

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>

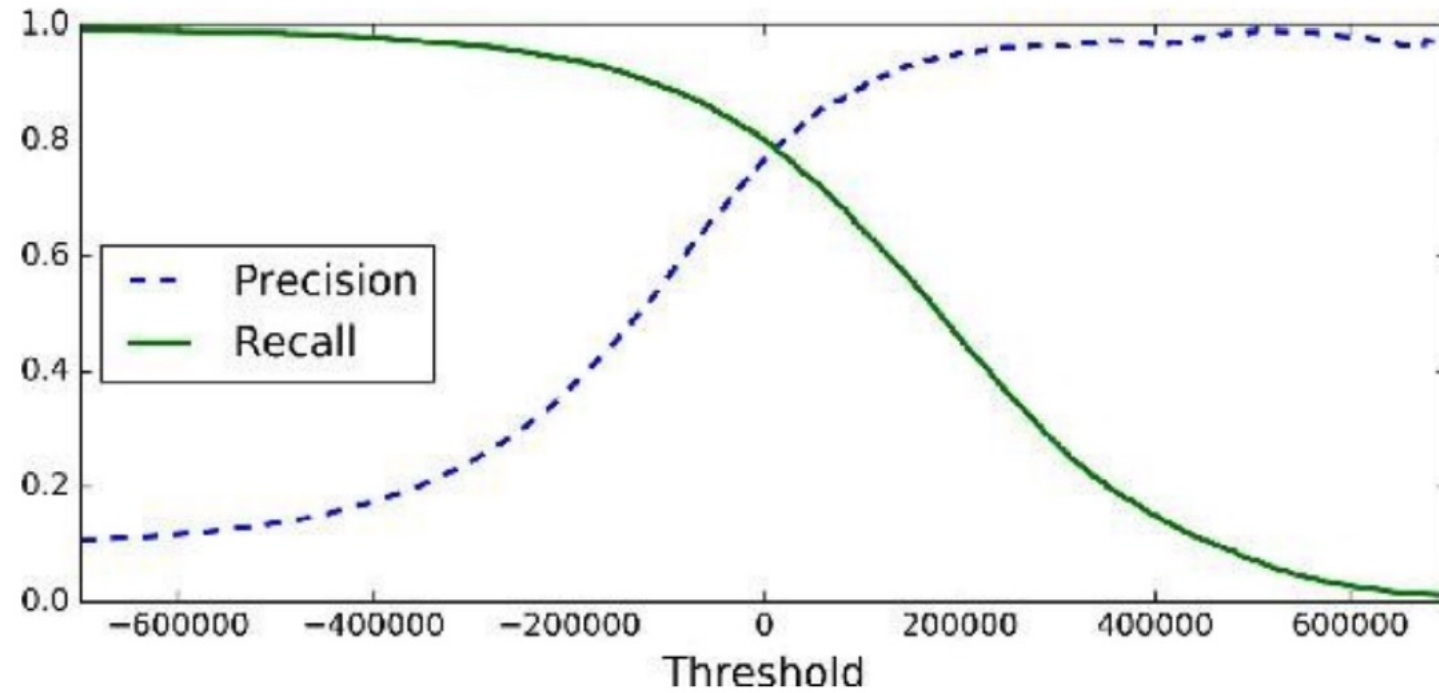
F_β -Score

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

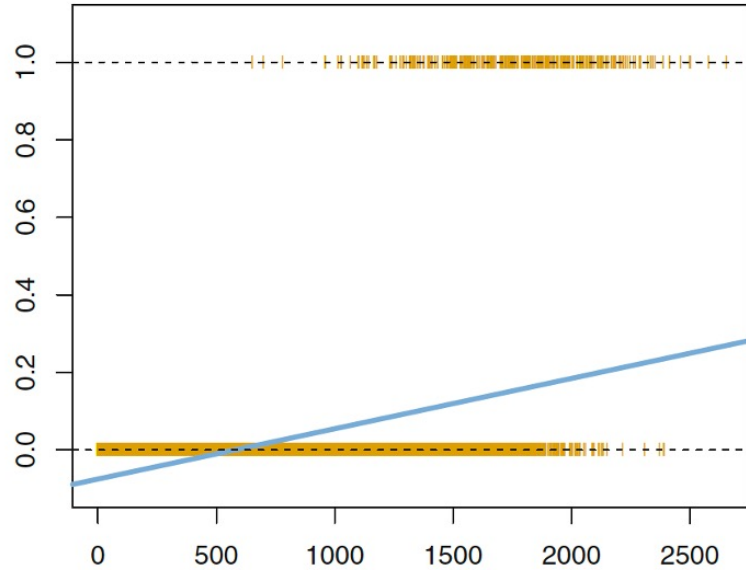
$$\beta = 1 \quad F_1 = \frac{2}{\text{recall}^{-1} + \text{precision}^{-1}} = 2 \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{\text{tp}}{\text{tp} + \frac{1}{2}(\text{fp} + \text{fn})}$$

harmonic mean

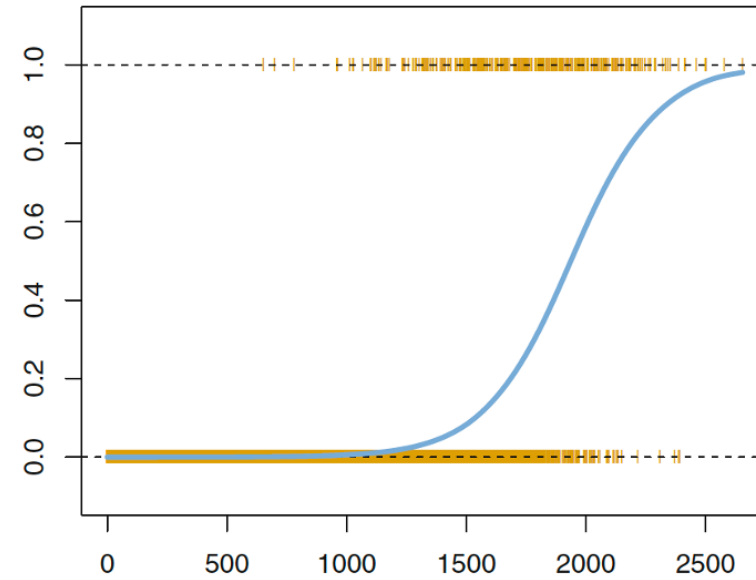
Precision – Racall Trade off



Logistic Regression



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



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

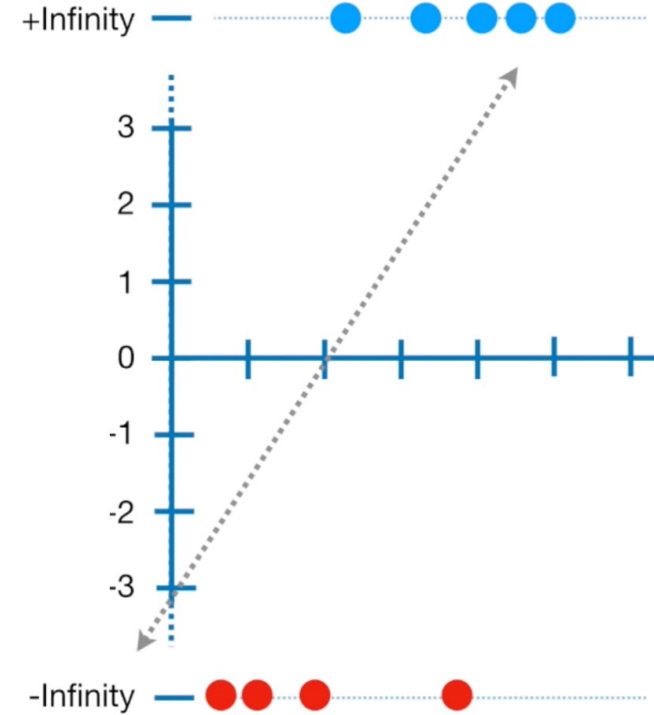
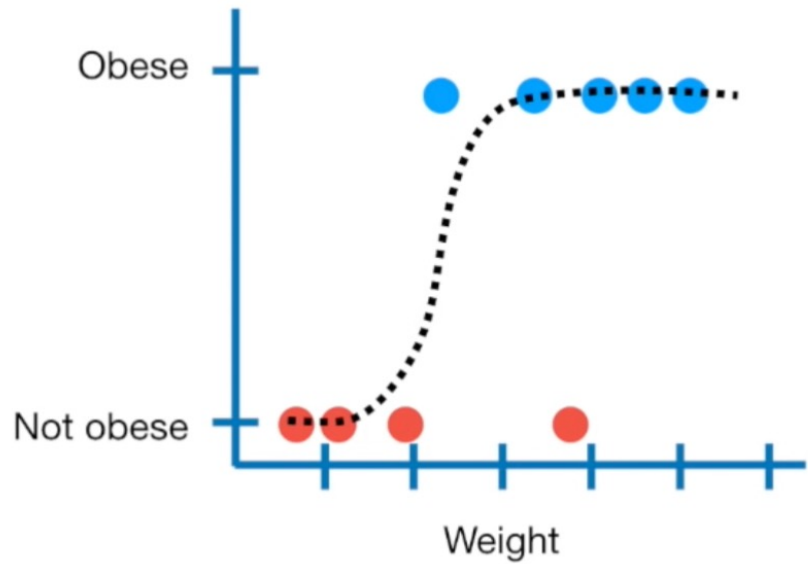
Logistic Regression

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

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

$$\log \left(\frac{p(X)}{1 - p(X)} \right) = \beta_0 + \beta_1 X \quad (\log \text{ odds})$$

