



**BITS Pilani**  
Pilani Campus

# Machine Learning

## AIML CLZG565

### Ensemble Learning

Raja vadhana P  
Assistant Professor,  
BITS - CSIS

## Disclaimer and Acknowledgement



- The content for these slides has been obtained from books and various other source on the Internet
- I here by acknowledge all the contributors for their material and inputs.
- I have provided source information wherever necessary
- I have added and modified the content to suit the requirements of the course

**Source:** Slides of Prof. Chetana, Prof.Seetha, Prof.Sugata, Prof.Monali, Prof. Raja vadhana , Prof.Anita from BITS Pilani , CS109 and CS229 stanford lecture notes and many others who made their course materials freely available online.

# Course Plan

M1	Introduction & Mathematical Preliminaries
M2	Machine Learning Workflow
M3	Linear Models for Regression
M4	Linear Models for Classification
M5	Decision Tree
M6	Instance Based Learning
M7	Support Vector Machine
M8	Bayesian Learning
M9	Ensemble Learning
M10	Unsupervised Learning
M11	Machine Learning Model Evaluation/Comparison

# Ensemble Learning

# The Single Model Philosophy

---

- Motivation: Occam's Razor
  - “one should not increase, beyond what is necessary, the number of entities required to explain anything”
- Infinitely many models can explain any given dataset
  - Might as well pick the smallest one...

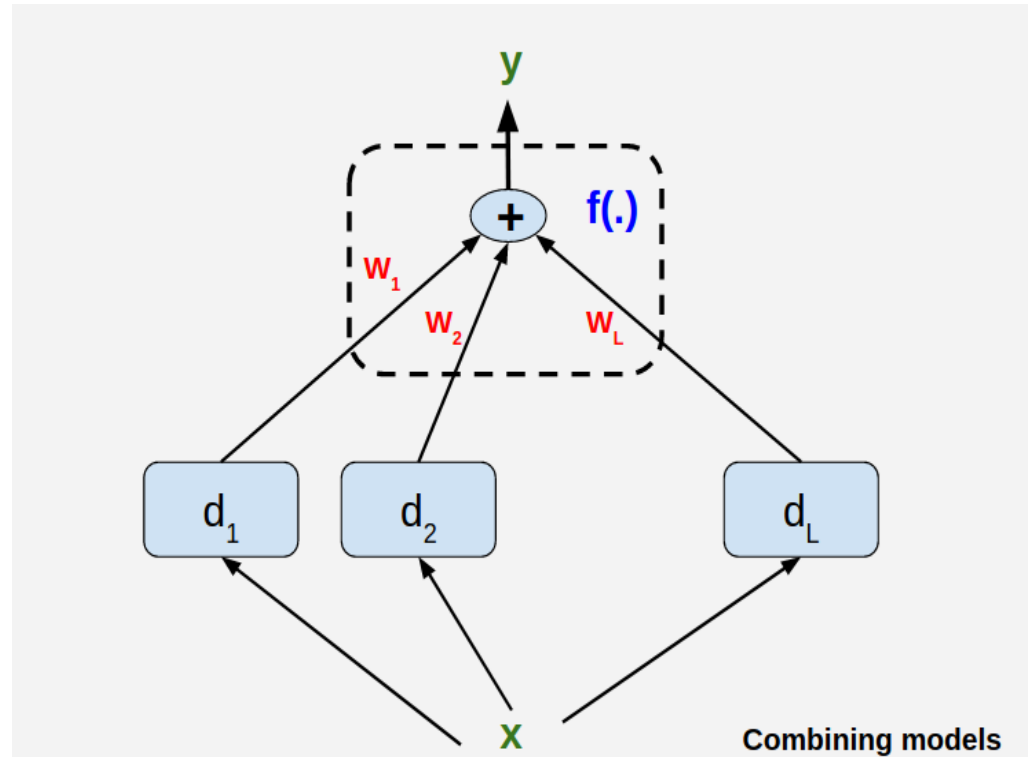
# Ensemble Philosophy

- No Free Lunch Theorem: There is no algorithm that is always the most accurate
- Each learning algorithm dictates a certain model that comes with a set of assumptions
  - Each algorithm converges to a different solution and fails under different circumstances
    - The best tuned learners could miss some examples and there could be other learners which works better on (may be only) those !
  - In the absence of a single expert ( a superior model ) , a committee (combinations of models) can do better !
    - A committee can work in many ways ...

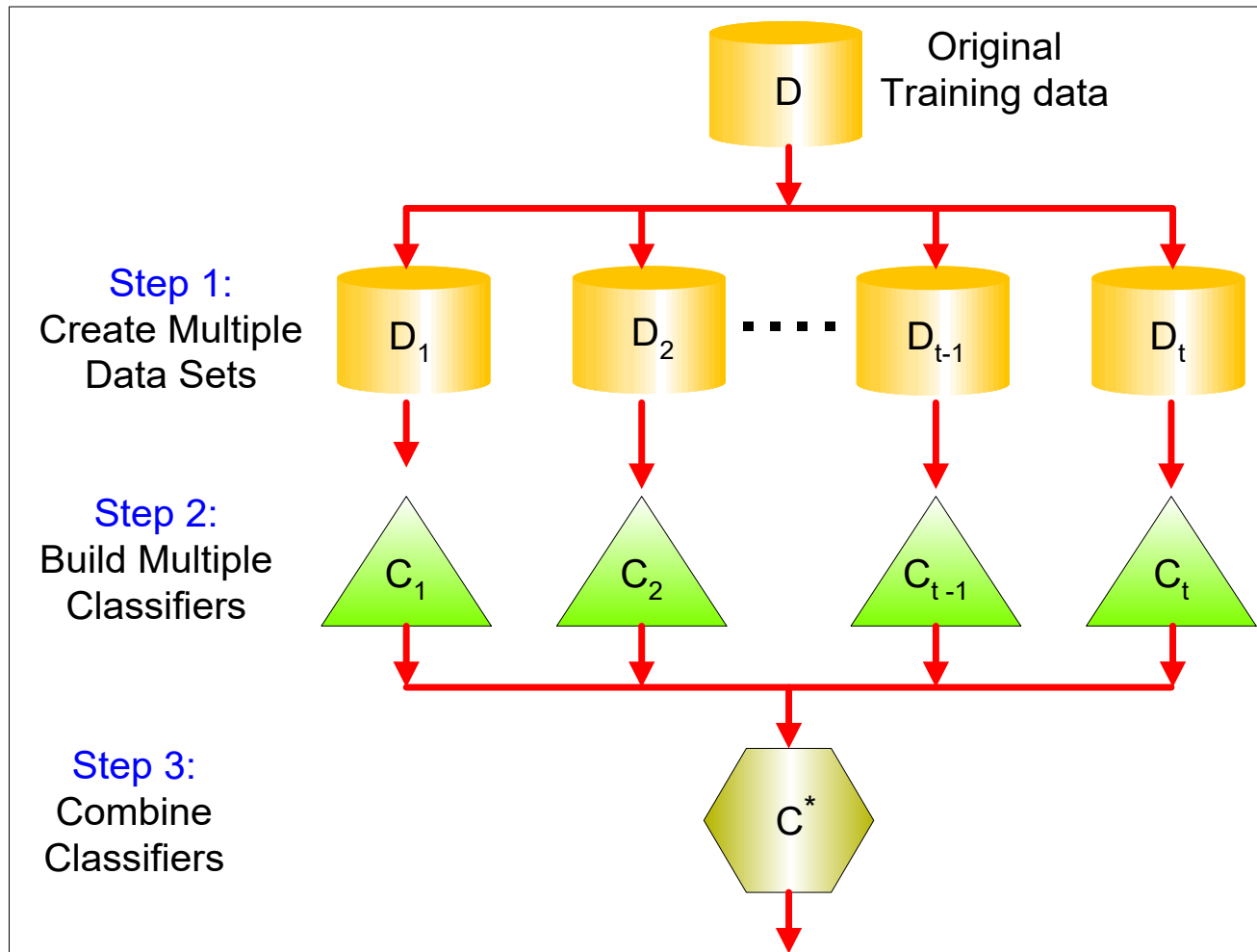
Weak learner does only slightly better than random guessing

# Committee of Models

- Committee Members are base learners !
- Major challenges dealing with this committee
  - Expertise of each of the members (Does it help / not?)
  - Combining the results from the members for better performance



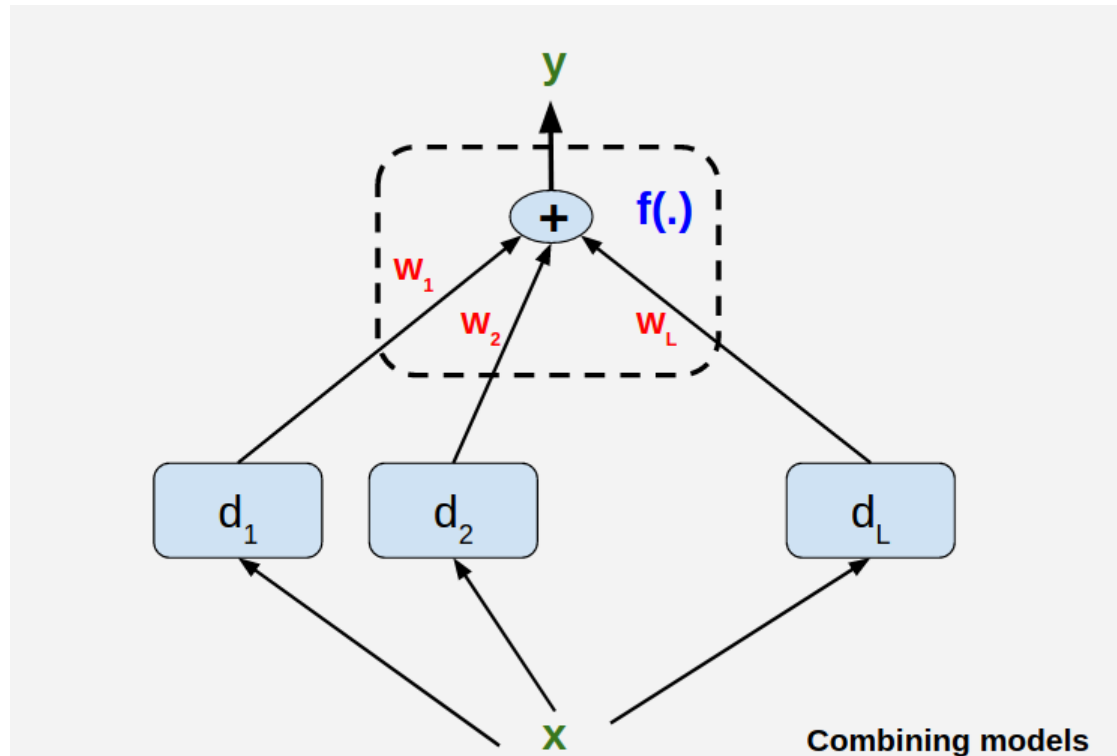
# General Approach





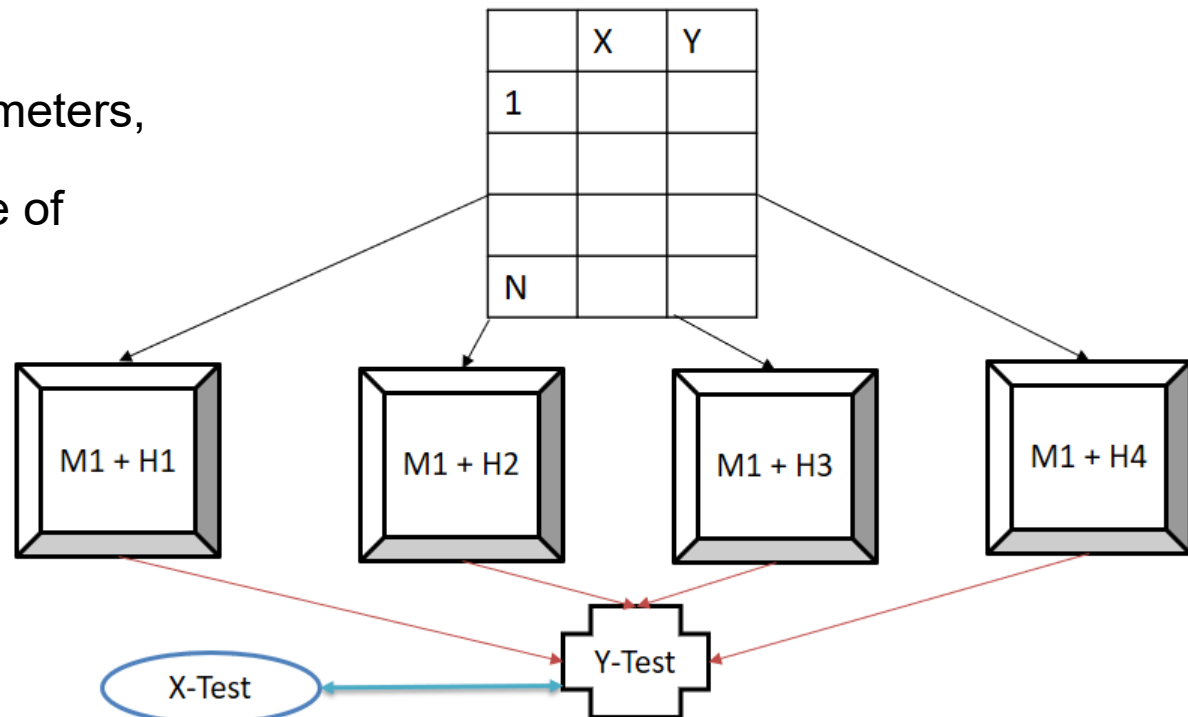
## Issue 1 : On the members ( Base Learners )

