Decision Trees, Random Forests, and Ensembles

Sudhakar Kumar, student, IIT Bombay
Rishav Arjun, student, IIT Bombay

SHALA2020.GITHUB.IO
2020.04.27 :: 21:00 UTC+5:30

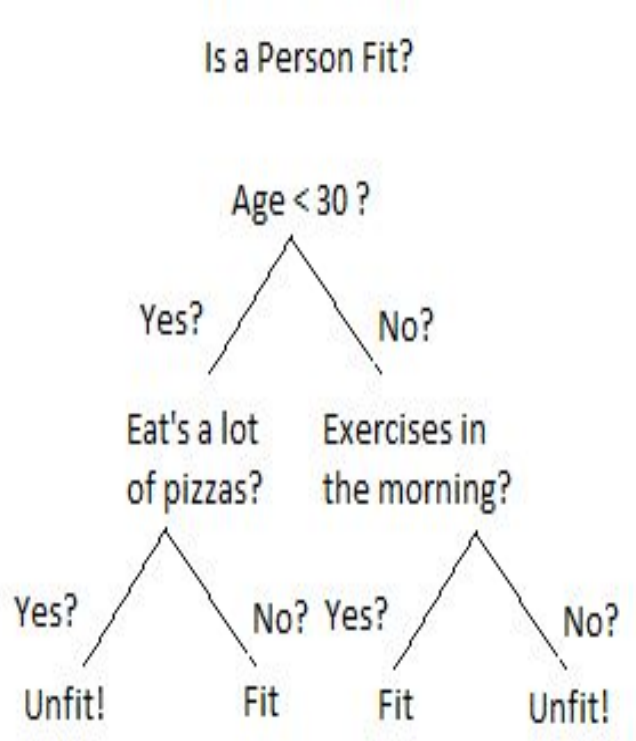
Learning objectives

- Define decision tree
- Difference between Decision tree and Linear approach
- Greedy algorithm (C4.5)
- Impurity metrics and their calculation
  - Information gain
  - Gini Index
- Hyperparameter
- Pruning effect (Reduce overfitting)
- Introduction of Ensemble of trees
- Bagging and Boosting
- Randomforest, AdaBoost, GB, SGB

What is Decision Tree ?

A decision tree is a flowchart-like structure in which each internal node represents a “test” on an attribute, each branch represents the outcome of the test, and each leaf node represents a class label.

The paths from root to leaf represent classification rules.



https://medium.com/greyatom/decision-trees-a-simple-way-to-visualize-a-decision-dc506a403aeb

Decision tree Vs Linear approach

Both are supervised learning model

Linear approach

- Generally used for a continuous target variable
- They depends on a polynomial (like a straight line) to fit a dataset

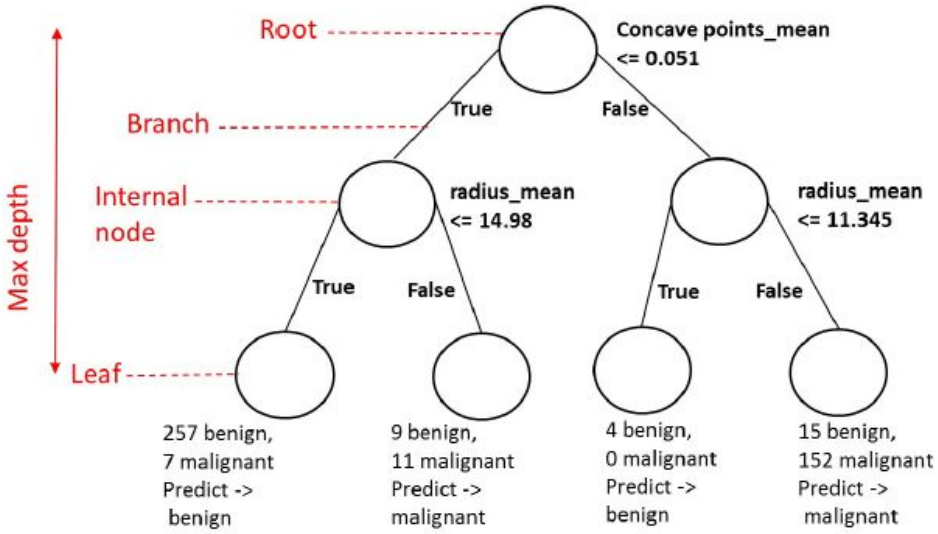
Tree approach

- Best for classification problems
- Based on tree-like model that only contains conditional control statements
- Required comparatively less amount of data preparation
- Decision planes are perpendicular to the corresponding feature axes

Building Blocks of a Decision-Tree

Three kinds of nodes:

1. Root: no parent node, question giving rise to two children nodes.
2. Internal node: one parent node, question giving rise to two children nodes.
3. Leaf: one parent node, no children nodes → Prediction



Essence of the learning algorithm

- Iteratively, at each node, find a variable and its threshold to split the data
- Such that purer samples of populations go into each of its children than what came into itself (the parent), on an average

Impurity metrics (Classification)

1. Entropy
2. Gini Index

$$Q_\tau(T) = \sum_{k=1}^{K} p_{\tau k} \ln p_{\tau k}$$

$$GINI(t) = 1 - \sum_{j} [p(j \mid t)]^2$$

p(j | t) is the relative frequency of class j at node t

Purer node → lower entropy

Information gain(IG): Difference between parent node impurity and **weighted** child node impurity

$$IG(\underbrace{f}_{feature}, \underbrace{sp}_{split-point}) = I(parent) - \left(\frac{N_{left}}{N} I(left) + \frac{N_{right}}{N} I(right) \right)$$

Information Criterion for Regression-Tree

$$I(\text{node}) = \underbrace{\text{MSE(node)}}_{mean-squared-error} = \frac{1}{N_{node}} \sum_{i \in node} \left(y^{(i)} - \hat{y}_{node}\right)^2$$

$$\underbrace{\hat{y}_{node}}_{mean-target-value} = \frac{1}{N_{node}} \sum_{i \in node} y^{(i)}$$

C4.5 – tree learning algorithm

Select a set of variables

- For each variable, find optimal threshold
  - Maximal decrease in (weighted) impurity
- Split based on the best variable (the one which corresponds to best Information gain)
- Repeat at each sub-tree

Pruning effect

- Pruning is a technique that reduces the size of decision trees by removing sections of the tree that provide little classification power.
- Shorter hypotheses are preferred as if a short hypothesis fits data it’s unlikely to be a coincidence
- Reduces the chance of overfitting and leads to more generalization
- Here, the node becomes a leaf and is assigned the most common classification
- *Note*: Nodes are removed only if the resulting tree performs no worse on the validation set

Two strategies of pruning

- Stop growing the tree earlier, before perfect classification
- Allow the tree to overfit the data, and then post prune the tree (XGBoost)

Post pruning Rule:

Prune (generalize) each rule by removing those preconditions whose removal improves accuracy over the validation dataset

Parameter and hyperparameter of CART model

Parameters: learned from data

- CART example: split-feature of a node, split-point of a node, ...

Hyperparameters: not learned from data, set prior to training

- CART example: max_depth , min_samples_leaf , splitting criterion …

Here again we can use hyperparameter tuning for improving the performance

Regression using a decision tree

- A decision tree based model is prefered when, there are non-linear or complex relationships between features and labels
- **Mean square error** or similar metrics can be used instead of Entropy or Gini impurity
- It involves the partitioning of data into subsets that contain instances with similar values
- Leaf node represents a decision on the numerical target. When the number of instances is more than one at a leaf node we calculate the average as the final value for the target

http://www.saedsayad.com/decision_tree_reg.htm

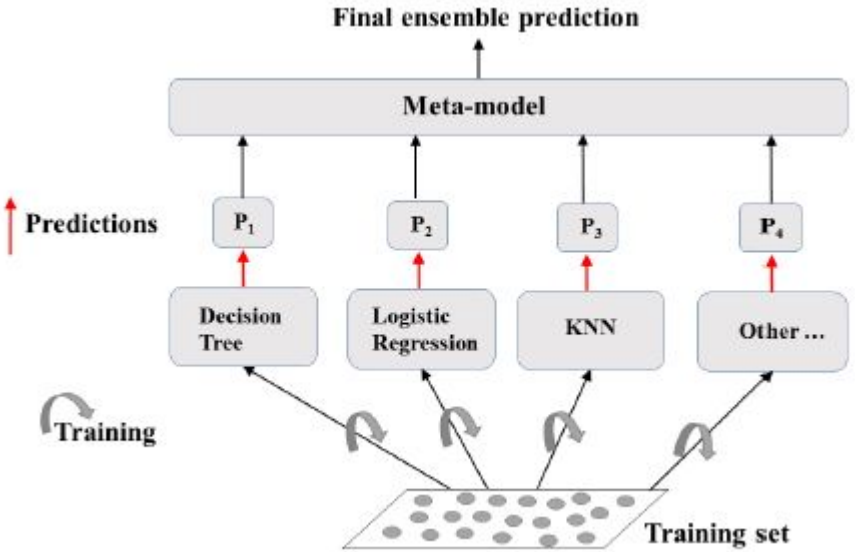
Hyperparameters of Random Forest

- max_depth: Longest path between the root node and the leaf node
- min_sample_split: the minimum required number of observations in any given node in order to split it
- max_leaf_nodes: condition on the splitting of the nodes in the tree and hence restricts the growth of the tree
- min_samples_leaf: minimum number of samples to be present in the leaf
- n_estimators: Number of trees
- max_sample: fraction of the original dataset to be given to any individual tree of random forest
- max_features: the number of maximum features provided to each tree in a random forest

https://www.analyticsvidhya.com/blog/2020/03/beginners-guide-random-forest-hyperparameter-tuning/

Ensemble Learning

- Train different models on the same dataset.
- Let each model make its predictions.
- Meta-model: aggregates predictions of individual models.
- Final prediction: more robust and less prone to errors.



