Ensemble Method

(1) Simple Ensemble Techniques

1.1 Max Voting

- used for classification problems mostly.
- multiple models are used to make prediction for each data points.
- Predictions ⇒ vote "GO BY MAJORITY"

Eg Input ($x_1, x_2, x_3 \ldots, x_i$)

Outputs

model(DTC)	model(KNN)	model(LR)	final pred
5	4	5	5
model(SVM)	model(Ridge Regression)		
4	4		

- final pred^n max voting = 4 | averaging = 4.40 | Weighted avg = 4.41

	model1	model2	model3	model4	model5	final
wt	0.23	0.23	0.18	0.18	0.18	
rating	5	4	5	4	4	4.41

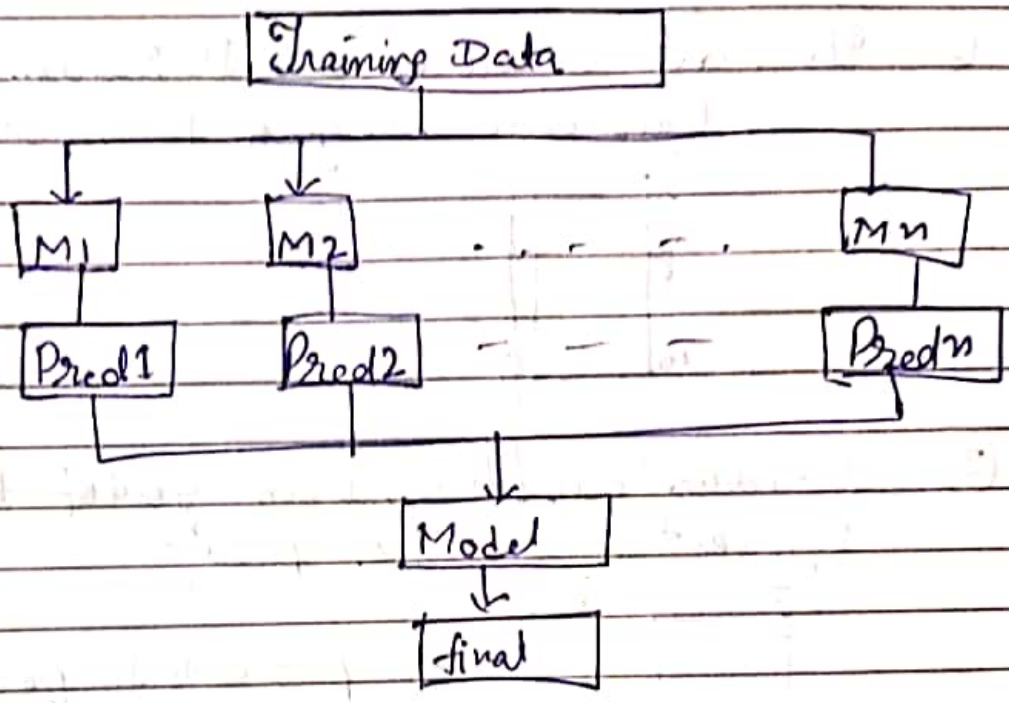
$5*0.23 + 4*0.23 + 5*0.18 + 4*0.18 + 4*0.18 = 4.41$

1.2 Averaging } Can be used for regression
1.3 Wt Average } Can be used for regression

2 Advanced Ensemble Techniques → Stacking, Blending, Bagging, Boosting

2.1 Stacking

Uses the pred^n from multiple models to build a new model. This model is used to make pred^n of test set.



We take the training data and run it through multiple models M1 to Mn, called base models, to generate pred^n

Pred 1 to Pred n are prediction & this is the input sent to (Model), instead of Max Voting.

~~Instead of~~ Model take this input & gives final pred^n.

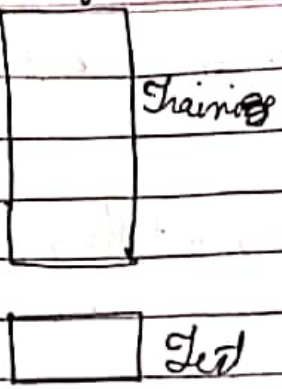
Right model is chosen acc to the problem. i.e. whether it is classification or regression.

"Overfitting" is an issue

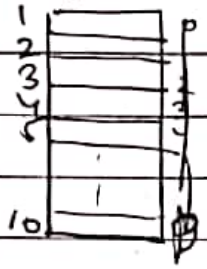
Reducing Overfitting

Data

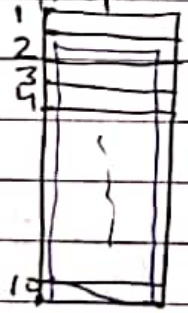
(1)



(2) Train data is divided into 10 random parts. So 1 train data has given 10 smaller Datas.

