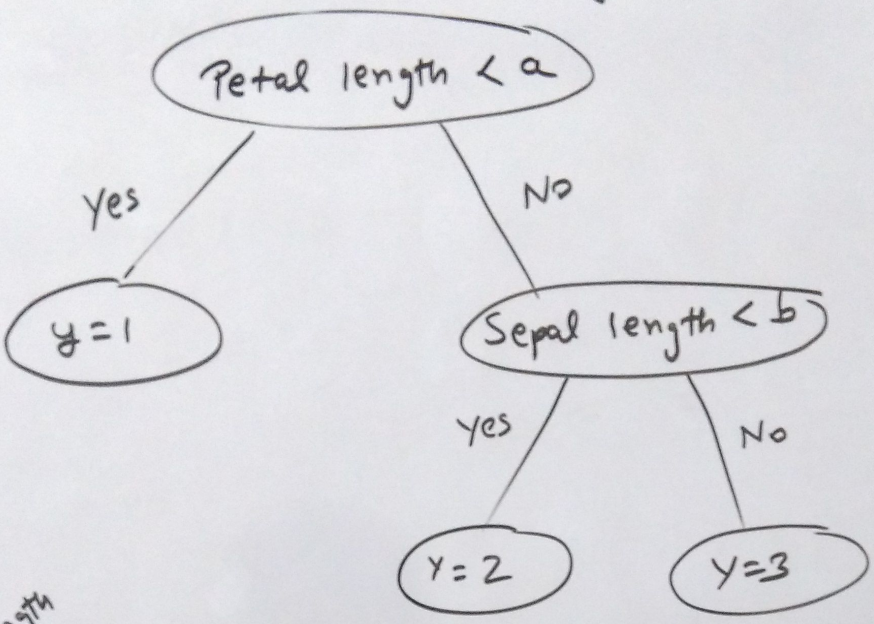
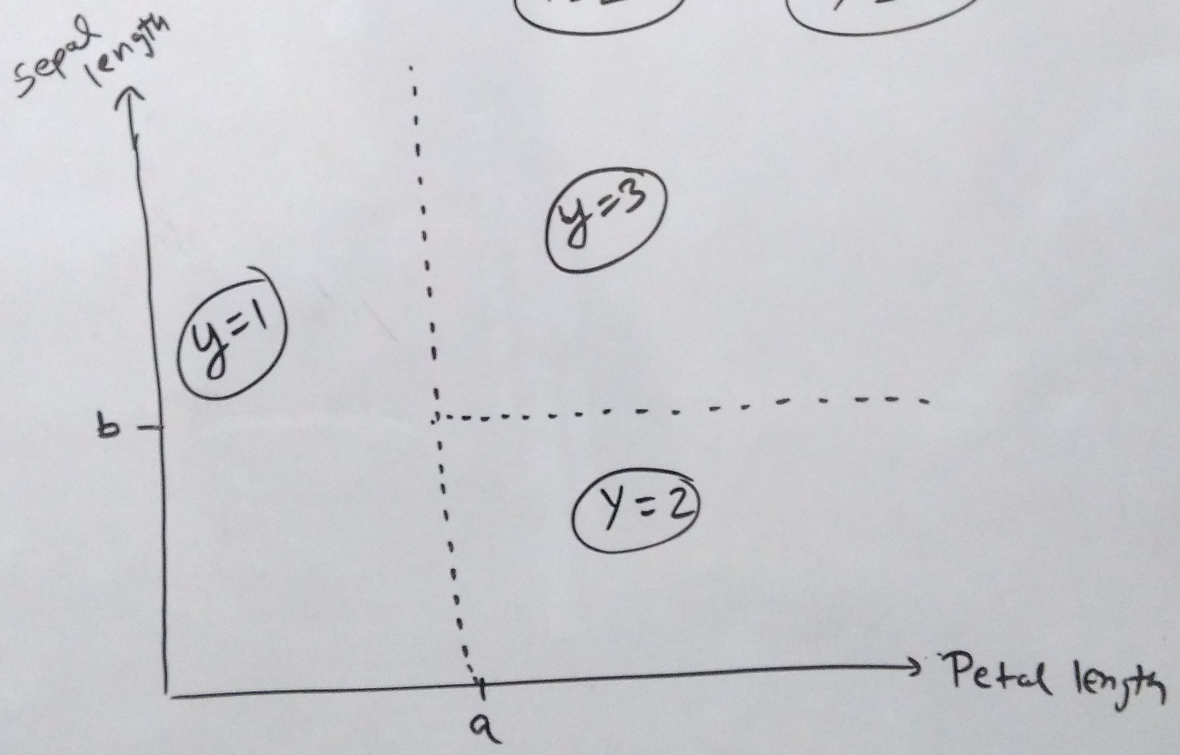
Decision tree