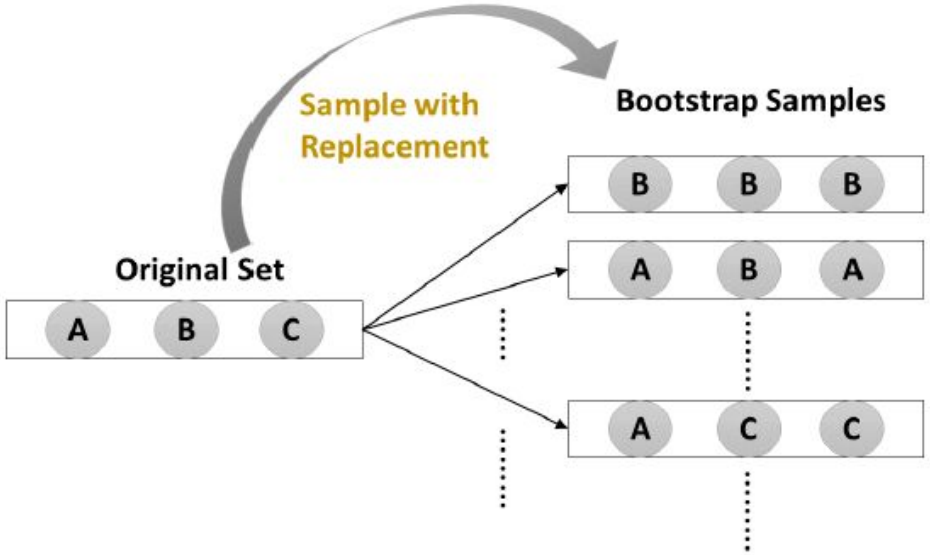
Bagging

- Also known as Bootstrap aggregating
- Uses a technique known as the bootstrap
- Reduces variance of individual models in the ensemble
- It considers homogeneous base models (weak learners), learns independently from each in parallel and combines them following some kind of deterministic averaging process
    - Classification: Aggregates predictions by majority voting
    - Regression: Aggregates predictions through averaging

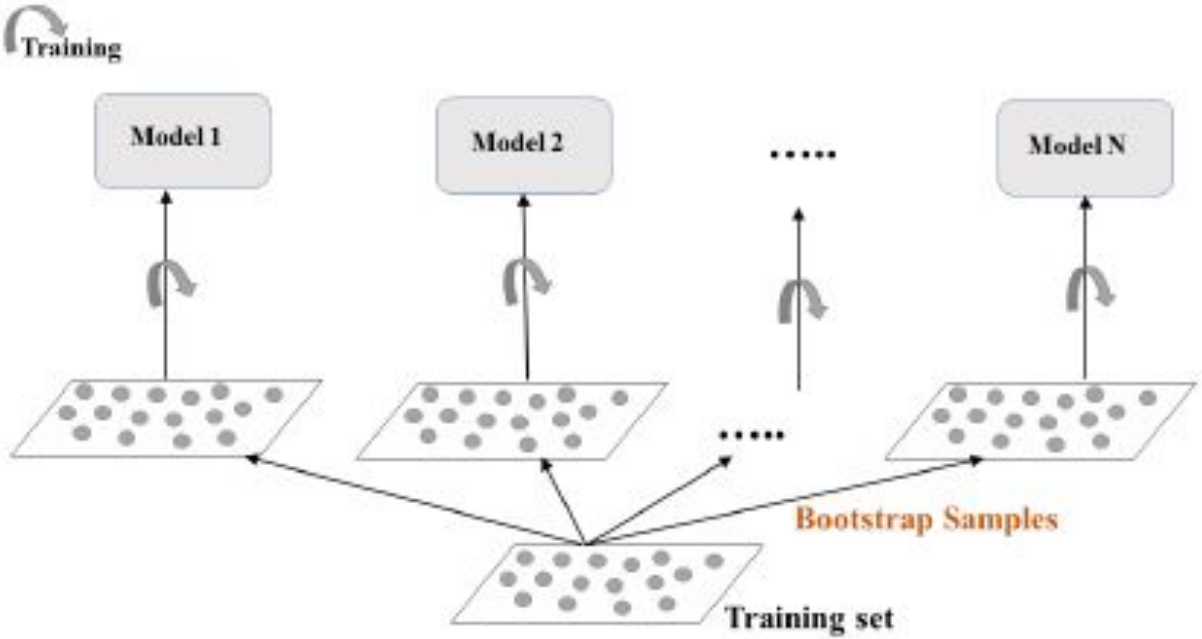
In case of **Random Forests**, base estimators are only decision trees. Each estimator is trained on a different bootstrap sample. RF introduces further randomization in the training of individual trees

