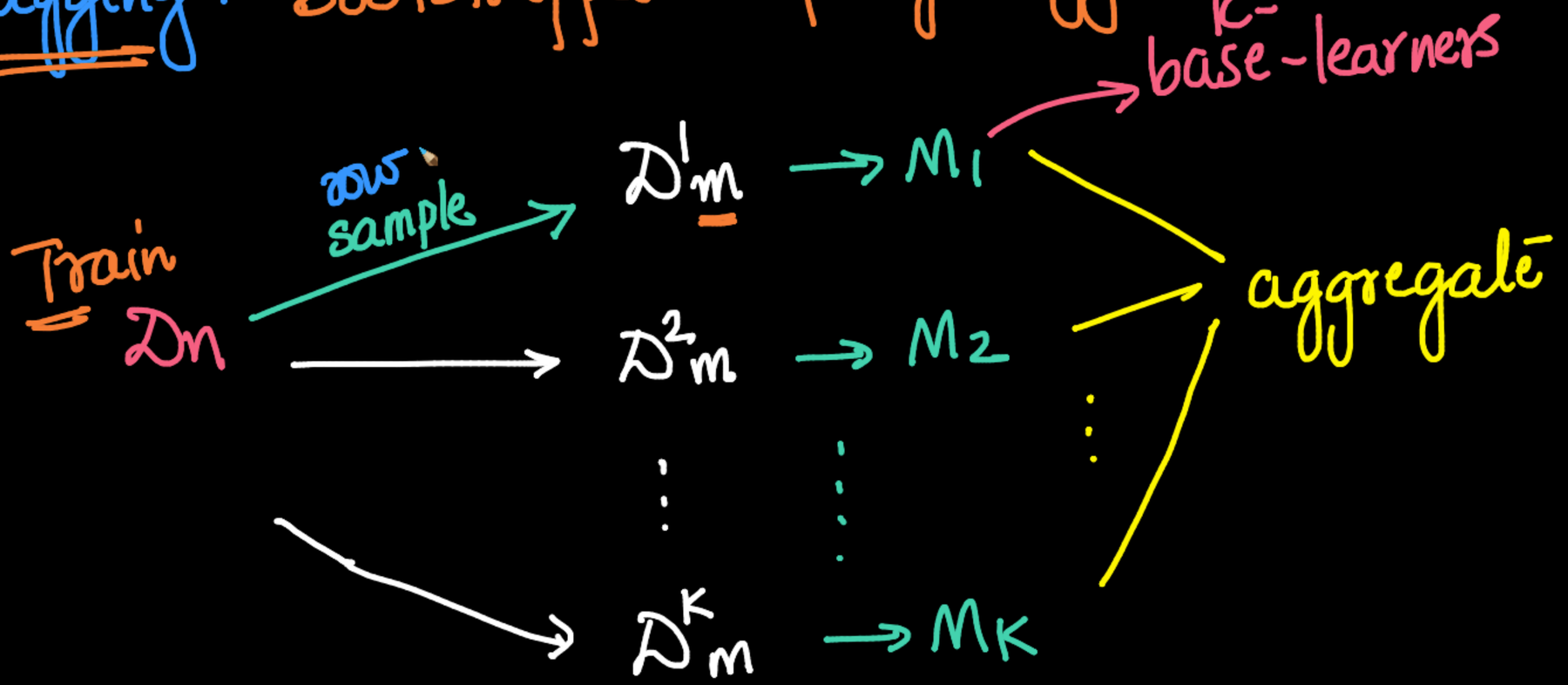
Topics:

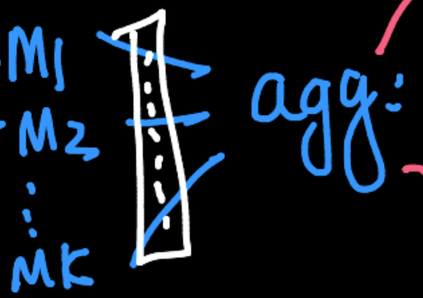
- Random forest
- Extra Trees
- Boosting (intuition)
- Gradient Boosting
- code (HR-domain)

Chat : interaction :--

Bagging : Bootstrapped sampling + agg...



Let $K = 100$



agg: Majority vote : classfn.

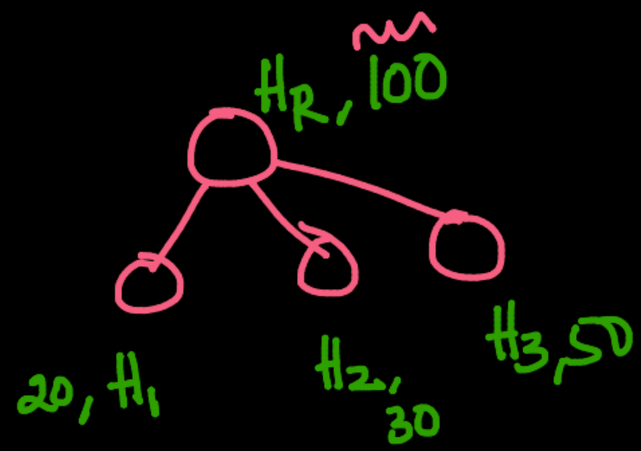
→ mean/median : reg...

please revise

Entropy : $-\sum_{i=1}^{K} p(y_i) \log(p(y_i))$

G.I : $1 - \sum_{i=1}^{K} (p(y_i))^2$ ✓

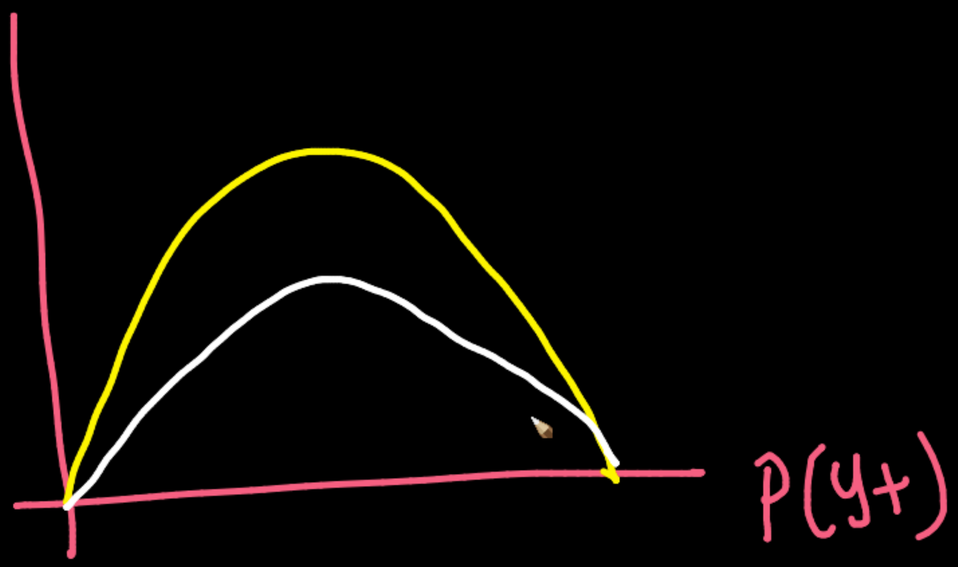
IG → Δ Entropy or Δ G.I



IG = H_R (higher)

$0.2 H_1 + 0.3 H_2 + 0.5 H_3$

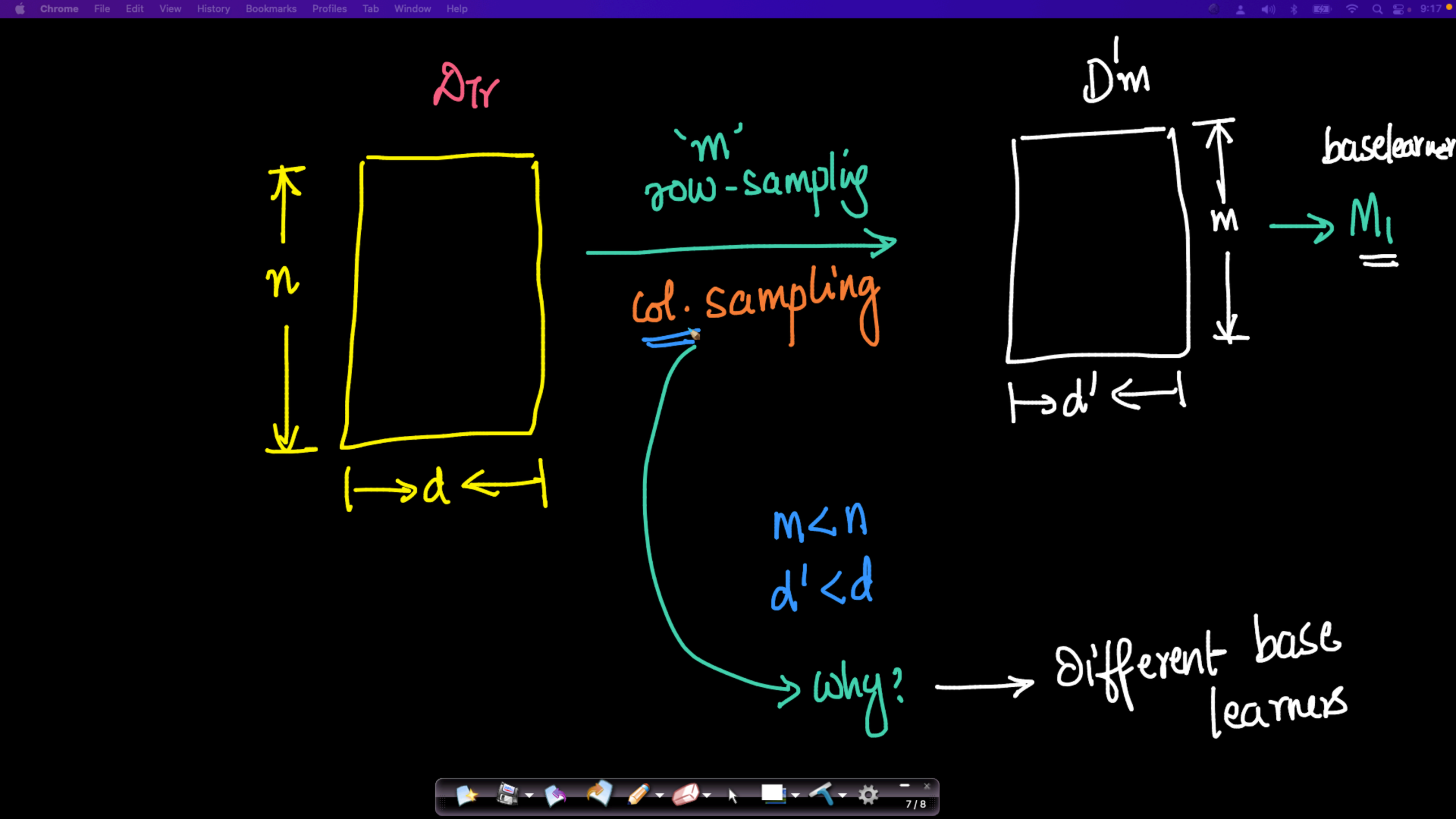
weighted H across all child nodes

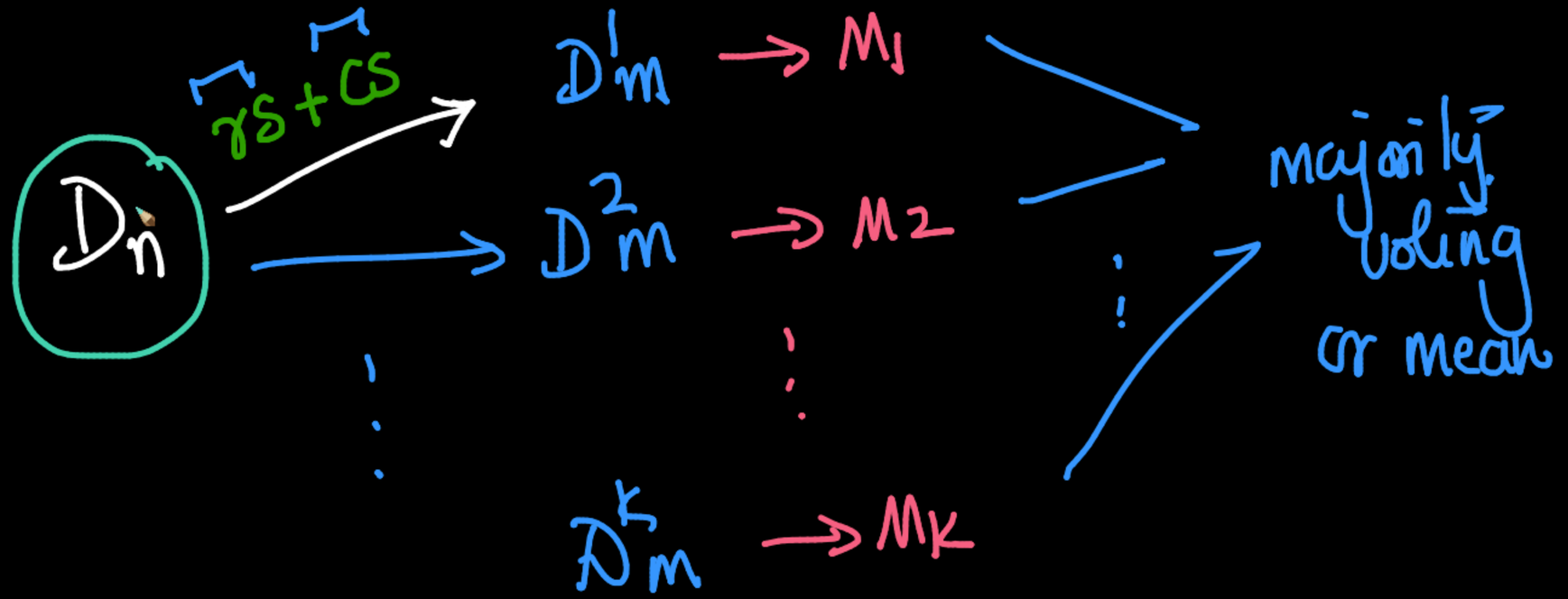

P(y+)

