Confusion Matrix

Sklearn Representation

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

A)

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

B)

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

C)

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

D)

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

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

F_β-Score

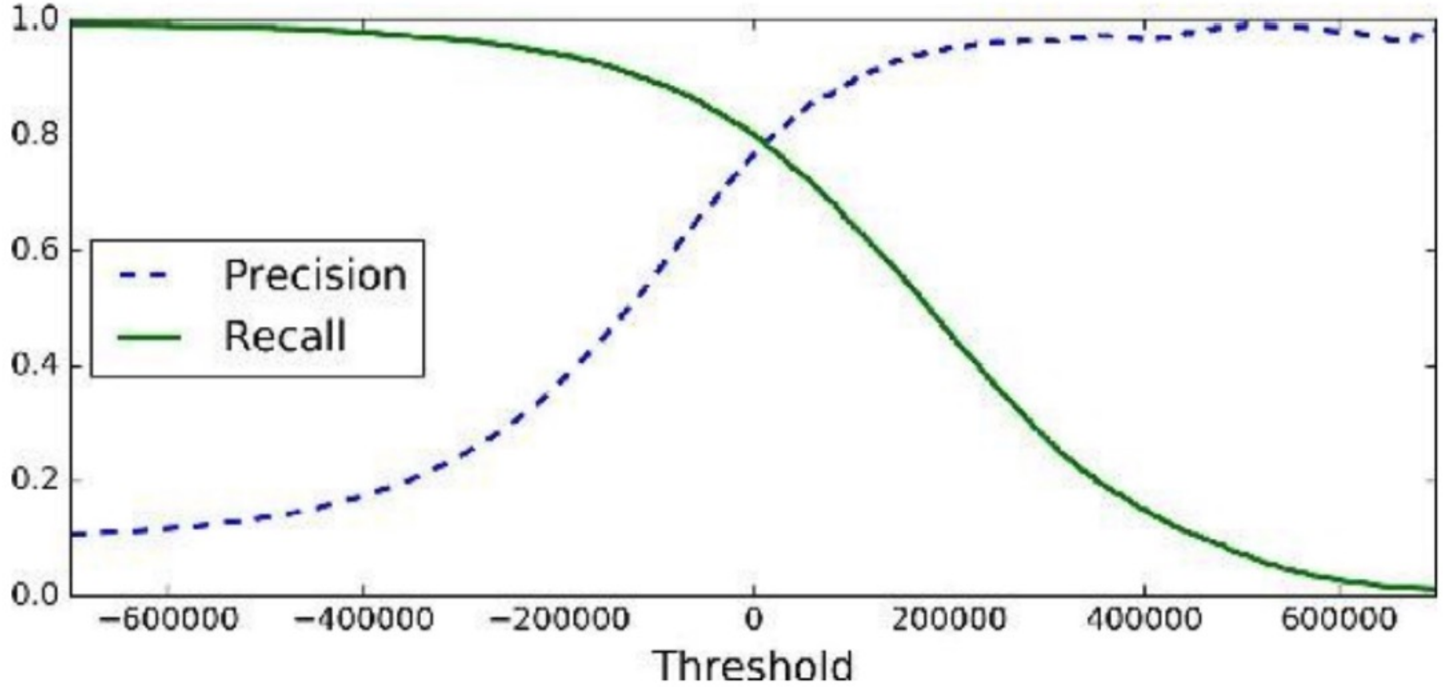
$$F_\beta = (1+\beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}}$$

$\beta = 1$

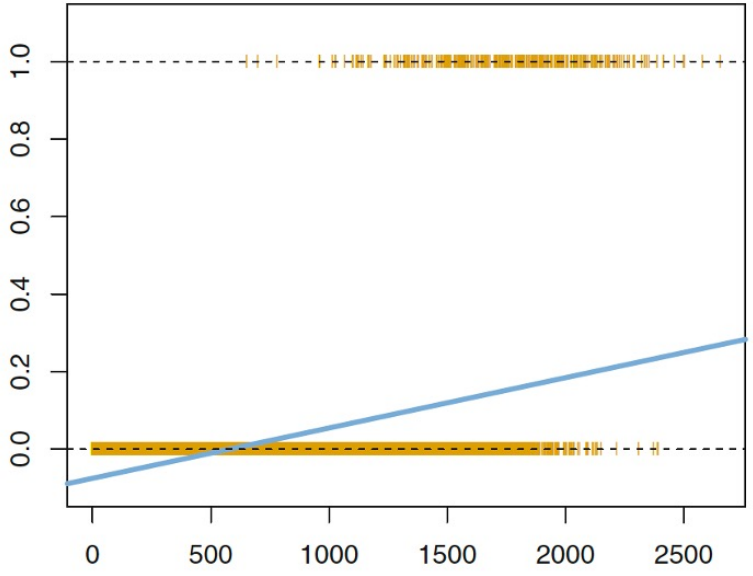
$$F_1 = \frac{2}{\text{recall}^{-1} + \text{precision}^{-1}} = 2\frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{\text{tp}}{\text{tp} + \frac{1}{2}(\text{fp} + \text{fn})}$$

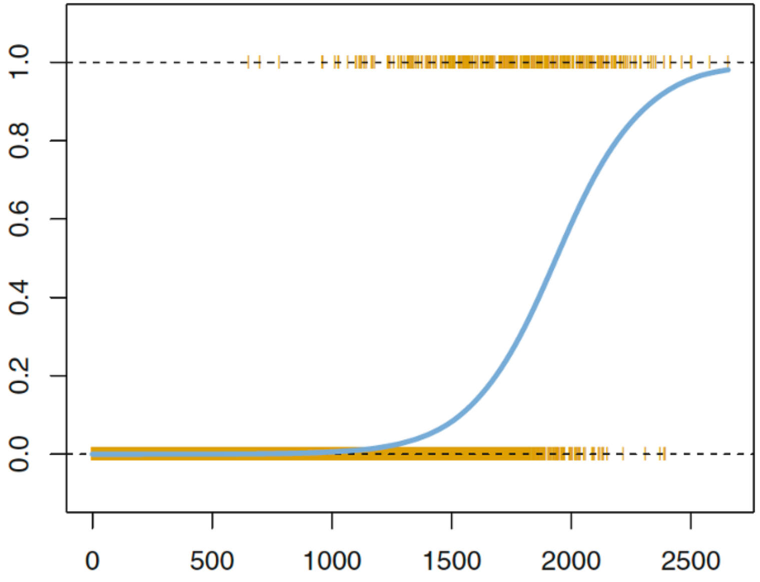
harmonic mean

Precission – Racall 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

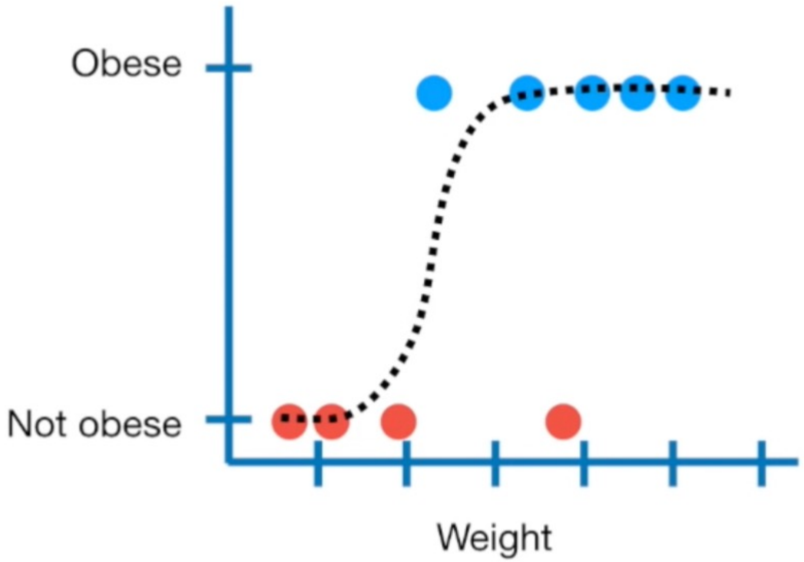
$$h_\theta(x) = \frac{1}{1 + e^{-(\theta^T x)}} = \frac{1}{1 + e^{-(\theta_0 + \theta_1 x)}}$$

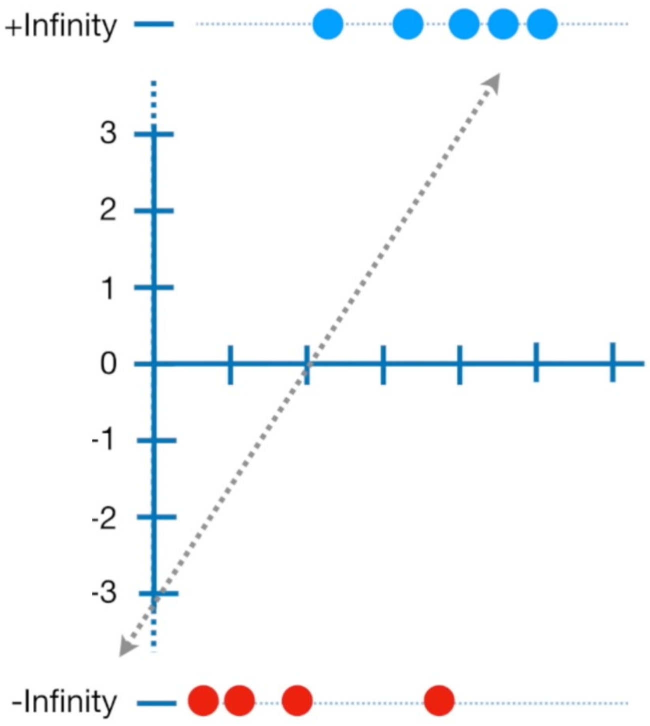
$$\frac{h_\theta(x)}{1 - h_\theta(x)} = e^{(\theta_0 + \theta_1 x)}$$