Bootstrap

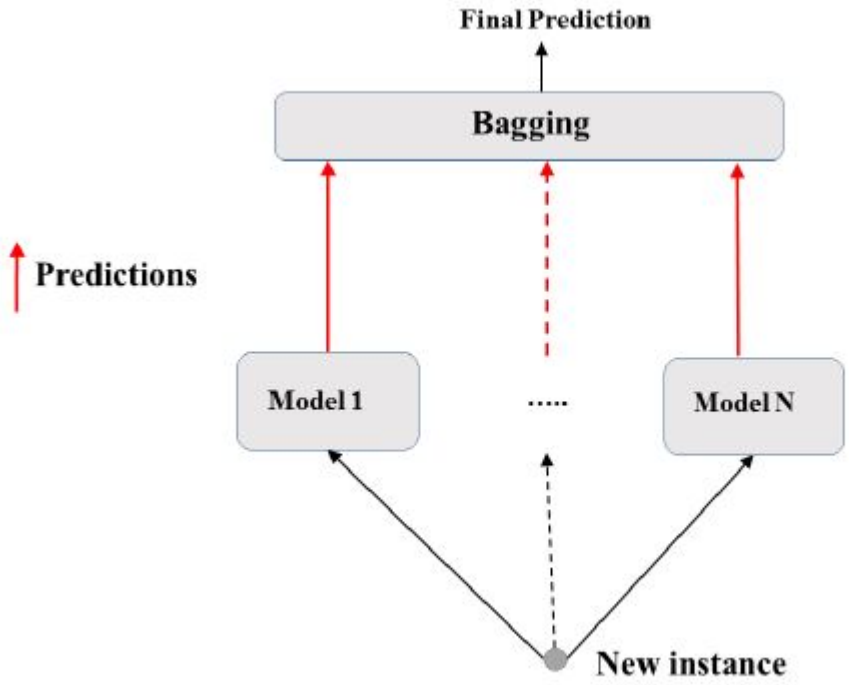
It is a resampling method by independently sampling with replacement from an existing sample data with same sample size



