파이썬 머신러닝 기초

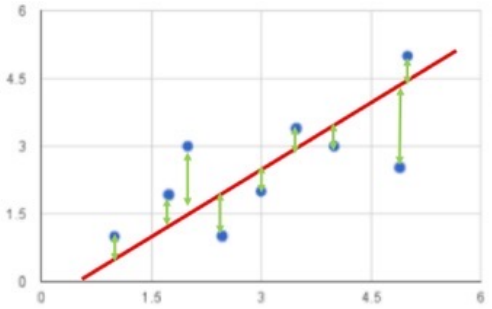
머신러닝 - 회귀2

회귀 (regression)

좋은 회귀 모델이란?

회귀 모델은 적절히 데이터에 적합하면서도 회귀 계수가 기하급수적으로 커지는 것을 제어할 수 있어야 한다

회귀 (regression)



Σ(실제 값 – 예측 값)^2

$= \Sigma(y - \hat{y})^2$

$= Error^2$

= RSS (Residual Sum of Squares)

= 오차제곱합

$\hat{Y}$

predicted values of Y

| 이전까지 하던 방식 RSS의 최소화

| 학습데이터에 너무 맞추는 Overfitting이 발생할 수 있다.

⇒ 이것을 방지하기 위해 회귀 계수가 커지지 않도록 하는 방법 = 규제

| 수식으로의 표현

비용함수 목표 = $Min(RSS(W) + alpha * ||W||_2^2)$

회귀 (regression)

비용함수 목표 = $Min(RSS(W) + alpha * ||W||_2^2)$

- $alpha$: 학습 데이터 적합 정도와 회귀 계수(W) 값의 크기를 제어를 수행하는 **하이퍼 파라미터** 입니다.

$Min(RSS(W)+alpha*||W||^2)$ | $Min(RSS(W)+alpha*||W||^2)$

알파↓ | 알파↑

기존의 RSS값이 되어버림. | W 값이 너무 커져버림

⬇ | ⬇

RSS(W)의 최소화 | **회귀 계수 값의 크기를 작게하여 과적합을 개선시킨다.**

회귀 (regression)

비용함수 목표 = $Min(RSS(W) + alpha * ||W||_2^2)$

| L1 방식(라쏘) : W의 절댓값에 대해 패널티를 부여하는 방식

| L2 방식(릿지) : W의 제곱에 대해 패널티를 부여하는 방식

OLS	Ridge	Lasso
$cost = \sum e_i^2$	$cost = \sum e_i^2 + \alpha \sum w_i^2$	$cost = \sum e_i^2 + \alpha \sum \|w_i\|$

