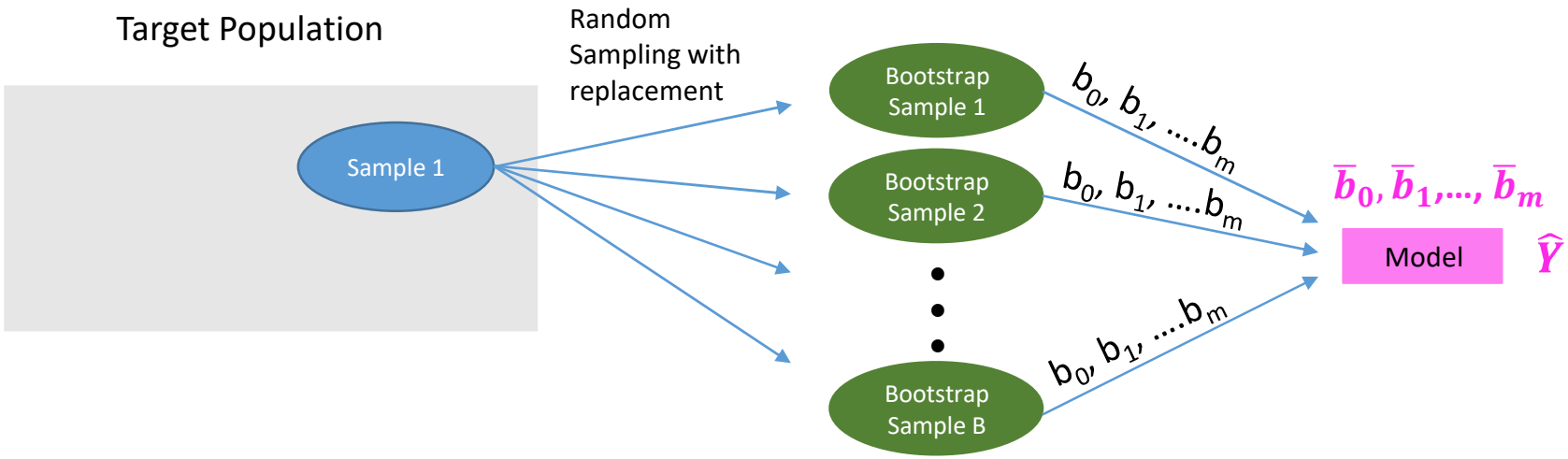
Random Forests

BC2407 Session 8

References

- Gareth James et. al. (2017) An Introduction to Statistical Learning.
  - Section 8.2.1 Bagging & 8.2.2 Random Forest, pp. 316 – 321.
  - Textbook download: https://www.statlearning.com
- Chew C.H. (2022) Artificial Intelligence, Analytics and Data Science, Vol. 2.
  - Est. Q3 2022.
- Research Papers
  - Breiman (1996a) Bagging Predictors. *Machine Learning, 24, 123-140.*
  - Breiman (1996b) Heuristics of Instability and Stabilization in Model Selection. *The Annals of Statistics, Vol. 24, No. 6.*
  - Breiman (2001) Random Forests. *Machine Learning, 45, 5–32.*

