핵심 머신러닝 - 5

생성자	㉜ 재환 김
태그	머신러닝

특성 공학 (Feature Engineering)

특성 공학은 모델의 학습 정확도를 높이기 위해 데이터를 가공하는 과정입니다. 이는 모델 성능을 좌우하는 핵심 단계로, 원본 데이터를 그대로 사용하는 대신 모델이 더 쉽게 이해하고 학습할 수 있도록 데이터의 특징을 추출하고 변환합니다.

- **필요성**: 예를 들어, 고객 데이터로 미래 고객 반응을 예측하는 프로젝트를 생각해봅시다. 이 경우, 단순히 과거 데이터를 그대로 사용하면 충분한 예측 성능을 얻기 어렵습니다.
  - 원본 데이터에는 불필요한 정보나 노이즈가 포함될 수 있으며, 모델이 학습하기 어려운 형태일 수 있습니다.
  - 따라서 데이터를 처리하여 핵심 특징(feature)을 추출하고, 이를 새로운 벡터 형태로 변환하는 과정이 필요합니다.
- **과정**:
  1. **데이터 변환**: 원본 데이터의 다차원 정보를 수치형 또는 범주형 데이터로 변환합니다.
     - 예를 들어, 텍스트를 숫자로 바꾸거나 날짜를 시간 경과로 변환하는 과정이 포함될 수 있습니다.
     - 이렇게 변환된 각 값은 모델의 입력으로 사용될 수 있는 특징 벡터가 됩니다.
  2. **차원 축소 (Dimensionality Reduction)**: 특징이 너무 많으면 모델이 과적합(overfitting)될 수 있습니다. 따라서 PCA(주성분 분석)와 같은 기법으로 주요 특징만 남기는 것도 특성 공학의 중요한 부분입니다.

레이블 예제 집합 (Labeled Example Set)

레이블 예제 집합은 머신 러닝에서 데이터 학습의 기본 구조입니다. 각 예제는 (x_i, y_i) 형태로 표현됩니다.

- **입력 데이터 (x_i)**: x_i는 특성 벡터(feature vector)라고 불리는 입력 데이터입니다.
  - 이 특성 벡터는 하나 이상의 변수를 포함하며, 모델이 예측할 수 있도록 정리된 데이터입니다.

    - 각 차원은 모델이 학습할 수 있는 중요한 특성을 나타냅니다.
        - 예를 들어, 고객의 나이, 구매 횟수, 성별 등이 특성 벡터에 포함될 수 있습니다.
- **레이블** (y_i): y_i는 각 입력 데이터에 해당하는 목표값, 즉 모델이 예측하려는 정답입니다.
    - 예를 들어, 주식 가격 예측 모델에서는 미래 주가가 레이블이 될 수 있고, 이미지 분류 문제에서는 특정 이미지의 클래스(고양이, 개 등)가 레이블이 될 수 있습니다.

하이퍼파라미터 튜닝 및 과적합 방지

하이퍼파라미터 튜닝과 과적합 해결은 모델 개발에서 중요한 과제입니다.

- **하이퍼파라미터 튜닝**: 하이퍼파라미터는 모델의 구조나 학습 과정을 설정하는 변수입니다. 예를 들어, 결정 트리의 깊이나 신경망의 층 수 등이 있습니다.
    - 적절한 하이퍼파라미터 값을 찾기 위해 교차 검증(cross-validation)을 통해 다양한 값을 시도하고, 가장 적합한 값을 선택하는 과정을 튜닝이라고 합니다.
    - 이 과정에서 과적합(overfitting, 모델이 학습 데이터에 너무 맞춰져 새로운 데이터에 취약해지는 현상)을 방지하는 것이 중요합니다.

5.1.1. 원핫 인코딩 (One-hot Encoding)

원핫 인코딩은 범주형 데이터를 처리하는 대표적인 방법입니다. 범주형 데이터는 그 자체로 수치적 의미가 없어 머신 러닝 모델에 직접 입력할 수 없습니다. 원핫 인코딩은 각 범주를 이진 벡터(binary vector)로 변환하여 모델이 수치적으로 해석할 수 있게 만듭니다.

작동 원리

- **범주형 데이터의 예시**: '색깔'이라는 변수에 '빨강', '노랑', '초록'이 있다고 가정해 봅시다.
- 이 범주형 데이터는 수치적 의미가 없으므로, 숫자로 변환해야 합니다.
- 원핫 인코딩은 각 범주에 새로운 이진 벡터를 할당합니다:
    - '빨강' → [1, 0, 0]
    - '노랑' → [0, 1, 0]
    - '초록' → [0, 0, 1]

이 방식으로 범주형 데이터를 숫자로 변환합니다. 변환된 데이터는 범주 간 서열 관계가 없음을 보장합니다.

문제 해결 방식

원핫 인코딩의 목적은 범주형 데이터의 서열 관계를 제거하는 것입니다. '빨강', '노랑', '초록'을 단순히 [1, 2, 3]으로 변환하면, 모델이 숫자 간 순서를 학습해 '빨강'이 '노랑'보다 작고 '초록'보다 더 작다는 잘못된 규칙을 학습할 수 있습니다.

이를 방지하고자 원핫 인코딩은 각 범주를 독립적인 벡터로 변환하여 모델이 범주 간 서열 관계를 학습하지 않게 합니다. 이로써 과적합을 방지하고 범주형 데이터의 특성을 정확히 반영할 수 있습니다.

단점

- **차원의 증가**: 원핫 인코딩의 단점은 범주 수가 많을 때 벡터의 차원이 크게 증가한다는 점입니다.
    - 예를 들어, 100개의 서로 다른 범주가 있다면 100개의 원핫 벡터가 생성되어 차원이 100배로 증가합니다.
    - 이는 학습 속도를 늦추고 계산 자원을 많이 소모하게 만듭니다.

이 문제를 해결하기 위해 **차원 축소** 기법을 적용하거나, **희소 행렬(sparse matrix)**을 사용하여 메모리 사용량을 줄일 수 있습니다.

5.1.2. 비닝 (Binning, Bucketing)

비닝 또는 **버켓팅**은 연속형 데이터를 구간으로 나누어 범주형 데이터로 변환하는 방법입니다. 연속형 변수는 무한한 값을 가질 수 있으므로, 이를 몇 개의 범주로 나누어 모델이 더 쉽게 학습할 수 있게 만드는 과정입니다.

비닝의 목적

- **연속형 변수 처리**: '나이'와 같은 연속형 변수를 그대로 사용하면 모델이 데이터의 작은 변화에 민감하게 반응할 수 있습니다. 따라서 나이를 특정 구간으로 나누어 처리하는 것이 효과적일 수 있습니다.
    - 예를 들어, 나이를 5세 간격으로 비닝한다면:
        - 0–4세 → 첫 번째 범주 (bin1)
        - 5–9세 → 두 번째 범주 (bin2)
        - 10–14세 → 세 번째 범주 (bin3)
        - 이런 식으로 나이를 구간으로 나누어 처리할 수 있습니다.

비닝의 장점

- **모델의 안정성 증가**: 비닝은 데이터의 변동성을 줄여 모델의 학습 안정성을 높입니다. 연속형 데이터를 그대로 사용할 때 발생할 수 있는 미세한 값 변화에 대한 민감도를 줄이고, 모델이 특정 값에 과도하게 의존하지 않도록 합니다.
- **과적합 방지**: 데이터를 몇 개의 구간으로 나누면 모델이 연속적인 데이터를 지나치게 학습하는 것을 방지할 수 있습니다. 이는 특히 데이터가 불규칙하게 분포된 경우에 효과적입니다.

비닝의 단점

- **정보 손실**: 데이터를 범주로 나누면 원래의 연속적 정보가 손실될 수 있습니다. 예를 들어, 24세와 26세는 매우 비슷한 값이지만, 비닝을 통해 각각 다른 구간에 속할 수 있습니다. 이 경우, 나이의 세부적인 차이를 반영하지 못할 수 있습니다.
- **구간 설정의 어려움**: 비닝할 때 구간을 어떻게 나누느냐에 따라 결과가 달라질 수 있습니다. 구간이 너무 크거나 작으면 데이터의 중요한 패턴을 놓칠 수 있으므로, 이를 적절하게 설정하는 것이 중요합니다.

비닝의 예시

앞서 설명한 비닝 예시는 '나이' 변수를 기준으로 구간을 설정하고 이를 처리하는 방식입니다.

- 예를 들어, 나이 변수를 5세 간격으로 나눈다면:
  - `age_bin1 = [0, 4]`
  - `age_bin2 = [5, 9]`
  - 이처럼 구간을 설정해 각 나이에 해당하는 범주로 나눌 수 있습니다.

이 과정에서 원래의 연속형 데이터를 구간화하여 **모델의 복잡도를 줄이고** 과적합을 방지할 수 있습니다.

데이터 변환의 중요성

이 글에서 다룬 두 가지 방법인 원핫 인코딩과 비닝은 **데이터 전처리 과정에서 매우 중요한 역할**을 합니다. 모델의 성능을 극대화하려면 데이터를 적절하게 변환하고 가공해야 합니다.

- **원핫 인코딩**은 범주형 데이터를 수치형으로 변환하여 서열 관계를 제거합니다.
- **비닝**은 연속형 데이터를 범주형으로 변환하여 변동성을 줄입니다.

5.1.3. 정규화 (Normalization)

정규화는 데이터를 특정 범위 안으로 변환하는 과정입니다. 일반적으로 데이터 값을 0과 1 사이로 변환하며, 이는 머신 러닝 알고리즘의 처리 속도와 성능을 향상시키는 데 중요한 역할을

합니다.

정규화 과정

- **공식**: 정규화는 다음과 같은 수식으로 표현됩니다.

$$x_{\text{norm}} = \frac{x - \min(X)}{\max(X) - \min(X)}$$

  여기서:

  - x는 정규화할 데이터 값입니다.
  - $\min(X)$는 데이터셋에서의 최소값,
  - $\max(X)$는 데이터셋에서의 최대값입니다.

  이를 통해, 데이터의 최소값은 0으로, 최대값은 1로 변환됩니다. 나머지 값들은 이 범위 내에서 적절히 분포되게 됩니다.

