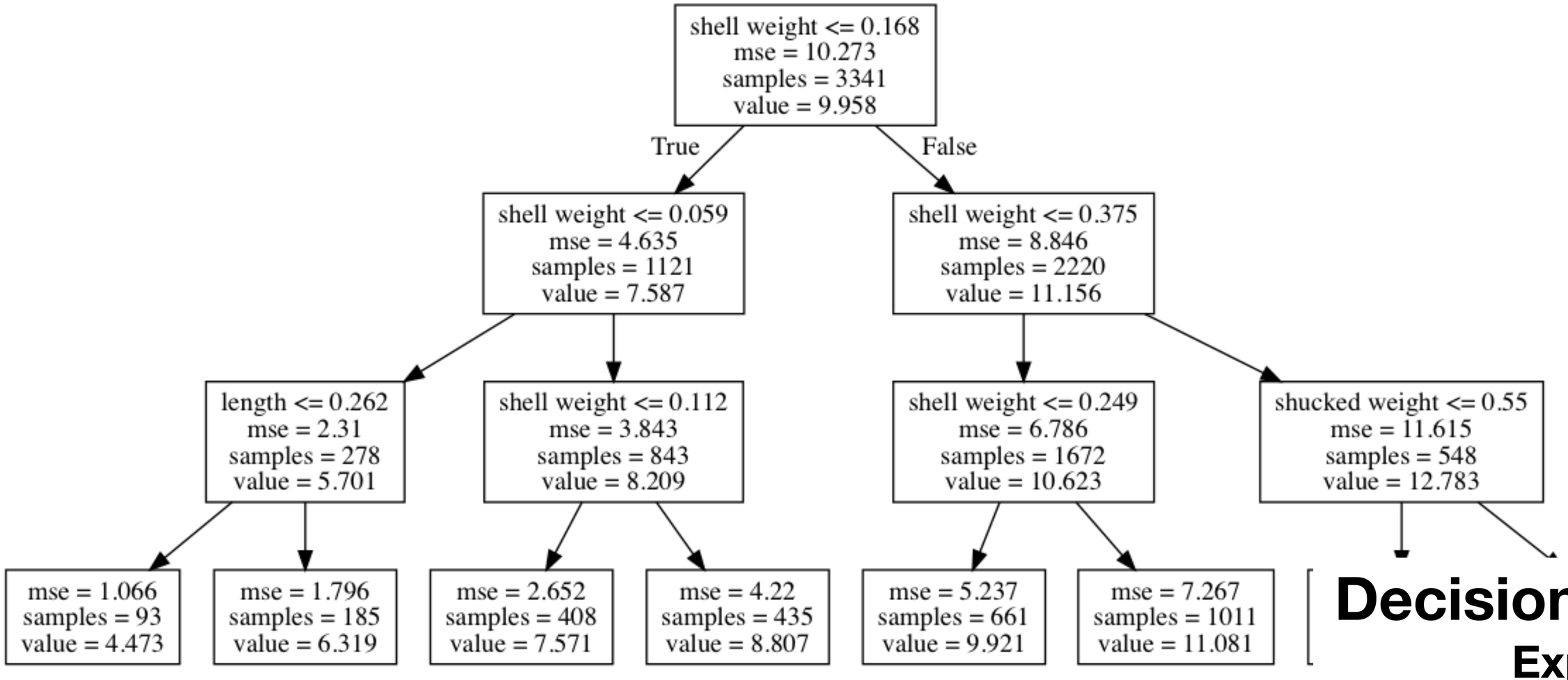


Decision Tree
Explained

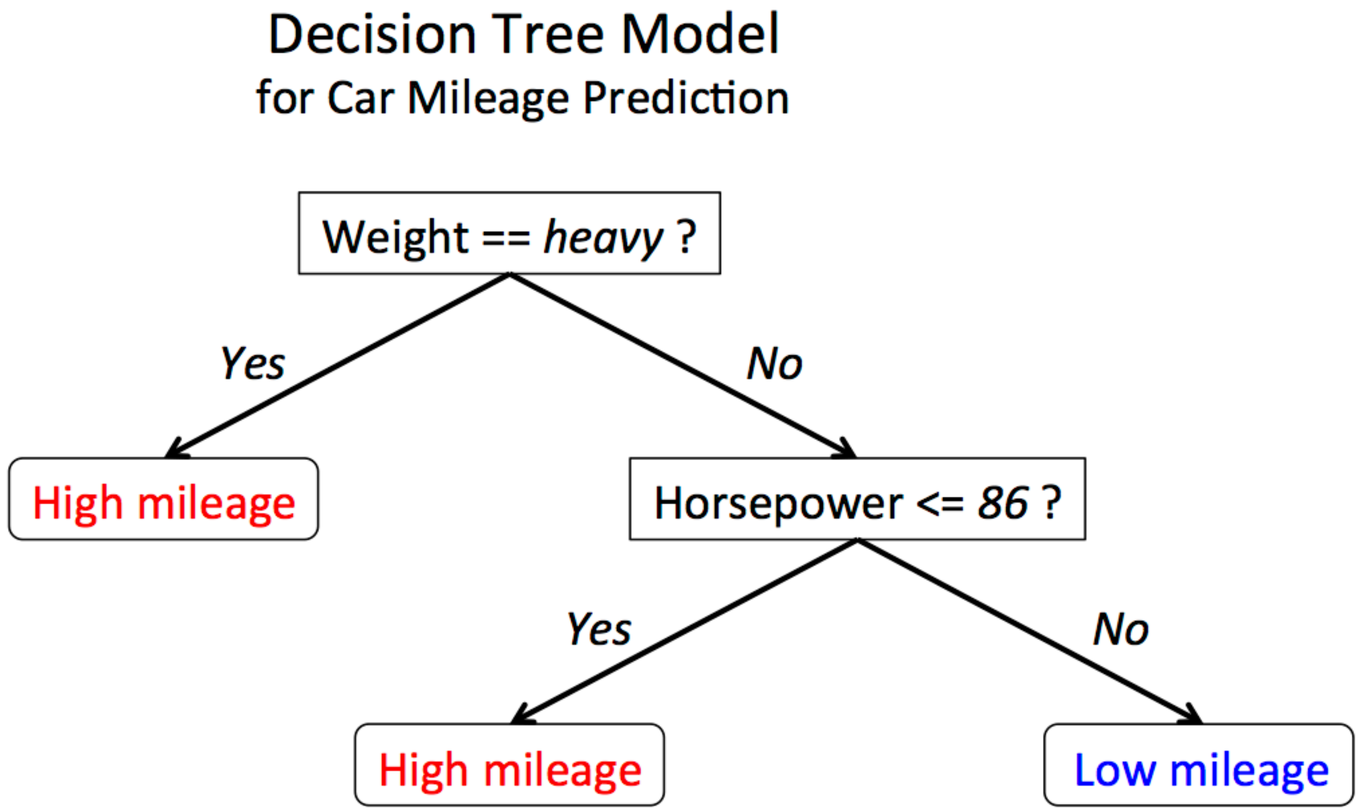
DS School

https://dsschool.co.kr

Overview

Decision Tree

가장 기본적인 머신러닝 알고리즘 중 하나
사람이 가장 이해하기 쉬운 머신러닝 알고리즘 중 하나이며, 그 성능도 좋은 편에 속한다



동작 원리 1

일단 Feature와 Label이 필요하다
이 데이터로 fit을 하면 모델을 학습할 수 있고, predict를 하면 모델을 예측할 수 있다

Dataset

PassengerId	Pclass	Sex_encode	Fare_fillin	Embarked_C	Embarked_S	Embarked_Q	Survived
1	3	0.0	7.2500	False	True	False	0
2	1	1.0	71.2833	True	False	False	1
3	3	1.0	7.9250	False	True	False	1
4	1	1.0	53.1000	False	True	False	1
5	3	0.0	8.0500	False	True	False	0

동작 원리 1

일단 Feature와 Label이 필요하다
이 데이터로 fit을 하면 모델을 학습할 수 있고, predict를 하면 모델을 예측할 수 있다

Dataset

Features

Label

PassengerId	Pclass	Sex_encode	Fare_fillin	Embarked_C	Embarked_S	Embarked_Q	Survived
1	3	0.0	7.2500	False	True	False	0
2	1	1.0	71.2833	True	False	False	1
3	3	1.0	7.9250	False	True	False	1
4	1	1.0	53.1000	False	True	False	1
5	3	0.0	8.0500	False	True	False	0

동작 원리2

fit을 하면, 트리는 먼저 가지고 있는 feature들을 바탕으로
이론적으로 만들 수 있는 모든 조건(condition)을 만든다

Features

PassengerId	Pclass	Sex_encode	Fare_fillin	Embarked_C	Embarked_S	Embarked_Q	Survived
1	3	0.0	7.2500	False	True	False	0
2	1	1.0	71.2833	True	False	False	1
3	3	1.0	7.9250	False	True	False	1
4	1	1.0	53.1000	False	True	False	1
5	3	0.0	8.0500	False	True	False	0

