Lecture link : https://www.scaler.com/meetings/i/dsml-advanced-ensemble-boosting-2/archive

Content

Recap

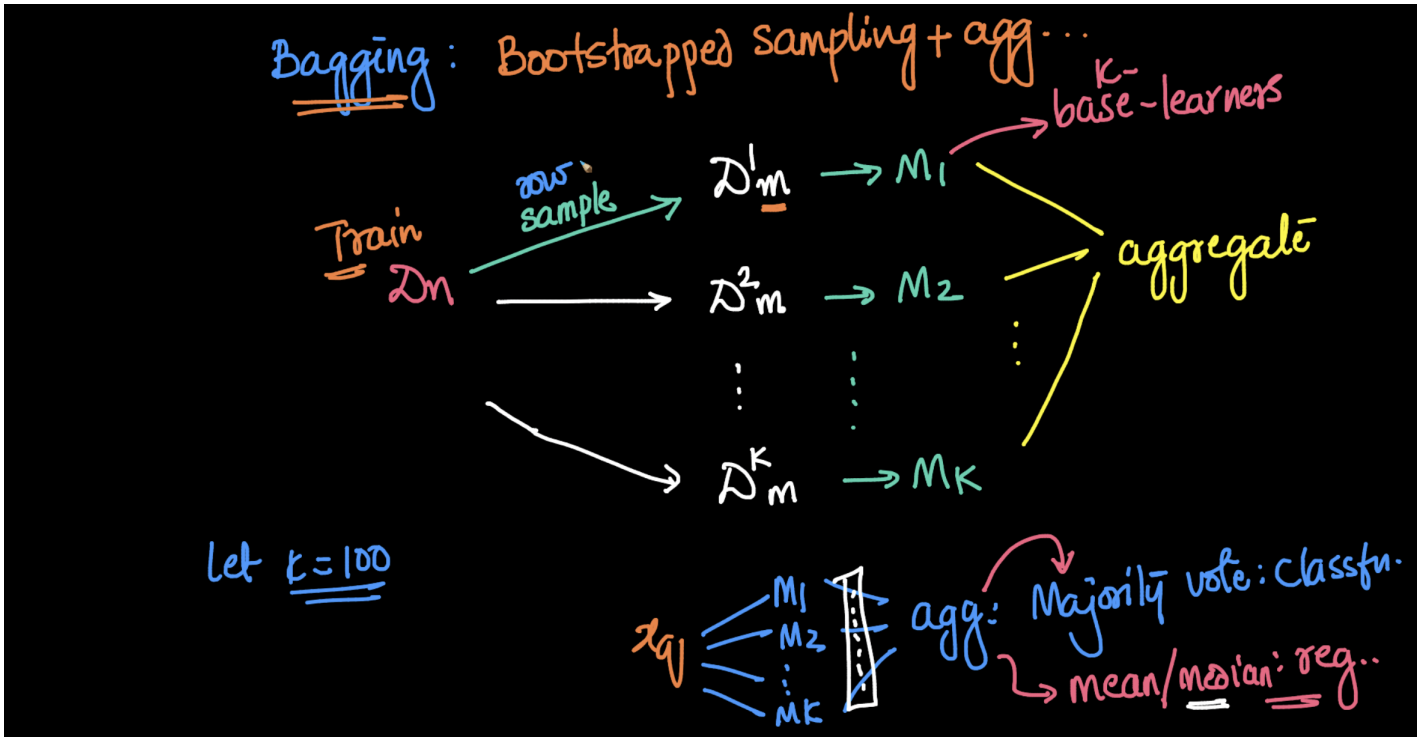
In previous lecture we saw what bagging is, lets take a quick recap of what it is and how it works

- Bagging is nothing but **Bootstrapped sampling + Aggregation**

How it works ?

Let us assume a train data set D with n data points i.e D_n

- Now, we **sample m data points with replacement** to get D'_m
- We do the sampling again for m points to get D'_2, **Repeat the same for k times** and we get D'_k
- Now, we **train k different models**($M_1, M_2, \ldots . M_k$) basing on the k datasets obtained , there models are called **Base Learners**.
- After training we **cross validate each model with remaining $n - m$ data points**
- Now, we do **Aggregation**
    - We use majority vote for Classification
    - We use Mean/Median for Regression



Let's now see what a Random forest is

Random Forest

- As you can understand from the name a Random forest is nothiing but a group of trees.
- It's basically a bagging based system with Decision tree
  - Base learners of Random Forest are Decision tree
- So, as a whole, Random forest is Applying Randomisation on the data and ensemble of Decision trees and later bagging these.

Let's see Random forest is built

- Let us assume we have a train data set (D_n) with n rows and and d columns
- And now we apply **row samplinig and column sampling** on the data to get new data set (D_m^1) with m rows and d' columns
  - where $m < n$ and $d' < d$

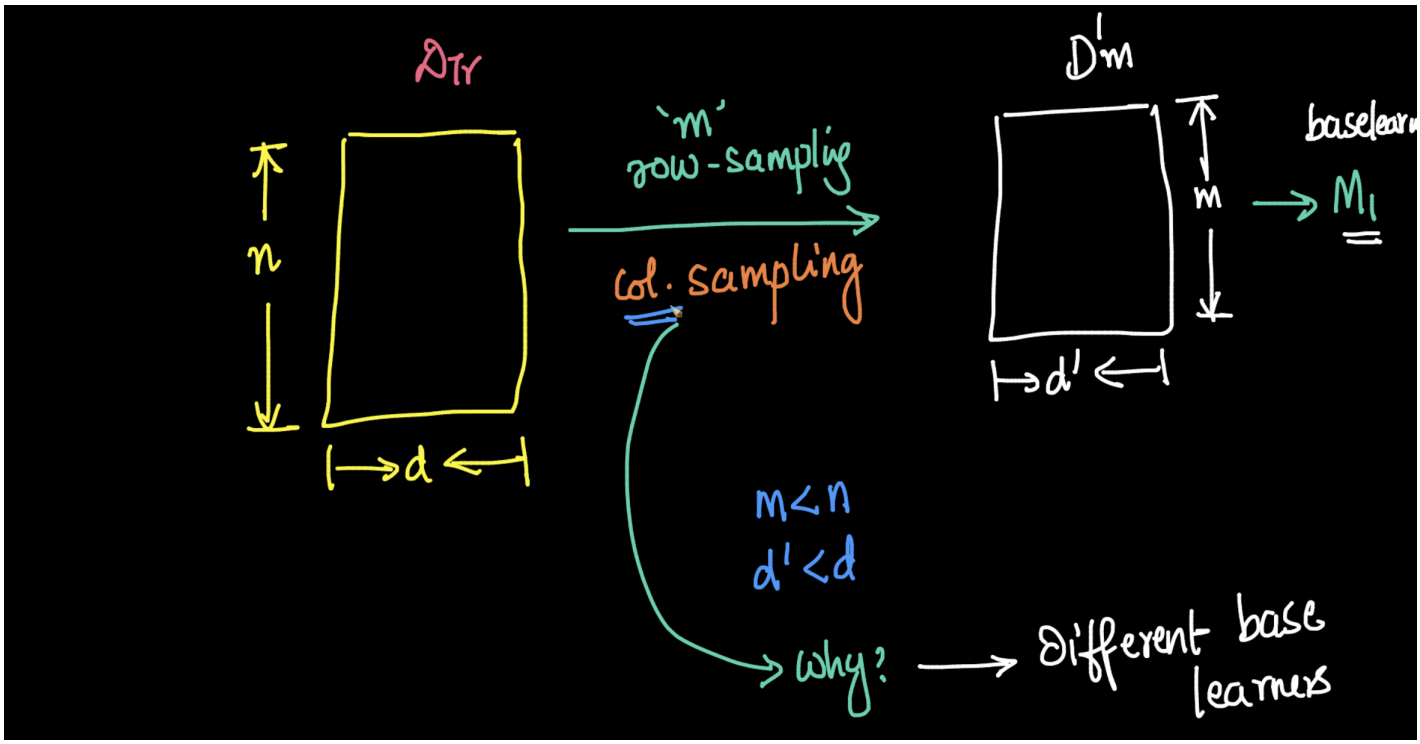
But why do we apply columnn sampling?

- Because we want all our **Base learners** to be different, choosing different set of features may help them to be different, for achieving this we do column sampling as well.

- Adding column sampling makes the models to be little more different

We do the same again and again to get k different data sets

- So we get D_m^1,D_m^2....D_m^k now the models M_1,M_2,...M_k will be different
- Now we aggregate the models



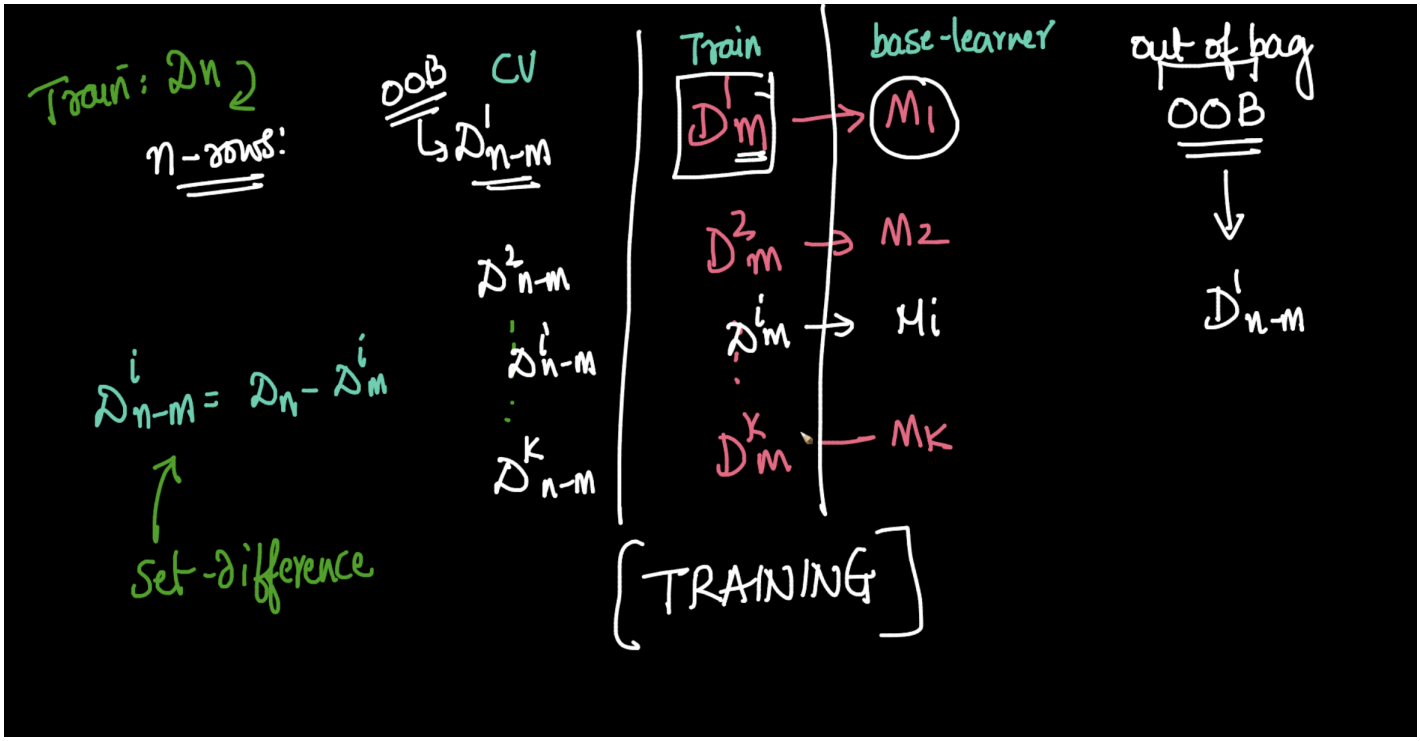
Do you see a problem here?

- We have k different train data sets (D_m^1,D_m^2....D_m^k) to train k different models (M_1,M_2,... M_k) respectively

Where is the cross validation data?

The remaining $n - m$ rows can be used as cross validation data

- for $D_{n-m}^i = D_n - D_m^i$,is Set Difference
  - This is reffered to as Out Of Bag(OOB)
- So, we train the models on (D_m^1,D_m^2....D_m^k)these data sets and cross validate them on (D_{n-m}^1,D_{n-m}^2,....D_{n-m}^k) this data which is OOB sample
- If some data points are not present in any of the train data sets, they will be in every cross validation data.

