Contents

[AI, ML AND DEEP LEARNING 3](#_Toc202808396)

[WHAT IS MACHINE LEARNING 3](#_Toc202808397)

[MACHINE LEARNING TERMINOLOGIES 4](#_Toc202808398)

[DEEP LEARNING 5](#_Toc202808399)

[GENERATIVE AI 6](#_Toc202808400)

[MACHINE LEARNING MODEL 11](#_Toc202808401)

[MACHINE LEARNING PROCESS 11](#_Toc202808402)

[MACHINE LEARNING MODEL AND DATASET 12](#_Toc202808403)

[DATA SET 12](#_Toc202808404)

[FEATURES AND LABELS 13](#_Toc202808405)

[TRAINING ML MODELS 15](#_Toc202808406)

[FEATURE ENGINEERING 15](#_Toc202808407)

[MACHINE LEARNING ALGORITHMS 15](#_Toc202808408)

[MACHINE LEARNING MODELS 16](#_Toc202808409)

[MACHINE LEARNING ALGORITHMS CLASSIFICATION 24](#_Toc202808410)

[GENERATIVE AI 24](#_Toc202808411)

[DEEP LEARNING ARCHITECTURE 26](#_Toc202808412)

[DIFFERENT ARCHITECTURE OF DEEP LEARNING 26](#_Toc202808413)

[RECURRENT NEURAL NETWORKS (RNNS) 26](#_Toc202808414)

[CONVOLUTIONAL NEURAL NETWORKS (CNNS) 26](#_Toc202808415)

[TRANSFORMER ARCHITECTURE 27](#_Toc202808416)

[FOUNDATION MODEL 28](#_Toc202808417)

[KEY CHARACTERISTICS 29](#_Toc202808418)

[LARGE LANGUAGE MODELS(LLMs) 30](#_Toc202808419)

[MODEL TYPES (LLM TYPES) 30](#_Toc202808420)

[HOW LLMs WORKS? 32](#_Toc202808421)

[APPLICATION OF GEN-AI 35](#_Toc202808422)

[GEN-AI USE CASES 35](#_Toc202808423)

[GEN-AI LIMITATIONS & CHALLENGES 35](#_Toc202808424)

[MACHINE LEARNING 37](#_Toc202808425)

[MACHINE LEARNING PROCESS 37](#_Toc202808426)

[TRAINING SET AND TEST SET 38](#_Toc202808427)

[FEATURE SCALING 39](#_Toc202808428)

[NORMALIZATION 40](#_Toc202808429)

[STANDARDIZATION 41](#_Toc202808430)

[EXAMPLE - NORMALIZATION 41](#_Toc202808431)

[DATA PRE-PROCESSING USING PYTHON 42](#_Toc202808432)

[DATA PROCESSING USING PYTHON 47](#_Toc202808433)

[REGRESSION 47](#_Toc202808434)

[SIMPLE LINEAR REGRESSION 48](#_Toc202808435)

[MULTIPLE LINEAR REGRESSION 48](#_Toc202808436)

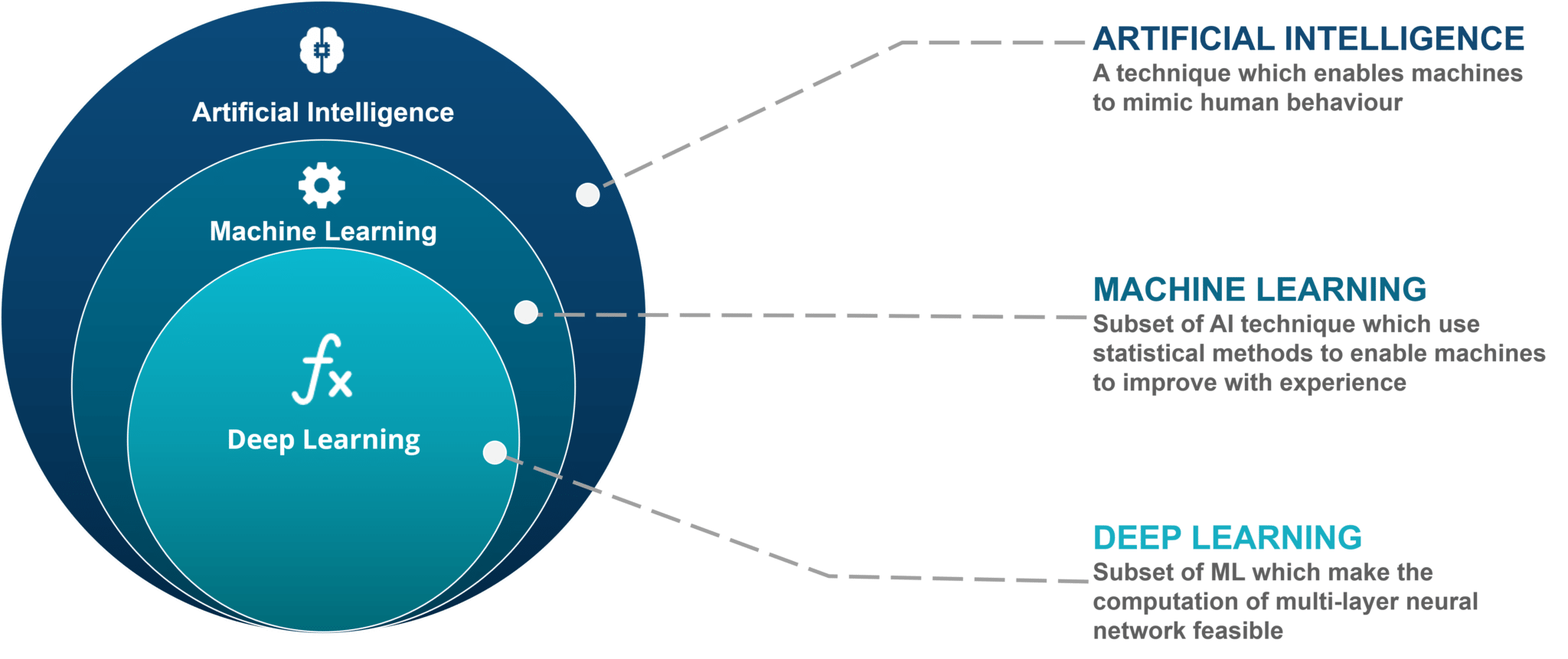
[POLYNOMIAL REGRESSION 48](#_Toc202808437)

