

4. 분류

1. 분류(Classification)의 개요

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

-> 대표적인 유형인 **분류(Classification)**: 기존 데이터가 어떤 레이블에 속하는지 패턴을 알고리즘으로 인지한 뒤에 새롭게 관측된 데이터에 대한 레이블을 판별

분류

- 베이즈(Bayes) 통계와 생성 모델에 기반한 나이브 베이즈(Naive Bayes)
- 독립변수와 종속변수의 선형 관계성에 기반한 로지스틱 회귀(Logistic Regression)
- 데이터 균일도에 따른 규칙 기반의 결정 트리(Decision Tree)
- 개별 클래스 간의 최대 분류 마진을 효과적으로 찾아주는 서포트 벡터 머신(Support Vector Machine)
- 근접 거리를 기준으로 하는 최소 근접(Nearest Neighbor) 알고리즘
- 심층 연결 기반의 신경망(Neural Network)
- 서로 다른(또는 같은) 머신러닝 알고리즘을 결합한 **앙상블(Ensemble)** #이번 장에서 다룰 알고리즘

앙상블

- 배깅(Bagging) 방식: 대표 방식인 랜덤 포레스트(Random Forest)는 뛰어난 예측 성능, 상대적으로 빠른 수행 시간, 유연성 등으로 많은 분석가가 애플리케이션에 활용하는 알고리즘
- 부스팅(Boosting) 방식: 그래디언트 부스팅(Gradient Boosting)은 이것의 예측 성능을 한 단계 발전시키면서도 수행 시간을 단축시킨 알고리즘이 계속 등장하면서 정형 데이터의 분류 영역에서 가장 활용도가 높은 알고리즘으로 자리 잡음

2. 결정 트리

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

규칙을 쉽게 표현하는 방법: if,else

결정 트리의 구조

- 규칙 노드: 규칙 조건. 데이터 세트에 피쳐가 있고 이러한 피쳐가 결합해 규칙 조건을 만들 때마다 만들어짐
- 리프 노드: 결정된 클래스 값
- 서브 트리: 새로운 규칙 조건마다 생성됨

많은 규칙이 있음 -> 분류를 결정하는 방식이 더욱 복잡해짐 -> 과적합으로 이어짐

= 트리의 깊이가 깊어질수록 결정 트리의 예측 성능이 저하될 가능성이 높음

= 적은 결정 노드로 높은 예측 정확도를 가지려면 데이터를 분류할 때 최대한 많은 데이터 세트가 해당 분류에 속할 수 있도록 결정 노드의 규칙이 정해져야 함 = 최대한 균일한 데이터 세트를 구성할 수 있도록 분할하는 것이 필요

결정 노드는 정보 균일도가 높은 데이터 세트를 먼저 선택할 수 있도록 규칙 조건을 만듦

정보 균일도가 데이터 세트로 쪼개질 수 있도록 조건을 찾아 서브 데이터를 만듦

-> 다시 이 서브 데이터 세트에서 균일도가 높은 자식 데이터 세트 쪼개는 방식을 자식 트리로 내려가는 반복하는 방식으로 데이터 값 예측

정보의 균일도를 측정하는 방법: 엔트로피를 이용한 정보 이득(Information Gain)지수와 지니계수

- 정보 이득: 엔트로피라는 개념을 기반

엔트로피는 주어진 데이터 집합의 혼잡도를 의미하는데, 서로 다른 값이 섞여 있으면 엔트로피가 높고, 같은 값이 섞여 있으면 엔트로피가 낮습니다.

정보 이득 지수 = $1 - \text{엔트로피 지수}$

결정 트리는 이 정보 이득 지수로 분할 기준을 정합니다.

즉, 정보 이득이 높은 속성을 기준으로 분할합니다.

지니 계수: 경제학에서 불평등 지수를 나타낼 때 사용하는 계수

0이 가장 평등하고 1로 갈수록 불평등합니다.

머신러닝에 적용될 때는 지니 계수가 낮을수록 데이터 균일도가 높은 것으로 해석해 지니 계수가 낮은 속성을 기준으로 분할합니다.