Bootstrap Design for Linear Regression Model



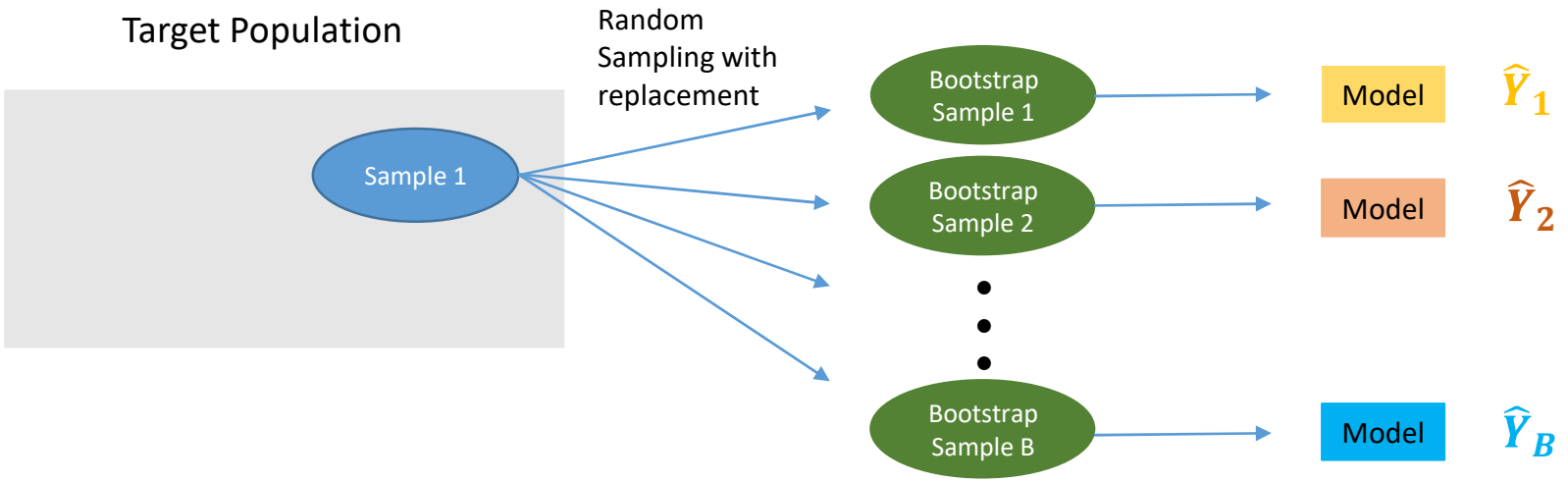
- Each bootstrap sample resulted in a different set of model coefficients b_0, b_1, ….b_m
- Take the mean of each model coefficients as the bootstrap estimate of the model coefficients for only one linear regression model.
  - For i = 0, 1,…m:

$$\bar{b}_i = \frac{\sum_{s=1}^{B} b_{i,s}}{B}$$

- Use that one linear regression model for prediction.
- Same process applies for any other type of model.

Sources: Chew C.H. (2022) AI, Analytics and Data Science, Vol. 2, Chap. 8.

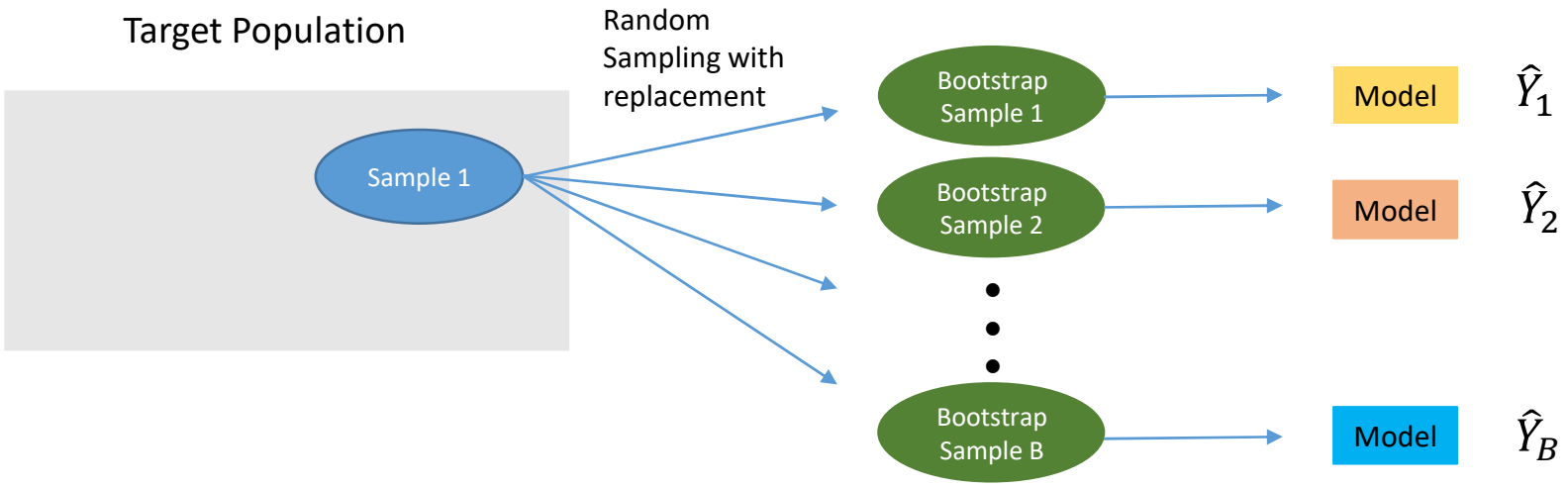
Bootstrap Aggregating (Bagging) Design



- Each bootstrap sample has the same size (n) as the original Sample 1.
  - Doubling bootstrap sample size to 2n, *"There is no improvement in accuracy." --Breiman (1996a).*
- Each of the bootstrap sample serves as a trainset for the same model type. e.g. CART.
  - Number of bootstrap samples = Number of models.
- B is chosen to be a large number. Breiman (1996a): 50 for Classification Tree, 25 for Regression Tree.
  - *"This does not mean that 50 or 25 were necessary or sufficient, but simply that they seemed reasonable. My sense of it is that fewer are required when y is numerical and more are required with an increasing number of classes." -- Breiman (1996a).*
  - *"…in practice we use a value of B sufficiently large for the error rate to have settled down." -- ISLR*

