Bagging
and
Random Forests

The Bootstrap Sample and Bagging

• • •

Simple ideas to improve any model via ensemble

Bootstrap Samples

- Random samples of your data *with replacement* that are the same size as original data.
- Some observations will not be sampled. These are called *out-of-bag observations*

Example: **Suppose you have 10 observations, labeled 1-10**

Bootstrap Sample Number	Training Observations	Out-of-Bag Observations
1	{1,3,2,8,3,6,4,2,8,7}	{5,9,10}
2	{9,1,10,9,7,6,5,9,2,6}	{3,4,8}
3	{8,10,5,3,8,9,2,3,7,6}	{1,4}

(Efron 1983) (Efron and Tibshirani 1986)

Bootstrap Samples

- Can be proven that a bootstrap sample will contain approximately 63% of the observations.
- The sample size is the same as the original data as some observations are repeated.
- Some observations left out of the sample (~37% out-of-bag)

- Utility:
  - Simulation
  - **Create Ensemble Models using different training sets (Bagging)**

Bagging

(Bootstrap Aggregating)

- Let k be the number of bootstrap samples
- For each bootstrap sample, create a classifier using that sample as training data
    - Results in k different models
- Ensemble those classifiers
    - A test instance is assigned to the class that received the highest number of votes.

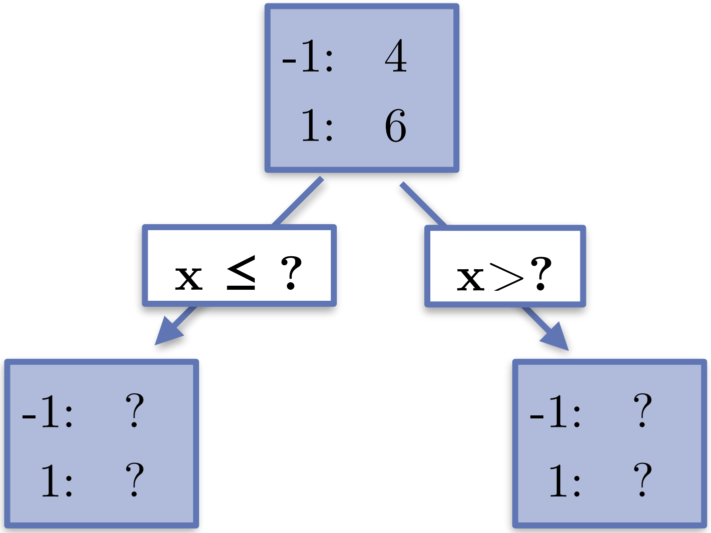
Bagging Example

input variable

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
y	1	1	1	-1	-1	-1	-1	1	1	1

target

- 10 observations in original dataset
- Suppose we build a decision tree with only 1 split.



A tree with one split called a **decision stump**

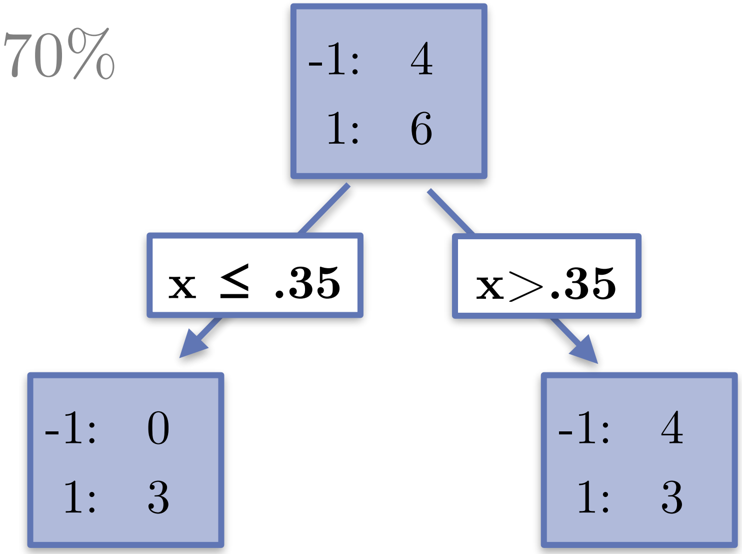
Bagging Example

input variable

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
y	1	1	1	-1	-1	-1	-1	1	1	1

target

- 10 observations in original dataset
- Suppose we build a decision tree with only 1 split.
- The best accuracy we can get is 70%
  - Split at x=0.35
  - Split at x=0.75



-1: 4
1: 6

x ≤ .35

x>.35

-1: 0
1: 3

-1: 4
1: 3

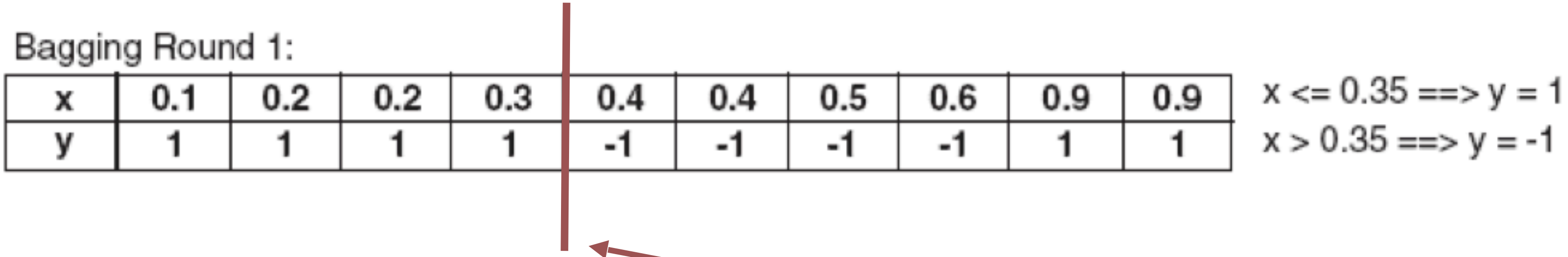
Bagging Example

Let's see how bagging might improve this model:

1. Take 10 Bootstrap samples from this dataset.
2. Build a decision stump for each sample.
3. Aggregate these rules into a voting ensemble.
4. Test the performance of the voting ensemble on the whole dataset.