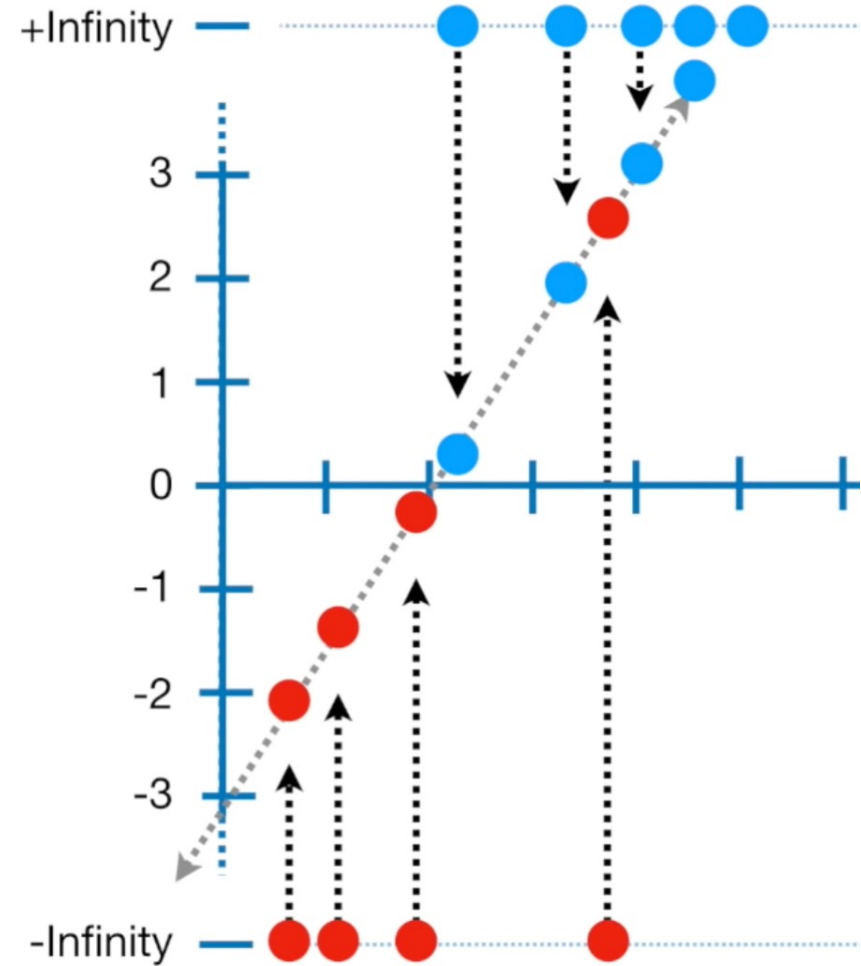
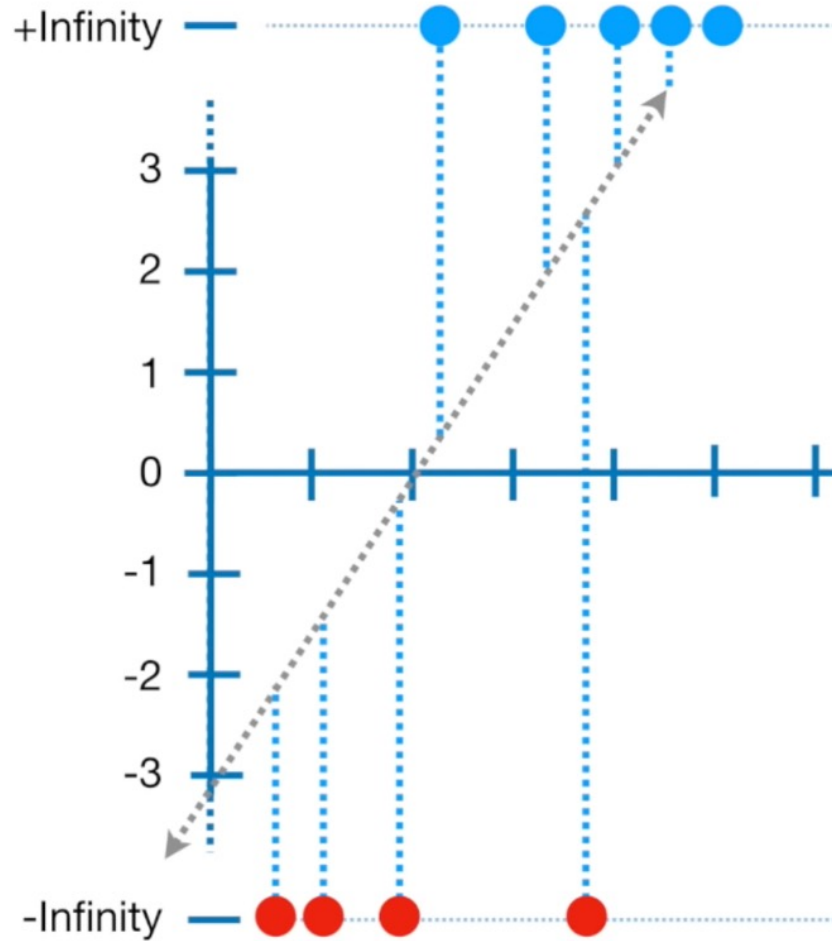
Logistic Regression



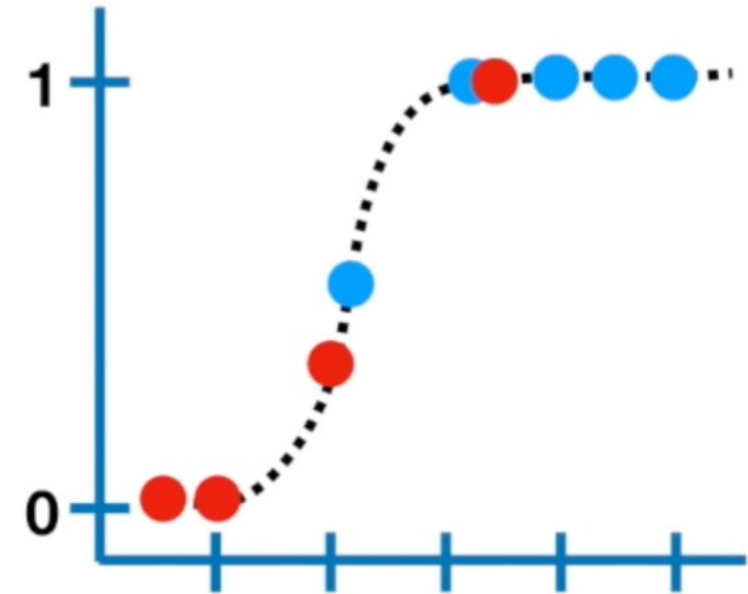
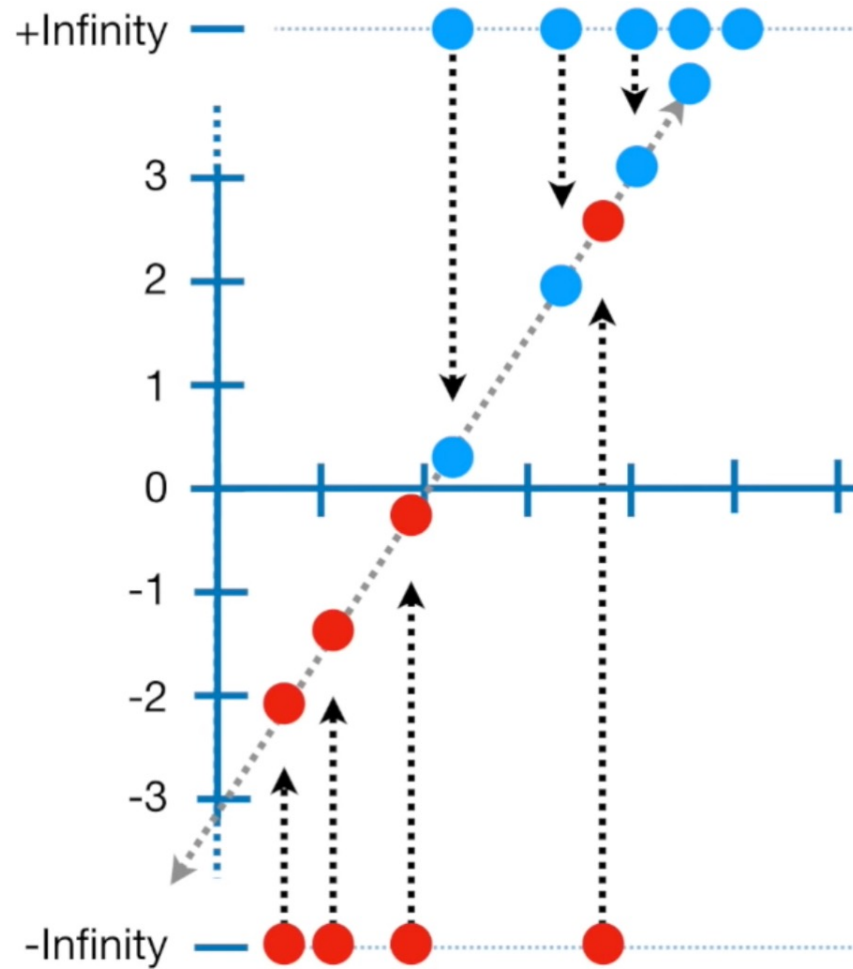
$$p(X) = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}}$$

$$\log \left(\frac{p(X)}{1 - p(X)} \right) = \beta_0 + \beta_1 X$$

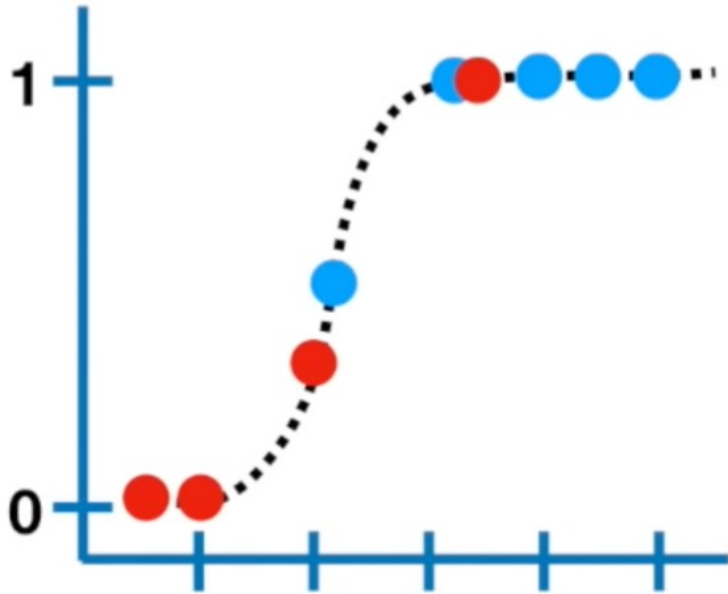
Logistic Regression



Logistic Regression



Logistic Regression



Likelihood:

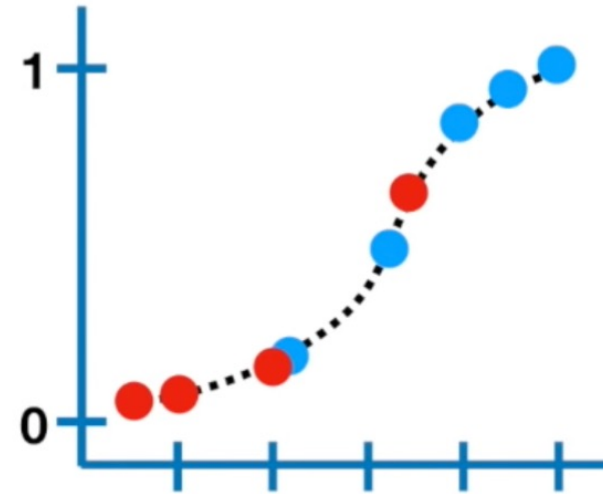
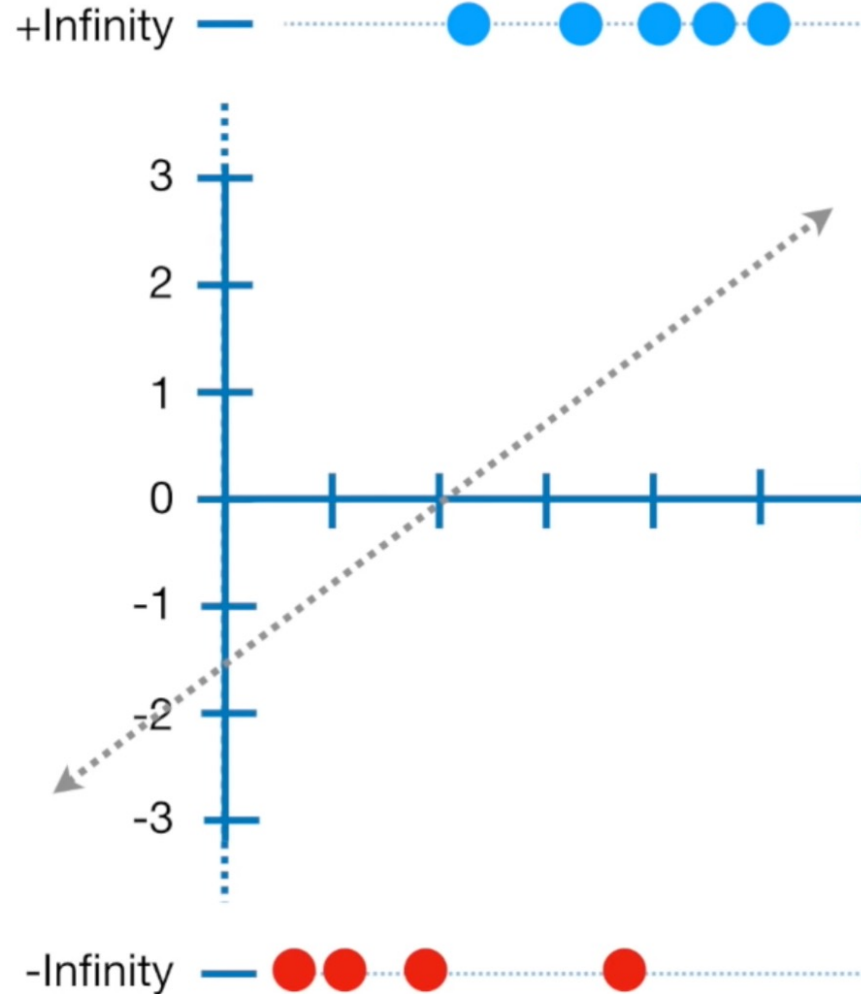
$$\ell(\beta_0, \beta_1) = \prod_{i:y_i=1} p(x_i) \prod_{i':y_{i'}=0} (1 - p(x_{i'}))$$

Log-likelihood:

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

$$\begin{aligned} &= \log(\mathbf{0.49}) + \log(\mathbf{0.9}) + \log(\mathbf{0.91}) + \log(\mathbf{0.91}) + \\ &\quad \log(\mathbf{0.92}) + \log(\mathbf{1 - 0.9}) + \log(\mathbf{1 - 0.3}) + \\ &\quad \log(\mathbf{1 - 0.01}) + \log(\mathbf{1 - 0.01}) \end{aligned}$$

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:

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

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

non – convex

Logistic Regression:

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

$$\begin{aligned} J(\theta) &= \frac{1}{m} \sum_{i=1}^m \text{Cost}(h_{\theta}(x^{(i)}), y^{(i)}) \\ &= -\frac{1}{m} \left[\sum_{i=1}^m y^{(i)} \log h_{\theta}(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_{\theta}(x^{(i)})) \right] \end{aligned}$$

cross-entropy

Workflow



CLEANSING

FILL NA

(...)



TRANSFORMATION

SCALING

NORMALISATION

ENCODING

DIM. REDUCTION

DISCRETISATION

(...)



TRAINING

REGRESSION

SUPPORT VECTOR

TREES

(...)

PIPELINE

DATA

TRAIN DATASET

TRAIN DATASET



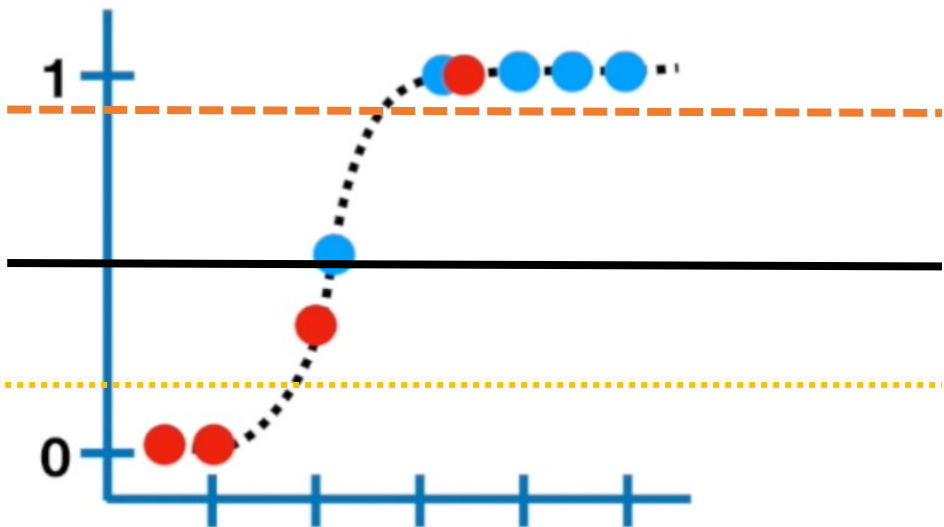
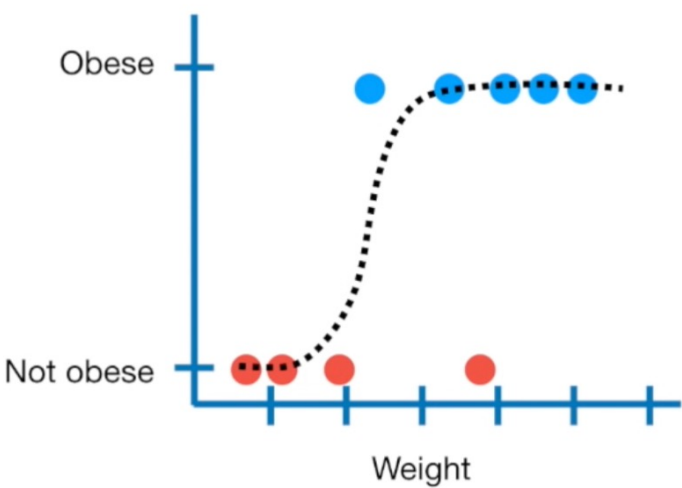
PIPELINE



