3.3 Random Forest

- 결정트리
  - 정답에 가장 빨리 도달하는 예/아니오 질문목록을 학습하는 게 목적
  - 정보 획득 (Information gain)을 최대화하는 질문을 적용.
  - 스무고개와 비슷

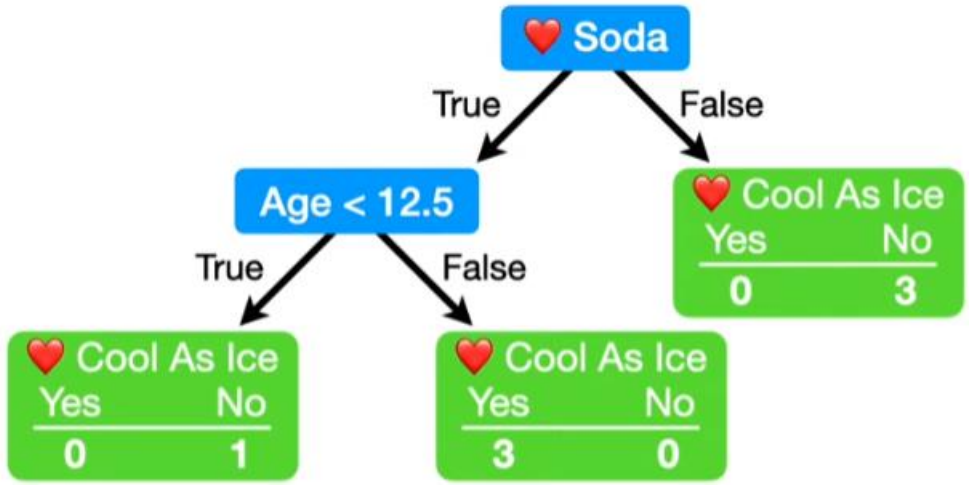
- 발전과정
  - ID3 알고리즘 발표(R. Quinlan, 1986년)
  - AdaBoosting발표(Freund, Schapire, 1997)
  - RandomForest 발표(Leo Breiman, 2001)
  - Gradient Boosting발표(2014)

3.3 Random Forest – decision tree

Q : 차가운 음료를 좋아하는 사람을 분류하려면?

Loves Popcorn	Loves Soda	Age	Loves Cool As Ice
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No

A : Soda 를 좋아하는지 확인,
Soda를 좋아하면 나이가 12.5살 이하인지 확인



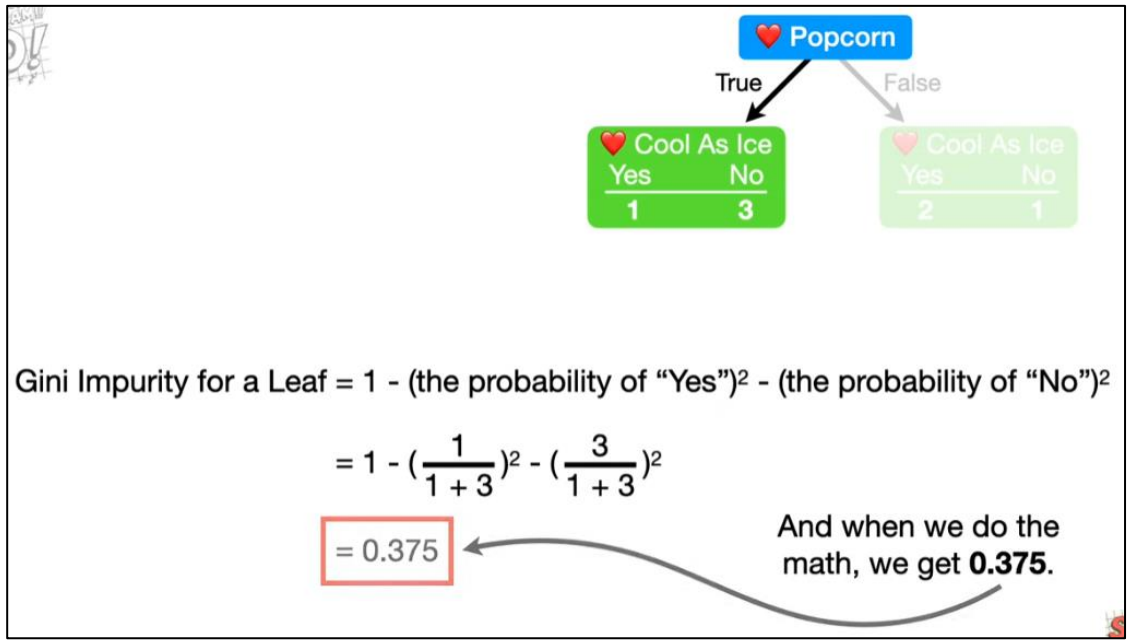
https://www.youtube.com/watch?v=_L39rN6gz7Y&list=PLblh5JKOoLUICTaGLRoHQDuF_7q2GfuJF&index=42

3.3 Random Forest – decision tree

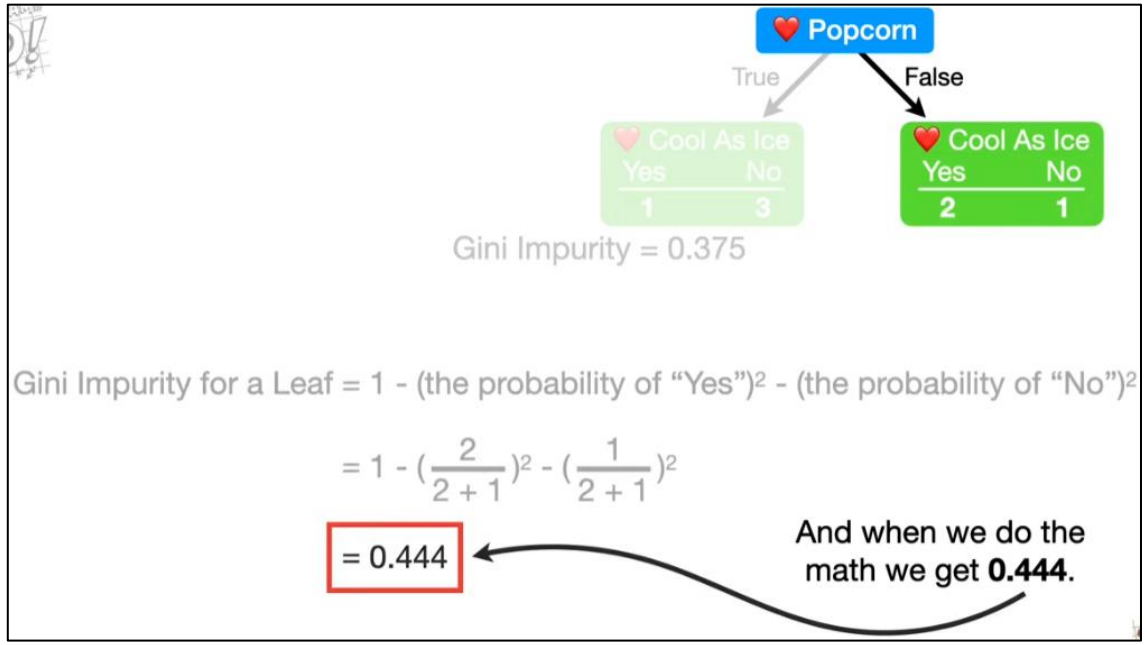
1. 팝콘 좋아하는 사람들의 목표값에 대한 Gini imputity(불순도) 계산

Gini Impurity for a Leaf = 1 - (the probability of "Yes")^2 - (the probability of "No")^2

1.1 'Loves Popcorn' == True 인 경우 Gini = 0.375



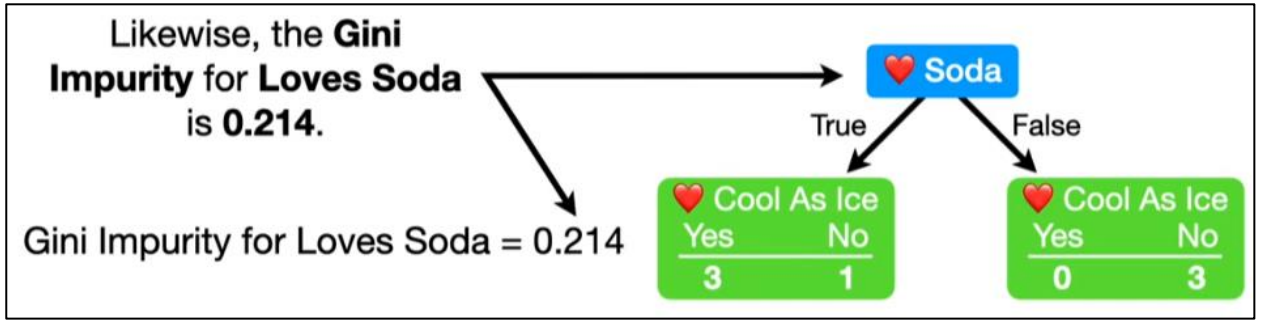
1.2 'Loves Popcorn == false 인 경우 Gini = 0.444



1.3 양쪽의 인원이 다르므로 가중평균 : $(\frac{4}{4+3})\,0.375 + (\frac{3}{4+3})\,0.444 = 0.405$