Condition

- **Pclass <= 1.5**
- **Pclass <= 2.5**
- **Sex_encode == “female”**
- **Embarked_C == True**
- **Embarked_S == True**
- **Embarked_Q == True**
- **Fare_fillin <= 7.25**
- **Fare_fillin <= 7.925**
- **Fare_fillin <= 8.05**
- **…**
- **…**
- **…**

동작 원리2

그 조건을 활용해서 가지를 치면 자연스럽게 나무가 만들어진다
여기서 어떤 조건이 가지의 위로 올라가고, 어떤 조건이 가지의 아래로 내려가는지가 중요하다

PassengerId	Pclass	Sex_encode	Fare_fillin	Embarked_C	Embarked_S	Embarked_Q	Survived
1	3	0.0	7.2500	False	True	False	0
2	1	1.0	71.2833	True	False	False	1
3	3	1.0	7.9250	False	True	False	1
4	1	1.0	53.1000	False	True	False	1
5	3	0.0	8.0500	False	True	False	0

Condition

- Pclass <= 1.5
- Pclass <= 2.5
- Sex_encode == “female”
- Embarked_C == True
- Embarked_S == True
- Embarked_Q == True
- Fare_fillin <= 7.25
- Fare_fillin <= 7.925
- Fare_fillin <= 8.05
- …
- …
- …

동작 원리2

그 조건을 활용해서 가지를 치면 자연스럽게 나무가 만들어진다
여기서 어떤 조건이 가지의 위로 올라가고, 어떤 조건이 가지의 아래로 내려가는지가 중요하다

PassengerId	Pclass	Sex_encode	Fare_fillin	Embarked_C	Embarked_S	Embarked_Q	Survived
1	3	0.0	7.2500	False	True	False	0
2	1	1.0	71.2833	True	False	False	1
3	3	1.0	7.9250	False	True	False	1
4	1	1.0	53.1000	False	True	False	1
5	3	0.0	8.0500	False	True	False	0

1. feature를 활용해 모든 조건(condition)을 만들고

Condition

- Pclass <= 1.5
- Pclass <= 2.5
- Sex_encode == “female”
- Embarked_C == True
- Embarked_S == True
- Embarked_Q == True
- Fare_fillin <= 7.25
- Fare_fillin <= 7.925
- Fare_fillin <= 8.05
- …
- …
- …

동작 원리2

그 조건을 활용해서 가지를 치면 자연스럽게 나무가 만들어진다
여기서 어떤 조건이 가지의 위로 올라가고, 어떤 조건이 가지의 아래로 내려가는지가 중요하다

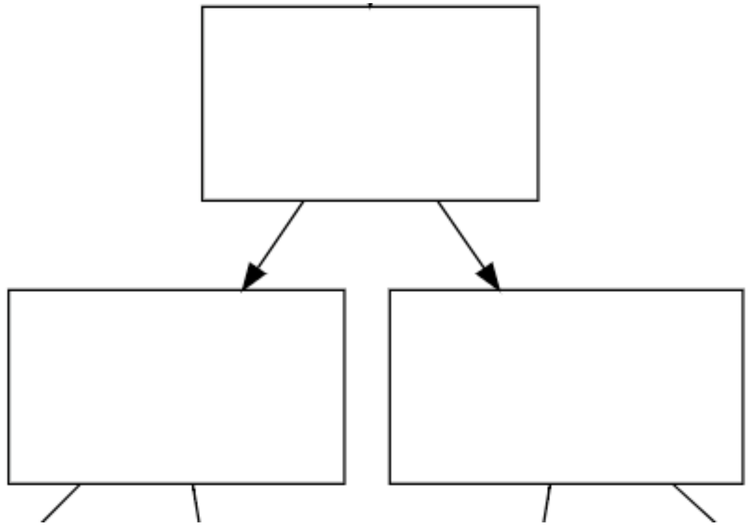
PassengerId	Pclass	Sex_encode	Fare_fillin	Embarked_C	Embarked_S	Embarked_Q	Survived
1	3	0.0	7.2500	False	True	False	0
2	1	1.0	71.2833	True	False	False	1
3	3	1.0	7.9250	False	True	False	1
4	1	1.0	53.1000	False	True	False	1
5	3	0.0	8.0500	False	True	False	0

1. feature를 활용해 모든 조건(condition)을 만들고

Condition

- Pclass <= 1.5
- Pclass <= 2.5
- Sex_encode == “female”
- Embarked_C == True
- Embarked_S == True
- Embarked_Q == True
- Fare_fillin <= 7.25
- Fare_fillin <= 7.925
- Fare_fillin <= 8.05
- …
- …
- …

2. 조건(condition)을 활용해 가지를 쳐서 나무를 만든다



동작 원리2

그 조건을 활용해서 가지를 치면 자연스럽게 나무가 만들어진다
여기서 어떤 조건이 가지의 위로 올라가고, 어떤 조건이 가지의 아래로 내려가는지가 중요하다

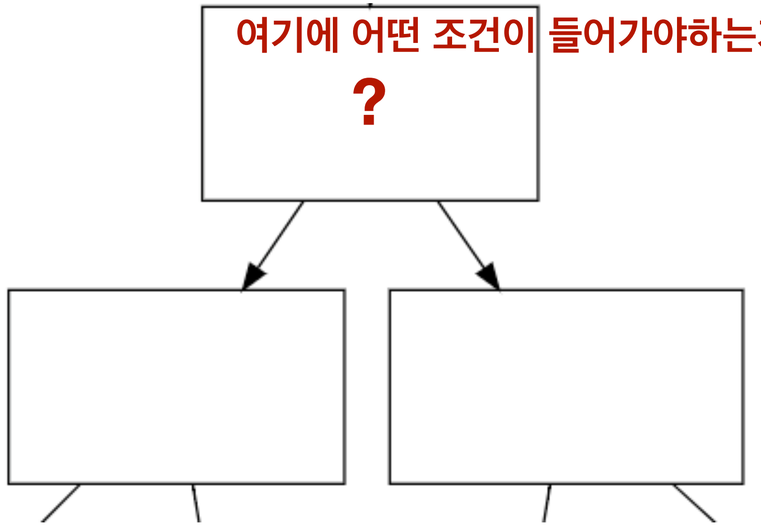
PassengerId	Pclass	Sex_encode	Fare_fillin	Embarked_C	Embarked_S	Embarked_Q	Survived
1	3	0.0	7.2500	False	True	False	0
2	1	1.0	71.2833	True	False	False	1
3	3	1.0	7.9250	False	True	False	1
4	1	1.0	53.1000	False	True	False	1
5	3	0.0	8.0500	False	True	False	0

1. feature를 활용해 모든 조건(condition)을 만들고

Condition

- Pclass <= 1.5
- Pclass <= 2.5
- Sex_encode == “female”
- Embarked_C == True
- Embarked_S == True
- Embarked_Q == True
- Fare_fillin <= 7.25
- Fare_fillin <= 7.925
- Fare_fillin <= 8.05
- …
- …
- …

2. 조건(condition)을 활용해 가지를 쳐서 나무를 만든다



동작 원리2

그 조건을 활용해서 가지를 치면 자연스럽게 나무가 만들어진다

여기서 어떤 조건이 가지의 위로 올라가고, 어떤 조건이 가지의 아래로 내려가는지가 중요하다

PassengerId	Pclass	Sex_encode	Fare_fillin	Embarked_C	Embarked_S	Embarked_Q	Survived
1	3	0.0	7.2500	False	True	False	0
2	1	1.0	71.2833	True	False	False	1
3	3	1.0	7.9250	False	True	False	1
4	1	1.0	53.1000	False	True	False	1
5	3	0.0	8.0500	False	True	False	0

