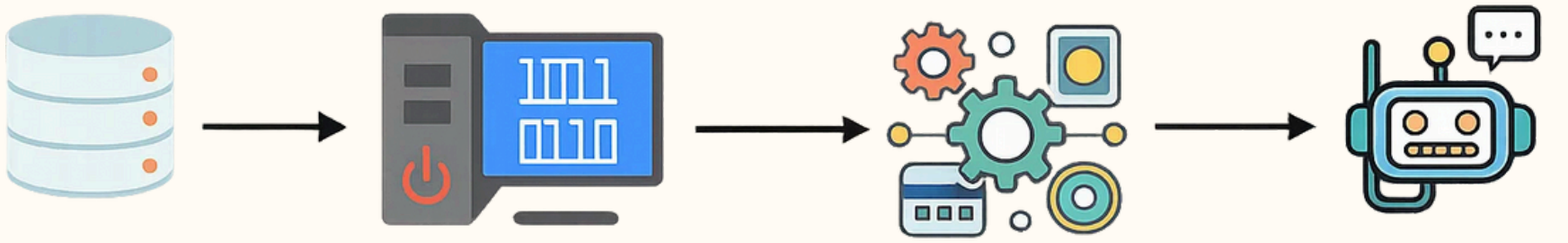
Machine learning



Data

Training the Machine

Build a Model

Rredicting Outcome

Machine Learning (ML) is something we use hundreds of times a day without noticing.

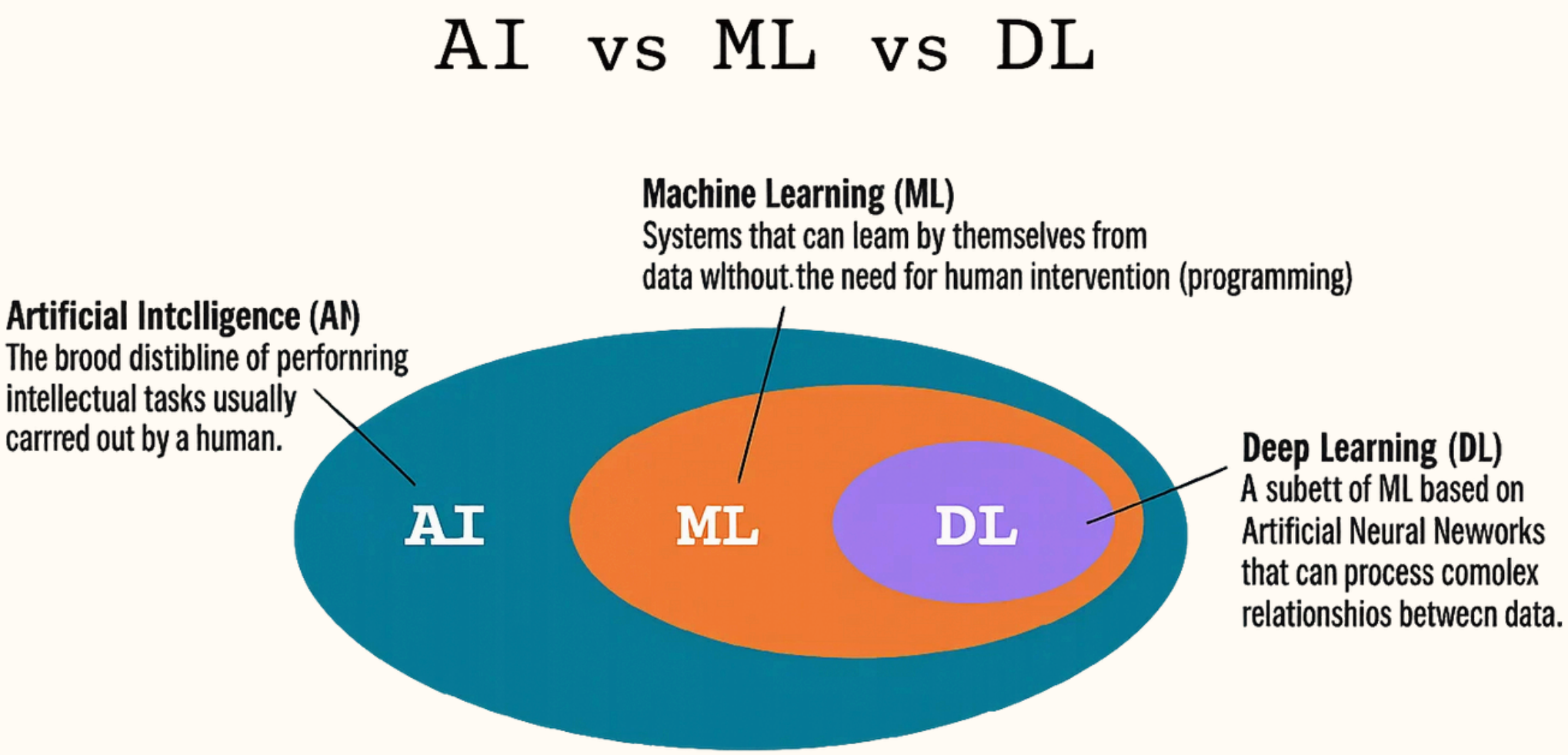
Google Search: ML learns to show the most relevant resuits among billions of possibilities

Facebook: ML recognizes faces in photos to identify people

Netflix/YouTube/Amazon: ML recommends content you are most likely to watch or buy

- **With classical programming, this would be impossible because we'd have to code billions of rules.**
- **ML allows a machine to learn by itself, without explicit programming.**

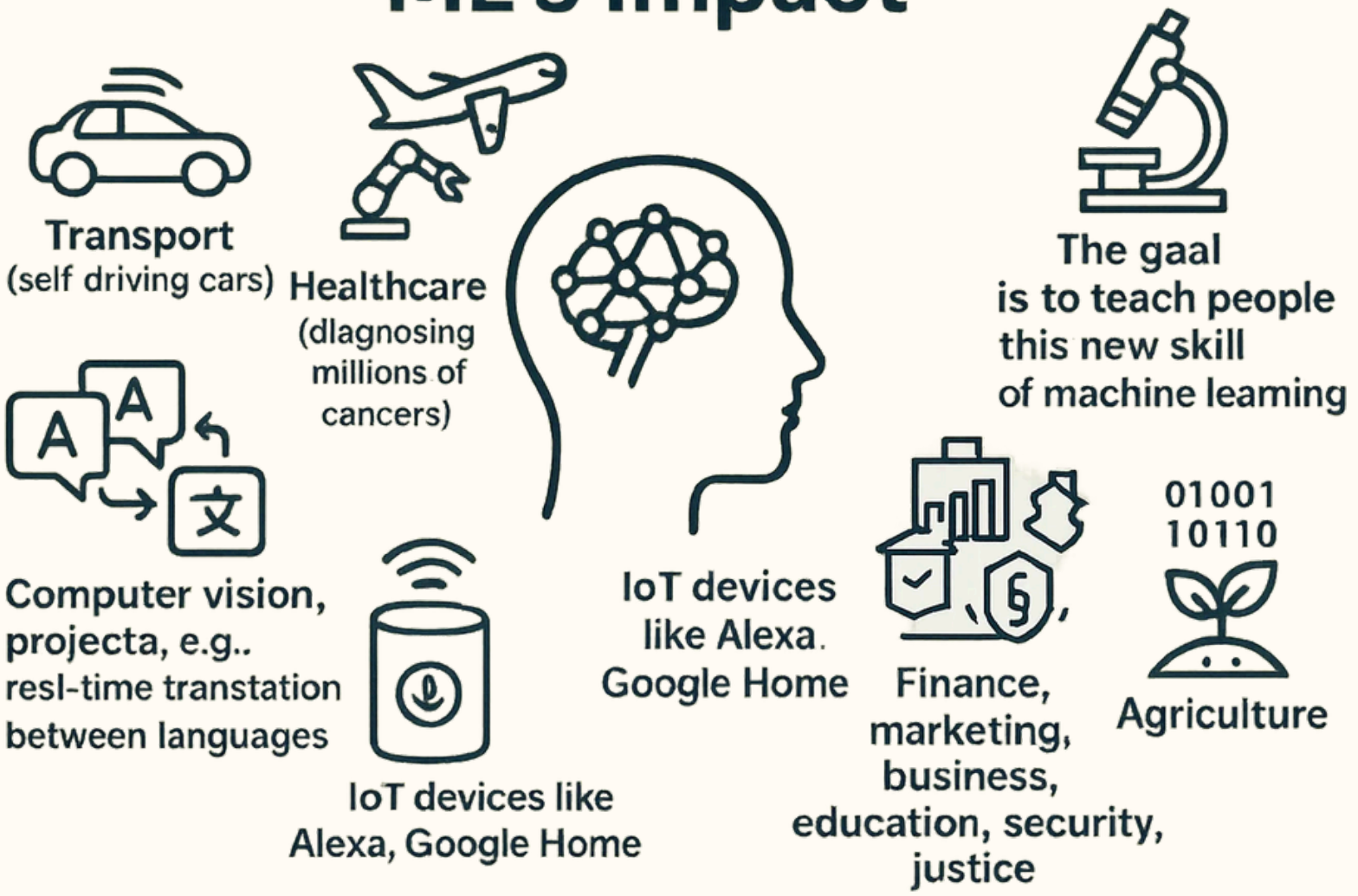
Machine learning vs. artificial intelligence



Aritficial Neural Networks (ANNs)
An approach to lUL based on a model of how the brain works, sametimes tust called neural nelworks.

Natural Language Processing (NLP)
Methods used to get computers to understand human seeech and text, usually involving ML.

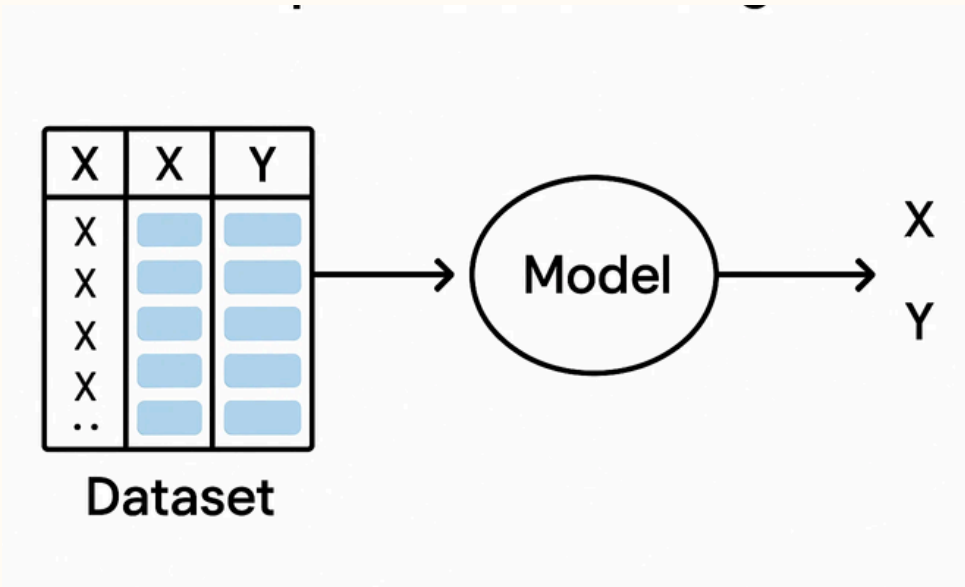
ML's Impact



How Humans Learn vs ML

Humans often learn from examples: e.g., learning Chinese with a book or a tutor.

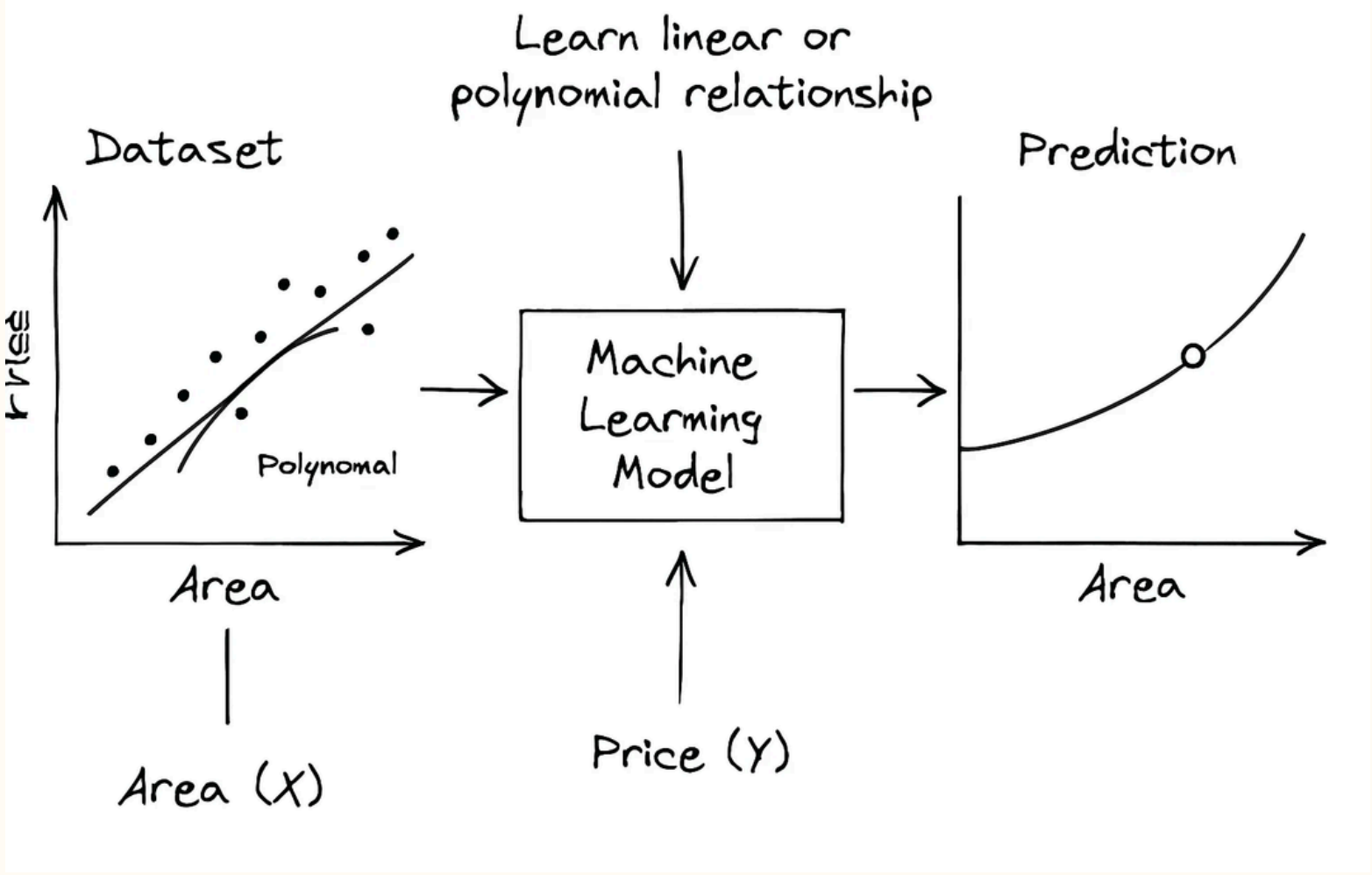
- ML supervisé (supervised learning) mimics this: you give the machine examples (dataset) and it creates a model.
- Dataset = table of data (like Excel).
- The machine learns the relationship between input X and output Y.



Example of Supervised Learning

Regression problem: predicting a continuous variable.

