파이썬을 이용한
빅데이터 분석

(유성준, 21세기사)

Chapter 4

트리를 이용한 데이터 분석

4.1 의사결정 트리를 이용한 데이터분석

1. Decision Tree 개념

Decision Tree란?

- 의사결정 트리 또는 **의사결정 나무(Decision Tree)**는 기계학습(Machine Learning)에서 지도학습(Supervised Learning)의 알고리즘으로 분류(Classification) 또는 회귀(Regression)분석 목적으로 사용

- 의사결정 규칙(Decision Rule)을 나무구조 표현을 통해 분류와 예측을 수행하는 분석 방법

- 분류 또는 예측 과정이 나무구조로 표현되어 비교적 쉽게 이해

- 목표변수 유형에 따른 의사결정 트리
  - 범주형 목표변수 : 분류 트리(Classification Tree)
    - 예) 성별(남,여), 혈액형(A, B, O, AB) 등
  - 연속형 목표변수 : 회귀 트리(Regression Tree)
    - 목표변수가 연속형인 경우, 평균과 표준편차에 기초해 분리 발생 → 회귀 트리 구성, 예) 가격, 키, 비율 등

1. Decision Tree 개념

Decision Tree 구성요소

- 뿌리 마디(Root Node)
  트리 구조가 시작되는 마디, 전체 자료로 구성

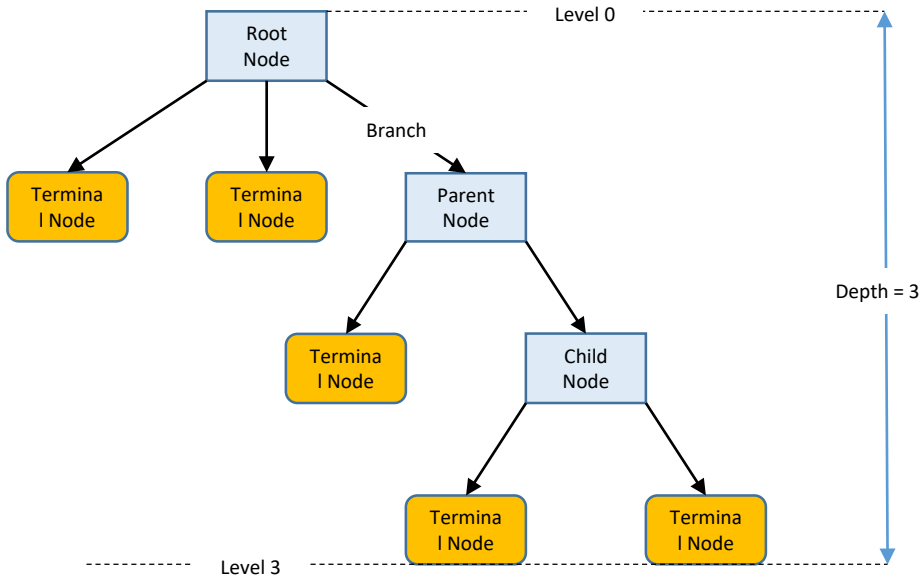
- 부모 마디(Parent Node)
  자식 마디의 상위 마디

- 자식 마디(Child Node)
  하나의 마디로부터 분리되어 나간 2개 이상의 마디들을 의미

- 끝 마디(Terminal Node) 또는 잎(Leaf Node)
  트리 줄기의 끝에 위치하고 있고 자식 마디가 없는 마디

- 가지(Branch)
  뿌리 마디로부터 끝 마디까지 연결된 마디들

- 깊이(Depth)
  뿌리 마디로부터 끝 마디를 이루는 층의 수



1. Decision Tree 개념

Decision Tree 분석과정

- 성장(Growing)
  분석의 목적에 따라 각 마디에서 적절한 최적의 분리기준(Split Criterion)을 찾아 트리를 성장시키는 과정, 적절한 정지규칙(Stopping Rule)을 통한 의사결정 트리 도출

- 가지치기(Pruning)
  오분류를 크게 할 위험(Risk)이 높거나 부적절한 추론규칙(Induction Rule)을 가지는 불필요한 가지(Branch)를 제거

- 타당성 평가
  이익 도표(Gain Chart), 위험 도표(Risk Chart) 또는 검증용 자료(Test Data)로 의사결정 트리 평가

- 해석 및 예측
  의사결정 트리를 해석하고 예측모형 결정

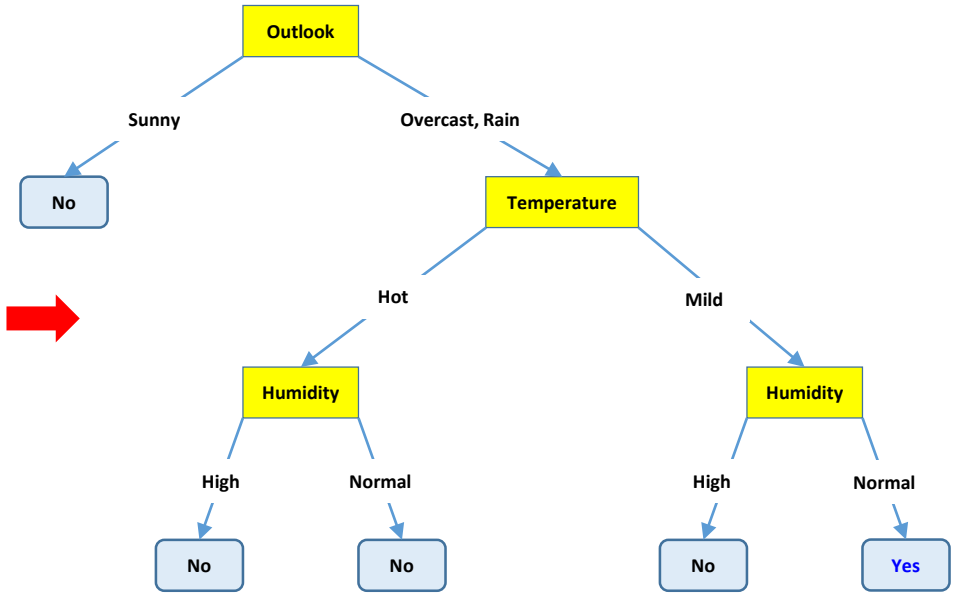
1. Decision Tree 개념

예제를 이용한 Decision Tree 동작 과정

Training Data

Day	Outlook (전망)	Temperature	Humidity	Play Tennis
D1	Sunny	Hot	High	No
D2	Overcast	Hot	High	No
D3	Overcast	Mild	Normal	Yes
D4	Rain	Mild	Normal	Yes
D5	Sunny	Mild	High	No
D6	Rain	Hot	High	No
D7	Overcast	Hot	Normal	No

Model : Decision Tree

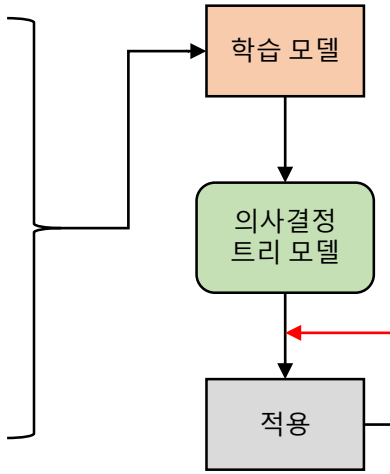


1. Decision Tree 개념

예제를 이용한 Decision Tree 동작 과정

Training Data Set

Day	Outlook	Temperature	Humidity	Play Tennis
D1	Sunny	Hot	High	No
D2	Overcast	Hot	High	No
D3	Overcast	Mild	Normal	Yes
D4	Rain	Mild	Normal	Yes
D5	Sunny	Mild	High	No
D6	Rain	Hot	High	No
D7	Overcast	Hot	Normal	No



Test Data Set

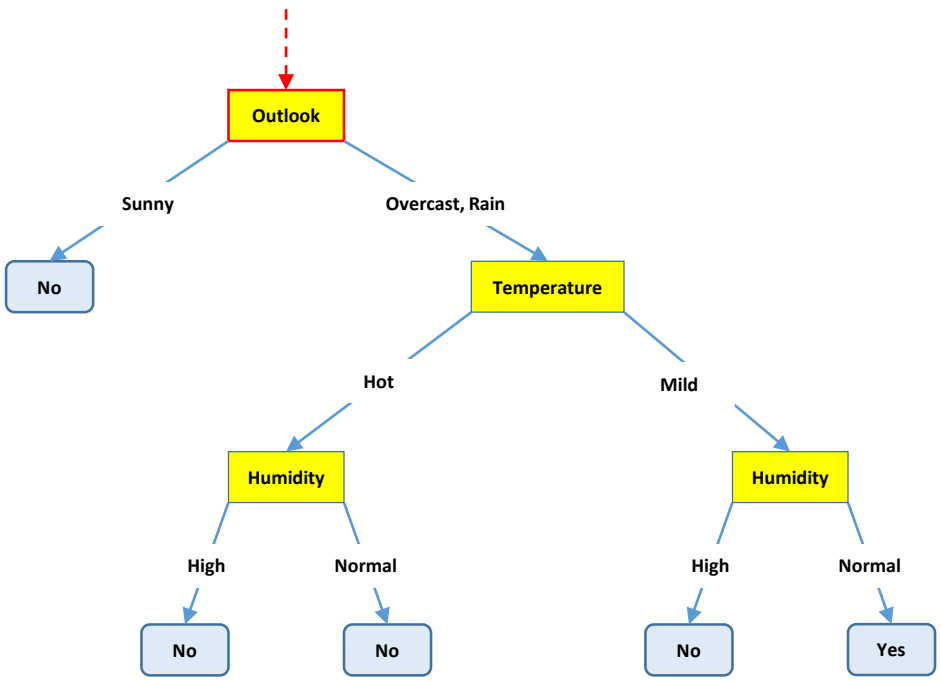
Day	Outlook	Temperature	Humidity	Play Tennis
D8	Rain	Mild	High	?
D9	Overcast	Mild	High	?
D10	Sunny	Hot	Normal	?