[SUPPORT VECTOR REGRESSION 48](#_Toc202808438)

[DECISION TREE REGRESSION 48](#_Toc202808439)

[RANDOM FOREST REGRESSION 48](#_Toc202808440)

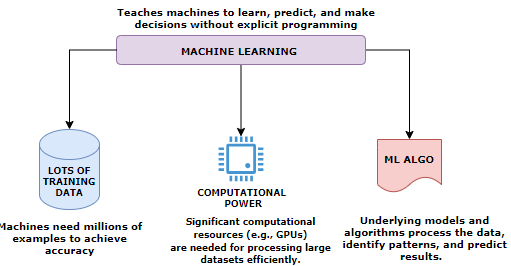
# AI, ML AND DEEP LEARNING



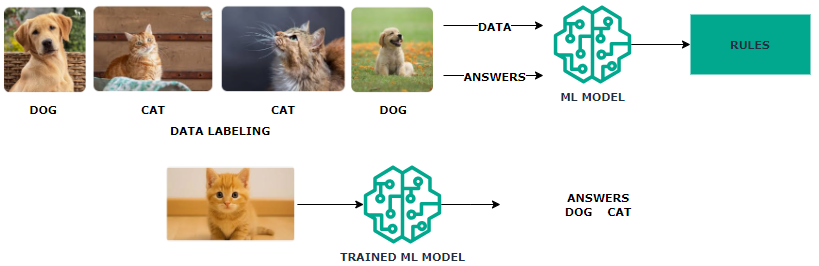
* Artificial intelligence is like the umbrella concept of machines that can ***perform complex intellectual tasks that are usually performed only by human***s.Those machines can mimic or simulate human cognitive functions.
* Inside the AI circle, a sub-field evolved which is called machine learning. Machine learning is adding the missing part, **the important self-learning capability to machines**. Using machine learning we can now handle much more complex scenarios while learning from the data. Instead of using rules based programming, we let the machines learn from the data.
* Deep learning is a sub-field of machine learning.(More details below)

# WHAT IS MACHINE LEARNING

* Machine learning is a subfield of artificial intelligence.
* It involves training computers on large amounts of data.
* Computers learn from data and make predictions or decisions.
* It allows computers to recognize patterns and improve their performance over time.
* Machine learning is used in various applications like image and speech recognition, recommendation systems, and predictive analytics.



## MACHINE LEARNING TERMINOLOGIES



* **Machine Learning Model** learns by extracting patterns from the existing data, establishing rules automatically.
* In above example with multiple images of “dogs” & “cats” we train the model by labeling the training data. This common type of machine learning where the system is trained with labeled data (e.g., pictures of dogs and cats) is called **Supervised Learning**
* After Training the **Trained Model** can be used to classify new, unseen data.
* Machine learning systems adapt to new data without changing the program's code.

### DEEP LEARNING

A close-up of a human head

AI-generated content may be incorrect.

* Deep learning is a subset of Machine Learning based on neural networks which mimic the structure of human neurons.
* Neural networks consist of multiple computational layers.
* Each layer processes and enhances the data before passing it to the next layer.
* This multi-layer structure improves accuracy for complex tasks (e.g., text generation, image generation).
* *Neural network is the model that generative tools like ChatGPT are also using*.

#### ADVANTAGES OF DEEP LEARNING

* **Higher Accuracy**: Suitable for solving complex problems.
* **Generative AI Application**: Tools like ChatGPT use deep learning neural networks to enhance text generation accuracy.
* **Utilization of GPUs**: Recent advancements in computational power have made deep learning models more accessible and efficient.

### GENERATIVE AI

A diagram of machine learning

AI-generated content may be incorrect.

