* **회귀(Regression) 개요**

회귀는 여러 개의 독립변수와 한 개의 종속변수 간의 상관관계를 모델링하는 기법을 통칭함.

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

* **분류(Classification)와 회귀(Regression)**

Classification 🡪 (결과값) Category 값 (이산 값)

Regression 🡪 (결과값) 숫자 값 (연속 값)

* **선형 회귀의 종류**

**일반 선형 회귀 :** 예측 값과 실제 값의 RSS(Residual Sum of Squares)를 최소화할 수 있도록 회귀계수를 최적화하며, 규제(Regularization)를 적용하지 않은 모델

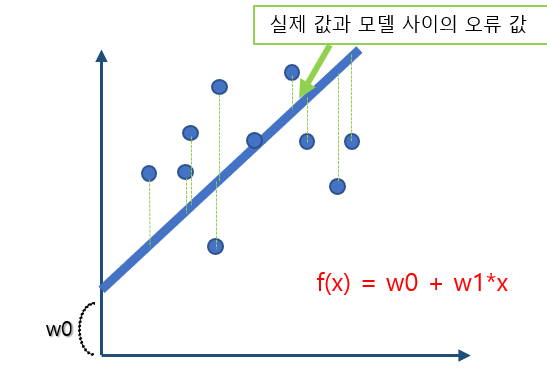
**릿지(Ridge) :** 릿지 회귀는 선형 회귀에 L2규제를 추가한 회귀 모델

**라쏘(Lasso) :** 라쏘 회귀는 선형 회귀에 L1규제를 적용한 방식

**엘라스틱넷(ElasticNet) :** L2, L1 규제를 함께 결합한 모델

**로지스틱 회귀( Logisitc Regression)** : 분류에 사용되는 선형모델

* **단순 선형 회귀(Simple Regression)을 통한 회귀의 이해**

****

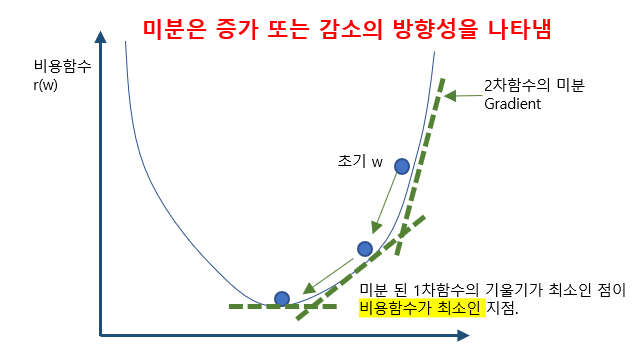
데이터의 잔차(오류 값)의 합이 최소가 될 수 있게 만듦.

* **RSS 기반의 회귀 오류 측정**

**RSS (비용함수/손실함수) (Loss/Cost function) :** 오류 값의 제곱을 구해서 더하는 방식

머신러닝 회귀 알고리즘은 RSS(오류 값)이 더 이상 감소하지 않는 최소의 오류 값을 구하는 것.

* **비용 최소화하기 – 경사 하강법(Gradient Descent) 소개**

****

미분된 1차 함수의 기울기가 감소하지 않는 지점 = 비용 함수가 최소인 지점 그때의 w를 반환.

* **RSS의 편미분**

=

*=*

* **경사 하강 법 정리 : 비용함수가 최소가 되는 값 찾기**

실제로는 편미분 값이 너무 클 수도 있어서 보정계수 n (0~1) 을 곱함.

새로운 =

새로운 *=이전*

* **사이킷런 LinearRegression 클래스**

**fit\_intercept :**  절편의 유무 ( default = True )

**normalize:** 세트 정규화 ( default = False ) 🡪 fit\_intercept 가 False 이면 무시됨.

**coef\_:** fit() 메서드 수행 시 회귀 계수가 배열로 저장됨.

**intercept\_:** intercept 값

* **선형 회귀의 다중 공선성 문제**

독립변수들 간의 상관이 있는 경우 🡪 다중 공선성(multi – collinearity) 문제

독립적인 중요한 피처만 남기고 제거하거나 규제함.

