Random Forest

- *Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.*

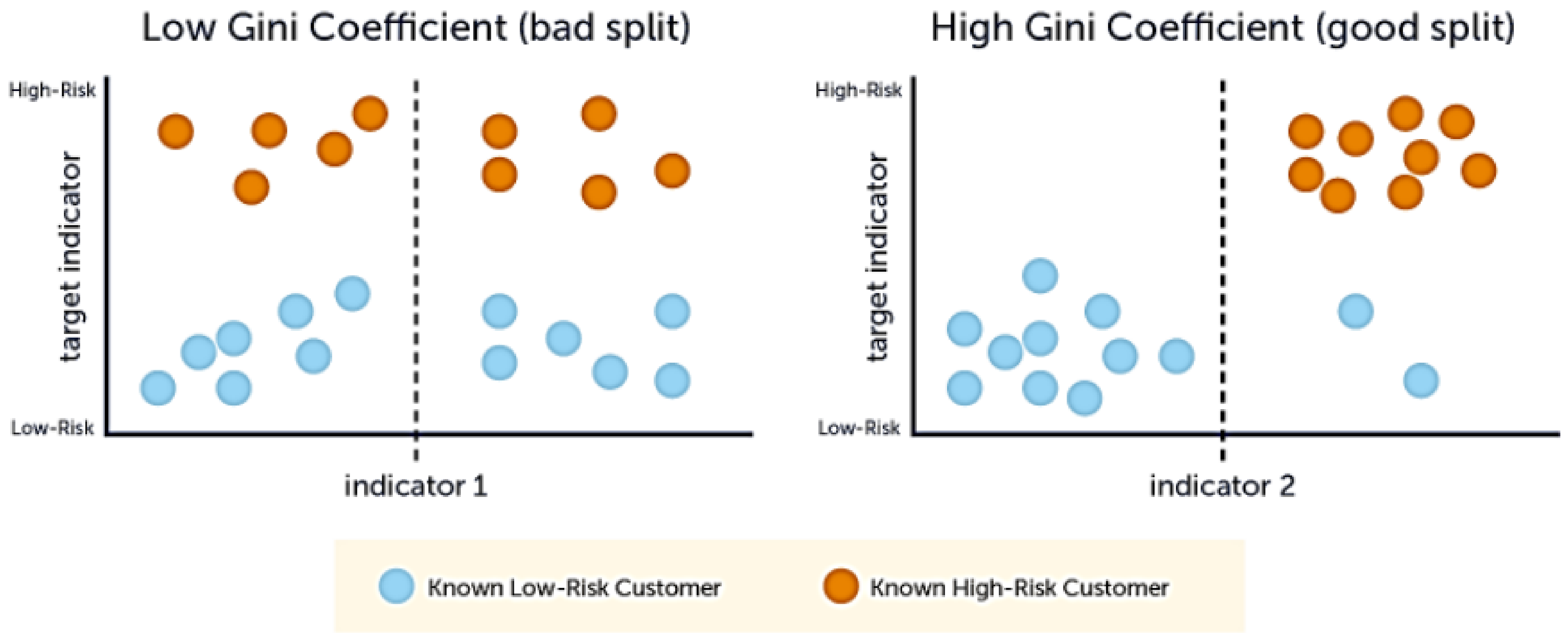
How does it split?

The most decisive factor for the efficiency of a decision tree is the efficiency of its splitting process.

We split at each node in such a way that the resulting **purity** is maximum.

Well, purity just refers to how well we can segregate the classes and increase our knowledge by the split performed.

An image is worth a thousand words.



Limitations to Decision Trees

Decision trees tend to have high variance when they utilize different training and test sets of the same data, since they tend to overfit on training data.

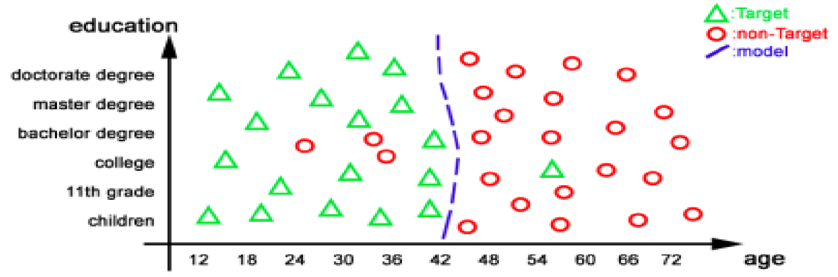
This leads to poor performance on unseen data. Unfortunately, this limits the usage of decision trees in predictive modeling.

However, using ensemble methods, we can create models that utilize underlying decision trees as a foundation for producing powerful results.

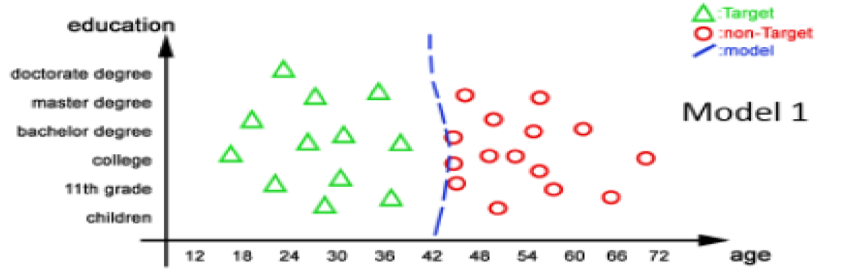
Overfitting

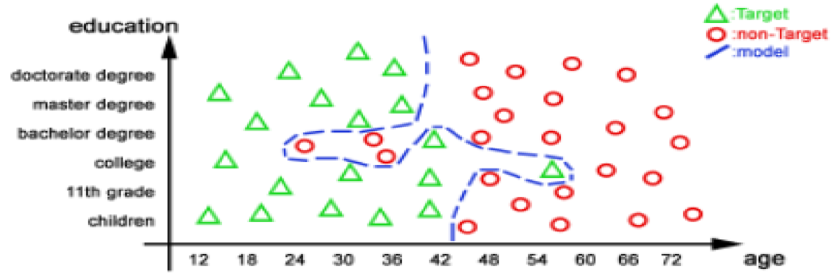
- A model overfits the training data when it is very accurate with that data, and may not do so well with new test data

