PRACTICE

ARTIFICIAL INTELLIGENCE

WHAT WE WILL LEARN TODAY?

지도 학습
Supervised Learning

선형 회귀 (Linear Regression)

로지스틱 회귀 (Logistic Regression)

결정 트리 (Decision Tree)

인공 신경망 (ARTIFICIAL NEURAL NETWORK)

퍼셉트론 (Perceptron)

합성곱 신경망 (Convolutional Neural Network)

비지도 학습
Unsupervised Learning

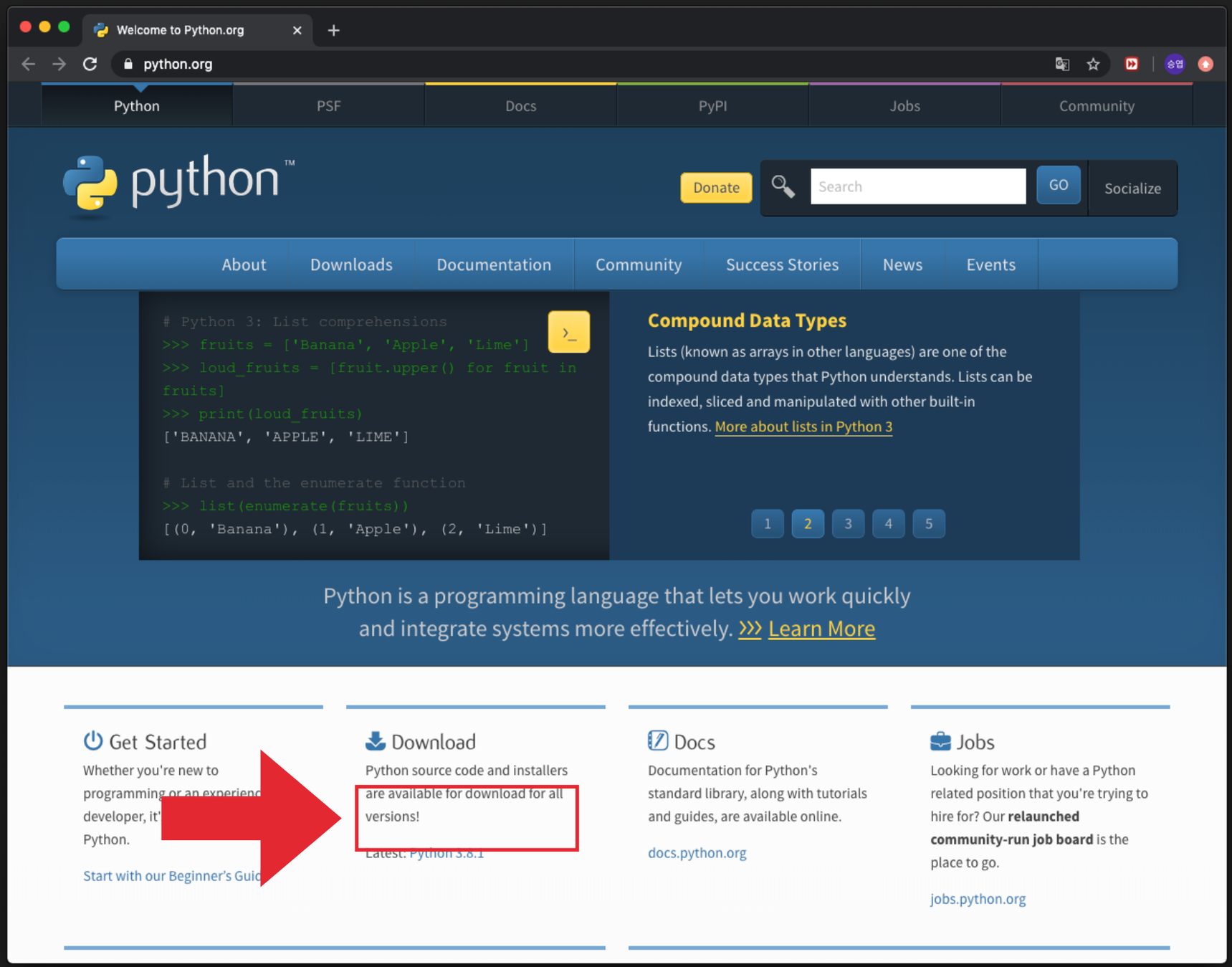
K-mean Clustering

BEFORE WE START...

BEFORE WE START...

Visual Studio Code

INSTALL PYTHON



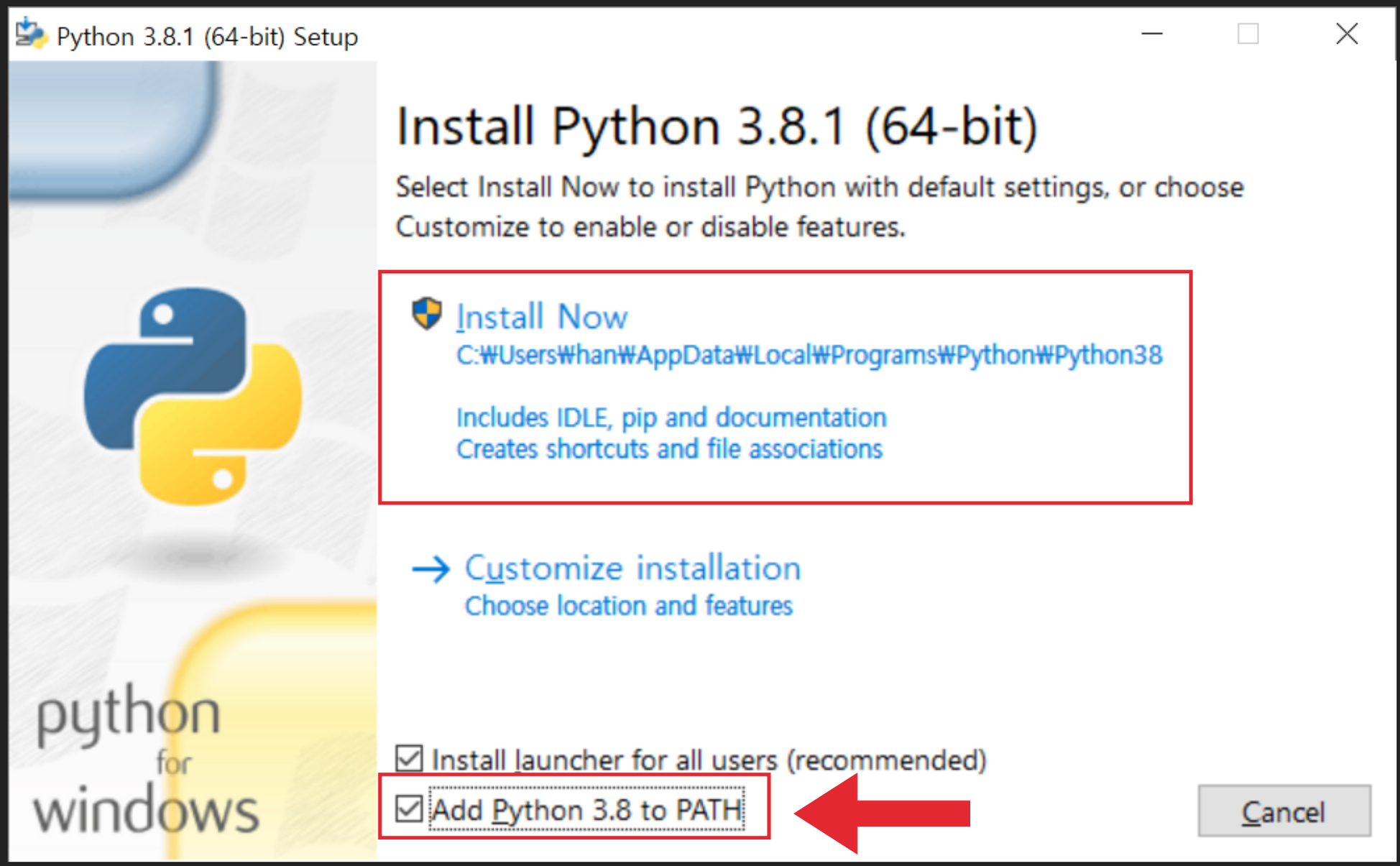
https://python.org

INSTALL PYTHON

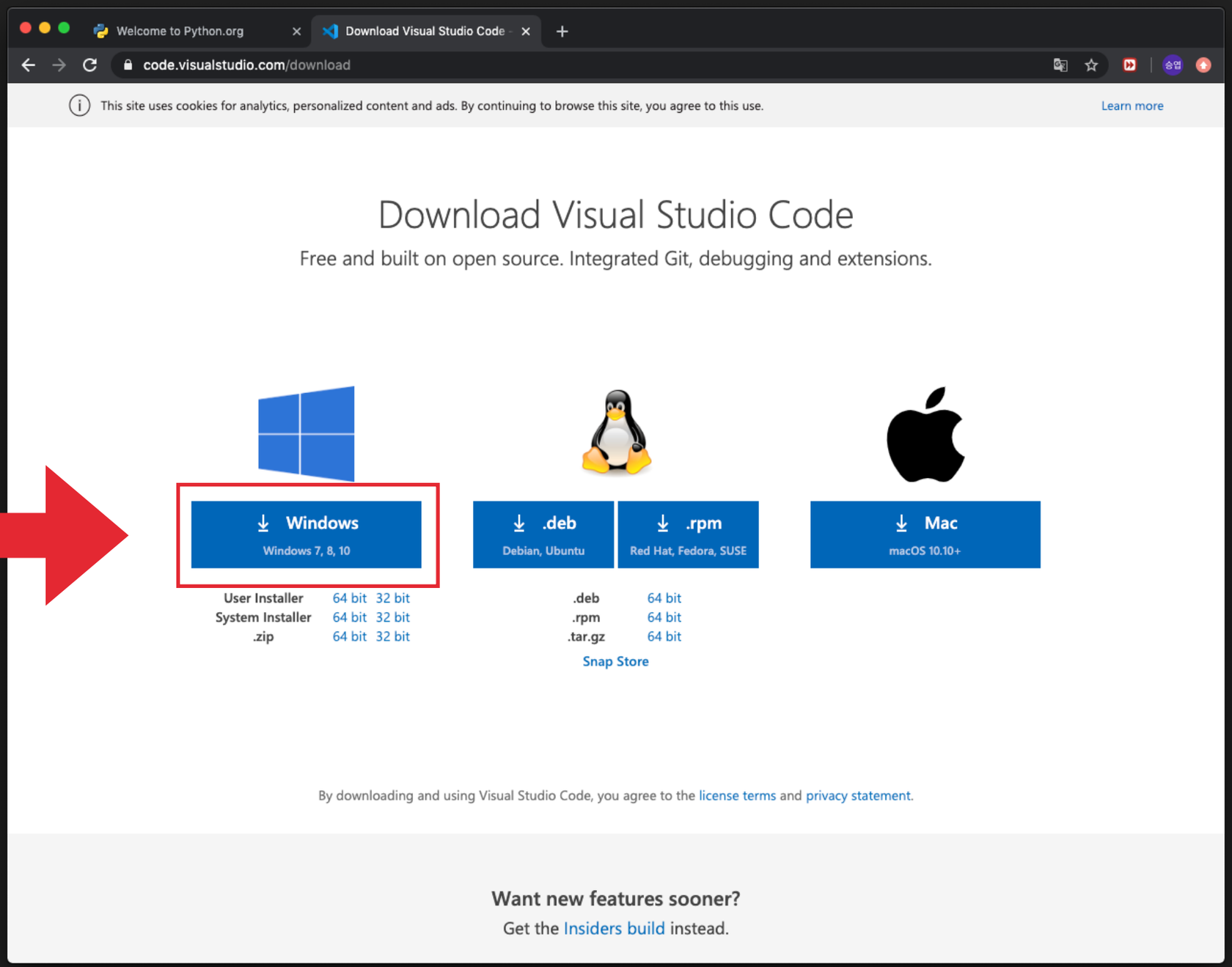
Files

Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		f215fa2f55a78de739c1787ec56b2bcd	23978360	SIG
XZ compressed source tarball	Source release		b3fb85fd479c0bf950c626ef80cacb57	17828408	SIG
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	d1b09665312b6b1f4e11b03b6a4510a3	29051411	SIG
Windows help file	Windows		f6bbf64cc36f1de38fbf61f625ea6cf2	8480993	SIG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	4d091857a2153d9406bb5c522b211061	8013540	SIG
Windows x86-64 executable installer		for AMD64/EM64T/x64	3e4c42f5ff8fcdbe6a828c912b7afdb1	27543360	SIG
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64	662961733cc947839a73302789df6145	1363800	SIG
Windows x86 embeddable zip file	Windows		980d5745a7e525be5abf4b443a00f734	7143308	SIG
Windows x86 executable installer	Windows		2d4c7de97d6fcd8231fc3decbf8abf79	26446128	SIG
Windows x86 web-based installer	Windows		d21706bdac544e7a968e32bbb0520f51	1325432	SIG

