StatQuest

Random Forest - Part 1

Building using, and Evaluating

- Decision trees are not flexing in order to classify new samples.

Building Random Forest

Step1: create a "bootsraped" dataset.

original-dataset (Target)

Chest pain	Good Blood	F3	F4	F5
No	No	No	125	Yes
Yes	Yes	Yes	180	No
Yes	Yes	No	210	No
Yes	No	Yes	167	Yes

Bootstrapped Dataset

Chest pain	Good blood	F3	F4	F5
Yes	Yes	Yes.	180	No
No	No	No	125	Yes
Yes	No	Yes	167	Yes
Yes	Yes	Yes	180	No

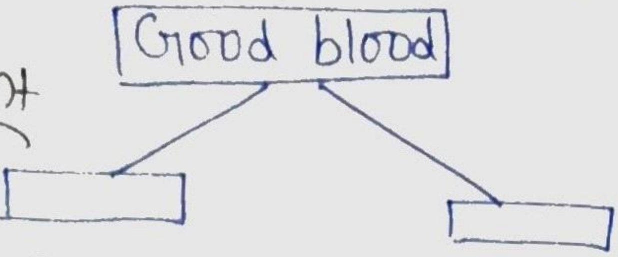
a) To create bootstrapped datset, ie of same size of original dataset, we just randomly select samples from the original dataset

b) we are allowed to pick the same sample more than once.

Step-2: create a decision tree using the bootsrapped dataset, but only use a random subset of variables (features) at each step.

In this case, we will randomly select two Variables, "chest" and "Good blood" ~~as for~~ as a condidate for root node.

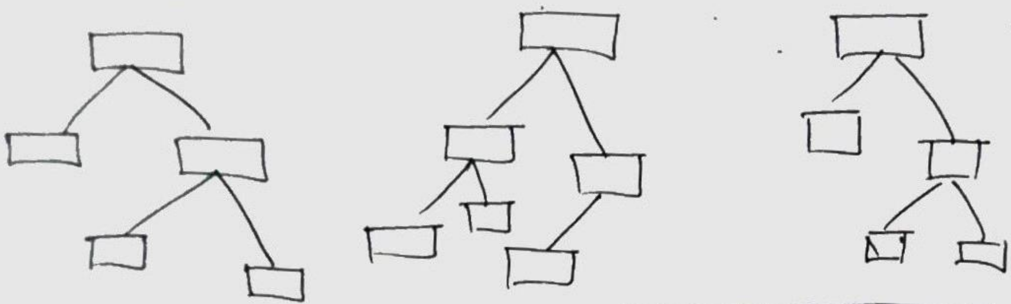
Assume, "Good blood" has lowest cross-entropy,



at this point also, rondomly select two variable from "3" variables left.

and build the tree as usual.

Now, go back to step 1 and repeat: Make a new bootstrapped dataset and build a tree consider rondom subset of variables at each split.



This will result in $\boxed{\text{wide variety}}$ of decision trees.

↓ Makes Rondom forest more effecting than individual decision trees.

How to predict?

Run the somple on all tree we have created. and calculates voting of yes vs No

Heart_disease (Target)	
Yes	No
5	1

⇒ class = Yes

Random Forests- Part 2

Missing data and sample clustering

Random forest Consider two types of missing data

1. Missing data in original dataset.
2. Missing data in New sample.

→ We can make initial guess → refine our guess

↓ using mean, median or other imputation methods.

initial guess

Chest pain	Good blood	Blocked Artery	Weight	Target
Yes	Yes	No	167.5	No

How to refine our guesses

Step-1: Build Random forest...

Step-2: Run all data in all of distinct decision trees.

Find the samples which end up at same 'leaf Node' as our data-sample which we want to make a guess.

No. of Rows = No. of Column = Total samples

Build Proximity Matrix

	1	2	3	4
1		2	1	1
2	2		1	1
3	1	1		8
4	1	1	8	0

Suppose → 3 and 4 end up in same leaf Node, we will add 1 in P[3][4] and P[4][3]

→ We divide each value by Total No. of Trees.

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

Random Forest

Terminology:

Bootstrapping the data plus using the aggregate to make a decision is called "Bagging".

→ Typically $1/3^{rd}$ samples does not end up in bootstrapped dataset.

