Introduction to Machine Learning (Part 1)

Speaker: Justin Wu

CloudMile

"Before we get started, does anyone want to get out?"

--- Steve Rogers, *Captain America 2*.

Prologue

- People are talking about AI and machine learning every day.

- Because they can be used to make money.

- We will teach you about the techniques in machine learning, as well as how to use it in real-world scenario.

What is Machine Learning?

- According to Wikipedia, machine learning is a field of computer science that uses statistical techniques to give computer systems the ability to "learn" with data, without being explicitly programmed.

- To put in a more human-friendly description, it is the process of making a machine learn pattern from data to perform a certain task.

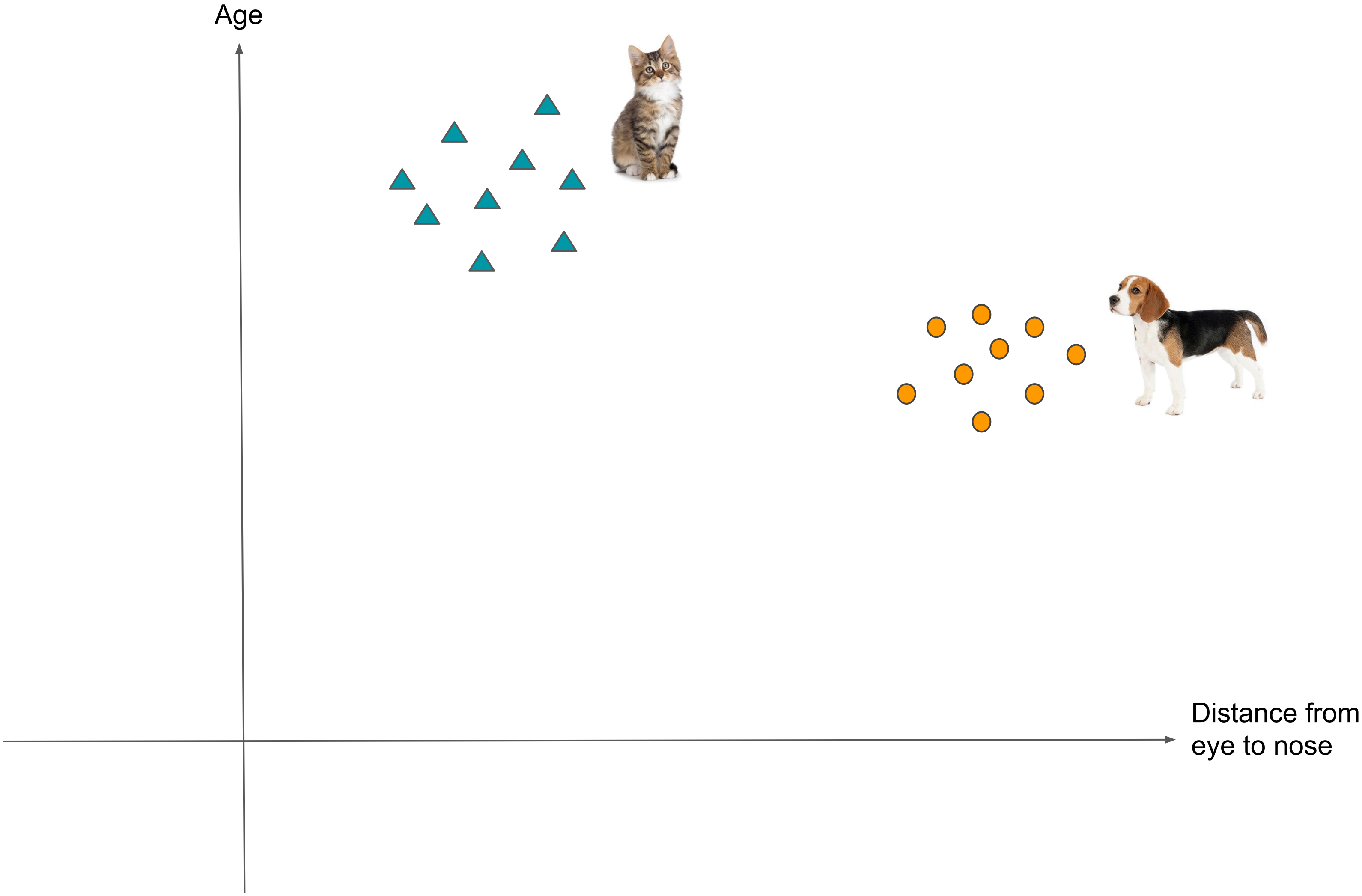
Let us see some examples

Assuming we have some data...

- Say that we have some sample data regarding animal, these sample data might look like this:

Data Index	Distance from eyes to nose	Age	Category
1	2.2	12	cat
2	6.1	10	dog
3	2.3	13	cat
4	1.9	14	cat
...	...	...	...

We might get a chart like this:



蒸蚌。

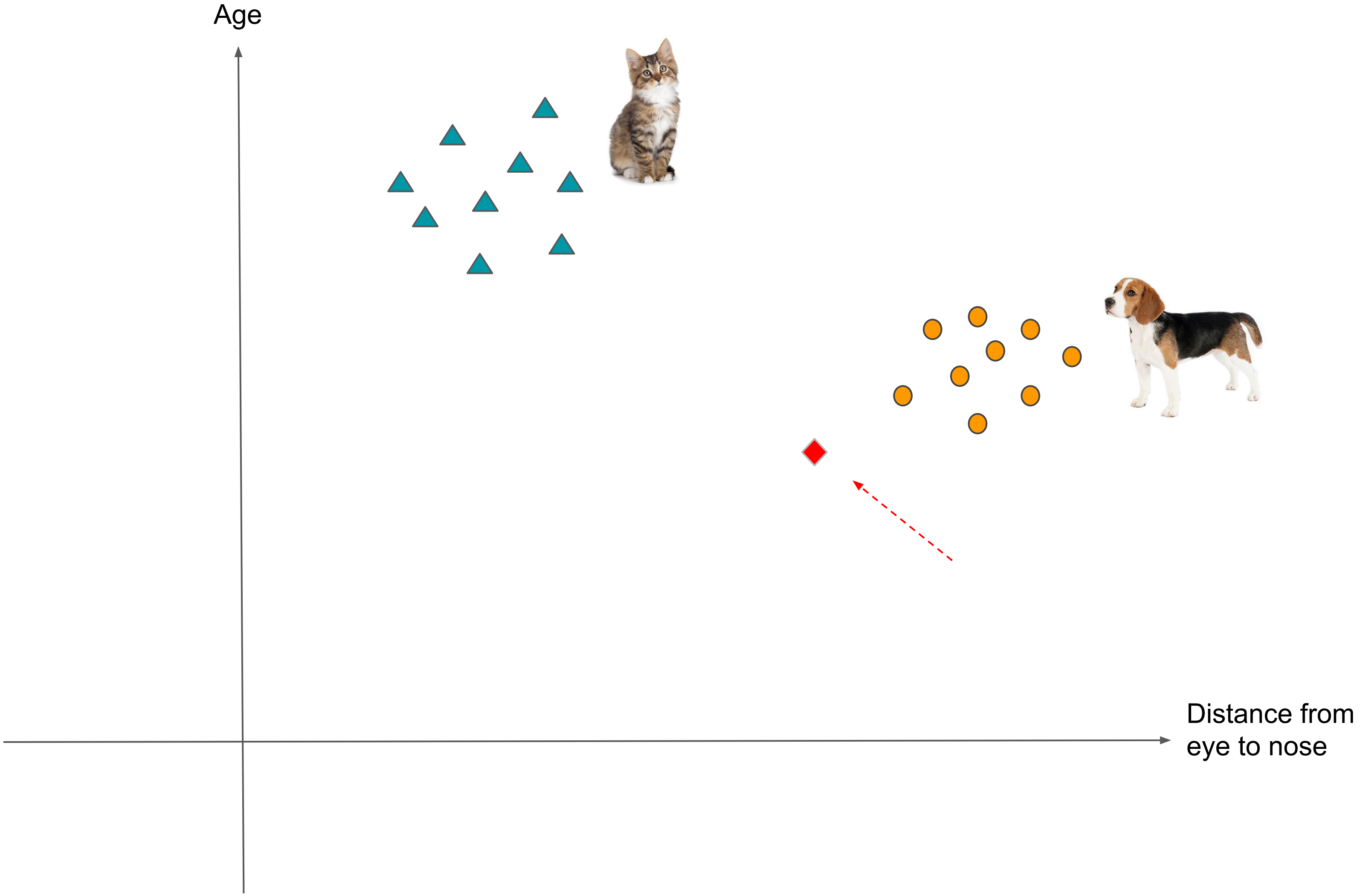
Consider this scenario...

- Assuming one day you see a row in the table that has the two numbers, but no category:

Data Index	Distance from eyes to nose	Age	Category
1	2.2	12	cat
2	6.1	10	dog
3	2.3	13	cat
4	1.9	14	cat
...	...	...	...

1009	4.9	9	?

We want to know the category for this data point



The most popular ML task: Classification

- This challenge is the most popular task in machine learning: classification.

- For such a task, we need to have a lot of data with known categories to serve as the *reference* when we want to classify new data. These known categories are officially called labels of the data.

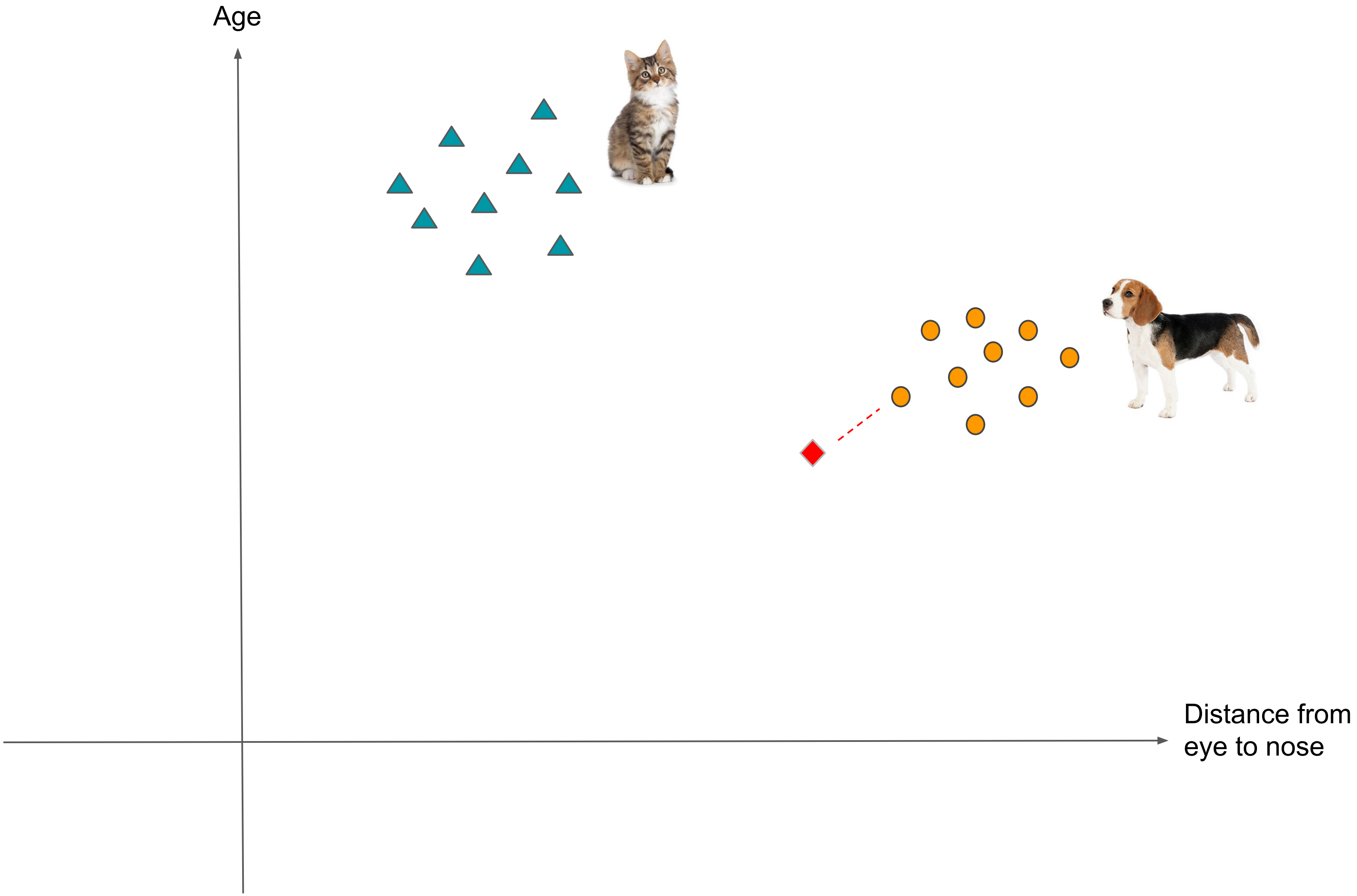
- This type of machine learning task requires data with "answer" beforehand. We call this type of task "supervised learning".

