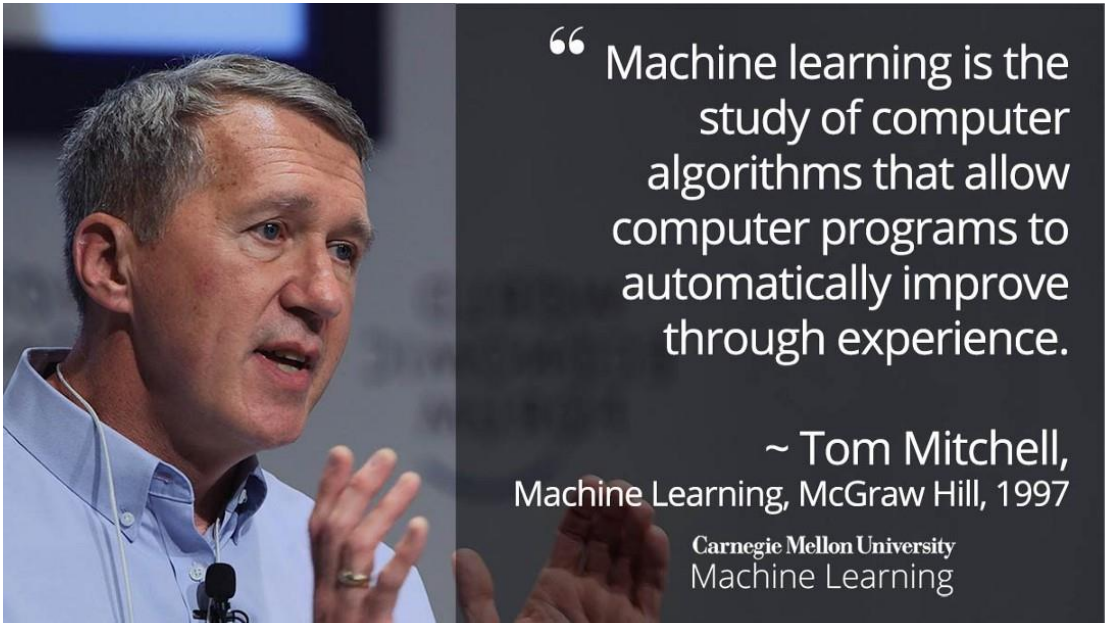
Decision Tree, Random Forest, & XGBoost

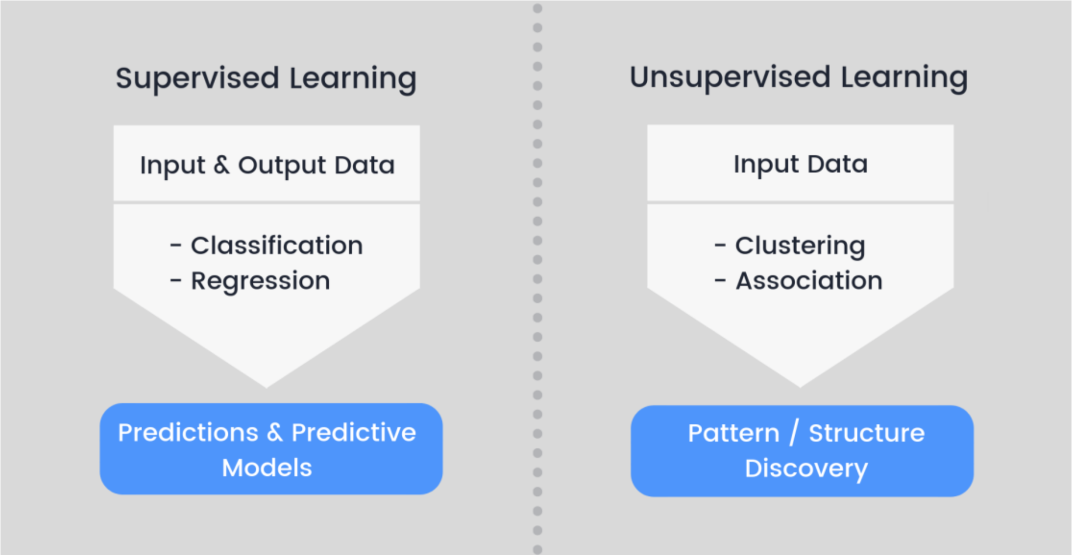
Xia Song

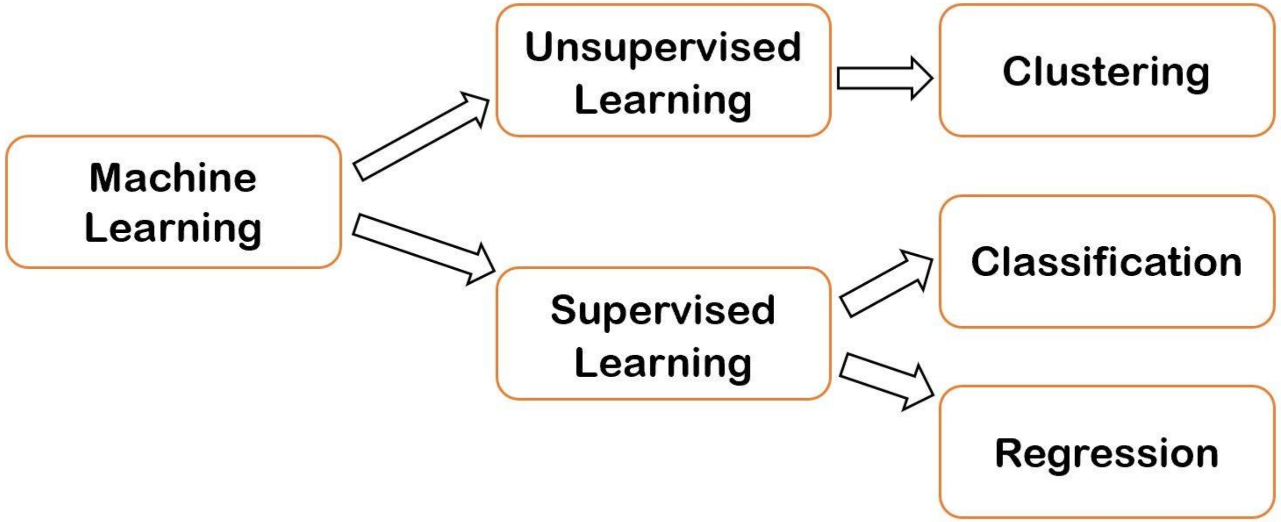
Outline

- **Machine Learning in General**
- **Decision Tree**
- **Ensemble Methods**
  - **Bagging: Random Forest**
  - **Boosting: XGBoost**

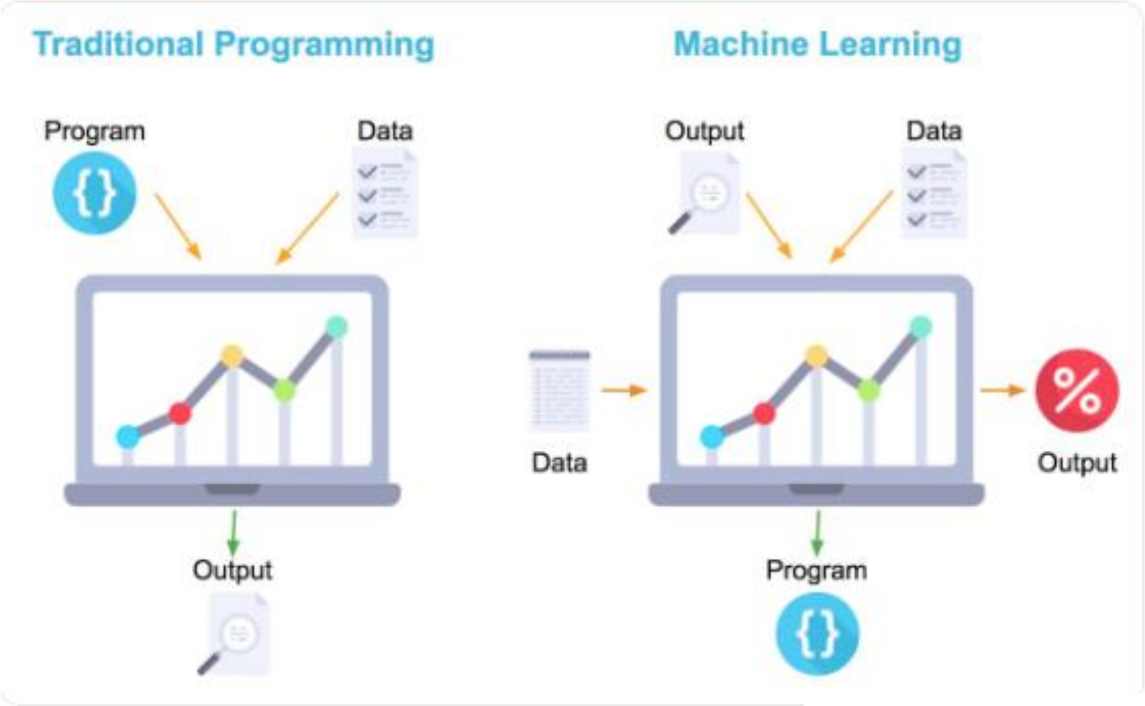
Machine Learning



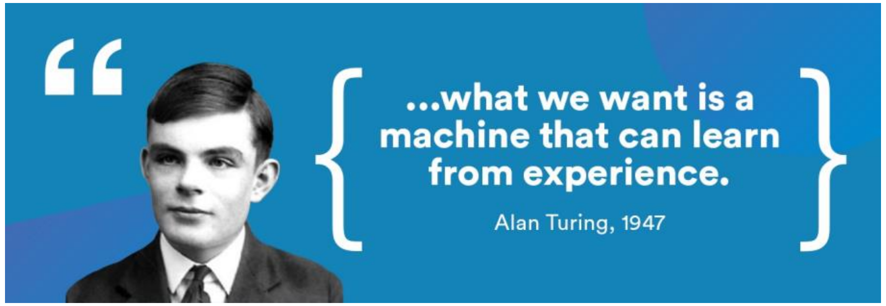


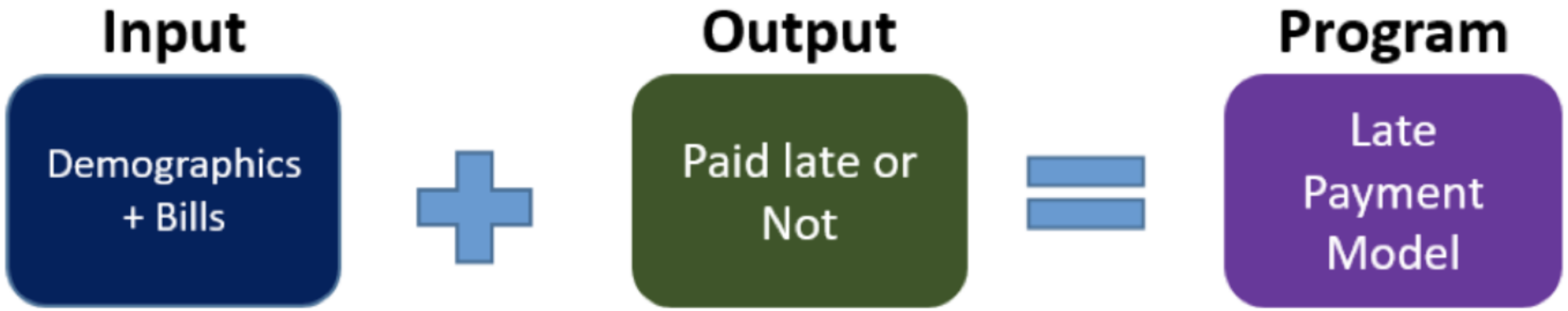


Machine Learning vs Traditional Programming



- Identify the business question you would like to ask.
- Identify the historical input.
- Identify the historically observed output (i.e., data samples for when the condition is true and for when it's false).



