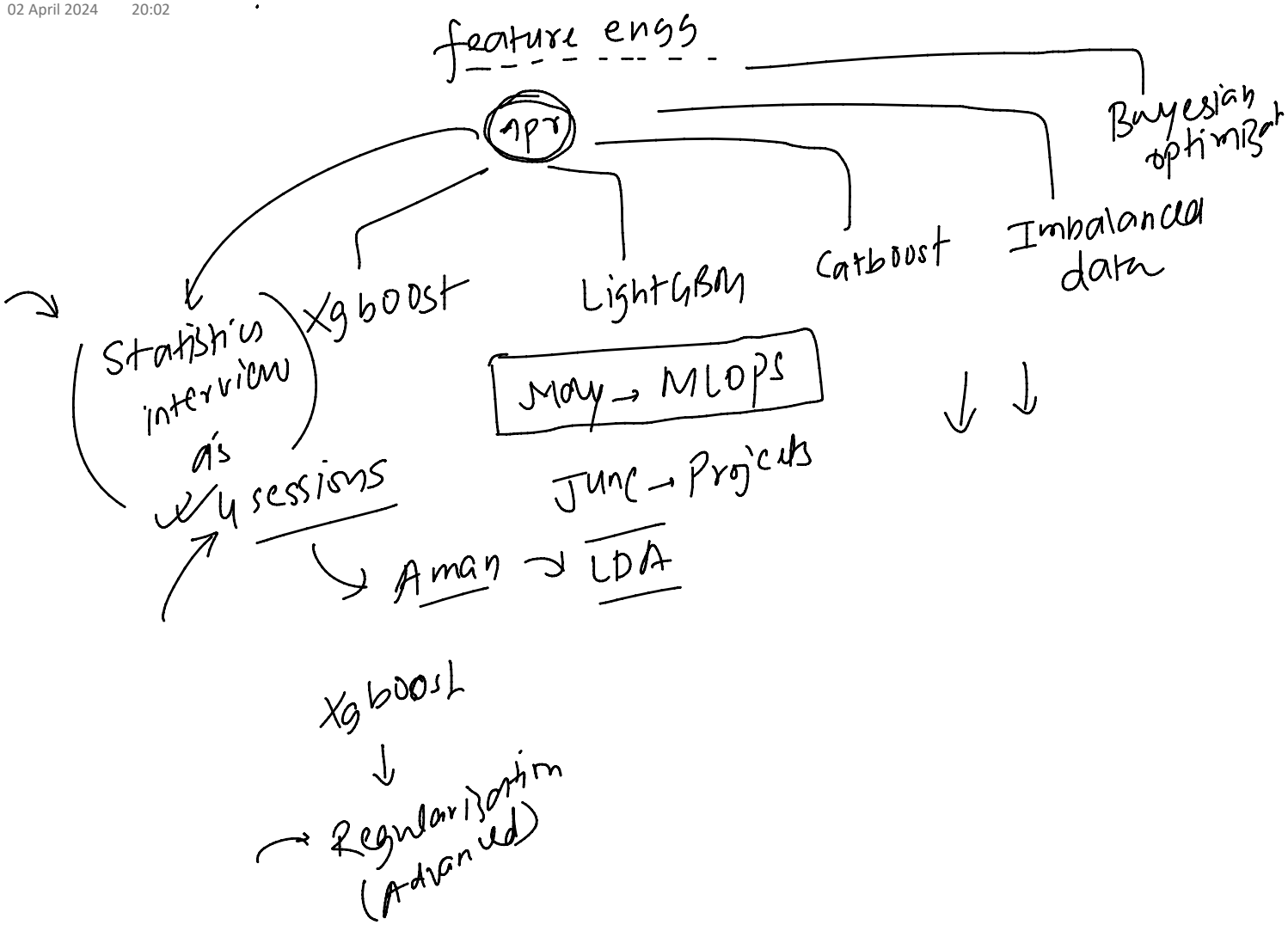
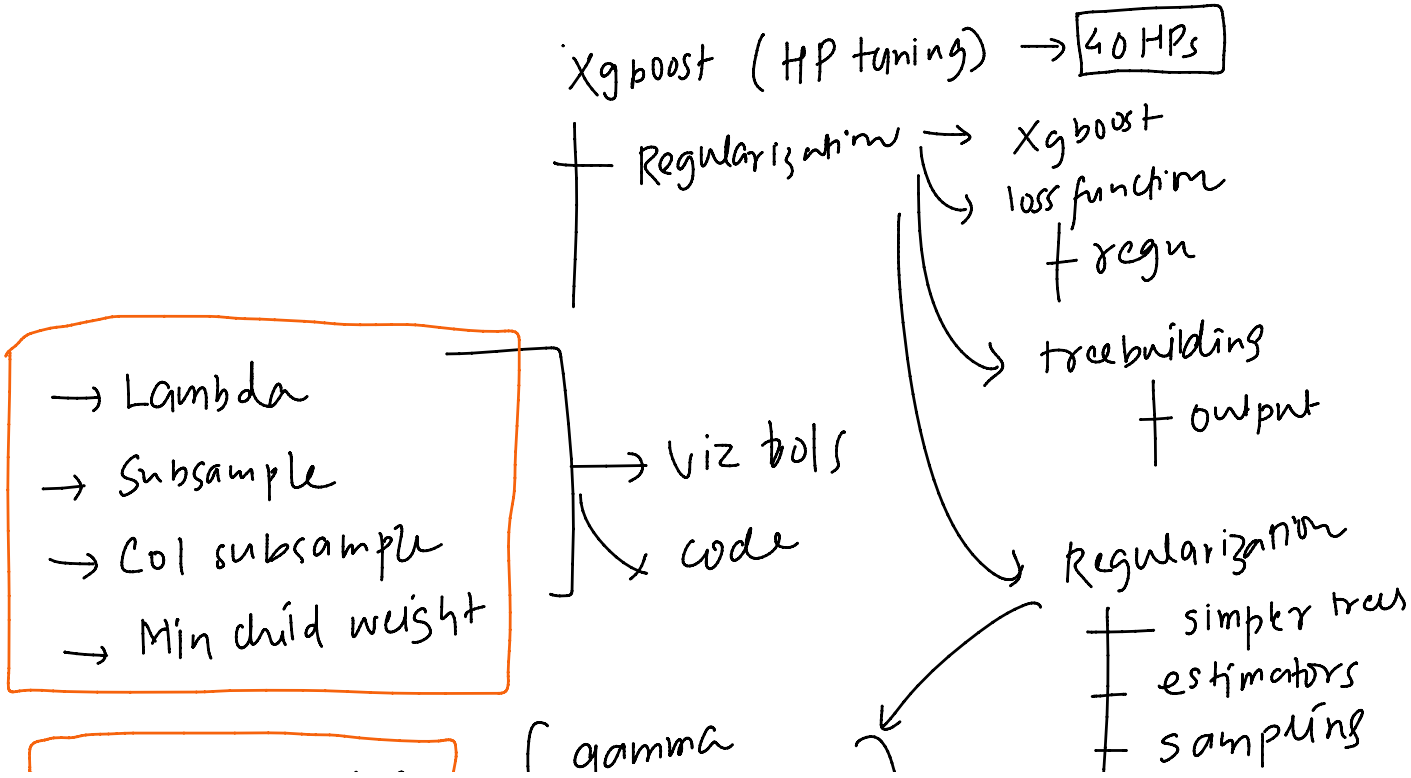
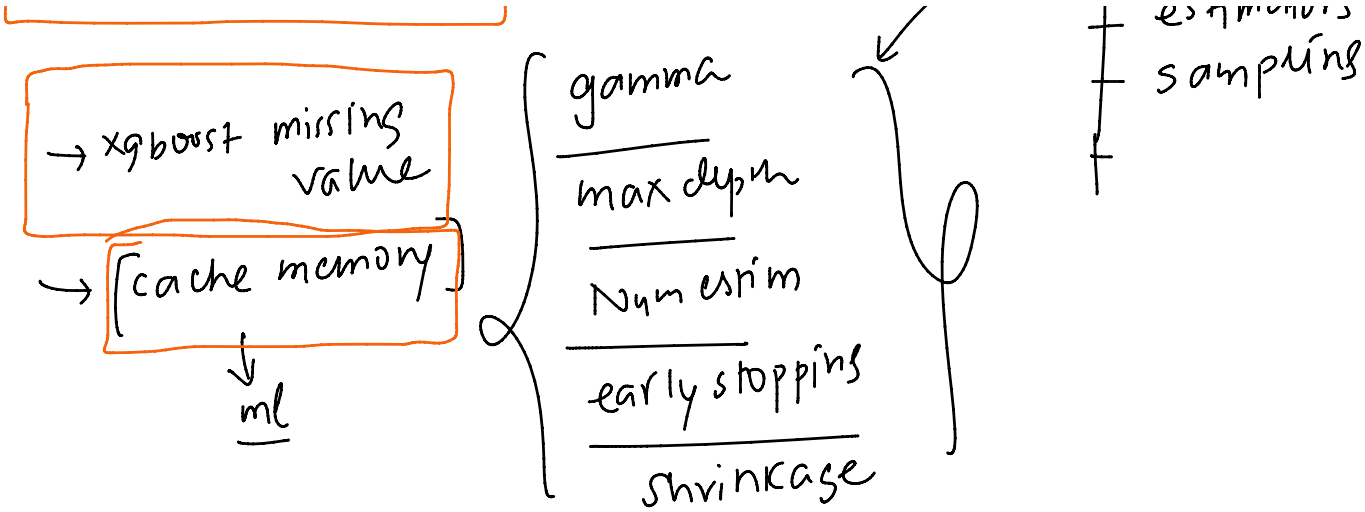
Plan of Attack