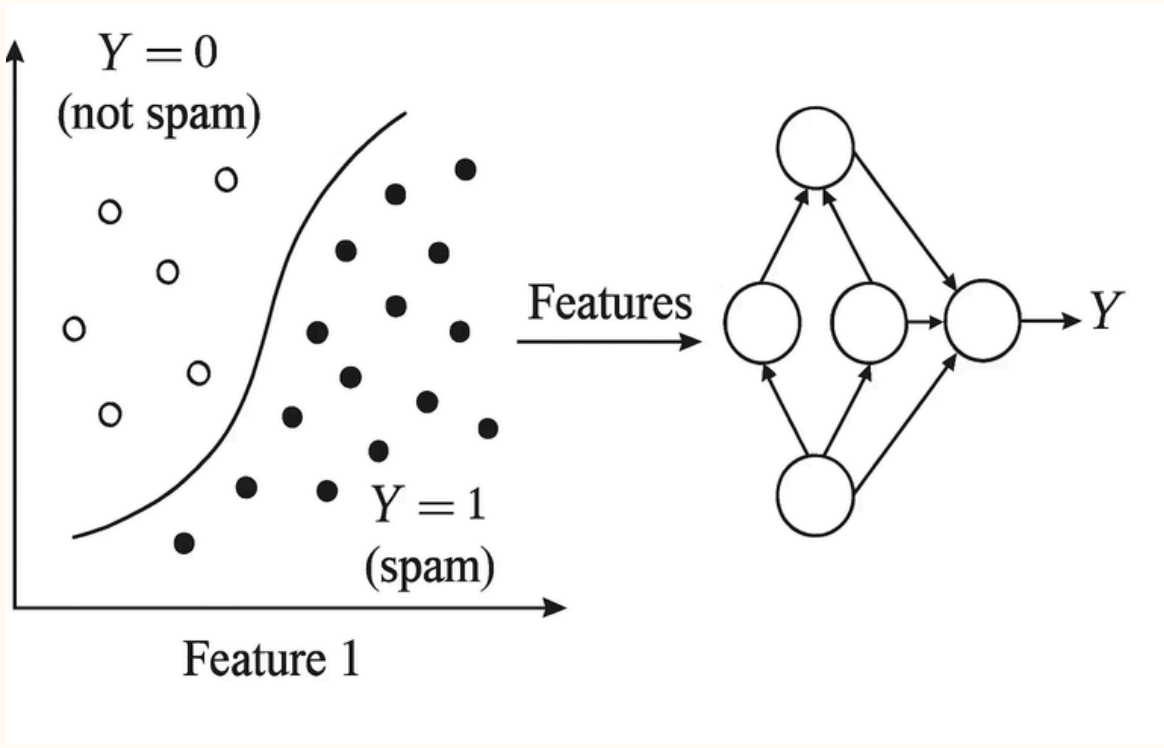
predicting apartment price from its area.
X = area, Y = price.
The ML model can learn linear or polynomial relationships from the dataset.



Classification problem: predicting a discrete variable.

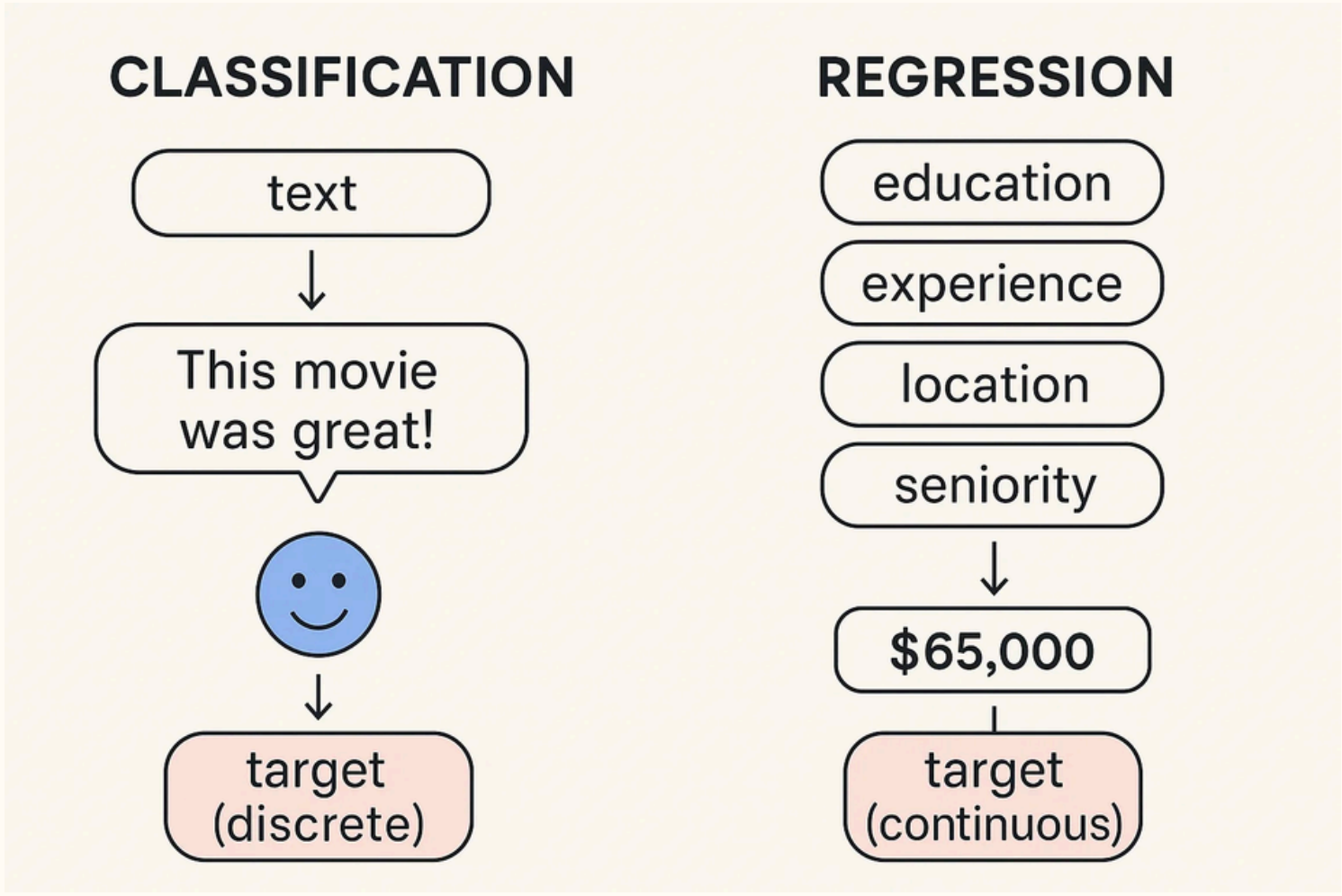
spam email filter.

- Y = 0 (not spam), Y = 1 (spam).
- Features may include number of spelling mistakes, number of links, etc.
- The ML model creates a decision boundary to separate classes.



Machine Learning Classification Vs. Regression

- **Classification** answers: “Which group/class does this belong to?”
- **Regression** answers: “What exact value/number should I predict?”



Example

1. Classification

Target variable (output): A category (discrete).

Example task: Predict whether the student will pass or fail the exam.

✓ Pass or Fail

Pass or Fail

(2 classes)

2. Regression

Target variable (outputi: A number (continuous.)

Example task: Predict the exact exam score (e.g., 78.5 out of 100).

✓ 78,5

(a cntinuous value)

- **Regression: continuous values (price, size).**
- **Classification: discrete categories (spam/non-spam, red/blue).**

Une des stratégies que l'on peut utiliser est l'apprentissage supervisé, qui consiste à montrer à la machine des exemples (X, Y) et à lui demander de trouver l'association qui relie X à Y, c'est-à-dire une association Y = f(X).
Ces exemples que l'on montre à la machine sont regroupés dans un dataset, et c'est là no

dataset

En apprentissage supervisé, un dataset contient toujours deux types de variables :

- **La target variable (Y) :** c'est l'objectif, ce que l'on veut que la machine apprenne à prédire.

Exemples : le prix d'un appartement, le cours de la bourse, ou encore identifier si un email est un spam ou non spam.

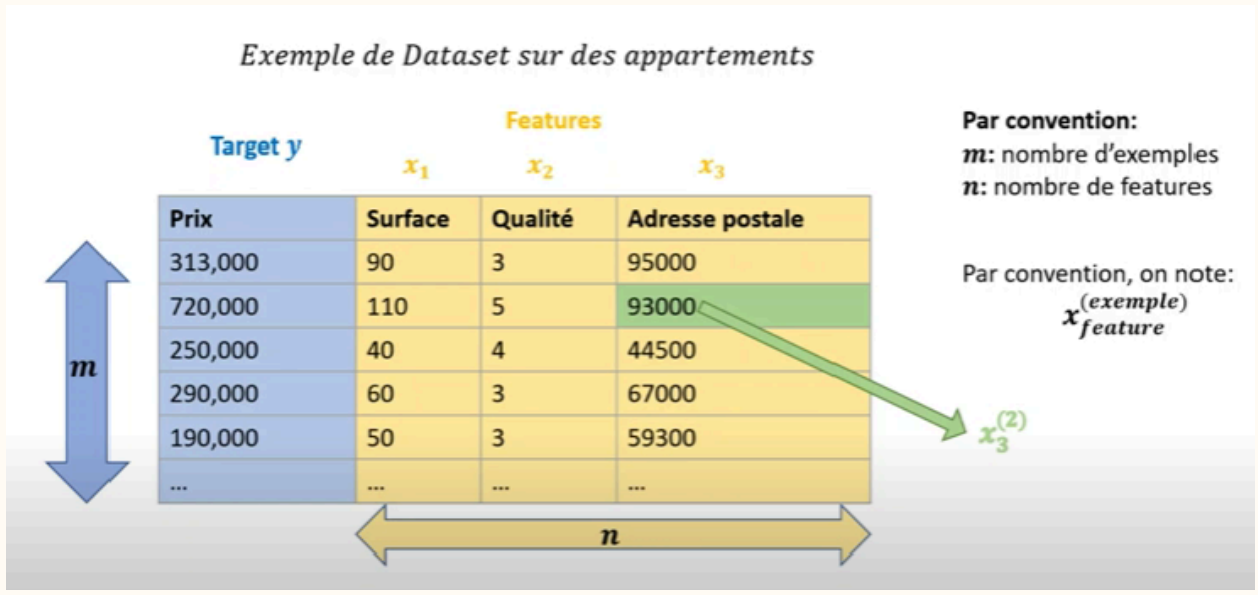
- **Les features (X) :** ce sont les facteurs qui influencent la valeur de Y.

Exemple :

- X_1 = la surface de l'appartement
- X_2 = la qualité de l'appartement
- X_3 = l'adresse postale
- etc.

Conventions en machine learning :

m = le nombre d'exemples dans le dataset (c'est-à-dire le nombre de lignes).
n = le nombre de features dans le dataset (c'est-à-dire le nombre de colonnes, en excluant la colonne Y).



Representation using vectors and matrices

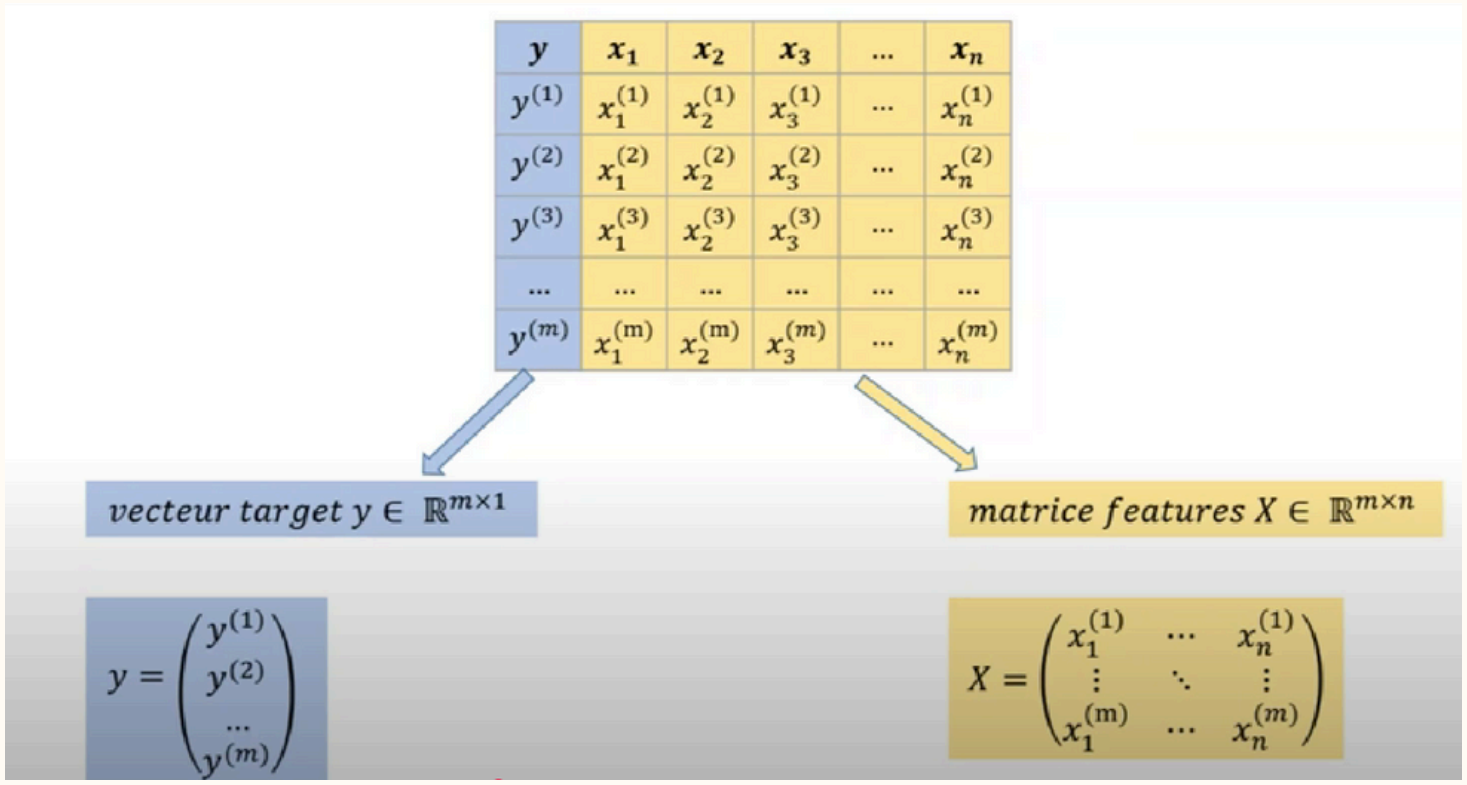
When you do machine learning, you will always see datasets that look like this:
We can represent the data as vectors and matrices:

- **The target vector Y:**

This is a vector with m rows, i.e., m elements (one for each example).

- **The feature matrix X:**

This is a matrix of dimension m × n (m rows = examples, n columns = features).



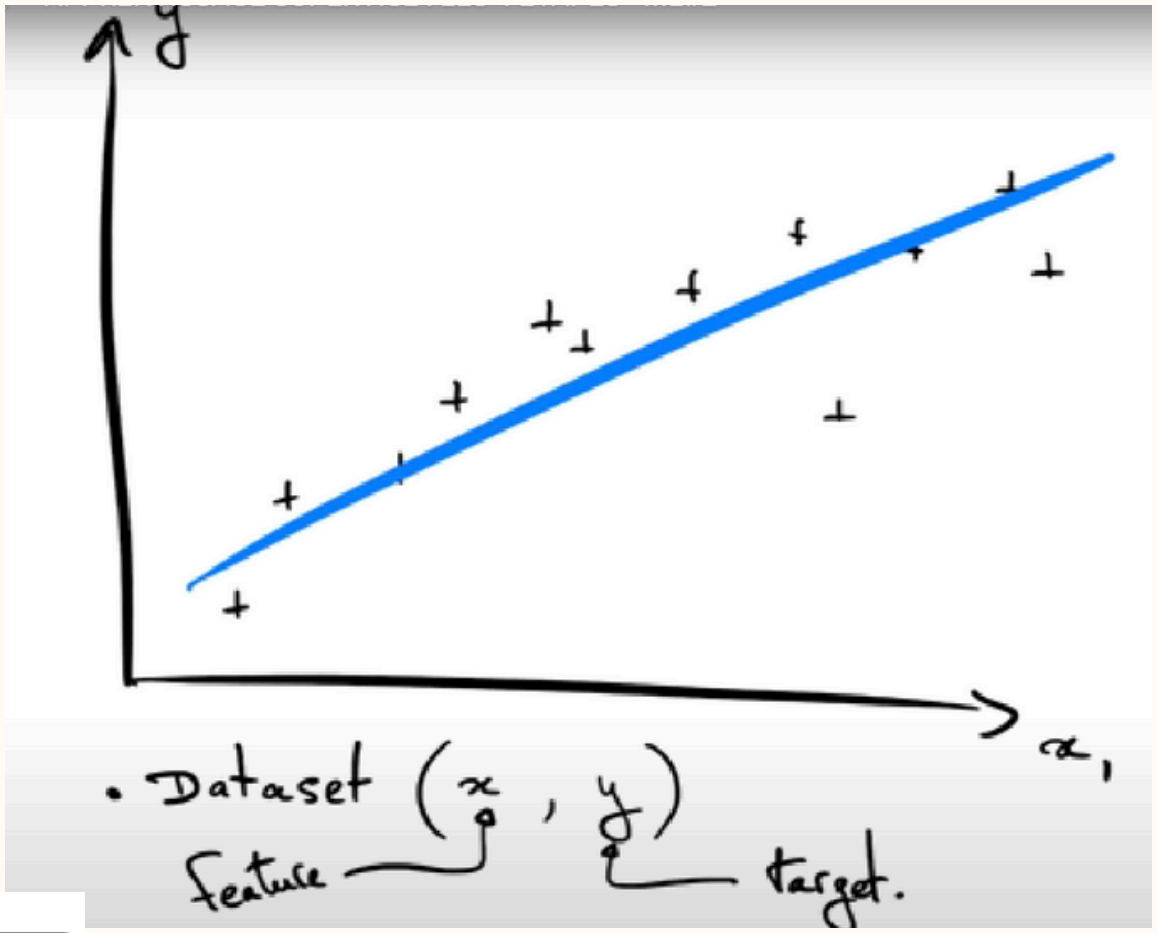
These vectors and matrices are fundamental, because they will be used throughout machine learning.