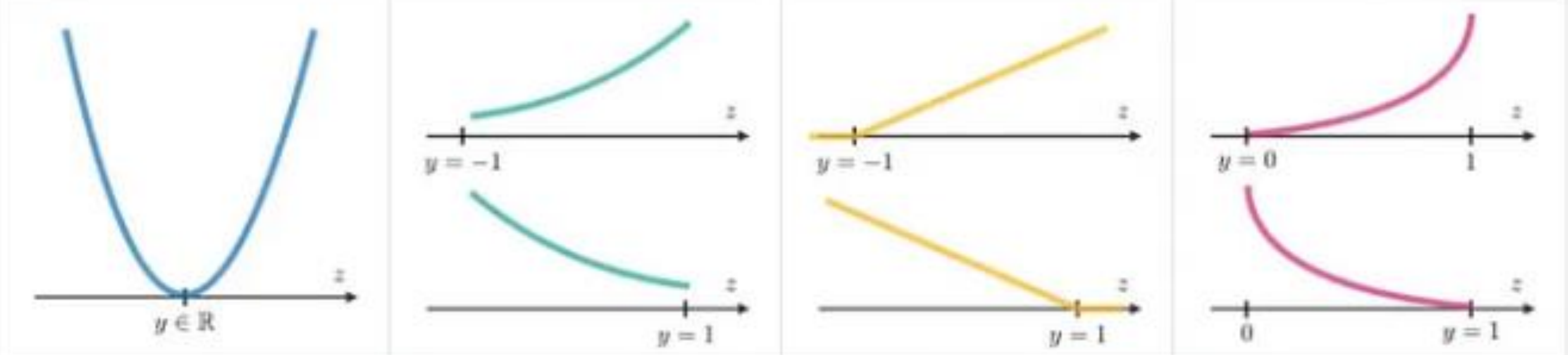


Logi Analytics 2020

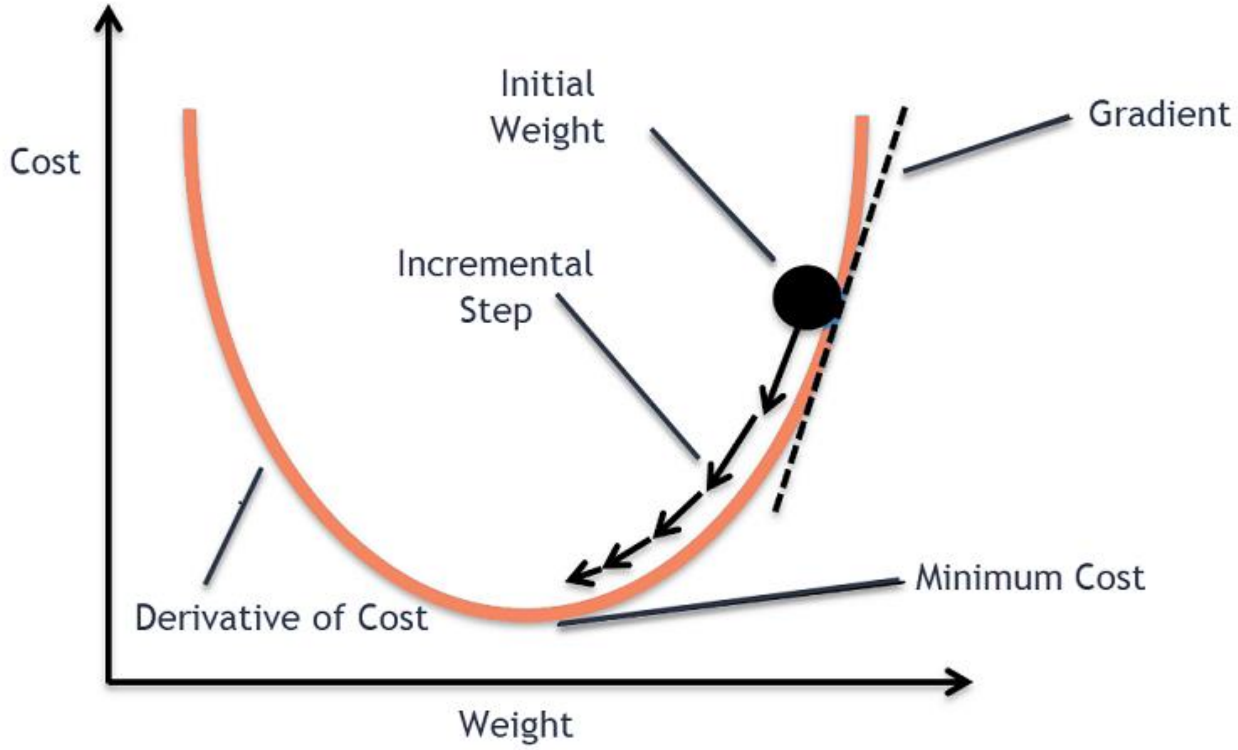
How does Machine Learning: Cost Function

$$f(m,b) = \frac{1}{N}\sum_{i=1}^{N}(y_i - (mx_i + b))^2$$

Least squared error	Logistic loss	Hinge loss	Cross-entropy
$\frac{1}{2}(y-z)^2$	$\log(1+\exp(-yz))$	$\max(0, 1-yz)$	$-\left[y\log(z)+(1-y)\log(1-z)\right]$
Linear regression	Logistic regression	SVM	Neural Network



How does Machine Learning: Gradient Descent



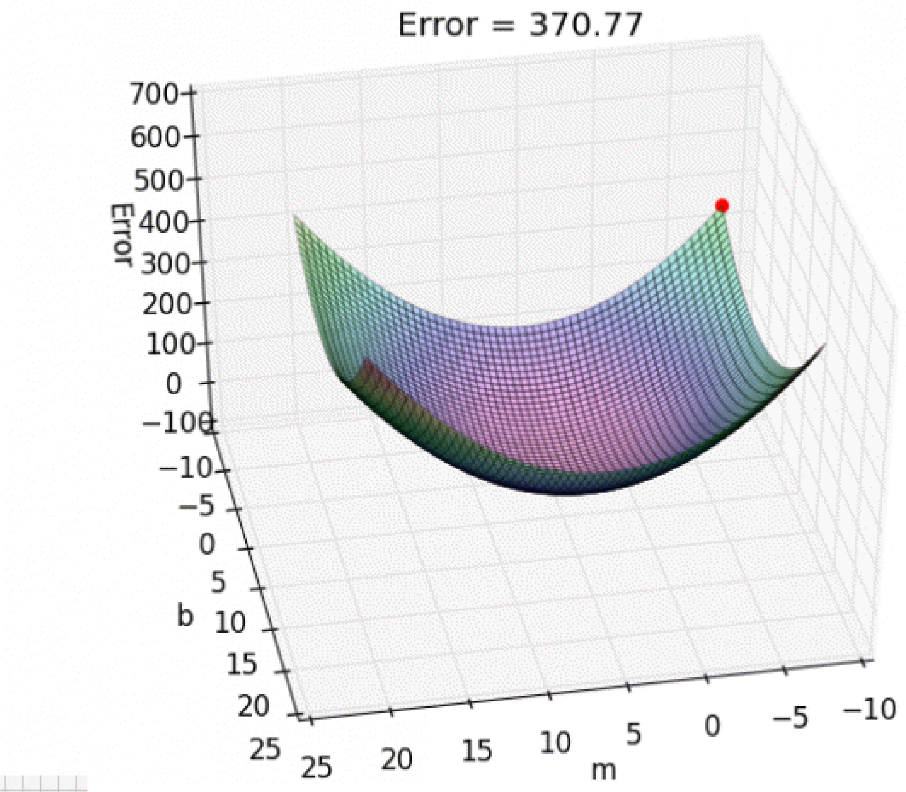
$$f'(m,b) = \begin{bmatrix} \frac{df}{dm} \\ \frac{df}{db} \end{bmatrix} = \begin{bmatrix} \frac{1}{N} \sum -2x_i(y_i - (mx_i + b)) \\ \frac{1}{N} \sum -2(y_i - (mx_i + b)) \end{bmatrix}$$

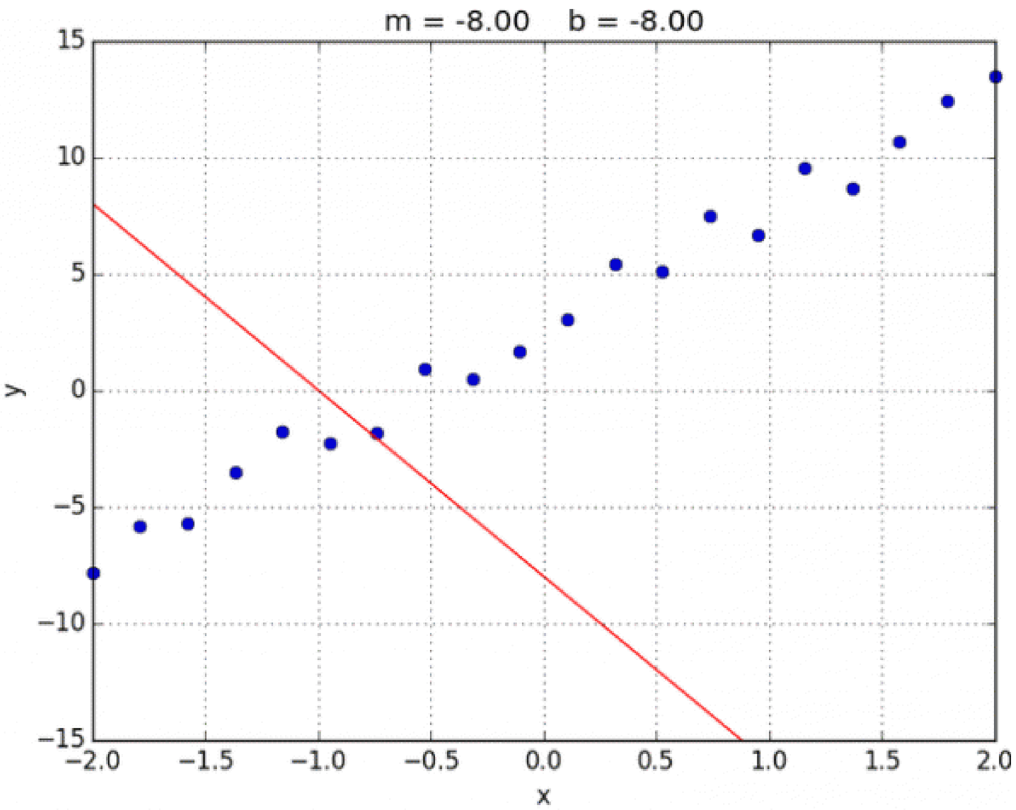
θ = θ - α * ∇J(θ)

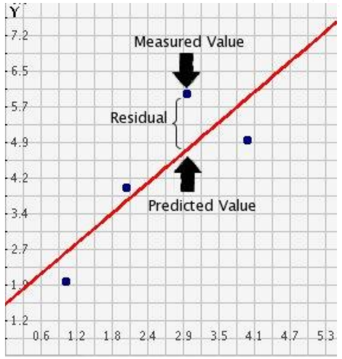
where

- **α** is the learning rate
- **∇J(θ)** is the gradient of the cost function with respect to θ.

How does Machine Learning: Gradient Descent







$$h_\theta(x) = mx + b$$

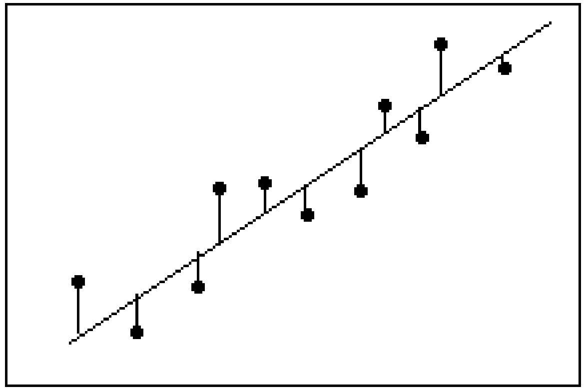
$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

