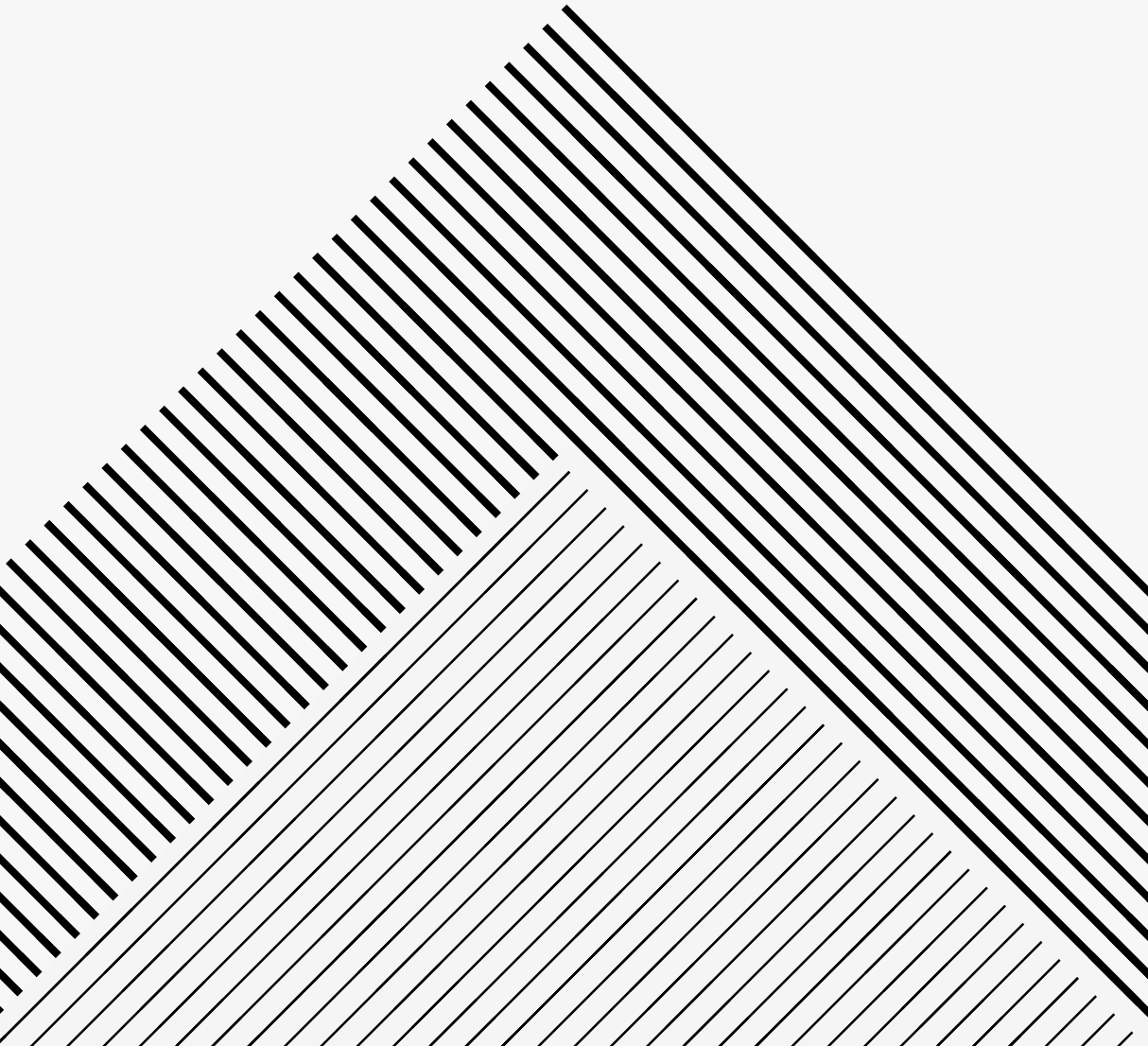
Regression

CS229



CH04. Regression (회귀)