02 April 2024 20:02





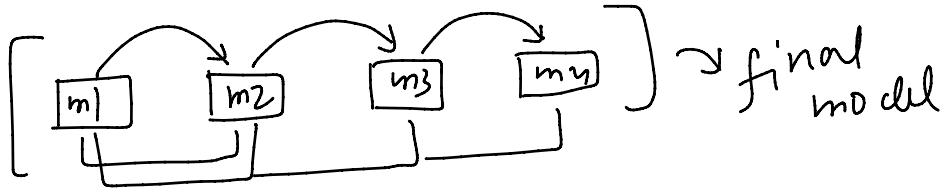

→ xgboost missing value
→ cache memory
ml
gamma
max depth
Num estim
early stopping
shrinkage
sampling

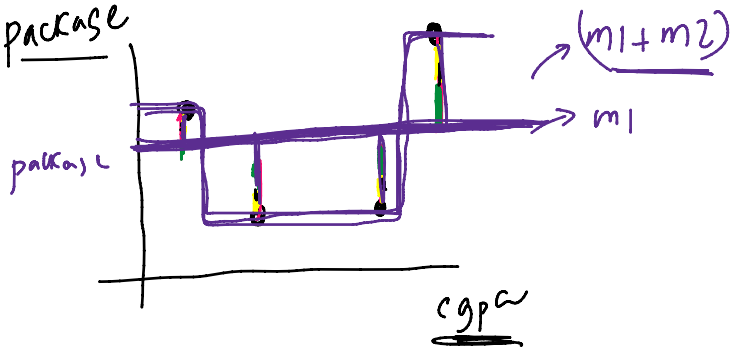
Quick Recap

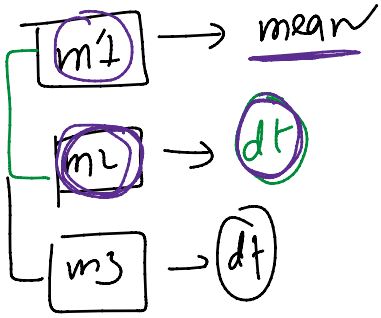
30 March 2024 23:25

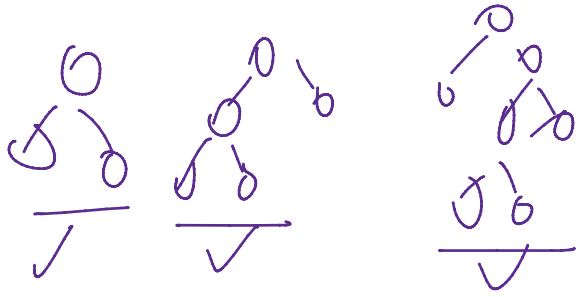
XGboost → Boosting algo (Gradient Boosting)

stagewise









loss fun

minimize $\left[L(y_i, \hat{y}_i)\right]$

$\left[f_t(x_i)\right]$

$f_t(x_i)$ → functions

function parameters

$$L = L(y_i, \hat{y}_i) + \Omega(f_t(x_i))$$

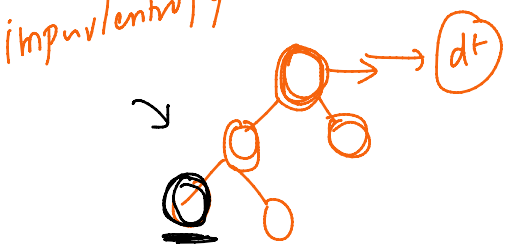
reg

$$\gamma T + \frac{1}{2}\lambda \sum_{i=1}^{T} w_i^2$$

regularization

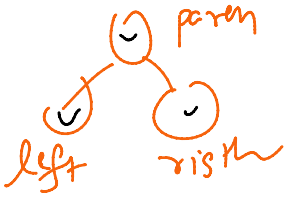
impurity

impurity



$$\text{similarity score} = \frac{(\text{sum of residuals})^2}{\text{num of residuals} + \lambda}$$