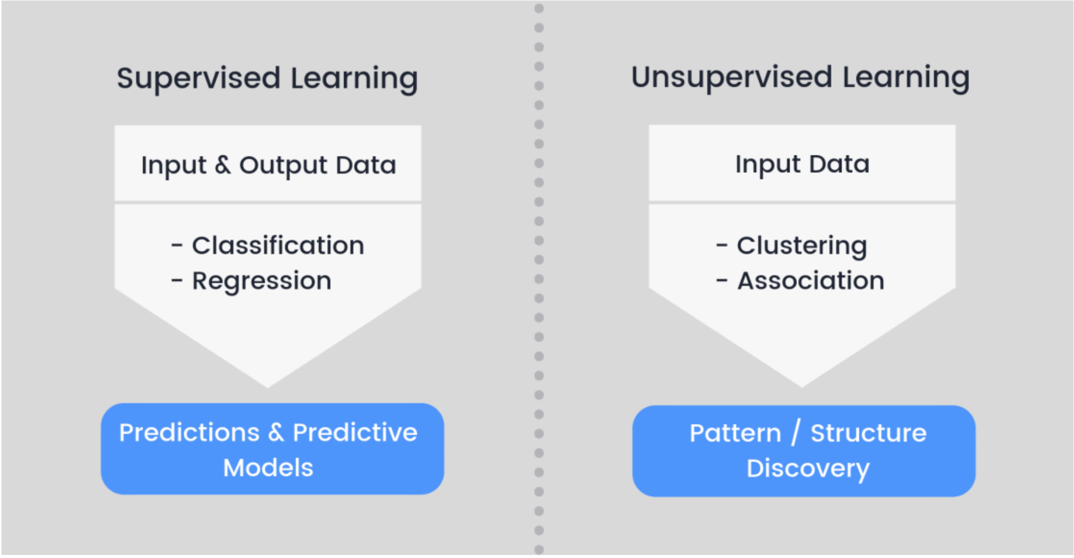
http://alykhantejani.github.io

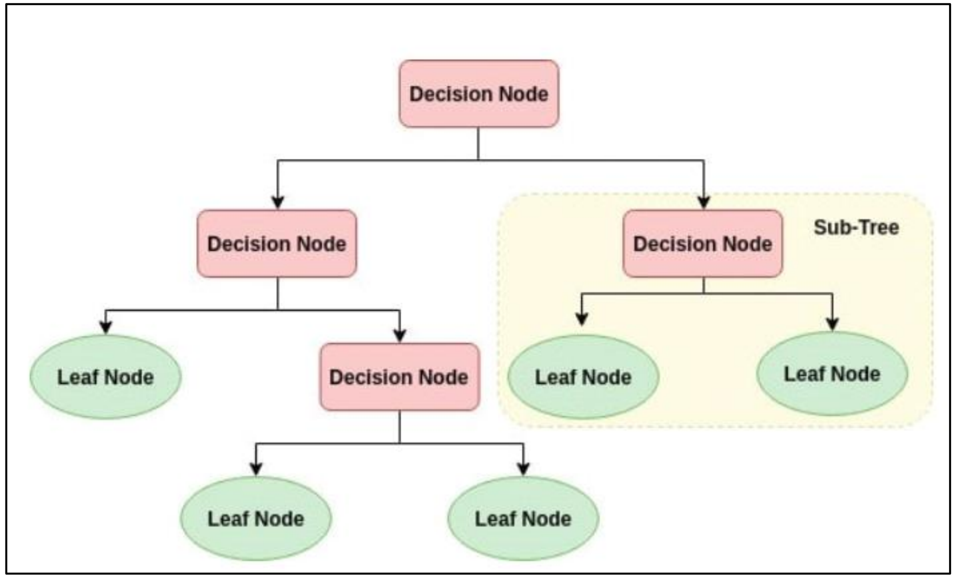
Machine Learning Algorithms

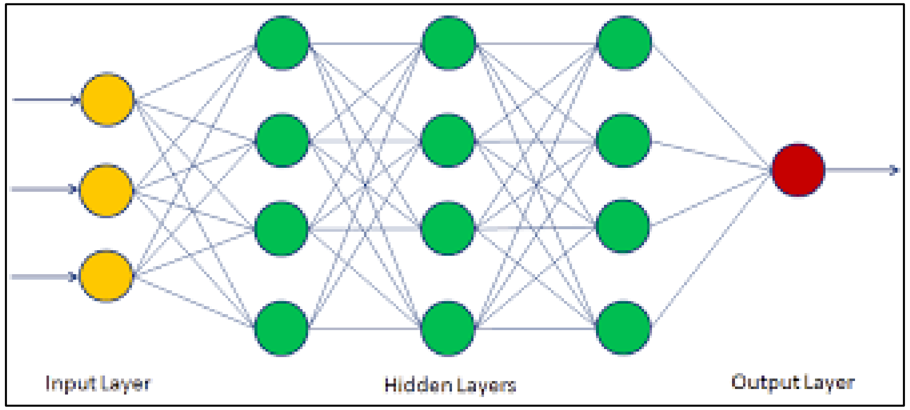
Three major classes of Supervised Learning algorithms

- **Linear Models**
- **Tree Learning Models**
- **Neural Network Models**





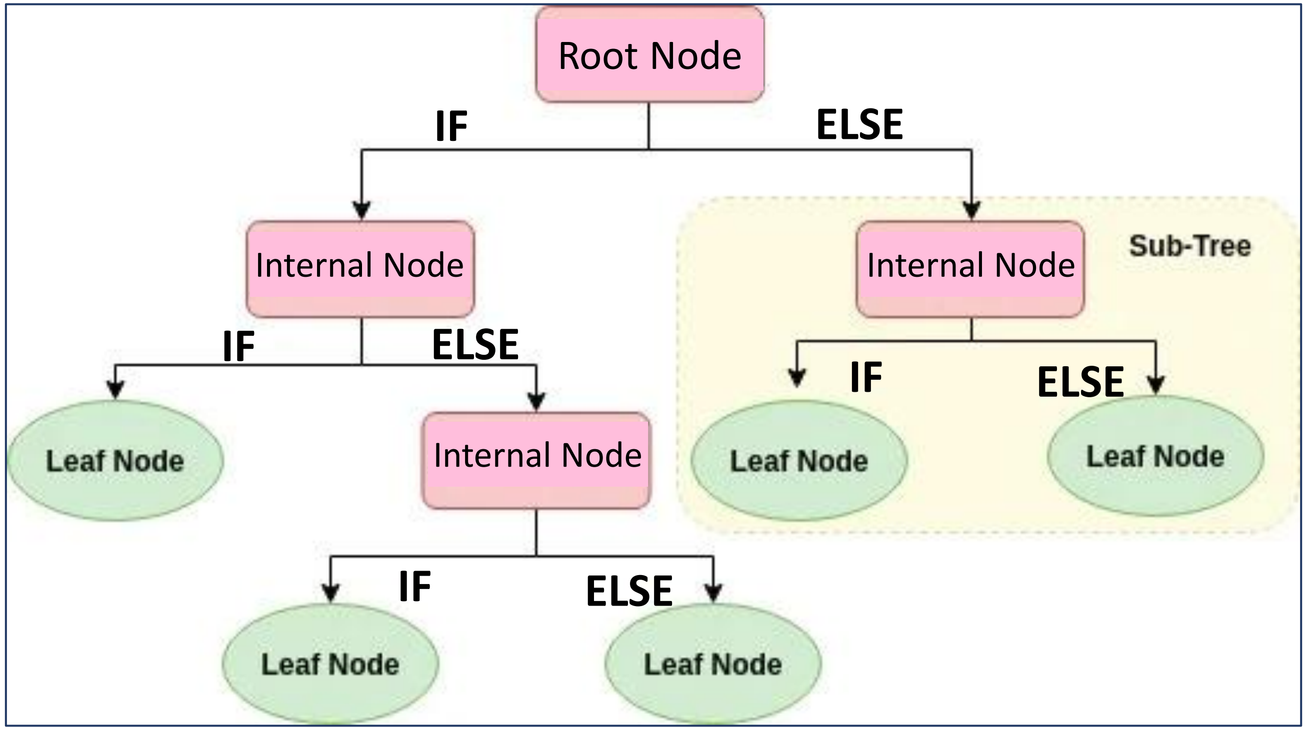




Outline

- **Machine Learning in General**
- **Decision Tree**
  - **Classification tree**
  - **Regression tree**
- **Ensemble Methods**
  - **Bagging: Random Forest**
  - **Boosting: XGBoost**

Decision Tree Components



Root node

It refers to the start of the decision tree with maximum split (information Gain)

Internal Node

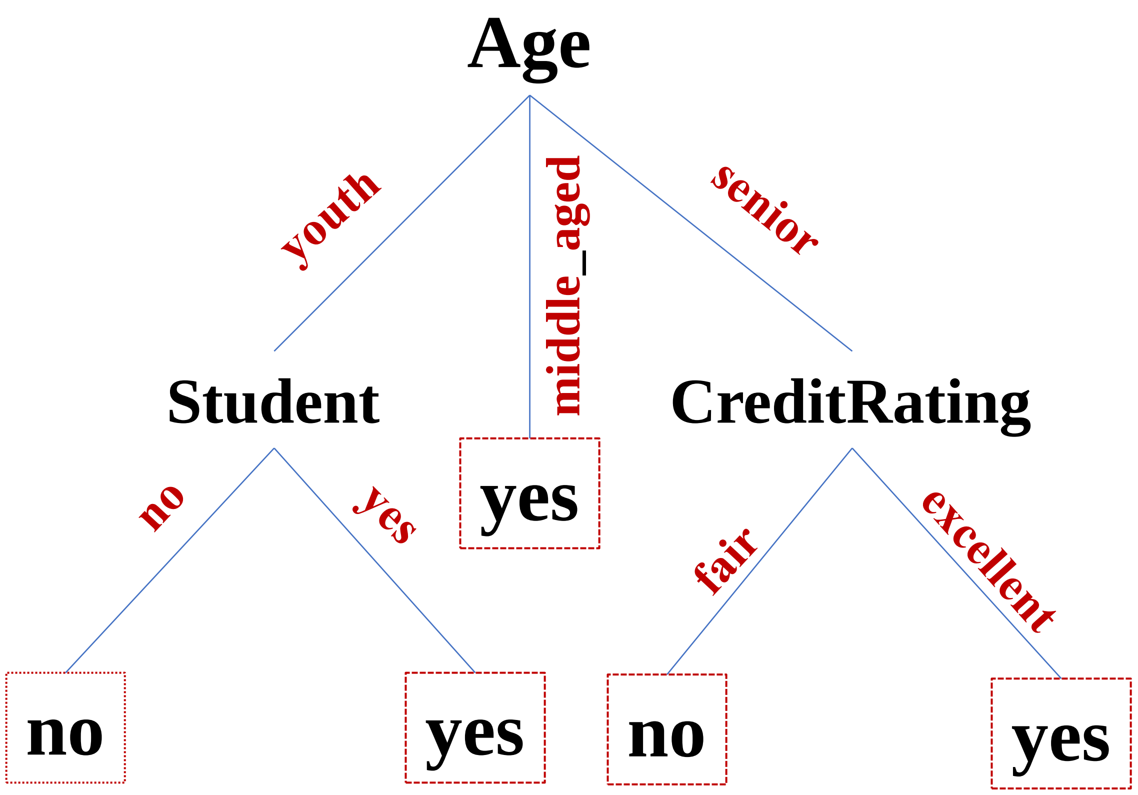
Node is a condition with multiple outcomes in the tree.

Leaf Node

This is the final decision(end point) of a node from the condition(question)

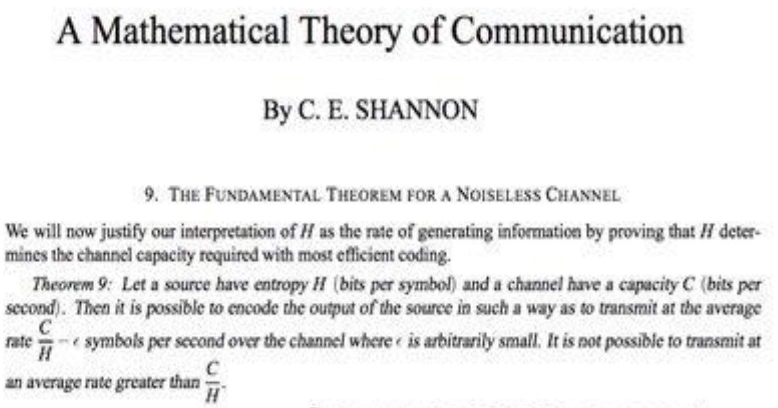
Decision Tree

Age	Income	Student	CreditRating	BuyComputer
youth	high	no	fair	no
youth	high	no	excellent	no
middle_aged	high	no	fair	yes
senior	medium	no	fair	yes
senior	low	yes	fair	yes
senior	low	yes	excellent	no
middle_aged	low	yes	excellent	yes
youth	medium	no	fair	no
youth	low	yes	fair	yes
senior	medium	yes	fair	yes
youth	medium	yes	excellent	yes
middle_aged	medium	no	excellent	yes
middle_aged	high	yes	fair	yes
senior	medium	no	excellent	no



'Buys Computer?' Decision Tree (Han, Kamber & Pei, 2011)

Node Selection of Decision Tree: Entropy–Based Measure



A Mathematical Theory of Communication

By C. E. SHANNON

9. THE FUNDAMENTAL THEOREM FOR A NOISELESS CHANNEL

We will now justify our interpretation of H as the rate of generating information by proving that H determines the channel capacity required with most efficient coding.

Theorem 9: Let a source have entropy H (bits per symbol) and a channel have a capacity C (bits per second). Then it is possible to encode the output of the source in such a way as to transmit at the average rate $\frac{C}{H} - \epsilon$ symbols per second over the channel where ϵ is arbitrarily small. It is not possible to transmit at an average rate greater than $\frac{C}{H}$.

