Random Forest

• Overall, Random Forest is accurate, efficient, and relatively quick to develop, making it an extremely handy tool for data professionals.

Random Forest developed by aggregated trees: grows multiple decision trees which are merged for a more accurate prediction.

- Used for Classification and Regression
- Avoids overfitting
- Can deal with large number of features.
- Helps with feature selection based on importance

Logic: multiple uncorrelated models (the individual decision trees) perform much better as a group than they do alone.

Is it Regression or Classification?

There are two main types of Trees:

1. Regression Trees (Continuous data types)
    - Decision trees where the target variable can take continuous values (typically real numbers).
    - Returns average value at every node
    - Recursive binary splitting is a greedy, top-down algorithm that tries to minimize the residual sum of squares that tries to minimize the residual sum of squares. This is an iterative process of splitting the data into partitions, and then splitting it up further on each of the branches.

2. Classification Trees (Yes/No types)

- The outcome is a variable like ‘spam’ or ‘not spam’. Here the decision variable deals with categorical or qualitative variables.
- The splits in data are made based on questions with qualitative answers, therefore, the residual sum of squares cannot be used as a measure here. Instead, classification trees are created based on measures like classification error rate, cross-entropy, etc..

Classification in Machine Learning

- The task of learning to distinguish points that belong to two or more categories in a dataset.

- **Classification is about predicting a label** (e.g. “diagnosed” or “not diagnosed”)
- **Regression is about predicting a quantity**.

Random forest is used on the job by data scientists in many industries including banking, stock trading, medicine, and e-commerce. It’s used to predict the things which help these industries run efficiently, such as customer activity, patient history, and safety.

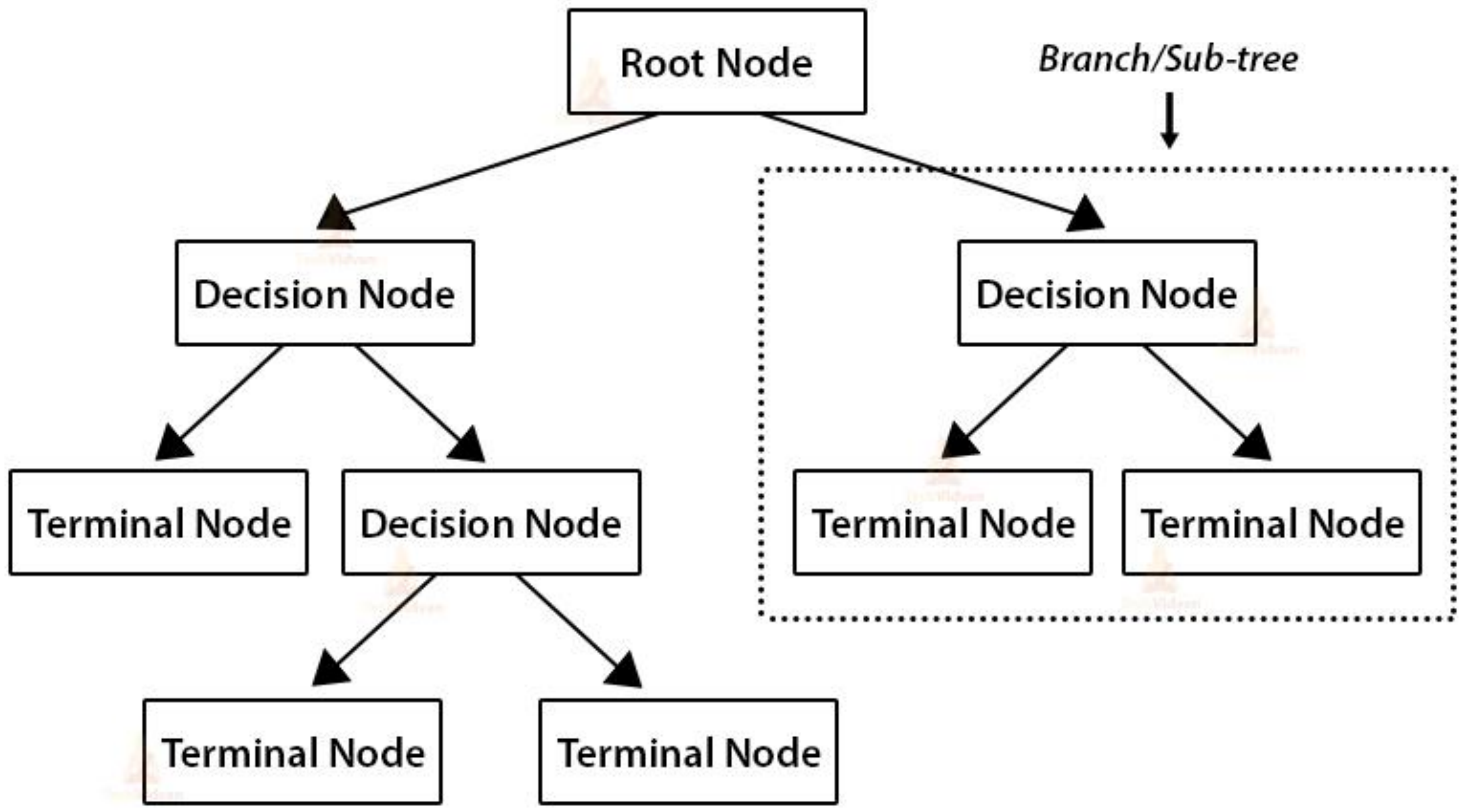
- It’s used by **retail companies** to recommend products and predict customer satisfaction as well.
- Scientists in China used Random Forest to study the spontaneous combustion patterns of coal to reduce safety risks in **coal mines**!
- In **healthcare,** Random Forest can be used to analyze a patient’s medical history to identify diseases.
    - Pharmaceutical scientists use Random Forest to identify the correct combination of components in a medication or predict drug sensitivity. Sometimes Random Forest is even used for computational biology and the study of genetics.

