

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

## 1. INSTALL & SETUP

---

BEFORE WE START...



## 1. INSTALL & SETUP

---

BEFORE WE START...



python

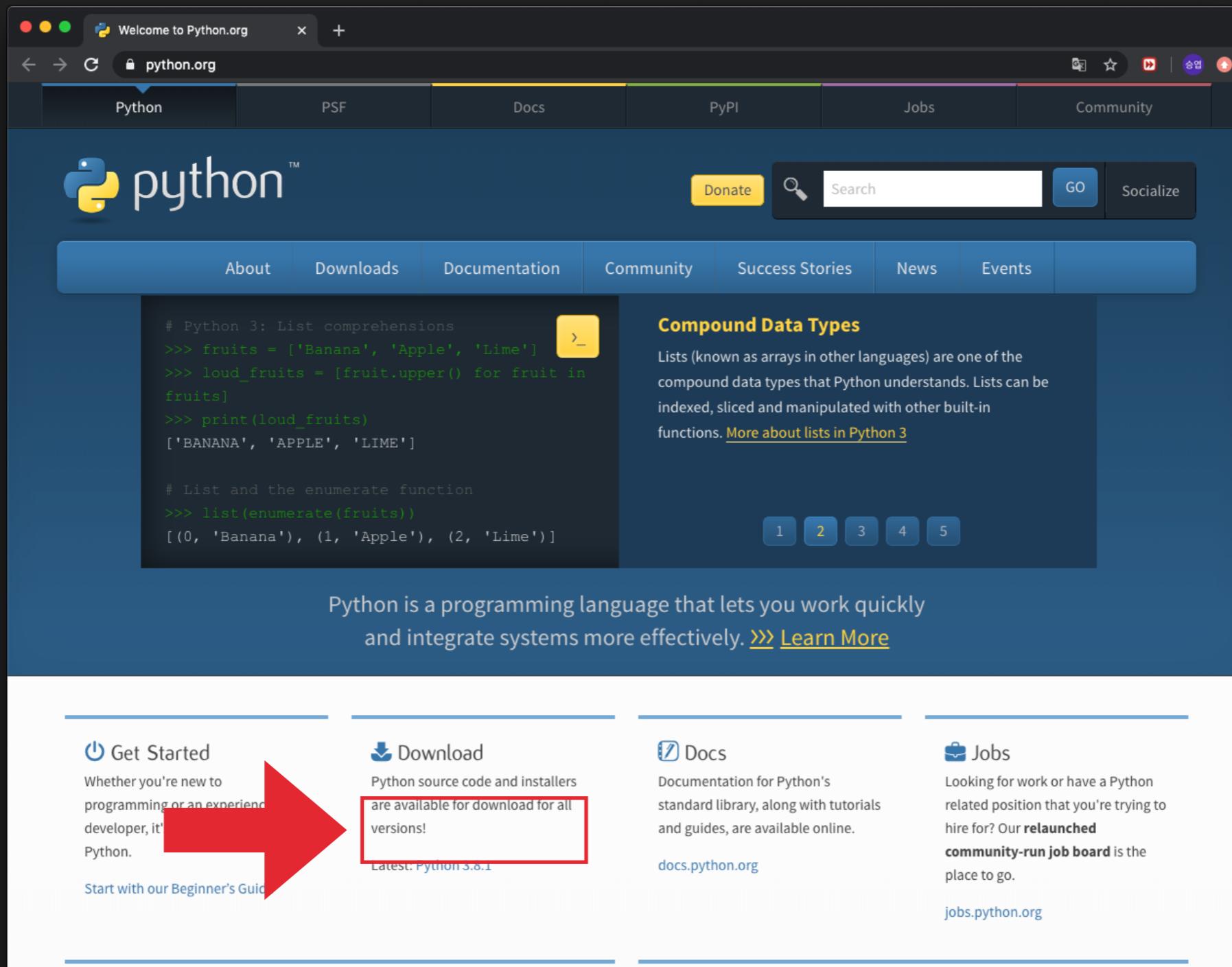


Visual Studio Code



## 1. INSTALL & SETUP

# INSTALL PYTHON



The screenshot shows the Python.org homepage. At the top, there's a navigation bar with links for Python, PSF, Docs, PyPI, Jobs, and Community. Below the header, the Python logo is prominently displayed. A search bar and a "Socialize" button are also present. The main content area features two code snippets:

```
# Python 3: List comprehensions
>>> fruits = ['Banana', 'Apple', 'Lime']
>>> loud_fruits = [fruit.upper() for fruit in
fruits]
>>> print(loud_fruits)
['BANANA', 'APPLE', 'LIME']

# List and the enumerate function
>>> list(enumerate(fruits))
[(0, 'Banana'), (1, 'Apple'), (2, 'Lime')]
```

To the right of the code, there's a section titled "Compound Data Types" with a brief description and a link to "More about lists in Python 3". Below the code snippets, there's a navigation bar with links for 1, 2, 3, 4, and 5. The footer contains four main sections: "Get Started", "Download", "Docs", and "Jobs". A large red arrow points from the "Get Started" section towards the "Download" section, which has a box around the text "are available for download for all versions!".

<https://python.org>

## 1. INSTALL & SETUP

---

# INSTALL PYTHON

## Files

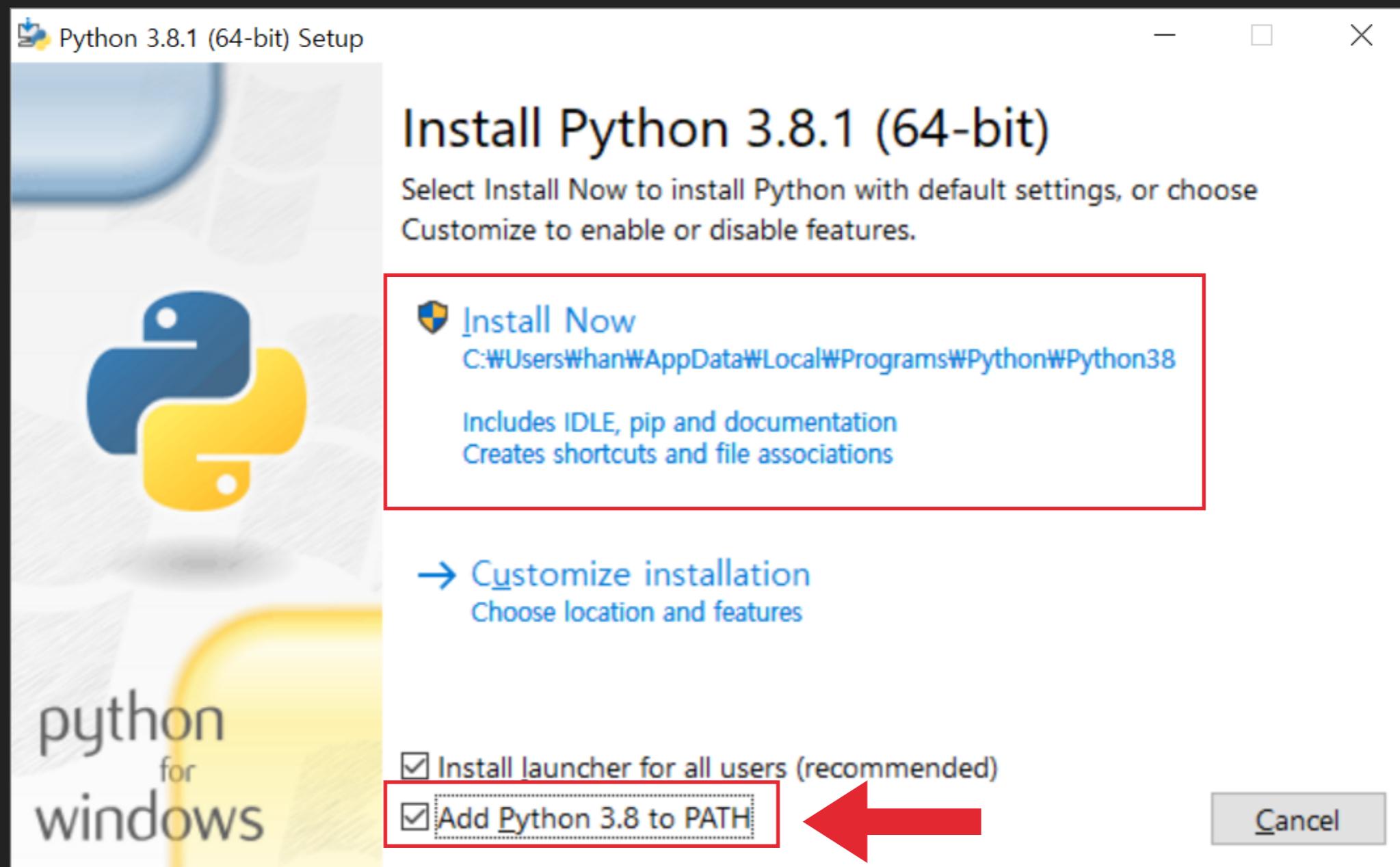
Version	Operating System	Description	MD5 Sum	File Size	GPG
<a href="#">Gzipped source tarball</a>	Source release		f215fa2f55a78de739c1787ec56b2bcd	23978360	<a href="#">SIG</a>
<a href="#">XZ compressed source tarball</a>	Source release		b3fb85fd479c0bf950c626ef80cacb57	17828408	<a href="#">SIG</a>
<a href="#">macOS 64-bit installer</a>	Mac OS X	for OS X 10.9 and later	d1b09665312b6b1f4e11b03b6a4510a3	29051411	<a href="#">SIG</a>
<a href="#">Windows help file</a>	Windows		f6bbf64cc36f1de38fbf61f625ea6cf2	8480993	<a href="#">SIG</a>
<a href="#">Windows x86-64 embeddable zip file</a>	Windows	for AMD64/EM64T/x64	4d091857a2153d9406bb5c522b211061	8013540	<a href="#">SIG</a>
<a href="#">Windows x86-64 executable installer</a>	Windows	for AMD64/EM64T/x64	3e4c42f5ff8fcfbe6a828c912b7afdb1	27543360	<a href="#">SIG</a>
<a href="#">Windows x86-64 web-based installer</a>	Windows	for AMD64/EM64T/x64	662961733cc947839a73302789df6145	1363800	<a href="#">SIG</a>
<a href="#">Windows x86 embeddable zip file</a>	Windows		980d5745a7e525be5abf4b443a00f734	7143308	<a href="#">SIG</a>
<a href="#">Windows x86 executable installer</a>	Windows		2d4c7de97d6fc8231fc3dec8abf79	26446128	<a href="#">SIG</a>
<a href="#">Windows x86 web-based installer</a>	Windows		d21706bdac544e7a968e32bbb0520f51	1325432	<a href="#">SIG</a>



## 1. INSTALL & SETUP

---

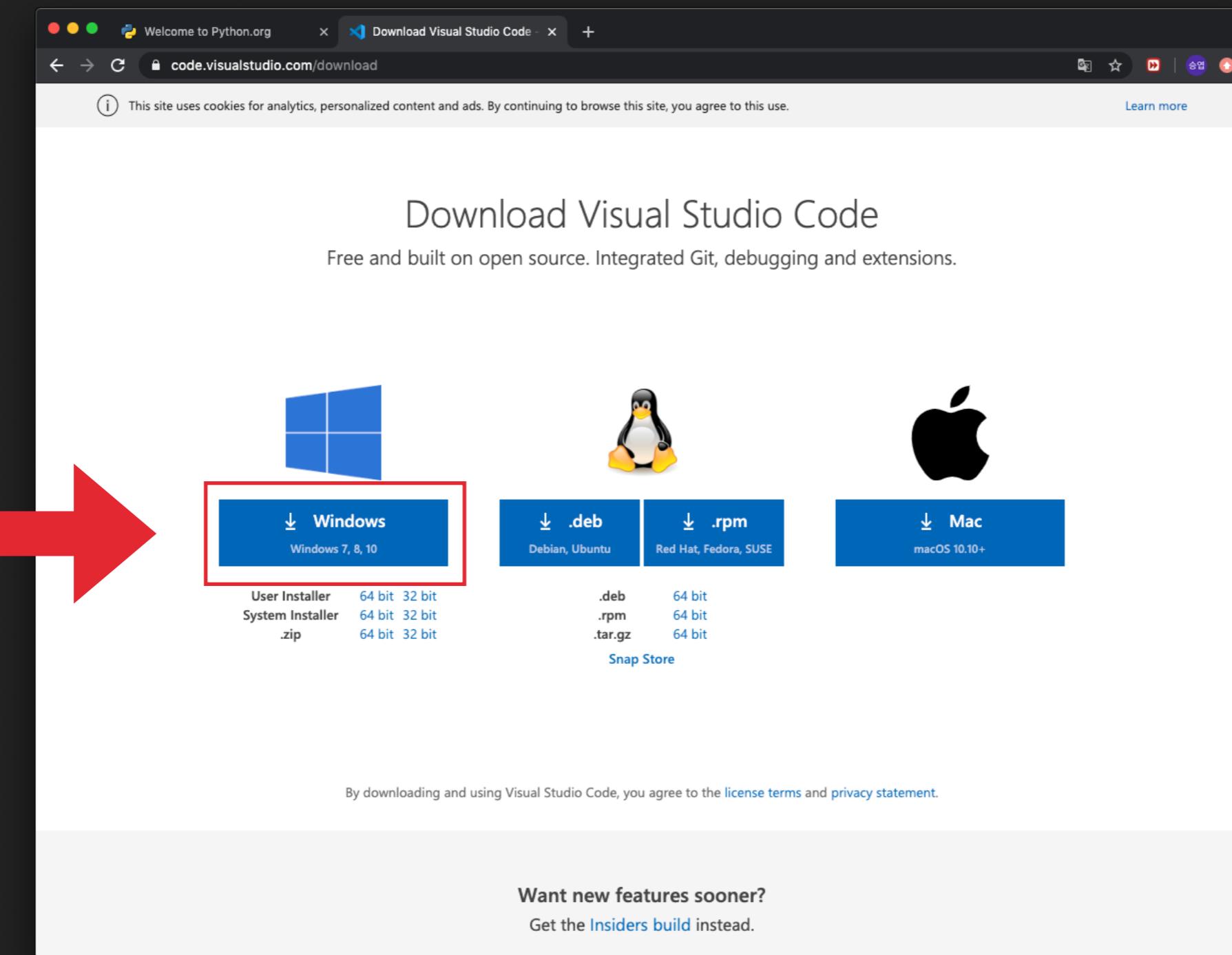
### INSTALL PYTHON



## 1. INSTALL & SETUP

---

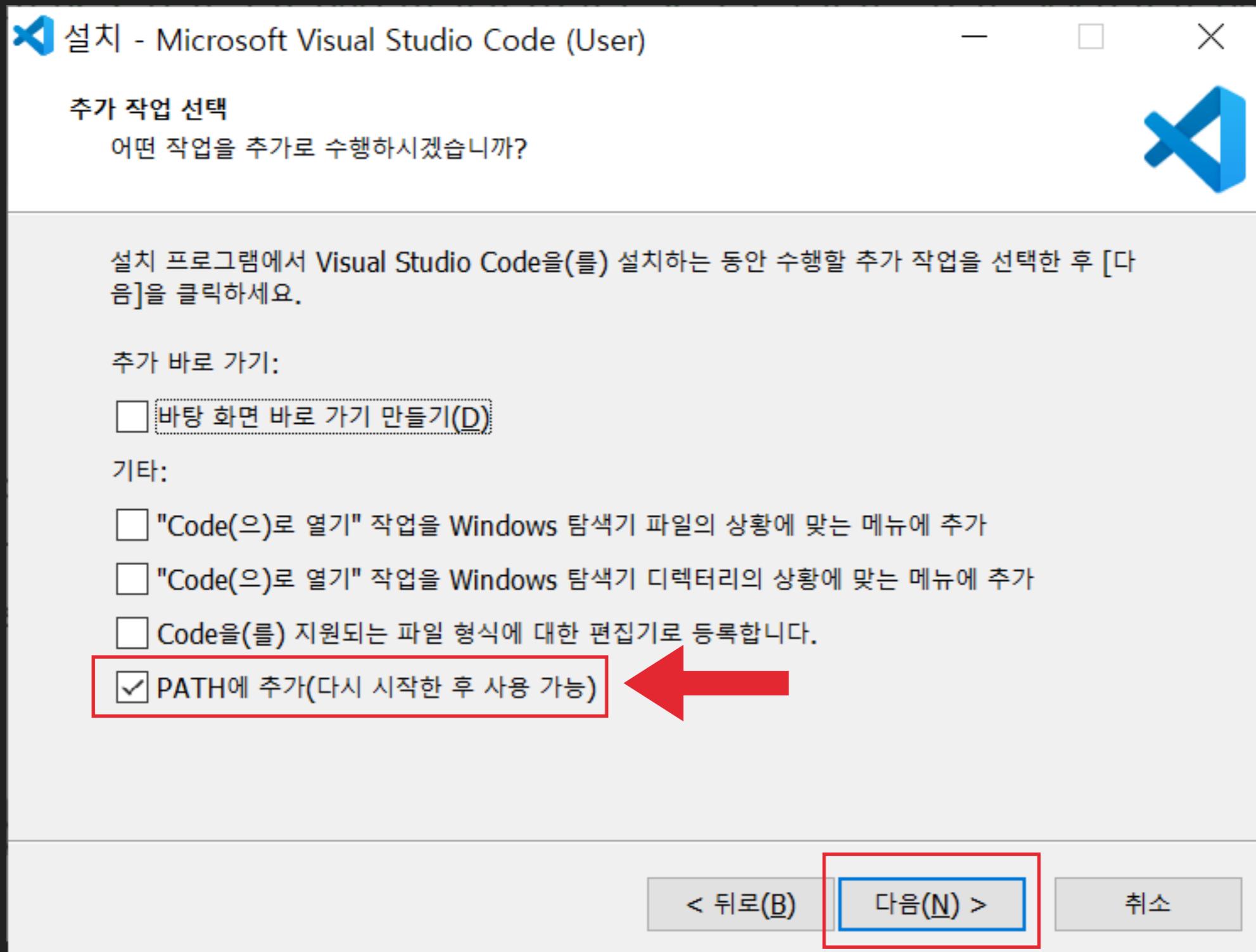
# INSTALL VISUAL STUDIO CODE



<https://code.visualstudio.com/download>

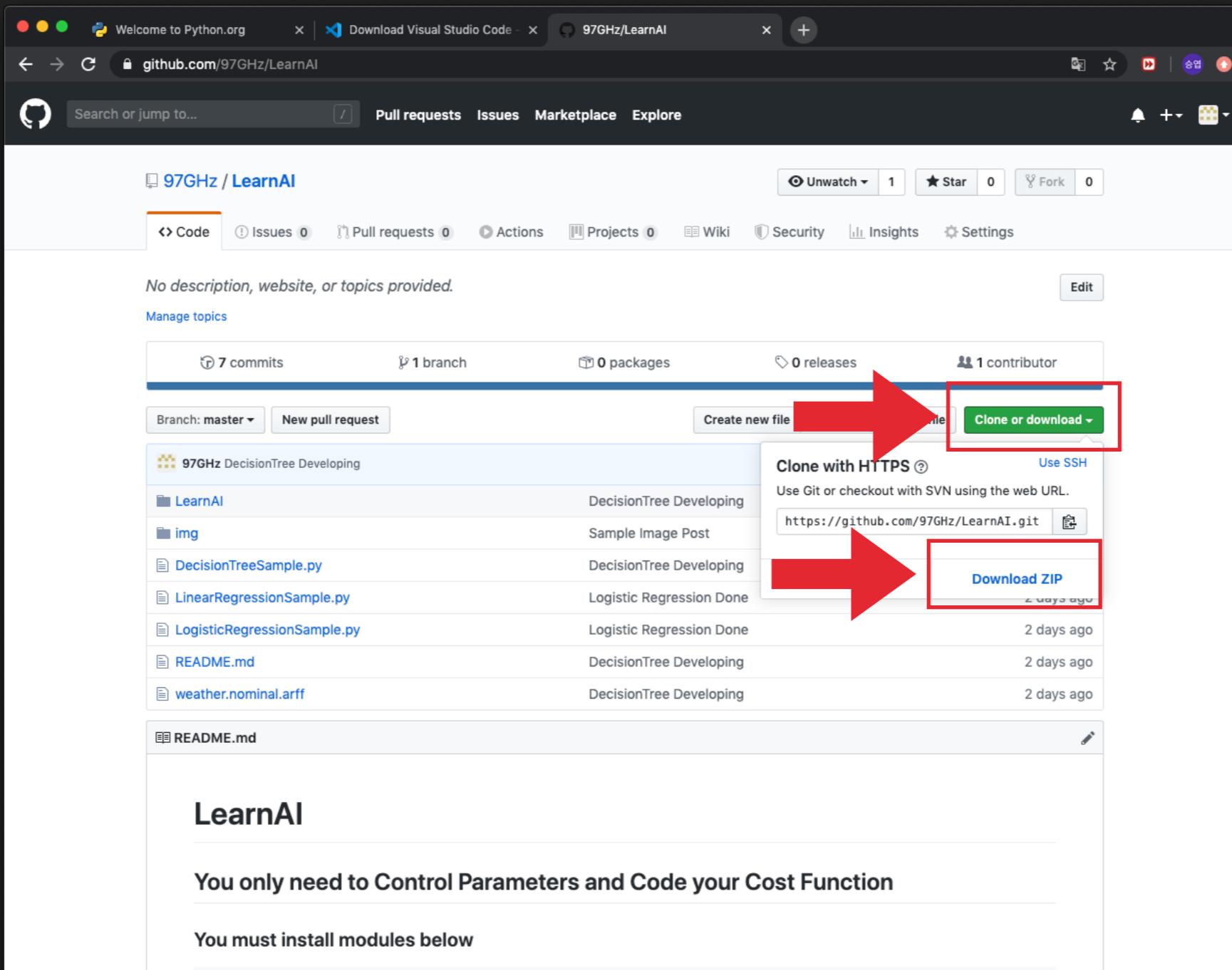
## 1. INSTALL & SETUP

# INSTALL VISUAL STUDIO CODE



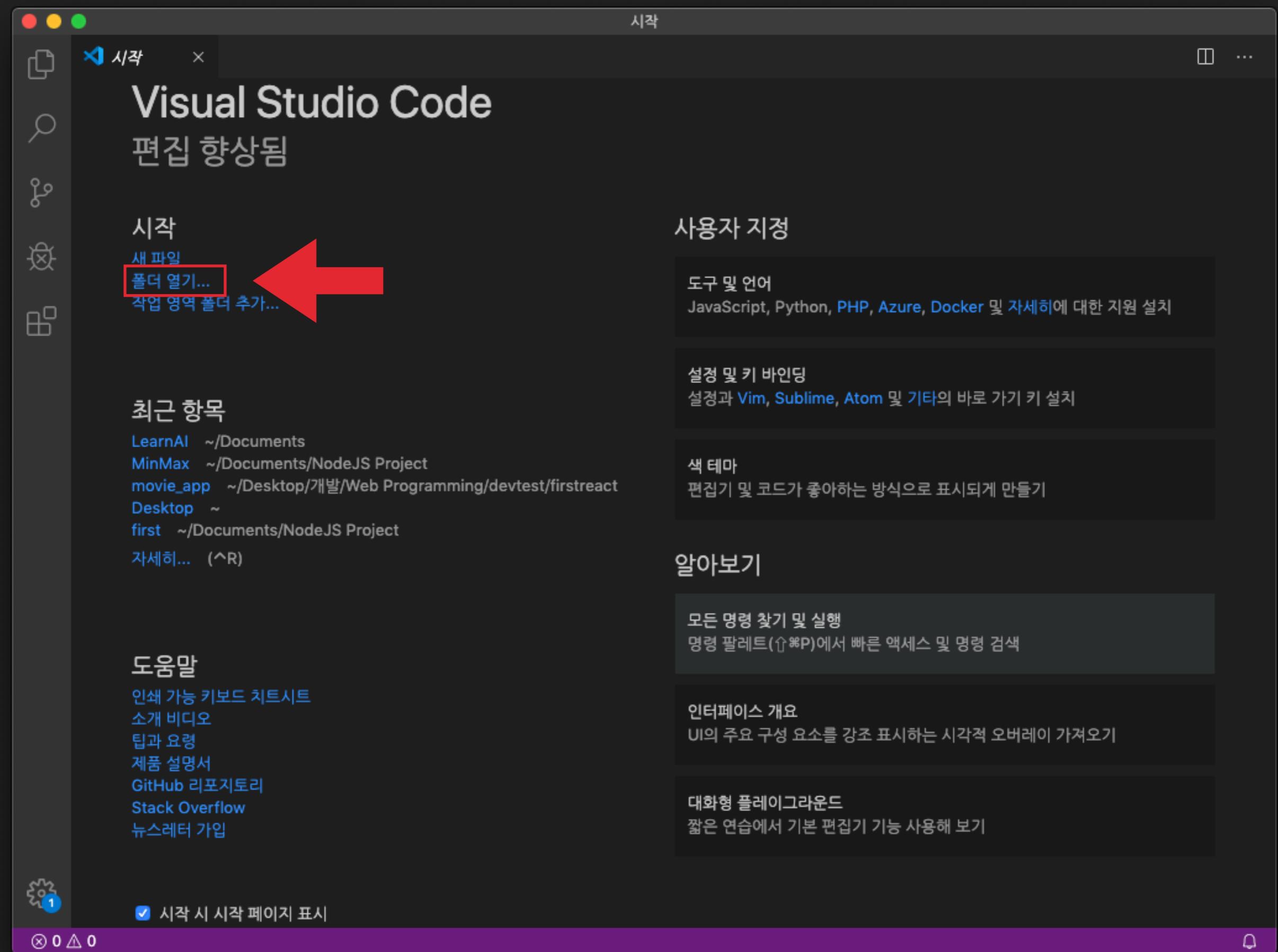
## 1. INSTALL & SETUP

# DOWNLOAD MODULES AND SAMPLES

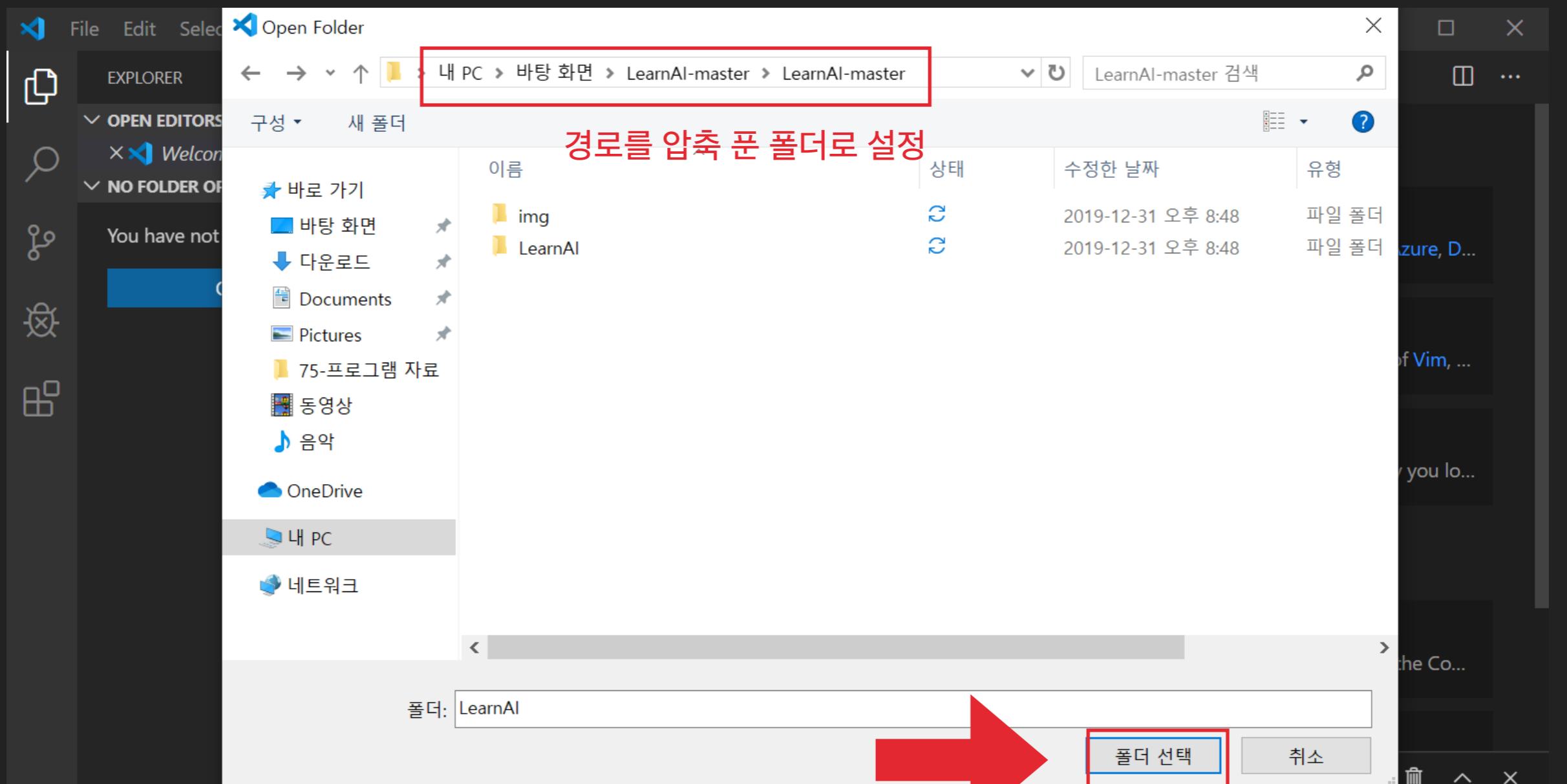


<https://github.com/97GHz/LearnAI>

# 1. INSTALL & SETUP



# 1. INSTALL & SETUP



Windows PowerShell  
Copyright (C) Microsoft Corporation. All rights reserved.