→ It is set of axis parallel Hyperplane.





Decision tree

Play Tennis Example

Outlook	Temp.	Humidity	Windy	Play Tennis
Sunny	Hot	High	False	No
Overcast	Mild	Normal	True	Yes
Rainy	cool	High	False	No

Entropy $H(Y)$: Randomness in target variable.

$$H(Y) = -\sum_{i=1}^{K} P(y_i) \log(P(y_i))$$

there $K=2$ because classification is Binary

$P(y_i)$ = Probabity of y_i = $\frac{y_i}{\sum y_i}$

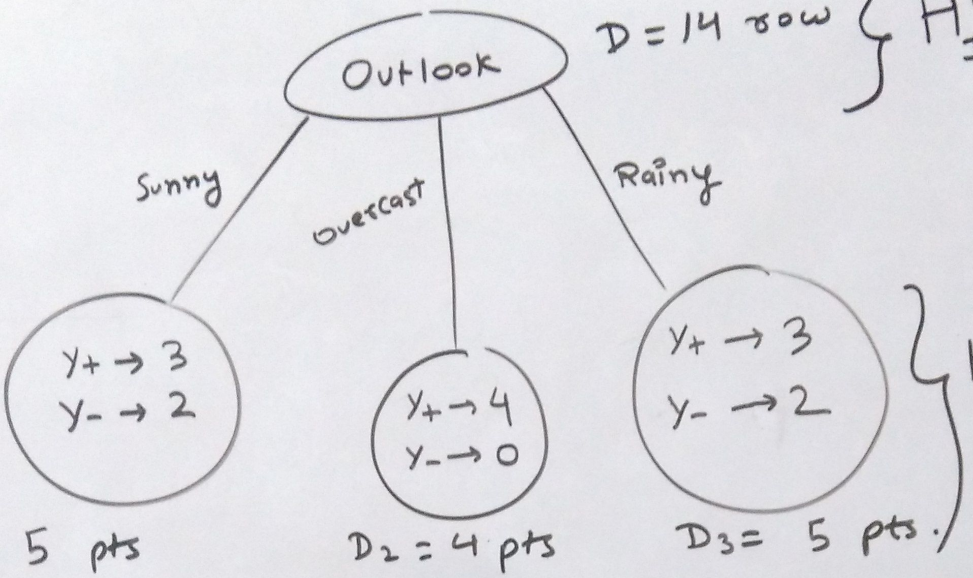
if $Y_+ = 99\%$, $Y_- = 1\%$ } $H(Y) = 0.08$ [low randomness]

if $Y_+ = 50\%$, $Y_- = 50\%$ } $H(Y) = 1$ [low randomness]

if $Y_+ = 0\%$, $Y_- = 100\%$ } $H(Y) = 0$ [No Randomness at all]

Binary $H(Y) = -\frac{C_1}{C_1+C_2} \cdot \log\left(\frac{C_1}{C_1+C_2}\right) - \frac{C_2}{C_1+C_2} \log\left(\frac{C_2}{C_1+C_2}\right)$

Decision tree



$D = 14$ row $\Rightarrow H_D^1(Y) = 0.94$

$H_D^2(Y) = ?$

$D_1 = 5$ pts $\quad D_2 = 4$ pts $\quad D_3 = 5$ pts.