5-1. 회귀소개

→ 여러개의 독립변수와 한개의 종속변수간의 상관관계를 모델링하는 기법

→ 머신러닝 회귀 예측의 핵심 = 최적의 regression hyperparameter를 찾는것

Linear Regression (선형 회귀) 9종

→ (실제값 − 예측값)^2 을 최소화하는 직선형 회귀선을 최적화

※ 대표적인 선형회귀 모델

규제 X

① 일반 선형회귀
- → 규제 X
- → Σ(예측−실제)^2 를 최소화하도록 회귀계수 최적화

규제 O

② 릿지 (Ridge) → feature의 값의 크기조정
- → ①에 L2 regression을 추가
  - └→ 상대적으로 큰 회귀계수 값에 예측 영향도를 감소시키기 위해서 회귀계수값을 더 작게 만듦

③ 라쏘 (Lasso) → feature의 개수조정
- → ①에 L1 regression을 추가
  - └→ 예측영향력이 작은 feature의 회귀계수를 0으로 만들어서 해당 feature이 선택되지 않도록한다.

④ 엘라스틱넷 (ElasticNet)
- → L2 + L1 + ①
- → feature이 많을때 사용

분류에 사용

⑤ 로지스틱회귀 (Logistic Regression)
- → 사실 Classification에 사용되는 Linear model이다.
- → 이진분류 + 희소(sparse)영역의 분류 (ex. 텍스트분류) 에서 강력함

4-2. 단순선형회귀를 통한 회귀 이해

→ 단순선형 회귀는 독립변수와 종속변수가 1개인 선형회귀이다.

→ $\hat{Y} = w_0 + w_1 X$ (= 예측값 (= 회귀선)) → 실제주택 가격은 이러한 항목값에서 오차값만큼의 오차를 빼거나 더한값이 된다.

- w_0: 절편 (=Intercept)
- w_1: 기울기

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

↳ 이러한 함수의 즉 비용함수가 반환하는 값을 지속적으로 감소시켜 (=오류값)

최종적으로 더이상 감소하지 않는 최소의 오류값을 구하는것이 목적이다.

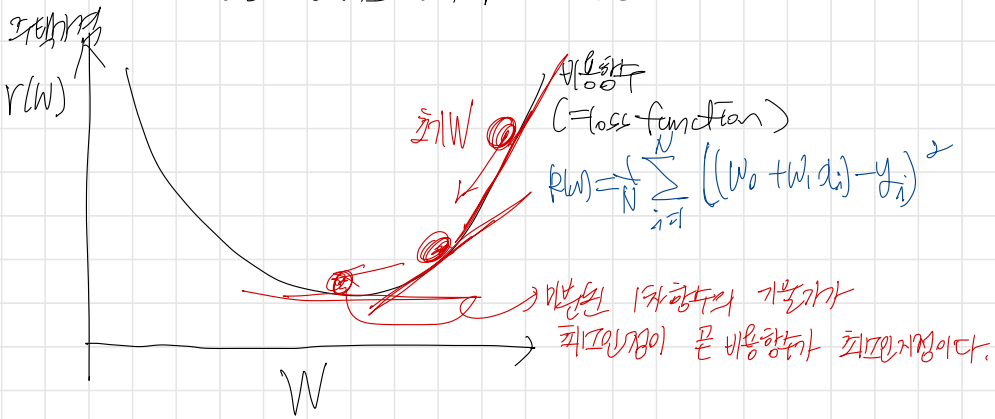
5-3. 비용최소화하기 — 경사하강법 (Gradient Descent)

→ 비용함수가 최소가되는 W parameter를 구해야 한다.

→ 특히나 W parameter의 개수가 많다면 고차방정식을 이용해서도 해결하기 힘들기 때문에 "정확적인 하강"을 이용하는 gradient descent를 사용하고자 한다.

→ 비용함수가 2차함수인 경우에는 경사하강법을 최초W에서부터 이분을 적용한뒤에 이 이분값이 제로 근접하는 방향으로 단계적으로 W를 update한다.

↳ 다시말해 이분된 1차함수의 기울기가 0에 근접하지 않는 W값까지를 비용함수 최적화지점으로 간주하고 그때의 W를 반환한다.



$$R(W) = \frac{1}{N}\sum_{i=1}^{N}\left((w_0 + w_1 x_i) - y_i\right)^2$$

→ 미분된 1차함수의 기울기가 최적화지점이 곧 비용함수 최적화지점이다.

