Artificial Intelligence

Instructor: Kietikul Jearanaitanakij

Department of Computer Engineering

King Mongkut’s Institute of Technology Ladkrabang

Lecture 8
Learning from examples

- Forms of Learning
- Supervised Learning
- Learning Decision Trees
- Evaluating the Hypothesis

- An agent is learning if it improves its performance on future tasks.

ทำไมต้องมีการ learning

- Why would we want an agent to learn? Why wouldn't the designers just program the agent to do the task?

ไม่สามารถเขียน logic ครอบคลุมทั้งหมดได้

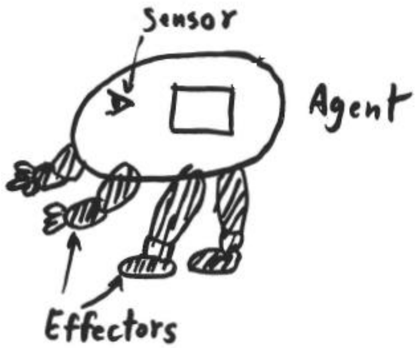
- There are three main reasons.
- **First**, the designers cannot anticipate all possible situations that the agent might find itself in. For example, a robot designed to navigate mazes must learn the layout of each new maze it encounters.
- **Second**, the designers cannot anticipate all changes over time; a program designed to predict tomorrow's stock market prices must learn to adapt when conditions change from boom to bust.

ไม่สามารถคาดการณ์การเปลี่ยนแปลงในอนาคต

- **Third**, sometimes human programmers have no idea how to program a solution themselves. For example, most people are good at recognizing the faces of family members, but even the best programmers are unable to program a computer to accomplish that task, except by using learning algorithms.

เขียน algorithm ไม่ออก → การจดจำใบหน้า

Forms of learning



1. Unsupervised learning → มี x ไม่มี y

- The agent learns patterns in the input even though no explicit label is supplied.
- The most common unsupervised learning task is clustering.
- For example, a taxi agent might gradually develop a concept of “Good traffic days” and “Bad traffic days” without ever being given labeled examples of each by a teacher.

ไม่มี y บอก

มันจะสังเกต input เรื่อยๆ

แบ่งแยกกลุ่ม

  - X1: Rain, BigEvent, WorkDay, MonthEnd
  - X2: NonRain, NonBigEvent, NonWorkDay, NonMonthEnd
  - X3: Rain, NonBigEvent, WorkDay, NonMonthEnd
  - X4: NonRain, BigEvent, NonWorkDay, MonthEnd

A taxi agent may cluster X1, X4 as one cluster and X2, X3 as another cluster.

มันไม่รู้ว่าอันไหน good or bad

ไม่เกี่ยวกับการแนะนำได้

2. Reinforcement learning → การทำให้เป็นเก่งขึ้นเรื่อยๆ → เราต้องทำการวน loop หลายๆรอบ

per loop get reward or punishment

- The agent learns from a series of reinforcements – rewards or punishments.
- For example, the lack of a tip at the end of the journey gives the taxi agent an indication that it did something wrong.
- It is up to the agent to decide which of the actions prior to the reinforcement were most responsible for it.

สมมติ X_i , X_{i+1} , X_{i+3} ; เป็น action

reward กลับมา → เรารู้ว่าประมาณนี้ดี

punishment กลับมา → เรารู้ว่ายังแย่ต้องไปปรับให้ดี

ปรับ action

เป็นการเรียนรู้ policy

3. Supervised learning → ที่เรียน รู้ x รู้ y

- The agent observes some example input-output pairs and learns a function that maps from input to output.

- For example, from a taxi agent problem. By using given labeled "input-output" examples, a taxi agent will learn a function that maps form input to the outputs "Good traffic days" and "Bad traffic days".

x y

  - X1: Rain, BigEvent, WorkDay, MonthEnd, "Bad traffic days"
  - X2: NonRain, NonBigEvent, NonWorkDay, NonMonthEnd, "Good traffic days"
  - X3: Rain, NonBigEvent, WorkDay, NonMonthEnd, "Good traffic days"
  - X4: NonRain, BigEvent, NonWorkDay, MonthEnd, "Bad traffic days"

Supervised Learning

The task of supervised learning is this:

> Given a **training set** of N example input–output pairs
>
> $$(x_1, y_1), (x_2, y_2), \ldots (x_N, y_N) ,$$
>
> where each y_j was generated by an unknown function $y = f(x)$, discover a function h that approximates the true function f.

ประมวลตามสมบัติของข้อมูล → ให้ h ใกล้ f ที่สุด

- Learning is a search through the space of possible hypotheses for one that will perform well, even on new examples beyond the training set.
- To measure the accuracy of a hypothesis we give it a test set of examples that are distinct from the training set.
- We say a hypothesis **generalizes** well if it correctly predicts the value of y for novel examples.

สามารถทำนายสิ่งที่ไม่เคยเห็นได้

- When the output **y** is one of a **finite set of values** (such as sunny, cloudy or rainy), the learning problem is called **classification**, and is called Boolean or binary classification if there are only two values.
- When **y** is a **real number** (such as tomorrow's temperature), the learning problem is called **regression**.

Overfitting

ตรงตามจริงข้อมูล

- Overfitting occurs when a statistical learning model describes random errors or noises instead of the underlying relationship.
  - A learning model that has been **overfit** will generally have poor predictive performance, as it can exaggerate minor fluctuations in the data.
  - A good learning model should be **generalize** to both unseen and training data.

ผมไปเจอที่ไม่เคย → แตก

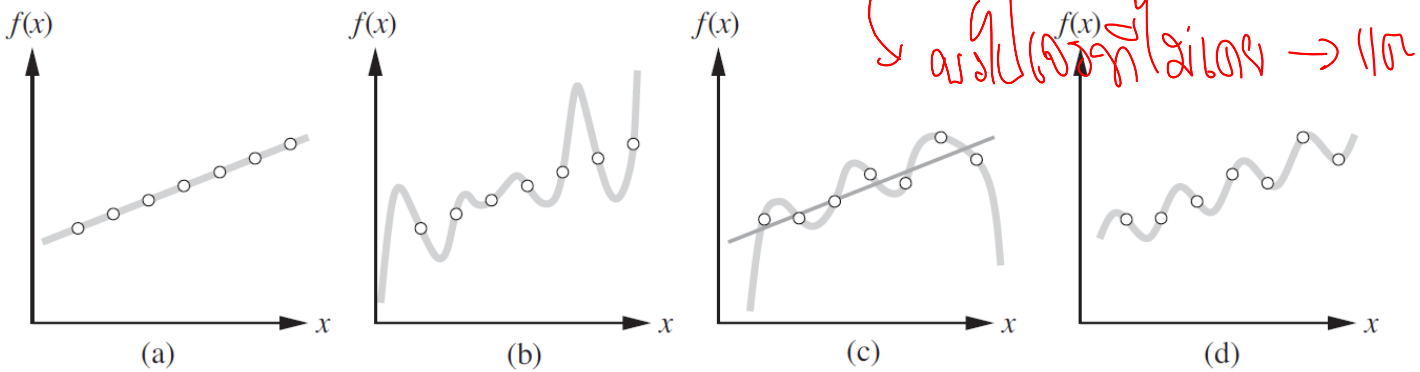


Fig. a : some data with an exact fit by a straight line (the polynomial 0.4x + 3)

Fig. b : a high-degree polynomial that is also consistent with the same data.

Fig. c : A straight line that is not consistent with any of the data points, but might generalize fairly well for unseen values of x.

Fig. d : data in can be fitted exactly by a function of the form ax + b + c sin(x). => tend to overfitting.

Learning Decision Trees (ID3 algorithm)

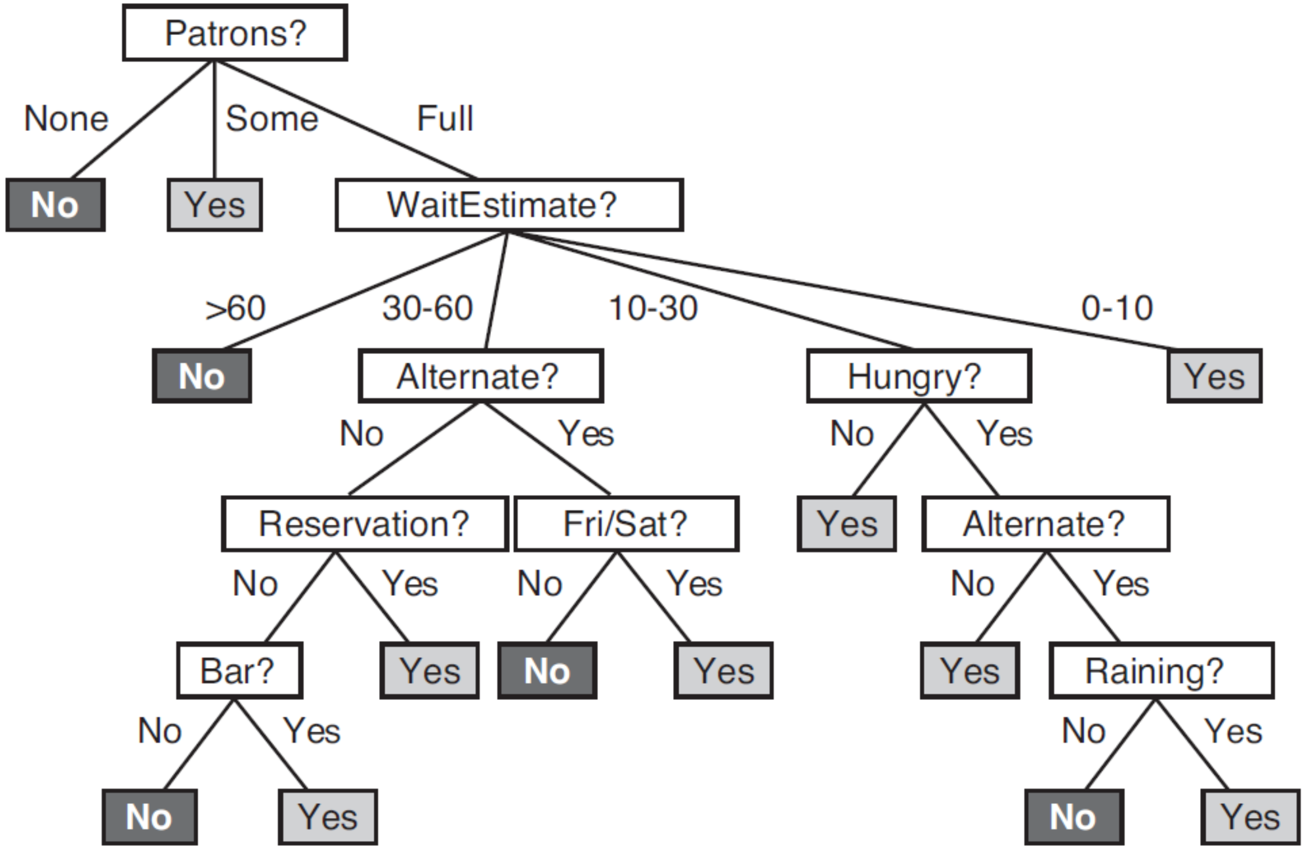
- A decision tree represents a function that takes as input a vector of attribute values and returns a "decision"—a single output value.
- As an example, we will build a decision tree to decide whether to wait for a table at a restaurant. The aim here is to learn a definition for the goal predicate WillWait .
- First we list the attributes that we will consider as part of the input

1. **Alternate**: whether there is a suitable alternative restaurant nearby.
2. **Bar** : whether the restaurant has a comfortable bar area to wait in.
3. **Fri/Sat**: true on Fridays and Saturdays.
4. **Hungry**: whether we are hungry.
5. **Patrons**: how many people are in the restaurant (values are None, Some, and Full).
6. **Price**: the restaurant's price range ($, $$, $$$).
7. **Raining**: whether it is raining outside.
8. **Reservation**: whether there is a reservation.
9. **Type**: the kind of restaurant (French, Italian, Thai, or burger).
10. **WaitEstimate**: the wait estimated by the host (0–10 minutes, 10–30, 30–60, or >60).

