Random forest

Tuesday, December 10, 2019 20:54

For the glory of God

What is Random forest ?

- Random forest is one of popular supervised machine learning techniques that operates by constructing a multitude of decision trees.
  (Note that it can be used for both classification and regression problems ; however, it typically works well with a classification problem)
- The first Random forest algorithm was proposed by Tin Kam Ho in 1995.
- The Random forest algorithm is a decision support tool that uses some set of rules to figure out the possible consequences.

Why Random forest ?

- The decision tree algorithm is easy to implement and use ; however, it does not work well in practice.
  - why does it not work well ?
    : As the algorithm generates deep decision trees, it may suffer from overfitting ; which is a critical problem.

What it means is that it works great with the data used to create them ; but it's not flexible when it comes to classifying new samples.

- Random forest prevents the overfitting issue by combining the simplicity of decision trees with flexibility resulting in a vast improvement in accuracy.

How does Random forest work ?

- Basically, the Random forest algorithm has two steps in general.
  - The first step is to create random forest. → It's an ensemble of decision trees (combination of learning models)
  - The second step is to make a prediction from the random forest.
- The whole process is shown below ;

Step 1 : Create a bootstrapped dataset

- Let's imagine that we have the following dataset from which we are going to build a tree.

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	Yes	167	Yes

- To create a bootstrapped dataset that is the same size as the original, we just randomly select samples from the original dataset.

  (Here, note that we are allowed to pick the same sample more than once)

Original Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	Yes	167	Yes

Bootstrapped Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	Yes	167	Yes
Yes	No	Yes	167	Yes

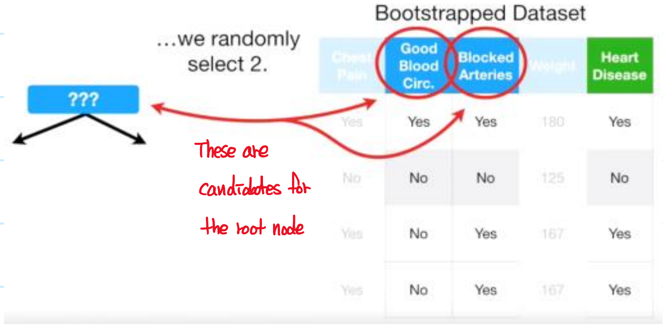
Actually, it's duplicated but it's okay!

· Step 2 : Create a decision tree using the bootstrapped dataset ; but only use a random subset of variables (or columns) at each step.

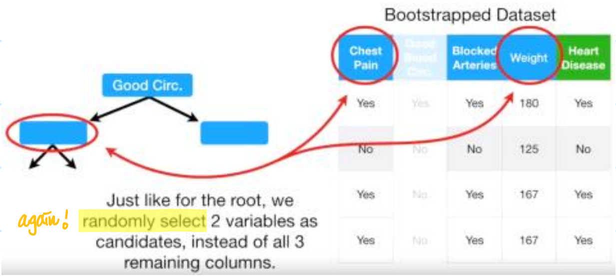
- In this example, let us say that we only consider two variables at each step.

  ↳ In fact, this number is treated as one of hyperparameters in the Random forest algorithm.

- Let's say that we randomly select two (Good blood circ, Blocked arteries) variables to figure out which one should be located at the root.

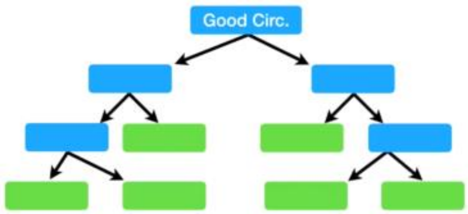


- Let's say that 'Good Blood Circulation' did the best job separating the samples. (⇔ The highest information gain)



↓ We just gray out it because it has been already selected.

- Repeat the process to build the tree but only considering a random subset of variables at each step



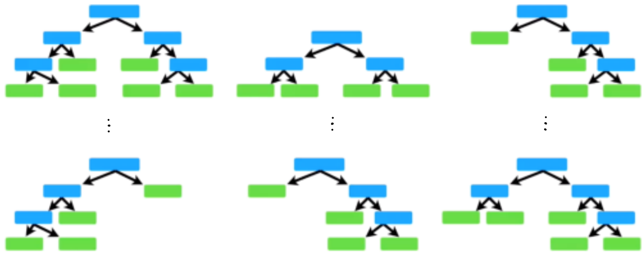
· Step 3 : Make a new bootstrapped dataset and build a tree considering a subset of variables at each step

- Let's say that you repeat this process 100 times ..

  ↳ This results in a wide variety of trees.

  ↳ The variety is what makes Random forests more effective than individual decision tree.

