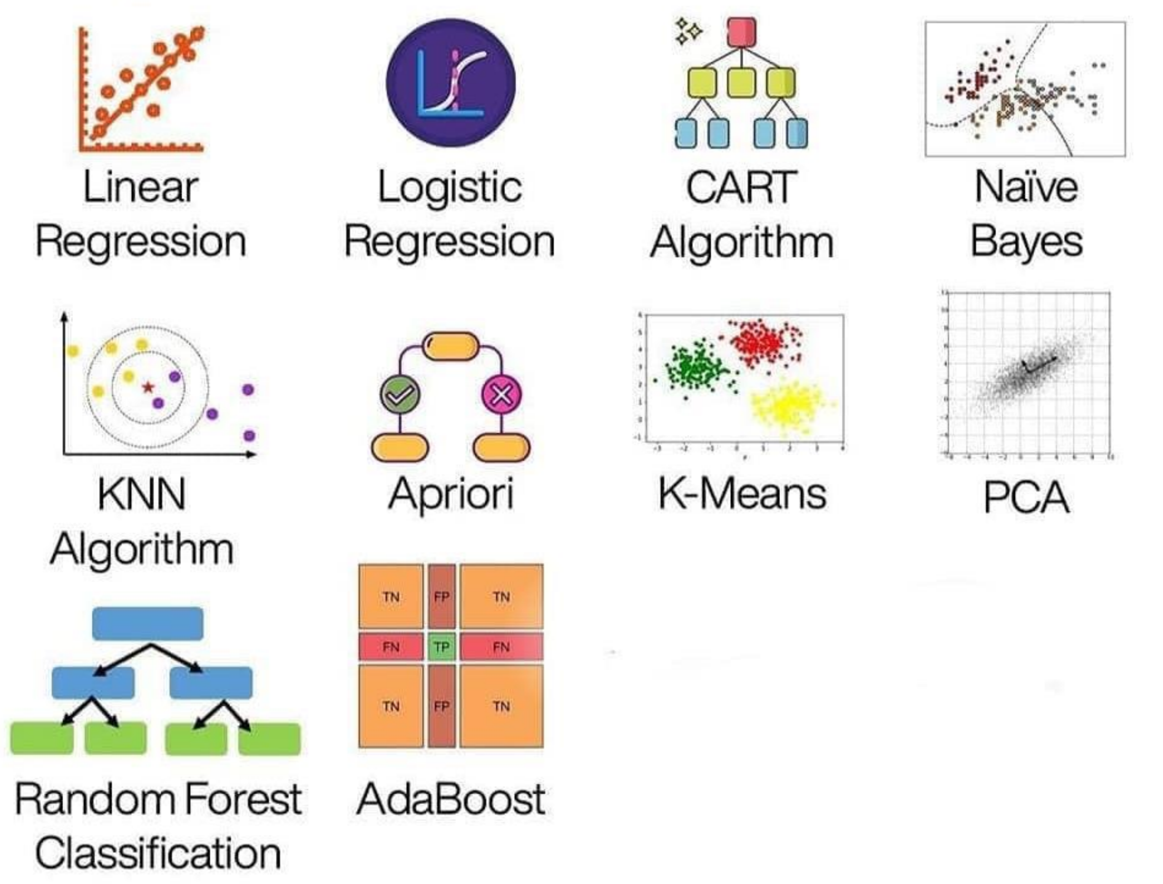
Machine Learning with Python



Lecture --

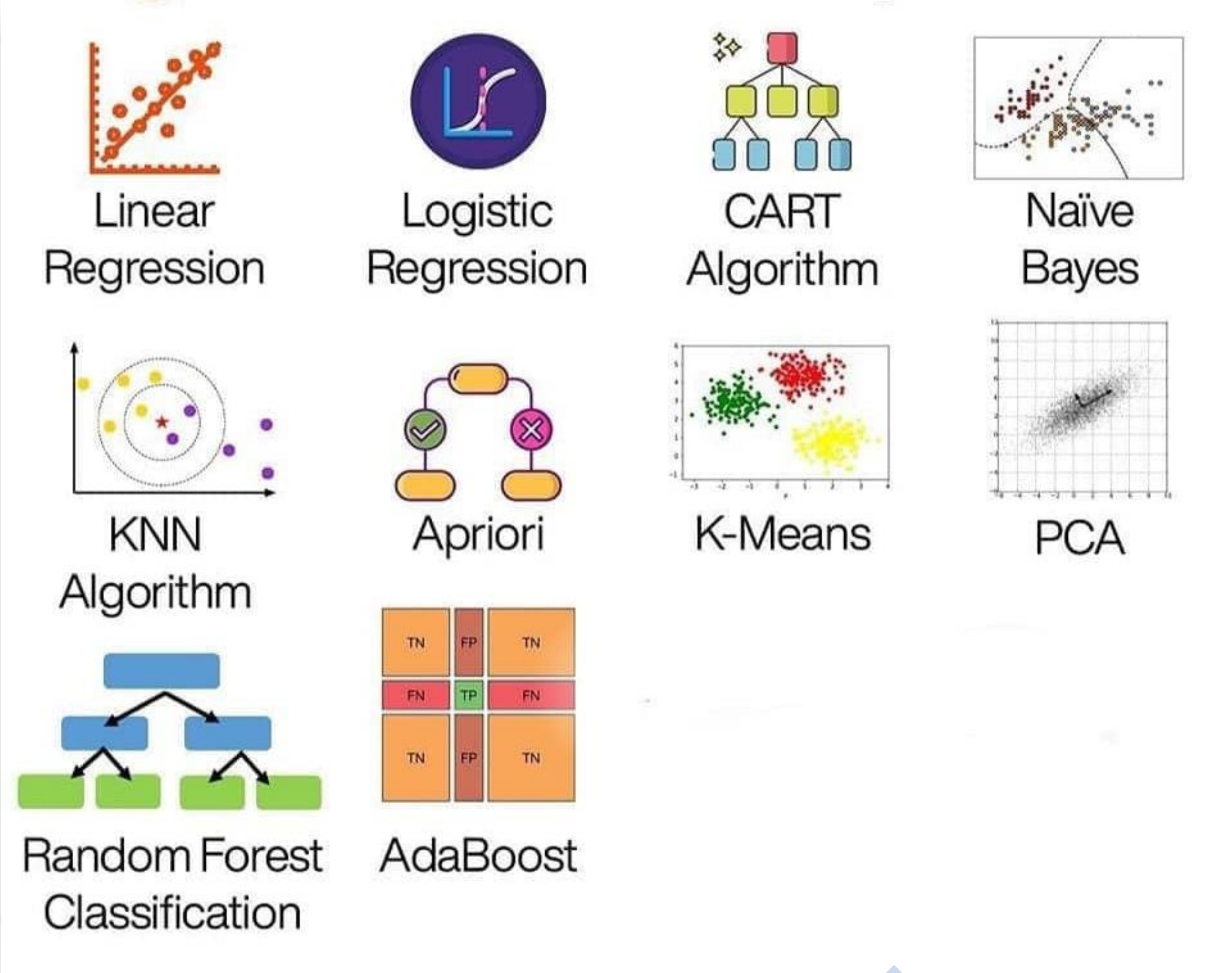
Dr. Sherif Eletriby

- **Classification**

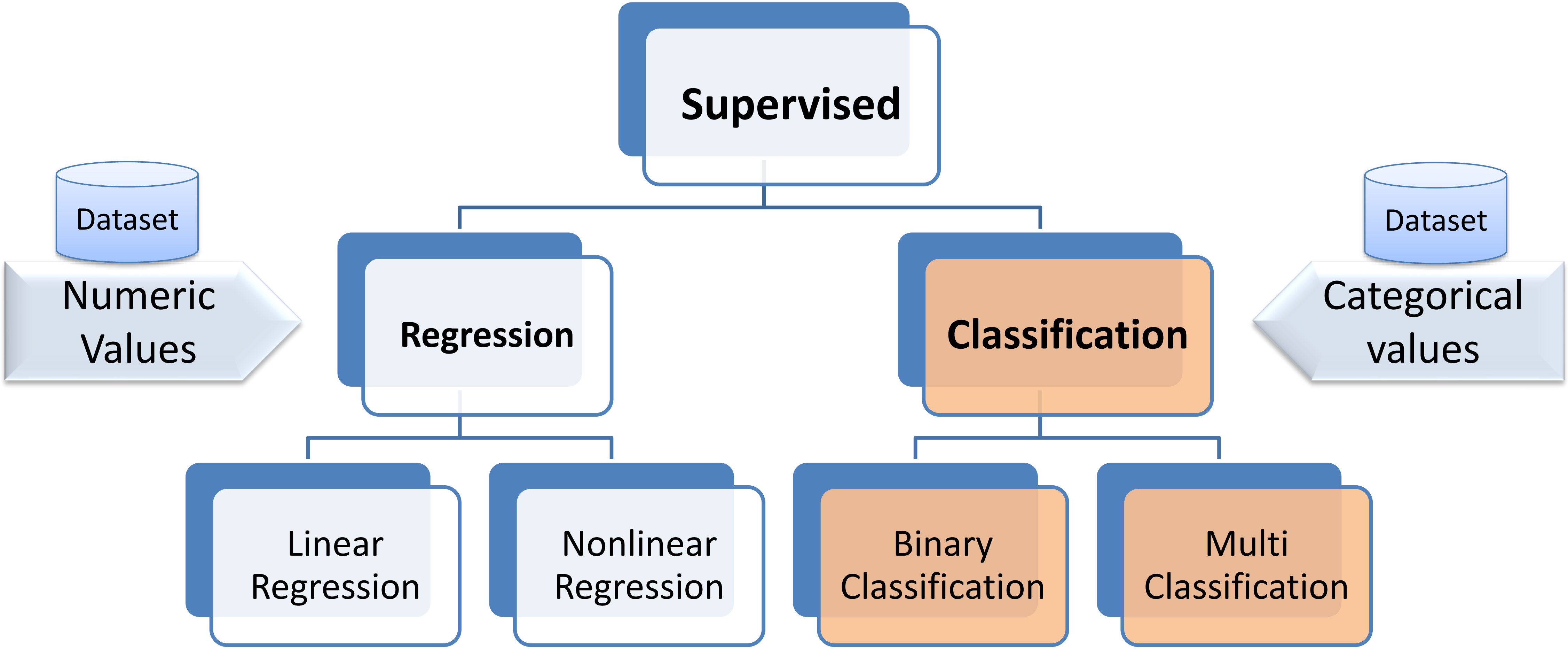
(Logistic Regression)

Classification
(Logistic Regression)

Machine learning algorithms



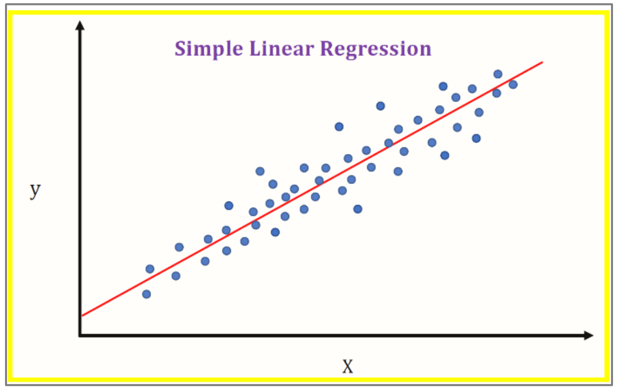
Supervised Learning التعلم بواسطة الإشراف

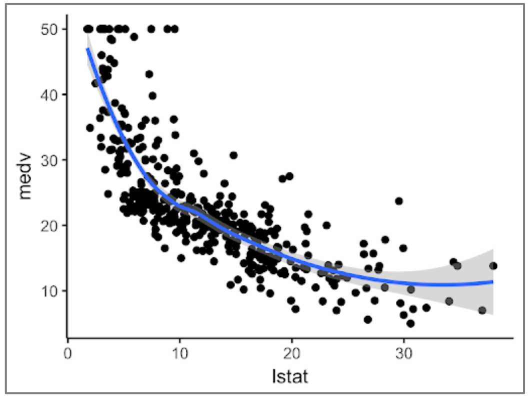