DecisionTreeClassifier(결정 트리 알고리즘을 사이킷런에서 구현)

- 지니 계수를 이용해 데이터 세트 분할
- 데이터 세트를 분할하는 데 가장 좋은 조건, 즉 정보 이득이 높거나 지니 계수가 낮은 조건을 찾아서 자식 트리 노드에 걸쳐 반복적으로 분할한 뒤, 데이터가 모두 특정 분류에 속하게 되면 분할을 멈추고 분류를 결정

결정 트리 모델의 특징

장점

- 정보의 '균일도'라는 률을 기반으로 하고 있어서 알고리즘이 쉽고 직관 (결정 트리가 률이 매우 명확하고, 이에 기반해 어떻게 규칙 노드와 리프 노드가 만들어지는지 알 수 있고, 시각화로 표현까지 할 수 있음)
- 각 피처의 스케일링과 정규화 같은 전처리 작업이 거의 필요 없음

단점

- 과적합으로 정확도가 떨어짐.
- 피처 정보의 균일도에 따른 률 규칙
(피처 정보의 균일도에 따른 률 규칙으로 서브 트리를 계속 만들다 보면 피처가 많고 균일도가 다양하게 존재할수록 트리의 깊이가 커지고 복잡해질 수밖에 없음. 복잡한 학습 모델은 실제 상황에 유연하게 대처 못 함) -> 트리의 크기를 사전에 제한하는 것이 오히려 성능튜닝에 더 도움이 됨

결정 트리 파라미터

DecisionTreeClassifier 클래스: 분류(사이킷에서 결정트리 알고리즘 구현)

DecisionTreeRegressor 클래스: 회귀(사이킷에서 결정트리 알고리즘 구현)

파라미터

- `min_samples_split`: 노드를 분할하기 위한 최소한의 샘플 데이터 수로 과적합을 제어하는 데 사용됨. 디폴트는 2이고 작게 설정할수록 분할되는 노드가 많아져서 과적합 가능성 증가
- `min_sample_leaf`:
분할이 될 경우 왼쪽과 오른쪽의 브랜치 노드에서 가져야 할 최소한의 샘플 데이터 수
- `max_features`: 최적의 분할을 위해 고려할 최대 피처 개수. 디폴트는 `None`으로 데이터 세트의 모든 피처를 사용해 분할 수행

- max_depth: 트리의 최대 깊이를 규정. 디폴트는 None.
- max_leaf_nodes:
말단 노드(Leaf)의 최대 개수

결정 트리 모델의 시각화

시각화된 도표를 통해 결정 트리 규칙 구성 보기

- 리프 노드: 더 이상 자식 노드가 없는 노드
최종 클래스(레이블) 값이 결정되는 노드 리프 노드가 되려면 오직 하나의 클래스 값으로 최종데이터가 구성되거나, 리프노드가 될 수 있는 하이퍼 파라미터 조건을 충족하면 됨
- 브랜치 노드: 자식 노드가 있는 노드. 자식 노드를 만들기 위한 분할 규칙 조건을 가지고 있음 ex) petal length(cm) <= 2.45와 같이 피처의 조건이 있는 것은 자식 노드를 만들기 위한 규칙 조

<노드 내에 기술된 지표의 의미>

- gini: 다음의 value=[]로 주어진 데이터 분포에서의 지니 계수
- samples: 현 규칙에 해당하는 데이터 건수
- : 클래스 값 기반의 데이터 건수
ex) 붓꽃 데이터 세트는 클래스 값으로 0, 1, 2를 가지고 있으며, 0 : Setosa, 1 : Versicolor, 2 : Virginica 품종을 가리킵니다.
만일 Value = [41, 40, 39]라면 클래스 값의 순서로 Setosa 41개, Versicolor 40개, Virginica 39개로 데이터가 구성돼 있다는 의미입니다

<붓꽃 데이터 세트 결정 트리 시각화>

1. 루트 노드인 1번 노드의 지표 설명 petal length (cm) <= 2.45 gini = 0.667 samples = 120 value = [41, 40, 39] class = setosa

->

samples = 120개는 전체 데이터가 120개라는 의미

value = [41, 40, 39]는 Setosa 41개, Versicolor 40개, Virginica 39개로 데이터 구성

sample 120개가 value = [41, 40, 39] 분포도로 되어 있으므로 지니 계수는 0.667

petal length (cm) <= 2.45 규칙으로 자식 노드 생성

class = setosa는 하위 노드를 가질 경우에 setosa의 개수가 41 개로 제일 많다는 의미

petal length (cm) <= 2.45 규칙이 True 또는 False로 분기하게 되면 2번, 3번 노드가 만들어짐

2. 2번 노드 gini = 0.0 samples = 41 value = [41, 0, 0] class = setosa

->

모든 데이터가 Setosa로 결정되므로 클래스가 결정된 리프 노드가 되고 더 이상 2번 노드에서 규칙을 만들 필요가 없음

즉, 2번 노드는 petal length (cm) <= 2.45가 True인 규칙으로 생성되는 리프 노드

41 개의 샘플 데이터 모두 Setosa이므로 예측 클래스는 Setosa로 결정
지니계수는 0임.

