Random Forest

Part 3: with R package randomForest

Source: Chew C.H. (2022) A.I., Analytics & Data Science, Vol. 2.

Random Forest with R

- What are the default settings for B and RSF size in R?
- Where do we see OOB error?
- How do we check if the error had converged?

  "...in practice we use a value of B sufficiently large for the error rate to have settled down."   -- ISLR

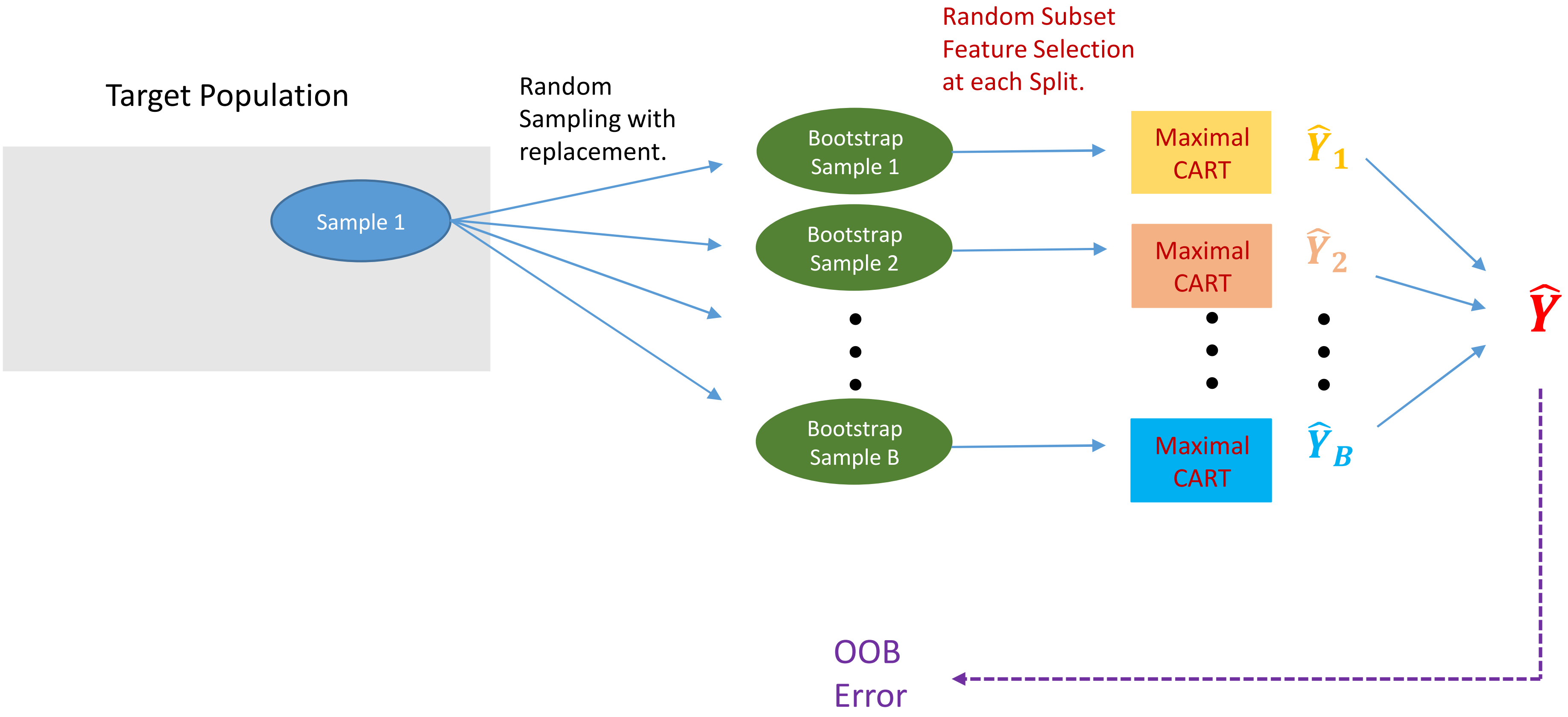
- What are the useful charts/tables and diagnostics in R?
- What to watch out for if we use Python implementation instead of R?

Source: Chew C.H. (2022) A.I., Analytics & Data Science, Vol. 2.

Random Forest = Bagging + Random Subset Feature

Source: Chew C.H. (2022) A.I., Analytics & Data Science, Vol. 2.

Random Forest Process



Random Subset Feature Selection at each Split.

Target Population

Random Sampling with replacement.

Sample 1

Bootstrap Sample 1

Bootstrap Sample 2

Bootstrap Sample B

Maximal CART

Maximal CART

Maximal CART

$\hat{Y}_1$

$\hat{Y}_2$

$\hat{Y}_B$

$\hat{Y}$

OOB Error

Source: Chew C.H. (2022) A.I., Analytics & Data Science, Vol. 2.

R settings

B = 500 [new default in R]

Categorical Y

- RSF size = floor(sqrt(M))

Continuous Y

- RSF size = floor(M/3)

Notes:

- M is the number of X variables.
- The floor() function is the greatest integer smaller than the value.
- *Examples: floor(3.1) = 3, floor(3.9) = 3.*
- For categorical Y, Breiman(2001) proposed int(log$_2$(M) + 1). Actually, not much difference if M is a small number.
- For continuous Y, correlation betw CARTs is less sensitive to RSF size.

  "Correlation increases quite slowly as the number of features used increases." --- Breiman(2001)
- *Hence, we allow bigger RSF size for continuous Y as risk of highly correlated CARTS is lower.*
- *Examples:*
  - If M = 10, floor(sqrt(10)) = floor(3.16…) = 3; floor(10/3) = floor(3.33…) = 3.
  - If M = 100, floor(sqrt(100)) = floor(10) = 10; floor(100/3) = floor(33.33…) = 33!

