BT2101 Tutorial 3
Ensemble Learning

Assignment 2

- Please set “shuffle=True”, “random_state” serves “shuffle” step

random_state : int, RandomState instance or None, optional, default=None

> If int, random_state is the seed used by the random number generator; If RandomState instance, random_state is the random number generator; If None, the random number generator is the RandomState instance used by np.random. Used when `shuffle` == True.

- http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html

Assignment 2

- Please set "shuffle=True", "random_state" serves "shuffle" step

Step 1. Do K-fold Cross Validation

- Set `n_splits=5` : 5-fold cross validation
- Set `random_state` to `12345`
- Set `shuffle` to `True`

Hint: Use `KFold` function

```
# Create a 5-fold cross validation
kf = KFold(n_splits=5, shuffle=True, random_state=12345)
```

Assignment 2

Comparison between L1 (Lasso) and L2 (Ridge) regularization

L1 regularization on least squares:

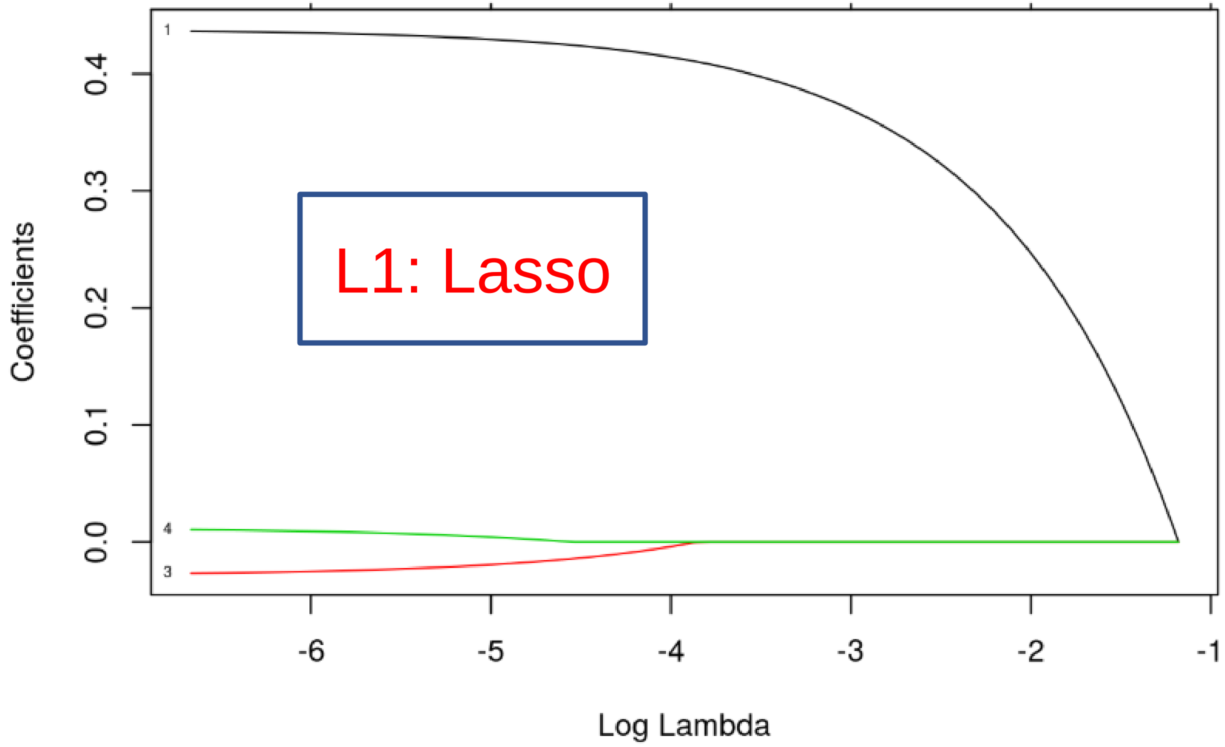
$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_j \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2 + \boxed{\lambda \sum_{i=1}^{k} |w_i|}$$

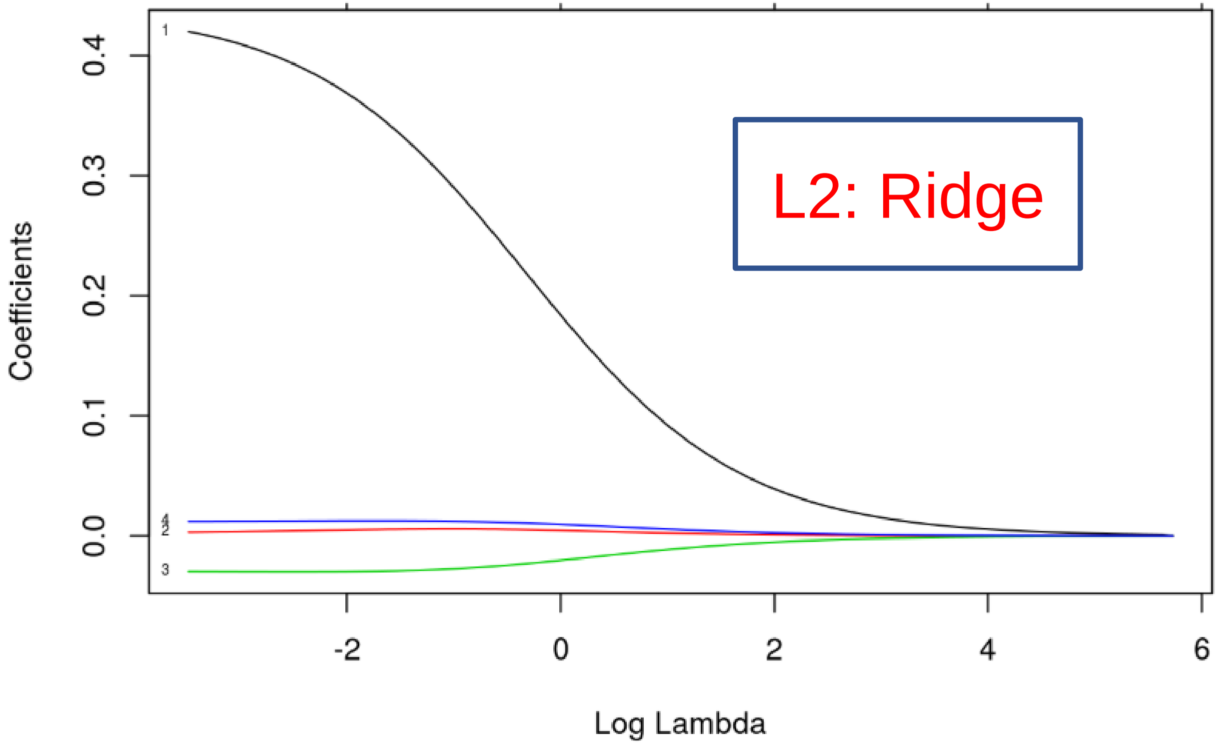
L2 regularization on least squares:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_j \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2 + \boxed{\lambda \sum_{i=1}^{k} w_i^2}$$

Assignment 2

Comparison between L1 (Lasso) and L2 (Ridge) regularization





http://wavedatalab.github.io/machinelearningwithr/post4.html

Assignment 2

Some online resources:

- http://statweb.stanford.edu/~tibs/sta305files/Rudyregularization.pdf
- https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-ridge-lasso-regression-python/
- http://scikit-learn.org/stable/auto_examples/linear_model/plot_logistic_l1_l2_sparsity.html
- <The Elements of Statistical Learning> Page 61-73

https://web.stanford.edu/~hastie/Papers/ESLII.pdf

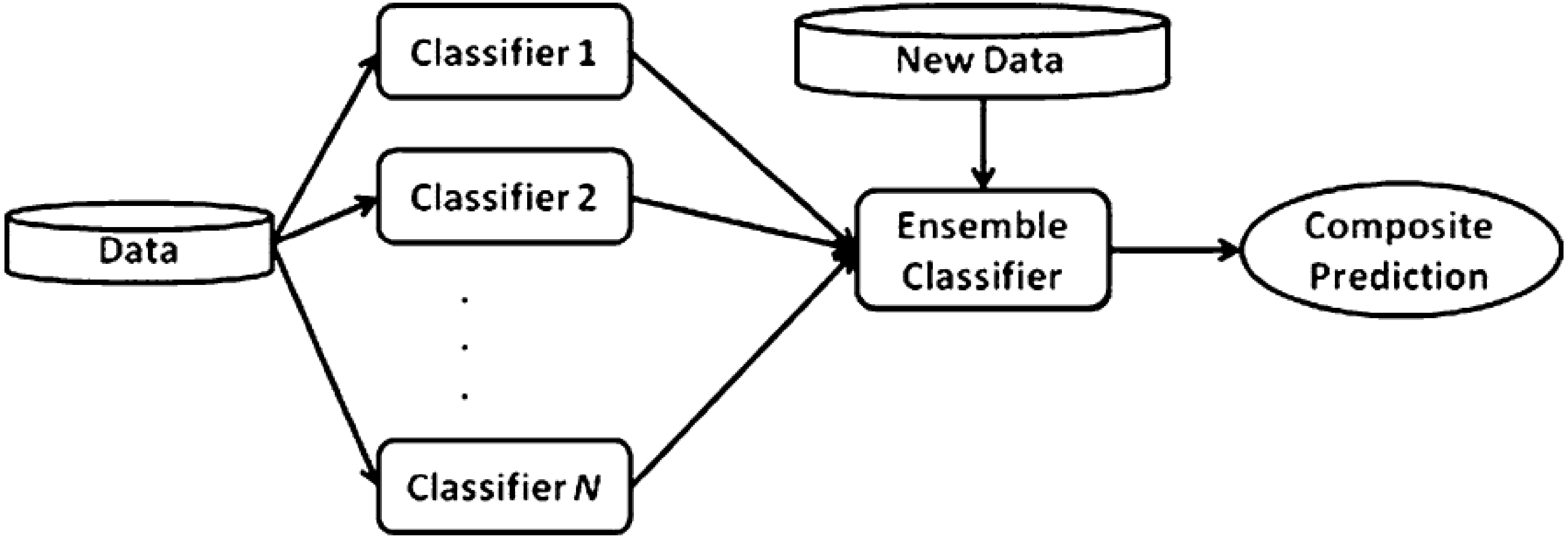
Agenda

- Understand Ensemble Learning
- Discussion about Programming Assignment 3
    - BAGGING
    - Random Forest
    - AdaBoost
- Python Implementation

Ensemble Learning Method

What is Ensemble Method?