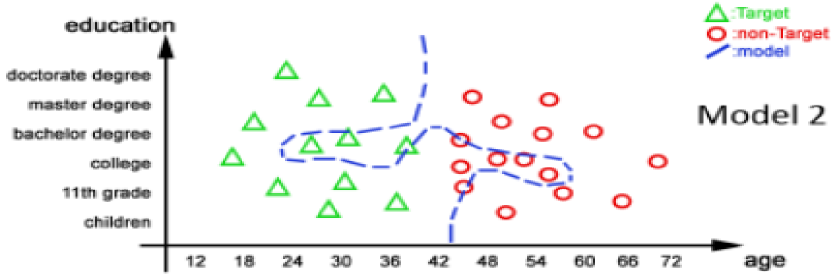
Training Data

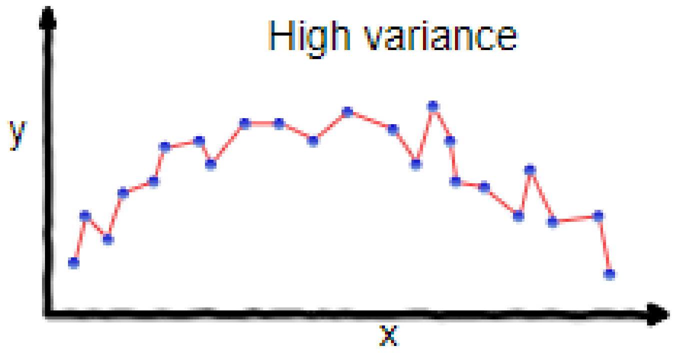


Test Data

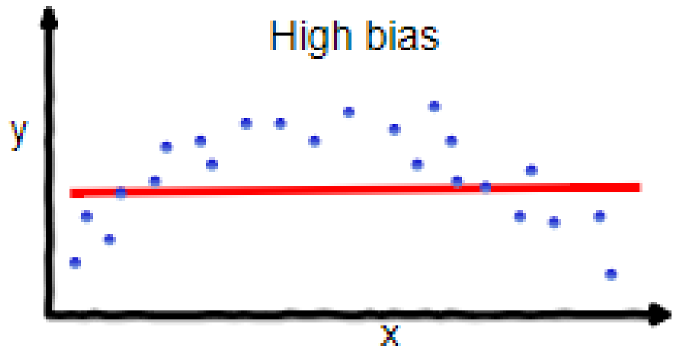




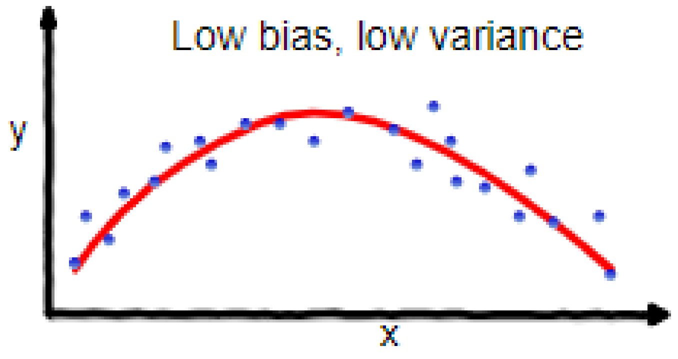




overfitting



underfitting

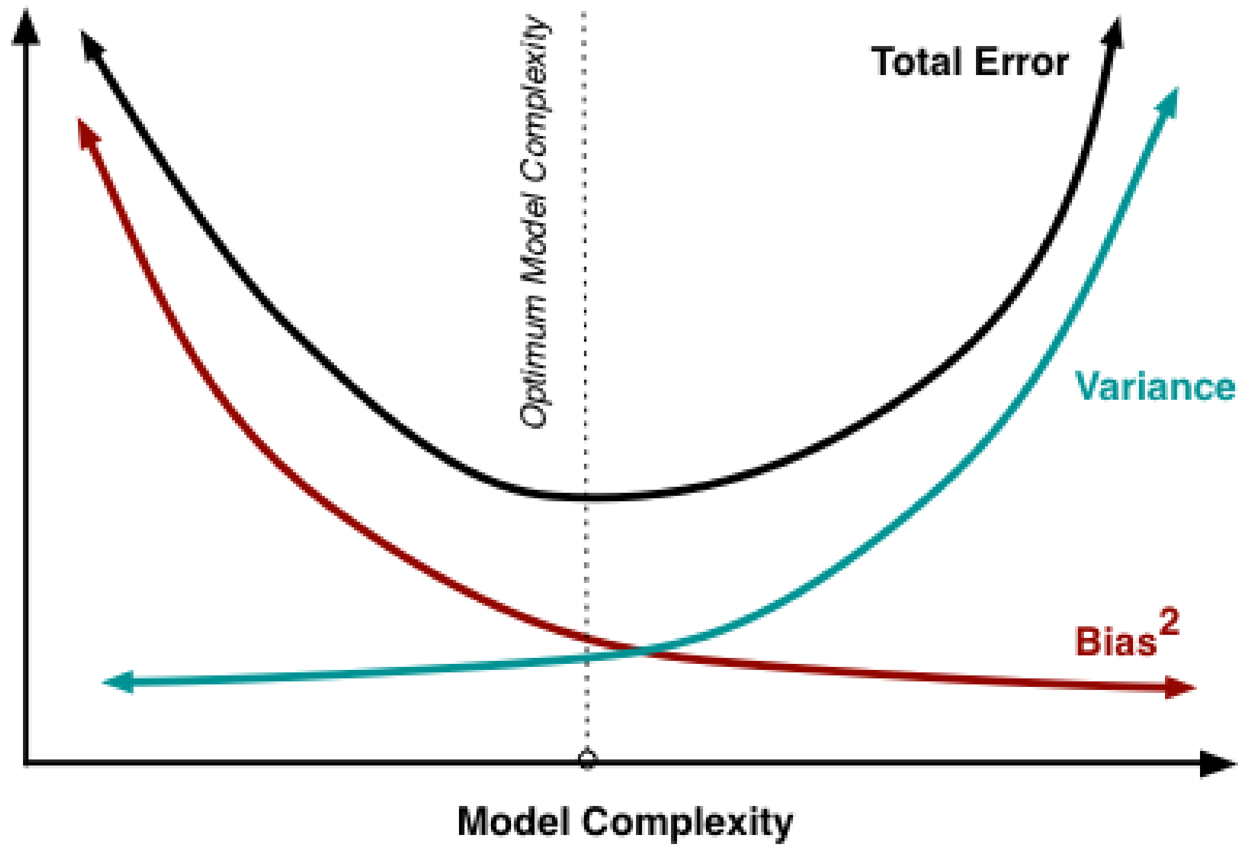


Good balance

Bias/Variance Tradeoff

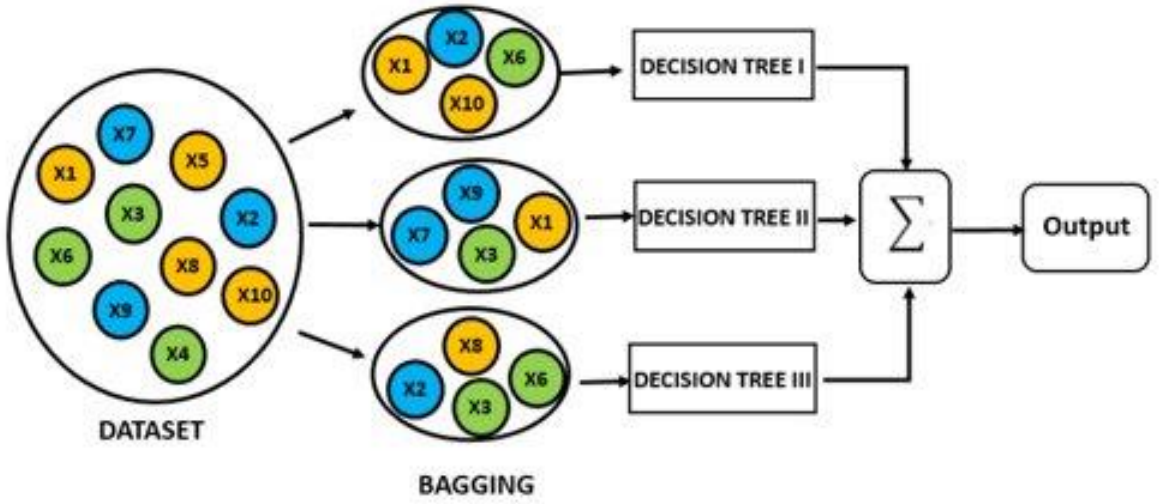
- Ensemble methods that minimize variance
  - Bagging
  - Random Forests

