

# 6 머신러닝 이해하기 - 이론과 실습

# CONTENTS

01

학습, 모델, 예측

02

분류(Classification)  
- 결정 트리  
(Decision Tree)

03

실습



# 01 학습, 모델, 예측

01-1 소단원명

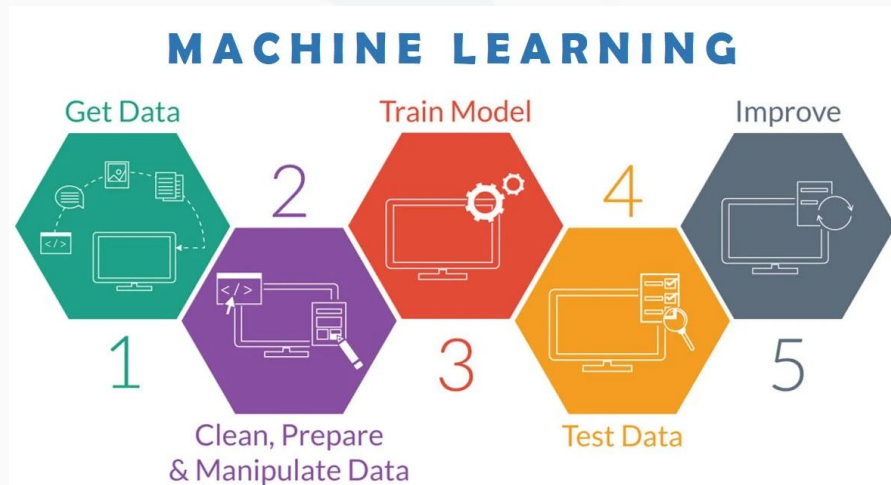
## 모델

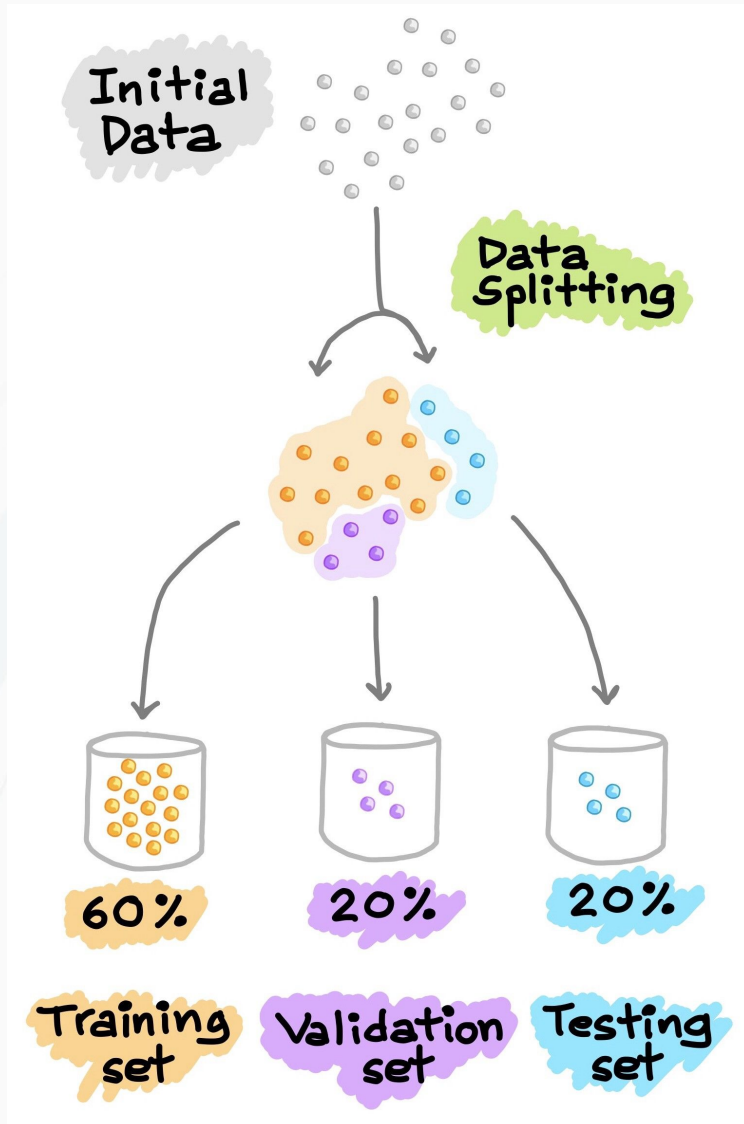
모델은 데이터와 밀접한 관계

어떤 모델을 사용하는가는 어떤 데이터를 사용하는가 중요한 영향을 미침

학습 > 테스트 -> 수정/개선 -> 다시 학습...

