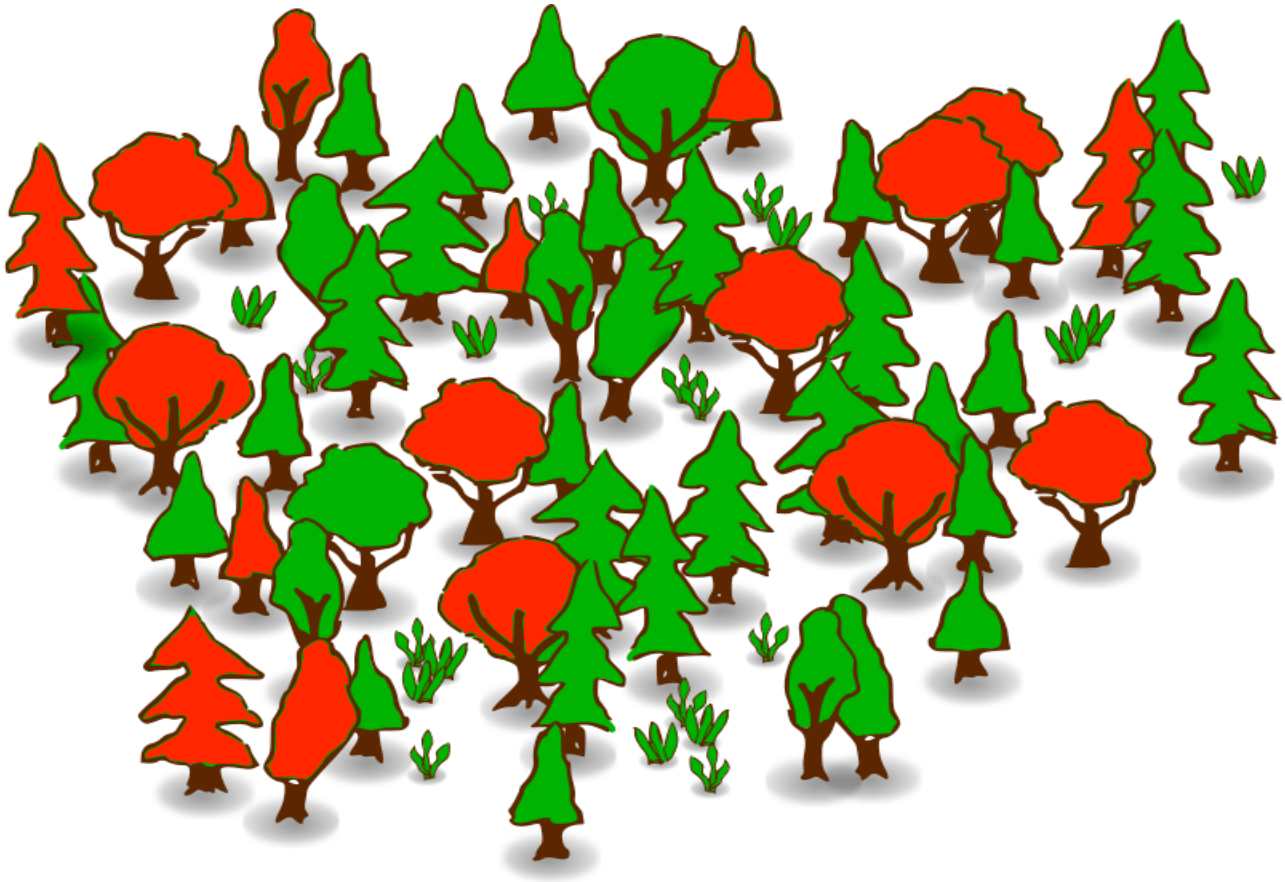
Ensemble Method: Random Forest



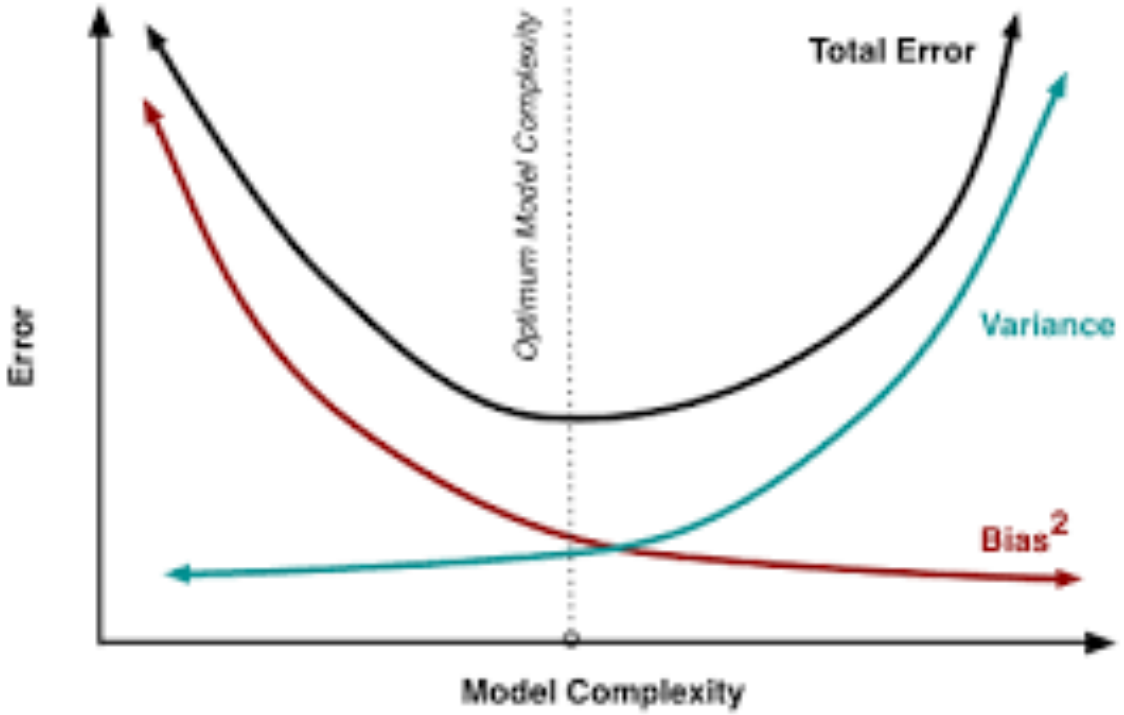
Problem with Decision Trees

A tree based model suffers from bias and variance problem and suffers from over fit problem.

If we make a small tree or simple model, it suffers from low variance and high bias, i.e., under fit

Decision tree are sensitive to the specific data on which they are trained. If the training data is changed the resulting tree can be quite different and in turn the predictions can be quite different

So there should be balance between these two types of errors. This is known as the trade-off balance of bias-variance errors.



Things to consider

Ensemble learning is one way to execute this trade off analysis.

The literary meaning of word “ensemble’ is group. Ensemble methods involve group of predictive models to achieve a better accuracy and model stability.

Ensemble methods impart supreme boost to tree base models

Some of the commonly used ensemble methods include: Bagging and Boosting

What is random forest?

Random forest is a flexible, easy to use machine learning algorithm that produces, even without hyper-parameter tuning, a great result most of the time. It is also one of the most used algorithms, because of its simplicity and diversity (it can be used for both classification and regression tasks).

Random Forest is a supervised learning algorithm. The “forest” it builds, is an ensemble of decision trees, usually trained with the “bagging” method. The general idea of bagging method is that a combination of learning models increases the overall result.

Put simply: random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.

What is random forest?

Random forests often also called **random decision forests** represent a Machine Learning task that can be used for classification and regression problems. They work by constructing a variable number of decision tree classifiers or regressors and the output is obtained by corroborating the output of the all the decision trees to settle for a single result. Because they work based on the simple concept of *wisdom of crowds,* random forests are very powerful machine learning tools, because they keep the simplicity of decision trees but employing the power of the ensemble.

What is random forest?

A random forest classifier is, as the name implies, a collection of decision trees classifiers that each do their best to offer the best output. Because we talk about classification and classes and there's no order relation between 2 or more classes, the final output of the random forest classifier is the **mode of the classes**.

This means the "winner" class is the one who appears most times in the list of outputs from all the decision trees used.

We all know Machine Learning models are never perfect and that sometimes they offer us incorrect data, especially when the data is sparse or we don't have lots of features we can look at.

But random forests are so far better than decision trees because they use the decisions from an ensemble of trees to figure out what is the final output. And while some decision trees may be wrong indeed, there's a high chance that, using carefully groomed data, most of the trees in the forest will be able to offer a better output.

What is random forest?

In real life, an ideal democracy is powerful because the power of all people who vote is greater than any individual's power and that helps pushing a democracy forward towards the benefit of everybody. But what happens if suddenly, an individual begins to gain more influence amongst other peers? This individual will begin influencing others and then they will push the direction of the democracy to something which might not be the best decision.

What is random forest?

That's also the case with decision trees. We need to make sure that decision trees don't get to influence each other. In Statistics language, that means finding a way to enforce the fact that *decision trees are not correlated,* and this is achieved by employing two methods.