3. 3번 노드 petal width(cm) <= 1.55 gini = 0.5 samples = 79 value = [0, 40, 39] class = versicolor

->

Petal length (cm) ≤ 2.45 가 False인 규칙 노드 79개의 샘플 데이터 중 Versicolor 40개, Virginica 39개로 여전히 지니 계수는 0.5로 높으므로 다음 자식 브랜치 노드로 분기할 규칙 필요
 petal width (cm) ≤ 1.55 규칙으로 자식 노드 생성.

4. 4번 노드 petal length (cm) ≤ 5.25 gini = 0.051 samples = 38 value = [0,37,1] class = versicolor

->

Petal width (cm) ≤ 1.55 가 True인 규칙 노드
 38개의 샘플 데이터 중 Versicolor 37개, Virginica가 1개로 대부분이 versicolor임.
 지니 계수는 0.051로 매우 낮으나 여전히 Versicolor와 Virginica가 혼재돼 있으므로 petal length(cm) ≤ 5.25 라는 새로운 규칙으로 다시 자식 노드 생성

5. 5번 노드 petal width(cm) ≤ 1.75 gini = 0.136 samples = 41 value = [0,3,38] class = virginica

->

Petal width (cm) ≤ 1.55 가 False인 규칙 노드
 41 개의 샘플 데이터 중 Versicolor 3개, Virginica가 38개로 대부분이 virginica임.
 지니 계수는 0.136으로 낮으나 여전히 Versicolor와 Virginica가 혼재되어 있으므로 petal width(cm) ≤ 1.75 라는 새로운 규칙으로 다시 자식 노드 생성

6. 각 노드의 색깔 : 붓꽃 데이터의 레이블

<결정 트리 알고리즘을 제어하는 하이퍼 파라미터 변경>

- max_depth 결정 트리의 최대 트리 깊이 제어
- min_samples_split 자식 노드를 분할해 만들기 위한 최소한의 샘플 데이터 개수 ex)
 min_samples_split=4로 설정하면 노드 안에 value가 [0,2,1] 이어도 더 이상 분할하지 않고 리프노드가 됨.
 자식 노드로 분할하려면 최소한 샘플 개수가 4개는 필요한데, 3개밖에 없음
- min_samples_leaf 분할될 경우 왼쪽과 오른쪽 자식 노드 각각이 가지게 될 최소 데이터 건수를 지정
 어떤 노드가 분할할 경우, 왼쪽과 오른쪽 자식 노드 중에 하나라도 min_samples_leaf로 지정된 최소 데이터 건수보다 더 작은 샘플 데이터 건수를 갖게 된다면, 해당 노드는 더 이상 분할하지 않고 리프 노드가 됨

<feature_importances_> 결정 트리는 균일도에 기반해 어떠한 속성을 규칙 조건으로 선택하느냐가 중요한 요건

사이킷런은 결정 트리 알고리즘이 학습을 통해 규칙을 정하는 데 있어 피처의 중요한 역할 지표를 DecisionTreeClassifier 객체의 feature_importances_ 속성으로 제공

feature_importances_는 ndarray 형태로 값을 반환하며 피처 순서대로 값이 할당 ex) [0.01667014 0.02500521 0.03200643 0.92631822]라면 첫 번째 피처의 피처 중요도가 0.01667014, 두 번째 피처는 0.02500521와 같이 매치됨

피처가 트리 분할 시 정보 이득이나 지니 계수를 얼마나 효율적으로 잘 개선시켰는지를 정규화된 값으로 표현

결정 트리 과적합(Overfitting)

사이킷런은 분류를 위한 테스트용 데이터를 쉽게 만들 수 있도록 `make_classification()` 함수 제공
`make_classification()` 함수의 반환값: 피처 데이터 세트와 클래스 레이블 데이터 세트