결과 추론

1. Decision Tree 개념

예제를 이용한 Decision Tree 동작 과정

트리 구조의 뿌리 노드로 부터 시작

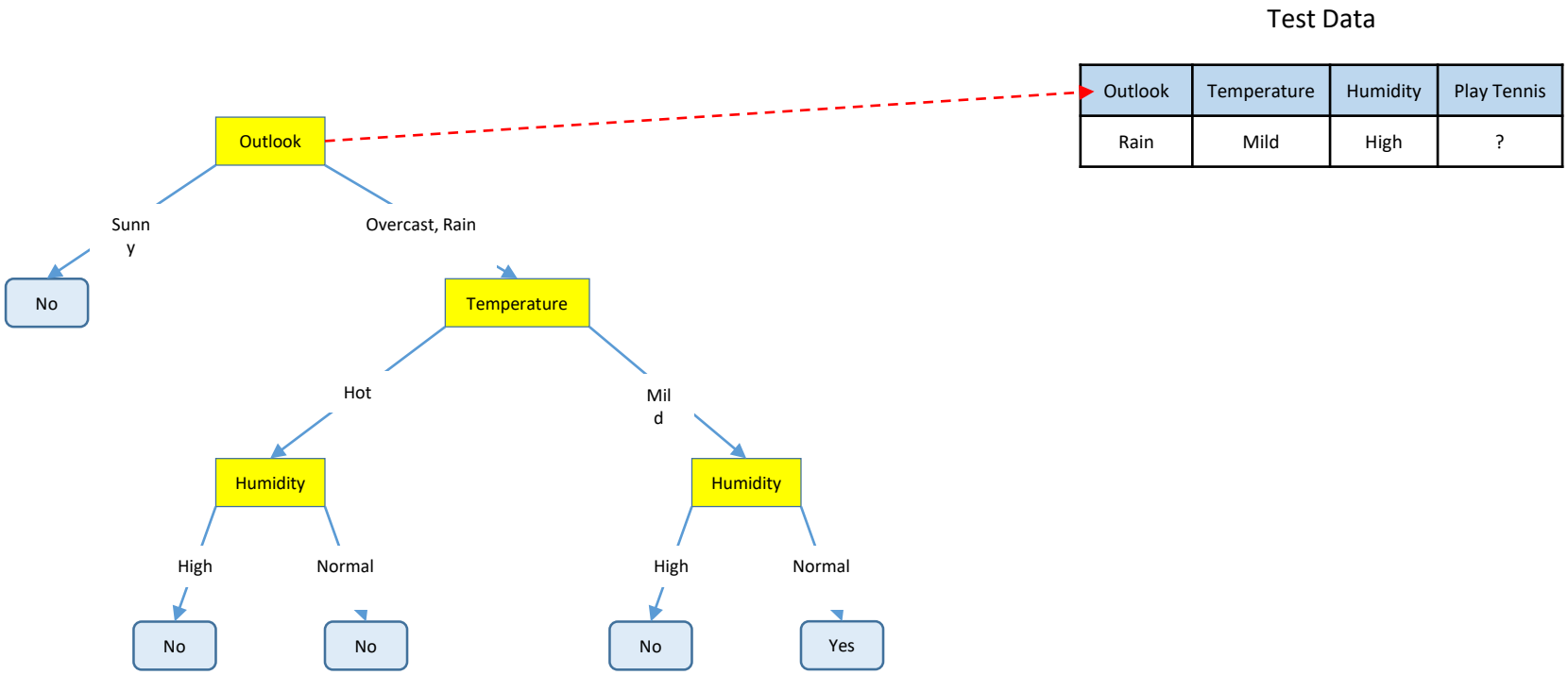


Test Data

Outlook	Temperature	Humidity	Play Tennis
Rain	Mild	High	?

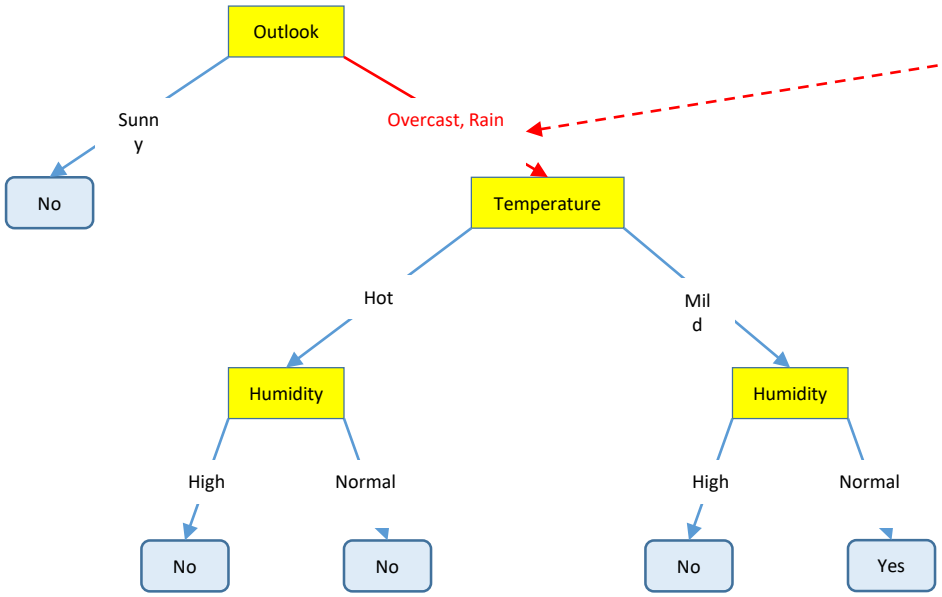
1. Decision Tree 개념

예제를 이용한 Decision Tree 동작 과정



1. Decision Tree 개념

예제를 이용한 Decision Tree 동작 과정

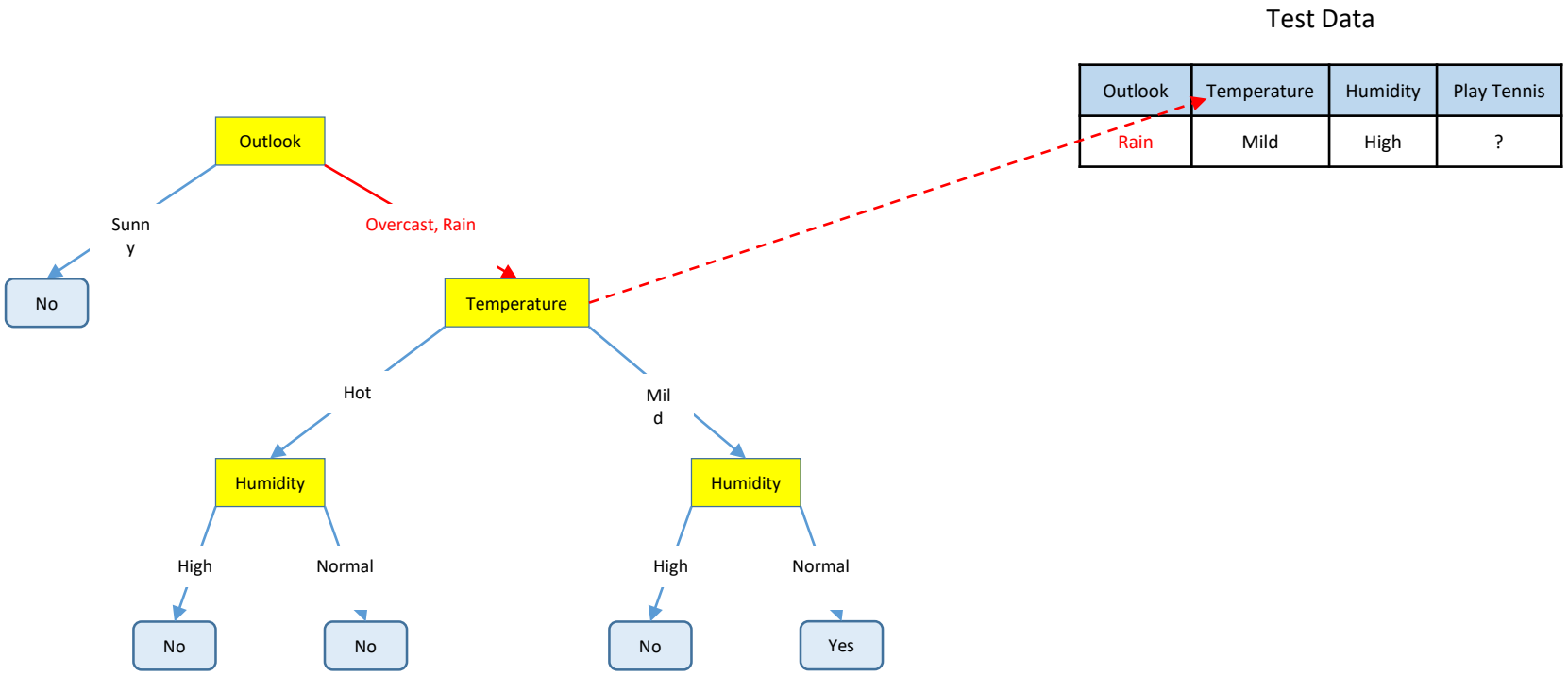


Test Data

Outlook	Temperature	Humidity	Play Tennis
Rain	Mild	High	?

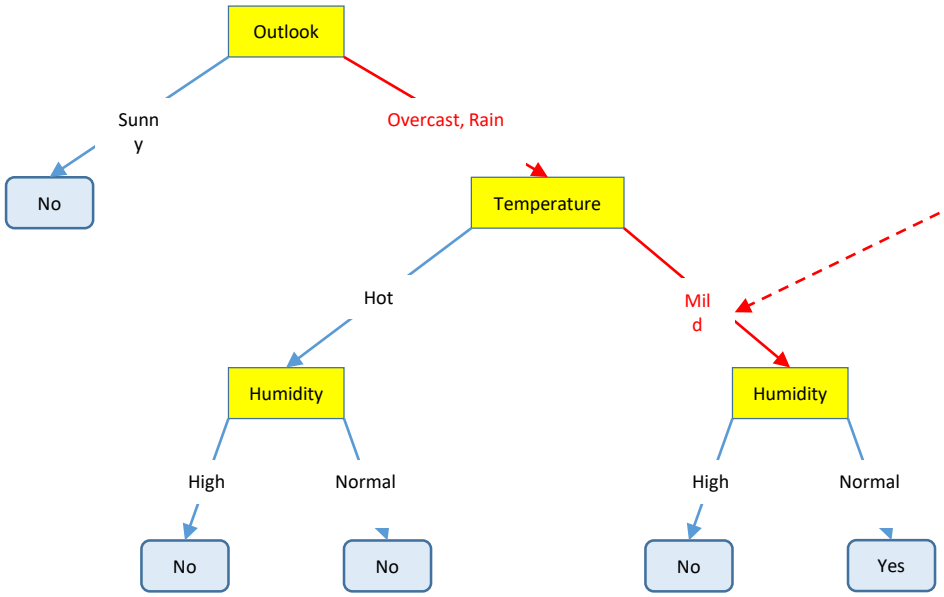
1. Decision Tree 개념

예제를 이용한 Decision Tree 동작 과정



1. Decision Tree 개념

예제를 이용한 Decision Tree 동작 과정

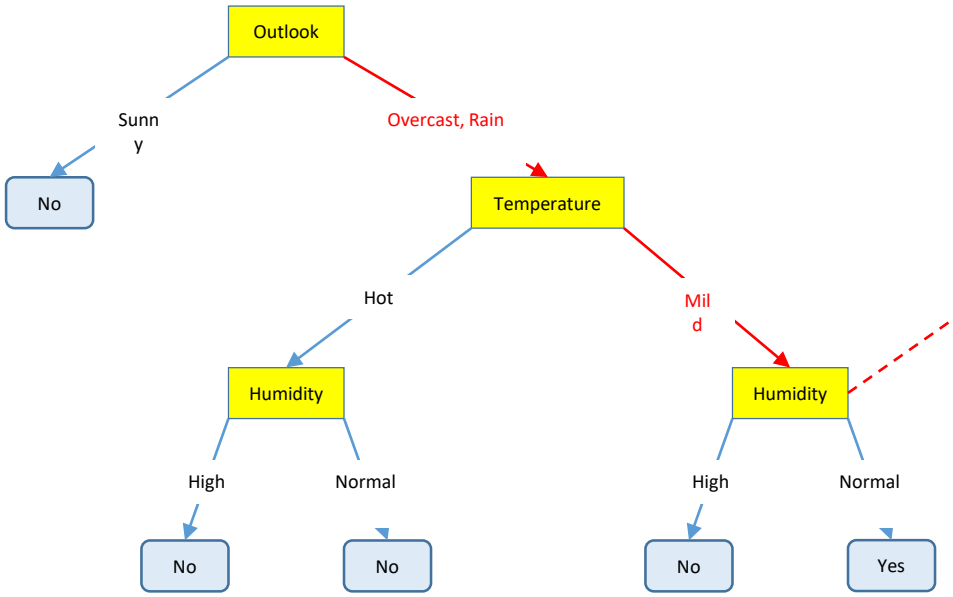


Test Data

Outlook	Temperature	Humidity	Play Tennis
Rain	Mild	High	?