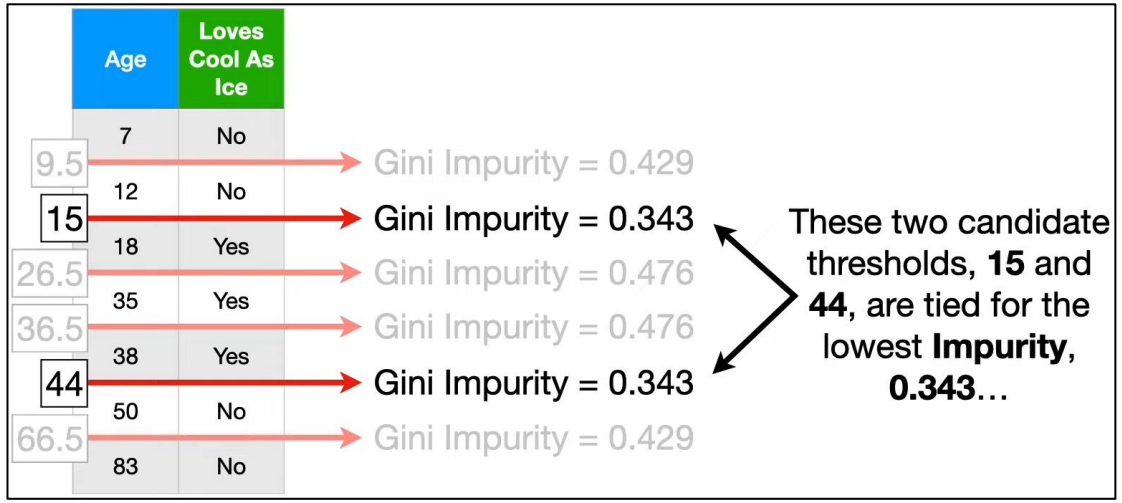
3.3 Random Forest – decision tree

2. Soda 좋아하는 사람들에 대해 Gini imputity(불순도) 계산



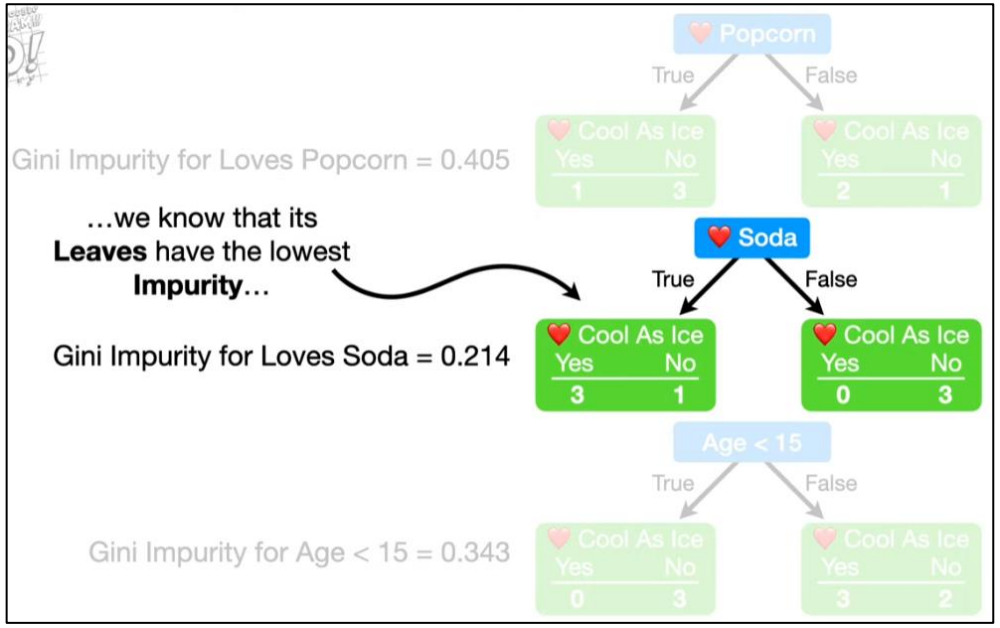
3. 나이 기준 불순도 계산

** 연속숫자형 또는 범주형 특성은 각각 계산 후 가장 낮은 기준 선택



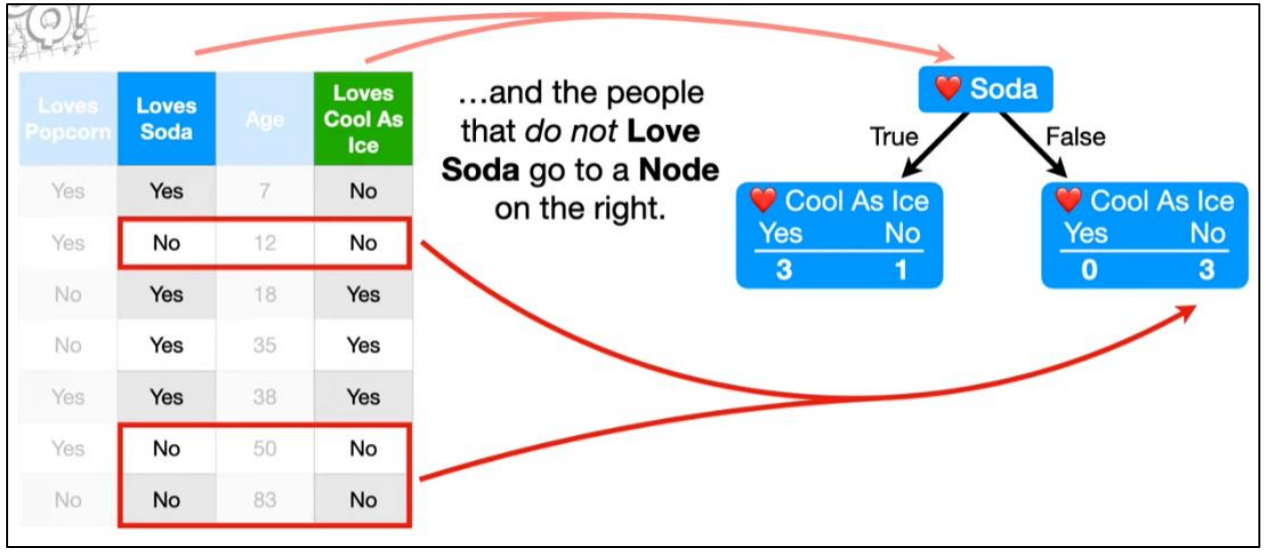
3.3 Random Forest – decision tree

4-1. 소다 좋아하는지 여부의 Gini impurity가 가장 낮음.



4.2 소다 좋아하는지 여부를 시작노드로 설정

→ 하위 노드에 대해 같은 작업 반복



3.3 Random Forest – decision tree

- 순수 노드(pure node) : 불순도가 0인 노드
- 모든 leaf node가 순수노드가 될 때까지 트리를 생성하면 복잡하고 과대적합됨.
- 가지치기 작업으로모델의 성능 개선 필요.
- 가지(pruning)치기 종류
  - 트리의 최대깊이(max_depth) 제한
  - 리프노드의 최대갯수 제한
  - 노드를 분할하기위한 최소갯수 지정
- 가지치기를 통한 성능개선
  - 암데이타셋에 제한없는 트리 생성 : 1.00, 0.937 성능 보임(과대적합, cell 4)
  - max_depth = 4 → 0.988, 0.951 (cell 5)
- 트리의 특성중요도
  - 트리를 만드는 결정에 각 특성이 얼마나 중요한지 평가지표(cell 9, 10)

3.3 Random Forest – 앙상블 모델의 필요성

- decision tree의 장단점
  - 비전문가도 모델을 이해하기가 비교적 쉽다.
  - 정규화,표준화 등의 데이터 전처리가 필요없다.
  - 이진특성과 연속특성이 혼합되어있어도 잘 작동.

  - 훈련데이타 범위 밖의 포인트에 대한 예측 불가(cell 12~)
  - 새 샘플의 예측성능은 떨어진다(가지치기를 해도 과대적합되는 경향이 있다).
  - → 앙상블 기법 필요.

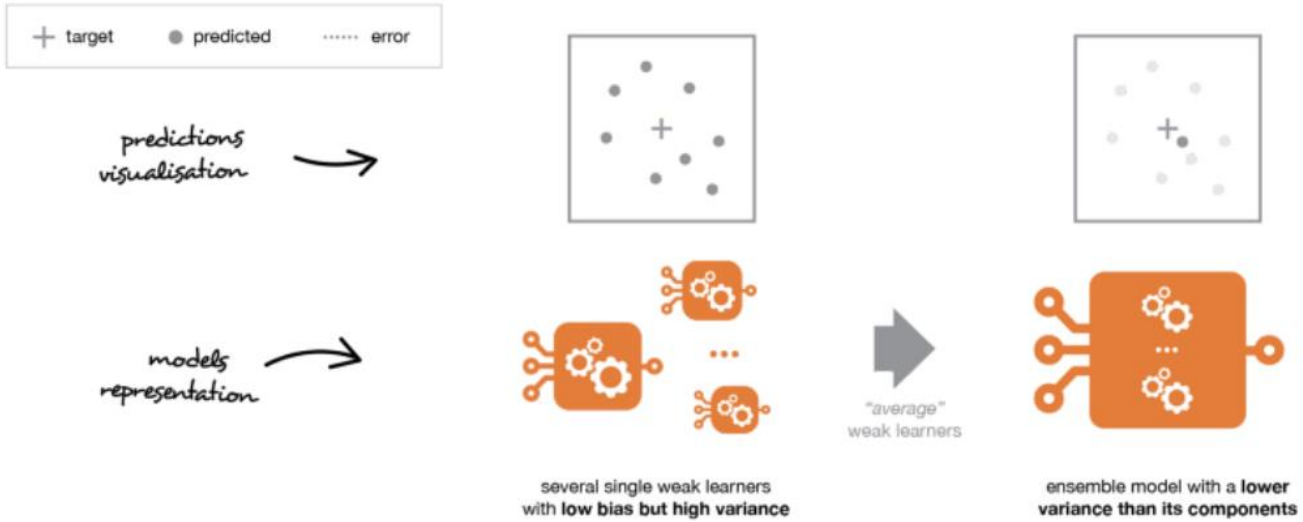
- 앙상블 모델
  - 좋은 모델은 낮은 bias, 낮은 variance를 가져야 함.
  - 일반적으로 bias를 낮추면 variance가 높아지고(과대적합), 반대의 경우 variance가 낮아지는 상충관계를 가짐.
  - 결정트리는 낮은 bias, 높은 variance 특징이 있음(과대적합).
  - 과대적합 모델을 많이 만들어서 결과를 합하면 variance를 낮출 수 있음.

3.3 Random Forest – 앙상블 유형

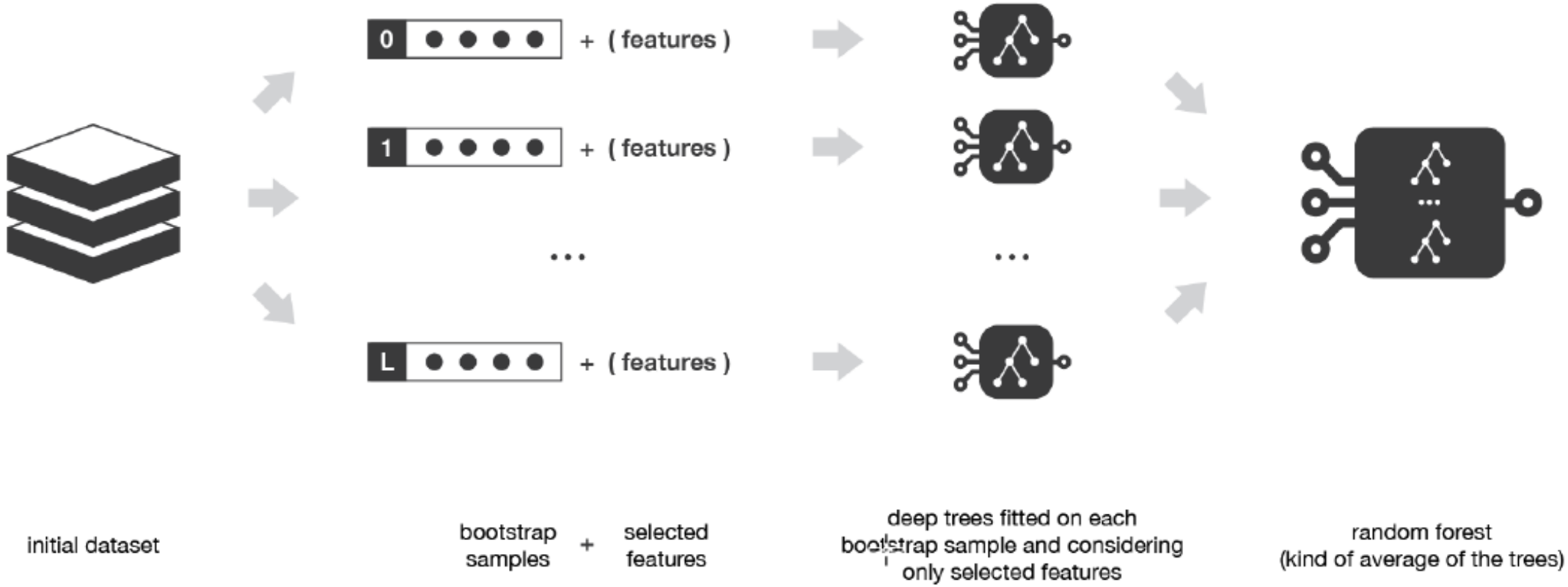
- bagging
  - 같은 유형의 (분산이 큰) weak learner 사용
  - 각 weak learner는 부트스트랩 방식의 데이터 서브셋을 사용하고 독립적(병렬) 학습
  - 각 weak learner 결과값의 평균을 취하는 방식으로 예측.
  - 분산을 줄이는데 유리.
  - Random Forest
- boosting
  - 같은 유형의 weak learner 사용
  - 각 weak learner는 이전 weak learner의 오답 데이타셋을 반영하여 조정되는 방식으로 학습.
  - bias를 줄이는데 유리(동시에 variance 도 줄이는 효과 가능)
  - AdaBoosting, GradBoosting
- stacking
  - 다른 특징의 weak learner 사용
  - 각 weak learner는 독립적(병렬) 학습
  - bias를 줄이는데 유리(동시에 variance 도 줄이는 효과 가능)

3.3 Random Forest

- Random forest 개요
  - ➢ 생성된 트리는 일부 데이터에 과대적합됨. → 과대적합 약점이 있는 결정트리를 보완하기 위한 기법.
  - ➢ 과대적합 경향 (low bias, high variance) 의 서로 다른 트리를 많이 만들어서 서로 다른 관점에서 과대적합되도록 구성하여 종합함으로서 과대적합을 줄임.
- 구현방법
  - ➢ 과대적합을 피하기 위한 기법 ==> real world samples 를 최대한 반영하도록 하는 데이터 샘플 설계 기법 필요.
  - ➢ 1) 부트스트랩 샘플링 : n 개의 샘플 중 중복을 허용하여 다른 n 개의 데이터셋을 만듬. - 트리마다 다른 학습데이타를 제공.
  - ➢ 2) 특성을 무작위로 선택.적용 : 트리의 노드마다 다른 특성조합을 적용하여 다른 학습모델을 생성. - 결측치 데이터 보완.



