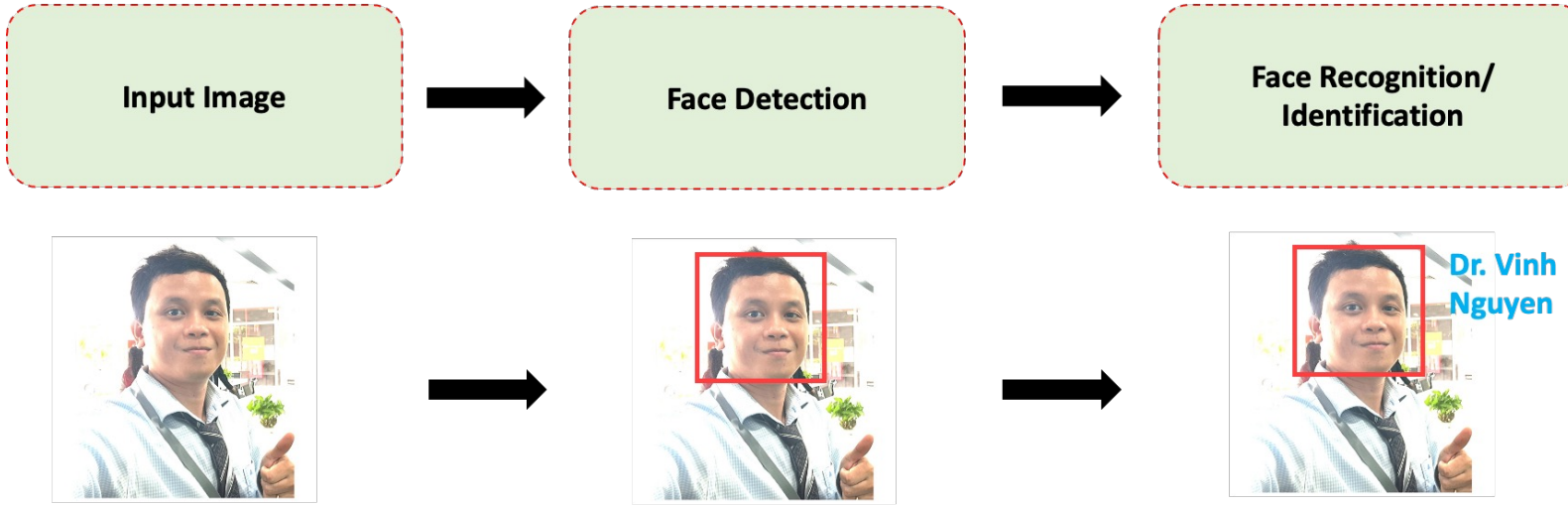
AI VIETNAM
All-in-One Course

Face Detection and Identification

(*Computer Vision Foundation*)



Vinh Dinh Nguyen
PhD in Computer Science

Year 2023

Outline

- Applications of Face Detection
- Face Detection in Computer Vision
- Haar Feature
- Integral Image
- Adaboost For Face Detection
- Cascade Classifier
- Face Recognition

Outline

- Applications of Face Detection
- Face Detection in Computer Vision
- Haar Feature
- Integral Image
- Adaboost For Face Detection
- Cascade Classifier
- Face Recognition

What is Face Detection

Locate human faces in images

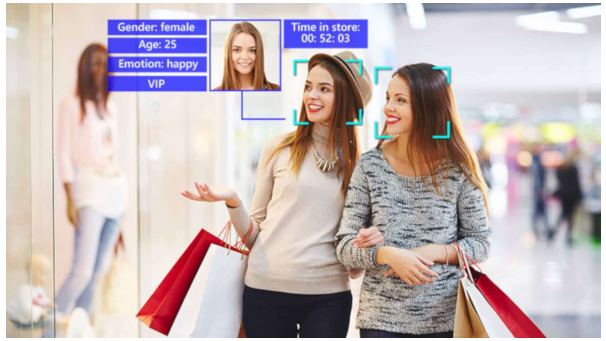
1. Application of face detection
2. Haar feature for face detection
3. Intergral Image
4. Cascade Classifier

Applications of Face Detection

Security Use Case of Face Recognition

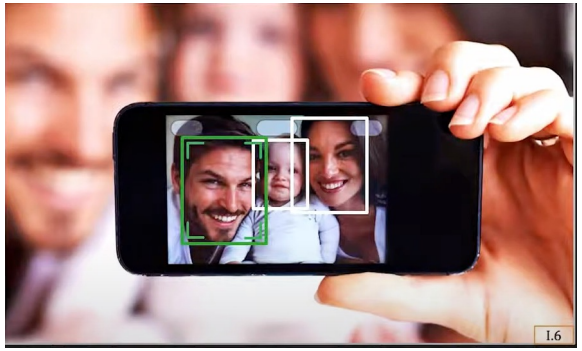
A Face Attendance System

Healthcare Application



Face Recognition in Retail

Face Recognition in Finance



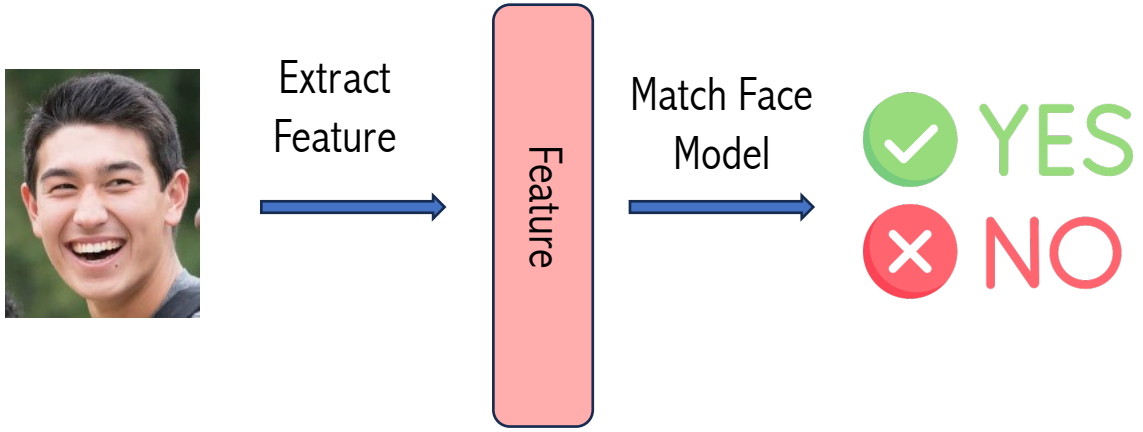
Automatic Camera Settings

Outline

- Applications of Face Detection
- Face Detection in Computer Vision
- Haar Feature
- Integral Image
- Adaboost For Face Detection
- Cascade Classifier
- Face Recognition

Face Detection in Computer Vision

Slides windows of differents sizes across image.
At each location match window to face model



Features:
How to extract feature?
Which features represent face well?

Classifier:
How to build a model and classify features as face or not?

What are good features?

Interest Point (Edge, Corners, SIFT,..)

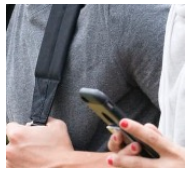
- Extreme Fast to Compute
- Millions of windows in an image

Facial Components (Templates)?

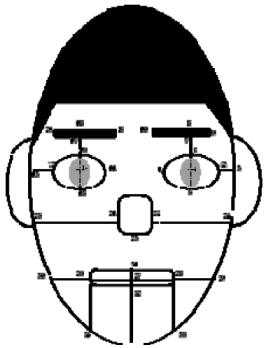
The key aspect in face recognition is detecting relevant features in human face like eyes, eyebrows, nose, lips. So how do we detect these features in real time/in an image ?

Discriminative Face / None Face

$\neq$

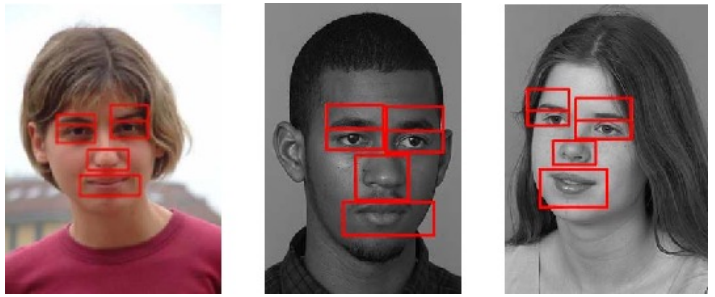


Methods for Face Detection



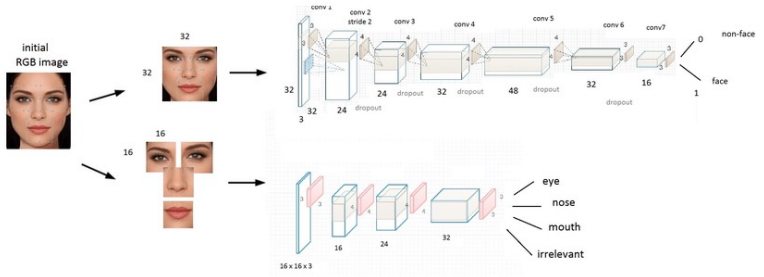
Knowledge Based

- Rule based (Ex: X must have eyes, x must have a nose)
- Too many rules and variables with this method



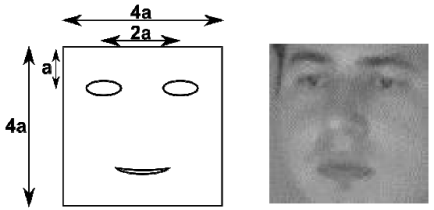
Feature Based

Locate and extract structural features in the face
Find a differential between facial and non facial regions in an image



Appearance Based

Learn the characteristics of a face. Example: CNN's
Accuracy depends on training data (which can be scarce)



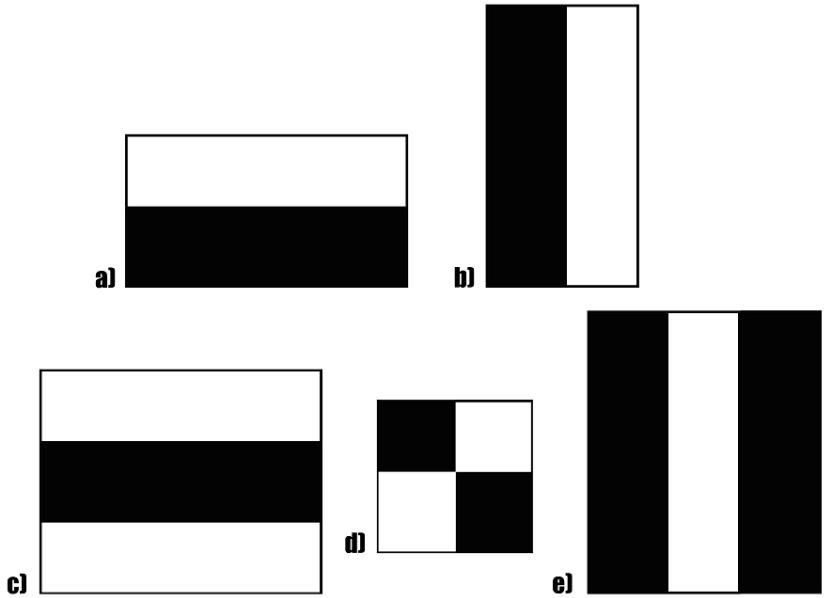
Template

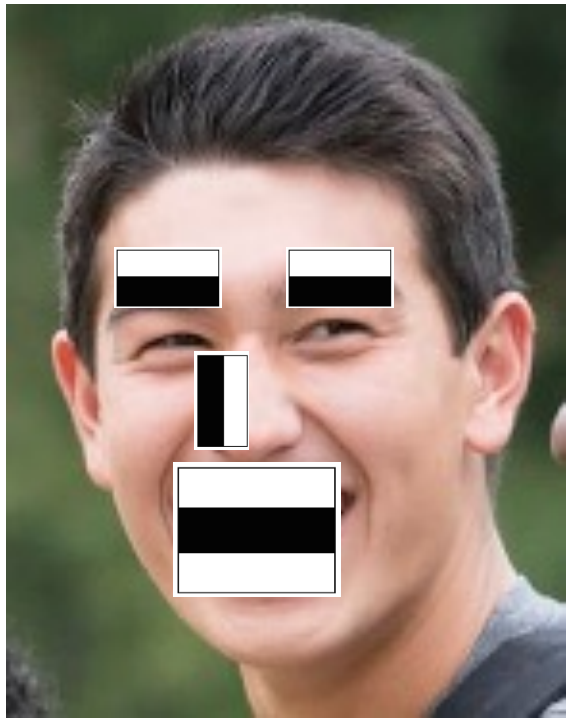
Using predefined templates for edge detection. Quick and easy
A trade off for speed over accuracy

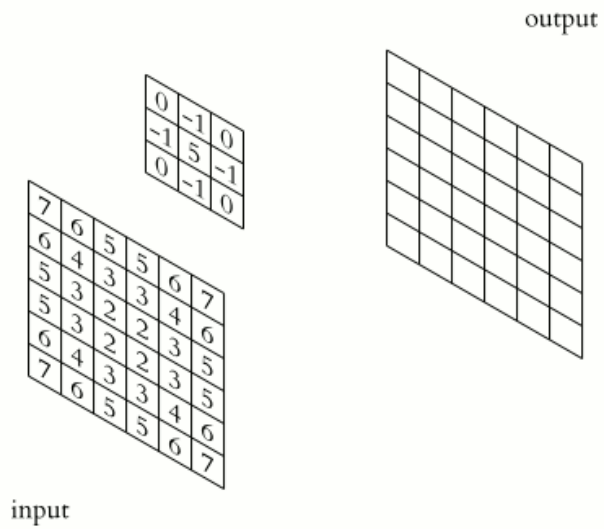
Outline

- Applications of Face Detection
- Face Detection in Computer Vision
- Haar Feature
- Integral Image
- Adaboost For Face Detection
- Cascade Classifier
- Face Recognition

Haar Feature



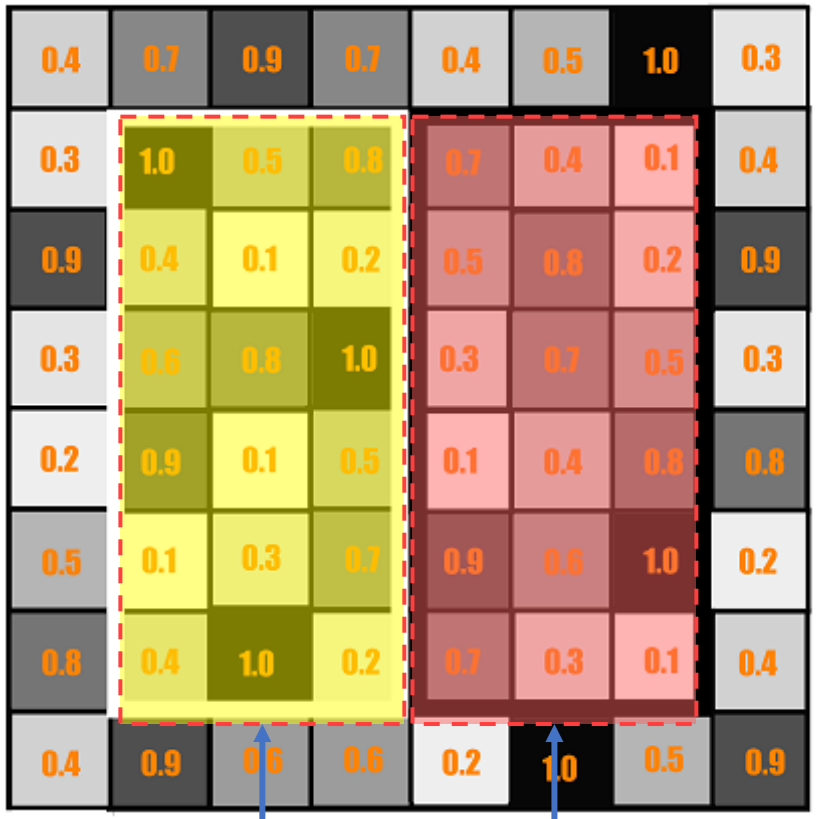


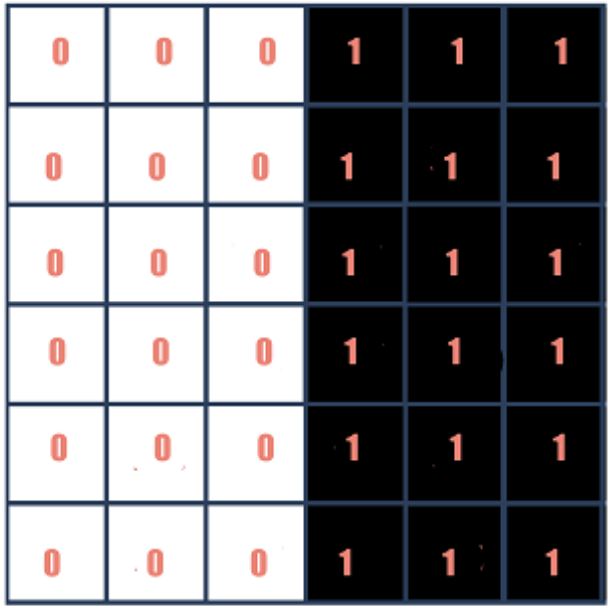


Viola-Jones uses 24*24 as base window size and calculates the above features all over the image shifting by 1 PX (Template + Feature).
There are over 160,000 possible feature combinations that can fit into a 24x24 pixel image, and over 250,000 for a 28x28 image

P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, Kauai, HI, USA, 2001, pp. I-I, doi: 10.1109/CVPR.2001.990517.