Sources: Chew C.H. (2022) AI, Analytics and Data Science, Vol. 2, Chap. 8 & Breiman (1996a)

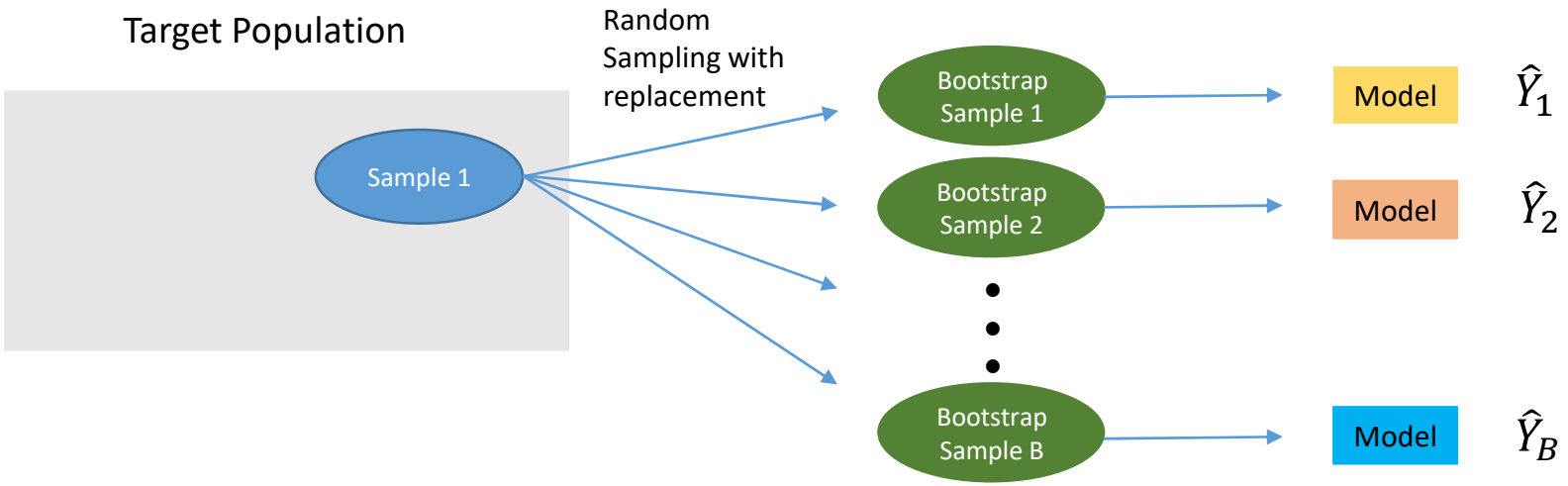
Bootstrap Aggregating (Bagging) Predicted Y



- Bagging Prediction of Y:
  - for continuous Y:
    - Take the average of $\hat{Y}_1, \hat{Y}_2, \ldots \hat{Y}_B$
  - For categorical Y:
    - Take the mode (i.e. majority vote) of $\hat{Y}_1, \hat{Y}_2, \ldots \hat{Y}_B$

Sources: Chew C.H. (2022) AI, Analytics and Data Science, Vol. 2: Chap. 8 & Breiman (1996a)

Bootstrap Aggregating (Bagging) Errors



- To get trainset error in Bagging, just take the average of the errors within the B Bootstrap samples.
- Q: How to get testset error in Bagging?
  - Breiman (1996a) did 90-10 train-test split from original sample 1 repeatedly (100 times).
  - A more efficient and natural testset procedure was subsequently devised in Breiman (2001) – OOB Error.

Sources: Chew C.H. (2020) AI, Analytics and Data Science, Vol. 2: Advanced Predictive Techniques, Chap. 8 & Breiman (1996a)

Learning Activity: Bootstrap 10 cases using Excel

Est: 10 mins.