새로운 크로스 플랫폼 PowerShell 사용 <https://aka.ms/pscore6>

PS C:\Users\han>

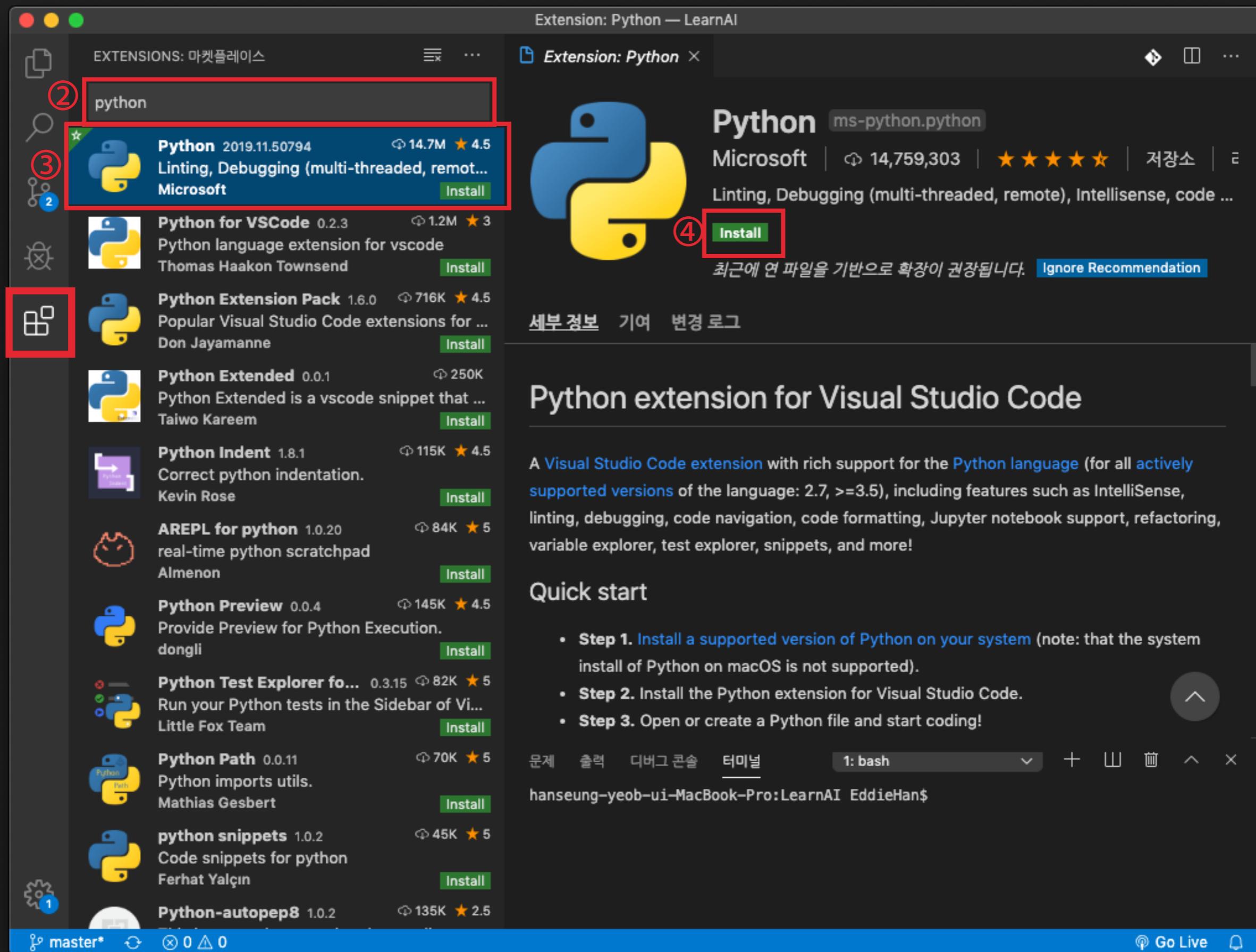


> OUTLINE

⊗ 0 △ 0

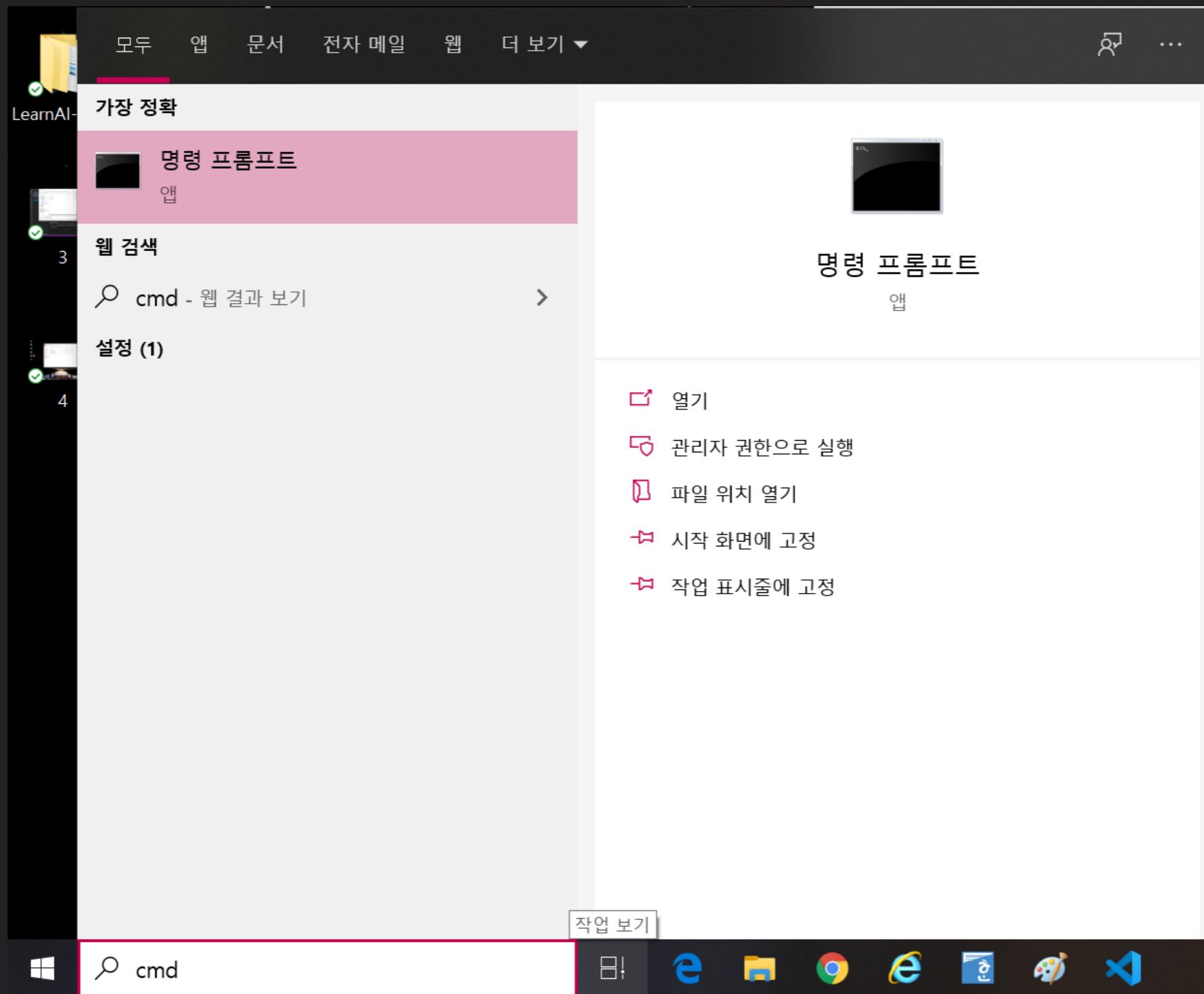
☺ 🔍

# 1. INSTALL & SETUP



## 1. INSTALL & SETUP

# INSTALL PYTHON MODULES



## 1. INSTALL & SETUP

---

# INSTALL PYTHON MODULES



A screenshot of a Windows Command Prompt window titled "명령 프롬프트". The window shows the following text:

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

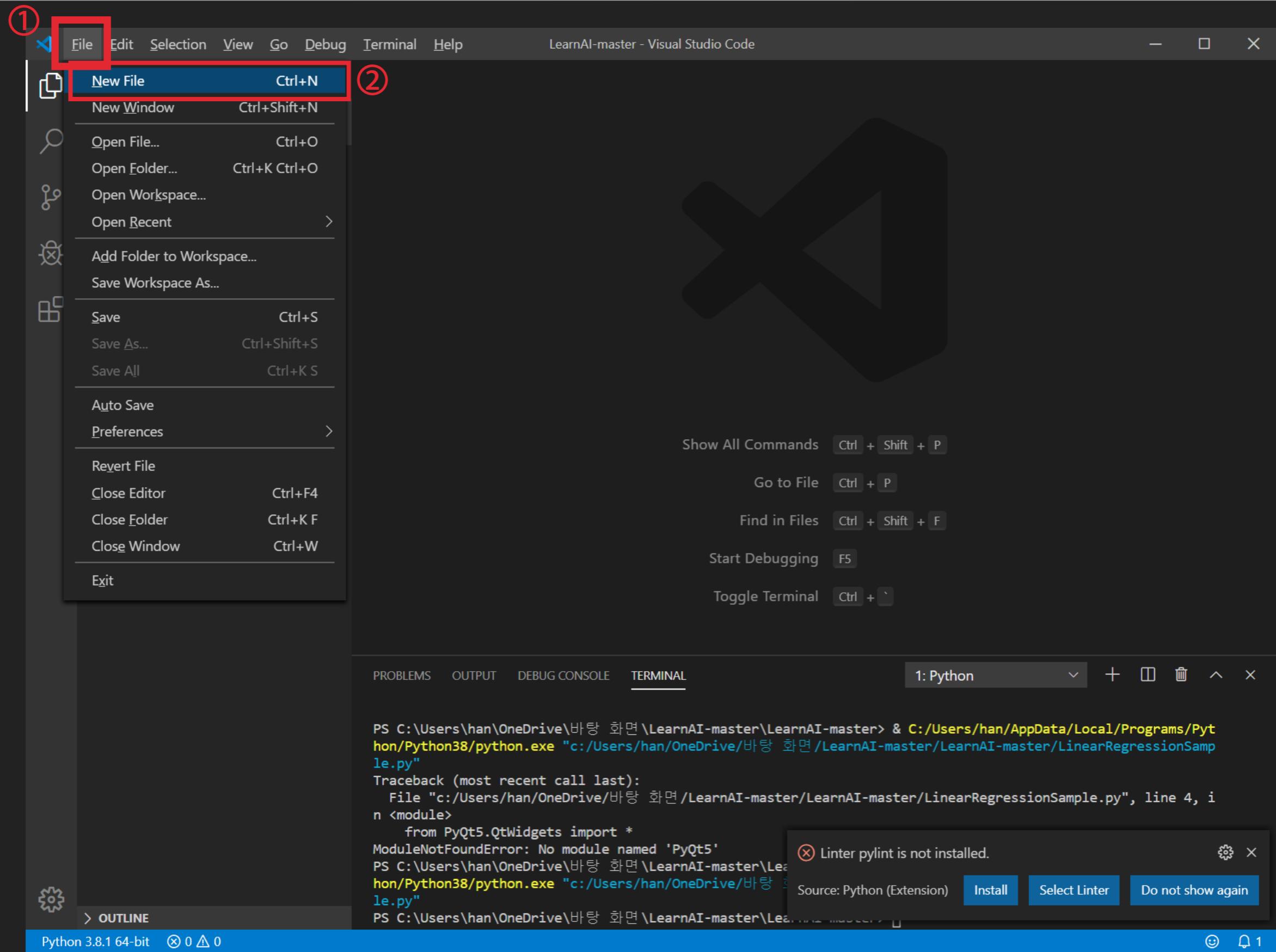
C:\Users\han>
```

Inside a white rectangular box with a black border, the following commands are listed:

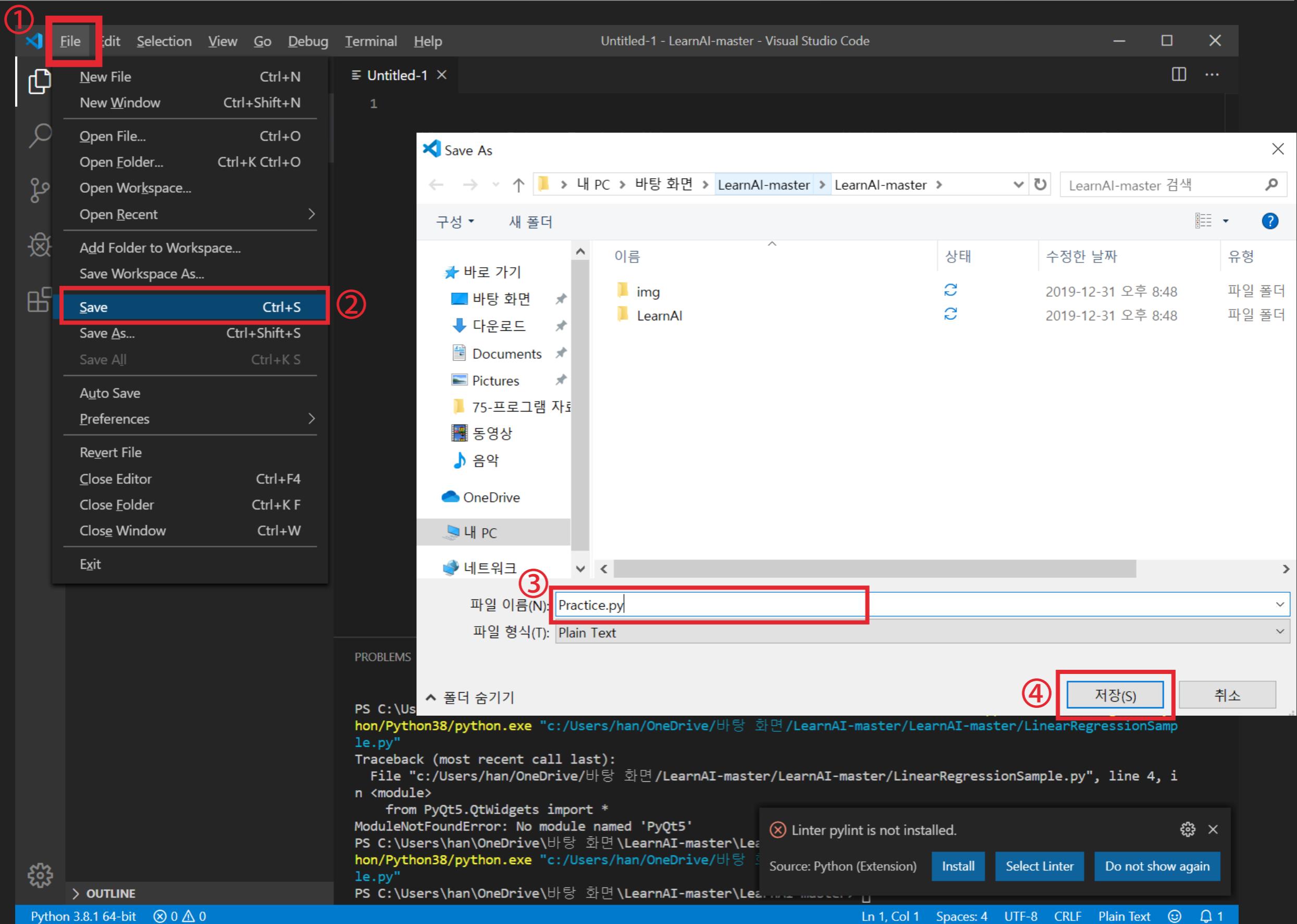
```
pip install PyQt5
pip install numpy
pip install matplotlib
```

To the right of the box, the text "대소문자 주의!" (Pay attention to case sensitivity!) is written in red.

# 1. INSTALL & SETUP



# 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_()
```

# 1. INSTALL & SETUP

## FOLLOW UP

The image shows a development environment with two main windows. On the left is a code editor for a file named `Practice.py`. The code implements a linear regression application using PyQt5. It defines a `Calc()` function that updates a screen (presumably a UI element not shown). The main part of the code sets up the application, defines data points `x` and `y`, initializes parameters `w`, `b`, and `cost`, and connects the `Calc()` function to the application's click event. The code concludes with `app.exec_()` to start the application loop.

```
Practice.py — LearnAI
Practice.py •
Practice.py > ...
1 import sys
2 from PyQt5.QtWidgets import *
3 from LearnAI.LinearRegression import LinearRegression
4
5 def Calc():
6     reg.updateScreen()
7
8
9 if __name__ == '__main__':
10    app = QApplication(sys.argv)
11
12    x = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
13    y = [2, 4, 5, 7, 10, 12, 14, 17, 18, 20]
14    w = [0]
15    b = [0]
16    cost = [0]
17    learningRate = [0.001]
18
19    reg = LinearRegression()
20    reg.setClickEvent(Calc)
21    reg.connectParameter(x, y, w, b, cost, learningRate)
22
23    app.exec_()
24
```

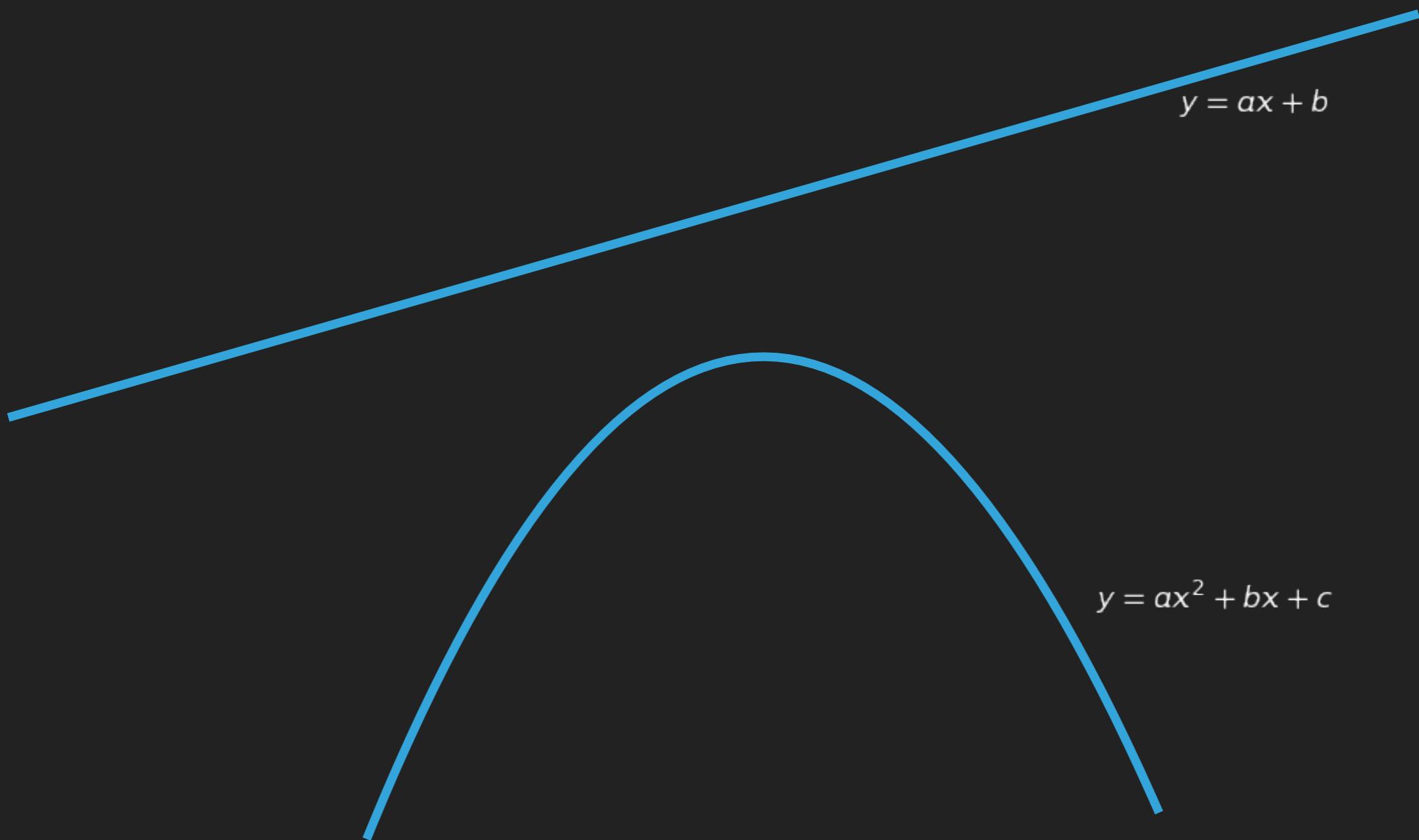
A large red arrow points from the run button in the code editor towards the terminal window. A large blue arrow points from the code editor towards the `Linear Regression` application window.

The right window is titled `Linear Regression` and displays a scatter plot of data points and a graph of the hypothesis function  $H(x) = Wx + B$ . The scatter plot has `x` values from 1 to 10 and `y` values from 0.0 to 20.0. The hypothesis function graph shows a horizontal line at  $y = 0.0$ . The application interface includes a status bar with `Count: 1`, `W: 0`, `B: 0`, and `Prev Cost: 0`, and a control panel with `Learning Rate 0.001` and a `Next` button.

## 2. LINEAR REGRESSION

---

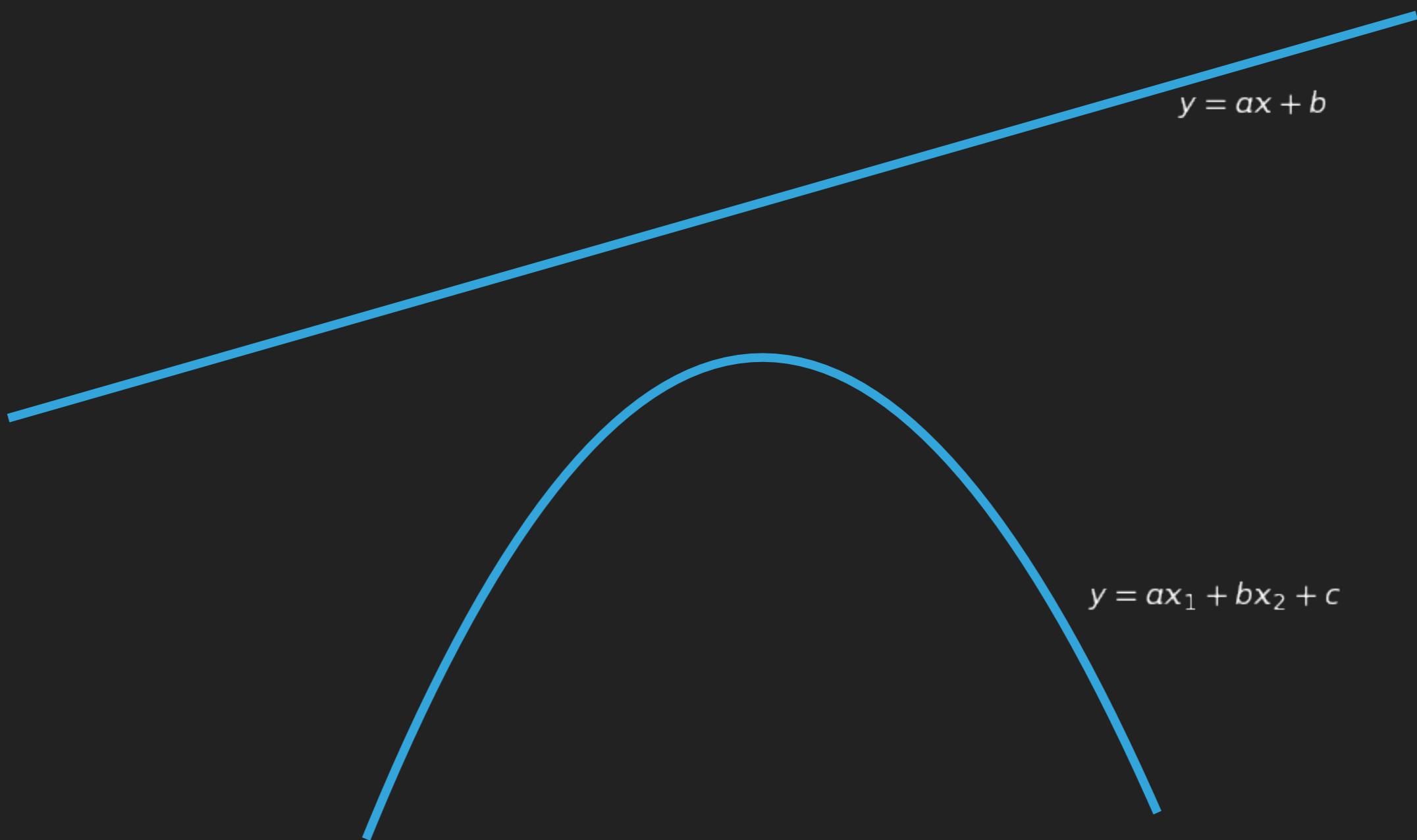
### WHAT IS LINEAR?



## 2. LINEAR REGRESSION

---

### WHAT IS LINEAR?



# WHY LINEAR?

걷는 걸음 수에 따른 체중 변화?

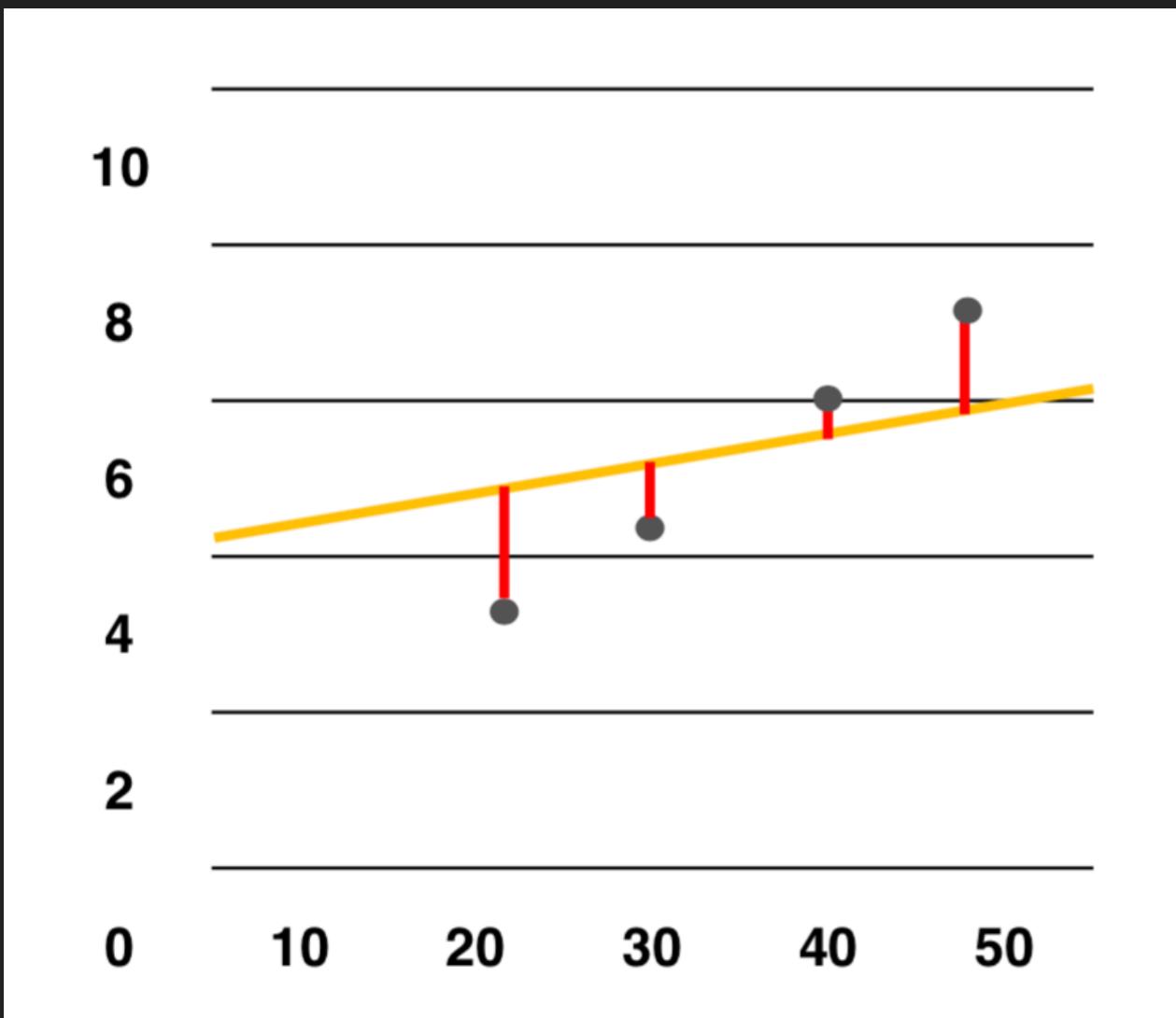
오늘 평균 기온에 따른 어묵 판매량?