- Regression
- Classification
- Clustering
- Dimensionality Reduction
- Neural Networks
- Ensemble Methods
- Deep Learning





## 모델 학습

학습(training)도 중요하지만, 제대로 모델이 만들어졌는지 확인하기 위해서는 테스트(test)도 중요합니다.

data를 분리하는 방법중 하나로는:

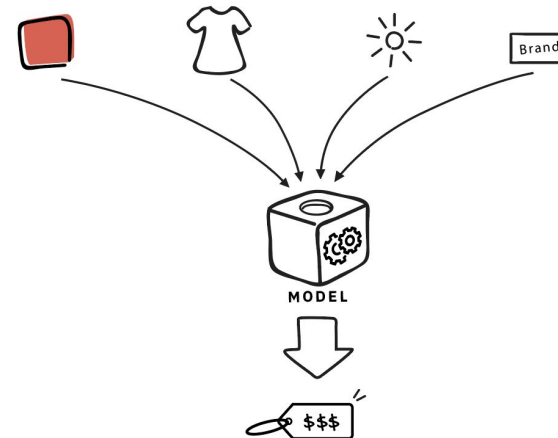
- training set
- validation set
- testing set

## 예측 (prediction)

- accuracy (정확도)
  - 정확도는 모델이 올바른 예측(correct prediction)을 하는 전체적인 빈도
- error rate (오류율)
  - 오류율은 전체적으로 모델이 잘못된 예측(wrong prediction)을 하는 빈도
- \*recall, \*\*precision 등등

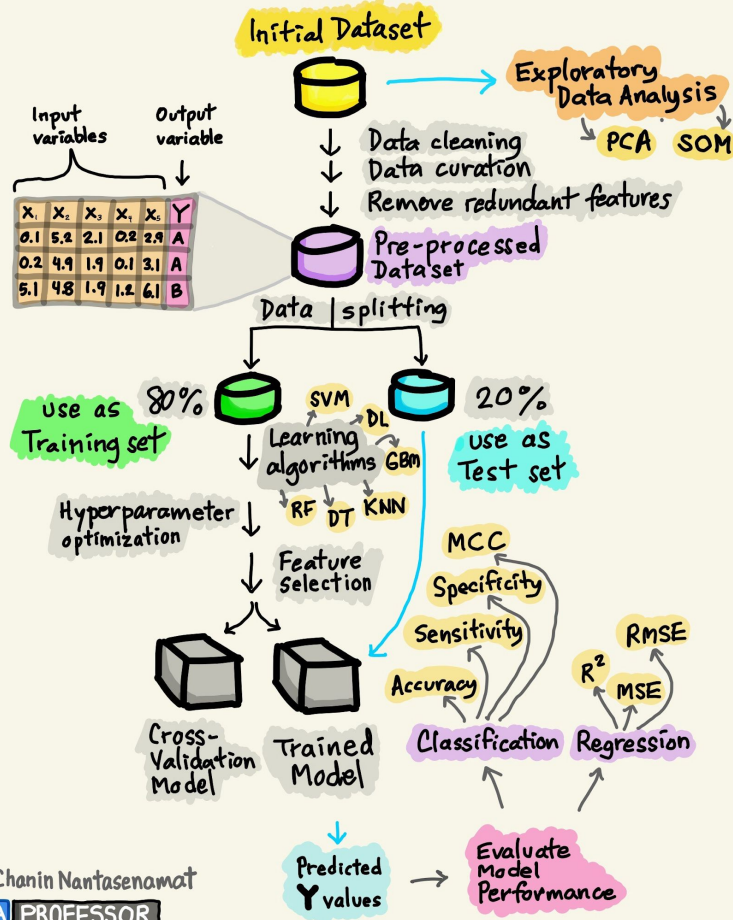
\*Recall is a measure that tells us how great our model is when all the actual values are positive.

\*\*Precision is when the model predicts a positive value then what are the odds that the model has made a correct prediction.





