CS167: Machine Learning

Linear Classifiers
Perceptron

Wednesday, March 25th, 2026

Today's Agenda

- Structure of various machine learning models

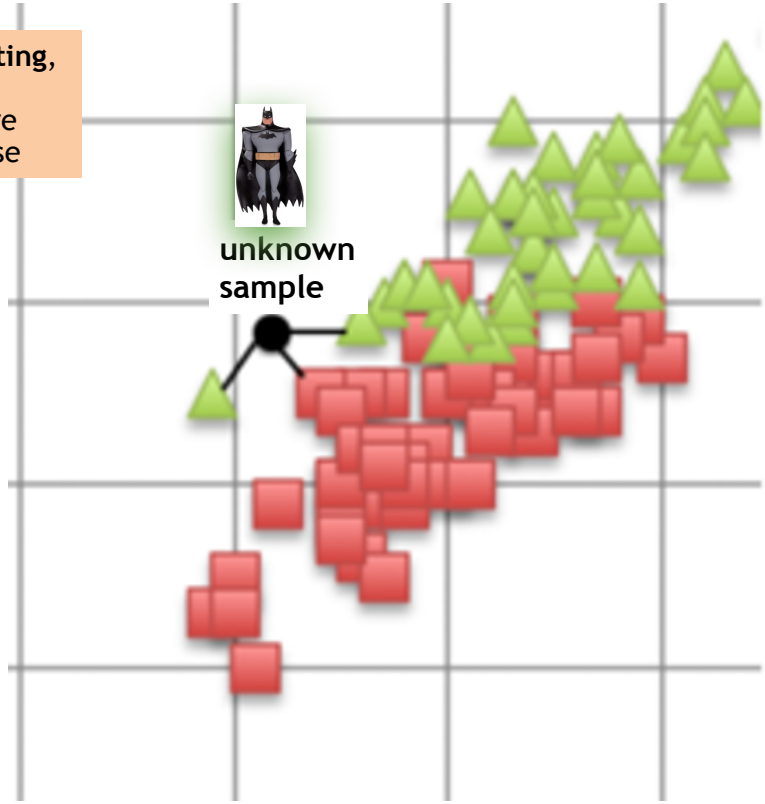
Structures of Various ML Approaches

- k-Nearest-Neighbor (kNN)
- Weighted k-Nearest Neighbor (weighted kNN)
- Decision Tree
- Random Forest

ML Approach#1: kNN

- **k-Nearest-Neighbor (kNN):** predict the *most commonly appearing* class among the k closest training examples

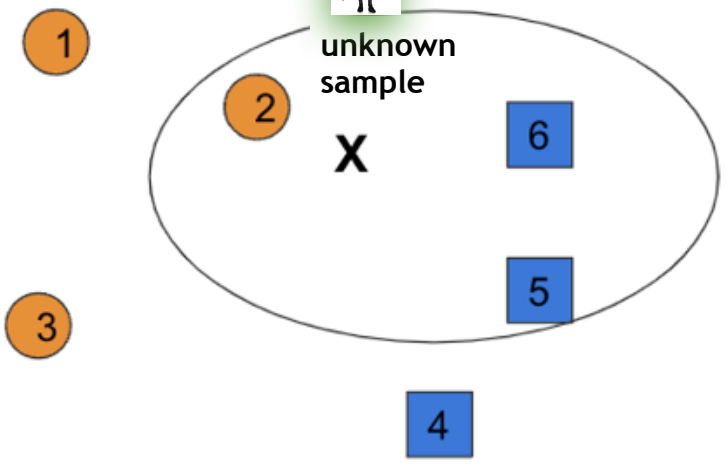
During **both steps training** and **testing**, we keep the training samples (like kNN). We don't create any structure like Tree or Forest or Something else



ML Approach#2: weighted kNN

- **Weighted kNN:** predict the target variable with the most weight (among the k closest training examples), where the weight is defined by the inverse distance function

During **both steps training** and **testing**, we keep the training samples (like kNN). We don't create any structure like Tree or Forest or Something else



$$w_{q,i} = \frac{1}{d(x_q, x_i)^2}$$

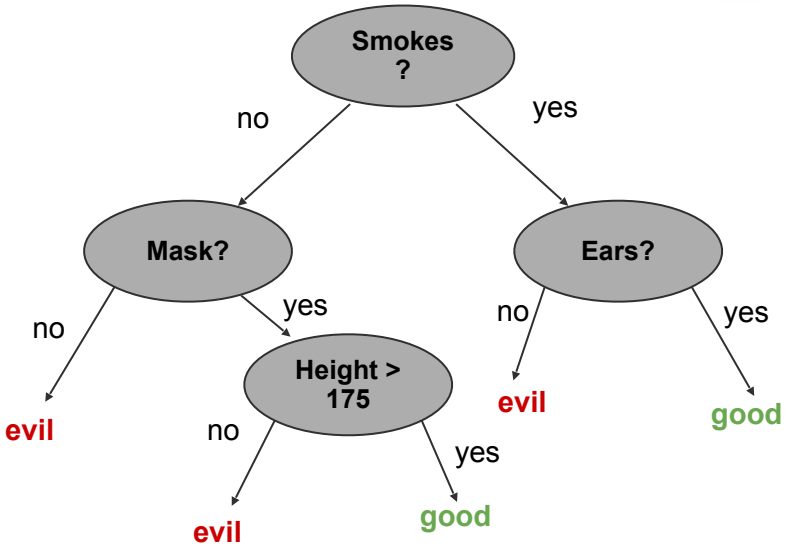
Example #	Distance	Weight
1	5	1/25
2	**1**	**1**
3	7	1/49
4	5	1/25
5	**4**	**1/16**
6	**3**	**1/9**

ML Approach#3: Decision Tree (Training Step)

- **Decision Tree:** during training step ***build the optimal tree structure*** by iteratively calculating entropies and information gains for different predictors

During **training step**, unlike kNN, we won't keep the training samples, instead calculated several entropy values and information gains to build the a **structure.** The specific structure is Decision Tree.

calculate entropy and information gain ***from these training samples*** to build the Decision Tree



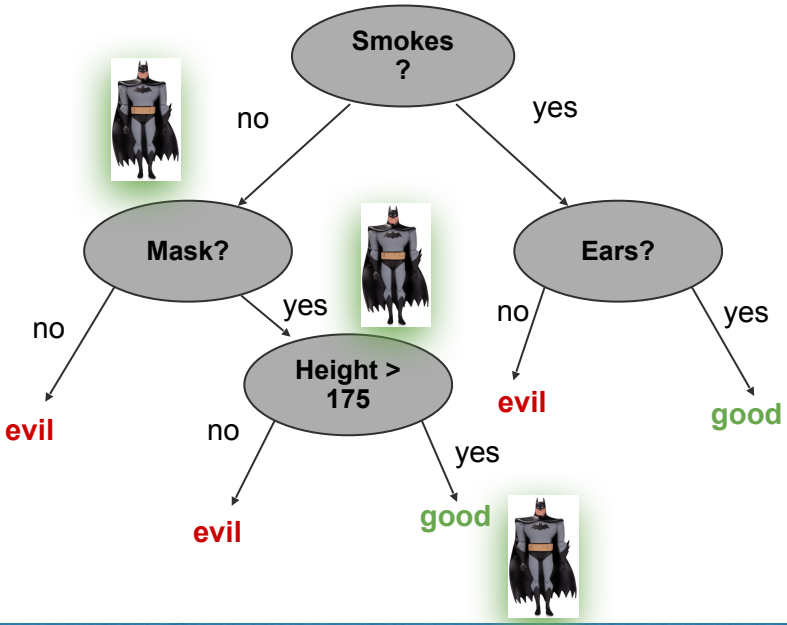
ML Approach#3: Decision Tree (Testing Step)