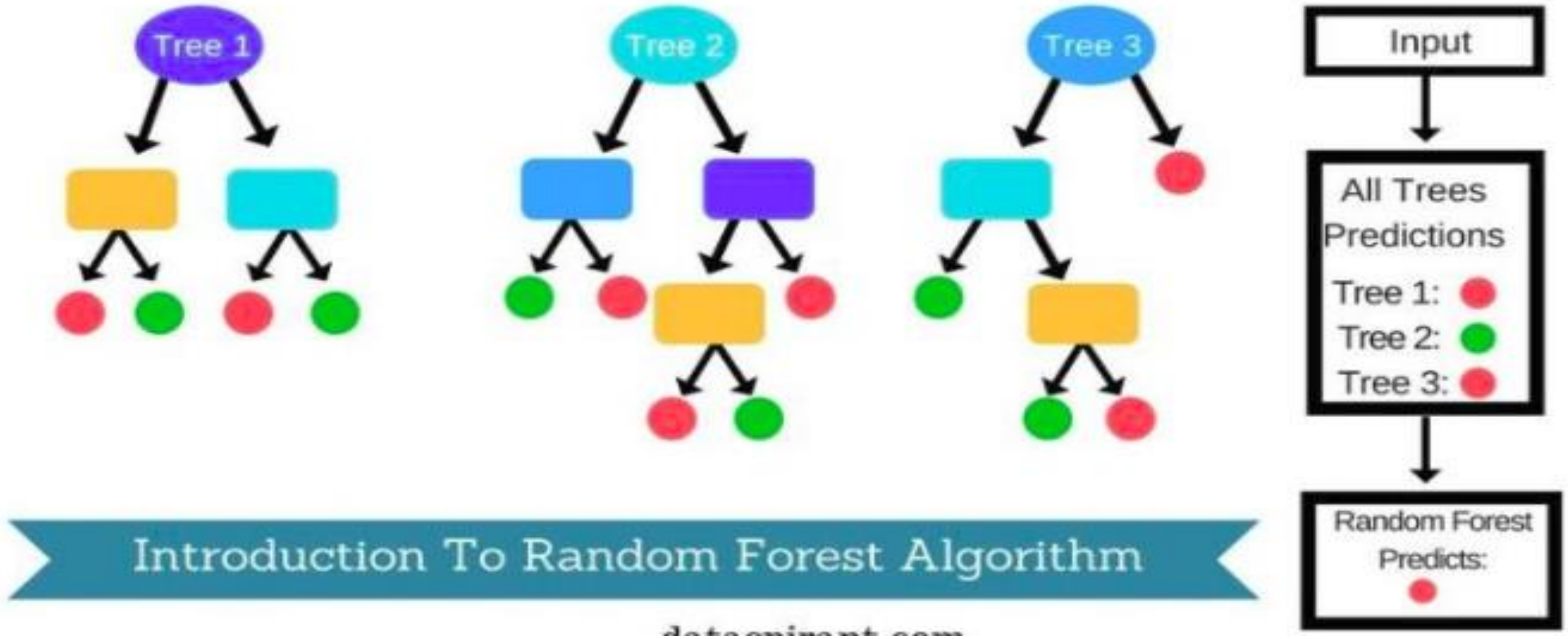
- Ensemble methods that minimize bias
  - Functional Gradient Descent
  - Boosting
  - Ensemble Selection



Ensemble Method:

- Ensembles are a divide-and-conquer approach used to improve performance.
- The main principle behind ensemble methods is that a group of “weak learners” can come together to form a “strong learner”.
- The figure below (taken from here) provides an example. Each classifier, individually, is a “weak learner,” while all the classifiers taken together are a “strong learner”.





Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.

Bootstrap Aggregating Trees

Through a process known as bootstrap aggregating (or bagging), it’s possible to create an ensemble (forest) of trees where multiple training sets are generated with replacement, meaning data instances — or in the case of this tutorial, patients — can be repeated. Once the training sets are created, a CART model can be trained on each subsample.

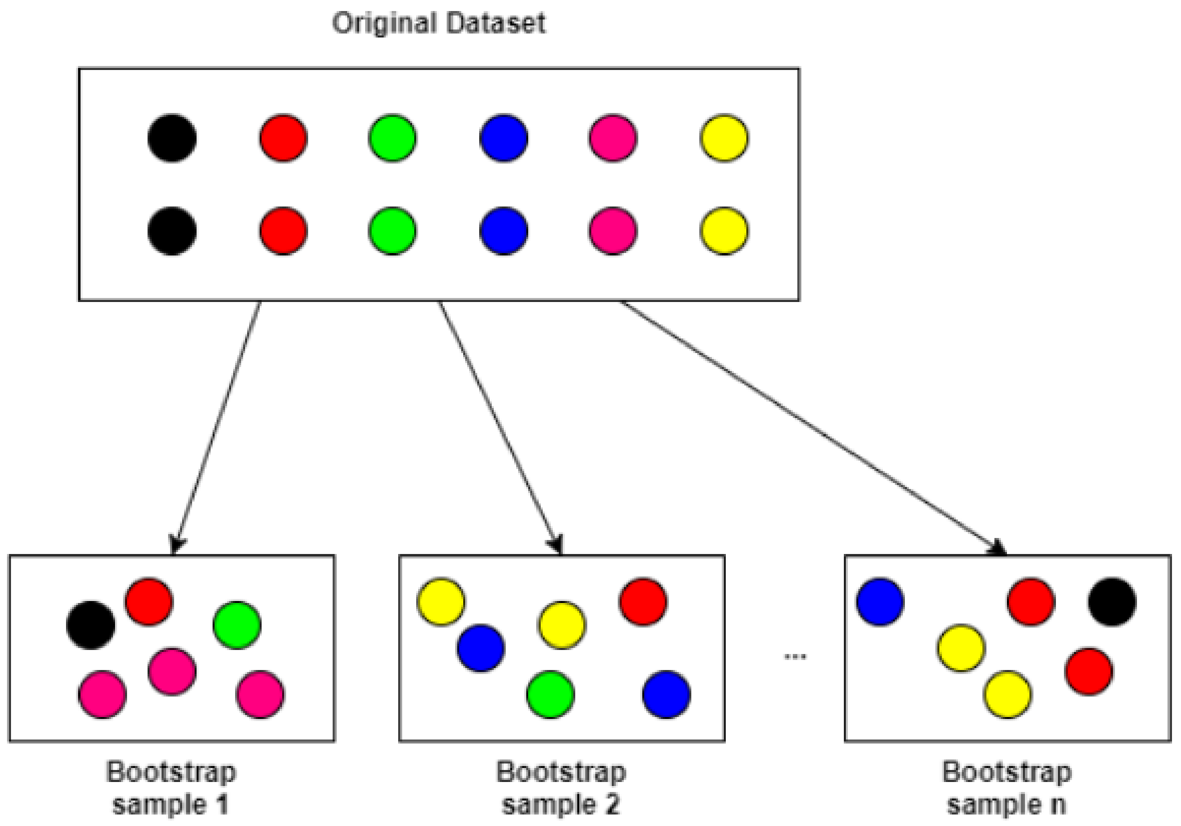
This approach helps reduce variance by averaging the ensemble's results, creating a majority-votes model.