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

$$\frac{\partial R(W)}{\partial w_1} = \frac{2}{N}\sum_{i=1}^{N} -x_i \times \left(y_i - (w_0 + w_1 x_i)\right) = -\frac{2}{N}\sum_{i=1}^{N}(y_i - \text{예측값}) \times x_i$$

→ 그러나 일반적인 gradient descent의 경우에는 속도가 매우느려 본편에서는 Stochastic Gradient Descent (확률적 경사 하강법)을 사용

└> 전체 데이터에서 랜덤하게 batch-size 만큼 데이터를 추출해서 이를 기반으로 w1-update, w0-update의 값을 계산하게 된다.

$$\hat{Y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = X_{mat} \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ \vdots & & & \\ x_{n1} & x_{n2} & \cdots & x_{nm} \end{bmatrix} * [w_1 \; w_2 \cdots w_m]^T + w_0$$

(Feature1, Feature2 ... Feature M)

shape(n,1) (m,n) (m,1)

* 내적 $(1,n)^T \rightarrow (n,1)$

= (m,1)

* 혹은 w_0를 이렇게 따로 더하지 말고 X_{mat}의 Feature 0을 (1값으로 채워진 열) 추가하고 w_0를 W배열에 앞에 추가해도 된것이다.

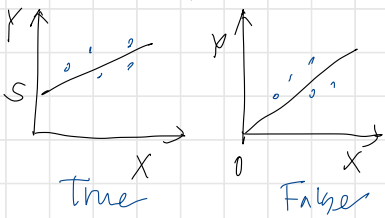
6-4. sklearn.linear_model.LinearRegression 사용하기 및 주요 파라미터 설명

→ 실제값과 예측값의 잔차 제곱합이 최소가 되는 직선을 구하여 OLS (ordinary least squares) 추정 방법으로 구현하는 class이다.

→ fit() 메서드로 X, y 배열을 입력받으면 회귀계수인 W를 coef_ 속성에 저장한다.

✳PARAMETERS

① fit_intercept



② normalize

→ fit_intercept = True인 경우 회귀를 수행하기 전에 입력데이터셋을 정규화

✳속성

① coef_

→ fit() 메서드를 수행했을 때 회귀계수가 배열형태로 저장하는 속성. (Target 개수, feature 개수)

② intercept_

→ ordinary Least Squares 기반의 회귀계수 계산은 입력 feature의 독립성에 영향을 많이받기 때문에 feature들간의 상관계수가 높은경우에 분산이 커져서 나쁜영향을 끼친다.

≒ 다중공선성 (multi-collinearity) 문제 발생

→ 따라서 상관관계가 높은 feature의 경우에는 독립적인 중요한 feature만 남기고 제거하거나 규제를 적용한다.

→ 또는 PCA를 이용해 차원축소를 하기도 한다.

※ 회귀평가 지표

→ 회귀의 성능을 위한 지표는 실제값과 회귀예측값의 차이를 기반으로 한다.

MAE $\frac{1}{n}\sum_{i}^{n}|Y_i - \hat{Y}_i|$

MSE $\frac{1}{n}\sum_{i}^{n}(Y_i - \hat{Y}_i)^2$

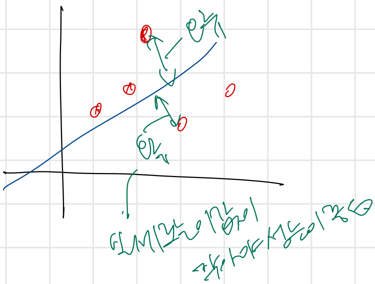
RMSE $\sqrt{\frac{1}{n}\sum_{i}^{n}(Y_i - \hat{Y}_i)^2} = \sqrt{MSE}$

R^2 $\frac{\text{예측값 분산}}{\text{실제값 분산}}$

✪ 회귀평가 기준을 적용할때에 주의할점

MAE → scoring 함수적용 → neg_mean_absolute_error

MSE → scoring 함수 적용 → neg_mean_squared_error



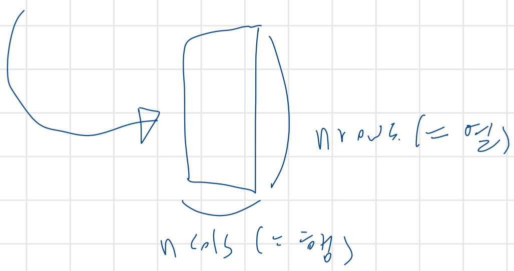
sklearn의 scoring 함수는 score이 클수록 좋은 평가 결과로 자동으로 평가 하는데, 실제값과 예측값의 차이를 기반하는 회귀평가 지표는 error 값이 크면 나쁜모델이므로 sklearn의 scoring 함수에 입력하려면 보정을해야 한다. → -1을 곱해주면됨 (실제값이기 때문에)

