ENSEMBLE LEARNING

→ Linear Regression
→ Logistic Regression
SVM
Decision Tree
KNN
Naive Bayes
K-means Clustering

} First level of ML

→ Ensemble learning - Weak learner

↓ 2nd level of ML

↓ Strong learner

→ Deep learning → 3rd level of ML

→ Transfer learning → 4th level of ML

→ Example

Images of cat & dog

We have certain features which differentiate cat & dog, like,

claws : big → weak learner [models which diff. nicely but in the case of exceptions, do not perform well.]
face : whisker
ears : pointed
limbs : big

} Aag- Aag weak learner

But, there can be some exceptions, like any one cat can have big claws, then in that case, our algorithm will fail.

⇒ Now, we combine, these 4 features and through majority voting, we classify the images.

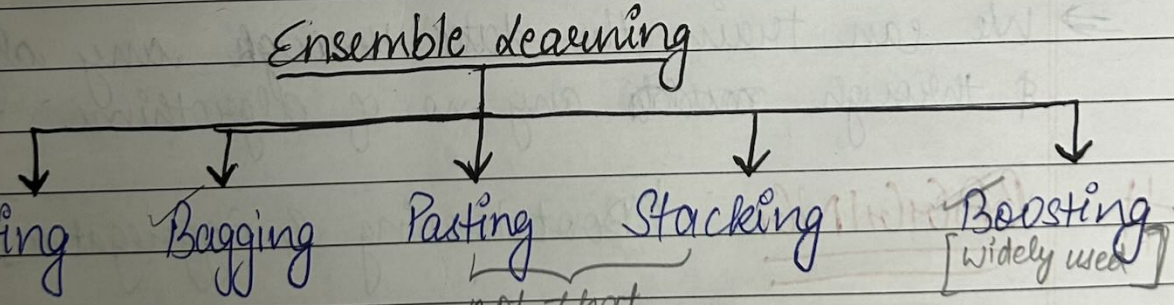
Eg: Dog, Dog, Dog → (3); Cat → (1)

→ There are 3 features saying, the image is of dog & 1 feature saying it is of cat through majority voting, we can conclude that image is of dog.

⇒ **Advantage of Ensemble learning -**

Bias get removed & things get neutral

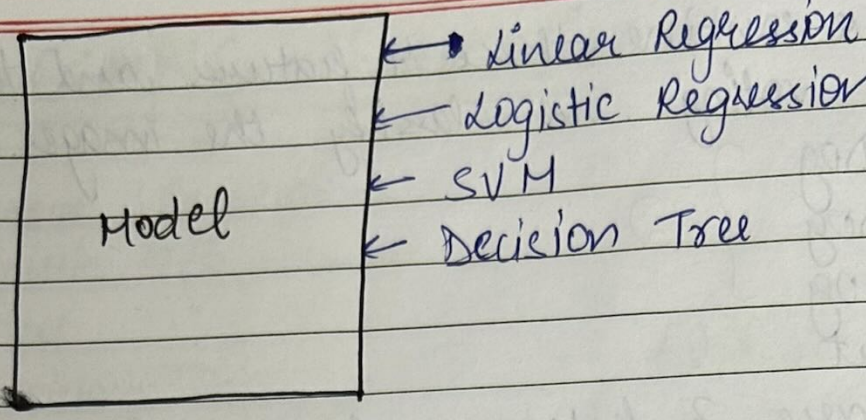
⇒ **Types of Ensemble Learning**



VOTING [voting Classifier]

→ takes same data

→ We train the same training data through different models.



Let's suppose,
the model is predicting image of cat & dog
⇒ The four above algorithms have given the following results -

→ 1
→ 1
→ 1
→ 0

} 1 (through majority voting)

* Final prediction is done through voting.

⇒ We can train the data through any algorithm & through any no. of algorithms.