- It does not help if all learners are good/bad at roughly same thing
  - Need Diverse Learners



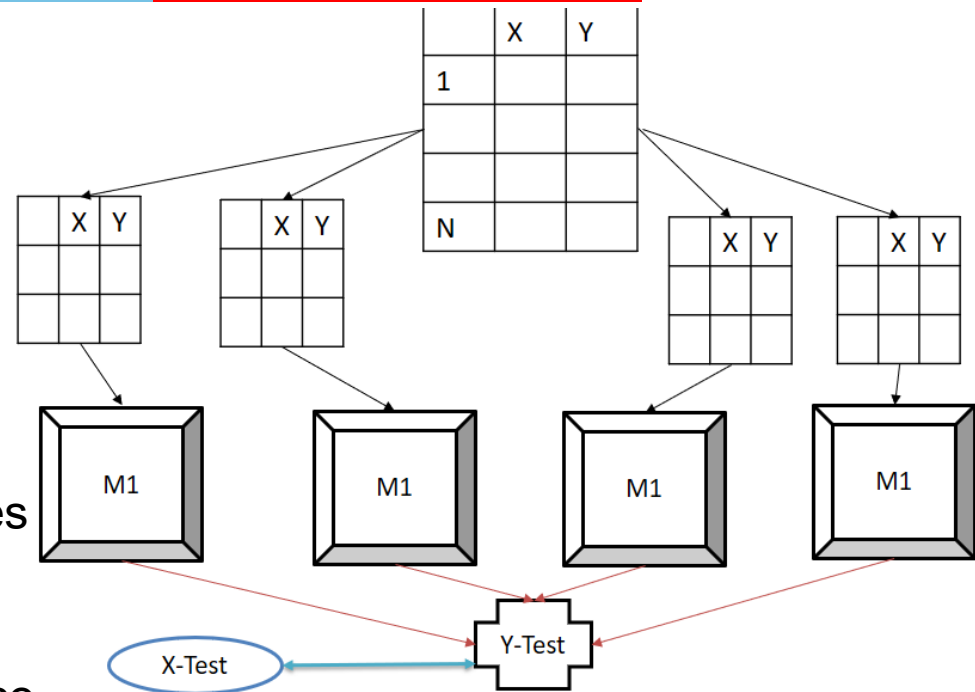
# Issue 1 : On the members ( Base Learners )

- Use Different Algorithms
  - Different algorithms make different assumptions
- Use Different Hyper parameters,
  - E.g. vary the structure of neural nets



# Issue 1 : On the members ( Base Learners )

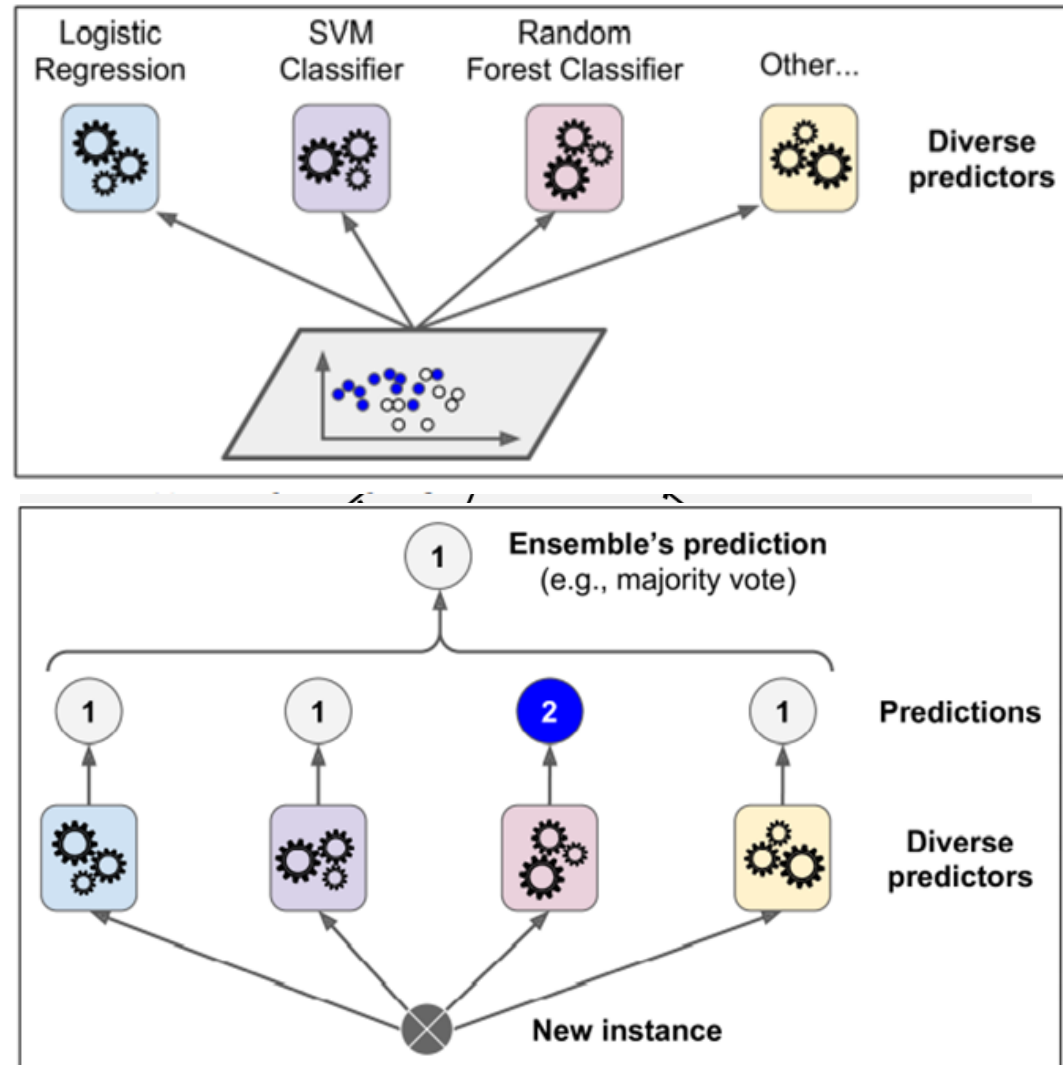
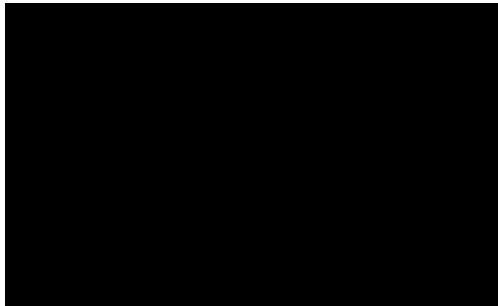
- Different input representations
  - Uttered words + video information of speakers clips
  - image + text annotations
- Different training sets
  - Draw different random samples of data
  - Partition data in the input space and have learners specialized in those spaces (mixture of experts)



## Issue -2 : Combining Results Base Learners

$$y = f(d_1, d_2, \dots, d_L | \Phi)$$

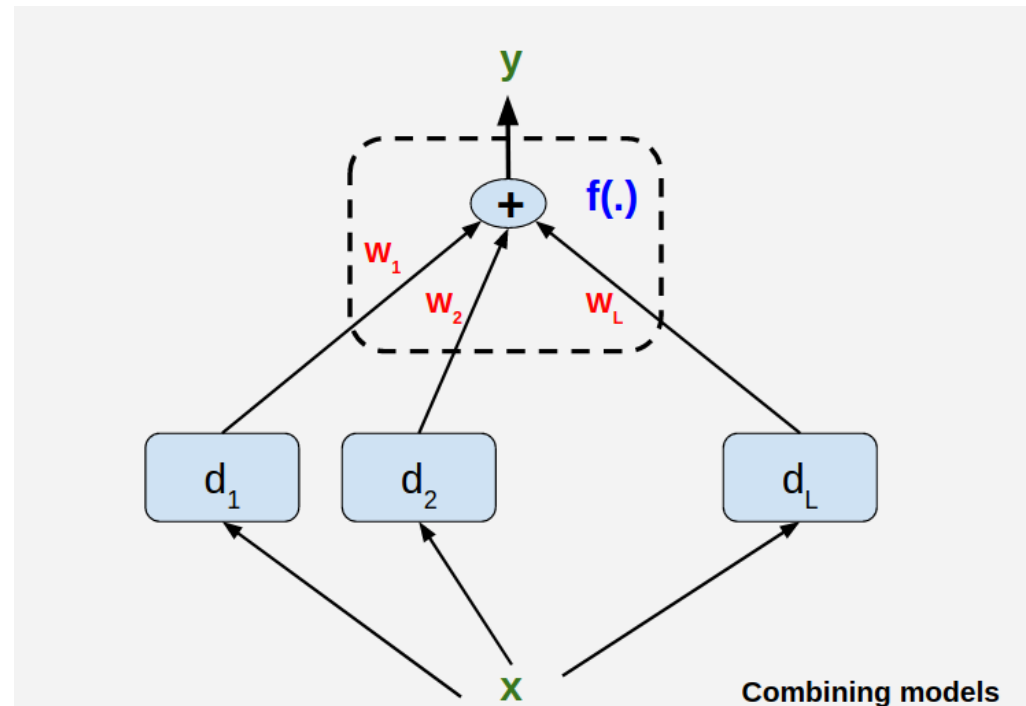
A Simple Combination Scheme:



## Issue -2 : Combining Results Base Learners

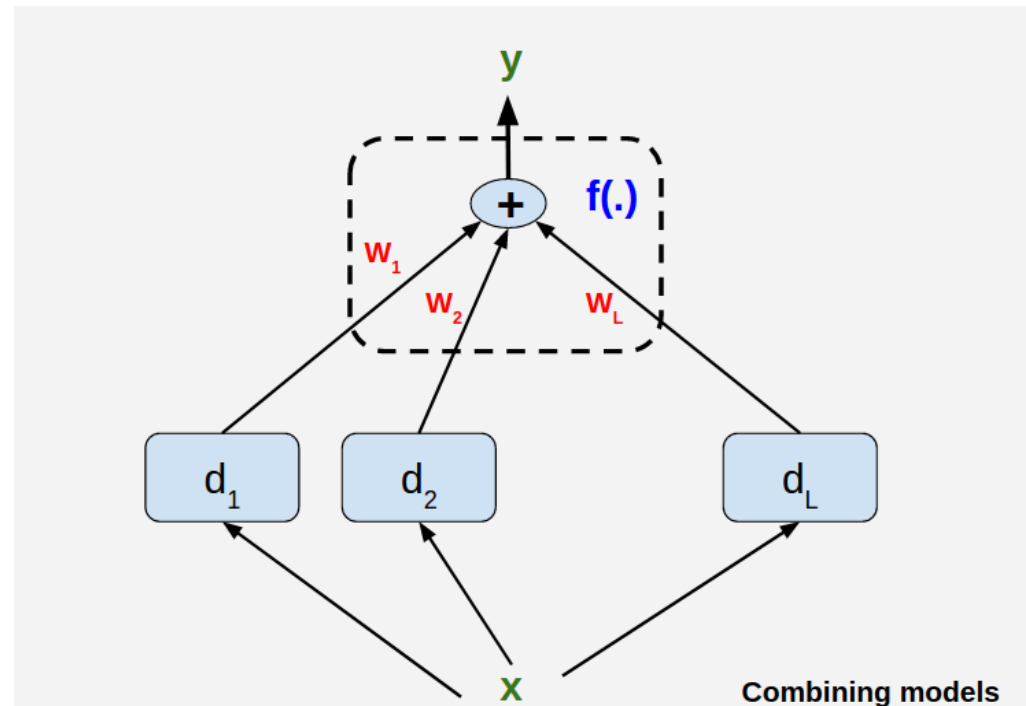
$$y = f(d_1, d_2, \dots, d_L | \Phi)$$

Rule	Fusion function $f(\cdot)$
Sum	$y_i = \frac{1}{L} \sum_{j=1}^L d_{ji}$
Weighted sum	$y_i = \sum_j w_j d_{ji}, w_j \geq 0, \sum_j w_j = 1$
Median	$y_i = \text{median}_j d_{ji}$
Minimum	$y_i = \min_j d_{ji}$
Maximum	$y_i = \max_j d_{ji}$
Product	$y_i = \prod_j d_{ji}$



## Issue -2 : Combining Results Base Learners

	$C_1$	$C_2$	$C_3$
$d_1$	0.2	0.5	0.3
$d_2$	0.0	0.6	0.4
$d_3$	0.4	0.4	0.2
<del>Sum</del> <b>Avg</b>	0.2	<b>0.5</b>	0.3
Median	0.2	<b>0.5</b>	0.4
Minimum	0.0	<b>0.4</b>	0.2
Maximum	0.4	<b>0.6</b>	0.4
Product	0.0	<b>0.12</b>	0.032



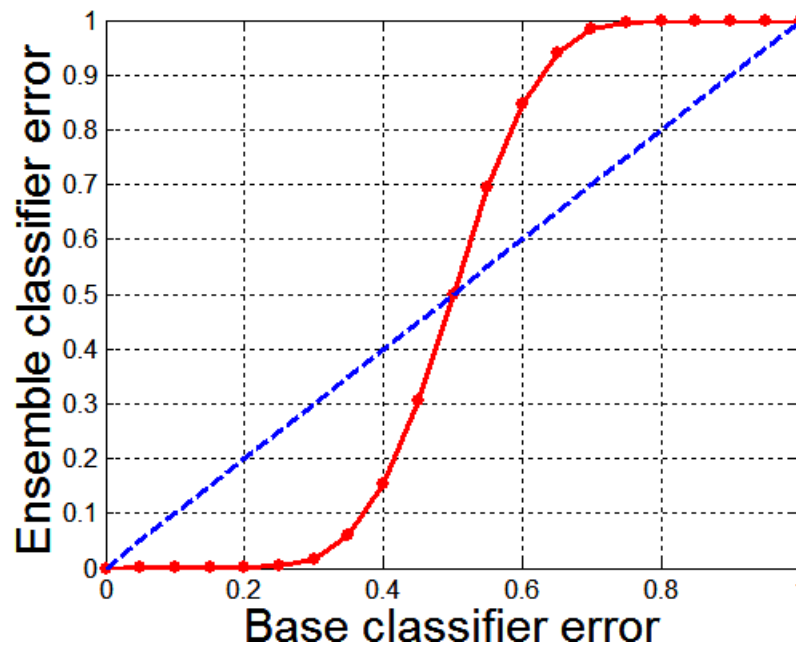
# When Ensemble Methods Work?

---

- Ensemble classifier performs better than the base classifiers when  $e$  is smaller than 0.5
- Necessary conditions for an ensemble classifier to perform better than a single classifier:
  - Base classifiers should be independent of each other
  - Base classifiers should do better than a classifier that performs random guessing

# Necessary Conditions for Ensemble Methods

- Ensemble Methods work better than a single base classifier if:
  - All base classifiers are independent of each other
  - All base classifiers perform better than random guessing (error rate  $< 0.5$  for binary classification)



Classification error for an ensemble of 25 base classifiers, assuming their errors are uncorrelated.



# Types of Ensemble Methods

---

- By manipulating training set
  - Example: bagging, boosting, random forests
- By manipulating input features
  - Example: random forests

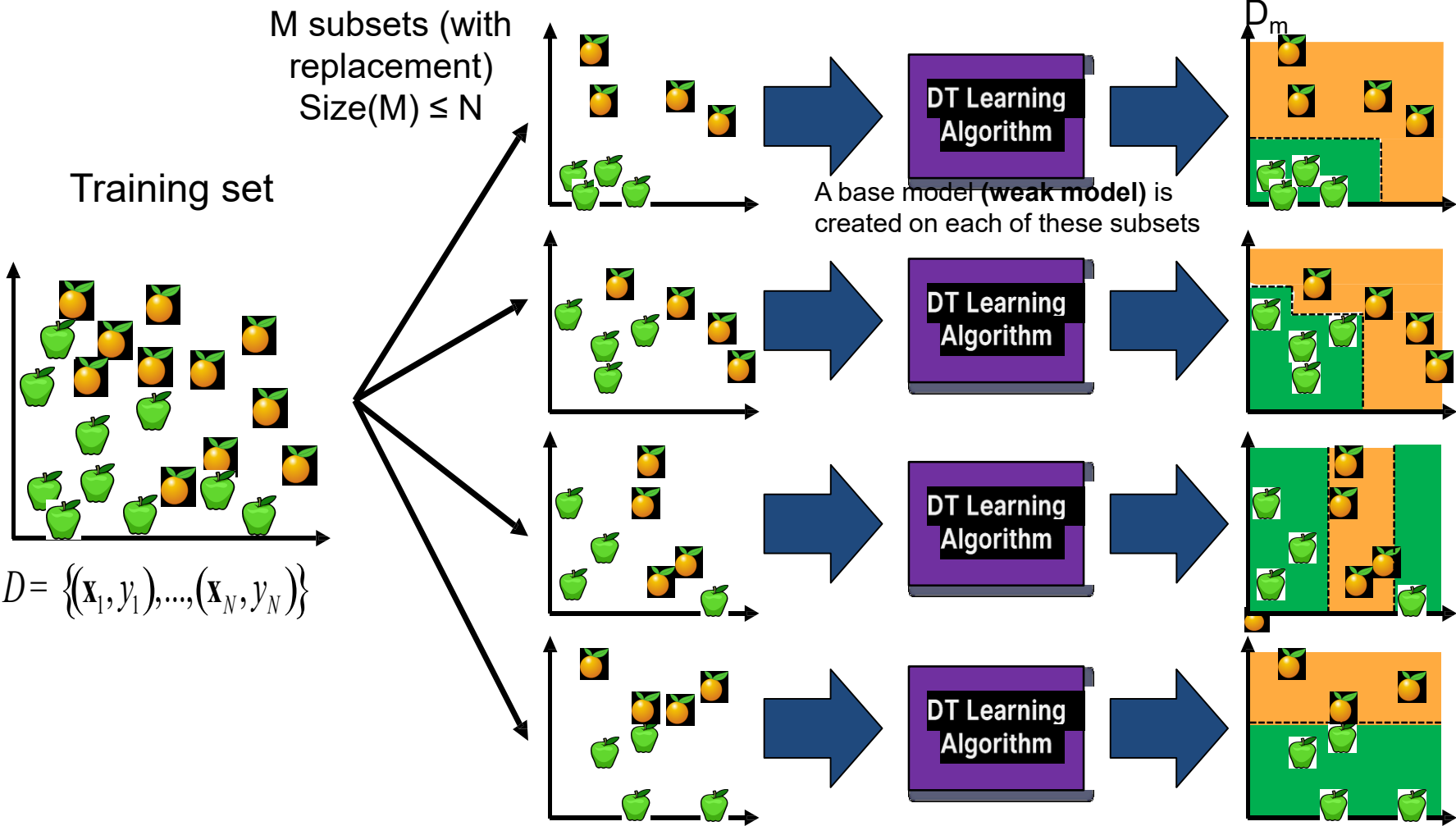
# Bagging

## Bootstrap Aggregating

# Bagging at training time

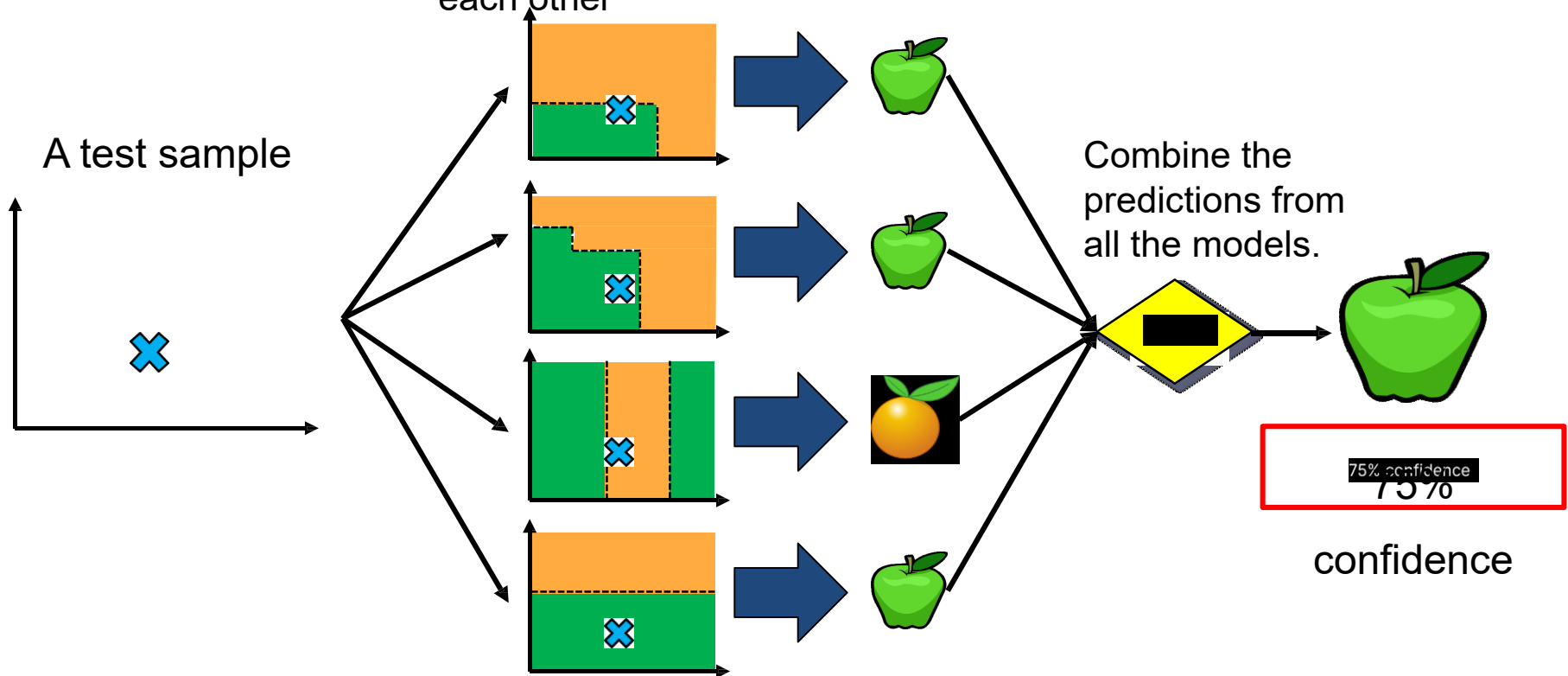
For  $m=1:M$ , Obtain bootstrap sample  $D_m$  from the training data  $D$