Haar Feature





=

SUM OF THE DARK PIXELS/NUMBER OF DARK PIXELS - SUM OF THE LIGHT PIXELS/NUMBER OF THE LIGHT PIXELS

(0.7 + 0.4 + 0.1 + 0.5 + 0.8 + 0.2 + 0.3 + 0.7 + 0.5 + 0.1 + 0.4 + 0.8 + 0.9 + 0.6 + 1.0 + 0.7 + 0.3 + 0.1)/18

-

(1.0 + 0.5 + 0.8 + 0.4 + 0.1 + 0.2 + 0.6 + 0.8 + 1.0 + 0.9 + 0.1 + 0.5 + 0.1 + 0.3 + 0.7 +0.4 + 1.0 + 0.2)/18

0.51-0.53 = -0.02

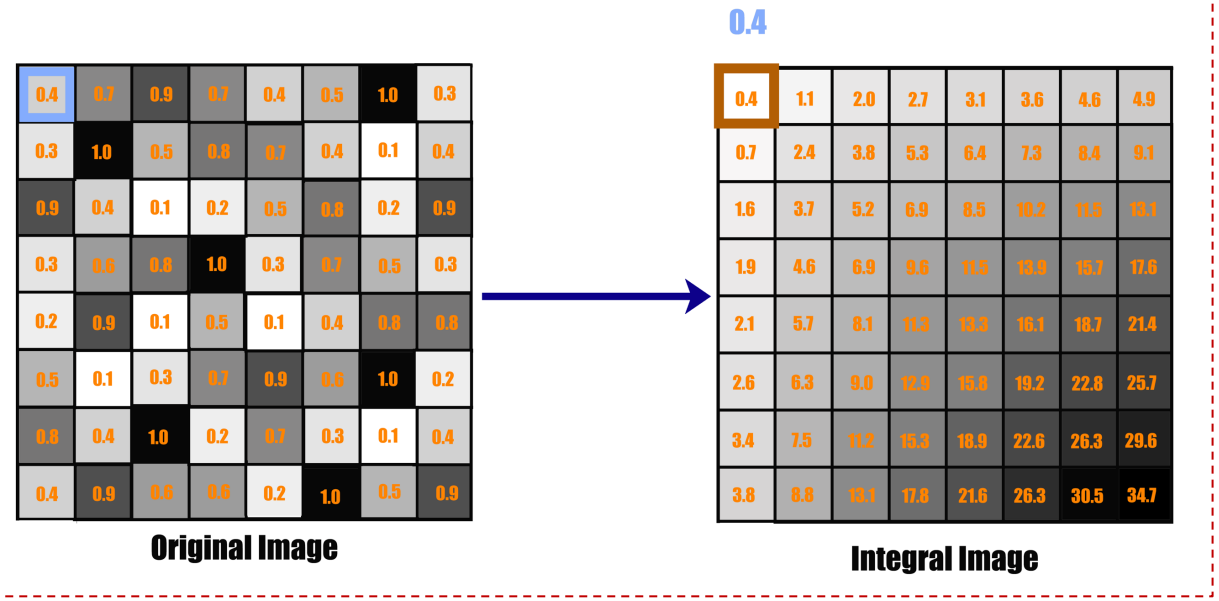
White region Dark region

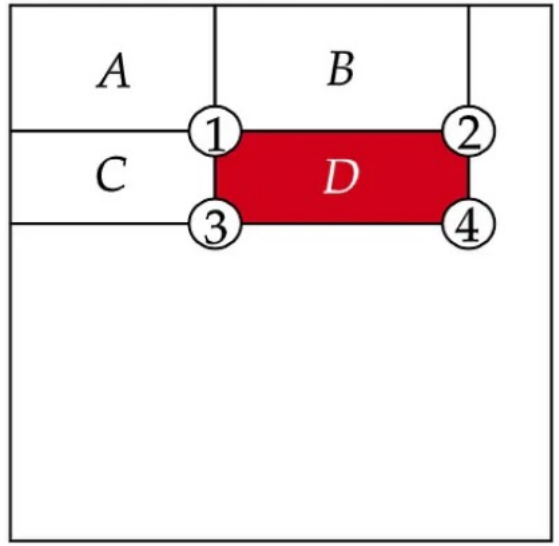
https://towardsdatascience.com/face-detection-with-haar-cascade-727f68dafd08

Outline

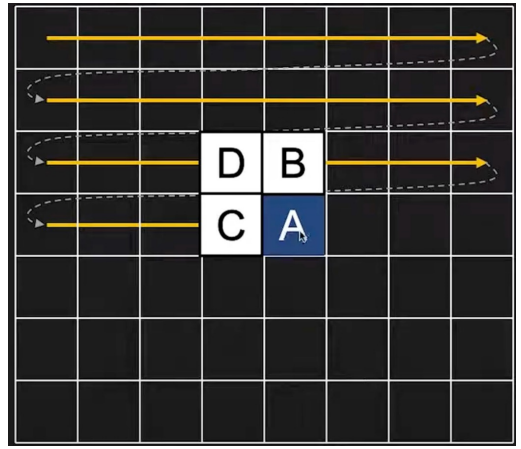
- Applications of Face Detection
- Face Detection in Computer Vision
- Haar Feature
- Integral Image
- Adaboost For Face Detection
- Cascade Classifier
- Face Recognition

Intergral Image



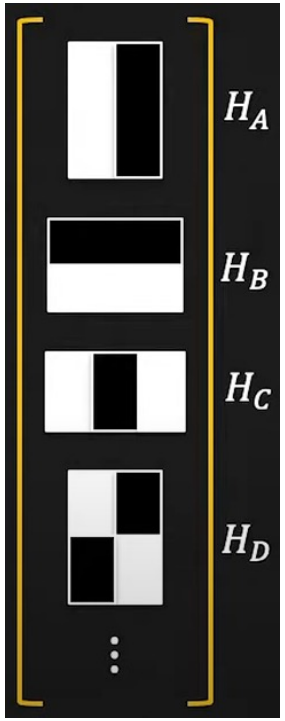


- The value of the integral image at point 1 is the sum of the pixels in rectangle A
- The value at point 2 is A + B
- The value at point 3 is A + C
- The value at point 4 is A + B + C + D.
- Therefore, the sum of pixels in region D can simply be computed as : 4+1−(2+3).



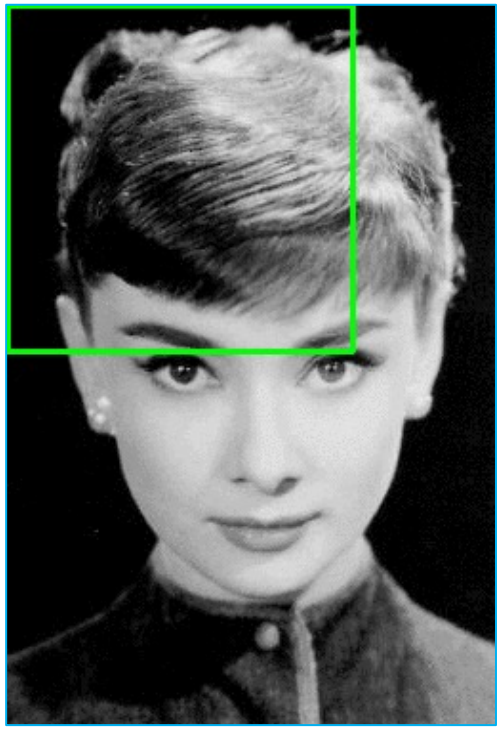
Raster Scanning

Haar Feature

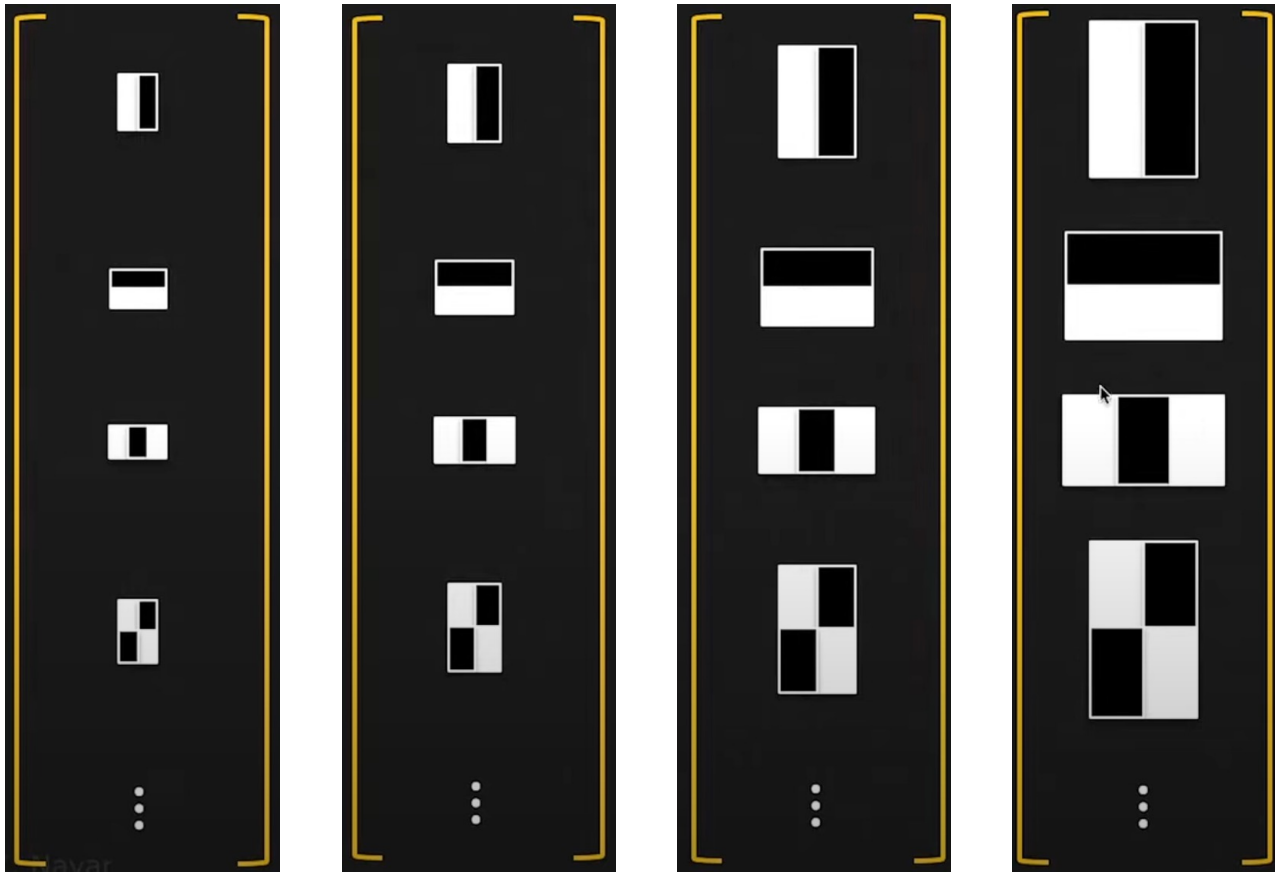


Haar Filters

Haar Features



Haar Features for Multiple Scales



$$V_A[i,j] = \sum_m \sum_n I[m-i, n-j] H_A[m,n]$$

Compute Haar Features at different scales to detect faces of different sizes

As stated previously there can be approximately 160,000 + feature values within a detector at 24*24 base resolution which need to be calculated. But it is to be understood that only a few set of features will be useful among all these features to identify a face.


Relevant feature


Irrelevant feature

Feature detecting a vertical edge is useful detecting a nose but irrelevant detecting a lip.

Can we create a good classifier using just a small subset of all possible features?

Outline

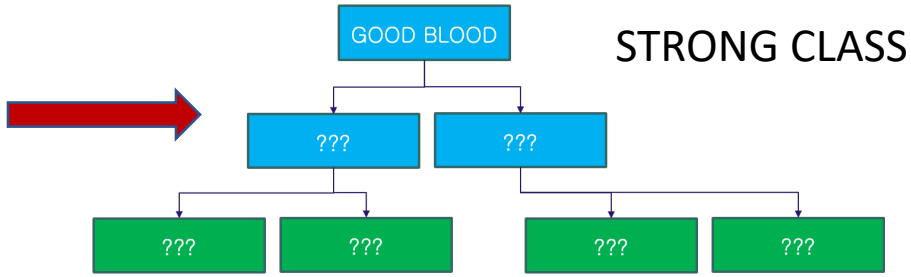
- Applications of Face Detection
- Face Detection in Computer Vision
- Haar Feature
- Integral Image
- Adaboost For Face Detection
- Cascade Classifier
- Face Recognition

Sample Dataset

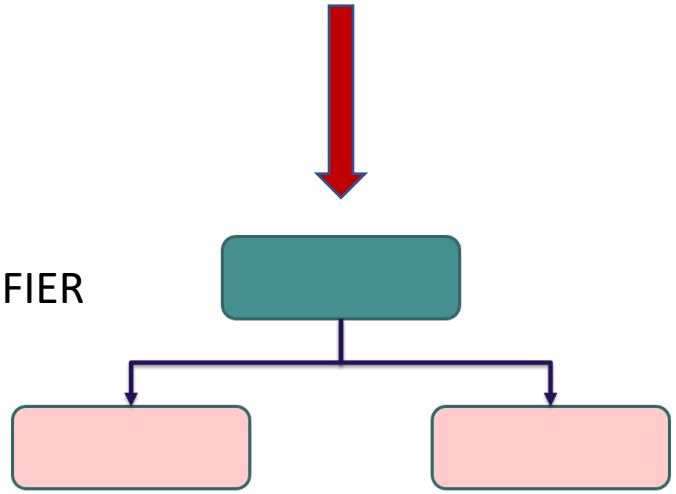
CHEST PAIN	GOOD BLOOD CIRCULATION	BLOCKED ARTERIES	WEIGHT	HEART DISEASE
NO	NO	NO	125	NO
YES	YES	YES	180	YES
YES	YES	NO	210	NO
YES	NO	YES	167	YES

Sample Dataset

CHEST PAIN	GOOD BLOOD CIRCULATION	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



