**Hyper parameters: (Random Forest)**

N\_estimators (Increase): Reduces Vairance

**Advantages**:

* It won’t overfit
* It can be parallelized using different available cores
* One of benefits of Random forest which excites me most is, the power of handle large data set with higher dimensionality. It can handle thousands of input variables and identify most significant variables so it is considered as one of the dimensionality reduction methods. Further, the model outputs **Importance of variable,**which can be a very handy feature (on some random data set).Random forests are extremely flexible and have very high accuracy.
* They also do not require preparation of the input data. You do not have to scale the data.
* It also maintains accuracy even when a large proportion of the data are missing.
* It has methods for balancing errors in data sets where classes are imbalanced.

**Disadvantages**:

* No interpretability
* In short, with random forest, you can train a model with a relative small number of samples and get pretty good results. It will, however, quickly reach a point where more samples will not improve the accuracy.
* it fails when there are rare outcomes or rare predictors, as the algorithm is based on bootstrap sampling. This makes it non-ideal if you're working with rare personality traits, high segmented customer behavior, or rare variants in genomics research.
* It surely does a good job at classification but not as good as for regression problem as it does not give precise continuous nature predictions. In case of regression, it doesn’t predict beyond the range in the training data, and that they may over-fit data sets that are particularly noisy.

**Hyper parameters: (Gradient Boosting)**

N\_estimators (increase): High variance

Learning\_rate(v) (Increases): High variance

**Advantages**:

* We can choose any loss function
* Good for low latency applications as depth is very less
* Although it may seem GBDTs are better than random forests, GBDTs are prone to overfitting, however there are strategies to overcome same and build more generalized trees using a combination of parameters like learning rate (shrinkage) and depth of tree.  Generally the two parameters are kept on the lower side to allow for slow learning and better generalization.
* It performs the optimization in function space (rather than in parameter space) which makes the use of custom loss functions much easier.
* Boosting focuses step by step on difficult examples that gives a nice strategy to deal with unbalanced datasets by strengthening the impact of the positive class.

**Disadvantages**:

* GBDT training generally takes longer because of the fact that trees are built sequentially. However benchmark results have shown GBDT are better learners than Random Forests.
* It will generally overfit
* It cannot be parallelized
* GBMs are harder to tune than RF. There are typically three parameters: number of trees, depth of trees and learning rate, and the each tree built is generally shallow.