* **Deep Learning is a subset of that uses neural networks to generate new content like text, images, videos, or audio.**
* **Unlike conventional AI, Generative AI doesn't just classify or predict data; it generates brand-new content based on its training data.**

#### CONVENTIONAL AI VERSUS GENERATIVE AI

**CONVENTIONAL AI**

A blue and white logo

AI-generated content may be incorrect.

* Operates by learning from training data and making predictions, classifications, or performing language processing/computer vision.
* Example: Trained on apple images, it tells whether a supplied image is of an apple.

**GENERATIVE AI**

A blue and white logo

AI-generated content may be incorrect.

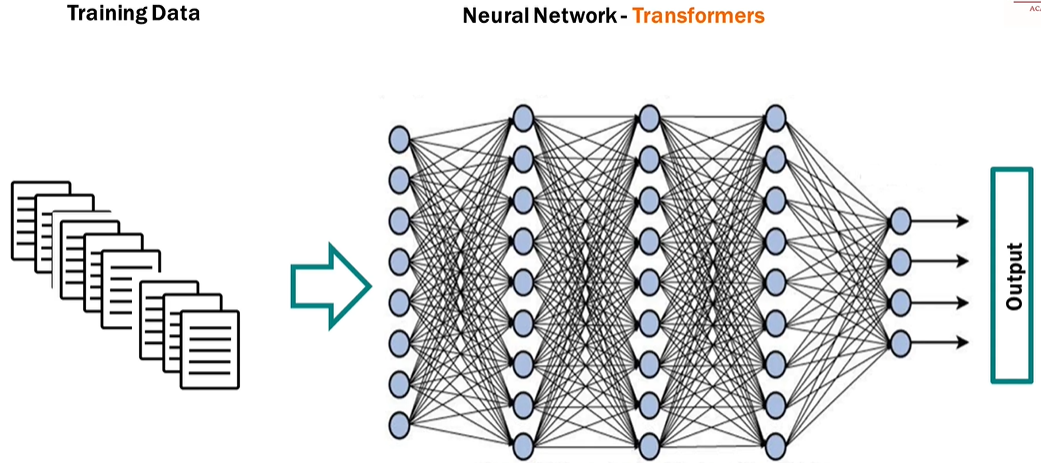
* Learning from training data and creates *new* content.
* Example: Trained on apple images, it generates a new apple image not extracted from the training data.

#### KEY TERMINOLOGIES

##### LARGE LANGUAGE MODEL

* LLMs are AI models designed to understand and generate human-like text.
* Unlike general generative AI (which handles text, images, audio, video, etc.), **LLMs deal exclusively with text—words, grammar, sentences, and contextual understanding.** Example: ChatGPT is a strong example of an LLM.

###### HOW LLMS WORK



* **NEURAL NETWORKS**:
  + LLMs use \*\*transformers\*\*, which are specialized neural networks designed to understand language, meaning, and context.
  + Transformers enable LLMs to process and generate text efficiently.
* **TRAINING**:
  + LLMs are trained on vast amounts of text data (e.g., Wikipedia, blogs, websites) and learn patterns, semantics, and context.
  + GPT-3, for instance, was trained on over \*\*500GB of data\*\* and used \*\*175 billion parameters\*\*; Google’s PaLM has \*\*540 billion parameters\*\*.
  + Training is crucial—larger datasets and more parameters improve accuracy and performance.
* **SEQUENTIAL WORD PREDICTION**:
  + LLMs generate output one word at a time, predicting the next word in sequence, resulting in sentences and paragraphs.

##### EMBEDDINGS IN GENERATIVE AI

* Embeddings are numerical representations of text—a way to convert words into numbers so machines can understand and process them.
* Machines do not inherently understand text; they operate using numbers. Embeddings allow machine learning models to interpret meaning, context, and relationships between words.

**KEY FUNCTIONALITY OF EMBEDDINGS**

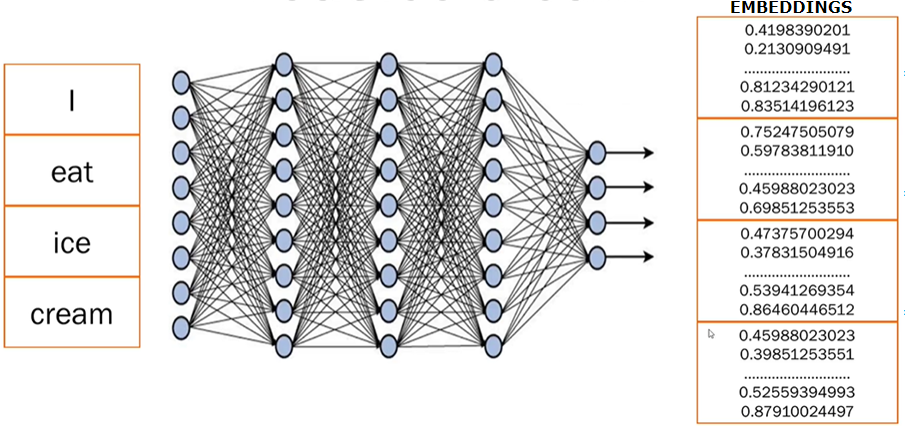
1. **CAPTURING MEANING**: Embeddings reflect the semantic meaning of words or sentences.
2. **CONTEXT UNDERSTANDING**: Embeddings account for the context of text (e.g., same word in different sentences/sentiments - “great” in sarcasm vs happiness).



