지도학습 : 트리 모델(Tree Model)

Decision Tree 모델

Decision Tree 기초 개념

Decision Tree(의사결정 나무)

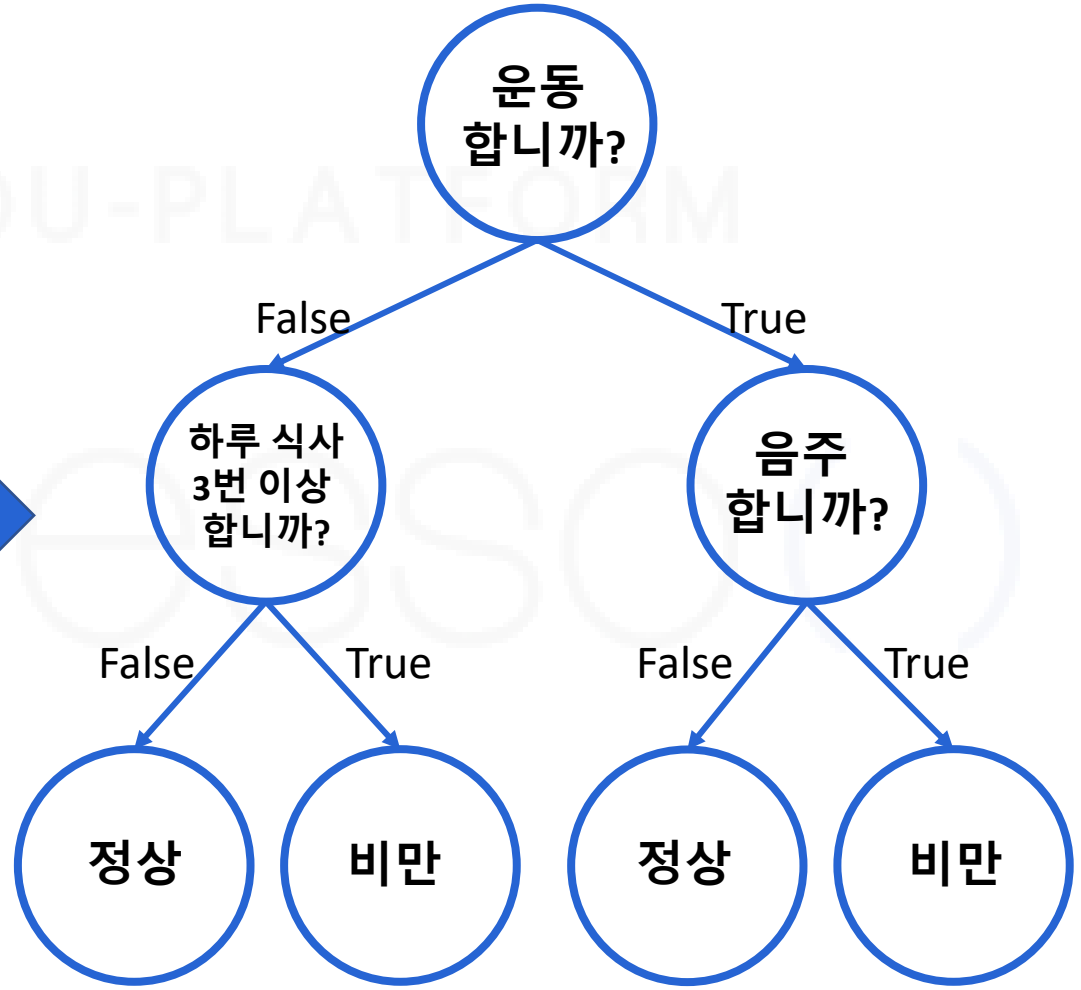
- **지도 학습 모델**
- **분류와 회귀 모두 사용 가능**
  - Classification Tree
  - Regression Tree
- **컴퓨터공학에서 사용하는 Tree 자료구조를 활용**
- **스무고개와 유사한 방법으로 분류 라벨을 결정**

Decision Tree의 쉬운 예시

- 정상, 비만을 분류하는 Decision Tree 모델

id	운동여부	음주여부	식사횟수	독서여부	라벨
1	True	False	2	False	정상
2	False	True	3	False	비만
3	True	True	2	True	정상
4	False	True	4	True	비만
5	True	False	6	False	비만

