Confusion Matrix

Sklearn Representation

Scikit learn documentation says — Wikipedia and other references may use a different convention for axes.

A)

Predicted Label \ Actual Label	1	0
1	TP	FP
0	FN	TN

B)

Predicted Label \ Actual Label	0	1
0	TN	FN
1	FP	TP

C)

Actual Label \ Predicted Label	1	0
1	TP	FN
0	FP	TN

D)

Actual Label \ Predicted Label	0	1
0	TN	FP
1	FN	TP

https://towardsdatascience.com/understanding-the-confusion-matrix-from-scikit-learn-c51d88929c79

F_β-Score

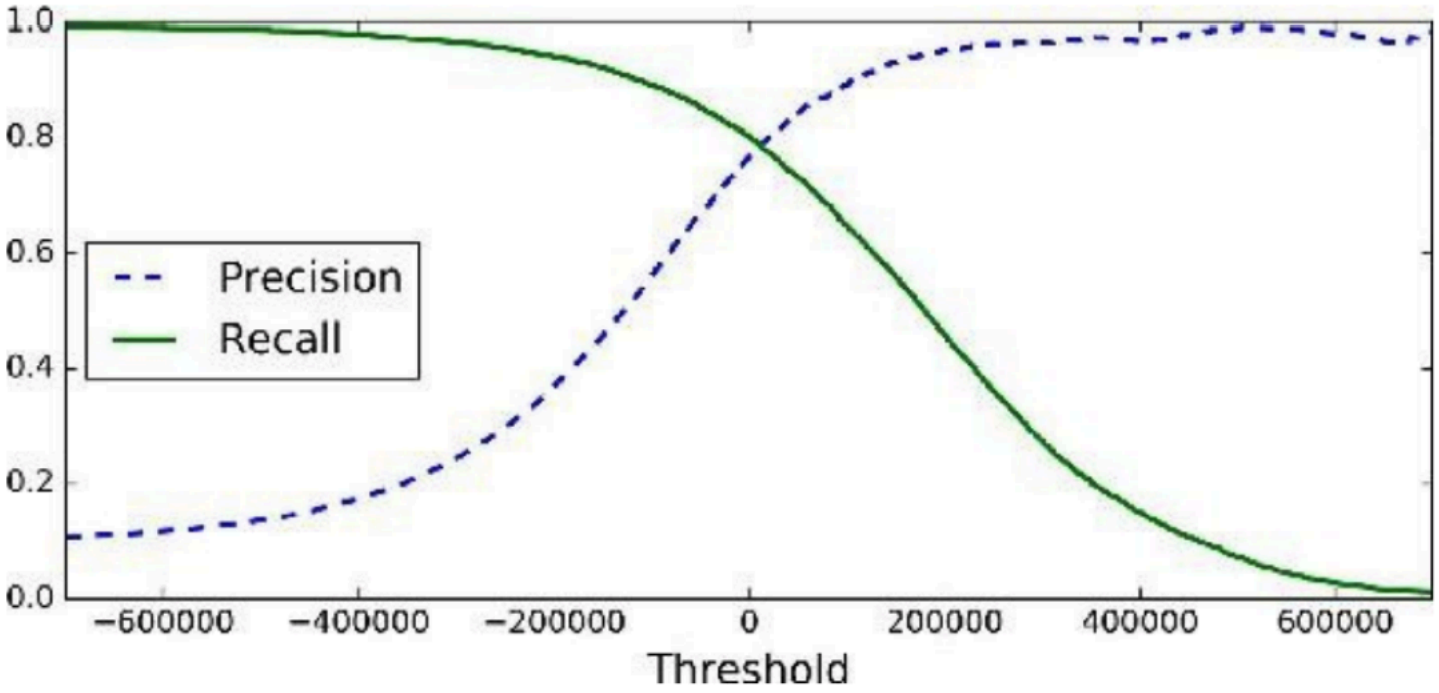
$$F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}}$$

$\beta = 1$

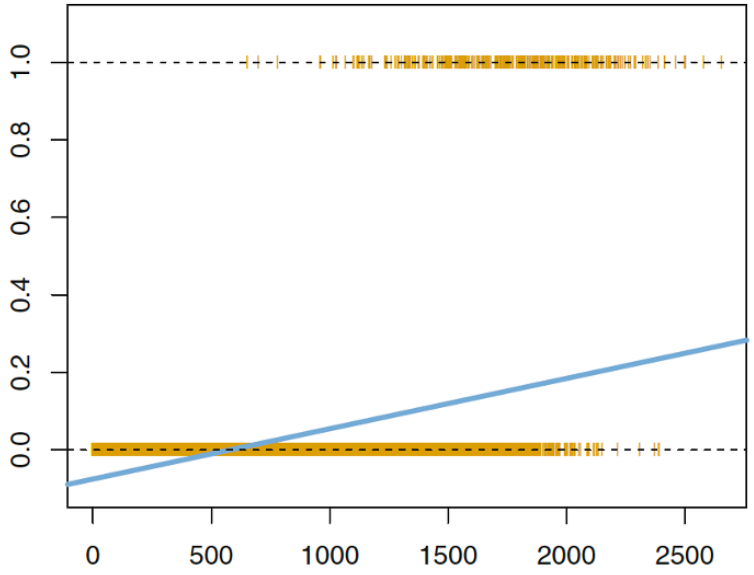
$$F_1 = \frac{2}{\text{recall}^{-1} + \text{precision}^{-1}} = 2\frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{\text{tp}}{\text{tp} + \frac{1}{2}(\text{fp} + \text{fn})}$$

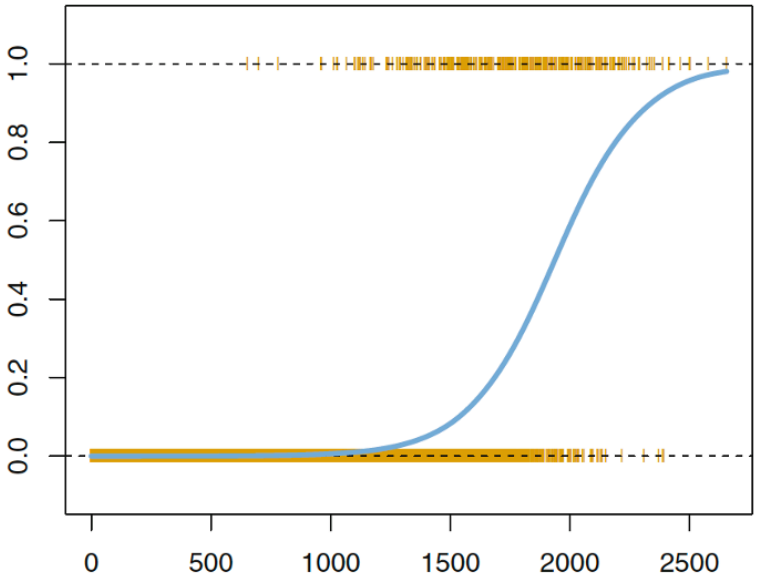
harmonic mean

Precision – Recall Trade off



Logistic Regression





$$p(X) = \beta_0 + \beta_1 X$$

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}}$$

Logistic Regression

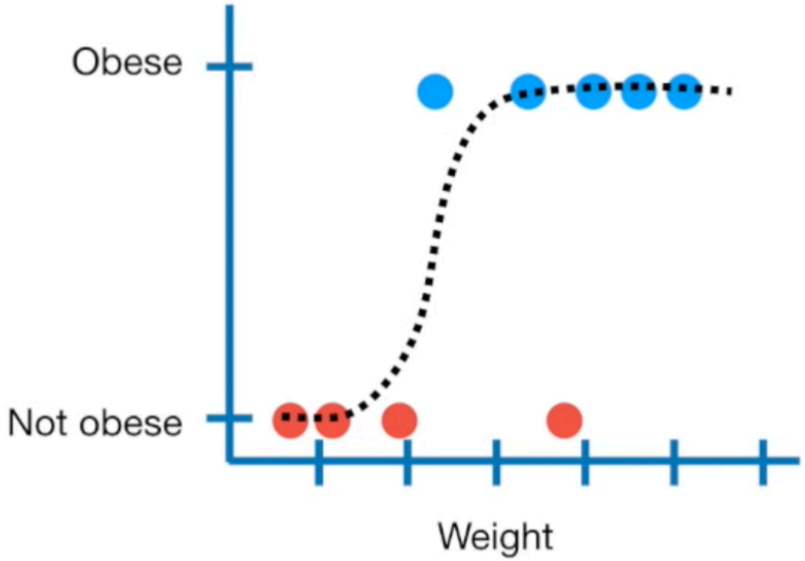
$$h_\theta(x) = \frac{1}{1 + e^{-(\theta^T x)}} = \frac{1}{1 + e^{-(\theta_0 + \theta_1 x)}}$$

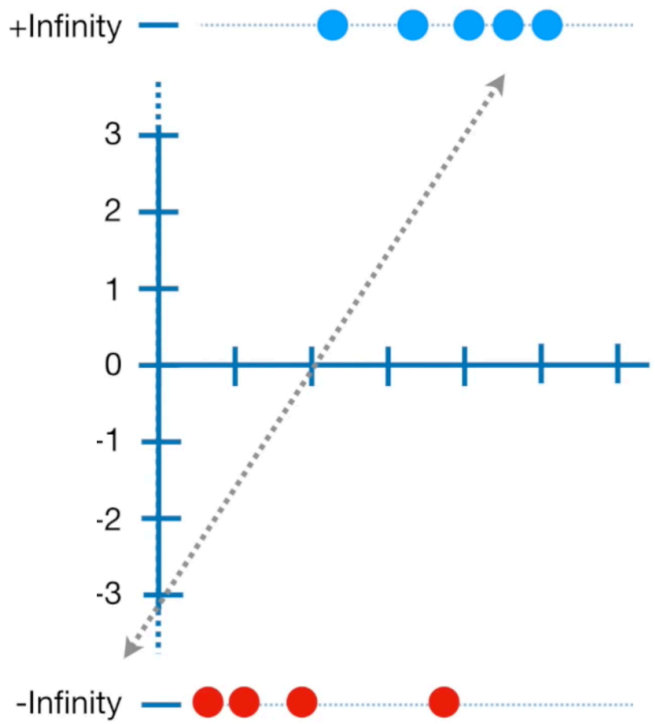
$$\frac{h_\theta(x)}{1 - h_\theta(x)} = e^{(\theta_0 + \theta_1 x)}$$

$$\log\left(\frac{h_\theta(x)}{1 - h_\theta(x)} \right) = \theta_0 + \theta_1 x$$

(log odds)

Logistic Regression

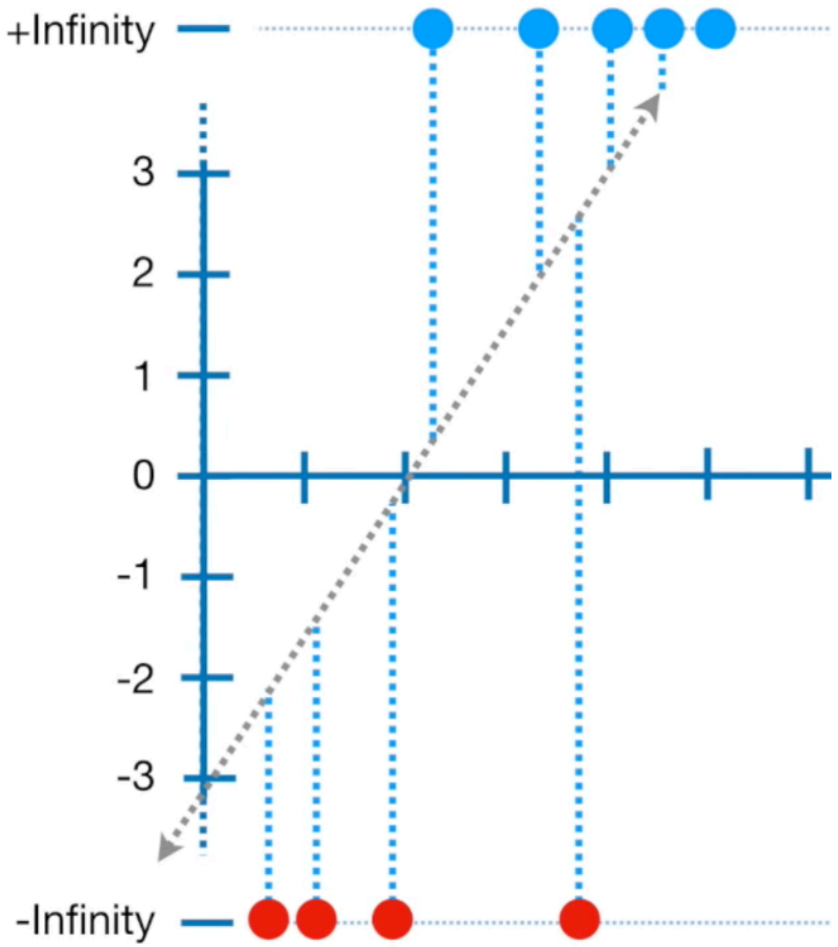


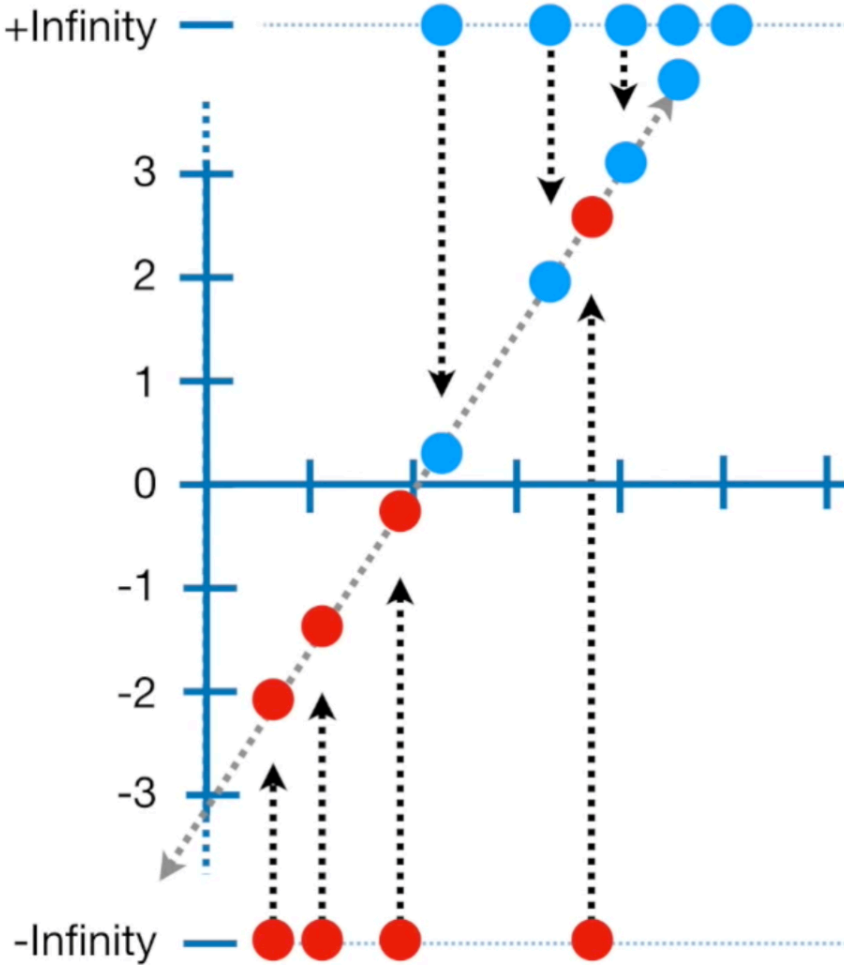


$$h_\theta(x) = \frac{1}{1 + e^{-(\theta_0 + \theta_1 x)}}$$

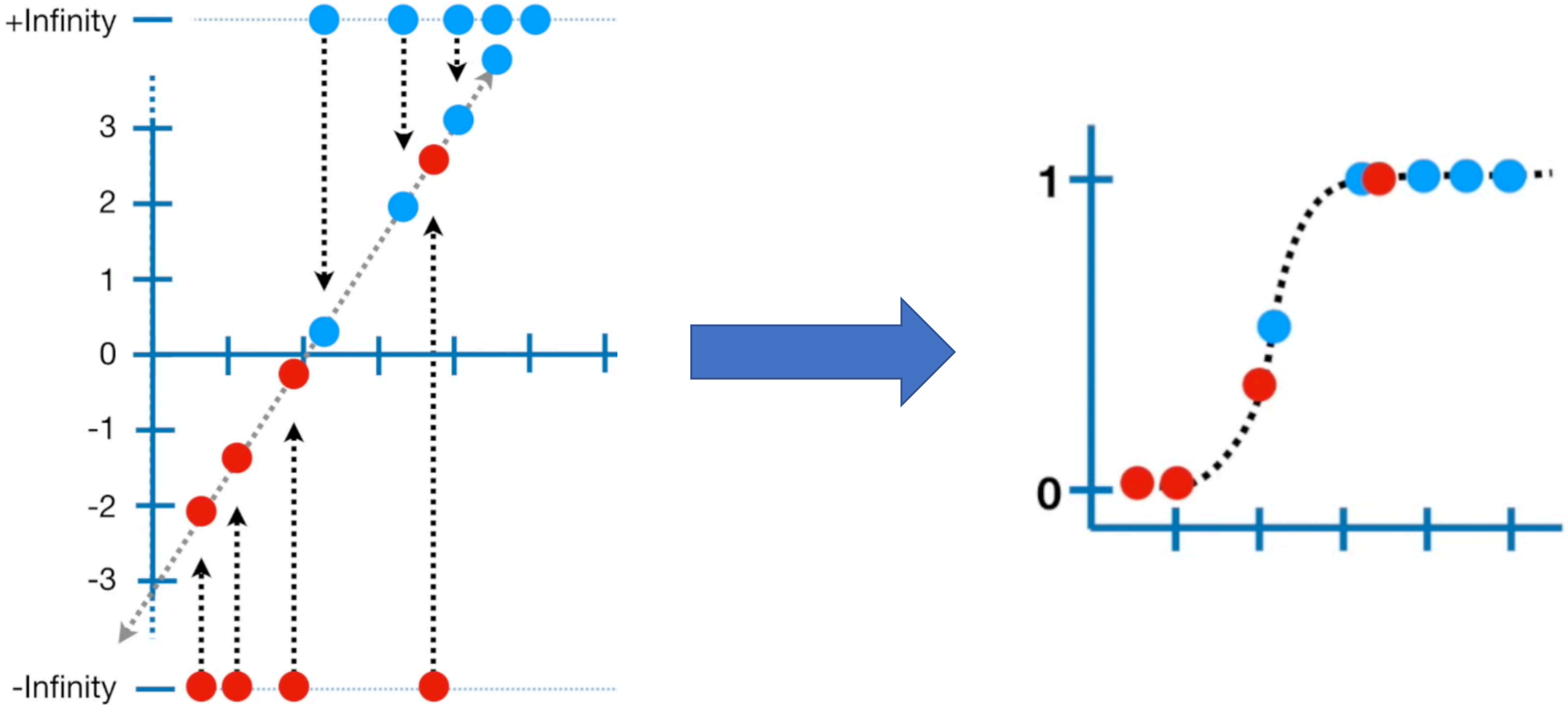
$$\log\left(\frac{h_\theta(x)}{1 - h_\theta(x)}\right) = \theta_0 + \theta_1 x$$

