Pattern Recognition
Lecture 02-1

Random Forest & Linear Classification

Prof. Jongwon Choi
Chung-Ang University
Fall 2022

Decision Tree

- **Warning!!**
  - The architecture of decision tree can be very various
  - The decision tree can be utilized for a lot of applications
  - However, we will target on “Supervised classification learning”

Decision Tree

- **Supervised learning?**
  - Machine learning when the data are fully categorized or labelled.

Peanut	Fish	Meat	Wheat	Water	Egg	Milk
0	0.1	0	0	0.1	0.1	0
0.3	0.2	0.9	0	0.9	0.8	0
0	0.8	0.3	0.5	0.4	0.1	0.2
0	0	0.8	0.2	0	0	0.1
0.5	0.1	0.2	0.9	0.2	0	0.3

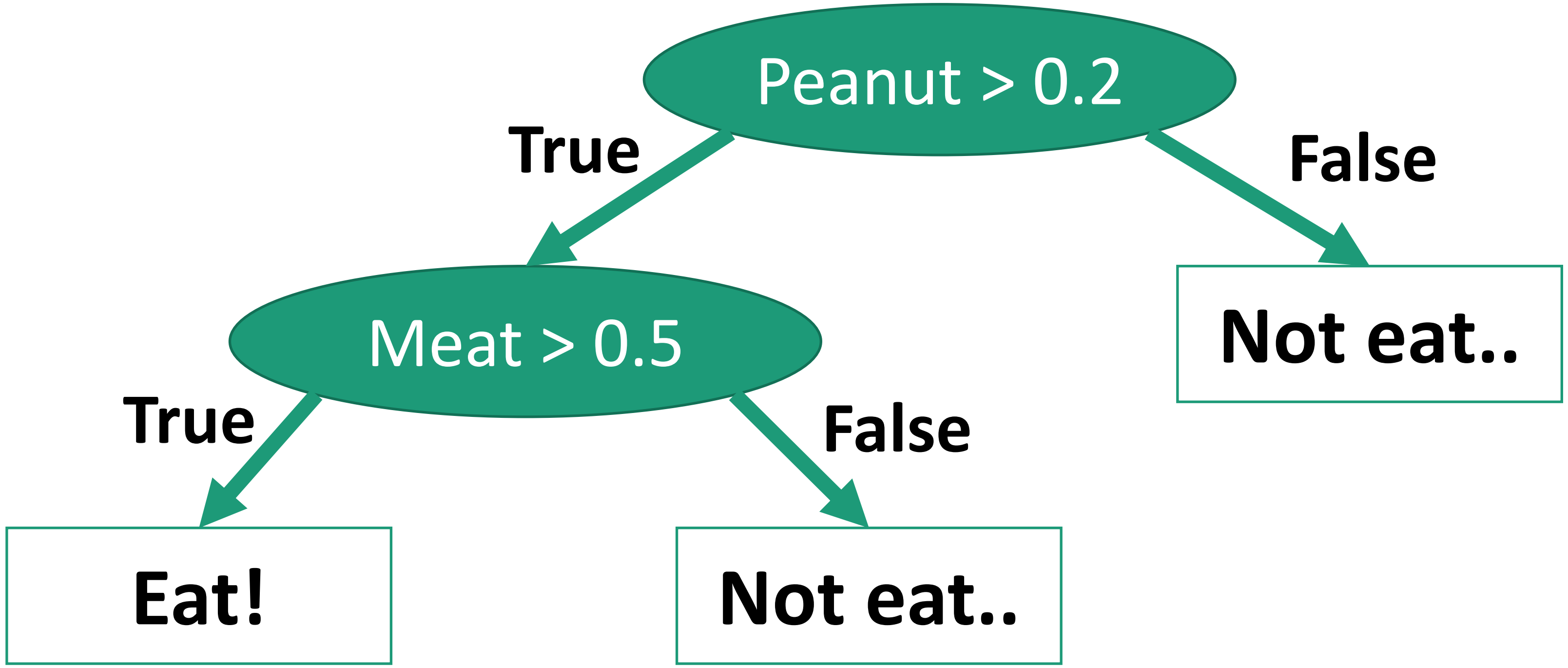
Dog eats?
0
1
0
1
0

Features (input) : Quantities of various ingredients

Labels (output) : Whether or not the dog eats

Decision Tree

- **Decision tree is a simple program**
  - Splitting rule – one splitting node decide “if-else” according to the features
  - Class prediction – the class label is annotated at the last node (leaf)