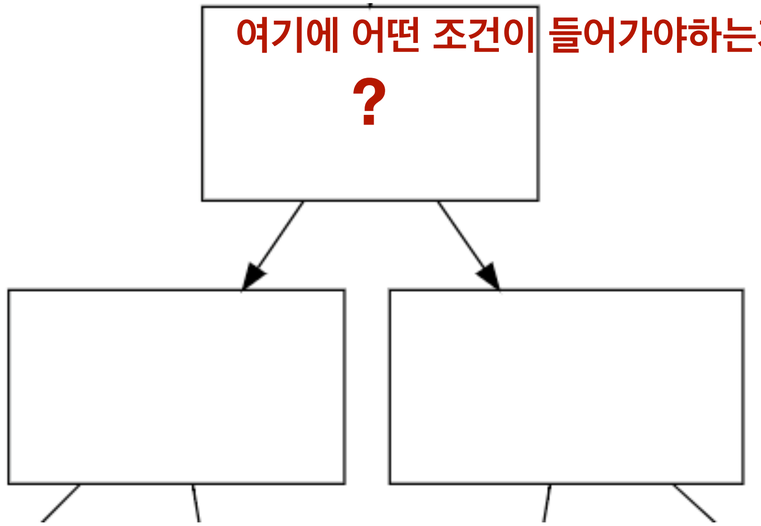
아마도 label에 따라 달라질 것

1. feature를 활용해 모든 조건(condition)을 만들고

Condition

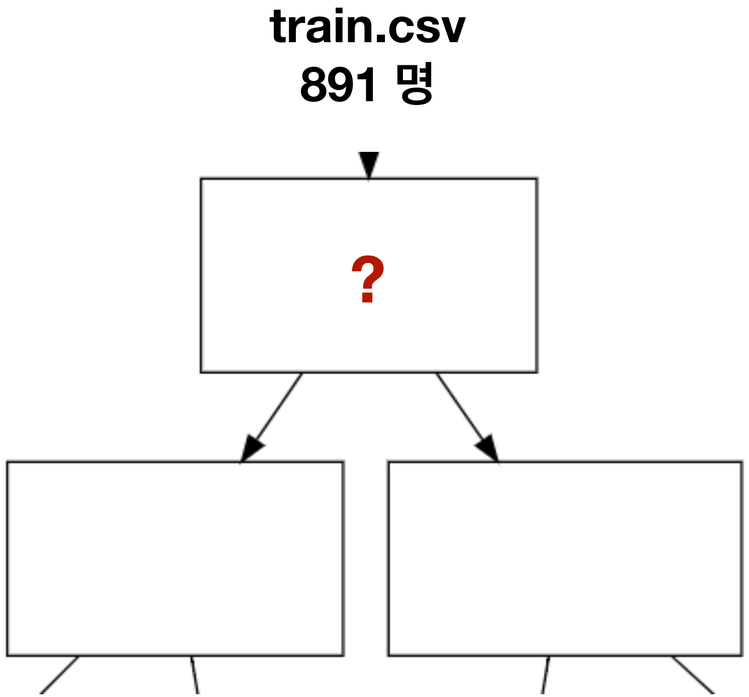
- Pclass <= 1.5
- Pclass <= 2.5
- Sex_encode == “female”
- Embarked_C == True
- Embarked_S == True
- Embarked_Q == True
- Fare_fillin <= 7.25
- Fare_fillin <= 7.925
- Fare_fillin <= 8.05
- …
- …
- …

2. 조건(condition)을 활용해 가지를 쳐서 나무를 만든다



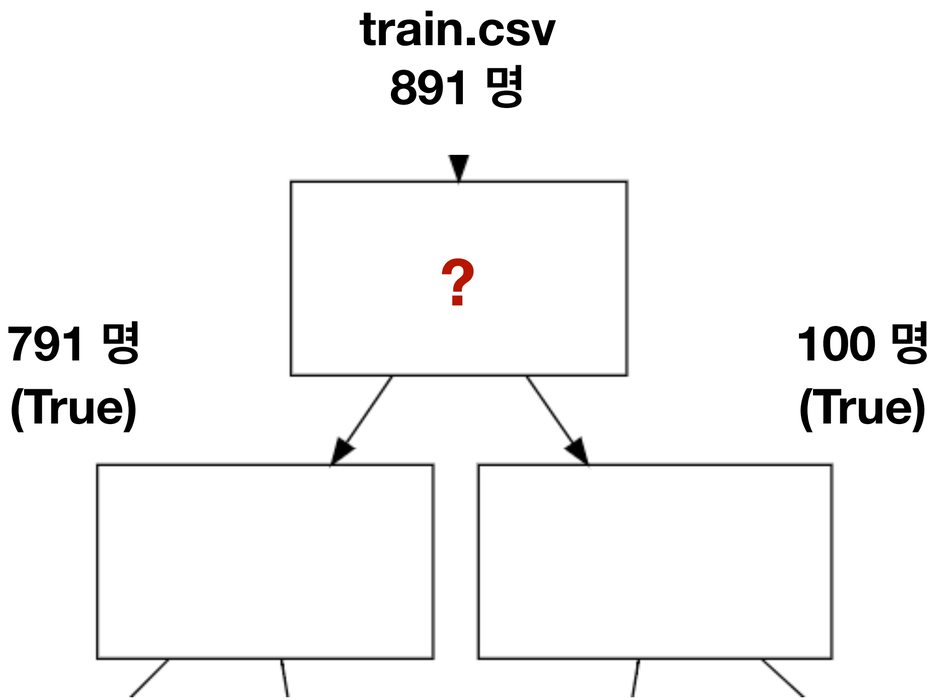
Gini Impurity

지니 불순도(Gini Impurity)는 어떤 조건이 좋은 조건이고, 어떤 조건이 좋지 않은 조건인지를 알 수 있다
지니 불순도가 낮을수록 좋은 조건이므로, 이 조건을 트리의 상위에 놓으면 된다



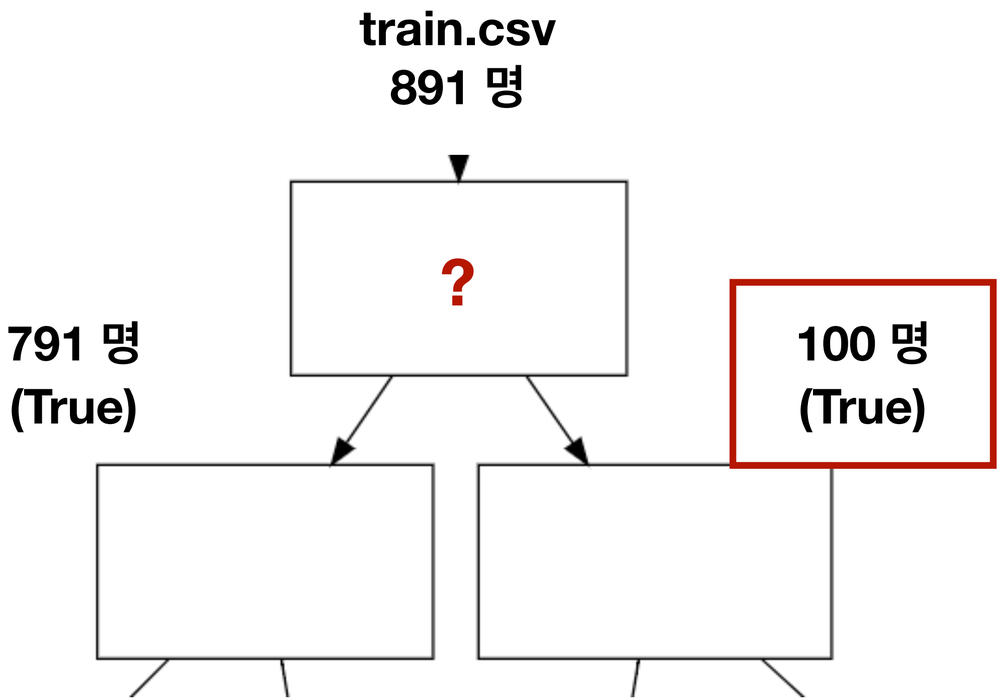
Gini Impurity

지니 불순도(Gini Impurity)는 어떤 조건이 좋은 조건이고, 어떤 조건이 좋지 않은 조건인지를 알 수 있다
지니 불순도가 낮을수록 좋은 조건이므로, 이 조건을 트리의 상위에 놓으면 된다