Examples for the restaurant domain.

Example	Input Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	$y_1 =$ *Yes*
$\mathbf{x}_2$	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	$y_2 =$ *No*
$\mathbf{x}_3$	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	$y_3 =$ *Yes*
$\mathbf{x}_4$	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*Yes*	*No*	*Thai*	*10–30*	$y_4 =$ *Yes*
$\mathbf{x}_5$	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	$y_5 =$ *No*
$\mathbf{x}_6$	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	$y_6 =$ *Yes*
$\mathbf{x}_7$	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	$y_7 =$ *No*
$\mathbf{x}_8$	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	$y_8 =$ *Yes*
$\mathbf{x}_9$	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	$y_9 =$ *No*
$\mathbf{x}_{10}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	$y_{10} =$ *No*
$\mathbf{x}_{11}$	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0–10*	$y_{11} =$ *No*
$\mathbf{x}_{12}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	$y_{12} =$ *Yes*

A decision tree for deciding whether to wait for a table.



However, this decision tree is not optimal (we will see later).

ลึกมาก

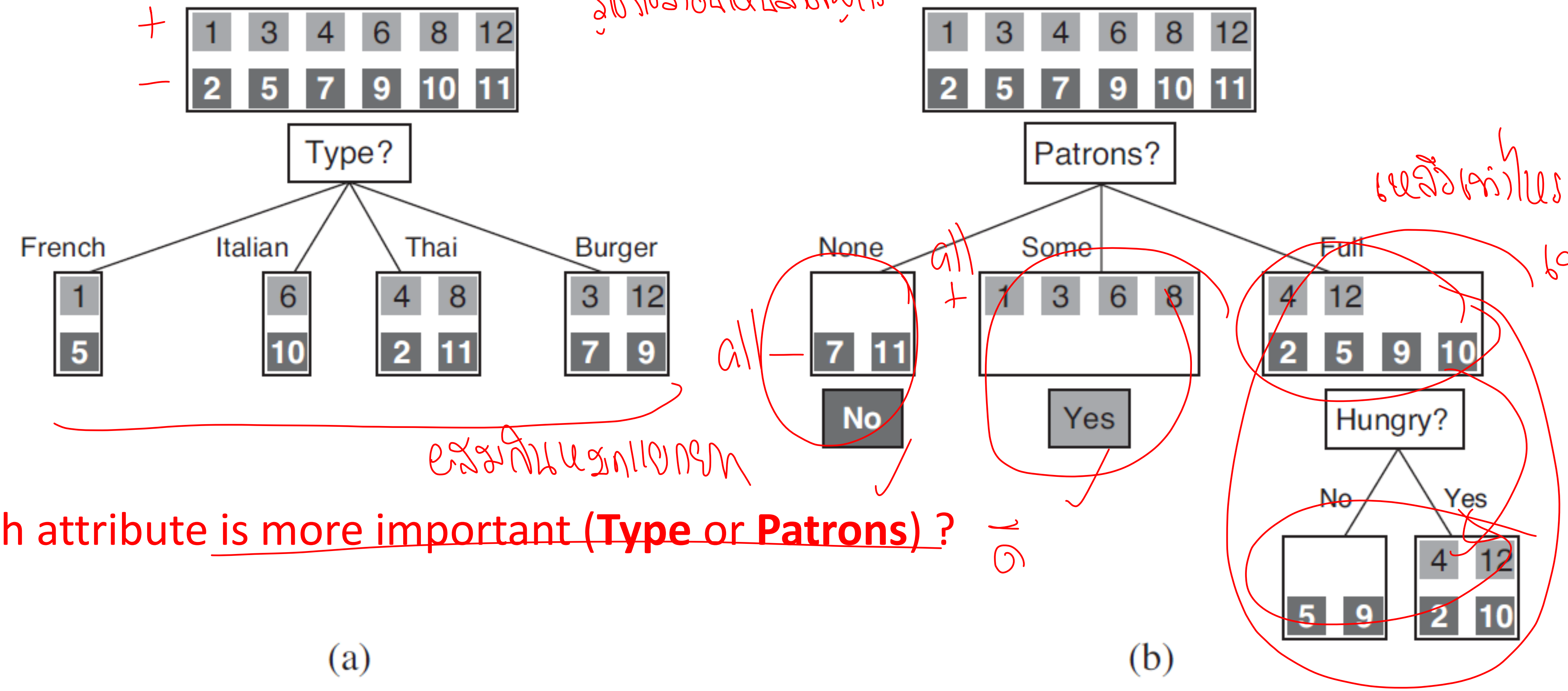
Creating the optimal decision tree

- We hope to get to the correct classification with a small number of tests, meaning that all paths in the tree will be short and the tree as a whole will be shallow. Hence, the optimal decision tree.

- The decision-tree-learning algorithm adopts a greedy divide-and-conquer strategy: always **test the most important attribute first**.

  node แรกๆ ควรมีความสำคัญมากสุด

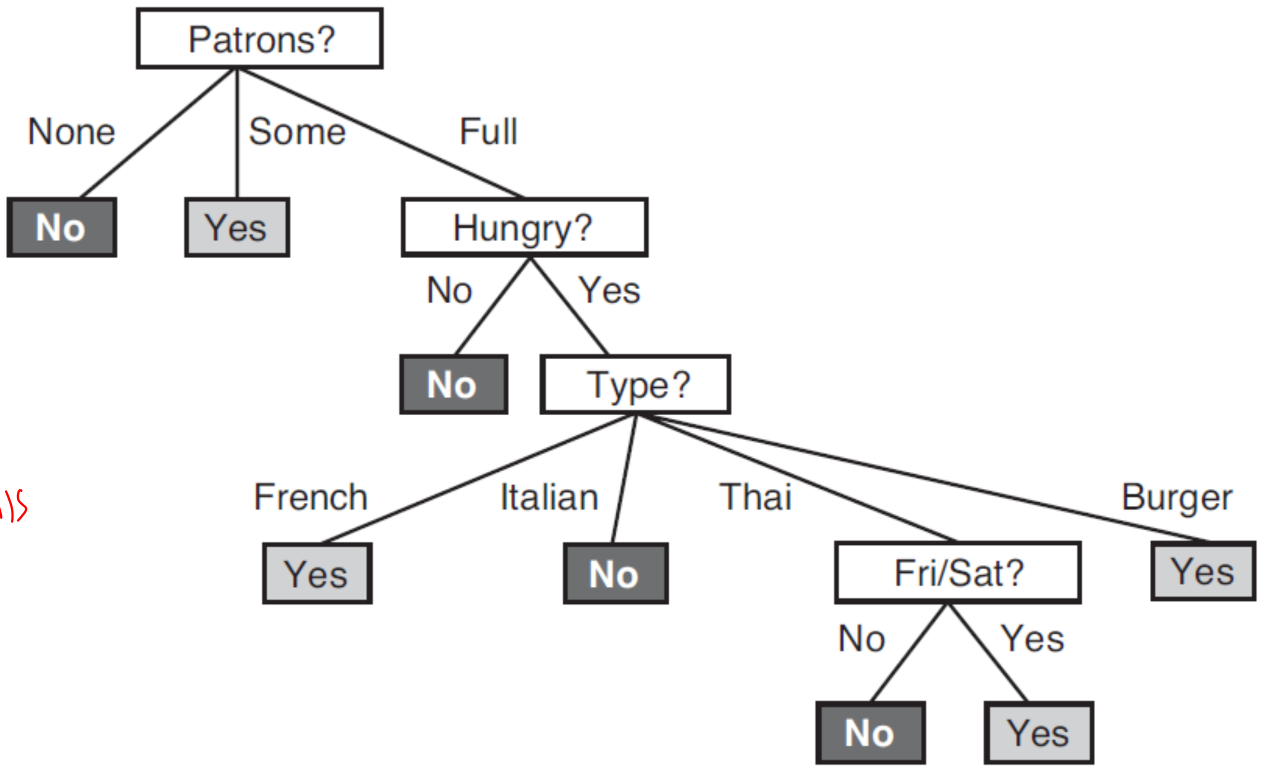
- By “most important attribute,” we mean the one that makes the most difference to the classification of an example.



Which attribute is more important (**Type** or **Patrons**) ?

Figure 18.4 Splitting the examples by testing on attributes. At each node we show the positive (light boxes) and negative (dark boxes) examples remaining. (a) Splitting on *Type* brings us no nearer to distinguishing between positive and negative examples. (b) Splitting on *Patrons* does a good job of separating positive and negative examples. After splitting on *Patrons*, *Hungry* is a fairly good second test.

The optimal decision tree induced from the 12-example training set.

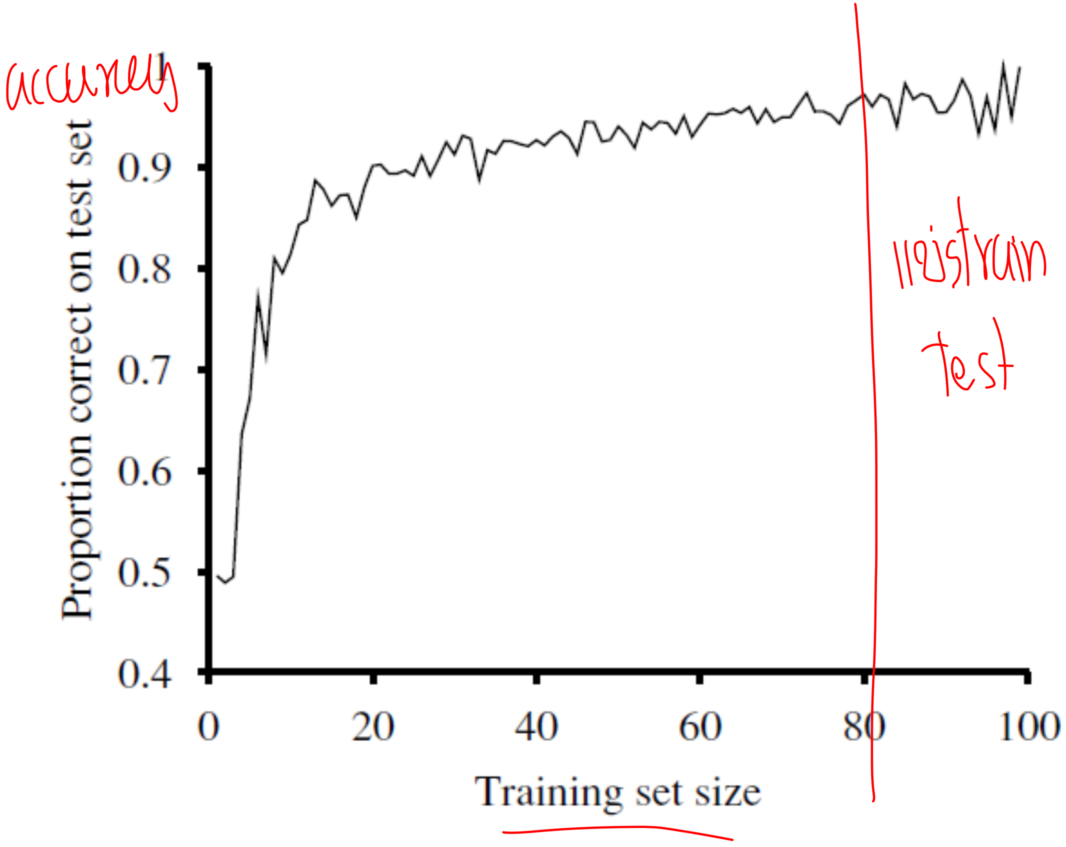


Depth =4

branch ก็อ่ไร

จะได้มาประมาณนี้

- We can evaluate the accuracy of a learning algorithm with a **learning curve**. We have 100 examples at our disposal, which we split into a training set and a test set.