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

(log odds)

Logistic Regression

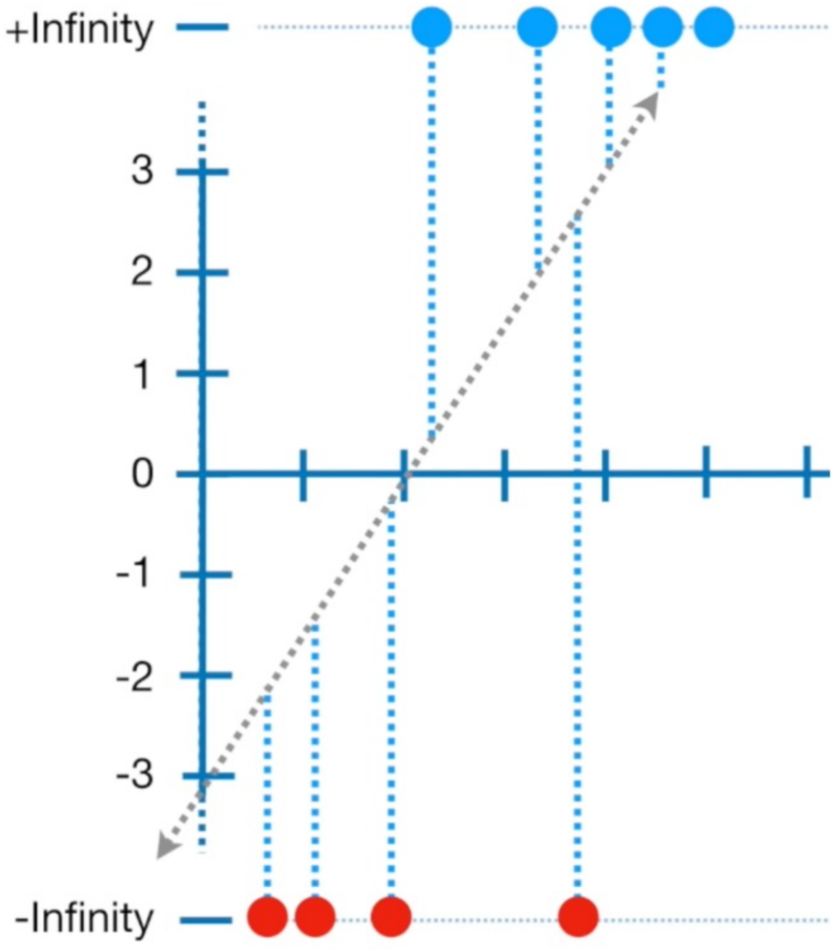


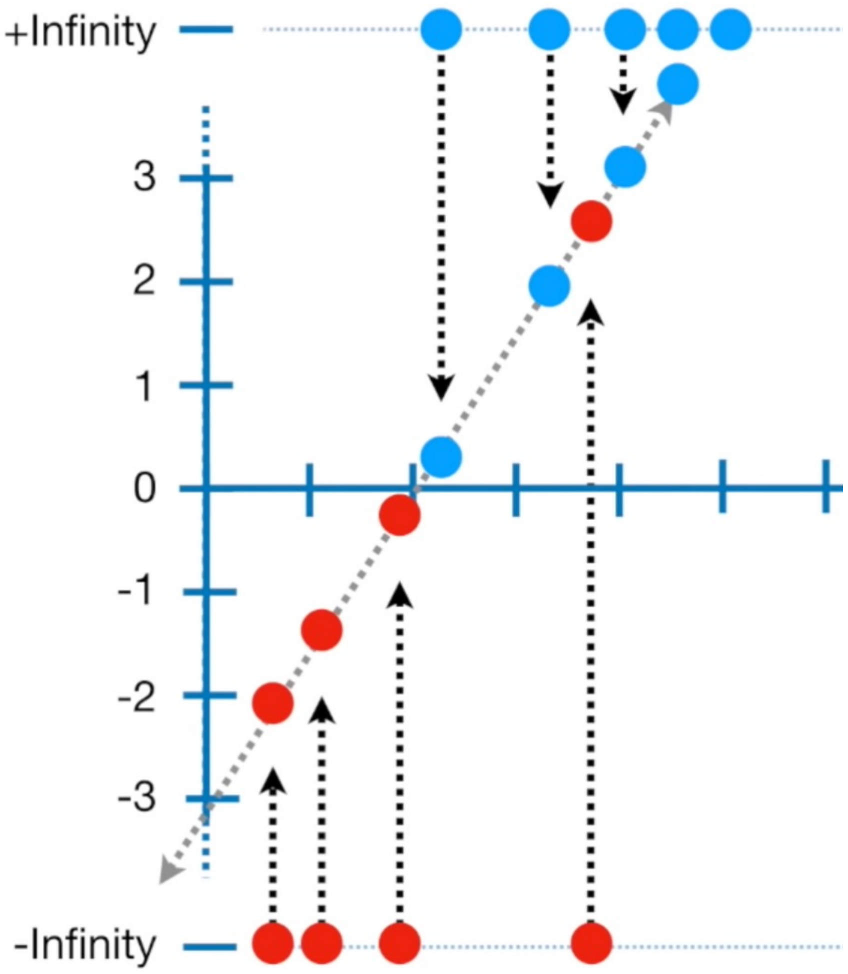


$$h_\theta(x) = \frac{1}{1 + e^{-(\theta_0 + \theta_1 x)}}$$

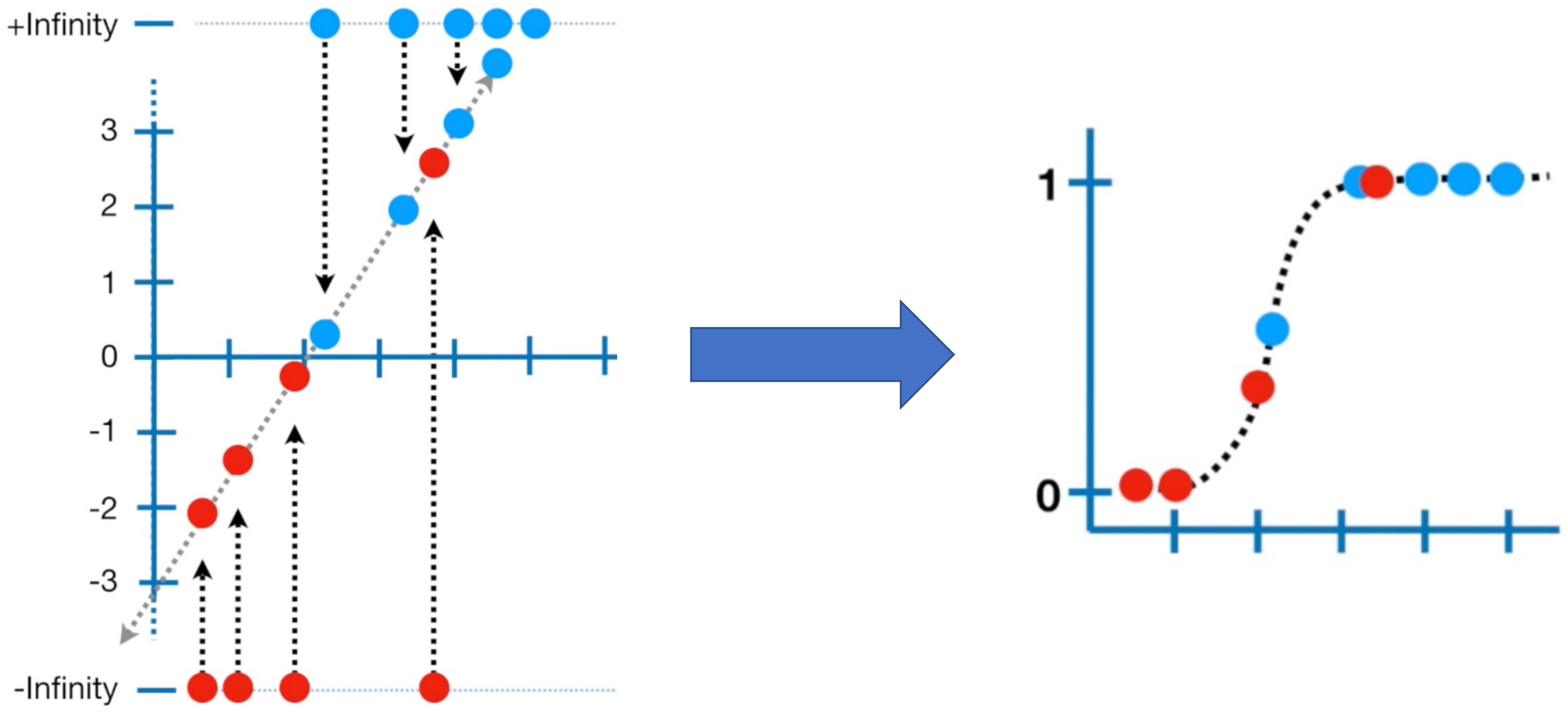
$$\log\left(\frac{h_\theta(x)}{1 - h_\theta(x)}\right) = \theta_0 + \theta_1 x$$