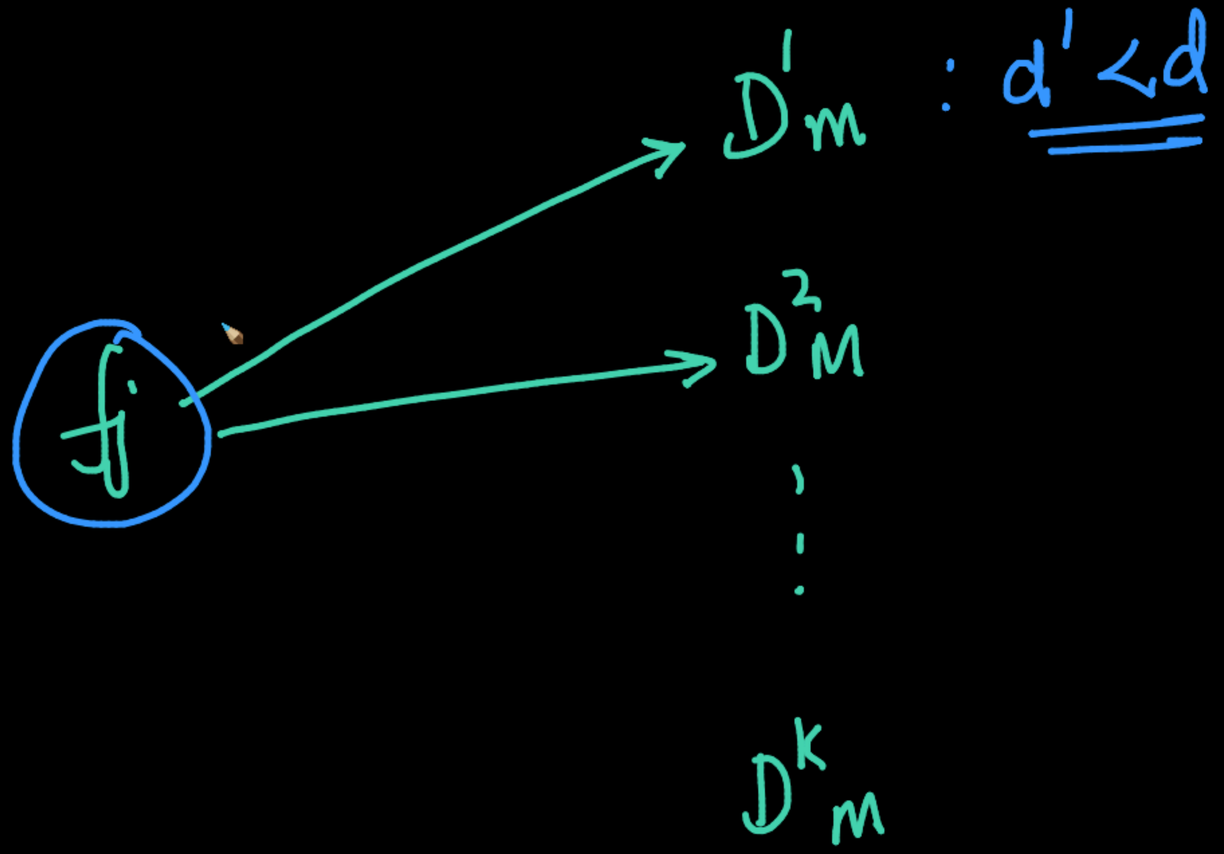
Random Forest

Bagging with DT (base-learners)

↳ Random + ensemble of trees + Bagging --


D_{Tr}
n
d
'm'
row-sampling
col. sampling
D'_m
m
d'
baselearner
M_1
m<n
d'<d
why?
Different base learners


D_n
$\gamma s + cs$
$D'_m \rightarrow M_1$
$D^2_m \rightarrow M_2$
$D^k_m \rightarrow M_k$
majority voting
or mean


D'_m : $d' < d$
D^2_M
D^k_m

Train : D_n

n-rows:

OOB D^i_{n-m} CV

$D^i_{n-m} = D_n - D^i_m$

Set-difference

D^1_{n-m} D^2_{n-m} D^i_{n-m} D^K_{n-m}

Train

base-learner

$D^1_m \rightarrow M_1$

$D^2_m \rightarrow M_2$

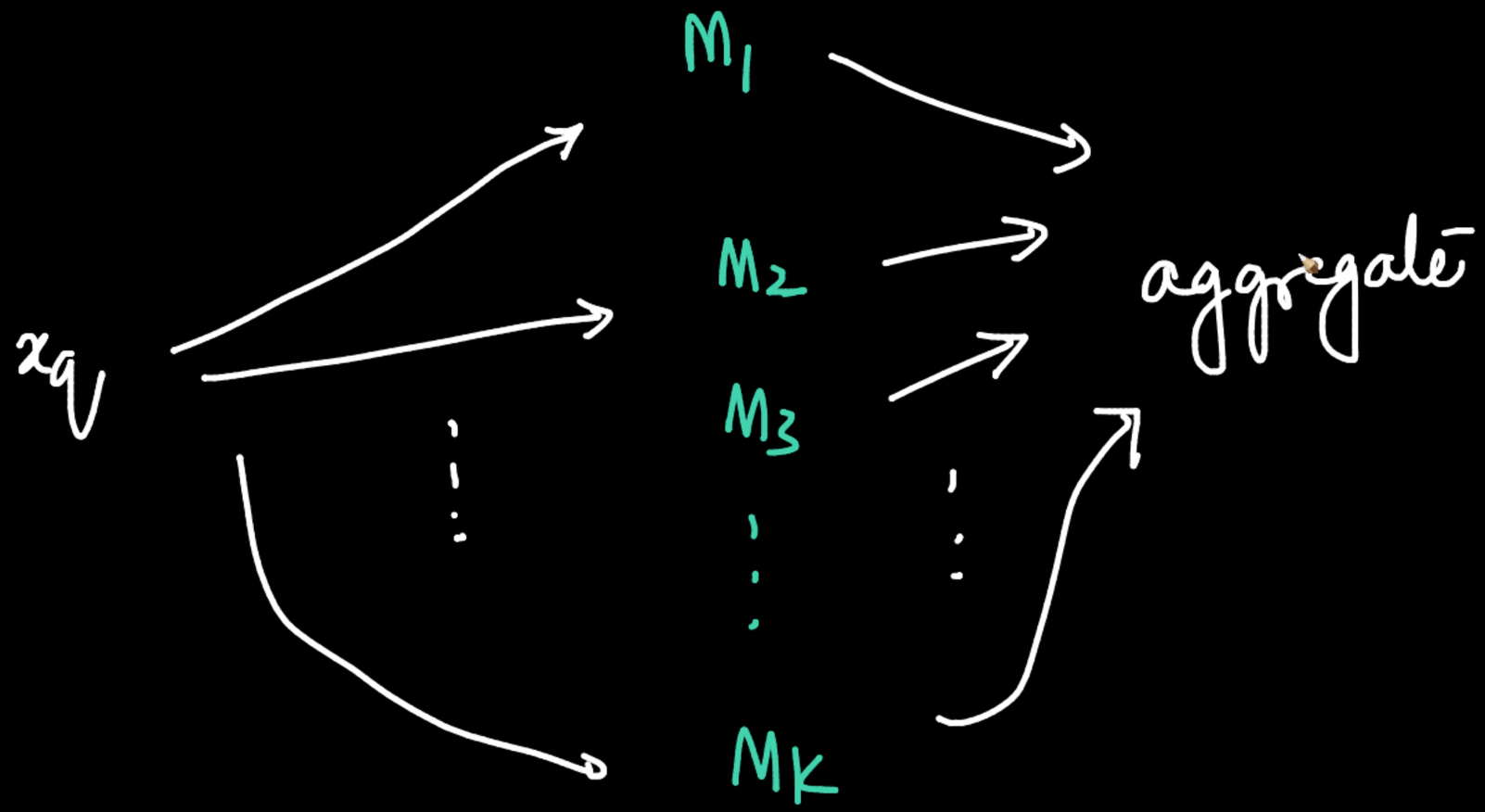
$D^i_m \rightarrow M_i$

$D^K_m \rightarrow M_K$

[TRAINING]

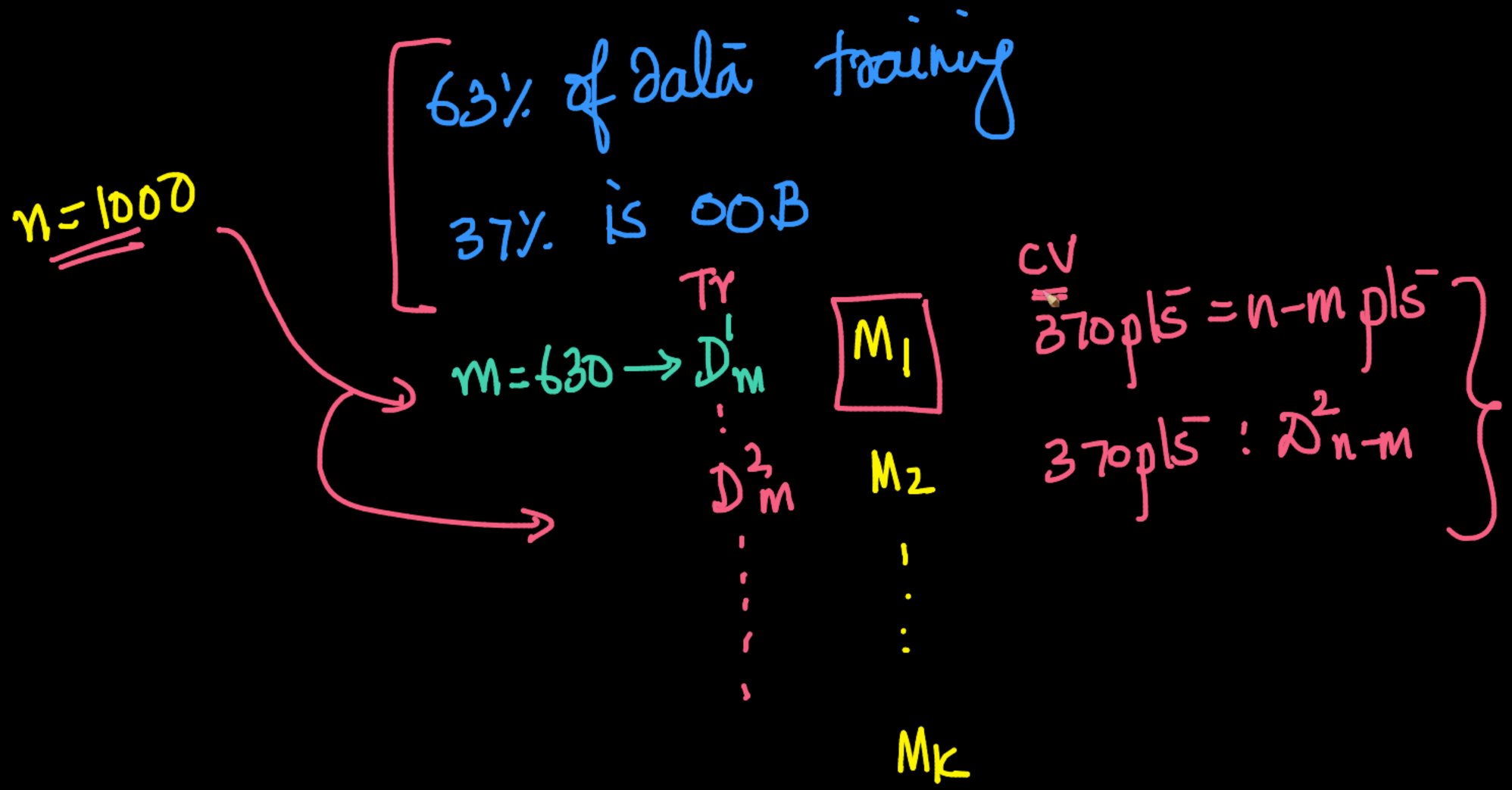
out of bag OOB D^1_{n-m}

Test-time



Feature Engineering is a must → Classical ML

↳ skip it in DL

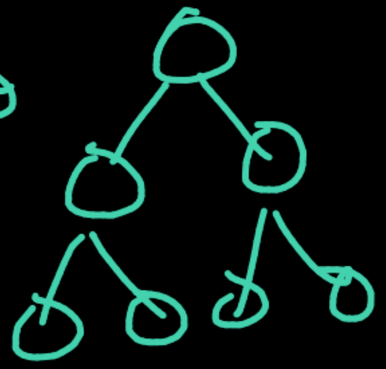

63% of data training
37% is OOB
n = 1000
m = 630
Tr
D'_m
D^2_m
M_1
M_2
M_K
CV
370pts = n-m pts
370pts : D^2_{n-m}