Logistic Regression



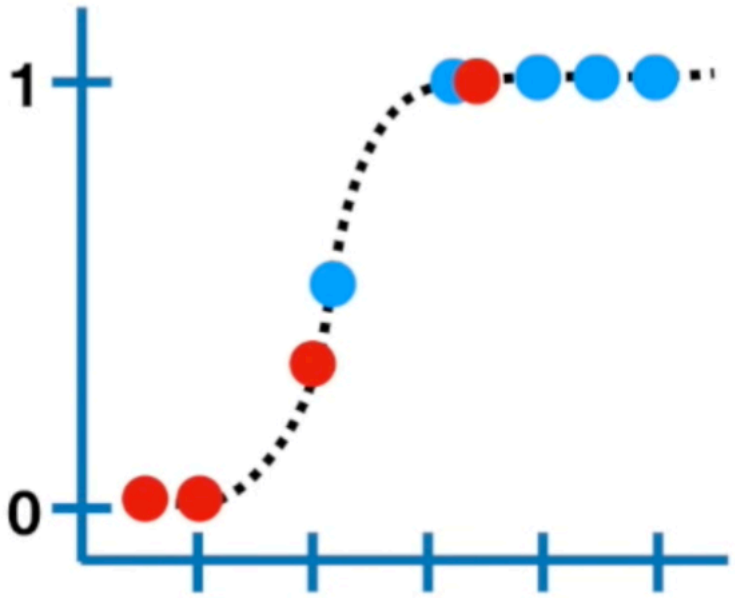


Logistic Regression



Logistic Regression

MLE – maximum likelihood estimation:



Likelihood:

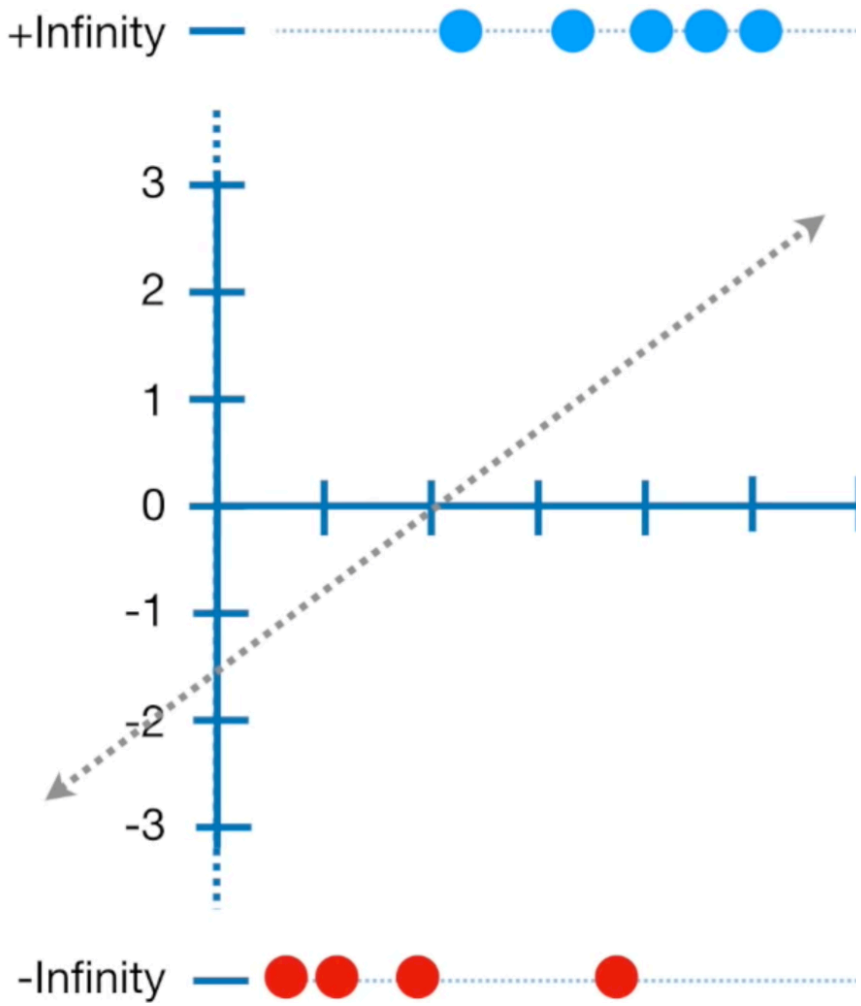
$$\ell(\beta_0, \beta_1) = \prod_{i:y_i=1} p(x_i) \prod_{i':y_{i'}=0} (1 - p(x_{i'}))$$

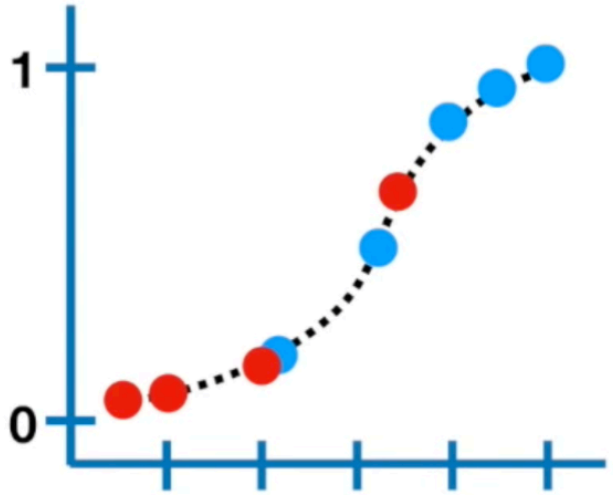
Log-likelihood:

$$\log \ell(\beta_0, \beta_1) = \sum_{i:y_i=1} \log p(x_i) + \sum_{i':y_{i'}=0} \log(1 - p(x_{i'}))$$

$$= \log(0.49) + \log(0.9) + \log(0.91) + \log(0.91) + \log(0.92) + \log(1 - 0.9) + \log(1 - 0.3) + \log(1 - 0.01) + \log(1 - 0.01)$$

Logistic Regression





$$= \log(0.22) + \log(0.4) + \log(0.8) + \log(0.89) + \log(0.92) + \log(1 - 0.6) + \log(1 - 0.2) + \log(1 - 0.1) + \log(1 - 0.05)$$

Logistic Regression – Loss function

Linear Regression:

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

$$\text{Cost}(h_\theta(x^{(i)}), y^{(i)}) = \frac{1}{2} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

non – convex

Logistic Regression:

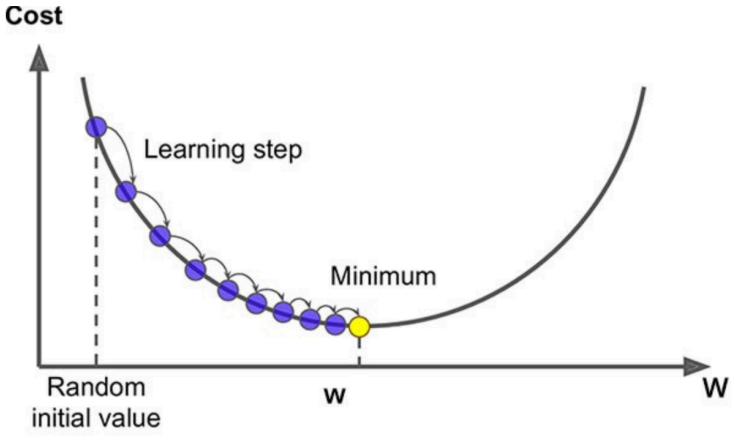
$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

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

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

cross-entropy

Gradient prosty *Gradient descent*



$$\theta_j := \theta_j - \eta \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

$\eta - learning\ rate\ (0.01)$

$$X = \begin{bmatrix} x_{1,1} & \cdots & x_{1,n} \\ x_{2,1} & & x_{2,n} \\ \vdots & \ddots & \vdots \\ x_{m,1} & \cdots & x_{m,n} \end{bmatrix}$$

$$\boldsymbol{x_i} = [x_{i,1} \quad x_{i,2} \quad \cdots \quad x_{i,n}]$$

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} (y_i \log(h_\theta(x_i)) - (1 - y_i) \log(1 - h_\theta(x_i)))$$

$$h_\theta(x_i) = \boldsymbol{g}(\boldsymbol{\theta^T x_i})$$

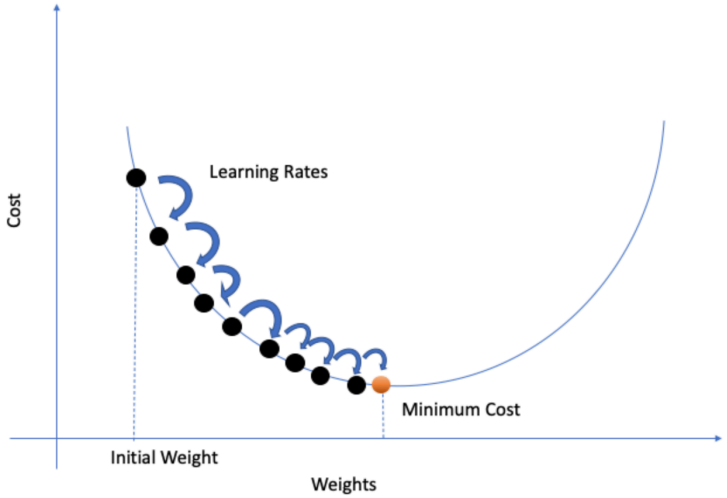
$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x_i) - y_i\right) x_{i,j}$$

```
eta = 0.1  # learning rate
n_iterations = 1000
m = 100

theta = np.random.randn(2,1)  # random initialization

for iteration in range(n_iterations):
    gradients = 2/m * X_b.T.dot(X_b.dot(theta) - y)
    theta = theta - eta * gradients
```

Stochastyczny Gradient *Stochastic Gradient Descent*



$$\theta_j := \theta_j - \eta \frac{\partial}{\partial \theta_j} J_i(\theta_0, \theta_1)$$

$\eta - learning\ rate\ (0.01)$

$$X = \begin{bmatrix} x_{1,1} & \cdots & x_{1,n} \\ x_{2,1} & & x_{2,n} \\ \vdots & \ddots & \vdots \\ x_{m,1} & \cdots & x_{m,n} \end{bmatrix}$$

$$\boldsymbol{x_i} = [x_{i,1} \quad x_{i,2} \quad \cdots \quad x_{i,n}]$$

$i - ustalone,\ losowe$

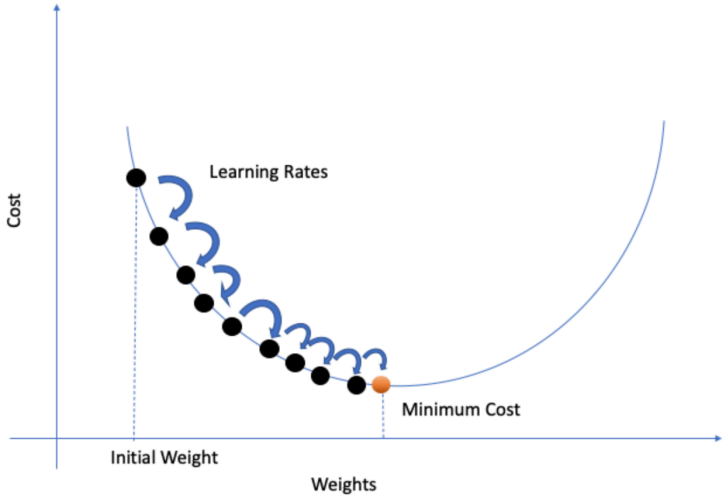
$$J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} (y_i \log(h_\theta(x_i)) - (1 - y_i) \log(1 - h_\theta(x_i)))$$

$$h_\theta(x_i) = \boldsymbol{g}(\boldsymbol{\theta^T x_i})$$

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x_i) - y_i\right) x_{i,j}$$

```
for epoch in range(n_epochs):
    for i in range(m):
        random_index = np.random.randint(m)
        xi = X_b[random_index:random_index+1]
        yi = y[random_index:random_index+1]
        gradients = 2 * xi.T.dot(xi.dot(theta) - yi)
        eta = learning_schedule(epoch * m + i)
        theta = theta - eta * gradients
```

Gradient z minigrupami *Mini-batch Gradient Descent*



$$\theta_j := \theta_j - \eta \frac{\partial}{\partial \theta_j} J_I(\theta_0, \theta_1)$$

$\eta - learning\ rate\ (0.01)$

$$X = \begin{bmatrix} x_{1,1} & \cdots & x_{1,n} \\ x_{2,1} & & x_{2,n} \\ \vdots & \ddots & \vdots \\ x_{m,1} & \cdots & x_{m,n} \end{bmatrix}$$

$$\boldsymbol{x_i} = [x_{i,1} \quad x_{i,2} \quad \cdots \quad x_{i,n}]$$

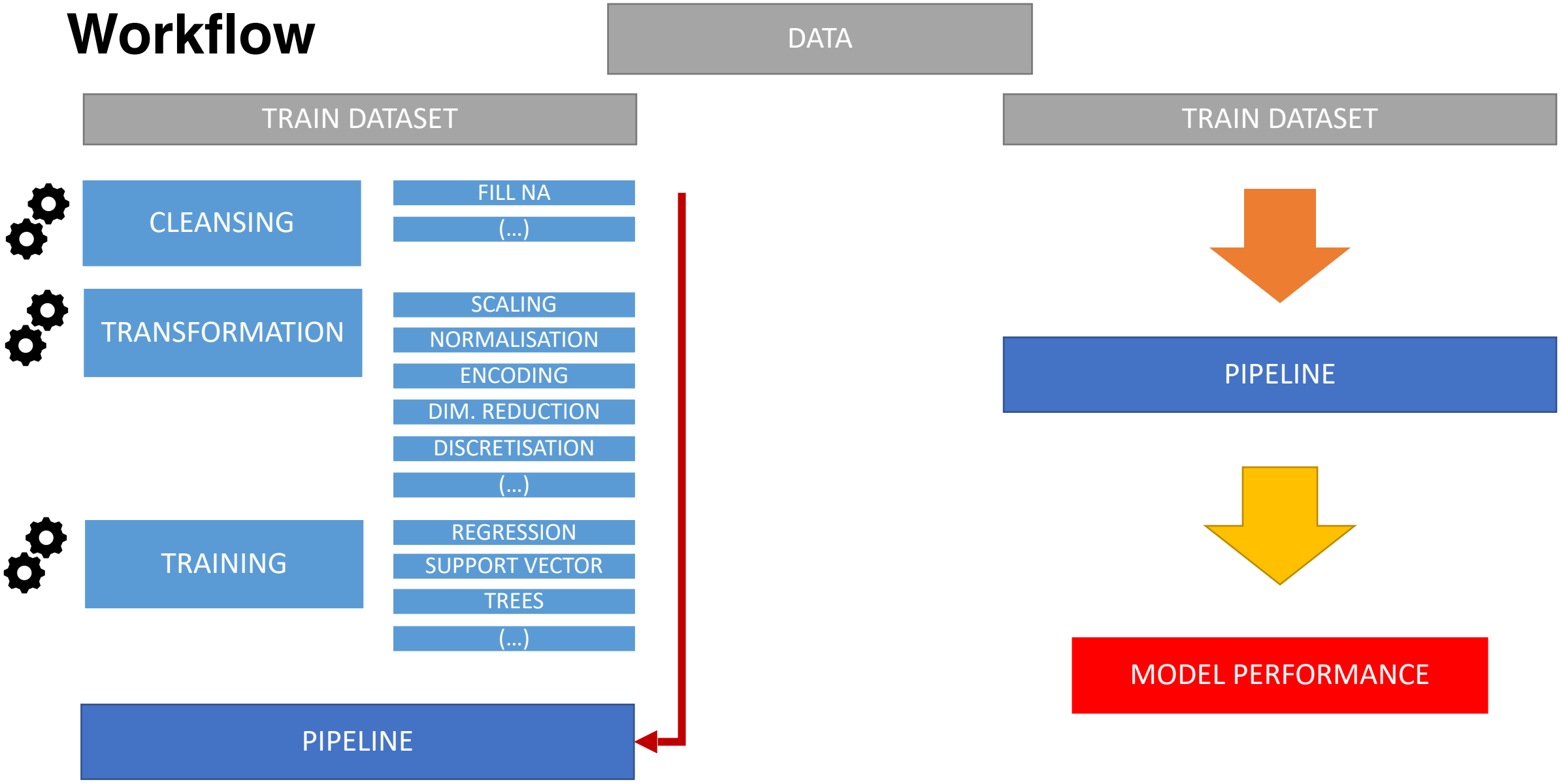
$I - ustalone,\ losowy\ podzbiór\ X$

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} (y_i \log(h_\theta(x_i)) - (1 - y_i) \log(1 - h_\theta(x_i)))$$