Claude Shannon，1948

$$H(X) = -\sum_{j=1}^{m} p_j log p_j$$

Low Entropy

P(r) = 1
P(g) = 0

Medium Entropy

P(r) = 0.75
P(g) = 0.25

High Entropy

P(r) = 0.5
P(g) = 0.5

Low Uncertainty

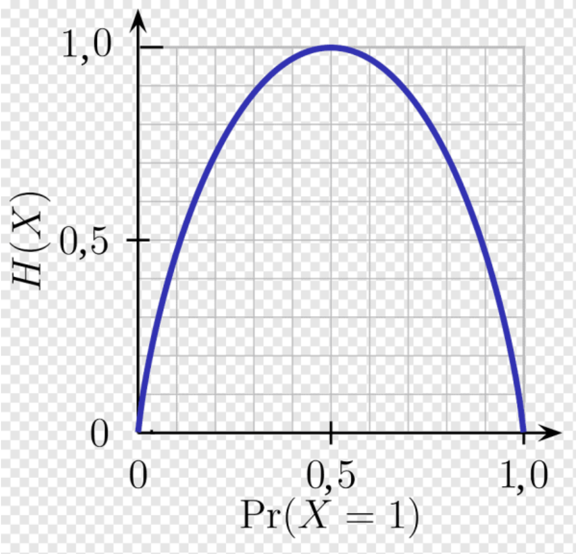
- 1 log1 = 0

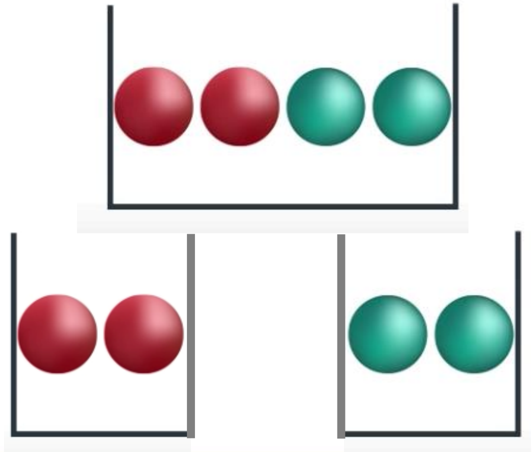
Medium Uncertainty

- 0.75 log0.75 –0.25log 0.25 = 0.81

High Uncertainty

- 0.5 log0.5 – 0.5 log0.5 = 1





$$GAIN_{split} = Entropy(p) - \left(\sum_{i=1}^{k} \frac{n_i}{n} Entropy(i) \right)$$

Node Selection of Decision Tree: Information Gain

Entropy I(X)

$$I(X) = -\sum_{j=1}^{m} p_j log p_j$$

Entropy for a Partition

$$I(X,Y) = -\sum_{i=1}^{k} \frac{S_i}{S} * H(X_i)$$

Information Gain

$$GAIN_{split} = Entropy(p) - \left(\sum_{i=1}^{k} \frac{n_i}{n} Entropy(i) \right)$$

$$\boldsymbol{I(X) = -\left(\frac{40}{80} log_2 \frac{40}{80} + \frac{40}{80} log_2 \frac{40}{80}\right) = 1}$$

40 40

Iris Versicolor Iris Setosa

Petal Length <= 2.45

$$\boldsymbol{I(X \mid PetalLengthleft) = -\left(\frac{35}{40} log_2 \frac{35}{40} + \frac{5}{40} log_2 \frac{5}{40}\right) = 0.54}$$

35 5

Iris Versicolor Iris Setosa

$$\boldsymbol{I(X \mid PetalLengthright) = -\left(\frac{5}{40} log_2 \frac{5}{40} + \frac{35}{40} log_2 \frac{35}{40}\right) = 0.54}$$

5 35

Iris Versicolor Iris Setosa

$$\boldsymbol{IG = 1 - \frac{40}{80} * 0.54 - \frac{40}{80} * 0.54 = 0.46}$$

Evaluate Classification Tree

		Actual class	
		Cat	Dog
Predicted class	Cat	5	2
	Dog	3	3

		Actual class	
		P	N
Predicted class	P	TP	FP
	N	FN	TN

true positive (TP)
eqv. with hit
true negative (TN)
eqv. with correct rejection
false positive (FP)
eqv. with false alarm, Type I error
false negative (FN)
eqv. with miss, Type II error

Precision (Positive Predictive Value: PPV)

$$\mathrm{PPV} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

Sensitivity, Recall, or True Positive Rate (TPR)

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{P}} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

Accuracy (ACC)

$$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{P} + \mathrm{N}} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}$$

F1 Score: the harmonic mean of precision and sensitivity

$$\mathrm{F}_1 = 2 \cdot \frac{\mathrm{PPV} \cdot \mathrm{TPR}}{\mathrm{PPV} + \mathrm{TPR}} = \frac{2\mathrm{TP}}{2\mathrm{TP} + \mathrm{FP} + \mathrm{FN}}$$

wikipedia

Decision Tree: Regression

cyl	hp	mpg
6	110	21
6	110	21
4	93	22.8
6	110	21.4
8	175	18.7
6	105	18.1
8	245	14.3
4	62	24.4
4	95	22.8
6	123	19.2
6	123	17.8
8	180	16.4
8	180	17.3
8	180	15.2
8	205	10.4
8	215	10.4
8	230	14.7
4	66	32.4
4	52	30.4
4	65	33.9
4	97	21.5
8	150	15.5
8	150	15.2
8	245	13.3
8	175	19.2
4	66	27.3
4	91	26
4	113	30.4
8	264	15.8
6	175	19.7
8	335	15
4	109	21.4

cyl = 6, 8

yes

no

hp >= 192

yes

no

13

cyl	hp	mpg
8	205	10.4
8	215	10.4
8	230	14.7
8	245	14.3
8	245	13.3
8	264	15.8
8	335	15
		13.4

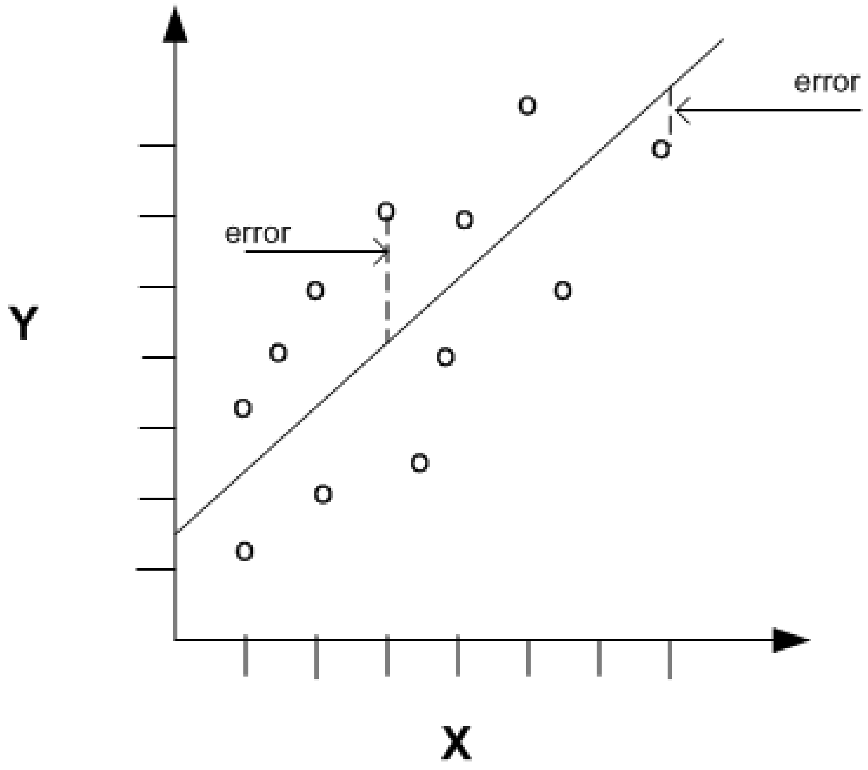
18

cyl	hp	mpg
6	105	18.1
6	110	21
6	110	21
6	110	21.4
6	123	19.2
6	123	17.8
8	150	15.5
8	150	15.2
6	175	19.7
8	175	18.7
8	175	19.2
8	180	16.4
8	180	17.3
8	180	15.2
		18.3

27

cyl	hp	mpg
4	93	22.8
4	62	24.4
4	95	22.8
4	66	32.4
4	52	30.4
4	65	33.9
4	97	21.5
4	66	27.3
4	91	26
4	113	30.4
4	109	21.4
		26.7

Evaluate Regression Tree



Mean squared error	$\text{MSE} = \frac{1}{n}\sum_{t=1}^{n} e_t^2$
Root mean squared error	$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n} e_t^2}$
Mean absolute error	$\text{MAE} = \frac{1}{n}\sum_{t=1}^{n} \lvert e_t \rvert$
Mean absolute percentage error	$\text{MAPE} = \frac{100\%}{n}\sum_{t=1}^{n} \left\lvert \frac{e_t}{y_t} \right\rvert$

Stackexchange

Decision Tree

Decision Trees are simple to understand and interpret, easy to use, versatile, and powerful

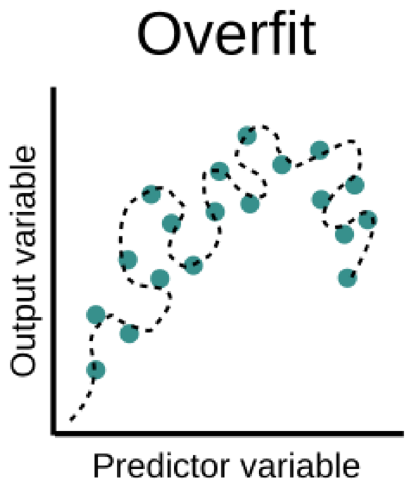
- Can generate understandable rules
- Perform classification without much computation
- Can handle continuous and categorical variables
- Provide a clear indication of which fields are most important for prediction or classification

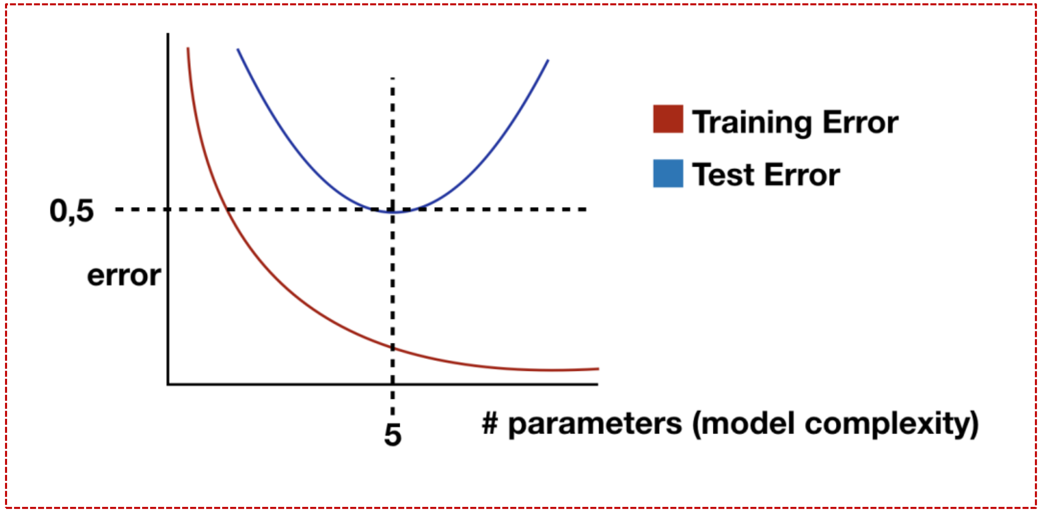
However, they do have a few limitations

- May make a complex tree with maximum depth
- Unstable as small variation in input data may result in completely different tree to get generated.
- As it is a greedy algorithm , may not find globally best tree for a data set .

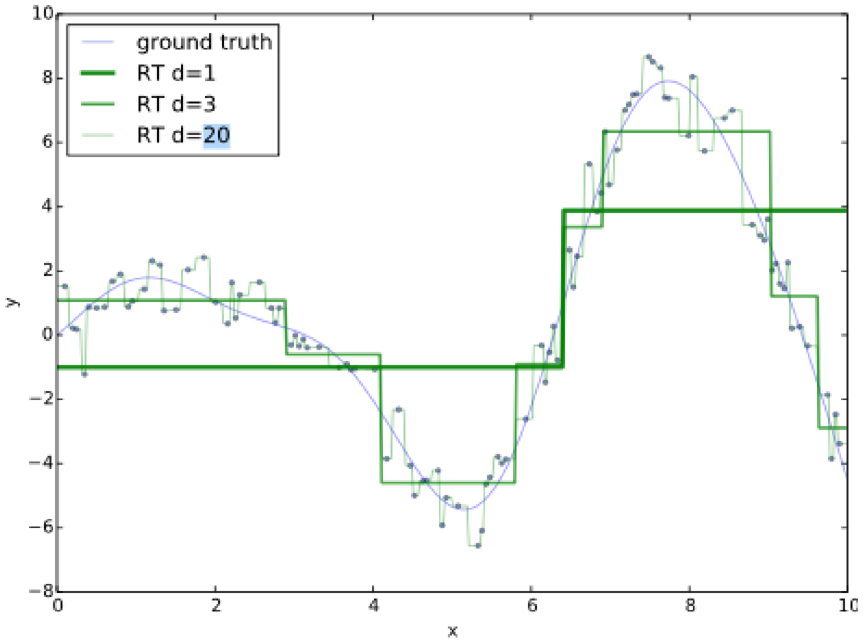
Decision Tree: Shortcomings

Decision Tree Overfit



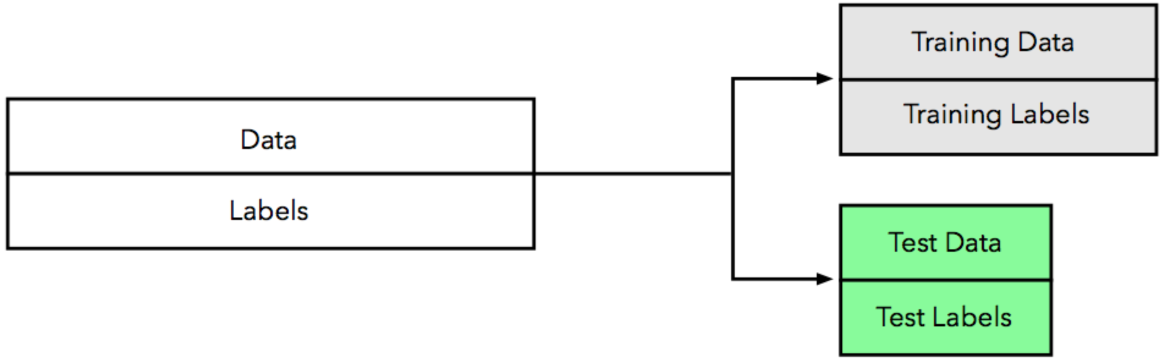


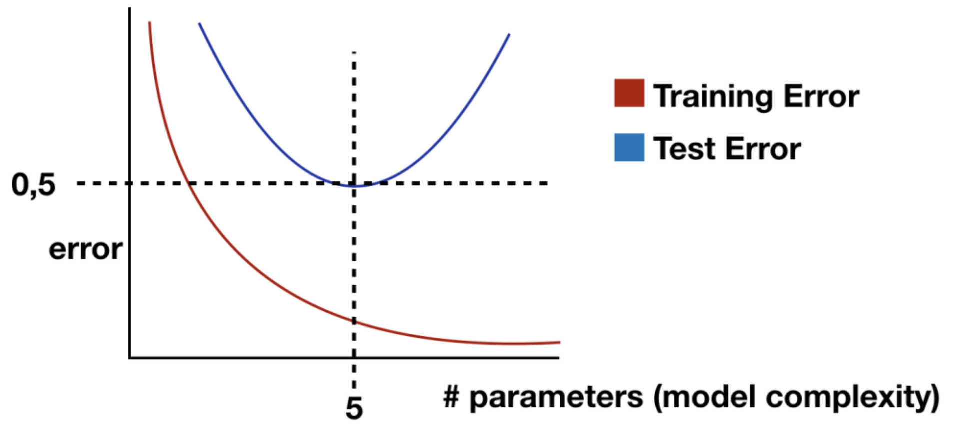
Decision Tree Unstable

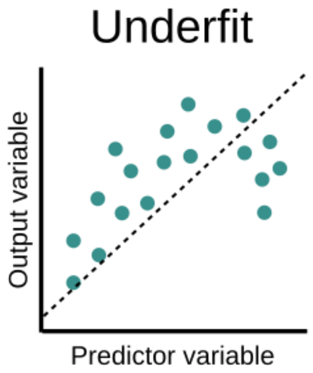


Decision tree is unstable because training a tree with a slightly different sub-sample caused the structure of the tree to change drastically.