Build a model  $G_m(x)$  from bootstrap data  $D_m$



# Bagging at inference time

The models run in **parallel** and are independent of each other



# The Bagging Model

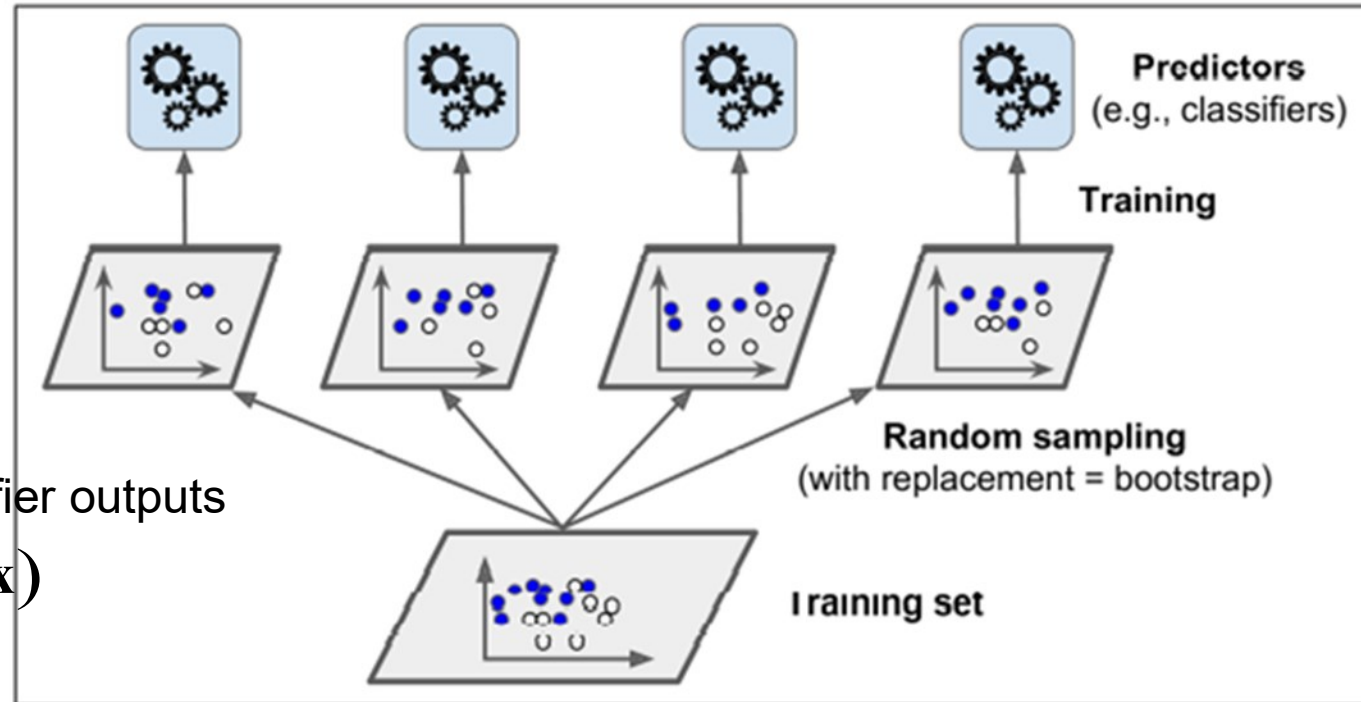
- Regression

$$\hat{y} = \frac{1}{M} \sum_{m=1}^M G_m(\mathbf{x})$$

- Classification:

- Vote over classifier outputs

$$G_1(\mathbf{x}), \dots, G_M(\mathbf{x})$$



# Bagging Algorithm

---

## Algorithm 4.5 Bagging algorithm.

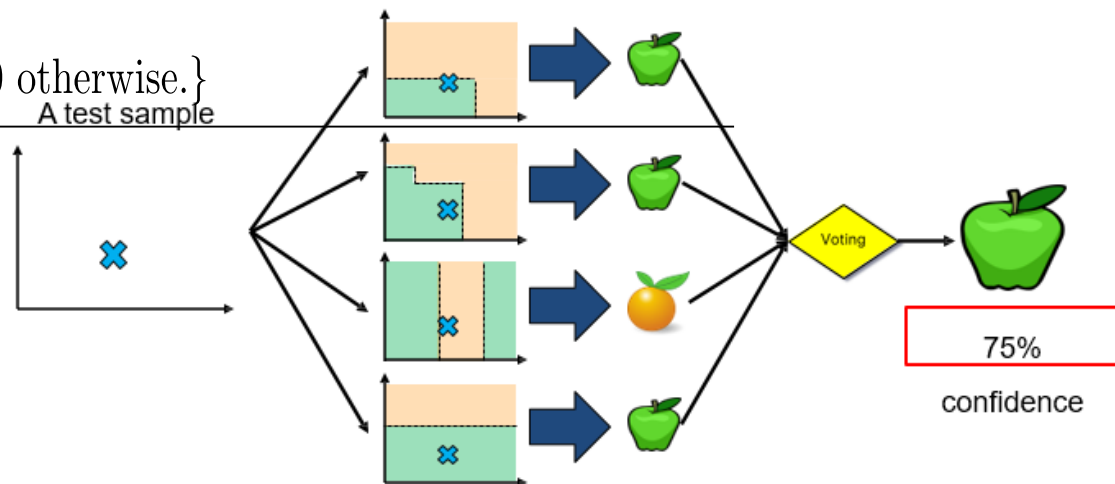
---

- 1: Let  $k$  be the number of bootstrap samples.
- 2: **for**  $i = 1$  to  $k$  **do**
- 3:   Create a bootstrap sample of size  $N$ ,  $D_i$ .
- 4:   Train a base classifier  $C_i$  on the bootstrap sample  $D_i$ .
- 5: **end for**
- 6:  $C^*(x) = \operatorname{argmax}_y \sum_i \delta(C_i(x) = y).$

$\{\delta(\cdot) = 1 \text{ if its argument is true and } 0 \text{ otherwise.}\}$

---

A test sample



# Bagging Example

Consider 1-dimensional data set:

**Original Data:**

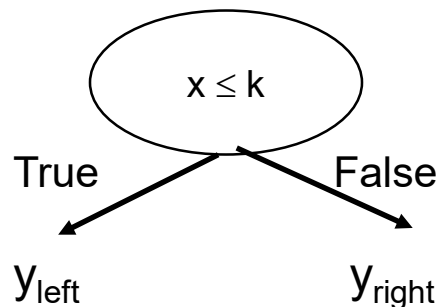
<b>x</b>	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
<b>y</b>	1	1	1	-1	-1	-1	-1	1	1	1

Classifier is a decision stump

Decision rule:  $x \leq k$  versus  $x > k$

$x < 0.35$  or  $x \geq 0.75$ .

Split point  $k$  is chosen based on entropy



Decision tree with one internal node (the root) which is immediately connected to the terminal nodes (its leaves). **Decision stump** makes a prediction based on the value of just a single input feature. Sometimes they are also called 1-rules

# Bagging Example

Bagging Round 1:

x	0.1	0.2	0.2	0.3	0.4	0.4	0.5	0.6	0.9	0.9
y	1	1	1	1	-1	-1	-1	-1	1	1

$x \leq 0.35 \rightarrow y = 1$

$x > 0.35 \rightarrow y = -1$

Bagging Round 2:

x	0.1	0.2	0.3	0.4	0.5	0.5	0.9	1	1	1
y	1	1	1	-1	-1	-1	1	1	1	1

$x \leq 0.7 \rightarrow y = 1$

$x > 0.7 \rightarrow y = -1$

Bagging Round 3:

x	0.1	0.2	0.3	0.4	0.4	0.5	0.7	0.7	0.8	0.9
y	1	1	1	-1	-1	-1	-1	-1	1	1

$x \leq 0.35 \rightarrow y = 1$

$x > 0.35 \rightarrow y = -1$

Bagging Round 4:

x	0.1	0.1	0.2	0.4	0.4	0.5	0.5	0.7	0.8	0.9
y	1	1	1	-1	-1	-1	-1	-1	1	1

$x \leq 0.3 \rightarrow y = 1$

$x > 0.3 \rightarrow y = -1$

Bagging Round 5:

x	0.1	0.1	0.2	0.5	0.6	0.6	0.6	1	1	1
y	1	1	1	-1	-1	-1	-1	1	1	1

$x \leq 0.35 \rightarrow y = 1$

$x > 0.35 \rightarrow y = -1$



# Bagging Example

Bagging Round 6:

x	0.2	0.4	0.5	0.6	0.7	0.7	0.7	0.8	0.9	1
y	1	-1	-1	-1	-1	-1	-1	1	1	1

$x \leq 0.75 \rightarrow y = -1$   
 $x > 0.75 \rightarrow y = 1$

Bagging Round 7:

x	0.1	0.4	0.4	0.6	0.7	0.8	0.9	0.9	0.9	1
y	1	-1	-1	-1	-1	1	1	1	1	1

$x \leq 0.75 \rightarrow y = -1$   
 $x > 0.75 \rightarrow y = 1$

Bagging Round 8:

x	0.1	0.2	0.5	0.5	0.5	0.7	0.7	0.8	0.9	1
y	1	1	-1	-1	-1	-1	-1	1	1	1

$x \leq 0.75 \rightarrow y = -1$   
 $x > 0.75 \rightarrow y = 1$

Bagging Round 9:

x	0.1	0.3	0.4	0.4	0.6	0.7	0.7	0.8	1	1
y	1	1	-1	-1	-1	-1	-1	1	1	1

$x \leq 0.75 \rightarrow y = -1$   
 $x > 0.75 \rightarrow y = 1$

Bagging Round 10:

x	0.1	0.1	0.1	0.1	0.3	0.3	0.8	0.8	0.9	0.9
y	1	1	1	1	1	1	1	1	1	1

$x \leq 0.05 \rightarrow y = 1$   
 $x > 0.05 \rightarrow y = 1$

# Bagging Example

Summary of Training sets:

Round	Split Point	Left Class	Right Class
1	0.35	1	-1
2	0.7	1	1
3	0.35	1	-1
4	0.3	1	-1
5	0.35	1	-1
6	0.75	-1	1
7	0.75	-1	1
8	0.75	-1	1
9	0.75	-1	1
10	0.05	1	1

# Bagging Example

- Assume test set is the same as the original data
- Use majority vote to determine class of ensemble classifier

**Original Data:**

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y	1	1	1	-1	-1	-1	-1	1	1	1

Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
1	1	1	1	-1	-1	-1	-1	-1	-1	-1
2	1	1	1	1	1	1	1	1	1	1
3	1	1	1	-1	-1	-1	-1	-1	-1	-1
4	1	1	1	-1	-1	-1	-1	-1	-1	-1
5	1	1	1	-1	-1	-1	-1	-1	-1	-1
6	-1	-1	-1	-1	-1	-1	-1	1	1	1
7	-1	-1	-1	-1	-1	-1	-1	1	1	1
8	-1	-1	-1	-1	-1	-1	-1	1	1	1
9	-1	-1	-1	-1	-1	-1	-1	1	1	1
10	1	1	1	1	1	1	1	1	1	1
Sum	2	2	2	-6	-6	-6	-6	2	2	2
Sign	1	1	1	-1	-1	-1	-1	1	1	1

Predicted  
Class

# Bagging – Effect on Bias

---

- If a base classifier is stable, i.e., robust to minor perturbations in the training set, then the error of the ensemble is primarily caused by bias in the base classifier.
- In this situation, bagging may not be able to improve the performance of the base classifiers significantly.
- It may even degrade the classifier's performance because the effective size of each training set is about 37% smaller than the original data.

