Ensemble :- Collection / grp of things

Used in conditions when we have multiple models being used together to build a ~~p~~ more powerful model.

There are 4 types of models :-

i) Bagging (Bootstrapped Aggregation)

ii) Boosting
iii) Stacking
iV) Cascading

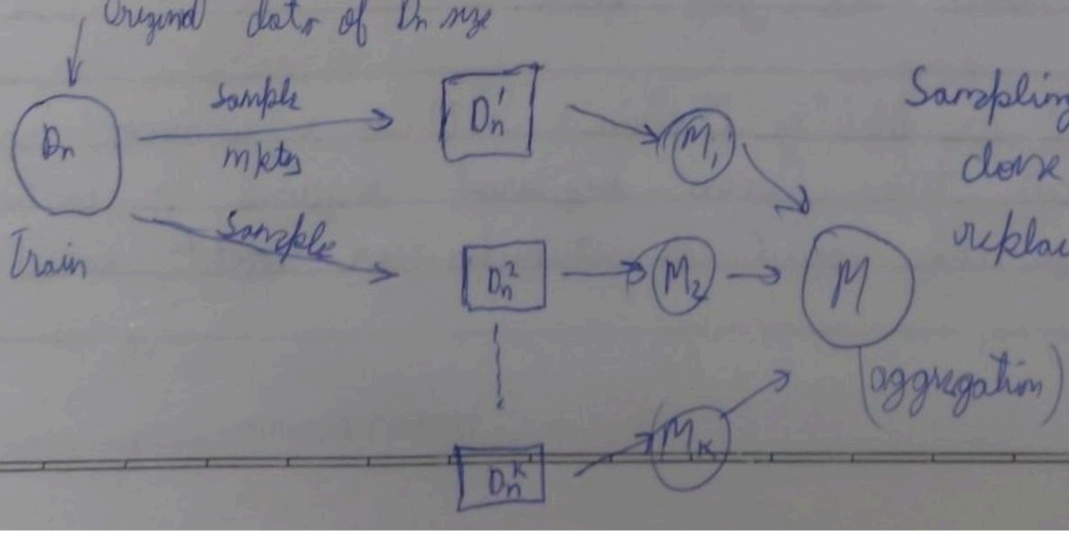
* Key aspect :-

$$\underbrace{M_1, M_2, M_3 \ldots\ldots M_K}$$

The more different these models are, the better you can combine them.

<u>Bagging</u> (Bootstrap Aggregation)

Popular Bagging model :- Random Forest.



Sampling is done with replacement

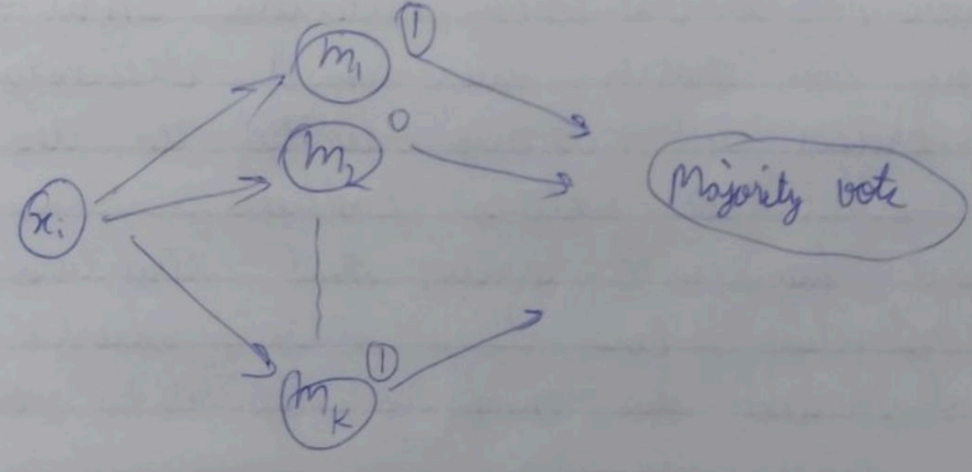
Each model M_i is built using D_n^i of size m $(m \leq n)$

