Week 6_예습과제_김정은

회귀

회귀란?

- 여러 개의 독립변수와 한 개의 종속 변수 간의 상관관계를 모델링하는 기법을 통칭
- 머신러닝의 관점에서 **독립 변수는 피처에 해당, 종속 변수는 결정 값**.
- 머신러닝 회귀 예측의 핵심은 주어진 **피처와 결정 값 데이터 기반에서 학습을 통해 최적의 회귀 계수를 찾아내는 것.**
- 가장 중요한 것은 **회귀 계수**
  - 회귀 계수가 선형이냐 아니냐에 따라 **선형 회귀와 비선형 회귀**로 나뉨
  - 독립 변수의 개수가 한 개인지 여러 개인지에 따라 **단일 회귀, 다중 회귀**로 나뉨
- 지도 학습은 두 가지 유형 → 분류, 그리고 회귀.
  - 가장 큰 차이 → 분류는 예측값이 카테고리와 같은 이산형 클래스 값
  - 회귀는 연속형 숫자값
- 여러 회귀 방법들 중에서 선형 회귀가 가장 많이 사용됨
  - 실제 값과 예측값의 차이(오류의 제곱 값)를 최소화하는 직선형 회귀선을 최적화하는 방식
  - 규제(Regularization) 방법에 따라 다시 나눌 수 있음
    - 규제 → 일반적인 선형 회귀의 과적합 문제를 해결하기 위해 회귀 계수에 페널티 값을 적용
  - 대표적인 선형 회귀 모델 → 일반 선형 회귀, 릿지, 라쏘, 엘라스틱넷, 로지스틱 회귀

단순 선형 회귀

- 독립 변수도 하나, 종속 변수도 하나.

- 실제 값과 회귀 모델의 차이에 따른 오류 값 → 남은 오류, 잔차 라고 부름.
- 최적의 회귀 모델을 만든다는 것 → 전체 데이터의 잔차(오류 값) 합이 최소가 되는 모델을 만들겠다. = 오류 값 합이 최소가 될 수 있는 최적의 회귀 계수를 찾는다.
- 오류값(잔차) → +, - 둘 다 될 수 있음
    - 따라서 전체 데이터의 오류 합을 구하기 위해 단순히 더했다가는? 오류 합이 줄어들 수가 있음 → 오류합 계산 시엔 다른 방식 취함
        - 절댓값을 취해서 더하거나,(Mean Absolute Error)
        - 오류 값의 제곱을 구해서 더하는 방식(RSS, Residual Sum of Square)
        - 일반적으로는 미분 등 계산을 편하게 하기 위해 RSS 방식 채용
            - RSS는 변수가 w_1, w_2 인 식으로 표현할 수 있으며, 이 RSS를 최소로 하는 w_1, w_2, 회귀 계수를 학습을 통해서 찾는 것이 핵심.
            - 회귀에서 RSS는 비용(Cost)
            - w 변수(회귀 계수)로 구성되는 RSS → 비용 함수
            - 회귀 알고리즘은 데이터를 계속 학습하며 이 비용 함수가 반환하는 값(오류 값)을 지속해서 감소시키고, 최종적으로는 더 이상 감소하지 않는 최소의 오류값을 구함

비용 최소화 - 경사 하강법(Gradient Descent)

- 경사 하강법 → 고차원 방정식에 대한 문제를 해결해주면서 비용 함수 RSS를 최소화하는 방법을 직관적으로 제공
- '점진적으로' 반복적인 계산을 통해 W 파라미터 값을 업데이트 → 오류 값이 최소가 되는 W 파라미터를 구하는 방식
- 예를 들어 비용 함수가 포물선 형태의 2차 함수라면, 경사 하강법은 최초 w에서부터 미분을 적용 → 이 미분 값이 계속 감소하는 방향으로 순차적으로 w 업데이트
    - 마침내 더 이상 미분된 1차 함수의 기울기가 감소하지 않는 지점을 비용함수가 최소인 지점으로 간주 → w 반환

사이킷런 LinearRegression 클래스 - Ordinary Least Squares

- 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 게수(Coefficients)인 W를 coef_ 속성에 저장

회귀 평가 지표

- 회귀의 평가를 위한 지표는 실제 값과 회귀 예측값의 차이 값이 중심
- 오류의 절댓값 평균이나 제곱, 제곱한 뒤 다시 루트를 씌운 평균값을 사용
- MAE, MSE, RMSE, R^2 사용