정규화의 필요성

- **데이터의 범위를 일관되게 맞추기**: 데이터셋의 값들이 극단적으로 크거나 작으면 학습 알고리즘의 성능에 악영향을 미칠 수 있습니다. 이를 방지하기 위해 값을 [0, 1] 범위로 정규화하면 모델이 효과적으로 학습할 수 있는 데이터로 변환됩니다.
  - 예를 들어, 온도와 키처럼 서로 다른 스케일을 가진 변수들을 정규화를 통해 동일한 스케일로 맞출 수 있습니다.
- **수치적 불안정성 방지**: 극단적으로 크거나 작은 값을 그대로 사용하면 계산 과정에서 **수치적 오버플로우(numerical overflow)** 문제가 발생할 수 있습니다. 정규화를 통해 이러한 문제를 예방할 수 있습니다.

정규화의 예시

- 원래 데이터가 [350, 1450] 범위에 있다면, 이를 [0, 1] 구간으로 변환할 수 있습니다.
- 예를 들어, 350은 0으로, 1450은 1로 변환됩니다. 중간값인 900은 0.5로 변환됩니다. 이렇게 값을 변환함으로써 데이터가 동일한 스케일로 조정되어 학습 효율성이 향상됩니다.

5.1.4. 표준화 (Standardization)

표준화는 데이터의 평균을 0으로 맞추고, 표준편차를 1로 만드는 과정입니다. 이는 데이터가 정규 분포(normal distribution)를 따르도록 변환하는 중요한 과정입니다.

표준화 공식

표준화는 다음과 같은 수식을 사용합니다.

$$z = \frac{x-\mu}{\sigma}$$

- 여기서 x는 표준화할 데이터 값,
- μ는 데이터의 평균값,
- σ는 데이터의 표준편차입니다.

이를 통해 데이터는 평균이 0, 표준편차가 1인 **표준 정규 분포**로 변환됩니다.

표준화가 필요한 이유

- **비정규 분포를 정규 분포로 변환**: 실험이나 데이터 수집 과정에서 얻은 데이터는 종종 정규 분포를 따르지 않습니다. 표준화를 통해 데이터를 정규 분포에 가깝게 변환하면, 많은 머신 러닝 알고리즘의 성능이 향상됩니다.
- **극단값(outliers) 처리**: 데이터에 극단값이 존재할 경우, 표준화를 통해 값들을 평균 주변에 고르게 분포시켜 극단값의 영향을 완화할 수 있습니다.

표준화와 정규화의 차이

표준화는 주로 **데이터를 정규 분포로 변환**하는 데 사용되는 반면, 정규화는 **데이터의 범위를 조정**하는 데 활용됩니다. 두 과정 모두 데이터 전처리에서 중요한 역할을 하지만, 그 목적과 적용 상황이 다릅니다.

5.1.5. 결측치 처리 방법 (Handling Missing Data)

결측치(missing data) 처리는 데이터 분석과 머신 러닝 모델링에서 핵심적인 단계입니다. 데이터셋의 결측치를 적절히 처리하지 않으면 모델의 성능이 저하될 수 있으므로, 이를 효과적으로 다루는 것이 필수적입니다.

결측치 처리 방법

1. **결측치 제거**: 결측치가 소수이고 중요도가 낮은 경우, 해당 데이터를 제거할 수 있습니다. 단, 결측치가 다수이거나 중요한 정보를 포함할 경우 이 방법은 부적절할 수 있습니다.
2. **평균 또는 중앙값으로 대체**: 연속형 데이터의 경우, 결측치를 평균값이나 중앙값으로 대체할 수 있습니다. 이 방법은 결측치를 추정하는 간단한 방법이지만, 데이터의 분포를 왜곡하지 않도록 주의해야 합니다.
3. **모델 기반 대체**: 머신 러닝 모델을 사용하여 결측치를 예측할 수 있습니다. 예를 들어, 회귀 모델을 통해 결측된 값을 추정할 수 있습니다.

결측치 처리의 중요성

- **데이터 품질 향상**: 결측치를 적절히 처리하면 데이터의 품질이 개선되고, 이는 모델의 성능 향상으로 이어집니다. 결측치를 그대로 두면 모델이 부정확한 학습을 할 수 있으므로, 결측치 처리는 데이터 전처리 과정에서 필수적입니다.

5.1.6. 데이터 대체 기법

데이터 대체 기법은 결측치를 효과적으로 처리하는 방법입니다. 결측치는 데이터셋에서 중요한 정보가 누락되었을 때 발생하며, 이를 적절히 처리하지 않으면 모델의 학습과 예측 성능에 심각한 영향을 미칠 수 있습니다.

1. 평균값 대체

결측치가 발생한 변수의 다른 값들의 평균으로 결측치를 대체하는 방법입니다. 이 방법은 데이터가 정규 분포를 따르거나 값의 편차가 크지 않을 때 주로 사용됩니다.

- **장점**: 계산이 간단하고 직관적입니다.
- **단점**: 데이터의 분산을 축소시킬 수 있으며, 너무 많은 결측치를 평균값으로 대체하면 데이터의 분포가 왜곡될 수 있습니다.

2. 중앙값 대체

특정 변수의 결측치를 중앙값(또는 중위수)으로 대체하는 방법입니다. 이는 데이터가 비대칭적이거나 이상치(outliers)가 있는 경우 유용합니다.

- **장점**: 이상치에 덜 민감하므로, 평균값보다 안정적인 대체 방법입니다.
- **단점**: 데이터가 정규 분포를 따를 경우, 중앙값 대체는 평균값 대체보다 효과가 떨어질 수 있습니다.

3. 특정 값으로 대체

범주형 데이터에서 결측값을 '모름' 또는 '기타'와 같은 특정 값으로 대체하거나, 연속형 데이터에서 결측치를 0 또는 특정 범위 내 중간값으로 대체하는 방법입니다.

- **예시**: 데이터셋에서 '성별' 변수가 결측된 경우, '모름'이라는 범주를 추가하여 결측된 데이터를 그 범주로 대체할 수 있습니다.
- **장점**: 해석이 명확하며, 결측치를 명시적으로 처리합니다.
- **단점**: 결측치가 많을 경우, 이 방법은 전체 데이터의 구조를 왜곡시킬 수 있습니다.

4. 지표 변수 추가 (Indicator Variable)

결측치가 발생한 위치를 명확히 표시하는 지표 변수를 추가하는 방법입니다. 데이터가 결측되었는지 여부를 나타내는 이진 변수(0 또는 1)를 추가하여 결측 여부를 기록합니다.

- **예시**: '나이' 변수가 결측된 경우, '나이_결측여부'라는 새로운 변수를 추가하고, 결측된 경우 1, 아닌 경우 0으로 값을 할당합니다.
- **장점**: 결측치의 존재 여부를 명확히 표시하여 결측치 처리의 투명성을 보장합니다.
- **단점**: 데이터에 변수가 하나 추가되어 차원이 늘어납니다. 또한, 모델이 이러한 변수를 적절히 해석하지 못할 수도 있습니다.

5. 모델 기반 대체

결측치를 예측하기 위한 별도의 머신 러닝 모델을 사용하여 결측값을 추정하는 방법입니다. 예를 들어, 회귀 모델을 사용하여 결측값을 예측할 수 있습니다.

- **장점**: 결측치를 데이터의 패턴에 맞게 예측할 수 있습니다.
- **단점**: 추가적인 모델링이 필요하고 시간이 오래 걸릴 수 있습니다. 결측치가 많은 경우에는 오차가 발생할 가능성이 높습니다.

6. 평균 대체 기법의 공식

이미지에서는 결측치를 평균값으로 대체하는 방법을 수식으로 설명하고 있습니다. 결측치가 발생한 변수의 평균을 계산하여 결측치를 대체하는 방식은 다음과 같은 수식으로 표현됩니다.

$$\hat{x}^m = \frac{1}{N} \sum_{i=1}^{N} x_i^m$$

- 여기서 $\hat{x}^m$은 결측값을 대체할 평균값이며, N은 데이터셋에서 결측치가 없는 값들의 개수입니다.
- 이는 결측치를 간단히 처리하는 기본적인 방법으로, 데이터가 균등하게 분포된 경우에 효과적입니다.

5.2. 학습 알고리즘 결정하기

머신 러닝에서 학습 알고리즘 선택은 매우 중요합니다. 선택한 알고리즘에 따라 모델의 성능이 크게 달라질 수 있으므로, 문제의 특성에 맞는 적절한 알고리즘을 결정해야 합니다.

1. 모델의 설명력 (Explainability)

설명력이 중요한 경우, 결과 해석이 쉽고 명확한 알고리즘을 선택해야 합니다. 예를 들어, **결정 트리(Decision Tree)**는 각 단계의 분기를 시각적으로 확인할 수 있어 결과 설명이 용이합니다.

- **결정 트리(Decision Tree)**: 이 알고리즘은 예측 과정에서 어떤 특성이 가장 중요한 역할을 했는지, 어떤 분기에서 결정이 이루어졌는지를 명확하게 보여줍니다. 따라서 예측 결과 해석이 필요한 비즈니스 환경이나 의료 분야에서 자주 사용됩니다.

- **예시**: 고객의 구매 여부를 예측하는 모델에서 나이, 소득, 구매 이력 등의 변수가 예측에 미치는 영향을 시각적으로 확인할 수 있습니다.

2. 시간과 자원 제약 고려

모든 알고리즘이 모든 문제에 적합한 것은 아닙니다. 일부 알고리즘은 시간과 자원을 많이 소모합니다. 시간이 부족하거나 자원이 제한된 상황에서는 상대적으로 **경량 알고리즘**을 선택해야 합니다.

- **KNN(K-최근접 이웃)**: KNN 알고리즘은 단순하지만, 데이터 규모가 커질수록 계산량이 증가하고 학습 속도가 느려집니다. 따라서 대규모 데이터 처리에는 적합하지 않을 수 있습니다.
- **신경망(Neural Networks)**: 복잡한 문제 해결에 강력하지만, 학습 시간이 길고 특히 깊은 신경망(Deep Learning)은 GPU 자원을 많이 소모합니다. 신경망은 대규모 데이터와 복잡한 문제에 적합하지만, 자원이 부족한 환경에서는 다른 알고리즘이 더 적절할 수 있습니다.