Logistic Regression



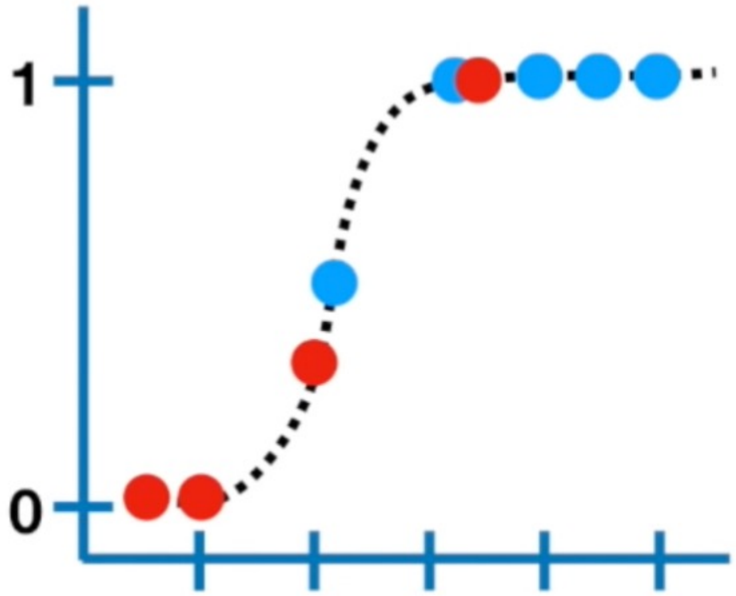


Logistic Regression



Logistic Regression

MLE – maximum likelihood estimation:



Likelihood:

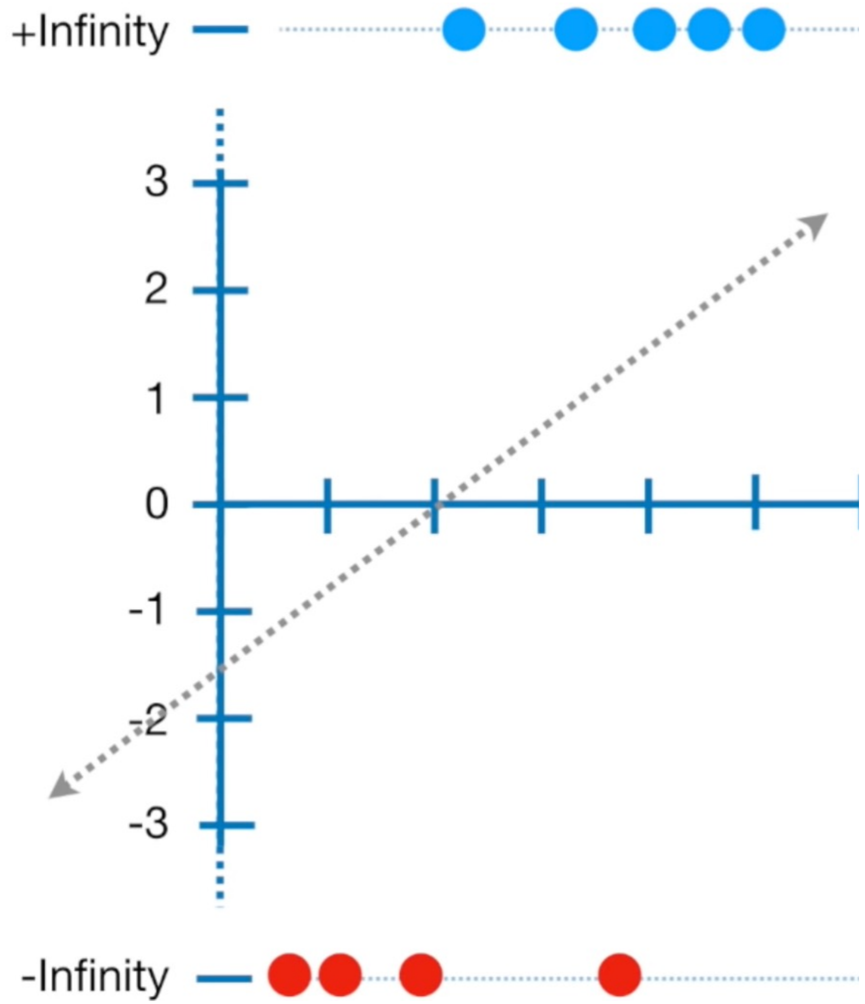
$$\ell(\beta_0, \beta_1) = \prod_{i:y_i=1} p(x_i) \prod_{i^{i}:y_{i'}=0} (1 - p(x_{i'}))$$

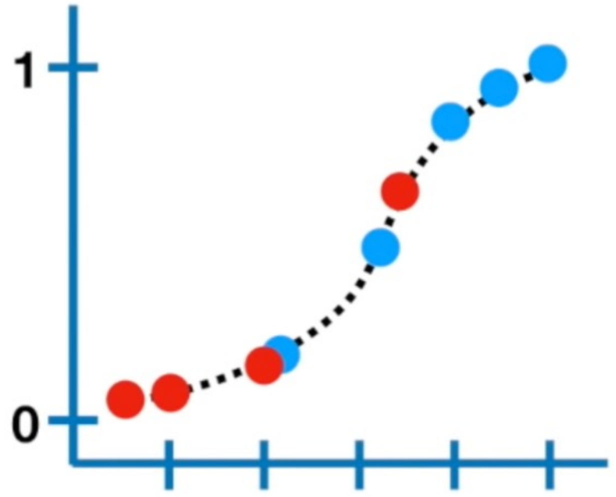
Log-likelihood:

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

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

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:

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

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

non – convex

Logistic Regression:

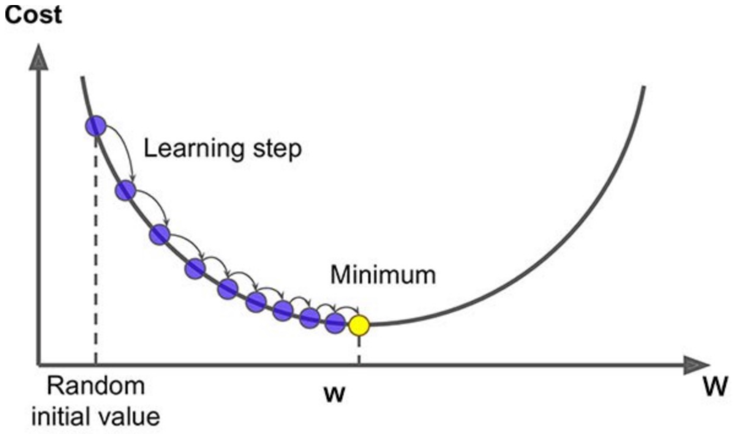
$$\mathrm{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

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

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

cross-entropy

Gradient prosty *Gradient descent*



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

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

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

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

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

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

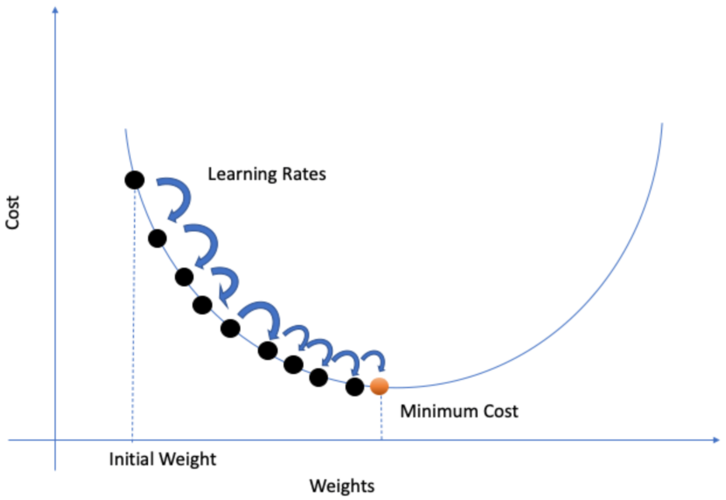
$$\boldsymbol{x}_{\boldsymbol{i}} = [x_{i,1} \quad x_{i,2} \quad \cdots \quad x_{i,n}]$$

```python
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*



$$i - ustalone, losowe$$

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

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

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

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

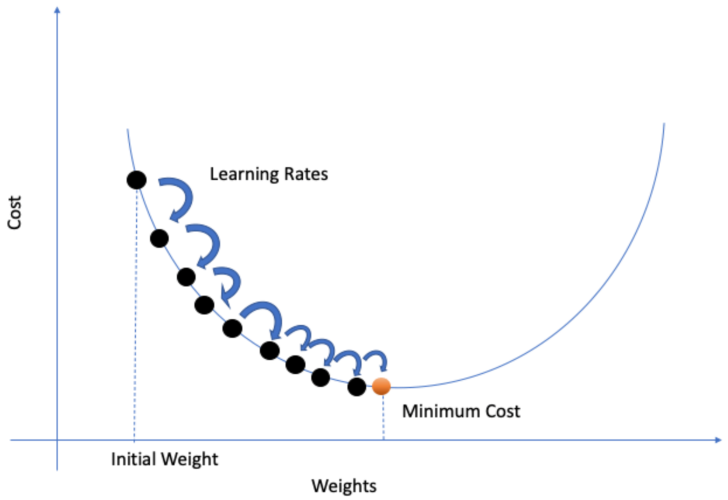
$$\eta - learning\ rate\ (0.01)$$

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

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