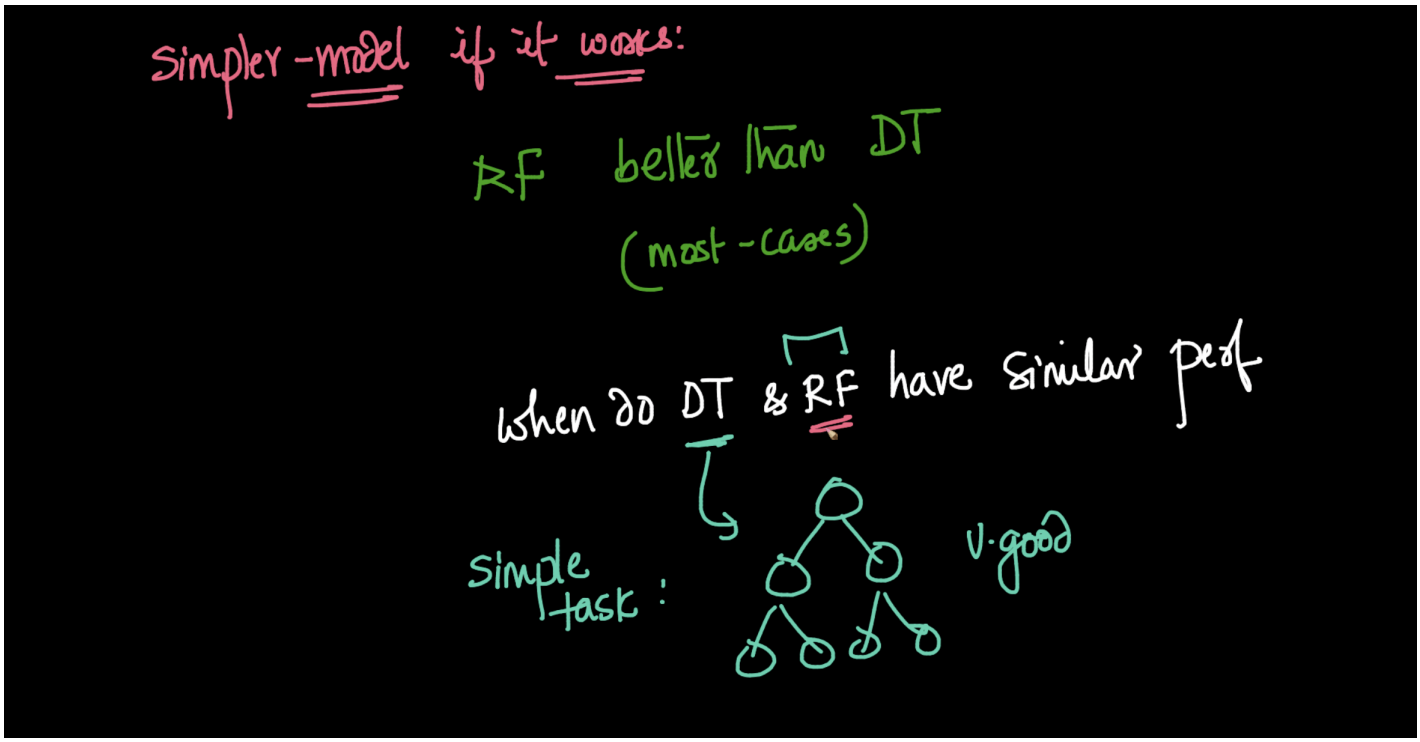


Will Random Forest always out perform Decision tree?

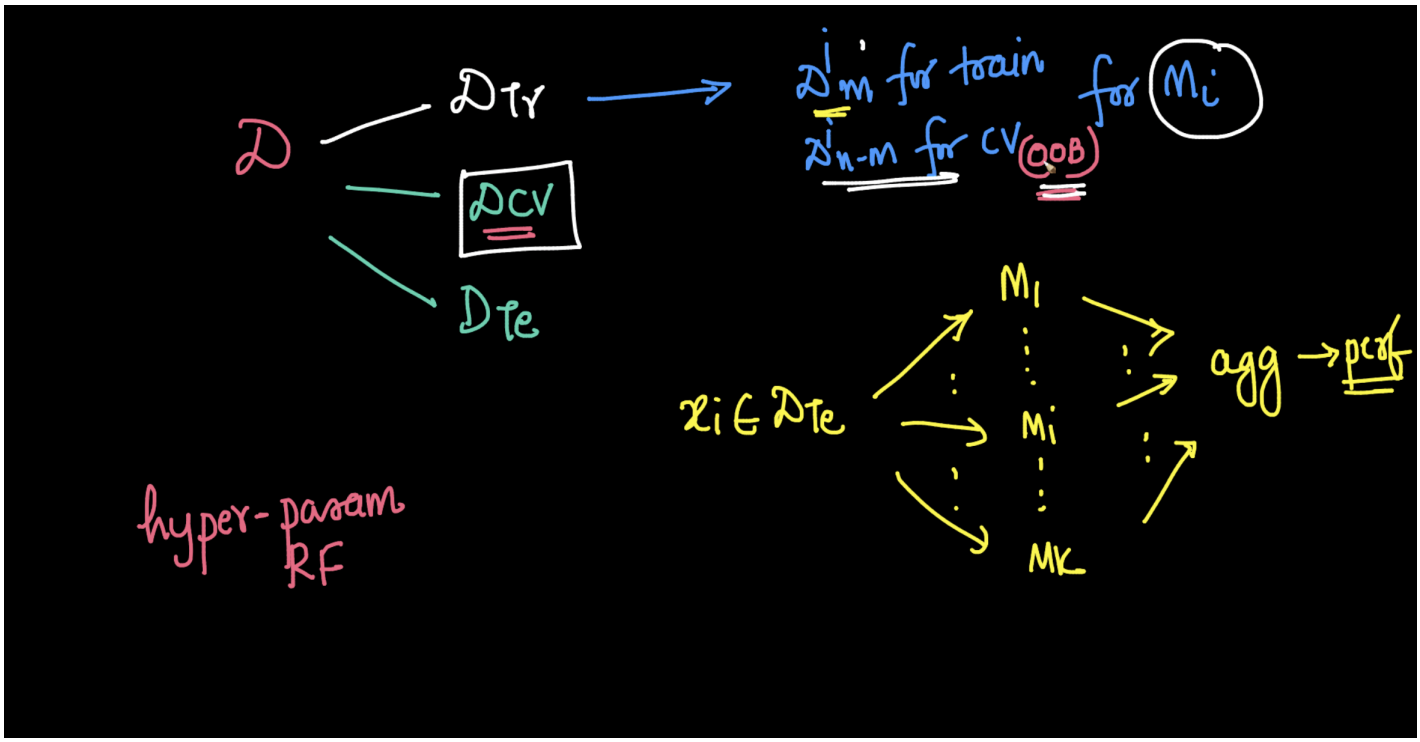
- Yes, in most cases of the cases they should ut perform

When will the performance of both Random Forest and Decision Tree be similar?

- This may happen when the data is very simple, that a simple Decisison tree by using simple rules can get a very good performance.

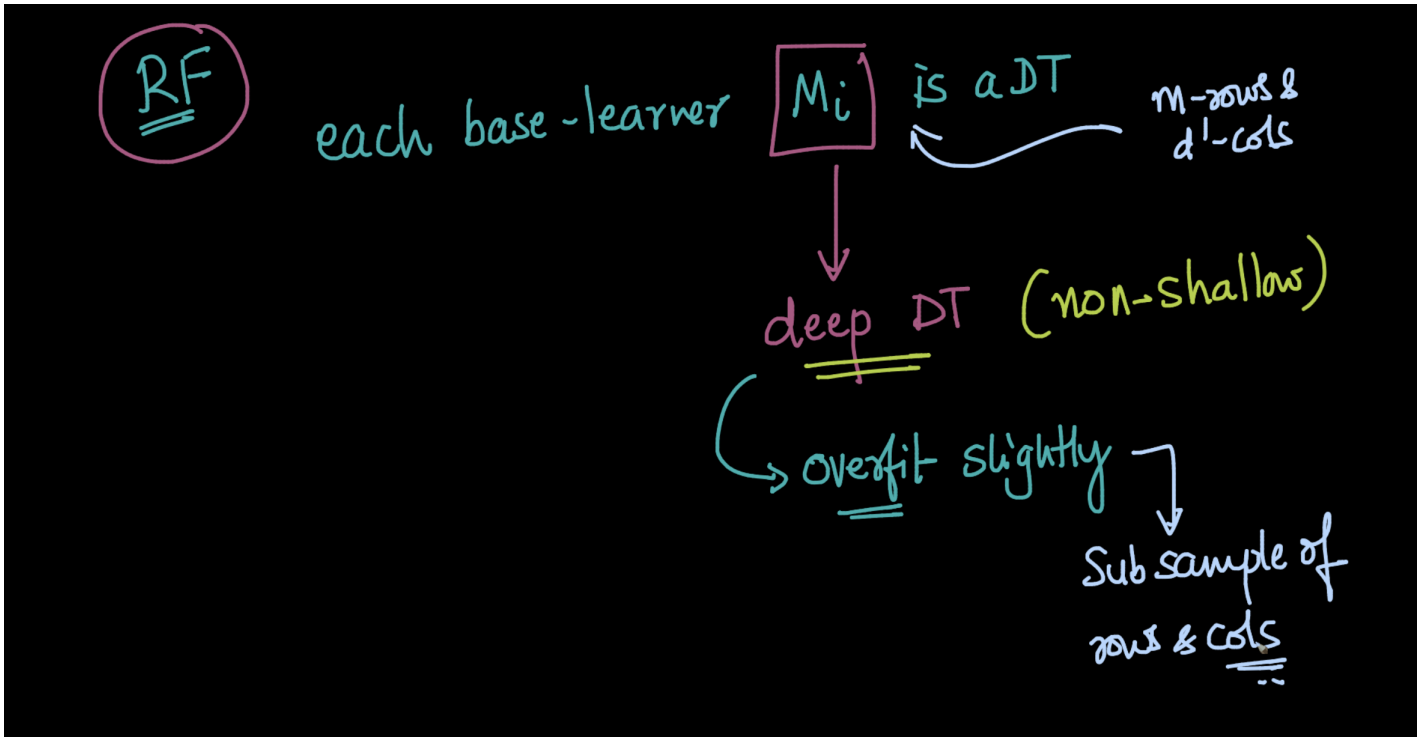


- In random forest The Base learners don't need seperate cross validation data set because the remaining data after sampling(OOB) are used for cross validation
    - i.e from D_n, D_m^i is sampled and D_{m-n}^i is used for cross validation
- But the random forest as a whole has the cross validation data and test data to tune the Hyper Parameters of random Forest

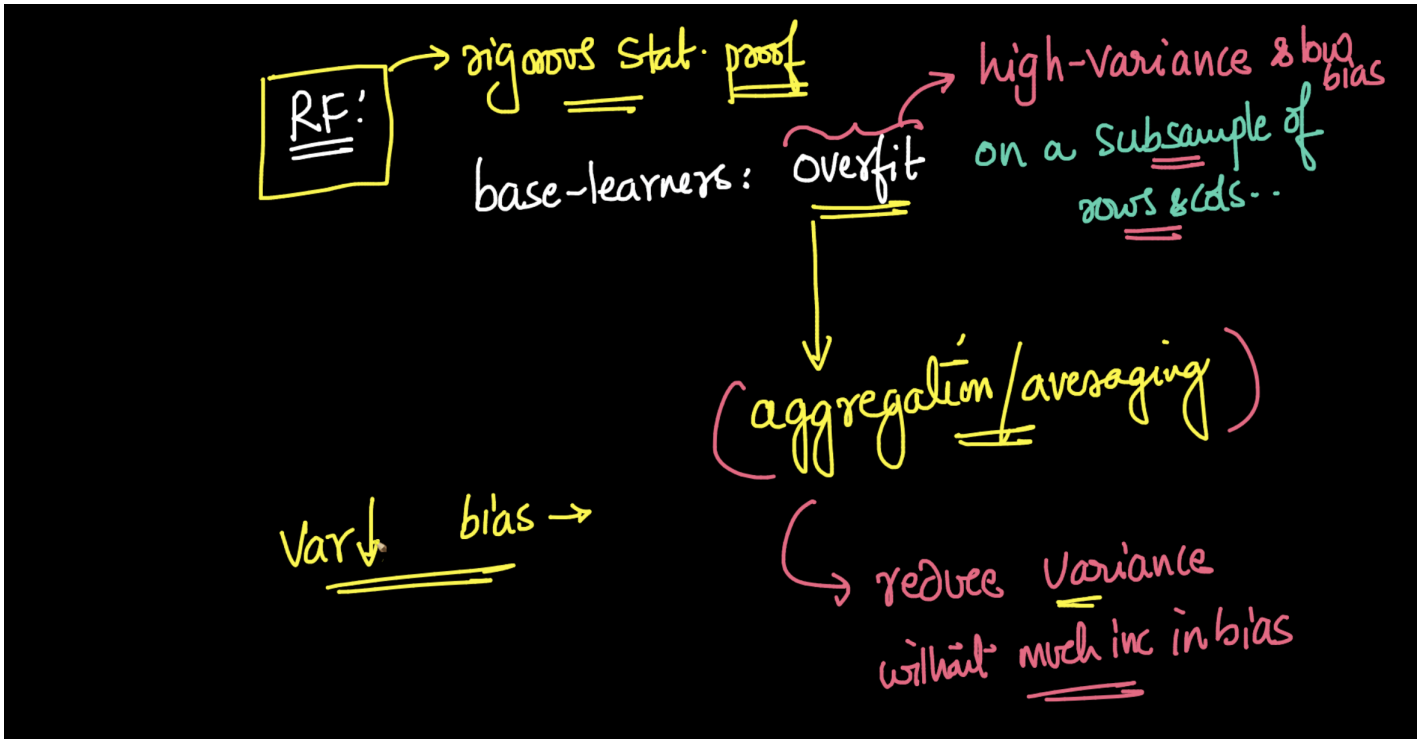


What happens if k increases?