* **회귀 평가 지표**

|  |  |  |
| --- | --- | --- |
| **평가지표** | **설명** | **수식** |
| **MAE** | Mean Absolute Error  실제 값과 예측 값의 차이를 절대값으로 변환해 평균. |  |
| **MSE** | Mean Squared Error  실제 값과 예측 값의 차이를 제곱해 평균. |  |
| **MSLE** | MSE에 로그를 적용. 결정 값이 클수록 오류 값도 커지기 때문에 전체 오류 값이 커지는 것을 방지함. |  |
| **RMSE** | MSE에 루트를 씌운 것.  MSE는 제곱 값으로 실제 오류 평균보다 더 커지는  특성존재 |  |
| **RMSLE** | RMSE에 로그를 적용. 결정 값이 클수록 오류 값도 커지기 때문에 전체 오류 값이 커지는 것을 방지함. |  |
|  | 분산 기반으로 예측 성능 평가  1에 가까울수록 예측 정확도가 높음. |  |

* **사이킷런 회귀 평가 API**

사이킷런은 RSME를 제공하지 않아, MSE에 제곱근을 씌워서 계산하는 함수를 직접 만들어야 함.

|  |  |  |
| --- | --- | --- |
| **평가방법** | **사이킷런 평가지표 API** | **Scoring 함수 적용 값** |
| **MAE** | metrics.mean\_absolute\_error | ‘neg\_mean\_absolute\_error’ |
| **MSE** | metrics.mean\_squared\_error | ‘neg\_mean\_squared\_error’ |
|  | metrics.r2\_score | ‘r2’ |

**MAE는 절댓값의 합이므로 음수가 나올 수 없습니다**.

Scoring 함수에 적용 시, score값이 클수록 좋은 성능이기 때문에, MAE에 **-1 을 곱하여,** **음수로 만들어 작은 오류 값이 더 큰 숫자로 인식**하게 합니다. ex) 10>1 🡪 -10 < -1

즉, neg\_mean\_absolute\_error 는 metrics.mean\_absolute\_error **\* (-1)** 인 것입니다.

* **다항회귀 (Polynomial Regression) 개요**

독립변수가 2차 3차 방정식과 같은 다항식으로 표현되는 것.