Source: Chew C.H. (2022) A.I., Analytics & Data Science, Vol. 2.

Comparisons of Different RSF Size Formula Outputs

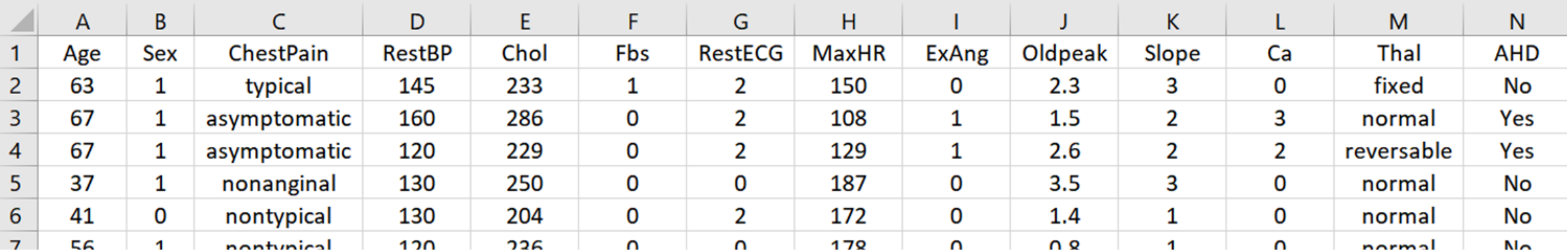
	For Categorical Y		For Continous Y
M	INT($\log_2$(M) + 1)	INT(sqrt(M))	INT(M/3)
5	3	2	1
10	4	3	3
15	4	3	5
20	5	4	6
25	5	5	8
30	5	5	10
50	6	7	16
100	7	10	33
500	9	22	166
1000	10	31	333

Source: Chew C.H. (2022) A.I., Analytics & Data Science, Vol. 2.

Dataset: Heart.csv

Source: https://archive.ics.uci.edu/ml/datasets/Heart+Disease

- 303 cases
- 13 Xs and 1 categorical Y (AHD)
- M = 13 and RSF size = floor(sqrt(13)) = floor(3.605…) = 3
- 6 missing values