Machine Learning Evaluation

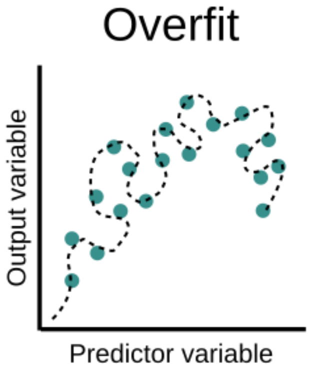






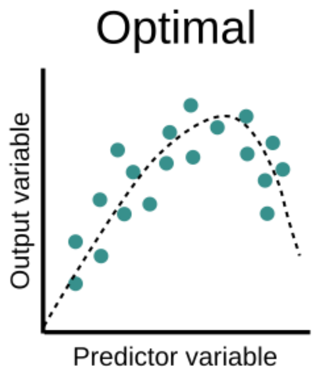
Large

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

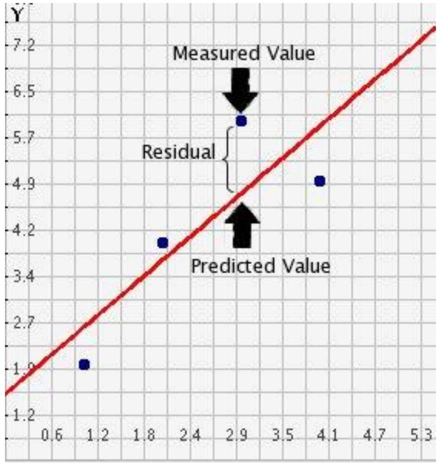


Small

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

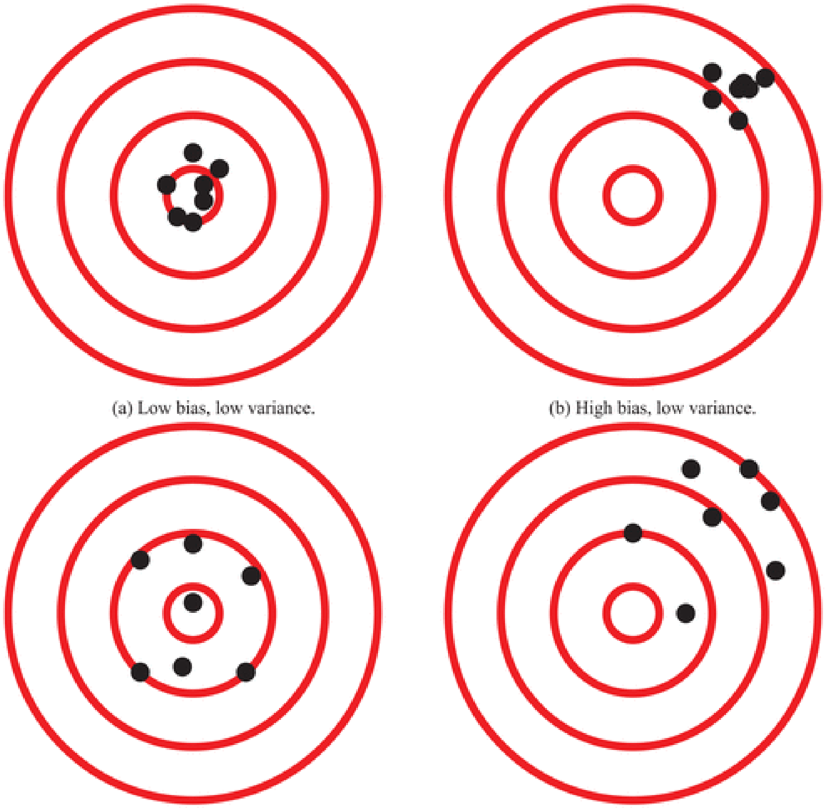


Variance vs Bias



(a) Bias and variance.

(b) Mean-squared error.



(a) Low bias, low variance. (b) High bias, low variance.

(c) Low bias, high variance. (d) High bias, high variance.

- **The bias is how far the points are from the center of the target on average**
- **The variance is a measure of how far the points are to the centroid of the points on average**

Shayan Doroudi, 2020

Outline

- **Machine Learning in General**
- **Decision Tree**
- **Ensemble Methods**
  - **Bagging: Random Forest**
  - **Boosting: XGBoost**

Ensemble Methods

The principle behind ensembles is the idea of "wisdom of the crowd". The collective predictions of many diverse models is better than any set of predictions made by a single model.

In ensemble learning theory, we call weak learners (base models) that can be used as building blocks for designing more complex models by combining several of them.

Weak Learner	Weak Learner
High Variance	**High Bias**
Low degree of freedom models	High degree of freedom models
Combining weak learners is to try reducing variance	Combining weak learners is to try reducing bias
Strong Learner	**Strong Learner**

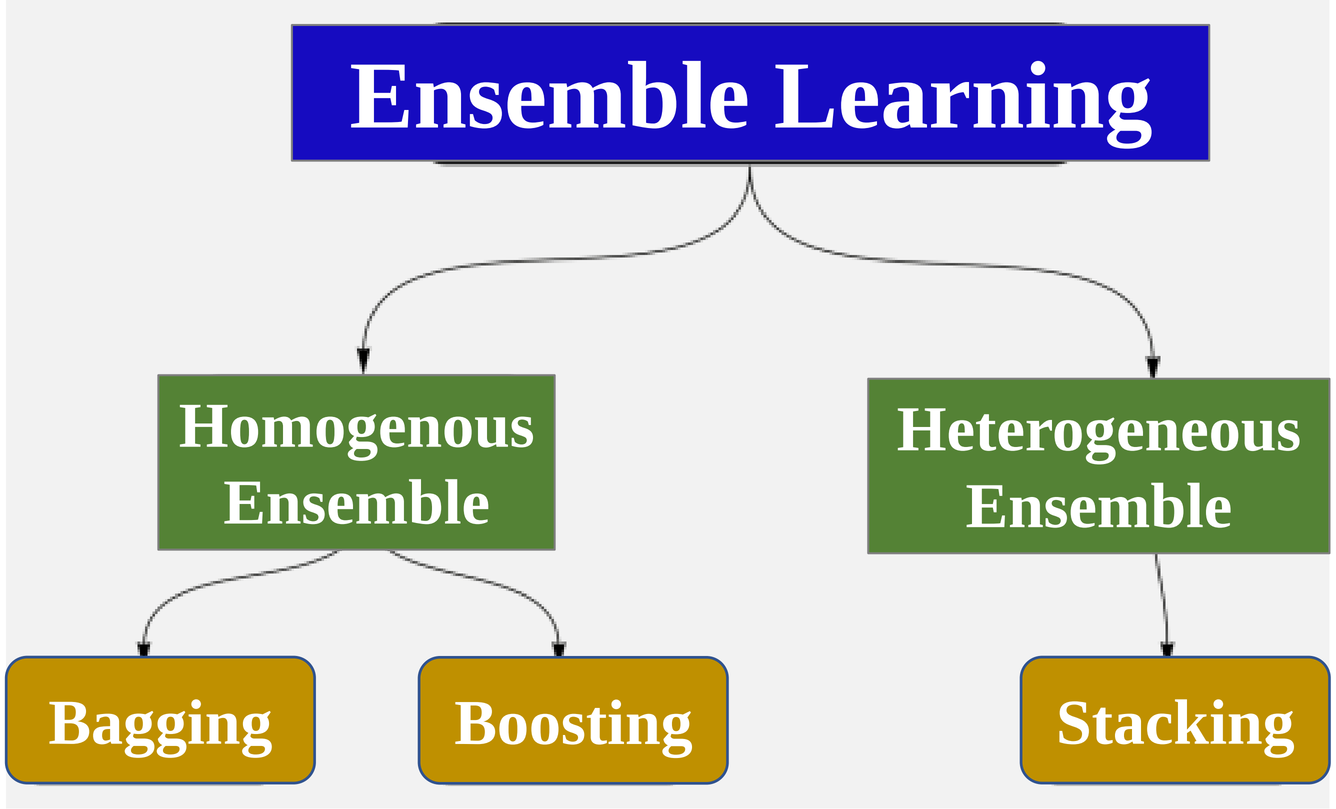
Ensemble Methods

Bagging

Combing homogenous weak learners, learns them independently from each other in parallel and combines them following some kind of deterministic averaging process

Boosting

Combing homogenous weak learners, learns them in a very additive way (a base model depends on the previous ones) and combines them following a deterministic strategy



Ensemble Learning

Assumption

- Each predictor makes error with probability p
- The predictors are independent

Majority voting of n predictors

- k predictors make an error

$$\binom{n}{k} p^k (1-p)^{n-k}$$

- Majority makes an error

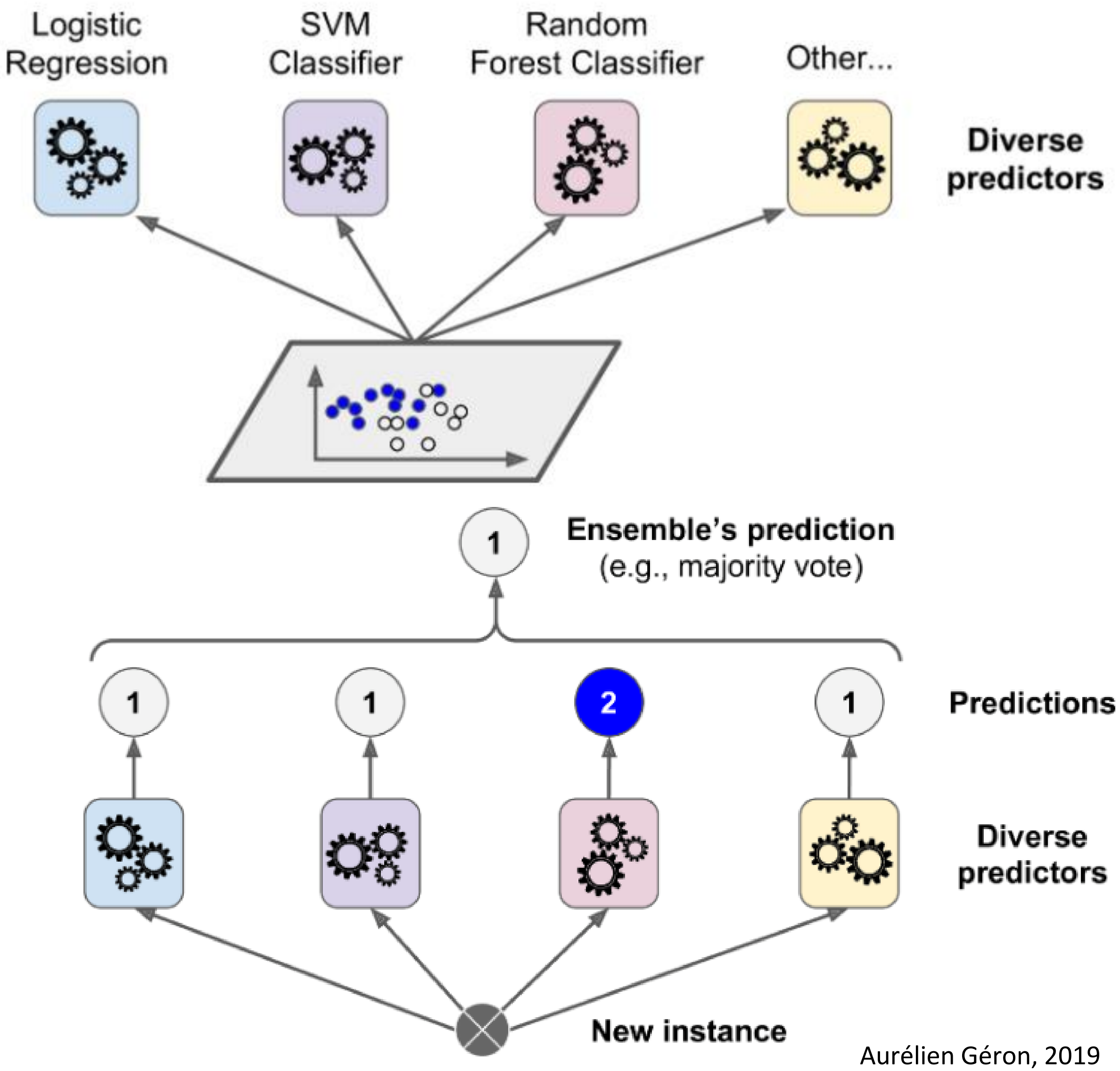
$$\sum_{k>n/2} \binom{n}{k} p^k (1-p)^{n-k}$$

