8. Decision Tree
2019년 가을 학기

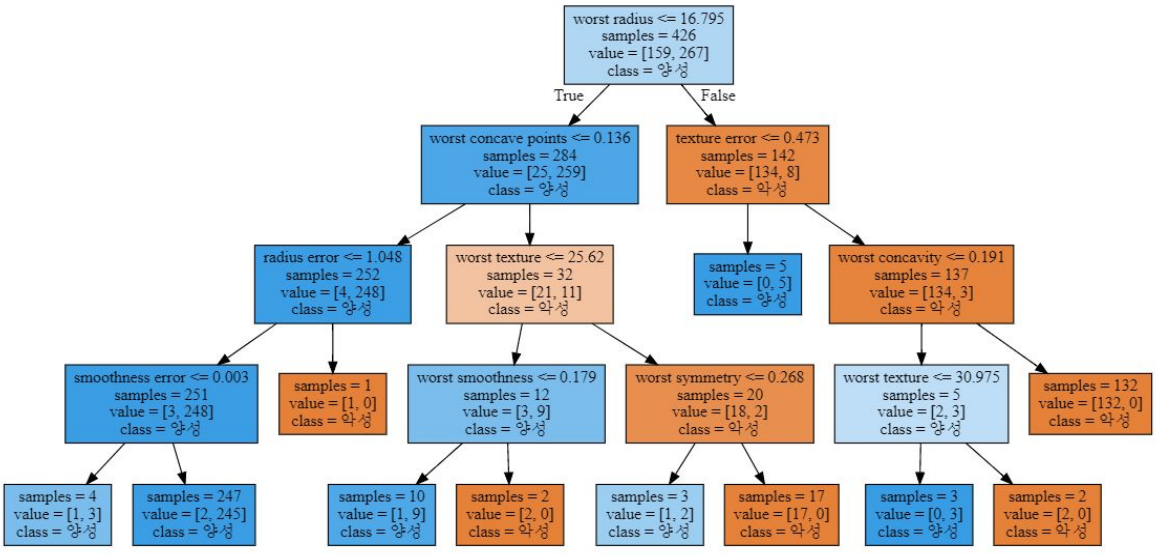
2019년 08월 26일
https://www.github.com/KU-BIG/KUBIG_2019_Autumn

이번 강의에서는 머신러닝과 통계적 지식 없이 이해할 수 있을 정도로 다분히 직관적인 결정트리를 소개한다. 결정트리는 모델의 학습 방법이 우리가 일상 속에서 이미 경험해본 방법과 닮아있어 이해하기 쉽다. 또한, 시각화도 어렵지 않아서 해석하기도 편하다. 이와 같이 편리한 결정트리는 앞으로 나올 강의들의 바탕이 되므로 이번 강의에서 잘 이해해두기로 하자.

1. Introduction

의사결정 나무는 어렸을 때 스무고개 놀이를 통해 정답을 맞추는 방식과 닮아있다. 스무고개는 정답을 맞추기 위해 질문을 하나씩 이어하면서 정답의 후보를 줄여나간다. 그리고 남은 정답의 후보가 충분히 좁혀지고 나면 우리는 정답을 맞출 수 있다. 스무고개를 통해 정답을 맞춰가는 것처럼 데이터 포인트를 분류해내거나 예측하는 모델이 바로 의사결정나무(decision tree)이다.

1.1 의사결정나무의 구조



[그림 1] https://github.com/rickiepark/introduction_to_ml_with_python/blob/master/02-supervised-learning.ipynb

의사결정나무는 이름 답게 구성 요소도 나무를 연상시킨다. 뿌리에 해당하는 Root node와 잎에 해당하는 leaf node가 있다. 두 node 간의 관계를 볼 때는 Parent node와 그 node에서 갈라져 나온 child node가 존재한다. 즉 root node는 child node일 수 없고, leaf node는 parent node일 수 없다. 유방암 데이터를 사용한 의사결정나무인 [그림 1]을 통해 더 자세히 알아보자.