3. 모델의 정확도 (Accuracy)

높은 예측 정확도가 최종 목표라면, 다양한 알고리즘을 실험하고 가장 정확한 알고리즘을 선택해야 합니다. **랜덤 포레스트(Random Forest)**, **서포트 벡터 머신(SVM)** 등은 높은 예측 성능으로 자주 사용되는 알고리즘입니다.

- **랜덤 포레스트(Random Forest)**: 여러 결정 트리를 결합하여 예측하는 방식으로, 개별 트리의 약점을 보완해 전체적인 성능을 향상시킵니다. 랜덤 포레스트는 높은 예측 성능을 보이며, 이상치와 노이즈에 강한 장점이 있습니다.
  - 예시: 주가 예측, 의료 데이터 분석 등 다양한 분야에서 높은 정확도를 보여줍니다.
- **SVM(서포트 벡터 머신)**: 데이터의 경계를 명확하게 분리하는 **초평면**을 찾아내는 알고리즘으로, 특히 고차원 데이터에서 우수한 성능을 발휘합니다. 다만 대규모 데이터에서는 학습 속도가 느려질 수 있습니다.

4. 모델 선택 시 고려해야 할 기타 요소

- **과적합(Overfitting) 방지**: 설명력과 정확도뿐 아니라, 모델이 훈련 데이터에 과적합되지 않도록 주의해야 합니다. 과적합은 모델이 훈련 데이터에 지나치게 의존해 새로운 데이터에 대해 좋은 성능을 내지 못하는 현상입니다. 이를 방지하기 위해 **교차 검증(Cross-validation)**과 같은 기법을 활용할 수 있습니다.
- **데이터의 특성**: 각 알고리즘은 데이터 특성에 따라 성능이 달라집니다. 예를 들어, 텍스트 데이터에는 **자연어 처리(NLP)** 알고리즘이, 이미지 데이터에는 **합성곱 신경망(CNN)**과 같은 특화된 알고리즘이 적합할 수 있습니다.

알고리즘 선택 시 고려해야 할 요소들

머신러닝 모델 구축 시 알고리즘 선택은 매우 중요합니다. 각 알고리즘은 데이터 특성, 연산 자원, 예측 목표 등에 따라 적합성이 달라집니다. 다음은 이 과정에서 고려해야 할 주요 요소들입니다.

1. 인메모리 vs 아웃오브메모리

- **인메모리 처리**: 데이터를 한 번에 RAM에 올릴 수 있을 때 효율적입니다. 메모리에 데이터를 적재한 상태에서 학습을 진행하는 알고리즘이 유리합니다.
- **아웃오브메모리 처리**: 데이터가 메모리 용량을 초과할 때는 점진적 학습 알고리즘(incremental learning algorithm)이 필요합니다. 이는 메모리 자원이 제한적일 때 특히 유용합니다.

2. 특성과 예제의 개수

데이터셋의 예제 수와 각 예제의 특징(feature) 수에 따라 적합한 알고리즘이 달라집니다.

- **특성과 예제가 많은 경우**: 랜덤 포레스트, 부스팅 알고리즘 등 복잡한 알고리즘이 적합할 수 있습니다.
- **특성과 예제가 적은 경우**: SVM이나 선형 회귀와 같은 비교적 단순한 알고리즘이 효과적일 수 있습니다.

3. 범주형 특성 vs 수치형 특성

- **범주형 데이터**: 특성값을 수치형으로 변환하지 않고 처리할 수 있는 알고리즘이 필요합니다. 결정 트리나 KNN이 범주형 특성을 잘 다룹니다.
- **수치형 데이터**: 선형 회귀나 SVM과 같은 알고리즘이 수치형 특성을 효과적으로 처리합니다.

4. 데이터의 비선형성 (Non-linearity)

데이터의 선형성 여부에 따라 적합한 알고리즘이 달라집니다.

- **선형적 데이터**: 데이터 관계가 직선으로 표현될 수 있다면 선형 회귀나 SVM이 적합합니다.
- **비선형적 데이터**: 선형으로 구분되지 않는 데이터는 심층 신경망(DNN), 다층 퍼셉트론(MLP) 등 비선형 관계를 처리할 수 있는 알고리즘이 필요합니다.

5. 훈련 속도

알고리즘의 훈련 속도는 모델 복잡도, 데이터 양, 가용 자원에 따라 달라집니다.

- **속도 우선**: 빠른 훈련이 필요하다면 로지스틱 회귀나 결정 트리 같은 경량 알고리즘이 적합합니다.
- **정확도 우선**: 시간이 많이 걸리더라도 높은 예측 성능이 중요하다면 랜덤 포레스트, 심층 신경망(RNN, CNN) 등 복잡한 알고리즘을 고려해볼 수 있습니다.

6. 예측 속도

예측 수행 속도도 중요한 고려 사항입니다. 일부 알고리즘은 빠른 예측 성능을 제공합니다.

- **실시간 예측**: SVM, KNN 등은 훈련에 시간이 걸릴 수 있지만, 실시간 예측이 필요한 경우 빠르게 결과를 제공할 수 있습니다.
- **대규모 예측**: 랜덤 포레스트나 신경망은 대규모 데이터를 한 번에 예측할 때 효과적입니다.

5.3. 세 가지 데이터 집합

머신 러닝에서 데이터를 훈련시키고 모델을 평가할 때는 데이터를 효율적으로 분할해야 합니다. 일반적으로 데이터를 세 가지 집합으로 나누어 사용합니다.

1. 훈련 집합 (Training Set)

- 훈련 집합은 모델이 학습하는 데 사용되는 데이터입니다. 데이터의 가장 큰 부분을 차지하며, 모델이 패턴을 찾고 각 예제를 학습하는 데 활용됩니다.
- 충분한 크기의 훈련 집합이 필요합니다. 그렇지 않으면 모델이 패턴을 제대로 학습하지 못할 수 있어, 일반적으로 가능한 한 많은 데이터를 할당합니다.

2. 검증 집합 (Validation Set)

- 검증 집합은 모델 훈련 중 과적합(Overfitting)을 방지하기 위해 사용됩니다. 이를 통해 모델이 훈련 데이터에 과도하게 맞추는 것을 막고, 새로운 데이터에 대한 일반화 능력을 평가합니다.
- 검증 집합의 성능은 모델의 하이퍼파라미터(예: 학습률, 정규화 강도 등)를 조정하는 데 활용됩니다.

3. 테스트 집합 (Test Set)

- 테스트 집합은 모델의 최종 성능을 평가하기 위한 데이터입니다. 이 데이터는 훈련과 검증 과정에서 전혀 사용되지 않아, 모델의 실제 성능을 객관적으로 측정할 수 있습니다.
- 테스트 집합에서의 성능이 모델의 최종 품질을 나타냅니다.

알고리즘 선택 가이드

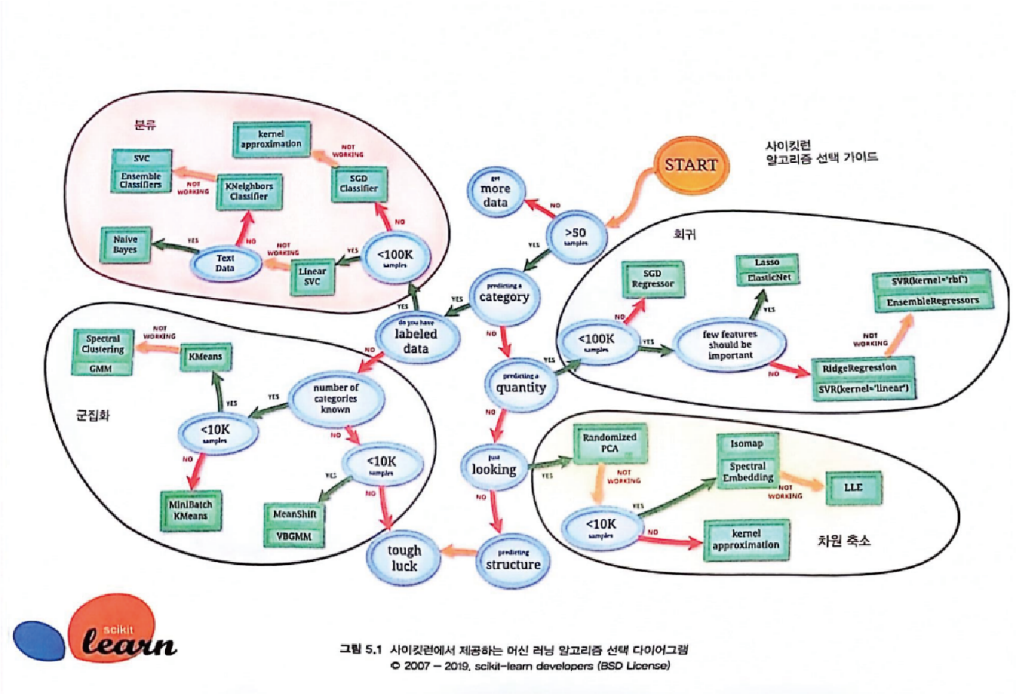

그림 5.1 사이킷런에서 제공하는 머신 러닝 알고리즘 선택 다이어그램
© 2007 – 2019, scikit-learn developers (BSD License)

머신 러닝 알고리즘 선택은 데이터의 특성, 크기, 문제 유형에 따라 달라집니다. 이미지 왼쪽의 다이어그램은 **사이킷런(SciKit-Learn)** 라이브러리가 제공하는 알고리즘 선택 과정을 도식화한 가이드로, 사용자가 특정 문제에 적합한 알고리즘을 결정하는 데 도움을 줍니다.

1. 데이터 크기와 알고리즘 선택

데이터 크기에 따라 적합한 알고리즘이 달라집니다. 이는 메모리 사용량과 연산 효율성을 고려한 결정입니다.