- The Decision tree base learners in random forest are Deep Decision Trees (non-shallow).
- As they are deep, the models overfit slightly on a sub sample of data, as the base learners are made of m rows and d' columns only.
- Now, we perfrom aggregation\averaging on these slightly overfit models, on a sub-sample of data, which have high variance adn low bias



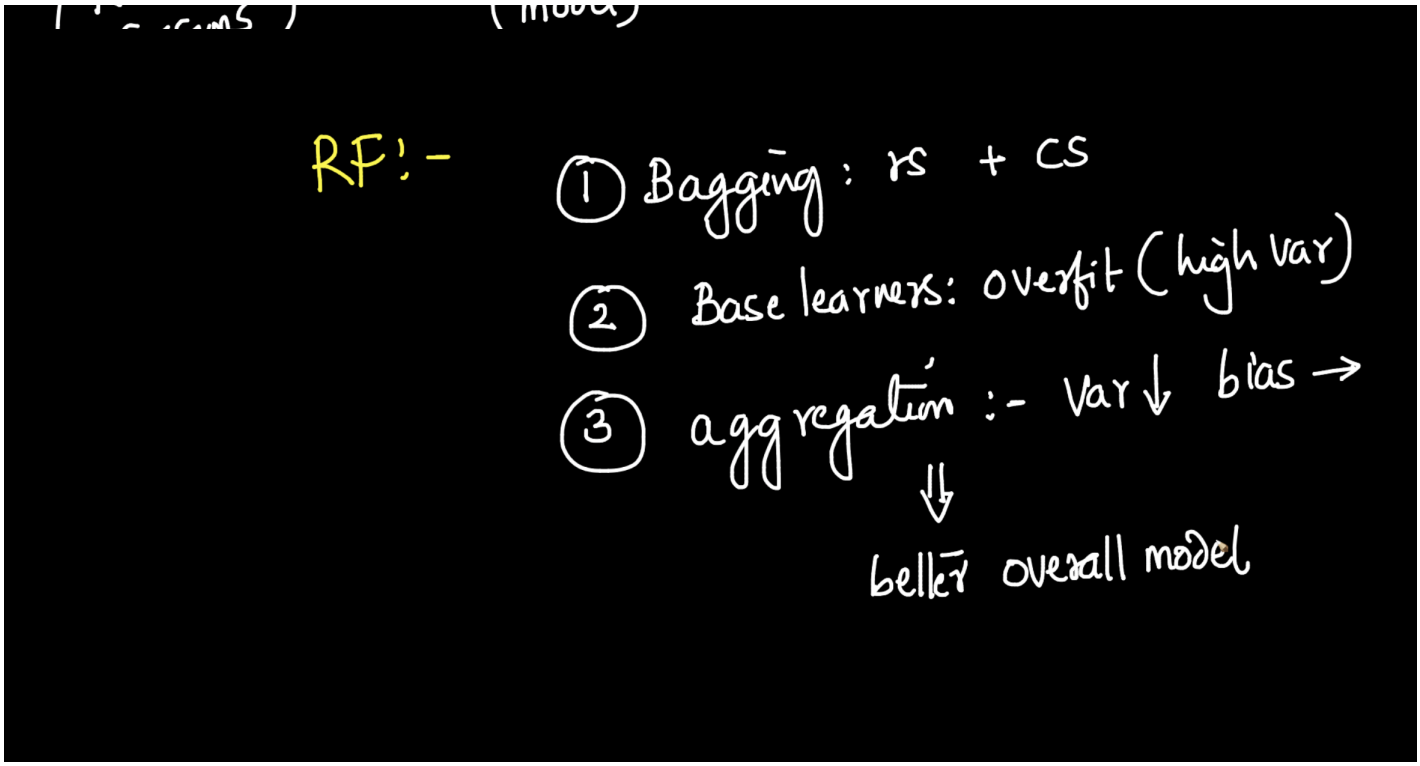
- But it is statistically proved that aggregating reduces the variance without much increase in the bias.
- In statistical Machine learning the error of a model can be represented as



- **Error = Bias^2 + Varinace + Irreducable error.**
- Here we saw due to aggregation, the variance decreases without trading-off to bias, due to which the overall error of the model decreases.
    - Therefore the error of the random forest reduces.

Now, as we saw each step of random forest

- Let's see the core working of a random forest
    1. It uses Bagging, which is row sampling, and column sampling
    2. It takes Base learners which are slightly overfit i.e having high variance and low bias
    3. It reduces this variance by aggregation without much increase in its bias.



Can we use Non Decision Tree models in Random forest?

- Yes we can you use slightly overfit models i.e having high variance models.
    - For example: We can use KNN with small k
    - We can use Logistic Regression with higher order features + low value of λ

Hyper-Parameter tuning of Random Forest

Now lets see various hyper parameters of Random Forest

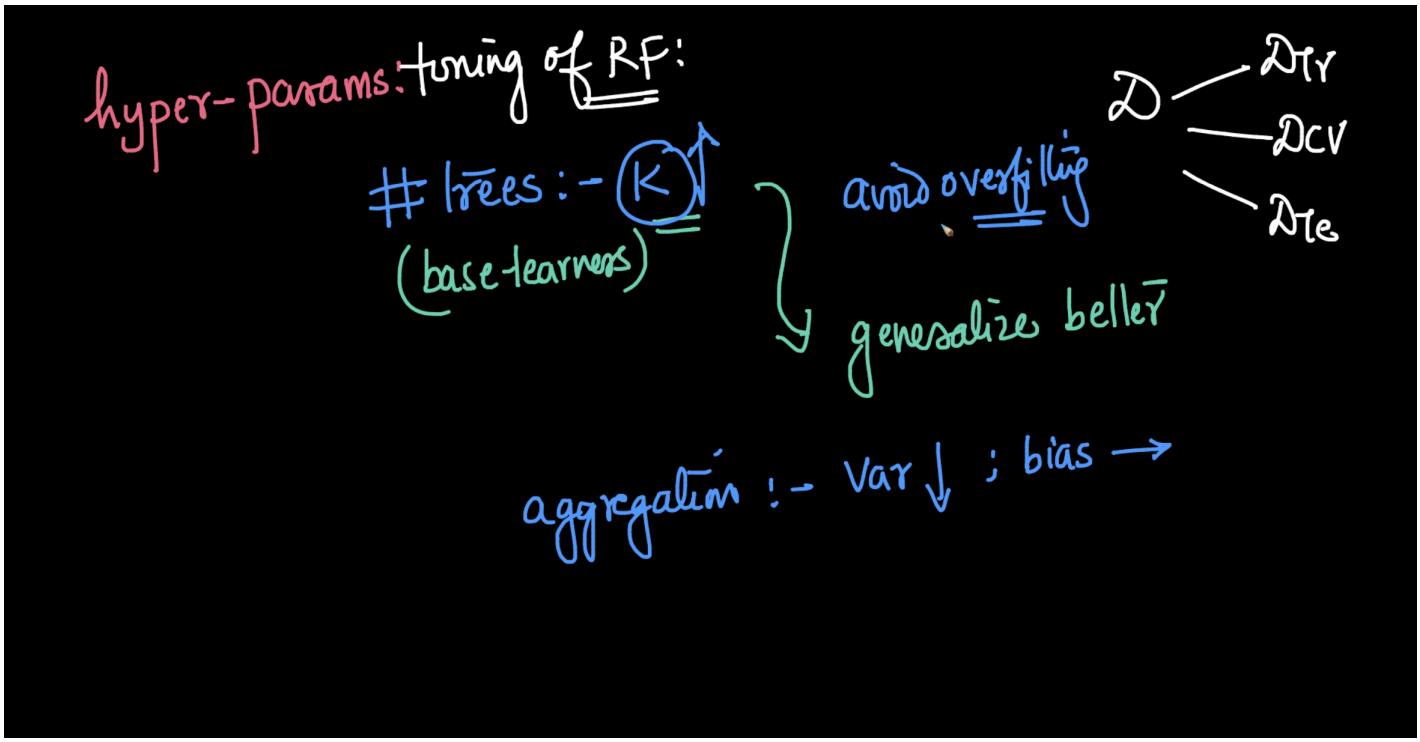
- Note that we consider the whole dataset D not only the D_{train} which is split into train data set for base learners

What are the hyper parameters for random forest?

1. Number of Trees(k)

What happens if number of base learners (k) increases?

- As k increases, we Aggregate more number of base learners
- We already saw that, Aggregation decreases the varaince without any significant increase in the bias.
- Hence, we can say that we avoid overfitting.



What if my base learners underfit instead being overfit?

- If the base learners underfit, the varaince is already low
- Hence the aggregation which should reduce the variance doesn't work as expected.

2. Row sample size (m)

- In some libraries $\frac{m}{n}$ is also considered where this value becomes maximum i.e 1 ,when $m = n$

What happens when $m = n$? ($d = d'$)

- All Base learners overfit on the whole data and they be similar.
- So, as the $\frac{m}{n}$ increases the over-fitting of the model increases

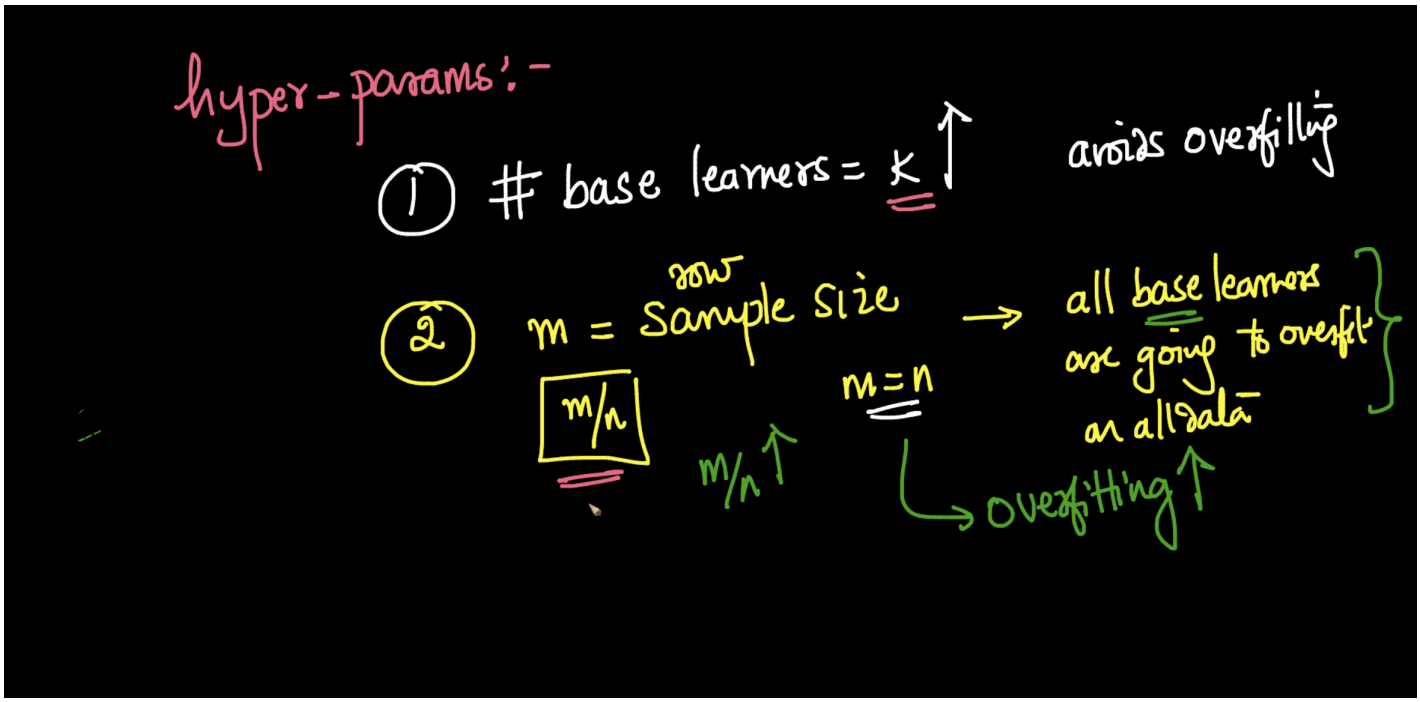
3. Number of columns Sampled (d')