Bagging Example

Classifier 1



Bagging Round 1:

x	0.1	0.2	0.2	0.3	0.4	0.4	0.5	0.6	0.9	0.9
y	1	1	1	1	-1	-1	-1	-1	1	1

x <= 0.35 ==> y = 1

x > 0.35 ==> y = -1

Best decision stump splits at x=0.35

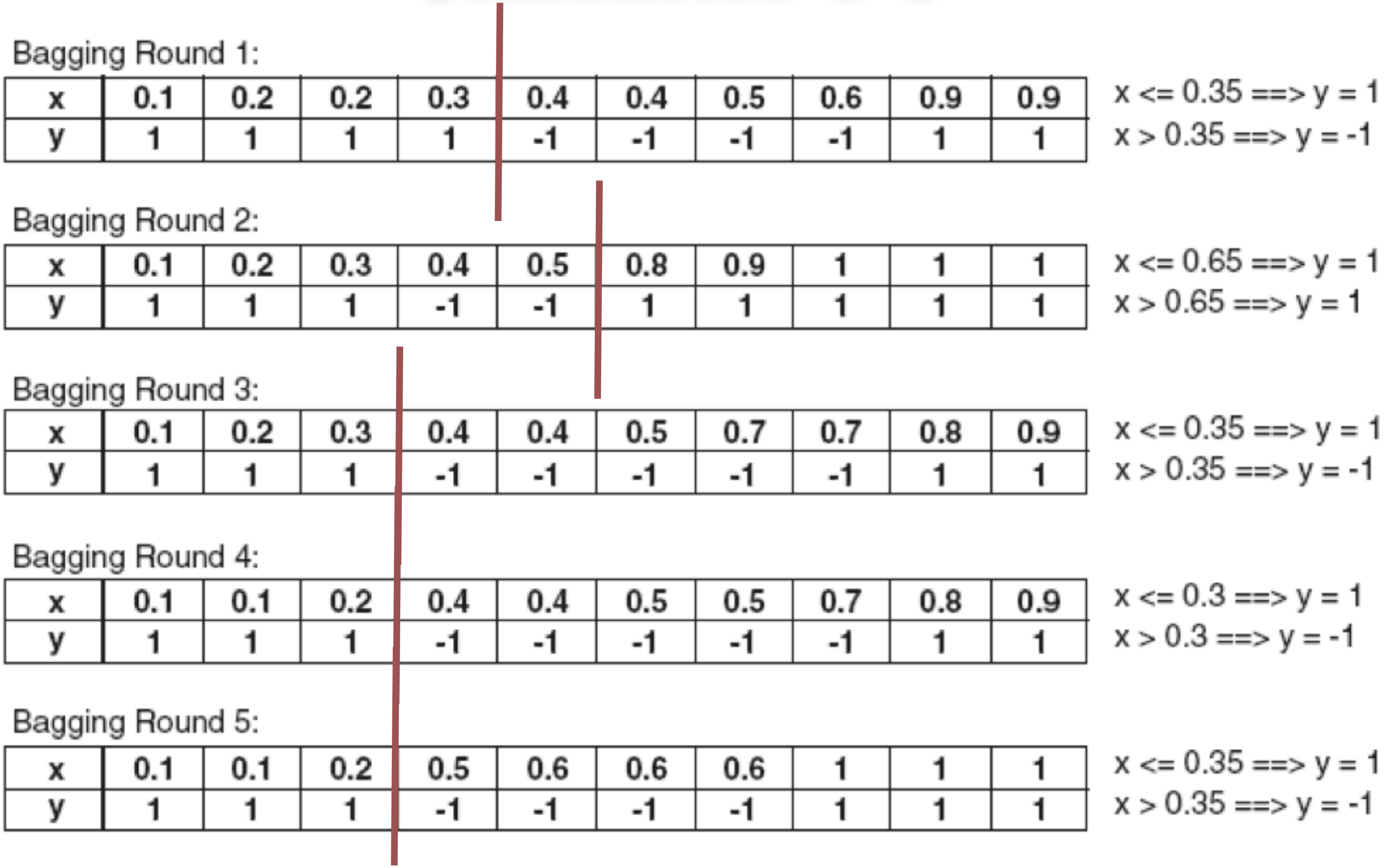
First bootstrap sample:

Some observations chosen multiple times.

Some not chosen.

Bagging Example

Classifiers 1-5



Bagging Round 1:

x	0.1	0.2	0.2	0.3	0.4	0.4	0.5	0.6	0.9	0.9
y	1	1	1	1	-1	-1	-1	-1	1	1

x <= 0.35 ==> y = 1
x > 0.35 ==> y = -1

Bagging Round 2:

x	0.1	0.2	0.3	0.4	0.5	0.8	0.9	1	1	1
y	1	1	1	-1	-1	1	1	1	1	1

x <= 0.65 ==> y = 1
x > 0.65 ==> y = 1

Bagging Round 3:

x	0.1	0.2	0.3	0.4	0.4	0.5	0.7	0.7	0.8	0.9
y	1	1	1	-1	-1	-1	-1	-1	1	1

x <= 0.35 ==> y = 1
x > 0.35 ==> y = -1

Bagging Round 4:

x	0.1	0.1	0.2	0.4	0.4	0.5	0.5	0.7	0.8	0.9
y	1	1	1	-1	-1	-1	-1	-1	1	1

x <= 0.3 ==> y = 1
x > 0.3 ==> y = -1

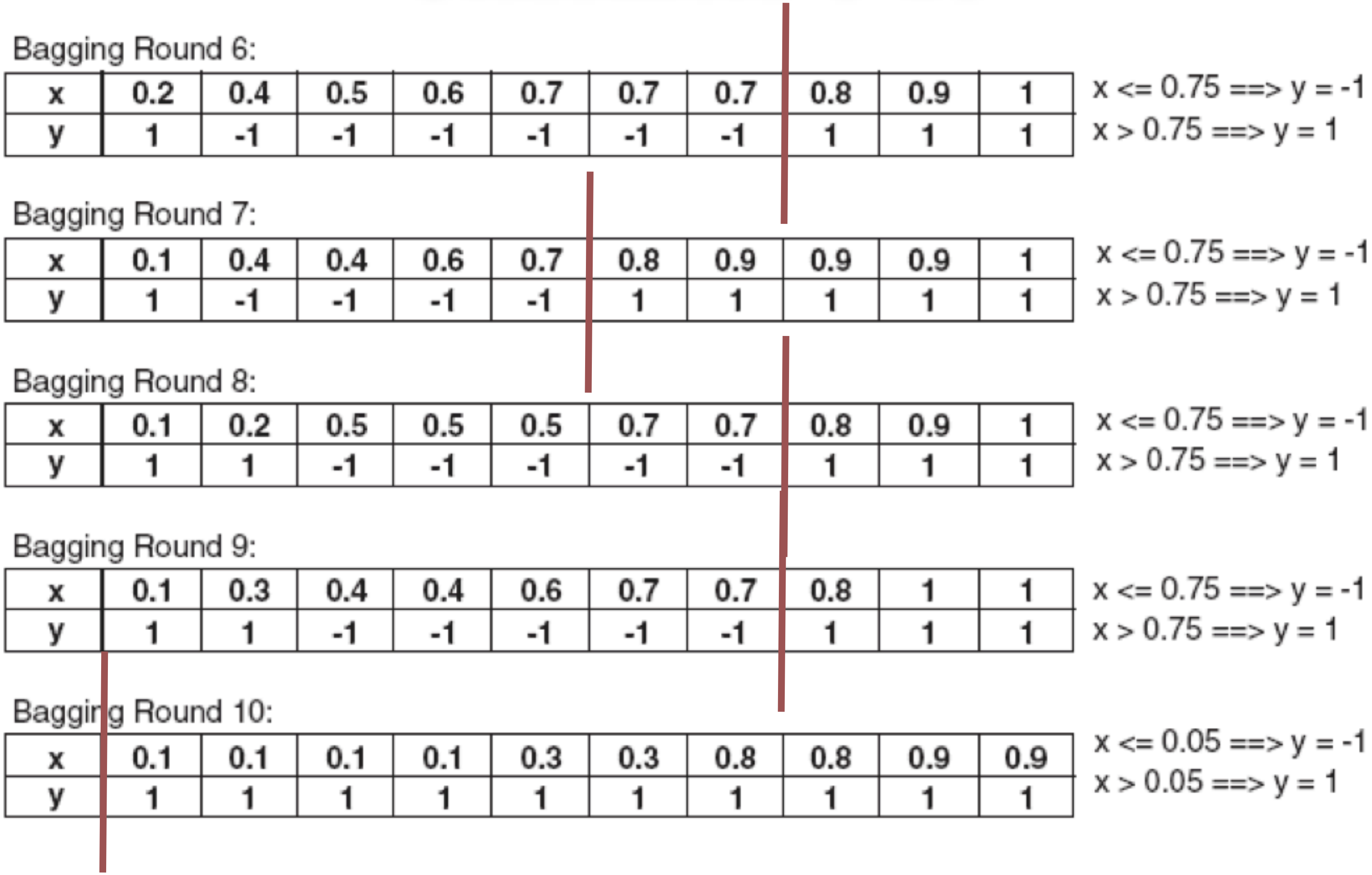
Bagging Round 5:

x	0.1	0.1	0.2	0.5	0.6	0.6	0.6	1	1	1
y	1	1	1	-1	-1	-1	-1	1	1	1

x <= 0.35 ==> y = 1
x > 0.35 ==> y = -1

Bagging Example

Classifiers 6-10



Bagging Round 6:

x	0.2	0.4	0.5	0.6	0.7	0.7	0.7	0.8	0.9	1
y	1	-1	-1	-1	-1	-1	-1	1	1	1

x <= 0.75 ==> y = -1
x > 0.75 ==> y = 1

Bagging Round 7:

x	0.1	0.4	0.4	0.6	0.7	0.8	0.9	0.9	0.9	1
y	1	-1	-1	-1	-1	1	1	1	1	1

x <= 0.75 ==> y = -1
x > 0.75 ==> y = 1

Bagging Round 8:

x	0.1	0.2	0.5	0.5	0.5	0.7	0.7	0.8	0.9	1
y	1	1	-1	-1	-1	-1	-1	1	1	1

x <= 0.75 ==> y = -1
x > 0.75 ==> y = 1

Bagging Round 9:

x	0.1	0.3	0.4	0.4	0.6	0.7	0.7	0.8	1	1
y	1	1	-1	-1	-1	-1	-1	1	1	1

x <= 0.75 ==> y = -1
x > 0.75 ==> y = 1

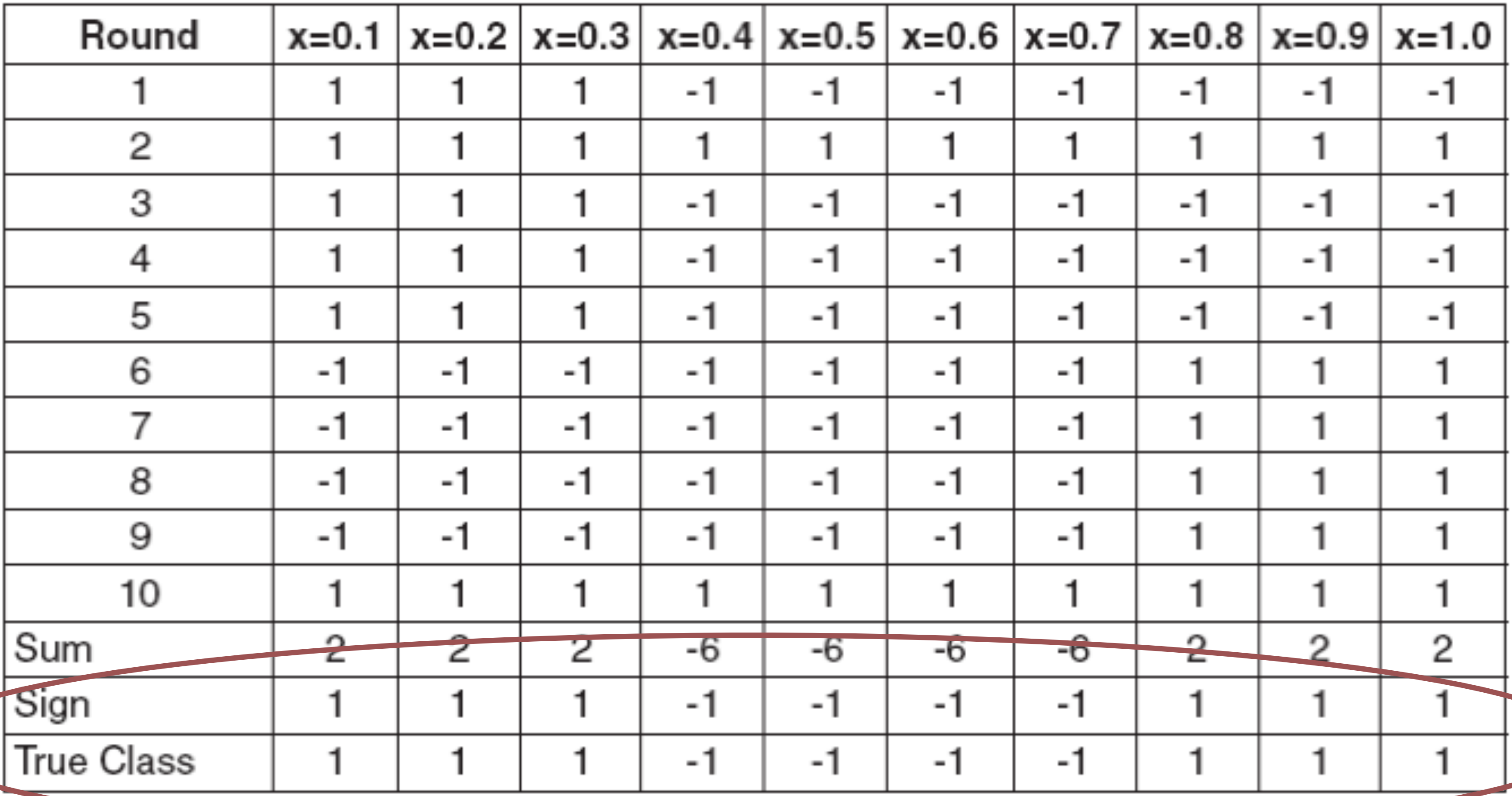
Bagging Round 10:

x	0.1	0.1	0.1	0.1	0.3	0.3	0.8	0.8	0.9	0.9
y	1	1	1	1	1	1	1	1	1	1

x <= 0.05 ==> y = -1
x > 0.05 ==> y = 1

Bagging Example

Predictions from each Classifier



Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
1	1	1	1	-1	-1	-1	-1	-1	-1	-1
2	1	1	1	1	1	1	1	1	1	1
3	1	1	1	-1	-1	-1	-1	-1	-1	-1
4	1	1	1	-1	-1	-1	-1	-1	-1	-1
5	1	1	1	-1	-1	-1	-1	-1	-1	-1
6	-1	-1	-1	-1	-1	-1	-1	1	1	1
7	-1	-1	-1	-1	-1	-1	-1	1	1	1
8	-1	-1	-1	-1	-1	-1	-1	1	1	1
9	-1	-1	-1	-1	-1	-1	-1	1	1	1
10	1	1	1	1	1	1	1	1	1	1
Sum	2	2	2	-6	-6	-6	-6	2	2	2
Sign	1	1	1	-1	-1	-1	-1	1	1	1
True Class	1	1	1	-1	-1	-1	-1	1	1	1