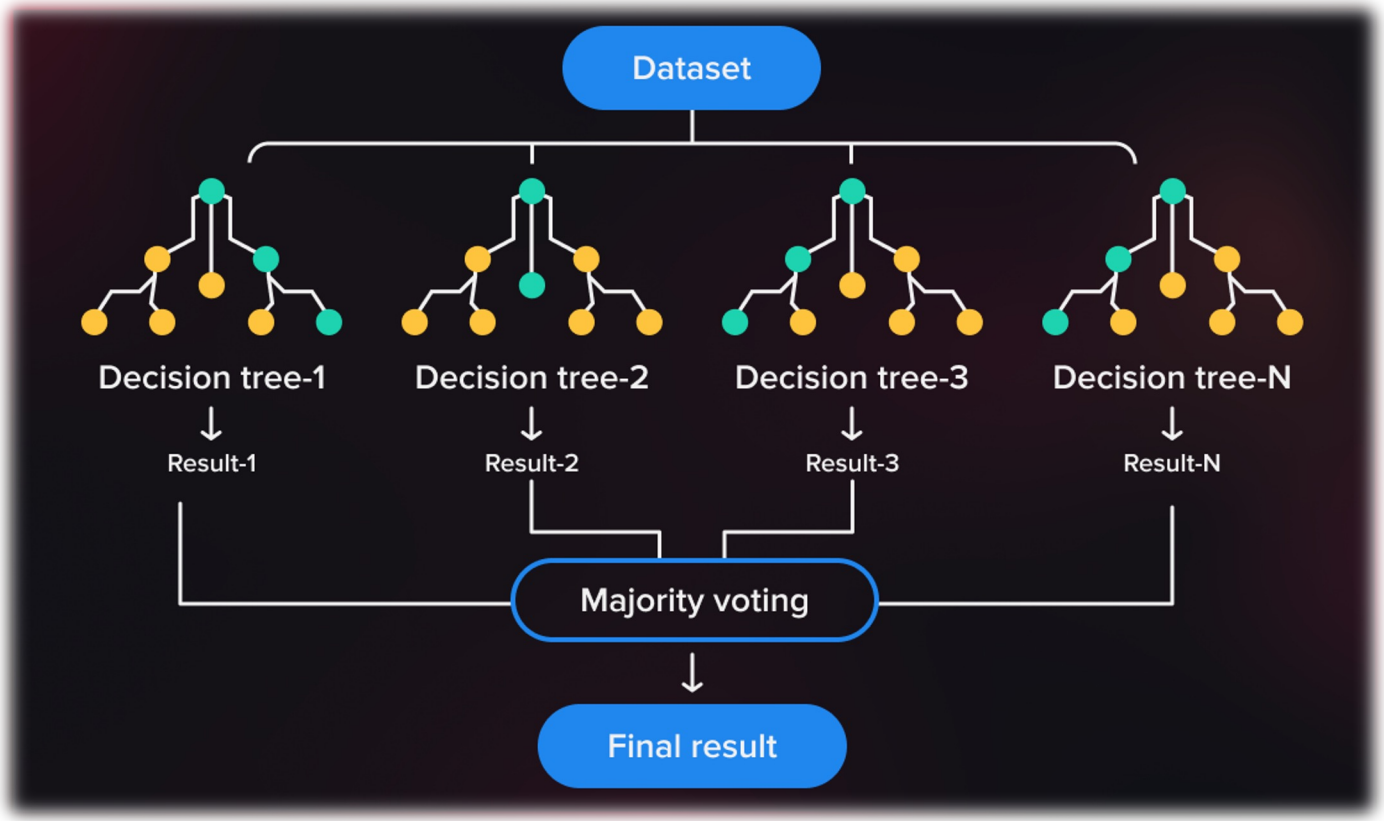
STRONG CLASSIFIER



WEAK CLASSIFIER

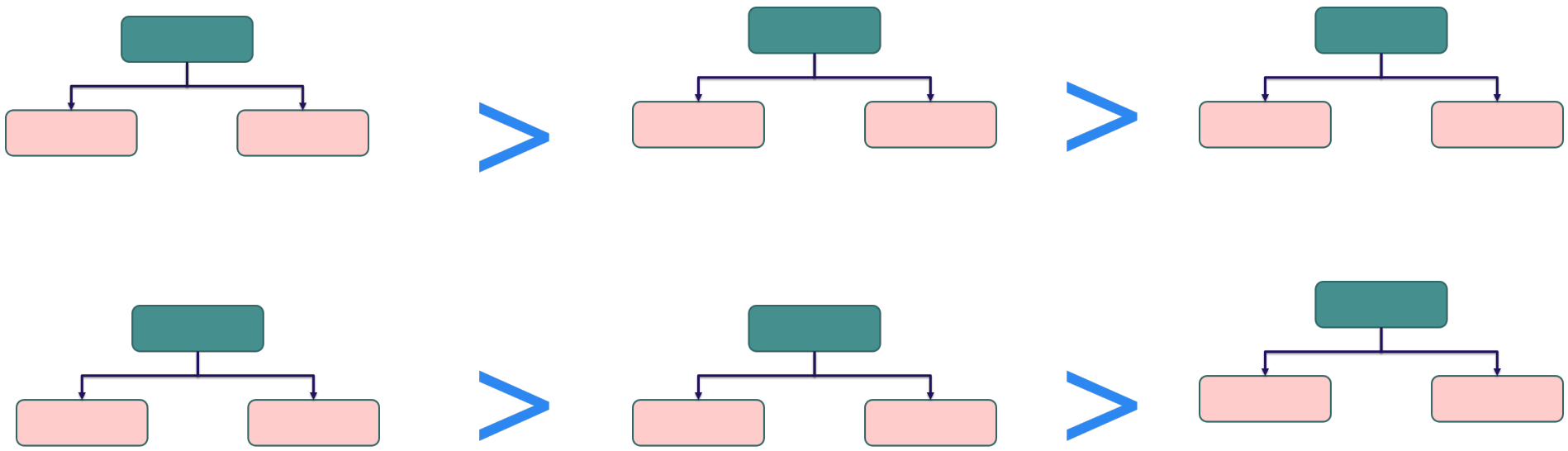
RANDOM FOREST

- Each tree in the random forest has equal votes(weights) on the final decision



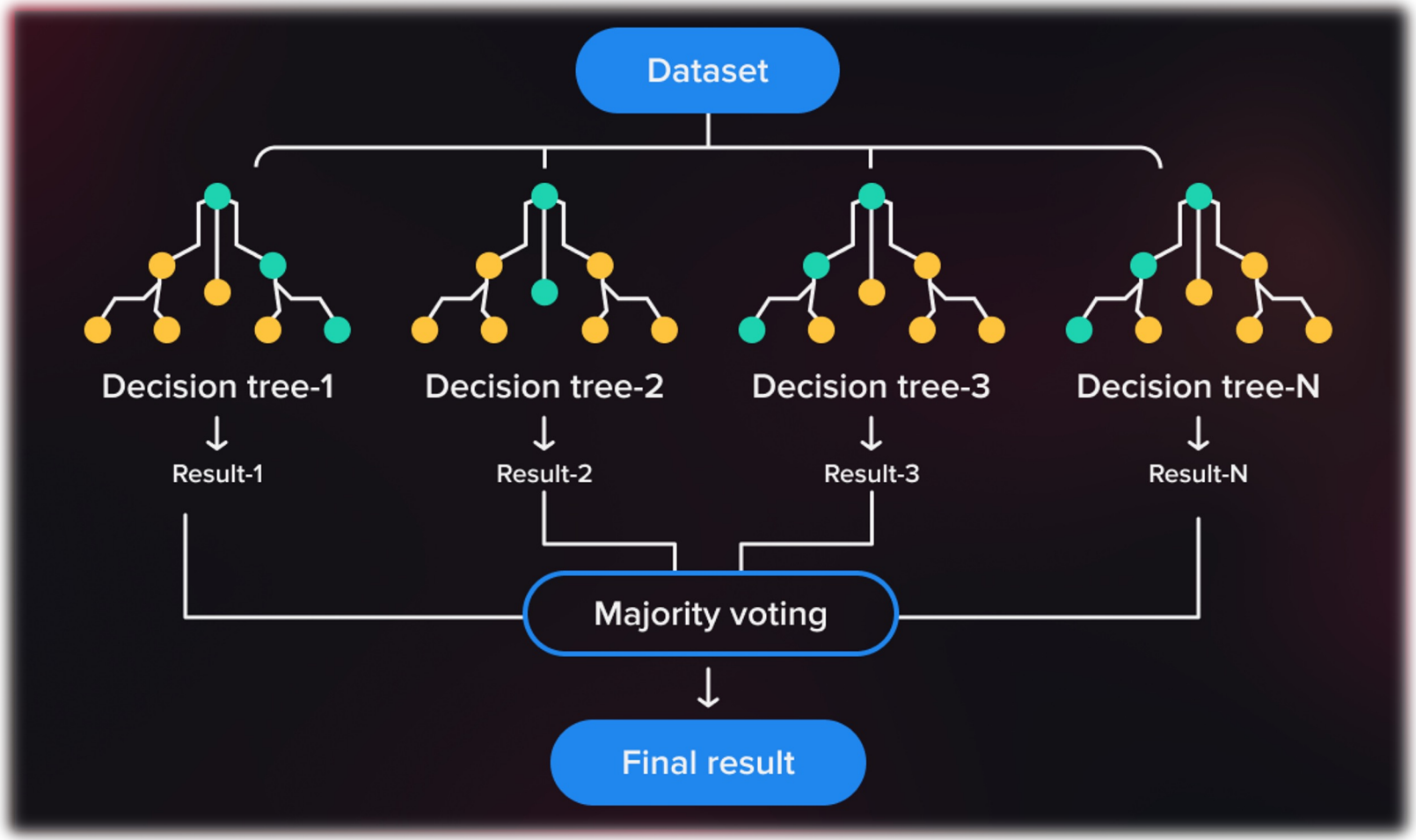
AdaBoost: FOREST OF STUMP

- Stump are not equally weighted in the final decision.
- Stump that create more error will have less contribution in the final decision

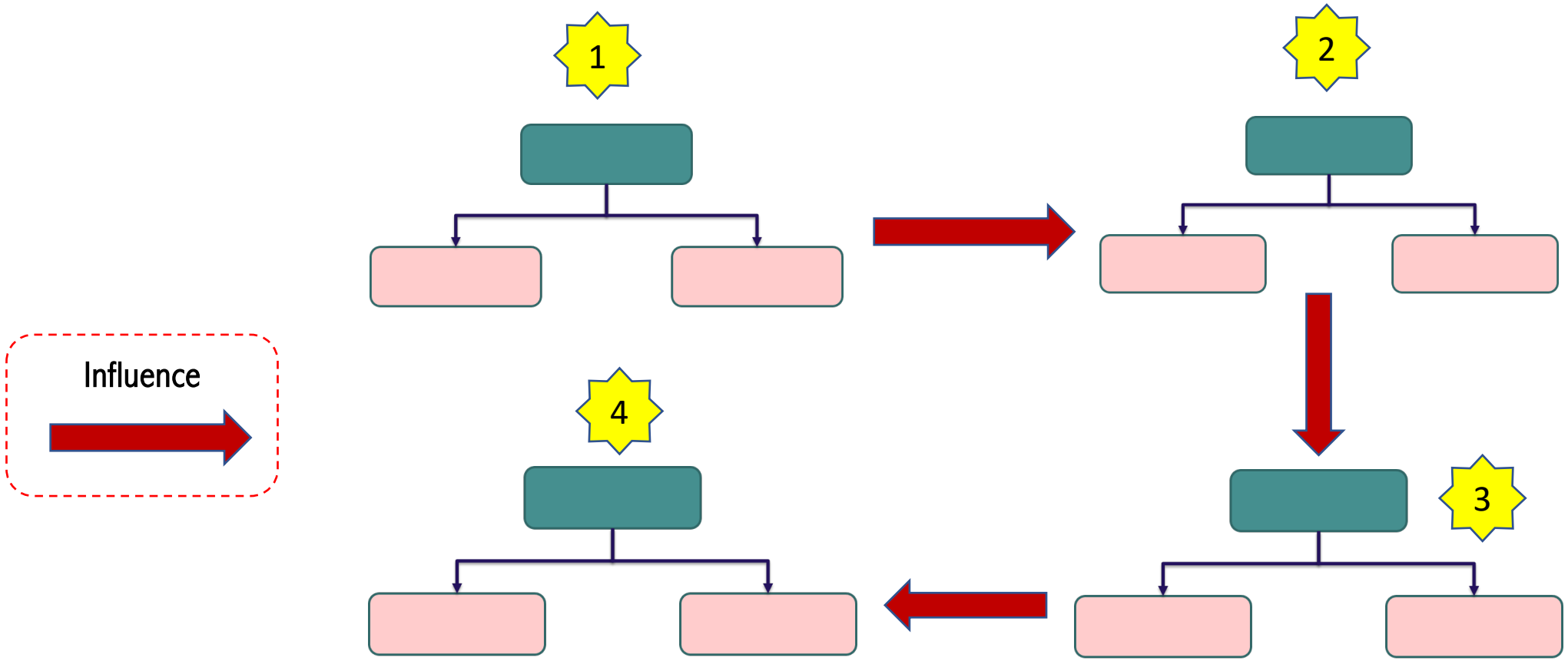


Random Forest

Tree are indepently created



AdaBoost: FOREST OF STUMP



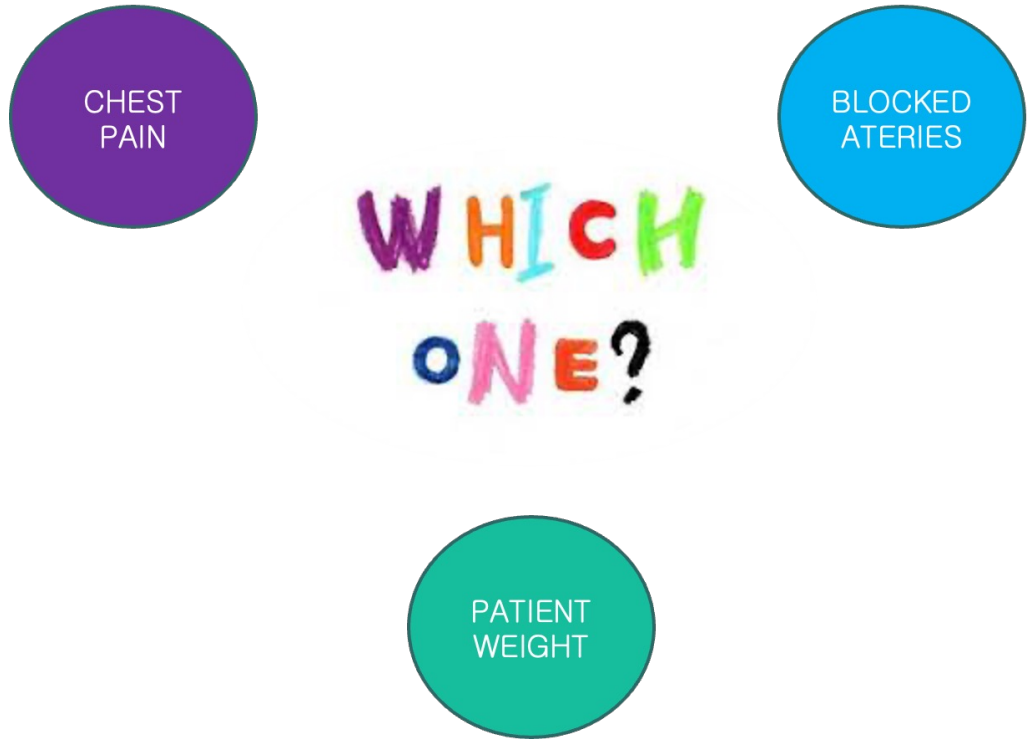
Heart Disease Dataset

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease
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

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

1st Stump in the Tree

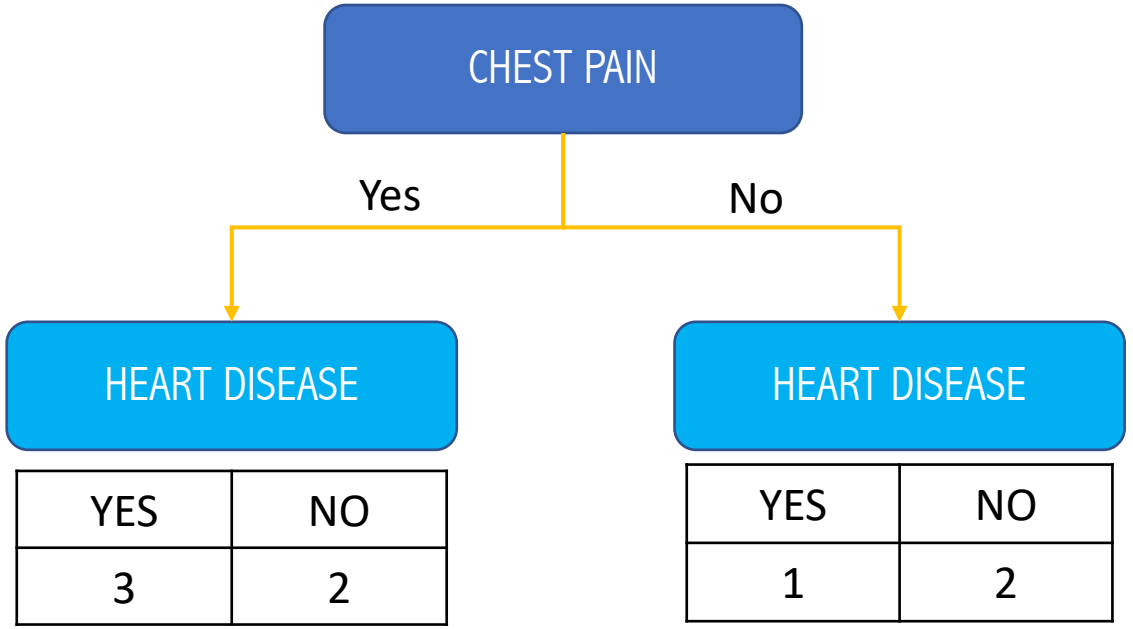
Chest Pain	Blocked Arteries	Patient Weight	Heart Diease
Yes	Yes	205	Yes
No	Yes	180	Yes
Yes	No	210	Yes
Yes	Yes	167	Yes
No	Yes	156	No
No	Yes	125	No
Yes	No	168	No
Yes	Yes	172	No