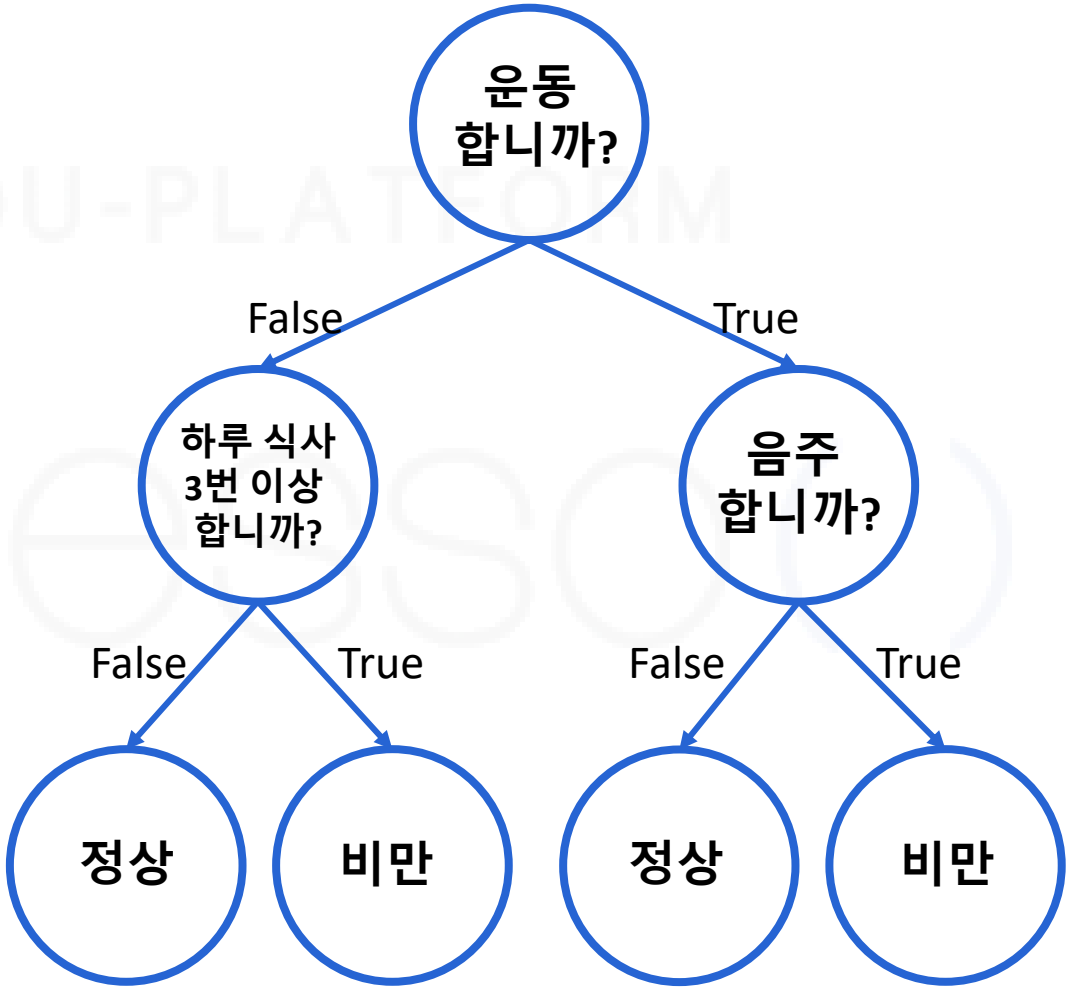
모델 학습



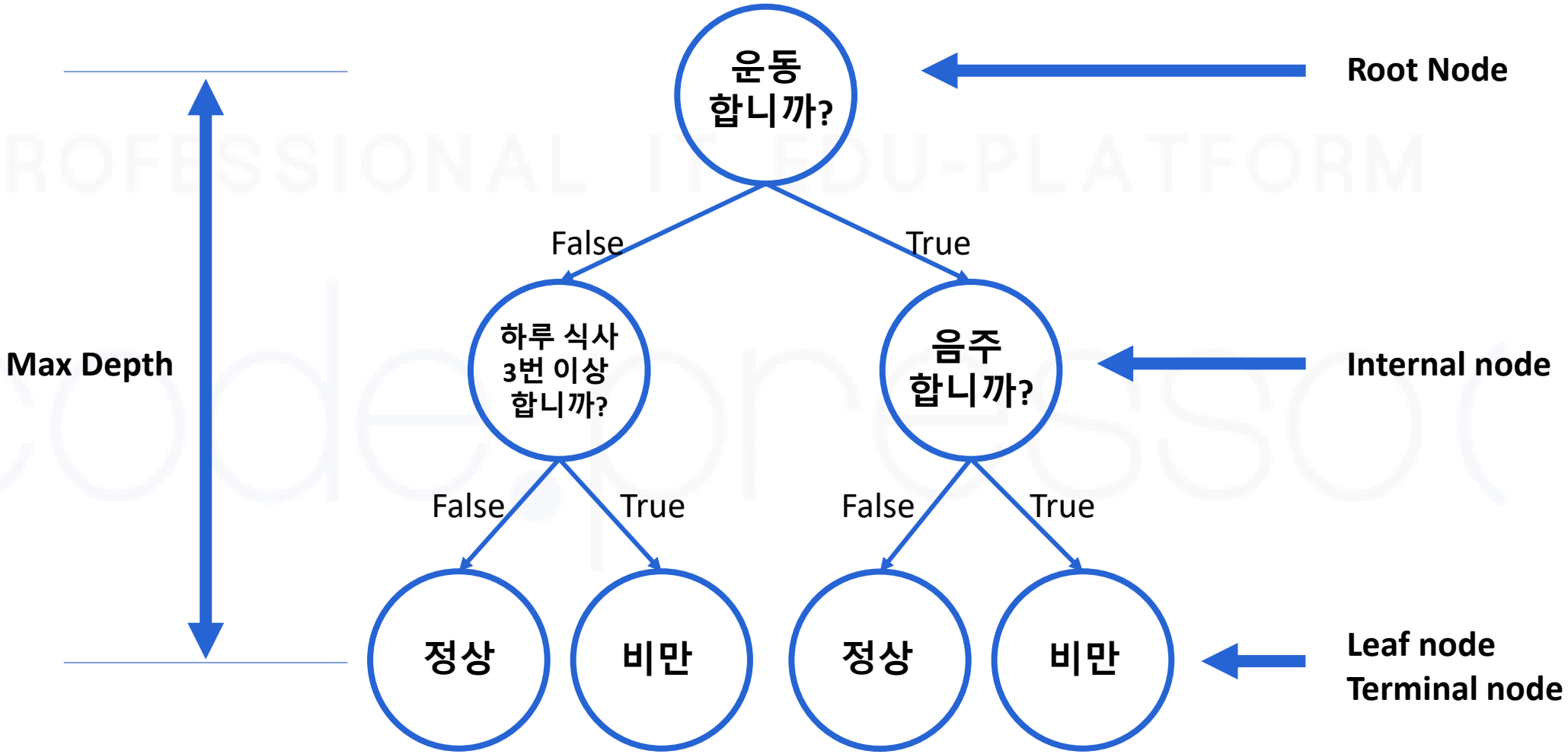
Decision Tree의 쉬운 예시

- 신규 데이터 정상, 비만 분류

id	운동여부	음주여부	식사횟수	독서여부	라벨
1	True	True	4	False	비만
2	False	True	1	False	정상



Decision Tree 주요 용어



Decision Tree 구현

- 데이터 준비 – 위스콘신 유방암 데이터

Code

```
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split

cancer = load_breast_cancer()

x_train, x_test, y_train, y_test = train_test_split(cancer.data,
                                                    cancer.target,
                                                    test_size=0.3,
                                                    random_state=12)
```

Decision Tree 구현

- **Decision Tree 모델 객체 생성, 학습**

Code

```
# DecisionTreeClassifier 임포트
from sklearn.tree import DecisionTreeClassifier

# DecisionTreeClassifier 객체 생성
dt = DecisionTreeClassifier(random_state=12)

# fit 함수로 Decision Tree 모델 학습
dt.fit(x_train, y_train)

# 학습 된 Tree의 Depth 확인 - get_depth() 함수 사용
print("Depth of tree: ", dt.get_depth())
# 학습 된 Tree의 리프 노드 개수 확인 - get_n_leaves() 함수 사용
print("Number of leaves: ", dt.get_n_leaves())
```

Decision Tree 구현

- **Decision Tree 모델 검증**

Code

```
# 테스트 세트 라벨 예측
y_pred = dt.predict(x_test)

from sklearn.metrics import accuracy_score, precision_score, recall_score
# accuracy, precision, recall 계산
accuracy = accuracy_score(y_pred, y_test)
precision = precision_score(y_pred, y_test)
recall = recall_score(y_pred, y_test)

# 성능 지표 출력
print("Accuracy: {:.3f}".format(accuracy))
print("Precision: {:.3f}".format(precision))
print("Recall: {:.3f}".format(recall))
```