Ensemble Classifier has 100% Accuracy

Bagging Summary

- Improves generalization error on models with high variance
- If base classifier is stable (not suffering from high variance), bagging can actually make it worse
- Bagging does not focus on any particular observations in the training data (unlike boosting)

Random Forests

Tin Kam Ho (1995, 1998)

Leo Breiman (2001)

Random Forests

- **Random Forests are ensembles of decision trees** similar in spirit to the one we just saw

- Ensembles of decision trees **work best when their predictions are not correlated** – i.e. when they each find different patterns in the data

- **Problem:** Bagging tends to create correlated trees
- **Random Forests' Solution:** (a) Randomly subset features considered for each split. (b) Use unpruned decision trees in the ensemble.

Random Forests

- A collection of unpruned decision or regression trees.
- Each tree is built on a bootstrap sample of the data ***and*** a subset of features are considered at each split.
  - The number of features considered for each split is a parameter called $mtry$.
  - Brieman (2001) suggests $mtry = \sqrt{p}$ where p is the number of features
  - I'd suggest setting $mtry$ equal to 5-10 values evenly spaced between 2 and p and choosing the parameter by validation
  - Overall, the model is relatively insensitive to values for $mtry$.
- The results from the trees are ensembled into one voting classifier.

Random Forests Summary

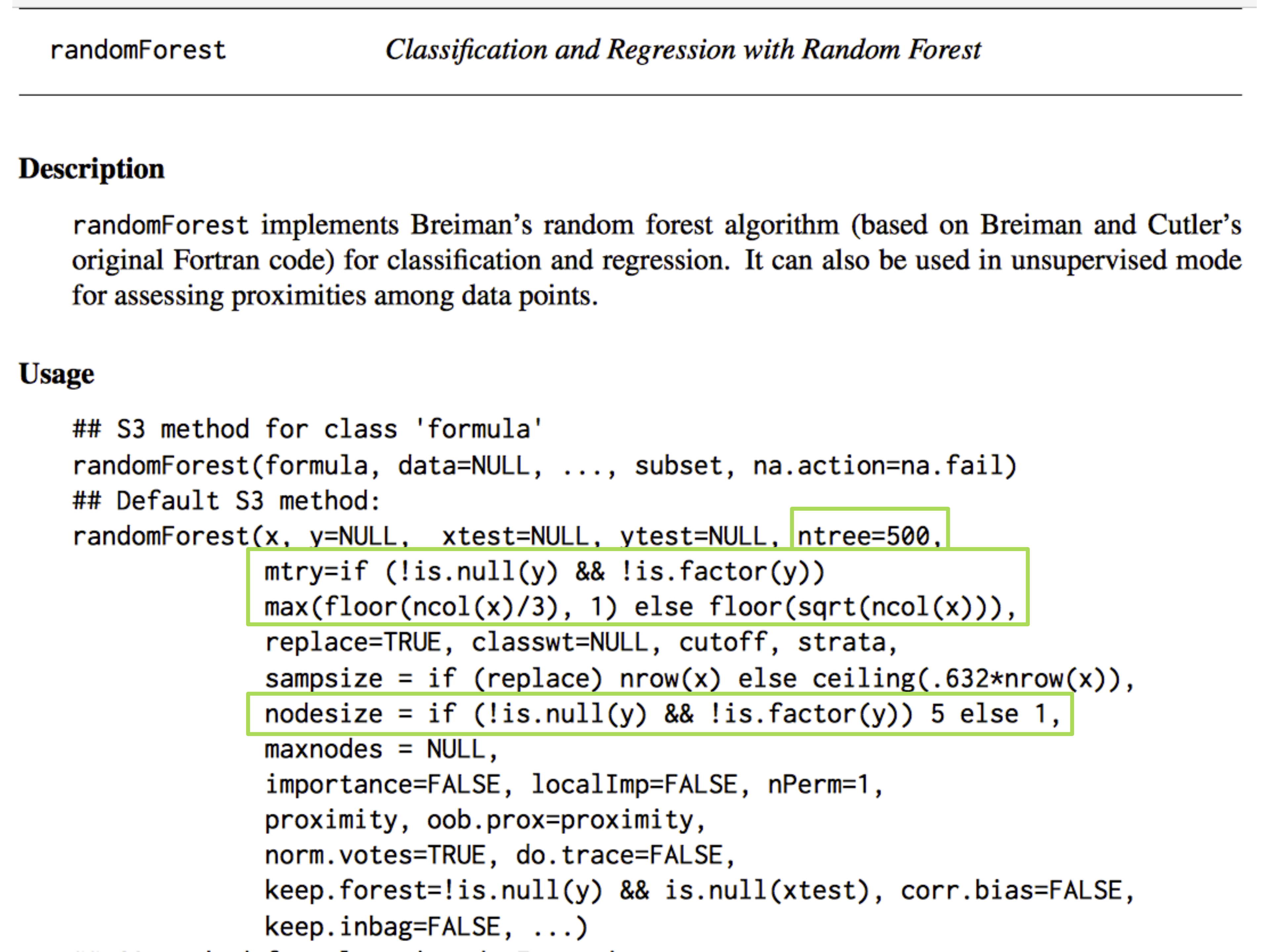
Advantages

- Computationally Fast – can handle thousands of input variables
- Trees can be trained simultaneously
- Exceptional Classifiers – one of most accurate available
- Provide information on variable importance for the purposes of feature selection
- Can effectively handle missing data (depends on implementation)

Disadvantages

- No interpretability in final model aside from variable importance
- Prone to overfitting
- Tuning parameters like the number of trees, the depth of each tree, the percentage of variables passed to each tree