Compute Gini Index For Chest Pain

Chest Pain	Heart Diease	Sample Weight
Yes	Yes	1/8
No	Yes	1/8
Yes	Yes	1/8
Yes	Yes	1/8
No	No	1/8
No	No	1/8
Yes	No	1/8
Yes	No	1/8

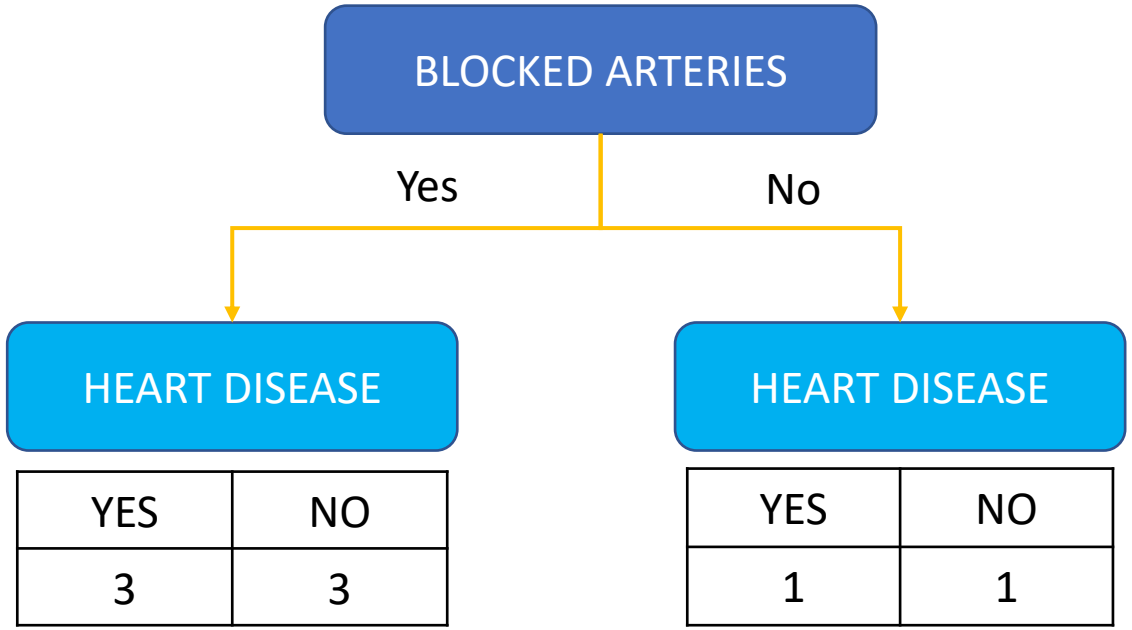
Gini index = 5/8 * (1 – (3/5)^2 – (2/5)^2) + 3/8* (1 - (1/3)^2 - (2/3)^2) = **0.57**



GINI INDEX FOR BLOCKED ARTERIES

Blocked Arteries	Heart Diease	Sample Weight
Yes	Yes	1/8
Yes	Yes	1/8
No	Yes	1/8
Yes	Yes	1/8
Yes	No	1/8
Yes	No	1/8
No	No	1/8
Yes	No	1/8

Gini index = 6/8 * (1 – (3/6)^2 – (3/6)^2) + 2/8* (1 - (1/2)^2 - (1/2)^2) = **0.5**



YES	NO
3	3

YES	NO
1	1

Gini Index for Heart Disease

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

Gini index = =4/8 * (1-(1/4)^2 - (3/4)^2) + 4/8* (1-(1/4)^2 -(3/4)^2) = 0.375

192.5

195

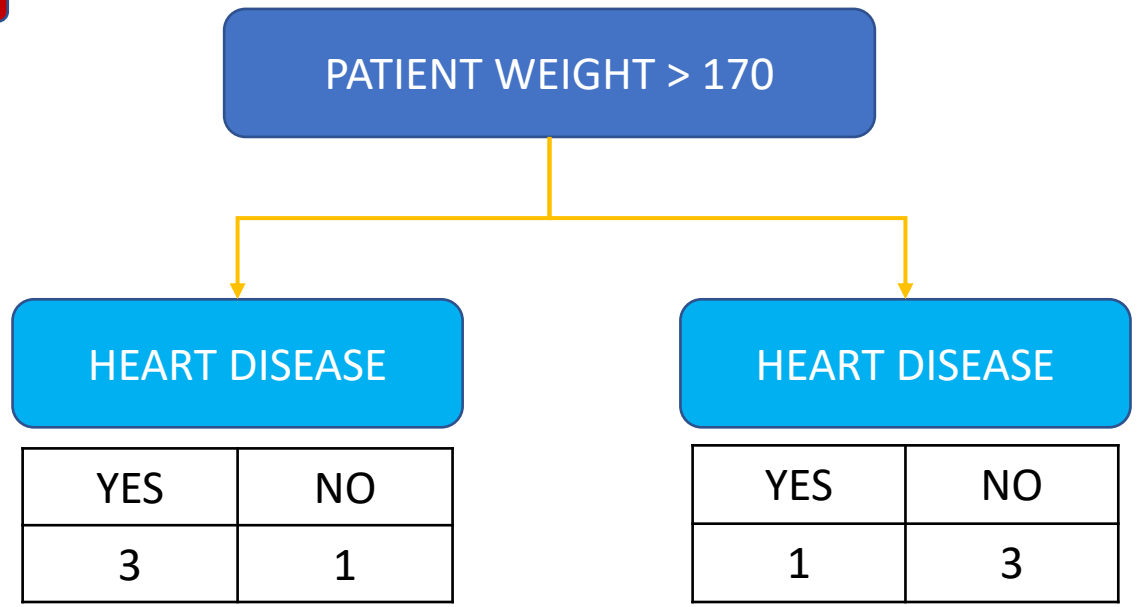
188.5

161.5

140.5

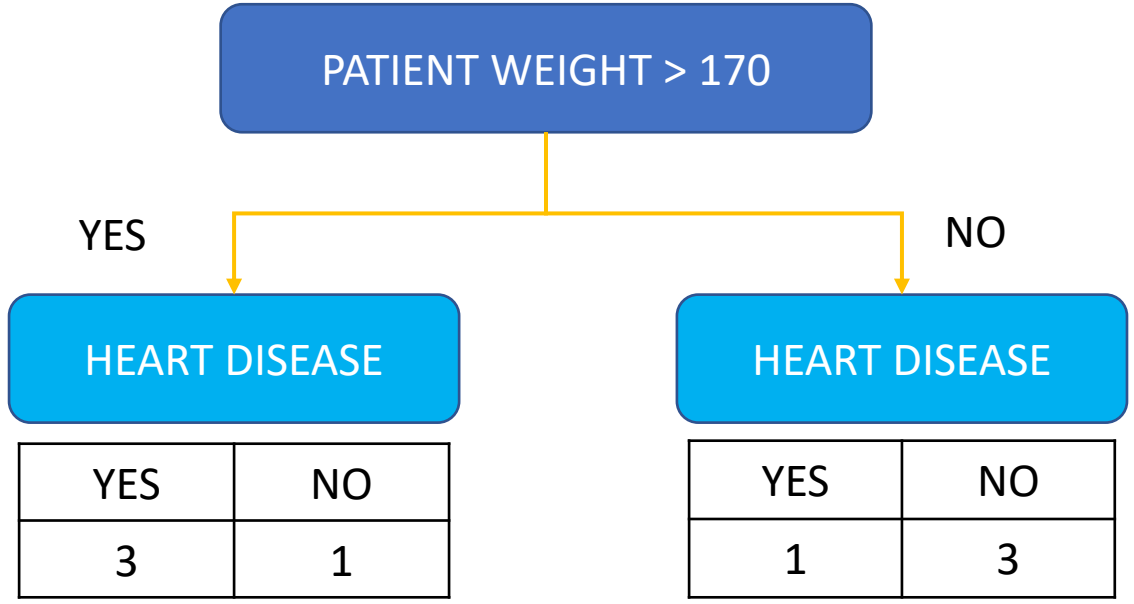
146.5

170



Amount of Say

How was this stump contribute to the final decision (classification)?



YES	NO
3	1

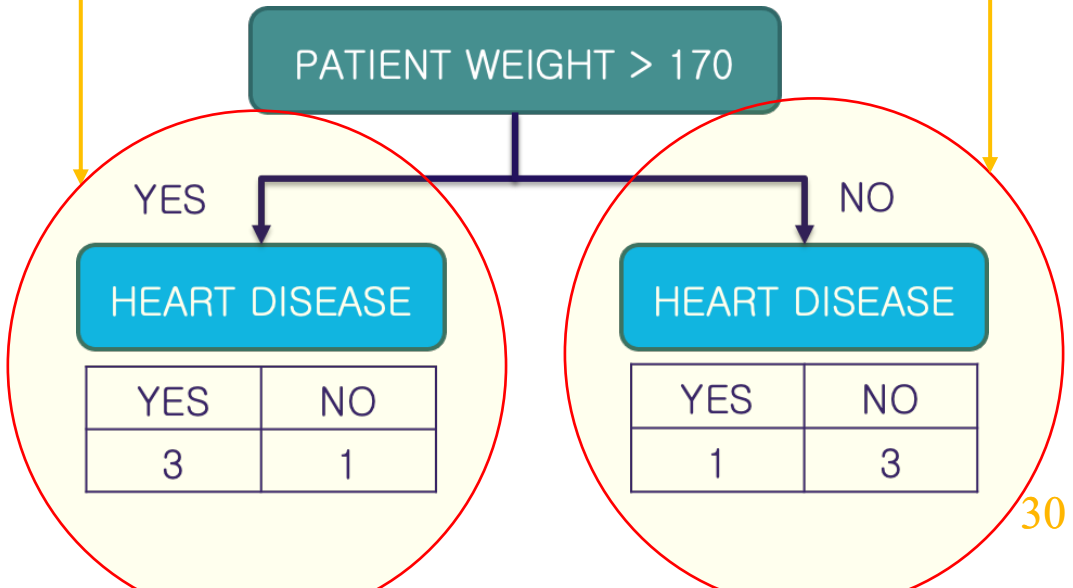
YES	NO
1	3

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

Amount of Say: Patient Weight

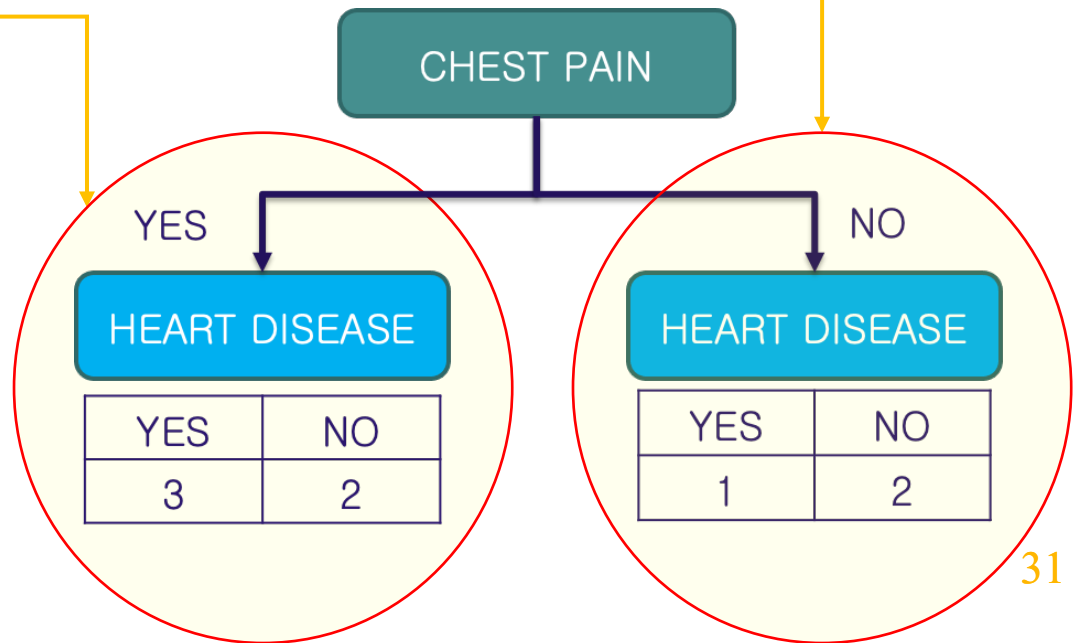
Chest Pain	Blocked Arteries	Patient Weight	Heart Diease	Sample Weight
Yes	Yes	205	Yes	1/8
No	Yes	180	Yes	1/8
Yes	No	210	Yes	1/8
Yes	Yes	167	Yes	1/8
No	Yes	156	No	1/8
No	Yes	125	No	1/8
Yes	No	168	No	1/8
Yes	Yes	172	No	1/8

- 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



Amount of say: Chest Pain

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

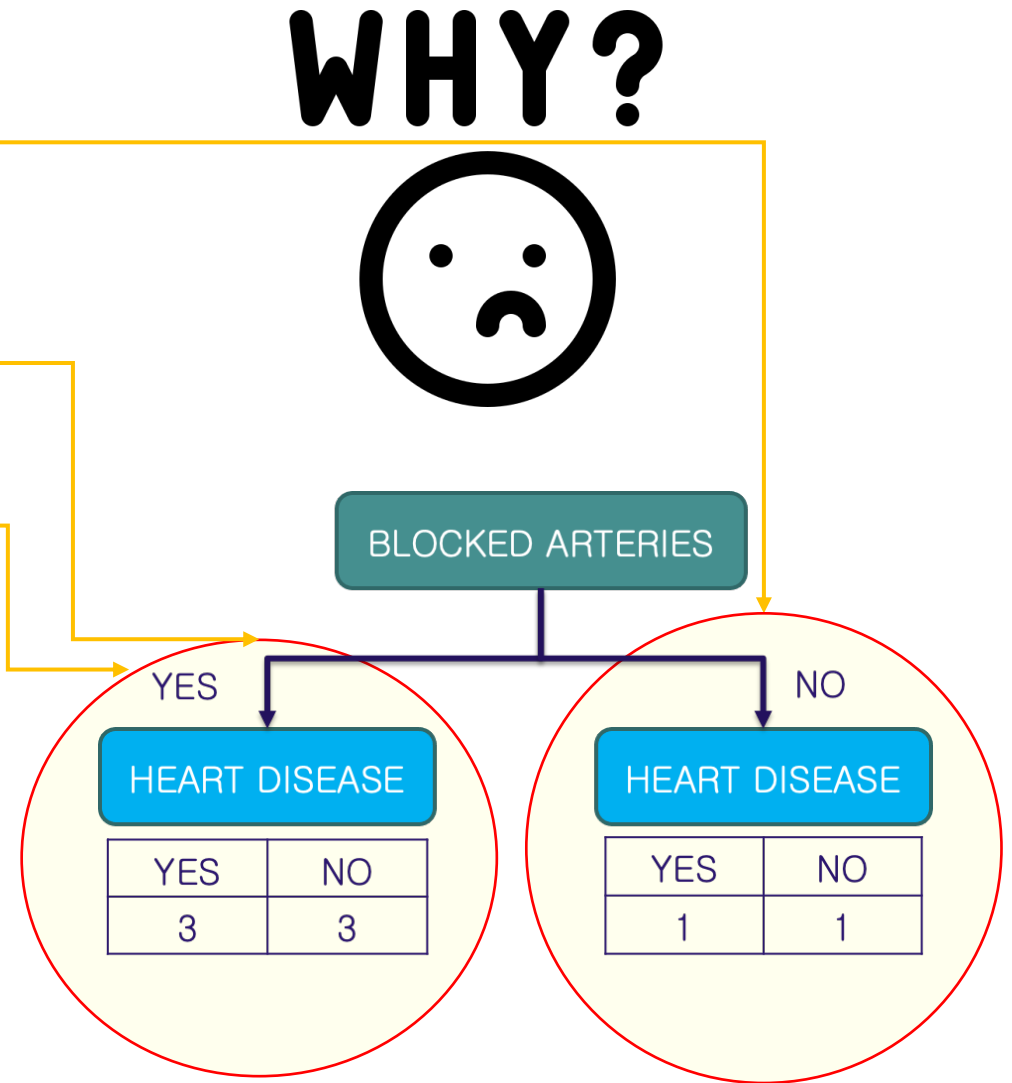


- Total error is equal to the sum of the weights of the incorrect classified
- Amount of say = 1/2*log((1-3/8) / (3/8)) = 0.25

$$\log(\text{Odds}) = \log\left(\frac{1 - \text{Probability of incorrect prediction}}{\text{Probability of incorrect prediction}}\right)$$

