Algorithm	Key Idea	Base Learners	How It Combines Trees	Strengths	Weaknesses
Random Forest	Bagging (Bootstrap Aggregation)	Decision Trees (usually deep)	Builds many trees independently, averages their predictions (for regression) or majority vote (for classification)	Reduces overfitting, robust, simple to tune	Can be slower with very large forests, less interpretable
AdaBoost	Boosting (sequential learning)	Weak learners (often shallow trees/stumps)	Trains trees sequentially; each tree focuses more on misclassified samples	Can improve accuracy over weak learners, simple boosting method	Sensitive to noisy data and outliers
Gradient Boosting	Boosting using gradients of loss function	Weak learners (usually shallow trees)	Trees are added sequentially to correct residual errors of previous ensemble	High accuracy, flexible with loss functions	Slower to train, prone to overfitting if not tuned
XGBoost	Optimized Gradient Boosting	Weak learners (trees)	Sequential boosting with additional regularization and parallelization	Very fast, handles missing values, regularization reduces overfitting	More complex to tune than basic gradient boosting

Summary of Key Differences:

1. **Random Forest** = parallel trees, reduces variance, uses bagging.
2. **AdaBoost** = sequential trees, focuses on misclassified samples.
3. **Gradient Boosting** = sequential trees, fits residual errors using gradient descent.
4. **XGBoost** = optimized gradient boosting with speed, regularization, and scalability improvements.

Here’s a **simple visual diagram** showing how each algorithm builds its trees:

Random Forest (Bagging)

Tree 1 \

Tree 2 }--> Combine by averaging / voting

Tree 3 /

...

Trees are built independently on random samples with random features.

AdaBoost (Sequential Boosting)

Tree 1 --> Learn errors of previous --> Tree 2 --> Learn errors --> Tree 3 --> ...

Weights are adjusted so misclassified samples get more focus in next tree.

Gradient Boosting

Tree 1 --> Compute residuals --> Tree 2 fits residuals --> Tree 3 fits new residuals --> ...

Each tree corrects the mistakes (residuals) of the ensemble using gradient of loss.

XGBoost (Optimized Gradient Boosting)

Tree 1 --> Compute residuals --> Tree 2 fits residuals (with regularization & pruning) --> Tree 3 --> ...

Faster, handles missing values, uses regularization, and can run parallel computations.