# BUILDING THE MACHINE LEARNING MODEL



By: Chanin Nantasenamat

DATA PROFESSOR

<http://youtube.com/dataprofessor>

January 1, 2020

## Building the Machine Learning model

크게보기

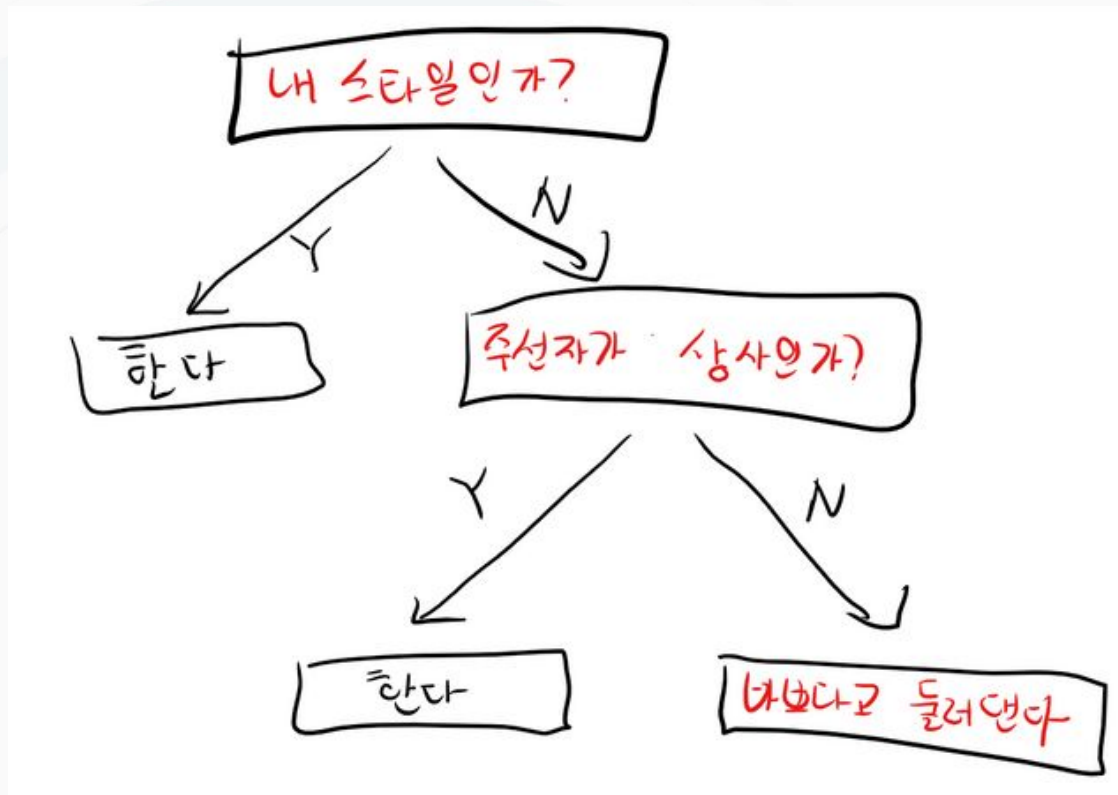
<https://towardsdatascience.com/how-to-build-a-machine-learning-model-439ab8fb3fb1>