- With n = 5, p = 0.2, majority error (3)

 Error = 0.2*0.2*0.2*(1-0.2)*(1-0.2)

 = 0.0051 < 0.01

 n = 20, p = 0.2; error = $2.7 * 10^{-9}$

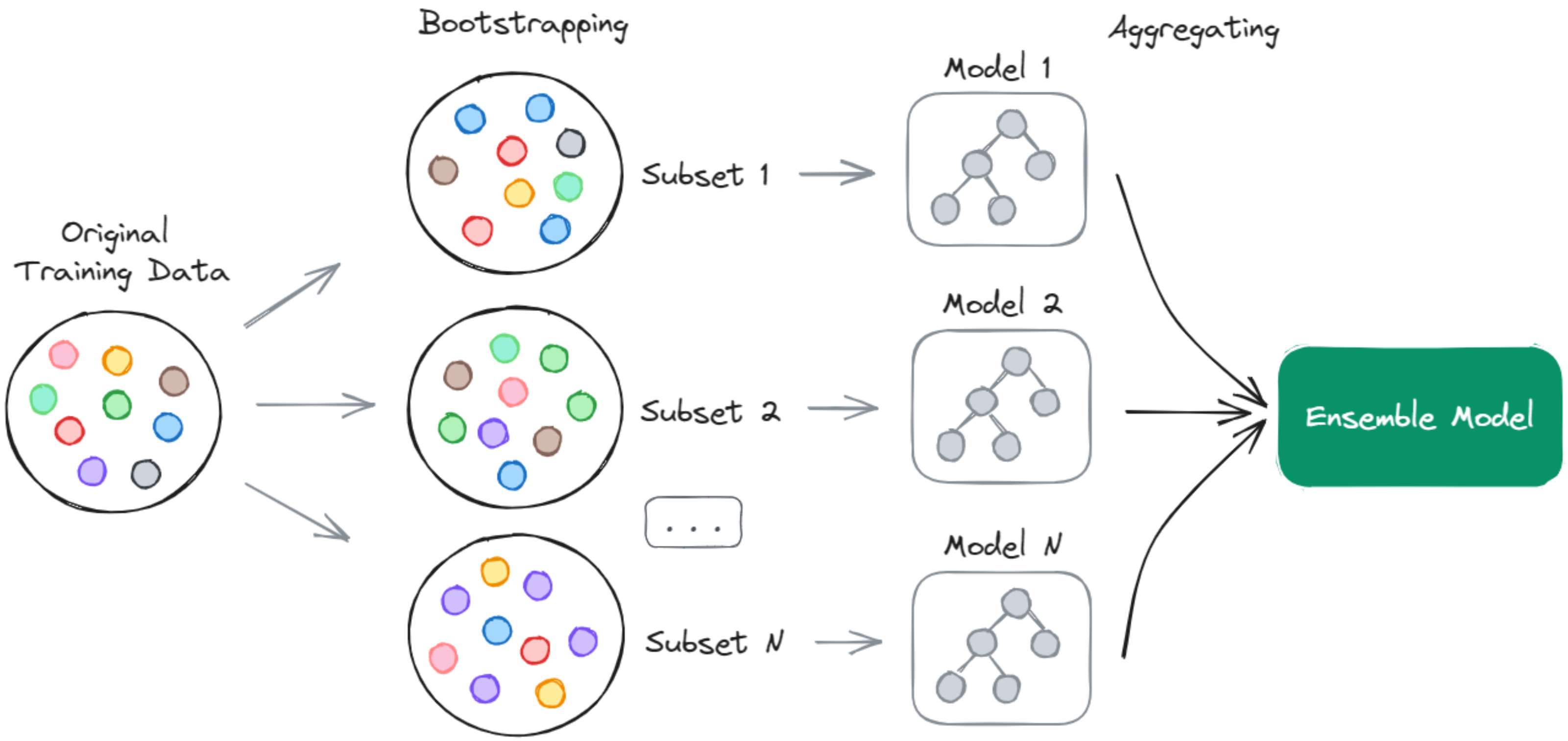


Aurélien Géron, 2019

Bootstrap + Aggregating

Ensemble Methods: Bagging

Bagging

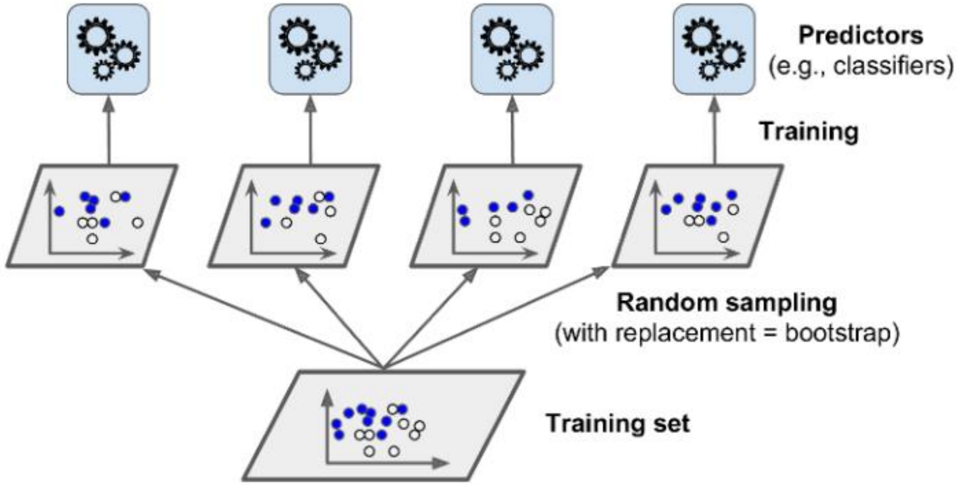


A large number of independent weak models are combined to learn the same task with the same goal. Each weak learner is trained on a random subsample of data sampled with replacement (bootstrapping, and then the models' predictions are aggregated).

Bagging can be applied to both classification and regression problems. For regression problems, the final predictions will be an average (soft voting) of the predictions from base estimators. For classification problems, the final predictions will be the majority vote (hard voting)

Tree-based Ensemble Learning: Bagging

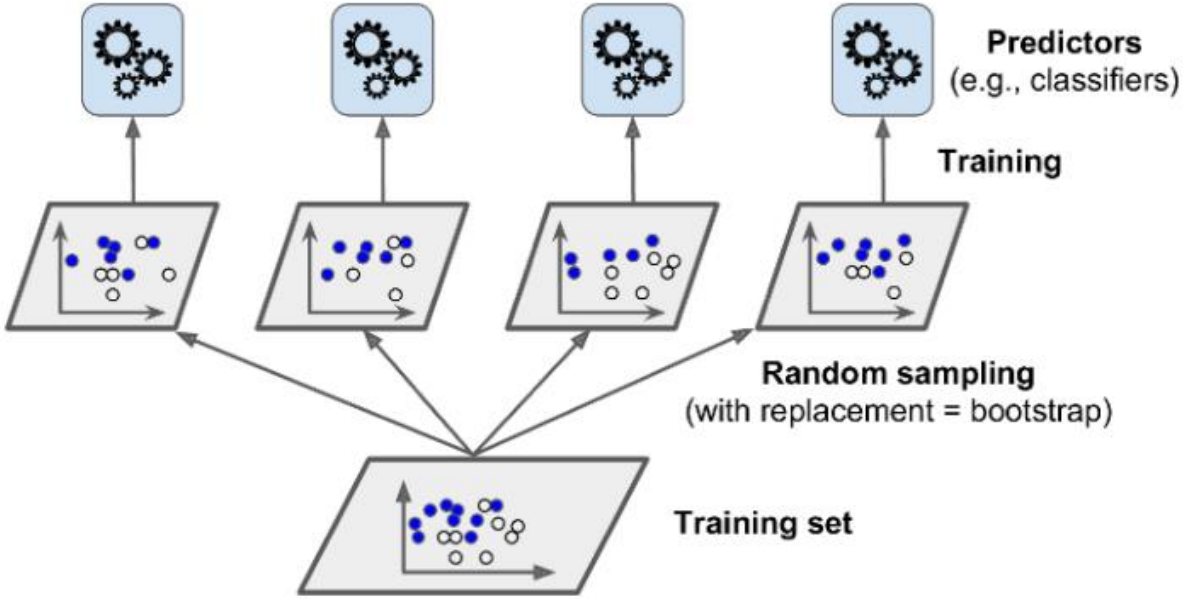
- When sampling is performed with replacement, this method called bagging (bootstrap aggregation).
  - A bootstrap sample is chosen at random with replacement from the data. Some observations end up in the bootstrap sample more than once, while others are not included ("out of bag")
- When sampling is performed without replacement, it is called pasting.
- Bagging reduces the variance of the base learner but has limit effect on the bias
- It is most effective if we use strong base learner that have very little bias but high variance. E.g. tree.



Leo Breiman, 1996

Random Forest

- Grow a forest of many trees (scikit learn default = 100)
- Grow each tree on an independent bootstrap sample from the training data.
- At each node:
  - Select m variables at random out of all M possible variables (independently for each node)
  - Find the best split on the selected m variables.
- Growth the trees to maximum depth
- Vote/average the trees to get predictions for new data



Sample N cases at random with replacement

Leo Breiman, 1996

Random Forest

Advantages	Explanation
Ability to learn non-linear decision boundary	It can model complex, non-linear relationships between features and the target variable.
High Accuracy	It reduces overfitting problem in decision trees and helps to improve the accuracy
Flexible and robust	It can handle a wide variety of data types including numeric and categorical data. It can handle outliers and missing values, and does not require feature scaling as it uses rule based approach instead of distance calculation.
Feature importance	Random forest provides information about the importance of each feature in the data.
Scalability	It can handle large datasets with high dimensionality.
Parallel processing	Trees can be created in parallel, since there is no dependence between iterations. Which can speed up the training time.

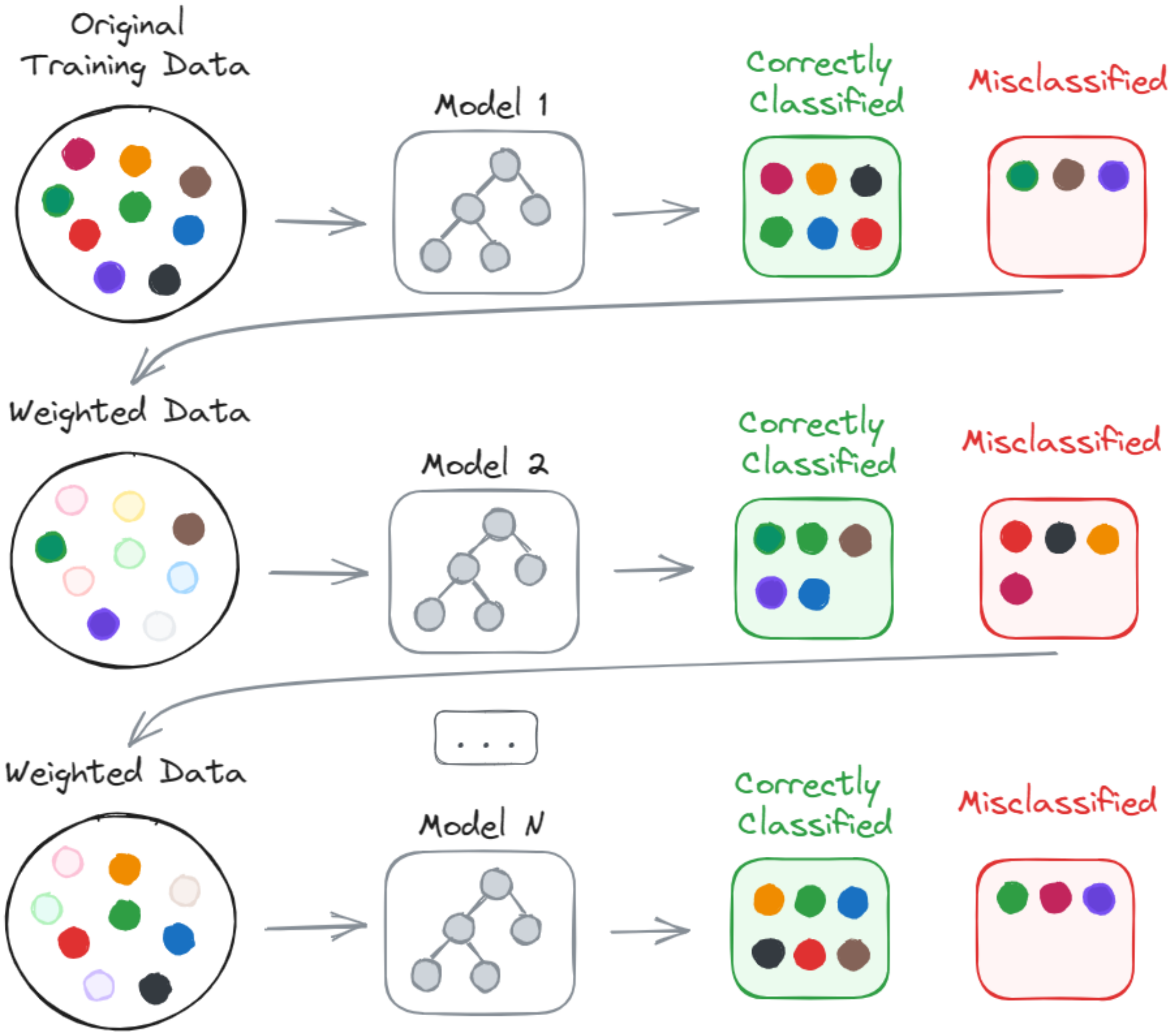
Ensemble Methods: Boosting

Boosting

The first boosting algorithm is AdaBoost which is created by Freund & Shapire in 1996.

