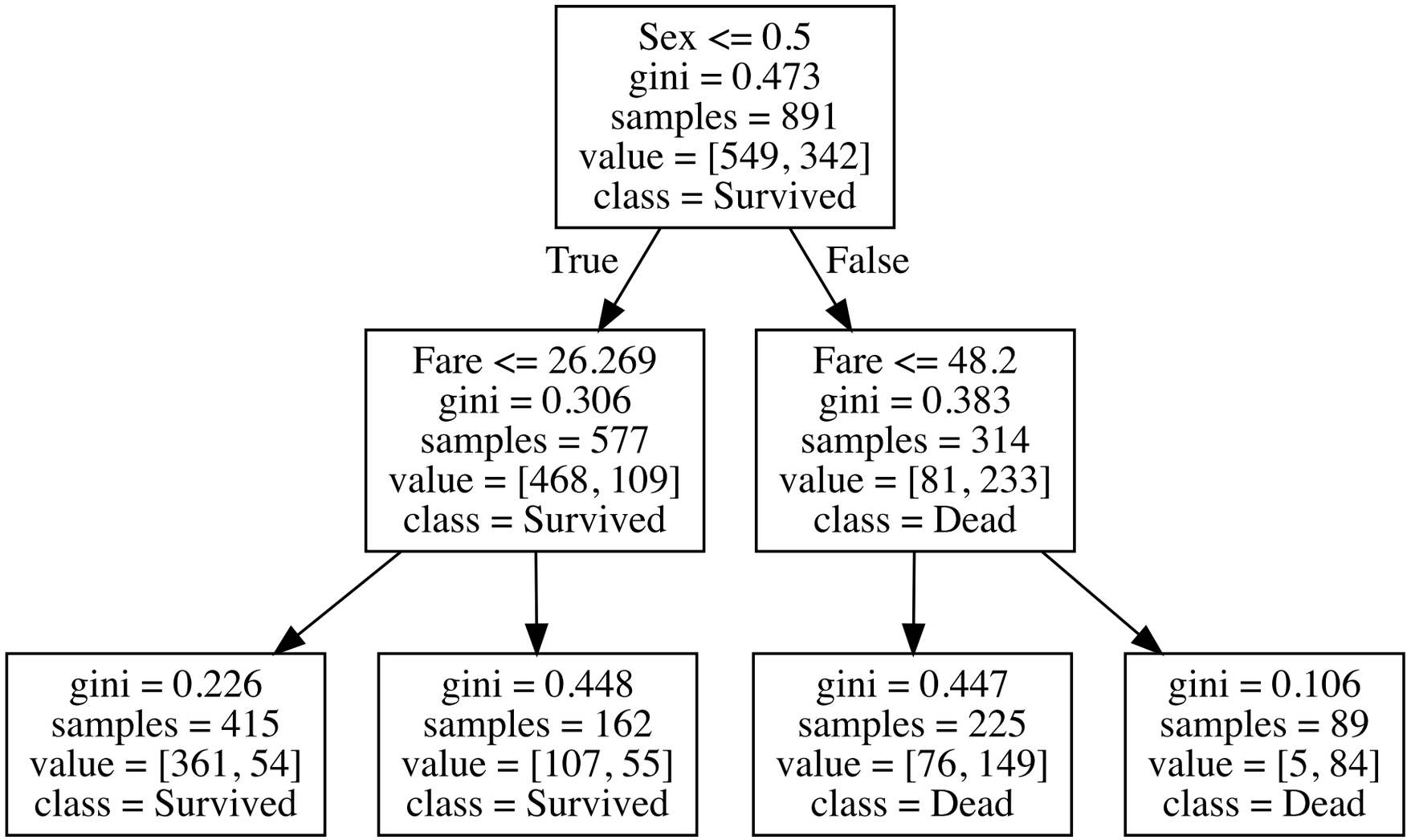
Tree methods

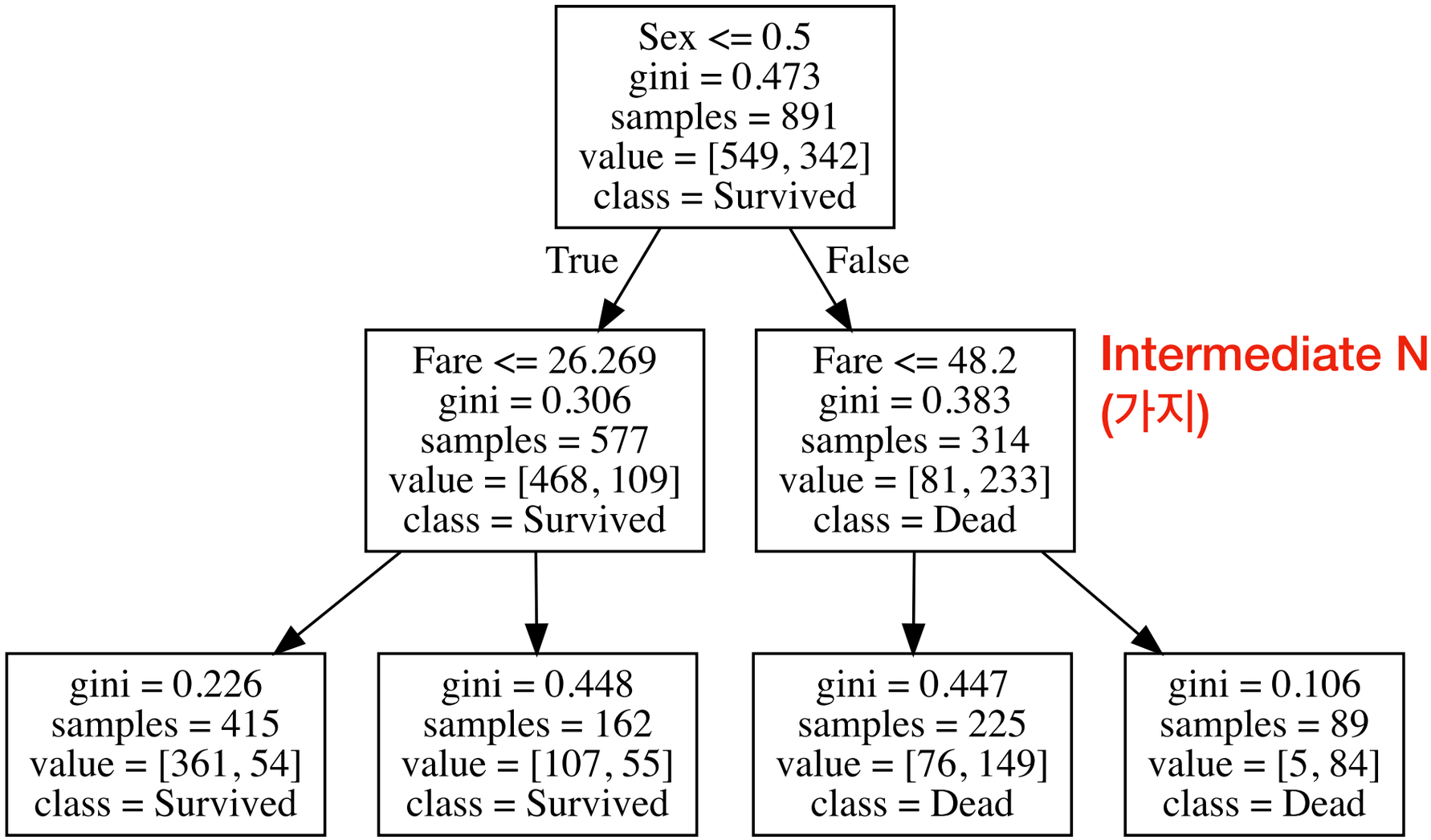
dataitgirls3

Instructor Sunmi Yoon

Decision Tree

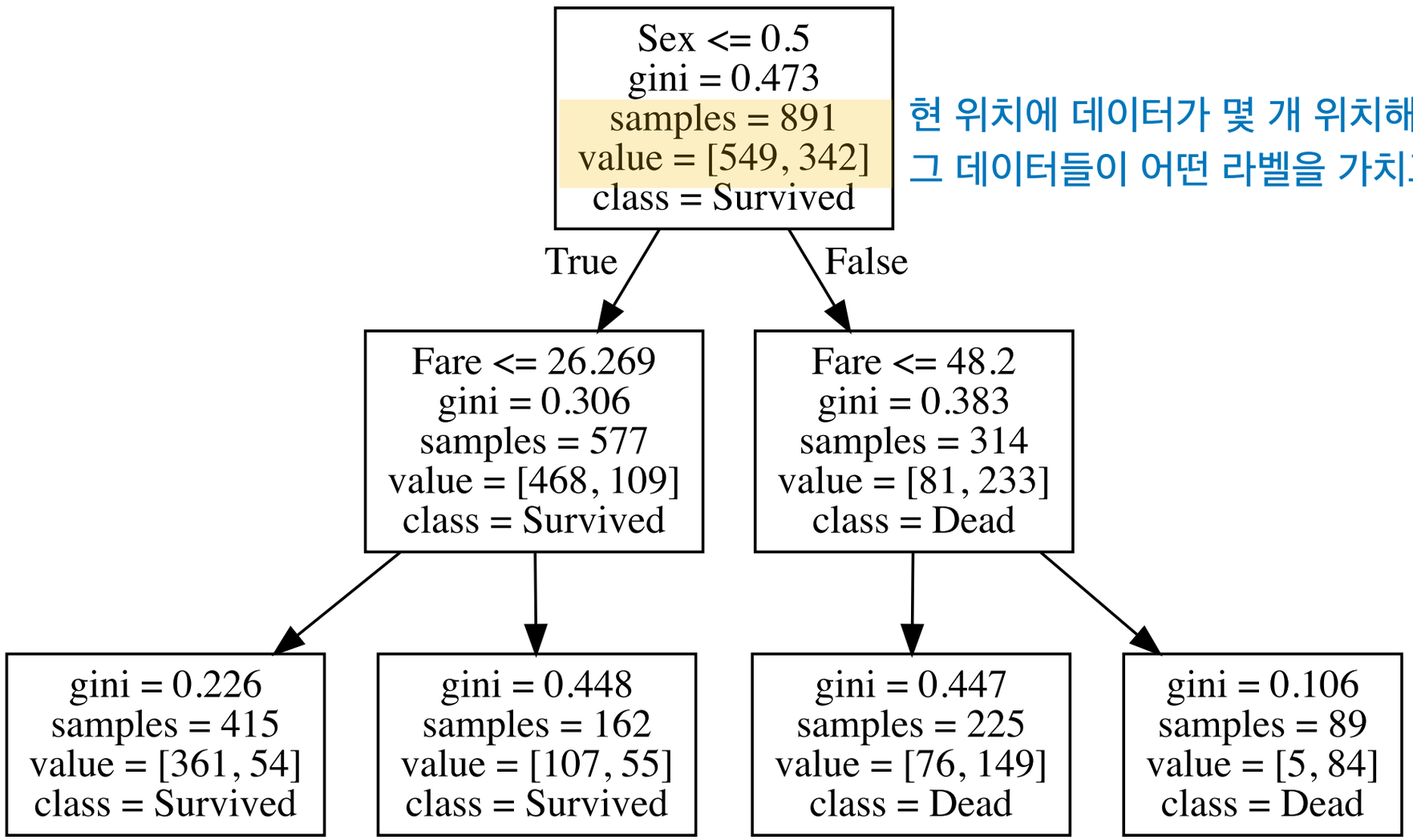

Sex <= 0.5
gini = 0.473
samples = 891
value = [549, 342]
class = Survived
True
False
Fare <= 26.269
gini = 0.306
samples = 577
value = [468, 109]
class = Survived
Fare <= 48.2
gini = 0.383
samples = 314
value = [81, 233]
class = Dead
gini = 0.226
samples = 415
value = [361, 54]
class = Survived
gini = 0.448
samples = 162
value = [107, 55]
class = Survived
gini = 0.447
samples = 225
value = [76, 149]
class = Dead
gini = 0.106
samples = 89
value = [5, 84]
class = Dead

Root Node (뿌리)