Amout of Say: Blocked Arteries

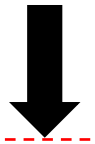
Chest Pain	Blocked Arteries	Patient Weight	Heart Diease	Sample Weight
Yes	Yes	205	Yes	1/8
No	Yes	180	Yes	1/8
Yes	No	210	Yes	1/8
Yes	Yes	167	Yes	1/8
No	Yes	156	No	1/8
No	Yes	125	No	1/8
Yes	No	168	No	1/8
Yes	Yes	172	No	1/8



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

Assumptions

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



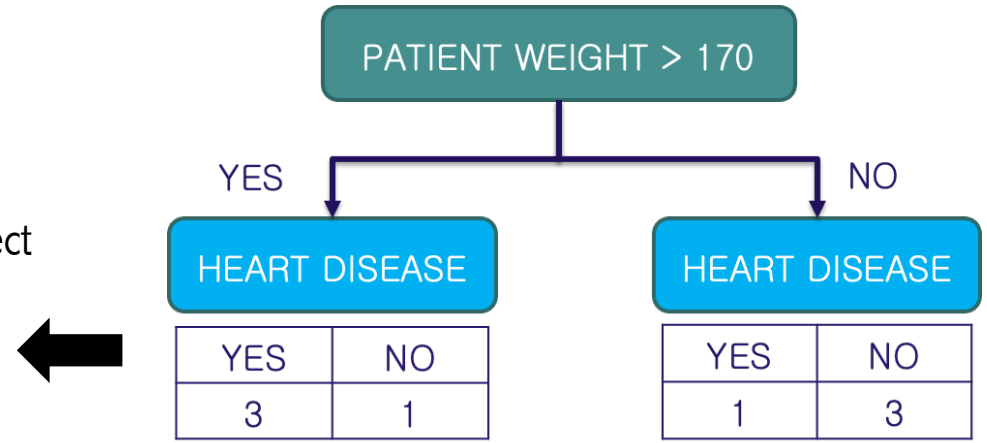
Unknown: Tiếp theo, chúng ta cần làm thế nào để sử dụng thông tin các weight của sample dự đoán sai này để xây dựng stump tiếp và khắc phục các dự đoán sai này

Idea: Improved Bootstrapped Dataset

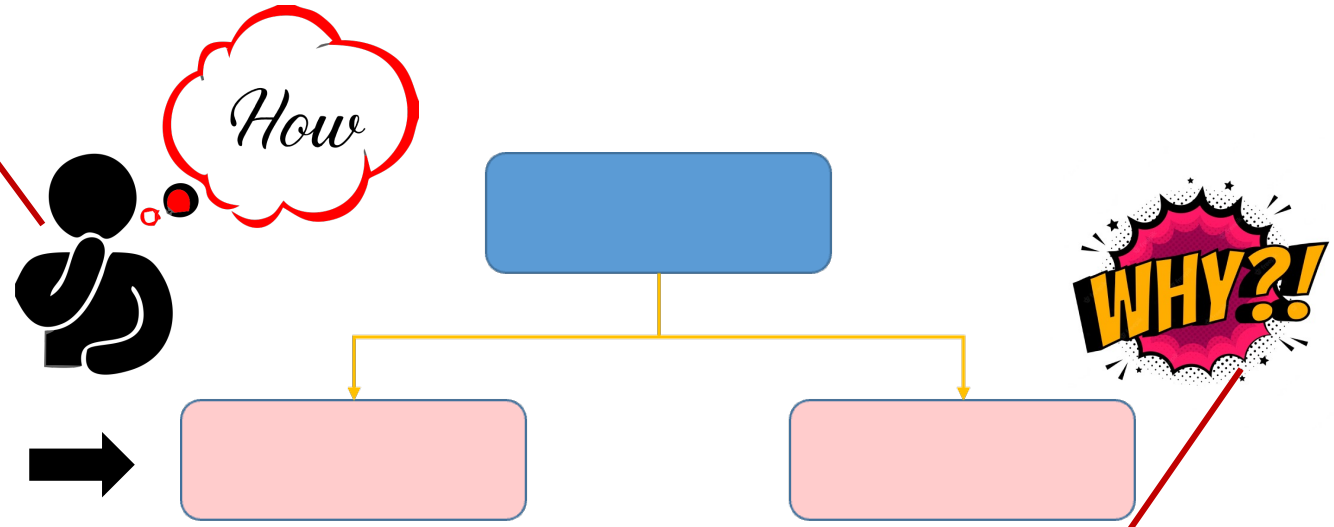
Chest Pain	Blocked Arteries	Patient Weight	Heart Disease
Yes	Yes	205	Yes
No	Yes	180	Yes
Yes	No	210	Yes
Yes	Yes	167	Yes
No	Yes	156	No
No	Yes	125	No
Yes	No	168	No
Yes	Yes	172	No

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease
Yes	Yes	205	Yes
No	Yes	180	Yes
Yes	No	210	Yes
Yes	Yes	167	Yes
Yes	Yes	167	Yes
Yes	Yes	167	Yes
Yes	Yes	172	No
Yes	Yes	172	No

Incorrect



Incorrect

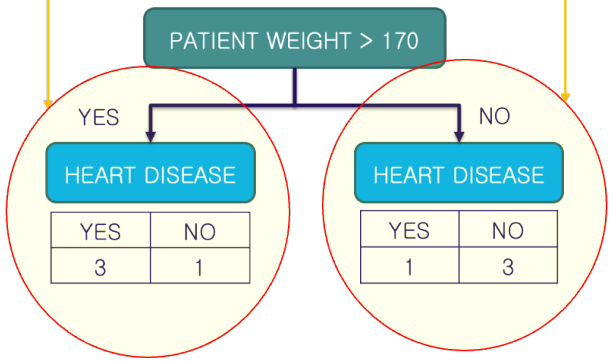


How to build next Stump

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



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

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

```
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
```

How to build next Stump

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

$$\text{New Sample Weight} = \text{sample weight} \times e^{\text{-amount of say}}$$

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

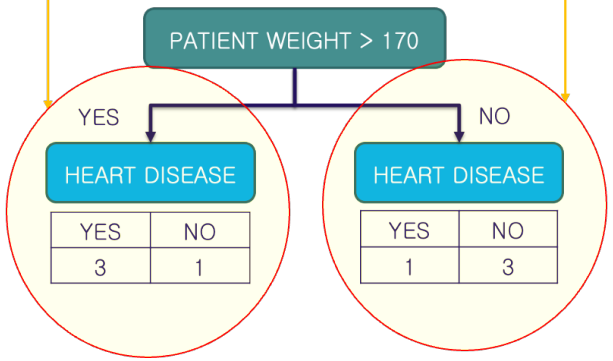
```
def update_weights_formular1(w_i, alpha, y, y_pred):
    result = w_i * np.exp(-alpha * y * y_pred)
    w_norm = result / np.sum(result)
    return w_norm
```

How to build next Stump

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

```
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
```

How to build next Stump

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
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

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

New sample weight = 1/8 * e^{0.55*0} = 0.125

```
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
```

New Sample Weight

Chest Pain	Blocked Arteries	Patient Weight	Heart Diease	Sample Weight	New Weight	Normal Weight
Yes	Yes	205	Yes	1/8	0.07	0.08
No	Yes	180	Yes	1/8	0.07	0.08
Yes	No	210	Yes	1/8	0.07	0.08
Yes	Yes	167	Yes	1/8	0.22	0.25
No	Yes	156	No	1/8	0.07	0.08
No	Yes	125	No	1/8	0.07	0.08
Yes	No	168	No	1/8	0.07	0.08
Yes	Yes	172	No	1/8	0.22	0.25
Sum				~1.0	0.86	~1.0

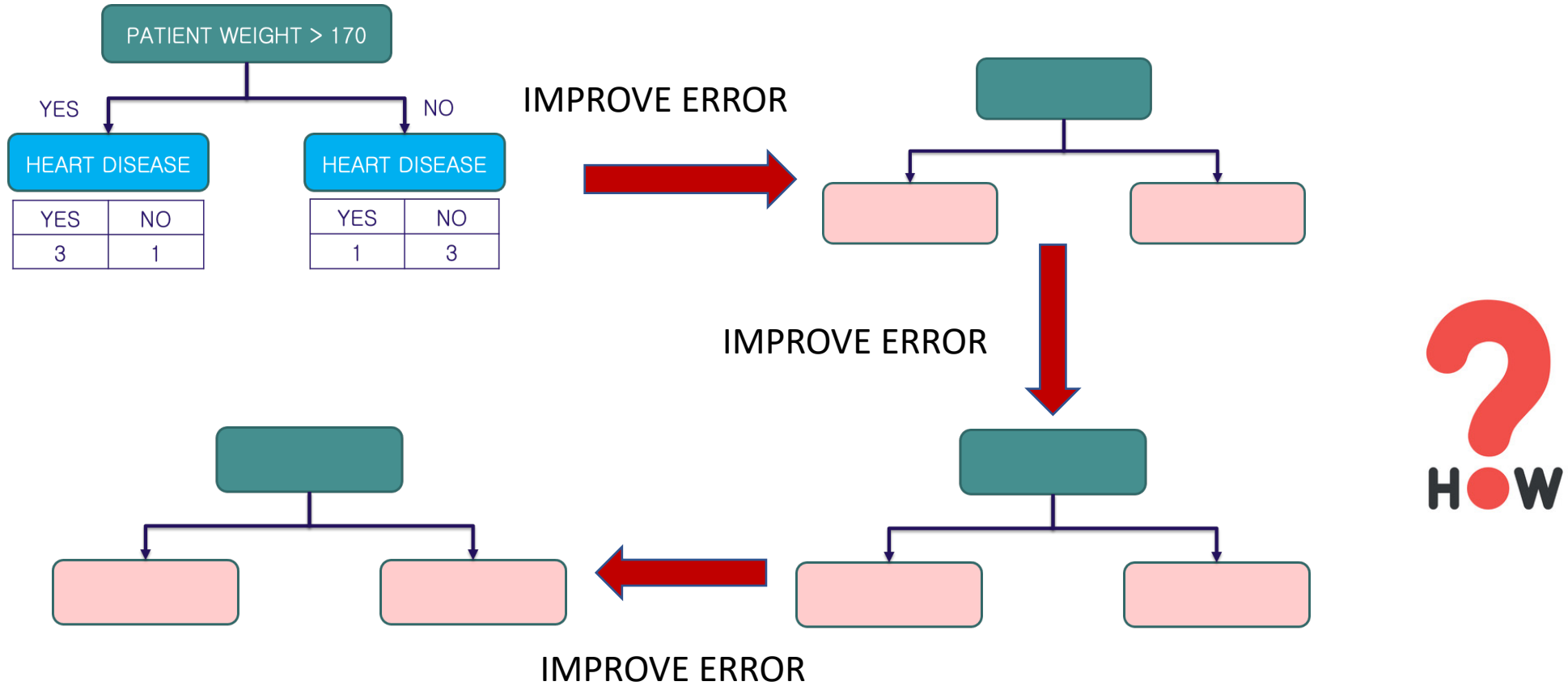
$$\text{New Sample Weight} = \text{sample weight} \times e^{\text{amount of say}}$$

Update

New Sample Weight

Chest Pain	Blocked Arteries	Patient Weight	Heart Diease	New Weight
Yes	Yes	205	Yes	0.08
No	Yes	180	Yes	0.08
Yes	No	210	Yes	0.08
Yes	Yes	167	Yes	0.25
No	Yes	156	No	0.08
No	Yes	125	No	0.08
Yes	No	168	No	0.08
Yes	Yes	172	No	0.25
Sum				~1.0

AdaBoost: FOREST OF STUMPS



New Dataset

Chest Pain	Blocked Arteries	Patient Weight	Heart Diease	Normal Weight
Yes	Yes	205	Yes	0.08
No	Yes	180	Yes	0.08
Yes	No	210	Yes	0.08
Yes	Yes	167	Yes	0.25
No	Yes	156	No	0.08
No	Yes	125	No	0.08
Yes	No	168	No	0.08
Yes	Yes	172	No	0.25
Sum				~1.0

Chest Pain	Blocked Arteries	Patient Weight	Heart Diease	Normal Weight
Yes	Yes	205	Yes	0.08
No	Yes	180	Yes	0.08
Yes	No	210	Yes	0.08
Yes	Yes	167	Yes	0.25
No	Yes	156	No	0.08
No	Yes	125	No	0.08
Yes	No	168	No	0.08
Yes	Yes	172	No	0.25
Sum				~1.0

New Dataset

Chest Pain	Blocked Arteries	Patient Weight	Heart Diease	Normal Weight
Yes	Yes	205	Yes	0.08
No	Yes	180	Yes	0.08
Yes	No	210	Yes	0.08
Yes	Yes	167	Yes	0.25
No	Yes	156	No	0.08
No	Yes	125	No	0.08
Yes	No	168	No	0.08
Yes	Yes	172	No	0.25
Sum				~1.0

Random [0,1] to select samples

Chest Pain	Blocked Arteries	Patient Weight	Heart Diease	Normal Weight
Yes	Yes	205	Yes	0.08
No	Yes	180	Yes	0.08
Yes	No	210	Yes	0.08
Yes	Yes	167	Yes	0.25
No	Yes	156	No	0.08
No	Yes	125	No	0.08
Yes	No	168	No	0.08
Yes	Yes	172	No	0.25
Sum				~1.0

New Dataset

Chest Pain	Blocked Arteries	Patient Weight	Heart Diease	Normal Weight	Range
Yes	Yes	205	Yes	0.08	[0-0.08]
No	Yes	180	Yes	0.08	(0.08-0.16]
Yes	No	210	Yes	0.08	(0.16-0.24]
Yes	Yes	167	Yes	0.25	(0.24-0.495]
No	Yes	156	No	0.08	(0.495-0.575]
No	Yes	125	No	0.08	(0.575-0.655]
Yes	No	168	No	0.08	(0.655-0.735]
Yes	Yes	172	No	0.25	(0.735-1.0]
Sum				~1.0	

New Dataset

Chest Pain	Blocked Arteries	Patient Weight	Heart Diease	Normal Weight
Yes	Yes	205	Yes	0.08
No	Yes	180	Yes	0.08
Yes	No	210	Yes	0.08
Yes	Yes	167	Yes	0.25
No	Yes	156	No	0.08
No	Yes	125	No	0.08
Yes	No	168	No	0.08
Yes	Yes	172	No	0.25
Sum				~1.0

Old dataset

Random [0,1] to select samples

CONTINUE TO BUILD THE NEXT STUMP

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

New dataset

Ý tưởng: các sample bị phân loại sai, sẽ được được nhiều hơn vào dataset mới

How to Classify The Final Result

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*
   $$C(\boldsymbol{x}) = \arg\max_k \sum_{m=1}^{M} \alpha^{(m)} \cdot \mathbb{I}(T^{(m)}(\boldsymbol{x}) = k).$$

https://hastie.su.domains/Papers/samme.pdf

How to Classify The Final Result

SAMME — Stagewise Additive Modeling using a Multi-class Exponential loss function