1. **RELATIONSHIPS**:

* Words with strong associations (e.g., "ice" and "cream") have similar embeddings, reflecting their connection.
* Models understand the proximity and sequence of words based on embeddings.

**HOW EMBEDDINGS ARE GENERATED**



1. **STEP 1:** **BREAKING TEXT INTO TOKENS**

* Sentences are broken into smaller pieces or tokens (e.g., splitting "I eat ice cream" into 4 tokens—"I", "eat", "ice", "cream").

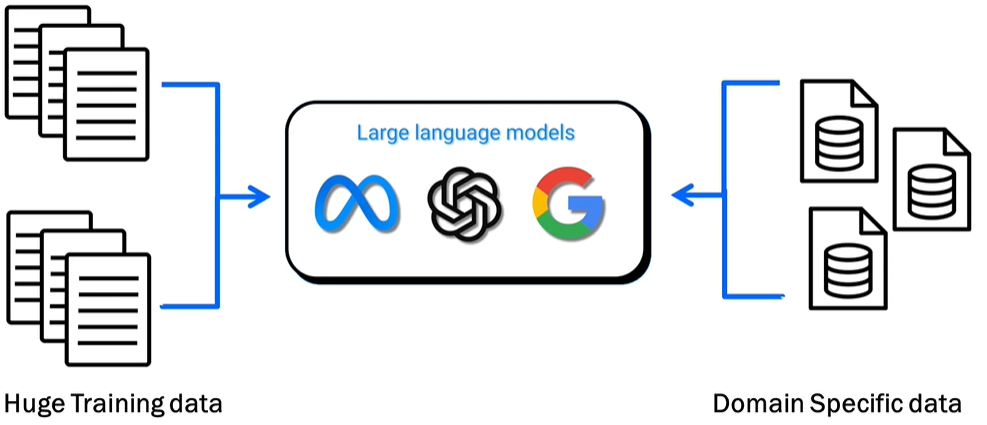
2. **STEP 2: NEURAL NETWORK PROCESSING**:

* Trained \*\*transformer models\*\* (like those behind ChatGPT) analyze the text, generate embeddings, and capture meaning, context, and relations between tokens.

1. **STEP 3: NUMERICAL EMBEDDINGS:**

* Each token is converted into numerical data (random numbers).
* These numbers represent embeddings, storing all learned information about the word or sentence.
* Only the transformer model understands what these embeddings mean based on its training.

##### FINE TUNING



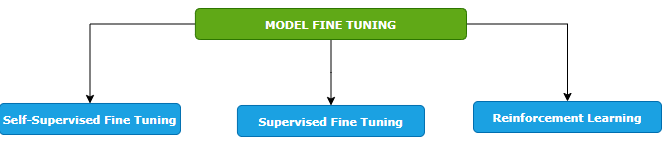
WHAT IS FINE TUNING

* Adjusting a pre-trained large language model (LLM) to provide better results on specific tasks or domain-specific datasets (e.g., healthcare, finance, etc.).

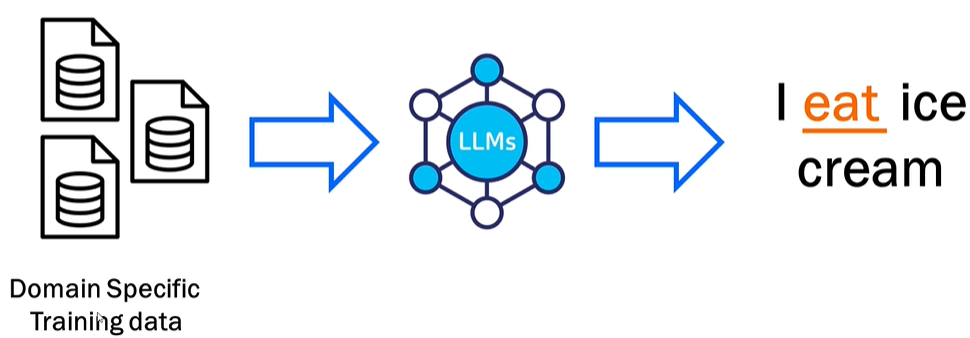
WHAT IS THE PURPOSE OF FINE TUNING

* Enhance the model's ability to generate focused and precise results on specialized datasets.
* Example: Fine-tuning an LLM on medical data improves its accuracy in answering medical queries compared to a general-purpose/pre-trained model.