Regression

- Given $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$
- Learn a function f(x) to predict y given x

❑ **y** is real-valued



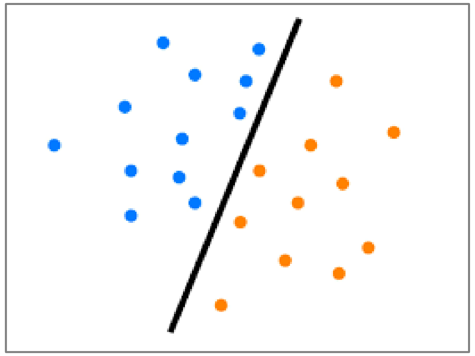


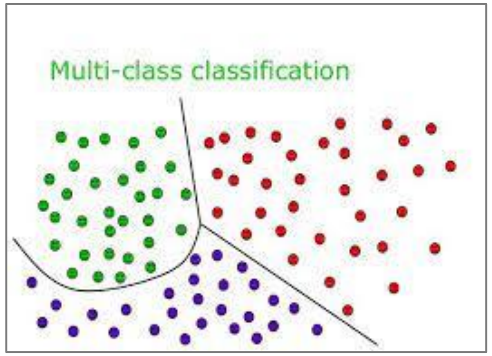
Output is **continuance** values

Classification

- Given $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$
- Learn a function f(x) to predict y given x