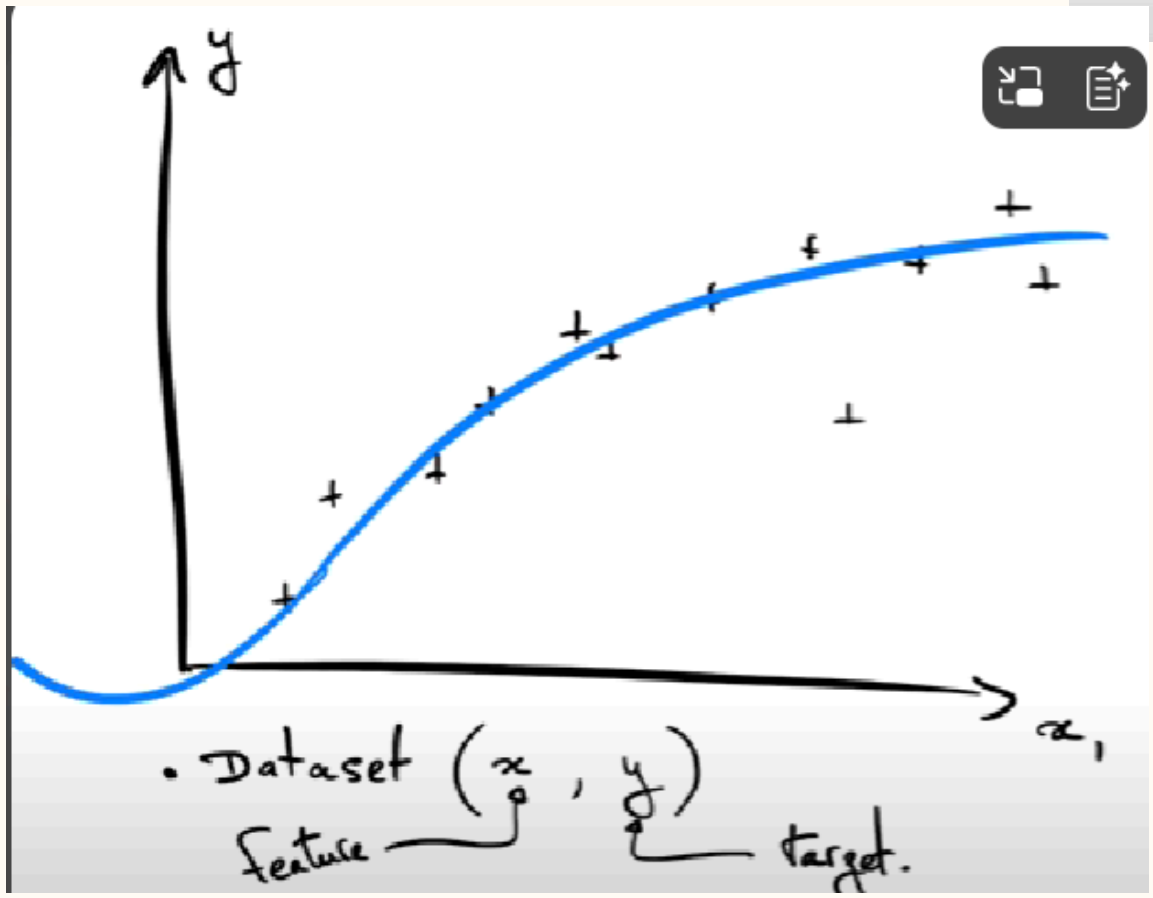
Model

From the dataset, we can visualize a scatter plot, and we might say, “Well, we could create a model.”
For example, a linear model:

- **f(x)=ax+b,**

a simple affine function like the ones we studied in middle school.





But we could also say, “No, maybe a polynomial model would fit the scatter plot better.”
In that case, it could be a 2nd-degree polynomial:

- **f(x)=ax2+bx+c,**

—or even a 3rd-degree polynomial:

- **f(x)=ax3+bx2+cx+d**

The idea is: we have a model, and this model has what we call parameters.
These parameters are the coefficients of the polynomial (a,b,c,da, b, c, da,b,c,d, etc.), as many coefficients as needed.
So these are called the parameters, which is our second fundamental notion in supervised learning:

- t**he model, and**
- **the parameters.**

It is important to point out that we decide which model the machine must use,
but it is the machine that learns the parameters of this model.

Loss function

Another fundamental notion in supervised learning is the concept of the **loss function (or cost function).**

This model that we use in relation to our dataset produces errors.

For example, let's consider an apartment with a surface area of 150 m².

In our dataset, we have a point showing that this apartment was worth around 200,000 euros.

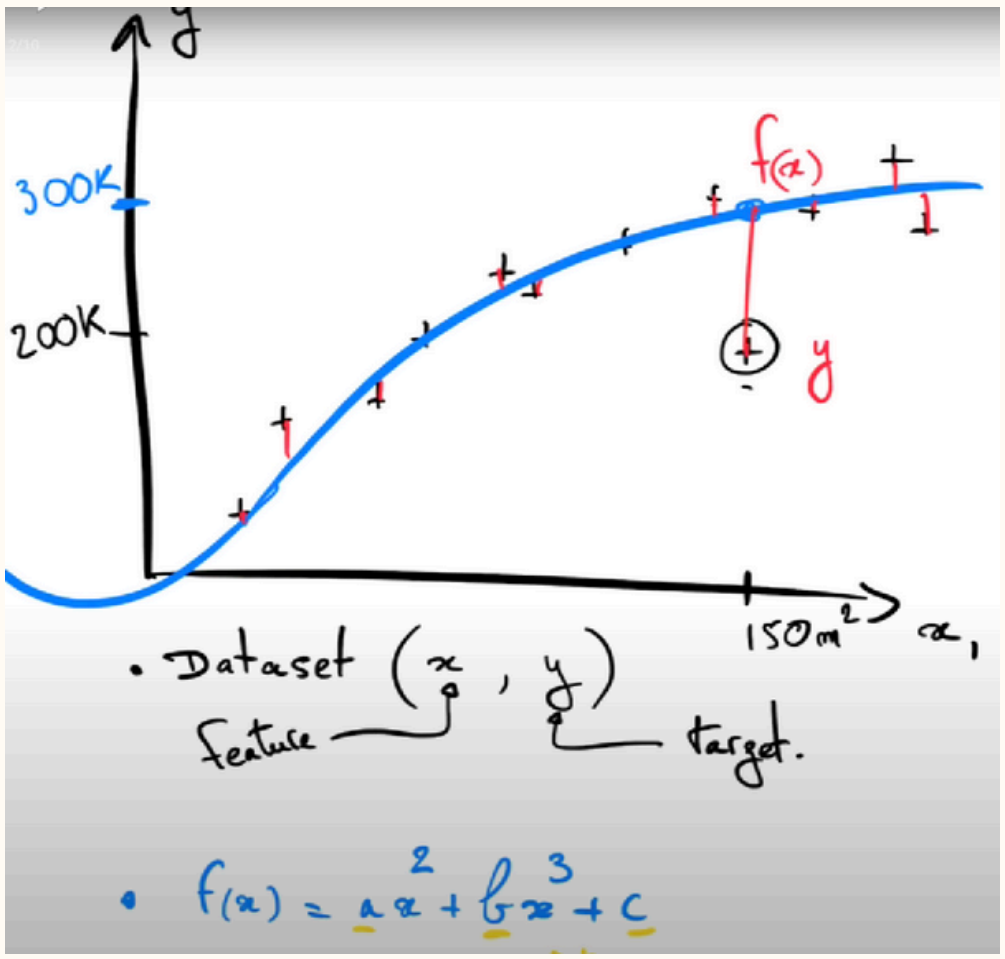
But **since our model is a generalization of all the points available in the dataset,**

it predicts a much higher value — maybe around 300,000 euros.

Well, **we then say there is an error between this known data point (the true value) and our prediction f(x)**.

T**his error** can be found for every point in the dataset.

And when we gather all these errors together, we define what's called the loss function (or cost function).



So, having a good model means having a model that produces small errors.

Logically, this leads us to the 4th fundamental notion in supervised learning,

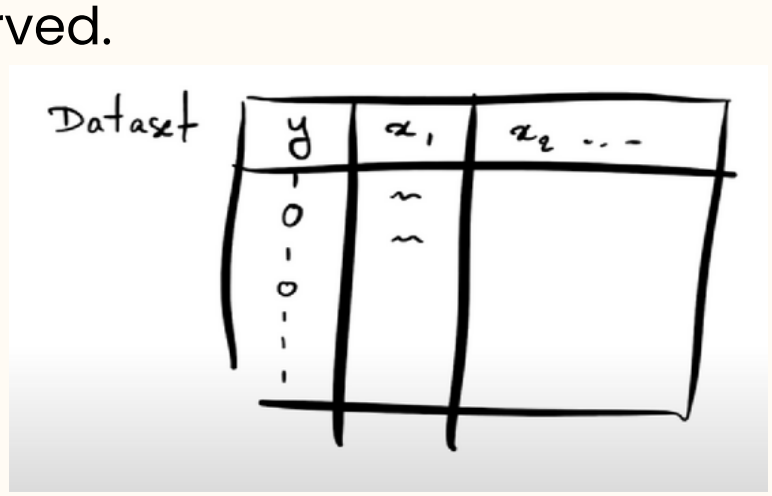
supervised learning

In machine learning, we develop a strategy that tries to find the parameters a,b,c,d,...a, b, c, d, ...a,b,c,d,...
that minimize the loss function — in other words, that minimize all of our errors.
To minimize this loss function, we use a learning algorithm.
There are many of them, but one of the most famous is the **gradient descent algorithm,**

Now, I've illustrated these four notions with an **example of a regression problem.**
But remember, in machine learning, there are also classification problems,
and in classification, we find the exact same notions.

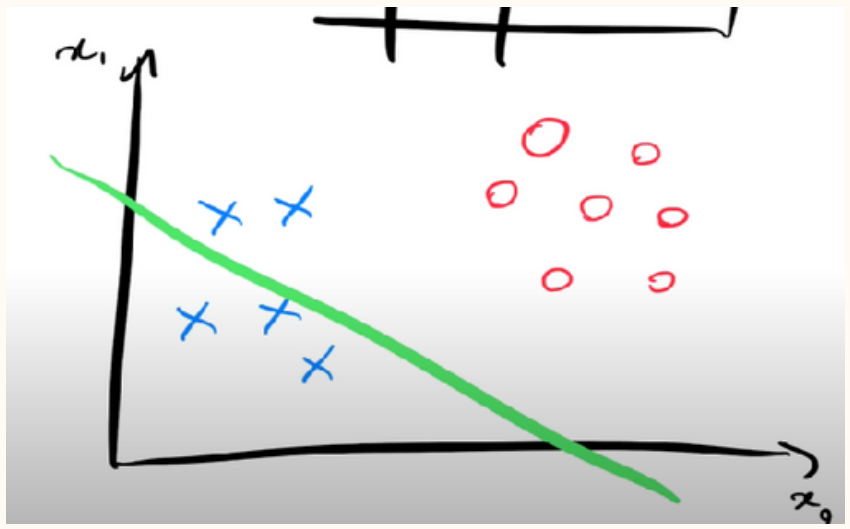
For example, suppose you want to predict whether a cell is cancerous or not.
You'll gather a dataset of cells that have been observed.

- Some will be labeled as cancerous (y=1),
- others as non-cancerous (y=0).

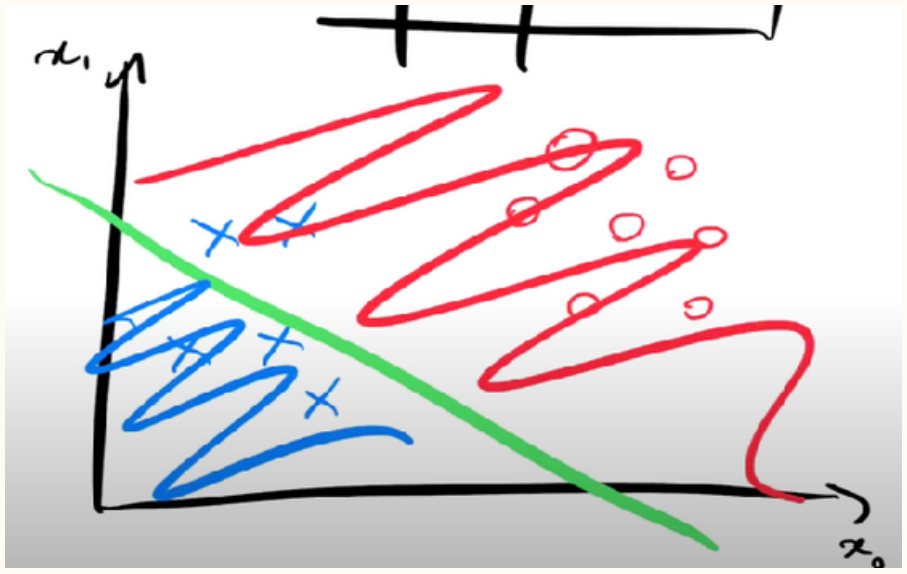


You'll measure different features of the cells —
maybe one factor is the size of the cell,
another factor is the size of the nucleus, etc.

Then, you'll associate these features with the examples and feed them into a model with parameters.
At first, the parameters are completely random.
So the model might initially say:
"Everything on this side is blue (non-cancerous), everything on that side is red (cancerous)."

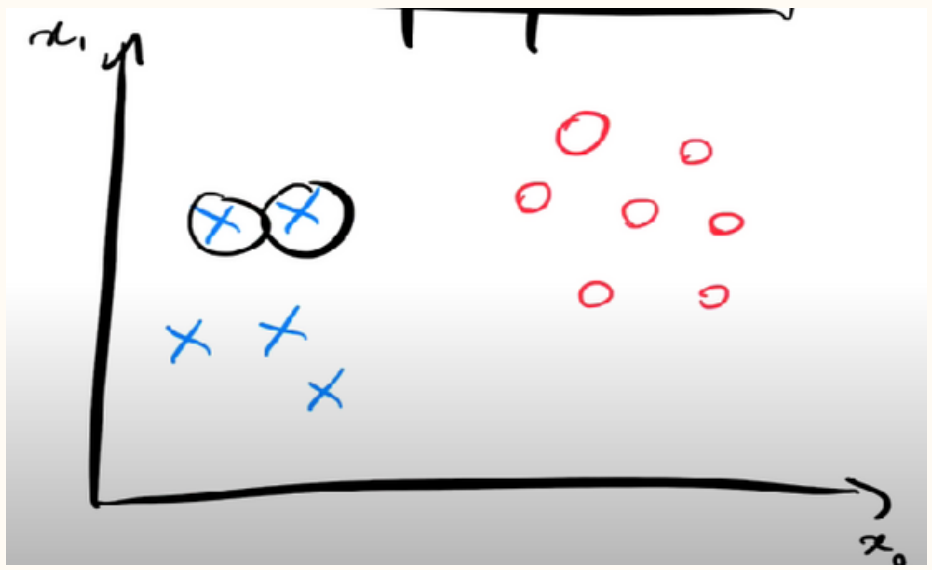


But the machine will see that its loss function is high,
because it misclassified some cells.
For example, two non-cancerous cells may have been wrongly classified as cancerous.



So, it will then adjust the parameters in the model in order to reduce the loss function.