1. Decision Tree 개념

예제를 이용한 Decision Tree 동작 과정

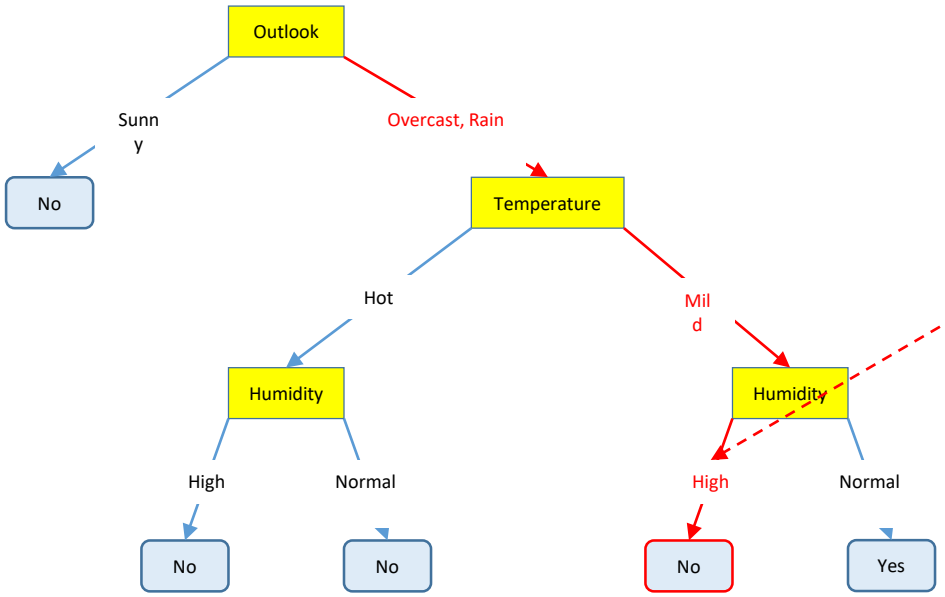


Test Data

Outlook	Temperature	Humidity	Play Tennis
Rain	Mild	High	?

2. Decision Tree 개념

예제를 이용한 Decision Tree 동작 과정



Test Data

Outlook	Temperature	Humidity	Play Tennis
Rain	Mild	High	?

Test Data에서 알고자 했던
Play Tennis 클래스 의 결과는
“No”

1. Decision Tree 개념

Decision Tree 분리기준(Split Criterion)

- 부모 마디(노드)로부터 자식 마디(노드)들이 형성될 때, 생성된 자식 노드에 속하는 자료의 순수도(Purity)가 가장 크게 증가하도록 트리를 형성하며 진행

- 입력 변수를 이용해 목표 변수의 분포를 얼마나 잘 구별하는 정도를 파악해 자식 마디가 형성되는데, 목표 변수의 구별 정도를 불순도(Impurity, 다양한 범주들의 개체들이 포함되어 있는 정도)에 의해 측정

1. Decision Tree 개념

Decision Tree 분리기준(Split Criterion)

지니 지수(Gini Index)

- 데이터 집합의 불순도를 측정
- 지니 지수는 0~1 사이의 값을 가지며, 어떤 데이터 집합에 속한 개체(레코드)들이 같은 범주(클래스)로 구성되어 있으면 지니 지수는 최솟값이 0을 갖고 해당 데이터 집합은 순수하다고 볼 수 있음
- 즉, 지니 지수가 작을수록 잘 분류된 것으로 볼 수 있음

엔트로피 지수(Entropy Index)

- 엔트로피는 주어진 데이터 집합의 혼잡도를 의미
- 주어진 데이터 집합에 서로 다른 범주(클래스)의 개체(레코드)들이 많이 섞여 있으면 엔트로피가 높고, 같은 범주의 개체들이 많이 있으면 엔트로피가 낮음
- 엔트로피 지수는 0~1 사이의 값을 가지며, 가장 혼잡도가 높은 상태(서로 다른 범주의 개체들이 섞여 있는 상태)는 1, 혼잡도가 가장 낮은 상태(하나의 범주의 개체로 구성된 상태)는 0
- 엔트로피 지수가 작을수록 잘 분류된 것으로 볼 수 있음

정보 이득(Information Gain)

- 상위 노드의 엔트로피 지수에서 하위 노드의 가중평균한 엔트로피 지수를 뺀 것을 의미
- 즉, 원래 상위 노드의 엔트로피를 구하고 어떤 속성을 선택한 후의 x개의 하위 노드로 분리된 것에 대한 가중평균한 엔트로피를 구한 값의 차를 의미(계산되어 나온 값이 클 수록 정보 이득이 큰 것을 의미, 선택한 어떤 속성이 분류하기 좋다고 볼 수 있음)

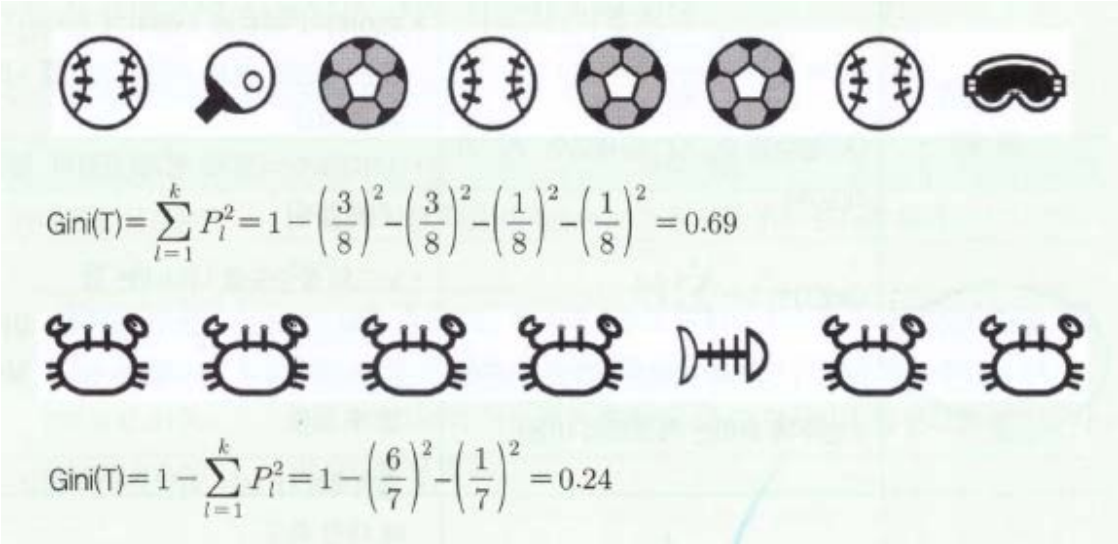
1. Decision Tree 개념

지니 지수(Gini Index) 계산 식

- $G = 1 - \sum_{j=1}^{c} P(j)^2 = 1 - \sum_{j=1}^{c} \left(\frac{n_j}{n}\right)^2$

  - C : 범주(클래스)의 수
  - P(j) : j번째 범주(클래스)에 분류될 확률
  - n : 노드에 속하는 개체 수
  - n_j : 노드에 속하는 수 중 j번째 범주(클래스)에 속하는 개체 수

  - 지니계수의 개념(참고: 빅데이터 데이터 분석(정보기술사연구회, 건기원)



2. 예제를 이용한 Decision Tree 실습

예제를 이용한 Decision Tree 지니 지수(Gini Index) 계산

Example Data

Outlook	Temperature	Humidity	Wind	PlayTennis
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

속성별로 지니 지수를 계산하는 경우

1. Outlook
2. Humidity
3. Wind

[참고] https://www.es.utoronto.ca/wp-content/uploads/2019/03/Lecture_09.pdf

2. 예제를 이용한 Decision Tree 실습

예제를 이용한 Decision Tree 지니 지수(Gini Index) 계산

Humidity 속성으로 분리하는 경우

	Class = N	Class = Y	
Left	4	3	7
Right	1	6	7
	5	9	14

- Left = {High}, Right = {Normal} 일 때
  - Left (High)의 지니 지수 계산

$$1 - \left\{\left(\frac{3}{7}\right)^2 + \left(\frac{4}{7}\right)^2\right\}$$
$$= 0.4698$$

  - Right (Normal)의 지니 지수 계산

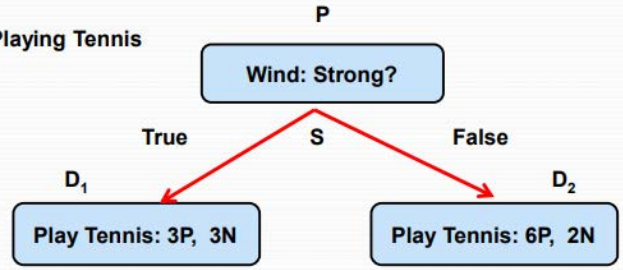
$$1 - \left\{\left(\frac{6}{7}\right)^2 + \left(\frac{1}{7}\right)^2\right\}$$
$$= 0.2449$$

  - (Humidity) 지니 지수 계산

$$\left\{0.4898X\left(\frac{7}{14}\right) + 0.2449X\left(\frac{7}{14}\right)\right\}$$
$$= 0.3674$$

Gini Index (Gini Impurity)

Example: Playing Tennis



gini (Play Tennis | Wind =True) = $1 - \left(\frac{3}{6}\right)^2 - \left(\frac{3}{6}\right)^2 = 0.5$

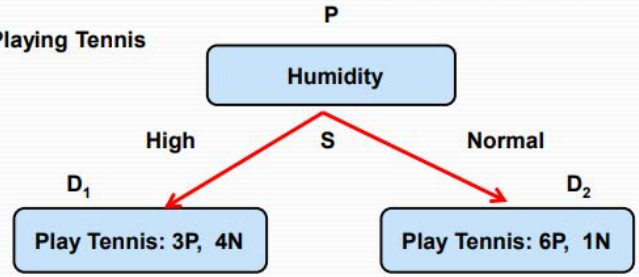
gini (Play Tennis | Wind = False) = $1 - \left(\frac{6}{8}\right)^2 - \left(\frac{2}{8}\right)^2 = 0.375$