평수에 따른 월세 비용?

## 2. LINEAR REGRESSION

---

### WHY LINEAR?



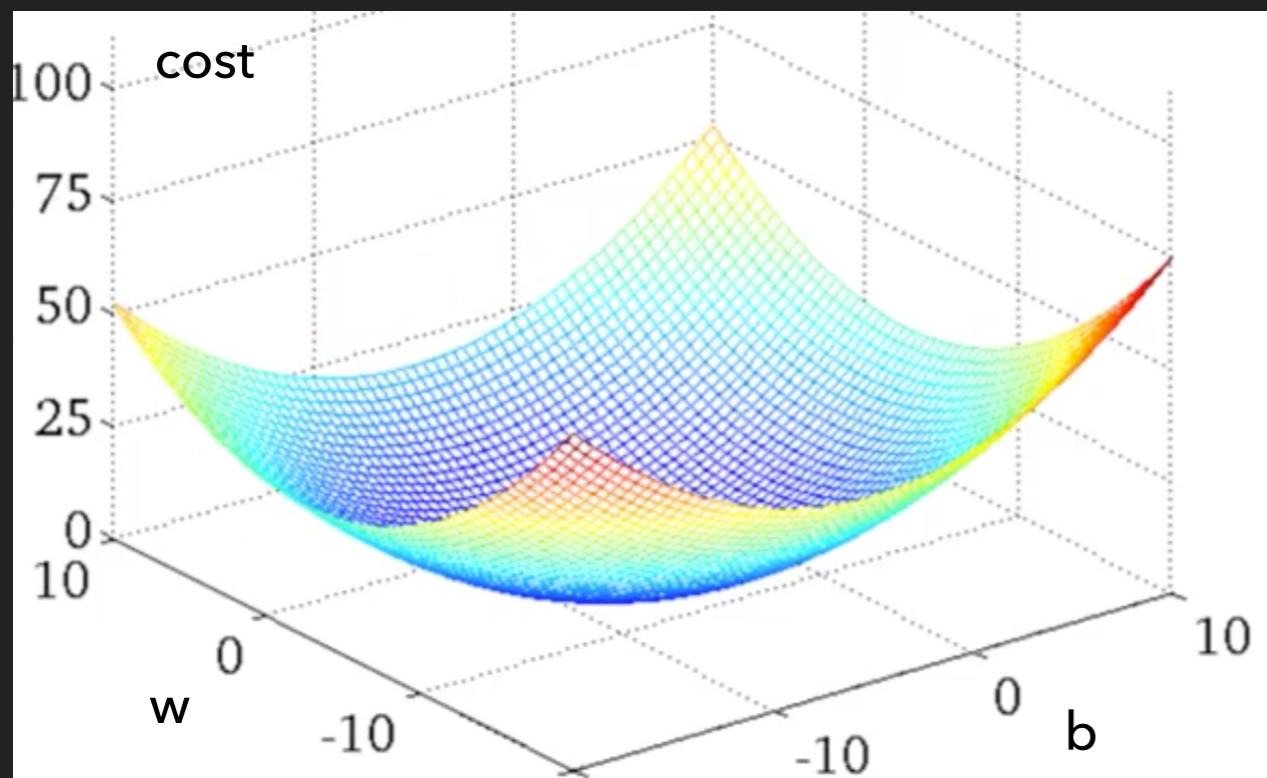
$$H(x) = wx + b$$

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

## 2. LINEAR REGRESSION

---

### WHY LINEAR?



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

## 2. LINEAR REGRESSION

---

### WHY LINEAR?

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

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

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

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

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

Must update weights simultaneously

## 2. LINEAR REGRESSION

---

# CODE REVIEW

```
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_()
```

## 2. LINEAR REGRESSION

---

# 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)]
```

## 2. LINEAR REGRESSION

---

### 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()
```

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

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

## 2. LINEAR REGRESSION

---

### 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

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

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

## 2. LINEAR REGRESSION

---

### 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()
```

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

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

## 2. LINEAR REGRESSION

---

### 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

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

## 2. LINEAR REGRESSION

---

### 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

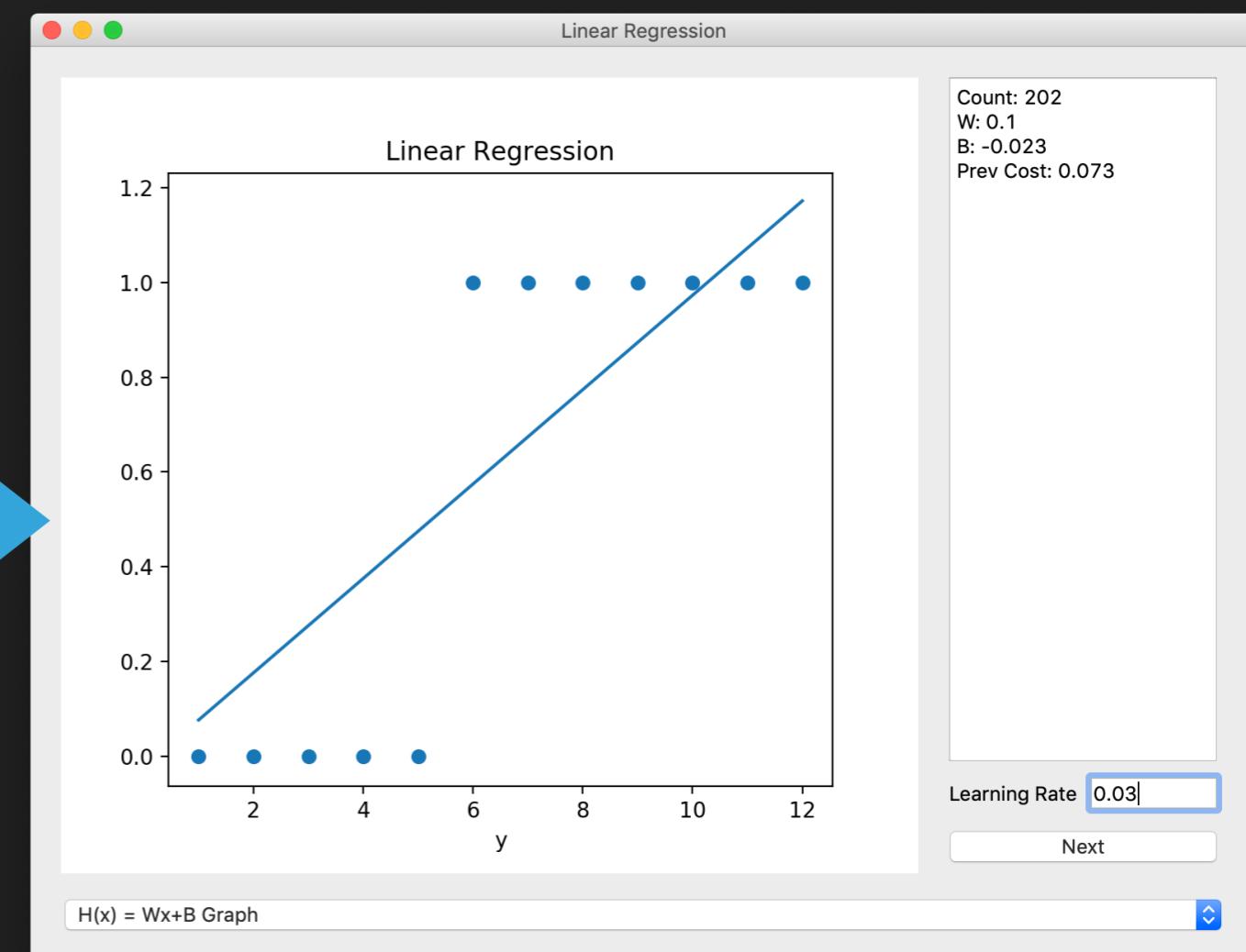
    reg.updateScreen()
```

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

## 2. LINEAR REGRESSION

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



### 3. LOGISTIC REGRESSION

---

## 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]

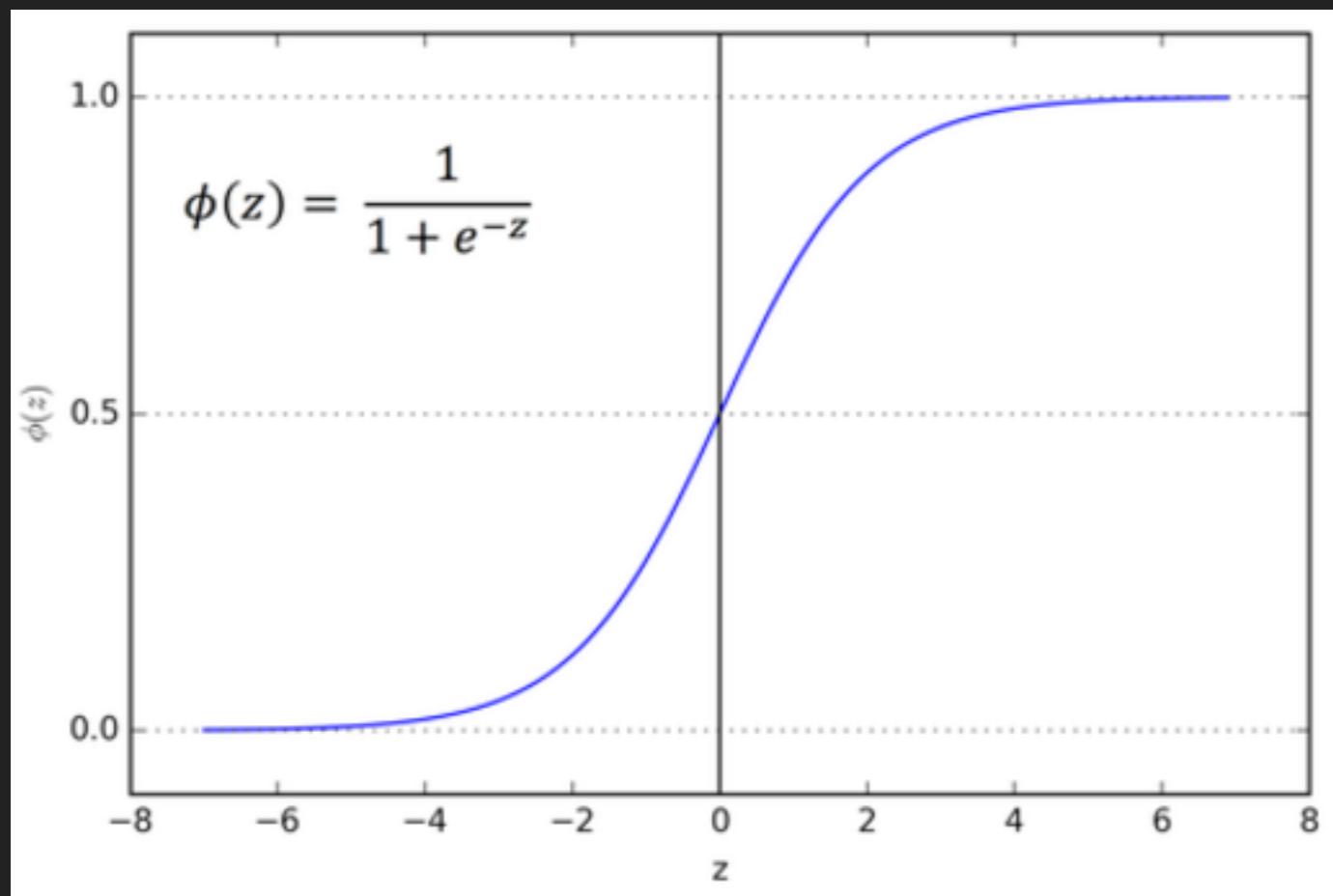
    reg = LogisticRegression()
    reg.setClickEvent(Calc)
    reg.connectParameter(x, y, w, b, cost, learningRate)

    app.exec_()
```

### 3. LOGISTIC REGRESSION

---

## HYPOTHESIS



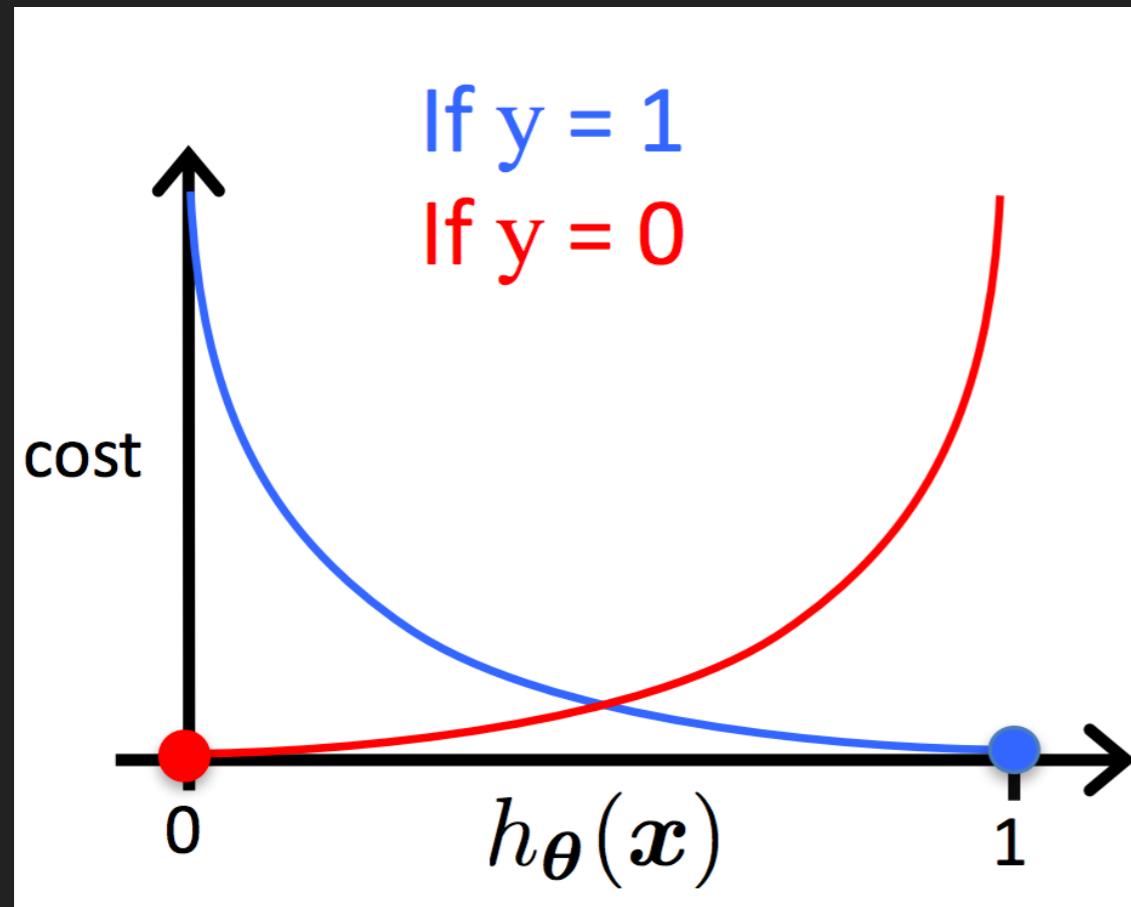
$$z = wx + b$$

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

### 3. LOGISTIC REGRESSION

---

## COST FUNCTION



$$z = wx + b$$

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

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

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

### 3. LOGISTIC REGRESSION

---

## COST FUNCTION

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

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

### 3. LOGISTIC REGRESSION

---

## COST FUNCTION

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

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

### 3. LOGISTIC REGRESSION

---

## 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()
```

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

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

### 3. LOGISTIC REGRESSION

---

## 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

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

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

### 3. LOGISTIC REGRESSION

---

## 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()
```

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

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

### 3. LOGISTIC REGRESSION

---

## UPDATE COST

```
def Calc():
    aw = 0.0
    ab = 0.0
    cost[0] = [REDACTED]
    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] -= [REDACTED]
        else:
            aw += (s-1)*xi
            ab += (s-1)
            cost[0] -= [REDACTED]
    aw /= len(x)
    ab /= len(x)
    cost[0] /= [REDACTED]
    w[0] -= learningRate[0]*aw
    b[0] -= learningRate[0]*ab

    reg.updateScreen()
```

Fill the Blanks

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

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

### 3. LOGISTIC REGRESSION

---

## 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()
```

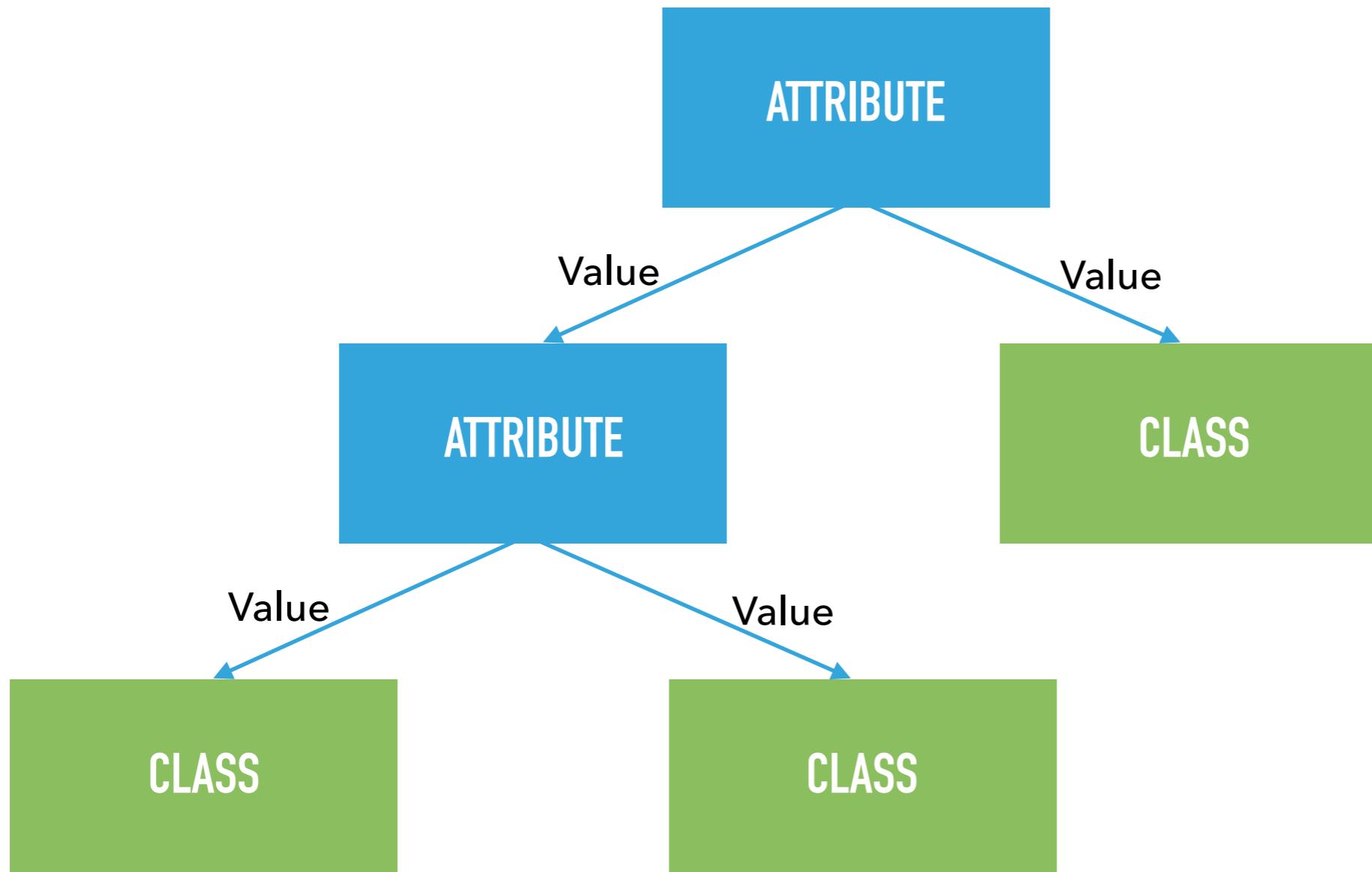
$$cost(w, b) = \frac{1}{m} \sum_{i=1}^m c(H(x), y)$$

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

## 4. DECISION TREE

---

# WHAT IS DECISION TREE?

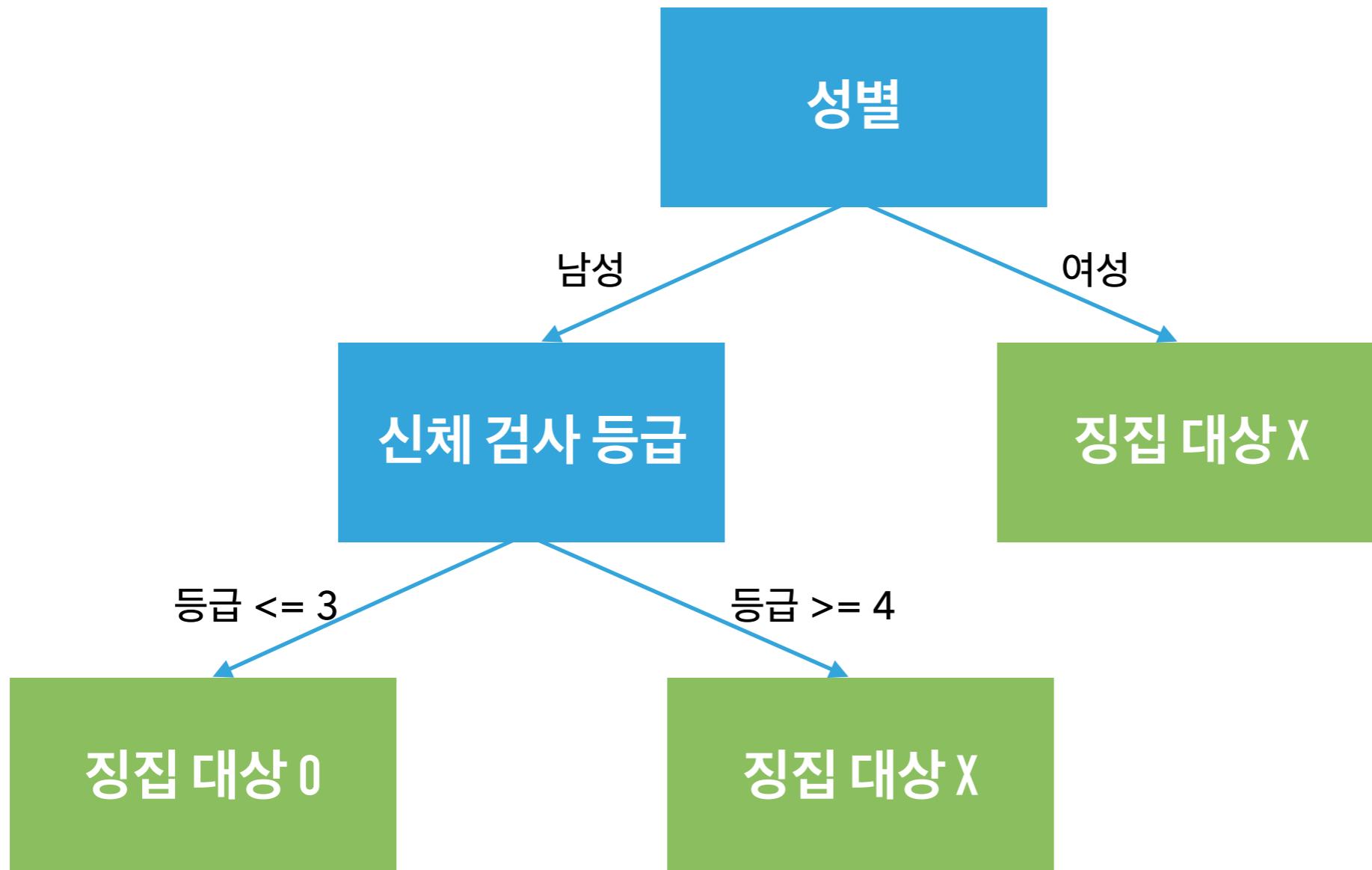


Attribute = {Value1, Value2, Value3, ...}

## 4. DECISION TREE

---

### WHAT IS DECISION TREE?



성별 = {남성, 여성}

등급 = {1, 2, 3, 4, 5, 6}

#### 4. DECISION TREE

---

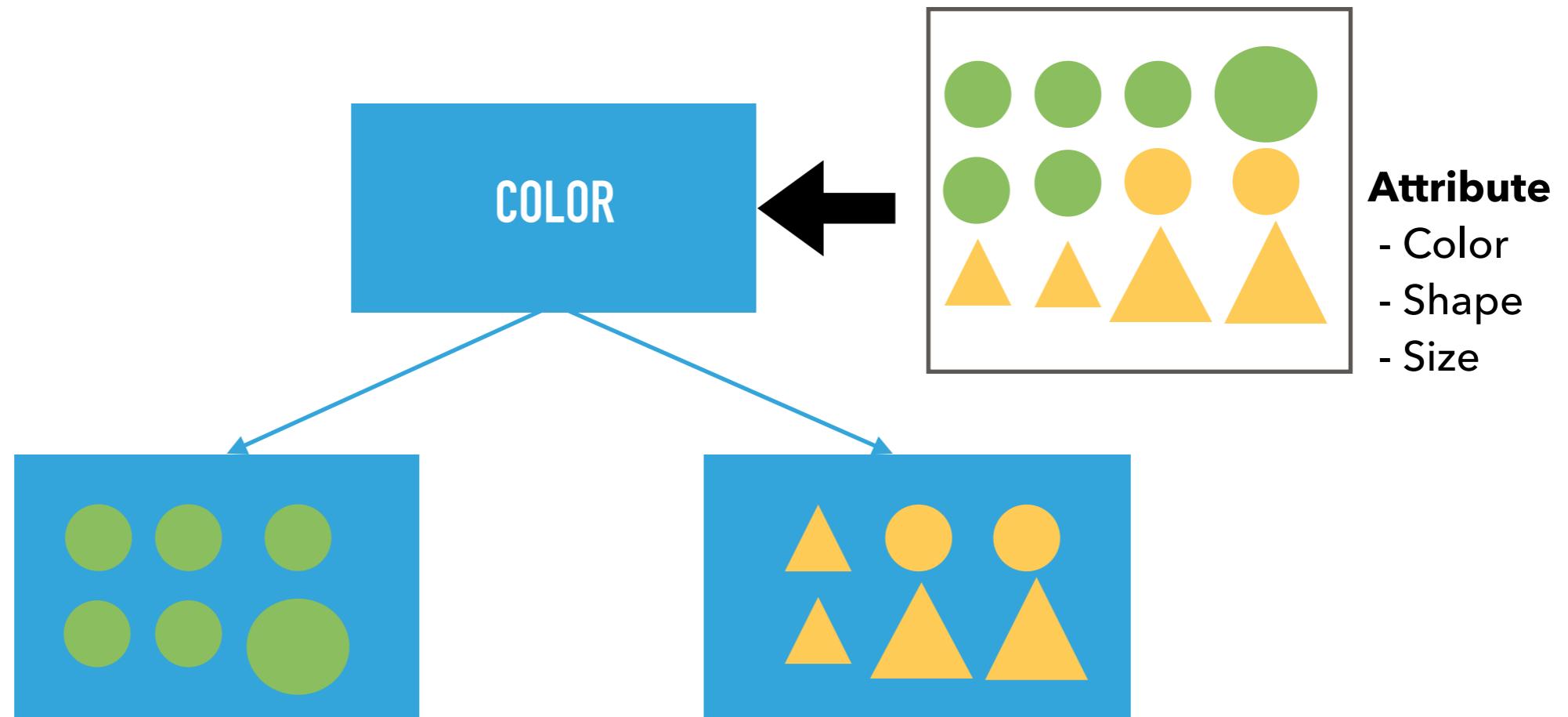
## HOW TO MAKE DECISION TREE?