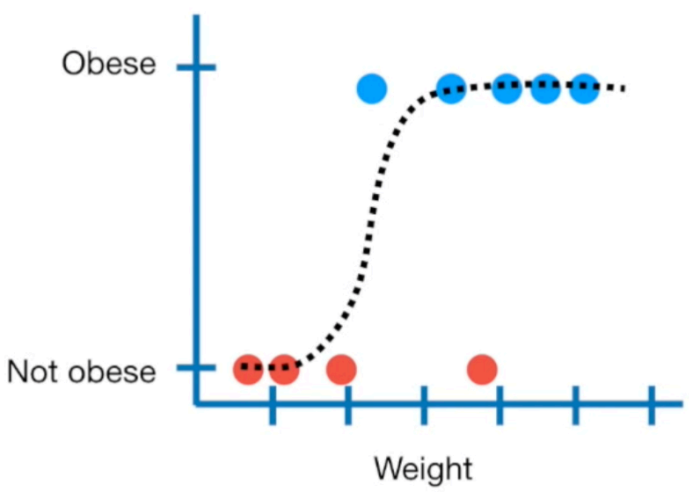
$$h_\theta(x_i) = \boldsymbol{g}(\boldsymbol{\theta^T x_i})$$

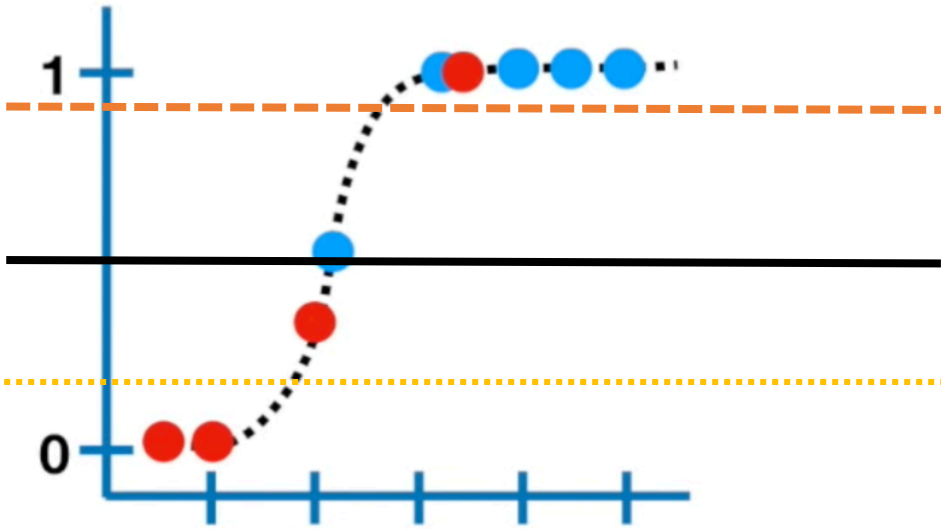
$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x_i) - y_i\right) x_{i,j}$$

Workflow


DATA
TRAIN DATASET
CLEANSING
FILL NA
(...)
TRANSFORMATION
SCALING
NORMALISATION
ENCODING
DIM. REDUCTION
DISCRETISATION
(...)
TRAINING
REGRESSION
SUPPORT VECTOR
TREES
(...)
PIPELINE
TRAIN DATASET
PIPELINE
MODEL PERFORMANCE

ROC-AUC



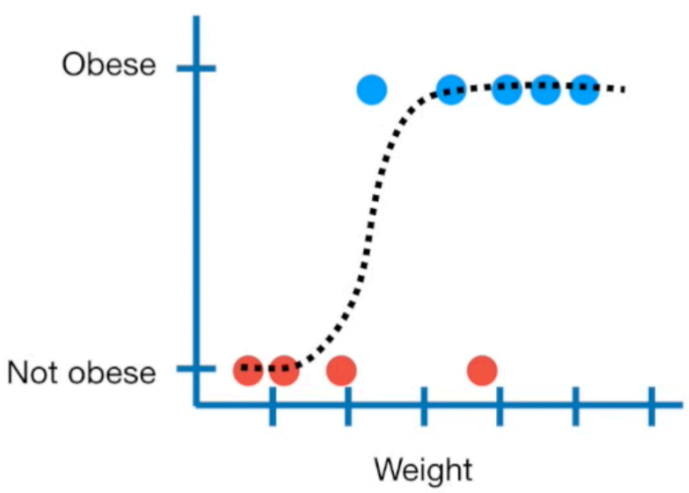


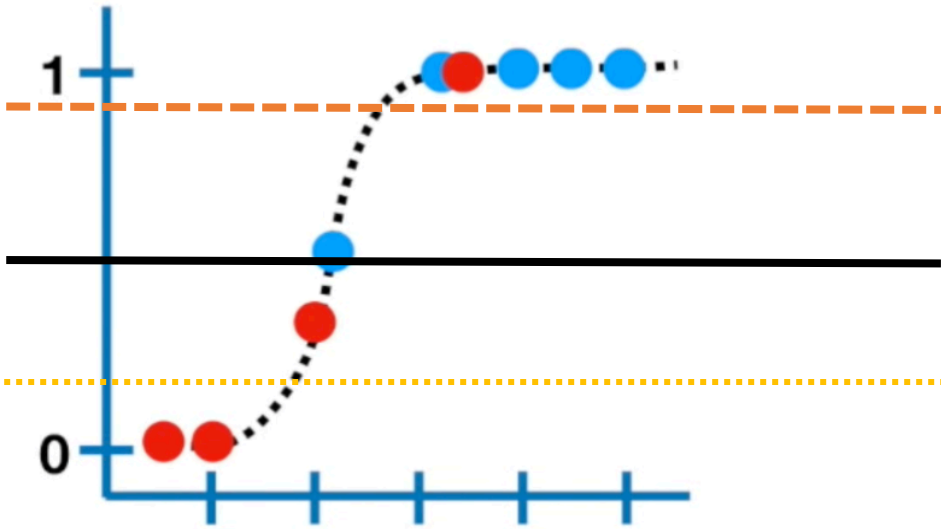
PREDICTIONS	ACTUALS 1	ACTUALS 0
1		
0		

PREDICTIONS	ACTUALS 1	ACTUALS 0
1		
0		

PREDICTIONS	ACTUALS 1	ACTUALS 0
1		
0		

ROC-AUC



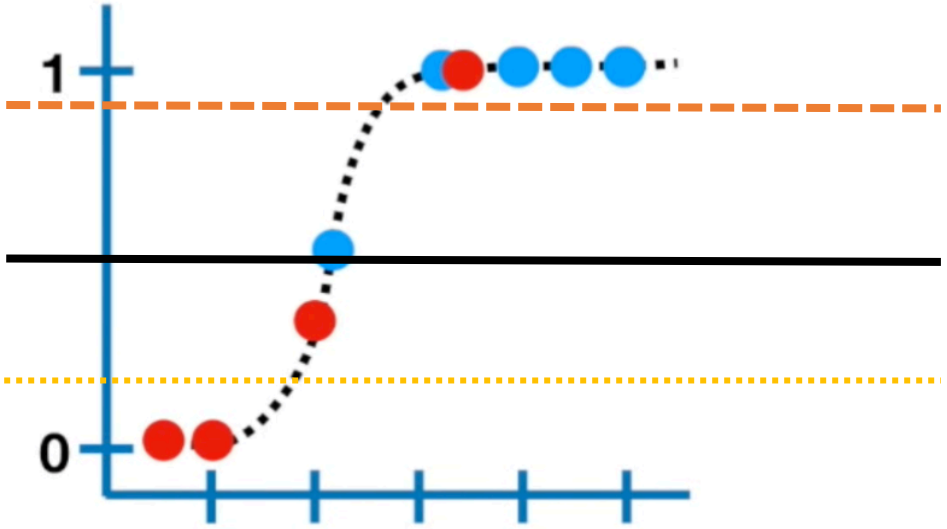


PREDICTIONS	ACTUALS 1	0
1	4	1
0	1	3

PREDICTIONS	ACTUALS 1	0
1	5	0
0	1	3

PREDICTIONS	ACTUALS 1	0
1	5	0
0	2	2

ROC-AUC



PREDICTIONS \ ACTUALS	1	0
1	4	1
0	1	3

$$TPR = \frac{TP}{TP + FN} = \frac{4}{4 + 1}$$

$$FPR = 1 - TPR = \frac{FP}{FP + TN} = \frac{1}{1 + 3}$$

PREDICTIONS \ ACTUALS	1	0
1	5	0
0	1	3

$$TPR = \frac{5}{5 + 1}$$

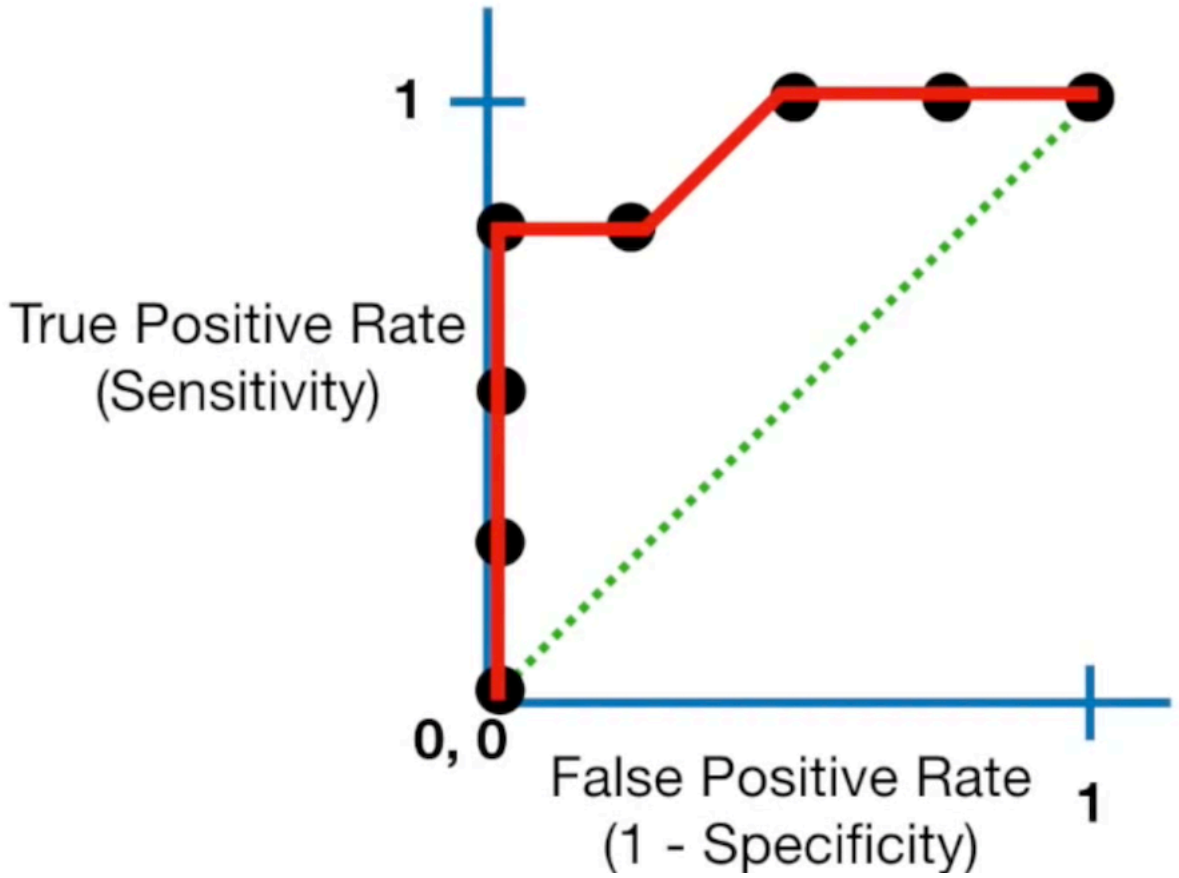
$$FPR = \frac{0}{0 + 3}$$

PREDICTIONS \ ACTUALS	1	0
1	5	0
0	2	2

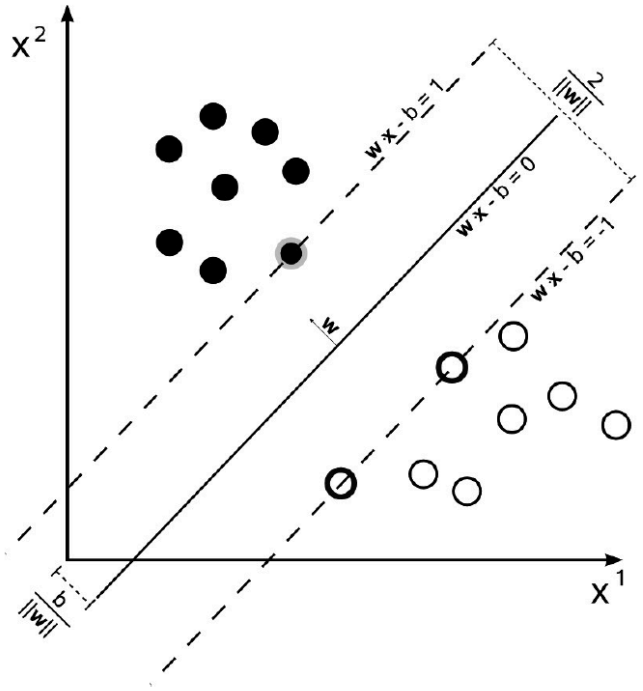
$$TPR = \frac{5}{5 + 2}$$

$$FPR = \frac{0}{0 + 2}$$

ROC-AUC


1
True Positive Rate
(Sensitivity)
0, 0
False Positive Rate
(1 - Specificity)
1

SVM – Hard Margin



$$f(x) = w^T x - b$$

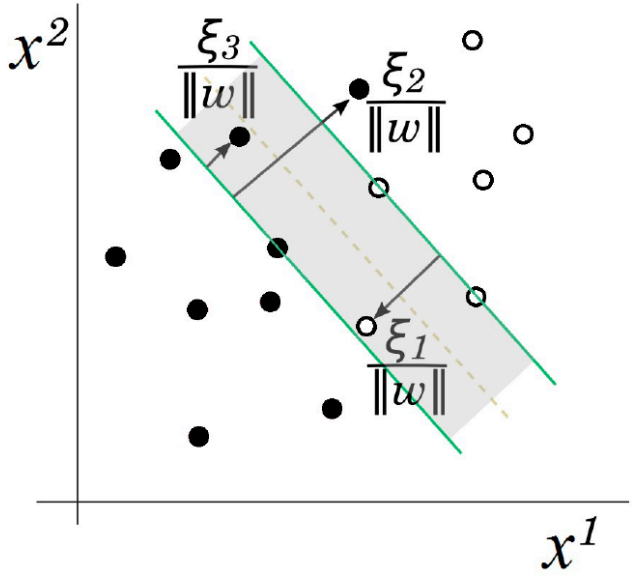
$$(w, b) = \arg\min_{w,b} \|w\|^2$$

Constraints:

$$w^T x_i - b \geq \quad 1, \; x_i \in Class$$
$$w^T x_i - b \leq -1, \; x_i \notin Class$$

$$\Rightarrow \quad (w^T x_i - b) y_i \geq \quad 1$$

SVM – Soft Margin



$$\xi_i = \max\{1 - f(x_i)\, y_i, 0\}$$

Hinge loss

$$f(x) = w^T x - b$$

$$(w, b) = \arg\min_{w,b} \sum_i \xi_i + \lambda \|w\|^2$$

$$\lambda > 0$$

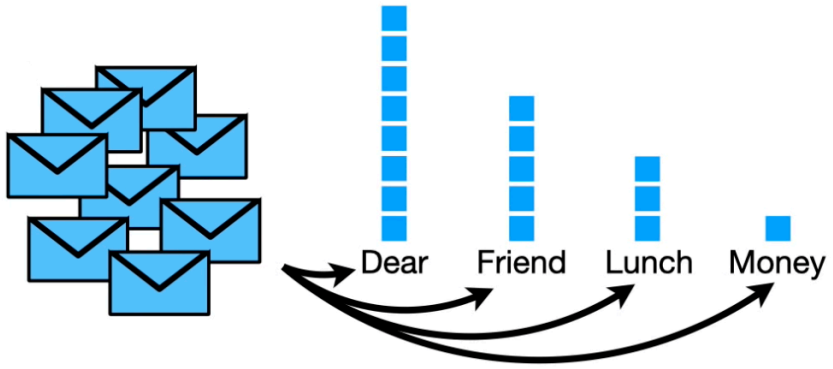
Constraints, for each i:

$$\xi_i \geq 0$$
$$(w^T x_i - b)\, y_i \geq 1 - \xi_i$$

Naïve Bayes

Likelihoods

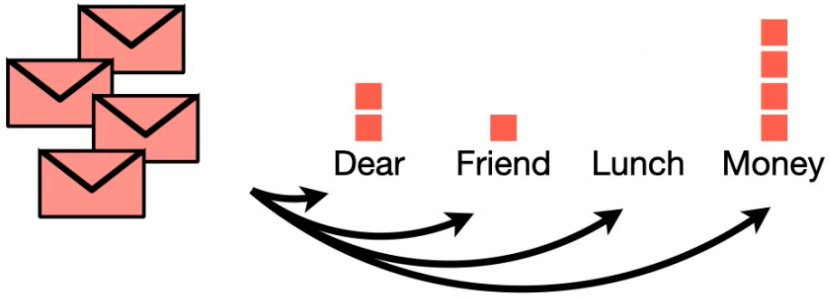
$$p(word \mid N) = \frac{\#word}{\#total\ words\ in\ N}$$



p(**Dear** | **N**) = 0.47
p(**Friend** | **N**) = 0.29
p(**Lunch** | **N**) = 0.18
p(**Money** | **N**) = 0.06

