5. 회귀

01. 회귀 소개

회귀: 여러 개의 독립변수와 한 개의 종속변수 간의 상관관계를 모델링하는 기법
ex) 아파트의 방 개수, 방 크기, 주변 학군 등 여러 개의 독립변수에 따라 아파트 가격이라는 종속변수가 어떤 관계를 나타내는지를 모델링하고 예측하는 것
Y = W1*X1* + *W2*X2 + W3*X3*+ ... + *Wn*Xn이라는 선형회귀식을 예로 들면,
Y: 종속변수, 즉 아파트 가격
X1,X2,...,Xn은 방 개수, 방 크기, 주변 학군 등 독립 변수
W1, W2, W3,...,W3: 이 독립변수의 값에 영향을 미치는 회귀 계수(Regression coefficients)
머신러닝의 관점) 독립변수는 피처, 종속변수는 결정 값

머신러닝 회귀 예측의 핵심: 주어진 피처와 결정 값 데이터 기반에서 학습을 통해 *최적의 회귀 계수*를 찾아내는 것

회귀의 유형
독립변수 개수 1개 : 단일 회귀
독립변수 개수 여러 개 : 다중 회귀

회귀 계수의 결합이 선형: 선형 회귀
회귀 계수의 결합이 비선형: 비선형 회귀

지도학습의 두 가지 유형인 *분류*와 *회귀*의 가장 큰 차이
-> 분류는 예측값이 이산형 클래스 값이고 *회귀는 연속형 숫자 값*

대표적인 선형 회귀 모델

- 일반 선형 회귀: 예측값과 실제 값의 RSS(Residual Sum of Squares)# 최소화할 수 있도록 회귀 계수를 최적화하며, 규제(Regularization)를 적용하지 않은 모델입니다
- 릿지(Ridge) : 릿지 회귀는 선형 회귀에 L2 규제를 추가한 회귀 모델입니다. 릿지 회귀는 L2 규제를 적용하는데, L2 규제는 상대적으로 큰 회귀 계수 값의 예측 영향도를 감소시키기 위해서 회귀 계수값을 더 작게 만드는 규제 모델입니다.
- 라쏘(Lasso) : 라쏘 회귀는 선형 회귀에 L1 규제를 적용한 방식입니다. L2 규제가 회귀 계수 값의 크기를 줄이는 데 반해, L1 규제는 예측 영향력이 작은 피처의 회귀 계수를 0으로 만들어 회귀 예측 시 피처가 선택되지 않게 하는 것입니다. 이러한 특성 때문에 L1 규제는 피처 선택 기능으로도 불립니다.
- 엘라스틱넷(ElasticNet) : L2, L1 규제를 함께 결합한 모델입니다. 주로 피처가 많은 데이터 세트에서 적용되며, L1 규제로 피처의 개수를 줄임과 동시에 L2 규제로 계수 값의 크기를 조정합니다.
- 로지스틱 회귀(Logistic Regression) : 로지스틱 회귀는 회귀라는 이름이 붙어 있지만, 사실은 분류에 사용되는 선형 모델입니다. 로지스틱 회귀는 매우 강력한 분류 알고리즘입니다. 일반적으로 이진 분류뿐만 아니라 희소 영역의 분류, 예를 들 어 텍스트 분류와 같은 영역에서 뛰어난 예측 성능을 보입니다.

02. 단순 선형 회귀를 통한 회귀 이해

단순 선형 회귀는 독립변수도 하나, 종속변수도 하나
ex) 주택 가격이 주택의 크기로만 결정된다고 하자.

일반적으로 주택의 크기가 크면 가격이 높아지는 경향이 있기 때문에 주택 가격은주택 크기에 대해 선형(직선 형태)의 관계로 표현할 수 있음

Ŷ = w0 + w1 * X
-> 기울기 w1, 절편intercept w0이 회귀 계수

실제 주택 가격 = w0 + w1 * X + 오류 값(잔차)

최적의 회귀 모델: 전체 데이터의 잔차 (오류 값) 합이 최소가 되는 모델. 오류 값 합이 최소가 될 수 있는 최적의 회귀 계수를 가짐

오류 합 계산

1. 절댓값을 취해서 더하기(Mean Absolute Error)
2. 오류 값의 제곱을 구해서 더하는 방식(RSS, Residual Sum of Square) -> Error^2 = RSS