- We learn a hypothesis h with the training set and measure its accuracy with the test set.
- We do this starting with a training set of size 1 and increasing one at a time up to size 99. For each size we actually repeat the process of randomly splitting 20 times, and average the results of the 20 trials.
- The curve shows that as the training set size grows, the accuracy increases.

In this graph, we reach 95% accuracy, and it looks like the curve might continue to increase with more data.

Information gain and Entropy

- A perfect attribute divides the examples into sets, each of which are all positive or all negative and thus will be leaves of the tree.
- The **Patrons** attribute is not perfect, but it is fairly good. → พอแยกได้
- A really useless attribute, such as **Type**, leaves the example sets with roughly the same proportion of positive and negative examples as the original set. → loss
- We will use the notion of **information gain**, which is defined in terms of **entropy** (Shannon and Weaver, 1949), to define a formal measure of “fairly good” and “really useless” → ได้ข้อมูลเยอะ
- **Entropy** is a measure of the uncertainty of a random variable; acquisition of information corresponds to a reduction in entropy.

(0~1) วัดความไม่แน่นอน → ยิ่งไม่แน่นอน → loss ยิ่งมาก

- A random variable with only **one value**—a coin that always comes up heads—has no uncertainty and thus its **entropy is defined as zero**; thus, we gain no information by observing its value.

เกิดแต่ x มาตลอด ส่วนมากบอกว่าจะเกิด y → entropy จะต่ำ

- If a training set contains **p positive** examples and **n negative** examples, then the entropy of the whole set is

informationที่ได้

$$\textit{Entropy}: I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) = -\frac{p}{p+n}\log_2\left(\frac{p}{p+n}\right) - \frac{n}{p+n}\log_2\left(\frac{n}{p+n}\right)$$

- We can check that the entropy of a fair coin flip is indeed 1:

$$I\left(\frac{1}{1+1}, \frac{1}{1+1}\right) = -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2} = 1.$$

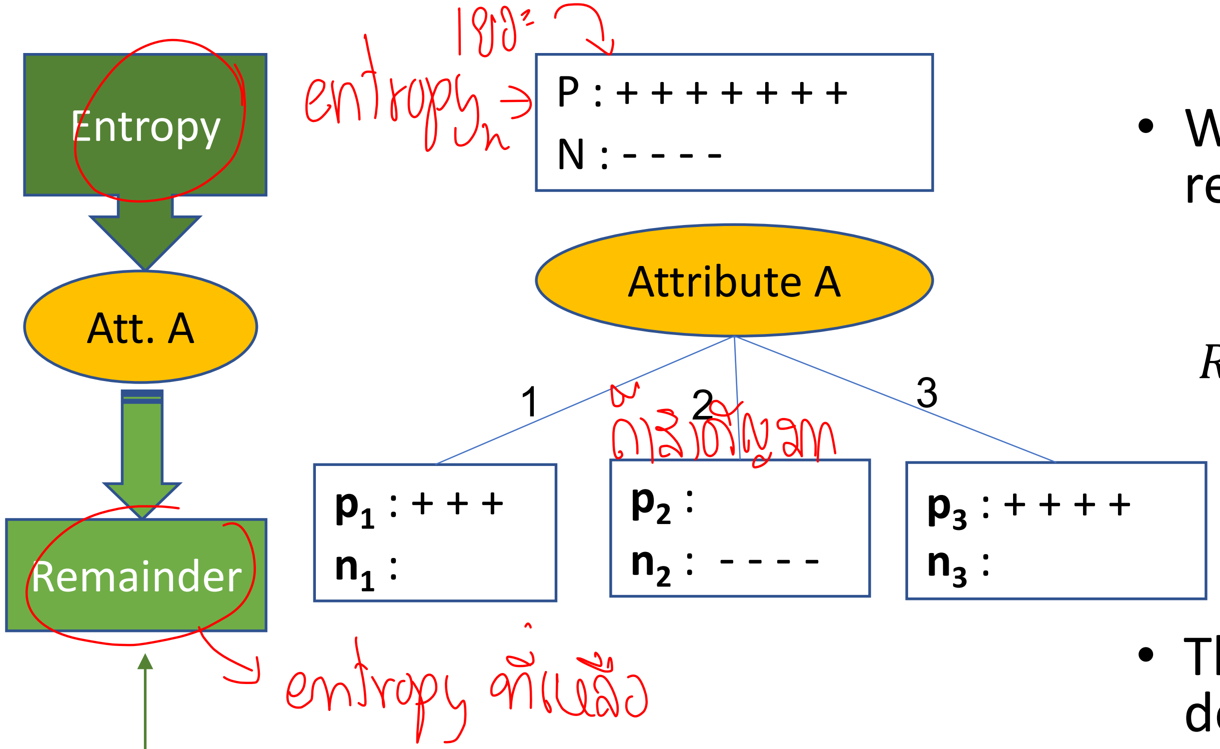
- A flip of a fair coin is equally likely to come up heads or tails, 0 or 1, and this counts as "1 bit" of entropy.

กรณีไม่แน่นอน

เชื่อถือ

- If the coin is loaded to give 99% heads, we get

$$I\left(\frac{99}{99+1}, \frac{1}{99+1}\right) = -\frac{99}{100}\log_2\frac{99}{100} - \frac{1}{100}\log_2\frac{1}{100} = 0.08.$$



Tell us how much information we still need to classify the remaining examples.

จะต้องน้อยๆ ✓

good

- We can measure how much the entropy remaining after testing the attribute A by

branch ที่แตกไปได้

$$Remainder(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p + n} . I\left(\frac{p_i}{p_i + n_i}, \frac{n_i}{p_i + n_i}\right)$$

ก่อนclassify

p_i, n_i หลังclassify

- The **information gain** from the attribute A is defined as

entropy ที่เหลือ

Attribute

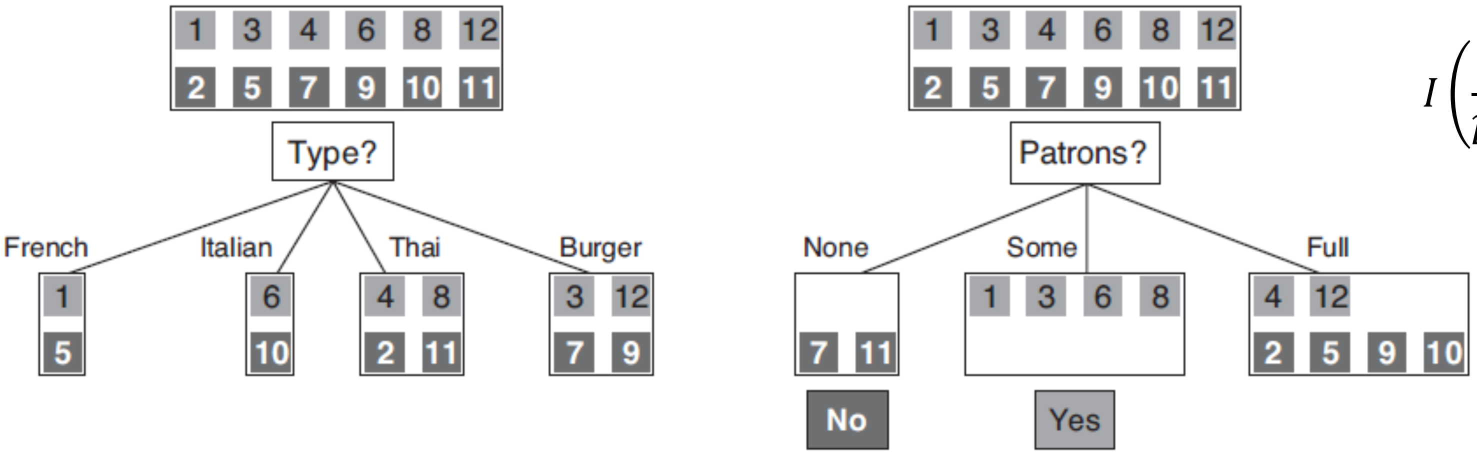
$$Gain(A) = I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) - Remainder(A)$$

(max) (min)

- **The more value of the information gain of the attribute A, the more important it is.**

ก็คือ entropy$_{n+1}$ นั่นเอง

- Now, let us calculate the information of both Patrons and Type



$$I\left(\frac{p}{p+n},\frac{n}{p+n}\right)=-\frac{p}{p+n}\log_2\frac{p}{p+n}-\frac{n}{p+n}\log_2\frac{n}{p+n}$$

$$Remainder(A)=\sum_{i=1}^{v}\frac{p_i+n_i}{p+n}.I\left(\frac{p_i}{p_i+n_i},\frac{n_i}{p_i+n_i}\right)$$

$$Gain(A)=I\left(\frac{p}{p+n},\frac{n}{p+n}\right)-Remainder(A)$$

$$Gain(Type)=I\left(\frac{6}{6+6},\frac{6}{6+6}\right)-\left[\frac{2}{12}.I\left(\frac{1}{2},\frac{1}{2}\right)+\frac{2}{12}.I\left(\frac{1}{2},\frac{1}{2}\right)+\frac{4}{12}.I\left(\frac{2}{4},\frac{2}{4}\right)+\frac{4}{12}.I\left(\frac{2}{4},\frac{2}{4}\right)\right]$$
$$=0$$

function useless

$$Gain(Patrons)=I\left(\frac{6}{6+6},\frac{6}{6+6}\right)-\left[\frac{2}{12}.I(0,1)+\frac{4}{12}.I(1,0)+\frac{6}{12}.I\left(\frac{2}{6},\frac{4}{6}\right)\right]$$
$$=1-[0+0+0.4594]$$
$$=0.5406$$

Therefore, Patrons is more important than Type.

