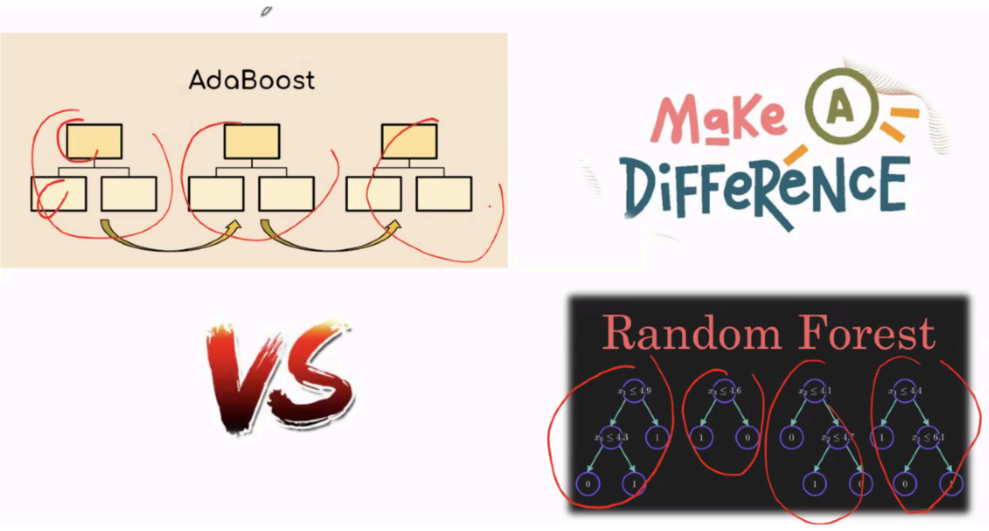
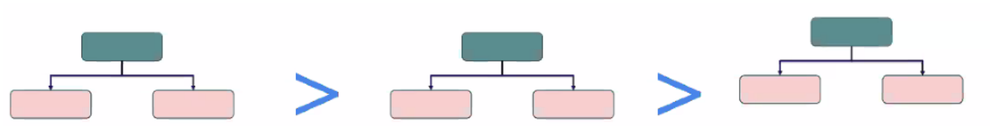
AIO2025 - Ada Boost

Hetegeneous - đồng nhất.
Boosting - ensemble modeling. Technique to build a string classifier from the number of weak classifiers.

Stump - Node or a Tree with only 2 leaves. Height of 1.



Stump with more error will contribute less in the final decision. More accurary -> More Weight to that Stump.

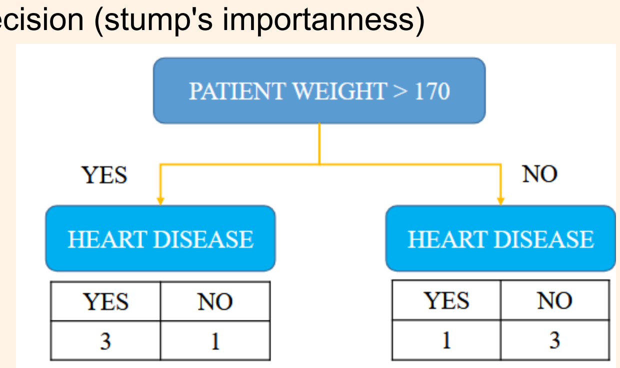


Difference between Random Forest and AdaBoost.

1. Weak learners is a *Stump*. AdaBoost combine a lot of stump, with each *stump* having 2 nodes and height of 1. (Overfitting and Underfitting are both weak learner)
2. Stumps have various *contribution* to the final result.
3. Each stump is created by considering *errors* of the previous stump.

