개인적인 공부를 위해서 필요한 내용을 퍼온 것입니다. 출처는 남겨놓았습니다.

**Decision Tree** (**의사결정 나무**) 개념 정리

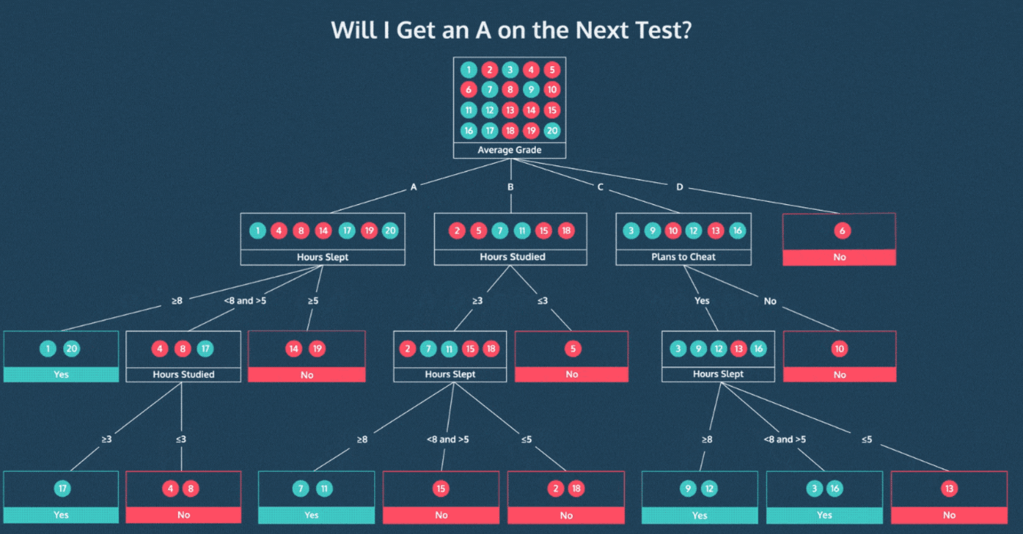
[의사결정 나무(Decision Tree) 쉽게 이해하기 - 아무튼 워라밸 (hleecaster.com)](http://hleecaster.com/ml-decision-tree-concept/)

**의사결정 나무(Decision Tree)**는 각 데이터들이 가진 속성들로부터 패턴을 찾아내서 분류 과제를 수행할 수 있도록 하는 지도학습 머신러닝 모델이다.

## 의사결정 나무란 무엇인가

예를 들면 이런 거다.

시험에서 A를 받은 데이터를 초록색 동그라미로 표현했다고 하자.



의사결정 나무는 대체 어떤 사람들이 그 A를 받았는지 나름의 기준이나 체크리스트(?) 같은 걸 만들어준다.

그래서 새로운 데이터가 들어오면 그 체크리스트를 바탕으로 하나씩 질문하고 (예를 들면 잠을 몇시간 잤냐, 공부는 몇 시간 했냐 등) 거기서 나온 답에 대한 다음 나무 줄기를 따라가다가 결국에는 ‘이건 어떤 레이블이겠구나’라고 분류를 해줄 수 있게 되는 것 일종의 스무고개 같은 거라 생각하면 편하다.

## scikit-learn 사용법

파이썬 라이브러리 scikit-learn을 사용하면 쉽게 트리를 만들 수 있다.

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

classifier = DecisionTreeClassifier**()**

classifier.fit**(**training\_points, training\_labels**)**

학습 데이터를 기반으로 트리를 생성할 때는 .fit() 메서드를 사용하면 끝이다. 다만, 의사결정 트리에서는 만약 데이터가 문자열로 이루어져 있다면 이걸 숫자로 map해서 넣어주는 게 좋다. 예를 들어 {“low”: 1, “mid”: 2, “high”: 3} 이런 식으로.

이제 새로운 데이터가 있을 때 예측을 해볼 수도 있고,

predictions = classifier.predict**(**test\_data**)**

미리 테스트 세트를 나눠놓았다면 분류 정확도를 확인해볼 수도 있다.

print**(**classifier.score**(**test\_data, test\_labels**))**

#### 한계 1.

그 순간만 보면 최선의 분할이 아니지만 이어서 나타나는 속성들에서 더 나은 분할을 발견하면 결과적으로 **더 좋은 트리를 만들 수도 있는데, 순간의 선택 때문에 그 가능성을 배제해버린다**는 뜻이다.