3.3 Random Forest 작동방식



3.3 Random Forest

1. Bootstrap dataset 생성
   - 같은 크기의 데이타셋 생성
   - 샘플은 중복해서 선택될 수있음.(통계적으로 33 % 는 미선택)

Original Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	Yes	167	Yes

Bootstrapped Dataset

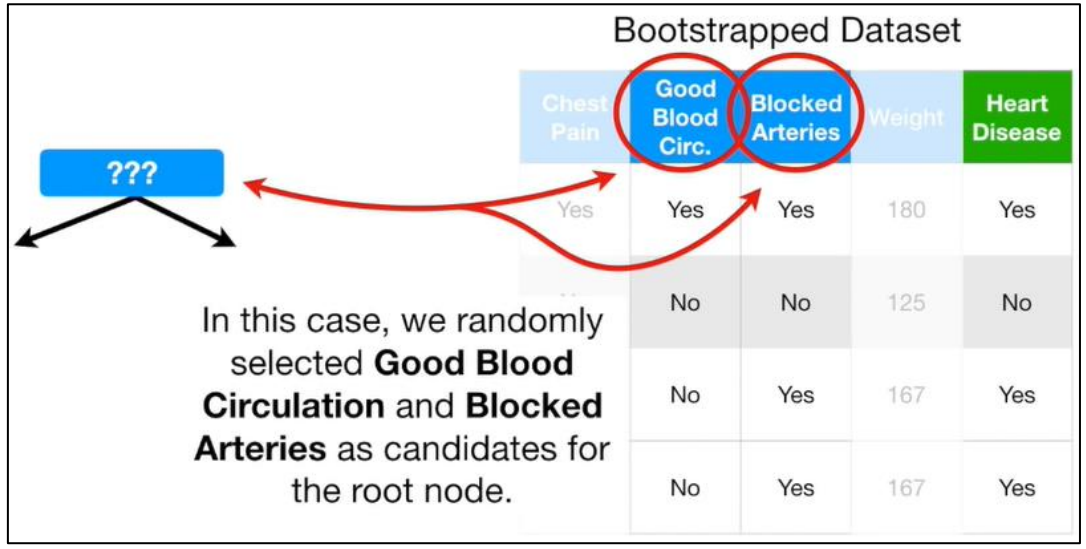
Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	Yes	167	Yes
Yes	No	Yes	167	Yes

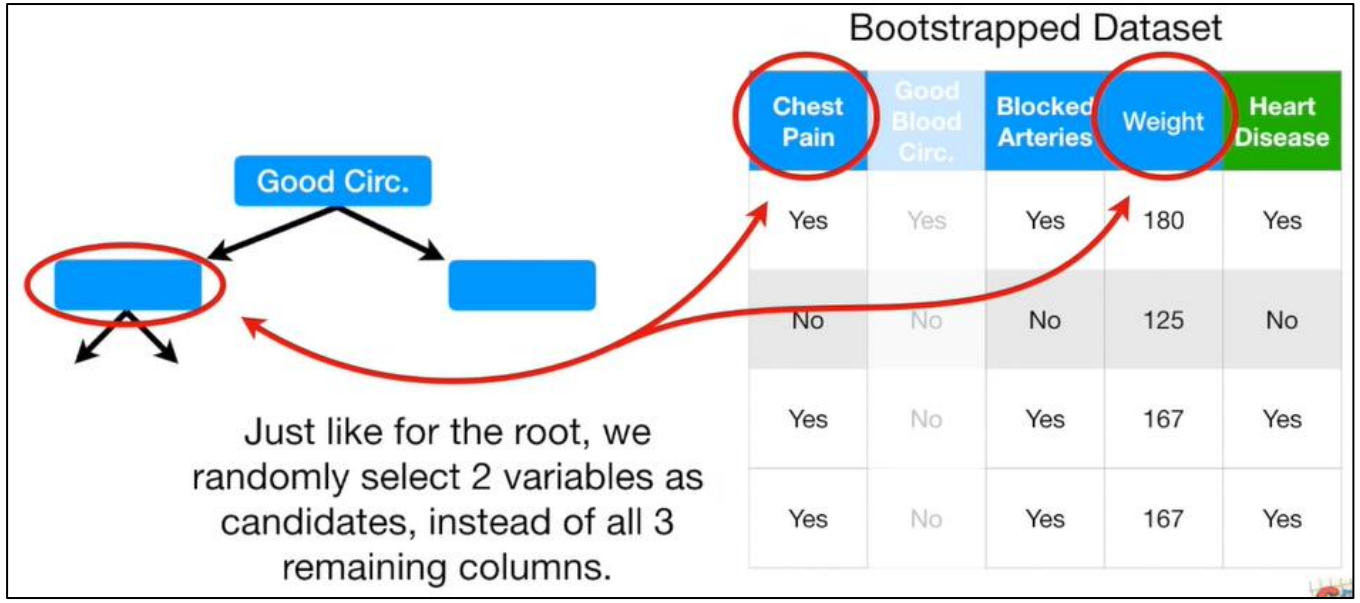
https://www.youtube.com/watch?v=J4Wdy0Wc_xQ&t=190s

3.3 Random Forest

2. 루트노드 특성 선택
   - 임의의 n개 특성선택(여기서는 2개)
   - 정보이득을 계산하여 루트노드 선택
   - 트리 생성

3. 2차 노드 특성 선택
   - 나머지 3 특성 중 임의의 2개 특성선택
   - 정보이득을 계산하여 노드 선택
   - 트리 확장

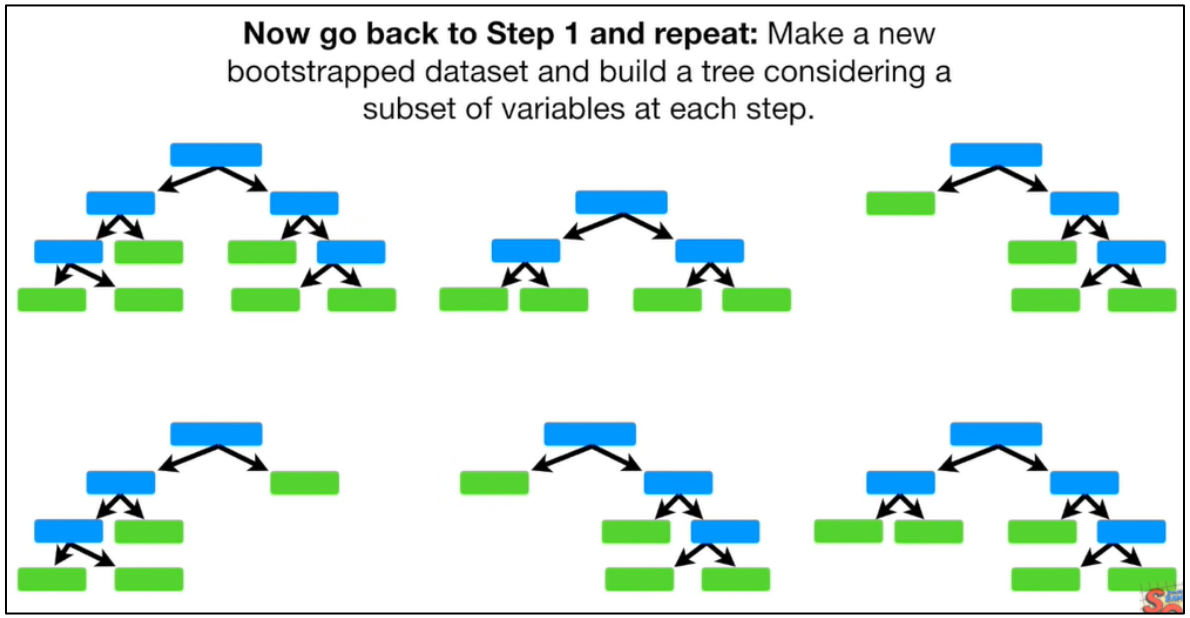




3.3 Random Forest

4. 하위 노드와 리프 생성
   - 모든 하위 노드에 대하여 선택되지 않은 특성 중 임의의 2개 특성 선택
   - 정보이득을 계산하여 노드 선택
   - 지정된 깊이까지 트리 생성

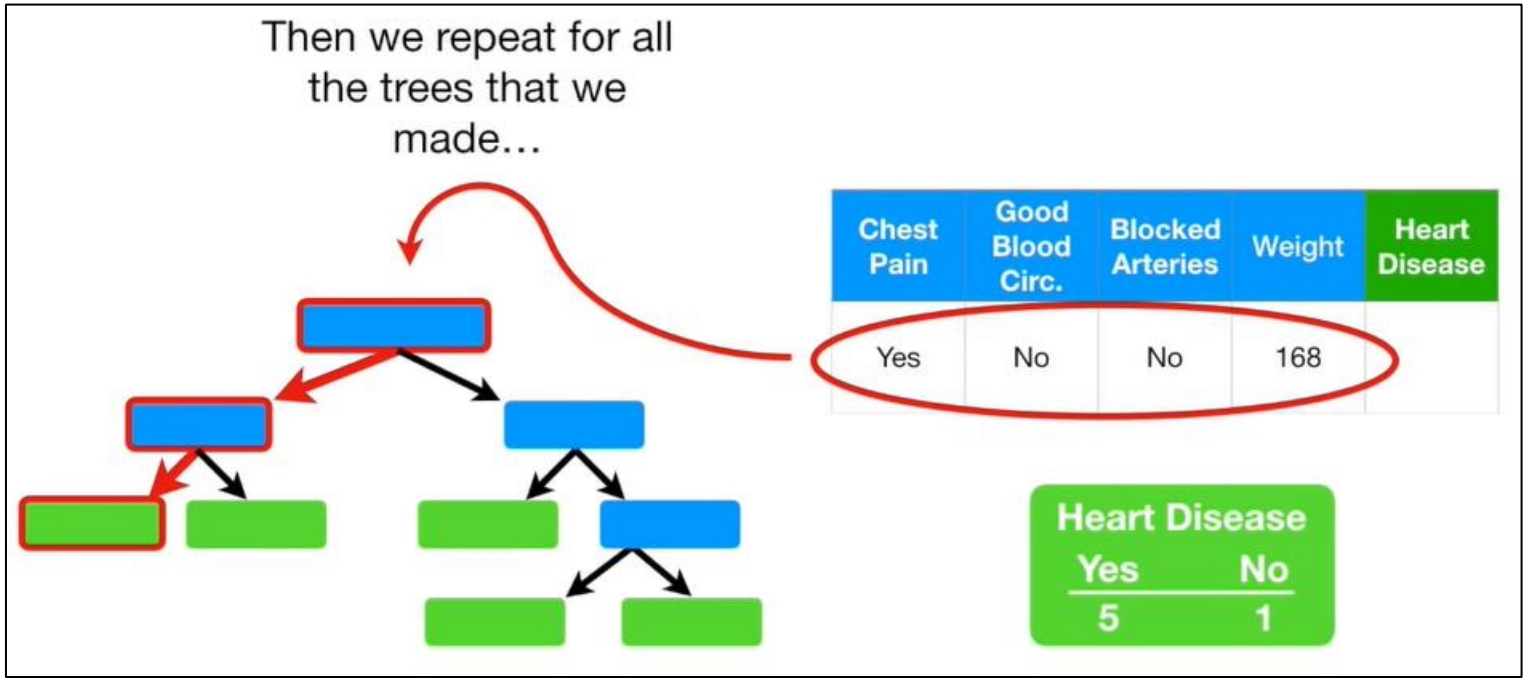
5. 위 과정을 지정된 트리 개수만큼 반복 작업하여 생성
   - 예제에서는 6 개의 트리 생성



3.3 Random Forest

6. 예측

- 예측 문제를 생성된 6개의 트리에 대입하여 예측
- 가장 빈도가 높은 예측값을 결과값으로 선택



7. 성능 개선

- 선택되는 임의의 특성 개수를 변화시켜가면서 최적 특성갯수 결정

3.3 Random Forest

- Random_Forest로 만들어진 결정경계는 개별트리에 비해 안정적인 경계를 보임(cell 17)
- 암데이타셋에 적용했을 때 1.00, 0.972(cell 18)
  - 결정트리 사용시 성능치는 0.951 이었음.
- 특성중요도 분석(cell 19)
  - 가장 중요한 특징으로 worst perimeter 선택(결정트리는 worst radius)
  - 무작위성으로 인하여 가능성 있는 많은 경우를 고려
  - 데 넓은 시각으로 데이터를 볼 수 있음.
- 중요 매개변수
  - n_estimators : 많을수록 좋음.
  - max_features : 무작위성 결정. 작을수록 과대적합 방지
  - 분류 기본값 : max_features = sqrt(n_features)
  - 회귀 기본값 : max_features = n_features(어떤데서는 n_features/3)
- 장단점
  - 단일트리에 비해 우수한 성능
  - 단일트리에 비해 시각화 및 해석이 어려움(많은 트리, 특성의 일부만 사용하므로 깊은 트리 생성)
  - 많은 메모리 및 CPU 자원 사용하고 훈련과 예측 속도가 느림

3.4 Boosting

- Boosting : 이전 weak learner 의 예측오류를 교정하면서 다음 weak learner를 생성하는 점진적인 학습 방법
- low variance, high bias 기본모델(얕은 트리)을 만든 후 bias를 줄이는 방향으로 학습(동시에 분산을 줄이는 효과도 가능)
- 이전 트리의 오차를 보완하는 방식으로 순차적으로 트리를 만듬.
- 데이터 무작위성이 없는 대신 강력한 가지치기 적용 – 최대 5 정도의 깊지 않은 트리를 사용하므로 자원사용량이 적고 빠름.
- Random forest보다 매개변수 설정에 민감하지만 더 높은 성능 제공.