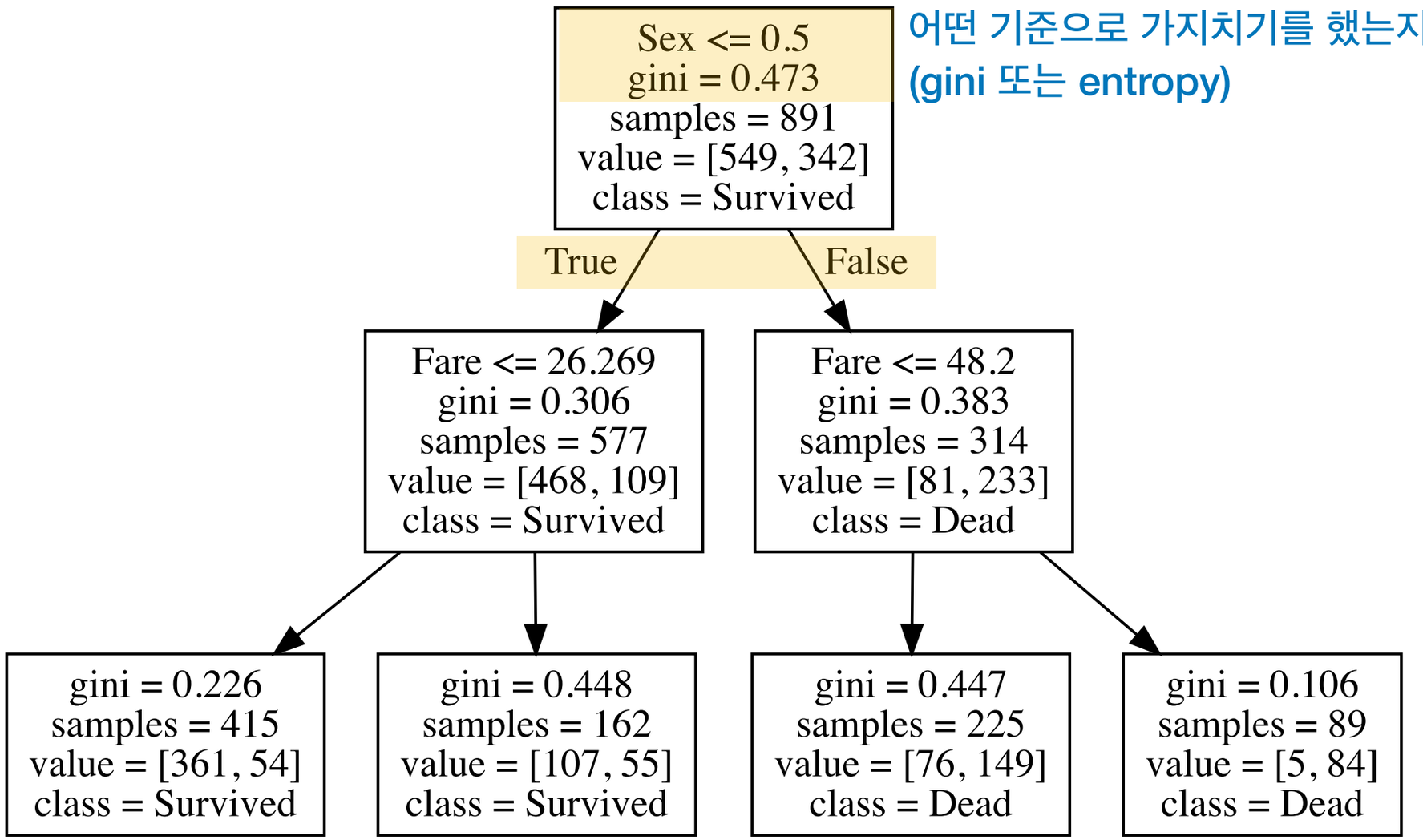


Intermediate Node (가지)