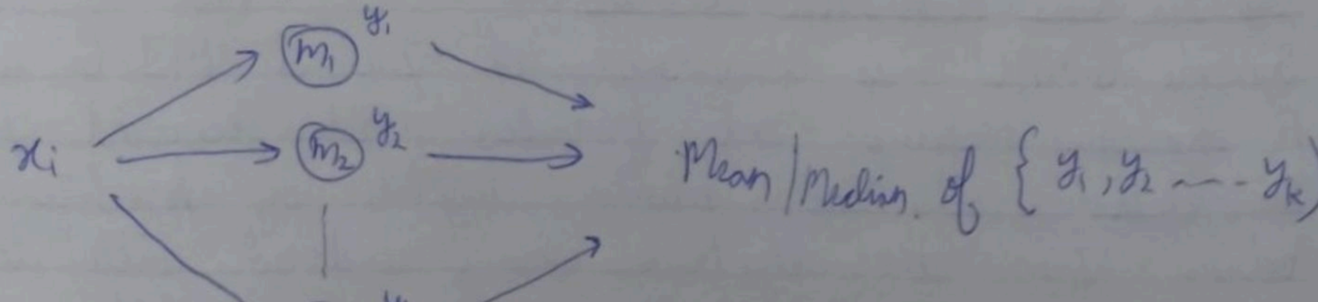
$\Rightarrow$ each model M_i has seen a different ~~subset~~ sample of data.

So, each model is Aggregated to create a one powerful model.

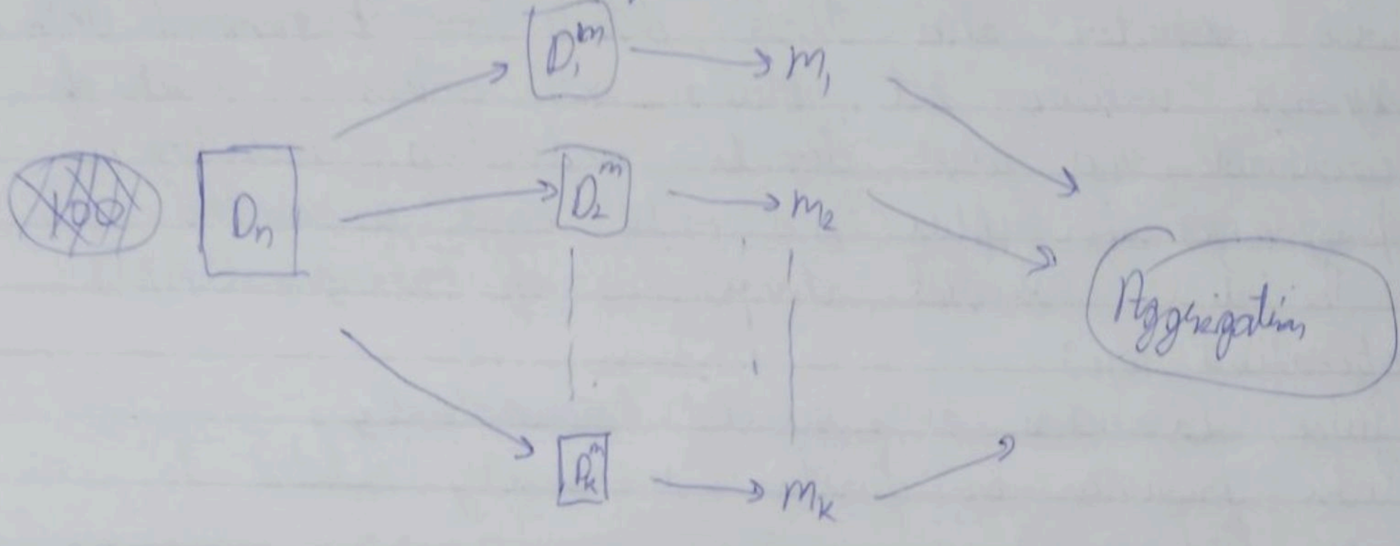
Aggregation :- Classification :- Majority vote
Regression :- Mean/Median

So if we get a new datapoint x_i, it will go to all the base models.





If model changes a lot with ~~...~~ changes in training data it is said to have high variance.



