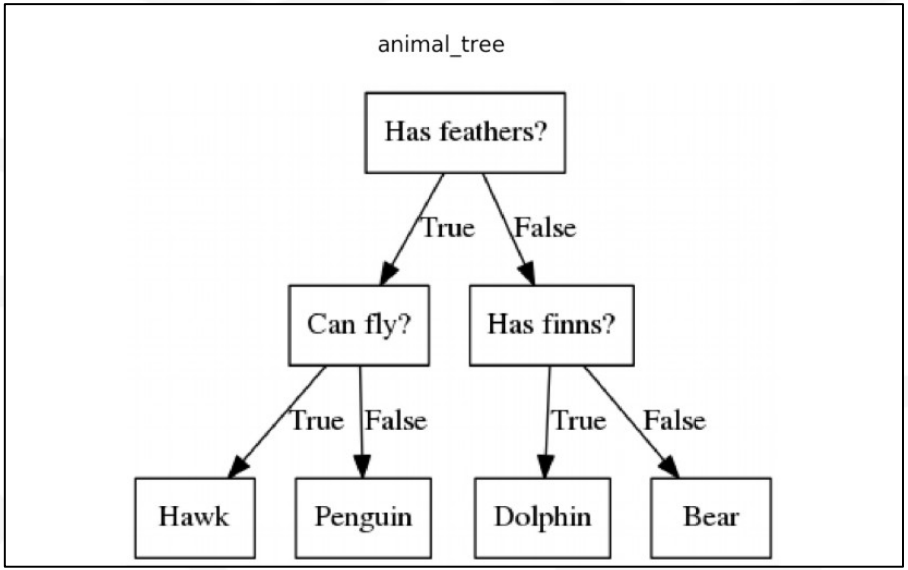
XGBoost

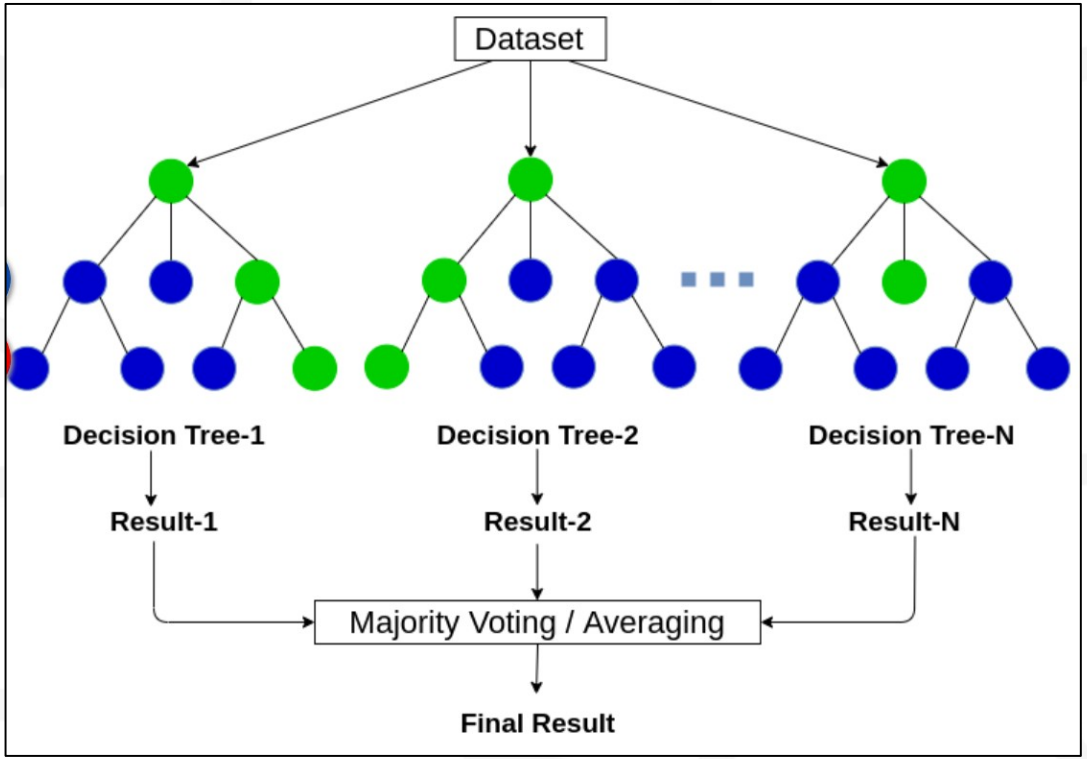
Basic Decision Tree

- Decision tree is a flow chart-type structure
- Each node represents a “test” which decide which direction the flow takes for a given data instance
- Can think of it as a sort of “20 questions” type of flow
- When you hit the terminal node at bottom – that is the predicted result