RandomForest Package in R



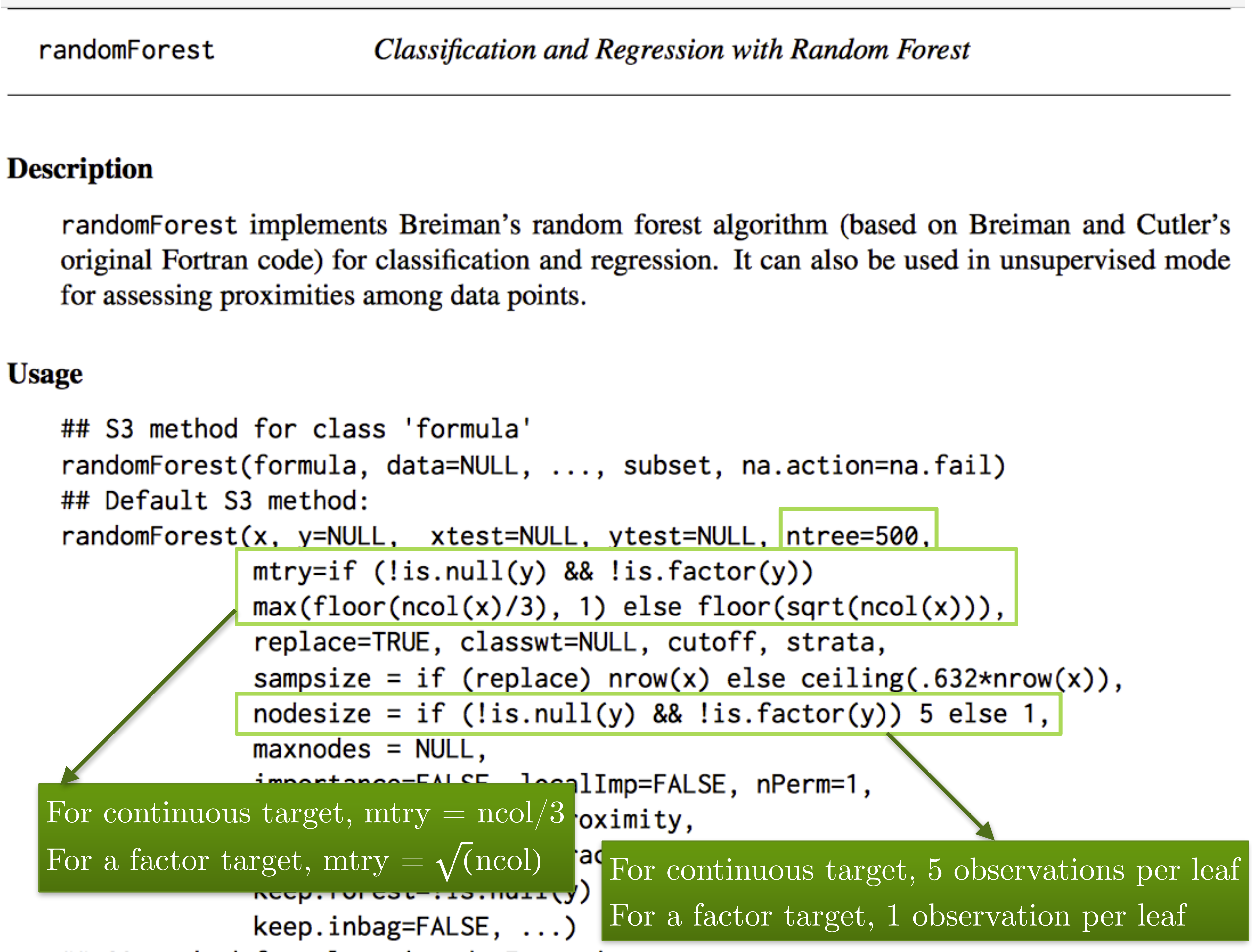
randomForest *Classification and Regression with Random Forest*

Description

randomForest implements Breiman's random forest algorithm (based on Breiman and Cutler's original Fortran code) for classification and regression. It can also be used in unsupervised mode for assessing proximities among data points.

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, ...)
```



randomForest *Classification and Regression with Random Forest*

Description

randomForest implements Breiman's random forest algorithm (based on Breiman and Cutler's original Fortran code) for classification and regression. It can also be used in unsupervised mode for assessing proximities among data points.

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,
                                 oximity,
                                 ac
             keep.forest=!is.null(y)
             keep.inbag=FALSE, ...)
```

For continuous target, mtry $=$ ncol$/3$

For a factor target, mtry $= \sqrt{}$(ncol)

For continuous target, 5 observations per leaf

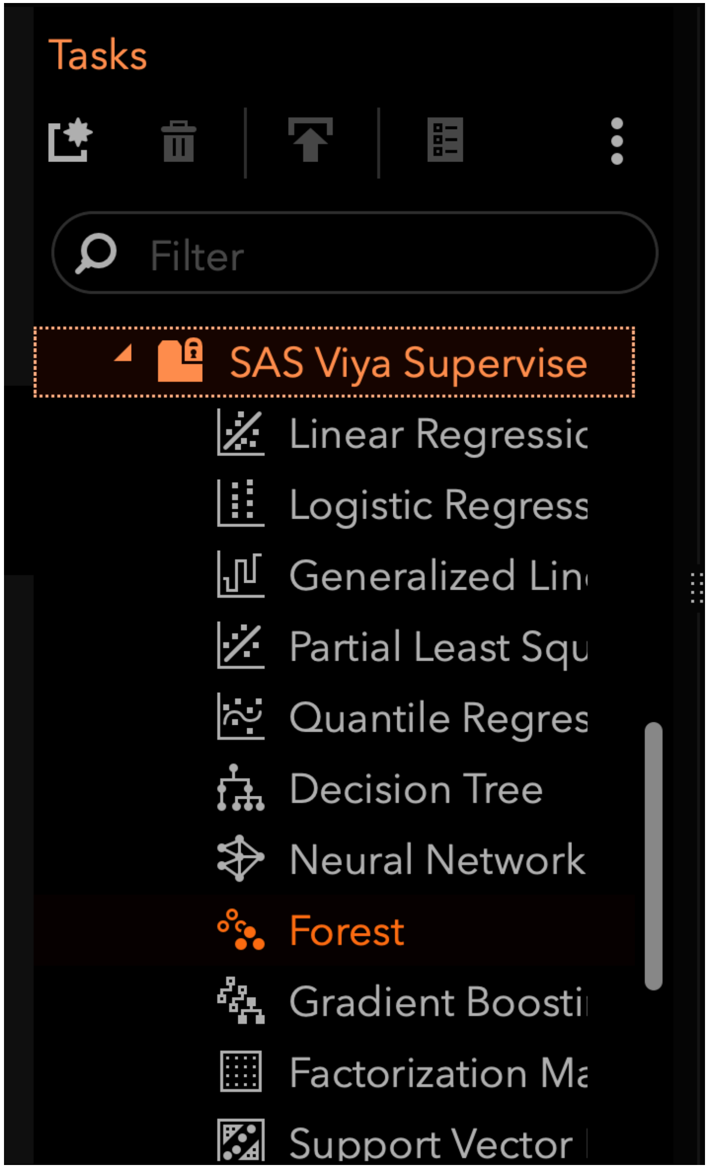
For a factor target, 1 observation per leaf