Prior probability $p(N) = \frac{\#N}{\#total\ emails}$

$$p(word \mid S) = \frac{\#word}{\#total\ words\ in\ S}$$



p(**Dear** | **S**) = 0.29
p(**Friend** | **S**) = 0.14
p(**Lunch** | **S**) = 0.00
p(**Money** | **S**) = 0.57

Prior probability $p(S) = \frac{\#S}{\#total\ emails}$

Dear Friend 

p(**N**) x p(**Dear** | **N**) x p(**Friend** | **N**)
= 0.09

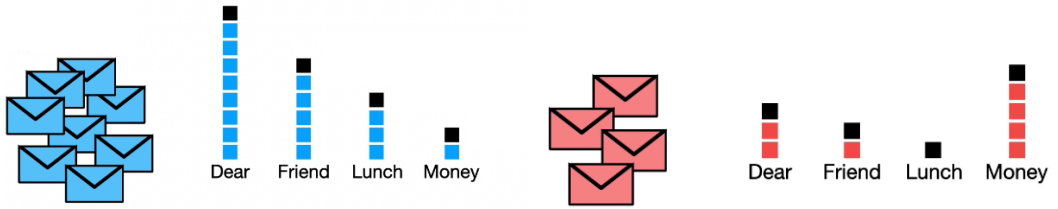
p(**S**) x p(**Dear** | **S**) x p(**Friend** | **S**)
= 0.01

Smoothing parametr α

Lunch Money Money Money Money 

p(**N**) x p(**Lunch** | **N**) x p(**Money** | **N**)^4
= 0.000002

p(**S**) x p(**Lunch** | **S**) x p(**Money** | **S**)^4
= 0



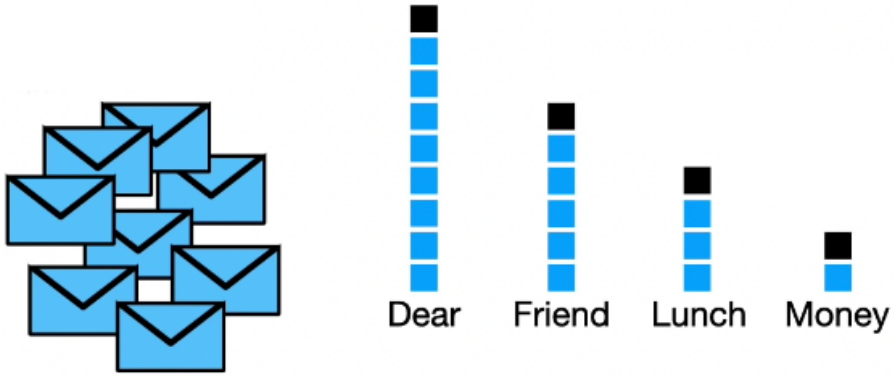
Naïve Bayes

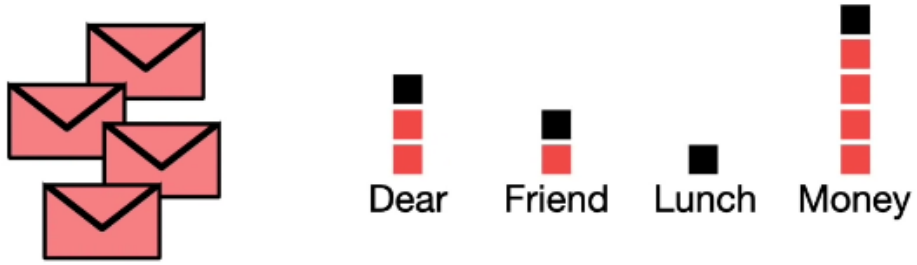
Smoothing parametr α

Lunch Money Money Money Money ✉

$p(\mathbf{N}) \times p(\mathbf{Lunch} \mid \mathbf{N}) \times p(\mathbf{Money} \mid \mathbf{N})^4 = 0.000002$

$p(\mathbf{S}) \times p(\mathbf{Lunch} \mid \mathbf{S}) \times p(\mathbf{Money} \mid \mathbf{S})^4 = 0$





$p(\mathbf{N}) \times p(\mathbf{Lunch} \mid \mathbf{N}) \times p(\mathbf{Money} \mid \mathbf{N})^4 = 0.00001$

$p(\mathbf{S}) \times p(\mathbf{Lunch} \mid \mathbf{S}) \times p(\mathbf{Money} \mid \mathbf{S})^4 = 0.00122$

Naïve Bayes

Why Bayes?

$$p(\mathbf{N}) \times p(\mathbf{Dear} \mid \mathbf{N}) \times p(\mathbf{Friend} \mid \mathbf{N})$$

$$p(\mathbf{S}) \times p(\mathbf{Dear} \mid \mathbf{S}) \times p(\mathbf{Friend} \mid \mathbf{S})$$

$$p(\mathbf{N} \mid \mathbf{Dear\ Friend}) = \frac{p(\mathbf{Dear\ Friend} \mid \mathbf{N})\ p(\mathbf{N})}{p(\mathbf{Dear\ Friend})}$$

$$p(\mathbf{S} \mid \mathbf{Dear\ Friend}) = \frac{p(\mathbf{Dear\ Friend} \mid \mathbf{S})\, p(\mathbf{S})}{p(\mathbf{Dear\ Friend})}$$

Why Naive?

Score for **Dear Friend** =

$$p(\mathbf{N}) \times p(\mathbf{Dear} \mid \mathbf{N}) \times p(\mathbf{Friend} \mid \mathbf{N}) = 0.08$$

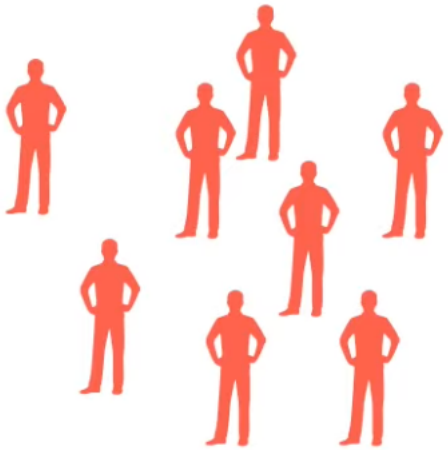
Score for **Friend Dear** =

$$p(\mathbf{N}) \times p(\mathbf{Friend} \mid \mathbf{N}) \times p(\mathbf{Dear} \mid \mathbf{N}) = 0.08$$

Naïve Bayes - Gaussian

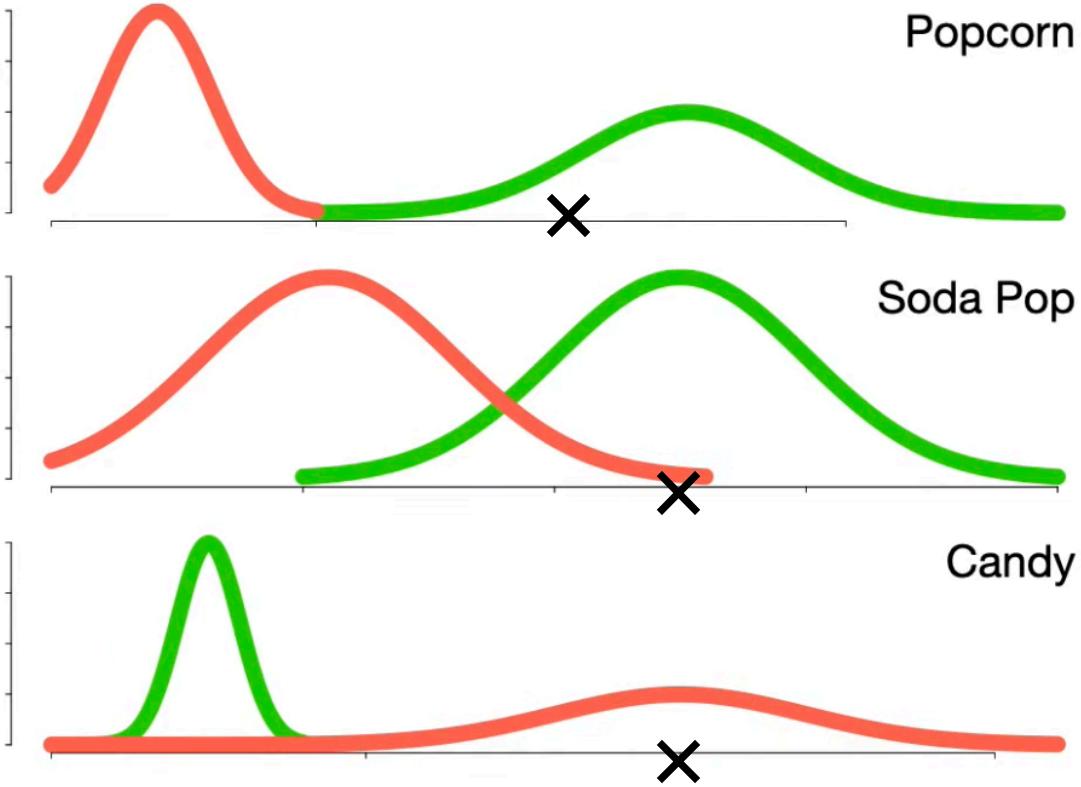
$$p(G) = \frac{\#G}{\#total}$$

Popcorn (grams)	Soda Pop (ml)	Candy (grams)
24.3	750.7	0.2
28.2	533.2	50.5
etc.	etc.	etc.



$$p(R) = \frac{\#R}{\#total}$$

Popcorn (grams)	Soda Pop (ml)	Candy (grams)
2.1	120.5	90.7
4.8	110.9	102.3
etc.	etc.	etc.



$$p(G) \times L(popcorn = 20 \mid G) \times L(soda\ pop \mid G) \times L(candy = 25 \mid G)$$

$$p(R) \times L(popcorn = 20 \mid R) \times L(soda\ pop \mid R) \times L(candy = 25 \mid R)$$

Entropy

Shannon's entropy

$$S = -\sum_{i=1}^{N} p_i \log_2 p_i$$

$$p_i = \frac{\#class_i}{\#total} = \frac{20}{20} = 1$$

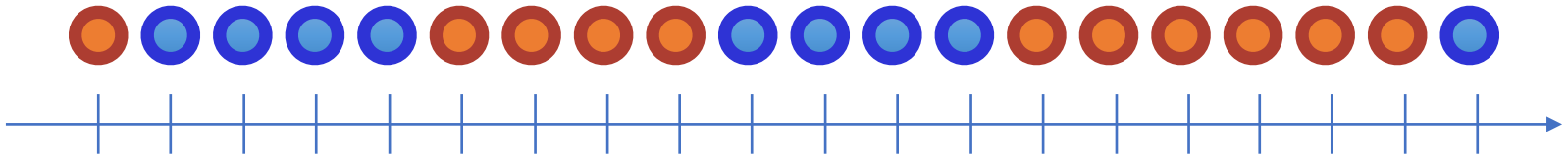
$$\log_2 1 = 0$$

$$S = -1 \times \log_2 0 = 0$$

Entropy

Shannon's entropy

$$S = -\sum_{i=1}^{N} p_i \log_2 p_i$$



$N =$ 

$p = \frac{9}{20}$

$p = \frac{11}{20}$

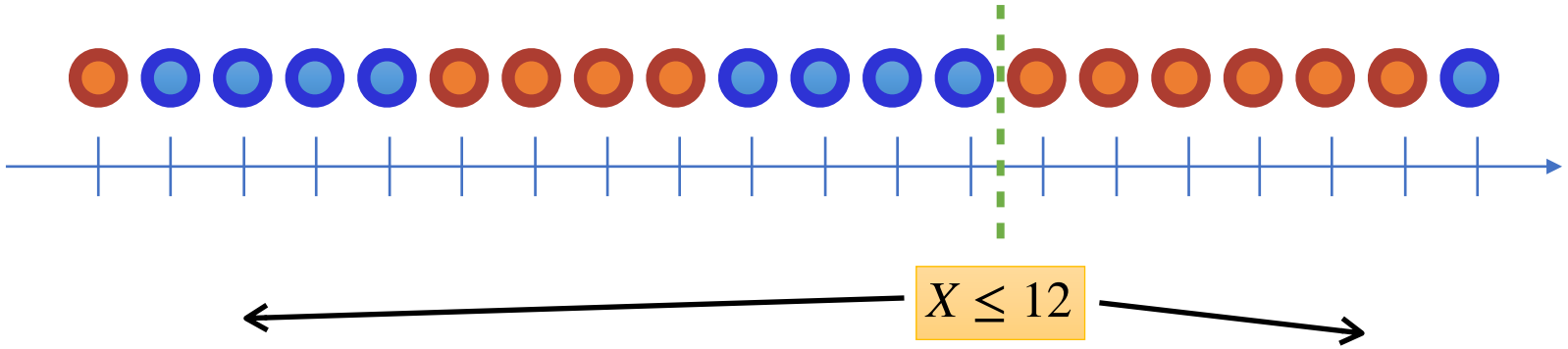
$$S = -\left(p_{orange} \log_2 p_{orange} + p_{blue} \log_2 p_{blue}\right)$$

$$S = -\left(\frac{9}{20} \log_2 \frac{9}{20} + \frac{11}{20} \log_2 \frac{11}{20}\right) \approx 1$$

Entropy

Shannon's entropy

$$S = -\sum_{i=1}^{N} p_i \log_2 p_i$$



$N =$ 

 $p = \frac{8}{13}$

 $p = \frac{5}{13}$

$$S_1 = -\left(\frac{8}{13}\log_2\frac{8}{13} + \frac{5}{13}\log_2\frac{5}{13}\right) \approx 0.96$$

$N =$ 

 $p = \frac{1}{7}$

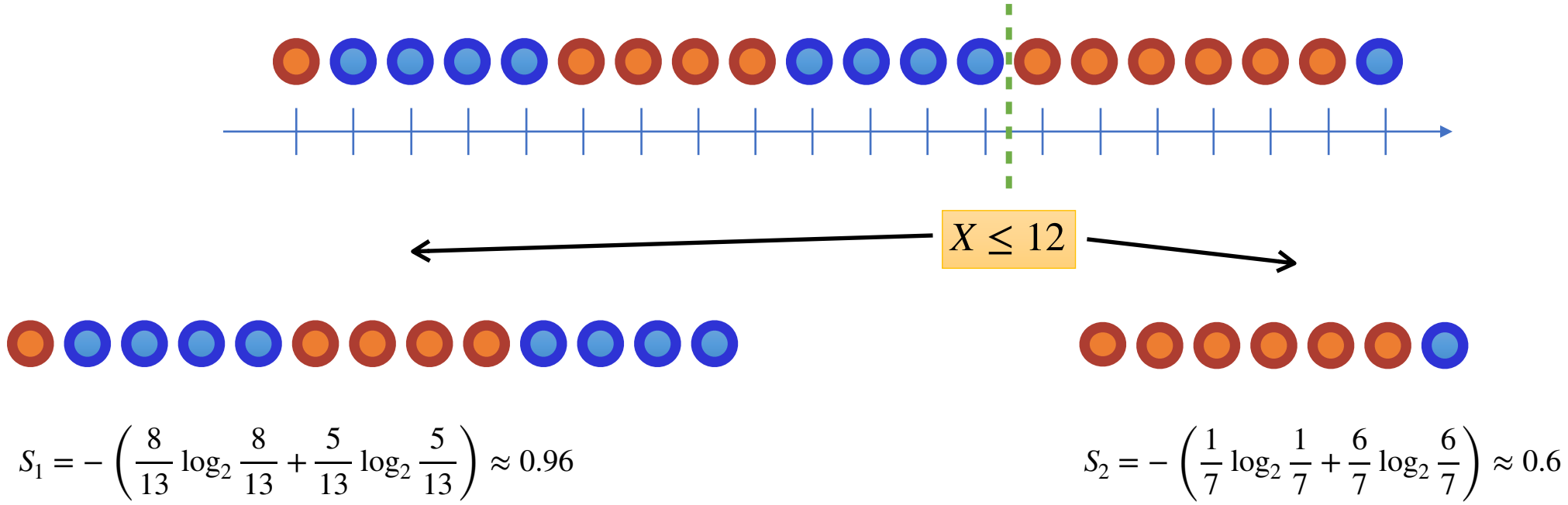
 $p = \frac{6}{7}$

$$S_2 = -\left(\frac{1}{7}\log_2\frac{1}{7} + \frac{6}{7}\log_2\frac{6}{7}\right) \approx 0.6$$

Entropy

$$S = -\sum_{i=1}^{N} p_i \log_2 p_i$$

Shannon's entropy



$$S_1 = -\left(\frac{8}{13}\log_2\frac{8}{13} + \frac{5}{13}\log_2\frac{5}{13}\right) \approx 0.96$$

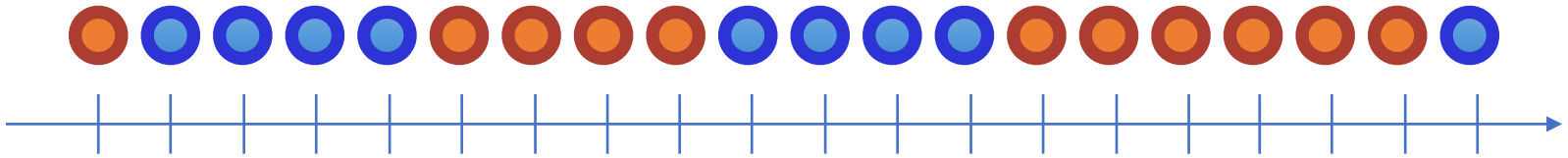
$$S_2 = -\left(\frac{1}{7}\log_2\frac{1}{7} + \frac{6}{7}\log_2\frac{6}{7}\right) \approx 0.6$$

As you can see, entropy has decreased in both groups, more so in the right group. Since entropy is, in fact, the degree of chaos (or uncertainty) in the system, the reduction in entropy is called information gain.

