Université Constantine 2
جامعة قسنطينة 2

Module : Machine Learning (ML – SDSI)

– Course 2 –

Chapter 2 : Supervised Learning for Classification

Habib-Ellah GUERGOUR
Faculty of NTIC / TLSI Department
Contact: habib.guergour@univ-constantine2.dz

Université Constantine 2
جامعة قسنطينة 2

Module : Machine Learning (ML – SDSI)

– Course 2 –

Chapter 2 : Supervised Learning for Classification

Habib-Ellah GUERGOUR
Faculty of NTIC / TLSI Department
Contact: habib.guergour@univ-constantine2.dz

Etudiants concernés

Faculté/Institut	Département	Niveau	Spécialité
NTIC	TLSI	M1	SDSI

Goals of the Chapter

- Understand the fundamentals of **supervised learning for classification**.
- Learn key concepts: labels, features, decision boundaries, and probabilistic vs. non-probabilistic classifiers.
- Explore common classification algorithms (**logistic regression**, **decision trees**, **SVM**, **NB**, etc.).
- Evaluate models using metrics like **accuracy**, **precision**, **recall**, **ROC-AUC…**

Main Titles

- Introduction
- Logistic Regression
- K-NN : K – Nearest Neighbors
- NB : Naïve Bayes
- DT : Decision Trees
- SVM : Support Vector Machines
- Evaluating Model Performance: **Metrics**

Introduction

Introduction to Supervised Learning for Classification

- **Classification** is a type of supervised learning where the goal is to predict discrete labels (categories) based on input features**.**
- Examples:
  - **Spam Detection**: Email is spam or not spam.
  - **Medical Diagnosis**: A tumor is benign or malignant.
  - **Sentiment Analysis**: A review is positive, neutral, or negative.

Types of Classification

- **Binary Classification**: Two possible labels (e.g., spam or not spam).
- **Multiclass Classification**: More than two categories (e.g., dog, cat, bird).
- **Multilabel Classification**: An instance can belong to multiple categories (e.g., a movie can be both action and comedy)..

Introduction to Supervised Learning for Classification

Common Classification Algorithms

- **Logistic Regression** (for binary classification).
- **k-Nearest Neighbors (k-NN)** (instance-based learning)
- **Support Vector Machines (SVM**) (effective for high-dimensional data).
- **Decision Trees** (interpretable, but prone to overfitting).
- **Random Forest** (ensemble of decision trees).
- **Neural Networks** (powerful for complex patterns)

Logistic Regression

Motivation for Logistic Regression

Why linear regression is not well-suitable?

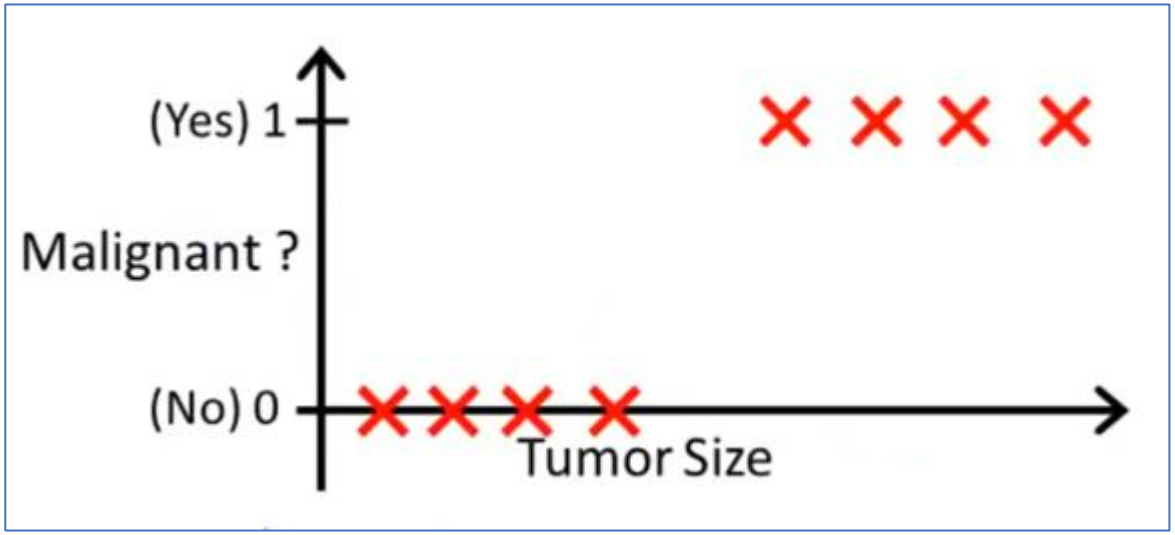
Example 1: Tumor Detection

- Given a tumor's size, we want to classify it as Benign (0) or Malignant (1).
- The output is binary (0 or 1), not continuous.

Logistic Regression

Motivation for Logistic Regression

Example 1: Tumor Detection

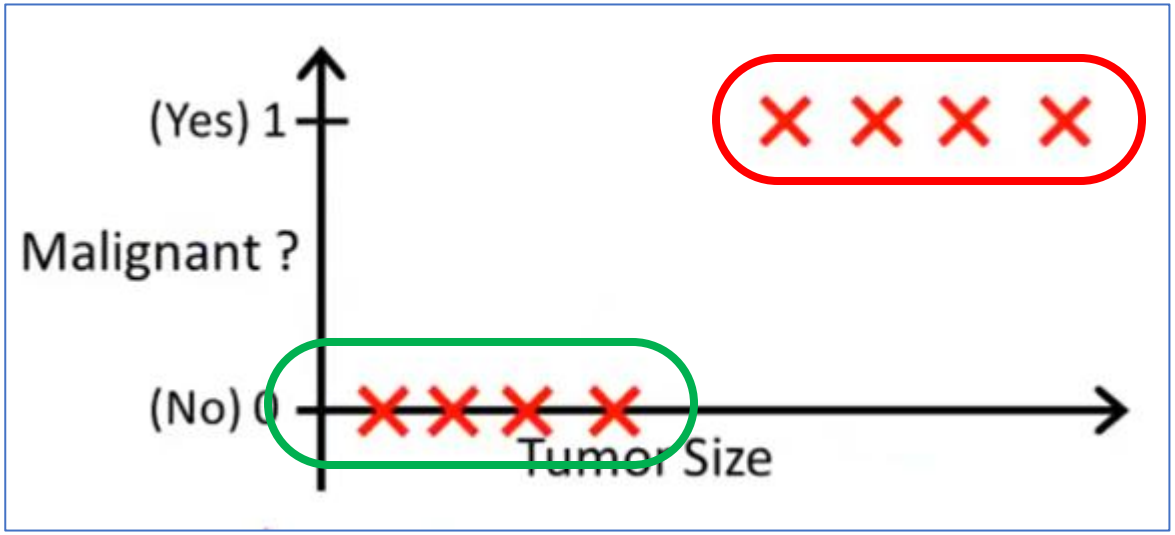


Logistic Regression

Motivation for Logistic Regression

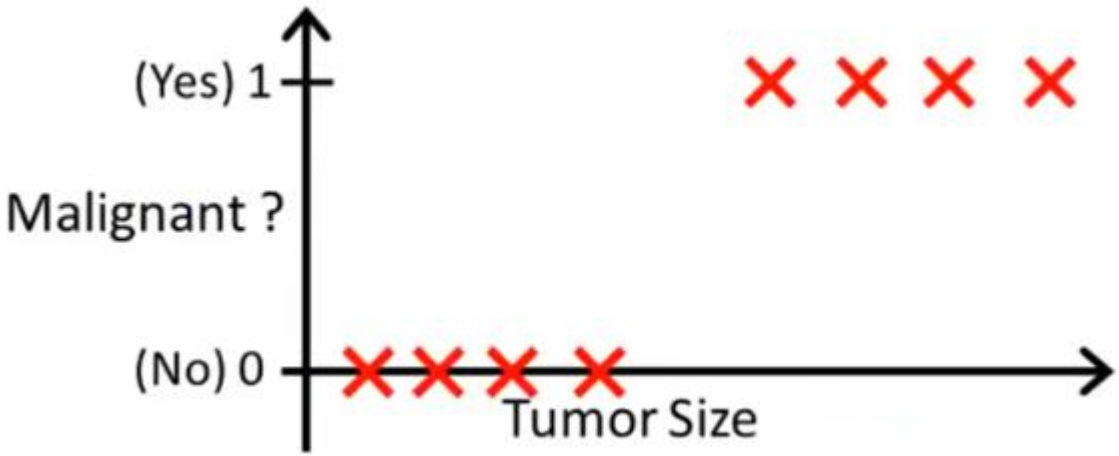
Example 1: Tumor Detection

- **Can linear regression be used for classification tasks**



Logistic Regression

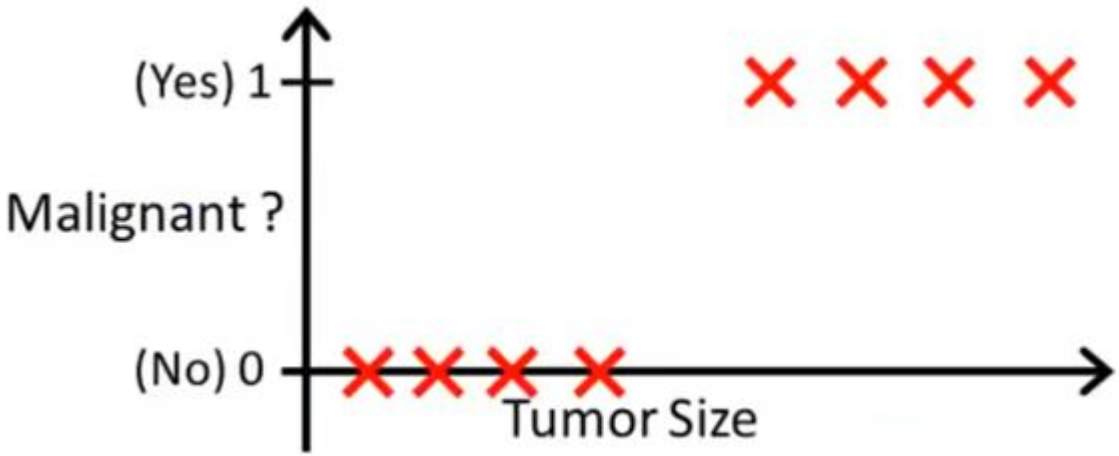
Motivation for Logistic Regression



Can linear regression be used for classification tasks

Logistic Regression

Motivation for Logistic Regression



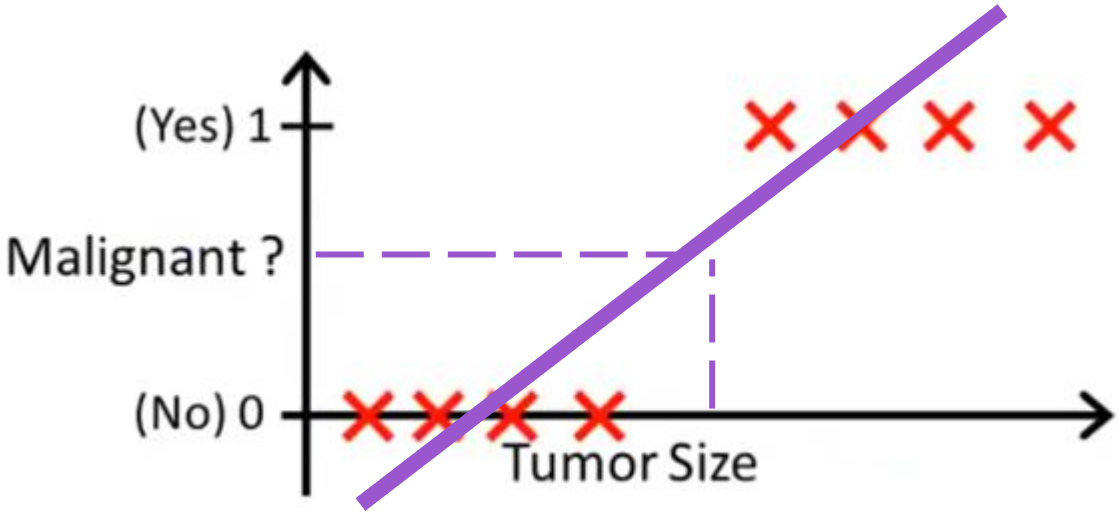
Can linear regression be used for classification tasks

A natural idea is to fit a linear model:

$$h_\theta(x) = \theta_0 + \theta_1 x$$

Logistic Regression

Motivation for Logistic Regression



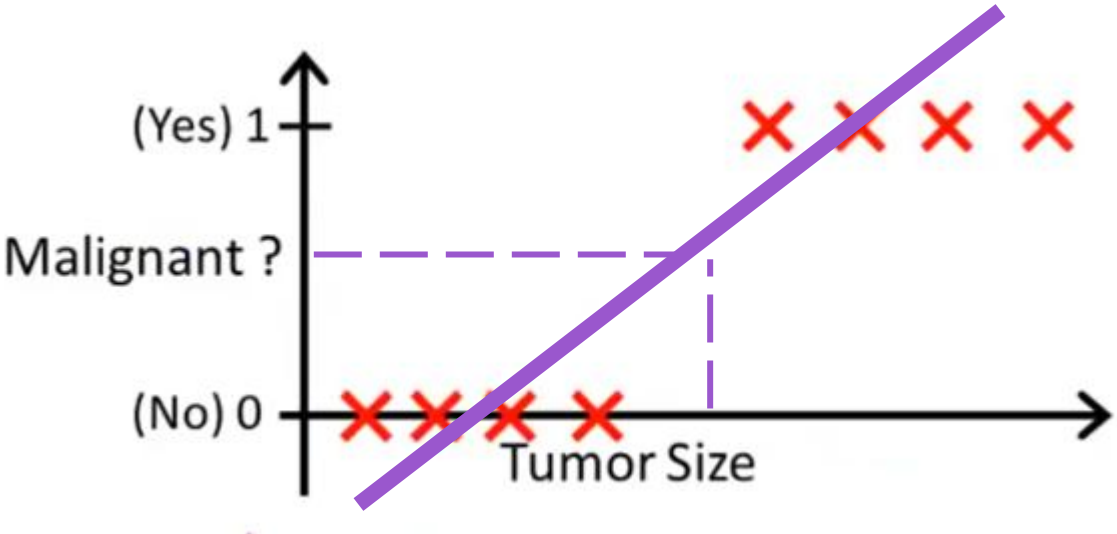
Can linear regression be used for classification tasks

A natural idea is to fit a linear model:

$$h_\theta(x) = \theta_0 + \theta_1 x$$

Logistic Regression

Motivation for Logistic Regression

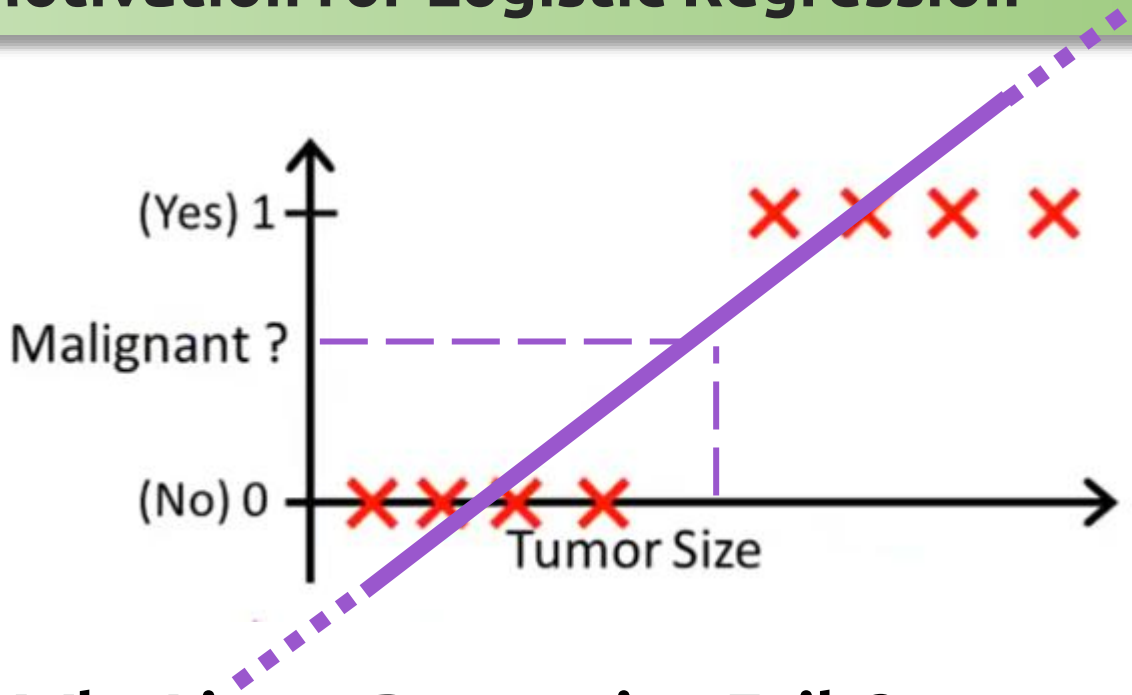


Can linear regression be used for classification tasks

Why Linear Regression Fails?

Logistic Regression

Motivation for Logistic Regression



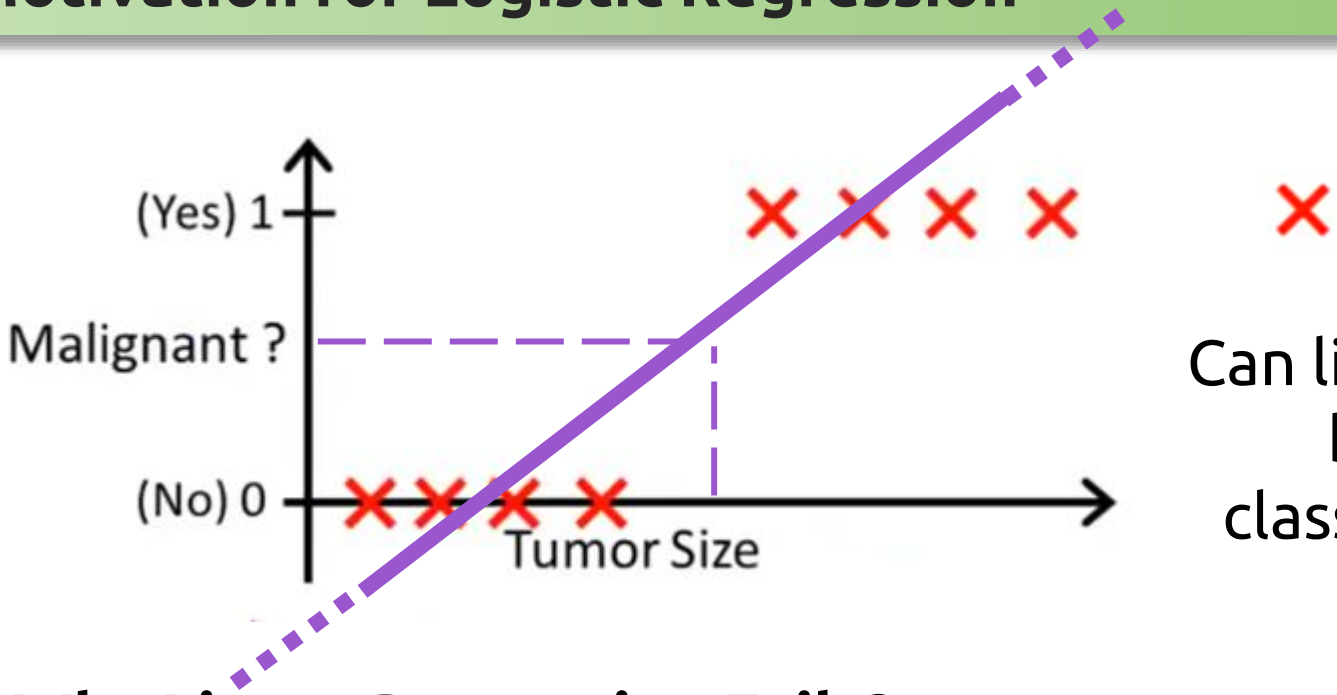
Can linear regression be used for classification tasks

Why Linear Regression Fails?

- Predictions can be outside 0 and 1

Logistic Regression

Motivation for Logistic Regression



Can linear regression be used for classification tasks

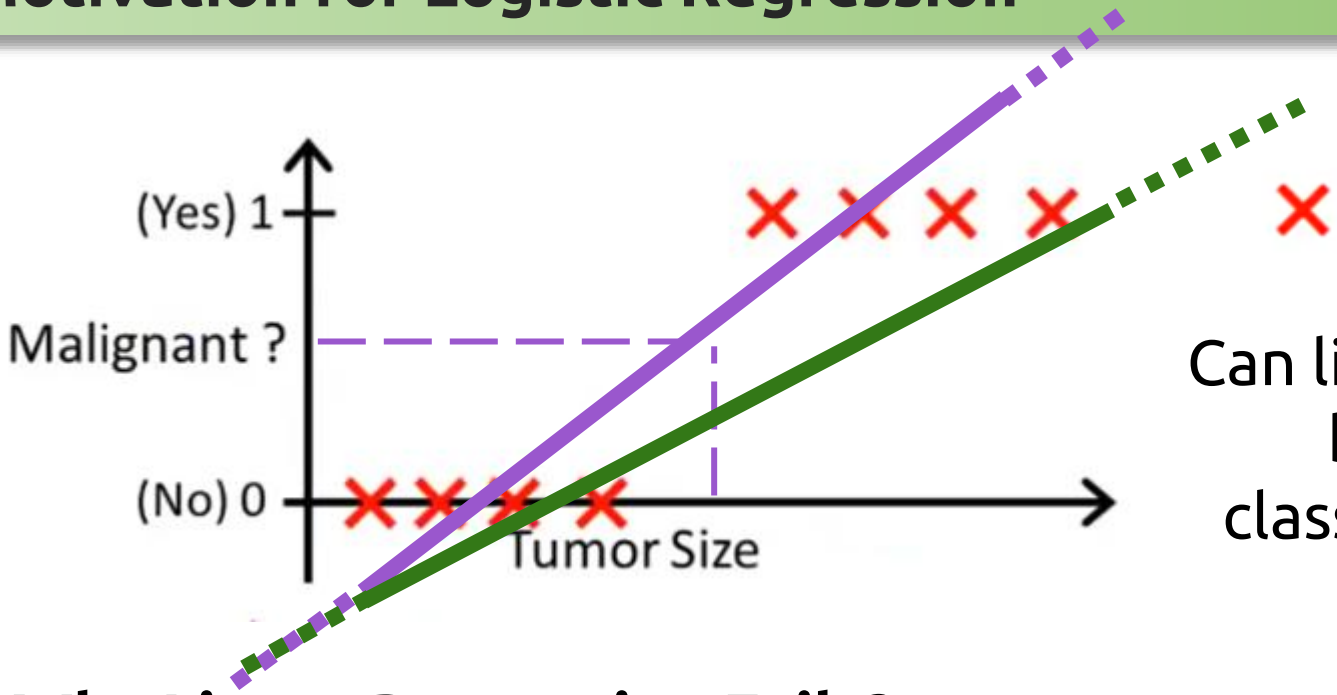
Why Linear Regression Fails?

- Predictions can be outside 0 and 1
- Poor classification boundary : If we set a threshold like $h_\theta(x) \geq 0.5$

→ the separation might not be clear (The decision boundary is sensitive to outliers)

Logistic Regression

Motivation for Logistic Regression



Can linear regression be used for classification tasks

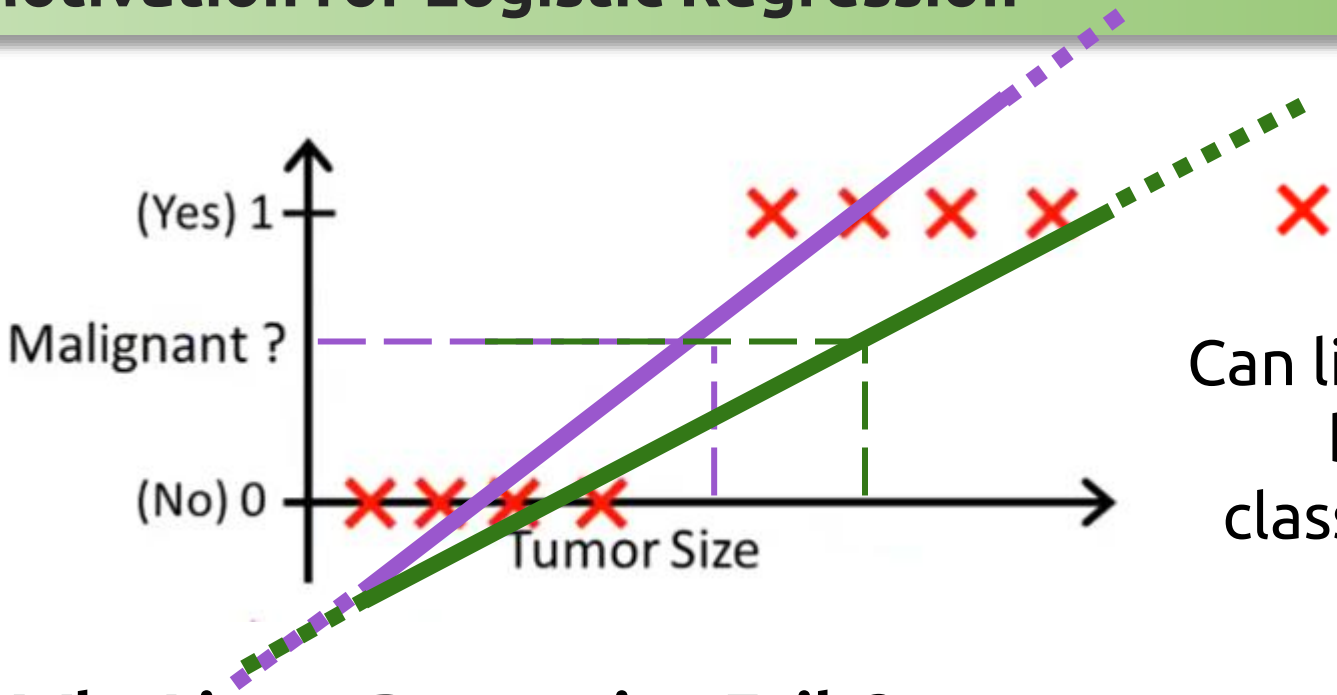
Why Linear Regression Fails?

- Predictions can be outside 0 and 1
- Poor classification boundary : If we set a threshold like $h_\theta(x) \geq 0.5$

→ the separation might not be clear (The decision boundary is sensitive to outliers)

Logistic Regression

Motivation for Logistic Regression



Can linear regression be used for classification tasks

Why Linear Regression Fails?

- Predictions can be outside 0 and 1
- Poor classification boundary : If we set a threshold like $h_\theta(x) \geq 0.5$

→ the separation might not be clear (The decision boundary is sensitive to outliers)

Need for a New Approach

- We need a function that outputs only valid class labels (0 or 1).
- We need a model that provides clearer decision boundaries between binary classes.

Motivation for Logistic Regression

→ Need for a New Approach

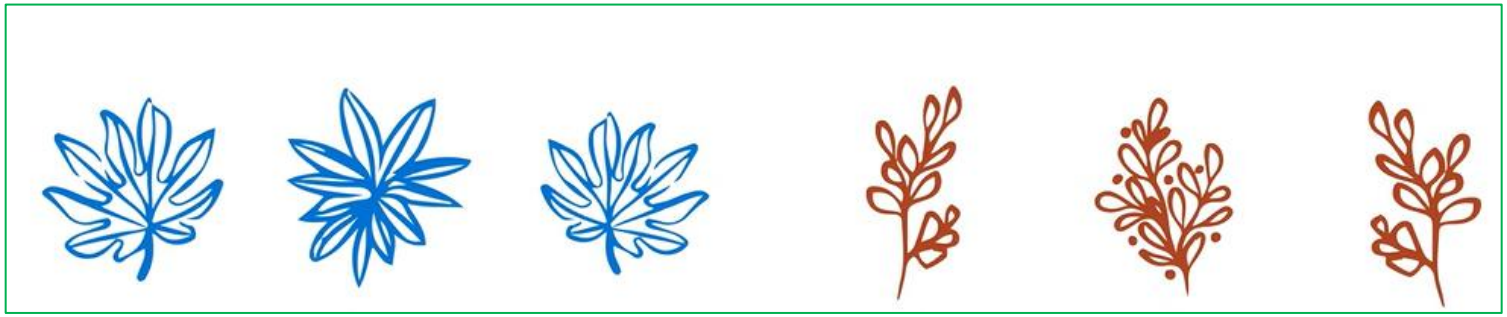
But, how ?

Example 2: Poisonous Plant Classification

Logistic Regression

Motivation for Logistic Regression

- **Example 2: Poisonous Plant Classification**
- Given features such as leaf shape, we classify a plant as Safe (0) or Poisonous (1).
- The output is binary (0 or 1), not continuous.

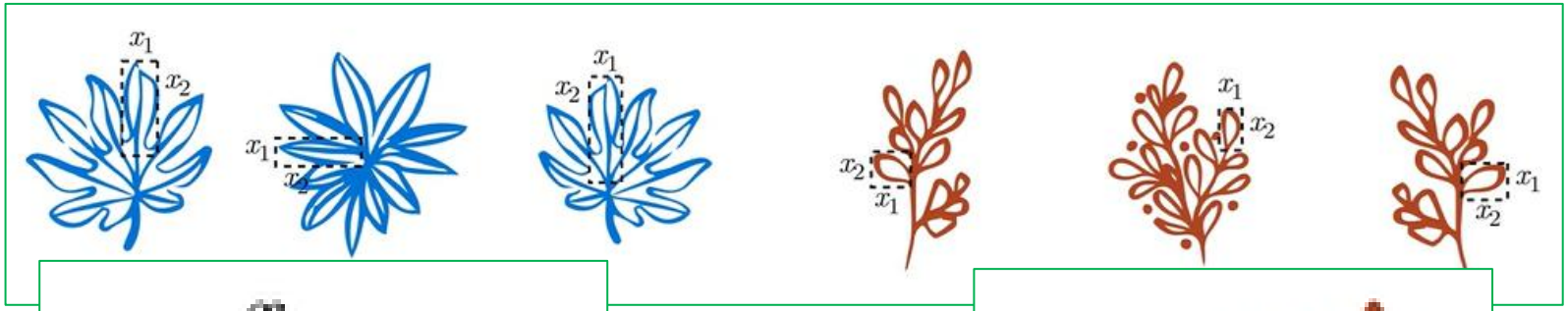


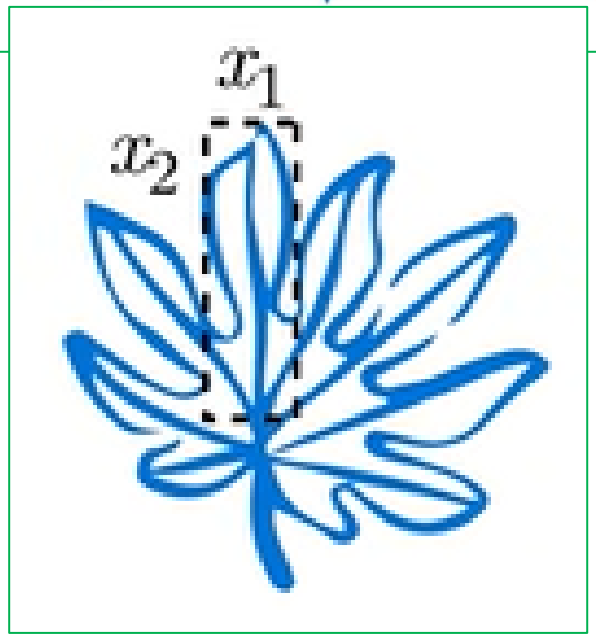
Poisonous (y = 1) Safe (y = 0)

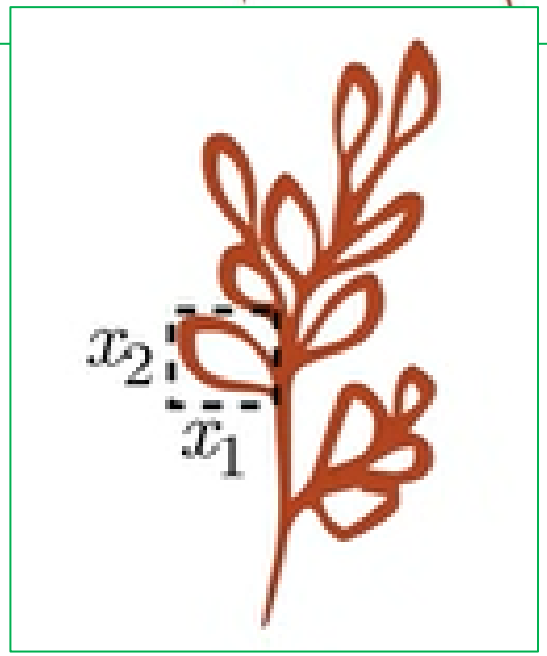
Logistic Regression

Motivation for Logistic Regression

- **Example 2: Poisonous Plant Classification**







Logistic Regression

Motivation for Logistic Regression

- **Example 2: Poisonous Plant Classification**

x_1 : largeur de la feuille
x_2 : longueur de la feuille

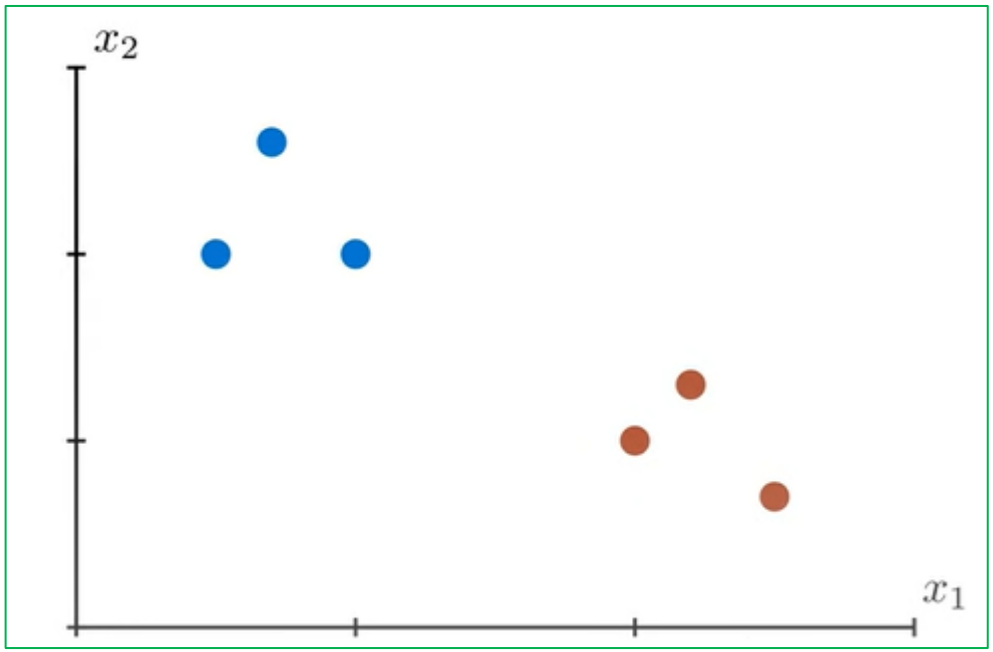
y	x_1	x_2
1	0.5	2.0
1	1.1	2.1
1	0.7	2.6
0	2.0	1.0
0	2.5	0.7
0	2.2	0.3

Motivation for Logistic Regression

- **Example 2: Poisonous Plant Classification**

x_1 : largeur de la feuille
x_2 : longueur de la feuille

y	x_1	x_2
1	0.5	2.0
1	1.1	2.1
1	0.7	2.6
0	2.0	1.0
0	2.5	0.7
0	2.2	0.3

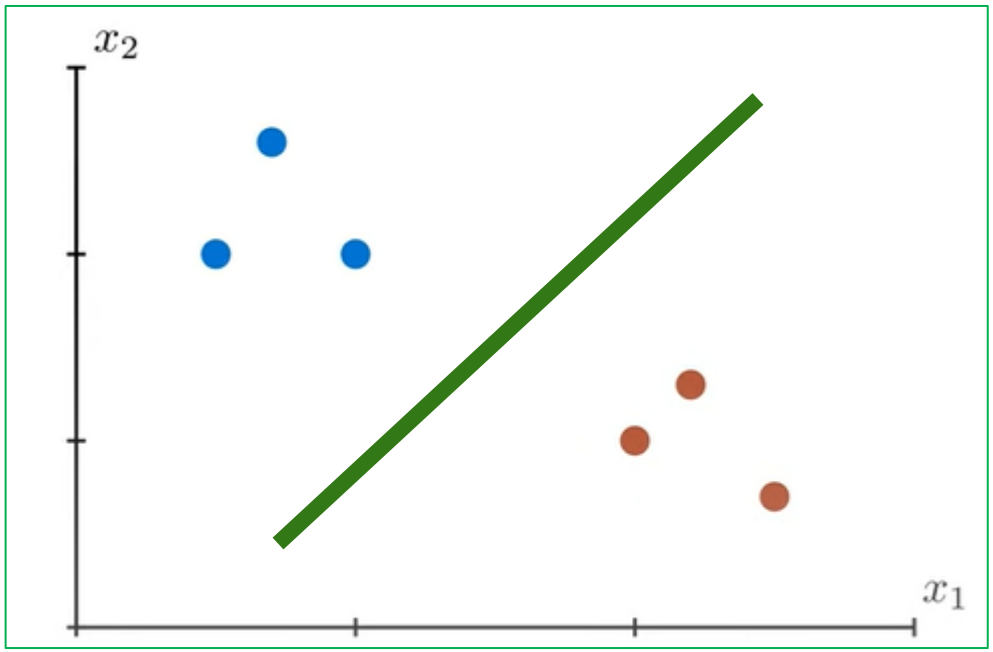


Motivation for Logistic Regression

- **Example 2: Poisonous Plant Classification**

x_1 : largeur de la feuille
x_2 : longueur de la feuille

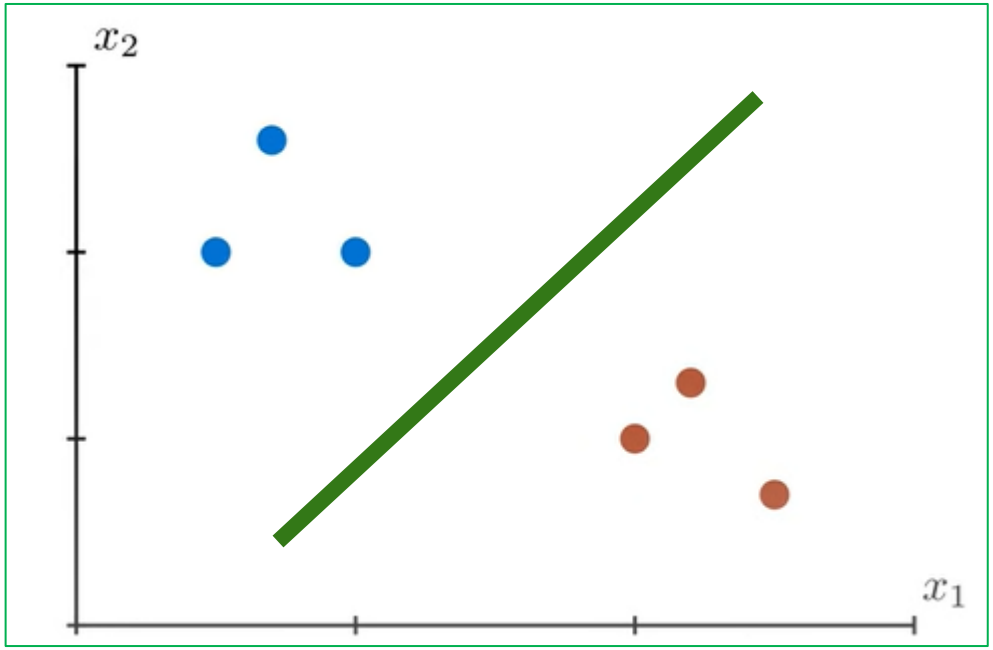
y	x_1	x_2
1	0.5	2.0
1	1.1	2.1
1	0.7	2.6
0	2.0	1.0
0	2.5	0.7
0	2.2	0.3



Motivation for Logistic Regression

- **Example 2: Poisonous Plant Classification**

$$z(x_1, x_2) = w_1 x_1 + w_2 x_2 + b$$

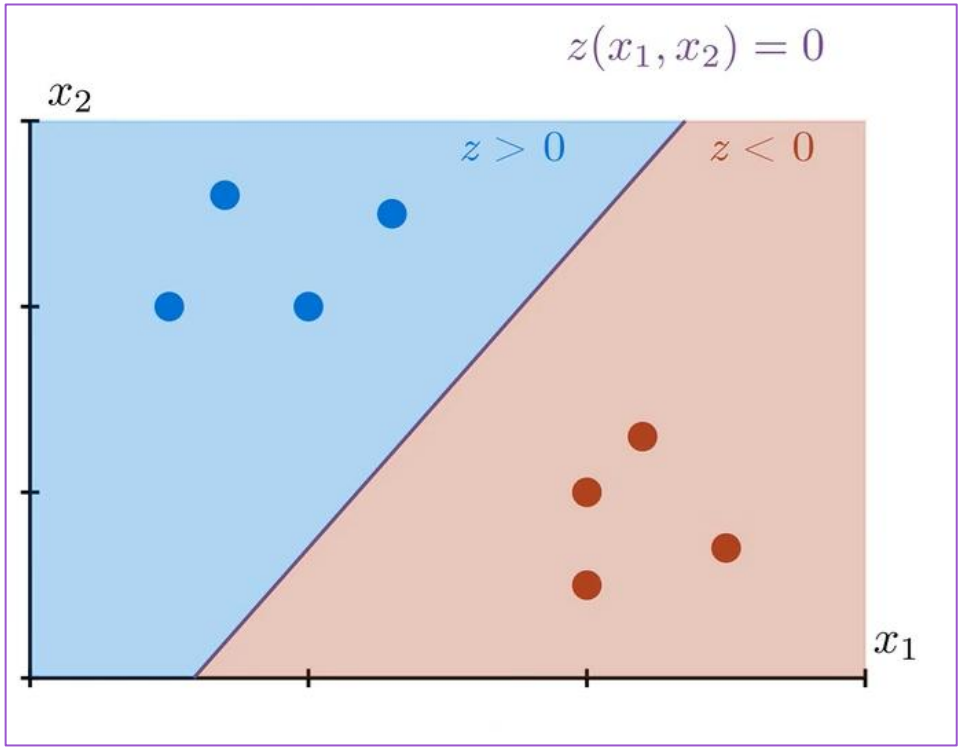


Motivation for Logistic Regression

- **Example 2: Poisonous Plant Classification**

$$z(x_1, x_2) = w_1 x_1 + w_2 x_2 + b$$

$$\begin{cases} y_{pred} = 0 & si \quad z < 0 \\ y_{pred} = 1 & si \quad z \geqslant 0 \end{cases}$$



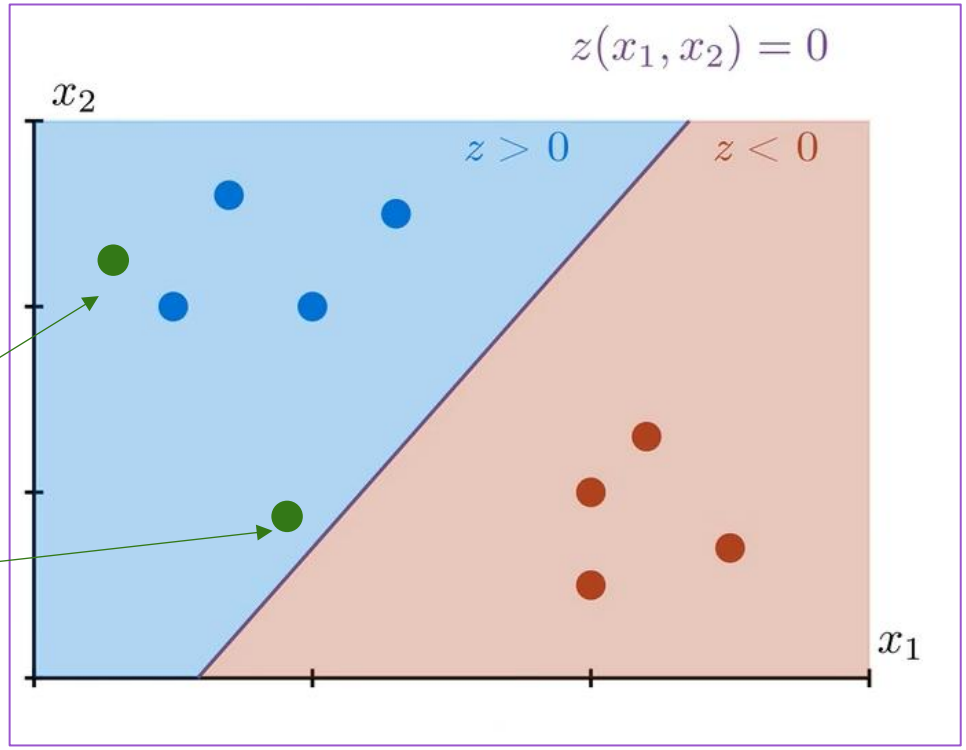
Motivation for Logistic Regression

- **Example 2: Poisonous Plant Classification**

$$z(x_1, x_2) = w_1 x_1 + w_2 x_2 + b$$

$$\begin{cases} y_{pred} = 0 & si \quad z < 0 \\ y_{pred} = 1 & si \quad z \geqslant 0 \end{cases}$$

What is the confidence level that these plants are poisonous



Logistic Regression

Motivation for Logistic Regression

Need for a New Approach

- Outputs values only between 0 and 1($0 \leq h_\theta(x) \leq 1$)
- Provides a clear Boundary Decision between poisonous and safe plants.

Logistic Regression

Motivation for Logistic Regression

Logistic Regression Model

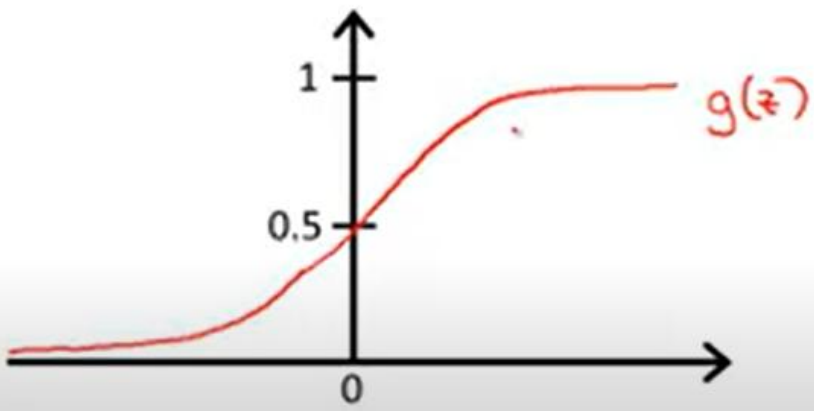
Want $0 \leq h_\theta(x) \leq 1$

$h_\theta(x) = g(\theta^T x)$

$g(z) = \frac{1}{1+e^{-z}}$

$z = \theta^T x$

$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$



→ Sigmoid function

→ Logistic function

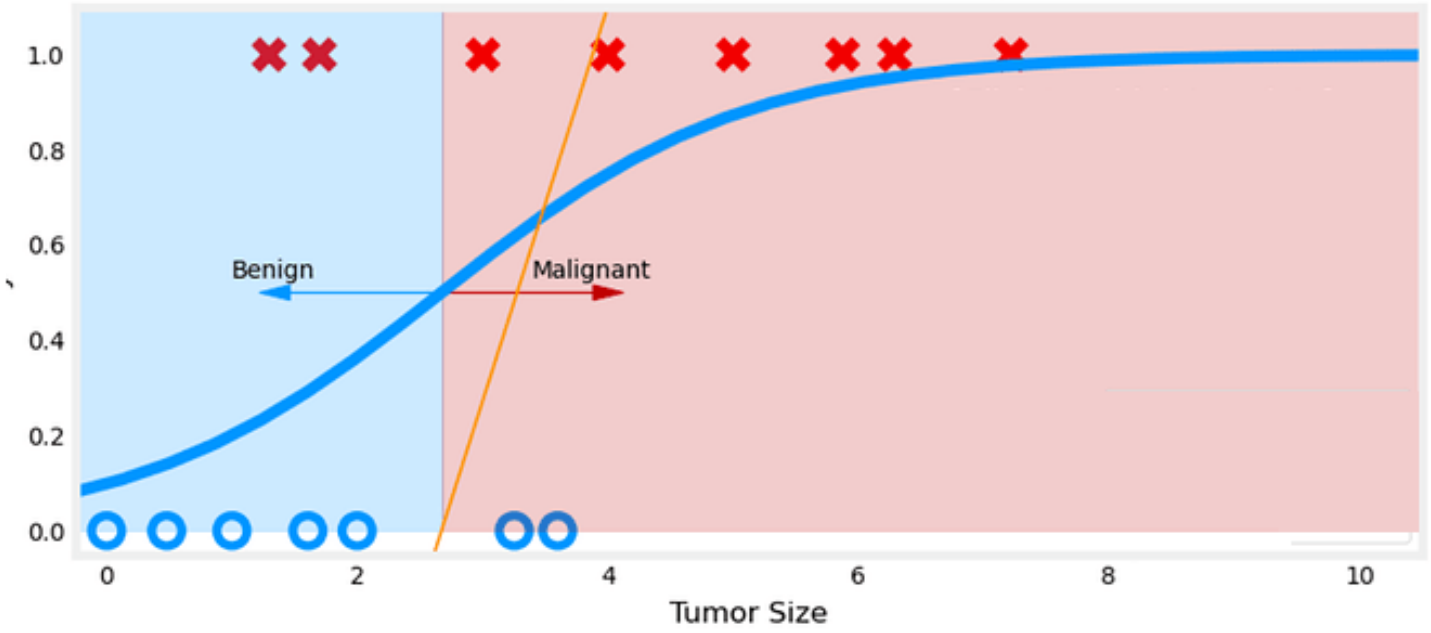
If $h_\theta(x) \geq 0.5$, predict $y = 1$ (malignant).

If $h_\theta(x) < 0.5$, predict $y = 0$ (benign).

Logistic Regression

Motivation for Logistic Regression

- **Example 1:**



Logistic Regression

Hypothesis Function

- **Logistic Regression Hypothesis Function**

Instead of a linear function, we use the Sigmoid function:

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

- This function always outputs values between 0 and 1, representing a probability.
- $\theta^T x$ is the linear combination of features and parameters in logistic regression.

Logistic Regression

Hypothesis Function

- **Logistic Regression Hypothesis Function**
- The probability of a given class is:

$$P(y = 1|x; \theta) = h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

$$P(y = 0|x; \theta) = 1 - h_\theta(x)$$

- This represents the probability that the output y is 1, given an input x and parameters θ.

Logistic Regression

Hypothesis Function

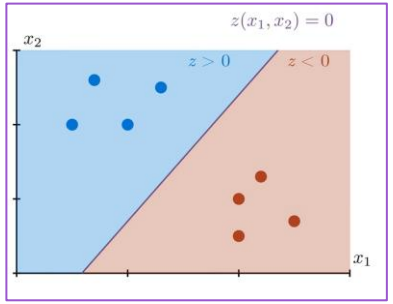
- **Logistic Regression Hypothesis Function**

$$\theta^T x = \sum_{i=0}^{n} \theta_i x_i$$

- x: The feature vector (input data)
- θ: The parameter vector (weights)
- $\theta^T x$: A single real-valued number that represents a weighted sum of the input.

Example:

$$\theta^T x = z(x_1, x_2) = w_1 x_1 + w_2 x_2 + b$$



Logistic Regression

Decision Boundary

- The decision boundary is the surface where $h_\theta(x) = 0.5$, i.e.,

$$\theta^T x = 0$$

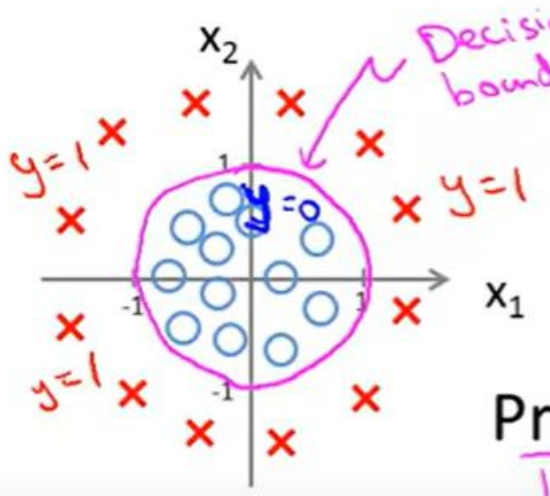
- **Linear <u>Decision Boundary</u>**: If *x* has two features, the boundary is a straight line.
- **Non-Linear <u>Decision Boundary</u>**: If using polynomial features, the boundary can be curved.
- Example in 2D Poisonous Plant Classification.

$$\theta_0 + \theta_1 x_1 + \theta_2 x_2 = 0$$

Logistic Regression

Decision Boundary

Non-linear decision boundaries



$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$$

$\theta = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$

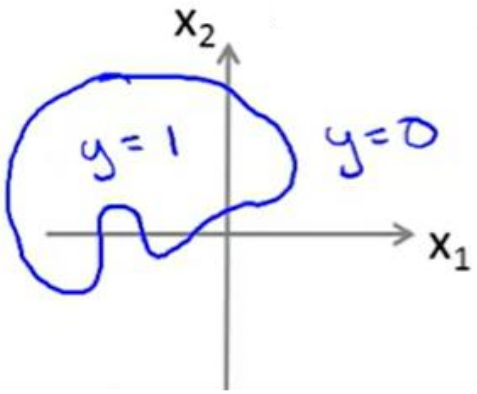
Predict "$y = 1$" if $-1 + x_1^2 + x_2^2 \geq 0$

$x_1^2 + x_2^2 = 1$

$x_1^2 + x_2^2 \geq 1$

Logistic Regression

Decision Boundary



$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_1^2 x_2 + \theta_5 x_1^2 x_2^2 + \theta_6 x_1^3 x_2 + \dots)$$

Logistic Regression

Cost Function : Log Loss

Why Not Use Mean Squared Error (MSE)?

- In linear regression, we minimize MSE:

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

- However, in logistic regression, using MSE leads to:
  - Non-convex loss function → Gradient descent may not converge.
  - Inefficient optimization → Poor updates for weights.

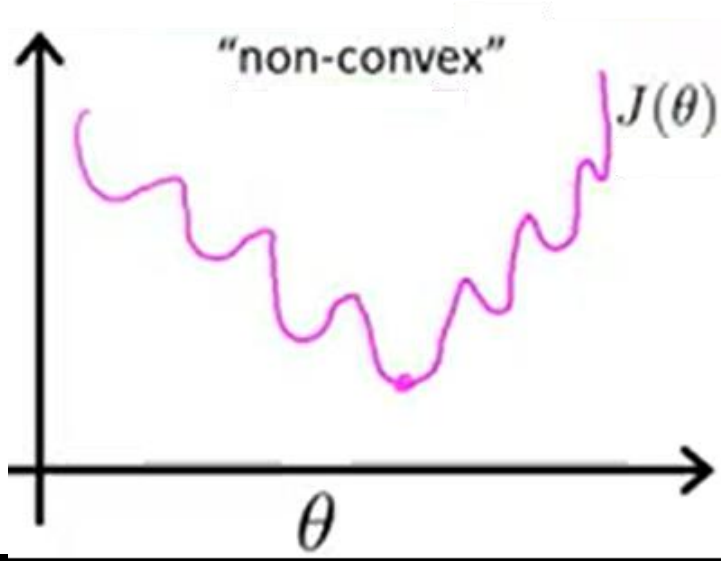
Logistic Regression

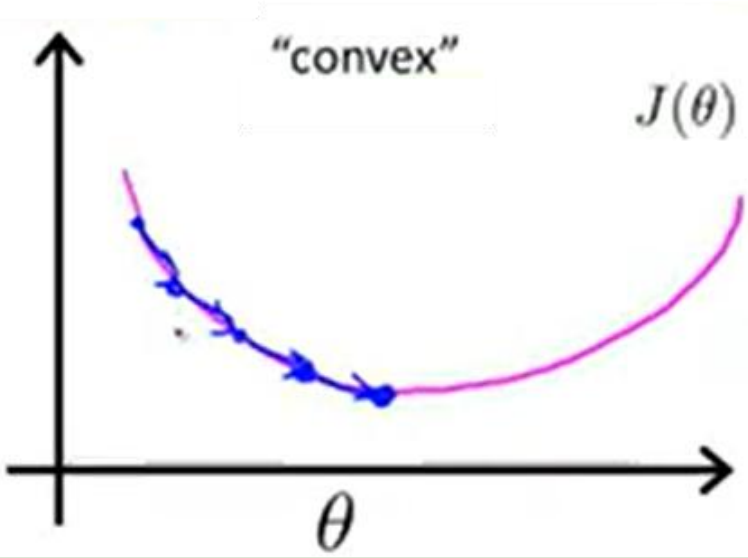
Cost Function : Log Loss

Why Not Use Mean Squared Error (MSE)?

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

$$\frac{1}{1+e^{-\theta^T x}}$$





Logistic Regression

Cost Function : Log Loss

Definition of Log Loss (Binary Classification)

- Log Loss is also called Binary Cross-Entropy Loss:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log h_\theta(x^{(i)}) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right]$$

where:

- $h_\theta(x^{(i)})$ is the **hypothesis function** (sigmoid output).
- $y^{(i)}$ is the **true class label** (0 or 1).

This function is convex, making it easy to minimize using gradient descent.

Logistic Regression

Cost Function : Log Loss

Understanding the Formula:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log h_\theta(x^{(i)}) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right]$$

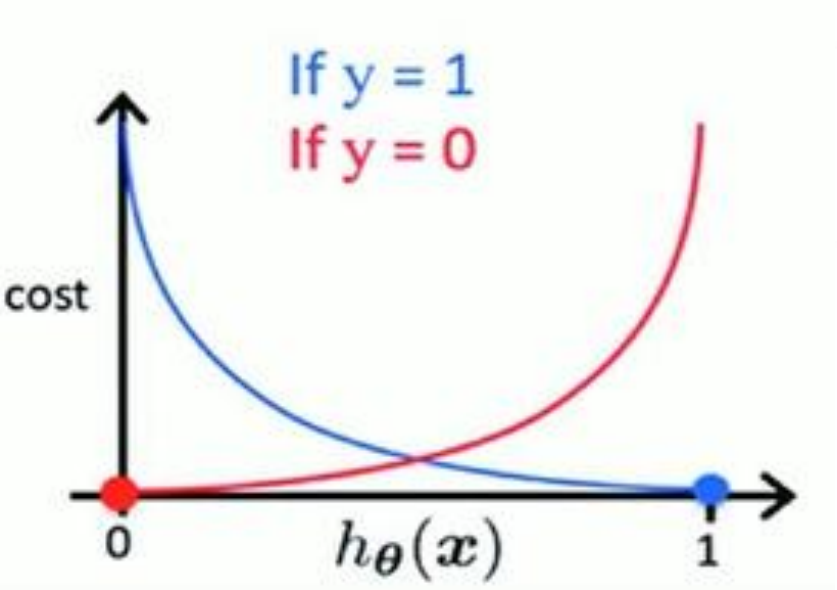
$$J(\theta) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1-h_\theta(x)) & \text{if } y = 0 \end{cases}$$

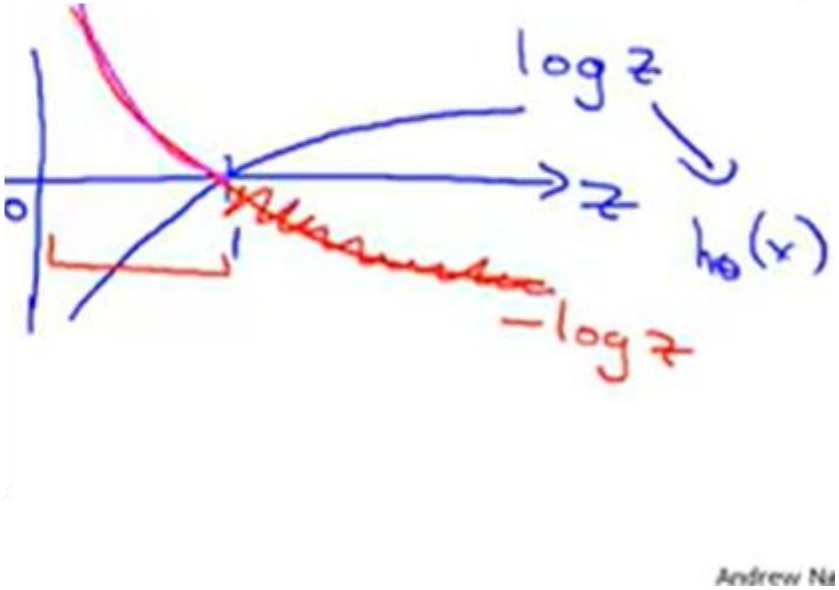
Logistic Regression

Cost Function : Log Loss

Understanding the Formula:

$$J(\theta) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$



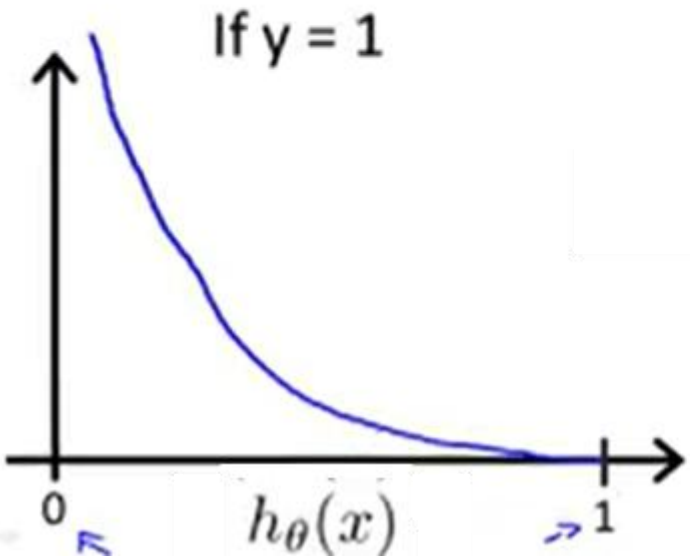


Logistic Regression

Cost Function : Log Loss

Understanding the Formula:

$$J(\theta) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$



If **true label** $y = 1$:

- Loss **decreases** as $h_\theta(x) \to 1$.

If **true label** $y = 0$:

- Loss **increases sharply** as $h_\theta(x) \to 1$.

Andrew Ng

Logistic Regression

Optimization (Gradient Descent Algorithm)

To find the best θ, we minimize $J(\theta)$ using Gradient Descent:

- This iteratively updates θ to reduce the loss.
- Similar to Linear Regression but adapted for classification.

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

- Matrix formula :

$$\theta := \theta - \alpha \nabla_\theta J(\theta)$$

where :

$$\nabla_\theta J(\theta) = \frac{1}{m} X^\top (h_\theta(X) - y)$$

Logistic Regression

Multiclass Classification

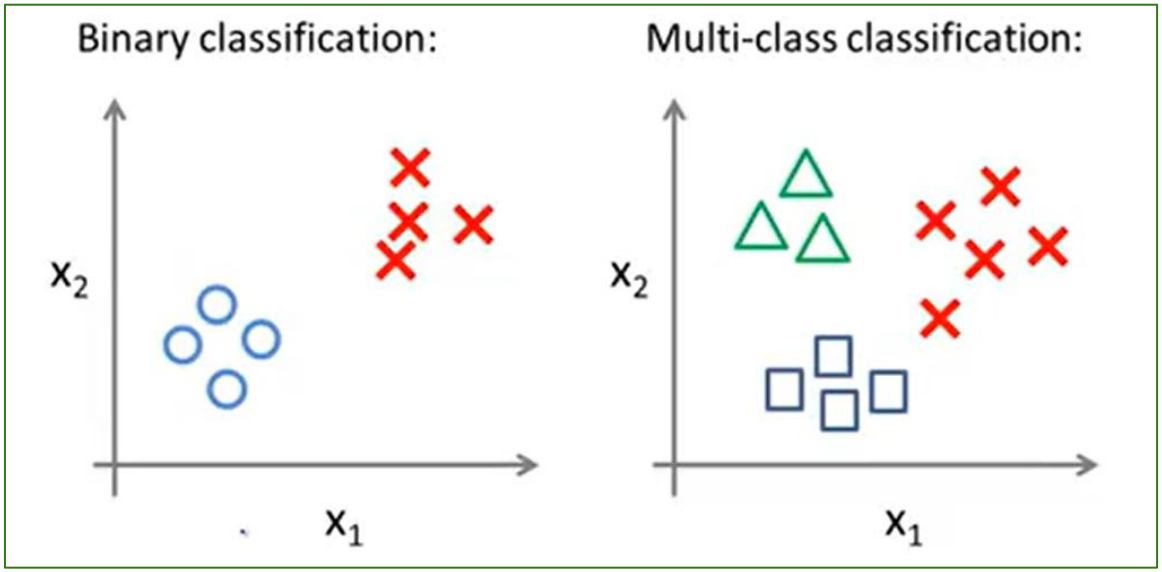
How about Multiclass Classification ?

- Multiclass classification deals with problems where there are more than two possible classes.
- Unlike binary classification (where we predict either 0 or 1), in multiclass classification, we have K possible classes (e.g., predicting if an image contains a cat, dog, or bird).
- Two common approaches:
  - One-vs-All (OvA) / One-vs-Rest (OvR)
  - Softmax Regression (Multinomial Logistic Regression)

Multiclass Classification (One-vs-All)

How One-vs-All Works?

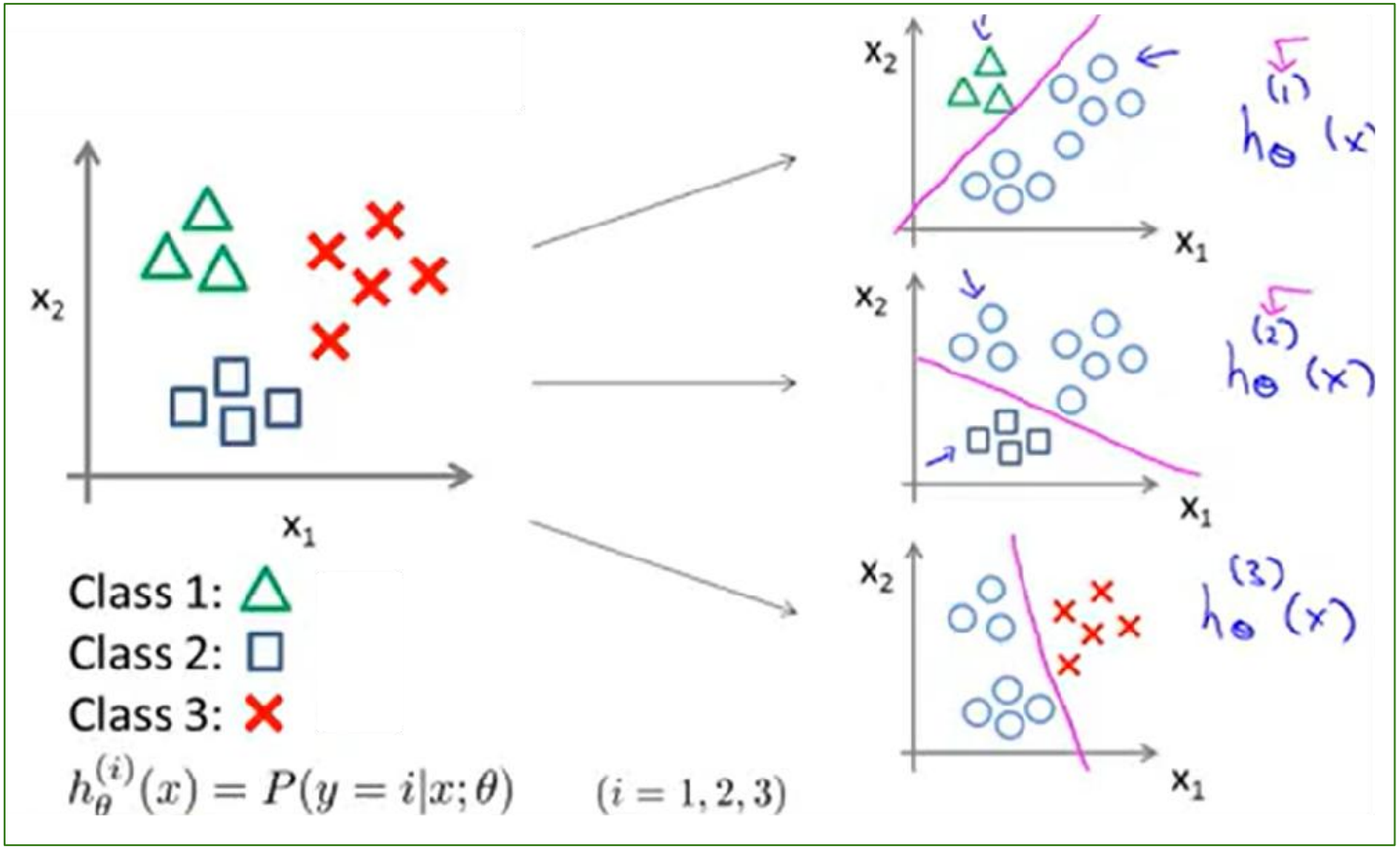
- Instead of training one classifier, we train K separate binary classifiers, each one distinguishing a particular class from all others.



Logistic Regression

Multiclass Classification (One-vs-All)

How One-vs-All Works?



How Softmax Works?

- Instead of K separate models, train one model with the Softmax function to predict probabilities for each class.
- Ensures all probabilities sum to 1.

$$P(y = k \mid x; \theta) = \frac{e^{\theta_k^T x}}{\sum_{j=1}^{K} e^{\theta_j^T x}}$$

- Final prediction:

$$\hat{y} = \arg\max_k P(y = k \mid x; \theta)$$

Cost Function for Softmax

- Loss Function: Categorical Cross-Entropy

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} \sum_{k=1}^{K} 1(y^{(i)} = k) \log P(y^{(i)} = k \mid x^{(i)}; \theta)$$

- Unlike logistic regression (binary cross-entropy), Softmax uses categorical cross-entropy.
- Trained using gradient descent.

Example – Poisonous Plant Classification

- Classes: Poisonous, Non-Toxic, Unknown
- **OvA:**
  - Train 3 classifiers (Poisonous vs. Rest, Non-Toxic vs. Rest, Unknown vs. Rest).
  - Predict based on highest probability.
- **Softmax:**
  - Train one model with Softmax.
  - Predict single probability distribution over 3 classes.

When to Use?

- If K is small, Softmax is often better
- If K is large, OvA might be preferable (less computation).
- If using Neural Networks, Softmax is the standard choice.

k-Nearest Neighbors (k-NN)

k-Nearest Neighbors (k-NN)

Intuition Behind k-NN

- Simple Analogy:

  "Tell me who your neighbors are, and I'll tell you who you are."

- k-NN classifies a data point based on the majority class of its k closest neighbors.

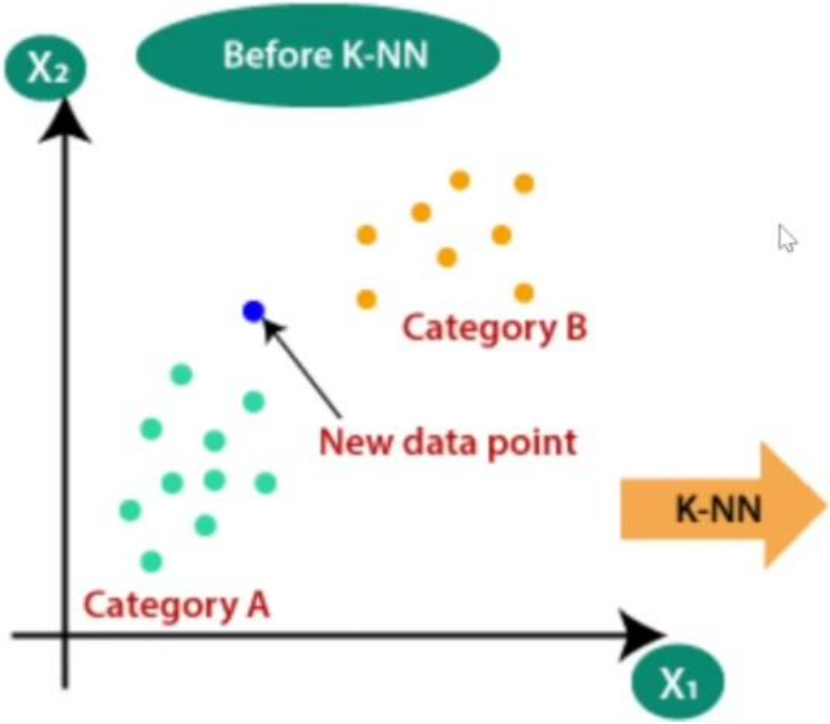
  - Example: If most of your 5 closest neighbors are "cat lovers," you're likely a " cat lover."

k-Nearest Neighbors (k-NN)

Intuition Behind k-NN

- Simple Analogy:

"Tell me who your neighbors are, and I'll tell you who you are."

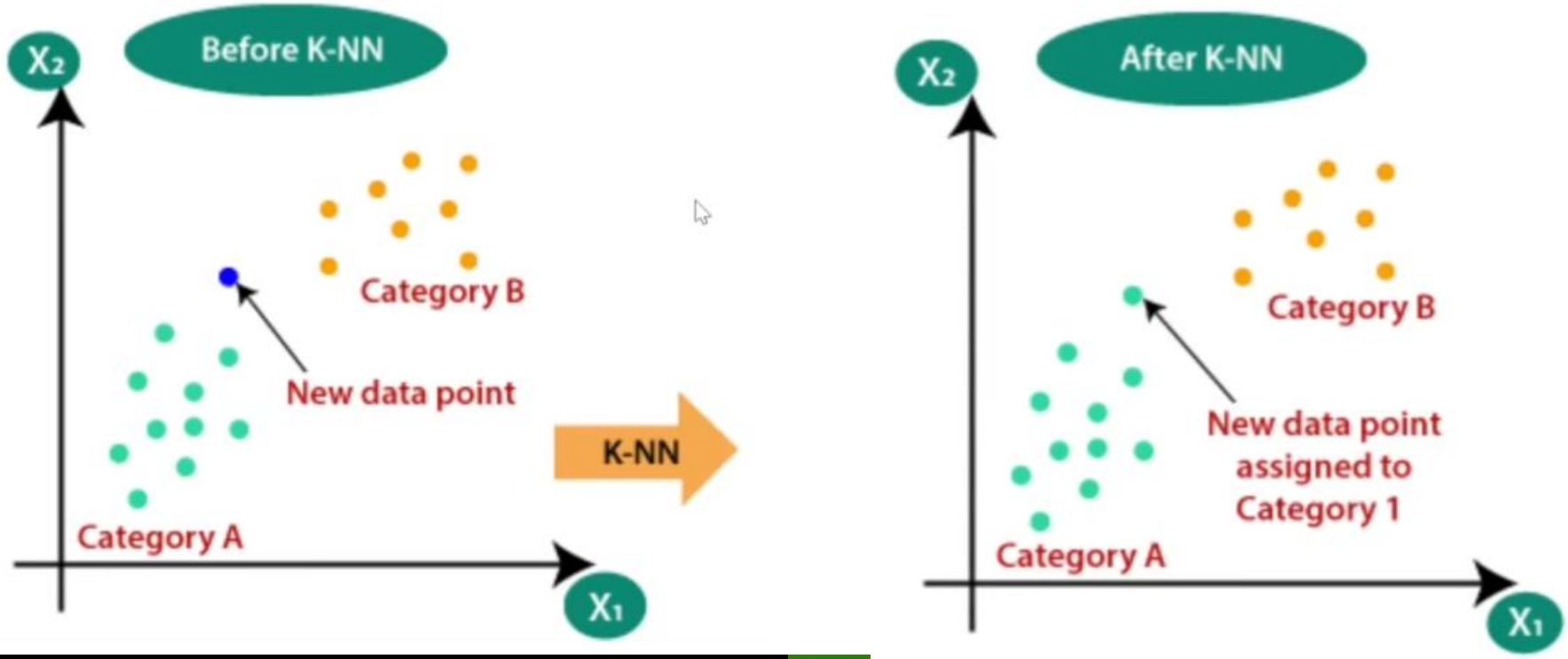


k-Nearest Neighbors (k-NN)

Intuition Behind k-NN

- Simple Analogy:

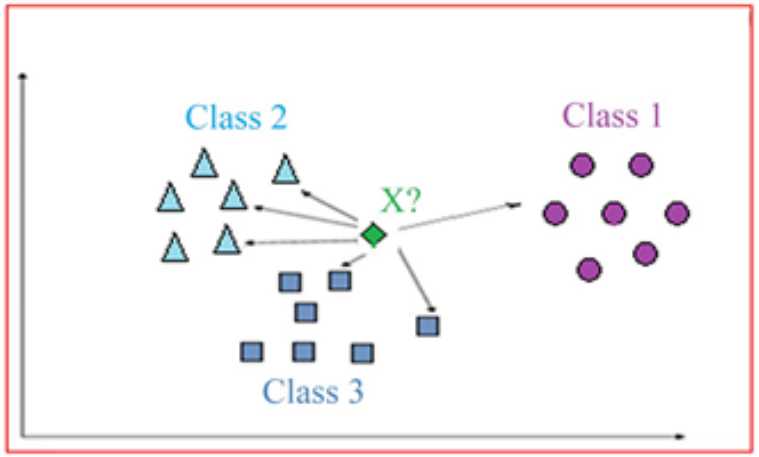
"Tell me who your neighbors are, and I'll tell you who you are."



k-Nearest Neighbors (k-NN)

How k-NN Works

Key Steps:



1. **Choose *k***: Number of neighbors to consider (e.g., *k*=3, *k*=5).
2. **Calculate Distance**: Measure distance between the query point and all training points.
3. **Find Neighbors**: Select the *k* closest points.
4. **Vote for Class**: Assign the majority class among the *k* neighbors.

k-Nearest Neighbors (k-NN)

How k-NN Works

Example: Classifying a Plant as Poisonous or Safe

- Features: Leaf color, shape, size.
- KNN Algorithm:
  1. Choose K=3.
  2. Compute the distances between the new plant and all known plants.
  3. Find the 3 nearest plants.
  4. If 2 out of 3 neighbors are poisonous, classify the plant as poisonous.

k-Nearest Neighbors (k-NN)

Distance Metrics

Key Steps:

- Euclidean Distance:

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

- Manhattan Distance:

$$d(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} |x_i - y_i|$$

- Cosine Similarity:

$$\cos(\theta) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|}$$

- Other Metrics: Minkowski, Hamming Distance (for categorical data).

Choosing *k*

- **Small k:**
  - High model complexity, sensitive to noise (overfitting).
- **Large k:**
  - Smooth decision boundaries, may underfit.
- **Best Practice:** Use cross-validation to select an optimal K.

k-Nearest Neighbors (k-NN)

Advantages & Disadvantages

- **Advantages:**
  - Simple and intuitive.
  - No training phase (lazy learning).
  - Works well with small datasets.
- **Disadvantages:**
  - Computationally expensive for large datasets.
  - Sensitive to irrelevant features.
  - Affected by class imbalance.

Naïve Bayes

What is Naïve Bayes?

- Naïve Bayes is a probabilistic classifier based on Bayes' Theorem with an assumption of conditional independence between features.
- Used for classification tasks like spam detection, sentiment analysis, and medical diagnosis.
- **Example Use Cases:**
  - **Spam Detection**: Classifying emails as spam or not spam.
  - **Medical Diagnosis**: Predicting whether a patient has a disease based on symptoms.
  - **Text Classification**: Sentiment analysis, topic categorization.

Bayes' Theorem: The Foundation of Naïve Bayes

- Bayes' Theorem states:

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

where:

$P(A \mid B)$ → **Posterior Probability** (probability of A given B).

$P(B \mid A)$ → **Likelihood** (probability of B given A).

$P(A)$ → **Prior Probability** (initial probability of A).

$P(B)$ → **Evidence** (probability of B occurring).

Bayes' Theorem

Bayes' Theorem: The Foundation of Naïve Bayes

- For classification, let: $X = (x_1, x_2, ..., x_n)$ be the feature vector.

  Y be the class label.

- The classifier predicts Y using:

$$P(Y \mid X) = \frac{P(X \mid Y)P(Y)}{P(X)}$$

Since $P(X)$ is the same for all classes, we only compare the numerator:

$$P(Y \mid X) \propto P(X \mid Y)P(Y)$$

Naïve Bayes

The "Naïve" Assumption: Feature Independence

Why "Naïve"?

- The model assumes that all features $x1,x2,...,xn$ are conditionally independent given the class label Y.

$$P(X \mid Y) = P(x_1, x_2, ..., x_n \mid Y) = P(x_1 \mid Y)P(x_2 \mid Y)...P(x_n \mid Y)$$

- Thus, the final classification rule becomes:

$$P(Y \mid X) \propto P(Y) \prod_{i=1}^{n} P(x_i \mid Y)$$

Example Dataset

Example

- Let's use a simple dataset to predict whether a fruit is a Banana or Orange based on two features:
  - Color: Yellow or Orange.
  - Size: Small or Large.

Color	Size	Fruit
Yellow	Small	Banana
Yellow	Large	Banana
Orange	Small	Orange
Orange	Large	Orange
Yellow	Large	Banana
Orange	Small	Orange

Example Dataset

Example

- Let's use a simple dataset to predict whether a fruit is a Banana or Orange based on two features:
  - Color: Yellow or Orange.
  - Size: Small or Large.

Color	Size	Fruit
Yellow	Small	Banana
Yellow	Large	Banana
Orange	Small	Orange
Orange	Large	Orange
Yellow	Large	Banana
Orange	Small	Orange

We want to predict the class of a new fruit with:

- **Color = Yellow**
- **Size = Small**

Example Dataset

Example

- Compute Prior Probabilities

$$P(\text{Banana}) = \frac{3}{6} = 0.5$$
$$P(\text{Orange}) = \frac{3}{6} = 0.5$$

- Compute Likelihoods
  - For Banana:
$$P(\text{Yellow}|\text{Banana}) = \frac{3}{3} = 1$$
$$P(\text{Small}|\text{Banana}) = \frac{1}{3} \approx 0.33$$
  - For Orange:
$$P(\text{Yellow}|\text{Orange}) = \frac{0}{3} = 0$$
$$P(\text{Small}|\text{Orange}) = \frac{2}{3} \approx 0.67$$

Color	Size	Fruit
Yellow	Small	Banana
Yellow	Large	Banana
Orange	Small	Orange
Orange	Large	Orange
Yellow	Large	Banana
Orange	Small	Orange

Example Dataset

Example

- Compute Posterior Probabilities

For Banana:

$$P(\text{Banana}|\text{Yellow}, \text{Small}) \propto P(\text{Yellow}|\text{Banana}) \cdot P(\text{Small}|\text{Banana}) \cdot P(\text{Banana})$$

$$P(\text{Banana}|\text{Yellow}, \text{Small}) \propto 1 \cdot 0.33 \cdot 0.5 = 0.165$$

For Orange:

$$P(\text{Orange}|\text{Yellow}, \text{Small}) \propto P(\text{Yellow}|\text{Orange}) \cdot P(\text{Small}|\text{Orange}) \cdot P(\text{Orange})$$

$$P(\text{Orange}|\text{Yellow}, \text{Small}) \propto 0 \cdot 0.67 \cdot 0.5 = 0$$

Since $P(\text{Orange}|\text{Yellow}, \text{Small}) = 0$, the fruit is classified as **Banana**.

Example

Handling Zero Probabilities

- To avoid this, we use Laplace Smoothing (add 1 to each count):

For Orange:

$$P(\text{Yellow}|\text{Orange}) = \frac{0+1}{3+2} = \frac{1}{5} = 0.2$$
$$P(\text{Small}|\text{Orange}) = \frac{2+1}{3+2} = \frac{3}{5} = 0.6$$

$$P(\text{Orange}|\text{Yellow}, \text{Small}) \propto 0.2 \cdot 0.6 \cdot 0.5 = 0.06$$

The fruit is still classified as **Banana** (since $0.165 > 0.06$).

Types of Naïve Bayes Classifiers

- Gaussian Naïve Bayes
- Multinomial Naïve Bayes
- Bernoulli Naïve Bayes

Types of Naïve Bayes Classifiers

Gaussian Naïve Bayes (For Continuous Data)

- Assumes features follow a normal (Gaussian) distribution.
- The Probability Density Function (PDF) is:

$$P(x_i \mid Y) = \frac{1}{\sqrt{2\pi\sigma_Y^2}} \exp\left(-\frac{(x_i - \mu_Y)^2}{2\sigma_Y^2}\right)$$

Mean: μ_Y is the average of feature x for class Y.

Variance: σ_Y^2 is the variance for class Y.

Types of Naïve Bayes Classifiers

Gaussian Naïve Bayes (For Continuous Data)

- Assumes features follow a normal (Gaussian) distribution.
- The Probability Density Function (PDF) is:

$$P(x_i \mid Y) = \frac{1}{\sqrt{2\pi\sigma_Y^2}} \exp\left(-\frac{(x_i - \mu_Y)^2}{2\sigma_Y^2}\right)$$

Mean: μ_Y

Variance:

Fruit	Weight (g)	Sugar (%)	Acidity (pH)	Label (1 = Banana, 0 = Orange)
Banana	120	15.5	4.8	1
Banana	130	16.0	4.9	1
Banana	125	14.8	4.7	1
Orange	150	9.0	3.2	0
Orange	160	8.5	3.0	0
Orange	140	9.2	3.1	0

Types of Naïve Bayes Classifiers

Multinomial Naïve Bayes (For Text Classification)

- Used when features are discrete counts (e.g., word frequencies in text).
- Assumes each class follows a multinomial distribution:

$$P(x_i \mid Y) = \frac{\text{count of } x_i \text{ in class } Y}{\text{total count of words in class } Y}$$

Types of Naïve Bayes Classifiers

Multinomial Naïve Bayes (For Text Classification)

- Used when features are discrete counts (e.g., word frequencies in text).
- Assumes each class follows a multinomial distribution:

$$P(x_i \mid Y) = \frac{\text{count of } x_i \text{ in class } Y}{}$$

Email Text	Label (Spam=1, Not Spam=0)
"Win a free lottery now"	1 (Spam)
"Limited-time offer, claim prize"	1 (Spam)
"Meeting at 10 AM tomorrow"	0 (Not Spam)
"Project deadline extended"	0 (Not Spam)

Types of Naïve Bayes Classifiers

Bernoulli Naïve Bayes (For Binary Features)

- Used when features are binary (0 or 1).
  - Example: Whether a word appears in an email (Spam detection).

$$P(x_i \mid Y) = p_i^{x_i}(1 - p_i)^{(1-x_i)}$$

where p_i is the probability of feature x_i occurring in class Y.

Types of Naïve Bayes Classifiers

Bernoulli Naïve Bayes (For Binary Features)

- Used when features are binary (0 or 1).
  - Example: Whether a word appears in an email (Spam detection).

$$P(x_i \mid Y) = p^{x_i}(1-p)^{(1-x_i)}$$

E-mail	"Offre"	"Gratuit"	"Cliquez"	"Urgent"	Classe (Spam=1, Non Spam=0)
1	1	1	1	0	1 (Spam)
2	0	1	1	1	1 (Spam)
3	1	0	0	0	0 (Non Spam)
4	0	1	0	0	0 (Non Spam)

Naïve Bayes

Training the Naïve Bayes Model

How to Train the Naïve Bayes Model?

1. **Calculate Class Priors:** $$P(Y) = \frac{\text{Number of samples in class Y}}{\text{Total number of samples}}$$

2. **Estimate Likelihoods** $P(x_i \mid Y)$:
   - For **Gaussian Naïve Bayes**, estimate mean μ_Y and variance σ_Y^2 for each feature.
   - For **Multinomial Naïve Bayes**, count occurrences of words in each class.
   - For **Bernoulli Naïve Bayes**, compute feature probabilities.
3. **Classify a New Sample:**
   - Compute $P(Y \mid X)$ for each class Y.
   - Choose the class with the highest probability.

Naïve Bayes

Example: Naïve Bayes for Spam Classification

- Dataset Example:

Email	Contains "money"? (x1)	Contains "offer"? (x2)	Contains "free"? (x3)	Spam (Y)
1	1	1	1	Yes
2	1	1	0	Yes
3	0	0	1	No
4	1	0	0	No

- **Training the Model**
  - Compute **prior probabilities** $P(Y)$.
  - Compute **likelihoods** $P(x_i \mid Y)$.
  - Apply **Bayes' Theorem** to classify new emails.

Naïve Bayes

Advantages and Disadvantages

Advantages:

- Fast and efficient, even for large datasets.
- Handles missing data well.
- Works well with text data (e.g., spam filtering).

Disadvantages:

- Feature independence assumption is often unrealistic.
- Performs poorly with highly correlated features.
- For continuous data, Gaussian Naïve Bayes assumes a normal distribution, which may not always hold.

Decision Tree

What is a Decision Tree ?

- A Decision Tree is a supervised learning algorithm used for classification and regression.
- It works by splitting data based on feature values, creating a tree-like structure where each internal node represents a decision rule and each leaf node represents a class label or regression value.
- **Example Use Cases:**
  - Medical Diagnosis: Classifying patients based on symptoms.
  - Loan Approval: Determining whether a customer is eligible for a loan.
  - Spam Detection: Filtering emails as spam or not spam.

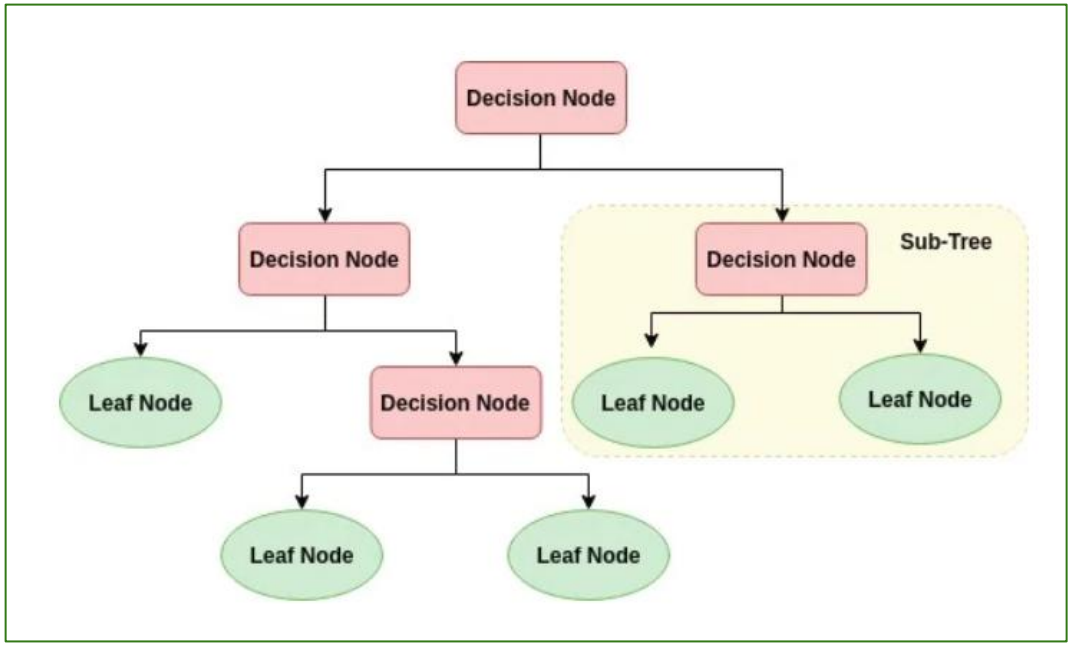
Why Use Decision Trees?

- Simple to understand & interpret (mimics human decision-making).
- Handles numerical & categorical data.
- No need for feature scaling.
- Works well with missing values.

Decision Trees

How a Decision Tree Works

Intuition

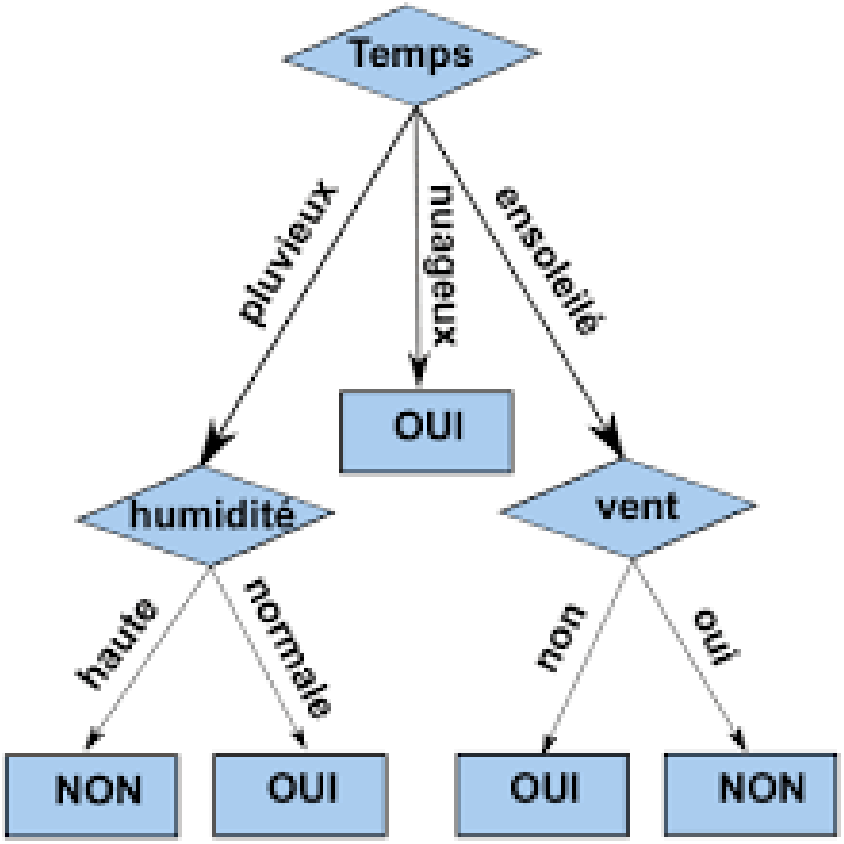


- A Decision Tree repeatedly splits the dataset into smaller subsets based on the feature that provides the most information gain.
- This process continues until a stopping criterion is met
  - e.g., max depth reached, no more splits possible.

Decision Trees

How a Decision Tree Works

Example: play tennis?



How to Split a Node?

To decide the best feature to split on, we measure impurity using one of the following:

Gini Impurity (Used in CART Algorithm)

- The Gini Impurity measures how impure a node is. It is defined as:

$$Gini = 1 - \sum_{i=1}^{C} p_i^2$$

where p_i is the probability of class i in the node.

How to Split a Node?

Gini Impurity (Used in CART Algorithm)

- Best split is the one that reduces Gini Impurity the most.
- Lower Gini = Better purity.

- Example: If a node contains 90% edible mushrooms and 10% poisonous mushrooms, then:

$$Gini = 1 - (0.9^2 + 0.1^2) = 1 - (0.81 + 0.01) = 0.18$$

How to Split a Node?

Entropy & Information Gain (Used in ID3 Algorithm)

- Entropy measures the randomness in a dataset:

$$Entropy = -\sum_{i=1}^{C} p_i \log_2 p_i$$

where p_i is the probability of class i.

- High entropy = High disorder (bad split).
- Low entropy = Low disorder (good split).
- The best feature to split on is the one that maximizes Information Gain:

$$IG = Entropy(parent) - \sum(\text{weight of child}) \times Entropy(child)$$

Decision Trees

Stopping Criteria

A Decision Tree stops growing when:

- All samples belong to the same class.
- Max tree depth is reached.
- Gini Impurity or Entropy is below a threshold.
- Number of samples in a node is too small.

Decision Trees

Advantages & Disadvantages

Advantages:

- Interpretable & easy to visualize.
- No need for feature scaling.
- Handles both numerical & categorical data.
- Can handle missing values well.

Disadvantages:

- Prone to overfitting (requires pruning).
- Not stable (small changes in data can lead to a different tree).
- Biased towards features with more levels.

Support Vector Machine (SVM)

Support Vector Machine

What is SVM?

- Support Vector Machine (SVM) is a powerful supervised learning algorithm used for classification and regression tasks. It finds the optimal hyperplane that best separates the data into different classes.
- Best suited for complex classification problems with clear margin of separation.
- Effective in high-dimensional spaces and works well for both linear and non-linear data.

Understanding the Core Idea of SVM

Finding the Optimal Hyperplane

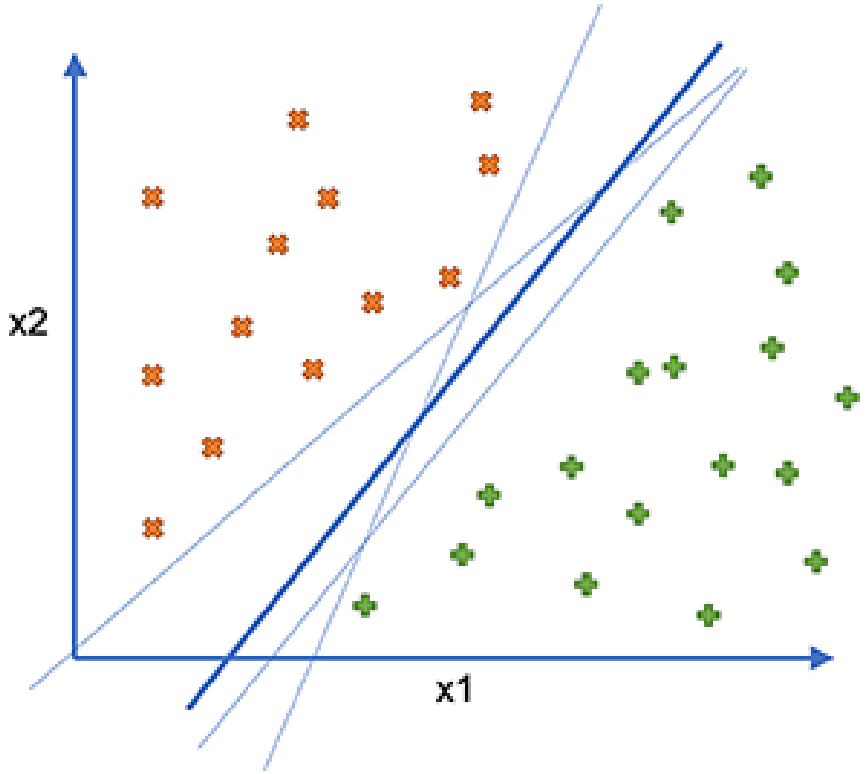
In a classification problem, SVM tries to find the best possible decision boundary that maximizes the margin between two classes.

- The hyperplane is the decision boundary that separates different classes.
- The margin is the distance between the hyperplane and the closest data points (support vectors).
- Support vectors are the points that are closest to the hyperplane and define its position.

Goal: Maximize the margin between support vectors while ensuring correct classification.

Understanding the Core Idea of SVM

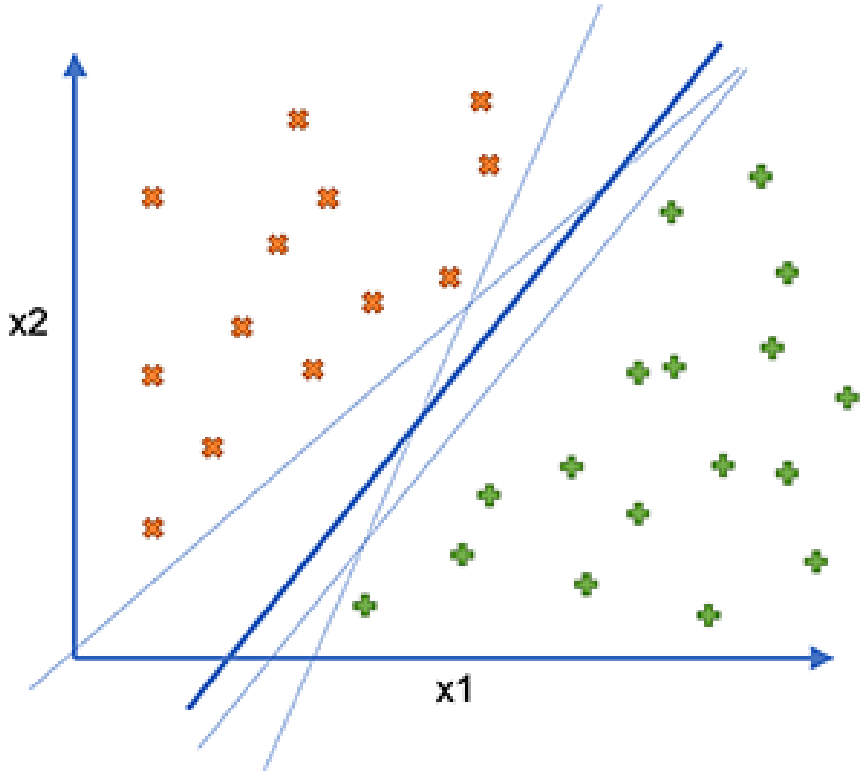
Example 1

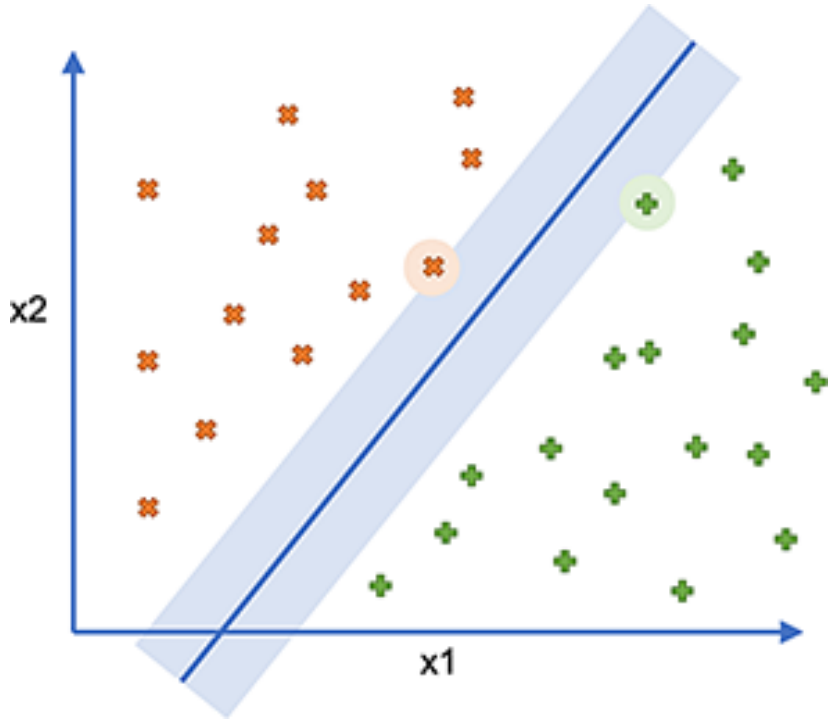


Support Vector Machine

Understanding the Core Idea of SVM

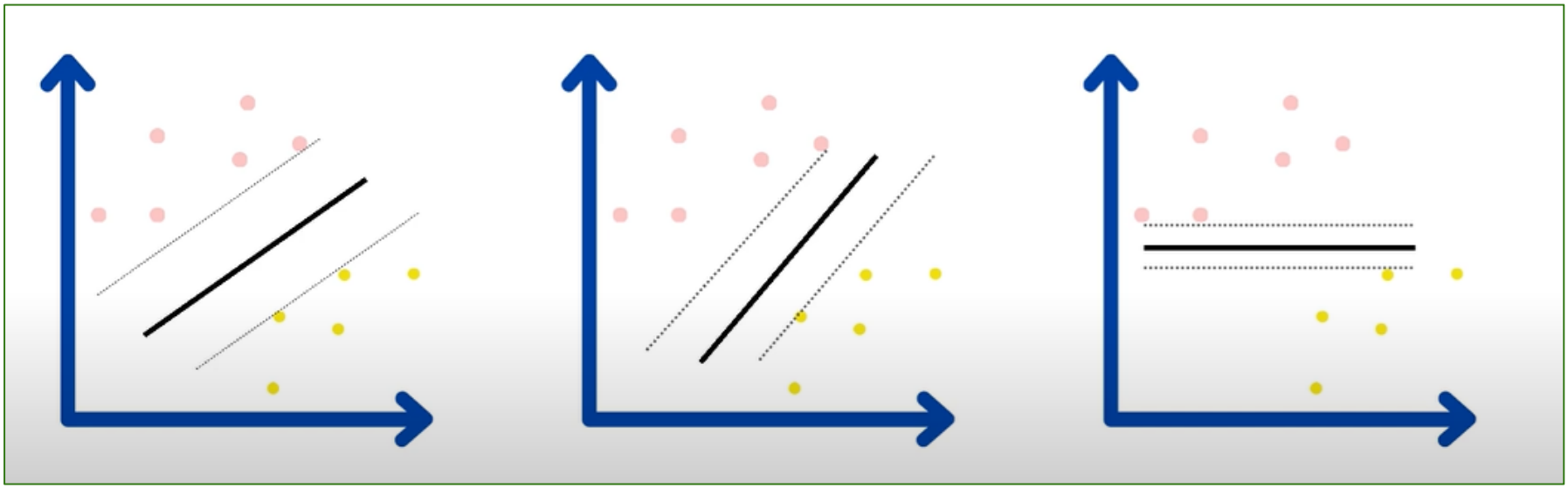
Example 1





Understanding the Core Idea of SVM

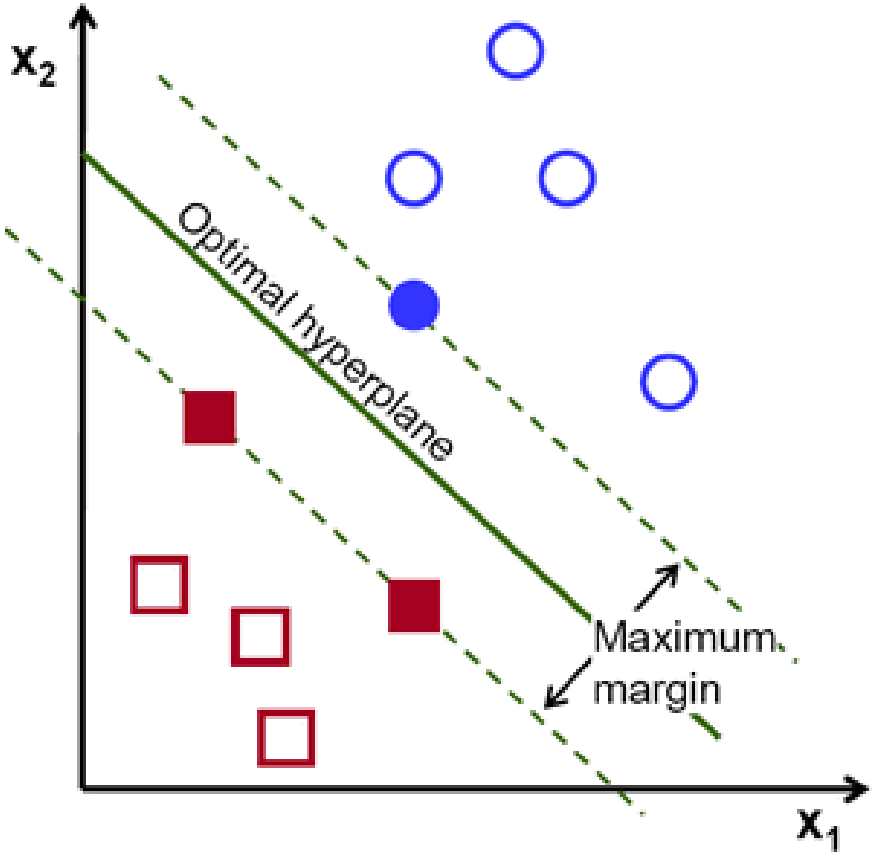
Example 2



Support Vector Machine

Understanding the Core Idea of SVM

Example 3



Mathematical Formulation of SVM

1- Defining the Hyperplane

SVM aims to maximize the margin between classes while ensuring correct classification. For a dataset with features X and labels y:

- A hyperplane is represented as:

$$w^T x + b = 0$$

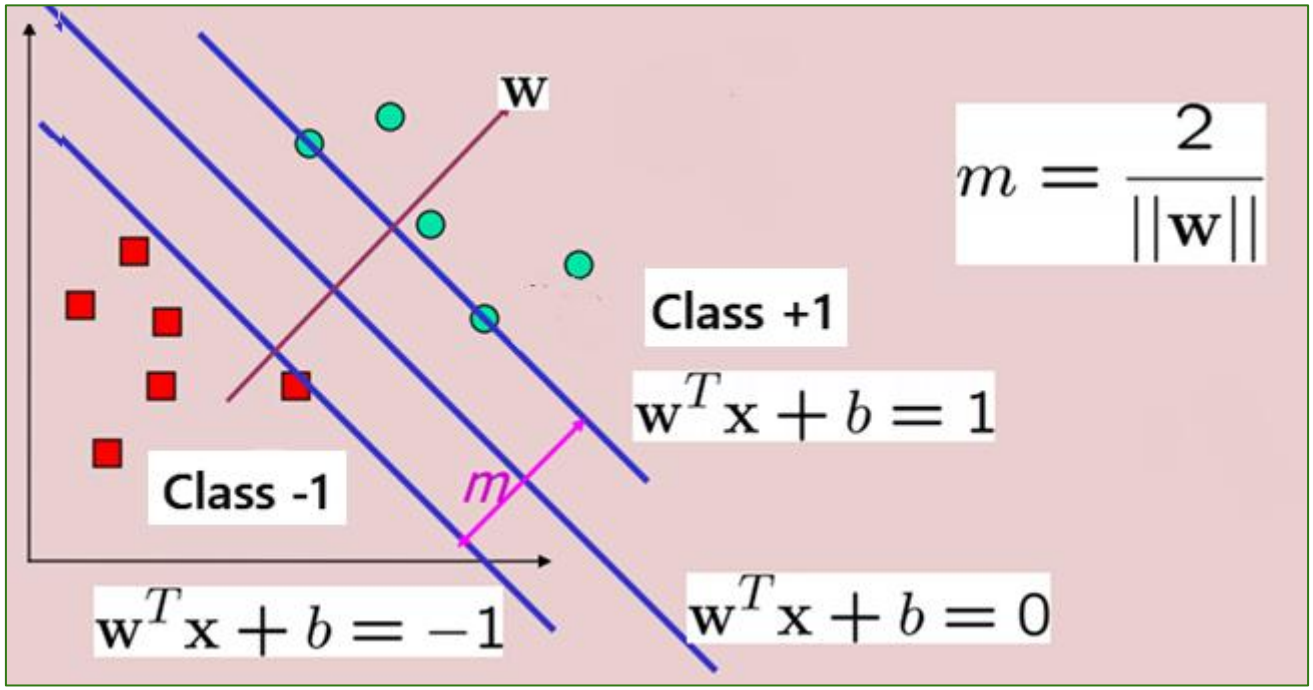
where:

- w is the weight vector (normal to the hyperplane),
- x is the feature vector,
- b is the bias term.

Mathematical Formulation of SVM

2- Decision Rule

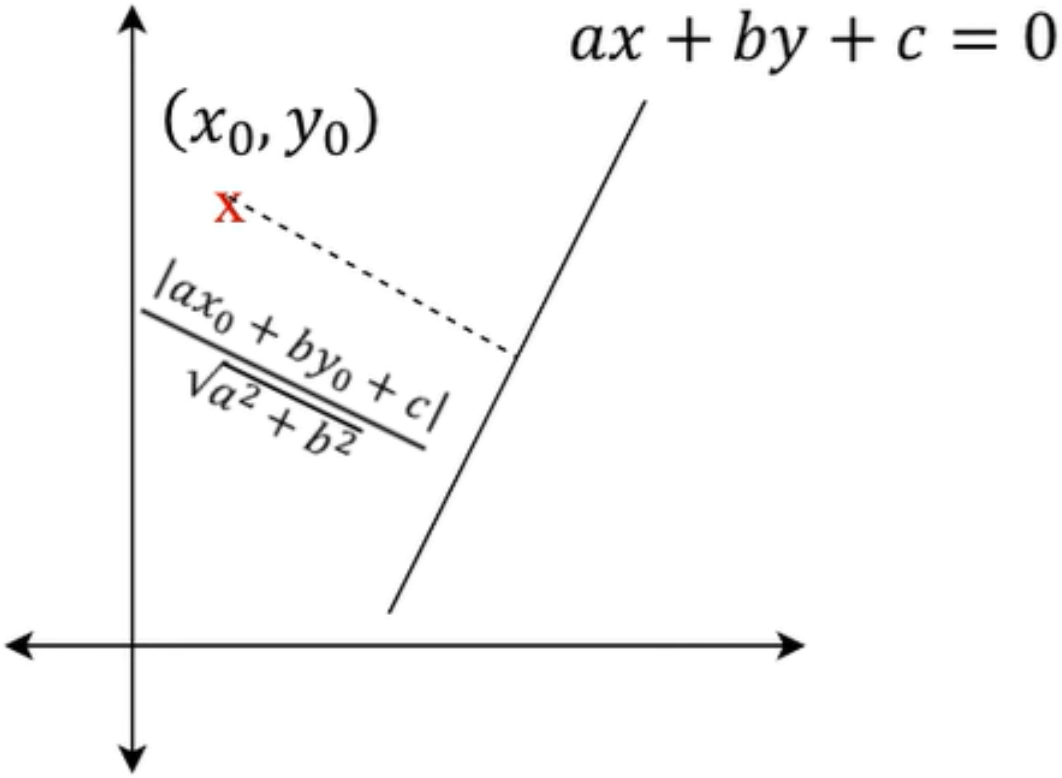
To classify a new point x, we use: $f(x) = w^T x + b$



Mathematical Formulation of SVM

2- Decision Rule

How to calculate distance?



3- Maximizing the Margin

- The margin is the distance between the hyperplane and the support vectors. The objective is to maximize this margin, which is defined as:

$$\text{Margin} = \frac{2}{||w||}$$

- This leads to the optimization problem:

$$\min \frac{1}{2}||w||^2$$

Subject to:

$$y_i(w^T x_i + b) \geq 1, \quad \forall i$$

Mathematical Formulation of SVM

3- Maximizing the Margin (Hinge Loss)

The **Hinge Loss** function in SVM is:

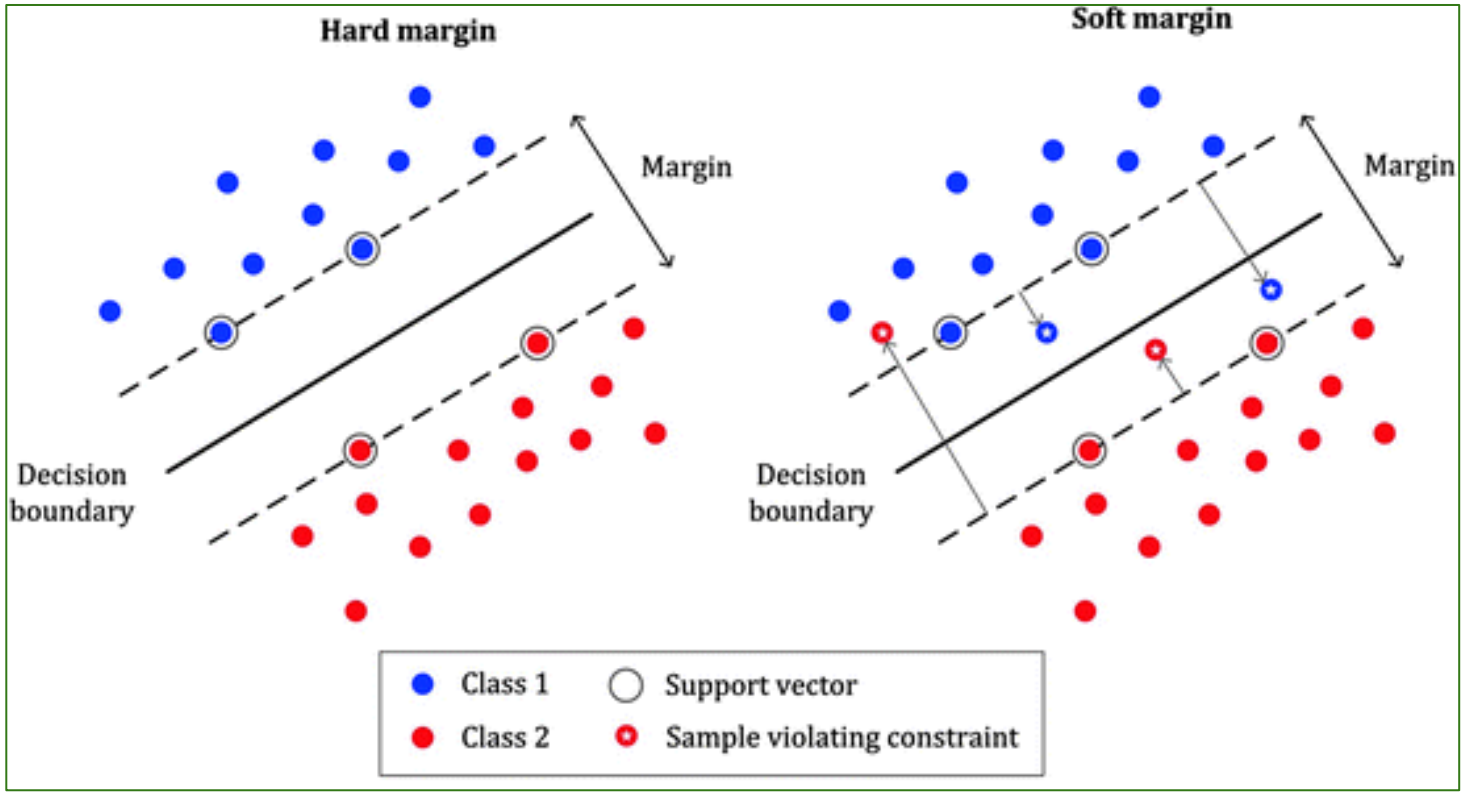
$$\text{Loss} = \sum_{i=1}^{n} \max(0, 1 - y_i(w \cdot x_i + b))$$

- If $y(w \cdot x + b) \geq 1$ → No penalty (correct classification & outside margin).
- If $y(w \cdot x + b) < 1$ → Penalty applied (inside the margin or misclassified).

Support Vector Machine

Soft Margin SVM (Handling Overlapping Classes)

For real-world data, strict separation may not always be possible due to noise or outliers.



Support Vector Machine

Soft Margin SVM (Handling Overlapping Classes)

For real-world data, strict separation may not always be possible due to noise or outliers.

- Solution: Introduce a slack variable ξ to allow some misclassifications:

$$y_i(w^T x_i + b) \geq 1 - \xi_i$$

where ξ_i is a penalty for misclassified points.

Support Vector Machine

Soft Margin SVM (Handling Overlapping Classes)

New Objective Function (Hinge Loss):

$$\min \frac{1}{2}||w||^2 + C\sum_{i=1}^{n}\xi_i$$

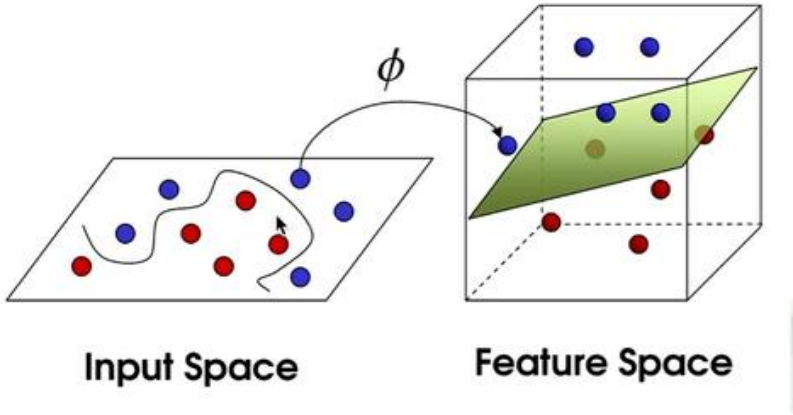
where :

- ξ_i is a penalty for misclassified points.
- C is a regularization parameter controlling trade-off between maximizing margin and minimizing misclassification.

Support Vector Machine

Non-Linear SVM: Kernel Trick

- If data is not linearly separable, we map it to a higher-dimensional space where a linear hyperplane can separate the classes.
- Solution: Use a kernel function $K(xi,xj)$ to compute similarity without explicitly transforming the data.



Support Vector Machine

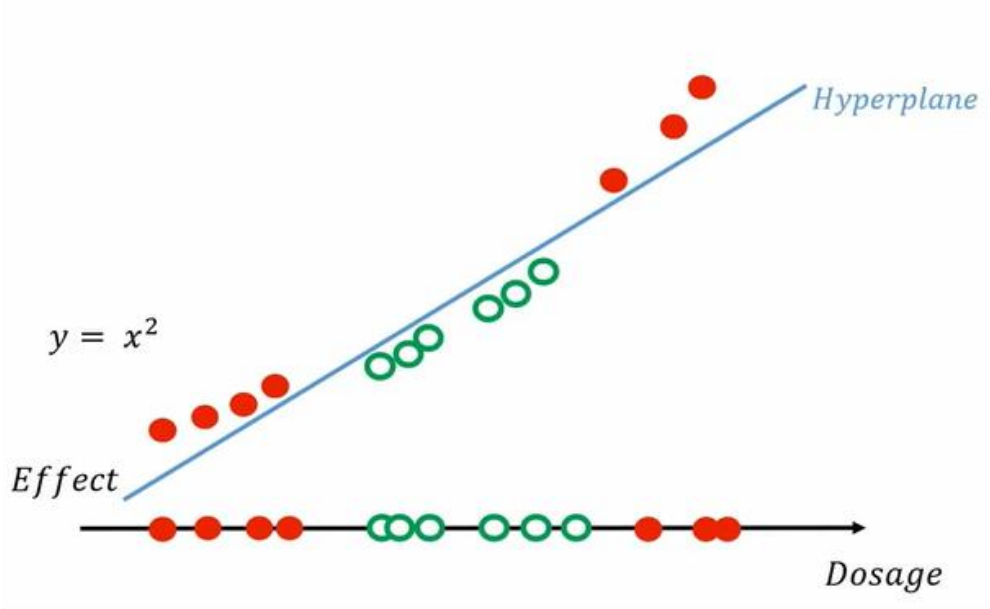
Non-Linear SVM: Kernel Trick

- Example 1:

Support Vector Machine

Non-Linear SVM: Kernel Trick

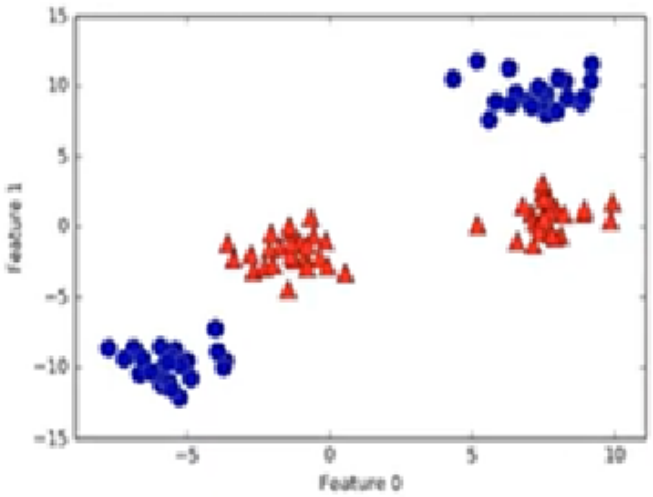
- Example 1:

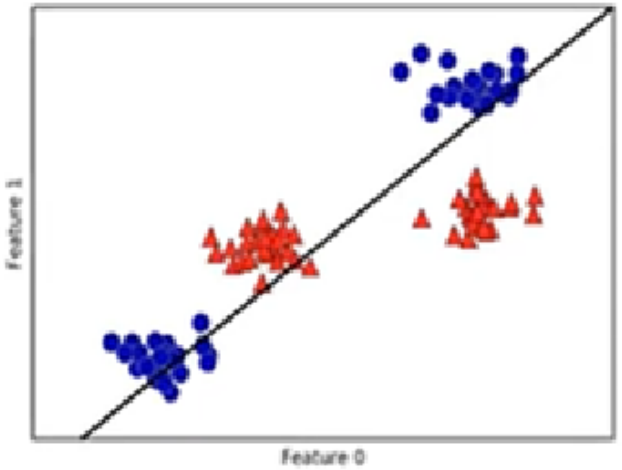


Support Vector Machine

Non-Linear SVM: Kernel Trick

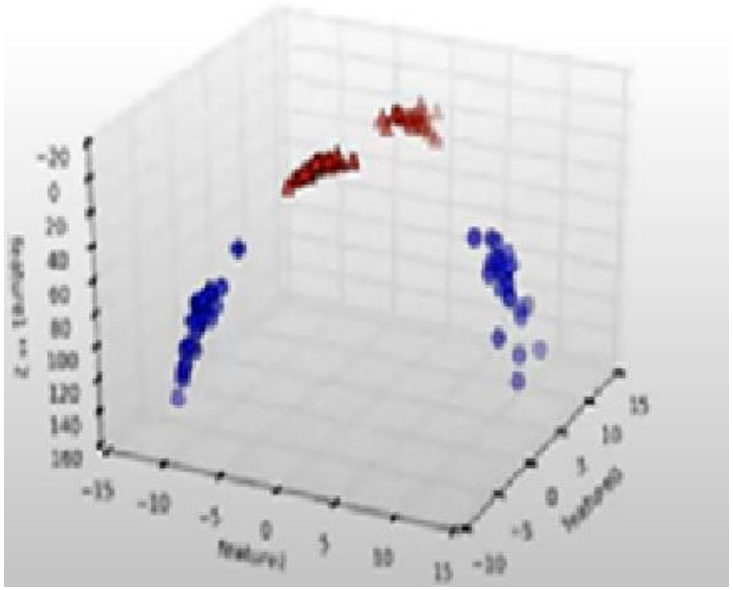
- Example 2:

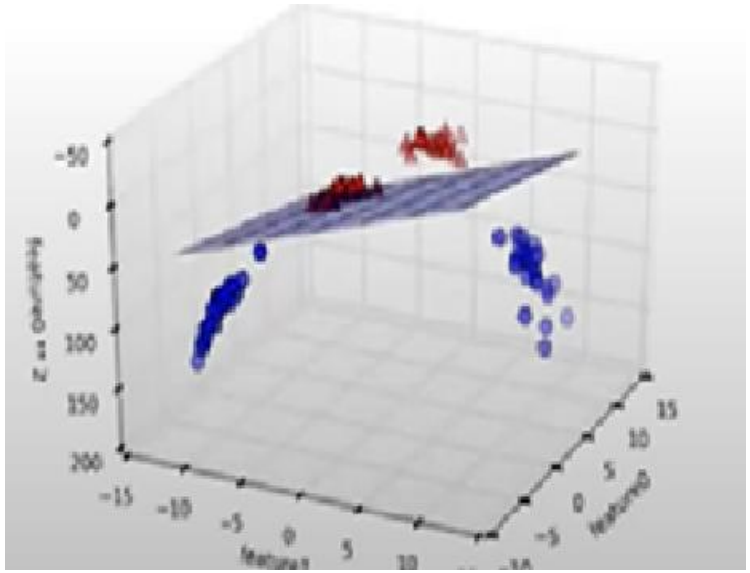


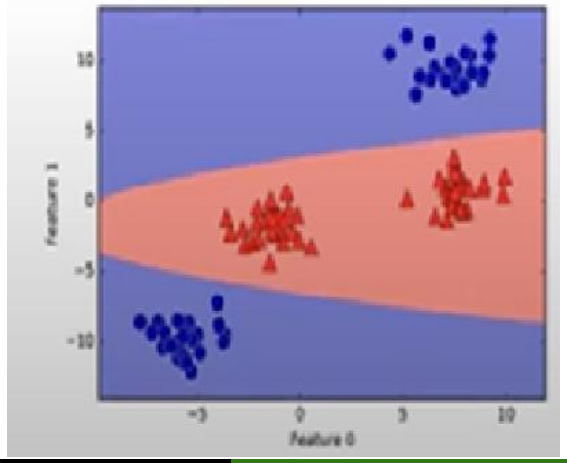


Support Vector Machine

Non-Linear SVM: Kernel Trick



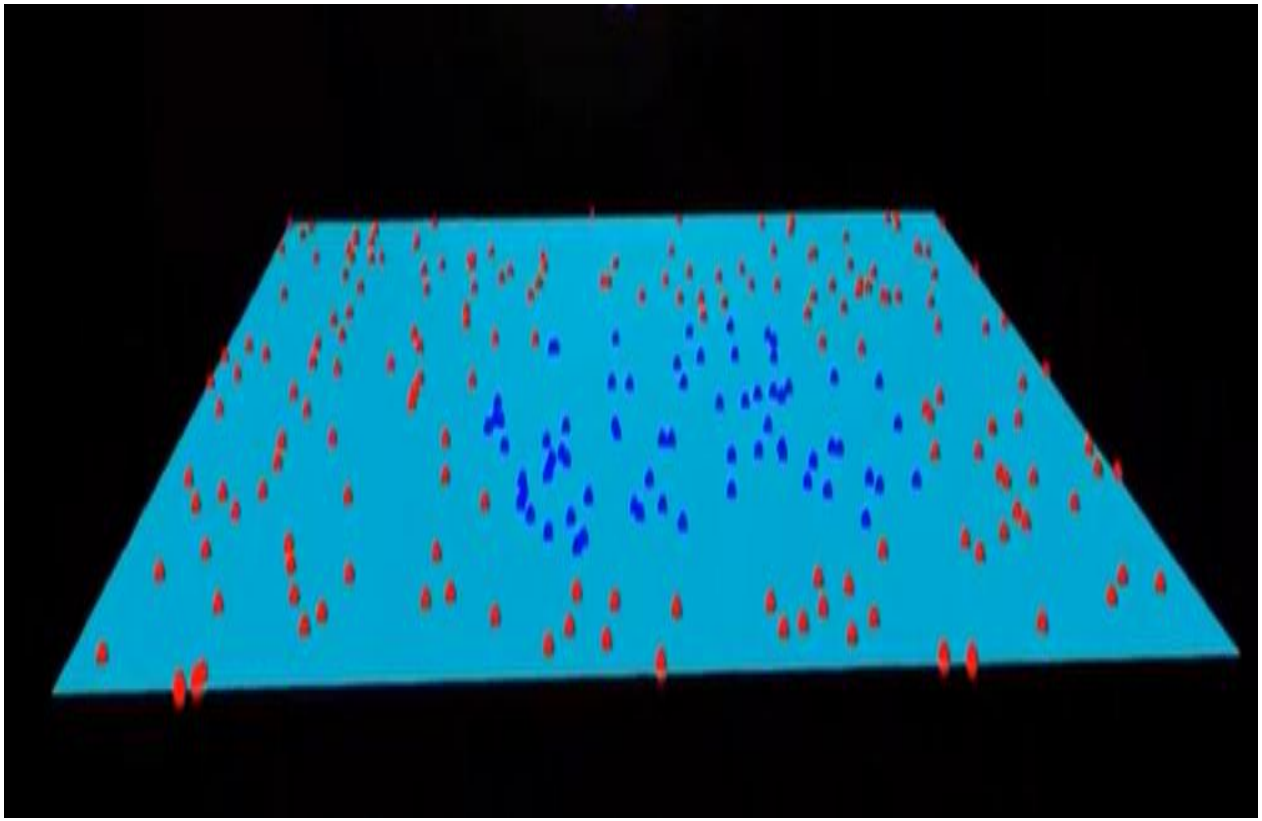




Support Vector Machine

Non-Linear SVM: Kernel Trick

- Example 3:



Watch Video

Support Vector Machine

Non-Linear SVM: Kernel Trick

Common Kernels:

- **1- Linear Kernel:**

$$K(x_i, x_j) = x_i^T x_j$$

Used when data is linearly separable.

- **2- Polynomial Kernel:**

$$K(x_i, x_j) = (x_i^T x_j + c)^d$$

Used for non-linear data with polynomial relationships.

- **3- Radial Basis Function (RBF) Kernel (Gaussian Kernel):**

$$K(x_i, x_j) = \exp\left(-\gamma ||x_i - x_j||^2\right)$$

Most commonly used for non-linear problems.

Hyperparameter γ controls influence of points.

Support Vector Machine

SVM for Multi-Class Classification

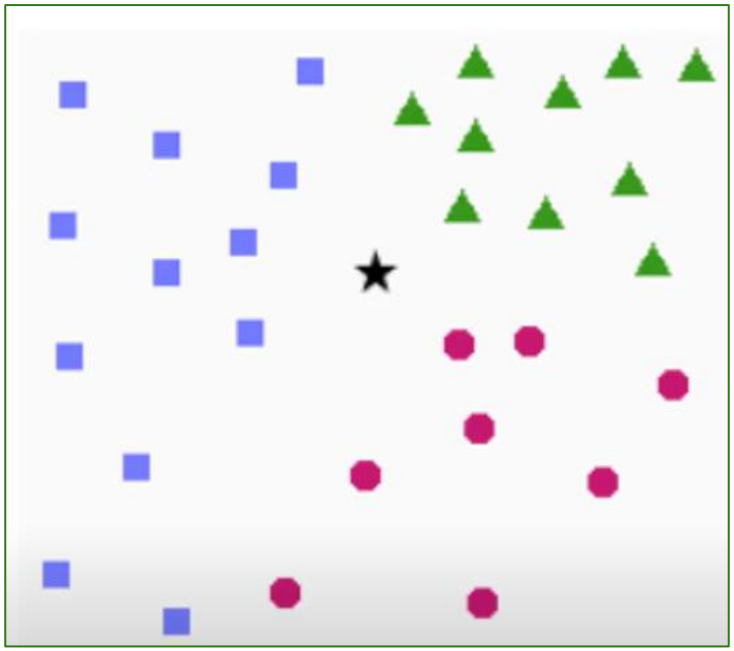
Since SVM is a binary classifier, we need strategies to handle multiple classes:

- **One-vs-All (OvA):**
  - Train N classifiers (one for each class vs. all others).
  - Choose the class with the highest confidence score.
- **One-vs-One (OvO):**
  - Train classifiers for every pair of classes (N × (N-1)/2 models).
  - The final class is chosen by majority vote.

Support Vector Machine

SVM for Multi-Class Classification

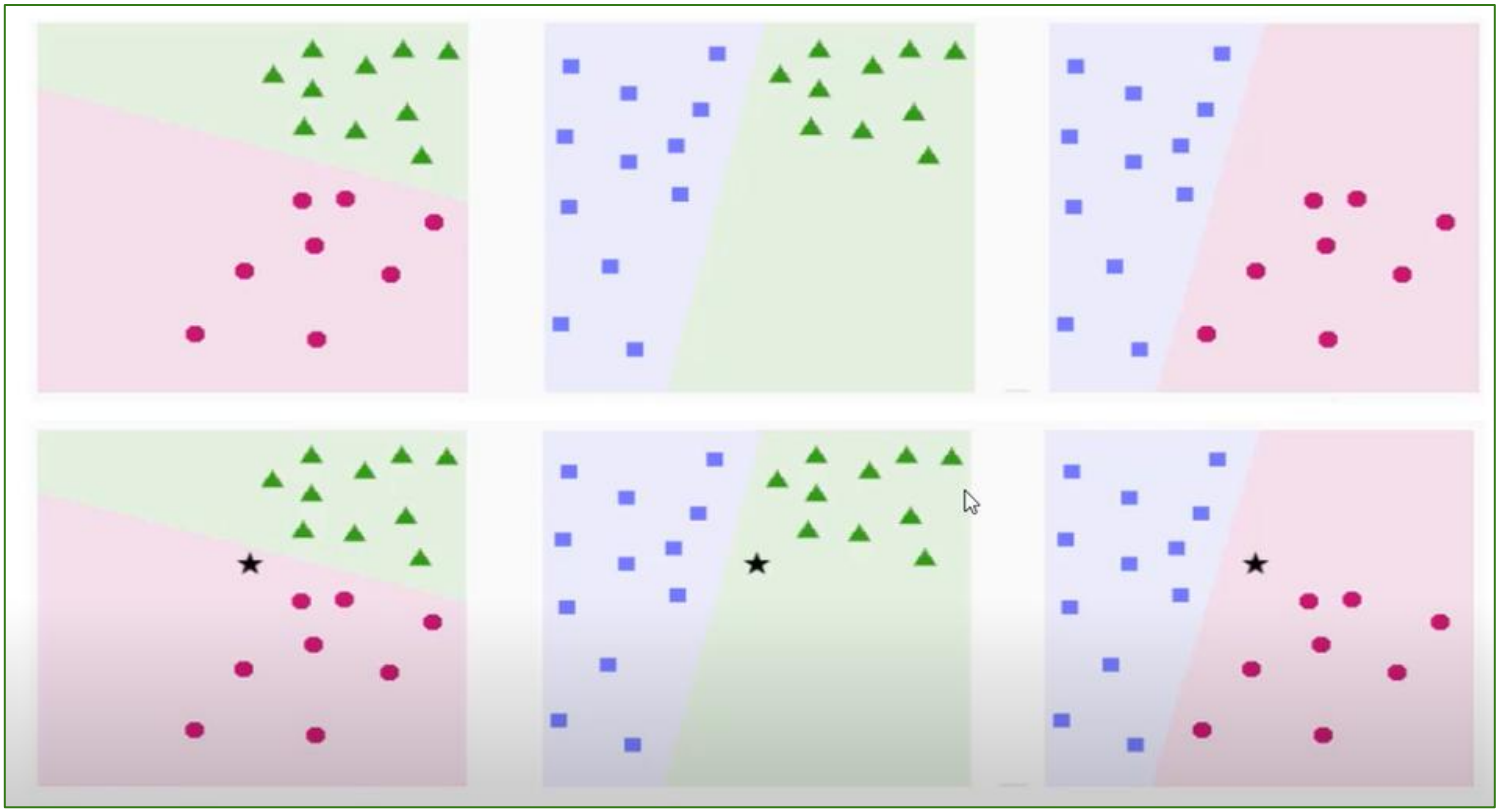
Example : How to use an SVM model for a problem with three classes?



Support Vector Machine

SVM for Multi-Class Classification

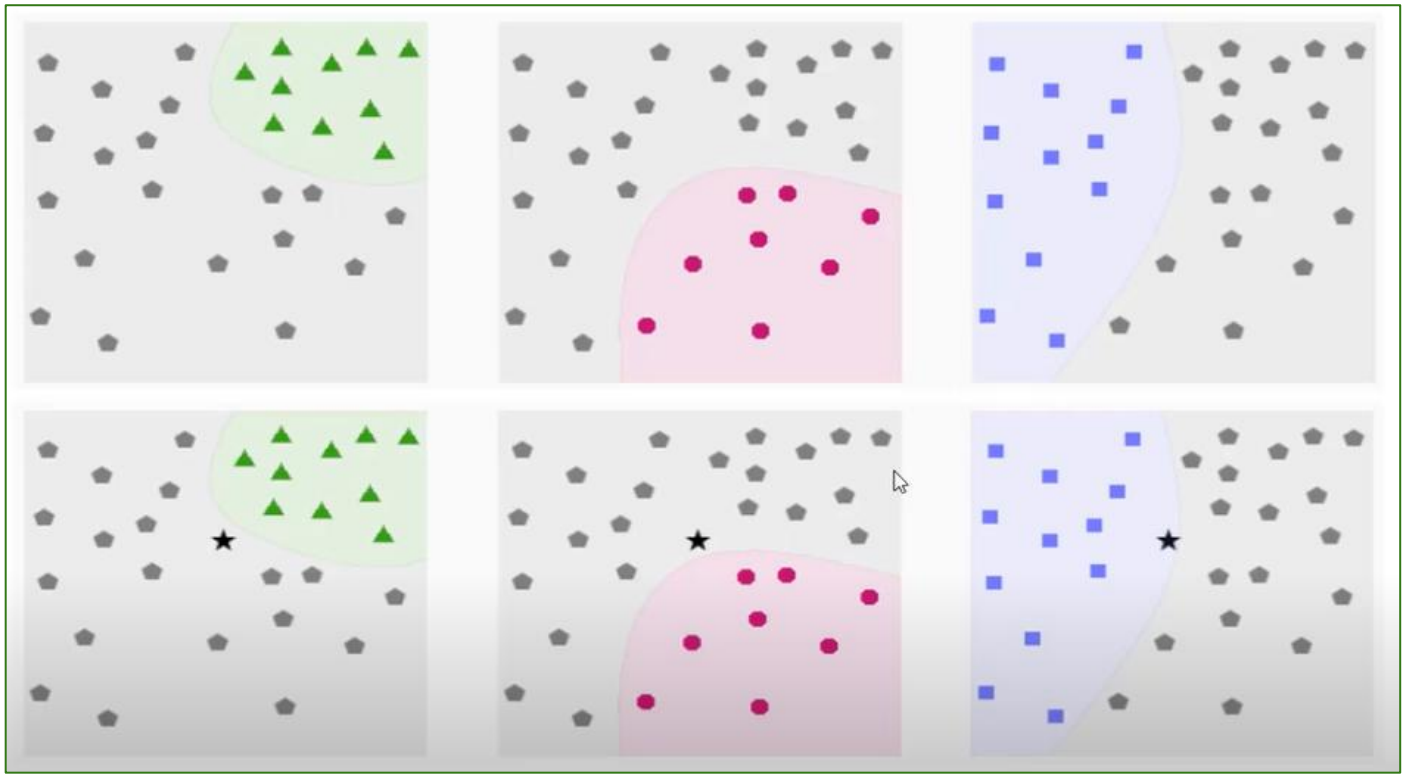
One-vs-One (OvO):



Support Vector Machine

SVM for Multi-Class Classification

One-vs-All (OvA):



Evaluating Model Performance: **Metrics**

Evaluating Model Performance: **Metrics**

Confusion Matrix

- Before diving into classification metrics, it's crucial to understand the **confusion matrix**, which serves as the foundation for most evaluation measures.
- A confusion matrix is a table that describes the performance of a classification model by comparing predicted labels with actual labels.

Evaluating Model Performance: **Metrics**

Confusion Matrix

	Predicted Positive	**Predicted Negative**
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

- **True Positives (TP):** The model correctly predicts positive cases.
- **False Positives (FP):** The model incorrectly predicts positive when it's actually negative.
- **False Negatives (FN):** The model incorrectly predicts negative when it's actually positive.
- **True Negatives (TN):** The model correctly predicts negative cases.

Evaluating Model Performance: **Metrics**

Classification Metrics Based on the Confusion Matrix

- Using the confusion matrix, we define different performance metrics:
- **Accuracy (Overall Correct Predictions)**
- **Precision (Positive Predictive Value)**
- **Recall (Sensitivity, True Positive Rate)**
- **F1-Score (Balance of Precision & Recall)**
- **ROC Curve & AUC (Area Under Curve)**
- **…**

Evaluating Model Performance: **Metrics**

Accuracy (Overall Correct Predictions)

- **Accuracy:**

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

- Measures how many predictions were correct. Problem:
- Can be misleading for imbalanced datasets (e.g., 99% non-poisonous, 1% poisonous → accuracy = 99% even if model never detects poisonous plants).

Evaluating Model Performance: **Metrics**

Precision (Positive Predictive Value)

- **Precision :**

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

- High precision → fewer false positives.
- Useful when FP is costly (The model predicts "poisonous," but the plant is actually safe → Not dangerous, just an unnecessary warning.)

Evaluating Model Performance: **Metrics**

Recall (Sensitivity, True Positive Rate)

- **Recall :**

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

- High recall → fewer false negatives.
- Useful when FN is costly (e.g., missing poisonous plants could be dangerous).

Recall (Sensitivity, True Positive Rate)

- **F1-Score**
- F1-score is the harmonic mean of Precision and Recall:

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

- It balances precision and recall.
- Useful when classes are imbalanced (e.g., detecting rare diseases).
- If false positives and false negatives have different costs, F1-score helps optimize both.

Evaluating Model Performance: **Metrics**

ROC Curve & AUC (Area Under Curve)

- **ROC Curve plots True Positive Rate (Recall) vs. False Positive Rate (FPR):**

$$FPR = \frac{FP}{FP + TN}$$

- **AUC (Area Under Curve)** measures the model's ability to separate classes:
  - AUC = 1 → Perfect classifier
  - AUC = 0.5 → Random guessing
  - AUC < 0.5 → Worse than random

Choosing the Right Metric

Scenario	Best Metric
Balanced dataset	Accuracy, F1-Score
Imbalanced dataset	Precision, Recall, F1-Score, AUC
False Positives costly (e.g., spam filter)	Precision
False Negatives costly (e.g., cancer detection)	Recall
Probabilistic outputs needed	Log Loss

Thank you for your attention…