python + AI

Introduction to Artificial Intelligence with Python

Learning

Supervised Learning

supervised learning

given a data set of input-output pairs, learn a function to map inputs to outputs

classification

supervised learning task of learning a function mapping an input point to a discrete category

Date	**Humidity** (relative humidity)	**Pressure** (sea level, mb)	Rain

Date	Humidity (relative humidity)	Pressure (sea level, mb)	Rain
January 1	93%	999.7	Rain
January 2	49%	1015.5	No Rain
January 3	79%	1031.1	No Rain
January 4	65%	984.9	Rain
January 5	90%	975.2	Rain

$f(humidity, pressure)$

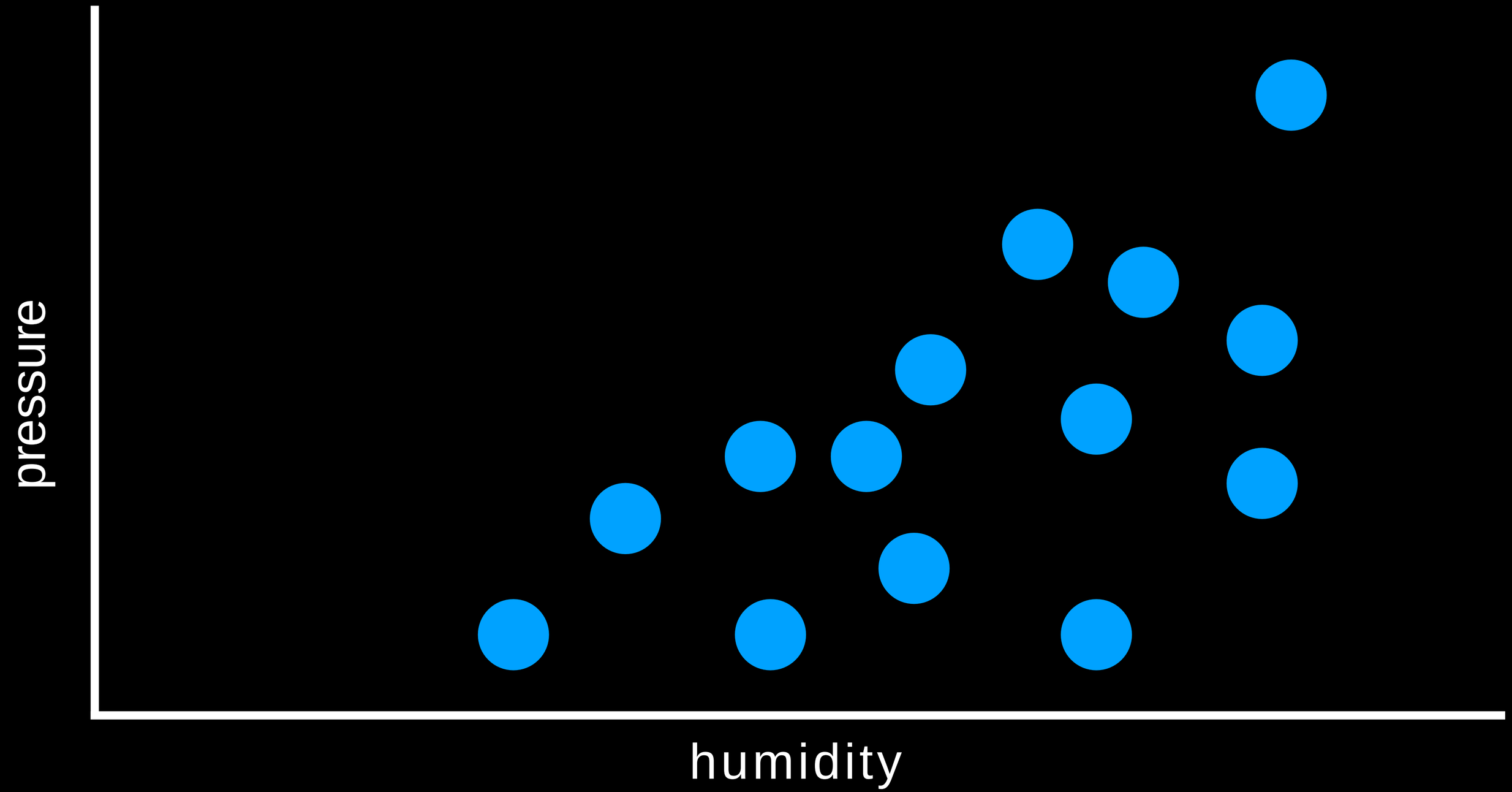
$f(93, 999.7) = \text{Rain}$

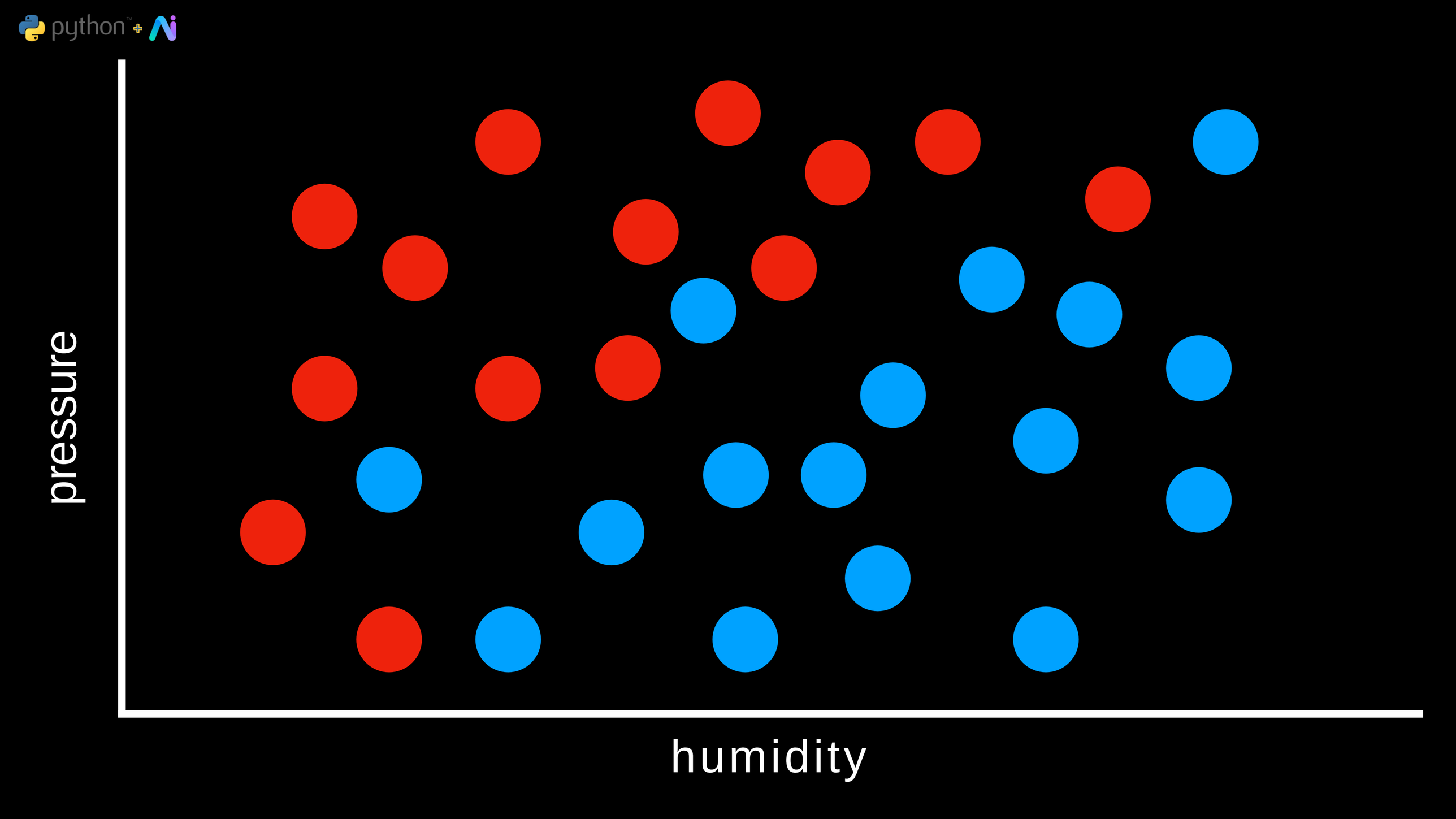