MODEL PERFORMANCE



ROC-AUC

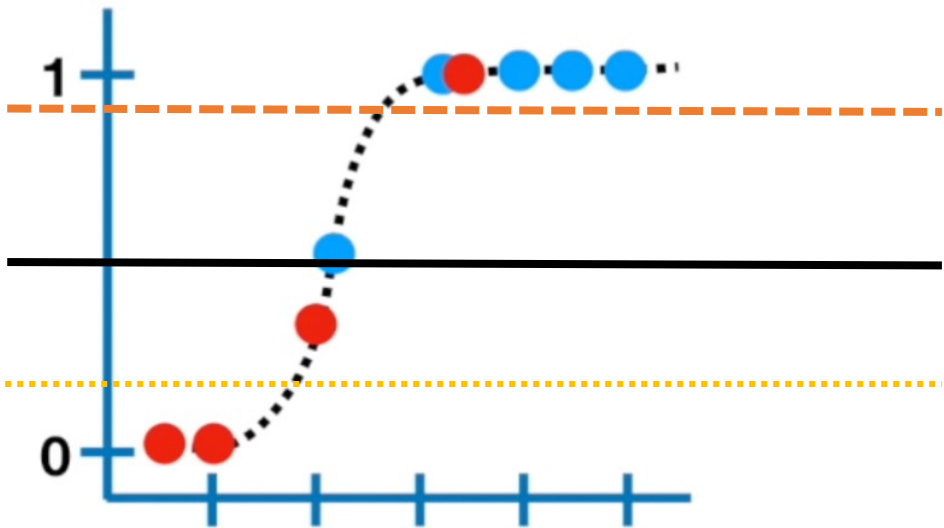
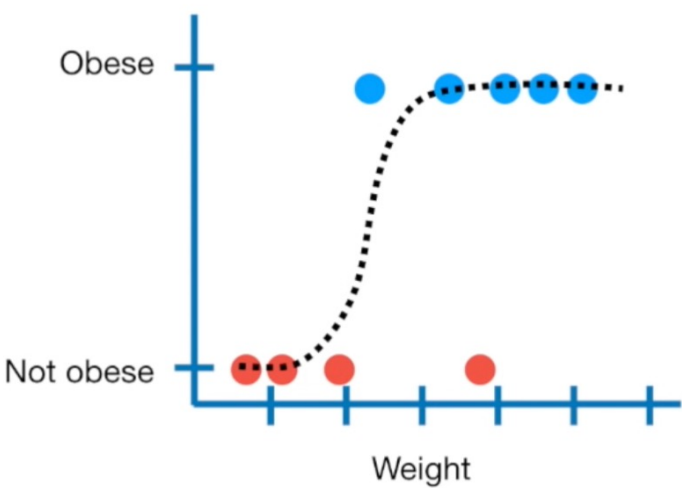


PREDICTIONS	ACTUALS	
	1	0
1		
0		

PREDICTIONS	ACTUALS	
	1	0
1		
0		

PREDICTIONS	ACTUALS	
	1	0
1		
0		

ROC-AUC

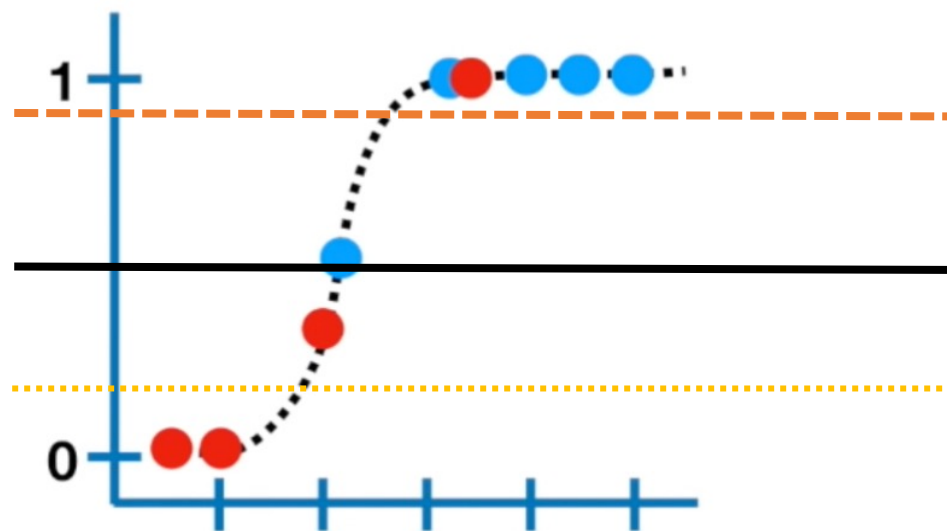


	ACTUALS	
	1	0
PREDICTIONS 1	4	1
PREDICTIONS 0	1	3

	ACTUALS	
	1	0
PREDICTIONS 1	5	0
PREDICTIONS 0	1	3

	ACTUALS	
	1	0
PREDICTIONS 1	5	0
PREDICTIONS 0	2	2

ROC-AUC



	ACTUALS	
	1	0
PREDICTIONS 1	4	1
PREDICTIONS 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	0
PREDICTIONS 1	5	0
PREDICTIONS 0	1	3

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

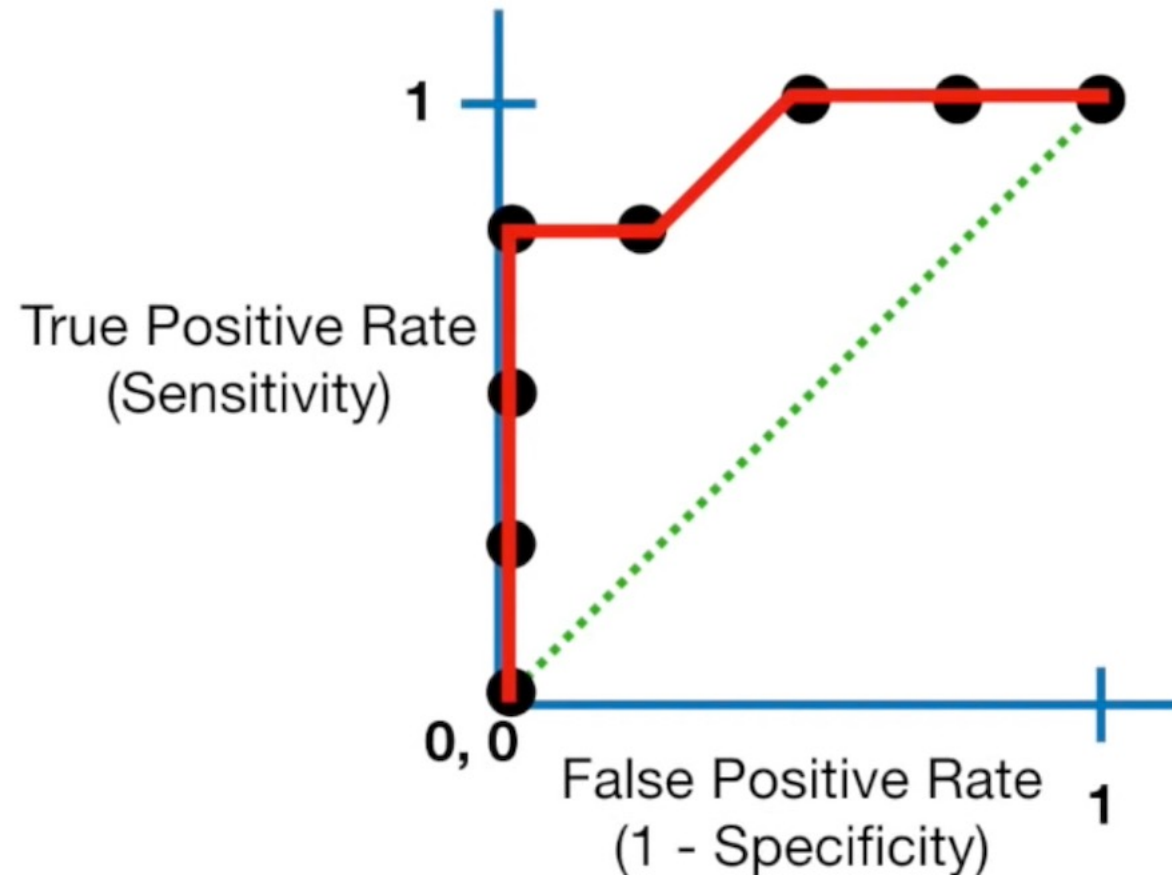
$$FPR = \frac{0}{0 + 3}$$

	ACTUALS	
	1	0
PREDICTIONS 1	5	0
PREDICTIONS 0	2	2

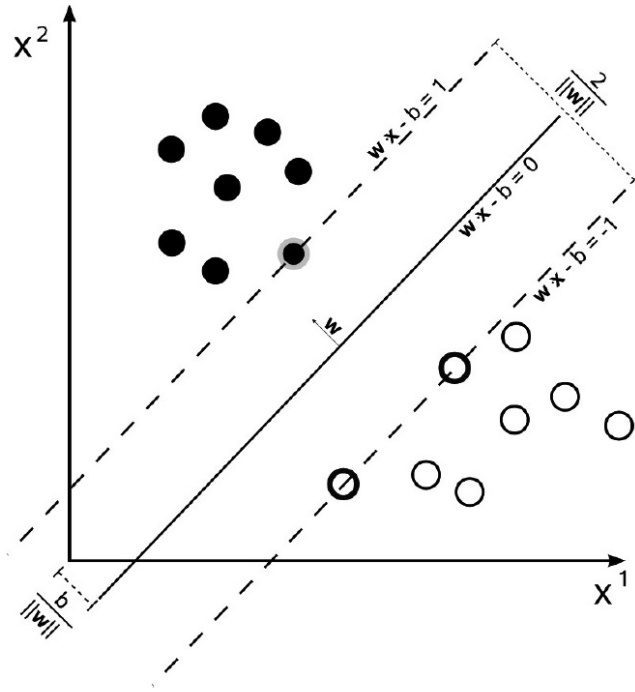
$$TPR = \frac{5}{5 + 2}$$

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

ROC-AUC



SVM – Hard Margin



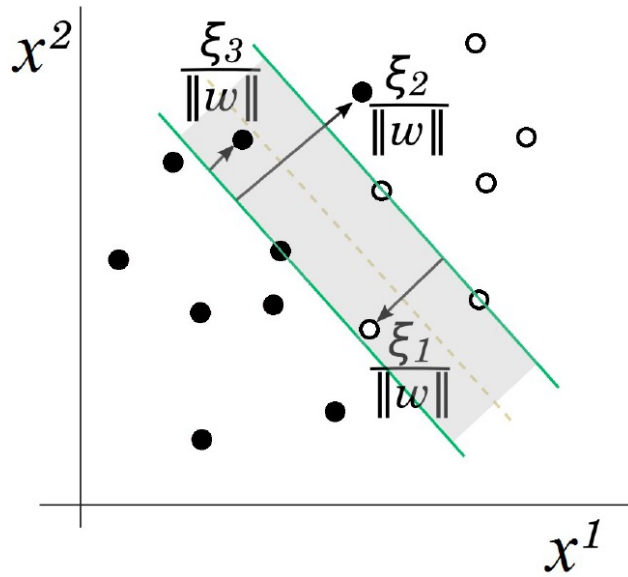
$$f(x) = w^T x - b$$

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

Constraints:

$$\begin{aligned} w^T x_i - b &\geq 1, & x_i &\in \text{Class} \\ w^T x_i - b &\leq -1, & x_i &\notin \text{Class} \end{aligned} \quad \longrightarrow \quad (w^T x_i - b)y_i \geq 1$$