- **데이터 크기가 10K 이하일 때**:
  - 적은 데이터로도 간단한 알고리즘이 좋은 성능을 낼 수 있습니다. 대표적으로 **KNN(K-최근접 이웃)**, **결정 트리(Decision Tree)**, **로지스틱 회귀(Logistic Regression)** 등이 있습니다.
  - **KNN**은 새로운 데이터와 기존 데이터 간의 거리를 계산해 가까운 이웃을 찾아 예측하는 인스턴스 기반 알고리즘입니다. 하지만 데이터가 커지면 계산량이 급증하므로 소규모 데이터에 적합합니다.
  - **결정 트리**는 직관적인 데이터 분류 방법으로, 적은 데이터에서 해석하기 쉬운 모델을 생성합니다.
- **데이터 크기가 10K에서 100K 사이일 때**:
  - 데이터가 커지면서 더 복잡한 알고리즘이 필요해집니다. **랜덤 포레스트(Random Forest)**, **SVM(서포트 벡터 머신)** 등이 권장됩니다.

    - **랜덤 포레스트**는 여러 결정 트리를 앙상블하여 성능을 높이는 방법으로, 과적합을 방지하고 안정적인 예측을 제공합니다.
    - **SVM**은 데이터를 고차원 공간으로 매핑해 최적의 초평면을 찾아 분류하는 알고리즘으로, 비교적 큰 데이터에서도 강력한 성능을 보입니다.
- **데이터 크기가 100K 이상일 때**:
    - 대규모 데이터셋 처리에는 더 복잡한 알고리즘이 필요하며, **심층 신경망(Deep Neural Network)**이나 **그라디언트 부스팅(Gradient Boosting)** 같은 알고리즘이 적합합니다.
    - **심층 신경망(DNN)**은 다층 신경망으로 복잡한 패턴을 학습합니다. 특히 대규모 데이터에서 뛰어난 성능을 보이며, 이미지나 텍스트 같은 비정형 데이터에 많이 사용됩니다.
    - **그라디언트 부스팅**은 여러 약한 학습기(weak learner)를 순차적으로 학습시켜 오차를 줄이는 방식으로 고성능 예측 모델을 만듭니다. 특히 **XGBoost** 같은 변형 알고리즘은 대규모 데이터에서도 효율적으로 작동합니다.

2. 문제의 유형(분류 vs 회귀)

- **분류 문제(Classification)**: 데이터를 몇 개의 고정된 클래스 중 하나로 분류하는 경우, **분류 알고리즘**이 필요합니다.
    - **로지스틱 회귀(Logistic Regression)**: 이진 분류 문제에 주로 사용되며, 클래스 경계가 선형적일 때 효과적입니다.
    - **결정 트리(Decision Tree)**와 **랜덤 포레스트(Random Forest)**: 복잡한 데이터에 효과적이며, 데이터의 분류 과정을 직관적으로 시각화할 수 있습니다.
    - **SVM(서포트 벡터 머신)**: 고차원 데이터 처리에 강력하며, 비선형 데이터도 커널 트릭을 통해 효과적으로 분류합니다.
- **회귀 문제(Regression)**: 연속적인 값을 예측해야 할 때 **회귀 알고리즘**을 사용합니다.
    - **선형 회귀(Linear Regression)**: 가장 기본적인 회귀 알고리즘으로, 데이터가 선형 관계를 가질 때 효과적입니다.
    - **랜덤 포레스트**와 **그라디언트 부스팅**: 복잡한 데이터 패턴을 포착하여 비선형 회귀 문제에서도 우수한 성능을 보입니다.

3. 데이터 전처리 및 스케일링

- **특성 스케일링 필요성**: 일부 알고리즘은 입력 데이터의 스케일에 민감합니다. **SVM**, **로지스틱 회귀** 등은 특성 간 크기 차이가 클 경우 성능이 저하될 수 있습니다. 이를 해결하

기 위해 **표준화(Standardization)**나 **정규화(Normalization)** 같은 전처리 과정으로 데이터 범위를 조정해야 합니다.

언더피팅과 오버피팅

1. 언더피팅(Underfitting)

언더피팅은 모델이 데이터를 충분히 학습하지 못한 상태를 의미합니다. 이는 모델이 지나치게 단순하거나 학습 시간이 불충분할 때 발생할 수 있습니다. 이 경우, 모델은 훈련 데이터와 테스트 데이터 모두에서 저조한 성능을 보입니다.

- **언더피팅의 주요 원인**:
  - 모델의 과도한 단순성 (예: 복잡한 비선형 데이터를 선형 모델로 처리하려는 시도)
  - 불충분한 학습 데이터
  - 부족한 학습 시간 또는 지나치게 낮은 학습률로 인한 불완전한 학습
- **해결 방안**:
  - **모델 복잡도 증가**: 더 정교한 모델 사용 또는 추가 특성(feature) 도입으로 모델의 패턴 학습 능력 향상
  - **훈련 데이터 확충**: 더 많은 데이터 수집 또는 데이터 증강(data augmentation) 기법 활용
  - **학습률 최적화**: 적절한 학습률 조정으로 모델의 빠른 수렴 유도

2. 오버피팅(Overfitting)

오버피팅은 모델이 훈련 데이터에 과도하게 적응하여 새로운 데이터에 대한 예측 성능이 저하되는 현상입니다. 이는 모델이 훈련 데이터의 노이즈까지 학습하여 일반화 능력이 떨어질 때 발생합니다.

- **오버피팅의 주요 원인**:
  - 과도하게 복잡한 모델 (예: 지나치게 깊은 결정 트리 또는 파라미터가 과다한 신경망)
  - 불충분하거나 편향된 훈련 데이터
  - 데이터의 실제 패턴뿐 아니라 노이즈까지 학습하는 경우
- **해결 방안**:
  - **정규화 기법(Regularization)**: L1 또는 L2 정규화를 통해 모델의 복잡도를 제어하고, 가중치(weight)에 제약을 가해 과도한 학습을 방지

- **교차 검증(Cross-validation)**: 데이터를 여러 부분으로 나누어 학습 및 평가함으로써 특정 데이터에 대한 과적합 방지
- **데이터 확충**: 더 다양한 데이터를 활용하여 모델의 학습 패턴을 다양화하고 과적합 위험 감소
- **드롭아웃(Dropout)**: 신경망에서 학습 중 일부 노드를 무작위로 비활성화하여 네트워크의 과적합 방지

데이터 분할 방식 (Train/Test Split)

머신 러닝 모델의 학습과 평가 시, 데이터를 훈련 집합, 검증 집합, 테스트 집합으로 나누는 것이 일반적입니다. 이 방식은 과적합을 방지하고 모델의 성능을 정확히 평가하는 데 도움이 됩니다.

1. 훈련 집합(Training Set)

- 모델 학습에 사용되는 주요 데이터셋입니다. 전체 데이터의 70%~80%를 차지하며, 모델이 데이터의 패턴을 학습하는 데 활용됩니다. 충분한 훈련 데이터는 모델이 다양한 패턴을 효과적으로 학습하는 데 필수적입니다.

2. 검증 집합(Validation Set)

- 훈련 과정 중 모델의 성능을 평가하고 하이퍼파라미터를 조정하는 데 사용됩니다. 보통 전체 데이터의 10%~15%를 차지합니다. 검증 집합은 모델의 훈련 데이터 과적합을 방지하는 데 중요한 역할을 합니다.

3. 테스트 집합(Test Set)

- 모델의 일반화 능력을 최종적으로 평가하는 데 사용됩니다. 훈련과 검증 과정에서 전혀 사용되지 않은 데이터로, 모델이 실제 환경에서 어떻게 작동할지 예측하는 데 활용됩니다.

언더피팅(Underfitting)과 오버피팅(Overfitting)의 시각적 예시

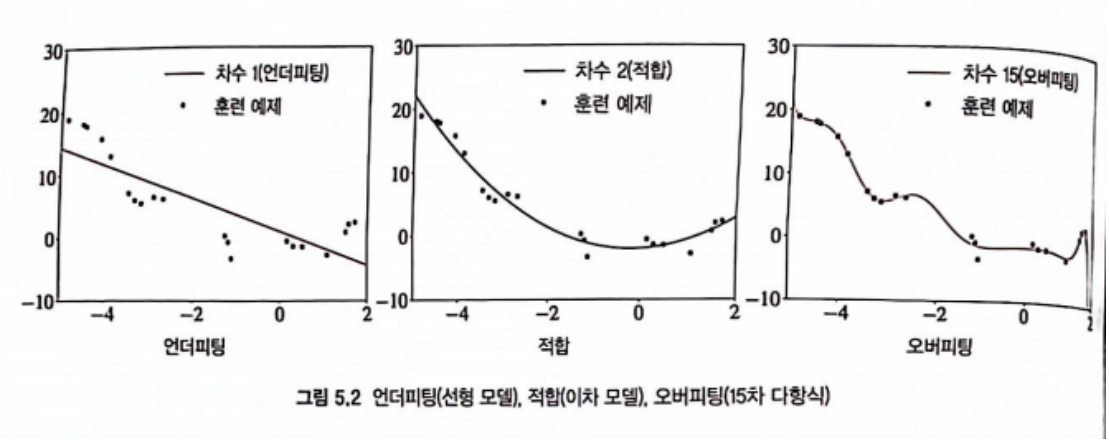

그림 5.2 언더피팅(선형 모델), 적합(이차 모델), 오버피팅(15차 다항식)

그림 5.2는 언더피팅, 적절한 모델, 오버피팅의 차이를 시각적으로 보여줍니다. 이 그래프들은 훈련 예제에 대한 모델의 예측 결과를 시각화하여 모델의 학습 상태를 명확히 나타냅니다.

1. 언더피팅 (Underfitting)

첫 번째 그래프는 **언더피팅**의 예시입니다. 이 그래프에서 모델은 데이터의 복잡성을 충분히 반영하지 못해, 단순한 1차 함수(선형 모델)로 데이터를 설명하려 합니다.