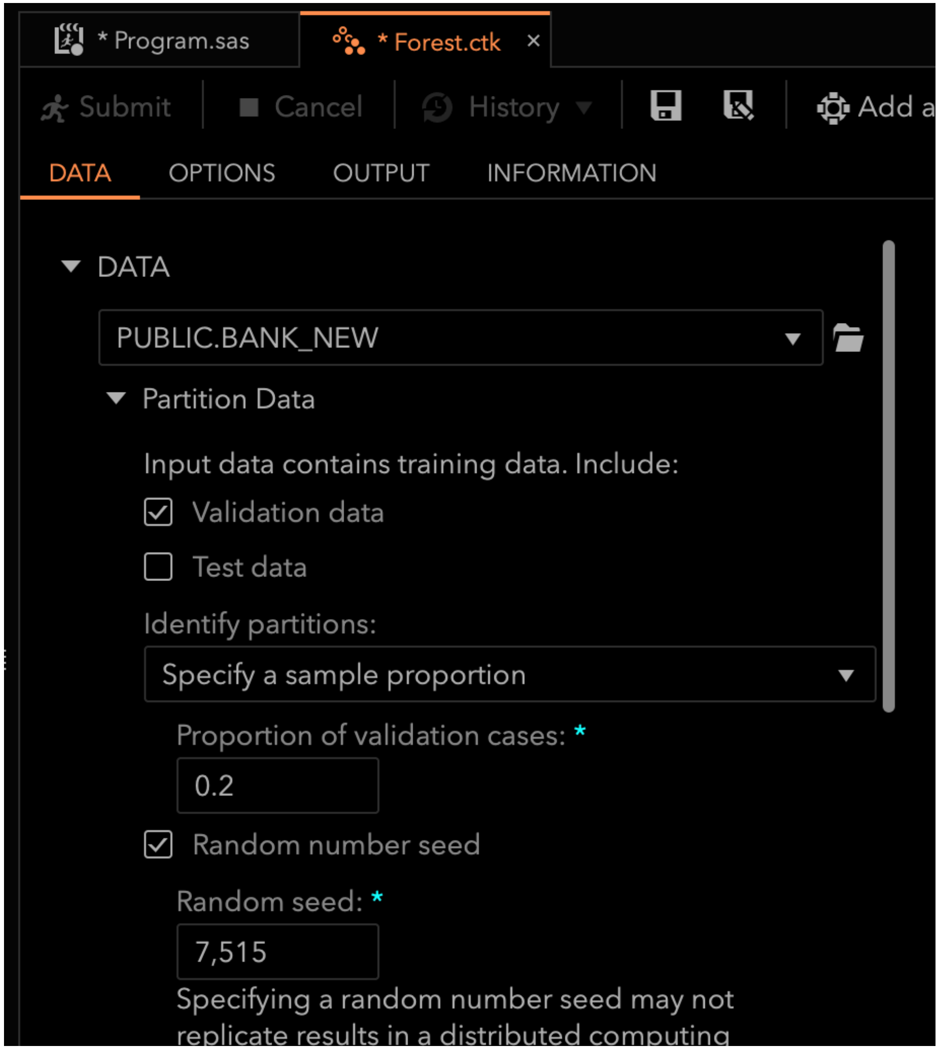
Viya Demo

• • •

BankData from Data Mining HW2

SAS Studio Tasks





DATA

PUBLIC.BANK_NEW

Partition Data

Input data contains training data. Include:

☑ Validation data

□ Test data

Identify partitions:

Specify a sample proportion

Proportion of validation cases: *

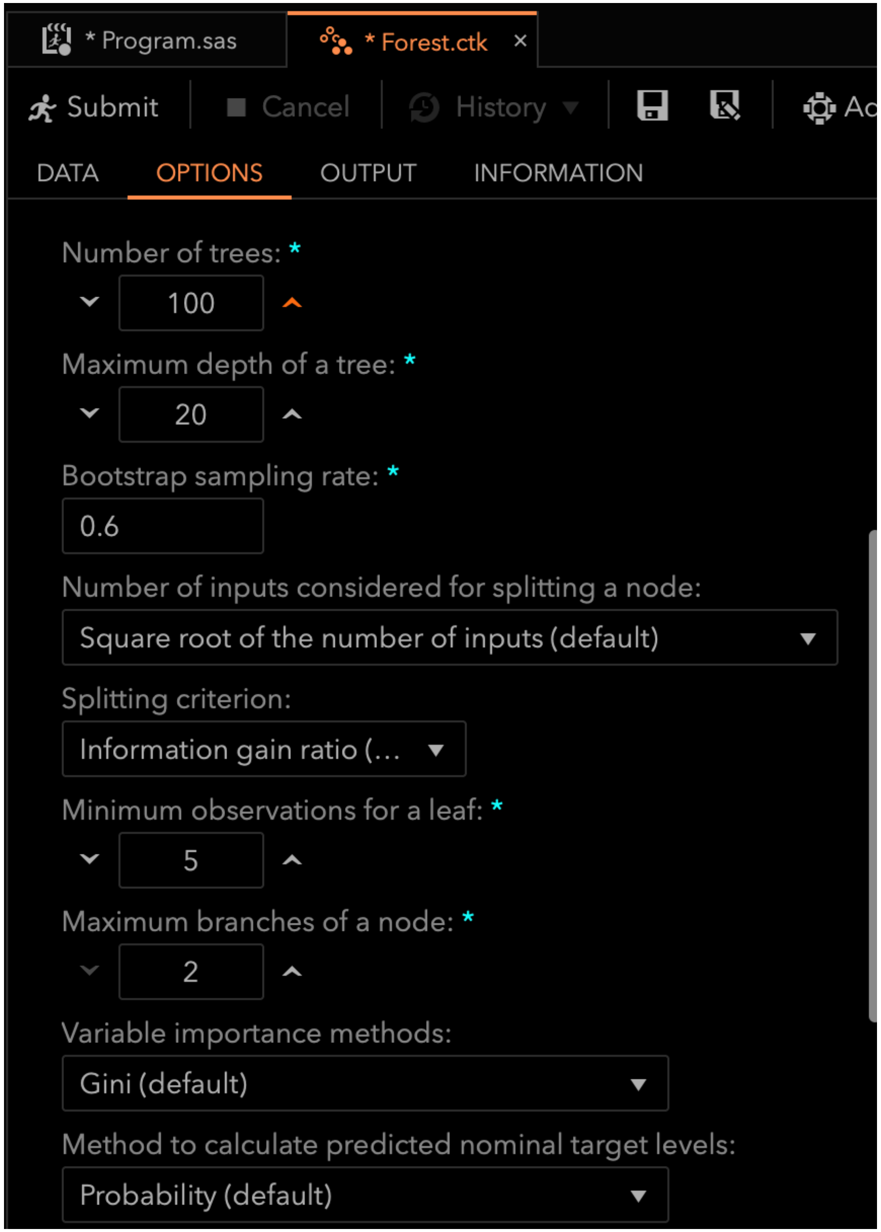
0.2

☑ Random number seed

Random seed: *

7,515

Specifying a random number seed may not replicate results in a distributed computing