3.4.1 AdaBoost

- Adaptive Boost
- 깊이가 2인 stump(weak learner) 사용 (RF는 상대적으로 깊은 트리 형성).
- 예측시 stump 별로 성능의 가중치가 다름 (RF는 트리들이 같은 가중치 가짐).
- 각 stump는 이전 stump의 오류를 반영하여 형성됨 - RF는 각 트리가 독립적.
- 한 stump 에서 잘 못 예측한 샘플의 비중을 높여서 다음 stump에 반영함.

데이타셋

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease
Yes	Yes	205	Yes
No	Yes	180	Yes
Yes	No	210	Yes
Yes	Yes	167	Yes
No	Yes	156	No
No	Yes	125	No
Yes	No	168	No
Yes	Yes	172	No

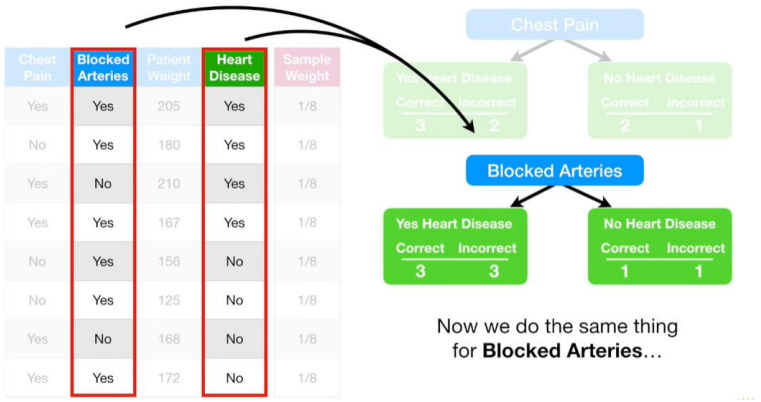
1. 샘플 가중치 초기화(1/n)

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	1/8
No	Yes	180	Yes	1/8
Yes	No	210	Yes	1/8
Yes	Yes	167	Yes	1/8
No	Yes	156	No	1/8
No	Yes	125	No	1/8
Yes	No	168	No	1/8
Yes	Yes	172	No	1/8

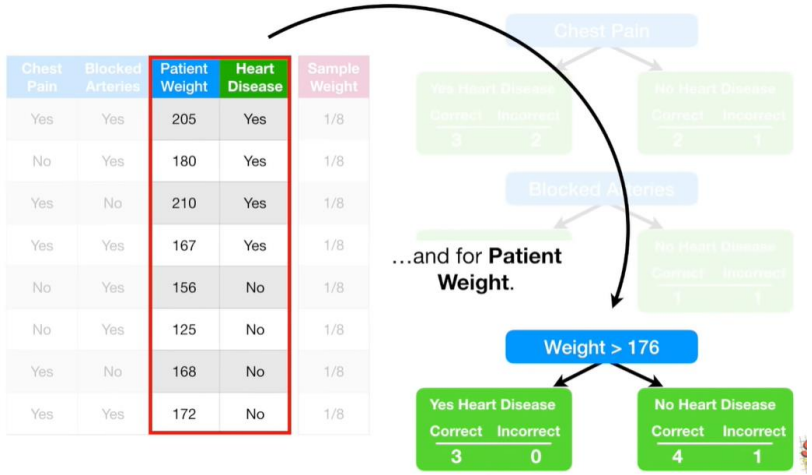
https://www.youtube.com/watch?v=LsK-xG1cLYA&list=PLblh5JKOoLUICTaGLRoHQDuF_7q2GfuJF&index=53

3.4.1 AdaBoost

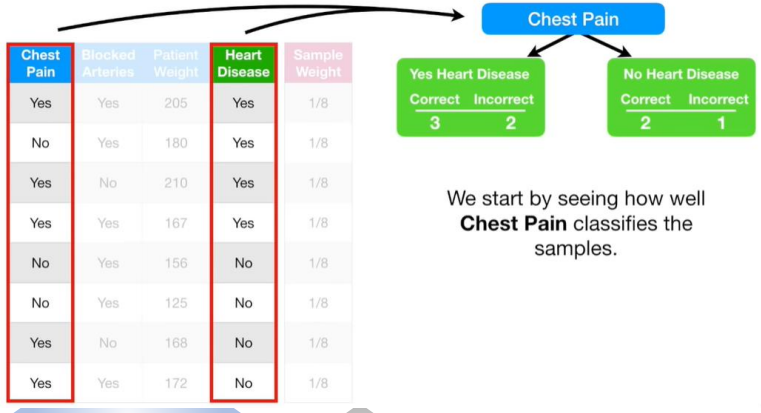
2-1. Blocked Arteries 특성으로 GI 계산



2-2. 체중 특성으로 GI 계산



2-3. Chest pain 특성으로 GI 계산

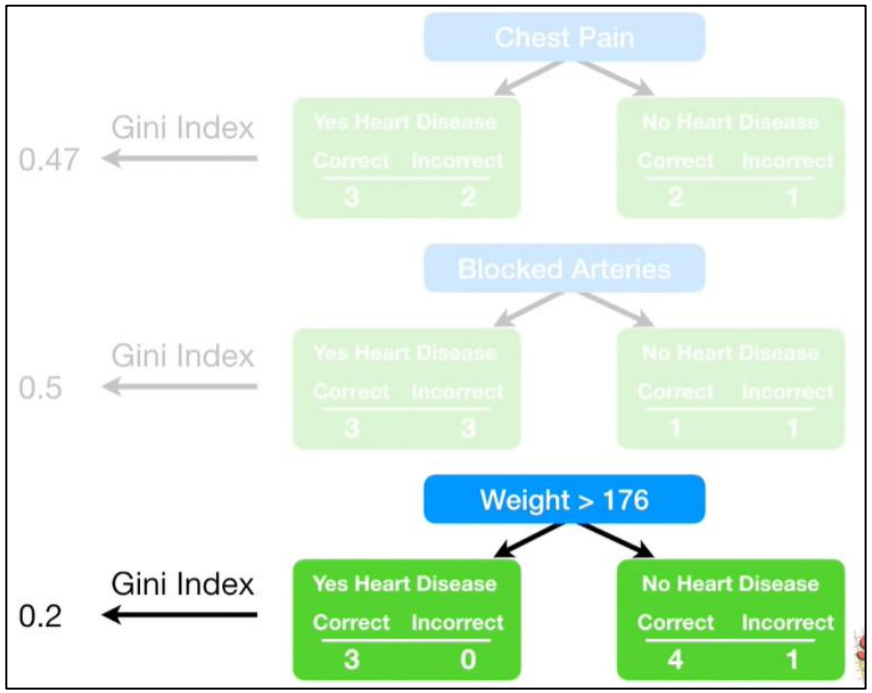


가슴통증 있는 경우 :
- 정상분류 : 3, 오분류 : 2

가슴통증 없는 경우 :
- 정상분류 : 2, 오분류 : 1

3.4.1 AdaBoost

3. Gini Index 계산 ➔ 체중이 첫번째 stump가 됨.



4. 가중치 갱신 - stump의 Amount of Say(잘 맞추는 정도) 계산

$$\text{Amount of Say} = \frac{1}{2}\log\left(\frac{1 - \text{Total Error}}{\text{Total Error}}\right)$$

Total Error : 틀린 비율 = 1/8
Amount of Say = 0.97

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	1/8
No	Yes	180	Yes	1/8
Yes	No	210	Yes	1/8
Yes	Yes	167	Yes	1/8
No	Yes	156	No	1/8
No	Yes	125	No	1/8
Yes	No	168	No	1/8
Yes	Yes	172	No	1/8

With **Patient Weight > 176**, the **Total Error** is **1/8**, so we just plug and chug…

$$\text{Amount of Say} = \frac{1}{2}\log(7) = 0.97$$

3.4.1 AdaBoost

4. 가중치 갱신 – 샘플별 가중치 계산

틀린 샘플의 가중치 계산 : (높임)

$$\text{New Sample Weight} = \text{sample weight} \times e^{\text{amount of say}}$$

$$= \frac{1}{8} e^{\text{amount of say}}$$

$$= \frac{1}{8} e^{0.97} = \frac{1}{8} \times 2.64 = 0.33$$

맞춘 샘플의 가중치 계산 : (낮춤)

$$\text{New Sample Weight} = \text{sample weight} \times e^{\text{-amount of say}}$$

$$= \frac{1}{8} e^{\text{-amount of say}}$$

$$= \frac{1}{8} e^{-0.97} = \frac{1}{8} \times 0.38 = 0.05$$

3.4.1 AdaBoost

4. 가중치 갱신
 –틀린 샘플의 가중치를 높게 갱신. 전체 합은 1로.

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight	New Weight	Norm. Weight
Yes	Yes	205	Yes	1/8	0.05	0.07
No	Yes	180	Yes	1/8	0.05	0.07
Yes	No	210	Yes	1/8	0.05	0.07
Yes	Yes	167	Yes	1/8	0.33	0.49
No	Yes	156	No	1/8	0.05	0.07
No	Yes	125	No	1/8	0.05	0.07
Yes	No	168	No	1/8	0.05	0.07
Yes	Yes	172	No	1/8	0.05	0.07

3.4.1 AdaBoost

5. 가중치를 반영하여 샘플을 다시 생성 –
   - 전체 샘플수는 같고, 가중치가 큰 샘플은 그 비중만큼 추가복제하여 샘플에 편입
   - 샘플 수만큼 0~1 사이의 난수 생성.

   → 누적 가중치 분포 대상으로 샘플 선정(가중치 크기만큼 선정확률 높아짐)

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	0.07
No	Yes	180	Yes	0.07
Yes	No	210	Yes	0.07
Yes	Yes	167	Yes	0.49
No	Yes	156	No	0.07
No	Yes	125		
Yes	No	168		
Yes	Yes	172		

Ultimately, this sample was added to the new collection of samples **4** times, reflecting its larger **Sample Weight**.

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease
No	Yes	156	No
Yes	Yes	167	Yes
No	Yes	125	No
Yes	Yes	167	Yes
Yes	Yes	167	Yes
Yes	Yes	172	No
Yes	Yes	205	Yes
Yes	Yes	167	Yes

3.4.1 AdaBoost

6. 다음 stump 작업을 위해 가중치 초기화 –
   - 틀린 샘플 가중치 반영됨

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
No	Yes	156	No	1/8
Yes	Yes	167	Yes	1/8
No	Yes	125	No	1/8
Yes	Yes	167	Yes	1/8
Yes	Yes	167	Yes	1/8
Yes	Yes	172	No	1/8
Yes	Yes	205	Yes	1/8
Yes	Yes	167	Yes	1/8