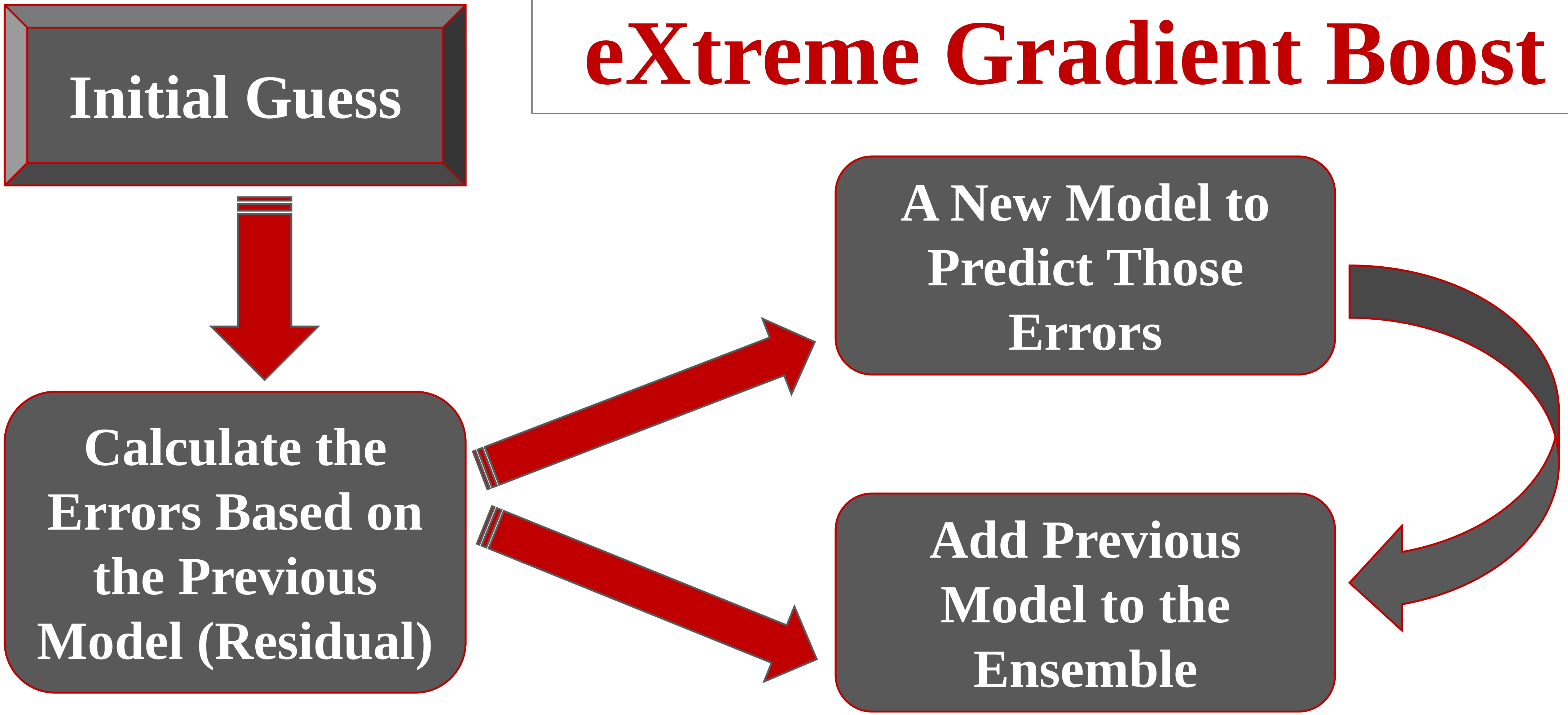
Relies on weak learners which only need to generate a hypothesis with a training accuracy greater than 0.5.

Examples are given weights. At each iteration, a new hypothesis is learned and the example are reweighted to focus the system on examples that the most recently learned classifier got wrong.



XGBoost

eXtreme Gradient Boost



XGBoost

- Training Loss: it is nothing but the difference between actual and predicted values, and it is used to measure how well model fit on training data.
- Regularization: measures complexity of trees.

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k)$$

Trade off:

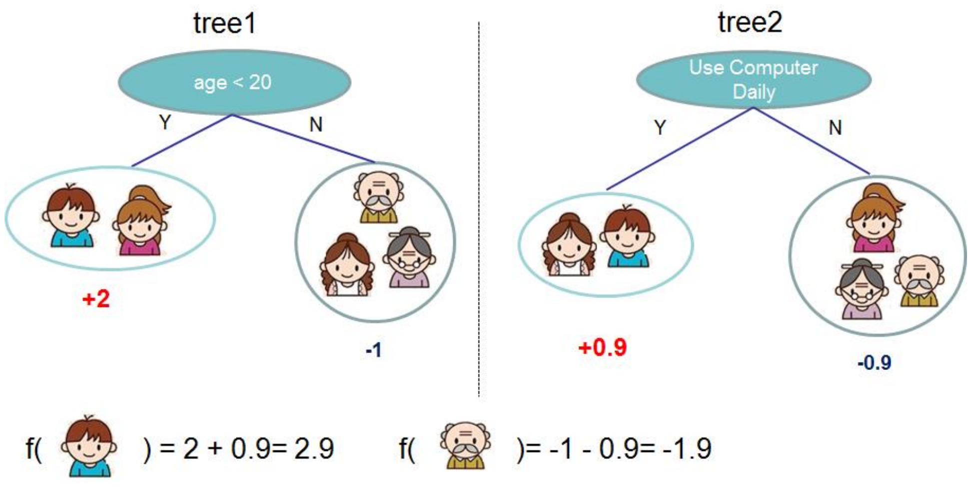
- Optimizing training Loss tend to create more complicated models.
- Optimizing regularization tends to generalize simpler models.

XGBoost

$$obj = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t)}\right) + \sum_{i}^{t} \Omega(f_i)$$

$$Obj\ Model = Training\ Loss + Regularization$$

Additive Boosting



$$\hat{y}_t = \hat{y}_{t-1} + \eta \Delta_t(X)$$

Current Model, Previous Model, New Model

$$Obj^{(t)} = \sum_{i=1}^{n} \left(l(y_i, \hat{y}_i^{t-1}) + f_{t-1}(x_i) \right) + \Omega(f_t)$$

Taylor Approximation

$$f(x + \Delta x) \approx f(x) + f'(x)\Delta x + \frac{1}{2} f''(x)\Delta x^2$$

$$Obj^{(t)} = \sum_{i=1}^{n} [l(y_i, \hat{y}_i^{t-1}) + g_i f_t(x_i) + \frac{1}{2} h_i f_i^2(x_i))] + \Omega(f_t)$$

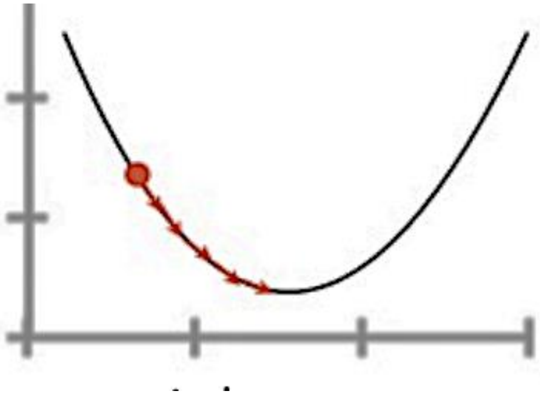
$$g_i = \partial_{\hat{y}^t} l(\hat{y}^t - y_i)^2 \qquad h_i = \partial_{\hat{y}^t}^2 l(\hat{y}^t - y_i)^2$$

Ensemble Methods: XGBoost

Month	Weight	Residual	Average
2	9	-5	14
4	11	-3	14
6	16	2	14
?	20	6	14

Regression $L(y_i, p_i) = \frac{1}{2}(y_i - p_i)^2$

Classification $L(y_i, p_i) = -[y_i log(p_i) + (1 - y_i) log(1 - p_i)]$



$$\mathcal{L}^{(t)} \simeq \sum_{i=1}^{n} [l(y_i, \hat{y}^{(t-1)}) + g_i f_t(\mathbf{x}_i) + \frac{1}{2} h_i f_t^2(\mathbf{x}_i)] + \Omega(f_t)$$

$$\sum_{i=1}^{n} L(y_i, p_i) = L(y_1, p_1^0) + g_1 O_{val} + \frac{1}{2} h_1 O_{val}^2$$
$$+ L(y_2, p_2^0) + g_2 O_{val} + \frac{1}{2} h_2 O_{val}^2$$
$$+ L(y_n, p_n^0) + g_n O_{val} + \frac{1}{2} h_n O_{val}^2 + \frac{1}{2} \lambda O_{val}^2$$
$$= (g_1 + g_2 + \cdots + g_n) O_{val} + \frac{1}{2}(h_1 + h_2 + \cdots + h_n + \lambda) O_{val}^2$$

Ensemble Methods: XGBoost

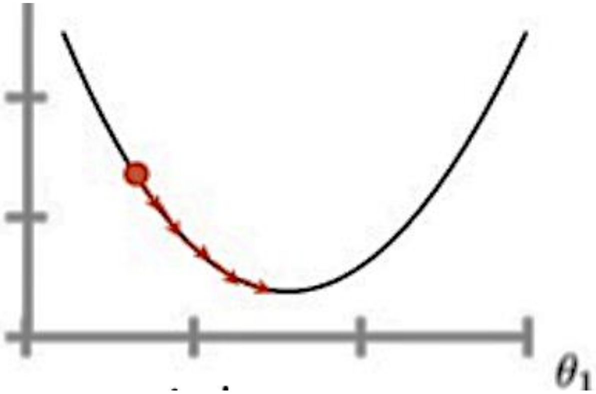
XGBoost is approximate greedy algorithm.