- **특징**:
  - 모델이 데이터의 복잡한 패턴을 반영하지 못합니다.
  - 훈련 데이터에 제대로 적합하지 못해 예측 성능이 매우 낮습니다.
  - 그래프상 훈련 예제(점)와 모델의 예측 선(실선) 사이에 큰 차이가 나타납니다.
- **원인**:
  - 모델이 지나치게 단순하거나 데이터의 복잡한 패턴을 포착하기에 학습이 부족합니다.
  - 적절한 특성이나 매개변수를 사용하지 않거나, 복잡한 비선형성을 처리할 수 없는 구조를 가집니다.
- **해결 방법**:
  - 더 복잡한 모델을 사용하거나 모델의 매개변수를 조정하여 데이터 패턴을 더 잘 반영합니다.
  - **특성 공학**을 통해 더 유용한 특성(feature)을 추가합니다.

2. 적절한 모델 (Well-fitted Model)

두 번째 그래프는 **적절하게 학습된 모델**을 보여줍니다. 이 모델은 훈련 데이터를 잘 설명하며 적절한 예측 성능을 발휘합니다.

- **특징**:

    - 모델이 훈련 데이터를 잘 설명하며, 훈련 예제와 예측 선이 조화롭게 일치합니다.
    - 훈련 데이터에 적합한 복잡도를 가진 모델이 적절한 학습을 수행한 결과입니다.
- **결과**:
    - 훈련 데이터뿐만 아니라 새로운 데이터(테스트 데이터)에 대해서도 좋은 성능을 기대할 수 있습니다.
    - 모델의 복잡도와 일반화 성능 간의 균형이 잘 맞아 과적합이나 과소적합 문제를 피할 수 있습니다.

3. 오버피팅 (Overfitting)

세 번째 그래프는 **오버피팅**을 나타냅니다. 모델이 너무 복잡하여 훈련 데이터에 지나치게 맞추려다 보니 데이터의 노이즈까지 학습한 상태입니다.

- **특징**:
    - 모델이 훈련 데이터에 과도하게 맞춰져 있습니다. 훈련 예제와 모델의 예측 선이 거의 완벽하게 일치하지만, 이는 과도한 학습의 결과입니다.
    - 오버피팅된 모델은 훈련 데이터에서는 매우 높은 성능을 보이지만, 새로운 데이터에서는 일반화 능력 부족으로 성능이 급격히 떨어질 수 있습니다.
- **원인**:
    - 모델의 복잡도가 데이터의 실제 패턴을 넘어 노이즈까지 학습한 경우 발생합니다.
    - **모델의 매개변수가 너무 많거나 모델 구조가 지나치게 복잡할 때** 나타납니다.
- **해결 방법**:
    - **정규화 기법(Regularization)**을 사용하여 모델의 복잡도를 줄입니다. 대표적으로 L1, L2 정규화 또는 드롭아웃(Dropout)을 활용할 수 있습니다.
    - **교차 검증(Cross-validation)**을 통해 과적합을 방지하고, 모델의 새로운 데이터에 대한 일반화 성능을 높입니다.
    - **데이터의 양을 늘리거나** 불필요한 특성을 제거하여 모델의 복잡도를 줄이는 방법도 오버피팅 해결에 도움이 됩니다.

언더피팅과 오버피팅을 해결하는 방법

언더피팅과 오버피팅은 머신 러닝에서 흔히 발생하는 문제입니다. 이를 해결하기 위한 몇 가지 효과적인 방법을 살펴보겠습니다.

언더피팅 해결 방법

1. **모델 복잡도 증가**:
    - 더 복잡한 모델이나 비선형 모델을 사용하여 데이터를 더 정확히 설명할 수 있습니다.
    - 예를 들어, 선형 회귀 대신 **다항 회귀(Polynomial Regression)**나 **신경망(Neural Network)** 같은 비선형 모델을 도입할 수 있습니다.
2. **피처 엔지니어링**:
    - 더 유용한 특성을 만들어 모델이 학습할 수 있도록 데이터를 보강합니다. 이를 통해 모델이 더 복잡한 패턴을 학습할 수 있게 됩니다.
3. **학습 시간 최적화**:
    - 모델의 학습 시간을 늘리거나 학습률(learning rate)을 적절히 조정하여 모델이 데이터를 충분히 학습할 수 있도록 합니다.

오버피팅 해결 방법

1. **정규화(Regularization)**:
    - L1 또는 L2 정규화를 사용하여 모델의 과도한 복잡성을 제한합니다.
    - **L2 정규화(Ridge Regression)**는 가중치의 크기를 제한하여 모델의 과적합을 방지합니다.
    - **L1 정규화(Lasso Regression)**는 일부 가중치를 0으로 만들어 불필요한 특성을 제거하고 모델을 간소화합니다.
2. **교차 검증(Cross-validation)**:
    - 데이터를 여러 번 나누어 모델을 학습하고 검증하는 과정에서 과적합을 방지합니다.
    - 이를 통해 모델의 일반화 성능을 평가하고 개선할 수 있습니다.
3. **데이터 확장**:
    - 더 많고 다양한 데이터를 수집하면 모델이 다양한 패턴을 학습할 수 있어 과적합을 줄일 수 있습니다. 특히 훈련 데이터가 부족할 때 오버피팅이 자주 발생하므로, 가능한 한 다양한 데이터를 확보하는 것이 중요합니다.
4. **드롭아웃(Dropout)**:
    - 신경망 모델에서 자주 사용되는 기법으로, 학습 과정에서 일부 뉴런을 무작위로 비활성화합니다. 이를 통해 네트워크의 과적합을 방지하고 일반화 성능을 향상시킬 수 있습니다.

5.5 규제화 (Regularization)

규제화는 학습 알고리즘의 모델이 과도하게 복잡해지는 것을 방지하기 위해 제약을 가하는 기법입니다. 이를 통해 모델의 일반화 성능을 높이고 오버피팅을 예방합니다. 규제화는 편향-분산 트레이드오프(bias-variance tradeoff)를 조절하는 중요한 역할을 수행합니다.

편향-분산 트레이드오프

- **편향(bias)**: 모델이 지나치게 단순하여 데이터의 복잡한 패턴을 포착하지 못하는 정도를 의미합니다. 모델의 복잡도가 낮으면 편향이 높아져 언더피팅 문제가 발생할 수 있습니다.
- **분산(variance)**: 모델이 너무 복잡하여 훈련 데이터에 과도하게 적합되는 정도를 나타냅니다. 분산이 지나치게 크면 오버피팅이 발생하게 됩니다.

규제화를 통해 모델의 복잡도를 적절히 조절하면, 편향과 분산 사이의 균형을 맞출 수 있습니다. 이로써 모델이 새로운 데이터에 대해 더 우수한 성능을 발휘할 수 있게 됩니다.

L1 규제화와 L2 규제화

L1과 L2 규제화는 가장 널리 사용되는 두 가지 규제 기법입니다. 이들은 모두 손실 함수에 추가적인 항(term)을 더해 모델의 복잡도를 제한합니다.

1. L1 규제화 (Lasso Regularization)

L1 규제화는 목표 함수에 가중치의 절댓값 합을 더해 일부 가중치를 0으로 만듭니다. 이로 인해 불필요한 특성을 자동으로 제거하는 **피처 선택(feature selection)** 효과를 얻습니다.

- **수식**:

  $\min_{w,b} \left(\frac{1}{N} \sum_{i=1}^{N} (y_i - f(w, x_i))^2 + C \sum_{j=1}^{D} |w_j| \right)$
  여기서:
  - w: 가중치(weight)
  - b: 절편(bias)
  - y_i: 실제 값
  - $f(w, x_i)$: 모델의 예측 값
  - C: 규제화의 강도를 조절하는 하이퍼파라미터

L1 규제화는 가중치를 0으로 만들어 일부 특성을 제거함으로써 모델을 더 간결하게 만듭니다. 이러한 특성 때문에 **피처 선택**이 중요한 문제에서 자주 활용됩니다.

- **장점**:

- L1 규제화는 모델의 일부 가중치를 0으로 만들어 불필요한 특성을 자연스럽게 제거합니다. 이로 인해 자동적인 특성 선택이 이루어지며, 결과적으로 해석하기 쉬운 모델을 얻을 수 있습니다.

2. L2 규제화 (Ridge Regularization)

L2 규제화는 목표 함수에 가중치의 제곱합을 추가합니다. 이로 인해 모든 가중치가 작아지도록 유도하여 모델의 복잡도를 줄이고 과적합을 방지합니다.

- **수식**: $\min_{w,b} \left(\frac{1}{N} \sum_{i=1}^{N} (y_i - f(w, x_i))^2 + C \sum_{j=1}^{D} w_j^2 \right)$

  L2 규제화에서는 가중치의 크기가 커지는 것을 방지하여 모델이 훈련 데이터에 지나치게 맞춰지는 것을 막습니다.

- **장점**:

- L2 규제화는 모델의 가중치가 과도하게 커지는 것을 방지하여 훈련 데이터에 과적합되는 것을 막습니다. 이 방식은 모든 특성을 유지하면서도 각 특성의 영향력을 균형 있게 조절할 수 있습니다.

- **L1과 L2 규제화의 차이**:
  - **L1 규제화**는 특성 선택 효과가 있어 복잡한 모델을 간소화하는 데 효과적입니다.
  - **L2 규제화**는 모든 특성에 일정한 규제를 적용하여 모델이 훈련 데이터의 노이즈에 과민하게 반응하지 않도록 합니다.

Elastic Net: L1과 L2 규제화의 결합

Elastic Net은 L1과 L2 규제화를 혼합하여 두 방식의 장점을 결합한 기법입니다. 이 방법은 L1과 L2 규제화의 혼합 비율을 조정함으로써 피처 선택과 과적합 방지 효과를 동시에 달성합니다.

- **수식**:

  $\min_{w,b} \left(\frac{1}{N} \sum_{i=1}^{N} (y_i - f(w, x_i))^2 + \alpha \left((1 - r) \sum_{j=1}^{D} |w_j| + r \sum_{j=1}^{D} w_j^2 \right) \right)$
  여기서 α는 규제화 강도, r은 L1과 L2의 비율을 조정하는 하이퍼파라미터입니다.