Gini

Gini Index, Gini Coeficient

$$G = \sum_k p_i(1 - p_i) = 1 - \sum_k p_i^2$$



$N =$ 

$p = \frac{9}{20}$

$p = \frac{11}{20}$

$$G = 1 - \left(p_{orange}^2 + p_{blue}^2\right)$$

$$G = 1 - \left(\left(\frac{9}{20}\right)^2 + \left(\frac{11}{20}\right)^2\right) \approx 0.5$$

Information Gain

Information Gain/Gini impurity

$$IG\left(Q\right) = \sum_{i} \frac{N_i}{N} H_i(Q)$$

$N =$ ilość elementów przed podziałem

$N_i =$ ilość elementów w grupie i

$Q =$ podział

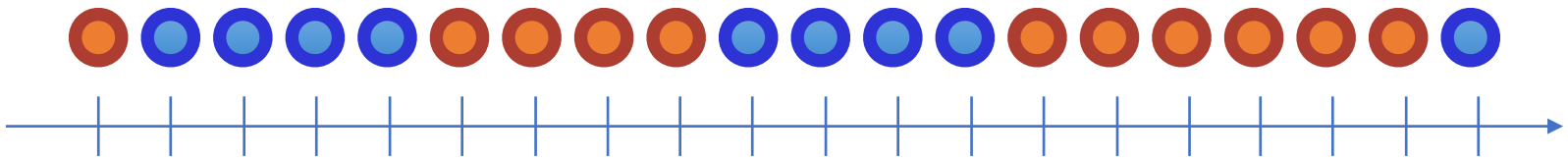
$H_i =$ jakość podziału, np. entropia, gini

Information Gain

Entropy

$$IG\left(Q\right) = \sum_{i} \frac{N_i}{N} H_i(Q)$$

H_i = entropia



$\mathbb{P} - perfect\ split$

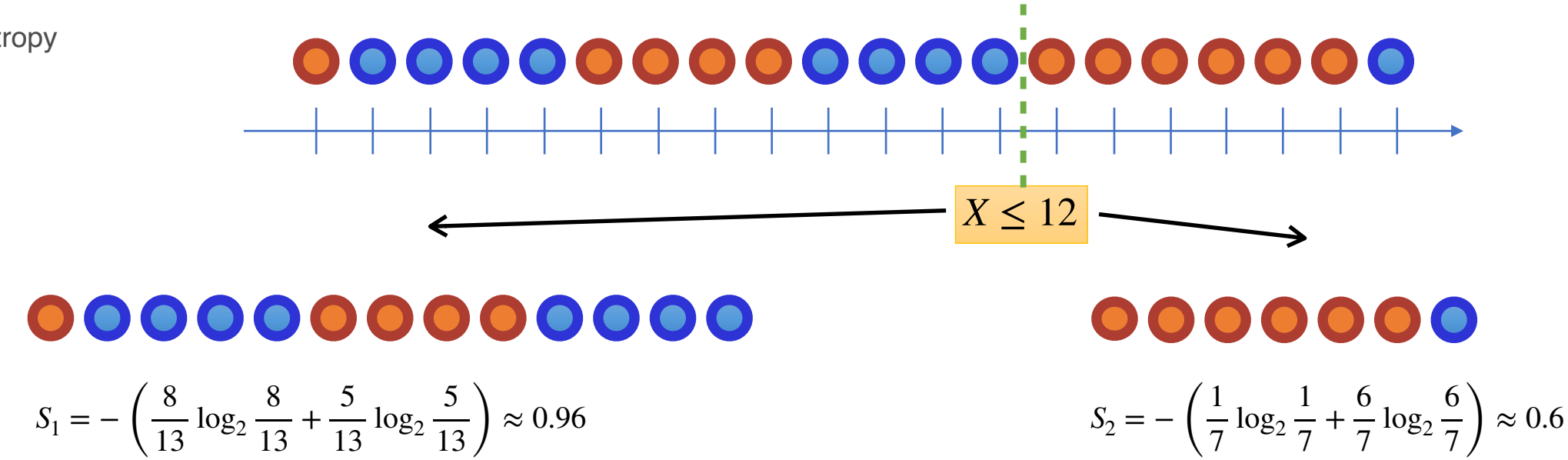
$S_1 = 0$

$S_2 = 0$

$$IG\left(\mathbb{P}\right) = \frac{N_1}{N} S_1 + \frac{N_2}{N} S_2 = \frac{N_1}{N} 0 + \frac{N_2}{N} 0 = 0$$

Information Gain

Entropy



$$S_1 = -\left(\frac{8}{13}\log_2\frac{8}{13} + \frac{5}{13}\log_2\frac{5}{13}\right) \approx 0.96$$

$$S_2 = -\left(\frac{1}{7}\log_2\frac{1}{7} + \frac{6}{7}\log_2\frac{6}{7}\right) \approx 0.6$$

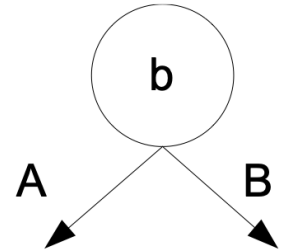
$$IG\,(x \leq 12) = entropy_{start} - \left(\frac{\#class_1}{\#total} entropy_1 + \frac{\#class_2}{\#total} entropy_2\right)$$

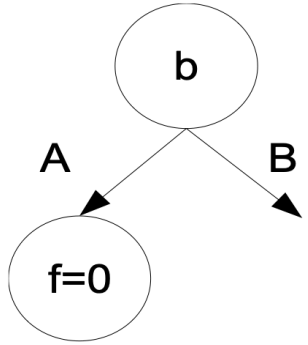
$$IG\,(x \leq 12) = S_0 - \left(\frac{13}{20}S_1 + \frac{7}{20}S_2\right) \approx 0.16$$

Decision Tree - ID3

a	b	c	d	e	f
A	A	C	A	A	0
A	B	B	B	B	0
A	A	A	A	C	0
B	B	A	C	D	0
B	B	B	C	B	1
B	B	C	C	D	1
B	B	B	B	B	1
A	B	A	B	C	1

		f=0	f=1	częstość wystąpień	entropia wartości atrybutu	entropia atrybutu
a	A	3	1	4	0.811278	0.811278
	B	1	3	4	0.811278	
b	A	2	0	2	0	0.688722
	B	2	4	6	0.918296	
c	A	2	1	3	0.918296	0.938722
	B	1	2	3	0.918296	
	C	1	1	2	1	
d	A	2	0	2	0	0.688722
	B	1	2	3	0.918296	
	C	1	2	3	0.918296	
e	A	1	0	1	0	0.844361
	B	1	2	3	0.918296	
	C	1	1	2	1	
	D	1	1	2	1	





$$\#A_a = 4$$
$$\#B_a = 4$$

$$p_A^{a,f=0} = \frac{3}{4}$$

$$p_A^{a,f=1} = \frac{1}{4}$$

$$p_B^{a,f=0} = \frac{1}{4}$$

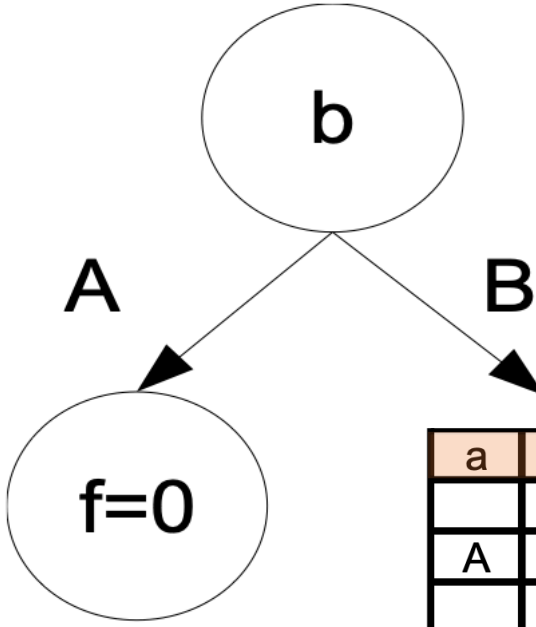
$$p_B^{a,f=1} = \frac{3}{4}$$

$$S_A^a = -\left(p_A^{a,f=0}\log_2 p_A^{a,f=0} + p_A^{a,f=1}\log_2 p_A^{a,f=1}\right)$$

$$= -\frac{3}{4}\log_2\frac{3}{4} - \frac{1}{4}\log_2\frac{1}{4} = 0.31127812 + 0.5 = 0.81127812$$

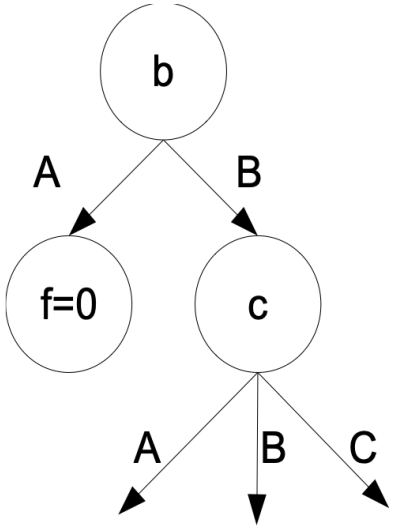
Decision Tree - ID3

a	b	c	d	e	f
A	A	C	A	A	0
A	B	B	B	B	0
A	A	A	A	C	0
B	B	A	C	D	0
B	B	B	C	B	1
B	B	C	C	D	1
B	B	B	B	B	1
A	B	A	B	C	1



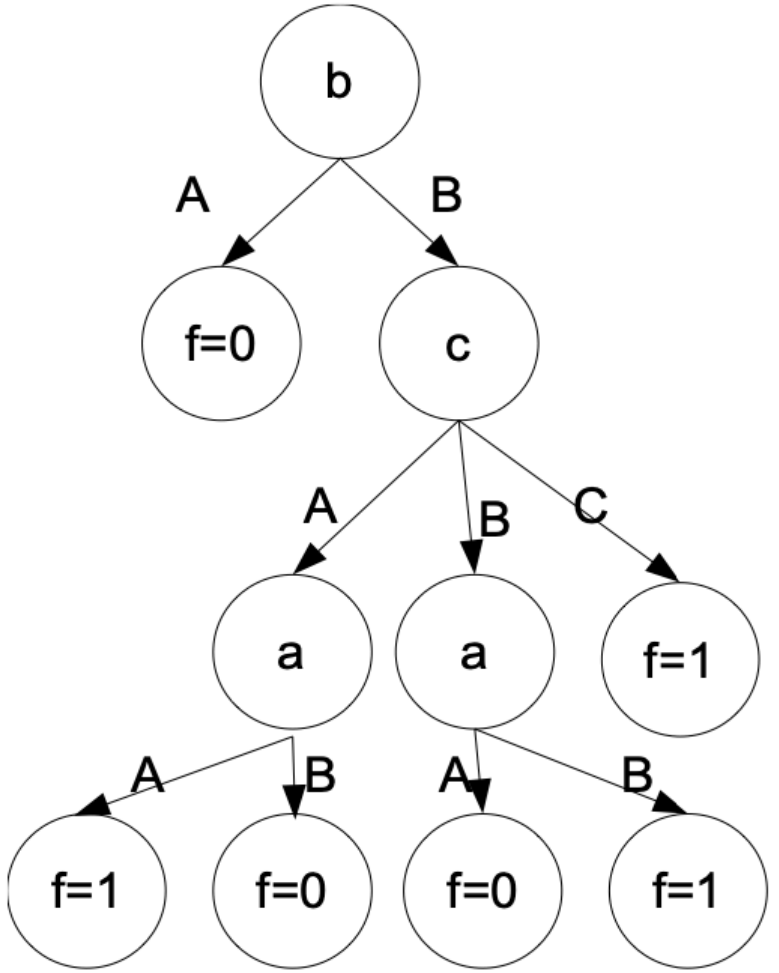
a	b	c	d	e	f
A	B	B	B	B	0
B	B	A	C	D	0
B	B	B	C	B	1
B	B	C	C	D	1
B	B	B	B	B	1
A	B	A	B	C	1

		f=0	f=1	częstość wystąpień	entropia wartości atrybutu	entropia atrybutu
a	A	1	1	2	1	0.874185
	B	1	3	4	0.811278	
c	A	1	1	2	1	0.792481
	B	1	2	3	0.918296	
	C	0	1	1	0	
d	A	0	0	0	0	0.918296
	B	1	2	3	0.918296	
	C	1	2	3	0.918296	
e	A	0	0	0	0	0.792481
	B	1	2	3	0.918296	
	C	0	1	1	0	
	D	1	1	2	1	



Decision Tree - ID3

a	b	c	d	e	f
A	A	C	A	A	0
A	B	B	B	B	0
A	A	A	A	C	0
B	B	A	C	D	0
B	B	B	C	B	1
B	B	C	C	D	1
B	B	B	B	B	1
A	B	A	B	C	1



Decision Tree - C4.5

https://www.mimuw.edu.pl/~awojna/SID/referaty/strzelczak/c4_5Main.html

Decision Tree - CART

Gini:

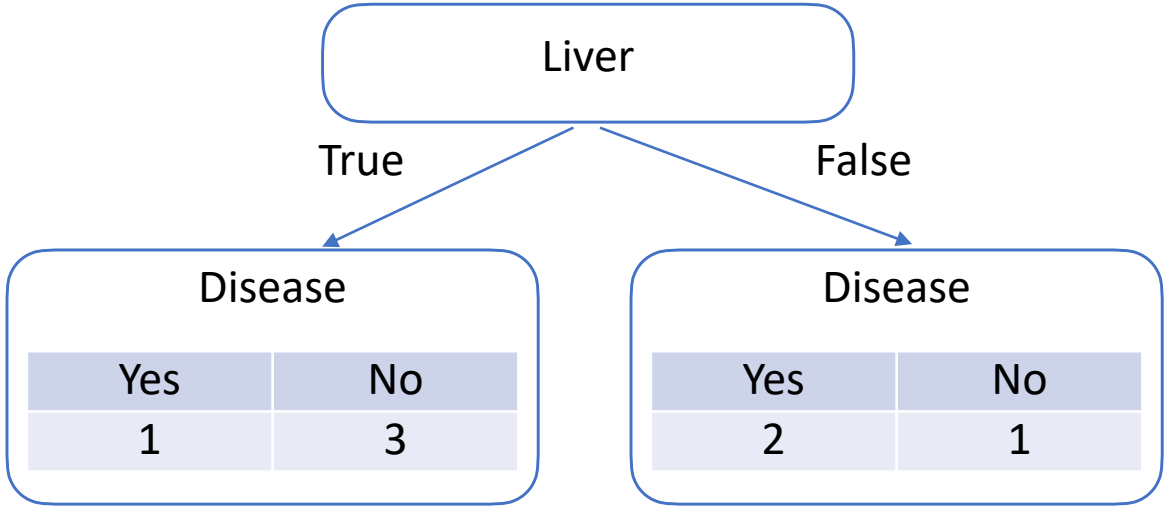
$$H(Q_m) = \sum_k p_{mk}(1 - p_{mk})$$

Log Loss or Entropy:

$$H(Q_m) = -\sum_k p_{mk} \log(p_{mk})$$

Decision Tree - CART

Liver	Lung	Age	Disease
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No



$$G = \sum_i p_i(1 - p_i) = 1 - \sum_i p_i^2$$

$$G_{True} = 1 - \left(\frac{1}{1+3}\right)^2 - \left(\frac{3}{1+3}\right)^2 = 0.375$$