`visualize_boundary()`: 머신러닝 모델이 클래스 값을 예측하는 결정 기준을 색상과 경계로 나타내 모델이 어떻게 데이터 세트를 예측 분류하는지 잘 이해할 수 있게 해줌

결정 트리의 기본 하이퍼 파라미터 설정은 리프 노드 안에 데이터가 모두 균일하거나 하나만 존재해야 하는 엄격한 분할 기준으로 인해 결정 기준 경계가 많아지고 복잡해짐

일부 이상치(Outlier) 데이터까지 분류하기 위해 분할이 자주 일어나서 결정 기준 경계가 매우 많아짐

-> 과적합

-> 해결방안 예시: `min_samples_leaf = 6`을 설정해 6개 이하의 데이터는 리프 노드를 생성할 수 있도록 리프 노드 생성 규칙 완화

결정 트리 실습 - 사용자 행동 인식 데이터 세트

결정 트리를 이용해 UCI 머신러닝 리포지토리(Machine Learning Repository)에서 제공하는 사용자 행동인식(Human Activity Recognition) 데이터 세트에 대한 예측 분류

GridSearchCV를 이용해 사이킷런 결정 트리의 깊이를 조절할 수 있는 하이퍼 파라미터인 `max_depth` 값을 변화시키면서 예측 성능확인 :

결정 트리의 트리 깊이(Tree Depth)가 예측 정확도에 주는 영향 알아보기

-> `mean_test_score`는 `max_depth`가 8일 때 0.854로 정확도가 정점이고, 이를 넘으면서 정확도가 계속 떨어짐 (과적합으로 인한 성능 저하)

3. 양상별 학습

양상별 학습 개요

양상별 학습(Ensemble Learning)을 통한 분류는 여러 개의 분류기 (Classifier)를 생성하고 그 예측을 결합함으로써 보다 정확한 최종 예측을 도출하는 기법

대부분의 정형 데이터 분류 시에는 양상별이 뛰어난 성능을 나타냄

양상별 학습의 유형

- 보팅(Voting) : 서로 다른 알고리즘을 가진 분류기를 결합

ex) 선형 회귀, K 최근접 이웃, 서포트 벡터 머신이라는 3개의 ML 알고리즘이 같은 데이터 세트에 대해 학습하고 예측한 결과를 가지고 보팅을 통해 최종 예측 결과를 선정하는 방식
- 배깅(Bagging) : 각각의 분류기가 모두 같은 유형의 알고리즘 기반이지만, 데이터 샘플링을 서로 다르게 가져가면서 학습을 수행해 보팅을 수행

개별 분류기가 부트스트래핑 방식으로 샘플링된 데이터 세트에 대해서 학습을 통해 개별적인 예측을 수행한 결과를 보팅을 통해서 최종 예측 결과를 선정하는 방식 교차검증과 달리 데이터 세트 간에 중첩을 허 대표적인 배깅 방식: 랜덤 포레스트 알고리즘
- 부스팅(Boosting): 여러 개의 분류기가 순차적으로 학습을 수행하되, 앞에서 학습한 분류기가 예측이 틀린 데이터에 대해서는 올바르게 예측할 수 있도록 다음 분류기에게는 가중치(weight)를 부여하면서 학습과 예측을 진행 대표적인 부스팅 모듈: 그래디언트 부스트, XGBoost(eXtra Gradient Boost), LightGBM(Light Gradient Boost)
- 스태킹 : 여러 가지 다른 모델의 예측 결괏값을 다시 학습 데이터로 만들어서 다른 모델(메타 모델)로 재학습시켜 결과를 예측

보팅 유형 - 하드 보팅(Hard Voting)과 스그트 보팅(Soft Voting)

1. 하드 보팅

예측한 결괏값들 중 다수의 분류기가 결정한 예측값을 최종 보팅 결괏값으로 선정. 다수결 원칙과 비슷.

ex) Classifier 1 번, 2번, 3번, 4번인 4개로 구성한 보팅 양상을 기법에서 분류기 1번, 3번, 4번이 1로 레이블 값을 예측하고 분류기 2번이 2로 레이블 값을 예측하면 다수결 원칙에 따라서 최종 예측은 레이블 값 1이 됨