SVM – Soft Margin



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

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

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

$$\lambda > 0$$

Constraints, for each i :

$$\xi_i \geq 0$$

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

Naïve Bayes

Likelihoods

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

$$\begin{aligned} p(\text{Dear} | N) &= 0.47 \\ p(\text{Friend} | N) &= 0.29 \\ p(\text{Lunch} | N) &= 0.18 \\ p(\text{Money} | N) &= 0.06 \end{aligned}$$

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

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

$$\begin{aligned} p(\text{Dear} | S) &= 0.29 \\ p(\text{Friend} | S) &= 0.14 \\ p(\text{Lunch} | S) &= 0.00 \\ p(\text{Money} | S) &= 0.57 \end{aligned}$$

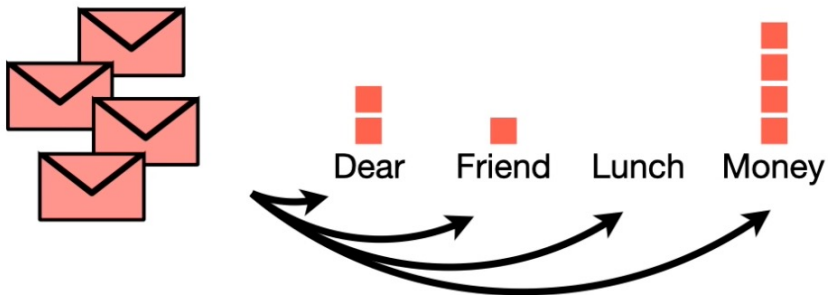
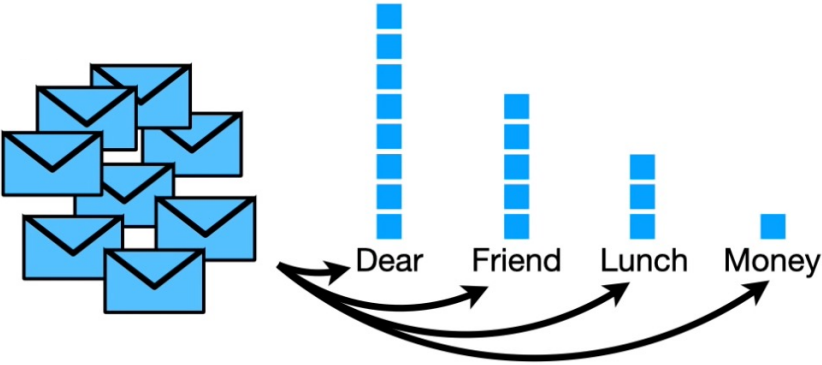
Prior probability $p(S) = \frac{\#S}{\# \text{total emails}}$

Dear Friend



$$p(N) \times p(\text{Dear} | N) \times p(\text{Friend} | N)$$

$$p(S) \times p(\text{Dear} | S) \times p(\text{Friend} | S)$$



Naïve Bayes - Gaussian

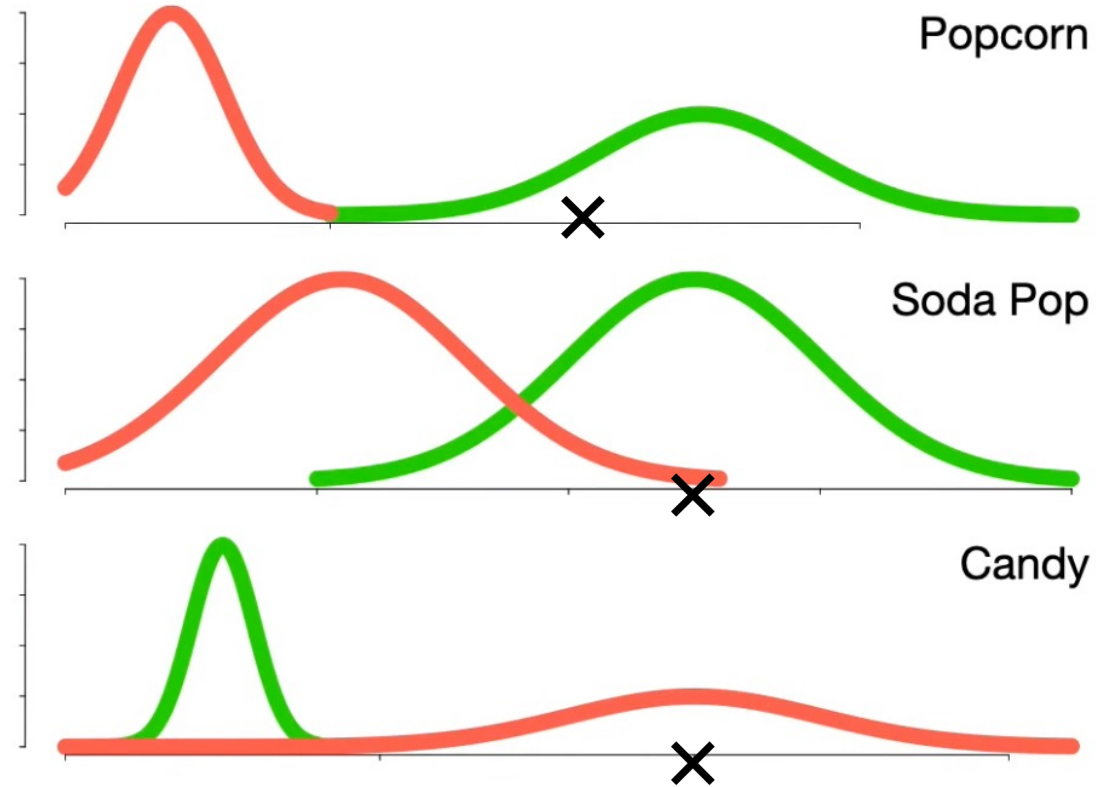
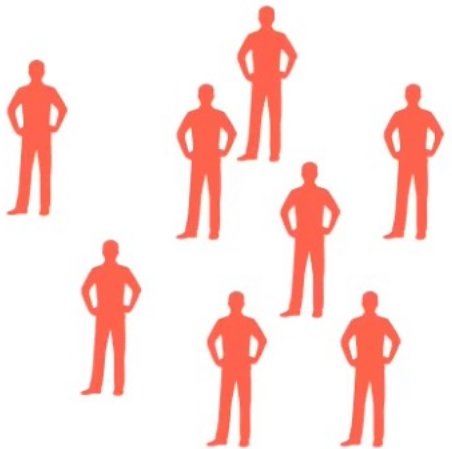


$$p(G) = \frac{\#G}{\#total}$$

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.



$$p(G)$$

$$\times L(\text{popcorn} = 20|G)$$

$$\times L(\text{soda pop}|G)$$

$$\times L(\text{candy} = 25|G)$$

$$p(R)$$

$$\times L(\text{popcorn} = 20|R)$$

$$\times L(\text{soda pop}|R)$$

$$\times L(\text{candy} = 25|R)$$



Decision Tree - CART

Gini:

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

Log Loss or Entropy:

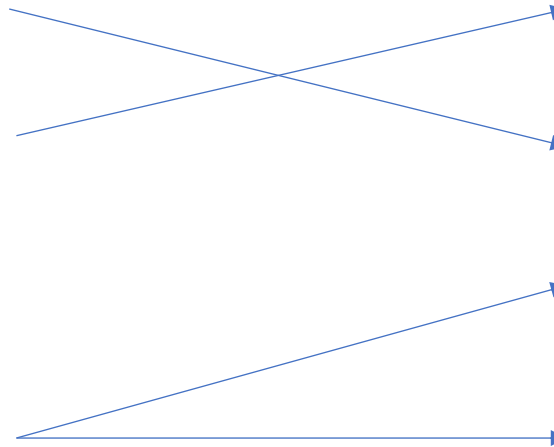
$$H(Q_m) = - \sum_k p_{mk} \log(p_{mk})$$

Random Forrest

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 Forrest

Step 2: Create Decision Tree .

Randomly select subset of features.



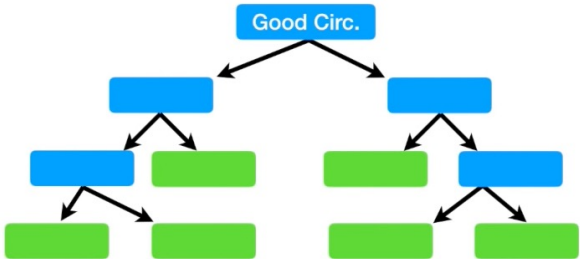
Find best split.



Randomly select subset of features to node split.



Create Tree considering subset of features.