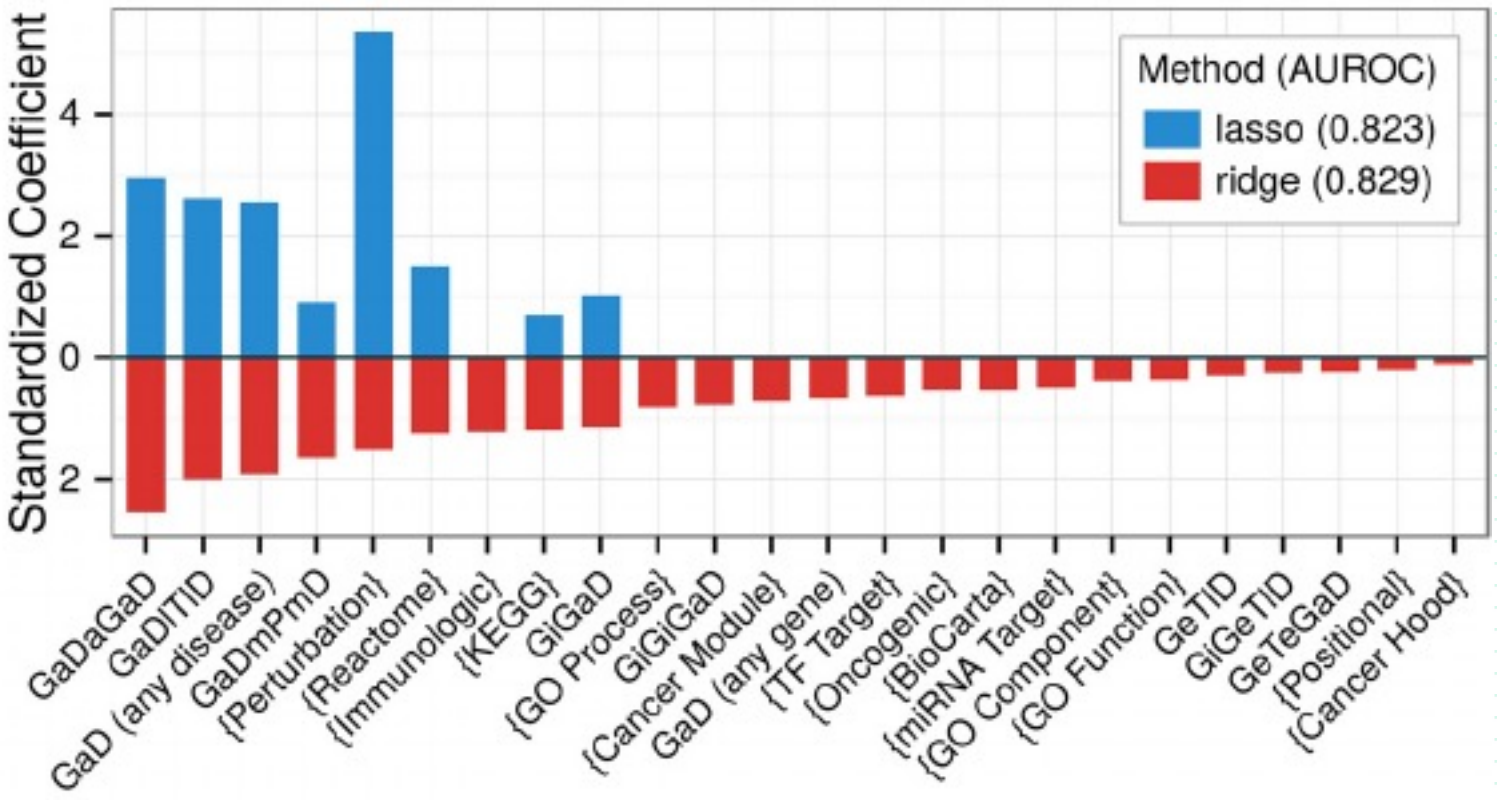
회귀 (regression)

- L2 방식(릿지)의 특성상 패널티를 크게 부여하더라도 0이 되지는 않는다

- 그러나 L1 방식(라쏘)의 경우 알파값에 따라 회귀 계수가 완전히 0이 될 수 있다.

- 라쏘 회귀는 불필요한 회귀 계수를 0으로 만드는 성향이 강한데, 이에 따라 불필요한 회귀계수를 완전히 삭제시켜 버릴 수 있다.

회귀 (regression)



회귀 (regression)

- 엘라스틱넷 회귀(Elastic Net Regression)

- L2규제와 L1규제를 결합한 회귀이다

- 배경 : 라쏘회귀가 불필요한 회귀 계수를 모두 0 으로 만드는 성향이 강한데,
이를 통해 alpha값에 따라 회귀 계수가 급격이 변동한다.

- 엘라스틱넷은 이를 완화하기 위해 L2 규제를 추가한 것이다

회귀 (regression)

- 엘라스틱넷 회귀(Elastic Net Regression)

- 하이퍼파라미터를 통해 L1규제와 L2규제의 비율을 정의해줄 수 있다

Elastic-Net Penalty:

$$R(w) := \frac{\varphi}{2} \sum_{i=1}^{n} w_i^2 + (1 - \varphi) \sum_{i=1}^{n} |w|$$

A convex combination of L1 and L2 Penalty.

선형 회귀 모델을 위한 데이터 변환

- 선형 회귀 모델은 피처값과 타깃값의 분포가 정규분포 형태일 때 가장 좋은 성능이 나타난다. 특히 타깃값에서 큰 영향을 받음

- 특히나 분포가 치우친 왜곡(skew)된 형태의 분포도일 경우 예측 성능에 부정적인 영향이 크다.

- ex) 단위가 다른 feature가 존재한다면?