Algorithm 2 *SAMME*

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*

$$\alpha^{(m)} = \log \frac{1 - err^{(m)}}{err^{(m)}} + \log(K - 1). \tag{1}$$

   (d) *Set*

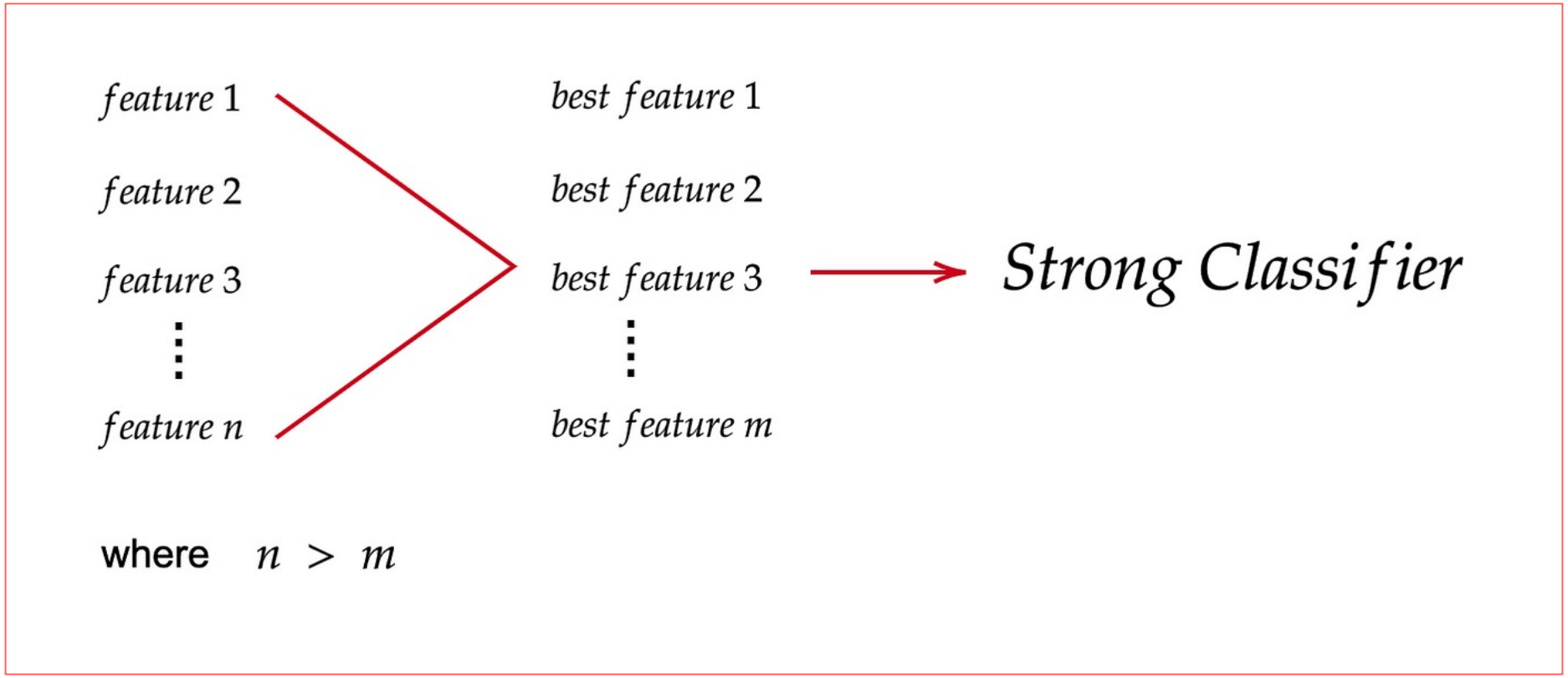
$$w_i \leftarrow w_i \cdot \exp\left(\alpha^{(m)} \cdot \mathbb{I}\left(c_i \neq T^{(m)}(\boldsymbol{x}_i)\right)\right),\ i = 1, \ldots, n.$$

   (e) *Re-normalize* w_i.
3. *Output*

$$C(\boldsymbol{x}) = \arg\max_k \sum_{m=1}^{M} \alpha^{(m)} \cdot \mathbb{I}(T^{(m)}(\boldsymbol{x}) = k).$$

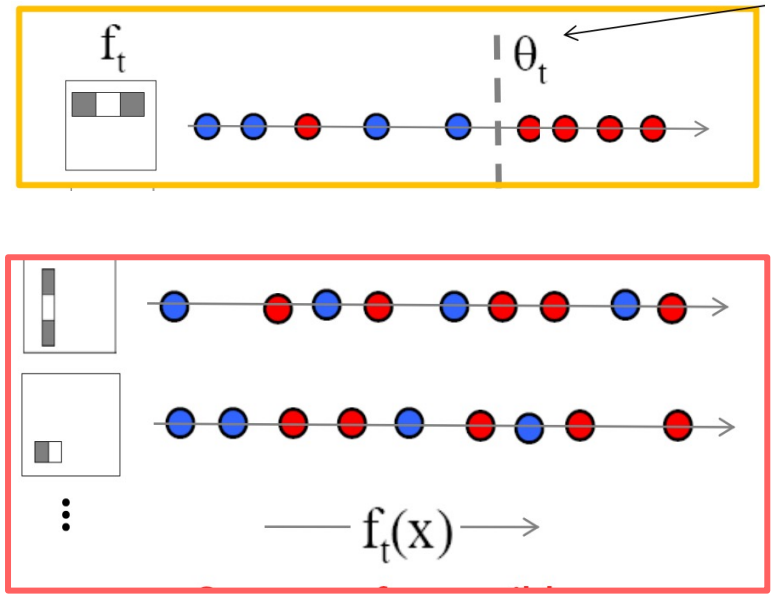
https://hastie.su.domains/Papers/samme.pdf

Adaboost for Face Detection



The goal of using the AdaBoost algorithm is to extract the best features from n features

AdaBoost for feature+classifier selection



θ^t is a threshold for classifier h^t

Resulting weak classifier:

$$h_t(x) = \begin{cases} +1 & \text{if } f_t(x) > \theta_t \\ 0 & \text{otherwise} \end{cases}$$

For next round, reweight the examples according to errors, choose another filter/threshold combo.

- Each weak classifier works on exactly one rectangle feature.
- Each weak classifier has 3 associated variables
  1. its threshold θ
  2. its polarity p
  3. its weight α

  h(x) = 1 if p*f(x) < pθ, else 0

  used for the combination step

  The code does not actually compute h.
- The polarity can be 0 or 1
- The weak classifier computes its one feature f
  - When the polarity is 1, we want f > θ for face
  - When the polarity is 0, we want f < θ for face
- The weight will be used in the final classification by AdaBoost.

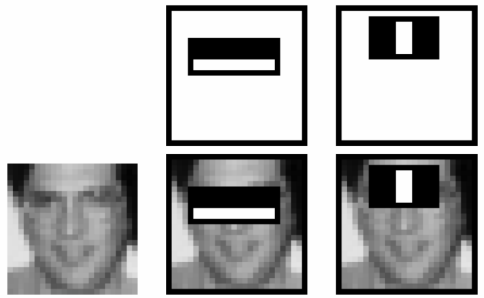
Want to select the single rectangle feature and threshold that best separates positive (faces) and negative (non-faces) training examples, in terms of weighted error

1. Initialized weights for each training example
2. Normalized all the weights
3. Then we selected the best weak classifier based on the weighted error of the training examples
4. Then updated the weights according to the error of the chosen best weak classifier
5. Repeated steps 2–4 N times, where N is the desired number of weak classifiers

AdaBoost for feature+classifier selection

- Given example images $(x_1, y_1), \ldots, (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples respectively.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of negatives and positives respectively.
- For $t = 1, \ldots, T$:
  1. Normalize the weights,
  $$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$
  so that w_t is a probability distribution.
  2. For each feature, j, train a classifier h_j which is restricted to using a single feature. The error is evaluated with respect to w_t, $\epsilon_j = \sum_i w_i \, |h_j(x_i) - y_i|$.
  3. Choose the classifier, h_t, with the lowest error ϵ_t.
  4. Update the weights:
  $$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$$
  where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.
- The final strong classifier is:
$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$
where $\alpha_t = \log \frac{1}{\beta_t}$

First two features selected by boosting:



This feature combination can yield 100% detection rate and 50% false positive rate

A 200-feature classifier can yield 95% detection rate and a false positive rate of 1 in 14084

Outline

- Applications of Face Detection
- Face Detection in Computer Vision
- Haar Feature
- Integral Image
- Adaboost For Face Detection
- Cascade Classifier
- Face Recognition

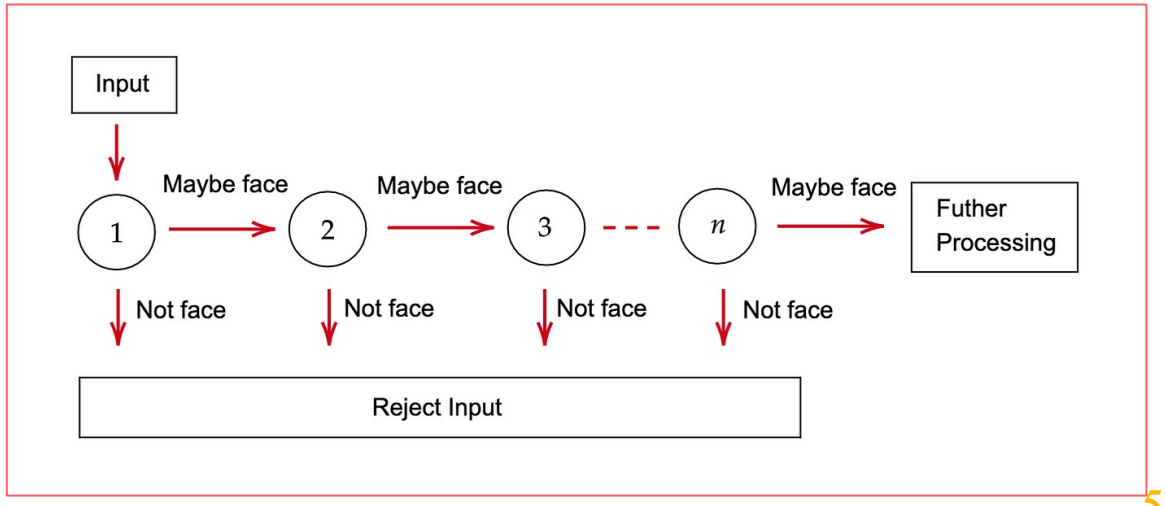
The Cascade Classifier

In an image, most of the image is a non-face region

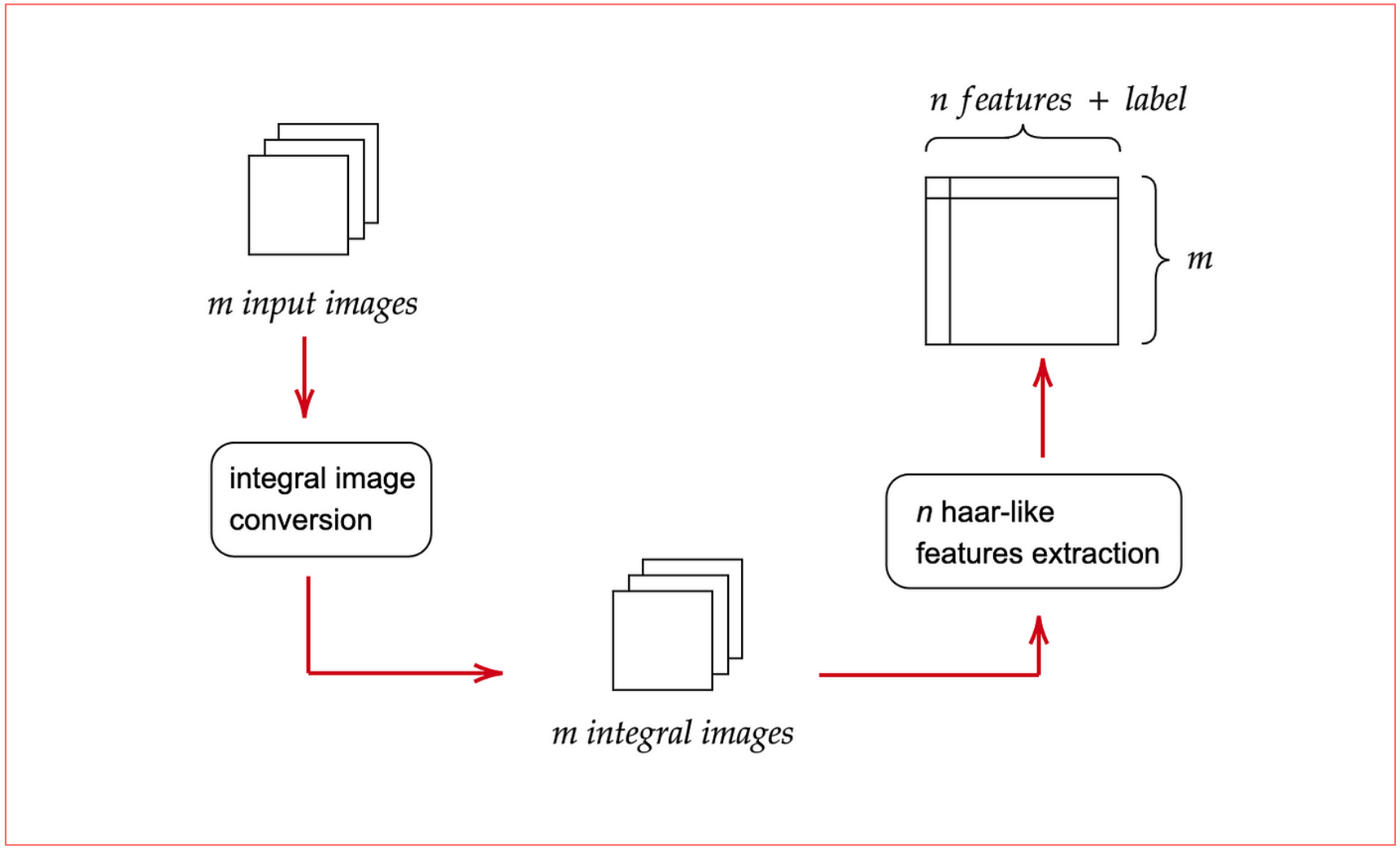
The key idea is to reject sub-windows that do not contain faces while identifying regions that do. Since the task is to identify properly the face, we want to minimize the false negative rate, i.e the sub-windows that contain a face and have not been identified as such.

A series of classifiers are applied to every sub-window. These classifiers are simple decision trees :

- if the first classifier is positive, we move on to the second
- if the second classifier is positive, we move on to the third.
- Any negative result at some point leads to a rejection of the sub-window as potentially containing a face

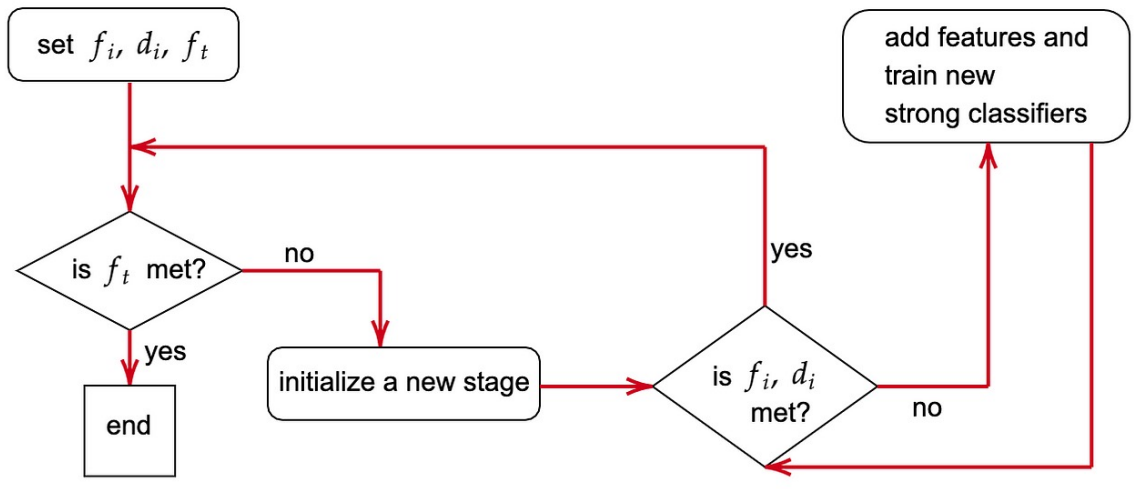


Data Preparation



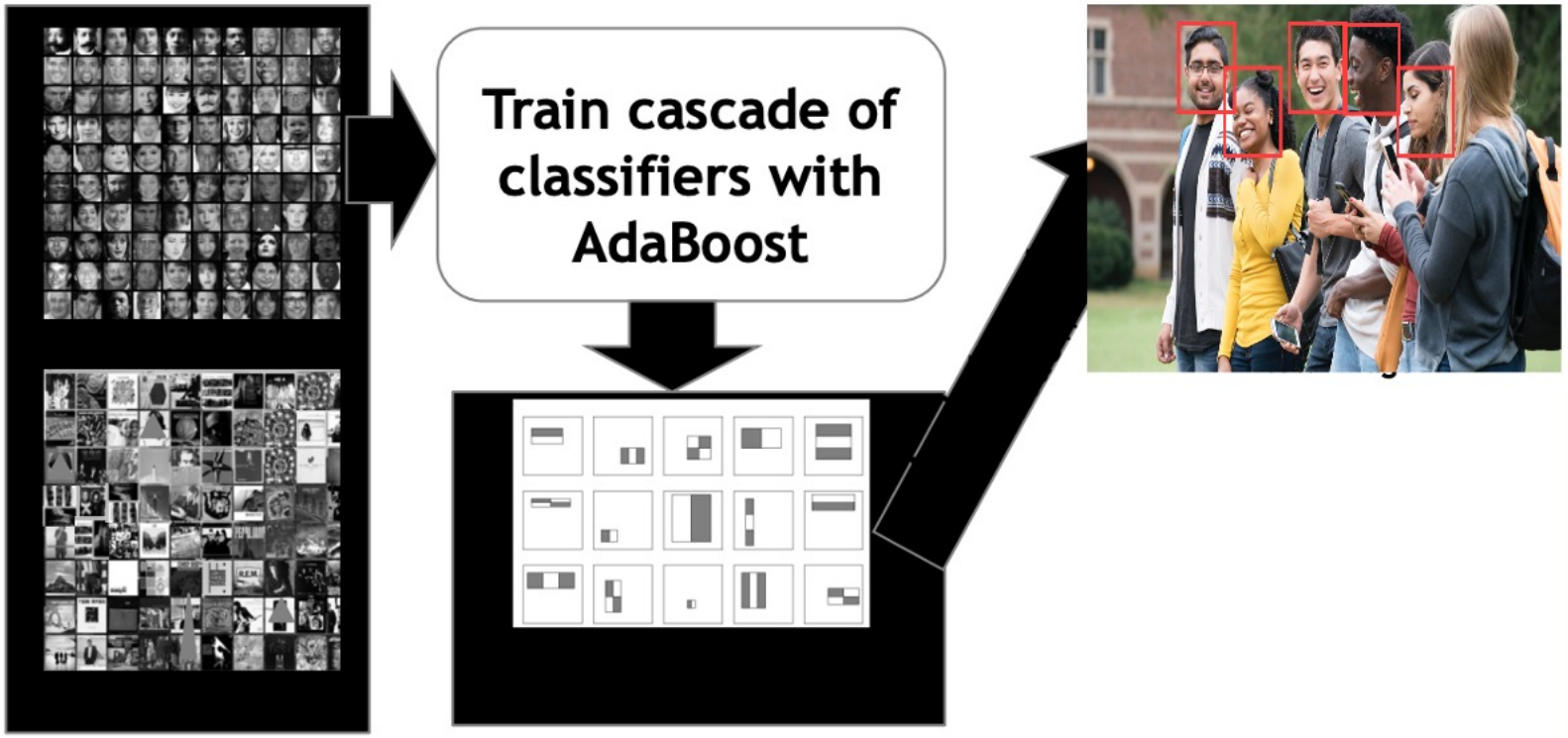
https://towardsdatascience.com/understanding-face-detection-with-the-viola-jones-object-detection-framework-c55cc2a9da14