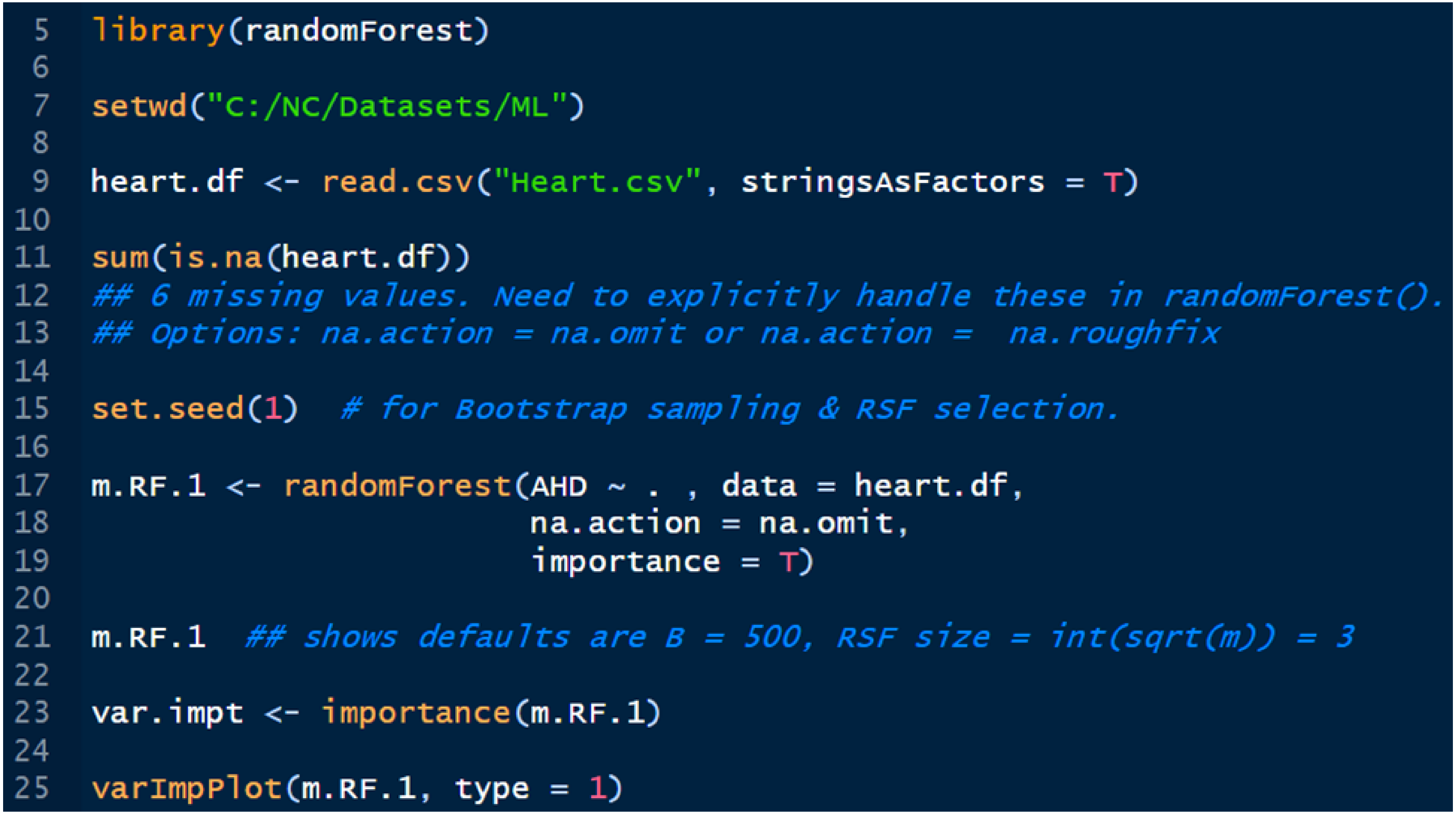


	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal	AHD
2	63	1	typical	145	233	1	2	150	0	2.3	3	0	fixed	No
3	67	1	asymptomatic	160	286	0	2	108	1	1.5	2	3	normal	Yes
4	67	1	asymptomatic	120	229	0	2	129	1	2.6	2	2	reversable	Yes
5	37	1	nonanginal	130	250	0	0	187	0	3.5	3	0	normal	No
6	41	0	nontypical	130	204	0	2	172	0	1.4	1	0	normal	No
7	56	1	nontypical	120	236	0	0	178	0	0.8	1	0	normal	No

Data Dictionary: Heart Data Dictionary.txt

Source: Chew C.H. (2022) A.I., Analytics & Data Science, Vol. 2.

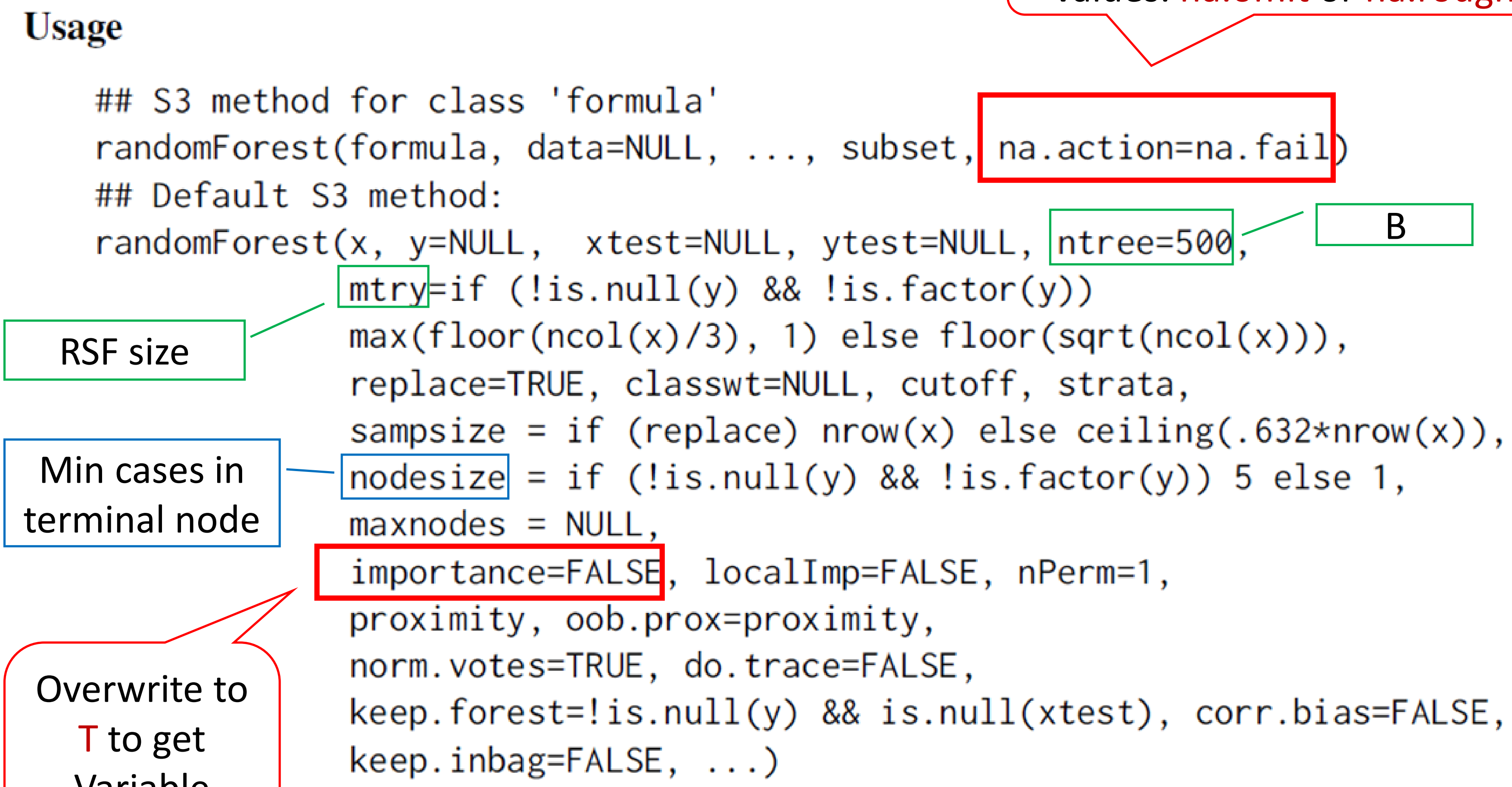
Rscript: RF Heart1.R



```r
library(randomForest)

setwd("C:/NC/Datasets/ML")

heart.df <- read.csv("Heart.csv", stringsAsFactors = T)

sum(is.na(heart.df))
## 6 missing values. Need to explicitly handle these in randomForest().
## Options: na.action = na.omit or na.action =  na.roughfix

set.seed(1)  # for Bootstrap sampling & RSF selection.

m.RF.1 <- randomForest(AHD ~ . , data = heart.df,
                       na.action = na.omit,
                       importance = T)

m.RF.1  ## shows defaults are B = 500, RSF size = int(sqrt(m)) = 3

var.impt <- importance(m.RF.1)

varImpPlot(m.RF.1, type = 1)
```

Source: Chew C.H. (2022) A.I., Analytics & Data Science, Vol. 2.