- **Decision Tree:** predict the target variable by determining the leaf node where the input sample lands after traversing the optimal decision tree

During **testing or inference**, we feed an unknown sample into the **Decision Tree** structure we built and determine its label from one of the leaf nodes (represented by color coded nodes)

unknown sample

	mask	cape	tie	ears	smokes	height
Batman	y	y	n	y	n	180



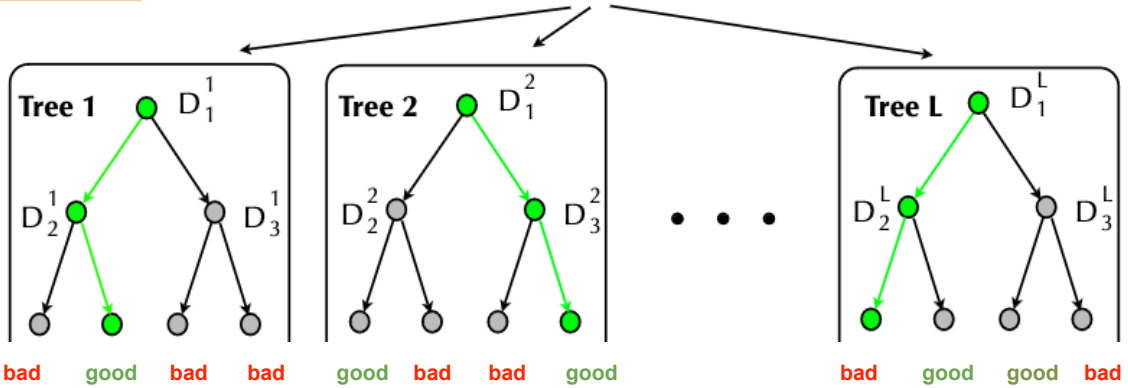
Since the unknown sample landed on a leaf node labeled 'Good,' the decision tree's prediction should be 'Good.'

ML Approach#4: Random Forest (Training Step)

- **Random Forest:** during training step ***build the optimal random forest structure*** (more than one Decision Tree) by iteratively calculating entropies and information gains for different predictors

During **training step**, unlike kNN, we won't keep the training samples, instead calculated several entropy values and information gains to build the **several tree structures.** This collection of trees is called Random Forest

Build several decision trees ***from these training samples***



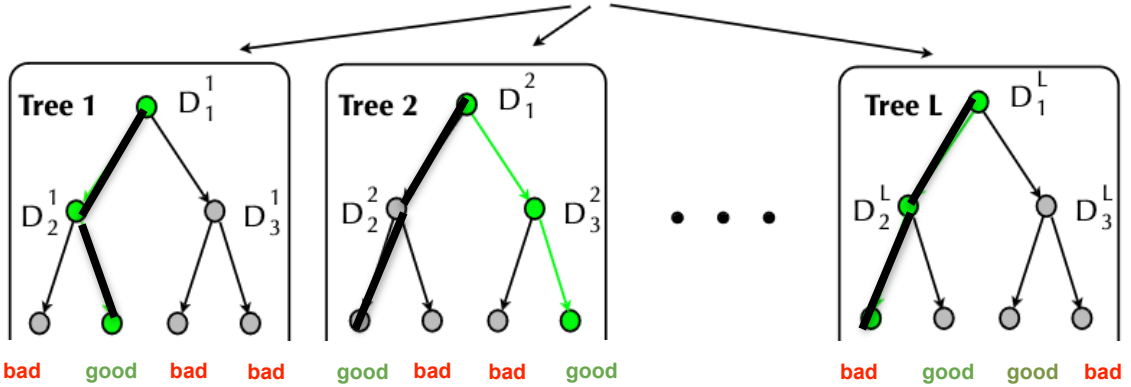
ML Approach#4: Random Forest (Training Step)

- **Random Forest:** predict the target variable by determining the leaf node where the input sample lands after traversing the optimal decision tree

During **testing or inference**, we feed an unknown sample into the **Random Forest** structure we built. We determine its label from one of the leaf nodes from each decision tree. Each outcome is aggregated to determine the final prediction