$gini_S(D) = \frac{D_1}{D} gini(D_1) + \frac{D_2}{D} gini(D_2)$

$= \frac{6}{14} \times 0.5 + \frac{8}{14} \times 0.375 = 0.4286$

Gini Index (Gini Impurity)

Example: Playing Tennis



gini (Play Tennis | Humidity = High) = $1 - \left(\frac{3}{7}\right)^2 - \left(\frac{4}{7}\right)^2 = 0.4898$

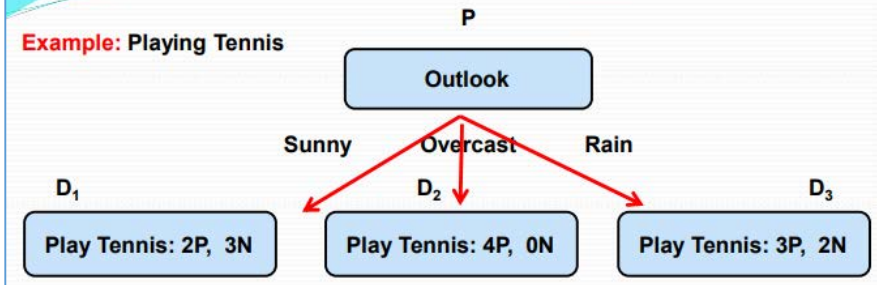
gini (Play Tennis | Humidity = Normal) = $1 - \left(\frac{6}{7}\right)^2 - \left(\frac{1}{7}\right)^2 = 0.2449$

$gini_S(D) = \frac{D_1}{D} gini(D_1) + \frac{D_2}{D} gini(D_2)$

$= \frac{6}{14} \times 0.4898 + \frac{8}{14} \times 0.2449 = 0.3674$

Gini Index (Gini Impurity)

Example: Playing Tennis



gini (Play Tennis | Outlook = Sunny) = $1 - \left(\frac{2}{5}\right)^2 - \left(\frac{3}{5}\right)^2 = 0.48$

gini (Play Tennis | Outlook = Overcast) = $1 - \left(\frac{4}{4}\right)^2 - \left(\frac{0}{4}\right)^2 = 0$

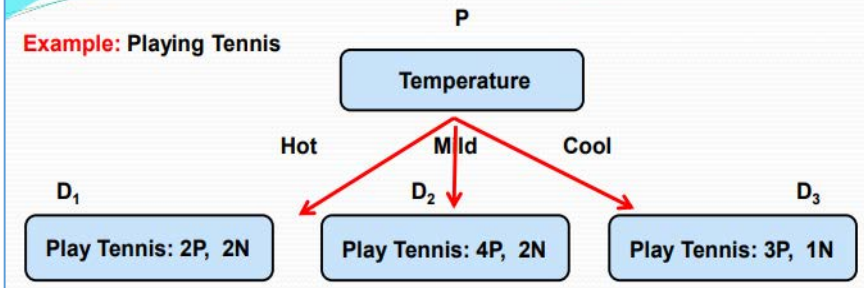
gini (Play Tennis | Outlook = Rain) = $1 - \left(\frac{3}{5}\right)^2 - \left(\frac{3}{5}\right)^2 = 0.48$

$gini_S(D) = \frac{D_1}{D} gini(D_1) + \frac{D_2}{D} gini(D_2) + \frac{D_3}{D} gini(D_3)$

$= \frac{5}{14} \times 0.48 + \frac{4}{14} \times 0 + \frac{5}{14} \times 0.48 = 0.3429$

Gini Index (Gini Impurity)

Example: Playing Tennis



gini (Play Tennis | Temperature = Hot) = $1 - \left(\frac{2}{4}\right)^2 - \left(\frac{3}{4}\right)^2 = 0.5$

gini (Play Tennis | Temperature = Mild) = $1 - \left(\frac{4}{6}\right)^2 - \left(\frac{2}{6}\right)^2 = 0.4444$

gini (Play Tennis | Temperature = Cool) = $1 - \left(\frac{3}{4}\right)^2 - \left(\frac{1}{4}\right)^2 = 0.375$

$gini_S(D) = \frac{D_1}{D} gini(D_1) + \frac{D_2}{D} gini(D_2) + \frac{D_3}{D} gini(D_3)$

$= \frac{4}{14} \times 0.5 + \frac{6}{14} \times 0.4444 + \frac{4}{14} \times 0.375 = 0.4405$

2/4

Gini 값이 가장 큰 Temperature를 먼저 분리 속성으로 정한다.

2. 예제를 이용한 Decision Tree 실습

예제를 이용한 Decision Tree 파이썬 코드 실습

```
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree

from IPython.display import Image

import pandas as pd
import numpy as np
import pydotplus
import os
```

- Scikit-learn (sklearn)
  - Python 프로그래밍 언어용 기계학습(Machine Learning) 관련 라이브러리(Library)

- 패키지 설명
  1. sklearn.metrics : scikit-learn 패키지 중 모형평가에 사용되는 서브 패키지
     classification_report : 주요 분류 측정 항목을 보여주는 보고서 모듈
     confusion_matrix : 분류의 정확성을 평가하기 위한 오차행렬 계산 모듈

2. 예제를 이용한 Decision Tree 실습

예제를 이용한 Decision Tree 파이썬 코드 실습

```
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree

from IPython.display import Image

import pandas as pd
import numpy as np
import pydotplus
import os
```

- 패키지 설명

  2. sklearn.model_selection : scikit-learn 패키지 중 클래스를 나눌 때, 함수를 통해 train/test를 나눌 때, 모델 검증에 사용되는 서브 패키지 train_test_split : 배열 또는 행렬을 임의의 훈련(train) 및 테스트(test) 하위 집합으로 분할하는 모듈

  3. sklearn.tree : scikit-learn 패키지 중 분류(Classification) 및 회귀(Regression)를 위한 의사결정 트리 기반 모델이 있는 서브 패키지 DecisionTreeClassifier : 의사결정 트리 분류 모듈

2. 예제를 이용한 Decision Tree 실습

예제를 이용한 Decision Tree 파이썬 코드 실습

```
1  from sklearn.metrics import classification_report, confusion_matrix
2  from sklearn.model_selection import train_test_split
3  from sklearn.tree import DecisionTreeClassifier
4  from sklearn import tree
5
6  from IPython.display import Image
7
8  import pandas as pd
9  import numpy as np
10 import pydotplus
11 import os
```

- 패키지 설명
  5. sklearn : 기계학습(Machine Learning) 관련 패키지
     tree : 분류(Classification) 및 회귀(Regression)를 위한 의사결정 트리 기반 모듈

  6. IPython.display : IPython 내에 정보를 보여주는 도구용 공용 API
     Image : raw 데이터가 있는 PNG, JPEG 이미지 객체를 만드는 모듈

- 패키지 설명
  8. pandas : 데이터를 구조화된 형식으로 가공 및 분석할 수 있는 자료구조를 제공하는 패키지
     as pd : pandas를 약칭 pd로 사용한다는 의미

  9. numpy : Numerical Python의 줄임말로 고성능 계산, 데이터 분석에 관련된 패키지
     as np : numpy를 약칭 np로 사용한다는 의미

2. 예제를 이용한 Decision Tree 실습

예제를 이용한 Decision Tree 파이썬 코드 실습

```
1  from sklearn.metrics import classification_report, confusion_matrix
2  from sklearn.model_selection import train_test_split
3  from sklearn.tree import DecisionTreeClassifier
4  from sklearn import tree
5
6  from IPython.display import Image
7
8  import pandas as pd
9  import numpy as np
10 import pydotplus
11 import os
```

- 패키지 설명

  10. pydotplus : 그래프를 생성하는 graphviz의 Dot 언어를 파이썬 인터페이스에 제공하는 모듈

  11. os : 운영체제(Operating System)와 상호작용하기 위한 기본적인 기능(경로 생성, 변경 등)이 제공되는 모듈

- pydotplus 설치

  anaconda prompt 를 관리자 권한으로 실행

  conda install -c conda-forge pydotplus

2. 예제를 이용한 Decision Tree 실습

예제를 이용한 Decision Tree 파이썬 코드 실습

```
1 tennis_data = pd.read_csv('playtennis.csv')
2 tennis_data
```

- 코드 설명
  1. 예제 데이터인 playtennis의 데이터가 저장되어 있는 csv 파일을 판다스의 read_csv함수를 사용해 로드(load)하고 판다스의 자료구조 중 하나인 데이터프레임 형식으로 변수 tennis_data에 저장

  2. 저장된 변수 tennis_data 확인

	Outlook	Temperature	Humidity	Wind	PlayTennis
0	Sunny	Hot	High	Weak	No
1	Sunny	Hot	High	Strong	No
2	Overcast	Hot	High	Weak	Yes
3	Rain	Mild	High	Weak	Yes
4	Rain	Cool	Normal	Weak	Yes
5	Rain	Cool	Normal	Strong	No
6	Overcast	Cool	Normal	Strong	Yes
7	Sunny	Mild	High	Weak	No
8	Sunny	Cool	Normal	Weak	Yes
9	Rain	Mild	Normal	Weak	Yes
10	Sunny	Mild	Normal	Strong	Yes
11	Overcast	Mild	High	Strong	Yes
12	Overcast	Hot	Normal	Weak	Yes
13	Rain	Mild	High	Strong	No

2. 예제를 이용한 Decision Tree 실습

예제를 이용한 Decision Tree 파이썬 코드 실습

```
1  tennis_data.Outlook = tennis_data.Outlook.replace('Sunny', 0)
2  tennis_data.Outlook = tennis_data.Outlook.replace('Overcast', 1)
3  tennis_data.Outlook = tennis_data.Outlook.replace('Rain', 2)
4
5  tennis_data.Temperature = tennis_data.Temperature.replace('Hot', 3)
6  tennis_data.Temperature = tennis_data.Temperature.replace('Mild', 4)
7  tennis_data.Temperature = tennis_data.Temperature.replace('Cool', 5)
8
9  tennis_data.Humidity = tennis_data.Humidity.replace('High', 6)
10 tennis_data.Humidity = tennis_data.Humidity.replace('Normal', 7)
11
12 tennis_data.Wind = tennis_data.Wind.replace('Weak', 8)
13 tennis_data.Wind = tennis_data.Wind.replace('Strong', 9)
14
15 tennis_data.PlayTennis = tennis_data.PlayTennis.replace('No', 10)
16 tennis_data.PlayTennis = tennis_data.PlayTennis.replace('Yes', 11)
17
18 tennis_data
```

	Outlook	Temperature	Humidity	Wind	PlayTennis
0	0	3	6	8	10
1	0	3	6	9	10
2	1	3	6	8	11
3	2	4	6	8	11
4	2	5	7	8	11
5	2	5	7	9	10
6	1	5	7	9	11
7	0	4	6	8	10
8	0	5	7	8	11
9	2	4	7	8	11
10	0	4	7	9	11
11	1	4	6	9	11
12	1	3	7	8	11
13	2	4	6	9	10