?randomForest() at console or open randomForest.PDF for documentation



Usage

```
## S3 method for class 'formula'
randomForest(formula, data=NULL, ..., subset, na.action=na.fail)
## Default S3 method:
randomForest(x, y=NULL,  xtest=NULL, ytest=NULL, ntree=500,
             mtry=if (!is.null(y) && !is.factor(y))
             max(floor(ncol(x)/3), 1) else floor(sqrt(ncol(x))),
             replace=TRUE, classwt=NULL, cutoff, strata,
             sampsize = if (replace) nrow(x) else ceiling(.632*nrow(x)),
             nodesize = if (!is.null(y) && !is.factor(y)) 5 else 1,
             maxnodes = NULL,
             importance=FALSE, localImp=FALSE, nPerm=1,
             proximity, oob.prox=proximity,
             norm.votes=TRUE, do.trace=FALSE,
             keep.forest=!is.null(y) && is.null(xtest), corr.bias=FALSE,
             keep.inbag=FALSE, ...)
```

To overwrite if there are missing values. na.omit or na.roughfix

B

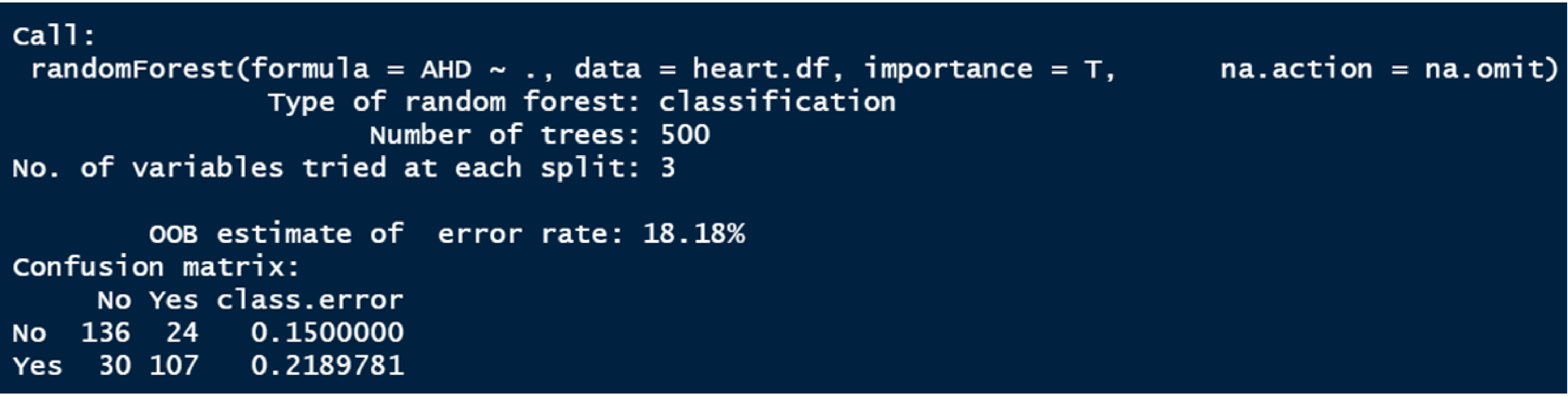
RSF size

Min cases in terminal node

Overwrite to T to get Variable Importance.

Source: Chew C.H. (2022) A.I., Analytics & Data Science, Vol. 2.

Results of Random Forest on Heart data



```
Call:
 randomForest(formula = AHD ~ ., data = heart.df, importance = T,      na.action = na.omit)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 3

        OOB estimate of  error rate: 18.18%
Confusion matrix:
     No Yes class.error
No   136  24   0.1500000
Yes   30 107   0.2189781
```