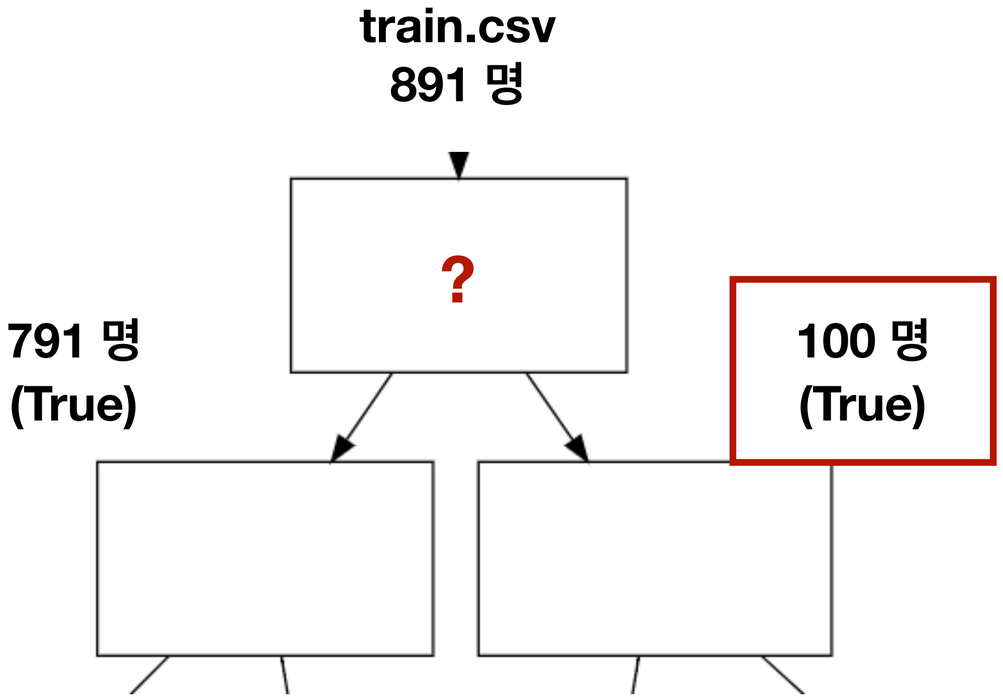
Gini Impurity

지니 불순도(Gini Impurity)는 어떤 조건이 좋은 조건이고, 어떤 조건이 좋지 않은 조건인지를 알 수 있다
지니 불순도가 낮을수록 좋은 조건이므로, 이 조건을 트리의 상위에 놓으면 된다



Gini Impurity

지니 불순도(Gini Impurity)는 어떤 조건이 좋은 조건이고, 어떤 조건이 좋지 않은 조건인지를 알 수 있다
지니 불순도가 낮을수록 좋은 조건이므로, 이 조건을 트리의 상위에 놓으면 된다

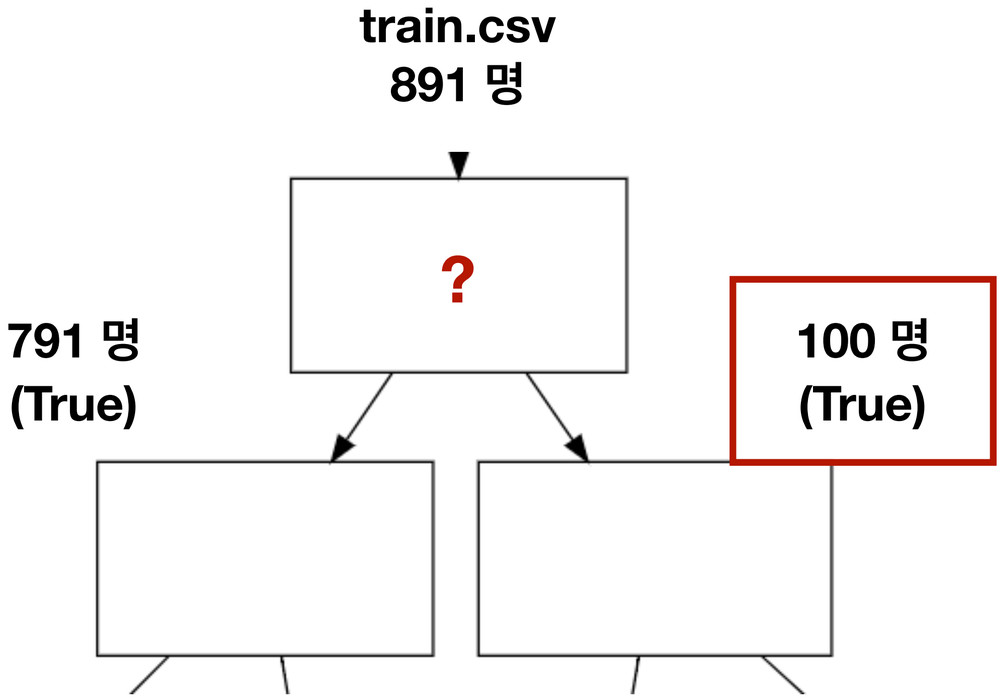


Gini Impurity 공식

$$I_G(p) = 1 - (p_{true})^2 - (p_{false})^2$$

Gini Impurity

지니 불순도(Gini Impurity)는 어떤 조건이 좋은 조건이고, 어떤 조건이 좋지 않은 조건인지를 알 수 있다
지니 불순도가 낮을수록 좋은 조건이므로, 이 조건을 트리의 상위에 놓으면 된다



Gini Impurity 공식

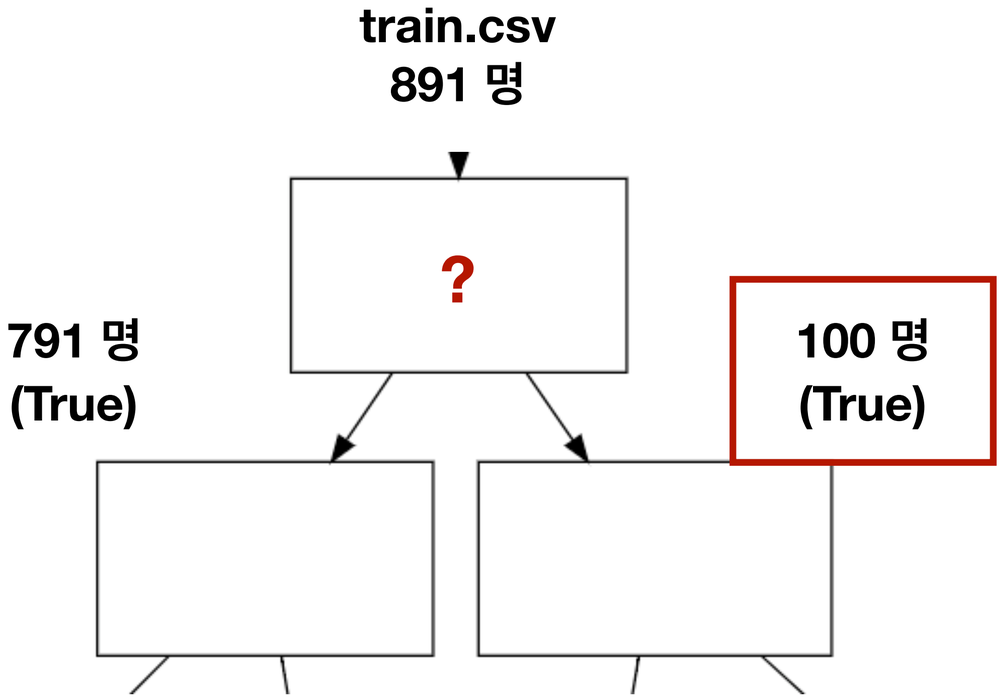
$$I_G(p) = 1 - (p_{true})^2 - (p_{false})^2$$

예시1: 100명 중에 100명 전부가 생존했을 경우 (좋은 조건)

$$I_G(p) = 1 - (p_{true})^2 - (p_{false})^2 = 1 - \left(\frac{100}{100}\right)^2 - \left(\frac{0}{100}\right)^2 = 0$$

Gini Impurity

지니 불순도(Gini Impurity)는 어떤 조건이 좋은 조건이고, 어떤 조건이 좋지 않은 조건인지를 알 수 있다
지니 불순도가 낮을수록 좋은 조건이므로, 이 조건을 트리의 상위에 놓으면 된다



Gini Impurity 공식

$$I_G(p) = 1 - (p_{true})^2 - (p_{false})^2$$

예시1: 100명 중에 100명 전부가 생존했을 경우 (좋은 조건)

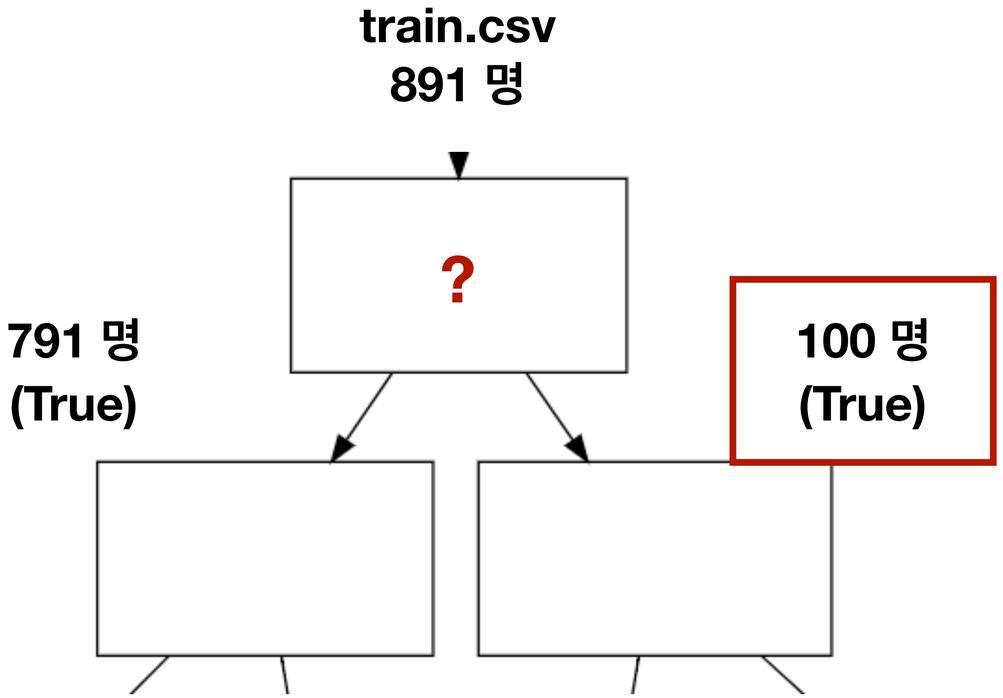
$$I_G(p) = 1 - (p_{true})^2 - (p_{false})^2 = 1 - \left(\frac{100}{100}\right)^2 - \left(\frac{0}{100}\right)^2 = 0$$