# 02 결정 트리 (Decision Tree)



Decision Tree = 결정트리, 의사결정트리, 의사결정나무, 결정나무

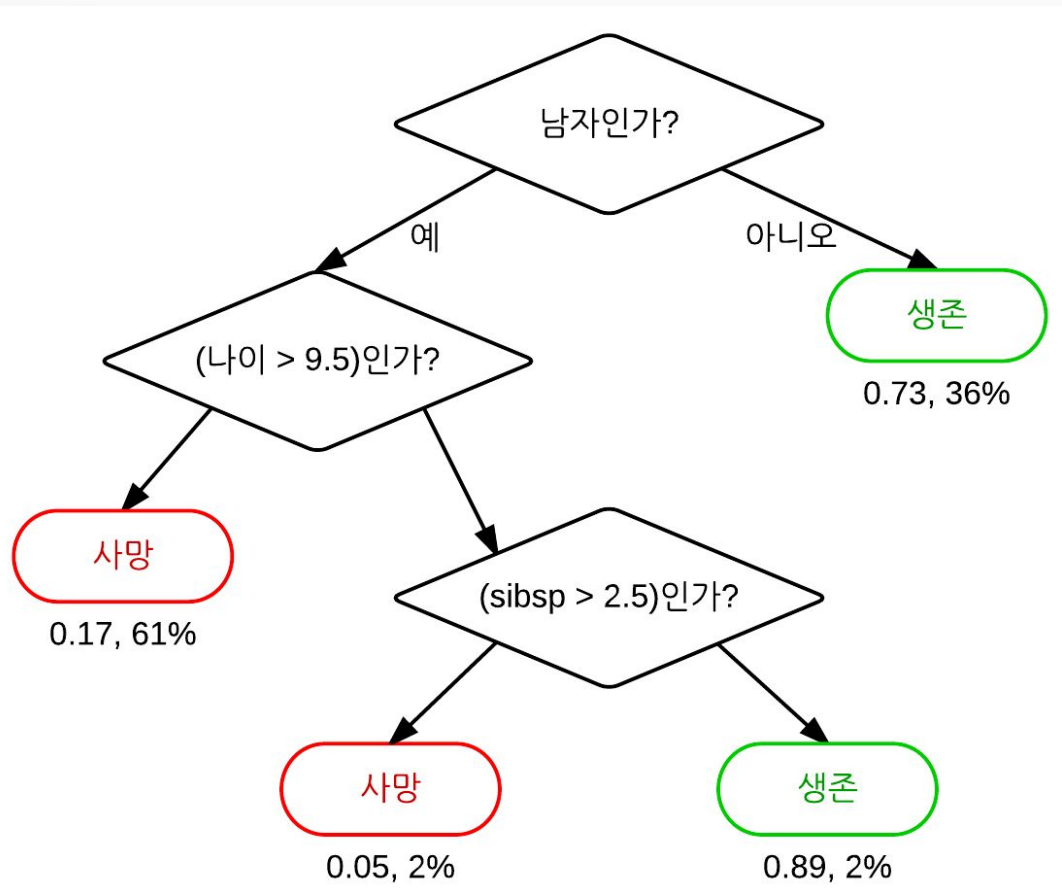


타이타닉호 탑승객의 생존 여부를 나타내는 결정 트리.

(“sibsp”는 탑승한 배우자와 자녀의 수를 의미한다.)

앞 아래의 숫자는 각각 생존 확률과 탑승객이 그 앞에 해당될 확률을 의미한다.

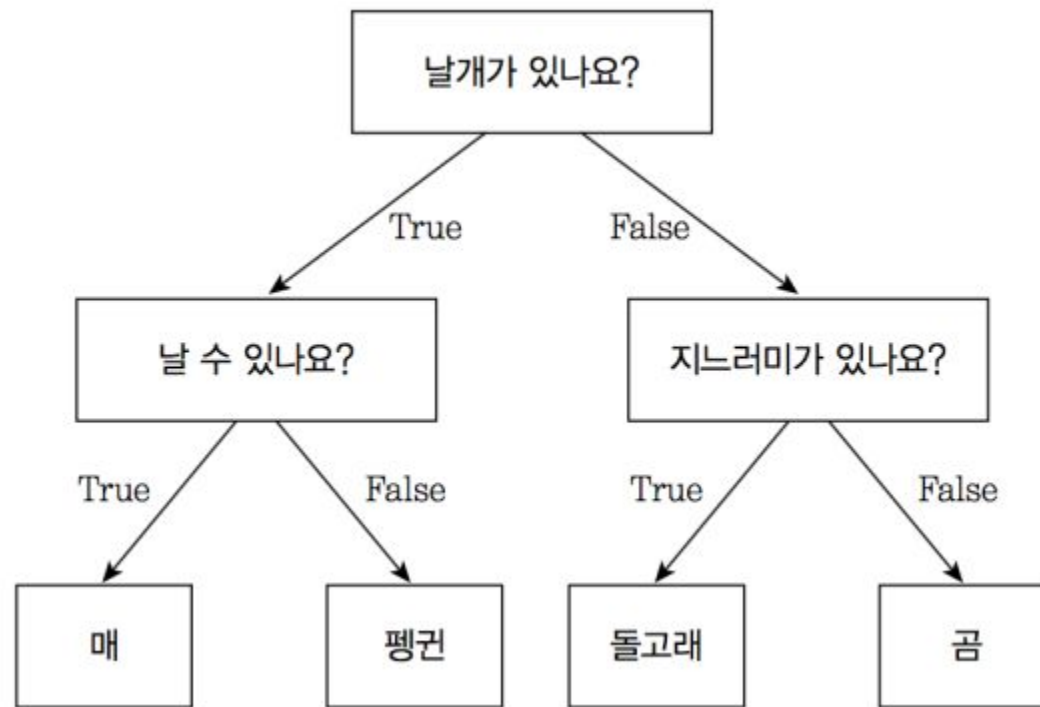
[출처: ko.wikipedia.org]



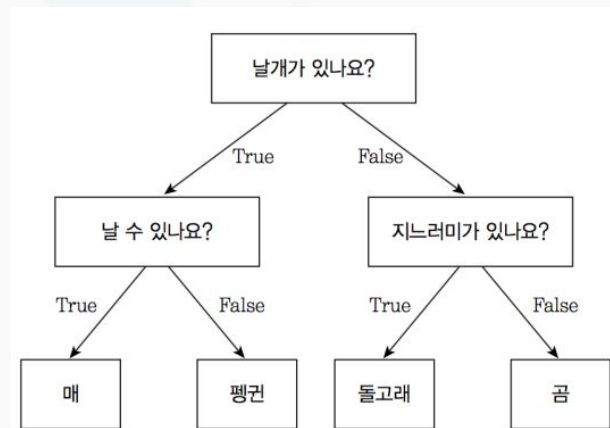
## 의사결정트리 (Decision Tree)

- 전형적인 분류 모델
- 매우 직관적인 방법 중 하나
- 다른 모델들과 다르게 결과물이 시각적으로 읽히기 쉬운형태가 장점
- 어떤 곳에 사용되나요?
  - 대출을 원하는 사람의 신용평가
  - 독버섯과 버섯을 분류
  - 실질적인 분류에 자주 사용됨

정보획득량 / 엔트로피의 개념 이해가 필요.