## 4. DECISION TREE

---

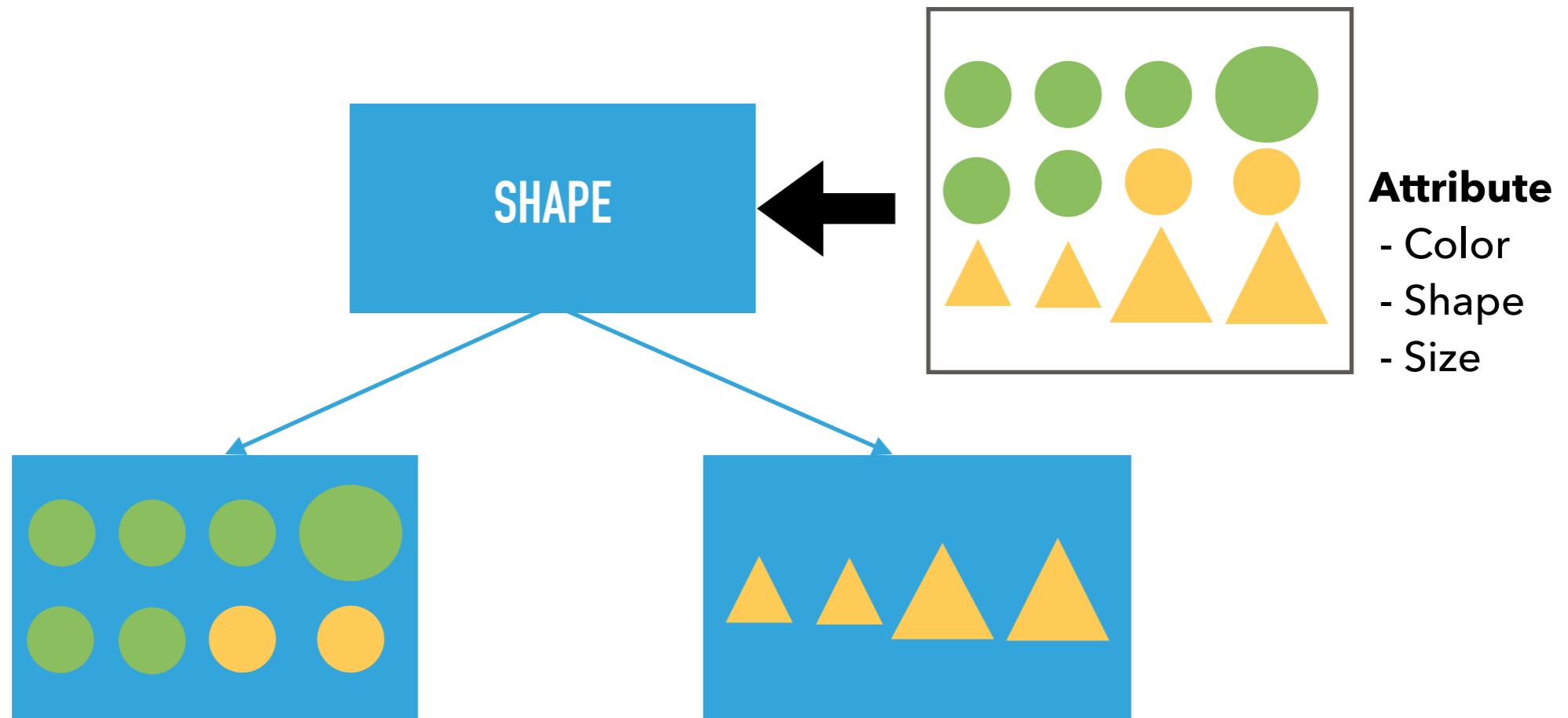
# HOW TO MAKE DECISION TREE?



## 4. DECISION TREE

---

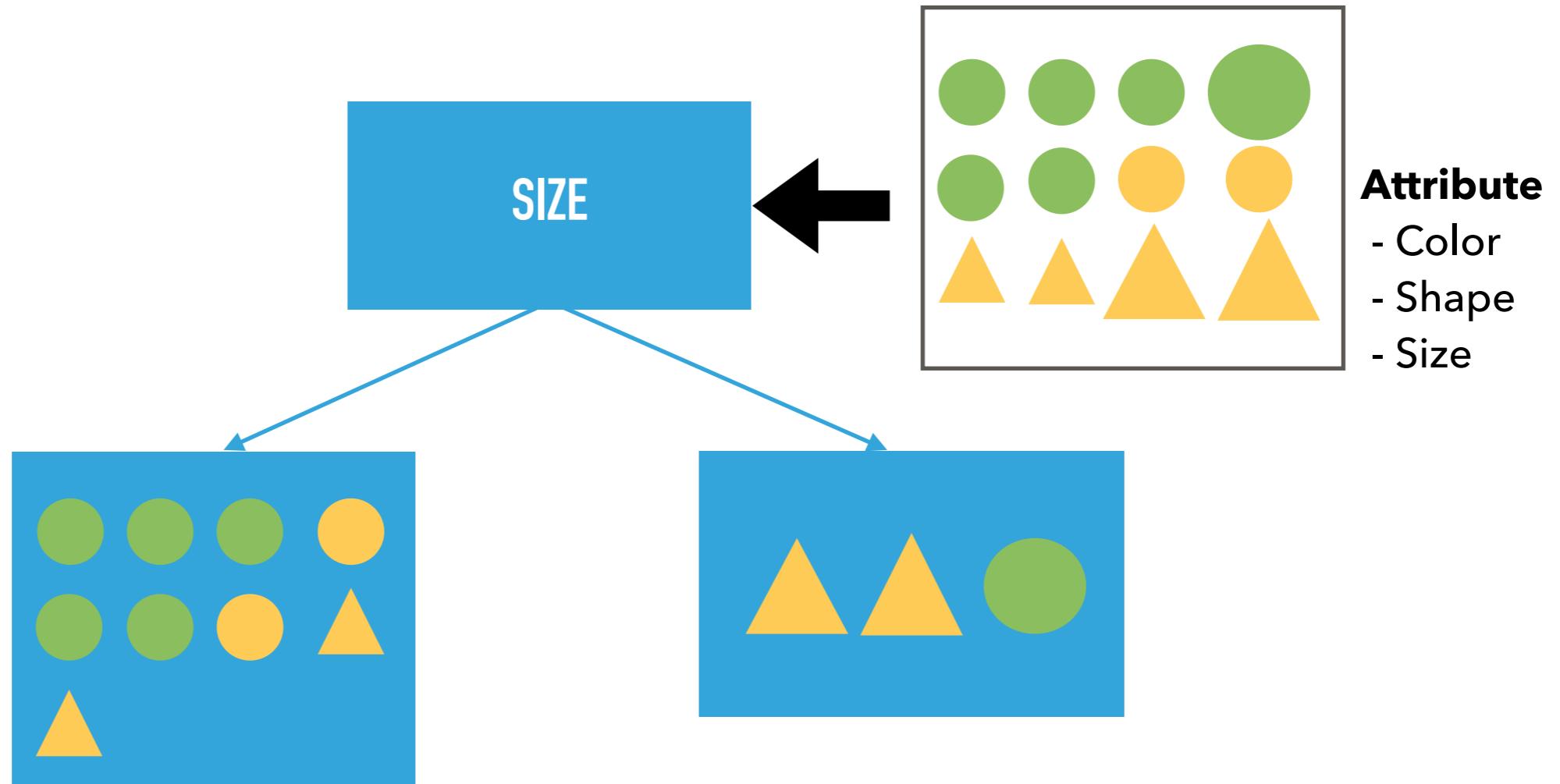
# HOW TO MAKE DECISION TREE?



## 4. DECISION TREE

---

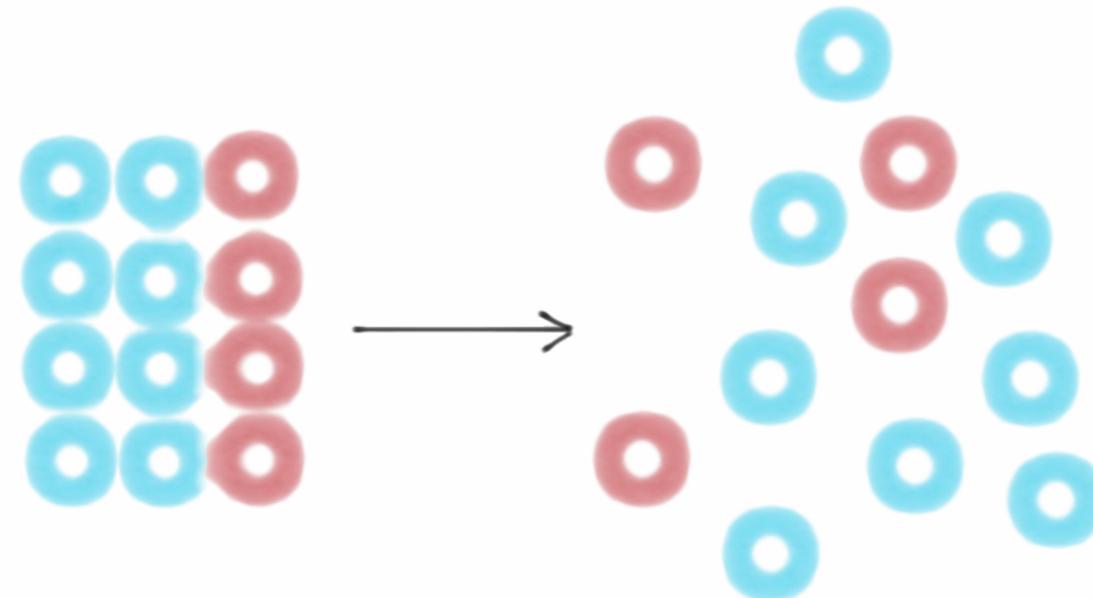
# HOW TO MAKE DECISION TREE?



## 4. 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

## 4. DECISION TREE

---

### 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?

## 4. DECISION TREE

---

### FOLLOW UP

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

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

    dt = DecisionTree()

    app.exec_()
```

## 4. DECISION TREE

# FOLLOW UP

	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

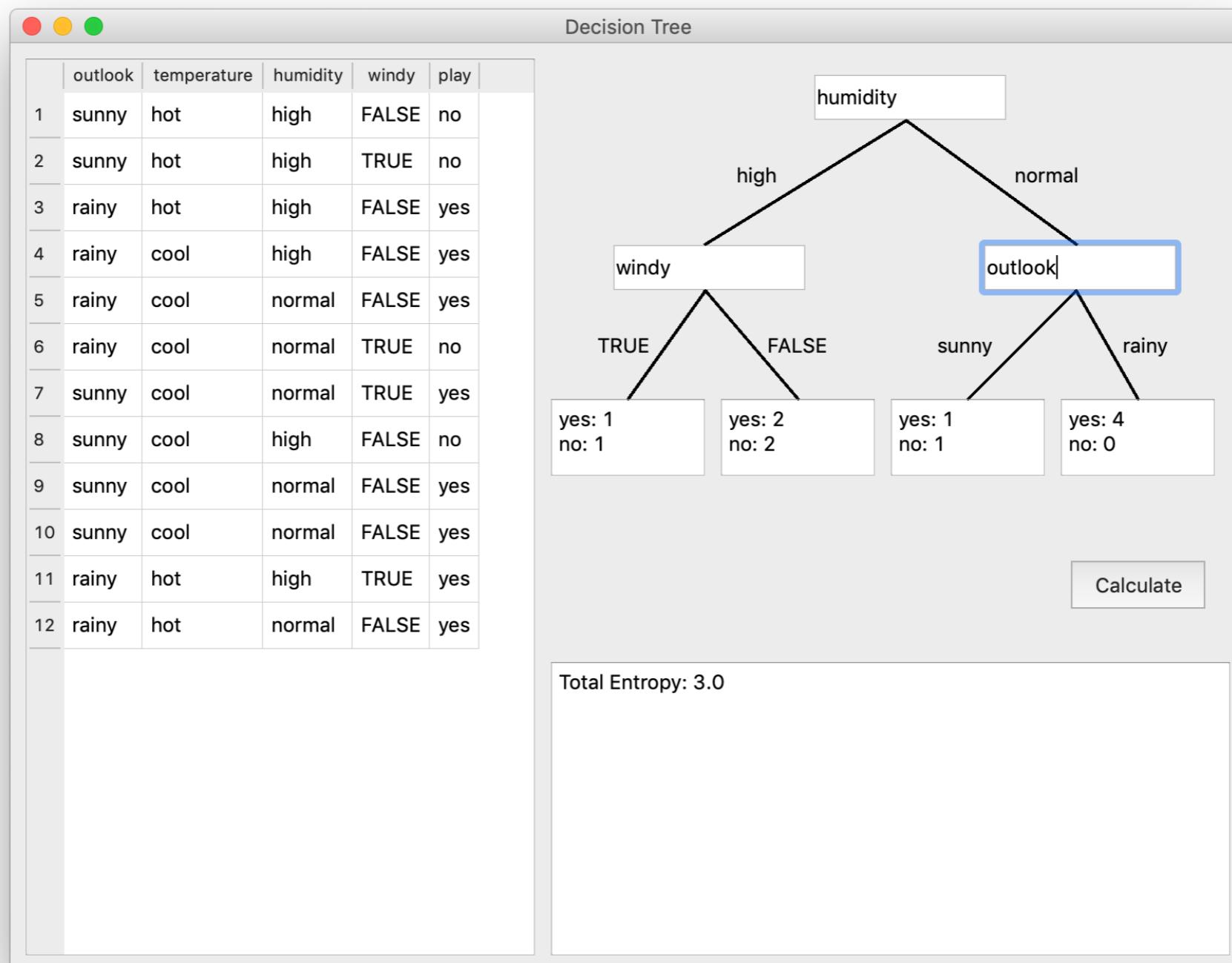
Decision Tree

```
graph TD; Root[ ] --- NodeL1L[ ]; Root --- NodeL1R[ ]; NodeL1L --- Leaf1[ ]; NodeL1L --- Leaf2[ ]; NodeL1R --- Leaf3[ ]; NodeL1R --- Leaf4[ ]
```

Input the Attributes and Make Entropy smaller!

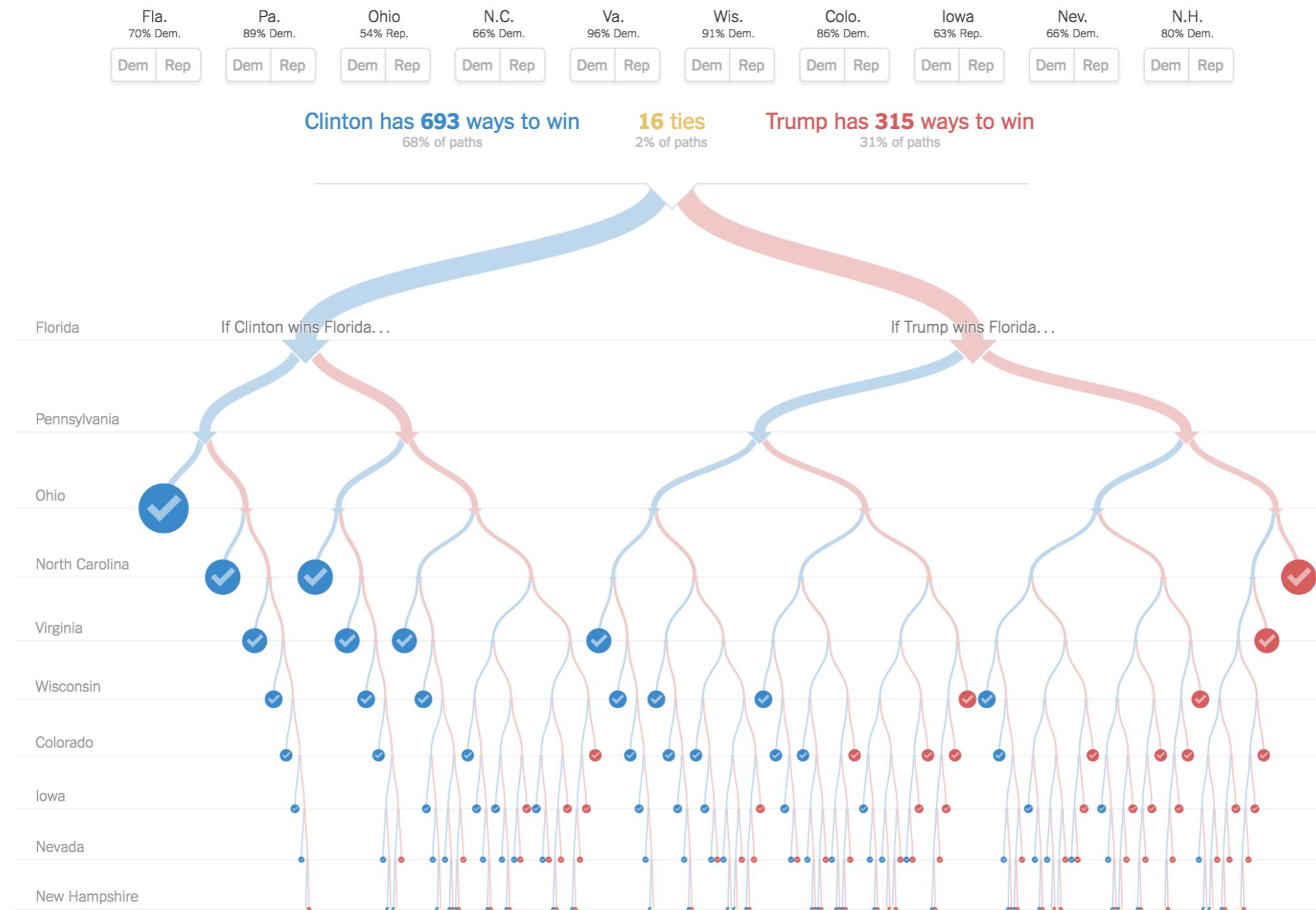
## 4. DECISION TREE

# FOLLOW UP



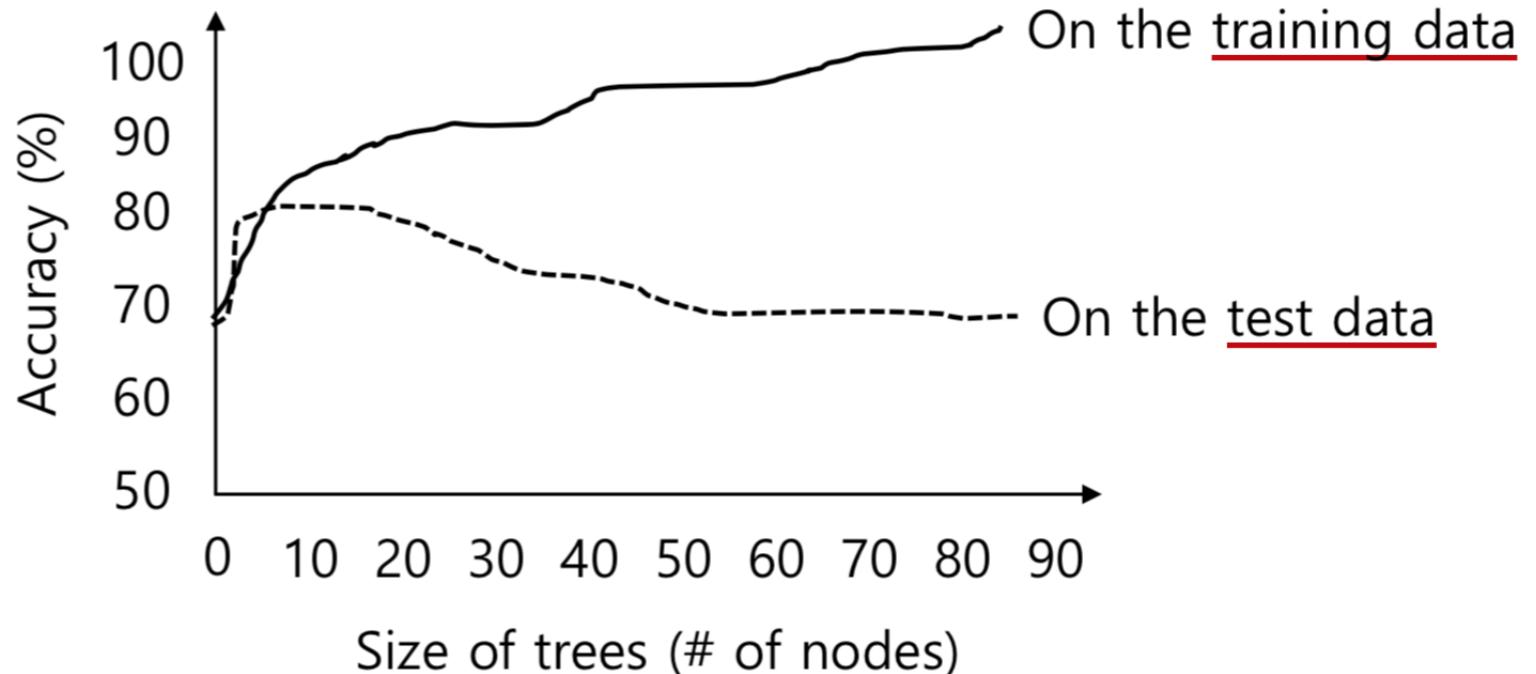
## 4. DECISION TREE

# DEEP DECISION TREE?

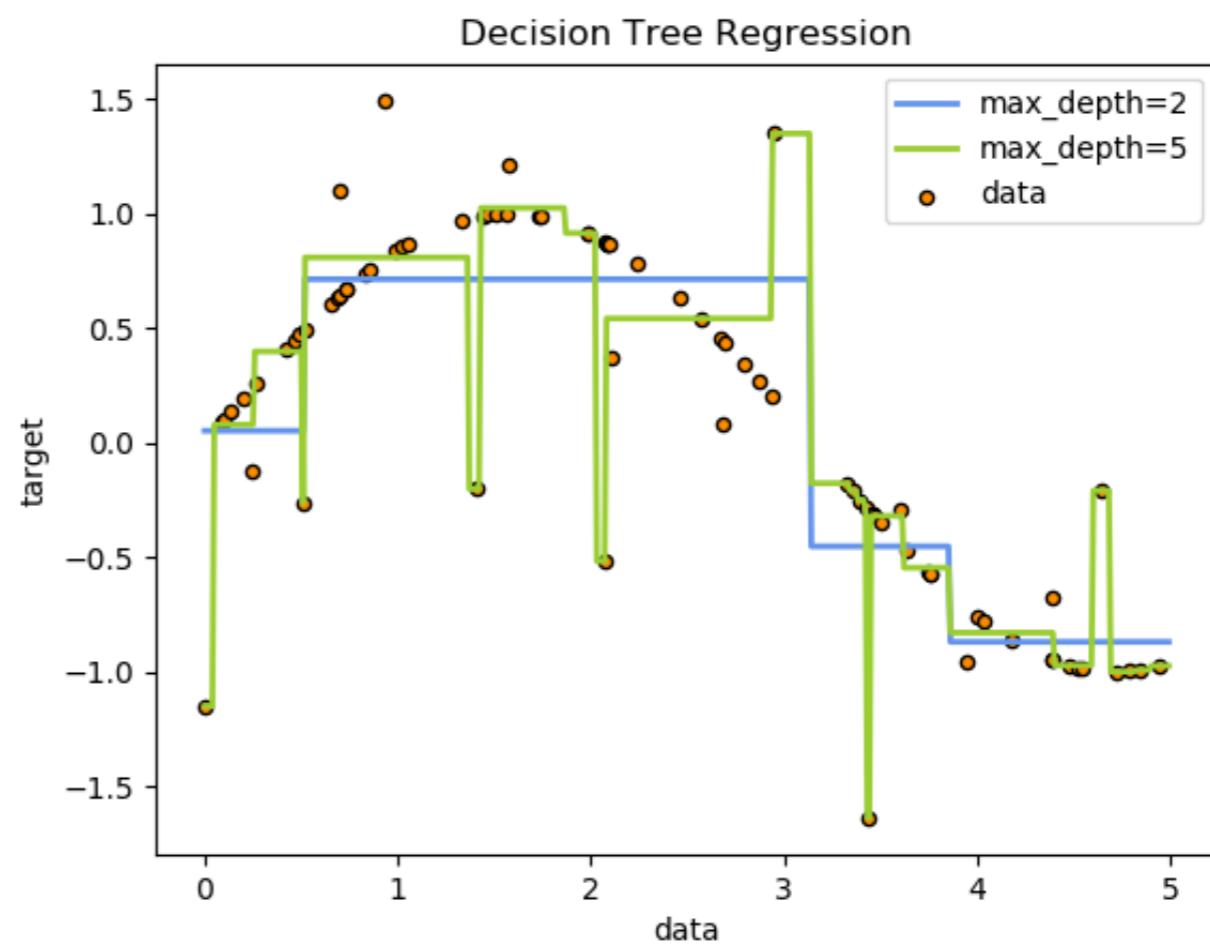


## 4. DECISION TREE

### DEEP DECISION TREE?

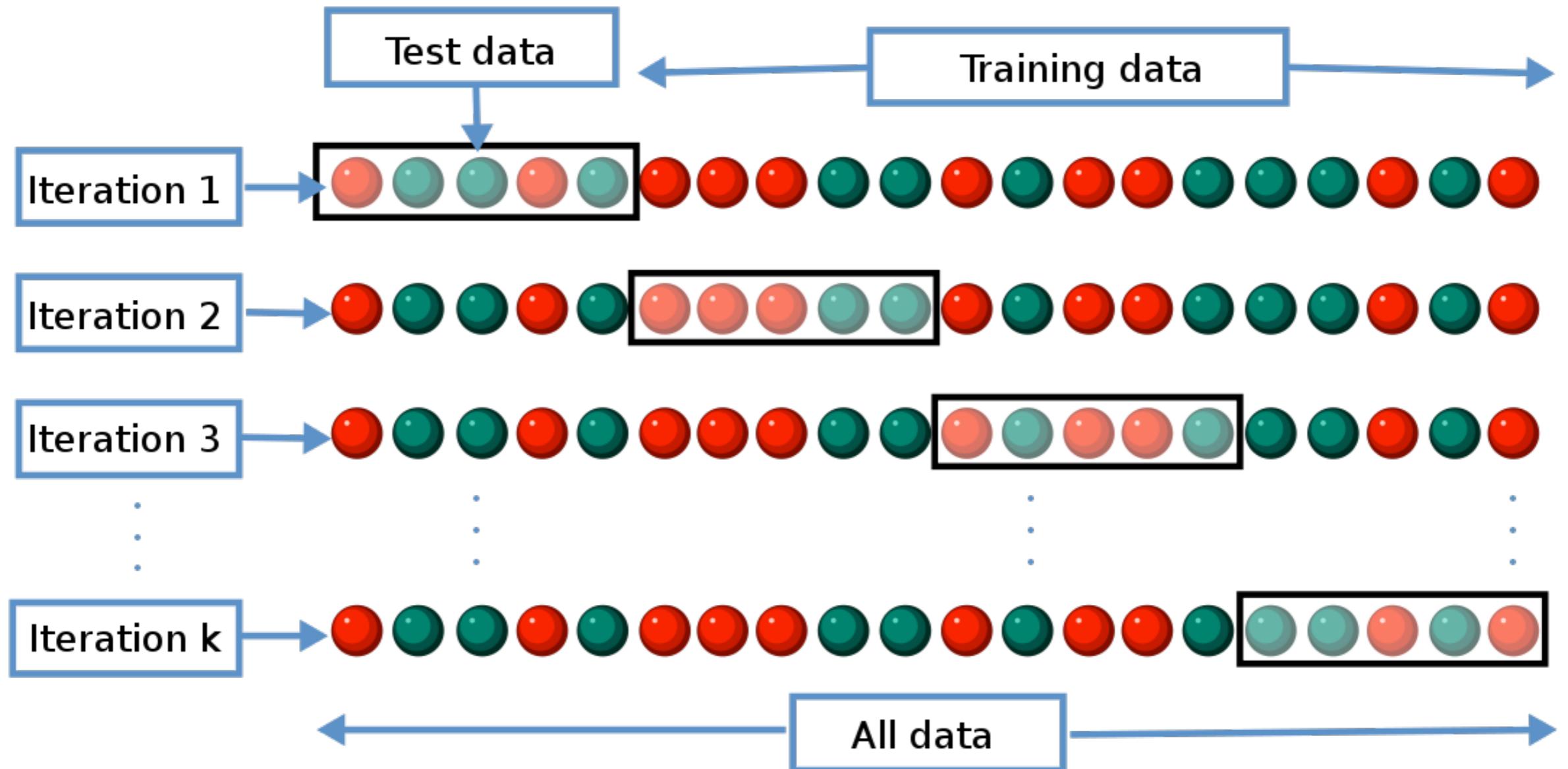


Overfitting Problem



## 4. DECISION TREE

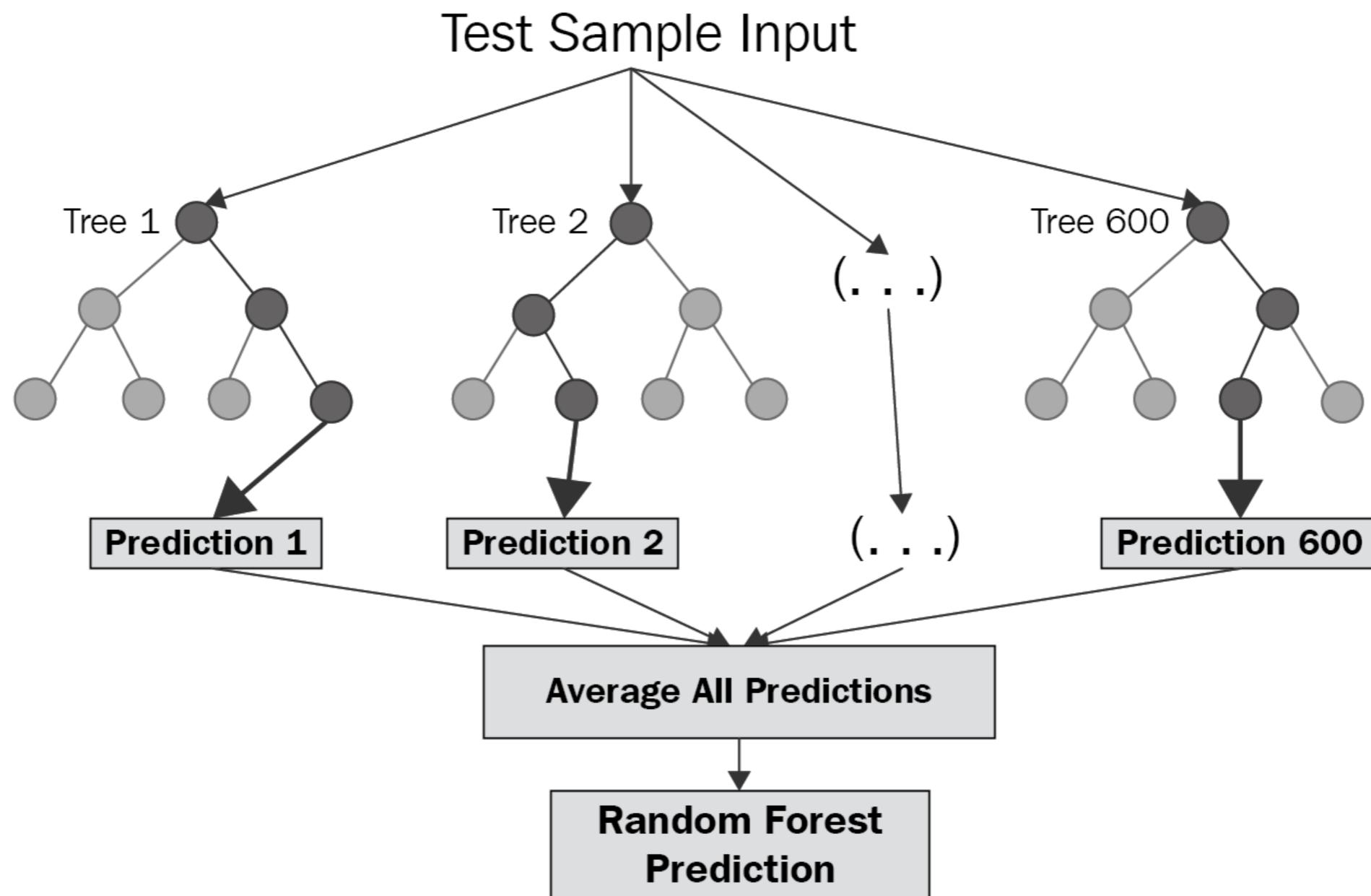
# CROSS VALIDATING



## 4. DECISION TREE

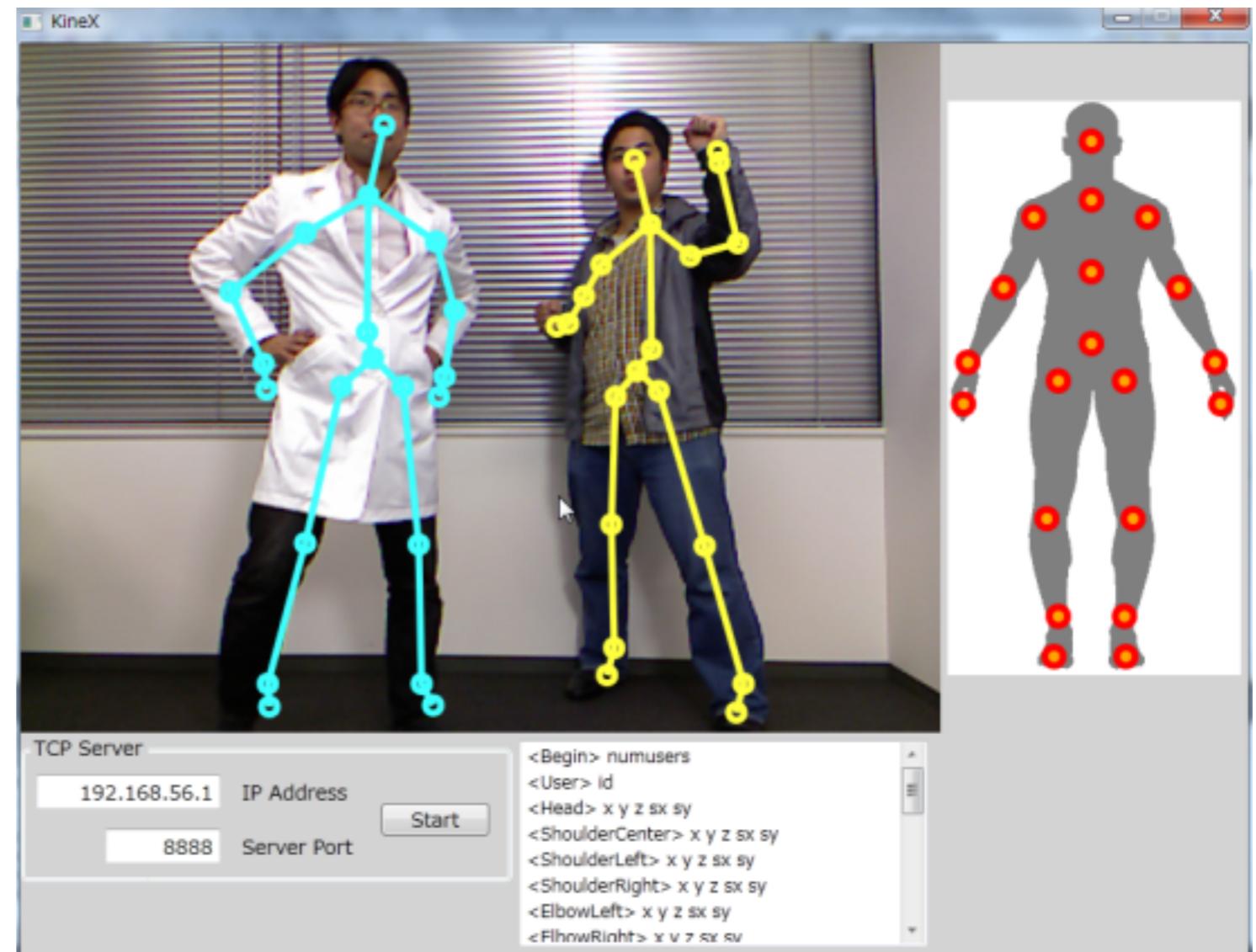
---

# RANDOM FOREST



## 4. DECISION TREE

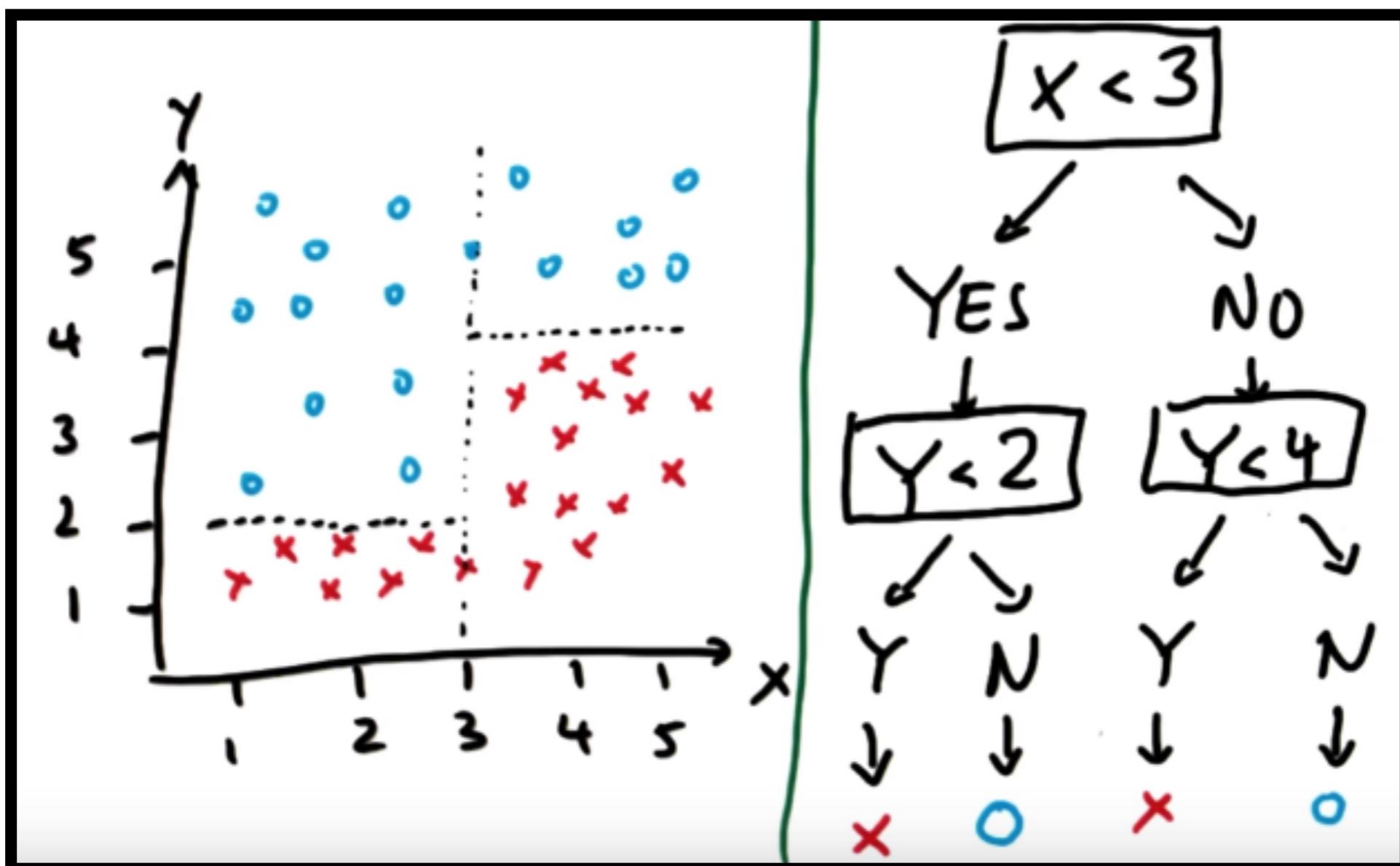
# RANDOM FOREST - XBOX360 KINECT



#### 4. DECISION TREE

---

ONLY DISCRETE DATA ARE USED?



## 4. DECISION TREE

---

WEKA

The screenshot shows a web browser window displaying the Weka Wiki page at [https://waikato.github.io/weka-wiki/downloading\\_weka/](https://waikato.github.io/weka-wiki/downloading_weka/). The left sidebar contains a navigation menu with sections like Home, Downloading and installing Weka, Requirements, Documentation, Getting help, Citing Weka, Literature, Development, History, Resources, FAQ, Not so FAQ, and Troubleshooting. The main content area is titled "Stable version" and discusses Weka 3.8. It includes sections for Windows, Mac OS, Linux, and Other platforms, each with a list of download links. The Windows section is highlighted with a red box around its content.

**Stable version**

Weka 3.8 is the latest stable version of Weka. This branch of Weka only receives bug fixes and upgrades that do not break compatibility with earlier 3.8 releases, although major new features may become available in packages. There are different options for downloading and installing it on your system:

**Windows**

- Click [here](#) to download a self-extracting executable for 64-bit Windows that includes Azul's 64-bit OpenJDK Java VM 11 (weka-3-8-4-azul-zulu-windows.exe; 118 MB)

This executable will install Weka in your Program Menu. Launching via the Program Menu or shortcuts will automatically use the included JVM to run Weka.

**Mac OS**

- Click [here](#) to download a disk image for Mac OS that contains a Mac application including Azul's 64-bit OpenJDK Java VM 11 (weka-3-8-4-azul-zulu-osx.dmg; 144 MB)

**Linux**

- Click [here](#) to download a zip archive for Linux that includes Azul's 64-bit OpenJDK Java VM 11 (weka-3-8-4-azul-zulu-linux.zip; 129 MB)

First unzip the the zip file. This will create a new directory called weka-3-8-4. To run Weka, change into that directory and type