- Think about



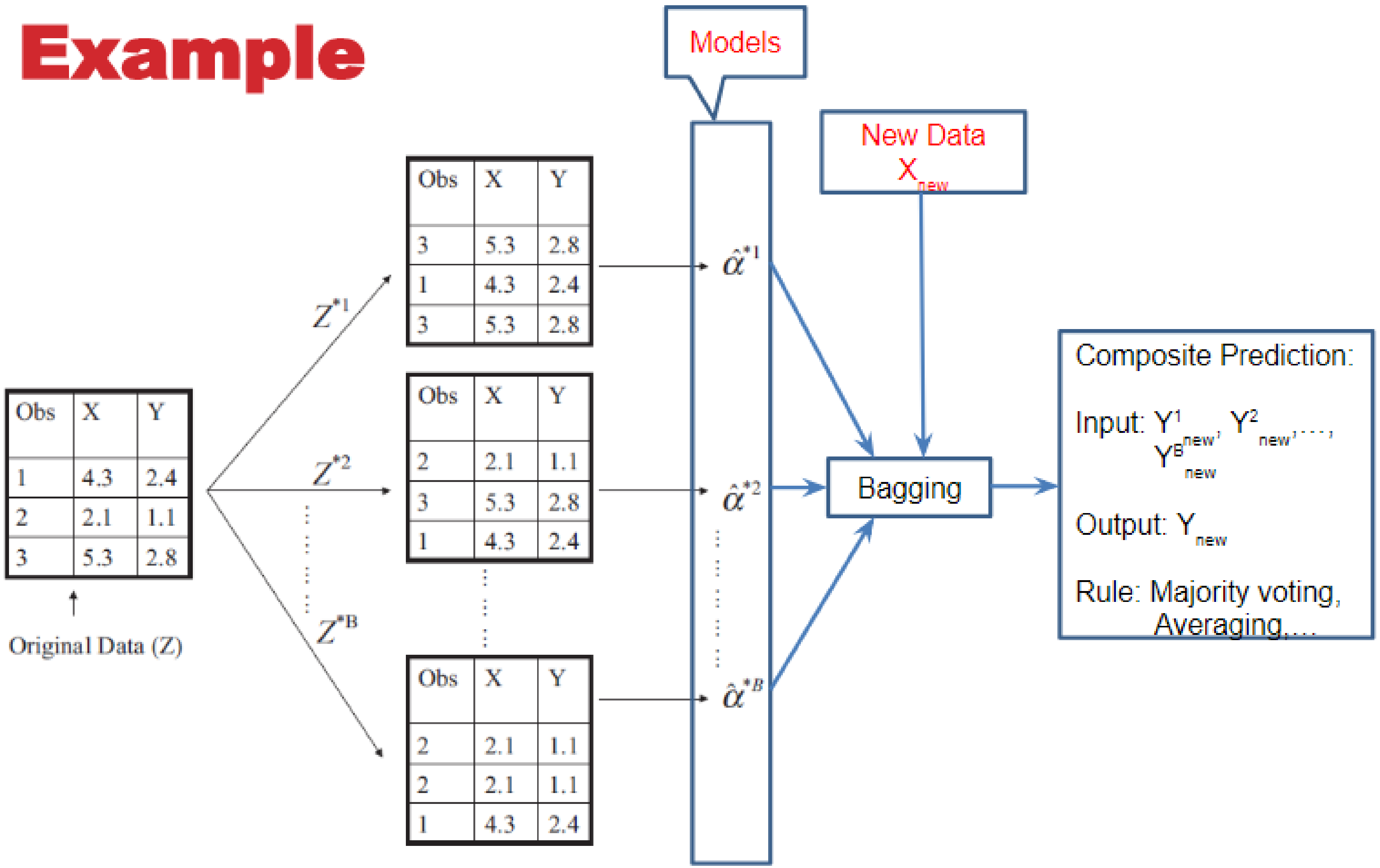
Learn from not just one classifier but a set of base classifiers, and combine their predictions for the classification of unseen instances using some form of voting (e.g., majority voting) (<Principles of Data Mining>)

Ensemble Method

- Bagging
- Random Forest
- Boosting
  - Adaboost
  - Gradient boosting
  - XGBoost
  - LightGBM
- Stacking

Bagging **Example**

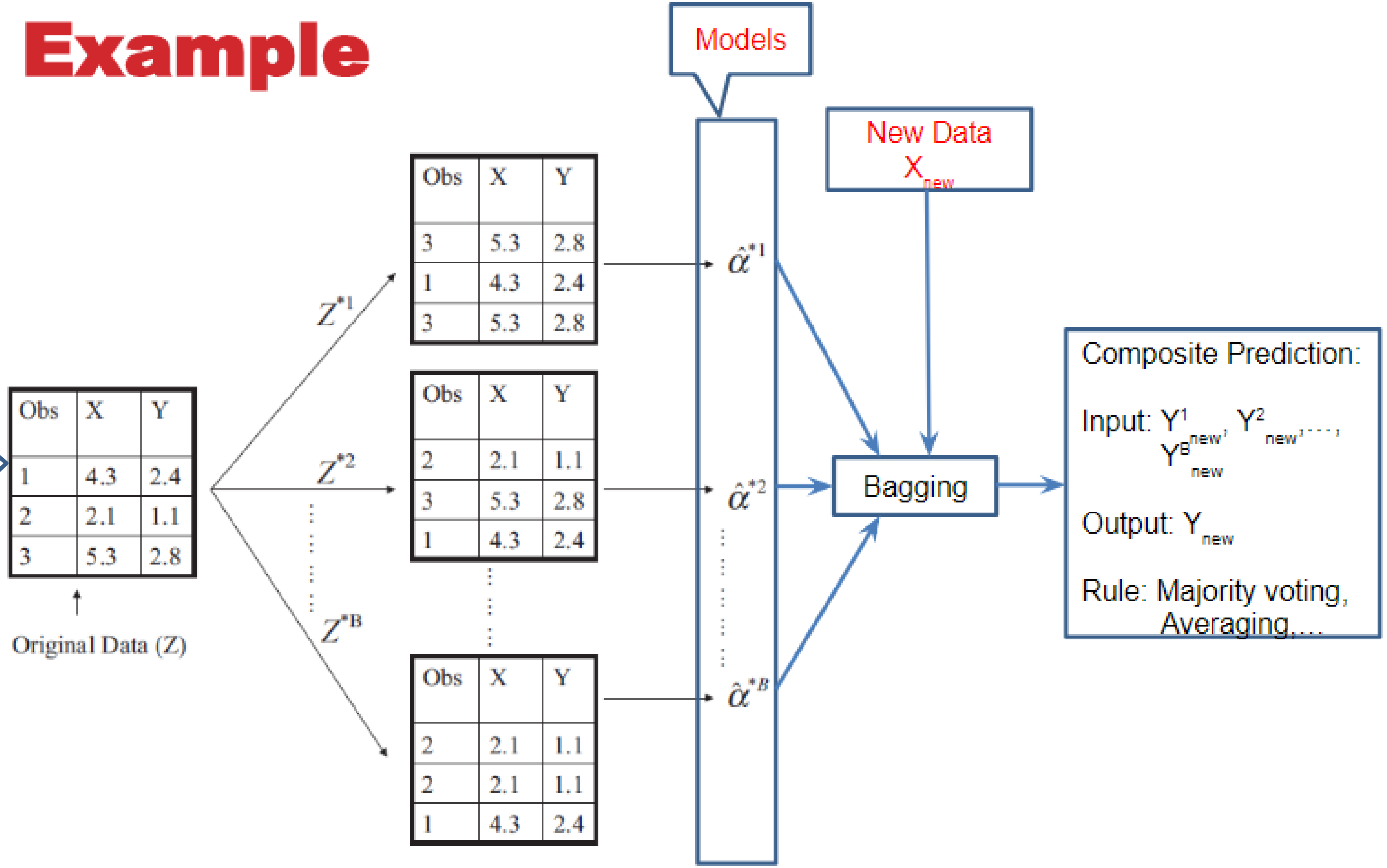
- Bagging



Bagging **Example**

Step 1.