- 코드 설명

  1 ~ 16. 변수 tennis_data의 각 컬럼(Outlook, Temperature, ...)의 값(Sunny, Overcast, ...)을 문자열(String) 타입에서 숫자(int) 타입으로 대치(replace)해 변수 tennis_data의 각 컬럼별 저장
  * 의사결정 트리 분류 모델에 train, test 데이터 값으로 사용하기 위한 전처리 과정

  18. 전처리 과정이 된 변수 tennis_data 확인

2. 예제를 이용한 Decision Tree 실습

예제를 이용한 Decision Tree 파이썬 코드 실습

```
1 X = np.array(pd.DataFrame(tennis_data, columns = ['Outlook', 'Temperature', 'Humidity', 'Wind']))
2 y = np.array(pd.DataFrame(tennis_data, columns = ['PlayTennis']))
```

- 코드 설명

  1. 변수 tennis_data의 컬럼(Outlook, Temperature, Humidity, Wind)의 값들을 데이터프레임 형태로 추출하고 np.array를 이용해 추출한 데이터를 배열 형태로 변환한 후 변수 X에 저장

  2. 변수 tennis_data의 컬럼(PlayTennis)의 값을 데이터프레임 형태로 추출하고 np.array를 이용해 추출한 데이터를 배열 형태로 변환한 후 변수 y에 저장

X

```
array([[0, 3, 6, 8],
       [0, 3, 6, 9],
       [1, 3, 6, 8],
       [2, 4, 6, 8],
       [2, 5, 7, 8],
       [2, 5, 7, 9],
       [1, 5, 7, 9],
       [0, 4, 6, 8],
       [0, 5, 7, 8],
       [2, 4, 7, 8],
       [0, 4, 7, 9],
       [1, 4, 6, 9],
       [1, 3, 7, 8],
       [2, 4, 6, 9]], dtype=int64)
```

y

```
array([[10],
       [10],
       [11],
       [11],
       [11],
       [10],
       [11],
       [10],
       [11],
       [11],
       [11],
       [11],
       [11],
       [10]], dtype=int64)
```

2. 예제를 이용한 Decision Tree 실습

예제를 이용한 Decision Tree 파이썬 코드 실습

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y)
```

- 코드 설명

  1. 로드(load)된 train_test_split 모듈을 이용해 변수 X에 입력한 4개 컬럼에 대한 데이터와 변수 y에 입력한 Playtennis 컬럼의 데이터를 train(훈련)과 test(테스트)로 구분해, 임의의 개수로 각각 변수 X_train, X_test, y_train, y_test에 저장

  *일반적으로 train / test의 비율 = train(7.5) : test(2.5)

2. 예제를 이용한 Decision Tree 실습

예제를 이용한 Decision Tree 파이썬 코드 실습

```
1 dt_clf = DecisionTreeClassifier()
2 dt_clf = dt_clf.fit(X_train, y_train)
```

- 코드 설명
  1. 로드(load)된 의사결정 트리 분류 모듈을 변수 dt_clf에 저장
  2. 의사결정 트리 분류 모듈이 저장된 변수 dt_clf의 함수 fit()에 변수 X_train, y_train을 입력해 의사결정 트리 분류 모델 생성, 생성한 모델을 다시 변수 dt_clf에 저장

```
1 dt_prediction = dt_clf.predict(X_test)
```

- 코드 설명
  1. 변수 dt_clf의 함수 predict()에 변수 X_test를 입력
     입력한 X_test에 대한 클래스 예측 값을 변수 dt_prediction에 저장

2. 예제를 이용한 Decision Tree 실습

예제를 이용한 Decision Tree 파이썬 코드 실습

```
1 print(confusion_matrix(y_test, dt_prediction))
```

```
[[1 1]
 [0 2]]
```

- 코드 설명
  1. 오차행렬을 계산하는 모듈 confusion_matrix()에 변수 y_test와 dt_prediction을 입력
     입력한 두 변수(y_test, dt_prediction)의 오차행렬을 print 문으로 출력

2. 예제를 이용한 Decision Tree 실습

예제를 이용한 Decision Tree 파이썬 코드 실습

```
1 print(classification_report(y_test, dt_prediction))
```

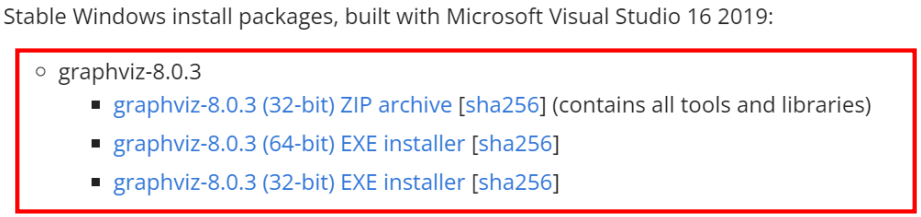
```
             precision    recall  f1-score   support

         10       1.00      0.50      0.67         2
         11       0.67      1.00      0.80         2

avg / total       0.83      0.75      0.73         4
```

- 코드 설명
  1. 분류 측정 항목을 보여주는 모듈인 classification_report()에 변수 y_test와 dt_prediction을 입력, 입력한 두 변수(y_test, dt_prediction)에 대한 분류 측정 항목을 print문으로 출력

2. 예제를 이용한 Decision Tree 실습

예제를 이용한 Decision Tree 파이썬 코드 실습

- 코드를 실행하기 전 IPython 내에서 그래프 표현을 위해 graphviz(Graph Visualization Software)를 다운로드 및 설치
- 다운로드 : http://www.graphviz.org/Download_windows.php (작업 환경이 Windows로 가정했을 시)



- Graphviz-8.0.3 파일을 다운로드
- 다운로드 한 graphviz-2.38.msi 파일을 설치 실행, 설치 진행 시 PATH 경로 추가 옵션 선택

- Graphviz-2.38을 설치할 경로 선택 부분에서 기본적으로 나오는 경로 선택
  - 설치될 기본 경로 : C:₩Program Files (x86) 아래에 위치

2. 예제를 이용한 Decision Tree 실습

예제를 이용한 Decision Tree 파이썬 코드 실습

```
1 os.environ["PATH"] += os.pathsep + 'C:/Program Files (x86)/Graphviz2.38/bin/'
```

- 코드 설명

  IPython 내에서 그래프를 생성할 수 있는 인터페이스 경로 추가 설정하는 부분

  1. Graphviz2.38이 설치되고 bin 폴더가 있는 경로인 'C:₩Program Files(x86)/Graphviz2.38/bin/'를 os 모듈 중 경로 구분 기호를 반환하는 함수인 os.pathsep을 이용해, 환경변수들을 나타내는 사전 함수인 os.environ['PATH']에 동적으로 할당해 저장

2. 예제를 이용한 Decision Tree 실습

예제를 이용한 Decision Tree 파이썬 코드 실습

```
1 feature_names = tennis_data.columns.tolist()
2 feature_names = feature_names[0:4]
```

- 코드 설명
  1. 트리 표현 함수(tree.export_graphviz())에 입력되는 파라미터 중 하나인 feature_names에 값을 입력하기 위해, 변수 tennis_data의 각 컬럼을 list형태로 변수 feature_names에 저장

  2. 저장된 변수 feature_names를 슬라이싱(0:4)해 Outlook, Temperature, Humidity, Wind의 컬럼을 추출해 다시 변수 feature_names에 저장

```
1 target_name = np.array(['Play No', 'Play Yes'])
```

- 코드 설명
  1. 트리 표현 함수(tree.export_graphviz())에 입력되는 파라미터 중 하나인 class_names에 값을 입력하기 위해, target class 값인 'Play No'와 'Play Yes'를 배열형태로 변수 target_name에 저장

2. 예제를 이용한 Decision Tree 실습

예제를 이용한 Decision Tree 파이썬 코드 실습

```
dt_dot_data = tree.export_graphviz(dt_clf, out_file = None,
                                   feature_names = feature_names,
                                   class_names = target_name,
                                   filled = True, rounded = True,
                                   special_characters = True)
```

- 코드 설명

  1. tree 패키지 중 의사결정 트리를 dot형식으로 내보내는 함수인 export_graphviz()를 이용해 트리 표현을 변수 dt_dot_data에 저장

  함수 export_graphviz에 들어가는 파라미터로는

  - dt_clf : 의사결정 트리 분류기(graphviz로 내보낼 의사결정 트리)
  - out_file : 의사결정 트리를 파일 또는 문자열로 반환(기본 : tree.dot, None일 경우 문자열로 반환)
  - feature_names : 각 features의 이름(문자열)
  - class_names : 각 대상 class의 이름을 오름차순으로 정렬 (True일 경우 class 이름의 symbol 표현)
  - filled : True 일 경우 분류를 위한 다수 클래스, 회귀 값의 극한 또는 다중 출력의 노드 순도를 나타내기 위해 노드를 색칠
  - rounded : True 일 경우 둥근 모서리가 있는 노드 상자를 그리고, Times-Roman 대신 Helvetica 글꼴 사용
  - special_characters : True 일 경우 특수 문자 표시

2. 예제를 이용한 Decision Tree 실습

예제를 이용한 Decision Tree 파이썬 코드 실습