INSTALL PYTHON

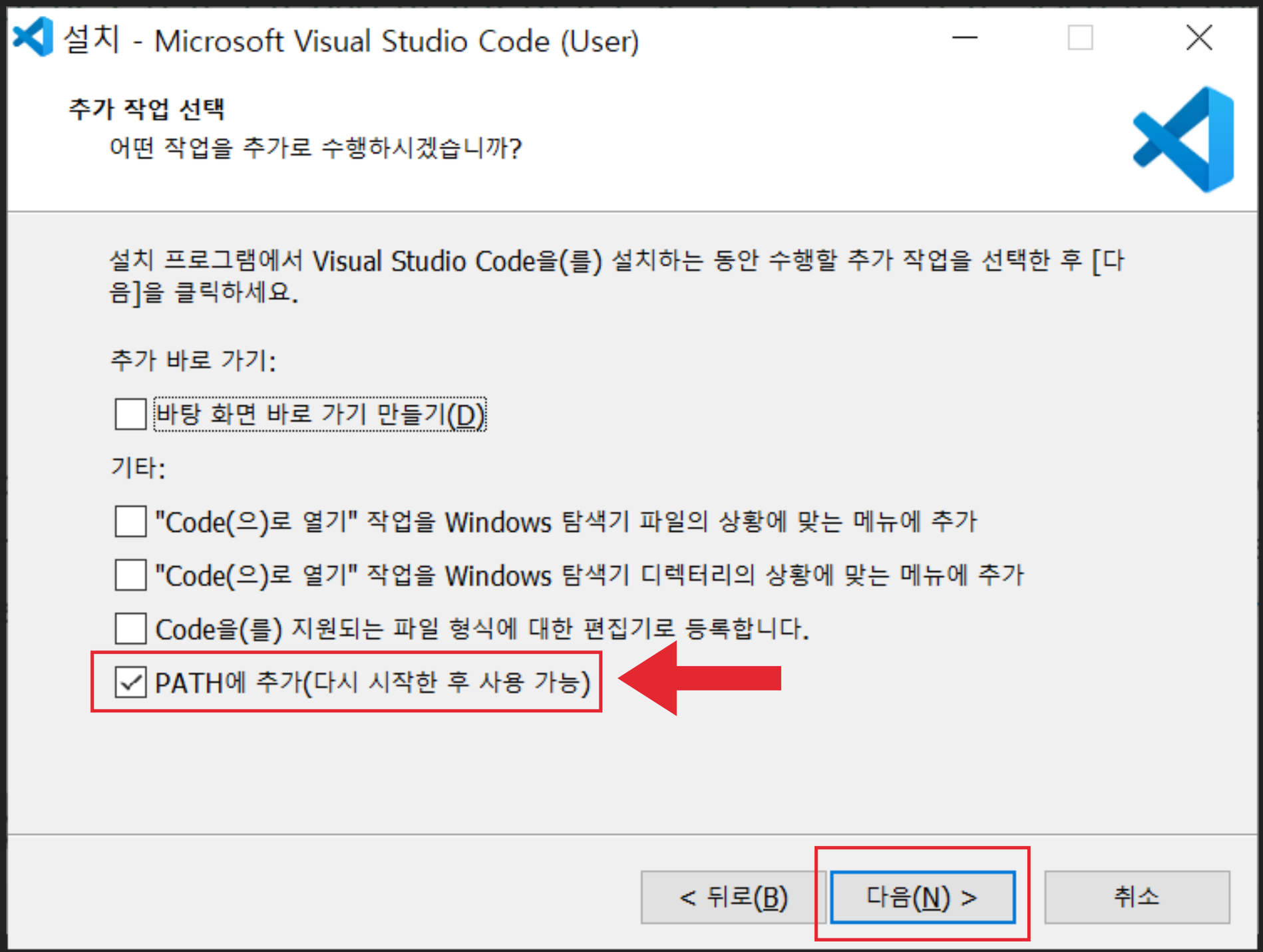


INSTALL VISUAL STUDIO CODE

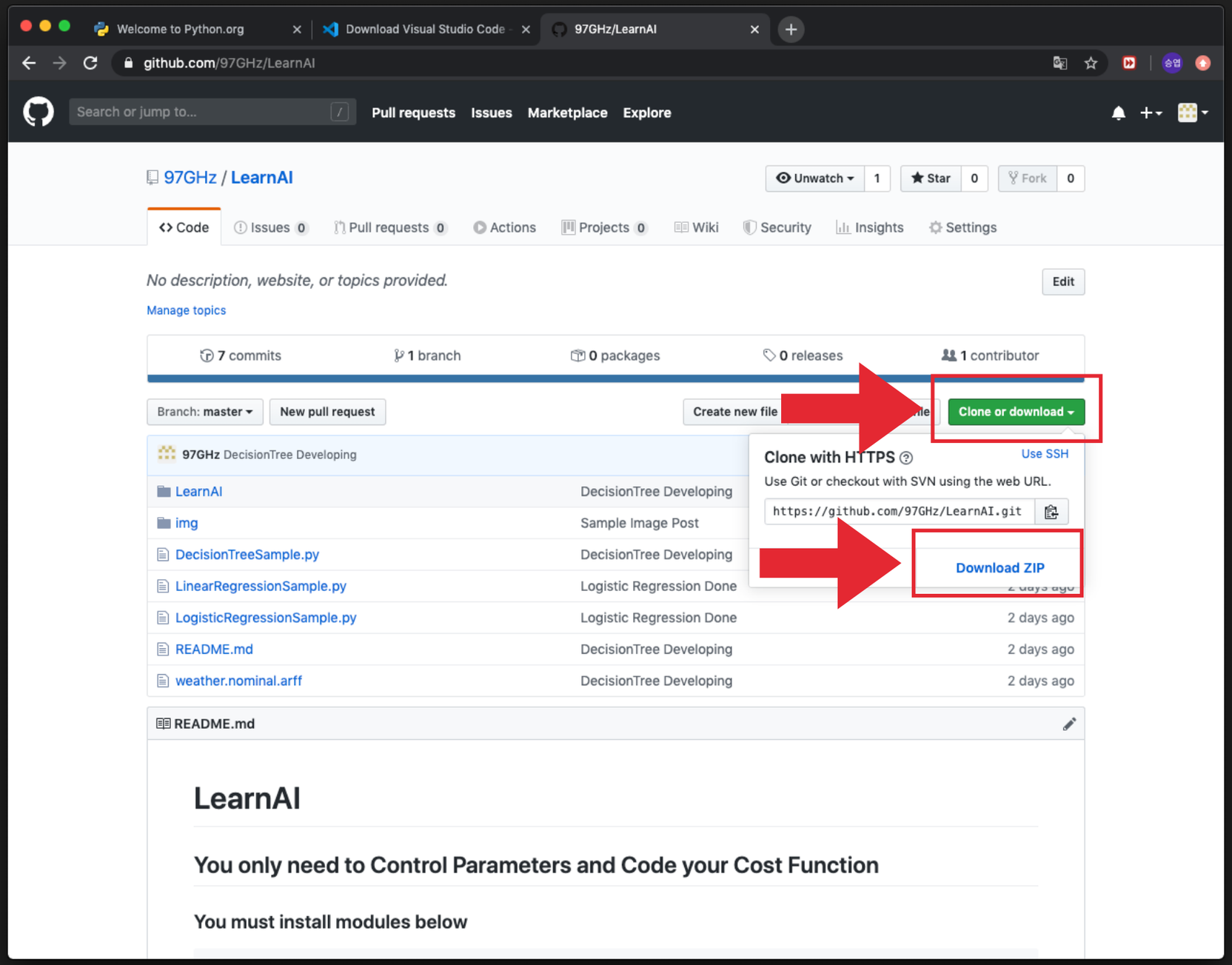


https://code.visualstudio.com/download

INSTALL VISUAL STUDIO CODE

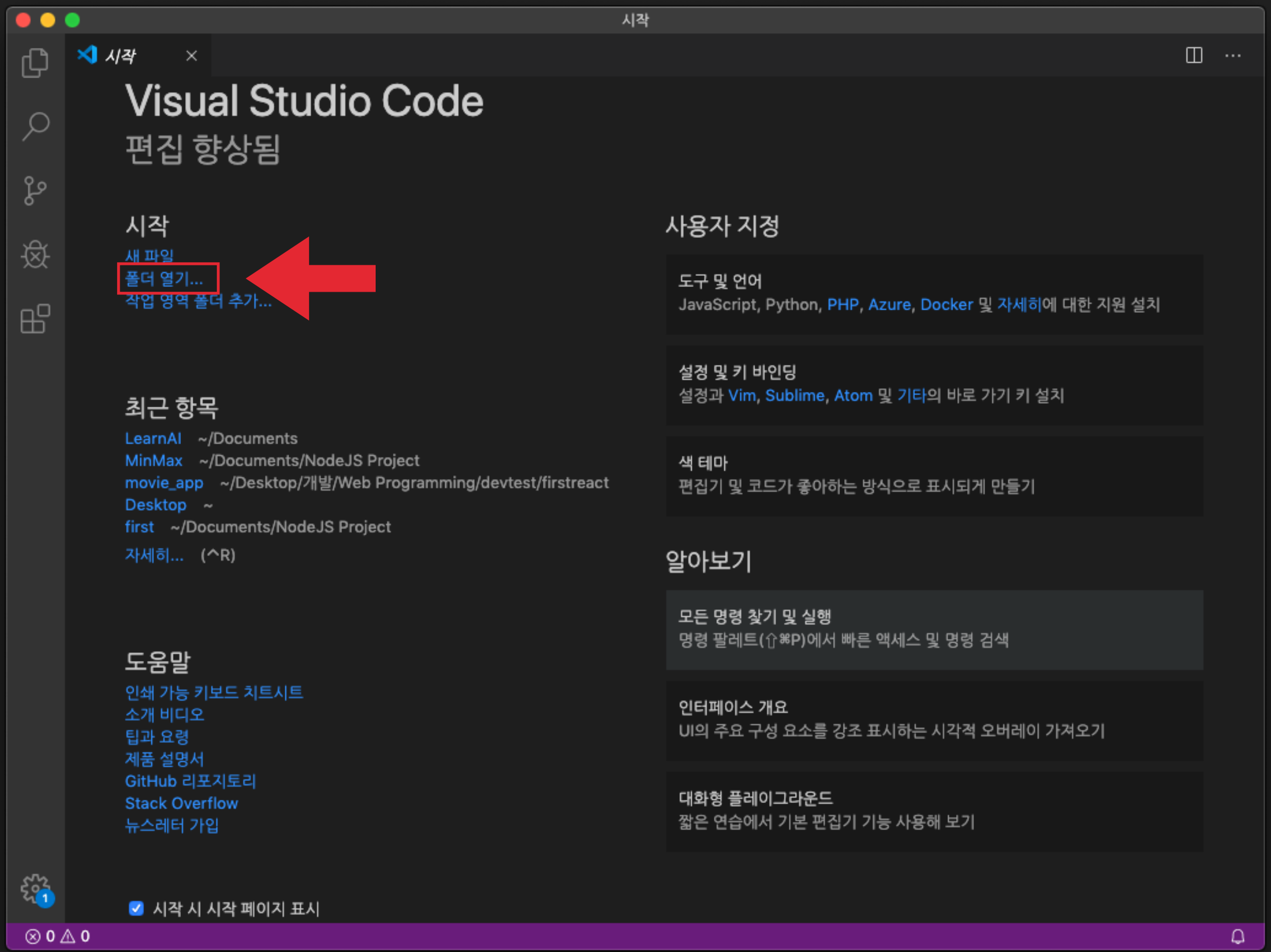


DOWNLOAD MODULES AND SAMPLES

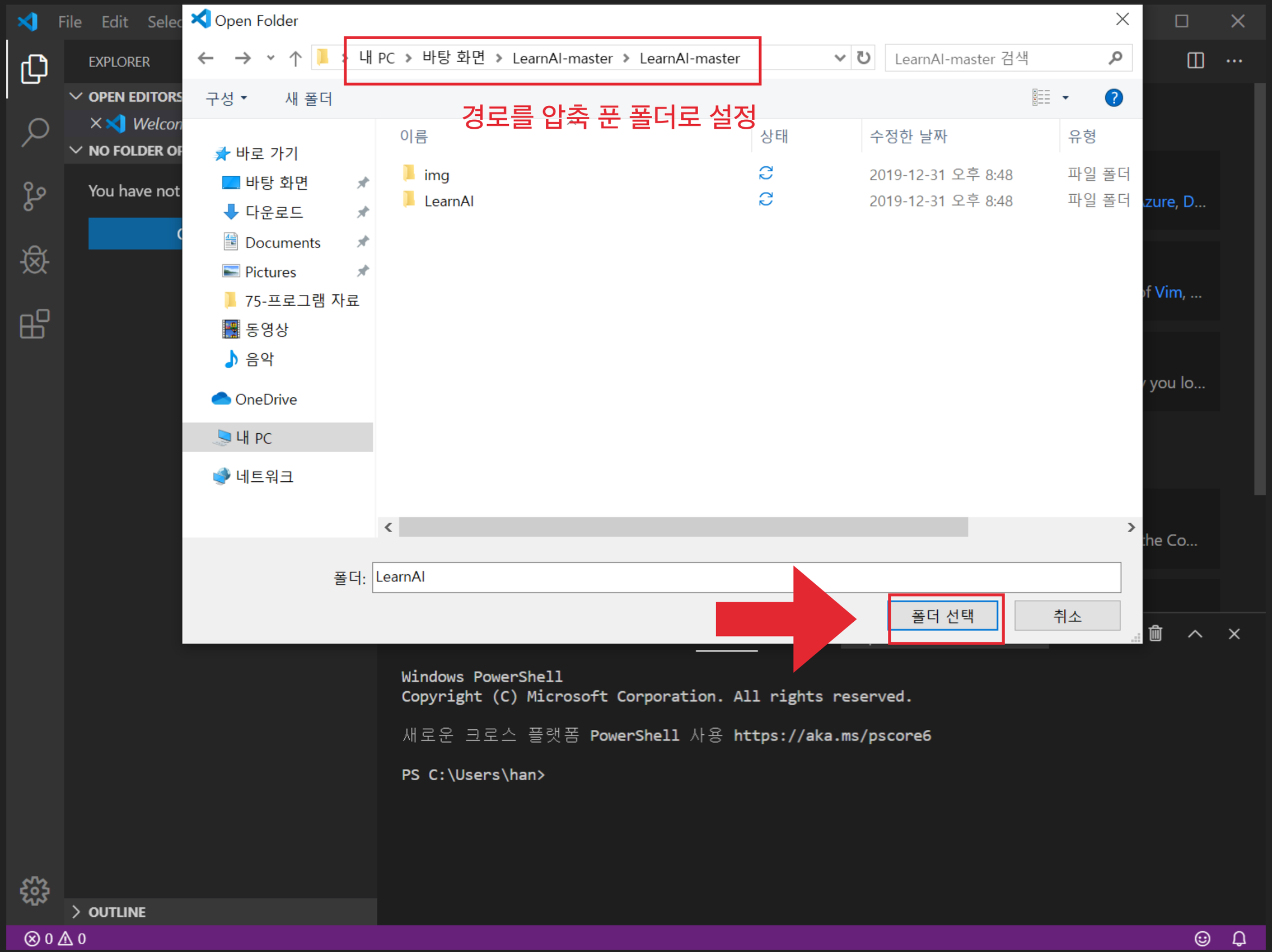


https://github.com/97GHz/LearnAI

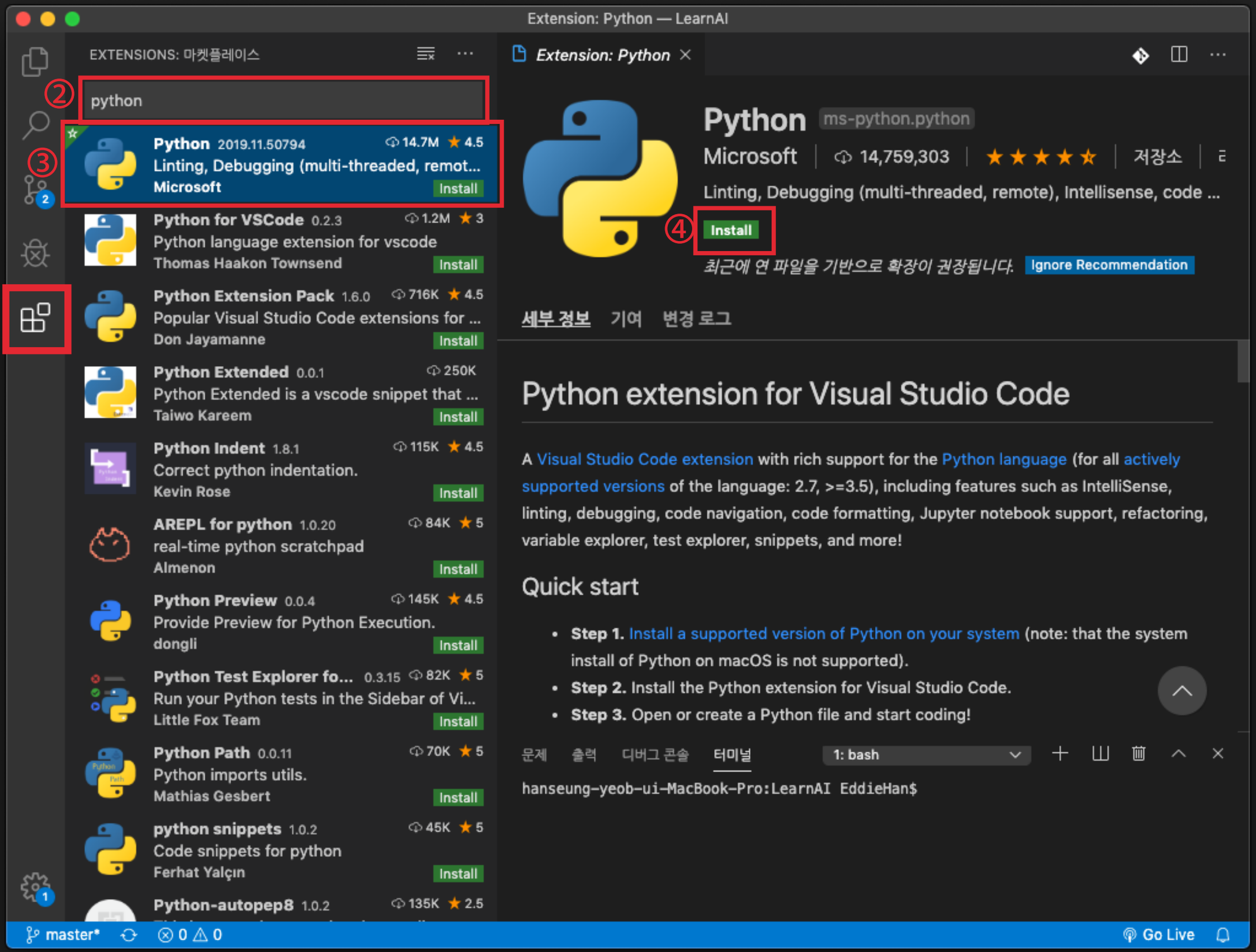
1. INSTALL & SETUP



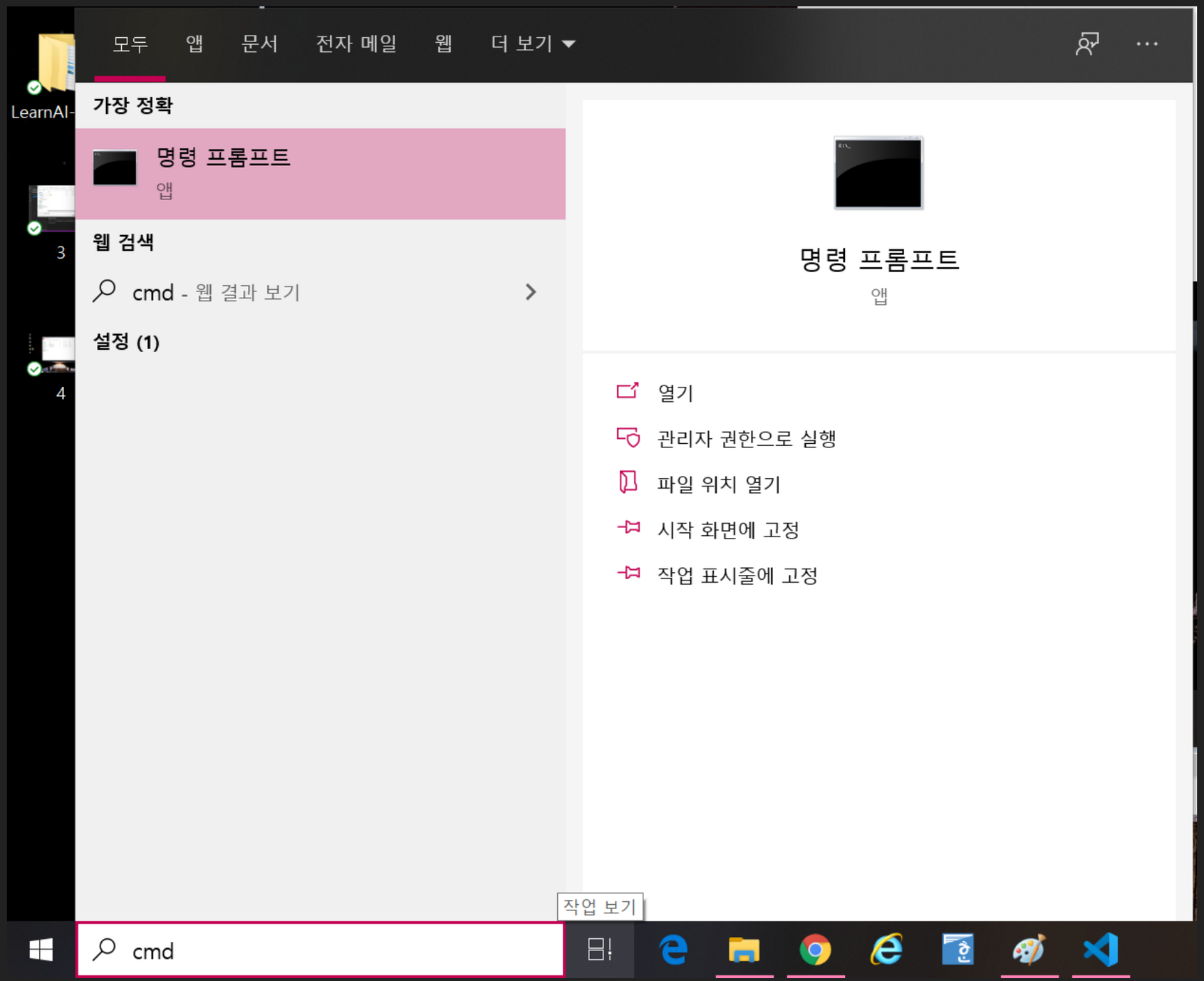
1. INSTALL & SETUP



1. INSTALL & SETUP



INSTALL PYTHON MODULES



INSTALL PYTHON MODULES

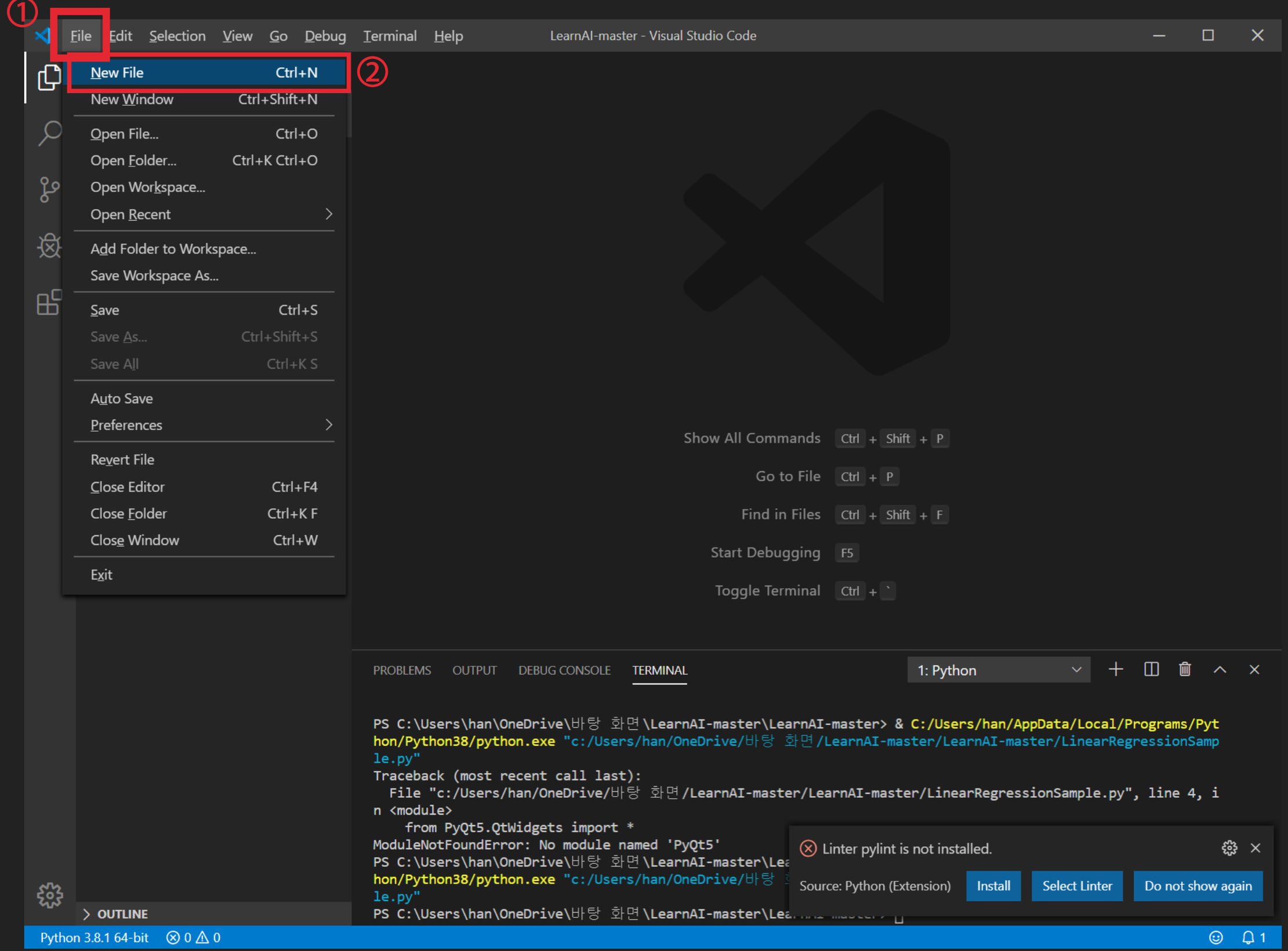
명령 프롬프트

```
Microsoft Windows [Version 10.0.18362.207]
(c) 2019 Microsoft Corporation. All rights reserved.

C:₩Users₩han>
```

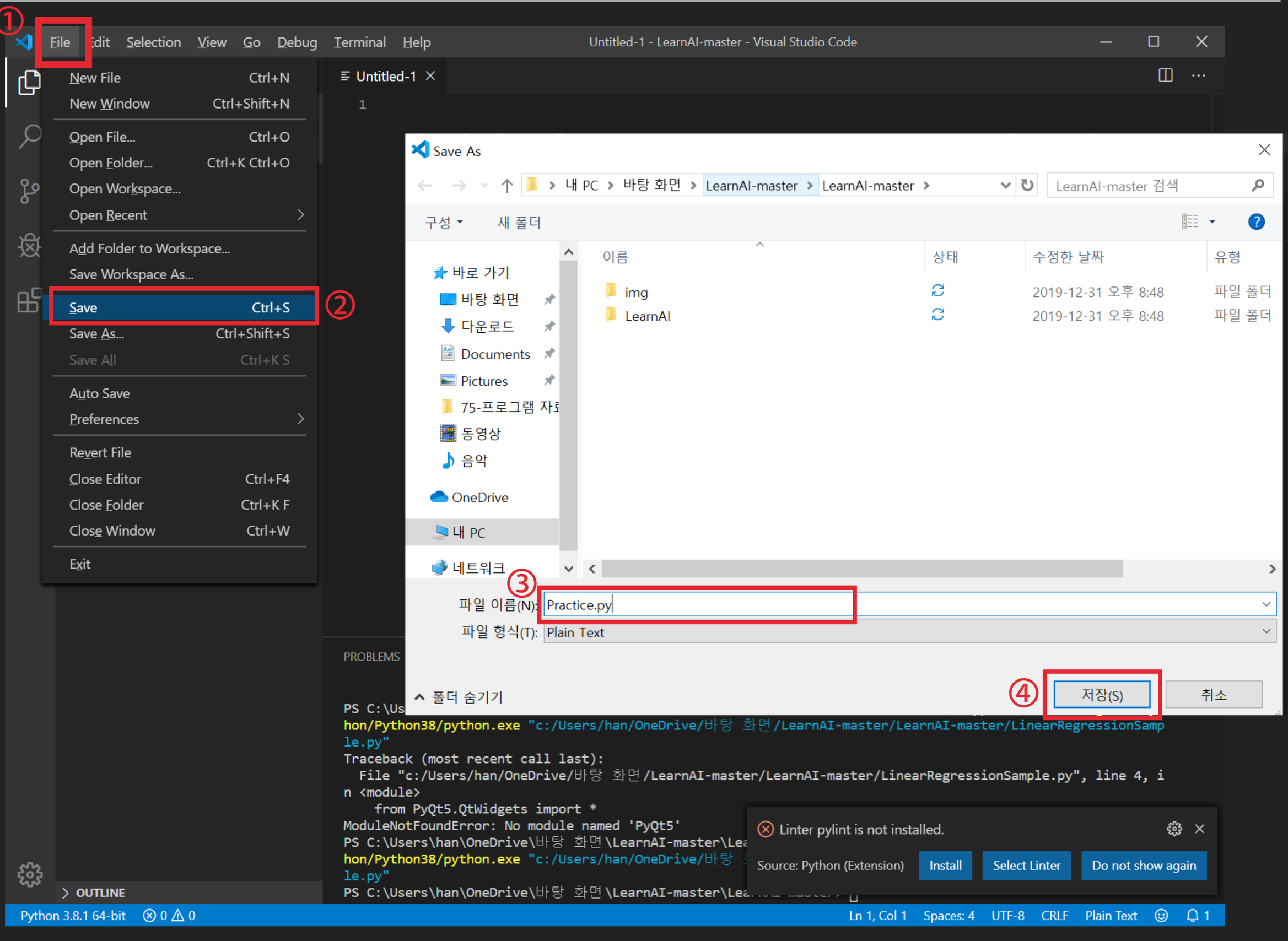
```
pip install PyQt5
pip install numpy
pip install matplotlib
```

대소문자 주의!

1. INSTALL & SETUP



1. INSTALL & SETUP



FOLLOW UP

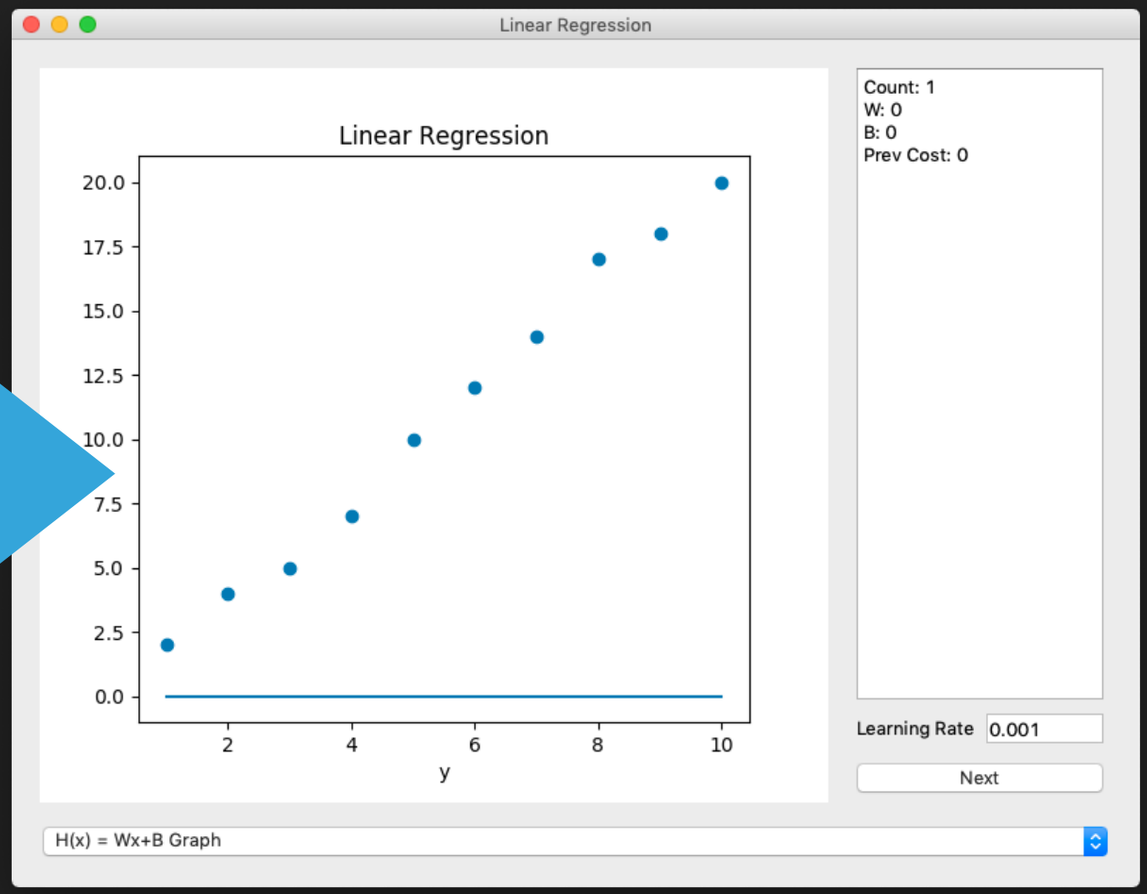
```
import sys
from PyQt5.QtWidgets import *
from LearnAI.LinearRegression import LinearRegression

def Calc():
    reg.updateScreen()


if __name__ == '__main__':
    app = QApplication(sys.argv)

    x = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
    y = [2, 4, 5, 7, 10, 12, 14, 17, 18, 20]
    w = [0]
    b = [0]
    cost = [0]
    learningRate = [0.001]

    reg = LinearRegression()
    reg.setClickEvent(Calc)
    reg.connectParameter(x, y, w, b, cost, learningRate)

    app.exec_()
```

FOLLOW UP

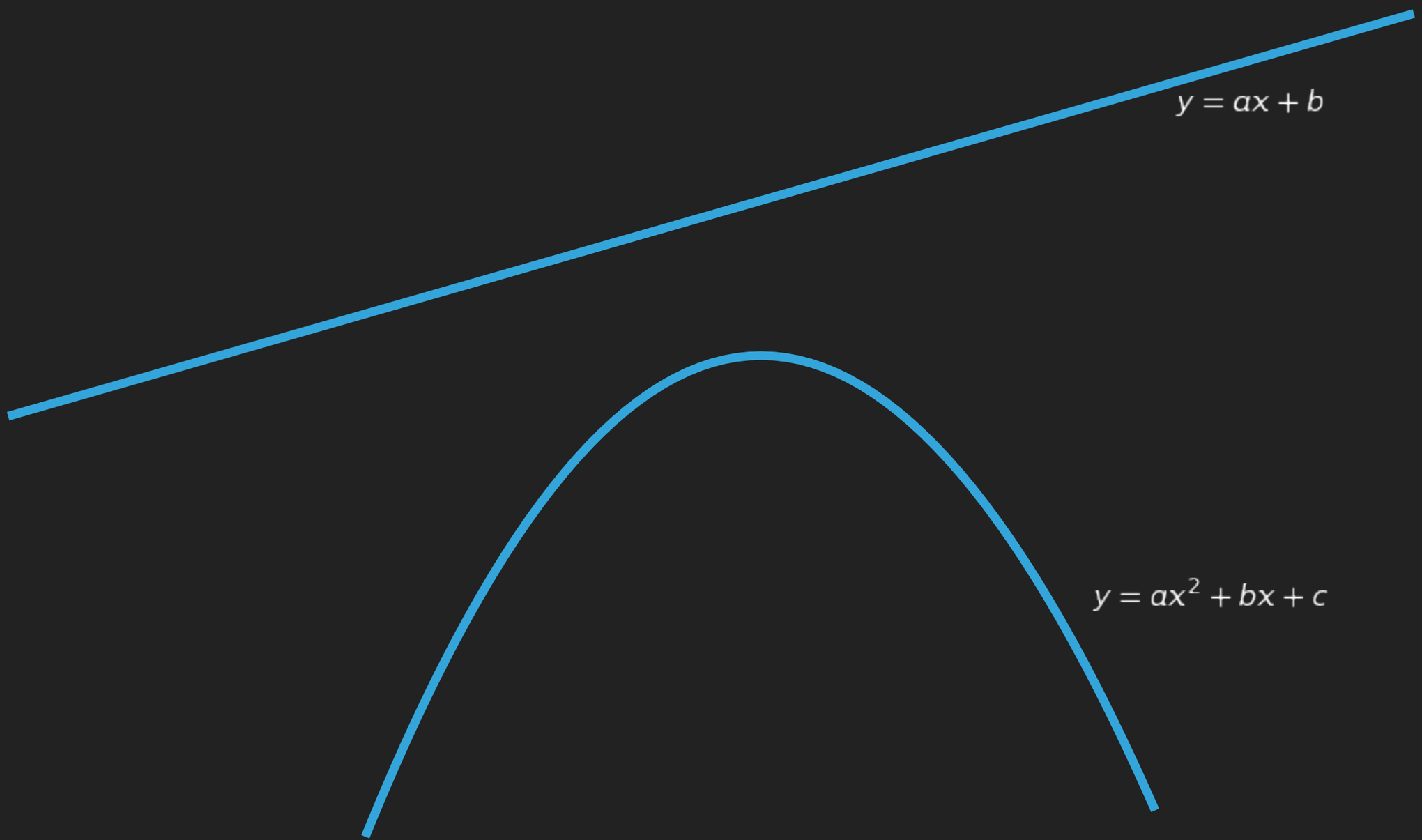
Practice.py — LearnAI

Practice.py

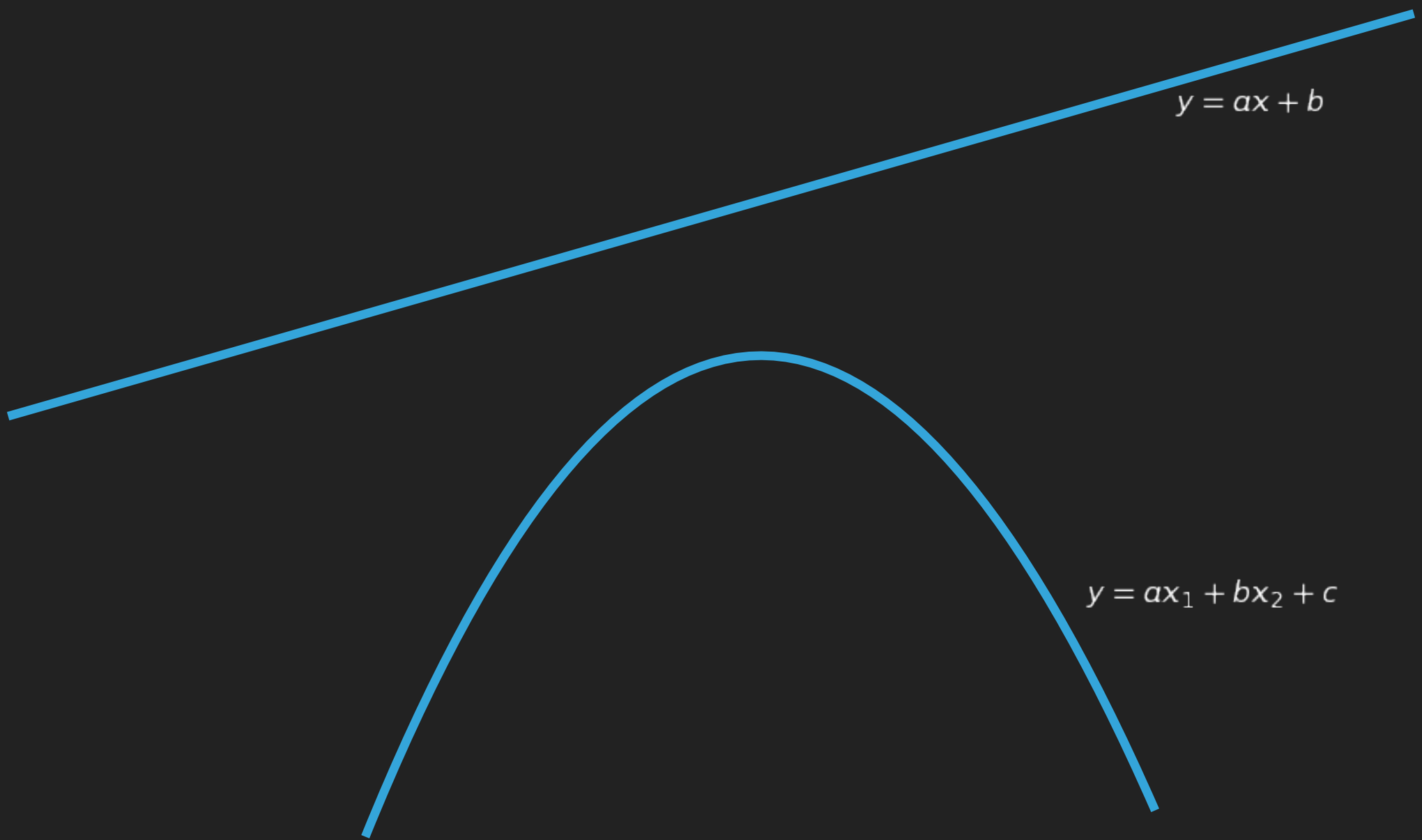
```python
import sys
from PyQt5.QtWidgets import *
from LearnAI.LinearRegression import LinearRegression

def Calc():
    reg.updateScreen()


if __name__ == '__main__':
    app = QApplication(sys.argv)

    x = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
    y = [2, 4, 5, 7, 10, 12, 14, 17, 18, 20]
    w = [0]
    b = [0]
    cost = [0]
    learningRate = [0.001]

    reg = LinearRegression()
    reg.setClickEvent(Calc)
    reg.connectParameter(x, y, w, b, cost, learningRate)

    app.exec_()
```



WHAT IS LINEAR?



WHAT IS LINEAR?



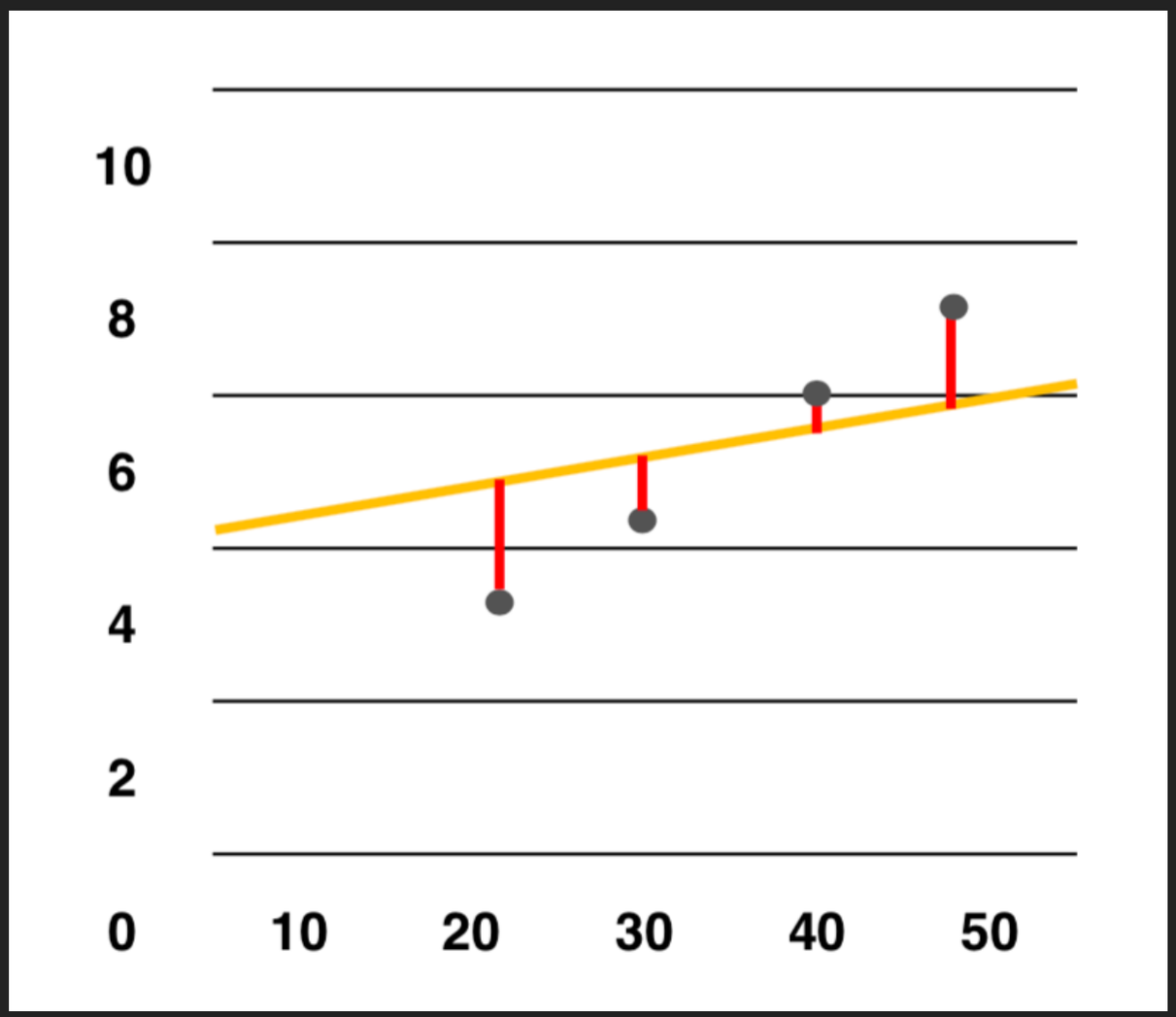
WHY LINEAR?

걷는 걸음 수에 따른 체중 변화?

오늘 평균 기온에 따른 어묵 판매량?

평 수에 따른 월세 비용?

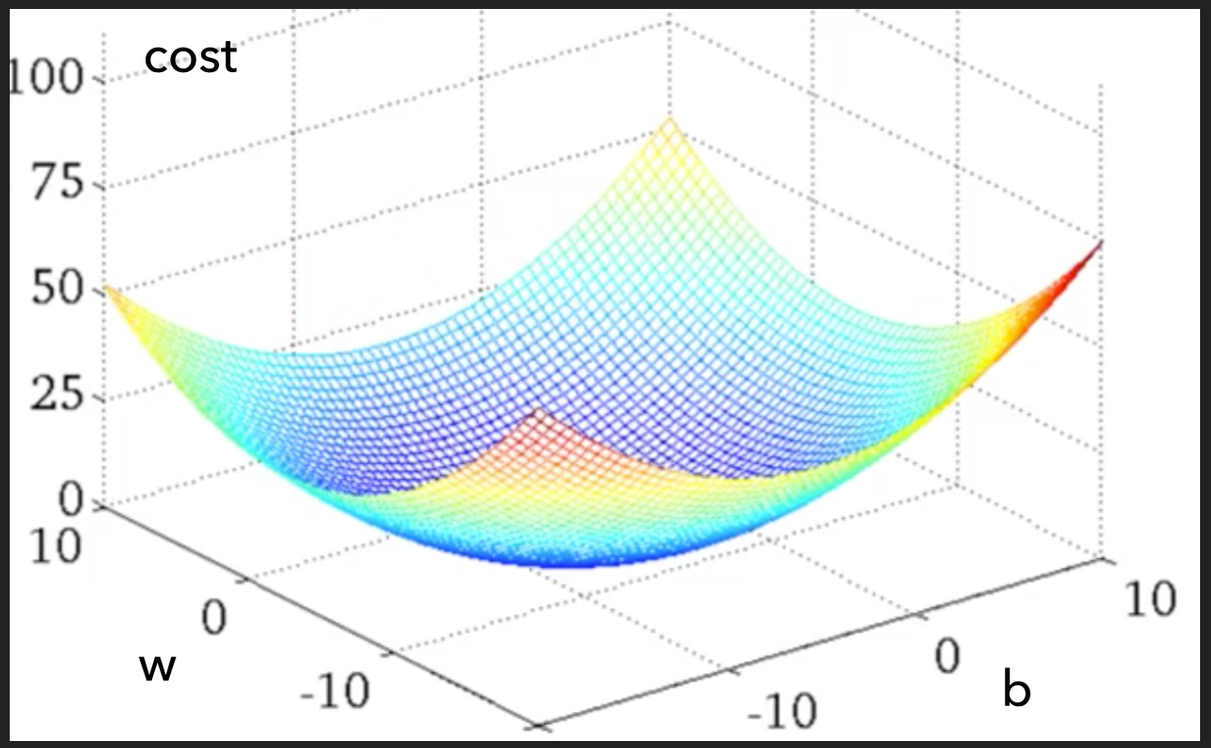
WHY LINEAR?



$$H(x) = wx + b$$

$$cost = \frac{1}{m}\sum_{i=1}^{m}(H(x^{(i)}) - y^{(i)})^2$$

WHY LINEAR?



$$cost(w,b) = \frac{1}{m}\sum_{i=1}^{m}(wx^{(i)} + b - y^{(i)})^2$$

WHY LINEAR?

$$cost(w,b) = \frac{1}{m}\sum_{i=1}^{m}(wx^{(i)} + b - y^{(i)})^2$$

$$\frac{\partial}{\partial w}cost(w,b) = \frac{2}{m}\sum_{i=1}^{m}(wx^{(i)} + b - y^{(i)}) * x^{(i)}$$

$$\frac{\partial}{\partial b}cost(w,b) = \frac{2}{m}\sum_{i=1}^{m}(wx^{(i)} + b - y^{(i)})$$

$$\mathrm{w} := \mathrm{w} - \alpha\frac{\partial}{\partial w}cost(w,b)$$

$$\mathrm{b} := \mathrm{b} - \alpha\frac{\partial}{\partial b}cost(w,b)$$

Must update weights simultaneously

CODE REVIEW

```python
import sys
from PyQt5.QtWidgets import *
from LearnAI.LinearRegression import LinearRegression

def Calc():
    reg.updateScreen()


if __name__ == '__main__':
    app = QApplication(sys.argv)

    x = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
    y = [2, 4, 5, 7, 10, 12, 14, 17, 18, 20]
    w = [0]
    b = [0]
    cost = [0]
    learningRate = [0.001]

    reg = LinearRegression()
    reg.setClickEvent(Calc)
    reg.connectParameter(x, y, w, b, cost, learningRate)

    app.exec_()
```

CHANGING PARAMETERS

```
import random

def Calc():
    w[0] += 0.1;
    b[0] -= 0.1;
    reg.updateScreen()

w = [random.uniform(-3, 3)]
b = [random.uniform(-3, 3)]
```

UPDATE WEIGHT

```
def Calc():
    aw = 0
    for (xi, yi) in zip(x, y):
        aw += (w[0] * xi + b[0] - yi) * xi
    aw /= len(x)
    w[0] -= learningRate[0]*aw

    reg.updateScreen()
```

$$\frac{\partial}{\partial w} cost(w,b) = \frac{2}{m}\sum_{i=1}^{m}(wx^{(i)} + b - y^{(i)}) * x^{(i)}$$

$$\mathrm{w} := \mathrm{w} - \alpha \frac{\partial}{\partial w} cost(w,b)$$

UPDATE WEIGHT

```
def Calc():
    aw = 0
    ab = [          ]
    for (xi, yi) in zip(x, y):
        aw += (w[0] * xi + b[0] - yi) * xi
        ab += [                              ]
    aw /= len(x)
    ab /= [          ]
    w[0] -= learningRate[0]*aw
    b[0] -= [                    ]

    reg.updateScreen()
```