맨 위에서 분기가 시작되는 지점을 root node라고 하는데, 그 root node에서부터 각 변수의 특성에 따라 나눠지고 있음을 확인할 수 있다. 구체적으로, root node를 보면 데이터에는 총 426개의 sample이 있고 악성 샘플이 159개, 양성 샘플이 267개 있다. 이 root node에서 worst radius라는 변수가 16.795 이하인지의 여부로 처음 분류가 일어났다.

worst radius라는 변수의 값을 기준으로 분류한 결과 worst radius가 16.795 이하인 데이터 포인트들은 왼쪽 True화살표를 따라서 악성 샘플이 25개, 양성샘플이 259개인 파티션을 구성한다. 이와 같은 방식으로 각 기준에 따라 데이터 포인트들을 분류해나갈 수가 있다. [그림 1]의 맨 아래층에서 가장 오른쪽 node를 보자. 이 파티션은 악성샘플만 2개인 sample로 구성되어 있음을 알 수 있다. 이와 같이 decision tree에서 더 이상 분기가 일어나지 않는 마지막 노드를 leaf node라고 한다.

Root node에서부터 leaf node까지의 일련의 node들의 집합을 path라고 한다. 그렇기에 leaf node의 개수가 path의 개수가 된다. 가장 오른쪽 leaf node로 가는 path를 살펴보자. Root node에서 worst radius > 16.795인 sample은 총 142개이다. 해당 노드의 오른쪽 child node는 142개의 sample중에서 texture error > 0.473인 137개의 데이터가 있다. 다시 이 노드의 오른쪽 child node를 보면 worst concavity > 0.191인 132개의 데이터가 모두 악성으로 더 이상 분류가 일어나지 않는 leaf node에 도달했다. 말로 풀어쓴 이 path를 수식화 하면 다음과 같다.

$$(\text{worst radius} > 16.795 \wedge \text{texture error} > 0.473 \wedge \text{worst concavity} > 0.191) \rightarrow \text{악성}$$

우리가 가지고 있는 데이터로 어떤 순서와 변수의 값을 기준으로 분류해나갈지 모형을 만들고 나면 새로운 데이터가 주어졌을 때 예측을 할 수 있다. 새로운 데이터 포인트가 가지고 있는 변수들의 값을 기준으로 우리가 만든 모델에 따라 분류해나가면 그 데이터 포인트가 위치하는 leaf가 있을 것이다.

위의 유방암 데이터의 사례와 같이 예측을 해야할 label이 이산형인 경우는 새로운 데이터가 도달한 leaf에서 다수를 구성하고 있는 class를 반환하면 된다. 예를 들어, 새로운 데이터 포인트가 worst radius, texture error, worst concavity라는 세 변수들 모두에서 우리의 모델이 기준으로 하는 값보다 작아 false를 반환한다면 132개의 샘플이 모두 악성으로 분류되었던 순수

노드(pure node)에 위치하게 된다. 따라서, 그 새로운 데이터 포인트는 우리의 decision tree 모델에 따라 악성으로 예측할 수 있다. 지금까지 모델의 구조와 이와 같은 모델로 어떻게 예측을 할 수 있는지 살펴보았다. 그런데 여기서 한 가지 의문점이 남는다. 각각의 노드들에서 하는 테스트는 어떤 기준으로 선택되어야할까? 그 답을 다음 불순도 지표라는 섹션에서 살펴본다.

2. 불순도 지표

의사결정나무는 불순도(impurity)가 감소하고 순도(homogenity)가 증가하는 방식으로 학습한다. 즉, 결정나무에서 child node들은 그 위의 parent node보다 더 동질적으로 구성되어 있어야 한다는 뜻이다. 이렇듯 의사결정나무에서는 매 노드마다 각각의 테스트의 결과들을 비교해서 가장 동질적인 집합을 갖는 child node들이 나오도록 하는 테스트를 선택한다. 그렇다면 동질적으로 질서정연하게 분류가 되었다는 것에 대한 척도가 필요할텐데 그 중에 대표적인 엔트로피(entropy)와 지니계수(Gini index)에 대해 알아보자.