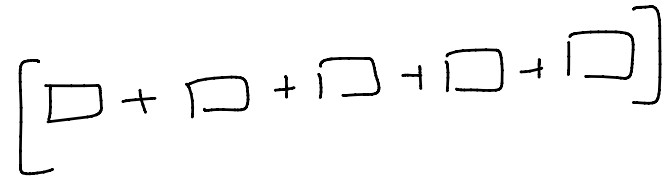
gain



$$\text{gain} = (SS_{left} + SS_{right} - SS_{pa}) - \gamma$$

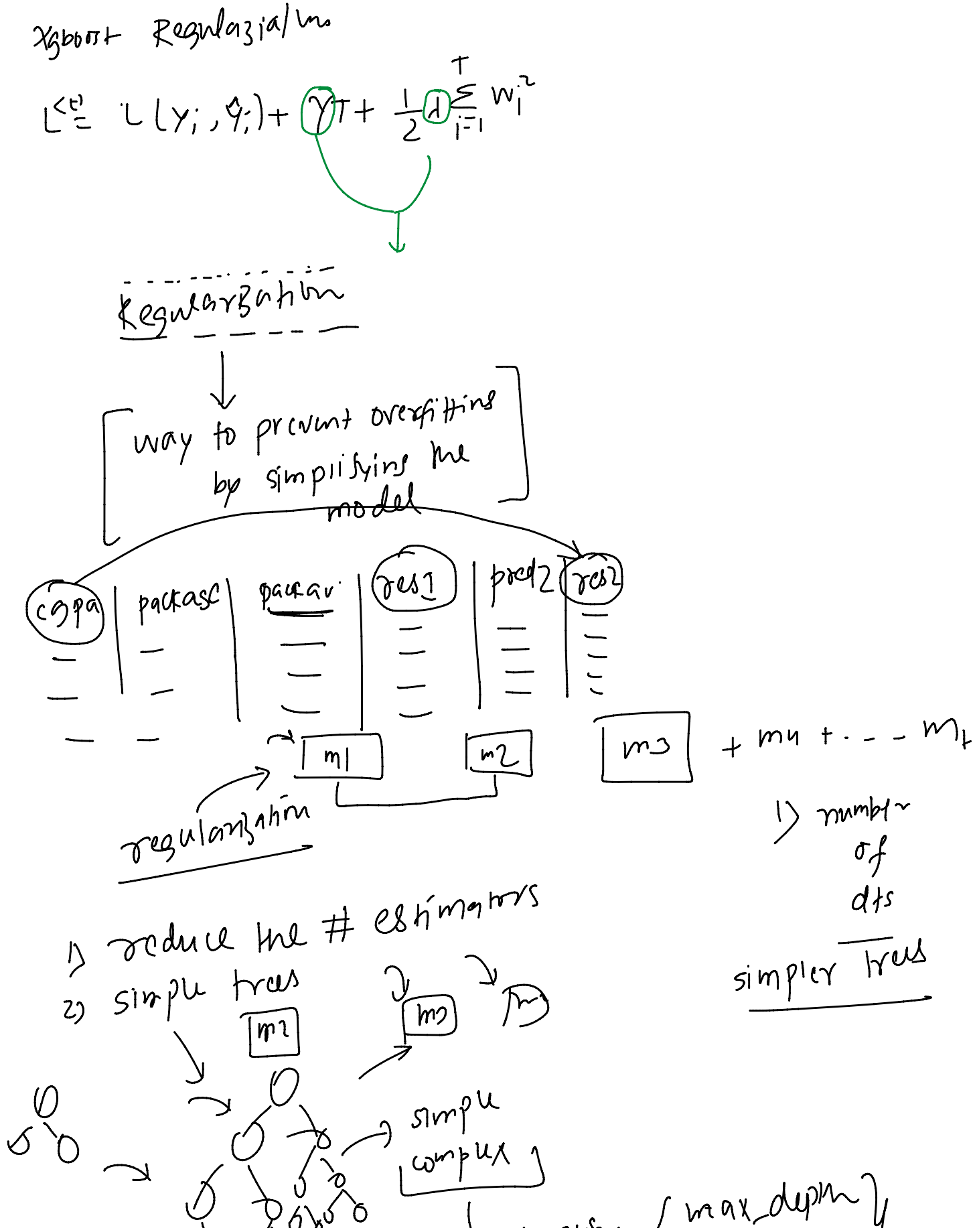
regression

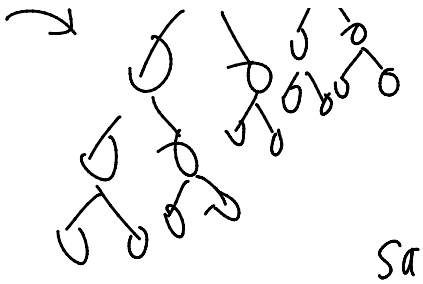
$$\text{output} = \frac{\text{sum of residuals}}{\#\text{ of residuals} + \lambda}$$



Ways to reduce overfitting in XGBoost

30 March 2024 23:57





simplify { max_depth, pruning }

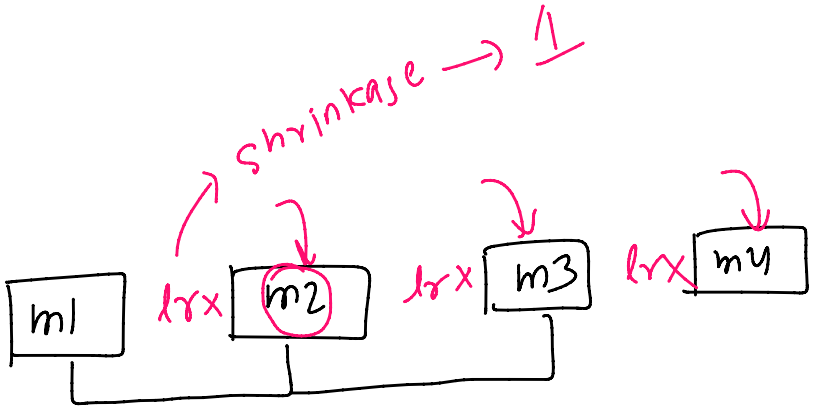
randomness

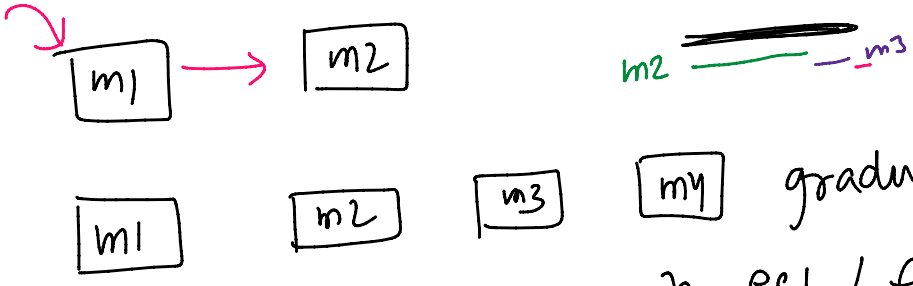
sampling

3) sampling { rows, feature } →

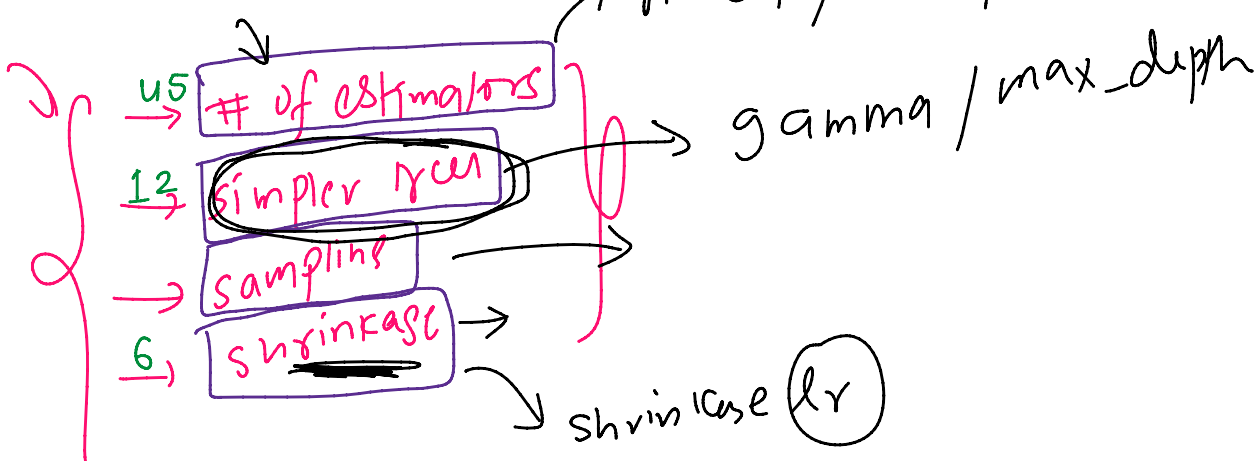
4)