예시1: 100명 중에 50명 생존, 50명 사망했을 경우 (안 좋은 조건)

$$I_G(p) = 1 - (p_{true})^2 - (p_{false})^2 = 1 - \left(\frac{50}{100}\right)^2 - \left(\frac{50}{100}\right)^2 = 0.5$$

Gini Impurity

지니 불순도(Gini Impurity)는 어떤 조건이 좋은 조건이고, 어떤 조건이 좋지 않은 조건인지를 알 수 있다
지니 불순도가 낮을수록 좋은 조건이므로, 이 조건을 트리의 상위에 놓으면 된다



Gini Impurity 공식

$$I_G(p) = 1 - (p_{true})^2 - (p_{false})^2$$

예시1: 100명 중에 100명 전부가 생존했을 경우 (좋은 조건)

$$I_G(p) = 1 - (p_{true})^2 - (p_{false})^2 = 1 - \left(\frac{100}{100}\right)^2 - \left(\frac{0}{100}\right)^2 = 0$$

예시1: 100명 중에 50명 생존, 50명 사망했을 경우 (안 좋은 조건)

$$I_G(p) = 1 - (p_{true})^2 - (p_{false})^2 = 1 - \left(\frac{50}{100}\right)^2 - \left(\frac{50}{100}\right)^2 = 0.5$$

결론: 이 공식은 '낮을 수록 좋은 조건'이다

동작 원리3

이 공식을 활용하면 조건(condition)마다의 gini impurity를 계산할 수 있다.
이 갓이 낮을수록 좋은 조건이다

1. 조건(condition)이 있으면

Condition

- Pclass <= 1.5
- Pclass <= 2.5
- Sex_encode == “female”
- Embarked_C == True
- Embarked_S == True
- Embarked_Q == True
- Fare_fillin <= 7.25
- Fare_fillin <= 7.925
- Fare_fillin <= 8.05
- …
- …
- …

동작 원리3

이 공식을 활용하면 조건(condition)마다의 gini impurity를 계산할 수 있다.
이 갓이 낮을수록 좋은 조건이다

1. 조건(condition)이 있으면

Condition

- Pclass <= 1.5
- Pclass <= 2.5
- Sex_encode == “female”
- Embarked_C == True
- Embarked_S == True
- Embarked_Q == True
- Fare_fillin <= 7.25
- Fare_fillin <= 7.925
- Fare_fillin <= 8.05
- …
- …
- …

2. 조건(condition)마다의 gini impurity를 구하면

$$\mathrm{I}_G(p) = 1 - (p_{true})^2 - (p_{false})^2$$

동작 원리3

이 공식을 활용하면 조건(condition)마다의 gini impurity를 계산할 수 있다.
이 갓이 낮을수록 좋은 조건이다

1. 조건(condition)이 있으면

Condition

- Pclass <= 1.5
- Pclass <= 2.5
- Sex_encode == “female”
- Embarked_C == True
- Embarked_S == True
- Embarked_Q == True
- Fare_fillin <= 7.25
- Fare_fillin <= 7.925
- Fare_fillin <= 8.05
- …
- …
- …

2. 조건(condition)마다의 gini impurity를 구하면

$$\mathrm{I}_G(p) = 1 - (p_{true})^2 - (p_{false})^2$$

3. 이 값이 낮을수록 좋은 조건이다

Condition

- Pclass <= 1.5
- Pclass <= 2.5
- Sex_encode == “female”
- Embarked_C == True
- Embarked_S == True
- Embarked_Q == True
- Fare_fillin <= 7.25
- Fare_fillin <= 7.925
- Fare_fillin <= 8.05
- …
- …
- …

동작 원리3

이 공식을 활용하면 조건(condition)마다의 gini impurity를 계산할 수 있다.
이 갓이 낮을수록 좋은 조건이다

1. 조건(condition)이 있으면

Condition

- Pclass <= 1.5
- Pclass <= 2.5
- Sex_encode == “female”
- Embarked_C == True
- Embarked_S == True
- Embarked_Q == True
- Fare_fillin <= 7.25
- Fare_fillin <= 7.925
- Fare_fillin <= 8.05
- …
- …
- …

2. 조건(condition)마다의 gini impurity를 구하면

avg gini impurity = 0.333

$$I_G(p) = 1 - (p_{true})^2 - (p_{false})^2$$

3. 이 값이 낮을수록 좋은 조건이다

Condition

- Pclass <= 1.5
- ~~Pclass <= 2.5~~
- Sex_encode == “female”
- ~~Embarked_C == True~~
- Embarked_S == True
- Embarked_Q == True
- Fare_fillin <= 7.25
- Fare_fillin <= 7.925
- Fare_fillin <= 8.05
- …
- …
- …

동작 원리

1) feature와 label이 주어지면, 2) 트리는 주어진 feature로 가능한 모든 조건을 만들고
3) 이 조건마다의 평균 gini impurity를 구한다. 4) 평균 gini impurity가 낮은 순으로 가지를 친다

PassengerId	Pclass	Sex_encode	Fare_fillin	Embarked_C	Embarked_S	Embarked_Q	Survived
1	3	0.0	7.2500	False	True	False	0
2	1	1.0	71.2833	True	False	False	1
3	3	1.0	7.9250	False	True	False	1
4	1	1.0	53.1000	False	True	False	1
5	3	0.0	8.0500	False	True	False	0

동작 원리

1) feature와 label이 주어지면, 2) 트리는 주어진 feature로 가능한 모든 조건을 만들고
3) 이 조건마다의 평균 gini impurity를 구한다. 4) 평균 gini impurity가 낮은 순으로 가지를 친다

PassengerId	Pclass	Sex_encode	Fare_fillin	Embarked_C	Embarked_S	Embarked_Q	Survived
1	3	0.0	7.2500	False	True	False	0
2	1	1.0	71.2833	True	False	False	1
3	3	1.0	7.9250	False	True	False	1
4	1	1.0	53.1000	False	True	False	1
5	3	0.0	8.0500	False	True	False	0

1. feature를 활용해 모든 조건을 만들고

Condition

- **Pclass <= 1.5**
- **Pclass <= 2.5**
- **Sex_encode == “female”**
- **Embarked_C == True**
- **Embarked_S == True**
- **Embarked_Q == True**
- **Fare_fillin <= 7.25**
- **Fare_fillin <= 7.925**
- **Fare_fillin <= 8.05**
- **…**
- **…**
- **…**

동작 원리

1) feature와 label이 주어지면, 2) 트리는 주어진 feature로 가능한 모든 조건을 만들고
3) 이 조건마다의 평균 gini impurity를 구한다. 4) 평균 gini impurity가 낮은 순으로 가지를 친다

PassengerId	Pclass	Sex_encode	Fare_fillin	Embarked_C	Embarked_S	Embarked_Q	Survived
1	3	0.0	7.2500	False	True	False	0
2	1	1.0	71.2833	True	False	False	1
3	3	1.0	7.9250	False	True	False	1
4	1	1.0	53.1000	False	True	False	1
5	3	0.0	8.0500	False	True	False	0

1. feature를 활용해 모든 조건을 만들고

Condition

- **Pclass <= 1.5**
- **Pclass <= 2.5**
- **Sex_encode == “female”**
- **Embarked_C == True**
- **Embarked_S == True**
- **Embarked_Q == True**
- **Fare_fillin <= 7.25**
- **Fare_fillin <= 7.925**
- **Fare_fillin <= 8.05**
- **…**
- **…**
- **…**

2. 조건마다의 gini impurity를 구하면

$$\mathrm{I}_G(p) = 1 - (p_{true})^2 - (p_{false})^2$$

동작 원리

1) feature와 label이 주어지면, 2) 트리는 주어진 feature로 가능한 모든 조건을 만들고
3) 이 조건마다의 평균 gini impurity를 구한다. 4) 평균 gini impurity가 낮은 순으로 가지를 친다