```
1 dt_graph = pydotplus.graph_from_dot_data(dt_dot_data)
```

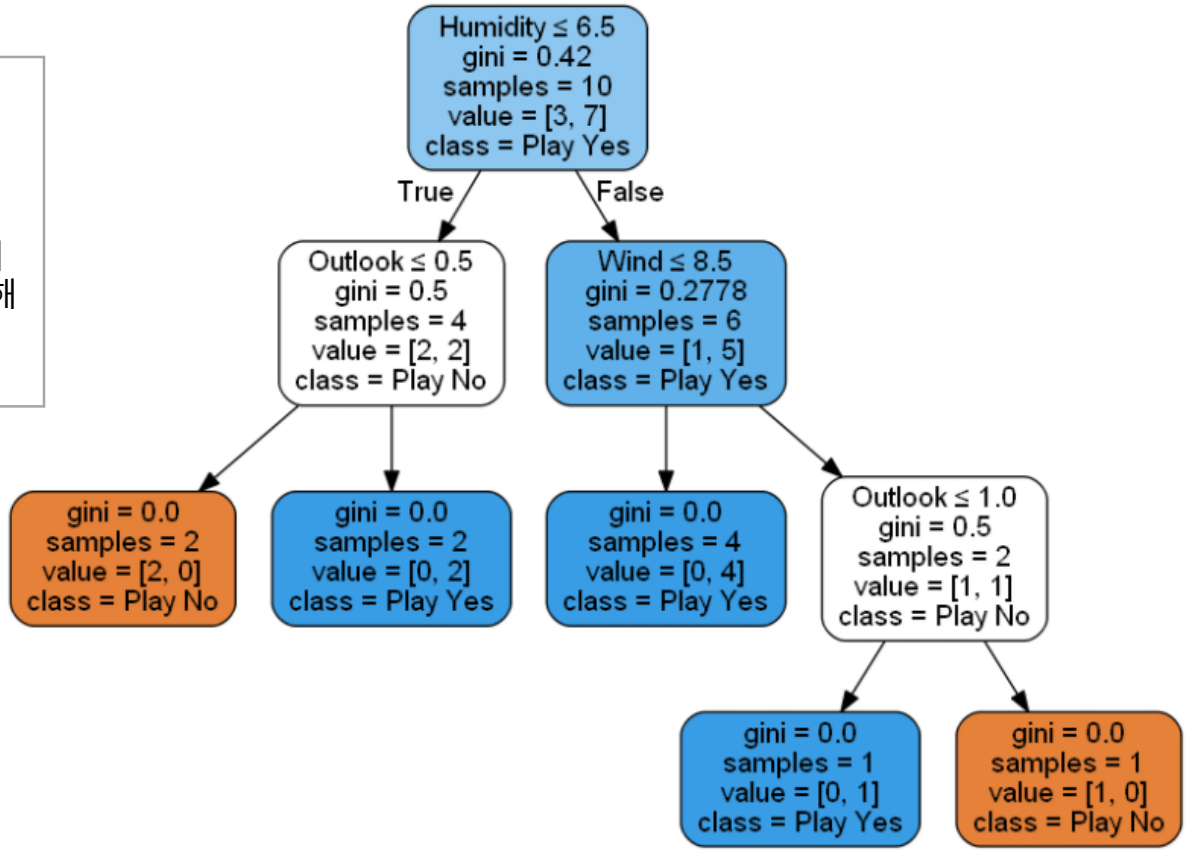
- 코드 설명
  1. Pydotplus 모듈 중 Dot 형식의 데이터로 정의된 그래프를 로드(load)하는 함수인 graph_from_dot_data()에 변수 dt_dot_data를 입력한 후 변수 dt_graph에 저장

2. 예제를 이용한 Decision Tree 실습

예제를 이용한 Decision Tree 파이썬 코드 실습