$H_{D_1} = 0.97 \quad H_{D_2} = 0 \quad H_{D_3} = 0.97$

Entropy at level 2

$$H_D^2(Y) = \frac{D_1}{D} \cdot H_{D_1} + \frac{D_2}{D} \cdot H_{D_2} + \frac{D_3}{D} \cdot H_{D_3}$$

Information Gain

$$IG(Y, \text{Outlook}) = H_D^1(Y) - H_D^2(Y)$$

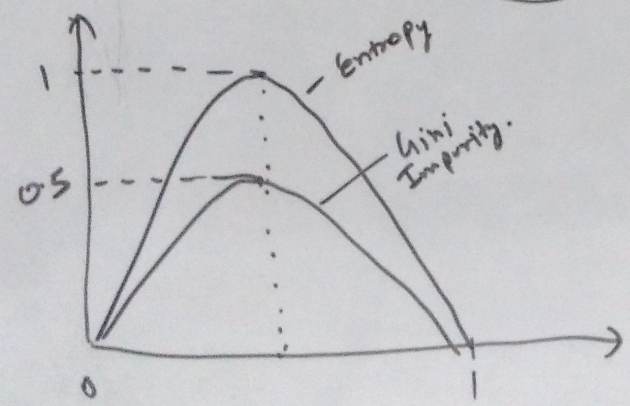
(want to maximize this)

Gini Impurity (Same as Entropy) - Easy computation

$$I_G(Y) = 1 - \sum_{i=1}^{K} (P(y_i))^2$$

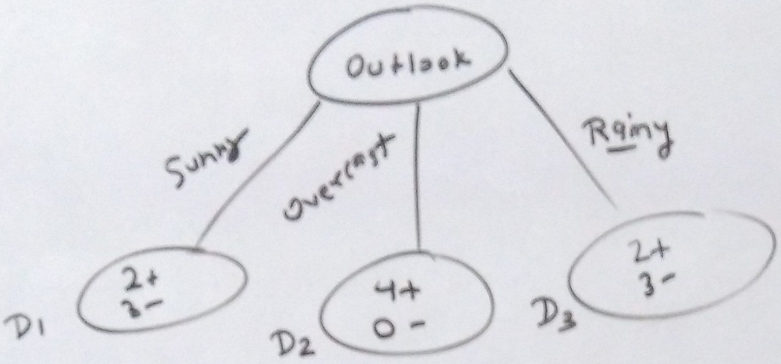
$\Rightarrow P(Y_+) = 0.5$
$P(Y_-) = 0.5$
$I_G(Y) = 0.5$

$\Rightarrow P(Y_+) = 1$
$P(Y_-) = 0$
$I_G = 0$



Decision tree

Construction :



(1) Choose Node with Max IG

find
- IG(Y, Outlook)
- IG(Y, Temp.)
- IG(Y, Humidity)
- IG(Y, Windy)

(2) Break Nodes till :

- we get Pure Node (all +ve or all -ve)
- very few pts in Node (lead to catch Noise and Overfit)
- If we are too deep. (set max_depth)

Decision tree

Construction :

Handling and Splitting of Numerical feature.

feature (f_i)	Y
2.2	1
2.6	1
3.5	0
3.8	0
4.6	1
5.3	0

- Sort Numerical feature in asc. order.

$f < 2.2 \rightarrow IG$
$f < 2.6 \rightarrow IG$
$f < 3.5 \rightarrow IG$
$f < 3.8 \rightarrow IG$
$f < 4.6 \rightarrow IG$
$f < 5.3 \rightarrow IG$

} Choose condition which give Max. Information Gain (IG_i)

Handling Category feature with many level like Zipcode → Convert into numerical.

Pincode	$Y = \{0,1\}$	$P(Y=1 \mid P_i)$
P_1	0	0.253
P_2	1	0.01
P_3	0	0.001
⋮	1	0.65
⋮	1	

Decision tree

Complexity : Train time complexity

for Breaking one Node [let if all feature is numerical.