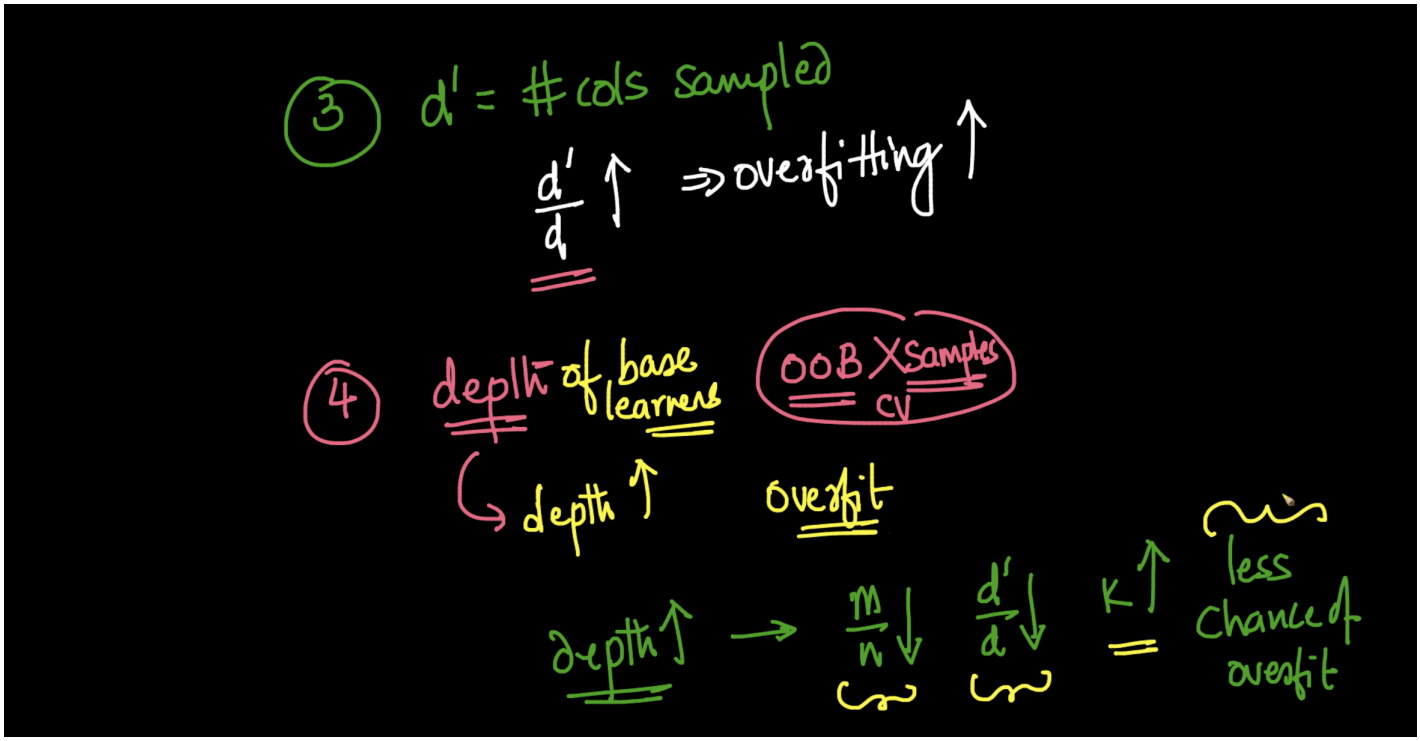
- Consider $m = n$

- As $\frac{d'}{d}$ increases, then all the features are being used by each of the tree.
- This implies the overfitting increases i.e the variance increases

4. Depth of Base learners

- This is optional, some libraries let us use this when we dont want to use the Out Of Bag (OOB) data fro cross validation
- As depth increases, the overfitting increases
    - When the ratios $\frac{m}{n}$ and $\frac{d'}{d}$ are very less, that is when the base leraners are seeing very few columns and rows , and when the number of trees increases (k) the overfitting chances are low.

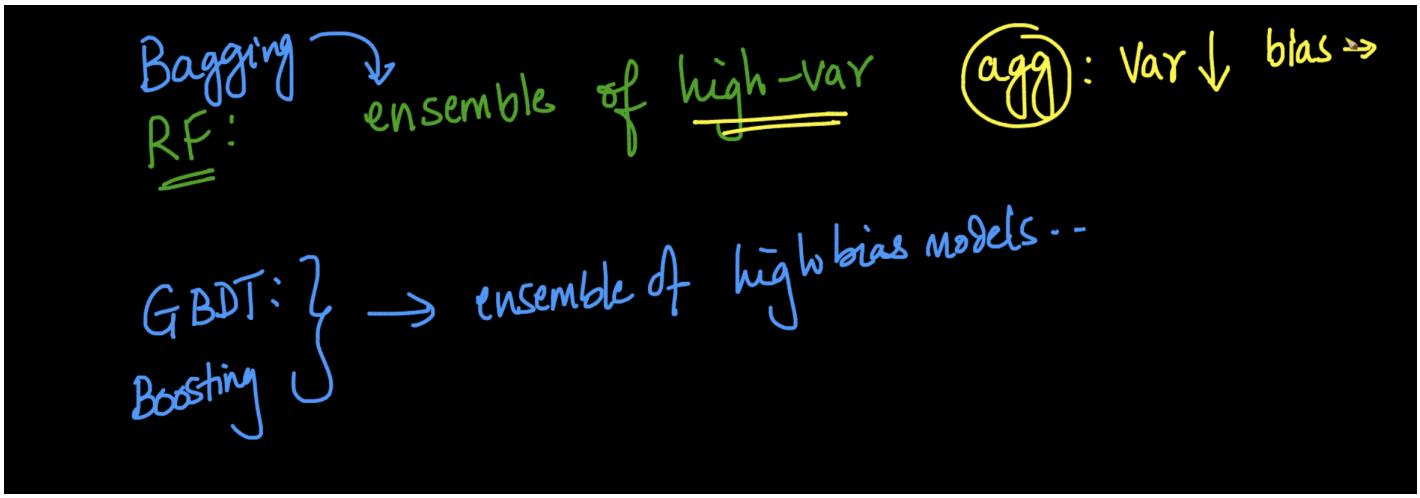




Questions

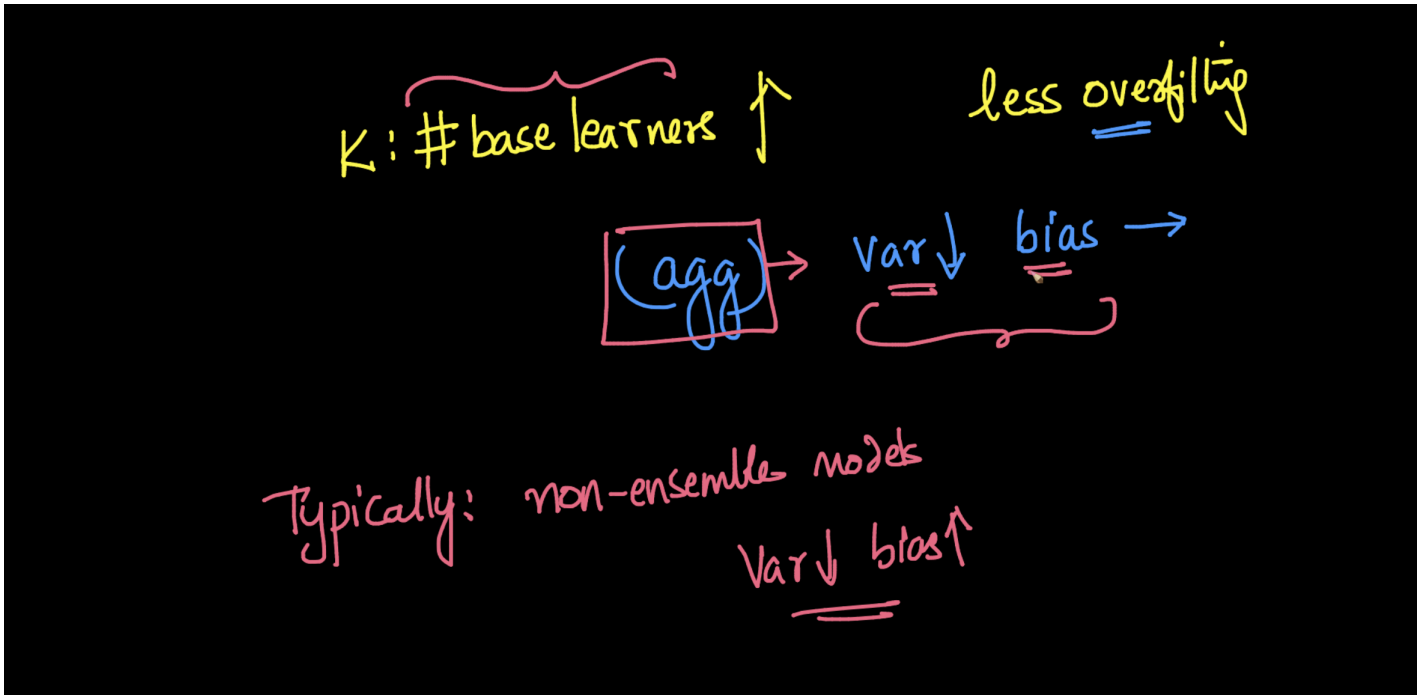
What if the models have high bias in an ensemble?

- We know Random forest is an ensemble of high varaince models, that is slightly overgfitting models
- We see the Gradient Boosted Decision Tree which applies boosting, which deals with high bias or underfitting models



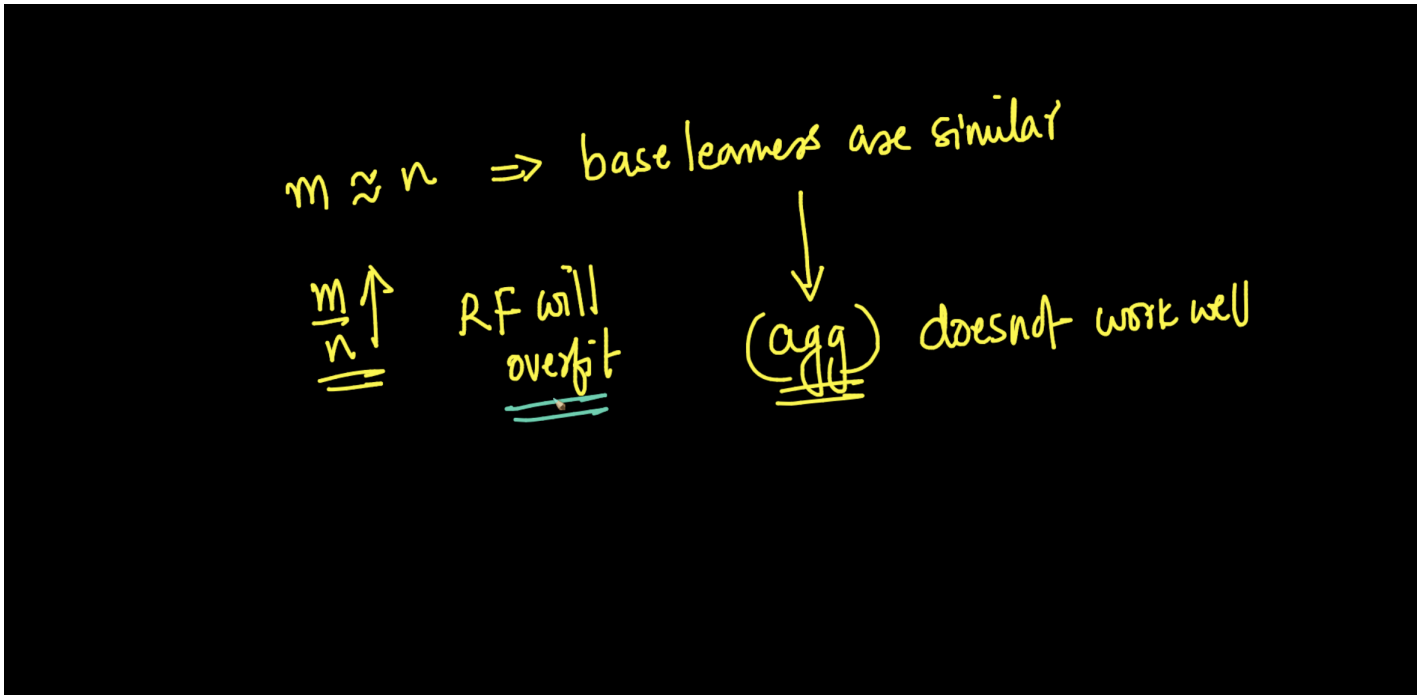
What if ensemble a deep decision tree and a simple linear classifier?

- This doesn't work as expected beacuse the random forest expects highy variance models, as it does aggregation at the end which reduces the variancce, but do not effect the bias.Due to which if a model of high bias is there, the high bias remains the same which we should avoid.



What happens when m is very close to n ?

