**Chapter 04. 분류**

**04-01. 분류의 개요**

지도학습: 레이블(Label), 명시적인 정답이 있는 데이터가 주어진 상태에서 학습하는 머신러닝 방식

지도학습의 대표적 유형 – 분류(Classification)

학습 데이터로 주어진 데이터의 피처와 레이블값을 머신러닝 알고리즘으로 학습해 모델을 생성하고, 이렇게 생성된 모델에 새로운 데이터 값이 주어졌을 때 미지의 레이블 값을 예측하는 것

즉, 기존 데이터가 어떤 레이블에 속하는지 패턴을 알고리즘으로 인지한 뒤에 새롭게 관측된 데이터에 대한 레이블을 판별하는 것

앙상블

: 기본 알고리즘 🡪 결정 트리

-배깅

* 대표적인 예: 랜덤 포레스트

뛰어난 예측 성능, 상대적으로 빠른 수행 시간, 유연성 등으로 많은 분석가가 애용

* 그래디언트 부스팅

뛰어난 예측 성능, 수행 시간이 너무 오래 걸림

* XgBoost와 LightGBM

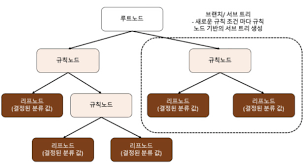
기존 그래디언트 부스팅의 예측 성능을 한단계 발전시키면서 수행 시간을 단축

-부스팅

**04-02. 결정 트리**

:ML 알고리즘 중 직관적으로 이해하기 쉬운 알고리즘

데이터에 있는 규칙을 학습을 통해 자동으로 찾아내 트리 기반의 분류 규칙을 만드는 것



규칙 노드: 규칙 조건

리프 노드: 결정된 클래스 값

서브 노드: 새로운 규칙 조건마다 생성되는 것

* 트리의 깊이가 깊어질수록 결정 트리의 예측 성능이 저하될 가능성이 높음
* 가능한 적은 결정 노드로 높은 예측 정확도를 가지려면 데이터를 분류할 때 최대한 많은 데이터 세트가 해당 분류에 속할 수 있도록 결정 노드의 규칙을 정해야 함
* 최대한 균일한 데이터 세트를 구성할 수 있도록 분할하는 것이 필요
* 장점: 정보의 균일도라는 룰을 기반으로 하여 알고리즘이 쉽고 직관적
* 단점: 과적합, 정확도가 떨어짐

결정 트리 파라미터

| **파라미터 명** | **설명** |
| --- | --- |
| **min\_samples\_split** | - 노드를 분할하기 위한 최소한의 샘플 데이터수 → 과적합을 제어하는데 사용 - Default = 2 → 작게 설정할 수록 분할 노드가 많아져 과적합 가능성 증가 |
| **min\_samples\_leaf** | - 리프노드가 되기 위해 필요한 최소한의 샘플 데이터수 - min\_samples\_split과 함께 과적합 제어 용도 - 불균형 데이터의 경우 특정 클래스의 데이터가 극도로 작을 수 있으므로 작게 설정 필요 |
| **max\_features** | - 최적의 분할을 위해 고려할 최대 feature 개수 - Default = None → 데이터 세트의 모든 피처를 사용 - int형으로 지정 →피처 갯수 / float형으로 지정 →비중 - sqrt 또는 auto : 전체 피처 중 √(피처개수) 만큼 선정 - log : 전체 피처 중 log2(전체 피처 개수) 만큼 선정 |
| **max\_depth** | - 트리의 최대 깊이 - default = None → 완벽하게 클래스 값이 결정될 때 까지 분할 또는 데이터 개수가 min\_samples\_split보다 작아질 때까지 분할 - 깊이가 깊어지면 과적합될 수 있으므로 적절히 제어 필요 |
| **max\_leaf\_nodes** | 리프노드의 최대 개수 |