Decision Tree의 학습(Fitting)

Decision Tree의 학습

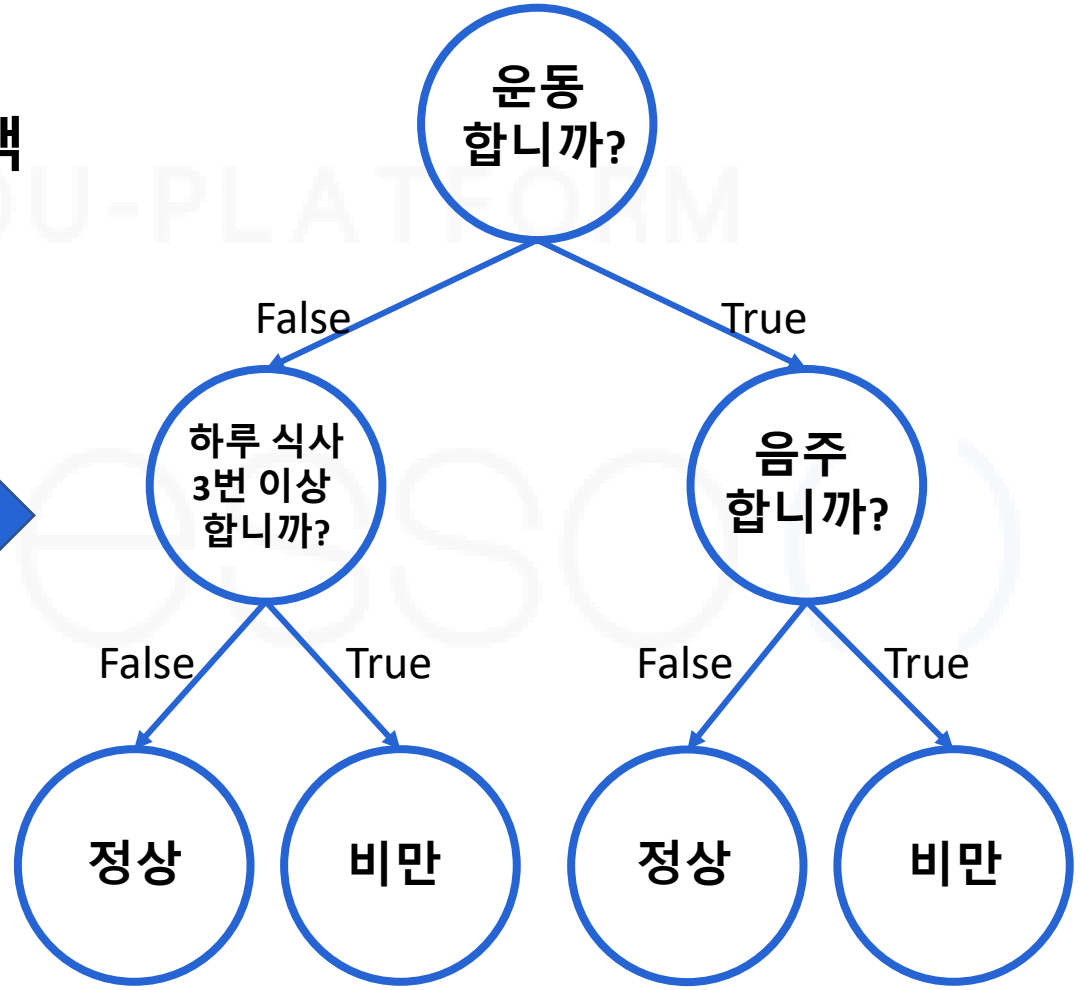
- **Decision Tree의 학습은 적절한 질문을 고르는 것이 중요**
  - 스무고개의 경우에도 좋은 질문을 먼저 던질수록 정답을 빨리 맞춤
- **정상/비만을 분류하는 모델을 만들기 위해서**
  - 운동 여부, 음주 여부, 식사 회수는 중요한 질문
  - 독서 여부는 중요하지 않은 질문
- **성별을 분류하는 모델을 만들기 위해서**
  - 키, 몸무게, 신발 사이즈, 머리카락 길이는 중요한 질문
  - 나이, IQ, 거주 도시는 중요하지 않은 질문

Decision Tree의 학습

- 운동 여부를 첫 번째 질문으로
- 식사회수와 음주여부를 두 번째 질문으로 선택

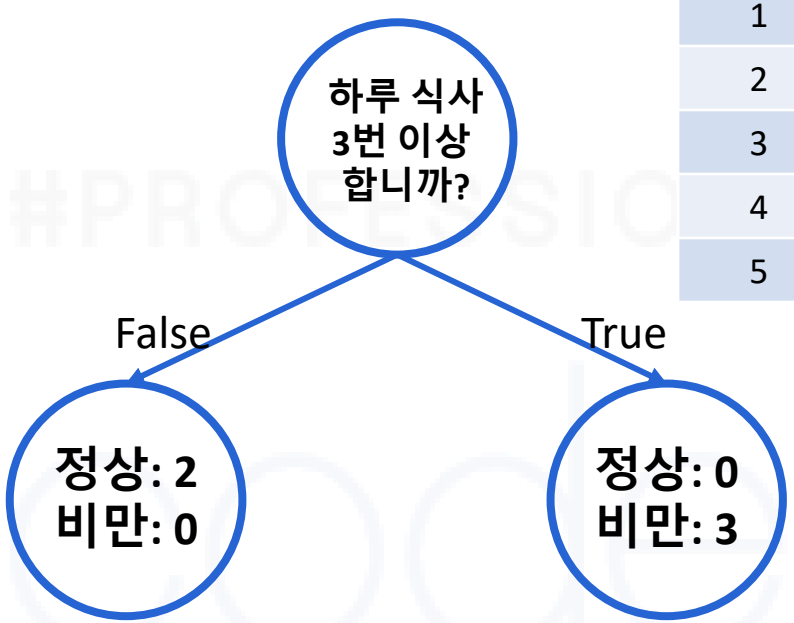
id	운동여부	음주여부	식사횟수	독서여부	라벨
1	True	False	2	False	정상
2	False	True	3	False	비만
3	True	True	2	True	정상
4	False	True	4	True	비만
5	True	False	6	False	비만