Now, suppose I change or just remove 100 pts from my ~~traing~~ training data so due to this out of K samples ~~...~~ only small no. of samples would be impacted due to which only small no. of Base models would be impacted due to which the overall aggregated model (with majority voting or mean) won't be impacted much. which shows that our final model won't change much

Hence, from here we can understand that Bagging with change in training data i.e, it has low variance

Hence from here we can understand that Bagging can reduce the variance of a model without impacting the Bias.

$$\text{model error} = \text{Bias}^2 + \text{Var}$$

Now, if we have a base model with low bias & high variance then using bagging on such multiple models would provide a final model with low bias and reduced variance.

↓ Core Idea of Bagging

Random Forest (Bagging)

Random → "Random" Bootstrap Sampling

Forest → Several Decision Trees

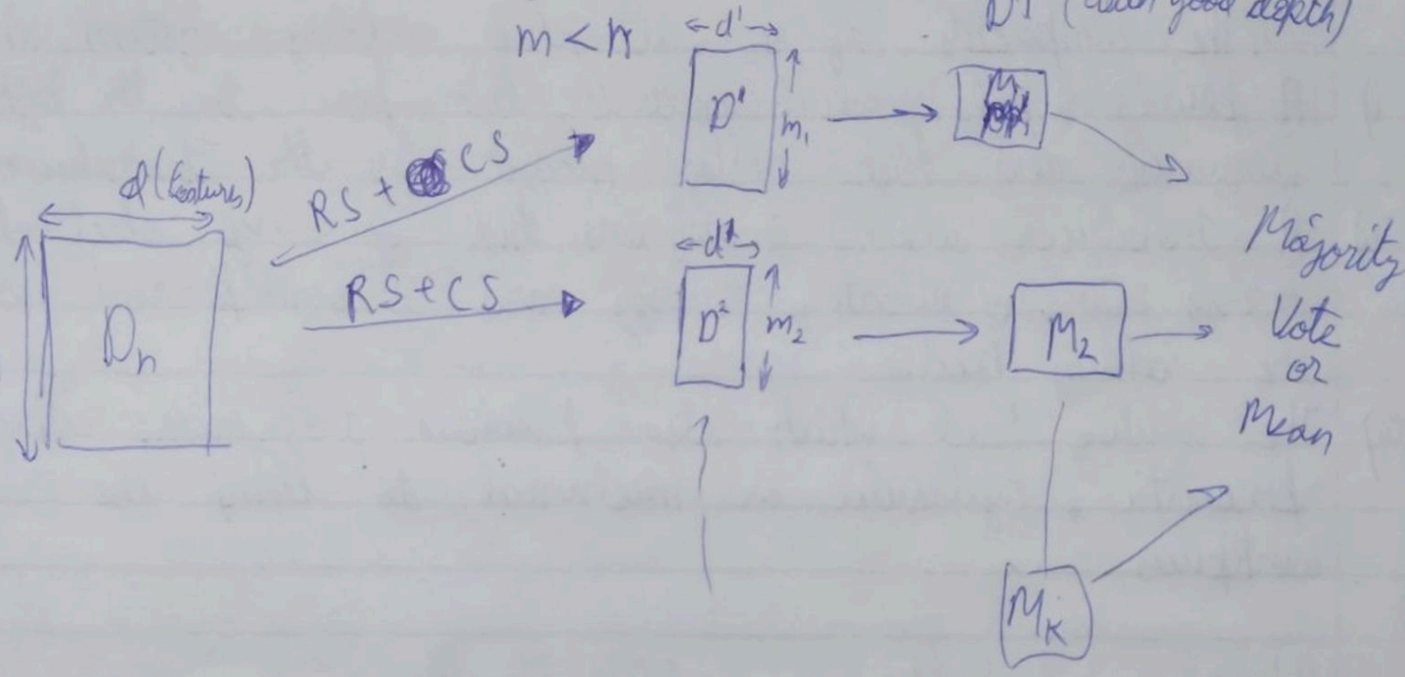
$$RF :- \underbrace{\text{Decision Trees}}_{\text{Base model}} + \text{Bagging} + \underbrace{\text{Col. Sampling}}_{\text{Feature Bagging}}$$