Features
(or Attributes)

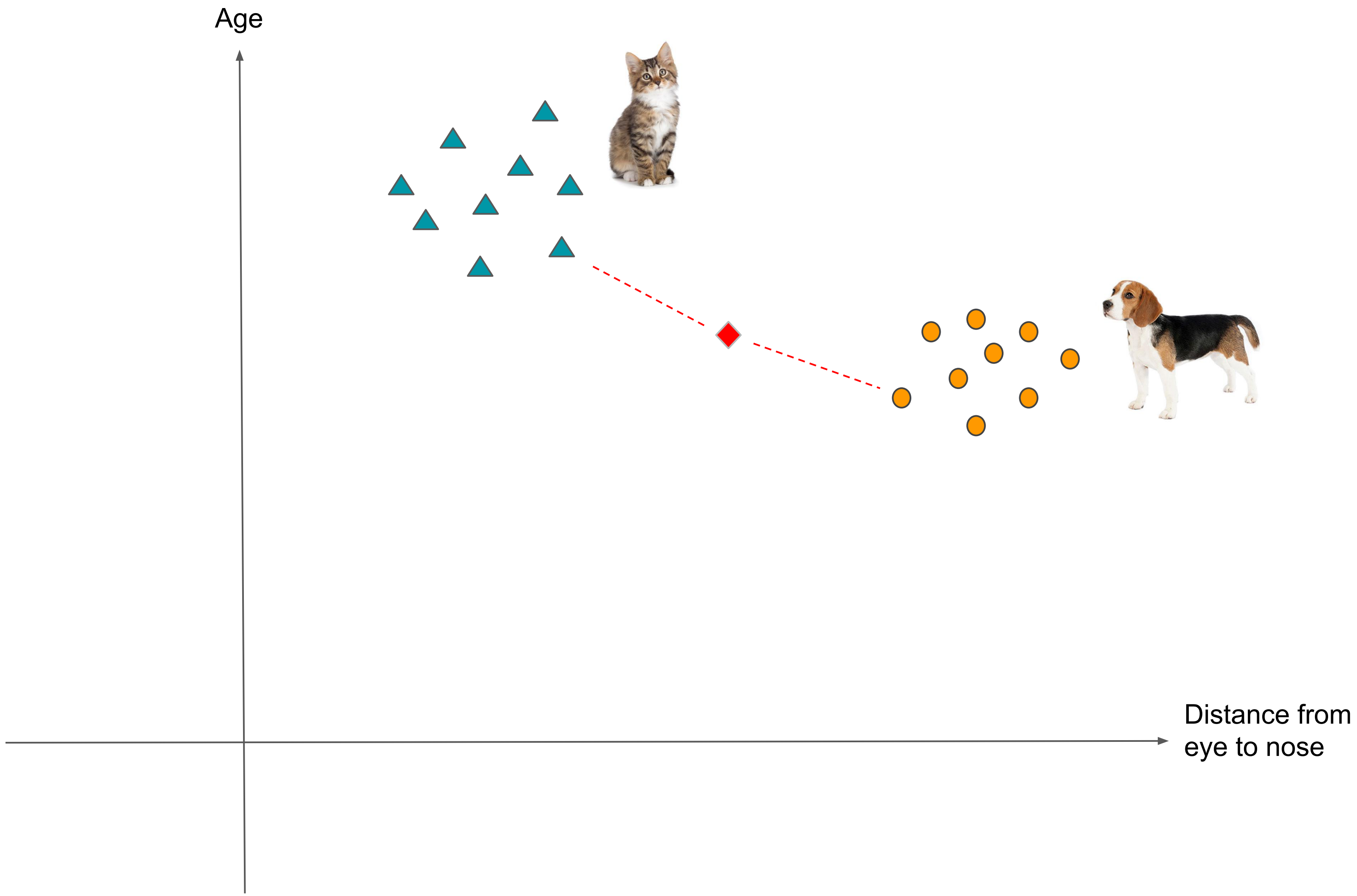
Label

Data Index	Distance from eyes to nose	Age	Category
1	2.2	12	cat
2	6.1	10	dog
3	2.3	13	cat
4	1.9	14	cat
...	...	...	...

A Simple Approach: Nearest Neighbor



What if the unknown point is here?



Introducing: k-Nearest Neighbors (kNN)

- The concept shown here is an intuitive way of classifying unseen data point: *you are similar to those who are close to you*.

- The kNN method finds *k* data points that are nearest to the subject that you want to classify. We then assign the category to which the majority of neighbors belong to the subject.

- We would normally choose *k* to be an odd number larger than 1.

- This method has *nothing* to do with "neural network" (also called NN).

How do we measure the performance of kNN?

- For a simple system like kNN, the classification performance is entirely dependent on the "goodness" of labeled data.

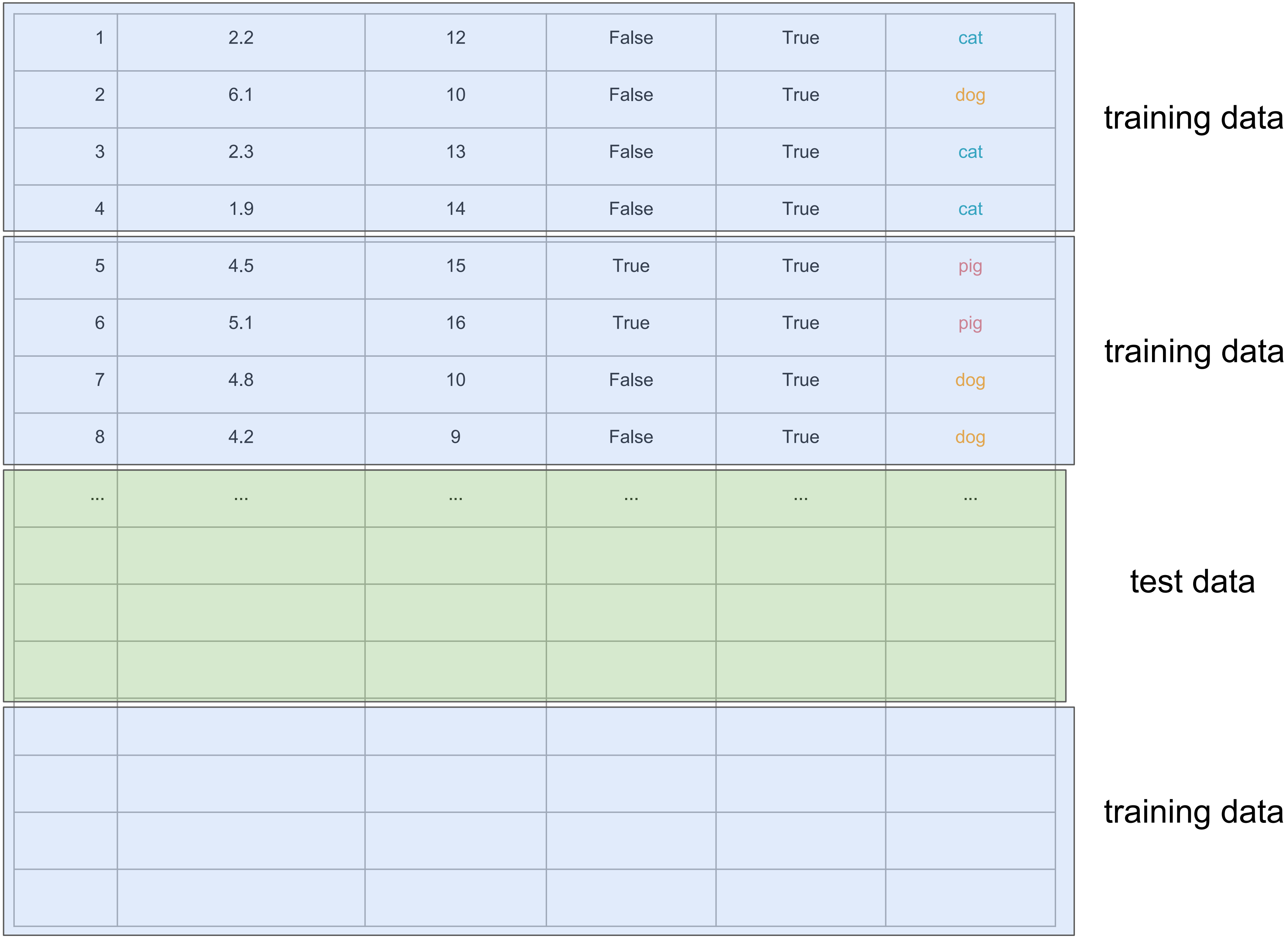
- To gauge the performance, we can split the data into two parts, where we use one part to act as the reference and the other part to test.

- The part acting as the reference is essentially the training set. Training set can defined as the data actually used to build the machine learning model, while the test set is used to provide a more objective classification result.

1	2.2	12	False	True	cat
2	6.1	10	False	True	dog
3	2.3	13	False	True	cat
4	1.9	14	False	True	cat
5	4.5	15	True	True	pig
6	5.1	16	True	True	pig
7	4.8	10	False	True	dog
8	4.2	9	False	True	dog
...	...	...	...	...	...

training data

test data



1	2.2	12	False	True	cat
2	6.1	10	False	True	dog
3	2.3	13	False	True	cat
4	1.9	14	False	True	cat

training data

5	4.5	15	True	True	pig
6	5.1	16	True	True	pig
7	4.8	10	False	True	dog
8	4.2	9	False	True	dog

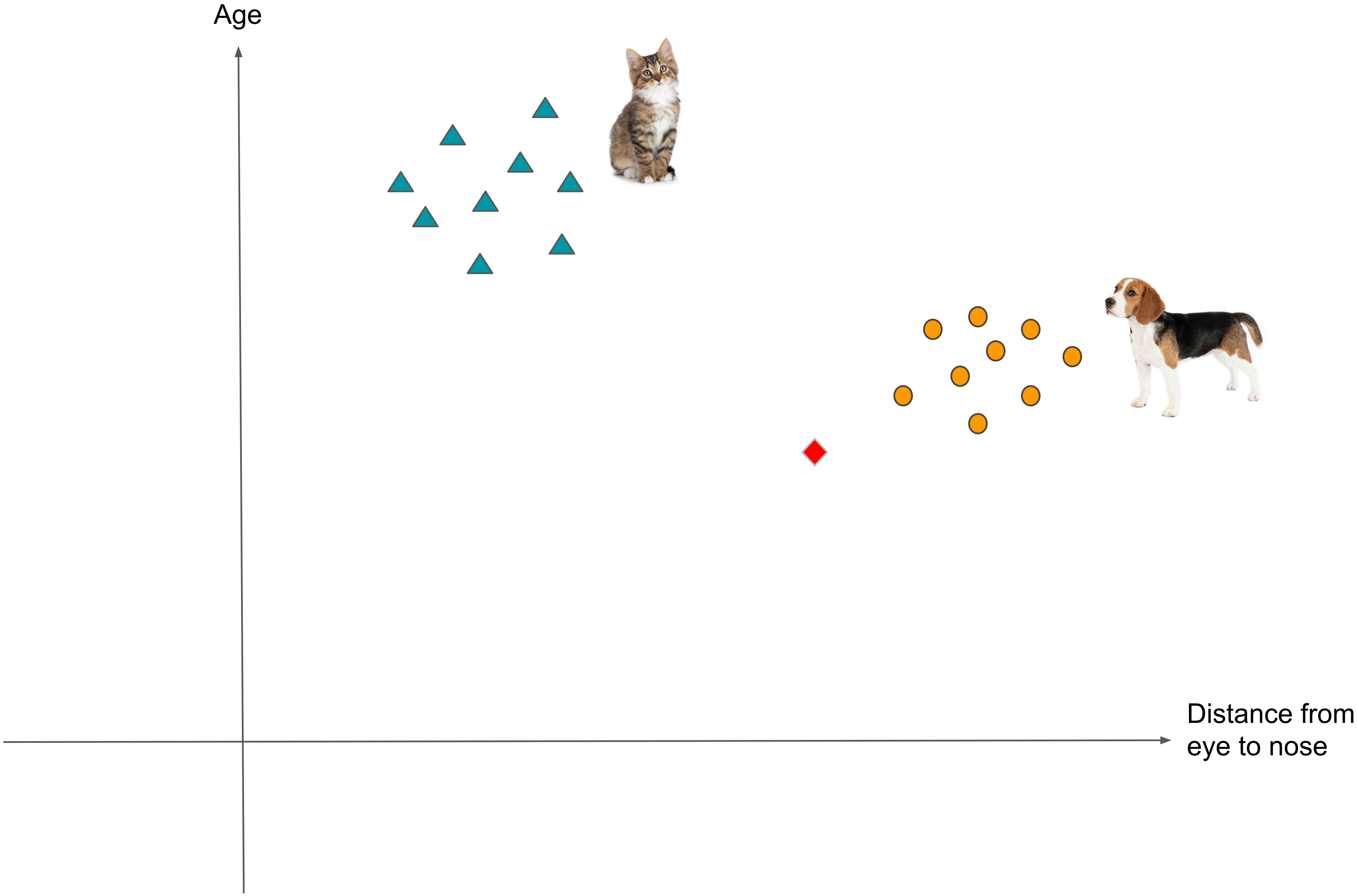
training data

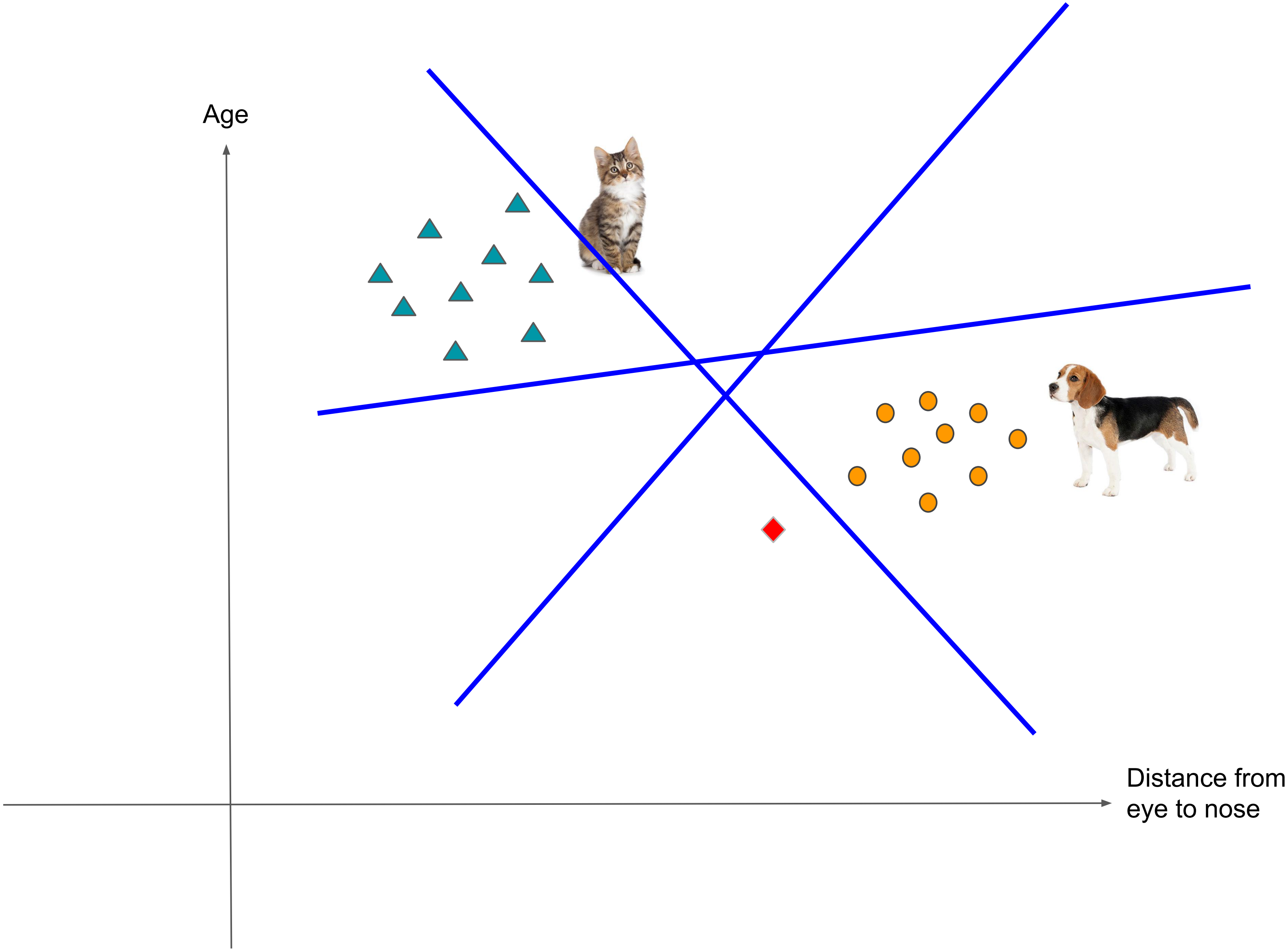
...	...	...	...	...	...