ไม่ต้อง train แต่ใช้การวัดระยะ

K Nearest Neighbor Algorithm (KNN)

ดูว่าใกล้ใครกี่ตัว → k ควรเป็นเลขคี่

- Besides ID3 algorithm, there is another simple classification which does not require any training parameter.
- **K nearest neighbors** is a simple algorithm that stores all available examples and classifies new examples based on a similarity measure (e.g., distance functions).
- KNN has been used in statistical estimation and pattern recognition already in the beginning of 1970’s as a **non-parametric** technique.

Reference: http://www.saedsayad.com/k_nearest_neighbors.htm

KNN Algorithm

- An example is classified by a majority vote of its neighbors.
- An unseen example is assigned to the class most common among its K nearest neighbors measured by a distance function.
- If K = 1, then an unseen example is simply assigned to the class of the nearest neighbor, only one neighbor which is nearest to an unseen example.

Dimensions

Distance functions

Euclidean $\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$

Manhattan $\sum_{i=1}^{k}|x_i - y_i|$

Minkowski $\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{1/q}$

- It should also be noted that all three distance measures are only valid for continuous variables. In the instance of categorical variables the Hamming distance must be used.
- Historically, the optimal K for most datasets has been between 3-10. That produces much better results than 1NN.

ອັກຮະ

Hamming Distance

$$x = y \Rightarrow D = 0$$

$$x \neq y \Rightarrow D = 1$$

X	Y	Distance
Male	Male	0
Male	Female	1

- **Example**: Consider the following data concerning **credit default**. **Age** and **Loan** are two numerical attributes and **Default** is the target.

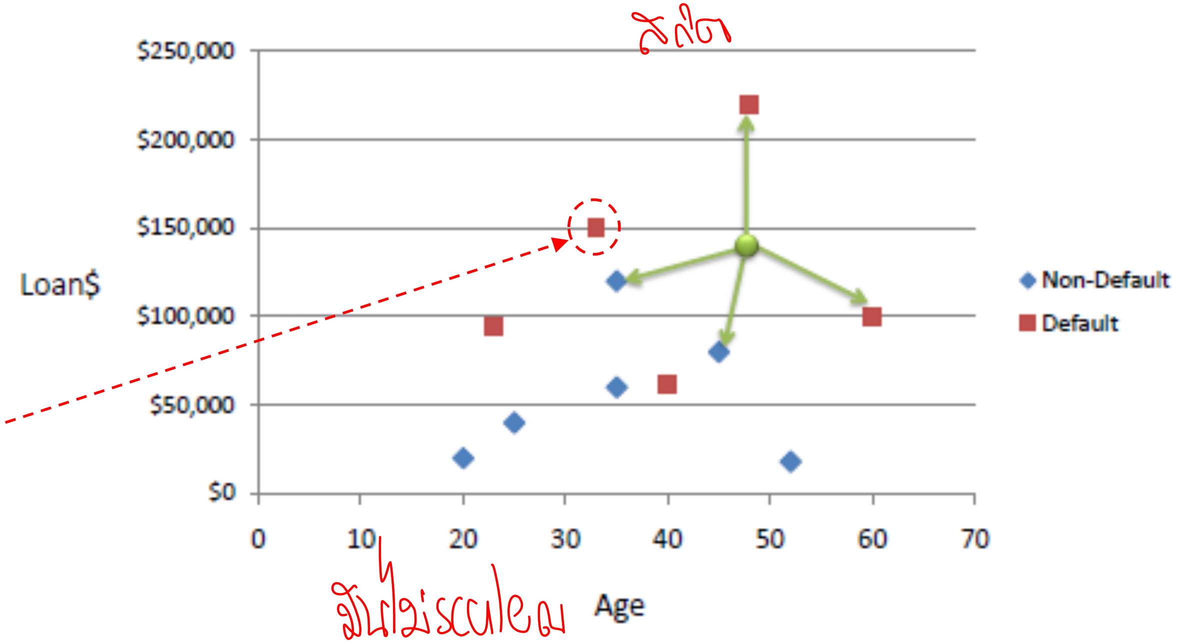
- We can now use the training set to classify an unknown case (Age=48 and Loan=$142,000) using Euclidean distance. If K=1 then the nearest neighbor is the last case in the training set with Default=Y.

D = Sqrt[(48-33)^2 + (142,000-150,000)^2] = 8,000.01 >> Default=Y

Age	Loan	Default	Distance	
25	$40,000	N	102000	
35	$60,000	N	82000	
45	$80,000	N	62000	
20	$20,000	N	122000	
35	$120,000	N	22000	2
52	$18,000	N	124000	
23	$95,000	Y	47000	
40	$62,000	Y	80000	
60	$100,000	Y	42000	3
48	$220,000	Y	78000	
33	$150,000	Y	8000	1
48	**$142,000**	?		

Euclidean Distance

$$D = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$$



ยังไม่scaleเลย

- With K=3, there are two Default=Y and one Default=N out of three closest neighbors. The prediction for the unknown case is again Default=Y.

Standardized Distance

- One major drawback in calculating distance measures directly from the training set is in the case where variables have different measurement scales or there is a mixture of numerical and categorical variables.
- For example, if one variable is based on annual income in dollars, and the other is based on age in years then income will have a much higher influence on the distance calculated.
- One solution is to standardize the training set.
- Using the standardized distance on the same training set, the unknown example returned a different neighbor.

Age	Loan	Default	Distance
0.125	0.11	N	0.7652
0.375	0.21	N	0.5200
0.625	0.31	N	0.3160
0	0.01	N	0.9245
0.375	0.50	N	0.3428
0.8	0.00	N	0.6220
0.075	0.38	Y	0.6669
0.5	0.22	Y	0.4437
1	0.41	Y	0.3650
0.7	1.00	Y	0.3861
0.325	0.65	Y	0.3771
0.7	**0.61**	?	

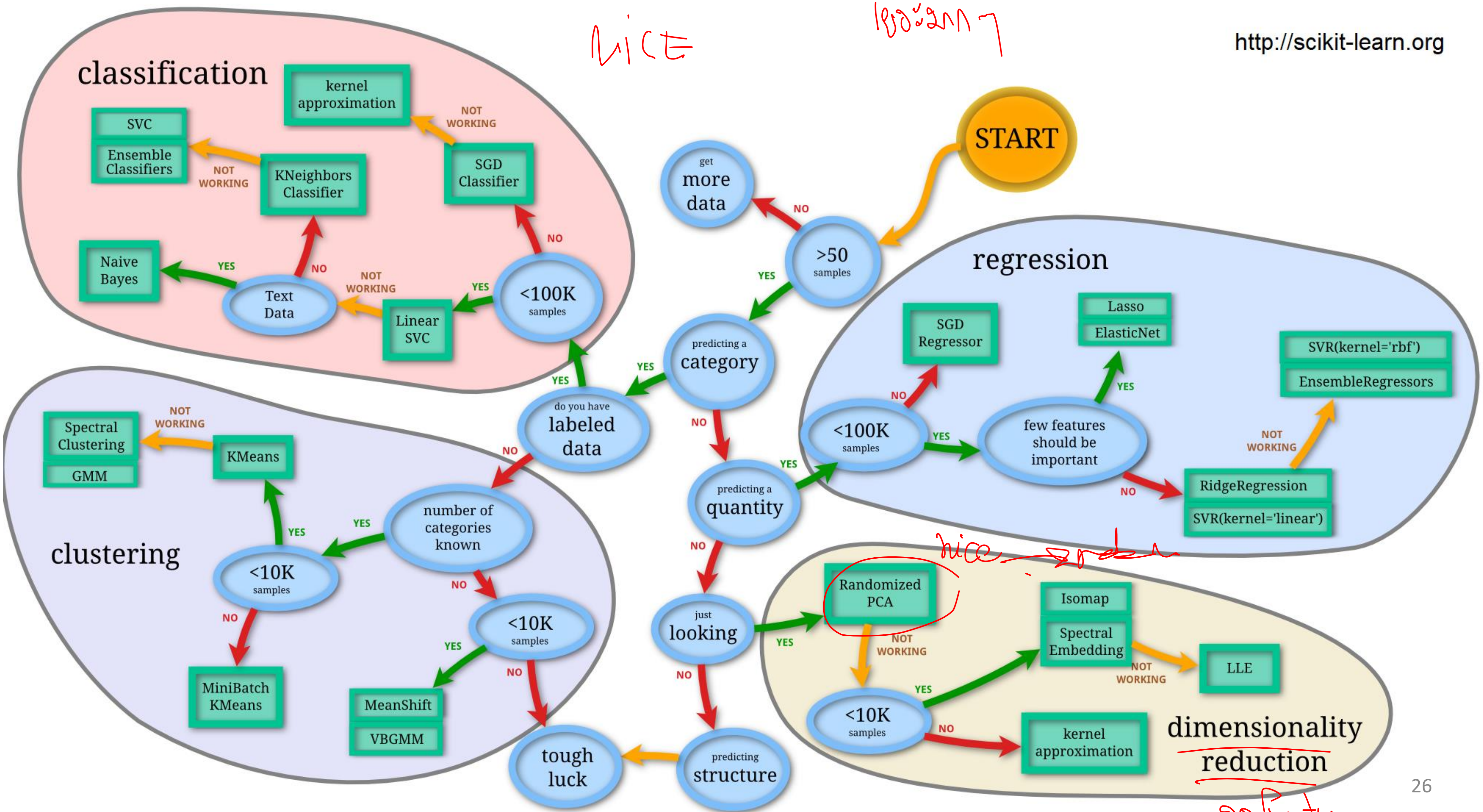
Standardized Variable

$$X_s = \frac{X - Min}{Max - Min}$$

rescale

ไม่เกิน 1.0

• Other machine learning algorithms



Evaluating the hypothesis

- We can simply use the error rate to check the quality of the hypothesis.

100% - accuracy

- $\text{Error rate} = \dfrac{\text{Number of examples which are wrong predicted}}{\text{Total number of examples}}$

- However, a hypothesis h has a low error rate on the training set does not mean that it will generalize well.

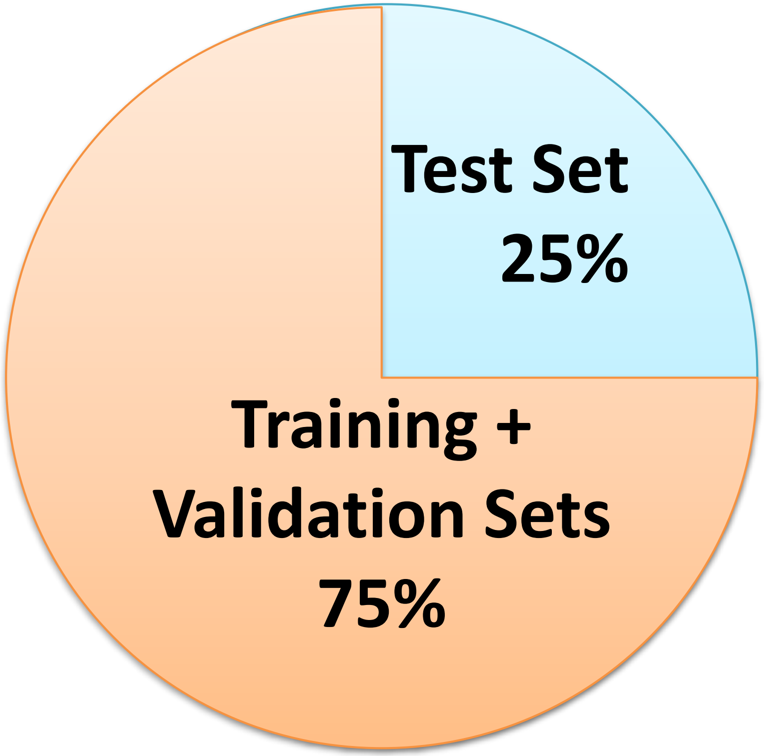
- To get the more accurate evaluation, we need to test the hypothesis on an unseen set (test set) of examples.

