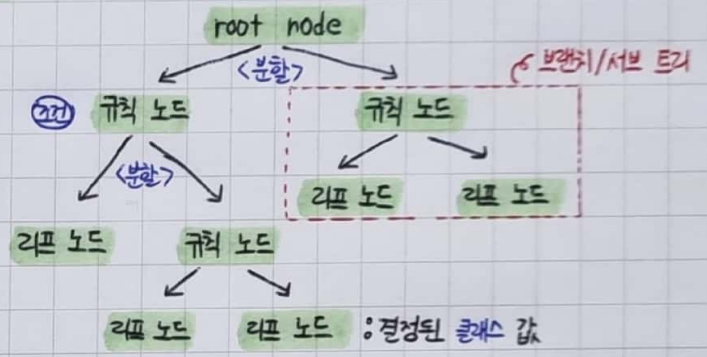
♥ TITLE : 4_분류 ♥ DATE : 2022.09.01

1 분류의 개요

- 지도학습: 레이블(정답)이 있는 데이터가 주어진 상태에서 학습하는 머신러닝 방식
- 분류 ┌ 대표적인 지도학습
  └ 학습 데이터로 주어진 데이터의 피쳐+레이블을 머신러닝 알고리즘으로 학습해 모델 생성 → 새로운 데이터가 주어지는 경우 이지의 레이블 값 예측
  ⇒ 레이블의 패턴 학습
- 분류 알고리즘
  ① 나이브 베이즈: 베이즈 통계 & 생성 모델 기반
  ② 로지스틱 회귀: 독립변수와 종속변수의 선형 관계성
  ③ 결정 트리: 데이터 균일도에 따른 규칙 기반
  ④ 서포트 벡터 머신(SVM): 개별 클래스 간의 최대 분류 마진 찾기
  ⑤ 최소 근접 알고리즘: 근접 거리 기준
  ⑥ 신경망: 심층 연결 기반
  ⑦ 앙상블: 서로 다른(또는 동일한) 머신러닝 알고리즘을 결합
  ↖ 대부분 동일한 알고리즘을 결합

2 결정 트리

- 데이터에 있는 규칙을 자동으로 찾아내어 트리 기반의 분류 규칙을 만드는 것
- 스무고개 게임과 유사
- 분류 기준의 선정에 따라 알고리즘의 성능이 크게 좌우됨.
- 용어 >



- 트리의 깊이 ∝ 복잡도 : 과적합으로 인한 성능 저하 위험성 ↑
  ⇒ 데이터 분류 시 최대한 적은 데이터 세트가 해당 분류에 속할 수 있도록
- 결정 노드는 정보 균일도가 높은 데이터를 먼저 선택할 수 있도록 규칙 조건 생성
  ⇒ 정보 균일도가 데이터 세트로 쪼개질 수 있도록 조건을 찾아 서브 데이터 세트를 만들고, 다시 이 서브 데이터 세트에서 균일도가 높은 자식 데이터 세트 쪼개는 방식을 자식 트리로 내려가면서 반복함.
- 정보 균일도의 측도
  ① 정보 이득 ┌ 엔트로피 기반 … 혼잡도 ⇒ 1 - entropy
  └ 정보 이득이 높은 속성을 기준으로 분할
  ② 지니 계수 ┌ 불평등 지수 ⇒ 높을수록 불평등
  ├ 지니 계수 $\propto \frac{1}{\text{데이터 균일도}}$
  └ 지니계수가 낮은 속성을 기준으로 분할

♥ TITLE : 4_분류 ♥ DATE : 2022.09.02

① 결정 트리 모델의 특징

- 균일도 기반 ⇒ 중요한 몇 개의 피처가 명확한 규칙 트리를 만드는 데 크게 기여
- 장점>
  i) 알고리즘이 쉽고 직관적 ii) 출이 매우 명확함 iii) 시각화가 편리함.
  iv) 각 피처의 스케일링/정규화 등의 전처리 작업이 필수적이지는 X
- 단점>
  • 과적합 위험성 ↑ ⇒ 깊이 제한 필요!

② 결정 트리 parameters

- 사이킷런: DecisionTree Classifier(분류), DecisionTreeRegressor(회귀)
- CART (Classification And Regression Trees) 알고리즘 기반
- 종류>
  i) min_samples_split
  ↳노드 분할을 위한 최소 샘플 데이터 수 ⇒ 과적합 제어
  ↳ default = 2 , min_samples_split의 크기 $\propto \frac{1}{\text{분할 노드 수}}$

  ii) min_samples_leaf
  ↳ leaf node가 되기 위한 최소 샘플 데이터 수 ⇒ 과적합 제어
  ↳ 비대칭적 데이터의 경우 특정 클래스의 데이터가 극도로 작아질 수 O → 주의!

