**1. 결정트리란?**

결정트리는 속성 기반 규칙의 조합을 통해 데이터를 분류하거나 회귀하는 지도학습 모델이다. 트리 구조로 구성되며, 각 노드는 데이터를 조건에 따라 분할하고, 최종 리프 노드(끝마디)는 예측 결과를 나타낸다.

* 분류(Classification): 리프 노드에서 가장 높은 빈도수를 가진 범주를 예측
* 회귀(Regression): 리프 노드의 종속변수 평균을 예측값으로 사용

**2. 결정트리 구조**

|  |  |
| --- | --- |
| Root Node | 트리의 시작점, 전체 데이터 |
| Parent/Child Node | 상하위 분기 노드 |
| Leaf Node | 분할의 마지막, 예측 결과가 나오는 곳 |
| Depth | 트리의 깊이 (분할 단계 수) |

트리는 입력 변수에 따라 반복적으로 데이터를 나누며, 각 분기점에서 최적의 분할 기준을 찾는다. 트리의 구조는 매우 직관적이며 사람이 해석하기 용이하다.

**3. 결정트리 수행과정**

1. **트리 형성**:
   * 입력 변수의 값을 기준으로 데이터 분할
   * 목적은 자식 노드의 순수도(purity)를 최대화 (또는 불순도 감소)
2. **가지치기(Pruning)**:
   * 과도하게 분기된 트리를 단순화하여 과적합 방지
   * 불필요하거나 오류를 유발할 수 있는 가지 제거
3. **타당성 평가 및 예측**:
   * 테스트 데이터를 이용해 성능 확인
   * 예측 결과를 실제 결과와 비교하여 평가

**4. CART 알고리즘 (Classification and Regression Tree)**

* 결정트리는 CART 알고리즘을 통해 학습된다.
* 각 노드는 \*\*비용 함수(cost function)\*\*를 최소화하는 방향으로 데이터를 분할
* 대표적인 비용 함수:
  + 지니 불순도(Gini Impurity)
  + 엔트로피(Entropy)

불순도가 작을수록 노드가 "더 순수"하다는 의미이며, 분할 기준의 선택은 이러한 순수도를 기준으로 한다.

**5. 지니 불순도 vs. 엔트로피**

실제로 두 방법 간의 성능 차이는 크지 않으며, 일반적으로는 지니 불순도가 기본값으로 사용된다.

|  |  |
| --- | --- |
| **지니 불순도** | **엔트로피** |
| 다른 클래스로 분류될 확률 | 샘플의 무질서 정도 |
| 계산 속도 빠름 (기본값) | 분할 균형이 좋음 |
| criterion="gini" (기본) | criterion="entropy" (옵션) |

**6. 결정트리 규제**

결정트리는 비파라미터 모델로서 데이터에 대한 가정을 하지 않으며, 노드마다 파라미터가 새롭게 학습된다. 이로 인해 자유도가 높고, 과적합 위험이 큼.

과적합 방지를 위한 규제 하이퍼파라미터 예시:

|  |  |
| --- | --- |
| **파라미터** | **설명** |
| max\_depth | 트리 최대 깊이 제한 |
| min\_samples\_split | 분할을 위한 최소 샘플 수 |
| min\_samples\_leaf | 리프 노드에 있어야 할 최소 샘플 수 |

적절한 규제를 적용하면 모델의 일반화 성능을 높일 수 있다.

**7. 회귀 결정트리**

* DecisionTreeRegressor 클래스를 활용
* 회귀 문제에서도 트리 기반으로 데이터를 분할
* 리프 노드의 평균값이 예측값이 됨
* 과적합 시에는 분류 문제와 마찬가지로 규제가 필요함

예: 잡음이 있는 2차 함수 데이터를 회귀 결정트리로 모델링 가능

**8. 결정트리의 단점**

1. **회전 민감성**:  
   결정트리는 항상 축에 수직인 경계를 기준으로 분할한다. 데이터가 회전되면 성능이 저하될 수 있음.
2. **높은 분산(Variance)**:  
   훈련 데이터의 작은 변화에도 예측 결과가 크게 달라지는 경향이 있음. 이를 해결하기 위해서는 \*\*앙상블 기법(예: 랜덤 포레스트)\*\*이 사용된다.

**9. 구현과 실습**

* **Scikit-learn 활용**
  + DecisionTreeClassifier, DecisionTreeRegressor
  + 시각화: export\_graphviz()
  + 예측 확률: predict\_proba()
* **예제**: 붓꽃 데이터셋
  + 꽃잎의 길이/너비를 기준으로 분류
  + max\_depth 설정에 따라 예측 성능 및 복잡도 변화