Get your training dataset, suppose data size is N (=3 in this example)



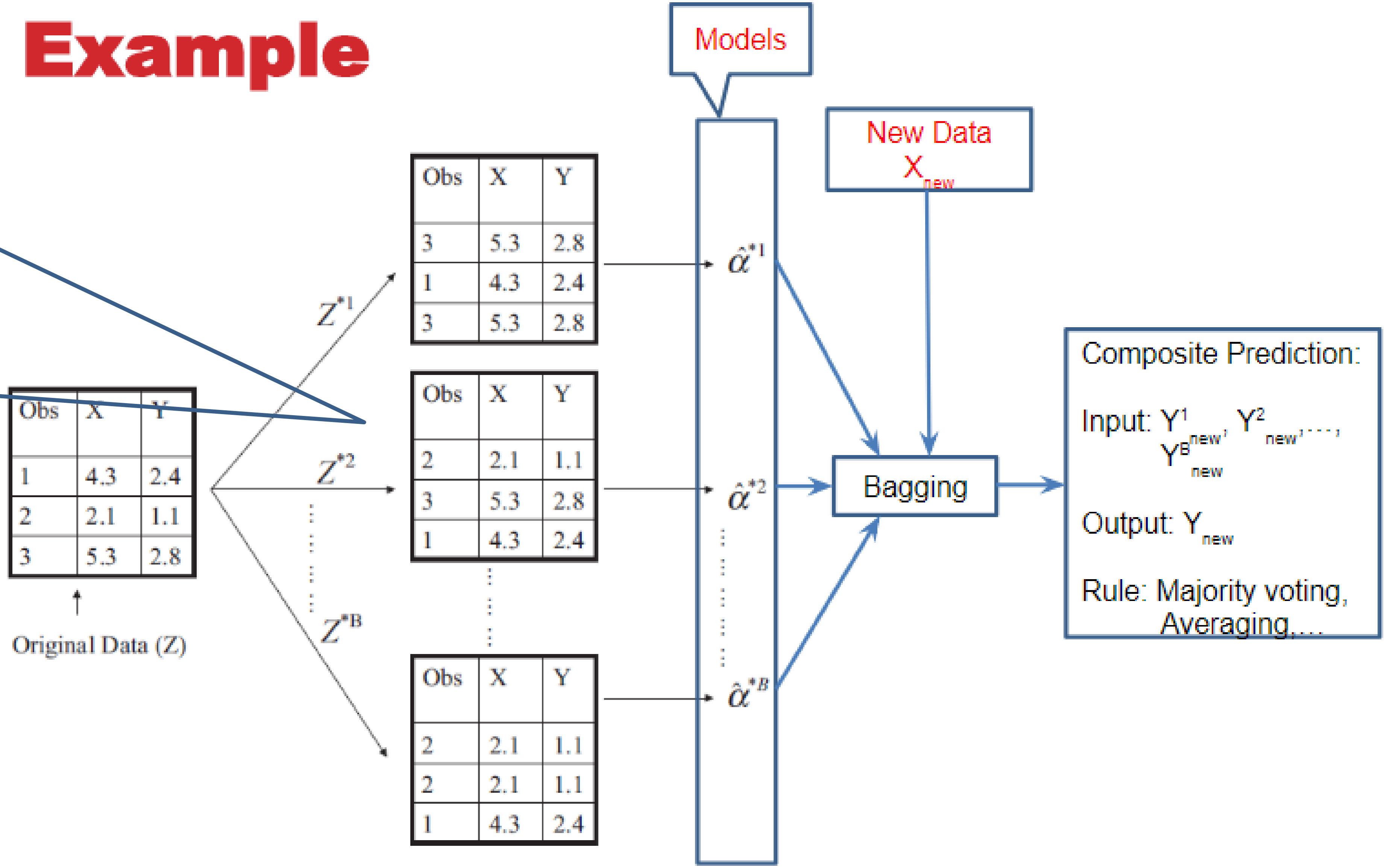
Bagging **Example**

Step 2.

Bootstrap sampling (with replacement) from the original dataset B times

You get B bootstrap samples

Each bootstrap sample size is N (usually same size as original dataset)

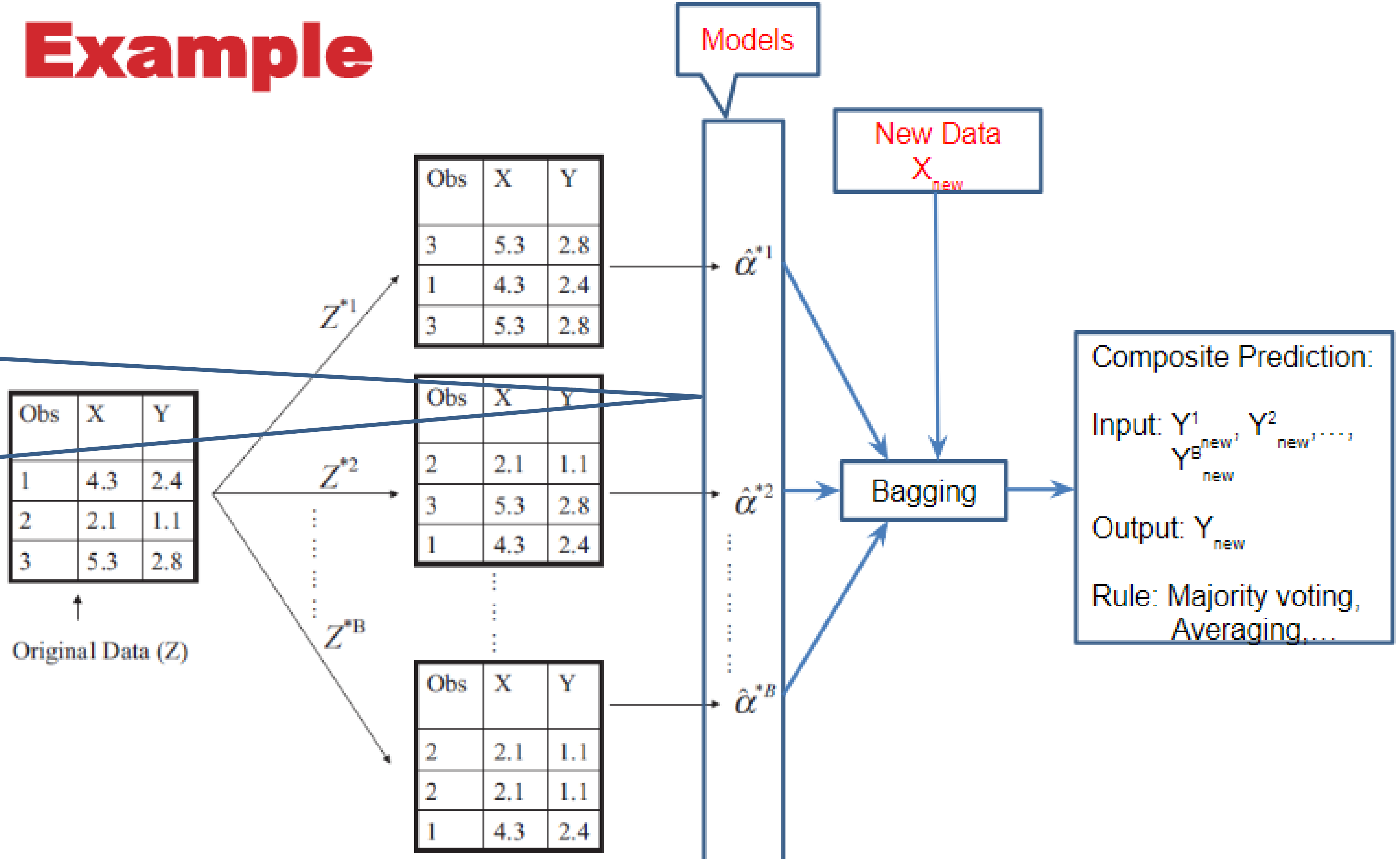


Bagging **Example**

Step 3.