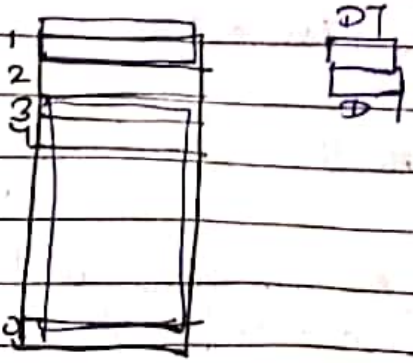


(3) To reduce "overfitting, train 9 out of these 10 parts, 10th part is used for pred^{n}.

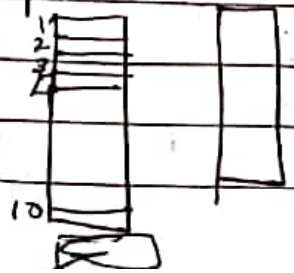


- part 2 to 10 for train
- part 1 for pred^{n}

(4) Repeat this

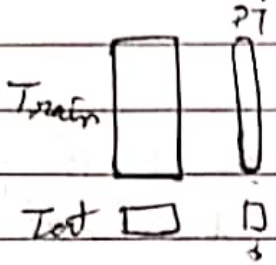


(5) finally pred^{n} for all 10 parts. Each of these pred^{n} is coming from a model that has not seen the same data points of

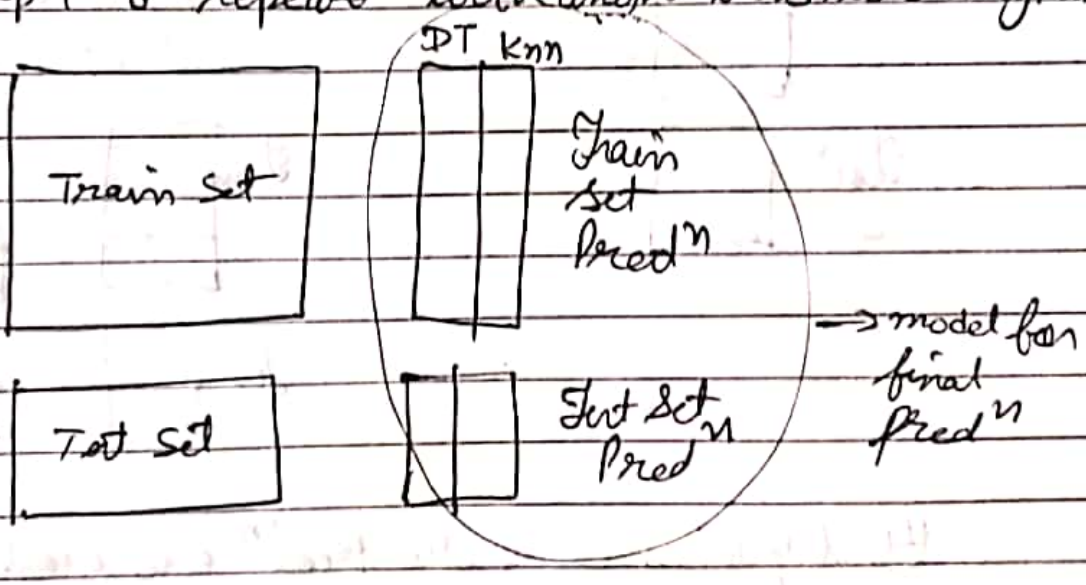


(6) We created 10 models to get

(6) for creating a test dataset, we can use the entire train set dataset..
Again train the model on entire training dataset & make pred^n on test



(7) Step 1-6 repeated with another base model (Dayknn)

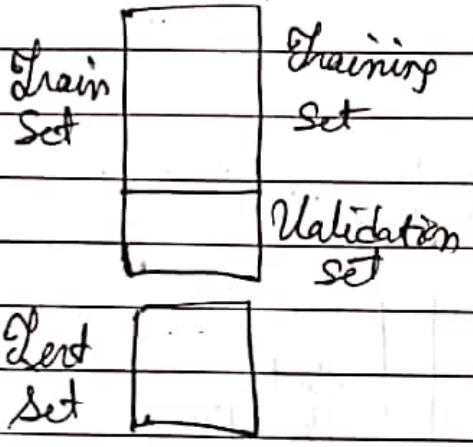


Eg process 1 to 6 is used to get DT, Knn dataset then the third model LR is used on pred^n of DT & Knn models