Simpler-model if it works:

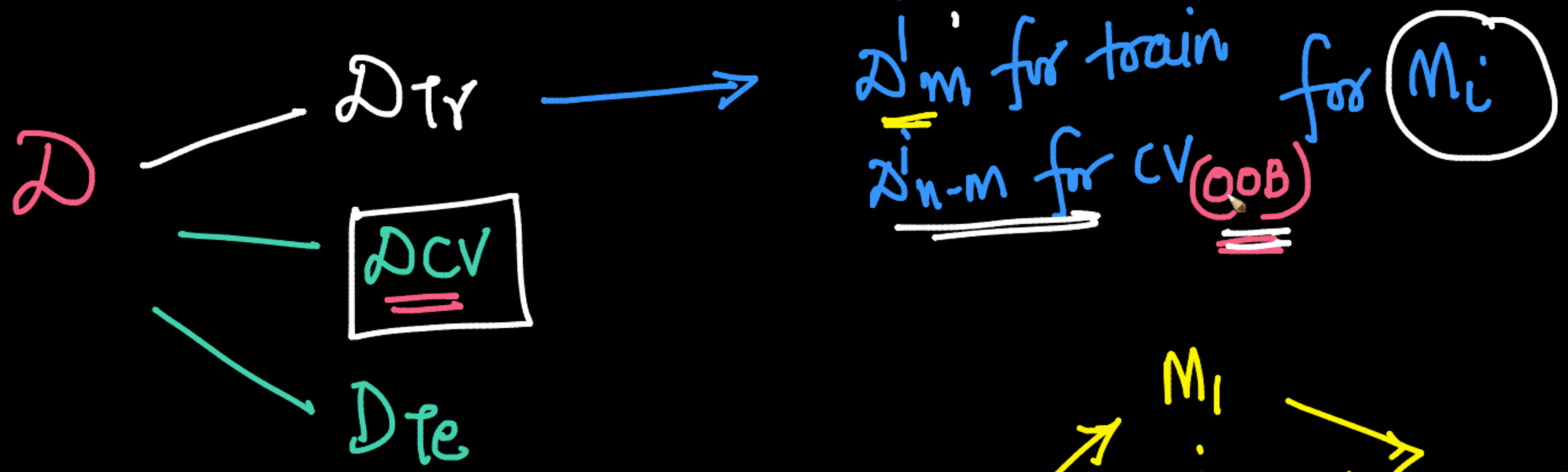
RF better than DT (most-cases)

When do DT & RF have similar perf

simple task :



v.good



hyper-param
RF

RF

each base-learner M_i is a DT — n-rows & d'-cols

↓

deep DT (non-shallow)

↳ overfit slightly

↓

Sub sample of rows & cols

RF: → rigorous Stat. proof

base-learners: overfit → high-variance & low bias on a subsample of rows & cols..

↓

(aggregation / averaging)

↳ reduce variance without much inc in bias

Var↓ bias →

Statistical ML

(Research Sessions)

Error (model) = $Bias^2$ + Var + irreducible error

Typically: Var ↓ Bias ↑

tradeoff

- Bias ↑ : underfit
- Var ↑ : overfit

RF :-

1. Bagging : rs + CS
2. Base learners: overfit (high var)
3. aggregation :- Var ↓ bias →

⇓

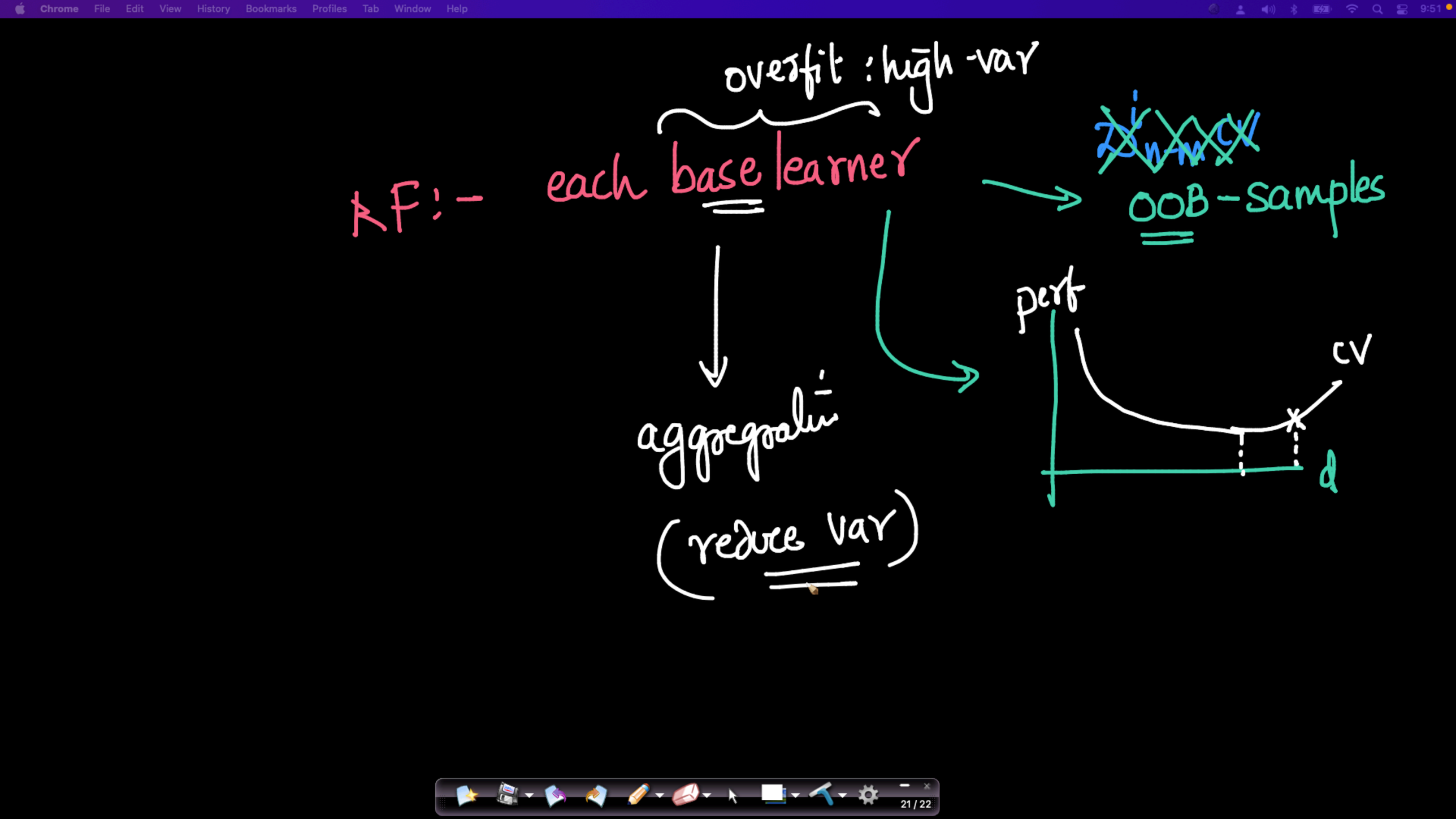
better overall model

RF: non-DT models ✓

use models with high-var (overfit)

↳ KNN with small K

↳ logistic reg with higher order features + low λ


overfit : high-var
RF :-
each base learner
OOB-samples
aggregation
(reduce var)
perf
CV
d

hyper-params: tuning of RF:

trees :- K ↑

(base learners)

avoid overfitting

generalize better

D → Dtr, Dcv, Dte

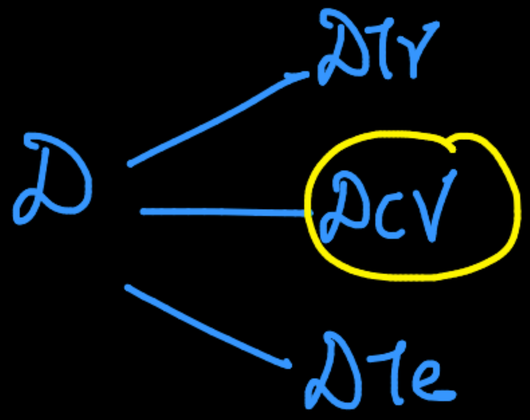
aggregation :- Var ↓ ; bias →

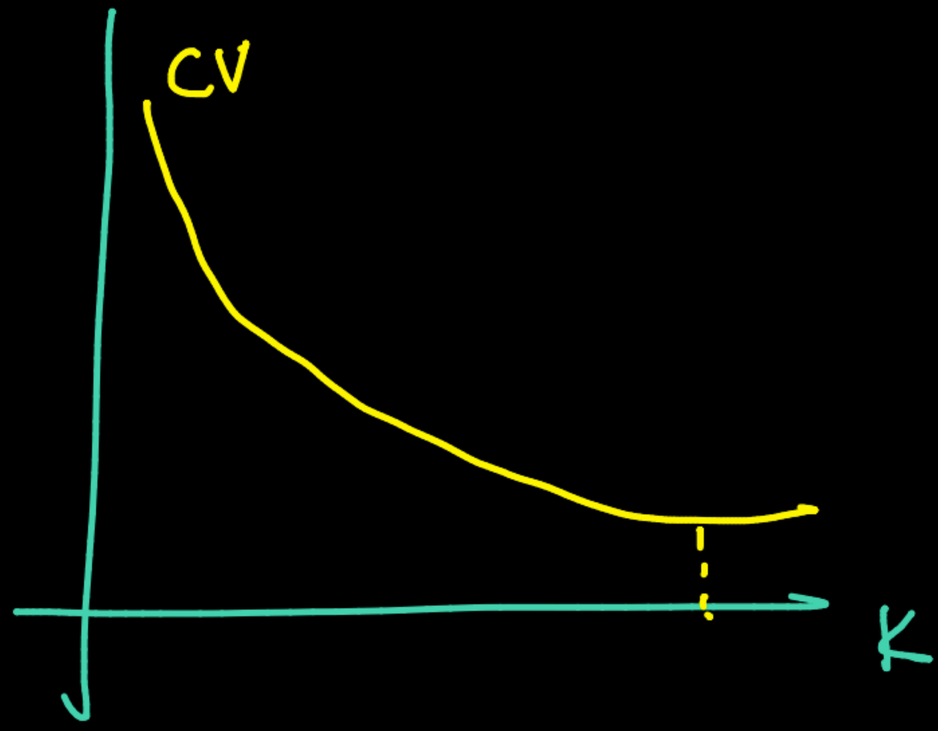
$K = 1 \rightarrow$ overfit slightly or massively ...