#### 한계 2.

또 다른 문제는 의사결정 나무가 학습 데이터에 **오버피팅** 한다는 거다. 트리의 구조가 훈련 데이터에 너무 의존적이라 현실의 데이터를 정확하게 나타내지 못한다는 뜻이다. 일반적으로 **나무가 커질수록**학습 데이터에 알맞게 모델이 만들어져서 현실 데이터로 일반화(generalization)하기 어려워지는 경향이 있다

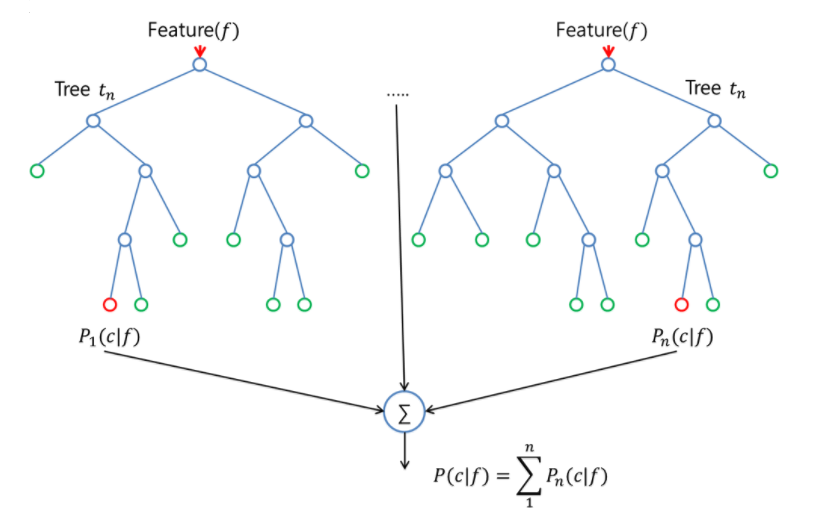
Random Forest(랜덤 포레스트) 개념 정리

[R (16) - 랜덤 포리스트 :: The best motivation is just doing it (tistory.com)](https://woochan-autobiography.tistory.com/424)

Decision Tree는 overfitting될 가능성이 높다는 약점을 가지고 있습니다. 가지치기를 통해 트리의 최대 높이를 설정해 줄 수 있지만 이로써는 overfitting을 충분히 해결할 수 없습니다. 그러므로 좀더 일반화된 트리를 만드는 방법을 생각해야합니다. 이는 Random Forest(랜덤 포레스트)의 기원이 되는 아이디어입니다. Random forest는 ensemble(앙상블) machine learning 모델입니다. 여러개의 decision tree를 형성하고 새로운 데이터 포인트를 각 트리에 동시에 통과시키며, 각 트리가 분류한 결과에서 투표를 실시하여 가장 많이 득표한 결과를 최종 분류 결과로 선택합니다. 랜덤 포레스트가 생성한 일부 트리는 overfitting될 수 있지만, 많은 수의 트리를 생성함으로써 overfitting이 예측하는데 있어 큰 영향을 미치지 못 하도록 예방합니다. 이번 포스팅에서는 랜덤 포레스트에 대해서 알아봅시다.

Decision Tree는 overfitting될 가능성이 높다는 약점을 가지고 있습니다. 가지치기를 통해 트리의 최대 높이를 설정해 줄 수 있지만 이로써는 overfitting을 충분히 해결할 수 없습니다. 그러므로 좀더 일반화된 트리를 만드는 방법을 생각해야합니다. 이는 Random Forest(랜덤 포레스트)의 기원이 되는 아이디어입니다.

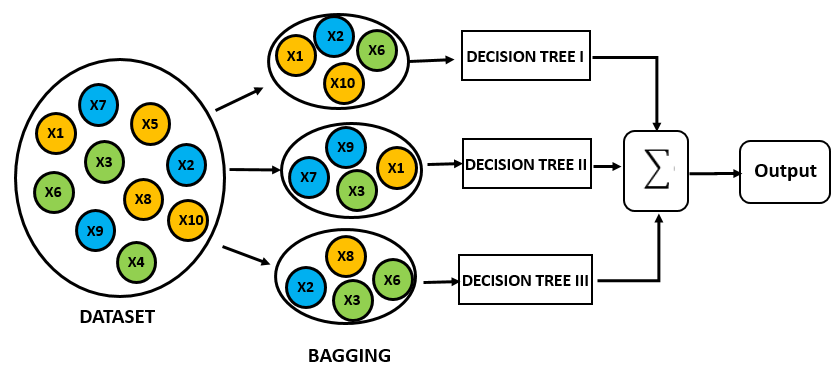
Random forest는 ensemble(앙상블) machine learning 모델입니다. 여러개의 decision tree를 형성하고 새로운 데이터 포인트를 각 트리에 동시에 통과시키며, 각 트리가 분류한 결과에서 투표를 실시하여 가장 많이 득표한 결과를 최종 분류 결과로 선택합니다.