2.2 Blending

Similar to stacking except that train data is splitted into training set and validation set. ~~and the pred^n set~~ & Validation set & pred^n are used to ~~reen~~ build a model to reun on test set

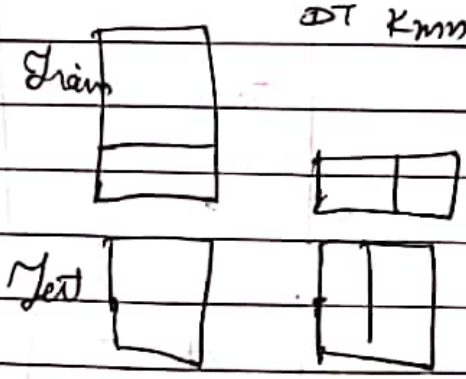
① Train set split



② models are fitted on trainy set.

③ Pred^n are made on Validation & test set

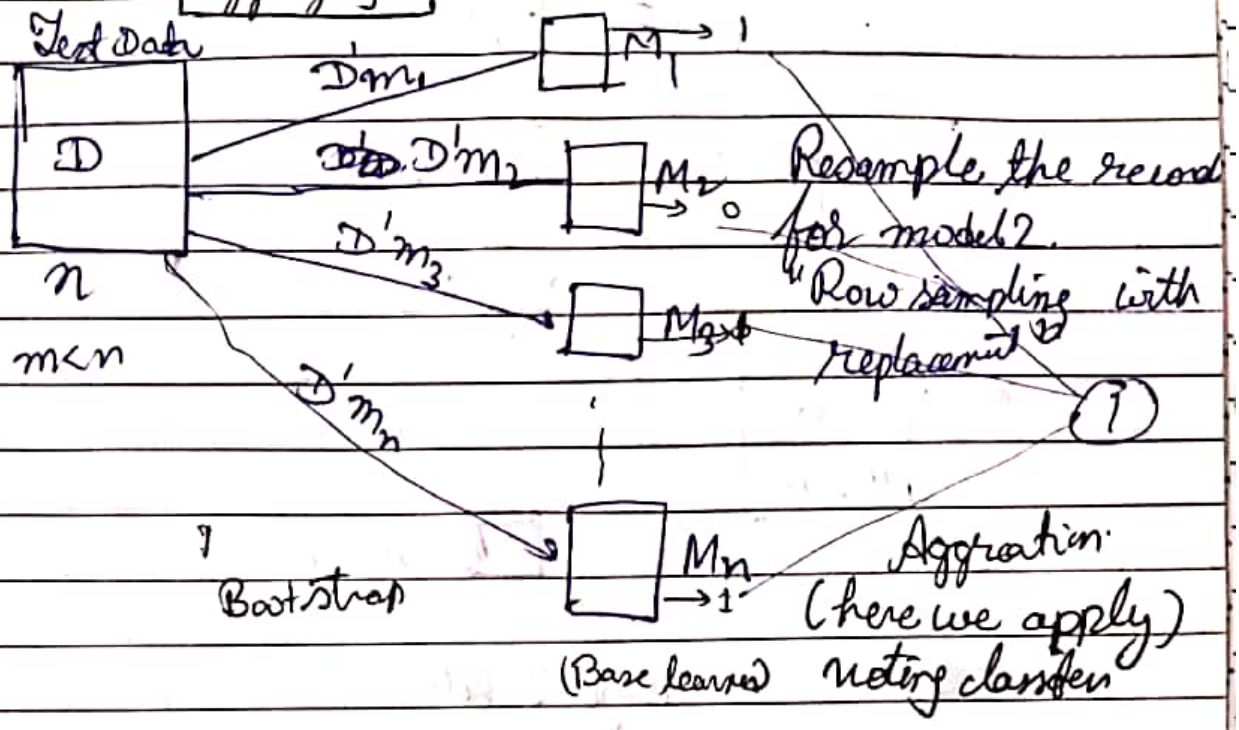
DT KNN



④ Validation set & it's pred^n are used as features to build a new model.