Fill the Blanks

$$\frac{\partial}{\partial b} cost(w,b) = \frac{2}{m} \sum_{i=1}^{m} (wx^{(i)} + b - y^{(i)})$$

$$\mathrm{b} := \mathrm{b} - \alpha \frac{\partial}{\partial b} cost(w,b)$$

UPDATE WEIGHT

```
def Calc():
    aw = 0
    ab = 0
    for (xi, yi) in zip(x, y):
        aw += (w[0] * xi + b[0] - yi) * xi
        ab += (w[0] * xi + b[0] - yi)
    aw /= len(x)
    ab /= len(x)
    w[0] -= learningRate[0]*aw
    b[0] -= learningRate[0]*ab

    reg.updateScreen()
```

$$\frac{\partial}{\partial b} cost(w,b) = \frac{2}{m} \sum_{i=1}^{m} (wx^{(i)} + b - y^{(i)})$$

$$\mathrm{b} := \mathrm{b} - \alpha \frac{\partial}{\partial b} cost(w,b)$$

UPDATE COST

```
def Calc():
    aw = 0
    ab = 0
    cost[0] = 
    for (xi, yi) in zip(x, y):
        aw += (w[0] * xi + b[0] - yi) * xi
        ab += (w[0] * xi + b[0] - yi)
        cost[0] += 
    aw /= len(x)
    ab /= len(x)
    cost[0] /= 
    w[0] -= learningRate[0]*aw
    b[0] -= learningRate[0]*ab

    reg.updateScreen()
```

Fill the Blanks

$$cost(w,b) = \frac{1}{m}\sum_{i=1}^{m}(wx^{(i)} + b - y^{(i)})^2$$

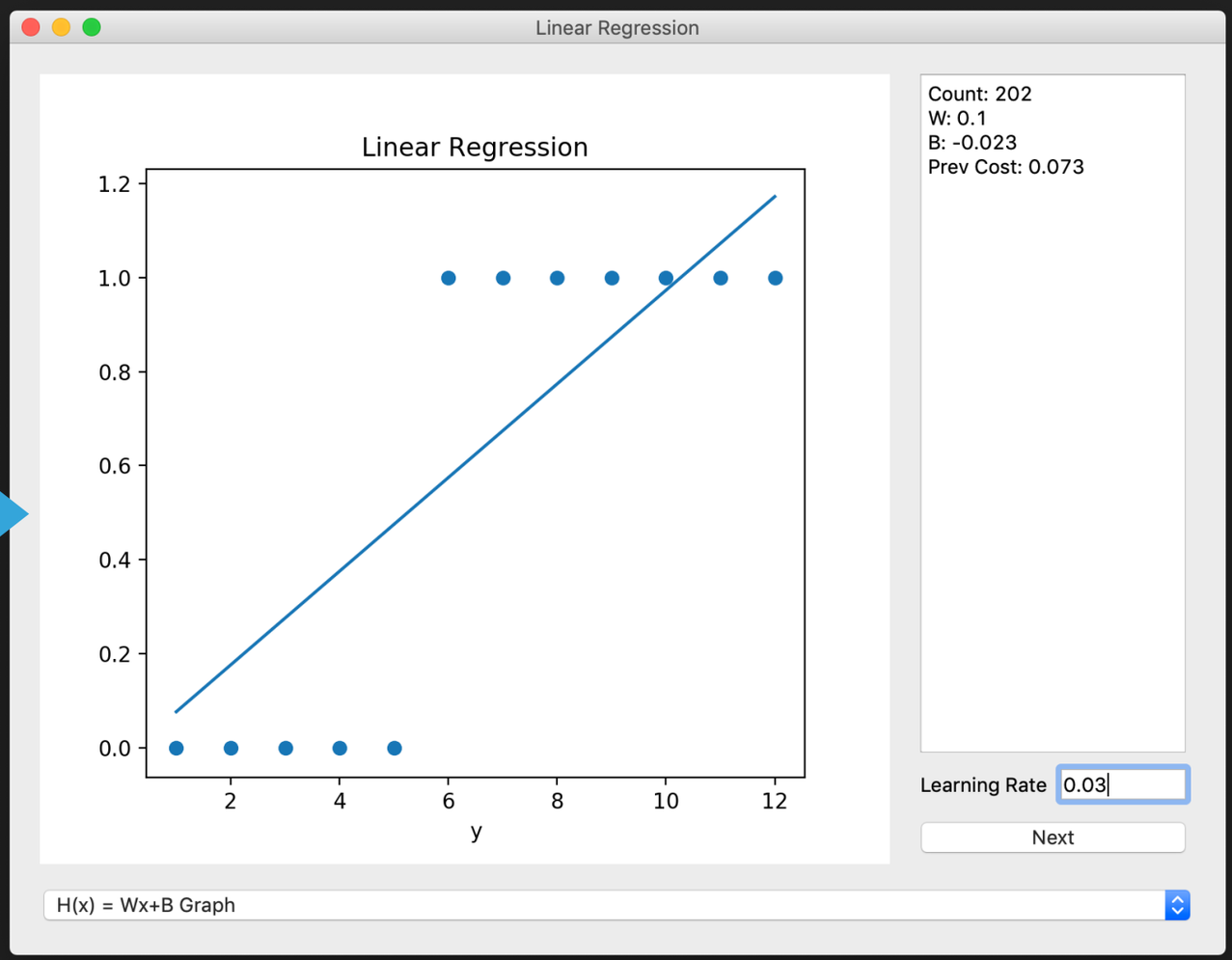
UPDATE COST

```
def Calc():
    aw = 0
    ab = 0
    cost[0] = 0
    for (xi, yi) in zip(x, y):
        aw += (w[0] * xi + b[0] - yi) * xi
        ab += (w[0] * xi + b[0] - yi)
        cost[0] += (w[0] * xi + b[0] - yi) ** 2
    aw /= len(x)
    ab /= len(x)
    cost[0] /= len(x)
    w[0] -= learningRate[0]*aw
    b[0] -= learningRate[0]*ab

    reg.updateScreen()
```

$$cost(w,b) = \frac{1}{m}\sum_{i=1}^{m}(wx^{(i)} + b - y^{(i)})^2$$

WHAT IF DATA ARE 0 OR 1?

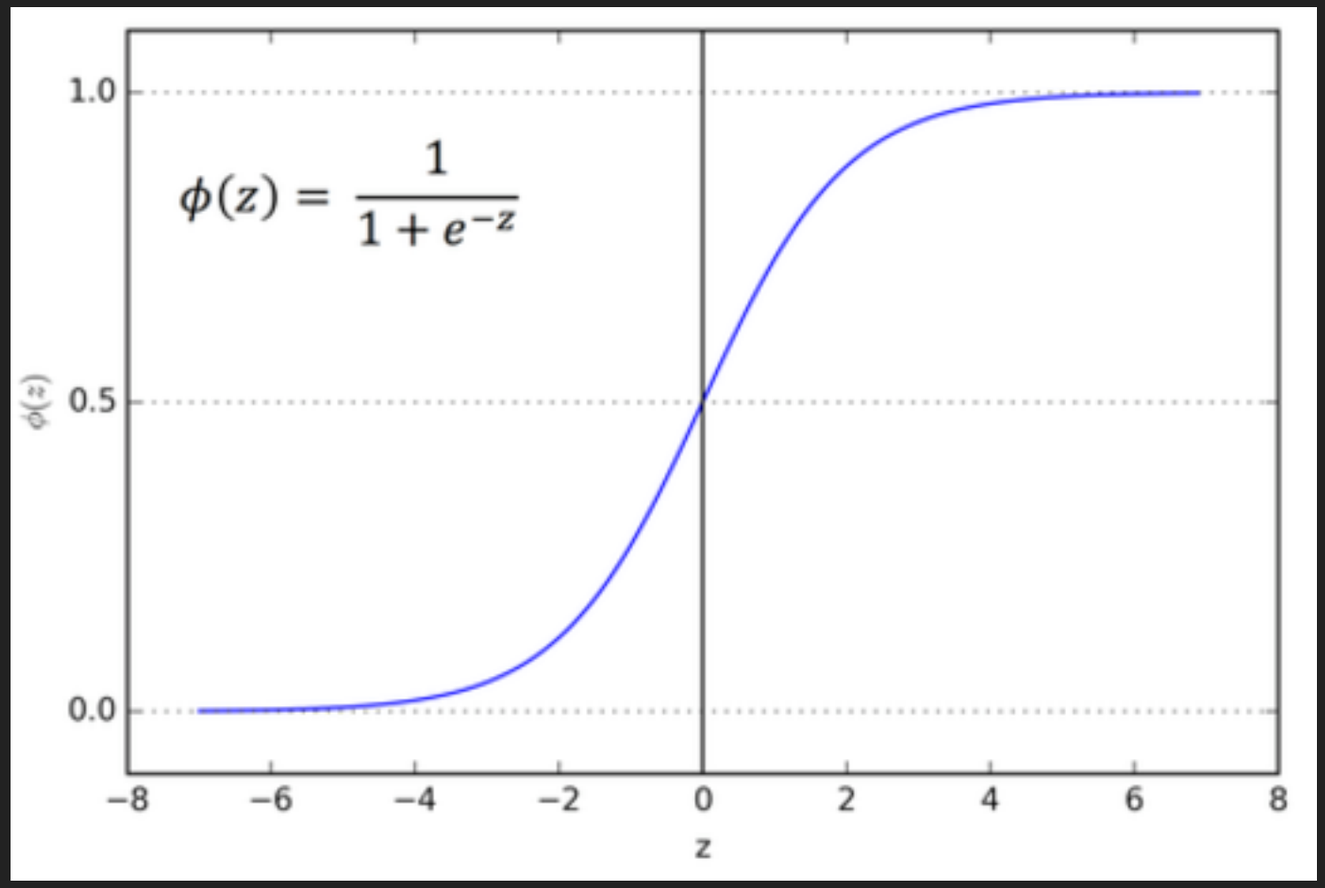
```
x = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
y = [0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1]

w = [random.uniform(-1, 1)]
b = [random.uniform(-1, 1)]
```



FOLLOW UP

```
import sys
import random
from PyQt5.QtWidgets import *
from LearnAI.LogisticRegression import
 LogisticRegression

def Calc():
    reg.updateScreen()

if __name__ == '__main__':
    app = QApplication(sys.argv)

    x = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
    y = [0, 0, 0, 0, 0, 1, 1, 1, 1, 1]
    w = [random.uniform(-3, 3)]
    b = [random.uniform(-3, 3)]
    cost = [0]
    learningRate = [0.1]

    reg = LogisticRegression()
    reg.setClickEvent(Calc)
    reg.connectParameter(x, y, w, b, cost, learningRate)

    app.exec_()
```

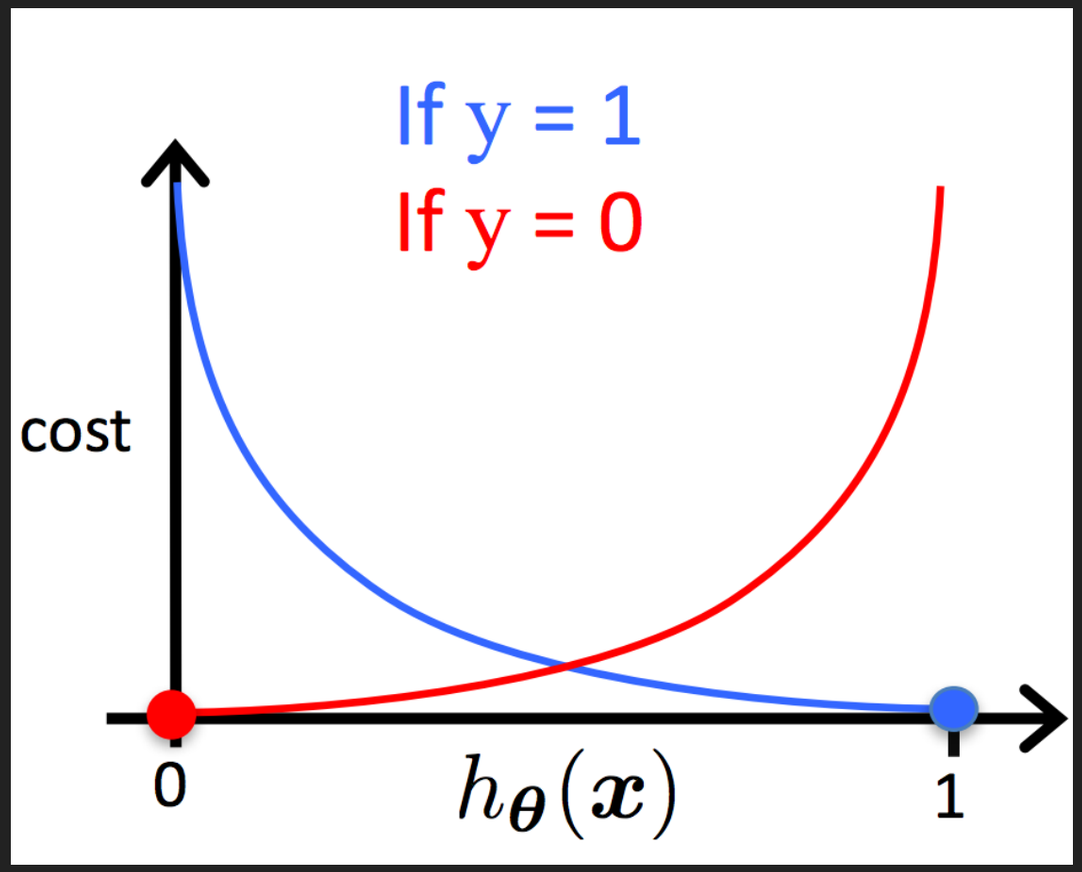
HYPOTHESIS



$$z = wx + b$$

$$H(z) = \frac{1}{1+e^{-z}}$$

COST FUNCTION



$$z = wx + b$$

$$H(x) = \sigma(z) = \frac{1}{1 + e^{-z}}$$

$$cost(w, b) = \frac{1}{m} \sum_{i=1}^{m} c(H(x), y)$$

$$c(H(x), y) = \begin{cases} -log(H(x)) & y = 1 \\ -log(1 - H(x)) & y = 0 \end{cases}$$

COST FUNCTION

$$\sigma(z)' = \sigma(z)(1 - \sigma(z))$$

$$\frac{\partial}{\partial w} c(H(x), y) = \begin{cases} -\frac{H(x)'}{H(x)} & y = 1 \\ \frac{H(x)'}{1-H(x)} & y = 0 \end{cases}$$

$$= \begin{cases} (H(x) - 1)x & y = 1 \\ H(x)x & y = 0 \end{cases}$$

$$= \begin{cases} (\sigma(z) - 1)x & y = 1 \\ \sigma(z)x & y = 0 \end{cases}$$

$$\frac{\partial}{\partial b} c(H(x), y) = \begin{cases} -\frac{H(x)'}{H(x)} & y = 1 \\ \frac{H(x)'}{1-H(x)} & y = 0 \end{cases}$$

$$= \begin{cases} (H(x) - 1) & y = 1 \\ H(x) & y = 0 \end{cases}$$

$$= \begin{cases} (\sigma(z) - 1) & y = 1 \\ \sigma(z) & y = 0 \end{cases}$$

COST FUNCTION

$$\frac{\partial}{\partial w} cost(w,b) = \sum_{i=1}^{m} \begin{cases} (\sigma(wx+b)-1)x & y=1 \\ \sigma(wx+b)x & y=0 \end{cases}$$

$$\frac{\partial}{\partial b} cost(w,b) = \sum_{i=1}^{m} \begin{cases} \sigma(wx+b)-1 & y=1 \\ \sigma(wx+b) & y=0 \end{cases}$$

UPDATE WEIGHTS

```
import math

def sigmoid(z):
    return 1.0/(1.0+math.exp(-z))

def Calc():
    aw = 0.0
    for (xi, yi) in zip(x, y):
        z = w[0]*xi+b[0]
        s = sigmoid(z)
        if yi == 0:
            aw += s*xi
        else:
            aw += (s-1)*xi
    aw /= len(x)
    w[0] -= learningRate[0]*aw

    reg.updateScreen()
```

$$\frac{\partial}{\partial w} cost(w,b) = \sum_{i=1}^{m} \begin{cases} (\sigma(wx+b)-1)x & y=1 \\ \sigma(wx+b)x & y=0 \end{cases}$$

$$\mathrm{w} := \mathrm{w} - \alpha \frac{\partial}{\partial w} cost(w,b)$$

UPDATE WEIGHTS

```
def Calc():
    aw = 0.0
    ab = [          ]
    for (xi, yi) in zip(x, y):
        z = w[0]*xi+b[0]
        s = sigmoid(z)
        if yi == 0:
            aw += s*xi
            ab += [  ]
        else:
            aw += (s-1)*xi
            ab += [      ]
    aw /= len(x)
    ab /= [      ]
    w[0] -= learningRate[0]*aw
    b[0] -= [                ]

    reg.updateScreen()
```

Fill the Blanks

$$\frac{\partial}{\partial b} cost(w,b) = \sum_{i=1}^{m} \begin{cases} \sigma(wx+b) - 1 & y = 1 \\ \sigma(wx+b) & y = 0 \end{cases}$$

$$\mathrm{b} := \mathrm{b} - \alpha \frac{\partial}{\partial b} cost(w,b)$$

UPDATE WEIGHTS

```
def Calc():
    aw = 0.0
    ab = 0.0
    for (xi, yi) in zip(x, y):
        z = w[0]*xi+b[0]
        s = sigmoid(z)
        if yi == 0:
            aw += s*xi
            ab += s
        else:
            aw += (s-1)*xi
            ab += (s-1)
    aw /= len(x)
    ab /= len(x)
    w[0] -= learningRate[0]*aw
    b[0] -= learningRate[0]*ab

    reg.updateScreen()
```

$$\frac{\partial}{\partial b} cost(w,b) = \sum_{i=1}^{m} \begin{cases} \sigma(wx+b) - 1 & y = 1 \\ \sigma(wx+b) & y = 0 \end{cases}$$

$$\mathrm{b} := \mathrm{b} - \alpha \frac{\partial}{\partial b} cost(w,b)$$

UPDATE COST

```
def Calc():
    aw = 0.0
    ab = 0.0
    cost[0] = 
    for (xi, yi) in zip(x, y):
        z = w[0]*xi+b[0]
        s = sigmoid(z)
        if yi == 0:
            aw += s*xi
            ab += s
            cost[0] -= 
        else:
            aw += (s-1)*xi
            ab += (s-1)
            cost[0] -= 
    aw /= len(x)
    ab /= len(x)
    cost[0] /= 
    w[0] -= learningRate[0]*aw
    b[0] -= learningRate[0]*ab

    reg.updateScreen()
```

Fill the Blanks

$$cost(w,b) = \frac{1}{m}\sum_{i=1}^{m} c(H(x),y)$$

$$c(H(x),y) = \begin{cases} -log(H(x)) & y = 1 \\ -log(1-H(x)) & y = 0 \end{cases}$$

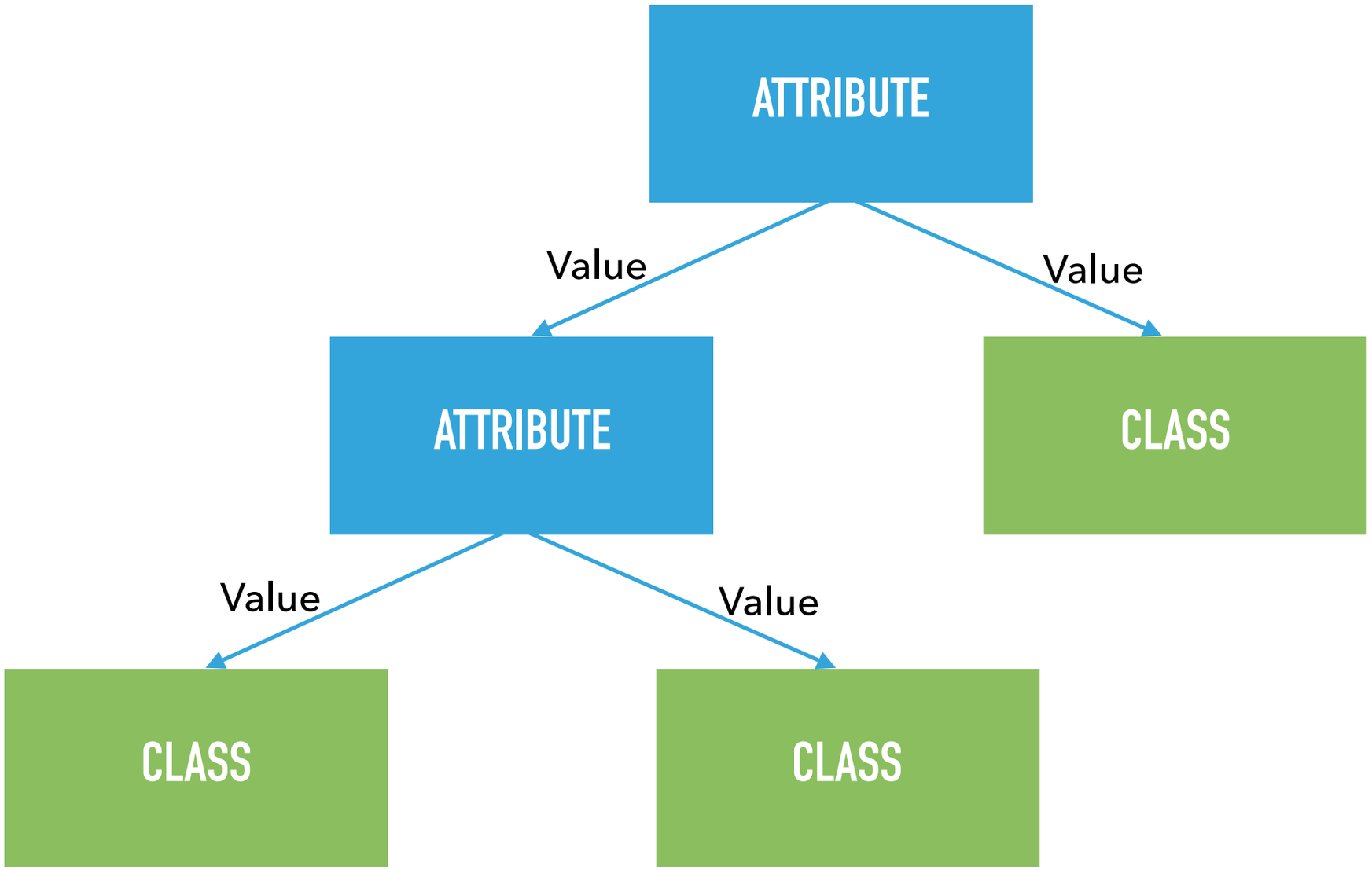
UPDATE COST

```
def Calc():
    aw = 0.0
    ab = 0.0
    cost[0] = 0
    for (xi, yi) in zip(x, y):
        z = w[0]*xi+b[0]
        s = sigmoid(z)
        if yi == 0:
            aw += s*xi
            ab += s
            cost[0] -= math.log(1-s)
        else:
            aw += (s-1)*xi
            ab += (s-1)
            cost[0] -= math.log(s)
    aw /= len(x)
    ab /= len(x)
    cost[0] /= len(x)
    w[0] -= learningRate[0]*aw
    b[0] -= learningRate[0]*ab

    reg.updateScreen()
```

$$cost(w,b) = \frac{1}{m}\sum_{i=1}^{m} c(H(x),y)$$

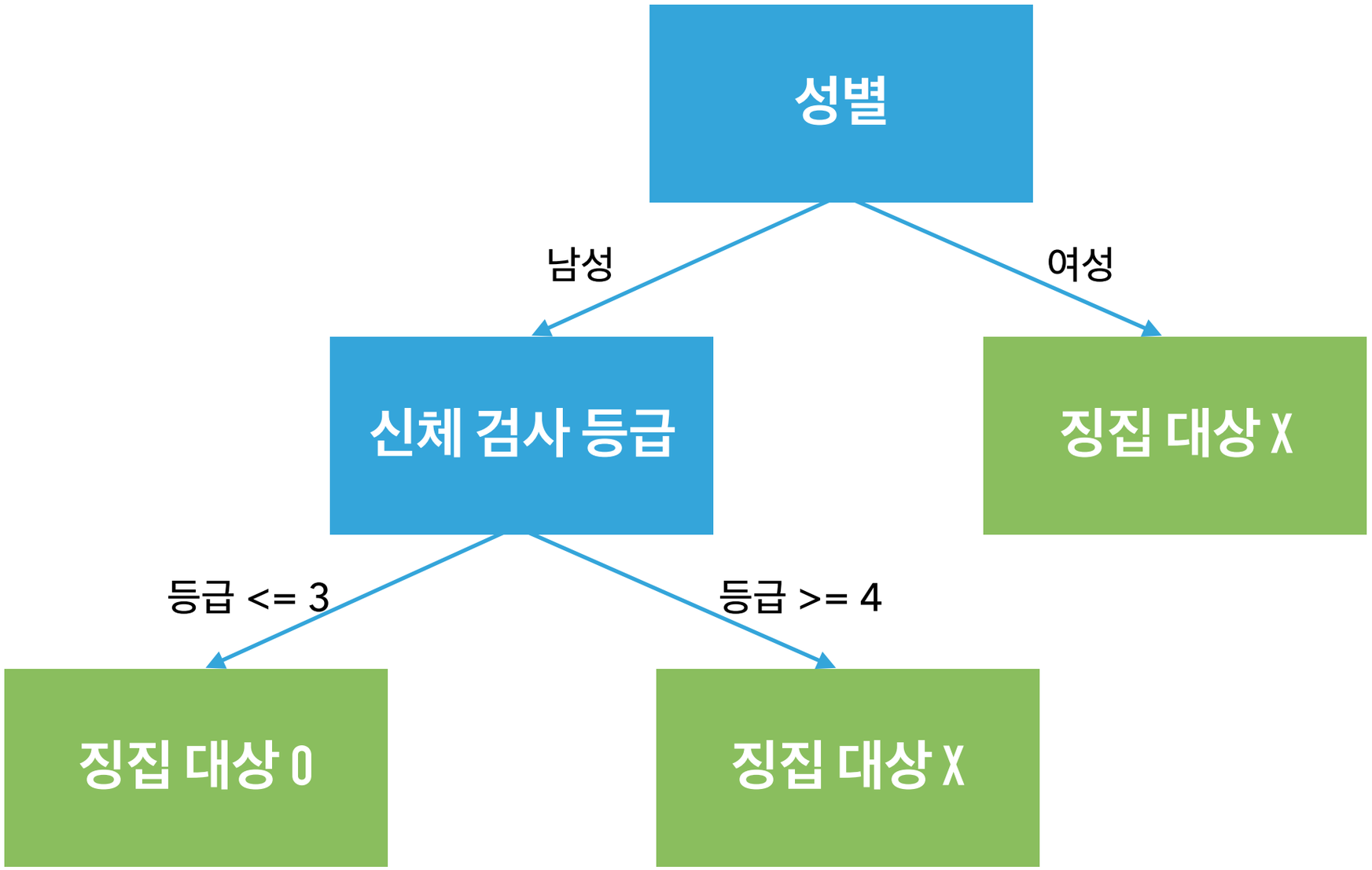
$$c(H(x),y) = \begin{cases} -log(H(x)) & y = 1 \\ -log(1-H(x)) & y = 0 \end{cases}$$

WHAT IS DECISION TREE?



Attribute = {Value1, Value2, Value3, …}

WHAT IS DECISION TREE?

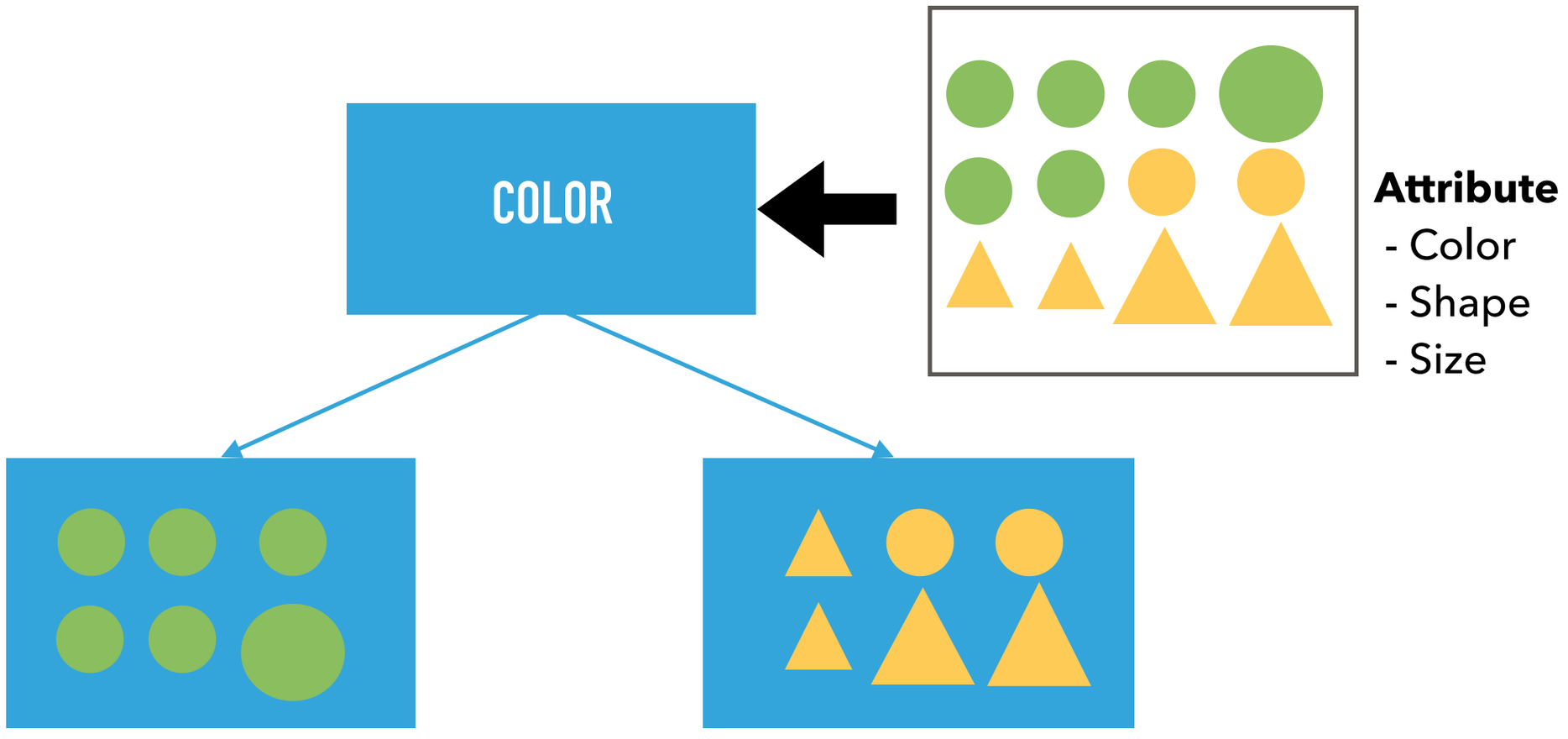


성별 = {남성, 여성}

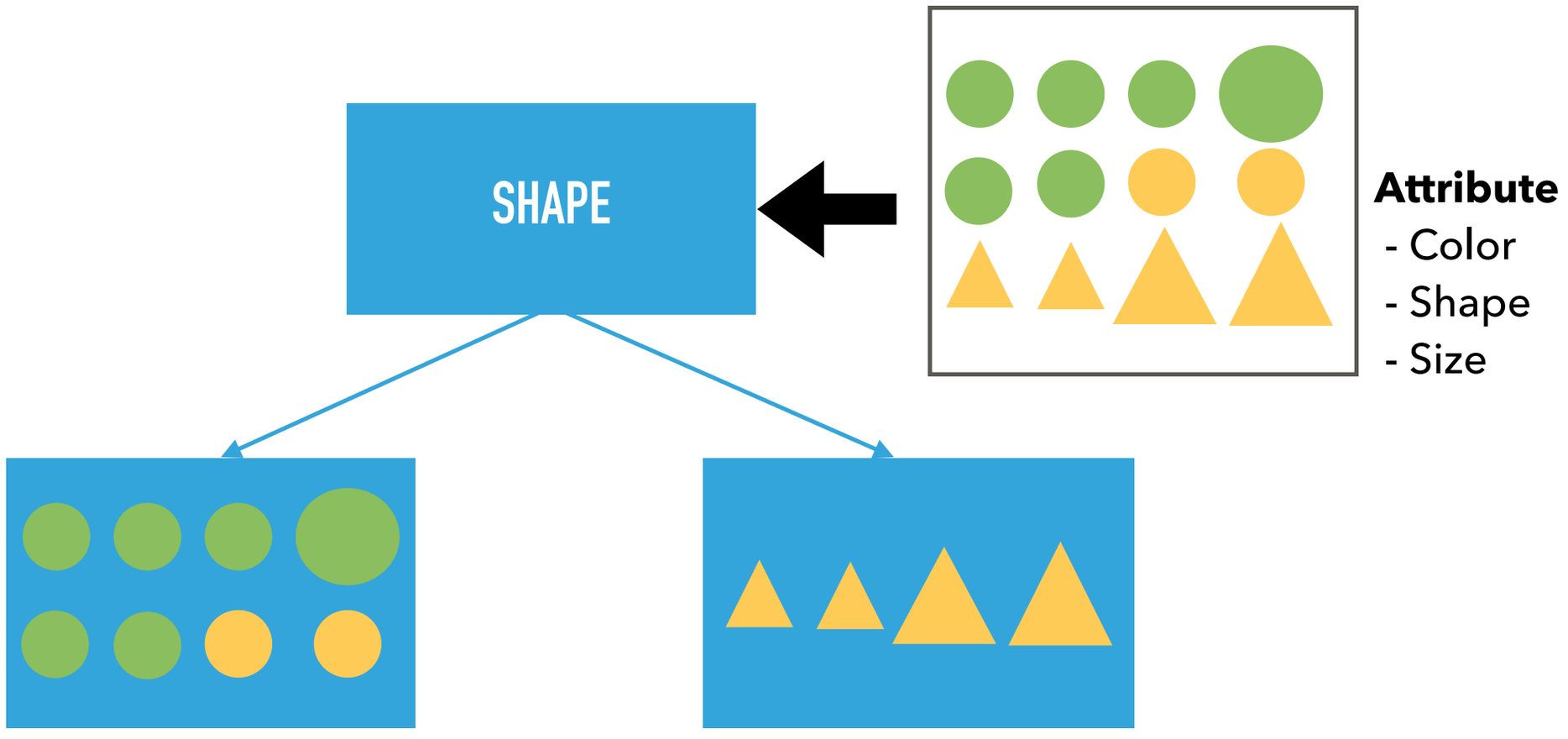
등급 = {1, 2, 3, 4, 5, 6}

HOW TO MAKE DECISION TREE?