Another important feature of bagging trees is that the resulting model uses the entire feature space when considering node splits.

Bagging trees allow the trees to grow without pruning, reducing the tree-depth sizes and resulting in high variance but lower bias, which can help improve predictive power.

However, a downside to this process is that the utilization of the entire feature space creates a risk of correlation between trees, increasing bias in the model.

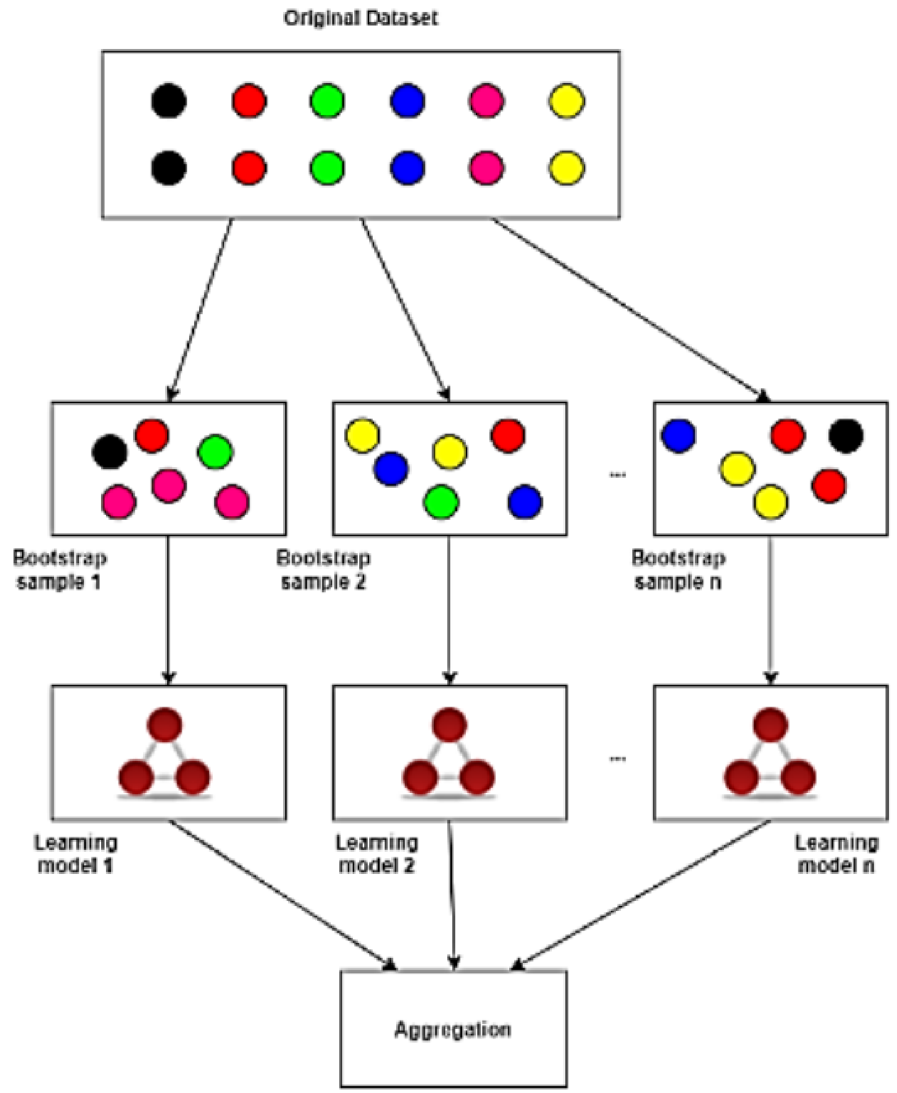
Bootstrap Aggregation



Subsetting for bootstrap aggregation.

Limitations to Bagging Tree

- The main limitation of bagging trees is that it uses the entire feature space when creating splits in the trees.
- If some variables within the feature space are indicative of certain predictions, you run the risk of having a forest of correlated trees, thereby increasing bias and reducing variance.
- However, a simple tweak of the bagging trees methodology can prove advantageous to the model’s predictive power.

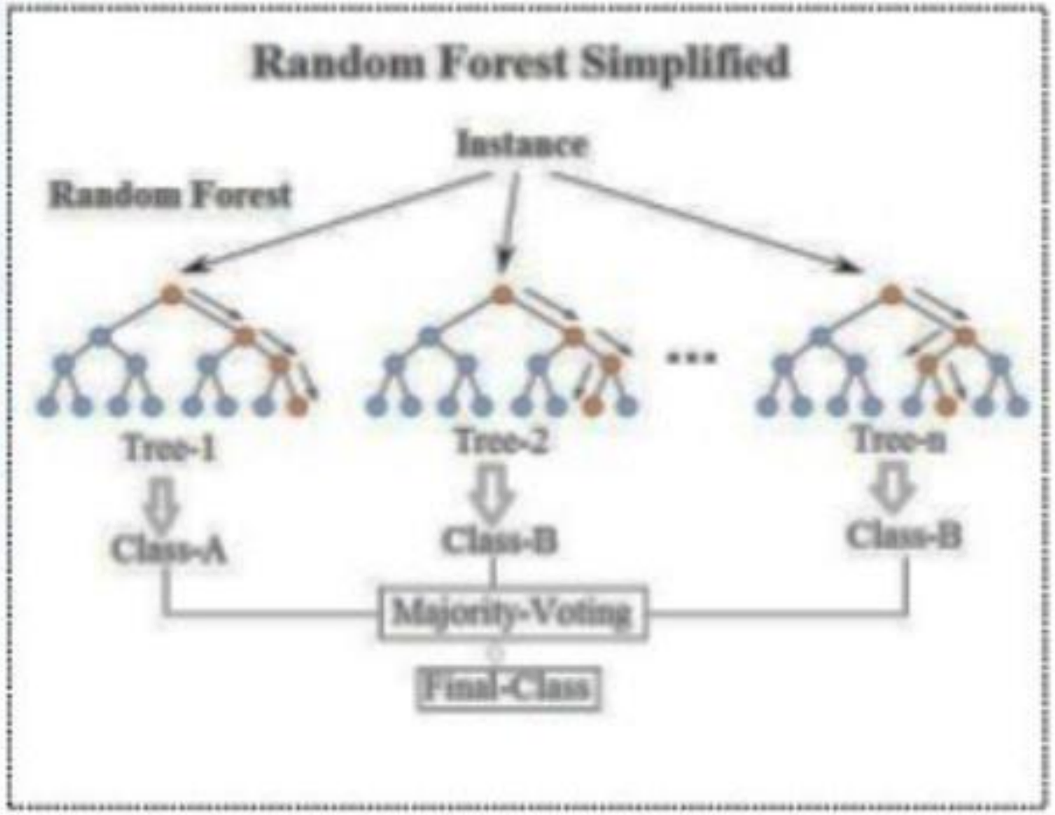


Bootstrapping and learning ensembles.