You train B base models using these B bootstrap samples

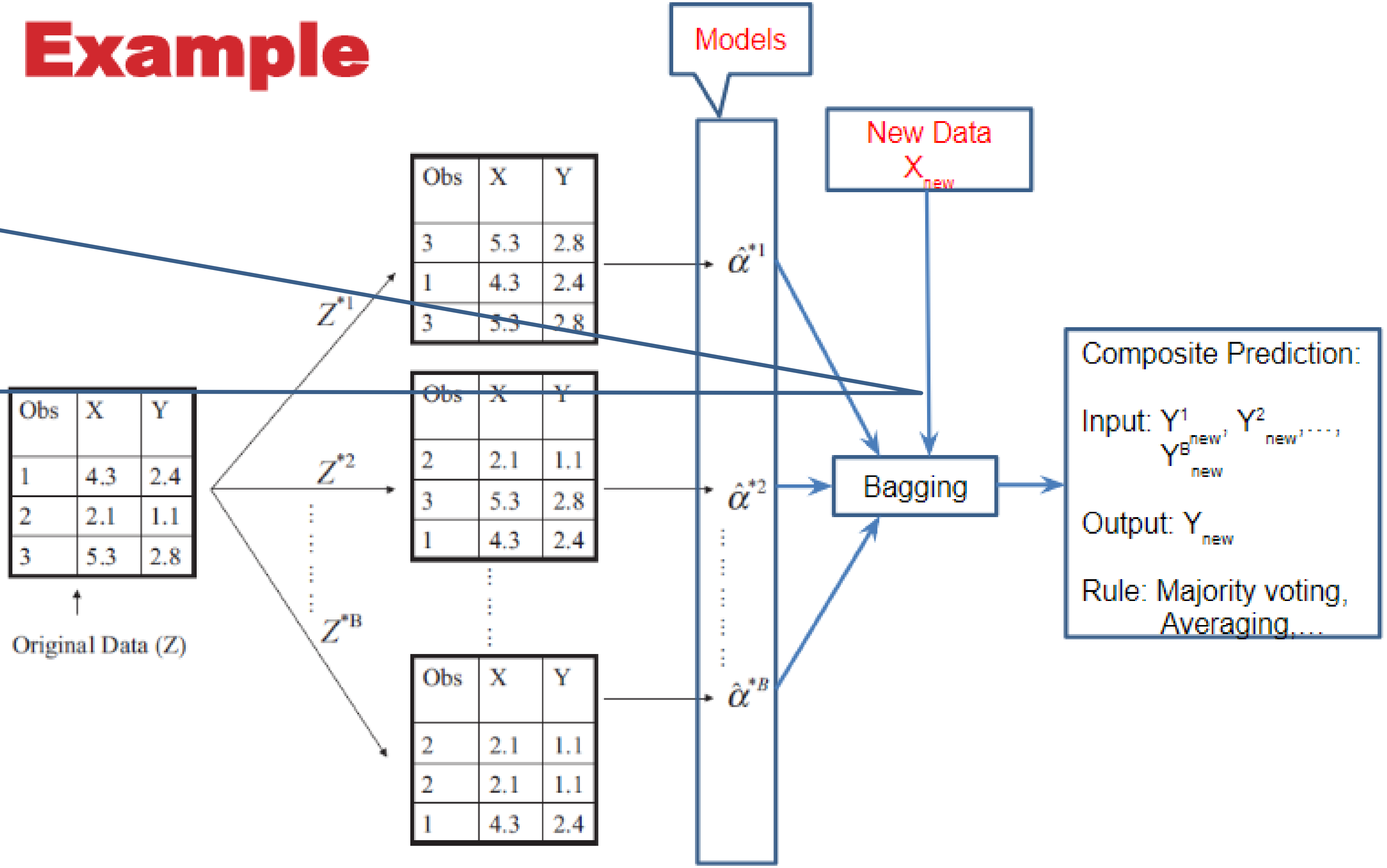
Now you have B base models



Bagging **Example**

Step 4.

Combine these base models into a “bag”



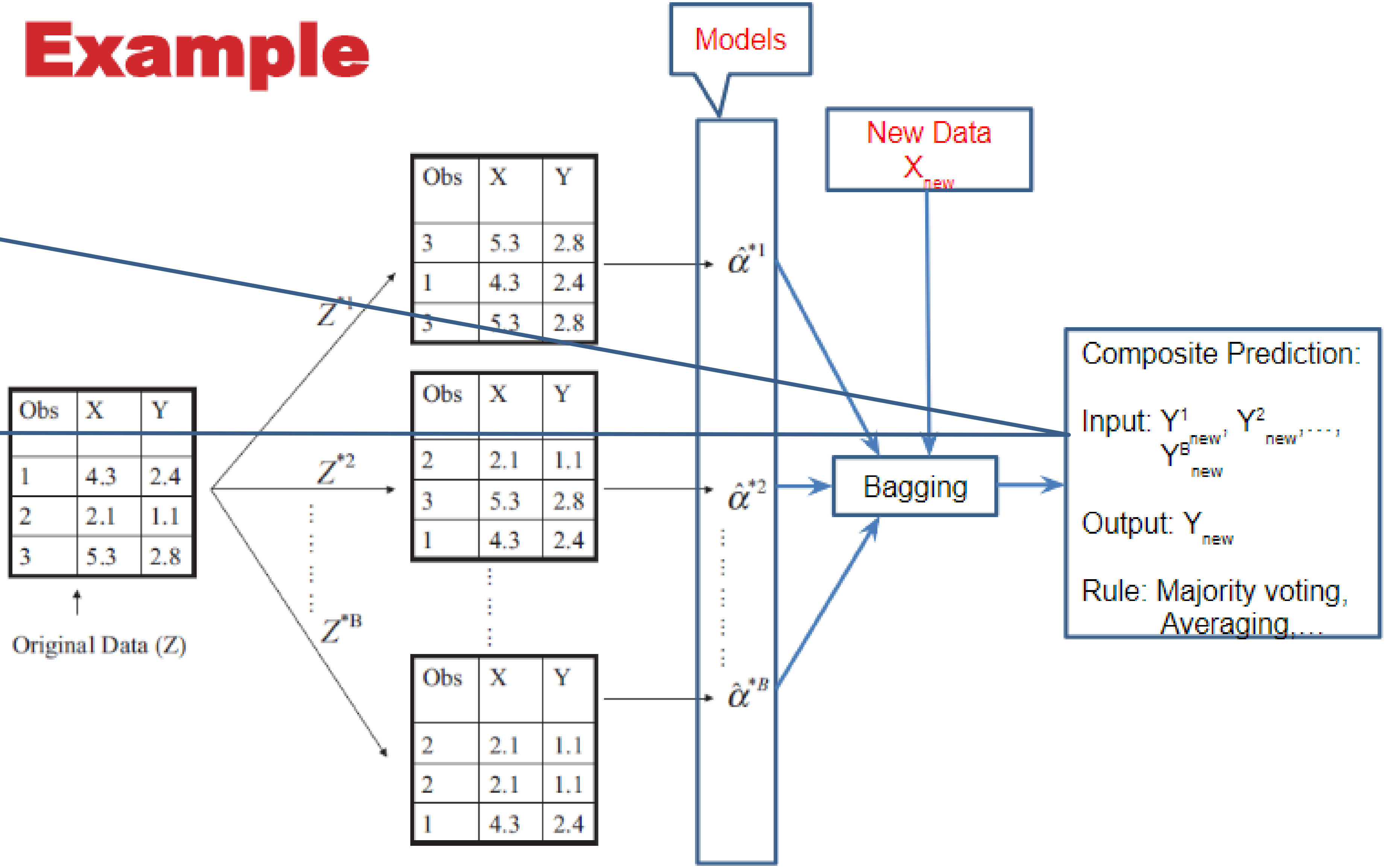
Bagging **Example**

Step 5.

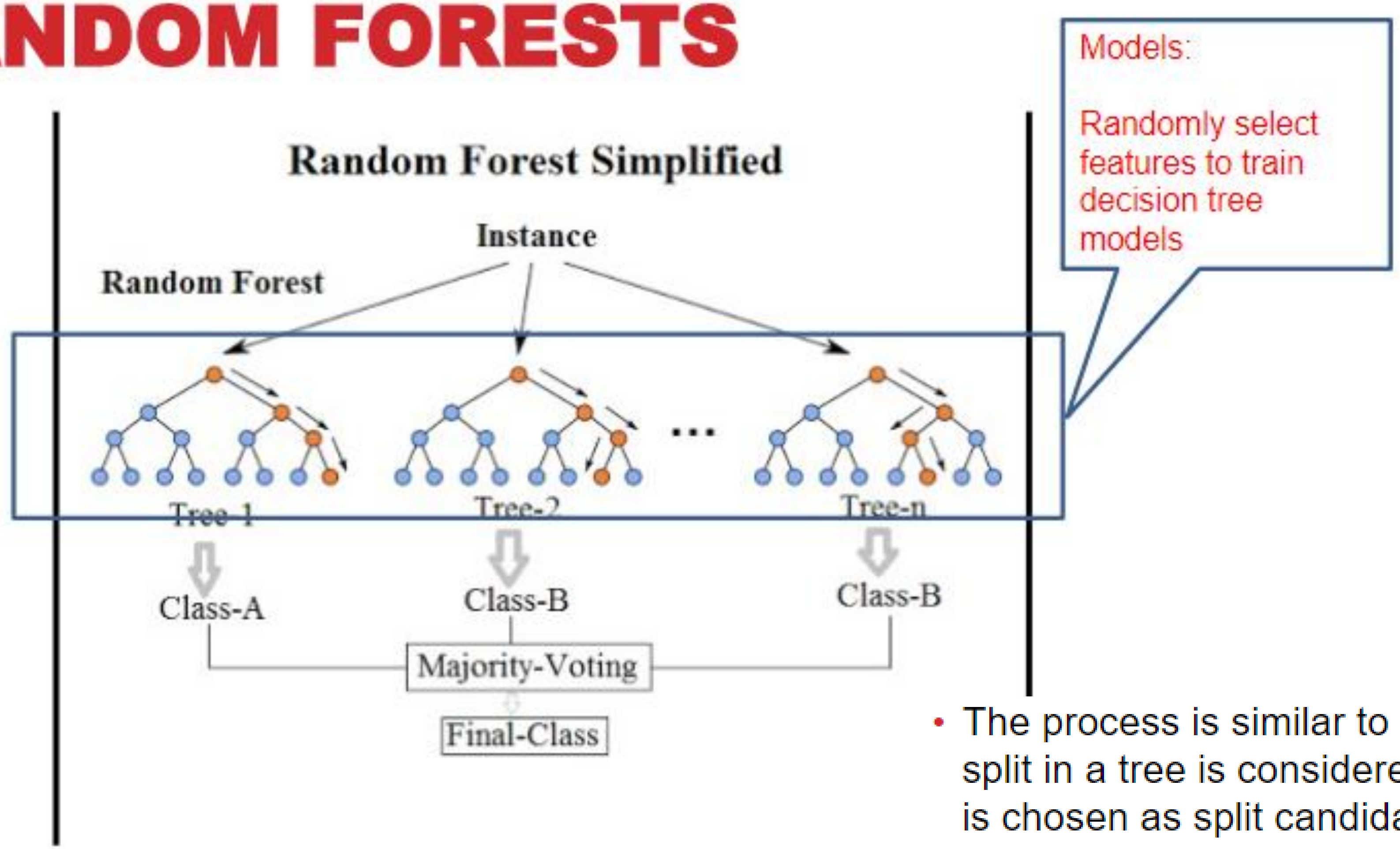
Suppose you want to predict output of new data sample X_{new}

Get predictions from B base models: $(Y^1_{new}, Y^2_{new}, \ldots, Y^B_{new})$

Composite prediction by averaging (for continuous output) or majority voting (for classification)