Bootstrap Sampling :- Row Sampling

Feature Bagging :- Column Sampling

$d' < d$

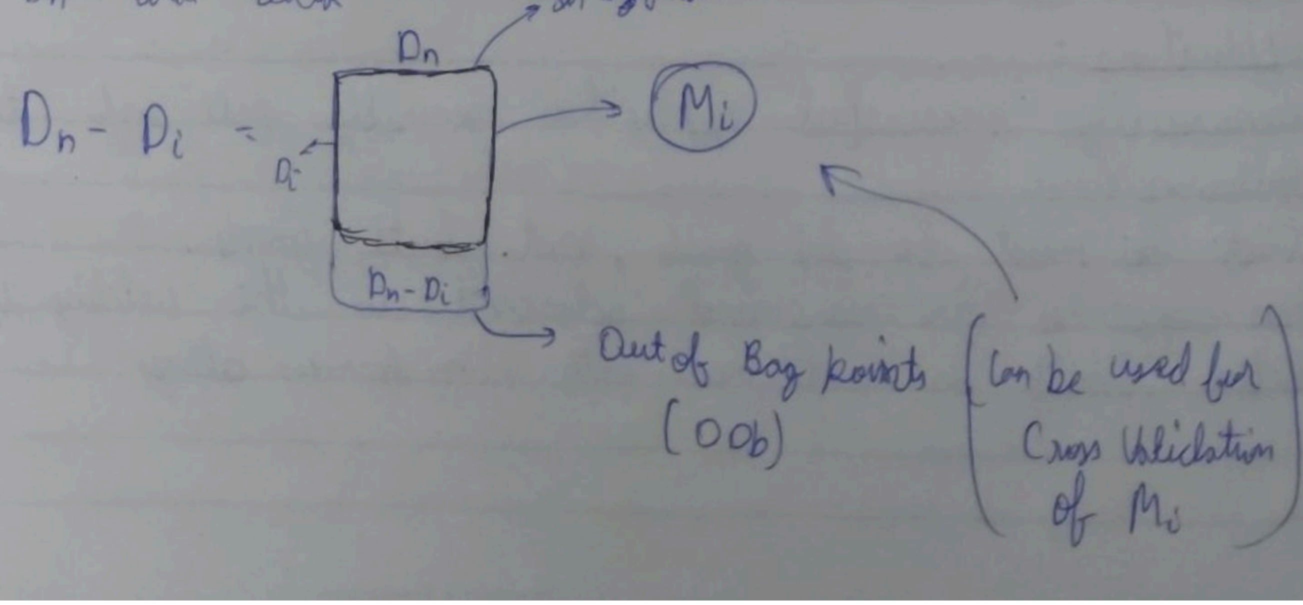
$m < n$



So considering my first base model as M_i

M_i gets trained on D' data having d' cols and m rows

$D_n \rightarrow$ Total data



Having reasonable depth

RF : DT base learner

+ Row Sampling with replacement
+ Col. Sampling
+ Aggregation (Majority Vote/ Mean or Median)

Bias - Variance Tradeoff

RF :- Reduce Variance

→ Low Bias Because base learners (M_i's) are low-bias.

→ Variance

final model

$$M = Agg(M_1, M_2, M_3, \ldots \to M_k)$$

So , $K\uparrow$, Variance $\downarrow$

$K\downarrow$, Variance $\uparrow$

Bias of M $\approx$ Bias of M_i

$\uparrow$ Final model

$\downarrow$ Base model

Other than K (No. of Base Learners) there is one more hyperparameter on which the variance depends :-