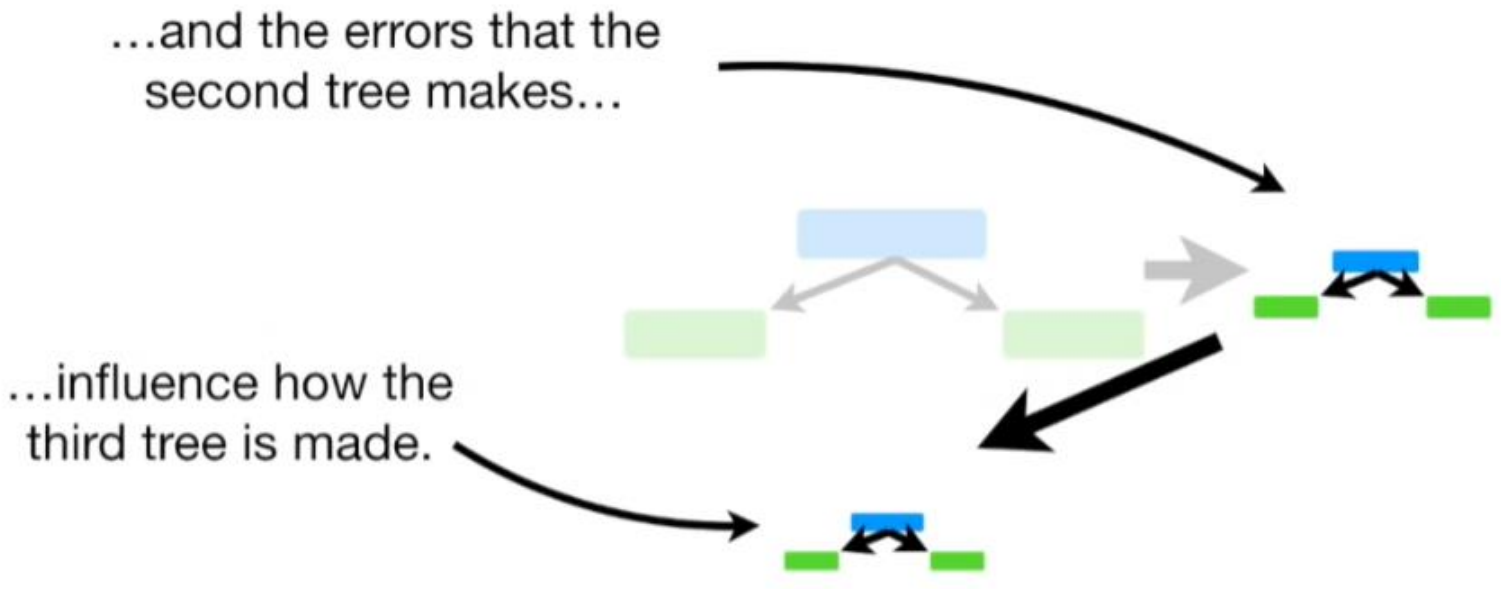
3.4.1 AdaBoost

7. 보정된 데이타셋 대상으로 다음 stump 생성

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
No	Yes	156	No	1/8
Yes	Yes	167	Yes	1/8
No	Yes	125	No	1/8
Yes	Yes	167	Yes	1/8
Yes	Yes	167	Yes	1/8
Yes	Yes	172	No	1/8
Yes	Yes	205	Yes	1/8
Yes	Yes	167	Yes	1/8

3.4.1 AdaBoost

이전 stump의 예측결과의 틀린 샘플에 가중치를 두는 방식으로 순차적으로 stump 생성.



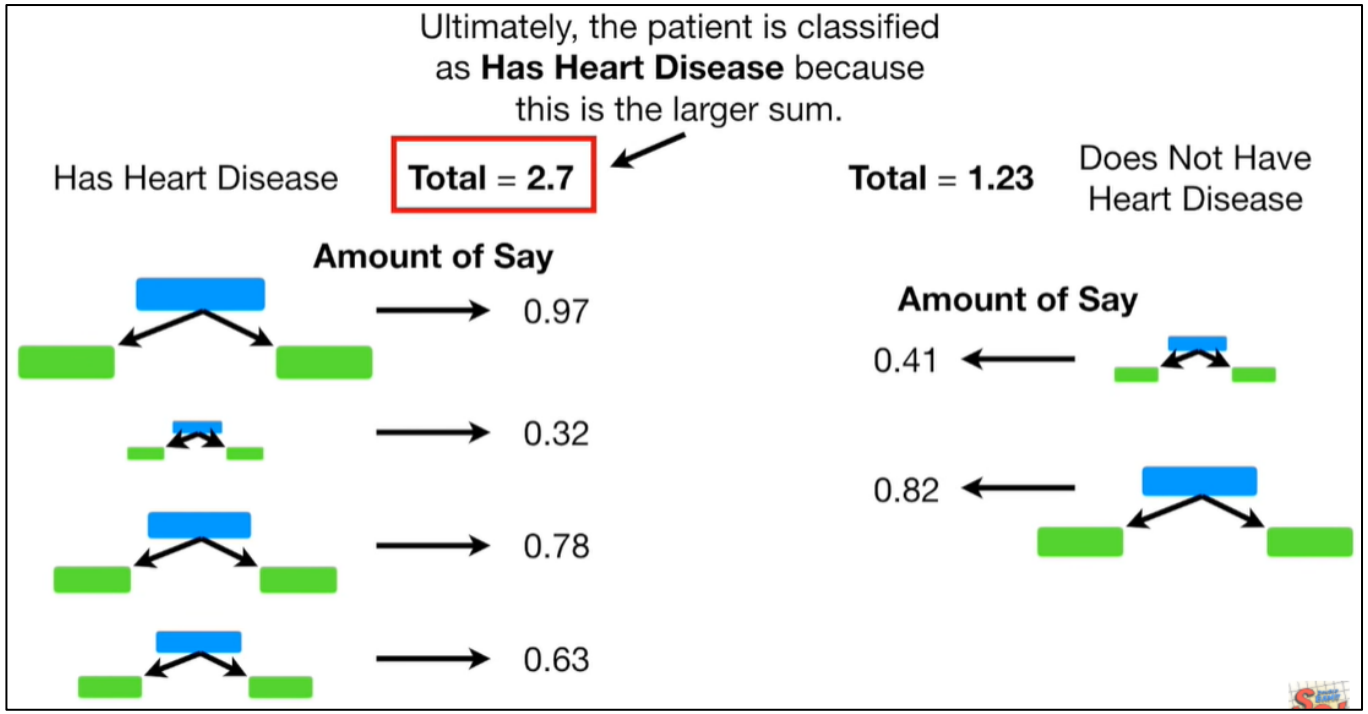
3.4.1 AdaBoost

8. 예측

– 학습에 사용된 stump들이 개별적으로 예측.

- 각 stump는 가중치 값(Amount of Say)을 가짐.

- amount of Say 합이 큰 결과를 선택.



3.4.1 AdaBoost

- scikit-learn의 max_depth 기본값
  - AdaBoostClassifier() : max_depth=1
  - AdaBoostRegressor() : max_depth=3
- 순차적으로 학습하므로 n_jobs = None

- 실습
  - two_moons 결정경계 : max_depth = 1 이므로 하나의 분할선(cell 20)
  - 암데이타셋에 적용(cell 21)
  - 1.000, 0.986 성능 보임.
  - 특성중요도를 보면 area error 가 가장 중요함.(cell 22)

3.4.2 Gradient Boost

- Gradient : 주어진 함수의 편미분 벡터. 음의 값이면 원함수의 값이 줄어드는 방향을 의미함(손실함수에 유용).
- 학습예측값과 실제 값의 잔차값을 이용.
- 잔차를 낮추는 방향으로 다음 트리 생성.
- 정의된 손실함수에 대하여 경사하강법(Gradient Descent)을 적용하여 다음 생성될 트리가 예측해야 할 값 보정.
- weak learner는 leaf 개수를 8 ~ 32 개 사용.
- 예측시 stump 별로 성능의 가중치가 다름.
- 샘플 선정 기본값은 전체이나 트리별로 부분 샘플을 지정할 수 있음.
- 사용특성도 기본값은 전체 특성이나 일부를 사용할 수도 있음.

https://www.youtube.com/watch?v=jxuNLH5dXCs&list=PLblh5JKOoLUICTaGLRoHQDuF_7q2GfuJF&index=56

3.4.2 Gradient Boost

데이타셋

Likes Popcorn	Age	Favorite Color	Loves Troll 2
Yes	12	Blue	Yes
Yes	87	Green	Yes
No	44	Blue	No
Yes	19	Red	No
No	32	Green	Yes
No	14	Blue	Yes

1. 초기 예측값(리프) 설정
   - 임의의 예측값으로 시작하여 목표값을 찾아가는 것이 목표
   - Troll2를 좋아하는 사람들의 log(odds) 계산
   - log(odds) = log(# of yes / # of No)

Likes Popcorn	Age	Favorite Color	Loves Troll 2
Yes	12	Blue	Yes
Yes	87	Green	Yes
No	44	Blue	No
Yes	19	Red	No
No	32	Green	Yes
No	14	Blue	Yes

...then the **log(odds)** that someone **Loves Troll 2** is...

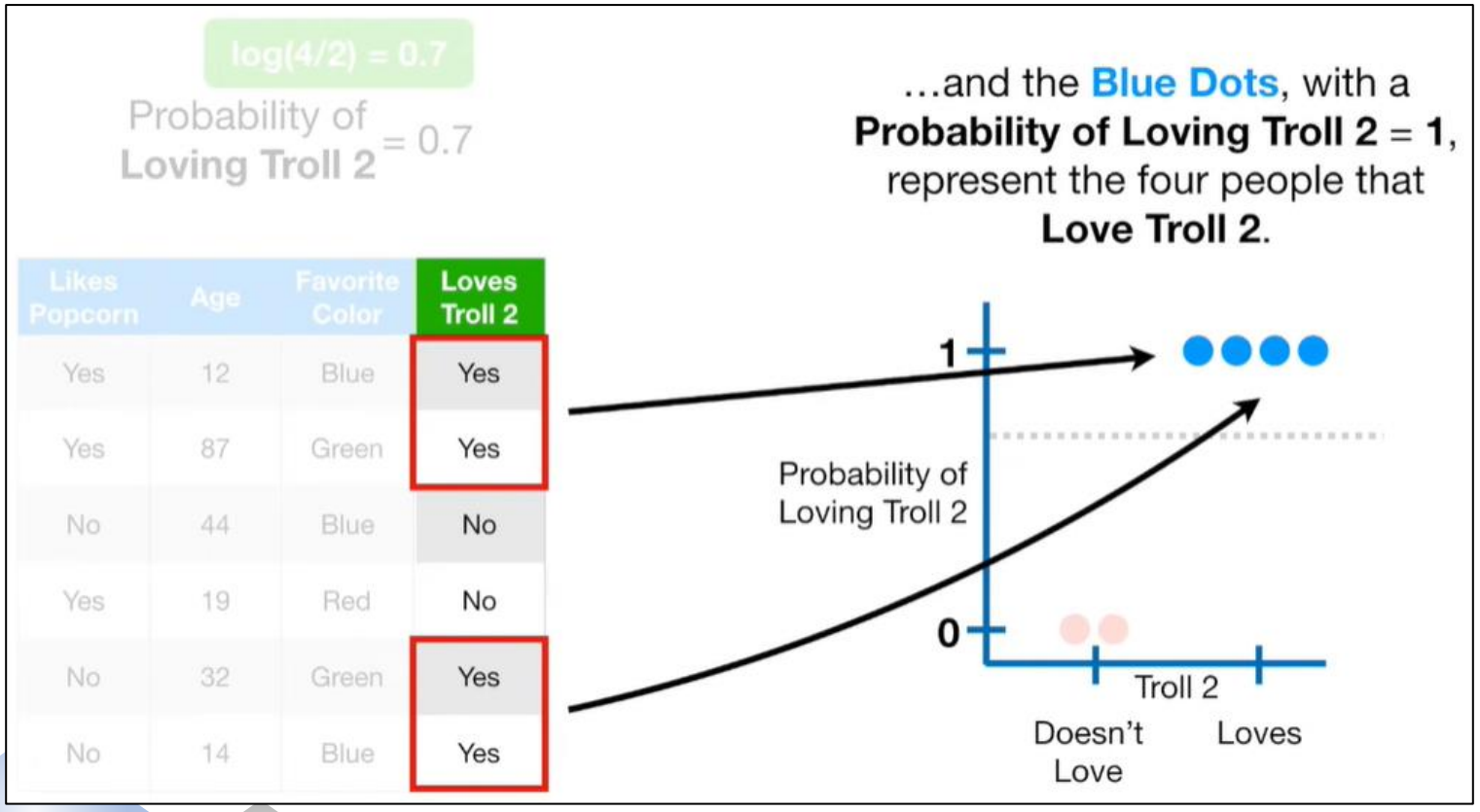
$$\log(\frac{4}{2}) = 0.7$$

https://www.youtube.com/watch?v=jxuNLH5dXCs&list=PLblh5JKOoLUICTaGLRoHQDuF_7q2GfuJF&index=56

3.4.2 Gradient Boost

1. 초기 예측값(확률) 설정
   - 샘플별 초기확률 설정
   - log(odds) 값을 확률로 변환 (= 0.7, 초기값 0.7과 계산식이 다름)
   - 회귀에서의 평균과 같은 개념. 0.5로 설정해도 됨.

$$\text{Probability of Loving Troll 2} = \frac{e^{\log(\text{odds})}}{1 + e^{\log(\text{odds})}}$$