Important of sample = Sample weight = 1 / number of samples = 1/8

- ? How much a stump contribute to the final decision (stump's importanness)

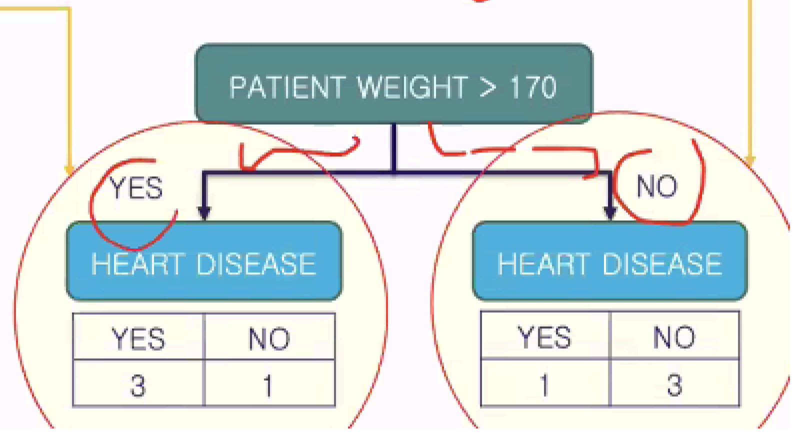


$$\text{Amount of Say} = \frac{1}{2}\log\left(\frac{1-\text{Total Error}}{\text{Total Error}}\right)$$

How many error does this stump make:

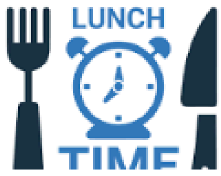
Chest Pain	Blocked Arteries	Patient Weight	Heart Diease	Sample Weight
Yes	Yes	205	Yes	1/8
No	Yes	180	Yes	1/8
Yes	No	210	Yes	1/8
Yes	Yes	167	Yes	1/8
No	Yes	156	No	1/8
No	Yes	125	No	1/8
Yes	No	168	No	1/8
Yes	Yes	172	No	1/8

sai



- Total error is equal to the sum of the weights of the incorrect classified
- *Amount of say* $= \frac{1}{2} * \log((1 - \frac{2}{8}) / (\frac{2}{8})) = 0.55$

Probability Vs. Odds



Your friend went fishing 10 times a month
- Caught a fish 4 times
- Failed to catch 6 times

What is the *probability* and *odds* of getting a Fish for lunch?

$$\text{Probability} = \frac{\text{Chance for catching fish}}{\text{Total chances}} = \frac{4}{10} = 0.4$$

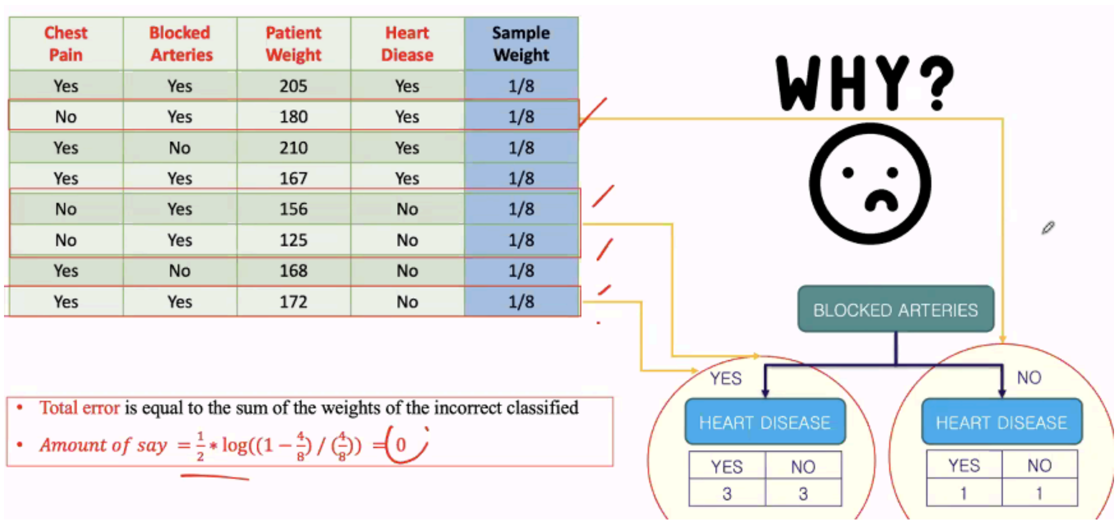
$$\text{Odds} = \frac{\text{Chance for catching fish}}{\text{Chance for not catching fish}} = \frac{4}{6} = 0.67$$

$$\text{Odds} = \frac{\text{Probability of catching fish}}{\text{Probability of not catching fish}} = \frac{4/10}{6/10} = 0.67$$

$$\text{Odds} = \frac{1 - \text{Probability of not catching fish}}{\text{Probability of not catching fish}} = \frac{4/10}{6/10} = 0.67$$