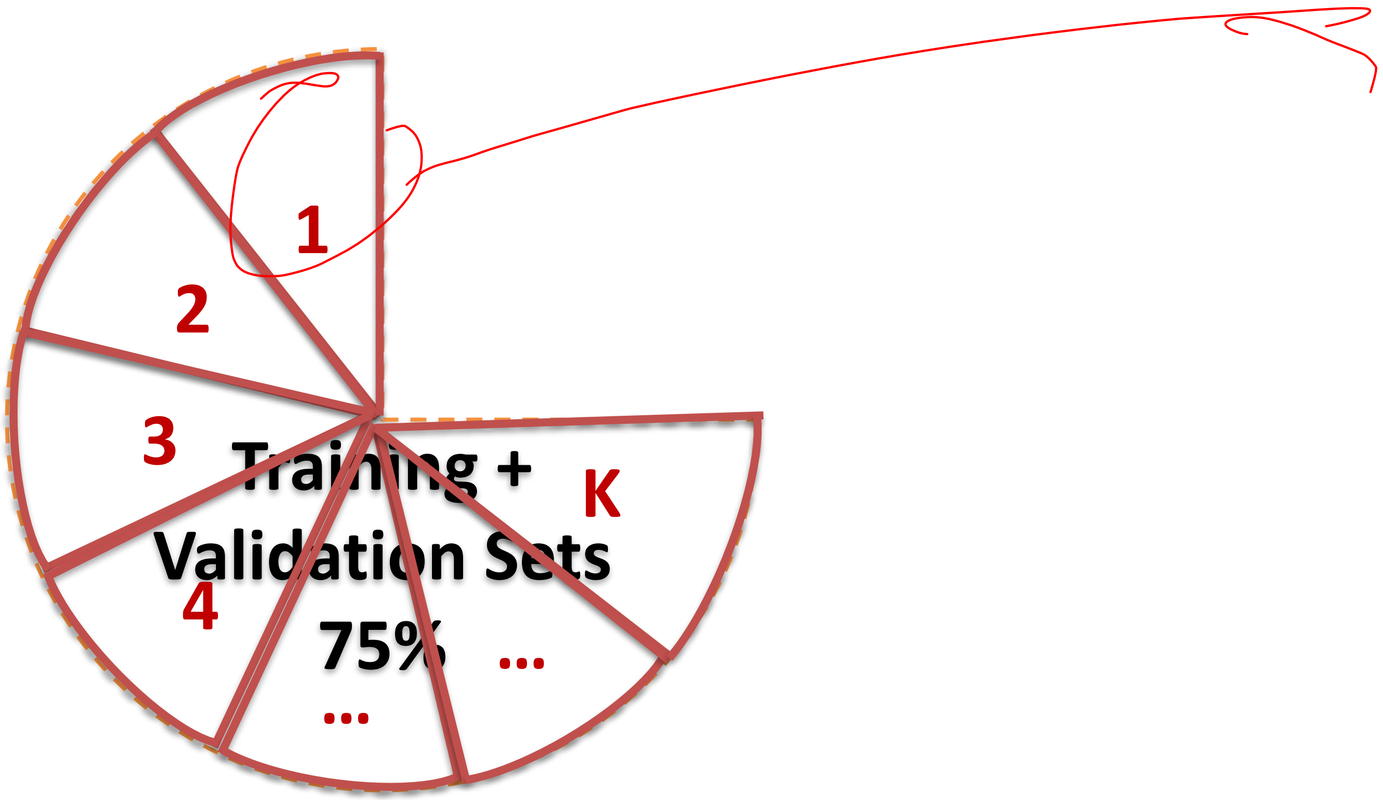
- **Holdout cross-validation** :

Randomly split the available data into a training set (e.g. 75%) and a test set (e.g. 25%).

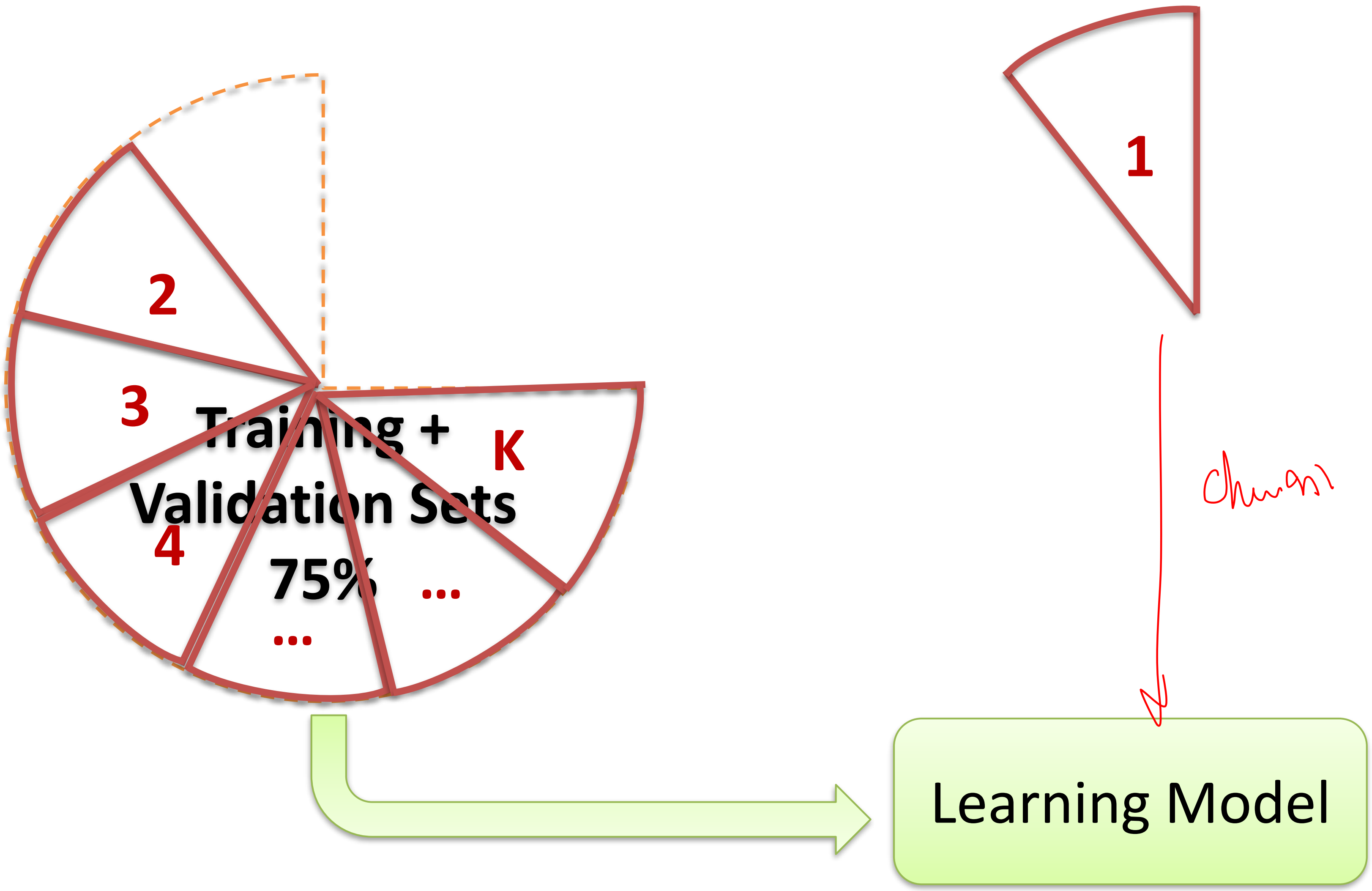
- **K-fold cross-validation**:
  - Split the training data into k equal subsets.
  - Then perform k rounds of learning. On each round, 1/k of the data is held out as a validation set and the remaining examples are used as training data.
  - The average error rate of the k rounds is finally calculated. Popular values for k are 5 and 10. The extreme is k=n (the total number of examples), also known as leave-one-out cross-validation or LOOCV.