Random Forest

- Creates multiple decision trees – each taking a different subset of the data
- When a data instance is presented to be predicted each tree is followed to a terminal node
- Then the results are found by:
  - Voting for classification
  - Averaging for regression

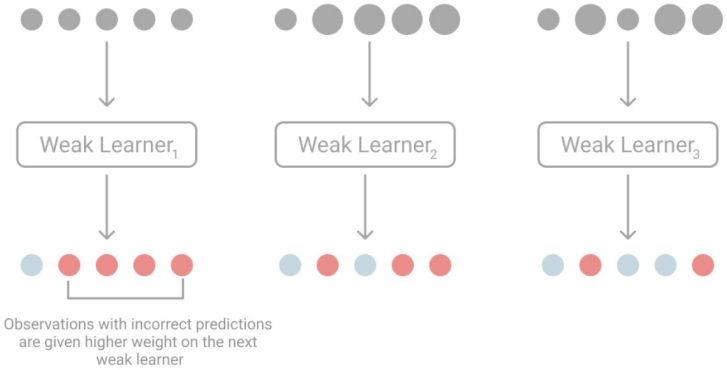


Boosting

Boosting is based on the assumption that it's much easier to find several simple rules to make a prediction than it is to find *the one* rule that is applicable to all data and generates the best possible prediction.

In boosting, you train the same **weak learner**, a model with simple rules, several times. Then combine its *weak* predictions into a single, more accurate result.

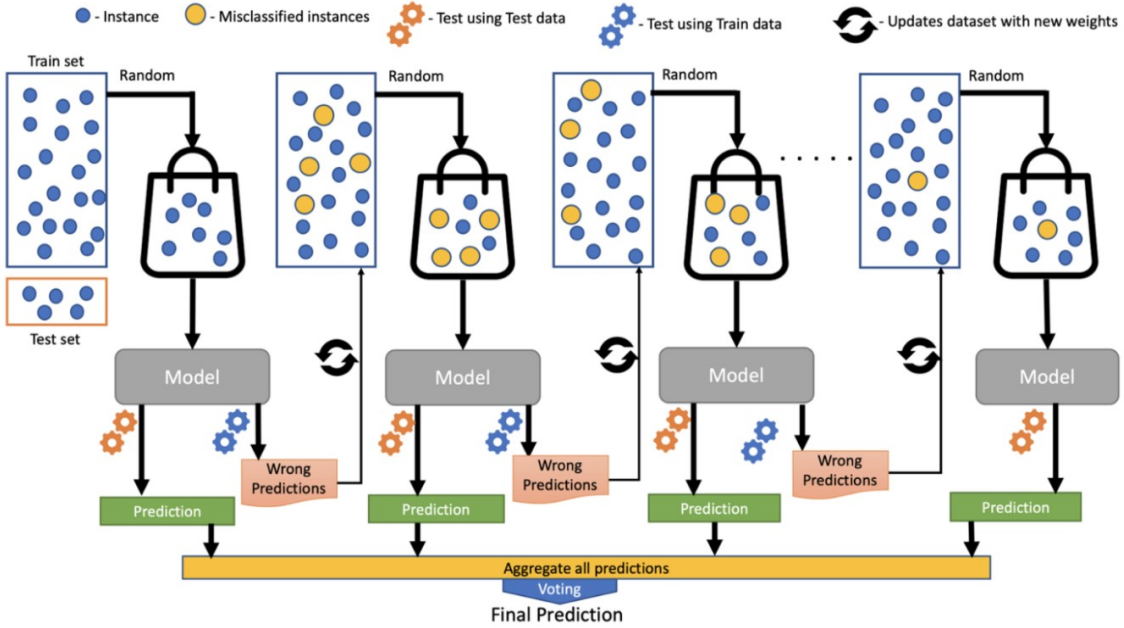
Predictions that were previously misclassified will have more weight on the next *weak learner*, so each *weak learner* focuses on the *hardest* observations.