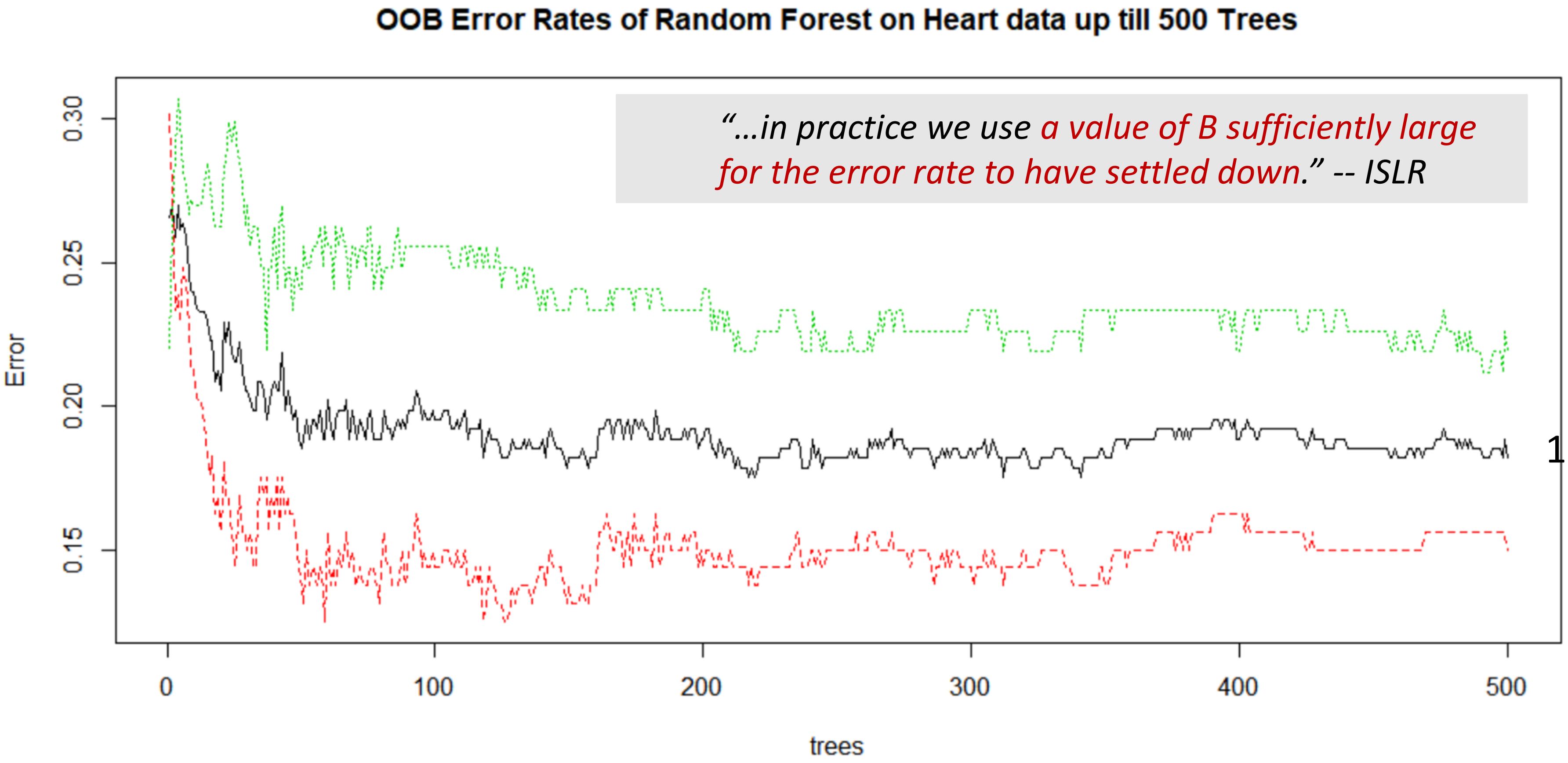
- B = 500, RSF size = 3; OOB overall error = (24 + 30)/297 = 18.18%.

- Q: How are the confusion matrix results determined?
  - Ans: From OOB data and majority rule.

- Q: Why did the confusion matrix contain only 297 cases when the dataset has 303 cases?
  - Ans: 6 cases has missing values and were omitted as na.action= na.omit

Source: Chew C.H. (2022) A.I., Analytics & Data Science, Vol. 2.

View how OOB error rates change with different number of trees with plot(m.RF.1)



"…in practice we use a value of B sufficiently large for the error rate to have settled down." -- ISLR

21.9%

18.18%

15%

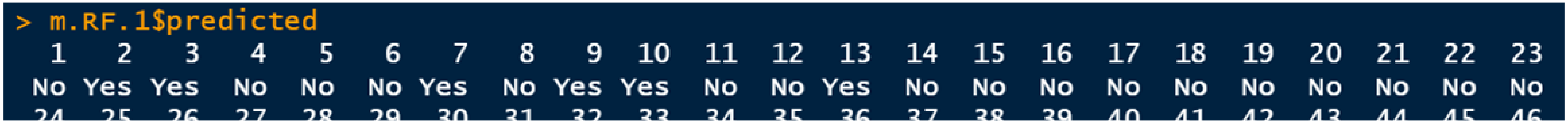
Errors stabilised after approx. 250 trees.

i.e. errors would be approx. the same if ntree = any number bigger than 250.

If errors are still exhibiting decreasing trend at 500 trees, set ntree to be a bigger number > 500 and view the plot to check stability of errors.

Source: Chew C.H. (2022) A.I., Analytics & Data Science, Vol. 2.