K-Fold Cross Validation

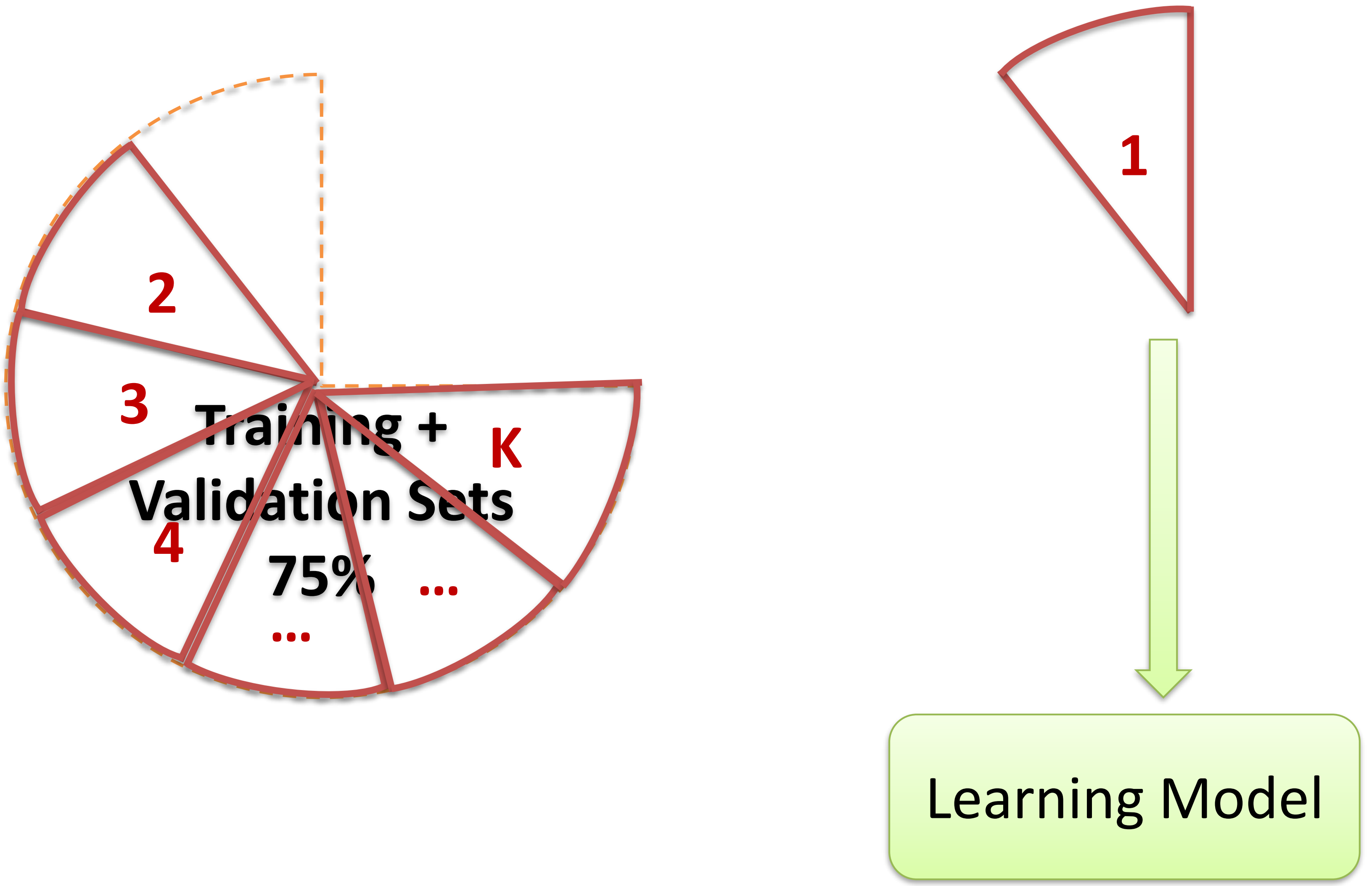




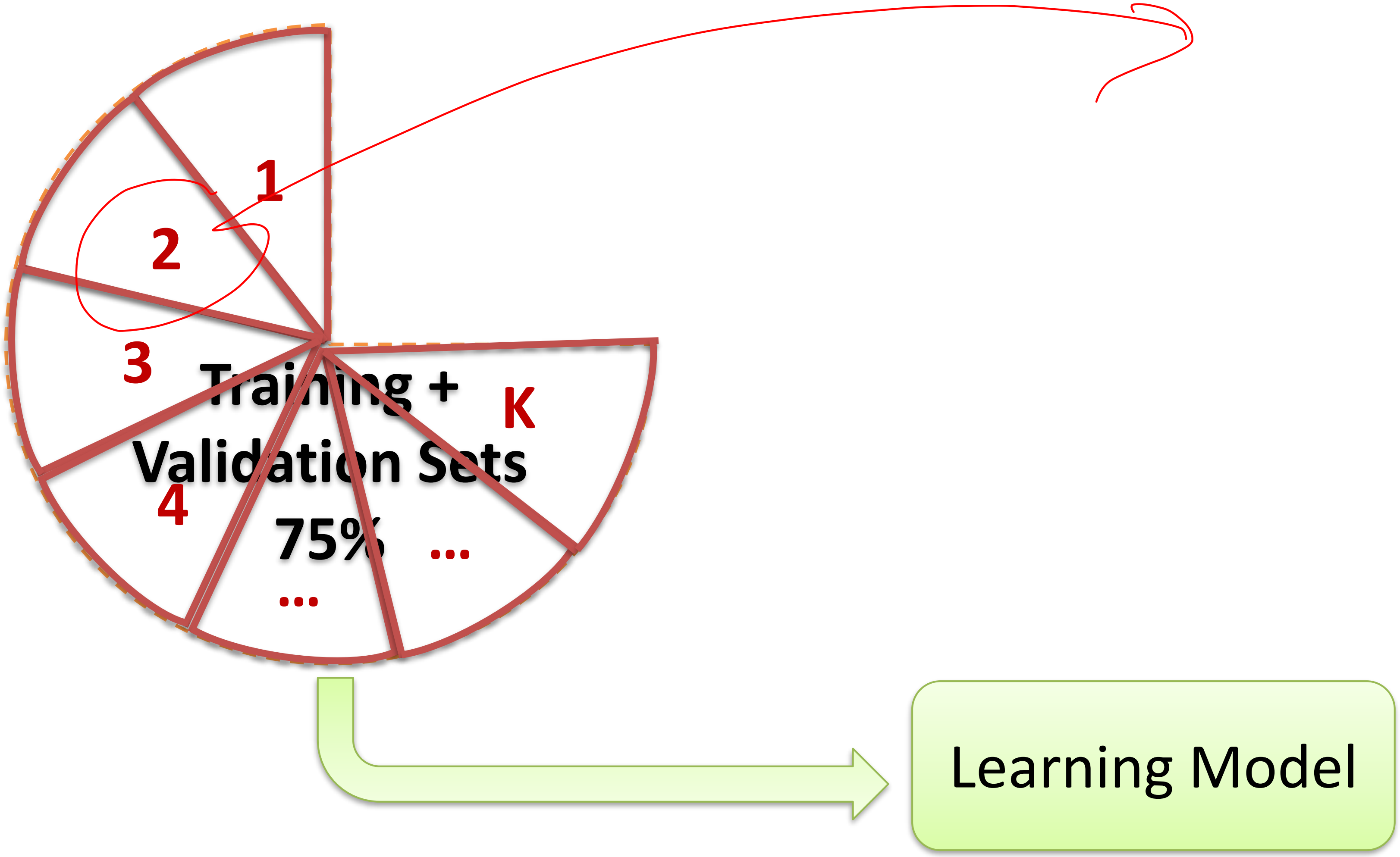
Learning Model



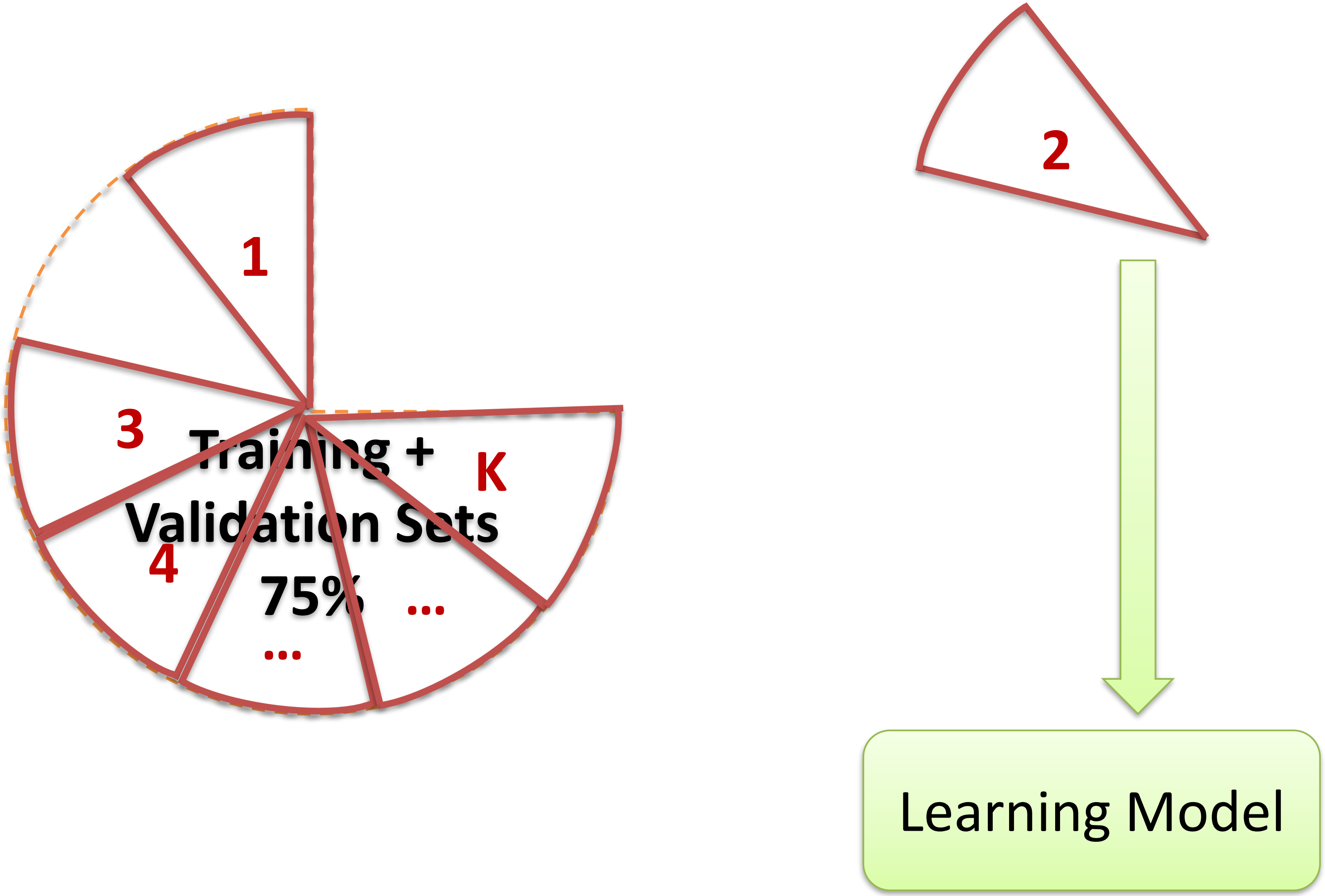
Training1 Accuracy/Error



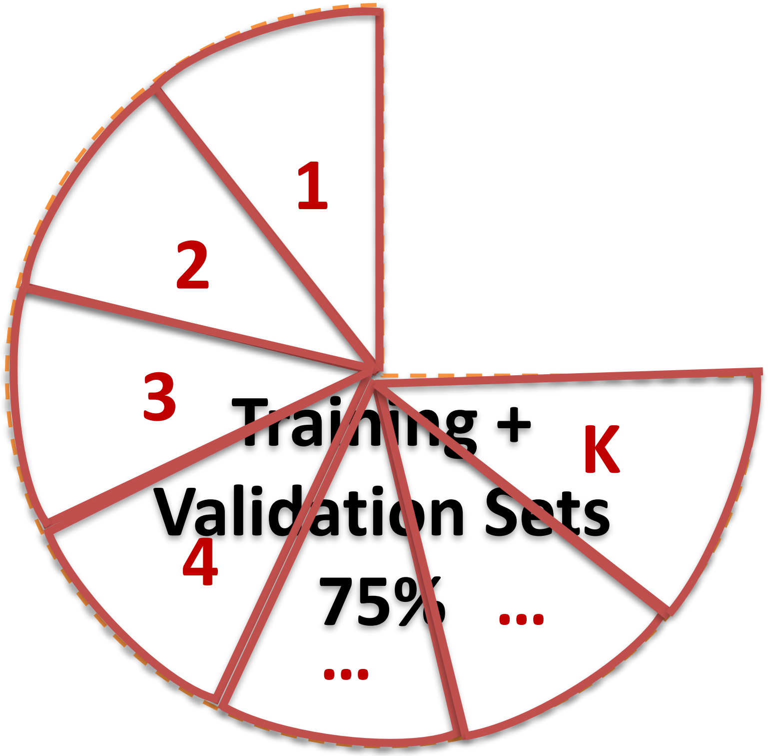
Validation1 Accuracy/Error



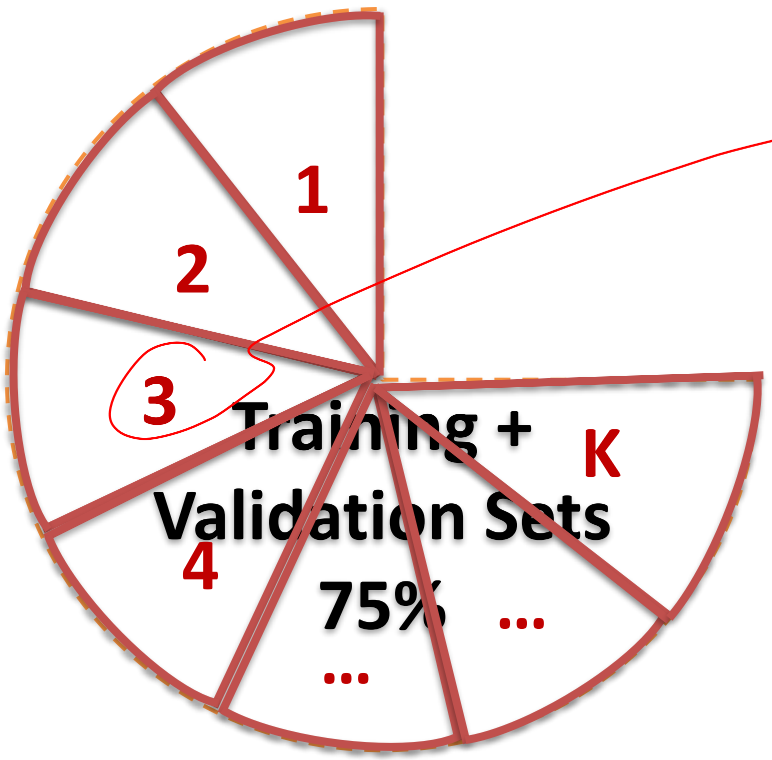
Training2 Accuracy/Error