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

RSS: 비용
w 변수(회귀 계수)로 구성되는 RSS: 비용 함수

머신 러닝 회귀 알고리즘은 데이터를 계속 학습하면서 이 비용 함수가 반환하는 값 (즉, 오류 값) 을 지속해서 감소시키고 최종적으로는 더 이상 감소하지 않는 최소의 오류 값을 구하는 것

03 비용 최소화하기 - 경사 하강법(Gradient Descendent) 소개

경사 하강법 : '점진적으로' 반복적인 계산을 통해 w 파라미터 값을 업데이트하면서 오류 값이 최소가 되는 W 파라미터를 구하는 방식

ex) 비용 함수가 아래로 볼록인 포물선 형태의 2차 함수
경사 하강법은 최초 w에서부터 미분을 적용한 뒤 이 미분 값이 계속 감소하는 방향으로 순차적으로 w를 업데이트함
마침내 더 이상 미분된 1차 함수의 기울기가 감소하지 않는 지점을 비용 함수가 최소인 지점으로 간주하고 그 때의 w를 반환

∂R(w)/∂w1 = 2/N * Σ_(i=1 to N) -xi * (yi - (w0 + w1xi)) = -2/N * Σ_(i=1 to N) x1 * (실제값i - 예측값i)
∂R(w)/∂w0 = 2/N * Σ_(i=1 to N) -(yi - (w0 + w1xi)) = -2/N * Σ_(i=1 to N) (실제값i - 예측값i)

<경사하강법의 프로세스>
Step 1. w1,w2를 임의의 값으로 설정하고 첫 비용 함수의 값을 계산합니다.
Step 2. w1을 w1 + η * 2/N * Σ_(i=1 to N) x1 * (실제값i - 예측값i).
w0 을 w0 + η * 2/N * Σ_(i=1 to N) x1 * (실제값i - 예측값i)으로 업데이트한 후 다시 비용 함수의 값을 계산합니다.
Step 3. 비용 함수가 감소하는 방향성으로 주어진 횟수만큼 Step 2를 반복하면서 w1과 w0을 계속 업데이트합니다

<경사하강법의 단점>
경사 하강법은 모든 학습 데이터에 대해 반복적으로 비용함수 최소화를 위한 값을 업데이트하기 때문에 수행 시간이 매우 오래 걸림
-> (미니 배치)확률적 경사하강법(Stochastic Gradient Descent)를 이용

-> 전체 입력 데이터로 w가 업데이트되는 값을 계산하는 것이 아니라 일부 데이터만 이용해 피가 업데이트되는 값을 계산하므로 경사 하강법에 비해서 빠른 속도를 보장 [

피처가 여러 개인 경우에 회귀 계수 도출하기

피처가 1개인 경우를 확장해 유사하게 도출
피처가 M개(X1,X2, ... ,X100) 있다면 그에 따른 회귀 계수도 M+1(1개는 w0)개로 도출됨
Ŷ + w0 + w1 * X1 + w2 * X2 + ... + w100 * X100

데이터의 개수가 N이고 피처 M개일 때 예측행렬 Ŷ
:

[[y1],
[y2],
...,
[yn]]

[[1 x11 x12 ... x1m],
[1 x21 x22 ... x2m],
...,
[1 xn1 xn2 ... xnm]] · [w0 w1 w2 ... wm].T

04 사이킷런 LinearRegression을 이용한 보스턴 주택 가격 예측

LinearRegression 클래스:

- 예측값과 실제 값의 RSS(Residual Sum of Squares)를 최소화해 OLS(Ordinary Least Squares) 추정 방식으로 구현한 클래스
- fit() 메서드로 X, y 배열을 입력받으면 회귀 계수(Coefficients)인 W를 coef_속성에 저장

다중 공선성(multi-collinearity) 문제
: Ordinary Least Squares 기반의 회귀 계수 계산은 입력 피처의 독립성에 많은 영향을 받음
피처 간의 상관관계가 매우 높은 경우 분산이 매우 커져서 오류에 매우 민감해짐

-> 해결방안1. 상관관계가 높은 피처가 많은 경우 독립적인 중요한 피처만 남기고 제거하거나 규제를 적용
-> 해결방안2. 매우 많은 피처가 다중 공선성 문제를 가지고 있다면 PCA를 통해 차원 축소를 수행

회귀 평가 지표

: 회귀의 평가를 위한 지표는 실제 값과 회귀 예측값의 차이 값을 기반으로 한 지표가 중심
오류의 절댓값 평균이나 제곱, 또는 제곱한 뒤 다시 루트를 씌운 평균값을 구함