The first method is based on choosing a random subset of features for each individual tree in the forest.
Let's say we have a dataset with *N* available features. A normal decision trees looks at all the *N* features, but with random forests, each individual tree will be able to analyze only a subset of $M<N$ features, with the *M* features being randomly chosen from the whole set of features. Because the trees are forced to look at different data points, it is natural that the results they obtain will be less correlated between each other.

What is random forest?

The second method is called *bootstrap aggregation* and it means choosing a random subset of the whole data entries in the dataset to train our decision trees. The sampling is done with replacement, which means the number of elements in the subset is actually the same as the whole set, so that all decision trees have a fair chance to get some proper training.

For example, if the dataset has the following rows for a specific column: [a, b, c, d, e], a decision tree might get to be trained on a subset that looks like this: [a, a, b, c, c]. You notice the set and the subset have the same number of elements, but the elements are chosen randomly from the whole possible combinations.

What is random forest?

Taking all of these into consideration, we can sketch the simple algorithm for building a random forest of decision trees. Let's say we want to build a random forest with *T* trees with a dataset of *M* rows and *N* features.

For each one of the *T* decision trees:

Select a random subset of features from the whole set of features

Select a random subset of rows from the whole dataset

Build the current tree with the current features and rows subsets

What is random forest?

If you can comprehend a single decision tree, the idea of *bagging,* and random subsets of features, then you have a pretty good understanding of how a random forest works:

The random forest combines hundreds or thousands of decision trees, trains each one on a slightly different set of the observations, splitting nodes in each tree considering a limited number of the features. The final predictions of the random forest are made by averaging the predictions of each individual tree.

To understand why a random forest is better than a single decision tree imagine the following scenario: you have to decide whether Tesla stock will go up and you have access to a dozen analysts who have no prior knowledge about the company. Each analyst has low bias because they don't come in with any assumptions, and is allowed to learn from a dataset of news reports.

What is random forest?

This might seem like an ideal situation, but the problem is that the reports are likely to contain noise in addition to real signals. Because the analysts are basing their predictions entirely on the data — they have high flexibility — they can be swayed by irrelevant information. The analysts might come up with differing predictions from the same dataset. Moreover, each individual analyst has high variance and would come up with drastically different predictions if given a *different* training set of reports.

The solution is to not rely on any one individual, but pool the votes of each analyst. Furthermore, like in a random forest, allow each analyst access to only a section of the reports and hope the effects of the noisy information will be cancelled out by the sampling. In real life, we rely on multiple sources (never trust a solitary Amazon review), and therefore, not only is a decision tree intuitive, but so is the idea of combining them in a random forest.

What is random forest?

The power of the algorithm stands again in the number of individual trees, because you know the mean value of a series of number is heavily influenced in the direction where most of the values in the series are located.
More high values pull the mean higher, while low values push the mean lower and equally distributed values keep the mean in the middle.

Real-Life Analogy

Andrew wants to decide where to go during one-year vacation, so he asks the people who know him best for suggestions. The first friend he seeks out asks him about the likes and dislikes of his past travels. Based on the answers, he will give Andrew some advice.

This is a typical decision tree algorithm approach. Andrew's friend created rules to guide his decision about what he should recommend, by using Andrew's answers.

Afterwards, Andrew starts asking more and more of his friends to advise him and they again ask him different questions they can use to derive some recommendations from. Finally, Andrew chooses the places that where recommend the most to him, which is the typical random forest algorithm approach.

Feature Importance

Another great quality of the random forest algorithm is that it is very easy to measure the relative importance of each feature on the prediction. Sklearn provides a great tool for this that measures a feature's importance by looking at how much the tree nodes that use that feature reduce impurity across all trees in the forest. It computes this score automatically for each feature after training and scales the results so the sum of all importance is equal to one.

If you don’t know how a decision tree works or what a leaf or node is, here is a good description from Wikipedia: '"In a decision tree each internal node represents a 'test' on an attribute (e.g. whether a coin flip comes up heads or tails), each branch represents the outcome of the test, and each leaf node represents a class label (decision taken after computing all attributes). A node that has no children is a leaf.'"

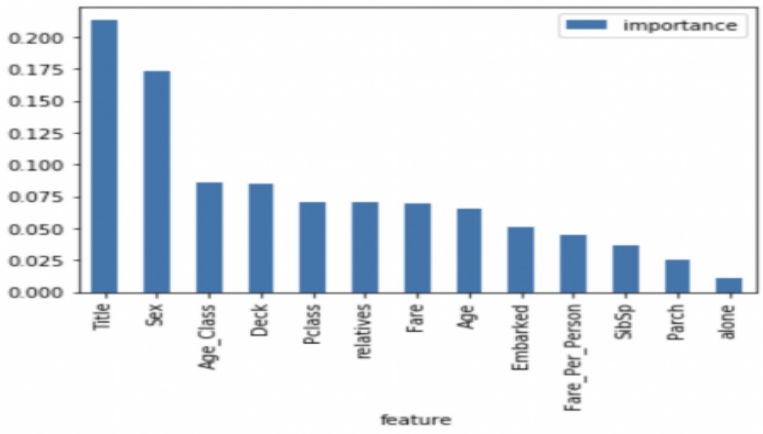
Feature Importance

By looking at the feature importance you can decide which features to possibly drop because they don't contribute enough (or sometimes nothing at all) to the prediction process. This is important because a general rule in machine learning is that the more features you have the more likely your model will suffer from overfitting and vice versa.