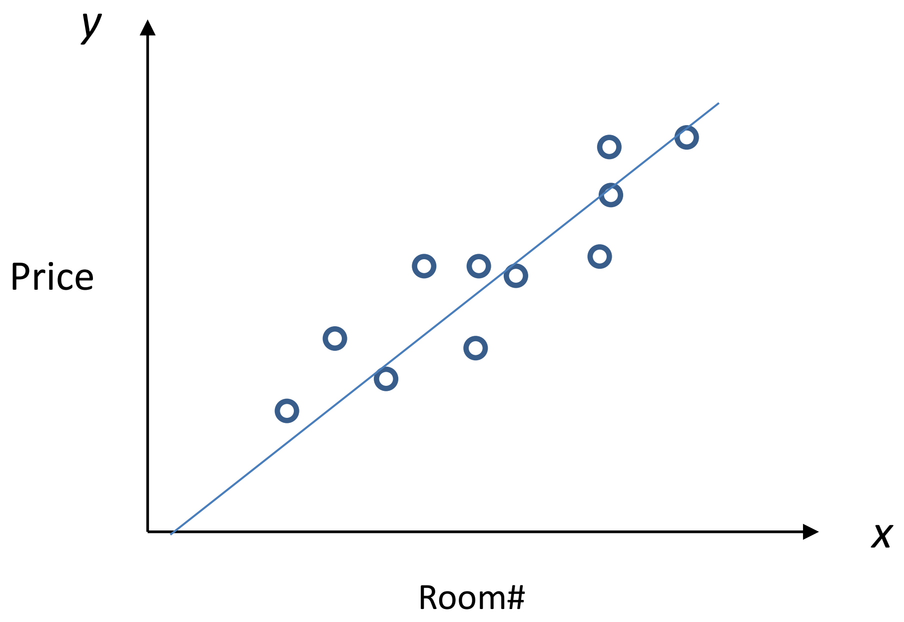
❑ **y** is categorical





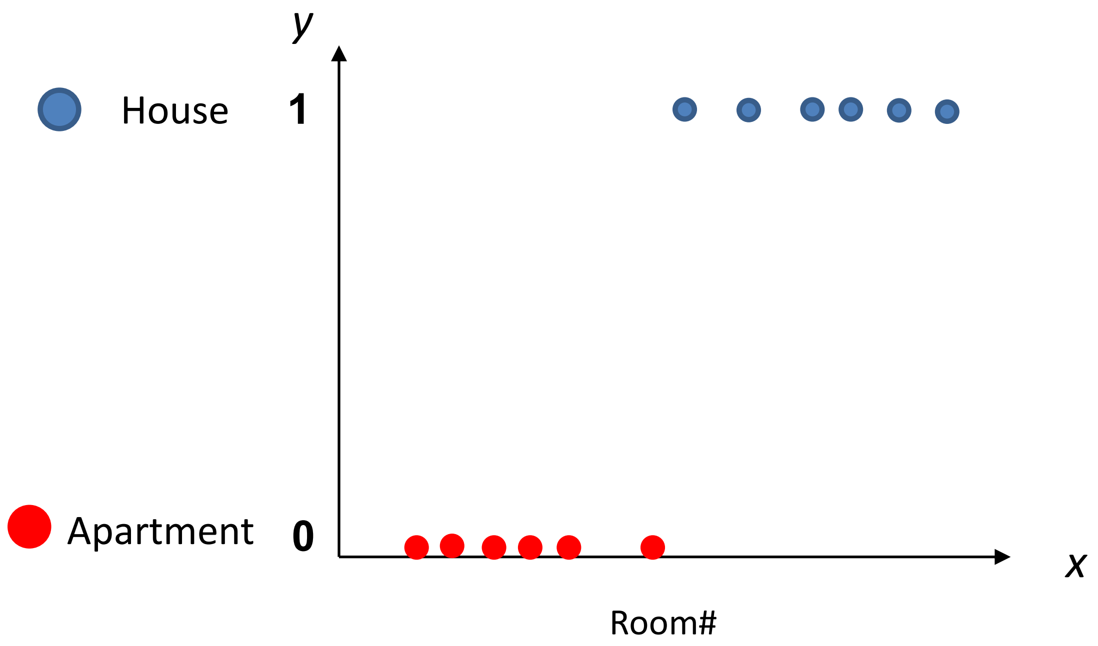
Output is **discrete** نفصل values

Room#	Price
3	50
5	70
4	100
6	150



Room#	Type
3	0
5	1
4	0
6	1

Label
House ------------- 1
Apartment ------- 0

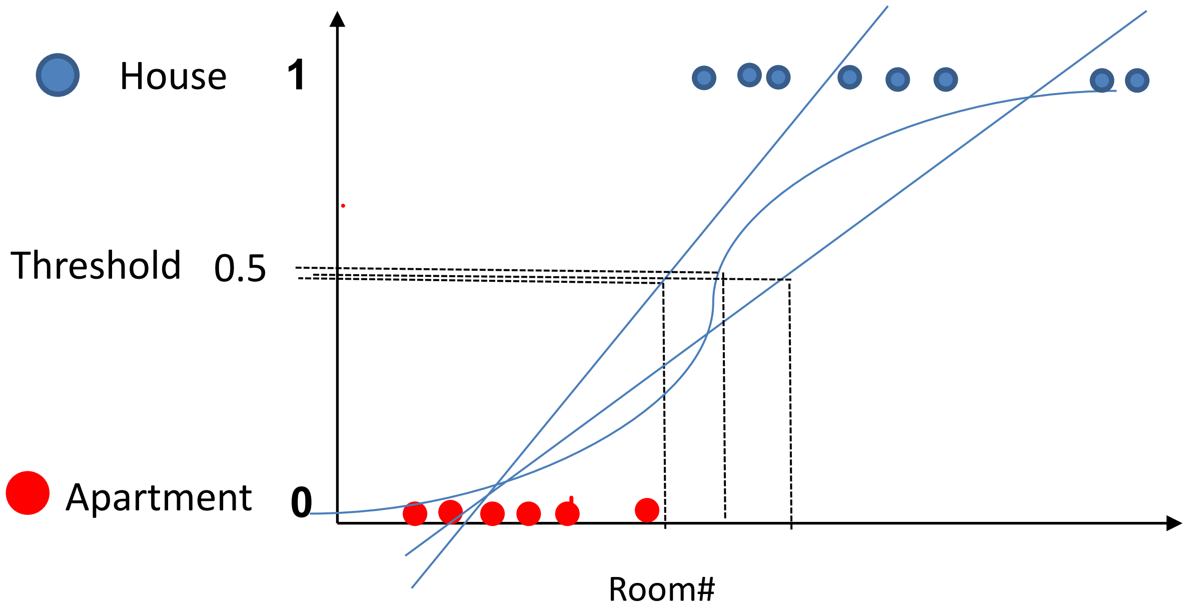


Classification

Room#	Type
3	0
5	1
4	0
6	1

If $h(x) \geq 0.5$, Then y=1

If $h(x) < 0.5$, Then y=0



Logistic Regression $0 \leq h(x) \leq 1$

$$h(x) = \theta_0 + \theta_1 x_1$$

Sigmoid Function $g(z) = \dfrac{1}{1 + e^{-z}}$

$$z = h(x)$$

Classification
Logistic Regression
التصنيف

Logistic Regression

In spite of its name, logistic regression is a model for classification not regression.

Used to model the probability of the class of x via a linear function.