Source

XGBoost

XGBoost, which stands for Extreme Gradient Boosting, is a tree-based ensemble method that uses an ensemble of basic decision trees (weak learners) to make its prediction.



Tree Based Method Analogy

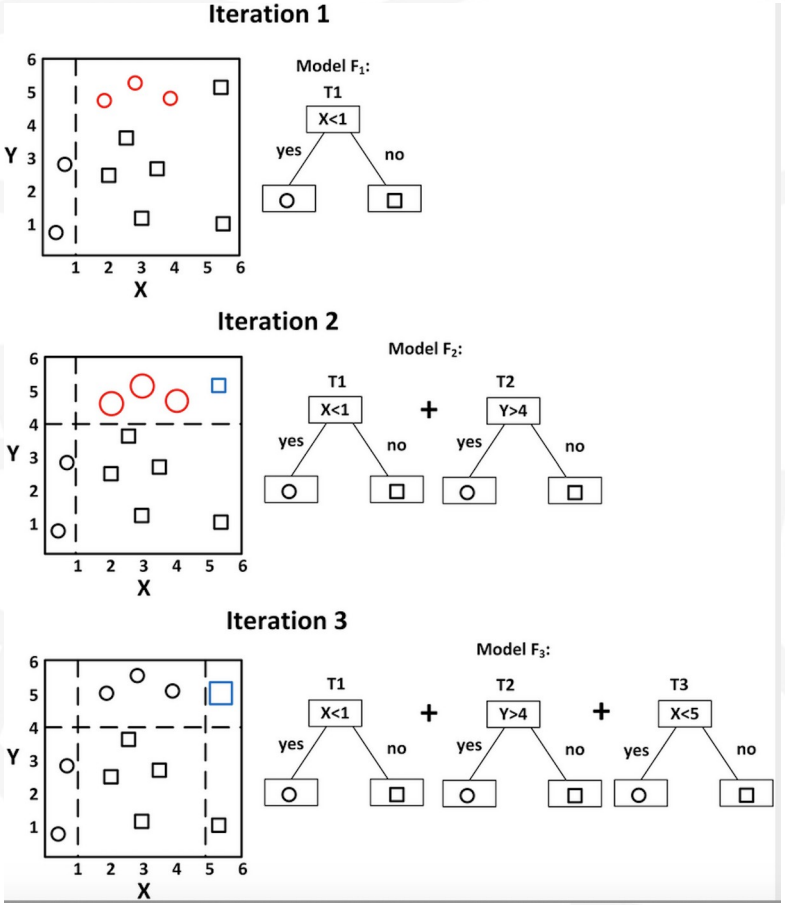
Decision Tree: Every hiring manager has a set of criteria such as education level, number of years of experience, interview performance. A decision tree is analogous to a hiring manager interviewing candidates based on his or her own criteria.

Random Forest: Now there is an interview panel where every interviewer has a vote. Every interviewer will only test the interviewee on certain randomly selected criteria.

XGBoost: This is an alternative approach where each interviewer alters the evaluation criteria based on feedback from the previous interviewer. This 'boosts' the efficiency of the interview process by deploying a more dynamic evaluation process.

Source

Example


Iteration 1
Model F_1:
T1
X<1
yes
no
Y
X
Iteration 2
Model F_2:
T1
X<1
yes
no
+
T2
Y>4
yes
no
Iteration 3
Model F_3:
T1
X<1
+
T2
Y>4
+
T3
X<5

Hyperparameters

XGBoost has numerous hyperparameters that can be set by the user. Some of the most common are given below:

- **learning_rate** - Shrinks the weight of new features in each boosting step. Must be between 0 and 1. Default is 0.3.
- **max_depth** - Maximum number of tree splits. Remember - the more splits, the more complex the tree. Can be any value greater than 0. Default is 6.
- **n_estimators** - Number of trees. Remember - generally, more trees improves performance but it also increases run time.
- **lambda** - L2 regularization on the weights. Can be any value. Default is 1.
- **alpha** - L1 regularization on the weights. Can be any value. Default is 0.

Pros & Cons

Pros

- Allows you to see feature importance
- Robust to outliers
- Helps to reduce overfitting
- Works well with large datasets
- Good model performance

Cons

- Harder to interpret than decision trees
- Many hyperparameters that can be tuned