What Is Random Forest?

edureka!

- Random Forest - a versatile algorithm capable of performing both
  i) Regression
  ii) Classification
- It is a type of ensemble learning method
- Commonly used predictive modelling and machine learning technique

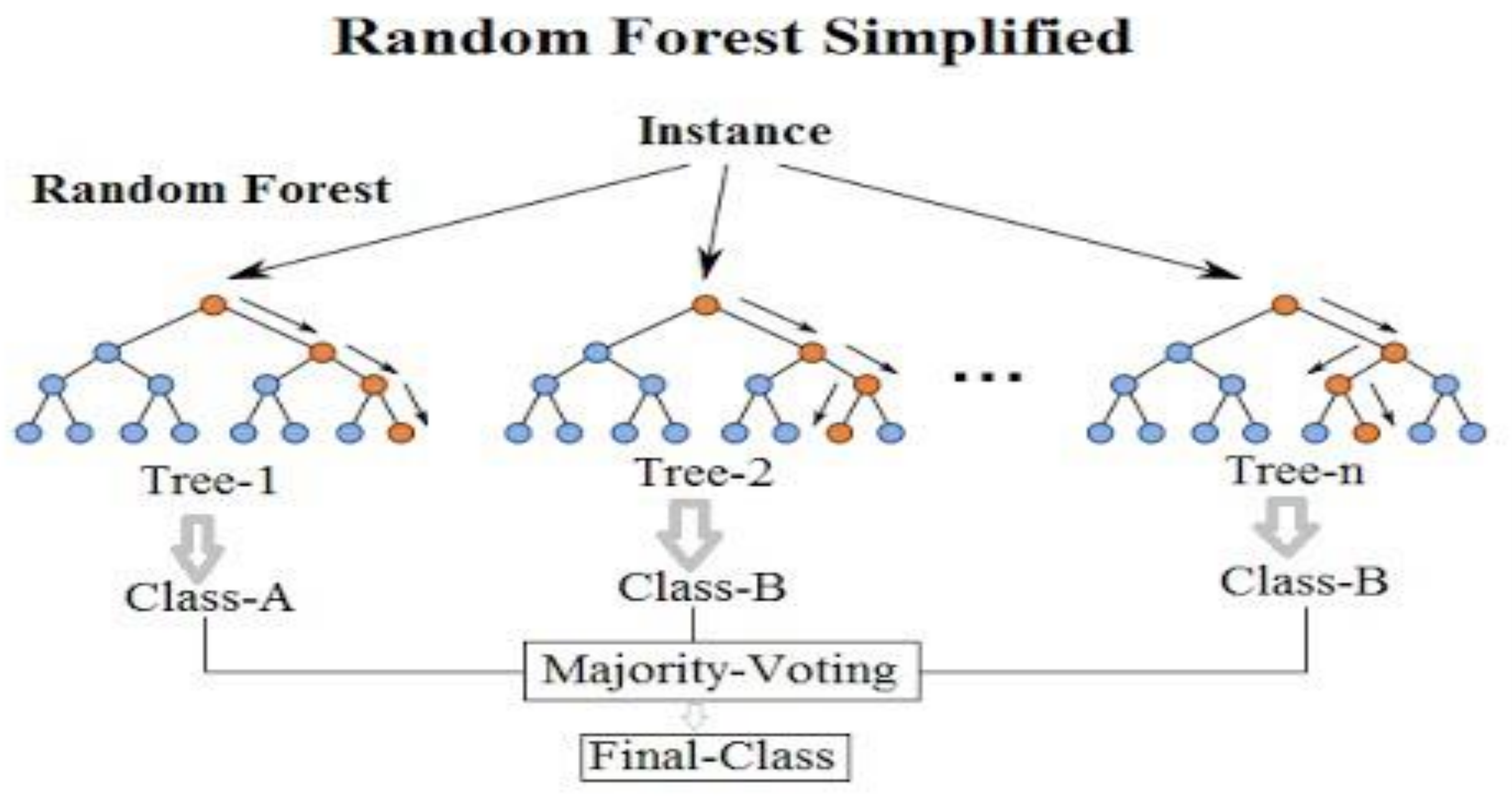


Random Forest

- Machine learning ensemble algorithm
- -- Combining multiple predictors
- Based on tree model
- For both regression and classification
- Automatic variable selection
- Handles missing values
- Robust, improving model stability and accuracy

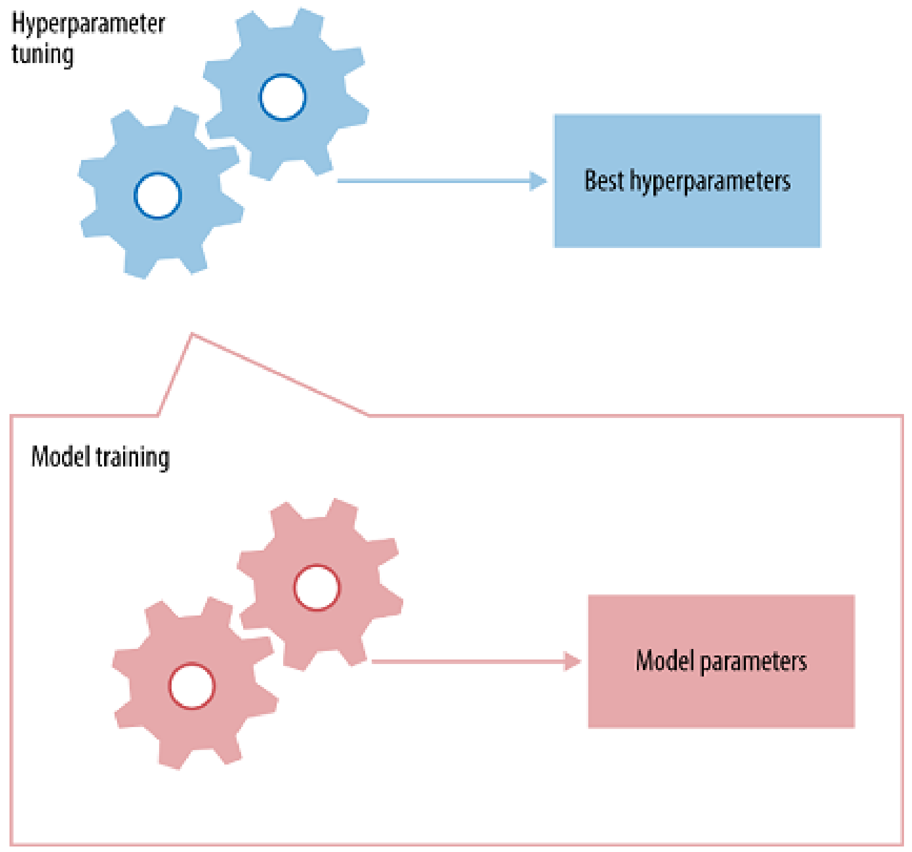
- State the question and determine required data
- Acquire the data in an accessible format
- Identify and correct missing data points/anomalies as required
- Prepare the data for the machine learning model
- Establish a baseline model that you aim to exceed
- Train the model on the training data
- Make predictions on the test data
- Compare predictions to the known test set targets and calculate performance metrics
- If performance is not satisfactory, adjust the model, acquire more data, or try a different modeling technique
- Interpret model and report results visually and numerically