```
./weka.sh
```

**Other platforms**

- Click [here](#) to download a zip archive containing Weka (weka-3-8-4.zip; 53 MB)

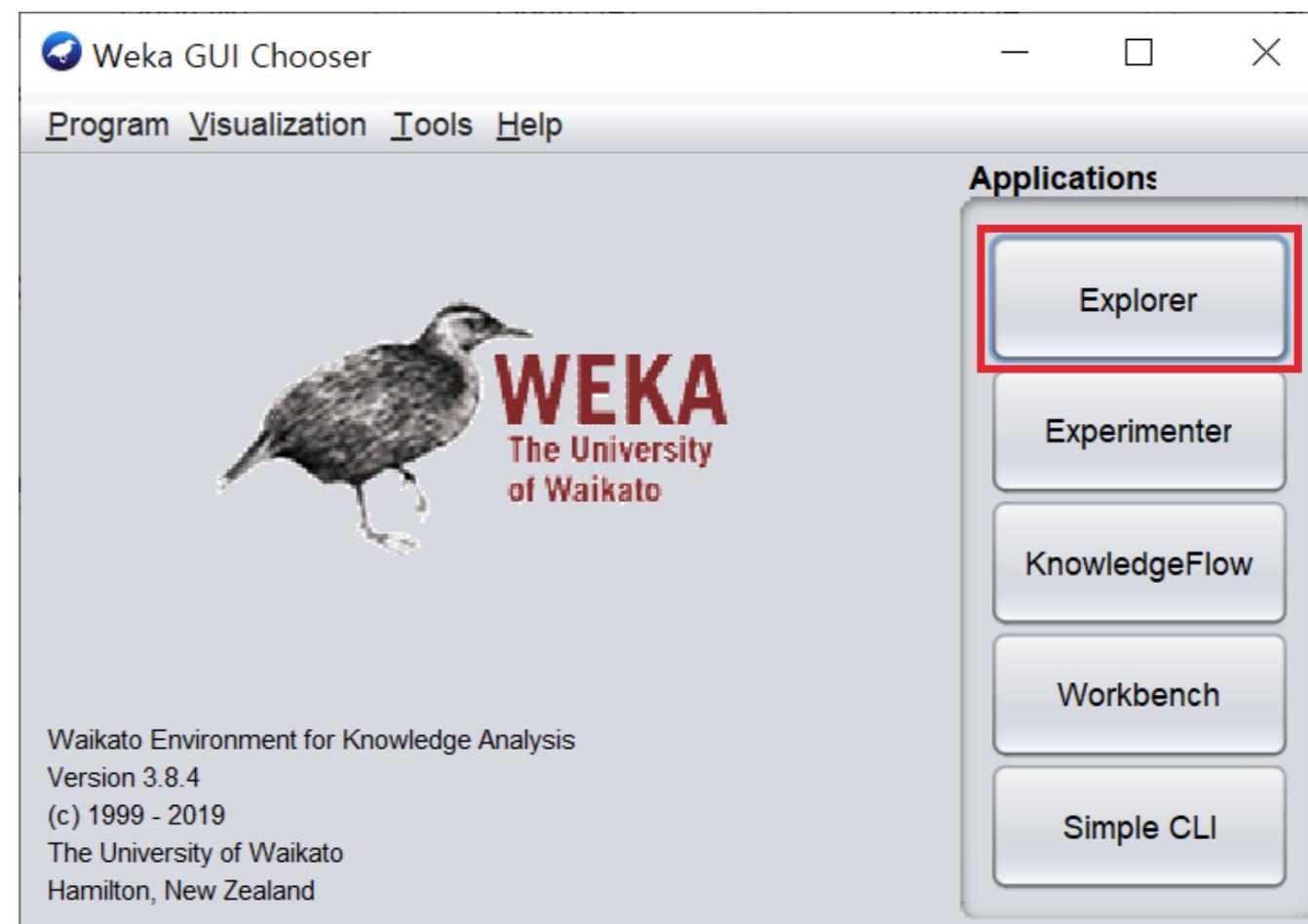
First unzip the zip file. This will create a new directory called weka-3-8-3. To run Weka, change into that directory and type

[https://waikato.github.io/weka-wiki/downloading\\_weka/](https://waikato.github.io/weka-wiki/downloading_weka/)

## 4. DECISION TREE

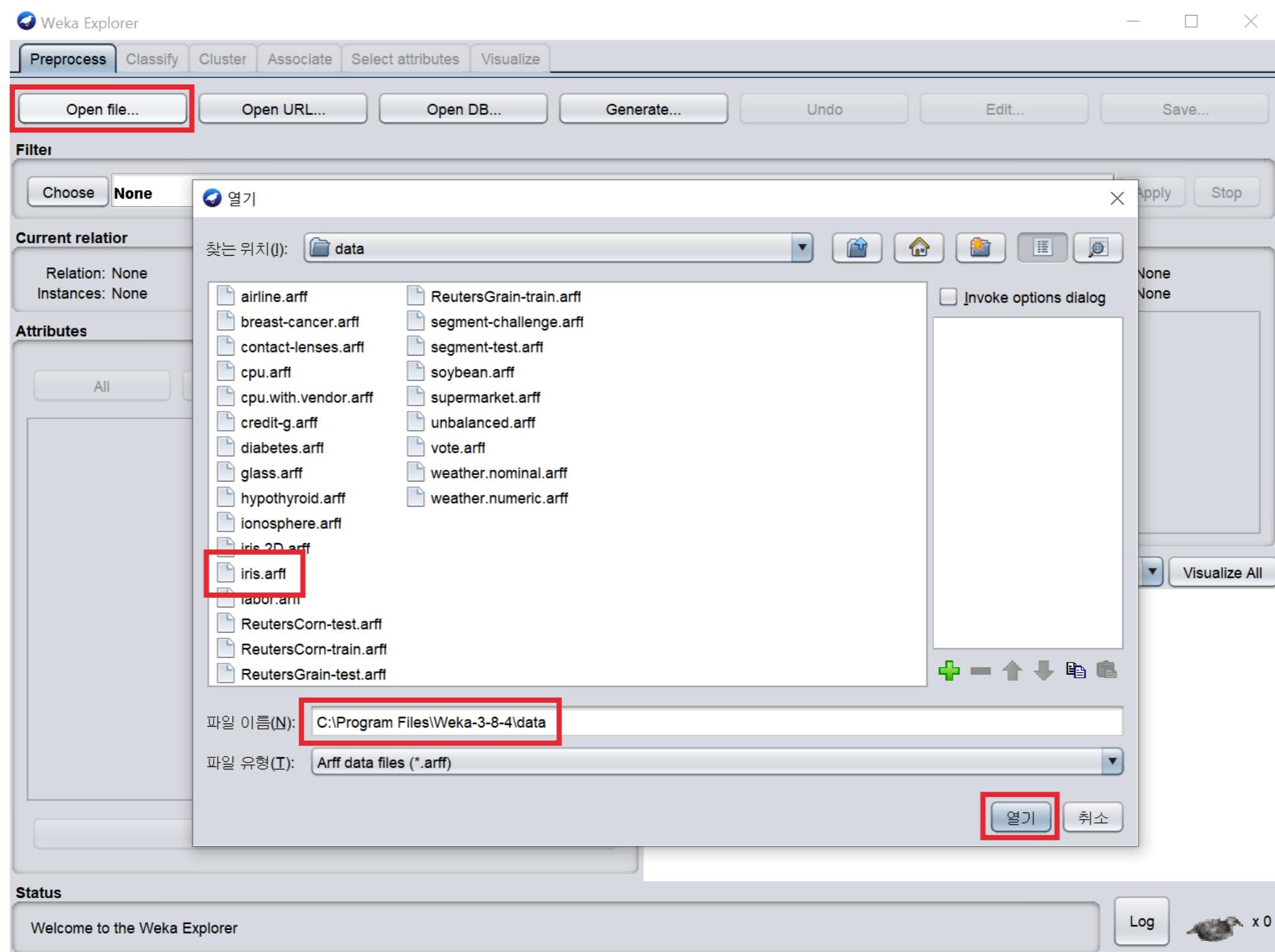
---

WEKA



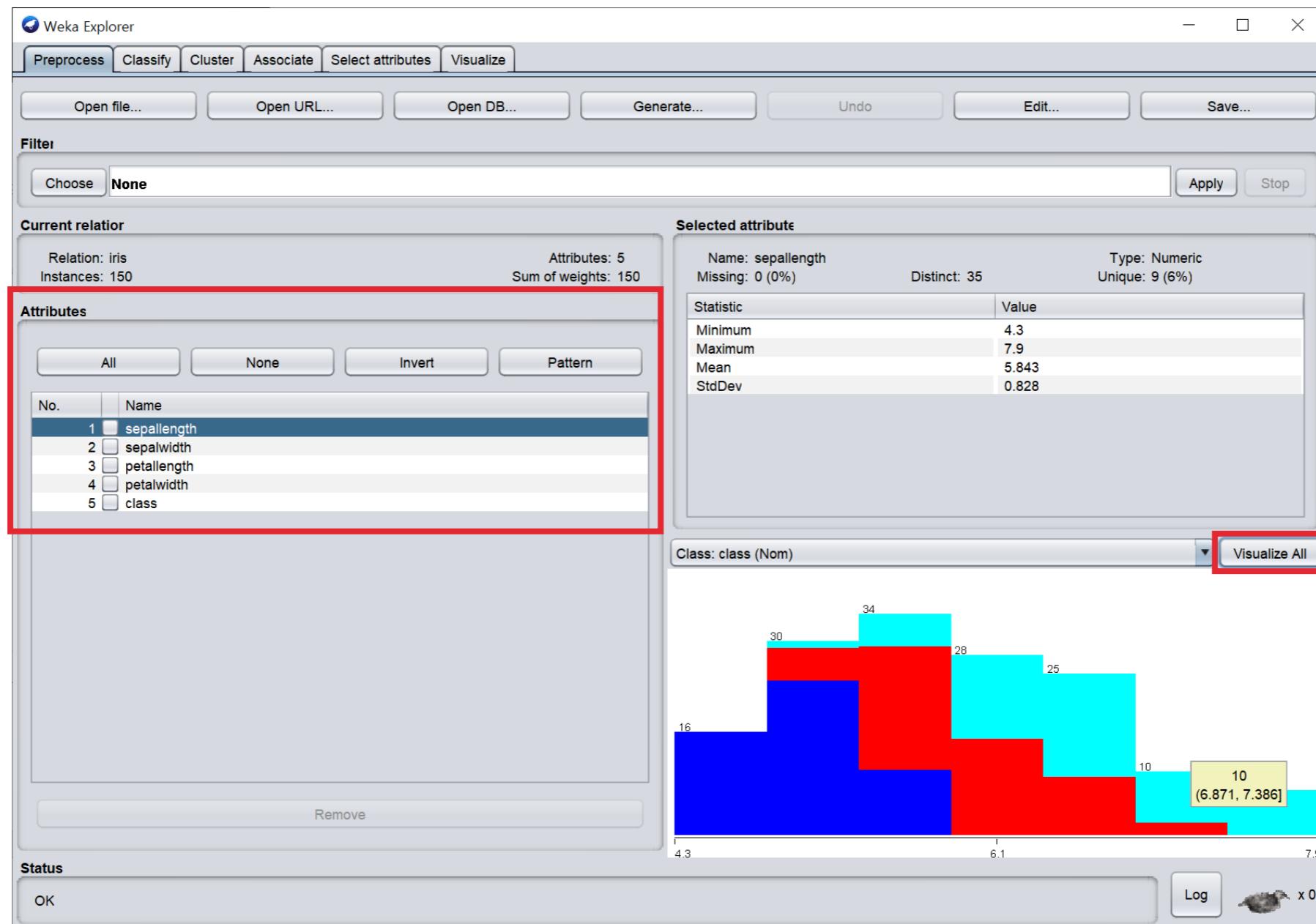
## 4. DECISION TREE

# WEKA



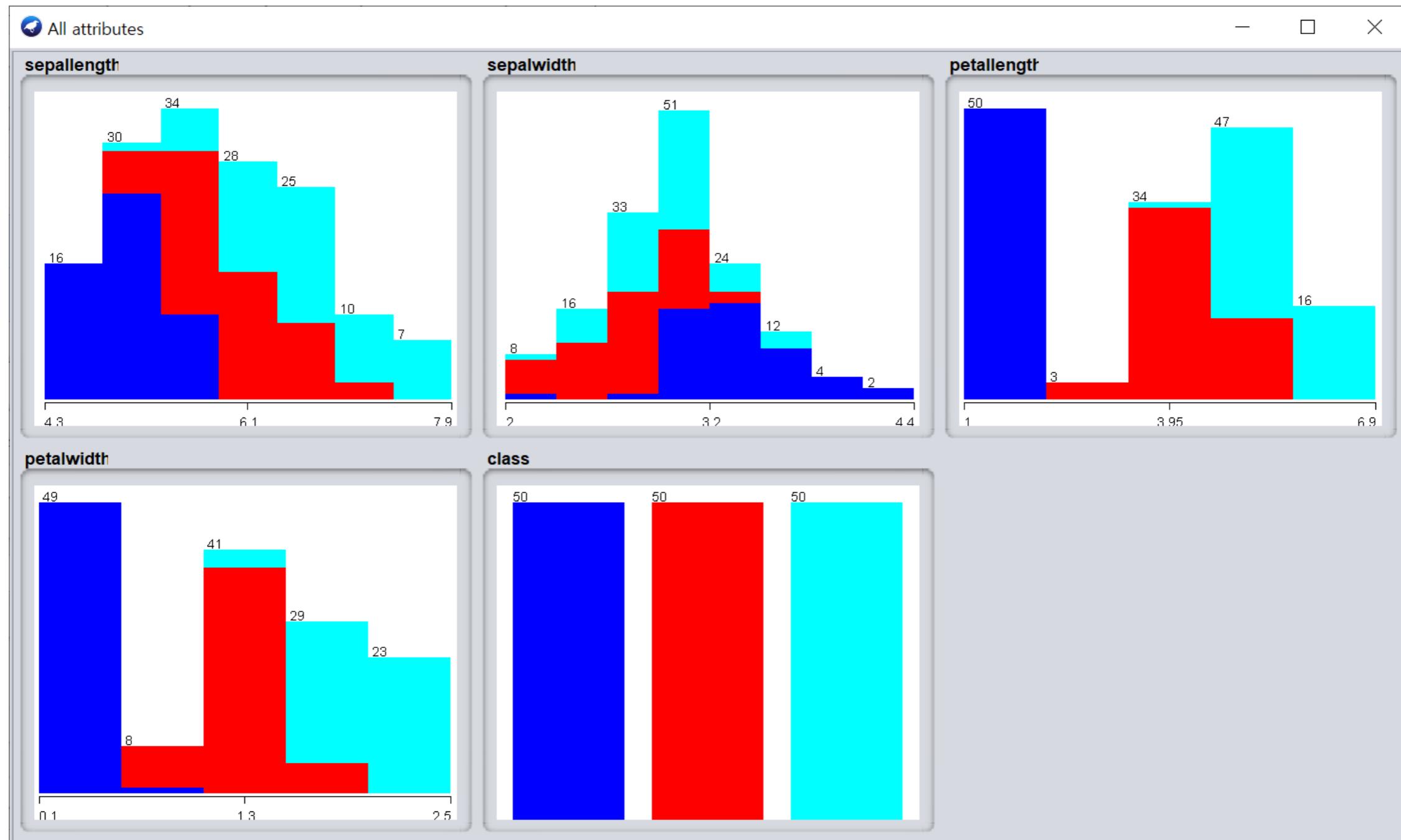
## 4. DECISION TREE

WEKA



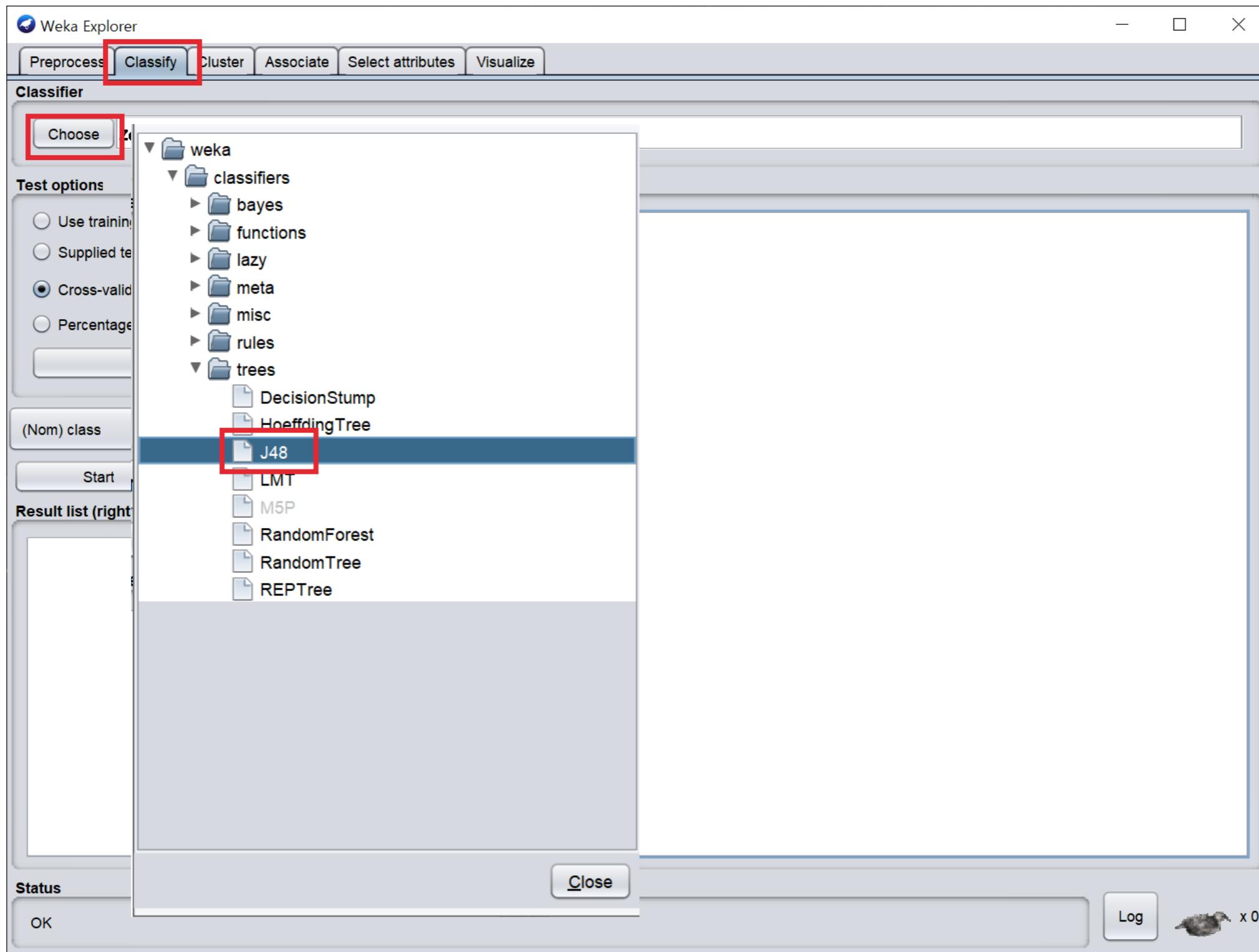
## 4. DECISION TREE

WEKA



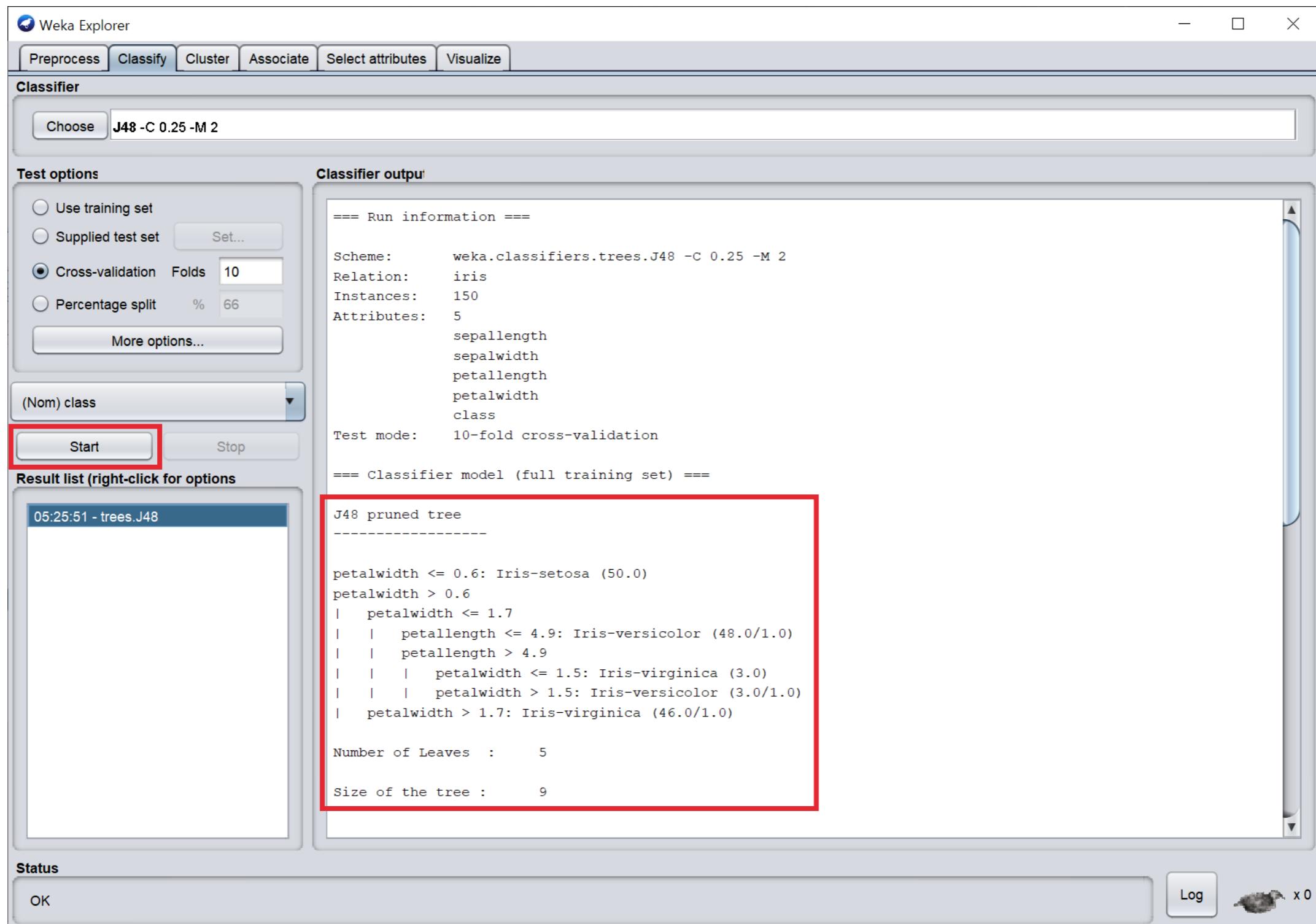
## 4. DECISION TREE

WEKA



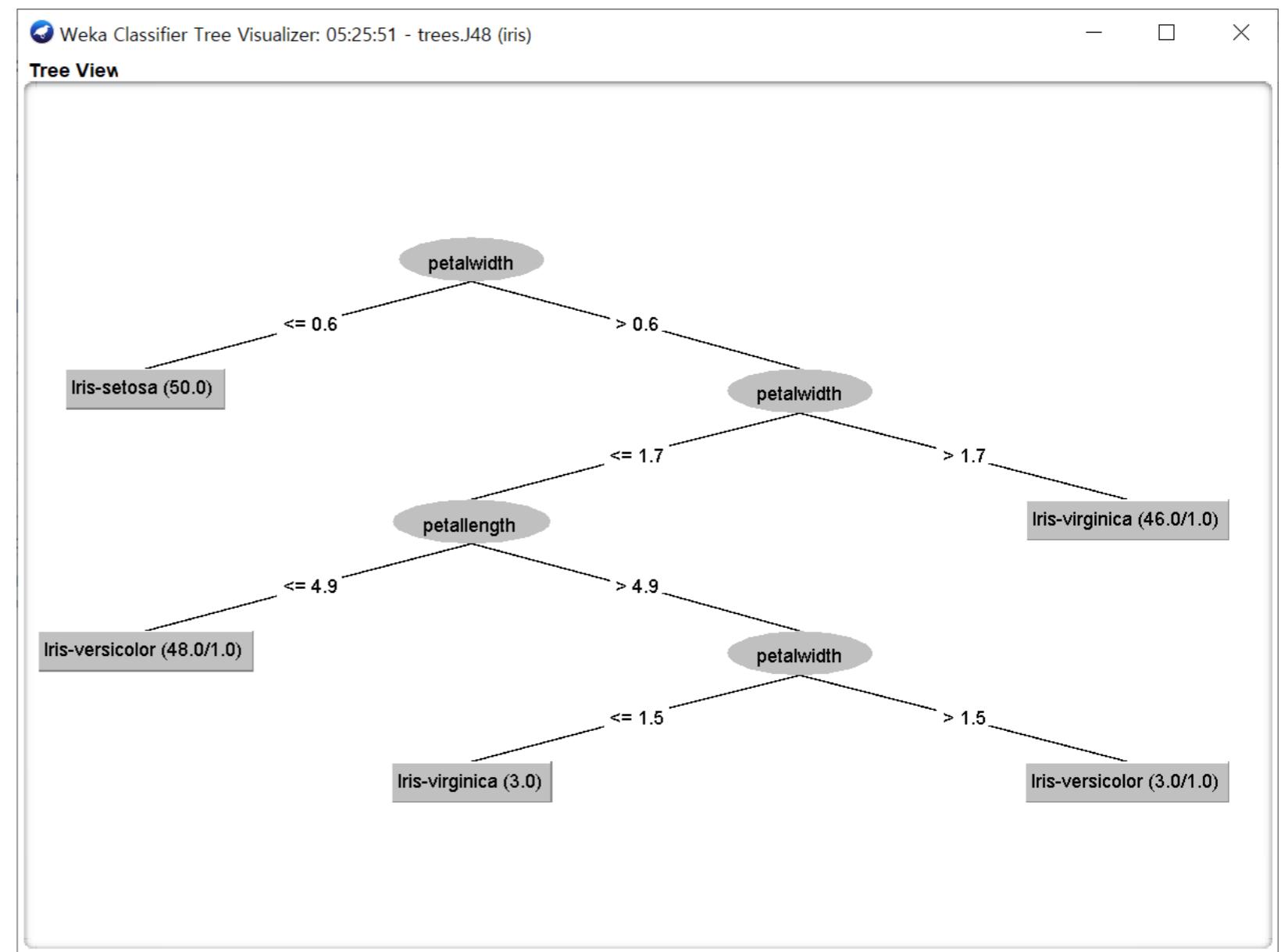
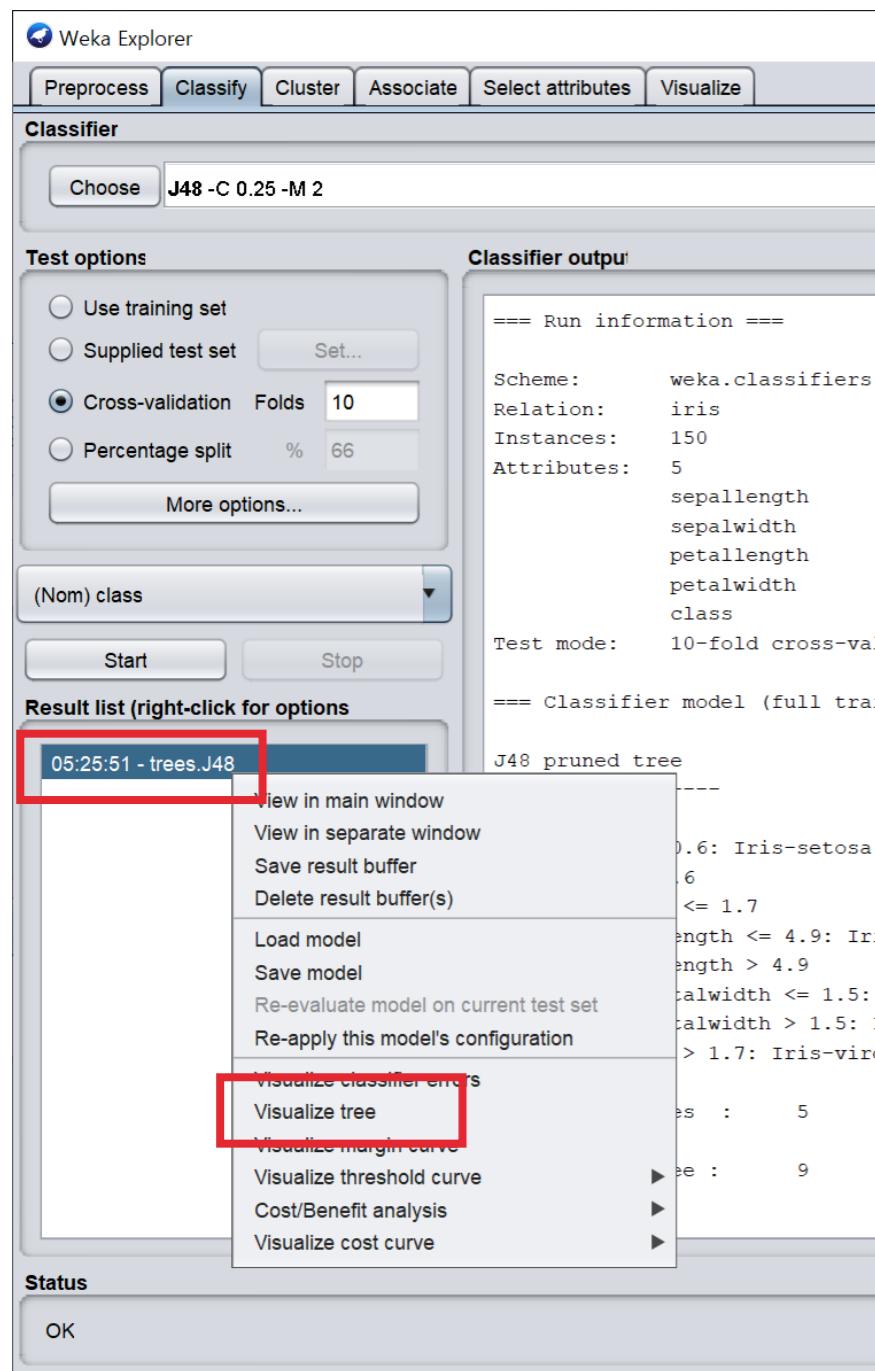
## 4. DECISION TREE

WEKA



## 4. DECISION TREE

### WEKA



## 5. K-MEAN CLUSTERING

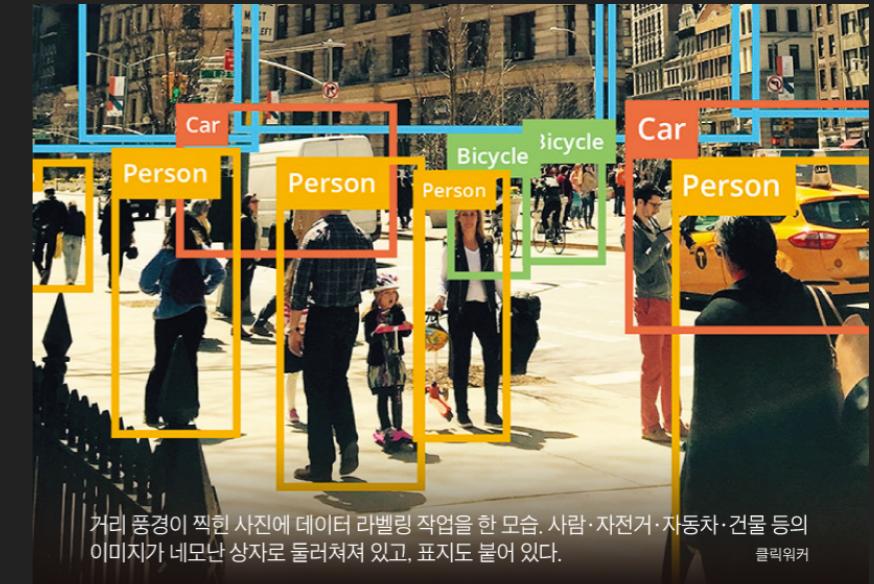
---