2. 소프트 보팅

분류기들의 레이블 값 결정 확률을 모두 더하고 이를 평균해서 이들 중 확률이 가장 높은 레이블 값을 최종 보팅 결괏값으로 선정. 일반적으로 소프트 보팅이 보팅 방법으로 적용

ex) 소프트 보팅은 각 분류기의 레이블 값 예측 확률을 평균 내어 최종 결정. 레이블 값 1의 확률이 0.65로 레이블 값 2인 확률 0.35보다 크면 레이블 값 1로 최종 보팅

보팅 분류기(Voting Classifier)

사이킷런은 보팅 방식의 양상을 구현한 *VotingClassifier* 클래스를 제공

VotingClassifier 클래스 주요 생성 인자: estimators와 voting

- estimators: 리스트 값으로 보팅에 사용될 여러 개의 Classifier 객체들을 튜플 형식으로 입력받음
- voting: 'hard' 시 하드 보팅, 'soft' 시 소프트 보팅 방식을 적용하라는 의미(기본은 'hard')

보팅 방식의 양상을 이용해 위스콘신 유방암 데이터 세트를 예측 분석

보팅으로 여러 개의 기반 분류기를 결합한다고 해서 무조건 기반 분류기보다 예측 성능이 향상되지는 않음. 그럼에도 불구하고 지금 소개하는 보팅을 포함해 배깅과 부스팅 등의 양상을 방법은 전반적으로 다른 단일 ML 알고리즘보다 뛰어난 예측 성능을 가지는 경우가 많음.

결정 트리 알고리즘의 장점은 그대로 취하고 단점은 보완하면서 편향-분산 트레이드오프의 효과를 극대화할 수 있음.(결정 트리 알고리즘의 단점을 수십~수천 개의 매우 많은 분류기를 결합해 다양한 상황을 학습하게 하기 때문)

4. 랜덤 포레스트

랜덤 포레스트의 개요 및 실습

배깅(bagging): 같은 알고리즘으로 여러 개의 분류기를 만들어서 보팅으로 최종 결정하는 알고리즘

대표적인 알고리즘: [랜덤포레스트](#)

<랜덤 포레스트>

- 여러 개의 결정 트리 분류기가 전체 데이터에서 배깅 방식으로 각자의 데이터를 샘플링해 개별적으로 학습을 수행한 뒤 최종적으로 모든 분류기가 보팅을 통해 예측 결정을 하게 됨
- 개별적인 분류기의 기반 알고리즘은 결정 트리이지만 개별 트리가 학습하는 데이터 세트는 전체 데이터에서 일부가 중첩되게 샘플링된 데이터 세트(여러 개의 데이터 세트를 중첩되게 분리하는 것: 부트스트래핑(bootstrapping) 분할 방식)

ex) 부트스트래핑 샘플링 방식 1 2 3 4 5 6 7 8 9 10

-> 서브세트 #1: 1 2 3 3 3 5 6 8 8 9

-> 서브세트 #2: 1 3 4 5 6 8 8 9 9 10 -> 서브세트 #3: 1 1 3 4 4 5 6 6 9 9

사이킷런은 *RandomForestClassifier* 클래스를 통해 랜덤 포레스트 기반의 분류 지원

랜덤 포레스트 하이퍼 파라미터 및 튜닝

트리 기반의 양상블 알고리즘의 단점: 하이퍼 파라미터가 너무 많고, 그로 인해서 튜닝을 위한 시간이 많이 소모됨

랜덤 포레스트의 하이퍼 파라미터(결정 트리에서 사용되는 하이퍼 파라미터와 같은 파라미터가 대부분임)

- `n_estimators` : 랜덤 포레스트에서 결정 트리의 개수를 지정합니다. 디폴트는 10개입니다. 많이 설정할 수록 좋은 성능을 기대할 수 있지만 계속 증가시킨다고 성능이 무조건 향상되는 것은 아닙니다.
- `max_features`는 결정 트리에 사용된 `max_features` 파라미터와 같습니다. 하지만 *RandomForestClassifier*의 기본 `max_features`는 'None'이 아니라 'auto', 즉 'sqrt'와 같습니다. 따라서 랜덤 포레스트의 트리를 분할하는 피처를 참조할 때 전체 피처가 아니라 $\text{sqrt}(\text{전체 피처 개수})$ 만큼 참조합니다 ex) 전체 피처가 16개라면 분할을 위해 4개 참조
- `max_depth`나 `min_samples_leaf`, `min_samples_split`과 같이 결정 트리에서 과적합을 개선하기 위해 사용되는 파라미터가 랜덤 포레스트에도 똑같이 적용될 수 있습니다.