test data

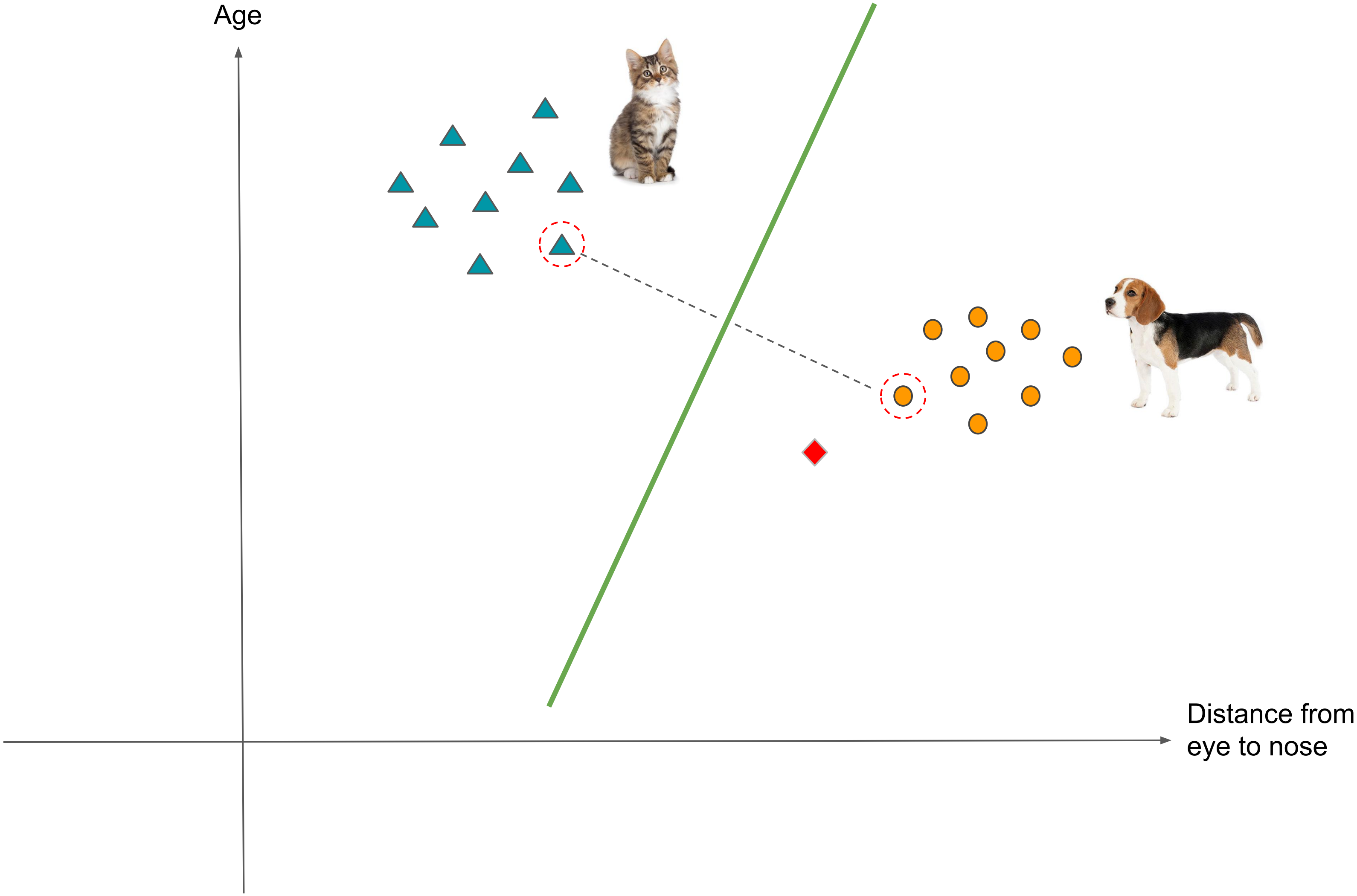
training data

Ok, that seems easy. What else?

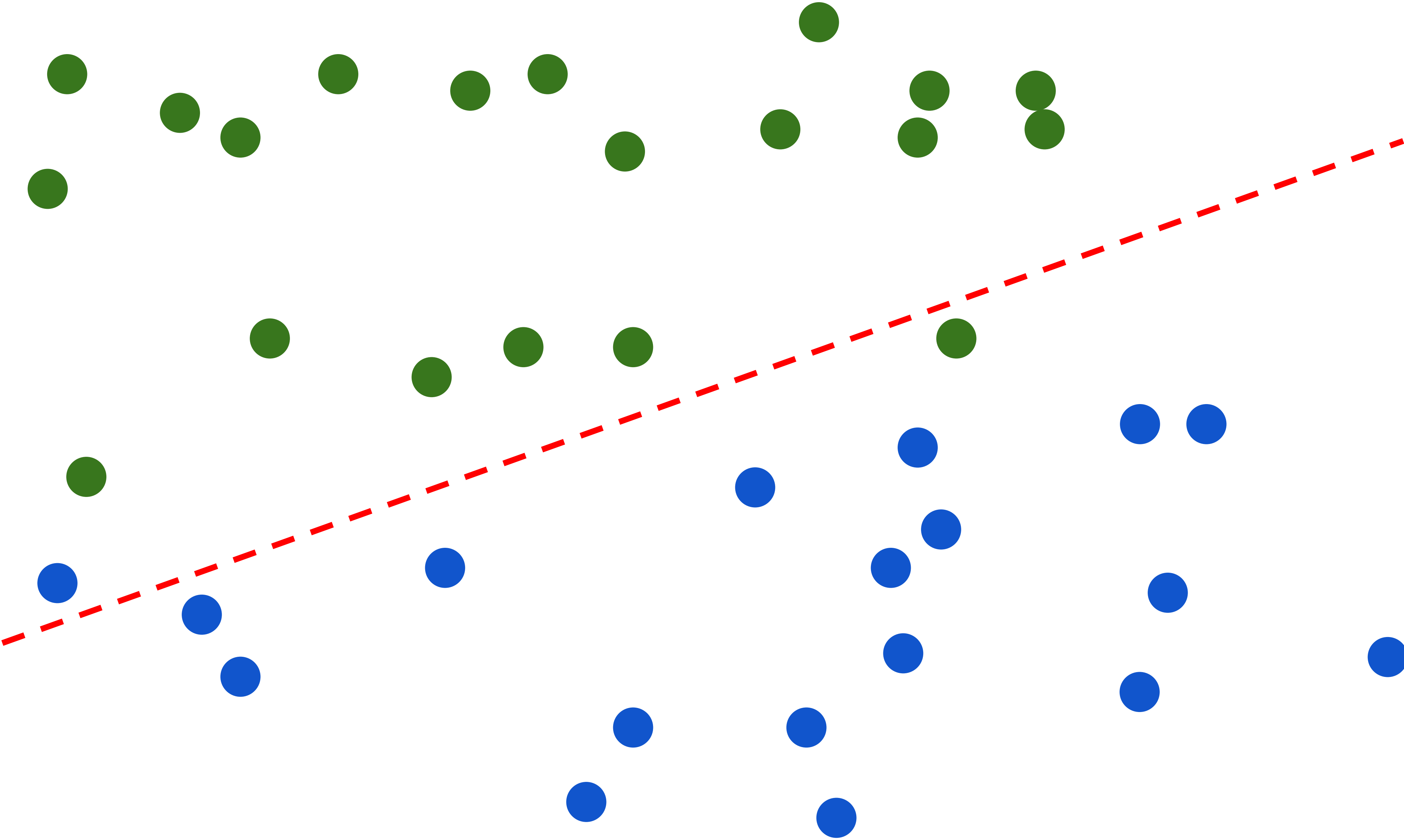


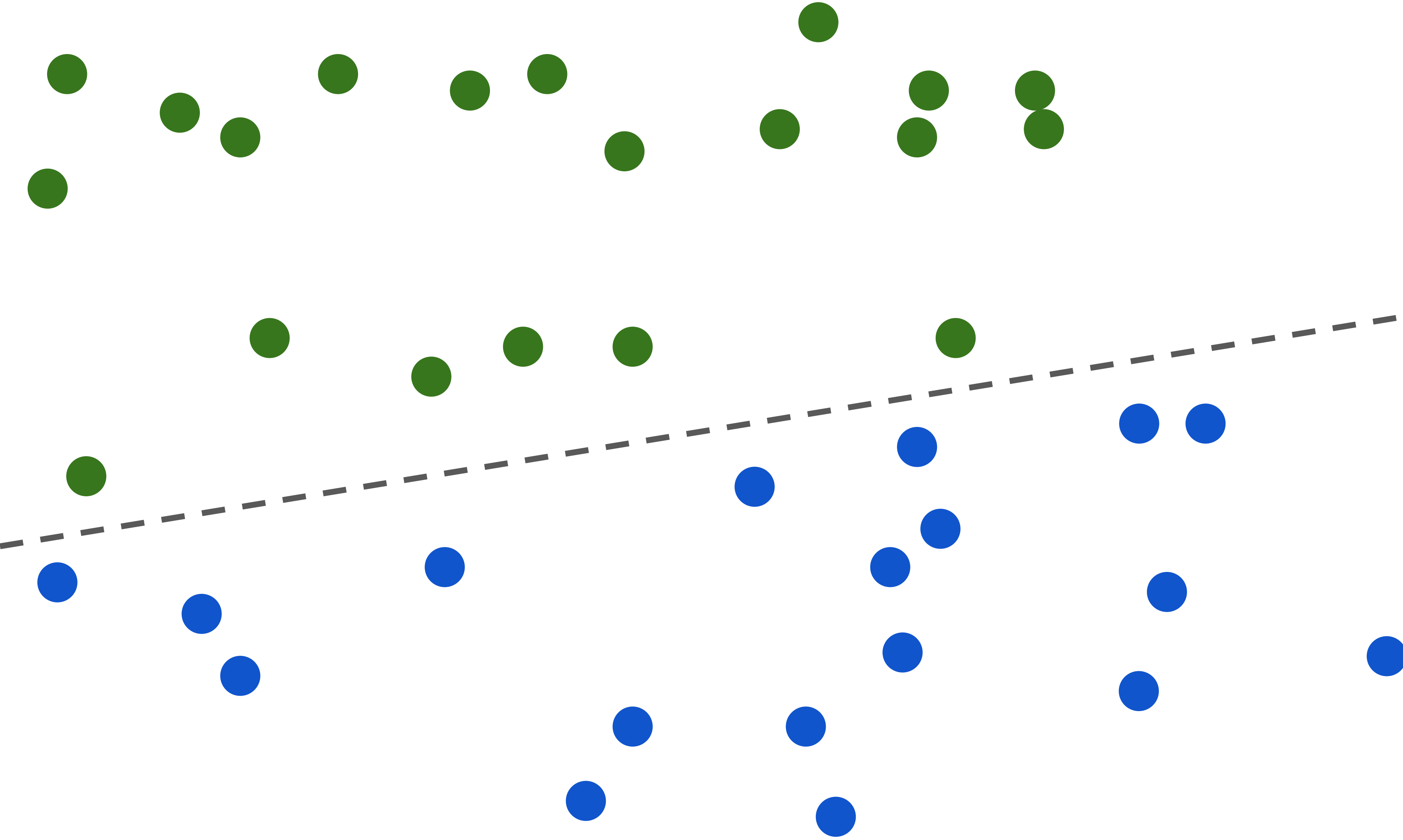

Age
Distance from
eye to nose

This is where Suppport Vector Machine (SVM) comes in.



預算案依法編列
理性審查
勿杯葛
啟動
屏東縣
恒春東港
觀光鐵道
台南市
先進運輸
系統
水上
高架化
民進黨自甘
立法院變立
編列違法
退回重擬
綠色執政
撒錢保證
違法編
民進黨自甘墮落
立法院變立法局
評估沒完
預算不得
退回重擬



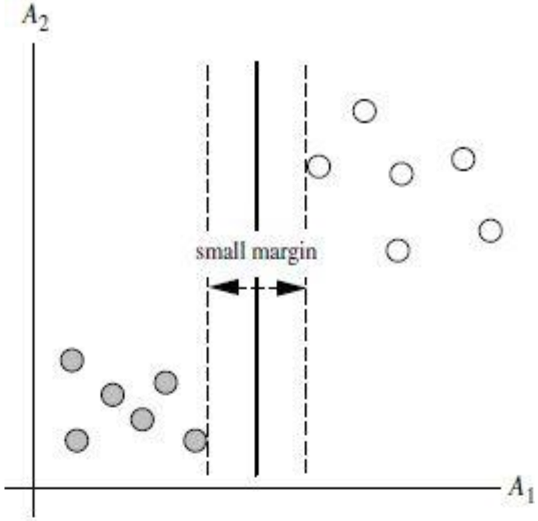


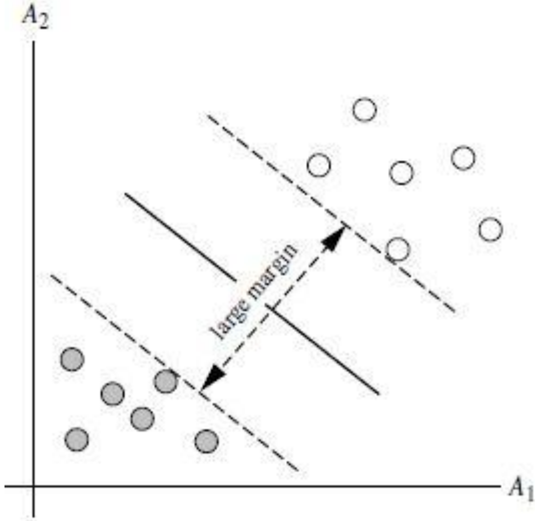
Support Vector Machine (SVM)

- To classify data, we can try to find a decision boundary between different data groups, whereas the boundary "cuts out" a space that has the maximum margin with all groups.