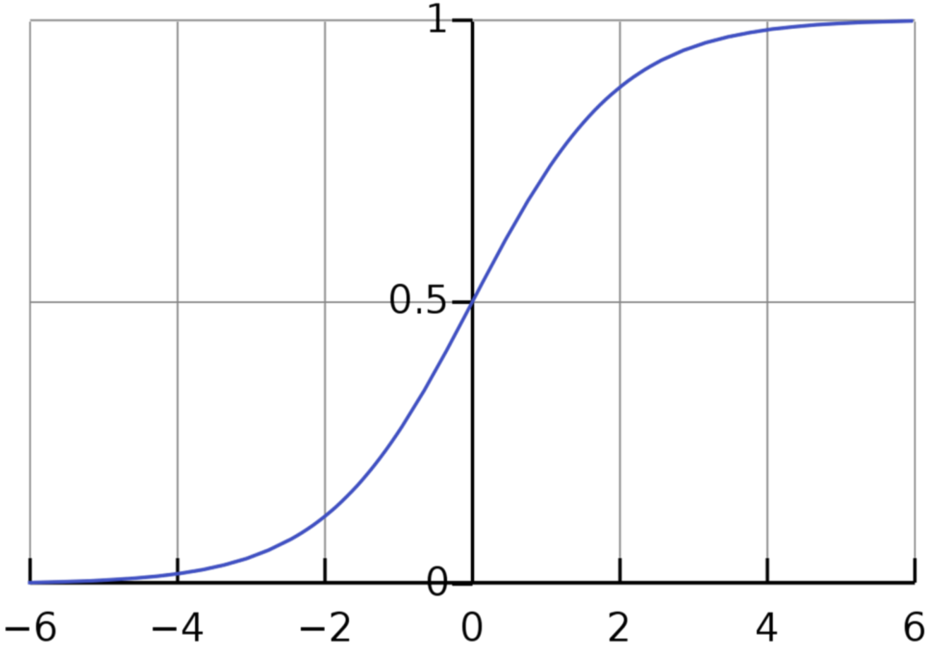
It can be interpreted as the probability that an input (X) belongs to the default class (Y=1).

The probability of x belong to a class is between 0 and 1.

Sigmoid Function

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

$$z = h(x) = \theta_j^T . x_i$$



It's an S-shaped curve that can take any real-valued number and map it into a value between 0 and 1

Probability and Odds

$$p(occurring) = \frac{outcome\ of\ interest}{all\ possible\ outcome}$$

$$odds = \frac{p(occurring)}{p(not\ occurring)}$$

$$odds = \frac{p}{1-p}$$

$$p(head) = \frac{1}{2} = 0.5$$

$$odds(heads) = \frac{0.5}{0.5} = 1 \quad or \ \ 1:1$$

$$p(1\ or\ 2) = \frac{2}{6} = 0.333$$

$$odds(1\ or\ 2) = \frac{0.333}{0.666} = \frac{1}{2} = 0.5 \quad or\ 1:2$$

$$\ln(odds) = ln\left(\frac{p}{1-p}\right) = logit(p)$$

Logistic Regression

$$ln(odds) = ln\left(\frac{p}{1-p}\right) = Logit(p)$$

$0 < x < 1$

$$logistic(x) \; = logit(x)^{-1}$$

$$Logistic(x) = \left(ln\left(\frac{x}{1-x}\right)\right)^{-1}$$

$0 < h(x) < 1$

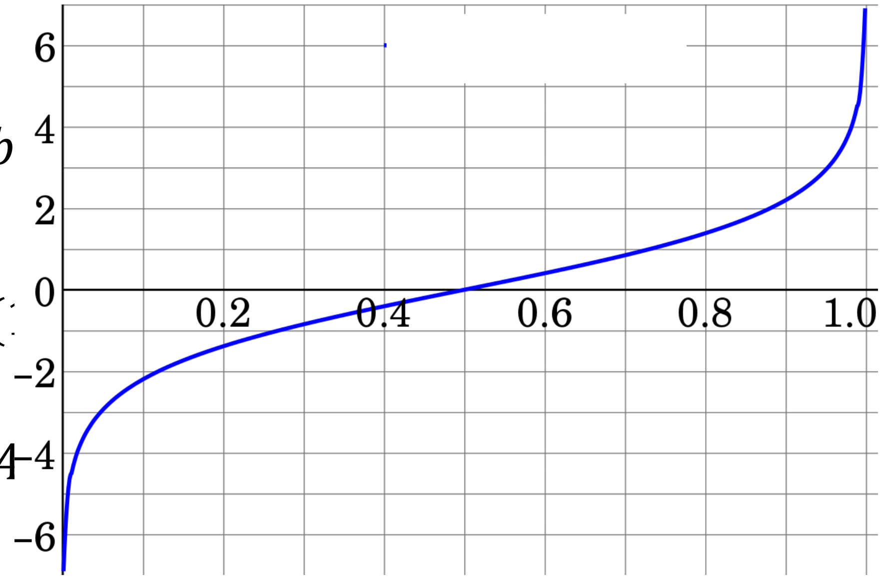
$$\boxed{Logistic(x) = \frac{1}{1+e^{-x}}}$$

Sigmoid Function

$log_a(b$

$log_{10}($

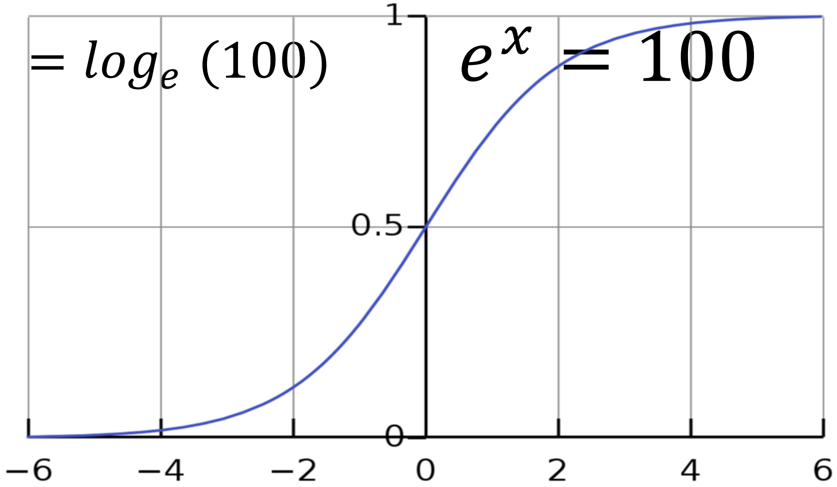
$log_7(4$



Euler's Nmber

$$ln(b) \; = log_e\ (b) \qquad e = 2.718$$

$ln(100) \; = log_e\ (100)$ $\quad e^x = 100$



Example:

$$h(x) = -2x + 6$$

$$z = h(x)$$

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

$p(y = 1|x; \theta)$

Estimated probability

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

0: benign
1: malignant

$x: tumor\ size$

x	y
3	1
2	1
1	1
5	0
4	0
6	0

$when\ z \geq 0$

$Then\ g(z) \geq 0.5$

$when\ z < 0$

$Then\ g(z) < 0.5$

$e = 2.718$

x	z	e^{-z}		$1 + e^{-(z)}$	g(z)	y
3	0	e^0	1	2	0.5	1
2	2	$\frac{1}{e^2}$	0.13536335	1.13536335	0.8808	1
1	4	$\frac{1}{e^4}$	0.018323237	1.018323237	0.9820	1
5	-4	e^4	54.57551085	55.57551085	0.0179	0
4	-2	e^2	7.387524	8.387524	0.1192	0
6	-6	e^6	403.1778962	404.1778962	0.00247	0