# Random Forest

# Random Forest

- Ensemble method specifically designed for decision tree classifiers
- Random Forests grows many trees
  - Ensemble of unpruned decision trees
  - Each base classifier classifies a “new” vector of attributes from the original data
  - Final result on classifying a new instance: voting.
  - Forest chooses the classification result having the most votes (over all the trees in the forest)

Lower correlation across trees

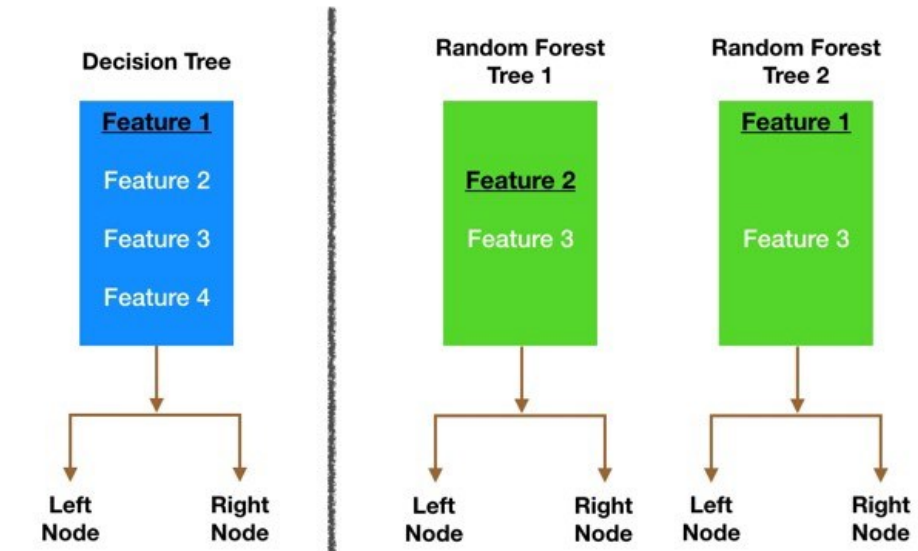
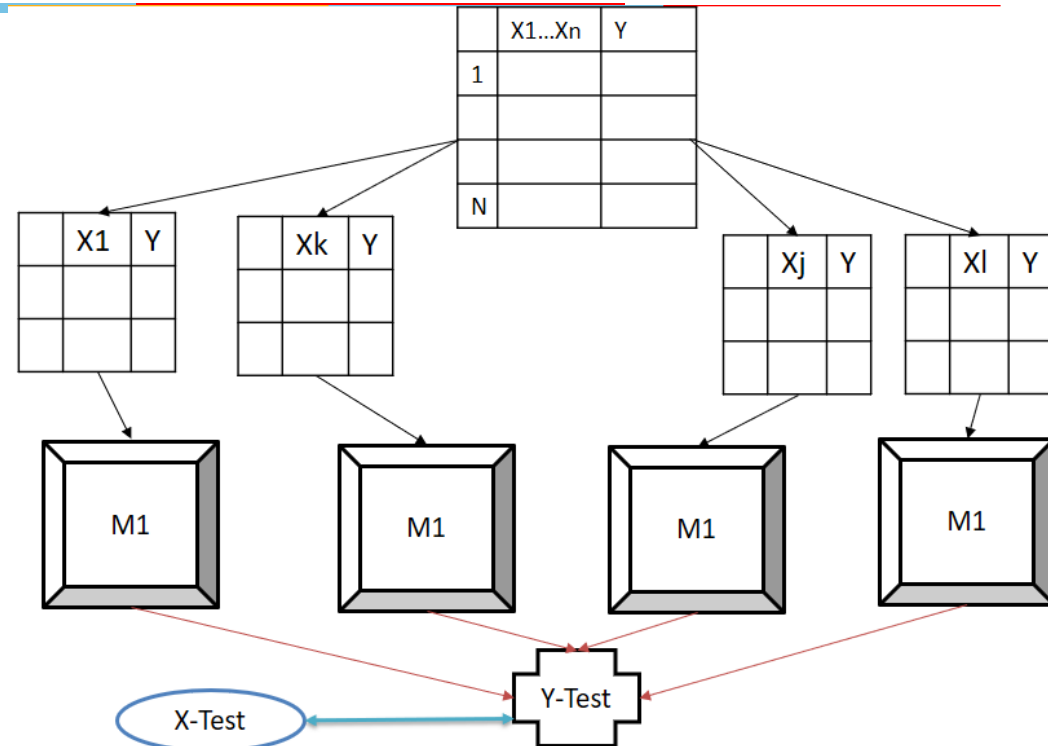


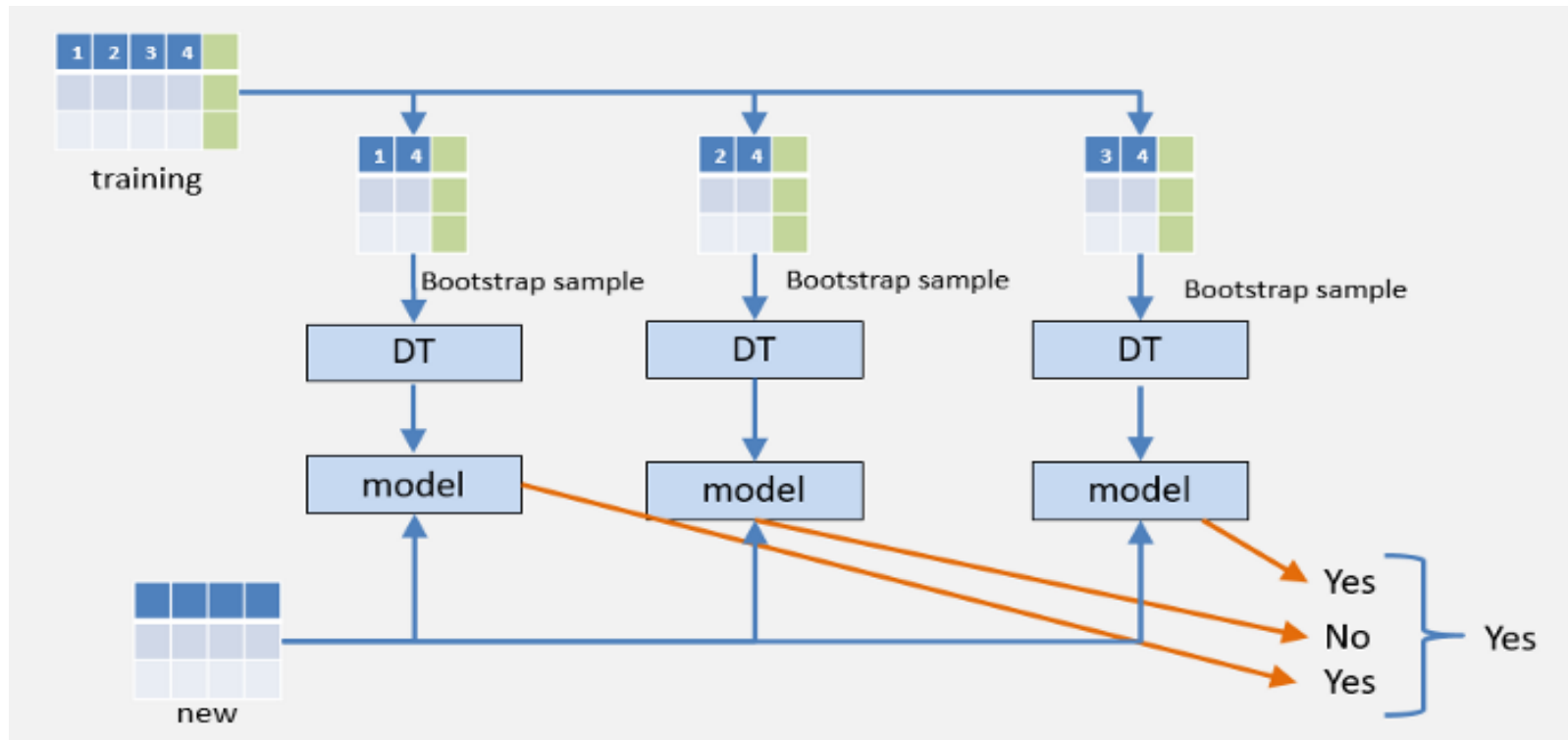
Image credit: <https://medium.com>

# Random Forest Algorithm

- Construct an ensemble of decision trees by manipulating **training set as well as features**
  - Use bootstrap sample to train every decision tree (similar to Bagging)
  - Use the following tree induction algorithm:
    - At every internal node of decision tree, **randomly sample  $p$  attributes** for selecting split criterion
    - Repeat this procedure until all leaves are pure (unpruned tree)



# Random Forest



- Trees that are trained on different sets of data (bagging)
- Trees use different features to make decisions.

Image credit: <https://medium.com>



# Random Forest

- **Random Forest** need features that have at least some predictive power.
- The trees of the forest and more importantly their predictions need to be uncorrelated (or at least have low correlations with each other).
- Algorithm can solve both type of problems i.e. classification and regression
- Power to handle large data set with higher dimensionality.
- It can handle thousands of input variables and **identify most significant variables so it is considered as one of the dimensionality reduction methods.**

# Random Forest

## Cheaper Feature Selection

```
>>> from sklearn.datasets import load_iris
>>> iris = load_iris()
>>> rnd_clf = RandomForestClassifier(n_estimators=500, n_jobs=-1)
>>> rnd_clf.fit(iris["data"], iris["target"])
>>> for name, score in zip(iris["feature_names"], rnd_clf.feature_importances_):
...     print(name, score)
...
sepal length (cm) 0.112492250999
sepal width (cm) 0.0231192882825
petal length (cm) 0.441030464364
petal width (cm) 0.423357996355
```

# Boosting

# Boosting

---

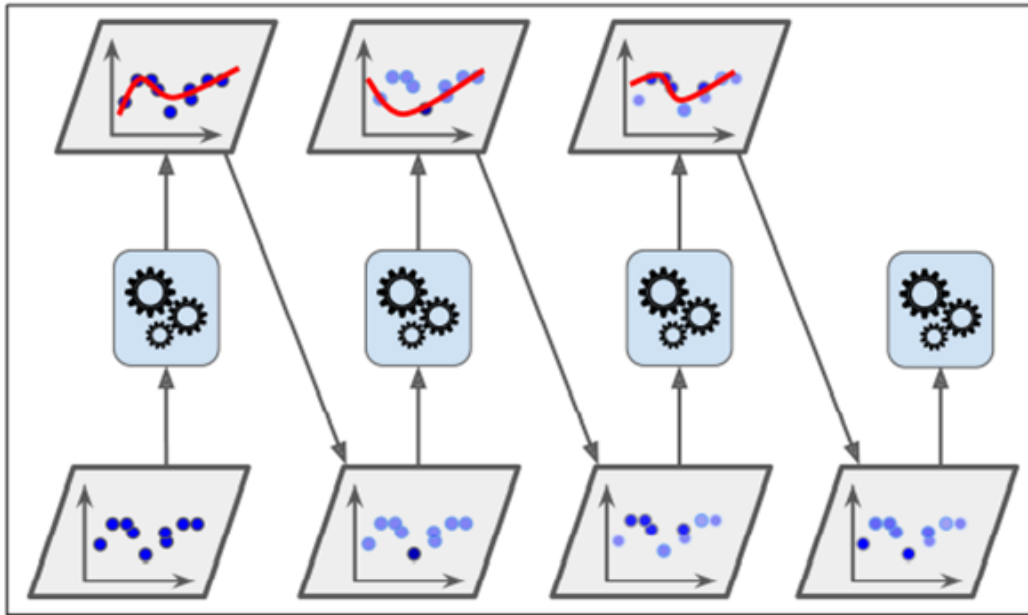
- What if a data point is incorrectly predicted by the first model, and then the next (probably all models), will combining the predictions provide better results? Such situations are taken care of by boosting.
- Boosting is a **sequential process**, where each subsequent model attempts to correct the errors of the previous model.