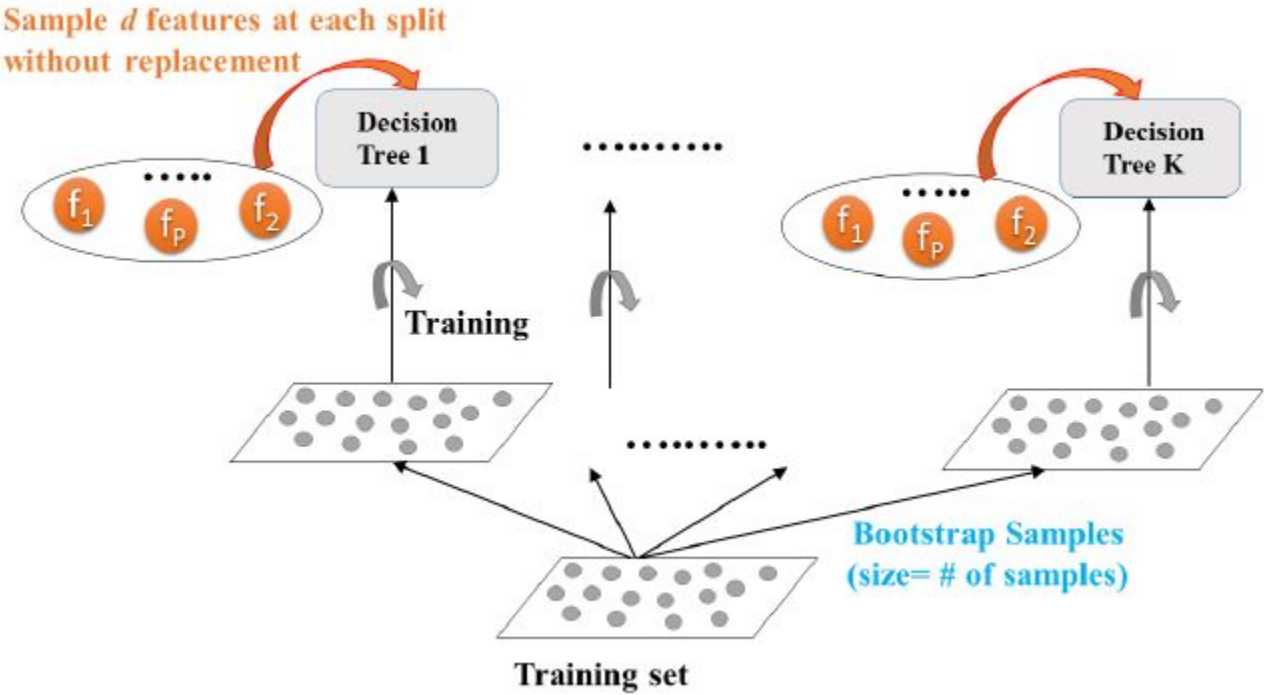
Bagging: Training



Bagging: Prediction



Random Forests: Training



Some interesting properties of random forest

Strengths:

- Getting variable importance
- Relatively less prone to overfitting
- Can handle a mixture of continuous and categorical inputs

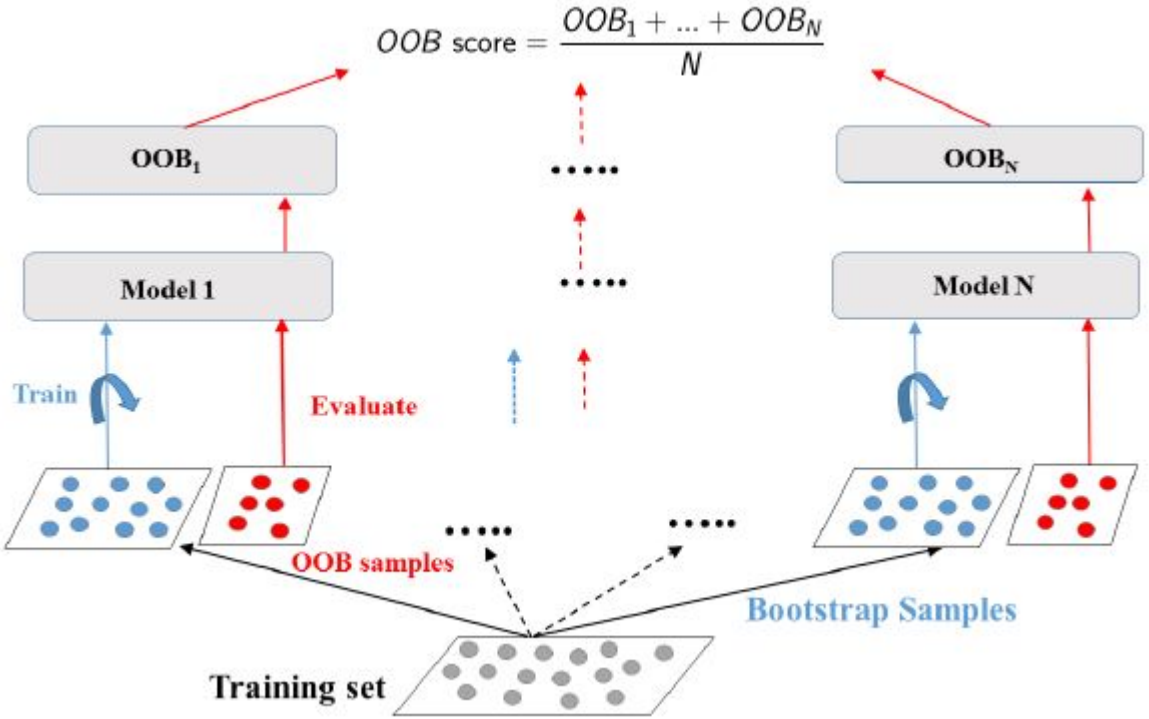
Weaknesses:

- Decision boundaries of trees are parallel to feature axes
- Do not learn feature transformations

Computing variable importance

- In RF, some instances may be sampled several times for one model during bootstrapping while, other instances may not be sampled at all
- On average each model, 63% of the training instances are sampled and the remaining 37% constitute the **OOB (out of bag) instances**
- In every tree grown in the forest, put down the oob cases and count the number of votes cast for the correct class. Now randomly permute the values of variable “x” in the oob cases and put these cases down the tree
- Subtract the number of votes for the correct class in the variable-x-permuted oob data from the number of votes for the correct class in the untouched oob data. The average of this number over all trees in the forest is the raw importance score for variable “x”
- *Note*: There is no effect of normalization or standardization of variables