Logistic Regression
Decision Boundary

Decision Boundary

$$h(x) = \theta_j^T . x_i$$

$$h(x) = z$$

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

$$\mathrm{g}(z) = p(y = 1|x; \theta)$$

$p(y = 1|x; \theta) \geq 0.5$ Threshold

$$when\ z \geq 0$$

$$Then\ p(y = 1|x; \theta) \geq 0.5$$

$\theta_j^T . x_i \geq 0$ Decision Boundary

$$h(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

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

$$h(x) = -3 + x_1 + x_2$$

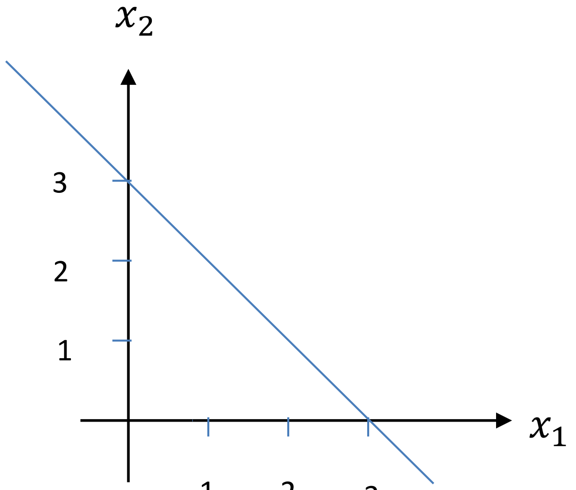
$$predicty = 1, if - 3 + x_1 + x_2 \geq 0$$

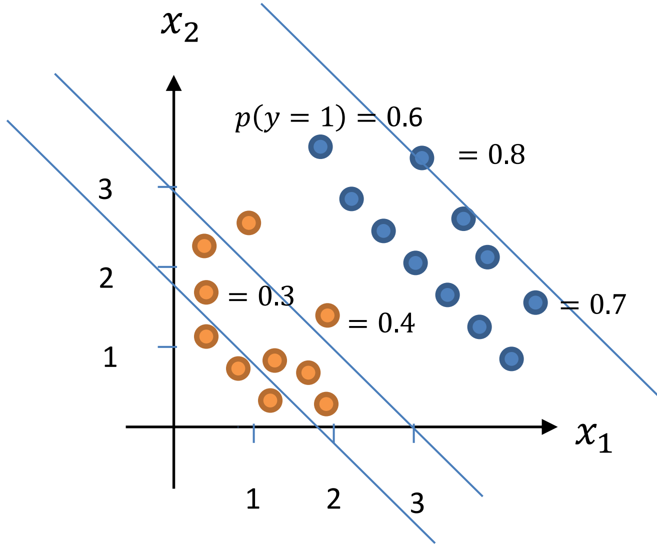
$$x_1 + x_2 \geq 3$$

$$x_1 + x_2 = 3$$

$$\theta_j = \begin{bmatrix} -4 \\ 1 \\ 1 \end{bmatrix}$$

$$h(x) = -4 + x_1 + x_2$$



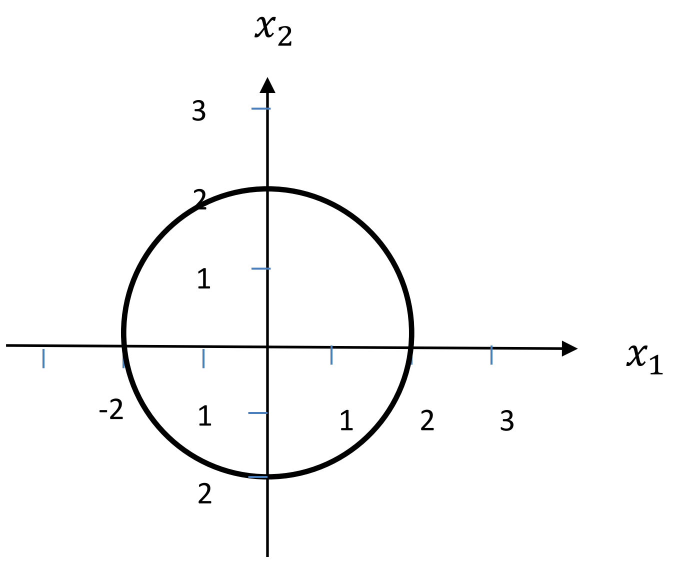


Decision boundary can be:

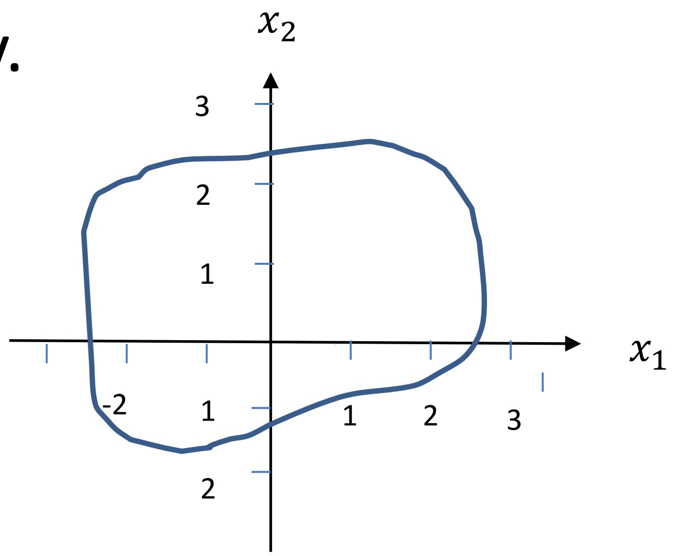
- Linear (a line).
- Higher order polynomial.
- Non-linear.

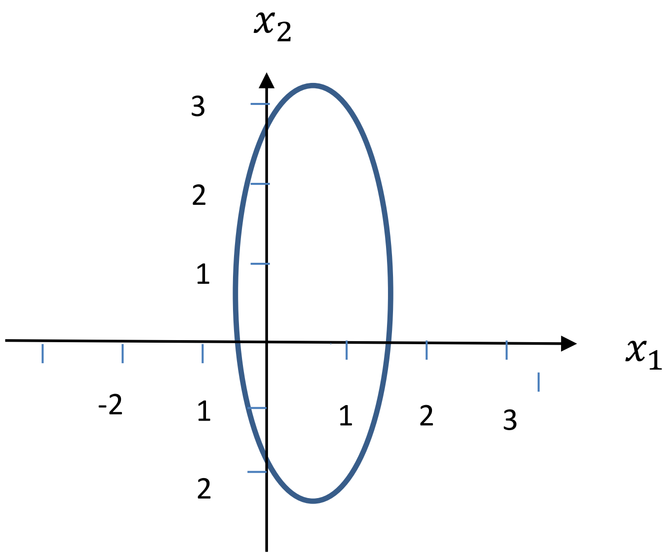
$$h(x) = x_1^2 + x_2^2 - 2$$

$$x_1^2 + x_2^2 \geq 2$$



- complex decision boundary.





Logistic Regression
Cost Function

In Linear regression, to find optimal values for θs:

1. Choose a cost function $$j(\theta) = \frac{1}{2}\sum_{i=1}^{m}(h(x^{(i)}) - y^{(i)})^2$$

2. Apply Gradient Descent $$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

$h(x) = \theta^T x$ is a linear function

What if we choose the same cost function
for Logistic regression?