```
1 Image(dt_graph.create_png())
```

- 코드 설명
  1. 변수 dt_graph에 대한 정보를 png파일로 생성하는 함수 create_png()를 사용한 후, 이미지 객체를 만드는 Image 모듈을 통해 그래프 표현



4.2 Random Forest를 이용한 데이터분석

앙상블(ensemble) learning

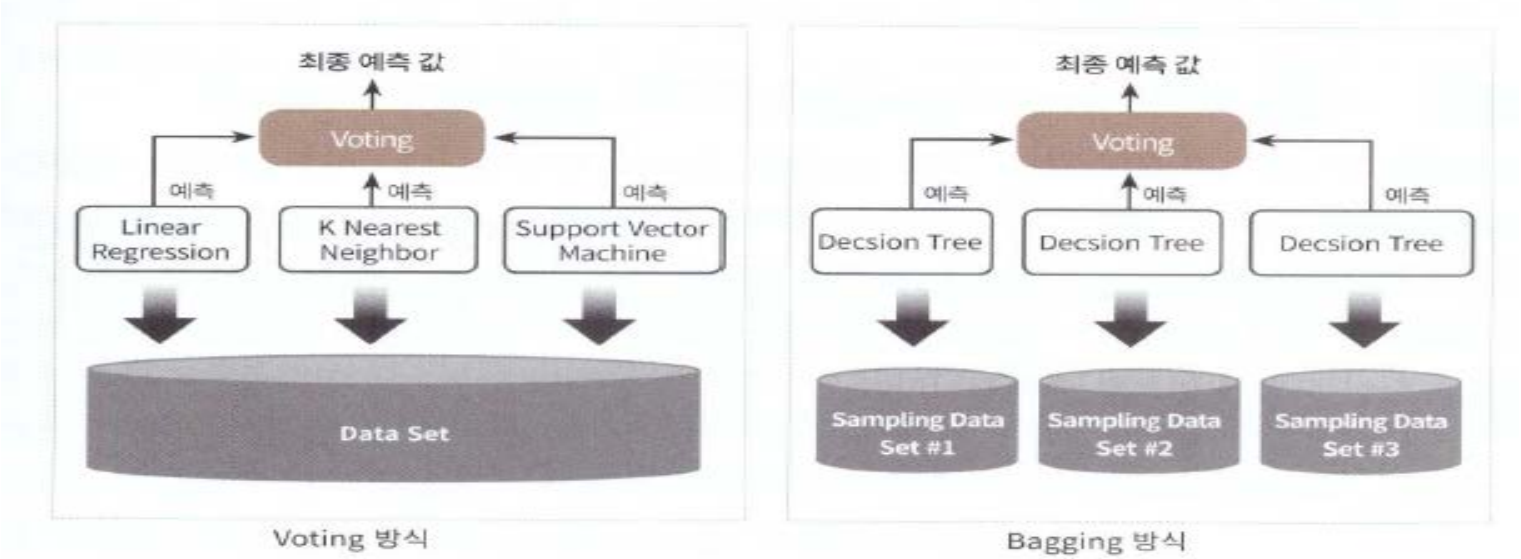
- 앙상블 학습 : 여러 개의 분류기를 생성하고 그 예측을 결합하여 더 좋은 분류기를 만드는 방법
- 앙상블학습의 종류

  Voting : 다른 알고리즘 분류기를 결합

  Bagging : 같은 분류기를 중복샘플링(**B**ootstrap **Agg**rating)여 결과를 결합(예 : 랜덤포레스트)

  bootstrap : 중복을 허용하여 샘플링, aggregating : 모델의 결과를 다수결로 합하는 방법

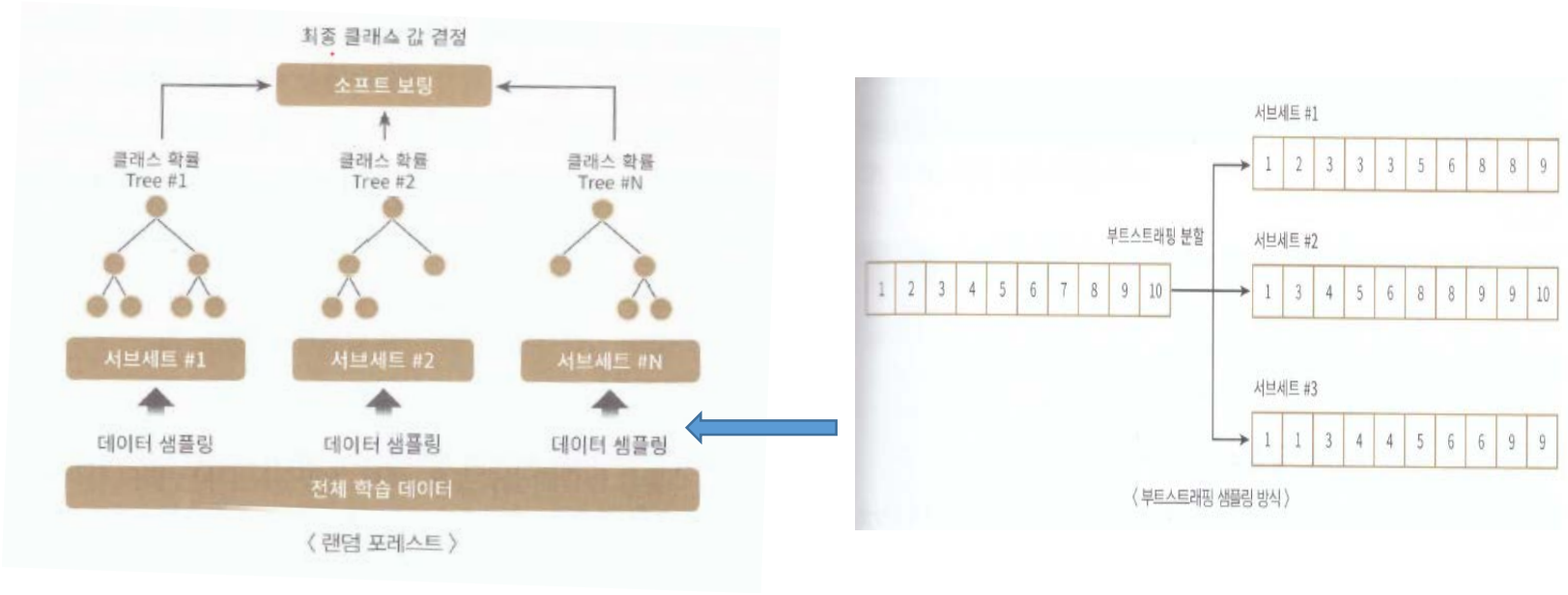
  Boosting : 분류기를 순차적으로 결합하되, 가중치를 부여하면서 진행(예 : 그래디언트 부스트, XGBoost)



[참고] 파이썬 머신러닝 완벽가이드, 위키북스

1. Random Forest 소개

- Random Forest는 2001년에 Leo Breiman에 의해 처음으로 소개
  Leo Breiman(https://newsarchive.berkeley.edu/news/media/releases/2005/07/07_breiman.shtml)
- Random Forest는 훌륭한 데이터 분석 알고리즘 중 하나
  - 데이터 분류(classification), 데이터 군집(clustering), Feature의 중요성 확인, 데이터 예측
  - 소프트 보팅 : 각 분류기의 결정 확률을 결합



랜덤포레스트 : https://bioinformaticsandme.tistory.com/167

1. Random Forest 이론

- Random Forest 예측 모듈

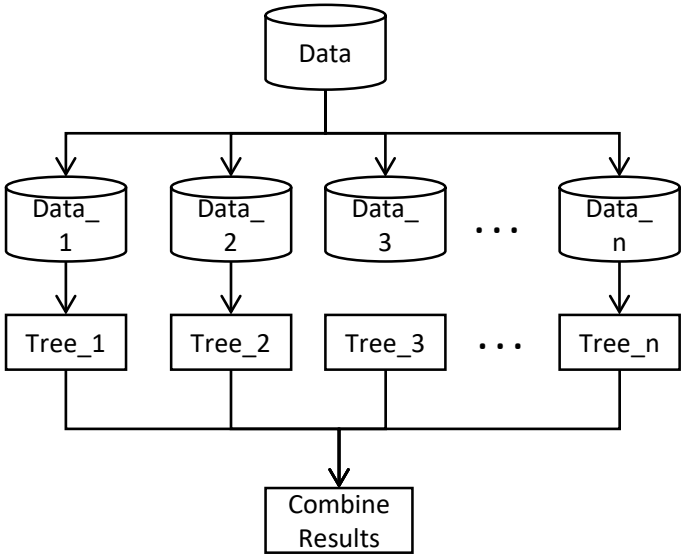
  1) Dataset에서 샘플 데이터를 선택 (bagging/bootstrap aggregation)

  2) 샘플 데이터를 이용해 Decision Tree를 생성

  3) 과정 1), 2)를 n번 반복

  4) 과정 3)을 통해 생성한 n개의 Decision Tree 를 이용해 예측

  5) 예측 결과에서 가장 많이 등장하는 결과를 선택하여 최종 결과로 선택



1. Random Forest 이론

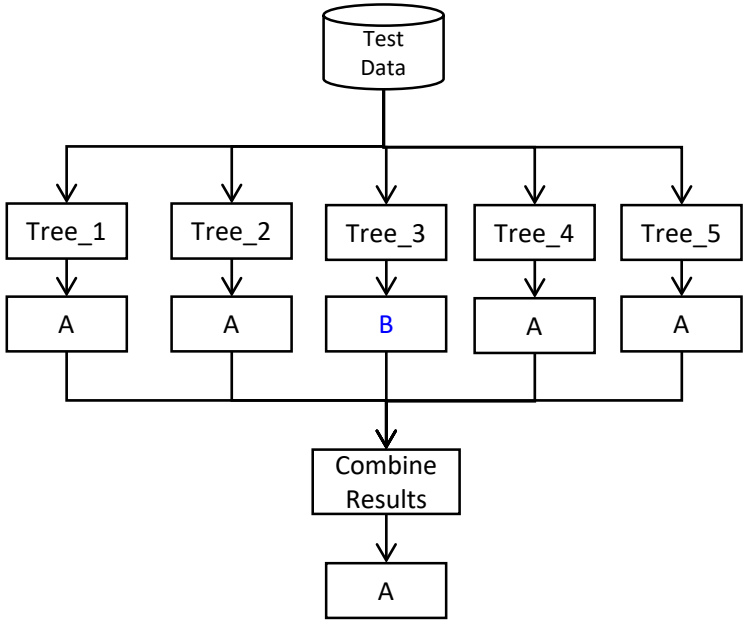
- Random Forest 예측 예:

  1) Test Data의 target은 A or B

  2) 그림은 Test Data의 target을 예측

  3) 4개의 tree에서의 예측 결과가 'A'
     - (A=) > (B=1)

  4) 최종 결과 Test Data를 A로 예측



1. Random Forest 이론

- Random Forest 특징

  1) 여러 개의 Decision Tree를 결합함으로 단일 Decision Tree의 결점을 극복

  2) Over-fitting 문제가 적음

  3) 구현이 간단함

  4) 병렬 계산이 간편함

1. Random Forest 이론

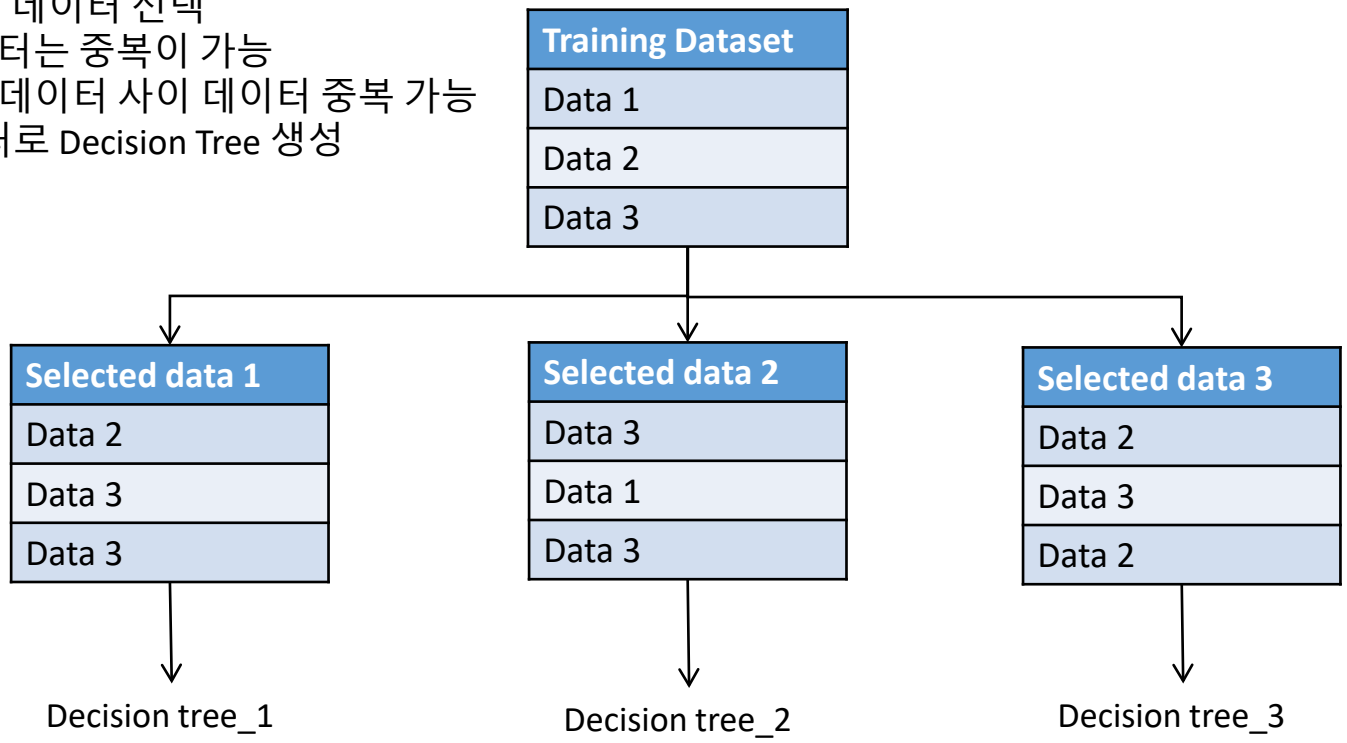
- Random Forest에서의 2가지 random

  1) Dataset에서 샘플 데이터를 random으로 선택

  2) 샘플 데이터에서 feature를 random으로 선택해 decision tree를 생성

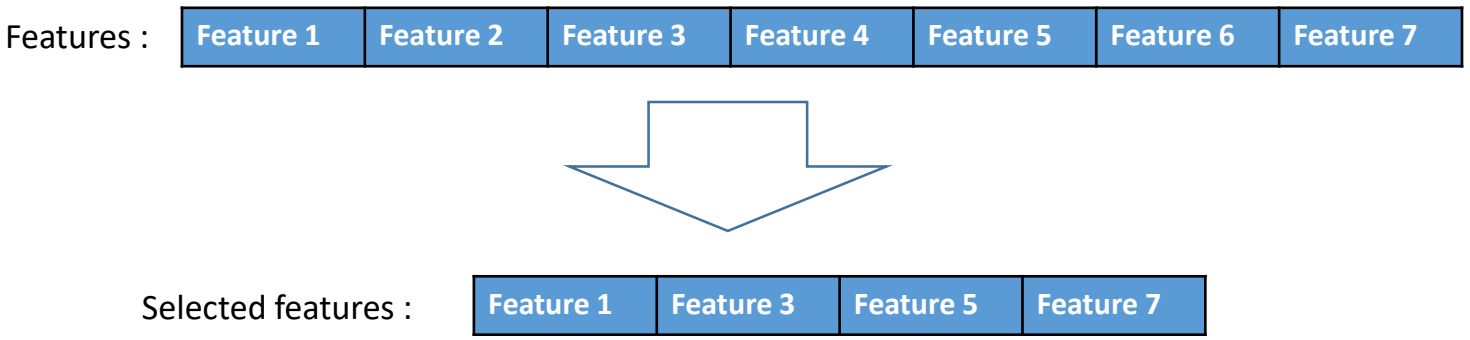
1. Random Forest 이론

- 샘플 데이터 선택 예:
  - Random으로 n개의 데이터 선택
  - 선택한 n개의 데이터는 중복이 가능
  - 선택한 t개의 샘플 데이터 사이 데이터 중복 가능
  - 선택한 샘플 데이터로 Decision Tree 생성



1. Random Forest 이론

- 선택한 샘플 데이터에서 random으로 f개 feature를 선택
  - 선택하는 feature의 개수는 Sqrt(전체 feature 수), Log2(전체 feature 수)등 방법으로 계산



1. Random Forest 이론

- **자기 성능 평가**

- Bagging
  - 약 2/3n개의 데이터를 이용해 매개 tree를 생성
  - 나머지 1/3n개데이터를 이용해 매개 tree의 성능을 평가
    - 매개 tree에 입력하는 데이터는 다름

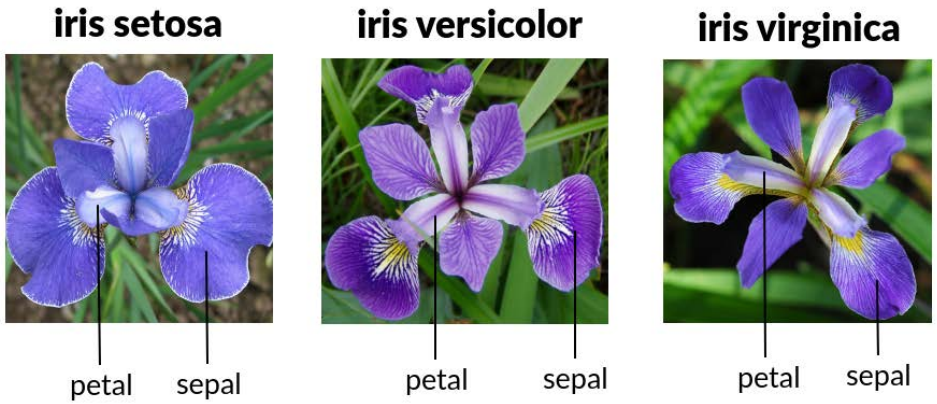
- Out-of-Bag(OOB)
  - OOB 데이터를 이용해 tree의 성능을 교정
  - OOB는 성능 통계에서 많이 사용됨

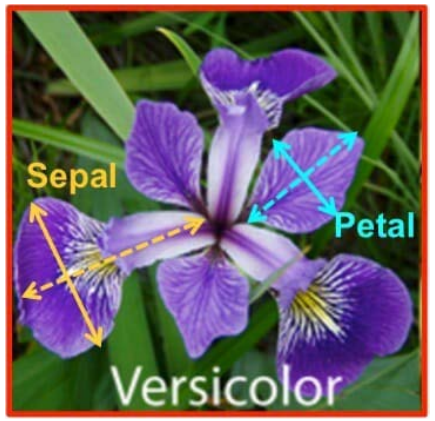
2. Iris 데이터를 이용해 간단한 Random Forest 구현


- Iris Data 소개(iris는 붓꽃이라고 부르며 그리스로 무지개를 뜻한다)

	sepal length (cm) 꽃받침	sepal width (cm) 꽃받침	petal length (cm) 꽃잎	petal width (cm) 꽃잎	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
…	…	…	…	…	…
148	6.2	3.4	5.4	2.3	2
149	5.9	3.0	5.1	1.8	2

- target:
  - 0 : setosa
  - 1 : versicolor
  - 2 : virginica





2. Iris 데이터를 이용해 간단한 Random Forest 구현

- 필요한 패키지 및 라이브러리 로드

▪ 패키지 설명
- numpy : 파이썬 언어를 위한 행렬, 벡터 등의 수학 계산을 위한 자료구조와 계산 함수를 제공하는 패키지
- pandas : 데이터 분석, 가공, 처리 등을 쉽게 하기 위한 자료구조와 처리 함수들을 제공하는 패키지

▪ 모듈 설명
- sklearn.metrics : scikit-learn 패키지 중 모듈 성능 평가 방법 모듈
- sklearn.datasets : scikit-learn 패키지 중 일반적인 참조 데이터 셋을 로드(load)하는 모듈

- 함수 설명
- load_iris : iris data 로드
- accuracy_score : 분석 결과의 accuracy를 측정