Decision Tree

- **There are many possible decision trees!**
  - We can change the splitting feature types
  - We can change the splitting thresholds
  - We can change the stop criterion (tree depth)

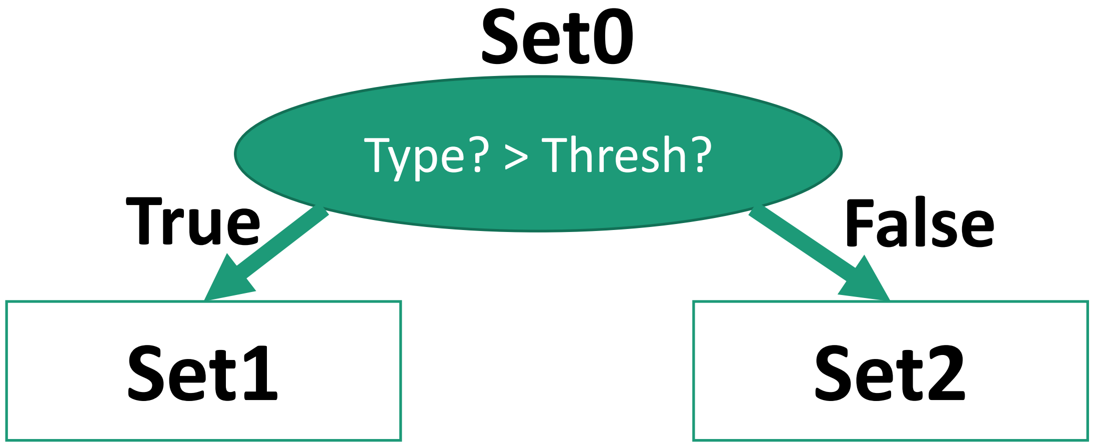
- Among the various decision trees,
  - **We need to fine the best model to represent the given data!**
  - **This is called “training” the supervised learning model**

Decision Tree

- **Let's find the best splitting node!**

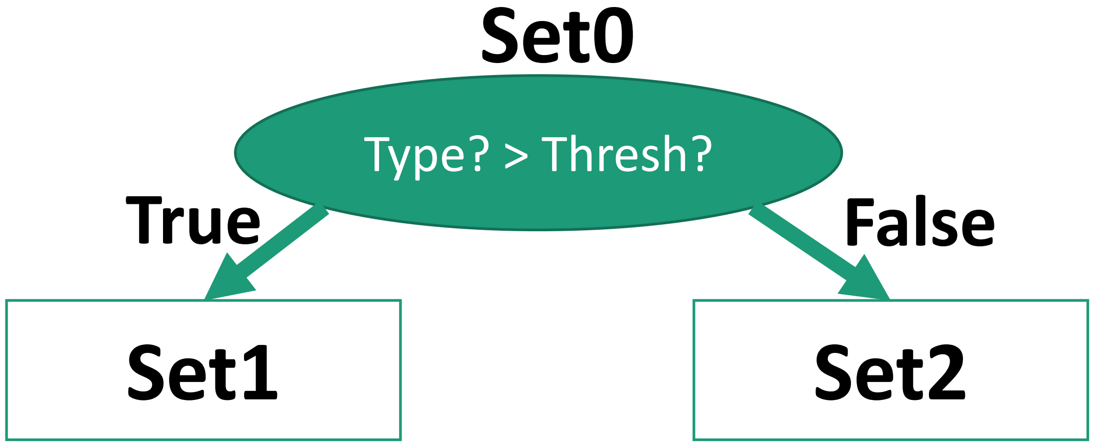
Peanut	Fish	Meat	Wheat	Water	Egg	Milk
0	0.1	0	0	0.1	0.1	0
0.3	0.2	0.9	0	0.9	0.8	0
0	0.8	0.3	0.5	0.4	0.1	0.2
0	0	0.8	0.2	0	0	0.1
0.5	0.1	0.2	0.9	0.2	0	0.3