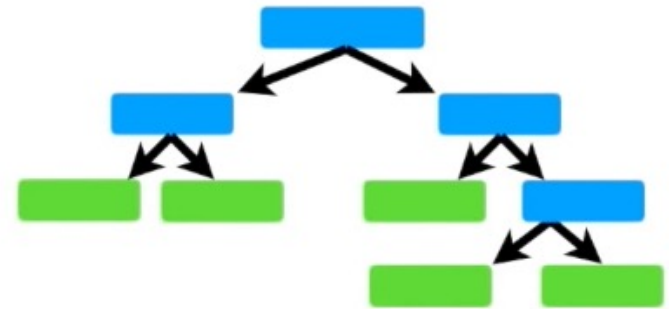
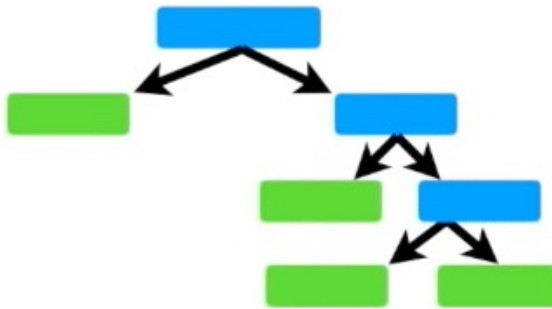
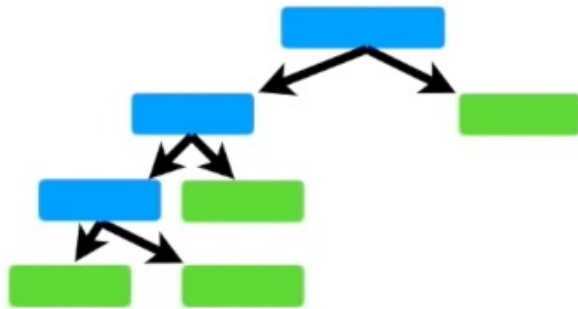
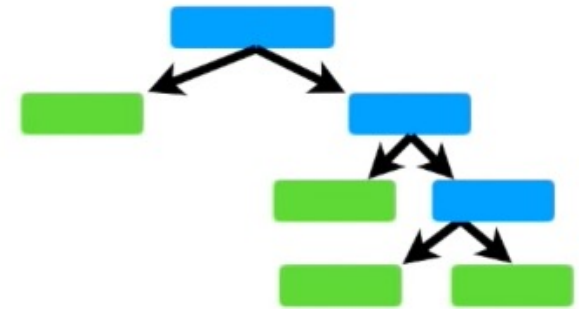
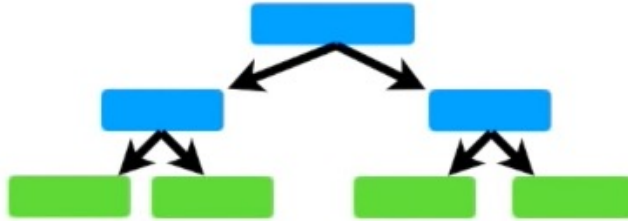
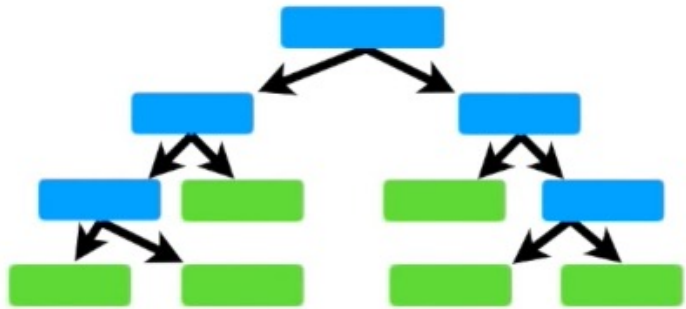


Bootstrapped Dataset

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

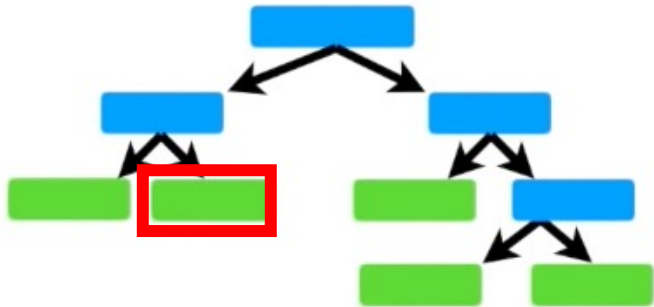
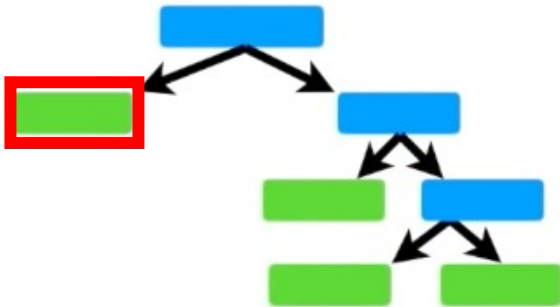
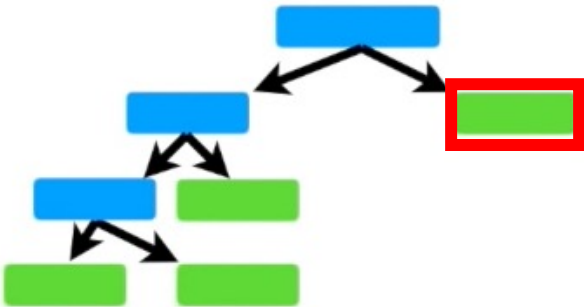
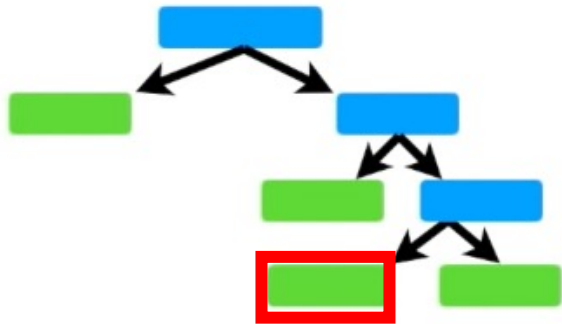
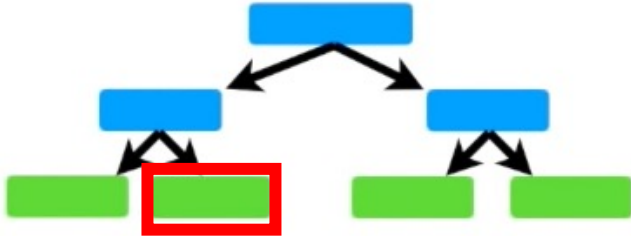
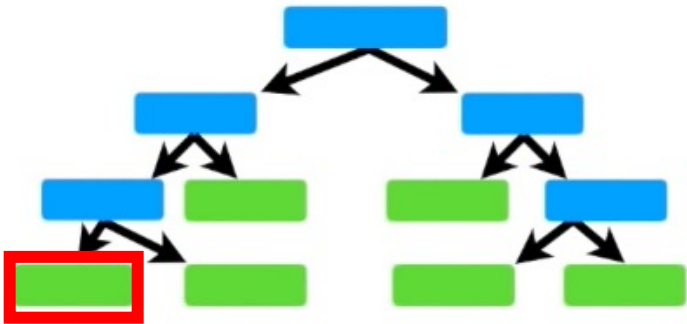
Repeat Step 1 & Step 2 creating next trees.



Random Forrest

Predictions.

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



Random Forrest

Out-Of-Bag Dataset

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_*** attribute 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.

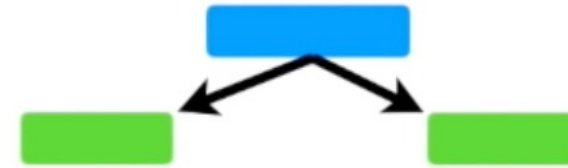
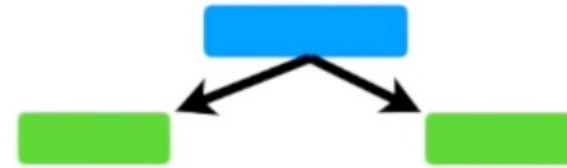
AdaBoost

Original Dataset

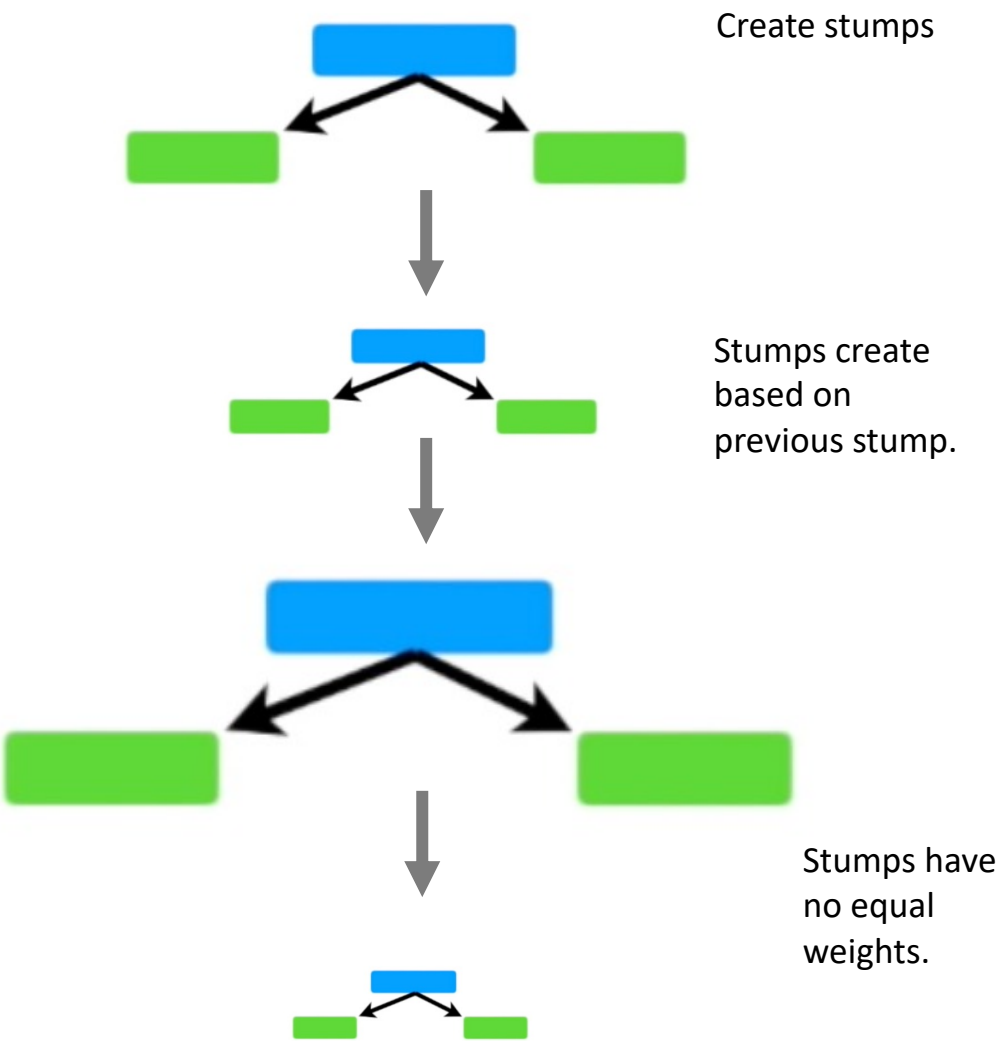
Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	Yes	167	Yes