$D_n \xrightarrow[CS]{RS} D'$

$d' < d$

$m < n$

$\left(\frac{d'}{d} = \text{Col. Sampling Ratio} \right)$

$\left(\frac{m}{n} = \text{Row Sampling Ratio} \right)$

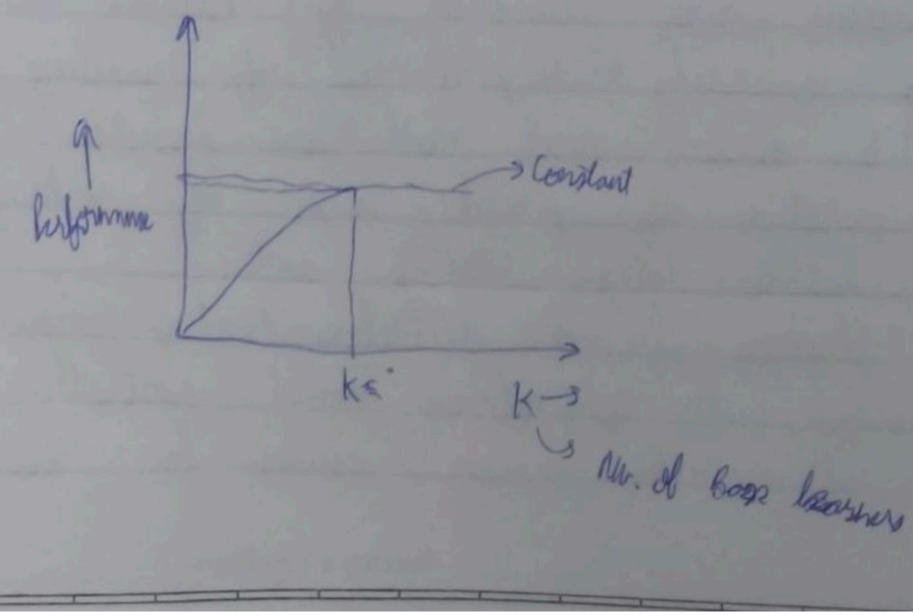
If Col. Sampling Ratio ↓ & Row Sampling Ratio ↓ → Each sample would be more different from others → (~~Change~~ Impact of change in training data would ↓ → Low variance

↓

But we can't keep these ratios too small that we don't have any data left to train our model.

D_n → Training data → D_1 — M_1

→ D_2 — M_2

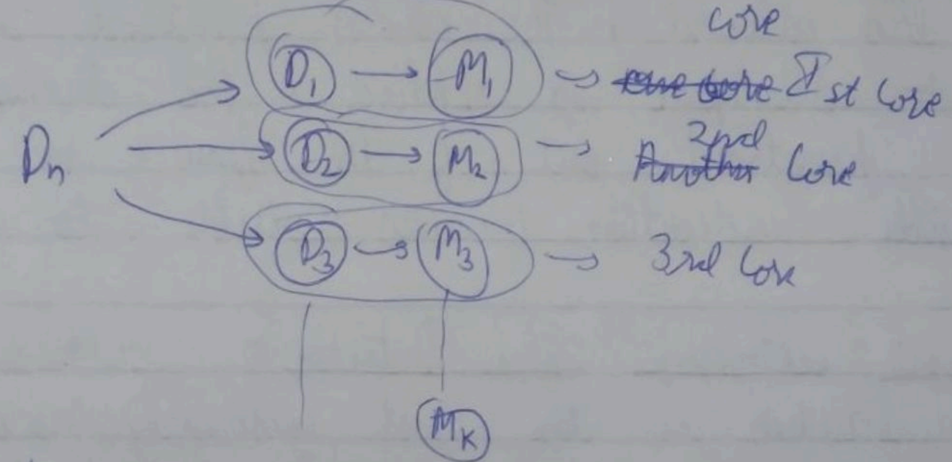


Train + Run Complexity

RF with ~~cod~~ K base learners (DT)

Train Time Complexity :- $O(n \log(nd) * k)$

If we have a multi core devices then ~~then~~ each base model can be trained in ~~the~~ separate core



That's why it's also called

Trivially Parallelizable

↓

Multiple models can be trained parallely

Run time Complexity :- $O(depth * k)$

Space Complexity :- $O(DT * k)$

Random Forest Hyperparameters:-

n_estimators = 10 // no. of Base learners
max_depth = None // Putting no limit of depth.
bootstrap = True // To ~~get~~ take bootstrap sample.
oob_score = ~~Fal~~ False // Will give Cross Validation Score of each DT if becomes True.
n_jobs = 1 // Scikit-learn will train each DT on ~~the~~ all CPU Core parallely if becomes -1.

Extremely Randomized Trees

It is almost similar to RF except one more Randomization factor.

In case of RF, to decide threshold for a column, all the values are first sorted and checked.

But in extremely randomized trees, only certain random samples are taken to check the threshold to split node of tree.

