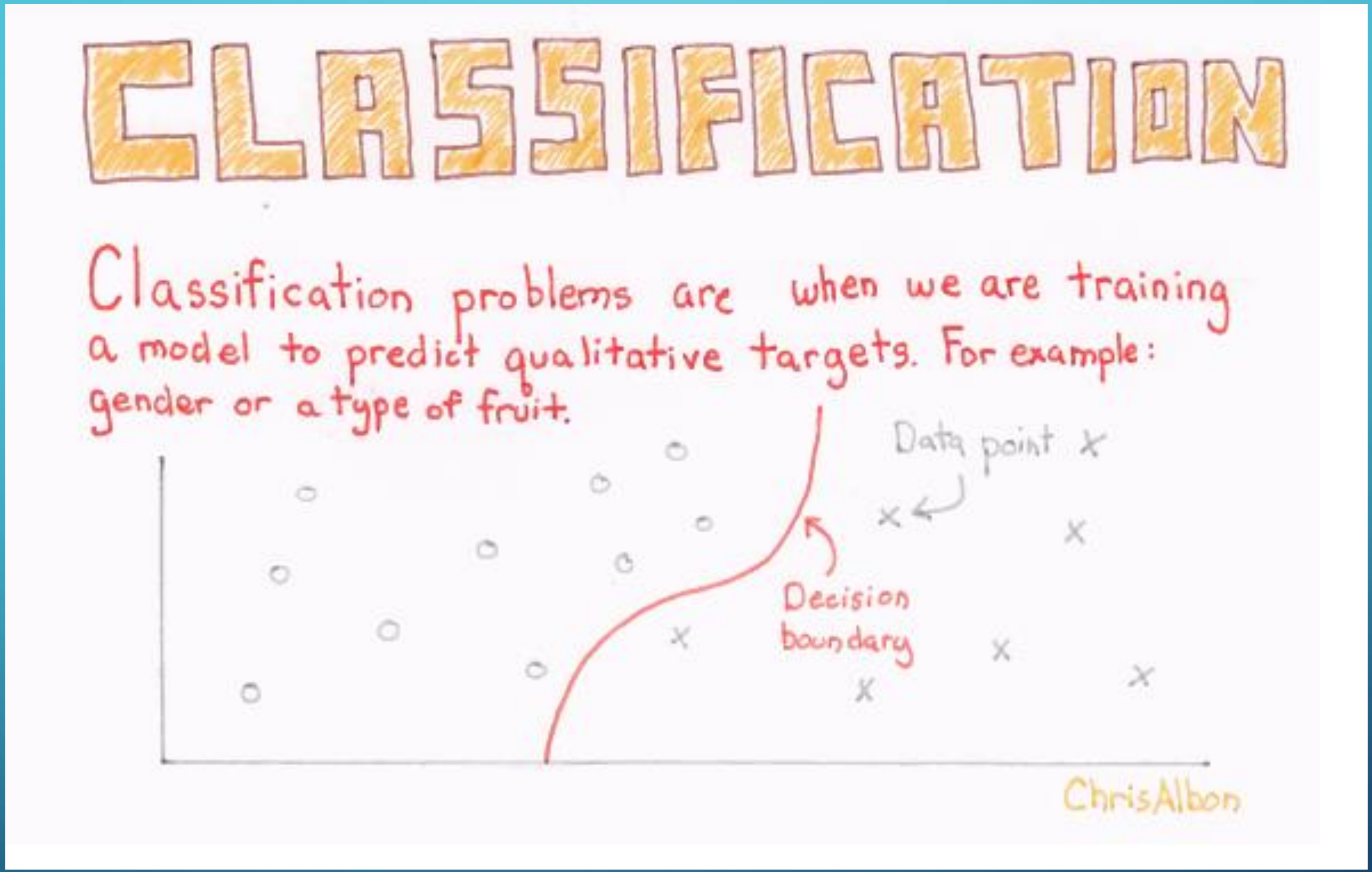
LECTURE 4

AGENDA

- Revisiting classification
- Logistic Regression
- K-Nearest Neighbours
- Naïve Bayes
- Decision Trees
- Random Forest
- Support Vectors

REVISITING CLASSIFICATION


CLASSIFICATION
Classification problems are when we are training a model to predict qualitative targets. For example: gender or a type of fruit.
Data point
Decision boundary
ChrisAlbon

Classification

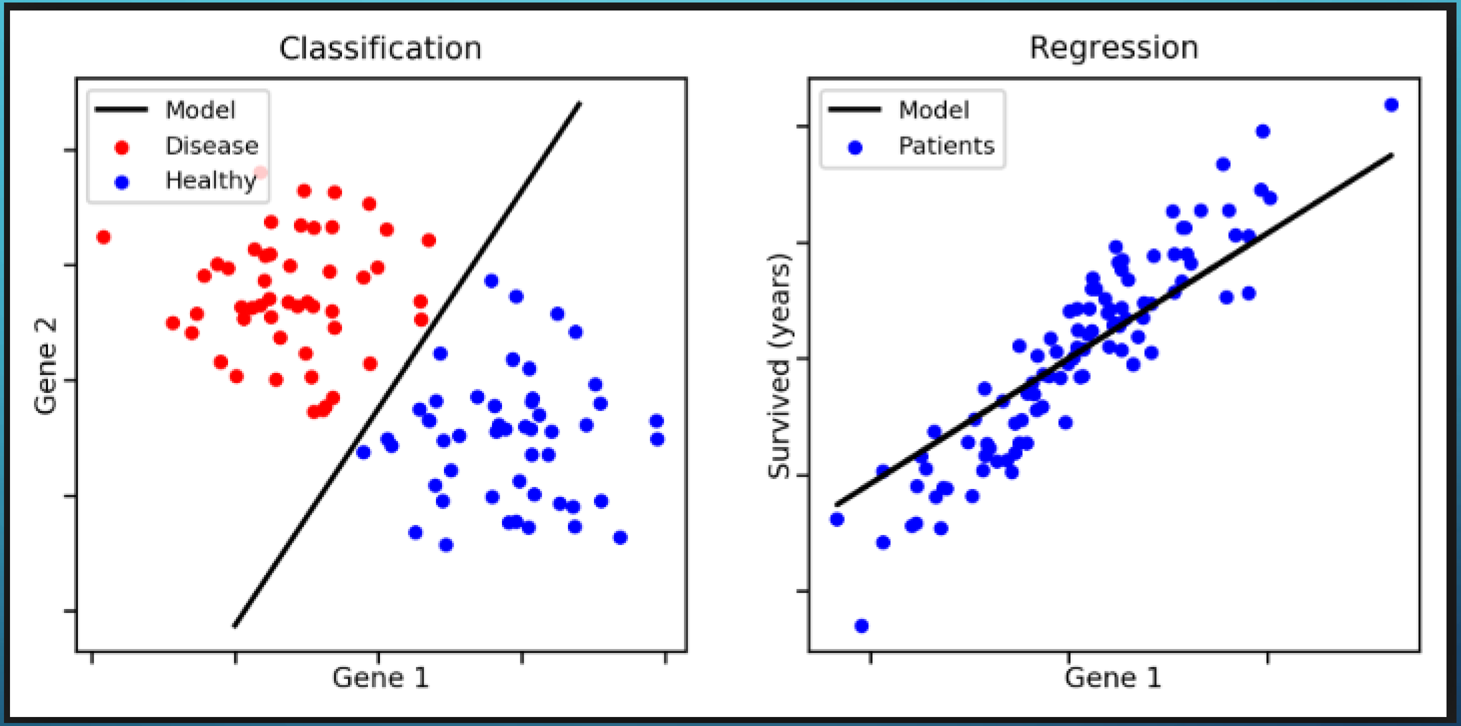
Email: Spam / Not Spam?
Online Transactions: Fraudulent (Yes / No)?
Tumor: Malignant / Benign ?

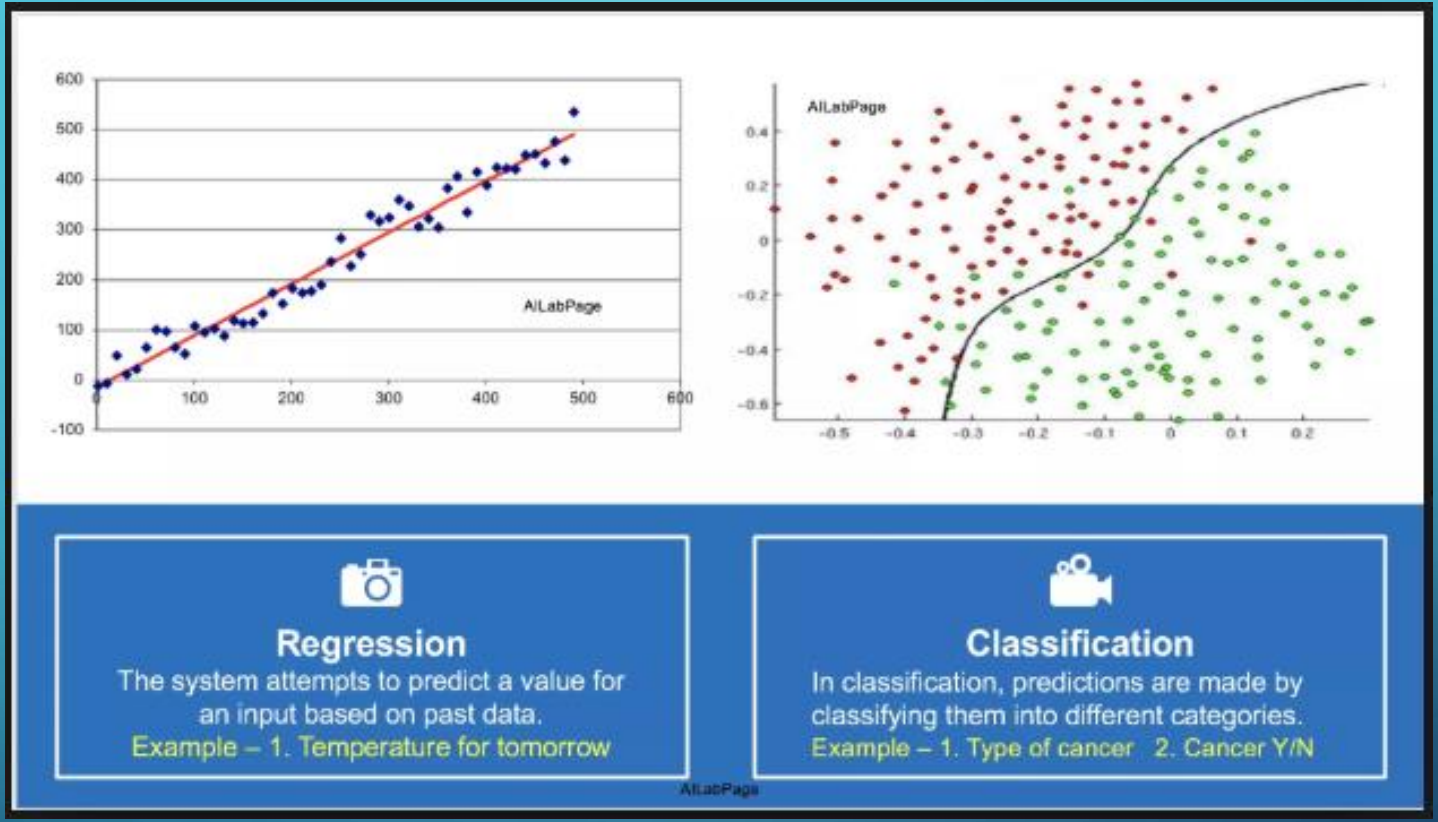
$y \in \{0, 1\}$

0: “Negative Class” (e.g., benign tumor)
1: “Positive Class” (e.g., malignant tumor)


(Yes) 1
Malignant ?
(No) 0
Tumor Size

CLASSIFICATION VS REGRESSION




AILabPage
AILabPage
Regression
The system attempts to predict a value for an input based on past data.
Example – 1. Temperature for tomorrow
Classification
In classification, predictions are made by classifying them into different categories.
Example – 1. Type of cancer 2. Cancer Y/N
AILabPage