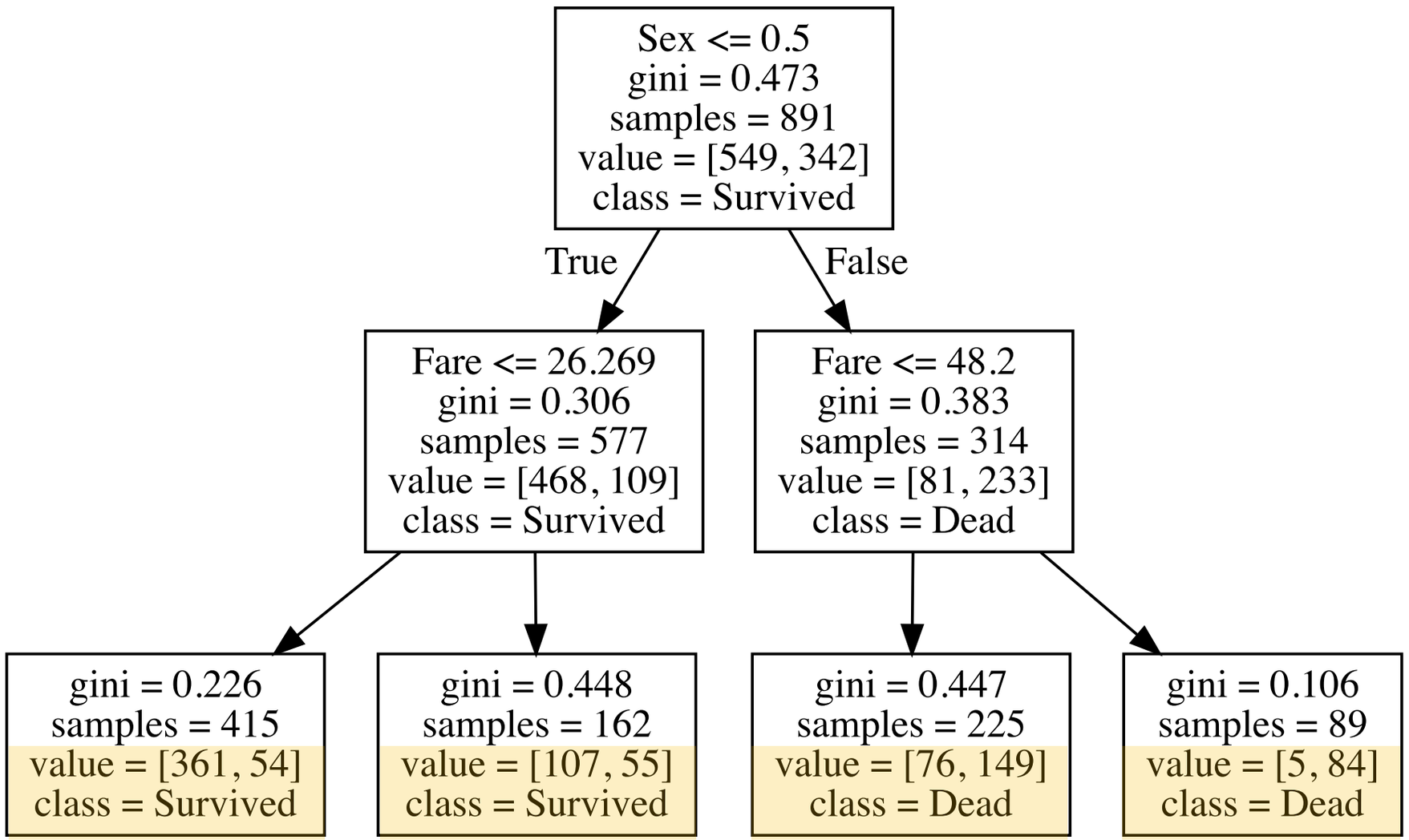
Terminal Node, Leaf (잎)



