**Why Random Forest is the greatest!**

[](https://medium.com/@menezesborges?source=post_page-----8ae226855565----------------------)

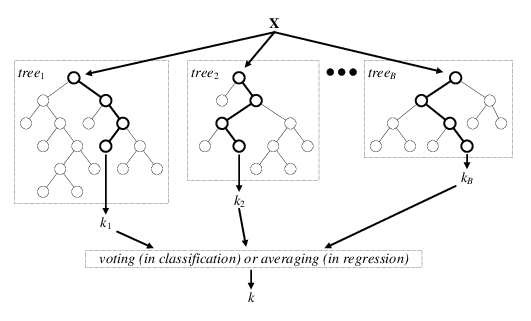
[Diogo Menezes Borges](https://medium.com/@menezesborges?source=post_page-----8ae226855565----------------------)

[Sep 20, 2018](https://medium.com/diogo-menezes-borges/random-forests-8ae226855565?source=post_page-----8ae226855565----------------------) · 4 min read

*I’ve start by talking about* [*Decision Trees*](https://medium.com/diogo-menezes-borges/a-breath-of-fresh-air-with-decision-trees-e660455bbfc8)*, then went on understanding what is* [*Ensemble Learnin*](https://medium.com/diogo-menezes-borges/ensemble-learning-when-everybody-takes-a-guess-i-guess-ec35f6cb4600)*g. Now its time to mix the two and get lost in the forest like our friend Vincent Vega.*

Random Forests is a [supervised learning](https://medium.com/diogo-menezes-borges/learn-to-classify-machine-learning-systems-in-3-criteria-9259da49a9fe) algorithm which, just as the name unveils, is an ensemble of several trees (i.e. [Decision Tree](https://medium.com/diogo-menezes-borges/ensemble-learning-when-everybody-takes-a-guess-i-guess-ec35f6cb4600) algorithm). Generally, they are trained via bagging method or sometimes pasting. For this post, I’ll not go into details on what’s Ensemble Learning or Bagging but you can get [enlightened](https://medium.com/diogo-menezes-borges/ensemble-learning-when-everybody-takes-a-guess-i-guess-ec35f6cb4600) on the matter here.

Decision Trees is a great algorithm, nevertheless, just one run on the data will probably not return the best prediction. However, what if you run several decision tree algorithms, with different subsets of your training data and in the end, see which was the most commonly given answer for each instance? This is the basis of Ensemble Learning and thus Random Forest. Just look at the picture for a simple example.



[Source](https://www.researchgate.net/figure/Architecture-of-the-random-forest-model_fig1_301638643)

You have your dataset X. You subdivide this dataset into several subsets (either by bagging or pasting). Per subset you train a Decision Tree algorithm. In the end, you have trained several decision trees each returning their prediction for each observation in the dataset. You are now able to choose, per observation, what is the most probable answer to be expected. Individually, the predictions made by each model may not be accurate but combined together those predictions will be closer to the mark on average.

Looking at it step-by-step, this is what a random forest model does:

1. Random subsets are created from the original dataset (bootstrapping).
2. At each node in the decision tree, only a random set of features are considered to decide the best split.
3. A decision tree model is fitted on each of the subsets.
4. The final prediction is calculated by averaging the predictions from all decision trees.

**Training a Random Forest in Python**

Let’s train a Random Forest classifier with 500 trees (each limited to a maximum of 16 nodes):

**from** sklearn.ensemble **import** RandomForestClassifier  
**from** sklearn.datasets **import** load\_iris  
**from** sklearn.model\_selection **import** train\_test\_split  
**import** pandas **as** pd  
**import** numpy **as** pdiris = load\_iris()  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(iris["data"], iris["target"], test\_size=0.3, random\_state=1)rnd\_clf= RandomForestClassifier(n\_estimators = 500, max\_leaf\_nodes= 16, n\_jobs = -1, random\_state=1)rnd\_clf.fit(X\_train,y\_train)  
y\_pred\_rf = rnd\_clf.predict(X\_test)





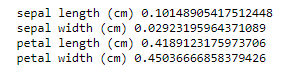
RandomForestClassifier has all the hyperparameters of a DecisionTreeClassifier to control the tree’s growth, with few exceptions, plus all hyperparameters from BaggingClassifier to control the ensemble. Moreover, the Random Forest introduces extra randomness when growing trees, since instead of searching for the best feature when splitting a node it searches for the best feature among a random subset of features. With this one can achieve higher tree diversity, avoids overfitting and allows for higher bias and lower variance yielding a better model.

**Feature Importance**

You probably already heard the expression “Feature Importance”. Well, this is one of the best qualities of Random Forests, because it allows you to measure the impact each predictor has on the final prediction.

for name, score in zip(iris[“feature\_names”], rnd\_clf.feature\_importances\_):  
 print(name,score)





This feature comes in very handy when you’re trying to figure out which features actually matter, in particular, if you need to perform feature selection.

**Tuning Hyperparameters**

By tuning the hyperparameters you can achieve two goals, or either increase the predictive power of the model or make it faster.

* **Increasing predictive power**

n\_estimators : Number of trees the algorithm growths before making the prediction. A higher number of tree is expected to increase the performance and make the prediction more stable with the disadvantage of slowing down the computation;

max\_features : corresponds to the maximum features Random Forest is allowed to try in an individual tree, when looking for the best split;

min\_sample\_leaf : minimum required number of leaves to make the split on an internal node.

* **Increasing processing speed**

n\_jobs : indicates the system how many processors it is allowed to use. Value of ‘-1’ means there is no limit;

random\_state : makes the model’s output replicable. It will always produce the same results when you give it a fixed value as well as the same parameters and training data.

oob\_score : Random Forest cross-validation method. These are called the out-of-bag samples. It is very similar to leave-one-out cross-validation method but without any computational burden.

**Conclusion**

Random Forest is a great algorithm, for both classification and regression problems, to produce a predictive model. Its default hyperparameters already return great results and the system is great at avoiding overfitting. Moreover, it is a pretty good indicator of the importance it assigns to your features.

The great disadvantage is that a high number of trees might make the computation process much slower and ineffective for real-time predictions.

***If you liked it, follow me for more publications and don’t forget, please, give it an applause!***

You the mighty reader applauding!

**Resources :**

* *Hands-on Machine Learning with Scikit-Learn & TensorFlow* by [Aurélien Géron](https://medium.com/u/c939be75faee?source=post_page-----8ae226855565----------------------), Chapter 7
* Analytics Vidhya, [*A Comprehensive Guide to Ensemble Learning (with Python codes)*](https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-for-ensemble-models/)