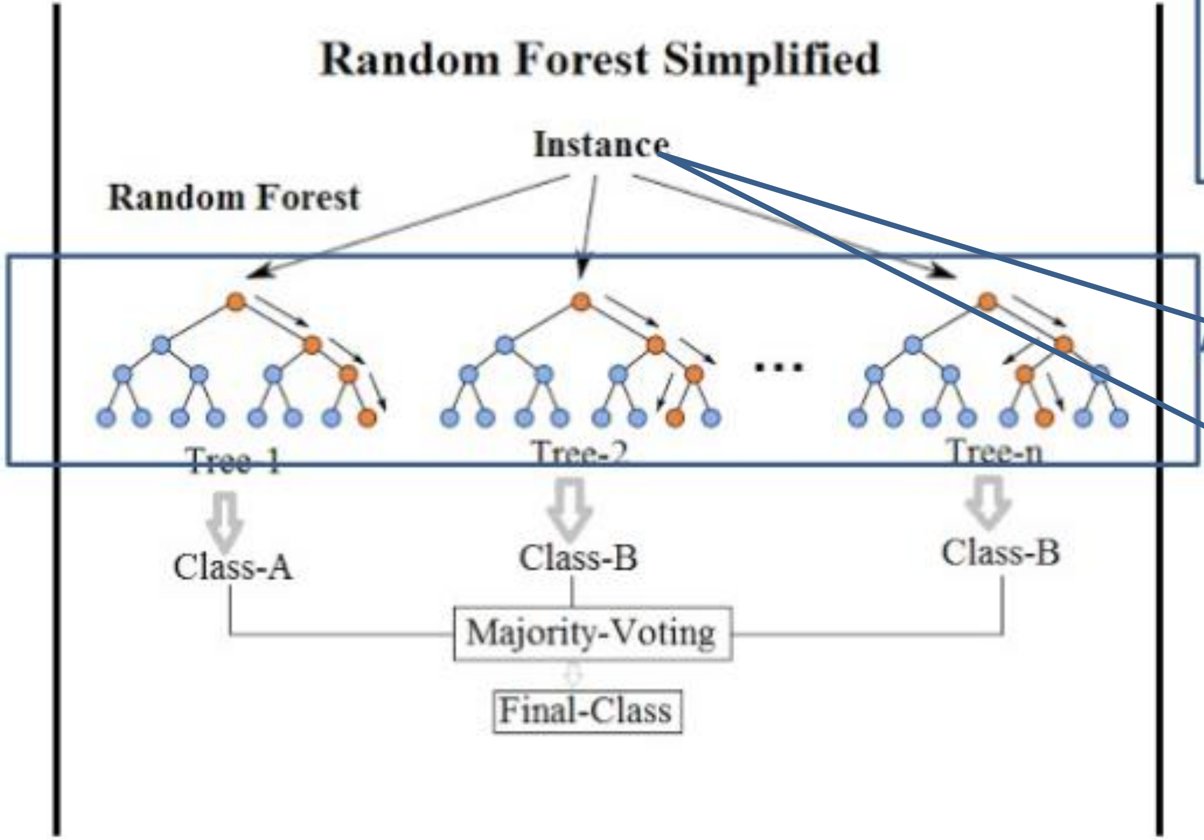


RANDOM FORESTS



- The process is similar to bagging except that, each time a split in a tree is considered, a random sample of m predictors is chosen as split candidate from the full set of p predictors
- In Practice, $m \approx \sqrt{p}$
- Bagging is a special case of random forest where $m = p$

RANDOM FORESTS



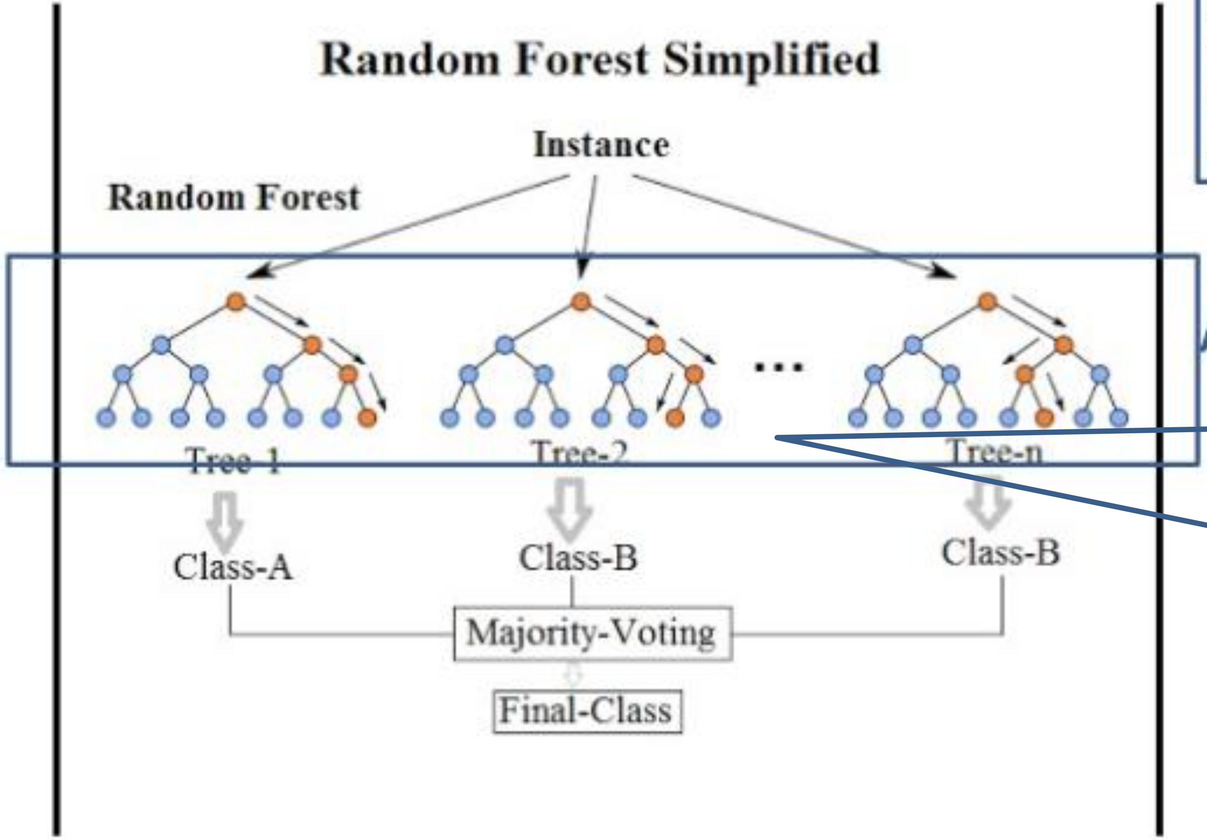
Models:

Randomly select features to train decision tree models

Step 1.

Get your training dataset, suppose data size is N

RANDOM FORESTS



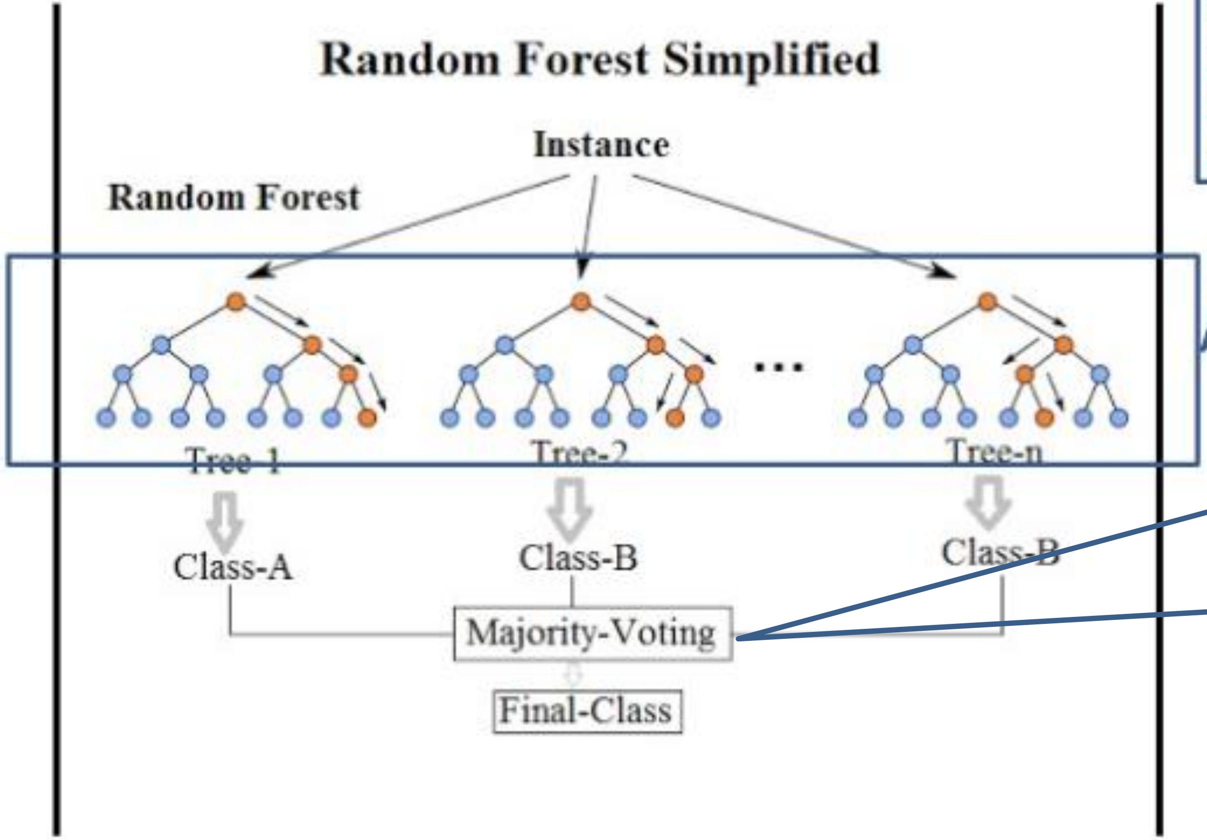
Models:

Randomly select features to train decision tree models

Step 2.