2.1 엔트로피 (entropy)

$$H(S) = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

데이터셋 S에 대한 엔트로피는 위와 같은 식으로 정의된다. 각 데이터 포인트가 $c_1, c_2, \ldots c_n$ 등 유한개의 클래스 중 하나에 속한다고 할 때, p_i는 한 데이터 포인트가 클래스 c_i에 속할 확률을 의미한다. 위에서 언급된 유방암 데이터로 root node에서의 엔트로피를 계산해보면 다음과 같다.

```
> p1=159/426
> p2=267/426
> H_S=-p1*log2(p1)-p2*log2(p2)
> H_S
[1] 0.953127
```

이제 root node 수준에서 데이터셋 전체의 엔트로피가 아니라 데이터셋이 여러 개의 파티션으로 나뉘어졌을 때의 엔트로피를 구해보자. 아래의 식에서 q_i는 데이터셋이 총 m개의 파티션으로 나뉘어졌을 때 파티션 S_i가 차지하는 비율을 의미하고 $H(S_i)$는 파티션 S_i 각각의 엔트로피를 의미한다. 이 공식은 각 파티션들의 엔트로피의 가중합이라고 이해할 수 있다.

$$H = \sum_{i=1}^{m} q_i H(S_i)$$

위에서 언급된 유방암 데이터에서 worst radius가 16.795 이하인가를 기준으로 테스트가 실시되어 두개의 파티션으로 나뉘어졌을 때, 엔트로피는 다음과 같다.

```
> q1=284/426
> q2=142/426
> q1+q2
[1] 1
> H_S1=(-25/284)*log2(25/284)+(-259/284)*log2(259/284)
> H_S1
[1] 0.4298536
> H_S2=(-134/142)*log2(134/142)+(-8/142)*log2(8/142)
> H_S2
[1] 0.3127334
> H=q1*H_S1+q2*H_S2
> H
[1] 0.3908135
```

Root node에서 엔트로피가 0.95였던 것을 기억한다면 약 0.55(=0.95-0.40)만큼 엔트로피가 감소하였다. 즉, 첫번째 분기를 통해 불순도는 감소하였고 순도는 증가하였으며 정보획득(information gain)이 있었음을 확인할 수 있다. ID3 알고리즘이 이와 같은 방식으로 엔트로피를 이용해 정보획득을 최대화하는 방향으로 각 분기에서 변수를 선택해 분류해나간다.

$$\text{GAIN}_{entropy} = Entropy_{parent} - \sum_{i=1}^{m} q_i Entropy_{child}(k)$$

Information gain은 각각 작지만 순수한 파티션들을 많이 생성하여 강제적으로 순수성을 얻으려는 경향이 있어 과적합(overfitting)을 일으킨다. 이를 방지하기 위해서는 information gain을 변형한 Gain Ratio(C4.5 알고리즘)을 활용하기도 한다. Gain Ratio는 Information Gain에 Split INFO를 penalty로 주어서 많이 파티션을 쪼갤수록 그 GAIN값이 줄어들게 하였다.

$$\text{GainRATIO}_{split} = \frac{\text{GAIN}_{entropy}}{SPlitINFO}, \quad SplitINFO = -\sum_{i=1}^{m} q_i \log q_i$$

2.2 지니불순도 (Gini impurity)