# DEFECT OF SUPERVISED LEARNING



## 5. K-MEAN CLUSTERING

# DEFECT OF SUPERVISED LEARNING

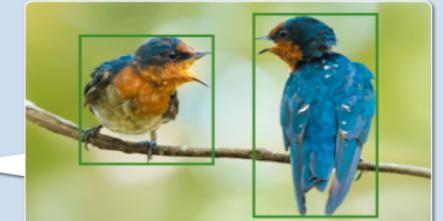


## AI의 머신러닝, 알고보니 中·인도의 값싼 노동력 덕분

### 대표적 데이터 라벨링 작업

#### 박스로 묶기

사진 속 물체를 박스로 표시해줌



#### 이미지 분류

어떤 장면의 사진인지 분류

사람들이 운동하는 사진을 보고  
이 운동이 야구인지 축구인지  
등을 분류



#### 시맨틱 분할

사진 내에서 픽셀(점) 단위로  
객체(대상)를 분류

이미지에서 여성과 자동차를  
세밀하게 분리

자료=아마존웹서비스(AWS)



### 데이터 라벨링 시장 규모 전망

2018년

5억달러

2023년

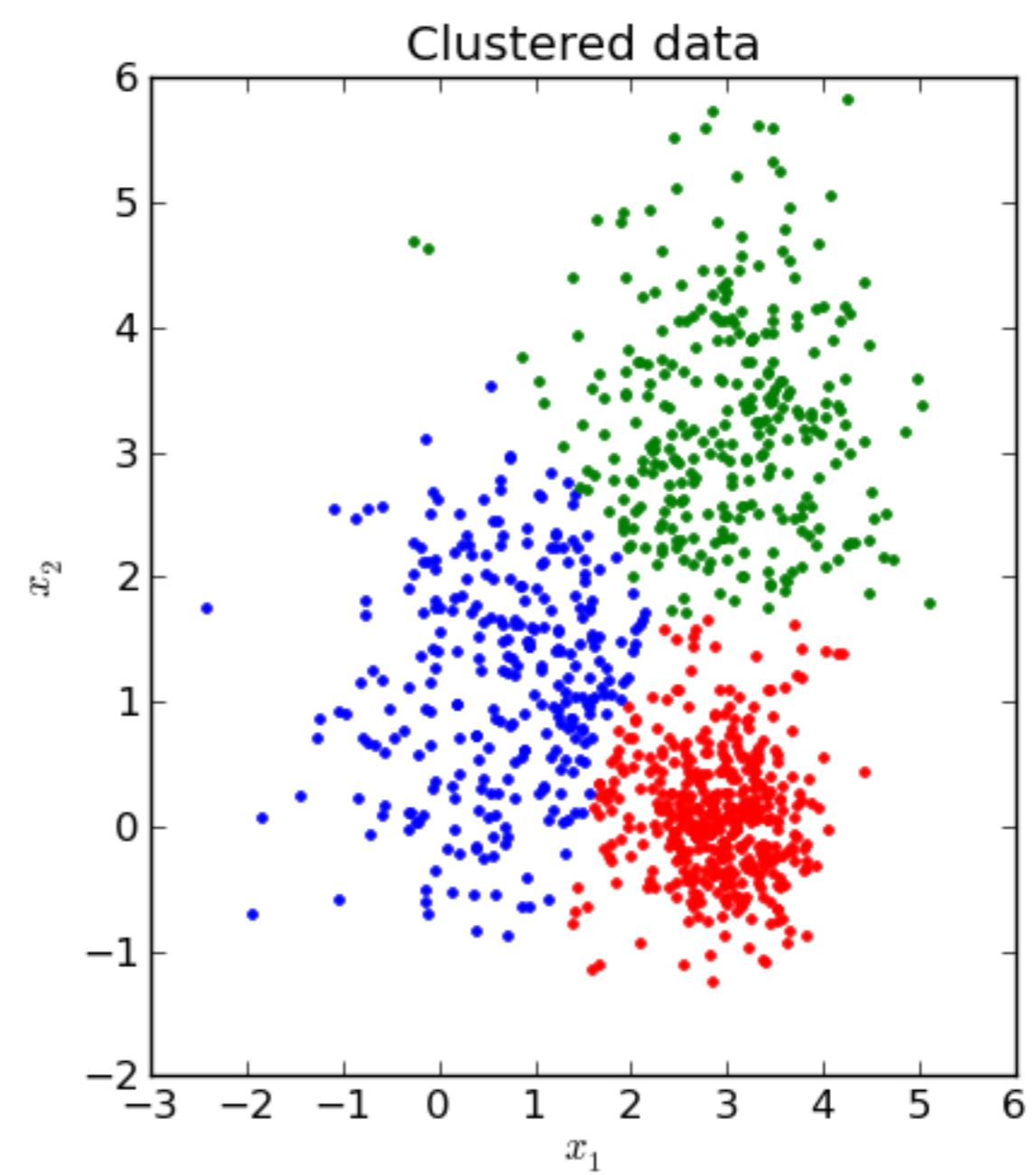
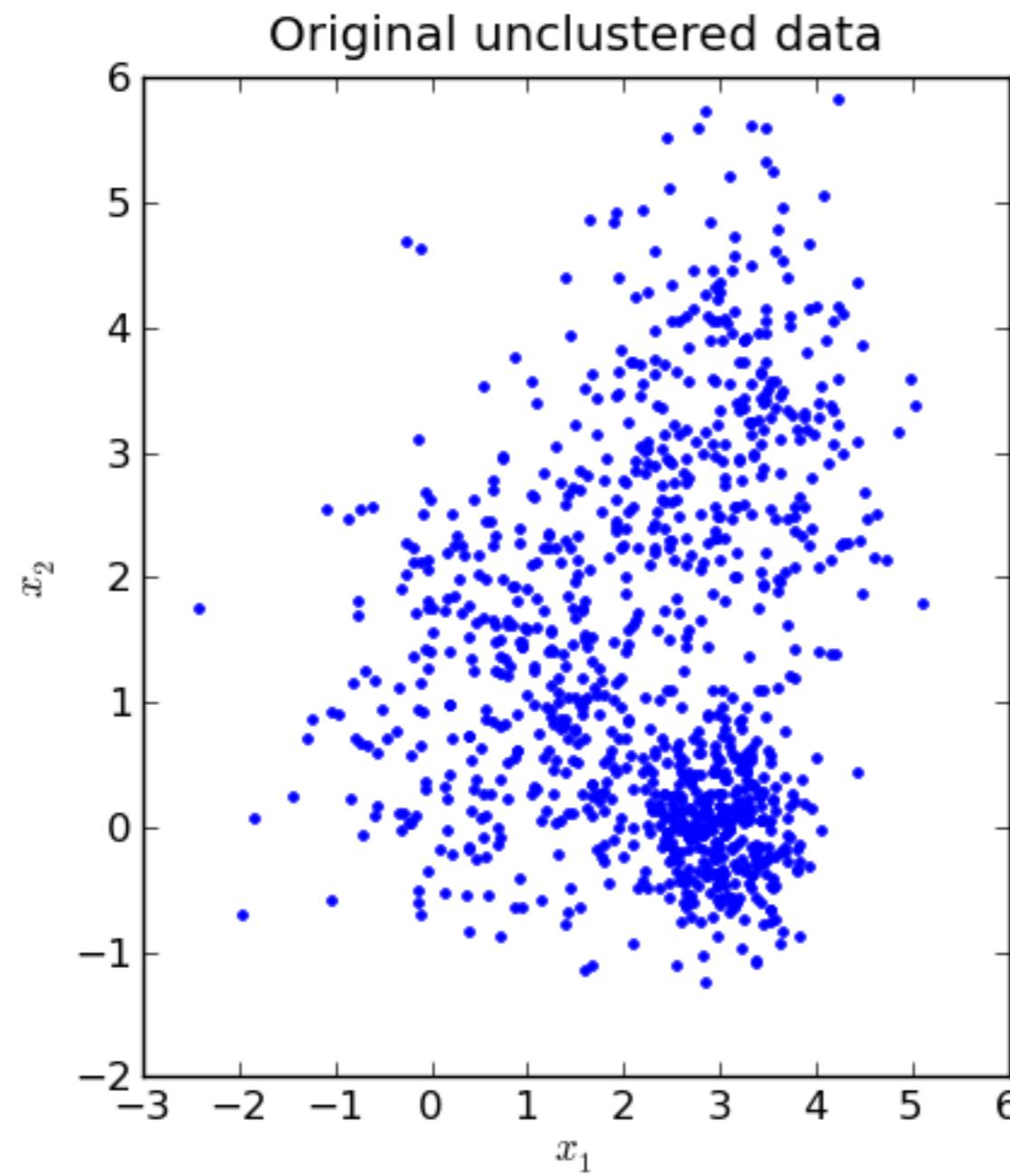
12억달러

자료=커드나리티카

## 5. K-MEAN CLUSTERING

---

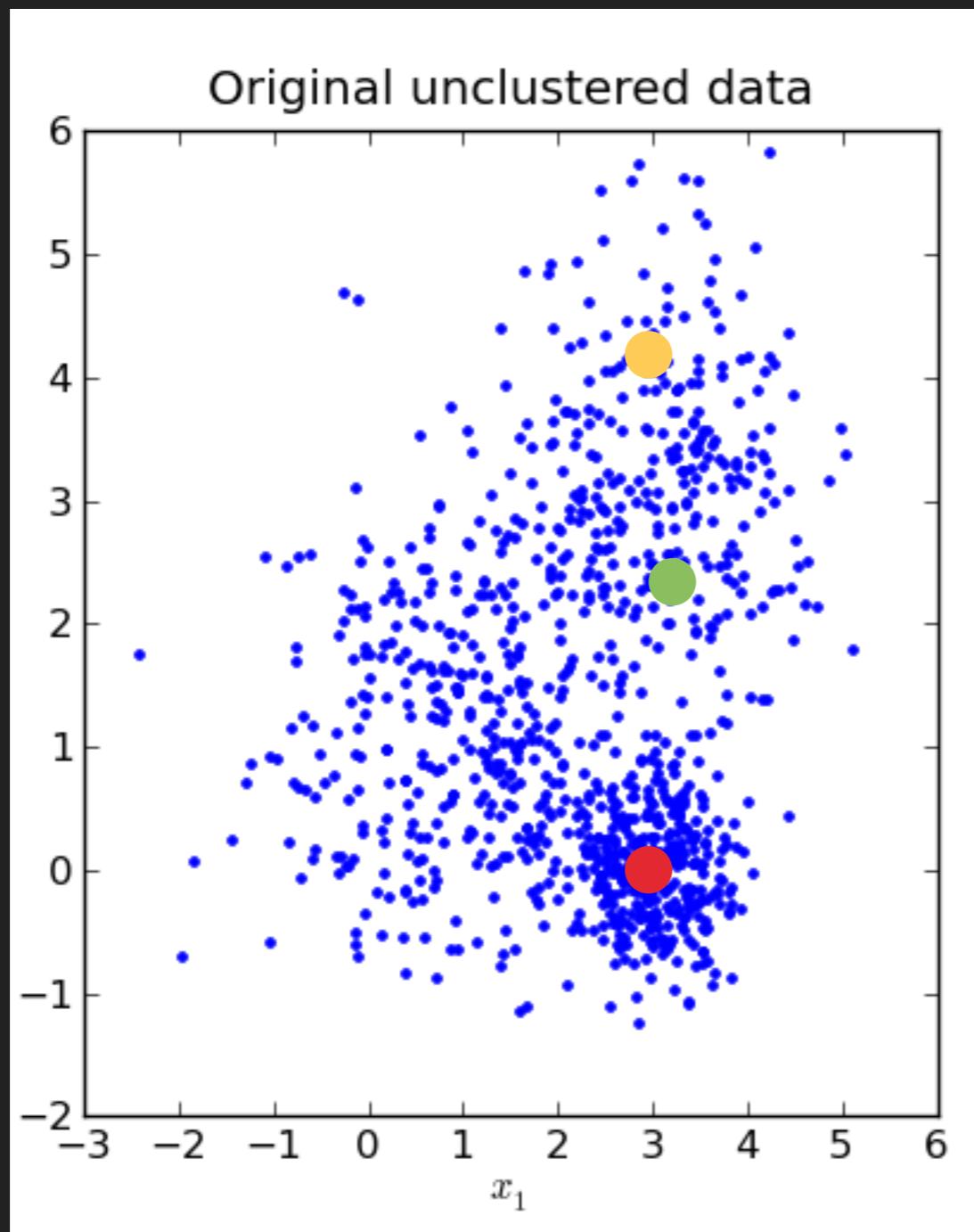
# WHAT ABOUT K-MEAN CLUSTERING?



## 5. K-MEAN CLUSTERING

---

# K-MEAN CLUSTERING?

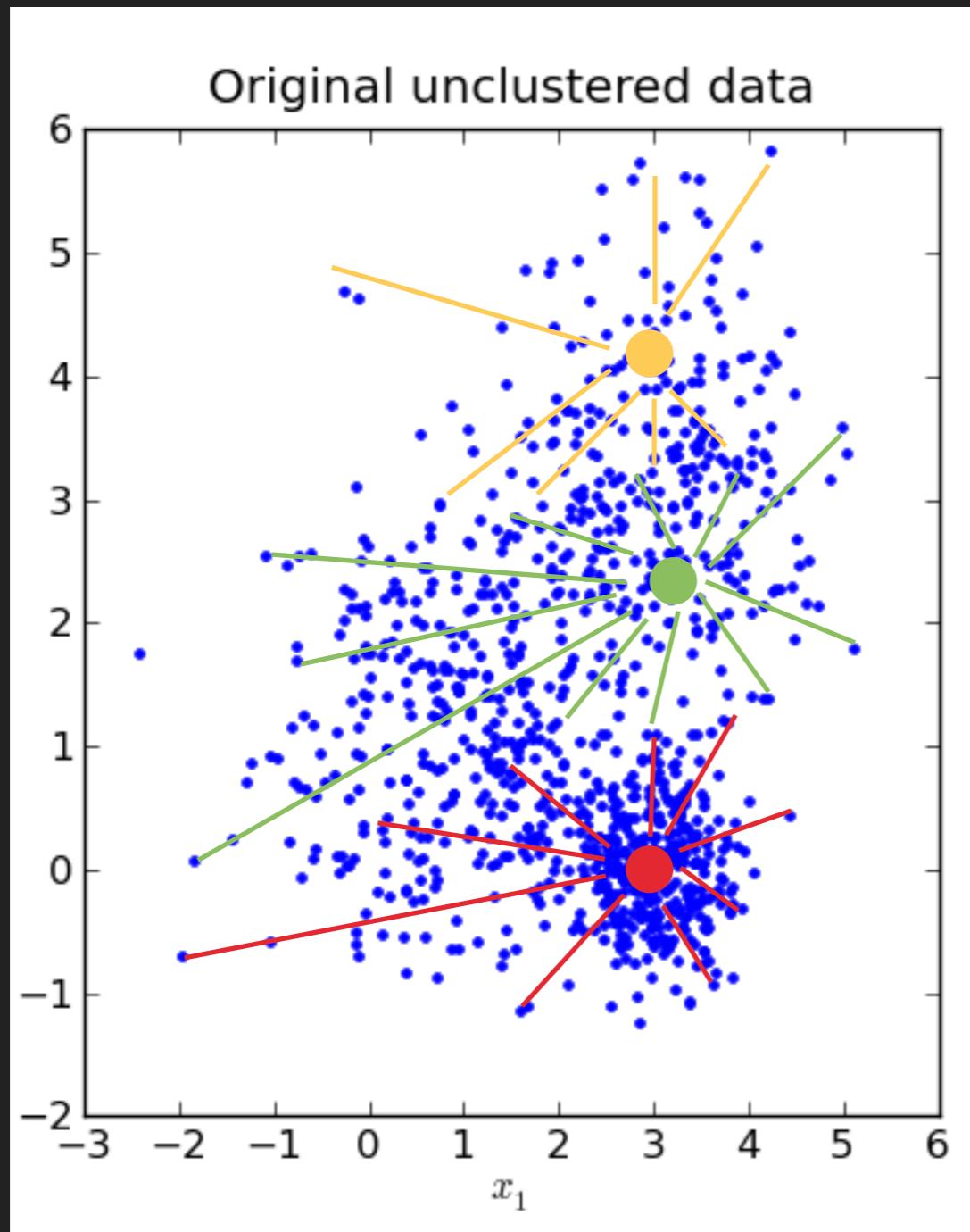


무작위 K개의 점을 지정

## 5. K-MEAN CLUSTERING

---

# K-MEAN CLUSTERING?

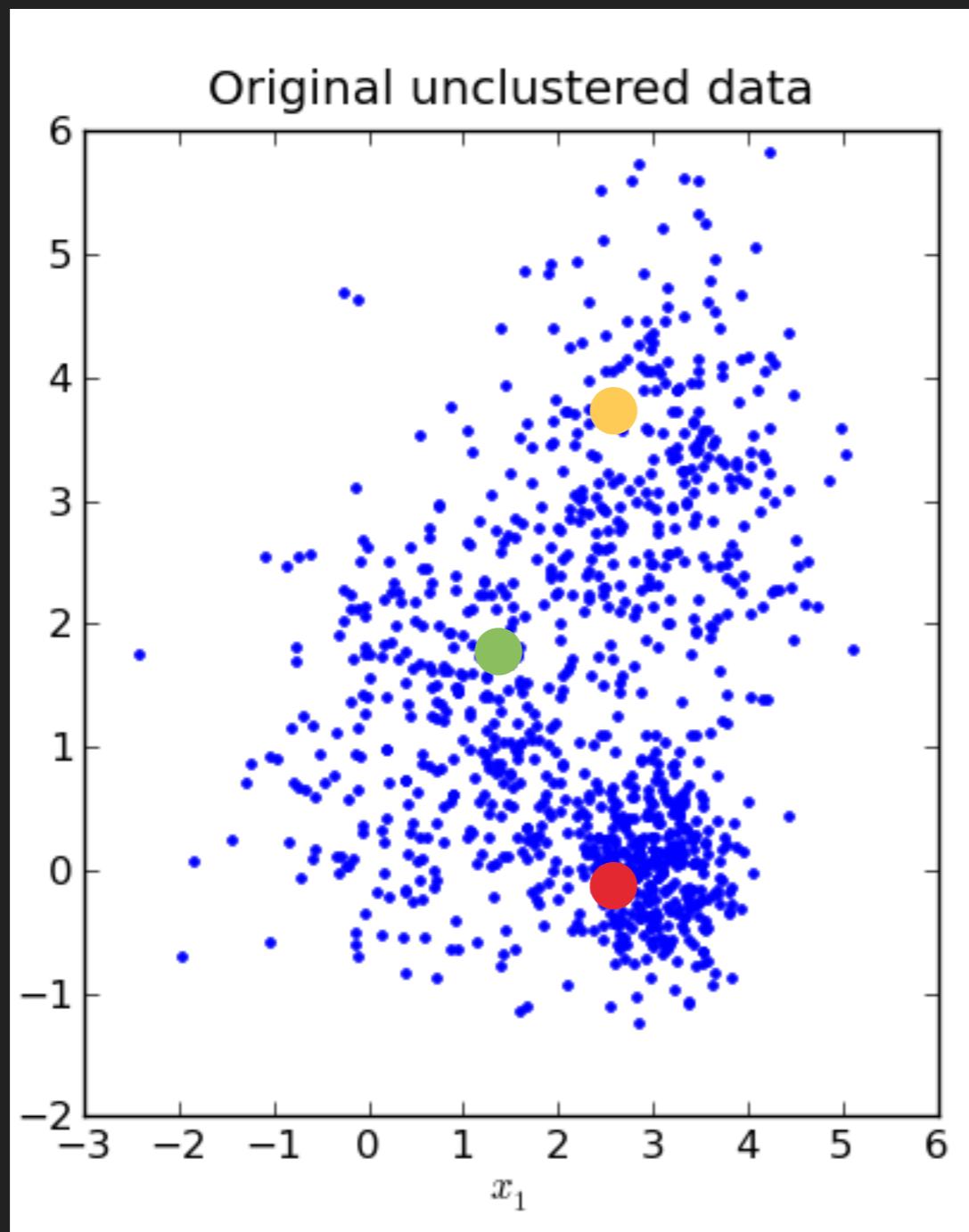


모든 데이터를 K개의 점 중  
제일 가까운 점과 연결

## 5. K-MEAN CLUSTERING

---

# K-MEAN CLUSTERING?

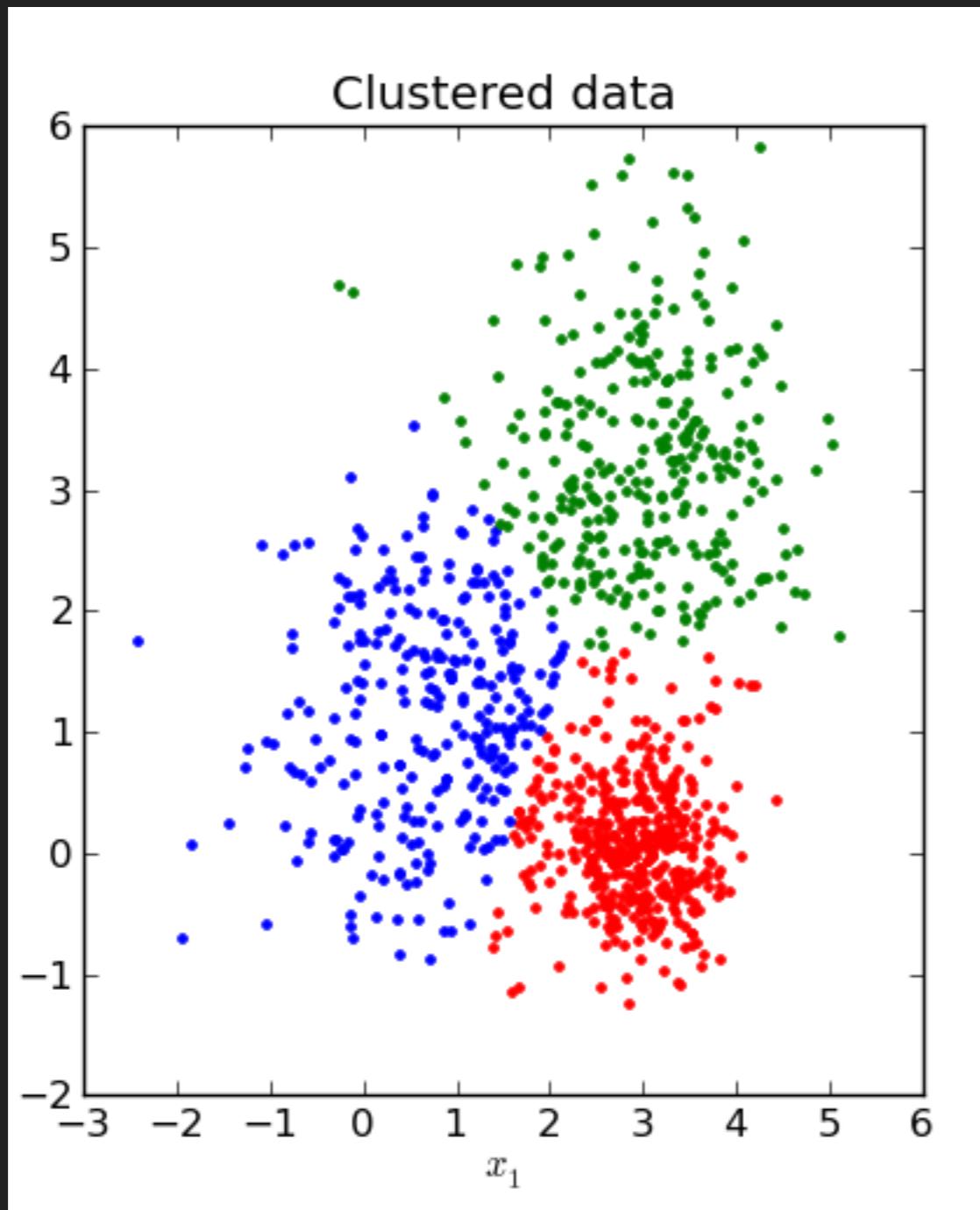


연결된 점들의 평균 좌표를 구해서  
그 좌표로 이동

## 5. K-MEAN CLUSTERING

---