```
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*



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

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

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

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

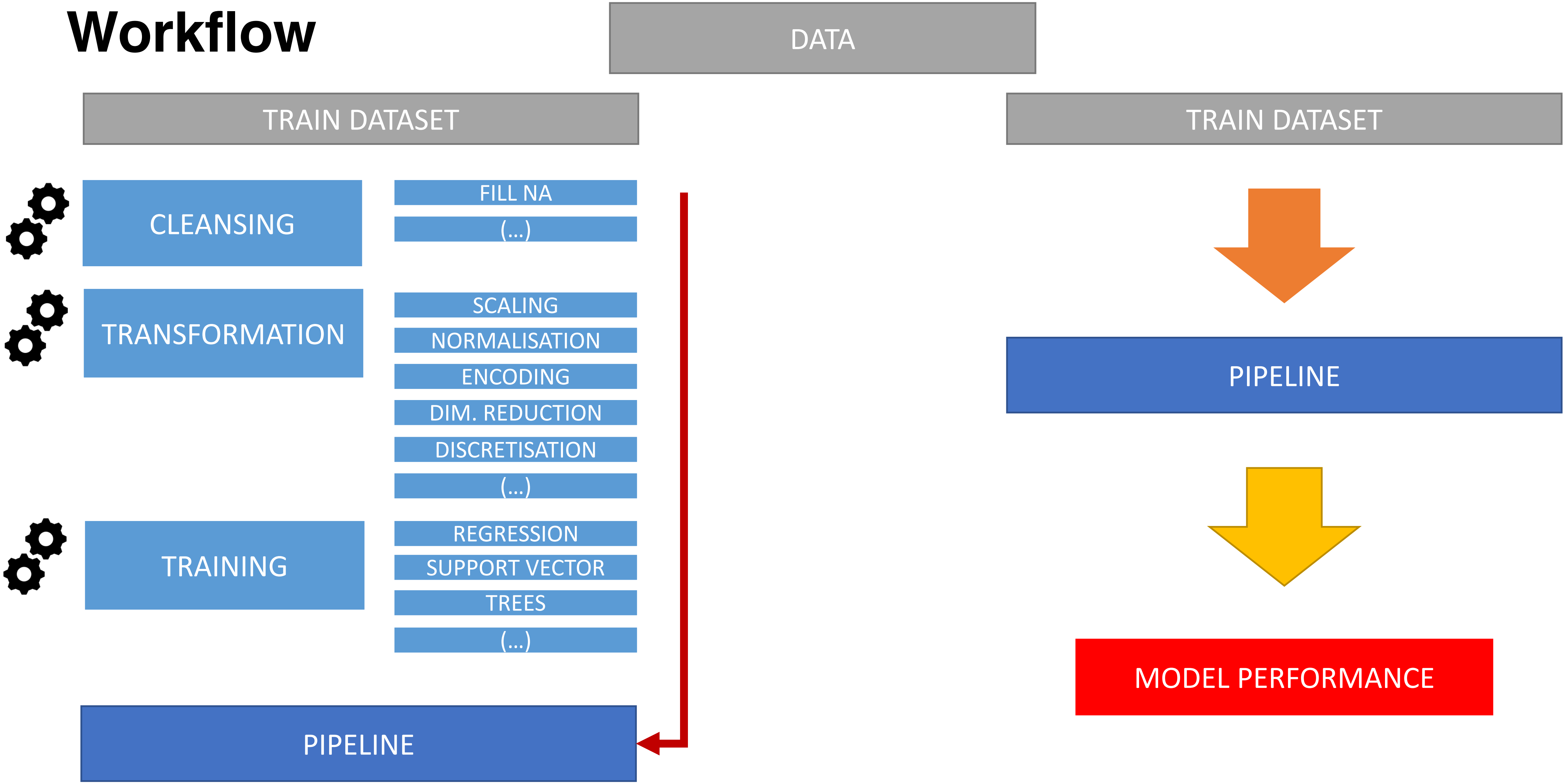
$$I - ustalone, losowy\ podzbiór\ X$$

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

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

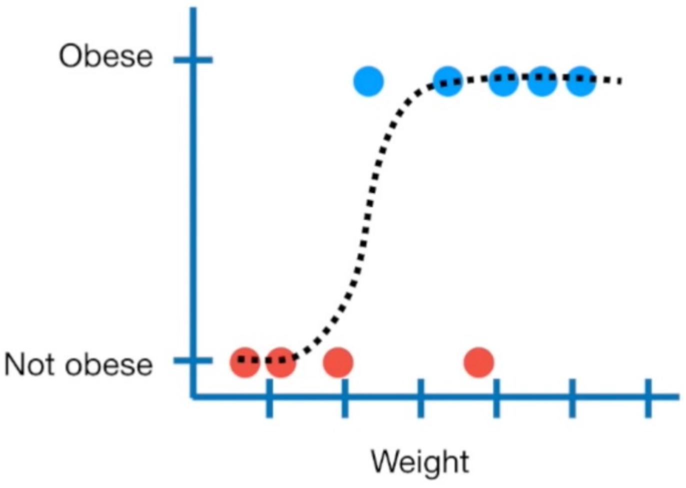
$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i) x_{i,j}$$