지니불순도는 모집단이 완전히 동질적인 원소들로만 구성되어있다면 랜덤하게 복원추출한 두 원소가 동일할 확률은 1이라는 사실을 기반으로 하는 지표이다. 두 종류의 값을 갖는 모집단이 가장 이질적으로 구성되어있는 0.5:0.5 비율에서 지니불순도는 최대값 0.5를 갖는다. 반면, population이 가장 동질적으로 1:0으로 구성되었을 때 지니불순도는 최소값 0을 갖는다. 그러므로 지니불순도 역시 크기가 클수록 불순도가 높음을 나타내는 척도라고 할 수 있다.

$$G(S) = \sum_{i=1}^{n} p_i(1 - p_i) = 1 - \sum_{i=1}^{n} p_i^2$$

지니불순도 식을 살펴보자. p_i는 i^{th} class에 속하는 원소들의 비율이다. 위의 식이 복원추출을 했을 때 n개의 class로 이루어져있는 상황에서 서로 다른 class에 속하는 두 원소를 선택할 확률이라는 점은 쉽게 이해할 수 있다. 이 확률의 합들이 클수록 어떤 모집단이 이질적으로 구성되어있다고 보며 유방암 데이터의 root node에서 지니불순도를 계산해보면 다음과 같다.

```
> p1=159/426
> p2=267/426
> G_S=p1*(1-p1)+p2*(1-p2)
> G_S
[1] 0.4678635
```

엔트로피를 계산했을 때와 마찬가지로 root node수준에서의 지니불순도뿐만 아니라 데이터셋이 여러개의 파티션으로 나뉘어졌을 때의 지니불순도를 구해보자. 아래의 식에서 R_k는 k번째 파티션이 데이터셋에서 차지하는 비율이다. 또한, 여기서 p_{ki} 는 k번째 파티션 내에서 ith class가 차지하고 있는 비율이다. 이 공식도 파티션이 데이터셋에서 차지하는 비율을 가중치로 하는 파티션들의 지니불순도의 가중합이라는 점을 발견할 수 있다.

$$G = \sum_{k=1}^{m} R_k \sum_{i=1}^{n} p_{ki}(1 - p_{ki}) = \sum_{k=1}^{m} R_k G(S_k)$$

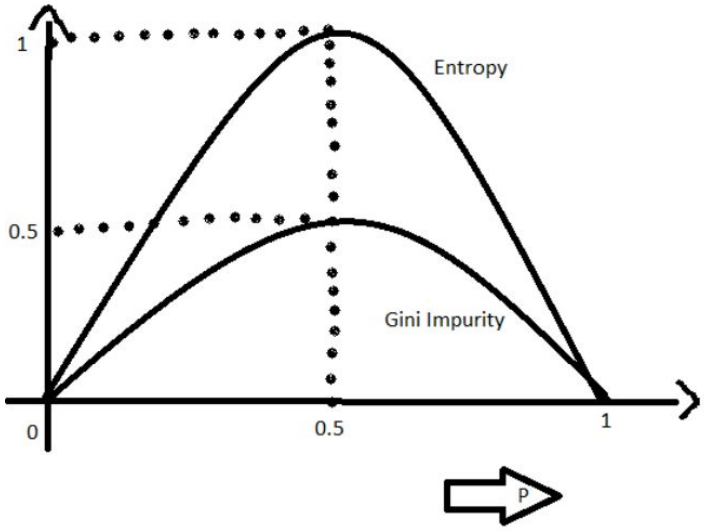
위에서 언급된 유방암 데이터에서 worst radius가 16.795이하인가의 여부로 두 개의 파티션으로 구분되었을 때의 지니불순도를 계산해보면 다음과 같다.

```
> R1=284/426
> p11=25/284
> p12=259/284
>
> R2=142/426
> p21=134/142
> p22=8/142
>
>
> G=R1*(p11*(1-p11)+p12*(1-p12))+R2*(p21*(1-p21)+p22*(1-p22))
> G
[1] 0.1424817
```

Root node의 Gini가 0.47인 것과 비교하면 지니불순도가 약 0.33(=0.47-0.14)만큼 감소했음을 확인할 수 있다. CART 알고리즘이 이와 같은 방식으로 지니불순도를 metric으로 사용한다. 아래는 지니불순도를 활용한 Gain을 구하는 수식이다.