현 위치에 데이터가 몇 개 위치해 있는지
그 데이터들이 어떤 라벨을 가치고 있는지



어떤 기준으로 가지치기를 했는지
(gini 또는 entropy)



Terminal Node에 도착한 데이터들을 어떻게 분류할 것인지

sklearn Code

```
tree = DecisionTreeClassifier(random_state=42, max_depth=2, max_leaf_nodes=4)
```

```
tree = tree.fit(train_X, train_y)
tree
```

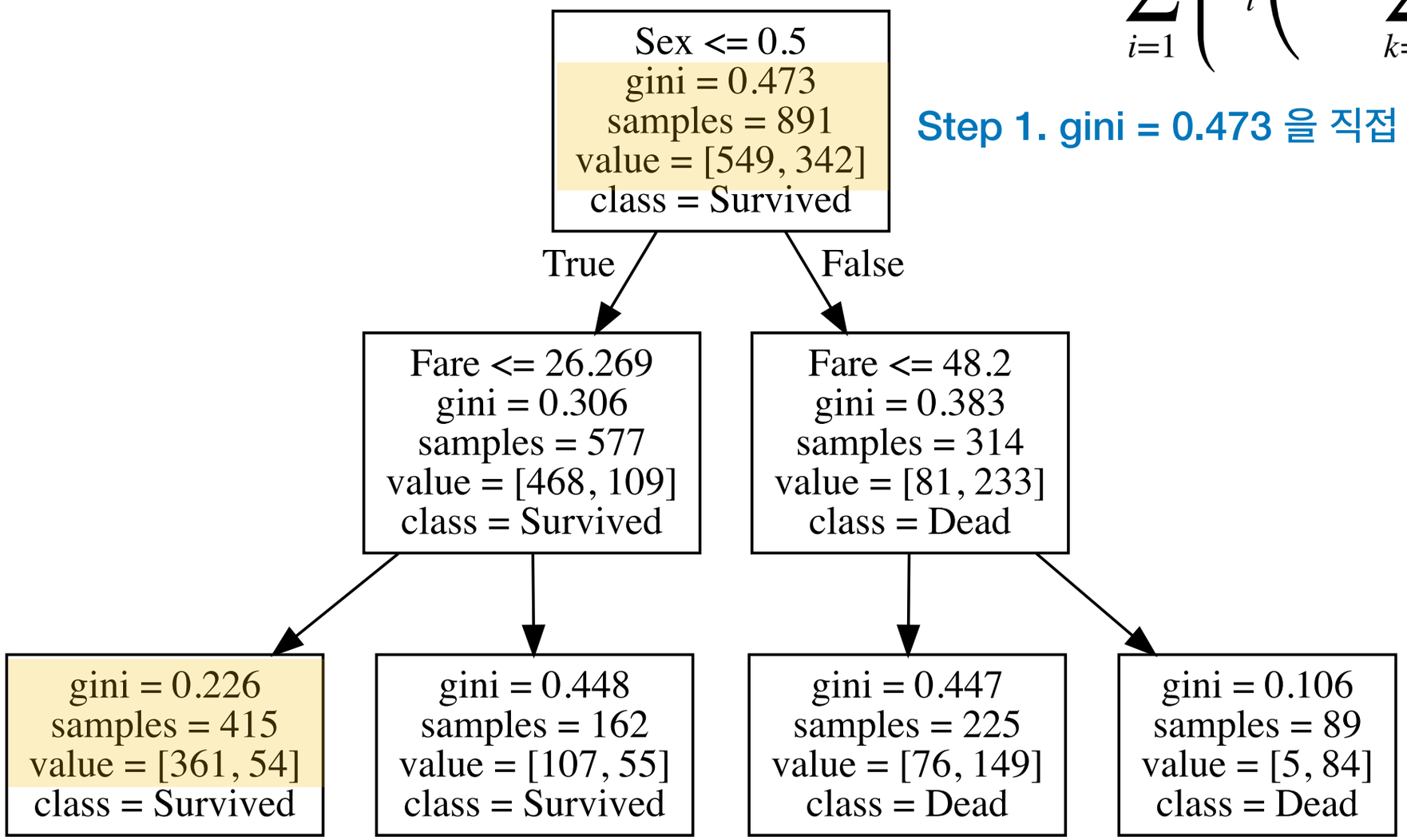
```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=2,
                       max_features=None, max_leaf_nodes=4,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=42, splitter='best')
```