unknown sample

	mask	cape	tie	ears	smokes	height
Batman	y	y	n	y	n	180



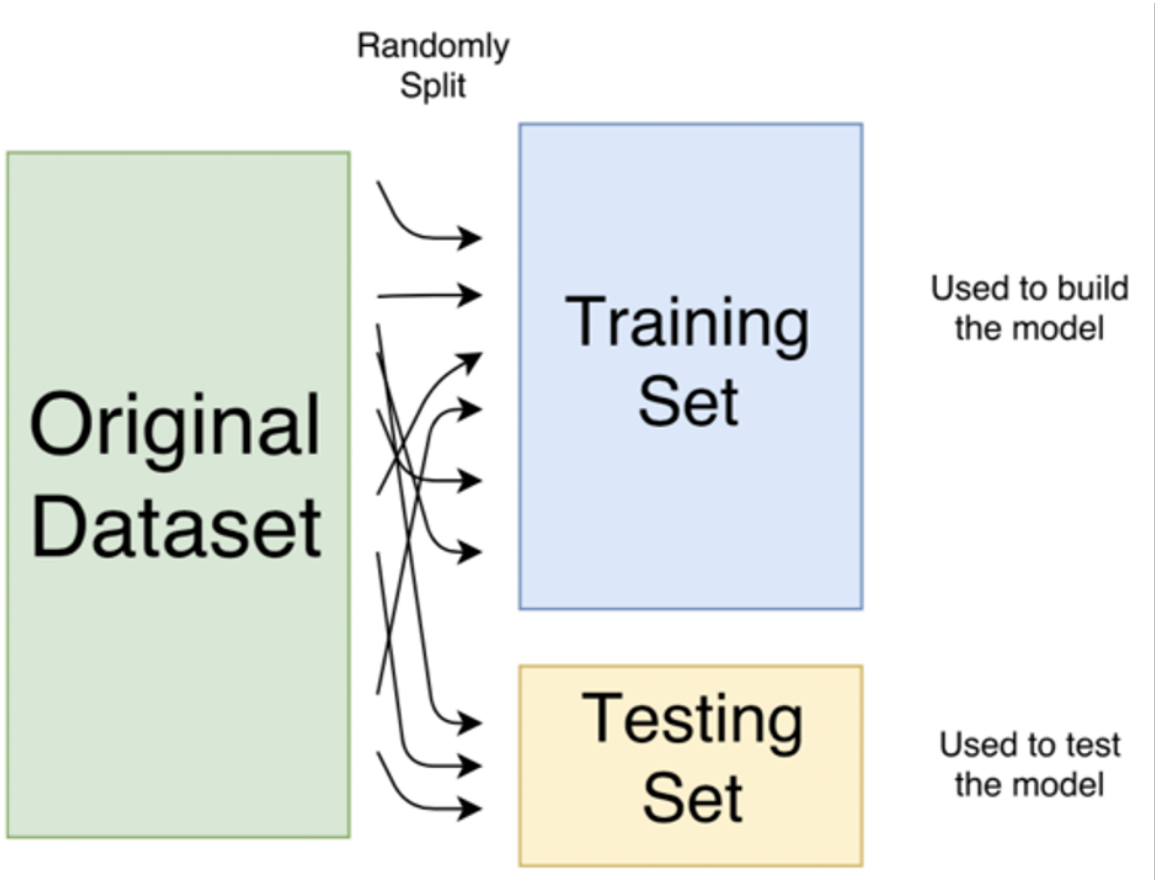
good good bad



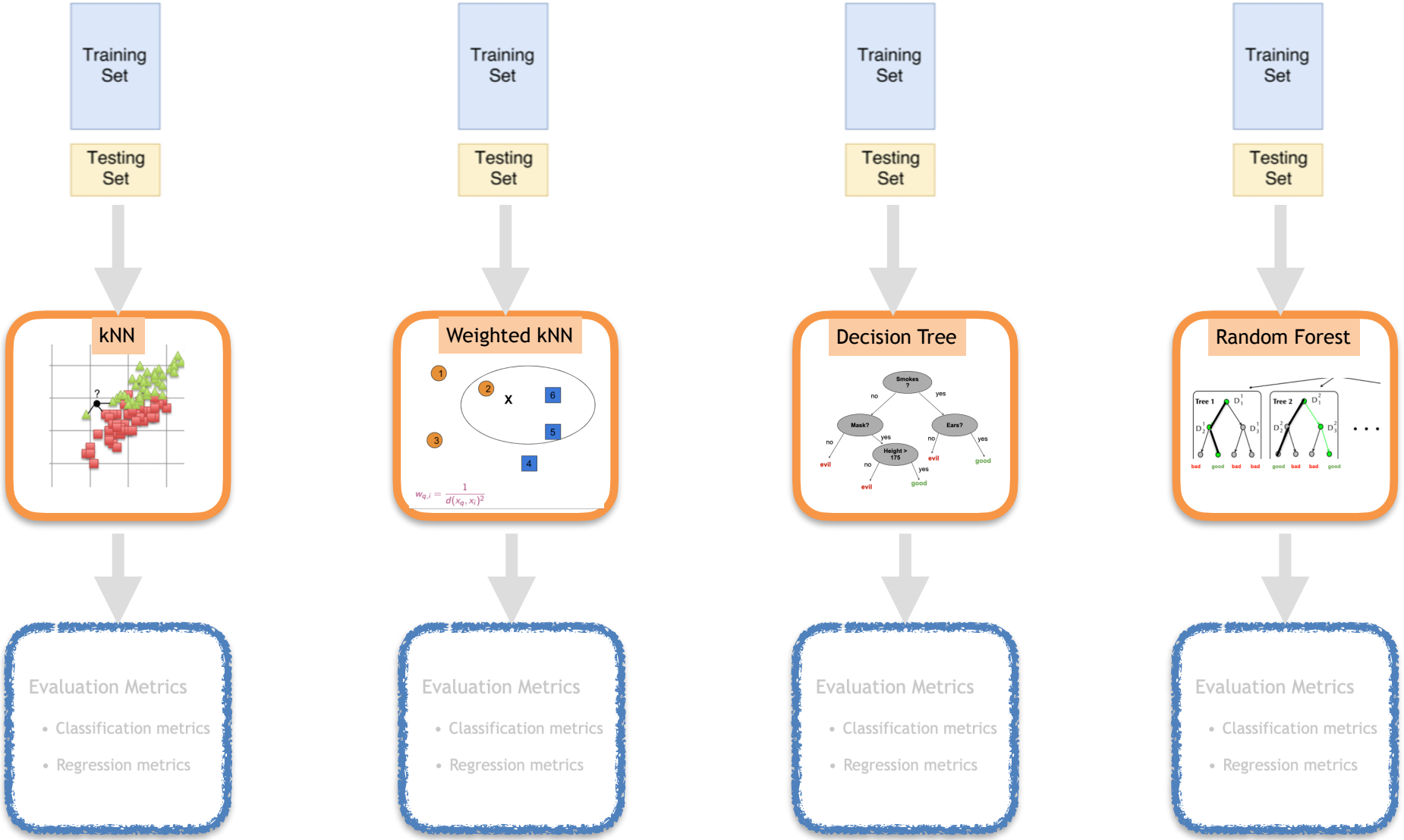
Since the unknown sample landed on a leaf node labeled 'Good,' 2 times and on a 'bad' leaf node 1 time; Random Forest prediction should be 'Good'

Cross Validation from Your Dataset

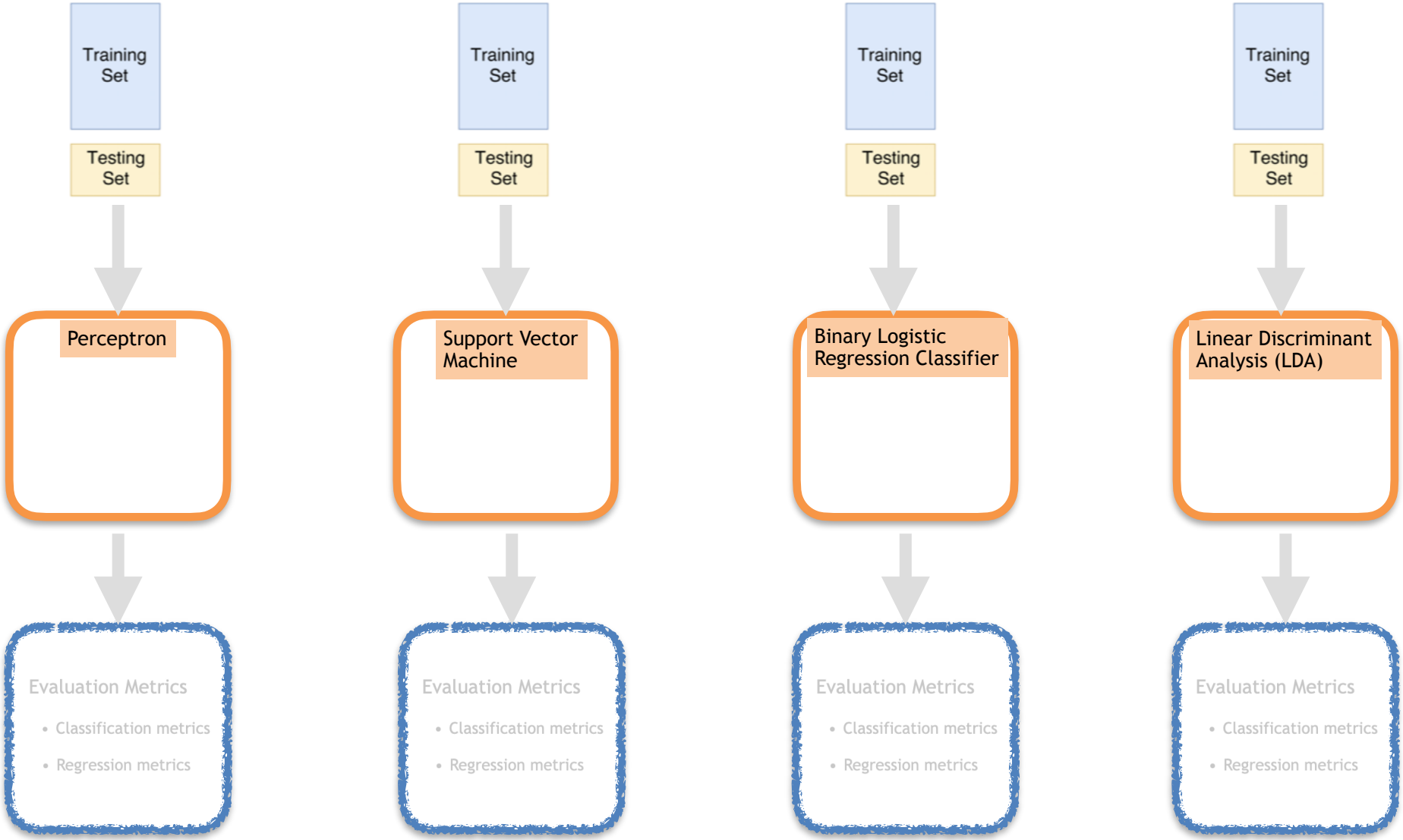
- Create a training and test split from the given dataset **only once**. Use **this same split to build** and **evaluate** various machine learning approaches to determine which one works best for the dataset