BAGGING [Bootstraping & Aggregator]

⇒ Bootstraping & aggregator

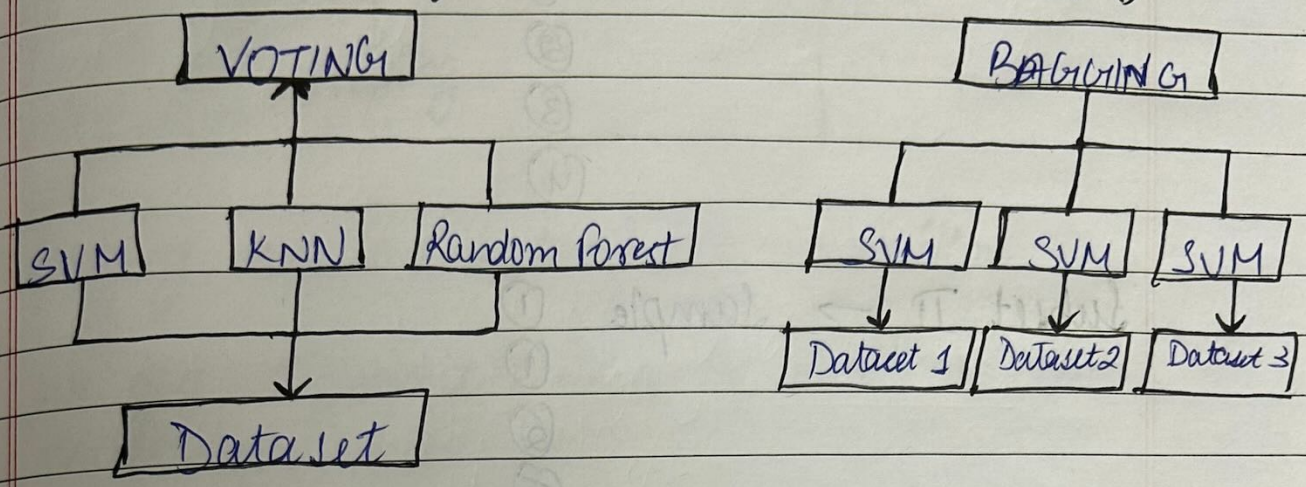
Bootstraping → different data
[data gets divided into subsets]

* Subsets were made using bootstrap

⇒ We use same model or algorithm on different (subsets) data.

Date / /
Page No.

Ques What is the difference b/w voting & bagging?



→ In voting, we apply different algorithms on the same dataset & through majority voting final prediction is done

→ In bagging, the same algorithm is applied on different datasets & by combining the results from them final prediction is done. (& they move parallelly)

→ Bootstraping
[sampling with replacement]

→ Example features (independent variable)

	Age	Exp	Salary	Selected → dependent variable
①	21	4	23	1
②	22	5	33	0
③	26	3	25	1
④	24	7	42	0
⑤	21	5	23	1