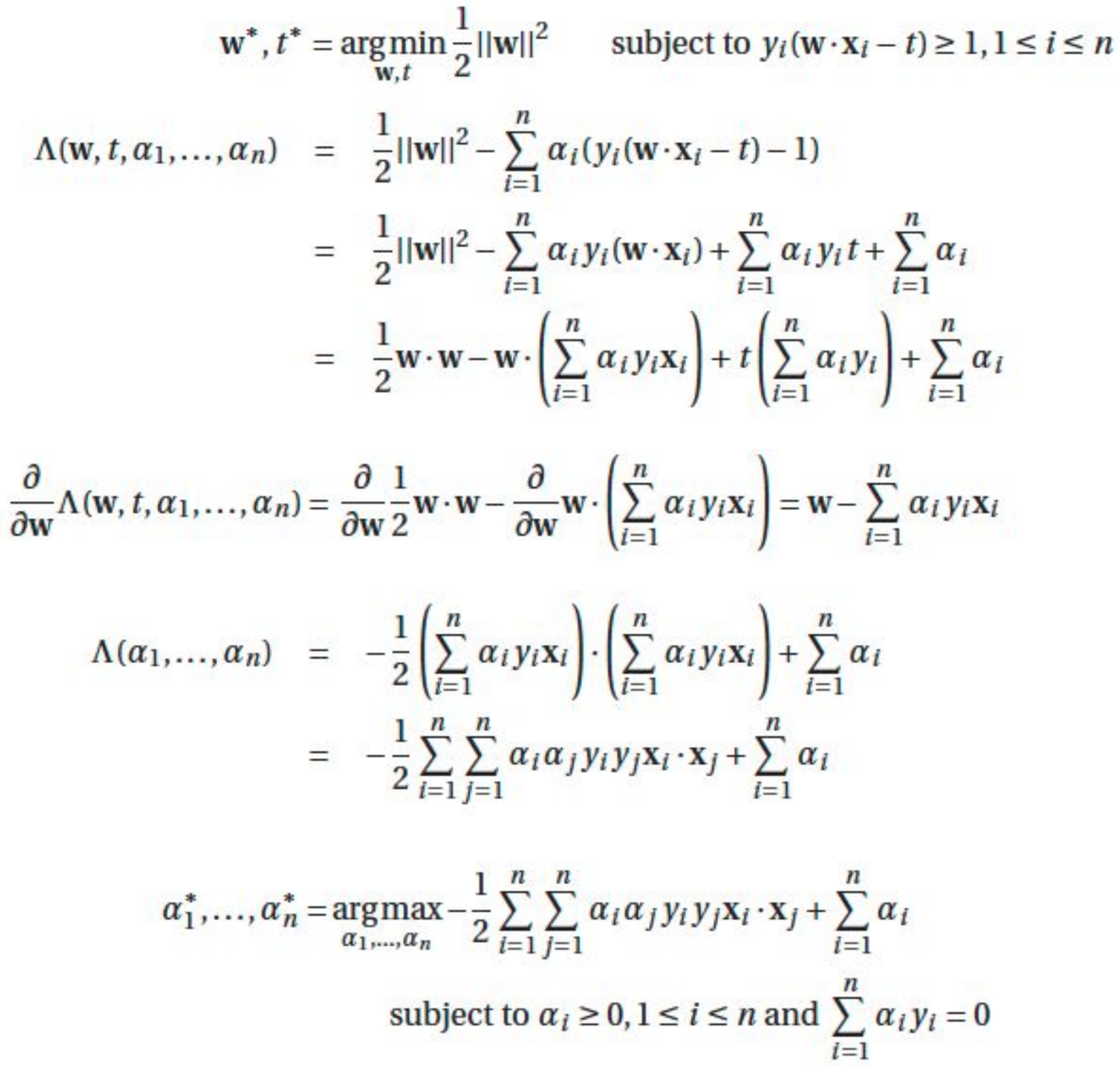
- This method sets out to find the most optimised decision boundary by finding data points that would be closest to the boundary as the "pivot". These closest data points are the support vector.

- SVM can be applied to classify between more than two categories.


A_2
small margin
A_1


A_2
large margin
A_1

In case you want to know the math ;)



$$\mathbf{w}^*, t^* = \operatorname*{arg\,min}_{\mathbf{w},t} \frac{1}{2}||\mathbf{w}||^2 \qquad \text{subject to } y_i(\mathbf{w}\cdot\mathbf{x}_i - t) \geq 1, 1 \leq i \leq n$$

$$\begin{aligned}
\Lambda(\mathbf{w}, t, \alpha_1, \ldots, \alpha_n) &= \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{n} \alpha_i (y_i(\mathbf{w}\cdot\mathbf{x}_i - t) - 1) \\
&= \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{n} \alpha_i y_i (\mathbf{w}\cdot\mathbf{x}_i) + \sum_{i=1}^{n} \alpha_i y_i t + \sum_{i=1}^{n} \alpha_i \\
&= \frac{1}{2}\mathbf{w}\cdot\mathbf{w} - \mathbf{w}\cdot\left(\sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i\right) + t\left(\sum_{i=1}^{n} \alpha_i y_i\right) + \sum_{i=1}^{n} \alpha_i
\end{aligned}$$

$$\frac{\partial}{\partial \mathbf{w}}\Lambda(\mathbf{w}, t, \alpha_1, \ldots, \alpha_n) = \frac{\partial}{\partial \mathbf{w}}\frac{1}{2}\mathbf{w}\cdot\mathbf{w} - \frac{\partial}{\partial \mathbf{w}}\mathbf{w}\cdot\left(\sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i\right) = \mathbf{w} - \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

$$\begin{aligned}
\Lambda(\alpha_1, \ldots, \alpha_n) &= -\frac{1}{2}\left(\sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i\right)\cdot\left(\sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i\right) + \sum_{i=1}^{n} \alpha_i \\
&= -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j + \sum_{i=1}^{n} \alpha_i
\end{aligned}$$

$$\alpha_1^*, \ldots, \alpha_n^* = \operatorname*{arg\,max}_{\alpha_1,\ldots,\alpha_n} -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j + \sum_{i=1}^{n} \alpha_i$$

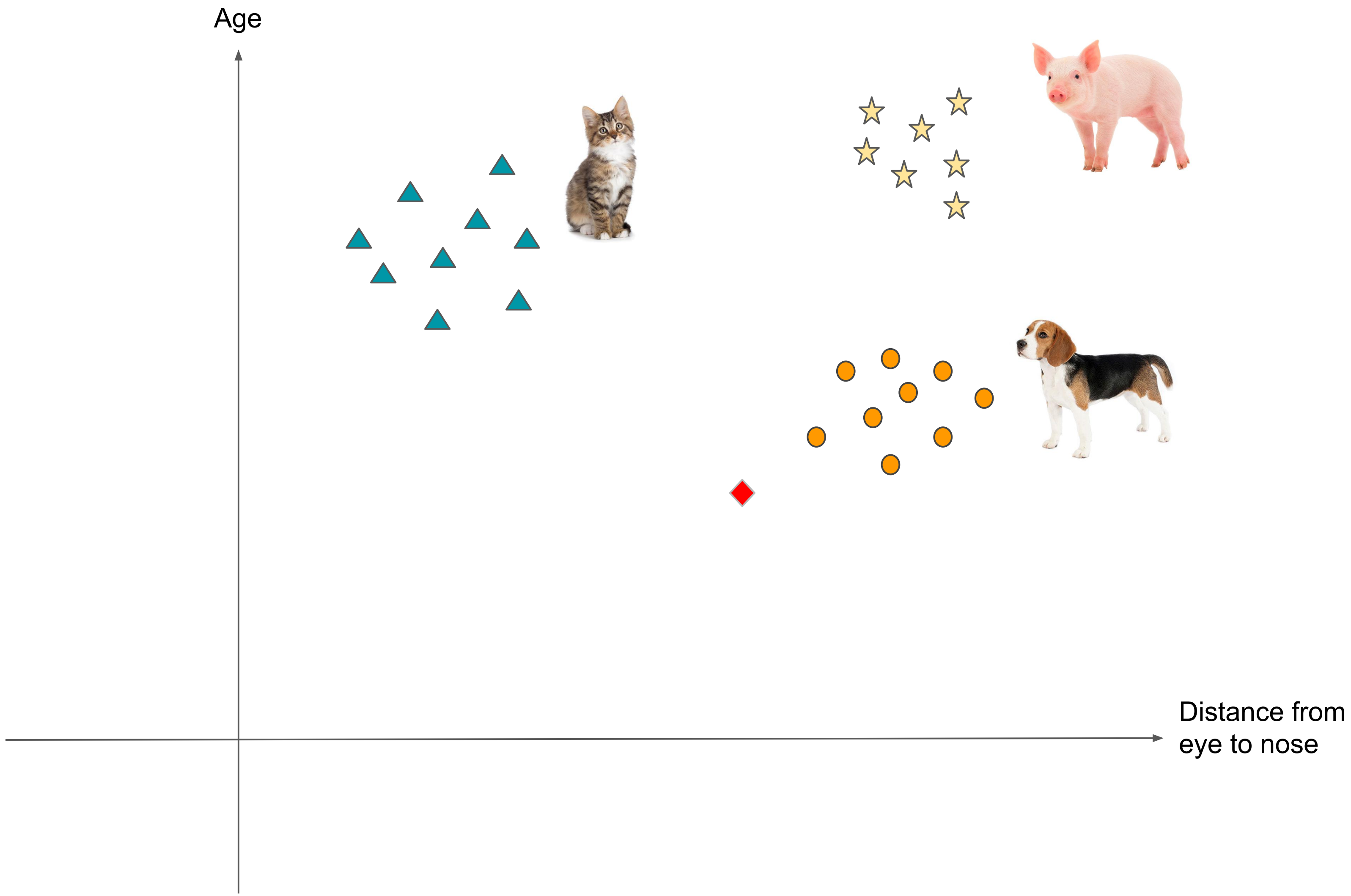
$$\text{subject to } \alpha_i \geq 0, 1 \leq i \leq n \text{ and } \sum_{i=1}^{n} \alpha_i y_i = 0$$

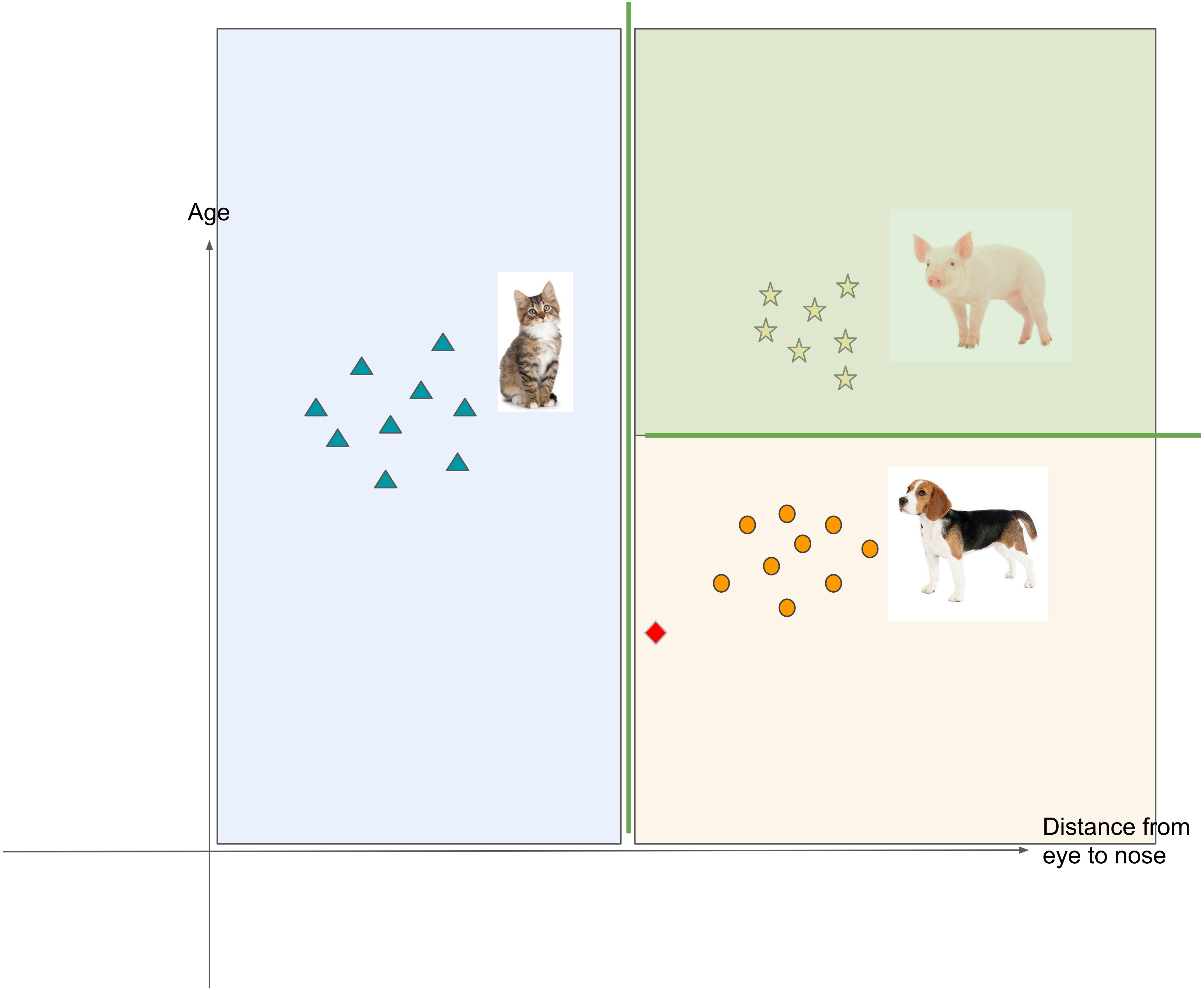
- Currently the most convenient and reliable library for running SVM is LibSVM, constructed in 2011 by Dr. Chih-Chung Chang and Dr. Chih-Jen Lin in National Taiwan University. (Citation: 37016)

- Originally written in C, it has been ported to Java, Python, R, MATLAB, etc.