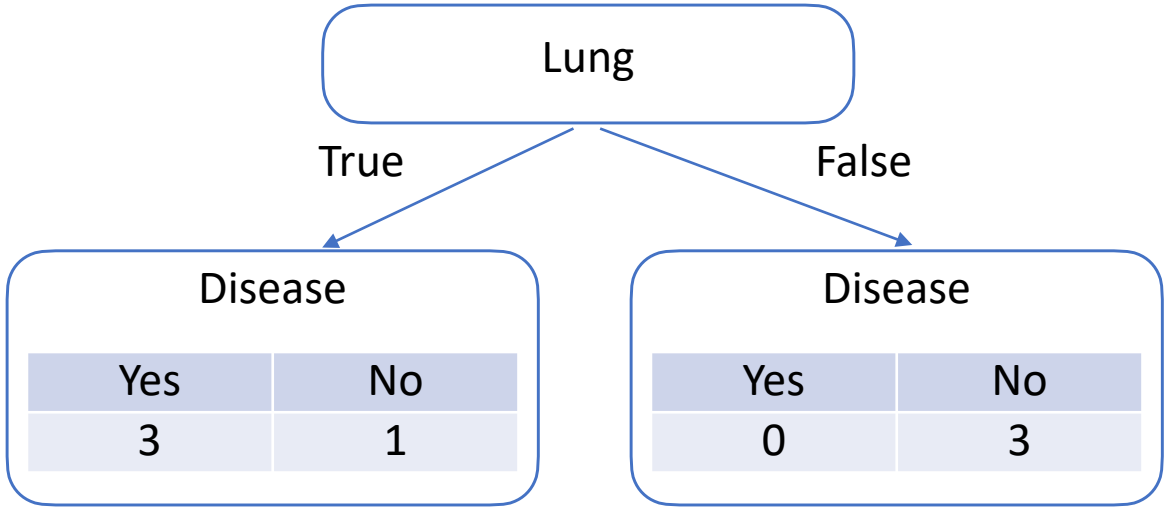
$$G_{False} = 1 - \left(\frac{2}{2+1}\right)^2 - \left(\frac{1}{2+1}\right)^2 = 0.444$$

$$IG\left(Q\right) = \sum_i \frac{N_i}{N} H_i(Q)$$

$$IG(Liver) = \frac{4}{7}0.375 + \frac{3}{7}0.444 = 0.405$$

Decision Tree - CART

Liver	Lung	Age	Disease
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No



$$G_{True} = 1 - \left(\frac{1}{3+1}\right)^2 - \left(\frac{3}{3+1}\right)^2 = 0.375$$

$$G_{False} = 1 - \left(\frac{0}{3+0}\right)^2 - \left(\frac{3}{3+0}\right)^2 = 0$$

$$IG(Lung) = 0.214$$

Decision Tree - CART

Age	Disease
7	No
12	No
18	Yes
35	Yes
38	Yes
50	No
83	No

9.5 | 15 | 26.5 | 36.5 | 44 | 66.5

$$IG(Age < 9.5) = 0.429$$

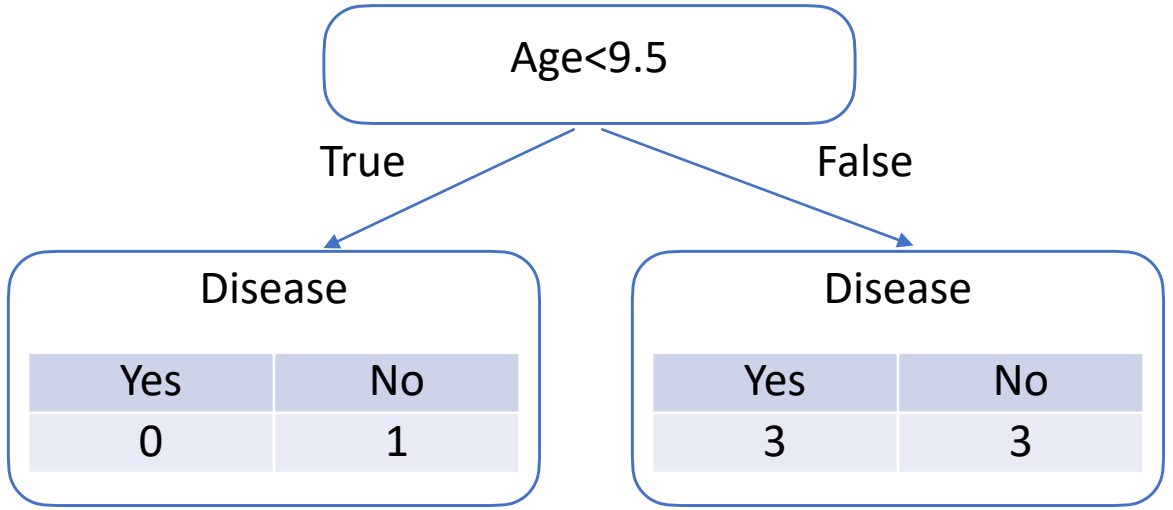
$$IG(Age < 15) = 0.343$$

$$IG(Age < 26.5) = 0.476$$

$$IG(Age < 36.5) = 0.476$$

$$IG(Age < 44) = 0.343$$

$$IG(Age < 44) = 0.429$$



$$G_{True} = 1 - \left(\frac{0}{0+1}\right)^2 - \left(\frac{1}{0+1}\right)^2 = 0$$

$$G_{False} = 1 - \left(\frac{3}{3+3}\right)^2 - \left(\frac{3}{3+3}\right)^2 = 0.5$$

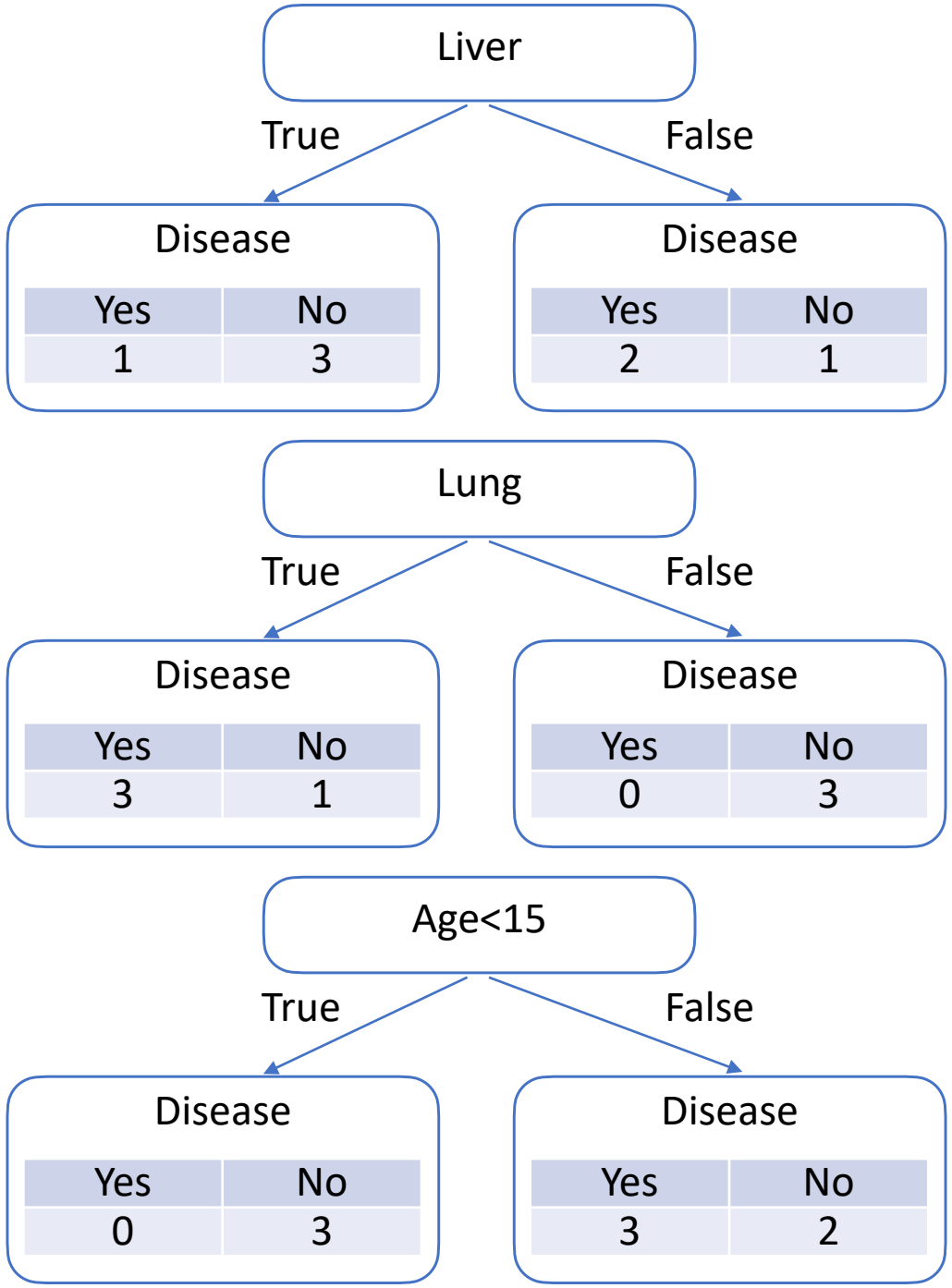
$$IG(Age = 9.5) = 0.429$$

Decision Tree - CART

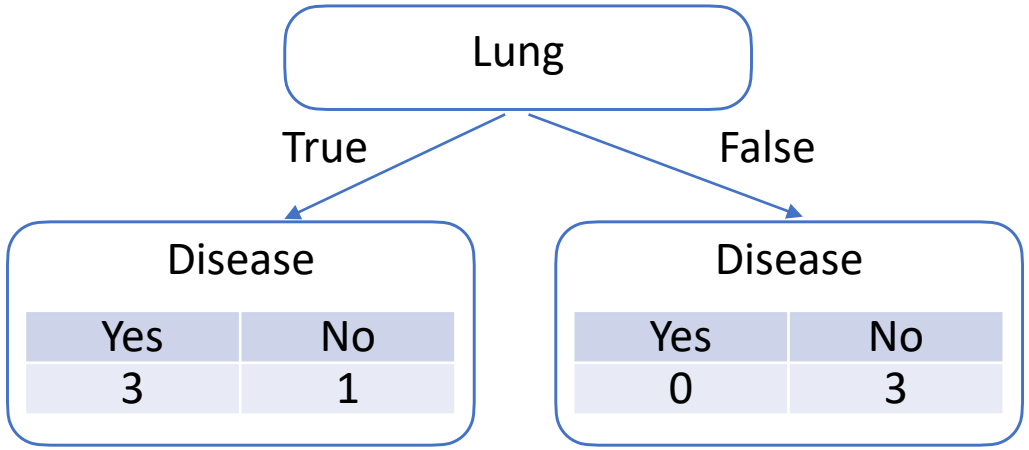
$IG(Liver) = 0.405$

$IG(Lung) = 0.214$

$IG(Age < 15) = 0.343$



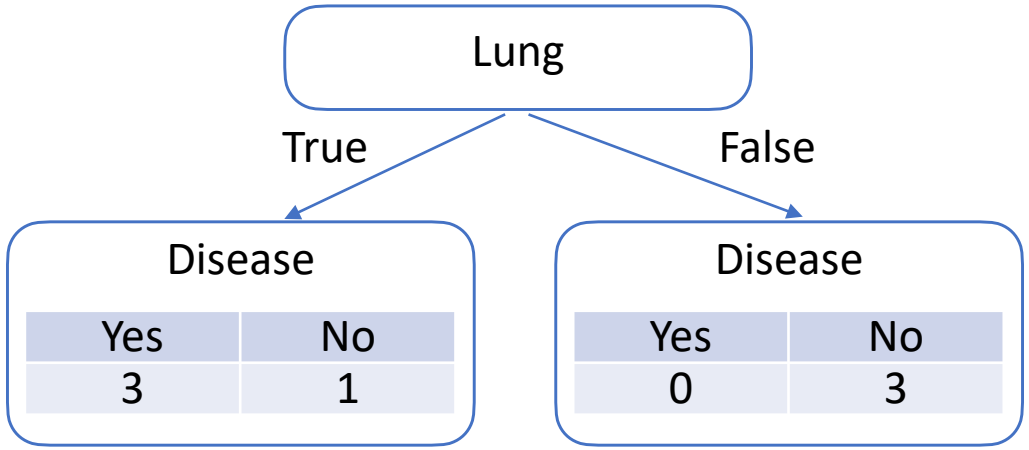
Decision Tree - CART



Liver	Lung	Age	Disease
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No

Liver	Lung	Age	Disease
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No

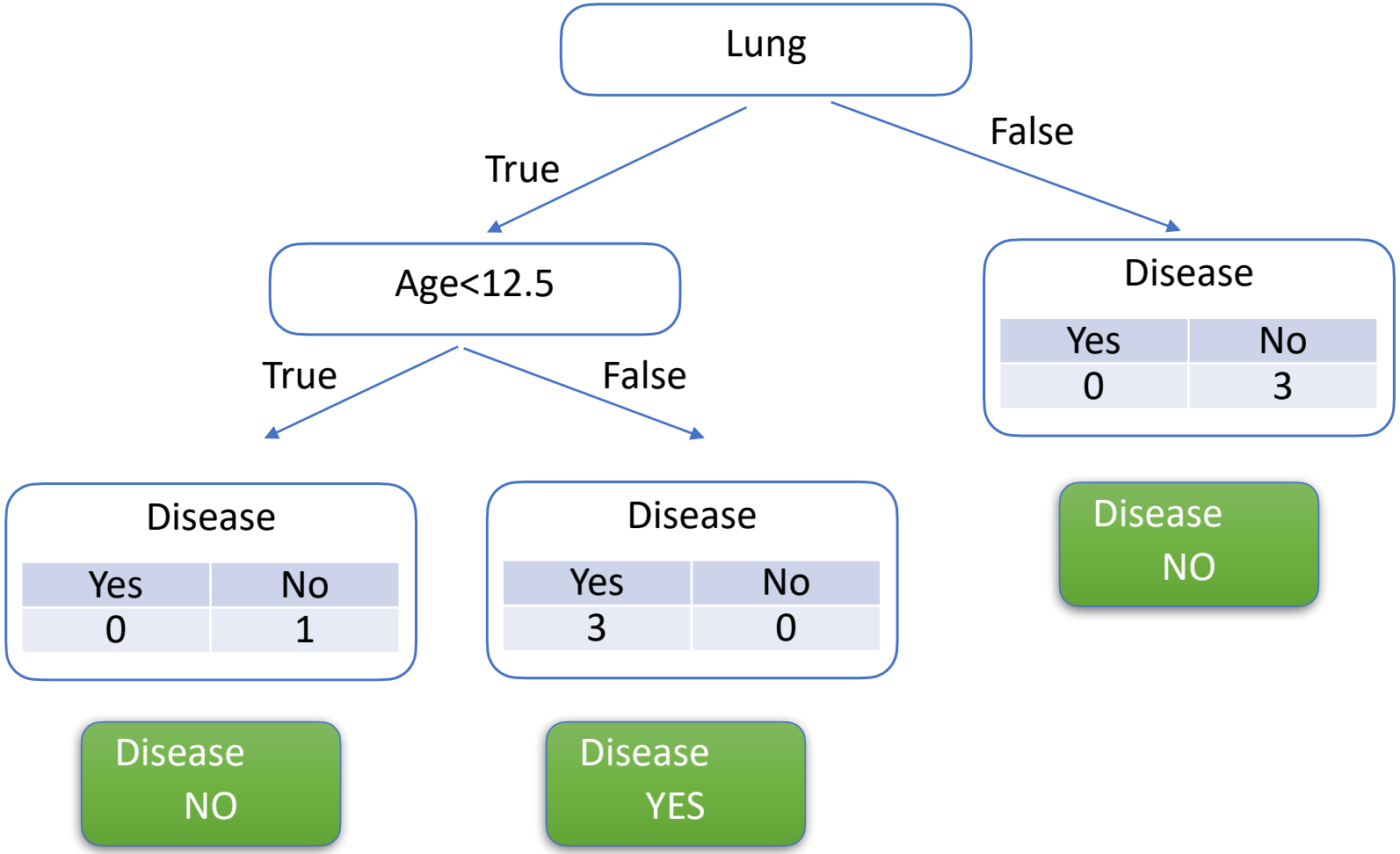
Decision Tree - CART



Liver	Lung	Age	Disease
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No

Liver	Lung	Age	Disease
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No

Decision Tree - CART



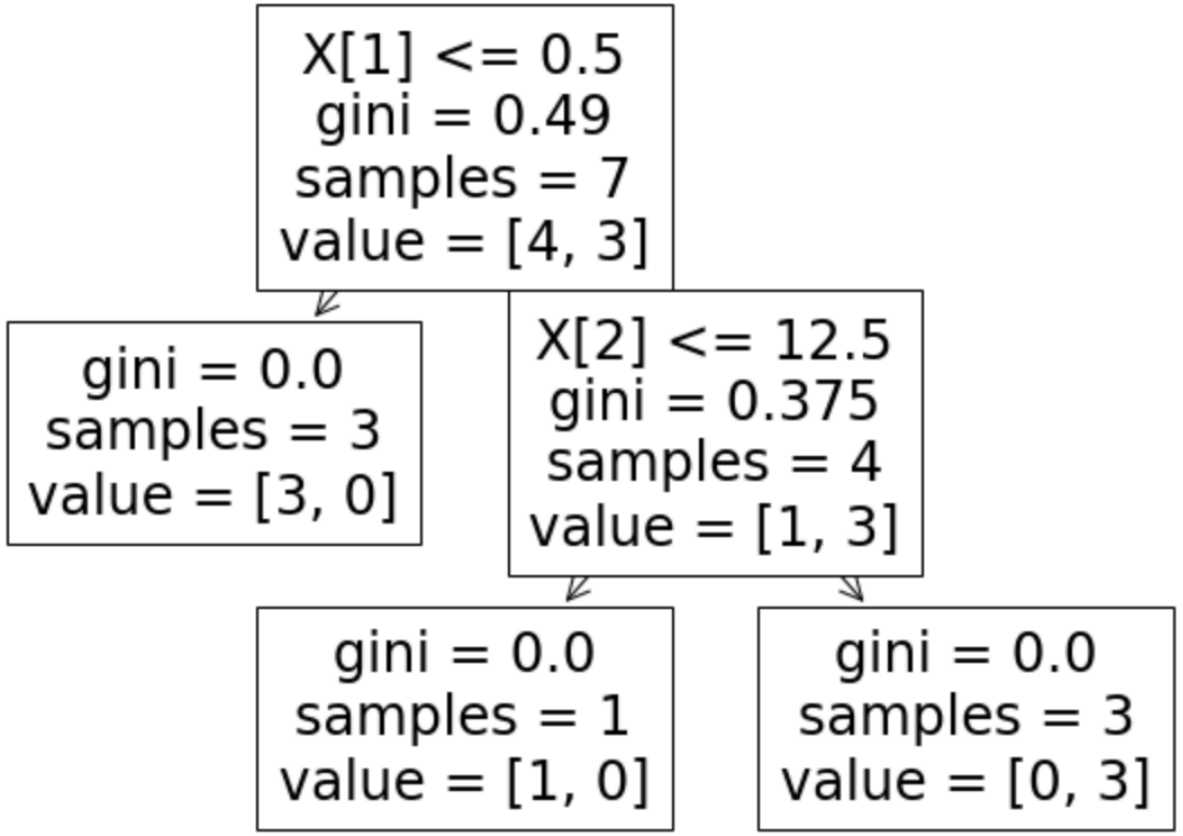
Decision Tree

Scikit-learn

	Liver	Lung	Age	Disease
0	1	1	7	0
1	1	0	12	0
2	0	1	18	1
3	0	1	35	1
4	1	1	38	1
5	1	0	50	0
6	0	0	83	0

```
from sklearn.tree import plot_tree

fig, ax = plt.subplots(figsize=(10, 7))
plot_tree(tree, ax=ax)
plt.show()
```

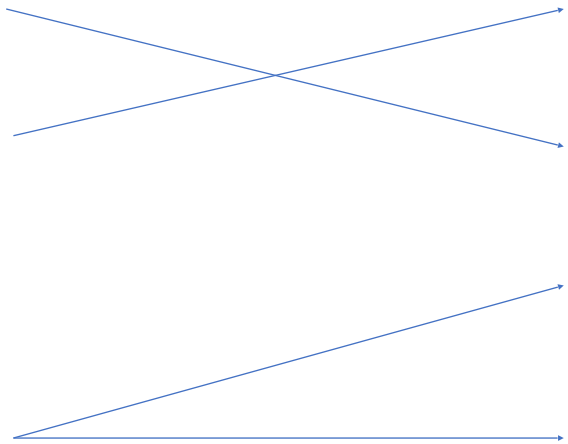


Random Forest

Step 1: Bootstrapping

Original Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	Yes	167	Yes



Bootstrapped Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	Yes	167	Yes
Yes	No	Yes	167	Yes

Random Forest

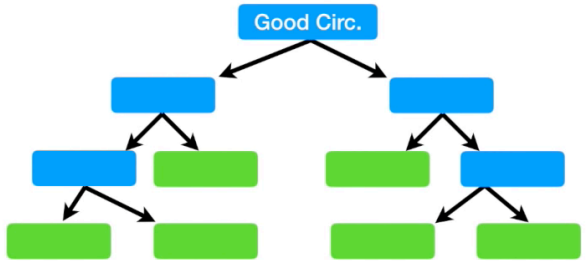
Step 2: Create Decision Tree .