PassengerId	Pclass	Sex_encode	Fare_fillin	Embarked_C	Embarked_S	Embarked_Q	Survived
1	3	0.0	7.2500	False	True	False	0
2	1	1.0	71.2833	True	False	False	1
3	3	1.0	7.9250	False	True	False	1
4	1	1.0	53.1000	False	True	False	1
5	3	0.0	8.0500	False	True	False	0

1. feature를 활용해 모든 조건을 만들고

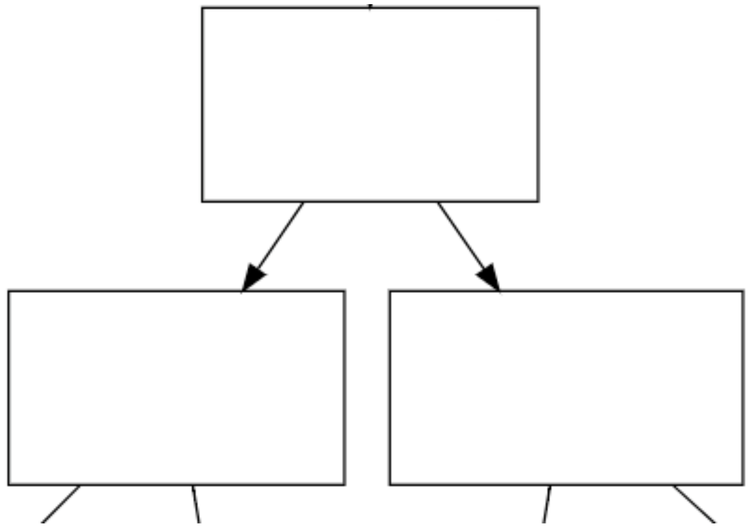
Condition

- **Pclass <= 1.5**
- **Pclass <= 2.5**
- **Sex_encode == “female”**
- **Embarked_C == True**
- **Embarked_S == True**
- **Embarked_Q == True**
- **Fare_fillin <= 7.25**
- **Fare_fillin <= 7.925**
- **Fare_fillin <= 8.05**
- **…**
- **…**
- **…**

2. 조건마다의 gini impurity를 구하면

$$I_G(p) = 1 - (p_{true})^2 - (p_{false})^2$$

3. 평균 gini impurity가 가장 좋은 조건을 최상위에 둔다



동작 원리

1) feature와 label이 주어지면, 2) 트리는 주어진 feature로 가능한 모든 조건을 만들고
3) 이 조건마다의 평균 gini impurity를 구한다. 4) 평균 gini impurity가 낮은 순으로 가지를 친다

PassengerId	Pclass	Sex_encode	Fare_fillin	Embarked_C	Embarked_S	Embarked_Q	Survived
1	3	0.0	7.2500	False	True	False	0
2	1	1.0	71.2833	True	False	False	1
3	3	1.0	7.9250	False	True	False	1
4	1	1.0	53.1000	False	True	False	1
5	3	0.0	8.0500	False	True	False	0

1. feature를 활용해 모든 조건을 만들고

Condition

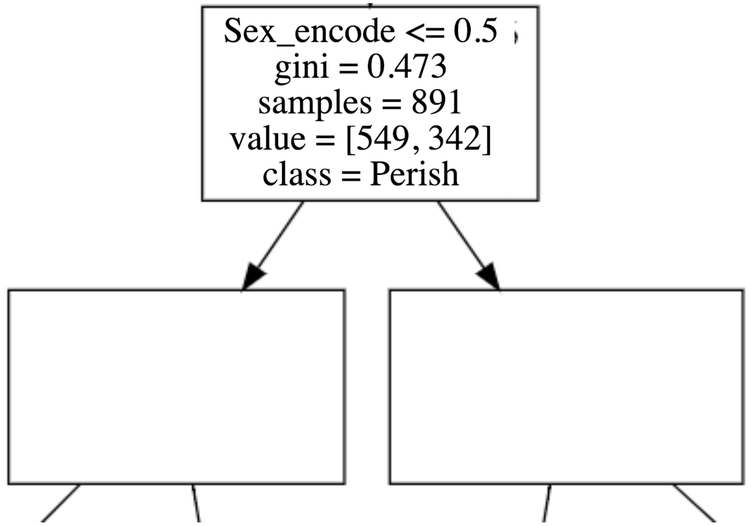
- **Pclass <= 1.5**
- **Pclass <= 2.5**
- **Sex_encode == “female”**
- **Embarked_C == True**
- **Embarked_S == True**
- **Embarked_Q == True**
- **Fare_fillin <= 7.25**
- **Fare_fillin <= 7.925**
- **Fare_fillin <= 8.05**
- **…**
- **…**
- **…**