⑤ Model is used to make pred^ns on test & meta features

2.3 (8) Bagging | Bootstrap Aggregating



$D'm_1 \neq D'm_2$ (although some of the records may be same)

Row sampling with Replacement => Bootstrap

R sklearn Bagging hyperparameters

base_estimator: DT by default else specify

n_estimator : no of base_estimator < small no: error prone. / large no: computationally expensive

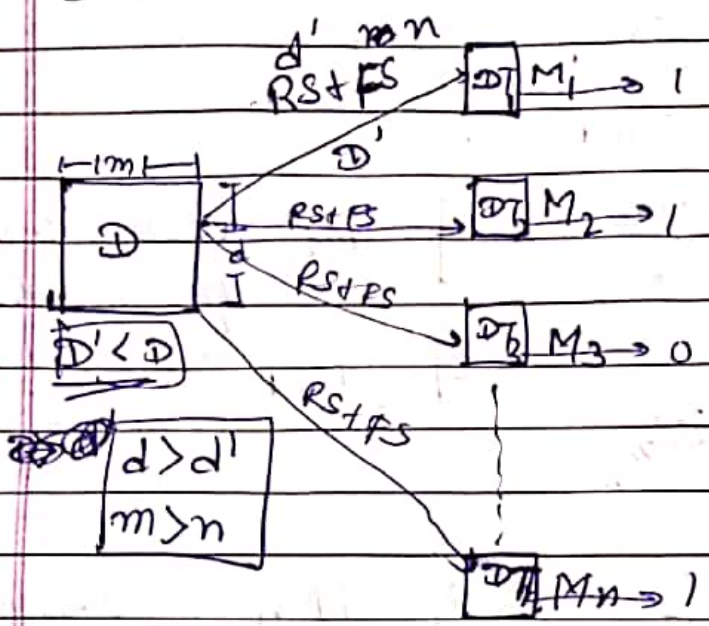
max_sample : size of subset.
max no of samples to train each base estimator

max_features : no of features from dataset.

n_jobs

(Row sampling + feature/column sampling with replacement)

RANDOM FOREST



Majority Vote

DT

DT to its complete depth → ① Low Bias
② High Variance

→ Low Bias ⇒ training error v. low
→ high variance ⇒ test error high
→ Overfitting

On combine all DTs first, majority vote, high variance converts to low variance.

~~1000 records~~

- If we have 1000 records, change of 200 records won't affect the result much as this 200 records will be properly splitted among dt

If it is a regression problem, we takes the mean or median of the output.

Hyperparameter ⇒ how many DTs for RFs.

Diff^nce fit(), transform(), fit_transform() and predict() methods in scikit learn.

Hyper Parameter Tuning

EDA → FE → FS → Model Creation → Model Deployment

Data Preprocessing

⇓

Transformer

Eg:- Standard Scalar.
$\mu = 0, \sigma = 1$

MinMax Scalar, PCA, etc

Imputer (handles NaN).

Model Training

⇓

Models

Eg: Linear Regression, Logistic, DT

f_1	f_2	f_3	f_4	o/p
x_1				
x_2				
⋮				
x_i				

⇓

$f_1' f_2' f_3' f_4'$ → model → fit / predict

Standard Scaler

$f_1 \rightarrow f_1'$

$\frac{x_i - \mu}{\text{final } \sigma}$

$\mu = 0, \sigma = 1$

fit → computes μ, σ

transform → applies formula

Bootstrapping

Sampling With replacement

Resampling method that uses random sampling with replacement

1	2	3
4	5	6
7	8	9
10	11	12

initial dataset

⇒

1	10	3	8	3
4	2	11	9	3
.	.	.	.	.
5	12	1	5	6

Bootstrap sample of size 5

Suppose you have an initial sample with 3 observations Using Bootstrap method, you'll create a new sample with 3 obs as well.

In this case, second observation was chosen randomly & will be the first observation in our new sample

X	Y
5	10
4	8
2	4

⇒

X	Y
4	8
2	4
4	8

It is very much possible for an already chosen sample to be chosen again

2.3.1 Boosting Algorithm

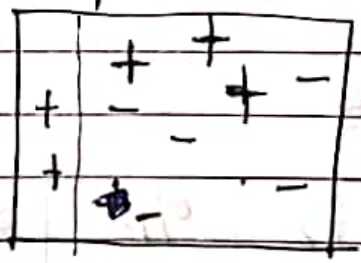
If a data point is predicted incorrectly by first model and then the next (probably all models), combining them will not give a very good result.

Such situation is taken care by boosting

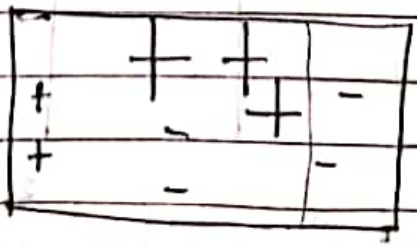
It is a sequential process where each subsequent model attempt to correct the errors of previous model. Next model is dependent on previous model

Steps

1. A subset is created from the original dataset
2. Initially all the points are given equal weights.
3. Base model is created on this subset.
4. This model makes a pred^n on entire dataset.