선형 회귀 모델을 위한 데이터 변환

- 데이터를 스케일링 그리고 정규화를 해주자
- 보통 이러면 성능이 좋아짐(꼭 그런 거는 아니기는 하지만 대부분)
- 왜곡된 데이터가 많다면 중요 피처들을 위주로 스케일링/정규화를 수행하자

선형 회귀 모델을 위한 데이터 변환

- StandardScaler : 평균이 0, 분산이 1인 표준 정규 분포를 가진 데이터 세트로 변환

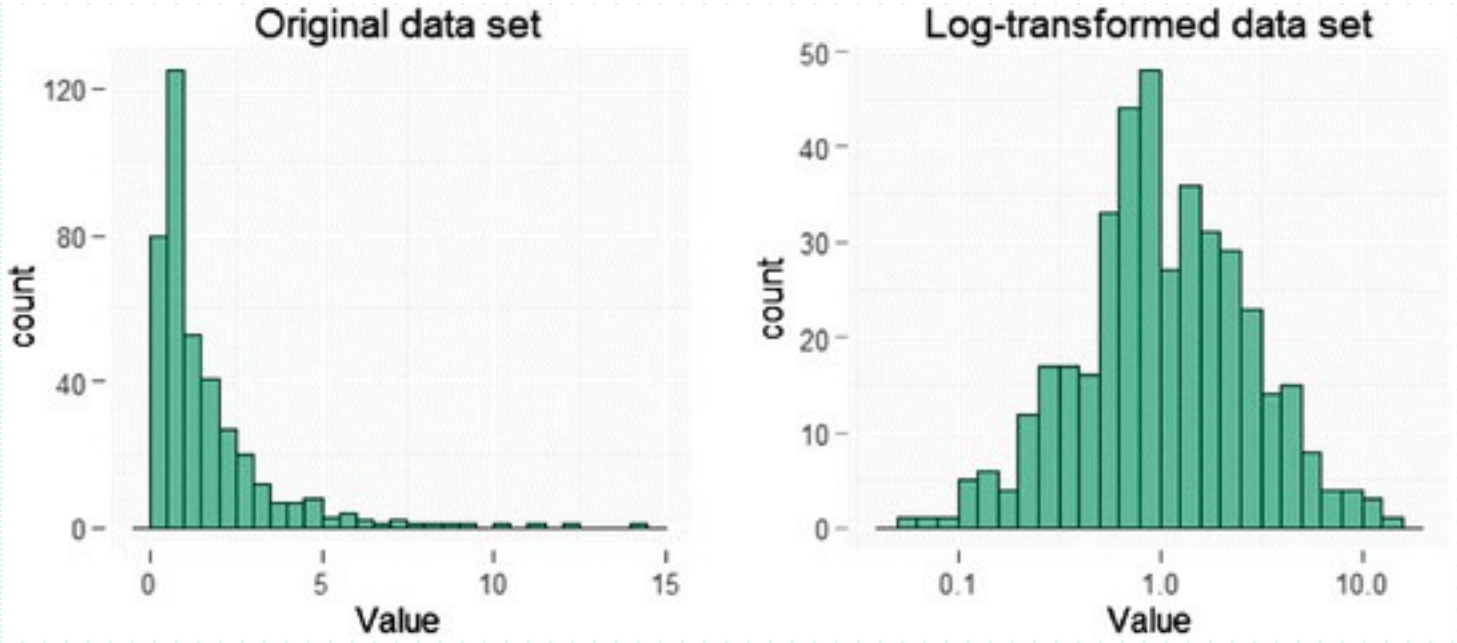
- MinMaxScaler : 최대값이 1, 최솟값이 0인 것으로 정규화

Standard Scaler	$\frac{x_i - \text{mean}(\boldsymbol{x})}{\text{stdev}(\boldsymbol{x})}$
MinMax Scaler	$\frac{x_i - \min(\boldsymbol{x})}{\max(\boldsymbol{x}) - \min(\boldsymbol{x})}$

선형 회귀 모델을 위한 데이터 변환

log 변환

가장 많이 사용되는 방법. 상당히 왜곡되어 있을 때 데이터 분포를 정규화해준다.