  iii) max_features
  ↳ 최적의 분할을 위해 고려할 최대 피쳐 개수, default=None ⇒ 모든 피처 사용
  ↳ int형: 대상 피처의 개수, float형: 전체 중 대상 피처의 퍼센트
  ↳ sqrt : $\sqrt{\text{전체 feature 개수}}$
  ↳ auto: sqrt와 동일
  ↳ log : log2(전체 개수)

  iv) feature_importances_ 속성
  ↳ ndarray 형태로 값 반환
  ↳ 값이 높을수록 중요도↑

  v) max_depth
  ↳ 트리의 최대 깊이 규정
  ↳ default = None ⇒ ① 완벽하게 클래스 결정 값이 될 때까지 깊이를 계속 키우며 분할하거나 노드가 가지는 데이터 개수가 ② min_samples_split보다 작아질 때까지 계속 깊이 증가
  ↳ 깊이 ∝ 분할 수 ⇒ 과적합 위험

  vi) max_leaf_nodes
  ↳ leaf node의 최대 개수

③ 결정 트리 모델의 시각화

- Graphviz 패키지 사용
- export_graphviz(...) : 학습된 결정 트리 규칙을 실제 트리 형태로 시각화
  ↑학습이 완료된 Estimator, 피처의 이름 리스트, 레이블 이름 리스트
- 예시>

petal length(cm) <= 2.45	← 규칙 조건
gini = 0.667	← value=[ ]로 주어진 데이터 분포에서의 지니 계수
samples = 120	← 현 규칙에 해당되는 데이터 개수
value = [41, 40, 39]	← 클래스 값 기반의 데이터 건수
class = setosa	← 하위 노드를 가질 경우 'setosa'의 개수가 제일 많다.

- 지니 계수가 0이 되면 노드의 분기 Stop
- 색이 짙어질수록 지니 계수 ↓, 해당 레이블에 속하는 데이터의 개수 ↑

④ 결정 트리 과적합

- 일부 이상치 데이터까지 분류하려 분할의 수 ↑ ⇒ 결정 기준 경계 ↑
  ↳ 학습 데이터에만 지나치게 최적화 → 과적합

♥ TITLE : 4_분류 ♥ DATE : 2022.09.03

3 앙상블 학습

① 개요

- 여러 개의 분류기를 생성하고 그 예측을 결합함으로써 보다 정확한 최종 예측을 도출하는 기법 ⇒ 신뢰성 ↑, 편향-분산 trade off의 효과 극대화
- 대부분의 정형 데이터 분류 시 우수한 성능을 보임
- 유형>
  - i) 보팅(voting)
    - ↳ 서로 다른 알고리즘을 가진 분류기들이 같은 데이터 세트에 대해 학습하고 예측한 결과를 가지고 투표를 통해 최종 예측 결과를 선정
  - ii) 배깅(bagging)
    - ↳ 서로 같은 알고리즘을 가진 분류기들이 부트스트랩 방식으로 샘플링된 데이터 세트에 대해 학습하고 예측한 결과를 가지고 투표를 통해 최종 예측 결과를 선정
    - ✶ 부트스트랩(BootStrap) 분할
      - ↳ 개별 Classifier에게 데이터를 샘플링해서 추출하는 방식
  - iii) 부스팅(Boosting)
    - ↳ 여러 개의 분류기가 순차적으로 학습을 수행하되, 앞에서 학습한 분류기가 예측이 틀린 데이터에 대해서는 올바르게 예측할 수 있도록 다음 분류기에는 가중치를 부여하며 학습·예측 진행
    - ↳ 예시> 그래디언트 부스트, XGBoost, LightGBM

⊕ 스태킹(Stacking)
- ↳ 여러 가지 다른 모델의 예측 결과값을 다시 학습 데이터로 만들어서 다른 모델로 재학습시켜 결과를 예측하는 방법

② 보팅의 유형

- i) Hard Voting
  - ↳ 다수결 원리와 비슷
  - ↳ 예측한 결괏값들 중 다수의 분류기가 결정한 예측값을 최종 보팅 결과 값으로 선정
- ii) Soft Voting
  - ↳ 분류기들의 레이블 값 결정 확률을 모두 더하고 이를 평균해서 이들 중 확률이 가장 높은 레이블 값을 최종 보팅 결과값으로 선정

- 일반적으로 Soft voting의 성능이 더 높기에 더 많이 사용되는 방식

③ 보팅 분류기

- 사이킷런의 VotingClassifier 클래스 이용
- parameters>
  - i) estimators : 리스트 값으로 보팅에 사용될 여러 개의 Classifier 객체들을 튜플 형식으로 입력
  - ii) voting : 방식 설정, default = 'hard'

♥ TITLE : 4_분류 ♥ DATE : 2022.09.04

4 랜덤 포레스트

- 배깅의 대표적 알고리즘
- 앙상블 알고리즘 중 비교적 빠른 속도, 다양한 영역에서 높은 예측 성능
- 결정 트리 알고리즘 기반 ⇒ 여러 개의 결정 트리 분류기가 전체 데이터에서 (배깅) 방식으로 각자의 데이터를 샘플링해 개별적으로 학습을 수행한 뒤 최종적으로 모든 분류기가 보팅을 통해 예측 결정
- 부트스트래핑 이용 ⇒ 데이터가 중첩된 개별 데이터 세트에 결정 트리 분류기를 각각 적용
- 사이킷런의 RandomForestClassifier 클래스 이용

• 하이퍼 파라미터 & 튜닝

i) n_estimators
- 결정 트리의 개수 지정 , default=10
- 많이 설정할수록 좋은 성능을 기대할 수 있지만 무조건 성능이 향상되는 것은 ×
- 늘릴수록 학습 수행 시간 ↑

ii) max_features
- 결정 트리에 사용된 max_features와 동일한 기능
- default = 'auto' ⇒ sqrt(전체 feature 개수) 만큼 참조

* max_depth나 min_samples_leaf와 같이 결정 트리에서 과적합을 개선하기 위해 사용되는 파라미터를 동일하게 적용할 수 O