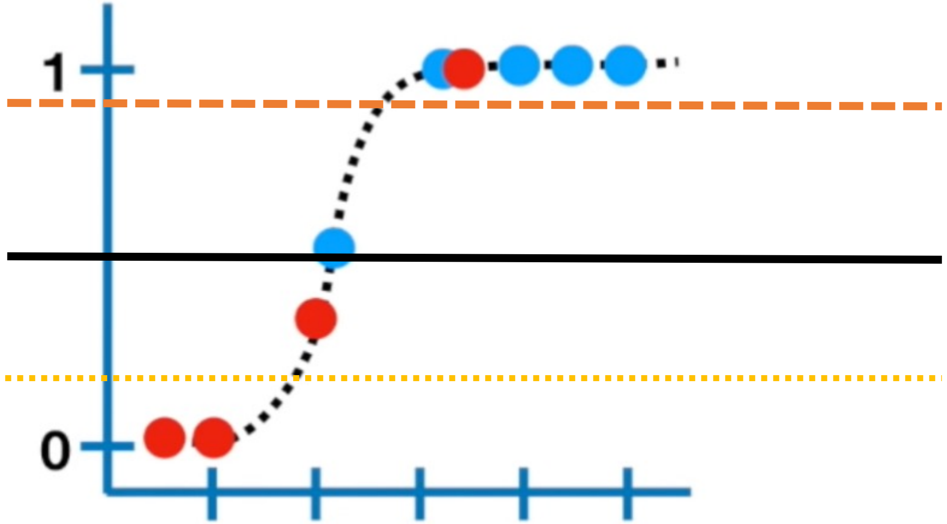
Workflow



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



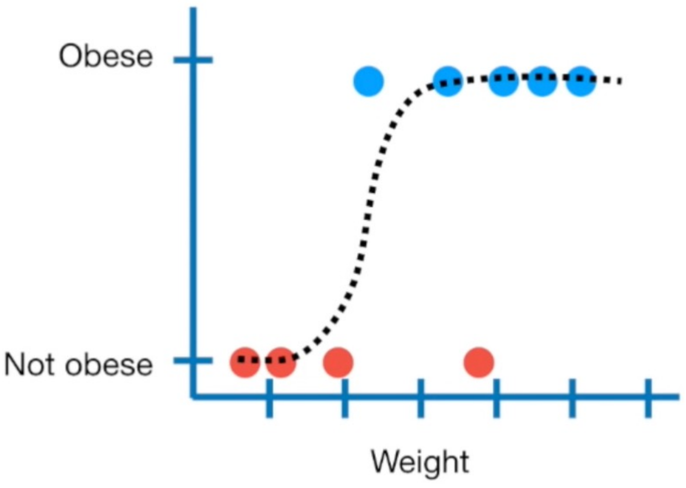


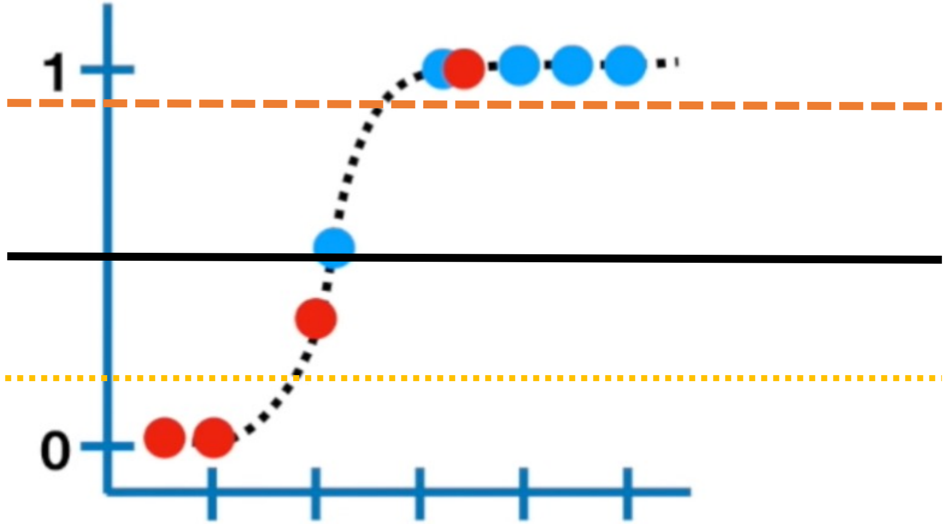
		ACTUALS	
		1	0
PREDICTIONS	1		
	0		

		ACTUALS	
		1	0
PREDICTIONS	1		
	0		

		ACTUALS	
		1	0
PREDICTIONS	1		
	0		

ROC-AUC





ACTUALS

PREDICTIONS	1	0
1	4	1
0	1	3

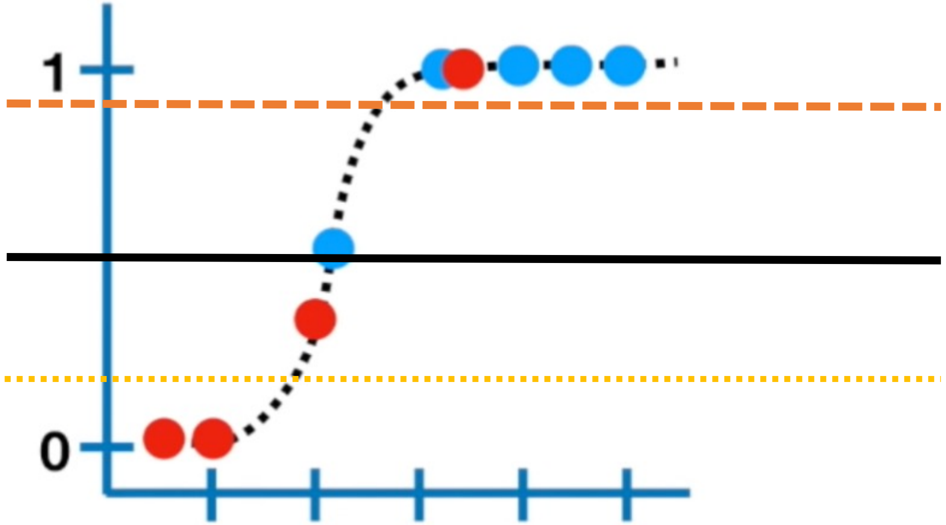
ACTUALS

PREDICTIONS	1	0
1	5	0
0	1	3

ACTUALS

PREDICTIONS	1	0
1	5	0
0	2	2

ROC-AUC



		ACTUALS 1	ACTUALS 0
PREDICTIONS	1	4	1
	0	1	3

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

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

		ACTUALS 1	ACTUALS 0
PREDICTIONS	1	5	0
	0	1	3

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

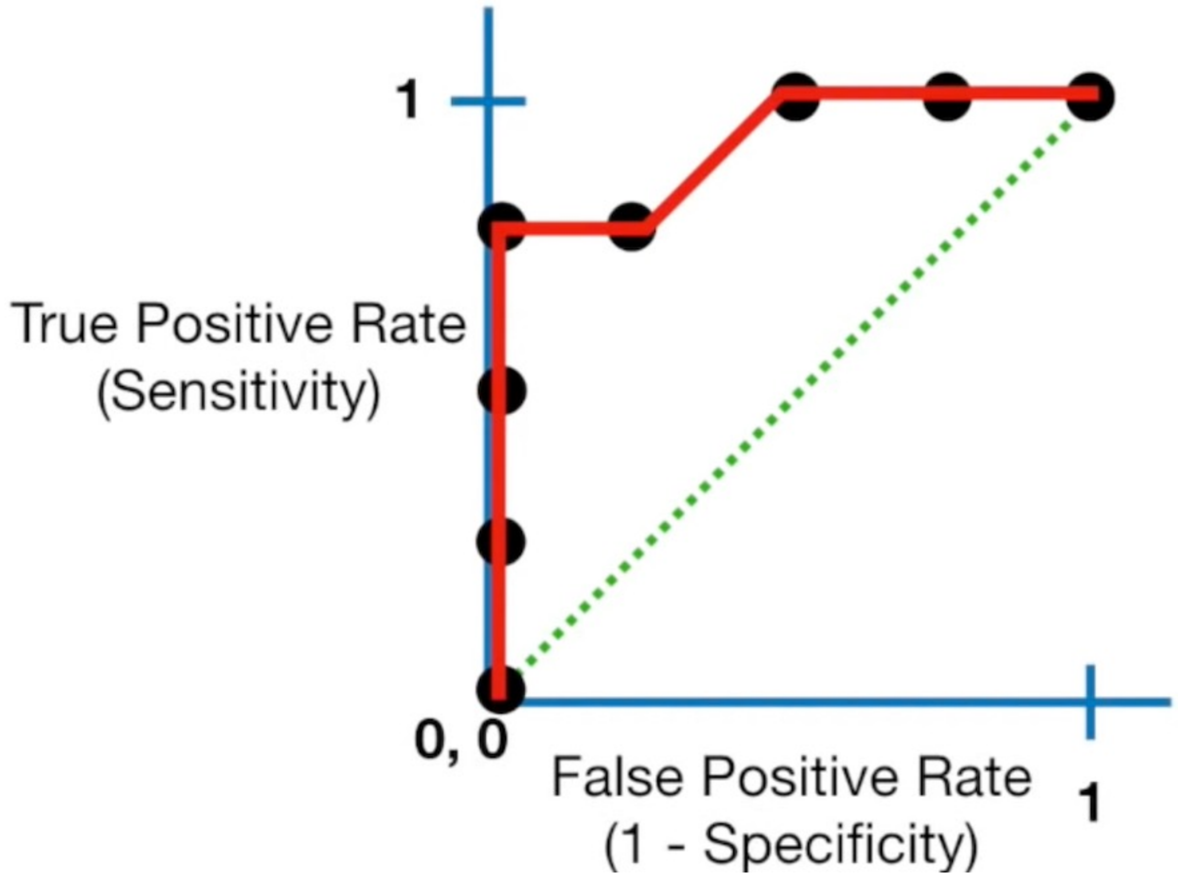
$$FPR = \frac{0}{0+3}$$

		ACTUALS 1	ACTUALS 0
PREDICTIONS	1	5	0
	0	2	2

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

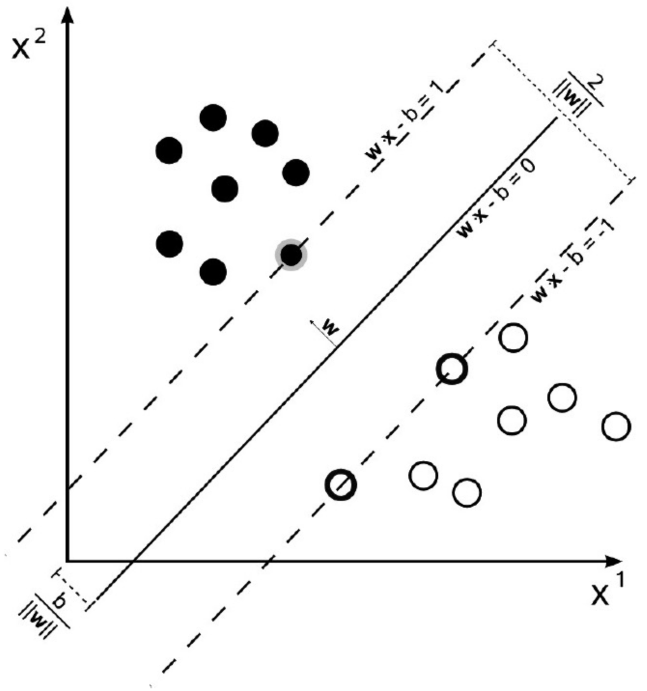
$$FPR = \frac{0}{0+2}$$

ROC-AUC



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

SVM – Hard Margin



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

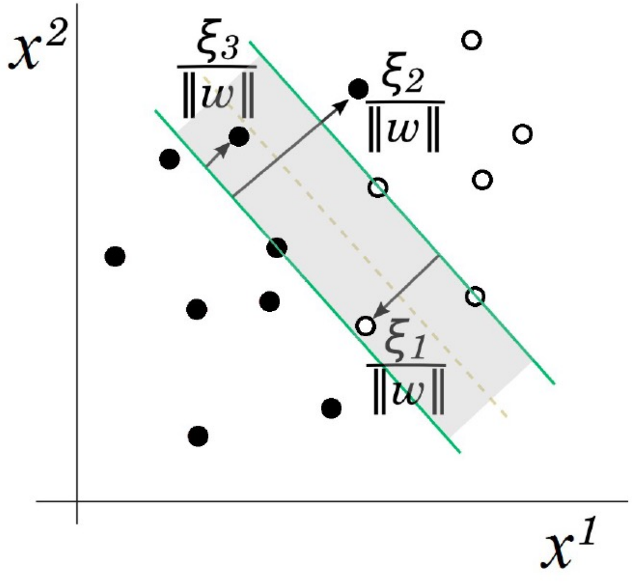
$$(w, b) = \underset{w, b}{\arg\min} ||w||^2$$

Constraints:

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

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

SVM – Soft Margin



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

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

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

$$\lambda > 0$$

Constraints, for each i:

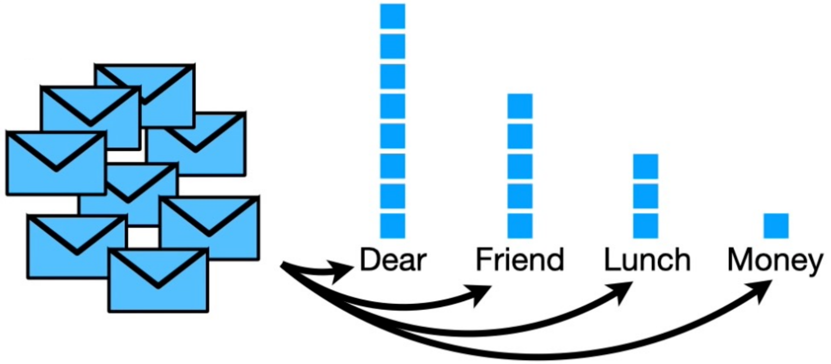
$$\xi_i \geq 0$$
$$(w^T x_i - b) y_i \geq 1 - \xi_i$$

Naïve Bayes

Likelihoods

Dear Friend 

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



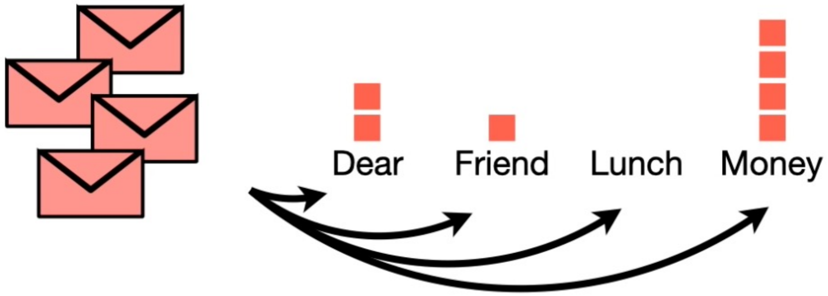
p(**Dear** | **N**) = 0.47
p(**Friend** | **N**) = 0.29
p(**Lunch** | **N**) = 0.18
p(**Money** | **N**) = 0.06