Decision Trees

A decision tree is another type of algorithm used to classify data.

In its simplest form, a decision tree is a type of flowchart that shows a clear pathway to a decision

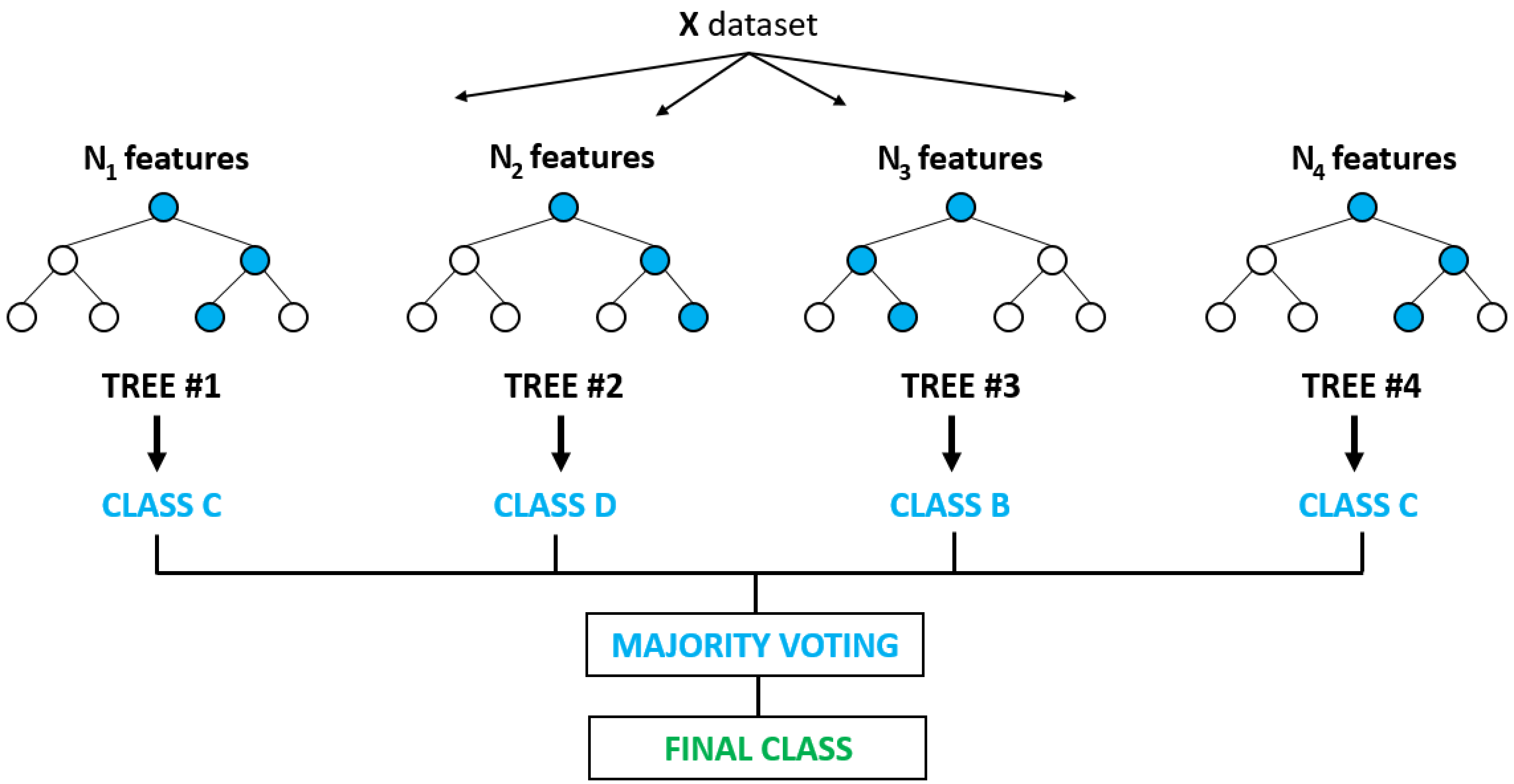
Parts of a Decision Trees in R



Library	Objective	Function	Class	Parameters	Details
rpart	Train classification tree in R	rpart()	class	formula, df, method	
rpart	Train regression tree	rpart()	anova	formula, df, method	
rpart	Plot the trees	rpart.plot()		fitted model	
base	predict	predict()	class	fitted model, type	
base	predict	predict()	prob	fitted model, type	
base	predict	predict()	vector	fitted model, type	
rpart	Control parameters	rpart.contr ol()		minsplit	Set the minimum number of observations in the node before the algorithm perform a split
				minbucket	Set the minimum number of observations in the final note i.e. the leaf
				maxdepth	Set the maximum depth of any node of the final tree. The root node is treated a depth 0
rpart	Train model with control parameter	rpart()		formula, df, method, control	

Are decision trees in Random Forest different from regular decision trees?