선형 회귀 모델을 위한 데이터 변환

- log 변환
- 저번에도 얘기한 것 처럼 np.log1p( )함수를 사용해야 한다. why?
- 언더플로우 되지 않게 조심하자

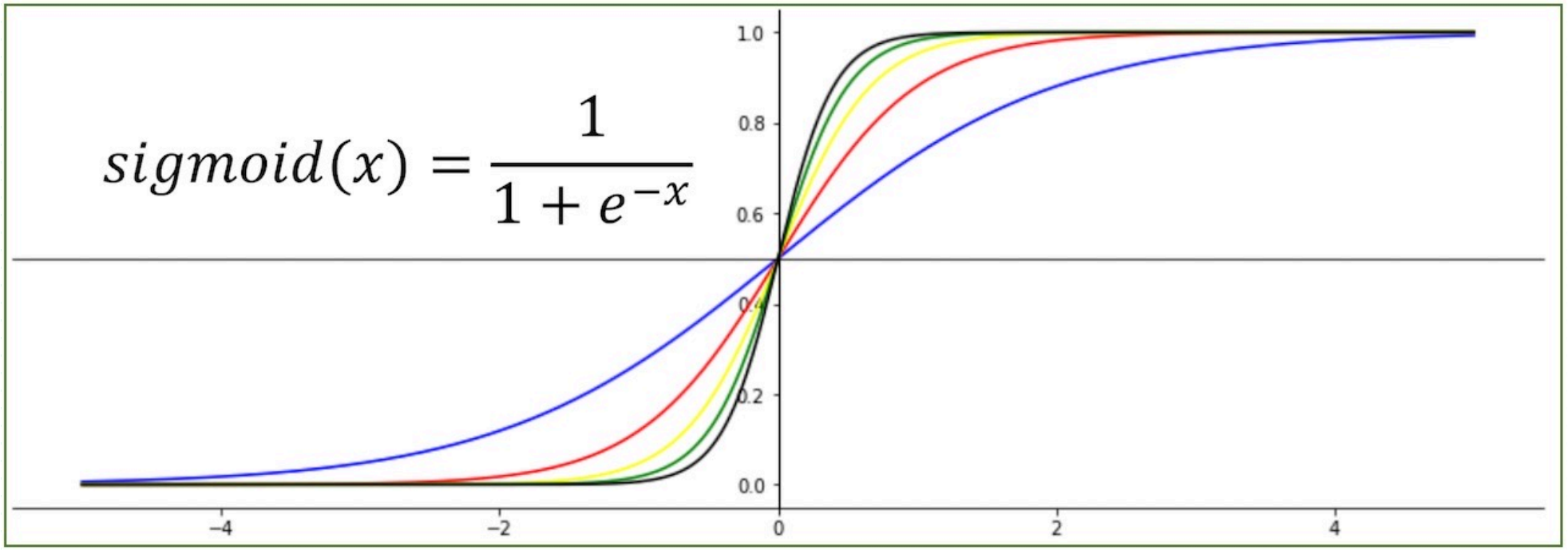
로지스틱 회귀

- 로지스틱 회귀는 선형 회귀 방식을 분류에 적용한 알고리즘.
- 하지만 선형 함수의 회귀 최적선을 찾는 것이 아니라 시그모이드 함수 최적선을 찾는다.
- 그리고 이 시그모이드 함수의 반환 값을 확률로 간주해 확률에 따라 분류를 결정

로지스틱 회귀

시그모이드 함수?