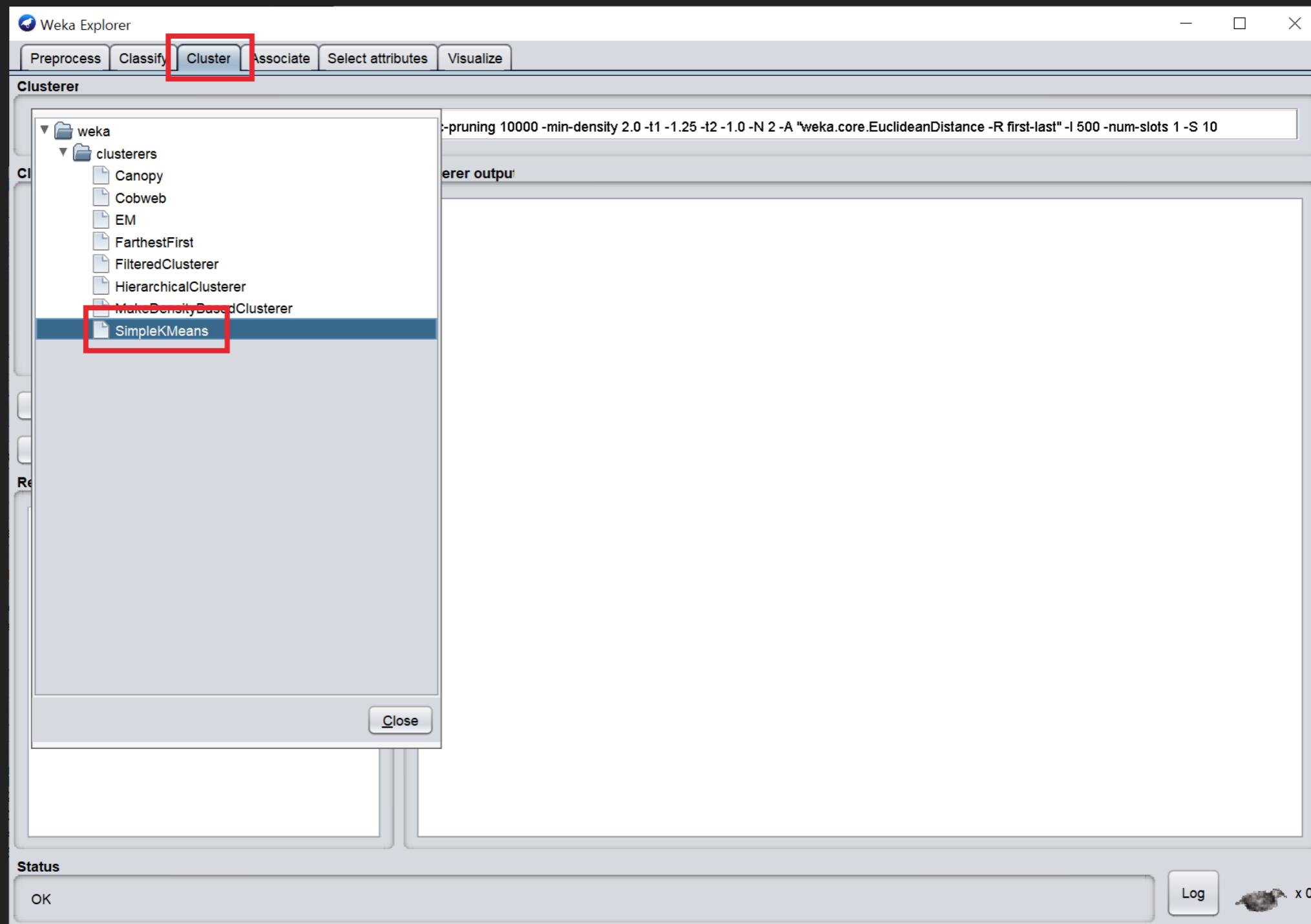
# K-MEAN CLUSTERING?



움직임이 없어질 때까지 반복

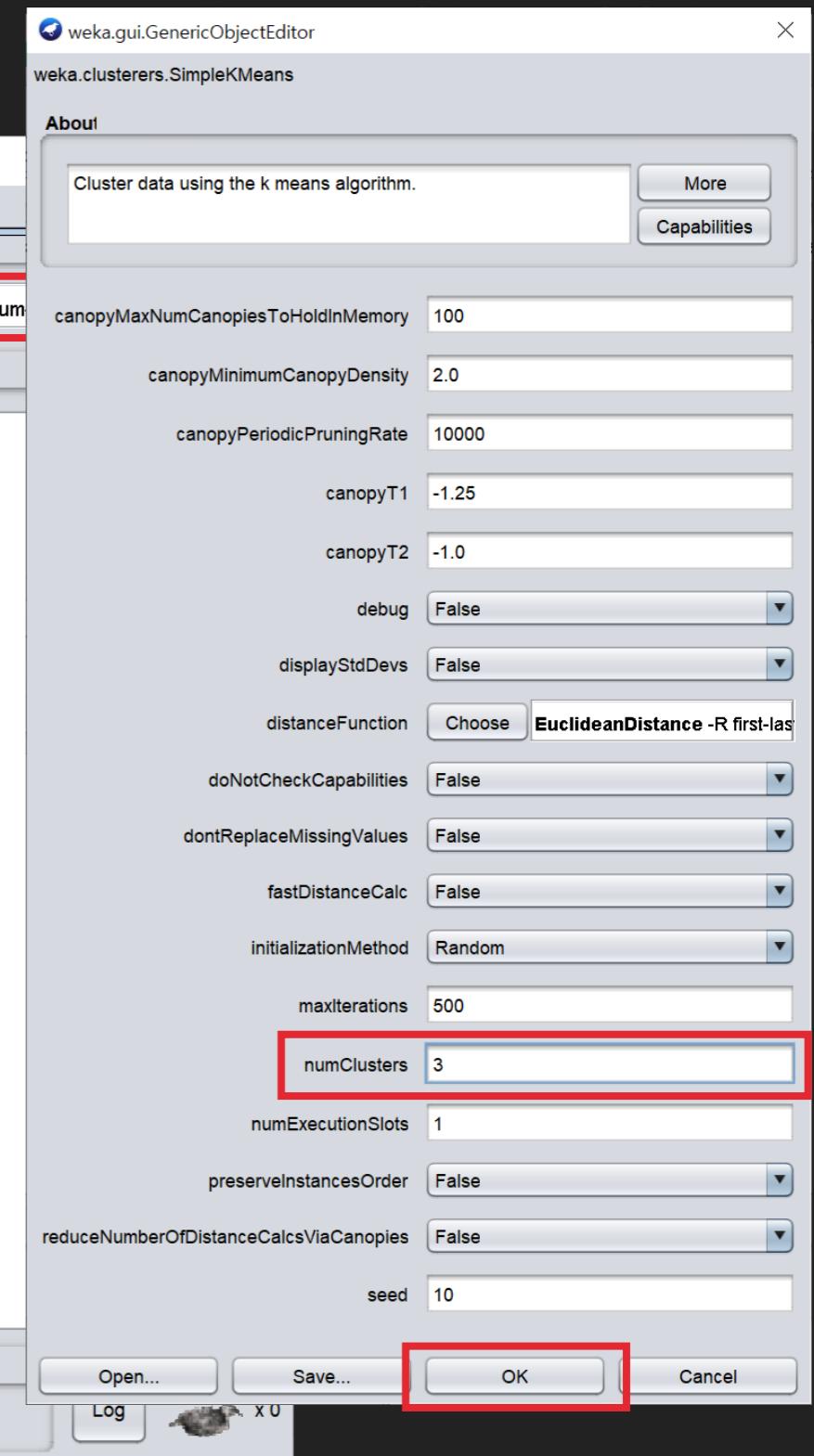
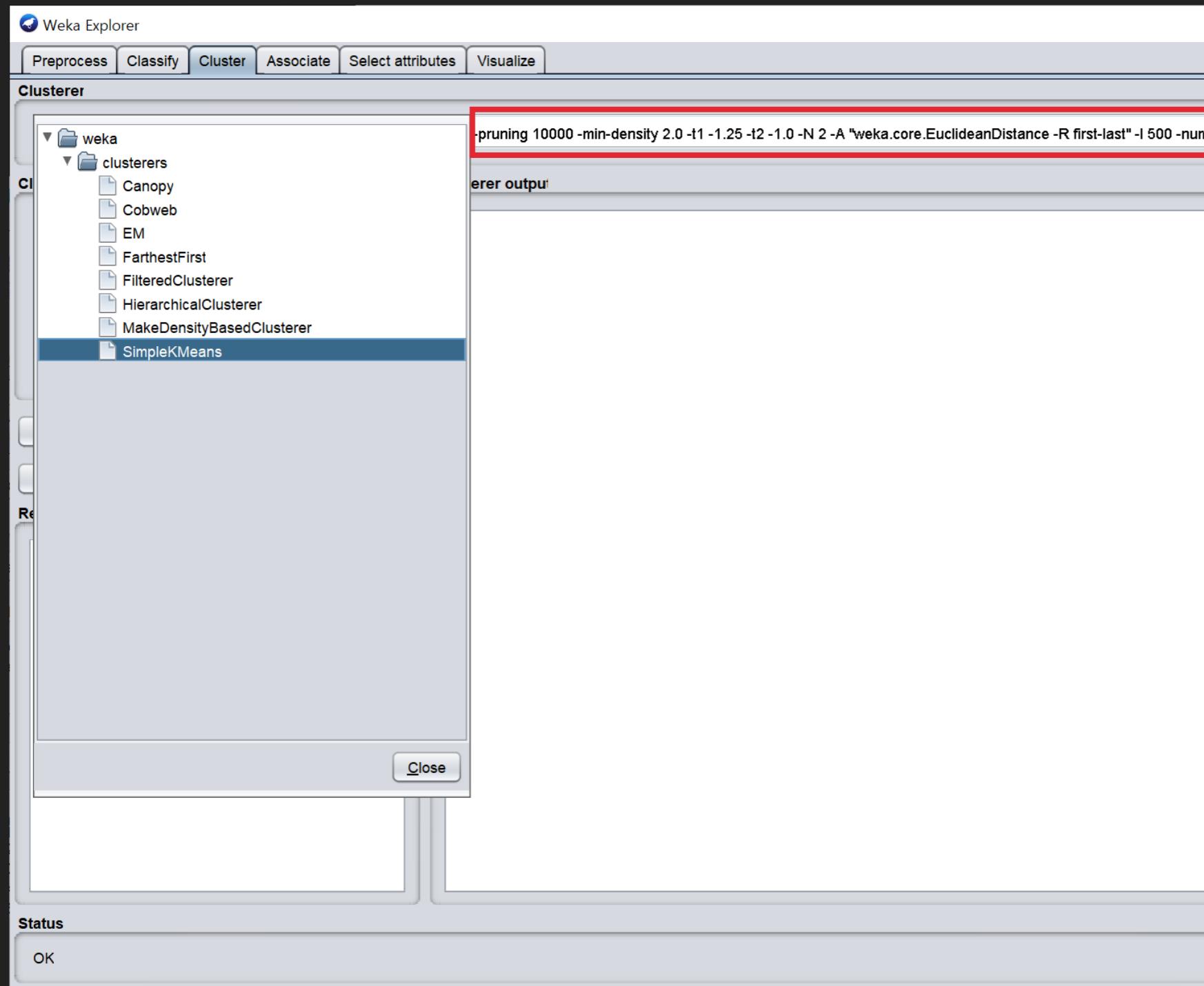
## 5. K-MEAN CLUSTERING

WEKA



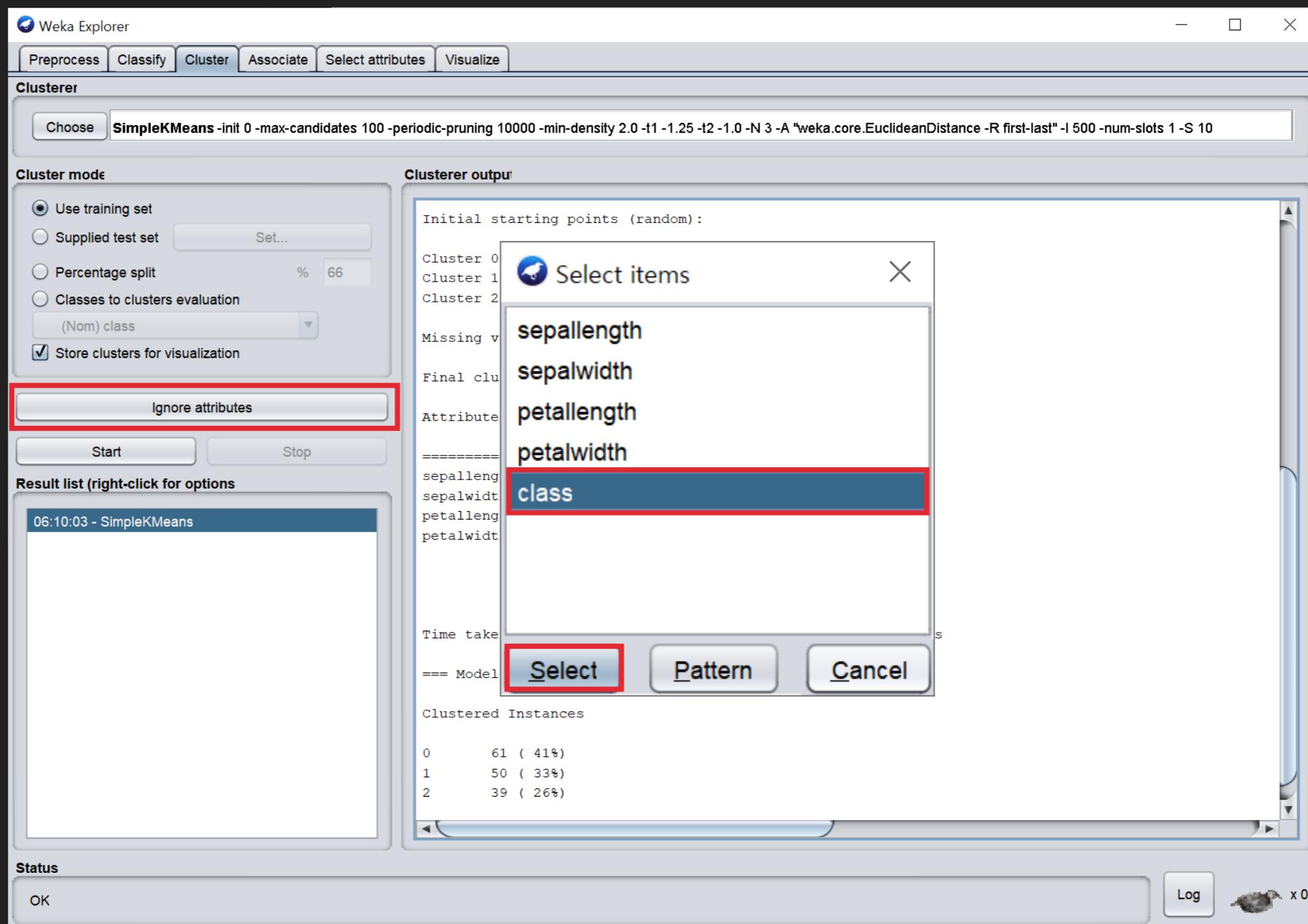
## 5. K-MEAN CLUSTERING

# WEKA



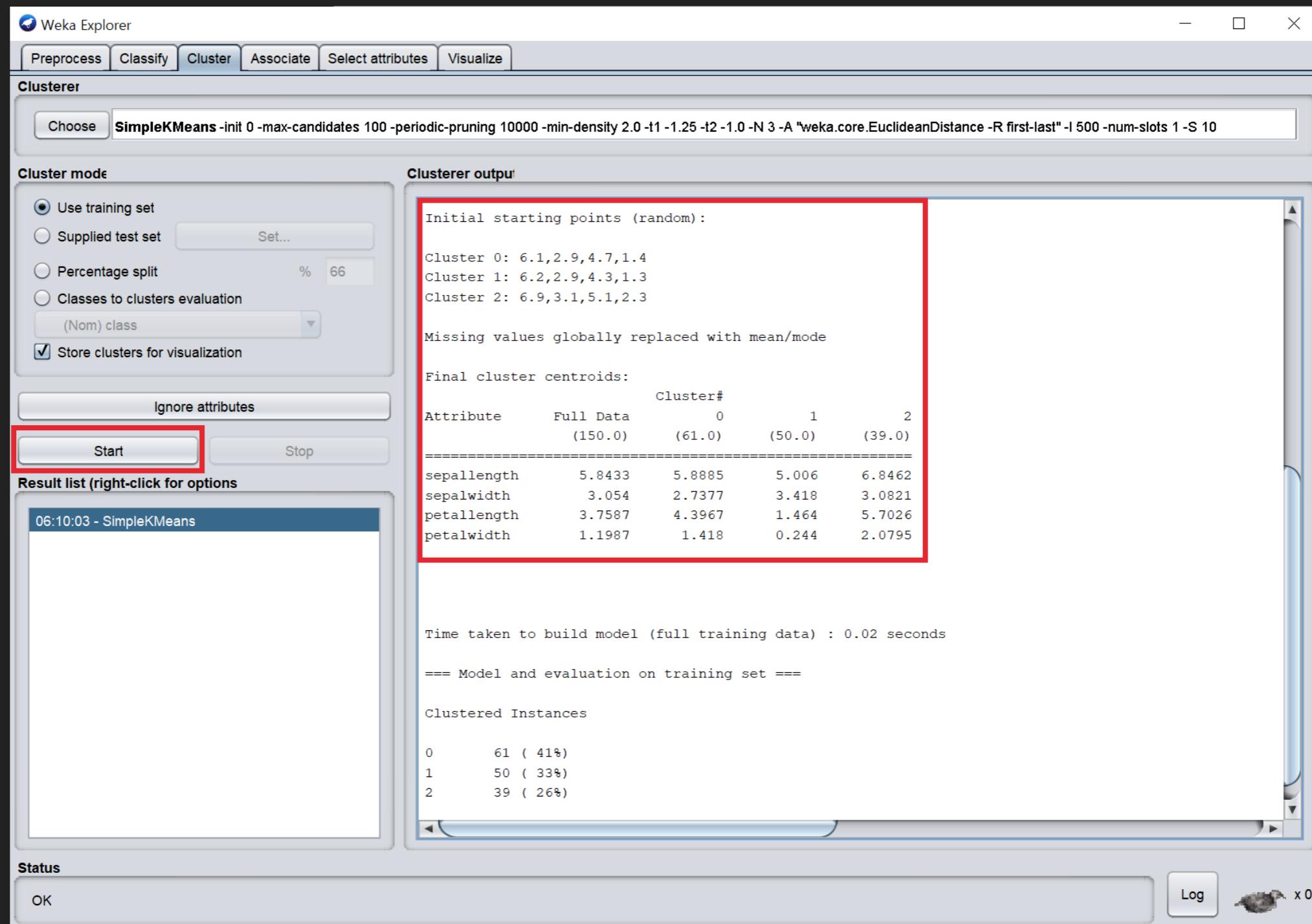
## 5. K-MEAN CLUSTERING

WEKA



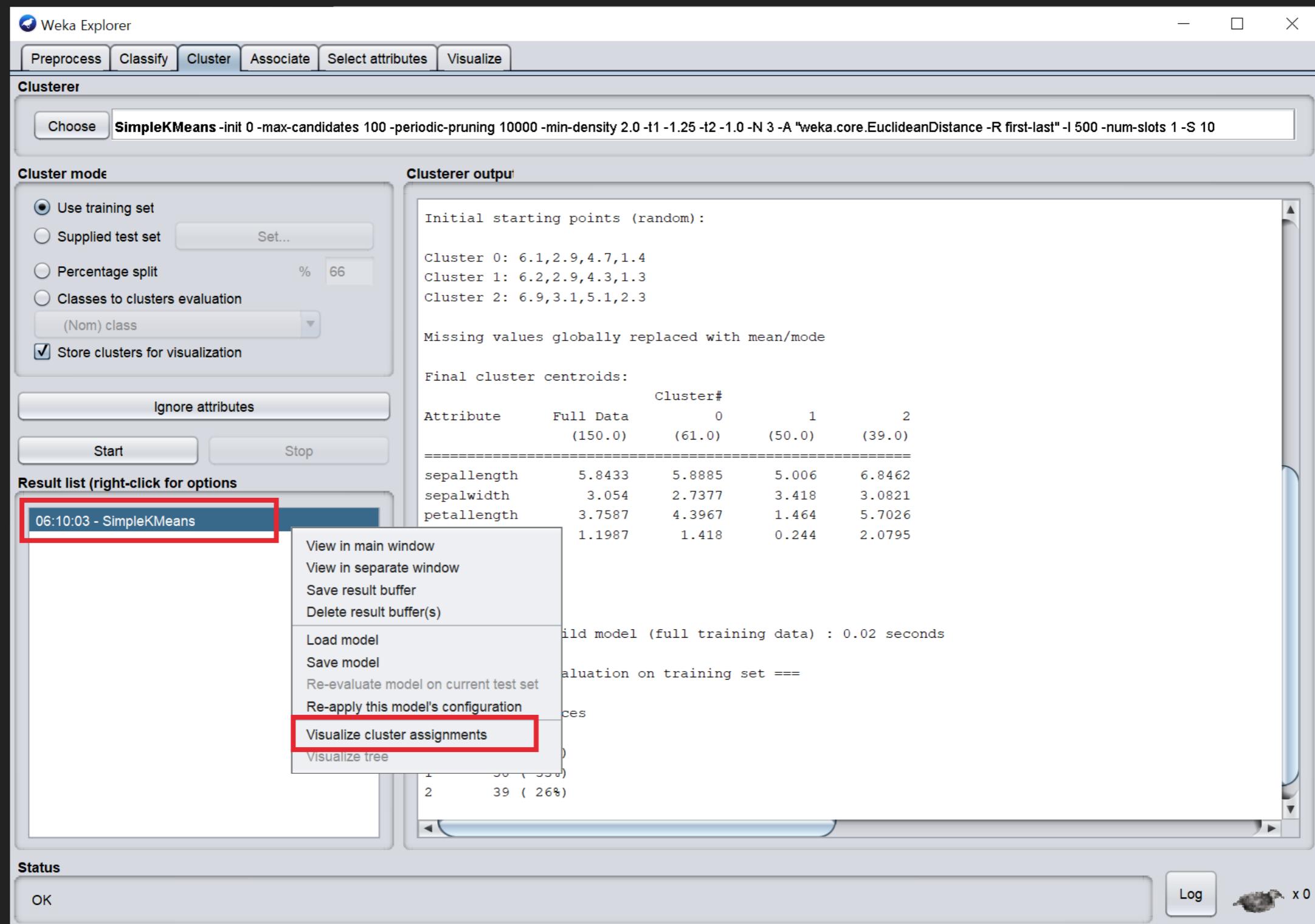
## 5. K-MEAN CLUSTERING

WEKA



## 5. K-MEAN CLUSTERING

# WEKA



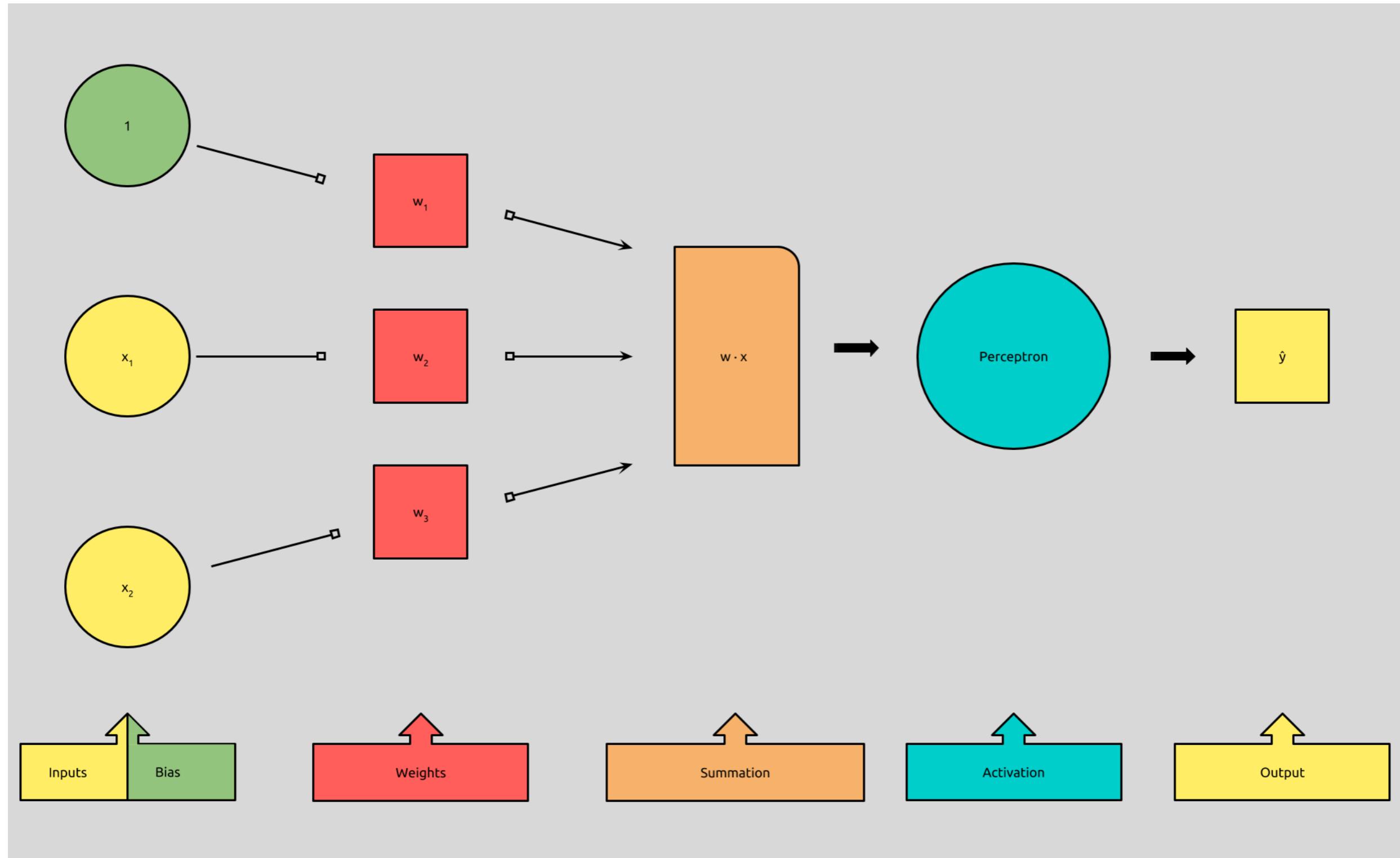
## 5. K-MEAN CLUSTERING

WEKA



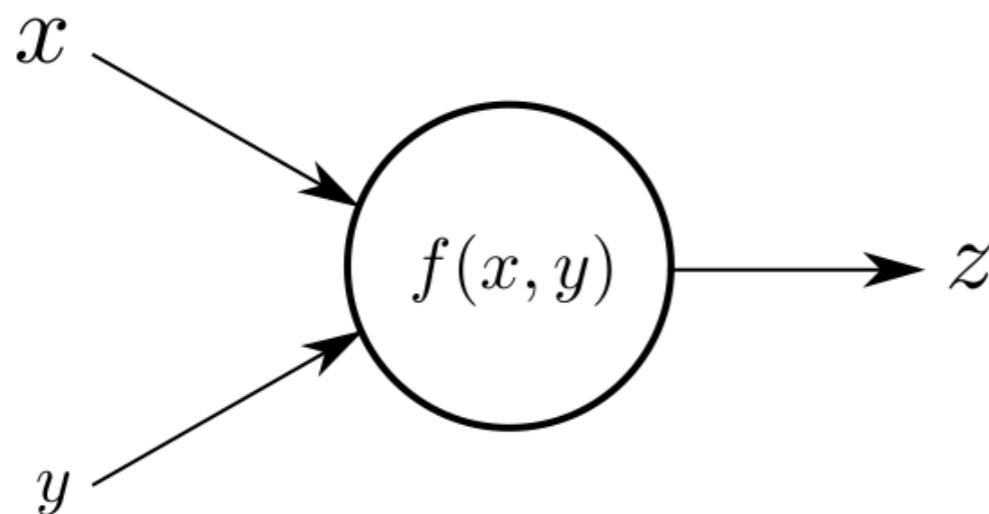
## 6. ARTIFICIAL NEURAL NETWORK

### PERCEPTRON

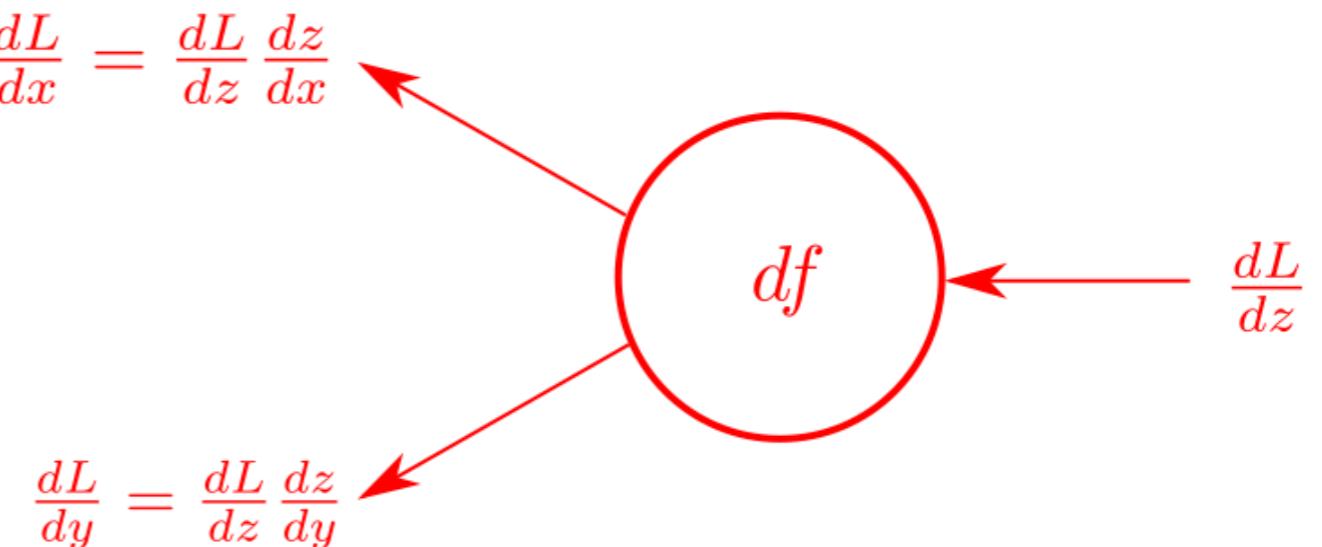


# PERCEPTRON

Forwardpass



Backwardpass

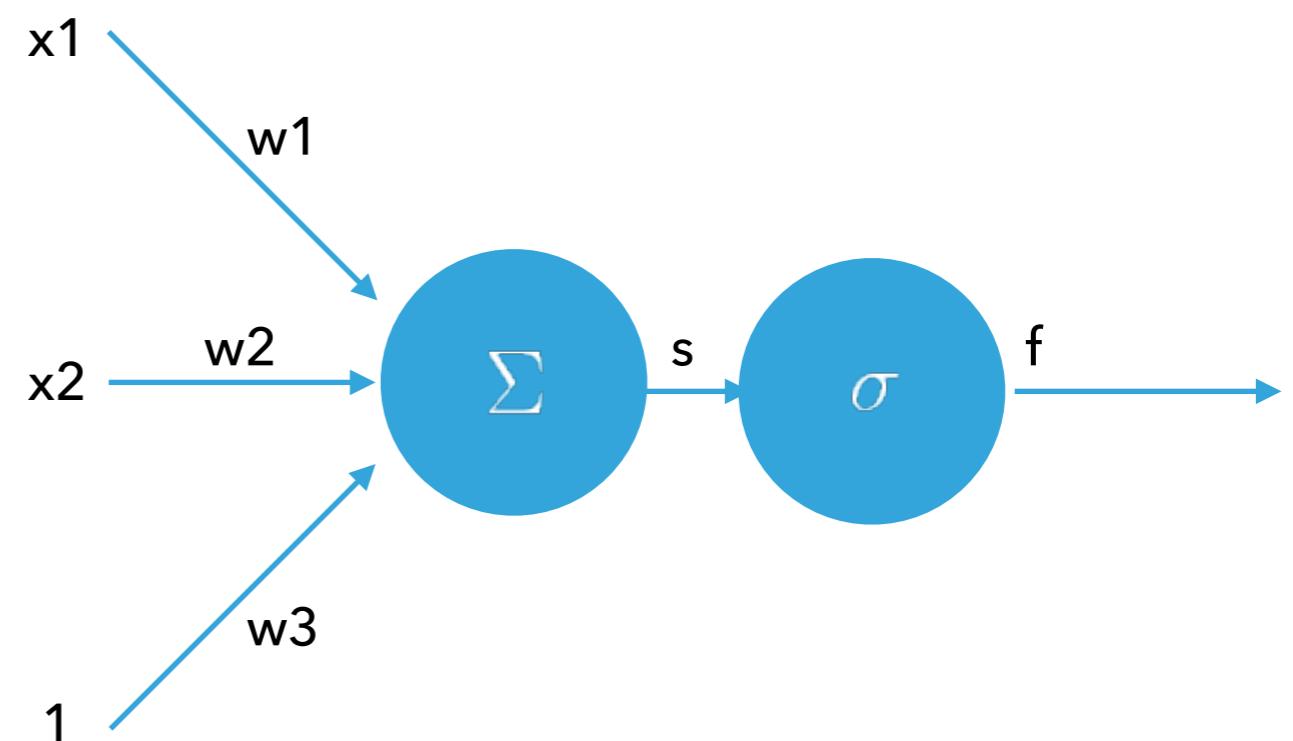
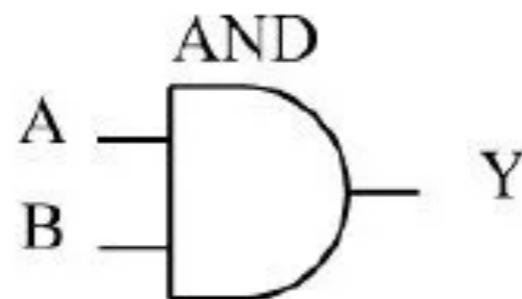


## 6. ARTIFICIAL NEURAL NETWORK

---

### MAKE AND GATE WITH PERCEPTRON

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



## 6. ARTIFICIAL NEURAL NETWORK

---

# MAKE AND GATE WITH PERCEPTRON