	A
1	Case ID
2	1
3	2
4	3
5	4
6	5
7	6
8	7
9	8
10	9
11	10

- Random Forest is built on bootstrap. To check and enhance your understanding of bootstrap with a simple Excel exercise.
- **Generate 3 bootstrap samples for the 10 case IDs using Excel.**
  - Hint: rand() in excel may be useful.
- **In each bootstrap sample, which cases are Out-Of-Bag (OOB)?**
- *Note: selecting the case into bootstrap will include all the attributes for that case inside the bootstrap (i.e. the entire row of data, not just the Case ID number).*
- *Solution: bootstrap ans.xlsx*

Out-of-Bag (OOB) Error as the natural Testset Error in Bagging

- For a bootstrap sample of size n, the probability that case i in the original sample is not in a specific bootstrap sample:
  - P(case i is OOB) = $(1 - 1/n)^n$

"each bagged tree makes use of around two-thirds of the observations" -- ISLR.

- $(1 - 1/n)^n \rightarrow 1/e$ as $n \rightarrow \infty$.
  - For 3 proofs, see https://socratic.org/questions/how-do-you-find-the-limit-of-1-1-x-x-as-x-approaches-infinity *[Optional. Skip if you are not interested in the maths.]*

- Since $1/e \approx 0.367879... \approx 0.3333... = 1/3$
- Thus, as a convenient approximation, P(case i is OOB) ≈ 1/3.
- Since bootstrap samples are independent and identically distributed, approximately 1/3 of cases in original sample do not appear in bootstrap sample s and hence, those OOB cases could serve as a testset while bootstrap sample s serve as trainset.
- Average across all B bootstrap samples to get the mean OOB testset error.

Results of Bagging Classification Trees (i.e. categorical Y) in Breiman (1986)

Table 2. Misclassification Rates (%)

Data Set	$\bar{e}_S$	$\bar{e}_B$	Decrease
waveform	29.1	19.3	34%
heart	4.9	2.8	43%
breast cancer	5.9	3.7	37%
ionosphere	11.2	7.9	29%
diabetes	25.3	23.9	6%
glass	30.4	23.6	22%
soybean	8.6	6.8	21%

Table 3. Standard Errors of Misclassification

Data Set	$SE(\bar{e}_S)$	$SE(\bar{e}_B)$
waveform	.2	.1
heart	.2	.1
breast cancer	.3	.2
ionosphere	.5	.4
diabetes	.4	.4
glass	1.1	.9
soybean	.4	.3

- Testset errors substantially decreased with Bagging.
- Standard errors decreased too.

Source: Breiman (1996a)

Results of Bagging Regression Trees (i.e. continuous Y) in Breiman (1986)

Table 8. Mean Squared Test Set Error

Data Set	$\bar{e}_S$	$\bar{e}_B$	Decrease
Boston Housing	20.0	11.6	42%
Ozone	23.9	18.8	21%
Friedman #1	11.4	6.1	46%
Friedman #2	31,100	22,100	29%
Friedman #3	.0403	.0242	40%

Table 9. Standard Errors

Data Set	$SE(\bar{e}_S)$	$SE(\bar{e}_B)$
Boston Housing	1.0	.6
Ozone	.8	.6
Friedman #1	.10	.06
Friedman #2	300	100
Friedman #3	.0005	.0003

- Testset errors substantially decreased with Bagging.
- Standard errors decreased too.

Source: Breiman (1996a)

But Bagging results are not as significant for Bagging KNN models

Table 11. Misclassification Rates for Nearest Neighbor

Data Set	$\bar{e}_S$	$\bar{e}_B$
waveform	26.1	26.1
heart	5.1	5.1
breast cancer	4.4	4.4
ionosphere	36.5	36.5
diabetes	29.3	29.3
glass	30.1	30.1

- Testset errors are almost the same with or without Bagging.
- KNN models are relatively stable models.

Source: Breiman (1996a)

Stability of the model

- Refers to the stability of the procedure in specifying the model.
  - Eg: Getting the model coefficients in linear reg via min. SSE.
- Improvement will occur [in Bagging] for unstable procedures where a small change in training set can result in large changes in the model.
- Examples of unstable procedures:
  - Neural Network
  - CART
  - MARS
  - Subset selection in Linear Regression
- Examples of stable procedures:
  - KNN
  - Ridge Regression

Consequences of Using Unstable Models

- Testset Error has “a large negative bias” --- Breiman (1996b)
  - i.e. testset error is much lower than reality.
- Some methods for estimating future prediction error does not work well
  - LOOCV is worse than 10-fold CV.
- Fortunately, unstable procedures can be stabilized
  - Via Bagging.