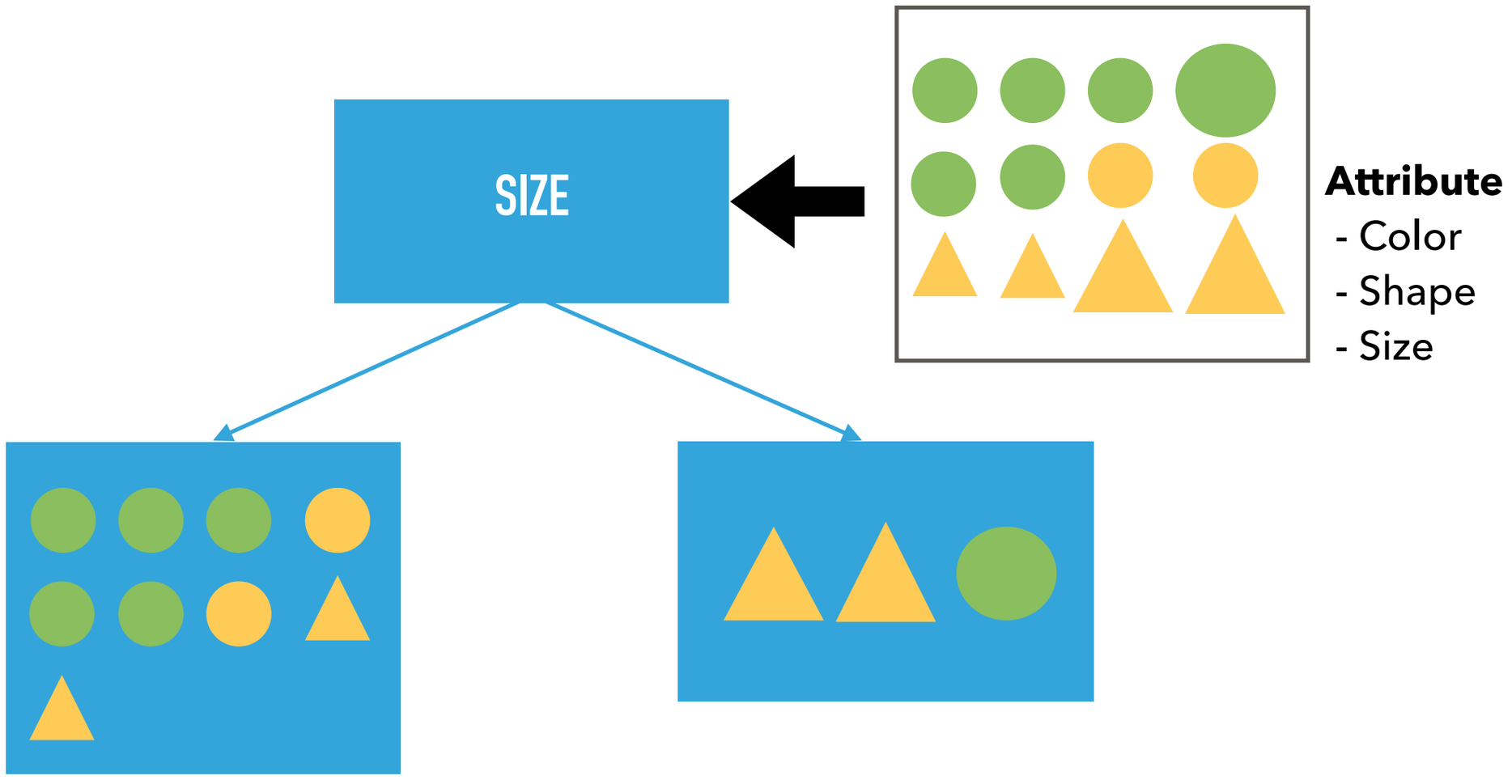
HOW TO MAKE DECISION TREE?



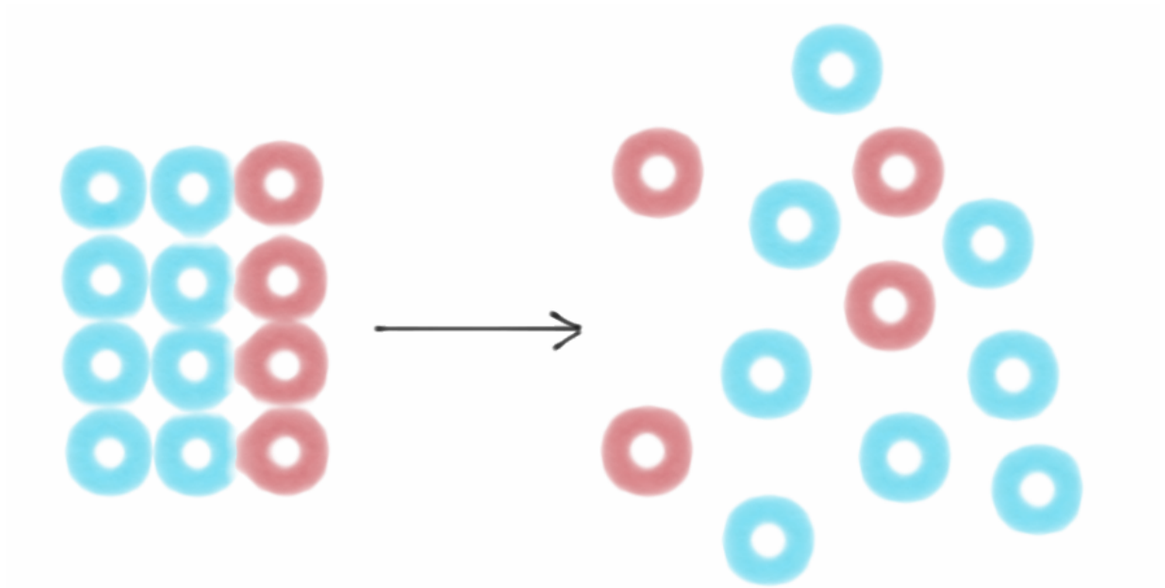
HOW TO MAKE DECISION TREE?



HOW TO MAKE DECISION TREE?



HOW TO MAKE DECISION TREE?



$$Entropy(S) = -p_+ log_2 p_+ - p_- log_2 p_-$$

테스트 샘플의 엔트로피가 0에 가까울수록 분류가 잘 됨

HOW TO MAKE DECISION TREE?

$$Entropy(S) = -p_+ log_2 p_+ - p_- log_2 p_-$$

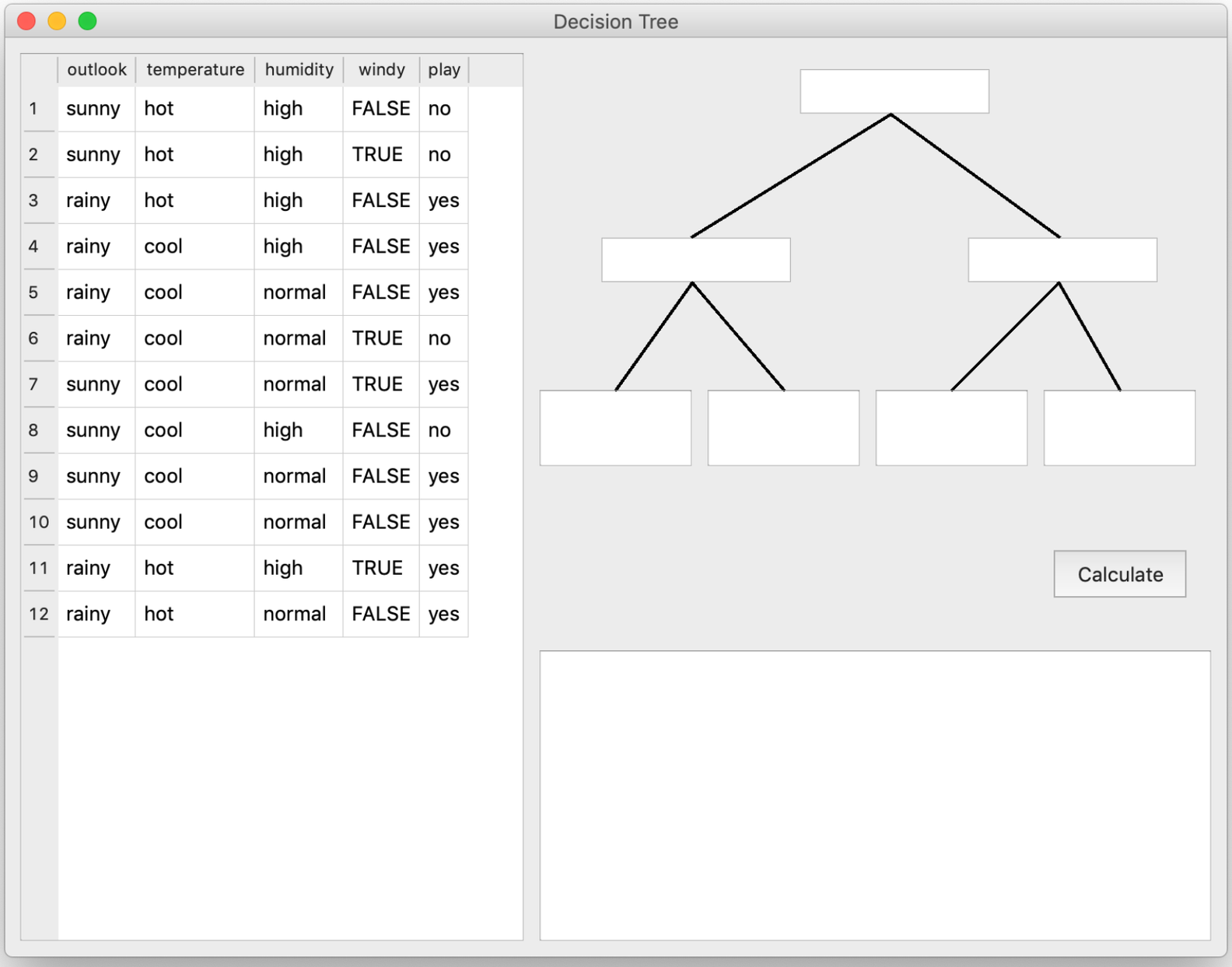
$$Gain(S, A) = Entropy(S) - \sum_{v \in Value(A)} \frac{S_v}{S} Entropy(S_v)$$

어떤 Attribute를 선택했을 때 엔트로피 얼마나 줄어드는지?

FOLLOW UP

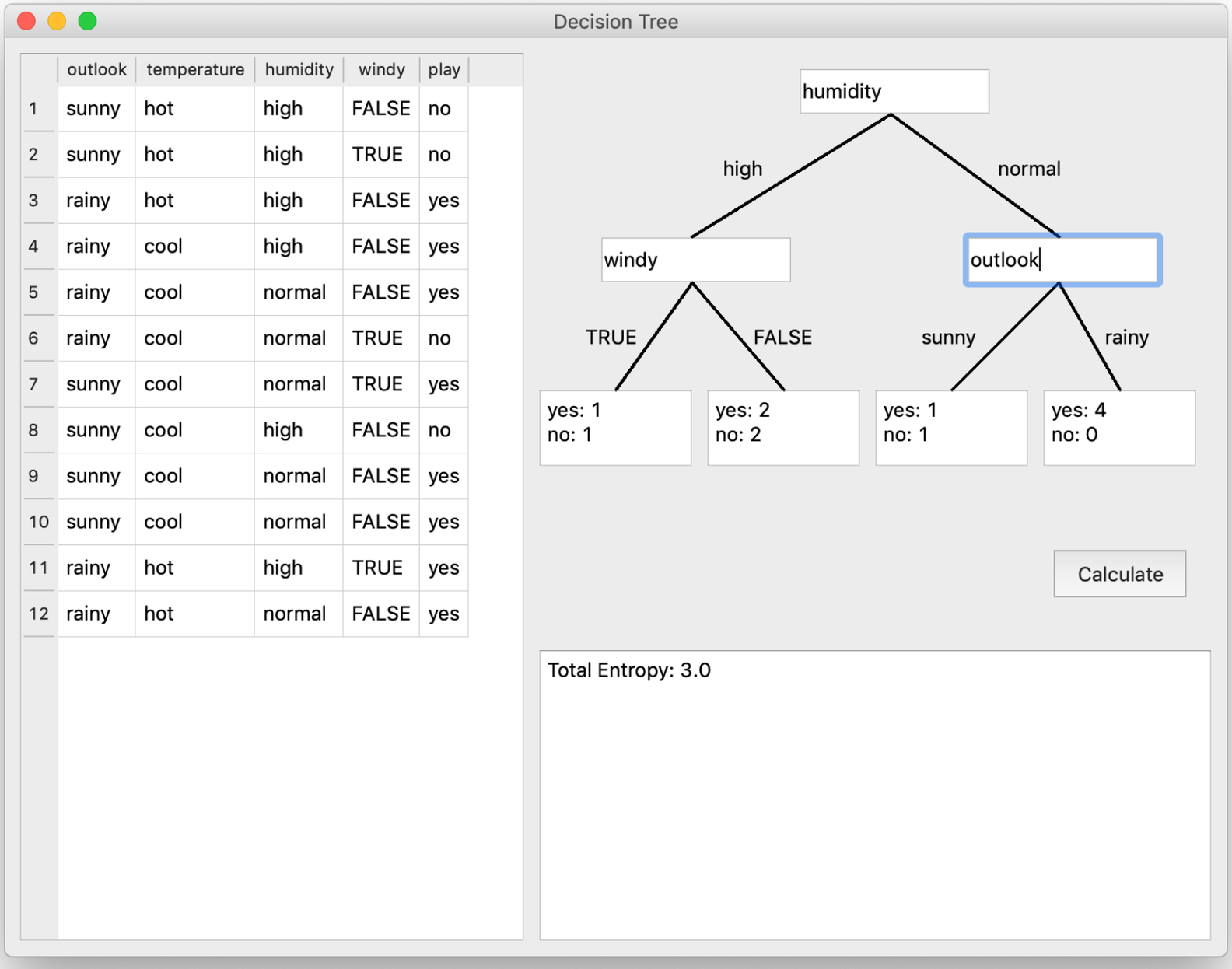
```
import sys
from PyQt5.QtWidgets import *
from LearnAI.DecisionTree import DecisionTree


if __name__ == '__main__':
    app = QApplication(sys.argv)

    dt = DecisionTree()

    app.exec_()
```

FOLLOW UP

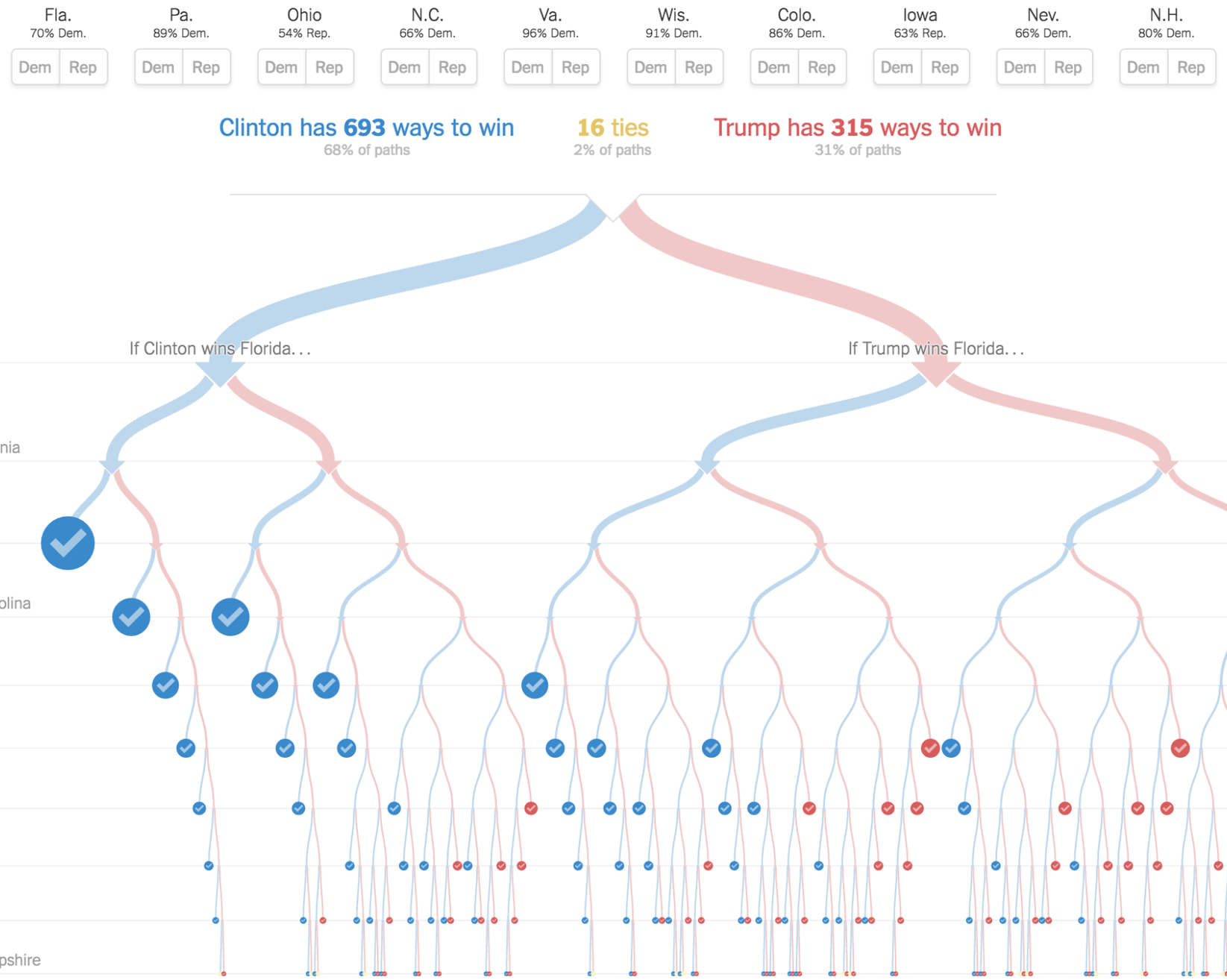


Input the Attributes and Make Entropy smaller!

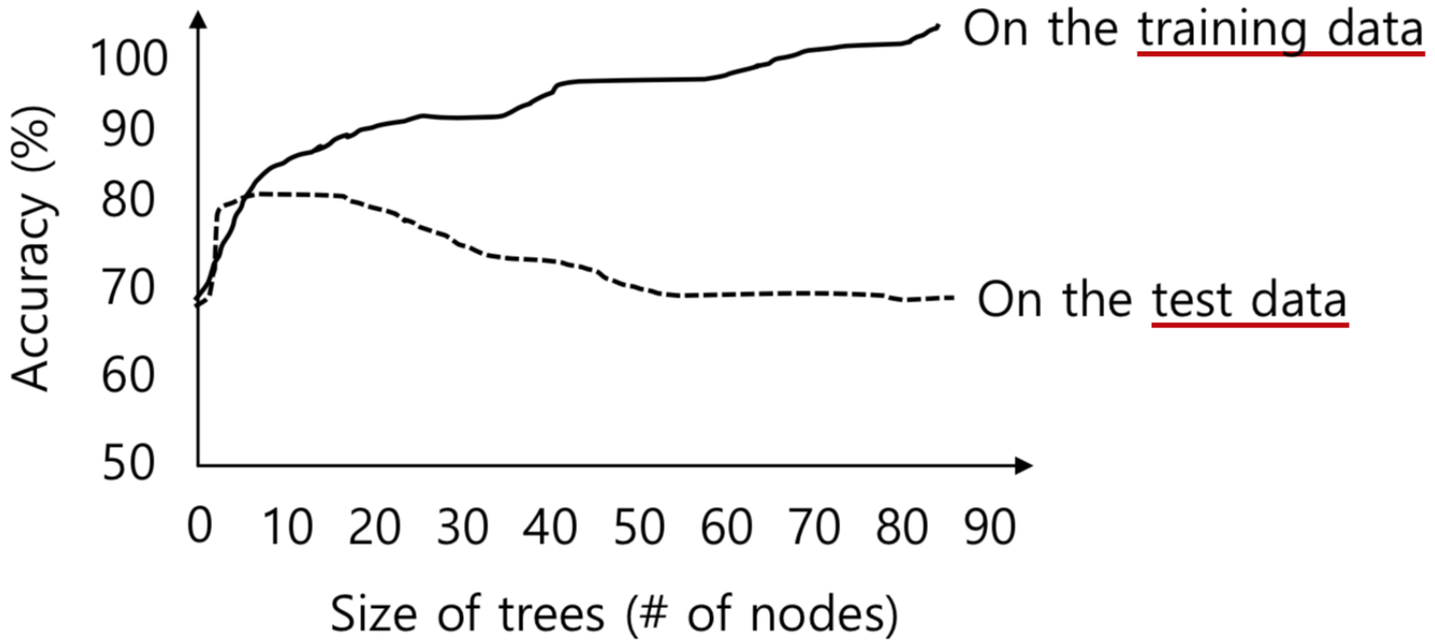
FOLLOW UP


Decision Tree
outlook temperature humidity windy play
1 sunny hot high FALSE no
2 sunny hot high TRUE no
3 rainy hot high FALSE yes
4 rainy cool high FALSE yes
5 rainy cool normal FALSE yes
6 rainy cool normal TRUE no
7 sunny cool normal TRUE yes
8 sunny cool high FALSE no
9 sunny cool normal FALSE yes
10 sunny cool normal FALSE yes
11 rainy hot high TRUE yes
12 rainy hot normal FALSE yes
humidity
high
normal
windy
outlook
TRUE
FALSE
sunny
rainy
yes: 1
no: 1
yes: 2
no: 2
yes: 1
no: 1
yes: 4
no: 0
Calculate
Total Entropy: 3.0

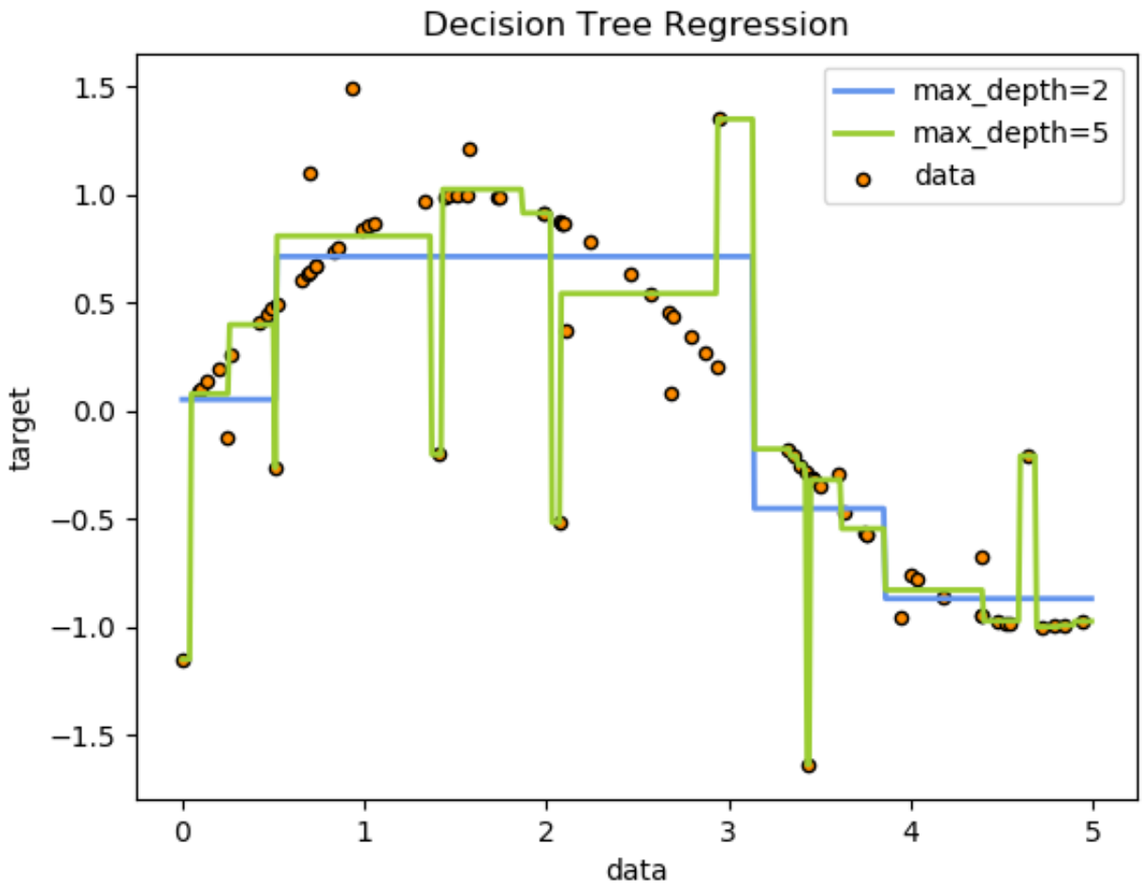
DEEP DECISION TREE?



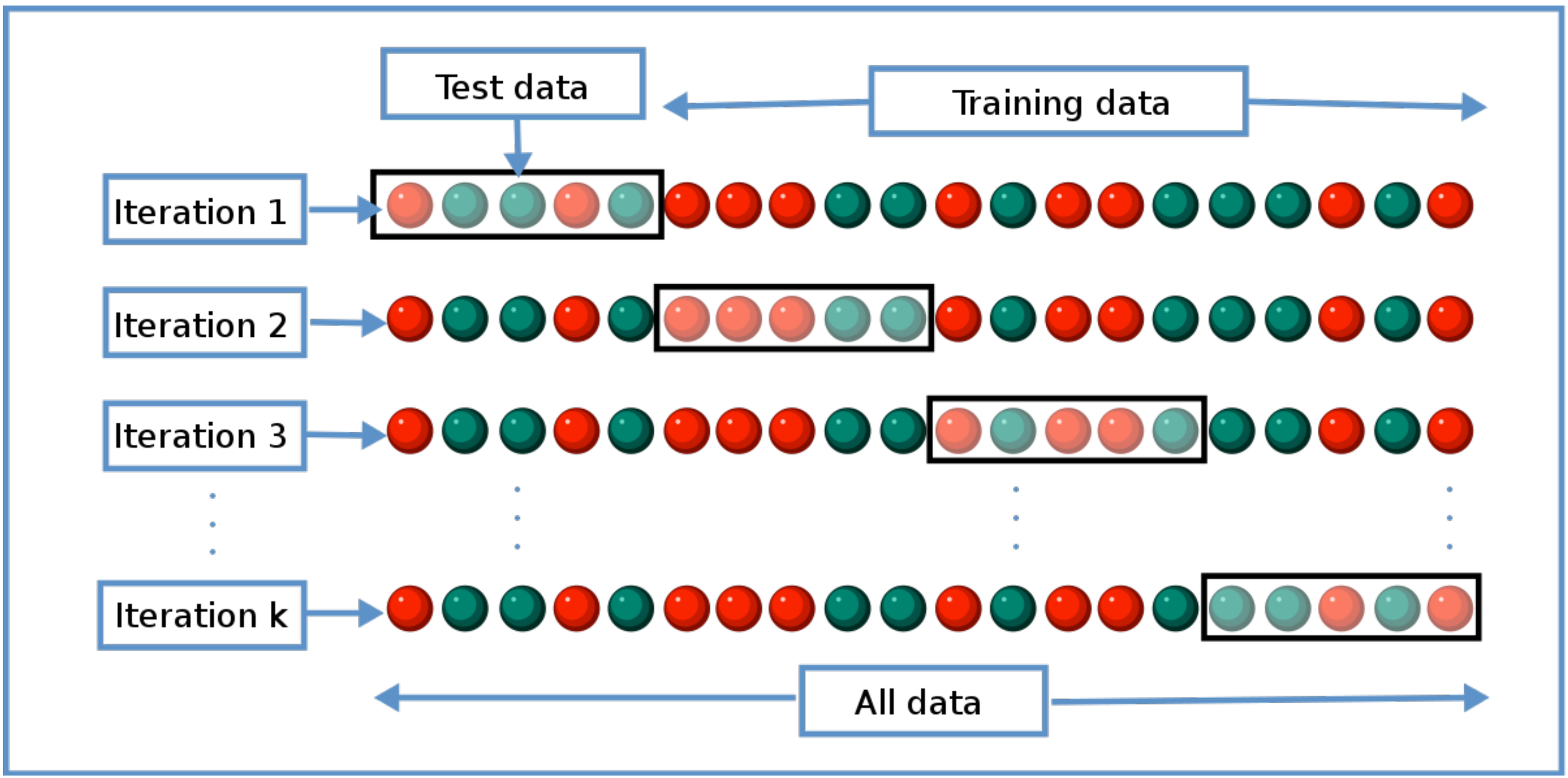
DEEP DECISION TREE?