Elastic Net은 L1 규제화의 특성 선택 효과와 L2 규제화의 가중치 균형 조정 효과를 동시에 활용합니다. 이러한 특성으로 인해 Elastic Net은 고차원 데이터 분석에 특히 유용한 방법입니다.

규제화의 목적

규제화의 최종 목적은 모델의 **일반화 성능을 높이고**, 과적합을 방지하는 것입니다. 이를 위해 모델의 가중치를 제어하여 모델이 훈련 데이터에 너무 민감하지 않도록 조정합니다.

규제화 강도 조절: 하이퍼파라미터 C

규제화에서 중요한 하이퍼파라미터 중 하나는 C입니다. C는 규제화의 강도를 조절하며, C 값이 클수록 규제화의 효과가 줄어들고, C값이 작을수록 규제화가 강해집니다. 이를 통해 모델의 복잡도를 적절히 조절할 수 있습니다.

- C 값이 너무 크면 규제화의 효과가 거의 없어져 과적합이 발생할 수 있습니다.
- C값이 너무 작으면 모델이 과도하게 단순해져 언더피팅이 발생할 수 있습니다.

따라서 적절한 C 값을 찾는 것이 매우 중요하며, 이는 **교차 검증**을 통해 최적의 값을 찾을 수 있습니다.

5.6 모델 성능 평가 방법

머신 러닝 모델이 훈련 데이터로 학습한 후, 테스트 데이터에 대한 성능을 평가하는 단계입니다. 이는 모델이 새로운 데이터를 얼마나 정확히 예측할 수 있는지 판단하는 과정입니다.

모델 평가의 목표

- **일반화 성능 평가**: 모델이 훈련 데이터뿐만 아니라 새로운 데이터에도 잘 작동하는지 평가합니다. 이를 **일반화 능력(generalization)**이라고 하며, 이 능력이 뛰어난 모델이 좋은 모델입니다.
- 모델의 성능은 주로 **테스트 집합(test set)**에서 평가됩니다. 테스트 집합은 훈련에 사용되지 않은 데이터로, 모델의 실제 예측 능력을 평가하는 기준이 됩니다.

모델 성능 평가 기준

1. **평균 제곱 오차(MSE: Mean Squared Error)**:
    - MSE는 회귀 문제에서 주로 사용하는 평가 지표로, 모델의 예측값과 실제값 차이의 제곱 평균입니다. 값이 작을수록 모델의 성능이 좋습니다.
    - MSE는 과적합 여부를 확인하는 데도 유용합니다. 훈련 데이터와 테스트 데이터의 MSE 차이가 크면 과적합일 가능성이 높습니다.
2. **정확도(Accuracy)**:
    - 분류 문제의 기본적인 평가 지표로, 모델이 정확히 분류한 데이터의 비율을 나타냅니다. 다만, 불균형한 데이터셋에서는 정확도만으로 성능을 평가하기에 한계가 있습니다.

3. **정밀도(Precision)와 재현율(Recall)**:
    - **정밀도**는 모델이 양성이라고 예측한 것 중 실제 양성인 데이터의 비율입니다. 예를 들어, 스팸 메일 분류에서 정밀도는 스팸으로 예측한 메일 중 실제 스팸인 비율입니다.
    - **재현율**은 실제 양성 데이터 중 모델이 정확히 예측한 비율입니다. 재현율이 높을수록 실제 양성을 잘 탐지하는 모델입니다.
4. **ROC 곡선과 AUC 값**:
    - ROC 곡선은 모델의 성능을 시각적으로 평가하는 방법입니다. 이 곡선 아래의 면적인 **AUC** 값이 클수록 모델의 분류 성능이 우수함을 나타냅니다.

5.6.1 혼동 행렬 (Confusion Matrix)

혼동 행렬은 분류 모델의 성능을 평가하는 핵심 도구입니다. 이는 모델의 예측 결과와 실제 결과를 비교하여 분류 정확도를 시각적으로 표현합니다. 주로 **이진 분류(binary classification)** 문제에 적용되며, 결과를 네 가지 범주로 구분합니다.

혼동 행렬의 구성 요소

1. **TP (True Positive)**: 실제 **양성** 데이터를 정확히 **양성**으로 예측한 경우.
2. **TN (True Negative)**: 실제 **음성** 데이터를 정확히 **음성**으로 예측한 경우.
3. **FP (False Positive)**: 실제 **음성** 데이터를 잘못 **양성**으로 예측한 경우. 일명 **제1종 오류**.
4. **FN (False Negative)**: 실제 **양성** 데이터를 잘못 **음성**으로 예측한 경우. 일명 **제2종 오류**.

실제 예시: 스팸 메일 분류

스팸 메일 분류기의 성능을 평가하는 혼동 행렬을 살펴보겠습니다. 여기서 "스팸"은 양성(positive), "정상 메일"은 음성(negative)으로 간주합니다:

	예측: 스팸	예측: 정상
실제: 스팸	23 (TP)	12 (FN)
실제: 정상	56 (FP)	556 (TN)

이 혼동 행렬은 다음을 보여줍니다:
• 23개의 실제 스팸 메일을 정확히 식별 (TP)
• 12개의 실제 스팸 메일을 정상으로 오분류 (FN)
• 56개의 정상 메일을 스팸으로 오분류 (FP)
• 556개의 정상 메일을 정확히 식별 (TN)

이러한 결과를 통해 분류기의 정확도, 정밀도, 재현율 등을 계산할 수 있습니다.

혼동 행렬을 이용한 지표 계산

- **정확도(Accuracy)**: 전체 예측 중에서 맞춘 비율을 나타냅니다.

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

- **정밀도(Precision)**: 양성으로 예측한 것들 중 실제 양성인 비율을 나타냅니다.

$$\text{Precision} = \frac{TP}{TP+FP}$$

- **재현율(Recall)**: 실제 양성 중에서 정확히 예측된 비율을 나타냅니다.

$$\text{Recall} = \frac{TP}{TP+FN}$$

- **F1 점수(F1 Score)**: 정밀도와 재현율의 조화 평균으로, 두 지표 사이의 균형을 평가하는 데 유용합니다.

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

5.6.2 정밀도(Precision)와 재현율(Recall)

정밀도와 재현율은 모델의 예측 성능을 평가하는 데 핵심적인 두 지표입니다. 이들은 특히 **양성(positive)** 예측과 밀접한 관련이 있으며, 모델이 실제 양성 데이터를 얼마나 정확하게 식별했는지를 측정하는 중요한 기준으로 활용됩니다.

1. 정밀도(Precision)

- **정밀도(Precision)**는 모델이 양성으로 예측한 것 중에서 실제로 양성인 데이터의 비율을 의미합니다. 즉, 모델이 예측한 양성 중에서 참(True Positive, TP)이 얼마나 많은지를 나타냅니다.
- **정밀도 수식**:

$$\text{Precision} = \frac{TP}{TP+FP}$$

  여기서:
  - **TP (True Positive)**: 실제로 양성인 데이터를 모델이 양성으로 정확히 예측한 경우.
  - **FP (False Positive)**: 실제로는 음성인 데이터를 모델이 잘못 양성으로 예측한 경우.
- **정밀도의 의미**:
  - 정밀도는 모델이 양성으로 예측한 결과 중 실제로 정확한 비율을 나타냅니다.

- 높은 정밀도는 모델이 양성으로 분류한 데이터 중 실제 양성인 경우가 많음을 의미합니다. 예를 들어, 스팸 필터에서 스팸으로 판단한 메일 중 실제 스팸인 비율이 높을수록 정밀도가 높아집니다.

2. 재현율(Recall)

- **재현율(Recall)**은 실제 양성 데이터 중에서 모델이 정확하게 예측한 비율을 의미합니다. 즉, 실제 양성 데이터 중에서 참(True Positive, TP)을 얼마나 많이 찾아냈는지를 나타냅니다.
- **재현율 수식**:

$$\text{Recall} = \frac{TP}{TP+FN}$$

  여기서:
  - **TP (True Positive)**: 실제로 양성인 데이터를 모델이 양성으로 정확히 예측한 경우.
  - **FN (False Negative)**: 실제로는 양성인 데이터를 모델이 잘못 음성으로 예측한 경우.
- **재현율의 의미**:
  - 재현율은 모델이 실제 양성 데이터를 얼마나 정확하게 식별했는지를 평가합니다.
  - 높은 재현율은 모델이 실제 양성 데이터의 대부분을 정확히 예측했음을 의미합니다. 예를 들어, 스팸 필터에서 실제 스팸 메일을 거의 모두 정확히 식별할수록 재현율이 높아집니다.

정밀도와 재현율의 상충 관계

- **정밀도와 재현율은 서로 상충(trade-off) 관계**에 있습니다. 정밀도를 높이면 재현율이 떨어지고, 재현율을 높이면 정밀도가 떨어지는 경향이 있습니다.
- 예를 들어, 모델이 양성 예측에 매우 보수적일 경우(즉, 확실한 경우에만 양성으로 예측), 정밀도는 높아지지만 재현율이 낮아집니다. 반면, 양성 예측을 광범위하게 할 경우(즉, 많은 데이터를 양성으로 예측), 재현율은 높아지지만 정밀도가 낮아집니다.

5.6.3 정확도(Accuracy)

- **정확도(Accuracy)**는 모델이 전체 데이터 중에서 올바르게 예측한 비율을 의미합니다. 이 지표는 **양성(positive)**과 **음성(negative)** 예측을 모두 포함하여 모델의 전반적인 성능을 평가하는 기준입니다.
- **정확도 수식**:

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

여기서:

  - **TP (True Positive)**: 실제로 양성인 데이터를 모델이 양성으로 정확히 예측한 경우.
  - **TN (True Negative)**: 실제로 음성인 데이터를 모델이 음성으로 정확히 예측한 경우.
  - **FP (False Positive)**: 실제로 음성인 데이터를 모델이 잘못 양성으로 예측한 경우.
  - **FN (False Negative)**: 실제로 양성인 데이터를 모델이 잘못 음성으로 예측한 경우.