2 Reasons Why Aggregating Different Bootstrap samples (Bagging) Is Not Useful

- What if different bootstrap samples result in the same or almost the same model?
  - Might as well just use one [original] sample.
  - Stable models
    - Example: KNN results in Breiman (1996a) Table 11.
    - Linear Regression with all Xs is more stable than Linear reg with selected Xs.
  - Unstable models but with dominant predictor X.
    - Example: Predict Y = BC2407 Grade, X1 = Hours studying, X2 = AB1202 Statistics Exam Marks, X3 = BC2407 CBA marks.
      - X3 is dominant in predicting Y.
- How to improve Bagging results?
  - How to infuse controlled instability even in presence of dominant X?
    - Maintain accuracy while increasing variance.
    - Hint: Stability results in Linear Regression.
    - Ans: Subset Selection (Different Choice of Xs)

Random Forest = Bagging + Random Subset Feature

Random Forest Enhances Bagging CART

- Random Forest
  1. Bagging maximal CART [maintain strength of each CART]
     - Do not prune any tree. Grow to the max.
  2. Random Subset Feature at each split. [reduce correlation betw CARTs]
- B = 500
- Random Subset Feature (RSF):
  - Randomly select a subset of X variables for consideration at each split, and select the best X in the subset.
  - Categorical Y:
    - RSF size = int(sqrt(M)) [from ISLR textbook] or,
    - RSF size = int($\log_2$(M) + 1) [from Breiman(2001)]
    - Not much difference if M (number of X variables) are small. [See excel file: Values of RSF.]
  - Continuous Y:
    - RSF size = int(M/3)
    - Correlation is less sensitive to RSF. "Correlation increases quite slowly as the number of features used increases."
  - Effectively:
    - Infuse instability by randomly ignoring some Xs, esp if dominant X not included.
    - De-correlates trees in the forest.
    - Use Bagging to stabilize results.

Random Forest Results in Breiman (2001)

Selection: Best of RSF = 1 and RSF = int($\log_2$(M)+1)

Table 2. Test set errors (%).

Data set	Adaboost	Selection	Forest-RI single input	One tree
Glass	22.0	20.6	21.2	36.9
Breast cancer	3.2	2.9	2.7	6.3
Diabetes	26.6	24.2	24.3	33.1
Sonar	15.6	15.9	18.0	31.7
Vowel	4.1	3.4	3.3	30.4
Ionosphere	6.4	7.1	7.5	12.7
Vehicle	23.2	25.8	26.4	33.1
German credit	23.5	24.4	26.2	33.3
Image	1.6	2.1	2.7	6.4
Ecoli	14.8	12.8	13.0	24.5
Votes	4.8	4.1	4.6	7.4
Liver	30.7	25.1	24.7	40.6
Letters	3.4	3.5	4.7	19.8
Sat-images	8.8	8.6	10.5	17.2
Zip-code	6.2	6.3	7.8	20.6
Waveform	17.8	17.2	17.3	34.0
Twonorm	4.9	3.9	3.9	24.7
Threenorm	18.8	17.5	17.5	38.4
Ringnorm	6.9	4.9	4.9	25.7

Source: Breiman (2001)

Dataset: Heart.csv

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

- 303 cases
- 13 Xs and 1 categorical Y (AHD)
- Data Dictionary: Heart Data Dictionary.txt
- 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

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)
```

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

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

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, ...)
```

i.e. B

i.e. RSF size

Min cases in terminal node

Override to T to get perturbation based measure of variable importance.

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%.
- Q1: How are the confusion matrix results determined?
  - Ans: From OOB data and majority rule.
- Q2: Why did the confusion matrix contain only 297 cases when the dataset has 303 cases?
  - Ans: 6 cases has missing values and were omitted in na.action= na.omit
- Q3: Verify if confusion matrix rows or columns represent actuals? [10 mins]
  - Ans: Check the distribution of Y in the data and compare against C.M.
- Q4: What is the meaning of the two class errors (0.15, 0.2189781)?
  - Ans: False Positive Rate and False Negative Rate.