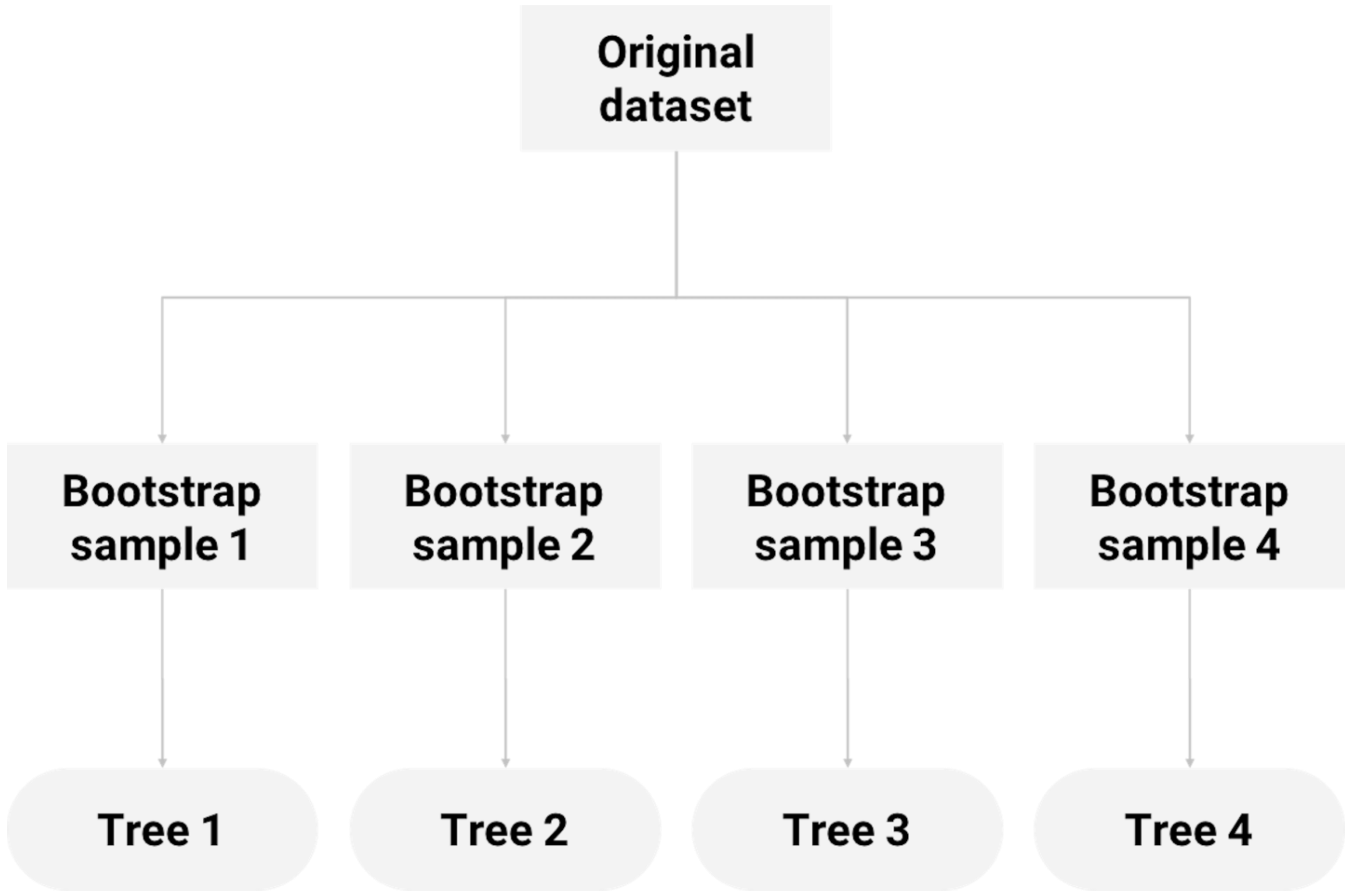
It seems like a decision forest would be a bunch of single decision trees, and it is… kind of. It’s a bunch of single decision trees but all the trees are mixed randomly instead of separate trees growing individually.