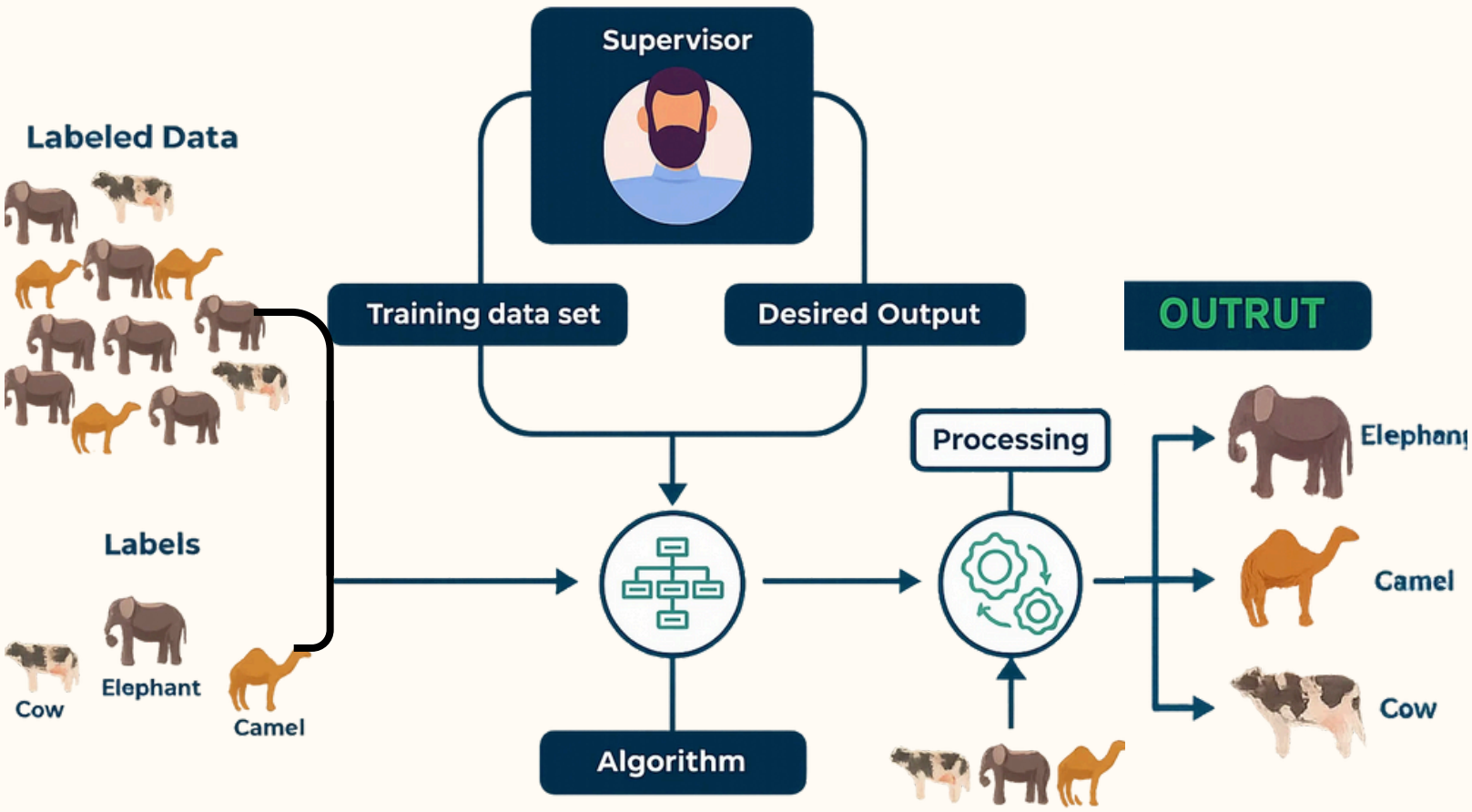
Eventually, it will find a model that looks more like this:
"Everything on this side is blue, everything on that side is red."
And in doing so, it has learned how to predict whether a cell is cancerous or not.



So what you really need to remember are the four fundamental notions in supervised learning:

- **Dataset (inputs X and outputs y)**
- **Model (with parameters)**
- **Loss function (to measure errors)**
- **Learning algorithm (to minimize the loss by adjusting parameters)**

Supervised Learning

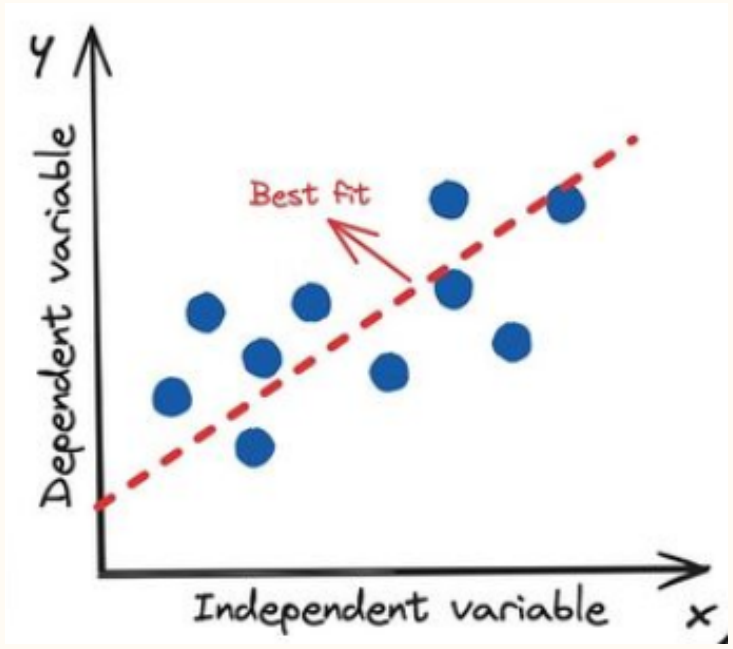


It is a type of machine learning that trains the model using labeled datasets to predict outcomes.

Key Algorithms You Should Know

- **Linear Regression**

It's a way to find a straight line that best describes the relationship between two variables:
X → independent variable (the input, e.g., hours studied)
Y → dependent variable (the output, e.g., exam score)
👉 The goal: predict Y from X using a straight line.

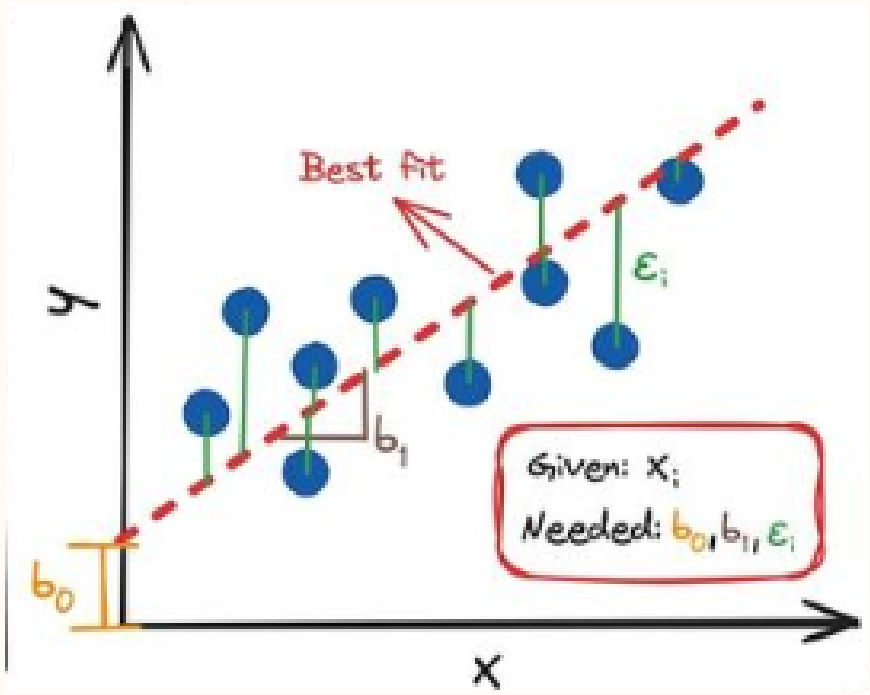


The Formula

The line looks like this:

- $Y = b_0 + b_1X + e$

b_0 → intercept (where the line crosses Y-axis)
b_1 → slope (how much Y changes when X increases by 1)
e → error (the difference between prediction and actual value)



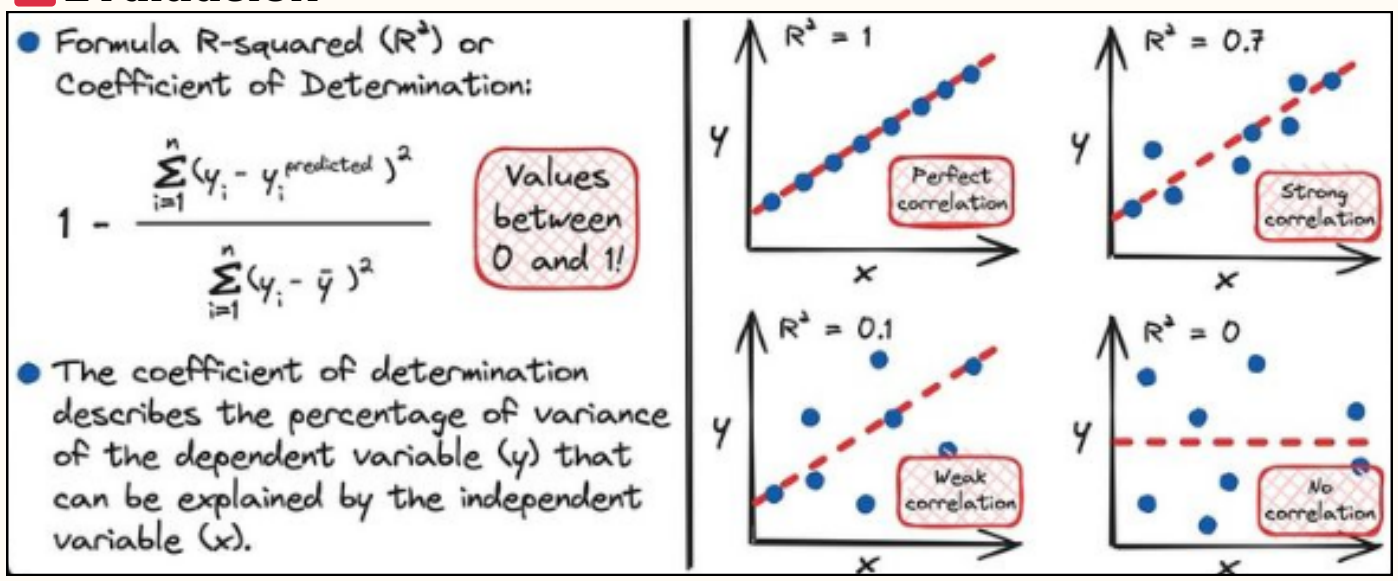
Key Ideas

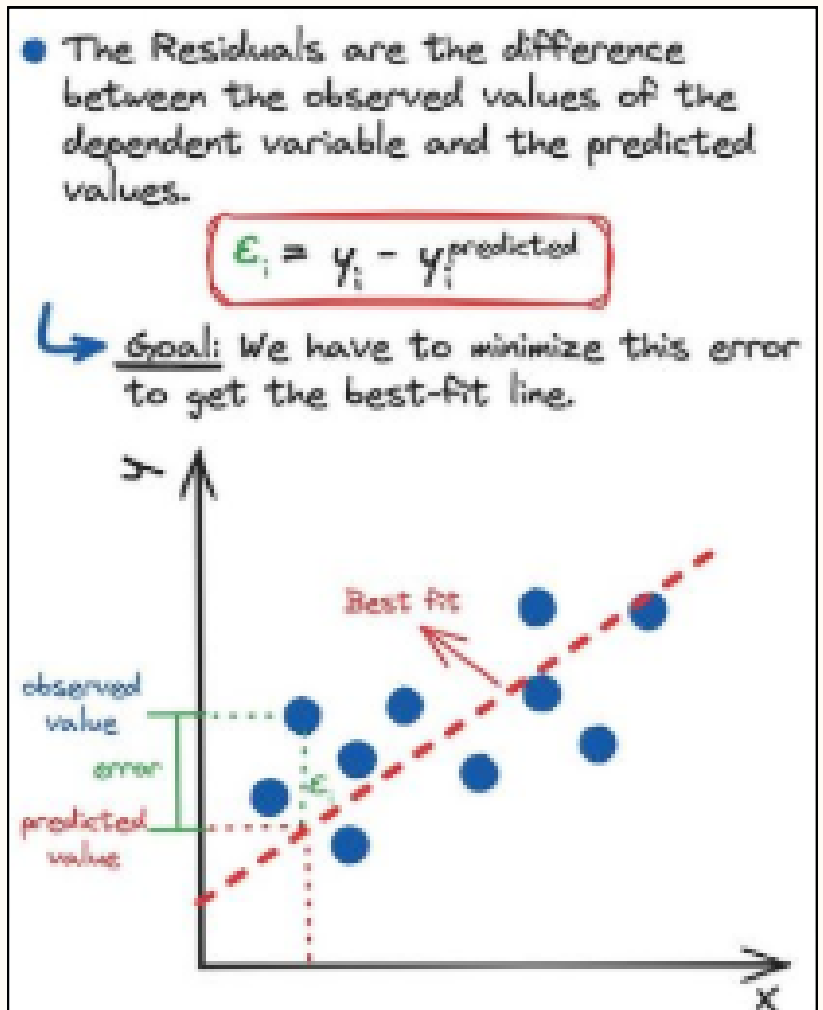
1. **Best Fit Line** → We draw the line that's "closest" to all points. (Not perfect, but best overall).
2. **Residuals (errors)** → The gap between actual Y and predicted Y.
   - Smaller gaps = better model.
3. **Slope & Intercept** → Found by minimizing the total squared errors (method called Least Squares).

Assumptions (when it works well)

- X and Y have a linear relationship.
- Errors (residuals) are random and not biased.
- Errors spread equally (not growing bigger with X).
- Errors roughly follow a normal distribution.

Evaluation



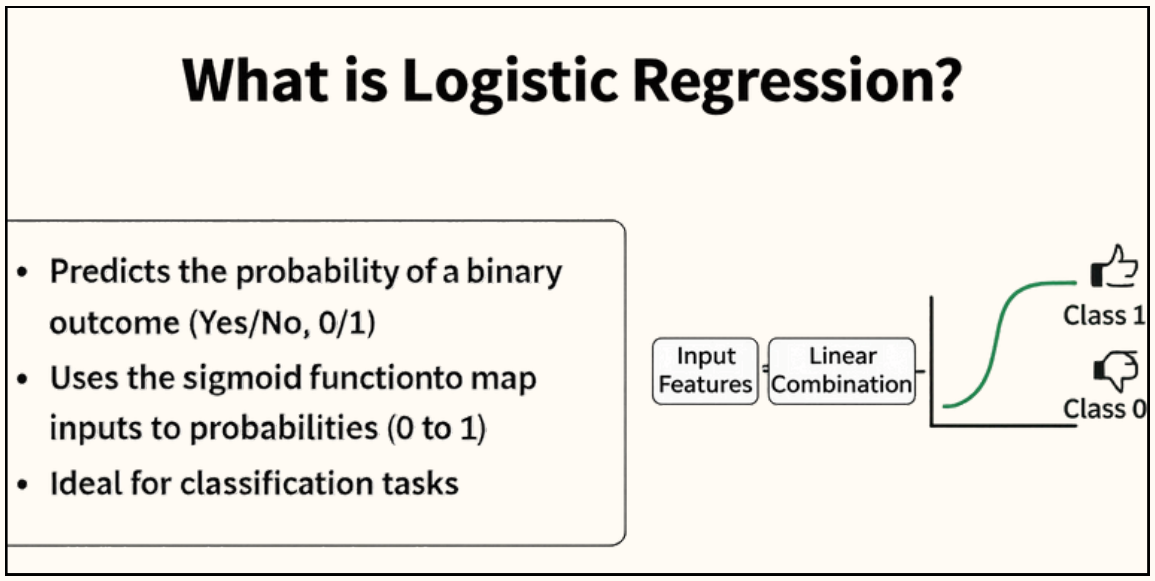


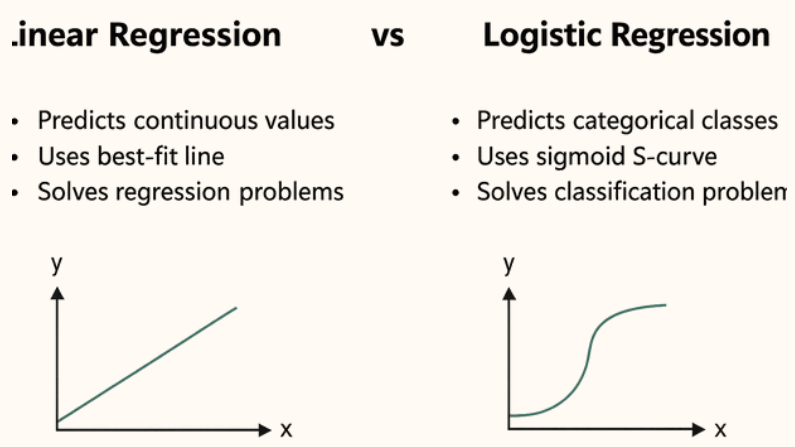
✅ **Pros:**
Easy to understand.
Fast to compute.
Works well on simple data.