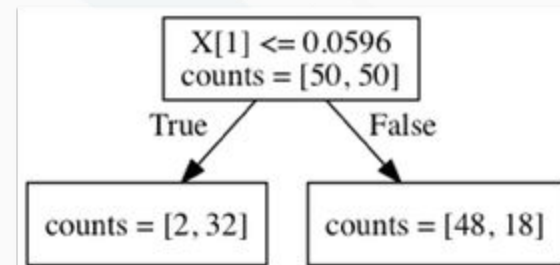
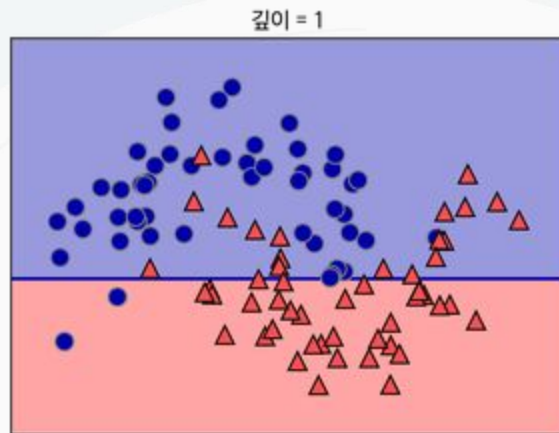


결정 트리(Decision Tree, 의사결정트리, 의사결정나무라고도 함)는 분류(Classification)와 회귀(Regression) 모두 가능한 지도 학습 모델 중 하나입니다. 결정 트리는 스무고개 하듯이 예/아니오 질문을 이어가며 학습합니다. 매, 펭귄, 돌고래, 곰을 구분한다고 생각해봅시다. 매와 펭귄은 날개를 있고, 돌고래와 곰은 날개가 없습니다. '날개가 있나요?'라는 질문을 통해 매, 펭귄 / 돌고래, 곰을 나눌 수 있습니다. 매와 펭귄은 '날 수 있나요?'라는 질문으로 나눌 수 있고, 돌고래와 곰은 '지느러미가 있나요?'라는 질문으로 나눌 수 있습니다.

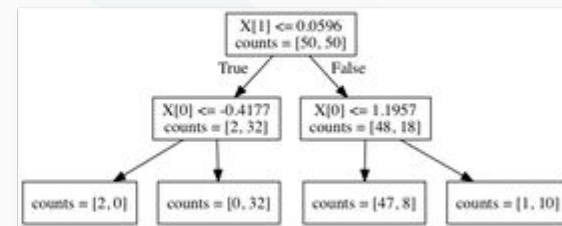
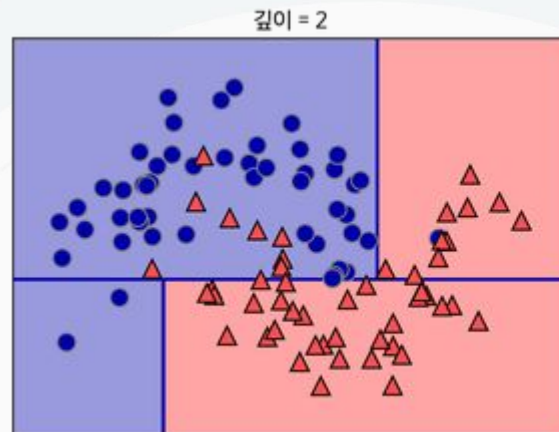




결정 트리 알고리즘의 프로세스를 간단히 알아보겠습니다.

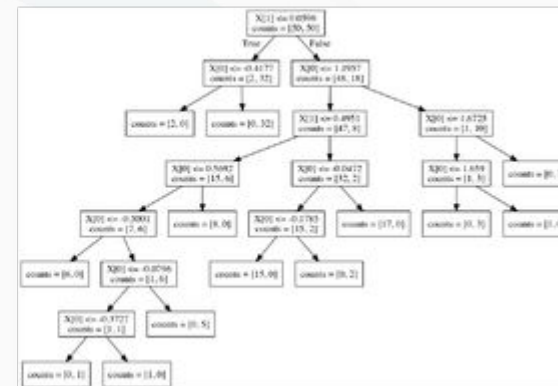


먼저 위와 같이 데이터를 가장 잘 구분할 수 있는 질문을 기준으로 나눕니다.