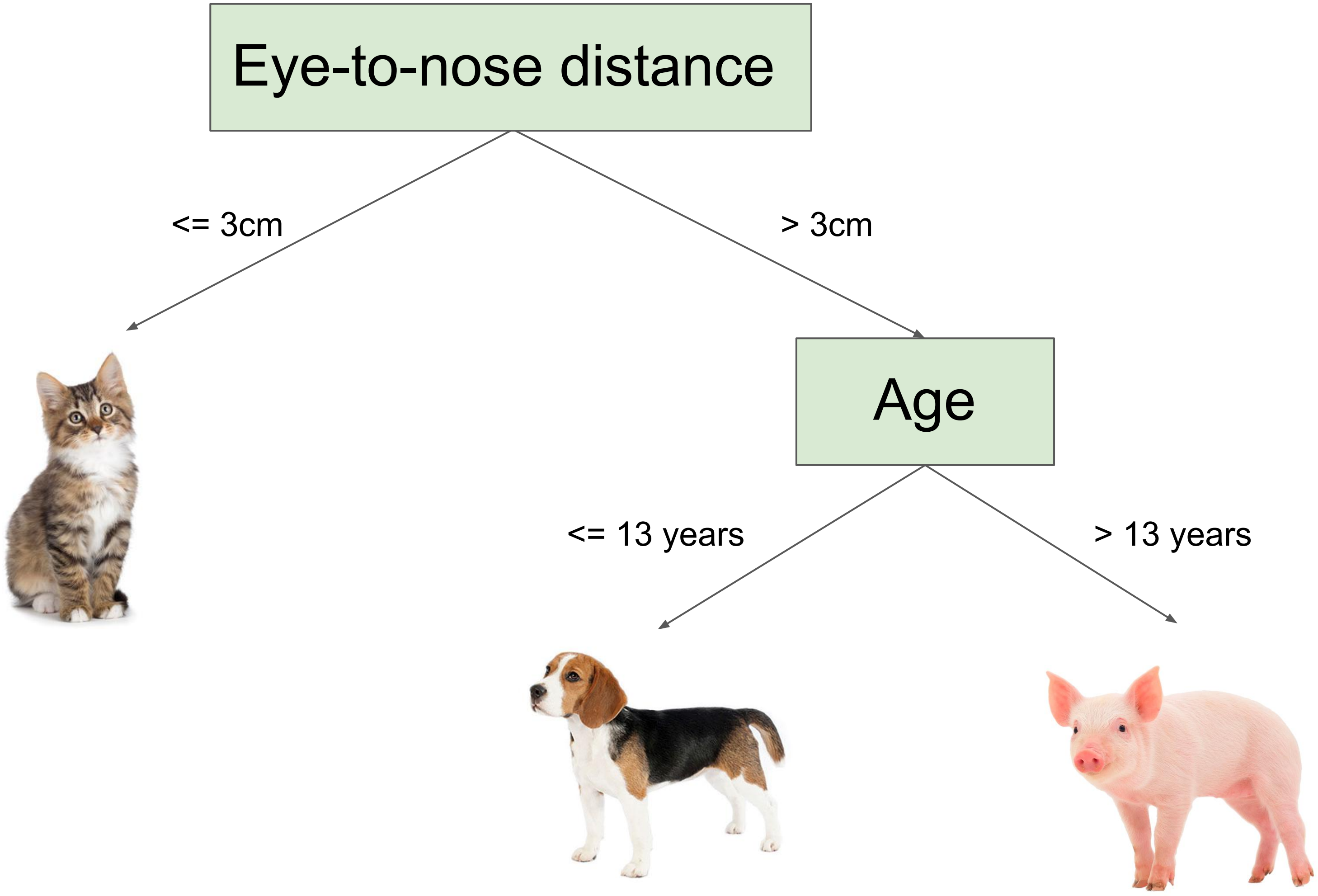
- The SVM toolkit you use in scikit-learn is also implemented based on libSVM.

Let's say your animal clinic gets a little busier




Age
Distance from eye to nose

Our logic



Eye-to-nose distance

<= 3cm

> 3cm

Age

<= 13 years

> 13 years

The concept you know all along: Decision Tree

- Before today you have probably already used decision tree in your daily life countless of times, e.g. deciding whether to watch a concert when you have a lot of work do, whether to ask a girl out, etc.

- We can apply the same thing to classification task in machine learning, just as how the zoologists classify animal into different species.

- A good decision tree should effectively classify the most amount of data within a reasonable height.

How do we determine the split?

- A good split should give you more certainty about the nature of data.

Data Index	Distance from eyes to nose	Age	Hoofed	Four-legged	Category
1	2.2	12	False	True	cat
2	6.1	10	False	True	dog
3	2.3	13	False	True	cat
4	1.9	14	False	True	cat
5	4.5	15	True	True	pig
6	5.1	16	True	True	pig
7	4.8	10	False	True	dog
8	4.2	9	False	True	dog
...	...	...	...	...	...

	✓	✓	✓	✗	
Data Index	Distance from eyes to nose	Age	Hoofed	Four-legged	Category
1	2.2	12	False	True	cat
2	6.1	10	False	True	dog
3	2.3	13	False	True	cat
4	1.9	14	False	True	cat
5	4.5	15	True	True	pig
6	5.1	16	True	True	pig
7	4.8	10	False	True	dog
8	4.2	9	False	True	dog
...	...	...	...	...	...

That's why we need to measure the *uncertainty*

- In information theory, we use entropy to describe the *uncertainty* of data:

$$H(Y) = -\sum_{i=1}^{k} P(Y = y_i) \log_2 P(Y = y_i)$$

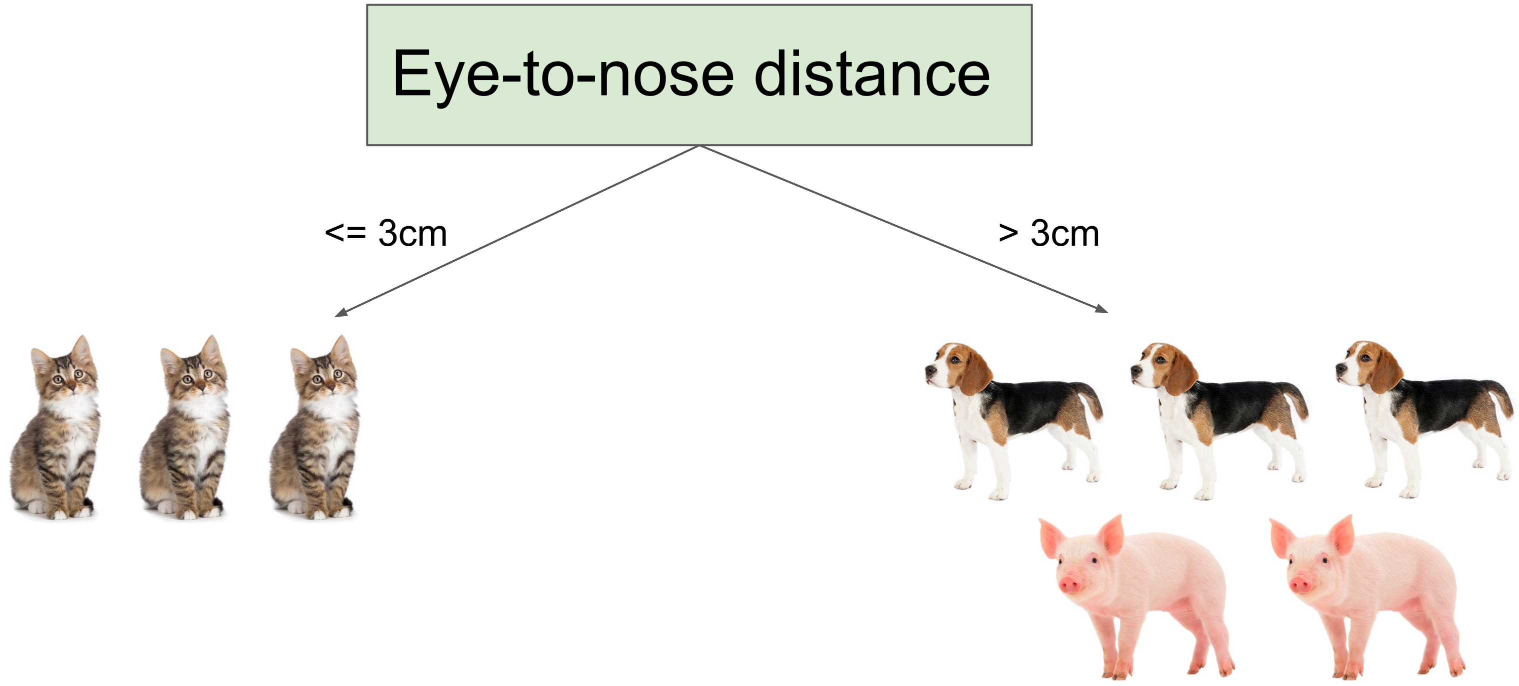
- Entropy is a physical measure of how "messy" a system is. In essence, the higher the entropy, the higher the uncertainty.

$$H(Y) = -\sum_{i=1}^{k} P(Y = y_i) \log_2 P(Y = y_i)$$

Data Index	Distance from eyes to nose	Age	Hoofed	Four-legged	Category
1	2.2	12	False	True	cat
2	6.1	10	False	True	dog
3	2.3	13	False	True	cat
4	1.9	14	False	True	cat
5	4.5	15	True	True	pig
6	5.1	16	True	True	pig
7	4.8	10	False	True	dog
8	4.2	9	False	True	dog

$$H = -\left(\frac{3}{8} \log_2 \frac{3}{8} + \frac{3}{8} \log_2 \frac{3}{8} + \frac{2}{8} \log_2 \frac{2}{8} \right) = 1.5613$$

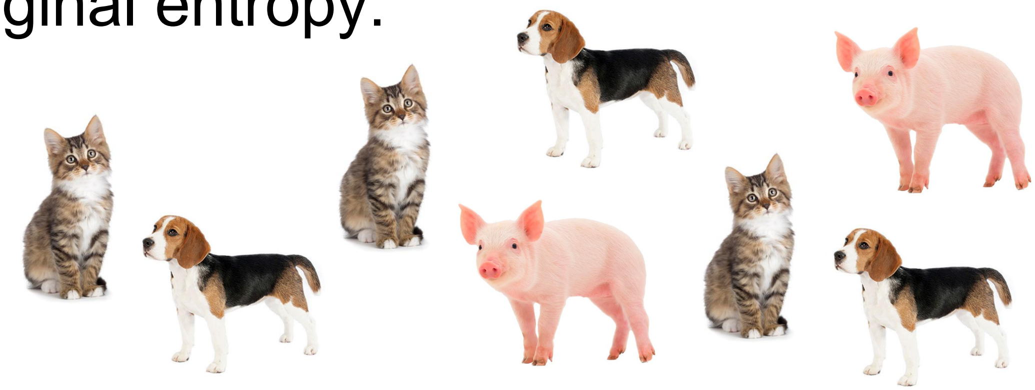
$$H(Y \mid X) = -\sum_{j=1}^{v} P(X = x_j) \sum_{i=1}^{k} P(Y = y_i \mid X = x_j) \log_2 P(Y = y_i \mid X = x_j)$$



$$H = -\frac{3}{8}\left(1 \log_2 1 + 0 \log_2 0 + 0 \log_2 0\right)$$

$$-\frac{5}{8}\left(0 \log_2 0 + \frac{3}{5} \log_2 \frac{3}{5} + \frac{2}{5} \log_2 \frac{2}{5}\right) = 0.2318$$

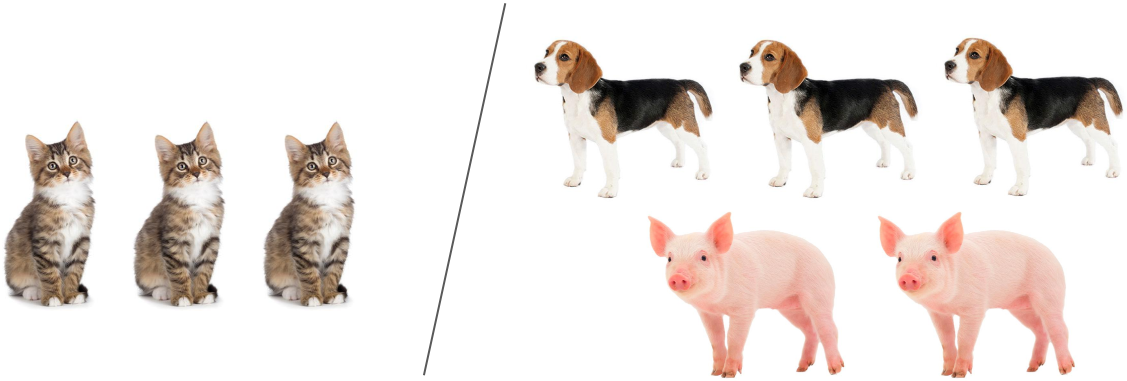
Original entropy:



= 1.5613

Difference: 1.3295
(or "information gain")

Entropy after split:



= 0.2318

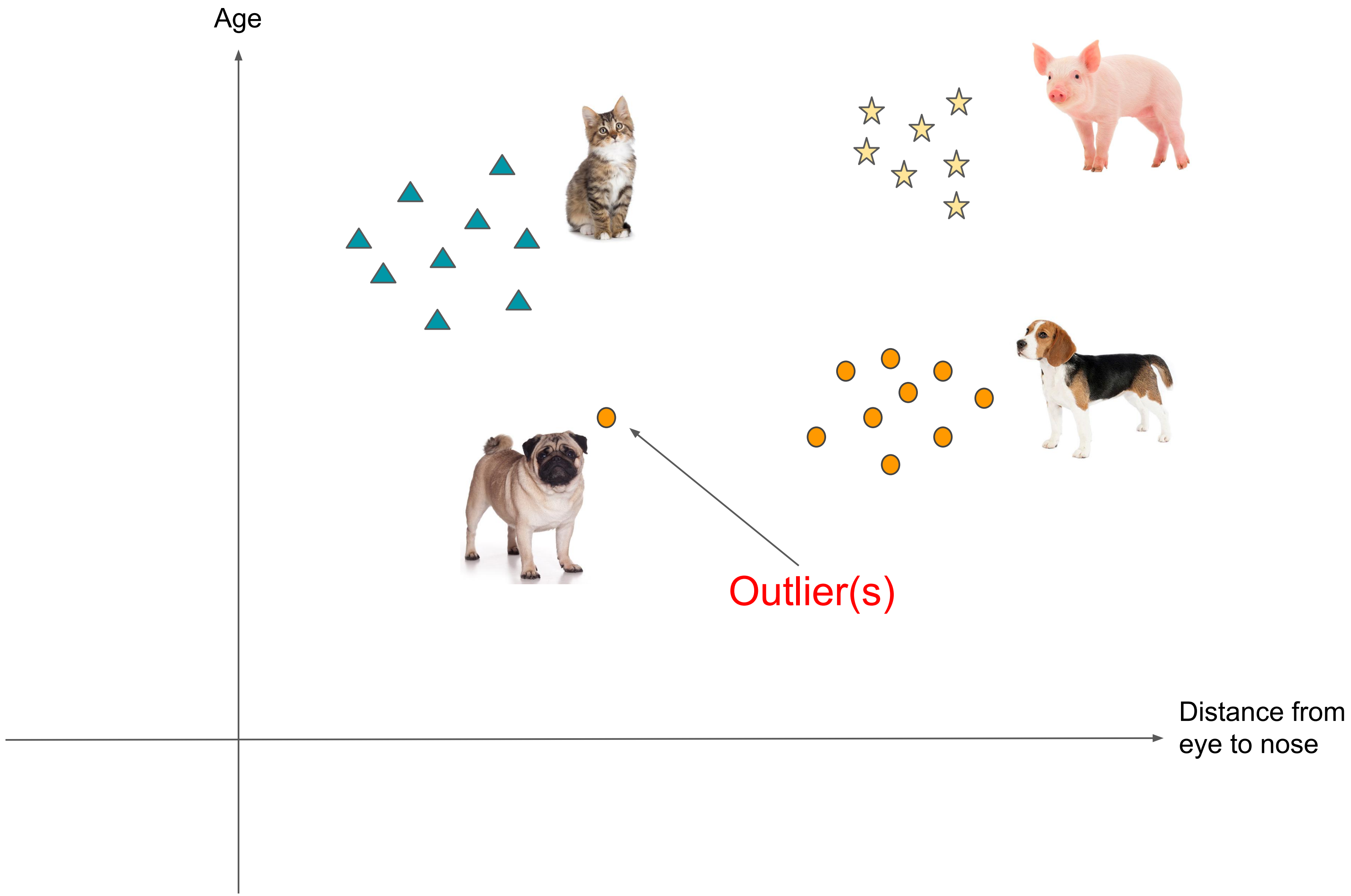
How to grow a decision tree

- To grow a decision tree classifier, we choose the attribute that gives us the most information gain to split the data. We then recursively do this to all subsets of data.