Generic ML Pipeline

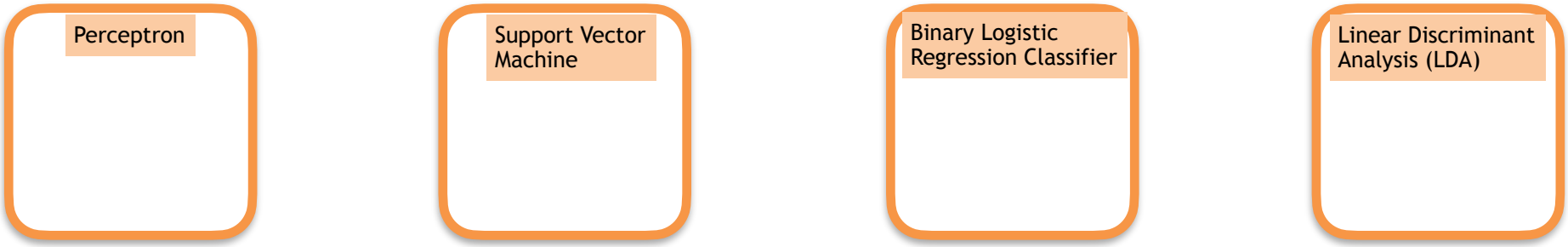


ML Pipeline with Other Approaches



Linear Classifiers

- These are all examples of linear classifiers



Unlike **Decision Tree** structure and **Random Forest** structure, the geometric structure off these classifiers will be **hyperplanes**

Today's Agenda

- Structure of various machine learning models
- Linear classifiers

Linear Classifiers

- Let's say an input sample will be represented by variables (x_0, x_1, x_2, …, x_n)
- Hence, its a data point in an *n-dimensional space*

- Then, **a linear classifier** can be expressed by the following equation:

$$w_0*x_0+w_1*x_1+\ldots+w_n*x_n = 0$$

- It means a linear classifier can be expressed by a set of weight values

(w_0, w_1,w_2, …, w_n)

Linear Classifiers in 2D Space

- Let's simplify and consider an input sample will be represented by variables (x_0, x_1). Hence, its a data point in an 2-*dimensional space*

- Then, **a linear classifier** can be expressed by the following equation:

$$w_0 * x_0 + w_1 * x_1 = 0$$

In 2D space, this equation represents a **geometric structure** known as line

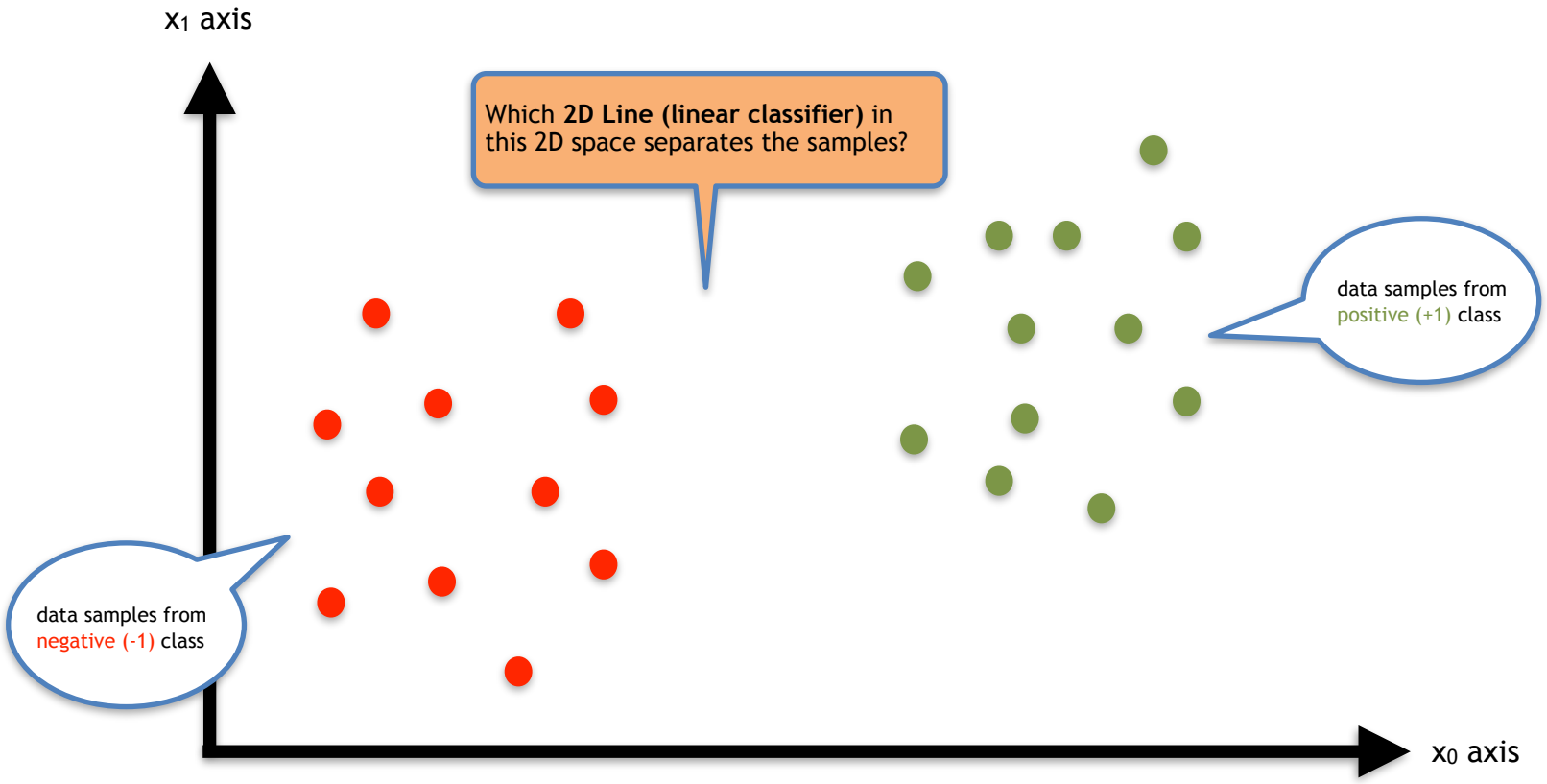
- It means a linear classifier can be expressed by a set of weight values