- Basic exploratory analysis
- Training and test set creation
- Model fitting using sklearn
- Hyperparameter optimization
- Out-of-bag error rate
- Calculating variable importance
- Test set calculations
- Cross validation
- ROC curve estimation


Random Forest Simplified
Instance
Random Forest
...
Tree-1
Tree-2
Tree-n
Class-A
Class-B
Class-B
Majority-Voting
Final-Class

Hyperparameter

- The best hyperparameters 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.



- n_estimators = number of trees in the foreset
- max_features = max number of features considered for splitting a node
- max_depth = max number of levels in each decision tree
- min_samples_split = min number of data points placed in a node before the node is split
- min_samples_leaf = min number of data points allowed in a leaf node
- bootstrap = method for sampling data points (with or without replacement)

Fitting Random Forest

- Now, let's create the model, starting with parameter tuning. Here are the parameters we will be tuning in this tutorial:
- **max_depth**: The maximum splits for all trees in the forest.
- **bootstrap**: An indicator of whether or not we want to use bootstrap samples when building trees.
- **max_features**: The maximum number of features that will be used in node splitting — the main difference I previously mentioned between bagging trees and random forest. Typically, you want a value that is less than *p*, where *p* is all features in your data set.
- **criterion**: This is the metric used to asses the stopping criteria for the decision trees.

Hyperparameter Optimization

- Utilizing the **GridSearchCV** functionality, let's create a dictionary with parameters we are looking to optimize to create the best model for our data. Setting the **n_jobs** to 3 tells the grid search to run three jobs in parallel, reducing the time the function will take to compute the best parameters. I included the timer to see how long different jobs took; that led me to ultimately decide to use three parallel jobs.
- This will help set the parameters we will use to tune one final parameter: the number of trees in our forest.

Hyper-Parameter in Random Forests

- The Hyperparameters in random forest are either used to increase the predictive power of the model or to make the model faster.
- I will here talk about the hyperparameters of sklearns built-in random forest function.

Business Use

Hyperparameters to be tuned

- Hyperparameters are the arguments that can be set before training and which define how the training is done.

The main hyperparameters in Random Forests are:

- The number of decision trees to be combined
- The maximum depth of the trees
- The maximum number of features considered at each split
- Whether bagging/bootstrapping is performed with or without replacement

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 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, described in their documentation.

The last important hyper-parameter we will talk about in terms of speed, is „min_sample_leaf“. This determines, like its name already says, the minimum number of leafs that are required to split an internal node.

2. Increasing the Models Speed

- The „n_jobs“ hyperparameter tells the engine how many processors it is allowed to use. If it has a value of 1, it can only use one processor. A value of “-1” means that there is no limit.
- „random_state“ 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.
- 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 is very similar to the leave-one-out cross-validation method, but almost no additional computational burden goes along with it.

Variable Importance

- Once we have trained the model, we can assess variable importance. One downside to using ensemble methods with decision trees is that you lose the interpretability a single tree gives. A single tree can outline for us important node splits, as well as variables that were important at each split.
- Fortunately, ensemble methods that rely on CART models use a metric to evaluate the homogeneity of splits. Thus, when creating ensembles, these metrics can be utilized to give insight into the important variables used in the training of the model. Two such metrics are **gini impurity** and **entropy**. Many people favor **gini impurity** because it has a lower computational cost than **entropy,** which requires calculating the logarithmic function. For more information, I recommend reading this article.
- Here we define each metric:
- and

Cross Validation

- Cross validation is a powerful tool that is used for estimating the predictive power of your model, and it performs better than the conventional training and test set. Using cross validation, we can create multiple training and test sets and average the scores to give us a less biased metric.
- In this case, we will create 10 sets within our data set that calculate the estimations we have done already, then average the prediction error to give us a more accurate representation of our model's prediction power. The model's performance can vary significantly when utilizing different training and test sets.
- For a more thorough explanation of cross validation I recommend reading An Introduction to Statistical Learnings with Applications in R, specifically chapter 5.1

K-Fold Cross Validation

- Here we are employing K-fold cross validation; more specifically, 10 folds. We are creating 10 subsets of our data on which to employ the training and test set methodology; then we will average the accuracy for all folds to give us our estimation.
- Within a random forest context, if your data set is significantly large, you can choose to not do cross validation and instead use the OOB error rate as an unbiased metric for computational costs. But for the purposes of this tutorial, I included it to show the different accuracy metrics available.

Test Set Metrics

- ROC Curve Metrics

- A receiver operating characteristic (ROC) curve calculates the false positive rates and true positive rates across different thresholds. Let's graph these calculations.
- If our curve is located in the top left corner of the plot, that indicates an ideal model; i.e., a false positive rate of 0 and true positive rate of 1. On the other hand, a ROC curve that is at 45 degrees is indicative of a model that is essentially randomly guessing.
- We will also calculate the area under the curve (AUC). The AUC is used as a metric to differentiate the prediction power of the model for patients with cancer and those without it. Typically, a value closer to 1 means that our model was able to differentiate correctly from a random sample of the two target classes of two patients with and without the disease.

Classification Report

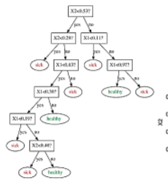
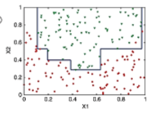
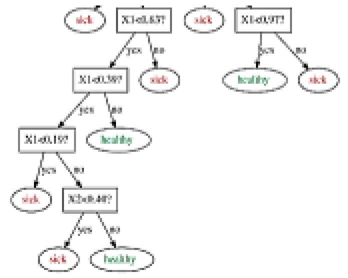
- I'm going to go ahead and create a classification report for our model, which is available through **sklearn.metrics.** This report provides many important classification metrics including:
- **Precision**: Also called the positive predictive value, this metric is the number of correct predictions divided by the number of correct predictions plus false positives, so .
- **Recall**: Also known as the sensitivity, this is the number of correct predictions divided by the total number of instances, so where is the number of false negatives.
- **f1-score**: This is defined as the *weighted harmonic mean* of both the precision and recall, where the best f1-score is 1 and this worst value is 0, as defined by sklean's documentation.
- **support**: The number of instances that are the correct target values.