Number of trees: *
100

Maximum depth of a tree: *
20

Bootstrap sampling rate: *
0.6

Number of inputs considered for splitting a node:
Square root of the number of inputs (default) ▼

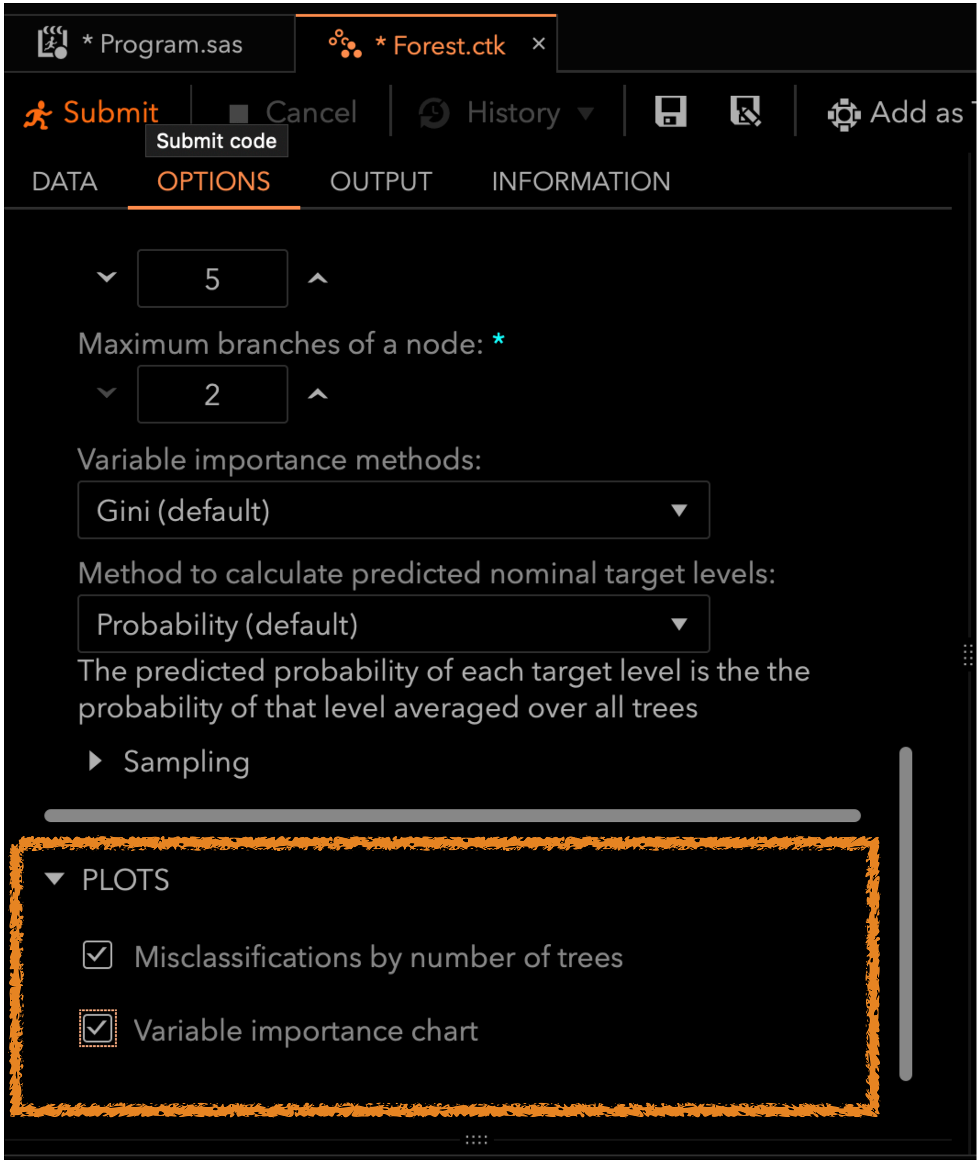
Splitting criterion:
Information gain ratio (… ▼

Minimum observations for a leaf: *
5

Maximum branches of a node: *
2

Variable importance methods:
Gini (default) ▼

Method to calculate predicted nominal target levels:
Probability (default) ▼



5

Maximum branches of a node: *
2

Variable importance methods:
Gini (default) ▼

Method to calculate predicted nominal target levels:
Probability (default) ▼

The predicted probability of each target level is the the probability of that level averaged over all trees