###### DIFFERENT WAYS TO FINE TUNNING A MODEL



SELF-SUPERVISED FINE TUNING



* Training the model on a domain-specific dataset. Means – we give the foundation model a big pile of training data that is specific to our domain, and the model will learn from it. In this way, the model learns to predict missing pieces of data.
* Example - when we say I ice cream, the model predicts that the missing word is eat.
* This is like how the foundation model is trained but the key difference here is that we are fine tuning the model by providing the domain specific data set. Example - Like if we want to train it on health care data set, we will pass it - the drug structure, scientific studies, all the documents that are related to drug and the model will learn from it and it would be able to generate content based on that.

SUPERVISED FINE TUNING

A screenshot of a computer

AI-generated content may be incorrect.

* The model is trained using a labeled dataset where both inputs and outputs are provided.
* Example: Input: "How do I find a broken bone?" → Output: "X-ray."
* Helps the model learn more precise responses based on labeled data.

REINFORCEMENT LEARNING

|  |  |
| --- | --- |
| A blue arrow with black text  AI-generated content may be incorrect. | A blue arrow pointing to a black arrow  AI-generated content may be incorrect. |
| **LOW SCORE FOR BAD RESULT** | **HIGH SCORE FOR GOOD RESULTS** |

* **Feedback-based learning**.
* The model generates outputs, and scores are assigned based on quality (high score for good results, low score for bad results).
* The model learns over time from the feedback to improve predictions.

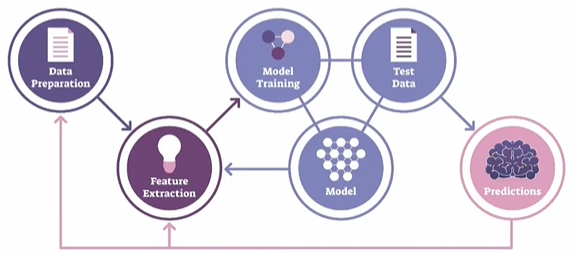
##### KEY FEATURES OF FINE TUNING

1. **STARTS FROM A PRE-TRAINED MODEL**: Fine tuning builds on top of a foundation model already trained on large datasets—it does not start from scratch.
2. **REQUIRES DOMAIN-SPECIFIC DATA**: We must provide good-quality, specific data for training tailored to your use case (e.g., drug data for healthcare).
3. **No Universal SOLUTION**: Each task/use case is unique, requiring case-specific implementation and variations.
4. **ITERATIVE PROCESS**: Fine tuning is repetitive and requires multiple cycles of iteration and adjustments for optimal results.

## MACHINE LEARNING MODEL

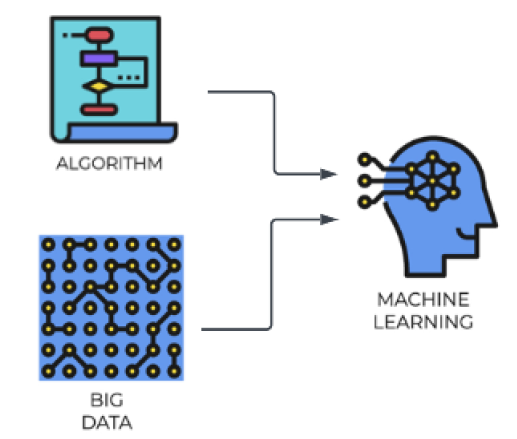
* This is an object that has been trained beforehand on a dataset. The model can then give output based on input values
* The machine learning object is nothing but a file or set of files. It would have the desired logic that would take in input and spur output data

## MACHINE LEARNING PROCESS



1. **DATA PREPARATION**:
   1. This step involves gathering and cleaning the data to make it suitable for analysis.
   2. It includes tasks like handling missing values, removing outliers, normalizing or standardizing the data, and splitting the data into training and testing sets.
2. **FEATURE EXTRACTION**:
   1. Feature extraction is the process of selecting or creating relevant features from the data that will be used as input for the machine learning model.
   2. It involves identifying the most informative attributes or transforming the data into a more meaningful representation that captures the underlying patterns and relationships.
3. **MODEL TRAINING**:
   1. Model training is the process of using the prepared dataset to teach the machine learning model to recognize patterns and make predictions.
   2. During this step, the model learns from the input data and adjusts its internal parameters to optimize its performance.
4. **MODEL**:
   1. The model is the representation of the learned patterns and relationships from the training data.
   2. It encapsulates the knowledge gained during the training process and can be used to make predictions or decisions on new, unseen data.
5. **TEST DATA**:
   1. Test data is a separate set of data that is not used during the model training. It is used to evaluate the performance of the trained model and measure its accuracy.
   2. The test data should be representative of the real-world scenarios the model will encounter.
6. **PREDICTION**:
   1. Once the model is trained and evaluated, it can be used to make predictions on new, unseen data.
   2. The model takes the input data, processes it using the learned patterns, and generates predictions or decisions based on the trained knowledge.
   3. The accuracy and reliability of the predictions depend on the quality of the model and the suitability of the input data.