So we have $\frac{1-\text{total error}}{\text{total error}}$. The larger total_error is, the smaller $\frac{1-\text{total error}}{\text{total error}}$ is.

if stump's guesses are 50/50 -> "Amount of say" is zero.



Chest Pain	Blocked Arteries	Patient Weight	Heart Diease	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

- Total error is equal to the sum of the weights of the incorrect classified
- *Amount of say* $= \frac{1}{2} * \log((1 - \frac{4}{8}) / (\frac{4}{8})) = 0$

Để khắc phục nhượt điểm của stump kế tiếp ?

Known: weight cho các sample dự đoán sai được sử dụng để tính "Amount of Say" cho từng stump hiện tại.

Unknown: Tiếp theo, chúng ta cần làm thế nào để sử dụng thông tin các weight của sample dự đoán sai này để xây dựng stump tiếp và khắc phục các dự đoán sai này

Ideas to Improve Bootstrapped Dataset -> Create a new boostrap dataset.
Build a new dataset where wrong answer appear more often. Make errors (wrong) guesses appear more in the new dataset.

- ? How to add incorrect classified to the new dataset
- $ Increase the sample weights of samples that were incorrectly classified and decrease sample weights of samples that were correctly classified.

alpha - Amount of say
goals: increase probability of incorrect classified sample to be added to the new dataset. (Update Amount of say)
Solution 1: *Update weight correct/incorrect classified (correct decrease, incorrect increase).* -> create new dataset with weak dataset.

❖ Solution 1: Label {-1,1}

Increase the sample weights of samples that were incorrectly classified and decrease the sample weights of samples that were correctly classified. Label {-1, 1}

Chest Pain	Blocked Arteries	Patient Weight	Heart Diease
Yes	Yes	205	Yes
No	Yes	180	Yes
Yes	No	210	Yes
Yes	Yes	167	Yes
No	Yes	156	No
No	Yes	125	No
Yes	No	168	No
Yes	Yes	172	No

New Sample Weight = sample weight x $e^{amount\ of\ say}$

New sample weight = 1/8 * e^{-0.55} = 0.07

```python
def update_weights_formular1(w_i, alpha, y, y_pred):
    result = w_i * np.exp(-alpha * y * y_pred)
    w_norm = result / np.sum(result)
    return w_norm
```

correct: $e^{-(\text{amount of say})}$
incorrect: $e^{(\text{amount of say})}$

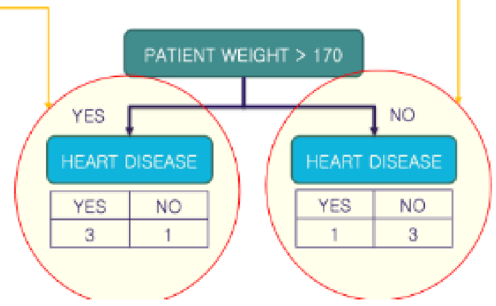
Solution 2: *Correct answer weight not change, incorrect -> increase weight*.

❖ Solution 2: Label {0,1}

Increase the sample weights of samples that were incorrectly classified and keep the sample weights of samples that were correctly classified. Label {0, 1}

Chest Pain	Blocked Arteries	Patient Weight	Heart Diease	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

- Total error is equal to the sum of the weights of the incorrect classified
- Amount of say = 1/2*log((1-2/8) / (2/8)) = 0.55



$$w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))]$$

New sample weight = 1/8 * e^{0.55*1} = 0.22

```python
def update_weights_formular2(w_i, alpha, y, y_pred):
    result = w_i * np.exp(alpha * (
        np.not_equal(y, y_pred)).astype(int))
    w_norm = result / np.sum(result)
    return w_norm
```

$G_m(x_i)$ Stump $! = y_i$ expected result. if == return 1, else 0. Only update weight for incorrect guesses.