깊이 = 9

A 2D scatter plot showing a dataset partitioned by a decision tree with a depth of 9. The plot features two classes of data points: blue circles and red triangles. The space is divided into several rectangular regions by vertical and horizontal lines. The regions are colored light blue or light red, representing the predicted class for that area. The blue circles are primarily located in the left and upper-middle regions, while the red triangles are concentrated in the right and lower-middle regions. The decision boundaries are defined by the lines separating these colored regions.





# 02 결정 트리 - 같이하기 (Decision Tree)

## 같이하기

이 데이터(매, 펭귄, 돌고래, 곰)를 가지고 예제를 만들어서 간단한 결정트리 모델을 학습, 예측 해 보도록 하겠습니다.

가장 처음으로 라이브러리를 импорт 합니다.

```
[1] from sklearn.tree import DecisionTreeClassifier
```

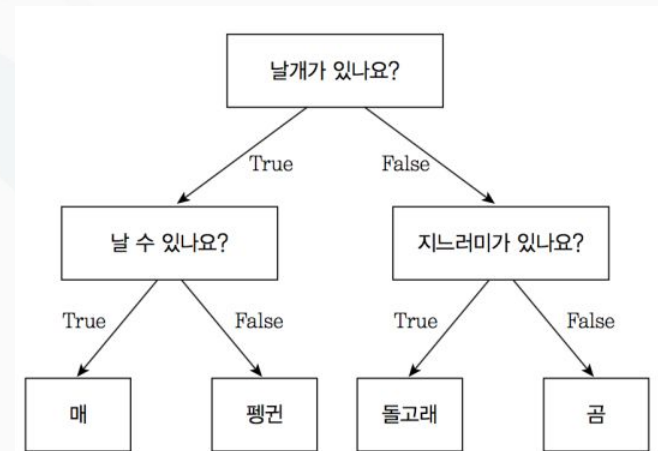


## 같이하기

## 학습데이터 만들기

- Yes, True -> 1 / No, False -> 0
- N/A(해당사항 없음) -> -1

날개가 있나요?	날 수 있나요?	지느러미가 있나요?	답
1	1	-1	매(1)
1	0	-1	펭귄(2)
0	-1	1	돌고래(3)
0	-1	0	곰(4)



같이하기  
학습데이터 만들기

```
# column 1 - 날개가 있나요?
# column 2 - 날 수 있나요?
# column 3 - 지느러미가 있나요?
x_train = [[1,1,-1],[1,0,-1],[0,-1,1],[0,-1,0]]
# 매 - 1, 펭귄 - 2, 돌고래 - 3, 곰 - 4
y_train = [[1],[2],[3],[4]]
```

날개가 있나요?	날 수 있나요?	지느러미가 있나요?	답
1	1	-1	매(1)
1	0	-1	펭귄(2)
0	-1	1	돌고래(3)
0	-1	0	곰(4)

## 같이하기

## 학습데이터 만들기

print() 함수를 사용해서 학습 데이터가 잘 만들어 졌는지 확인합니다.

```
[3] print(x_train)  
     print(y_train)
```

```
[[1, 1, -1], [1, 0, -1], [0, -1, 1], [0, -1, 0]]  
[[1], [2], [3], [4]]
```

## 모델 만들기

DecisionTreeClassifier()?

**sklearn.tree**.DecisionTreeClassifier

<https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html>



```
DecisionTreeClfModel = DecisionTreeClassifier()
```

\*sklearn에서는 의사결정트리 알고리즘을 구현하는 라이브러리를 제공한다. DecisionTreeClassifier()를 사용하여 의사결정트리 알고리즘을 구현할 수 있다.

## 모델 만들기

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',  
                        max_depth=None, max_features=None, 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, presort='deprecated',  
                        random_state=None, splitter='best')
```



## 학습하기



```
DecisionTreeClfModel.fit(x_train,y_train)
```

fit()

<https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier.fit>

## 학습하기



```
DecisionTreeClfModel.fit(x_train,y_train)
```

***X{array-like, sparse matrix} of shape (n\_samples, n\_features)***

The training input samples. Internally, it will be converted to `dtype=np.float32` and if a sparse matrix is provided to a sparse `csc_matrix`.

## 학습하기



```
DecisionTreeClfModel.fit(x_train,y_train)
```

***yarray-like of shape (n\_samples,) or (n\_samples, n\_outputs)***

The target values (class labels) as integers or strings.

## 예측하기