결정 트리 모델의 식가화

Graphviz 패키지를 이용하여 시각화 – export\_graphviz() API 사용

예) 붓꽃 데이터 세트를 이용한 규칙 트리 생성과정 탐구

**from** **sklearn.tree** **import** DecisionTreeClassifier

**from** **sklearn.datasets** **import** load\_iris

**from** **sklearn.model\_selection** **import** train\_test\_split

**import** **warnings**

warnings.filterwarnings('ignore')

*# DecicionTreeClassifier 생성*

dt\_clf = DecisionTreeClassifier(random\_state=156)

*# 붓꽃 데이터를 로딩하고, 학습과 테스트 데이터 세트로 분리*

iris\_data = load\_iris()

X\_train, X\_test, y\_train, y\_test = train\_test\_split(iris\_data.data, iris\_data.target, test\_size=0.2, random\_state=11)

*# DecisionTreeClassifier 학습*

dt\_clf.fit(X\_train, y\_train)

**from** **sklearn.tree** **import** export\_graphviz

*# export\_graphviz( )의 호출 결과로 out\_file로 지정된 tree.dot 파일을 생성함*

export\_graphviz(dt\_clf, out\_file="tree.dot", class\_names = iris\_data.target\_names,

feature\_names = iris\_data.feature\_names, impurity=**True**, filled=**True**)

**import** **graphviz**

*# 위에서 생성된 tree.dot 파일을 Graphiviz 가 읽어서 시각화*

**with** open("tree.dot") **as** f:

dot\_graph = f.read()

graphviz.Source(dot\_graph)

텍스트, 스크린샷, 도표, 친필이(가) 표시된 사진

자동 생성된 설명

* 리프 노드: 더 이사 자식 노드가 없는 노드, 최종 클래스 값이 결정되는 노드
* 결정 트리는 규칙 생성 로직을 미리 제어하지 않으면 완벽하게 클래스 값을 구별해내기 위해 트리노드를 계속해서 만들어가기 때문에 모델이 과적합되기 쉬워진다
* 어떠한 속성을 규칙 조건으로 선택하느냐가 중요한 요건

Feature\_importances\_ 을 이용하여 피처의 중요한 역할 지표를 바로 판단 가능

결정 트리 과적합

이상치 데이터까지 분류하기 위해 분할이 자주 일어나 결정 기준 경계가 많아지고 이러한 모델은 학습 데이터 세트의 특성과 약간만 다른 형태의 데이터 세트를 예측하면 예측 정확도가 떨어진다