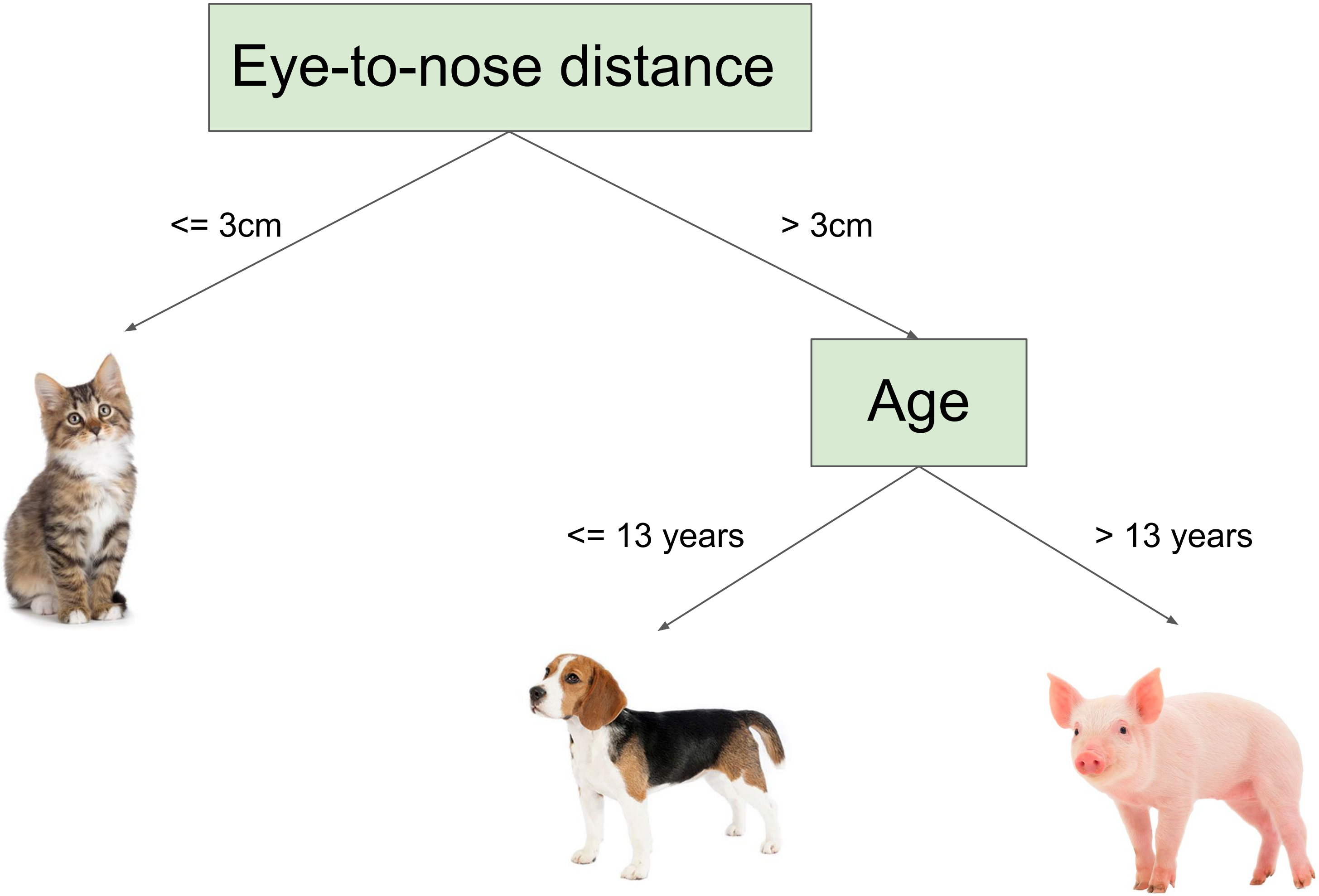
- We can keep growing the tree until all data in each leaf node has the same output (label).

- We can also stop when the data in the node all has the same attributes.

What happens in real world, though...



Effects of Outliers



Eye-to-nose distance

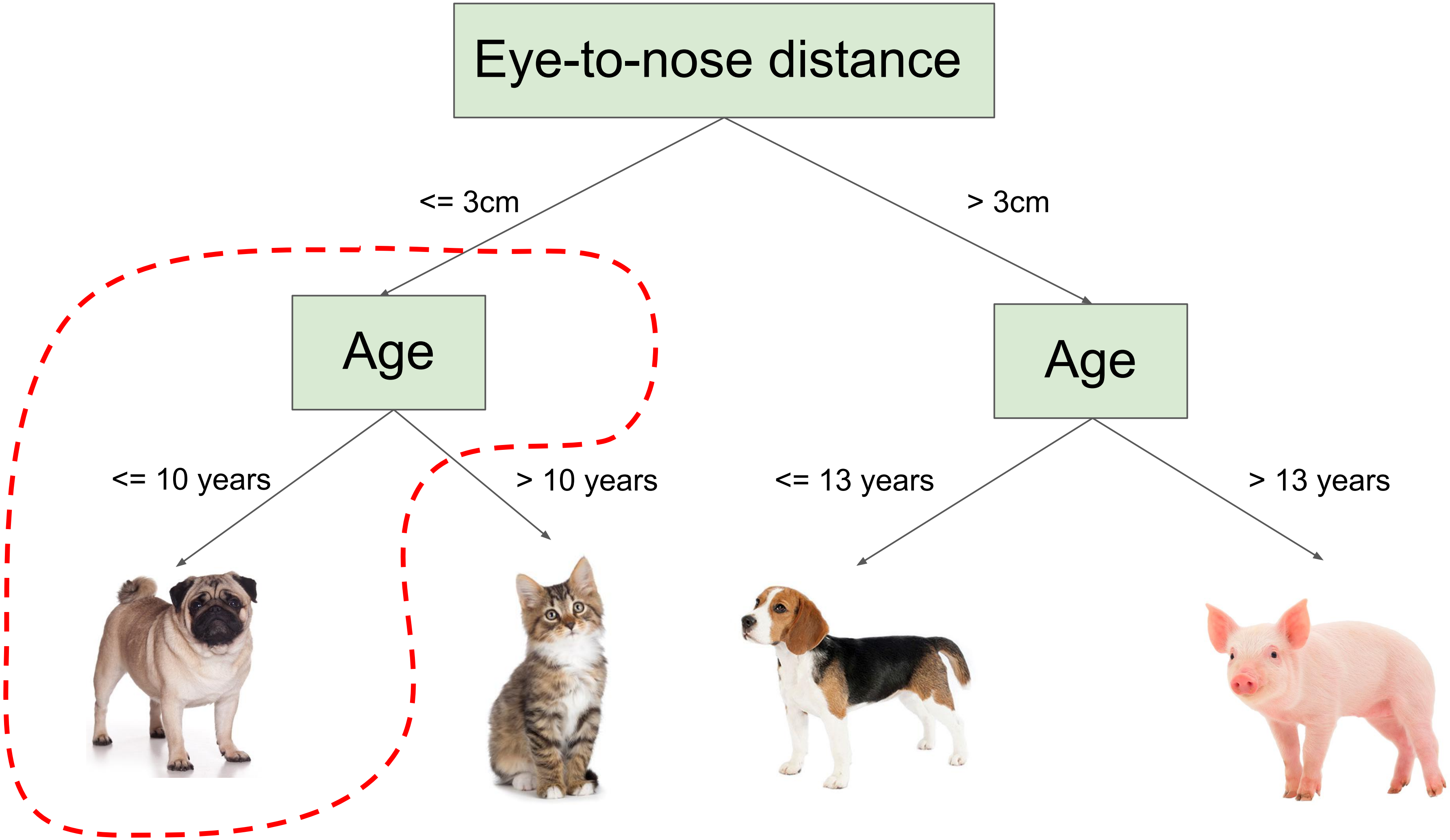
<= 3cm

> 3cm

Age

<= 13 years

> 13 years


Eye-to-nose distance
<= 3cm
> 3cm
Age
Age
<= 10 years
> 10 years
<= 13 years
> 13 years

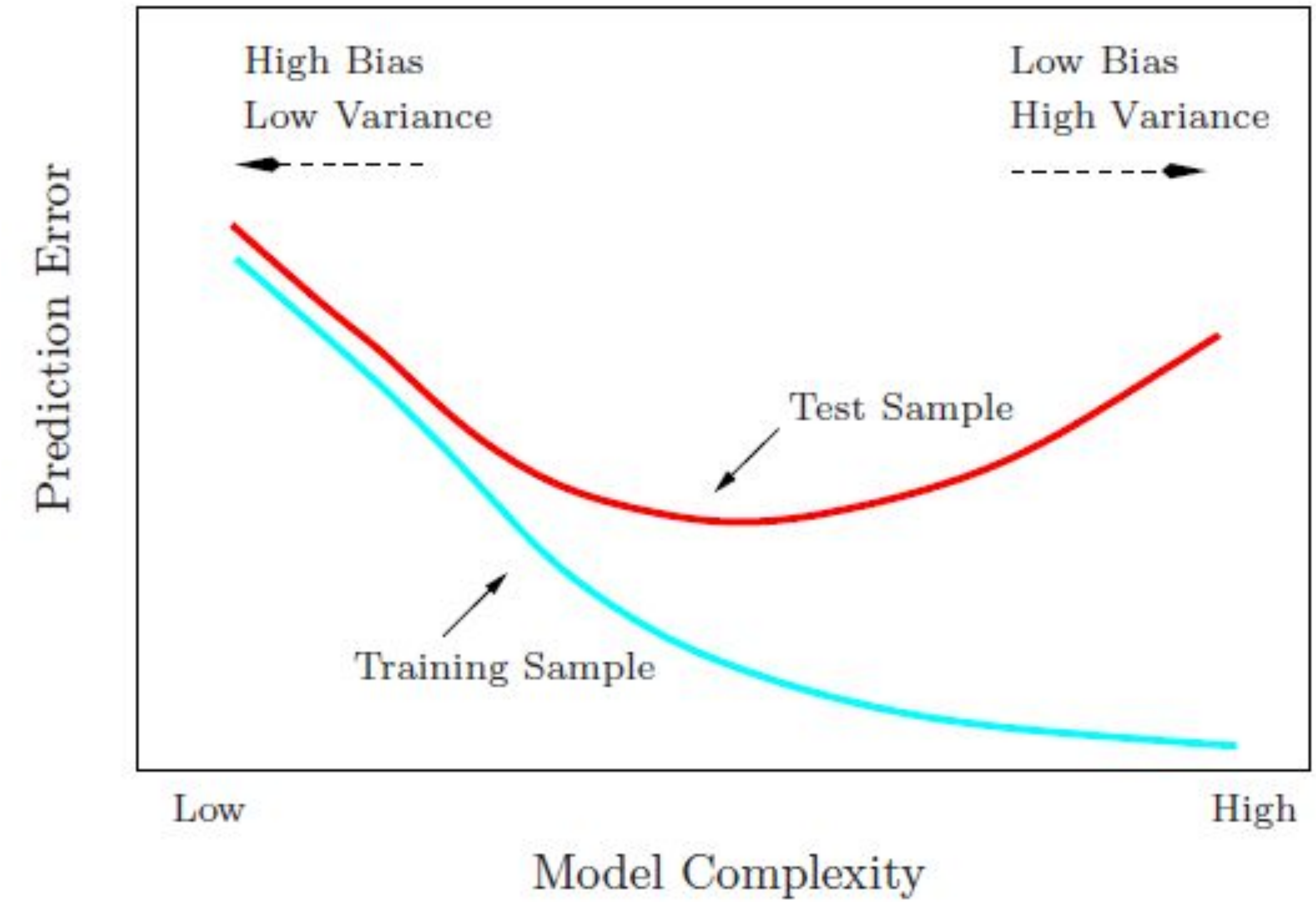
The annoying (& often unavoidable) thing: Overfitting

- The process of training a model according to some data can also be called as "fitting" the data. If, however, we fit the model *too* precisely according to the data, we might risk overfitting.

Applicable to most people (data)
=> Can be generalized

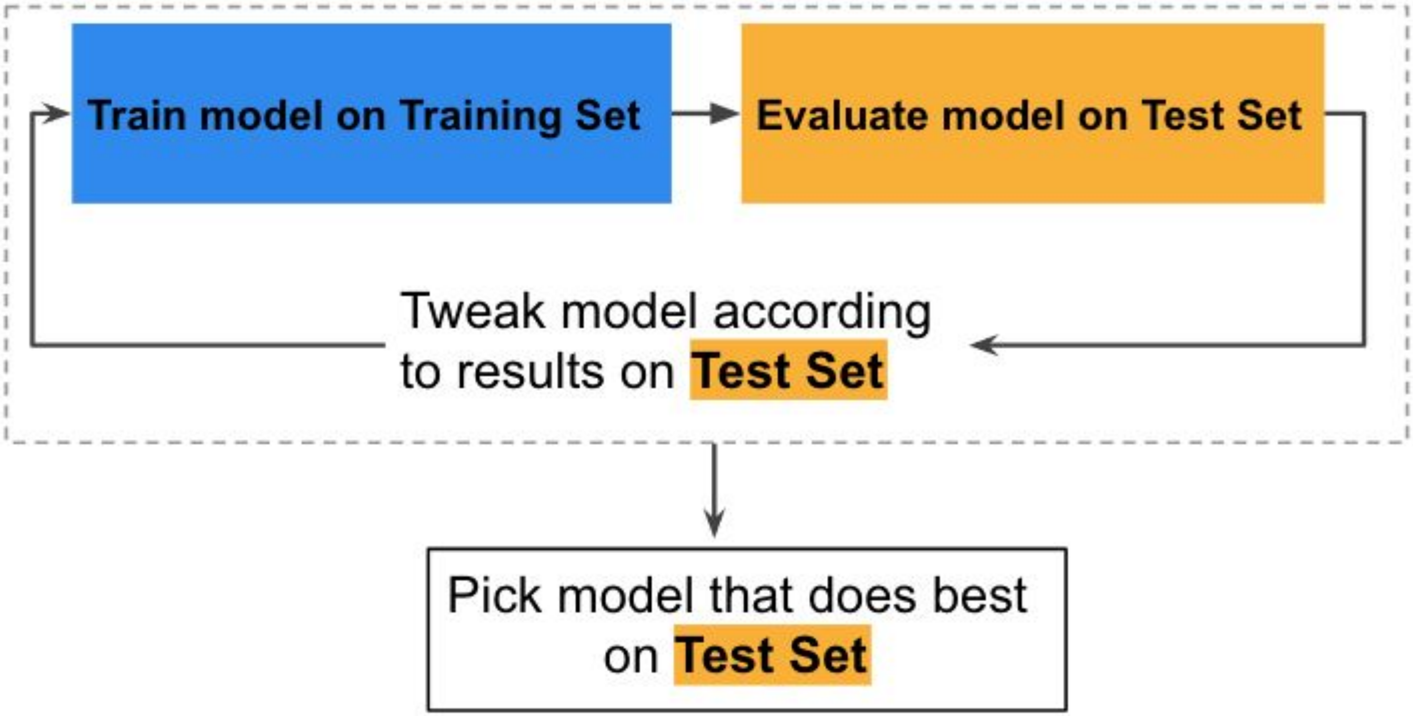
Overfit!