```
1 from sklearn.datasets import load_iris
2 from sklearn.metrics import accuracy_score
3 import numpy as np
4 import pandas as pd
```

2. Iris 데이터를 이용해 간단한 Random Forest 구현

- Training data와 test data 설정

```
#loading the iris dataset
iris = load_iris()

#training data 설정
x_train = iris.data[:-30]
y_train = iris.target[:-30]
#test data 설정
x_test = iris.data[-30:] # test feature data
y_test = iris.target[-30:] # test target data
```

데이터의 시작부터 끝에서 30번째 하나 전까지를 선택

iris target 데이터에서 마지막 30번째부터 끝까지 선택

- x_*은 feature data 를 의미하고 y_*은 target data를 의미
- 전체 iris data에서 처음부터 마지막 30번째 데이터까지를 training data로 설정
- 마지막 30개 데이터를 test 데이터로 설정
- iris.data[:-30]에서 ':' 앞에 수치가 없으면 처음 위치부터 임을 표시하고 -30은 뒤로 30번째 데이터까지를 표시함
- iris.data[-30:]에서 -30은 마지막 30번째 데이터를 의미하고 ':'뒤에 수치가 없으면 끝을 표시함

▪ 함수 설명
- load_iris(): iris data를 호출
- iris.data : iris data에서 feature 데이터를 호출
- iris.target : iris data에서 target 데이터를 호출

2. Iris 데이터를 이용해 간단한 Random Forest 구현

- Training data의 target 출력

```
1 print (y_train)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2]
```

- Test data의 target 출력

```
1 print (y_test)
```

```
[2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2]
```

– Test data의 target은 전부 2(virginica)

– Training data와 test data의 분리가 합리하지 않음

2. Iris 데이터를 이용해 간단한 Random Forest 구현

- **Random Forest 분류기 생성**

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

- Random Forest 분류기 생성을 위해 RandomForestClassifier 클래스를 import
- Sklearn.ensemble 모듈은 분류, 회귀 및 이상 탐지를 위한 ensemble-based 방법을 포함

2. Iris 데이터를 이용해 간단한 Random Forest 구현

- **Random Forest 분류기 생성**
  - 10개의 tree를 가진 random foreset 생성

```
#RandomForestClassifier libary를 import
from sklearn.ensemble import RandomForestClassifier
#tree 의 개수 Random Forest 분류 모들 생성
rfc = RandomForestClassifier(n_estimators=10)
rfc
```

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_split=1e-07, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            n_estimators=10, n_jobs=1, oob_score=False, random_state=None,
            verbose=0, warm_start=False)
```

▪ 클래스 설명
- RandomForestClassifier: Random Forest 분류기를 포함한 클래스

▪ 변수 설명
- n_estimators : Decision Tree의 개수 (default=10)
- max_features : 최대 고려하는 feature의 개수(default=auto)
- oob_score : out-of-bag(OOB) 사용여부 (default=False)
- 참조 : http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

2. Iris 데이터를 이용해 간단한 Random Forest 구현

- **Random Forest 분류기**

```
1 rfc.fit(x_train, y_train)
2 #Test data를 입력해 target data를 예측
3 prediction = rfc.predict(x_test)
4 #예측 결과 precision과 실제 test data의 target 을 비교
5 print (prediction==y_test)
```

```
[ True  True  True False  True  True False False  True False  True  True
  True False False  True  True  True False  True  True  True  True  True
  True  True  True  True  True  True]
```

- Line 1 : 훈련 데이터를 입력해 random forest 분류기를 학습 시킴
- Line 3 : 분류기에 테스트 데이터를 입력해 목표 값을 예측(분류)
- Line 5 : 분류(예측)한 목표 값과 실제 목표 값을 비교하고 출력

▪ 함수 설명
- rfc.fit() : 입력 데이터를 이용해 분류기 학습
- rfc.predict(): 입력 데이터의 분류 결과 예측

▪ 변수 설명
- x_train : 훈련 데이터 feature 값
- y_train : 훈련 데이터 target 값
- x_test : 테스트 데이터 feature 값
- y_test : 테스트 데이터 target 값
- prediction : 테스트 데이터의 target 예측 값

2. Iris 데이터를 이용해 간단한 Random Forest 구현

- **Random Forest 분류기 성능 평가 (1)**

```
#Random forest 정확도 측정
rfc.score(x_test, y_test)
```

```
0.76666666666666672
```

▪ 함수 설명
- rfc.score() : RandomForestClassifier 클래스 안에 있는 분류 결과의 정확도(Accuracy)를 계산하는 함수

▪ 변수 설명
- x_test : 테스트 데이터 feature 값
- y_test : 테스트 데이터 target 값

2. Iris 데이터를 이용해 간단한 Random Forest 구현

- **Random Forest 분류기 성능 평가 (2)**

▪ 모듈 설명
- sklearn.metrics : scikit-learn 패키지 중 모듈 성능 평가방법 모듈

▪ 함수 설명
- accuracy_score() : 분류 결과의 accuracy를 계산 (상세 설명 7장 참조)
- classification_report(): 분류 결과의 precision, recall을 계산 (상세 설명 7장 참조)

▪ 변수 설명
- y_test : 테스트 데이터 target 값
- prediction : 테스트 데이터의 target 예측 값

```
1 from sklearn.metrics import accuracy_score
2 from sklearn.metrics import classification_report
3
4
5 print ("Accuracy is : ",accuracy_score(prediction, y_test))
6 print ("=============================================================")
7 print (classification_report(prediction, y_test))
```

```
Accuracy is :  0.766666666667
=============================================================
             precision    recall  f1-score   support

          1       0.00      0.00      0.00         7
          2       0.77      1.00      0.87        23

avg / total       0.59      0.77      0.67        30
```

❖ Training data와 test data를 잘 분리 하지 못한 이유로 분류 성능이 낮음

3. Random Forest 성능 제고 방법

- **Training, Test 데이터 재 생성**

```
1 from sklearn.model_selection import train_test_split
2 x = iris.data
3 y = iris.target
4 X_train, X_test, Y_train, Y_test = train_test_split(x, y, test_size=0.2)
5 print (y_test)
6 print (Y_test)
```

```
[2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2]
[1 2 1 2 2 2 2 1 0 1 1 1 2 2 0 1 2 1 0 1 2 1 2 1 2 2 2 1 0 1]
```

  - Line 4 : 데이터를 무작위로 혼합한 후 x의 **80%**를 X_train, x의 20%를 X_test, y의 80%를 Y_train, y의 20%를 Y_test

▪ 함수 설명
- Train_test_split() : train set과 test set 분리하고, test set의 size를 전체 데이터의 20%로 설정(상세 설명 7장 참조)

▪ 변수 설명
- x : iris 데이터의 feature 값
- y : iris 데이터의 target 값
- y_test : 앞에서 마지막 30개 데이터를 선택한 테스트 데이터 target 값
- Y_test : Train_test_split()를 이용해 생성한 테스트 데이터 target 값

3. Random Forest 성능 제고 방법

- **Random Forest 분류기 성능 평가**

▪ 함수 설명
- rfc.predict(): 입력 데이터의 분류 결과 예측
- accuracy_score() : 분류 결과의 accuracy를 계산 (상세 설명 7장 참조)
- classification_report(): 분류 결과의 precision, recall을 계산 (상세 설명 7장 참조)

▪ 변수 설명
- Y_tset : Train_test_split()를 이용해 분리한 테스트 데이터 target 값
- prediction_1 : 테스트 데이터(X_train)의 target 예측 값

```
1  clf = RandomForestClassifier(n_estimators=10) # Random Forest
2  clf.fit(X_train, Y_train)
3  prediction_1 = clf.predict(X_test)
4  #print (prediction_1 == Y_test)
5  print ("Accuracy is : ",accuracy_score(prediction_1, Y_test))
6  print ("============================================================")
7  print (classification_report(prediction_1, Y_test))
```

```
Accuracy is :  0.966666666667
============================================================
             precision    recall  f1-score   support

          0       1.00      1.00      1.00        12
          1       1.00      0.89      0.94         9
          2       0.90      1.00      0.95         9

avg / total       0.97      0.97      0.97        30
```

- 분류기의 성능 accuracy, precision, recall 모두 제고
- Training 데이터를 제대로 선택하는 것이 매우 중요함

3. Random Forest 성능 제고 방법

- **Random Forest 분류기 성능 높이는 방법**

1) Tree의 개수 (n_estimators)를 변경
   - Tree의 개수를 적당히 확장하면 모듈의 성능을 높일 수 있음
   - Tree의 개수가 지나치게 많으면 성능은 오히려 낮아짐

2) max_features의 값을 변경

* (iris) data set의 size가 작은 이유로 성능 차이가 없음

```
# Initialize the model
clf_2 = RandomForestClassifier(n_estimators=200, # Number of trees
                               max_features=4,    # Num features considered
                                  oob_score=True)    # Use OOB scoring*
clf_2.fit(X_train, Y_train)
prediction_2 = clf_2.predict(X_test)
print (prediction_2 == Y_test)
print ("Accuracy is : ",accuracy_score(prediction_2, Y_test))
print ("=====================================================")
print (classification_report(prediction_2, Y_test))
```

```
[ True  True  True  True  True  True  True  True  True  True False  True
  True  True  True  True  True  True  True  True  True  True  True  True
  True  True  True  True  True]
Accuracy is :  0.966666666667
=====================================================
             precision    recall  f1-score   support

          0       1.00      1.00      1.00        12
          1       1.00      0.89      0.94         9
          2       0.90      1.00      0.95         9

avg / total       0.97      0.97      0.97        30
```

- 변수 설명
- prediction_2 : tree가 200개인 Random Forest를 이용해 예측한 테스트 데이터 target 값

3. Random Forest 성능 제고 방법

- **각 feature의 중요도 확인**

```
for feature, imp in zip(iris.feature_names, clf_2.feature_importances_):
    print(feature, imp)
```

```
sepal length (cm) 0.0142678005207
sepal width (cm) 0.0179653187533
petal length (cm) 0.386870282196
petal width (cm) 0.58089659853
```

– 분류기를 생성할 때 RandomForestClassifier()의 파라미터 'oob_score=True' 선택

[iris decision tree 그림 그리기]
https://towardsdatascience.com/how-to-visualize-a-decision-tree-from-a-random-forest-in-python-using-scikit-learn-38ad2d75f21c

★ 정리하기

- Random Forest는 여러 개의 decision tree를 결합해 하나의 모형을 생성
- Random Forest 모듈에서 tree의 개수가 많을수록 좋은 것은 아님
- 분류 모듈에서 training data와 test data의 선택은 매우 중요함