View RF prediction for each case via m.RF.1$predicted

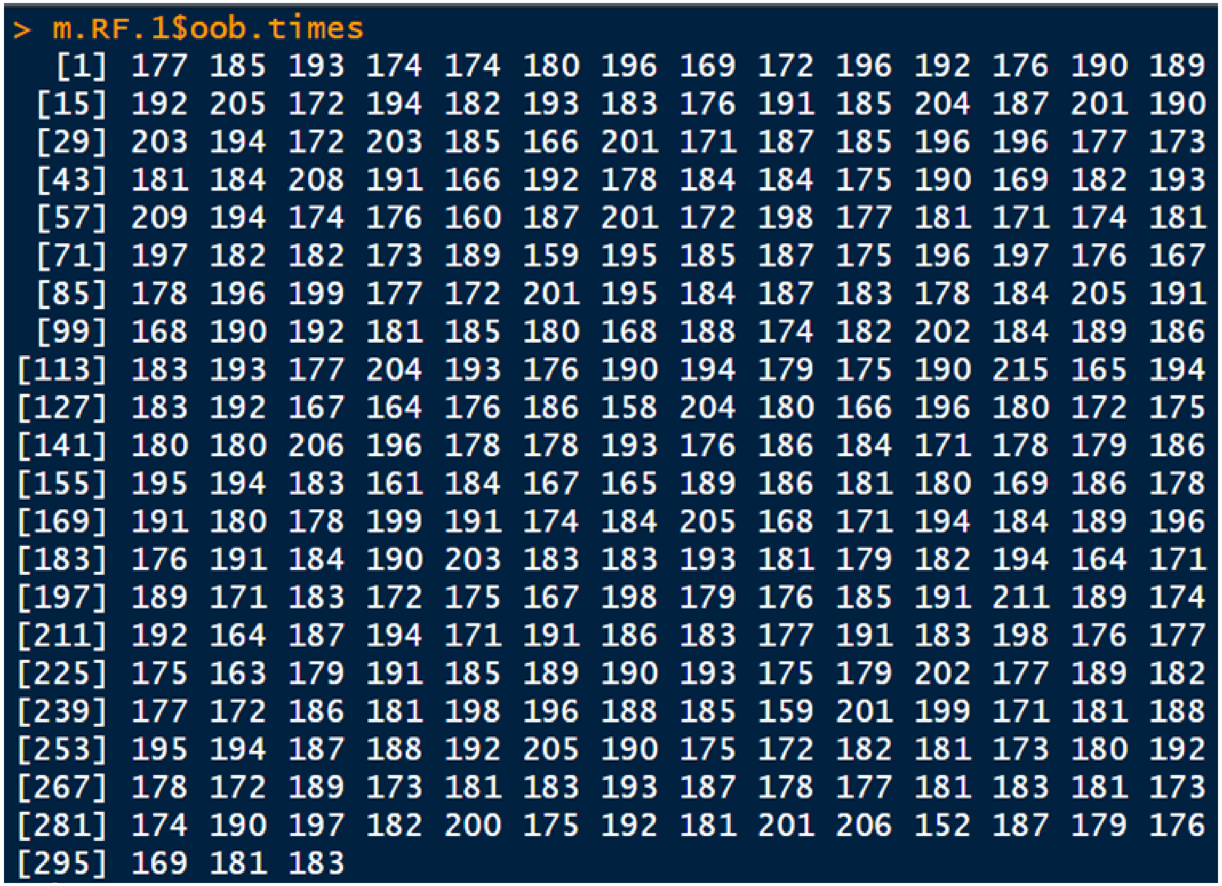


```
> m.RF.1$predicted
  1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 
 No Yes Yes  No  No  No Yes  No Yes Yes  No  No Yes  No  No  No  No  No  No  No  No  No  No 
 24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46 
```

Source: Chew C.H. (2022) A.I., Analytics & Data Science, Vol. 2 & randomForest Documentation.

To check how many times each case is OOB among the 500 trees, m.RF.1$oob.times

P(case i is OOB) = $(1 - 1/n)^n = (1 - 1/297)^{297} \approx 0.367$



```
> m.RF.1$oob.times
  [1] 177 185 193 174 174 180 196 169 172 196 192 176 190 189
 [15] 192 205 172 194 182 193 183 176 191 185 204 187 201 190
 [29] 203 194 172 203 185 166 201 171 187 185 196 196 177 173
 [43] 181 184 208 191 166 192 178 184 184 175 190 169 182 193
 [57] 209 194 174 176 160 187 201 172 198 177 181 171 174 181
 [71] 197 182 182 173 189 159 195 185 187 175 196 197 176 167
 [85] 178 196 199 177 172 201 195 184 187 183 178 184 205 191
 [99] 168 190 192 181 185 180 168 188 174 182 202 184 189 186
[113] 183 193 177 204 193 176 190 194 179 175 190 215 165 194
[127] 183 192 167 164 176 186 158 204 180 166 196 180 172 175
[141] 180 180 206 196 178 178 193 176 186 184 171 178 179 186
[155] 195 194 183 161 184 167 165 189 186 181 180 169 186 178
[169] 191 180 178 199 191 174 184 205 168 171 194 184 189 196
[183] 176 191 184 190 203 183 183 193 181 179 182 194 164 171
[197] 189 171 183 172 175 167 198 179 176 185 191 211 189 174
[211] 192 164 187 194 171 191 186 183 177 191 183 198 176 177
[225] 175 163 179 191 185 189 190 193 175 179 202 177 189 182
[239] 177 172 186 181 198 196 188 185 159 201 199 171 181 188
[253] 195 194 187 188 192 205 190 175 172 182 181 173 180 192
[267] 178 172 189 173 181 183 193 187 178 177 181 183 181 173
[281] 174 190 197 182 200 175 192 181 201 206 152 187 179 176
[295] 169 181 183
```

Checking:

Case 1: P(OOB) = 177/500 = 0.354

Case 2: P(OOB) = 185/500 = 0.37

Case 3: P(OOB) = 193/500 = 0.386

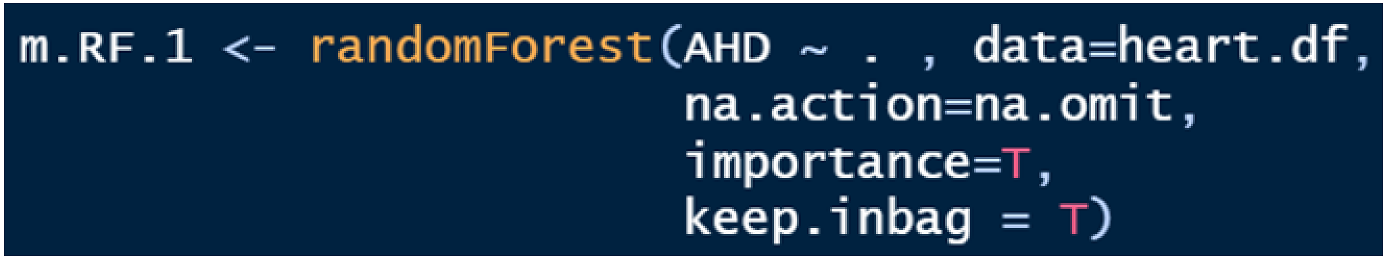
…

Interpretation:

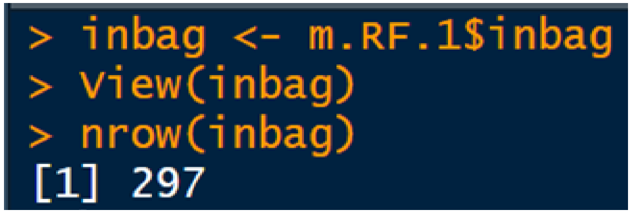
case 1 is not inside 177 of the 500 trees in the forest, …

Source: Chew C.H. (2022) A.I., Analytics & Data Science, Vol. 2.