[](https://miro.medium.com/max/1638/1*9Cw6GyeSKOpWqnT91lygdA.png)Source: [Analytics Vidhya](https://medium.com/greyatom/a-trip-to-random-forest-5c30d8250d6a)

랜덤 포레스트가 생성한 일부 트리는 overfitting될 수 있지만, 많은 수의 트리를 생성함으로써 overfitting이 예측하는데 있어 큰 영향을 미치지 못 하도록 예방합니다. 이번 포스팅에서는 랜덤 포레스트에 대해서 알아보도록 하겠습니다.

**Bagging**

랜덤 포레스트는 제일 먼저 *bagging* 이라는 과정을 거칩니다. Bagging은 트리를 만들 때 training set의 부분집합을 활용하여 형성하는 것을 말합니다. 예를 들어, training set에 1000 개의 데이터가 존재한다고 가정하면 각 트리를 생성할 때 100 개의 데이터만 임의로 선택하여 트리를 만드는데 활용할 수 있는 것입니다. 즉 모든 트리는 각기 다른 데이터를 바탕으로 형성되지만 모두 training set의 부분집합입니다.

[](https://miro.medium.com/max/1678/1*Wf91XObaX2zwow7mMwDmGw.png)Source: [Medium](https://medium.com/greyatom/a-trip-to-random-forest-5c30d8250d6a)

이렇게 100 개의 데이터를 임의로 선택할 때 한가지 중요한 것은 바로 중복을 허용한다는 것입니다(with replacement). 이렇게 중복을 허용함으로써 1000 개의 training set에서 100 개만 뽑기보다 1000 개씩 매번 뽑아도 unique한 데이터셋을 형성할 수 있으므로 N 개의 training set이 있다면 임의로 N 개의 데이터를 중복을 허용하여 선택함으로써 각 트리를 형성합니다.

**Bagging Features**

랜덤 포레스트는 트리를 형성할 때 데이터셋에만 변화를 주는 것이 아닌 feature를 선택하는데 있어서도 변화를 줍니다. Feature를 선택할때도 기존에 존재하는 feature의 부분집합을 활용합니다. 예를 들어, 자동차 데이터셋에 다음과 같은 feature들이 있고 이를 통해 자동차의 등급(very good, good, acceptable, unacceptable)을 분류한다고 생각해봅시다.

* 가격
* 유지 보수 비용
* 문의 개수
* 탑승 인원 수
* 트렁크 사이즈
* 안전 등급

위 데이터를 제일 처음 분류할 때, 고려할 feature를 임의로 선택하여 가격과 문의 개수, 그리고 안전 등급 3개의 feature중에 선택할 수 있습니다. 여기서 선택된 best feature로 데이터를 분류한 뒤에는 마찬가지로 고려할 feature를 임의로 선택하여 유지 보수 비용과 문의 개수, 그리고 트렁크 사이즈 3개의 feature중에 선택할 수 있습니다. 이러한 과정을 트리가 만들어질 때 까지 반복합니다.

위 예에서는 단지 3개의 feature만을 임의로 선택하였는데 이는 예시로 든 feature 수가 적기 때문이고, 일반적으로 M 개의 feature가 존재할 때, 임의로 선택하는 feature의 수는 √MM 을 활용합니다. 즉, 25 개의 feaure가 존재한다면 임의로 5개의 feature를 선택하고 그 중에서 가장 information gain이 높은 feature를 선택하여 데이터를 분류하게 됩니다.

**Classify**

여러개의 트리를 형성하였다면 이제 classify를 수행할 수 있습니다. 예를 들어, 8 개의 트리를 형성하고 임의의 자동차 데이터를 각 트리에 전달하였을 때 나온 결과가 다음과 같다고 가정해봅시다.

["very good", "very good", "good", "very good", "acceptable", "very good", "good", "very good"]

위 결과에서 very good 이 5 번 등장하여 가장 많은 득표수를 나타냈으므로 임의의 자동차는 very good 으로 분류될 것입니다. Test 단계에서는 모든 데이터셋에 대해서 위 과정을 거치게 되고, 그렇게 나온 결과와 ground truth를 비교하여 accuracy를 측정하게 됩니다.