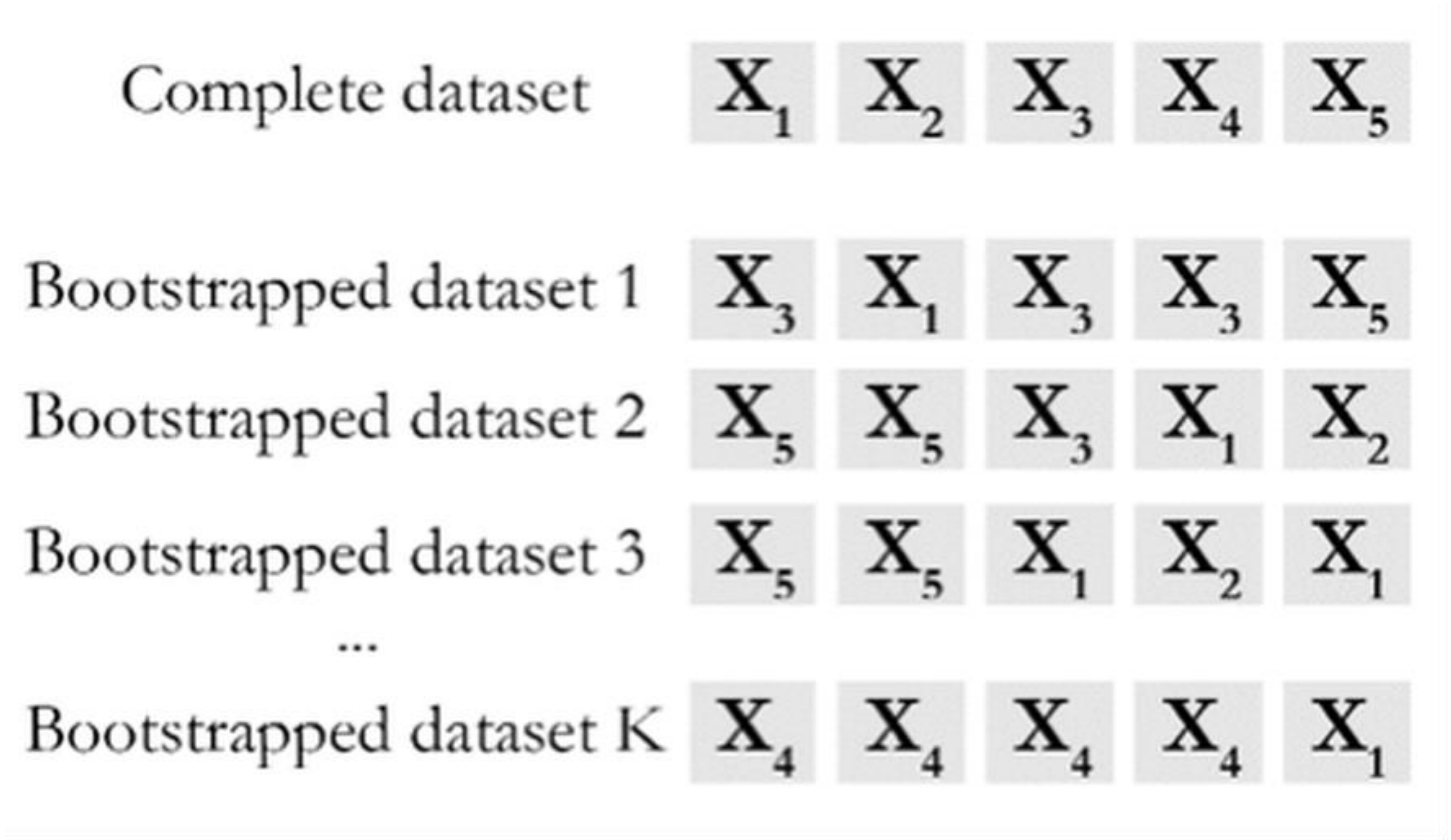

X dataset
N_1 features
N_2 features
N_3 features
N_4 features
TREE #1
TREE #2
TREE #3
TREE #4
CLASS C
CLASS D
CLASS B
CLASS C
MAJORITY VOTING
FINAL CLASS

Ensemble Learning

Type of Supervised Learning Technique in which the basic idea is to generate multiple models on a training dataset and then simply combine (average) their Output Rules or their Hypothesis to generate a Strong Model which performs very well and does not overfits and which balances the Bisa-Variance Tradeoff too.

How are the trees in a Random Forest trained?





Complete dataset	X_1	X_2	X_3	X_4	X_5
Bootstrapped dataset 1	X_3	X_1	X_3	X_3	X_5
Bootstrapped dataset 2	X_5	X_5	X_3	X_1	X_2
Bootstrapped dataset 3	X_5	X_5	X_1	X_2	X_1
...					
Bootstrapped dataset K	X_4	X_4	X_4	X_4	X_1

The basic algorithm for a regression or classification random forest can be generalized as follows:

```
1.  Given a training data set
2.  Select number of trees to build (n_trees)
3.  for i = 1 to n_trees do
4.  |  Generate a bootstrap sample of the original data
5.  |  Grow a regression/classification tree to the bootstrapped data
6.  |  for each split do
7.  |  | Select m_try variables at random from all p variables
8.  |  | Pick the best variable/split-point among the m_try
9.  |  | Split the node into two child nodes
10. |  end
11. | Use typical tree model stopping criteria to determine when a
    | tree is complete (but do not prune)
12. end
13. Output ensemble of trees
```