* 직선보다 **다항 회귀 곡선형으로** 표현한 것이 **예측 성능이 더 높음.**

**회귀에서 선형/비선형 회귀를 나누는 기준 : 회귀계수가 선형/비선형인지**

**선형 회귀 : y = w0 + w1\*x1 + w2\*x2 + w3\*x1\*x2**

**비선형 회귀 : y = w1\*cos(x+w4) + w2\*cos(2\*x+w4) + w3**

* **사이킷런에서의 다항회귀 PolynomialFeatures**

degree 차수에 따라 다항식을 적용

단항피처 [] 를 Degree=2 2차 다항 피처로 변환

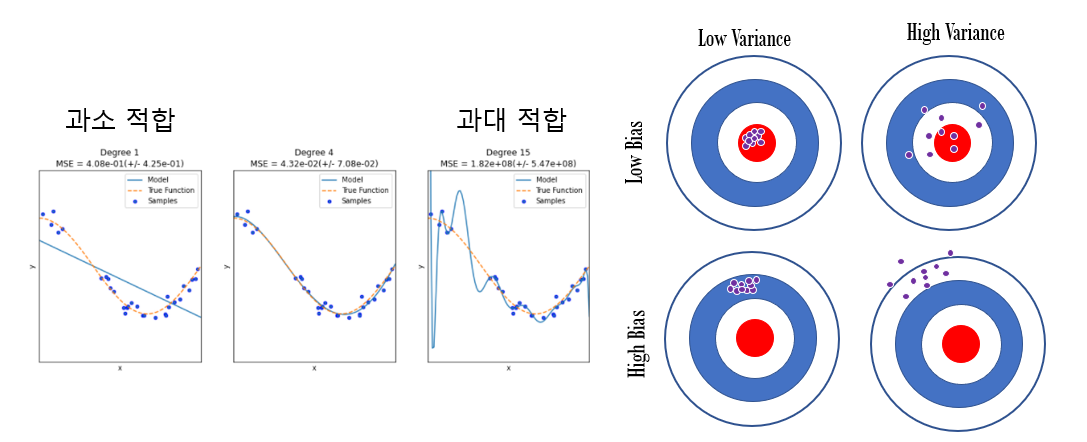
* (

단항피처 []를 Degree=3 3차 다항 피처로 변환

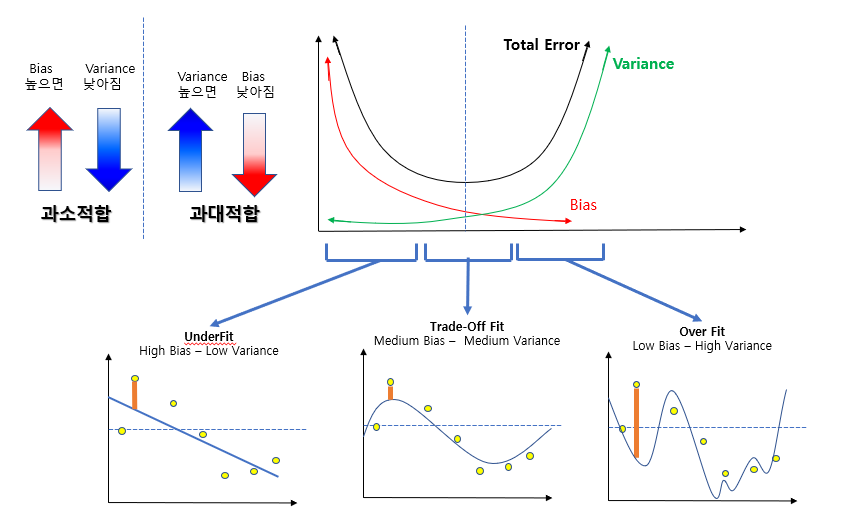
* (

PolynomialFeatrues 변환 🡪 LinearRegression 학습 (2번을 거쳐야 함.)

* **Pipeline 클래스** 이용 ( PolynomialFeatures 변환 + LinearRegression 학습/예측 결합)
* **편향-분산 트레이드 오프(Bias-Variance Trade Off)**



\*Bias : 타겟 값과 멀어져있으면 Bias up // \*Variance : 변동성



* **규제 선형 회귀 개요**

다중 선형 회귀에서 Degree =15 하게 되면, Overfiting 🡪 test데이터 세트 형편없는 예측 성능

따라서 회귀 모델은 **회귀 계수가 기하급수적으로 커지는 것을 제어** 해야함.

텍스트이(가) 표시된 사진

자동 생성된 설명

비용함수 목표 = **Min ( RSS(W) + alpha\*|| )**

**\*alpha는 학습데이터 적합 정도와 회귀 계수 값의 크기 제어를 수행하는 튜닝 파라미터**

**alpha 감소 : RSS(W) 최소화**

alpha가 0인 경우 W가 커도 alpha\*||가 0이 되어 비용 함수는 Min(RSS(W))

**alpha 증가 : 회귀 계수 W 감소**

alpha가 무한대인 경우 alpha\*||도 무한대가 되므로 비용 함수는 W를 0에 가깝게 최소화해야 함.

* **규제 선형 회귀의 유형**

alpha값으로 페널티를 부여해 회귀 계수 값의 크기를 감소시켜 과적합을 개선하는 방식 : 규제

규제 – **L2 : 릿지(Ridge)회귀** : alpha\*|| 와 같이 에 대해 페널티 부여

- **L1 : 라쏘(Lasso)회귀** : alpha\*||과 같이 **|w|**에 대해 페널티 부여

🡪 **L1규제는 불필요한 회귀계수를 0으로 만들고 제거함**. (Feature Selection)

**- L1 + L2 : ElasticNet** : 주로 피처가 많은 데이터 세트에서 적용됨.

**Min ( RSS(W) + alpha\*|| + alpha\* )**

🡪 L1규제로 피처의 개수 줄이고 L2규제로 계수 값의 크기를 조정함.