Create n base models:
Each time, randomly select a subset of original features to get the base model

RANDOM FORESTS



Models:

Randomly select features to train decision tree models

Step 3.

Composite prediction: Similar to bagging

Boosting Method

Difference?

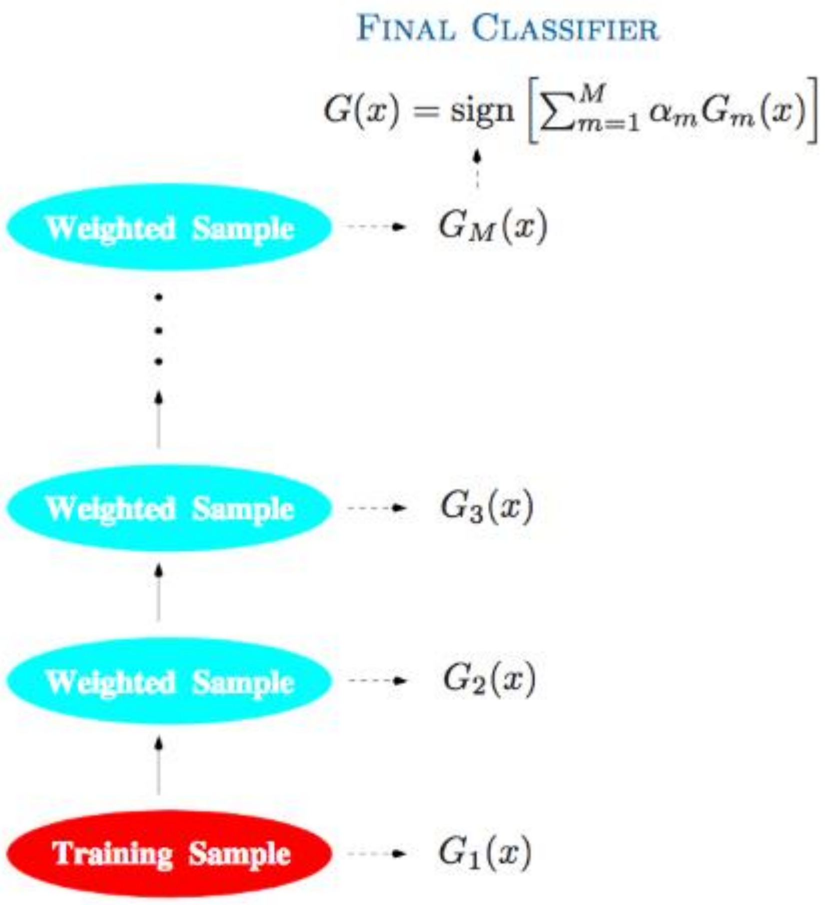
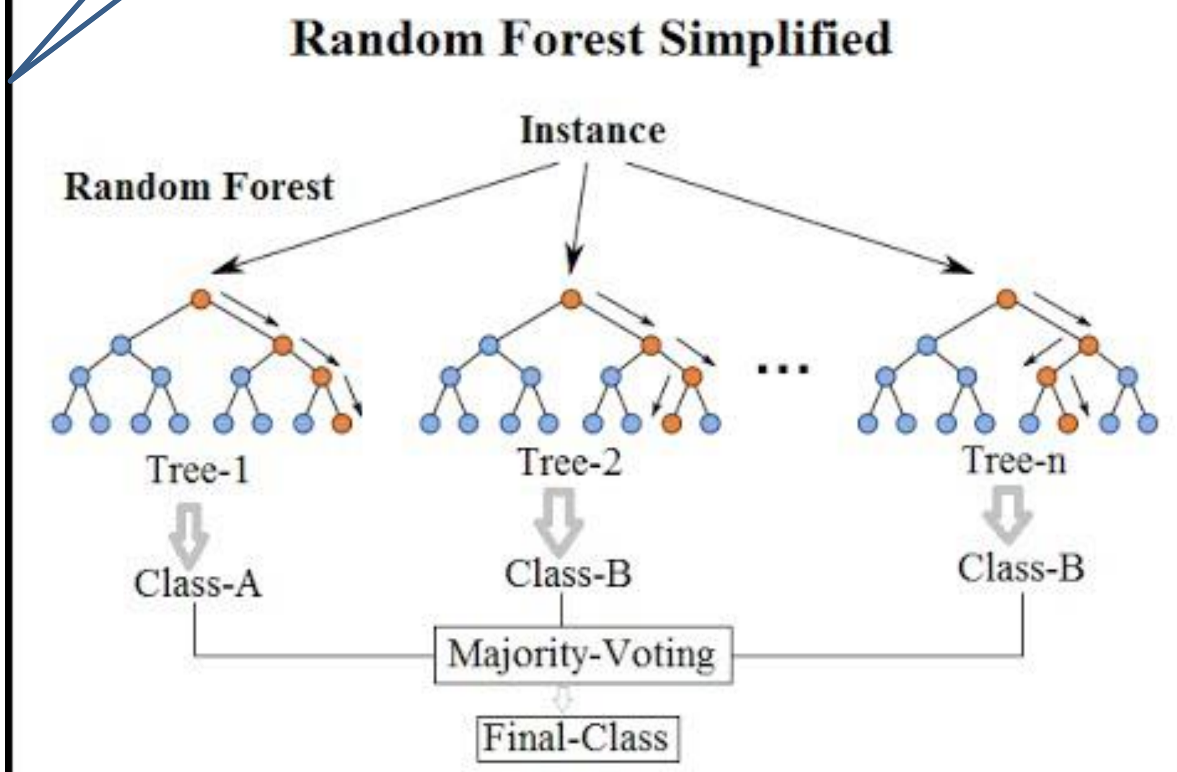


FIGURE 10.1. *Schematic of AdaBoost. Classifiers are trained on weighted versions of the dataset, and then combined to produce a final prediction.*



AdaBoost-Binary Classification

Algorithm 10.1 *AdaBoost.M1.*

1. Initialize the observation weights $w_i = 1/N,\ i = 1, 2, \ldots, N$.
2. For $m = 1$ to M:
   (a) Fit a classifier $G_m(x)$ to the training data using weights w_i.
   (b) Compute
   $$\text{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$$
   (c) Compute $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$.
   (d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))],\ i = 1, 2, \ldots, N$.
3. Output $G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.

<The Elements of Statistical Learning> Page 339

AdaBoost-Example for Step 2(b) to 2(d)

Algorithm 10.1 *AdaBoost.M1.*

1. Initialize the observation weights $w_i = 1/N,\ i = 1, 2, \ldots, N$.
2. For $m = 1$ to M:
   (a) Fit a classifier $G_m(x)$ to the training data using weights w_i.
   (b) Compute
   $$\mathrm{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$$
   (c) Compute $\alpha_m = \log((1 - \mathrm{err}_m)/\mathrm{err}_m)$.
   (d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))],\ i = 1, 2, \ldots, N$.
3. Output $G(x) = \mathrm{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.

AdaBoost-Example for Step 2(b) to 2(d)

Row	x1	x2	x3	y	Weight$^{(m-1)}$
1	0.77	0.35	0.31	1	0.2
2	0.87	0.78	0.59	1	0.2
3	0.34	0.32	0.23	-1	0.2
4	0.02	0.91	0.16	-1	0.2
5	0.61	0.55	0.04	1	0.2

Suppose we have this dataset: N=5

In step m, we train the dataset on y and get model $G_m(x)$

AdaBoost-Example for Step 2(b) to 2(d)

Row	x1	x2	x3	y	Weight$^{(m-1)}$	G_m
1	0.77	0.35	0.31	1	0.2	1
2	0.87	0.78	0.59	1	0.2	1
3	0.34	0.32	0.23	-1	0.2	1
4	0.02	0.91	0.16	-1	0.2	1
5	0.61	0.55	0.04	1	0.2	1

Predicting output using model $G_m(x)$

AdaBoost-Example for Step 2(b) to 2(d)

Row	x1	x2	x3	y	Weight$^{(m-1)}$	G_m	$I(y \neq y^{(m)})$
1	0.77	0.35	0.31	1	0.2	1	0
2	0.87	0.78	0.59	1	0.2	1	0
3	0.34	0.32	0.23	-1	0.2	1	1
4	0.02	0.91	0.16	-1	0.2	1	1
5	0.61	0.55	0.04	1	0.2	1	0

(b) Compute

$$\mathrm{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$$

Compute $\mathrm{err}_m=0.4$

AdaBoost-Example for Step 2(b) to 2(d)

Row	x1	x2	x3	y	$\text{Weight}^{(m-1)}$	G_m	$I(y \neq y^{(m)})$	$\text{Weight}^{(m)}$
1	0.77	0.35	0.31	1	0.2	1	0	0.2
2	0.87	0.78	0.59	1	0.2	1	0	0.2
3	0.34	0.32	0.23	-1	0.2	1	1	0.3
4	0.02	0.91	0.16	-1	0.2	1	1	0.3
5	0.61	0.55	0.04	1	0.2	1	0	0.2