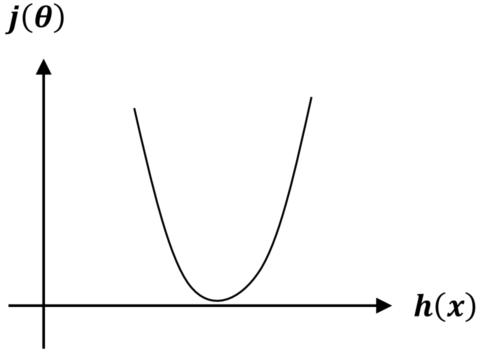
$$g(z) = \frac{1}{1 + e^{-(\theta^T x)}}$$

is a non-linear function

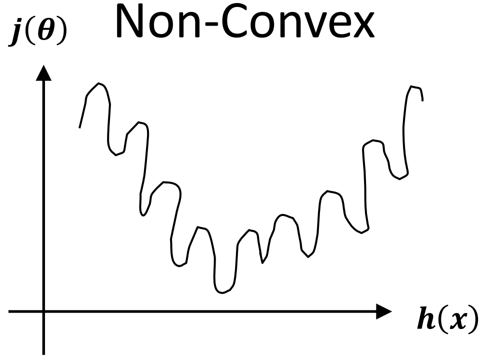
$$z = h(x)$$

Can we find a convex cost function?

Convex

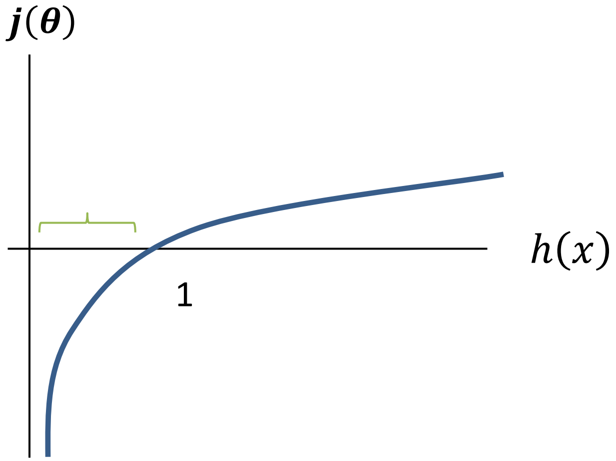


Non-Convex



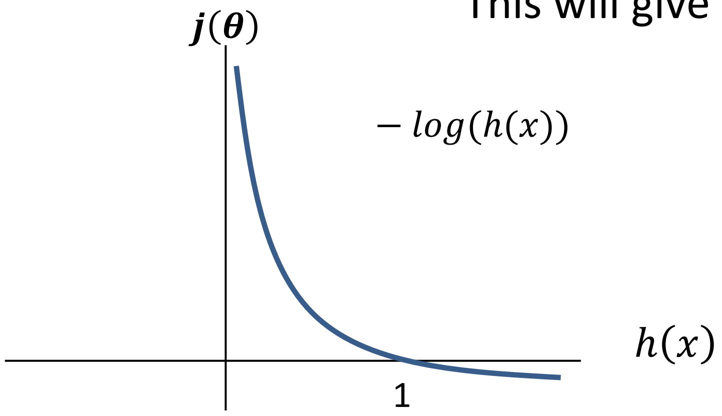
$$j(\theta) = cost(h(x), y)$$

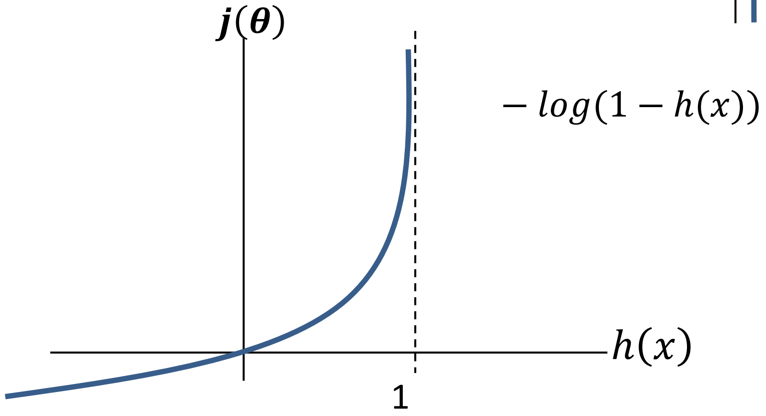
Actual y	Predicted h(x)	$j(\theta)$
1	0.9	- 0.105



$$cost(h(x), y) = \begin{cases} -\log(h(x)), & if\ y = 1 \\ -\log(1 - h(x)), & if\ y = 0 \end{cases}$$

This will give us the convexity





Actual y	Predicted h(x)	$j(\theta)$
1	0.9	0.105
1	0.5	0.693
1	0.1	2.302
1	0.01	4.605

Actual y	Predicted h(x)	$1 - h(x)$	$j(\theta)$
0	0.9	0.1	2.302
0	0.5	0.5	0.693
0	0.1	0.9	0.105
0	0.01	0.99	0.01

Logistic Regression Cost Function

$$cost(h(x), y) = \begin{cases} -\log(h(x)), & if\ y = 1 \\ -\log(1 - h(x)), & if\ y = 0 \end{cases}$$

IF/Then version

$$j(\theta) = y * -\log\big(h(x)\big) - (1 - y) * \log(1 - h(x))$$

$$j(\theta) = -y\log\big(h(x)\big) - (1 - y)\log(1 - h(x))$$

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

Binary Cross-Entropy Cost Function

Logistic Regression
Gradient Descent

Logistic Regression

Step 1: $h(x) = \theta_i^T . x_i$ Learning Model Linear classifier

$z = h(x)$

Step 2: $g(z) = \frac{1}{1 + e^{-(z)}}$ convert a real value into one that can be interpreted as a probability

Step 3: $j(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log\left(g\left(z^{(i)}\right)\right) + \left(1 - y^{i}\right)\log(1 - g(z^{(i)}))\right]$ Binary Cross-Entropy Cost Function

Step 4: $minimize\ (j(\theta))$ Using an optimization algorithm

Gradient Descent

$$\boxed{\theta_j := \theta_j - \gamma \frac{d}{d\theta_j} J(\theta)}$$

Gradient Descent

Linear Regression

Logistic Regression

$$\boxed{\boldsymbol{\theta_j := \theta_j - \gamma \frac{d}{d\theta_j} J(\theta)}}$$

$$\frac{d}{d\theta_j} j(\theta) = \sum_{i=1}^{m} \left(h(x)^i - y^i\right).\, x_j^i$$

$$\frac{d}{d\theta_j} j(\theta) = \frac{1}{m} \sum_{i=1}^{m} \left(g(z)^i - y^i\right).\, x_j^i$$

$$\boldsymbol{\theta_j := \theta_j - \gamma}$$

Repeat

Batch Gradient Descent

Stochastic Gradient Descent (SGD)

$for\ i = 1,2, \dots, m:$

$$\boldsymbol{\theta_j := \theta_j - \gamma}\left(g(z)^i - y^i\right).\, x_j^i$$

Gradient Descent

Linear Regression

Logistic Regression

$$\boxed{\boldsymbol{\theta_j := \theta_j - \gamma \frac{d}{d\theta_j} J(\theta)}}$$

$$\frac{d}{d\theta_j} j(\theta) = \sum_{i=1}^{m} \left(h(x)^i - y^i\right) . x_j^i$$

$$\frac{d}{d\theta_j} j(\theta) = \frac{1}{m} \sum_{i=1}^{m} \left(g(z)^i - y^i\right) . x_j^i$$

$$\boldsymbol{\theta_j := \theta_j - \gamma} \ \frac{1}{m} \sum_{i=1}^{m} \left(g(z)^i - y^i\right) . x_j^i$$