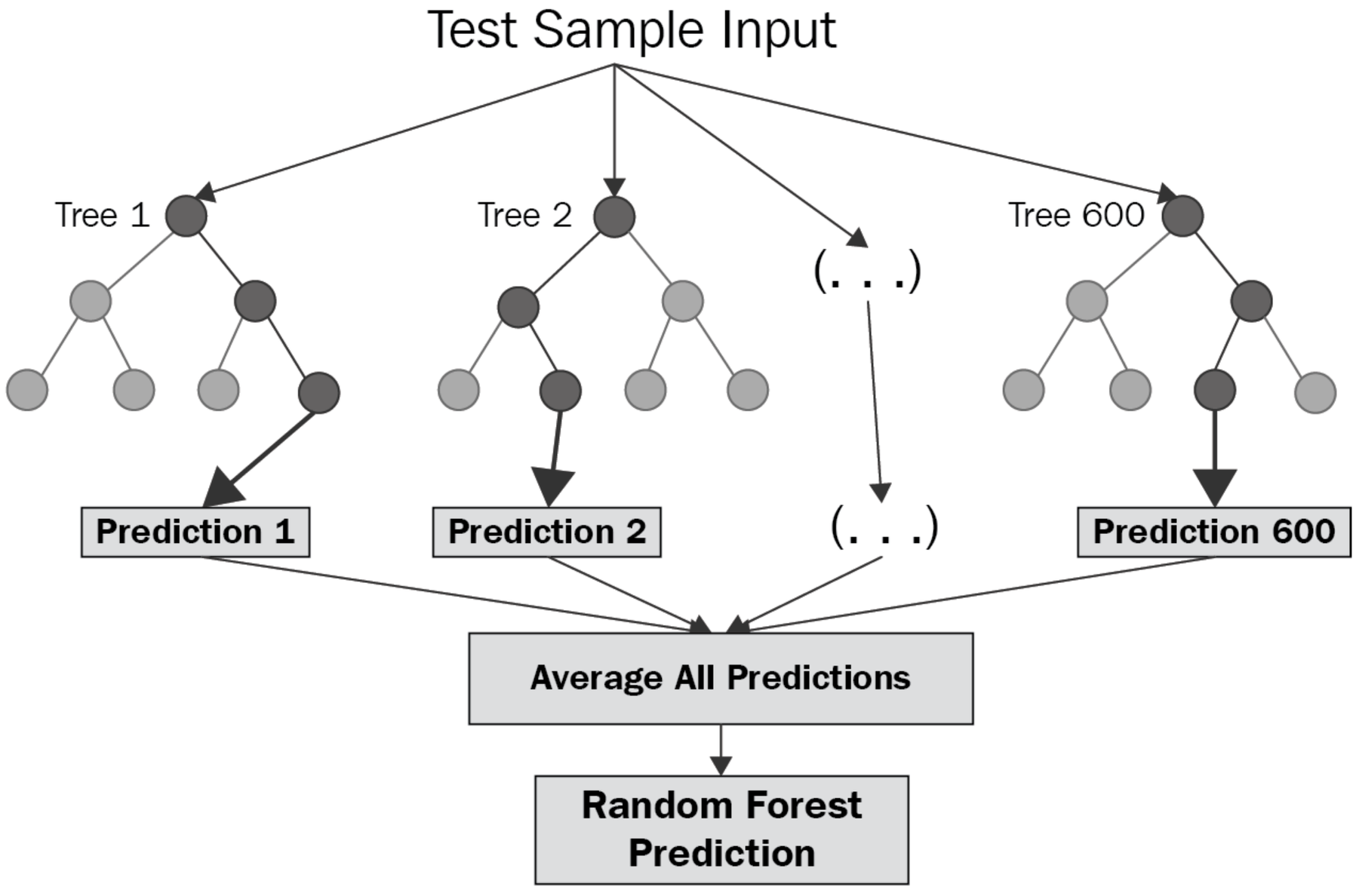
Overfitting Problem



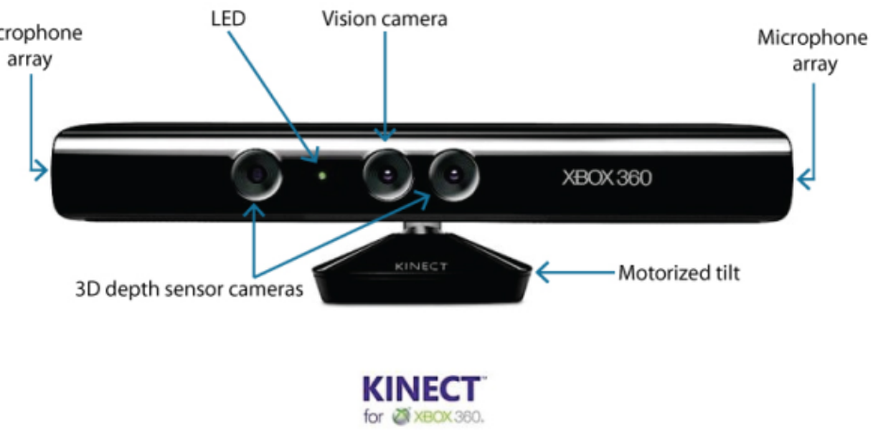
CROSS VALIDATING

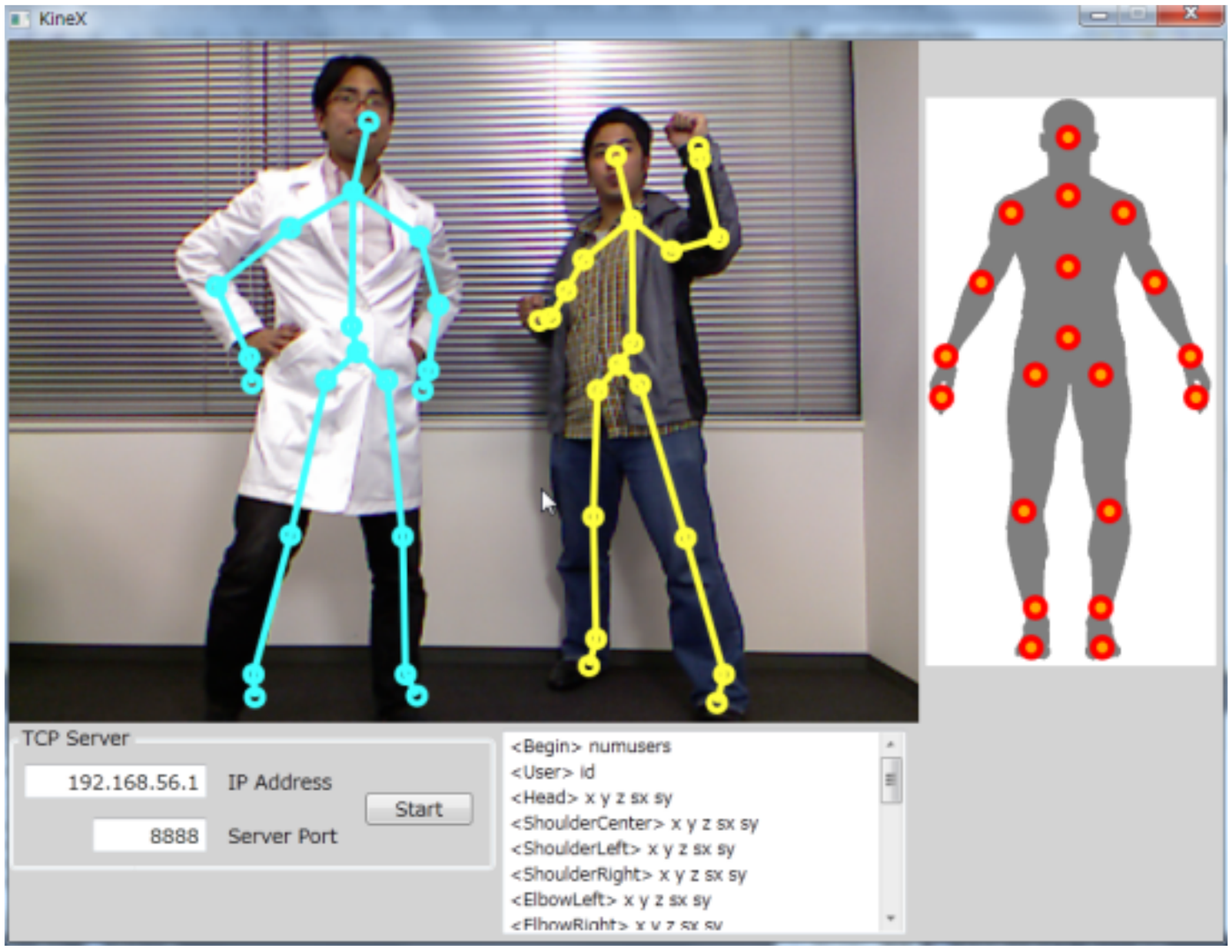


RANDOM FOREST

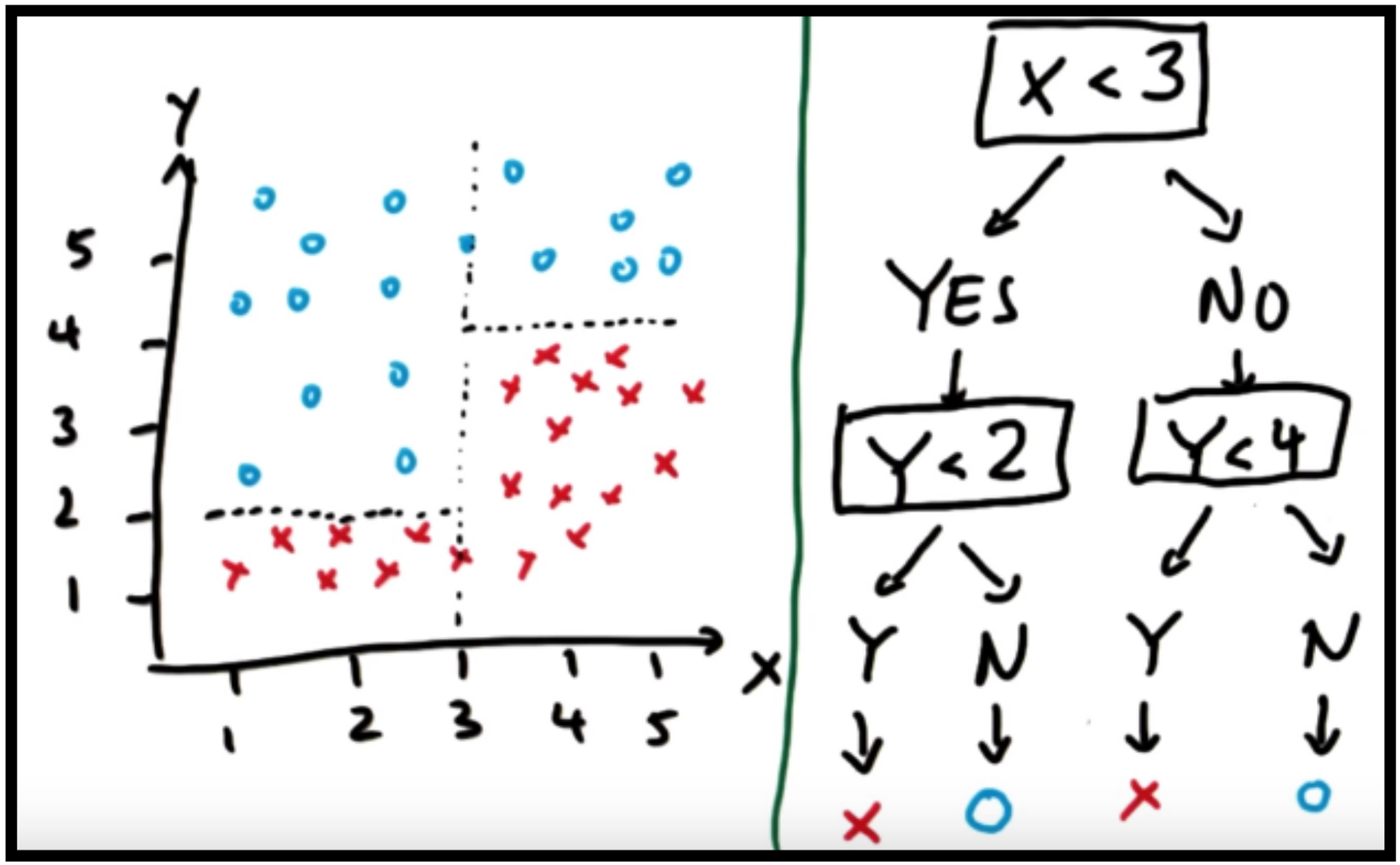


RANDOM FOREST – XBOX360 KINECT



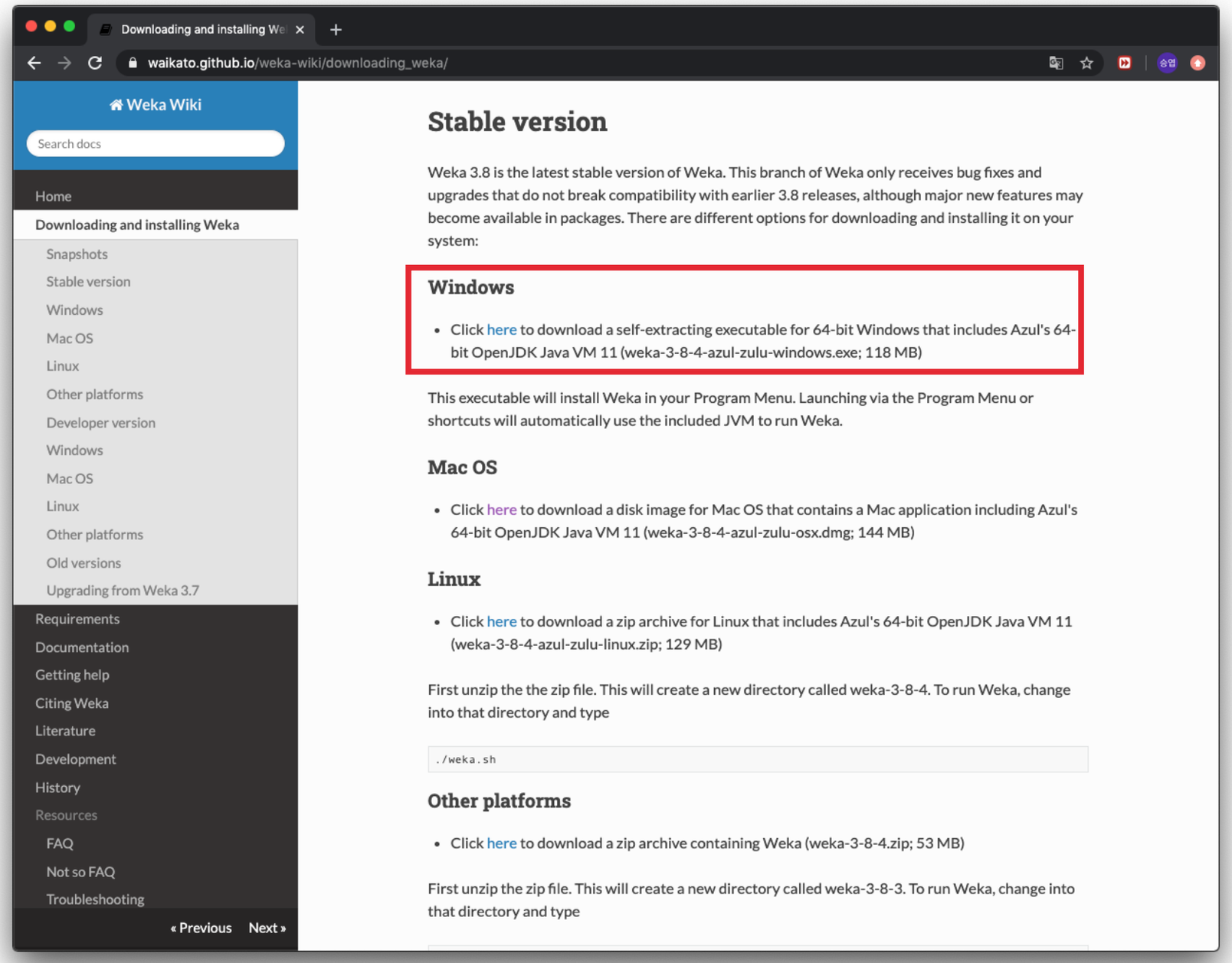


ONLY DISCRETE DATA ARE USED?



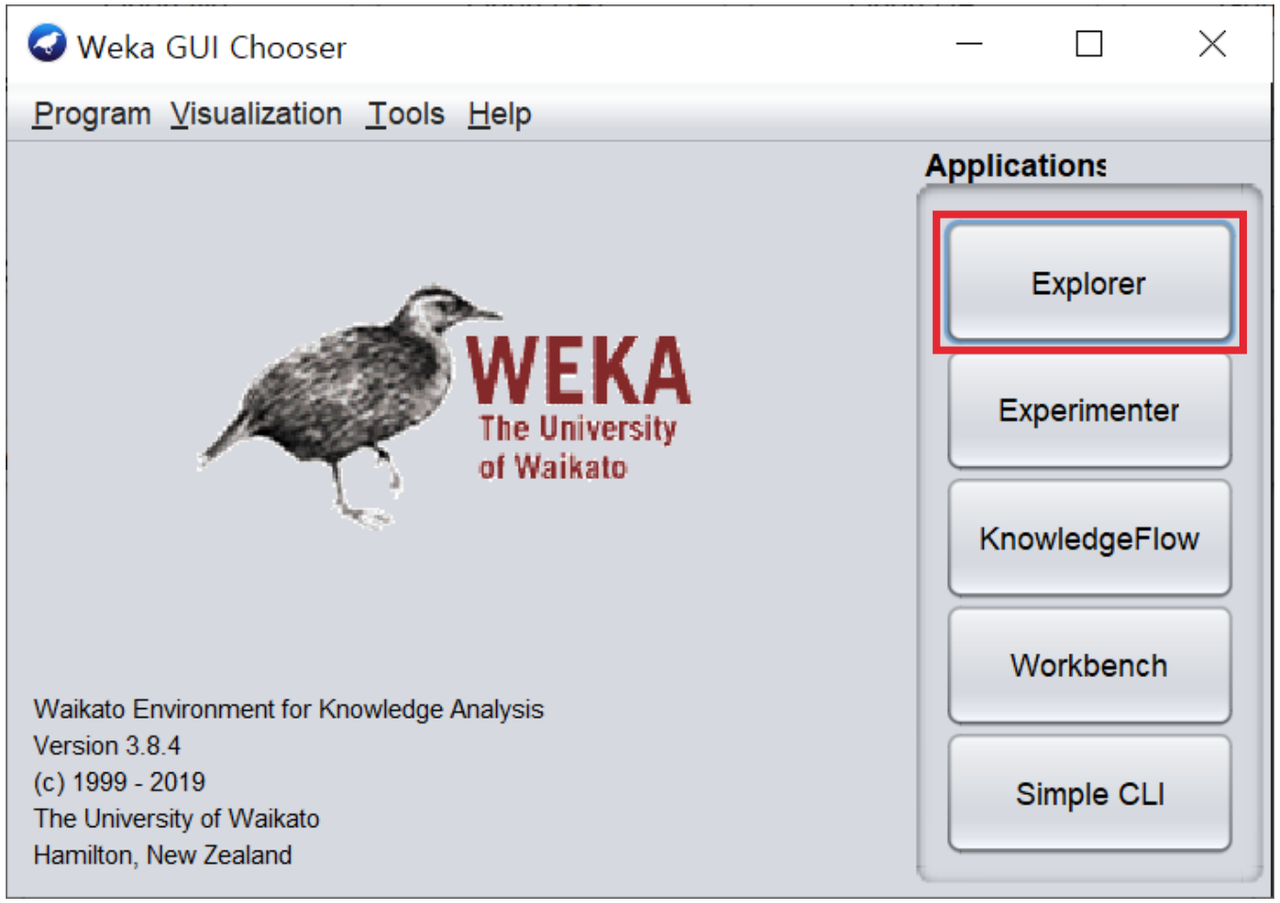
4. DECISION TREE

WEKA

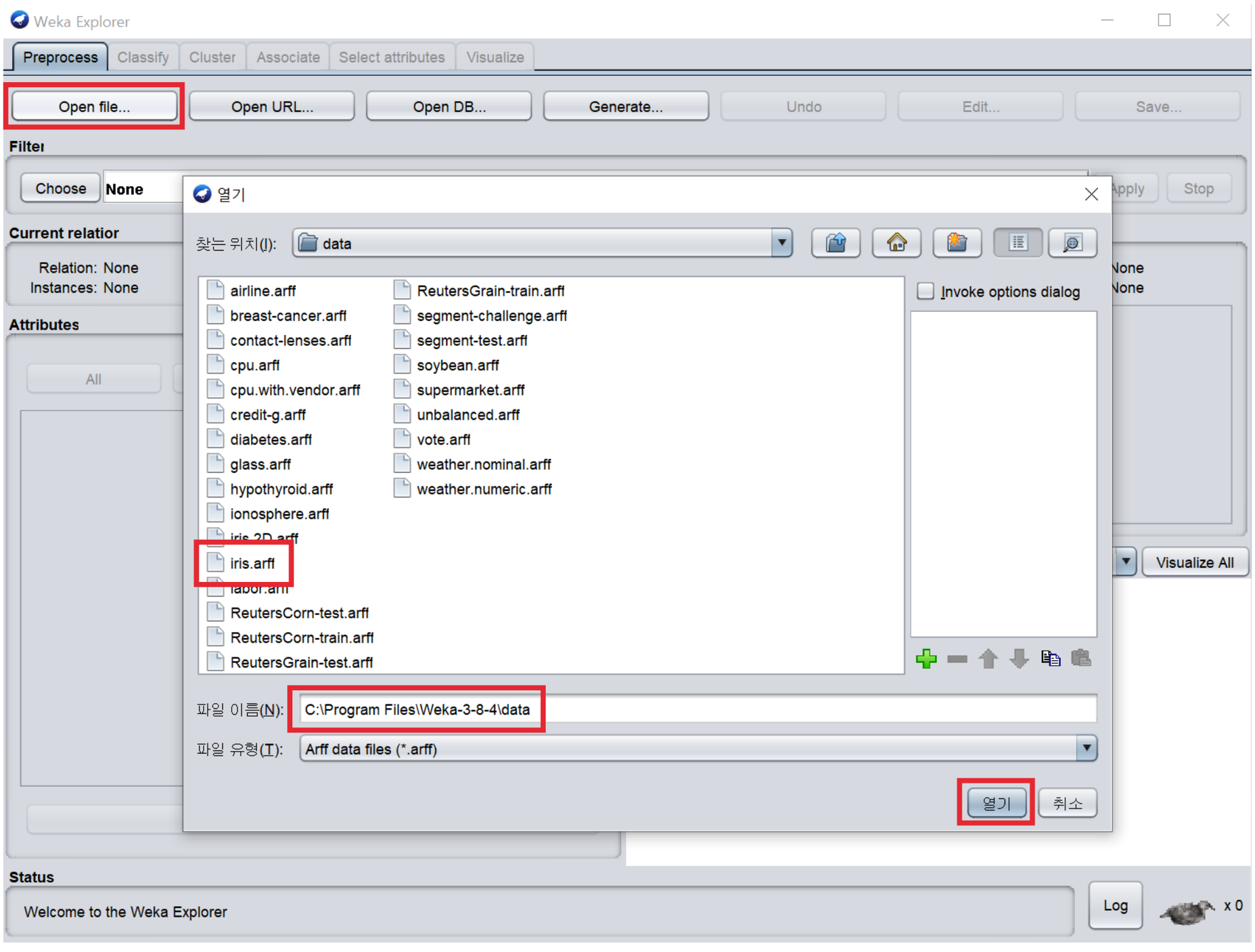


https://waikato.github.io/weka-wiki/downloading_weka/

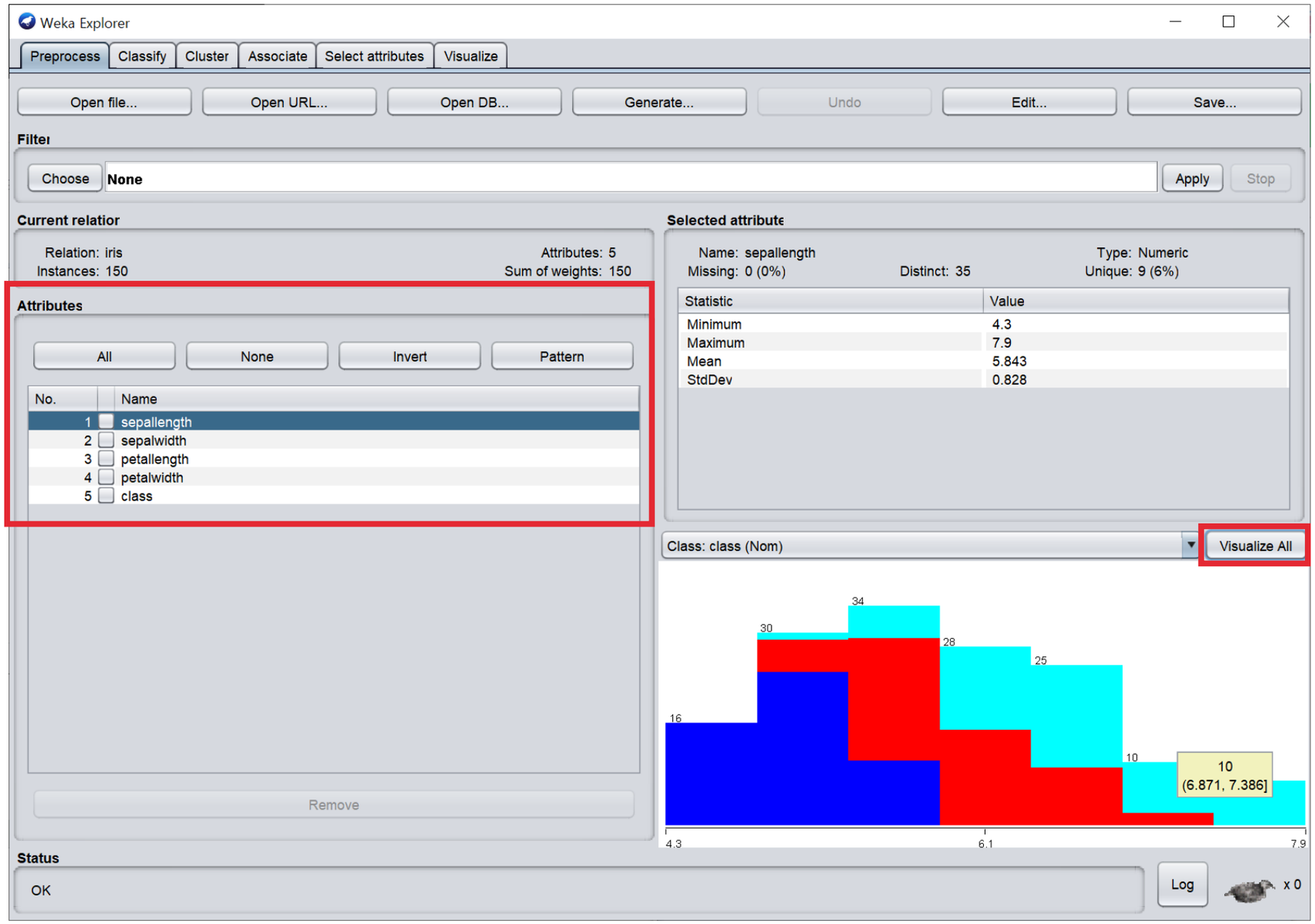
WEKA


Weka GUI Chooser
Program Visualization Tools Help
WEKA
The University of Waikato
Waikato Environment for Knowledge Analysis
Version 3.8.4
(c) 1999 - 2019
The University of Waikato
Hamilton, New Zealand
Applications
Explorer
Experimenter
KnowledgeFlow
Workbench
Simple CLI

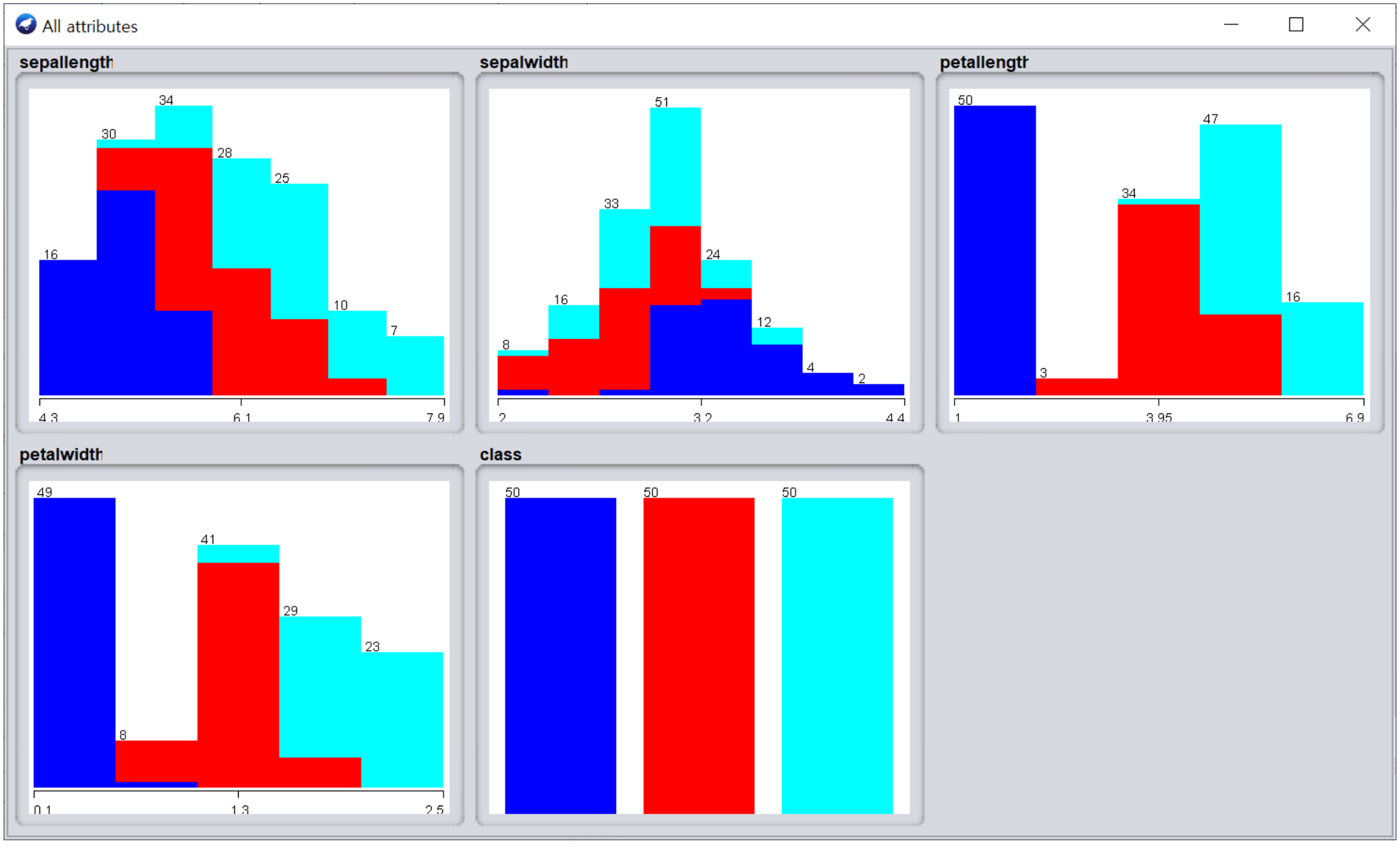
WEKA


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Open file...
Open URL...
Open DB...
Generate...
Undo
Edit...
Save...
Filter
Choose
None
Apply
Stop
Current relation
Relation: None
Instances: None
Attributes
All
열기
찾는 위치(I):
data
airline.arff
breast-cancer.arff
contact-lenses.arff
cpu.arff
cpu.with.vendor.arff
credit-g.arff
diabetes.arff
glass.arff
hypothyroid.arff
ionosphere.arff
iris.arff
ReutersCorn-test.arff
ReutersCorn-train.arff
ReutersGrain-test.arff
ReutersGrain-train.arff
segment-challenge.arff
segment-test.arff
soybean.arff
supermarket.arff
unbalanced.arff
vote.arff
weather.nominal.arff
weather.numeric.arff
Invoke options dialog
파일 이름(N):
C:\Program Files\Weka-3-8-4\data
파일 유형(T):
Arff data files (*.arff)
열기
취소
None
None
Visualize All
Status
Welcome to the Weka Explorer
Log
x 0