AdaBoost Algorithm + the Cascade Classifier



In their paper, Viola & Jones (2001) mentioned that their Cascade Classifier has 38 stages (38 strong classifiers) that are made up of over 6000 features

Dicussion



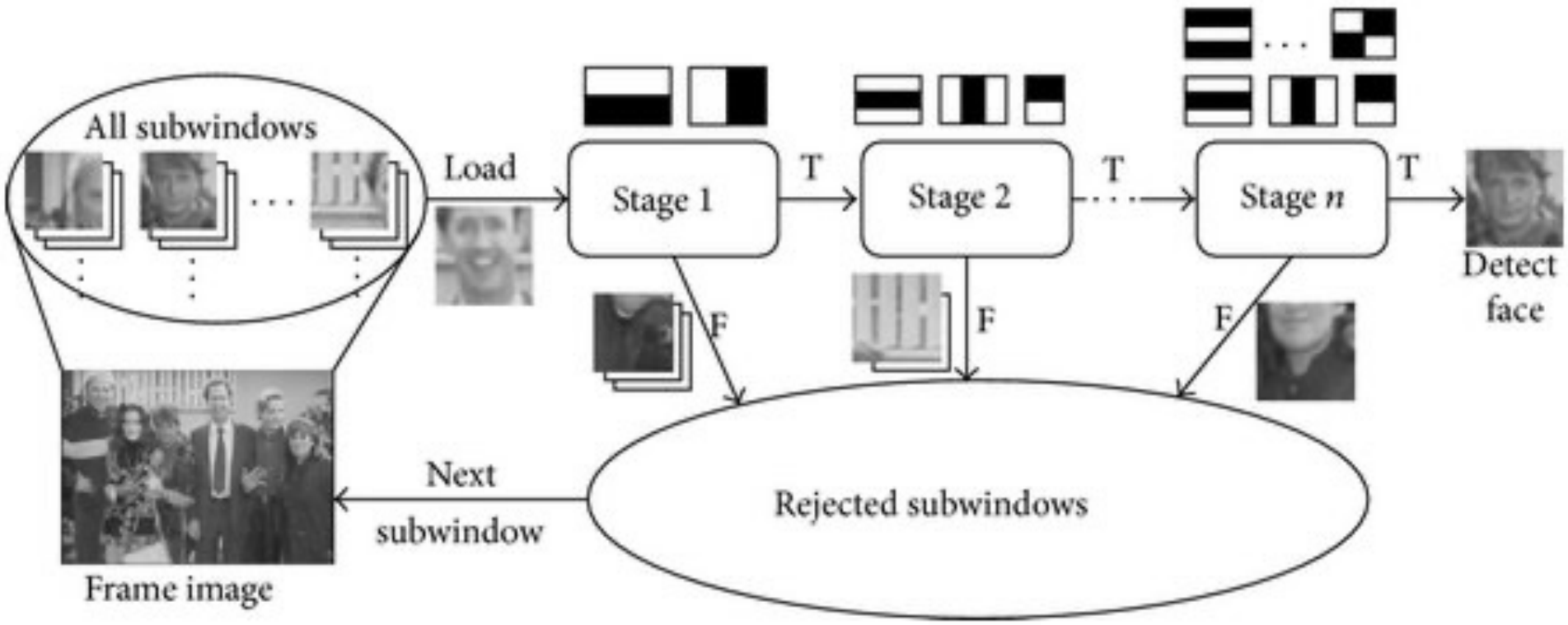
Train with 5K positives, 350M negatives

Real-time detector using 38 layer cascade

6061 features in final layer

[Implementation available in OpenCV:
http://www.intel.com/technology/computing/opencv/]

Dicussion

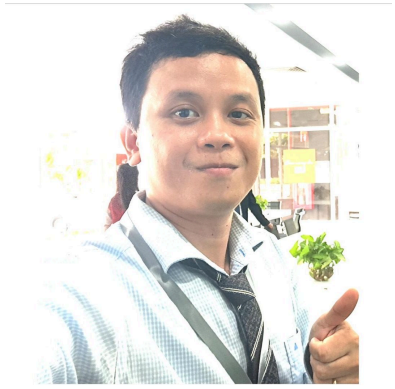

All subwindows
Load
Stage 1
T
Stage 2
T
Stage n
T
Detect face
F
F
F
Next subwindow
Rejected subwindows
Frame image

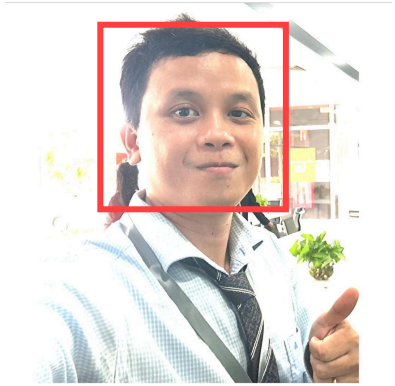
Outline

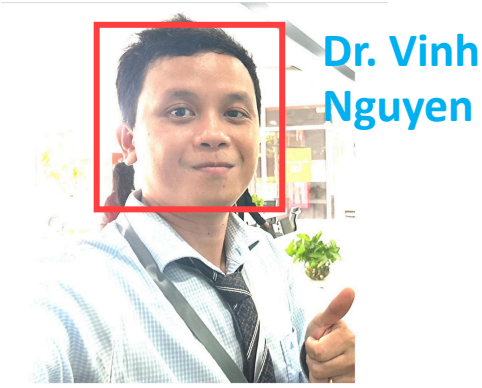
- Applications of Face Detection
- Face Detection in Computer Vision
- Haar Feature
- Integral Image
- Adaboost For Face Detection
- Cascade Classifier
- Face Recognition

Face Recognition/Identification

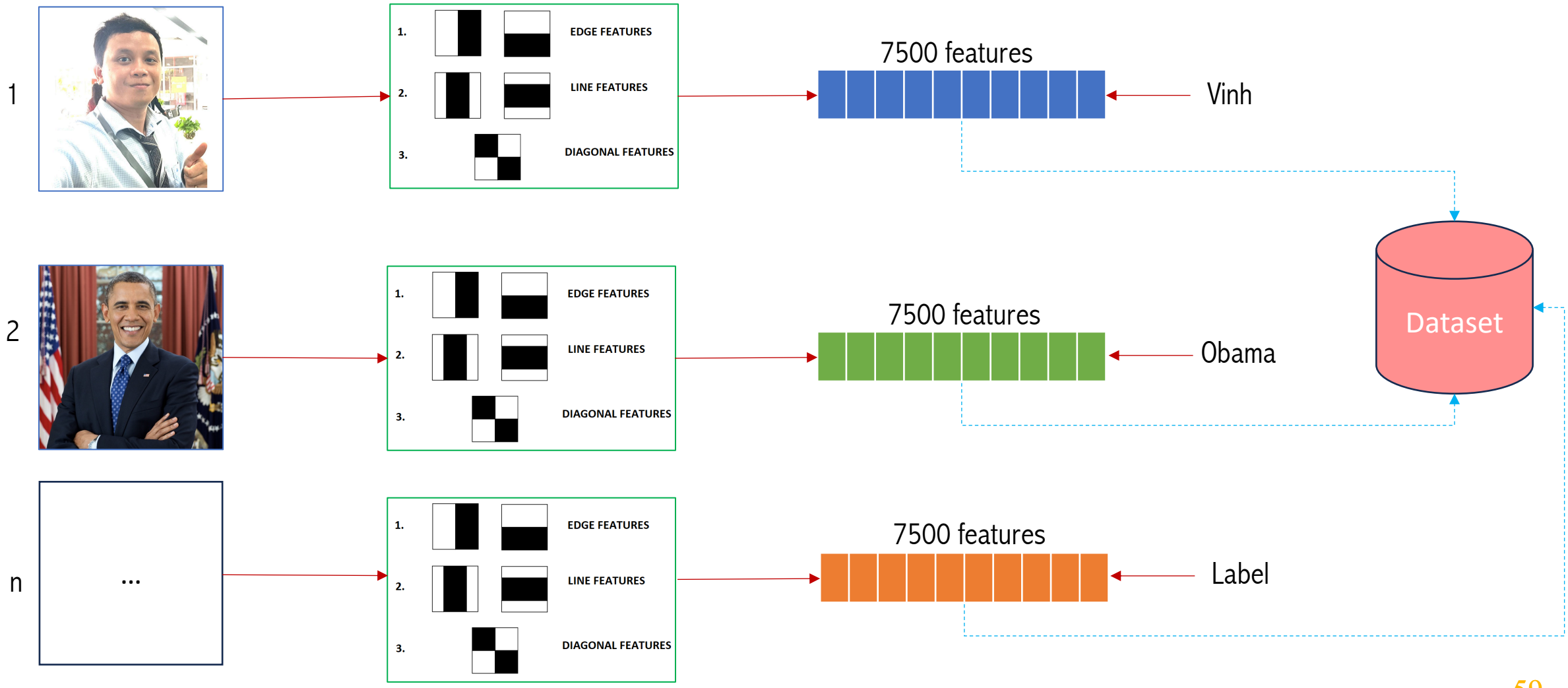
Input Image → Face Detection → Face Recognition/ Identification