# Boosting

Train predictors sequentially, each trying to correct its predecessor

Scalability

## Ada Boost



An iterative procedure to adaptively change distribution of training data by focusing more on previously misclassified records

Initially, all  $N$  records are assigned equal weights (for being selected for training)

Unlike bagging, weights may change at the end of each boosting round

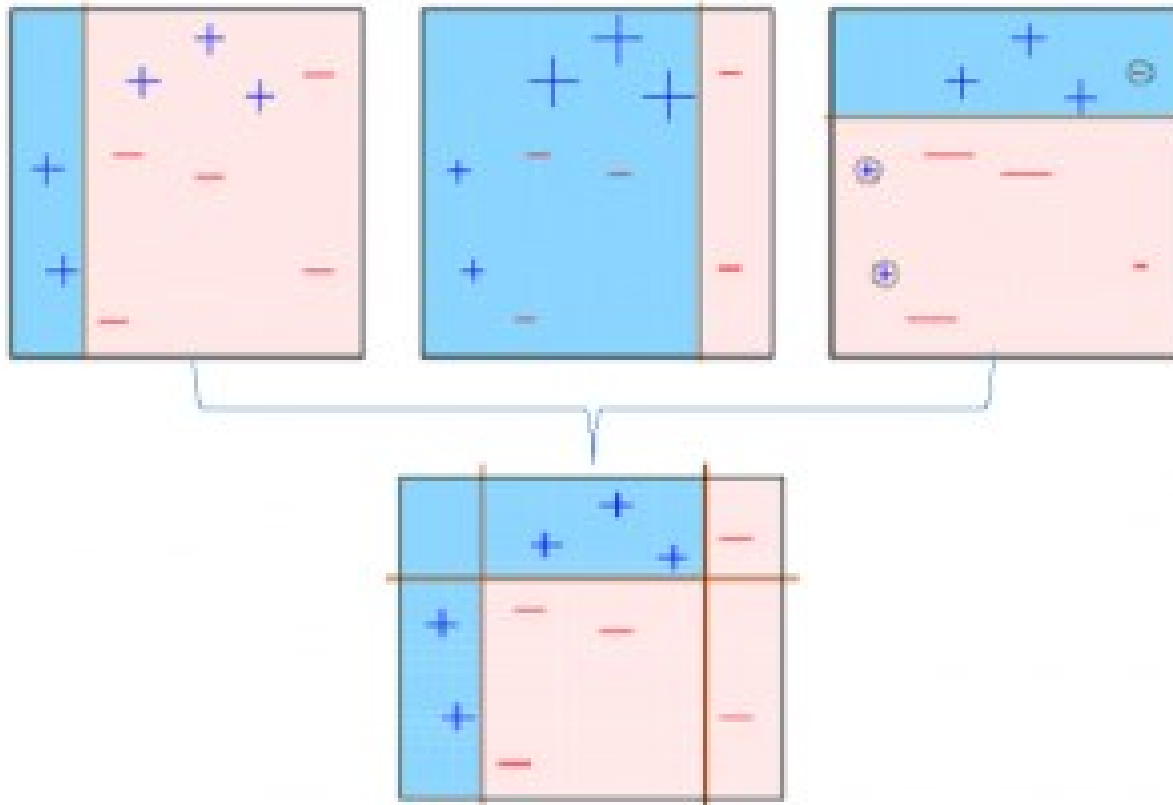
# Boosting

- Records that are wrongly classified will have their weights increased in the next round
- Records that are classified correctly will have their weights decreased in the next round

Original Data	1	2	3	4	5	6	7	8	9	10
Boosting (Round 1)	7	3	2	8	7	9	4	10	6	3
Boosting (Round 2)	5	4	9	4	2	5	1	7	4	2
Boosting (Round 3)	4	4	8	10	4	5	4	6	3	4

- Example 4 is hard to classify
- Its weight is increased, therefore it is more likely to be chosen again in subsequent rounds

# Boosting



- Multiple models are created, each correcting the errors of the previous model.
- The final model (strong learner) is the weighted mean of all the models (weak learners).
- Individual models would not perform well on the entire dataset, but they work well for some part of the dataset. Thus, each model actually boosts the performance of the ensemble.

# Ada Boost - Adaptive boosting Algorithm

For every instance  $j$  :  $w_j^{(0)} = 1/n$

For every predictor/model/estimator  $i$  :

Sample  $D_i$  instances using bagging

Fit a classifier

Compute the weighted error rate of the predictor on

all instances: 
$$\varepsilon_i = \frac{1}{N} \sum_{j=1}^N w_j \delta(C_i(x_j) \neq y_j)$$

Compute the predictor's weight

$$\alpha_i = \frac{1}{2} \ln \left( \frac{1 - \varepsilon_i}{\varepsilon_i} \right)$$

For every instance  $j$ :

Normalize all the instances weights

$$w_j^{(i+1)} = \frac{w_j^{(i)}}{Z_i} \begin{cases} \exp^{-\alpha_i} & \text{if } C_i(x_j) = y_j \\ \exp^{\alpha_i} & \text{if } C_i(x_j) \neq y_j \end{cases}$$

where  $Z_i$  is the normalization factor.

$i$  is the classifier/model and  $j$  is the instance



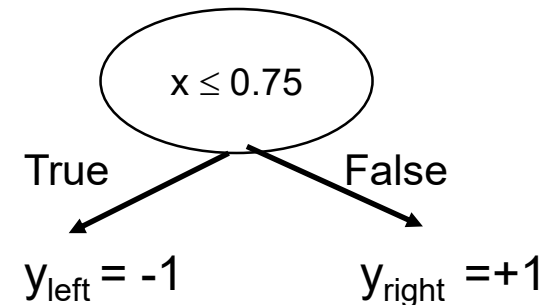
# Ada Boost - Adaptive boosting Algorithm

- Initially, all observations ( $n$ ) in the dataset are given equal weights ( $1/n$ ).
- A model is built on a subset of data. Usually, decision trees are used for modelling.
- Using this model, predictions are made on the whole dataset.
- Errors are calculated by comparing the predictions and actual values.
- Calculate the importance of the model
- Weights can be determined using the error value. For instance, higher the error more is the weight assigned to the observation.
- This process is repeated until the error function does not change, or the maximum limit of the number of estimators is reached.

Original Data :										
x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y	1	1	1	-1	-1	-1	-1	1	1	1
Weight	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
C1(Xj)	-1	-1	-1	-1	-1	-1	-1	1	1	1
$\epsilon_1 = 0.03 \quad \alpha_1 = 1.7380$										
Weight -C1	0.5686	0.5686	0.5686	0.0176	0.0176	0.0176	0.0176	0.0176	0.0176	0.0176
Normalized Wj	0.311	0.311	0.311	0.01	0.01	0.01	0.01	0.01	0.01	0.01

Error rate:  $N$  input samples, (Averaging over weights of training examples of misclassified points)

Round 1 Sample for Base Classifier C1										
x	0.1	0.4	0.5	0.6	0.6	0.7	0.7	0.7	0.8	1
y	1	-1	-1	-1	-1	-1	-1	-1	1	1



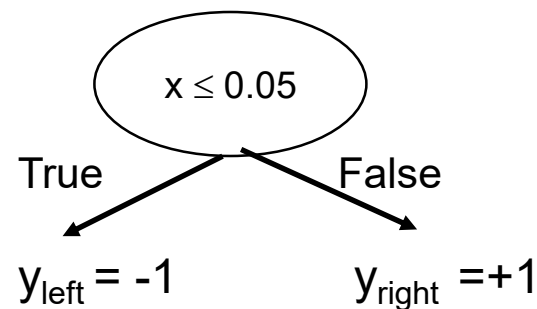
# Ada Boost - Adaptive boosting Algorithm



Original Data :										
x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y	1	1	1	-1	-1	-1	-1	1	1	1
Weight	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
y-pred	-1	-1	-1	-1	-1	-1	-1	1	1	1
$\epsilon_1 = 0.03 \quad \alpha_1 = 1.7380$										
Weight -C1	0.5686	0.5686	0.5686	0.0176	0.0176	0.0176	0.0176	0.0176	0.0176	0.0176
Normalized Wj	0.311	0.311	0.311	0.01	0.01	0.01	0.01	0.01	0.01	0.01

C2(xj)	1	1	1	1	1	1	1	1	1	1
$\epsilon_2 = 0.004 \quad \alpha_2 = 2.7587$										
Weight -C2	0.0197	0.0197	0.0197	0.1578	0.1578	0.1578	0.1578	0.0006	0.0006	0.0006
Normalized Wj	0.028	0.028	0.028	0.228	0.228	0.228	0.228	0.001	0.001	0.001

Round 2 Sample for Base Classifier C2										
x	0.1	0.1	0.2	0.2	0.2	0.2	0.3	0.3	0.3	0.3
y	1	1	1	1	1	1	1	1	1	1



# Ada Boost – Weight Update

Weight Update:

$$w_j^{(i+1)} = \frac{w_j^{(i)}}{Z_i} \begin{cases} \exp^{-\alpha_i} & \text{if } C_i(x_j) = y_j \\ \exp^{\alpha_i} & \text{if } C_i(x_j) \neq y_j \end{cases}$$

where  $Z_i$  is the normalization factor.

$i$  is the classifier/model and  $j$  is the instance

<- Eqn:5.88

- Reduce weight if correctly classified else increase
- If any intermediate rounds produce error rate higher than 50%, the weights are reverted back to  $1/n$  and the resampling procedure is repeated
- Prediction:  

$$C^*(x) = \arg \max_y \sum_{j=1}^T \alpha_j \delta(C_j(x) = y)$$

# Ada Boost Algorithm – Version 1

(with bootstrapping + average of error term)



---

## Algorithm 5.7 AdaBoost Algorithm

---

- 1:  $w = \{w_j = 1/n \mid j = 1, 2, \dots, n\}$ .    {Initialize the weights for all  $n$  instances.}
  - 2: Let  $k$  be the number of boosting rounds.
  - 3: for  $i = 1$  to  $k$  do
  - 4:    Create training set  $D_i$  by sampling (with replacement) from  $D$  according to  $w$ .
  - 5:    Train a base classifier  $C_i$  on  $D_i$ .
  - 6:    Apply  $C_i$  to all instances in the original training set,  $D$ .
  - 7:     $\epsilon_i = \frac{1}{n} [\sum_j w_j \delta(C_i(x_j) \neq y_j)]$     {Calculate the weighted error}
  - 8:    if  $\epsilon_i > 0.5$  then
  - 9:      $w = \{w_j = 1/n \mid j = 1, 2, \dots, n\}$ .    {Reset the weights for all  $n$  instances.}
  - 10:    Go back to Step 4.
  - 11:    end if
  - 12:     $\alpha_i = \frac{1}{2} \ln \frac{1-\epsilon_i}{\epsilon_i}$ .
  - 13:    Update the weight of each instance according to equation (5.88).
  - 14: end for
  - 15:  $C^*(x) = \arg \max_y \sum_{j=1}^T \alpha_j \delta(C_j(x) = y)$ .
-

# AdaBoost Example (version 1)

Original Data:

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y	1	1	1	-1	-1	-1	-1	1	1	1

Training sets for the first 3 boosting rounds:

Boosting Round 1:

x	0.1	0.4	0.5	0.6	0.6	0.7	0.7	0.7	0.8	1
y	1	-1	-1	-1	-1	-1	-1	-1	1	1

Boosting Round 2:

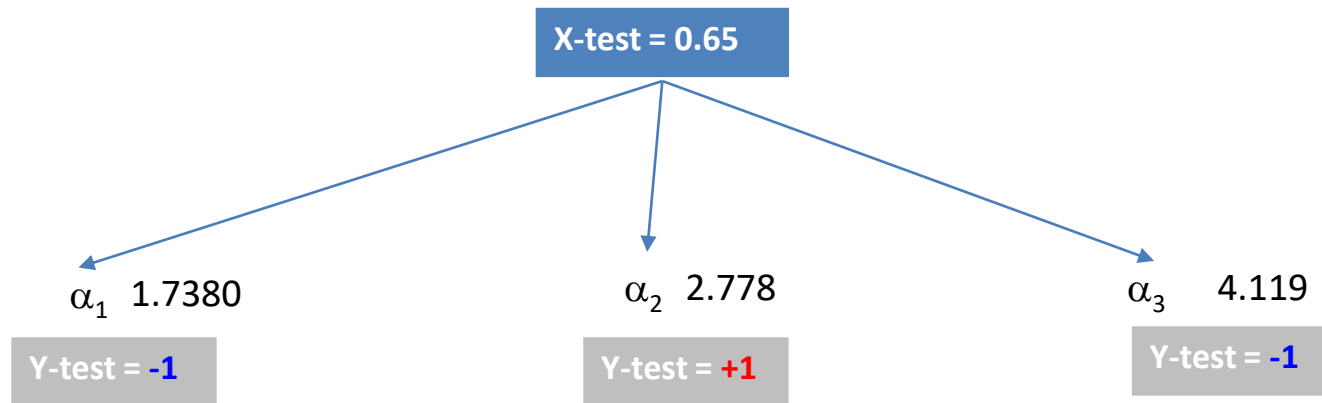
x	0.1	0.1	0.2	0.2	0.2	0.2	0.3	0.3	0.3	0.3
y	1	1	1	1	1	1	1	1	1	1