```
#날개가 있고, 날수 없음 => 펭귄 - 2  
x_test = [[1, 0, -1]]
```



```
y_pred = DecisionTreeClfModel.predict(x_test)  
print(y_pred)
```

## 예측하기

▶ #날개가 있고, 날수 없음 => 펭귄 - 2  
 x\_test = [[1, 0, -1]]

▶ y\_pred = DecisionTreeClfModel.predict(x\_test)  
 print(y\_pred)

[2]

날개가 있나요?	날 수 있나요?	지느러미가 있나요?	답
1	1	-1	매(1)
1	0	-1	펭귄(2)
0	-1	1	돌고래(3)
0	-1	0	곰(4)



## 예측하기 ep. 2

날 수 있는지 없는지 정확하게 알 수 없을때.

- 50%만 정확할때
- 80%정도 정확할때

날개가 있나요?	날 수 있나요?	지느러미가 있나요?	답
1	1	-1	매(1)
1	0	-1	펭귄(2)
0	-1	1	돌고래(3)
0	-1	0	곰(4)

## 예측하기 ep. 2

날 수 있는지 없는지 정확하게 알 수 없을때.

- 50%만 정확할때

날개가 있나요?	날 수 있나요?	지느러미가 있나요?	답
1	1	-1	매(1)
1	0	-1	펭귄(2)
0	-1	1	돌고래(3)
0	-1	0	곰(4)



```
x_test2 = [[1, 0.5, -1]]
y_pred = DecisionTreeClfModel.predict(x_test2)
print(y_pred)
```

[2]

## 예측하기 ep. 2

날 수 있는지 없는지 정확하게 알 수 없을때.

- 80%정도 정확할때

날개가 있나요?	날 수 있나요?	지느러미가 있나요?	답
1	1	-1	매(1)
1	0	-1	펭귄(2)
0	-1	1	돌고래(3)
0	-1	0	곰(4)



```
x_test2 = [[1, 0.8, -1]]
y_pred = DecisionTreeClfModel.predict(x_test2)
print(y_pred)
```

```
[1]
```

## 결정 트리의 장점과 단점

### 장점

- 쉽고 직관적입니다.
- 각 피처의 스케일링과 정규화 같은 전처리 작업의 영향도가 크지 않습니다.

### 단점

- 규칙을 추가하며, 서브트리(sub-tree)를 만들어 나갈수록 모델이 복잡해지고, 과적합에 빠지기 쉽습니다.
  - 트리의 크기를 사전에 제한하는 튜닝이 필요합니다.

## Making a Model In Real World

- 실제로 의사결정트리 그래프를 기반으로 학습 세트를 만드는 것은 매우 드뭅니다.
- 여러가지 모델을 가지고 '학습 → 예측 → 모델 수정 → 학습자료 수정 → 학습'을 반복하게 됩니다.



# 03 결정 트리 - 실습 (Decision Tree)

## Activity

## 학습 데이터 만들기

직접 학습데이터를 수집해 봅니다.

소개팅에 나간 그 사람이 성공할 수 있을까요?

예제)

ID	성격	경제력	외모	성공
1	1	1	2	1
2	1	2	1	1
3	1	3	2	0
4	2	1	1	1
5	2	2	2	0
...	...	...	...	...



## 학습 데이터 만들기

```
# [column 0 - ID 고유번호는 포함시키지 않음]
# column 1 - 성격: 1(high), 2(mid), 3(low)
# column 2 - 경제력: 1(high), 2(mid), 3(low)
# column 3 - 외모: 1(high), 2(low)
# column 4 - 성공
```

```
x_train = [
    [1, 1, 2], # ID = 1
    [1, 2, 1], # ID = 2
    [1, 3, 2], # ID = 3 Fail
    [2, 1, 1], # ID = 4
    [2, 2, 2], # ID = 5
    [2, 3, 1], # ID = 6 Fail
    [3, 1, 2], # ID = 7
    [3, 2, 1], # ID = 8
    [3, 3, 2], # ID = 9 Fail
    [1, 3, 1] # ID = 10 Fail
]
```