모델 학습



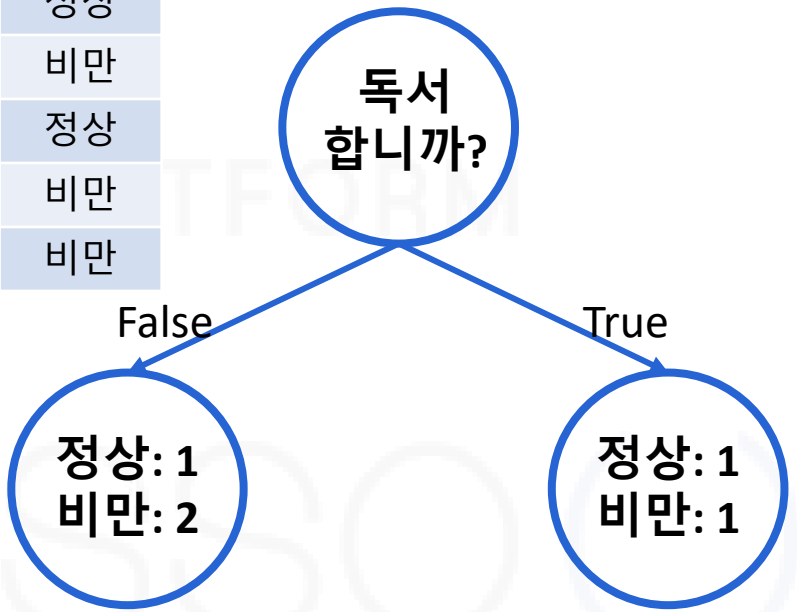
Decision Tree는 어떤 기준으로 질문을 선택할까?

id	운동여부	음주여부	식사횟수	독서여부	라벨
1	True	False	2	False	정상
2	False	True	3	False	비만
3	True	True	2	True	정상
4	False	True	4	True	비만
5	True	False	6	False	비만



- 노드 내에서의 동질성 높음
- 노드의 순도 높음
- 노드의 불순도 낮음

좋은 질문!



- 노드 내에서의 동질성 낮음
- 노드의 순도 낮음
- 노드의 불순도 높음

좋지 않은 질문!

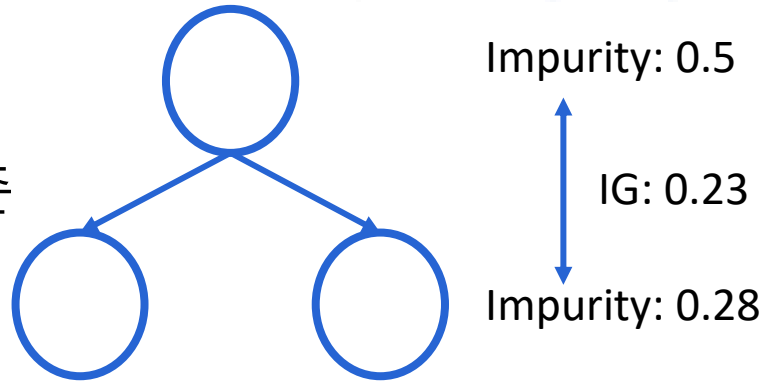
불순도, 엔트로피 그리고 정보 획득

- Decision Tree 모델은 아래 기준으로 질문(Feature)를 선택
- 다음 단계 노드들의 순도가 높아 지도록
- 다음 단계 노드들의 불순도가 낮아 지도록
- 다음 단계 노드들의 정보 획득이 높아 지도록

불순도, 엔트로피 그리고 정보 획득

- **불순도(Impurity), 엔트로피 계산을 위한 지표**
  - $Gini = 1 - \sum_{i=1}^{n} p^2(c_i)$
  - $Entropy = \sum_{i=1}^{n} -p(c_i)log_2(p(c_i))$

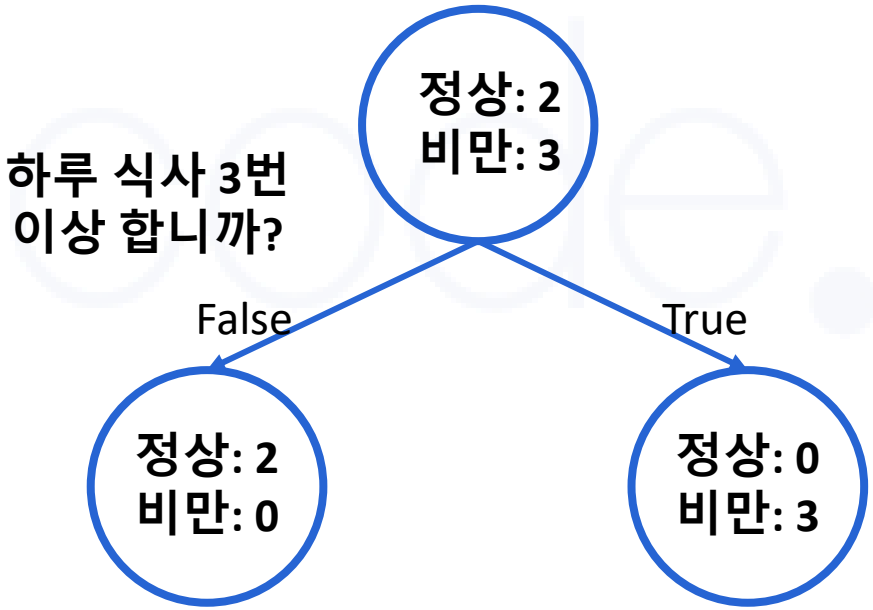
- **정보 획득(Information Gain)**
  - 특정 단계와 그 다음 단계 사이에 불순도가 감소하는 수준