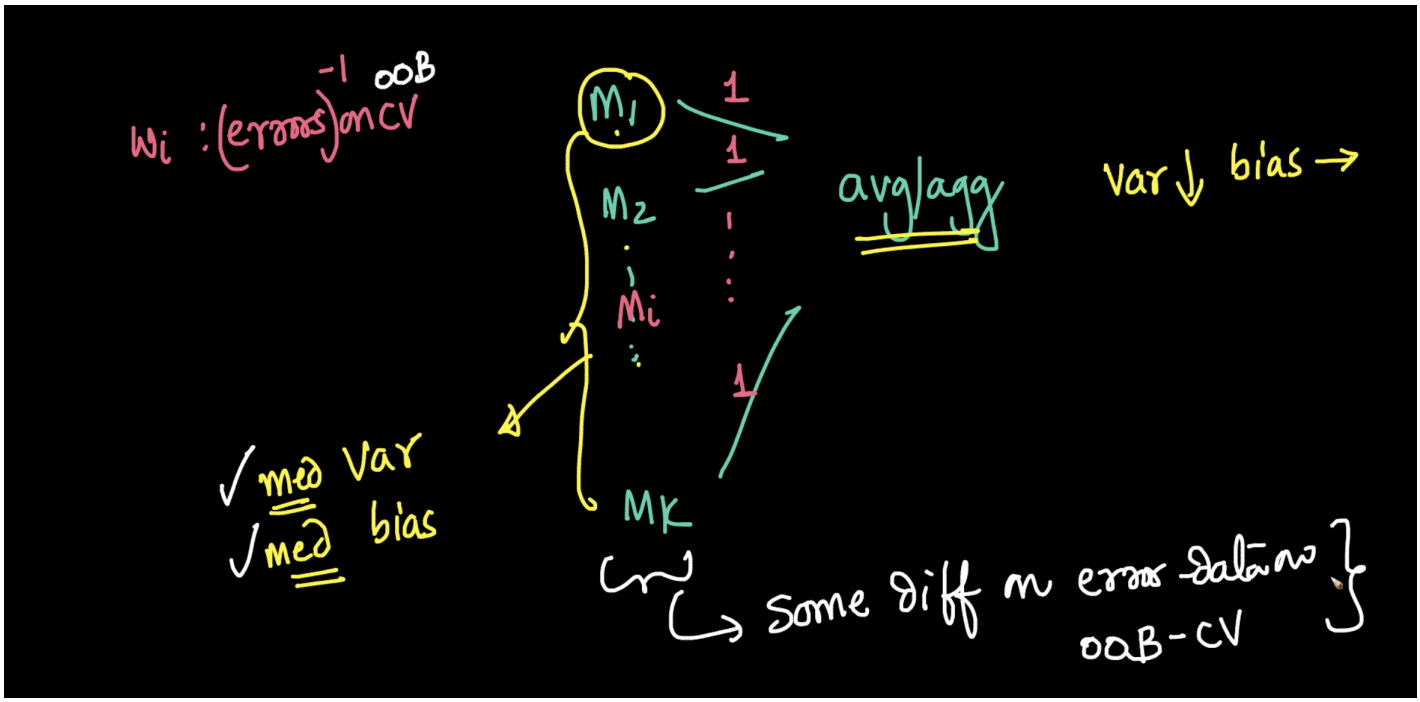
- When m is close to n the base learners overfit on the whole train data set. Due to which Aggregation doesn't work in reducing the variance
- So, when $\frac{m}{n}$ increases, Random forest overfits.
- But whwn $\frac{m}{n}$ decreases the base leraners overfit, due to which we require more number of trees for to reduce the variance.



Double-click (or enter) to edit

So here, Should we take weighted average of the base models in aggregation ?

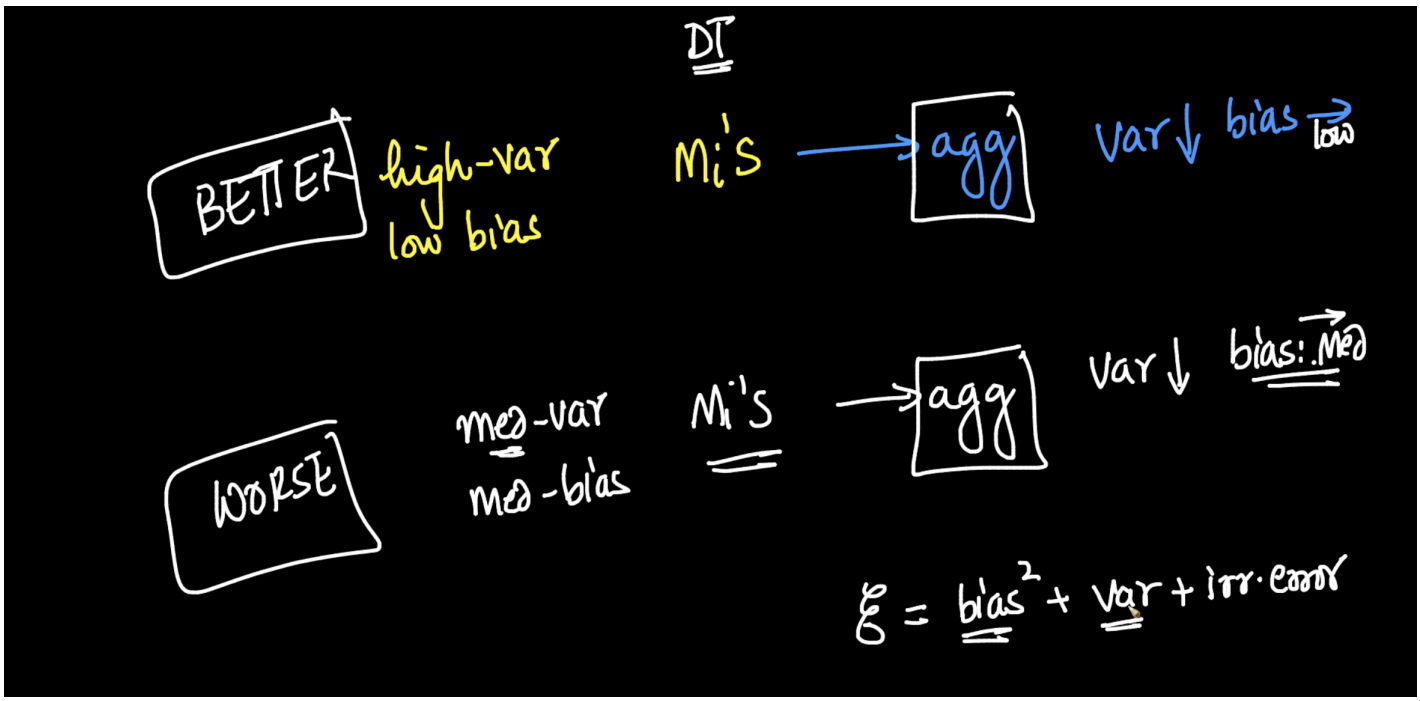
No, this is not required beacuse we are taking Base learners with equal weight



- We saw that aggregation is the key in Random forest ,
- But we might have two different types of Base learners in the random forest

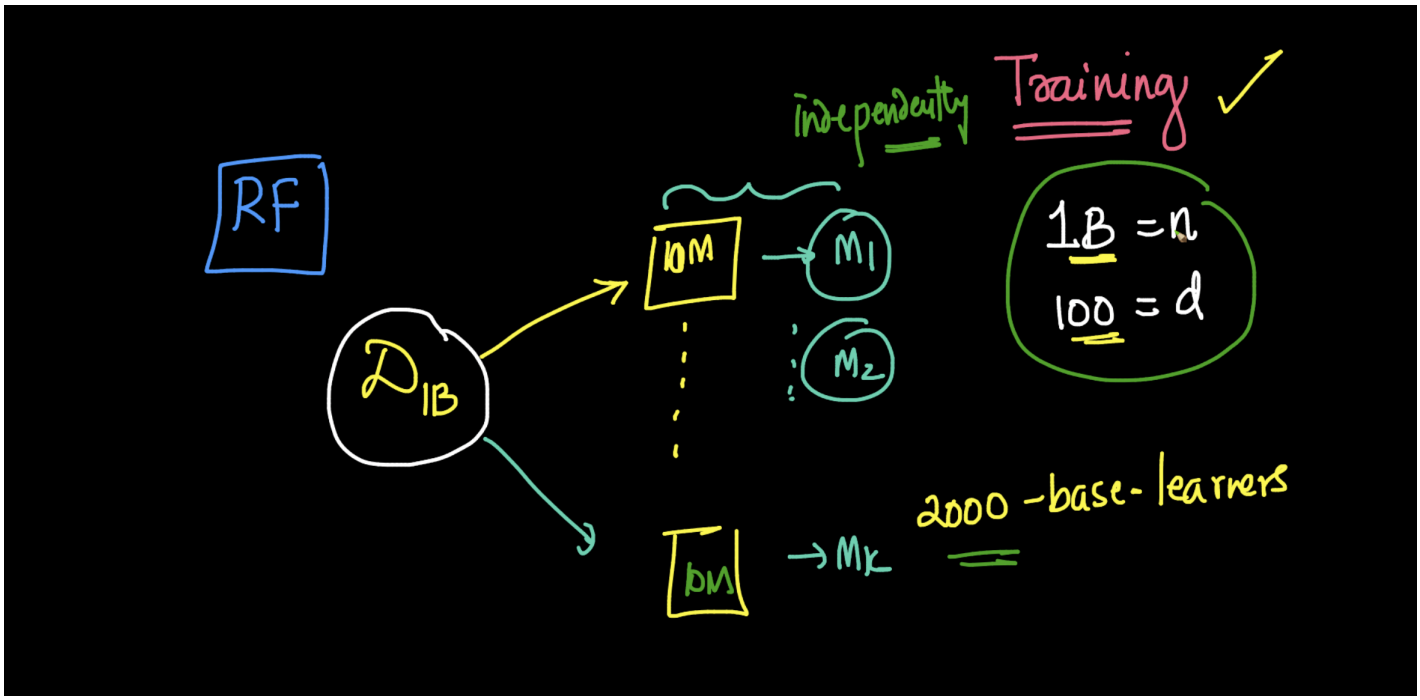
    - With high variance and low bias or
    - With medium varaince and medium bias
- Here as wwe do aggregation both the models become
    - with low variance and low bias and
    - with low variance and medium bias respectively

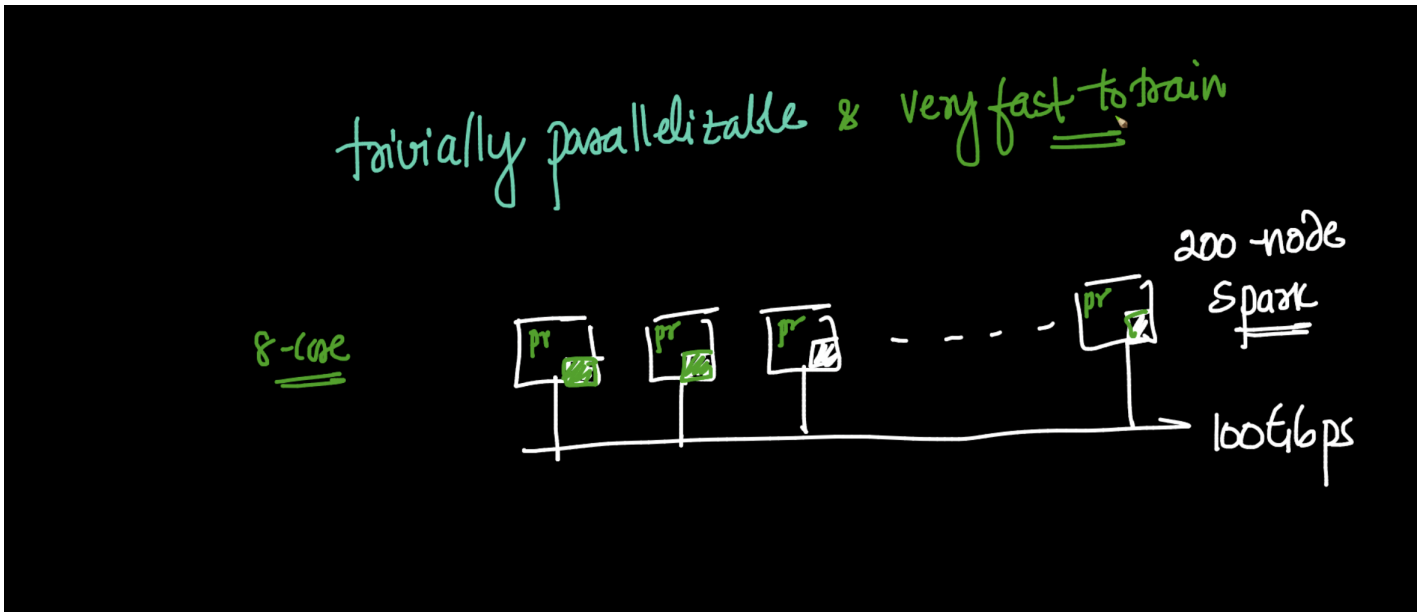
So it's better to have high variance and low bias base learners.