- **Bootstrap** randomly performs row sampling and feature sampling from the dataset to form sample datasets for every model.
- **Aggregation** reduces these sample datasets into summary statistics based on the observation and combines them. Bootstrap Aggregation can be used to reduce the variance of high variance algorithms such as decision trees.
- **Variance** is an error resulting from sensitivity to small fluctuations in the dataset used for training.
  - High variance will cause an algorithm to model irrelevant data, or noise, in the dataset instead of the intended outputs, called signal. This problem is called **overfitting**.
  - An overfitted model will perform well in training but won't be able to distinguish the noise from the signal in an actual test.
- **Bagging** is the application of the bootstrap method to a high variance machine learning algorithm.

- In the random forest approach, a large number of decision trees are created. Every observation is fed into every decision tree. The most common outcome for each observation is used as the final output. A new observation is fed into all the trees and taking a majority vote for each classification model.
- The idea is to build lots of Trees in such a way to make the Correlation between the Trees smaller.
- An error estimate is made for the cases which were not used while building the tree. That is called an **OOB (Out-of-bag)** error estimate which is mentioned as a percentage.
- The R package **"randomForest"** is used to create random forests.

OOB (Out-of-bag) error estimate

- In each tree of the random forest, the out-of-bag error is calculated based on predictions for observations that were not in the bootstrap sample for that tree.

- Error rate is approximated by the out-of-bag error during the training process.
  - Classification: accuracy
  - Regression: R^2 and RMSE

- Finding parameters that would produce a low out-of-bag error is often a key consideration in **parameter tuning**

What are the advantages of Random Forest?

The key benefits of using Random Forest are:

- Accuracy
- Efficiency
- Ease of use
- Ability to correct for decision trees’ habit of overfitting to their training set.
- Versatility – can be used for classification or regression
- More beginner friendly than similarly accurate algorithms like neural nets

What are the disadvantages of Random Forest?

- Because random forest uses many decision trees, it can require a lot of memory on larger projects. This can make it slower than some other, more efficient, algorithms.
- Sometimes, because this is a decision tree-based method and decision trees often suffer from overfitting, this problem can affect the overall forest. This problem is usually prevented by Random Forest by default because it uses random subsets of the features and builds smaller trees with those subsets. This can slow down processing speed but increase accuracy.

Caret package

- *Caret Package is a comprehensive framework for building machine learning models in R.*
- Caret is short for Classification And Regression Training.
- It integrates all activities related to model development in a streamlined workflow. For nearly every major ML algorithm available in R.

Hyperparameters

- A model hyperparameter is the parameter whose value is set before the model start training. They cannot be learned by fitting the model to the data.
- Hyperparameters are configurations that cannot be learnt from the regular data that we provide to the algorithm; these are inbuilt to the algorithm and each algorithm has its own predefined set of hyperparameters.
- Hyperparameters are often tuned for increasing model accuracy
- Responsible for deciding how quickly a model can fit onto the data to produce accurate results.