Gini Impurity의 계산 예제

- 정상/ 비만 데이터로 Gini Impurity(Gini Index)의 계산

$$Gini = 1 - \sum_{i=1}^{n} p^2(c_i)$$



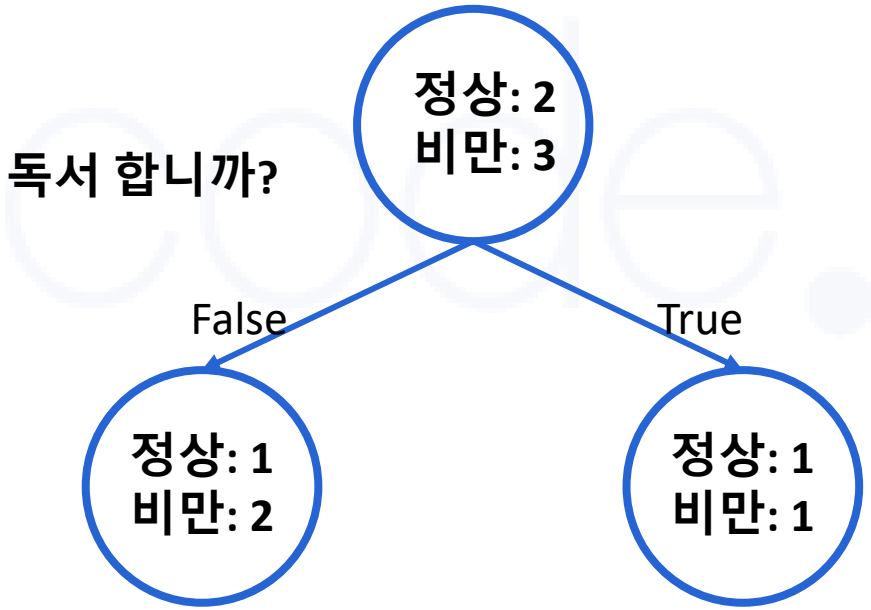
1 – (2/5)**2 – (3/5) **2 = 0.48

(1 – (2/2)**2 – (0/2)**2) +
(1 – (0/3)**2 – (3/3)**2) = 0

Gini Impurity의 계산 예제

- 정상/ 비만 데이터로 Gini Impurity(Gini Index)의 계산

$$Gini = 1 - \sum_{i=1}^{n} p^2(c_i)$$



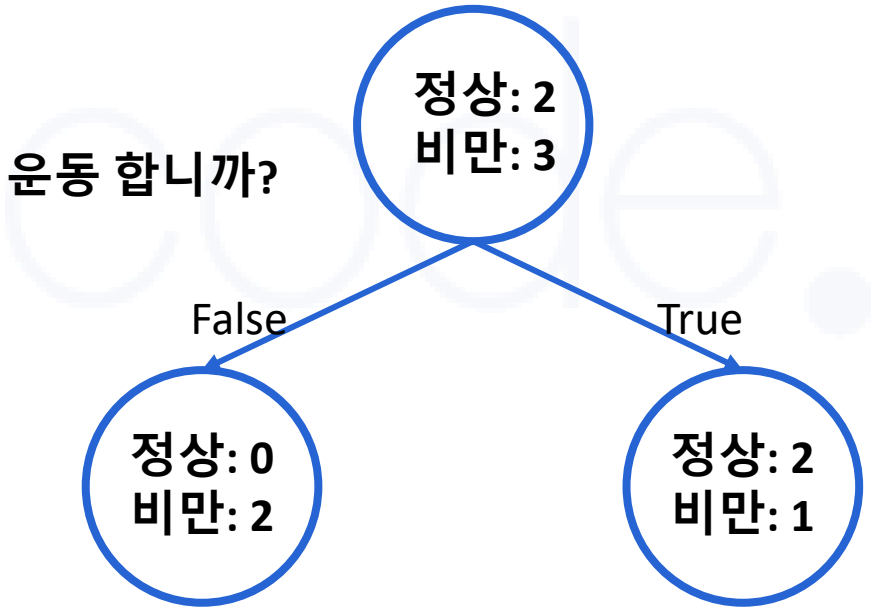
1 – (2/5)**2 – (3/5) **2 = 0.48

(1 – (1/3)**2 – (2/3)**2) +
(1 – (1/2)**2 – (1/2)**2) = 0.94

Gini Impurity의 계산 예제

- 정상/ 비만 데이터로 Gini Impurity(Gini Index)의 계산

$$Gini = 1 - \sum_{i=1}^{n} p^2(c_i)$$



1 – (2/5)**2 – (3/5) **2 = 0.48

(1 – (0/2)**2 – (2/2)**2) +
(1 – (2/3)**2 – (1/3)**2) = 0.44

Gini Impurity의 계산 예제

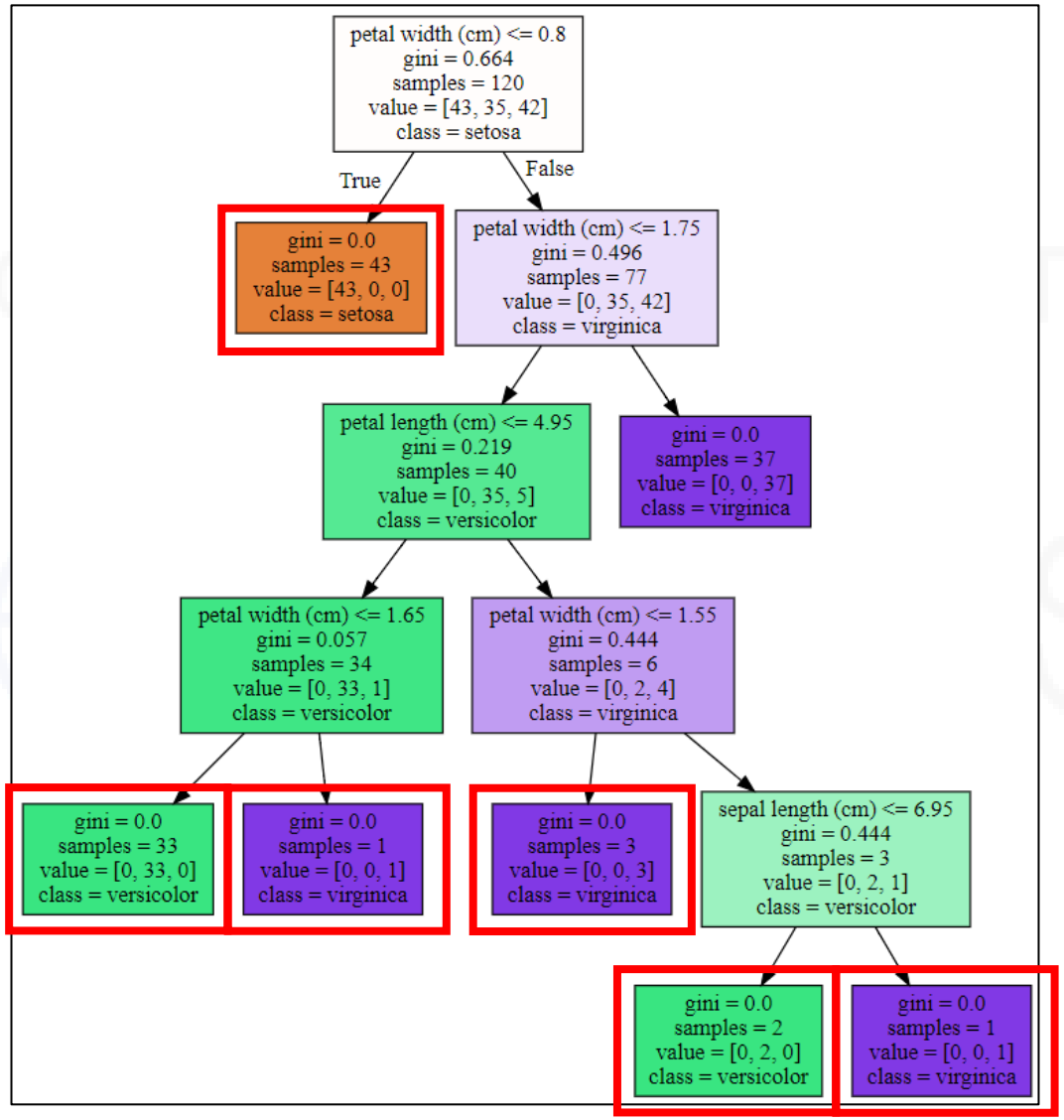
- Root node Impurity: 0.48

- 다음 단계의 Impurity
  - 식사 횟수를 선택할 경우: 0
  - 독서 여부를 선택할 경우: 0.94
  - 운동 여부를 선택할 경우: 0.44