다항 회귀와 과(대)적합/과소적합 이해

다항 회귀 이해

- 세상의 모든 관계를 직선으로만 표현할 수는 없음
- 회귀가 독립변수의 단항식이 아닌 2차, 3차 방정식과 같은 다항식으로 표현 → 다항(Polynomial) 회귀
    - 비선형 회귀로 혼동하기 쉽지만 선형 회귀
    - 사이킷런은 다항 회귀를 위한 클래스를 명시적으로 제공하지는 않음
        - 선형 회귀이기 때문에 비선형 함수를 선형 모델에 적용시키는 방법을 사용해 구현
        - PolynomialFeatures 클래스를 통해 피처를 Polynomial(다항식) 피처로 변환
        - PolynomialFeatures 클래스는 degree 파라미터를 통해 입력 받은 단항식 피처를 degree에 해당하는 다항식 피처로 변환

다항 회귀를 이요한 과소적합 및 과적합 이해

- 다항 회귀는 다항식의 차수가 높아질수록 매우 복잡한 피처 간의 관계까지 모델링 가능

- 하지만 다항 회귀의 차수(degree)를 높일수록 학습 데이터에만 너무 맞춘 학습이 이루어져서 정작 테스트 데이터 환경에서는 오히려 예측 정확도가 떨어짐. → 차수가 높을수록 과적합 문제 발생!
- 좋은 예측 모델 → 학습 데이터의 패턴을 잘 반영하면서도 복잡하지 않은, 균형 잡힌 (Balanced) 모델

편향-분산 트레이드오프(Bias-Variance Trade Off)

- 머신러닝이 극복해야 할 가장 중요한 이슈 중 하나.
- 지나치게 한 방향성으로 치우친 경향 → 고편향(High Bias)성을 가지고 있다고 표현
- 학습 데이터 하나, 하나의 특성을 반영 하면서 매우 복잡한 모델 완성, 지나치게 높은 변동성 → 고분산(High Variance)성을 가졌다고 표현
    - **저편향/저분산** → 예측 결과가 실제 결과에 근접, 예측 변동이 크지 않고 특정 부분에 집중되어 있음, 아주 뛰어난 성능
    - **저편향/고분산** → 예측 결과가 실제 결과에 비교적 근접, 하지만 예측 결과가 실제 결과를 중심으로 꽤 넓은 부분에 분포
    - **고편향/저분산** → 정확한 결과에서 벗어나면서 예측이 특정 부분에 집중
    - **고편향/고분산** → 정확한 예측 결과를 벗어나면서 넓은 부분에 분포
- 일반적으로 **편향과 분산은 한쪽이 높으면 한쪽이 낮아지는 경향**
    - **편향이 높으면 분산은 낮아지고(과소적합)**
    - **분산이 높으면 편향이 낮아진다.(과적합)**
    - 편향을 낮추고 분산을 높이면서 오류가 가장 낮아지는 ‘골디락스’ 지점을 통과하면서 분산을 지속적으로 높이면 → 전체 오류값이 오히려 증가하면서 예측 성능은 다시 저하
    - **높은 편향 / 낮은 분산에서 과소적합되기 쉬움**
    - **낮은 편향 / 높은 분산에서 과적합되기 쉬움**
    - 편향과 분산이 서로 트레이드오프를 이루면서 오류 Cost 값이 최대로 낮아지는 모델을 구축하는 것이 가장 효율적인 모델을 만드는 방법

규제 선형 모델 - 릿지, 라쏘, 엘라스틱넷

규제 선형 모델의 개요

- 비용 함수는 학습 데이터의 잔차 오류 값을 최소로 하는 RSS 최소화 방법과 과적합을 방지하기 위해 회귀 계수 값이 커지지 않도록 하는 방법이 균형을 이뤄야 함
- alpha → 학습 데이터 적합 정도와 회귀 계수 값의 크기 제어를 수행하는 튜닝 파라미터
- alpha를 0에서부터 지속적으로 값을 증가시키면 회귀 계수 값의 크기를 감소시킬 수 있음
- **규제(Regularization)** → 비용 함수에 alpha 값으로 페널티를 부여해 회귀 계수 값의 크기를 감소시켜 과적합 개선
    - 크게 L2, L1 방식으로 나뉨
    - L2 규제 적용한 회귀 → 릿지(Ridge) : W의 제곱에 페널티 부여
    - L1 규제 적용한 회귀 → 라쏘(Lasso) : W의 절댓값에 페널티 부여

릿지 회귀

- 사이킷런은 Ridge 클래스를 통해 릿지 회귀 구현
- 주요 생성 파라미터 → alpha = 릿지 회귀의 alpha L2 규제 계수에 해당
- alpha 값이 증가하면서 릿지 회귀 계수는 지속적으로 감소.
- 그러나 릿지 회귀의 경우, 회귀 계수를 0으로 만들지는 않음

라쏘 회귀

- W의 절댓값에 페널티를 부여하는 L1 규제를 선형 회귀에 적용한 것.
- 불필요한 회귀 계수를 급격하게 감소시켜 0으로 만들고 제거함.
- 적절한 피쳐만 회귀에 포함시키는 피처 선택의 특성을 가지고 있음.
- 사이킷런은 Lasso 클래스를 통해 라쏘 회귀 구현
- 주요 생성 파라미터 → alpha = 라쏘 회귀의 alpha L1 규제 계수에 해당
- alpha 값이 증가하면서 일부 피처의 회귀 계수는 아예 0으로 바뀜.
- 회귀 계수가 0인 피처는 회귀 식에서 제외되면서 피처 선택의 효과를 얻을 수 있음.