Impurity

Impurity

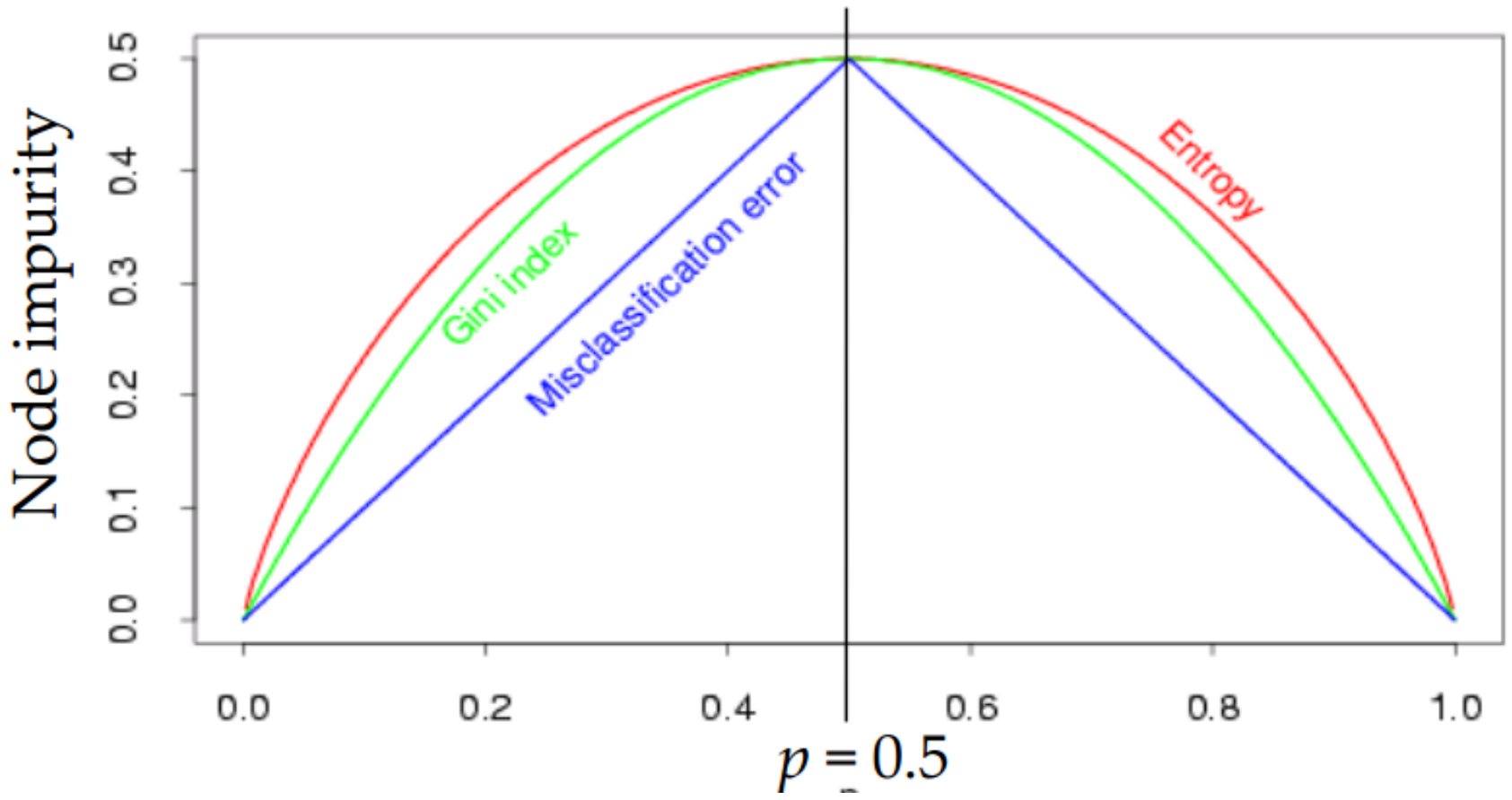
의사결정나무는 Impurity (불순도, 불확실성)이 낮아지는 방법으로 학습합니다.
순도가 증가하는 것을 두고 Information gain이라고 하기도 합니다.
오늘은 의사결정나무의 불순도 측정 방법 중, Gini Index를 공부합니다.

$$G = \sum_{i=1}^{d} \left(R_i \left(1 - \sum_{k=1}^{m} p_{ik}^2 \right) \right)$$