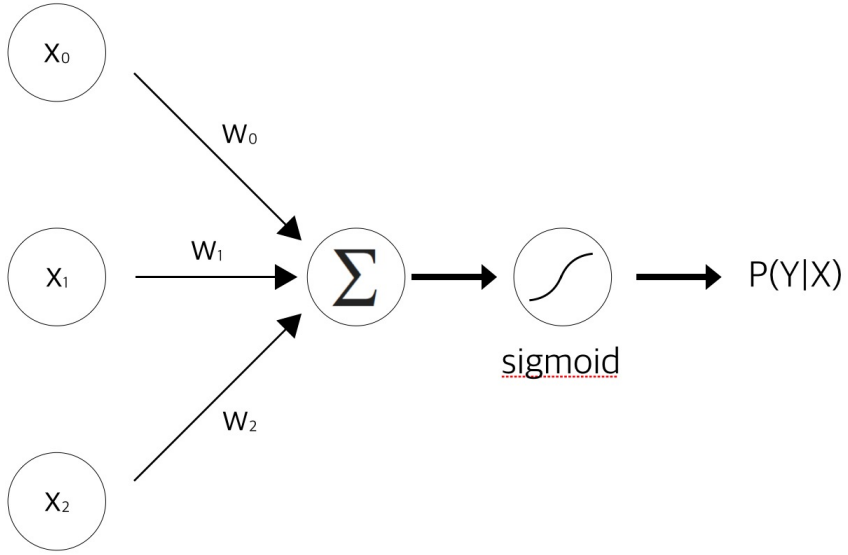
$$sigmoid(x) = \frac{1}{1 + e^{-x}}$$



로지스틱 회귀

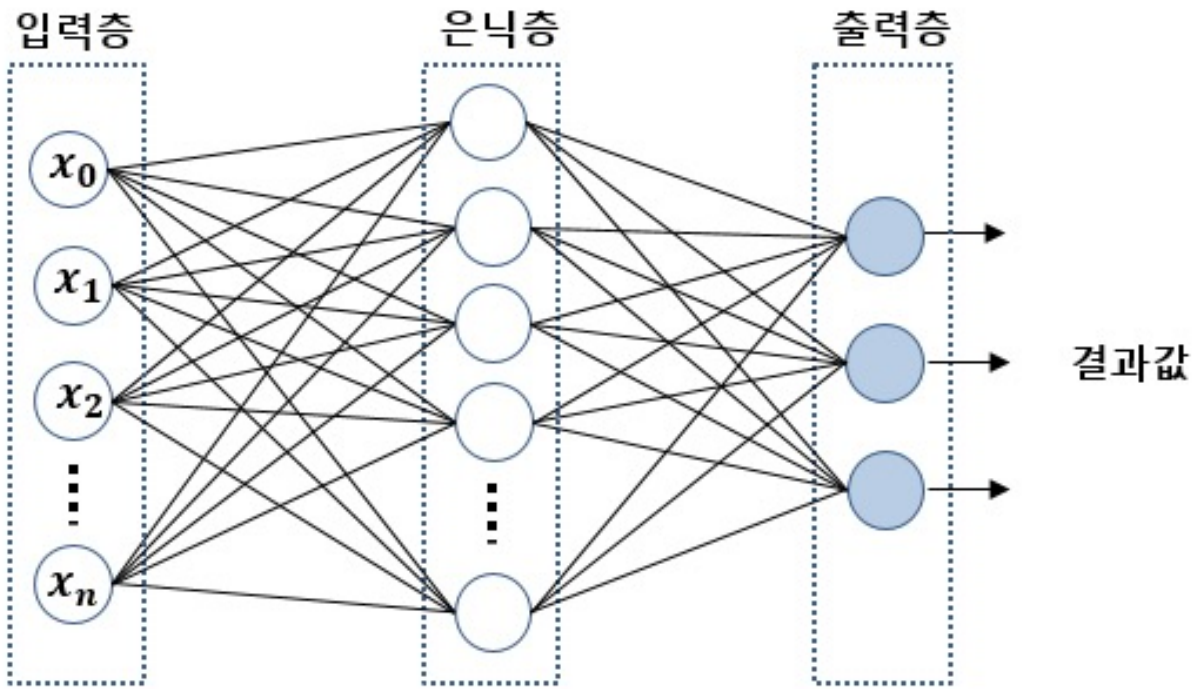
| 특히나 시그모이드 함수는 미분이 쉽기 때문에 우리가 최적화할 때 간단하게 이용할 수 있다.

| 시그모이드 함수는 그 사용도가 매우 다양 (역전파에 매우 유리)