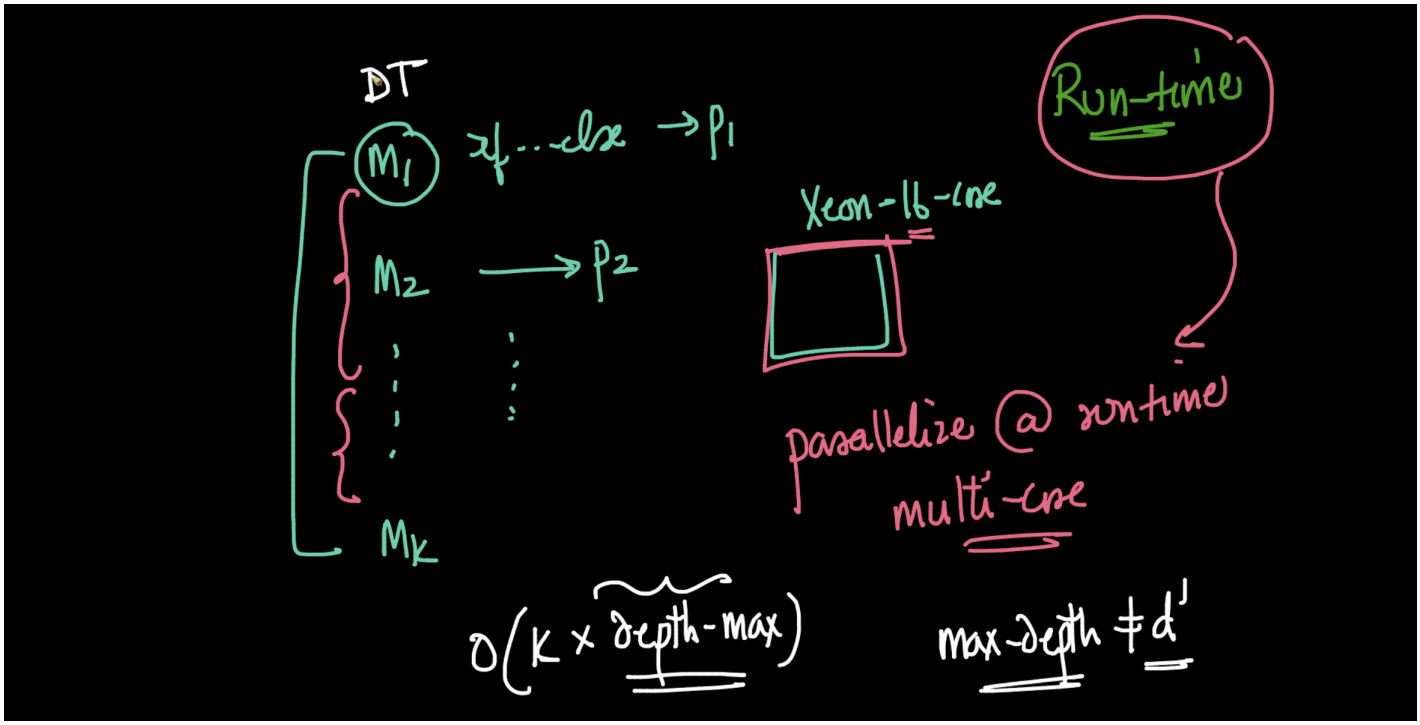


Training a random forest

- With present day improvements we can easily run a Random forest with a billion data with 10 million sample size and with thousand base learners
- These base learners can be trivially parallelised.
- As each models is **trained independently** we can even take these models to distributed computing
- In a distributed computing system each processor is given with different model's data set , the system can work on multiple models parallely, if the processor is a 8 core processor the datasets of 8 different trees can be worked on a system.
- So, we are parallelisig the Random forest on multiple cores due to which the process becomes fast.
- The time complexity here is O(k * max_depth of tree)


RF
Independently
Training
1B = n
100 = d
10M
M1
M2
DIB
DM
→ MK
2000 -base- learners

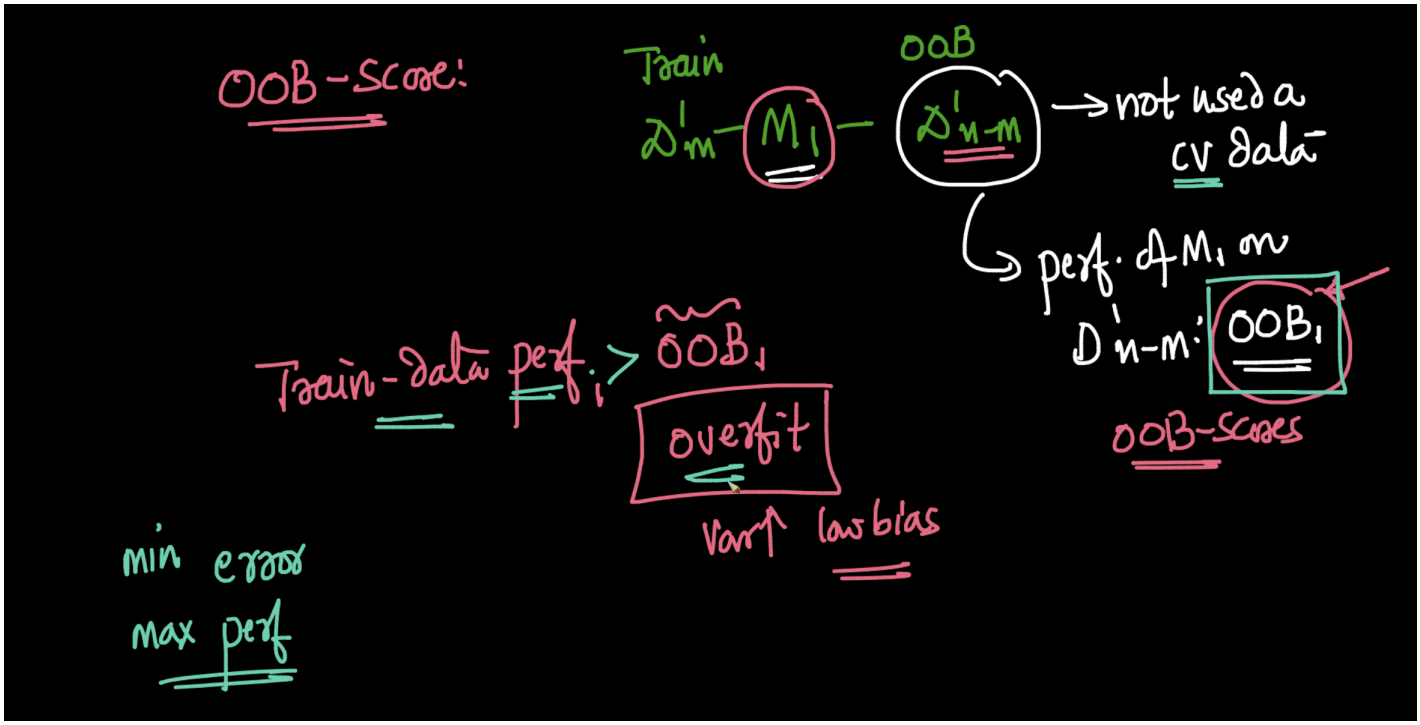

trivially parallelizable & very fast to train
8-core
200 node
Spark
100Gbps



OOB Score

As we already studied that after sampling the data for each model we have some data remaning which is the **Out Of Bag** data

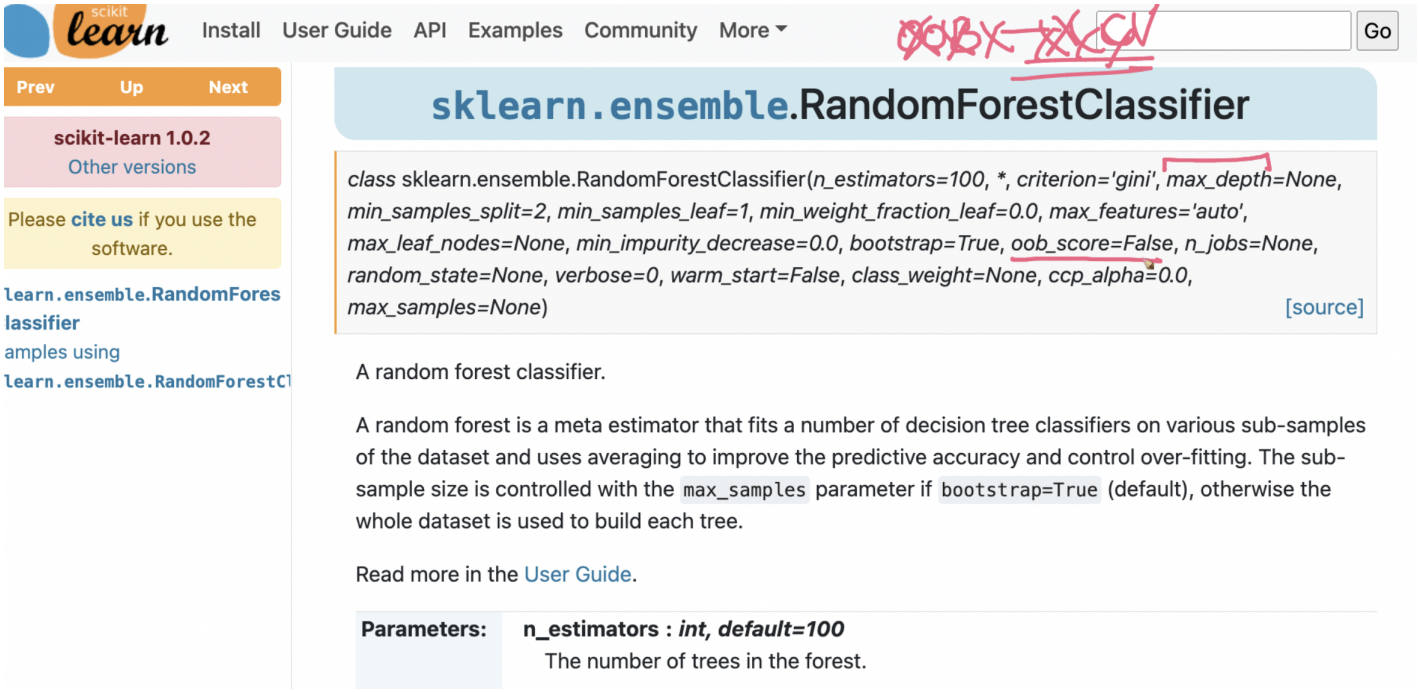
- This OOB data (D'_{n-m}) is not used as cross validation data, it is used to find the performace of the model on data, this is called the **OOB score** of the data
- When the trained data performance is more than the OOB data perfromance, we can conclude that the model is over-fitting, which we want upto some extend
- In the same way we can say that the model is overfitting if the trained data error is less than OOB data error



Sklearn library (terms and usage):

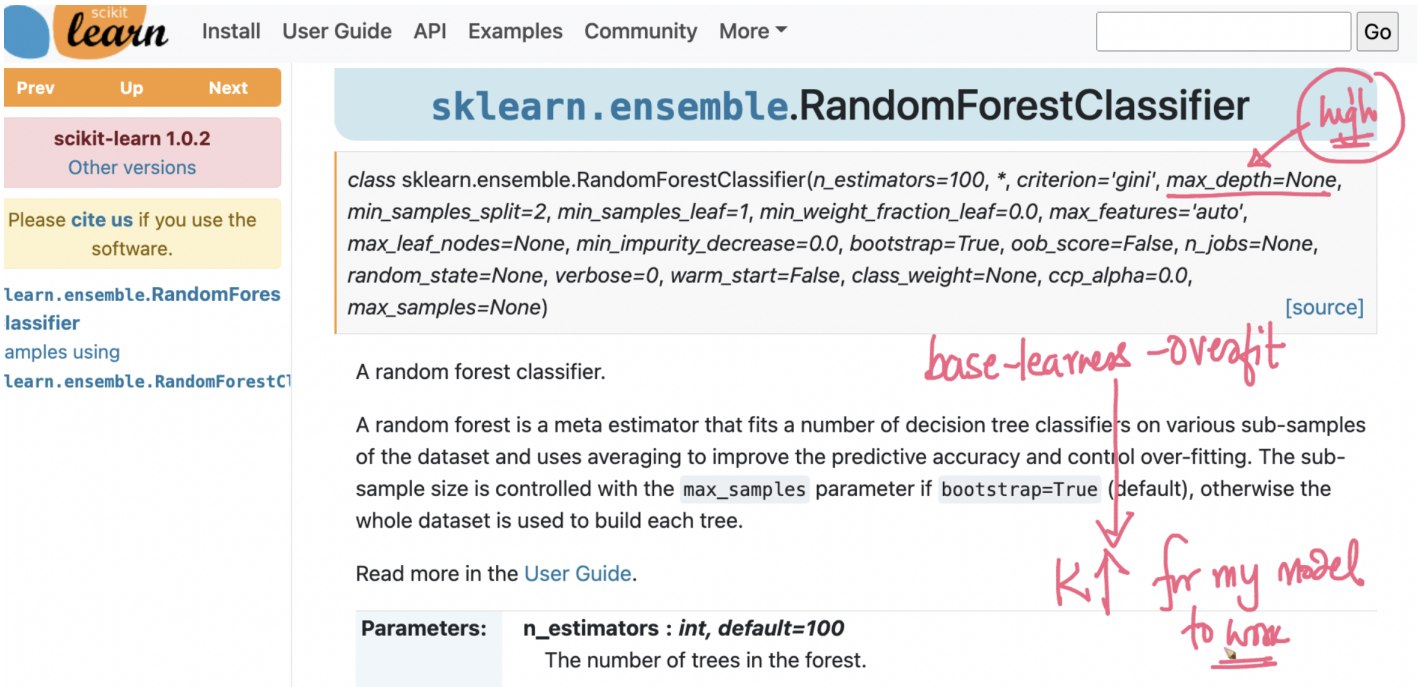
OOB score

- By default the OOB score is set to false, we can keep it true to get the OOB scores of each model
- Which can be compared by the perfromance of the trained data know the extent of overfitting in the base learner model



max depth

- This is used to decide the depth of the each Decision tree in the model
- As we already discussed, as depth increases the model overfits more due to which we may need more base learners for model to work