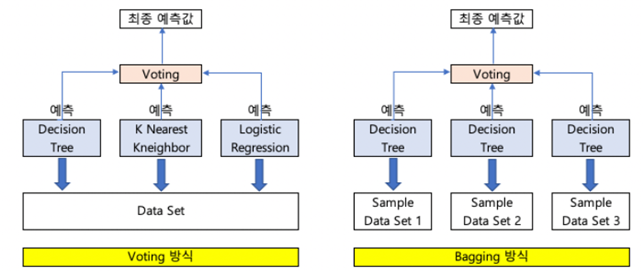
**04-03. 앙상블 학습**

앙상블 학습을 통한 분류: 여러 개의 분류기를 생성하고 그 예측을 결합함으로써 보다 정확한 최종 예측을 도출하는 기법

최종 목표는 다양한 분류기의 예측 결과를 결합함으로써 단일 분류기보다 신뢰성이 높은 예측값을 얻는 것

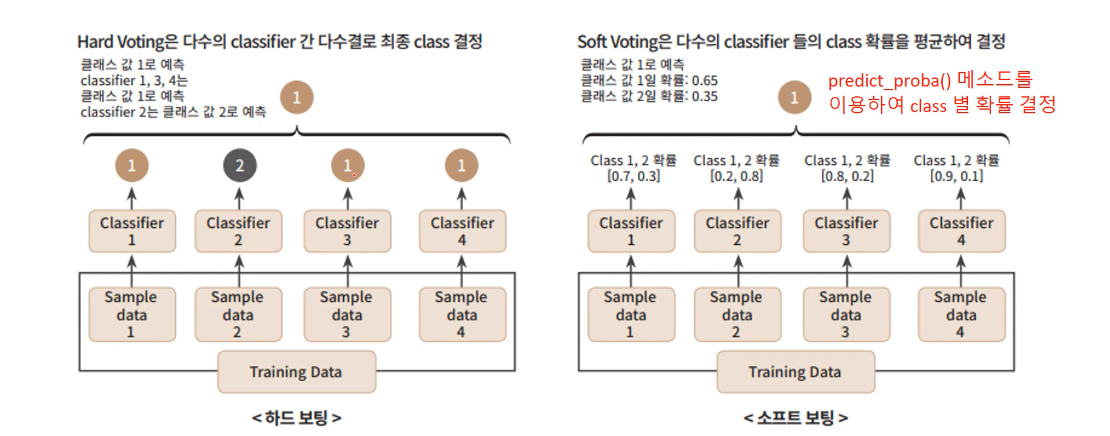
앙상블 학습의 유형

* 보팅: 서로 다른 알고리즘을 가진 분류기를 결합하여 투표를 통해 최종 예측 결과를 결정
* 배깅: 같은 유형의 알고리즘 기반이지만 데이터 샘플링을 다르게 가져가 학습을 수행한 분류기가 투표를 통해 최종 예측 결과를 결정 예) 랜덤 포레스트 알고리즘
* 부스팅: 여러 개의 분류기가 순차적으로 학습을 수행하되 앞에서 학습한 분류기가 예측이 틀린 데이터에 대해서는 올바르게 예측할 수 있도록 다음 분류기에 가중치를 부여하며 학습과 예측을 진행하는 것
* 스태깅: 여러 가지 다른 모델의 예측 결괏값을 다시 학습 데이터로 만들어서 다른 모델로 재학습시켜 결과를 예측하는 방법



보팅 유형 – 하드 보팅과 소프트 보팅

* 하드 보팅: 예측한 결괏값들 중 다수의 분류기가 결정한 예측값을 최종 보팅 결과값으로 선정
* 소프트 보팅: 분류기들의 레이블 값 결정 확률을 모두 더하고 이를 평균해서 이들 중 확률이 가장 높은 레이블 값을 최종 보팅 결과값으로 선정



보팅 분류기

사이킷런에서 VotingClassifier 클래스 제공

텍스트, 스크린샷, 폰트, 도표이(가) 표시된 사진

자동 생성된 설명

**04-04. 랜덤 포레스트**

* 배깅의 대표적인 알고리즘.
* 앙상블 알고리즘 중 비교적 빠른 수행 속도를 가지며 다양한 영역에서 높은 예측 성능을 보임.
* 결정 트리 알고리즘을 기반으로하여 결정 트리의 쉽고 직관적인 장점을 가지고 있음.
* 여러 개의 결정 트리 분류기가 전체 데이터에서 배깅 방식으로 각자의 데이터를 샘플링해 개별적으로 학습을 수행한 뒤 최종적으로 모든 분류기가 보팅을 통해 예측 결정을 함.

랜덤 포레스트 하이퍼 파라미터 및 튜닝

GridSearchCV를 이용하여 랜덤 포레스트의 하이퍼 파라미터를 튜닝

|  |  |
| --- | --- |
| **이름** | **내용** |
| n\_estimators | 랜덤 포레스트에서 결정 트리의 개수를 지정한다. 디폴트는 10개이며 많이 설정할 수록 좋은 성능을 기대할 수 있지만, 계속 증가시킨다고 무조건 향상되는것은 아니다 , 또한 증가에 따른 학습 수행시간 역시 오래걸리게 됨을 감안해야한다.(적을경우 과소적합) |
| max\_features | 결정트리에 사용되는 max\_features와 같은것이다. 현재 모델에서는 디폴트 값이 'auto'이다. 즉, 'sqrt'와 같다.(피쳐가 16개라면 4개 참조/ 줄일 수록 다양한 트리 생성) |
| max\_depth | 노드의 깊이이다. 너무깊어지게 되면 과적합이 일어난다. |
| class\_weight | 불균형(imbalanced)클래스인 경우 사용 |
| min\_samples\_leaf | 결정트리와 같다(과적합일 경우 높이면 된다 |