로지스틱 회귀

역전파 맛보기



회귀 트리

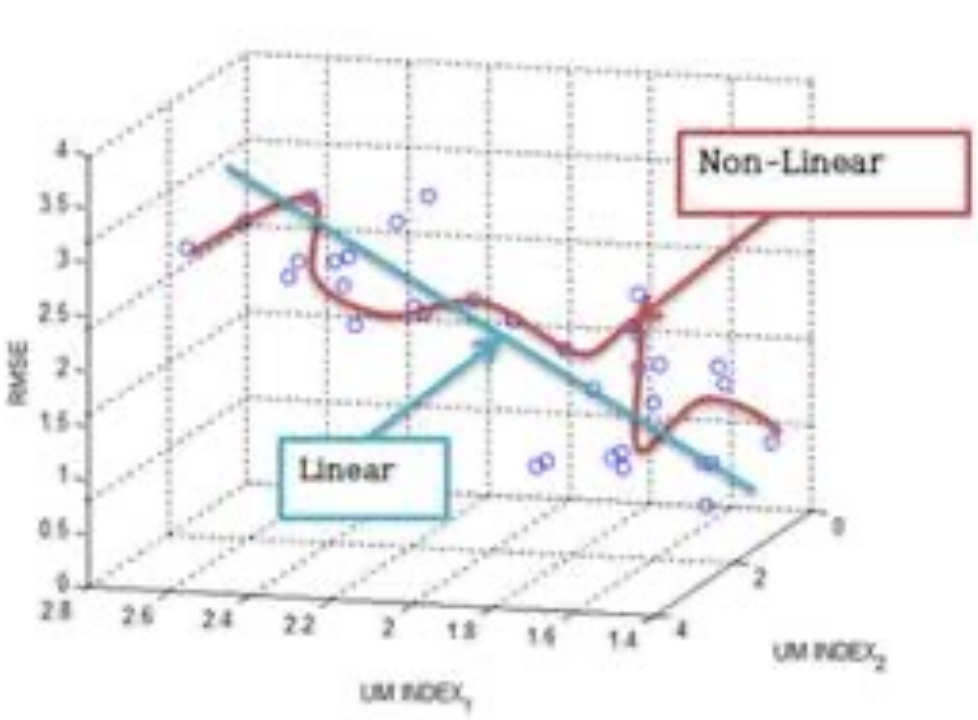
- 트리로 회귀를 만들 수는 없을까?

- 비선형 회귀의 대표적인 예 : 회귀트리

- 사실 비선형 회귀는 선형 회귀의 결합들로 만들어진 형태라고 생각하면 된다.

회귀 트리

비선형 회귀



회귀 트리

| 회귀 트리

| 트리 기반의 회귀는 회귀를 위한 트리를 생성하고 이를 기반으로 회귀 예측을 하는 것

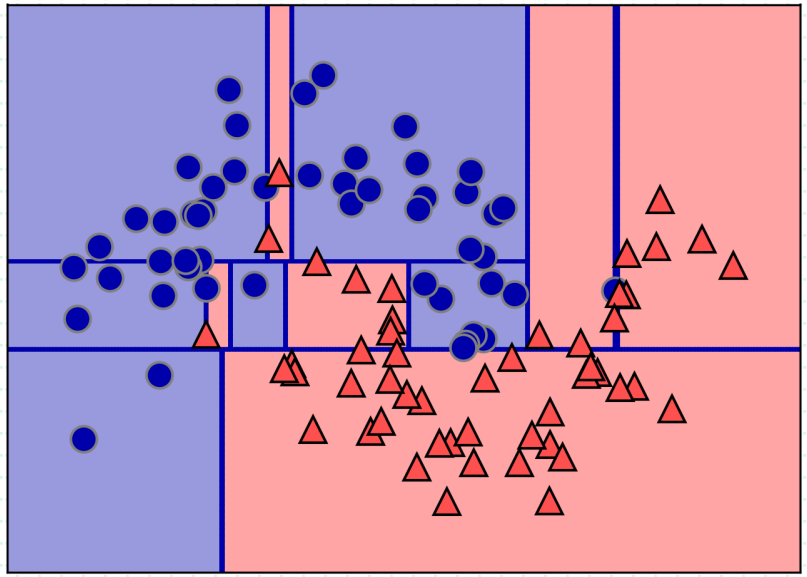
| 분류 트리와 크게 다르지 않지만 리프 노드에서 예측 결정값을 만드는 과정에서 차이가 있는데, 회귀 트리의 경우 특정 클래스의 레이블이 아닌