▸ Sampling

▾ PLOTS

☑ Misclassifications by number of trees

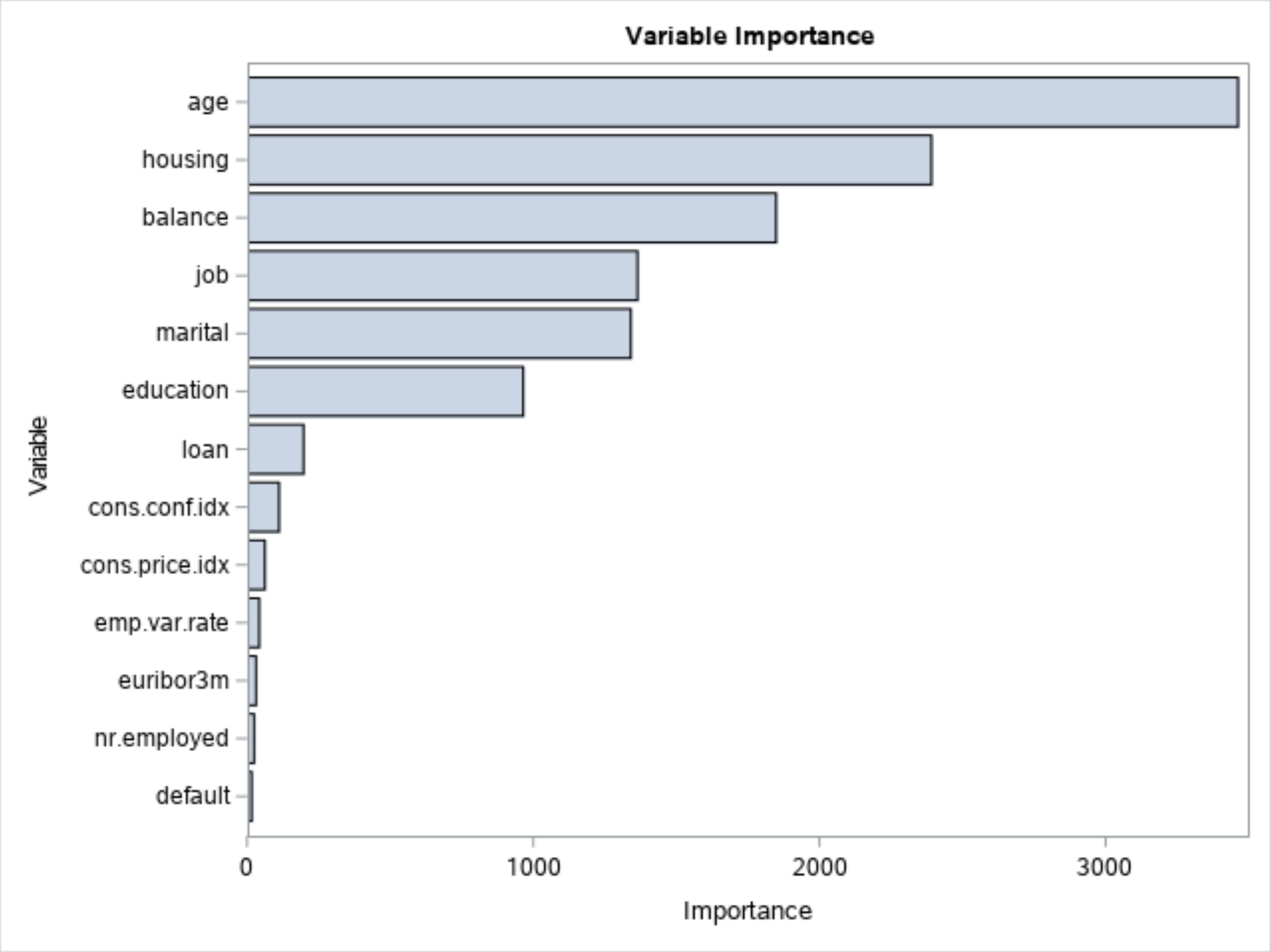
☑ Variable importance chart

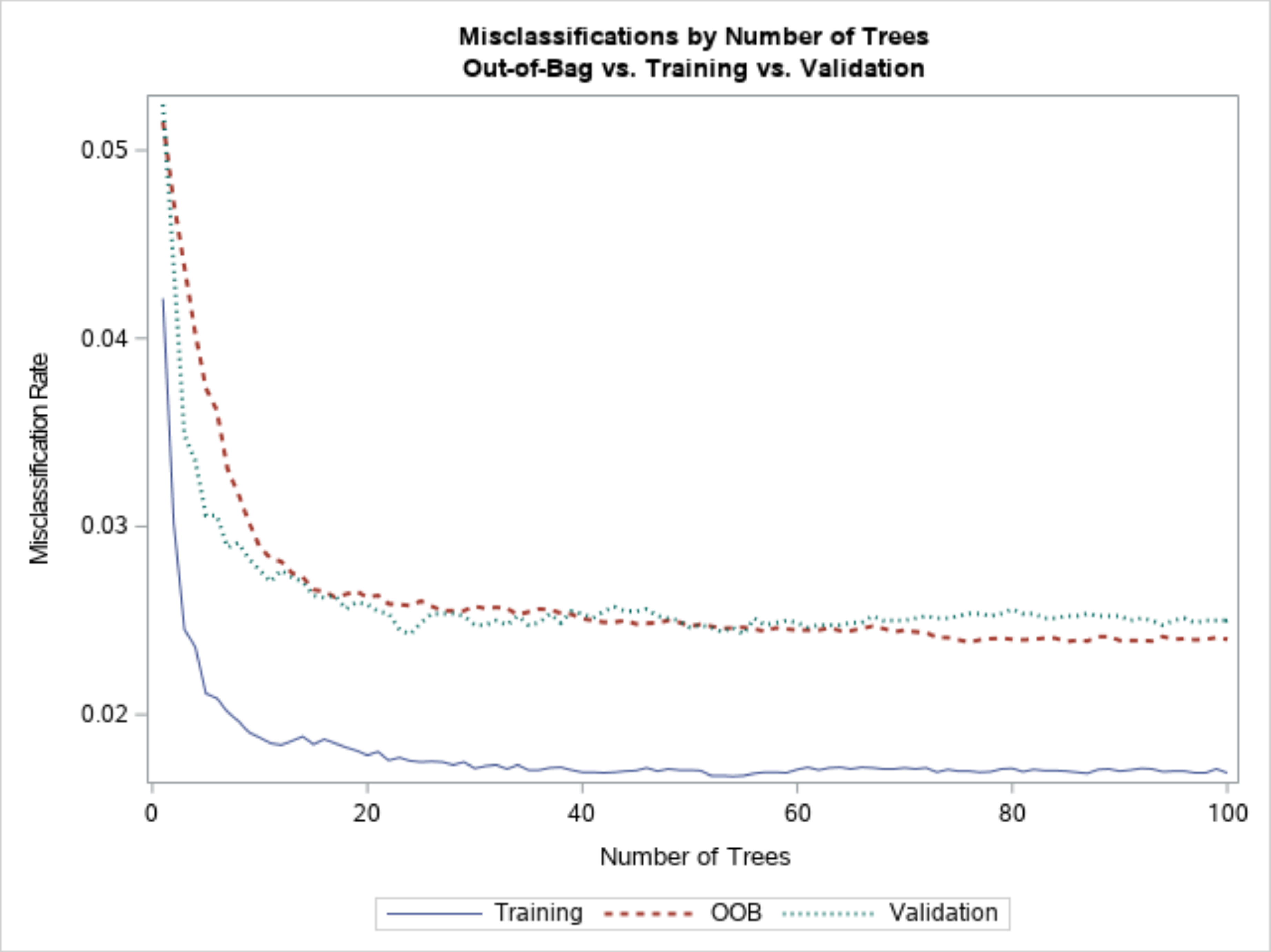
```
/* Task code generated by SAS® Studio 5.1
 * Generated on '10/28/19, 7:57 PM'
 * Generated by 'slrace'

ods noproctitle;
proc forest data=PUBLIC.BANK_NEW;
    partition fraction(validate=0.2 seed=7515);
    target 'next.product'n / level=nominal;
    input age 'emp.var.rate'n 'cons.price.idx'n 'cons.conf.idx'n euribor3m
        'nr.employed'n balance / level=interval;
    input job marital education default housing loan / level=nominal;
    ods output FitStatistics=Work._Forest_FitStats_
        VariableImportance=Work._Forest_VarImp_;
run;

proc sgplot data=Work._Forest_FitStats_;
    title3 'Misclassifications by Number of Trees';
    title4 'Out-of-Bag vs. Training vs. Validation';
    series x=Trees y=MiscTrain;
    series x=Trees y=MiscOob /lineattrs=(pattern=shortdash thickness=2);
    series x=Trees y=MiscValid /lineattrs=(pattern=dot thickness=2);
    yaxis label='Misclassification Rate';
    label Trees='Number of Trees';
    label MiscTrain='Training';
    label MiscOob='OOB';
    label MiscValid='Validation';
run;
title3;

proc sgplot data=Work._Forest_VarImp_;
    title3 'Variable Importance';
    hbar variable / response=importance nostatlabel categoryorder=respdesc;
run;
```


Variable Importance
age
housing
balance
job
marital
education
loan
cons.conf.idx
cons.price.idx
emp.var.rate
euribor3m
nr.employed
default
Variable
0
1000
2000
3000
Importance


Misclassifications by Number of Trees
Out-of-Bag vs. Training vs. Validation
Misclassification Rate
0.05
0.04
0.03
0.02
0
20
40
60
80
100
Number of Trees
Training
OOB
Validation