$f(49, 1015.5) = \text{No Rain}$

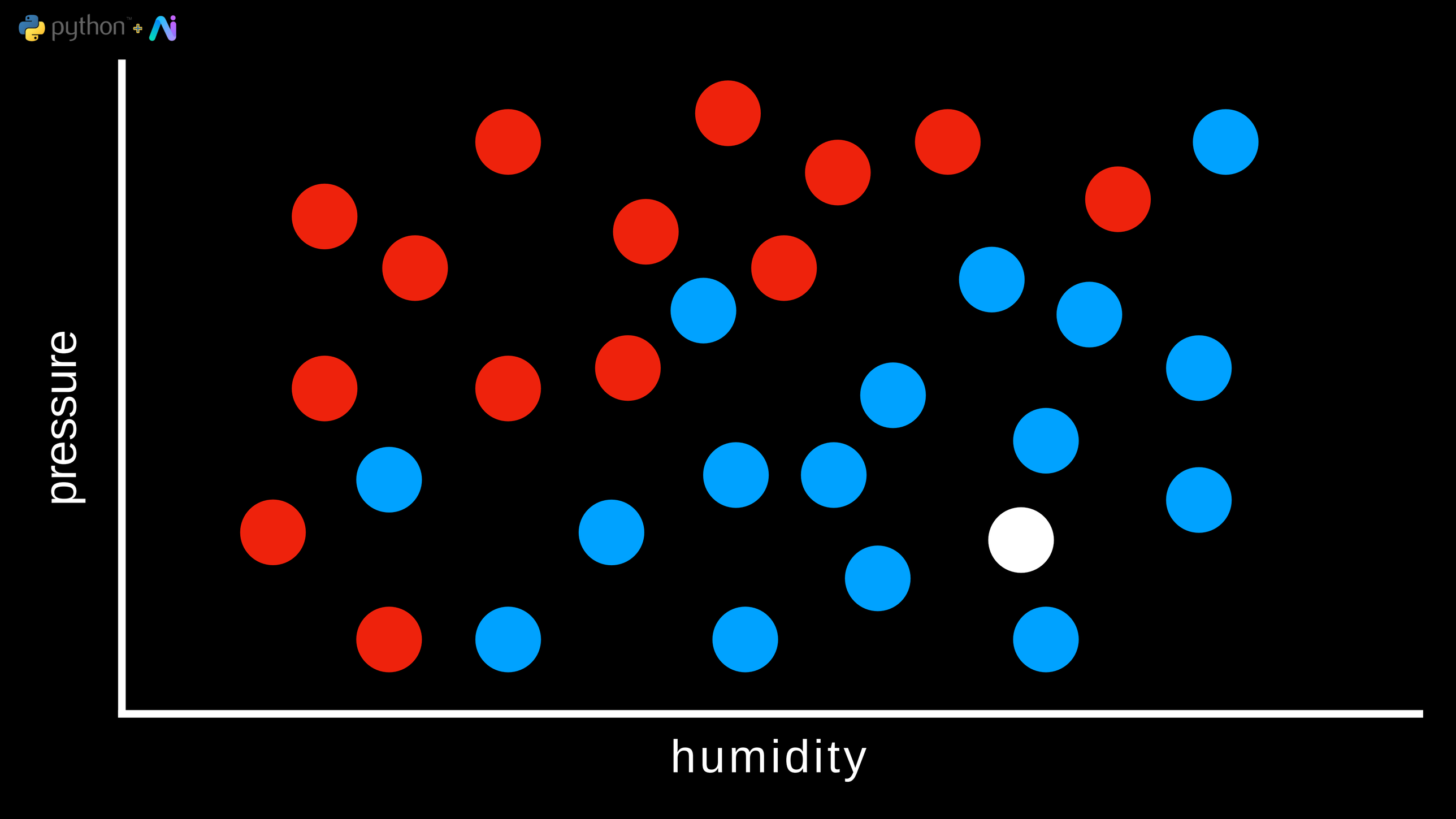
$f(79, 1031.1) = \text{No Rain}$

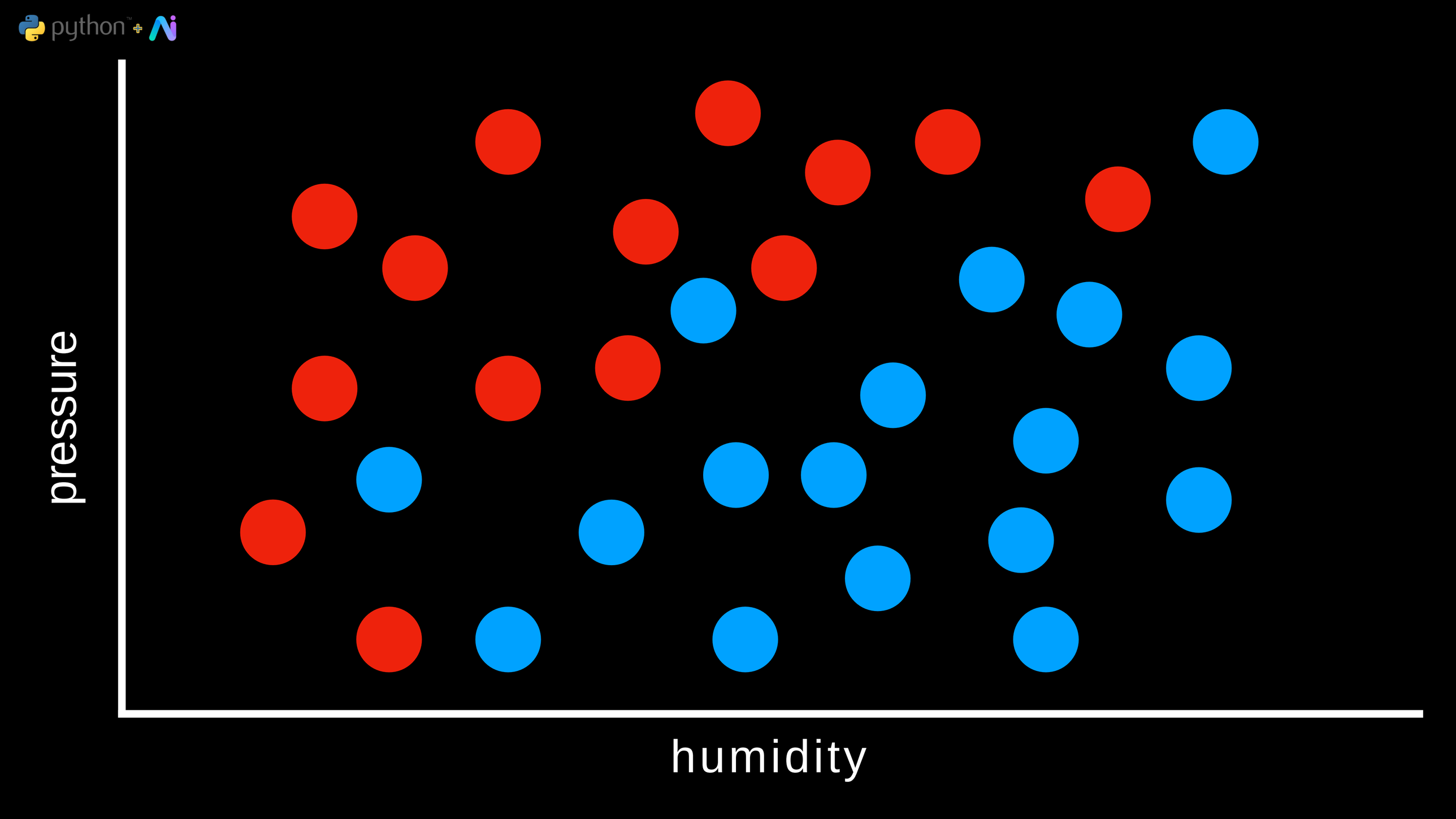
$h(humidity, pressure)$


pressure
humidity


pressure
humidity

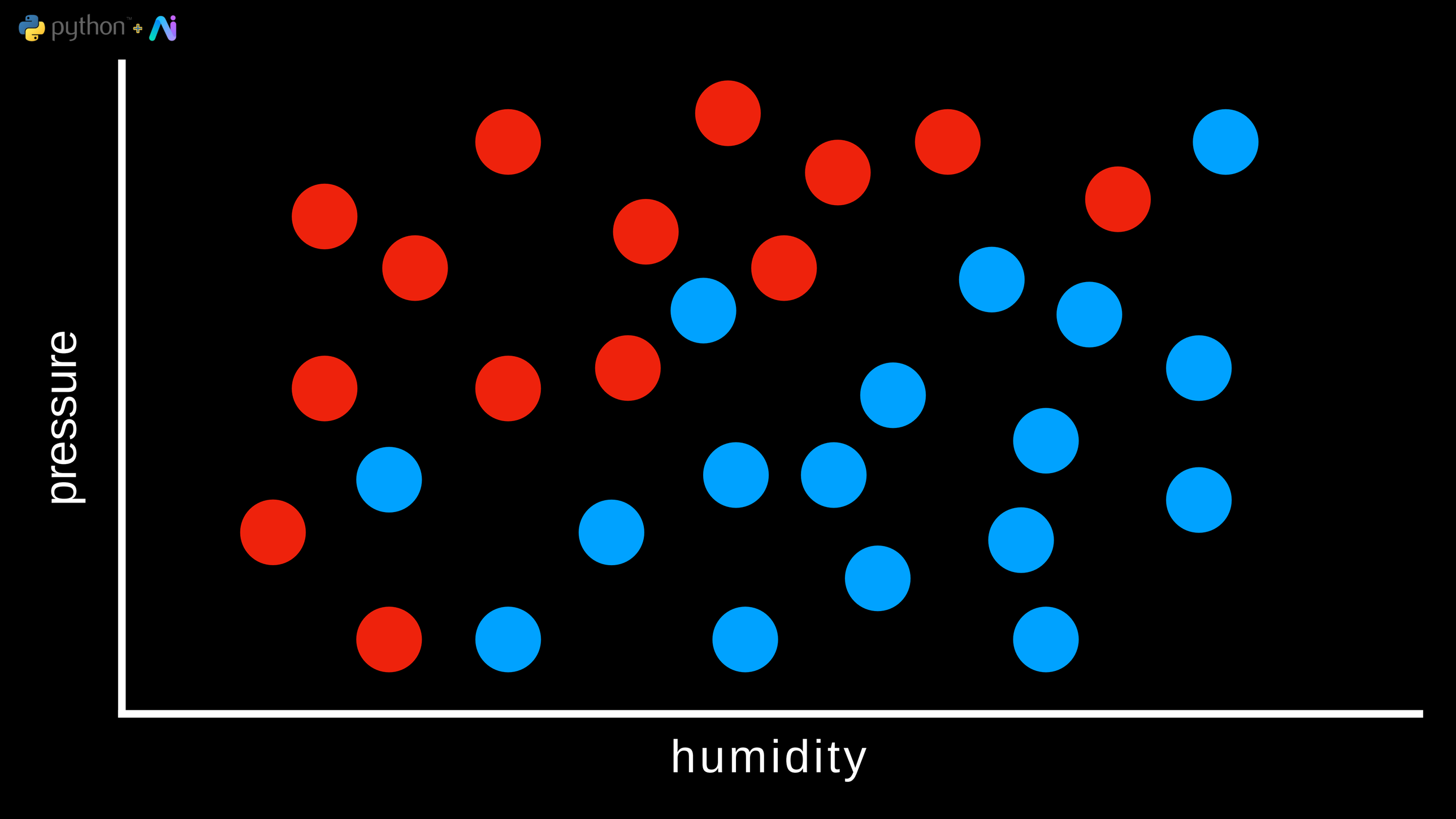

pressure
humidity

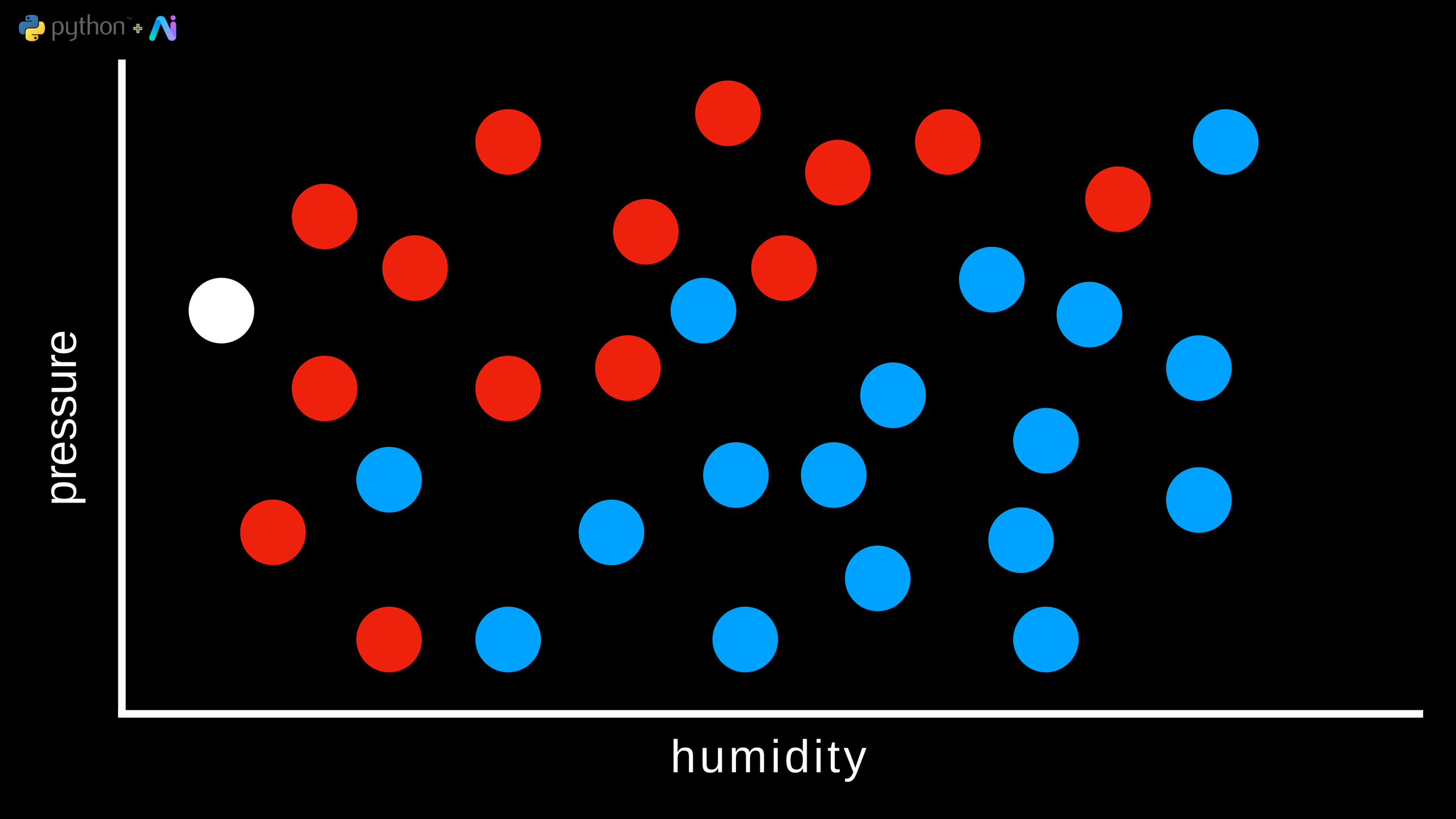

pressure
humidity

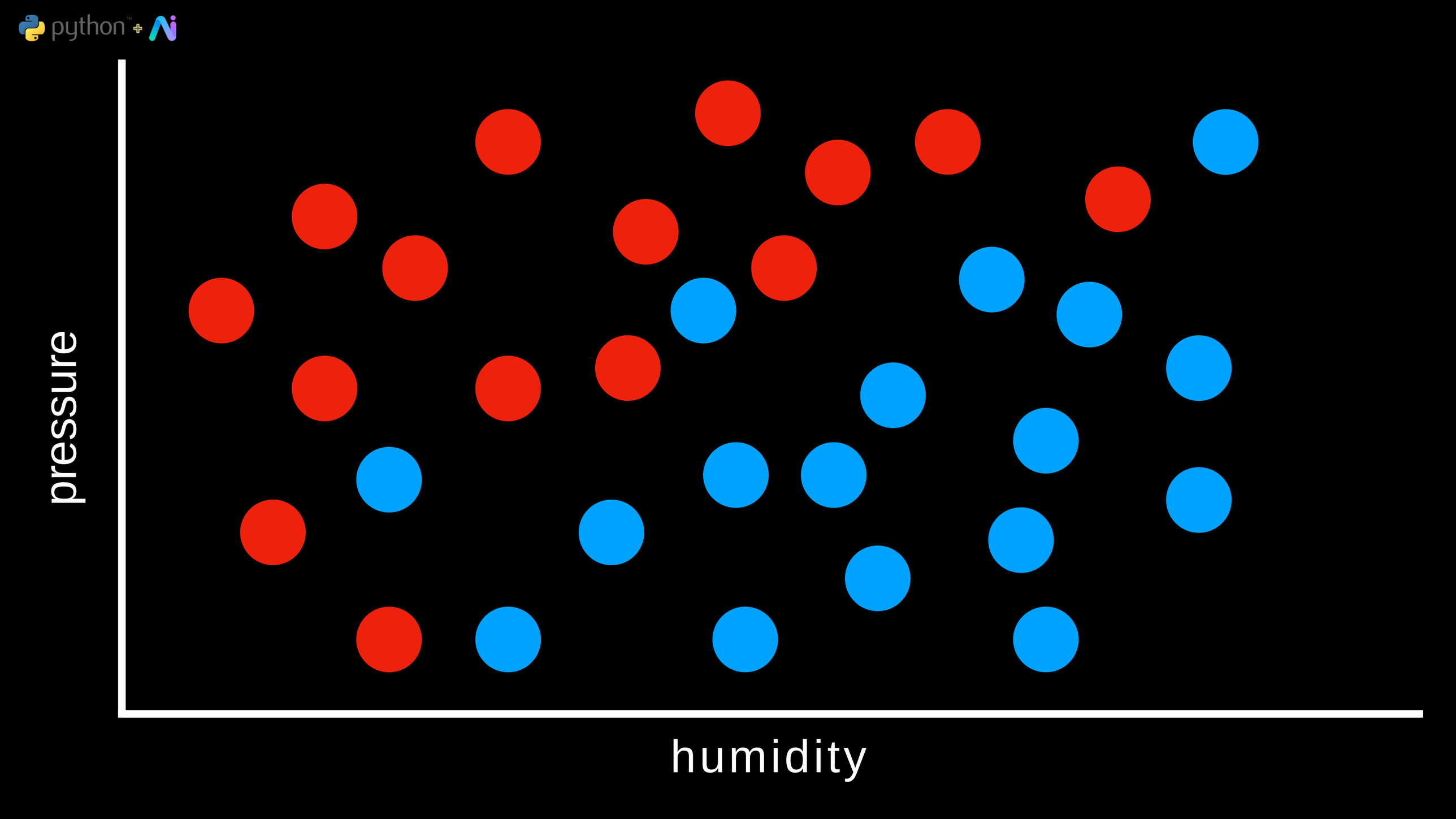

pressure
humidity

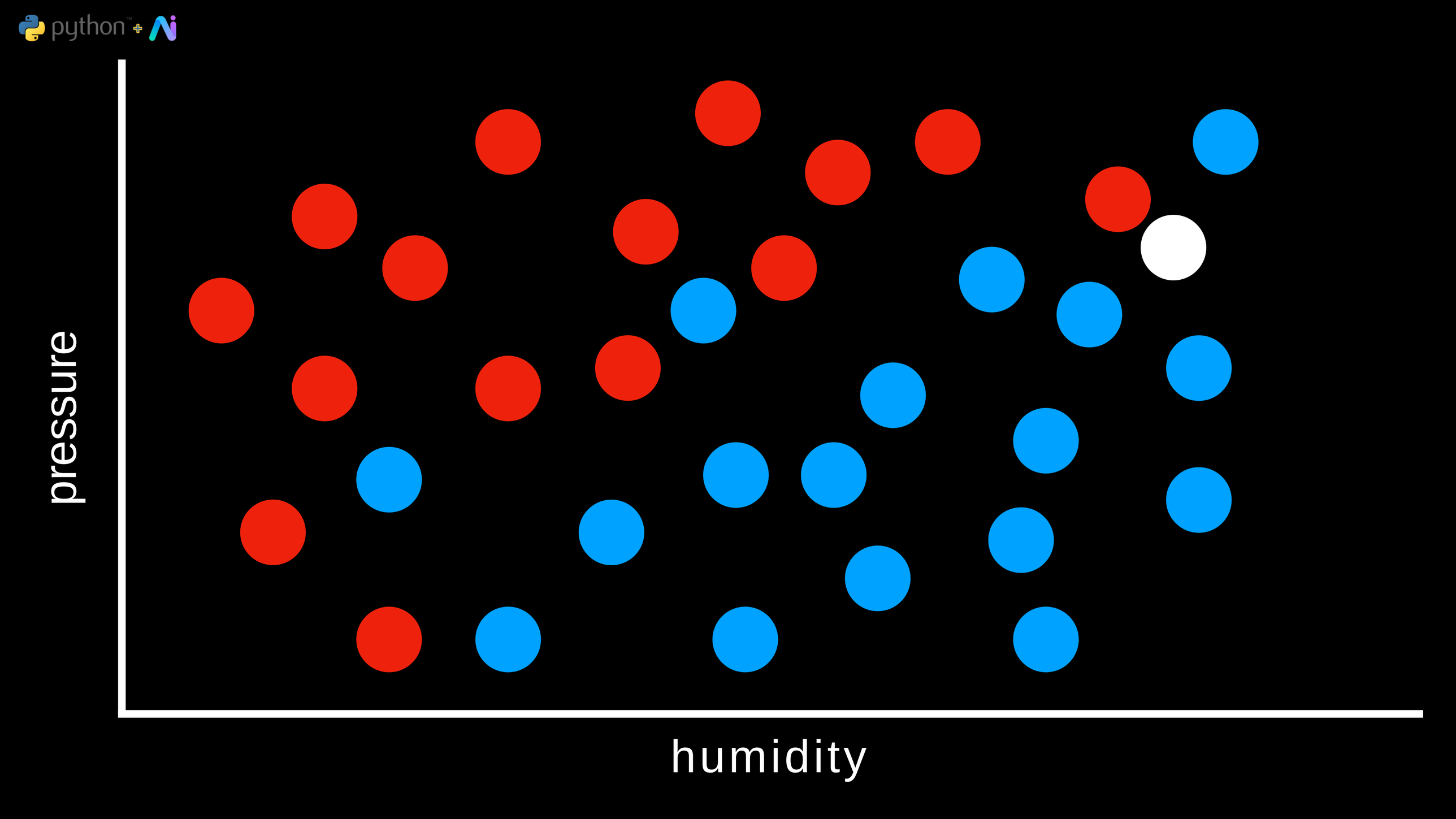
nearest-neighbor classification

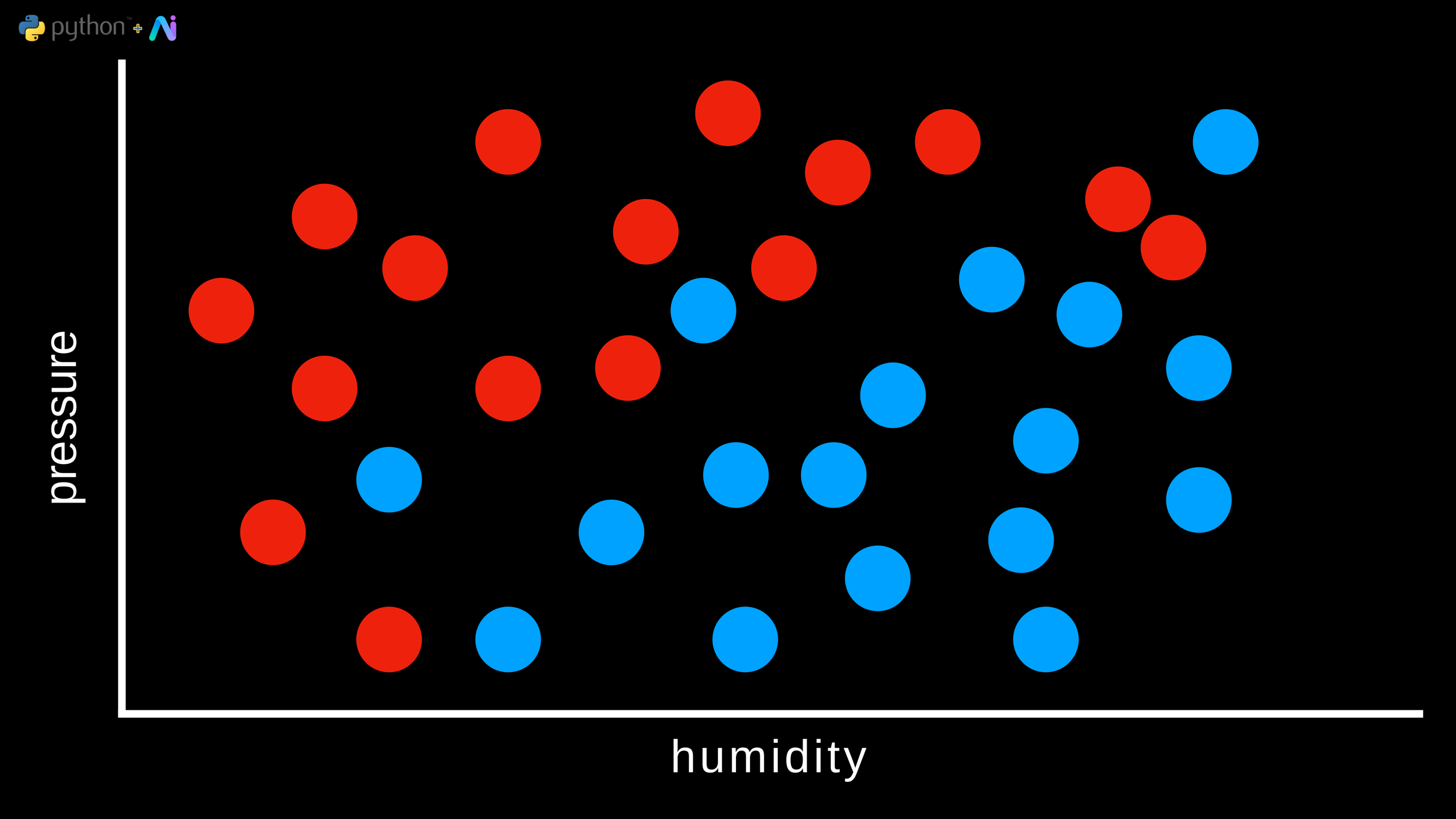
algorithm that, given an input, chooses the class of the nearest data point to that input

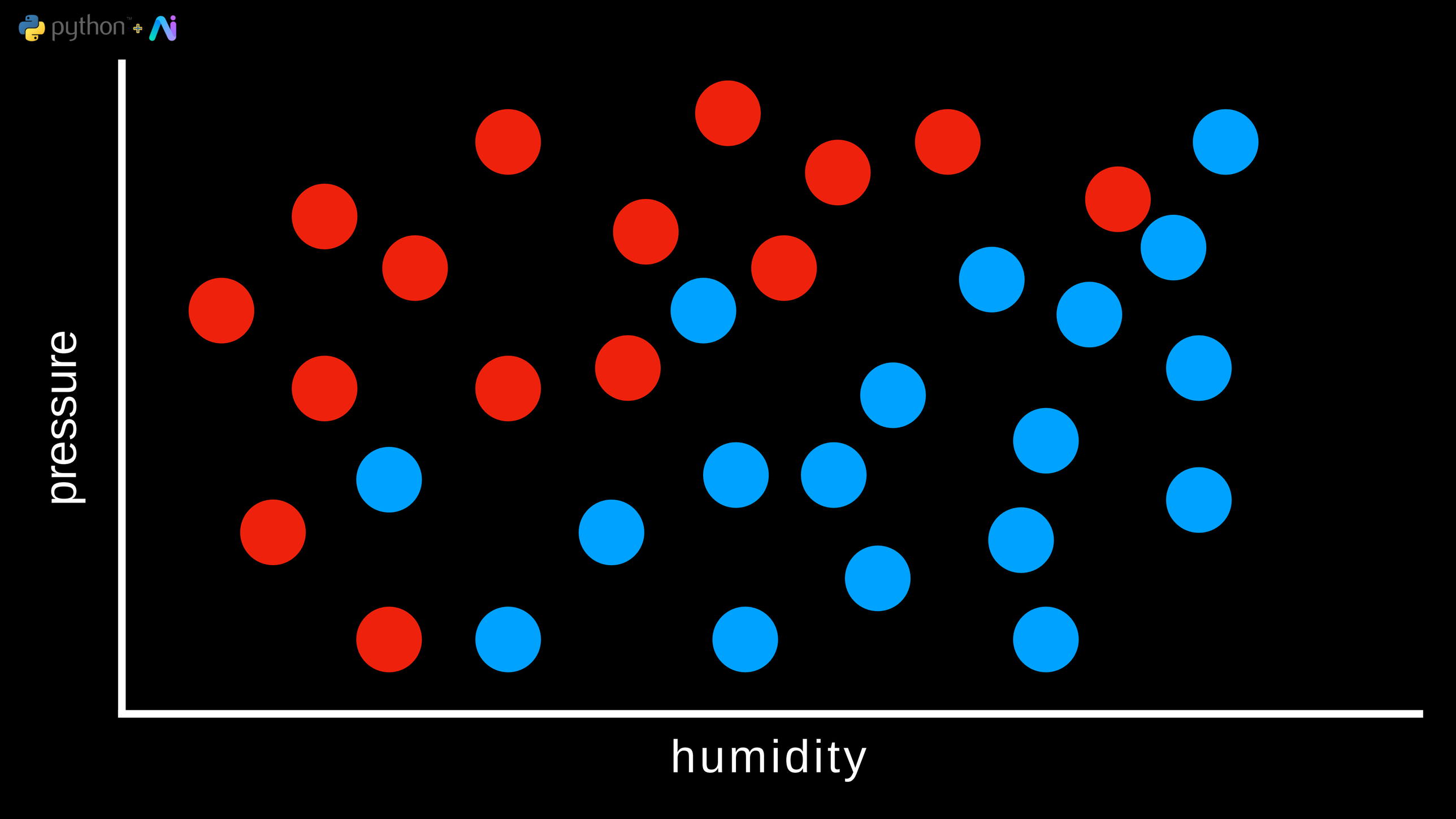

pressure
humidity


pressure
humidity


pressure
humidity

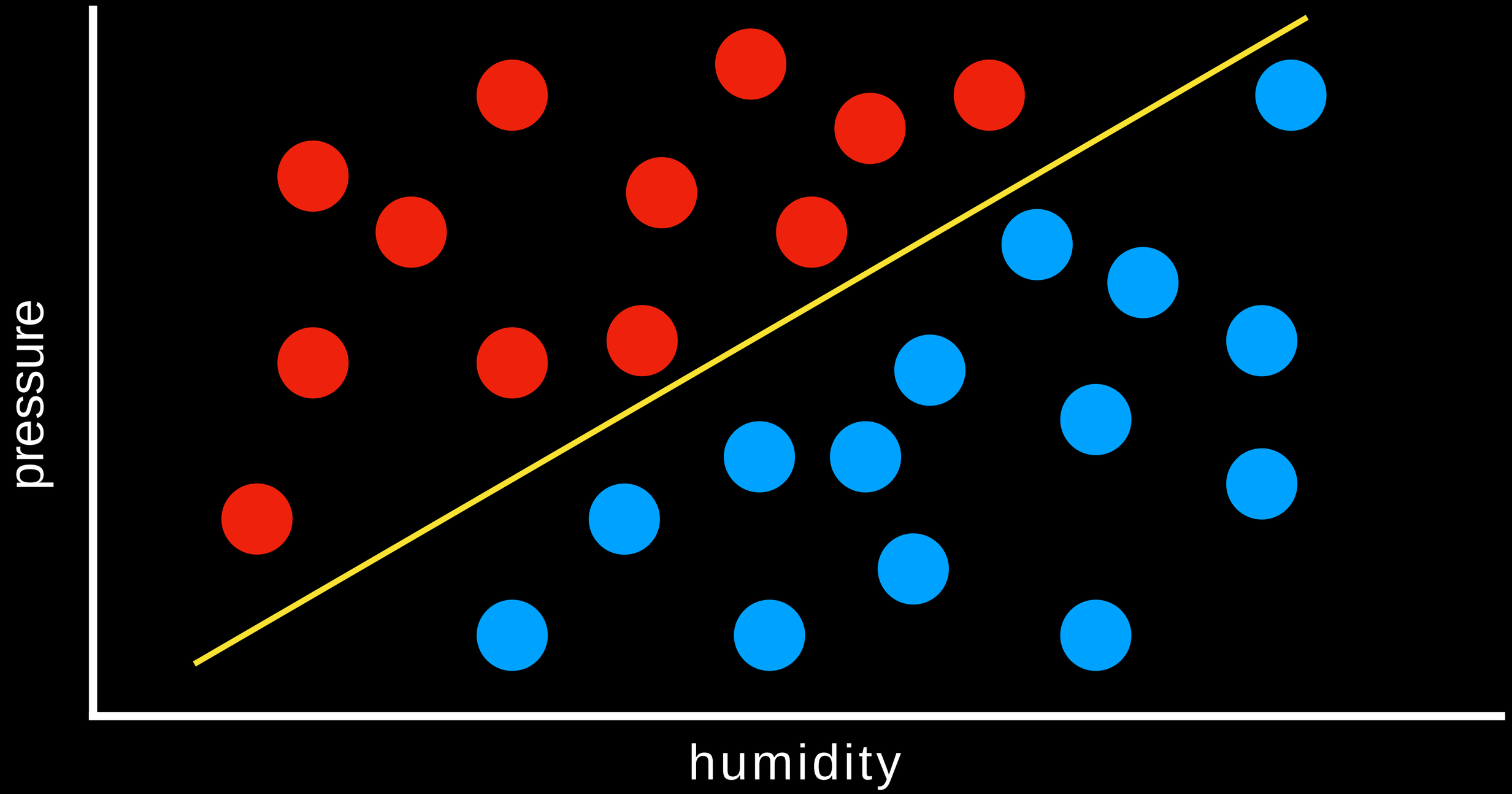