Update weights;
If you do not renormalize and sum up to 1

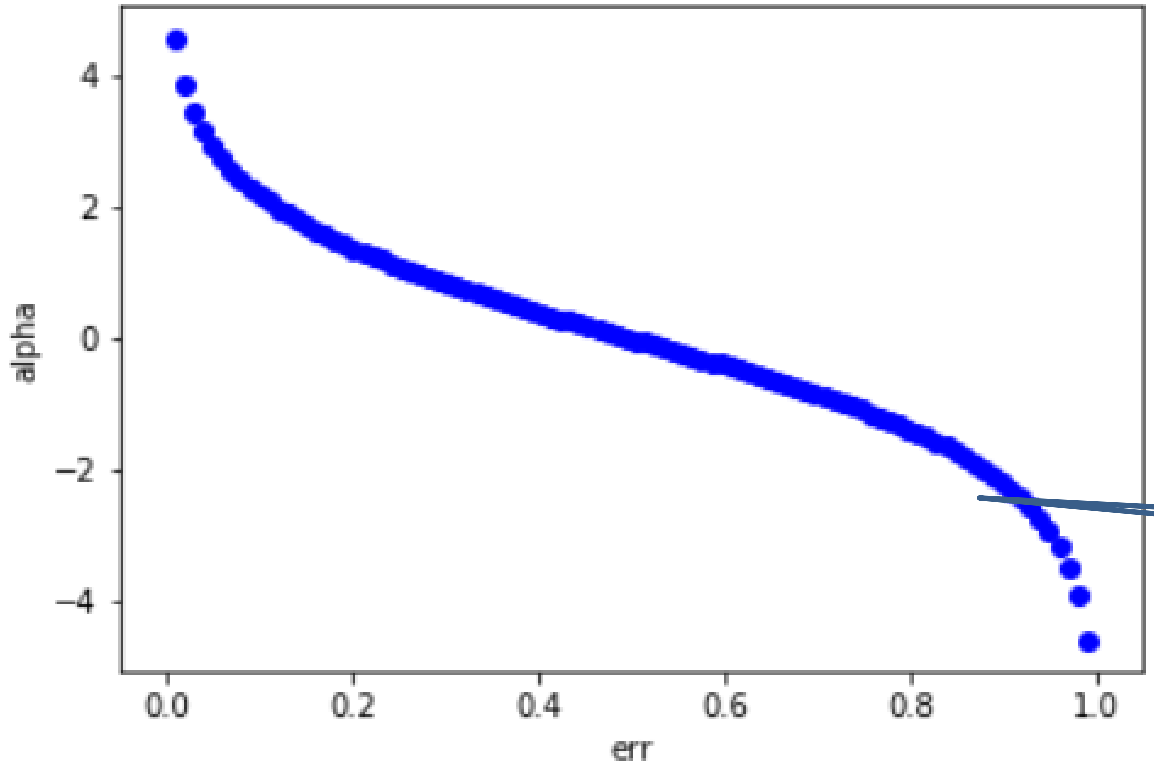
(c) Compute $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$.

(d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))]$, $i = 1, 2, \ldots, N$.

AdaBoost-Example for Step 2(b) to 2(d)

(c) Compute $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$.

(d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))]$, $i = 1, 2, \ldots, N$.



Usually,

If $0 < err_m < 0.5$ and misclassified on i:

$$w_i \leftarrow w_i \times \frac{1 - err_m}{err_m}: w_i \text{ is increasing}$$

Unusually,

If $0.5 < err_m < 1$ and misclassified on i:

$$w_i \leftarrow w_i \times \frac{1 - err_m}{err_m}: w_i \text{ is decreasing}$$

Intuition: If err_m>0.5, model error is even higher than “random guess” (50% accuracy/error), then do the opposite to whatever this model tells you.

AdaBoost-Example for Step 2(b) to 2(d)

Quiz: Do we need a model with high error rate (e.g., Error=90%)?

Suppose 3 Financial Analysts:
Analyst 1. Accuracy=95%, Error=5%
Analyst 2. Accuracy=51%, Error=49%
Analyst 3. Accuracy=5%, Error=95%

Do you think Analyst 3 is a good analyst or not?

AdaBoost-Example for Step 2(b) to 2(d)

Row	x1	x2	x3	y	$Weight^{(m)}$
1	0.77	0.35	0.31	1	0.2
2	0.87	0.78	0.59	1	0.2
3	0.34	0.32	0.23	-1	0.3
4	0.02	0.91	0.16	-1	0.3
5	0.61	0.55	0.04	1	0.2

Goto step m+1;

In step m+1, we train and get new model $G_{m+1}(x)$, and new predictions G_{m+1}, and update weights;

Repeat the procedure until getting M models

AdaBoost-Example for Step 3

Algorithm 10.1 *AdaBoost.M1.*

1. Initialize the observation weights $w_i = 1/N,\ i = 1, 2, \ldots, N$.
2. For $m = 1$ to M:
   (a) Fit a classifier $G_m(x)$ to the training data using weights w_i.
   (b) Compute
   $$\mathrm{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$$
   (c) Compute $\alpha_m = \log((1 - \mathrm{err}_m)/\mathrm{err}_m)$.
   (d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))],\ i = 1, 2, \ldots, N$.
3. Output $G(x) = \mathrm{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.

AdaBoost-Example for Step 3

Suppose you want to predict the output of a new data sample x_{new}

Model m	α_m	$G_m(x_{new})$: Prediction on this new data sample	$\alpha_m * G_m(x_{new})$
1	2	+1	+2
2	1	-1	-1
3	-2	+1	-2

3. Output $G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.

The signum function of a real number x is defined as follows:

$$\operatorname{sgn}(x) := \begin{cases} -1 & \text{if } x < 0, \\ 0 & \text{if } x = 0, \\ 1 & \text{if } x > 0. \end{cases}$$

Output of x_{new}: sign(-1) = -1

Intuition: If err_m>0.5, model error is even higher than “random guess” (50% accuracy/error), then do the opposite to whatever this model tells you.

AdaBoost-Example for Step 2(a)

Algorithm 10.1 *AdaBoost.M1.*

1. Initialize the observation weights $w_i = 1/N,\ i = 1, 2, \ldots, N$.
2. For $m = 1$ to M:
   (a) Fit a classifier $G_m(x)$ to the training data using weights w_i.
   (b) Compute
   $$\text{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$$
   (c) Compute $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$.
   (d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))],\ i = 1, 2, \ldots, N$.
3. Output $G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.

Select and extract a classifier G_m from the pool of classifiers, which minimizes err_m

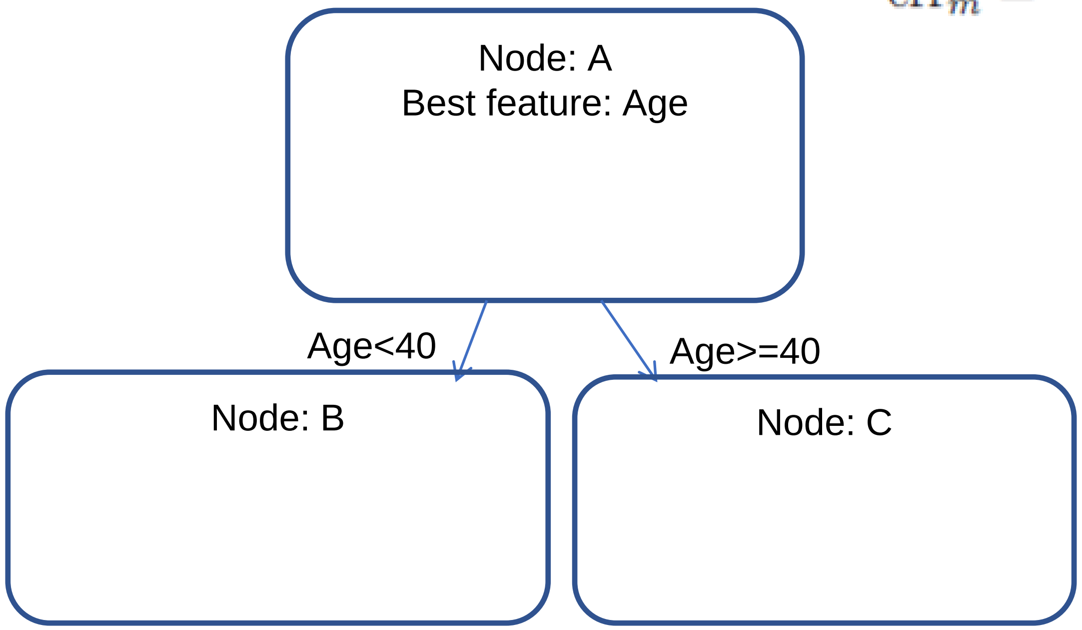
$$\text{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$$

http://www.inf.fu-berlin.de/inst/ag-ki/adaboost4.pdf

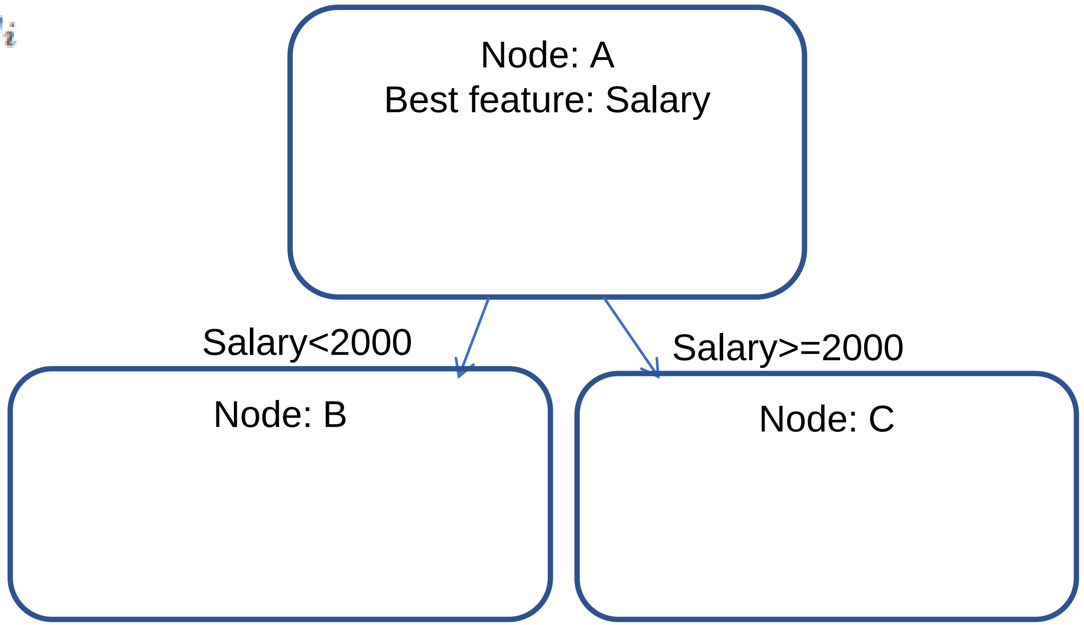
AdaBoost-Example for Step 2(a)

Select and extract a classifier G_m from the pool of classifiers, which minimizes err_m

$$\mathrm{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$$



Model 1: err=40%



Model 2: err=10%

Which model do you choose in this step m?