- MAE: Mean Absolute Error(MAE)이며 실제 값과 예측값의 차이를 절댓값으로 변환해 평균한 것 API metrics.mean_absolute_error
- MSE: Mean Squared Error(MSE)이며 실제 값과 예측값의 차이를 제곱해 평균한 것 API metrics.mean_squared_error
- RMSE: MSE 값은 오류의 제곱을 구하0로 실제 오류 평균보다 더 커지는 특성이 있으므로 MSE에 루트를 씌운 것 API metrics.mean_squared_error를 그대로 사용하되 squared 파라미터를 False로 설정

- R^2: 분산 기반으로 예측 성능을 평가합니다. 실제 값의 분산 대비 예측값의 분산 비율을 지표로 하며, 1에 가까울수록 예측 정확도가 높음 API metrics.r2_score

05 다중 회귀와 과(대)적합/과소적합 이해

다항 회귀 이해

회귀가 독립변수의 단항식이 아닌 2차,3차 방정식과 같은 다항식으로 표현되는 것을 다항(Polynomail) 회귀라고 함
다항회귀 표현
: y = w0 + w1 * x1 + w2 * x2 + w3 * x1 * x2 + w4 * x1^2 + w5 * x2^2 과 같음

다항 회귀도 *선형 회귀*임
선형 회귀/비선형 회귀를 나누는 기준은 회귀 계수가 선형/비선형인지에 따른 것이지 독립변수의 선형/비선형 여부와는 무관

Z = [x1,x2,x1*x2,x1^2,x2^2] 로 한다면 y = w0 + w1*z1 + w2*z2 +... 로 표현할 수 있기에 여전히 선형 회귀

사이킷런은 다항 회귀를 위한 클래스를 명시적으로 제공하지 않음
-> 대신 비선형 함수를 선형 모델에 적용시키는 방법을 사용해 구현
-> 사이킷런은 PolynomialFeatures 클래스를 통해 피처를 Polynomial(다항식) 피처로 변환
PolynomialFeatures 클래스는 degree 파라미터를 통해 입력받은 단항식 피처를 degree에 해당하는 다항식 피처로 변환
다른 전처리 변환클래스와 마찬가지로 PolynomialFeatures 클래스는 fit(), transform() 메서드를 통해 이 같은 변환 작업을 수행

다항 회귀를 이용한 과소적합 및 과적합 이해 다항 회귀:
피처의 직선적 관계가 아닌 복잡한 다항 관계를 모델링할 수 있음
다항식의 차수가 높아질수록 매우 복잡한 피처 간의 관계까지 모델링이 가능
과적합 문제 발생할 수 있음(다항 회귀의 차수(degree)를 높일수록 학습 데이터에만 너무 맞춘 학습이 이뤄져서 정작 테스트 데이터 환경에서는 오히려 예측 정확도가 떨어짐)

편향-분산 트레이드오프(Bias-Variance Trade off)

편향-분산 트레이드오프는 머신러닝이 극복해야 할 가장 중요한 이슈 중의 하나
매우 단순화된 모델로서 지나치게 한 방향성으로 치우친 경향이 있는 모델은 *고편향(High Bias)성*을 가졌다고 표현
학습 데이터 하나 하나의 특성을 반영하면서 매우 복잡한 모델이 되었고 지나치게 높은 변동성을 가지는 모델을 *고분산(High Variance)성*을 가졌다고 표현

저편향/저분산 - 결과가 실제 결과에 매우 잘 근접하면서도 예측 변동이 크지 않고 특정 부분에 집중돼 있는 아주 뛰어난 성능을 보여줌
저편향/고분산 - 예측 결과가 실제 결과에 비교적 근접하지만, 예측 결과가 실제 결과를 중심으로 꽤 넓은 부분에 분포되어있음
고편향/저분산 - 정확한 결과에서 벗어나면서도 예측이 특정 부분에 집중
고편향/저분산 - 정확한 예측 결과를 벗어나면서도 넓은 부분에 분포
-> 일반적으로 편향과 분산은 한쪽이 높으면 한쪽이 낮아지는 경향이 있음
-> 편향을 낮추고 분산을 높이면서 전체 오류가 가장 낮아지는 '골디락스' 지점을 통과하면서 분산을 지속적으로 높이면 전체 오류 값이 오히려 증가하면서 예측 성능이 다시 저하