Decision Tree의 학습

- 모든 Feature 들에 대해서 바로 다음 단계 불순도, 엔트로피 계산
- 불순도, 엔트로피가 가장 낮아지는 Feature 선택
- 모든 Leaf 노드들의 Impurity가 0이 될 때까지 반복

Decision Tree의 학습



Feature들의 중요도

- 학습 된 Decision Tree 모델은 Feature들의 중요도 정보를 포함하고 있음

Code

```
dt = DecisionTreeClassifier(random_state=12)
dt.fit(x_train, y_train)

for i in range(0,len(cancer.feature_names)):
  print('{0}: {1:.3f}'.format(cancer.feature_names[i], dt.feature_importances_[i]))
```

Result

```
mean radius: 0.000
mean texture: 0.000
mean perimeter: 0.000
mean area: 0.014
mean smoothness: 0.007
```

```
mean compactness: 0.000
mean concavity: 0.000
mean concave points: 0.041
mean symmetry: 0.000
mean fractal dimension: 0.000
```

Decision Tree의 특징 및 Hyperparameter

Decision Tree의 장점

- **학습 된 모델의 해석 력이 높음.**
  - 학습 된 모델을 사람이 이해 가능
  - 학습 된 Tree를 시각화 할 수 있음
- **Scaling, One Hot Encoding 등 데이터 전처리의 영향이 크지 않음**
- **Feature Selection이 자동으로 수행 됨**
- **학습 된 모델의 Prediction 속도가 빠름**

Decision Tree의 단점

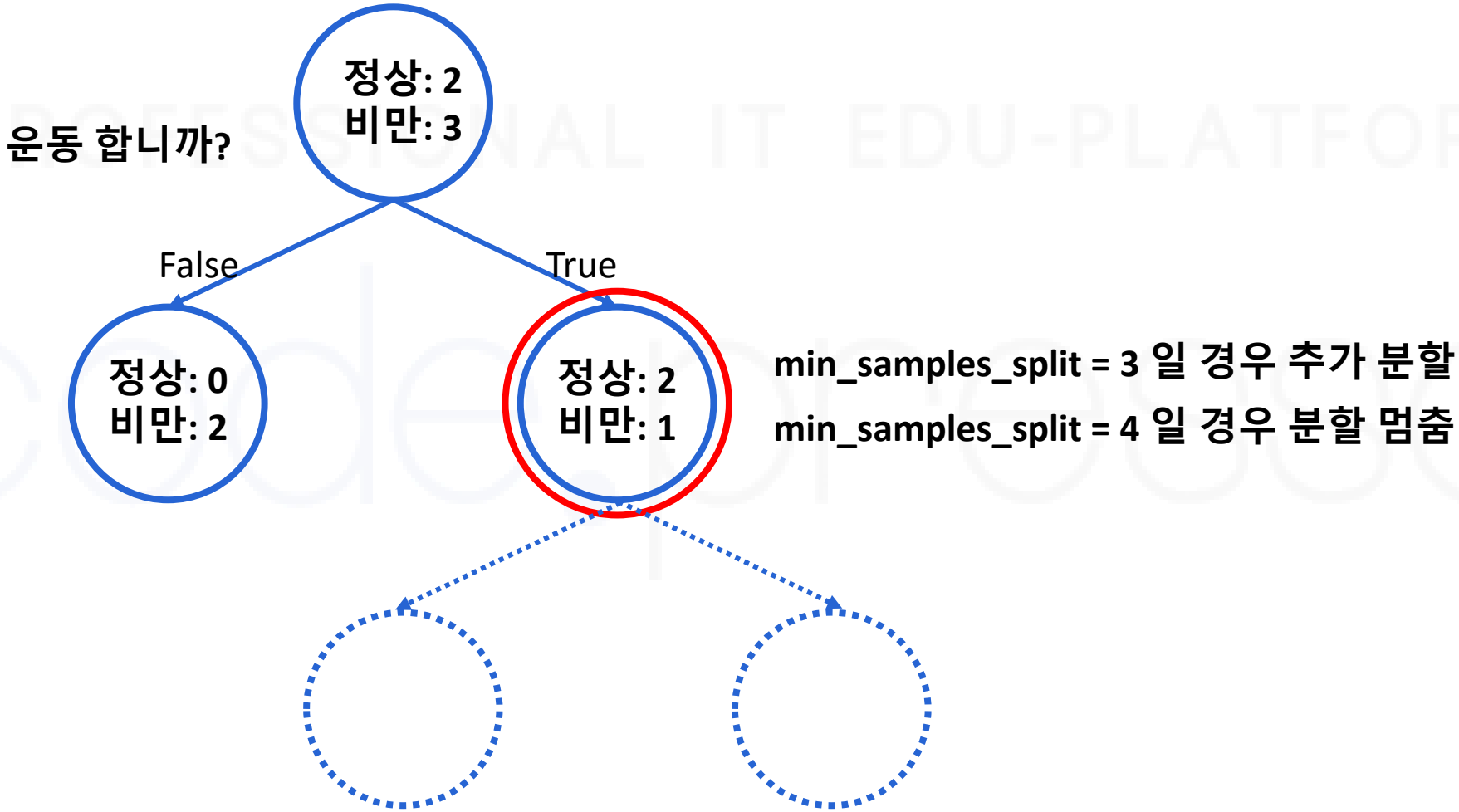
- **Overfitting 되기 쉬움**
  - 불순도/엔트로피가 0이 될 때까지 집요하게 Tree 가지를 분리
  - Training Dataset에 최적화 될 가능성 존재
- **모델을 학습하기 위한 시간/공간 복잡도가 높음**
- **다른 분류 알고리즘에 비해 예측 정확도가 상대적으로 낮음**
  - 앙상블 학습 기법으로 보완