cf. sklearn의 scoring 함수

⇒ cross_val_score, GridSearchCV

cf. matplotlib을 이용한 여러개의 plot 그리기

→ matplotlib.subplots()는 여러개의 그래프를 한번에 표현하기위해 사용되는데, 인자로 ncols와 nrows를 입력



※ 회귀분석 // 시계열 모형에서의 예측오차 측정방법

1. $ME = \dfrac{\sum_{1}^{n}(Y_i - \hat{Y}_i)}{n}$

2. $RMSE = \sqrt{\dfrac{\sum_{1}^{n}(Y_i - \hat{Y}_i)^2}{n}}$

(편차의 정도가 너무 큼)

3. $MAE = \dfrac{\sum_{1}^{n}|Y_i - \hat{Y}_i|}{n}$

4. $MPE = \dfrac{\sum_{1}^{n}\frac{Y_i - \hat{Y}_i}{Y_i}\times 100}{n}$

⇒ RMSE, MAE와 같은 지표들 모두 참값과 예측값의 절대적인 예측오차에 기반하여 예측오차의 크기를 나타내는데, 절대적인 예측오차 뿐만 아니라 상대적인 의미의 예측오차도 필요하다.

→ 예를 들면 참값 10에 대한 예측오차 1과 참값 100에 대한 예측오차 1은 상대적으로 크기가 다를 것이다.

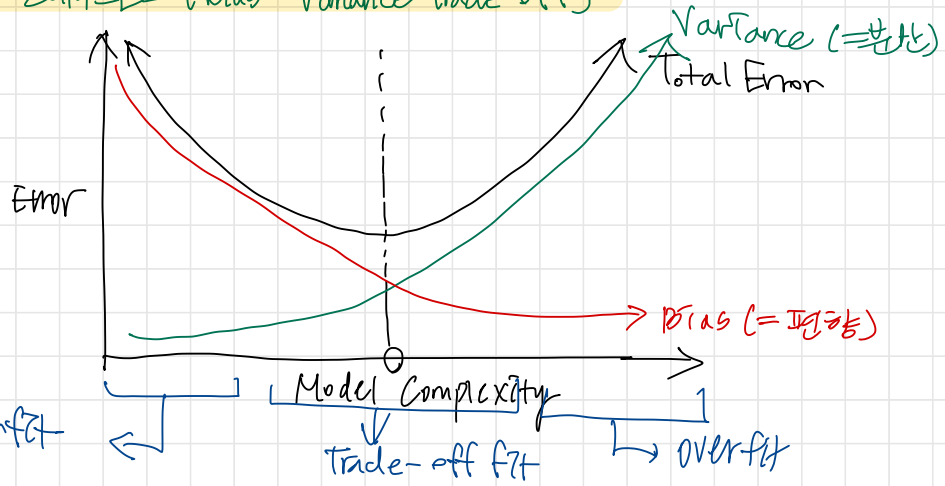
5. $MAPE = \dfrac{\sum_{1}^{n}\frac{|Y_i - \hat{Y}_i|}{Y_i}\times 100}{n}$

오차는 양수, 음수 모두 존재할 수 있기 때문에 합계를 그냥 내면 0이 될 수 있다는 MPE의 문제점을 개선한 방법이다.

6. $MASE = \dfrac{\sum_{1}^{n}\left(\dfrac{|Y_i - \hat{Y}_i|}{\frac{\sum_{2}^{n}|Y_i - Y_{i-1}|}{n-1}}\right)}{n} = \dfrac{\sum_{1}^{n}|Y_i - \hat{Y}_i|}{\frac{n}{n-1}\times\sum_{2}^{n}|Y_i - Y_{i-1}|}$

→ MAE는 데이터의 척도에 따라 유동적일 수 있다는 단점을 보완하기 위해서 데이터를 척도화하여 값을 계산한다.

㊗ 편향-분산 트레이드오프 (Bias-Variance Trade off)



- 지나치게 한방향성으로 치우친 경향이 있는 오델을 **고편향 (High Bias)** 라고 한다.
- 학습 하나 하나의 특성을 반영하면서 복잡한, 고차원 오델이 되고 지나치게 높은 변동성을 가진 오델은 **고분산 (High Variance)** 라고한다.