Feature Importance

a table and visualization showing the importance of 13 features, which I used during a supervised classification project with the famous Titanic dataset on kaggle

feature	importance
Title	0.213
Sex	0.173
Age_Class	0.086
Deck	0.085
Pclass	0.071
relatives	0.070
Fare	0.069
Age	0.065
Embarked	0.051
Fare_Per_Person	0.045
SibSp	0.037
Parch	0.025
alone	0.011



Difference between Decision Trees and Random Forests

While random forest is a collection of decision trees, there are some differences.

If you input a training dataset with features and labels into a decision tree, it will formulate some set of rules, which will be used to make the predictions.

For example, to predict whether a person will click on an online advertisement, you might collect the ads the person clicked on in the past and some features that describe his/her decision. If you put the features and labels into a decision tree, it will generate some rules that help predict whether the advertisement will be clicked or not. In comparison, the random forest algorithm randomly selects observations and features to build several decision trees and then averages the results.

Difference between Decision Trees and Random Forests

Another difference is "deep" decision trees might suffer from overfitting. Most of the time, random forest prevents this by creating random subsets of the features and building smaller trees using those subsets. Afterwards, it combines the subtrees. It's important to note this doesn’t work every time and it also makes the computation slower, depending on how many trees the random forest builds.

Choice between Decision Trees and Random Forests

Usually random forests will perform better than simple decision trees. As with all Machine Learning problems, the decision to choose one model over another is a process of experimentation and depends on the data you have and on the objectives you have set for your model. Decision Trees might be more suitable when you need a simple model, that computes on small or simple datasets. If the data is simple in terms of features or entries, then a decision tree will most probably perform very well.

Difference between Decision Trees and Random Forests

A decision tree might also be more appropriate when you want a model that's easy to understand and visualize. Also, if you want your model to be explainable, decision trees will work very well for you. That means your results will be somehow more predictable and you can easy explain to other interested parties how it works.

You can also choose decision trees when you don't worry or you don't want to worry about certain features in your dataset being correlated. If the features are correlated, a decision tree will perform way more poorly than a random forest.

For all the other cases, you can certainly go with random forests. As a general rule of thumb here, we might say that you should try decision trees first and random forests after and see which works best for you.

.

Tuning Parameters

Random forests are considered as a black box models which takes in input and gives out predictions, without worrying too much about what is going on the back end. The black box models have few parameters to play with.

Each of these parameters have some effect on either performance of the model or the resource - speed. Tuning the parameter results in the optimized value of the parameters.

While model parameters are learned during training – such as the slope and intercept in a linear regression- hyperparameters must be set by the developer before training.

.

Parameters to tune

Parameters in random forest are either to increase the predictive power of the model or make it easier to train the model.

Scikit-Learn implements a set of default hyper-parameters for all models, but these are not guaranteed to be optimal for a problem. The best hyper-parameters are usually impossible to determine ahead of time, and tuning a model is where machine learning turns from a science into trial and error based engineering.

Tuning these parameters relies more on experimental results than theory, and thus best method to determine the optimal settings is to try many different combinations evaluate the performance of each model.

Important Hyperparameters

The hyperparameters in random forest are either used to increase the predictive power of the model or to make the model faster. Let's look at the hyperparameters of sklearns built-in random forest function.

1. Increasing the predictive power

Firstly, there is the **n_estimators** hyperparameter, which is just the number of trees the algorithm builds before taking the maximum voting or taking the averages of predictions. In general, a higher number of trees increases the performance and makes the predictions more stable, but it also slows down the computation.

Another important hyperparameter is **max_features,** which is the maximum number of features random forest considers to split a node. Sklearn provides several options, all described in the documentation.

Important Hyperparameters

The last important hyperparameter is **min_sample_leaf.** This determines the minimum number of leafs required to split an internal node.

2. Increasing the model's speed

The **n_jobs** hyperparameter tells the engine how many processors it is allowed to use. If it has a value of one, it can only use one processor. A value of “-1” means that there is no limit.

The **random_state** hyperparameter makes the model’s output replicable. The model will always produce the same results when it has a definite value of random_state and if it has been given the same hyperparameters and the same training data.