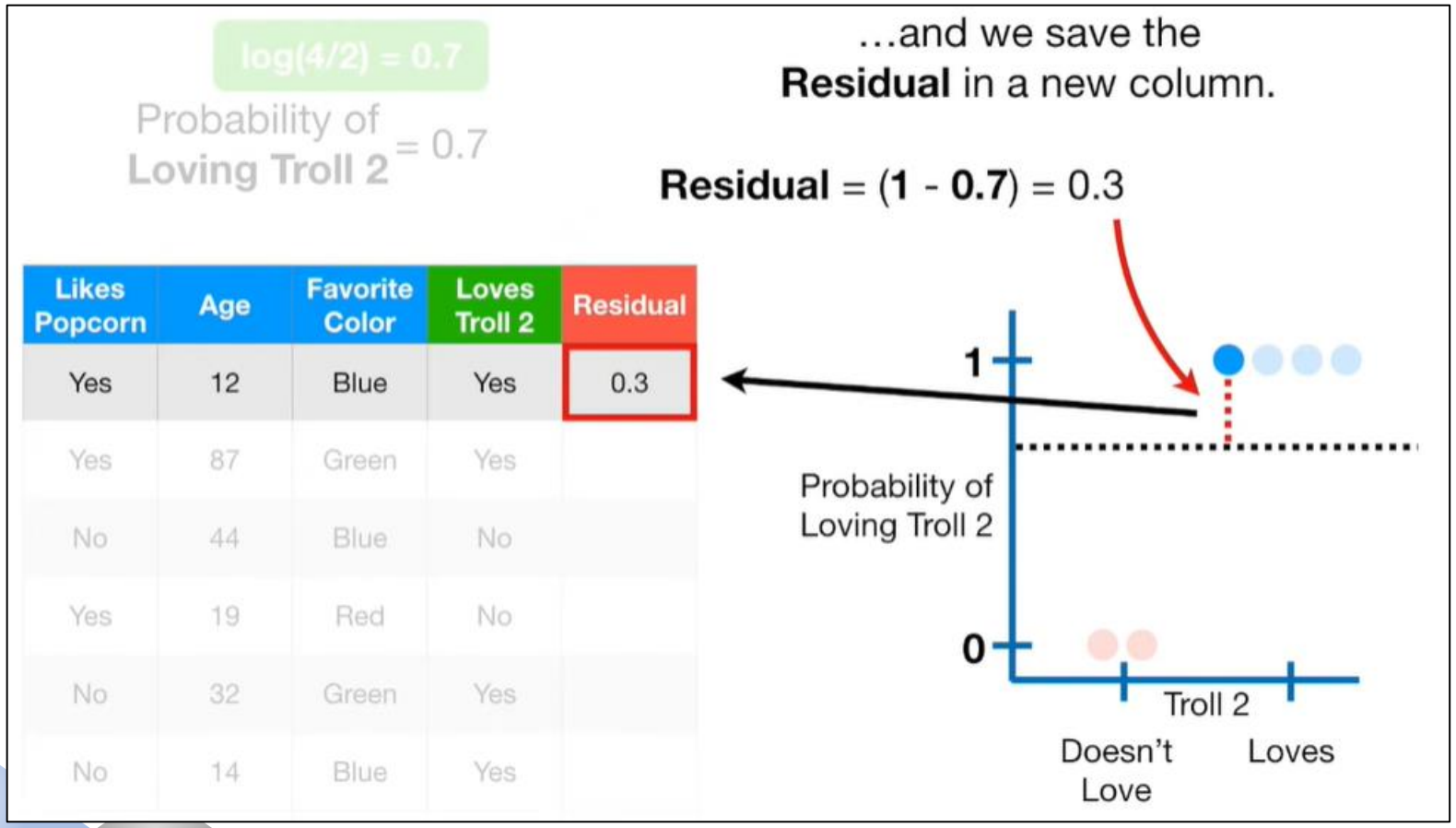


3.4.2 Gradient Boost

2. 잔차 계산 1

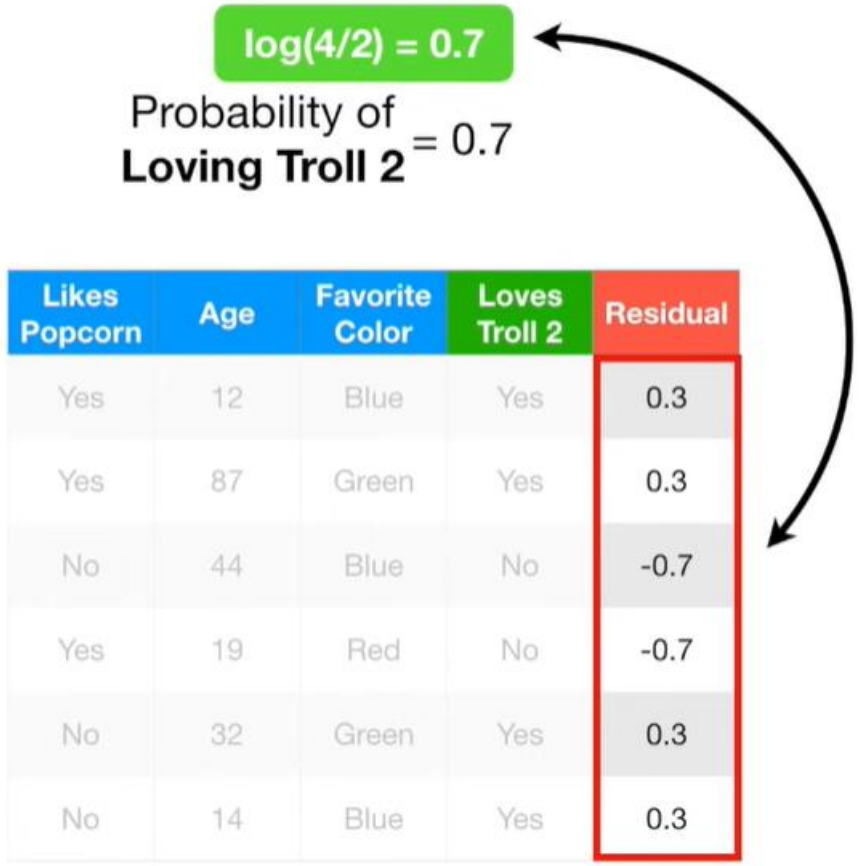
- 처음 예측값 : 0.7로 일괄 설정
- 샘플데이타별로 잔차 계산
- Yes인 샘플의 잔차 : 1 – 0.7 = 0.3
- No인 샘플의 잔차 : 0 – 0.7 = -0.7

Residual = (Observed - Predicted)



3.4.2 Gradient Boost

2. 잔차 계산 2



Likes Popcorn	Age	Favorite Color	Loves Troll 2	Residual
Yes	12	Blue	Yes	0.3
Yes	87	Green	Yes	0.3
No	44	Blue	No	-0.7
Yes	19	Red	No	-0.7
No	32	Green	Yes	0.3
No	14	Blue	Yes	0.3

- 영화 좋아한다 = 1.
- 0 ~ 1 확률을 Sigmoid 함수로 표현함.
- 학습을 진행하면서 잔차를 줄이는 것이 목적

- 처음 잔차가 0.3인 샘플
  - 목표값이 1인 샘플.
  - 확률변환값이 1에 가까워지는 것이 목표.

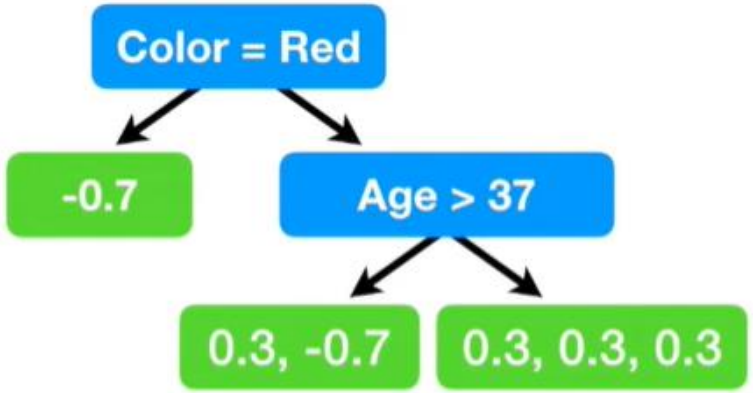
- 처음 잔차가 -0.7인 샘플
  - 목표값이 0인 샘플.
  - 확률변환값이 0에 가까워지는 것이 목표.

3.4.2 Gradient Boost

3. 처음 트리 생성

- Gini Index 적용하여 루트 노드 결정
- 3 개의 리프 사용
- (보통 리프 개수는 8 ~ 32 개 사용)

Likes Popcorn	Age	Favorite Color	Loves Troll 2	Residual
Yes	12	Blue	Yes	0.3
Yes	87	Green	Yes	0.3
No	44	Blue	No	-0.7
Yes	19	Red	No	-0.7
No	32	Green	Yes	0.3
No	14	Blue	Yes	0.3



In this simple example, we are limiting the number of leaves to **3**.

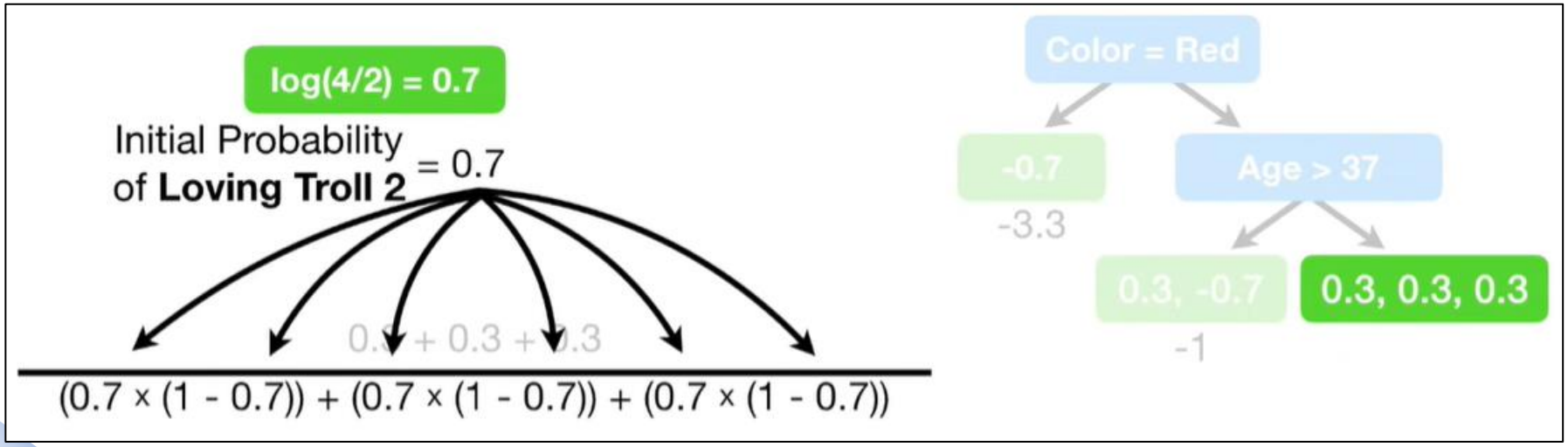
3.4.2 Gradient Boost

4. 확률잔차 → log(odds) 값으로 변환
   - 잔차값은 확률 연산결과인 반면, 모델개선을 위한 연산은 log(odds)값이 대상임.
   - 잔차값 → log(odds) 형식으로 변환 필요.
   - 잔차값이 줄어드는 방향으로 log(odds) 값 갱신

리프값 변환공식 :

$$\frac{\sum \text{Residual}_i}{\sum \left[\text{Previous Probability}_i \times (1 - \text{Previous Probability}_i)\right]}$$

리프값 변환 :