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

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

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

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



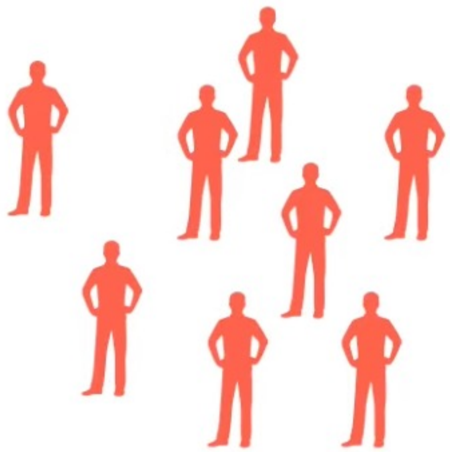
p(**Dear** | **S**) = 0.29
p(**Friend** | **S**) = 0.14
p(**Lunch** | **S**) = 0.00
p(**Money** | **S**) = 0.57

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

Naïve Bayes - Gaussian

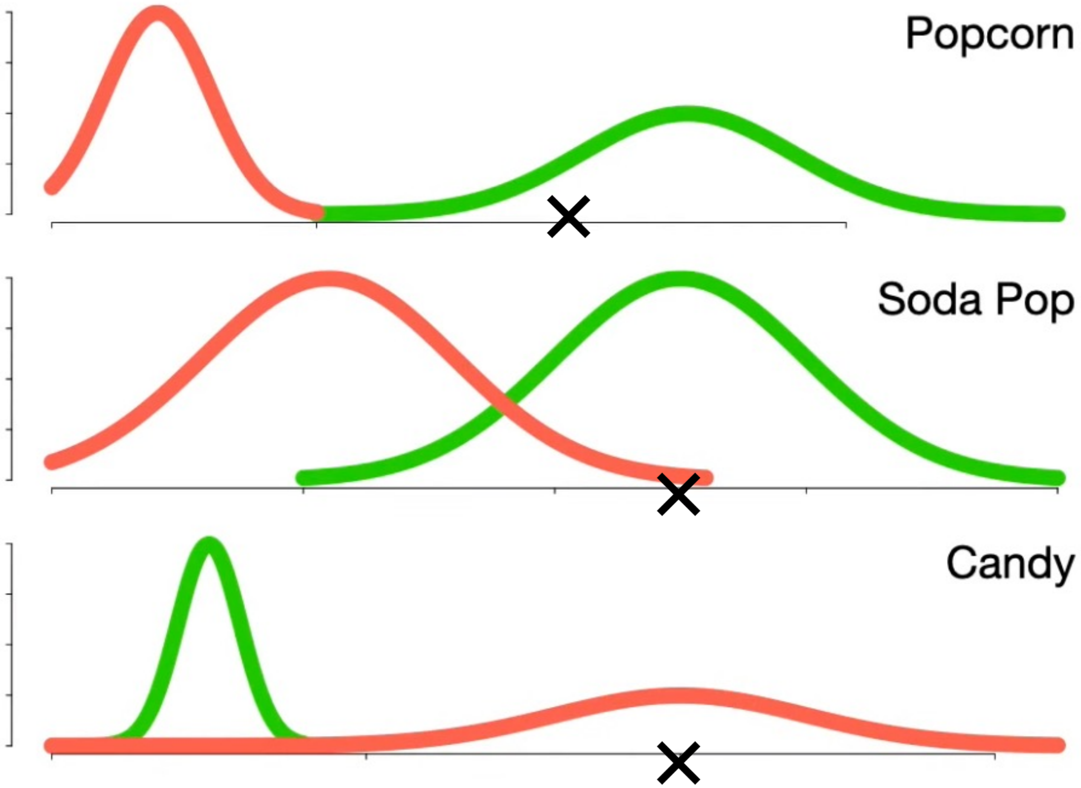
$$p(G) = \frac{\#G}{\#total}$$

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.



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

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

Decission Tree - CART

Gini:

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

Log Loss or Entropy:

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

Random Forrest

Step 1: Bootstraping

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 Forrest

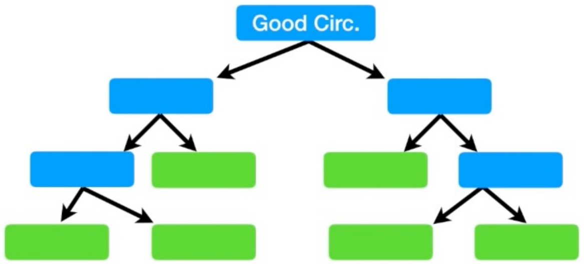
Step 2: Create Decision Tree .

Randomly select subset of features.

Find best split.

Randomly select subset of features to node split.

Create Tree considering subset of features.

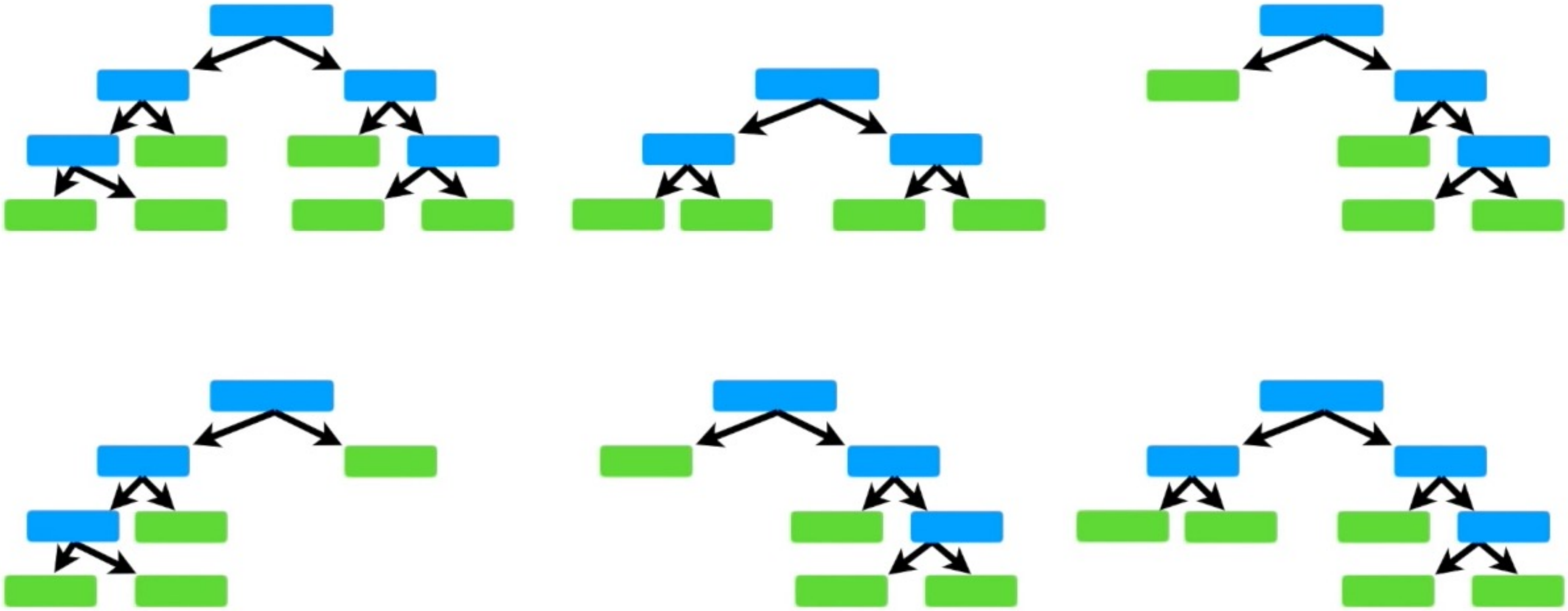


Bootstrapped Dataset

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 Forrest

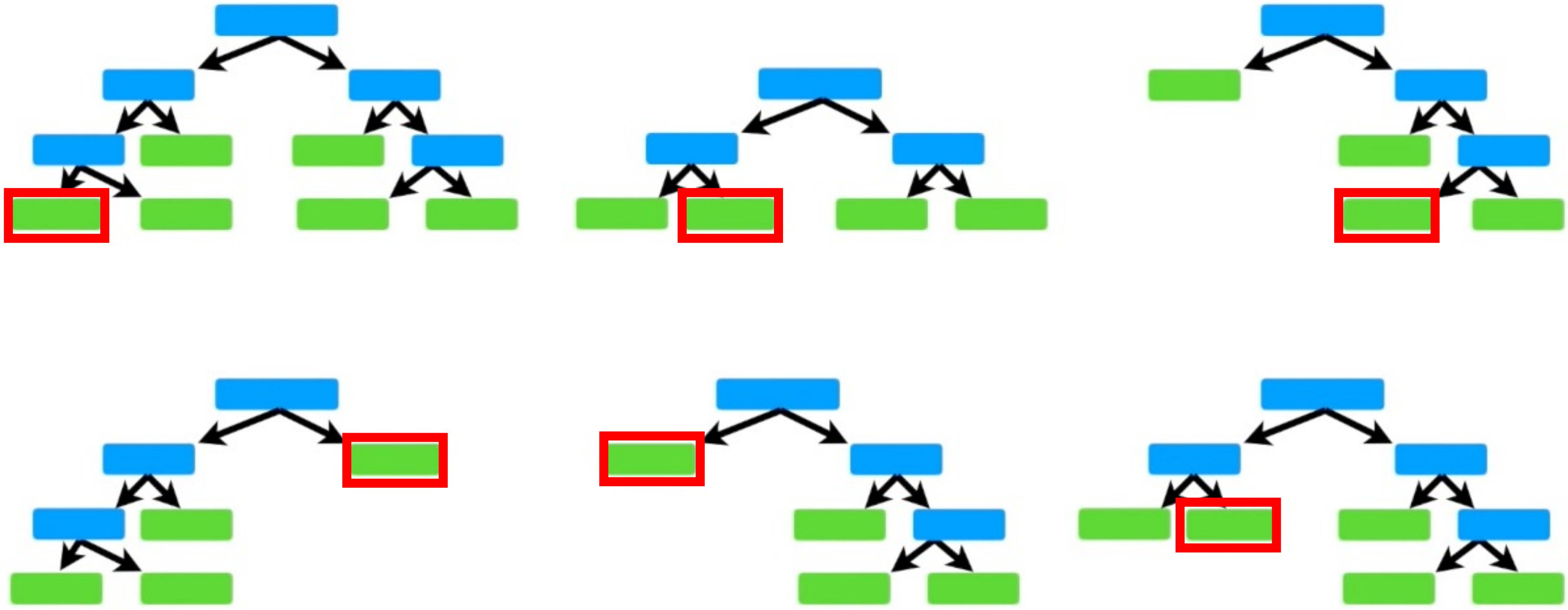
Repeat Step 1 & Step 2 creating next trees.