- **정확도의 의미**:
- 정확도는 모델이 전체 데이터에서 얼마나 정확하게 예측했는지를 나타냅니다.
- 정확도는 모델 성능을 직관적으로 평가할 수 있는 지표입니다. 하지만 불균형한 데이터셋에서는 문제가 발생할 수 있습니다. 예를 들어, 음성 데이터가 압도적으로 많은 경우, 모델이 대부분을 음성으로 예측하더라도 높은 정확도를 기록할 수 있습니다. 따라서 정확도만으로 모델의 성능을 평가하는 것은 한계가 있습니다.

정밀도, 재현율, 정확도의 적용 사례

- **스팸 필터**:
  - **정밀도**가 높은 스팸 필터는 스팸으로 분류한 메일이 실제로 스팸일 가능성이 높습니다. 다만, 일부 스팸 메일을 놓칠 수 있습니다.
  - **재현율**이 높은 스팸 필터는 대부분의 스팸 메일을 효과적으로 식별하지만, 일부 정상 메일을 스팸으로 잘못 분류할 위험이 있습니다.
- **질병 진단**:
  - **재현율**은 질병 진단에서 중요합니다. 실제 질병이 있는 환자를 놓치지 않고 진단하는 것이 핵심이므로, 높은 재현율을 가진 모델이 필요합니다.
  - **정밀도**는 양성 예측의 오류가 심각한 결과를 초래할 수 있는 상황에서 중요합니다. 예를 들어, 무고한 사람을 범죄자로 오판하는 경우, 높은 정밀도가 필수적입니다.

5.6.5 AUC (Area Under the Curve)

AUC는 **ROC 곡선(Receiver Operating Characteristic curve)** 아래의 면적을 의미하며, 분류 모델의 성능을 평가하는 데 자주 사용됩니다. AUC는 0에서 1 사이의 값을 가지며, 값이 클수록 모델의 성능이 뛰어나다는 것을 나타냅니다.

1. ROC 곡선

ROC 곡선은 **참 양성 비율(TPR: True Positive Rate)**과 **거짓 양성 비율(FPR: False Positive Rate)**을 비교하여 모델의 성능을 시각적으로 평가하는 그래프입니다. 이 곡선은 모델의 예측 결과에 따라 다양한 임곗값(threshold)을 설정하면서 TPR과 FPR의 변화 양상을 보여줍니다.

- **참 양성 비율(TPR)**: 실제 양성 중에서 모델이 양성으로 정확히 예측한 비율입니다. 이는 재현율(Recall)과 같은 의미입니다.

  $$\text{TPR} = \frac{TP}{TP+FN}$$

  여기서:

  - **TP (True Positive)**: 실제 양성인 데이터를 모델이 양성으로 예측한 경우.
  - **FN (False Negative)**: 실제 양성인 데이터를 모델이 음성으로 잘못 예측한 경우.
- **거짓 양성 비율(FPR)**: 실제 음성 중에서 모델이 양성으로 잘못 예측한 비율입니다.

  $$\text{FPR} = \frac{FP}{FP+TN}$$

  여기서:

  - **FP (False Positive)**: 실제 음성인 데이터를 모델이 양성으로 잘못 예측한 경우.
  - **TN (True Negative)**: 실제 음성인 데이터를 모델이 음성으로 정확히 예측한 경우.

2. ROC 곡선 해석

- **완벽한 모델**은 ROC 곡선에서 좌측 상단 모서리에 가까운 경로를 그립니다. 이 경우 AUC 값은 1에 가까워지며, 거짓 양성 비율이 0이고 참 양성 비율이 1인 이상적인 상태를 의미합니다.
- **무작위 예측 모델**의 ROC 곡선은 대각선(45도 선)을 따라갑니다. 이때 AUC 값은 0.5로, 모델이 무작위로 예측하고 있음을 나타냅니다.
- **나쁜 모델**의 ROC 곡선은 대각선 아래에 위치할 수 있습니다. AUC 값이 0.5보다 작으면 모델의 성능이 무작위 예측보다 못할 수 있음을 시사합니다.

3. AUC 값의 의미

- **AUC = 0.5**: 모델이 무작위 예측과 동일한 성능을 보입니다.
- **AUC = 1**: 모델이 완벽하게 모든 양성과 음성을 구분합니다.
- **AUC 값이 0.7 이상**: 일반적으로 모델이 신뢰할 만한 성능을 보인다고 평가됩니다.

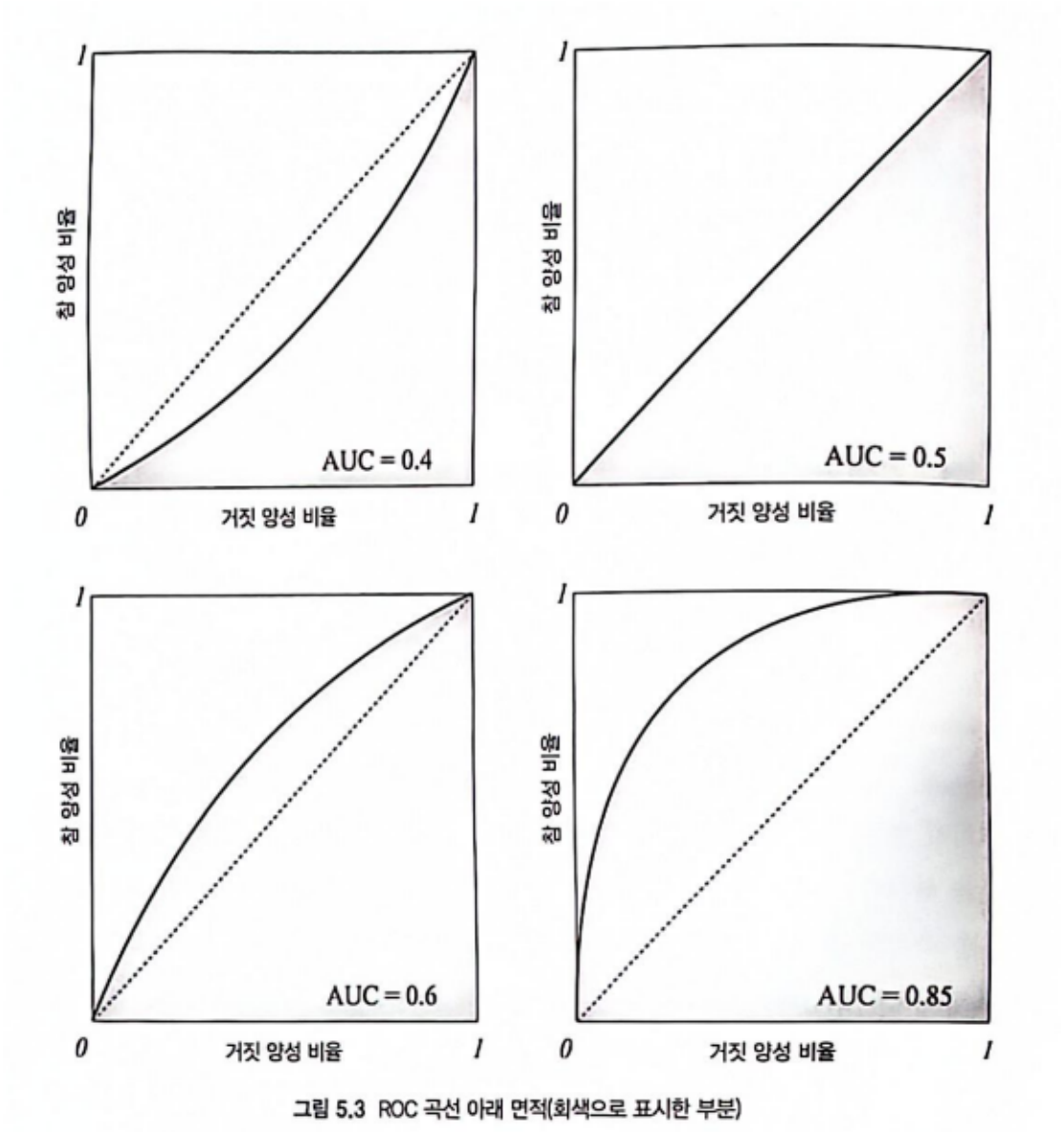

그림 5.3 ROC 곡선 아래 면적(회색으로 표시한 부분)

그림 5.3은 다양한 AUC 값에 따른 ROC 곡선을 보여줍니다. 이를 통해 모델 성능의 변화를 시각적으로 확인할 수 있습니다.

- **AUC = 0.4**: 모델의 성능이 무작위 예측보다 낮습니다.
- **AUC = 0.5**: 모델이 무작위 예측과 동일한 성능을 보입니다.
- **AUC = 0.6**: 모델이 어느 정도 유용하지만, 뛰어난 성능은 아닙니다.
- **AUC = 0.85**: 모델이 상당히 우수한 성능을 보이며, 양성과 음성을 잘 구분합니다.

5.6.4 비용 민감 정확도 (Cost-Sensitive Accuracy)

모든 분류 문제에서 오류의 비용이 동일한 것은 아닙니다. 특정 예측 오류가 더 큰 비용을 초래할 수 있습니다. 이런 경우 **비용 민감 정확도(cost-sensitive accuracy)**를 고려해야 합

니다. 이는 양성 예측과 음성 예측의 오류 비용을 다르게 설정하여 모델을 평가하는 방식입니다.

비용 민감 정확도 계산 방법

1. **FN (False Negative)**와 **FP (False Positive)**에 각각 비용을 설정합니다. 예를 들어, 잘못된 음성 예측(FN)이 잘못된 양성 예측(FP)보다 더 큰 비용을 유발한다면, FN에 더 높은 비용을 부여합니다.
2. FN과 FP의 오류 비용을 포함한 정확도를 계산하여 모델의 성능을 평가합니다.

이 방식은 비용 구조가 다른 상황에서 모델의 실제 성능을 평가하는 데 유용합니다.

5.7 하이퍼파라미터 튜닝

하이퍼파라미터란?