→ Samples

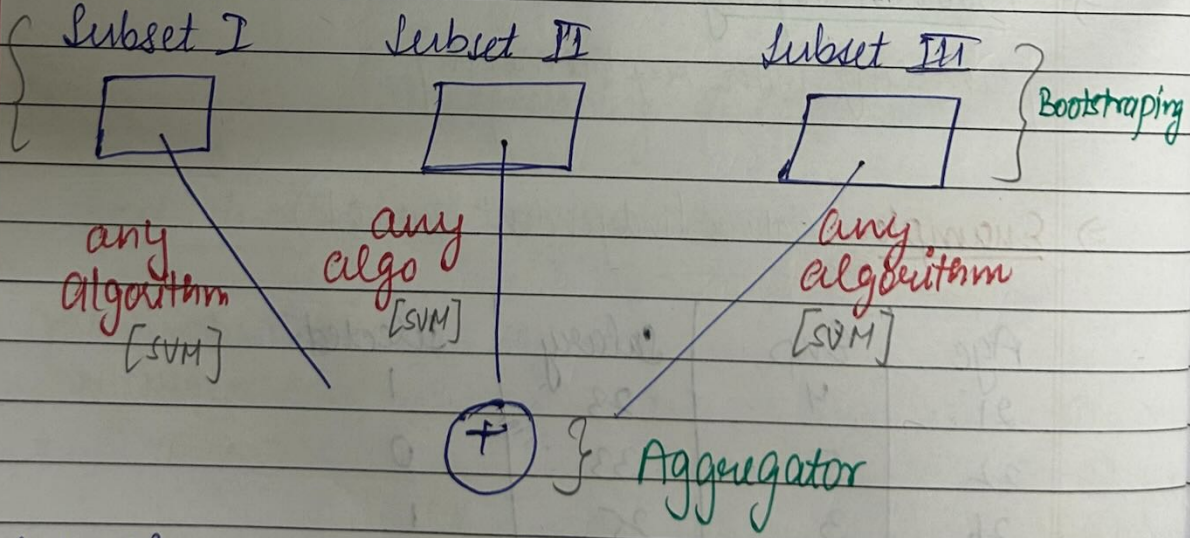
Subset I → Sample ① ② ② ③ ④

Subset II → Sample ① ① ② ⑤ ②

Subset III → Sample ⑤ ⑤ ④ ④ ③

* Samples can be repeated in a subset

⇒



* Subsetting
  → Bootstraping
  [sampling with replacement]

Date / /
Page No.

* Sampling without replacement
↓
Pasting

⇒ Bagging uses same training algorithm

⇒ we give different subset (of data) to each model

⇒ **Procedures of Bootstraping**

→ Random subspace → Random patches

→ **Random subspace** - only features subsetting is done

Random Patches - column & features subsetting is done

⇒ **Hyperparameters**

→ **oob_score** - Out of Bag Evaluation

Eg
We have 5 samples (or features)
Subset I contains samples - 1, 2, 2, 3
then sample 4 & 5 are out-of-bag & 1, 2, 3 is in-bag

⇒ If oob_false
then out of bag is not being used

→ If we use OOB, then training will be done on iib (in-bag) samples & testing will be done on oob samples.
(in this case, we do not require train-test split)

→ OOB
↳ Testing accuracy
Calculates
(bcoz ~~do~~ that data is unseen).

→ Base_estimator - it means which model is being used
(konsa model use kr rhe hai, woh batana padta hai)

→ N_estimators - no. of models.
(kitne models use kr rhe hai)

→ n_jobs - By default it is none
None → using only 1 processor
-1 → using all the processors

→ Bootstrap - By default = True

→ max_samples - hme kitne samples chahiye
1 → means all samples

→ Bootstrap_features - does subsetting of features
By default = False (doesn't do subsetting)
True (does subsetting)

→ max_features - takes all the features

Date / /
Page No.

* **Column & Row Sampling**
  (is done through)
  - Random Subspace
  - Random Patches

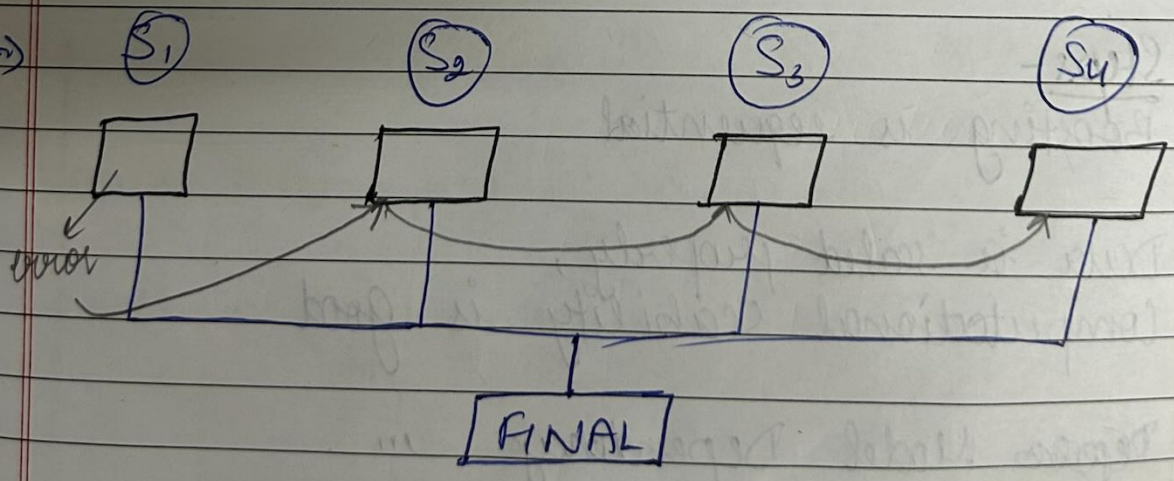
* All ensemble learning techniques can be found in sklearn.