## MACHINE LEARNING MODEL AND DATASET



### DATA SET

* A dataset is a collection of structured or unstructured data that is used for analysis, machine learning, or other data-related tasks. It typically consists of multiple data points or instances, where each data point represents an observation or sample.
* A dataset can include various types of data such as numerical, categorical, text, images, or time series.
* Datasets are crucial for training and evaluating machine learning models. They provide the necessary information for the model to learn patterns and relationships.
* Datasets can be sourced from various places, including public repositories, research institutions, or generated internally within an organization
* Common operations on datasets include cleaning, transforming, splitting into training and testing sets, and feature engineering.

#### EXAMPLE

A screenshot of a white grid

AI-generated content may be incorrect.

* Let's say we are building a ML model that can predict the house prices over time. For that we need to have prior information in place about the houses that were already sold over the previous years.
* The information can be the location of the house, the number of rooms, the agent involved in selling the house,

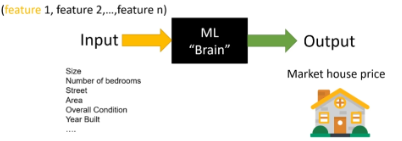
the area in square feet, and what was the price in USD. (as shown in above table)

* To train the model using historical data, we need to select the relevant features/column. ( for example, Agent column is irrelevant for price prediction)
* **This is the historical data is then used to training the ML model. Hence the data like - location, the number of rooms, and the area will become the features. Features are the measurable property within the dataset and the price is going be the label.**

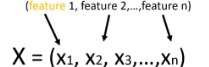
### FEATURES AND LABELS

#### FEATURES

* Features, also known as input variables or independent variables, are the measurable properties or characteristics of the data that are used to make predictions or classifications.
* They represent the input data that the model will analyze and learn patterns from. For example, in the context of predicting house prices, features could include the location, number of rooms, and area of a house.
* Each feature is represented by a specific value or attribute.



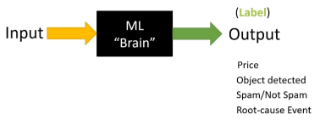
* Selecting the right features is a critical step to ensure the accuracy of ML models, something that is called feature selection.



***Mathematically the list of input features will be represented as a vector with the size of N like x1,x2, x3 until Xn***

#### LABELS

* Labels, also known as target variables or dependent variables, are the values that the model aims to predict or classify.
* Labels represent the desired output or outcome based on the input data. In the house price prediction example, the label would be the actual price of the house.
* The model analyzes the provided features and attempts to learn patterns that will help predict or estimate the label.



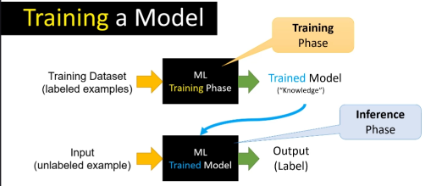
#### FEATURE AND LABELS

|  |
| --- |
| * During the training process, the model is presented with a dataset that contains both the features and the corresponding labels.      * The model learns the relationship between the features and labels by adjusting its internal parameters to minimize the difference between its predicted output and the actual labels. * Once trained, the model can then be used to make predictions or classifications on new, unseen data by analyzing the features and producing a predicted label. * In summary, features are the measurable properties of the data that serve as input for the model, while labels are the desired output or outcome that the model aims to predict or classify. The relationship between the features and labels is learned by the model during the training process. |

**IN A NUTSHELL, the summary is**

|  |
| --- |
| * Building a model to predict house prices over time * Prior information about sold houses including location, number of rooms, agent, area, and price in USD * Training the model using historical data, selecting relevant features and the price as the label * Cleaning the data, removing missing values, and performing feature engineering * Splitting the dataset into training and testing sets, using the training set to train the model and the testing set to evaluate its accuracy * Testing the accuracy and precision of the model by comparing its predictions with the actual values * Deploying the model and allowing users to make predictions based on new input data, such as location, number of rooms, and area. |

### TRAINING ML MODELS



1. As part of the prerequisite of most machine learning projects, we need a training data set. A training data set is a large group of labeled examples. A machine-learning system is going to learn patterns inside the training data set and stored that knowledge in something that is called a **model**.
2. This model is supposed to define as close as possible the relationship between features and the target label. In a common type of machine learning method called "supervised learning" the way to create this kind of model is based on analyzing a large group of labeled examples.
3. Once we have trained our model with those labeled examples, we can use that trained model to predict the label on unlabeled examples.

#### LIFECYCLE

The lifecycle of a model in a machine learning system, we have two main phases

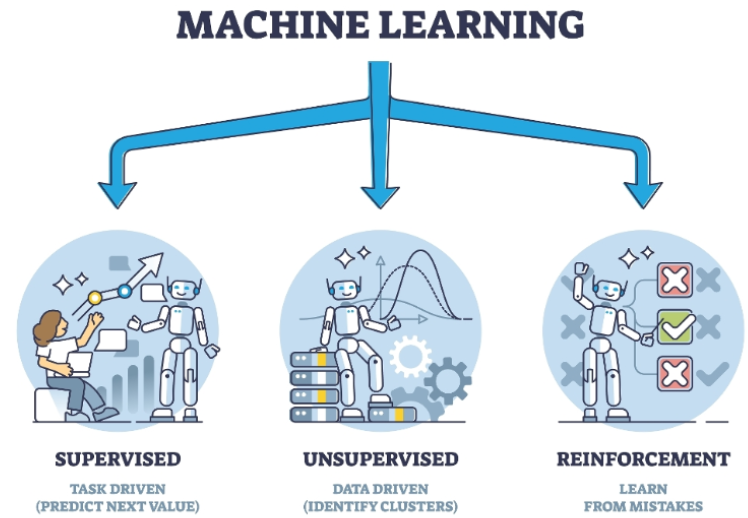
1. **TRAINING PHASE (LEARNING PHASE):** The main idea is to utilize or use some learning algorithms that will build the model using the training data set.
2. **INFERENCE PHASE (LEARNING PHASE):** In machine learning inference means applying the trained model in an actual machine learning system working in a production environment for making ongoing predictions.