pressure
humidity

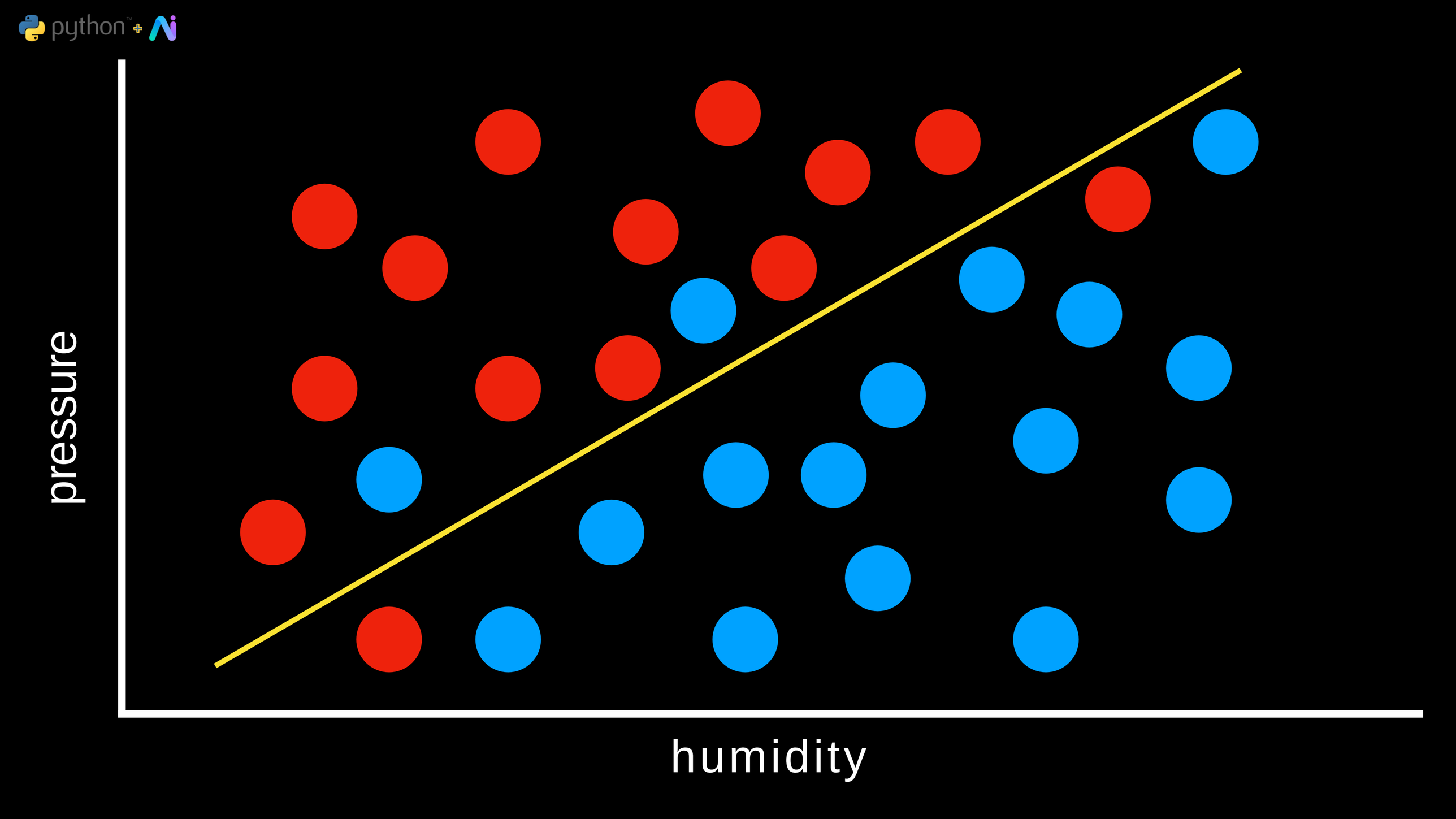

pressure
humidity


pressure
humidity

k-nearest-neighbor classification

algorithm that, given an input, chooses the most common class out of the k nearest data points to that input


pressure
humidity


pressure
humidity

x_1 = Humidity

x_2 = Pressure

$$h(x_1, x_2) = \begin{cases} \text{Rain} & \text{if } w_0 + w_1 x_1 + w_2 x_2 \geq 0 \\ \text{No Rain} & \text{otherwise} \end{cases}$$

Weight Vector $\mathbf{w}$: (w_0, w_1, w_2)

Input Vector $\mathbf{x}$: $(1, x_1, x_2)$

$\mathbf{w} \cdot \mathbf{x}$: $w_0 + w_1x_1 + w_2x_2$

$$h(x_1, x_2) = \begin{cases} 1 \text{ if } w_0 + w_1x_1 + w_2x_2 \geq 0 \\ 0 \text{ otherwise} \end{cases}$$