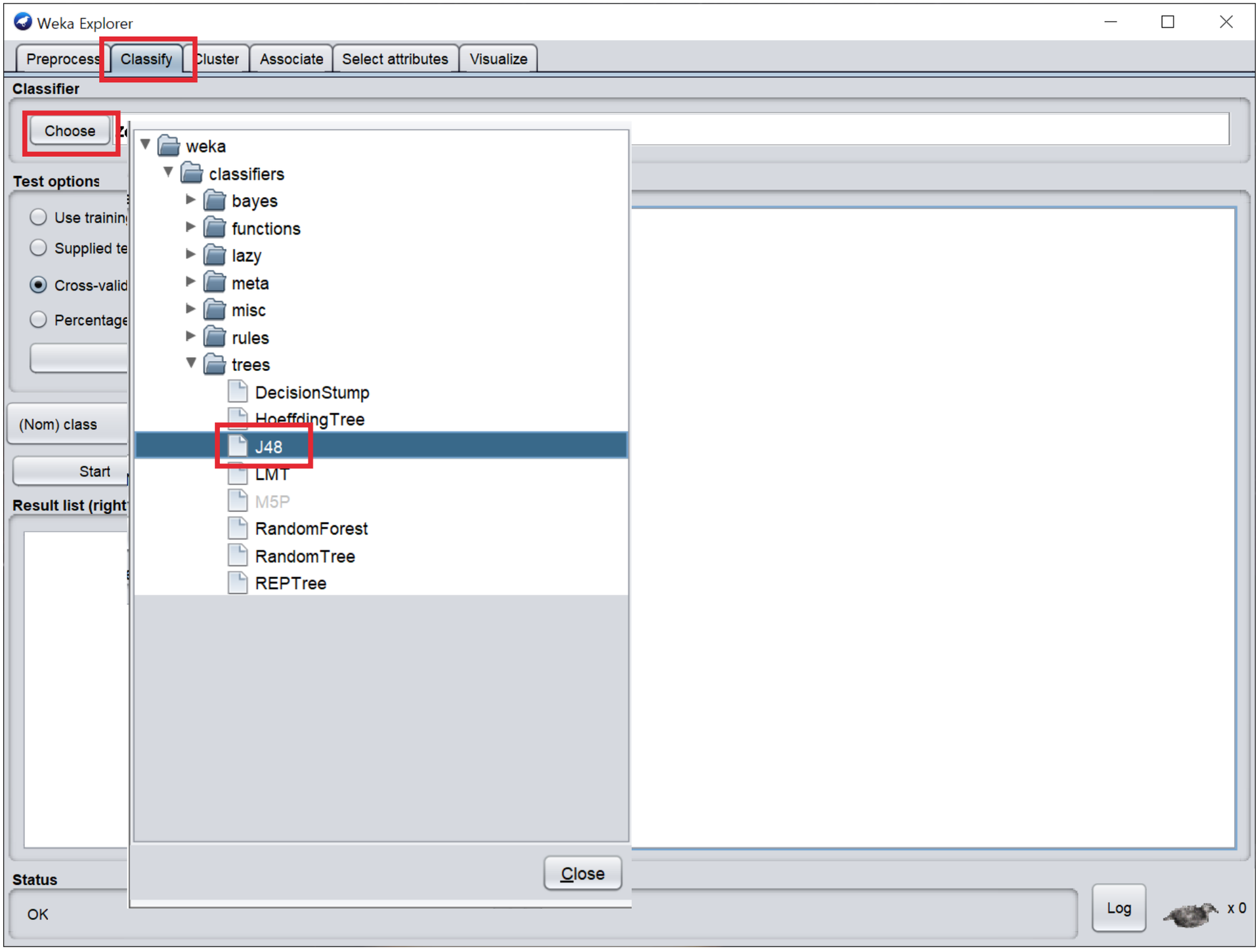
WEKA


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Open file...
Open URL...
Open DB...
Generate...
Undo
Edit...
Save...
Filter
Choose
None
Apply
Stop
Current relation
Relation: iris
Instances: 150
Attributes: 5
Sum of weights: 150
Attributes
All
None
Invert
Pattern
No.
Name
1 sepallength
2 sepalwidth
3 petallength
4 petalwidth
5 class
Remove
Selected attribute
Name: sepallength
Missing: 0 (0%)
Distinct: 35
Type: Numeric
Unique: 9 (6%)
Statistic
Value
Minimum 4.3
Maximum 7.9
Mean 5.843
StdDev 0.828
Class: class (Nom)
Visualize All
34
30
28
25
16
10
10
(6.871, 7.386]
4.3
6.1
7.9
Status
OK
Log
x 0

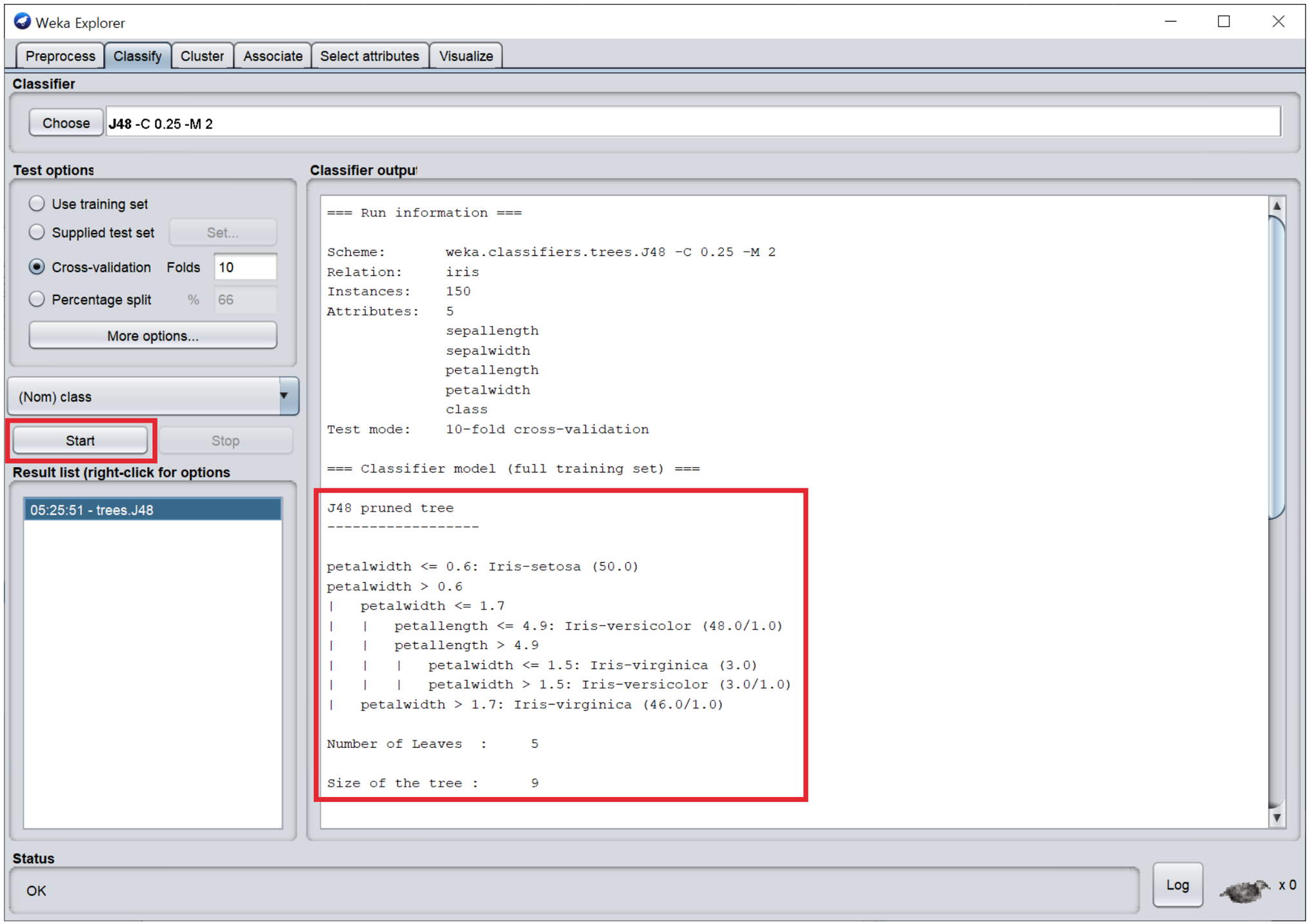
WEKA


All attributes
sepallength
34
30
28
25
16
10
7
4.3
6.1
7.9
sepalwidth
51
33
24
16
12
8
4
2
2
3.2
4.4
petallengtl
50
47
34
16
3
1
3.95
6.9
petalwidth
49
41
29
23
8
0.1
1.3
2.5
class
50
50
50

WEKA

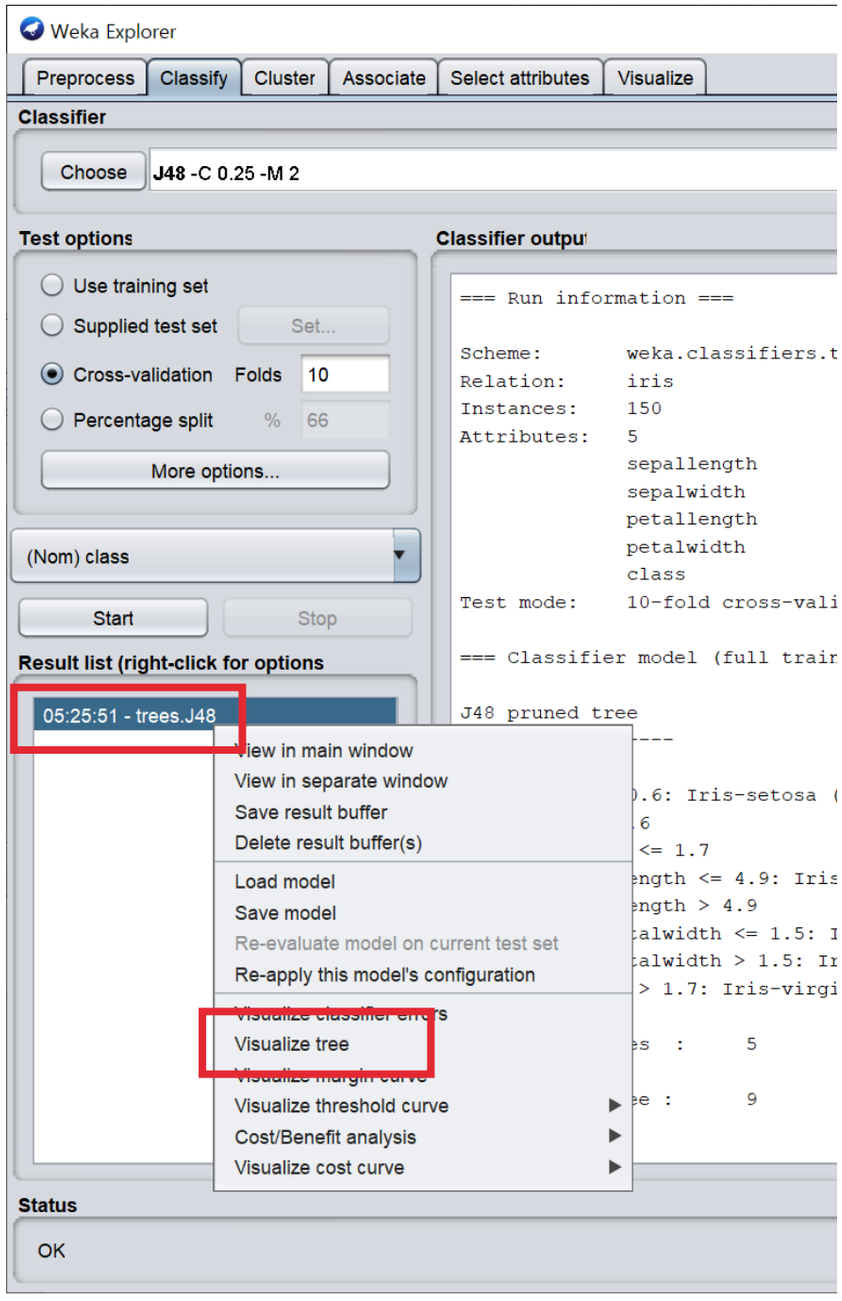

Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
Test options
Use trainin
Supplied te
Cross-valid
Percentage
(Nom) class
Start
Result list (right
Status
OK
Log
x 0
weka
classifiers
bayes
functions
lazy
meta
misc
rules
trees
DecisionStump
HoeffdingTree
J48
LMT
M5P
RandomForest
RandomTree
REPTree
Close

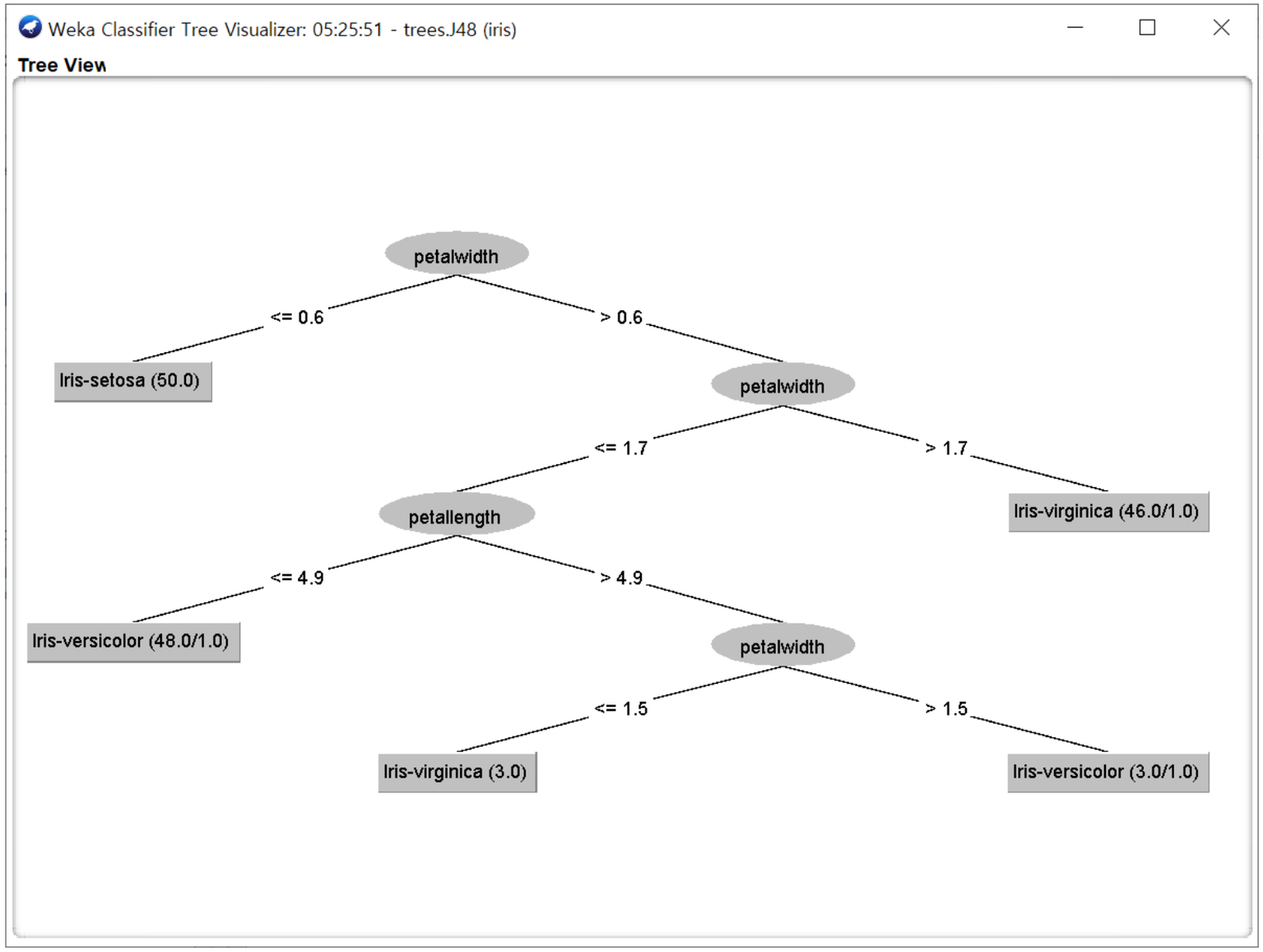
WEKA


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
J48 -C 0.25 -M 2
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
10
Percentage split
%
66
More options...
(Nom) class
Start
Stop
Result list (right-click for options
05:25:51 - trees.J48
Classifier output
=== Run information ===
Scheme: weka.classifiers.trees.J48 -C 0.25 -M 2
Relation: iris
Instances: 150
Attributes: 5
sepallength
sepalwidth
petallength
petalwidth
class
Test mode: 10-fold cross-validation
=== Classifier model (full training set) ===
J48 pruned tree
petalwidth <= 0.6: Iris-setosa (50.0)
petalwidth > 0.6
| petalwidth <= 1.7
| | petallength <= 4.9: Iris-versicolor (48.0/1.0)
| | petallength > 4.9
| | | petalwidth <= 1.5: Iris-virginica (3.0)
| | | petalwidth > 1.5: Iris-versicolor (3.0/1.0)
| petalwidth > 1.7: Iris-virginica (46.0/1.0)
Number of Leaves : 5
Size of the tree : 9
Status
OK
Log
x 0

4. DECISION TREE

WEKA





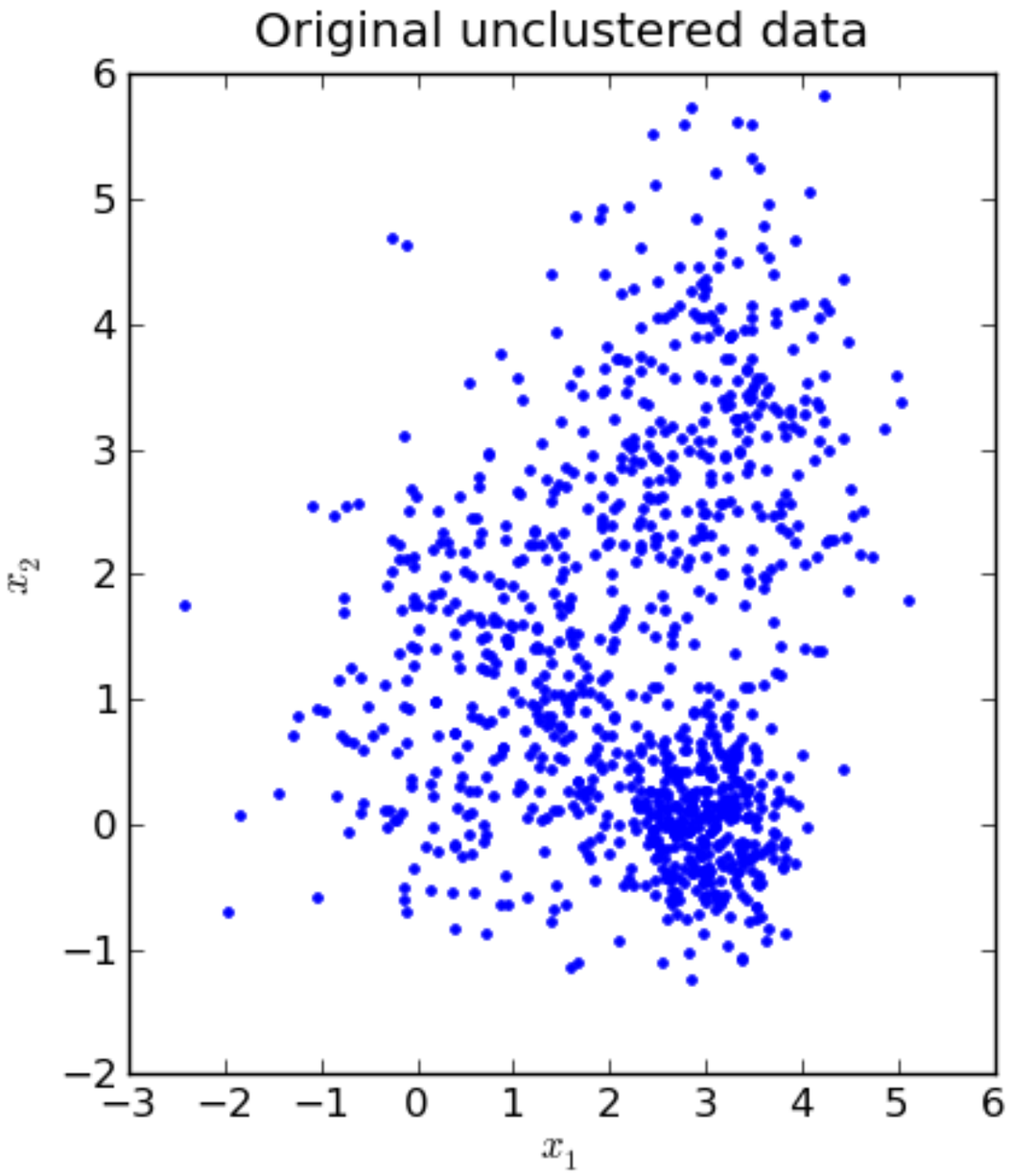
DEFECT OF SUPERVISED LEARNING

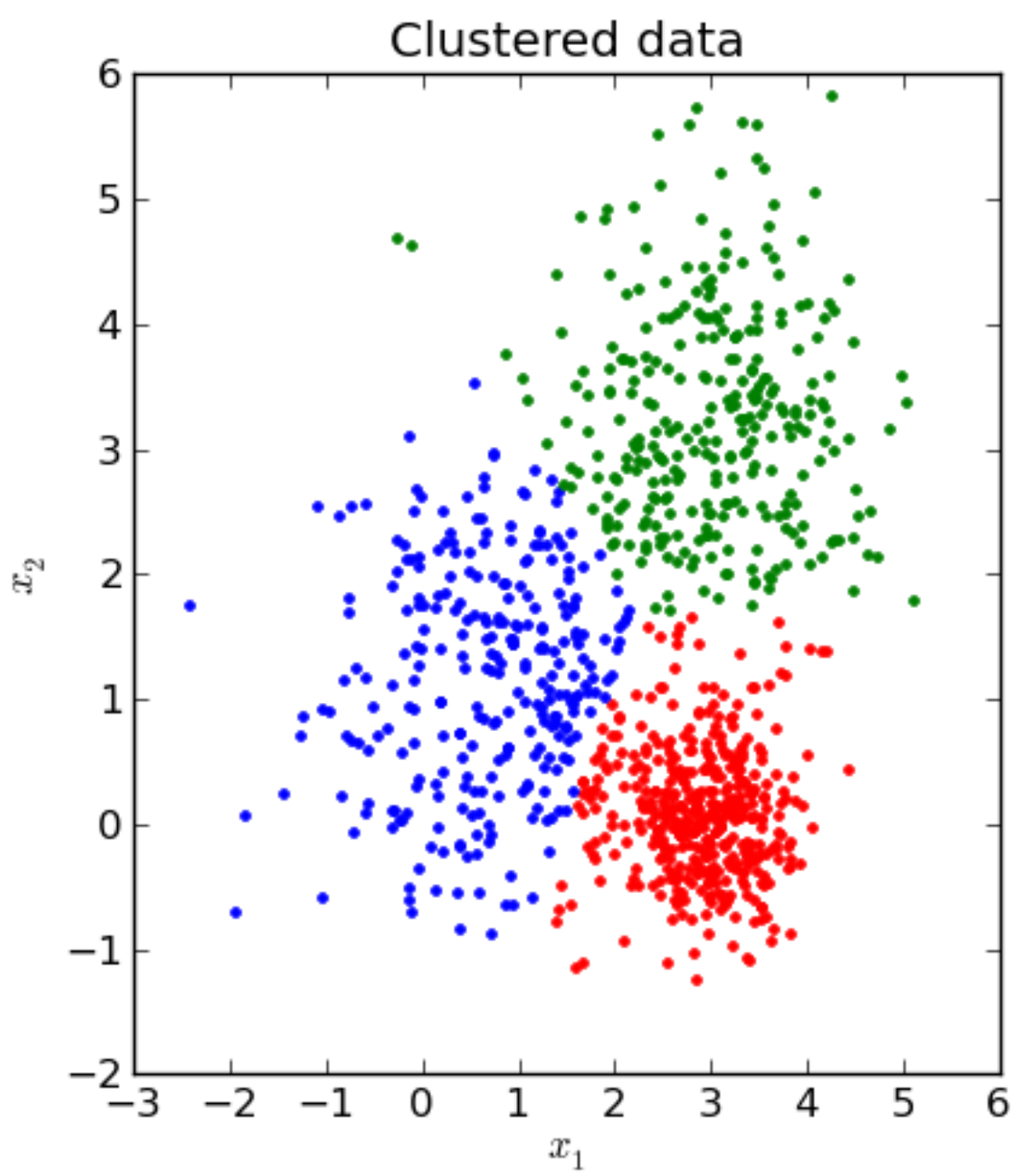
DEFECT OF SUPERVISED LEARNING

AI의 머신러닝, 알고보니 中·인도의 값싼 노동력 덕분

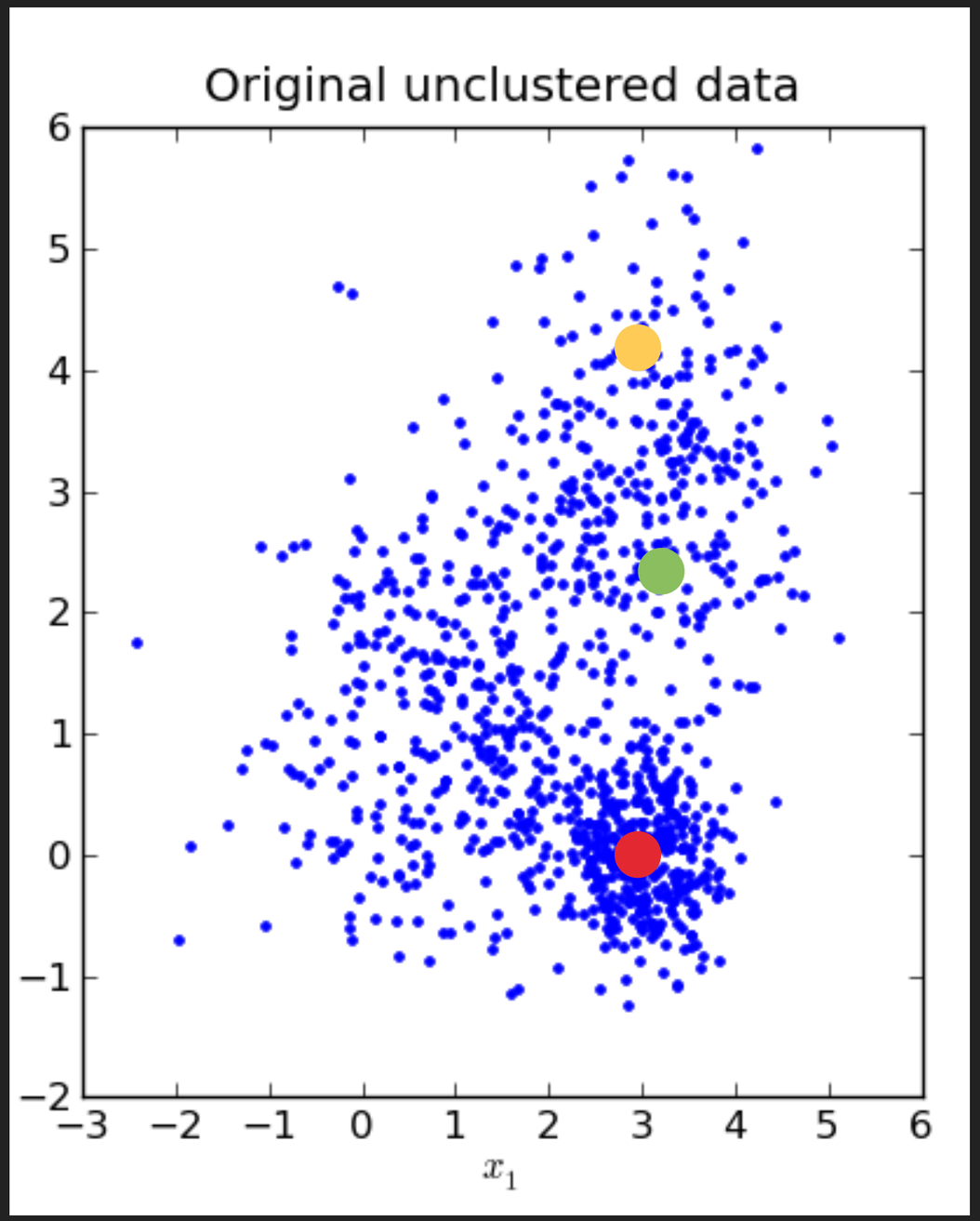
https://biz.chosun.com/site/data/html_dir/2019/12/19/2019121900144.html

WHAT ABOUT K-MEAN CLUSTERING?



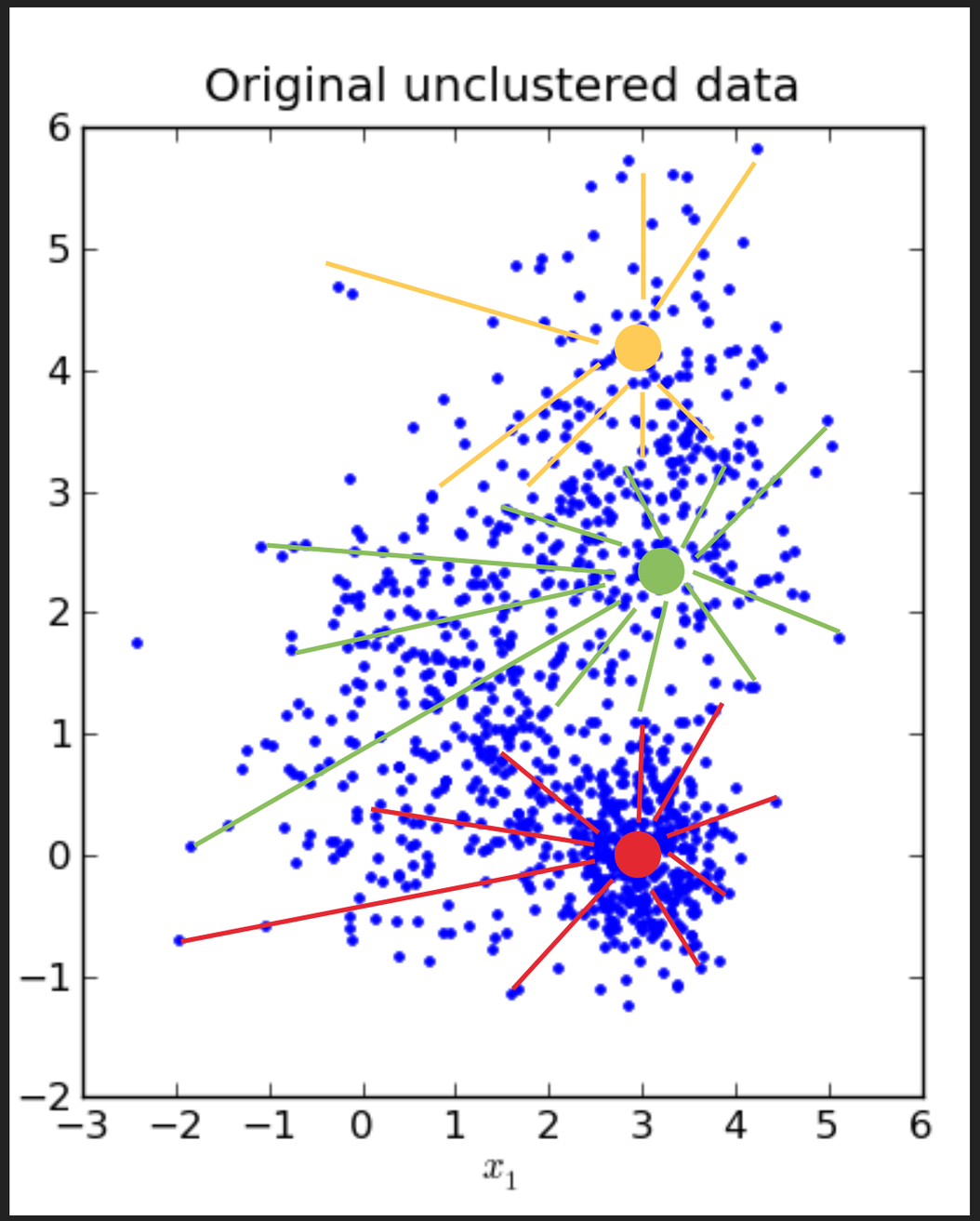


K-MEAN CLUSTERING?