AdaBoost-Binary Classification

Algorithm 10.1 *AdaBoost.M1.*

1. Initialize the observation weights $w_i = 1/N,\ i = 1, 2, \ldots, N$.
2. For $m = 1$ to M:
   (a) Fit a classifier $G_m(x)$ to the training data using weights w_i.
   (b) Compute
   $$\mathrm{err}_m = \frac{\sum_{i=1}^N w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^N w_i}.$$
   (c) Compute $\alpha_m = \log((1 - \mathrm{err}_m)/\mathrm{err}_m)$.
   (d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))],\ i = 1, 2, \ldots, N$.
3. Output $G(x) = \mathrm{sign}\left[\sum_{m=1}^M \alpha_m G_m(x)\right]$.

Limitation:
Adaboost may still have overfit issue.

Ensemble Performance

Performance of Binary Classifier

- Confusion Matrix

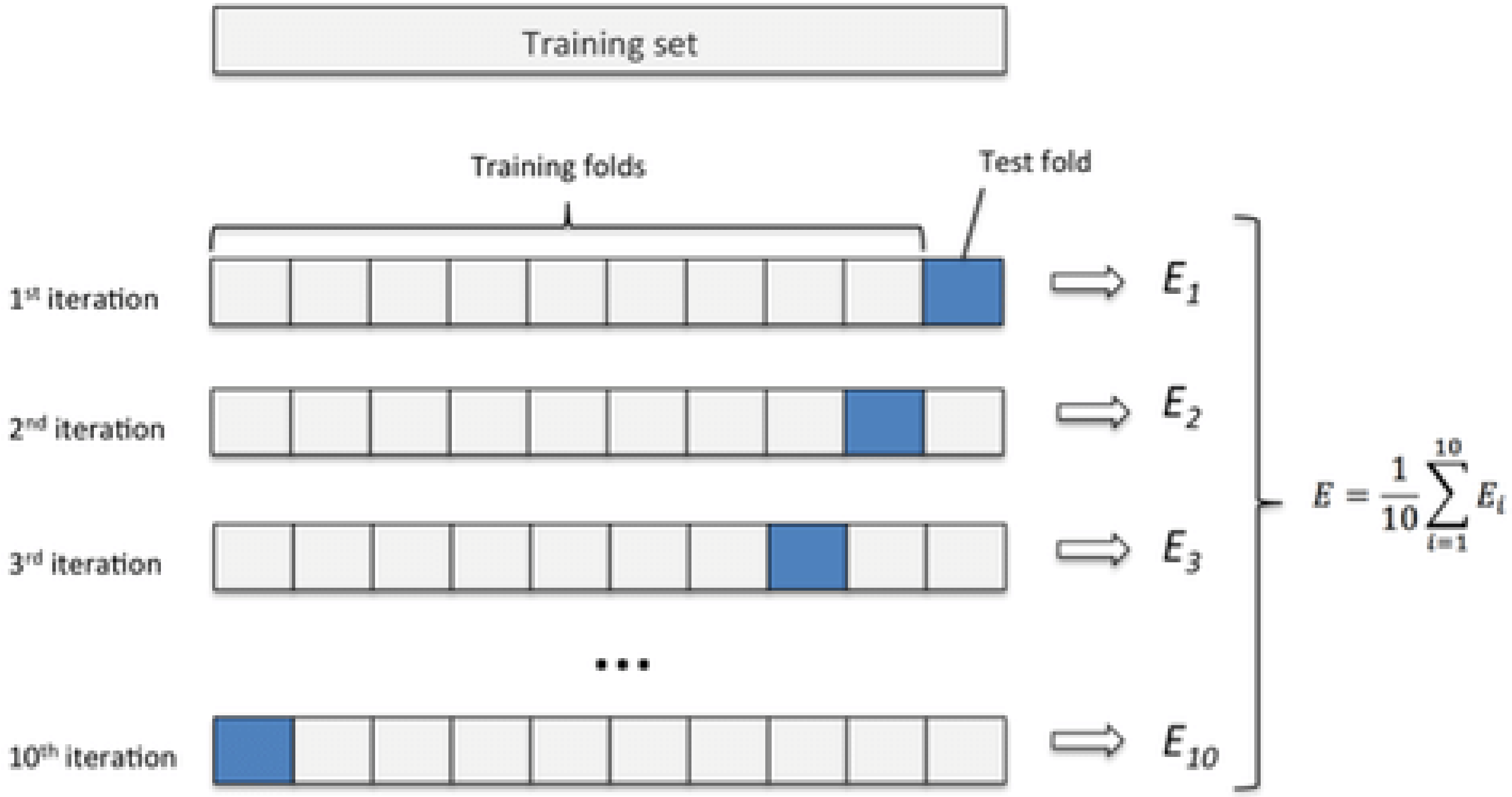
Correct classification	Classified as	
	+	−
+	true positives	false negatives
−	false positives	true negatives

True Positive Rate or Hit Rate or Recall or Sensitivity or TP Rate	TP/P	The proportion of positive instances that are correctly classified as positive
False Positive Rate or False Alarm Rate or FP Rate	FP/N	The proportion of negative instances that are erroneously classified as positive
False Negative Rate or FN Rate	FN/P	The proportion of positive instances that are erroneously classified as negative = 1 − True Positive Rate

True Negative Rate or Specificity or TN Rate	TN/N	The proportion of negative instances that are correctly classified as negative
Precision or Positive Predictive Value	TP/(TP+FP)	Proportion of instances classified as positive that are really positive
F1 Score	(2 × Precision × Recall) /(Precision + Recall)	A measure that combines Precision and Recall
Accuracy or Predictive Accuracy	(TP + TN)/(P + N)	The proportion of instances that are correctly classified
Error Rate	(FP + FN)/(P + N)	The proportion of instances that are incorrectly classified

Cross Validation

- K-Fold Cross Validation (e.g., K=10)



- Model Evaluation
- Model Comparison
- Model Tuning

All observations are used for both training and validation, and each observation is used for validation exactly once.

Python Implementation

Implementation in Python

BT2101 Introduction to Ensemble Learning

1 Goal

In this notebook, we will explore **Ensemble Learning** including:

- Bagging
- Random Forest
- AdaBoost

For the **Decision Tree** method, you will:

- Use open-source package to do ensemble learning
- Compare performances of different methods

```
# -*- coding:utf-8 -*-
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from math import sqrt
from __future__ import division
%matplotlib inline
```

2 Ensemble Learning

Programming Assignment 3

Using the BT2101 Tutorial 3 Programming code (Ensemble Learning.ipynb), please answer the questions in the jupyter notebook

Answer all in the jupyter notebook.

Instructions

Submit Python Notebook to IVLE folder, and Naming the file:
AXXXX_T3_program.ipynb

Include your answers in the jupyter notebook

- You need to show outputs, instead of just showing functions.

Submit a FINAL program by Tue Sep-18 (by lunchtime)

- Based on Ensemble Learning.ipynb

Thank you!

Appendix: AdaBoost-MultiClass Classification

Algorithm 1 *AdaBoost (Freund & Schapire 1997)*

1. *Initialize the observation weights* $w_i = 1/n,\ i = 1, 2, \ldots, n$.
2. *For* $m = 1$ *to* M*:*
   (a) *Fit a classifier* $T^{(m)}(\boldsymbol{x})$ *to the training data using weights* w_i.
   (b) *Compute*
   $$err^{(m)} = \sum_{i=1}^{n} w_i \mathbb{I}\left(c_i \neq T^{(m)}(\boldsymbol{x}_i)\right) / \sum_{i=1}^{n} w_i.$$
   (c) *Compute*
   $$\alpha^{(m)} = \log \frac{1 - err^{(m)}}{err^{(m)}}.$$
   (d) *Set*
   $$w_i \leftarrow w_i \cdot \exp\left(\alpha^{(m)} \cdot \mathbb{I}\left(c_i \neq T^{(m)}(\boldsymbol{x}_i)\right)\right),\ i = 1, 2, \ldots, n.$$
   (e) *Re-normalize* w_i.
3. *Output*
   $$\boxed{C(\boldsymbol{x}) = \arg\max_k \sum_{m=1}^{M} \alpha^{(m)} \cdot \mathbb{I}(T^{(m)}(\boldsymbol{x}) = k).}$$

https://web.stanford.edu/~hastie/Papers/samme.pdf