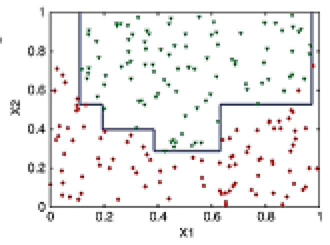
Visualiza

- Each split leads to a ... arts. Thus, the final decision boundary
- Each split leads to a ... arts. Thus, the final decision boundary

Visualization:

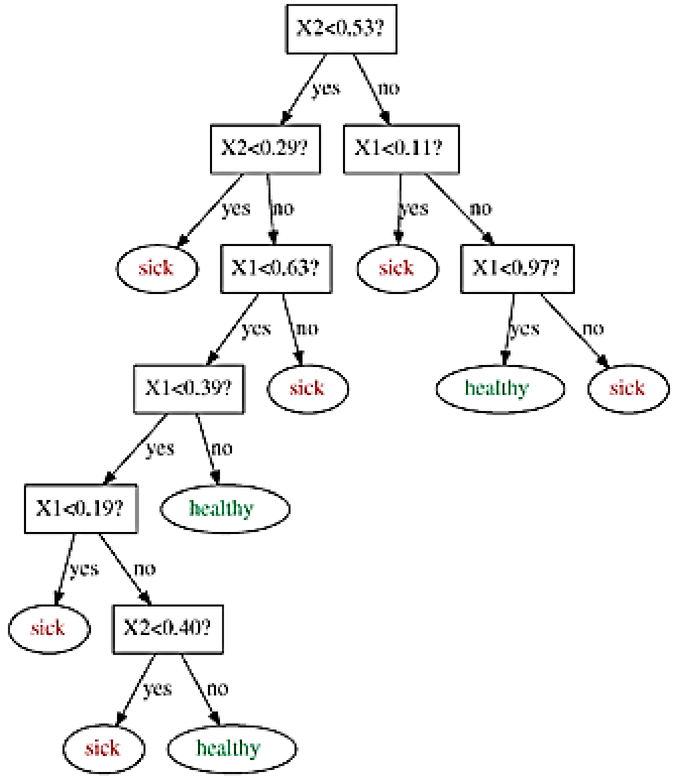
- Each split leads to a straight line classifying the dataset into two parts. Thus, the final decision boundary will consist of straight lines (boxes).
- Each split leads to a straight line classifying the dataset into two parts. Thus, the final decision boundary will consist of straight lines (or boxes).

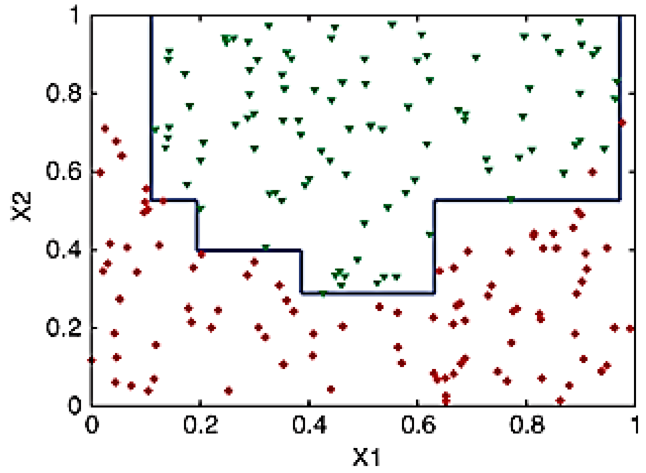






Visualization

- Each split leads to a straight line classifying the dataset into two parts. Thus, the final decision boundary will consist of straight lines (boxes).
- Each split leads to a straight line classifying the dataset into two parts. Thus, the final decision boundary will consist of straight lines (or boxes).





https://www.datascience.com/resources/notebooks/random-forest-intro

- Ensembles are a divide-and-conquer approach used to improve performance. The main principle behind ensemble methods is that a group of “weak learners” can come together to form a “strong learner”. The figure below (taken from here) provides an example. Each classifier, individually, is a “weak learner,” while all the classifiers taken together are a “strong learner”.

Tuning in Random Forest

- As we know, strings of a guitar may loosen up and sound different over time. Hence, strings must be adjusted i.e tightened or loosened so they produce the “right ”sound. In the same manner, the parameters in the random forest are tuned to increase the predictive power or to make the model faster.
- Taking you through sklearn’s few inbuilt parameters:
- n_estimators is the number of decision trees that the algorithm creates. As the number tree increases, the performance increases and the predictions are more stable but it slows down the computation.
- max_features is the maximum number of features that the algorithm can assign to an individual tree.
- n_jobs is the number of jobs to run in parallel. If n_jobs=1, it uses one processor. If n_jobs=-1, then the number of jobs is set to the number of cores available.
- max_depth is the maximum depth of the tree.
- criterion is the function to measure the quality of a split. Supported criteria are “gini” for the Gini impurity and “entropy” for the information gain.
- To view the complete parameter list, visit sklearn’s Random Forest module.

Disadvantages of Random Forest

- Random Forest is difficult to interpret. Because of averaging the results of many trees becomes hard for us to figure out why a random forest is making predictions the way it is.
- Random Forest takes a longer time to create. It is computationally expensive compared to a Decision Tree.

Performance:

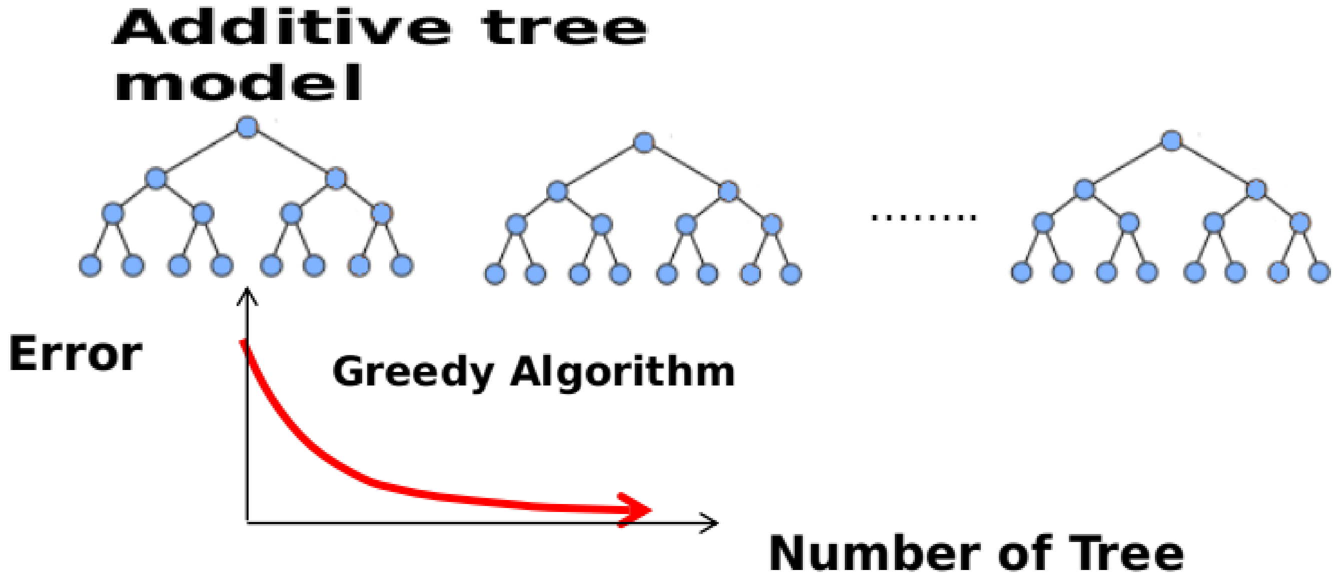
- The Receiver Operating Characteristic (ROC) is a plot that can be used to determine the performance and robustness of a binary or multi-class classifier.
- The x-axis is the false positive rate (FPR) and the y-axis is the true positive rate (TPR).
- The ROC plot gives information about the about true postive/negative rate and false positive/negative rate and something called the C-statistic or area under ROC curve (AUROC) for each class predicted by the classifier (there is one ROC for each class predicted by the classifier).
- The AUROC is defined as the probability that a randomly selected positive sample will have a higher prediction value than a randomly selected negative sample.