### FEATURE ENGINEERING

### MACHINE LEARNING ALGORITHMS

|  |  |
| --- | --- |
|  | * When building a machine learning model, we need to train it on a dataset that consists of input features (also known as independent variables) and corresponding output labels (also known as dependent variables). The dataset serves as the training data for the model to learn patterns and relationships between the input features and output labels. * To enable the model to learn from the dataset, we need to choose and apply a suitable machine learning algorithm. These algorithms are designed to process the input features and labels and make predictions or classifications based on the patterns they discover in the data. * The chosen algorithm takes the dataset as input and applies its specific mathematical or statistical techniques to train the model. It learns from the provided dataset by adjusting its internal parameters to minimize the difference between the predicted output and the actual output labels. * Once the model has been trained using the chosen algorithm, it can be used to make predictions or classifications on new, unseen data. The trained model has learned the patterns in the dataset and can apply this knowledge to make predictions on similar data instances. |

### MACHINE LEARNING MODELS



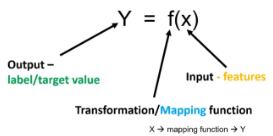
1. **SUPERVISED LEARNING MODELS**
   1. In supervised learning, for training data set that we feed in the features along with the labels(output)
2. **UNSUPERVISED LEARNING MODELS**:
   1. In unsupervised learning – we don’t know what the output will be, hence we cannot provide labels in training data
   2. Here we just want the model to split our data into different groups or cluster
3. **REINFORCEMENT LEARNING MODELS**:
   1. These models learn through interaction with an environment. They receive feedback in the form of rewards or penalties, allowing them to learn optimal strategies for decision-making.
   2. Example :
      1. Let's take the example where we trying to teach a system how to play a game of chess against a human. So now, instead of we feeding in data about the different chess moves, we ask the model to play against a human.
      2. Hence, the system would go ahead and make a set of moves, and let's say it loses. So now it understands that if it performs particular set of actions, it results in a loss,then the next time around it will go and change its actions.
      3. It'll keep on doing this, so it's performing reinforcement of its actions. At the end when it does win a game,that's the reward. So now it understands that based on certain set of actions that it takes, it's gonna get a reward.

#### SUPERVISED LEARNING

* The name supervised learning originates from the idea that training a machine while using this type of approach is like how humans are learning under the supervision of a teacher.
* Example - In regular school class, we have a group of students and a teacher. During a lecture about some specific topic, the teacher will provide several examples while teaching something. The students will use those examples

analyze and memorize them, something that will help them to extract the patterns from those examples. At a later stage, based on the information provided, the students will be able to solve similar problems. Overall, the teacher decided what kind of examples to present and how many, he or she basically supervised the learning process.

* **In supervised learning, we train machines by providing them with a set of examples, each provided example is a pair consisting of an input object and the desired output value for that object, it’s called label data set.**



* A machine learning black box with an input and output is basically a kind of data transformation that can be presented as a generic formula.
  + **"X" is the input into the machine which can be a group of values called "features".**
  + **"Y" is the output of that machine, the target value.**
  + **Functions with the input x are basically some mathematical transformation function or mapping function discovered by the algorithm doing the training process.**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **RELATION BETWEEN MAPPING FUNCTION AND ML ALGORITHM**  **WHAT IS A MAPPING FUNCTION?**   * Think of a mapping function as a recipe or rule that tells us how to turn input into output. * In machine learning, this function is what the algorithm is trying to learn from data.   **REAL-WORLD EXAMPLE: PREDICTING HOUSE PRICES:** Let’s say we want to predict the price of a house based on its size.   * Input (X): Size of the house (e.g., 1000 sq ft) * Output (Y): Price of the house (e.g., ₹50 lakhs)   The ML algorithm’s job is to learn a function like:  **Price = f(Size)**  This function could be:   * A straight line (if price increases steadily with size) * A curve (if price increases faster for bigger houses)   **HOW ML ALGORITHM USES THE MAPPING FUNCTION?**   1. **Step 1**: We give it data: Sizes and prices of many houses. 2. **Step 2**: It finds a pattern: Learns the best function (mapping) that connects size to price. 3. **Step 3**: We use it to predict: For a new house size, it uses the function to predict the price.   Example in Simple Terms (Training Data)   |  |  | | --- | --- | | Size (sq ft) | Price (₹ lakhs) | | 1000 | 50 | | 1500 | 75 | | 2000 | 100 |  * The ML algorithm might learn: **Price = 0.05 X Size** * So, for a 1200 sq ft house: **Price = 0.05 X 1200 = ₹60** * A mapping function is the rule that connects input to output. * An ML algorithm learns this rule from data. * Once learned, it can make predictions on new data.   **Here's a simple chart that shows how a machine learning algorithm learns a mapping function from house size to price:**     * *Blue dots: Real data points (house size vs. price).* * *Red line: The learned function (Price = 0.05 × Size), which the ML algorithm uses to make predictions***.**   **This is a basic example of how ML finds a pattern (a mapping function) in data and uses it to predict outcomes.** |