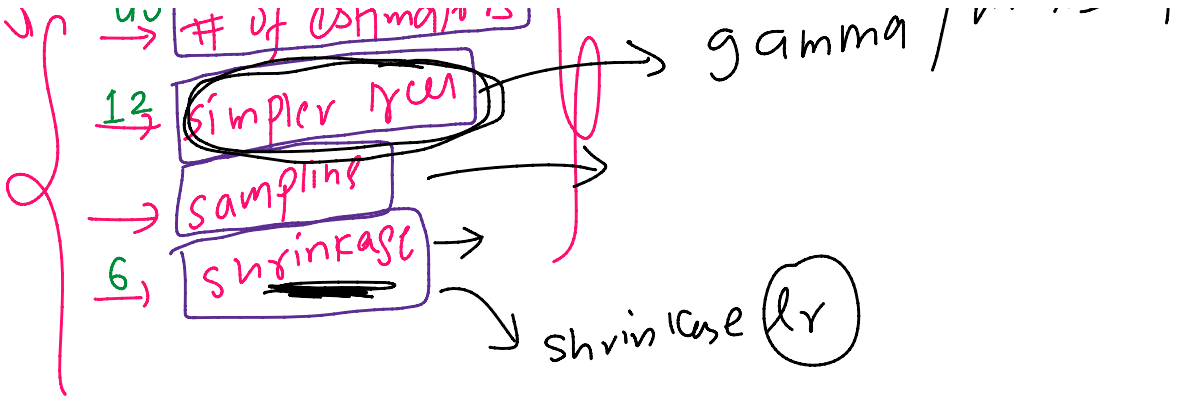
0.5





gradual nature





gamma →

λ → simpler tree

1. Gamma (γ)

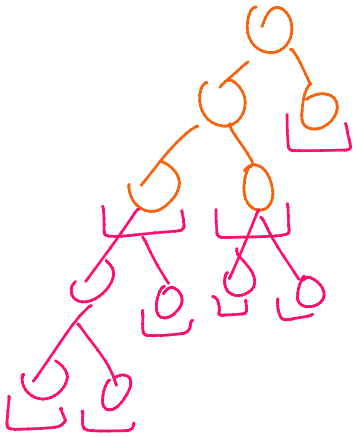
30 March 2024 23:26

$$L = L(y_i, \hat{y}_i) + \gamma T + \frac{1}{2}\lambda \sum_{i=1}^{T} w_i^2$$

post pruning

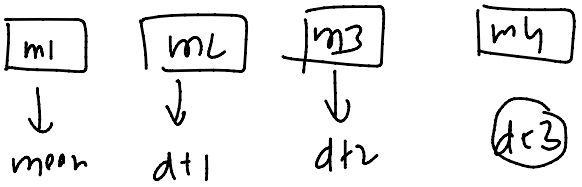
$T \rightarrow$ represent

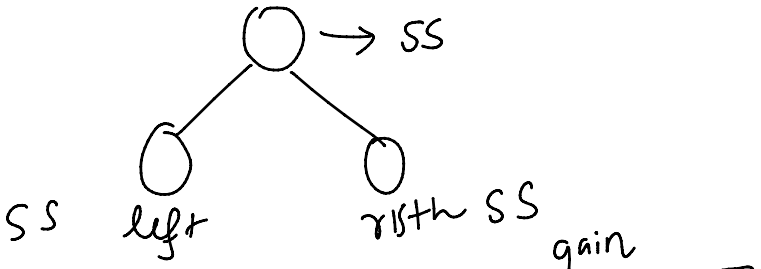
t iter $\quad f_t(x_i)$



3 leaf nodes

penalise γT number of leaf nodes

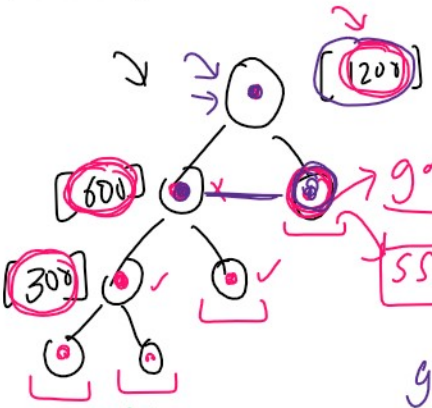




$$\text{gain} = SS_{left} + SS_{right} - SS_{parent}$$

gain

my

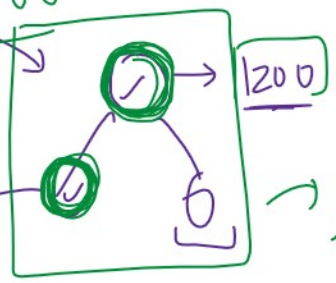


my gains

$SS_{left} + SS_{right} - SS_p$

gain

1200-100 = 1100



final pruning

$gain = SS_{left} + SS_{right} - SS_p$

1200-600-400

$gain = \frac{1}{2}[SS_{left} + SS_{right} - SS_p] - \gamma$

600-800 = -200

300-800 = -500 -ve

pruning

$gain = \frac{1}{2}$

$\gamma = 100$

[300 - 100]

[200] +ve

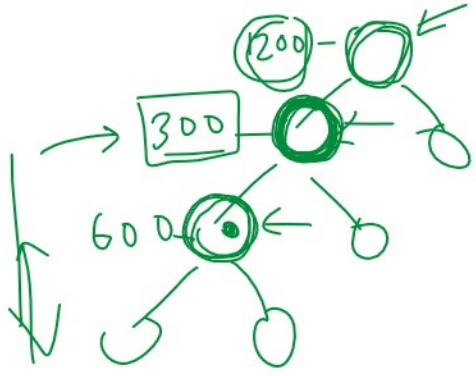
$\gamma = 800$

600 - 100 = 500 +ve

post pruning

1200-500 = +ve

300



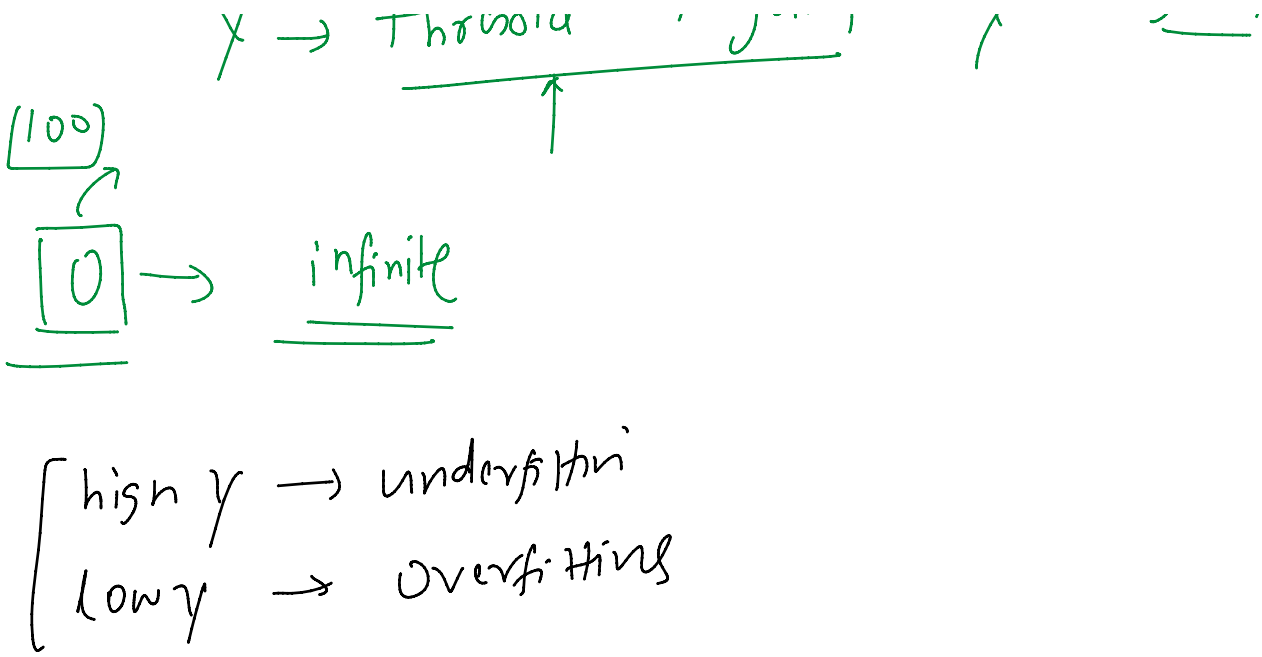
$\gamma = 500$

300-500 = -200

γ

600 - 500 = 100 +ve

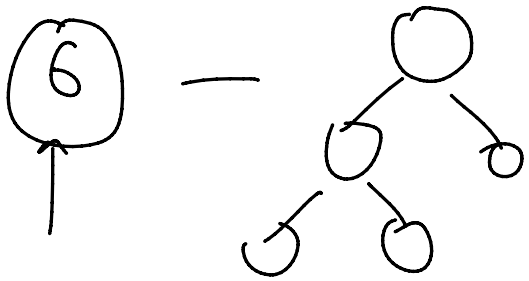
$\gamma \rightarrow$ threshold on gain $\gamma \rightarrow$ gain