(w_0, w_1)

Linear Classifiers in 2D Space

- Let's simplify and consider an input sample will be represented by variables (x_0, x_1). Hence, its a data point in an 2-*dimensional space*
- Then, **a linear classifier** can be expressed by the following equation:

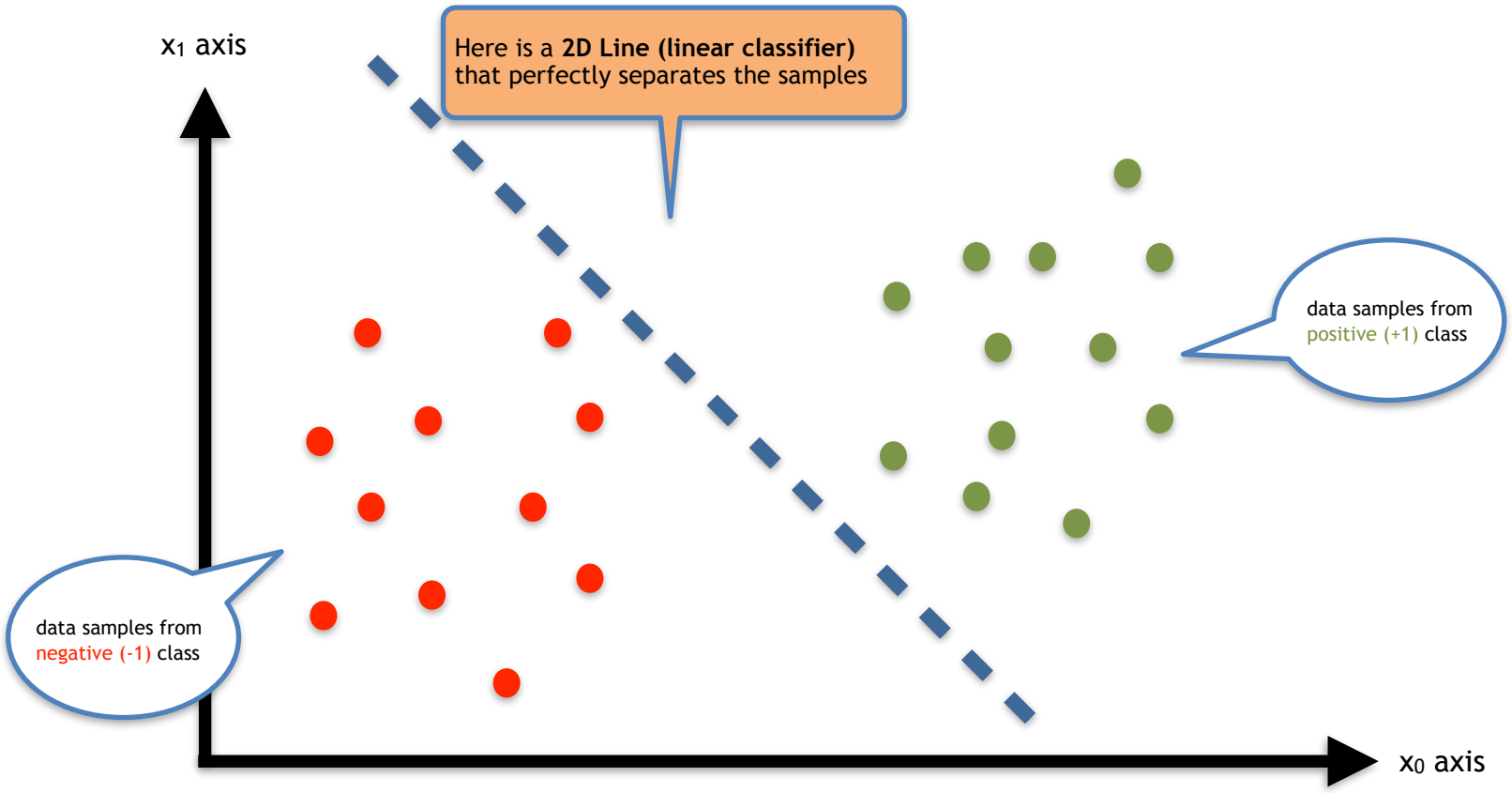
$$w_0 * x_0 + w_1 * x_1 = 0$$



Linear Classifiers in 2D Space

- Let's simplify and consider an input sample will be represented by variables (x_0, x_1). Hence, its a data point in an 2-*dimensional space*
- Then, **a linear classifier** can be expressed by the following equation:

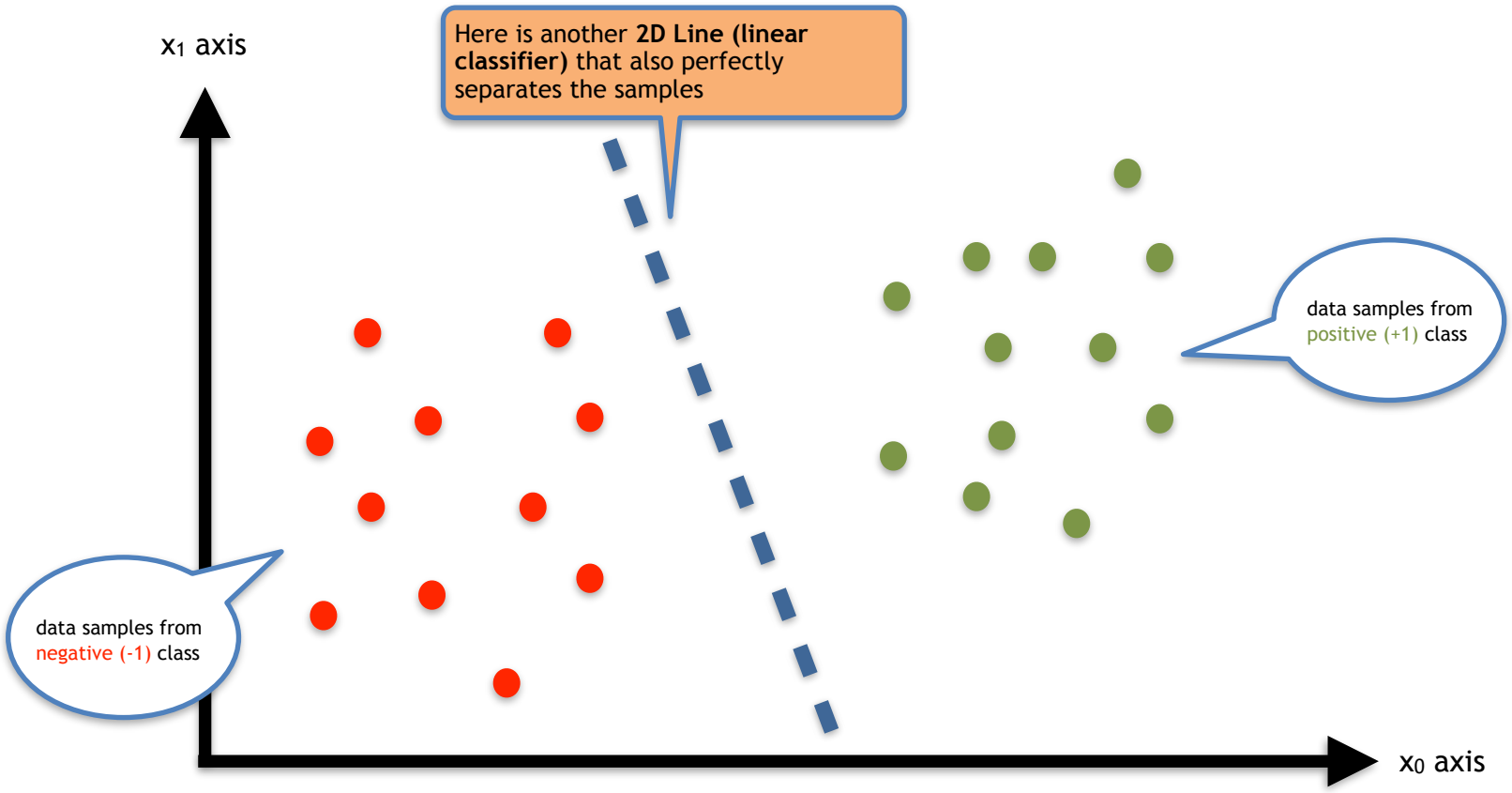
$$w_0 * x_0 + w_1 * x_1 = 0$$



Linear Classifiers in 2D Space

- Let's simplify and consider an input sample will be represented by variables (x_0, x_1). Hence, its a data point in an 2-*dimensional space*
- Then, **a linear classifier** can be expressed by the following equation:

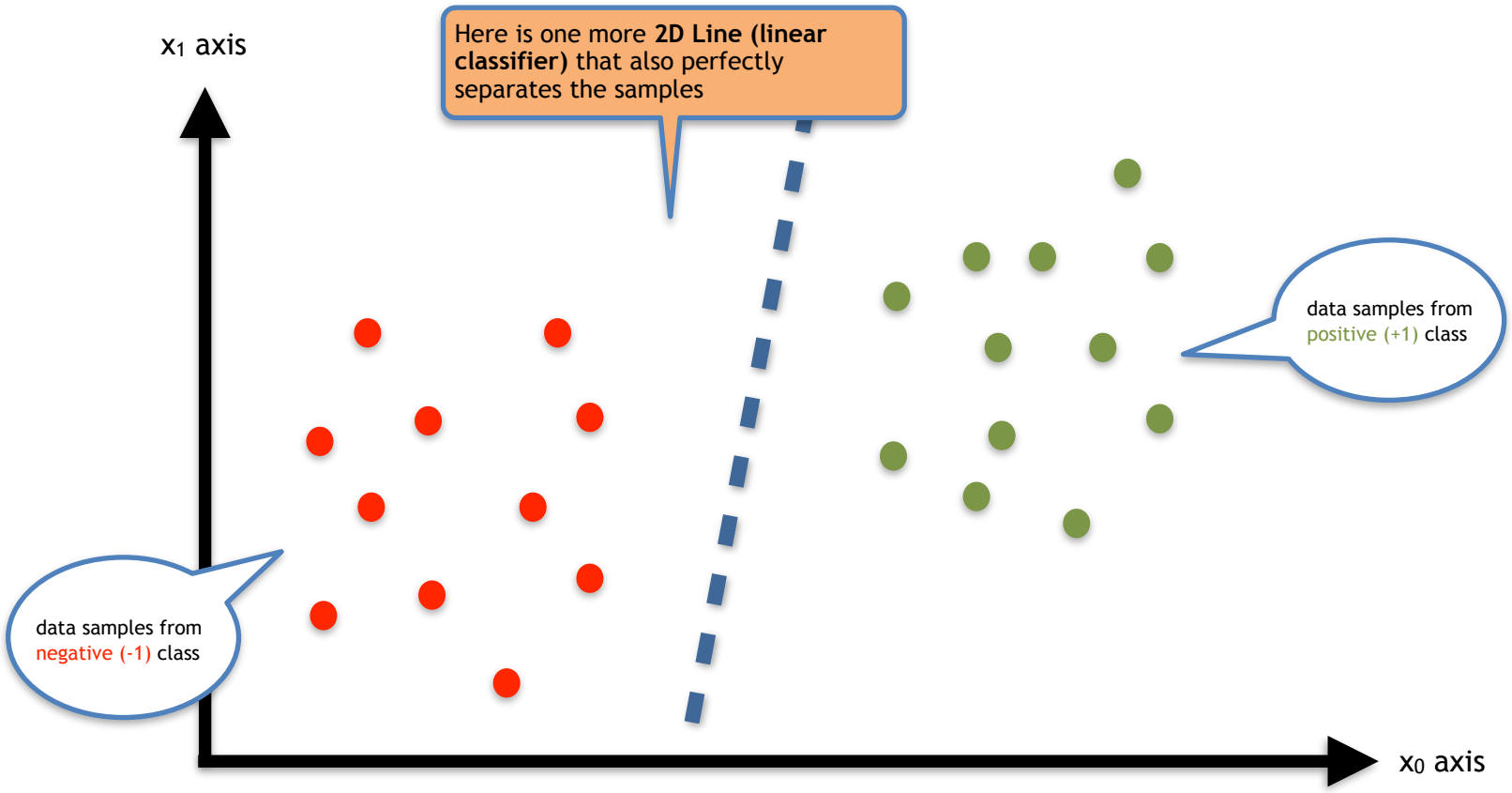
$$w_0 * x_0 + w_1 * x_1 = 0$$



Linear Classifiers in 2D Space

- Let's simplify and consider an input sample will be represented by variables (x_0, x_1). Hence, its a data point in an 2-*dimensional space*
- Then, **a linear classifier** can be expressed by the following equation:

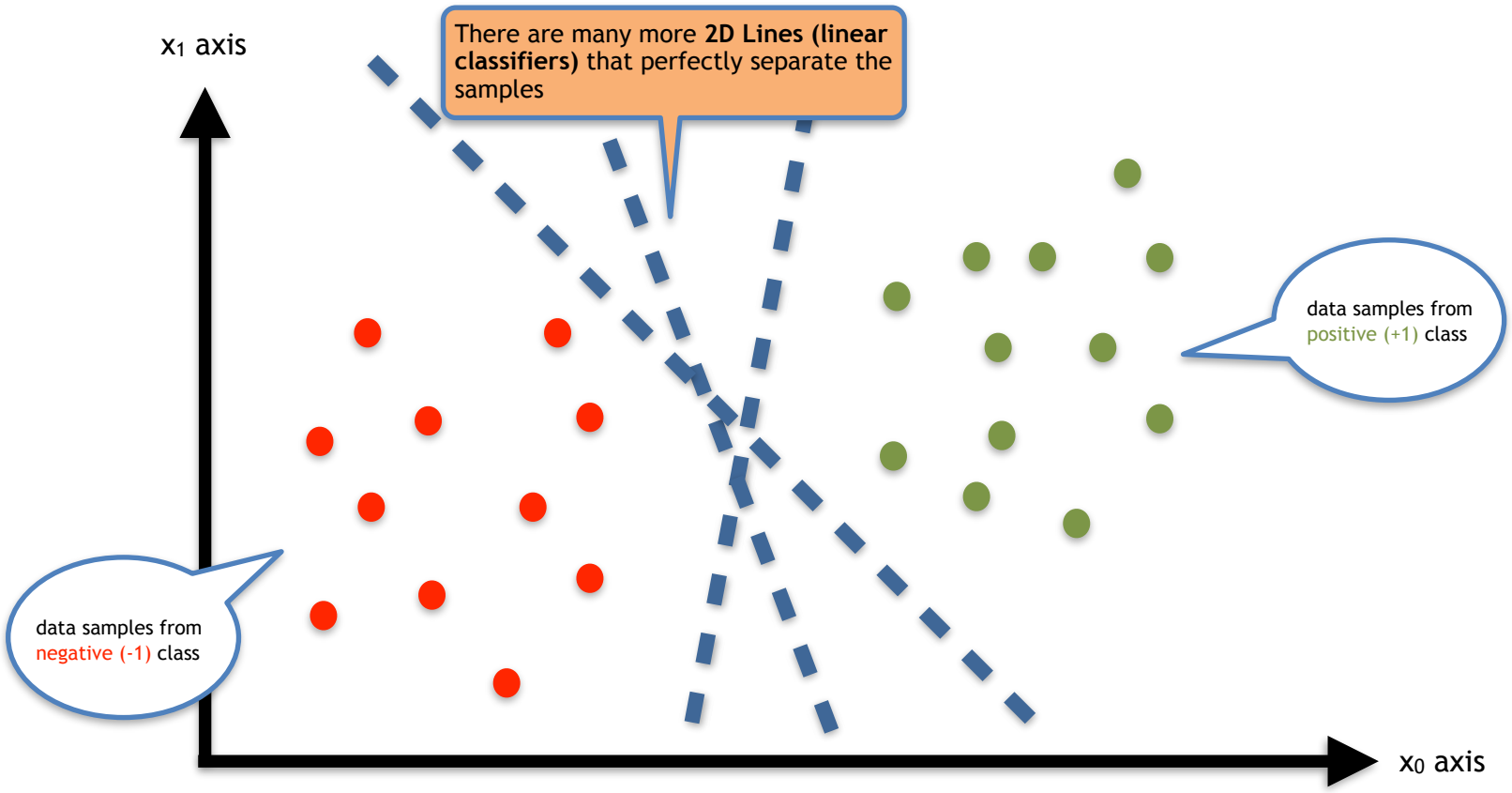
$$w_0 * x_0 + w_1 * x_1 = 0$$



Linear Classifiers in 2D Space

- Let's simplify and consider an input sample will be represented by variables (x_0, x_1). Hence, its a data point in an 2-*dimensional space*
- Then, **a linear classifier** can be expressed by the following equation:

$$w_0 * x_0 + w_1 * x_1 = 0$$



Linear Classifiers in 3D Space

- Let's simplify and consider an input sample will be represented by variables (x_0, x_1, x_2). Hence, its a data point in an 3-*dimensional space*

- Then, **a linear classifier** can be expressed by the following equation:

$$w_0*x_0+w_1*x_1+w_2*x_2 = 0$$

In 3D space, this equation represents a **geometric structure** known as plane

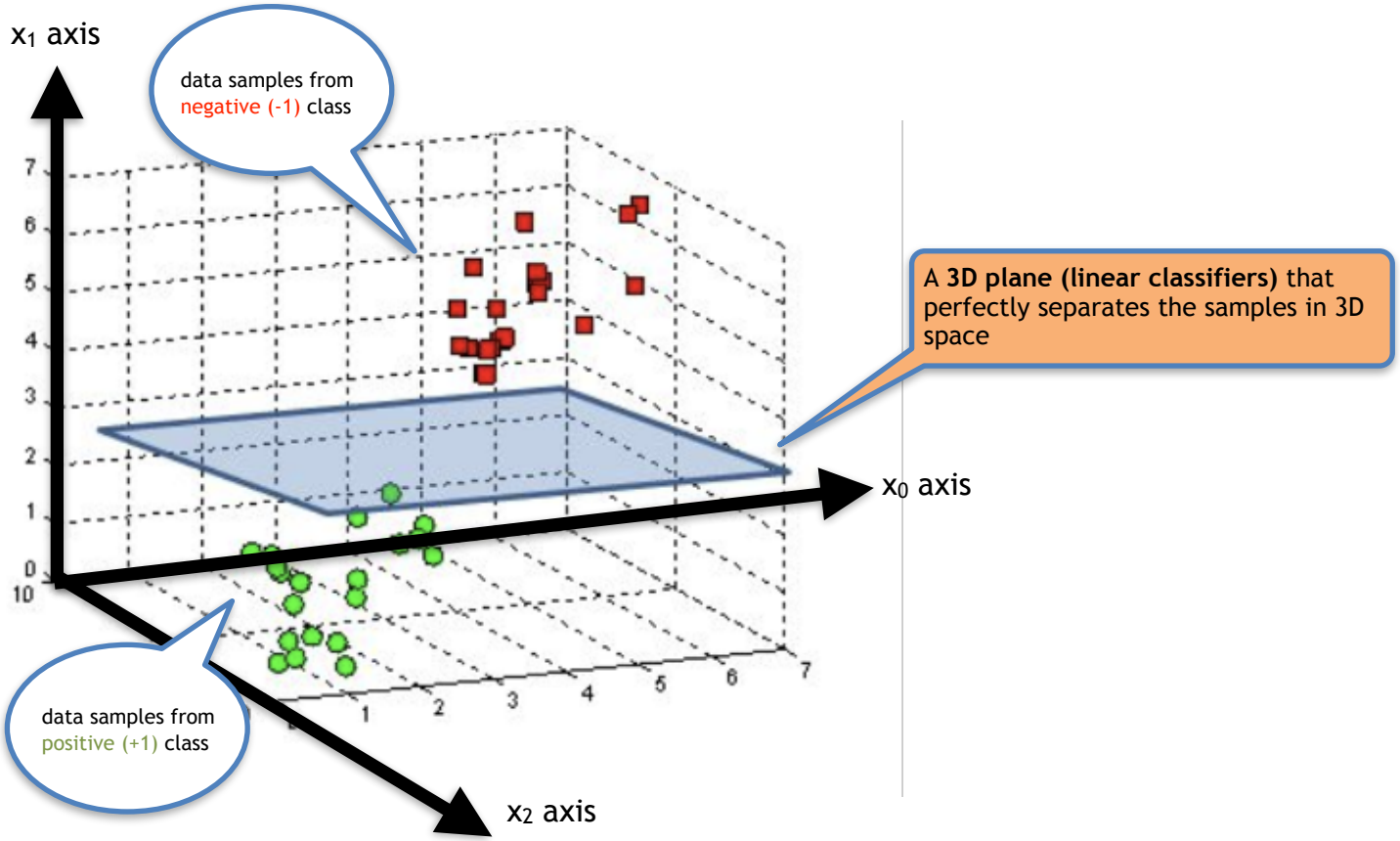
- It means a linear classifier can be expressed by a set of weight values

(w_0, w_1, w_2)

Linear Classifiers in 3D Space

- Let's simplify and consider an input sample will be represented by variables (x_0, x_1, x_2). Hence, its a data point in an 3-*dimensional space*
- Then, **a linear classifier** can be expressed by the following equation:

$$w_0*x_0+w_1*x_1+w_2*x_2 = 0$$



Linear Classifiers in *n-D Space*

- In an *n-dimensional space*, the following equation represents a linear classifier: $w_0+w_1*x_1+\ldots+w_n*x_n = 0$

This equation represents a **geometric structure** known as hyperplane

- The linear classifier can be expressed by a set of weight values

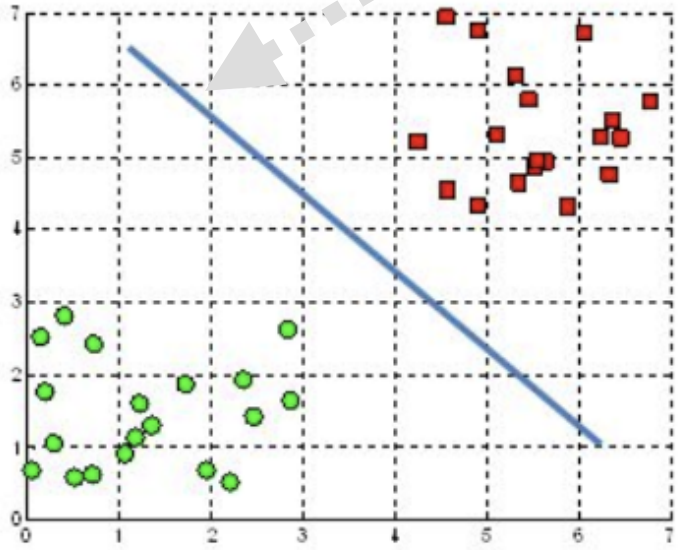
$(w_0, w_1, w_2, \ldots, w_n)$

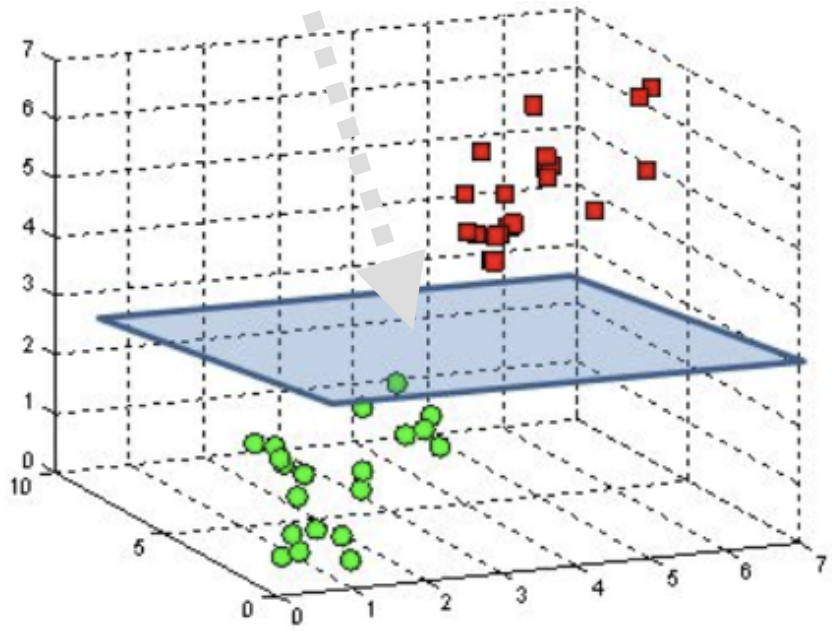
Let's assume we are solving a binary classification problem

- Data separation or decision rule:
  - Use: $w_0+w_1*x_1+\ldots+w_n*x_n > 0$ to denote positive classifications
  - Use: $w_0+w_1*x_1+\ldots+w_n*x_n \leq 0$ to denote negative classifications

Separation by Hyperplanes

- Assume we have some data that is **linearly separable:**
  - In 2 dimensions, we can draw a 2D line to separate the data
  - In 3 dimensions, that separator becomes a 3D *plane*
  - In more than 3 dimensions, that separator becomes a n-D *hyperplane*



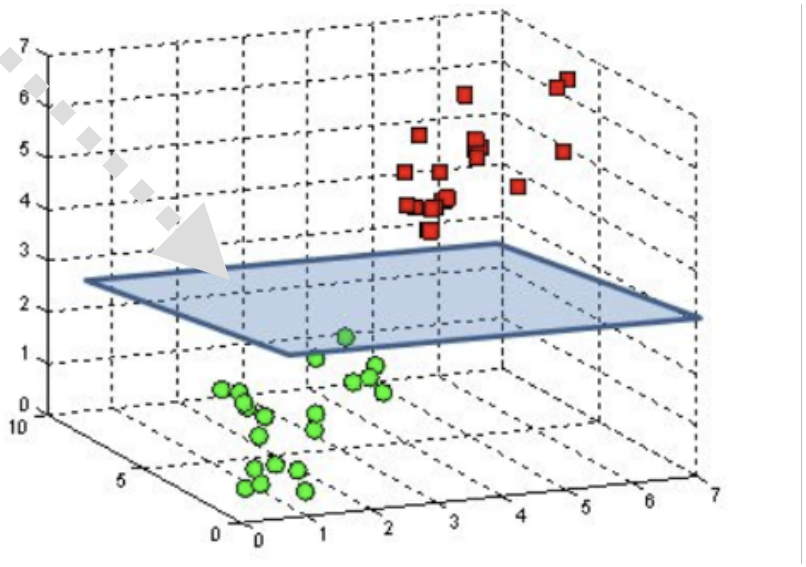


Separation by Hyperplanes

- Assume we have some data that is **linearly separable:**
- Let's say the input samples are: X_1 = [1,2,3], X_2 = [2,4,6], … X_{20} = [2,4,-6]
- The labels of the input samples are: Y_1 = +1, Y_{12} = +1, …, Y_{20} = -1

$$W \cdot X_t + w_o \geq +1 \text{ for all the positive class samples}$$

$$W \cdot X_t + w_o \leq -1 \text{ for all the negative class samples}$$



New Linear Classifier: Perceptron

- Let's build a mathematical model consisting of a *single neuron* is called a **perceptron** expressed by a function $f(x,w)$ with two components applied in a sequential manner:
  - Component 1: a linear model of the form (it is a hyperplane which we just discussed):

$$w_0*x_0+w_1*x_1+\ldots+w_n*x_n$$

  - Component 2: **a step function o()** which will
    - produce +1 if the function value is positive or
    - produce -1 if the function value is negative