* Bagging classifier is used for both → Regression
  ↳ classification

* random_state → jab hum 2nd or 3rd time run karte hai, toh result same aaye, isliye hum random state use karte hai

BOOSTING

In bagging, we were unaware of the errors or misclassifications happening in the models.



In boosting, we pass on the errors

S1 error → S2 error → S3 → S4

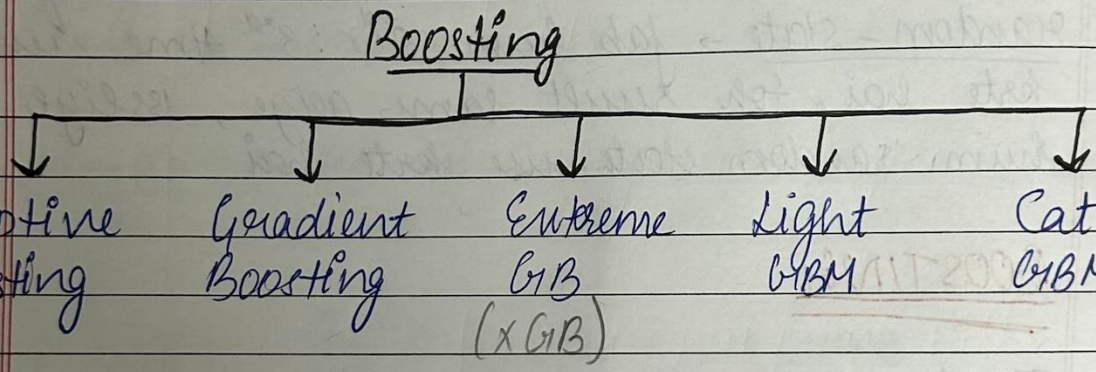
In bagging, the models move parallely but

⇒ In boosting, the models move sequentially

⇒ Boosting is done sequentially.

⇒ In boosting, models are dependent on each other, they share the ~~two~~ error & boost the dependencies.

⇒ Types of Boosting



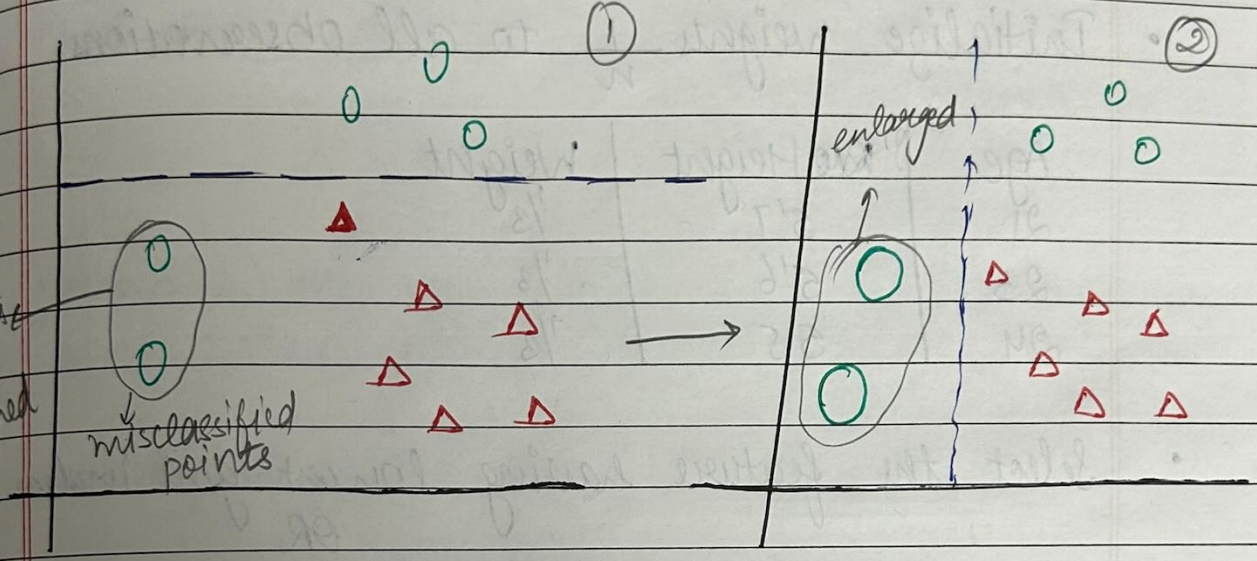
⇒ Ada Boost [Adaptive Boosting]