OOB Evaluation



Boosting

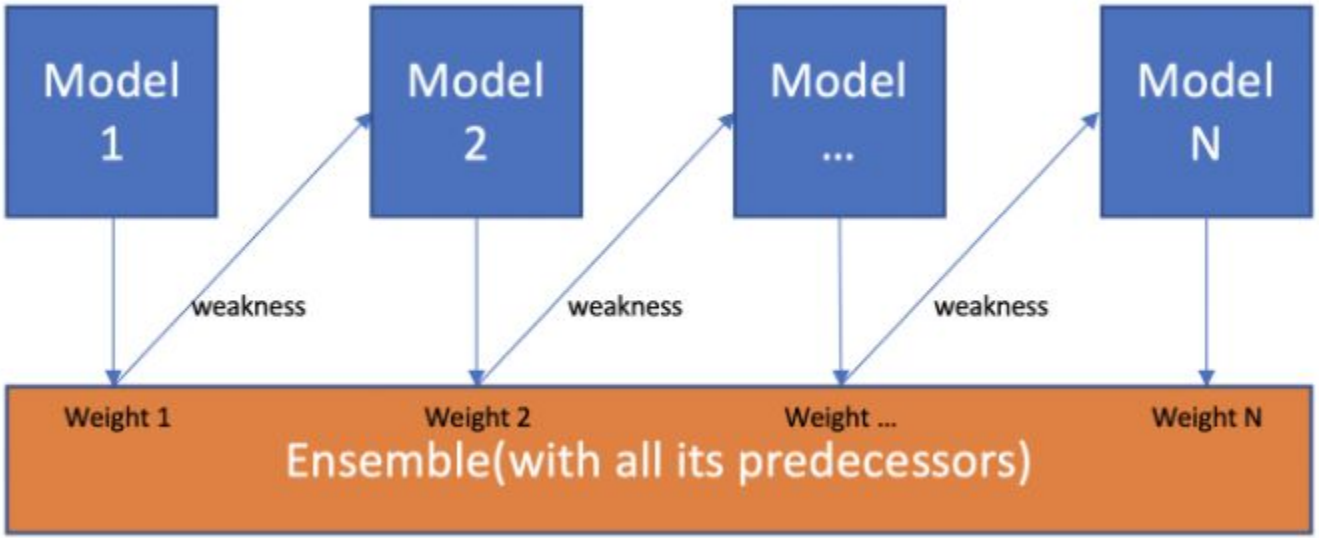
- It considers homogeneous base models, learns them **sequentially** in a very adaptative way (a base model depends on the previous ones) and combines them following a deterministic strategy
- Combined weak models are no longer fitted independently from each others
- Each new model focus its efforts on the most difficult observations to fit up to now, so that we obtain, a strong learner at the end

Note: We are not using Bootstrap in this case

Popular boosting methods: **Adaboost, Gradient Boosting**

https://towardsdatascience.com/ensemble-methods-bagging-boosting-and-stacking-c9214a10a205

Boosting training



Adaboost (Adaptive Boosting)

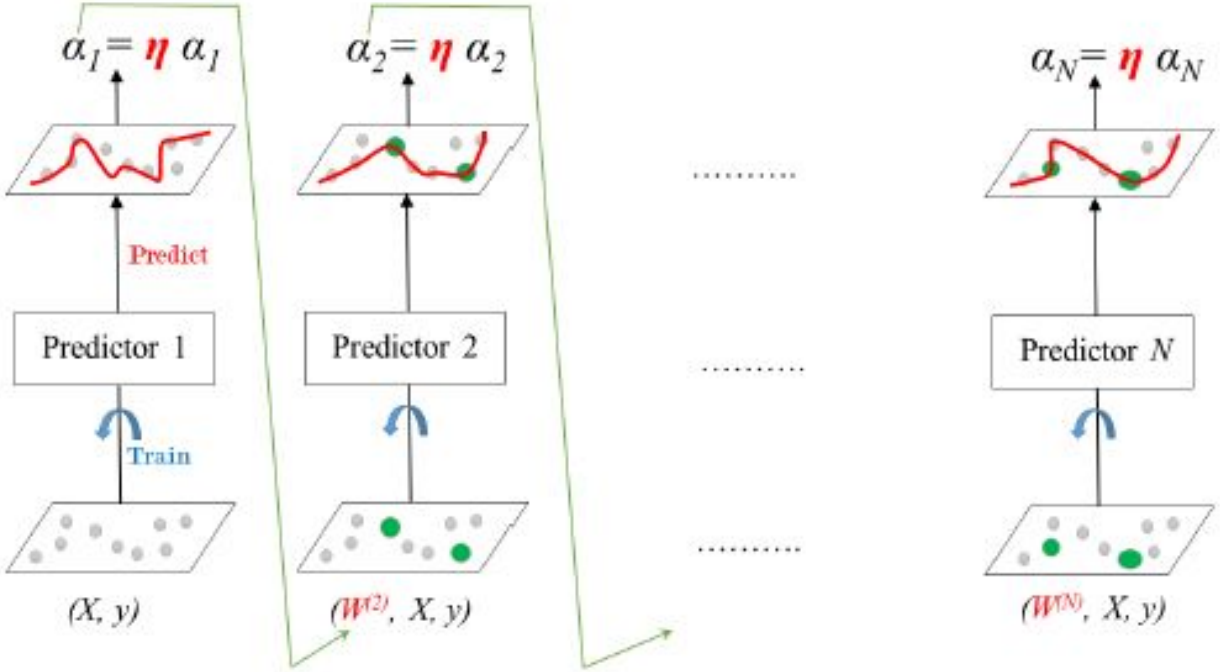
- Each predictor pays more attention to the instances wrongly predicted by its predecessor
- Achieved by changing the **weights of training instances**
- Each predictor is assigned a coefficient α
- α depends on the predictor's training error

Classification: Aggregates predictions by majority voting

Regression: Aggregates predictions through averaging

Adaboost training

Learning rate: $0 < \eta \leq 1$



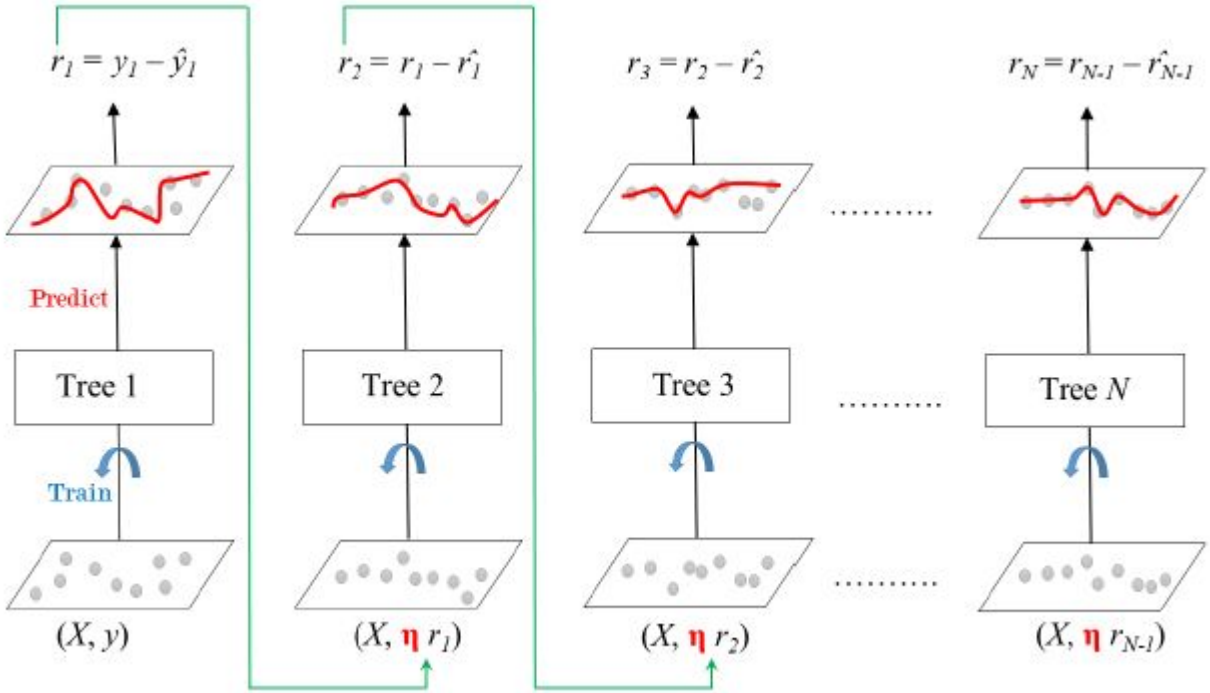
Gradient Boosting (GB)

- Does not tweak the weights of training instances.
- Fit each predictor is trained using its predecessor's residual errors as labels.
- Gradient Boosted Trees: a CART is used as a base learner.

Regression:

- $ypred = y_1 + \eta r_1 + ... + \eta r_N$

Gradient Boosted Trees for Regression



Stochastic Gradient Boosting (SGB)

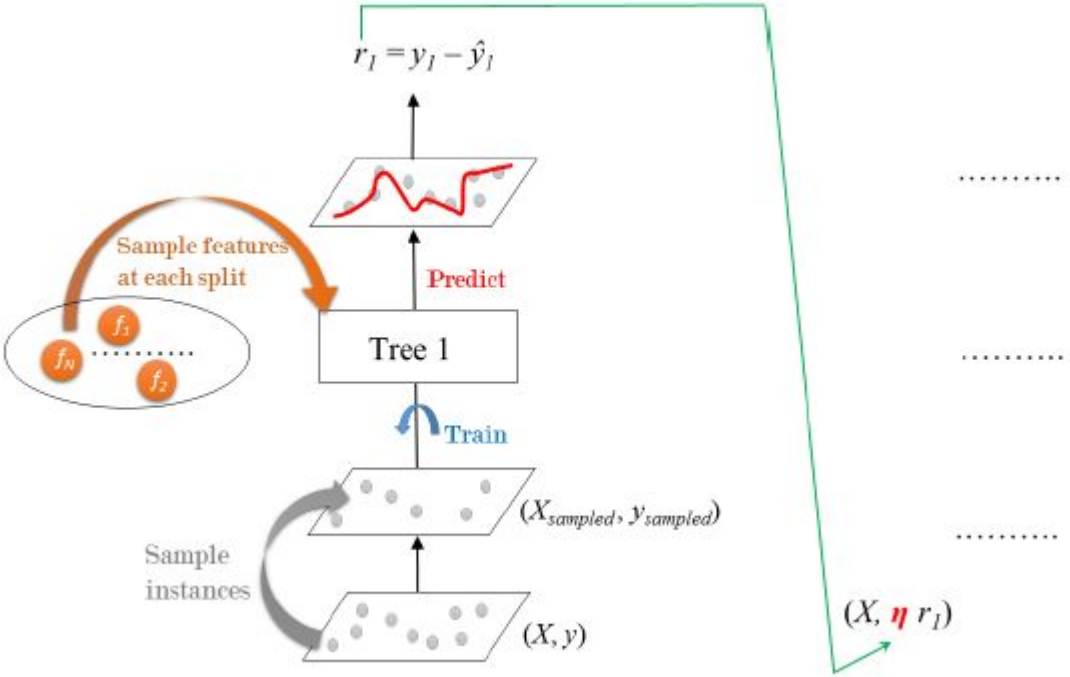
Gradient Boosting: Cons

- GB involves an exhaustive search procedure
- Each CART is trained to find the best split points and features which may lead to CARTs using the same split points and maybe the same features

In SGB

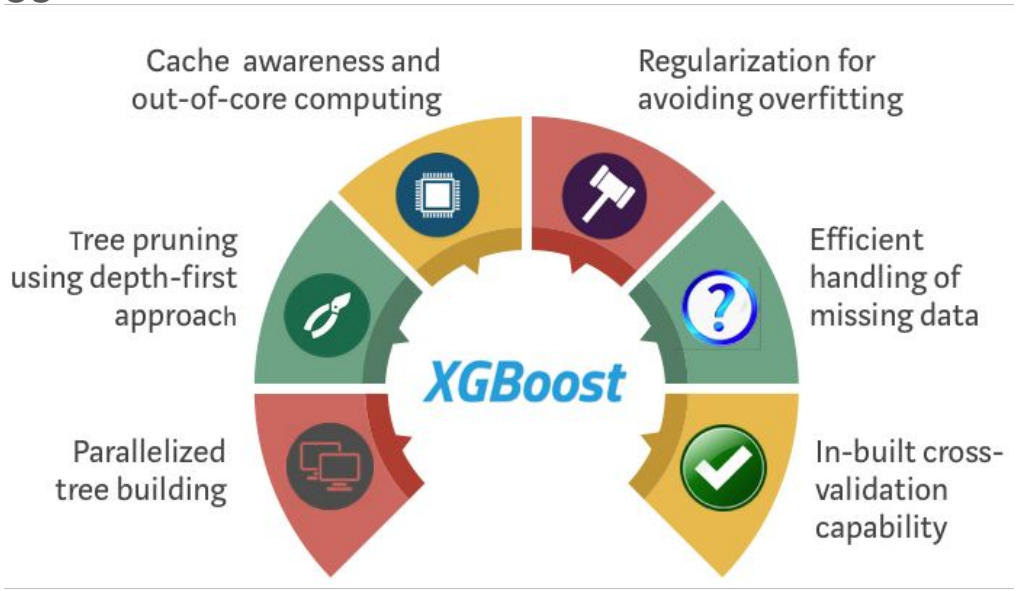
- Each tree is trained on a subset of rows of the training data (40% - 80%)
- Features are sampled (without replacement) when choosing split points
- Result: further ensemble diversity, further variance to the ensemble of trees

SGB: Training



eXtreme Gradient Boosting (XGBoost)

- Perfect combination of software and hardware optimization techniques
- Better Execution Speed.
- Better Model Performance
- Support regularized Gradient Boosting with both L1 and L2 regularization



https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d