Dog eats?
0
1
0
1
0



Decision Tree

- **Let's find the best splitting node!**
  - There can be various method to score the combination of feature type and threshold.
    - 1. Split the input samples into two balanced sets
    - 2. Split the input samples to obtain the highest accuracy
    - 3. Split the input samples to result one perfect leaf node
    - 4. etc…



Decision Tree

- **Let's find the best splitting node!**
  - 1. Split the input samples into two balanced sets

Peanut	Fish	Meat	Wheat	Water	Egg	Milk
0	0.1	0	0	0.1	0.1	0
0.3	0.2	0.9	0	0.9	0.8	0
0	0.8	0.3	0.5	0.4	0.1	0.2
0	0	0.8	0.2	0	0	0.1
0.5	0.1	0.2	0.9	0.2	0	0.3

Dog eats?
0
1
0
1
0

Peanut > 0.2

Meat > 0.5

Egg – Impossible

Milk > 0.1

Decision Tree

- **Let’s find the best splitting node!**
  - 2. Split the input samples to obtain the highest accuracy

Peanut	Fish	Meat	Wheat	Water	Egg	Milk
0	0.1	0	0	0.1	0.1	0
0.3	0.2	0.9	0	0.9	0.8	0
0	0.8	0.3	0.5	0.4	0.1	0.2
0	0	0.8	0.2	0	0	0.1
0.5	0.1	0.2	0.9	0.2	0	0.3

Dog eats?
0
1
0
1
0

Meat > 0.7

Decision Tree

- **Let’s find the best splitting node!**
  - 3. Split the input samples to result one perfect leaf node

Peanut	Fish	Meat	Wheat	Water	Egg	Milk
0	0.1	0	0	0.1	0.1	0
0.3	0.2	0.9	0	0.9	0.8	0
0	0.8	0.3	0.5	0.4	0.1	0.2
0	0	0.8	0.2	0	0	0.1
0.5	0.1	0.2	0.9	0.2	0	0.3

Dog eats?
0
1
0
1
0

Meat > 0.8
Milk > 0.2
Peanut > 0.4

Decision Tree

- **Supervised learning notation**

$\mathbf{X} =$

Peanut	Fish	Meat	Wheat	Water	Egg	Milk
0	0.1	0	0	0.1	0.1	0
0.3	0.2	0.9	0	0.9	0.8	0
0	0.8	0.3	0.5	0.4	0.1	0.2
0	0	0.8	0.2	0	0	0.1
0.5	0.1	0.2	0.9	0.2	0	0.3

n

d

$\mathbf{b} =$

Dog eats?
0
1
0
1
0

n

Decision Tree

- **Cost of the best split estimation**
- **Assume that:**
  - 'n' samples
  - 'd' feature types
  - 'k' discrete thresholds

- We compute "n" labels for "k*d" combinations
  - $O(ndk)$

- Thus, sometimes, we decide the feature types randomly!
- When k is small, the computation reduces much

Decision Tree

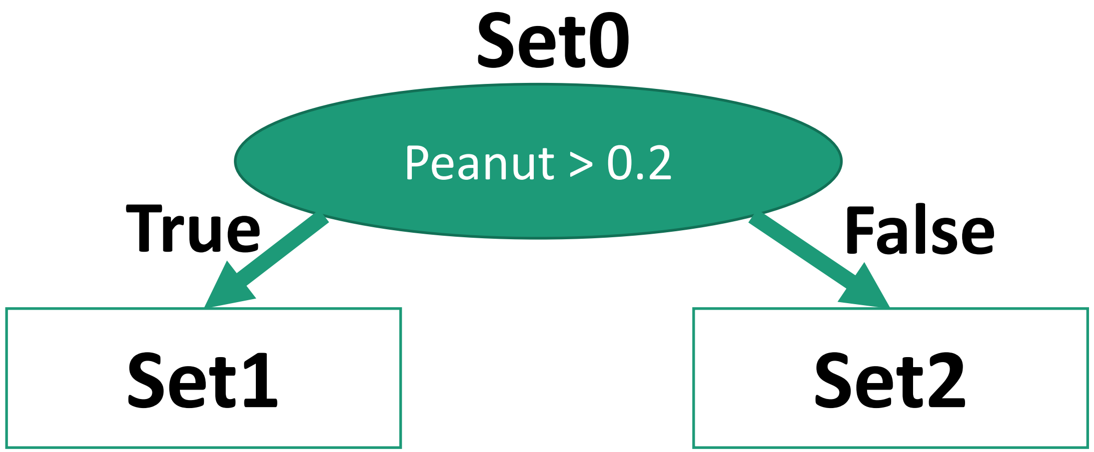
- **Decision tree learning (Sequential)**
  - It is computationally infeasible to find the best decision tree!
  - We need to try every combination of sequential split nodes