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

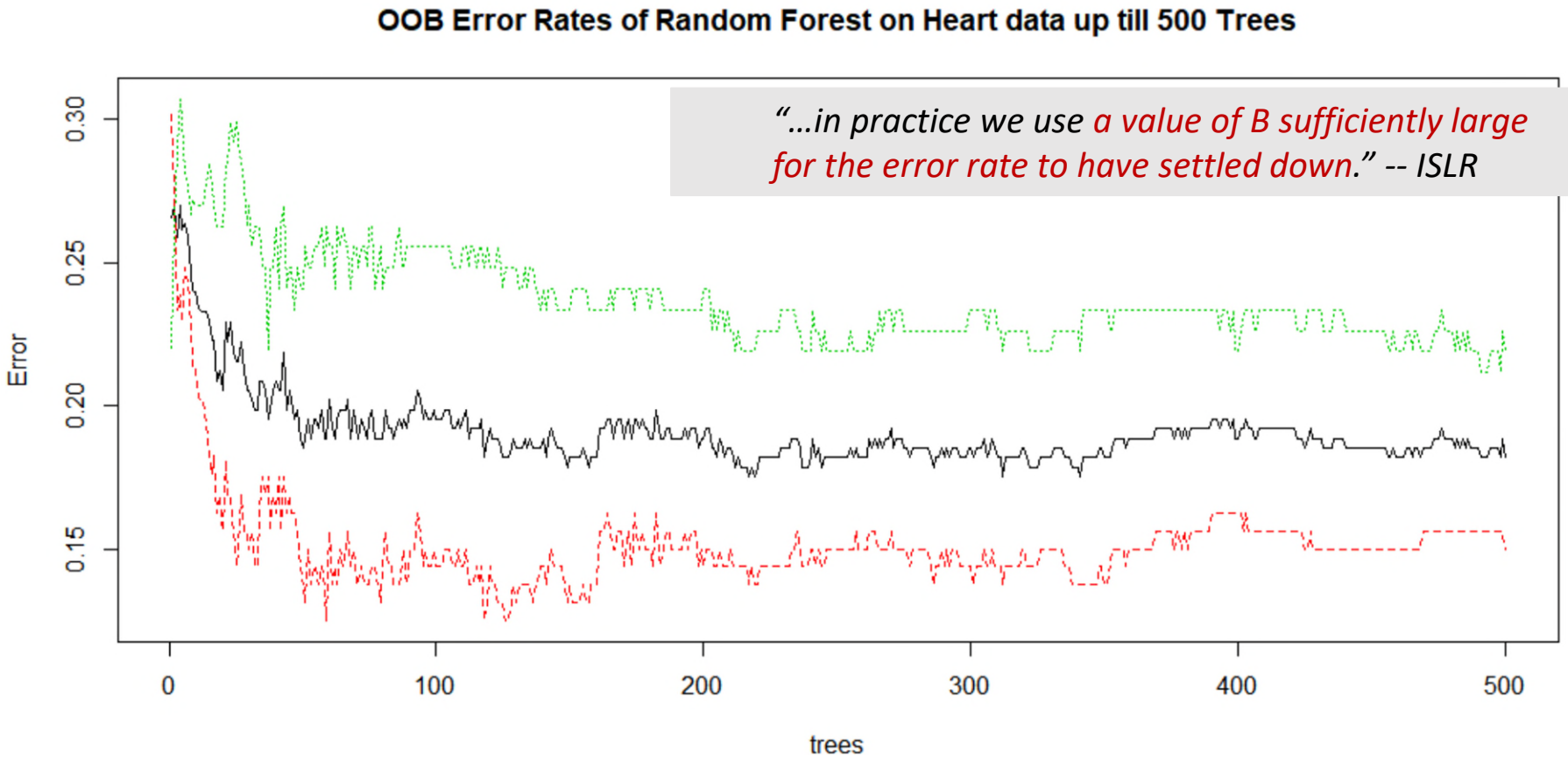
Verify if RF confusion matrix rows represent actual Y or model predicted Y

```
> summary(na.omit(heart.df)$AHD)
 No Yes
160 137
```

```
> m.RF.1$confusion
     No Yes class.error
No  136  24   0.1500000
Yes  30 107   0.2189781
```

- Summary() shows the distribution of AHD in the missing values removed data: 160 actual No, 137 actual Yes. Total 297 cases.
- The rows in the confusion matrix total 160 and 137 respectively. Hence, rows represent actual Y values and columns represent model predicted Y values.
- False Positive Rate = 24/(136 + 24) = 0.15
- False Negative Rate = 30/(30 + 107) = 0.2189781

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

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.