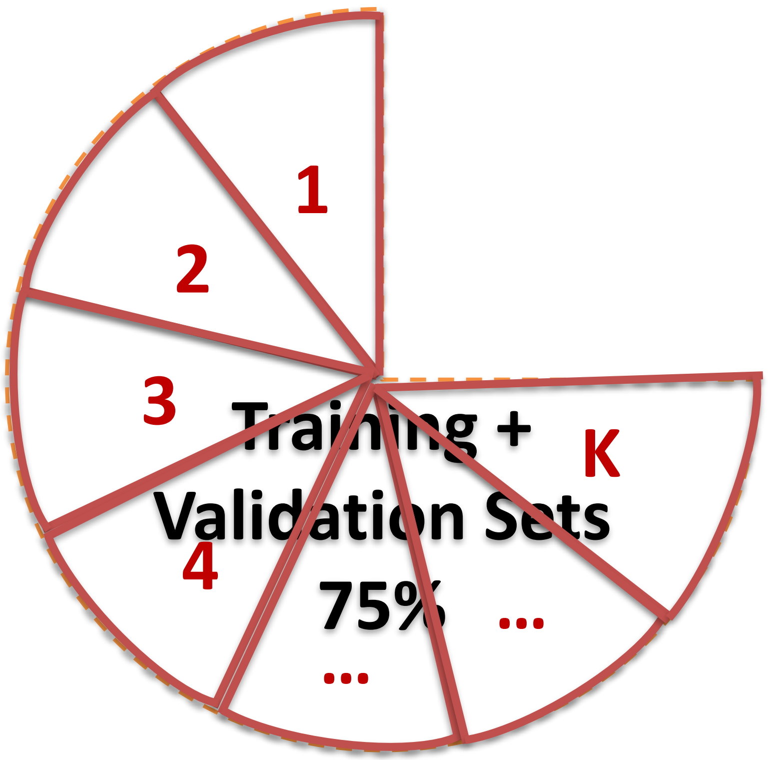
Validation2 Accuracy/Error



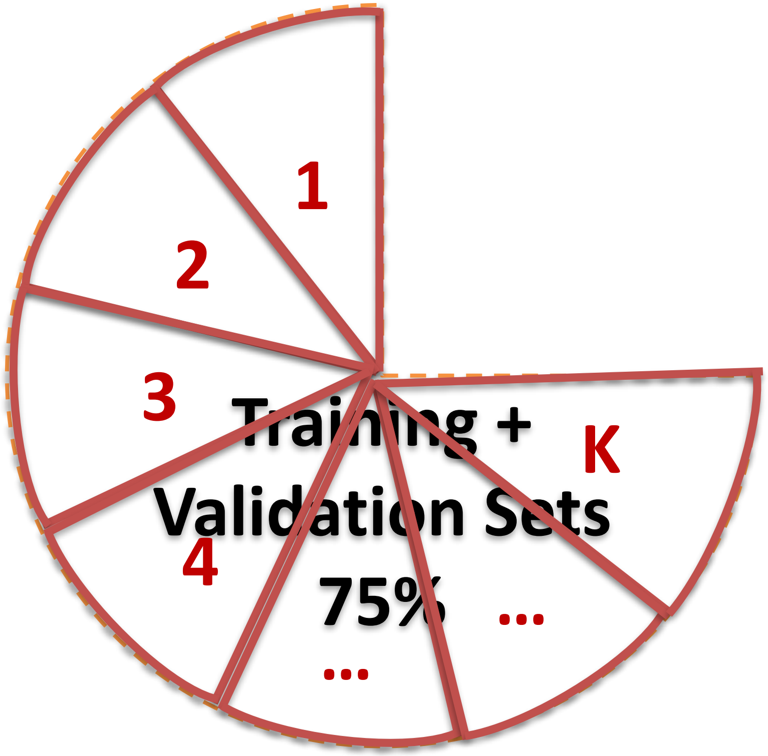
Learning Model



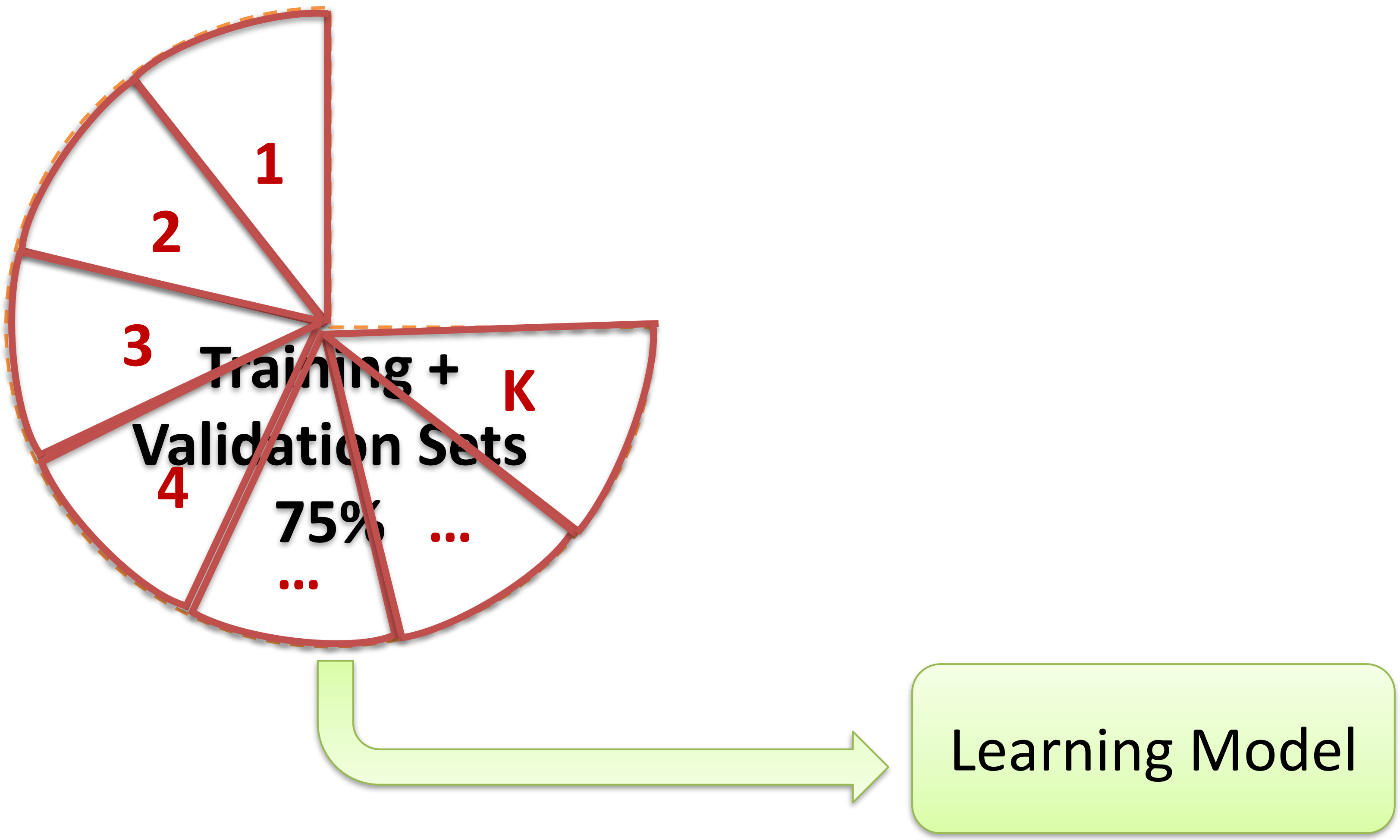
Learning Model



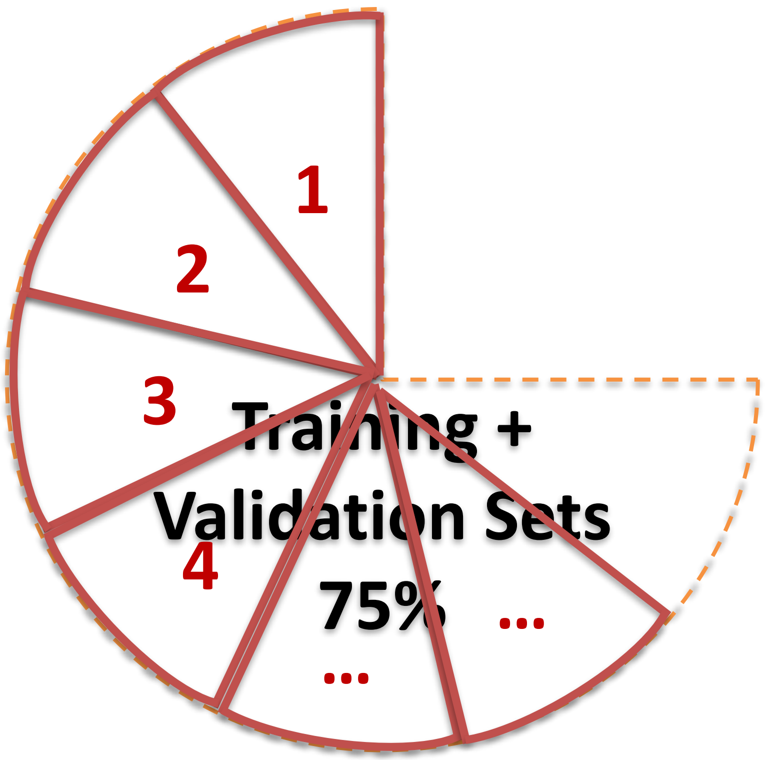
Learning Model

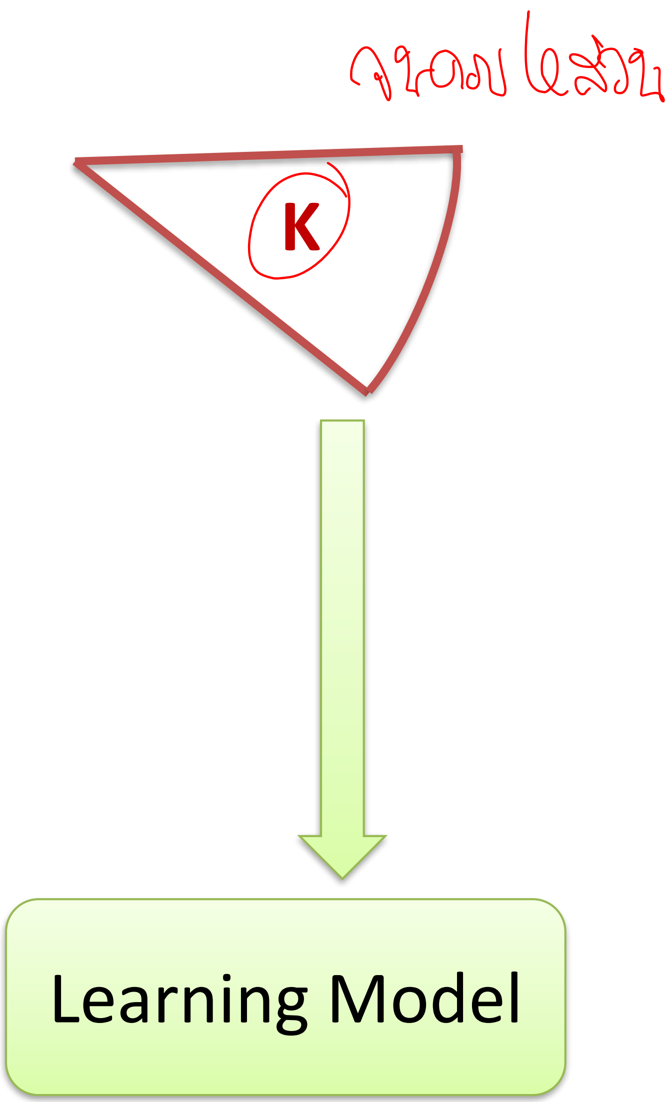


Learning Model



Training K Accuracy/Error

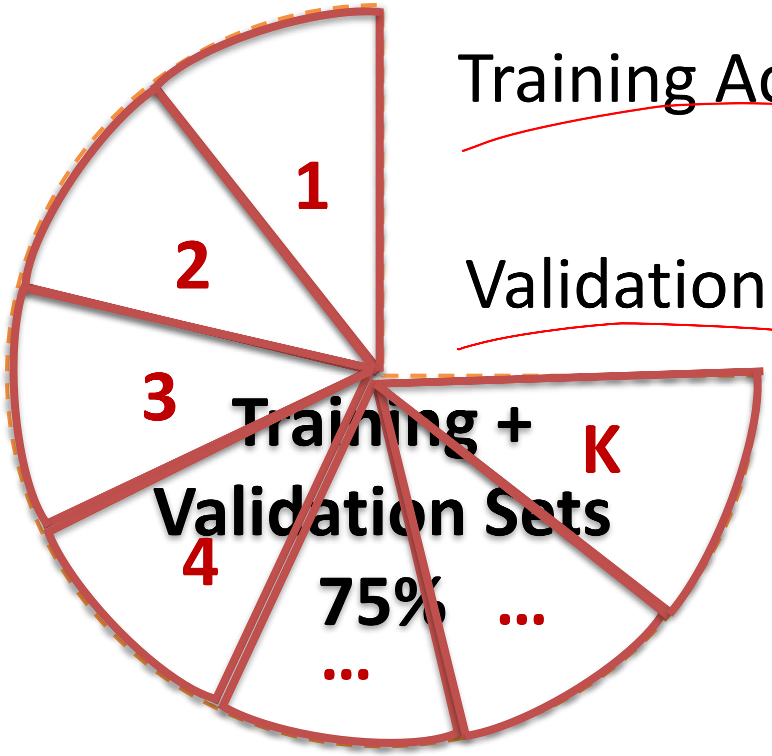