Important Hyperparameters

Lastly, there is the **oob_score** (also called oob sampling), which is a random forest cross-validation method. In this sampling, about one-third of the data is not used to train the model and can be used to evaluate its performance. These samples are called the out-of-bag samples. It's very similar to the leave-one-out-cross-validation method, but almost no additional computational burden goes along with it.

Bootstrap

Bootstrap is a statistical resample method

When you can’t afford to get more sample (medical data)

Used in statistic when you want to estimate a statistic of a random sample (a statistic mean, variance, mode etc.)

Drawing randomly with replacement set of available data.

Bootstrap

Ideally we would like to take more samples, but n is all what we have got

What to do?

What we do is sample our data set with replacement.

We repeat the above step for a large number of times, say P times. Once done we have P number of bootstrap random samples

We then take the statistic of each bootstrap random sample and average it.

Bootstrap

We are getting the range for that mean of the population.

Important: Bootstrapping does not give “better” estimates. It gives a better range for the estimate

Bootstrap

Original data (random sample)

{2,4, 5,6, 6 }, mean = 4.6

Bootstrap samples:

{2,5,5,6,6}, mean = 4.8

{4,5,6,6,6}, mean = 5.4

{2,2,4,5,5} , mean = 3.6

{2,2,2,4,6}, mean = 3.2

{4,6,6,6,6} = mean = 5.6

mean = 4.52

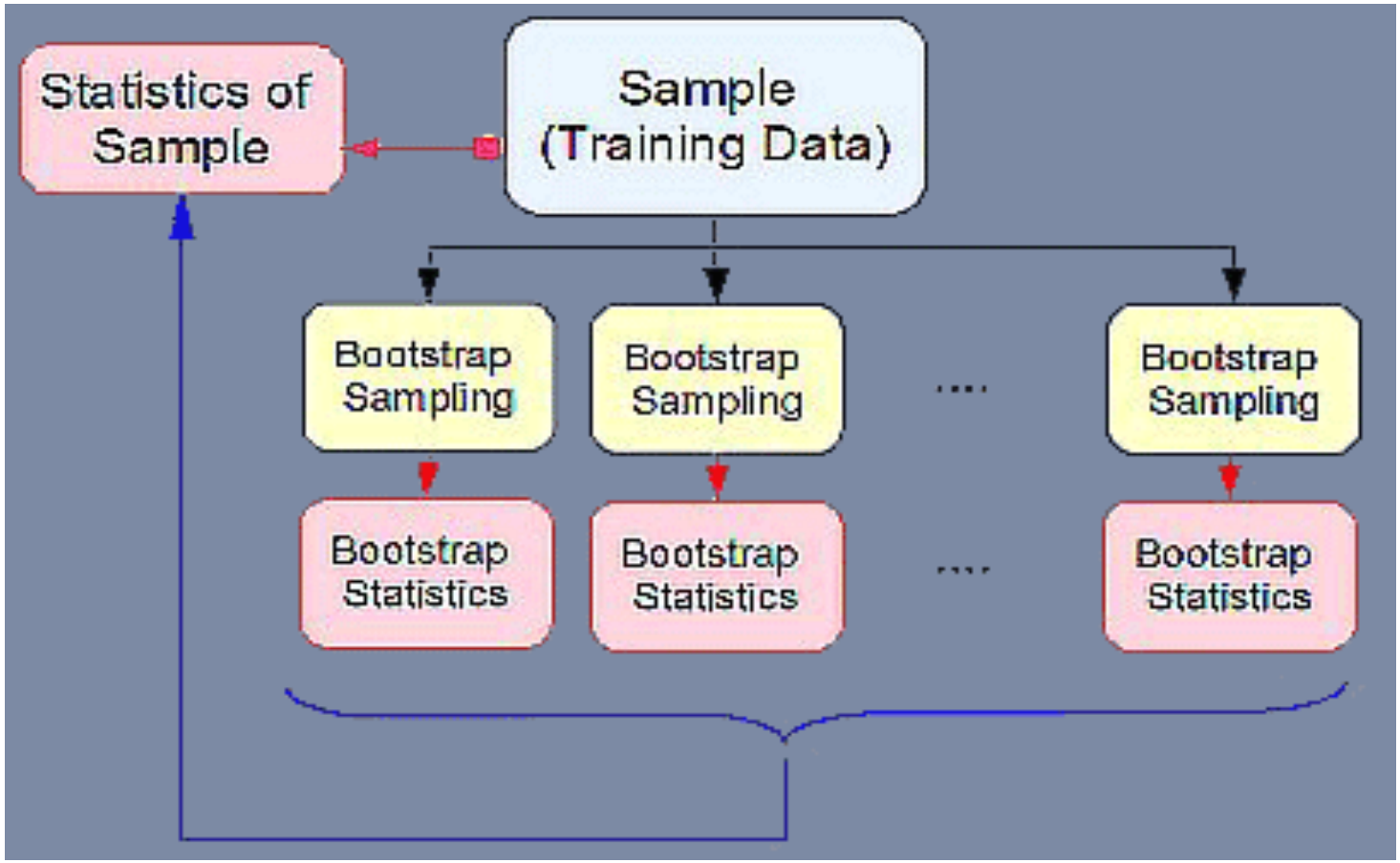
Bootstrap

- It is especially useful when the sample size that we are working with is small. These kind of techniques assume nothing about the distribution of our data.

- It comes in handy when there is doubt that the useful distribution assumptions and asymptotic results are valid and accurate.

- Bootstrapping is a nonparametric method which lets compute estimated standard errors, confidence intervals and hypothesis testing.

Bootstrap


Statistics of Sample
Sample (Training Data)
Bootstrap Sampling
Bootstrap Sampling
....
Bootstrap Sampling
Bootstrap Statistics
Bootstrap Statistics
....
Bootstrap Statistics