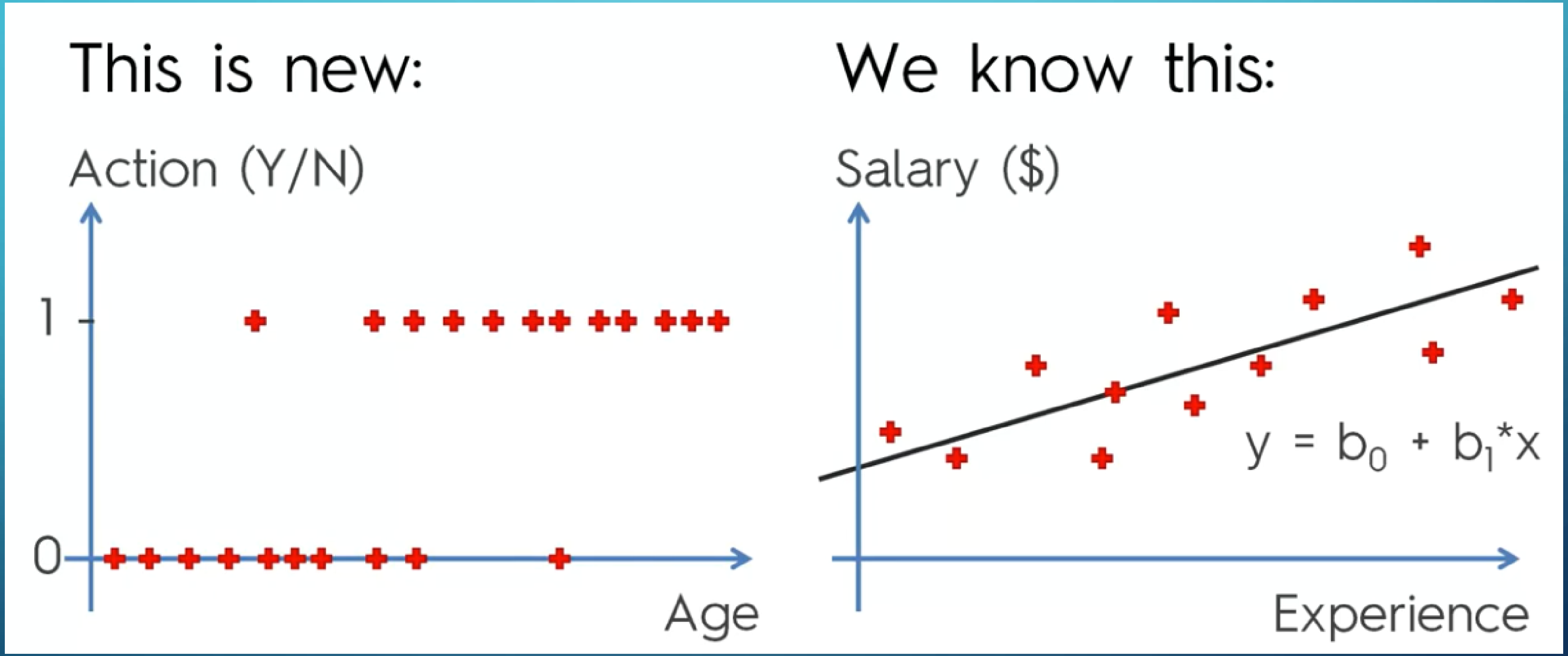
KEY DIFFERENCES

Classification	Regression
A model where the target variable can take a discrete set of values	A model where the target variable can take continuous values typically real numbers
The dependent variables are categorical	The dependent variables are numerical
Decision Boundary	Curves following the trend
Ex: Predicting whether an email is spam or not, predicting whether the credit card transaction is fraud or not, predicting whether a customer will take a loan or not	Ex: Predicting GDP of a country, predicting the product price, predicting product price, predicting the house selling price, score prediction

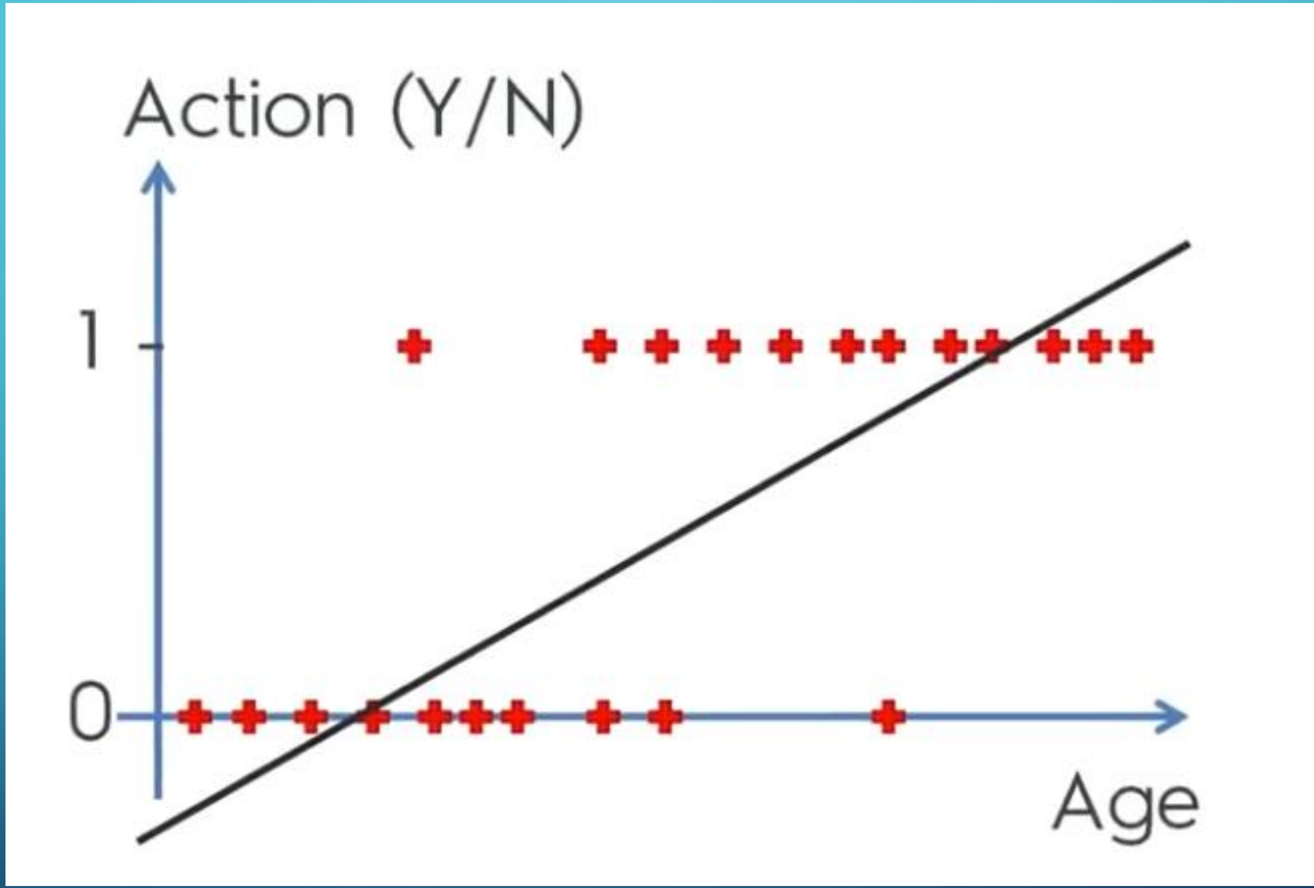
LOGISTIC REGRESSION

- Effective classification algorithm
- Similar to Linear Regression
- Building block of Neural Networks

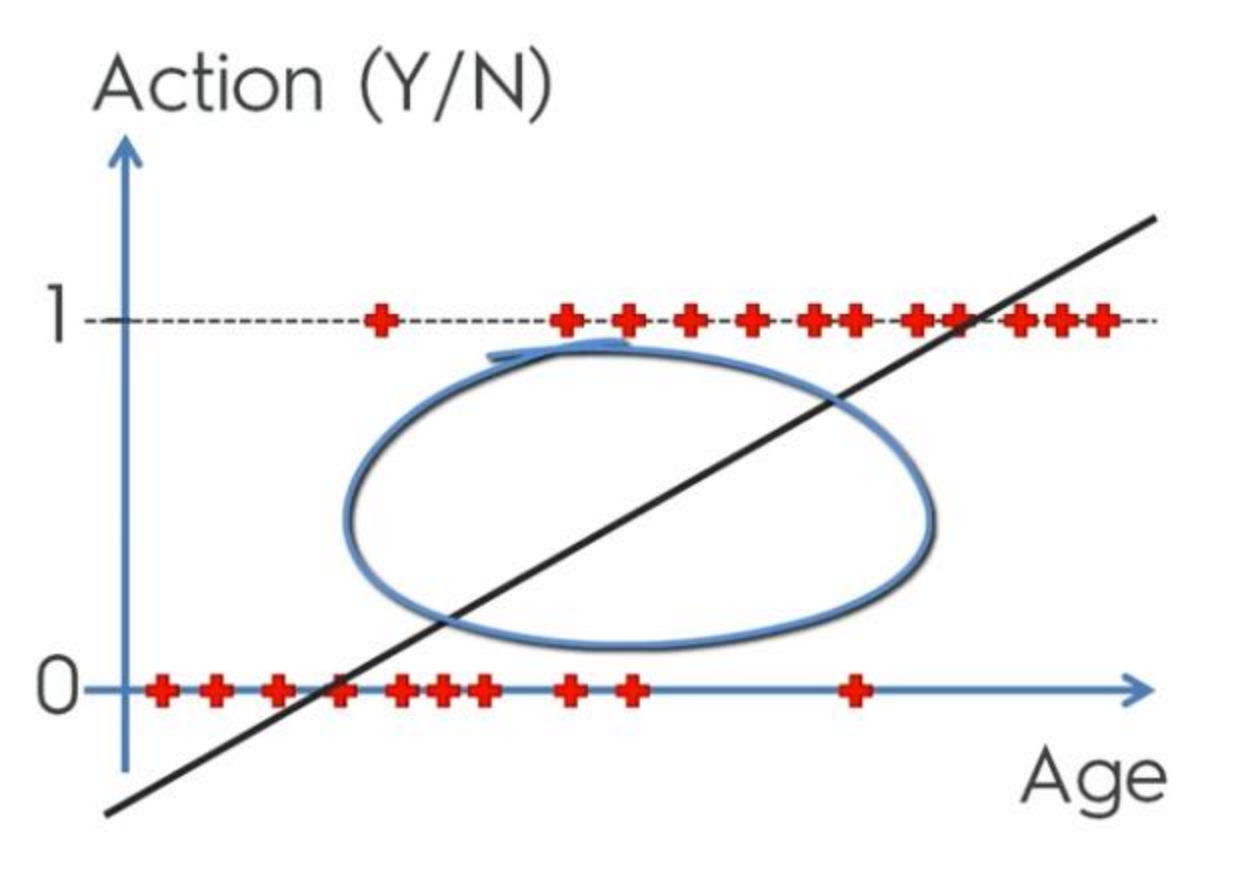
NOTE: Name has ‘Regression’ but it is still a classification algorithm


This is new:
Action (Y/N)
1
0
Age
We know this:
Salary ($)
$y = b_0 + b_1*x$
Experience

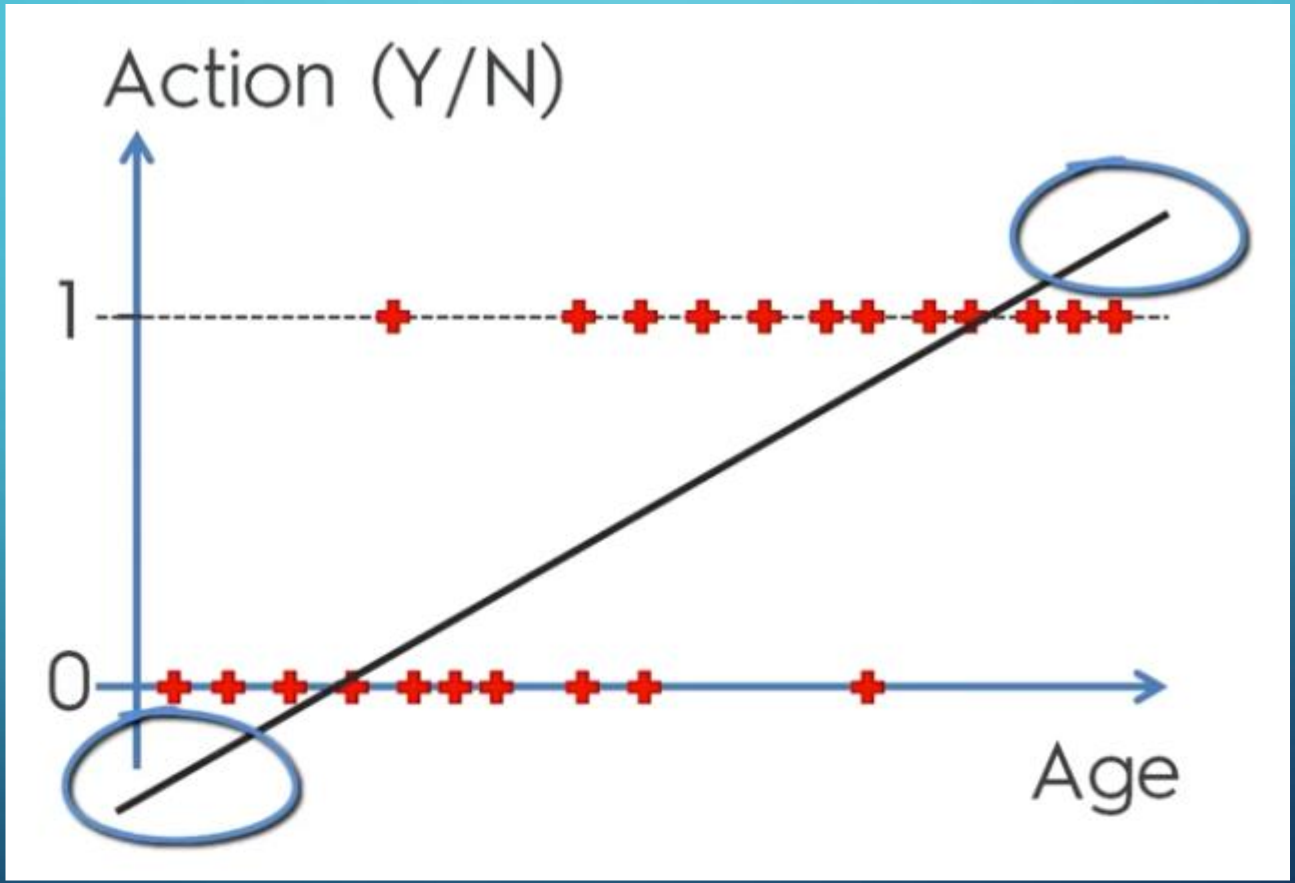
AS PER REGRESSION:

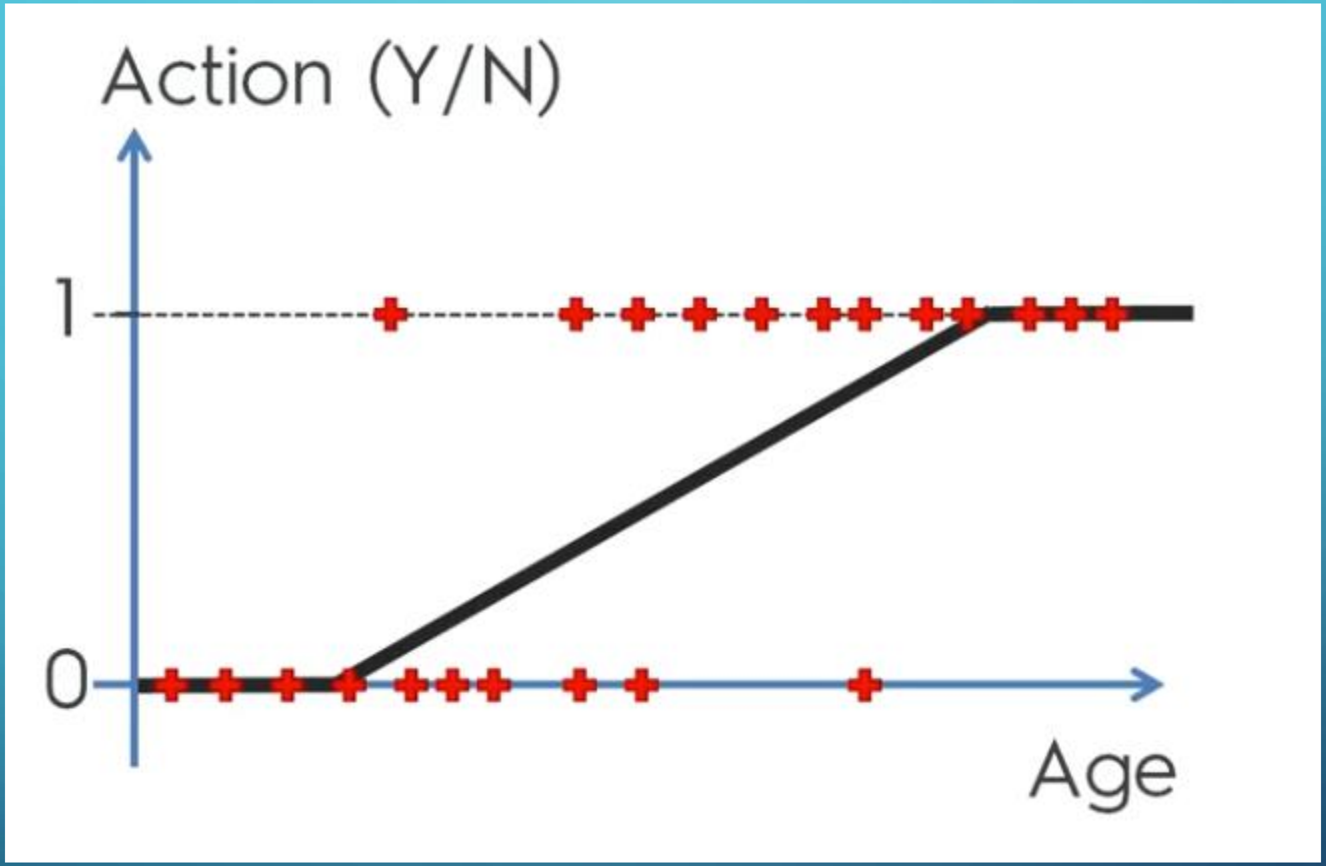


IMPORTANT SECTION

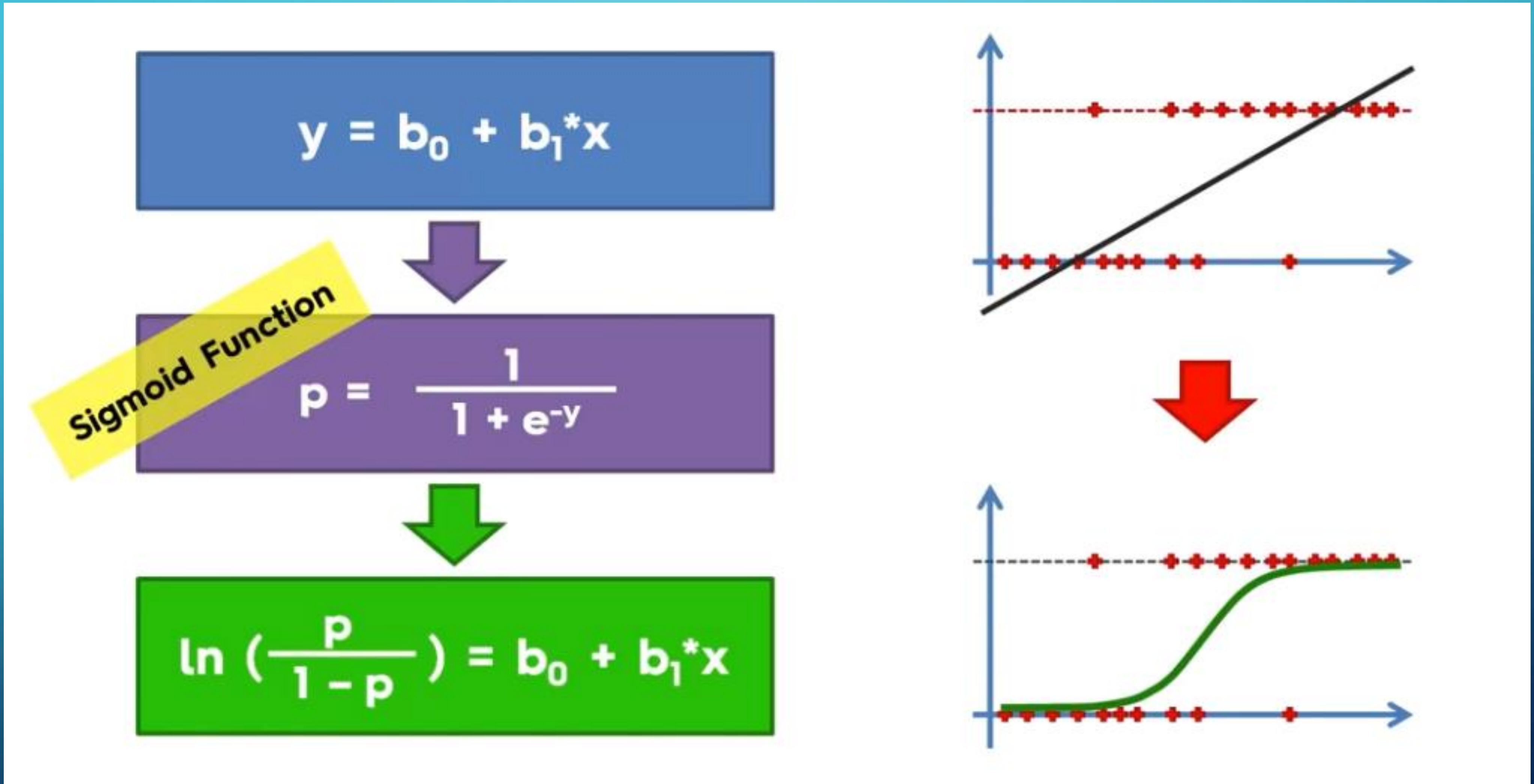


INSIGNIFICANT





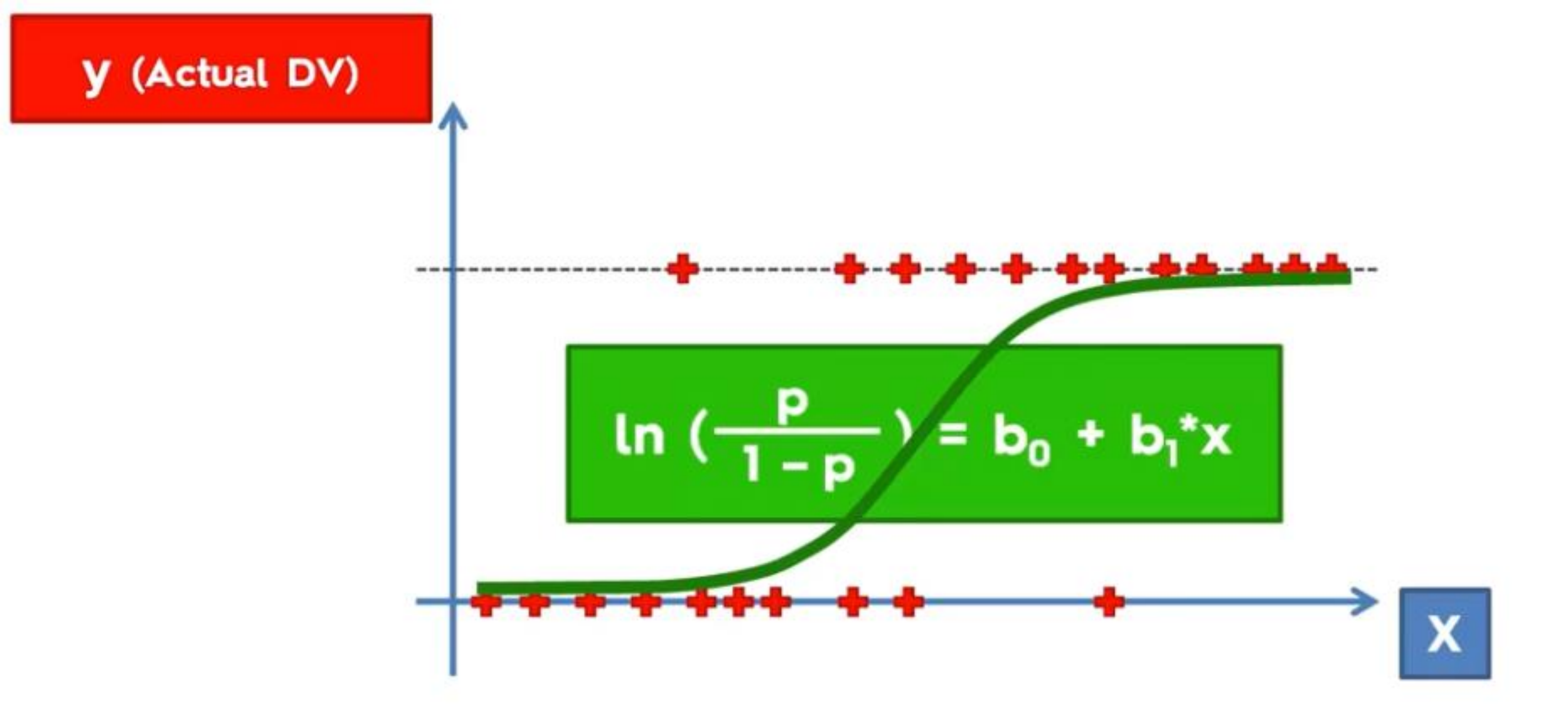
Action (Y/N)
1
0
Age

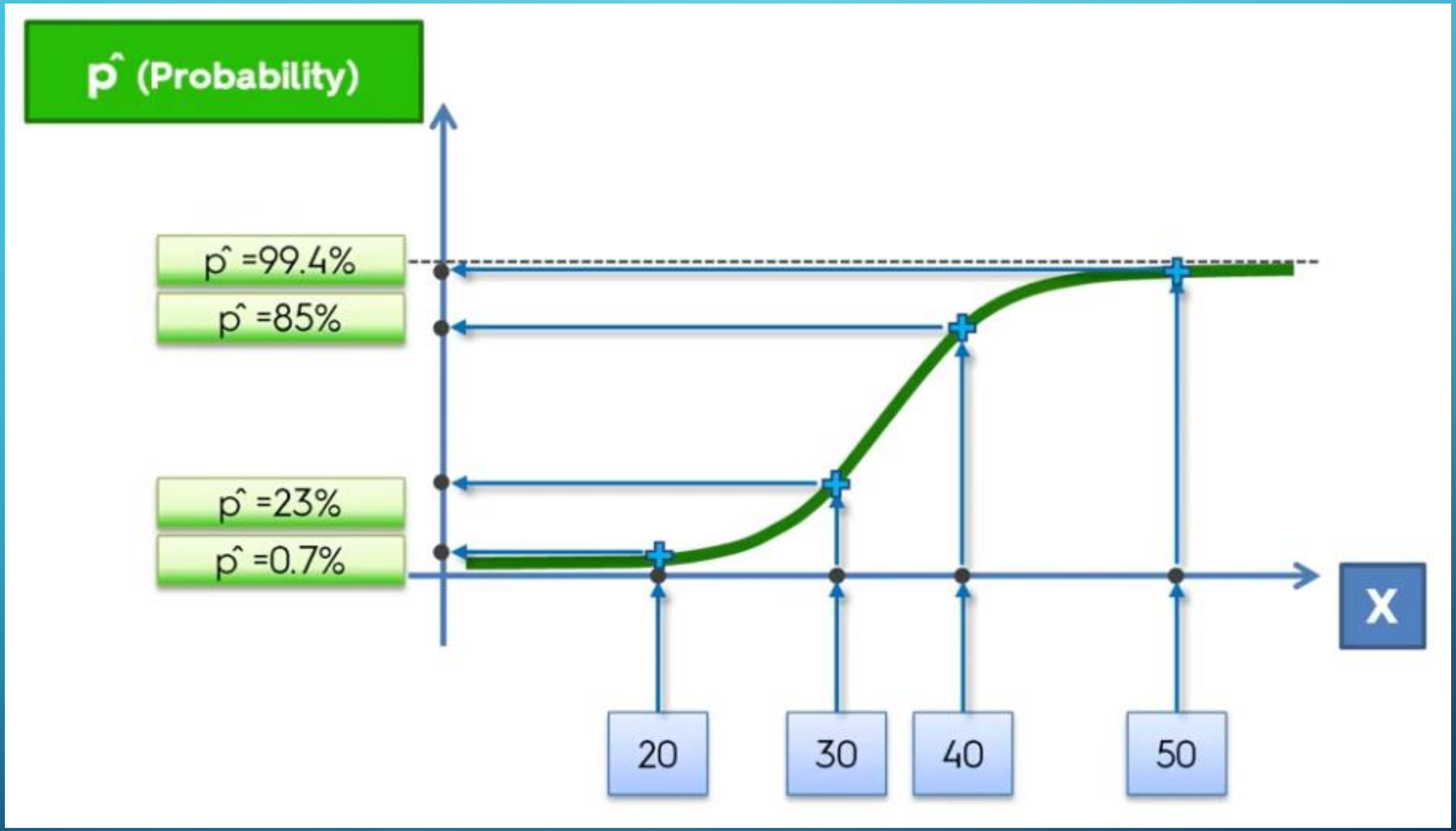


$y = b_0 + b_1*x$
Sigmoid Function
$p = \frac{1}{1 + e^{-y}}$
$\ln\left(\frac{p}{1-p}\right) = b_0 + b_1*x$

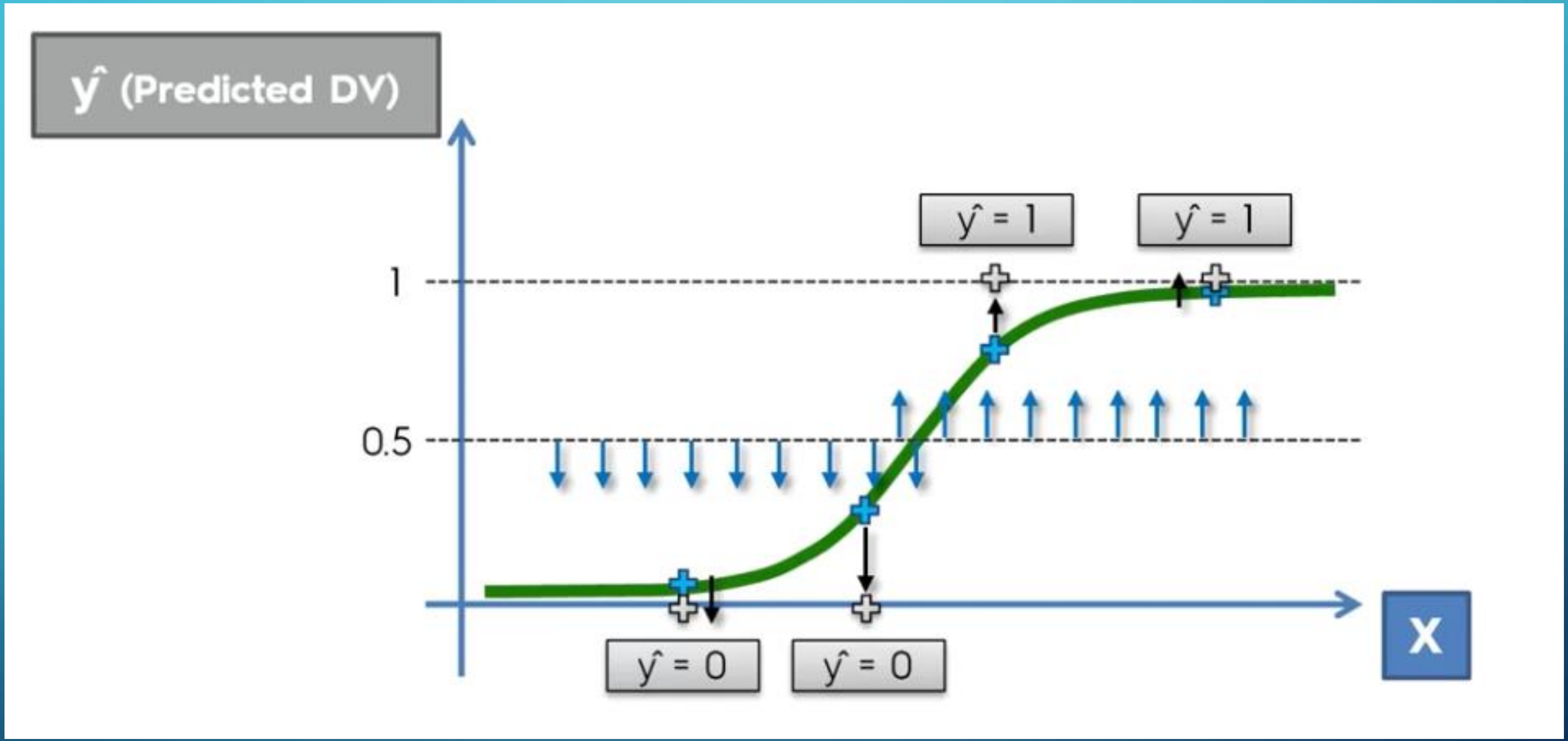

WHAT JUST
HAPPENED
???

ACTIVATION FUNCTION





p̂ (Probability)
p̂ =99.4%
p̂ =85%
p̂ =23%
p̂ =0.7%
X
20
30
40
50


ŷ (Predicted DV)
ŷ = 1
ŷ = 1
1
0.5
ŷ = 0
ŷ = 0
X

GOAL:

Logistic Regression Model

Want $0 \leq h_\theta(x) \leq 1$

Interpretation of Hypothesis Output

$h_\theta(x)$ = estimated probability that y = 1 on input x

Example: If $x = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ \text{tumorSize} \end{bmatrix}$

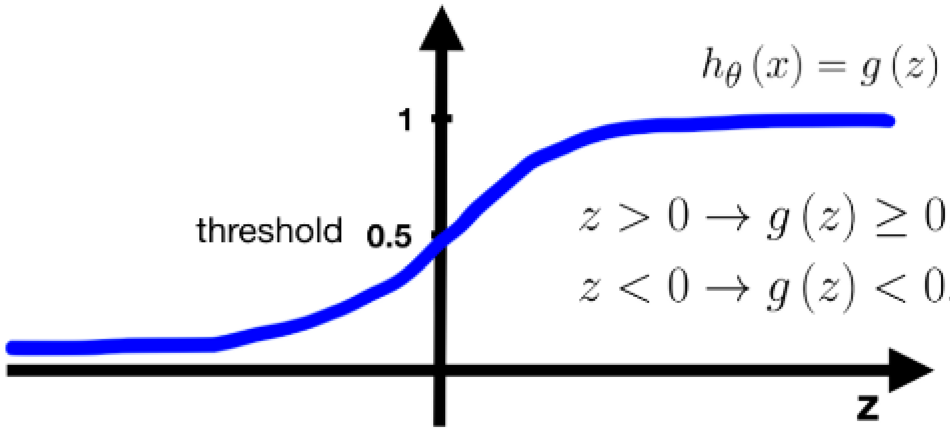
$$h_\theta(x) = 0.7$$

Tell patient that 70% chance of tumor being malignant

DECISION BOUNDARY

$$h_\theta(x) = g(z) = \frac{1}{1+e^{-z}}$$

$$(z = \theta_0 + \theta_1 x_1 + \ldots + \theta_n x_n = \theta^T x)$$