- **하이퍼파라미터**는 모델 학습 전에 설정하는 변수로, 데이터 학습 과정에 중요한 영향을 미칩니다.
    - 예시: SVM(Support Vector Machine) 모델의 **C 값**과 **감마(γ)**가 대표적입니다. 신경망(Neural Network)에서는 학습률(learning rate)과 은닉층의 수 등이 하이퍼파라미터에 해당합니다.
- 이러한 하이퍼파라미터는 모델이 자동으로 결정하지 않으며, 사용자가 사전에 설정해야 합니다. 적절한 설정 없이는 모델 성능이 크게 저하될 수 있습니다.

하이퍼파라미터 튜닝의 필요성

- 모델의 최적 성능을 위해서는 적절한 하이퍼파라미터 값을 찾아야 합니다.
- 하이퍼파라미터 값이 모델 성능에 큰 영향을 미치므로, 이를 최적화하는 과정은 필수적입니다.

하이퍼파라미터 튜닝 방법

1. 그리드 탐색(Grid Search)

- **그리드 탐색(Grid Search)**은 하이퍼파라미터 튜닝의 가장 일반적인 방법 중 하나입니다.
- 이 방법은 미리 정의된 하이퍼파라미터의 범위 내에서 가능한 모든 조합을 시도하여 최적의 하이퍼파라미터 조합을 찾습니다.
    - 예를 들어, **SVM 모델**에서 **C 값**과 **감마(γ) 값**을 조정할 때, C 값으로 0.001, 0.01, 0.1, 1, 10 등을 시도하고, 감마 값도 다양하게 실험하여 모든 가능한 조합을 탐색합

니다.

- **장점**: 최적의 하이퍼파라미터 조합을 찾을 수 있습니다.
- **단점**: 모든 조합을 시도해야 하므로 계산량이 많고 시간이 오래 걸립니다.

2. 랜덤 탐색(Random Search)

- **랜덤 탐색(Random Search)**은 그리드 탐색의 대안으로, 미리 정의된 범위 내에서 하이퍼파라미터 조합을 무작위로 선택해 성능을 평가합니다.
  - 그리드 탐색과 달리 모든 조합을 시도하지 않고, 무작위로 선택한 일부 조합만으로 실험을 진행합니다.
- **장점**: 계산량이 줄어들어 탐색 시간을 단축할 수 있습니다.
- **단점**: 모든 조합을 시도하지 않아 최적의 하이퍼파라미터 조합을 놓칠 가능성이 있습니다.

3. 베이지안 최적화(Bayesian Optimization)

- **베이지안 최적화**는 과거 시도 결과를 바탕으로 최적의 하이퍼파라미터 값을 찾아가는 방식입니다.
  - 기존의 그리드 탐색이나 랜덤 탐색보다 **더 적은 실험**으로도 최적의 값을 찾을 수 있습니다.
  - 이 방법은 **가능성 있는 하이퍼파라미터 조합**을 먼저 시도하고, 그 결과를 바탕으로 이후 실험에서 더욱 최적화된 값들을 시도합니다.
- **장점**: 그리드 탐색이나 랜덤 탐색보다 효율적이며, 빠르게 최적의 값을 찾을 수 있습니다.
- **단점**: 설정과 이해가 상대적으로 복잡하며, 초기 세팅에 많은 시간과 노력이 필요할 수 있습니다.

하이퍼파라미터 튜닝의 실전 적용

SVM을 예로 든 튜닝 과정

- **SVM 모델**의 주요 하이퍼파라미터는 **C 값**과 **감마(γ)** 값입니다.
  - **C 값**은 모델의 **오차 허용 범위**를 조정하는 하이퍼파라미터입니다. C 값이 크면 모델은 훈련 데이터에 더 정확히 맞추려 하고, 작으면 더 일반화된 모델이 됩니다.
  - **감마 값(γ)**은 데이터 포인트 간 거리에 따른 중요도를 조정합니다. 감마 값이 크면 모델은 훈련 데이터에 더 민감해지지만, 너무 크면 과적합(overfitting)이 발생할 수 있습니다.

- 예를 들어, **C 값**과 **감마 값**에 대해 0.001, 0.01, 0.1, 1, 10, 100, 1000 등의 범위를 설정한 뒤, 그리드 탐색이나 랜덤 탐색으로 가능한 조합들을 실험하여 최적의 조합을 찾습니다.

결론

- **하이퍼파라미터 튜닝**은 모델 성능 최적화를 위해 필수적입니다. 적절한 하이퍼파라미터 값으로 모델의 예측 능력을 극대화하는 것이 목표입니다.
- **그리드 탐색**과 **랜덤 탐색**은 가장 일반적인 하이퍼파라미터 튜닝 기법으로, 각각 장단점이 있습니다.
  - **그리드 탐색**은 모든 조합을 시도하지만, 시간이 많이 소요됩니다.
  - **랜덤 탐색**은 시간은 단축되지만, 최적의 조합을 놓칠 가능성이 있습니다.
- **베이지안 최적화**는 그리드 탐색과 랜덤 탐색보다 효율적으로, 적은 실험으로 최적의 하이퍼파라미터 값을 찾는 데 유리합니다.

5.7.1 교차 검증 (Cross-validation)

교차 검증의 개념

- 교차 검증(Cross-validation)은 모델의 성능을 평가할 때 훈련 데이터의 일부를 검증 데이터로 활용하는 기법입니다.
- 이 방법은 훈련 데이터 전체를 한 번에 사용하지 않고, 여러 부분으로 나누어 모델을 훈련하고 평가합니다. 특히 데이터가 제한적일 때 과적합을 방지하고 모델의 일반화 성능을 효과적으로 평가할 수 있습니다.

교차 검증의 필요성

- 하이퍼파라미터 튜닝 시 훈련 데이터만을 사용하여 모델을 평가하면 과적합이 발생할 수 있습니다. 이는 모델이 훈련 데이터에만 과도하게 최적화되어 새로운 데이터에 대한 성능이 저하될 수 있음을 의미합니다.
- 교차 검증은 이러한 문제를 해결하고 모델의 **일반화 성능**을 평가하는 데 사용됩니다. 데이터의 일부를 훈련에, 나머지를 검증에 번갈아 사용함으로써 모델이 훈련 데이터에 과적합되는 것을 방지합니다.

교차 검증 방법

가장 널리 사용되는 교차 검증 방식은 k-폴드 교차 검증(k-fold cross-validation)입니다. 여기서 **k**는 데이터셋을 나누는 폴드(fold)의 수를 의미하며, 일반적으로 5폴드 또는 10폴드를 주로 사용합니다.

1. k-폴드 교차 검증

k-폴드 교차 검증은 데이터셋을 **k개의 폴드**로 나눈 후, 각 폴드를 한 번씩 검증 집합으로 사용하고 나머지 폴드를 훈련에 활용하는 방식입니다.

예를 들어, **5-폴드 교차 검증**을 수행하는 경우:

- 데이터셋을 5개의 부분으로 나눕니다. 각 부분을 F_1, F_2, F_3, F_4, F_5라고 가정합니다.
- 첫 번째 폴드 F_1을 검증 집합으로, 나머지 F_2, F_3, F_4, F_5를 훈련 집합으로 사용해 모델을 학습시킵니다.
- 다음으로, 두 번째 폴드 F_2를 검증 집합으로, 나머지 F_1, F_3, F_4, F_5를 훈련 집합으로 사용하여 다시 학습합니다.
- 이 과정을 F_1부터 F_5까지 각 폴드를 한 번씩 검증 집합으로 사용하며, 총 5개의 모델을 학습하고 검증합니다.
- 최종적으로, 각 폴드에서 얻은 성능 결과의 평균을 계산하여 모델의 전체 성능을 평가합니다.

2. 교차 검증의 과정

1. 데이터셋을 **k개의 부분**으로 분할합니다.
2. 각 부분을 순차적으로 검증 데이터로 사용하고, 나머지 부분을 훈련 데이터로 활용하여 모델을 학습합니다.
3. 모든 검증 결과를 기록하고, 이들의 평균을 최종 평가 결과로 사용합니다.

장점:

- 모든 데이터가 훈련과 검증에 각각 한 번씩 사용되므로, 데이터가 제한적일 때 특히 유용합니다.
- 모델의 일반화 성능을 더 정확하고 신뢰성 있게 평가할 수 있습니다.

단점:

- 폴드 수가 증가할수록 학습과 검증을 반복해야 하므로, 계산 시간이 크게 늘어날 수 있습니다.

하이퍼파라미터 튜닝과 교차 검증

하이퍼파라미터 튜닝 과정에서의 교차 검증

하이퍼파라미터 튜닝은 모델의 학습에 필요한 파라미터 값을 최적화하는 과정입니다. 이 과정에서 교차 검증을 활용하면 과적합을 방지하고 더 나은 하이퍼파라미터 값을 찾을 수 있습

니다.

- 예를 들어, **SVM 모델**에서 **C 값**과 **감마 값**을 튜닝할 때 교차 검증을 통해 각 하이퍼파라미터 조합의 성능을 평가할 수 있습니다.
- **그리드 탐색(Grid Search)** 또는 **랜덤 탐색(Random Search)**과 같은 튜닝 기법을 사용할 때, 교차 검증으로 각 조합의 성능을 평가하고 최적의 하이퍼파라미터 조합을 선택할 수 있습니다.

최종 평가

- 하이퍼파라미터 튜닝 완료 후, 선택된 최적의 하이퍼파라미터를 적용하여 **테스트 집합**에서 모델을 최종 평가합니다.
- 이 과정을 통해 튜닝 중 모델이 과적합되지 않았는지, 그리고 새로운 데이터에 대해서도 잘 작동하는지 확인할 수 있습니다.

결론

- **교차 검증**은 데이터가 적거나 모델이 과적합될 가능성이 있을 때, 모델의 성능을 더 정확히 평가하는 강력한 방법입니다.
- **k-폴드 교차 검증**은 데이터를 여러 부분으로 나누어 각 부분을 번갈아 검증 데이터로 사용함으로써, 훈련 데이터의 과적합을 방지하고 일반화 성능을 높이는 데 매우 유용합니다.
- **하이퍼파라미터 튜닝** 과정에서 교차 검증을 활용하면, 다양한 하이퍼파라미터 조합에 대한 모델의 성능을 평가하고 최적의 조합을 찾을 수 있습니다.