(OOB)

100/1000 K = V·large $\rightarrow$ avoid overfilling

$$\begin{cases} 1B = n \\ m = 100 \end{cases}$$





K is a hyperparam fw RF

base-learner :- decision stump

underfit = low var & high bias (def)

RF agg :- ↓ var + bias →

↳ will not work as expected

hyper-params:-

1. # base learners = K ↑ → avoids overfitting

2. m = Sample size (row) → all base learners are going to overfit on all data

m/n

m/n ↑ → $m = n$ → overfitting ↑

(3) d' = # cols sampled

$\frac{d'}{d}$ ↑ ⇒ overfitting ↑

(4) depth of base learners

↳ depth ↑ overfit

OOB X Samples CV

depth ↑ → $\frac{m}{n}$ ↓ $\frac{d'}{d}$ ↓ K ↑ = less chance of overfit

aggregation:

outlier (x)

seen only some base-learners

Viz a RF is tricky

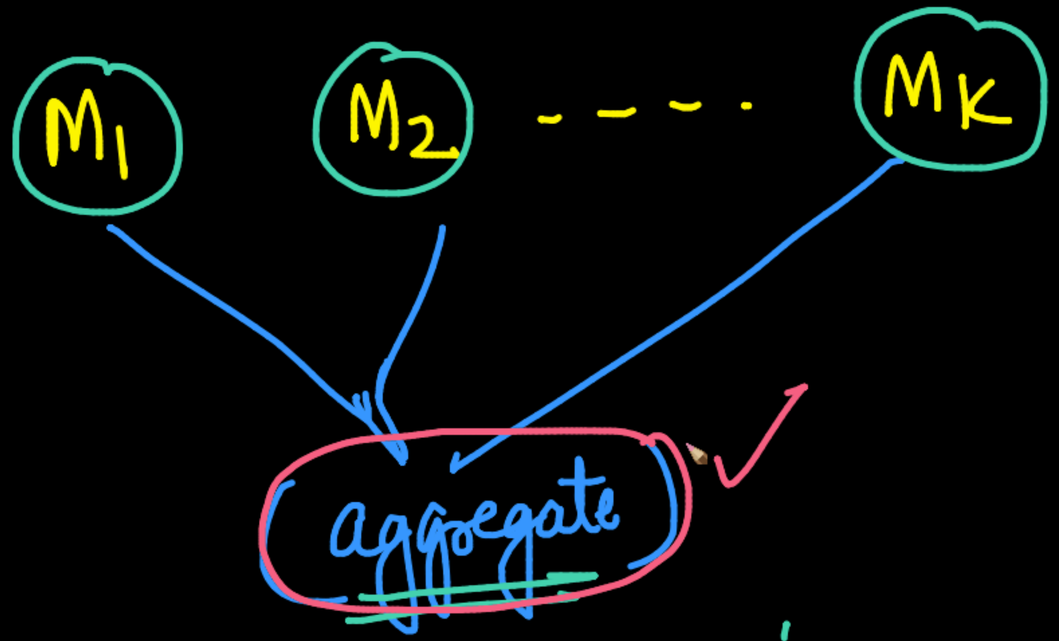
d = 3

d' = 2

1000 base-learners

n = 1Million

overfit on a subsample of some size

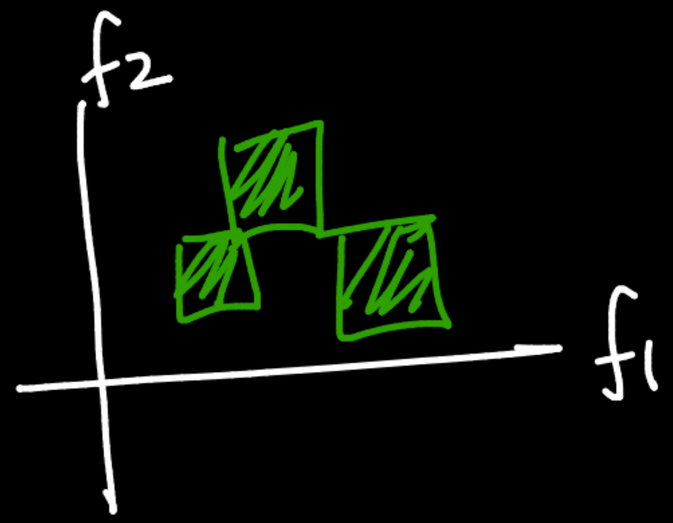


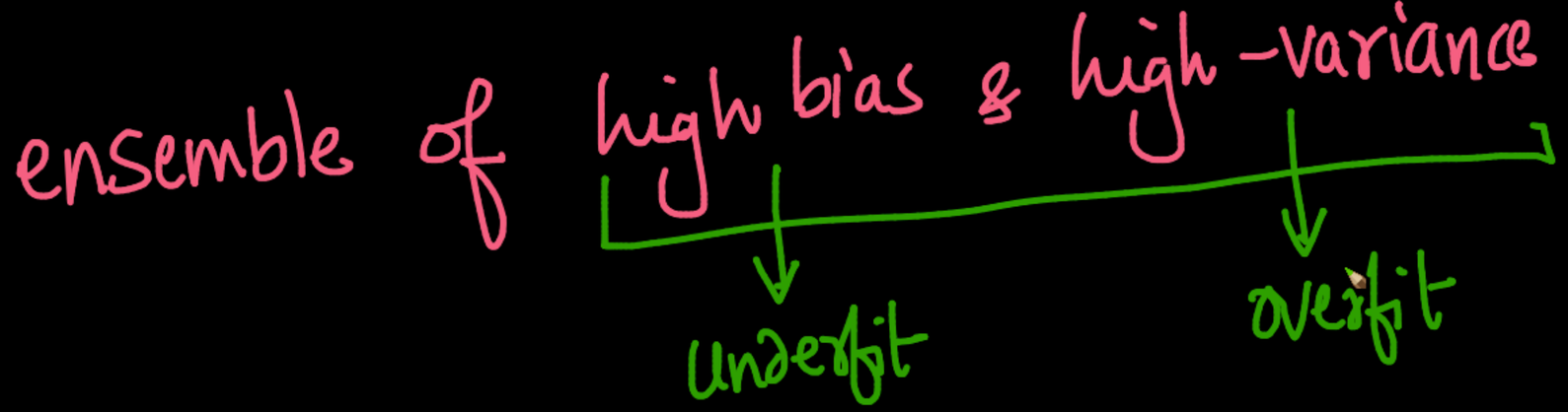
reduce overfitting ⇒ var ↓ bias →

base learners ↑

K = 1 base-learner — overfit

DT → deep tree ⇒




ensemble of
high bias & high -variance
underfit
overfit

Bagging
RF: → ensemble of high-var

agg: Var ↓ bias →

GBDT: } → ensemble of high bias models..
Boosting }

DT — large depth — high var

Linear-Classifier — Underfit — high bias

agg — var ↓ — bias →

K: # base learners ↑

less overfitting

Typically: non-ensemble models

Var ↓ bias ↑

how size of sample :

$\frac{m}{n}$ ↓

↳ smaller data

{ overfit the base learner } Var ↑↑

more base learners K ↑

Var ↓

$m \approx n \Rightarrow$ base learners are similar

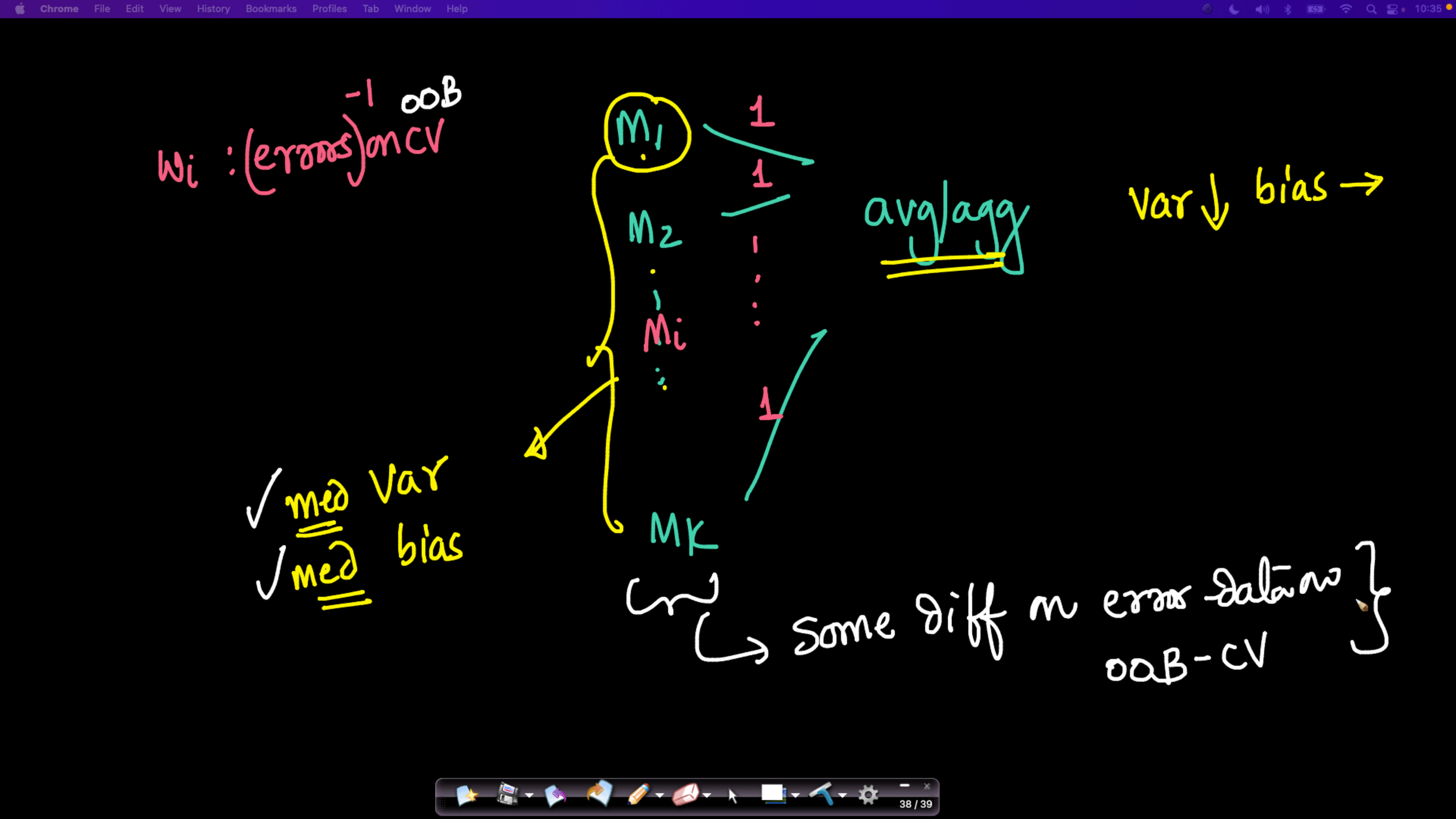
$\frac{m}{n} \uparrow$ RF will overfit

(agg) doesn't work well

model DT

$n = 1$ → trivially overfit

$n \downarrow$ easily overfit


-1 OOB
Wi : (errors) on CV
M1
M2
Mi
MK
1
1
1
avg/agg
var ↓ bias →
✓ med var
✓ med bias
Some diff in error data w
OOB-CV
38 / 39