- Most common decision tree learning algorithm in practice:
  - Greedy recursive splitting

Decision Tree

Peanut	...
0	
0.3	
0	
0	
0.5	

Dog eats?
0
1
0
1
0

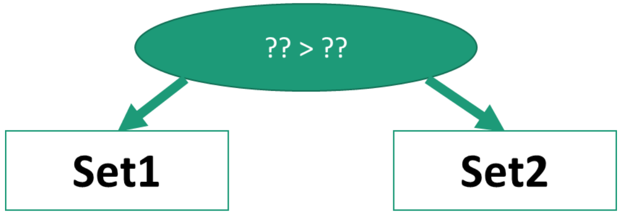


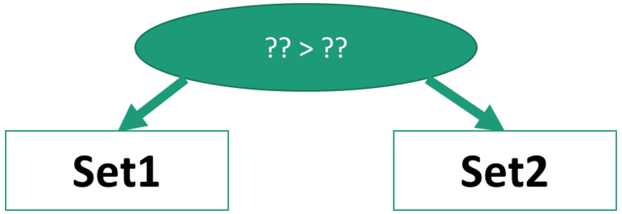
Peanut	...
0.3	
0.5	

Dog eats?
1
0

Peanut	...
0	
0	
0	

Dog eats?
0
0
1





Decision Tree

- **Decision tree learning (Sequential)**
  - It is computationally infeasible to find the best decision tree!
  - We need to try every combination of sequential split nodes

- Most common decision tree learning algorithm in practice:
  - Greedy recursive splitting
  - **Then, until when???**
    - When all the samples are categorized well?
    - Or, we can define the maximum depth value

➡ **Actually, this problem is not that easy…**

Training Generalization

- With the infinite depth,
  - The <u>**training accuracy**</u> is ‘1.0’ (because one leaf can contain one sample)
  - It perfectly labels the data we used to train the decision tree

  - Then, for prediction, some additional samples are given,
    - What is the <u>**testing accuracy**</u> on the new data?
    - Conventionally, the testing accuracy becomes much low…

  - Overfitting: Lower accuracy on new (test) data
    - The model gets too specific to the training dataset
    - However, our goal of supervised learning was “prediction”!

Training Generalization

- **Memorization vs. Learning**
  - Memorization : Only can do well on the training data
  - Learning : Generalize the model on various situations

- The problem is…
  - **THE TEST DATA CANNOT INFLUENCE THE TRAINING PHASE IN ANY WAY**

Training Generalization

- **THE TEST DATA CANNOT INFLUENCE THE TRAINING PHASE IN ANY WAY**

- **Thus, we need some assumptions on training/testing data**
  - The training and test data need to be related in some way
  - Most common assumption: independent and identically distributed (IID)

Training Generalization

- **IID Assumption**
  - All examples come from the same distribution
  - The examples are sampled independently (order doesn’t matter)

- Examples
  - Pick a card, put it back in the deck, re-shuffle, repeat
  - Pick a card, put it back in the deck on the bottom, repeat
  - Pick a card, re-shuffle, repeat

Training Generalization

- **IID Assumption**
  - All examples come from the same distribution
  - The examples are sampled independently (order doesn’t matter)

- Actually, the IID assumption is rarely true:
  - But it is often a good approximation
- We do not assume the IID across features!!