Weight Vector $\mathbf{w}$: (w_0, w_1, w_2)

Input Vector $\mathbf{x}$: $(1, x_1, x_2)$

$\mathbf{w} \cdot \mathbf{x}$: $w_0 + w_1 x_1 + w_2 x_2$

$$h_{\mathbf{w}}(\mathbf{x}) = \begin{cases} 1 \text{ if } \mathbf{w} \cdot \mathbf{x} \geq 0 \\ 0 \text{ otherwise} \end{cases}$$

perceptron learning rule

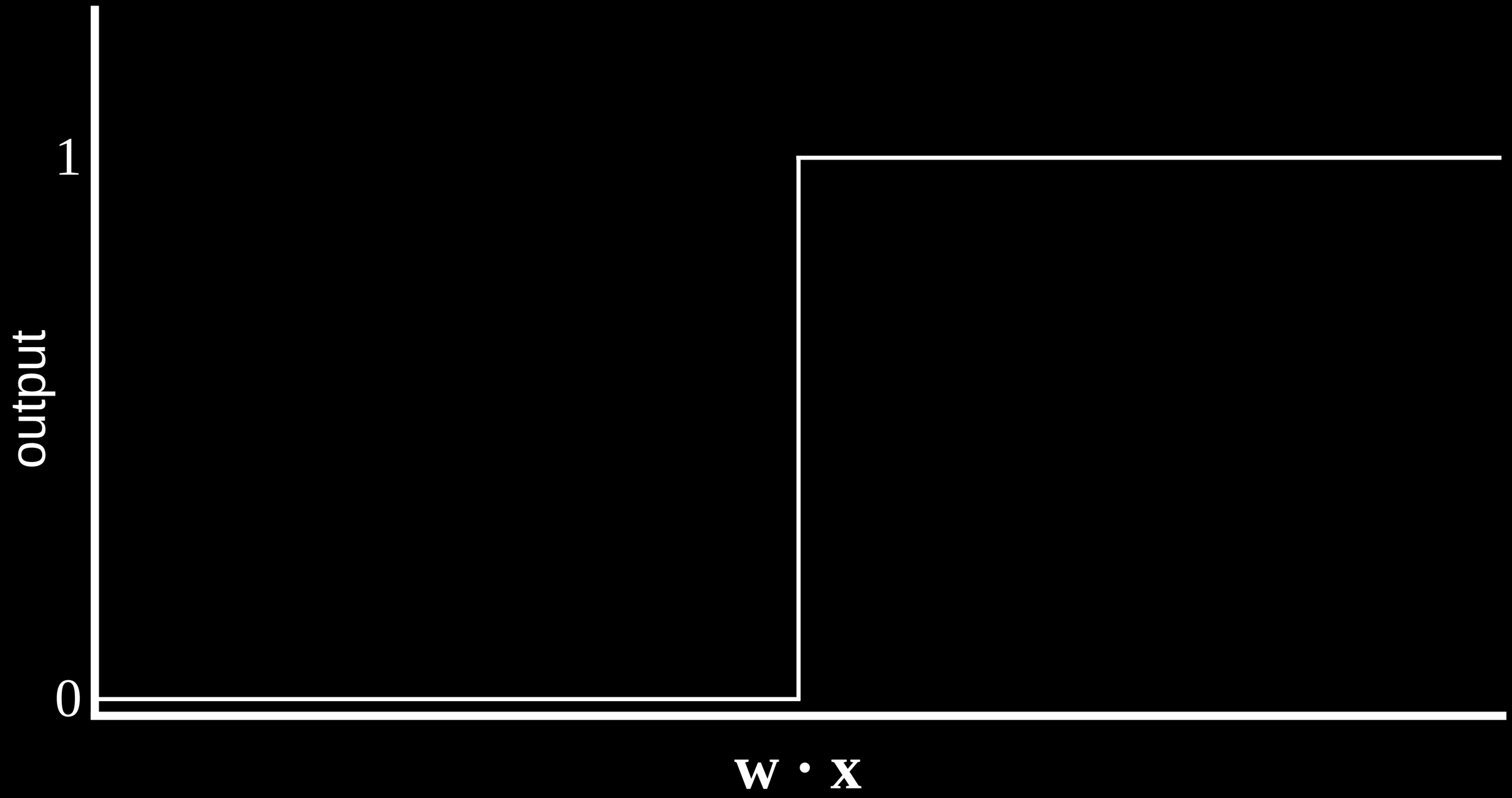
Given data point $(\mathbf{x}, y)$, update each weight according to:

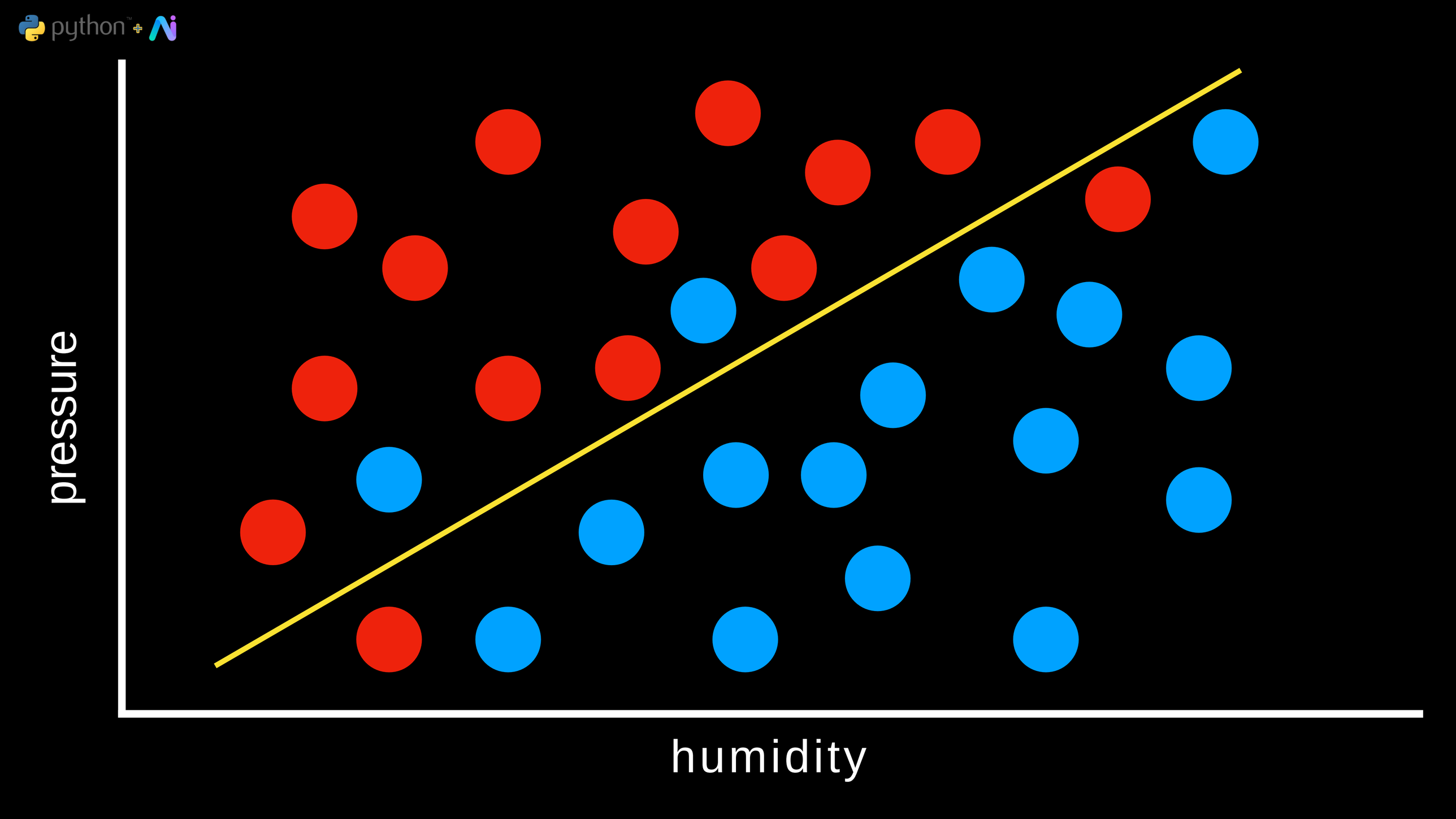
$$w_i = w_i + \alpha(y - h_{\mathbf{w}}(\mathbf{x})) \times x_i$$

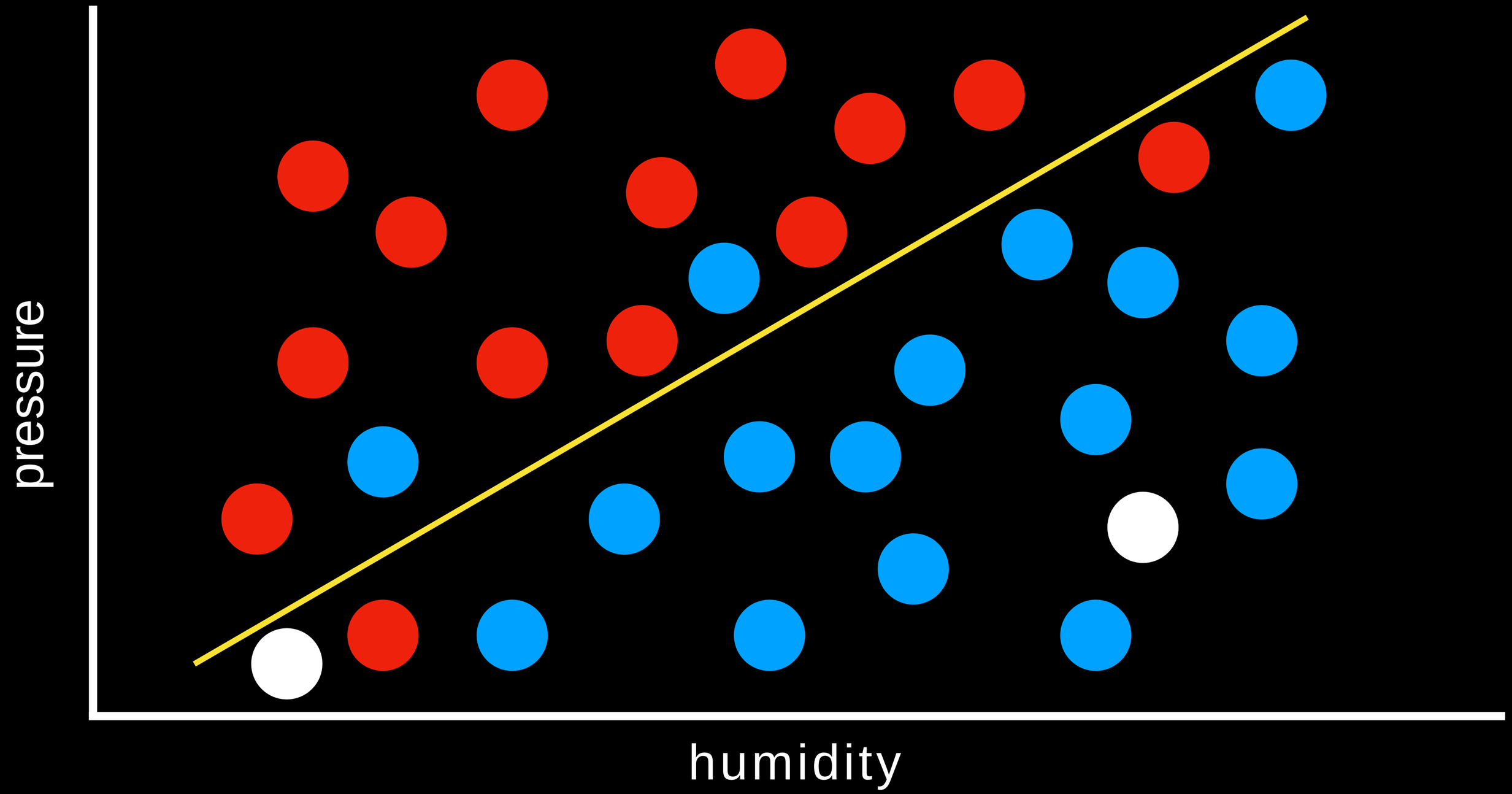
perceptron learning rule

Given data point $(\mathbf{x}, y)$, update each weight according to:

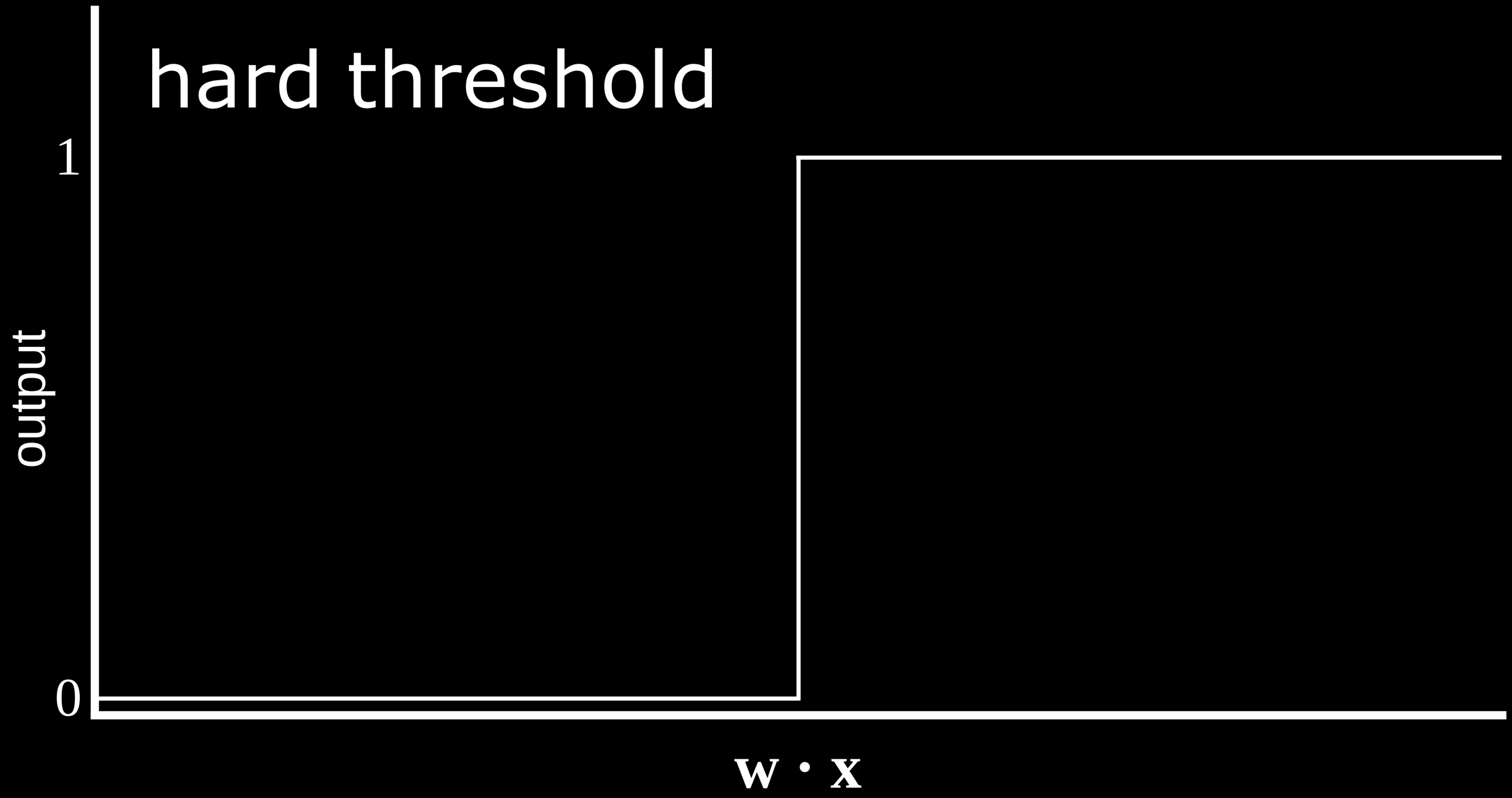
$$w_i = w_i + \alpha(\text{actual value} - \text{estimate}) \times x_i$$

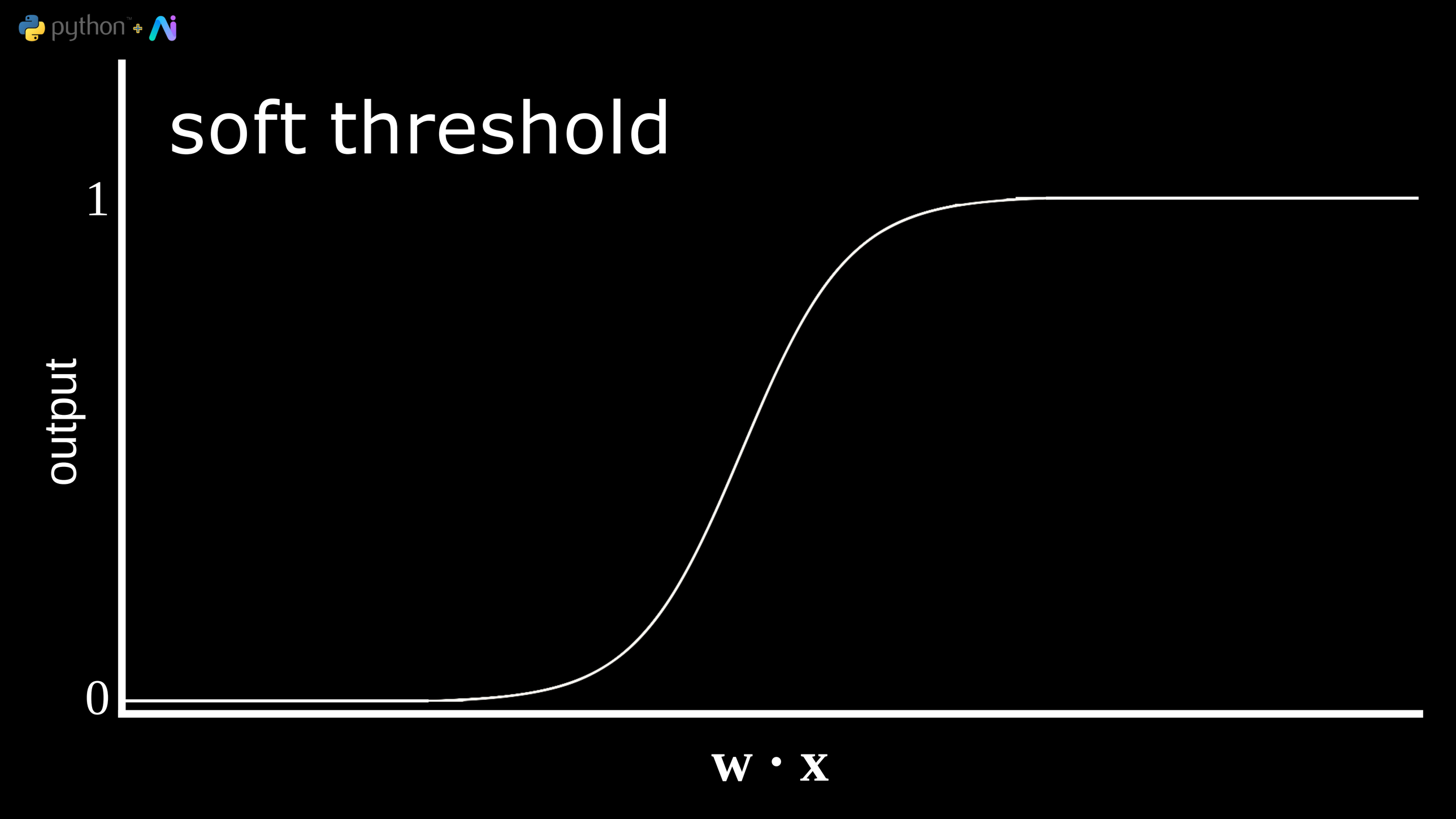

1
0
output
w · x


pressure
humidity

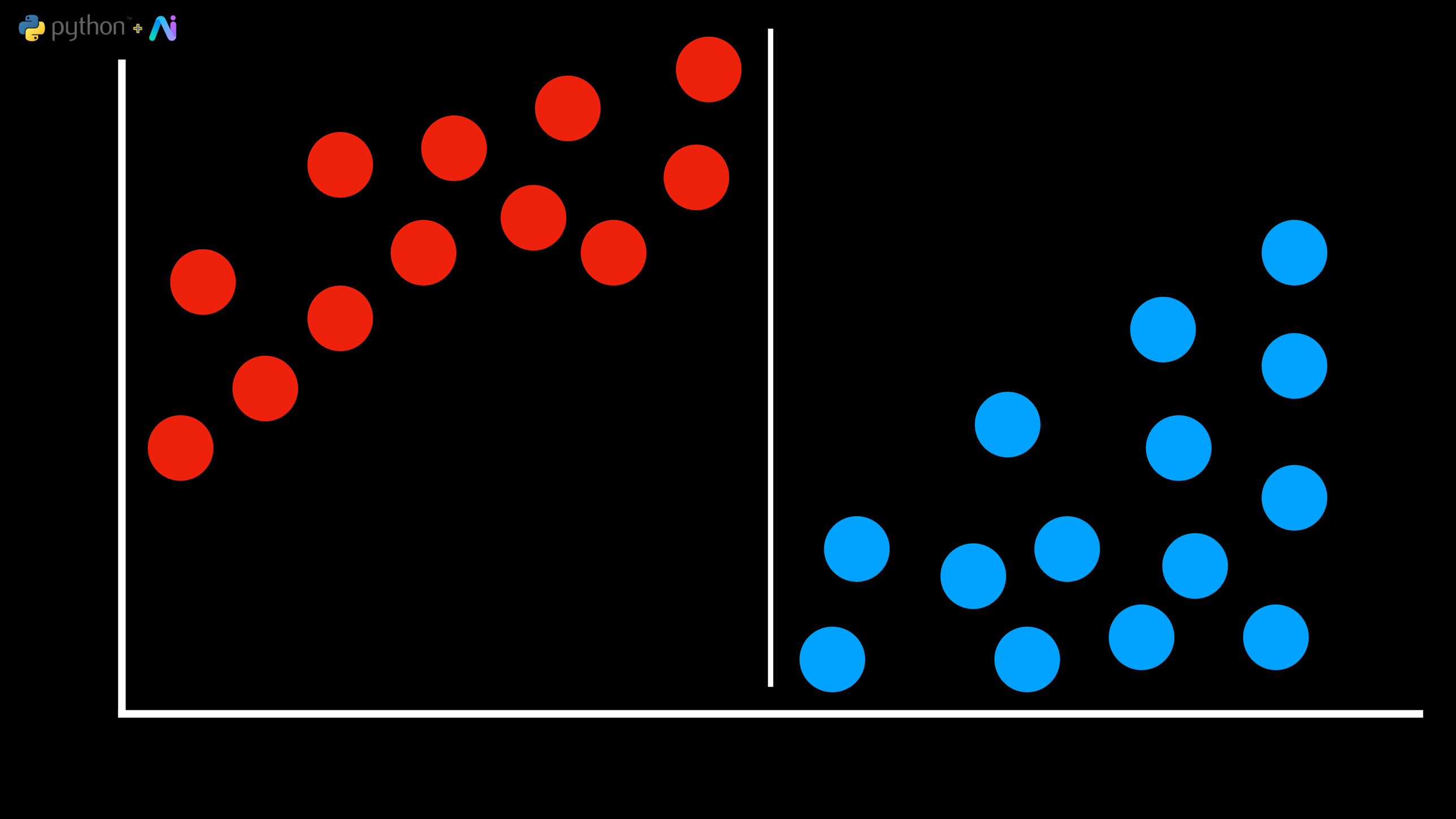

pressure
humidity

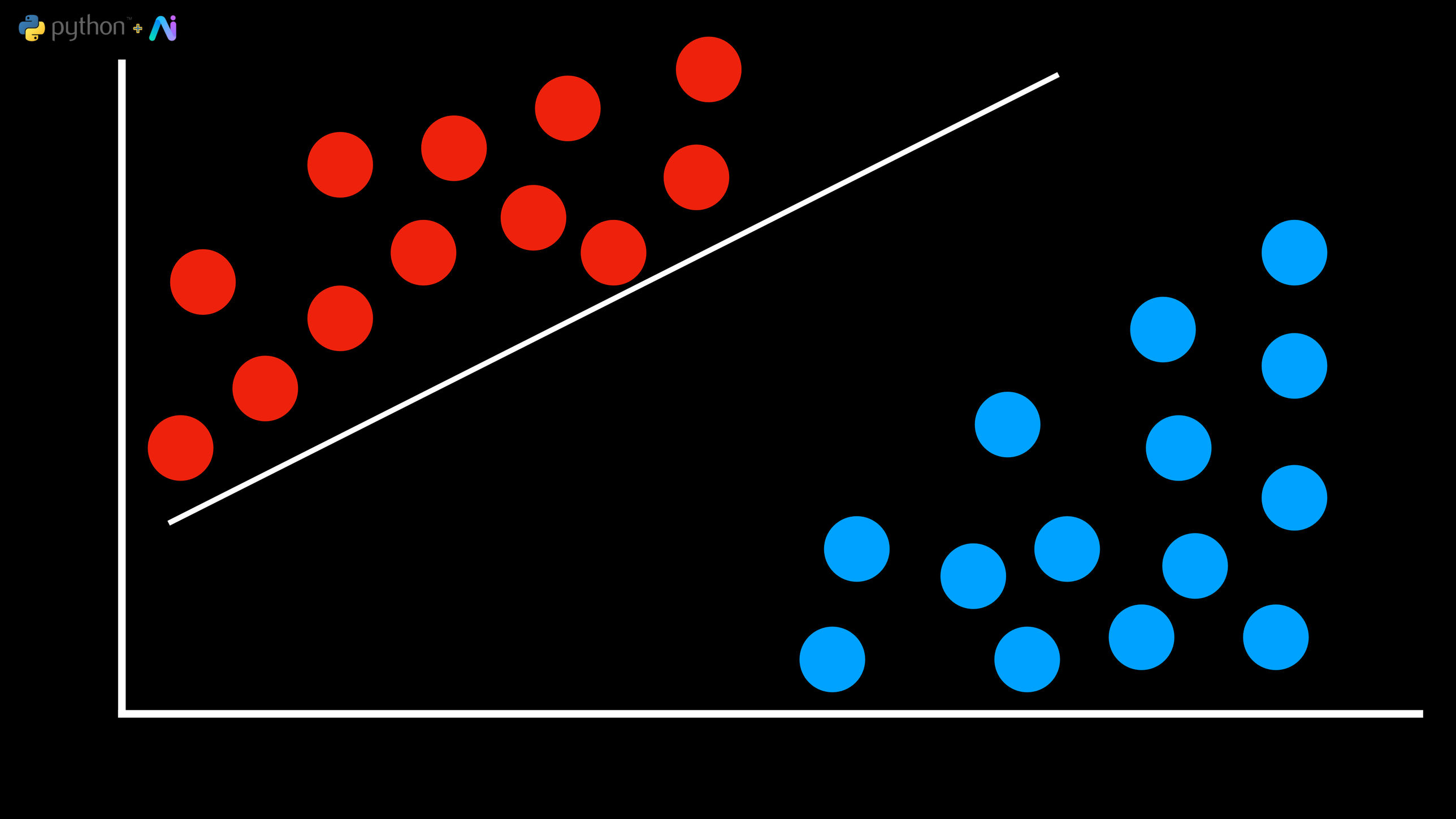
hard threshold

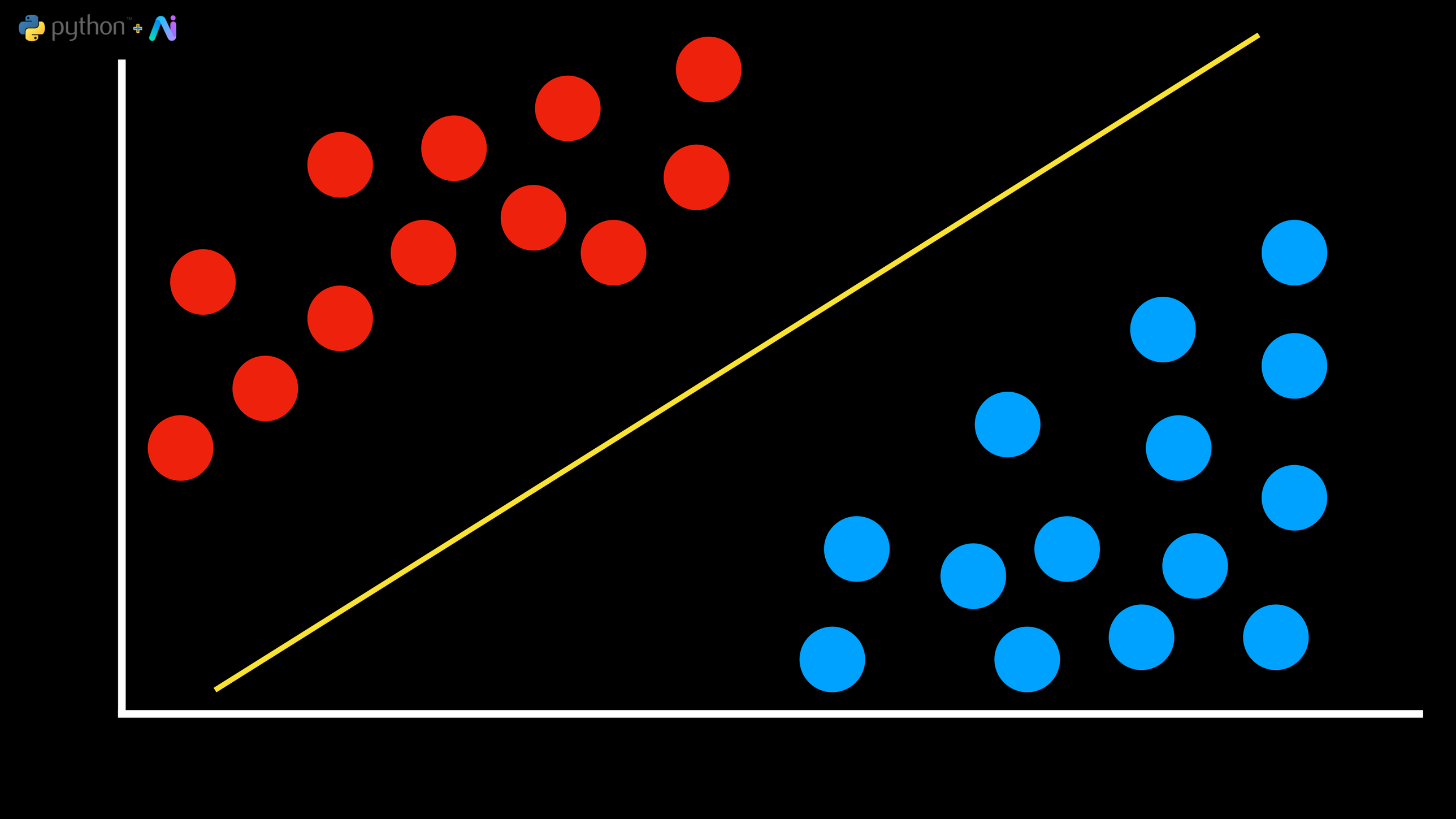



soft threshold
1
0
output
w · x

Support Vector Machines

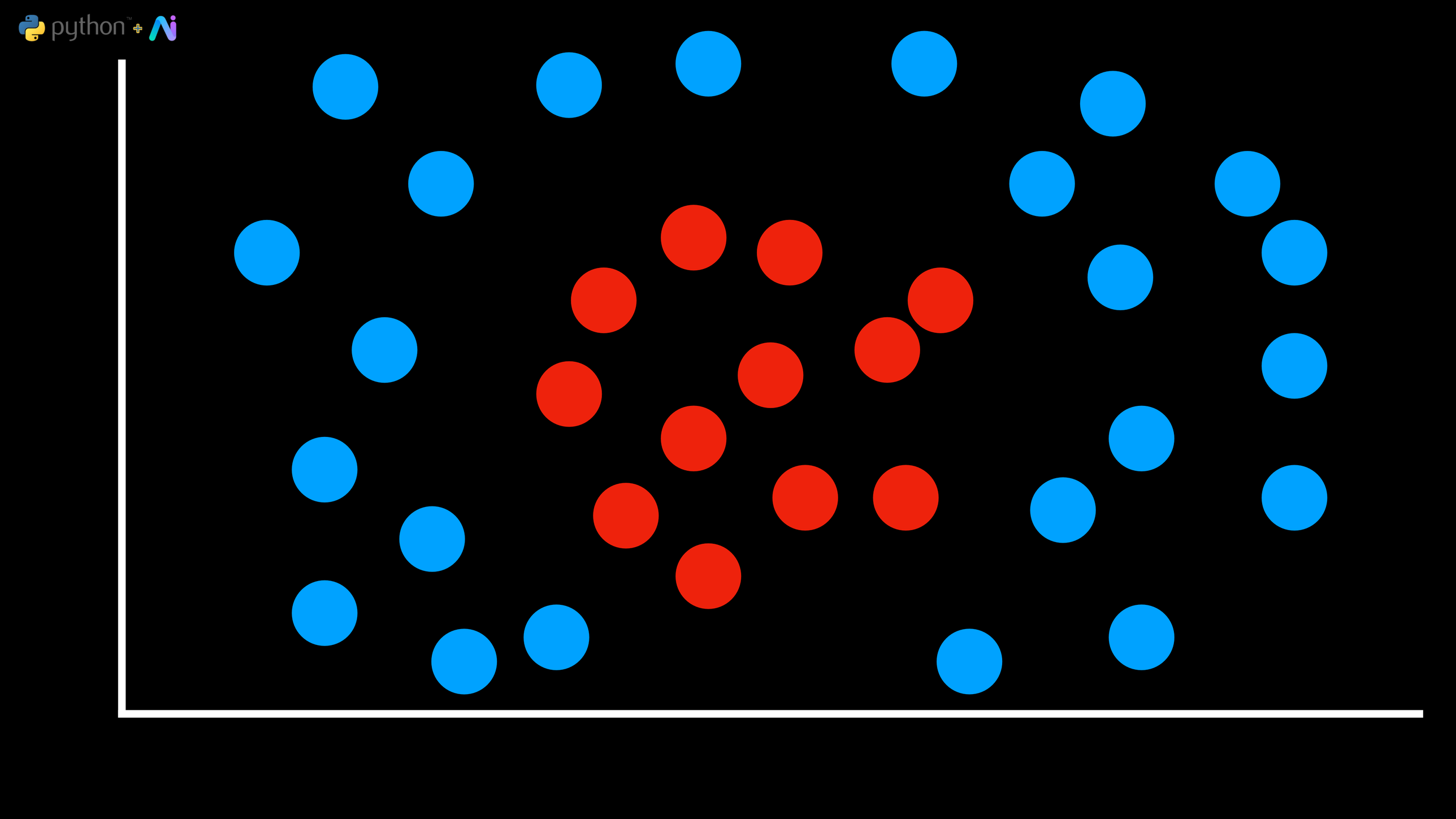


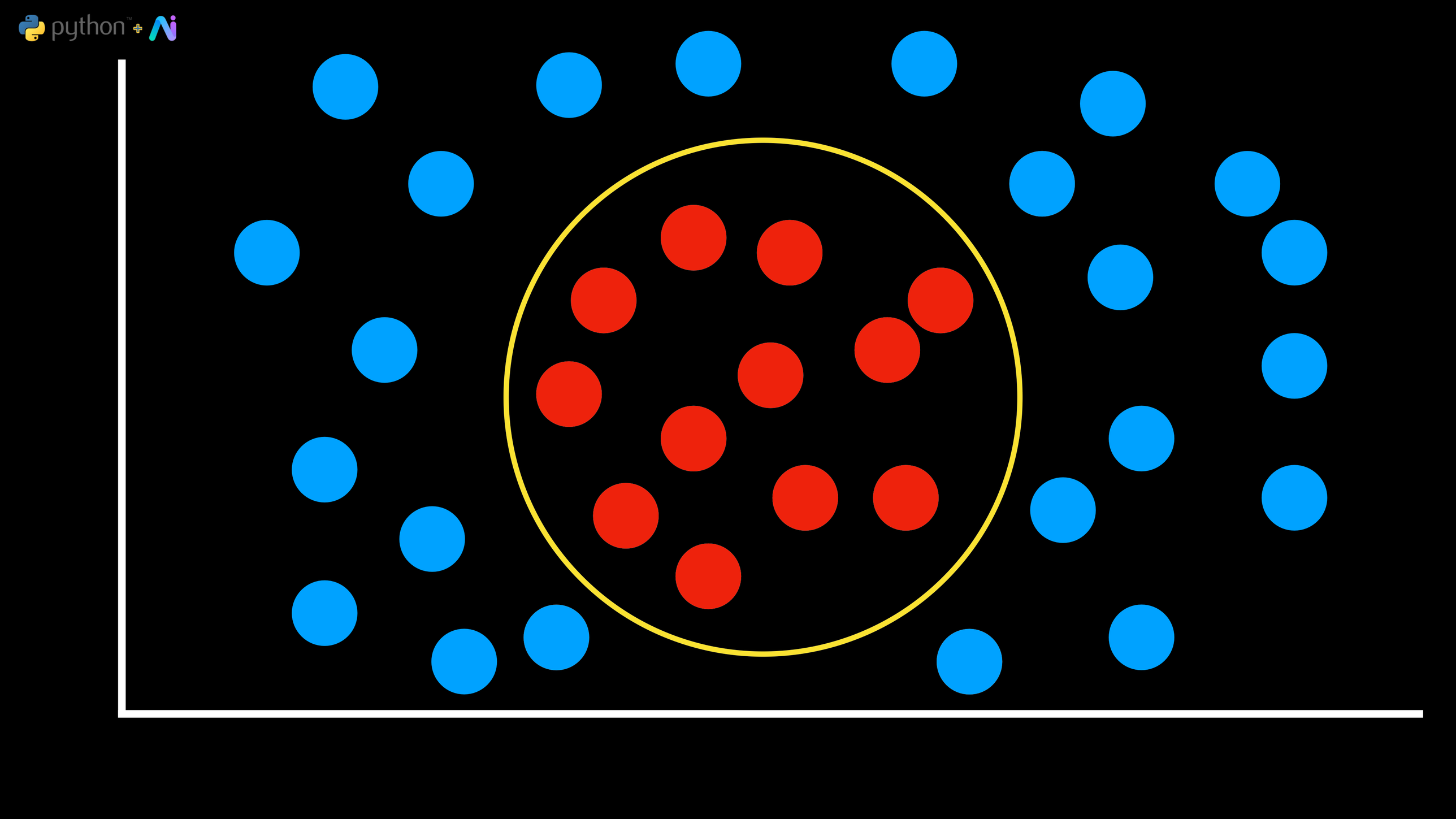




maximum margin separator

boundary that maximizes the distance between any of the data points





regression

supervised learning task of learning a function mapping an input point to a continuous value

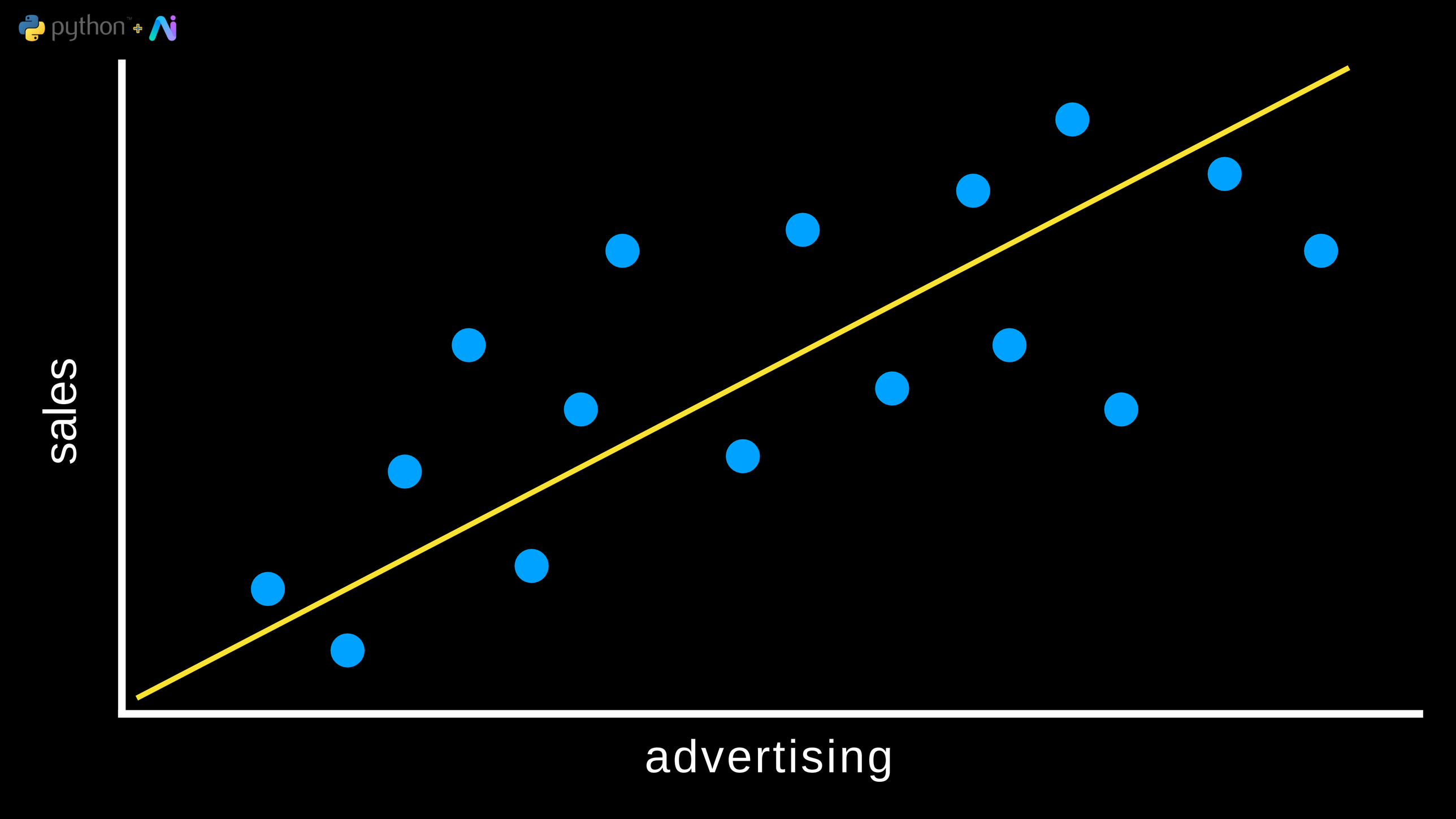
$$f(advertising)$$

$$f(1200) = 5800$$

$$f(2800) = 13400$$

$$f(1800) = 8400$$

$$h(advertising)$$


sales
advertising

Evaluating Hypotheses

loss function

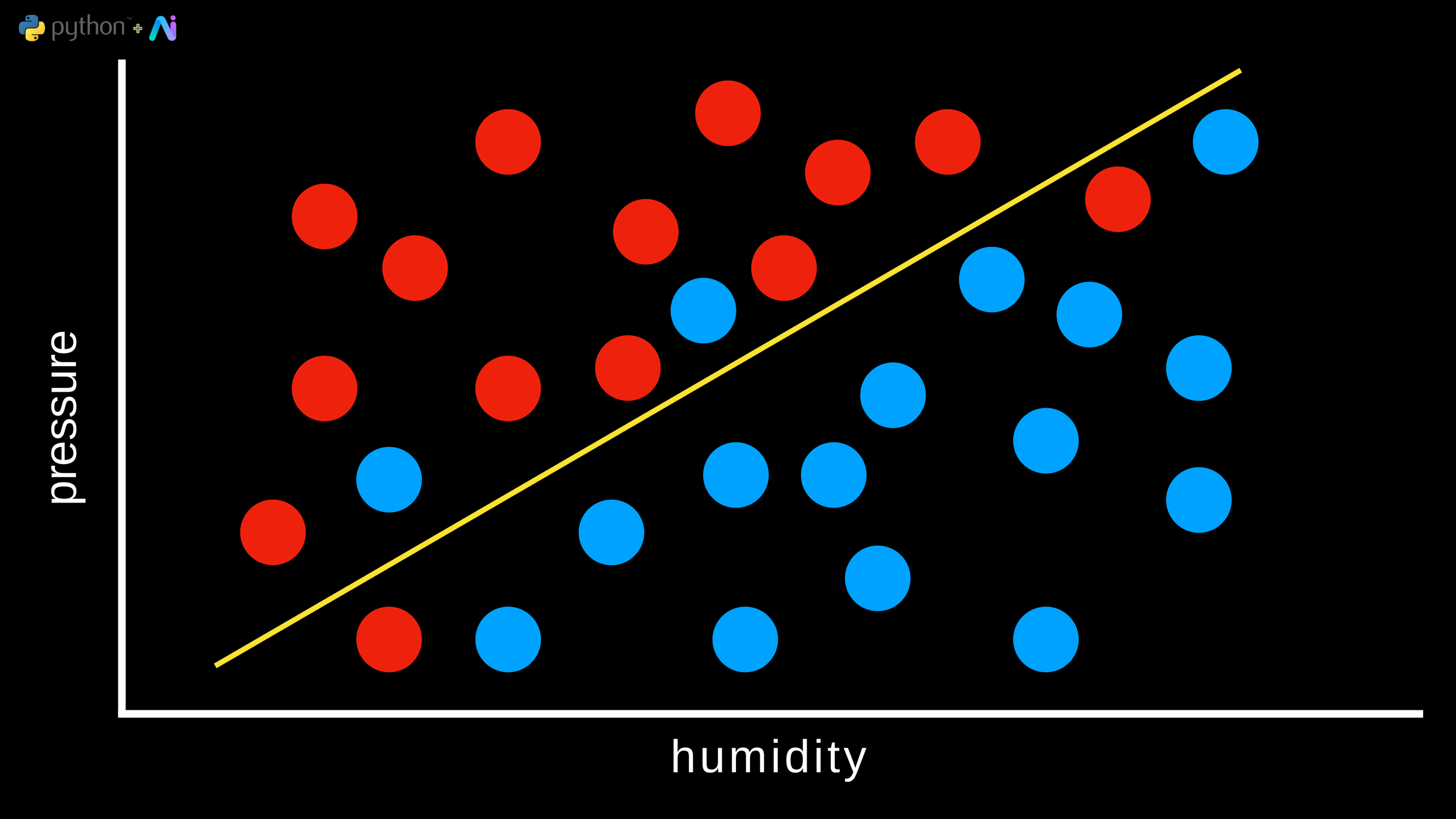
function that expresses how poorly our hypothesis performs

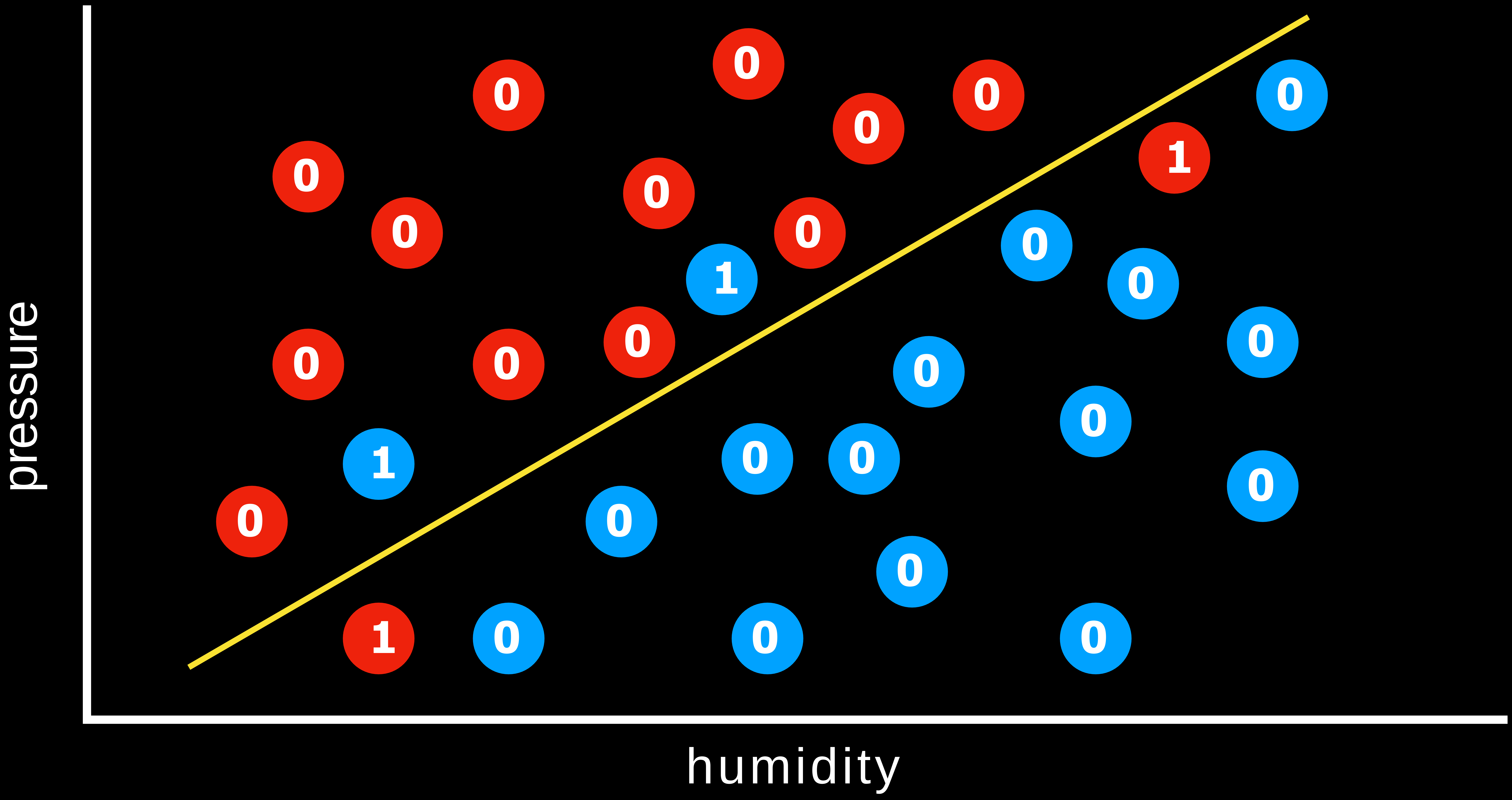
0-1 loss function

$L(\text{actual}, \text{predicted}) =$

0 if $\text{actual} = \text{predicted}$,

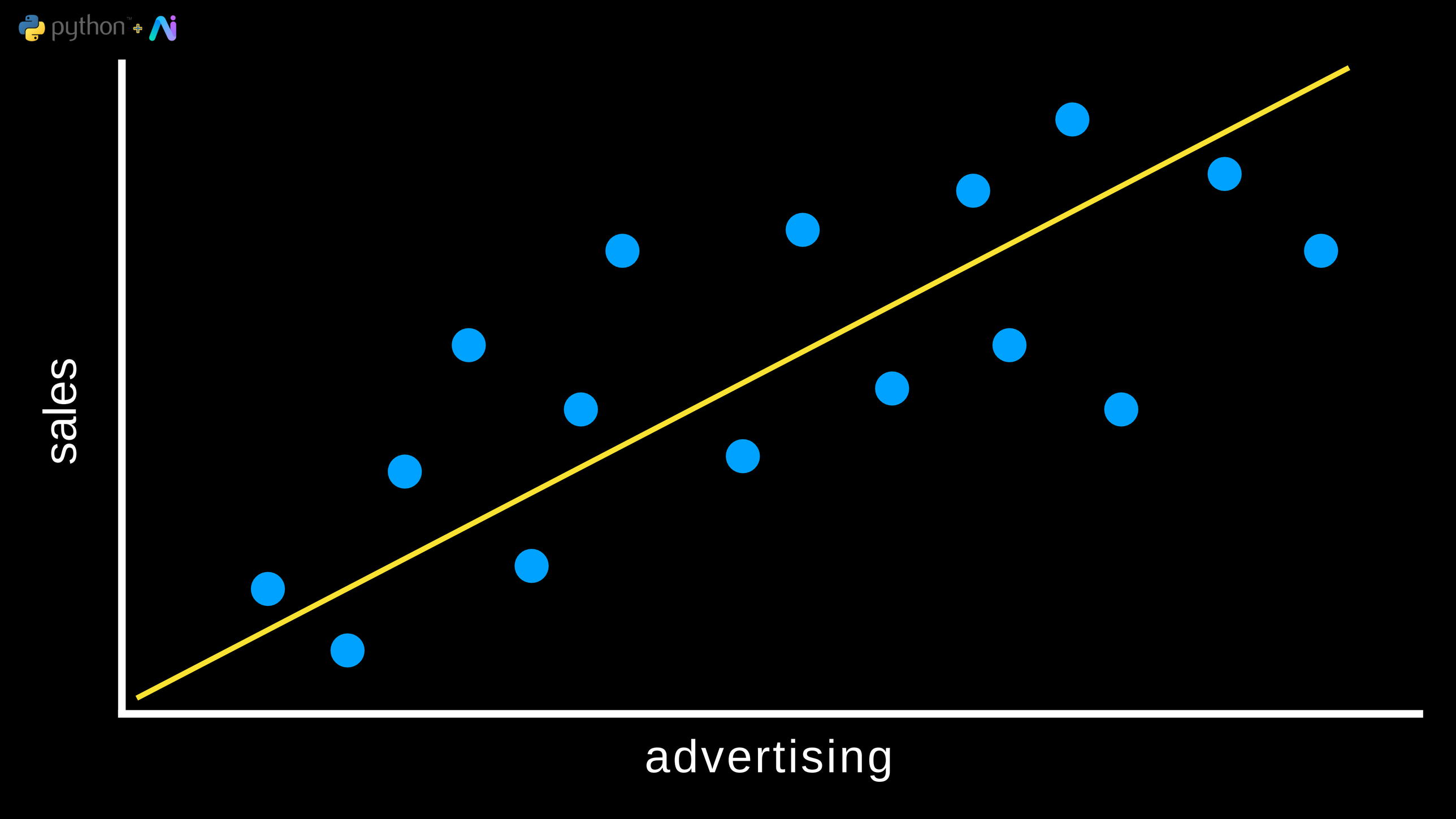
1 otherwise

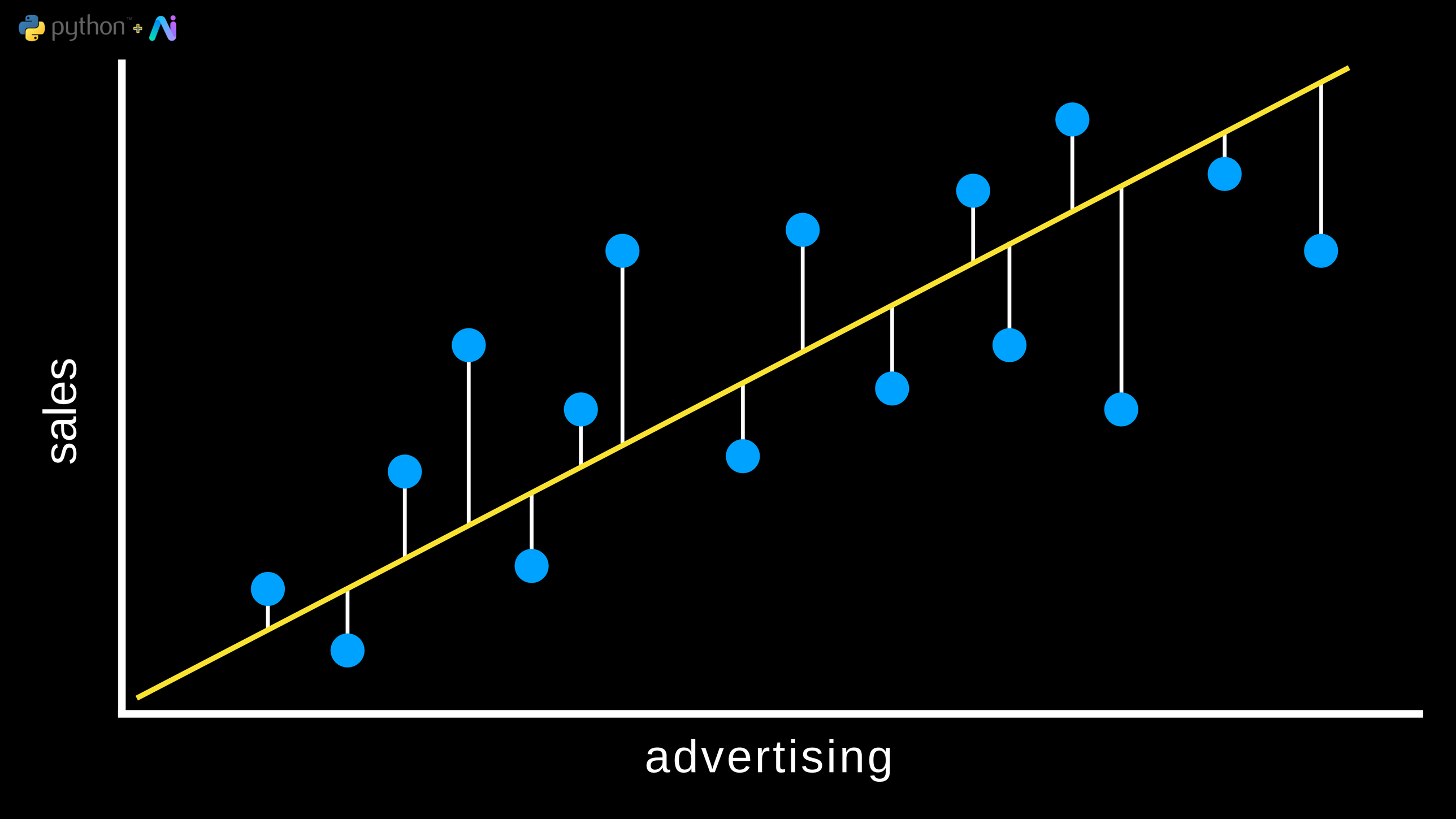

pressure
humidity


pressure
humidity

L_1 loss function

$$L(\text{actual}, \text{predicted}) = |\ \text{actual} - \text{predicted}\ |$$


sales
advertising

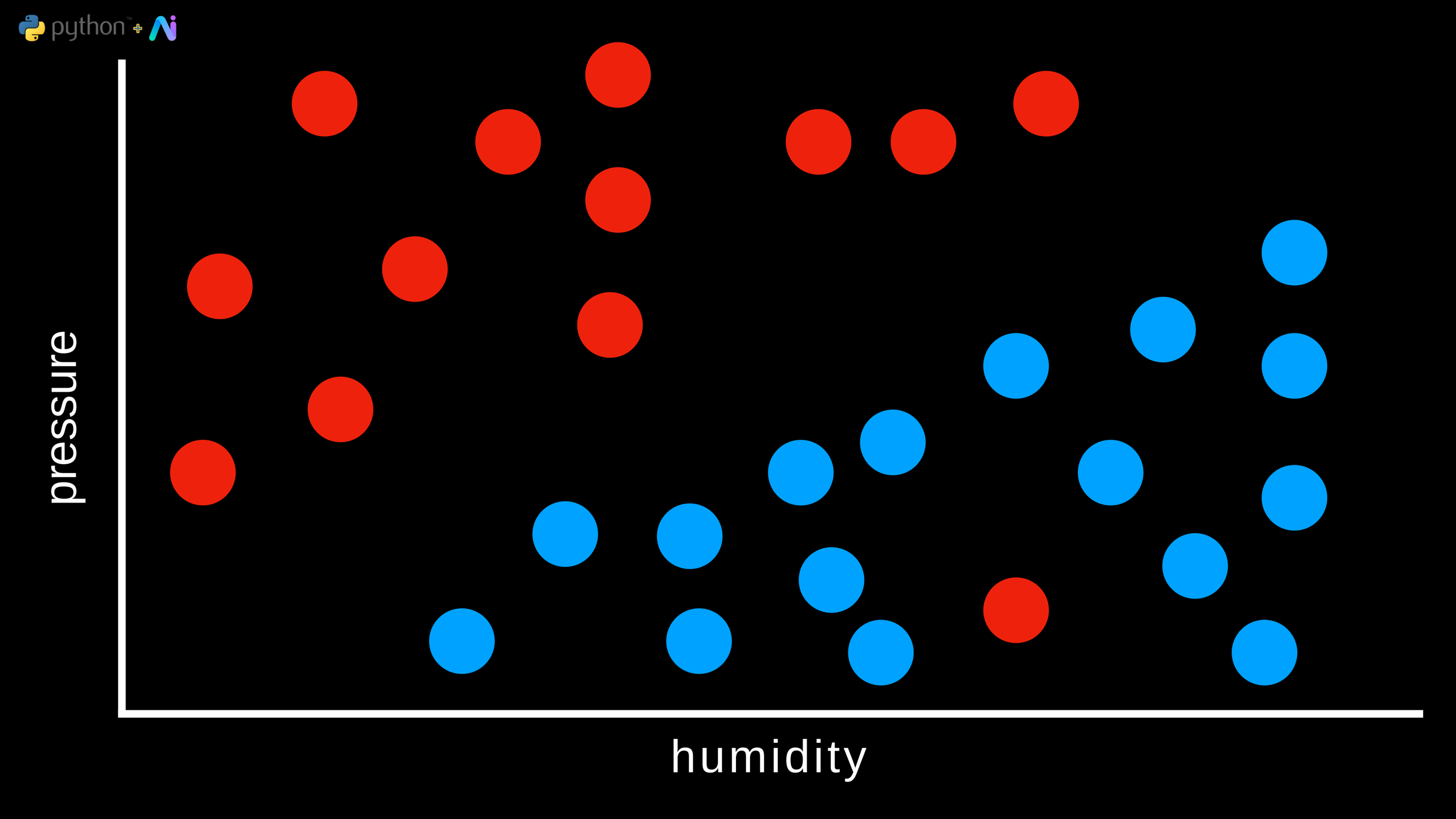

sales
advertising

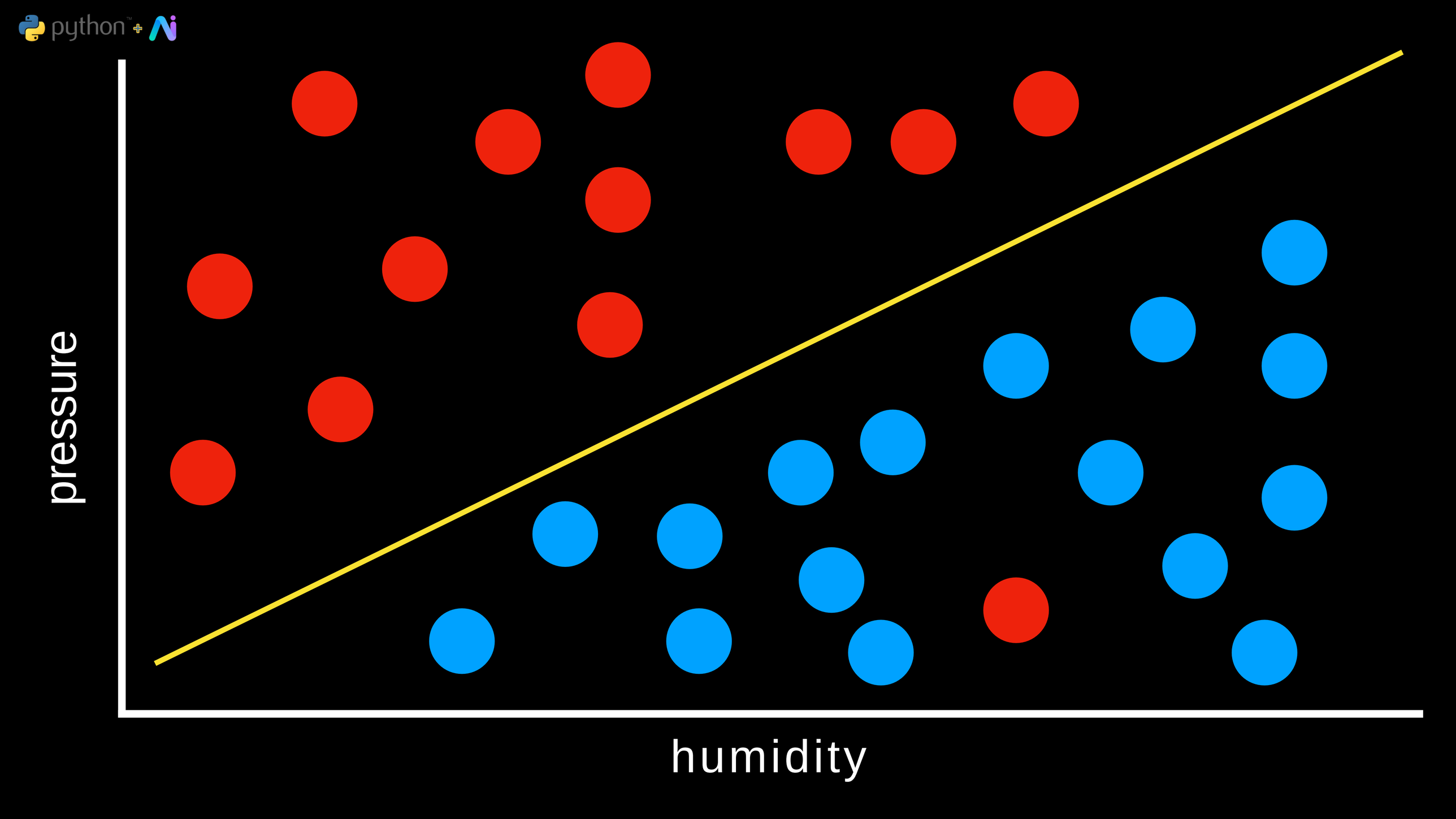
L_2 loss function

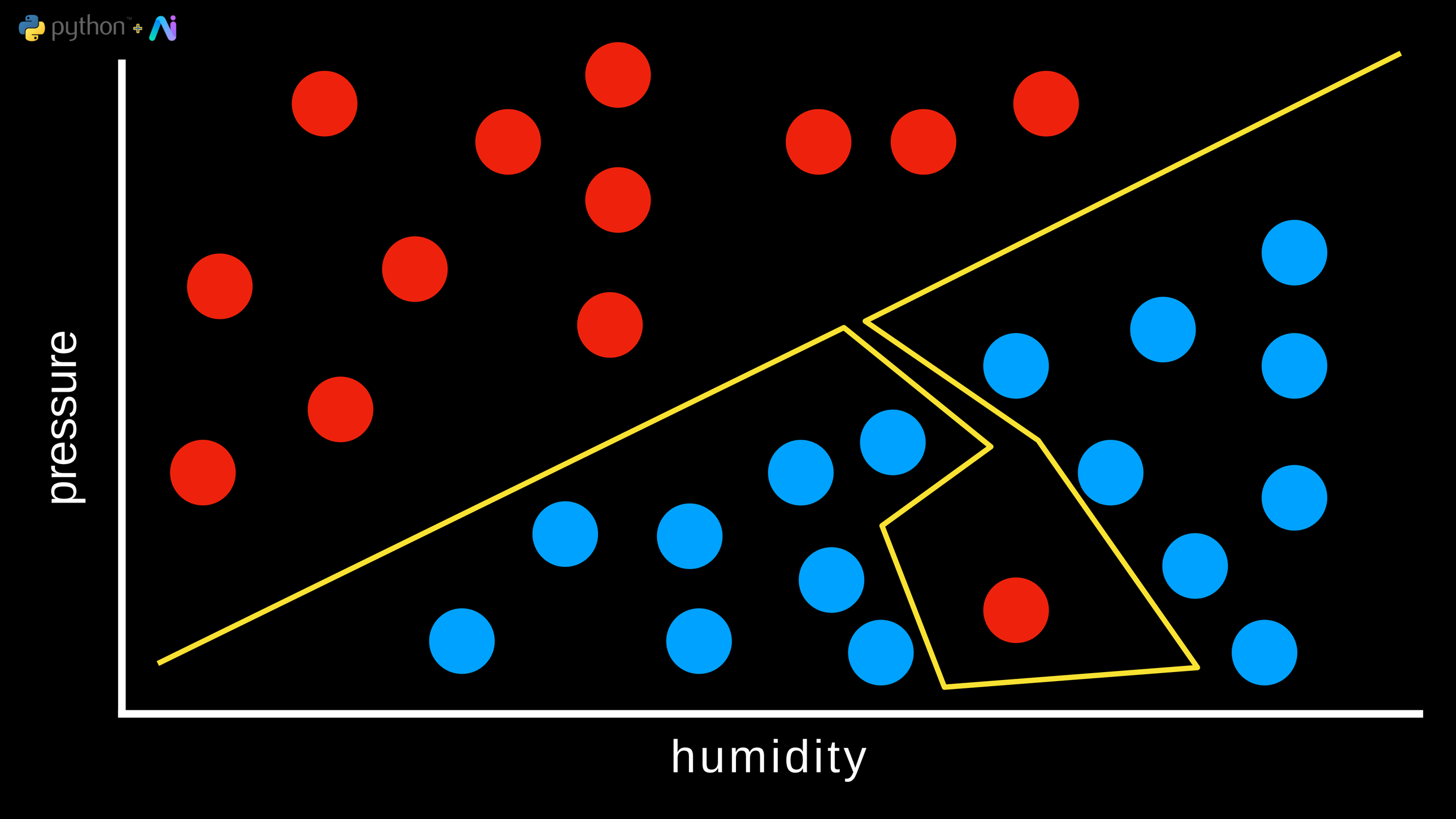
$$L(\text{actual}, \text{predicted}) = (\text{actual} - \text{predicted})^2$$

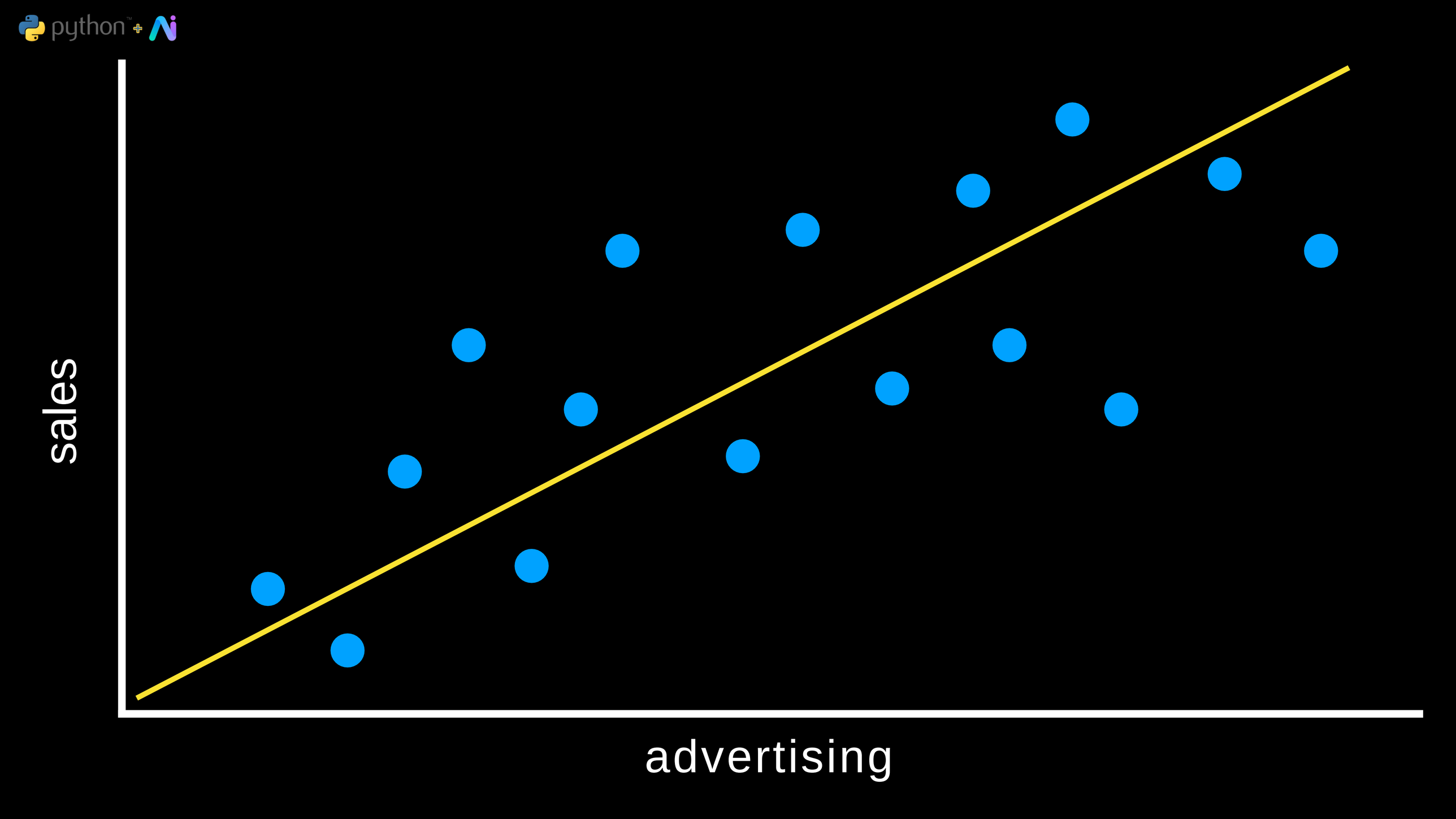
overfitting

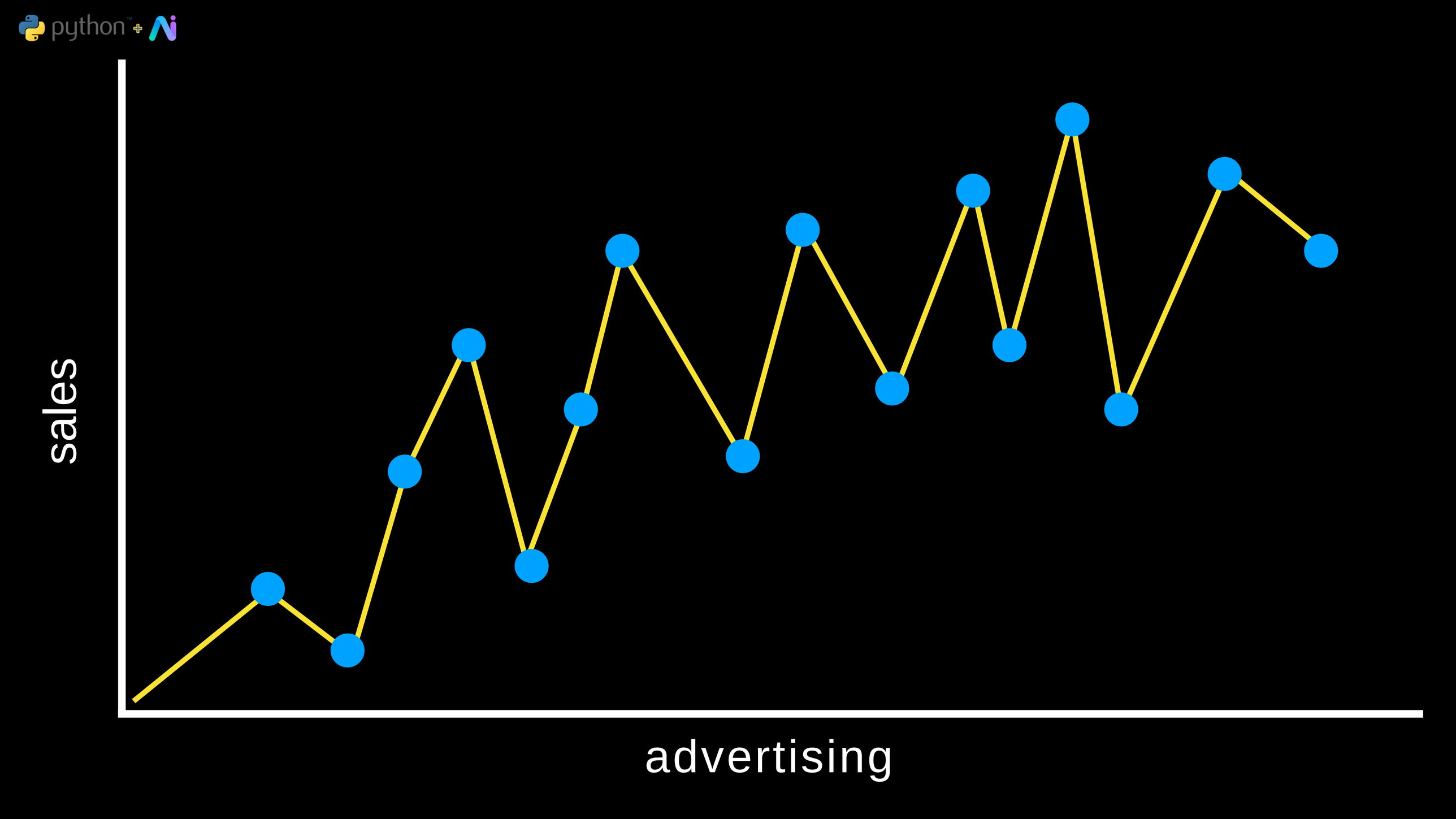
a model that fits too closely to a particular data set and therefore may fail to generalize to future data


pressure
humidity


pressure
humidity


pressure
humidity


sales
advertising


sales
advertising

$$cost(h) = loss(h)$$

$$cost(h) = loss(h) + complexity(h)$$

$$cost(h) = loss(h) + \lambda complexity(h)$$

regularization

penalizing hypotheses that are more complex to favor simpler, more general hypotheses

$$cost(\mathrm{h}) = loss(\mathrm{h}) + \lambda complexity(\mathrm{h})$$

holdout cross-validation

splitting data into a **training set** and a **test set**, such that learning happens on the training set and is evaluated on the test set

k-fold cross-validation

splitting data into k sets, and experimenting k times, using each set as a test set once, and using remaining data as training set

scikit-learn

Learning

- Supervised Learning
- Reinforcement Learning
- Unsupervised Learning

Supervised Learning

python + AI

Introduction to Artificial Intelligence with Python