1.1 - Introduction to Machine Learning

Giovanni Della Lunga
giovanni.dellalunga@unibo.it

Introduction to Machine Learning for Finance

Bologna - February-March, 2026

Introduction to this Course

- **Welcome to the Course!**
  - Instructor: Giovanni Della Lunga
  - Contact: giovanni.dellalunga@unibo.it
  - Course Duration: February-March 2025
- **Course Goals:**
  - Develop a solid understanding of Machine Learning (ML) fundamentals.
  - Explore ML applications in finance, including prediction and risk assessment.
  - Gain hands-on experience with Python tools for ML.

Lecture Overview

In this lesson, we will explore the foundational concepts of Machine Learning (ML).

Key Topics Covered

- What is Machine Learning? Definitions and core principles.
- Types of ML: Supervised, Unsupervised, Semi-supervised, and Reinforcement Learning.
- ML workflow: Data gathering, preprocessing, modeling, and evaluation.
- Essential ML concepts: Features, labels, cost functions, and optimization.
- Introduction to Python tools for ML: Pandas, scikit-learn, and Jupyter Notebooks.
- Using Pandas for financial data gathering.

Introduction to Machine Learning for Finance

Learning Objectives By the end of this lesson, students will:

- Understand the fundamental concepts of ML and its role in finance.
- Recognize different types of ML models (Supervised, Unsupervised, etc.).
- Learn how to gather and preprocess financial data using Python (Pandas, DataReader, yFinance).
- Gain an overview of the ML workflow from data collection to model optimization.

Working Environment

Getting Started with Python

Course Environment

In this course, we will use Python mainly within a **Jupyter Notebook** environment.

Alternative Approach

Alternatively, you can write a Python script in a file with a `.py` extension, which can be executed via the command line:

```
python filename.py
```

What is Jupyter Notebook?

Jupyter Notebook is an extremely powerful and flexible tool for Python programming that allows you to integrate:

- Code execution
- Notes and documentation
- Interactive visualizations

All in a single interactive environment!

Popular Uses

Particularly appreciated in data science, data analysis, and teaching thanks to its ability to combine code execution, result visualization, textual annotations, and more in an easily shareable and reproducible format.

Types of Cells in Jupyter Notebook

Jupyter Notebooks support two types of cells:

Text Cells

- Write formatted text using Markdown syntax
- Insert descriptive text
- Include images, formulas (LaTeX), tables, and more
- Facilitate clear documentation of data analysis

Code Cells

- Write and execute Python code
- Execute by clicking the triangle on the left
- Different cells can contain different instructions
- Functions defined in previous cells can be reused

Important

A function defined in a previous cell can only be used if that cell has been executed!

Text Formatting with Markdown

Markdown allows you to enrich text cells with various formatting options:

Syntax:

```
# Heading
## Subheading

- Bulleted item
1. Numbered item

[Link text](URL)

**bold** *italic*
```

Result:

- **Headings:** # for level 1, ## for level 2
- **Lists:** - for bullets, 1. for numbers
- **Links:** [Text](URL)
- **Emphasis:** **bold**, *italic*

This creates a structured and readable document!

Magic Commands in Jupyter Notebook

Jupyter Notebooks support **magic commands**, special commands that start with:

- % for single-line commands
- %% for cell-wide commands

Common Magic Commands

`%run`	Execute an external Python file
`%timeit`	Evaluate execution time of a single line of code
`%matplotlib inline`	Integrate Matplotlib plots directly in the notebook
`%load`	Load code from an external file into a cell
`%reset`	Clear all variables defined in the notebook
`%pwd, %cd`	Manage working directory path

Exploring Magic Commands

Discover All Magic Commands

Type `%lsmagic` in a cell to access the complete list of available magic commands!

```
%lsmagic
```

This allows you to explore additional tools and functionalities offered by Jupyter Notebook.

Local Configuration for Jupyter Notebook

To start working with Jupyter Notebooks in your local development environment, complete these preliminary steps:

1. **Install Python via Anaconda**
   - Anaconda is a Python distribution that already includes Jupyter and other useful libraries for data science
   - Follow detailed instructions available on the Anaconda website
2. **Select an Integrated Development Environment (IDE)**
   - Visual Studio Code (VS Code) is an excellent choice
   - Free and versatile
   - Supports Python through specific extensions

Local Execution or on Colab

Jupyter Notebooks can be executed in two ways:

Local Execution

- Run on your computer
- Full control over environment
- No internet required

Remote Server (Google Colab)

- Run on cloud infrastructure
- Access from any device
- Easy sharing with others

Flexibility

This flexibility allows users to access their notebooks from any internet-connected device and easily share their work with others.

Google Colab Overview

What is Google Colab?

A platform that allows you to write Python code directly in your browser without any installation.

Getting Started:

- Access Google Colab via the link
- Select "New notebook" to create a new workspace
- Save notebooks to your Google Drive folder
- Consult the guide for complete introduction to Colab features

Important Limitation

The free version of Google Colaboratory runtime does not save information permanently - work is deleted after the session ends!

Virtual Environments

Important Requirement

The Python kernel must be installed within a dedicated development environment known as a **virtual environment**.

Why Use Virtual Environments?

- Separate and isolate libraries and dependencies needed for the kernel
- Avoid conflicts between different configurations
- Ensure proper code execution in the desired context
- Particularly advantageous for projects requiring specific library versions

Managing Environments with Conda

Conda provides useful functions for working with virtual environments:

- Creating separate environments
- Managing environment configurations
- Activating different environments
- Each environment has unique configurations and dependencies

Benefits

This capability greatly simplifies transitioning between different environments, ensuring that each project or kernel has the necessary resources to operate efficiently and without interference.

```
# Example conda commands
conda create -n myenv python=3.9
conda activate myenv
conda install jupyter
```

Summary

Key Takeaways

- Python will be used within Jupyter Notebook environment
- Jupyter Notebooks offer interactive code execution in cells
- Two cell types: Text (Markdown) and Code (Python)
- Can run locally or on cloud platforms like Google Colab
- Virtual environments isolate dependencies and configurations
- Conda manages virtual environments efficiently

What is Machine Learning?

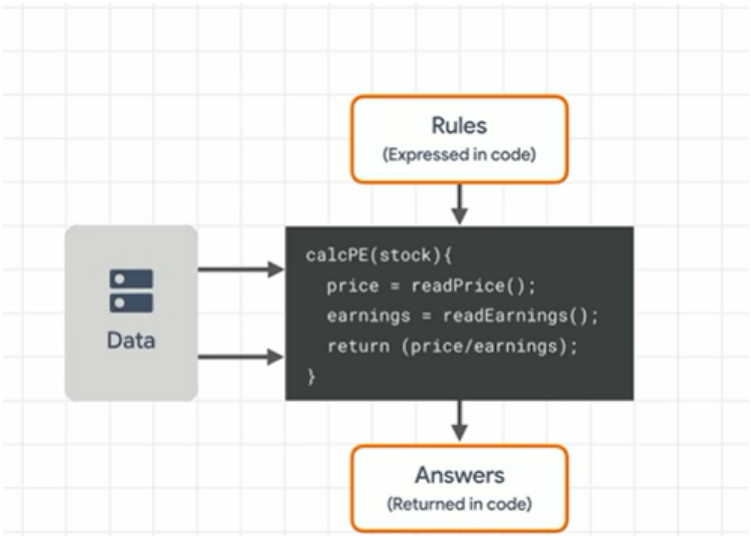
Basic Definitions

Defining Machine Learning

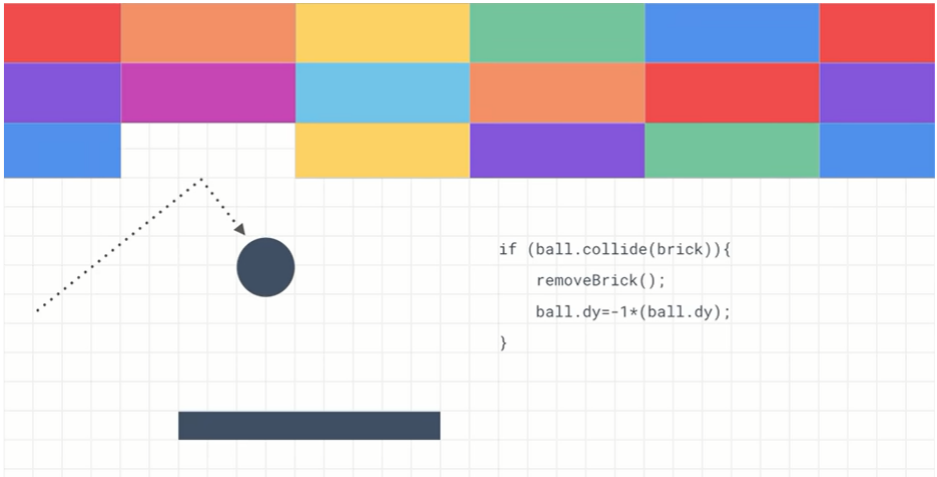
- Machine Learning is a scientific field within artificial intelligence (AI) that focuses on developing algorithms capable of learning from data.
- These algorithms identify patterns, relationships, or recurring motifs in data, which can include numerical information, textual content, images, or statistical records.
- Machine learning enables systems to perform tasks, make predictions, and improve their performance over time without requiring explicitly programmed task-specific rules that is without requiring developers to manually encode all the decision rules: instead, the system learns these patterns from data.
- This process involves feeding data into algorithms that, through analysis, uncover underlying structures to predict outcomes, classify information, or group similar entities.

Defining Machine Learning

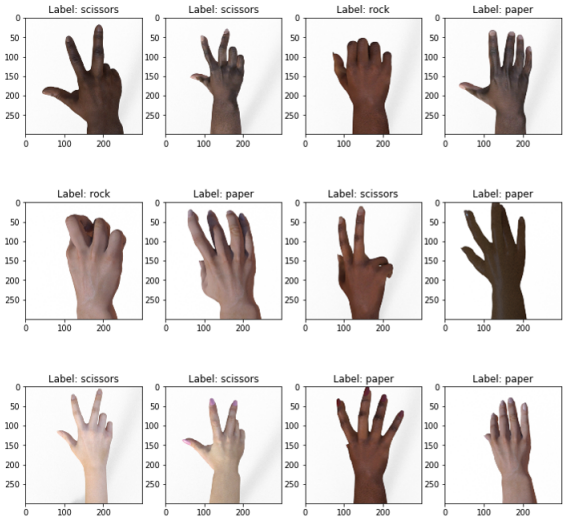
Defining Machine Learning



Defining Machine Learning

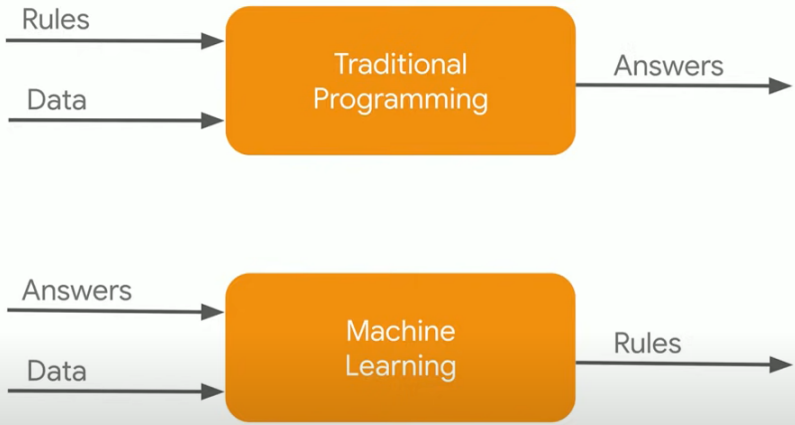


Defining Machine Learning

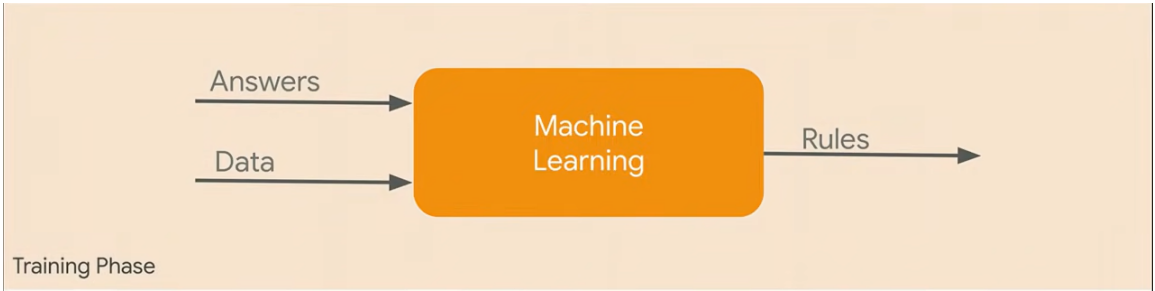



```
Image shape: (300, 300, 3)

[[[254 254 254]
  [253 253 253]
  [254 254 254]
  ...
  [251 251 251]
  [250 250 250]
  [250 250 250]]

 [[254 254 254]
  [254 254 254]
  [253 253 253]
  ...
  [250 250 250]
  [251 251 251]
  [249 249 249]]

 [[254 254 254]
  [254 254 254]
  [254 254 254]
  ...
  [251 251 251]
  [250 250 250]
  [252 252 252]]

 ...
```

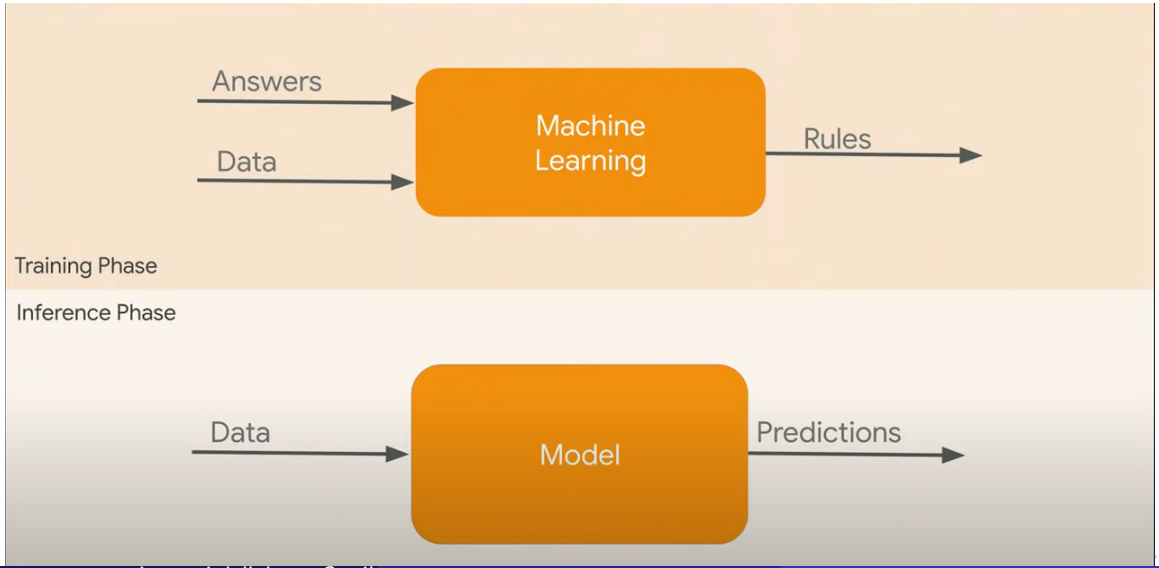
Defining Machine Learning



Defining Machine Learning



Defining Machine Learning



Machine Learning in Finance: Prediction vs Decision

It's very important to underline since beginning a **very important** difference between generic machine learning and machine learning for finance.

- In many domains, ML is presented as *prediction*: learn $Y = f(X)$ and forecast Y.
- In finance, ML is often about *decision support*, not direct forecasting or simple classification.

Key distinction

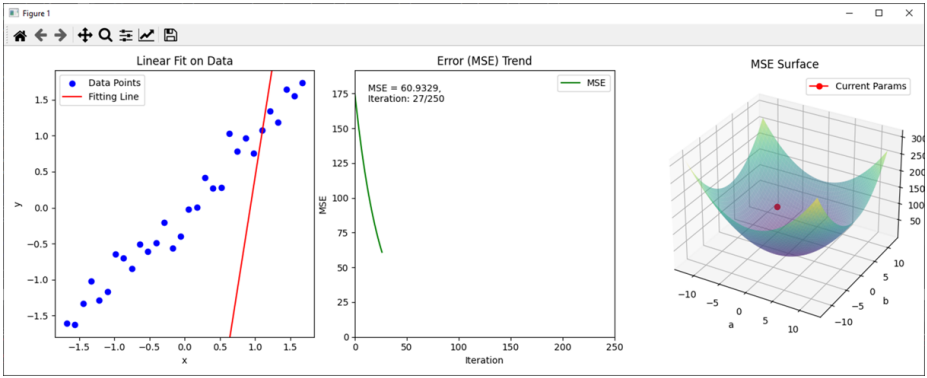
- **Prediction task:** minimize a statistical loss (e.g., MSE, log-loss).
- **Decision task:** maximize an economic objective (e.g., risk-adjusted performance, drawdown control, transaction-cost aware PnL).

Implication

- For example, a model with small prediction error may still produce a poor trading strategy if it ignores risk, costs, constraints, or non-stationarity.

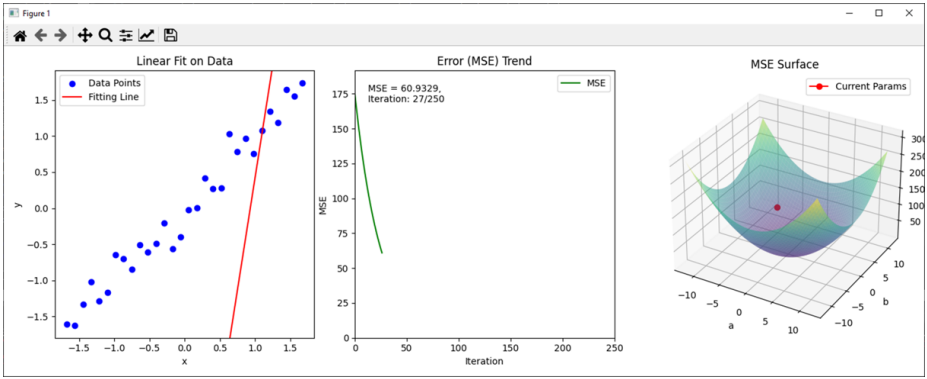
Defining Machine Learning

Machine Learning is a process where a **Model** learns from **Data** to make predictions or decisions. It compares its predictions to the actual outcomes, calculating the **Error**. Then, through **Optimization**, the model adjusts itself to reduce this error and improve its performance over time.



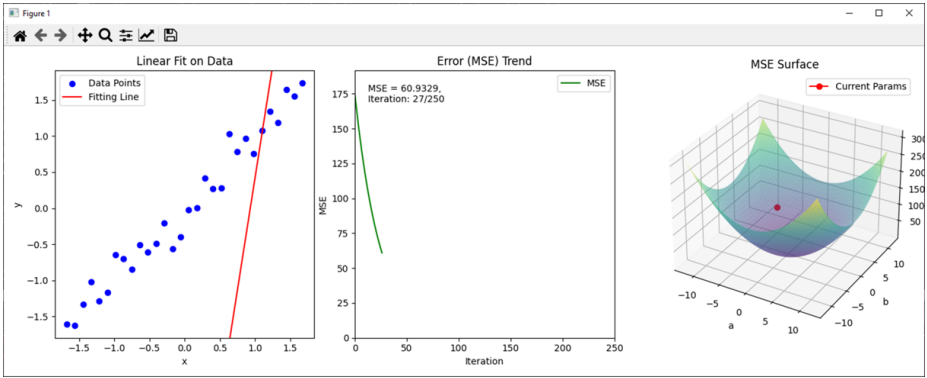
Defining Machine Learning

Data are the foundation of machine learning, as models rely on it to identify patterns and make predictions. In a dataset, **features** are the input variables that describe the data, while **labels** are the target outputs the model aims to predict or classify. Features provide the information, and labels define what the model is learning to predict.



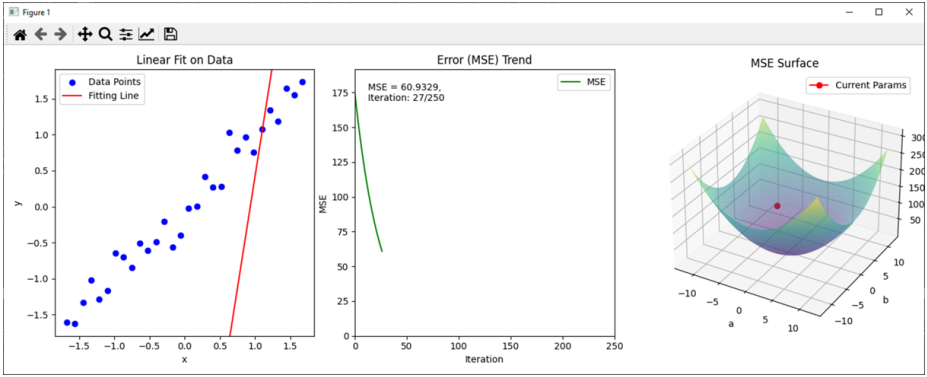
Defining Machine Learning

The **model** is the core of machine learning; it defines how data is processed to generate predictions. In simple linear regression, the model represents the relationship between input features and a target label as a straight line: $y = a + b * X$. Here, b (slope) and a (intercept) are parameters that the model learns from data. By adjusting these parameters during training, the model captures the underlying trend, enabling it to make accurate predictions on new data.



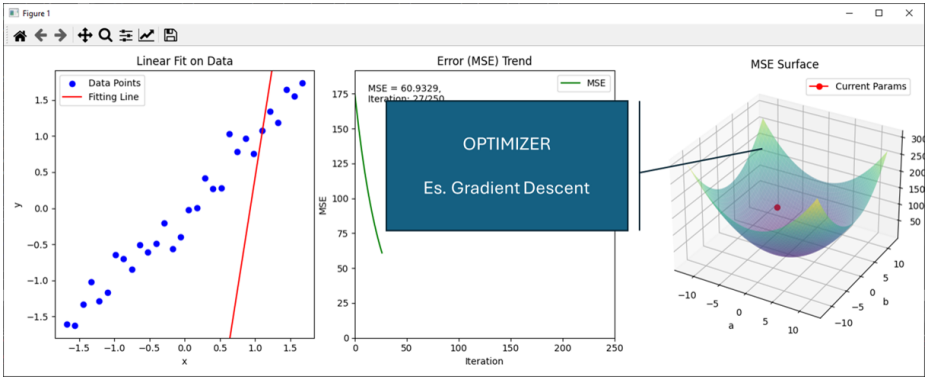
Defining Machine Learning

An **error measure** is essential in machine learning to quantify how far the model's predictions are from the actual values. It provides a way to evaluate the model's performance and guides improvements. By minimizing the error through optimization, we ensure the model learns effectively and generalizes well to new data.



Defining Machine Learning

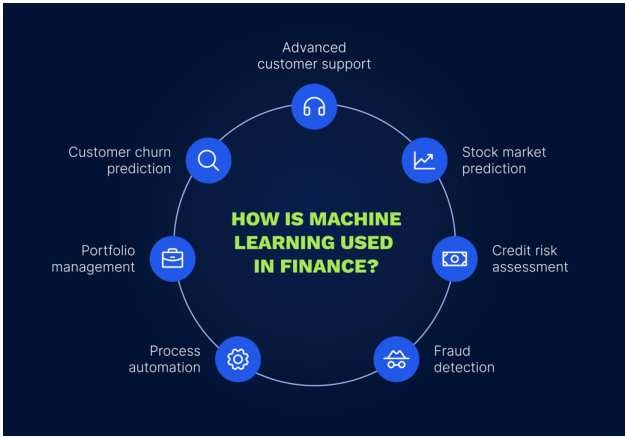
An **optimizer** is used to adjust the model's parameters (e.g., weights in linear regression) to minimize the error. It works by iteratively updating the parameters in the direction that reduces the error, typically using methods like gradient descent. The optimizer calculates the gradient of the error with respect to the parameters and updates them step by step, ensuring the model improves its predictions over time.



Defining Machine Learning

Click the link below to play the video: **Play Video**

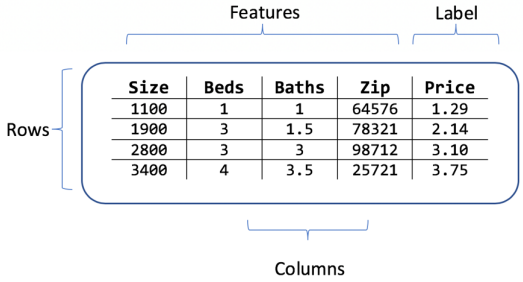
Machine Learning finds applications across diverse fields.



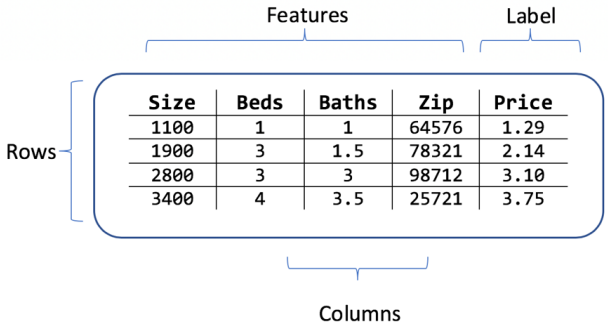
Before discussing the main types of ML models, it is useful to underline again the importance of some basic concepts because they are common to the entire field of machine learning:

- **Features**: The input variables used to make predictions.
- **Labels**: The target outputs that the model aims to predict.
- **Loss Function**: A metric that quantifies the error of the model's predictions.
- **Optimization**: The process of adjusting model parameters to minimize the loss function.

- The data for **ML Models** contains what are referred to as **features** and **labels**;
- *Labels* are the values of the target that is to be predicted;
- *Features* are the variables from which the predictions are to be made (if you are from statistics then think of it as an explanatory variable);



- For example, when predicting the **price of a house**, the *features* could be the *square meters of living space*, *the number of bedrooms*, *the number of bathrooms*, *the size of the garage* and so on.
- The *label* would be *the house price*;



- In Machine Learning a **Cost Function** or loss function is used to represent how far away a mathematical model is from the real data;
- One adjust the mathematical model usually by varying parameters within the model so as to minimize the cost function;
- Let's take for example a very simple model of the form

$$y = \theta_0 + \theta_1 x$$

  where the θs are the parameters that we want to find to give us the best fit to the data;
- Call this function $h_\theta(x)$ to emphasize the dependence on both the variable x and the two parameters θ_0 and θ_1;

- We want to measure how far away the data, the $y^{(n)}$s are from the function $h_\theta(x)$;
- For example in a linear regression problem, a common way to do this is via the quadratic cost function

$$J(\theta) = \frac{1}{2N} \sum_{n=1}^{N} \left[h_\theta \left(x^{(n)} \right) - y^{(n)} \right]^2 \tag{1}$$

- We want the parameters that minimize (**??**), almost always you are going to have to do this numerically;
- If we have a nice convex function then there is a numerical method that will converge to the solution, it is called **gradient descent**.

- The quadratic cost function is probably one of the most used cost function for the regression problem;
- What if you have a classification problem?
- There are a lot of different possibilities. For example, for a simple binary classification problem we can use the logistic loss function

$$L(y, \hat{y}) = -[y \log \hat{y} + (1 - y) \log (1 - \hat{y})] \tag{2}$$

- y is the true label (0 or 1).
- $\hat{y}$ is the predicted probability of class 1.
- The function penalizes incorrect predictions more severely.

$$L(y, \hat{y}) = -[y \log \hat{y} + (1 - y) \log (1 - \hat{y})] \tag{3}$$

When $y = 1$

- $L(1, \hat{y}) = -\log \hat{y}$
- If $\hat{y}$ is close to 1, the loss is small.
- If $\hat{y}$ is close to 0, the loss is large.

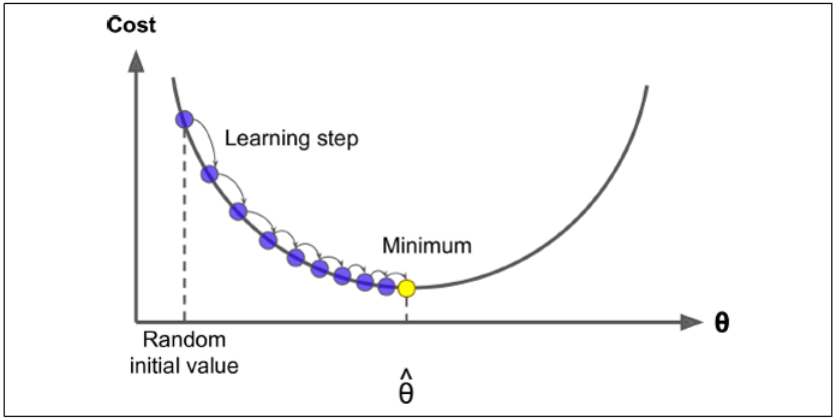
When $y = 0$

- $L(0, \hat{y}) = -\log (1 - \hat{y})$
- If $\hat{y}$ is close to 0, the loss is small.
- If $\hat{y}$ is close to 1, the loss is large.

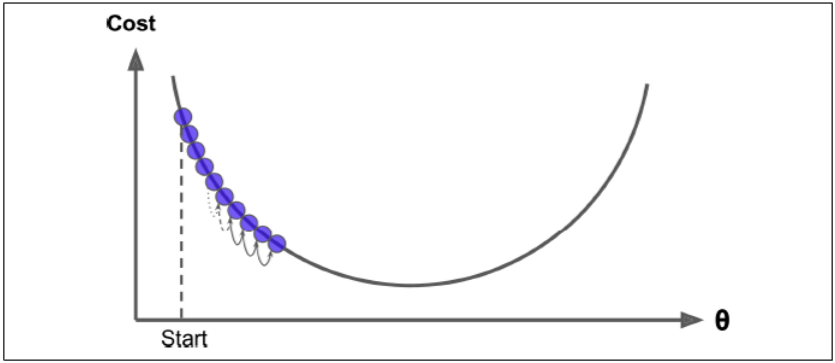
- Optimization in machine learning refers to the process of adjusting a model's parameters to minimize (or maximize) a given objective function.
- This function, often called the loss function, measures how well the model's predictions match the actual data.
- The goal is to find the best set of parameters that improve the model's performance.
- Although in this introduction the terms "loss" and "cost" are sometimes used interchangeably, it is worth emphasizing that from a more rigorous point of view when we talk about "loss" we refer to the error on the single observation, while when we talk about "cost" we refer to an aggregation (average, sum, ...) over the whole dataset.

- Gradient Descent is a very generic optimization algorithm capable of finding optimal solutions to a wide range of problems.
- Gradient Descent Algorithm measures the local gradient of the error function with regards to the parameter vector θ, and it goes in the direction of descending gradient.
- Once the gradient is zero, you have reached a stationary point (minimum, maximum, or saddle point). In convex problems, this guarantees a global minimum.
- Concretely, you start by filling θ with random values (this is called random initialization), and then you improve it gradually, taking one step at a time, each step attempting to decrease the cost function (e.g., the MSE), until the algorithm converges to a minimum

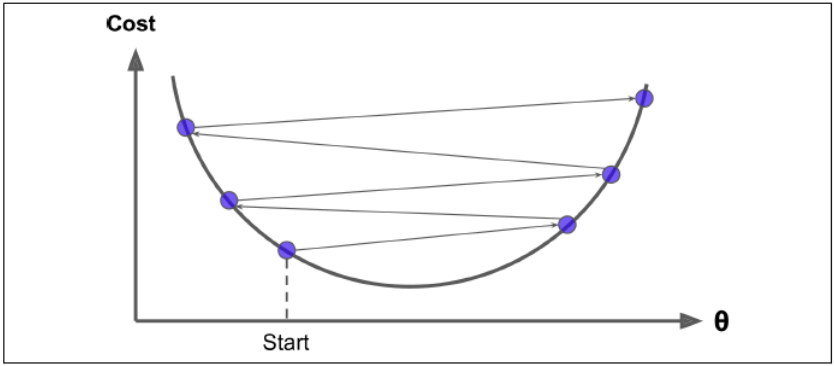
An important parameter in Gradient Descent is the size of the steps, determined by the learning rate hyperparameter.



If the learning rate is too small, then the algorithm will have to go through many iterations to converge, which will take a long time...



... on the other hand, if the learning rate is too high, you might jump across the valley. This might make the algorithm diverge failing to find a good solution.



Having defined an appropriate learning rate, the scheme works as follow

- Start with an initial guess for each parameter θ_k;
- Move θ_k in the direction of the slope

$$\text{New } \theta_k = \text{Old } \theta_k - \beta \frac{\partial J}{\partial \theta_k}$$

where β is our learning rate;

In the above description of gradient descent we have used all of the data points simultaneously. This is called **batch gradient descent**

Gradient Descent: What Does $\nabla J(\theta) = 0$ Mean?

- If $\nabla J(\theta) = 0$, θ is a **stationary point**.
- A stationary point can be:
  - a **local minimum**,
  - a **local maximum**,
  - a **saddle point**.

When does "zero gradient" guarantee a minimum?

- If $J(\theta)$ is **convex**, any stationary point is a **global minimum**.
- In **non-convex** problems (common in modern ML), gradient descent may converge to local minima or saddle points.

Why this matters in finance

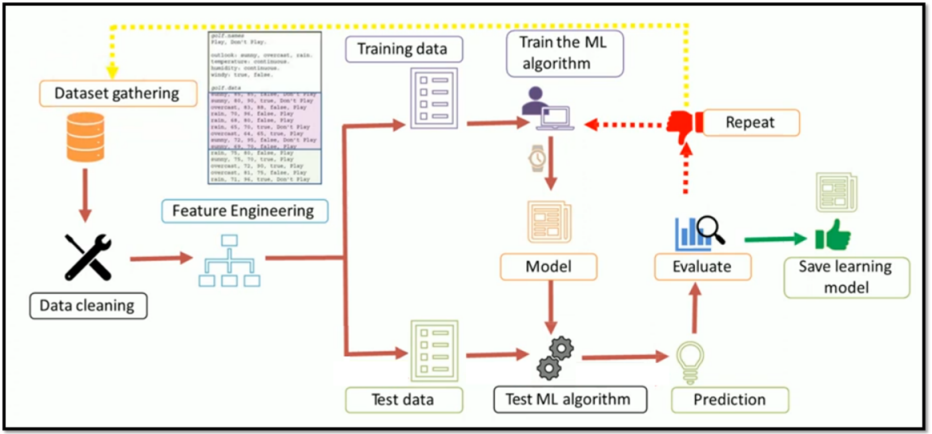
- Complex models (trees, neural nets) are typically non-convex or non-smooth: optimization and generalization must be treated carefully.

The Machine Learning Process

General Workflow of a ML Application

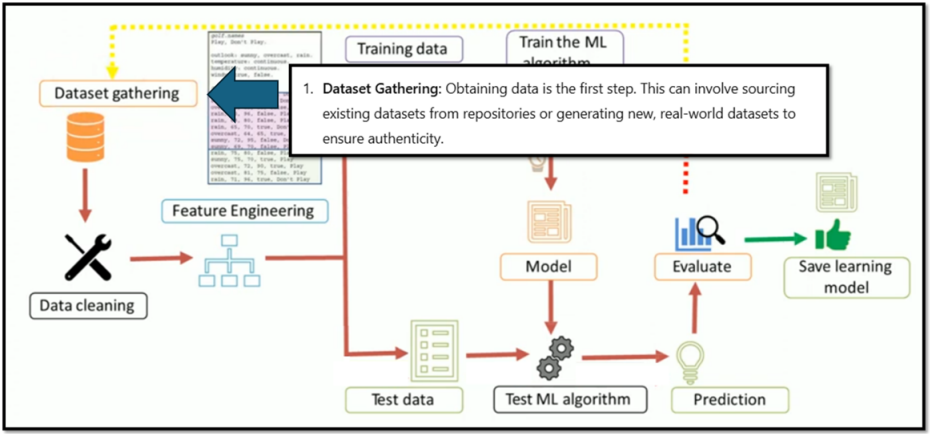
The Machine Learning Process

The development and deployment of ML models follow a structured, multi-step process:



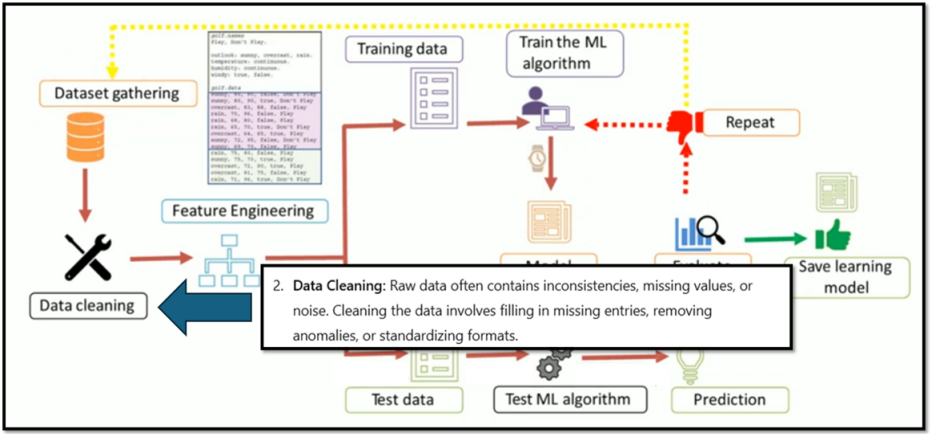
The Machine Learning Process

The development and deployment of ML models follow a structured, multi-step process:



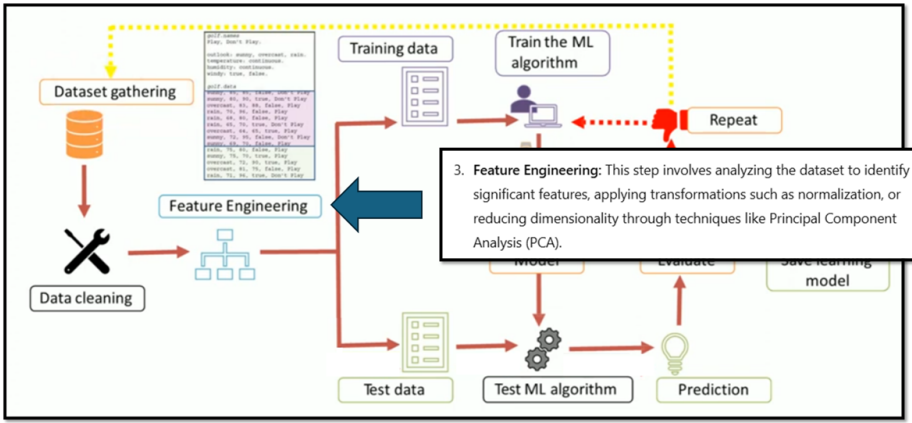
The Machine Learning Process

The development and deployment of ML models follow a structured, multi-step process:



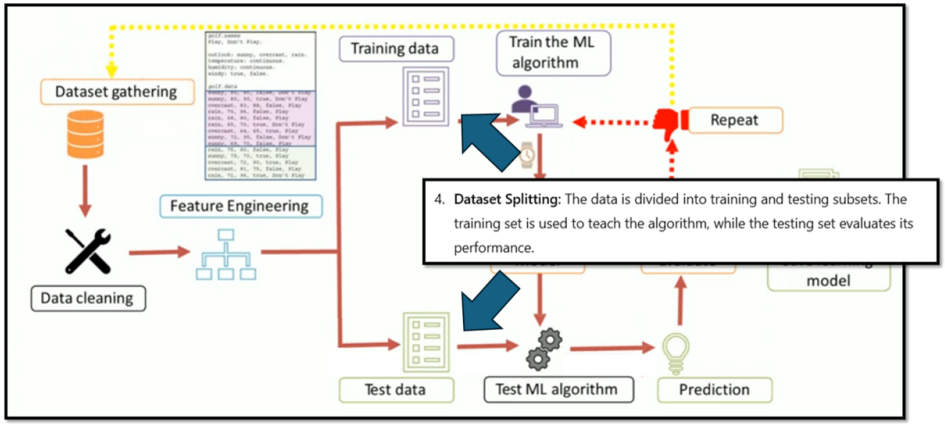
The Machine Learning Process

The development and deployment of ML models follow a structured, multi-step process:



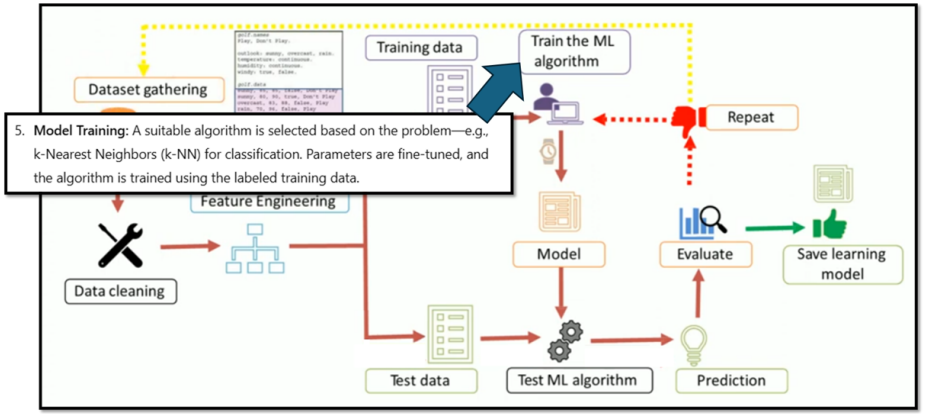
The Machine Learning Process

The development and deployment of ML models follow a structured, multi-step process:



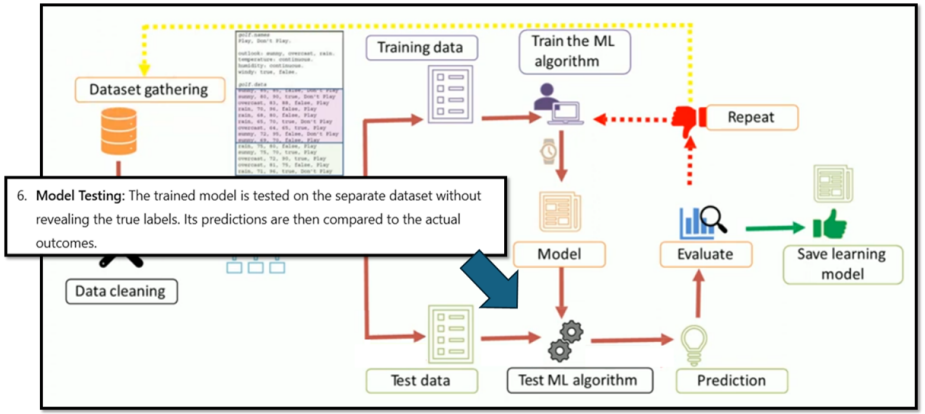
The Machine Learning Process

The development and deployment of ML models follow a structured, multi-step process:



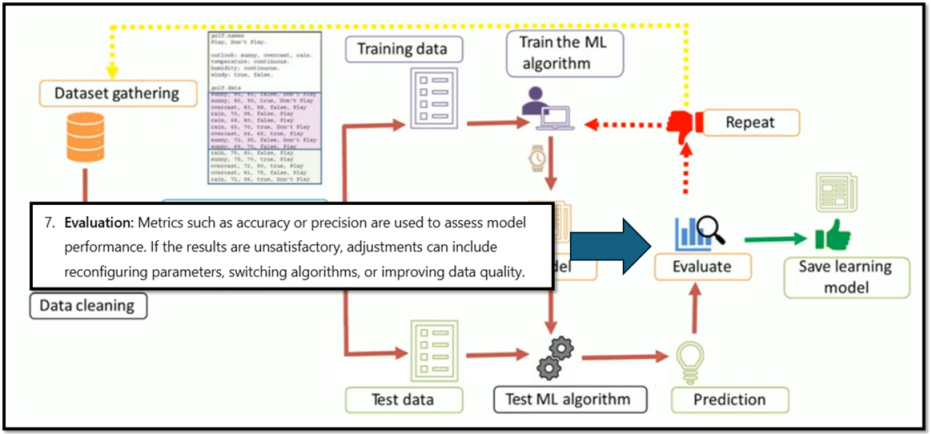
The Machine Learning Process

The development and deployment of ML models follow a structured, multi-step process:



The Machine Learning Process

The development and deployment of ML models follow a structured, multi-step process:



A Critical Finance Warning: Data Leakage

Definition

- **Data leakage** occurs when information from the future (or from the test set) inadvertently enters the training process.

Why it is especially severe in finance

- Financial time series are time-ordered: the model must only use information available at decision time.
- Small leakages can create *illusory* performance that disappears live.

Common leakage sources (we will revisit later)

- Random train/test splits on time series.
- Feature engineering computed using the full sample (e.g., normalization with full-sample mean/variance).
- Labels that use future information without proper alignment (e.g., using returns that overlap the feature window).

A Critical Finance Warning: Time Series Splitting

Standard ML split (often wrong for finance)

- Random split assumes i.i.d. data and breaks temporal causality.

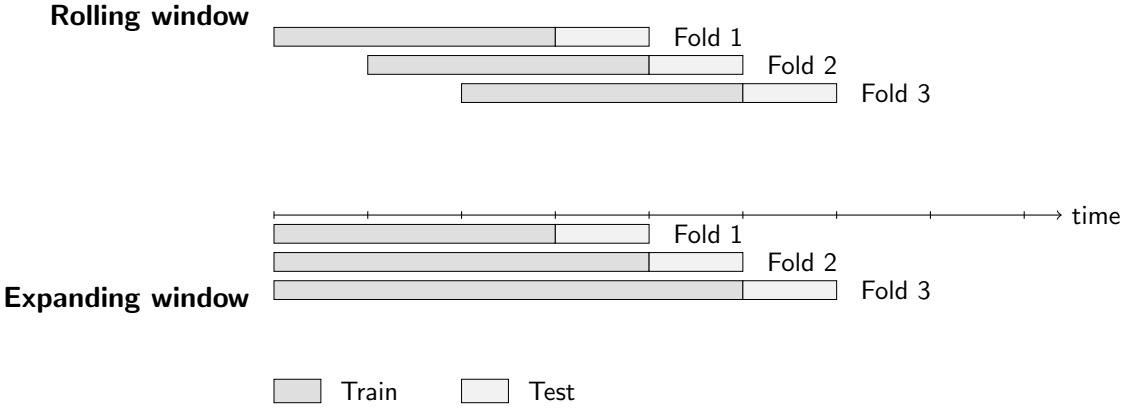
Preferred approach in finance: walk-forward (rolling) evaluation

- Train on a past window, validate/test on a future window, then roll forward.
- Two common schemes:
  - **Rolling window:** fixed-size training window.
  - **Expanding window:** training set grows over time.

Goal

- Mimic the real deployment setting: at time t, only data up to t can be used.

Walk-Forward Schemes: Rolling vs Expanding Window



- **Rolling window:** fixed-length training set, more adaptive to regime changes.
- **Expanding window:** growing training set, more data but potentially slower adaptation.

Overfitting and Generalization

Core idea

- A model can fit training data extremely well but perform poorly on unseen data.

Terminology

- **Training error:** performance on the data used to fit parameters.
- **Test error:** performance on unseen (future) data.
- **Generalization:** the ability to keep good performance out-of-sample.

Why overfitting is a major risk in finance

- Low signal-to-noise ratio, regime changes, non-stationarity.
- Many features and repeated trials can produce *data-mining bias*.

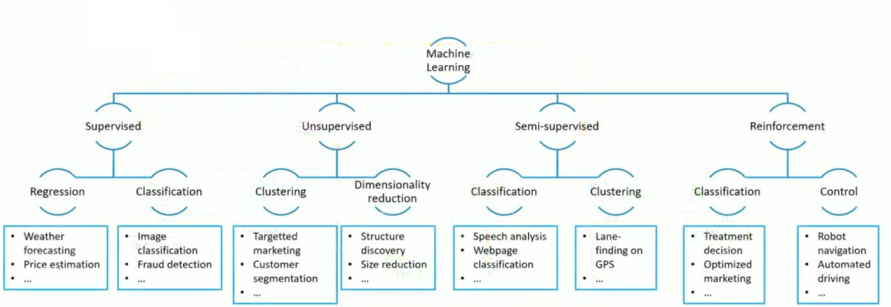
Typical remedies (we will cover later)

- Cross-validation adapted to time series, regularization, simpler models, and strong baselines.

Types of Machine Learning Models

Types of Machine Learning

ML algorithms can be broadly categorized into four types:

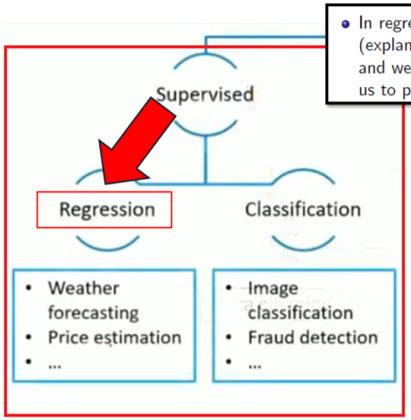


Supervised Learning

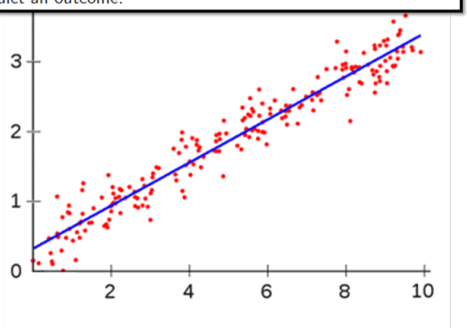
- In supervised learning, the algorithm is trained on labeled data, meaning both input variables (features) and corresponding output variables (labels) are provided.
- The goal is to learn the mapping between inputs and outputs.

Supervised learning includes:

- **Regression**: Predicting continuous numerical values, such as temperatures or stock prices.
- **Classification**: Categorizing data into discrete groups, such as identifying whether an email is spam or not.

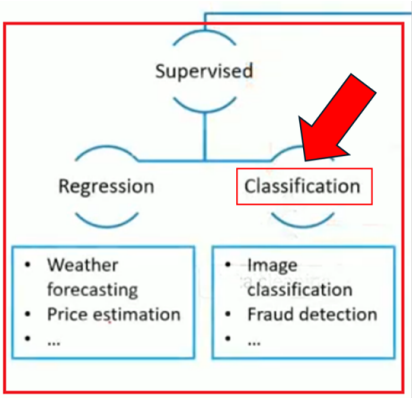


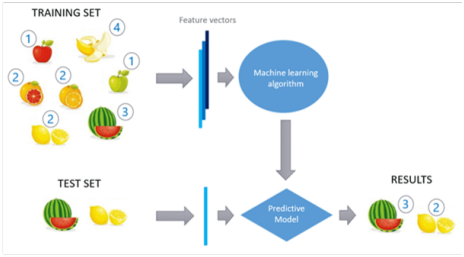
- In regression analysis, we are given a number of predictor (explanatory) variables and a continuous response variable (outcome), and we try to find a relationship between those variables that allows us to predict an outcome.



Supervised Learning

Types of Machine Learning





- From a formal poin of view, supervised learning process involves input variables, which we call X, and an output variable, which we call Y.
- We use an algorithm to learn the mapping function from the input to the output.
- In simple mathematics, the output Y is a dependent variable of input X as illustrated by:

$$Y = f(X)$$

Here, our end goal is to try to **approximate the mapping function** f, so that we can **predict** the output variables Y when we have new input data X.

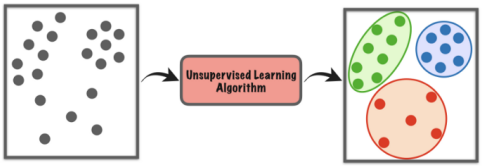
Unsupervised Learning

- Unsupervised learning involves analyzing data without labeled outputs.
- The algorithm identifies hidden patterns or structures.

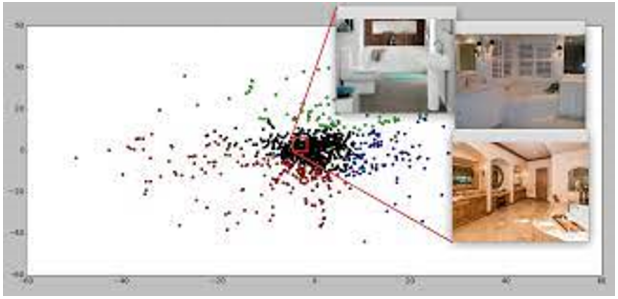
Two primary uses include:

- **Clustering**: Grouping similar data points, for instance, segmenting customers for targeted marketing.
- **Dimensionality Reduction**: Simplifying datasets by reducing features while preserving essential information, often used to speed up computations or eliminate noise.

- Unsupervised algorithms attempt to find some sort of underlying structure in the data.
- Are some observations clustered into groups? Are there interesting relationships between different features? Which features carry most of the information?



- The data for **Unsupervised Learning** consists of features but no labels because the model is being used to identify patterns not to forecast something.
- For example we could use an unsupervised model **to group** (and **NOT to classify**) the houses that exist in a certain neghborhood without trying to predict any price.



Semi-Supervised Learning

- Semi-supervised learning combines elements of supervised and unsupervised methods.
- It operates on datasets where only some entries are labeled. Techniques may include assigning labels to unlabeled data or training the model iteratively on both labeled and unlabeled subsets.
- Applications include speech analysis and web page classification.

Reinforcement Learning

- Reinforcement learning involves algorithms that learn through interaction with their environment.
- These models improve by receiving rewards or penalties for their actions, refining their strategies over time.
- Applications range from training autonomous vehicles to optimizing marketing strategies.

Reinforcement Learning in Finance: Typical Use-Cases

A realistic perspective

- In finance, RL is more common in **control** and **execution** problems than in direct price prediction.

Examples

- **Optimal execution / liquidation:** minimize costs and market impact.
- **Market making:** manage inventory risk while quoting bid/ask.
- **Portfolio rebalancing:** dynamic allocation with transaction costs and constraints.

Key message

- RL is powerful but sensitive to model risk, non-stationarity, and realistic simulation assumptions.

Conclusions

Conceptual Summary

- **Machine Learning must be a process**, not a black box:

$$\text{Data} \rightarrow \text{Model} \rightarrow \text{Loss} \rightarrow \text{Optimization}$$

- A model does not learn "the truth": it learns **parameters** that minimize a **chosen objective**.
- In finance, the key mindset shift is:

Prediction accuracy is not always equivalent to **Economic value**

- ML in finance is often **decision support** (risk/cost/constraints aware), not only forecasting.

Core ML Building Blocks (Finance View)

- **Features** X: information available **at decision time** (no future information).
- **Labels** Y: what the model is asked to predict (return, direction, risk, regime, default, ...).
- **Loss / Cost function**: defines what "error" means.
  - *Statistical loss*: MSE, log-loss, accuracy, ...
  - *Economic loss*: transaction-cost aware P&L, drawdown control, risk-adjusted objective, ...
- **Optimization**: how parameters are adjusted (e.g., gradient descent).

Finance warning: a poorly chosen objective can produce a statistically good model with systematically bad economic outcomes.

Finance Takeaways: What Can Go Wrong (Even in "Intro" ML)

- **Data leakage** can create illusory performance that disappears live.
- **Time ordering matters**: standard random train/test splits often break temporal causality.
- **Non-stationarity and regime changes** imply that good in-sample fit can be meaningless.
- **Overfitting risk is structurally high** due to low signal-to-noise and repeated trials.

Operational mindset: in finance, **validation design** and **data handling** can matter more than model sophistication.

Bridge to Lesson 1.2 — Data Gathering with Pandas

- In Lesson 1.1, we introduced the ML workflow and why finance requires extra care (leakage, time splits, non-stationarity).
- **Next step:** build the **data layer** for ML in finance:
  - gathering market and macro data with **Pandas** and related tools,
  - understanding **financial data structures** (OHLC bars and alternatives),
  - performing **transformations** (prices vs returns) and sanity checks,
  - learning typical issues with public data (missing values, revisions, survivorship bias).
- **Key message for Lesson 1.2:** financial data are **not generated for ML**; therefore **preparation and validation** are first-class modeling steps.