high γ → underfitting
low γ → overfitting

2. Max Depth

30 March 2024 23:27



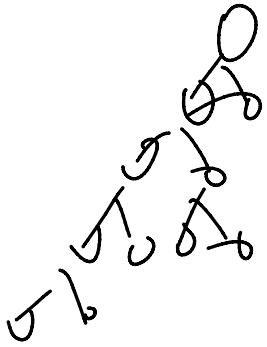
max_depth = 6

max depth = 2 { pre pruning }

why do need max_ [gamma]

computation → max. dept = Xone

1000 estimate

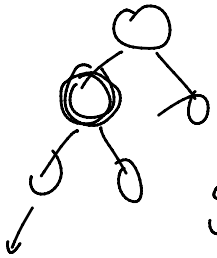


3. Min Child Weight → pruning → simpler trees

30 March 2024 23:27

tricky → complex

cover → boosting



gradient (g)
hessian (h)

cover define → sum of hessians

gradient → derivative

$L \rightarrow \hat{y}$

$\frac{\partial L}{\partial \hat{y}}$ gradient (g)

xgboost

$\frac{1}{2}(y_i - \hat{y}_i)^2$

$\frac{\partial^2 L}{\partial \hat{y}^2}$ → hessian (h)

rcg

rcg

$h_i = 1$

classifi

last stage probability

$h_i = p_i(1-p_i)$

sum of resid

$$w = \frac{\text{sum of residual}}{\text{\# of residual} + \lambda}$$

$$w = \frac{\text{sum of resid}}{p(1-p) + \lambda}$$

sum of hessians

$1 \times n = n$

min child weight → pruning parameter

→ data points

cover

$h_1 + h_2 + h_3 + h_4 - h_4$ sum of hessians

min child weight

Reg → $h_i = 1$

Classification

cover

[cover]

?

reg → cover is nothing but number of datapoints

min child weight = 100 →

cover > min child weight

reg case

min child weight → min_samples_split (res)

min child wt ↑ overfit in ↓

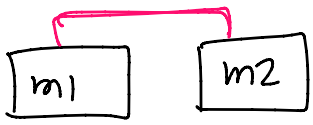
Classification

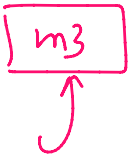
$h_i = p_i(1-p_i)$

↳ prev stage prob

res

$h_i = 1$





cgpa	placement	pred	res	pred2	res
8	1	0.5	0.5	0.7	0.3
7	0	0.5	-0.5	0.3	-0.3
4	0	0.5	-0.5	0.2	-0.2
5	1	0.5	0.5	0.8	0.2

(1)

$h_i = p_i(1-p_i) = 0.7(1-0.7) = 0.21$

$0.3(1-0.3) = 0.21$

$0.2(1-0.2) = 0.16$

$0.2(1-0.2) = 0.16$

0.16

$hi \rightarrow \max\ (0.25)$

$1\ (1-1)$

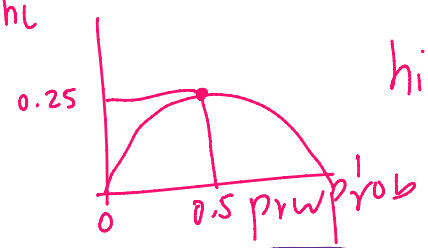
$0, 1, 0.5, 0.6, 0.3$ $\quad 0.5$

4 models

1 2 3 4 5 $\rightarrow$ $h_i = 0$ $\quad 0.25$

$h_i = 0$

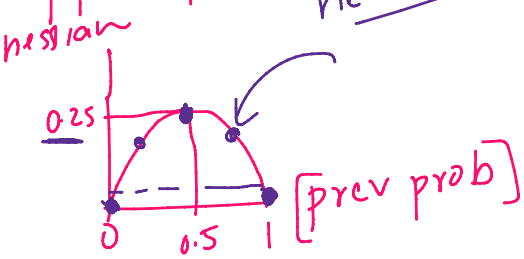
combination h_i



100 dts

$h_i = p_i(1-p_i)$

hessian



prev prob

$x_i \rightarrow$ 0.2 $\rightarrow$ 0 1

0 1

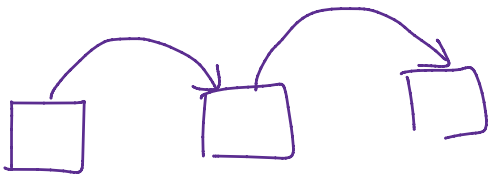
0.5

$x_i \rightarrow 0.9$ 1

0.5

$x_i \rightarrow$ 0.5 uncertain

0.5



hessian gives a quantitative value to each data point

hessian of

to each data point

↓

explain its importance

0.25 ↑ → next day

0 → not so imp

1 → 0.25

2 → 0.1

10

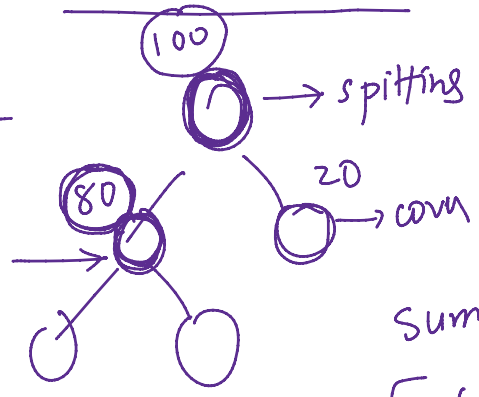
3 → 0

4 → 0

sum of hessians ↑ [gain]

↳ importance

classification



cover

↓

sum of hessians

collective number

sum of hess (cover)