5. Errors are calculated using actual & predicted values.
6. Incorrectly predicted have higher weights
7. Another model is created will predict on entire dataset.

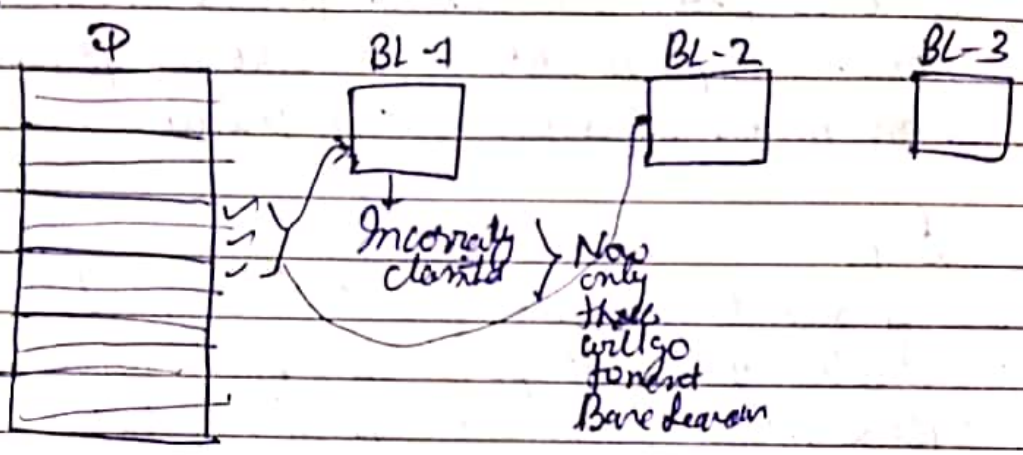


8. Multiple models are created, each correcting prev model.
9. final model / strong learner is weighted mean of all models

- AdaBoost
- GBM ~~& XGBM~~
- XGBM
- Light GBM
- Cat GBM

~~AdaBoost~~

Boosting



AdaBoosting

→ w = 1/n (no of records)

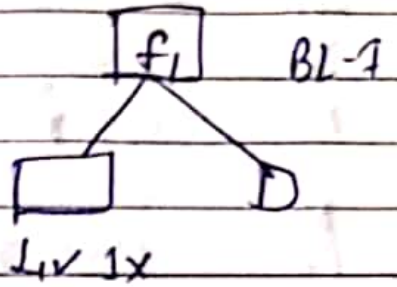
	f_1	f_2	f_3	o/p	Sample wt	Updated wt	Normalized
1				Y	1/7	0.05	0.05/0.68
2				N	1/7	~~0.05~~ 0.34	0.34/0.68
3				Y	1/7	0.05	
4				N	1/7	0.05	
5				Y	1/7	0.05	
6					1/7	0.05	
7					1/7	0.05	
					$\Sigma w_i = 1$	$\Sigma w_i \neq 1$	

BL-1 (DT) with depth 1 → f_1 | f_2 | f_3 | stamps

for each feature create stamp

Step-1

from there we have to select BL-1 based on entropy & / gini index
(here entropy selected)



Step-2

- for this(1x) find total ~~error~~ error.

Total error = 1/7

Step-3

- Performance of stump = $\frac{1}{2} \log_e \left(\frac{1-TE}{TE}\right)$

$= \frac{1}{2} \log_e \frac{1-1/7}{7} = \frac{1}{2} \log_e 6$

$= 0.896$

Step-4

New sample wt = weight $\times e^{\text{performance say}} = \frac{1}{7} \times e^{0.895}$

(incorrect ones = ~~1/7 × e^0.895~~

$= 0.349$

Correctly classified = wt $\times e^{-\text{performance say}}$

$= \frac{1}{7} \times e^{-0.895} = 005$

S-5 Sum of updated wt ≠ 1 ∴ normalize

$\therefore w_i' = w_i' / 0.68$

S-6 Create a new dataset based on updated values will mostly select wrong records.

f_1	f_2	f_3	o/p	Normalised	
			Y	0.07	0 - 0.07
			N	0.51	0.07 - 058
			Y	0.07	0.58 - 065
			N	0.07	0.65 - 0.72
				"	
				"	
				"	

S-7 We'll create bucket

To select data, algo runs 8 iterations to select data. Let in first iteration, Algo selects a random value of 0.43 and check in which bucket it falls

1st

2nd 0.31

↑