$$\text{GAIN}_{Gini} = Gini_{parent} - \sum_{k=1}^{m} R_k Gini_{child}(k)$$

[그림2]는 class에 속한 data의 비율 p를 x축으로 두고, y축에는 Entropy와 Gini impurity의 값을 두었다. 이를 통해서 Entropy가 Gini impurity 보다 더욱 확실하게 차이를 보여주는 것을 알 수 있다. 초창기의 알고리즘은 Gini를 사용하였으나 최신 알고리즘은 Entropy를 이용하는 추세이다.

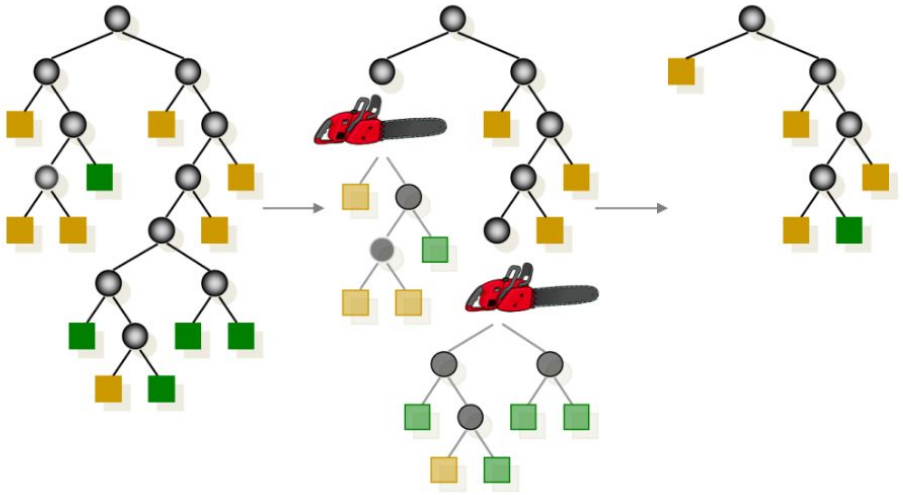


[그림 2] https://medium.com/@rubeen.786.mr/a-gentle-introduction-to-decision-tree-learning-db58f5b955e8

3. 가지치기 (Pruning)

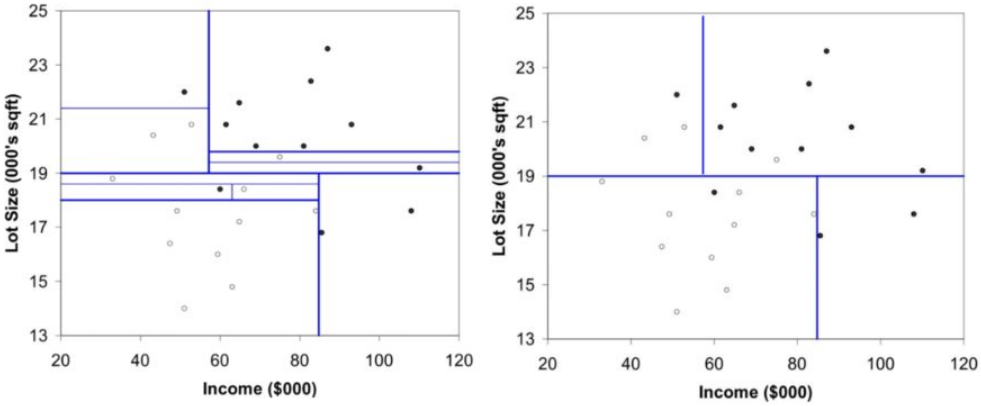
지금까지 어떤 기준으로 트리가 노드를 분기해나가는지 살펴보았다. 순도의 증가와 불순도의 감소로 정보획득을 하고, 이 정보획득량을 최대화하는 방향으로 트리는 생성된다. 불순도의 지표로는 엔트로피와 지니 불순도가 있다. 그런데 최적의 결정트리를 선택한다는 건 불순도의 감소만으로는 해결될 수 없는 부분이 있다. 바로 과적합(Overfitting)의 문제이다.