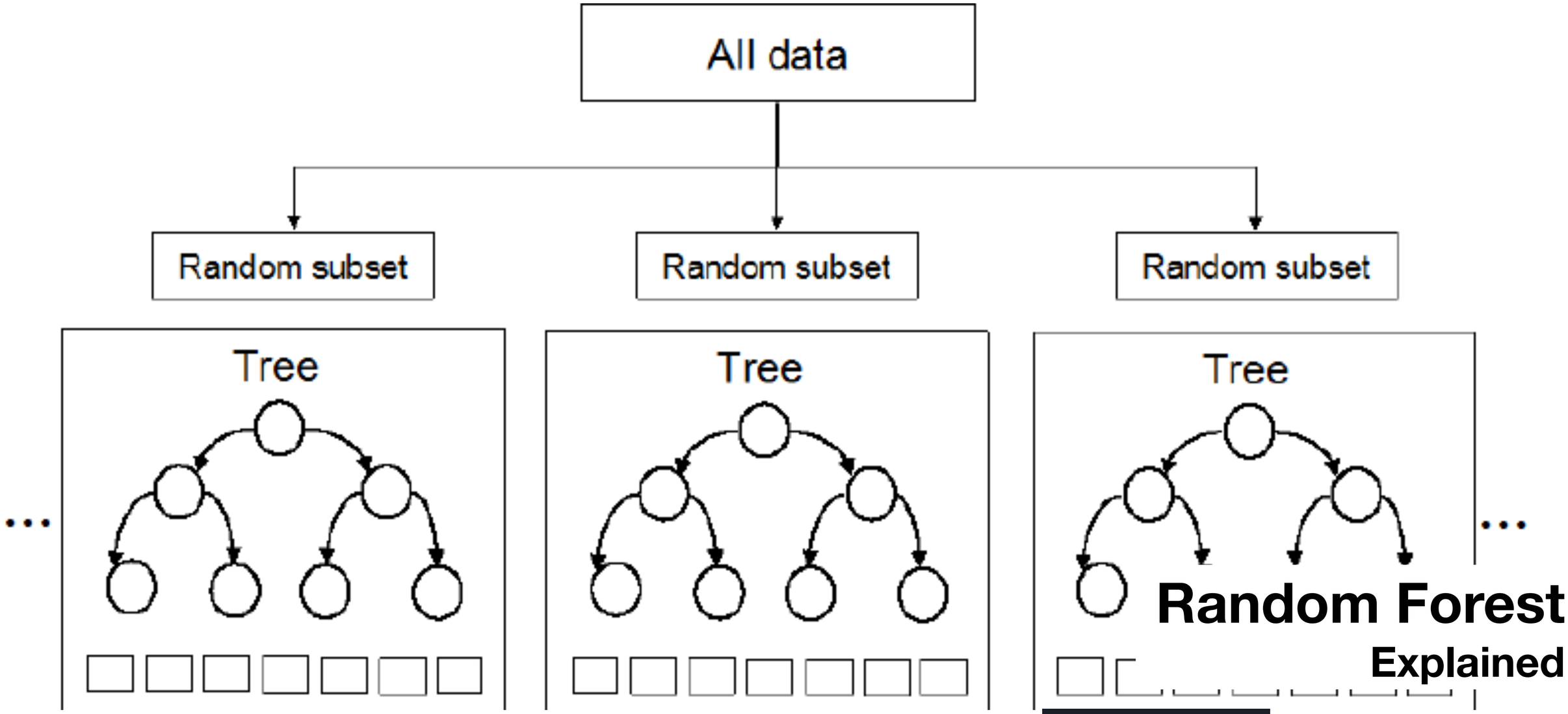
2. 조건마다의 gini impurity를 구하면

$$I_G(p) = 1 - (p_{true})^2 - (p_{false})^2$$

3. 평균 gini impurity가 가장 좋은 조건을 최상위에 둔다

성별 조건이 gini impurity가 가장 낮으므로
이 조건을 최상위에 둔다





Random Forest Explained

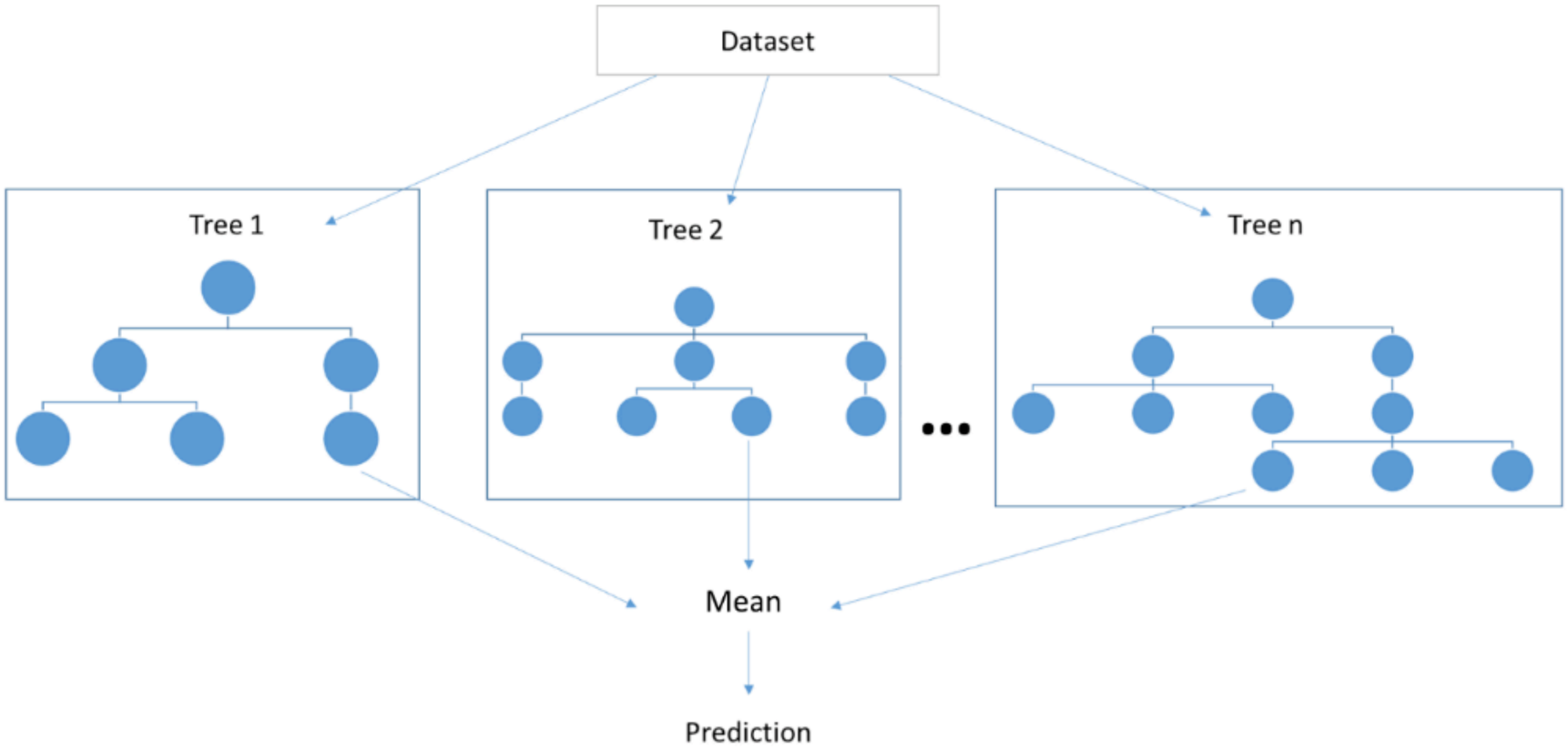
DS School

https://dsschool.co.kr

Random Forest

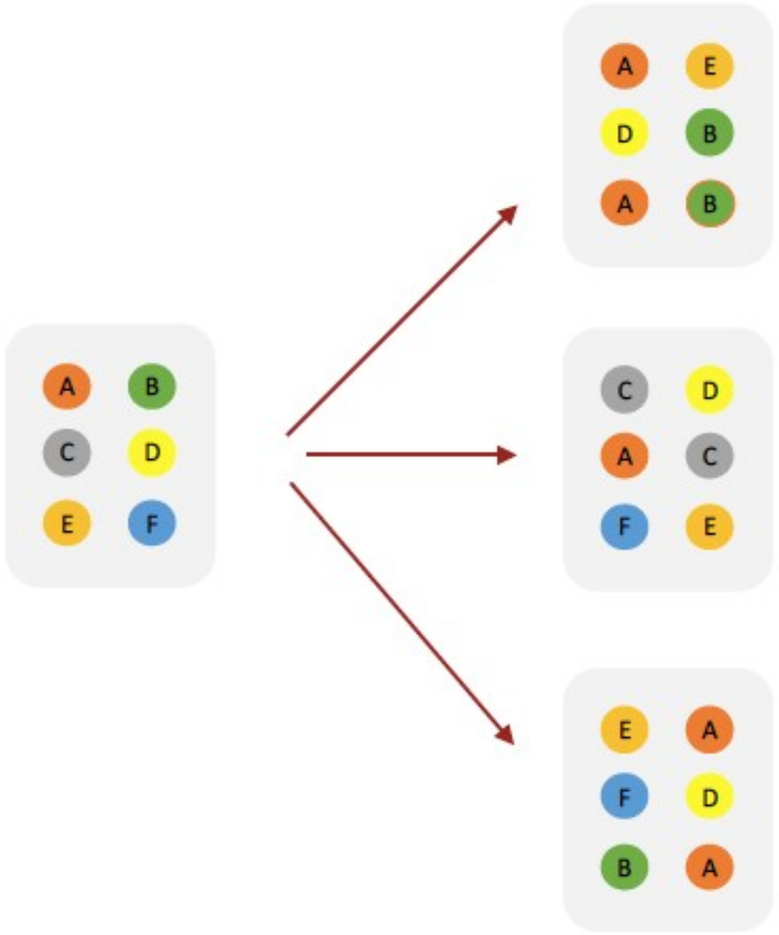
Decision Tree를 응용한 알고리즘

Decision Tree에 배깅(Bagging, Bootstrap aggregating)이라는 방식을 적용하는 원리이다



Bagging(Bootstrap aggregating)의 원리

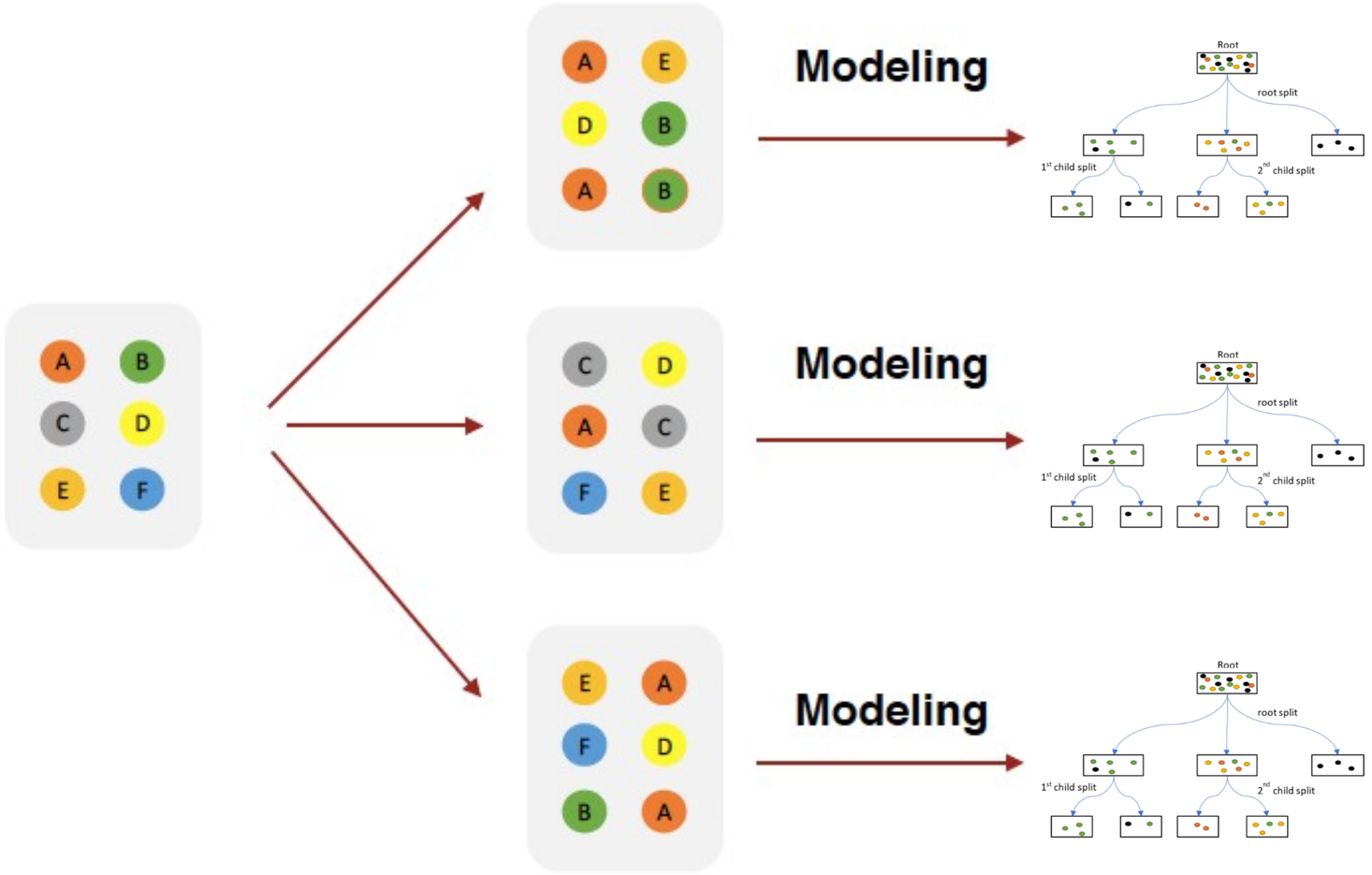
먼저 원본 데이터가 주어졌을 때, 이 데이터를 복원 추출(sampling with replacement)합니다.
복원추출이라 함은 데이터를 랜덤 샘플링하되, 중복을 허용하면서 랜덤하게 샘플링 하는 것입니다.