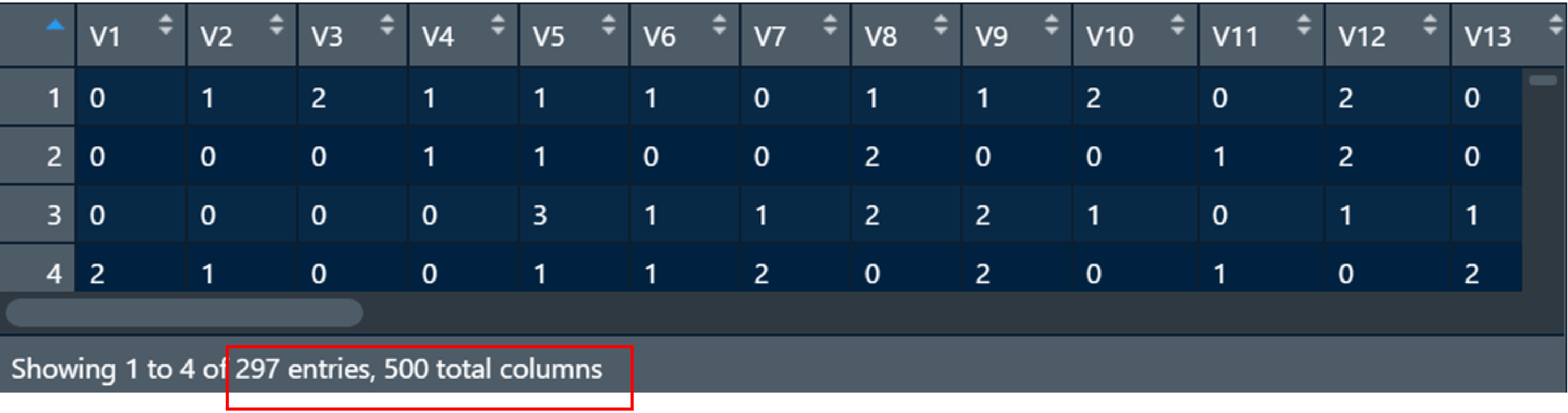
To check whether each case is inbag or OOB in which tree, override the parameter keep.inbag = T in randomForest()



```
m.RF.1 <- randomForest(AHD ~ . , data=heart.df,
                       na.action=na.omit,
                       importance=T,
                       keep.inbag = T)
```



```
> inbag <- m.RF.1$inbag
> View(inbag)
> nrow(inbag)
[1] 297
```

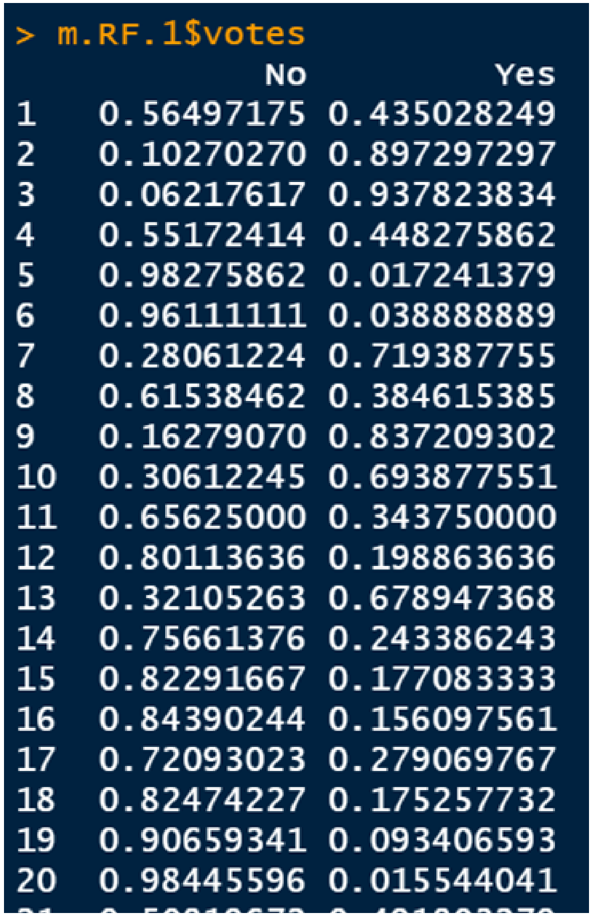


	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13
1	0	1	2	1	1	1	0	1	1	2	0	2	0
2	0	0	0	1	1	0	0	2	0	0	1	2	0
3	0	0	0	0	3	1	1	2	2	1	0	1	1
4	2	1	0	0	1	1	2	0	2	0	1	0	2

Showing 1 to 4 of 297 entries, 500 total columns

Source: Chew C.H. (2022) A.I., Analytics & Data Science, Vol. 2.