$f_1, f_2 - f_d$

f_1 ↓ sort $(n \log n)$; f_2 ↓ sort $(n \log n)$; f_d ↓ $(n \log n)$

$(n \log n) \; (n \log n) \cdots (n \log n) = \boxed{O(n \log n d)}$

↓

Global NP hard indeterministic problem

Test time $= \boxed{O(\text{Max\_depth})}$

Depth ↓ ; Underfit ; interpretability ↑

Depth ↑ ; Overfit ; interpretability ↓

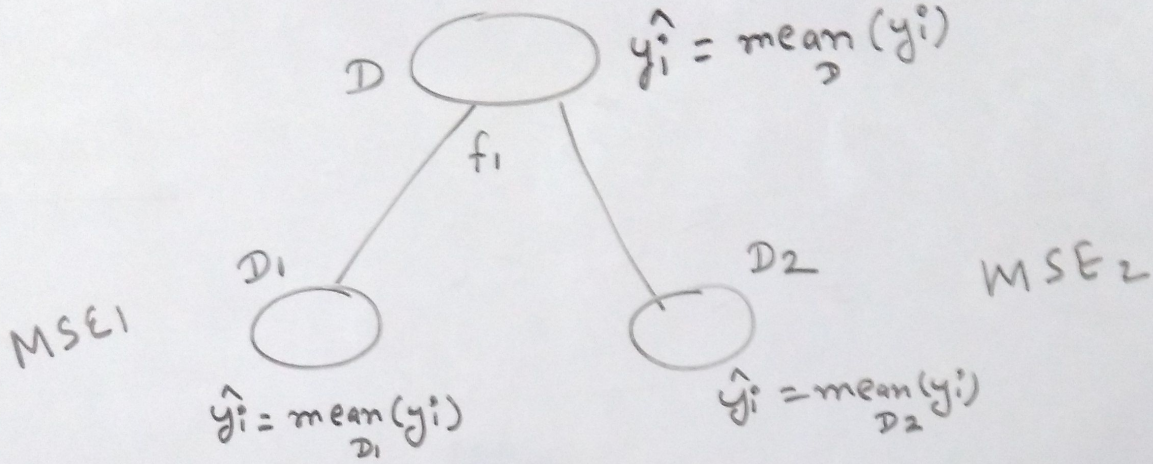
Hyperparameter :- Depth of tree

- Min. point to break Node

Decision tree

Regression

Classification → diff. of entropy / IG
Regression → diff of MSE / MAE



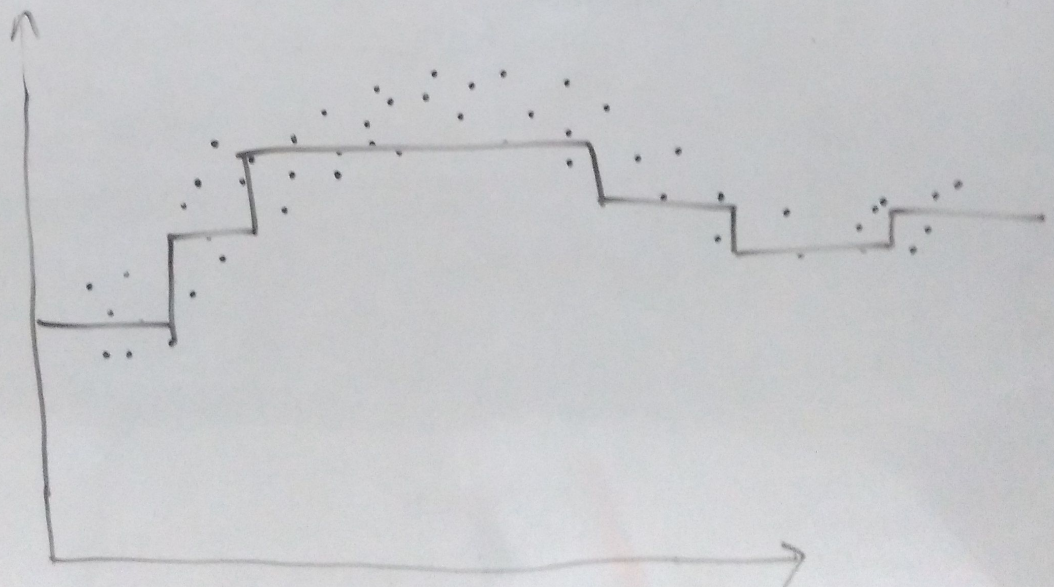
Level 1 $\boxed{MSE^1 = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2}$