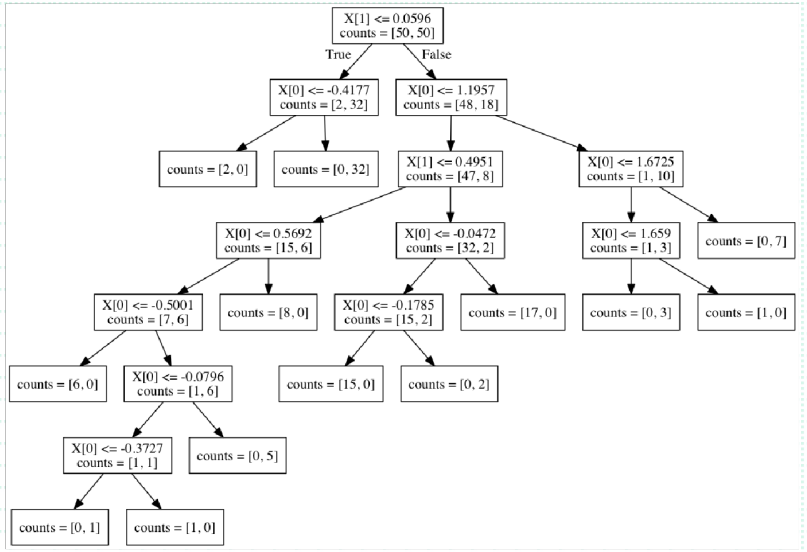
리프 노드에 속한 데이터 값의 평균값을 구해 회귀 예측값을 계산

회귀 트리

우리가 알던 트리

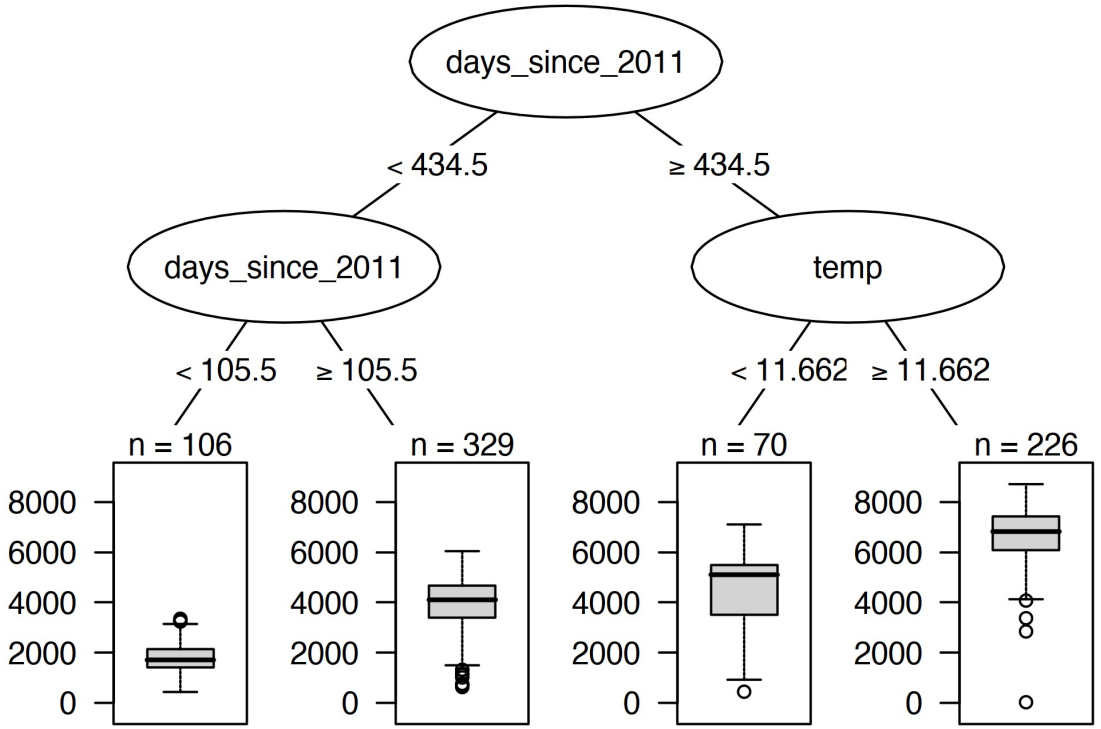
깊이 = 9





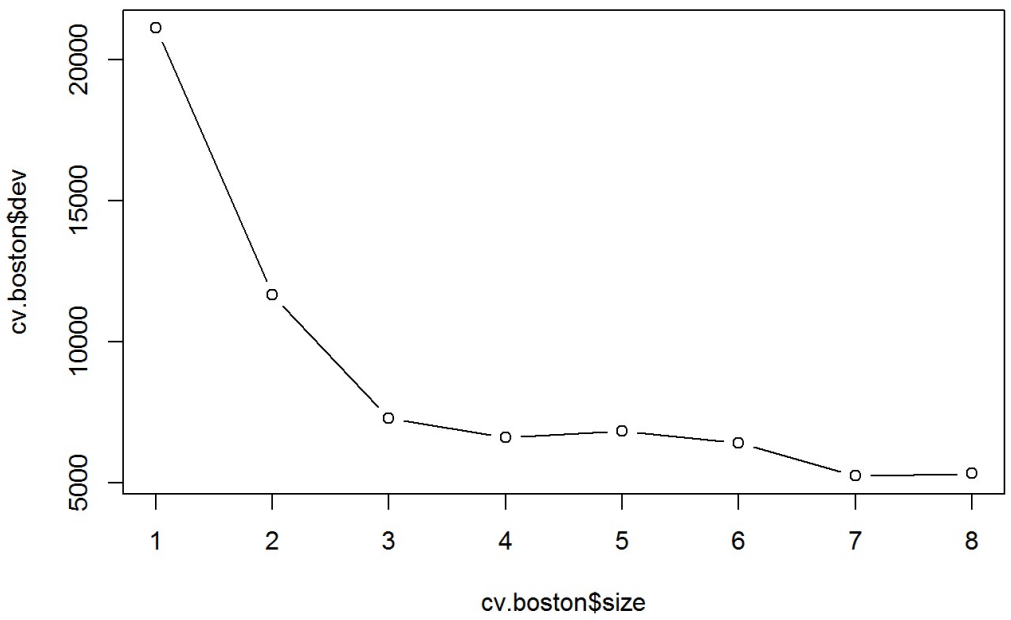
회귀 트리

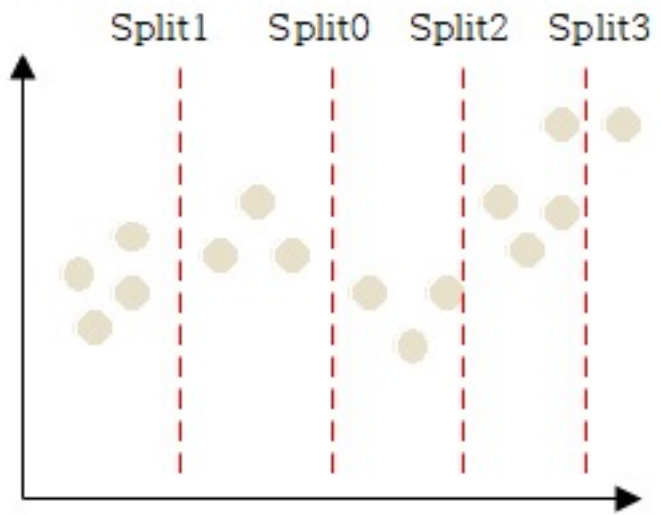
회귀 트리


days_since_2011
< 434.5
≥ 434.5
days_since_2011
temp
< 105.5
≥ 105.5
< 11.662
≥ 11.662
n = 106
n = 329
n = 70
n = 226
8000
6000
4000
2000
0

회귀 트리

회귀 트리


cv.boston$dev
20000
15000
10000
5000
1
2
3
4
5
6
7
8
cv.boston$size


Split1
Split0
Split2
Split3

회귀 트리

- 따라서 결정트리, 랜덤 포레스트, GBM, XGBoost, LightGBM 등 모든 트리 기반의 알고리즘은 분류 뿐만 아니라 회귀도 가능하다

- 또 성능도 괜찮게 나온다. (회귀 대회에서 이것들 쓴 것 같다 맞나..?)

정리

- 선형 회귀는 실제값과 예측값의 차이인 오류를 최소로 줄일 수 있는 선형 함수를 찾아서
- 오류는 RSS를 많이 사용하고 규제를 사용한다.
- 선형회귀를 기반으로 하는 로지스틱 회귀는 분류에서 많이 사용된다(특히 이진 분류)
- 회귀 트리 많이 사용하는 것이 좋다.
- 아 그리고 회귀는 데이터 정규화, 인코딩이 매우매우매우 중요하다.