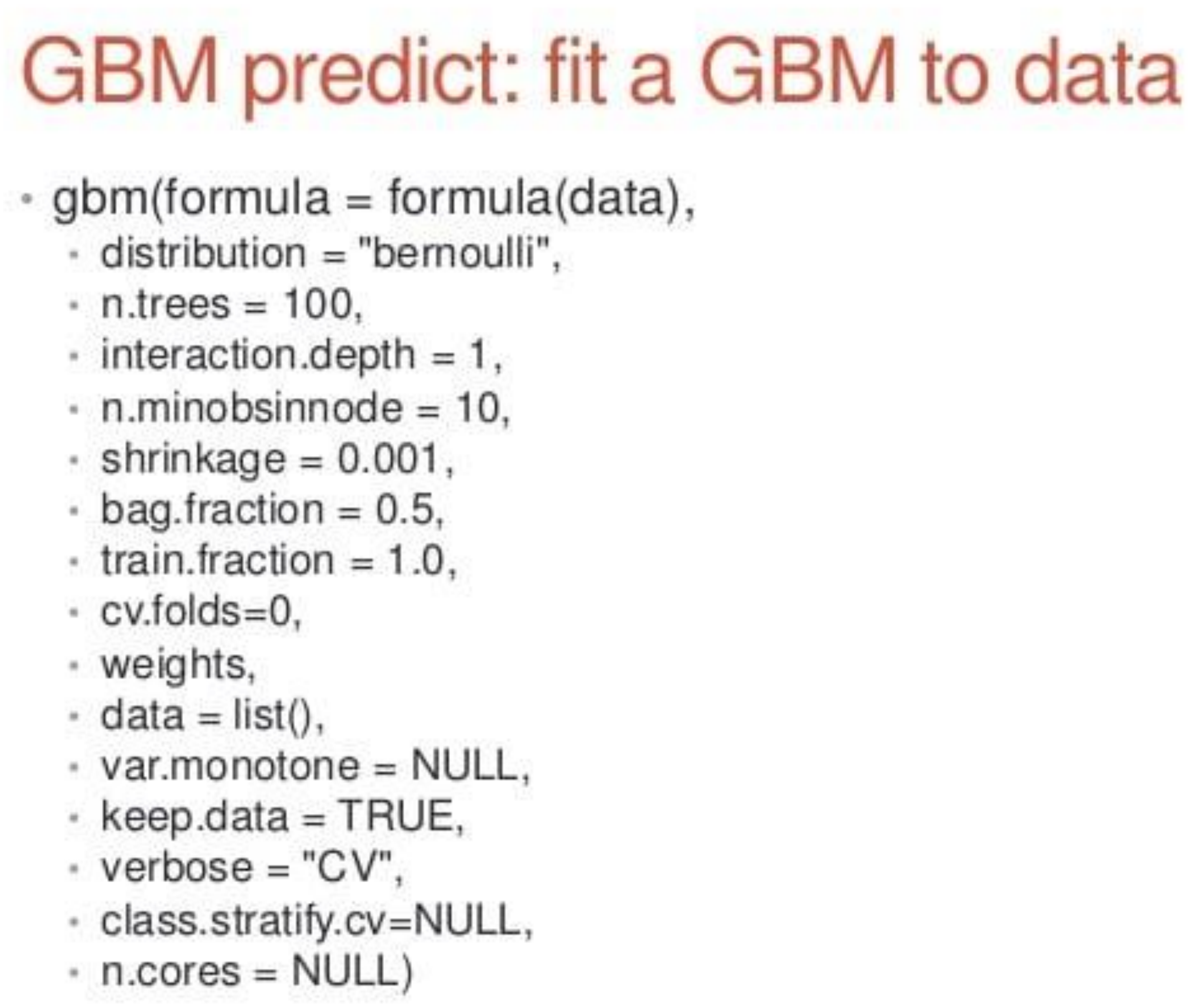
Hyperparameters

These are the major hyperparameters that are present implicitly in the random forest classifier which is required to be tuned in order to increase the accuracy of our training model.

- **n_estimators:**
- **max_depth**
- **min_samples_split**
- **min_samples_leaf:**
- **max_features**
- **max_leaf_nodes**
- **max_samples**

We will proceed as follow to train the Random Forest:

- Step 1) Import the data
- Step 2) Train the model
- Step 3) Construct accuracy function
- Step 4) Visualize the model
- Step 5) Evaluate the model
- Step 6) Visualize Result



GBM predict: fit a GBM to data

- gbm(formula = formula(data),
  - distribution = "bernoulli",
  - n.trees = 100,
  - interaction.depth = 1,
  - n.minobsinnode = 10,
  - shrinkage = 0.001,
  - bag.fraction = 0.5,
  - train.fraction = 1.0,
  - cv.folds=0,
  - weights,
  - data = list(),
  - var.monotone = NULL,
  - keep.data = TRUE,
  - verbose = "CV",
  - class.stratify.cv=NULL,
  - n.cores = NULL)