06 규제 선형 모델 - 릿지, 라쏘, 엘라스틱넷

규제 선형 모델의 개요
회귀 모델은 적절히 데이터에 적합하면서도 회귀 계수가 기하급수적으로 커지는 것을 제어할 수 있어야 함

비용 함수는 학습 데이터의 잔차 오류 값을 최소로 하는 RSS 최소화 방법과 과적합을 방지하기 위해 회귀 계수 값이 커지지 않도록 하는 방법이 서로 균형을 이뤄야 함

비용 함수 목표 = Min(RSS(W)) ＋ alpha * (W의 L2norm)^2

alpha : 학습 데이터 적합 정도와 회귀 계수 값의 크기 제어를 수행하는 튜닝 파라미터
alpha값을 크게 하면 비용 함수는 회귀 계수 W의 값을 작게 해 과적합을 개선 할 수 있으며 alpha 값을 작게 하면 회귀 계수 W의 값이 커져도 어느 정도 상쇄가 가능하므로 학습 데이터 적합을 더 개선할 수 있음

규제(Regularization): 비용 함수에 alpha 값으로 페널티를 부여해 회귀 계수 값의 크기를 감소시켜 과적합을 개선하는 방식

- L2 규제: alpha * (W의 L2norm)*2 사용
  -> L2 규제를 적용한 회귀를 *릿지(Ridge) 회귀*라고 함
- L1규제: alpha * (W의 L1norm)*2 사용
  -> L2 규제를 선형 회귀에 적용한 것이 라쏘(Lasso)회귀 **릿지 회귀**
  사이킷런은 Ridge 클래스를 통해 릿지 회귀를 구현.
  Ridge 클래스의 주요 생성 파라미터는 alpha이며, 이는 릿지 회귀의 alpha L2 규제 계수에 해당

라쏘 회귀
L2 규제를 선형 회귀에 적용한 것이 라쏘(Lasso)회귀
L2 규제가 회귀 계수의 크기를 감소시키는 데 반해, L1 규제는 불필요한 회귀 계수를 급격하게 감소시켜 0으로 만들고 제거
이러한 측면에서 L1 규제는 적절한 피처만 회귀에 포함시키는 피처 선택의 특성을 가지고 있음

사이킷런은 Lasso 클래스를 통해 라쏘 회귀를 구현.
Lasso 클래스의 주요 생성 파라미터는 alpha이며, 이는 라쏘 회귀의 alpha L1 규제 계수에 해당

엘라스틱넷 회귀

엘라스틱넷(Elastic Net) 회귀: L2 규제와 L1 규제를 결합한 회귀
엘라스틱넷 회귀 비용함수의 목표: RSS(W) ＋ alpha * (W의 L2norm)^2 + RSS(W) ＋ alpha * (W의 L1norm)^2 식을 최소화하는 W를 찾는 것

사이킷런은 ElasticNet 클래스를 통해서 엘라스틱넷 회귀를 구현
ElasticNet 클래스의 주요 생성 파라미터는 alpha와 ll_ratio
-alpha: ElasticNet 클래스의 alpha는 Ridge와 Lasso 클래스의 alpha값과는 다름.
엘라스틱넷의 규제는 a * LI + b * L2로 정의될 수 있으며, 이때 a는 L1 규제의 alpha값, b는 L2 규제의 alpha 값
따라서 ElasticNet 클래스의 alpha 파라미터 값은 a + b

- ll_ratio 파라미터 값: a / (a ＋ b) ll_ratio가 0이면 a가 0이므로 L2 규제와 동일 ll_ratio가 1이면 b가 0이므로 L1 규제와 동일

선형 회귀 모델을 위한 데이터 변환
선형 회귀 모델은 피처값과 타깃값의 분포가 정규 분포(즉 평균을 중심으로 종 모양으로 데이터 값이 분포된 형태) 형태를 매우 선호

왜곡(Skew)된 형태의 분포도일 경우 예측 성능에 부정적인 영향을 미칠 가능성이 높음 (타깃값이 피처값보다 예측성능에 부정적 영향을 더 미침)
=> 따라서, 선형 회귀 모델을 적용하기 전에 먼저 데이터에 대한 스케일링/정규화 작업을 수행