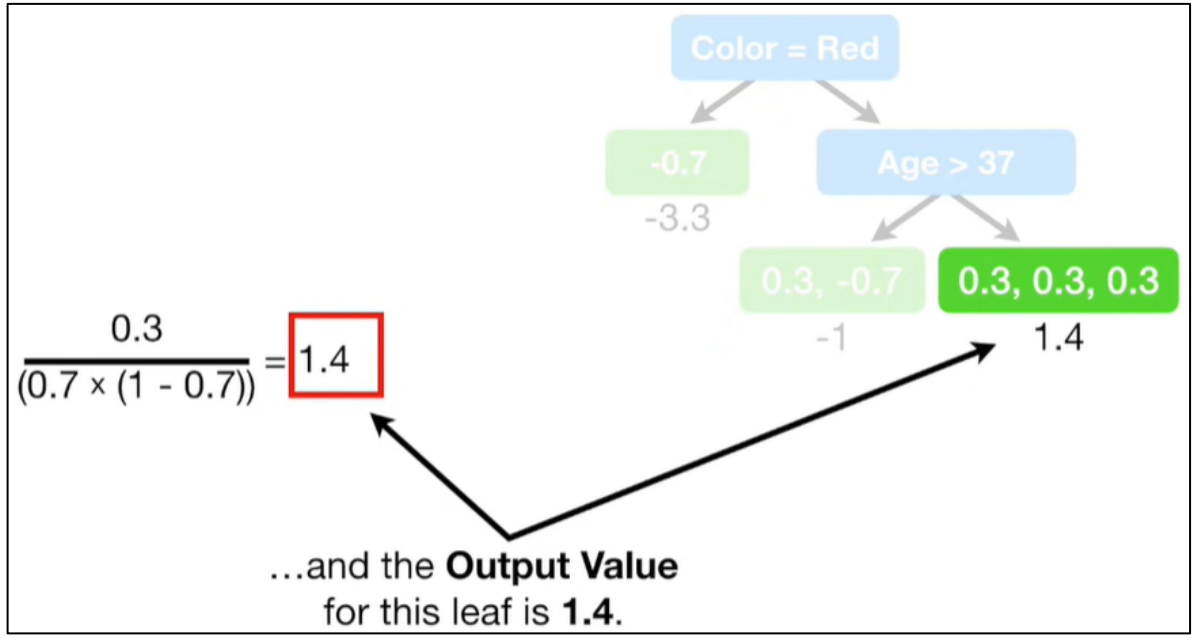


3.4.2 Gradient Boost

4. 잔차갱신 방법

- -0.7인 리프
  - 목표값이 0인 샘플.
  - 큰 음수일 수록 log(odds) 값 작아지고 → probability → 0 수렴
- 0.3 리프
  - 목표값이 1인 샘플.
  - 큰 양수일수록 log(odds) 값 커지고 → 확률 → 1 수렴
- 0.3/0.7 혼합 리프
  - 하나는 0, 하나는 1인 샘플 혼합.
  - 트리 더 만들면서 계산필요.

3 개의 리프값 결과 :

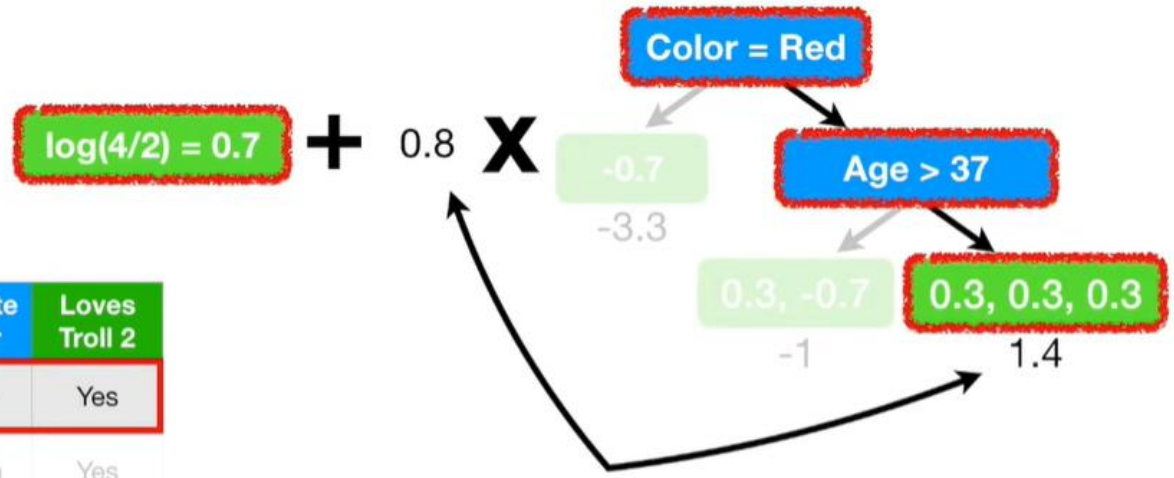


3.4.2 Gradient Boost

5-1. 두번째 log(odds) 계산

- 샘플별로 기존 log(odds) 값(0.7) + 변환된 값(1.4)으로 갱신
- 변화하는 크기는 학습률(여기서는 0.8)를 반영하여 조정
- 참값이 Yes인 샘플은 갱신값이 커지고 No인 샘플은 갱신값이 작아져야 함.

Likes Popcorn	Age	Favorite Color	Loves Troll 2
Yes	12	Blue	Yes
Yes	87	Green	Yes
No	44	Blue	No
Yes	19	Red	No
No	32	Green	Yes
No	14	Blue	Yes



3.4.2 Gradient Boost

5-2. 두번째 log(odds) 계산 → 확률로 변환

- 갱신된 log(odds) 값(=1.8)을 확률로 변환
- 기존 0.7 → 0.9 로 정답에 가까워짐.
- 전체 샘플을 대상으로 예측확률값 갱신

확률로 변환

$$\text{Probability} = \frac{e^{1.8}}{1 + e^{1.8}} = 0.9$$

log(odds) Prediction = 0.7 + (0.8 × 1.4) = 1.8

1번 샘플 결과

Likes Popcorn	Age	Favorite Color	Loves Troll 2	Predicted Prob.
Yes	12	Blue	Yes	0.9
Yes	87	Green	Yes	
No	44	Blue	No	
Yes	19	Red	No	
No	32	Green	Yes	
No	14	Blue	Yes	

We save the new **Predicted Probability** here.

$$\text{Probability} = \frac{e^{1.8}}{1 + e^{1.8}} = 0.9$$

log(odds) Prediction = 0.7 + (0.8 × 1.4) = 1.8

3.4.2 Gradient Boost

6. 새로운 샘플별 예측값을 기준으로 잔차 갱신

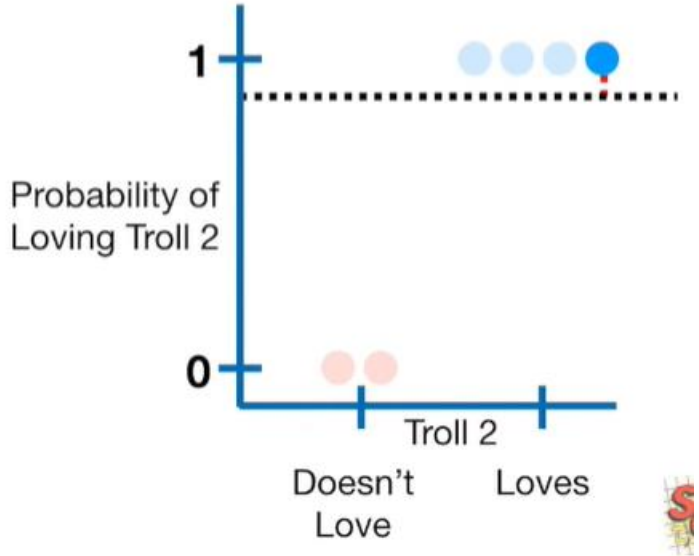
- 1번 샘플 : Yes – 0.9 = 0.1
- 학습은 잔차가 줄어드는 방향으로 진행됨.
- 주) 두번째 샘플은 확률이 더 떨어짐(0.7 -> 0.5)
- 많은 트리가 필요한 이유.

Likes Popcorn	Age	Favorite Color	Loves Troll 2	Predicted Prob.
Yes	12	Blue	Yes	0.9
Yes	87	Green	Yes	0.5
No	44	Blue	No	
Yes	19	Red	No	

NOTE: This new predicted probability is worse than before, and this is one reason why we build a lot of trees, and not just one.

$$\text{Probability} = \frac{e^{-0.1}}{1 + e^{-0.1}} = 0.5$$

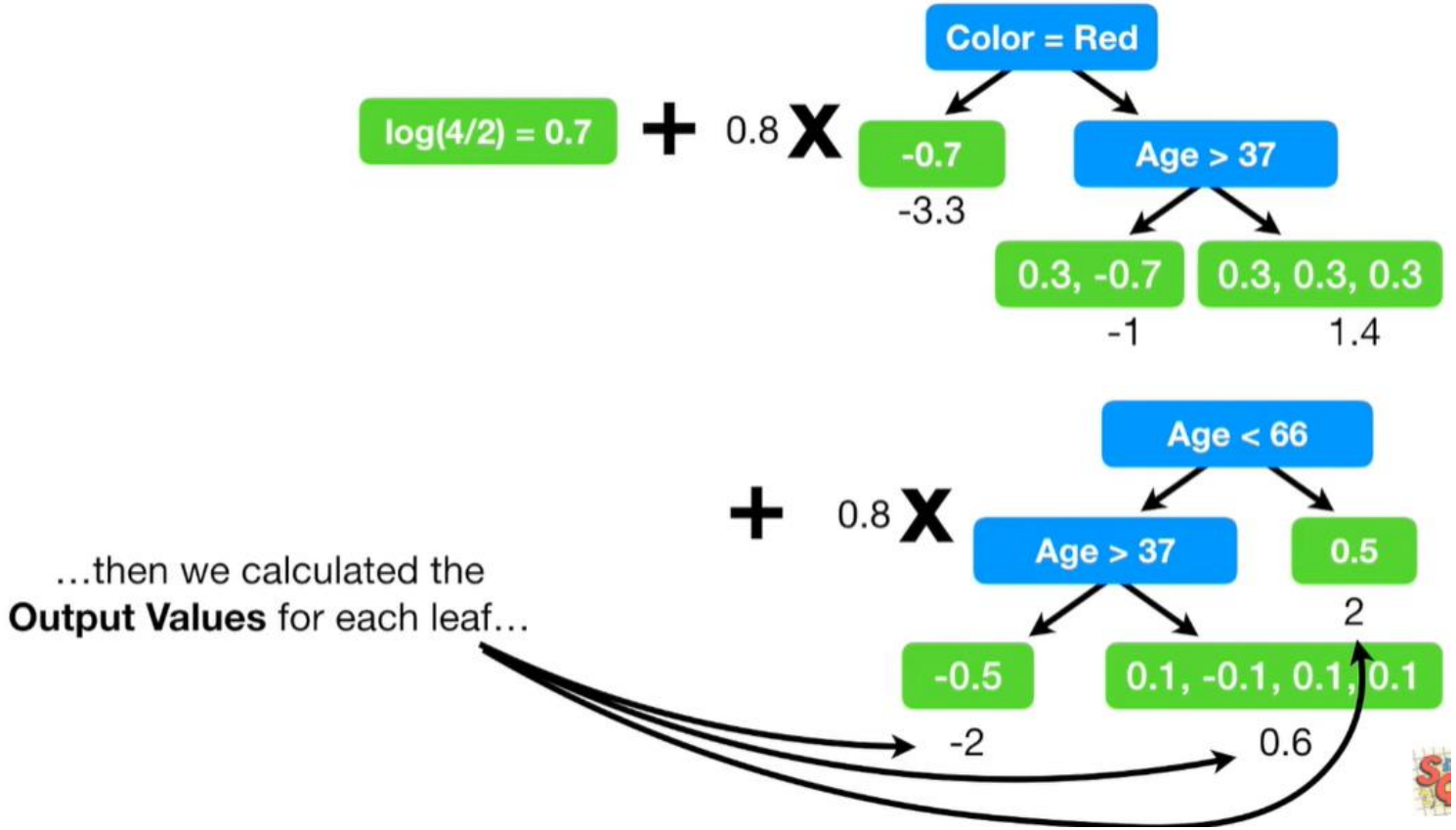
Likes Popcorn	Age	Favorite Color	Loves Troll 2	Predicted Prob.	Residual
Yes	12	Blue	Yes	0.9	0.1
Yes	87	Green	Yes	0.5	0.5
No	44	Blue	No	0.5	-0.5
Yes	19	Red	No	0.1	-0.1
No	32	Green	Yes	0.9	0.1
No	14	Blue	Yes	0.9	0.1