DT

BETTER

high-var
low bias

M_i's → agg

var ↓ bias → low

WORSE

med-var
med-bias

M_i's → agg

var ↓ bias: → Med

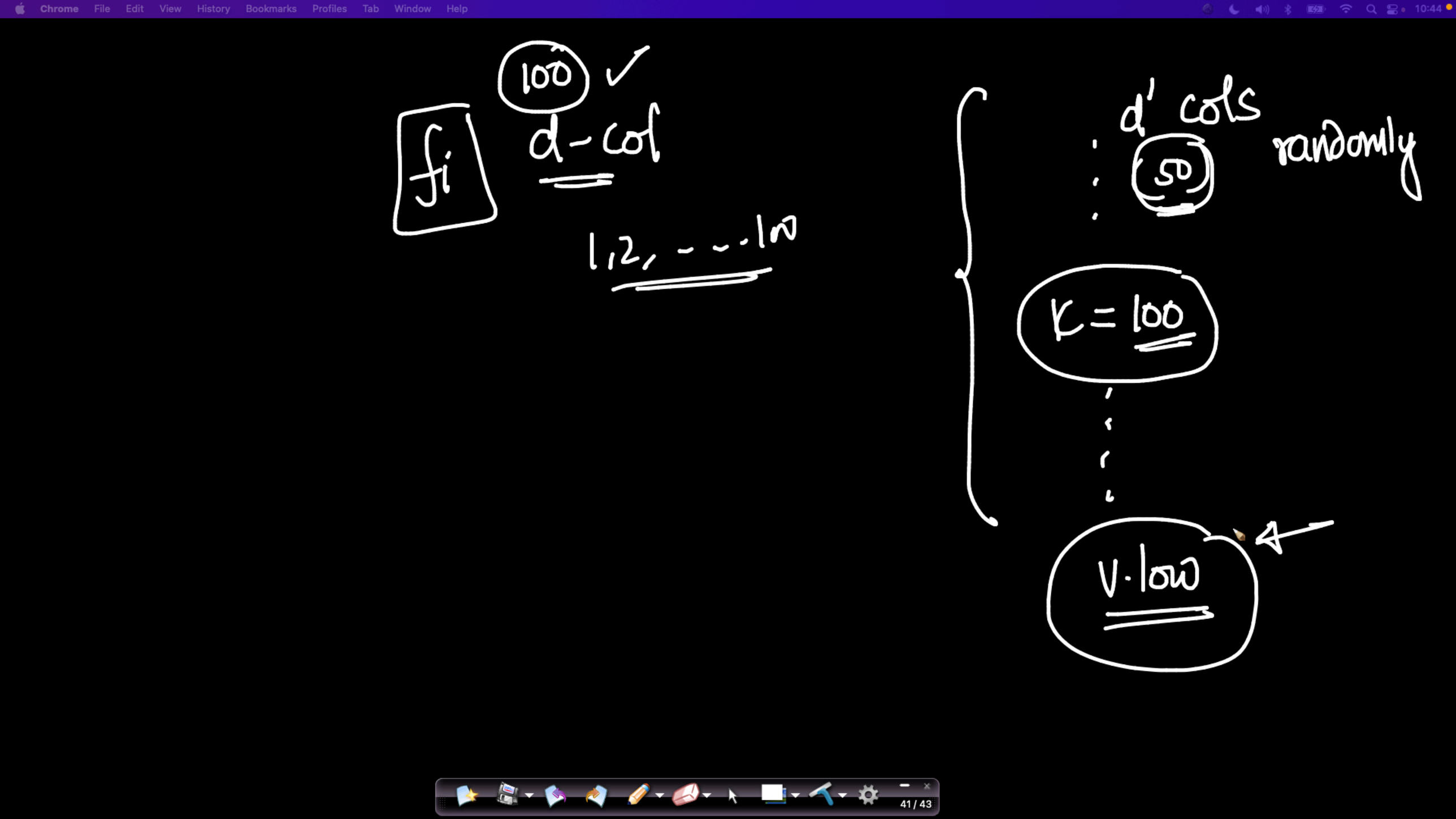
$$E = \text{bias}^2 + \text{var} + \text{irr. error}$$

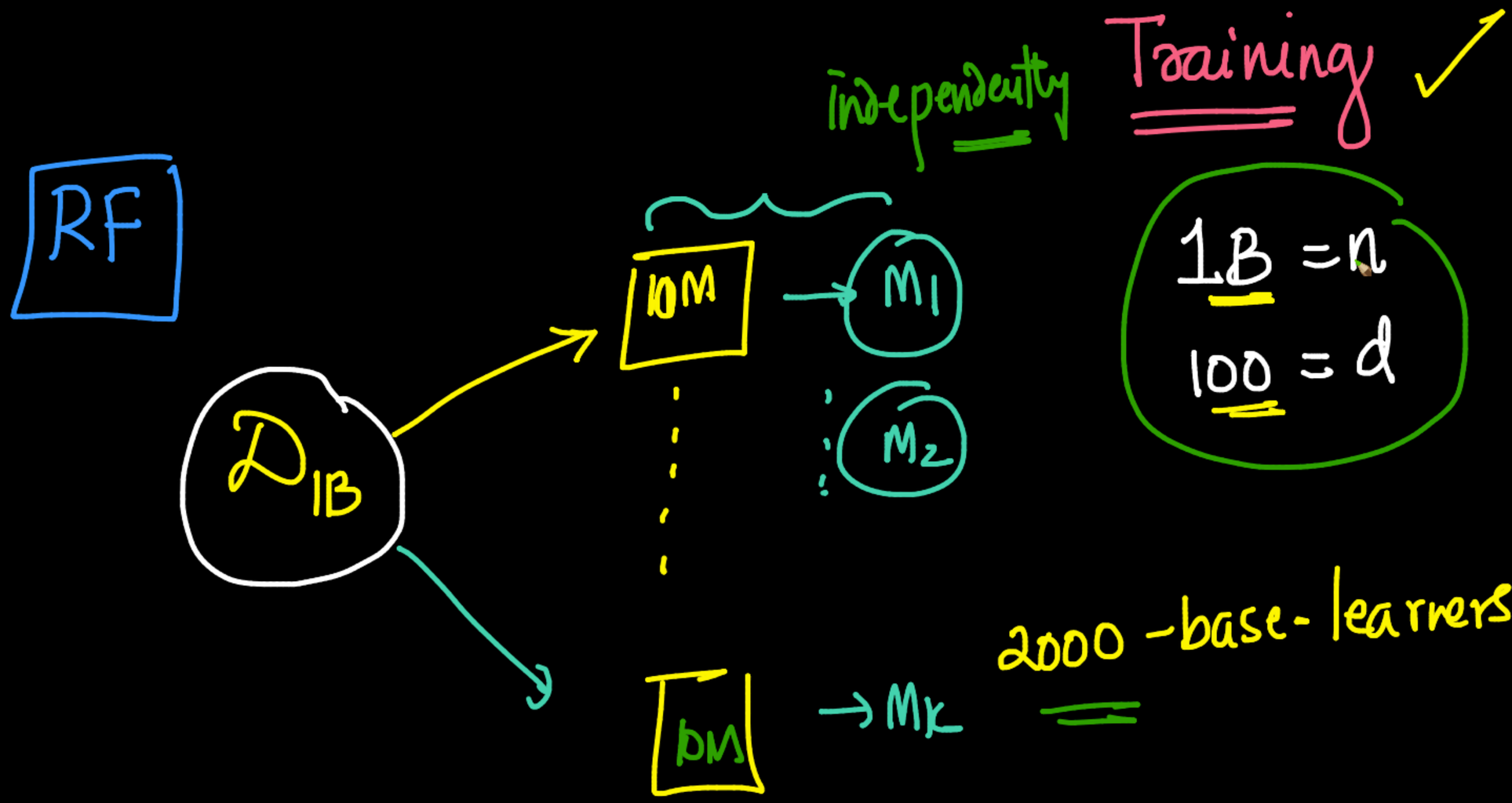
RF → ~~OOB~~ for CV

max depth = 15

min depth = 5

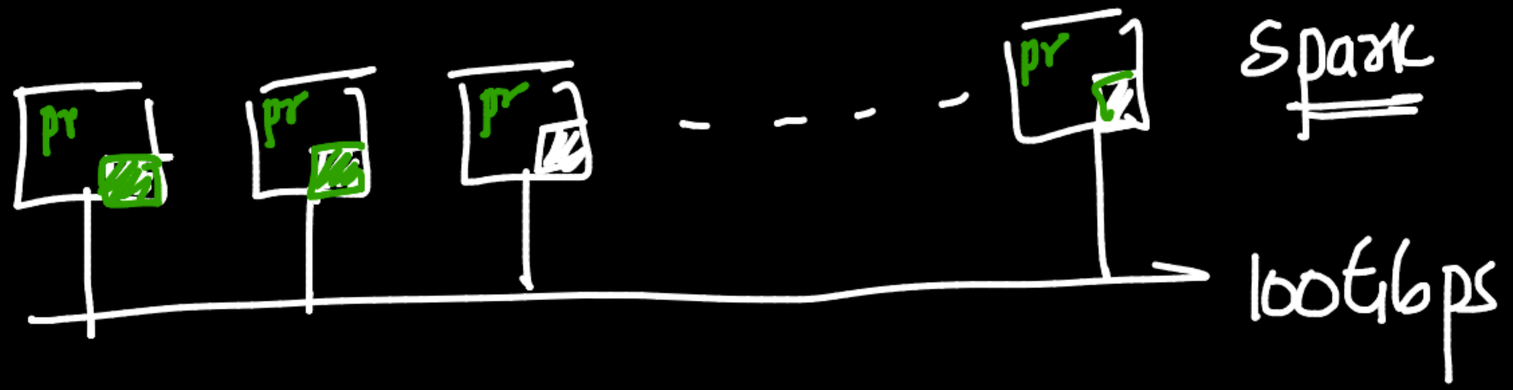
base-learner

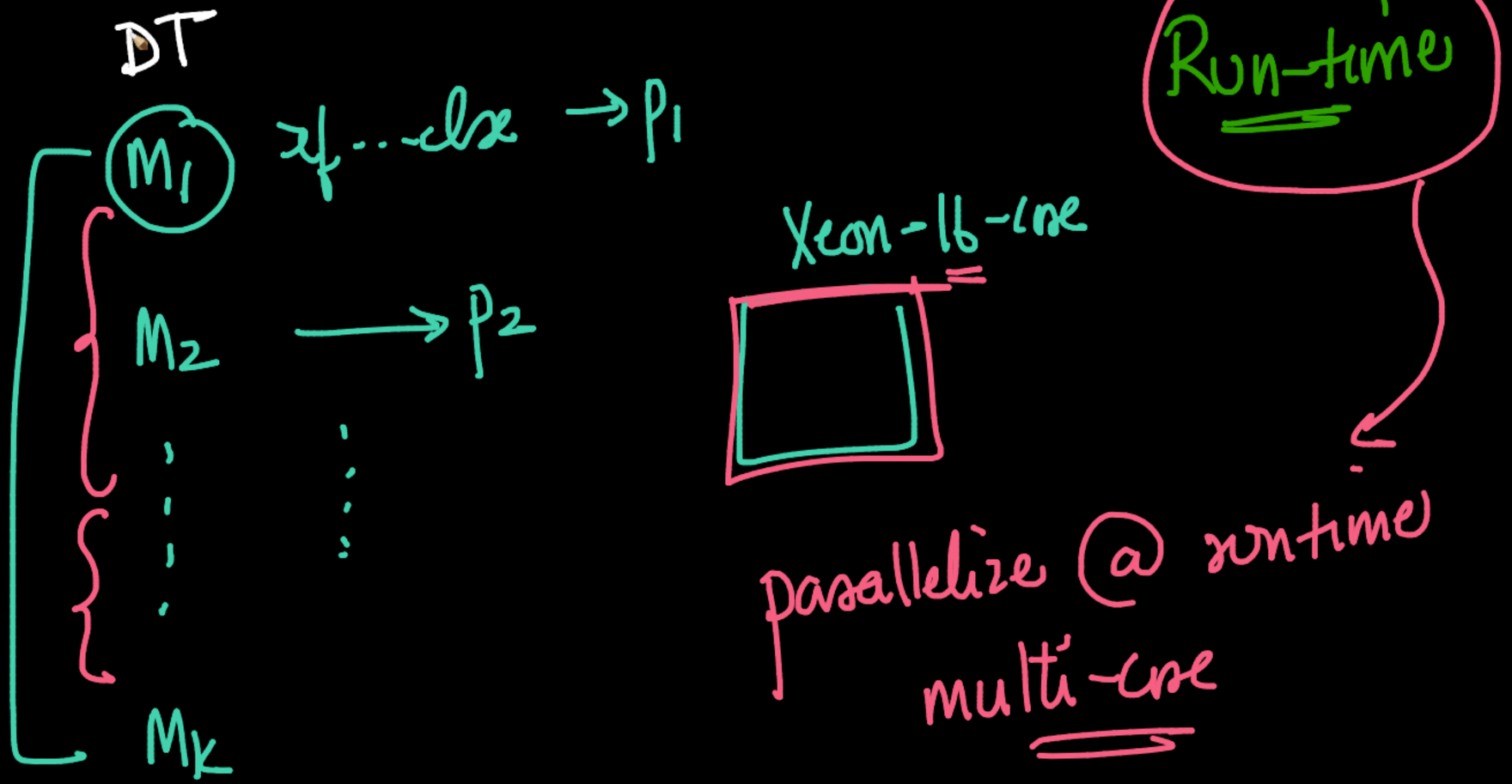

100
d-col
1, 2, ---. 100
d' cols
randomly
SD
K = 100
V. 100


RF
D_{IB}
DM
DM
M_1
M_2
M_k
independently
Training ✓
1B = n
100 = d
2000 - base - learners