[min child weight] → 10

↓

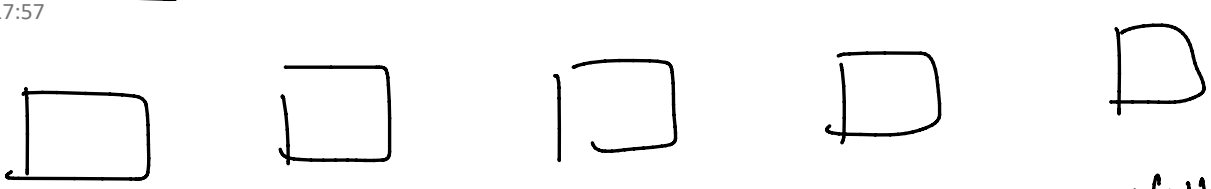
threshold on

importance of

data

4. Num Estimators

02 April 2024 17:57



n_estimator ↑ (1000) → overfitting

n_estimator ↓ (1) → Underfitting

5. Early Stopping

02 April 2024 08:28

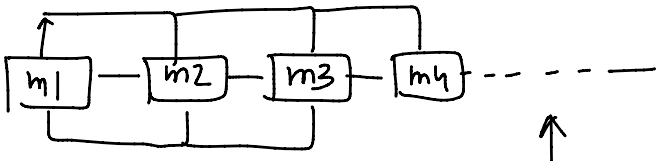
early stopping

XGboost

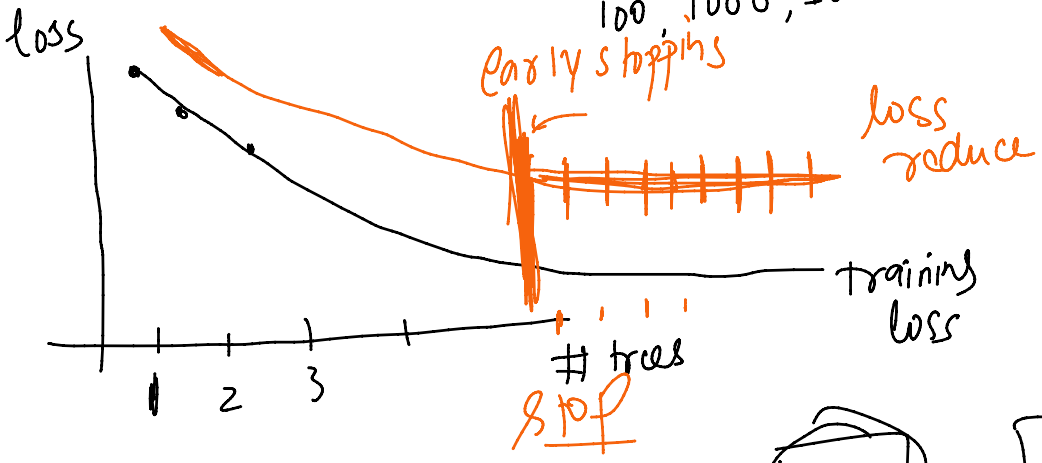
Train loss

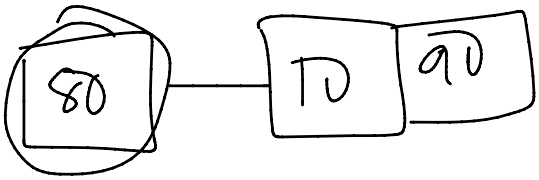
test loss

10



100, 1000, 10000





80 - 10 = 70 mode

patience 80 model

80 81 ... 90

X

80 →

6. Shrinkage

30 March 2024 23:26

[5.23)

m1 lr * m2

1 1

best possible number

1 2 3 4 5 6 [5.23]

0.1

0.01

1. 1.1 1.2 1.3 ... 2 2.1 2.2 ...

5.1 5.2 5.3

1 1.01 1.02 ... 1.23 ... 2 2.01 ...

3 .. 5.01 ... 5.23

lr → ↑ (1)

lr → ↓ 0.00001 5 6

5.23

0.3 ← 0.0

0.3

7.5 + 5 = 12.5

7.5 + 0.3 × 5 = (9)

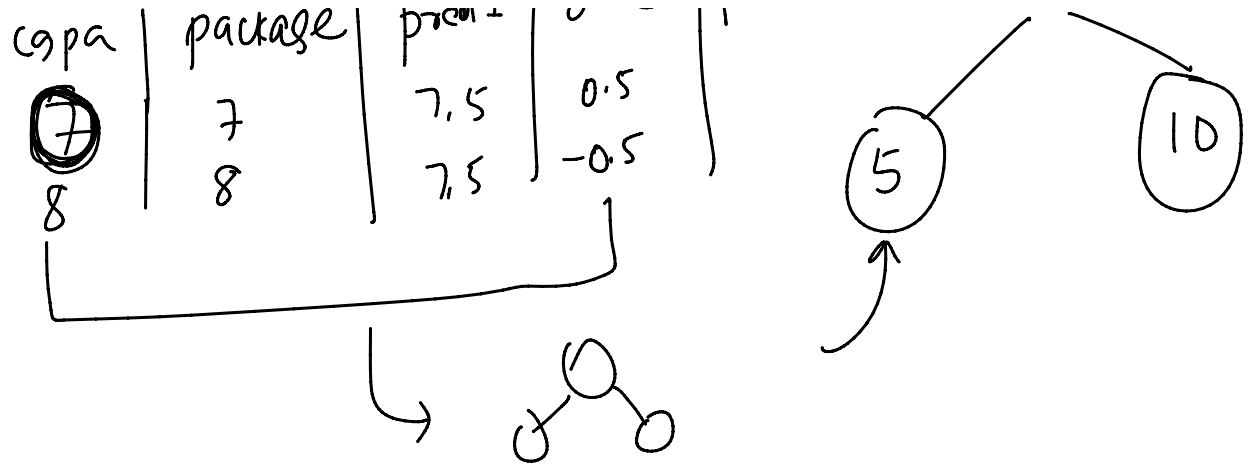
7.5 + 0.5

m1 lr * m2

7.5 dt

if cgpa < 7.5

cgpa	package	pred1	res(1	pred
			0.5	


cgpa
package
7
8
7
8
7.5
7.5
0.5
-0.5
5
10

7. Lambda (and Alpha)

30 March 2024 23:26

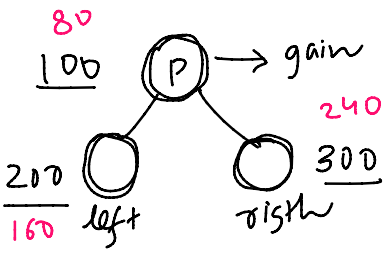
$$L = \mathcal{L}(y_i, \hat{y}_i) + \Omega(f_t(x_i)) \qquad \alpha$$

$$\gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

$\lambda = 0$ $\boxed{\lambda = 10}$

$$\text{similarity score} = \left[\frac{(\text{sum of residual})^2}{\#\text{residuals} + \lambda}\right] \rightarrow \text{gain}$$

$\lambda \uparrow$ SS $\downarrow$



$$SS_{left} + SS_{right} - SS_P$$

$$\text{output } w^* = \frac{\text{sum of residual}}{\#\text{residuals} + \lambda}$$

$\lambda \uparrow \downarrow$ underfitting overfitting

$200 + 300 - 100 = \boxed{400}$ $\lambda = 0$

$160 + 240 - 80 = 320$ $\lambda = 10$

$\boxed{\lambda \uparrow}$ pruning ($\uparrow$)