무작위 K개의 점을 지정

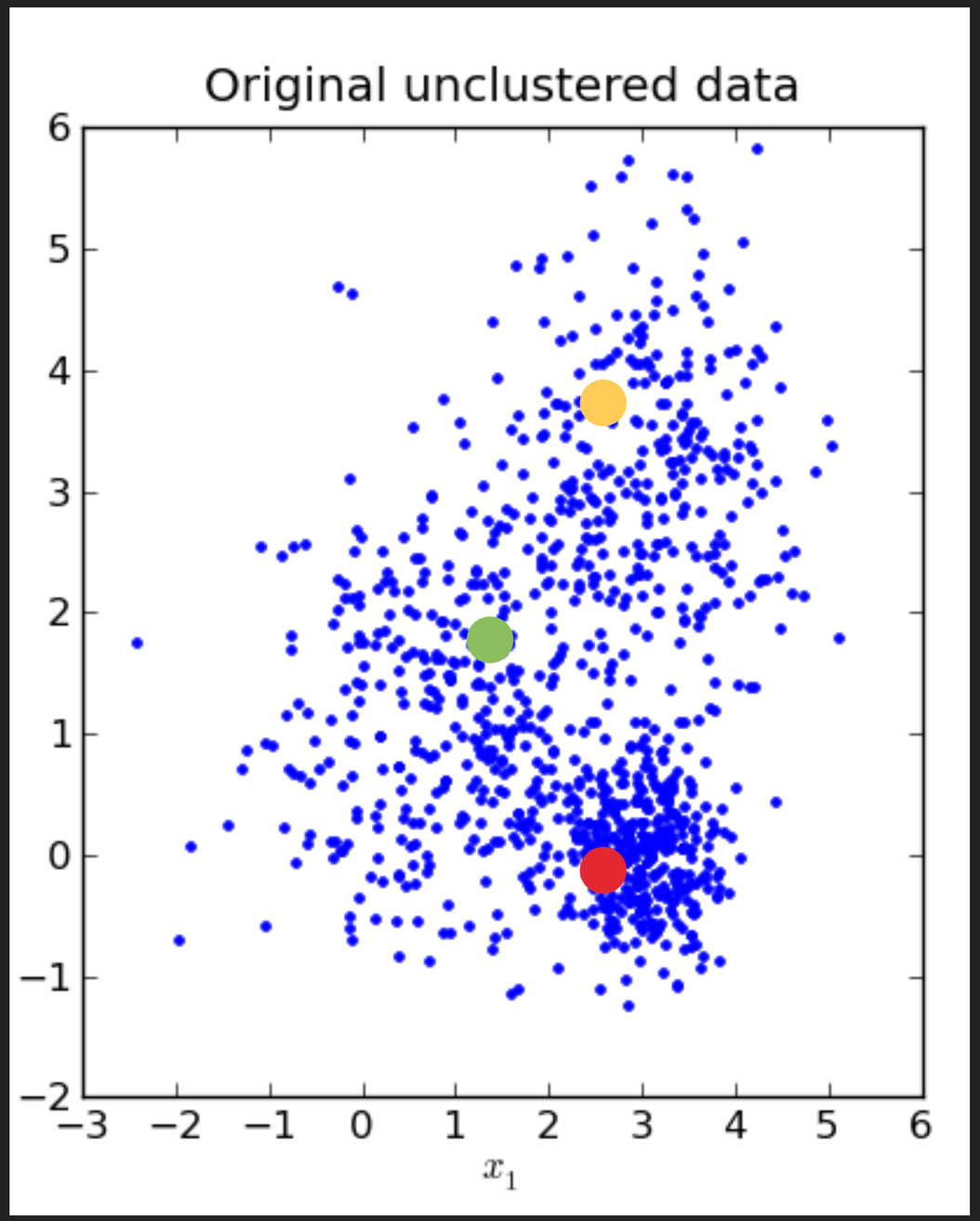
K-MEAN CLUSTERING?



모든 데이터를 K개의 점 중

제일 가까운 점과 연결

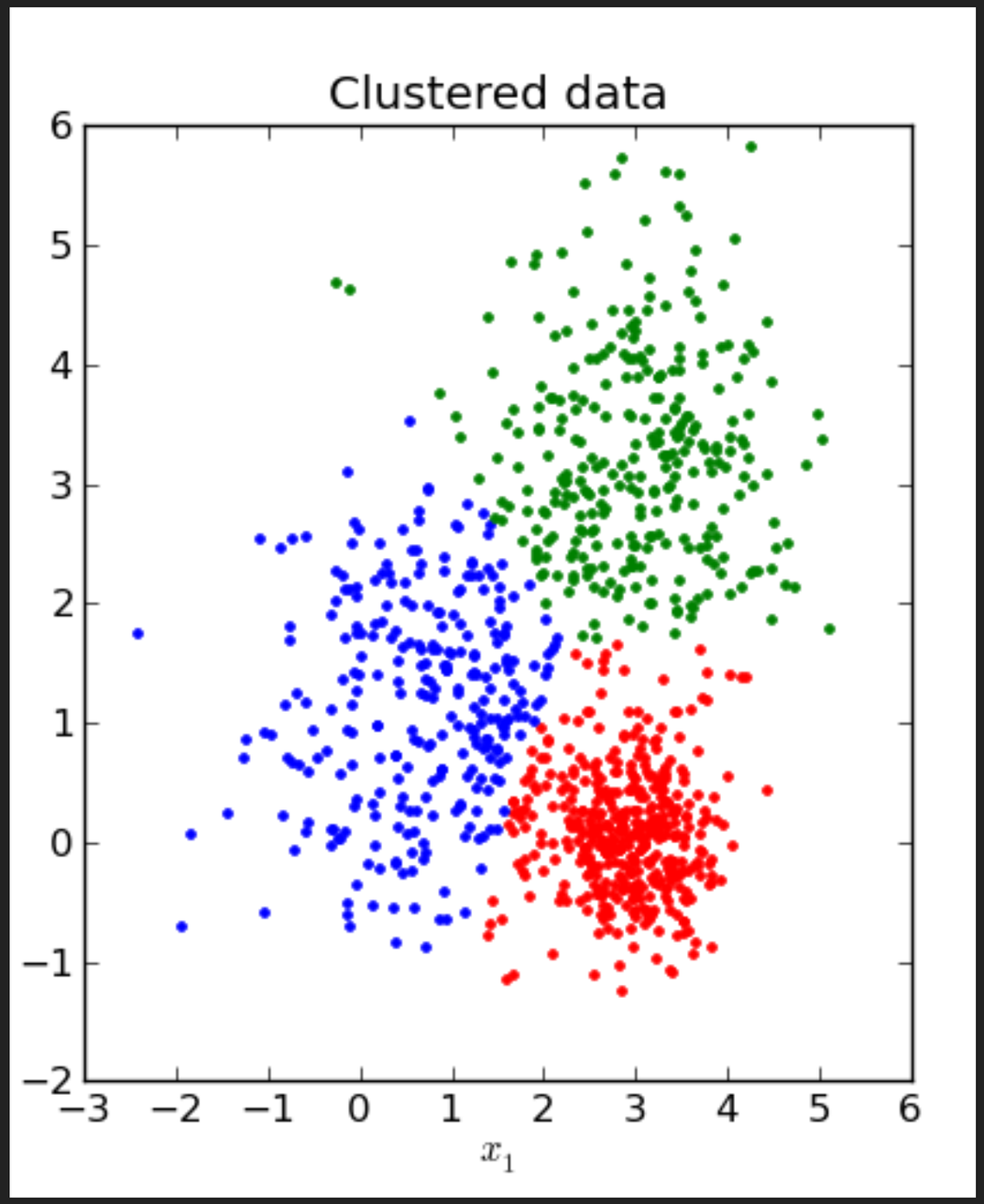
K-MEAN CLUSTERING?



연결된 점들의 평균 좌표를 구해서

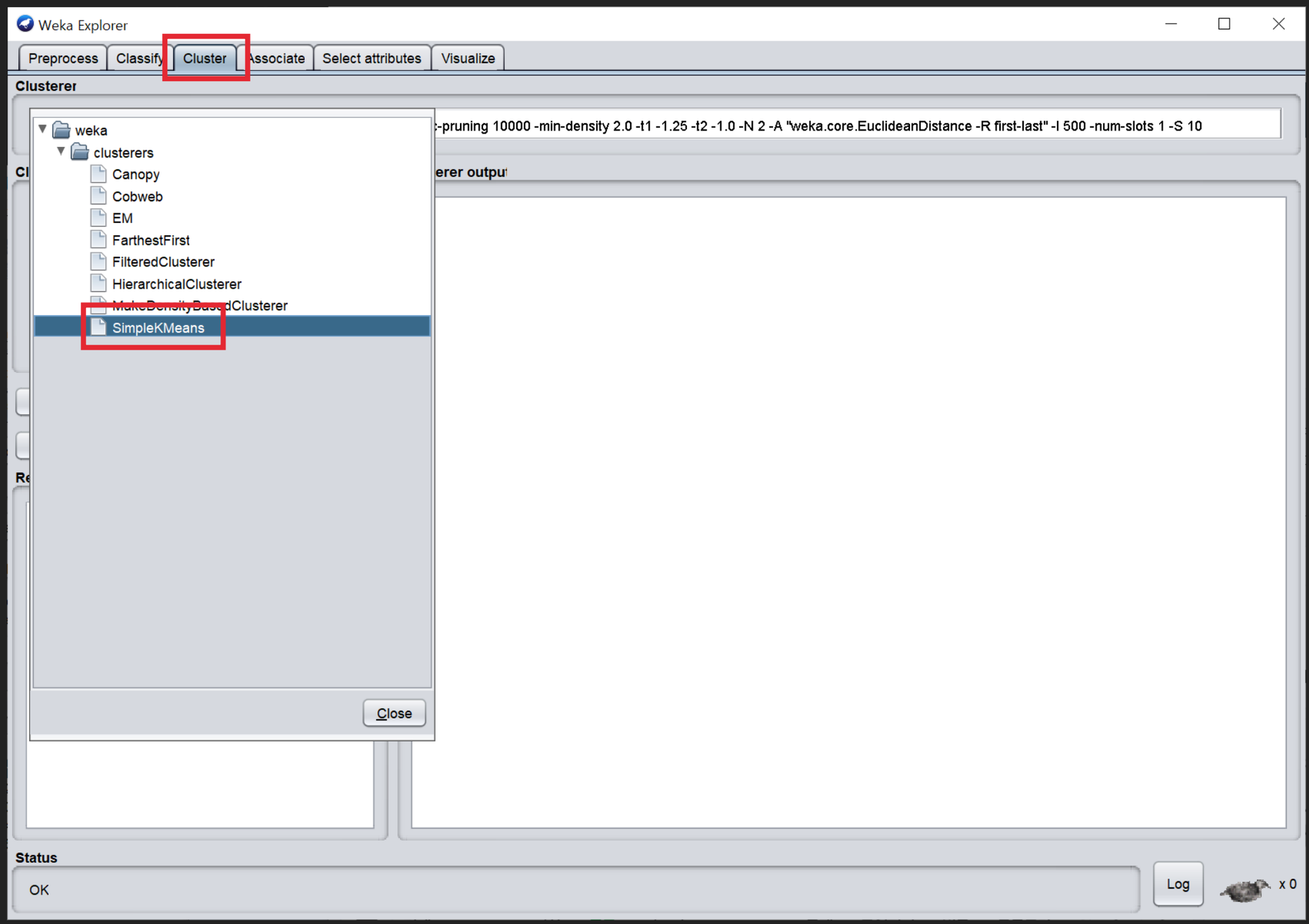
그 좌표로 이동

K-MEAN CLUSTERING?



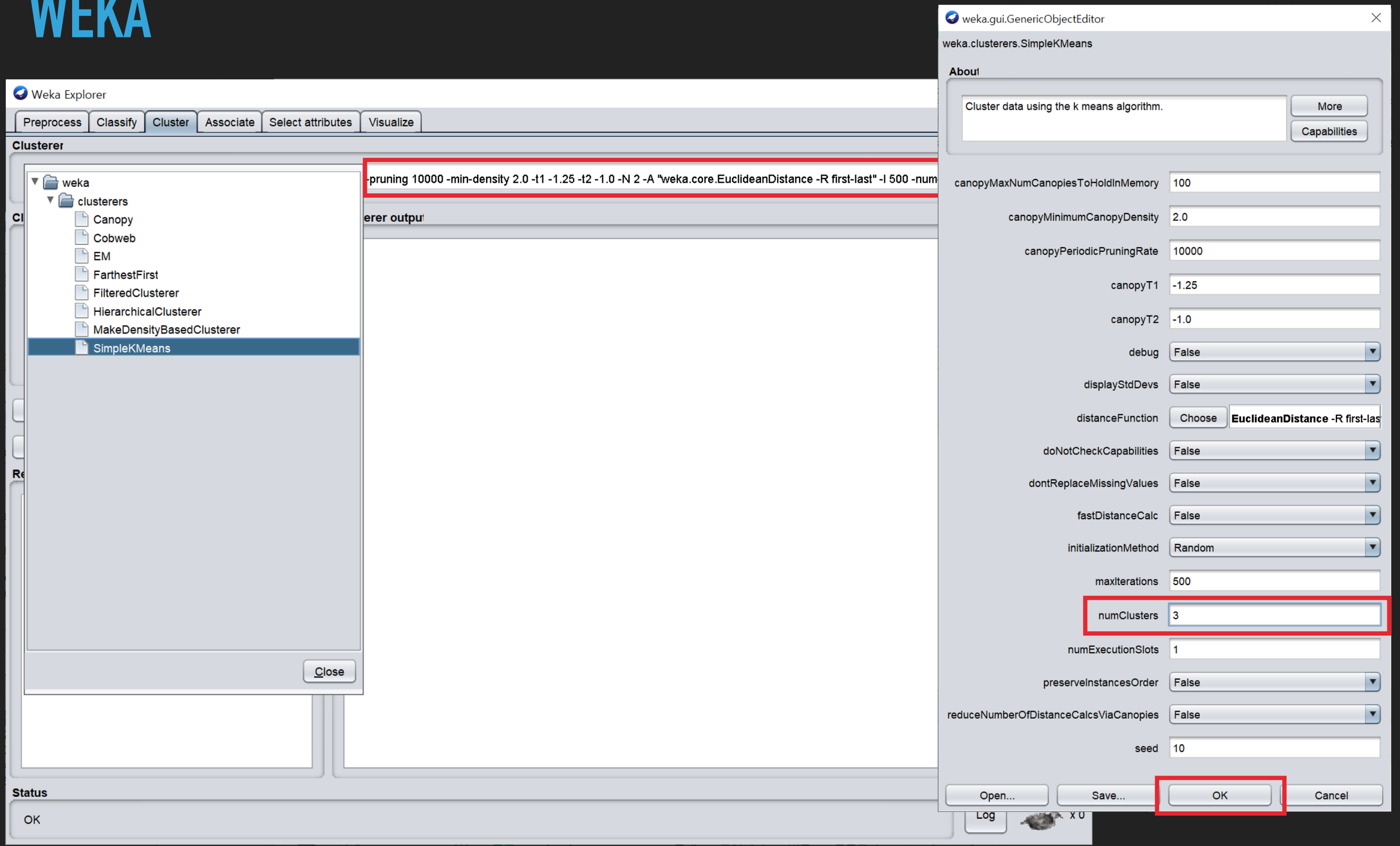
움직임이 없어질 때까지 반복

WEKA

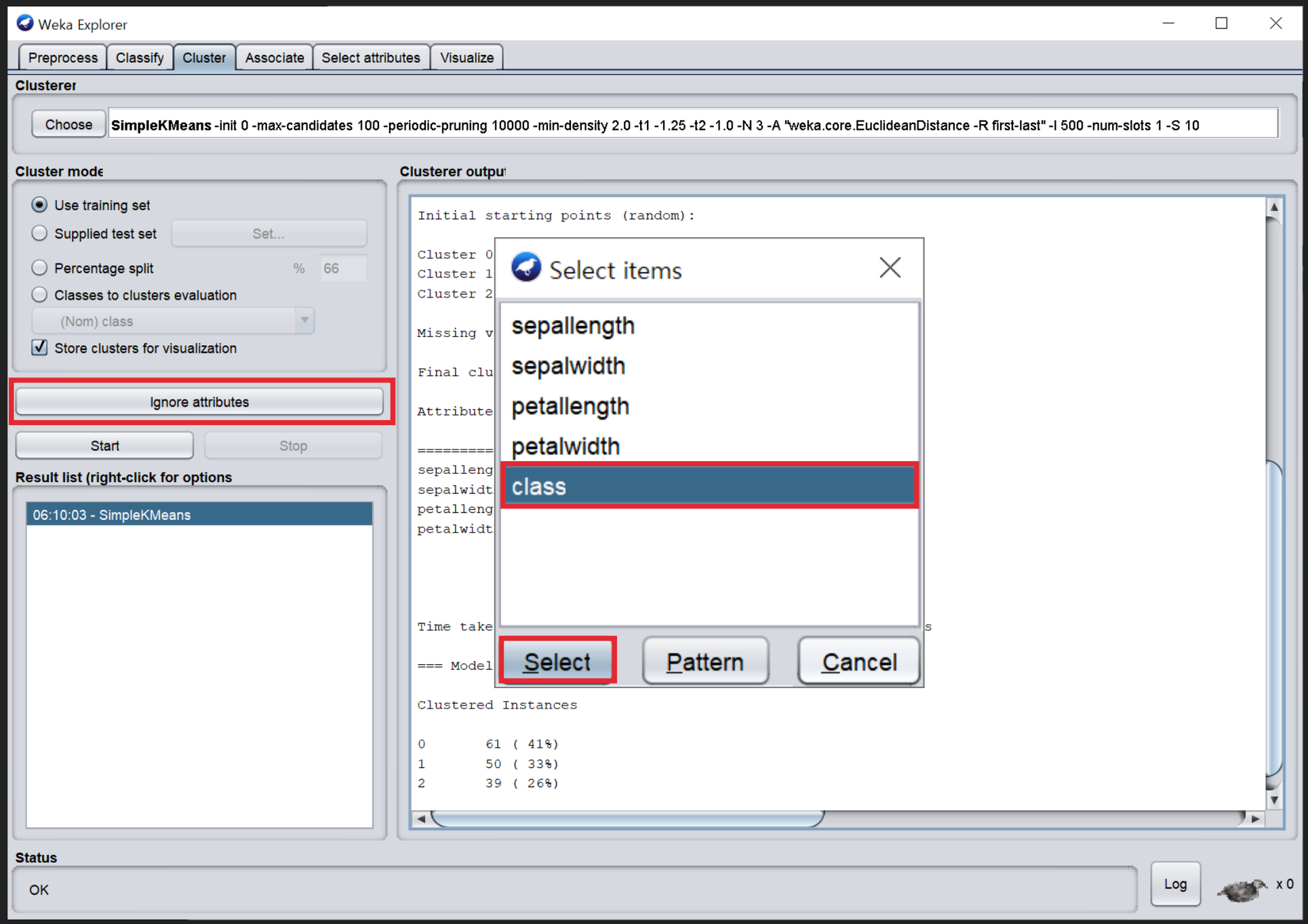

Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Clusterer
-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 2 -A "weka.core.EuclideanDistance -R first-last" -I 500 -num-slots 1 -S 10
weka
clusterers
Canopy
Cobweb
EM
FarthestFirst
FilteredClusterer
HierarchicalClusterer
SimpleKMeans
Close
Status
OK
Log
x 0

5. K-MEAN CLUSTERING

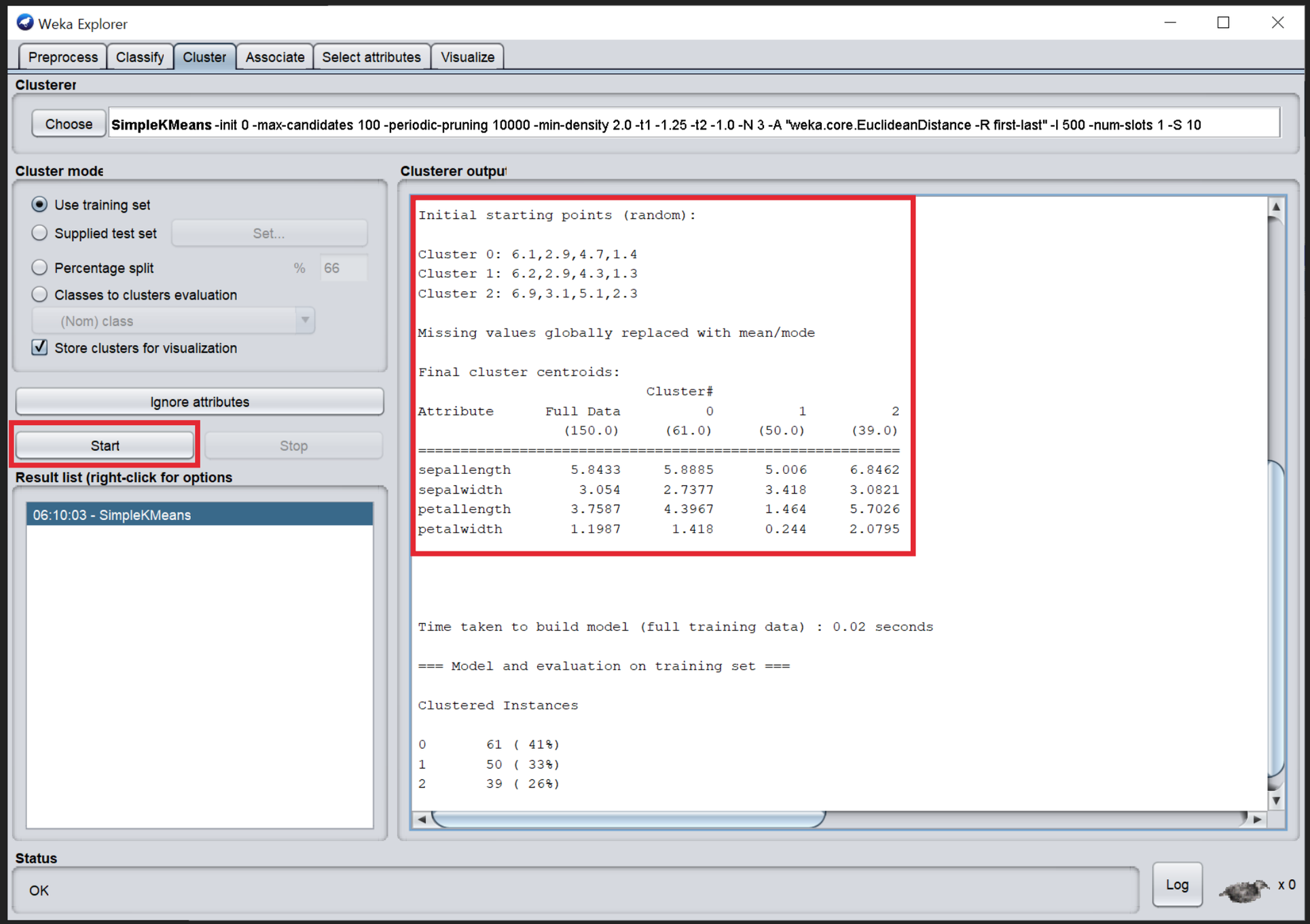
WEKA



WEKA

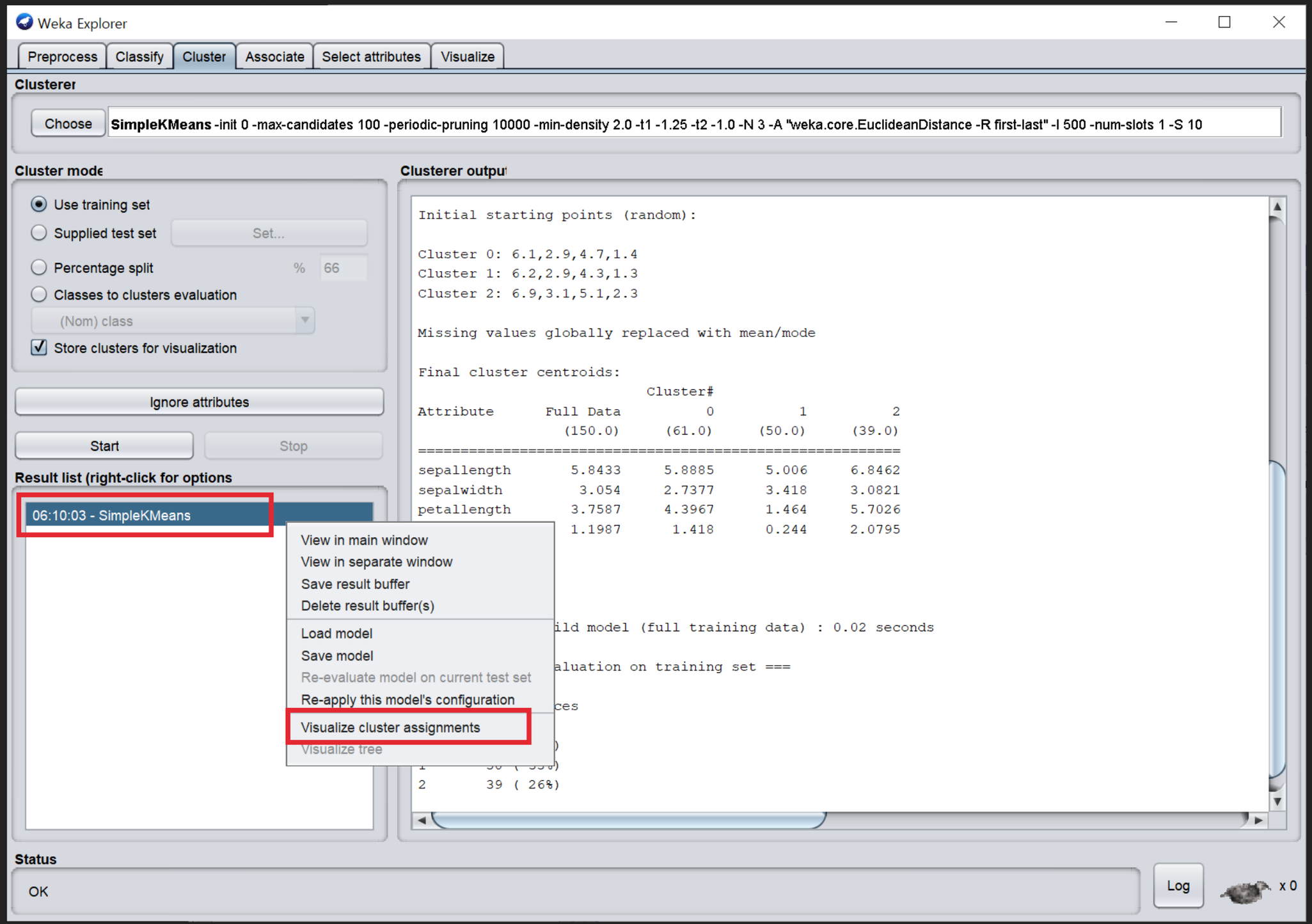

Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Clusterer
Choose
SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 3 -A "weka.core.EuclideanDistance -R first-last" -I 500 -num-slots 1 -S 10
Cluster mode
Use training set
Supplied test set
Set...
Percentage split
%
66
Classes to clusters evaluation
(Nom) class
Store clusters for visualization
Ignore attributes
Start
Stop
Result list (right-click for options
06:10:03 - SimpleKMeans
Clusterer output
Initial starting points (random):
Cluster 0
Cluster 1
Cluster 2
Missing v
Final clu
Attribute
=========
sepalleng
sepalwidt
petalleng
petalwidt
Time take
s
=== Model
Clustered Instances
0 61 (41%)
1 50 (33%)
2 39 (26%)
Select items
sepallength
sepalwidth
petallength
petalwidth
class
Select
Pattern
Cancel
Status
OK
Log
x 0

WEKA


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Clusterer
Choose
SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 3 -A "weka.core.EuclideanDistance -R first-last" -I 500 -num-slots 1 -S 10
Cluster mode
Use training set
Supplied test set
Set...
Percentage split
%
66
Classes to clusters evaluation
(Nom) class
Store clusters for visualization
Ignore attributes
Start
Stop
Result list (right-click for options
06:10:03 - SimpleKMeans
Clusterer output
Initial starting points (random):
Cluster 0: 6.1,2.9,4.7,1.4
Cluster 1: 6.2,2.9,4.3,1.3
Cluster 2: 6.9,3.1,5.1,2.3
Missing values globally replaced with mean/mode
Final cluster centroids:
Cluster#
Attribute Full Data 0 1 2
(150.0) (61.0) (50.0) (39.0)
sepallength 5.8433 5.8885 5.006 6.8462
sepalwidth 3.054 2.7377 3.418 3.0821
petallength 3.7587 4.3967 1.464 5.7026
petalwidth 1.1987 1.418 0.244 2.0795
Time taken to build model (full training data) : 0.02 seconds
=== Model and evaluation on training set ===
Clustered Instances
0 61 (41%)
1 50 (33%)
2 39 (26%)
Status
OK
Log
x 0

5. K-MEAN CLUSTERING

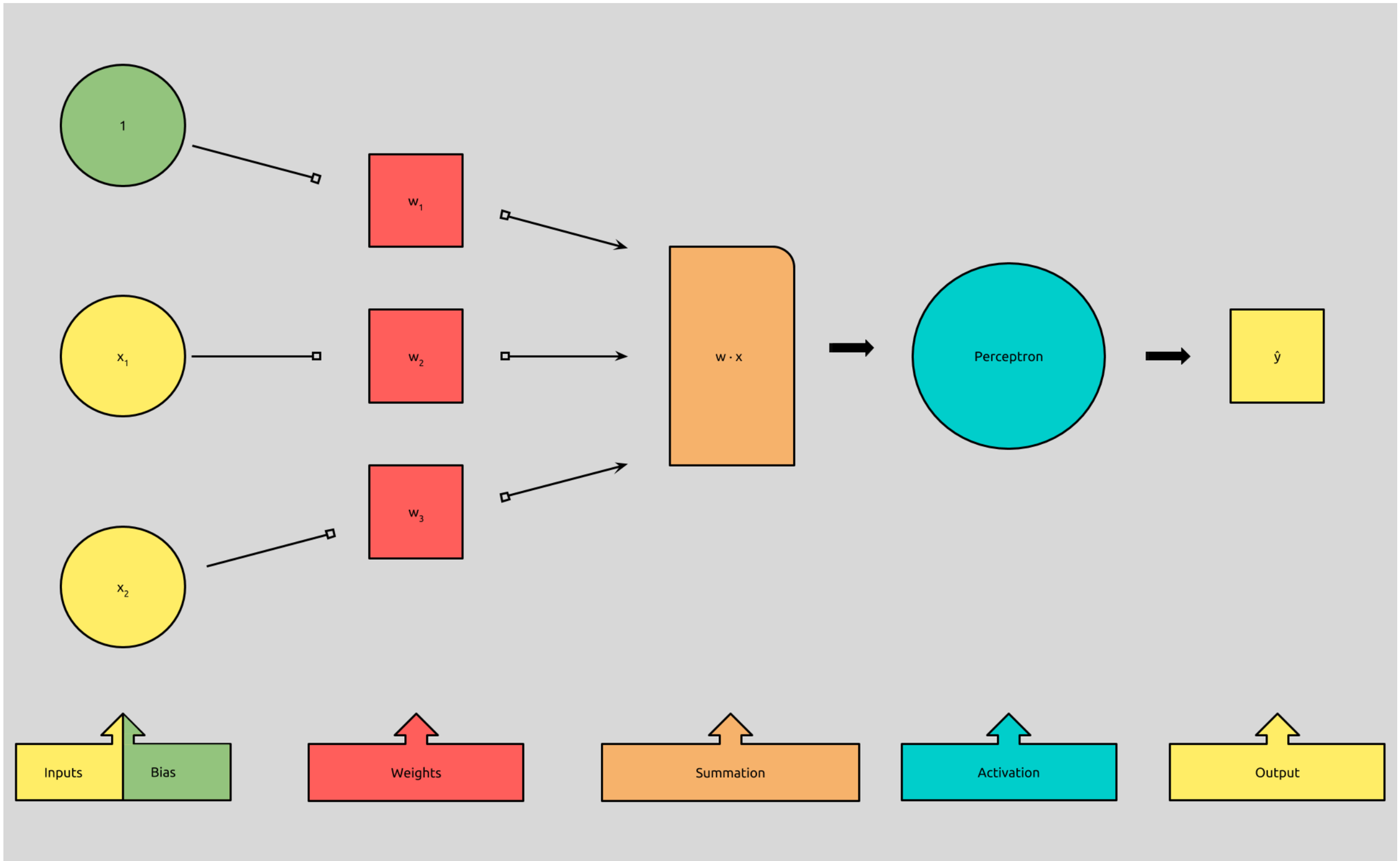
WEKA



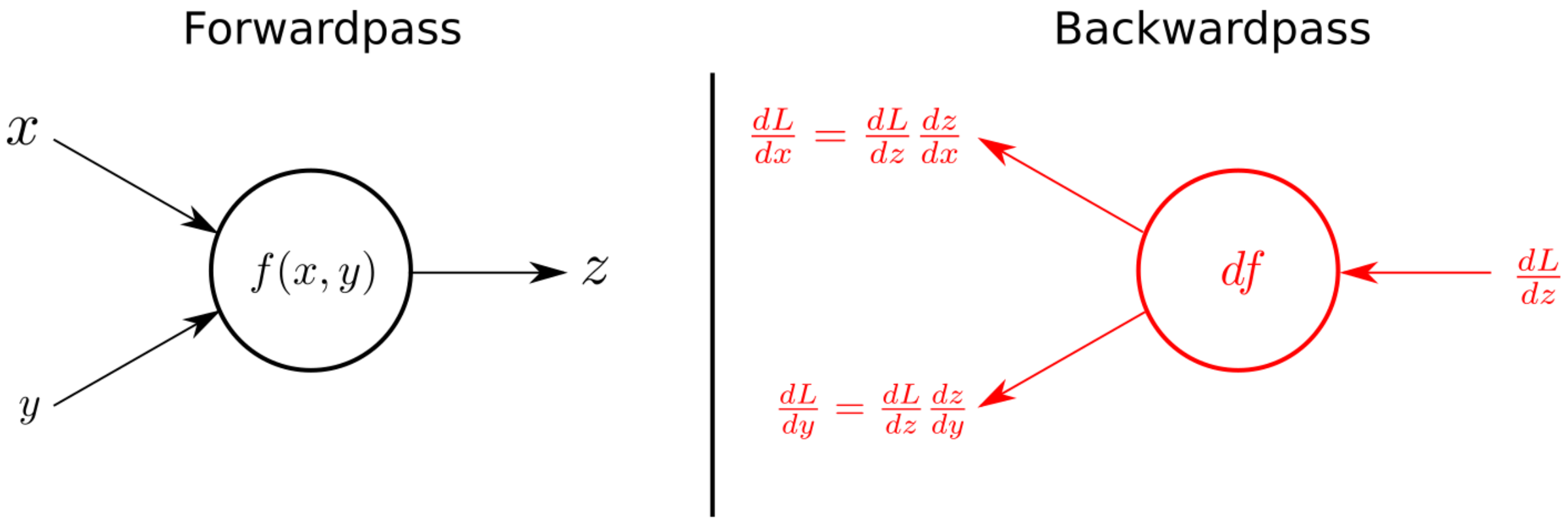
WEKA


Weka Clusterer Visualize: 06:10:03 - SimpleKMeans (iris)
X: sepallength (Num)
Y: petalwidth (Num)
Colour: Cluster (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: iris_clusterec
2.5
1.3
0.1
4.3
6.1
7.9
X
Y
Class colour
cluster0
cluster1
cluster2