Standardize by dividing to the total value
Create Stump of a column -> Continues to Calc Gini for each other columns -> new stump -> repeat. But repeat how many times ?

- ? However, the later stump only reduce error of its previous stump. So a pair of stump have independent contribution.
- $ Pros: have contribution of each stump. Each stump have something to say

Ada Boost loss functions

Algorithm 1 *AdaBoost (Freund & Schapire 1997)*

1. *Initialize the observation weights* $w_i = 1/n,\ i = 1, 2, \ldots, n.$
2. *For* $m = 1$ *to* M*:*
   (a) *Fit a classifier* $T^{(m)}(\boldsymbol{x})$ *to the training data using weights* w_i.
   (b) *Compute*
   $$err^{(m)} = \sum_{i=1}^{n} w_i \mathbb{I}\left(c_i \neq T^{(m)}(\boldsymbol{x}_i)\right) / \sum_{i=1}^{n} w_i.$$
   (c) *Compute*
   $$\alpha^{(m)} = \log \frac{1 - err^{(m)}}{err^{(m)}}.$$
   (d) *Set*
   $$w_i \leftarrow w_i \cdot \exp\left(\alpha^{(m)} \cdot \mathbb{I}\left(c_i \neq T^{(m)}(\boldsymbol{x}_i)\right)\right),\ i = 1, 2, \ldots, n.$$
   (e) *Re-normalize* w_i.
3. *Output*
$$C(\boldsymbol{x}) = \arg\max_k \sum_{m=1}^{M} \alpha^{(m)} \cdot \mathbb{I}(T^{(m)}(\boldsymbol{x}) = k).$$

Ada Boost for multi-class exponential loss function (just 1 changed)
Phân loại đa lớp:

- Tính tổng "tiếng nói" của từng lớp, lớp nào có tổng "tiếng nói" lớn nhất thì chọn.
- Thì K tương đương với số lớp á. Thay vì binary 0 1 thì trong multi class thì K nó nhiều trường hợp hơn thôi á

Vì sao lại `+ log(K - 1)`

Algorithm 2 *SAMME*

1. *Initialize the observation weights* $w_i = 1/n,\ i = 1, 2, \dots, n$.
2. *For* $m = 1$ *to* M*:*
   (a) *Fit a classifier* $T^{(m)}(\boldsymbol{x})$ *to the training data using weights* w_i.
   (b) *Compute*
   $$err^{(m)} = \sum_{i=1}^{n} w_i \mathbb{I}\left(c_i \neq T^{(m)}(\boldsymbol{x}_i)\right) / \sum_{i=1}^{n} w_i.$$
   (c) *Compute*
   $$\alpha^{(m)} = \log \frac{1 - err^{(m)}}{err^{(m)}} + \log(K - 1). \quad (1)$$
   (d) *Set*
   $$w_i \leftarrow w_i \cdot \exp\left(\alpha^{(m)} \cdot \mathbb{I}\left(c_i \neq T^{(m)}(\boldsymbol{x}_i)\right)\right),\ i = 1, \dots, n.$$
   (e) *Re-normalize* w_i.
3. *Output*
$$C(\boldsymbol{x}) = \arg\max_k \sum_{m=1}^{M} \alpha^{(m)} \cdot \mathbb{I}(T^{(m)}(\boldsymbol{x}) = k).$$

Note: công thức gốc lấy từ lý thuyết giả thiết thống kê (hypothesis)

Weight dự đoán sai = total weight - weight tự đoán đúng, và ngược lại.

lúc suy ra => Log 2 vế

`np.sign` - source

$$\text{sign(x)} \begin{cases} -1 & \text{if } x < 0 \\ 0 & \text{if } x == 0 \\ 1 & \text{if } x > 0 \end{cases}$$