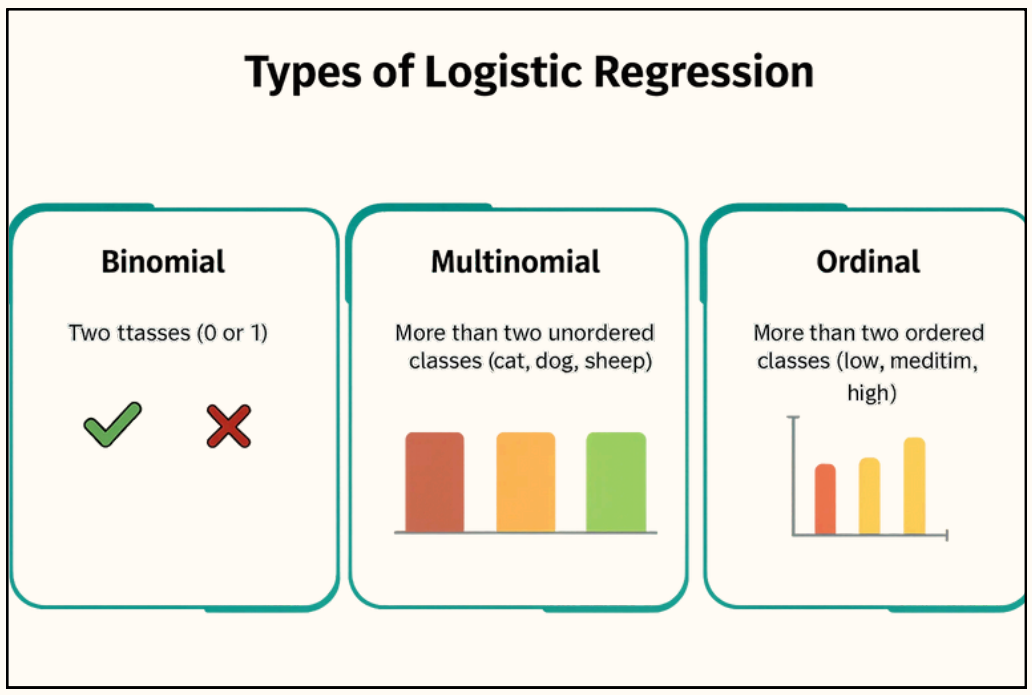
❌ **Cons:**
Relies on assumptions.
Doesn't work well for complex or nonlinear data.
Sensitive to outliers (strange points can distort the line).

• Logistic Regression

Logistic Regression is needed when the output is categorical (0/1).







- **Binomial Logistic Regression:** This type is used when the dependent variable has only two possible categories. Examples include Yes/No, Pass/Fail or 0/1. It is the most common form of logistic regression and is used for binary classification problems.
- **Multinomial Logistic Regression:** This is used when the dependent variable has three or more possible categories that are not ordered. For example, classifying animals into categories like "cat," "dog" or "sheep." It extends the binary logistic regression to handle multiple classes.
- **Ordinal Logistic Regression:** This type applies when the dependent variable has three or more categories with a natural order or ranking. Examples include ratings like "low," "medium" and "high." It takes the order of the categories into account when modeling.

- **Data points are separate:** Each example shouldn't affect the others.
- **Outcome is yes/no:** The result can only be one of two options.
- **Predictors affect chances linearly:** Inputs change the odds in a straight-line way (on a log scale).
- **No extreme values:** Very unusual numbers can mess up the model.
- **Enough data:** More examples give better, stable results.

Sigmoid Function Explained

- **Formula:**

$\sigma(z)=1/1+e-z$

- **What it does:**

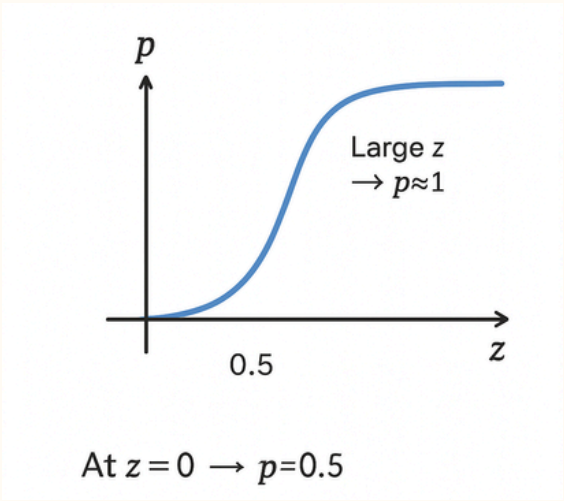
It takes any real number z (from $-\infty$ $+\infty$)
And squeezes it into a range between 0 and 1.

- **Why it's useful:**

Probabilities must be between 0 and 1.
The sigmoid gives a smooth "S-shaped" curve to map values nicely into probabilities.

Key points:

If z=0z , then $\sigma(0)=0.5$
If z is a very large positive number, $\sigma(z)\approx1$
If z is a very large negative number, $\sigma(z)\approx0$



Workflow of Logistic Regression (Architecture)

- **Input Features (X)**

These are your independent variables (e.g., age, income, hours studied, etc.).
They go into the model as a feature vector [x_1, x_2, ..., x_n]

- **Weighted Sum (Linear Combination)**

Each feature gets multiplied by a weight (w), and a bias/intercept term b is added.
Mathematically:
$z=w \cdot X+bz$
This is the linear regression part.

- **Sigmoid Function (Logistic Function)**

The output zzz from the linear step is fed into the sigmoid function:

- $\sigma(z)=1/1+e-z$

This "squashes" the continuous value into a probability between 0 and 1.

- **Probability Output**

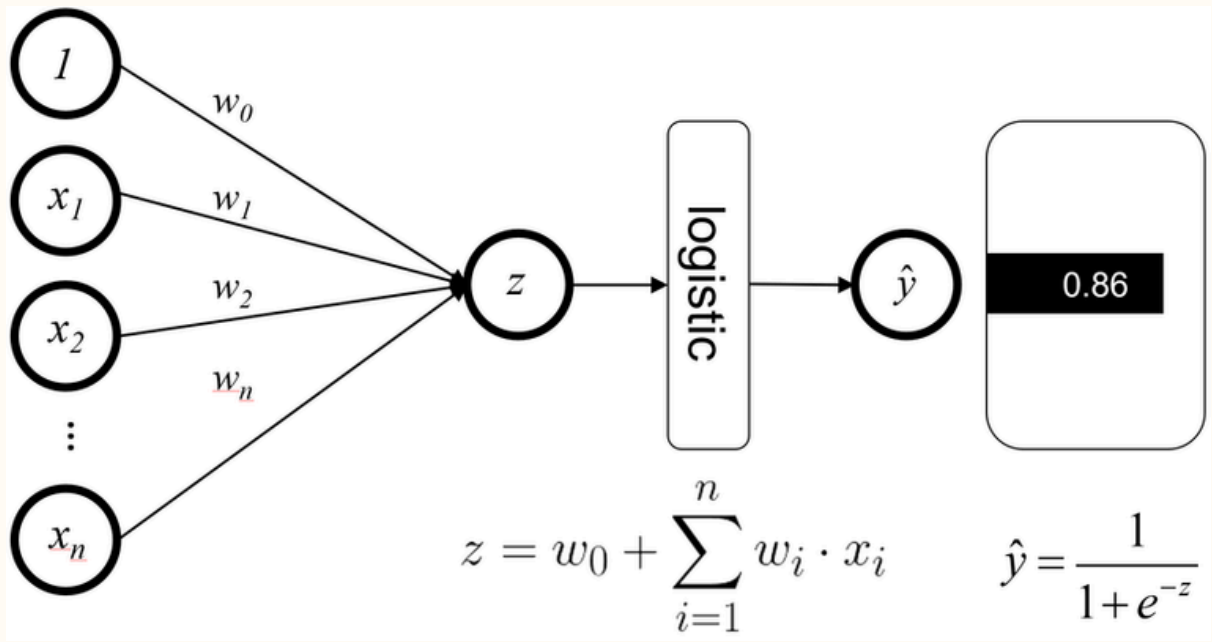
The sigmoid output gives the probability of belonging to Class 1.
Example:
If $\sigma(z)=0.85$ the model predicts an 85% chance of being Class 1.

- **Decision Boundary (Classification Rule)**

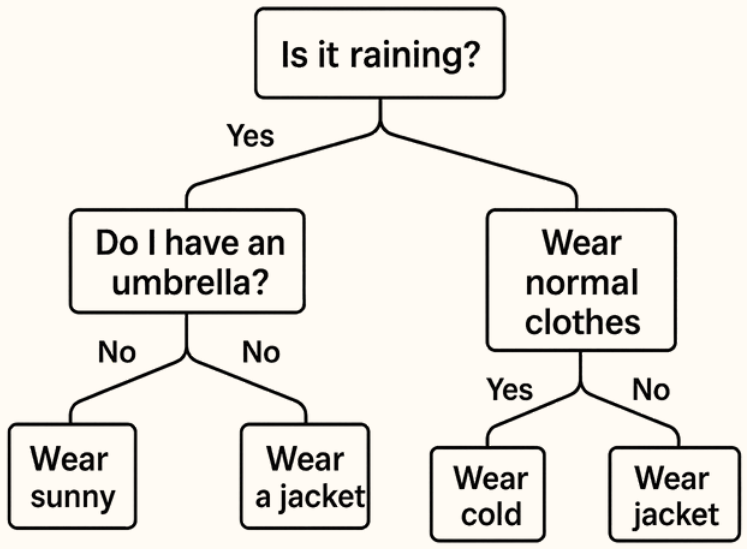
We apply a threshold (usually 0.5):
If $\sigma(z) \geq 0.5 \rightarrow$ Predict Class 1.
If $\sigma(z) < 0.5 \rightarrow$ Predict Class 0.

- **Training (Learning the Parameters)**

The model adjusts weights (w) and bias (b) using:
Maximum Likelihood Estimation (MLE).
Optimization via Gradient Descent.
Goal: Maximize how well the predicted probabilities match the actual labels.



• Decision Tree

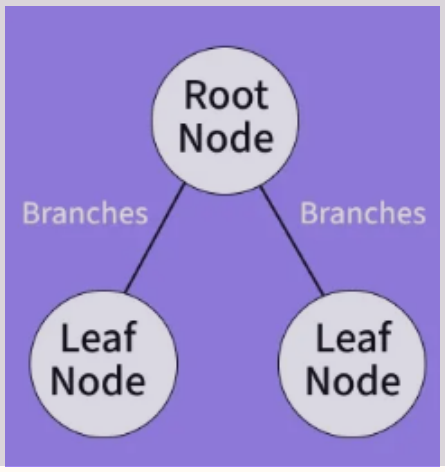


A **Decision Tree is** a method of making predictions by asking a series of simple questions about your data, following the answers step by step until you reach a decision.

is a supervised machine learning algorithm used for classification and regression tasks.
It works like a flowchart, where each node represents a decision based on a feature, and each branch leads to either another decision or a final outcome (leaf).

What This Analogy Means for a Decision Tree

- Features = Questions (rain, cold, umbrella)
- Branches = Answers (Yes/No)
- Leaves = Final decision (jacket, light clothes, raincoat)



Key Concepts

Information Gain

Measures how much an attribute helps to reduce uncertainty.
Choose the attribute that gives maximum information gain.
Formula:

$$Gain(S, A) = Entropy(S) - \sum_{v} \frac{|S_v|}{|S|} Entropy(S_v)$$

Where:

- S = all training examples
- A = an attribute
- Sv = subset of examples for value v of attribute A

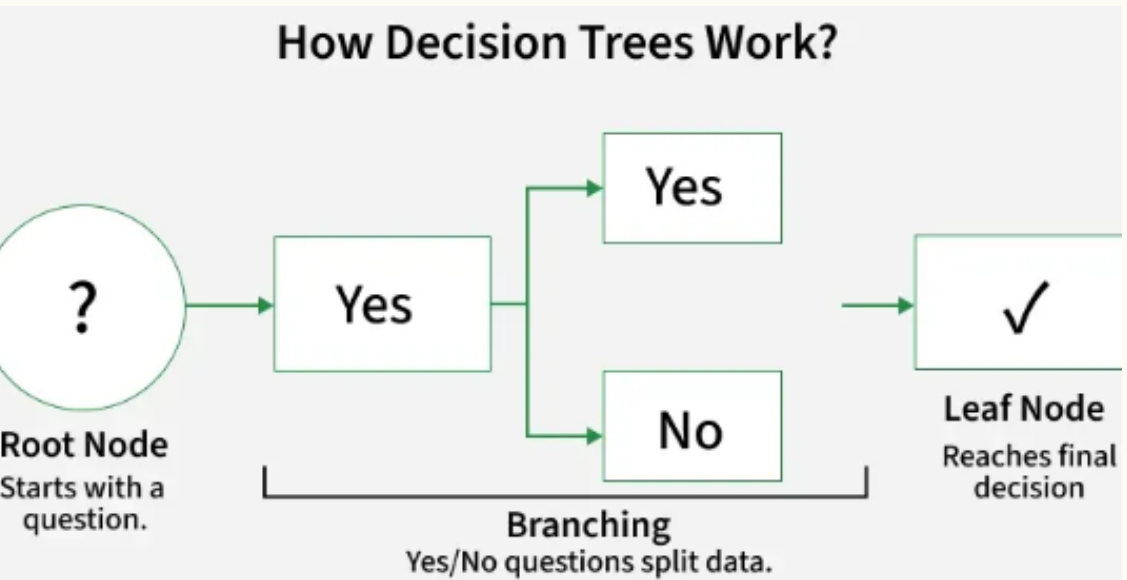
Entropy

- Measures the randomness or uncertainty in your data.

Formula:

Steps to use it:

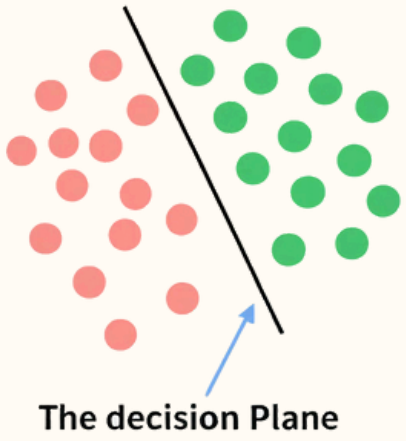
a. Compute entropy for the dataset.
b. Compute entropy for each attribute.
c. Calculate information gain for each attribute.
d. Pick the attribute with highest gain to split next.



- **Start at the Root:** Begin with one main question based on your data.
- **Ask Questions:** Each question splits the data into “yes” or “no” groups.
- **Follow Branches:** Depending on the answer, move down the corresponding branch.
- **Keep Splitting:** Keep asking questions on each branch until the data is clearly divided.
- **Reach a Leaf:** Stop when no more questions help — the leaf gives the final answer or prediction.

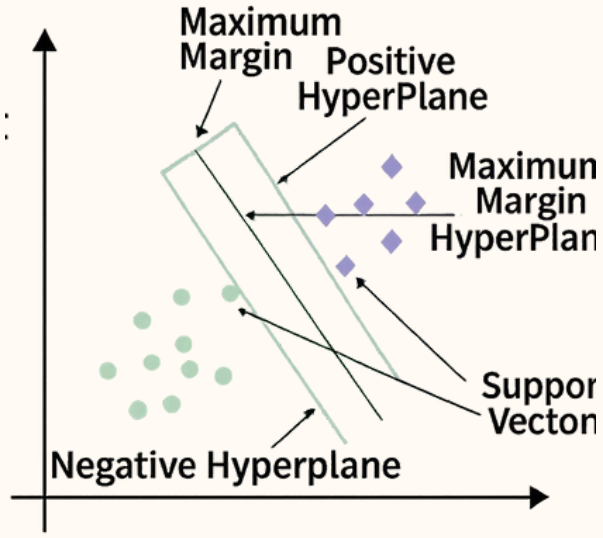
SVM(Support Vector Machine)

- Support Vector Machine is used for classification and regression.
- It works by finding decision planes that separate data into different classes.



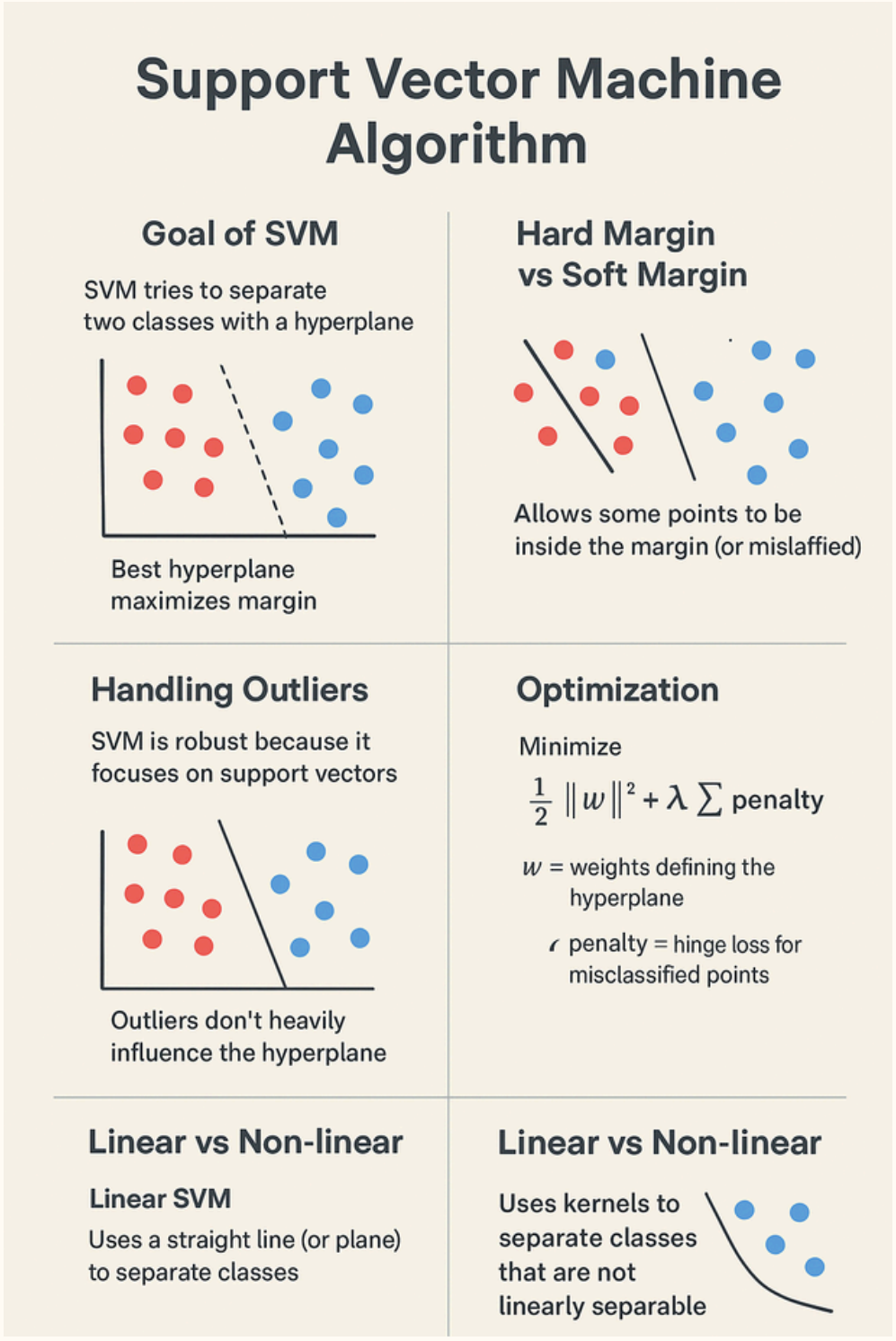
Support Vector Machine (SVM) is a supervised machine learning algorithm used for classification and regression tasks. It tries to find the best boundary known as hyperplane that separates different classes in the data. It is useful when you want to do binary classification like spam vs. not spam or cat vs. dog.
The main goal of SVM is to maximize the margin between the two classes. The larger the margin the better the model performs on new and unseen data.

Support Vectors & Hyperplane



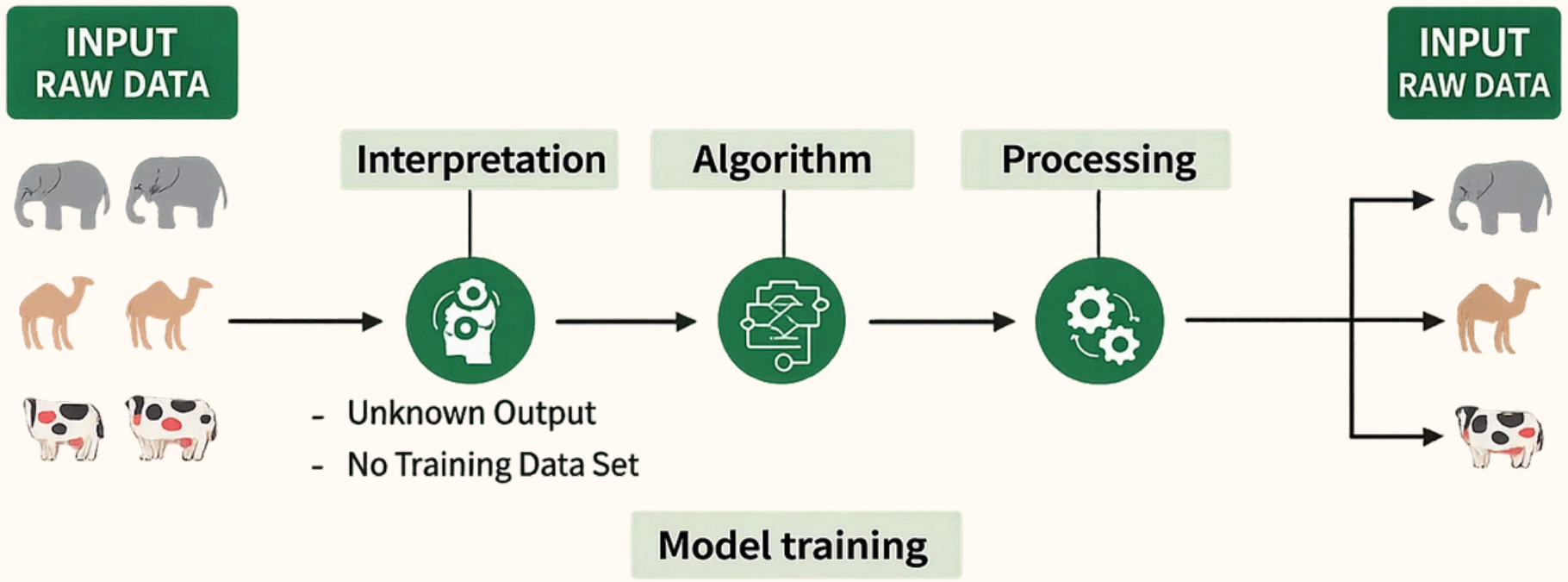
- **Hyperplane:** The line (or plane) that separates different classes in the data.
- **Support Vectors:** The data points closest to the hyperplane; they determine its position.
- **Margin:** The distance between the hyperplane and the support vectors; SVM maximizes this.
- **Kernel:** A function that transforms data into a higher dimension to handle non-linear separation.

- **Hard Margin:** Finds a hyperplane that perfectly separates the classes.
- **Soft Margin:** Allows some misclassified points to handle imperfect data.
- **C (Regularization):** Controls the trade-off between wide margin and misclassification penalty.
- **Hinge Loss:** Penalizes points on the wrong side of the margin.



Unsupervised Learning

Unsupervised learning is a type of machine learning that analyzes and models data without labelled responses or predefined categories. Unlike supervised learning, where the algorithm learns from input-output pairs, unsupervised learning algorithms work solely with input data and aim to discover hidden patterns, structures or relationships within the dataset independently, without any human intervention or prior knowledge of the data's meaning.



The image shows set of animals like elephants, camels and cows that represents raw data that the unsupervised learning algorithm will process.

- **The "Interpretation"** stage signifies that the algorithm doesn't have predefined labels or categories for the data. It needs to figure out how to group or organize the data based on inherent patterns.
- **An algorithm** represents unsupervised learning process which can be clustering, dimensionality reduction or anomaly detection to identify patterns in the data.
- **The processing** stage shows the algorithm working on the data.
- **The output** shows the results of the unsupervised learning process. In this case, the algorithm might have grouped the animals into clusters based on their species (elephants, camels, cows).

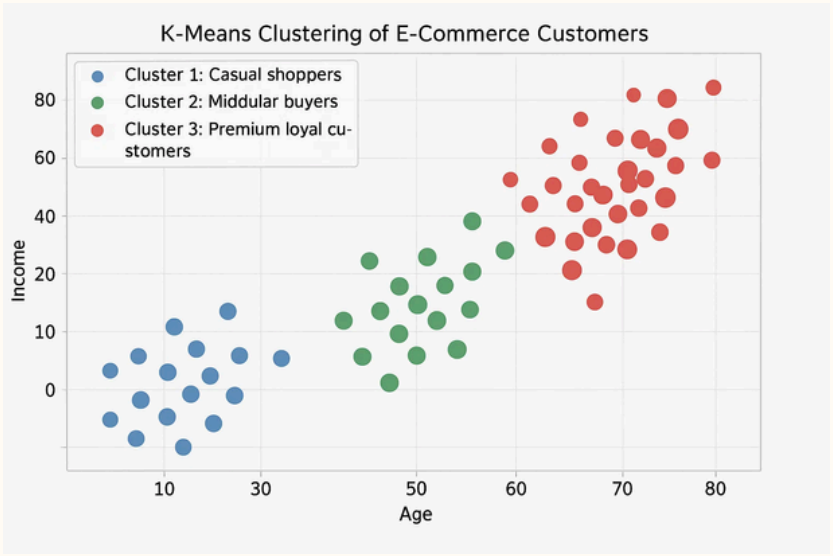
Suppose you run an e-commerce store and have customer data like:

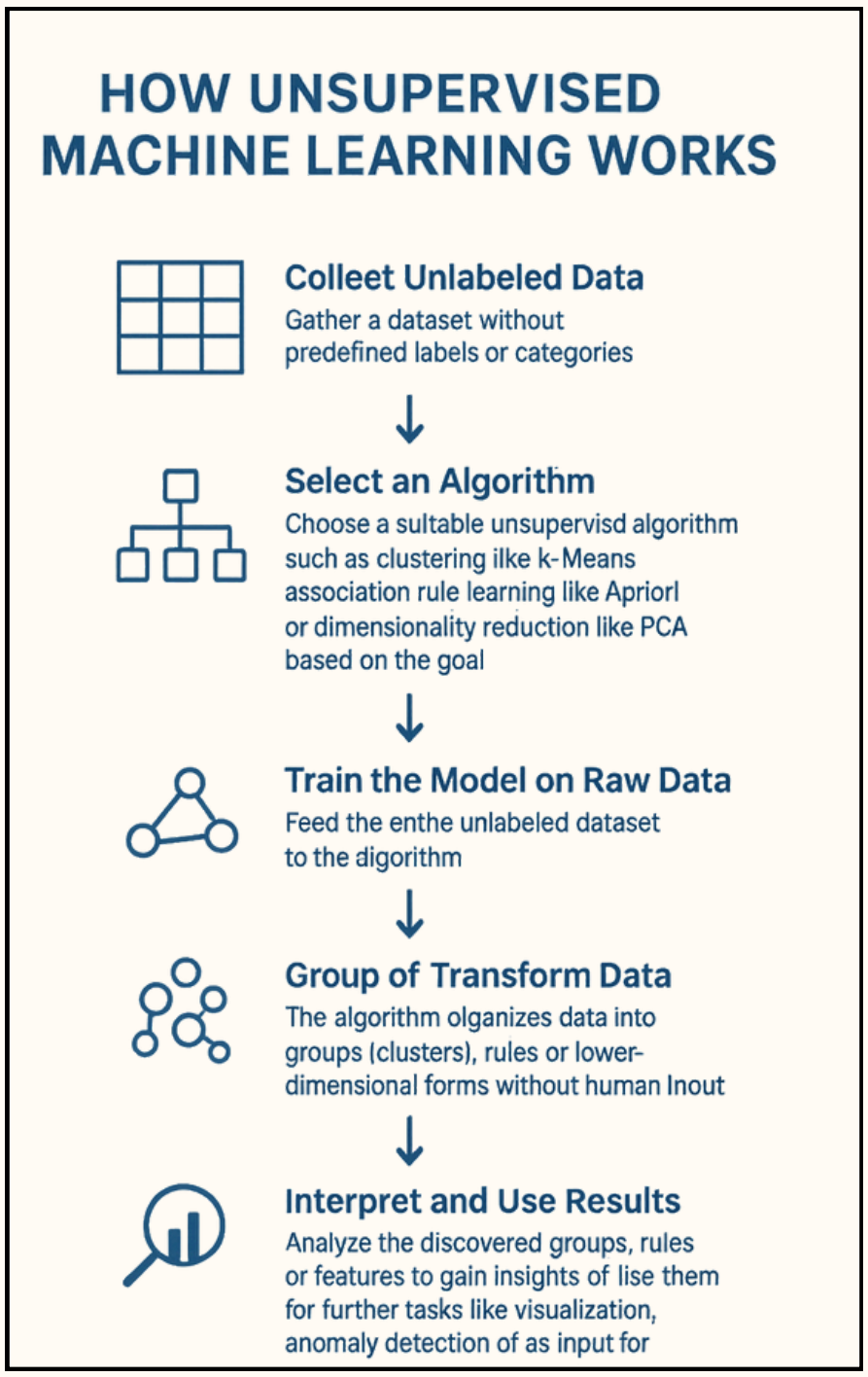
- Age
- Income
- Number of purchases per month

But you don't know in advance which type of customers exist (no labels like "loyal customer" or "casual shopper").
If you apply an unsupervised learning algorithm like K-Means clustering, it may automatically group your customers into **clusters such as:**

- **Cluster 1:** Young, low income, few purchases → Casual shoppers
- **Cluster 2:** Middle-aged, medium income, frequent purchases → Regular buyers
- **Cluster 3:** Older, high income, very frequent purchases → Premium loyal customers

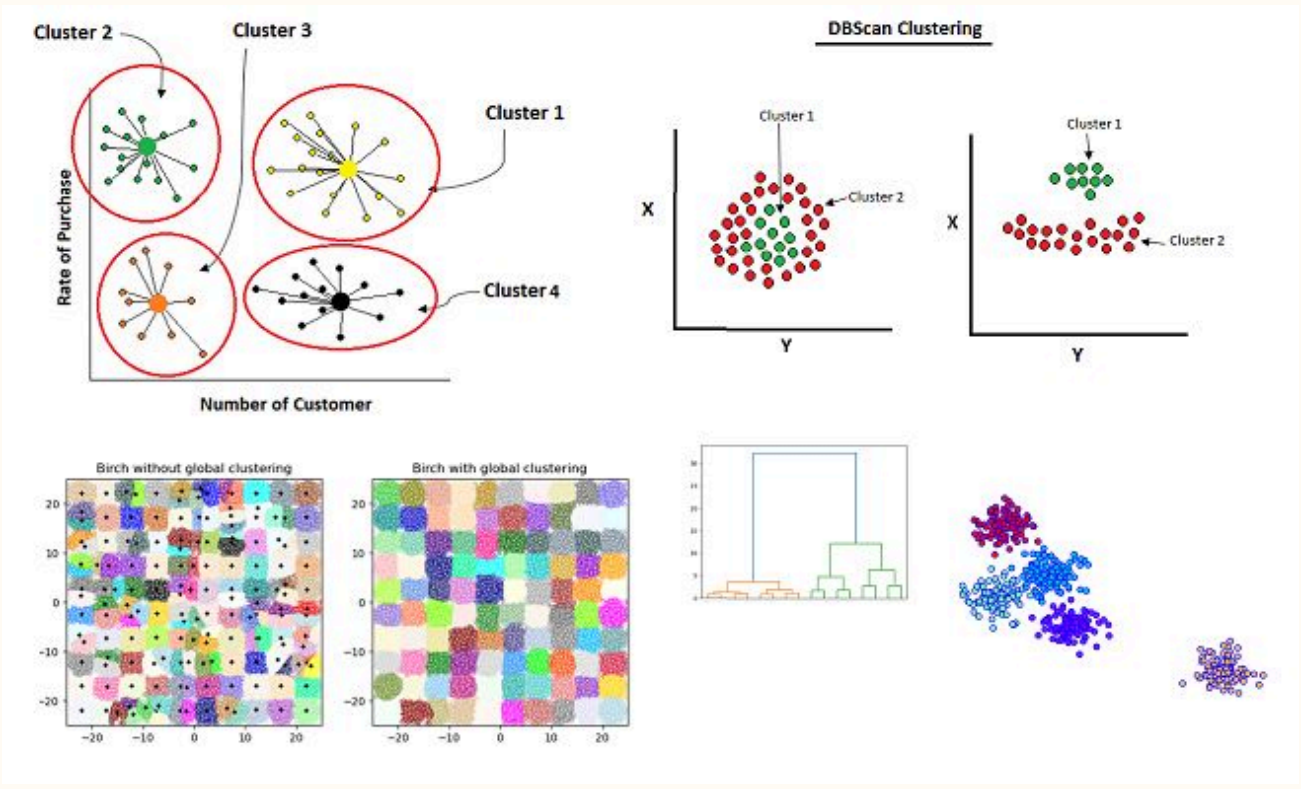
Here, the algorithm discovered these groups by itself without you telling it what the categories were.





Clustering Algorithms

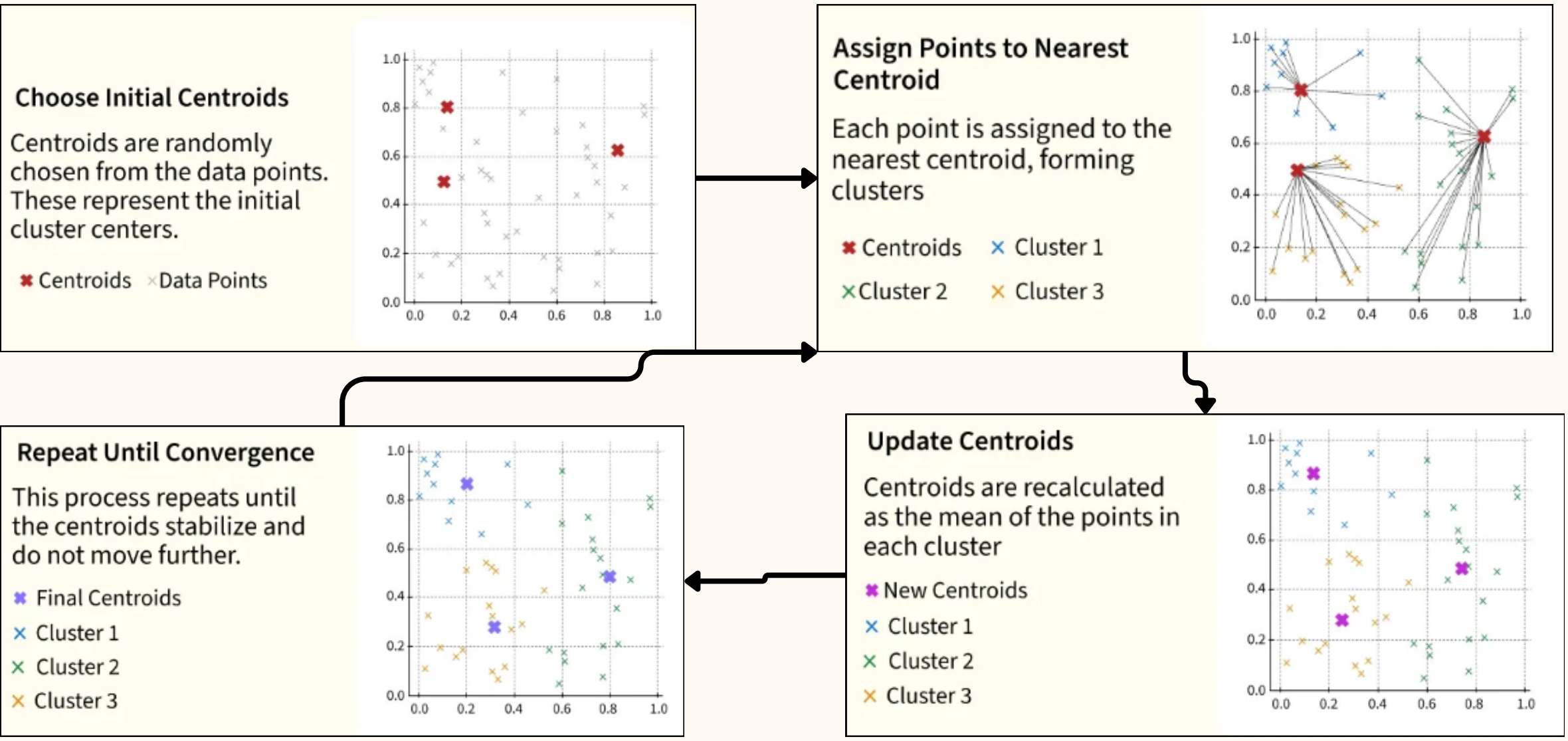
is an unsupervised machine learning technique that groups unlabeled data into clusters based on similarity. Its goal is to discover patterns or relationships within the data without any prior knowledge of categories or labels.



K-means Clustering

Groups data into K clusters based on how close the points are to each other.

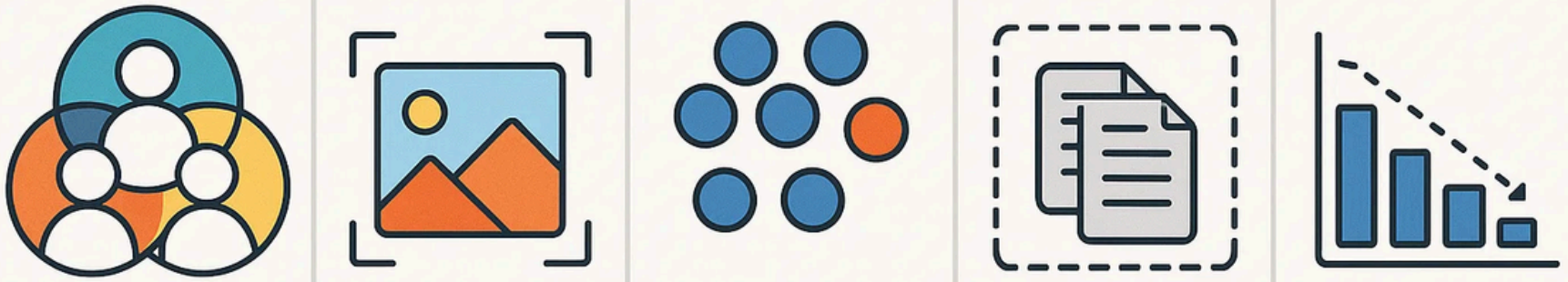
Working of K-Means Clustering



- **Pick centers:** Choose k random points to start (these are the centers of the groups, called centroids).
- **Assign items:** For each item, check which center it's closest to (using distance, usually Euclidean). Put the item in that group.
- **Move centers:** Recalculate each group's center by averaging all the items inside it.
- **Repeat:** Keep reassigning items and moving centers until the groups stop changing (or until we've repeated enough times).

The idea: Items in the same group should look alike, and items in different groups should be different.

WHY USE K-MEANS CLUSTERING?



DATA SEGMENTATION

Segmenting data into distinct groups

IMAGE COMPRESSION

Grouping similar pixels into clusters

ANOMALY DETECTION

Detecting anomalies or outliers

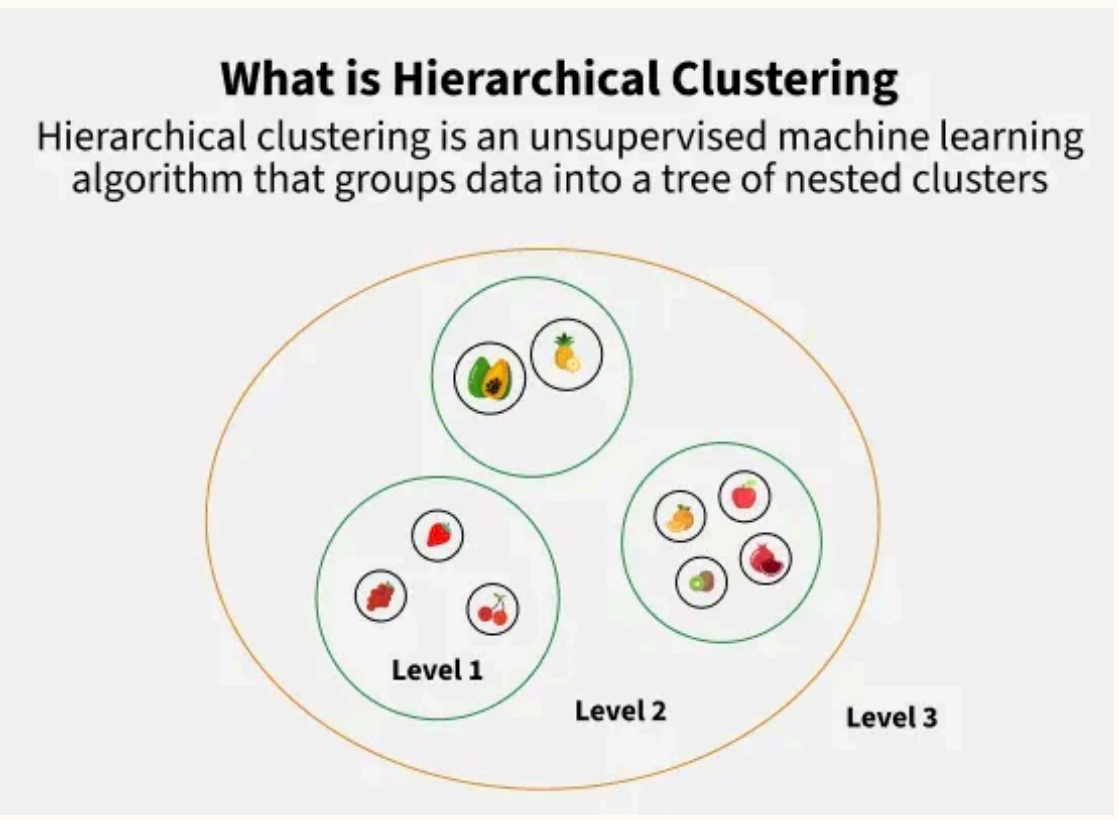
DOCUMENT CLUSTERING

Grouping similar documents together

ORGANIZING LARGE DATASETS

Organizing data into manageable chunks

Hierarchical Clustering in Machine Learning



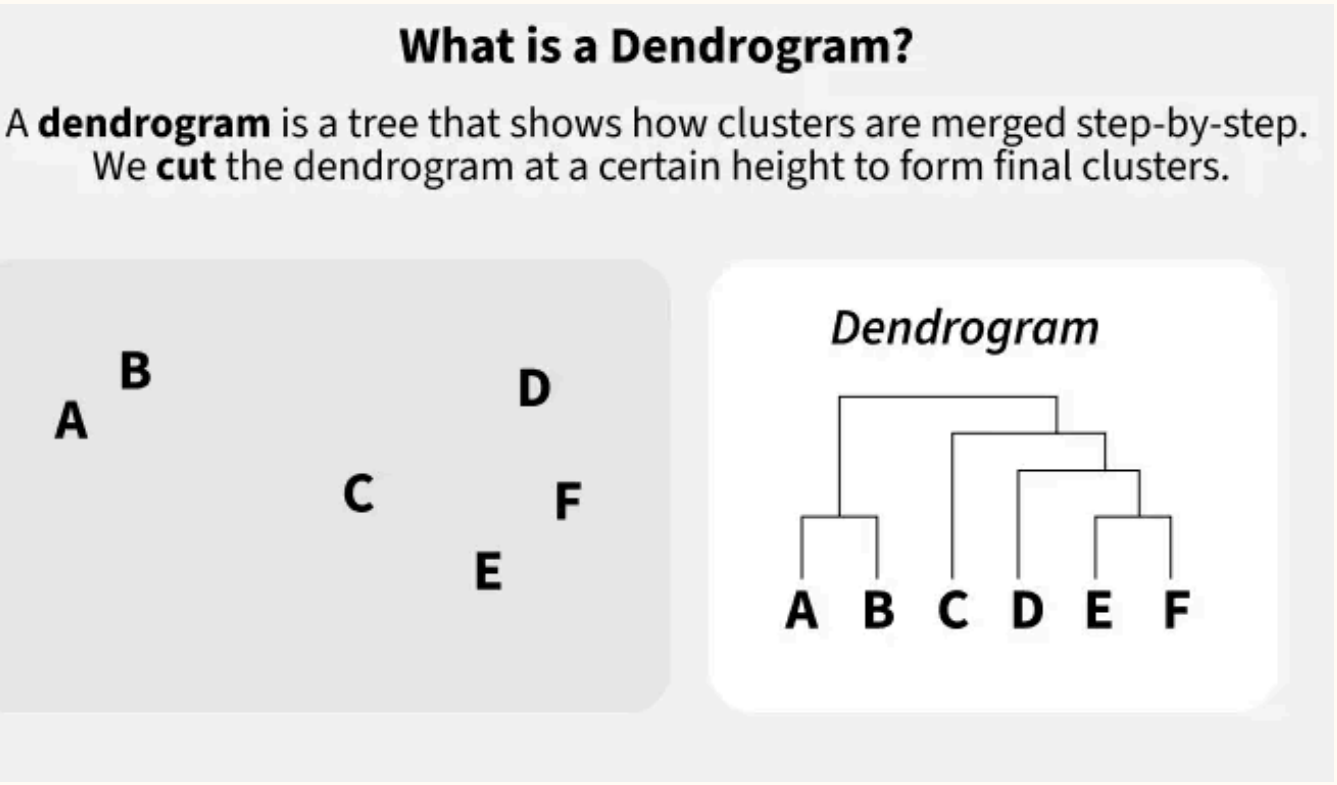
Imagine we have four fruits with different weights: an apple (100g), a banana (120g), a cherry (50g) and a grape (30g).

Hierarchical clustering starts by treating each fruit as its own group.

- **Start** with each fruit as its own cluster.
- **Merge** the closest items: grape (30g) and cherry (50g) are grouped first.
- **Next,** apple (100g) and banana (120g) are grouped.
- **Finally,** these two clusters merge into one.

Finally all the fruits are merged into one large group, showing how hierarchical clustering progressively combines the most similar data points.

Dendrogram



A **dendrogram** is just a tree diagram that shows how things get grouped step by step.

1. At the bottom, every point (like P, Q, R, S, T) is alone.
2. As you go up, the closest ones (most similar) get joined together.
3. The lines show which points or groups got merged.

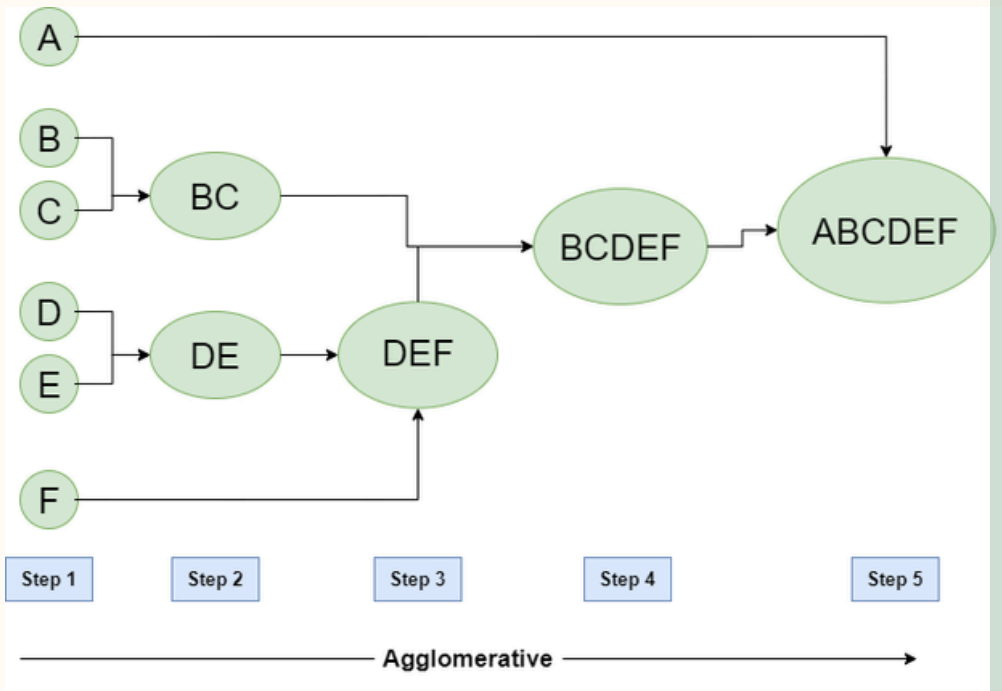
The height of the line tells how different they are:

- Short line = very similar
- Tall line = less similar

Types of Hierarchical Clustering

1. Hierarchical Agglomerative Clustering

It is also known as the bottom-up approach or hierarchical agglomerative clustering (HAC). Bottom-up algorithms treat each data as a singleton cluster at the outset and then successively agglomerate pairs of clusters until all clusters have been merged into a single cluster that contains all data.



How it works:

- **Start with each point alone**

If you have 5 points (P, Q, R, S, T), you start with 5 tiny clusters.

- **Measure closeness**

Calculate how close each point is to every other point (their distance).

- **Merge the closest two**

Pick the two points (or clusters) that are closest and join them into one bigger cluster.

- **Update distances**

Now treat that new cluster as a single group. Recalculate how close it is to the other clusters.

- **Repeat**

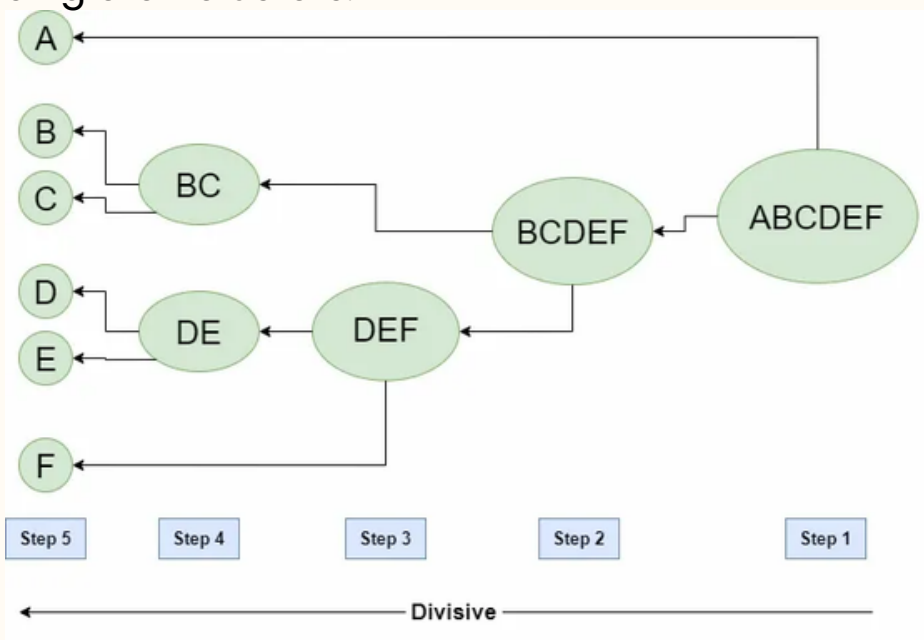
Keep merging the closest clusters, one by one, until everything becomes one big cluster.

- **Draw the dendrogram**

While merging, draw a tree diagram that shows who merged with who, and how similar they were (shorter line = more similar).

2. Hierarchical Divisive clustering

Divisive clustering is also known as a top-down approach. Top-down clustering requires a method for splitting a cluster that contains the whole data and proceeds by splitting clusters recursively until individual data have been split into singleton clusters.



- **Start big**

Put all data points together in one giant cluster.

- **Find the biggest difference inside**

Look for the two points that are most different (farthest apart).
Use them to split the big cluster into two smaller clusters.

- **Keep splitting**

Take one of the new clusters.
Again, find the most different points inside it and split it into two.

- **Repeat**

Continue splitting clusters, one at a time.

- **Stop**

Either when each point is alone in its own cluster,
Or when you've reached the number of clusters you want.

Density-Based Clustering (DBSCAN)

DBSCAN groups points that are close together and calls them a cluster.
If a point is too far away from others, it's marked as noise (outlier)

👉 Analogy:

Imagine people standing on a field:

- If many people stand close together → they form a group (cluster).
- If someone is standing alone far away → they are considered noise.
- Groups can be any shape (circle, line, snake-like), not just neat round circles.

Why it's special (compared to K-Means or hierarchical)

- **Any shape clusters**: It doesn't need clusters to be round. Clusters can be long, curved, or weirdly shaped.
- **Handles noise:** Points that don't belong anywhere are left alone as outliers.

- **K-Means & Hierarchical:**

1. Work best when clusters are round/compact.
2. Struggle with weird shapes (like curved lines).
3. Don't naturally handle outliers well (they usually force every point into some cluster).

- **DBSCAN:**

4. Can find clusters of any shape (curvy, long, irregular).
5. Leaves out noise/outliers instead of forcing them into a cluster.
6. Better for real-world messy data.

👉 Think of it like this:

- K-**Means/Hierarchical = neat boxes/circles** 📦⭕
- **DBSCAN = freeform drawing ✏️ that adapts to how the data is actually shaped.**

Key Parameters in DBSCAN

eps (neighborhood radius):

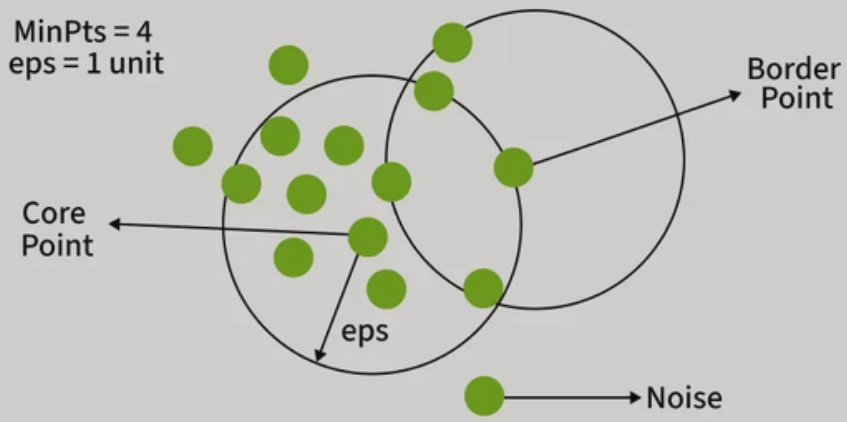
- Think of it as how close points need to be to each other to be considered part of the same group.
- Too small: many points are left out as "noise."
- Too large: separate groups might merge into one big cluster.
- You can pick a good value by looking at a k-distance graph.

MinPts (minimum points):

- Minimum number of points needed to form a cluster.
- Rule of thumb: MinPts ≥ D + 1, where D is the number of features (dimensions).
- Usually, MinPts = 3 works fine for simple cases.

How Does DBSCAN Work?

- **Core points:** Have enough neighbors (≥ MinPts) within eps. They are the heart of clusters.
- **Border points:** Near core points but don't have enough neighbors to be core points.
- **Noise points**: Don't belong to any cluster; considered outliers.



Steps of the DBSCAN Algorithm

Identify Core Points:

- Check each point. If it has at least MinPts neighbors within distance eps, it's a core point.
- Form Clusters:
- Start a new cluster from an unassigned core point.
- Add all points that are density-reachab**le (within eps of the core or connected through other core points).**

Density Connectivity:

- Two points are density-connected if you can "hop" from one to the other through a chain of points, each within eps of the next, and at least one is a core point.
- This allows clusters to grow organically, even if they have irregular shapes.

Label Noise:

Any point not assigned to a cluster after all core points are processed is labeled noise.

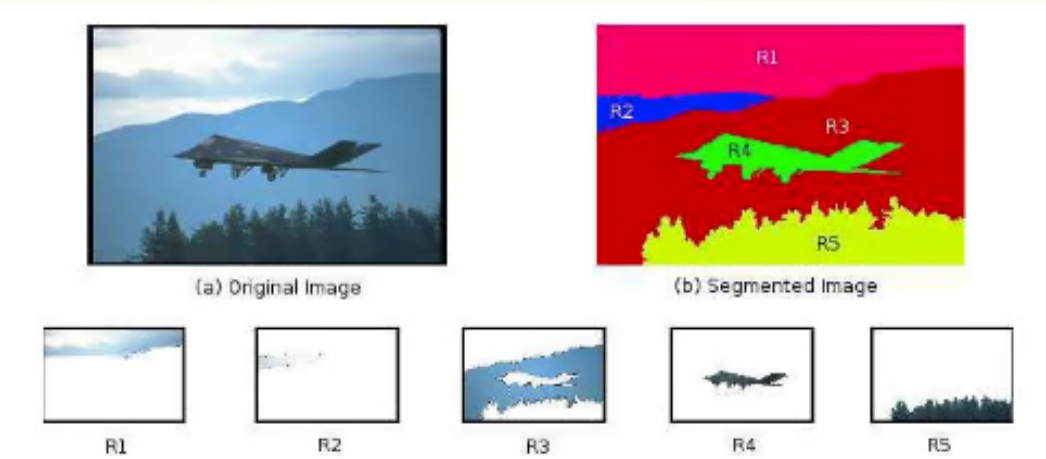
ML | Mean-Shift Clustering

Category: Unsupervised clustering algorithm (density-based)
Idea:

- Every point moves towards the area with the highest density of points (the mode).
- Think of points “climbing uphill” to the nearest hilltop — the hilltops become cluster centers.

Why Use Mean Shift Clustering?

Image segmentation



Purpose:

- Divide an image into regions (segments) based on pixel color or intensity.
- No need to know the number of segments in advance.
- Works well with irregular shapes and noisy images.

How It Works for Images

1. Treat each pixel as a point in color (or intensity) space.
2. Shift pixels toward areas with higher pixel density (mode).
3. Pixels that converge to the same mode form a segment.
4. Result: The image is divided into regions where pixels have similar color or intensity.

Object tracking in video analysis

Purpose:

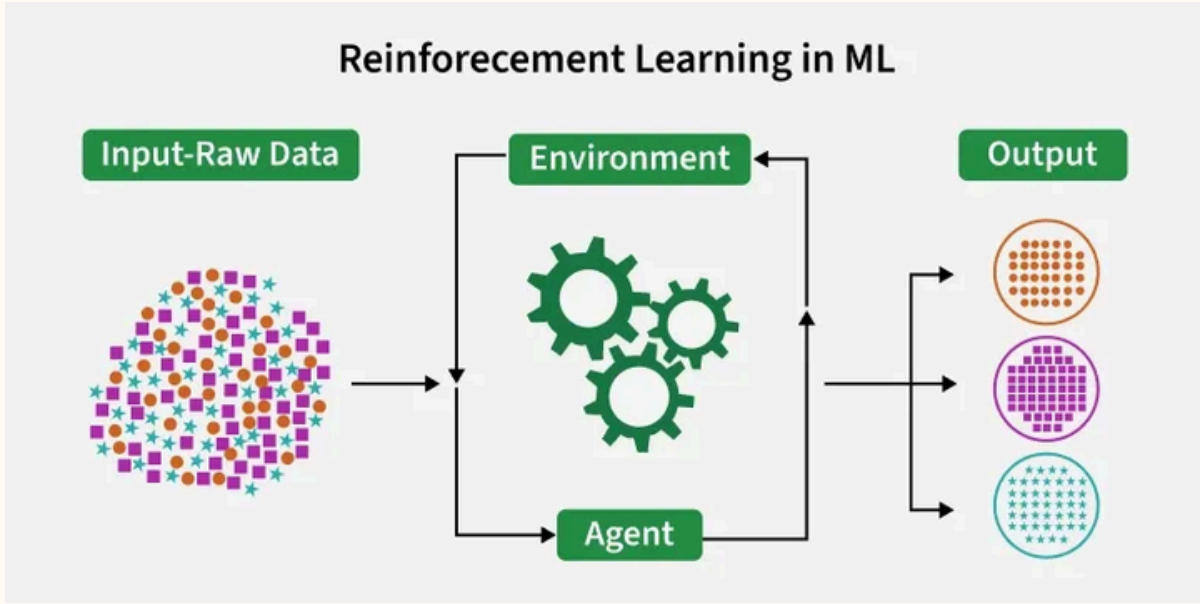
- Track objects in video frames by following areas with high pixel or feature density.
- Works for objects of different shapes and sizes.
- Can adapt to changes in movement, position, and appearance.

How It Works in Videos

1. Select the object in the first frame. Treat its pixels/features as a dense region.
2. Search in the next frame for regions with similar pixel/feature density.
3. Shift the search window toward the area of maximum density.
4. Repeat for each frame to track the object continuously.

Reinforcement Learning

Reinforcement Learning (RL) is a branch of machine learning that focuses on how agents can learn to make decisions through trial and error to maximize cumulative rewards. RL allows machines to learn by interacting with an environment and receiving feedback based on their actions. This feedback comes in the form of rewards or penalties.



Core Components

Component	Role	Example in Self-Driving Car
Policy	Chooses actions	Stop when pedestrian
Reward Signal	Tells agent good/bad	+10 safe driving, -50 collision
Value Function	Estimates long-term benefit	Avoid risky shortcut for
Model	Simulates environment	Predict other vehicles'

Reinforcement Learning (RL) revolves around the idea that an agent (the learner or decision-maker) interacts with an environment to achieve a goal. The agent performs actions and receives feedback to optimize its decision-making over time.

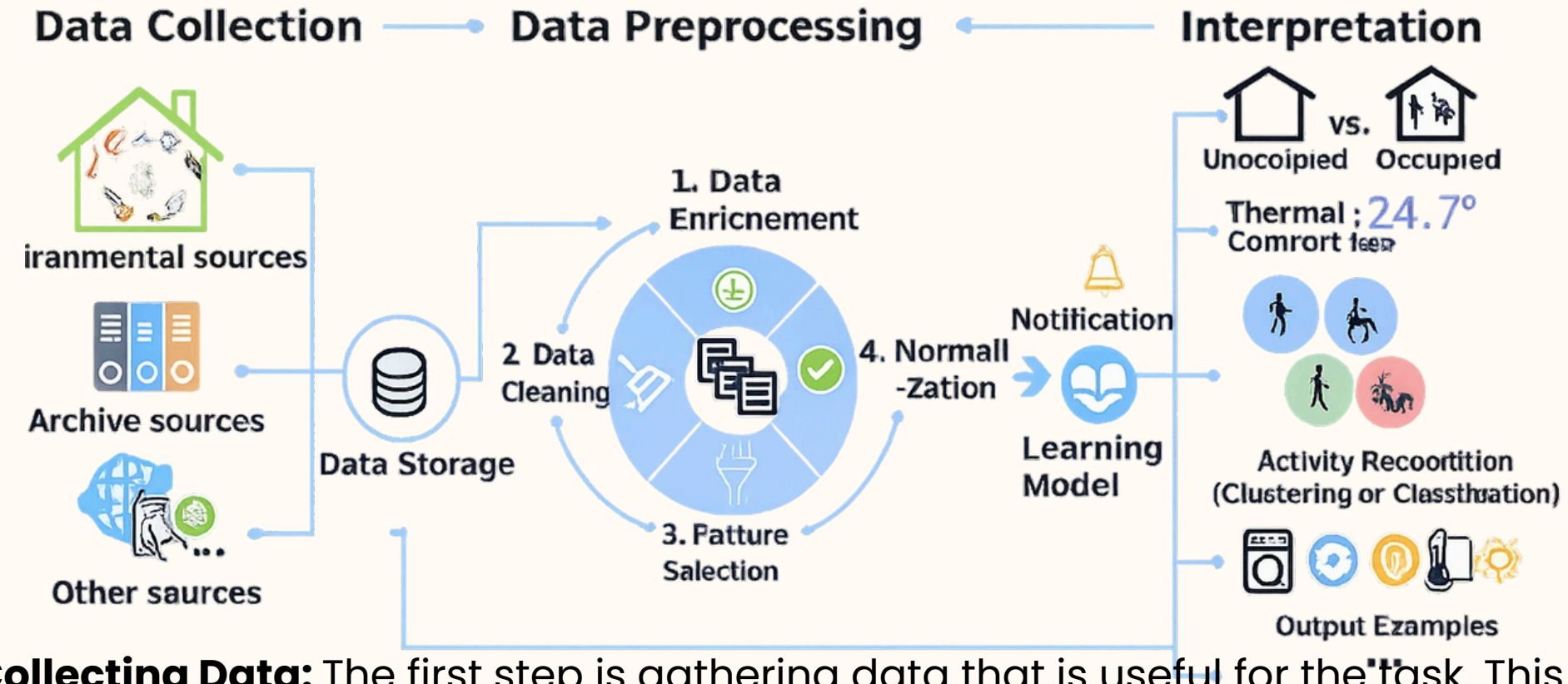
- **Agent**: The decision-maker that performs actions.
- **Environmen**t: The world or system in which the agent operates.
- **State:** The situation or condition the agent is currently in.
- **Action:** The possible moves or decisions the agent can make.
- **Reward:** The feedback or result from the environment based on the agent's action.

RL Feedback Loop

The agent learns by interacting with the environment in a loop:

1. **Observe State:**
   - The agent sees the current situation of the environment.
   - Example: A self-driving car sees a red light ahead**.**
2. **Choose Action:**
   - Based on its policy, the agent decides what to do.
   - Example: Stop, accelerate, or turn.
3. **Receive Reward:**
   - The environment responds with a reward or penalty.
   - Example: +10 for stopping safely, -50 for running a red light.
4. **Update Knowledge:**
   - The agent updates its policy or value function to improve future decisions.
5. **Repeat:**
   - The agent keeps exploring new actions and exploiting known good ones to maximize cumulative reward.

Machine Learning Pipeline



Collecting Data: The first step is gathering data that is useful for the task. This could be anything from numbers and images to text. The more data available, the better the machine can learn.
Cleaning Data: Once the data is collected, it often needs to be cleaned up. This includes fixing errors, filling in missing information, or formatting the data in a way that the machine can use.
Choosing a Model: There are different methods (or models) used in machine learning, depending on the problem you're solving.
Training the Model: In this step, the model learns from the data, finding patterns and making connections to help it make better predictions.
Testing and Checking: After training, the model is tested on new data it hasn't seen before. This helps make sure the model works well and can handle new situations, not just the data it was trained on.
Improving the Model: Based on how the model performs during testing, it might be adjusted to improve its accuracy. This could mean adding more data or changing settings.
Making Predictions: After the model is trained and performs well, it can start making predictions or decisions based on new data. This is when the machine applies what it has learned to real-world tasks.