```
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))

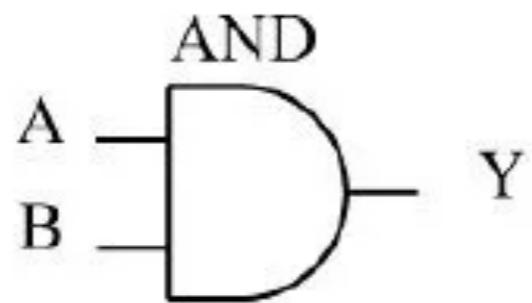
    for epoch in range(501):
        if epoch % 100 == 0:
            print('epoch: ', epoch)
        for x_bar, y_bar in zip(x, y):
            s = x_bar[0] * w[0] + x_bar[1] * w[1] + w[2]
            f = activate(s)
            loss = (f - y_bar)**2
            if epoch % 100 == 0:
                print('Loss: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

## 6. ARTIFICIAL NEURAL NETWORK

---

### MAKE AND GATE WITH PERCEPTRON

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



A diagram illustrating the backpropagation of error gradients through a neural network model of an AND gate. The network consists of three layers: an input layer with nodes A and B, a hidden layer with a summation node ( $\Sigma$ ), and an output layer with a sigmoid node ( $\sigma$ ). Blue arrows indicate the flow of gradients from the output layer back through the hidden layer to the input layer. The equations show the calculation of the gradient of the loss function L with respect to the weights w1, w2, and w3.

$$\frac{dL}{dw_1} = \frac{dL}{ds} \frac{ds}{dw_1}$$
$$\frac{dL}{dw_2} = \frac{dL}{ds} \frac{ds}{dw_2}$$
$$\frac{dL}{dw_3} = \frac{dL}{ds} \frac{ds}{dw_3}$$
$$\frac{dL}{ds} = \frac{df}{ds} \frac{dL}{df}$$
$$\frac{dL}{df}$$

## 6. ARTIFICIAL NEURAL NETWORK

---

# MAKE AND GATE WITH PERCEPTRON

```
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
```

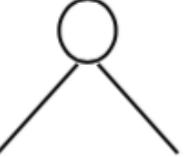
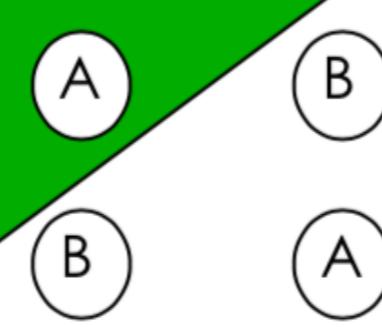
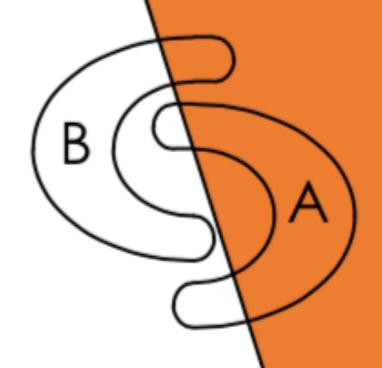
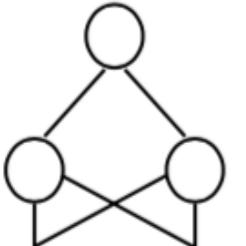
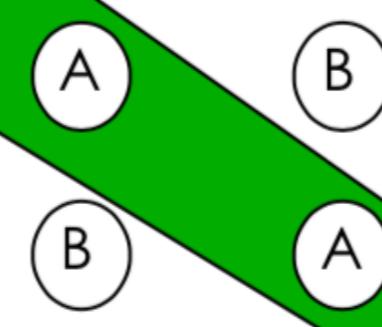
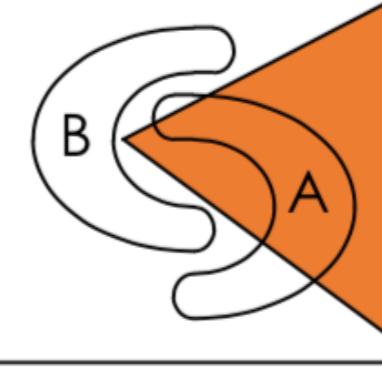
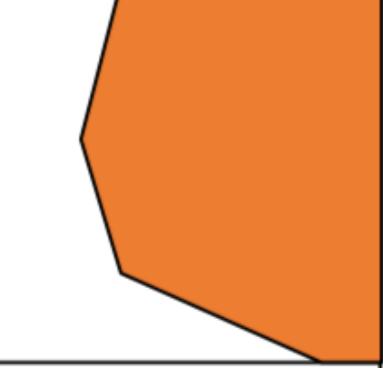
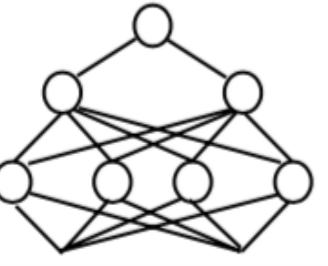
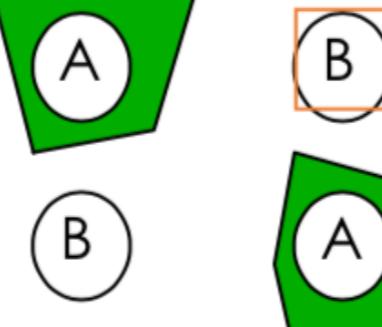
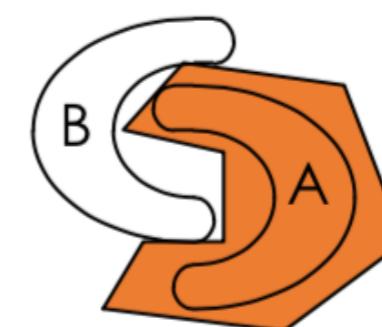
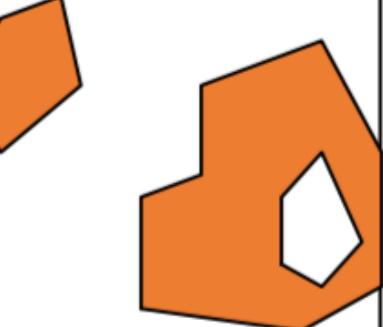
# WHY NEURAL NETWORK?

**Non Linearity**

## 6. ARTIFICIAL NEURAL NETWORK

---

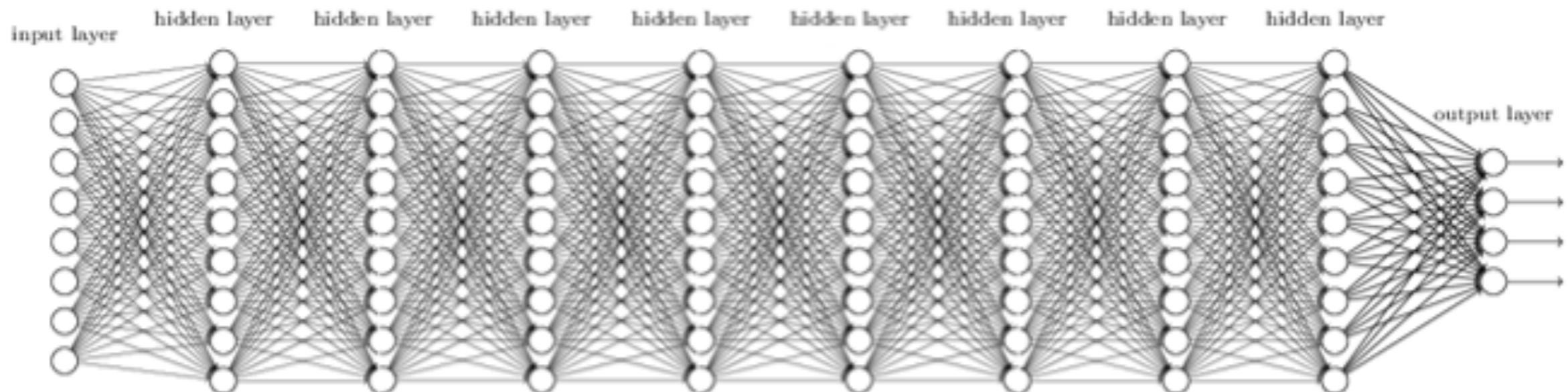
# WHY NEURAL NETWORK?

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

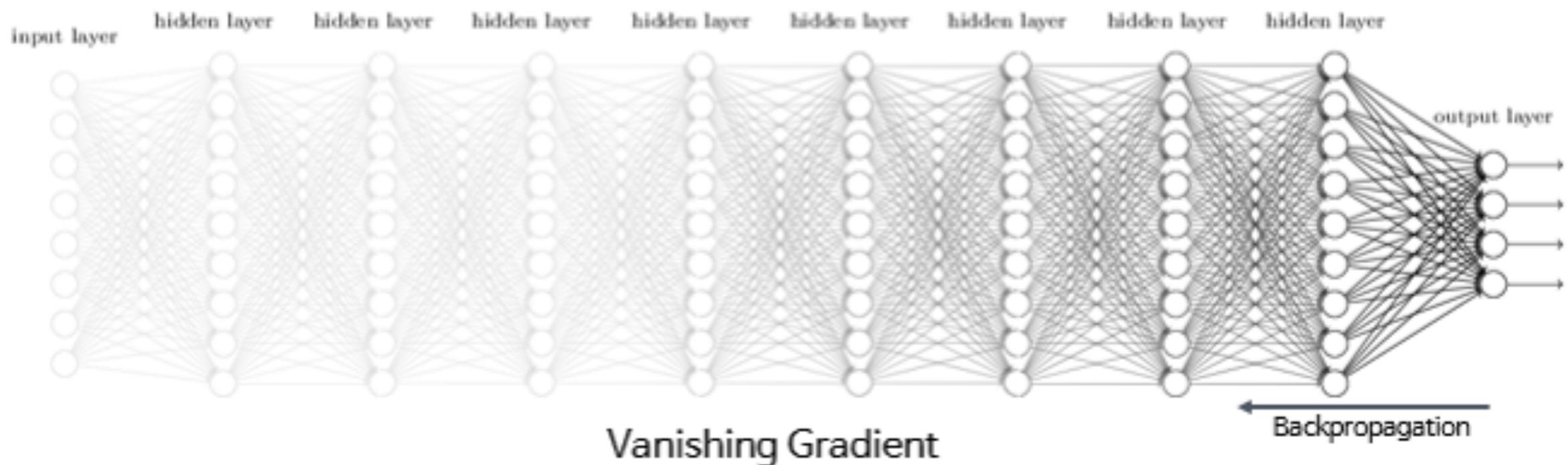
## 6. ARTIFICIAL NEURAL NETWORK

---

# THEN DEEPER THE BETTER?



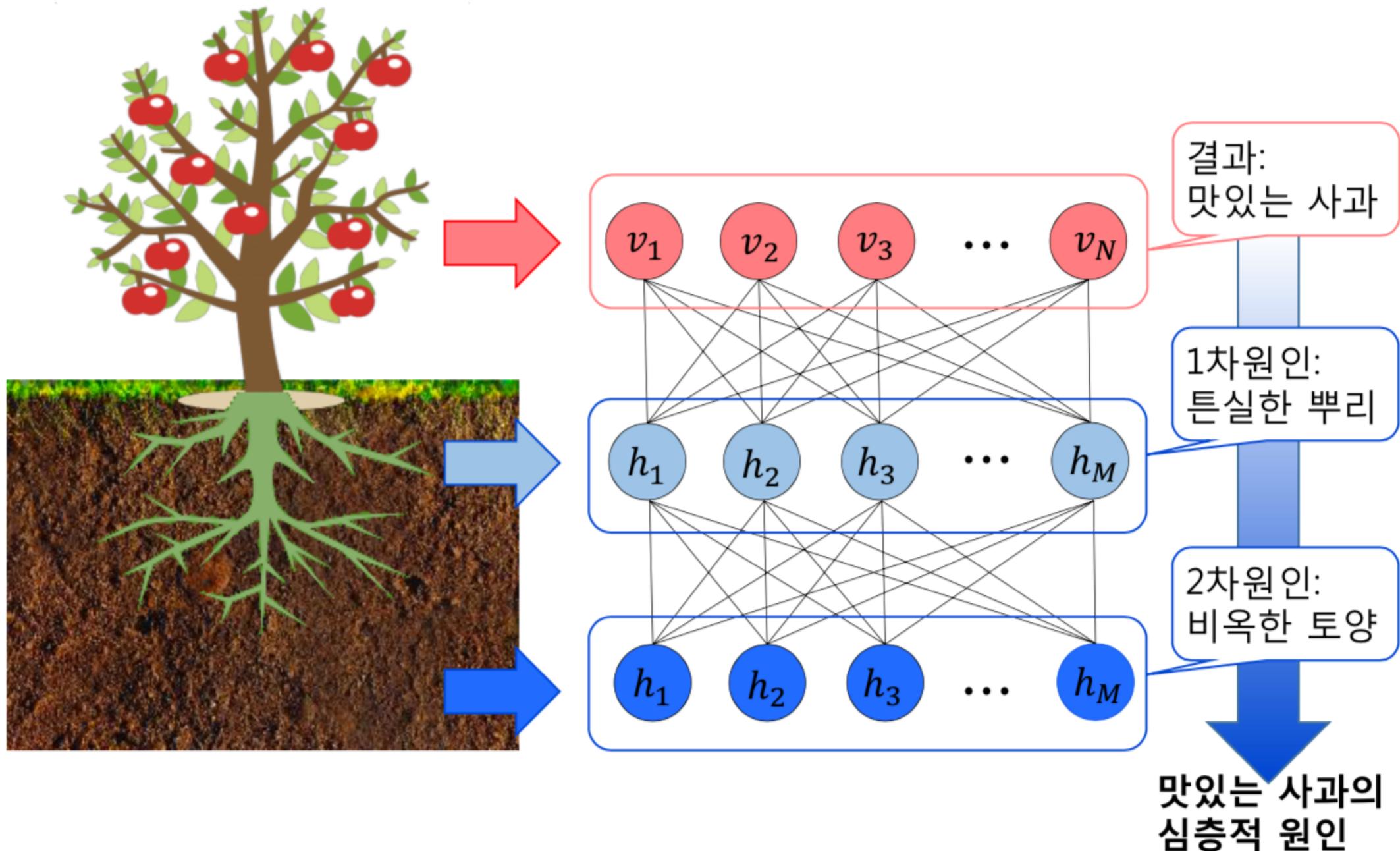
Deep Neural Network



Vanishing Gradient

Backpropagation

# WHAT IS DEEP LEARNING ACTUALLY?



**Deep means '심층' not '깊은'**