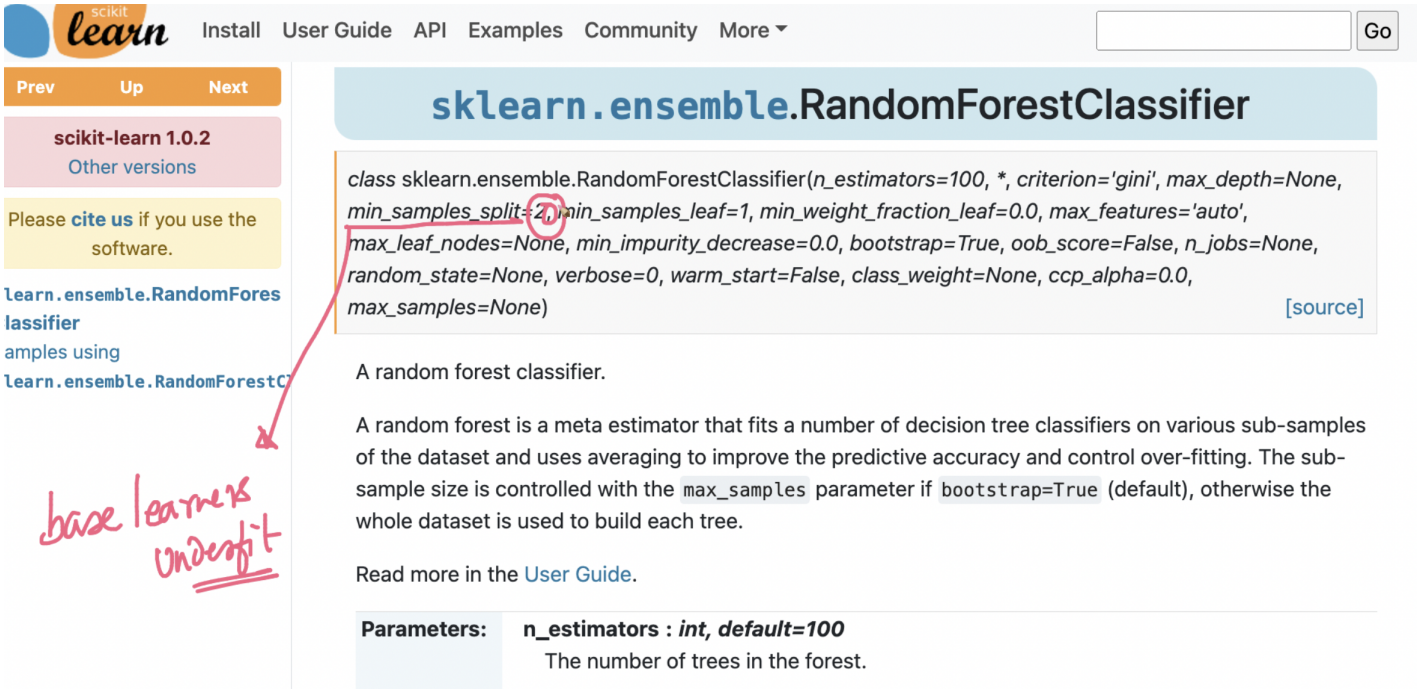


min_samples_split

- This is used to set the minimum number of data points required in a node to split further

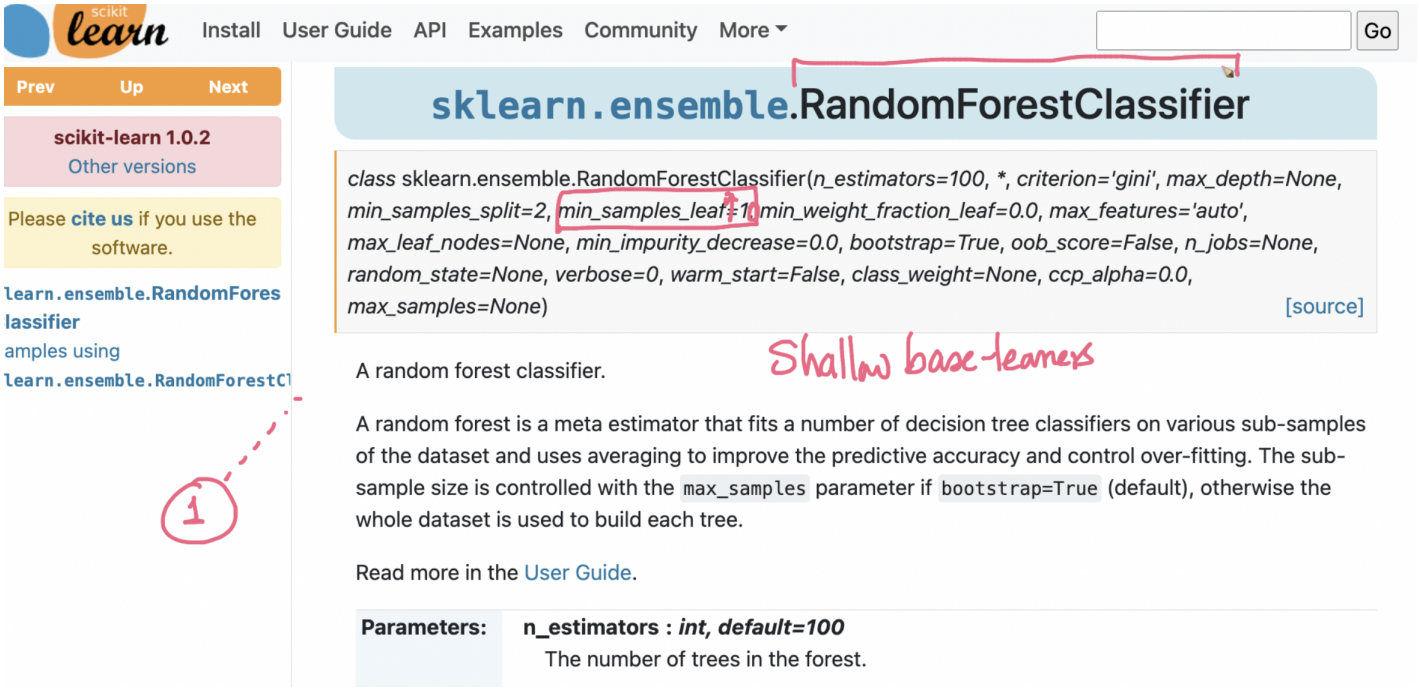
What happens if we increase the min_sample_split?

- If we increase the min_sample_split value the splitting stops if there are less number of datapoints than the set value, which controls the overfitting and may also result in underfitting and shallow trees.



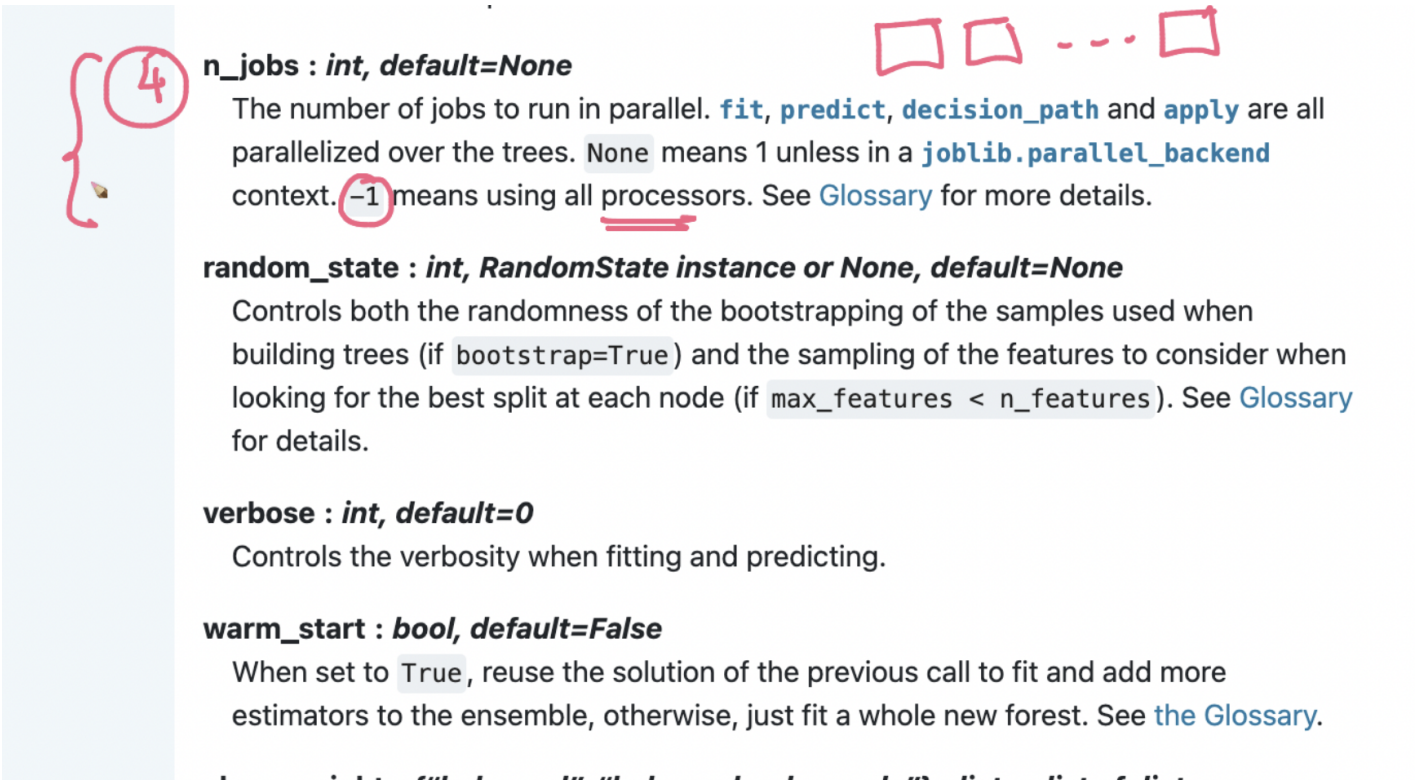
min_samples_leaf

- This gives the minimum number of data points required in the leaf node.
- If the value of min_samples_leaf is increased the base learners become shallower



n_jobs

- This gives the number of jobs that can be run in paralle.
- This does the multi core parallelisation
- If the value of n_jobs is set to 4, four cores of your processor will be used
- We can use all the cores of the processor by setting this value to -1.



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.

max_samples

- This is used to set the number of samples to draw from the dataset to train each of the base learners
- This is nothing but the row sampling which we studied earlier and represented as m
- We can give both int and float values, if we give a int value it takes as the number of samples to be considered and float is for percentage of the rows to be considered

max_samples : *int or float, default=None*

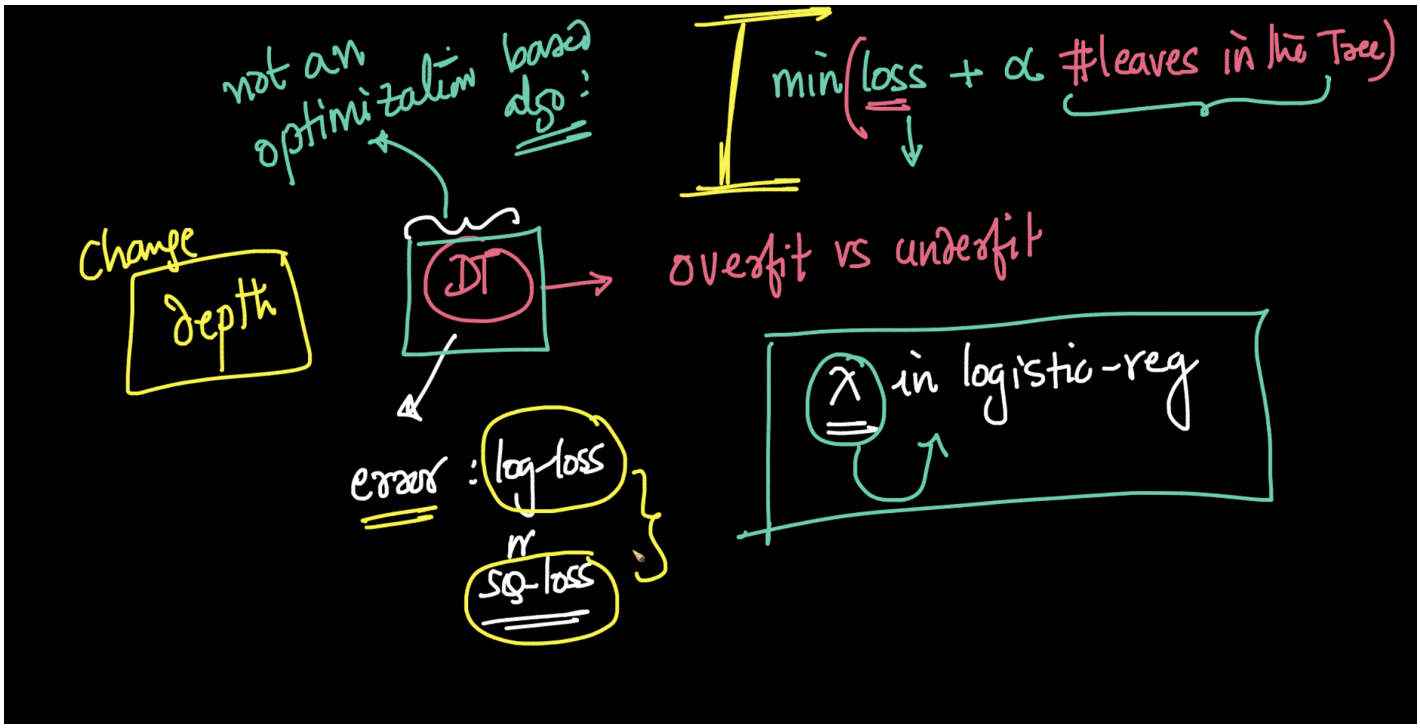
If bootstrap is True, the number of samples to draw from X to train each base estimator.