↳ pruning

$\boxed{\gamma} = \boxed{100}$ – fixed

$\boxed{\text{gain}} - \gamma$ (100) → node prune

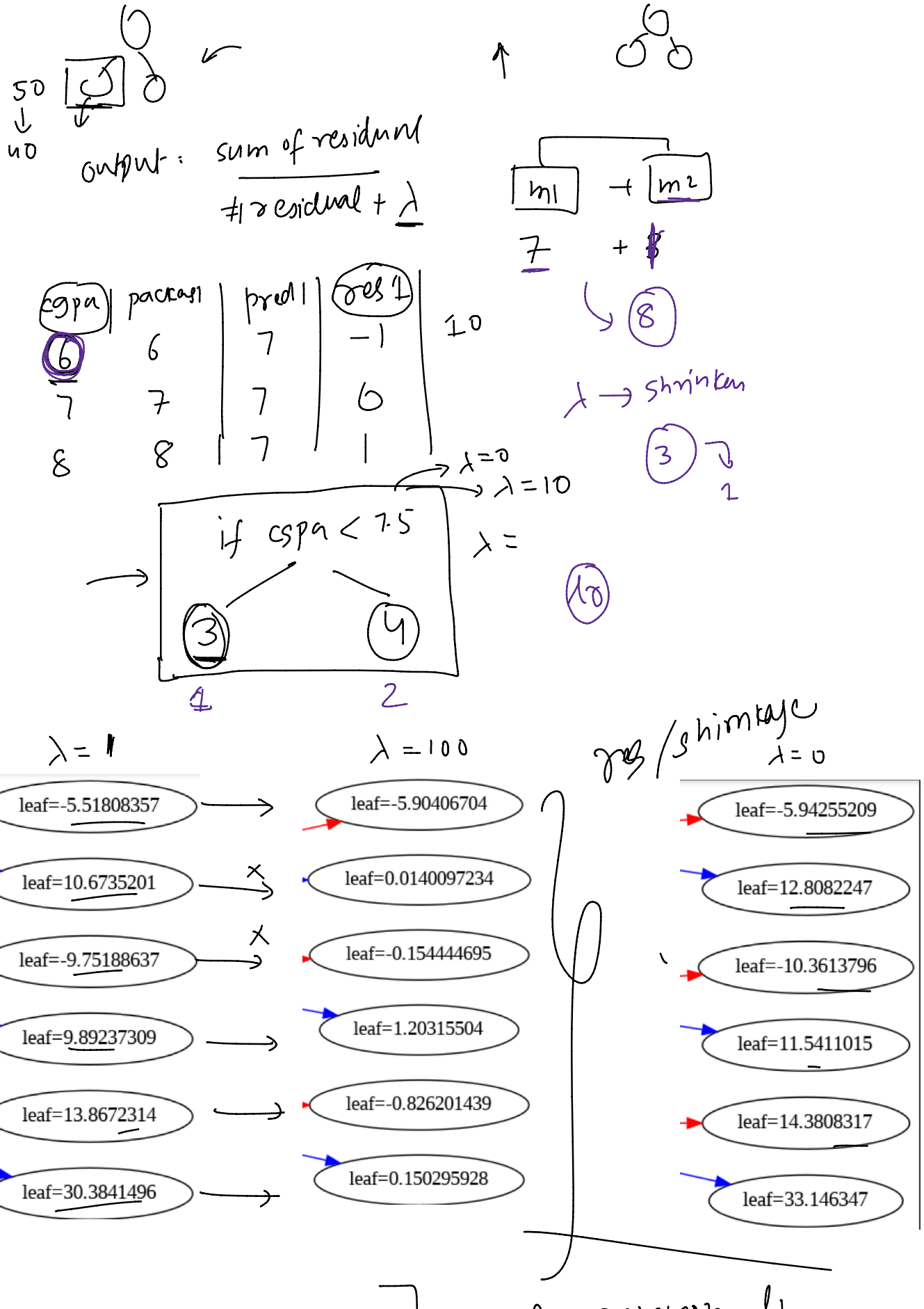
reduce → $\lambda \uparrow$

$\lambda \uparrow$ gain ($\downarrow$) γ constant

prune ← node → gain -ve

$\lambda = 0$ $\lambda = 10$

output $\downarrow$


50
40
output : sum of residual / # residual + λ
m1 + m2
7 + 8
8
λ → shrinkage
3
1
cgpa | package | pred1 | res1
6 | 6 | 7 | -1
7 | 7 | 7 | 0
8 | 8 | 7 | 1
10
if cspa < 7.5
λ=0
λ=10
λ =
10
3
4
4
2
λ = 1
λ = 100
gb / shrinkage
λ = 0
leaf=-5.51808357
leaf=10.6735201
leaf=-9.75188637
leaf=9.89237309
leaf=13.8672314
leaf=30.3841496
leaf=-5.90406704
leaf=0.0140097234
leaf=-0.154444695
leaf=1.20315504
leaf=-0.826201439
leaf=0.150295928
leaf=-5.94255209
leaf=12.8082247
leaf=-10.3613796
leaf=11.5411015
leaf=14.3808317
leaf=33.146347
common

$\lambda \uparrow$ overfitting ↓

$\lambda \downarrow$ overfitting ↑

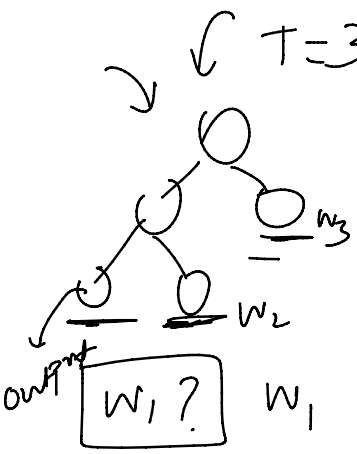
l_2 common l_1

λ ↑ small

$$L^{<t>} = L(y_i, \hat{y}_i) + \Omega(f_t(x_i))$$

$$\gamma T + \frac{1}{2} d \sum_{i=1}^{I} w_i^2$$

weight l_2



$$\frac{1}{2}\left[w_1^2 + w_2^2 + w_3^2\right]$$

reg_lambda → l_2 reg

reg_alpha →

↓ → l_1

$$\gamma T + \frac{1}{2} \alpha \sum_{i=1}^{T} \|w_i\|$$

l_1 reg

$$\frac{1}{2} \alpha \left[|w_1| + |w_2| + |w_3|\right]$$

λ, α → loss

l_2 reg >> l_1 reg

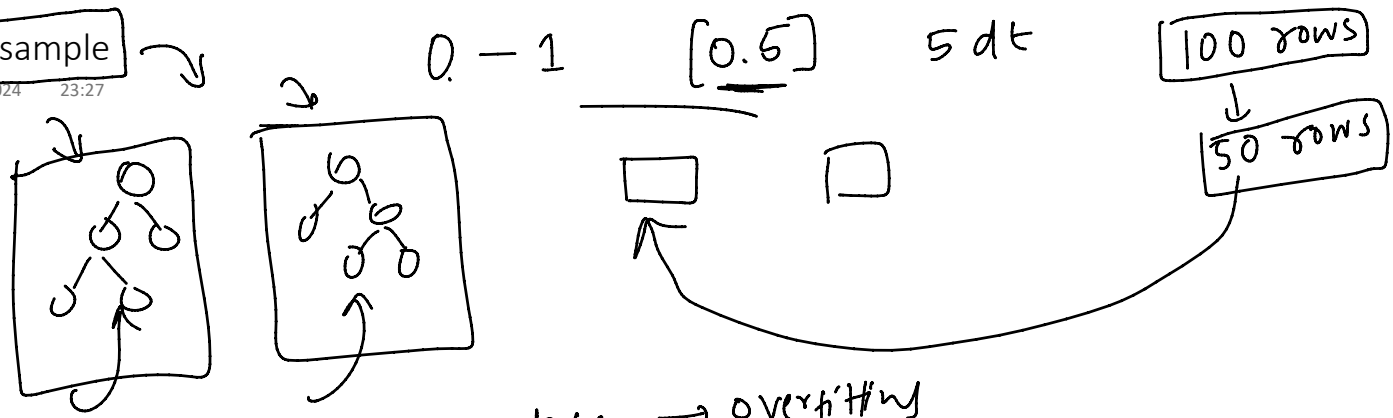
elastic net

$$\gamma T + \frac{1}{2}\lambda \sum_{i=1}^{T} w_i^2 + \frac{1}{2}\alpha \sum_{i=1}^{T} |w_i|$$

8. Subsample

30 March 2024 23:27

0 – 1 [0.5] 5 dt



subset of the data → overfitting reduce

0.11 – 11.1.