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, …

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

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

```
> m.RF.1$votes
             No        Yes
1   0.56497175 0.43502825
2   0.10270270 0.89729730
3   0.06217617 0.93782383
4   0.55172414 0.44827586
5   0.98275862 0.01724138
6   0.96111111 0.03888889
7   0.28061224 0.71938776
8   0.61538462 0.38461538
9   0.16279070 0.83720930
10  0.30612245 0.69387755
11  0.65625000 0.34375000
12  0.80113636 0.19886364
13  0.32105263 0.67894737
14  0.75661376 0.24338624
15  0.82291667 0.17708333
16  0.84390244 0.15609756
17  0.72093023 0.27906977
18  0.82474227 0.17525773
```

```
> 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
```

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?

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.

Learning Activity: Random Forest on Heart data

Est. Duration: 40 mins

Instructor solution in RF Heart2.R

- Run RF Heart1.R.
- Execute Random Forest with different settings on Heart.csv to predict AHD and save the respective OOB error in a table. How does the default settings for B and RSF size fare in terms of OOB error?
    1. B = 25, RSF size = 1
    2. B = 25, RSF size = int(sqrt(m)) = 3
    3. B = 25, RSF size = m = 13
    4. B = 100, RSF size = 1
    5. B = 100, RSF size = int(sqrt(m)) = 3
    6. B = 100, RSF size = m = 13
    7. B = 500, RSF size = 1
    8. B = 500, RSF size = int(sqrt(m)) = 3
    9. B = 500, RSF size = m = 13
- Based on the Random Forest model with the default settings, which variables are important in predicting AHD? Explain how "variable importance" is determined.

Key Findings in Learning Activity

set.seed(1)

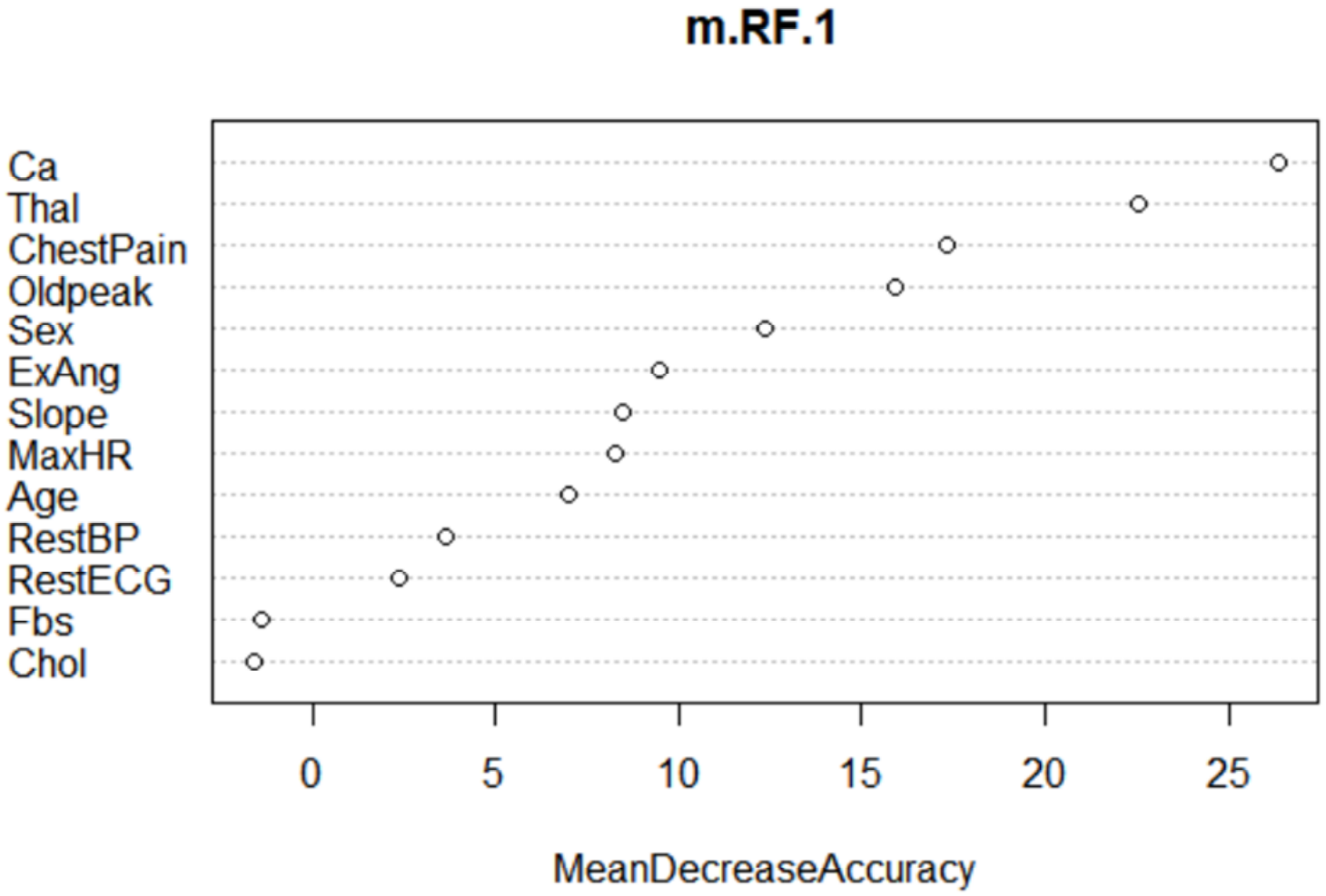
	B	RSF	OOB.error
7	500	1	0.1649832
8	500	3	0.1683502
4	100	1	0.1784512
5	100	3	0.1885522
2	25	3	0.1919192
6	100	13	0.1952862
9	500	13	0.2053872
1	25	1	0.2188552
3	25	13	0.2255892