엘라스틱넷 회귀

- L2 규제와 L1 규제를 결합한 회귀
- 라쏘 회귀가 서로 상관관계가 높은 피처들의 경우에 이들 중에서 중요 피처만을 셀렉션하고 다른 피처들은 모두 회귀 계수를 0으로 만드는 성향이 강함
    - 따라서 alpha 값에 따라 회귀 계수의 값이 급격히 변동할 수도 있는데, 이를 완화하기 위해 L2 규제를 추가한 것.
    - 단점 → 수행시간이 오래 걸림.
- 사이킷런은 ElasticNet 클래스를 통해 엘라스틱넷 회귀를 구현.
- 주요 생성 파라미터 → alpha, l1_ratio, alpha는 L1 규제의 알파값과 L2 규제의 알파값을 더한 것. l1_ratio 파라미터 = a / (a+b)
- alpha 값에 따른 피처들의 회귀 계수들 값이 라쏘보다는 상대적으로 0이 되는 값이 적음.

선형 회귀 모델을 위한 데이터 변환

- 선형 모델은 일반적으로 피처와 타깃값 간에 선형의 관계가 있다고 가정, 최적의 선형함수를 찾아내 결괏값 예측
- 선형 회귀 모델은 피처값과 타깃값의 분포가 정규 분포인 형태를 매우 선호.
- 타깃값의 경우 정규 분포 형태가 아니라 특정 값의 분포가 치우친 형태라면, 예측 성능에 부정적인 영향을 미칠 가능성이 높음.
- 선형 회귀 모델을 적용하기 전에 먼저 데이터에 대한 스케일링/정규화 작업을 수행하는 것이 일반적
    - 하지만 무조건 예측 성능이 오르는 건 아니고, 중요 피처들이나 타깃값의 분포가 심하게 왜곡됐을 경우, 사용.
- 타깃값의 경우 일반적으로 **로그 변환 적용.**

로지스틱 회귀

- 선형 회귀 방식을 분류에 적용한 알고리즘.
- 선형 회귀와 다른 점은 학습을 통해 선형 함수의 회귀 최적선을 찾는 것이 아니라 시그모이드(Sigmoid) 함수 최적선을 찾고 이 함수의 반환값을 확률로 간주해 확률에 따라 분류를 결정함.
- 사이킷런은 **LogisticRegression** 클래스를 제공. → 다양한 최적화 방안 선택 가능
- **최적화를 선택하는 solver 파라미터** → 'lbfgs', 'liblinear', 'newton-cg', 'sag', 'saga'
- **일반적으로는 lbfgs 또는 liblinear 선택**.
- 가볍고 빠르면서, 이진 분류 예측 성능도 뛰어남. → **이진 분류의 기본 모델**로 사용하는 경우가 많음!
- 희소한 데이터 세트 분류에서도 뛰어난 성능을 보임 → **텍스트 분류에서 자주 사용됨**.

회귀 트리

- 트리 기반의 회귀 → 회귀 트리 이용
  - 회귀를 위한 트리를 생성, 이를 기반으로 회귀 예측
  - 분류 트리와 크게 다르지는 않음.
  - 다만 리프노드에서 예측 결정값을 만드는 과정에 차이가 존재.
    - 분류 트리가 특정 클래스 레이블을 결정하는 것과는 달리 회귀 트리는 리프 노드에 속한 데이터 값의 평균값을 구해 회귀 예측값을 계산
- 결정 트리, 랜덤 포레스트, GBM, XGBoost, LightGBM 등 앞에서 본 모든 트리 기반의 알고리즘은 **분류뿐 아니라 회귀도 가능. → 트리 생성이 CART 알고리즘에 기반하고 있기 때문**
- 사이킷런에서는 결정 트리, 랜덤 포레스트, GBM에서 CART 기반의 **회귀 수행**을 할 수 있는 **Estimator 클래스를 제공**(XGBoost, LightGBM도 사이킷런 래퍼 클래스를 통해 이를 제공)
- 선형 회귀는 직선으로 예측 회귀선을 표현하는 데 반해, 회귀 트리의 경우 분할되는 데이터 지점에 따라 **브랜치를 형성, 계단 형태로 회귀선**을 만듦.