Boosting Round 3:

x	0.2	0.2	0.4	0.4	0.4	0.4	0.5	0.6	0.6	0.7
y	1	1	-1	-1	-1	-1	-1	-1	-1	-1

Summary:

Round	Split Point	Left Class	Right Class	alpha
1	0.75	-1	1	1.738
2	0.05	1	1	2.7784
3	0.3	1	-1	4.1195



$$C^*(x) = \arg \max_y \sum_{j=1}^T \alpha_j \delta(C_j(x) = y)$$

$$C^*(x\_test) = \arg \max_y \sum_{j=1}^3 \alpha_j \delta(C_j(x\_test) = y)$$

$$y = +1 \longrightarrow \sum_{j=1}^3 \alpha_j \delta(C_j(x\_test) = y) = 2.778$$

$$y = -1 \longrightarrow \sum_{j=1}^3 \alpha_j \delta(C_j(x\_test) = y) = 1.738 + 4.119 \approx 5.857$$

Answer :  $C^*(x\_test) = -1$

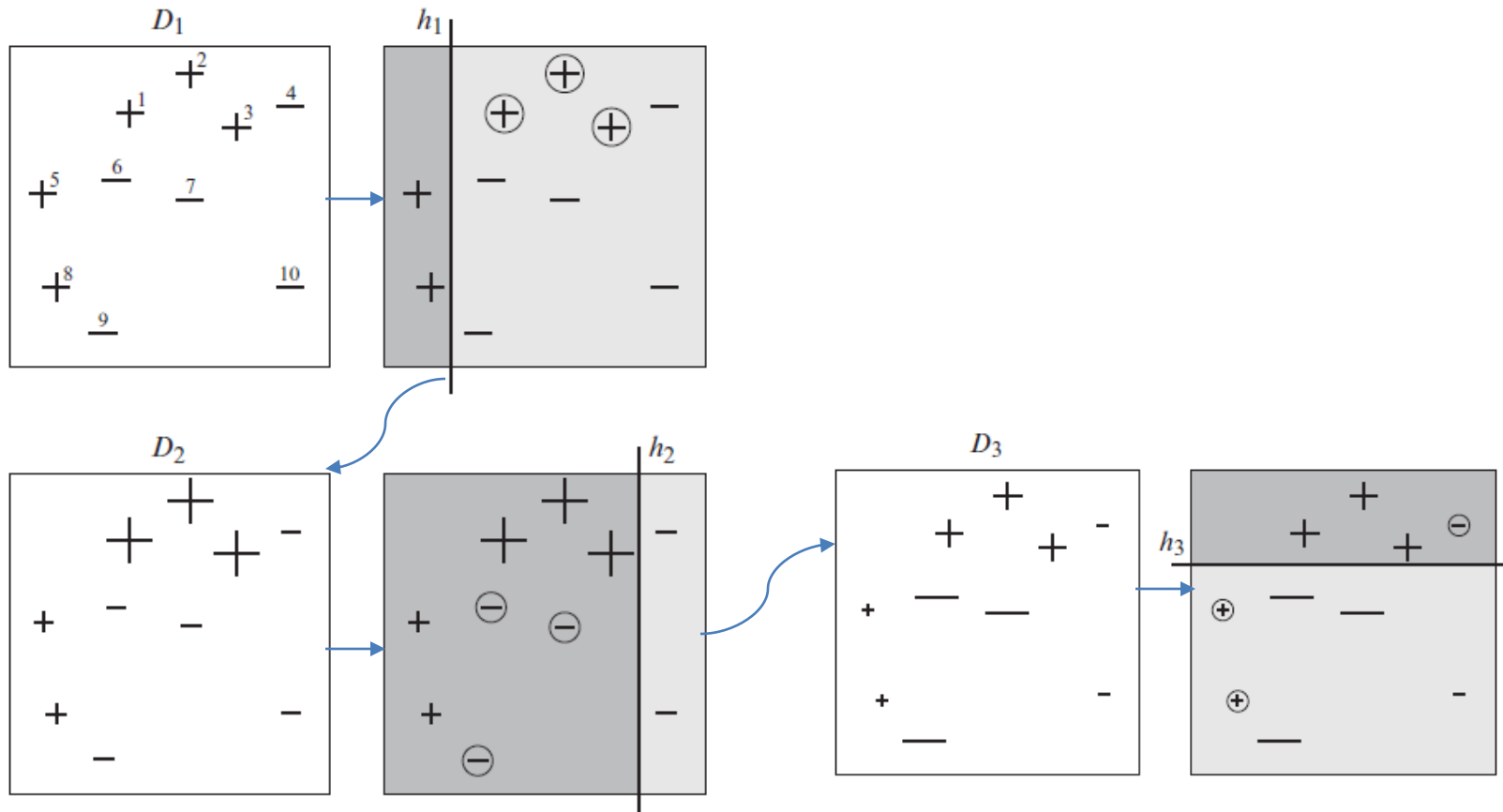
OR

$$H(\mathbf{x}) = \text{sign} \left( \sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$$

**AdaBoost error function takes into account the fact that only the sign of the final result is used, thus sum can be far larger than 1 without increasing error**

# AdaBoost Example (Visualization)

Source Credit : Schapire and Freund, 2012



# AdaBoost Example (version 2)

How do we combine the results now?

$$H = \text{sign} \left( 0.42 \begin{array}{|c|c|} \hline \text{shaded} & \text{unshaded} \\ \hline \end{array} + 0.65 \begin{array}{|c|c|} \hline \text{shaded} & \text{unshaded} \\ \hline \end{array} + 0.92 \begin{array}{|c|c|} \hline \text{shaded} & \text{unshaded} \\ \hline \end{array} \right)$$

$$= \begin{array}{|c|c|c|c|} \hline \text{shaded} & + & + & - \\ \hline + & - & - & - \\ \hline + & - & & - \\ \hline \end{array}$$

Source Credit : Schapire and Freund, 2012



# AdaBoost Example (version 2)

**Table 1.1**

The numerical calculations corresponding to the toy example in figure 1.1

	1	2	3	4	5	6	7	8	9	10	
$D_1(i)$	<u>0.10</u>	<u>0.10</u>	<u>0.10</u>	0.10	0.10	0.10	0.10	0.10	0.10	0.10	$\epsilon_1 = 0.30, \alpha_1 \approx 0.42$
$e^{-\alpha_1 y_i h_1(x_i)}$	1.53	1.53	1.53	0.65	0.65	0.65	0.65	0.65	0.65	0.65	
$D_1(i) e^{-\alpha_1 y_i h_1(x_i)}$	0.15	0.15	0.15	0.07	0.07	0.07	0.07	0.07	0.07	0.07	$Z_1 \approx 0.92$
$D_2(i)$	0.17	0.17	0.17	0.07	0.07	<u>0.07</u>	<u>0.07</u>	0.07	<u>0.07</u>	0.07	$\epsilon_2 \approx 0.21, \alpha_2 \approx 0.65$
$e^{-\alpha_2 y_i h_2(x_i)}$	0.52	0.52	0.52	0.52	0.52	1.91	1.91	0.52	1.91	0.52	
$D_2(i) e^{-\alpha_2 y_i h_2(x_i)}$	0.09	0.09	0.09	0.04	0.04	0.14	0.14	0.04	0.14	0.04	$Z_2 \approx 0.82$
$D_3(i)$	0.11	0.11	0.11	<u>0.05</u>	<u>0.05</u>	0.17	0.17	<u>0.05</u>	0.17	0.05	$\epsilon_3 \approx 0.14, \alpha_3 \approx 0.92$
$e^{-\alpha_3 y_i h_3(x_i)}$	0.40	0.40	0.40	2.52	2.52	0.40	0.40	2.52	0.40	0.40	
$D_3(i) e^{-\alpha_3 y_i h_3(x_i)}$	0.04	0.04	0.04	0.11	0.11	0.07	0.07	0.11	0.07	0.02	$Z_3 \approx 0.69$

Calculations are shown for the ten examples as numbered in the figure. Examples on which hypothesis  $h_t$  makes a mistake are indicated by underlined figures in the rows marked  $D_t$ .

# Ada Boost Algorithm – Version 2

(with bootstrapping + sum for error term)



**INPUT:** training data  $X, y = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ ,  
the number of iterations  $T$

1: Initialize a vector of  $n$  uniform weights  $\mathbf{w}_1 = [\frac{1}{n}, \dots, \frac{1}{n}]$

2: **for**  $t = 1, \dots, T$

3:   Train model  $h_t$  on  $X, y$  with instance weights  $\mathbf{w}_t$

4:   Compute the weighted training error rate of  $h_t$ :

$$\epsilon_t = \sum_{i: y_i \neq h_t(\mathbf{x}_i)} w_{t,i}$$

5:   Choose  $\beta_t = \frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right)$

6:   Update all instance weights:

$$w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i)) \quad \forall i = 1, \dots, n$$

7:   Normalize  $\mathbf{w}_{t+1}$  to be a distribution:

$$w_{t+1,i} = \frac{w_{t+1,i}}{\sum_{j=1}^n w_{t+1,j}} \quad \forall i = 1, \dots, n$$

8: **end for**

9: **Return** the hypothesis

$$H(\mathbf{x}) = \text{sign} \left( \sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$$

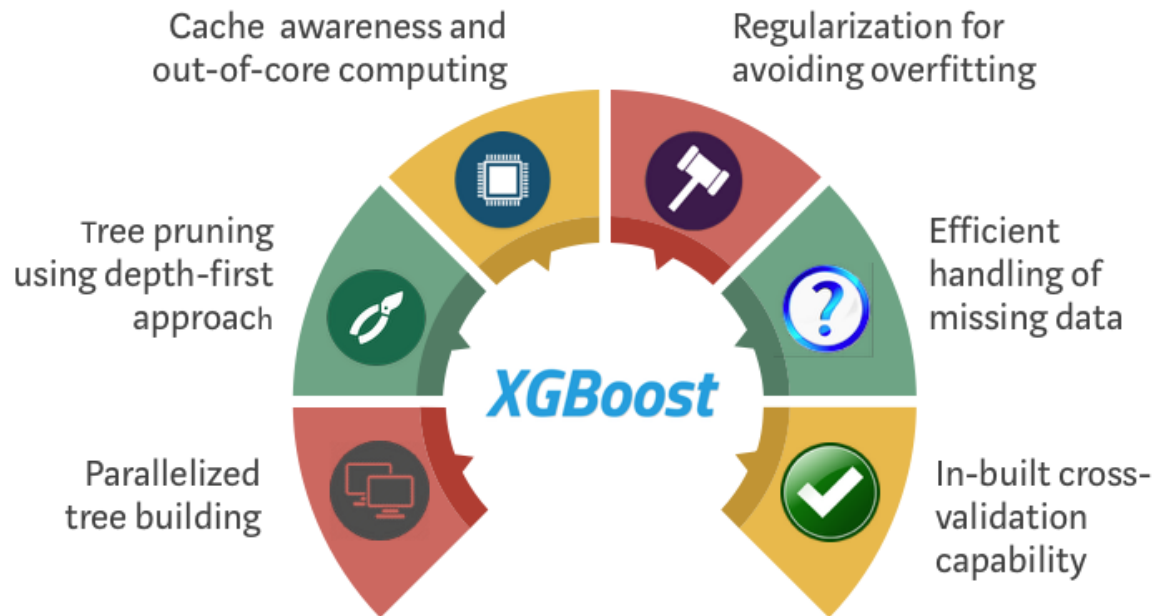
Member classifier with less error are given more weight in final ensemble hypothesis.  
Final prediction is a weighted combination of each members prediction

# Gradient Boosting

---

- The idea of gradient boosting originated in the observation by [Leo Breiman](#) that boosting can be interpreted as an optimization algorithm on a suitable cost function
- optimize a cost function over function space by iteratively choosing a function (weak hypothesis) that points in the negative gradient direction.
- predictor can be any machine learning algorithm like SVM, Logistic regression, KNN , Decision tree etc. But Decision tree version of gradient boosting is much popular
- In Gradient Boosting, "shortcomings" are identified by gradients.
- Recall that, in Adaboost, "shortcomings" are identified by high-weight data points.
- Both high-weight data points and gradients tell us how to improve our model.