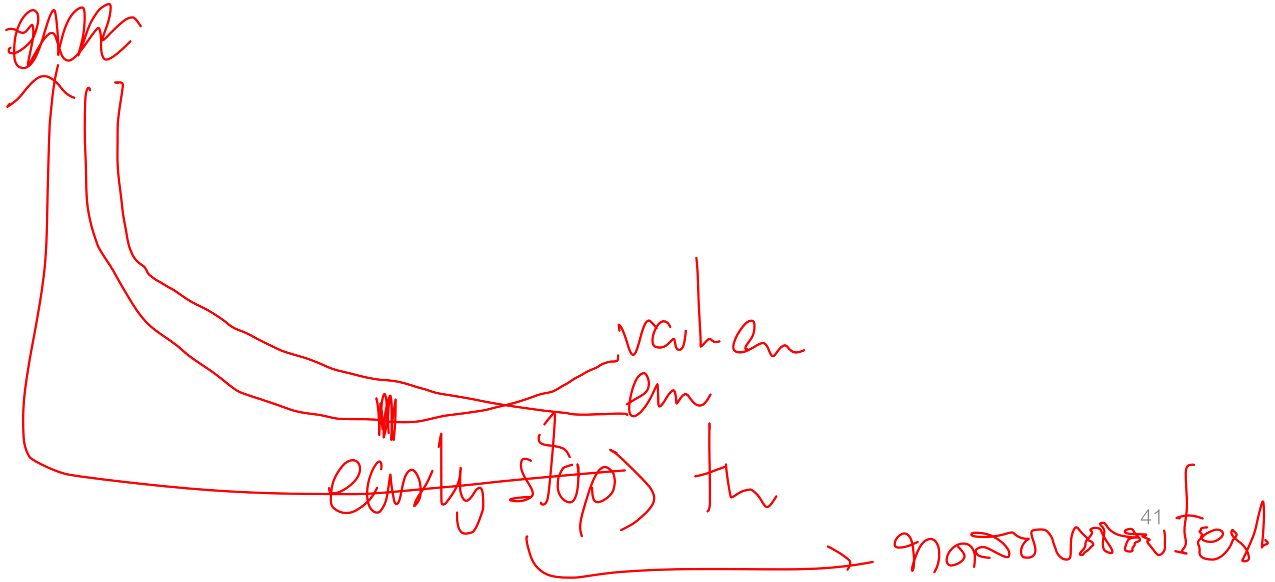


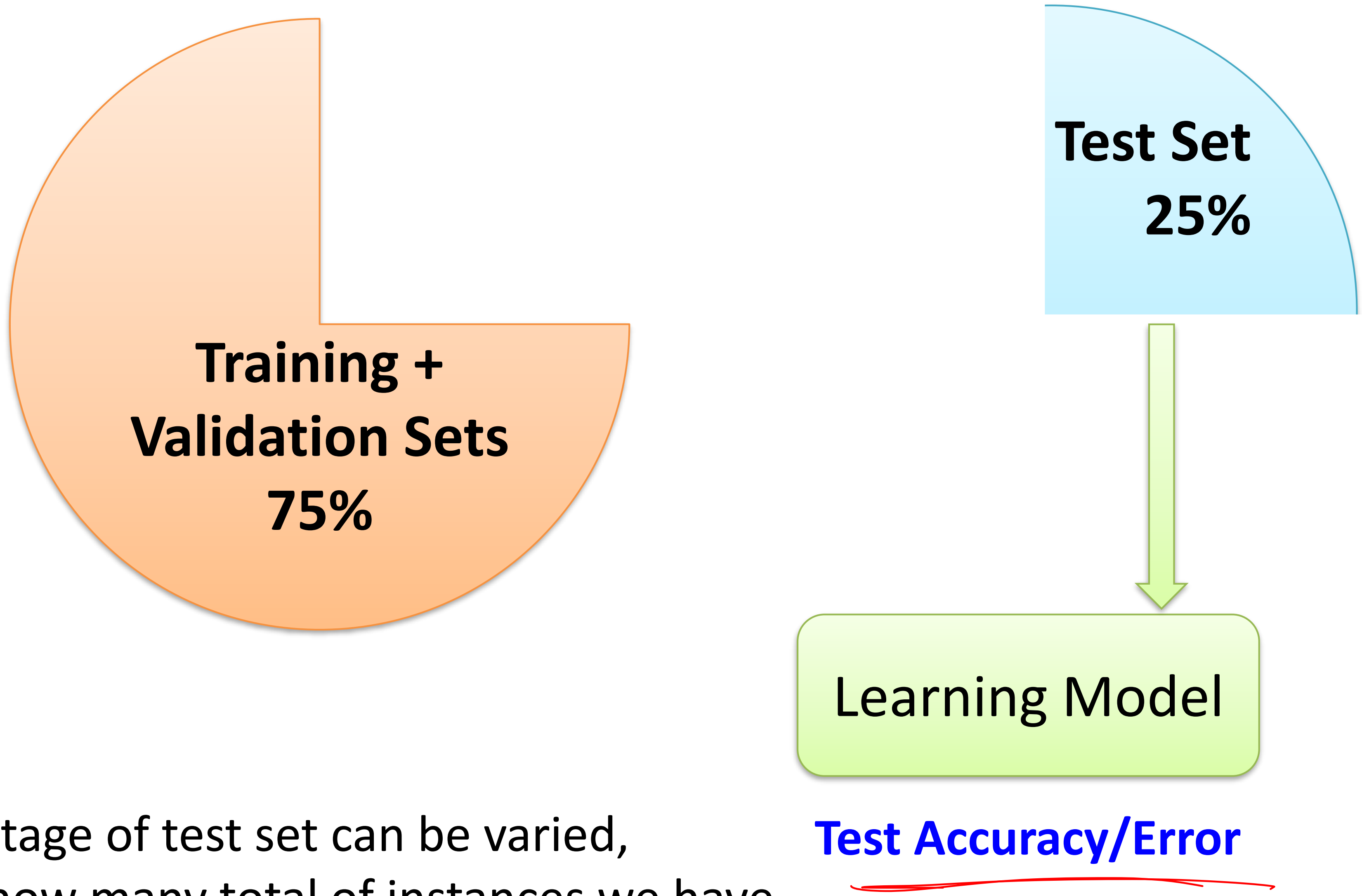
Validation K Accuracy/Error

Training Accuracy = $(\sum_{i=1}^{K} Train\_i\_Acc)/\mathrm{K}$

Validation Accuracy = $(\sum_{i=1}^{K} Valid\_i\_Acc)/\mathrm{K}$







Note: Percentage of test set can be varied, depends on how many total of instances we have.