- For example, we might get the following Random forest from this example.



... It will be 100 trees

· Step 4 : Measure the accuracy with Out-of-Bag (OOB) samples

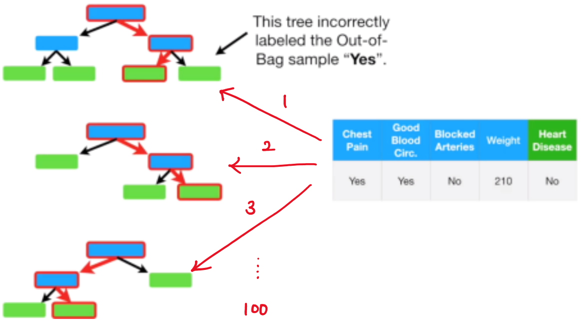
- The main idea is to measure how accurate our random forest is by the proportion of OOB samples that were correctly classified by the Random forest.

- Then, what is OOB ?

  ↳ Recall that we allowed duplicate entries in the bootstrapped dataset.

  ↳ This means we have a dataset from the original dataset that was not used to create the tree.

- We can run the OOB through and see if it correctly classifies the sample with all of the trees.



↳ Let's say, for this example, we have { Yes : 20 ; thus, the OOB correctly classified.
No : 80 (∵ The real data says 'No')

- In a similar way, we will repeat the process with the all OOB datasets with all of the trees to measure the accuracy.

· Step 5 : Tune hyperparameters to obtain the best Random forest model

- Typically, the proportion of OOB samples that were incorrectly classified is defined as the 'Out-of-Bag Error'.

- We are encouraged to choose the best hyperparameters to reduce the OOB error (i.e. the best model).

What are hyperparameters of Random forest algorithm?

1) Number of trees : Increasing the number of trees generally decreases the variance of the overall model.

2) Number of features to consider per split : Considering more features increases the chance of finding a better split ; however, it also increases the variance

3) Tree size : It can be controlled by defining maximum depth, maximum number of nodes, and minimum number of nodes at leaf. Larger trees generally are more complex and increase the possibility to overfit.

(e.g. The deeper, the more capture the information ; however, the more possibility to overfit)

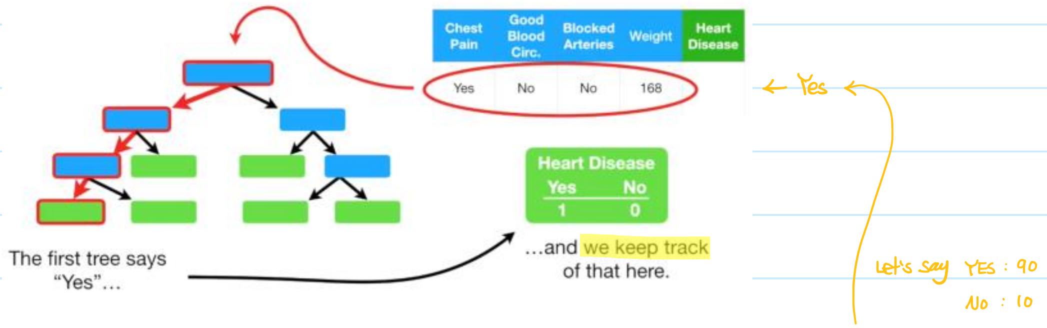
Hyperparameter	Description	Typical default values
mtry	Number of drawn candidate variables in each split	$\sqrt{p}$, $p/3$ for regression
sample size	Number of observations that are drawn for each tree	n
replacement	Draw observations with or without replacement	TRUE (with replacement)
node size	Minimum number of observations in a terminal node	1 for classification, 5 for regression
number of trees	Number of trees in the forest	500, 1000
splitting rule	Splitting criteria in the nodes	Gini impurity, p-value, random

Table 1: Overview of the different hyperparameter of random forest and typical default values. n is the number of observations and p is the number of variables in the dataset.

· Step 6 : Validate the Random forest by using a new sample data

- Using the dataset, we test the variables with all trees and count how many 'YES' (or 'No') from the Random forest.

- For example, the following screenshot shows the first test.



- After running the data down all of the trees in the Random forest, we see which option received more votes.

What are advantages and disadvantages of Random forest ?

a) Advantages

· It works for both classification and regression problems.

· It handles the missing values and maintains accuracy for missing data.

· It rarely overfits compared with decision tree.

· It handles large dataset with higher dimensionality.

· There are relatively few hyperparameters compared to other techniques ; and they are even straightforward to understand.

b) Disadvantages

· It does usually great job at classification but not as good as for regression problem.

· We have little control on what the model does.