trivially parallelizable & very fast to train

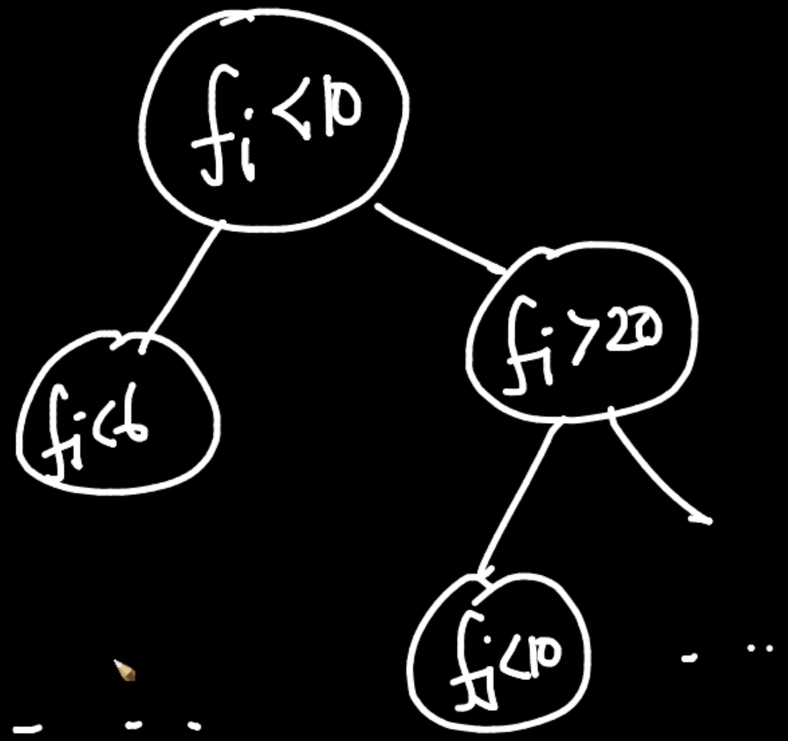
8-core



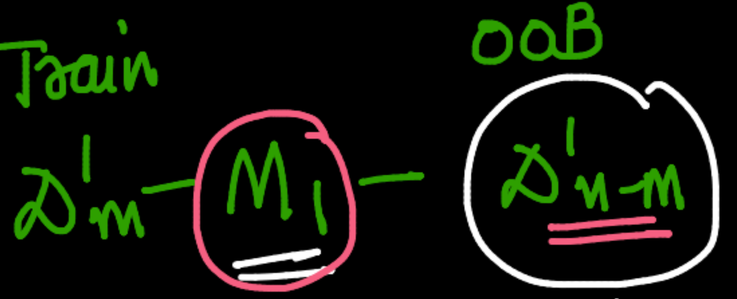


$$O(k \times \overbrace{\text{depth-max}})$$

max-depth ≠ d'


fi<10
fi<6
fi>20
fi<10

OOB-Score:



→ not used a CV data

perf. of M_1 on D'_{n-m}: OOB_1

OOB-Scores

Train-data perf. $>$ $\widetilde{OOB_1}$

overfit

Var↑ low bias

min error

max perf

not the best impl

Prev Up Next

scikit-learn 1.0.2
Other versions

Please cite us if you use the software.

sklearn.ensemble.RandomForestClassifier
Examples using sklearn.ensemble.RandomForestC

sklearn.ensemble.RandomForestClassifier

class sklearn.ensemble.RandomForestClassifier(*n_estimators=100*, ***, *criterion='gini'*, *max_depth=None*, *min_samples_split=2*, *min_samples_leaf=1*, *min_weight_fraction_leaf=0.0*, *max_features='auto'*, *max_leaf_nodes=None*, *min_impurity_decrease=0.0*, *bootstrap=True*, *oob_score=False*, *n_jobs=None*, *random_state=None*, *verbose=0*, *warm_start=False*, *class_weight=None*, *ccp_alpha=0.0*, *max_samples=None*) [source]

Xgboost ← GBDT

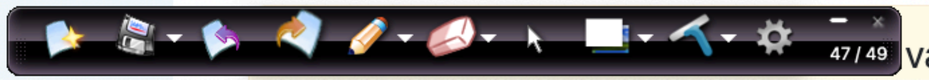
A random forest classifier.

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the `max_samples` parameter if `bootstrap=True` (default), otherwise the whole dataset is used to build each tree.

Read more in the User Guide.

Parameters:

n_estimators : *int, default=100*
The number of trees in the forest.



value of `n_estimators` changed from 10 to

sklearn.ensemble.RandomForestClassifier

class sklearn.ensemble.RandomForestClassifier(*n_estimators=100*, ***, *criterion='gini'*, *max_depth=None*, *min_samples_split=2*, *min_samples_leaf=1*, *min_weight_fraction_leaf=0.0*, *max_features='auto'*, *max_leaf_nodes=None*, *min_impurity_decrease=0.0*, *bootstrap=True*, *oob_score=False*, *n_jobs=None*, *random_state=None*, *verbose=0*, *warm_start=False*, *class_weight=None*, *ccp_alpha=0.0*, *max_samples=None*) [source]

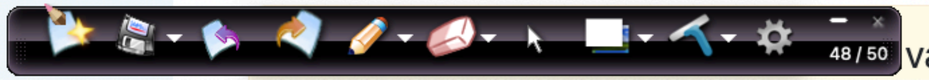
A random forest classifier.

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the `max_samples` parameter if `bootstrap=True` (default), otherwise the whole dataset is used to build each tree.

Read more in the User Guide.

Parameters:

n_estimators : *int, default=100*
The number of trees in the forest.



value of `n_estimators` changed from 10 to

sklearn.ensemble.RandomForestClassifier

class sklearn.ensemble.RandomForestClassifier(*n_estimators=100*, ***, *criterion='gini'*, *max_depth=None*, *min_samples_split=2*, *min_samples_leaf=1*, *min_weight_fraction_leaf=0.0*, *max_features='auto'*, *max_leaf_nodes=None*, *min_impurity_decrease=0.0*, *bootstrap=True*, *oob_score=False*, *n_jobs=None*, *random_state=None*, *verbose=0*, *warm_start=False*, *class_weight=None*, *ccp_alpha=0.0*, *max_samples=None*) [source]

X OOB as CV

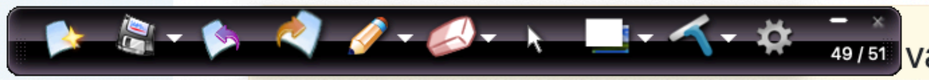
A random forest classifier.

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the `max_samples` parameter if `bootstrap=True` (default), otherwise the whole dataset is used to build each tree.

Read more in the User Guide.

Parameters:

n_estimators : *int, default=100*
The number of trees in the forest.



value of `n_estimators` changed from 10 to

sklearn.ensemble.RandomForestClassifier

high

class sklearn.ensemble.RandomForestClassifier(*n_estimators=100*, ***, *criterion='gini'*, *max_depth=None*, *min_samples_split=2*, *min_samples_leaf=1*, *min_weight_fraction_leaf=0.0*, *max_features='auto'*, *max_leaf_nodes=None*, *min_impurity_decrease=0.0*, *bootstrap=True*, *oob_score=False*, *n_jobs=None*, *random_state=None*, *verbose=0*, *warm_start=False*, *class_weight=None*, *ccp_alpha=0.0*, *max_samples=None*) [source]

base-learners - overfit

A random forest classifier.

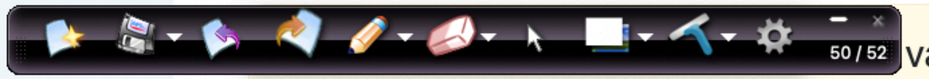
A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the `max_samples` parameter if `bootstrap=True` (default), otherwise the whole dataset is used to build each tree.

K ↑ for my model to work

Read more in the User Guide.

Parameters:

n_estimators : *int, default=100*
The number of trees in the forest.



value of `n_estimators` changed from 10 to

Toggle Menu

Prev Up Next

scikit-learn 1.0.2
Other versions

Please cite us if you use the software.

sklearn.ensemble.RandomForestClassifier
Examples using sklearn.ensemble.RandomForestC

sklearn.ensemble.RandomForestClassifier

class sklearn.ensemble.RandomForestClassifier(*n_estimators=100*, ***, *criterion='gini'*, *max_depth=None*, *min_samples_split=2*, *min_samples_leaf=1*, *min_weight_fraction_leaf=0.0*, *max_features='auto'*, *max_leaf_nodes=None*, *min_impurity_decrease=0.0*, *bootstrap=True*, *oob_score=False*, *n_jobs=None*, *random_state=None*, *verbose=0*, *warm_start=False*, *class_weight=None*, *ccp_alpha=0.0*, *max_samples=None*) [source]

A random forest classifier.

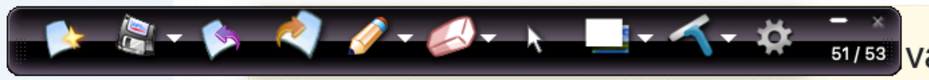
A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the `max_samples` parameter if `bootstrap=True` (default), otherwise the whole dataset is used to build each tree.

Read more in the User Guide.

Parameters:

n_estimators : *int, default=100*

The number of trees in the forest.



value of `n_estimators` changed from 10 to

sklearn.ensemble.RandomForestClassifier

class sklearn.ensemble.RandomForestClassifier(*n_estimators=100*, ***, *criterion='gini'*, *max_depth=None*, *min_samples_split=2*, *min_samples_leaf=1*, *min_weight_fraction_leaf=0.0*, *max_features='auto'*, *max_leaf_nodes=None*, *min_impurity_decrease=0.0*, *bootstrap=True*, *oob_score=False*, *n_jobs=None*, *random_state=None*, *verbose=0*, *warm_start=False*, *class_weight=None*, *ccp_alpha=0.0*, *max_samples=None*) [source]

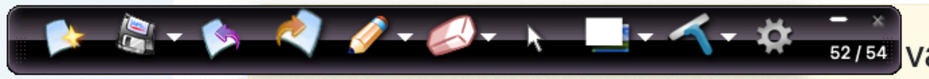
A random forest classifier.

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the `max_samples` parameter if `bootstrap=True` (default), otherwise the whole dataset is used to build each tree.

Read more in the User Guide.

Parameters:

n_estimators : *int, default=100*
The number of trees in the forest.



value of `n_estimators` changed from 10 to

sklearn.ensemble.RandomForestClassifier

class sklearn.ensemble.RandomForestClassifier(*n_estimators=100*, ***, *criterion='gini'*, *max_depth=None*, *min_samples_split=2*, *min_samples_leaf=1*, *min_weight_fraction_leaf=0.0*, *max_features='auto'*, *max_leaf_nodes=None*, *min_impurity_decrease=0.0*, *bootstrap=True*, *oob_score=False*, *n_jobs=None*, *random_state=None*, *verbose=0*, *warm_start=False*, *class_weight=None*, *ccp_alpha=0.0*, *max_samples=None*) [source]

A random forest classifier.

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the `max_samples` parameter if `bootstrap=True` (default), otherwise the whole dataset is used to build each tree.

Read more in the User Guide.

Parameters:

n_estimators : *int, default=100*
The number of trees in the forest.

base learners underfit

value of `n_estimators` changed from 10 to

sklearn.ensemble.RandomForestClassifier

class sklearn.ensemble.RandomForestClassifier(*n_estimators=100*, ***, *criterion='gini'*, *max_depth=None*, *min_samples_split=2*, *min_samples_leaf=1*, *min_weight_fraction_leaf=0.0*, *max_features='auto'*, *max_leaf_nodes=None*, *min_impurity_decrease=0.0*, *bootstrap=True*, *oob_score=False*, *n_jobs=None*, *random_state=None*, *verbose=0*, *warm_start=False*, *class_weight=None*, *ccp_alpha=0.0*, *max_samples=None*) [source]

Shallow base-learners

A random forest classifier.

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the `max_samples` parameter if `bootstrap=True` (default), otherwise the whole dataset is used to build each tree.

Read more in the User Guide.

Parameters:

n_estimators : *int, default=100*
The number of trees in the forest.

value of `n_estimators` changed from 10 to

sklearn.ensemble.RandomForestClassifier

class sklearn.ensemble.RandomForestClassifier(*n_estimators=100*, ***, *criterion='gini'*, *max_depth=None*, *min_samples_split=2*, *min_samples_leaf=1*, *min_weight_fraction_leaf=0.0*, *max_features='auto'*, *max_leaf_nodes=None*, *min_impurity_decrease=0.0*, *bootstrap=True*, *oob_score=False*, *n_jobs=None*, *random_state=None*, *verbose=0*, *warm_start=False*, *class_weight=None*, *ccp_alpha=0.0*, *max_samples=None*) [source]

A random forest classifier.

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the `max_samples` parameter if `bootstrap=True` (default), otherwise the whole dataset is used to build each tree.

Read more in the User Guide.