Level 2 $MSE^2 = \left(\frac{D1}{D} \cdot MSE_1 + \frac{D2}{D} \cdot MSE_2\right)$

or

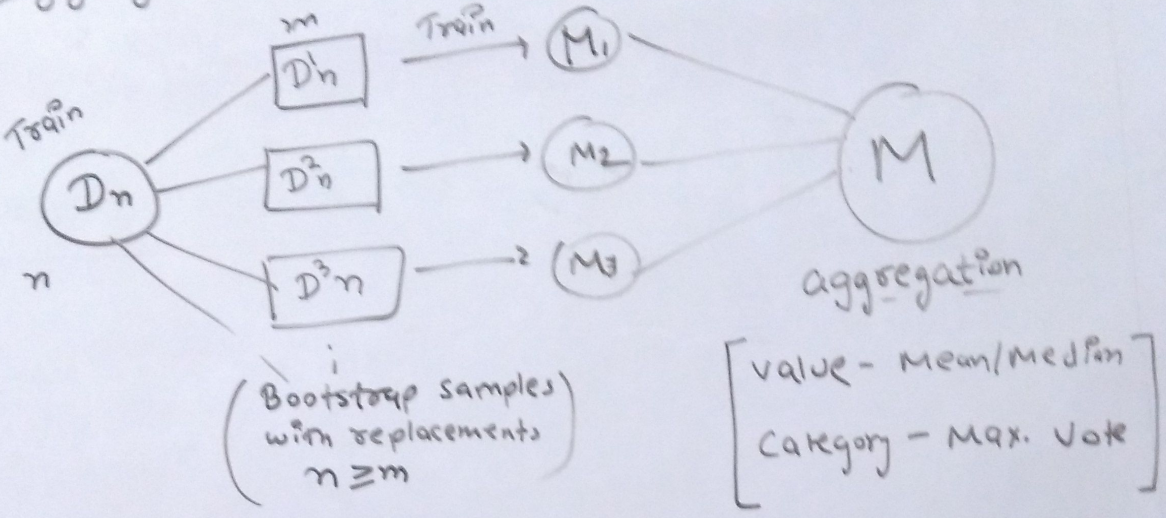
$\boxed{MSE = \omega_1 \cdot MSE_{D_1} + \omega_2 \cdot MSE_{D_2}}$

Chose feature to maximize : $(MSE^1 - MSE^2)$



Decision tree - Ensembles

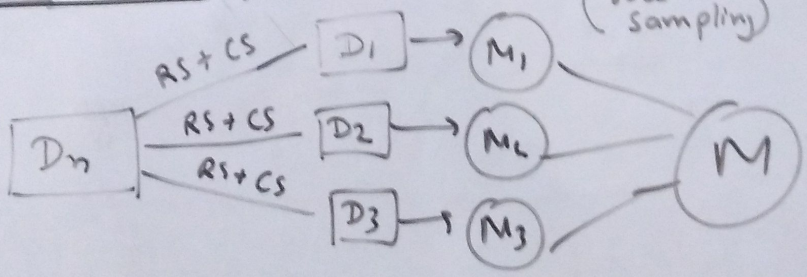
Bagging (Bootstrapped Aggregation)



Model Error = $Bias^2$ + Variance + constant

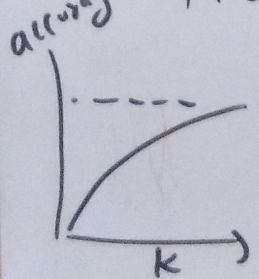
(Help in Reducing Variance)

Random Forest = Decision tree + Bagging + Column Sampling
(row sampling) (feature Bagging)



Each model $M_1, M_2 \cdot M_k$ have high variance.