Extreme Trees :- Col. Sampling + Row Sampling + Agg + Randomization while Selecting threshold

This led to more uniqueness in Base models.

Reduce Variance Better than RF.

8.) Random Forest Cons

RF :- DT + RS + CS + Agg.

Not good in large dimensionality data,
Categorical features with large no. of categories

We have seen that almost all the cons that hold for DT also hold for RF except some.

<u>Bias</u> Variance <u>Tradeoff</u> :-

In Case of DT, Depth is the major factor of Bias Variance Tradeoff.

In Case of RF, K is the factor of Bias Var. tradeoff

Boosting Intuition

B In Bagging $\boxed{\text{we}}$ do,

Bagging $\underbrace{\text{high var + low bias}}_{\text{Base models}}$ + $\overbrace{\text{Randomization + Aggregation}}^{\text{Reduce Var.}}$
(CS, RS)

Boosting Low Var + High Bias + Additively combine
{ Reduce Bias while keeping our variance low }

Core idea on how actually boosting reduces bias.

In Stage 0, $D_{Train} \rightarrow M_0$ — DT (Complexity) (Shallow Tree)
High Bias & Low Variance

A high bias & low-variance model will be created from D_{train} (whole data) having high bias & low variance.

$-\log_{10}(2.3 \times 10^{-11})$

High Bias :- Large training error

Stages

(0) a) $D_{train} \longrightarrow M_0$

$(x_i, y_i)_{i=1}^{n}$ $\quad y = h_0(x)$ → Prediction

b) $y_i - h_0(x) \Rightarrow$ Errors

Now at the end of stage 0, we have $\{x_i, y_i, error_i\}_{i=1}^{n}$

(1) M_1, the model in stage 1 takes x_i as input and tries to predict $error_i$.

$M_1 \longleftarrow \{x_i, error_i\}$

↓ → Predicts

$h_1(x) \approx error_i(y_i - h_0(x))$

~~$h_1(x)$ tries to~~ pr

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

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

(2) $\{x_i, err_i\}$

$\quad \hookrightarrow y_i - F_1(x_i)$

model at end of stage 2,

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

At the end of stage K :-

$$F_K(x) = \underbrace{\sum_{i=0}^{K} \alpha_i \, h_i(x)}_{\text{additive weighted model.}}$$

$h_i(x)$ → trained to fit the residual error at the end of previous stage.

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

Final model ends up having low residual error.

~~so, it is~~ at the end it is reducing Training error

Train error ↓ → Bias ↓

Regularization & Shrinkage

$$F_M(x) = h_0(x) + \sum_{m=1}^{M} \gamma_m h_m(x)$$

M → # of base models

As M↑ ⇒ overfit ↑ ⇒ variance ↑

As # Base model ↑, Bias ↓ but variance ↑.

Shrinkage

$F_m(x) = F_{m-1}(x) + \nu \cdot \gamma_m h_m(x)$, $0 < \nu < 1$

If $\nu = 1$, the eq^n becomes as previous

If ν is small, overfit ↓

~~For~~ : To prevent overfitting we have 2 hyperparameters M and ν.

Train & Run Time Complexity

Train complexity :- $O(n \log_n d * M)$

RF is parallizable but GBDT are sequential so it takes more time to train than RF.

Runtime

GBDT : $O(depth * M)$

Space Complexity

$O($ Store each tree + $\gamma_m)$

~~S~~ Q

Scikit learn GBDT = GBDT + Row sampling

→ from RF

Xgboost = GBDT + (RS + CS)

booster = 'gbtree' / 'gblinear'
Both tree & linear model can be high bias & low variance model.

colsample_bytree :- While constructing tree, check particular sample of columns

~~column sample~~
colsample_bylevel = 1 , While constructing each node of a tree ~~..~~ only a random sample is considered.

reg_alpha = 0 , L1
reg_lambda = 1 , L2

$$F_m(x) = h_0(x) + \gamma_1 h_1(x) + \gamma_2 h_2(x) + \ldots$$

γ_1 ↓ Weights

$$\underbrace{\alpha L1}_{\text{Regularizer}} + \underbrace{\lambda L2}_{\text{Regularizer}}$$