# XGBoost



- XGBoost (Extreme Gradient Boosting) uses the gradient boosting (GBM) framework at its core.
- optimized distributed gradient boosting library designed to be highly **efficient**, **flexible** and **portable**

# Gradient Boosting - Idea

- You are given  $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ , and the task is to fit a model  $F(x)$  to minimize square loss

- There are some mistakes:

$$F(x_1) = 0.8, \text{ while } y_1 = 0.9,$$

$$F(x_2) = 1.4 \text{ while } y_2 = 1.3...$$

How can you improve this model?

- **Rules:**

- You are not allowed to remove anything from  $F$  or change any parameter in  $F$ .
- You can add an additional model (regression tree)  $h$  to  $F$ , so the new prediction will be  $F(x) + h(x)$ .

# Gradient Boosting

You wish to improve the model such that

- $F(x_1) + h(x_1) = y_1$
- $F(x_2) + h(x_2) = y_2 \dots$
- $F(x_n) + h(x_n) = y_n$

Or, equivalently

$$h(x_1) = y_1 - F(x_1)$$

$$h(x_2) = y_2 - F(x_2) \dots$$

$$h(x_n) = y_n - F(x_n)$$

Fit a regression tree  $h$  to data

$$(x_1, y_1 - F(x_1)), (x_2, y_2 - F(x_2)), \dots, (x_n, y_n - F(x_n))$$

- Simple solution:  $y_i - F(x_i)$  are called residuals
- These are the parts that existing model  $F$  cannot do well.
- The role of  $h$  is to compensate the shortcoming of existing model  $F$
- If the new model  $F + h$  is still not satisfactory, we can add another regression tree...



# Gradient boosting: Summary

- Gradient boosting involves three elements:
  - A loss function to be optimized
    - For example, regression may use a squared error and classification may use logarithmic loss.
  - A weak learner to make predictions E.g Decision tree/Decision stump
  - An additive model to add weak learners to minimize the loss function.
    - Trees are added one at a time, and existing trees in the model are not changed.
    - A gradient descent procedure is used to minimize the loss when adding trees.
    - Instead of parameters, we have weak learners
    - After calculating the loss, to perform the gradient descent procedure, we must add a tree to the model that reduces the loss (i.e. follow the gradient).

# Gradient boosting algorithm

---

let  $F_0$  be a “dummy” constant model

**for**  $m = 1, \dots, M$

**for** each pair  $(\mathbf{x}_i, y_i)$  in the training set

        compute the pseudo-residual  $R(y_i, F_{m-1}(\mathbf{x}_i)) = \text{negative gradient of the loss}$

    Train a regression sub-model  $h_m$  on the pseudo-residuals

    Add  $h_m$  to the ensemble:  $F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \rho \cdot h_m(\mathbf{x})$

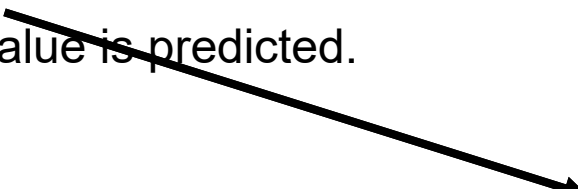
**return** the ensemble  $F_M$



# Gradient boosting: Example

Height	Age	Gender	Weight
5.4	28	Male	88
5.2	26	Female	76
5	28	Female	56
5.6	25	Male	73
6	25	Male	77
4	22	Female	57

$F_0$  be a “dummy” constant model  
Average value is predicted.



Height	Age	Gender	Actual Weight	Predicted weight 1
5.4	28	M	88	71.2
5.2	26	F	76	71.2
5	28	F	56	71.2
5.6	25	M	73	71.2
6	25	M	77	71.2
4	22	F	57	71.2

# Iteration 1

compute the pseudo-residual  $R(y_i, F_{m-1}(x_i)) =$   
negative gradient of the loss

$$y_i - F(x_i) = -\frac{\partial J}{\partial F(x_i)}$$

Height	Age	Gender	Weight	Predicted Weight 1	Pseudo Residuals 1
5.4	28	Male	88	71.2	88-71.2 = 16.8
5.2	26	Female	76	71.2	76-71.2 = 4.8
5	28	Female	56	71.2	56-71.2 = -15.2
5.6	25	Male	73	71.2	73-71.2 = 1.8
6	25	Male	77	71.2	77-71.2 = 5.8
4	22	Female	57	71.2	57-71.2 = -14.2

Residual:  $h_1(x) = y - F_0(x)$

A tree with maximum leaf nodes as 4(hyper parameter for DT) using Height, Age and Gender to predict the residuals(Error)



# Iteration 1

Combining the trees to make the new prediction:  $F_m(x) = F_{m-1}(x) + \rho \cdot h_m(x)$

**Learning rate : 0.1**

Height	Age	Gender	Weight	Predicted Weight 1	Pseudo Residuals 1	Predicted weight 2
5.4	28	Male	88	71.2	$88 - 71.2 = 16.8$	$71.2 + 0.1 * 16.8 = 72.9$
5.2	26	Female	76	71.2	$76 - 71.2 = 4.8$	$71.2 + 0.1 * (-5.2) = 70.7$
5	28	Female	56	71.2	$56 - 71.2 = -15.2$	$71.2 + 0.1 * (-5.2) = 70.7$
5.6	25	Male	73	71.2	$73 - 71.2 = 1.8$	$71.2 + 0.1 * 3.8 = 71.6$
6	25	Male	77	71.2	$77 - 71.2 = 5.8$	$71.2 + 0.1 * 3.8 = 71.6$
4	22	Female	57	71.2	$57 - 71.2 = -14.2$	$71.2 + 0.1 * (-14.2) = 69.8$

# Iteration 1

Residual:  $h_2(x) = y - F_1(x)$

Height	Age	Gender	Weight	Predicted Weight 1	Pseudo Residuals 1	Predicted weight 2	Pseudo Residuals2
5.4	28	Male	88	71.2	$88 - 71.2 = 16.8$	$71.2 + 0.1 * 16.8 = 72.9$	$88 - 72.9 = 15.1$
5.2	26	Female	76	71.2	$76 - 71.2 = 4.8$	$71.2 + 0.1 * (-5.2) = 70.7$	$76 - 70.7 = 5.3$
5	28	Female	56	71.2	$56 - 71.2 = -15.2$	$71.2 + 0.1 * (-5.2) = 70.7$	$56 - 70.7 = -14.7$
5.6	25	Male	73	71.2	$73 - 71.2 = 1.8$	$71.2 + 0.1 * 3.8 = 71.6$	$73 - 71.6 = 1.4$
6	25	Male	77	71.2	$77 - 71.2 = 5.8$	$71.2 + 0.1 * 3.8 = 71.6$	$77 - 71.6 = 5.4$
4	22	Female	57	71.2	$57 - 71.2 = -14.2$	$71.2 + 0.1 * (-14.2) = 69.8$	$57 - 69.8 = -12.8$



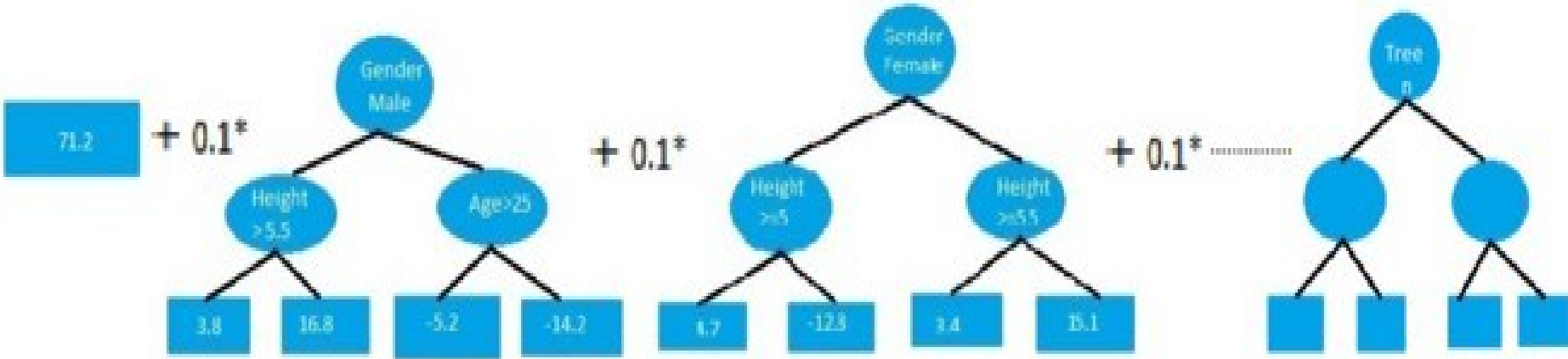
Example credit: <https://medium.com/nerd-for-tech/gradient-boost-for-regression-explained-6561eec192cb>

# Iteration 2

Height	Age	Gender	Weight	Predicted Weight 1	Pseudo Residuals 1
5.4	28	Male	88	71.2	$88 - 71.2 = 16.8$
5.2	26	Female	76	71.2	$76 - 71.2 = 4.8$
5	28	Female	56	71.2	$56 - 71.2 = -15.2$
5.6	25	Male	73	71.2	$73 - 71.2 = 1.8$
6	25	Male	77	71.2	$77 - 71.2 = 5.8$
4	22	Female	57	71.2	$57 - 71.2 = -14.2$

Predicted weight 2	Pseudo Residuals2
$71.2 + 0.1 * 16.8 = 72.9$	$88 - 72.9 = 15.1$
$71.2 + 0.1 * (-5.2) = 70.7$	$76 - 70.7 = 5.3$
$71.2 + 0.1 * (-5.2) = 70.7$	$56 - 70.7 = -14.7$
$71.2 + 0.1 * 3.8 = 71.6$	$73 - 71.6 = 1.4$
$71.2 + 0.1 * 3.8 = 71.6$	$77 - 71.6 = 5.4$
$71.2 + 0.1 * (-14.2) = 69.8$	$57 - 69.8 = -12.8$

Predicted Weight 3
$71.2 + 0.1 * 16.8 + 0.1 * 15.1 = 74.4$
$71.2 + 0.1 * (-5.2) + 0.1 * (-4.7) = 70.2$
$71.2 + 0.1 * (-5.2) + 0.1 * (-4.7) = 70.2$
$71.2 + 0.1 * 3.8 + 0.1 * 3.4 = 71.9$
$71.2 + 0.1 * 3.8 + 0.1 * 3.4 = 71.9$
$71.2 + 0.1 * (-14.2) + 0.1 * (-12.8) = 68.5$



Example credit: <https://medium.com/nerd-for-tech/gradient-boost-for-regression-explained-6561eec192cb>

# References

- Introduction to Data Mining, by Pang-Ning Tan, Michael Steinbach, Vipin Kumar
- Bishop - Pattern Recognition And Machine Learning - Springer 2006
- [https://en.wikipedia.org/wiki/Gradient\\_boosting#:~:text=Gradient%20boosting%20is%20a%20machine,which%20are%20typically%20decision%20trees](https://en.wikipedia.org/wiki/Gradient_boosting#:~:text=Gradient%20boosting%20is%20a%20machine,which%20are%20typically%20decision%20trees)
- <https://medium.com/nerd-for-tech/gradient-boost-for-regression-explained-6561eec192cb>

## Next Session Plan

- Unsupervised Learning