Bias/Variance Tradeoff



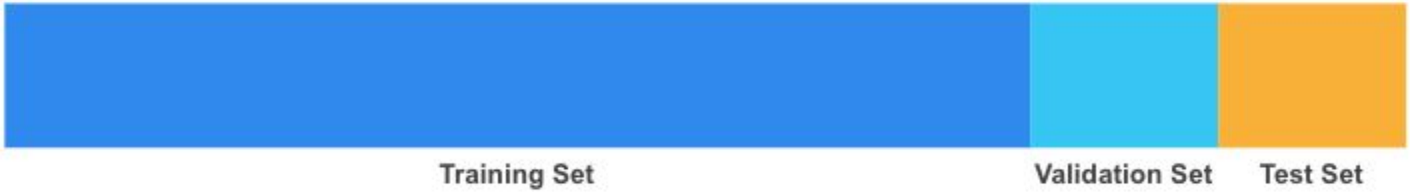
How to avoid overfitting

- The first and foremost essential step to avoid overfitting would be splitting the dataset into training set and test set.


Train model on Training Set
Evaluate model on Test Set
Tweak model according to results on Test Set
Pick model that does best on Test Set

- You can also allocate part of your data as the validation set.

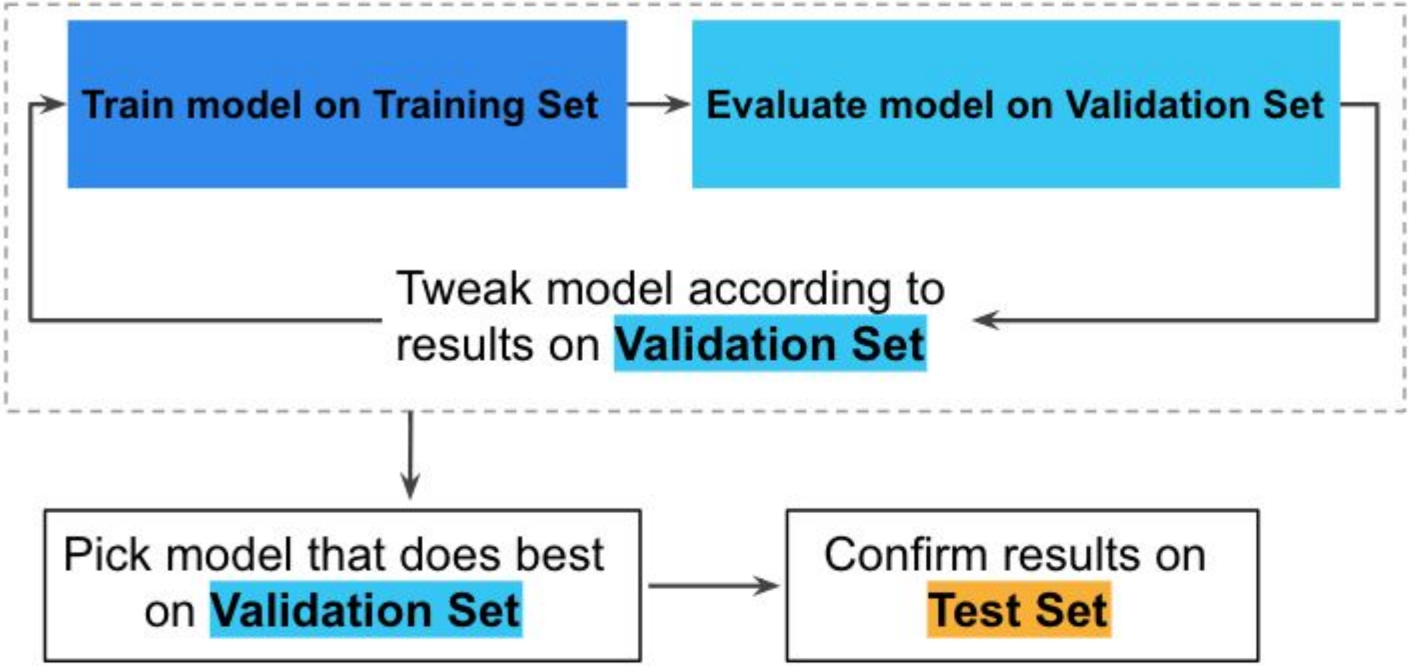
Validation Set



Training Set

Validation Set

Test Set



Train model on Training Set

Evaluate model on Validation Set

Tweak model according to results on **Validation Set**

Pick model that does best on **Validation Set**

Confirm results on **Test Set**

- For decision tree, we can also try the following to avoid overfitting:
  - early stopping: set maximum depth for the tree, stop splitting when data of a node reaches a minimum number, etc.
  - pruning: grow a complex tree, prune later.

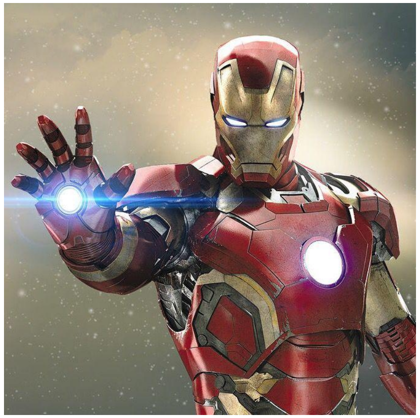
- Or, we can try other kind of decision tree-variants. The following two methods are what we call "ensemble methods".

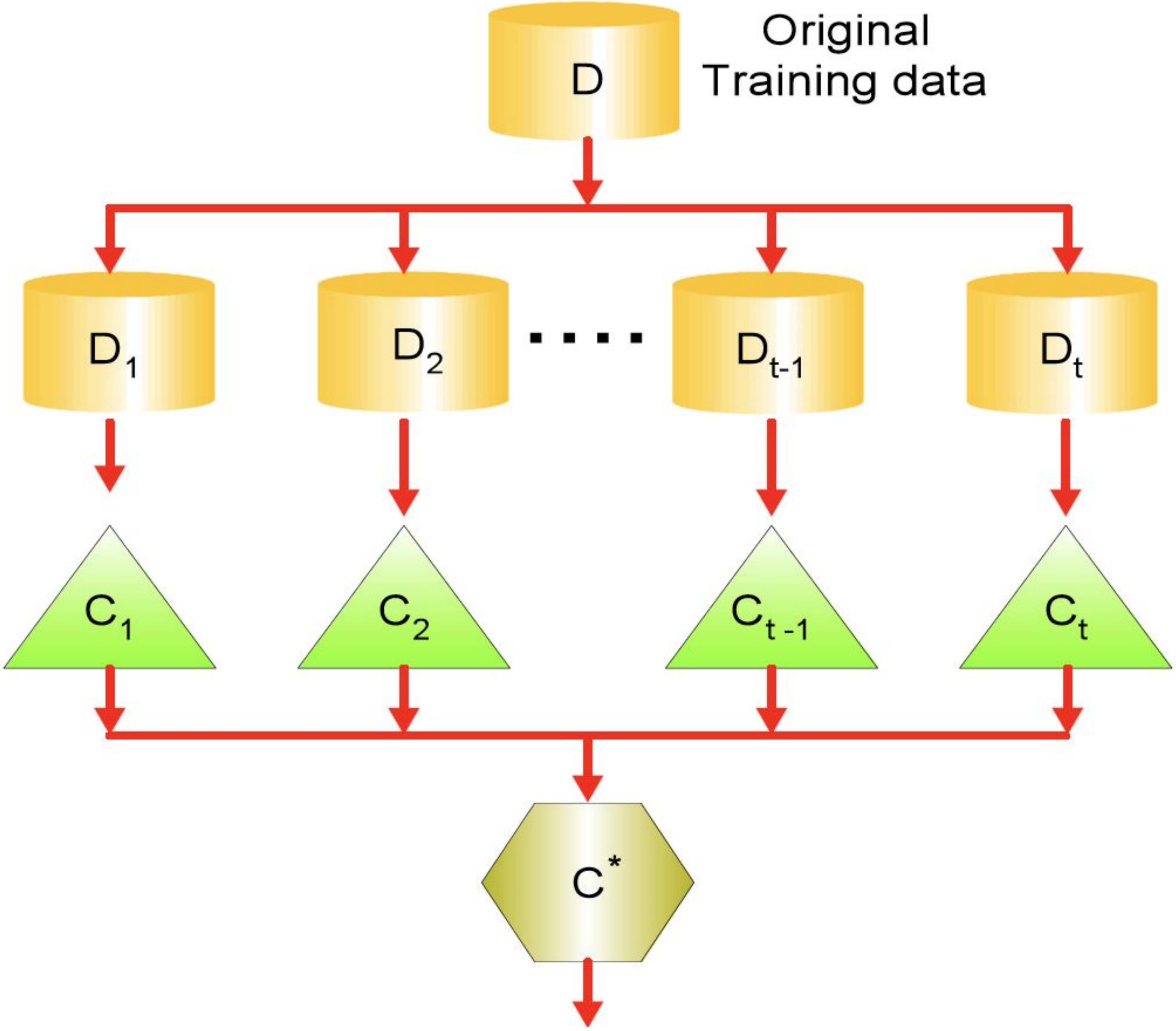
Bagging (or Bootstrap Aggregating)

- In statistics, bootstrapping is any test or metric that relies on random sampling with replacement.

- Bootstrap aggregating, or Bagging, refers to building multiple decision trees using multiple subsets of randomly sampled data from the original pool. We can then use these trees to provide a majority of "votes" on the answer.

>





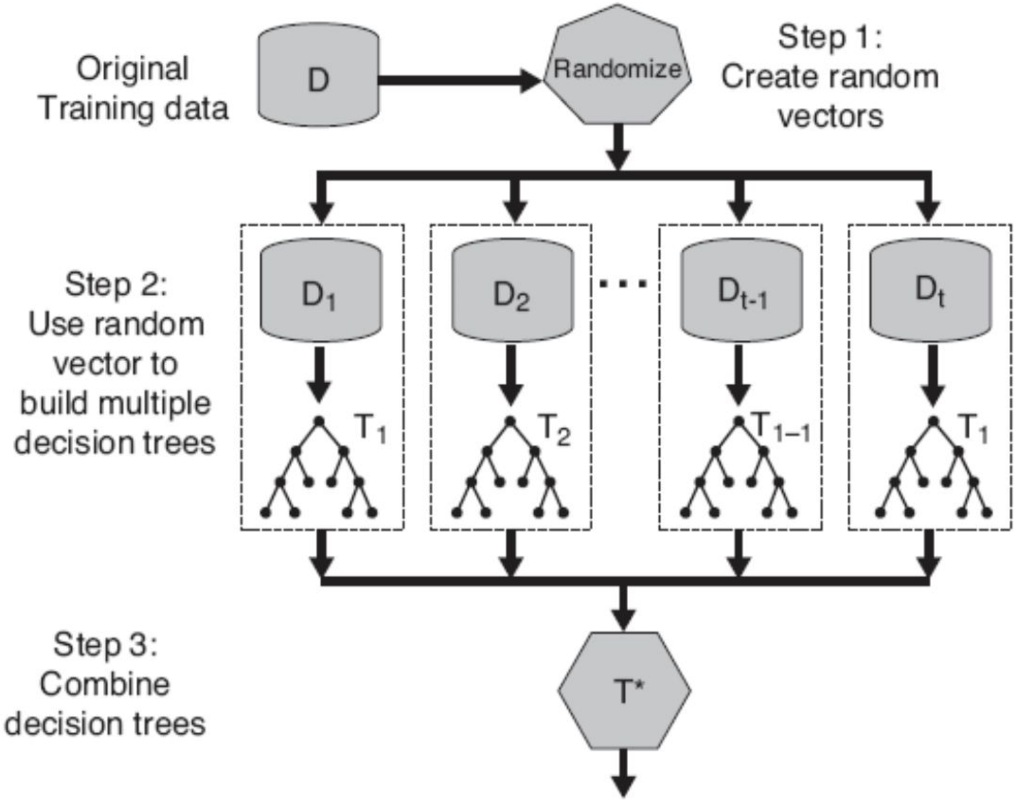
Step 1:
Create Multiple
Data Sets

Step 2:
Build Multiple
Classifiers

Step 3:
Combine
Classifiers

Another common method: Random Forest

- In addition to randomly sampling data to create different subsets (bootstrapping), random forest creates multiple decision trees by randomly selecting different subsets of attributes to find the best split at each node.


Original Training data
D
Randomize
Step 1: Create random vectors
Step 2: Use random vector to build multiple decision trees
D1
D2
...
Dt-1
Dt
T1
T2
T1–1
T1
Step 3: Combine decision trees
T*

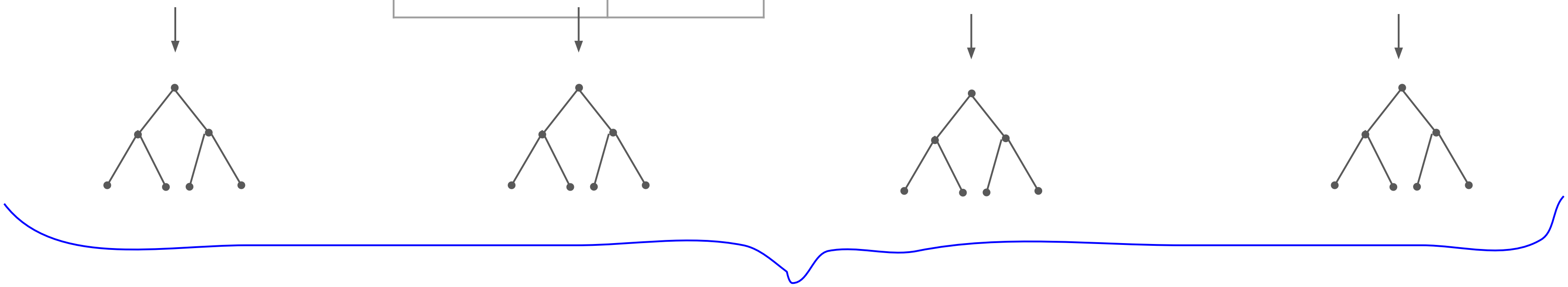
Data Index	Distance from eyes to nose	Age	Hoofed	Four-legged	Category
1	2.2	12	False	True	cat
2	6.1	10	False	True	dog
3	2.3	13	False	True	cat
...	...	...	...	...	...