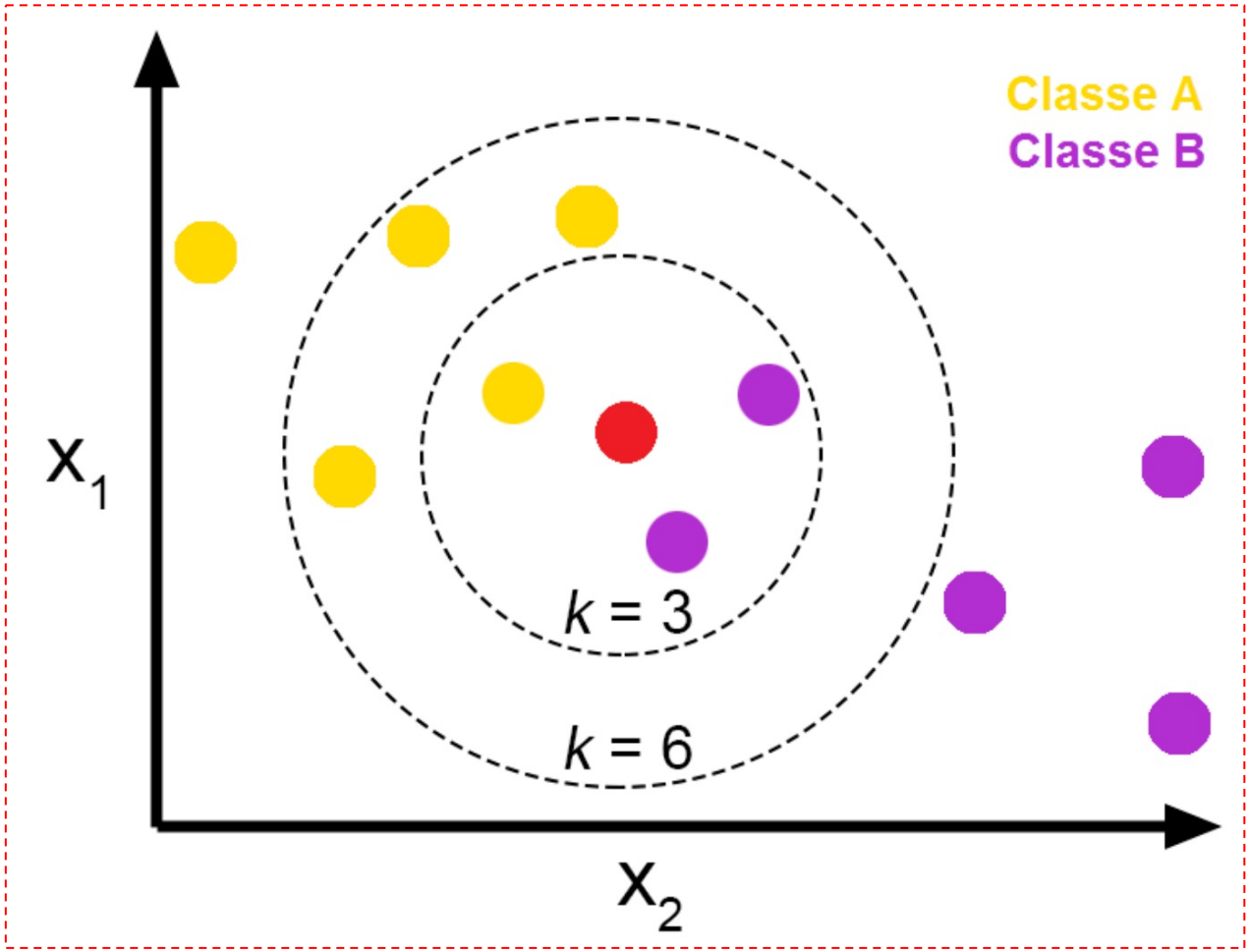




Data Preparation

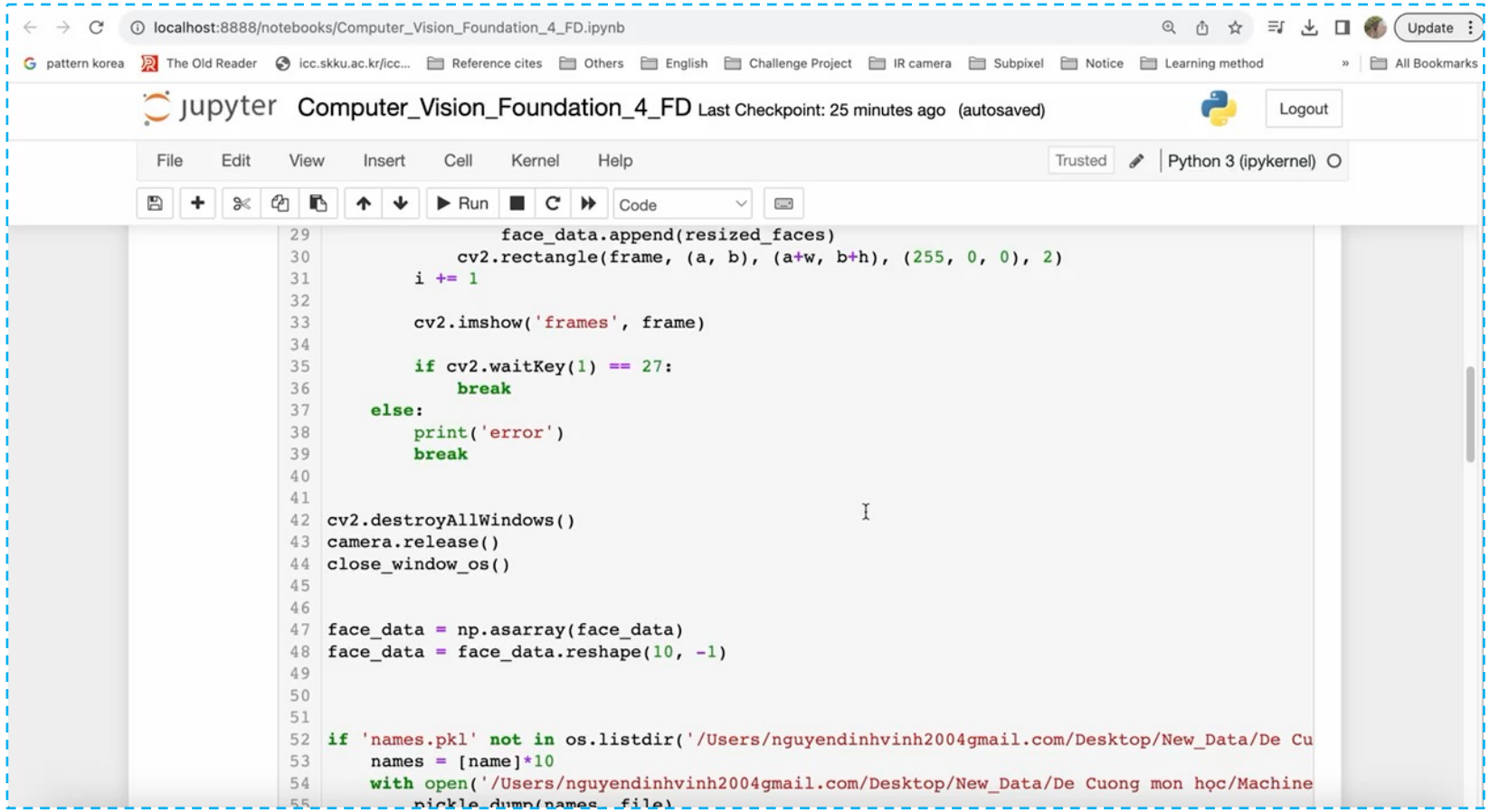


KNN Review



KNN for Face Recognition

Face Recognition with KNN

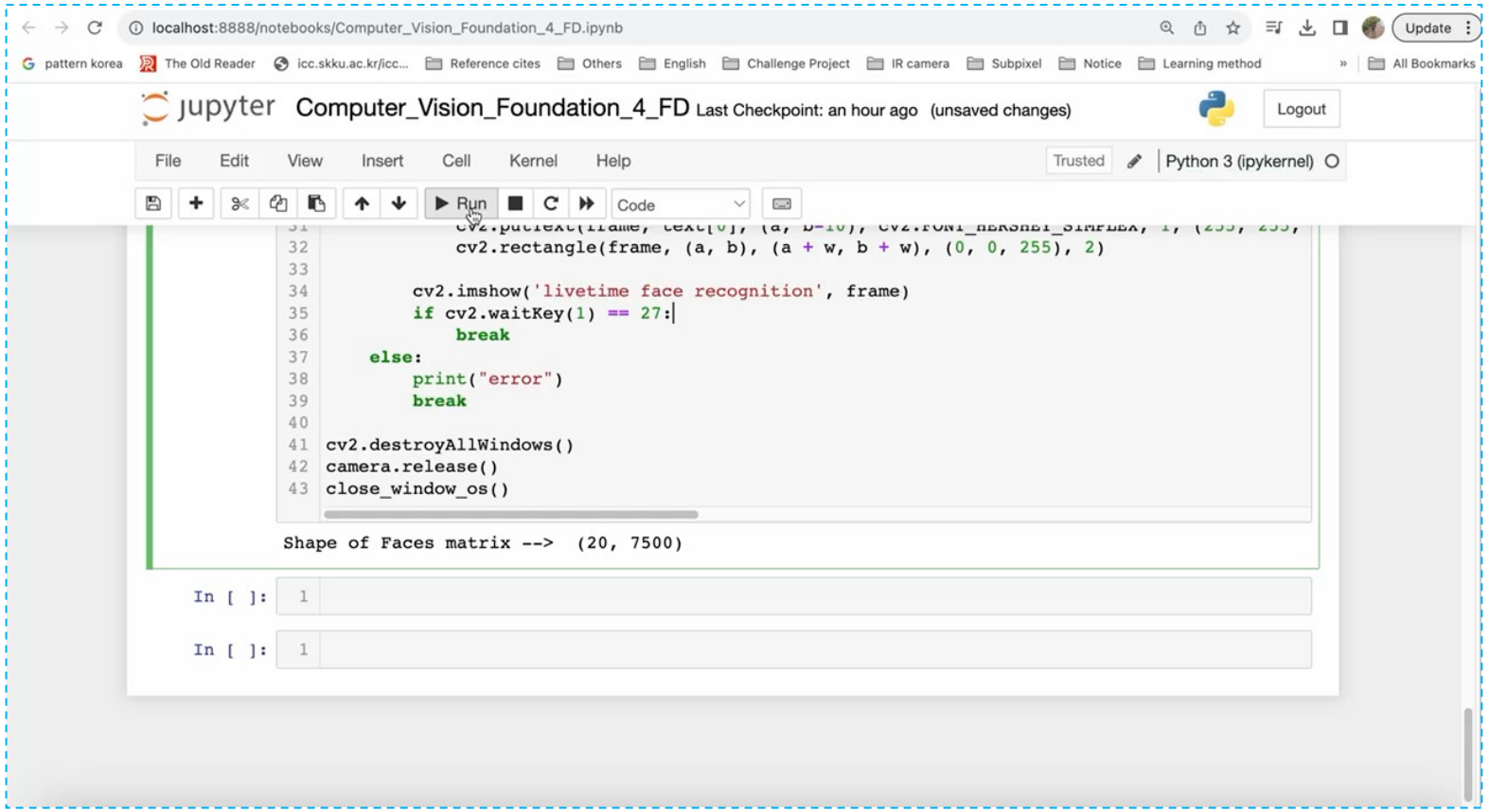


Data preparation in 10 seconds (Demo)

Collect new faces and store to the database

Play Video

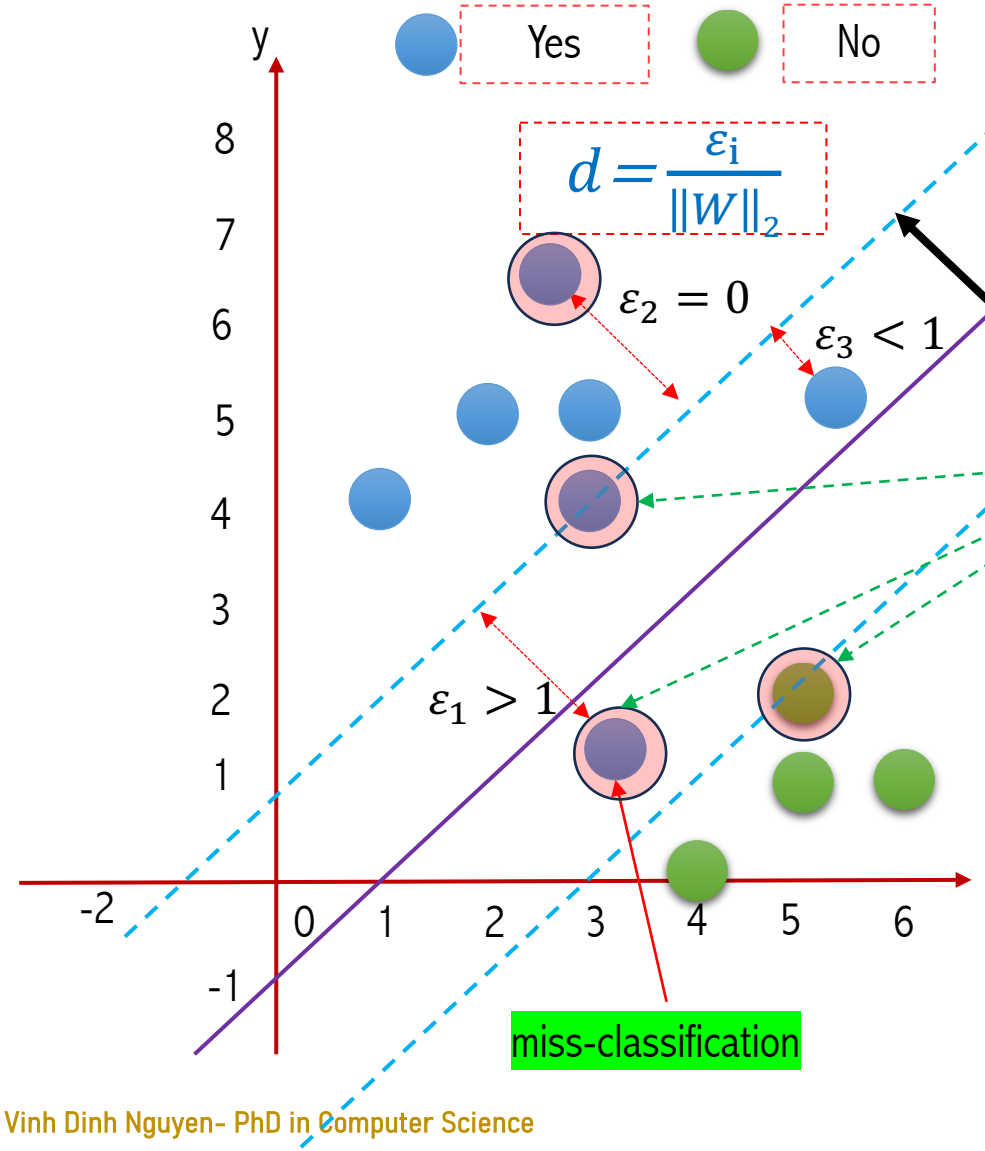
Face Recognition with KNN



Provide instant detection and indentification after 10 seconds

Play Video

SVM Review



$$d = \frac{|2|}{\sqrt{A^2+B^2}} = \frac{2}{\|W\|_2}$$

Slack variable in SVM

Support Vector

$$\text{Max } \frac{2}{\|W\|_2} \text{ such that } Y * (w^T X + b) \geq 1$$

$$\min \frac{1}{2} \|W\|_2^2 + C \sum_i^N \varepsilon_i \text{ such that } Y * (w^T X + b) \geq 1 - \varepsilon_i$$

Allow for miss classification

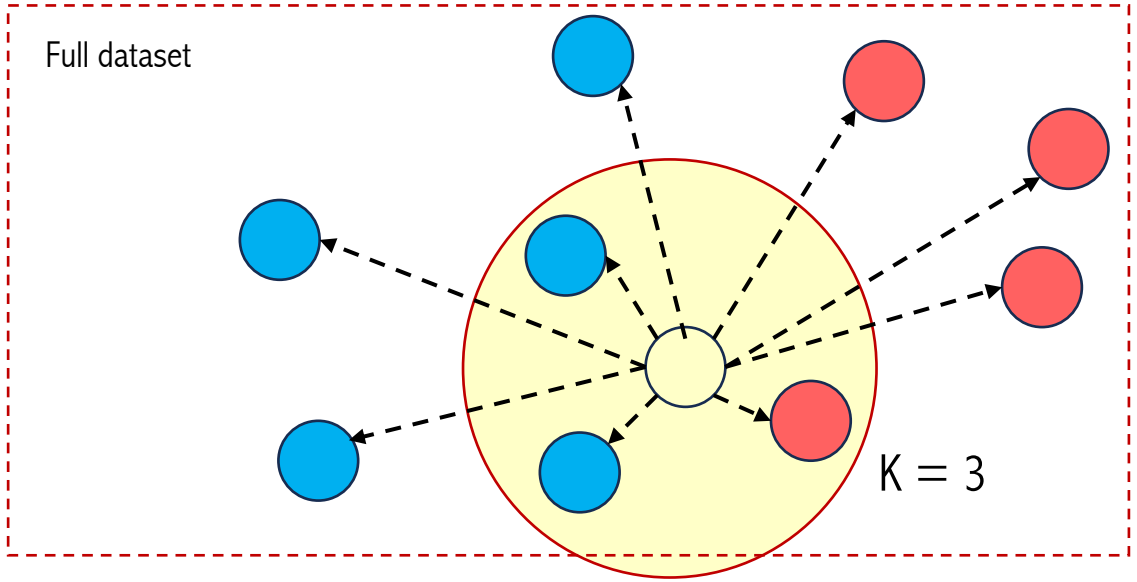
ε_i is a distance measure of the data points from their corresponding blue line.

C controls how much weight should set on the misclassification data

SVM for Face Recognition

Decision Tree Review

KNN Limitations



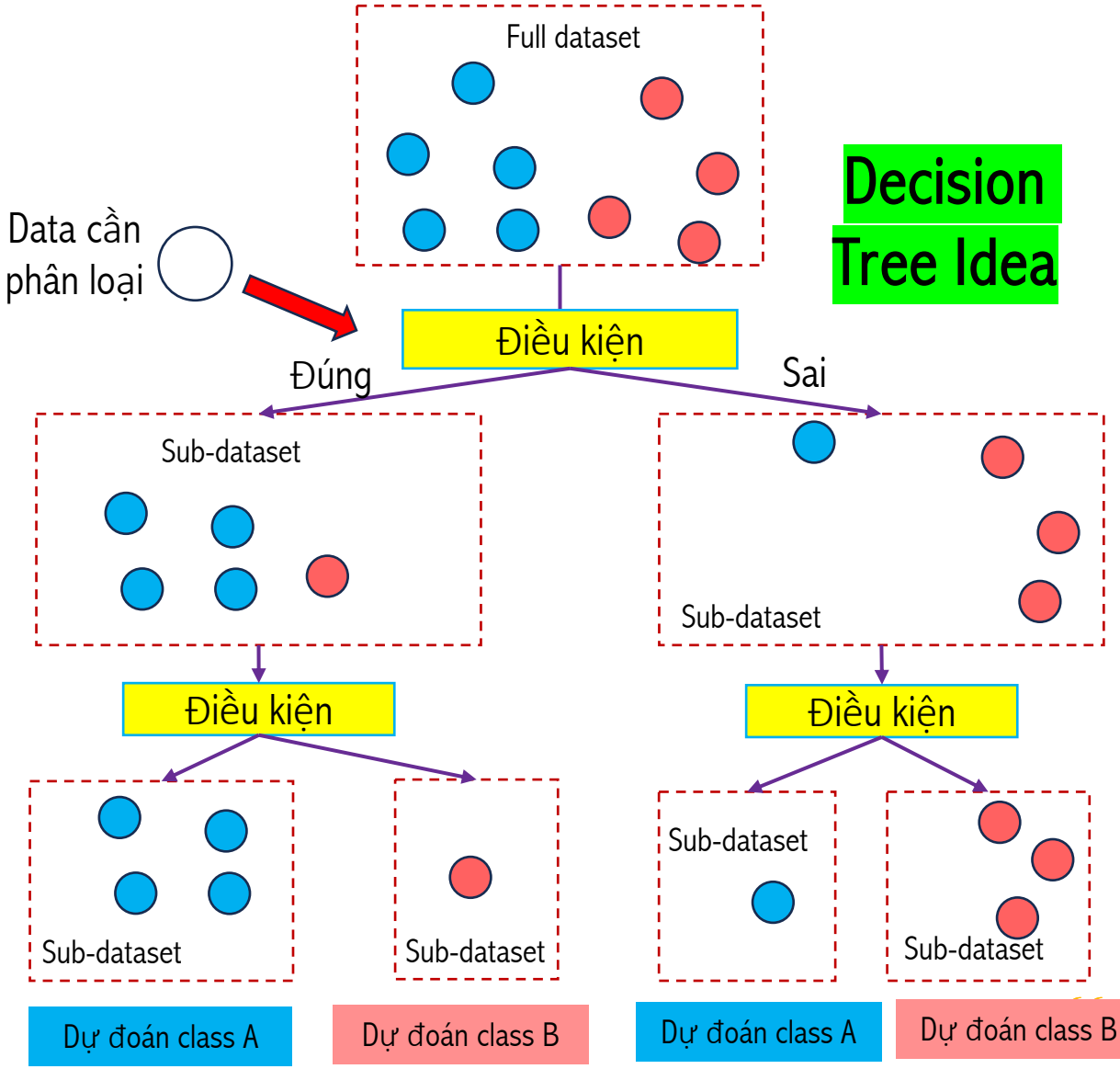
Data cần phân loại

Class A

Class B

KNN has some drawbacks and challenges, such as computational expense, slow speed, for large datasets

Decision Tree Idea



Decision Tree for Face Recognition


Thank you