Random Forrest

Predictions.

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



Random Forrest

Out-Of-Bag Dataset

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

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

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.*

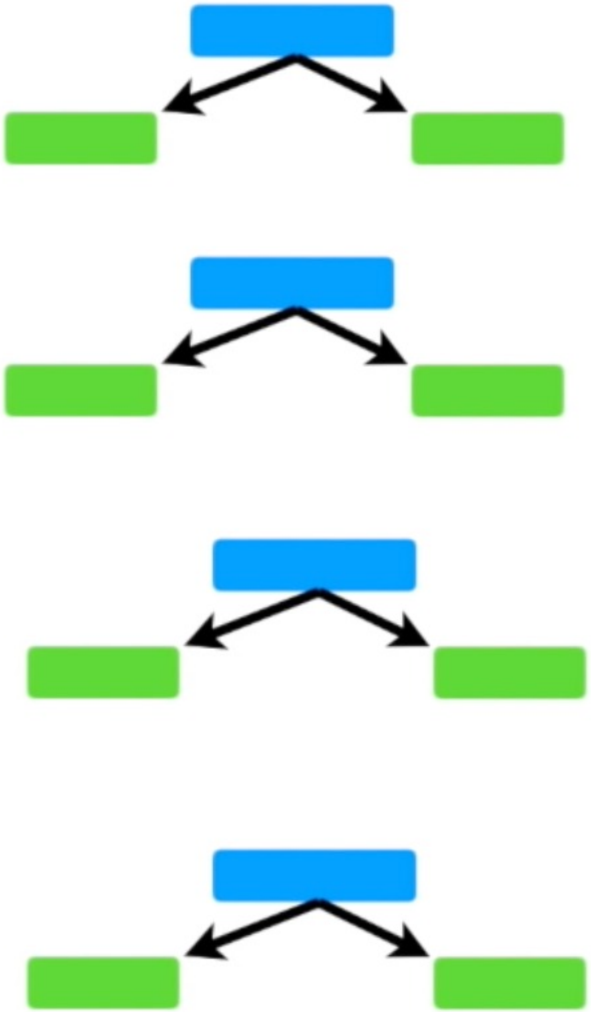
oob_score_ : ***float***

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

AdaBoost

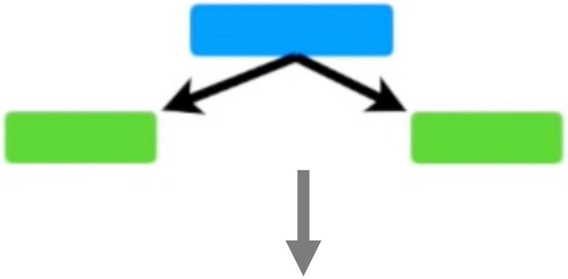
Original Dataset

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

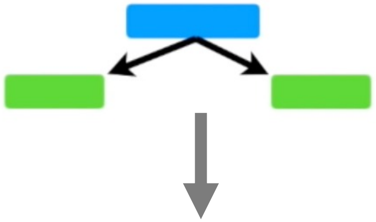


stump

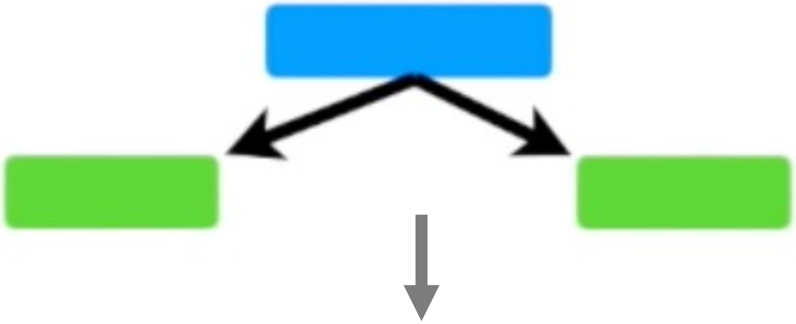
AdaBoost



Create stumps

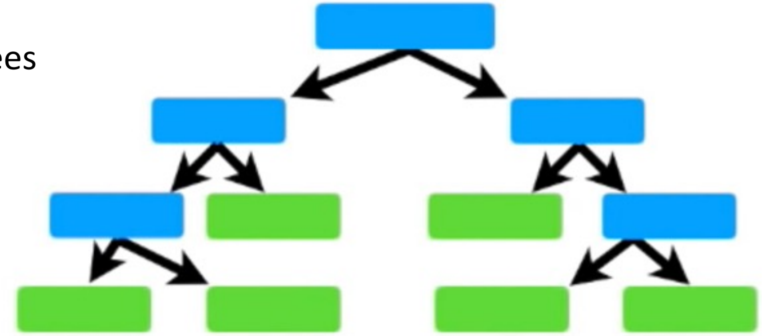


Stumps create based on previous stump.

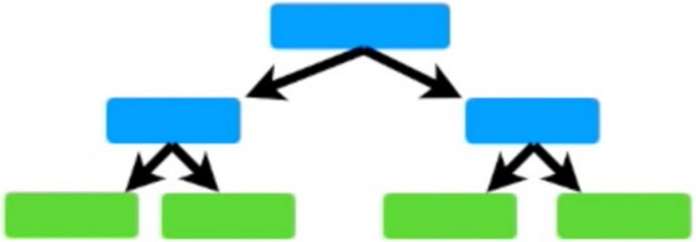


Stumps have no equal weights.

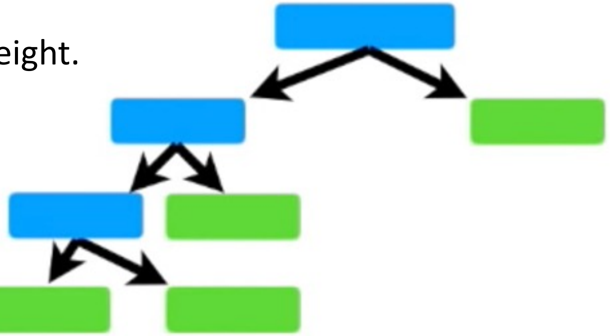
Create full trees



Independent trees



Each tree has equal weight.



Step 1. Find best split minimizing Gini

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

AdaBoost

Step 1. Find best split minimizing Gini

Step 2. Update sample weights.

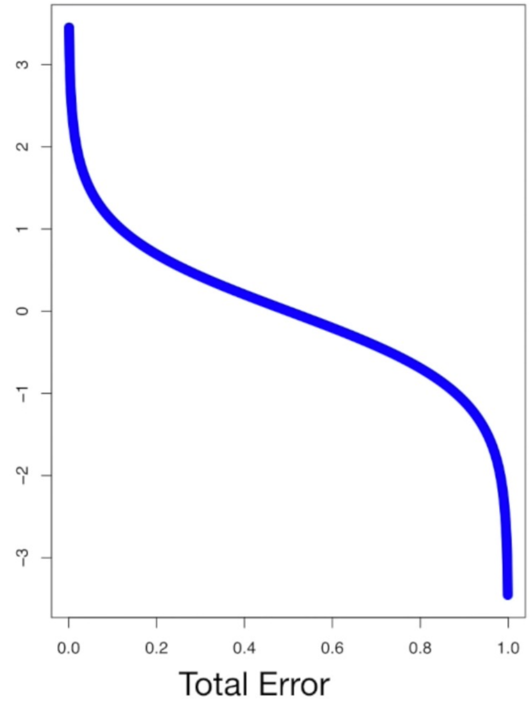
Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	New Weight
Yes	Yes	205	Yes	0.05
No	Yes	180	Yes	0.05
Yes	No	210	Yes	0.05
Yes	Yes	167	Yes	0.33
No	Yes	156	No	0.05
No	Yes	125	No	0.05
Yes	No	168	No	0.05
Yes	Yes	172	No	0.05

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

$total\_error$ - 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

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
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

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.