3.4.2 Gradient Boost

7. 같은 과정으로 두번째 트리 생성 및 잔차 갱신

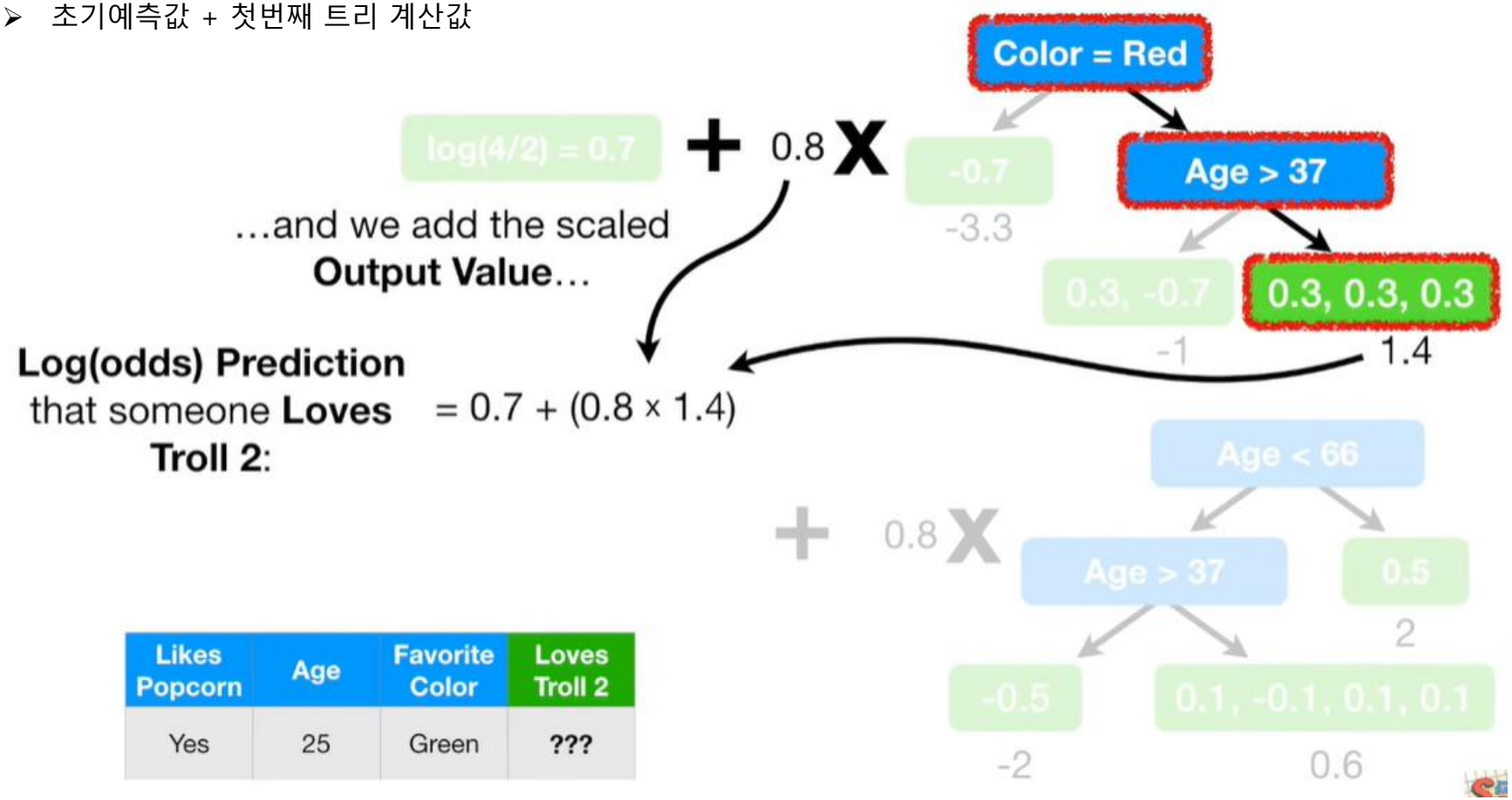
- 모델생성 종료 시점
- ➔ 지정된 트리가 모두 생성되거나 잔차더 줄지 않을 때까지.



3.4.2 Gradient Boost

8. 예측(1)

- 초기예측값 + 첫번째 트리 계산값

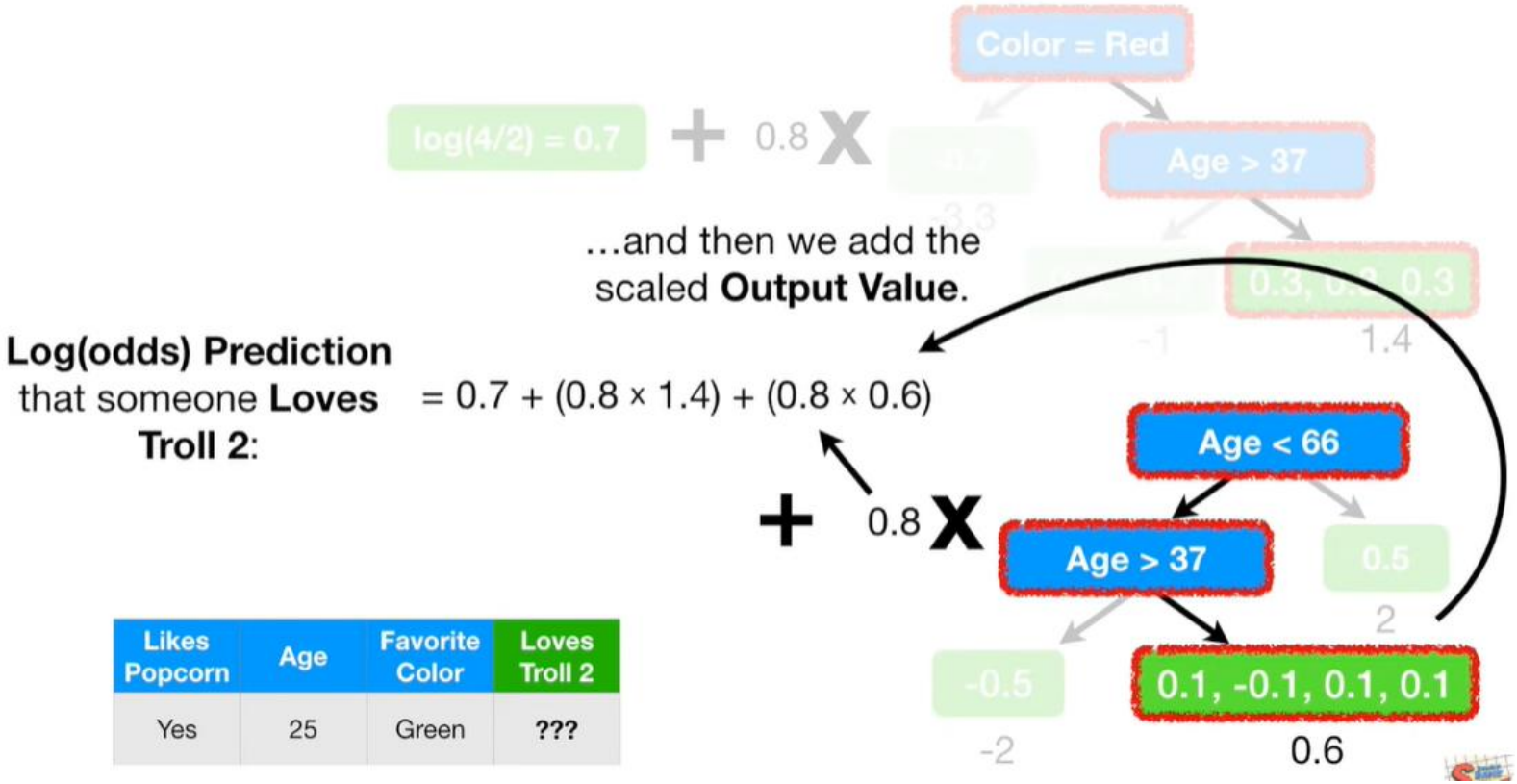


Likes Popcorn	Age	Favorite Color	Loves Troll 2
Yes	25	Green	???

3.4.2 Gradient Boost

8. 예측(2)

- 초기예측값 + 첫번째 트리 계산값 + 두번째 트리 계산값 + ...

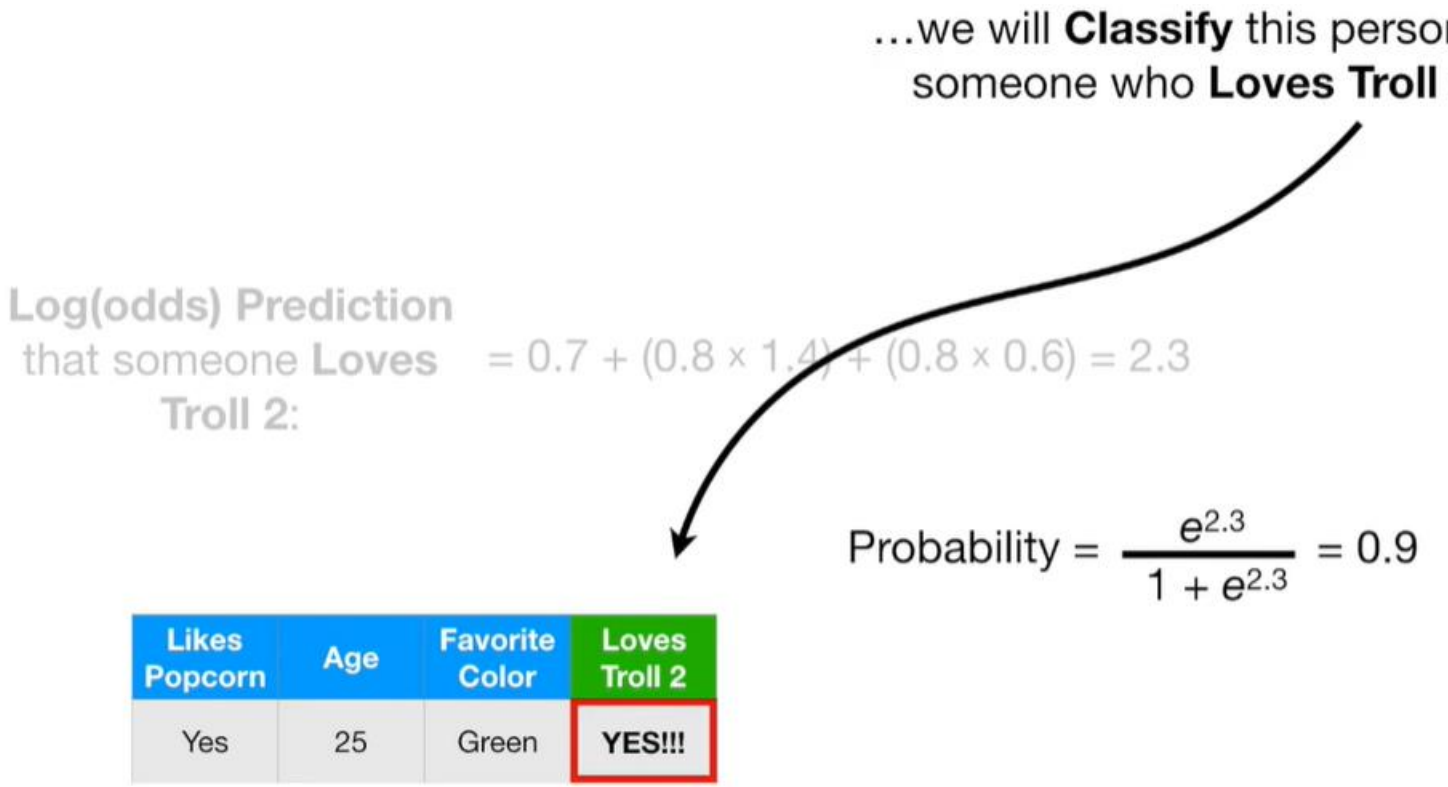


Likes Popcorn	Age	Favorite Color	Loves Troll 2
Yes	25	Green	???

3.4.2 Gradient Boost

8. 예측(3)

- logs(odds) 예측값을 확률로 변환 → 0.9



3.4.3 중요매개변수.장단점

- 암데이타셋에 기본설정값 적용(cell 23)
  - 깊이 : 3, 트리수 : 100개, 학습률 : 0.1
  - 1.000, 0.958. → 과대적합
- 깊이 조절로 튜닝(cell 24)
  - max_depth=1
  - 0.991, 0.972
- 또는 학습률 낮춤(cell 25)
  - learning_rate=0.01
  - 0.991, 0.972

- 중요 매개변수
  - 학습률(learning rate), 크면 보정을 강하게 하므로 복잡한 모델을 만듬.
  - 트리 개수(n_estimators), max_depth(5 이하)
  - 특성 중요도 : 랜덤 포레스트와 비슷함. 일부 특성 무시.
- 장단점
  - 특성스케일 조정 불필요. 이진/연속 특성 데이터 구분없이 잘 동작.
  - 랜덤포레스트가 잘 작동하더라도 예측시간이 중요하거나 좀 더 성능향상 필요할 때 적용.
  - 매개변수 조정이 어렵고 훈련시간이 길다.