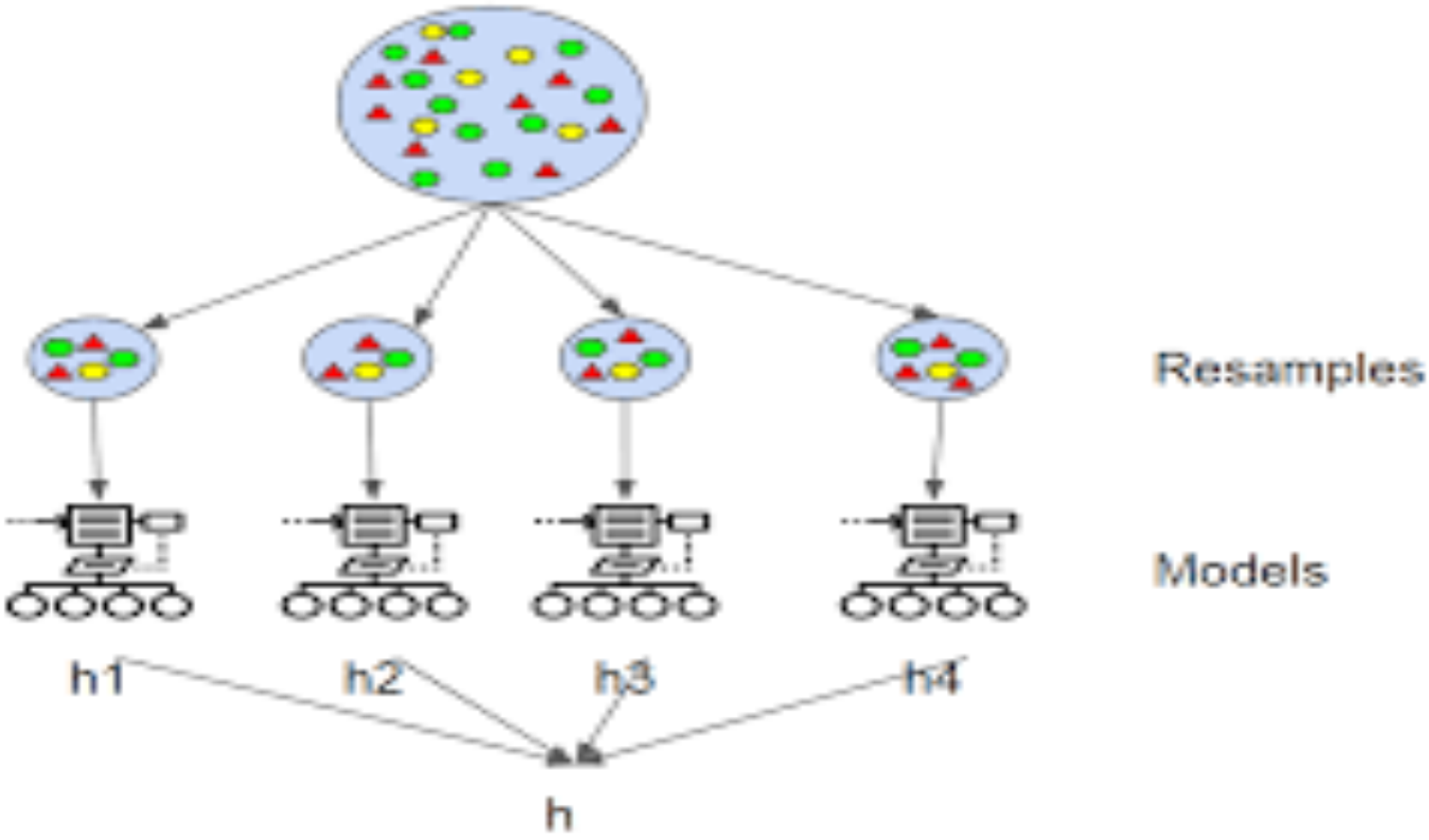
Bootstrap

- What are the assumption of Bootstrap?
- Bootstrap method has assumptions
  - Sample is a valid representative of the population
  - Bootstrap method will take sampling with replacement from the sample. Each sub sampling is independent and identical distribution (i.i.d). In other word, it assumes that the sub samples come from the same distribution of the population, but each sample is drawn independently from the other samples.

- Generally bootstrapping follows the same basic steps:
  1. Resample a given data set a specified number of times
  2. Calculate a specific statistics from each sample
  3. Find the standard deviation of the distribution of that statistic

Bagging (Bootstrap Aggregation)

Bagging is a technique used to reduce the variance by combining the result of multiple classifiers modeled on different sub-samples of the same data set.



Bagging

The following steps are followed:

1. **Create multiple Data sets:**
   - Sampling is done with replacement on the original data and new data sets are formed