Decision Tree의 Hyperparameter

- Decision Tree의 Hyperparameter들은 대부분 Overfitting을 완화하는 목적
- Tree의 크기를 제한함으로써 Overfitting을 완화
- Decision Tree의 학습을 일찍 종료 시켜 Tree의 크기를 제한

Hyperparameter	설명	Default
max_depth	Tree의 Depth를 제한	None
min_samples_split	노드를 분할하는 최소 샘플의 개수	2
min_samples_leaf	Leaf 노드를 결정하는 최소 샘플의 개수	1
max_leaf_nodes	Leaf 노드의 개수를 제한	None
max_features	학습 시 사용하는 Feature의 개수를 제한	None

min_samples_split 사례



Decision Tree의 Hyperparameter 실습

- Hyperparameter 설정 없이 depth와 leaf 노드 개수 값 확인

Code

```
dt = DecisionTreeClassifier(random_state=12)
dt.fit(x_train, y_train)

print("Max Depth: ", dt.get_depth())
print("Number of leaves: ", dt.get_n_leaves())
```

Decision Tree의 Hyperparameter 실습

- max_depth 설정 후 depth와 leaf 노드 개수 값 확인

Code

```
dt = DecisionTreeClassifier(max_depth=3, random_state=12)
dt.fit(x_train, y_train)

print("Max Depth: ", dt.get_depth())
print("Number of leaves: ", dt.get_n_leaves())
```

Decision Tree의 Hyperparameter 실습

- max_leaf_nodes 설정 후 depth와 leaf 노드 개수 값 확인

Code

```
dt = DecisionTreeClassifier(max_leaf_nodes=9, random_state=12)
dt.fit(x_train, y_train)

print("Max Depth: ", dt.get_depth())
print("Number of leaves: ", dt.get_n_leaves())
```

지도학습 : 트리 모델(Tree Model)

Random Forest 모델

Ensemble Learning
앙상블 학습 기법

Ensemble Learning(앙상블 학습 기법)의 개념

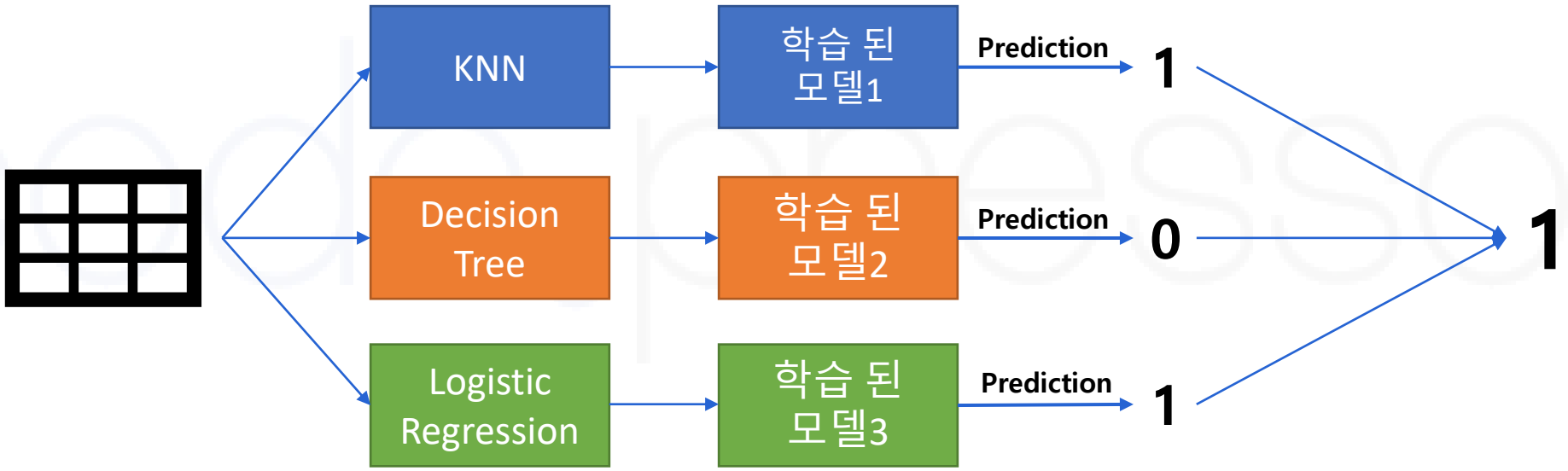
- 다수의 ML 모델을 결합하여 학습, 예측 하는 기법
- Decision Tree 등의 단순한 모델을 여러 개 결합하여 사용
- 단일 ML 모델을 사용하는 것 보다 일반적으로 예측 성능이 높음
- 최신 앙상블 기법은 정형 데이터 셋에 대해서는 딥러닝에 필적하는 성능을 보임
  - Kaggle에서 많이 사용 됨