Distance from eyes to nose	Age
2.2	12
6.1	10
2.3	13
...	...

Distance from eyes to nose	Hoofed
2.2	False
6.1	False
2.3	False
...	...

Hoofed	Four-legged
False	True
False	True
False	True
...	...

Age	Four-legged
12	True
10	True
13	True
...	...



The result

Evaluating a model

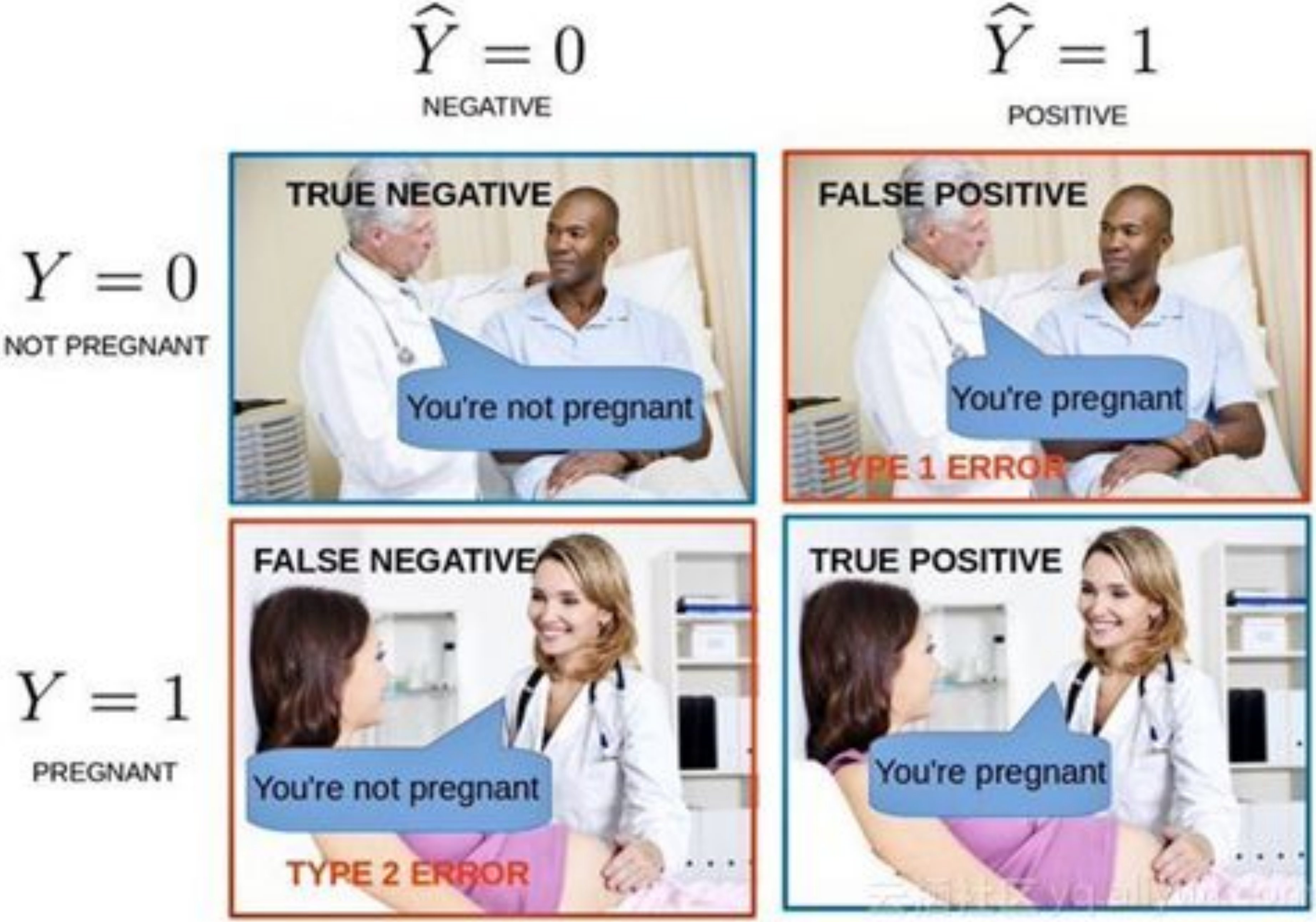
- To evaluate how well a (binary) classification model performs, we use metrics such as precision, recall, and accuracy.

		Actual value	
		True	False
Predicted value	Positive	True positive (TP)	False positive (FP)
	Negative	False negative (FN)	True negative (TN)

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN}$$


$\widehat{Y} = 0$
NEGATIVE
$\widehat{Y} = 1$
POSITIVE
$Y = 0$
NOT PREGNANT
TRUE NEGATIVE
You're not pregnant
FALSE POSITIVE
You're pregnant
TYPE 1 ERROR
$Y = 1$
PREGNANT
FALSE NEGATIVE
You're not pregnant
TYPE 2 ERROR
TRUE POSITIVE
You're pregnant

Scenario 1:

		Actual value	
		Have cancer	Safe
Predicted value	Have cancer	9	5
	Safe	1	1000

$$\text{Accuracy} = \frac{9+1000}{9+5+1+1000} = 99.41\%$$

$$\text{Precision} = \frac{9}{9 + 5} = 64.29\%$$

Scenario 2:

		Actual value	
		Have cancer	Safe
Predicted value	Have cancer	5	1
	Safe	5	1004

$$\text{Accuracy} = \frac{5+1004}{9+5+1+1000} = 99.41\%$$

$$\text{Precision} = \frac{5}{5 + 1} = 83.33\%$$

Scenario 1:

		Actual value	
		Have cancer	Safe
Predicted value	Have cancer	45	5
	Safe	190	1000

$$\text{Precision} = \frac{45}{45 + 5} = 90\%$$

$$\text{Recall} = \frac{45}{45 + 190} = 19.15\%$$

Scenario 2:

		Actual value	
		Have cancer	Safe
Predicted value	Have cancer	200	800
	Safe	35	205

$$\text{Precision} = \frac{200}{200 + 800} = 20\%$$

$$\text{Recall} = \frac{200}{200 + 35} = 85.11\%$$

That's why we have F1-score

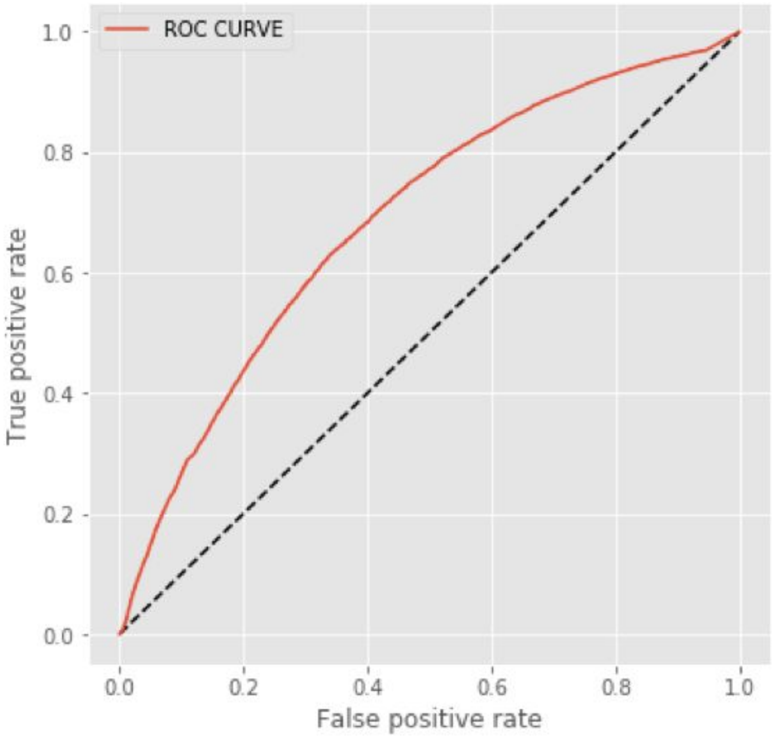
- F1-score (or F-score) is the harmonic mean between precision and recall:

$$F1 = 2 * \frac{1}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

Also, the ROC curve

Receiver Operating Characteristic (ROC) curve

- Sometimes we hope to investigate the ability of a model getting the most amount of correct classification while minimizing the false positives.
- A receiver operating characteristic curve plots the relationship between false positive rate and true positive rate of a model.



Area Under Curve (AUC) = 0.6876

"And that's all I have to say about that."

--- Forrest Gump, *Forrest Gump*.

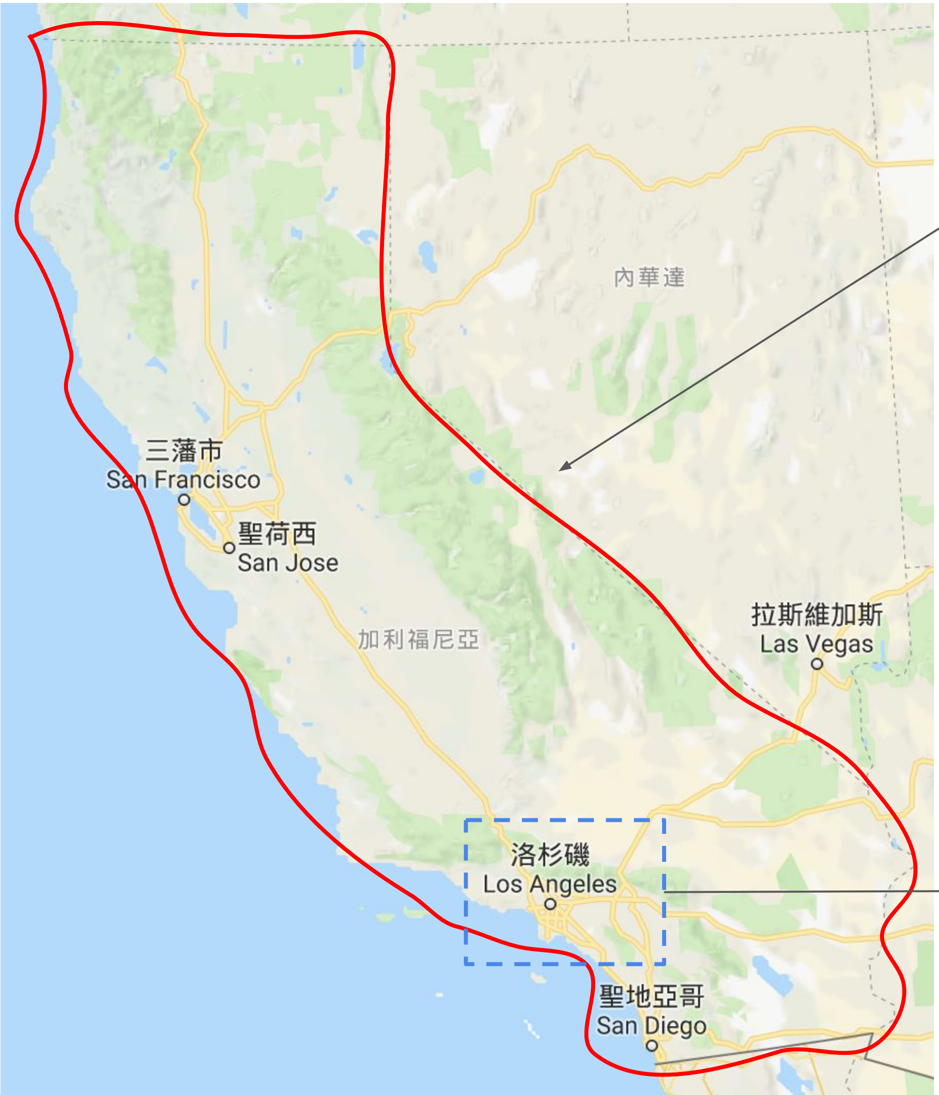
Workshop: SVM, Decision tree, Random Forest

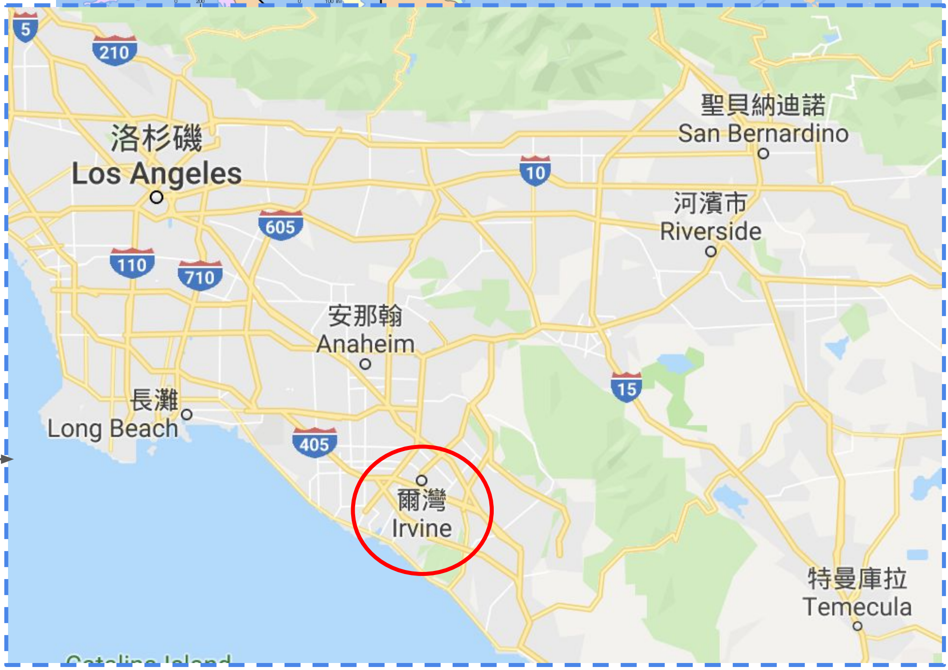
- After the lecture, we can now see how we actually use them in practice.

- In the following workshop(s), we will use Python along with scikit-learn, a simple and efficient package for machine learning and data analysis.

- We will also use dataset from the famous UC Irvine Repository.

What is UC Irvine Repository?





The Iris Dataset

- This is perhaps the best known dataset to be found in the pattern recognition literature. Compiled in 1936 by Ronald Fisher, this dataset remains a classic in the field and is referenced frequently to this day.
- The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are NOT linearly separable from each other.

Iris Setosa

Iris Versicolor

Iris Virginica

What are the attributes?

- The iris dataset possesses the following attributes: **sepal length**, **sepal width**, **petal length**, **petal width**.