Randomly select subset of features.

Find best split.

Randomly select subset of features to node split.

Create Tree considering subset of features.

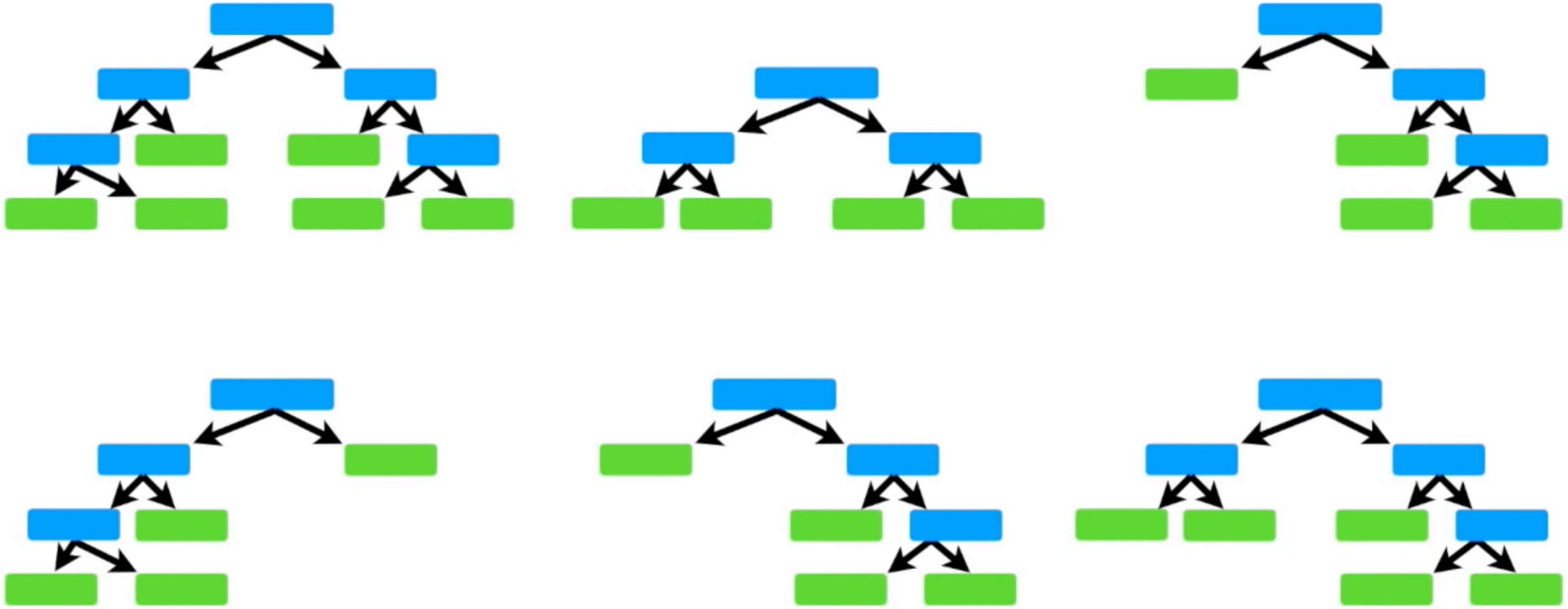


Bootstrapped Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	Yes	167	Yes
Yes	No	Yes	167	Yes

Random Forest

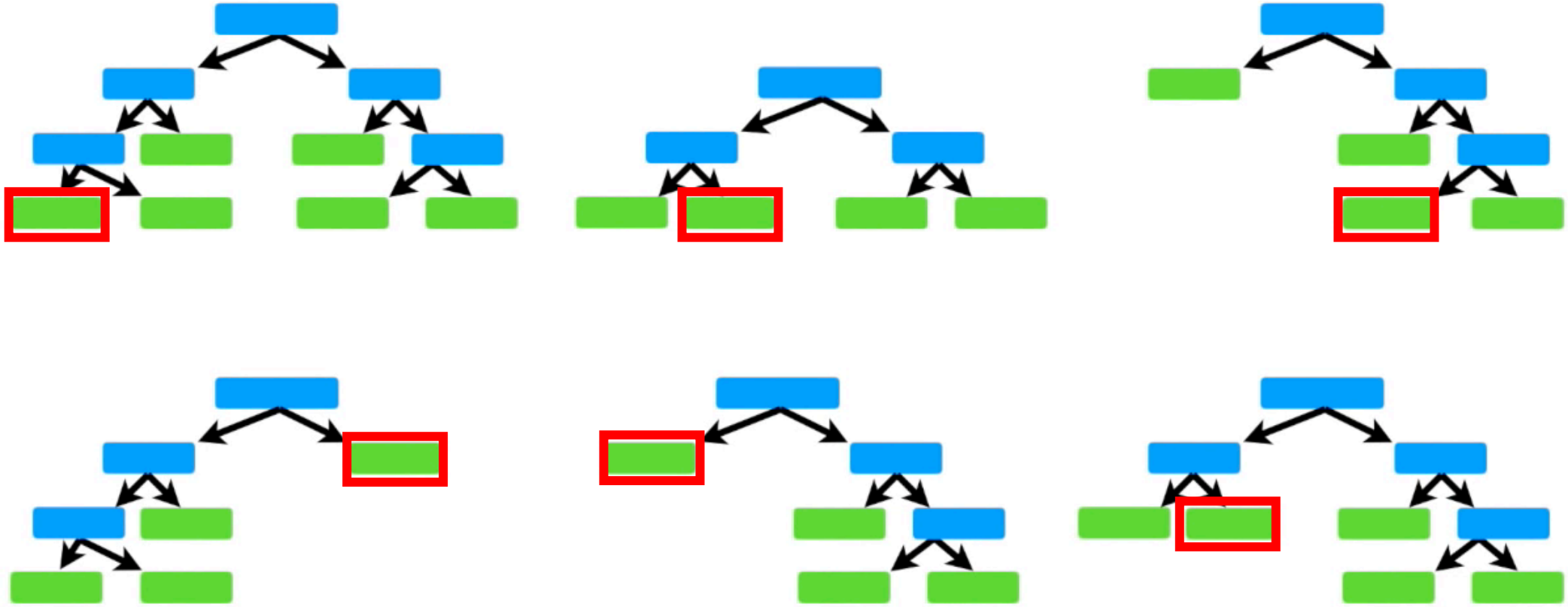
Repeat Step 1 & Step 2 creating next trees.



Random Forest

Predictions

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	No	No	168	



Random Forest

Out-Of-Bag Dataset

Original Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	Yes	167	Yes

Bootstrapped Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	Yes	167	Yes
Yes	No	Yes	167	Yes

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Sugar
Yes	Yes	No	210	No

We can make predictions for *oob* subset and calculate metrics.

In sklearn module sklearn.ensemble. RandomForestClassifier has ***oob_score_***
attrbiiute returning accuracy.

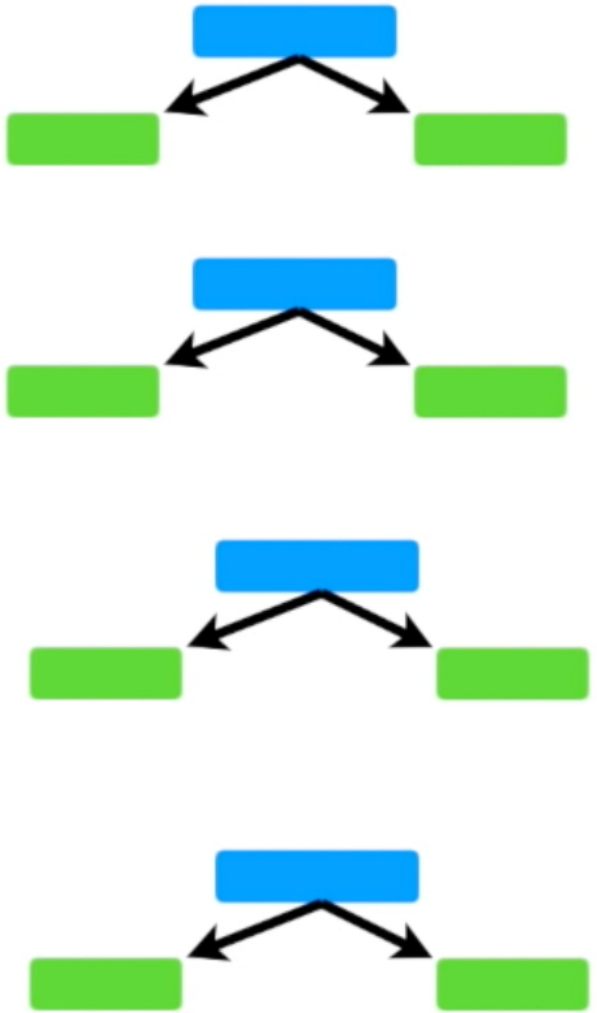
oob_score_ : *float*

Score of the training dataset obtained using an out-of-bag estimate. This attribute exists only when `oob_score` is True.

AdaBoost

Original Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	Yes	167	Yes



stump

AdaBoost

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	1/8
No	Yes	180	Yes	1/8
Yes	No	210	Yes	1/8
Yes	Yes	167	Yes	1/8
No	Yes	156	No	1/8
No	Yes	125	No	1/8
Yes	No	168	No	1/8
Yes	Yes	172	No	1/8

Step 1. Find best split minimising Gini Index

AdaBoost

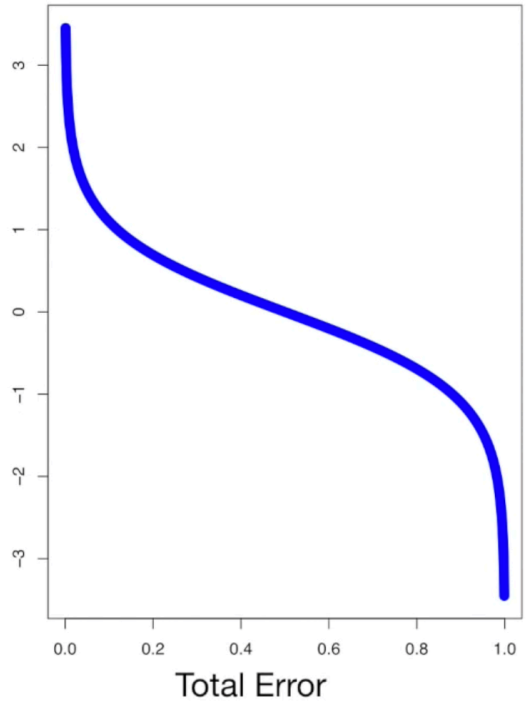
Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	New Weight
Yes	Yes	205	Yes	0.05
No	Yes	180	Yes	0.05
Yes	No	210	Yes	0.05
Yes	Yes	167	Yes	0.33
No	Yes	156	No	0.05
No	Yes	125	No	0.05
Yes	No	168	No	0.05
Yes	Yes	172	No	0.05

Step 1. Find best split minimizing Gini

Step 2. Update sample weights.

$$I = \frac{1}{2}\log(\frac{1 - total\_error}{total\_error})$$

$total\_error$ - sum of incorrect classified samples weights



$New\ weigth = weight \times e^{I}$ - True

$New\ weigth = weight \times e^{-I}$ - False

AdaBoost

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	0.07
No	Yes	180	Yes	0.07
Yes	No	210	Yes	0.07
Yes	Yes	167	Yes	0.49
No	Yes	156	No	0.07
No	Yes	125	No	0.07
Yes	No	168	No	0.07
Yes	Yes	172	No	0.07

Step 1. Find best split minimizing Gini

Step 2. Update sample weights.

Step 3. Normalize sample weights.

AdaBoost

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	0.07
No	Yes	180	Yes	0.07
Yes	No	210	Yes	0.07
Yes	Yes	167	Yes	0.49
No	Yes	156	No	0.07
No	Yes	125	No	0.07
Yes	No	168	No	0.07
Yes	Yes	172	No	0.07

Step 1. Find best split minimizing Gini

Step 2. Update sample weights.

Step 3. Normalize sample weights.

Step 3. Bootstrap dataset using new sample weights.

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
No	Yes	156	No	1/8
Yes	Yes	167	Yes	1/8
No	Yes	125	No	1/8
Yes	Yes	167	Yes	1/8
Yes	Yes	167	Yes	1/8
Yes	Yes	172	No	1/8
Yes	Yes	205	Yes	1/8
Yes	Yes	167	Yes	1/8

AdaBoost

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
No	Yes	156	No	1/8
Yes	Yes	167	Yes	1/8
No	Yes	125	No	1/8
Yes	Yes	167	Yes	1/8
Yes	Yes	167	Yes	1/8
Yes	Yes	172	No	1/8
Yes	Yes	205	Yes	1/8
Yes	Yes	167	Yes	1/8

Step 1. Find best split minimizing Gini

Step 2. Update sample weights.

Step 3. Normalize sample weights.

Step 4. Bootstrap dataset using new sample weights.

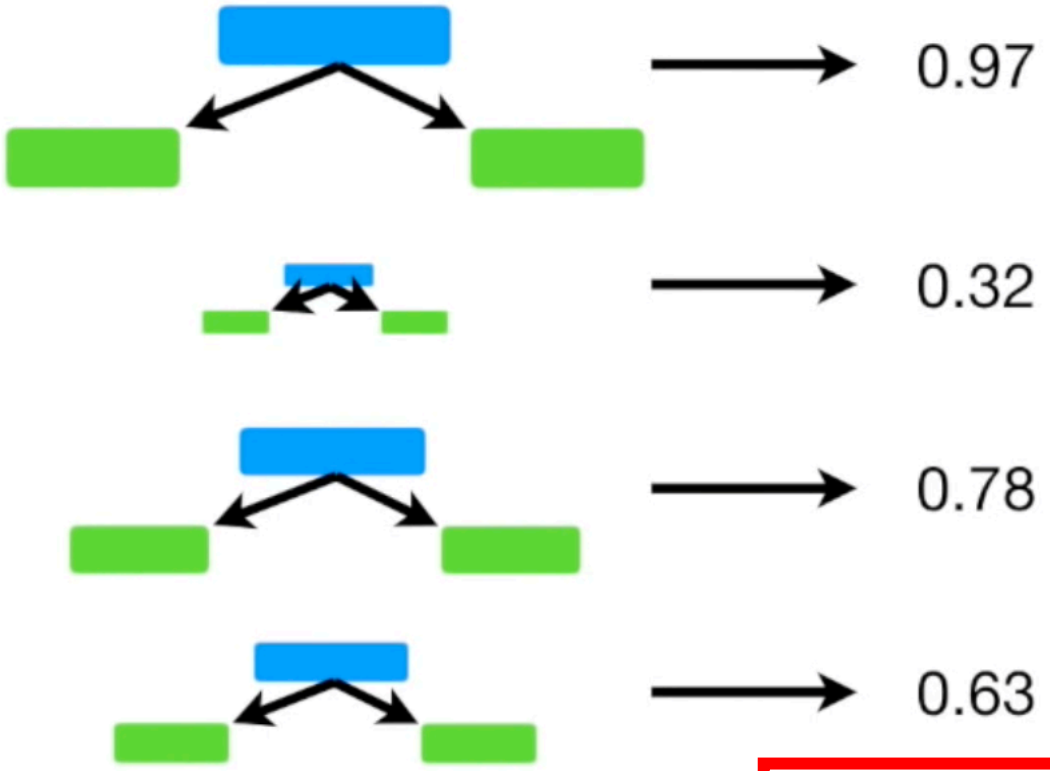
Step 5. Repeat using bootstrap dataset.