$M = agg(M_1, M_2, M_3 \cdots M_K)$



K↑ ; Variance↓

K↓ ; Variance↑

$D_n \xrightarrow[CS\ (50\%)]{RS\ (20\%)} D_n'$

CSR ↓ ; Variance↓
RSR

Train $O(n \log n \cdot K)$

Test $O(K \cdot depth)$

Space: $O(\#T \times K)$

Boosting

D_{train} $\{x_i, y_i\}_{i=1}^{n}$ $\longrightarrow$ M_0 (high bias Model)

$y = h_0(x)$

error = $y_i - h_0(x)$ [Residual error]

Stage 1:

$\{x_i, error_i\}_{i=1}^{n}$ $\longrightarrow$ M_1

$y = h_1(x)$

Model at the end of stage 1 = $F_1(x)$

$$F_1(x) = \alpha_0 h_0(x) + \alpha_1 h_1(x)$$

error = $y_i - F_1(x)$

Stage 2:

$\{x_i, error_i\}_{i=1}^{n}$ $\longrightarrow$ M_2

$y = h_2(x)$

$$F_2(x) = \alpha_0 h_0(x) + \alpha_1 h_1(x) + \alpha_2 h_2(x)$$

or

$$F_2(x) = F_1(x) + \alpha_2 h_2(x)$$

Stage K

$$F_K(x) = \sum_{i=0}^{K} \alpha_i h_i(x)$$

Boosting

Residual, loss & Gradient

$$F_K(x) = \sum_{i=0}^{K} \alpha_i h_i(x)$$

at the end of K stage/iteration

residual = error$_i$ = $y_i - F_K(x)$

if Regression Problem:

Loss: $L(y_i, F_K(x)) = (y_i - F_K(x))^2$

$$\frac{\partial L}{\partial F_K(x)} = \frac{\partial}{\partial F_K(x)} \left[(y_i - F_K(x))^2 \right]$$

let $F_K(x) = z$

$$\frac{\partial L}{\partial z} = -2(y_i - z)$$

$$-\frac{dL}{\partial z} = 2(y_i - z)$$

negative gradient $\simeq$ residual

here $-\frac{\partial L}{\partial z}$ is known as pseudo residual.

[gt have ability to penalize any differentiable loss function.]

$$\text{error}_i = \underbrace{y_i - F_K(x)}_{\text{residual}} \xrightarrow{\text{replace by}} \underbrace{2(y_i - F_K(x))}_{\text{pseudo residual.}}$$

$\{x_i, \underbrace{\text{error}_i}_{\text{Pseudo}}\}$

Gradient Boosting algorithm

Input: $\{x_i, y_i\}_{i=1}^{n}$;

$L(y, F_x(x))$: A differentiable loss function.

M : No of iteration

1.) Initialize the Model with some constant:

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

here γ is mean (y_i) for regression.

2.) for $m = 1$ to M.

- Compute Pseudo Residuals:

$$r_{im} = \text{error}_i = -\left[\frac{\partial L(y_i, F_{m-1}(x_i))}{\partial F_{m-1}(x_i)}\right]$$

- Fit a Base learner $h_m(x)$ to psuedo residual train using $\{x_i, r_{im}\}_{i=1}^{n}$

- Compute Multiplier γ_m by optimization problem:

$$\gamma_m = \text{argmin}_{\gamma} \sum L(y_i, F_{m-1}(x_i) + \gamma \cdot h_m(x))$$

- Update Model

$$F_m(x) = F_{m-1}(x) + \gamma_m \cdot h_m(x)$$

3.) Output Final Model.

Gradient Boosting

Regularization :

GBDT are prone to overfitting, so add shrinkage coefficient (ν)

$$F_m(x) = F_{m-1}(x) + \nu \cdot \gamma_m h_m(x)$$

$0 \leq \nu \leq 1$ $\therefore$ default = 0.1

$\Rightarrow$ GBDT Hyperparameter M and ν.

Train : $O(n \log n \cdot d * M)$

M = No. of Base learner.

Test : $O(\underbrace{\text{depth}}_{\text{Small}} * M + \underbrace{M}_{\text{add all tree.}}) \approx O(\text{depth} \times M)$

Space : $O(\text{each tree} + \underbrace{\text{each } \gamma}_{\text{store M No. of } \gamma})$.

Adaboost . upsample wrong classified example. than train again.

$$F_K(x) = \alpha_1 \cdot h_1(x) + \alpha_2 h_2(x) + \alpha_3 h_3(x)$$