결정트리모델

- 입력값과 출력값의 관계를 나무 모양으로 모델링하는 지도학습 기법

- 분류 트리와 회귀 트리로 구분

주요 알고리즘

- ID3 : 분류문제에만 적용 가능. 모든 피처는 범주형이어야 함.

    - 분기 규칙 : 정보 이득 최대화

- C4.5 : ID3에서 발전됨.

    - 피처 범주형 제약 사라짐(숫자형 변수 여러 구간으로 분할_

    - 학습한 트리를 if-then 규칙의 집합으로 변환하는 과정 추가

    - 정확도 계산함. 전체 정확도가 증가하는 방향으로 일부 규칙 제거.

    - 분기 규칙 : 정보 이득 최대화

- CART : C4.5와 비슷하나 회귀모델이 추가됨. if-then 규칙 집합도 제공 안 함.

    - 분류 : 지니 불순도 최소화

    - 회귀 : MSE 최소화

- CHAID : 통계 검정 기반으로 분기 수행

    - 분류 : 카이제곱 검정

    - 회귀 : F 검정

- 장점: 이해하기 쉽고 시각화하기 쉽습니다. 특별한 전처리가 필요하지 않으며, 범주형 변수를 다루기에 용이합니다.

- 단점: 과적합(Overfitting)되기 쉽고, 데이터의 작은 변화에 민감할 수 있습니다.

파라미터	파라미터 설명	입력값
criterion	분기 규칙 선택	default= gini, {gini,entropy}
max_depth	트리 깊이 상한선(1 이상의 정수)	default=None , {None,int}
min_samples_split	노드에서 분기를 진행하는 최소한의 샘플 숫자 선택 (	default= 2, {int,float}

min_samples_leaf	리프 노드에 있을 샘플 개수 최소값 선택. 주로 과적합 방지 용도.	default = 1 ,{int,flaot}
max_features	각 노드에서 분기 확인을 위한 피처 수.	default=None {None,int,float,sqrt,log2}
random_state	랜덤성 제어	default=None , {int}
ccp_alpha	클래스 라벨별 가중치 설정	default=0
class_weight	가지치기 알고리즘의 복잡도 하이퍼파라미터. 과적합 방지용	default=None , {None,dict,balanced}

실제 활용 팁

- 결정 트리는 과적합되기 쉬우므로 과도하게 많은 피처를 학습에 사용하지 않도록 유의한다.

- 트리의 크기를 제어한 max_depth를 기본값 그대로 둘 경우 트리의 크기가 매우 커질 수 있다.

랜덤포레스트모델

-랜덤 포레스트는 여러 개의 결정 트리를 생성하고, 각 트리의 예측을 기반으로 최종 예측을 만듭니다. 각 트리는 부트스트랩 샘플(중복이 허용된 랜덤 샘플)을 사용하고, 각 노드에서 랜덤한 일부 특성을 고려하여 트리를 구성합니다.

- 부트스트래핑: 단순 복원 임의추출법(랜덤샘플링)으로 크기가 동일한 여러 개의 표본자료를 생성한다.

- 배깅: 여러 부트스트랩 자료를 생성하여 학습하는 모델링으로 분류기를 생성한 후 그 결과를 앙상블 하는 방법이다.

- 앙상블 학습: 여러 모델을 학습시켜 결합하는 방식의 학습방법으로 일반화 성능을 향상시켜 과적합을 해결할 수 있음.

- 장점: 과적합을 줄이고 안정적인 예측을 제공합니다. 다양한 특성을 고려하므로 다양성이 높습니다.

- 단점: 시각화가 어려우며, 특정 문제에서는 성능이 떨어질 수 있습니다.

파라미터	파라미터 설명	입력값
n_estimators	트리 개수	default=100 , {int}
bootstrap	부트스트랩 샘플 사용 여부	default=squared_error , {squared_error, absolute_error, poisson}
ood_score	bootstrap=True 일 때만 유효하며, 일반화 점수를 계산할 때 OOB샘플을 사용할지 선택	default=True, {bool}

max_samples	bootstrap=True 일 때만 유효하며, 각각의 베이스 학습기를 학습하기 위해 추출할 샘플 개수를 지정	
Criterion(회귀)	분기 기준	

실제 활용 팁

- max_depth등의 하이퍼파라미터는 트리 크기 등의 복잡도를 늘리므로 이들을 적절한 값으로 설정하여 모델을 제어

- criterion을 absoulte_error로 하면 squared_error보다 느려짐

GBT(그래이디언트부스팅트리모델)

-그레이디언트 부스팅은 약한 학습자(약한 결정 트리)를 순차적으로 학습시켜 강력한 예측 모델을 만듭니다. 이전 트리의 오차를 보완하는 방식으로 학습이 진행됩니다.

1. GBT 회귀

- 직렬로 학습하는 가산 모델

- 그리디 알고리즘 방식으로 진행

- 손실함수는 제곱 오차 손실 또는 절대 오차 손실로 정의된다.

2. GBT 분류

- GBT 회귀 모델에서의 예측값 $F_m(x_i) = ₩sum h_m(x_i)$를 클래스 또는 클래스의 확률 값으로 변환하는 과정을 추가로 진행해야 한다.

- 변환 방법은 손실 함수에 따라 달라진다.

- 약학 학습기 h_m은 분류 모델이 아니라 여전히 회귀 모델이라는 사실을 명심해야 한다.

    - 이는 예측하고자 하는 값이 연속값인 그레이디언트기 때문!

- 장점: 높은 예측 성능을 보이며, 이상치에 강한 모델을 생성합니다. 다양한 손실 함수를 사용할 수 있어 다양한 문제에 적용 가능합니다.

- 단점: 학습 속도가 느리고, 하이퍼파라미터 튜닝이 필요합니다.

파라미터	파라미터 설명	입력값
loss	손실 함수 선택	default=squared_error, {squared_error, absolute_error, huber,quantile}
learning_rate	모든 규제를 위해 각 부스팅 단계에 적용하는 학습률	default=0.1 {float}
n_estimators	부스팅 단계의 수	default=100, {int}
subsample	모델의 규제를 위해 각 트리를 만들 때 전체 데이터 대신 subsample의 비율만을 랜덤 샘플링 하여 사용.	default=1.0. {float}

- 조기 종료 로직

파라미터	파라미터 설명	입력값
validation_fraction	조기 종료를 위해 사용할 검증 데이터셋 비율	
n_iter_no_change	조기 종료를 적용하고자 모니터링할 이터레이션의 횟수	
tol	조기 종료를 위한 허용 오차	

실제 활용 팁

- 모델 복잡도가 증가하는 하이퍼파라미터 n_estimators와 모델을 규제하는 learning_rate,subsamples, validation_fraction, n_inter_no_change,tol등을 상충 관계

- GBT 모델에서 가장 중요한 것은 n_estimators와 learning_rate를 먼저 고려