주어진 데이터셋에서 불순도 감소를 유일한 기준으로 트리를 형성한다면 우리가 가지고 있는 training set을 모두 분류해내는 가장 깊이가 깊고 leaf의 순도가 100%인 fully grown tree가 좋을 것이다. 그러나 이렇게 되면 우리가 가지고 있는 training set에서는 정확도가 100%가 나올 수 있지만, 새로 주어진 데이터에서는 어느 순간부터 오히려 정확도가 떨어진다. 그러므로 결정트리의 과적합을 방지하기 위해 가지치기(pruning)라는 작업이 필요하다.



[그림 3] https://ratsgo.github.io/machine%20learning/2017/03/26/tree/

가지치기는 모델이 과적합이 되지 않도록 tree의 노드수를 의도적으로 줄여주는 작업을 말하는데 사전가지치기와 사후가지치기로 구분된다. 사전 가지치기는 트리의 최대 깊이나 리프의 최대 개수를 제한하거나, 노드가 분할하기 위한 포인트의 최소 개수를 지정하는 방법이다. 이에 반해 사후 가지치기는 트리를 만든 후 데이터 포인트가 적은 노드를 삭제하거나 병합하는 전략이다.



[그림 4] https://ratsgo.github.io/machine%20learning/2017/03/26/tree/

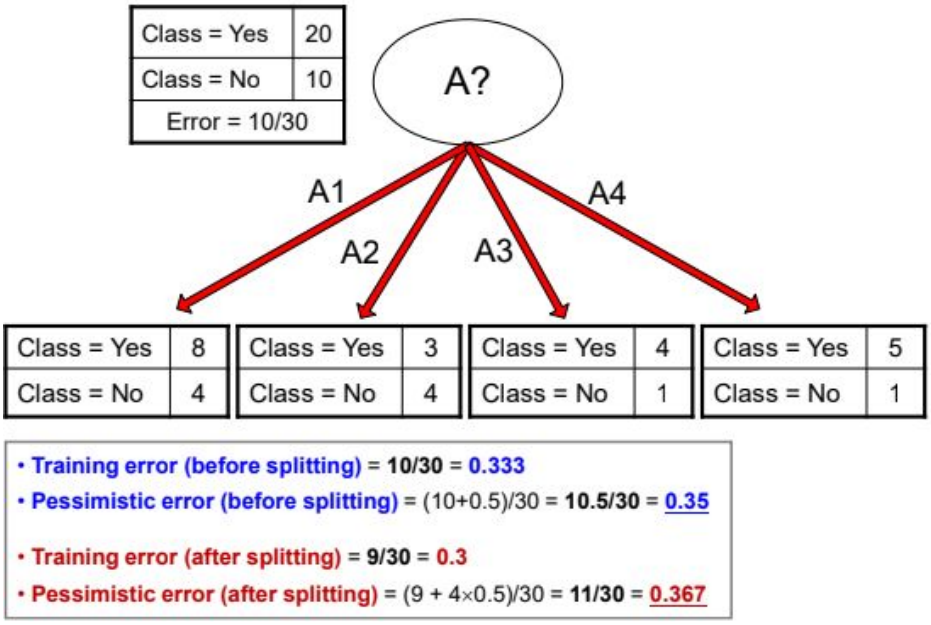
[그림 4]에서 왼쪽이 training set을 정확히 분류하지만 과적합이 된 사례이고 오른쪽이 가지치기를 통해 과적합을 방지한 경우다. 오른쪽 그림에서 y축의 19를 기준으로 첫번째 분기가 일어나고 그 분기 내에서 다시 x축의 값을 기준으로 분기가 생긴 후로는 더이상 노드를 생성하지 않음을 확인할 수 있다. 반면, 왼쪽 그림과 같은 경우에는 모델이 지나치게 복잡해서 새로운 데이터가 주어졌을 때 제대로 분류해내지 못할 가능성이 높다. 이와 같은 과적합의 문제를 방지하기 위해 모델을 형성할 때, 가지치기의 과정이 고려되어야 한다.