**🡪 alpha 파라미터** : L1규제의 alpha 값 (a)+ L2규제의 alpha 값 (b)

**🡪 l1\_ratio 파라미터** : a / ( a +b )

* **선형회귀 모델을 위한 데이터 변환**

회귀모델과 같은 선형 모델은 일반적으로 feature와 target값 간에 선형관계가 있다고 가정.

* **선형회귀 모델은 feature 값과 target 값의 분포가 정규분포를** 매우 선호함
* **회귀를 위한 데이터 변환 방법**

**선형 회귀의 데이터 인코딩은 원-핫 인코딩을 적용함** ( 레이블 인코딩(category) X )

|  |  |
| --- | --- |
| **변환 대상** | **설명** |
| **target 값 변환** | 회귀에서 target은 반드시 정규 분포를 가져야 함.  주로 **로그변환**을 적용 |
| **feature 값 변환** | **1.** **StandardScaler** 클래스 이용 (평균 0 분산 1인 표준 정규 분포)  **MinMaxScaler** 클래스 이용 (최솟값 0 최댓값 1인 정규화 수행) |
| **2.** 스케일링/정규화를 수행한 데이트 세트에  다시 **다항 특성을** 적용하여 변환하는 방법  1번 방법 이후 예측 성능 향상이 없을 경우 2번 시행 |
| **3.** **로그 변환 (매우 유용) :** 정규 분포에 가까운 형태로 값이 분포됨.  1번방법 – 예측 성능 향상을 크게 기대하기 어려움  2번방법 – feature의 개수가 많으면 과 적합 발생 가능 |

* **로지스틱 회귀 개요**

로지스틱 회귀는 선형 회귀 방식을 분류에 적용한 알고리즘

Sigmoid함수 최적선을 찾고, Sigmoid 함수의 반환 값을 확률로 간주해 분류를 결정함. 텍스트이(가) 표시된 사진

자동 생성된 설명

단순 선형 회귀 : y=w1x + w0가 있다고 할 때

로지스틱 회귀는 0과 1을 예측하기에 단순 회귀식은 의미가 없음.

따라서 Odds를 선형 회귀식에 적용

**Odds ) = p /(1-p)**

확률 p의 범위는 ( 0,1)이고, 선형회귀 반환값의 범위는 ( **-∞, ∞** )이므로 로그 변환 수행

**Log( Odds(p) ) = w1x + w0**

해당 식을 p에 대해 정리하면 **p(x) =**

로지스틱 회귀는 학습을 통해 **Sigmoid 함수의 w 값을 찾아**, 예측하는 것.

* **로지스틱 회귀 특징과 사이킷런 로지스틱 회귀 클래스**

로지스틱 회귀는 가볍고 빠르고 이진 분류 예측 성능도 뛰어남.