Ensemble Learning의 종류

- Voting
- Bagging
- Boosting

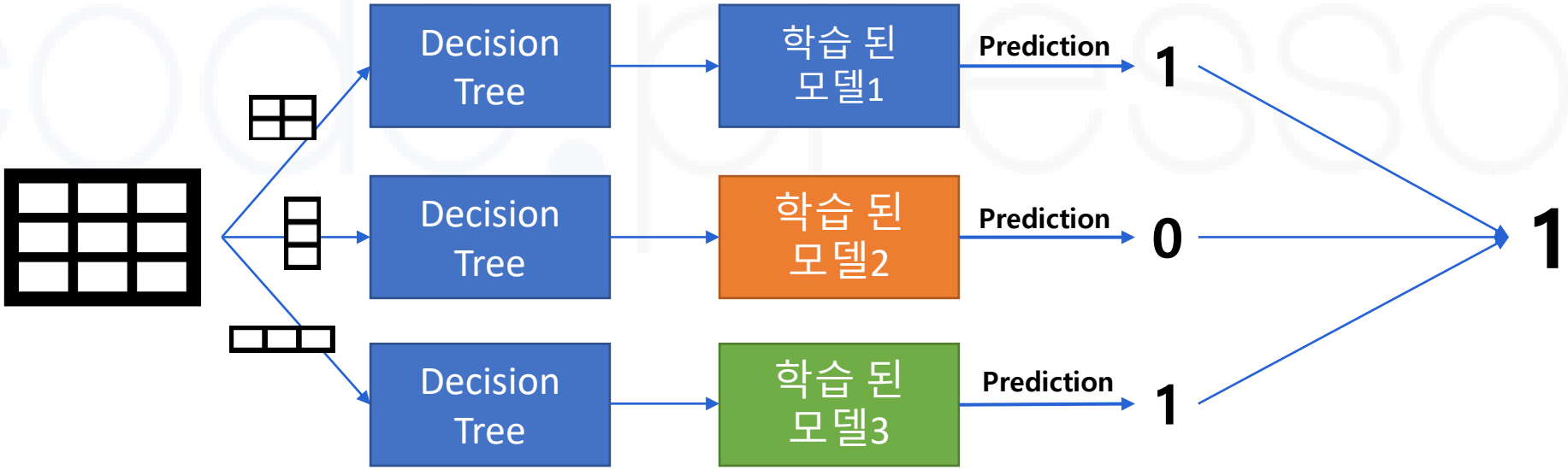
Voting 기법

- 서로 다른 ML 모델을, 동일한 데이터 세트로 학습
- 다수의 학습 된 모델로 Prediction 한 값으로 최종 투표



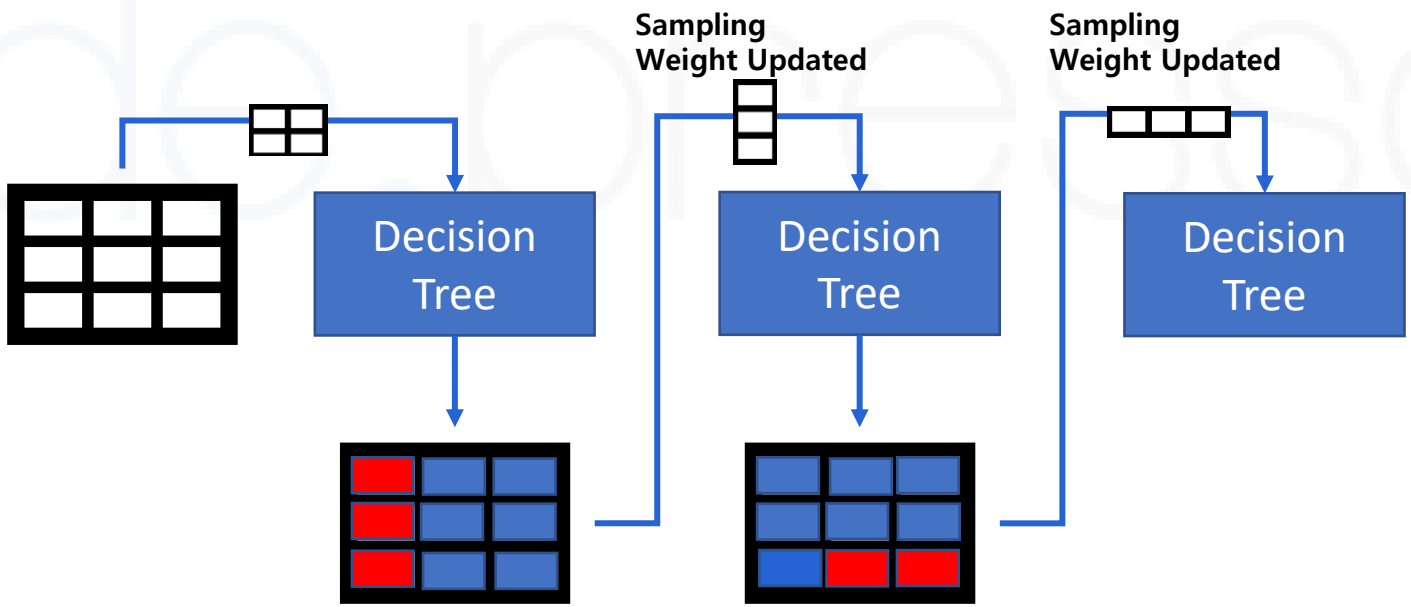
Bagging 기법

- Bagging은 Bootstrap Aggregating의 약자
- 전체 학습 데이터 셋에서 무작위 복원 샘플 데이터 추출
- 동일한 ML 모델을, 서로 다른 샘플 데이터 셋으로 학습. 각 학습은 독립적으로 수행



Boosting 기법

- Bagging과 유사하게 동일한 ML 모델을, 서로 다른 샘플 데이터 셋으로 학습
- 학습이 순차적이며, 이전 단계의 학습 결과를 토대로 다음 단계 샘플링에 가중치를 결정
- 예측 성능이 가장 높지만, 학습 속도가 매우 느림



Voting 기법의 구현

- Logistic Regression과 Decision Tree를 활용한 Voting Classifier

Code

```
from sklearn.ensemble import VotingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier

# LogisticRegression 및 DecisionTreeClassifier 객체 생성
lr = LogisticRegression(random_state=12)
dt = DecisionTreeClassifier(random_state=12)

# VotingClassifier 객체 생성
voting = VotingClassifier(estimators=[('LR',lr), ('DT',dt)], voting='soft')

# VotingClassifier 학습 및 검증
voting.fit(x_train , y_train)
pred = voting.predict(x_test)

print('Accuracy: {0:.3f}'.format(accuracy_score(y_test , pred)))
```

Random Forest 모델

Random Forest 모델

- Bagging 기법을 사용하는 대표적인 앙상블 학습 모델
- Base 모델로 Decision Tree를 사용
- 일반적인 특징은 Decision Tree와 유사함
- Decision Tree의 장점인 높은 모델 해석 력은 해당하지 않음

Random Forest 모델의 장점/단점

- **장점**
  - Decision Tree의 장점을 대부분 포함(모델 해석력 제외)
  - Decision Tree에 비해 Overfitting의 위험성이 상대적으로 적음
  - 단일 ML 모델보다 일반적으로 예측 성능이 높음
- **단점**
  - 느린 학습 속도
  - Hyperparameter 튜닝의 어려움(너무 많은 조합이 가능)

Scikit Learn Random Forest 실습

- **RandomForestClassifier 클래스 사용**
- **대부분의 Hyperparameter는 Decision Tree와 유사**
  - max_depth, max_leaf_nodes, max_features, …
- **n_estimators: 생성되는 Decision Tree의 개수**
  - Dafault 값은 100: 100개의 Decistion Tree 학습 및 예측 Voting
  - 많이 생성한다고 무한대로 예측 성능이 좋아 지지는 않음

Scikit Learn Random Forest 실습

- **RandomForestClassifier 사용**

Code

```
# RandomForestClassifier 임포트
from sklearn.ensemble import RandomForestClassifier

# RandomForestClassifier 객체 생성
rf = RandomForestClassifier(max_depth=5, random_state=12)

# RandomForestClassifier 객체 학습 및 검증
rf.fit(x_train, y_train)
pred = rf.predict(x_test)

print('Accuracy: {0:.3f}'.format(accuracy_score(y_test, pred)))
```