Training Generalization

- **Amount of overfitting**
  - **When we define the amount of overfitting by:**
  - $\boldsymbol{E_{approx} = E_{test} - E_{train}}$,

  - It tends to get smaller as the number of samples gets larger
    - Small dataset vs. Large dataset
  - It tends to grow as model get more “complicated”
    - Small depth vs. Large depth

Training Generalization

- **Validation error**
  - Split the training set into a partial training set and a validation set

	Peanut	Fish	Meat	Wheat	Water	Egg	Milk	Dog eats?
Training Set	0	0.1	0	0	0.1	0.1	0	0
	0.3	0.2	0.9	0	0.9	0.8	0	1
	0	0.8	0.3	0.5	0.4	0.1	0.2	0
Validation Set	0	0	0.8	0.2	0	0	0.1	1
	0.5	0.1	0.2	0.9	0.2	0	0.3	0

Training Generalization

- **Cross Validation**
  - Ex. 3-fold cross validation

Fold 1:	**Fold 2:**	**Fold 3:**
Train	**Train**	**Validation**
Train	**Validation**	**Train**
Validation	**Train**	**Train**
E_{Valid1}	E_{Valid2}	E_{Valid3}

$$E_{Cross-Valid} = (E_{Valid1} + E_{Valid2} + E_{Valid3})/3$$

Parameter & Hyper-parameter

- Parameter – The values that are estimated by the training algorithm
- Hyper-parameter – The control values conventionally defined by user
  - We use the validation error to find the best hyper-parameter

- When the splitting optimization algorithm is fixed,

- **There are many possible decision trees!**
  - We can change the splitting feature types
  - We can change the splitting thresholds
  - We can change the stop criterion (tree depth)

Parameter ← (splitting feature types, splitting thresholds)

Hyper-parameter ← (stop criterion)

Ensemble Models - Definition

- **How can we acquire the accurate & real-time classifiers?**
  - Decision tress and naïve Bayes are fast, but not accurate
  - k-NN is accurate, but not fast

- **We can consider the 'ensemble' model**
  - Ensemble model – Classifiers that have classifiers as input
  - Also called 'meta classifier'
  - Ex. Averaging, boosting, bootstrapping, bagging, cascading, etc.
  - The ensemble model often show higher accuracy than separated inpu classifier

Ensemble Models - Averaging

- **Input to averaging is the predictions of a set of models:**
  - A prediction from a decision tree
  - A prediction from another decision tree
  - A prediction from naïve Bayes
  - A prediction from k-NN
  - Etc.

- **Simple model averaging:**
  - Take the mode of the predictions (i.e. average probabilities)

Ensemble Models - Averaging

- **A common variation is ‘stacking’**
  - Fit another classifier that uses the predictions as features

- **Averaging/stacking often performs bettern than individual models**
  - Typically used by Kaggle winners!
  - Most of saturated research area conventionally utilize these models
  - Since the separated input classifers can be run in parallel, these models are often used for real applications and systems.

Ensemble Models - Averaging

- **Why can Averaging work??**
  - When there are three independent classifiers with probability 0.80,

  - $p(all\ 3\ right) = 0.8^3 = 0.512$
  - $p(2\ right,\ 1\ wrong) = 3 * 0.8^2(1 - 0.8) = 0.384$
  - $p(1\ right,\ 2\ wrong) = 3 * 0.8(1 - 0.8)^2 = 0.096$
  - $p(all\ 3\ wrong) = (1 - 0.8)^3 = 0.008$
  - Thus, the ensemble model's probability is 0.896 (0.512 + 0.384)