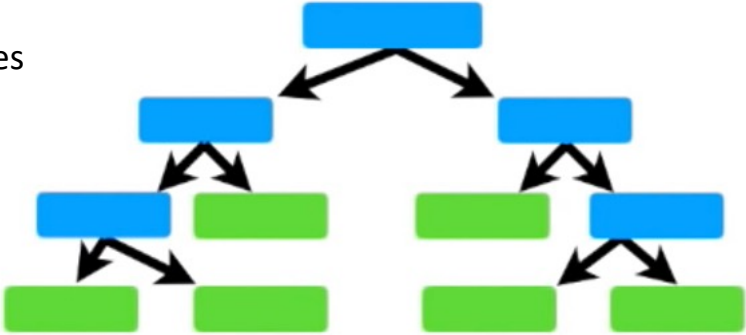
stump



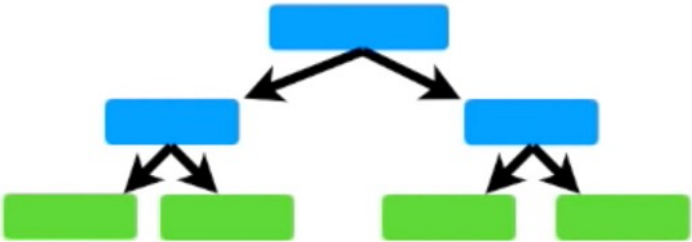
AdaBoost



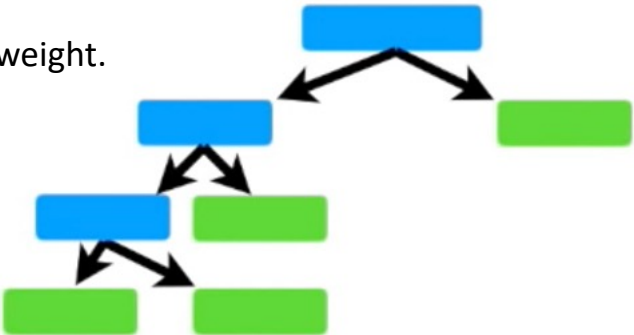
Create full trees



Independent trees



Each tree has equal weight.



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 minimizing Gini



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

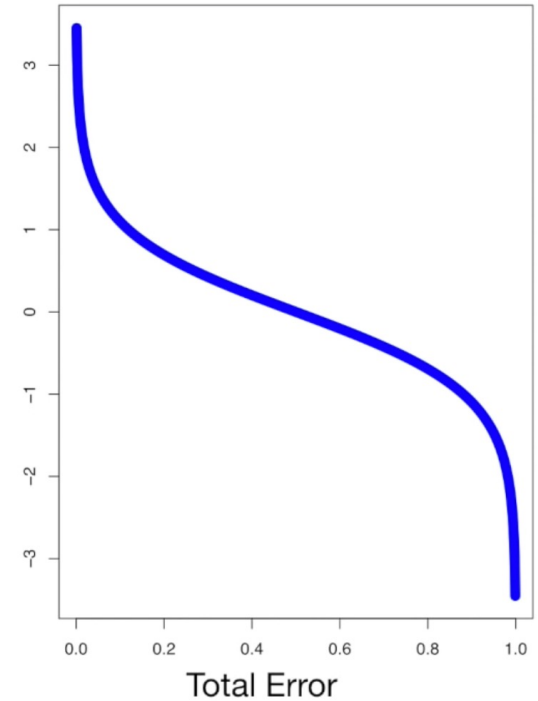
Step 2. Update sample weights.

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

total_error - sum of incorrect classified samples weights

New weight = *weight* × e^I - True

New weight = *weight* × 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.

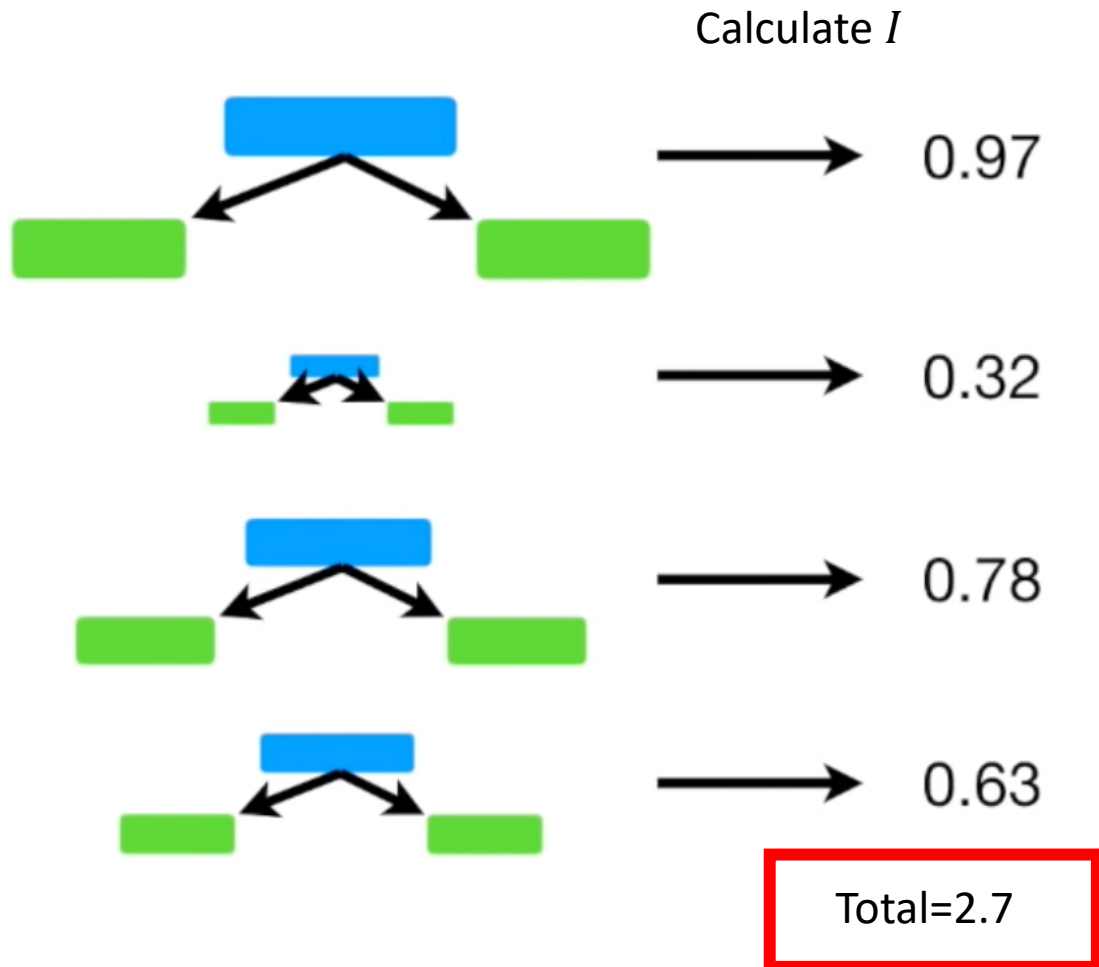
Step 3. Normalize sample weights.

Step 4. Bootstrap dataset using new sample weights.

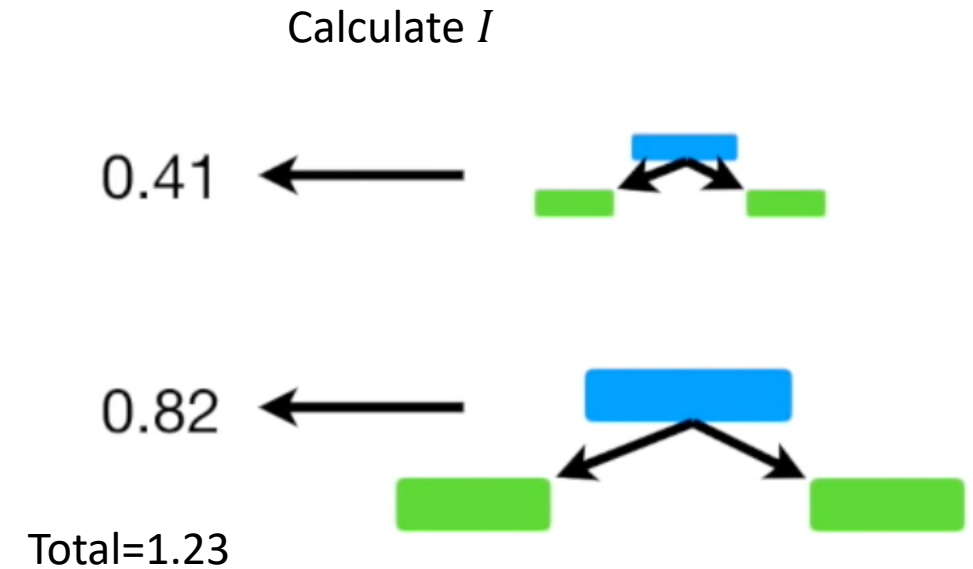
Step 5. Repeat using bootstrap dataset.