<사이킷런을 이용해 *피처 데이터 세트*에 적용하는 변환 작업>

1. StandardScaler 클래스를 이용해 평균이 0, 분산이 1인 표준 정규 분포를 가진 데이터 세트로 변환하거나 MinMaxScaler 클래스를 이용해 최솟값이 0이고 최댓값이 1인 값으로 정규화를 수행합니다.
2. 스케일링/정규화를 수행한 데이터 세트에 다시 다항 특성을 적용하여 변환하는 방법입니다. 보통 1번 방법을 통해 예측 성능에 향상이 없을 경우 이와 같은 방법을 적용합니다.
3. 원래 값에 log 함수를 적용하면 보다 정규 분포에 가까운 형태로 값이 분포됩니다. 이러한 변환을 로그 변환(Log Transformation)이라고 부릅니다. 로그 변환은 매우 유용한 변환이며, 실제로 선형 회귀에서는 앞에서 소개한 1. 2 번 방법보다 로그 변환이 훨씬 많이 사용되는 변환 방법입니다. 왜냐하면 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 때문입니다

<*타깃값*의 경우>
일반적으로 로그 변환 적용

07 로지스틱 회귀

로지스틱 회귀: 선형 회귀 방식을 분류에 적용한 알고리즘

로지스틱 회귀와 선형 회귀와 다른 점:
회귀가 선형인가 비선형인가는독립변수가 아닌 가중치(weight) 변수가 선형인지 아닌지를 따름
로지스틱 회귀은 학습을 통해 선형 함수의 회귀 최적선을 찾는 것이 아니라 시그모이드(Sigmoid) 함수 최적선을 찾고 이 시그모이드 함수의 반환 값을 확률로 간주해 확률에 따라 분류를 결정함

시그모이드 함수의 정의:
y = 1/(1 + e^(-x))

- x 값이 十, -로 아무리 커지거나 작아져도 y 값은 항상 0과 1 사이 값을 반환. x 값이 커지면 1에 근사하며 x 값이 작아지면 0에 근사
- x=0일 때 0.5

지금까지는 연속형 값을 구하는데 회귀를 사용
이번에는 분류 문제에 적용(종양의 크기에 따라 악성 종양인지(Yes = 1) 그렇지 않은지(No=0)를 회귀를 이용해 1과 0의 값으로 예측)

사이킷런은 로지스틱 회귀를 위해서 LogisticRegression 클래스를 제공

- lbfgs: 사이킷런 버전 0.22부터 solver의 기본 설정값입니다. 메모리 공간을 절약할 수 있고, CPU 코어 수가 많다면 최적화를 병렬로 수행할 수 있습니다.
- liblinear: 사이킷런 버전 0.21 까지에서 solver의 기본 설정값입니다. 다차원이고 작은 데이터 세트에서 효과적으로 동작하지만 국소 최적화(Local Minimum)에 이슈가 있고. 병렬로 최적화할 수 없습니다.
- newton-cg: 좀 더 정교한 최적화를 가능하게 하지만, 대용량의 데이터에서 속도가 많이 느려집니다.
- sag: Stochastic Average Gradient로서 경사 하강법 기반의 최적화를 적용합니다. 대용량의 데이터에서 빠르게 최적화합니다.
- saga: sag와 유사한 최적화 방식이며 L1 정규화를 가능하게 해줍니다.

08 회귀 트리

머신러닝 기반의 회귀는 *회귀 계수를 기반으로 하는 최적 회귀 함수*를 도출하는 것이 주요 목표

이 절에서는 회귀 함수를 기반으로 하지 않고 결정 트리와 같이 트리를 기반으로 하는 회귀 방식에 대하여

트리 기반의 회귀: 회귀 트리를 이용하는 것. 회귀를 위한 트리를 생성하고 이를 기반으로 회귀 예측을 하는 것

분류 트리와의 차이점:
리프 노드에서 예측 결정 값을 만드는 과정에서, 분류 트리가 특정 클래스 레이블을 결정하는 것과는 달리 회귀 트리는 *리프 노드에 속한 데이터 값의 평균값을 구해 회귀 예측값을 계산*합니다

결정 트리, 랜덤 포레스트, GBM, XGBoost, LightGBM 등의 앞 4장의 분류에서 소개한 모든 트리 기반의 알고리즘은 분류뿐만 아니라 회귀도 가능
-> 트리 생성이 CART(T (Classification And Regression Trees) 알고리즘에 기반하고 있기 때문

- 사이킷런에서는 결정 트리, 랜덤 포레스트, GBM에서 CART 기반의 회귀 수행을 할 수 있는 Estimator 클래스를 제공
- XGBoost(->XGBRegressor), LightGBM(->LGBMRegressor)도 사이킷런 래퍼 클래스를 통해 제공
-