   - The new data sets can have a fraction of the columns as well as rows, which are generally hyper-parameters in a bagging model

2. **Create multiple Classifiers:**
   - Classifiers are built on each data set
   - Generally the same classifier is built on each data set and predictions are made.

Bagging

3. **Combine Classifiers:**
    - The prediction of all the classifiers are combined using mean, median or mode value depending on the problem at hand

There are various implementations of bagging models, Random Forest is one of them.

How does it work?

- Multiple trees are grown during modeling in Random forest
- To classify a new object based on attributes, each tree gives a classification and votes for that class.
- The forest chooses the classification having the most votes and in case of regression, it takes the average of outputs by different trees

Random Forest

Working : Each tree is grown as follows:

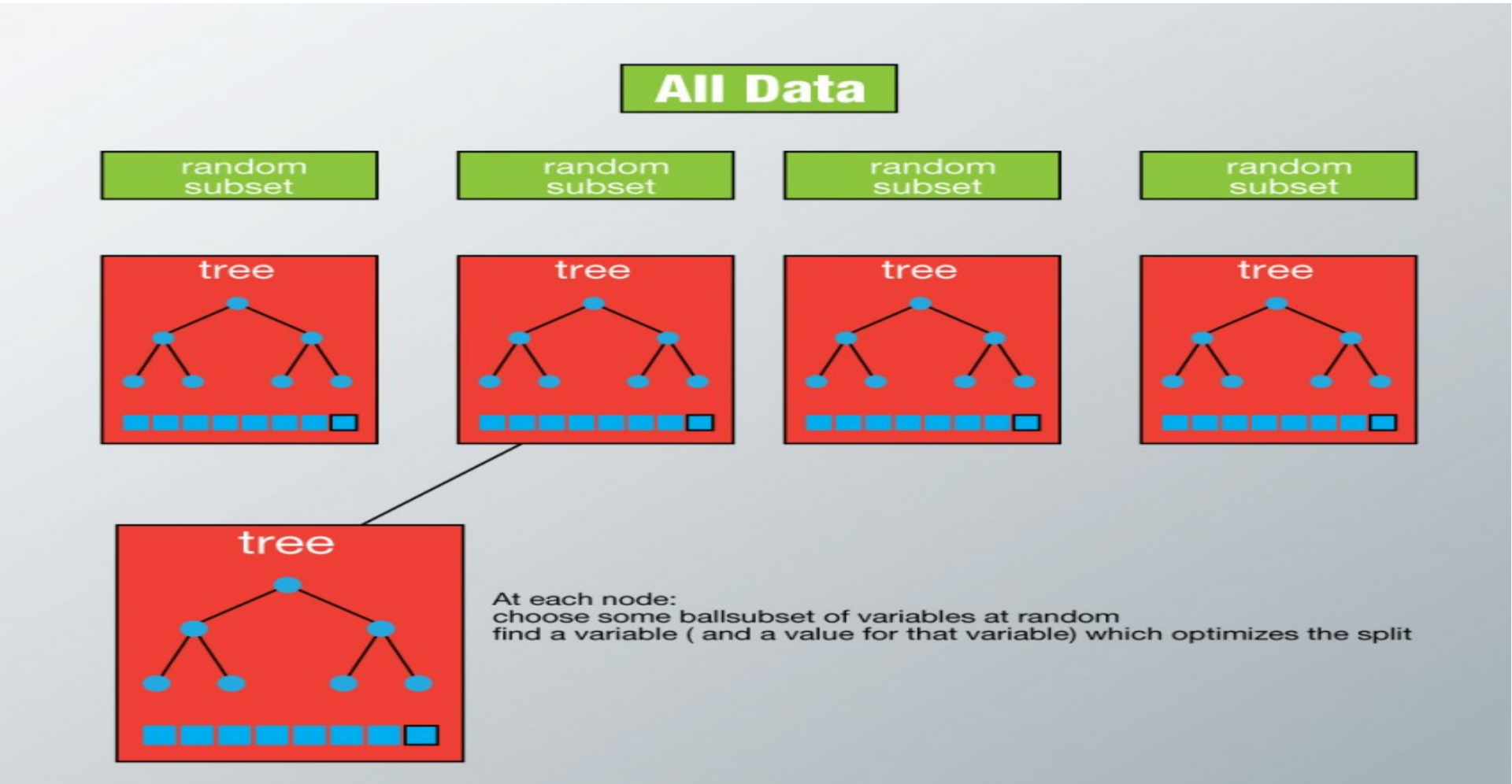
1. Assume number of cases in the training set is N. Then, sample of these N cases is taken at random but with replacement. This sample will be the training set for growing the tree.