Repeat

Batch Gradient Descent

Stochastic Gradient Descent (SGD)

$for\ i = 1,2, \dots, m:$

$$\boldsymbol{\theta_j := \theta_j - \gamma}\left(g(z)^i - y^i\right). x_j^i$$

Logistic Regression
Gradient Descent
Cross Entropy Function Derivative

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

$$j(\theta) = -y\log\big(g(z)\big) - (1-y)\log(1-g(z))$$

$$j(\theta) = \begin{cases} -\,log(g(z)), & if\ y = 1 \\ -\,log(1-g(z)), & if\ y = 0 \end{cases}$$

$$\frac{d}{d\theta_j} j(\theta) = \begin{cases} -\frac{1}{g(z)} . \frac{d}{d\theta_j} g(z) & if\ y = 1 \\ -\frac{1}{1-g(z)} . \frac{d}{d\theta_j} (-g(z)) & if\ y = 0 \end{cases}$$

$$\frac{d}{d\theta_j} j(\theta) = -y \frac{1}{g(z)} . \frac{d}{d\theta_j} g(z) - (1-y) \frac{1}{1-g(z)} . \frac{d}{d\theta_j} (-g(z))$$

$$\frac{d}{d\theta_j} j(\theta) = \left(-\frac{y}{g(z)} + \frac{(1-y)}{1-g(z)}\right) . \frac{d}{d\theta_j} g(z)$$

$$\frac{d}{d\theta_j} g(z) = \frac{d}{d\theta_j} \frac{1}{1+e^{-z}}$$

$$\frac{d}{d\theta_j} g(z) = \frac{d}{d\theta_j} (1+e^{-z})^{-1}$$

$$\frac{d}{d\theta_j} g(z) = -(1+e^{-z})^{-2} . \frac{d}{d\theta_j} (1+e^{-z})$$

$$\frac{d}{d\theta_j} g(z) = -(1+e^{-z})^{-2} . \frac{d}{d\theta_j} e^{-z}$$

$$\frac{d}{d\theta_j} g(z) = -(1+e^{-z})^{-2} . e^{-z} . \frac{d}{d\theta_j} - \mathrm{z}$$

$$\frac{d}{d\theta_j} g(z) = (1+e^{-z})^{-2} . e^{-z} . \frac{d}{d\theta_j} \left[\theta^T . x_j\right]$$

$$\frac{d}{d\theta_j} g(z) = (1+e^{-z})^{-2} . e^{-z} . x_j$$

$$\frac{d}{d\theta_j} j(\theta) = \left(-\frac{y}{g(z)} + \frac{(1-y)}{1-g(z)}\right) . \frac{d}{d\theta_j} g(z)$$

$$\frac{d}{d\theta_j} j(\theta) = \left(-\frac{y}{g(z)} + \frac{(1-y)}{1-g(z)}\right)$$

$$\frac{d}{d\theta_j} j(\theta) = \left(-y\left(1-g(z)\right) + (1-y)g(z)\right) . x_j$$

$$\frac{d}{d\theta_j} j(\theta) = (-y + y.g(z) + g(z) - y.g(z)) . x_j$$

$$\frac{d}{d\theta_j} j(\theta) = (-y + (g(z)) . x_j$$

$$\frac{d}{d\theta_j} g(z) = (1 + e^{-z})^{-2} . e^{-z} . x_j$$

$$\frac{d}{d\theta_j} g(z) = \frac{e^{-z}}{(1 + e^{-z})^2} . x_i$$

$$= \frac{1}{1+e^{-z}} . \left(1 - \frac{1}{1+e^{-z}}\right) . x_j$$

$$\frac{d}{d\theta_j} g(z) = g(z) . \left(1 - g(z)\right) . x_j$$

$$\frac{d}{d\theta_j} j(\theta) = \frac{1}{m} \sum_{i=1}^{m} \left(g(z)^i - y^i\right) . x_j^i$$

$$\boxed{\boldsymbol{\theta_j} := \boldsymbol{\theta_j} - \boldsymbol{\gamma} \frac{1}{m} \sum_{i=1}^{m} \left(g(z)^i - y^i\right) . x_j^i}$$

Logistic Regression
Multi-class Classification

Multiclass Classification

Binary Classification

$\{x_i, y\}^m$ $y \epsilon \{0,1\}$

Multiclass Classification

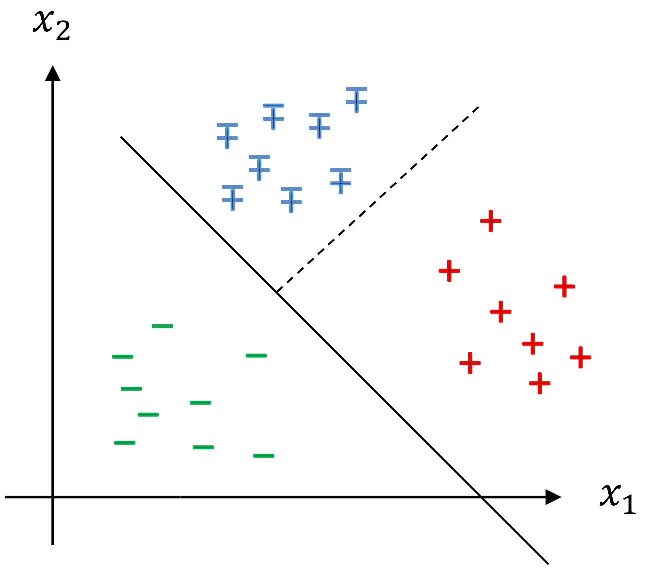
$\{x_i, y\}^m$ $y \epsilon \{1, 2, 3, \ldots N\}$

Covid-19

Positive $+$

Negative $-$

Positive/No symptoms $\mp$

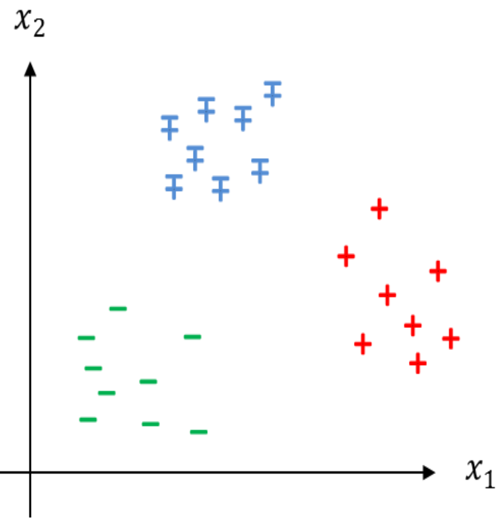


Two techniques:

- One vs. All
- One vs. One

One vs. All

N-binary classifiers



Features		Class
x1	x2	-
x1	x2	+
x1	x2	+
x1	x2	∓

Training set for +

Features		Class
x1	x2	0
x1	x2	1
X1	x2	1
x1	x2	0

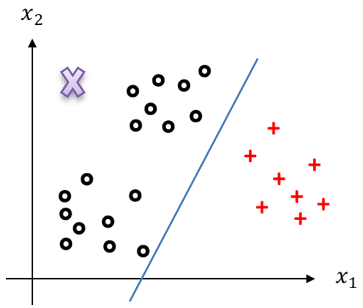
x1	**x2**	

Training Binary Classifier 1

$\theta^T.x_j$	g(z)	GD

Classification model 1

$p(y = 1|x;\theta)$



Training set for -

Features		Class
x1	x2	1
x1	x2	0
x1	x2	0
x1	x2	0

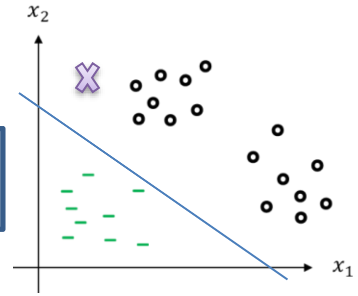
x1	**x2**	

Training Binary Classifier 2

$\theta^T.x_j$	g(z)	GD

Classification model 2

$p(y = 1|x;\theta)$



Training set for ∓

Features		Class
x1	x2	0
x1	x2	0
x1	x2	0
x1	x2	1

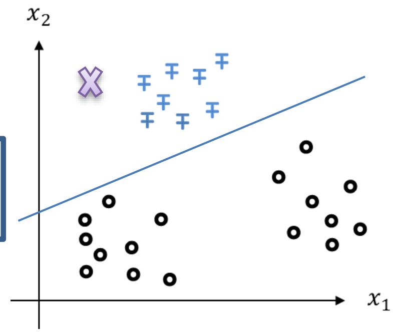
x1	**x2**	

Training Binary Classifier 3

$\theta^T.x_j$	g(z)	GD

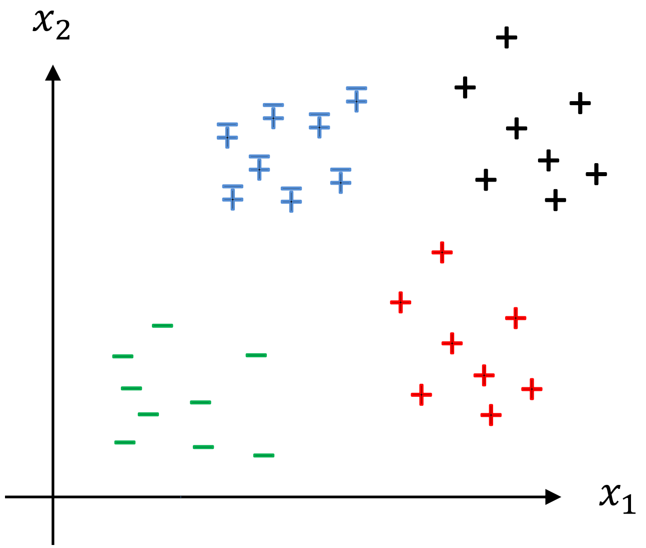
Classification model 3

$p(y = 1|x;\theta)$



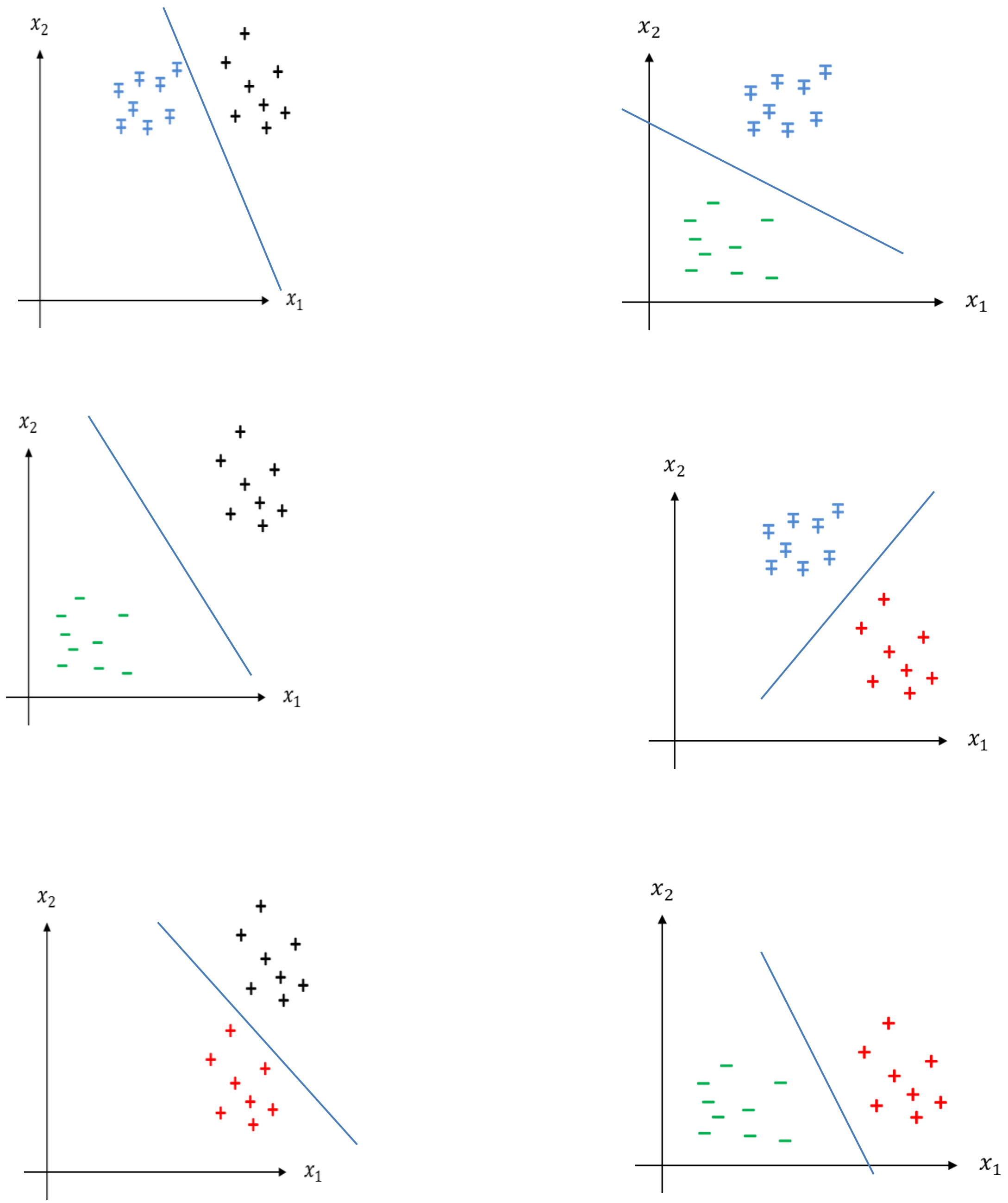
Predicted y= Max (, ,)

One vs. One (OvO)



Binary classifier

$$N * (N - 1)/2$$



This link of that tutorial can be a good starting point for beginners, I will use the “**titanic dataset**” from the famous kaggle competition

https://www.kaggle.com/c/titanic/overview

https://towardsdatascience.com/machine-learning-with-python-classification-complete-tutorial-d2c99dc524ec

Machine Learning with Python: Classification (complete tutorial) | by Mauro Di Pietro | Towards Data Science

The End