유런 5주차 예습과제 필사

▼ ch 5. 회귀

▼ 01. 회귀 소개

- 회귀 분석 : 데이터 값이 평균과 같은 일정한 값으로 돌아가려는 경향을 이용한 통계학 기법
- 회귀
  - 여러 개의 독립변수와 한 개의 종속변수 간의 상관관계를 모델링하는 기법을 통칭
  - 예) 독립변수 : 아파트의 방 개수, 방 크기, 주변 학군 / 종속변수 : 아파트 가격 → 어떤 관계를 나타내는지 모델링하고 예측하는 것
  - 선형 회귀식 예시

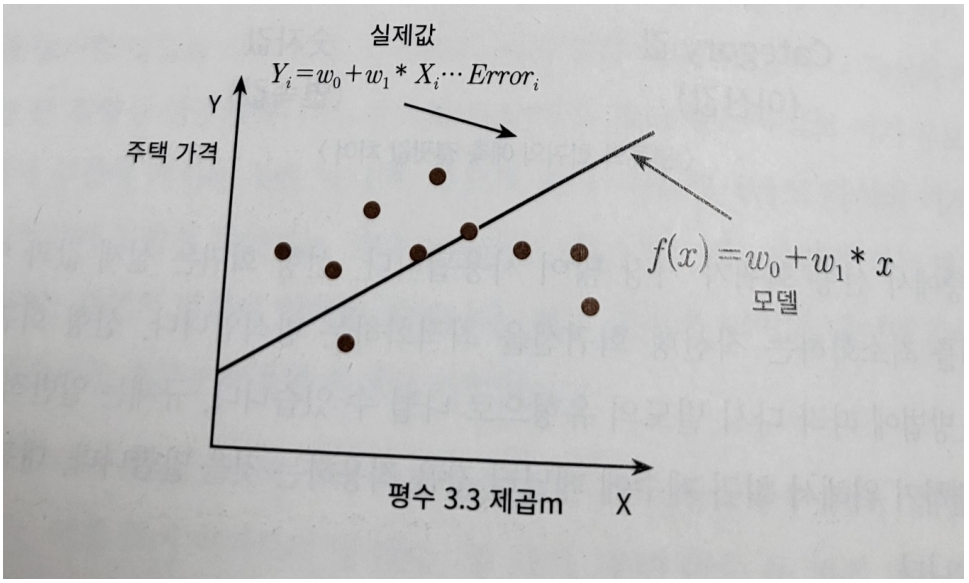
$$Y = W_1 * X_1 + W_2 * X_2 + W_3 * X_3 + ... + W_n * X_n$$

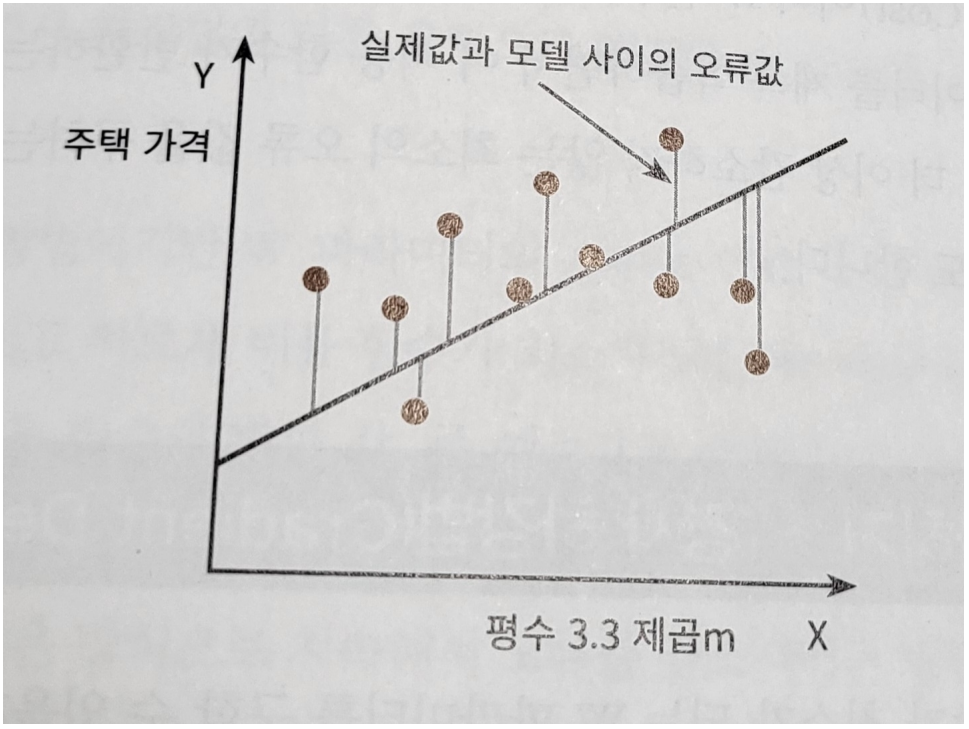
    - Y는 종속변수, X는 독립변수들, W는 독립변수의 값에 영향을 미치는 회귀 계수를 의미함
    - 머신 러닝 관점에서 독립변수 : 피처, 종속변수 : 결정 값
    - 머신러닝 회귀 예측의 핵심 : 주어진 피처와 결정 값 데이터 기반에서 **학습을 통해 최적의 회귀 계수를 찾아내는 것**
  - 회귀는 회귀 계수의 선형/비선형 여부, 독립변수의 개수, 종속변수의 개수에 따라 여러 유형으로 나눌 수 있음
  - 회귀에서 가장 중요한 것 ⇒ 회귀 계수
    - 회귀 계수가 선형 : 선형 회귀 / 회귀 계수가 선형이 아님 : 비선형 회귀
    - 독립 변수의 개수가 1개 : 단일 회귀 / 독립 변수의 개수가 여러 개 : 다중 회귀
- 지도학습
  - 분류, 회귀 두 유형으로 나뉨
  - 분류 : 예측값이 카테고리와 같은 이산형 클래스의 값 vs 회귀 : 연속형 숫자 값
- 선형 회귀