Ensemble Models – Random Forest

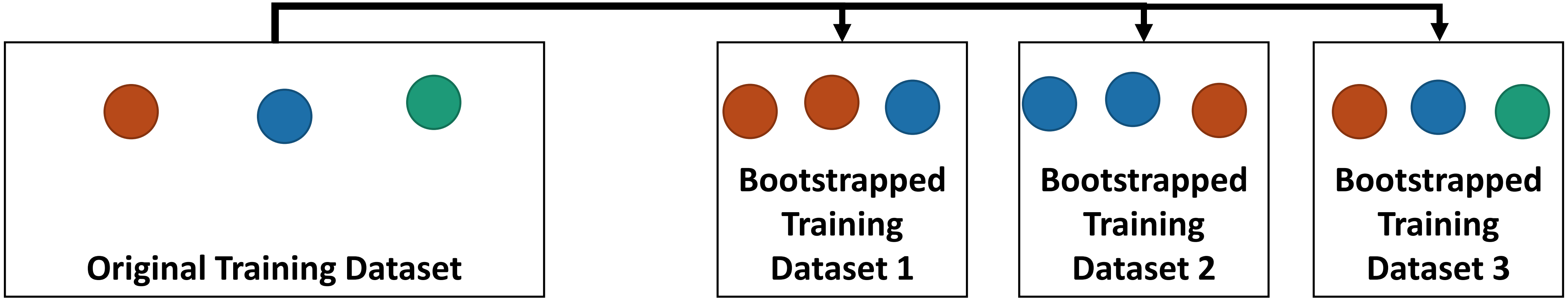
- **Random forests average a set of deep decision trees**
  - One of the best classifiers for real applications
  - Any predictions are very fast (Especially, in parallel)

- However, the multiple decision trees are not independent!
  - Since we train the decision trees with same training data and rules

- To solve the problem, the random forest utilizes:
  - Booststrapping
  - Random trees

Ensemble Models – Random Forest

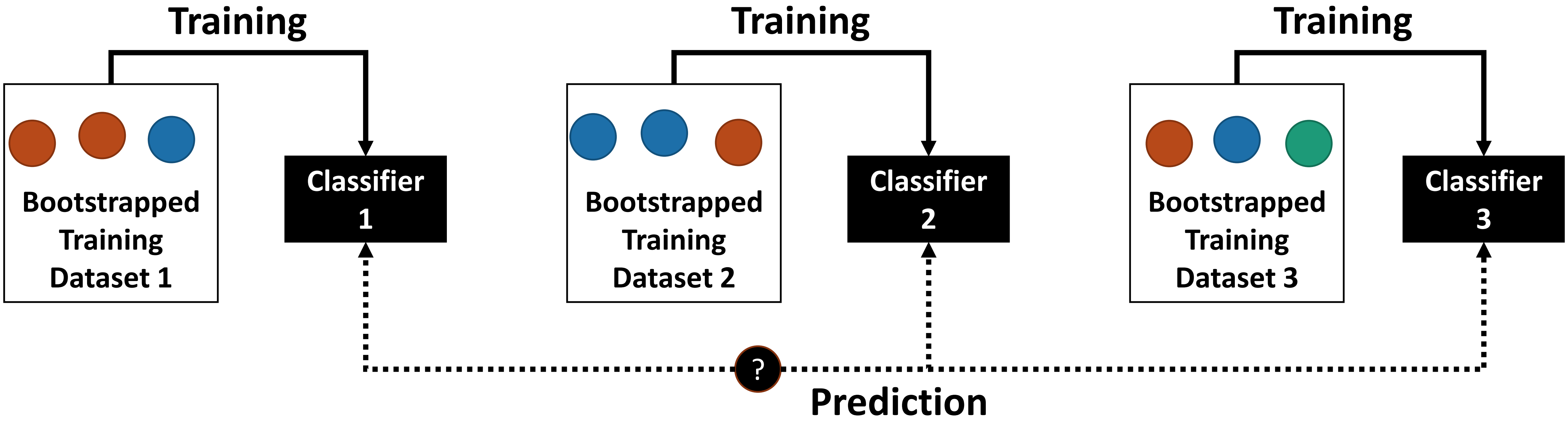
- **Bootstrap sampling**
  - Extract a new training dataset from the original training dataset
  - Randomly select samples from the original training dataset
    - Then, some samples can be duplicated or missing



  - Roughly maintain the trends
  - We can obtain the varying data from one training data