AdaBoost

$$I = \frac{1}{2} \log\left(\frac{1 - \text{total_error}}{\text{total_error}}\right)$$



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



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: $F_0(x) = \operatorname{argmin}_{\gamma} \sum_{i=1}^n L(y_i, \gamma)$

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

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 $i = 1, \dots, n$

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

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

(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 $F_M(x)$

Gradient Boost

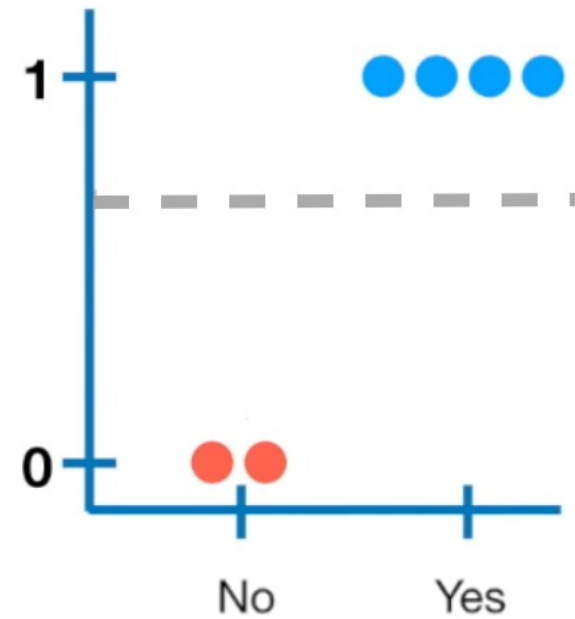
Chest Pain	Age	Color	Heart Disease	Residual
Yes	12	Blue	Yes	0.3
Yes	87	Green	Yes	0.3
No	44	Blue	No	-0.7
Yes	19	Red	No	-0.7
No	32	Green	Yes	0.3
No	14	Blue	Yes	0.3

1. Initialise predication value

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

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

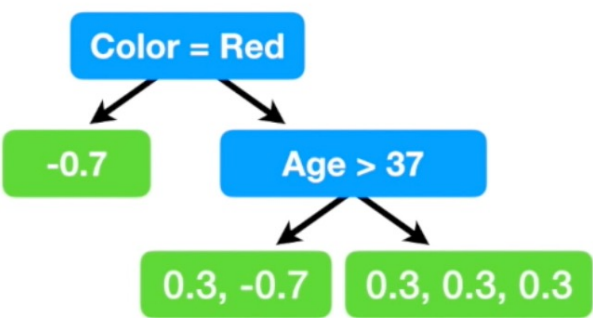
2. Calculate (pseudo) residuals



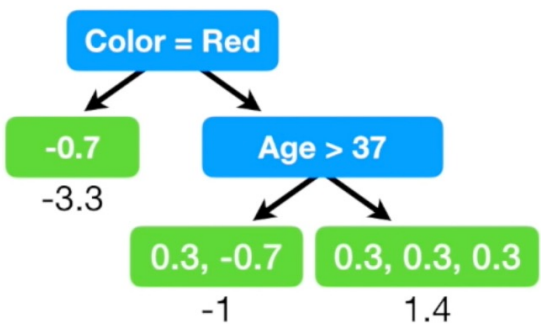
Grsdient Boost

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



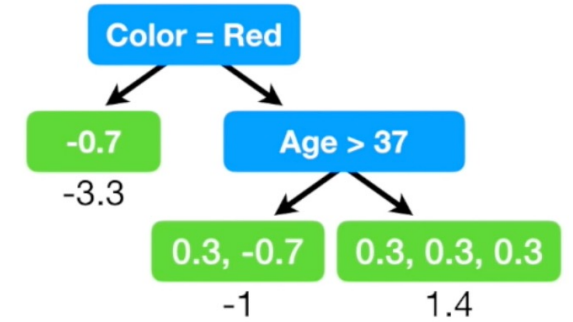
Gradient Boost

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.

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

α – learning rate



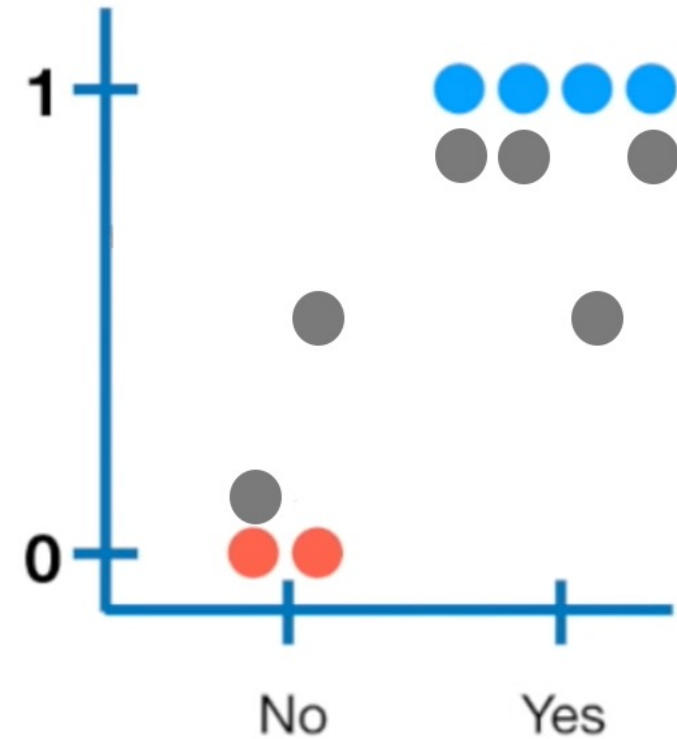
6. Calculate probabilities.

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

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

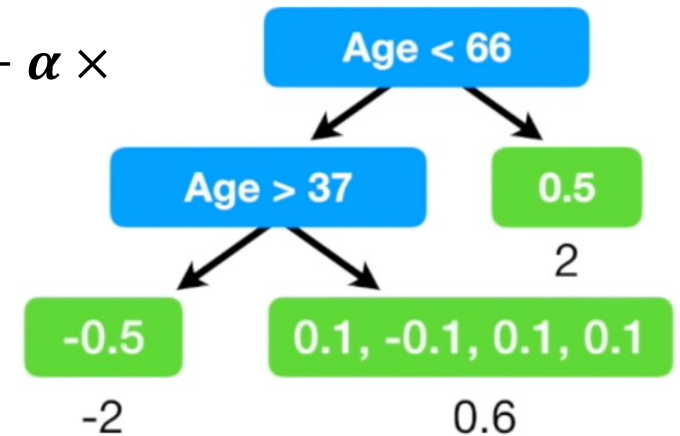


8. Repeat.

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

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



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

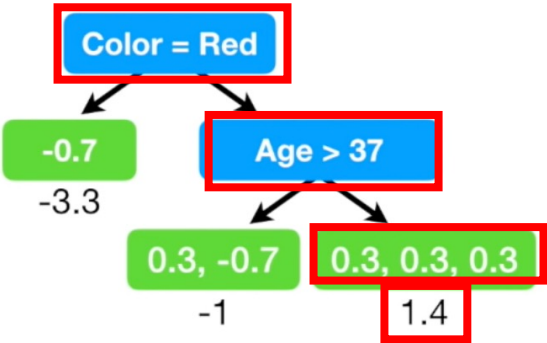
residuals

Grsdient Boost

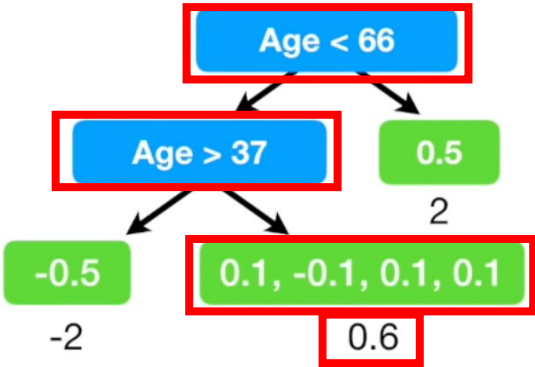
Chest Pain	Age	Color
Yes	25	Green

Predictions

initial log(odds) + $\alpha \times$



$\alpha \times$



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

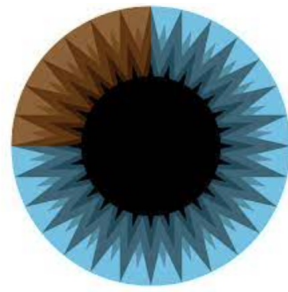
Regularisation:

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

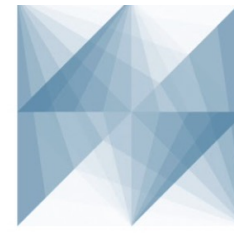
Sklearn Pipelines



<https://machinelearningmastery.com>



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

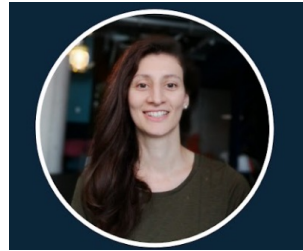


Machine Learning Study Groups

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

towards data science

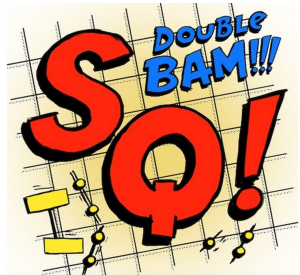
<https://towardsdatascience.com>



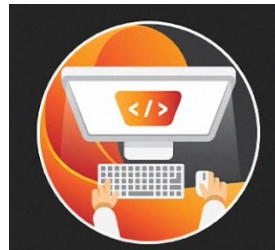
<https://www.youtube.com/c/TechWorldwithNana>



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



<https://www.youtube.com/c/joshstarmar>



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

kaggle