사전 가지치기(pre-pruning)은 fully grown tree를 만들기 전에 알고리즘을 멈추게 하는 것이다. 결정트리는 마지막 분기 이후의 모든 데이터가 같은 클래스를 가지고 있거나 모든 variable의 값이 같을 때 분기를 멈추게 된다. 사전 가지치기는 이에 더욱 제약을 가하여 정지 조건을 늘린다. 주로 현재 노드를 분기해도 impurity가 더이상 향상되지 않을 때 멈춘다는 조건을 사용한다.

사후 가지치기(post-pruning)은 결정나무를 완전히 자라게 한 후에 불필요한 가지를 잘라주는 것이다. 사후 가지치기의 조건은 sub tree를 leaf node로 합칠 때 그 전보다 error가 줄어들 때 진행하는 것이다. 이때 error는 misclassification error를 변형한 generalized error(pessimistic error)를 활용한다. 그 이유로는 우리가 줄이고자 하는 목표는 train data의 error가 아닌 test data의 error이므로 train error를 변형시켜 test error의 예측치로 만들어야하기 때문이다.

$$error_{gen}(Train) = error(Train) + \Omega \times \frac{k}{N_{train}}$$

Generalized error(Model) = Train Error(Model, Train data) + $\Omega \times$ Complexity(Model). Ω는 hyperparameter로 분석자가 알아서 정하면 된다. k는 leaf node의 개수, N_{train}은 train data의 개수이다. 위의 식은 train error에 complexity를 penalty로 두어 모델이 복잡할수록 error가 더욱 커지게된다. 이는 모델이 복잡할수록 overfitting이 발생할 확률이 크다는 것을 반영하였다. 아래의 그림을 통해서 사후 가지치기에 대해서 이해해보길 바란다.



가지치기를 한 후의 pessimistic error가 더 작으므로 위의 경우에서는 가지치기를 진행한다.

4. 결정트리의 장점과 단점

결정트리는 모델을 쉽게 시각화할 수 있어서 비전문가가 이해하기 쉽다는 장점이 있다. 또한, 각 특성이 개별적으로 처리되어서 데이터를 분할할 때 데이터 스케일의 영향을 받지 않는다. 따라서 특성의 표준화나 정규화 같은 전처리 과정이 필요없다. 또한, 계산복잡성 대비 상대적으로 높은 성능을 보여준다는 장점을 가지고 있다.

그러나 결정경계가 데이터 축에 수직이어서 특정 데이터에만 잘 작동할 가능성이 높다는 단점을 가지고 있다. 뿐만 아니라 사전 가지치기를 사용하더라도 과대적합되는 경향이 있어서 일반화 성능이 좋지 않다는 한계점도 가지고 있다. 이러한 문제점을 극복할 수 있는 방법 중 하나로 여러 모델을 연결하여 더 강력한 모델을 만드는 앙상블 기법이 대안으로 제시된다. 다음 강의에서는 이 앙상블 기법에 대해서 알아본다.

Reference

안드레아스 뮐러, 세라 가이도, 파이썬 라이브러리를 활용한 머신러닝, 한빛미디어, 2017

조엘 그루스, 밑바닥부터 시작하는 데이터 과학, 인사이트, 2016

https://bit.ly/2KXBm1T

https://ratsgo.github.io/machine%20learning/2017/03/26/tree/

https://en.wikipedia.org/wiki/Decision_tree_learning#Gini_impurity