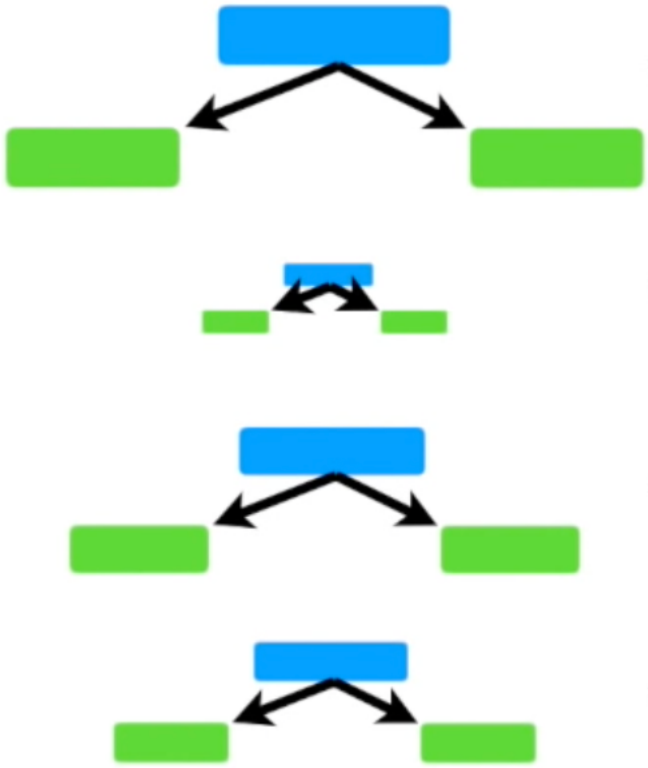
Step 5. Repeat using bootstrap dataset.

AdaBoost

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

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

Calculate I



0.97

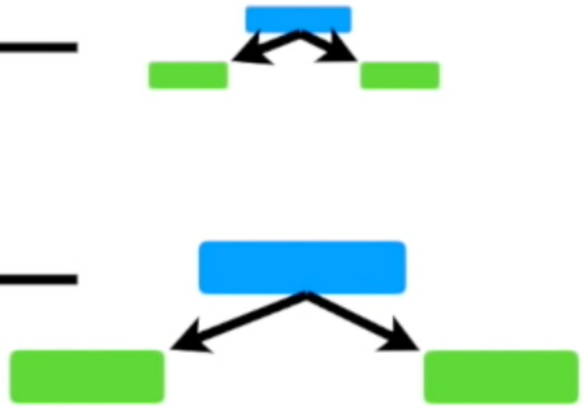
0.32

0.78

0.63

Total=2.7

Calculate I



0.41

0.82

Total=1.23

Grsdient Boost

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

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

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 $F_M(x)$

Grsdient Boost

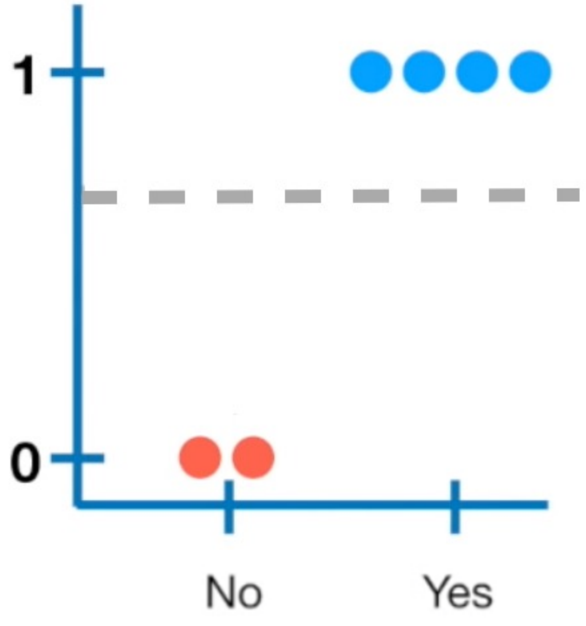
Chest Pain	Age	Color	Heart Disease	Residual
Yes	12	Blue	Yes	0.3
Yes	87	Green	Yes	0.3
No	44	Blue	No	-0.7
Yes	19	Red	No	-0.7
No	32	Green	Yes	0.3
No	14	Blue	Yes	0.3

1. Initialise predication value

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

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

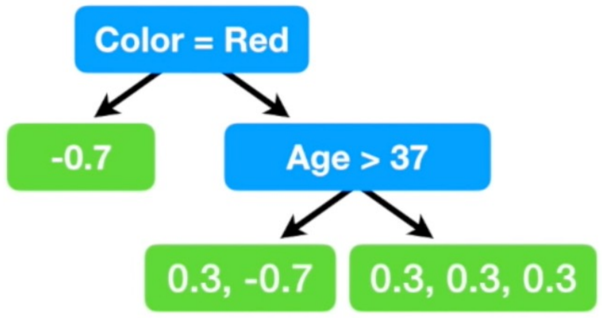
2. Calculate (pseudo) residuals



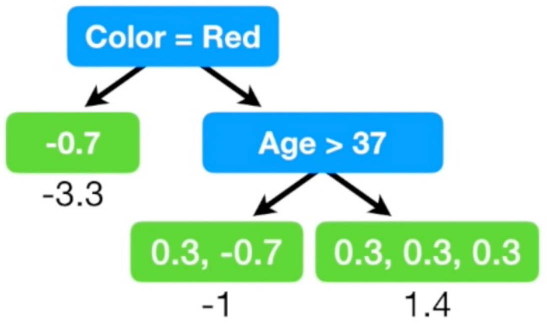
Grsdient Boost

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

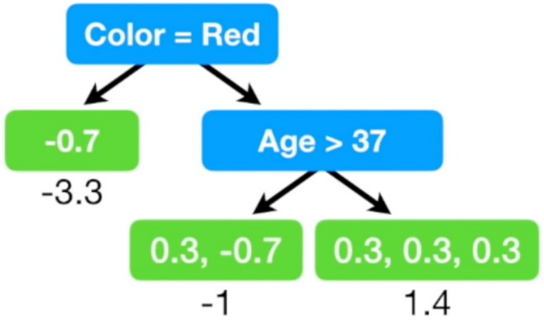


Grsdient Boost

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.

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



$\boldsymbol{\alpha}$ – learning rate

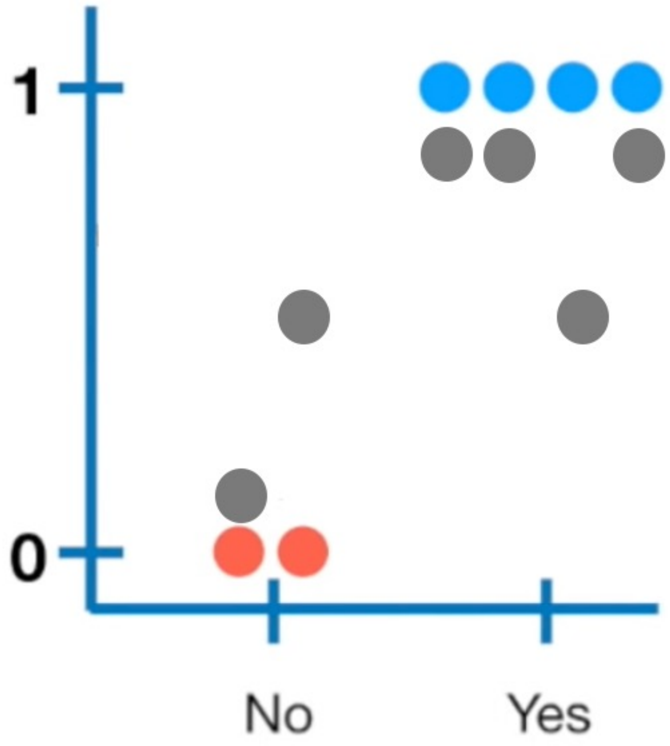
6. Calculate probabilities.

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

7. Calculate residuals.

Grsdient 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

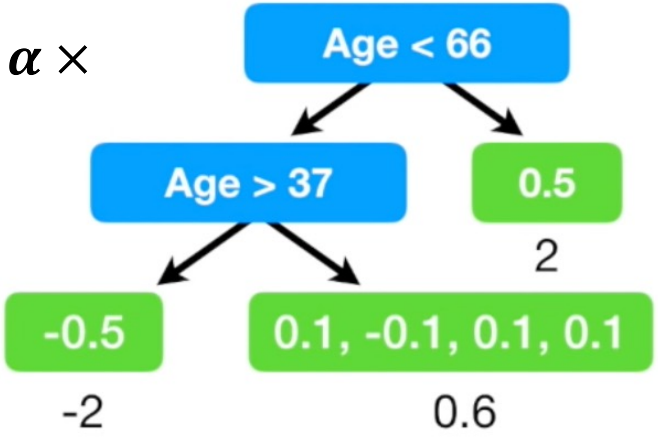


Grsdient Boost

8. Repeat.

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

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



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

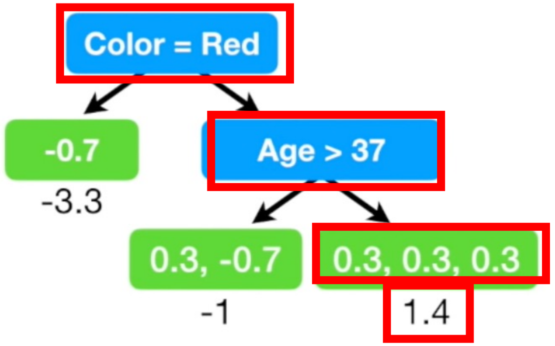
residuals

Grsdient Boost

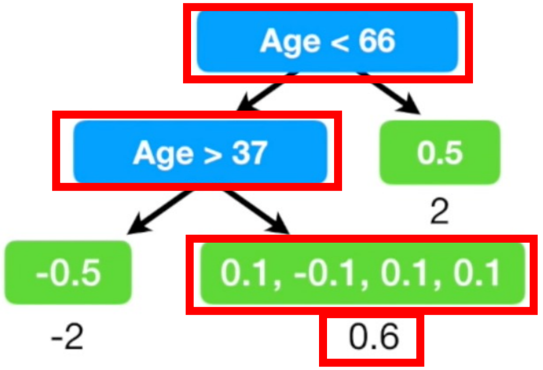
Chest Pain	Age	Color
Yes	25	Green

Predictions

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



$\boldsymbol{\alpha} \times$



pred log(odds)

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

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

MACHINE LEARNING MASTERY

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