Steps -

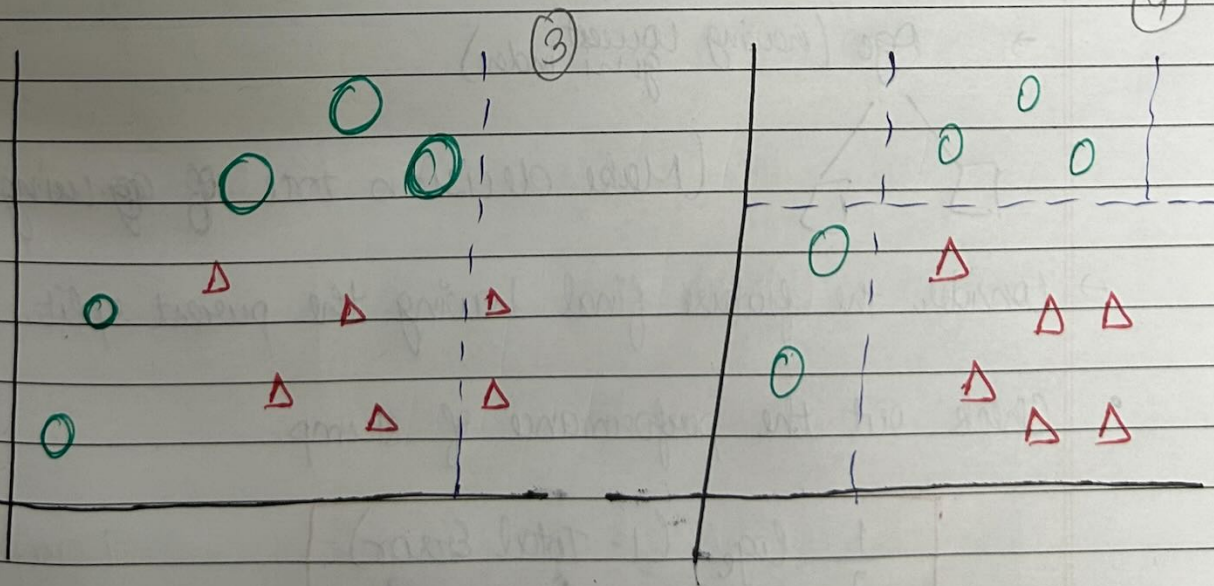
- Boosting is sequential
- Data is scaled properly, Computational scability is good
- ~~Depen~~ Model Dependency
- Models used by Ada Boost →

- Next model corrects the error of its previous model
- Missing values & outliers are handled by AdaBoost
- It handles quantitative & qualitative values

⇒



→ Due to decision stamp, splitting is done
→ If points are :
Correctly classified → weights ↓ (decreased)
Misclassified → weights ↑ (increased)



→ **Steps for AdaBoost**

Age	Height
21	5'7
23	5'6
24	5'5

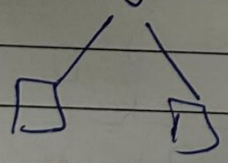
- Initialize weights $\frac{1}{n}$ to all observations.

Age	Height	Weight
21	5'7	1/3
23	5'6	1/3
24	5'5	1/3

- Select the feature having lowest gini index
OR
high information gain
[purest split / high probability score]

& we make a decision tree.

→ Age (having lowest gini index)



(Make decision tree of eg. using age)

→ Consider the feature final having the purest split.

- Check out the performance of stump.

$$\frac{\frac{1}{2}\log_e(1 - \text{Total Error})}{\text{Total Error}}$$

7 samples

5 (RC) Rightly classified

2 (MC) Misclassified

Date / /
Page No.

- Decrease the weights of rightly classified points & increase the weights of misclassified points.

$$\text{New Weight} = (\text{Old Weight}) \cdot e^{\pm \text{performance}}$$

(RC → +, MC → −)

Check the probability, RC ↑, MC ↓ (Focus on points having less probability)

- Normalise the new weights (& give these weights to the model)

* Keep repeating the steps.

⇒ **Hyperparameters -**

→ Base estimators (By default - decision tree)
→ Learning rate
→ No. of estimators