또한 로지스틱 회귀는 **희소한 데이터 세트 분류에도 뛰어난 성능**을 보여 **텍스트 분류에서도 자주 사용됨.**

**LogisticRegression 하이퍼 파리미터**

penalty = ( ‘l1’ , ‘l2’ ) L1규제 또는 L2규제

**C=** ( 1/alpha ) C는 alpha의 역수 값으로, C 값이 작을수록 규제강도가 큼

* **회귀 트리 개요**

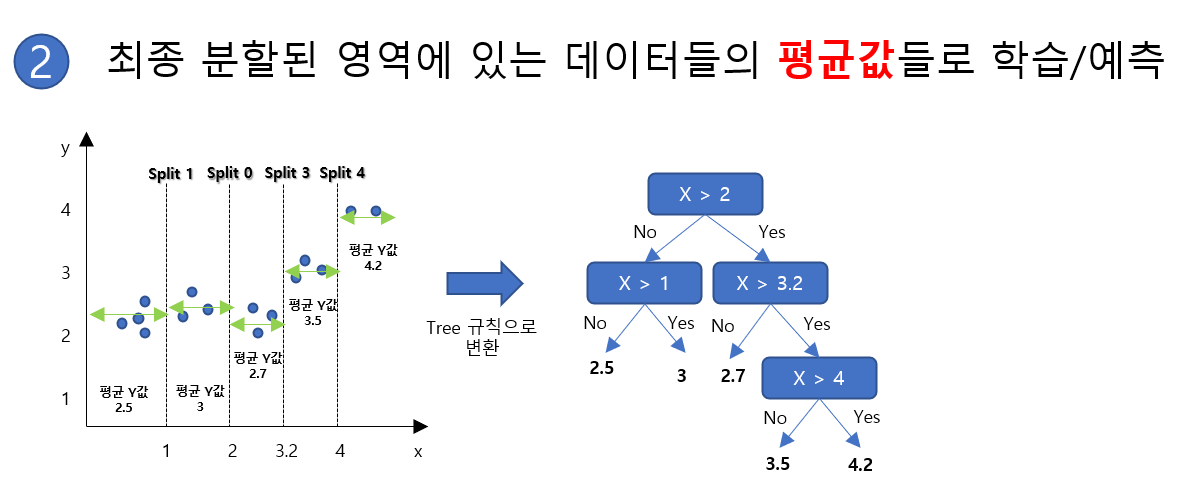
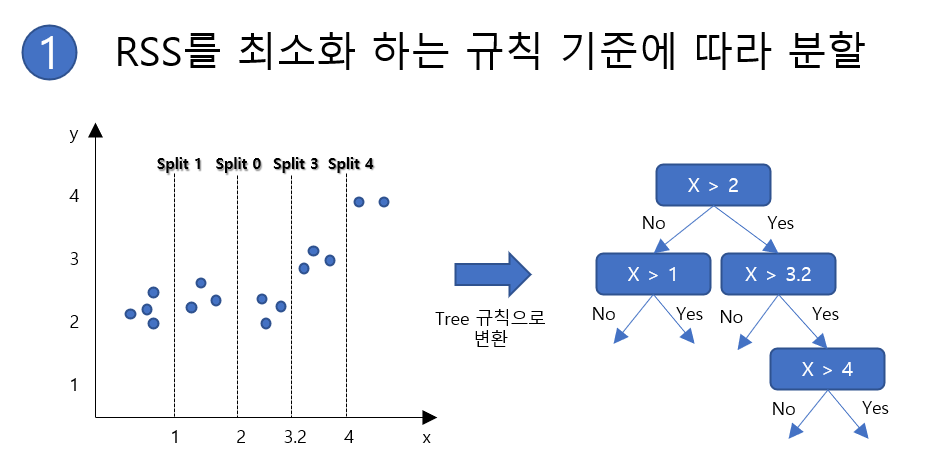
사이킷런의 **결정 트리 및 결정 트리 기반 앙상블 알고리즘**은 **분류 회귀** 모두 가능

**트리가 CART(Classification and Regression Tree)기반**으로 만들어졌기 때문

CART는 **분류와 회귀** 모두 가능한 분할 알고리즘

CART 회귀 트리의 **분할 기준은 RSS(SSE)가 최소**가 될 수 있는 기준을 찾아 분할됨

최종 분할이 완료된 후 각 분할 영역에 있는 데이터 결정 값들의 **평균 값**으로 학습/예측함



* **사이킷런의 회귀 트리 지원**

|  |  |  |
| --- | --- | --- |
| **알고리즘** | **회귀 Estimator 클래스** | **분류 Estimator 클래스** |
| Decision Tree | DecisionTreeRegressor | DecisionTreeClassifier |
| Gradient Boosting | GradientBoostingRegressor | GradientBoostingClassifier |
| XGBoost | XGBRegressor | XGBClassifier |
| LightGBM | LGBMRegressor | LGBMClassifier |

* **회귀 트리의 오버 피팅**

복잡한 트리 구조를 가질 경우 오버 피팅하기 쉬우므로

**트리의 크기와 노드 개수의 제한** 등의 방법을 통해 **오버 피팅을 개선**할 수 있음