Month	Weight	Residual	Average
2	9	-5	14
4	11	-3	14
6	16	2	14
?	20	6	14

$$= (g_1 + g_2 + \cdots + g_n)O_{val} + \frac{1}{2}(h_1 + h_2 + \cdots + h_n + \lambda)O_{val}^2$$

$$\frac{d}{dO_{val}}(g_1 + g_2 + \cdots + g_n)O_{val} + \frac{1}{2}(h_1 + h_2 + \cdots + h_n + \lambda)O_{val}^2 = 0$$

$$O_{val} = \frac{-(g_1 + g_2 + \cdots + g_n)}{(h_1 + h_2 + \cdots + h_n + \lambda)}$$



Regression

$$L(y_i, p_i) = \frac{1}{2}(y_i - p_i)^2 \Rightarrow \begin{cases} g_i = -(y_i - p_i) \\ h_i = 1 \end{cases}$$

$$O_{val} = \frac{\sum Residuals}{(Number\ of\ Residuals + \lambda)}$$

$$L(y_i, p_i) = -[y_i log(p_i) + (1 - y_i)log(1 - p_i)] \Rightarrow \begin{cases} g_i = -(y_i - p_i) \\ h_i = p_i \times (1 - p_i) \end{cases}$$

Classification

$$O_{val} = \frac{\sum Residuals}{\sum[p_i^{Prev} \times (1 - p_i^{Prev})] + \lambda}$$

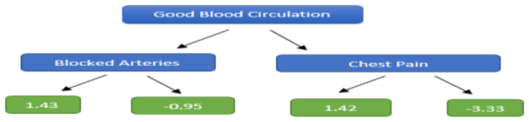
Ensemble Methods: XGBoost

Regression $$O_{val} = \frac{\sum Residuals}{(Number\ of\ Residuals + \lambda)}$$

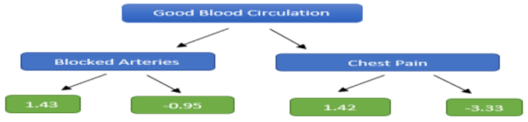
Classification $$O_{val} = \frac{\sum Residuals}{\sum[p_i^{Prev} \times (1 - p_i^{Prev})] + \lambda}$$

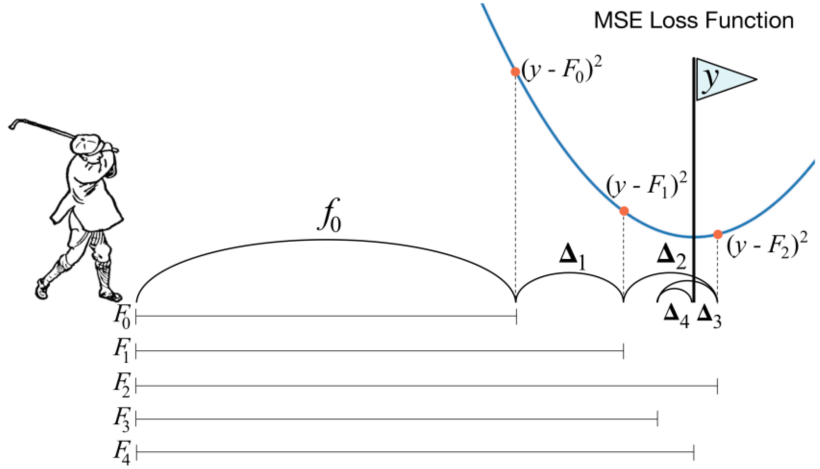
$$\hat{y}_t = \hat{y}_{t-1} + \eta \times O_{val}$$

Initial Guess + Learning Rate ×



+ Learning Rate ×

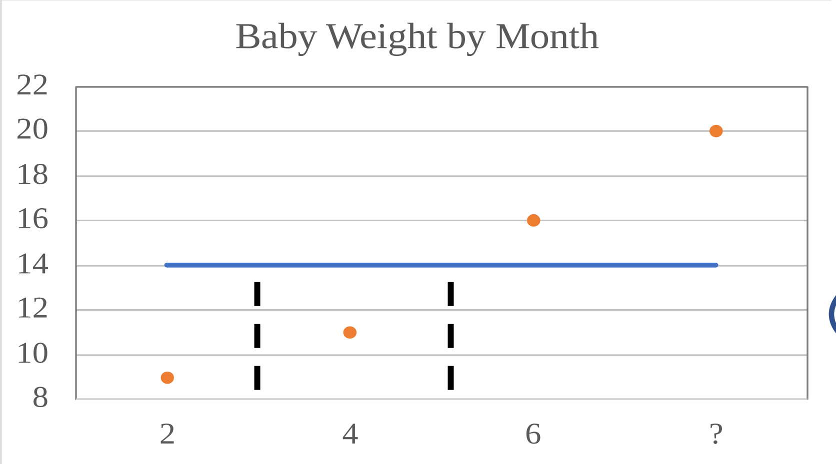




Ensemble Methods: XGBoost

XGBoost is sparse aware algorithm (sparsity: presence of missing data).

Month	Weight	Residual	Average
2	9	-5	14
4	11	-3	14
6	16	2	14
?	20	6	14



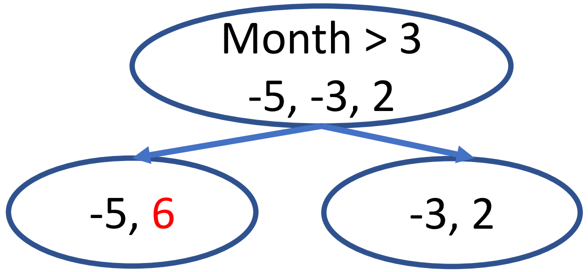
Regression

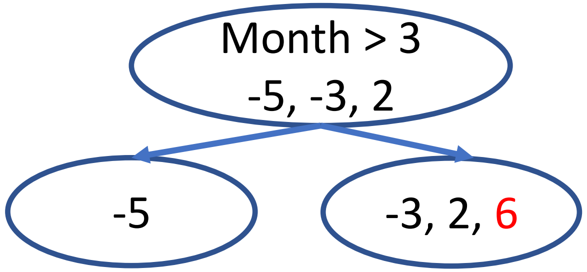
$$\textit{Similarity Score} = \frac{(\sum \textit{Residuals})^2}{\textit{Number of Residuals}}$$

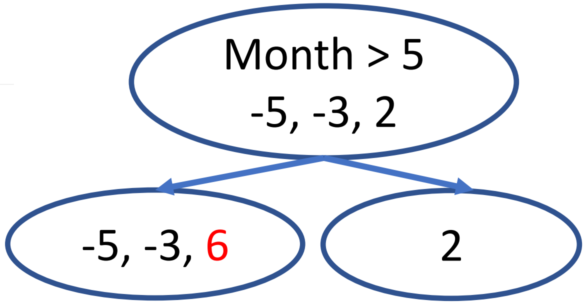
Classification

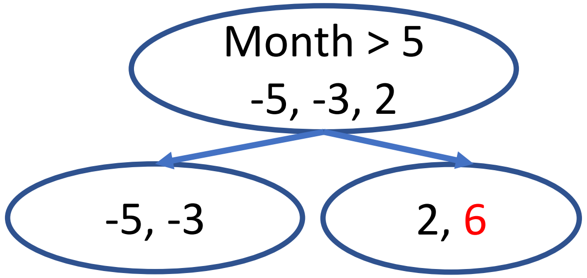
$$\textit{Similarity Score} = \frac{(\sum \textit{Residuals})^2}{\sum[\textit{PreP} \times (1 - \textit{PreP})]}$$

$$\textit{Gain} = \textit{Left}_{\textit{Similarity}} + \textit{Right}_{\textit{Similarity}} - \textit{Root}_{\textit{Similarity}}$$





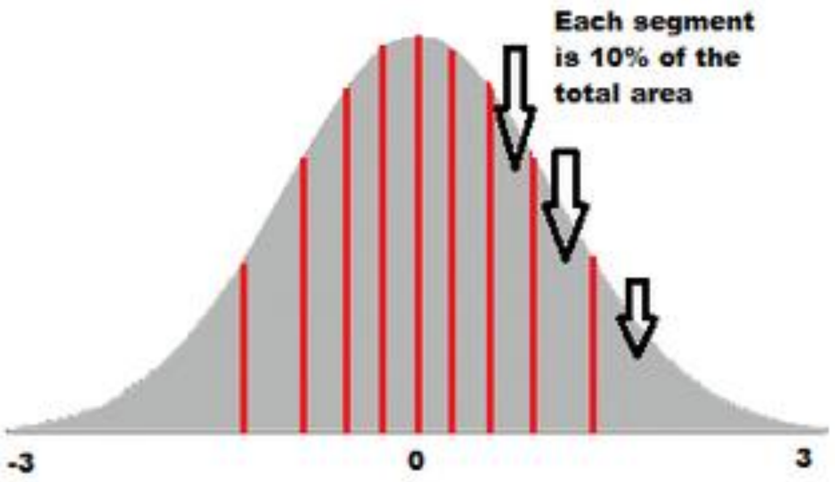




Ensemble Methods: XGBoost

XGBoost is approximate greedy algorithm.

Month	Weight	Residual	Average
2	9	-5	14
4	11	-3	14
6	16	2	14
?	20	6	14



- We need to try every single threshold to get the best splitting points for each variable.
- In order to save computing time, we could use quantiles thresholds to make approximate splitting. The more quantiles the model result will be more accurate, however it need more time to calculate. By default, XGBoost have about 33 quantiles.
- Usually percentiles of a feature are used to make candidates distributed evenly on the data.

Ensemble Methods: XGBoost

XGBoost is Cache Aware Access

Regression

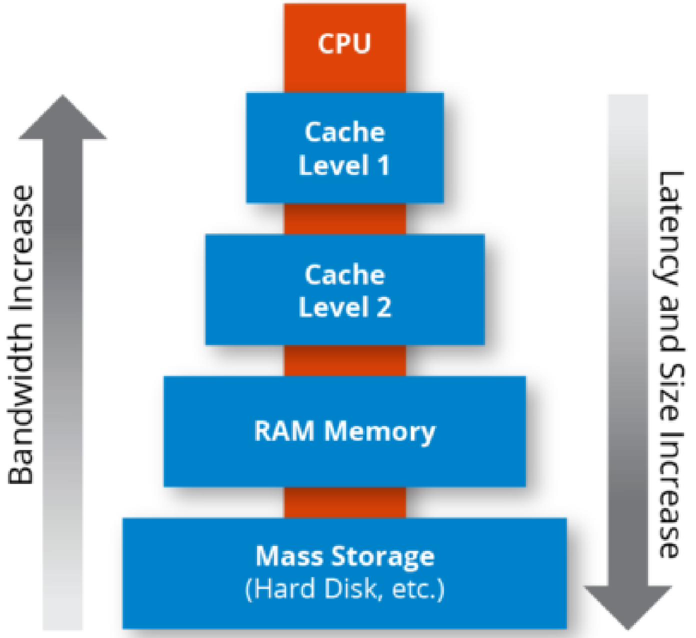
$$L(y_i, p_i) = \frac{1}{2}(y_i - p_i)^2 \Rightarrow \begin{cases} g_i = -(y_i - p_i) \\ h_i = 1 \end{cases}$$

$$O_{val} = \frac{\sum Residuals}{(Number\ of\ Residuals + \lambda)}$$

Classification

$$L(y_i, p_i) = -[y_i log(p_i) + (1 - y_i) log(1 - p_i)] \Rightarrow \begin{cases} g_i = -(y_i - p_i) \\ h_i = p_i \times (1 - p_i) \end{cases}$$

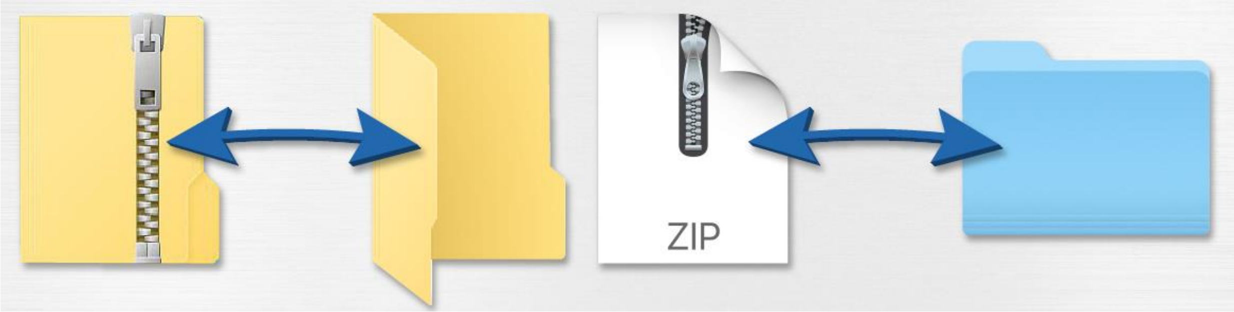
$$O_{val} = \frac{\sum Residuals}{\sum[p_i^{Prev} \times (1 - p_i^{Prev})] + \lambda}$$



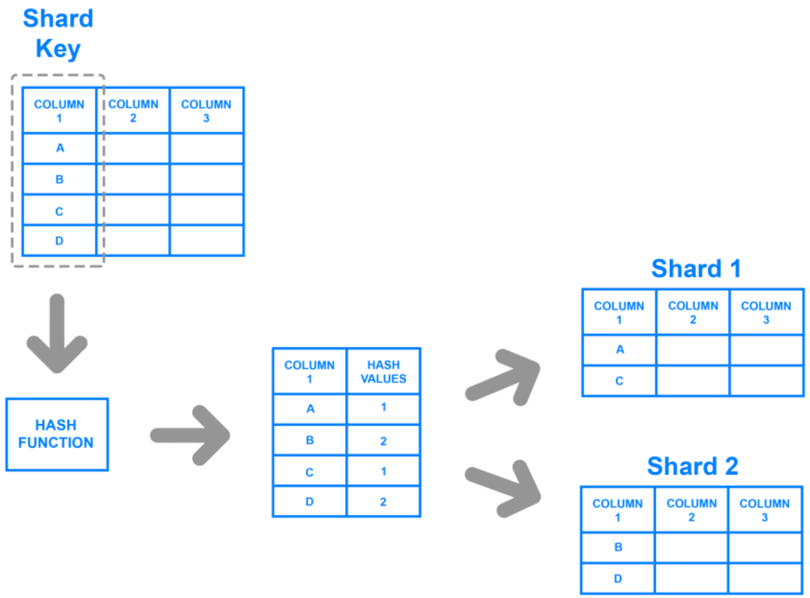
- XGBoost need a lot of time to calculate the Gradients and Hessians for each new functions.
- XGBoost store the Gradient and Hessian into the Cache so that it can rapidly calculate similarity scores and output values.

Ensemble Methods: XGBoost

XGBoost blocks for Out-of-Core Computation



Because reading and writing data to the hard drive is super slow, XGBoost tries to minimize these actions by compressing the data. Therefore, by spending a little bit of CPU time to uncompressing the data to avoid spending a lot of time to access the hard drive.



When there is more than one Hard Drive available for storage, XGBoost Sharding to speed up disk access.