$\boxed{33\% \text{ of data}}$ → Called the (out-of bag dataset)
↓
doesn't appear in bootstrapped dataset.

* Since out of bag-datasets were not used in ~~classi~~ building decision tree, we will use it to evaluate our model.

* The proportion of out of bag samples incorrectly classified by Random Forest is known as "out of bag error".

1. Build Random Forest

2. Estimate Accuracy of Random Forest

(Change No. of variables)

Usually, we start with $\sqrt{\text{No. of variables}}$.

Random Forest Part 2.2

Use proximity matrix to calculate missing values.

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	Yes	???	???	No

frequency(Yes) = $\frac{1}{3}$, f(No) = $\frac{2}{3}$

weighted_f(Yes) = $\frac{1}{3} \times$ (The weight for Yes) (0.1) = $\boxed{0.03}$

The weight of Yes = $\left(\frac{\text{Proximity of Yes}}{\text{All Proximities}}\right)$ = Proximity value for sample 2

1	2	3	4
0.1	0.1	0.8	

$= \left(\frac{0.1}{0.1+0.1+0.8}\right) = \frac{0.1}{1} = 0.1$

weighted_f(No) = $\frac{2}{3} *$ (The weight for No) (0.9) = 0.6

The weight for No = sample 1 and sample 3

$\frac{0.8+0.1}{0.1+0.8+0.1} = \frac{0.9}{1} = 0.9$

(No > Yes)

New value = No

~~Lecture-23 SVM: for linearly non-seperable data~~ 3

weightage_average of (weight) = 125 * (proximity weight)

+ (180) * (proximity_weight)

+ (210) * (proximity weight)

Proximity weight (125) = $\frac{0.1}{0.1+0.1+0.8}$ = 0.1

proximity weight (180) = $\frac{0.1}{0.1+0.1+0.8}$ = 0.1

Proximity_weight (210) = $\frac{0.8}{1}$ = 0.8

= 125 * (0.1) + 180 * (0.1) + 210 * (0.8) = (198.5)

After refining guesses, we run random forest again to refine guesses, until missing values converge. (6 to 7 times).

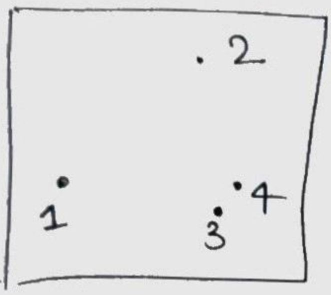
Random Forest- Clustering

Suppose:

$$P = \begin{array}{|c|c|c|c|} \hline & 2 & 1 & 1 \\ \hline 2 & & 1 & 1 \\ \hline 1 & 1 & & 10 \\ \hline 1 & 1 & 10 & \\ \hline \end{array} \rightarrow \underset{\substack{\text{Total} \\ \text{(No. of Trees)}}}{\underset{\downarrow}{P/10}} = \begin{array}{|c|c|c|c|} \hline & 0.2 & 0.1 & 0.1 \\ \hline 0.2 & & 0.1 & 0.1 \\ \hline 0.1 & 0.1 & & 1 \\ \hline 0.1 & & \boxed{1} & \\ \hline \end{array} \rightarrow$$

$$1 - P/10 = \text{distence matrix} = \begin{array}{c|c|c|c|c|} & 1 & 2 & 3 & 4 \\ \hline 1 & & 0.8 & 0.9 & 0.9 \\ \hline 2 & 0.8 & & 0.9 & 0.9 \\ \hline 3 & 0.9 & 0.9 & & 0 \\ \hline 4 & 0.9 & & 0 & \\ \hline \end{array}$$

$\Rightarrow$ $\boxed{\text{sample '3' and '4' are closest.}}$



How to handle missing data from new sample

				Target
Yes	NO	222	168	

Step1 - create copies of this same with all target values.

Yes	NO	222	168	Yes
yes	NO	222	168	NO

use iterative method to fill missing values.

suppose we got

yes	NO	Yes	168	Yes
Yes	No	No	168	No

→ Run through random forest and see how many times it has been correctly classified

If (Yes, Yes) was classified more than (No, No), then prediction for missing data is (Yes) and target is (Yes)