Low Variance

Low Bias

High Bias

편향이 높을수록 예측에서 멀어진다.

분산이높아짐 ← High Variance

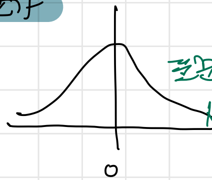
예측의 분포가 넓다.

∴ 머신러닝오델의 목표

= 편향과 분산이 서로 trade-off를 이루면서 cost(cost)값이 최대로 낮아지는 오델을 구축하는것. → ∴ 규제(Regularization)

※ 실제회귀데이터에 적용 - kaggle Bike Sharing Demand

→ target 값의 분포가 왜곡된 형태를 이루고 있는지 확인해보는 것이 회귀에서 낮은 예측 성능이 발생할 경우에 가장 먼저 ~~보아야~~ 하는 부분이다.
이때 정규분포를 이루고 있다면 특정값에 치우친 분포, 즉 왜곡된 분포가 아니기 때문에 보통 **로그를 적용해 변환**을 해준다.



$N(\mu, \sigma^2)$
※정규분포
↓
표준정규분포
$N(0, 1^2)$

→ sklearn을 이용한다면 이렇게 변경된 target값을 기반으로 학습하고 예측한 값을 다시 expm1() 함수를 이용하여 원래 scale로 원상복귀하면 된다.

※ 스케일 변환에는 이 외에도

- MinMaxScaler (최대는 1, 최소는 0으로 바꾸어주는 방법)
- StandardScaler (평균이 0, 분산이 1인 표준정규분포의 ~~형태로~~ 변환시킴)

→ 예측성능이 크게 좋아지진 않는다.

feature의 개수가 많은 경우에는 → 스케일링/정규화를 수행한 데이터셋에 다시 다항 특성을 적용하여 변환하는 방법

다항변환으로 형성되는 feature의 개수가 너무 많아지게 되어서 overfitting이 발생할 수 있다. (다항 회귀의 제일 큰 문제 = 과적합)

따라서 log는 줄여서 scale을 바꾸어주는 로그변환을 이용해서 정규분포와 최대한 가까운 형태로 바꾸어준다.

※ 데이터의 가공을 하여 모델투입을 추가적으로 해야하기에 고려해야함

① Feature dataset의 데이터분포도 → 각각의 모듈의 describe() 함수를 이용해 결측의 대략적인 분포를 살펴보고, 이에 반환값이 1이상일때 내용정보가 높다고 판단을 일반적으로 한다.
(만일 핫인코딩된 데이터를 볼때는 안된다)

② 이상치 데이터처리 (outlier)

→ 특히나 예측을 함에 있어서 target data에 상관계수가 높은 데이터의 경우에는 이상치를 제거하고자 하는 노력이 중요하다.

※ Stratified K-Fold

— k-fold교차검증은 kfold dataset을 만들고 k번학습한다.

→ 이때 stacking 모델은 두개가 필요하다.

① 개별적인 기반모델

② 이 개별기반모델의 예측데이터를 가지고 stacking의 학습데이터로 결합하여 최종 meta model의 학습용 feature set과 테스트용 feature set을 만든다.