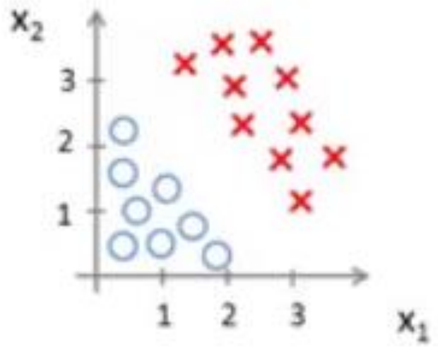


$$z > 0 \rightarrow g(z) \geq 0.5$$

$$z < 0 \rightarrow g(z) < 0.5$$

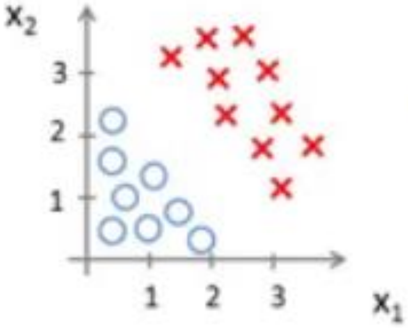
$$\boxed{\begin{array}{c} h_\theta(x) \geq 0.5 \rightarrow y = 1 \\ h_\theta(x) < 0.5 \rightarrow y = 0 \\ 0 \leq h_\theta(x) \leq 1 \end{array}}$$

Decision Boundary



- $h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$

Decision Boundary



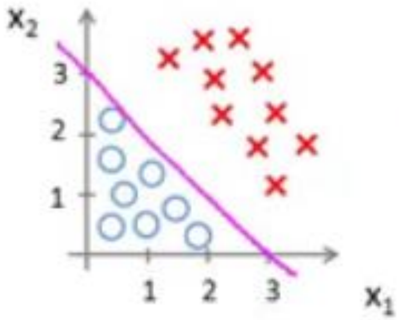
$$\theta = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix}$$

$$\rightarrow h_\theta(x) = g(\underbrace{\theta_0}_{= -3} + \underbrace{\theta_1}_{= 1} x_1 + \underbrace{\theta_2}_{= 1} x_2)$$

Predict "$y = 1$" if $-3 + x_1 + x_2 \geq 0$

Decision Boundary

$\theta = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix}$



$$h_\theta(x) = g(\underbrace{\theta_0}_{=-3} + \underbrace{\theta_1}_{=1} x_1 + \underbrace{\theta_2}_{=1} x_2)$$

x_1, x_2

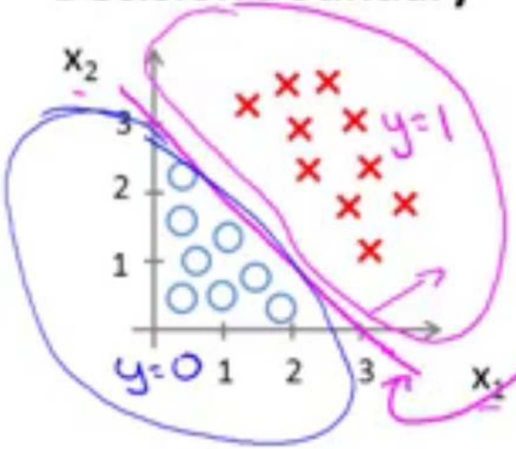
Predict "$y = 1$" if $\underbrace{-3 + x_1 + x_2 \geq 0}_{\theta^T x}$