Parameters:

n_estimators : *int, default=100*
The number of trees in the forest.

value of `n_estimators` changed from 10 to

scikit learn
Prev Up Next
scikit-learn 1.0.2
Other versions
Please cite us if you use the software.
sklearn.ensemble.RandomForestClassifier
Examples using sklearn.ensemble.RandomForestC

min_weight_fraction_leaf : *float, default=0.0*

The minimum weighted fraction of the sum total of weights (of all the input samples) required to be at a leaf node. Samples have equal weight when sample_weight is not provided.

max_features : *{"auto", "sqrt", "log2"}, int or float, default="auto"*

The number of features to consider when looking for the best split:

- If int, then consider `max_features` features at each split.
- If float, then `max_features` is a fraction and `round(max_features * n_features)` features are considered at each split.
- If "auto", then `max_features=sqrt(n_features)`.
- If "sqrt", then `max_features=sqrt(n_features)` (same as "auto").
- If "log2", then `max_features=log2(n_features)`.
- If None, then `max_features=n_features`.

Note: the search for a split does not stop until at least one valid partition of the node samples is found, even if it requires to effectively inspect more than `max_features` features.

max_leaf_nodes : *int, default=None*

Grow trees with `max_leaf_nodes` in best-first fashion. Best nodes are defined as relative reduction in impurity. If None then unlimited number of leaf nodes.

sklearn.ensemble.RandomForestClassifier

class sklearn.ensemble.RandomForestClassifier(*n_estimators=100*, ***, *criterion='gini'*, *max_depth=None*, *min_samples_split=2*, *min_samples_leaf=1*, *min_weight_fraction_leaf=0.0*, *max_features='auto'*, *max_leaf_nodes=None*, *min_impurity_decrease=0.0*, *bootstrap=True*, *oob_score=False*, *n_jobs=None*, *random_state=None*, *verbose=0*, *warm_start=False*, *class_weight=None*, *ccp_alpha=0.0*, *max_samples=None*) [source]

A random forest classifier.

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the `max_samples` parameter if `bootstrap=True` (default), otherwise the whole dataset is used to build each tree.

Read more in the User Guide.

Parameters:

n_estimators : *int, default=100*
The number of trees in the forest.

value of `n_estimators` changed from 10 to

sklearn.ensemble.RandomForestClassifier

class sklearn.ensemble.RandomForestClassifier(*n_estimators=100*, ***, *criterion='gini'*, *max_depth=None*, *min_samples_split=2*, *min_samples_leaf=1*, *min_weight_fraction_leaf=0.0*, *max_features='auto'*, *max_leaf_nodes=None*, *min_impurity_decrease=0.0*, *bootstrap=True*, *oob_score=False*, *n_jobs=None*, *random_state=None*, *verbose=0*, *warm_start=False*, *class_weight=None*, *ccp_alpha=0.0*, *max_samples=None*) [source]

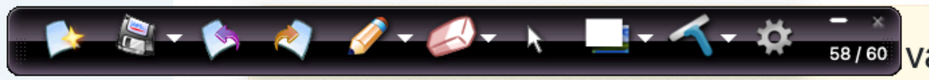
A random forest classifier.

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the `max_samples` parameter if `bootstrap=True` (default), otherwise the whole dataset is used to build each tree.

Read more in the User Guide.

Parameters:

n_estimators : *int, default=100*
The number of trees in the forest.



value of `n_estimators` changed from 10 to

oob_score : *bool, default=False*

Whether to use out-of-bag samples to estimate the generalization score. Only available if bootstrap=True.

n_jobs : *int, default=None*

The number of jobs to run in parallel. `fit`, `predict`, `decision_path` and `apply` are all parallelized over the trees. `None` means 1 unless in a `joblib.parallel_backend` context. `-1` means using all processors. See Glossary for more details.

random_state : *int, RandomState instance or None, default=None*

Controls both the randomness of the bootstrapping of the samples used when building trees (if `bootstrap=True`) and the sampling of the features to consider when looking for the best split at each node (if `max_features < n_features`). See Glossary for details.

verbose : *int, default=0*

Controls the verbosity when fitting and predicting.

warm_start : *bool, default=False*

When set to `True`, reuse the solution of the previous call to fit and add more estimators to the ensemble, otherwise, just fit a whole new forest. See the Glossary.

class_weight : *{"balanced", "balanced_subsample"}, dict or list of dicts, default=None*

form `{class_label: weight}`. If not given, all

scikit learn
Prev Up Next
scikit-learn 1.0.2
Other versions
Please cite us if you use the software.
sklearn.ensemble.RandomForestClassifier
Examples using sklearn.ensemble.RandomForestCl

For multi-output, the weights of each column of y will be multiplied.

Note that these weights will be multiplied with sample_weight (passed through the fit method) if sample_weight is specified.

ccp_alpha : ***non-negative float, default=0.0***

Complexity parameter used for Minimal Cost-Complexity Pruning. The subtree with the largest cost complexity that is smaller than `ccp_alpha` will be chosen. By default, no pruning is performed. See Minimal Cost-Complexity Pruning for details.

New in version 0.22.

100 0.1

max_samples : ***int or float, default=None***

If bootstrap is True, the number of samples to draw from X to train each base estimator.

m

- If None (default), then draw `X.shape[0]` samples.
- If int, then draw `max_samples` samples.
- If float, then draw `max_samples * X.shape[0]` samples. Thus, `max_samples` should be in the interval `(0.0, 1.0]`.

New in version 0.22.

Attributes: **base_estimator_** : ***DecisionTreeClassifier***

reate the collection of fitted sub-estimators.

1.10.8. Minimal Cost-Complexity Pruning

CCP

Minimal cost-complexity pruning is an algorithm used to prune a tree to avoid over-fitting, described in Chapter 3 of [BRE]. This algorithm is parameterized by $\alpha \geq 0$ known as the complexity parameter. The complexity parameter is used to define the cost-complexity measure, $R_\alpha(T)$ of a given tree T:

$$R_\alpha(T) = R(T) + \alpha|\widetilde{T}|$$

where $|\widetilde{T}|$ is the number of terminal nodes in T and $R(T)$ is traditionally defined as the total misclassification rate of the terminal nodes. Alternatively, scikit-learn uses the total sample weighted impurity of the terminal nodes for $R(T)$. As shown above, the impurity of a node depends on the criterion. Minimal cost-complexity pruning finds the subtree of T that minimizes $R_\alpha(T)$.

The cost complexity measure of a single node is $R_\alpha(t) = R(t) + \alpha$. The branch, T_t, is defined to be a tree where node t is its root. In general, the impurity of a node is greater than the sum of impurities of its terminal nodes, $R(T_t) < R(t)$. However, the cost complexity measure of a node, t, and its branch, T_t, can be equal depending on α. We define the effective α of a node to be the value where they are equal, $R_\alpha(T_t) = R_\alpha(t)$ or $\alpha_{eff}(t) = \frac{R(t)-R(T_t)}{|T|-1}$. A non-terminal node with the smallest value of α_{eff} is the weakest link and will be pruned. This process stops when the pruned tree's minimal α_{eff} is greater than the `ccp_alpha` parameter.

Examples:

- Post pruning decision trees with cost complexity pruning

not an optimization based algo :

$$\min(\text{loss} + \alpha \cdot \#\text{leaves in the Tree})$$

Change depth

DT

overfit vs underfit

λ in logistic-reg

error : log-loss or sq-loss

ccp

depth

$\min\left(\text{loss}_{\text{train}} + \alpha \cdot \#\text{leaf-nodes}\right)$

$\alpha \uparrow$: donot overfit

lower depth

regularization as it controls the tree from overfitting

sklearn.ensemble.RandomForestClassifier

class sklearn.ensemble.RandomForestClassifier(*n_estimators=100*, ***, *criterion='gini'*, *max_depth=None*, *min_samples_split=2*, *min_samples_leaf=1*, *min_weight_fraction_leaf=0.0*, *max_features='auto'*, *max_leaf_nodes=None*, *min_impurity_decrease=0.0*, *bootstrap=True*, *oob_score=False*, *n_jobs=None*, *random_state=None*, *verbose=0*, *warm_start=False*, *class_weight=None*, *ccp_alpha=0.0*, *max_samples=None*) [source]

base-learners shallow

A random forest classifier.

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the `max_samples` parameter if `bootstrap=True` (default), otherwise the whole dataset is used to build each tree.

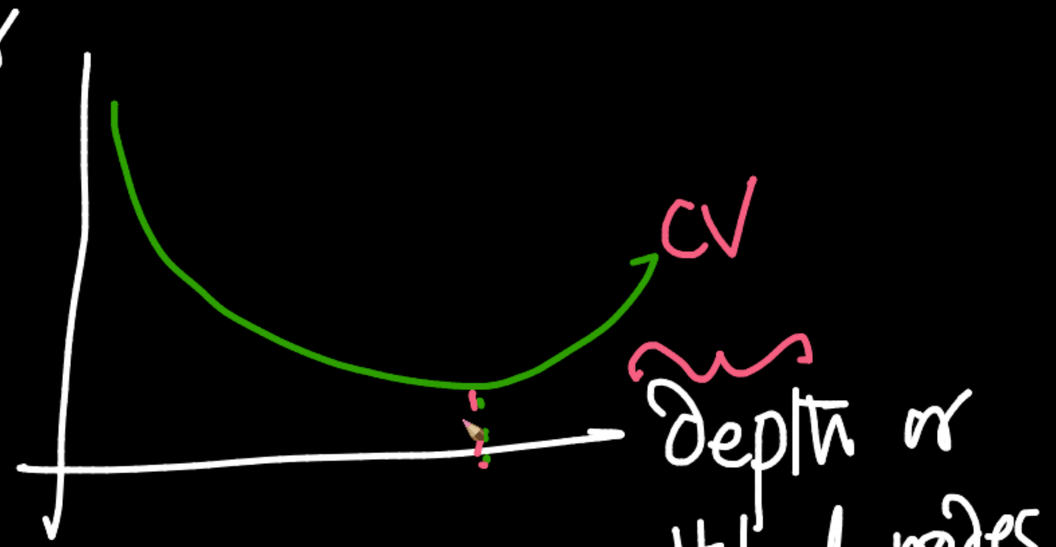
Read more in the User Guide.

Parameters:

n_estimators : *int, default=100*
The number of trees in the forest.

value of `n_estimators` changed from 10 to

①



② CCP

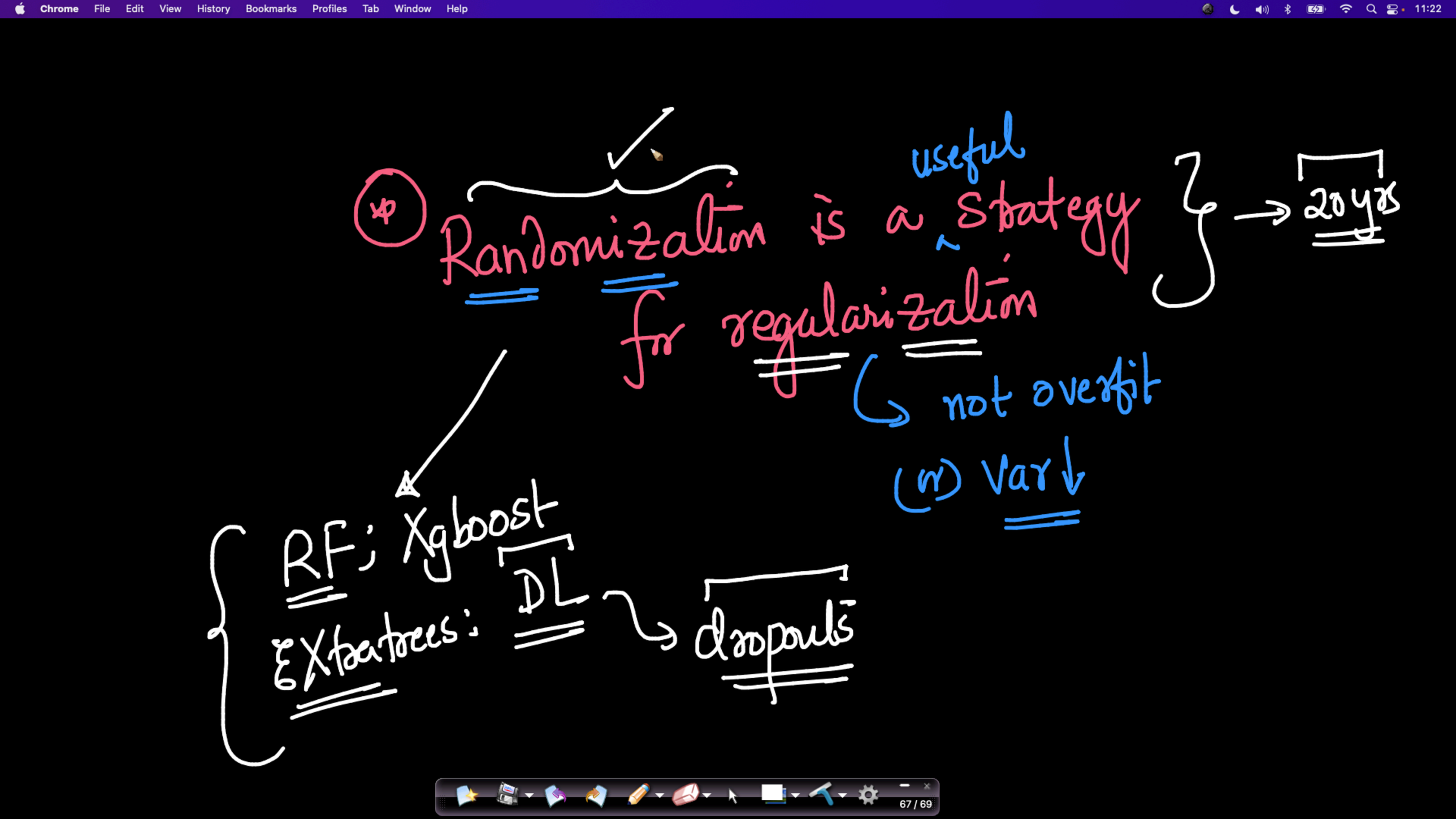
$$\min \left(\text{loss}_{\text{train}} + \alpha \cdot \#\text{leaf-nodes} \right)$$