Random Sampling with replacement

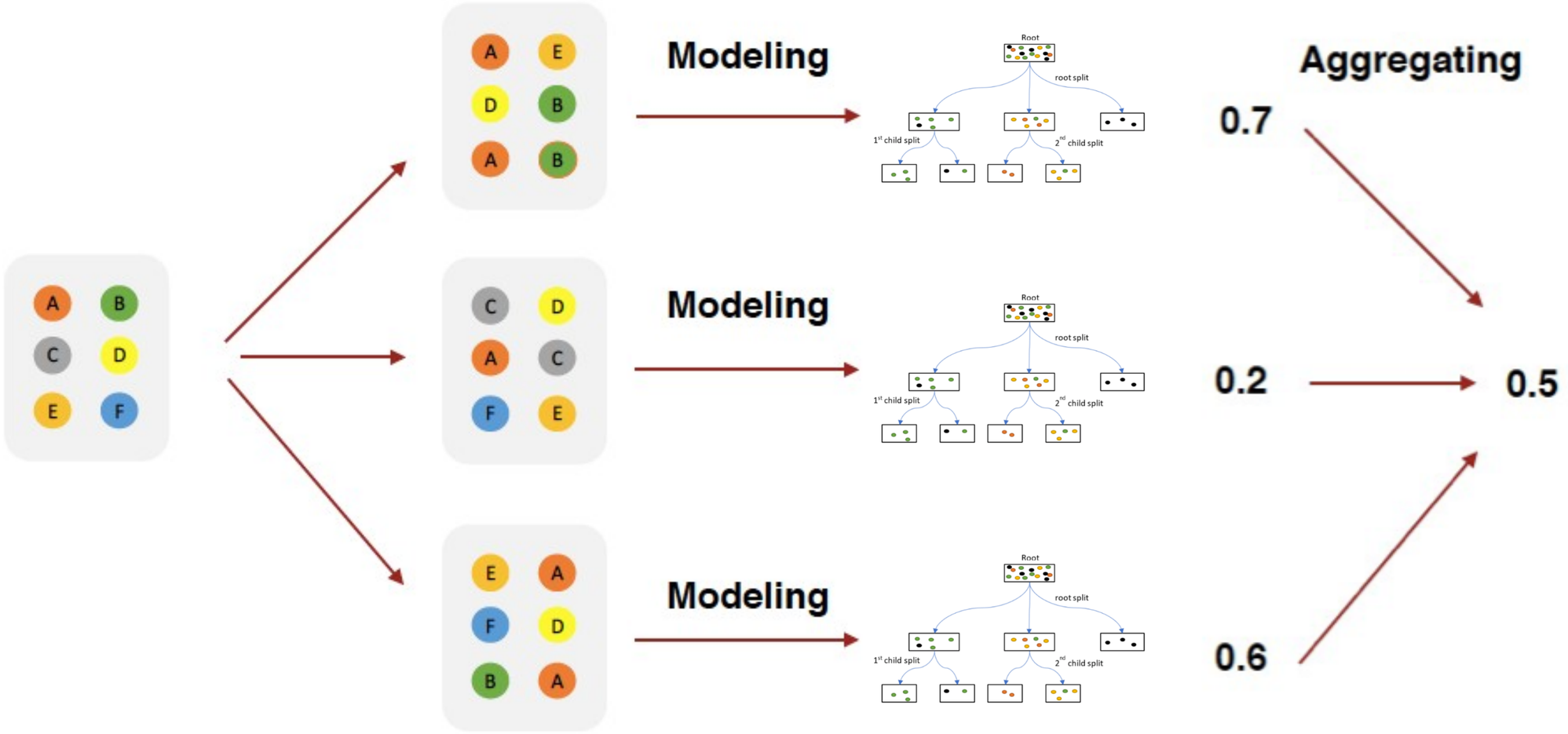
Bagging(Bootstrap aggregating)의 원리

이 샘플링한 데이터로 Decision Tree를 만듭니다
이 방식을 사용하면 Decision Tree를 무제한으로 만들 수 있습니다



Bagging(Bootstrap aggregating)의 원리

이 트리들 여러 개를 섞어서 사용하는 것을
바로 Random Forest라고 합니다

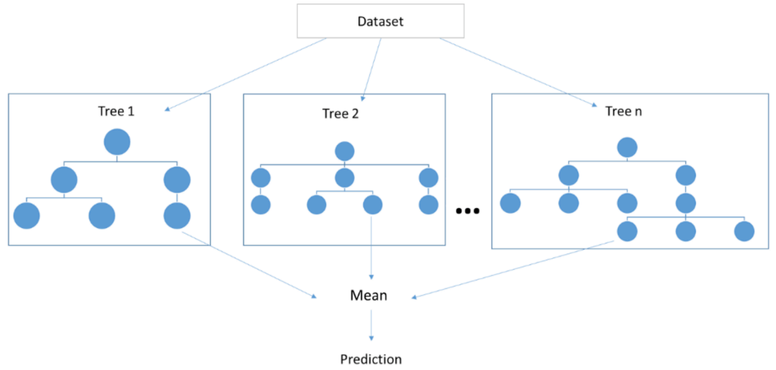


Random Forest의 장점

Random Forest는 Decision Tree를 응용한 방식이지만
그 응용 원리가 매우 간결하며, 거의 모든 상황에서 Decision Tree보다 성능이 좋습니다.

장점

- 원리가 매우 간결하다
- **Decision Tree보다 거의 언제나 좋은 성능을 보장한다.**
- 현장에서 자주 쓰이는 알고리즘이다.

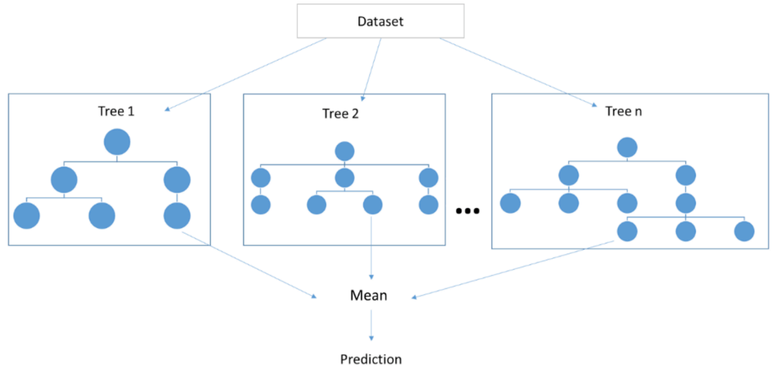


Random Forest의 장점

Random Forest는 Decision Tree를 응용한 방식이지만
그 응용 원리가 매우 간결하며, 거의 모든 상황에서 Decision Tree보다 성능이 좋습니다.

장점

- **원리가 매우 간결하다**
- **Decision Tree보다 거의 언제나 좋은 성능을 보장한다.**
- **현장에서 자주 쓰이는 알고리즘이다.**



결론: 앞으로는 Decision Tree보다 Random Forest를 기본으로 사용할 것