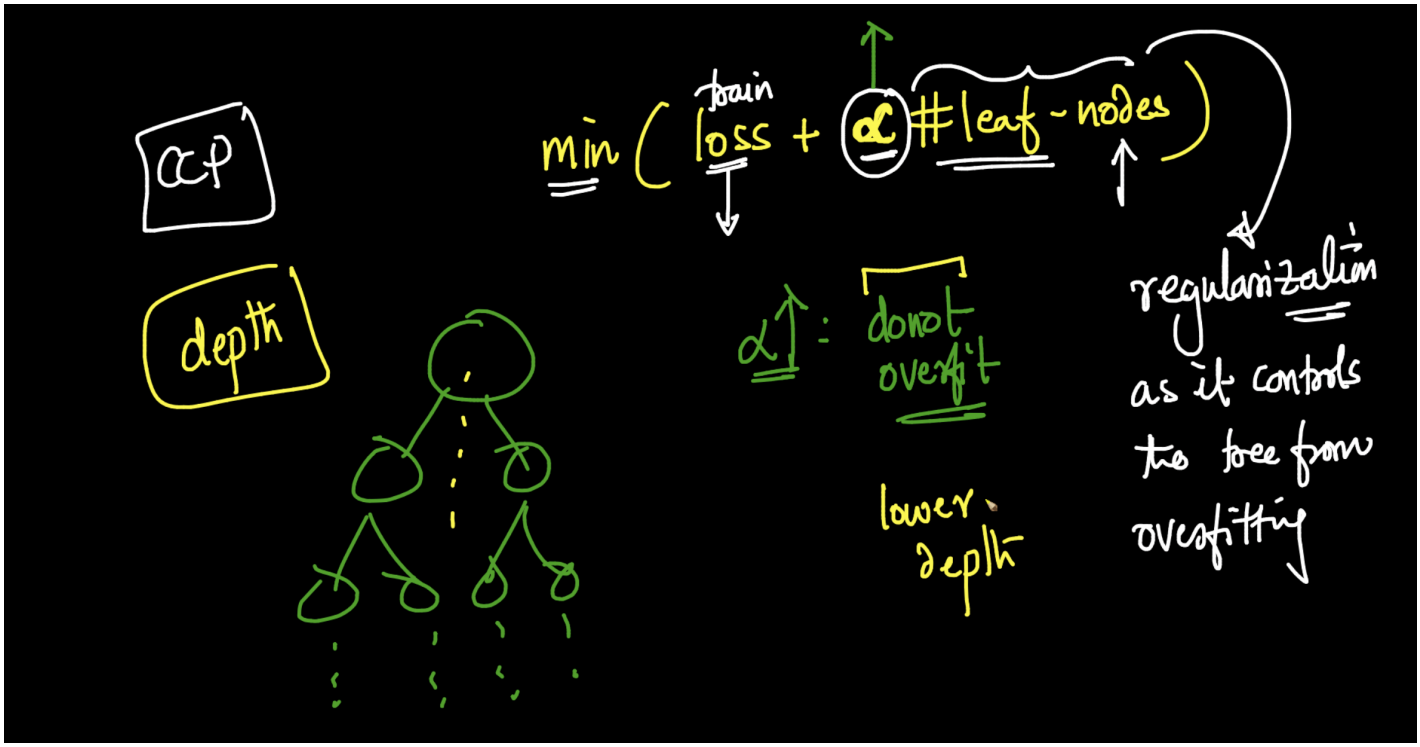
- If None (default), then draw `X.shape[0]` samples.
- If int, then draw `max_samples` samples.
- If float, then draw `max_samples * X.shape[0]` samples. Thus, `max_samples` should be in the interval `(0.0, 1.0]`.

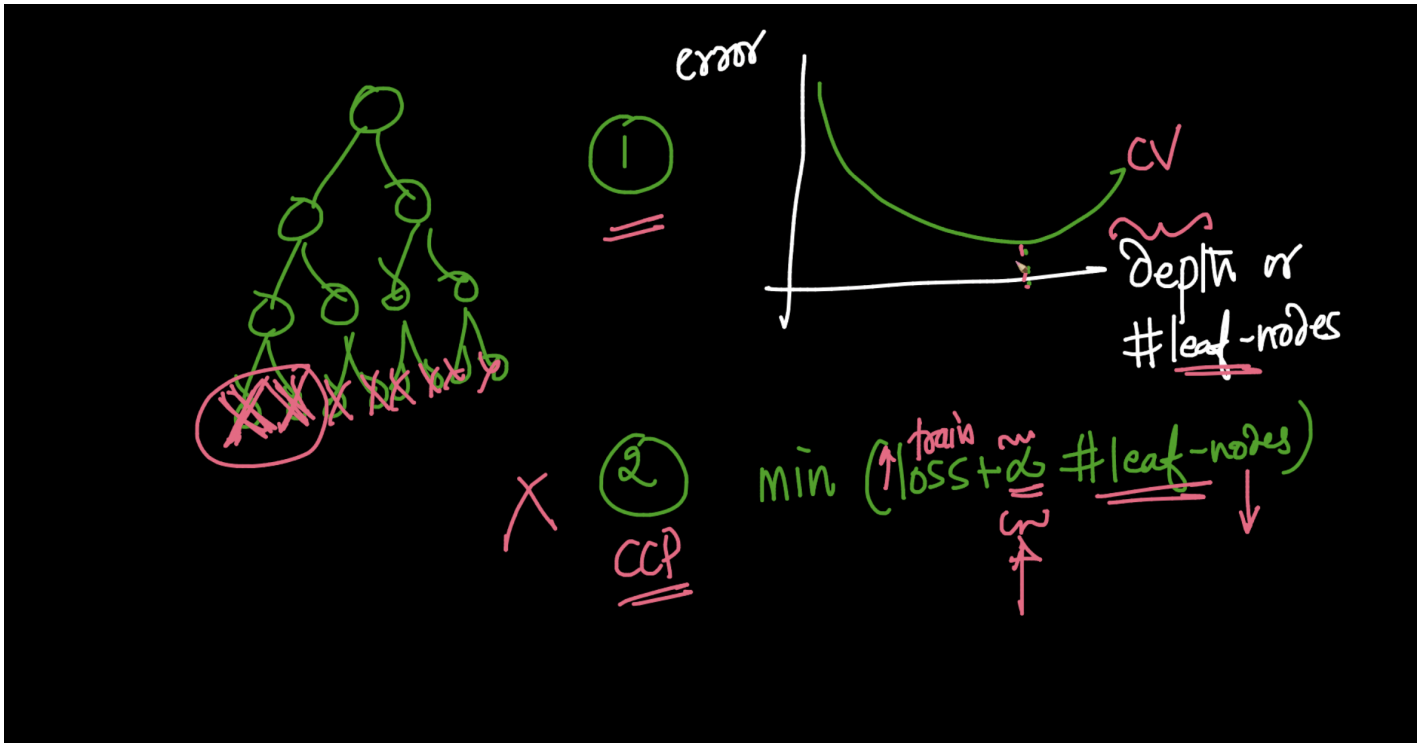
New in version 0.22.

ccp_alpha

- ccp is cost complexity pruning
- This is basicaly used for pruning the base learners
- We can control the overfitting and undefitting of the base learners using the value α, this is almost similar tp λ whixh we used in linear and logistic regression
- so the idea here is to minimise the loss associated with the decision tree and α times the number of terminal nodes (leaves) which controls overfitting
  - min(loss + α * number of leaves in the tree)
- As the depth of tree increases we know the loss decreases, where the number of leaf nodes in increases, this trade-off between the loss and number of leaves can be controlled using α like regularisation.







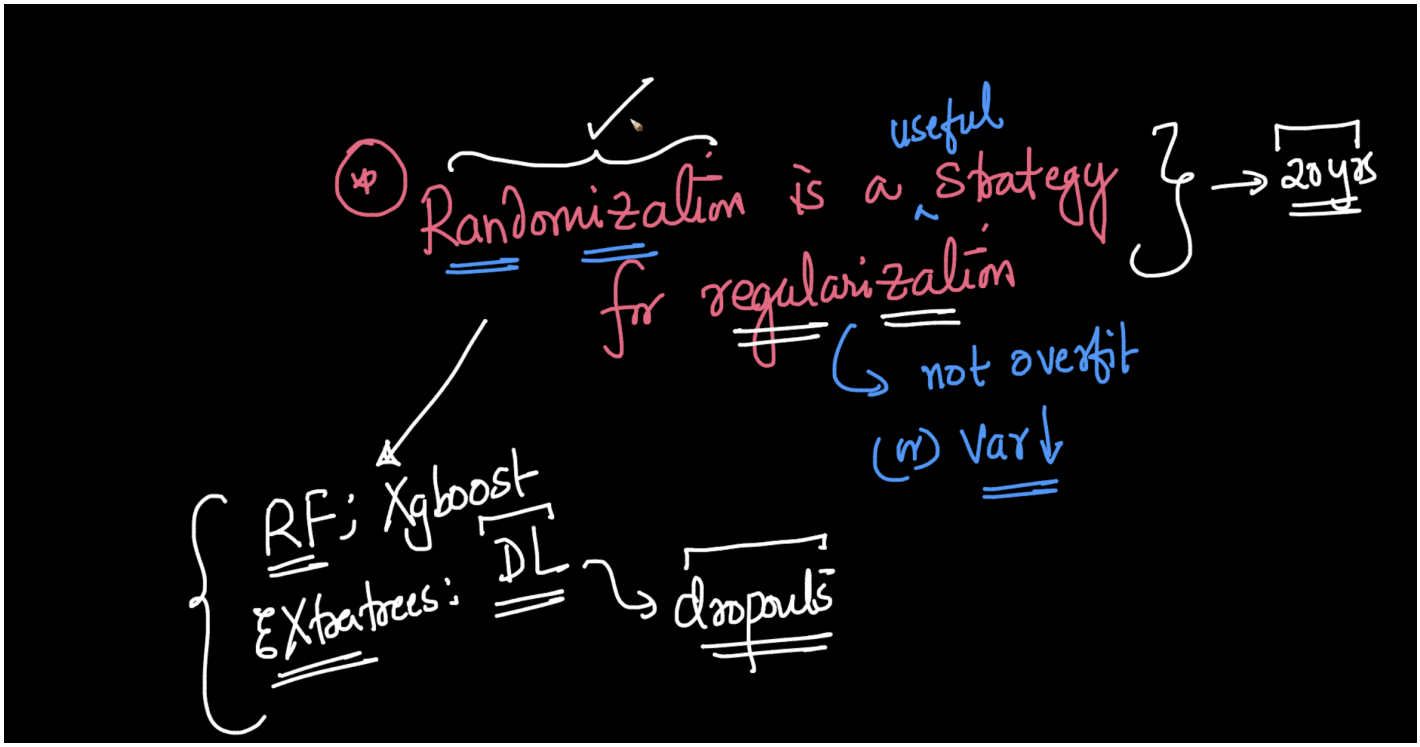
Extra Trees or Extremely Randomised Trees

let's compare these with Random Forest

What do we do in random forest?

- We do random row sampling and column sampling and then we do aggregation, in which this randomisation and aggregation play a key role in reducing the variace keeping the bias similar, further avoiding overfitting

Randomisation is a great, useful and very powerful strategy for Regularization



what do extra trees do?

In extra tree does random row and column sampling and aggregation just like decision trees but it also randomly picks the threshold (τ) to split for numerical features.

- In decsision trees we saw using the features values as threshold we calculate information gain and then choose the thresold of the split , basing on the values of information gain
- In an extra tree we choose this threshold (τ) also randomly ,adding one more randomisation on top of random forest,
    - say there are m rows in a feature we randomly select few rows and set the threshold basing on their IG values
- This is useful when the m is large, but these require more base laerners as the trees are not perfect
- Hence these are not widely used


Col. sampling