AdaBoost

$$I = \frac{1}{2}\log\left(\frac{1 - total\_error}{total\_error}\right)$$

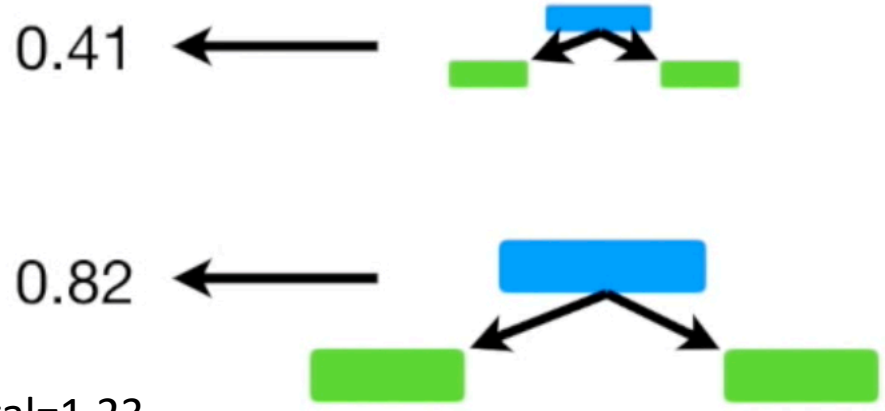
Chest Pain	Blocked Arteries	Patient Weight	Heart Disease
No	Yes	156	No

Calculate I



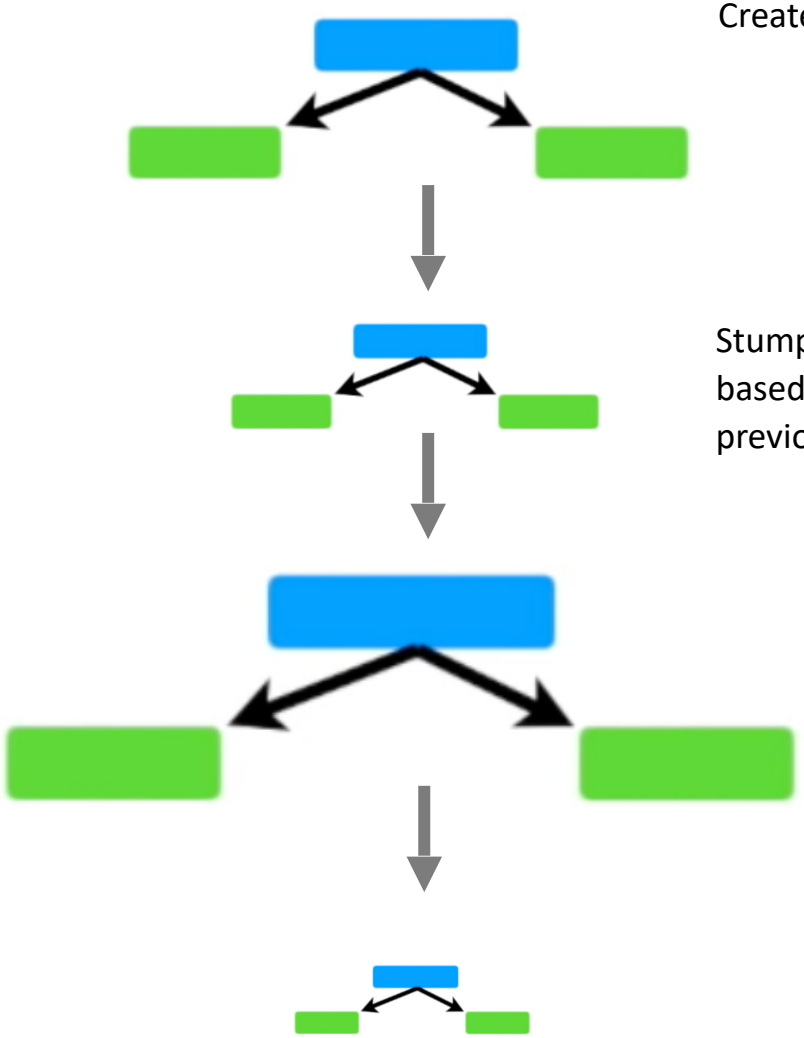
Total=2.7

Calculate I



Total=1.23

AdaBoost



Create stumps

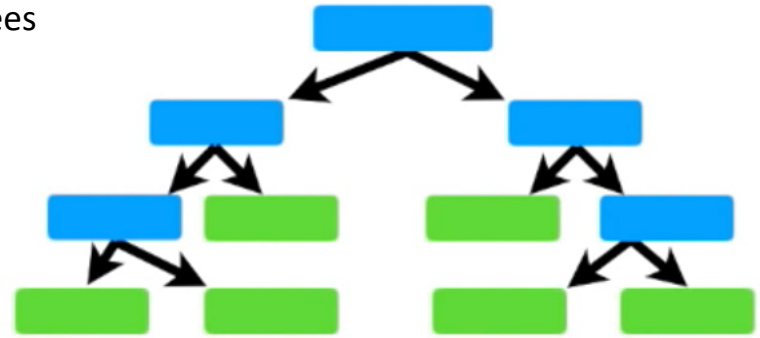
Stumps create based on previous stump.

Stumps have no equal weights.

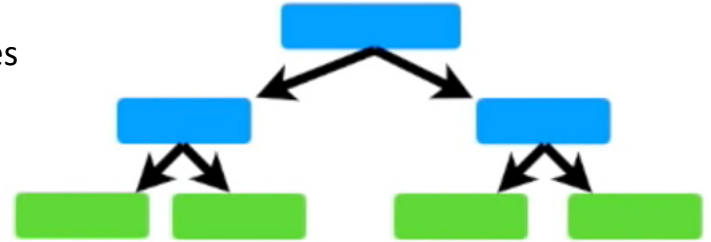
vs.

Random Forest

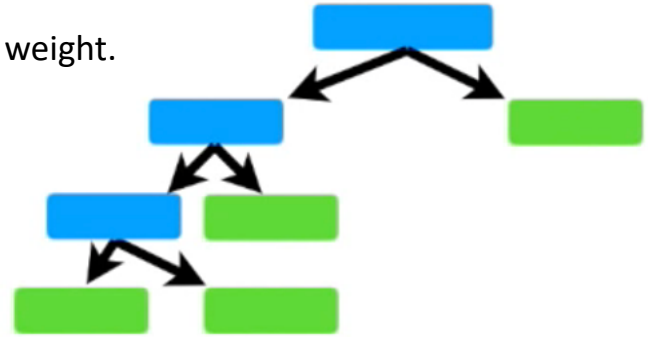
Create full trees



Independent trees



Each tree has equal weight.



Gradient Boost

Input: Data $\{(x_i, y_i)\}_{i=1}^{n}$, and a differentiable **Loss Function** $L(y_i, F(x))$

Step 1: Initialize model with a constant value: $F_0(x) = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, \gamma)$

Step 2: for $m = 1$ to M:

(A) Compute $r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$ for $i = 1,\ldots,n$

(B) Fit a regression tree to the r_{im} values and create terminal regions R_{jm}, for $j = 1 \ldots J_m$

(C) For $j = 1 \ldots J_m$ compute $\gamma_{jm} = \underset{\gamma}{\operatorname{argmin}} \sum_{x_i \in R_{ij}} L(y_i, F_{m-1}(x_i) + \gamma)$

(D) Update $F_m(x) = F_{m-1}(x) + \nu \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$

Step 3: Output $F_M(x)$

$$\sum_{i=1}^{N} \boldsymbol{y_i} \times \log(\boldsymbol{p}) + (1 - \boldsymbol{y_i}) \times \log(1 - \boldsymbol{p})$$

Gradient Boost

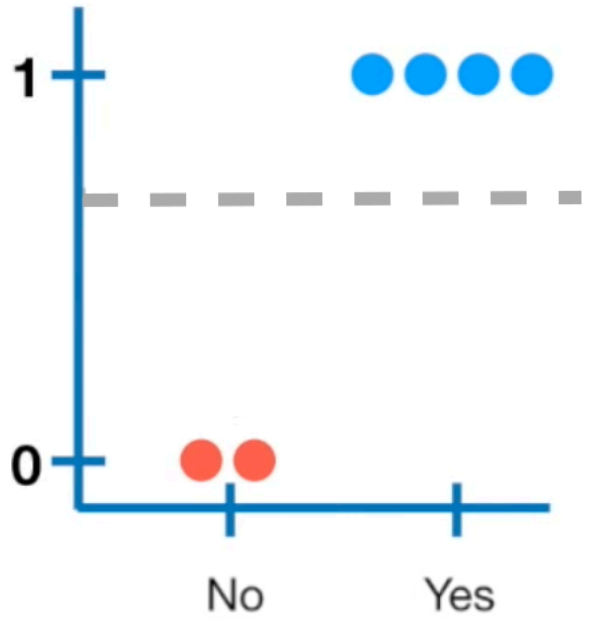
Chest Pain	Age	Color	Heart Disease	Residual
Yes	12	Blue	Yes	0.3
Yes	87	Green	Yes	0.3
No	44	Blue	No	-0.7
Yes	19	Red	No	-0.7
No	32	Green	Yes	0.3
No	14	Blue	Yes	0.3

1. Initialise predication value

$$\log(\text{odds}) = \log\frac{\#p}{\#n} = 0.69314 \approx 0.7$$

$$p(X) = \frac{e^{\log\frac{\#p}{\#n}}}{1 + e^{\log\frac{\#p}{\#n}}} = 0.667 \approx 0.7$$

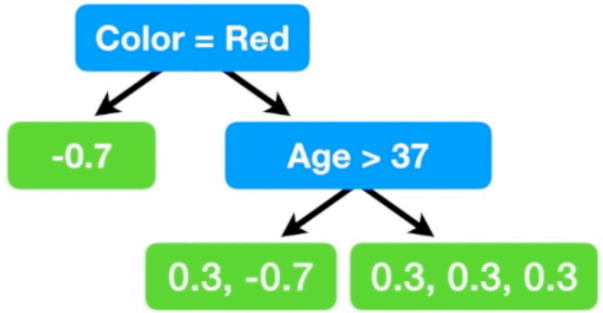
2. Calculate (pseudo) residuals



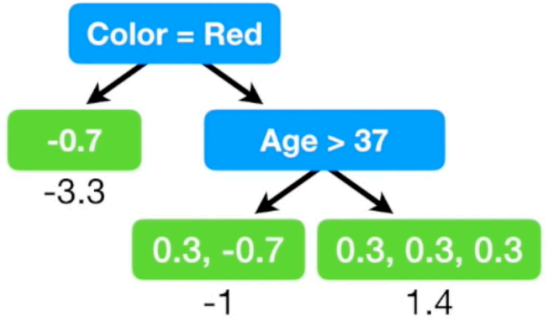
Gradient Boost

Chest Pain	Age	Color	Heart Disease	Residual
Yes	12	Blue	Yes	0.3
Yes	87	Green	Yes	0.3
No	44	Blue	No	-0.7
Yes	19	Red	No	-0.7
No	32	Green	Yes	0.3
No	14	Blue	Yes	0.3

3. Build tree to **predict residuals**



4. Calculate output value



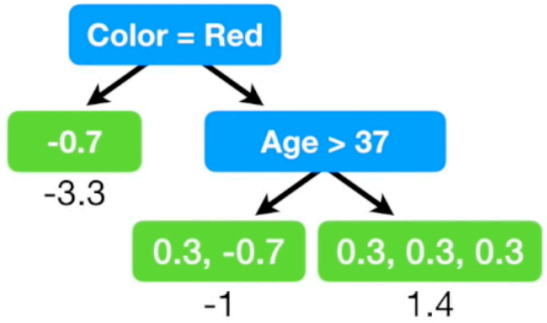
Gradient Boost

Chest Pain	Age	Color	Heart Disease	Predicted Prob.
Yes	12	Blue	Yes	0.9
Yes	87	Green	Yes	0.5
No	44	Blue	No	0.5
Yes	19	Red	No	0.1
No	32	Green	Yes	0.9
No	14	Blue	Yes	0.9

5. Update log-odds.

$$\text{new } \log(\text{odds}) = \log(\text{odds}) + \alpha \times$$

α – learning rate



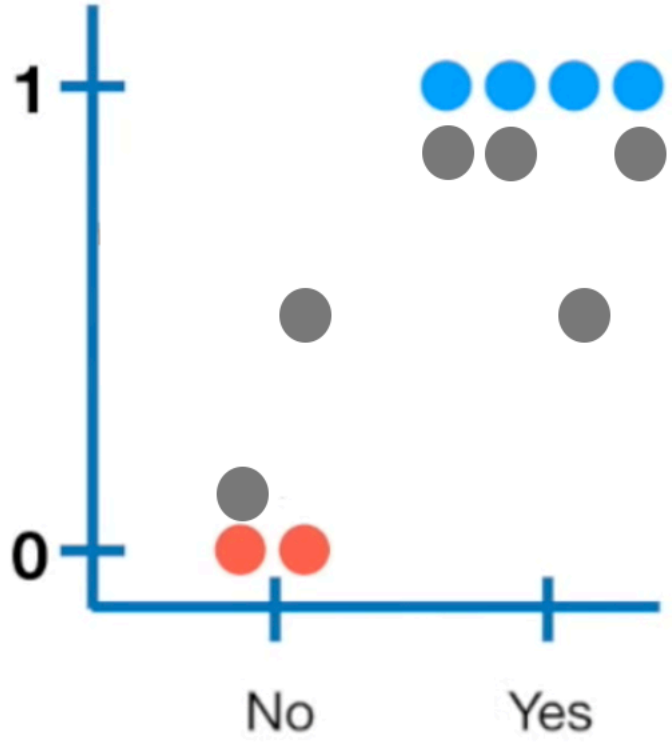
6. Calculate probabilities.

$$p_{new}(X) = \frac{e^{\log(odds)}}{1 + e^{\log(odds)}}$$

7. Calculate residuals.

Gradient Boost

Chest Pain	Age	Color	Heart Disease	Predicted Prob.	Residual
Yes	12	Blue	Yes	0.9	0.1
Yes	87	Green	Yes	0.5	0.5
No	44	Blue	No	0.5	-0.5
Yes	19	Red	No	0.1	-0.1
No	32	Green	Yes	0.9	0.1
No	14	Blue	Yes	0.9	0.1

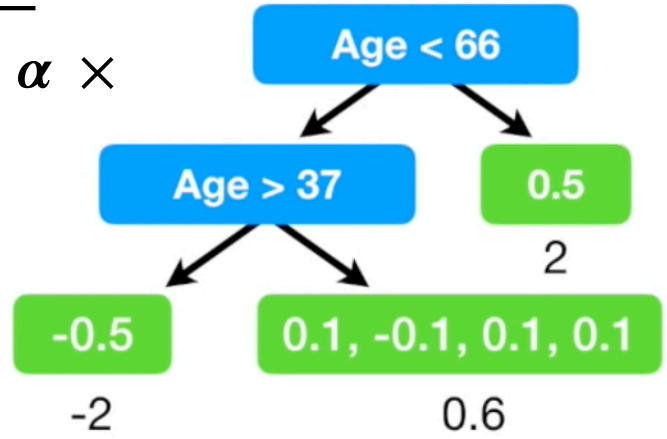


8. Repeat.

Gradient Boost

Chest Pain	Age	Color	Heart Disease	Residual
Yes	12	Blue	Yes	0.1
Yes	87	Green	Yes	0.5
No	44	Blue	No	-0.5
Yes	19	Red	No	-0.1
No	32	Green	Yes	0.1
No	14	Blue	Yes	0.1

$$\text{new } \log(\text{odds}) = \log(\text{odds}) + \alpha \times$$



$$p_{new}(X) = \frac{e^{\log(odds)}}{1 + e^{\log(odds)}}$$

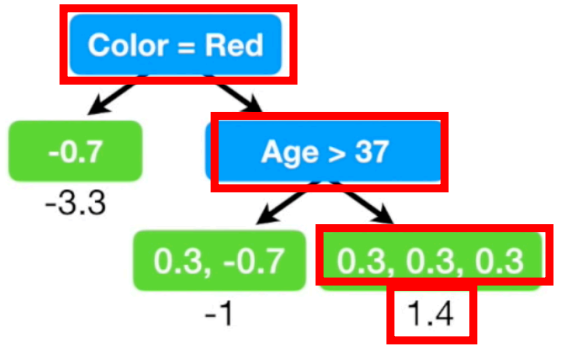
residuals

Gradient Boost

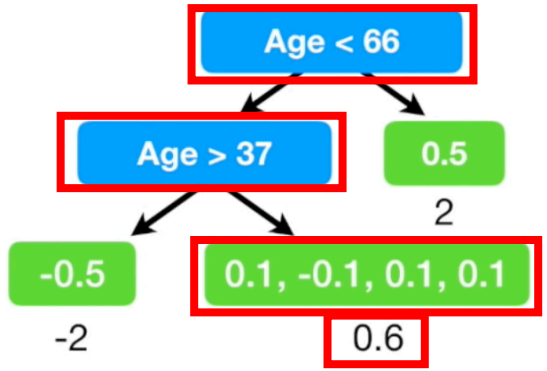
Chest Pain	Age	Color
Yes	25	Green

Predictions

$$\text{initial } \log(\text{odds}) + \alpha \times$$



$$\alpha \times$$



$$\text{pred } \log(\text{odds})$$

$$p_{pred}(X) = \frac{e^{\text{pred } \log(\text{odds})}}{1 + e^{\text{pred } \log(\text{odds})}}$$

Summary

Metrics:

- Confusion Matrix (TP/FP/TN/FN)
- Accuracy
- Precision/Recall/F-score
- ROC-AUC

Machine Learning:

- Logistic Regression
- Support Vector Machine (+kernel trick)
- K Nearest Neighbours
- Naïve Bayes (Gaussian/Multinomial)
- Decision Tree
- Random Forrest
- Boosting (AdaBoost/Gradient Boost)

SVM vs. Logistic Regression
Random Forrest vs. Decision Trees
Random Forrest vs. Gradient Boost
Boosting vs. Bagging

Random in Random Forrest

Imbalanced Dataset

Regularisation:

- Lasso (L1)
- Ridge (L2)
- ElasticNet

Sklearn

- Pipelines
- Grid Search
- Custom Estimator

(Stochastic) Gradient Descent

TODO:

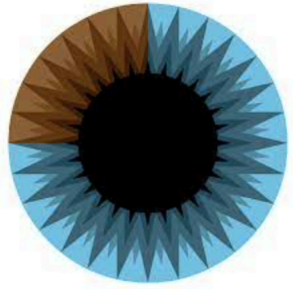
- XGBoost

https://machinelearningmastery.com

towards
data science

https://towardsdatascience.com

https://www.youtube.com/c/joshstarmer



https://www.youtube.com/c/3blue1brown

https://www.youtube.com/c/TechWorldwithNana

https://www.youtube.com/c/TechWithTim

kaggle

Machine Learning Study Groups

https://www.youtube.com/channel/UCMEQFEKrsRFBXnUIreTACxg

TensorFlow

https://www.youtube.com/c/TensorFlow