※ CV set 기반의 stacking

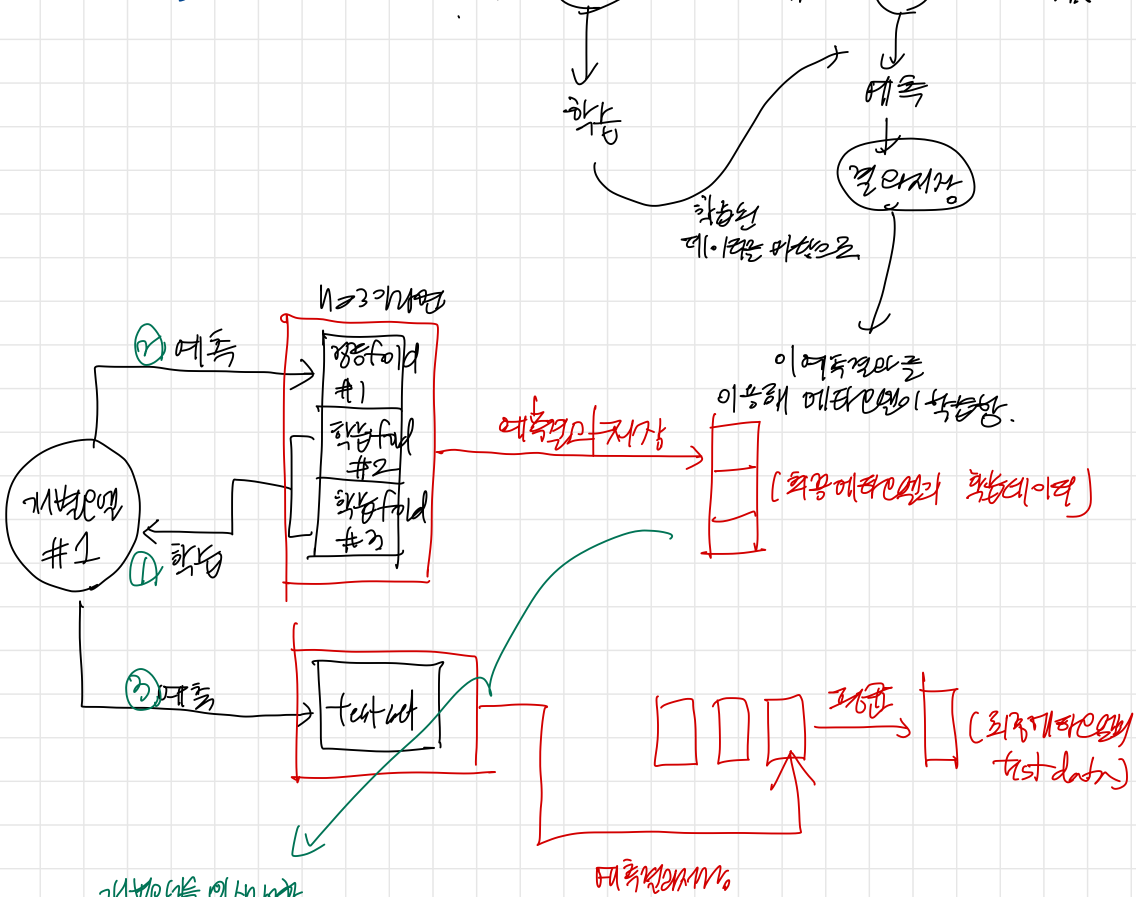
☆ (=Stratified K-Fold) 와 비슷한 원리 이용가능.

(과적합을 개선하기 위해서 최종메타 모델을 위한 데이터셋을 만들때 교차검증기반으로 예측된 결과 데이터세트를 이용)

① 학습용 데이터를 m개의 fold로 나눔

② 마지막 반복에서 개별모델의 예측값으로 학습데이터와 test data 생성

① 학습용데이터를 m개 중 n-1개는 학습을 위해, 나머지 1개는 검증을 위해.



개별모델들의 생성한
학습데이터 = 최종메타모델이 train에 사용

개별모델들의 test 예측결과 = 최종메타모델의 test에 사용

⊛ 정리

① 비용함수 (회귀계수 w를 변수로 가정)를 최소화하는 회귀계수 w의 값을 찾는 것이 목적

→ 이를 위해 경사하강법을 사용

→ 비용함수의 최적화 기법☆

But 학습데이터에 overfitting될 문제도 발생가능

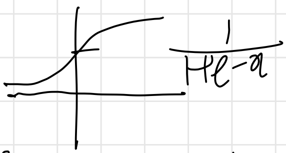
해결책

→ Regression
- Lasso (L_1규제) ⇒ $RSS(W) + \alpha \times ||W||_1$
- Ridge (L_2규제) ⇒ $RSS(W) + \alpha \times ||W||_2^2$
- ElasticNet ($L_1 + L_2$) ⇒ $RSS(W) + \alpha_1 ||W||_1 + \alpha_2 \times ||W||_2^2$

⇓

이를 최소화 하는 것!!!

② Logistic 회귀



→ 선형함수 대신 최적의 Sigmoid 함수를 도출하여 독립변수(feature)를 이 Sigmoid 함수에 입력해 반환된 결과를 확률값으로 바꾸어 예측 label 결정

③ 회귀트리 — 리프노드에 속한 data의 평균을 구해 예측값 계산

④

편향된 데이터의 분포곡선

→ 이때 log변환 적용

⊕ object형 데이터는 원-핫 인코딩 적용 (특히 선형회귀)