Extra Trees or Extremely Randomized trees

RF: random: Col. sampling + row-sampling + agg

→ overfit → agg ✓ → var ↓ bias →

random-zati


Randomization is a useful Strategy for regularization
2 yrs
not overfit
(or) Var↓
RF; Xgboost
EXtratrees: DL
dropouts

Extra trees:

- Col. sampling
- Row. sampling
- Aggregation
- randomly picks the 'c' to split for numerical features

recap:

f_i	y_i
2.1	1
2.2	1
2.3	0
2.6	1
2.8	1
2.9	0

DT

$f_i < \tau$

$\tau_1 = 2.1 \rightarrow \Delta H : IG_2$

$\tau_2 = 2.2 \rightarrow IG_2$

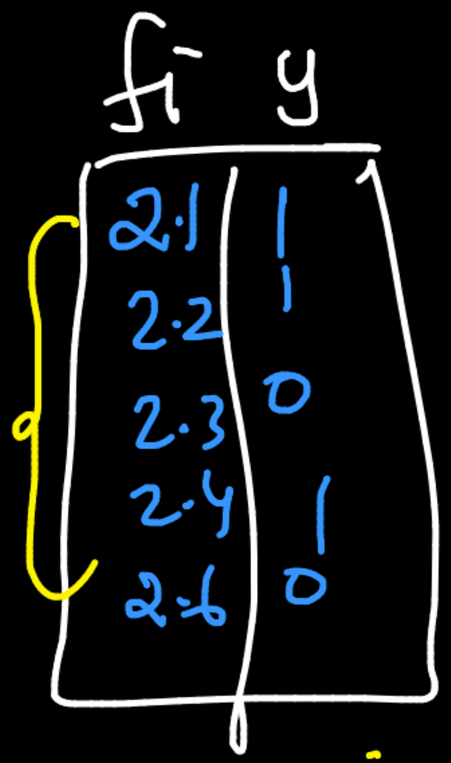
$\vdots$

$\tau_6 = 2.9 \rightarrow IG_1$

Chrome File Edit View History Bookmarks Profiles Tab Window Help 11:27

Extra Trees!

more randomization



$\tau_1 = 2.3 \rightarrow IG_{2.3}$

$\tau_2 = 2.6 \rightarrow IG_{2.6}$

τ_{10}

m is large

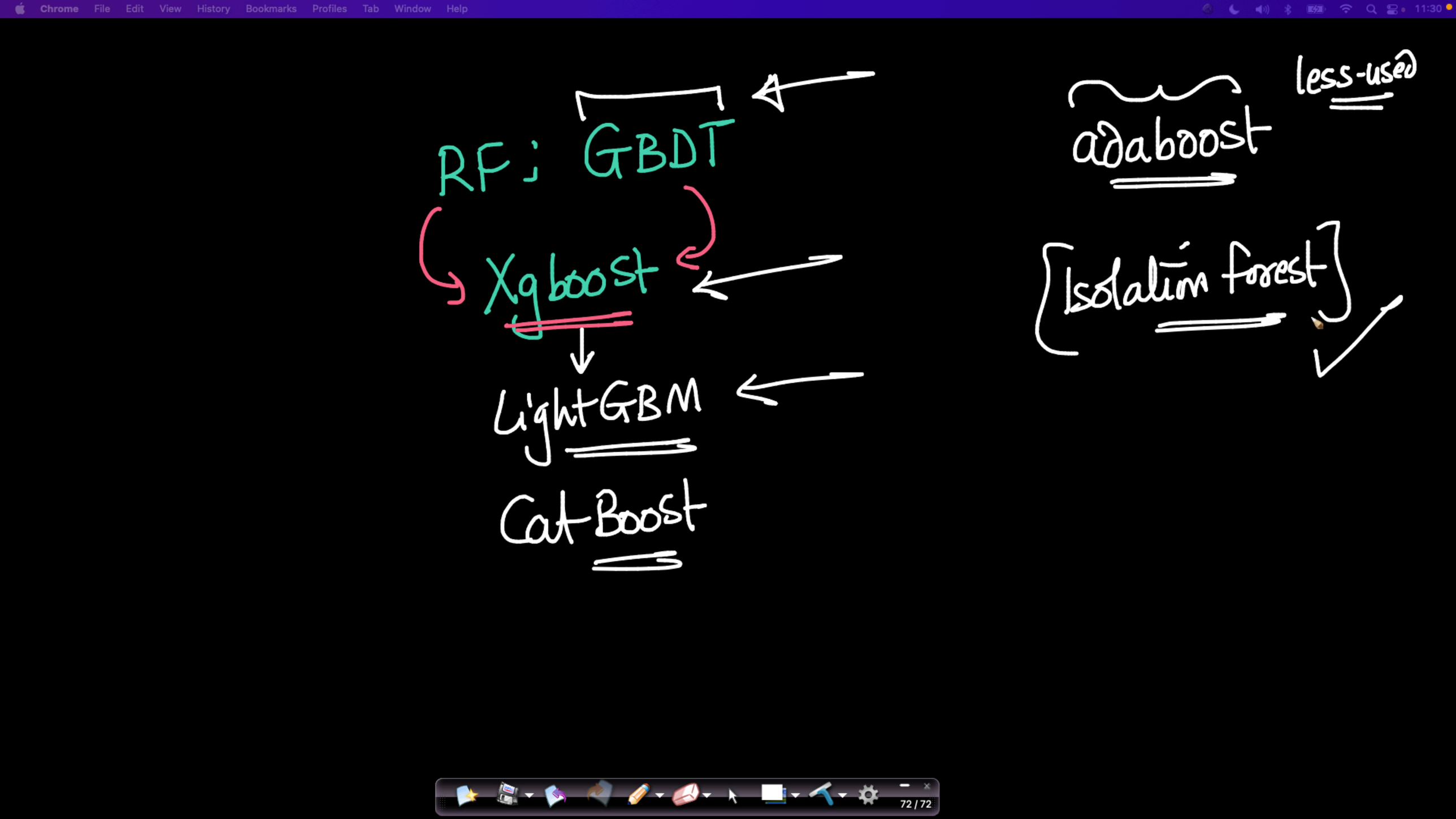
(2006) Extra trees:- one more randomization (τ)

↳ m is large (useful) → speed up

typical ~ more base-learners to perform well

RF; GBDT

not widely used


RF ; GBDT
Xgboost
LightGBM
CatBoost
adaboost
less-used
Isolation forest

RF:

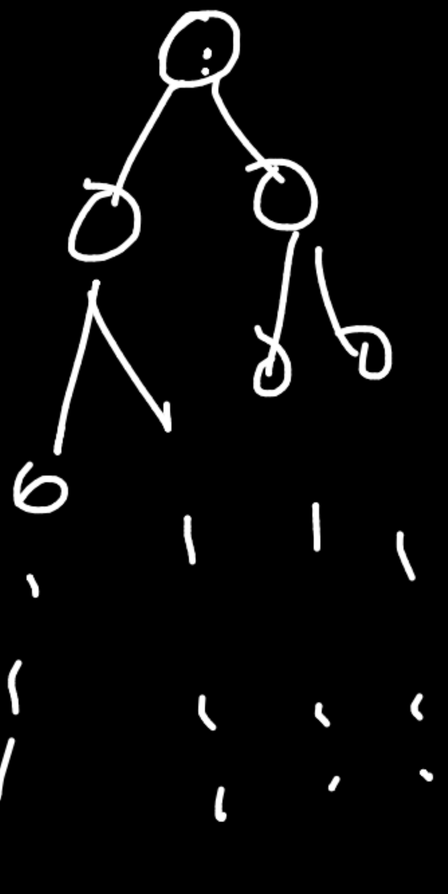
→ d'm is not too large

→ trivially parallizable: n is large...

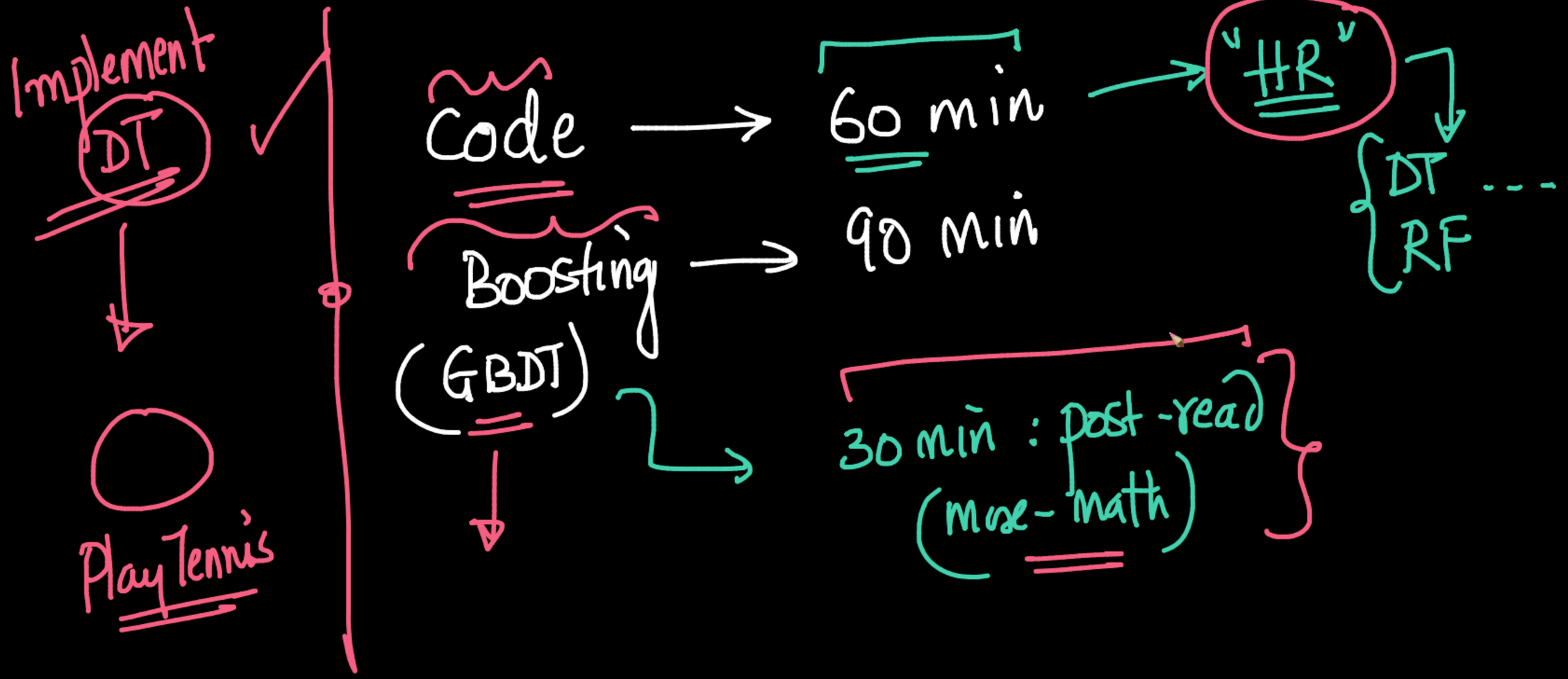
→ K; $\frac{m}{n}$; $\frac{d'}{d}$

d↑

1000



DT


Implement
DT
Play Tennis
Code
60 min
HR
DT
RF
...
Boosting
(GBDT)
90 min
30 min : post-read
(more-math)

STATS 361: Causal Inference

Stefan Wager
Stanford University

Spring 2020