· A large number of trees can make the algorithm too slow and ineffective for real-time predictions.

Is Random forest a black box algorithm ?

· In short answer, it can be either Yes or No. Let's take a deep dive into the discussion.

a) Random forest as a black box

· Most literatures on Random forest lead us the conclusion that Random forest is typically treated as a black box.

· They mostly claim as following ;

- Random forest generally consists of a large number of deep trees.
- Here, each tree is trained on bagged data using random selection of features.
- Therefore, gaining a full understanding of the decision process is almost impossible.

b) Turning a black box into a white box

· Someone may claim that some knowledge can be obtained from the Random forest algorithm.

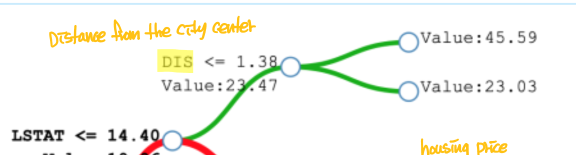
· For example,

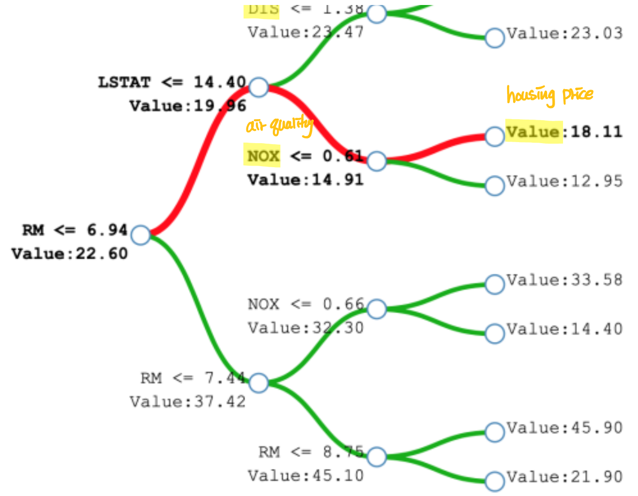
- One way of getting an insight into Random forest is to compute features importances.
  ↳ It can be done by permuting the values of each feature one by one and checking how it changes the model performance.
- This is useful to get some insights ; however, it never gives us an insight in understanding how the algorithm makes the decision.

which features are important for overall random forest model.

· Then, how could we figure out the decision way of Random forest algorithm ?

- First of all, it is intuitively clear that a tree makes there is a path from the root of the tree to the leaf, consisting of a series of decisions.
- This means that we would be able to generate a prediction path.
- For example, let's say that we are now talking about a decision tree for regression problem.





Prediction: 18.11 ≈ 22.60 (trainset mean) - 2.64(loss from RM) - 5.04(loss from LSTAT) + 3.20(gain from NOX)

RM	LSTAT	NOX	DIST	
3.1	4.5	0.54	2.6	Predict
6.5	16.1	0.12	2.2	Predict
7.1	10.5	0.31	1.8	Predict

↳ Here, it's written down in terms of value changes along the prediction path.

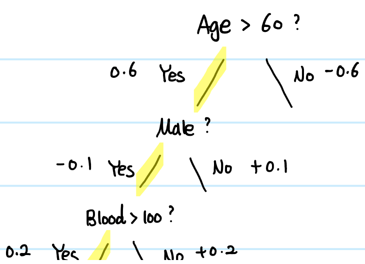
· Therefore, every prediction can be trivially presented as a sum of feature contributions, showing how the features lead to a particular prediction.

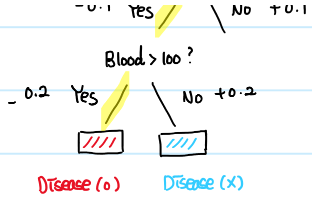
- For classification example,

For example, there is a RF model which predicts — ***a patient** X coming to **hospital has high probability of readmission or not?*** *For sake of simplicity, let's consider we only have 3 features* — `patient's blood pressure data`, `patient's age` and `patient's sex`. *Now, if our model says that* `patient A` *has 80% chances of readmission, how can we know* ***what is special in that person A that our model predicts he/she will be readmitted ?*** . *In this case, tree interpreter tells the* ***prediction path*** *followed for that particular patient.*

Patient A			
Feature	**Value**	**Contributions**	**Path**
Bias		0.3	0.3
Age	65	0.6	0.3+0.6
Sex	M	-0.1	0.3+0.6-0.1
Blood pressure	120	-0.2	0.3+0.6-0.1-0.2
Prediction			0.6

e.g.




-0.1 Yes
No +0.1
Blood > 100 ?
- 0.2 Yes
No +0.2
Disease (o)
Disease (X)