    - 가장 많이 사용되는 회귀
    - 실제 값과 예측값의 차이(오류의 제곱 값)를 최소화하는 직선형 회귀선을 최적화하는 방식
    - 규제(regularization) 방법에 따라 별도의 유형으로 나뉠 수 있음
        - 규제 : 일반적인 선형 회귀의 과적합 문제를 해결하기 위해 회귀 계수에 패널티 값을 적용하는 것
    - 대표적인 선형 회귀 모델
        - 일반 선형 회귀 : 예측값과 실제 값의 RSS를 최소화할 수 있도록 회귀 계수 최적화, 규제 X
        - 릿지 : 선형 회귀에 L2 규제를 추가한 회귀 모델, L2 규제는 상대적으로 큰 회귀 계수 값의 예측 정확도를 감소시키기 위해 회귀 계수값을 더 작게 만드는 규제 모델
        - 라쏘 : 선형 회귀에 L1 규제를 추가한 회귀 모델, L1 규제는 예측 영향력이 작은 피처의 회귀 계수를 0으로 만들어 회귀 예측 시 피처가 선택되지 않도록 함(=피처 선택 기능)
        - 엘라스틱넷 : L2, L1 규제를 함께 결합한 모델, 주로 피처가 많은 데이터 세트에서 적용됨, L1 규제로 피처의 개수를 줄임과 동시에 L2 규제로 계수 값의 크기 조정
        - 로지스틱 회귀 : 사실은 분류에 사용되는 선형 모델, 매우 강력한 분류 알고리즘으로 이진 분류, 희소 영역의 분류에 뛰어남

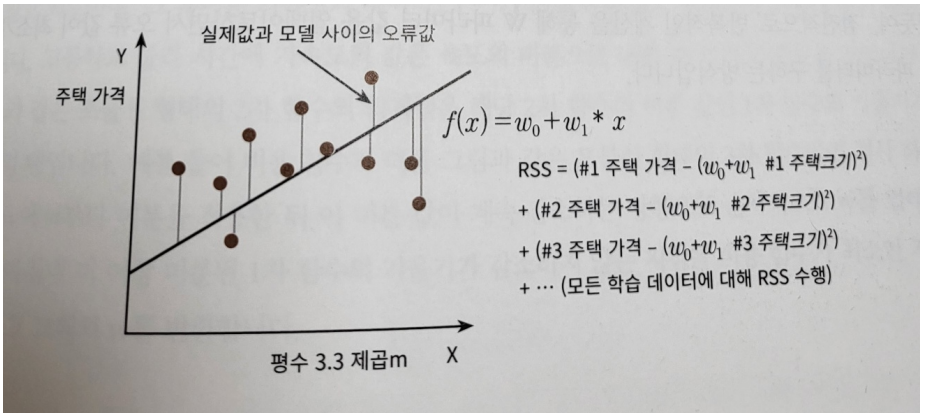
▼ 02. 단순 선형 회귀를 통한 회귀 이해

- 단순 선형 회귀
    - 독립변수 1개, 종속변수 1개인 선형 회귀
    - 예) 독립변수 : 주택의 크기, 종속변수 : 주택 가격 → 특정 기울기와 절편을 가진 1차 함수식으로 모델링
        - 독립 변수가 1개인 단순 선형 회귀 : 기울기 w1과 절편 w0을 회귀 계수로 지칭
        - 실제 주택 가격 : 1차 함수 값에서 실제 값만큼의 오류 값을 빼거나 더한 값이 됨





- 잔차 : 실제 값과 회귀 모델 차이에 따른 오류 값
- 최적의 회귀 모델 = 전체 데이터의 잔차 합이 최소가 되는 모델, 오류 값 합이 최소가 될 수 있는 최적의 회귀 계수를 찾는다는 의미
- 오류 값은 +나 -가 될 수 있음 → 절댓값을 취해 더하거나(Mean Absolute Error) 오류 값의 제곱을 구해서 더하는 방식(RSS)를 취함
    - 일반적으로 RSS 방식으로 오류 합을 구함 ⇒ Error^2 = RSS



- RSS를 최소로 하는 회귀 계수를 학습을 통해 찾는 것이 머신러닝 기반 회귀의 핵심
- RSS는 회귀 식의 독립 변수 X, 종속 변수 Y가 중심 변수가 아니라 w 변수(회귀 변수)가 중심 변수 (독립 변수와 종속 변수는 RSS에서 모두 상수로 간주)
- RSS는 학습 데이터의 건수로 나누어서 다음과 같이 정규화된 식으로 표현됨

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

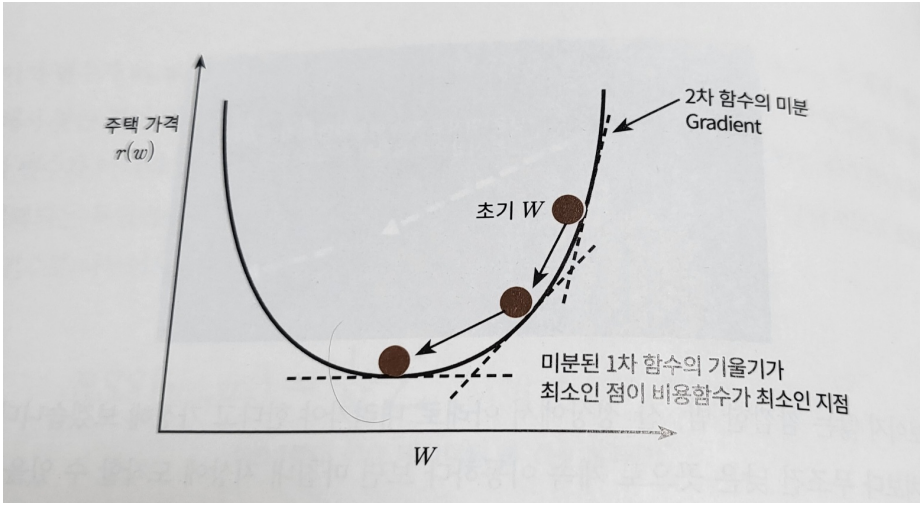
(i는 1부터 학습 데이터의 총 건수 N까지)

  - RSS는 비용이며 w 변수로 구성되는 RSS를 비용 함수라고 함
  - 머신러닝 회귀 알고리즘 : 데이터를 계속 학습 → 비용 함수가 반환하는 값을 지속해서 감소시키고 최종적으로는 더 이상 감소하지 않는 최소의 오류 값을 구함
  - 비용 함수 = 손실 함수

▼ 03. 비용 최소화하기 - 경사 하강법(Gradient Descent) 소개

- 비용 함수가 최소가 되는 W 파라미터 구하는 방법
  - 경사 하강법
    - ‘점진적으로’ 반복적인 계산을 통해 W 파라미터 값을 업데이트하면서 오류 값이 최소가 되는 W 파라미터를 구하는 방식
    - 핵심 : 어떻게 하면 오류가 작아지는 방향으로 W값을 보정할 수 있는가

- 고차원 방정식에 대한 문제를 해결해주면서 비용 함수 RSS를 최소화하는 방법을 직관적으로 제공
- '데이터를 기반으로 알고리즘이 스스로 학습한다'는 머신러닝의 개념을 가능하게 만들어준 핵심 기법 중 하나
- 예) 비용 함수가 포물선 형태의 2차 함수라면? 최초 w에서 미분을 적용한 뒤 미분 값이 계속 감소하는 방향으로 순차적으로 w 업데이트 → 미분된 1차 함수 기울기가 더 이상 감소하지 않는 지점을 비용 함수가 최소인 지점으로 간주 → 그때의 w 반환



- 비용함수를 R(w)로 지칭, R(w)는 두 개의 w 파라미터인 w0, w1을 각각 가지고 있어 각 변수에 편미분을 적용해야 함.
- 편미분을 통해 R(w)를 최소화하는 w0, w1 값을 얻을 수 있음
  - 업데이트는 새로운 w1을 이전 w1에서 편미분 결과값을 마이너스하면서 적용
  - 편미분 값이 너무 클 수 있어 보정 계수를 곱함(=학습률)

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

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

- 경사 하강법의 일반적인 프로세스
  1. w1,w0를 임의 값으로 설정, 첫 비용 함수의 값 계산
  2. w1, w0을 위 수식으로 업데이트한 후 다시 비용 함수의 값 계산

3. 비용함수가 감소하는 방향성으로 주어진 횟수만큼 2단계를 반복하면서 w1, w0 업데이트

- 경사 하강법을 파이썬 코드로 구현하기
  - Y = 4X+6을 근사하기 위해 100개의 데이터 세트 만듦 → 경사 하강법 이용해 회귀 계수 w0, w1 도출
  - get_cost() : 비용함수, 실제 y값과 예측된 y값을 인자로 받아 1/N*[1부터 N까지 (실제값 - 예측값)^2]을 계산해 반환함
  - gradient_descent() 함수 : w1과 w0을 모두 0으로 초기화한 후 tiers 개수만큼 반복하면서 두 값을 업데이트함
    - 위에서 무작위로 생성한 X와 y를 받음 (X, y는 모두 넘파이 ndarray)
  - get_weight_update() 함수 : w1_update로 -보정계수*w1 편미분 값을, w0_update로 -보정계수*w0 편미분 값을 넘파이 dot 행렬 연산으로 계산한 뒤 이를 반환함
  - gradient_descent_steps() : get_weight_updates()을 경사 하강 방식으로 반복적으로 수행하여 w1, w0을 업데이트하는 함수
  - 최종적으로 예측갑소가 실제값의 RSS 차이를 계산하는 get_cost() 함수를 생성하여 예측 오류 계산
- 경사 하강법 단점 : 비용함수 최소화를 위한 값을 업데이트하기 때문에 수행 시간이 매우 오래 걸림 → 실전에서 대부분 확률적 경사 하강법 이용
  - 확률적 경사 하강법 : 전체 입력 데이터가 아니라 일부 데이터만 이용해 w가 업데이트되는 값을 계산해 빠른 속도 보장
    - stochastic_gradient_descent_steps() 함수로 구현함
      - batch_size만큼 데이터를 추출해 w 업데이트
    - 경사 하강법으로 구한 w1, w0와 큰 차이가 없고 예측 오류 비용 또한 비슷함
- 피처가 여러 개인 경우 회귀 계수 도출 방법
  - 피처가 M개 있다면 회귀 계수는 M+1개로 도출됨
  - 예측 회귀식 : Y = w0+ w1*X1+ … + w100*X100과 같이 만들 수 있음

        - 데이터 개수가 N, 피처 M개의 입력 행렬을 Xmat, 회귀 계수를 W 배열로 표기 → 예측 행렬 Y = np.dot(Xmat, W^t)+w0
        - 회귀 예측값 : Y = Xmat*W^T

▼ 04. 사이킷런 LinearRegression을 이용한 보스턴 주택 가격 예측

- 사이킷런 linear_models 모듈 : 다양한 종류의 선형 기반 회귀를 클래스로 구현해 제공
- LinearRegression 클래스 - Ordinary Least Squares
    - 예측값과 실제값의 RSS를 최소화해 OLS 추정 방식으로 구현한 클래스
    - fit() 메서드로 X,y 배열을 입력받으면 회귀 계수인 W를 coef_속성에 저장
    - 입력 파라미터
        - fit_intercept : boolean 값으로 디폴트는 True. 절편 값을 계산할 것인지 말지 결정
            - False로 지정하면 절편이 사용되지 않고 0으로 지정됨
        - normalize : boolean 값으로 디폴트는 False(파라미터 무시), True이면 회귀 수행 전 데이터 세트를 정규화함
        - coef_ : fit() 메서드를 수행했을 때 회귀 계쑤가 배열 형태로 저장하는 속성, Shape는 (Target 값 개수, 피처 개수)
        - intercept_ : 절편 값
    - OLS 기반 회귀 계수 계산 : 입력 피처의 독립성에 많은 영향을 받음
        - 피처 간의 상관관계가 매우 높은 경우 분산이 매우 커져서 오류에 매우 민감해짐(=다중 공선성 문제)
    - 일반적으로 상관관계가 높은 피처가 많은 경우 독립적인 중요한 피처만 남기고 제거하거나 규제 적용
    - 매우 많은 피처가 다중 공선성 문제를 가지고 있다면 PCA를 통해 차원 축소할 수도 있음
- 회귀 평가 지표
    - 실제 값과 회귀 예측값의 차이를 기반으로 한 지표가 중심
    - 오류의 절댓값 평균이나 제곱, 또는 제곱한 뒤 다시 루트를 씌운 평균값을 구함

    - MAE : Mean Absolute Error로 실제값과 예측값이 차이를 절댓값으로 변환해 평균한 것
        - 사이킷런 평가 지표 API : metrics.mean_absolute_error
    - MSE : Mean Square Error로 실제값과 예측값이 차이를 제곱해 평균한 것
        - 사이킷런 평가 지표 API : metrics.mean_squared_error
    - RMSE : MSE에 루트를 씌운 것
        - 사이킷런은 RMSE를 제공하지 않아 직접 MSE에 제곱근을 씌워서 계산하는 함수 만들어야 함
        - 사이킷런 평가 지표 API : metrics.mean_squared_error() 함수를 그대로 사용하되, squared 파라미터를 False로 지정해 사용함
    - R^2 : 분산 기반으로 예측 성능 평가, 실제 값의 분산 대비 예측값의 분산 비율을 지표로 함. 1에 가까울수록 예측 정확도가 높음
        - 사이킷런 평가 지표 API : metrics.r2_score
    - cross_val_score, GridSearchCV와 같은 Scoring 함수에 회귀 평가 지표 적용시 유의점
        - 예) MAE의 scoring 파라미터 값을 살펴보면 'neg_'라는 접두어가 붙어 있음 → 음수 값을 가진다는 의미인데 MAE는 절댓값의 합이기 때문에 음수가 될 수 없음
        - 'neg'를 적용하여 음수값을 반환하는 이유? Scoring 함수가 score값이 클수록 좋은 평가 결과로 자동 평가하기 때문 → 회귀 평가 지표의 경우 값이 커지면 오히려 나쁜 모델이므로 scoring 함수에 반영하려면 보정 필요해 음수를 곱합
- LinearRegression을 이용해 보스턴 주택 가격 회귀 구현
    - 사이킷런은 보스턴 주택 가격 데이터 세트를 load_boston()을 통해 제공
    - Seaborn의 regplot()함수는 X, Y 축 값의 산점도와 함께 선형 회귀 직선을 그려줌
    - Matplotlib.subplots()를 이용해 각 축마다 칼럼과 가격의 관계를 표현함
    - 다른 칼럼보다 RM(양 방향의 선형성 가장 큼)과 LSTAT의 PRICE 영향도(음 방향의 선형성이 가장 큼)가 가장 두드러지게 나타남.
    - train_test_split()을 이용해 학습과 테스트 데이터 세트를 분리해 학습과 예측 수행

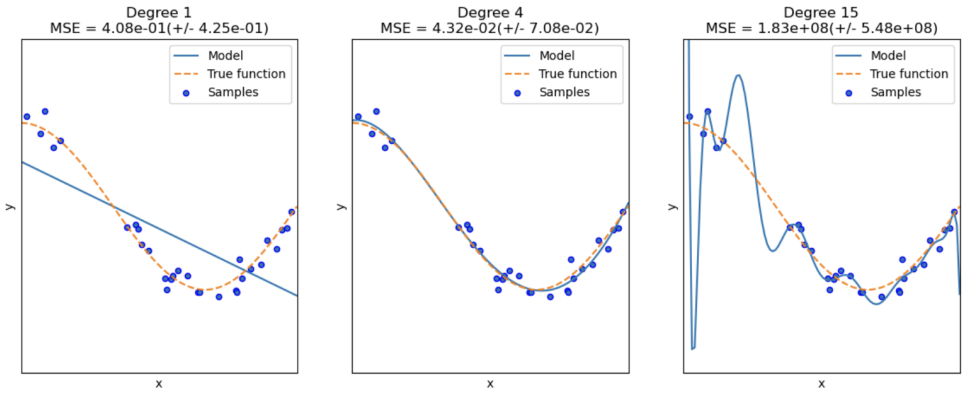
    - mean_squared_error()와 r2_score() API를 이용해 MSE와 R2 Score 측정
    - 선형회귀로 생성한 주택가격 모델의 절편과 회귀계수 값은 각각 intercept_속성, coef_속성에 저장됨
    - coef_속성을 피처별 회귀 계수 값으로 다시 매핑, 높은 값 순으로 출력 → Series sort_values() 함수 이용
    - 5개의 폴드 세트에서 cross_val_score()를 이용해 교차 검증 → MSE와 RMSE 측정
        - RMSE를 제공하지 않으므로 sqrt()를 직접 적용
        - scoring 수치 값을 음수 값으로 반환함

▼ 05. 다항 회귀와 과(대)적합/과소적합 이해

- 다항 회귀 : 독립변수의 단항식이 아닌 2차, 3차 방정식과 같은 다항식으로 표현되는 것
    - 예) y=w0+w1*x1+w2*x2+w3*x1*x2
- 다항 회귀는 선형 회귀 (회귀 계수가 선형임, 독립변수의 선형/비선형 여부와는 무관)
- 예측 성능 : 단순 선형 회귀 직선형으로 표현한 것 < 다항 회귀 곡선형으로 표현한 것
- 사이킷런은 다항 회귀를 위한 클래스를 명시적으로 제공하지 않음
    - 비선형 함수를 선형 모델에 적용시키는 방법을 사용해 구현함
    - PolynomialFeatures 클래스를 통해 피처를 다항식 피처로 변환
        - degree 파라미터를 통해 입력받은 단항식 피처를 degree에 해당하는 다항식 피처로 변환
        - fit, transform 메서드를 사용함
- 예) 3차 다항 회귀 함수를 임의로 설정하고 회귀 계수 예측
    - 결정함수식 : y=1+2x1+3x1^2+4X2^3
    - 함수 polynomial_func()를 만듦 → 3차 다항 계수 피처 값이 입력되면 결정 값 반환 → 선형 회귀 적용 → 다항 회귀로 구현됨
    - 피처 개수가 2개에서 10개로 늘어남. 회귀 계수도 10개로 늘어남.

- 사이킷런은 PolymonialFeatures로 피처를 변환한 후 LinearRegression 클래스로 다항 회귀 구현
- 사이킷런 Pipeline 객체를 이용해 피처 변환과 선형 회귀를 한 번에 구현하는 것이 더 명료함

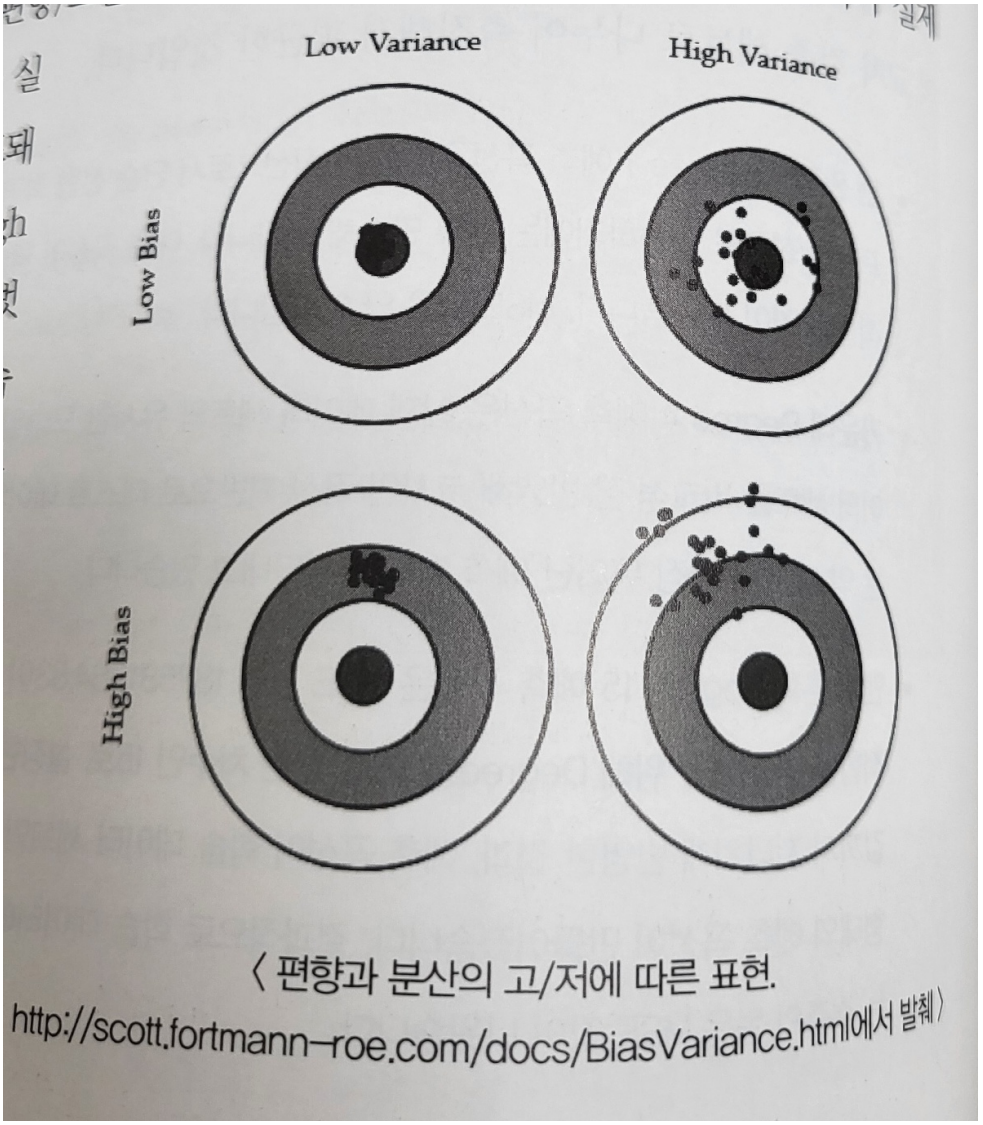
- 다항 회귀를 이용한 과소적합 및 과적합 이해
  - 다항 회귀의 차수를 높일수록 학습 데이터에만 너무 맞춘 학습 이뤄짐 → 테스트 데이터 환경에서 오히려 예측 정확도 떨어짐
  - ⇒ 차수가 높아질수록 과적합 문제 크게 발생



    - 실선 : 다항 회귀 예측 곡선 , 점선 : 실제 데이터 세트 X,Y의 곡선
    - 맨 왼쪽 Degree 1 예측 곡선 : 단순 선형 회귀와 동일 → 너무 단순함 → 예측 곡선이 학습 데이터의 패턴을 제대로 반영하지 못하는 과소적합 모델
    - Degree 4 예측 곡선 : 실제 데이터 세트와 유사, 학습 데이터 세트를 비교적 잘 반영해 테스트 데이터를 잘 예측함.
    - Degree 15 예측 곡선 : 데이터 세트의 변동 잡음값까지 지나치게 반영한 결과, 예측 곡선이 학습 데이터 세트만 정확히 예측하고 테스트 값의 실제 곡선과는 완전히 다른 형태의 예측 곡선이 만들어짐 → 과적합 심한 모델
    - 좋은 예측 모델 : 과소적합도 과적합도 아닌 균형잡힌 모델

- 편향-분산 트레이드오프 (bias-variance trade off)
  - 머신러닝이 극복해야 할 가장 중요한 이슈 중 하나
  - 매우 단순화된 모델 : 지나치게 한 방향성으로 치우침 → 고편향성을 가짐
  - 매우 복잡한 모델 : 지나치게 높은 변동성을 가짐 → 고분산성을 가짐

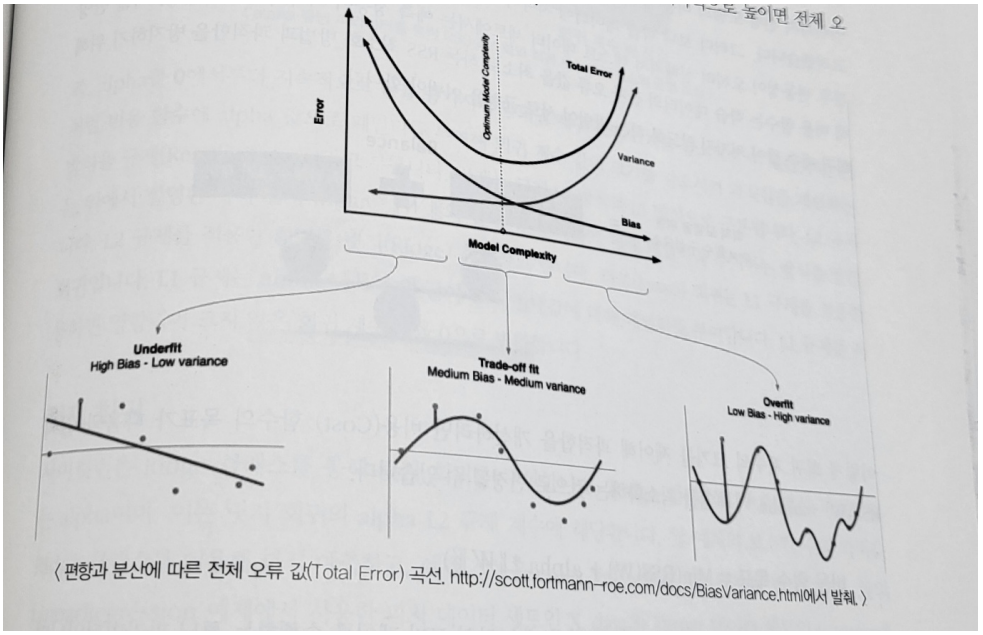
- 양궁 예시
  - 저편향/저분산 : 예측 결과가 실제 결과에 매우 근접, 아주 뛰어난 성능을 보여줌 (아주 드물게 좋은 경우)
  - 저편향/고분산 : 예측 결과가 실제 결과에 비교적 근접, 꽤 넓은 부분에 분포됨
  - 고편향/저분산 : 정확한 결과에서 벗어나면서도 예측이 특정 부분에 집중됨
  - 고편향/고분산 : 정확한 예측 결과를 벗어나면서도 넓은 부분에 분포됨



  - 일반적으로 편향과 분산은 한쪽이 높으면 한쪽이 낮아지는 경향이 있음
    - 편향이 높으면 분산 낮아짐(과소적합)

- 분산이 높으면 편향 낮아짐(과적합)
- 편향이 너무 높으면 전체 오류가 높음
- 편향을 점점 낮추면 분산이 높아지고 전체 오류 낮아짐
- 편향을 낮추고 분산을 높이면서 전체 오류가 가장 낮아지는 '골디락스' 지점을 통과하면서 분산을 지속적으로 높이면 → 전체 오류 값이 오히려 증가하면서 예측 성능이 다시 저하됨

  - 편향과 분산이 서로 트레이드오프를 이루면서 오류 Cost 값이 최대로 낮아지는 모델을 구축하는 것이 가장 효과적



〈편향과 분산에 따른 전체 오류 값(Total Error) 곡선. http://scott.fortmann-roe.com/docs/BiasVariance.html에서 발췌.〉

▼ 06. 규제 선형 모델 - 릿지, 라쏘, 엘라스틱넷

- 좋은 머신러닝 회귀 모델의 특징
  - 적절히 데이터에 적합하면서도 회귀 계수가 기하급수적으로 커지는 것을 제어할 수 있어야 함
  - 비용 함수는 학습 데이터의 잔차 오류를 최소로 하는 RSS 최소화 방법과 과적합을 방지하기 위해 회귀 계수 값이 커지지 않도록 하는 방법이 서로 균형을 이뤄야 함.
  - 비용 함수의 목표 = Min(RSS(W) + alpha*||W||22)
    - alpha : 학습 데이터 적합 정도와 회귀 계수 값의 크기 제어를 수행하는 튜닝 파라미터
    - alpha가 0 또는 매우 작은 값이라면 비용 함수 식은 기존과 동일, 무한대 또는 큰 값이라면 W 값을 0 또는 매우 작게 만들어야 cost가 최소

화되는 비용 함수 목ㅍ표 달성

        - ⇒ alpha 값을 크게 하면 비용 함수는 회귀 계수 값을 작게 해 과적합을 개선할 수 있음
        - ⇒ alpha 값을 작게 하면 회귀 계수 값이 커져도 어느 정도 상쇄 가능 → 학습 데이터 적합 개선 가능
        - alpha를 0에서부터 지속적으로 값을 증가시켜 회귀 계수 값의 크기를 감소시킬 수 있음
    - 규제 : 비용 함수에 alpha 값으로 페널티를 부여해 회귀 계수 값의 크기를 감소시켜 과적합을 개선하는 방식
        - L1와 L2 방식으로 구분됨
            - L2 규제 : W의 제곱에 대해 패널티를 부여하는 방식 → 이를 적용한 회귀를 '릿지 회귀'라고 함
            - L1 규제 : W의 절댓값에 대해 패널티를 부여 → 이를 적용한 회귀를 '라쏘 회귀'라고 함
- 릿지 회귀
    - 사이킷런 Ridge 클래스로 구현
        - 주요 생성 파라미터 : alpha = alpha L2 규제 계수
        - 보스턴 예제
            - 앞의 규제가 없는 선형 회귀보다 릿지는 더 뛰어난 예측 성능을 보여줌
            - 알파 값이 커질수록 회귀 계수 값을 작게 만듦
- 라쏘 회귀
    - 사이킷런 Lasso 클래스로 구현
        - 주요 생성 파라미터 : alpha = alpha L1 규제 계수
    - W의 절댓값에 패널티 부여
    - 비용함수의 목표는 식을 최소화하는 W를 찾는 것
    - 불필요한 회귀 계수를 급격하게 감소시켜 0으로 만들고 제거함 (=피처 선택 특성)
        - 보스턴 예제
            - 릿지보다는 떨어지는 수치지만 선형 회귀보다는 성능 향상됨

- 엘라스틱넷 회귀
  - L2 규제와 L1 규제를 결합한 회귀
  - 엘라스틱넷 회귀 비용함수의 목표 : RSS(W)+alpha2*||W||22 + alpha1*||W||1 식을 최소화하는 W를 찾는 것
  - 라쏘 회귀가 서로 상관관계가 높은 피처들의 경우에 이들 중에서 중요 피처만을 셀렉션하고 다른 피처들은 모두 회귀 계수를 0으로 만드는 성향이 강함
    - 이런 성향으로 alpha 값에 따라 회귀 계수 값이 급격히 변동할 수 있기에 L2 규제를 라쏘 회귀에 추가한 것
  - 단점 : L1과 L2 규제가 결합된 규제로 인해 수행 시간이 상대적으로 오래 걸림
  - 사이킷런은 ElasticNet 클래스를 통해 회귀 구현
    - 주요 생성 파라미터 : alpha(a+b), l1_ratio(a/(a+b))
    - 엘라스틱넷의 규제 : a_*L1+b*L2 → a는 L1의 alpha값, b는 L2의 alpha값
    - l1_ratio가 1이면 L1 규제와 동일, 0이면 L2 규제와 동일
  - alpha값에 따른 피처들의 회귀 계수들 값이 라쏘보다는 상대적으로 0이 되는 값이 적음
- 선형 회귀 모델을 위한 데이터 변환
  - 일반적으로 피처와 타겟값 간에 선형 관계가 있다고 가정하고 이러한 최적의 선형 함수를 찾아내 결과값을 예측함
  - 피처값과 타겟값의 분포가 정규 분포인 형태를 매우 선호함
    - 타겟값이 분포가 치우친 왜곡된 형태의 분포도일 경우 예측 성능에 부정적인 영향을 미칠 가능성이 높음 → 선형 회귀 모델 적용 전 데이터 스케일링/정규화 작업 수행하는 것이 일반적
  - 사이킷런을 이용해 피처 데이터 세트에 적용하는 변환 작업 방법
    1. StandardScaler 클래스를 이용해 평균이 0, 분산이 1인 표준 정규 분포를 가진 데이터 세트로 변환 or MinMaxScaler 클래스를 이용해 최솟값이 0이고 최댓값이 1인 값으로 정규화 수행

       (-) 예측 성능 향상을 크게 기대하기 어려운 경우가 많음

2. 스케일링/정규화 수행한 데이터 세트에 다시 다항 특성을 적용하여 변환 (보통 1번 방법을 통해 예측 성능 향상이 없을 경우 이 방법 적용)

   (-) 피처의 개수가 매우 많을 경우에는 다항 변환으로 생성되는 피처 개수가 기하급수로 늘어나 과적합 이슈가 발생할 수 있음

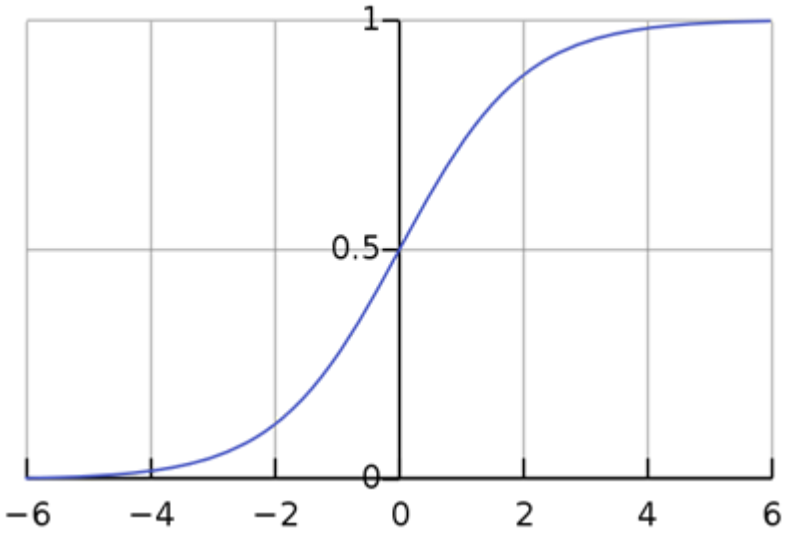
3. 원래 값에 log 함수를 적용해 정규 분포에 가까운 형태로 값을 분포시킴(=로그 변환)

   a. 1,2,번 방법보다 훨씬 많이 사용되는 변환 방법

- 타깃값 변환 방법 : 일반적으로 로그 변환 적용
- 보스턴 예제에 1,2,3 방법을 순차적으로 적용해보고 각 경우 별 예측 성능 측정
  - get_scaled_data() 함수 : method 인자로 변환 방법을 결정하며 1,3 방법 중 하나를 선택함.
  - 결과를 보면 표준 정규 분포와 최솟값/최댓값 정규화로 피처 데이터 세트를 변경해도 성능상 개선은 없음
  - 1차 변환 후 2차 다항식 변환을 했을 때 alpha값이 개선되어 성능이 개선되지만 피처의 개수가 많을 경우 적용하기 힘듦. 데이터 건수가 많아지면 계산 오래 걸리는 한계도 있음.
  - 로그 변환은 대개 좋은 성능 향상이 있음. → 데이터 값의 분포가 심하게 왜곡되어 있을 경우 로그 변환 적용이 좋음

▼ 07. 로지스틱 회귀

- 선형 회귀 방식을 분류에 적용한 알고리즘 (이진 분류 예측 성능 뛰어남)
- 선형 회귀 계열
- 회귀가 선형인가 비선형인가? → 가중치 변수가 선형인가 아닌가에 따름
- Sigmoid(시그모이드) 함수 최적선을 찾고 이 시그모이드 함수의 반환 값을 확률로 간주해 확률에 따라 분류를 결정함
  - 시그모이드 함수는 S자 커브 형태를 가짐

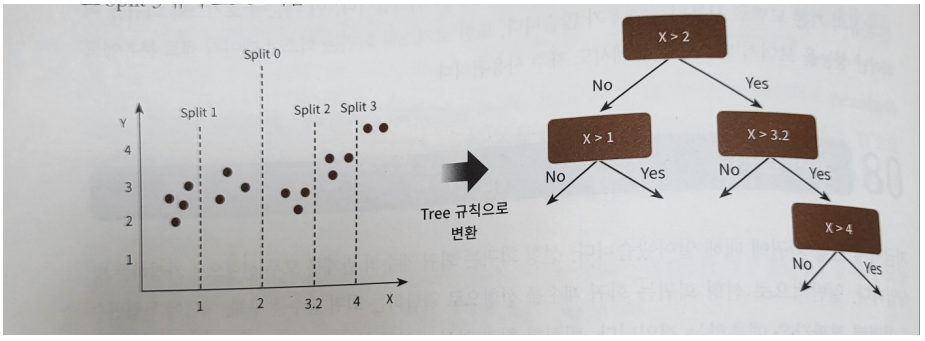


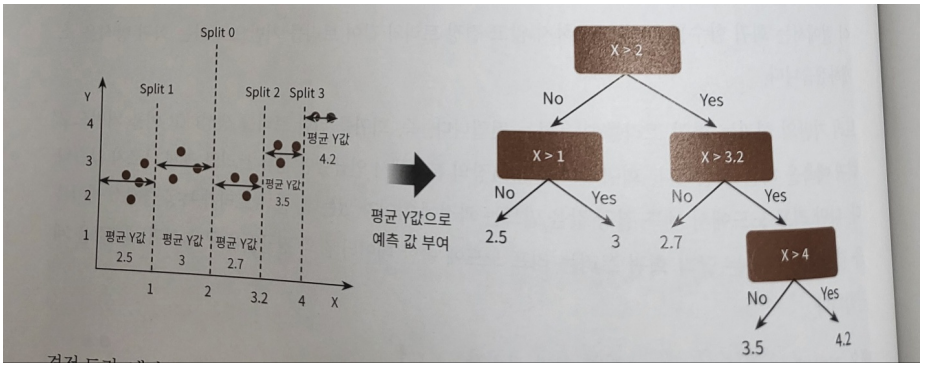
    - x값이 +,-로 아무리 커지거나 작아져도 y 값은 항상 0과 1 사이 값을 반환함
    - x값이 커지면 1에 근사, x값이 작아지면 0에 근사. x값이 0이면 0.5
- 예) 악성 종양인지, 그렇지 않은지 1과 0의 값으로 예측
    - 종양 크기를 X축, 악성 종양 여부를 Y축에 표시할 수 있음 → 선형 회귀 선을 그릴 수 있지만 이 경우 0과 1을 제대로 분류하지 못함
    - S자 커브 형태의 시그모이드 함수를 이용하면 좀 더 정확하게 0과 1을 분류할 수 있음
- 사이킷런은 로지스틱 회귀를 위해 LogisticRegression 클래스를 제공
    - 회귀 계수 최적화 : 경사 하강법 등 다양한 최적화 방안 선택
    - solver 파라미터의 lbfgs, lib linear, newton-cg, sag, saga 값을 적용해 최적화 선택 가능
        - 다양한 solver 값이 있지만 성능 차이는 미비함. 일반적으로 lbfgs 또는 liblinear 선택
- 로지스틱 회귀를 사용해 위스콘신 유방암 데이터 세트로 암 여부 판단
    - 데이터에 먼저 정규 분포 형태의 표준 스케일링을 적용한 후 train_test_split()으로 데이터 세트 분리
    - 기본 solver 값은 lbfgs 이므로 인자값을 입력하지 않으면 자동으로 lbfgs가 할당됨
    - max_iter로 solver로 지정된 최적화 알고리즘이 최적 수렴할 수 있는 최대 반복 횟수 지정 가능

- 여러 데이터 세트에 적용해봐도 solver별 차이는 크지 않음
- penalty : 규제 유형 설정하는 파라미터 / C : 규제 강도를 조절하는 alpha 값의 역수 (1/alpha)
  - liblinear, saga : L1, L2 규제가 모두 가능 / lbfgs, newton-cg, sag : L2 규제만 가능

▼ 08. 회귀 트리

- 트리 기반의 회귀 : 회귀 트리를 이용함
  - 회귀를 위한 트리를 생성하고 이를 기반으로 회귀 예측을 하는 것
  - 회귀 트리는 분류 트리와 크게 다르지 않음
    - but 회귀 트리는 리프 노드에 속한 데이터 값의 평균값을 구해 회계 예측값을 계산함
    - 다음 그림과 같이 리프 노드 생성 기준에 부합하는 트리 분할이 완료되었다면 리프 노드에 소속된 데이터 값의 평균값을 구해 최종적으로 리프 노드에 결정 값으로 할당함





- 사이킷런 랜덤 포레스트 회귀인 RandomForestRegressor를 이용해 보스턴 주택 가격 예측 수행

- 회귀 트리 Regressor 클래스는 선형 회귀와 다른 처리 방식이므로 coef_ 속성이 없음
  - 대신 feature_importances_를 이용해 피처별 중요도를 알 수 있음
- 회귀 트리의 경우 분할되는 데이터 지점에 따라 브랜치를 만들면서 계단 형태로 회귀선을 만듦