1 → 100% 0.5

Subsample (1) → 100%

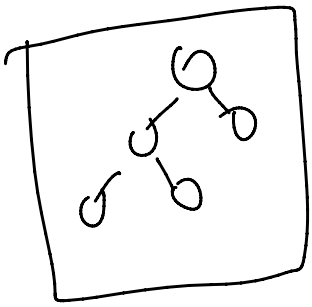
0.1

[0.5 – 0.8]

9. Col Subsample

30 March 2024 23:28

rows → features



$f_1\ f_2 \cdots f_{10}$

→ $f_1\ f_2\ f_3\ f_4\ f_6$

$f_{10}\ f_7\ f_4\ f_3\ f_1$

col subsample

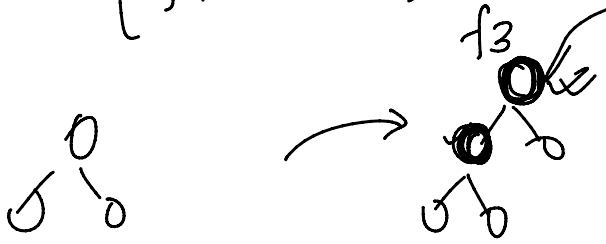
→ [by tree] 0-1

[by level] 0-1

by node 0-1

[colsample_by tree] = 0.5

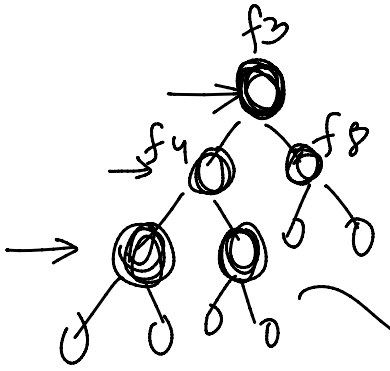
$[f_1 \cdots f_{10}]$ → 5 features → $f_1\ f_3\ f_5\ f_7\ f_9$



colsample_by level [0-1]

↳ 0.5 ↘

$f_1\ f_3\ f_5\ f_7\ f_9$



$f_1\ f_3\ f_5\ f_7\ f_9$

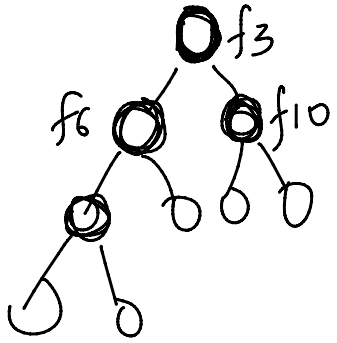
$\rightarrow f_2\ f_4\ f_6\ f_8\ f_{10}$

$f_1\ f_5\ f_6\ f_7\ f_{10}$

by tree **by level**

colsample_by node [0-1] → random forest

↘ 0.5



$f_1\ f_2\ f_3\ f_4\ f_5$

$f_1\ f_5\ f_6\ f_7\ f_8$

$f_6\ f_7\ f_8\ f_9\ f_{10}$

by tree by level **by node**

How XGBoost Handles Missing Values?

04 April 2024 17:40

Cache Aware Access

04 April 2024 17:40

XGBoost Hyperparameters

04 April 2024 09:21

Hyperparameter Group	Hyperparameter	Description	Possible Values	General Advice
Tree Complexity and Structure	max_depth	Maximum depth of each tree.	Any positive integer	Controls overfitting; deeper trees capture more complex patterns.
	max_leaves	Maximum number of terminal nodes in a tree.	Any positive integer or 0 (no limit)	More leaves increase model complexity and flexibility.
	max_bin	Maximum number of bins for histogram-based methods.	Any positive integer	More bins allow finer feature splits but increase computation.
	max_delta_step	Maximum step for each tree's weight estimation.	Any non-negative float	Helps in logistic regression with imbalanced data. Rarely needed otherwise.
	grow_policy	Strategy to grow trees.	'depthwise', 'lossguide'	Determines how the tree nodes are expanded.
	tree_method	Algorithm for tree construction.	'auto', 'exact', 'approx', 'hist', 'gpu_hist'	Choose based on dataset size and computational resources.
	num_parallel_tree	Number of trees grown per round (for forests).	Any positive integer	More trees increase ensemble's robustness but

		per round (for forests).		ensemble's robustness but also computational load.
	interaction_constraints	Constraints on feature interactions.	List of pairs of feature indices	Limits interactions between features, based on domain knowledge.
Regularization	gamma	Minimum loss reduction required to	Any non-negative float	Controls complexity; higher values
	min_child_weight	Minimum sum of instance weight in a child.	Any non-negative float	Higher values prevent overfitting by avoiding creating nodes with few samples.
	reg_alpha	L1 regularization term on weights.	Any non-negative float	Encourages sparsity in the leaf weights, helping to regularize.
	reg_lambda	L2 regularization term on weights.	Any non-negative float	Smoothens the leaf weights to prevent overfitting.
	monotone_constraints	Monotonicity constraints on features.	Dictionary or string	Enforces the model to learn a monotonic relationship with the target variable.

Sampling and Column Subsampling	subsample	Subsample ratio of the training instances.	Float between 0 and 1	Lower values can help prevent overfitting but too low can cause underfitting.
	colsample_bytree	Subsample ratio of columns when constructing each tree.	Float between 0 and 1	Controls the number of features considered by each tree, influencing overfitting.
	colsample_bylevel	Subsample ratio of columns for each depth level.	Float between 0 and 1	Adjusts feature sampling as trees grow deeper, impacting model complexity.
	colsample_bynode	Subsample ratio of columns for each split.	Float between 0 and 1	Fine-tunes the amount of feature consideration at each node split.

	sampling_method	Method to sample training instances.	'uniform', 'gradient_based'	'gradient_based' can focus on harder to learn samples, affecting model focus.
Boosting and Learning	n_estimators	Number of boosting rounds or trees.	Any positive integer	More rounds increase complexity; should be tuned with learning rate.
	learning_rate	Step size shrinkage used in update to prevent overfitting.	Any positive float	Lower rate requires more trees but can yield better generalization.
	booster	Type of model to run at each iteration.	'gbtree', 'gblinear', 'dart'	Selects the type of model to use at each step of boosting.

Objective and Evaluation	objective	The loss function to be minimized.	String or custom function	Should align with your task (regression, classification).
	eval_metric	Evaluation metric for validation data.	String, list of strings, or custom function	Should reflect the objective of the model; used for tuning and early stopping.

Miscellaneous Controls	scale_pos_weight	Balancing of positive and negative weights.	Any positive float	Important for imbalanced datasets to balance the influence of classes.
	base_score	The initial prediction score.	Any float	Generally, the default is sufficient; it's the initial prediction for all instances.
	random_state ✓	Random number seed.	Any integer or None	Ensures reproducibility of your model's results.
	missing	Value to represent missing data.	Typically `np.nan`	Ensures the model correctly handles missing values in your data.

	n_jobs	Number of parallel threads.	Any non-negative integer	Optimize to leverage hardware capabilities for faster training.
	verbosity	Degree of verbosity.	Integer from 0 (silent) to 3 (debug)	Useful for monitoring the training process.
	validate_parameters	Whether to validate input parameters.	True or False	Helps in identifying potentially incorrect parameter settings.
	predictor	Type of predictor algorithm to use.	'cpu_predictor', 'gpu_predictor'	Useful when explicitly specifying to use CPU or GPU resources.
Early Stopping and Callbacks	early_stopping_rounds	Rounds without improvement to stop training.	Any positive integer	Prevents overfitting by stopping training when validation metric stops improving.
	callbacks	Custom callback functions for training.	List of callback functions	Allows custom operations to be applied at each training iteration.

Categorical Features and GPU Support	enable_categorical	Support for categorical features.	True or False	Utilize to handle categorical features directly without preprocessing.
	gpu_id	ID of the GPU to use.	Any valid GPU ID	Specifies the GPU to use for training, if applicable.
	max_cat_to_onehot	Threshold for one-hot encoding for categorical features.	Any positive integer	Decides when to use one-hot encoding based on the number of categories.
	max_cat_threshold	Maximum categories considered for each split in partition-based trees.	Any positive integer	Helps to prevent overfitting with high cardinality categorical features.