Step 1. gini = 0.473 을 직접 계산해 얻어보세요

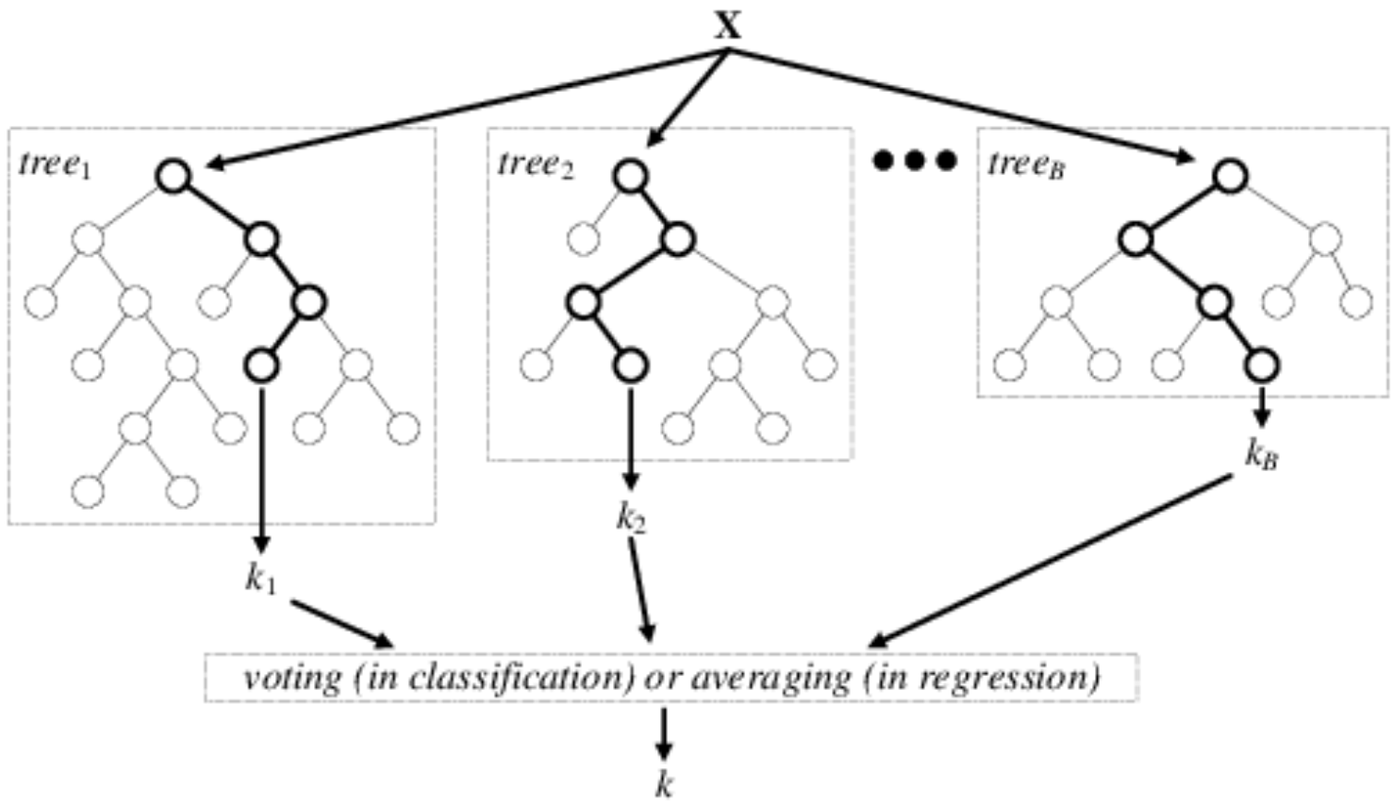
Step 2. gini = 0.226 을 직접 계산해 얻어보세요



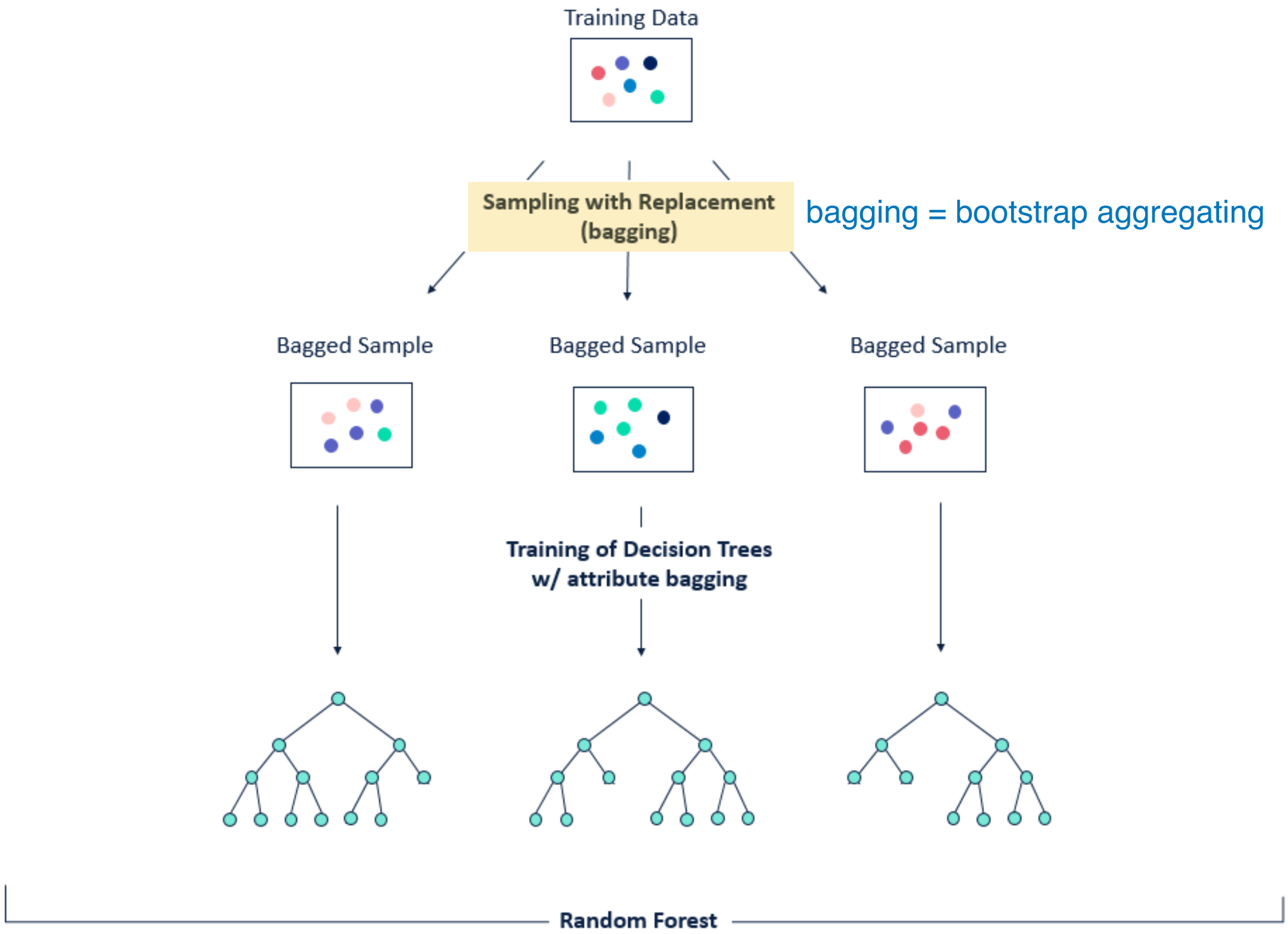
https://imgur.com/n3MVwHW

Random Forest

여러 트리들을 '다르게' 만든다.



https://www.researchgate.net/figure/Architecture-of-the-random-forest-model_fig1_301638643



https://community.alteryx.com/t5/Alteryx-Designer-Knowledge-Base/Seeing-the-Forest-for-the-Trees-An-Introduction-to-Random-Forest/ta-p/158062

Bagging

배깅(bagging)은 bootstrap aggregating의 약자로, 부트스트랩(bootstrap)을 통해 조금씩 다른 훈련 데이터에 대해 훈련된 기초 분류기(base learner)들을 결합(aggregating)시키는 방법이다.

부트스트랩이란, 주어진 훈련 데이터에서 중복을 허용하여 원 데이터셋과 같은 크기의 데이터셋을 만드는 과정을 말한다. 배깅을 통해 랜덤 포레스트를 훈련시키는 과정은 다음과 같이 세 단계로 진행된다.

1. 부트스트랩 방법을 통해 N개의 훈련 데이터셋을 생성한다.
2. N개의 기초 분류기(트리)들을 훈련시킨다.
3. 기초 분류기(트리)들을 하나의 분류기(랜덤 포레스트)로 결합한다(평균 또는 과반수투표 방식 이용).

Wikipedia 랜덤포레스트 > 배깅을 이용한 포레스트 구성

sklearn Code

```
forest = RandomForestClassifier(random_state=42, n_estimators=100,
                                max_features='auto', n_jobs=2, verbose=True)
```

```
forest = forest.fit(train_X, train_y)
forest
```

```
[Parallel(n_jobs=2)]: Using backend ThreadingBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done  46 tasks      | elapsed:    0.1s
[Parallel(n_jobs=2)]: Done 100 out of 100 | elapsed:    0.1s finished
```

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                       max_depth=None, max_features='auto', max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=2,
                       oob_score=False, random_state=42, verbose=True,
                       warm_start=False)
```

끝