PERCEPTRON

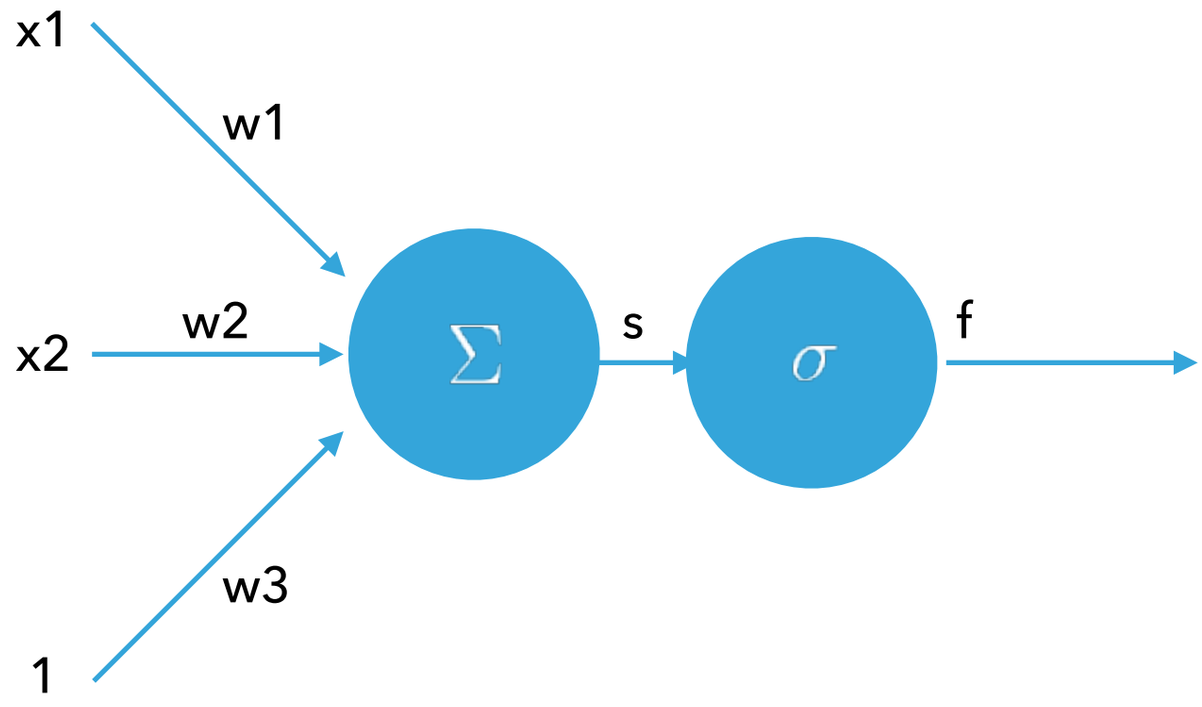

1
w_1
x_1
w_2
w_3
x_2
$w \cdot x$
Perceptron
$\hat{y}$
Inputs
Bias
Weights
Summation
Activation
Output

PERCEPTRON


Forwardpass
Backwardpass
x
y
$f(x, y)$
z
$\frac{dL}{dx} = \frac{dL}{dz}\frac{dz}{dx}$
$\frac{dL}{dy} = \frac{dL}{dz}\frac{dz}{dy}$
df
$\frac{dL}{dz}$

MAKE AND GATE WITH PERCEPTRON

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

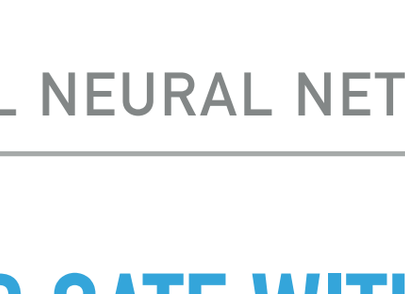


MAKE AND GATE WITH PERCEPTRON

```python
import random
import math

def activate(p):
    return 1.0 / (1.0 + math.exp(-p))

if __name__ == '__main__':
    x = [[0, 0], [0, 1], [1, 0], [1, 1]]
    y = [0, 0, 0, 1]
    w = []

    for i in range(3):
        w.append(random.uniform(-1, 1))

    for epoch in range(501):
        if epoch % 100 == 0:
            print('epoch: ', epoch)
        for x_bar, y_bar in zip(x, y):
            s = x_bar[0] * w[0] + x_bar[1] * w[1] + w[2]
            f = activate(s)
            loss = (f - y_bar)**2
            if epoch % 100 == 0:
                print('Loss: ', loss)
                print(x_bar[0], x_bar[1], round(f, 2))
```

MAKE AND GATE WITH PERCEPTRON

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1



$$\frac{dL}{dw_1} = \frac{dL}{ds}\frac{ds}{dw_1}$$

$$\frac{dL}{dw_2} = \frac{dL}{ds}\frac{ds}{dw_2}$$

$$\frac{dL}{dw_3} = \frac{dL}{ds}\frac{ds}{dw_3}$$

$$\frac{dL}{ds} = \frac{df}{ds}\frac{dL}{df}$$

Σ

σ

$$\frac{dL}{df}$$

MAKE AND GATE WITH PERCEPTRON

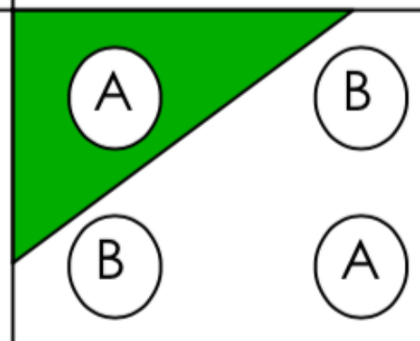
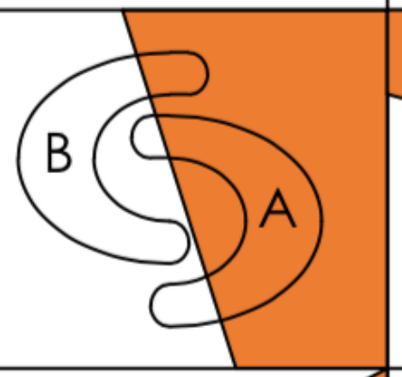
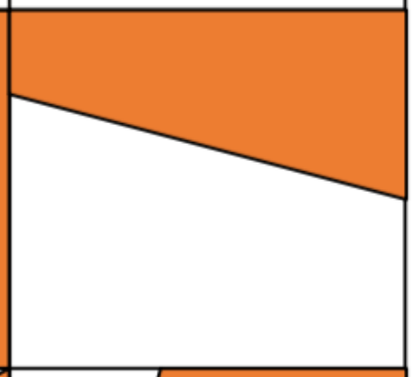
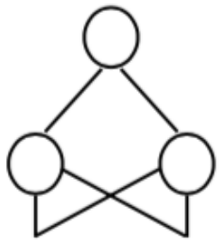
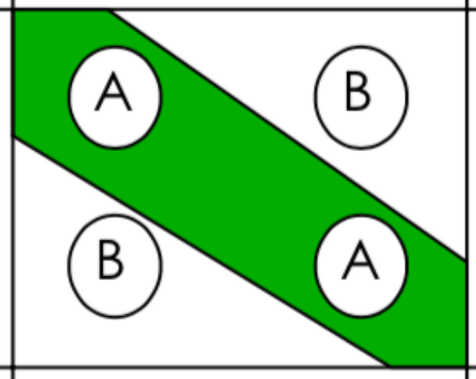
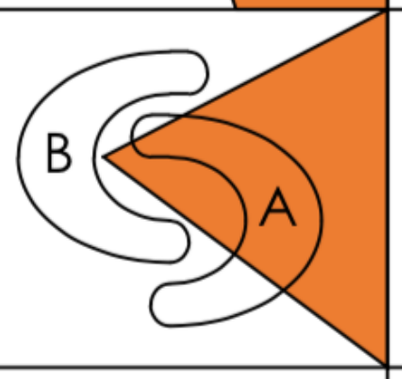
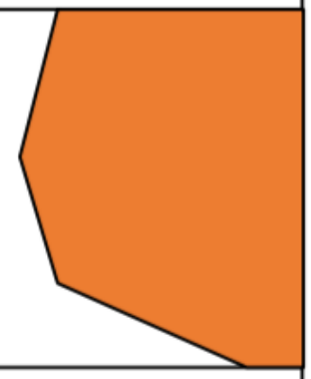
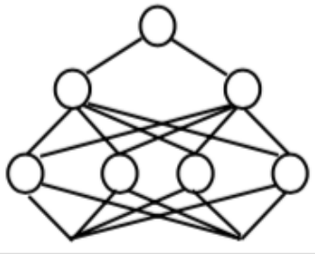
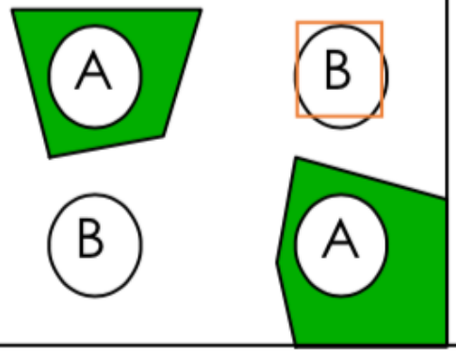
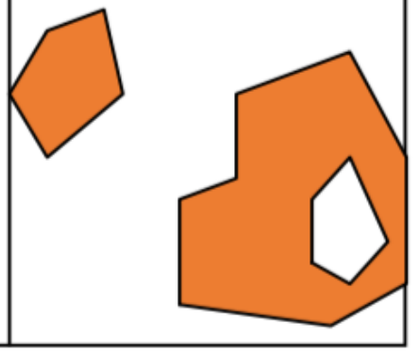
```python
import random
import math

def activate(p):
    return 1.0 / (1.0 + math.exp(-p))

if __name__ == '__main__':
    x = [[0, 0], [0, 1], [1, 0], [1, 1]]
    y = [0, 0, 0, 1]
    w = []

    for i in range(3):
        w.append(random.uniform(-1, 1))

    for epoch in range(501):
        if epoch % 100 == 0:
            print('epoch: ', epoch)
            print("w: ", w)
        for x_bar, y_bar in zip(x, y):
            s = x_bar[0] * w[0] + x_bar[1] * w[1] + w[2]
            f = activate(s)
            loss = (f - y_bar)**2
            if epoch % 100 == 0:
                print('Loss: ', loss)
                print(x_bar[0], x_bar[1], round(f, 2))

            dLdf = 2*(f - y_bar)
            dfds = activate(s)*(1-activate(s))
            dsdw0 = x_bar[0]
            dsdw1 = x_bar[1]
            dsdw2 = 1

            w[0] -= dsdw0 * dfds * dLdf
            w[1] -= dsdw1 * dfds * dLdf
            w[2] -= dsdw2 * dfds * dLdf
```

```
epoch:  500
w:  [5.538945620318824, 5.5318751010436715, -8.382551971239408]
Loss:  5.233703759954025e-08
0 0 0.0
Loss:  0.0029862218276803477
0 1 0.05
Loss:  0.002994274887514747
1 0 0.05
Loss:  0.004228987506300576
1 1 0.93
```

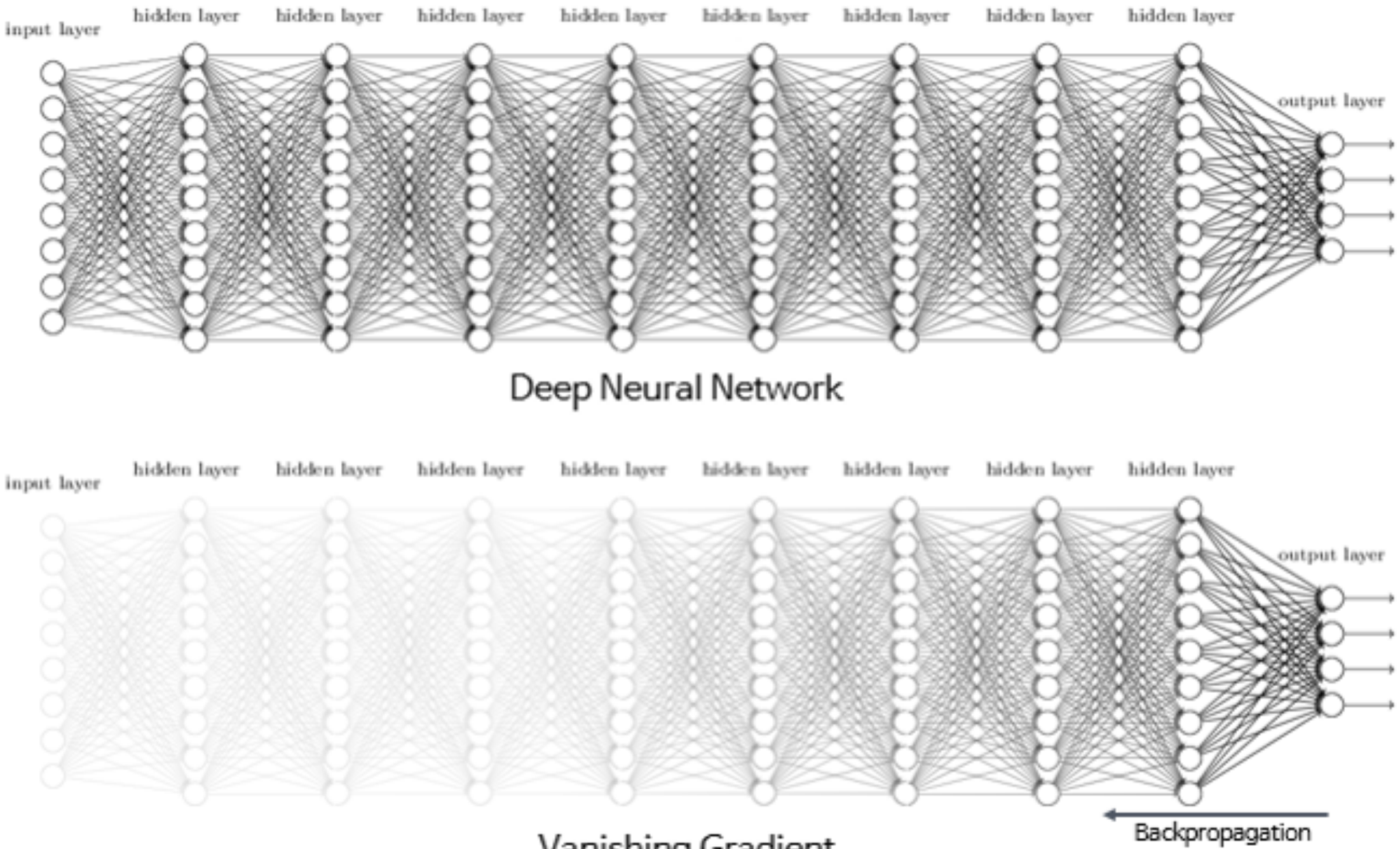
WHY NEURAL NETWORK?

Non Linearity

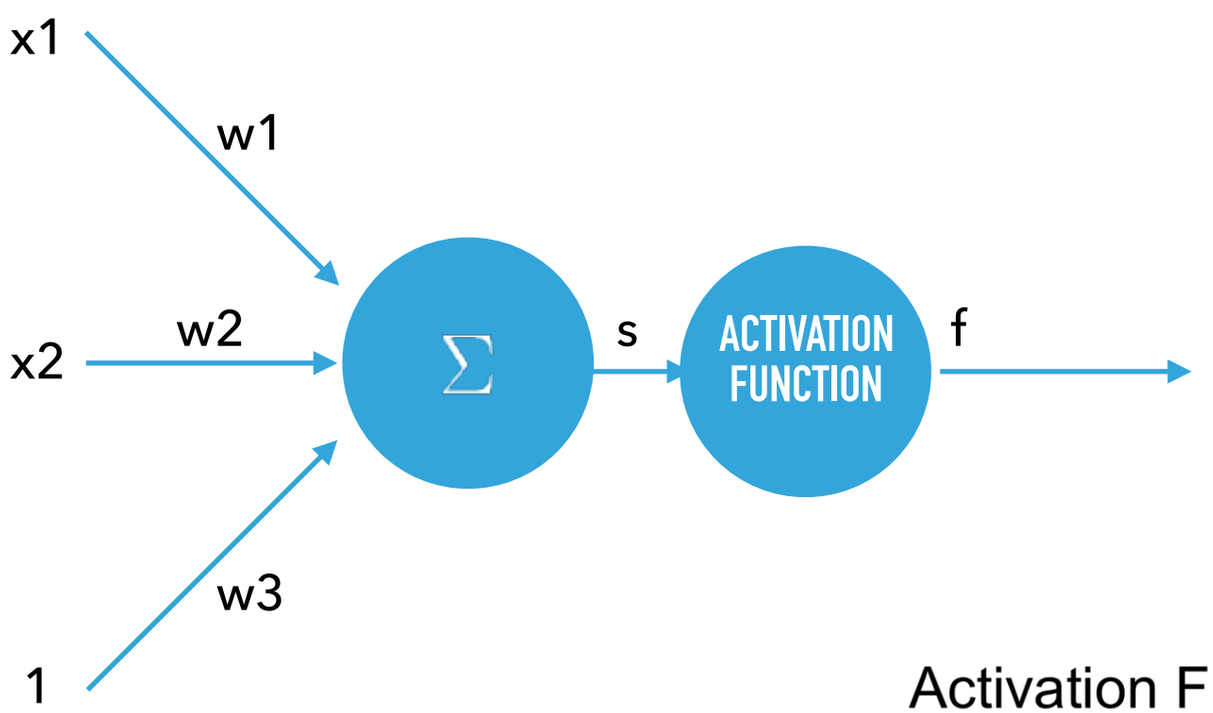
WHY NEURAL NETWORK?

Structure	*Types of Decision Regions*	*Exclusive-OR Problem*	*Classes with Meshed regions*	*Most General Region Shapes*
Single-Layer 	*Half Plane Bounded By Hyper plane*			
Two-Layer 	*Convex Open Or Closed Regions*			
Three-Layer 	Arbitrary (Complexity Limited by No. of Nodes)			

THEN DEEPER THE BETTER?



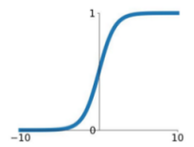
ACTIVATION FUNCTION?



Activation Functions

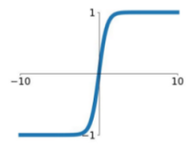
Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



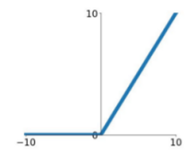
tanh

$\tanh(x)$



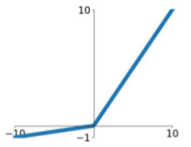
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

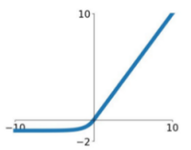


Maxout

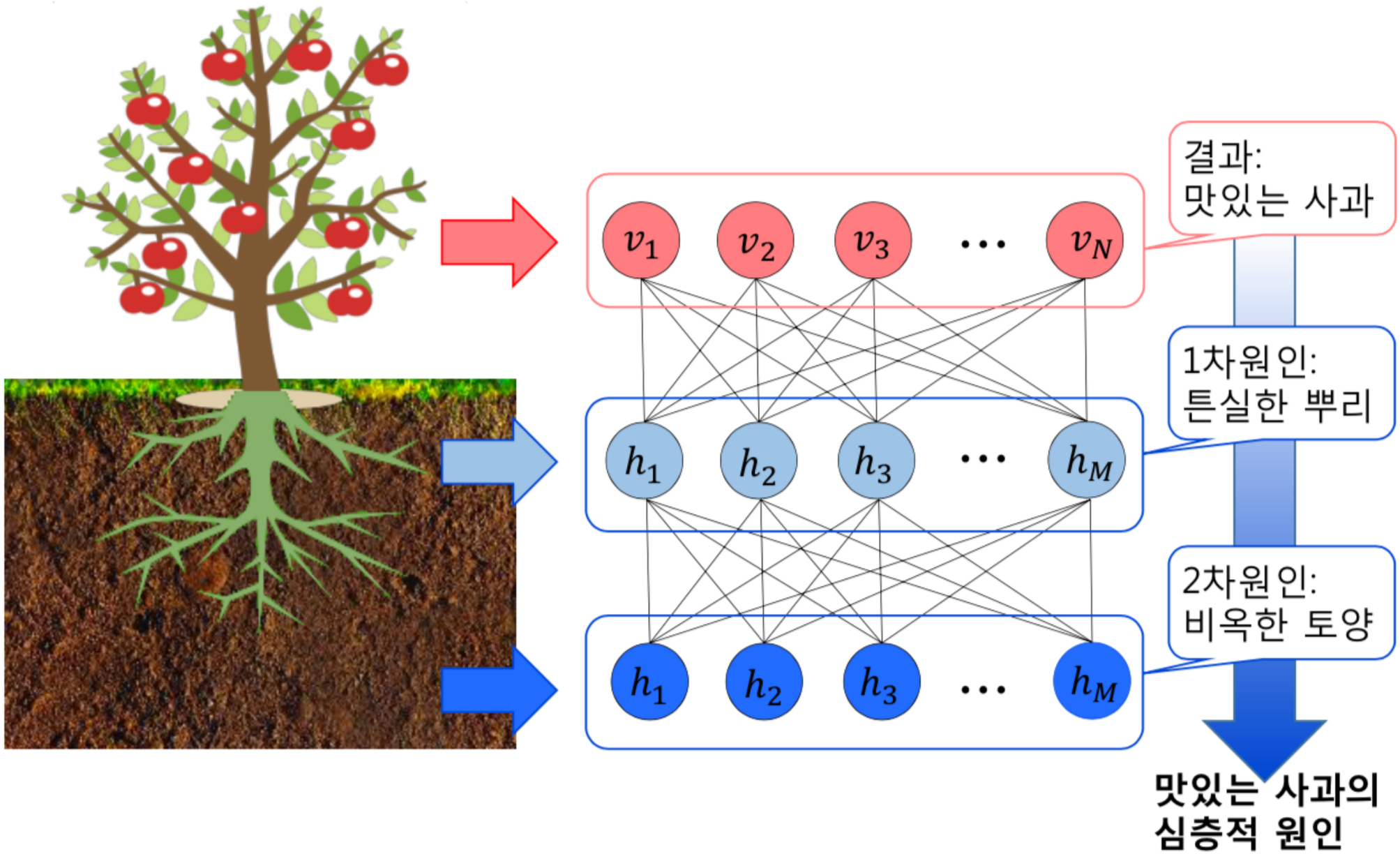
$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

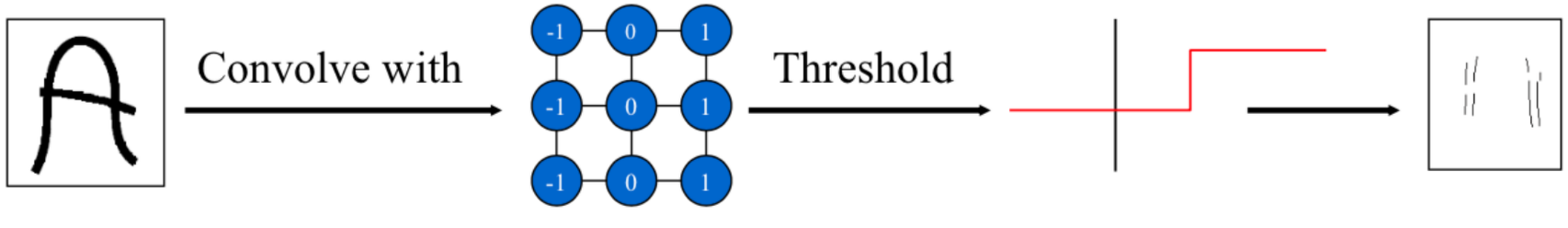


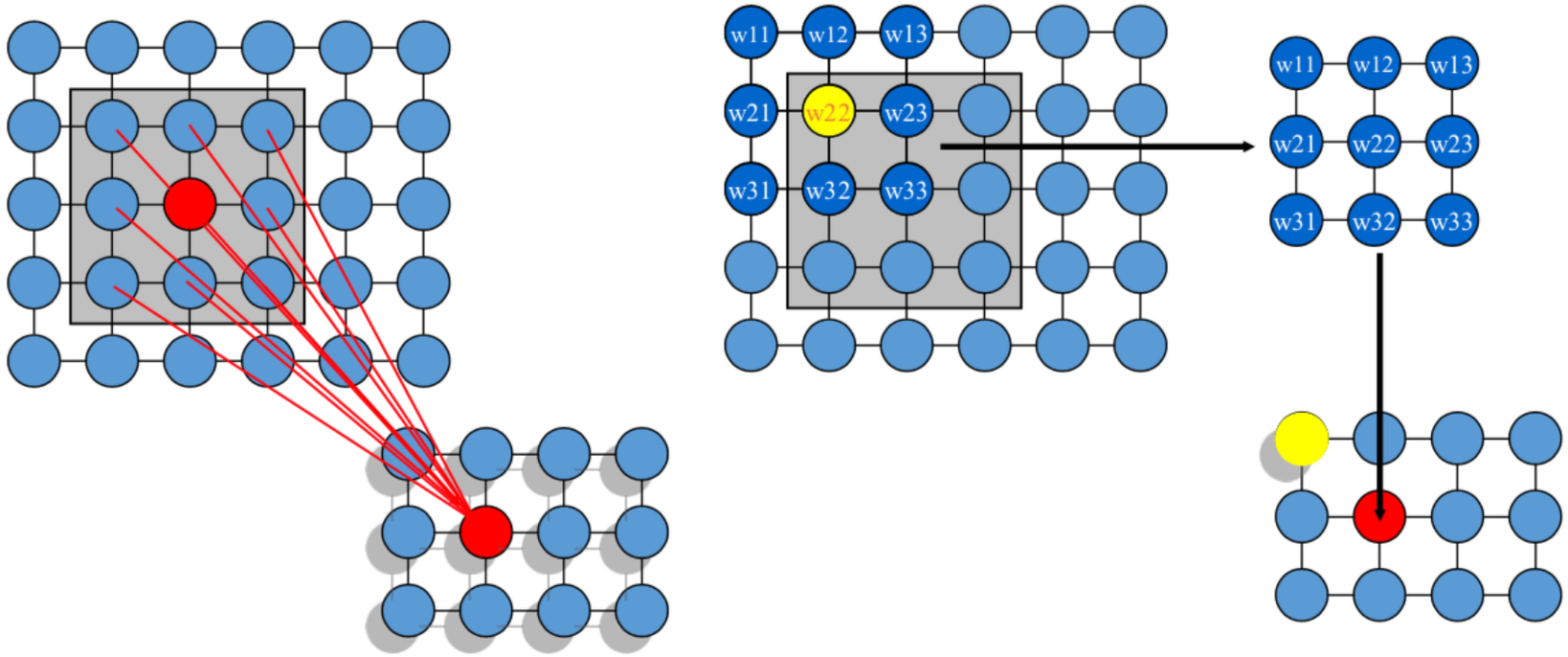
WHAT IS DEEP LEARNING ACTUALLY?



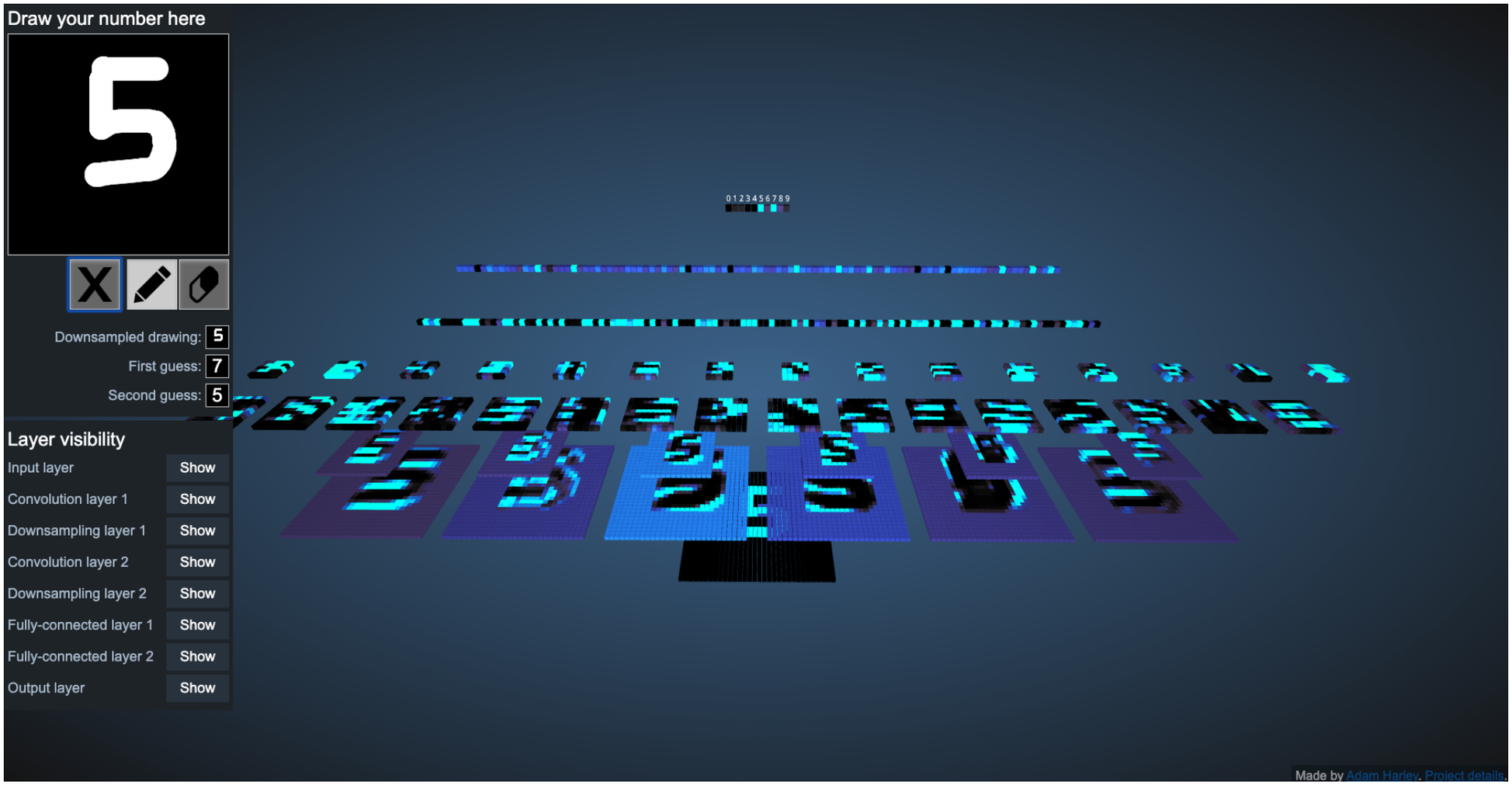
Deep means '심층' not '깊은'

DEEP CONVOLUTIONAL NEURAL NETWORK





DEEP CONVOLUTIONAL NEURAL NETWORK



https://cs.ryerson.ca/~aharley/vis/conv/