View RF vote for each case in the dataset via m.RF.1$votes



```
> m.RF.1$votes
           No         Yes
1  0.56497175 0.435028249
2  0.10270270 0.897297297
3  0.06217617 0.937823834
4  0.55172414 0.448275862
5  0.98275862 0.017241379
6  0.96111111 0.038888889
7  0.28061224 0.719387755
8  0.61538462 0.384615385
9  0.16279070 0.837209302
10 0.30612245 0.693877551
11 0.65625000 0.343750000
12 0.80113636 0.198863636
13 0.32105263 0.678947368
14 0.75661376 0.243386243
15 0.82291667 0.177083333
16 0.84390244 0.156097561
17 0.72093023 0.279069767
18 0.82474227 0.175257732
19 0.90659341 0.093406593
20 0.98445596 0.015544041
```

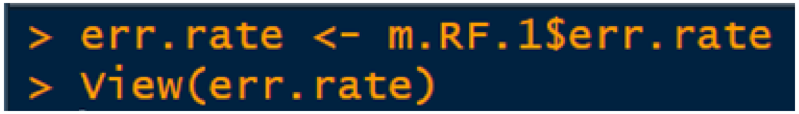
Q: Consider case 1. Does this mean 56% of the 500 trees voted AHD = No and 44% of the 500 trees voted AHD = Yes?

Ans: No. Not 500 trees. Only in those trees (approx. 1/3 of 500) for which case 1 is OOB.

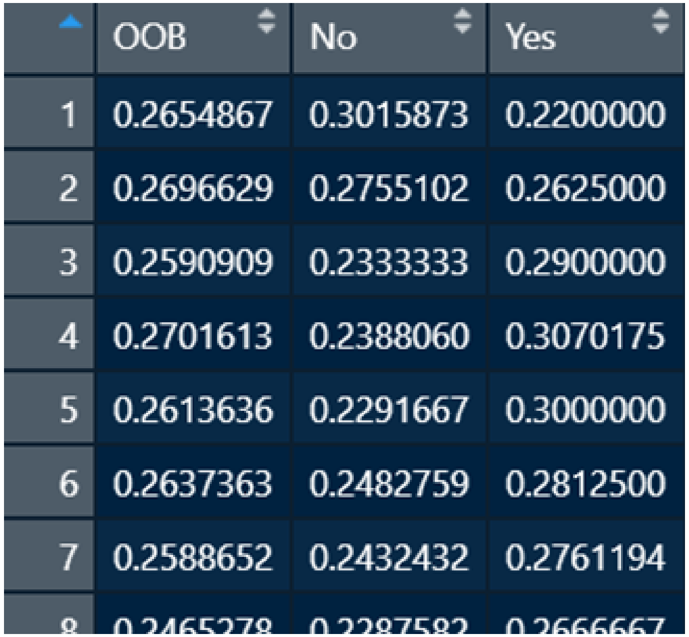
votes (classification only) a matrix with one row for each input data point and one column for each class, giving the fraction or number of (OOB) 'votes' from the random forest.

Source: Chew C.H. (2022) A.I., Analytics & Data Science, Vol. 2 & randomForest Documentation.

Caution: m.RF.1$err.rate is not the error rate at each tree



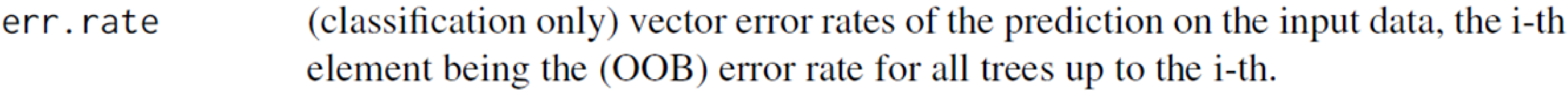
```
> err.rate <- m.RF.1$err.rate
> View(err.rate)
```



	OOB	No	Yes
1	0.2654867	0.3015873	0.2200000
2	0.2696629	0.2755102	0.2625000
3	0.2590909	0.2333333	0.2900000
4	0.2701613	0.2388060	0.3070175
5	0.2613636	0.2291667	0.3000000
6	0.2637363	0.2482759	0.2812500
7	0.2588652	0.2432432	0.2761194
8	0.2465278	0.2287582	0.2666667

Q: Consider row 4. What is the meaning of OOB = 0.27? Does this mean the 4th tree OOB error is 27%?

Ans: OOB error using the first 4 trees is 27%



`err.rate` (classification only) vector error rates of the prediction on the input data, the i-th element being the (OOB) error rate for all trees up to the i-th.

Source: Chew C.H. (2022) A.I., Analytics & Data Science, Vol. 2 & randomForest Documentation.

Python RF Implementation

- Categorical Y:
  - sklearn.ensemble.RandomForestClassifier
  - https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
- Compared to R randomForest():
  - Python used default B = 100. Please overwrite to 500.
  - Python also used default RSF size = int(sqrt(M)).

Source: Chew C.H. (2022) A.I., Analytics & Data Science, Vol. 2.

Python RF Implementation

- Continuous Y:
  - sklearn.ensemble.RandomForestRegressor
  - https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html
- Compared to R randomForest():
  - Python used default B = 100. Please overwrite to 500.
  - Python used default RSF size = M. Please overwrite to int(M/3).

Source: Chew C.H. (2022) A.I., Analytics & Data Science, Vol. 2.

Summary

- Random Forest
  - Bagging
    - 500 Bootstrap samples
    - Maximal CART per Bootstrap sample.
  - Random Subset Feature Selection
    - Default size floor(M/3) for continuous Y
    - Default size floor(sqrt(M)) for categorical Y
  - Errors calculated from OOB cases
    - Check errors stabilised before reaching ntree (default 500).
  - Random Forest has a special way to assess variable importance.

Source: Chew C.H. (2022) A.I., Analytics & Data Science, Vol. 2.