- “Assuming that one is not interested in a specific trade-off between true positive rate and false positive rate (that is, a particular point on the ROC curve), the AUC [AUROC] is useful in that it aggregates performance across the entire range of trade-offs. Interpretation of the AUC is easy: the higher the AUC, the better, with 0.50 indicating random performance and 1.00 denoting perfect performance.”

- The AUROCs for different classifiers can be compared against each other.
- Alternatives to this metric include using the scikit-learn confusion matrixcalculator for the prediction results and using the resultant matrix to derive basic positive/negative accuracy, F1-scores, etc.

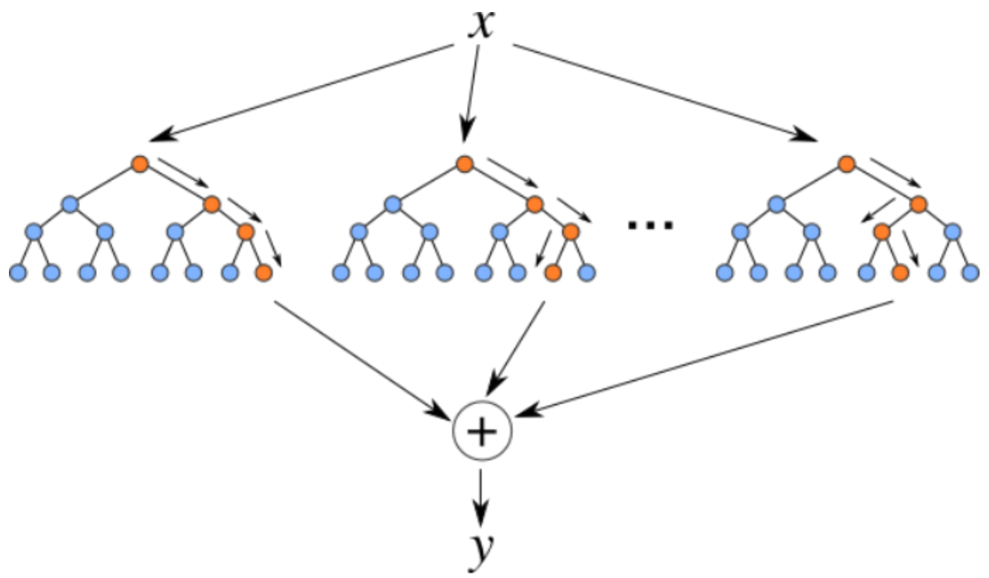
XGBoost

- Additive tree model: add new trees that complement the already-built ones
- Response is the optimal linear combination of all decision trees
- Popular in Kaggle competitions for efficiency and accuracy

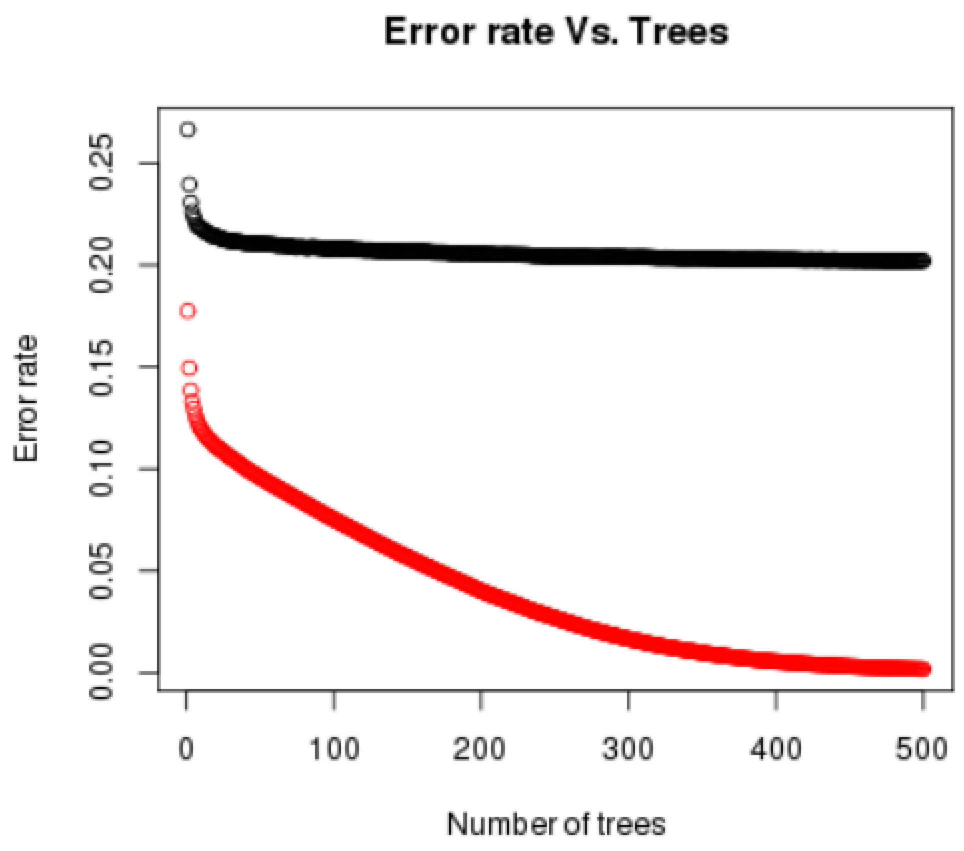


XGBoost

- Additive tree model: add new trees that complement the already-built ones
- Response is the optimal linear combination of all decision trees
- Popular in Kaggle competitions for efficiency and accuracy



XGBoost



Confusion matrix		Prediction	
		0	1
Truth	0	35744	1467
	1	8201	2999

	Accuracy
Overall	80.0%
Target = 1	26.8%
Target = 0	96.1%

- Predicting wine quality using Random Forests
- https://datascienceplus.com/predicting-wine-quality-using-random-forests/