$\to x_1 + x_2 \geq 3$

$x_1 + x_2 = 3$

Decision Boundary

$\theta = \begin{bmatrix}-3\\1\\1\end{bmatrix}$



$\rightarrow h_\theta(x) = g(\underbrace{\theta_0}_{-3} + \underbrace{\theta_1}_{1} x_1 + \underbrace{\theta_2}_{1} x_2)$

Decision bound.

x_1, x_2

Predict "$y = 1$" if $\underbrace{-3 + x_1 + x_2 \geq 0}_{\theta^T x}$

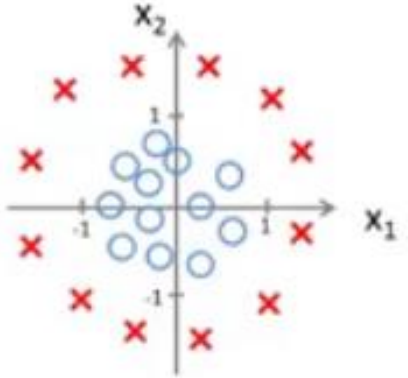
$\rightarrow x_1 + x_2 \geq 3$

$x_1 + x_2 = 3$

$x_1 + x_2 < 3$

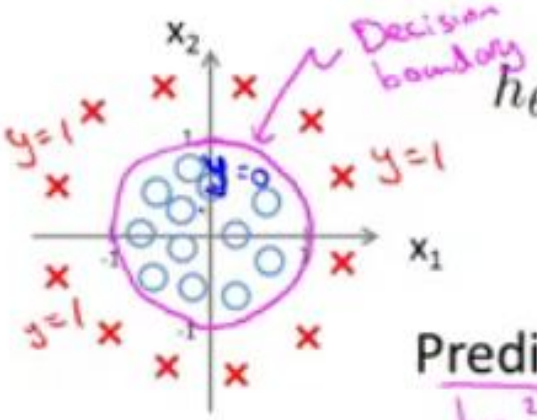
$\rightarrow y = 0$

Non-linear decision boundaries



$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$$

Non-linear decision boundaries



$$h_\theta(x) = g(\overset{=-1}{\theta_0} + \overset{=0}{\theta_1} x_1 + \overset{=0}{\theta_2} x_2 + \underset{=1}{\theta_3} x_1^2 + \underset{=1}{\theta_4} x_2^2)$$

$$\theta = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

Predict "$y = 1$" if $\underbrace{-1 + x_1^2 + x_2^2 \geq 0}_{x_1^2 + x_2^2 \geq 1}$

$\boxed{x_1^2 + x_2^2 = 1}$

COST FUNCTION

Training set: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \cdots, (x^{(m)}, y^{(m)})\}$

m examples $\quad x \in \begin{bmatrix} x_0 \\ x_1 \\ \dots \\ x_n \end{bmatrix} \quad x_0 = 1, y \in \{0, 1\}$

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

How to choose parameters θ ?

Cost function

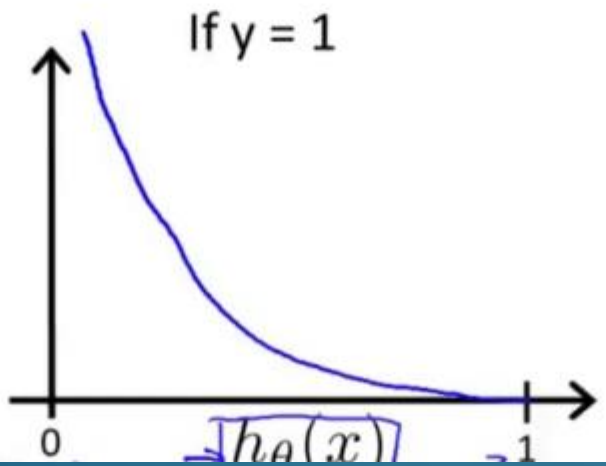
Linear regression: $J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{2}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$

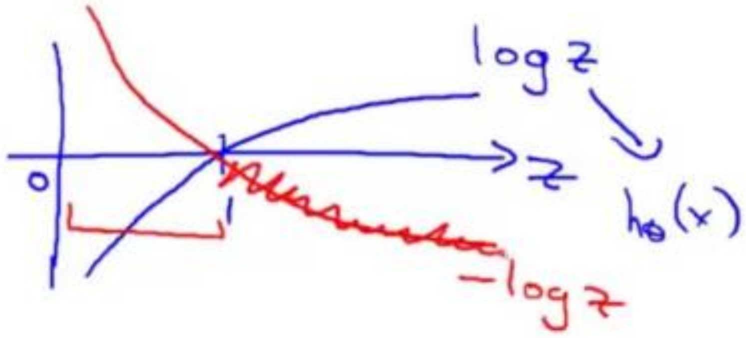
Logistic regression cost function

$$\mathrm{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

Logistic regression cost function

$$\mathrm{Cost}(h_\theta(x), y) = \begin{cases} \boxed{-\log(h_\theta(x))} & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$





$$\text{Cost} = 0 \text{ if } y = 1, h_\theta(x) = 1$$

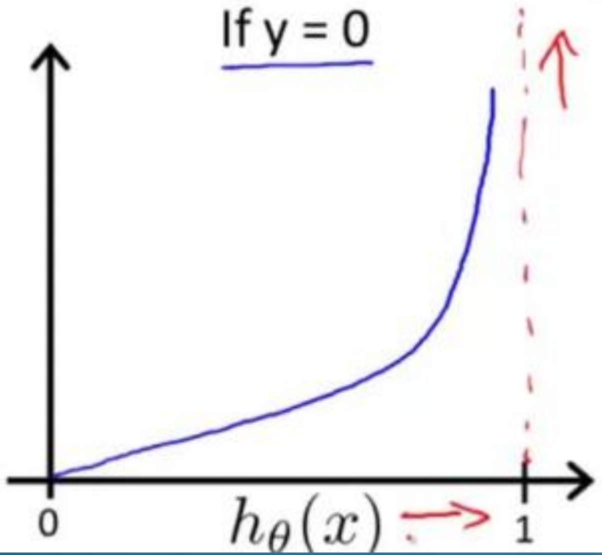
$$\text{But as} \quad h_\theta(x) \to 0$$

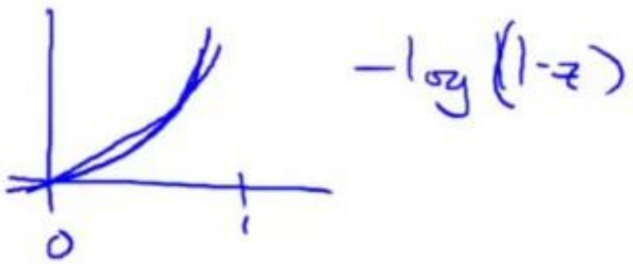
$$Cost \to \infty$$

Captures intuition that if $h_\theta(x) = 0$, (predict $P(y = 1|x;\theta) = 0$), but $y = 1$, we'll penalize learning algorithm by a very large cost.

Logistic regression cost function

$$\mathrm{Cost}(h_\theta(x^{(i)}, y^{(i)}) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ \boxed{-\log(1 - h_\theta(x)) \quad \text{if } y = 0} \end{cases}$$





Logistic regression cost function

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \mathrm{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

$$\mathrm{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

Note: $y = 0$ or 1 always

Logistic regression cost function

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \mathrm{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

$$= -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

Gradient Descent

$$J(\theta) = -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

Want $\min_\theta J(\theta)$:

Repeat $\{$

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

$\}$ (simultaneously update all θ_j)

Gradient Descent

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log\left(1 - h_\theta(x^{(i)})\right)\right]$$

Want $\min_\theta J(\theta)$:

Repeat $\{$

$$\theta_j := \theta_j - \alpha \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

(simultaneously update all θ_j)

$\}$

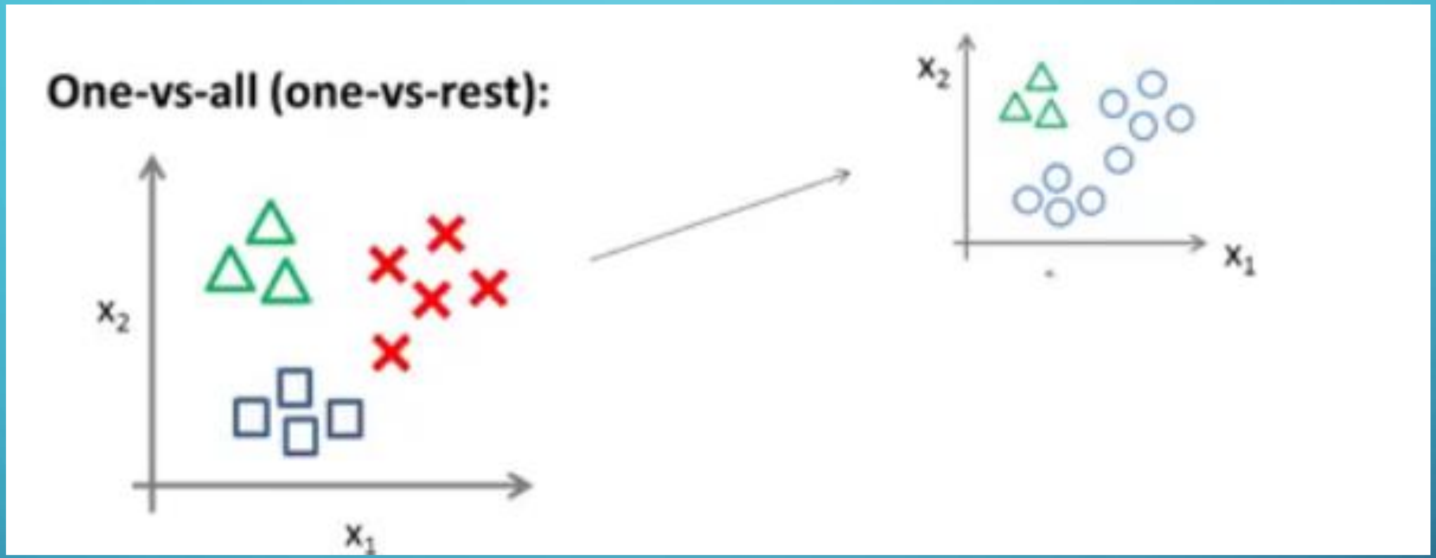
Algorithm looks identical to Linear Regression!!

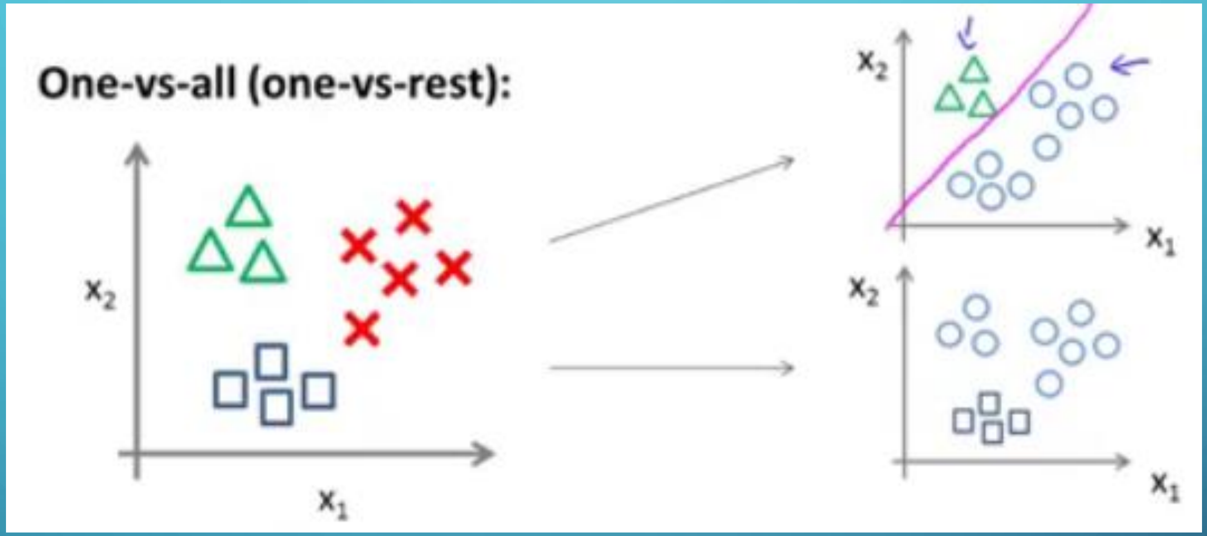
MULTI-CLASS CLASSIFICATION

One vs All!!!

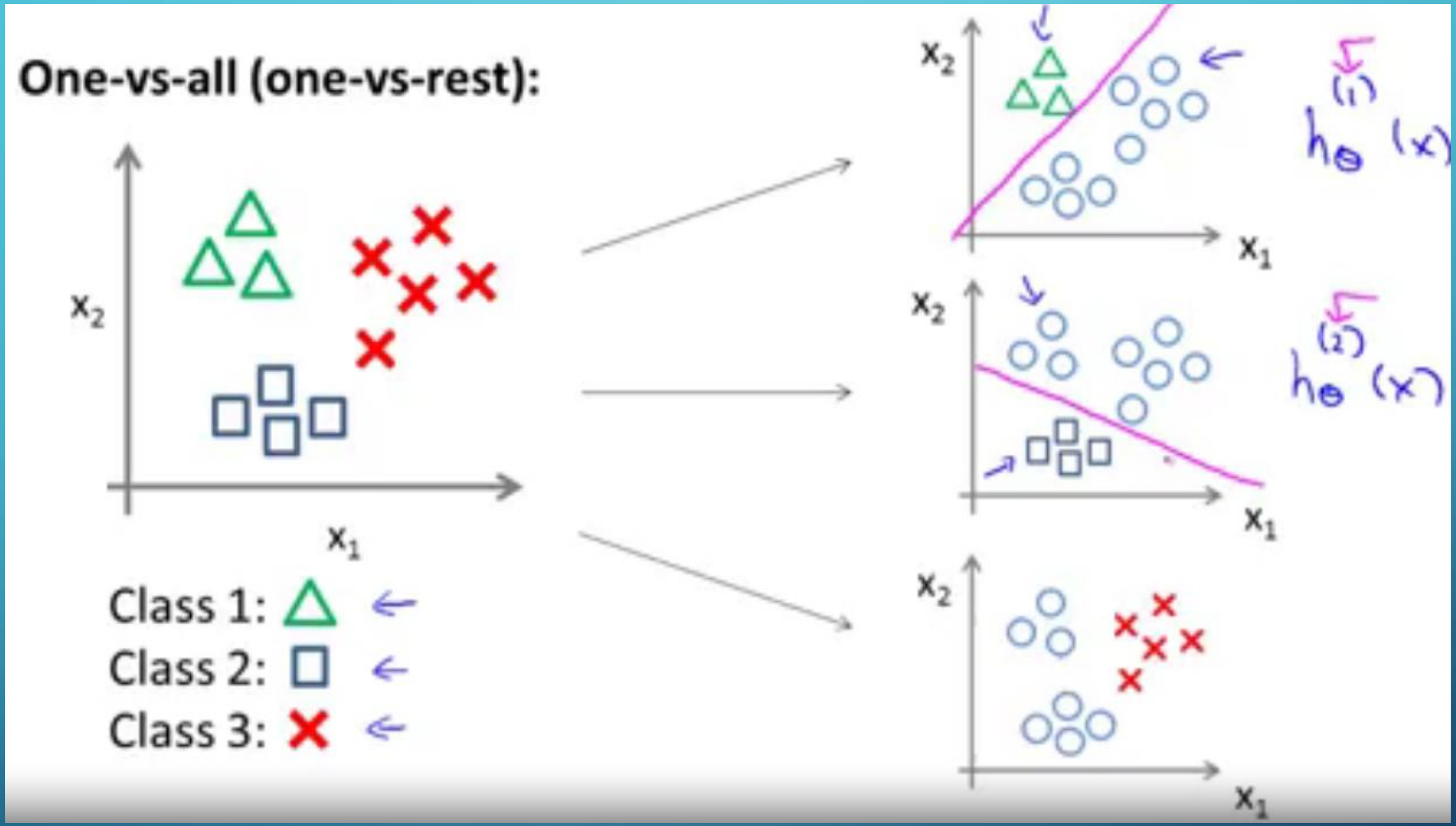
- Email foldering/tagging: Work, Friends, Family, Hobby
- Medical: Not ill, Cold, Flu
- Weather: Sunny, Cloudy, Rain, Snow


Binary classification:
Multi-class classification:
x2
x1
x2
x1


One-vs-all (one-vs-rest):
x2
x1
x2
x1



One-vs-all (one-vs-rest):
x_2
x_1
x_2
x_1
x_2
x_1



One-vs-all (one-vs-rest):
x2
x1
Class 1:
Class 2:
Class 3:

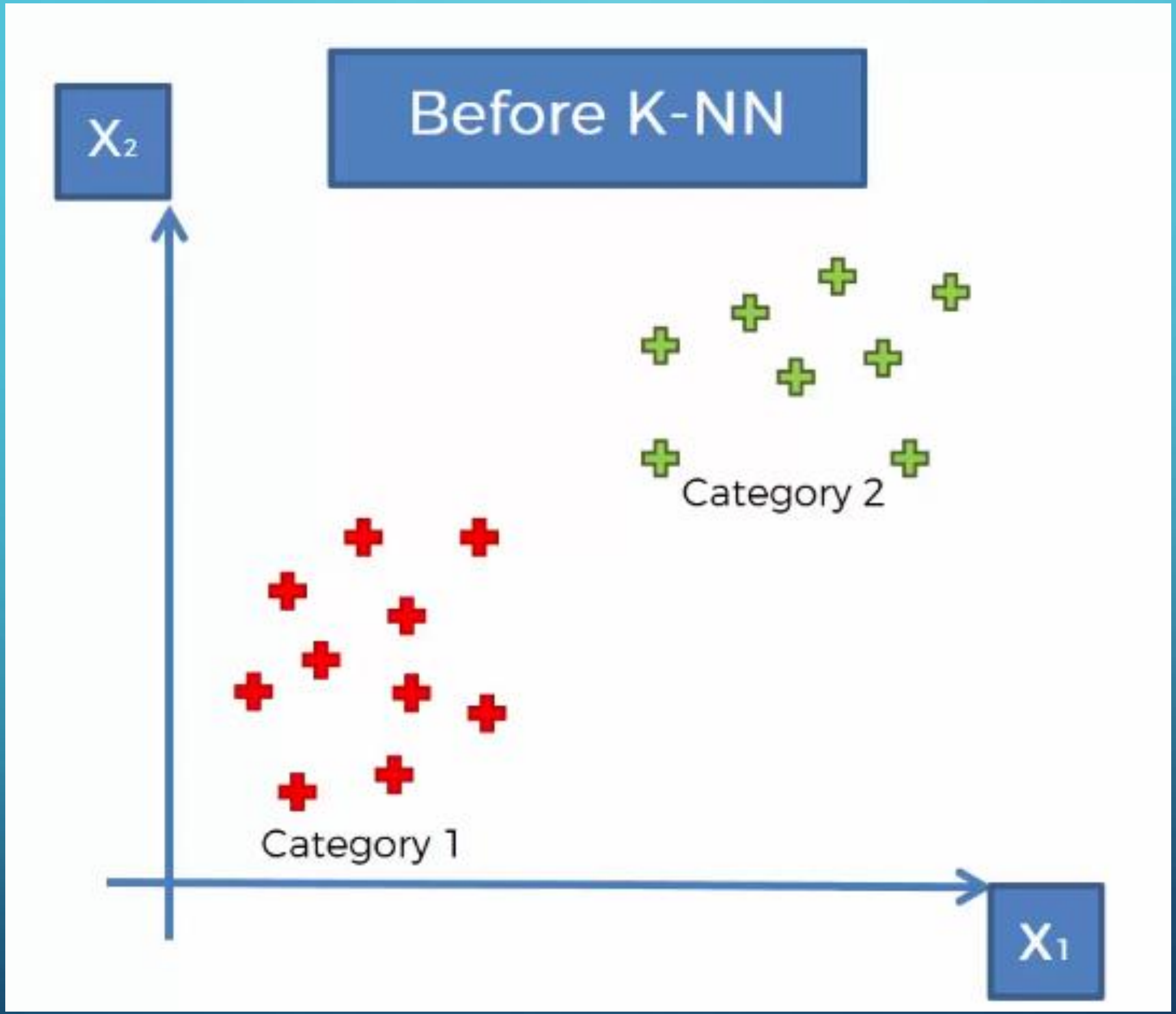
ONE VS ALL

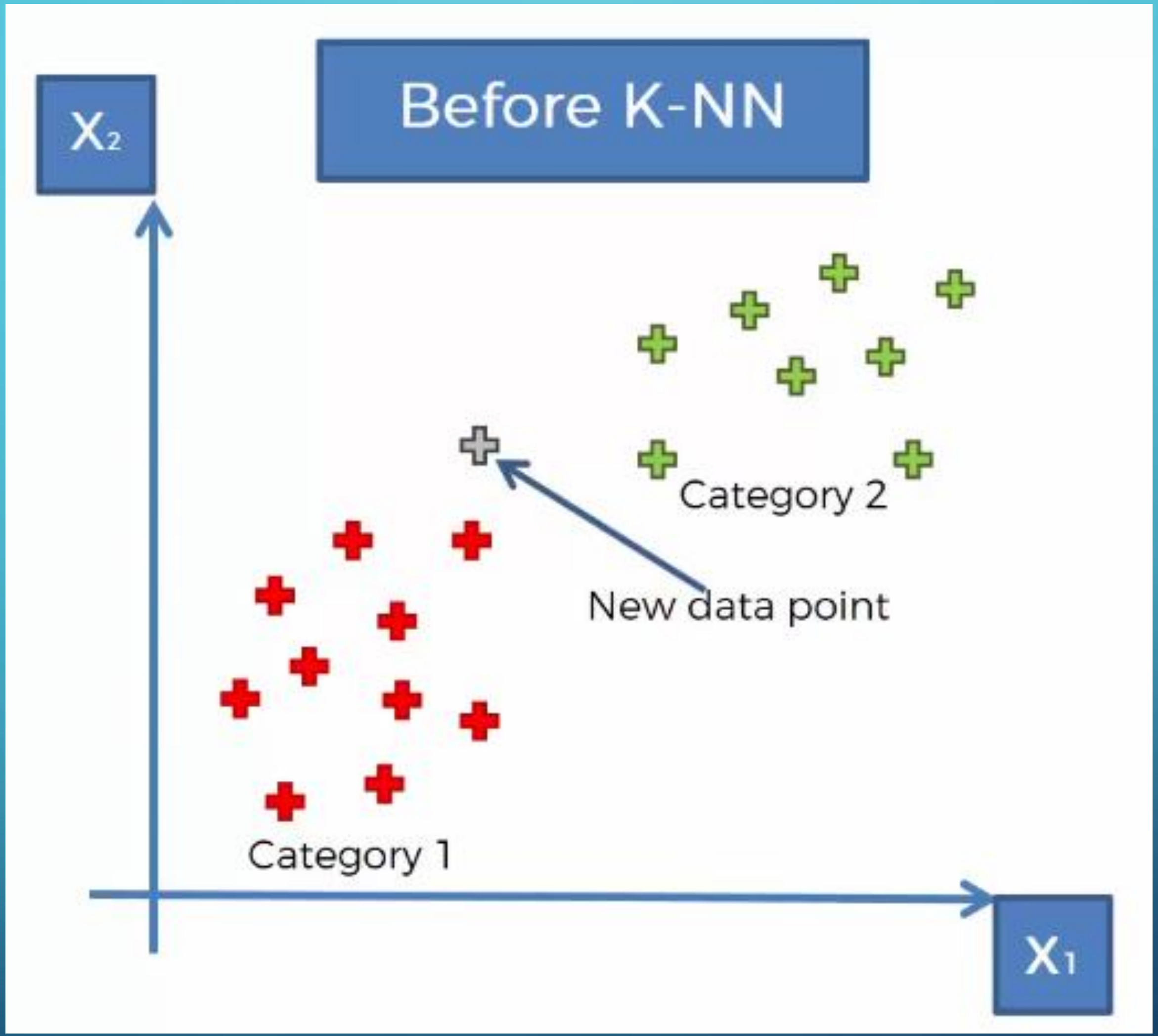
- Train a logistic regression classifiers for each class i, to predict the probability for y=I
- On a new input x, to make a prediction, pick the class i that has maximum probablity

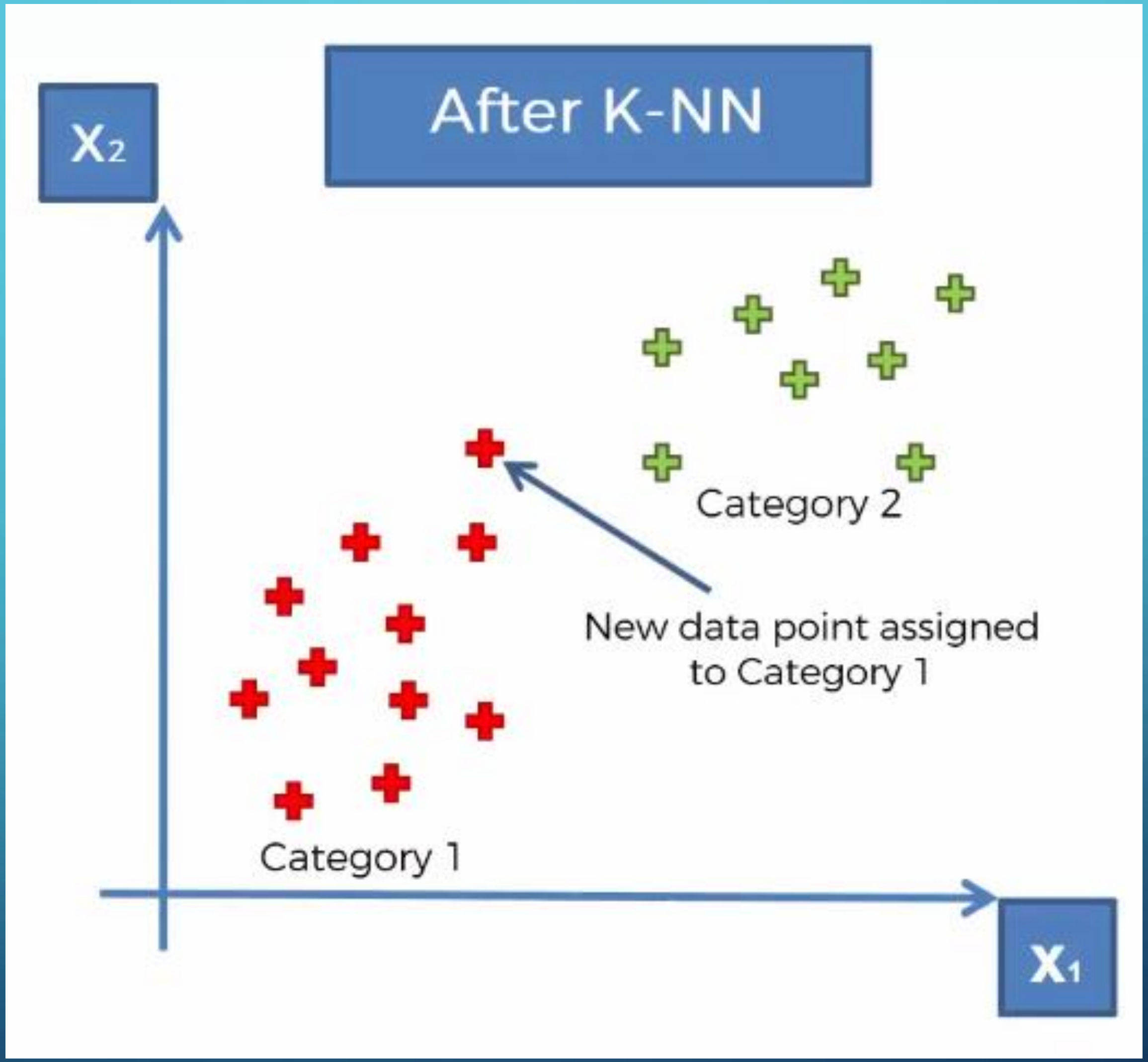
K-NEAREST NEIGHBOURS

OVERVIEW

- Simplest of all machine learning algorithms
- A non-parametric method for classification
- Instance-based/Lazy learning
- Based on “majority voting” classification


Before K-NN
X_2
Category 2
Category 1
X_1


Before K-NN
X_2
Category 2
New data point
Category 1
X_1


After K-NN
X2
X1
Category 2
Category 1
New data point assigned
to Category 1

HOW DID IT DO THAT?

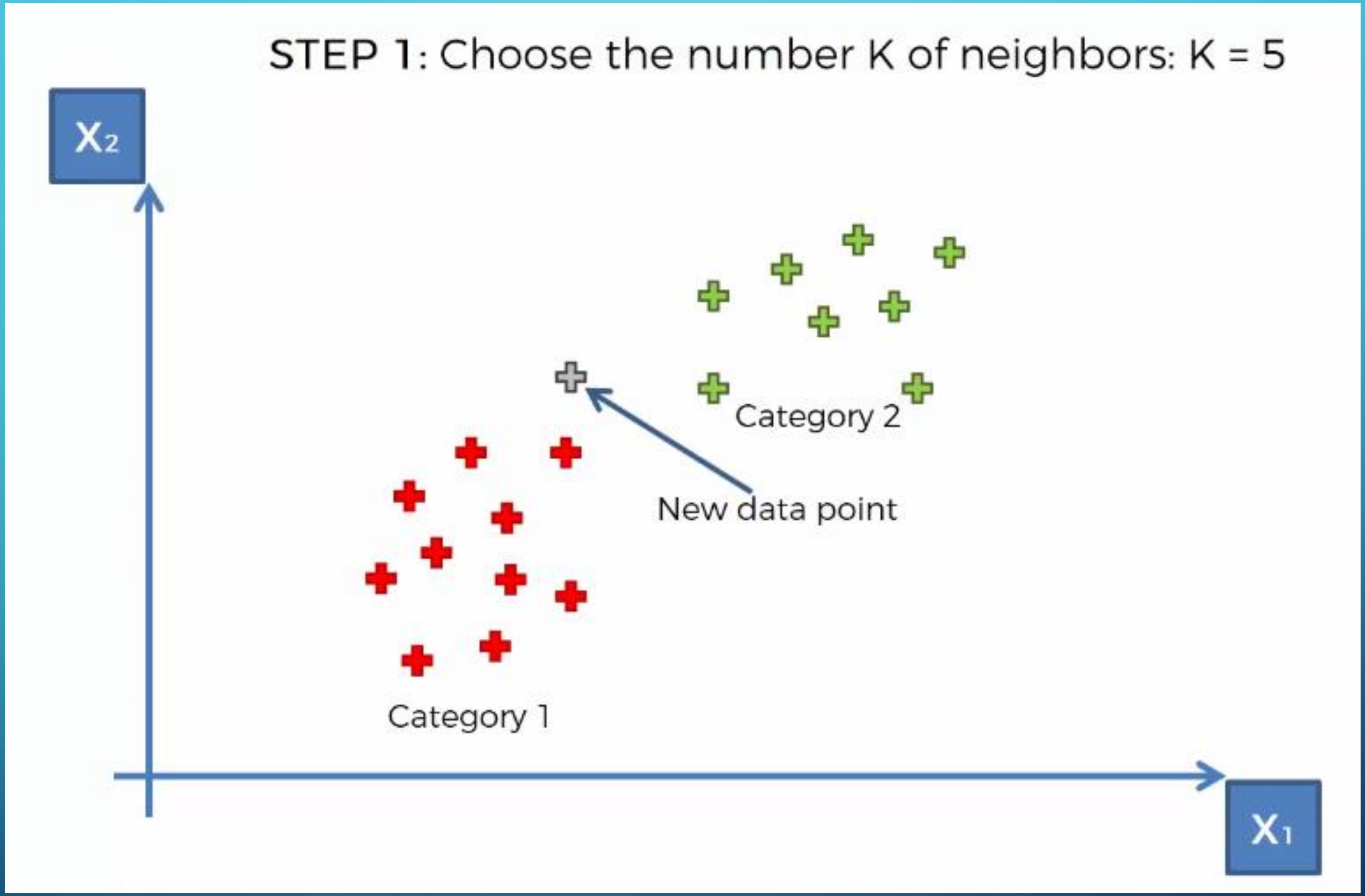
STEP 1: Choose the number K of neighbors

STEP 2: Take the K nearest neighbors of the new data point, according to the Euclidean distance

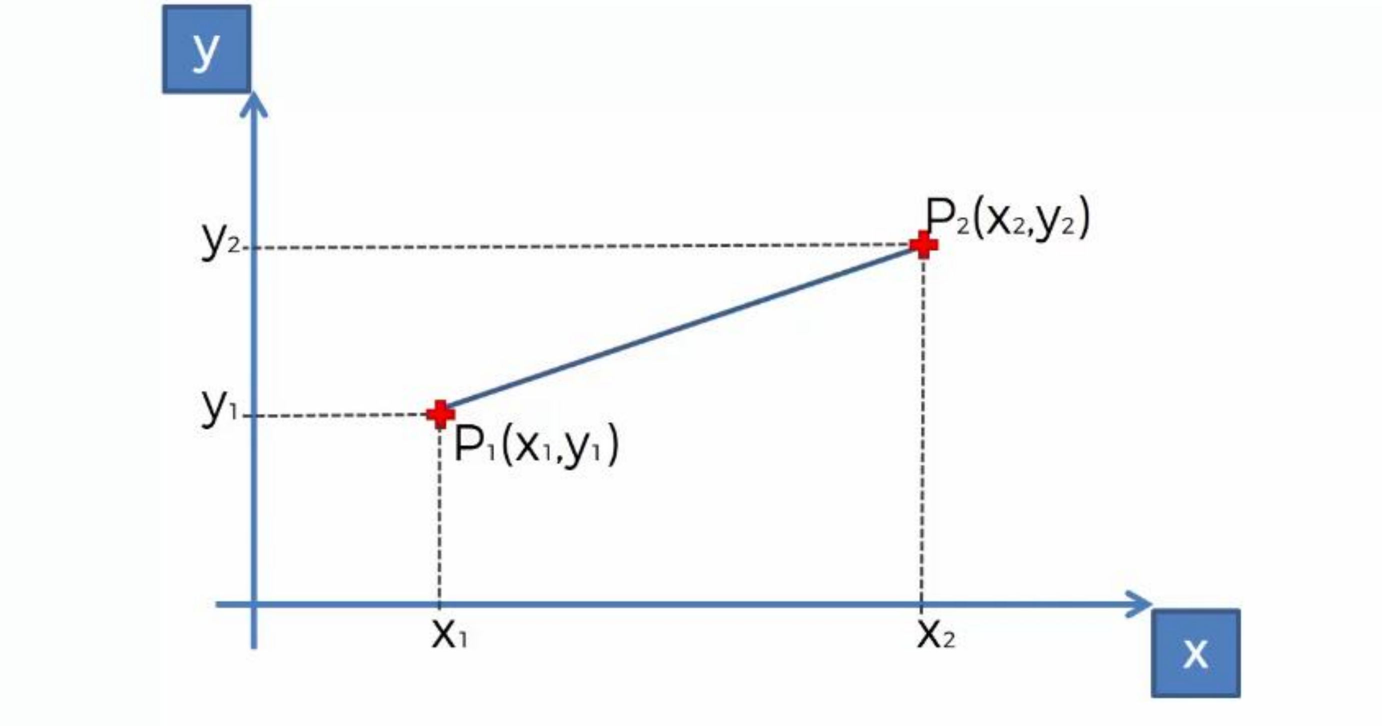
STEP 3: Among these K neighbors, count the number of data points in each category

STEP 4: Assign the new data point to the category where you counted the most neighbors

Your Model is Ready


STEP 1: Choose the number K of neighbors: K = 5
X2
New data point
Category 2
Category 1
X1

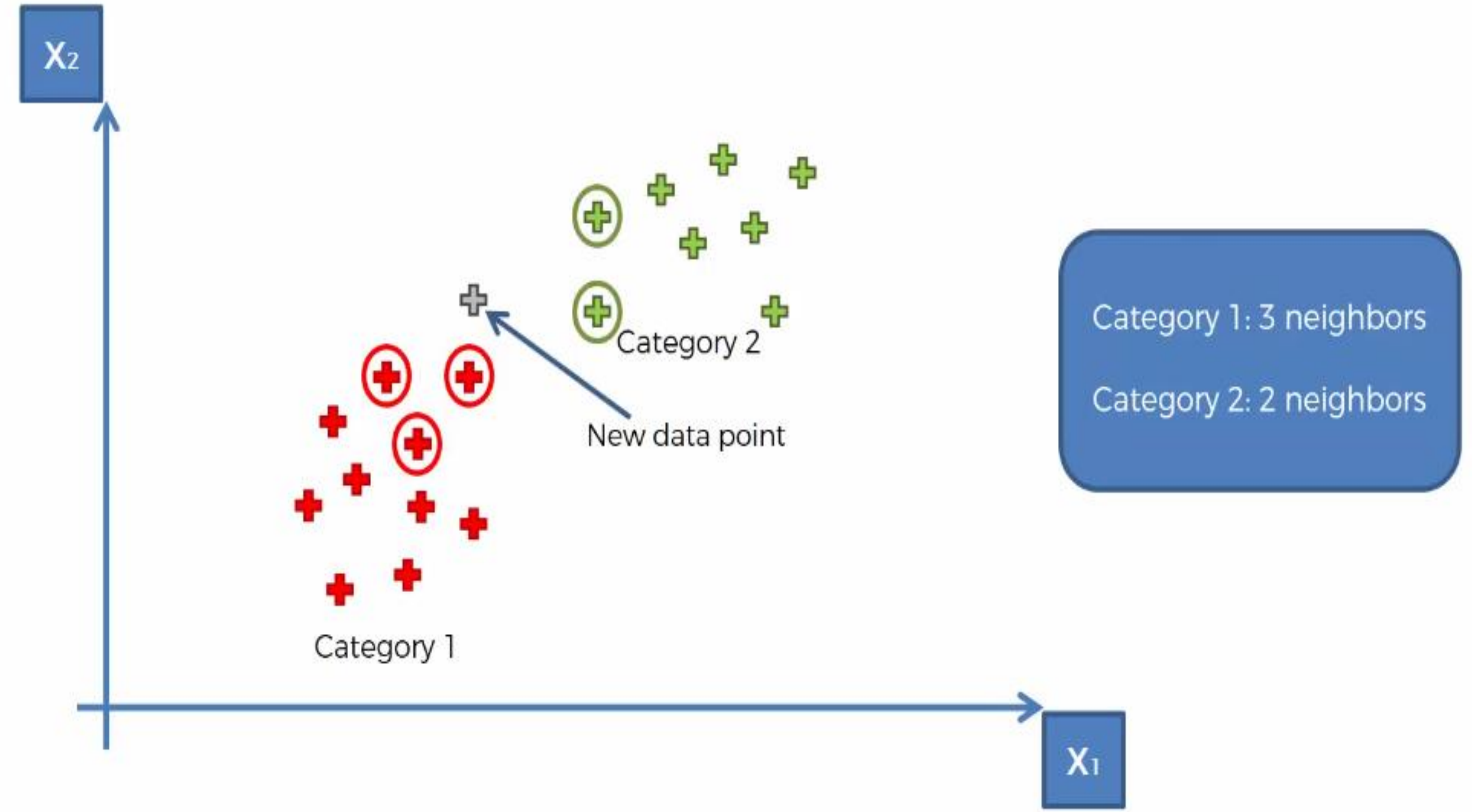
EUCLIDEAN DISTANCE



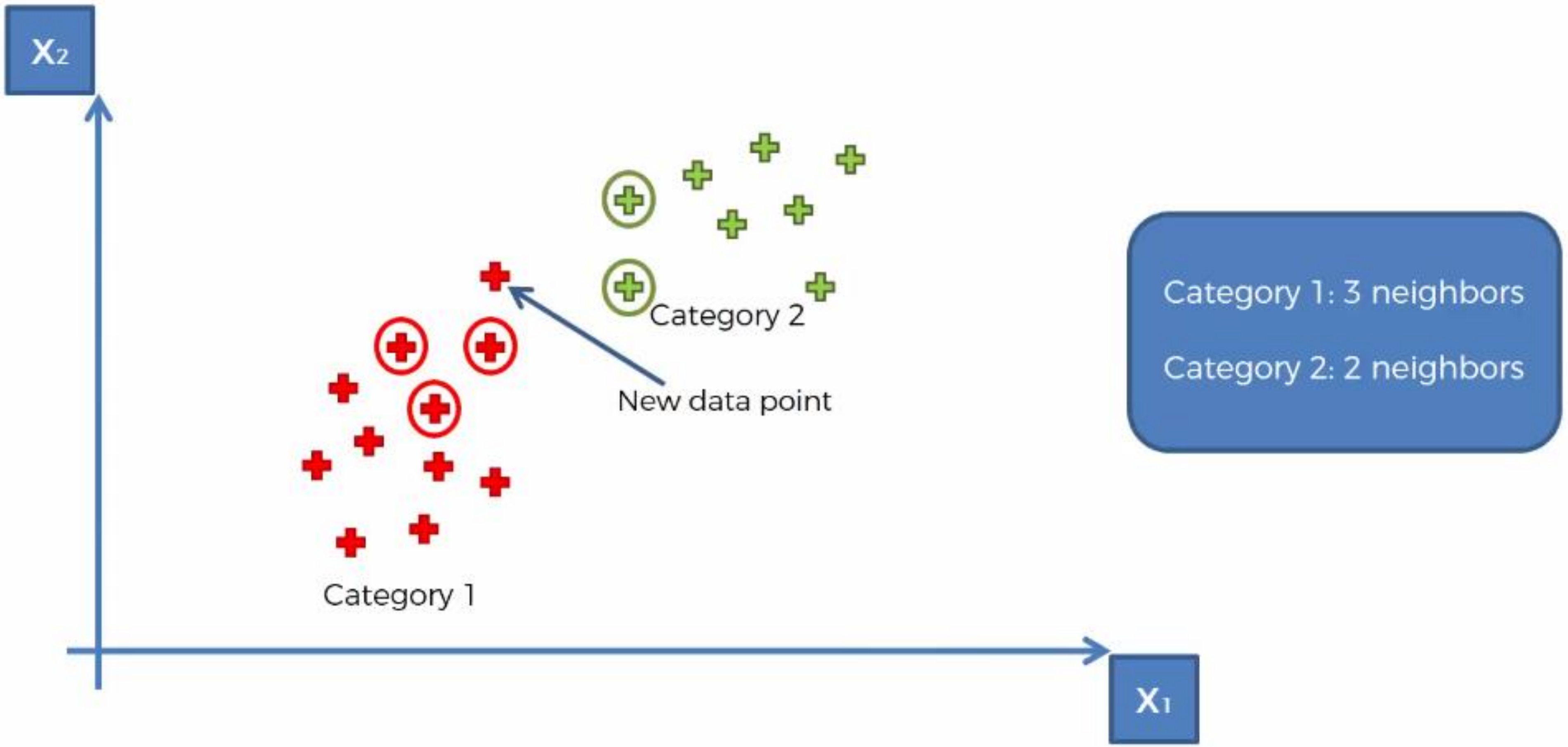
$$\text{Euclidean Distance between } P_1 \text{ and } P_2 = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

STEP 2: Take the K = 5 nearest neighbors of the new data point, according to the Euclidean distance

STEP 3: Among these K neighbors, count the number of data points in each category



STEP 4: Assign the new data point to the category where you counted the most neighbors



WITH N-EXAMPLES, HOW FAST ARE TRAINING AND PREDICTION?

TRAIN: O(1)
TEST: O(N)

This is bad: we want classifiers that are **fast** at prediction; **slow** for training is ok

HYPERPARAMETERS

- Parameters of model which we, as ML developer, need to tune to build the efficient model
- Hyperparameters are set by developer during training and is used during testing
- K value in K-NN algorithm is the hyperparameter

WHAT IS THE BEST VALUE OF K TO USE?

WHAT IS THE BEST VALUE OF K TO USE?

- Very problem-dependent
- Must try them all out and see what works best

HOW TO SET HYPERPARAMETERS?

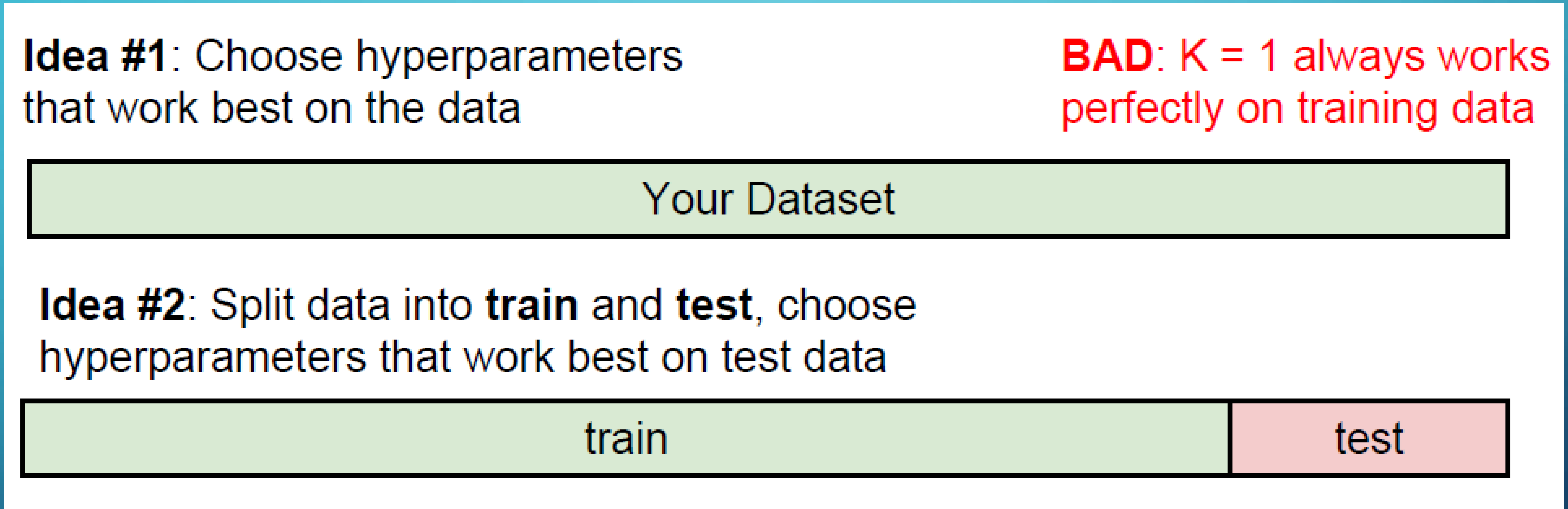
Idea #1: Choose hyperparameters
that work best on the data

Your Dataset

Idea #1: Choose hyperparameters that work best on the data

BAD: K = 1 always works perfectly on training data

Your Dataset


Idea #1: Choose hyperparameters
that work best on the data
BAD: K = 1 always works
perfectly on training data
Your Dataset
Idea #2: Split data into **train** and **test**, choose
hyperparameters that work best on test data
train
test

Idea #1: Choose hyperparameters that work best on the data

BAD: K = 1 always works perfectly on training data

Your Dataset

Idea #2: Split data into **train** and **test**, choose hyperparameters that work best on test data

BAD: No idea how algorithm will perform on new data

train	test

Idea #1: Choose hyperparameters that work best on the data

BAD: K = 1 always works perfectly on training data

Your Dataset

Idea #2: Split data into **train** and **test**, choose hyperparameters that work best on test data

BAD: No idea how algorithm will perform on new data

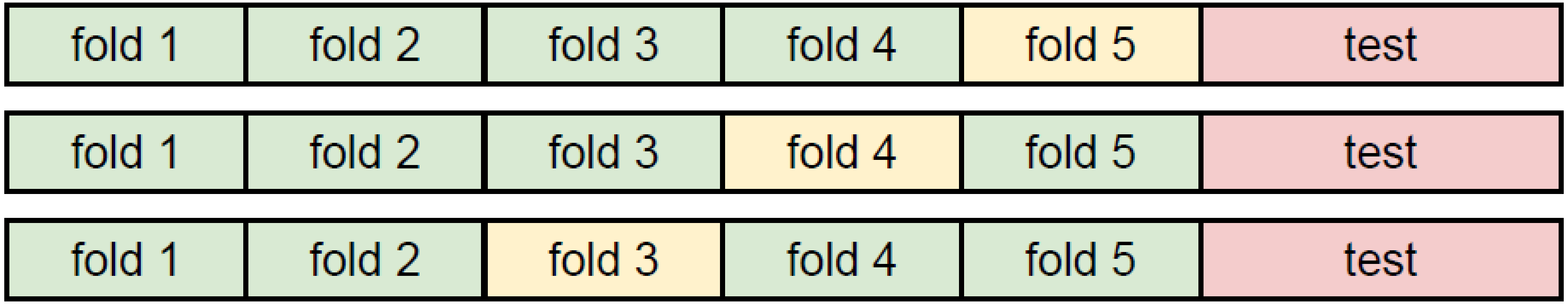
train	test

Idea #3: Split data into **train**, **val**, and **test**; choose hyperparameters on val and evaluate on test

Better!

train	validation	test

Idea #4: **Cross-Validation**: Split data into **folds**,
try each fold as validation and average the results



Useful for small datasets, but not used too frequently in deep learning

K-NN SUMMARY:

- Predicts labels based on nearest training examples
- It is never used on images
- Very slow at test time
- Choose hyperparameters using the **validation set**; only run on the test set once at the very end!

THANK YOU