

- 사이킷런에서는 분류 알고리즘이나 회귀 알고리즘에 사용되는 하이퍼 파라미터를 순차적으로 입력해 학습을 하고 측정을 하면서 가장 좋은 파라미터를 알려준다.
- GridSearchCV가 없다면 max_depth 가 3일때 가장 최적의 스코어를 뽑아내는지 1일때 가장 최적인 스코어를 뽑아내는지 일일이 학습을 해야 한다.
- 하지만, grid 파라미터 안에서 집합을 만들고 적용하면 최적화된 파라미터를 뽑아낼 수 있다.

참조:

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html