```
y_train = [
    [1], # ID = 1
    [1], # ID = 2
    [0], # ID = 3
    [1], # ID = 4
    [0], # ID = 5
    [0], # ID = 6
    [1], # ID = 7
    [1], # ID = 8
    [0], # ID = 9
    [0] # ID = 10
]
```

## 불러오기(import)

1. `sklearn.tree` 에서 `DecisionTreeClassifier`를 불러옵니다.
2. `Alias(as)` 없이 이름은 그대로 사용합니다.

## 학습 데이터 만들기

직접 학습데이터를 수집해 봅니다.

[총합:1.5]

- x\_train - feature
- y\_train - class / output / answer

## 학습 데이터 만들기

1. `print()` 함수를 사용하여 `x_train` 데이터를 확인
  2. `print()` 함수를 사용하여 `y_train` 데이터를 확인
- 개수와 정렬이 정확한지 확인이 중요

## 모델 만들기

1. 변수의 이름 = `DecisionTreeClfModel`
2. `DecisionTreeClassifier()` 함수를 사용
3. `DecisionTreeClfModel`이 잘 만들졌는지 확인
  - a. `print()` 함수 사용

## 학습하기

1. DecisionTreeClfModel 모델의 fit() 함수를 사용
2. 첫번째 parameter(매개변수)는 feature(속성) 데이터가 들어가야 함으로 x\_train
3. 두번째 parameter를 output/label 데이터가 들어가야 함으로 y\_train
4. 어떤 예러도 없는 경우는 아래와 같이 출력됩니다.

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',  
                        max_depth=None, max_features=None, 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, presort='deprecated',  
                        random_state=None, splitter='best')
```

## 예측하기

1. 변수이름 = x\_test
2. 예측하고 싶은 데이터를 생성
  - 2차원 list 배열을 사용
3. #2의 데이터를 x\_test에 assign
4. 변수이름 = y\_pred
5. DecisionTreeClfModel의 predict() 함수를 사용해서 나온값을 y\_pred에 assign
  - a. parameter에 #3에서 만든 x\_test를 사용
6. print() 함수를 사용해서 y\_pred이 예측된 값을 사용합니다.
7. 1-6를 예측하고 싶은 데이터를 바꿔서 실행



```
# column 1 - 성격  
# column 2 - 경제력  
# column 3 - 외모
```



## 시각화

조금 더 자세히, 시각적으로 보기 위해서 아래와 같은 라이브러리를 사용합니다:

1. `sklearn.tree`의 `export_graphviz`
2. `sklearn`의 `tree`
3. `IPython.display`의 `SVG`
4. `graphviz`의 `Source`
5. `IPython.display`의 `display`

## 시각화

```
from sklearn.tree import export_graphviz
from sklearn import tree
from IPython.display import SVG
from graphviz import Source
from IPython.display import display
```

```
# [column 0 - ID 고유번호는 포함시키지 않음]
# column 1 - 성격
# column 2 - 경제력
# column 3 - 외모
# column 4 - 성공
labels = ['성격', '경제력', '외모']
```

```
y_labels = ['실패', '성공']
```

```
graph = Source(tree.export_graphviz(DecisionTreeClfModel ,feature_names = labels, class_names =
y_labels , filled = True))
display(SVG(graph.pipe(format='svg')))
```

## 시각화

- Gini Index(지니계수)란?



## 시각화

- Gini Index(지니계수)란?
  - 전체 **Box**안에 특정 **Class**에 속하는 관측치의 비율을 모두 제외한 값
  - 그림의 예
    - 좌측 그림의  $Gini = 1 - [(7/12)^2 + (5/10)^2] = 0.4097$
    - 우측 그림에서의  $Gini = 1 - [(0/10)^2 + (9/11)^2] = 0.3305$
    - 우측이 분류가 더 잘 됨
  - Gini Index는 작은 숫자일수록 분류가 더 잘되어지는 데이터라 할 수 있습니다.





## Reference

<https://www.fiverr.com/azizpresswala/create-machine-learning-or-deep-learning-models>

<https://towardsdatascience.com/how-to-build-a-machine-learning-model-439ab8fb3fb1>

<https://srnghn.medium.com/machine-learning-trying-to-predict-a-numerical-value-8aafb9ad4d36>

<https://towardsdatascience.com/is-accuracy-everything-96da9afd540d>

<https://steemit.com/kr/@gillime/r-4-decision-tree>

[https://ko.wikipedia.org/wiki/%EA%B2%B0%EC%A0%95\\_%ED%8A%B8%EB%A6%AC\\_%ED%95%99%EC%8A%B5%EB%B2%95](https://ko.wikipedia.org/wiki/%EA%B2%B0%EC%A0%95_%ED%8A%B8%EB%A6%AC_%ED%95%99%EC%8A%B5%EB%B2%95)

<https://gomguard.tistory.com/86>

<https://bkshin.tistory.com/entry/%EB%A8%B8%EC%8B%A0%EB%9F%AC%EB%8B%9D-4-%EA%B2%B0%EC%A0%95-%ED%8A%B8%EB%A6%ACDecision-Tree>

<https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier.fit>

<https://injo.tistory.com/15>