set.seed(2020)

	B	RSF	OOB.error
5	100	3	0.1683502
7	500	1	0.1717172
8	500	3	0.1717172
4	100	1	0.1818182
2	25	3	0.1851852
9	500	13	0.1851852
3	25	13	0.1885522
6	100	13	0.1919192
1	25	1	0.2154882

- Trying different seeds, default settings of B and RSF size works quite well.

Key Findings in Learning Activity – Variable Impt



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

X Variable Importance is based on Mean Decrease in Accuracy using Permutation approach

- To understand the importance of each X in a forest of trees.
- Mean Decrease Accuracy is computed using Permutation
  - Breaks the relationship (if any) between the specific X and Y.
  - What is the increase in error if the randomly permutated X is used instead of the original X?
    - If the error increase a lot, then X is important.
    - Else, X is not important.
    - Example:
      - Y = Overall Marks in BC2407 course.
      - X_1 = random number
      - X_2 = AB1202 Statistics overall marks [pre-requisite course]
      - X_3 = Marks in BC2407 Project

na.action = na.roughfix vs rfImpute()

- Normally used if there are many unresolved missing values.
- na.roughfix will just set all categorical NAs to mode of that column and continuous NAs to median of that column.
- rfImpute() will impute missing values based on proximities of cases
  - "based on the frequency that pairs of data points are in the same terminal nodes" in the trees.
  - The more stable, many trees version of surrogate in CART.
  - The output of rfImpute() is a dataset with all missing values imputed.

Usage

```
## Default S3 method:
rfImpute(x, y, iter=5, ntree=300, ...)
## S3 method for class 'formula'
rfImpute(x, data, ..., subset)
```

See https://www.rdocumentation.org/packages/randomForest/versions/4.6-14/topics/rfImpute

Learning Activity: Random Forest vs MARS on Resale Flat Price

Est. Duration: 30 mins

- Dataset: resale-flat-prices-2019.csv [from MARS seminar]
- Run flatprice-RF.R
  - set.seed(2) and do 70-30 train-test split.
  - Includes data prep.
- Construct the following models and get respective testset RMSE:
  1. MARS (degree 2) [already done in Rscript]
  2. Random Forest (default settings for B & mtry)
- Which variables are important?

- *Instructor solution in flatprice-RF solution.R*

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 override to 500.
  - Python used same default RSF size = sqrt(num of X variables).
  - Used Mean Gini Increase as measure of variable importance. Permutation based approach not avail within this library. Need to import another python library (e.g. rfpimp).

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 override to 500.
  - Python used default RSF size = num of X variables. Please override to int(num of X variables/3).
  - Permutation based approach for assessing variable importance not avail within this library. Need to import another python library (e.g. rfpimp).

Summary

- Random Forest
  - Bagging
    - Bootstrap samples
    - One maximal CART per Bootstrap sample.
  - Random Subset Feature Selection
    - Default size int(M/3) for continuous Y
    - Default size int(sqrt(M)) for categorical Y
  - Errors calculated from OOB cases
    - Check errors stabilised before reaching ntree (default 500).
  - Permutation based approach for assessing variable importance