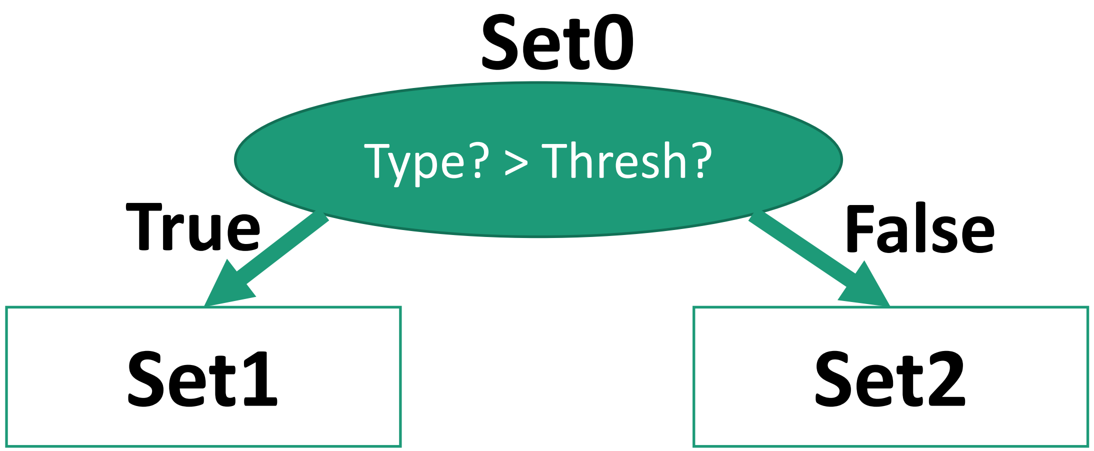
Ensemble Models – Random Forest

- **Bagging**
  - Use the boostrap samples for ensemble learning
    - Generate several bootstrap sample sets
    - Fit a classifier to each bootstrap sample set
    - At test time, average the prediction!



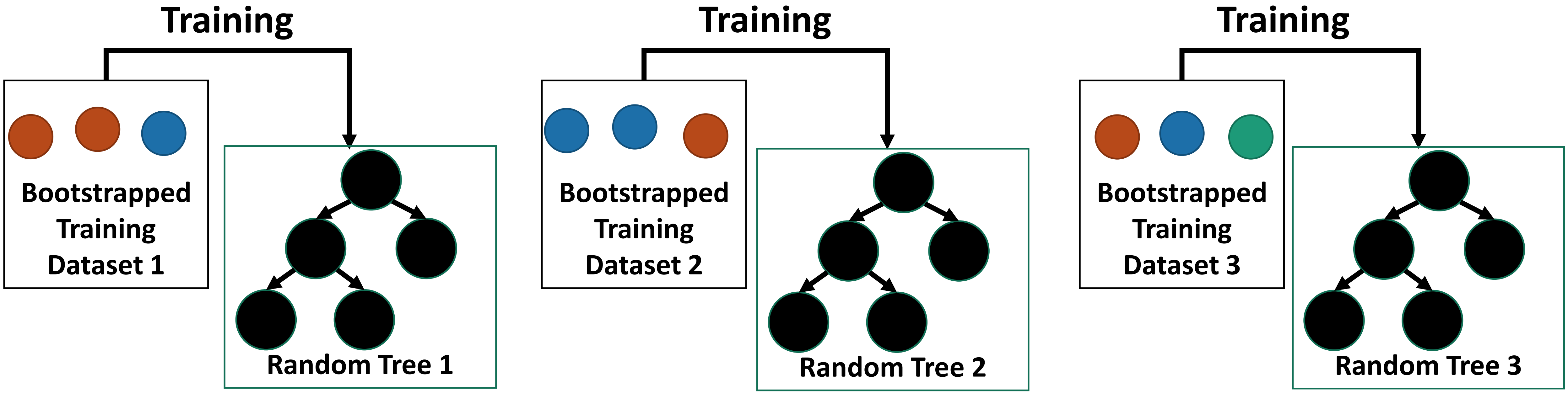
Ensemble Models – Random Forest

- **Random trees**
  - Conventional binary decision tree
  - But, the feature type is randomly chosen at every node
  - We can simply reduce the dependency among the multiple decision trees



Ensemble Models – Random Forest

- **Random Forest**
  - Bagging + Random Trees
  - The multiple decision trees become independent (not perfectly), and the resulting ensemble model often works well



Ensemble Models – Boosting

- **Cascade classifier**
  - A meta classifier consists of several classifiers that are applied subsequently
- **Boosting**
  - When a sample passes all the classifiers, it is classified as the final label
  - Only a part of classifiers are applied until the sample is rejected
  - When there are a number of outliers, it works rapidly and efficiently