2. If there are M input variables, a number m<M is specified such that at each node, m variables are selected at random out of the M. The best split of these m is used to split the node. The value of m is held constant while we grow the forest.

3. Each tree is grown to the largest extent possible

Random Forest

4. Predict new data by aggregating the predictions of the n trees (i.e., majority votes for classification, average for regression)



How does Random Forest work

Each tree is grown as follows:

1. **Random Record Selection**: Each tree is trained on roughly 2/3rd of the total training data. Cases are drawn at random with replacement from the original data.

1. **Random Variable Selection**: Random number of predictors (say, m) are selected at random out of all the predictor variables and the best split on these m is used to split the node.

- The number of features that can be searched at each split point (m) must be specified as a parameter to the algorithm.
  - For classification a good default is: m = sqrt(p)
  - For regression a good default is: m = p/3

How does Random Forest work

3. For each tree, using the leftover (1/3rd) data, calculate the misclassification rate, out of bag (OOB) error rate. Aggregate error from all trees to determine overall OOB error rate for the classification.

4. Each tree gives a classification on leftover data (OOB). The forest chooses the classification having the most votes over all the trees in the forest.

Estimated Performance

- For each bootstrap sample taken from the training data, there will be samples left behind that were not included. These samples are called Out-of-Bag samples or OOB

- The performance of each model on its left out samples when averaged can provide an estimated accuracy of the bagged models. This estimated performance is often called the OOB estimate of performance

- These performance measures are reliable test error estimate and correlate well with cross validation estimates.

Advantages

- Can solve both type of problems, i.e., classification and regression
- Can handle large data set with higher dimensionality. It can handle thousands of input variables
- Can identify most significant variables so it is considered as one of the dimensionality reduction methods

Advantages contd.

- Can estimate missing data and maintains accuracy when a large proportion of the data are missing.
- It has methods for balancing errors in data sets where data in imbalanced
- Can be used for clustering, data views and outlier detection.
- Very little of no preprocessing is involved.

Disadvantages

•Random forest is not as good as for regression problem as it does not give precise continuous nature predictions. In case of regression, it does not predict beyond the range in the training data, and that they may over-fit data sets that are particularly noisy.

•Random forest feels like a black box approach. You may have very little control on what the model does. What you can do best is try different parameters and random seeds.

•It is biased towards the categorical variable having multiple levels (categories). It is because feature selection based on impurity reduction is biased towards preferring variables with more categories so variable selection (importance) is not accurate for this type of data.

Random forests applications and use cases

All of the previous sections bring us to a simple conclusions: a random forest is like a decision tree, but on steroids. So all the uses cases that apply to a decision tree will most definitely also apply to a random forest. That being said:

- You can use random forests both for classification and regression tasks.
- Use random forests when you have tried a decision tree already and you've come to the conclusion that you need higher accuracy and computational costs in terms of money, data or time are not a problem
- Use random forests when you have tried a decision tree already and after testing and validation you observe your decision tree is overfitting
- Use random forests if your dataset has too many features for a decision tree to handle

Random Forest Machine Learning Algorithm

Applications of Random Forest Machine Learning Algorithms

- Random Forest algorithms are used by banks to predict if a loan applicant is a likely high risk.
- They are used in the automobile industry to predict the failure or breakdown of a mechanical part.
- These algorithms are used in the healthcare industry to predict if a patient is likely to develop a chronic disease or not.
- Recently, the algorithm has also made way into predicting patterns in speech recognition software and classifying images and texts.

Random Forest Machine Learning Algorithm

https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm