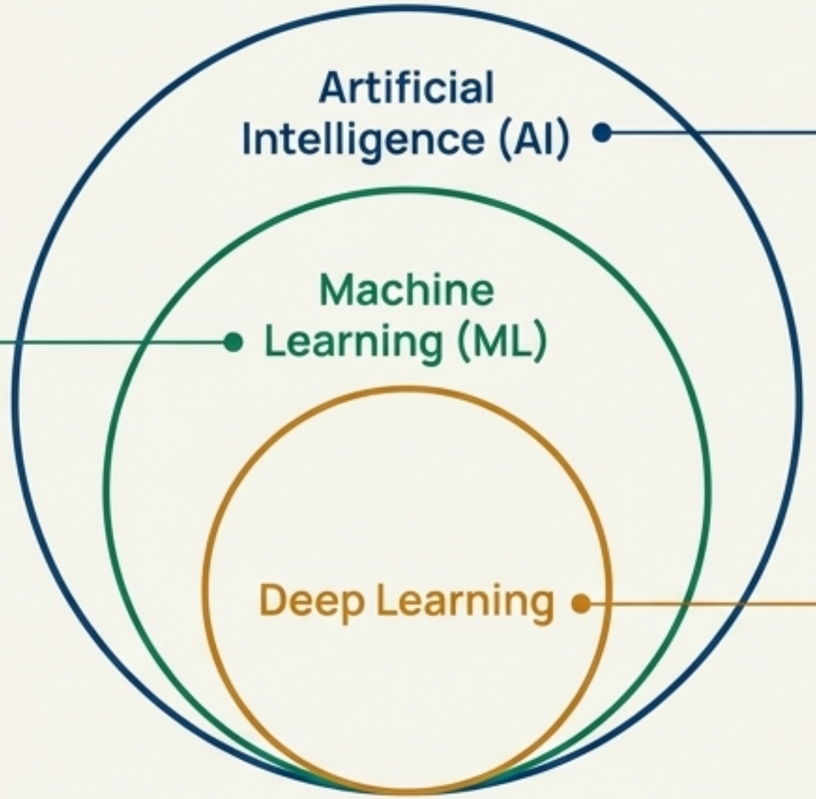
From Concepts to Creation: A Practical Journey into Modern AI

An essential guide for technical professionals on the evolution from traditional Machine Learning to Generative AI.

Understanding the Landscape: AI, Machine Learining, and Deep Learning



The capability of a machine to imitate intelligent human behavior, enabling it to learn, reason, and solve problems autonomously.

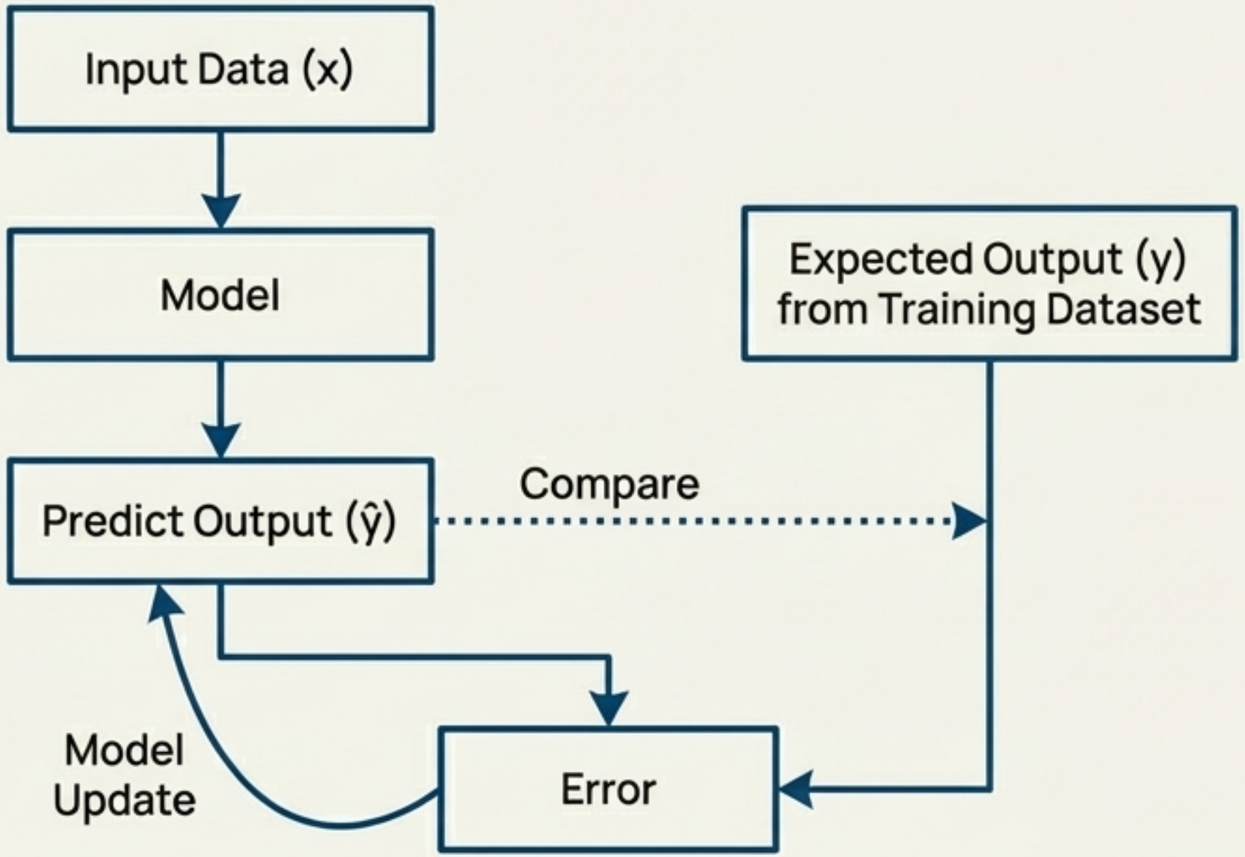
Machine Learning (ML)
A subset of AI that enables systems to learn from data, identify patterns, and make decisions without explicit programming.

A specialized subset of ML that uses deep neural networks (with multiple layers) to analyze and learn from vast amounts of data.

These terms are not interchangeable. Machine Learning is the engine that powers most modern AI, and Deep Learning is a powerful technique within ML.

The Foundation: How Supervised Learning Works

Supervised Learning uses **labeled data** to train models. The model learns by comparing its predicted output to the expected, correct output and adjusting until the error is minimized.



1. Regression

Predicting a continuous numerical value (e.g., house price, temperature, number of defects).

2. Classification

Predicting a discrete category or label (e.g., Spam/Not Spam, Fraud/Not Fraud, defect type).

The Challenge: A Common Software Testing Scenario

An Interview Question for Technical Leads

> **“Assume that you have got only 2 days to test an entire regression suite of 1000+ tests (only 20% of them are automated), what will be your strategy?”**

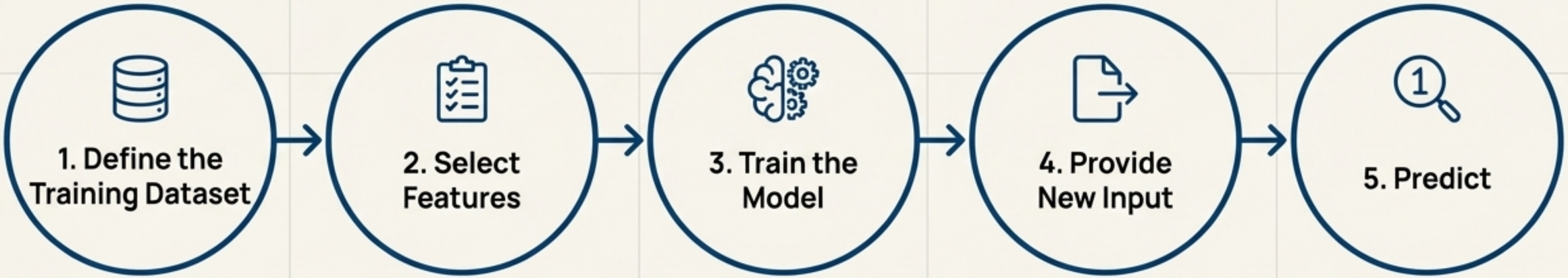
Context: This isn’t just a time management problem; it’s a risk assessment problem. How do you prioritize where to focus limited manual testing resources for maximum impact?

A traditional approach might involve guessing based on module complexity or recent changes. A data-driven approach is better.

The Solution with Regression: Defect Prediction Modeling

Instead of relying on intuition, we can use a regression model to predict the *number of expected defects* for each upcoming code change. This allows us to rank and prioritize test cases based on risk.

How It Works: The 5-Step Process



1. Define the Training Dataset

Extract historical metadata from Git/Bitbucket commits and label it with the actual number of defects found.

2. Select Features

Identify the key input variables from the commit metadata.

3. Train the Model

Use a supervised regression algorithm to learn the relationship between features and defect count.

4. Provide New Input

Feed the model metadata from a new, unseen commit.

5. Predict

The model analyzes the new commit and outputs a predicted number of defects.

Under the Hood: The Data Behind Defect Prediction

Feature Selection (The Inputs)

- **Commit_ID**: Unique commit hash
- **Developer_Name**: Name of the developer
- **Core_Banking_Module**: Area affected (Payments, Loan, etc.)
- **Lines_Changed**: Number of lines modified
- **Files_Changed**: Number of files affected
- **Code_Churn (%)**: Percentage of frequently changed lines
- **Time_Since_Last_Commit (hrs)**: Hours since the last commit
- **Branch_Type**: Feature/Hotfix/Release
- **PR_Review_Score (1-5)**: Pull request review score
- **Merged_Without_Review**: 1 for Yes, 0 for No
- **Reverted_Commit**: 1 for Yes, 0 for No
- **Past_Defects_In_Module (%)**: Percentage of previous defects in the module

Target Variable (Y): Predicted_Defects (Count) – A numeric count, making this a regression problem.

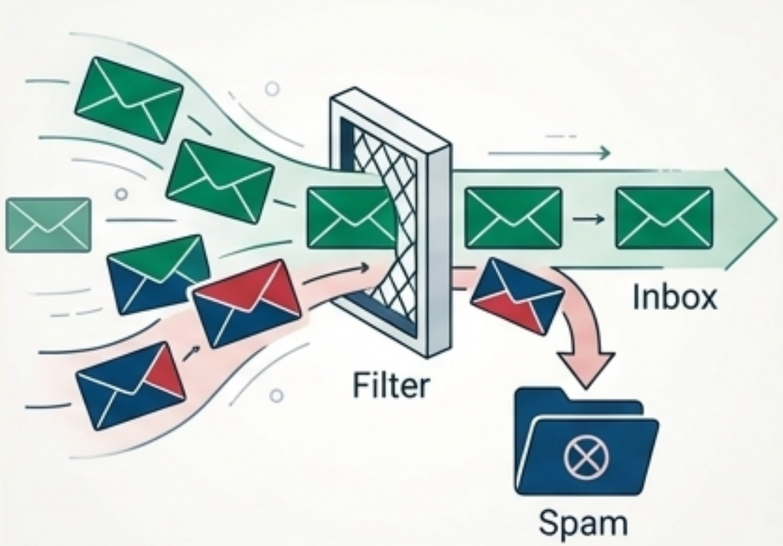
Example Training Dataset

Commit_ID (Hash)	Developer_Name	Core_Banking_Module	Lines Changed	Files_Changed	Code_Churn (%)	Time_Since_Last_Commit (hrs)	Branch_Type (Feature/ Hotfix/ Release)	PR_Review_Score (1-S)	Merged_Without_Review (1=Yes, 0=No)	Reverted_Commit (1= Yes, 0=No)	Past_Defects_In_Module (%)	Predicted_Defect (Count)
a1b2c3d4	Kumaravel	Payments Processing	450	8	55	72	Hotfix	2	1	0	30	5
b2c3d4eS	Meenakshi	Loan Management	120	3	10	20	Feature	4	0	0	12	1
c3d4e5f6	Murugan	Ledger Balancing	500	7	60	90	Release	1	1	1	40	7
d4e5f6g7	Venkatesh	Interest Calculation	90	2	5	12	Feature	5	0	0	8	0
e5f6g7h8	Selvamani	KYC Compliance	300	6	45	48	Hotfix	3	1	0	25	3
f6g7h8l9	Lakshmi	Ledger Balancing	275	5	35	55	Release	2	1	1	28	4

The Other Side of the Coin: Classification Algorithms

A classification algorithm predicts a **discrete category or label** based on input data. The output is not a number, but a group or class.

The Classic Example: Spam Detection



Input: Email content, sender, subject line.

Features: Presence of certain keywords (“**prize**,” “**free**”), sender reputation, link patterns.

Output Labels: “**Spam**” or “**Not Spam**”.

This same logic can be applied to complex problems in software development, such as triaging test automation failures.

Classification in Practice: Predicting Test Failure Root Cause

The Problem

A test automation suite runs overnight, and 100 tests fail. Triaging them is time-consuming. Are the failures due to a real bug, a flaky script, a bad environment, or incorrect test data?

The Solution with Classification

A model can be trained on historical test failure logs to automatically classify each new failure into a root cause category.

Key Features from Test Logs (The Inputs):

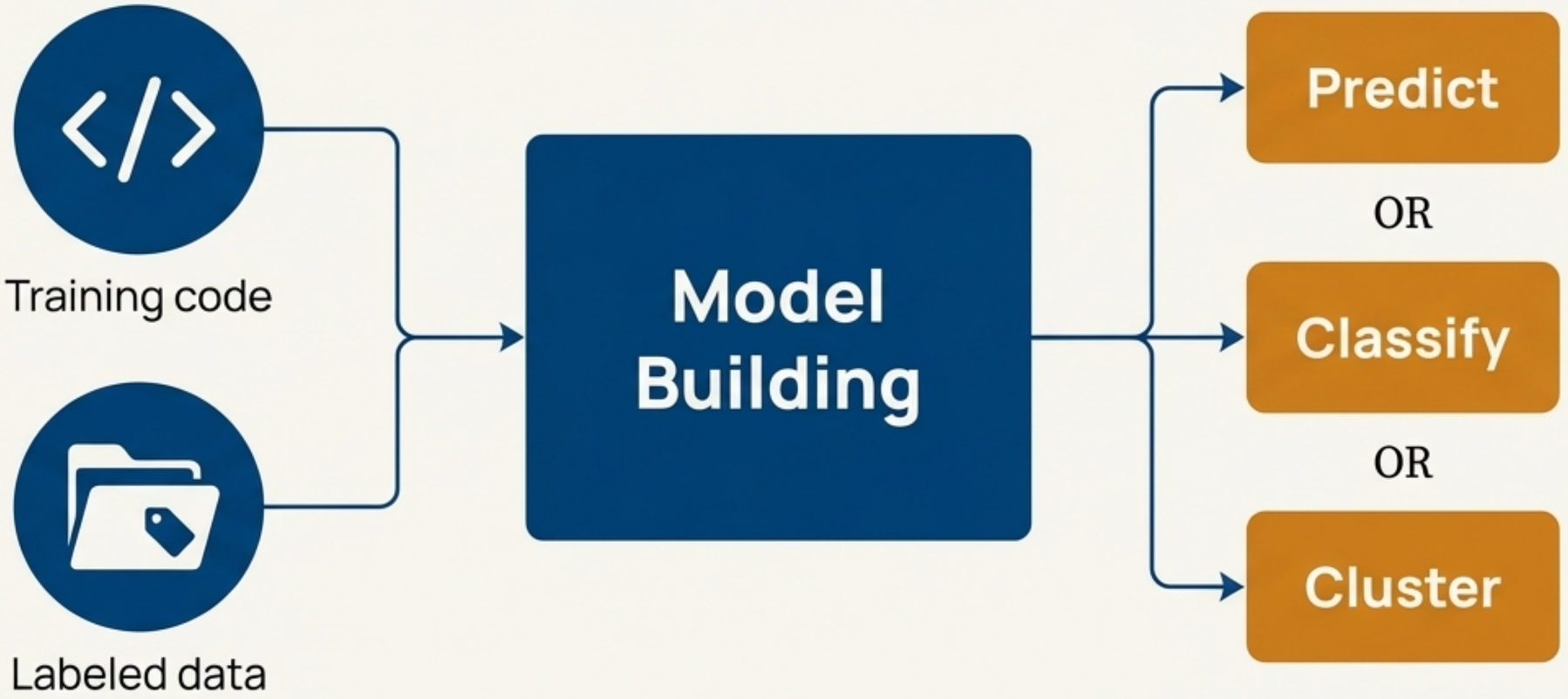
- **Exception_Type:** (e.g., `NoSuchElementException`, `TimeoutException`)
- **Failed_Step_Time (ms):** How long it took for the step to fail.
- **Browser_Console_Error:** Were there any front-end errors?
- **App_Log_Error:** Were there any application-side server errors?
- **Stack_Trace_Keywords:** Textual hints from the stack trace.
- **Past_Failures (%):** Historical flakiness for the same script.

Target Variable (The Output Categories):

Defect_Classification
(e.g., TS1: Locator Issue, TD2: Incorrect Data, EN2: Network Latency).

Defect_Classification
TS1 (Locator Issue)
TS4 (Wait Timeout)
TS2 (Incorrect Assertion)
TD1 (Missing Test Data)
EN5 (Incorrect Environment Setup)
EN2 (Network Latency Issue)

Supervised Learning Recap: From Labeled Data to Actionable Insights



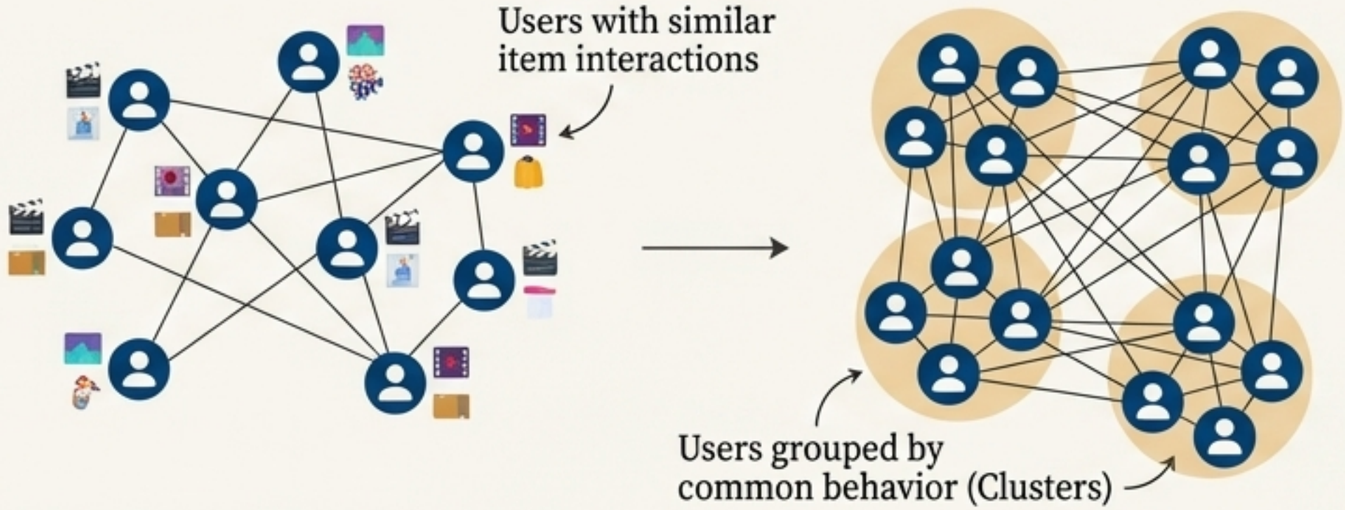
By learning from historical, labeled data, we can build models that automate complex decision-making processes, saving time and improving accuracy in software development and testing.

Beyond Labels: Unsupervised Learning and Pattern Discovery

Unsupervised models work with unlabeled data. Their goal is not to predict a known output, but to find inherent structures, groupings, or patterns within the data itself.

The Classic Example: Recommender Systems

How does Netflix or Amazon know what you want to watch or buy next? They use unsupervised learning to group you with other users who have similar behaviors.



Key Technique: Collaborative Filtering

- The model groups users based on common behavior (e.g., watching the same movies, buying the same products) without any explicit labels.
- If User Group A watches Movie X and Movie Y, and a new user watches Movie X, the system recommends Movie Y. The pattern is discovered automatically based on user similarity.

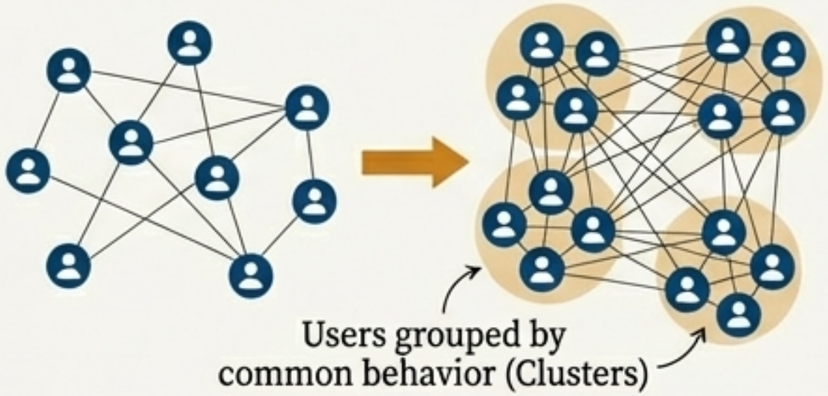
The Hybrid Approach: Combining Learning Methods

The most powerful systems, like those at Netflix and Amazon, use a mix of both supervised and unsupervised learning.

Unsupervised (Collaborative Filtering)

Suggests items based on hidden user patterns and similarities. This is great for discovering new interests.

**Users who watched 'Maharaja' also watched 'Meiyazhagan'."*



Supervised (Content-Based Filtering)

Uses explicit preferences (the 'labels') like ratings, purchase history, or watch time to make recommendations. This is great for refining suggestions based on known tastes.

**Because you gave 'Maharaja' a 5-star rating, we are recommending other Thriller movies."*

The two approaches solve different parts of the recommendation problem, and their combination provides a more robust and personalized experience.

A Shift in Objective: From Discriminating to Generating

The Story So Far: *Training Methods*

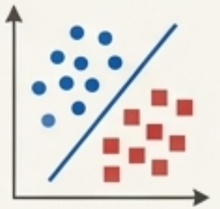
We've explored the *training methods* for ML models.

- **Supervised:** Learning from a "teacher" with labeled data.
- **Unsupervised:** Finding patterns on its own in unlabeled data.

The Next Question: What is the model's ultimate *goal*?

The Model's Goal

Discriminative Models

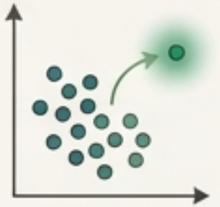


Their goal is to learn the boundary between different groups of data.

They are trained to make a prediction or classify a data point. (e.g., Is this fraud or not fraud? What is the price of this house?).

This is what we've mostly seen so far.

Generative Models



Their goal is to learn the underlying distribution of the data itself.

They are trained to ***create new data samples*** that are statistically similar to the training data.

This is the fundamental difference that enables Generative AI.

The New Frontier: Generative AI and Large Language Models (LLMs)

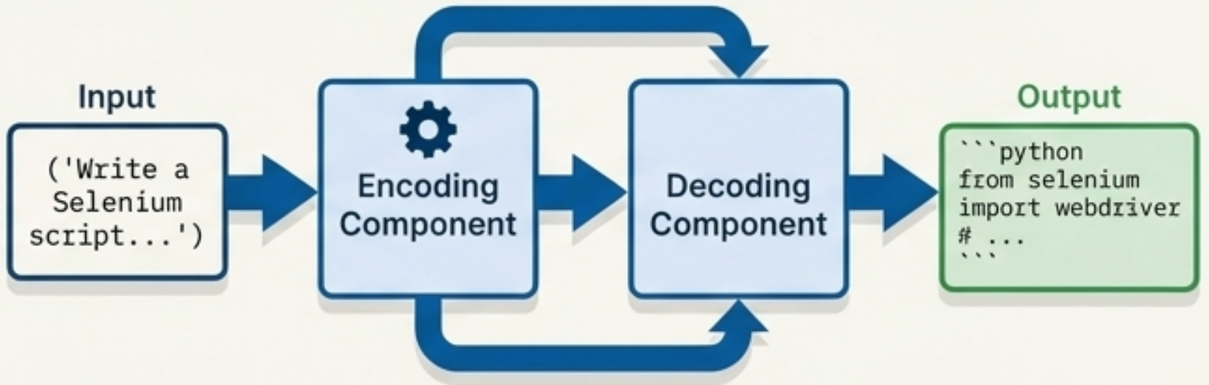
What is Generative AI?

A class of AI systems that can create new, original content (text, images, code, etc.) by learning the patterns and structure from existing datasets.

How does it work? The GPT Model

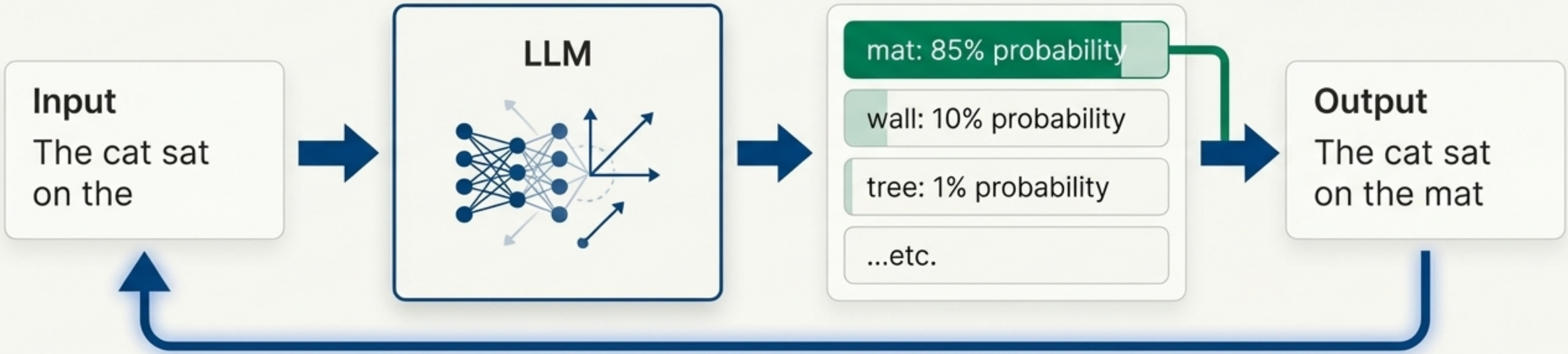
GPT stands for Generative Pre-trained Transformer.

- **Pre-training:** The model is first trained on a massive amount of unlabeled data (like the entire internet) in an unsupervised way. It learns grammar, facts, and reasoning abilities. This creates the foundational LLM.
- **Transformer:** This is the neural network architecture that makes LLMs so powerful. It uses encoding and decoding components to process input and generate the most probable output.



How an LLM ‘Thinks’: A Step-by-Step Prediction

At its heart, an LLM is a next-word prediction engine on a massive scale. It calculates the probability of the most likely next word (or ‘token’) given the sequence of words that came before it.



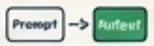
Complex and creative responses emerge from this simple, iterative process of predicting the next most likely token.

The Paradigm Shift: From Training a Model to Designing a Prompt

The “Old” Way: Traditional ML

-  Requires deep ML expertise.
-  Requires collecting and labeling training examples.
-  Requires explicitly training or fine-tuning a model for a specific task.
-  Requires significant compute time and hardware for training.
- $\min_{n \to 0}(x - c)$ The core skill is minimizing a loss function.

The “New” Way: Prompting a Pre-trained LLM

-  Does NOT require deep ML expertise to get started.
-  Does NOT require a large set of custom training examples.
-  Does NOT require training a new model from scratch.
-  A single, large, pre-trained model can be used for many different tasks.
-  The core skill is shifting to **Prompt Design**: crafting effective instructions to guide the model's output.

The barrier to leveraging AI is lowering, but the value is shifting from building models to effectively interacting with them.