Two typical tasks performed by Supervised Learning

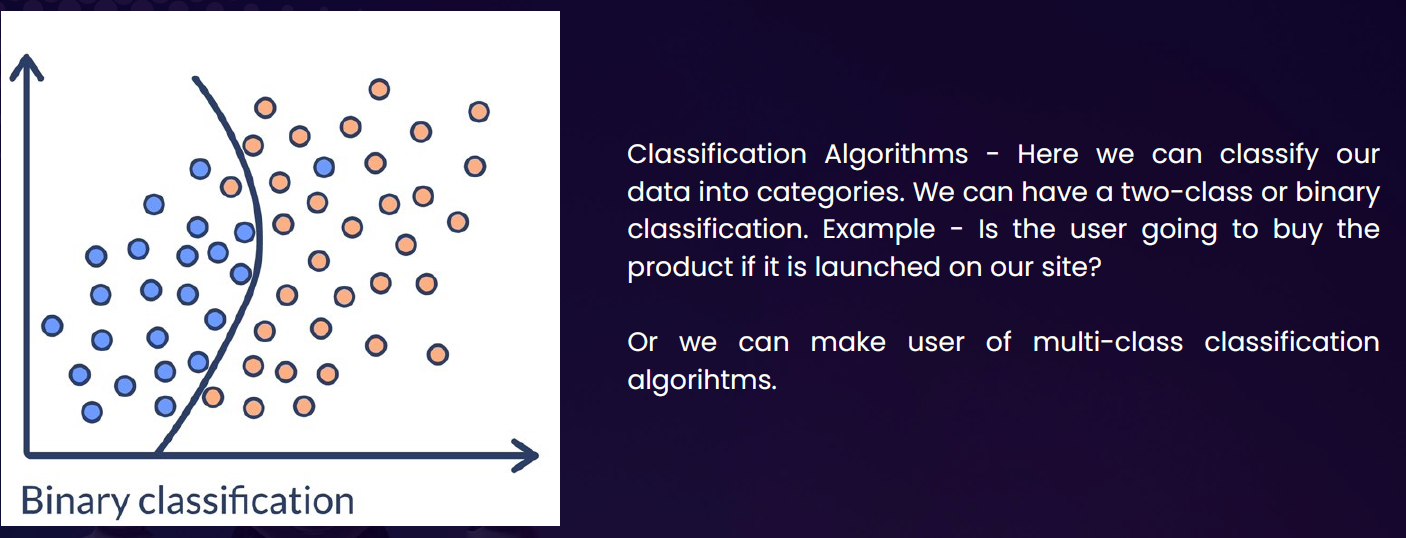
1. **CLASSIFICATION**
2. **REGRESSION**

##### CLASSIFICATION (BINARY CLASSIFICATION)

A black and green rectangular sign with white text

AI-generated content may be incorrect.

**Classification** is a type of supervised learning where the goal is to **predict a category or class**. Instead of predicting a number (like in regression), classification predicts a **label**.



**Real-Life Example: Email Spam Detection**

Imagine if we have a bunch of emails, and each one is labeled as either:

* **Spam**
* **Not Spam**

We train a machine learning model with this data. The model learns patterns like:

1. Emails with “win money” → likely spam
2. Emails from your contacts → likely not spam

Once trained, the model can classify **new emails** as spam or not spam.

**Other Examples of Classification**

|  |  |
| --- | --- |
| Problem | Classes (Labels) |
| Disease diagnosis | Sick / Healthy |
| Image recognition | Cat / Dog / Bird |
| Customer feedback sentiment | Positive / Negative / Neutral |
| Loan approval | Approved / Rejected |

###### MULTICLASS CLASSIFICATION



Multiclass classification is when a machine learning model predicts **one label out of three or more possible categories**.

|  |  |
| --- | --- |
| **Problem** | **Classes (Labels)** |
| Handwritten digit recognition | 0 to 9 |
| Animal image classification | Cat / Dog / Bird / Horse |
| Exam grading | A / B / C / D / F |

* SVM(Support Vector Machines) are one such classification ML Model

##### REGRESSION ALGORITHM

A graph with arrows and dots

AI-generated content may be incorrect.

* **Regression** is a type of **supervised learning** where the goal is to **predict a continuous value** (a number) based on input data.

**Real-World Example: Predicting House Prices**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Imagine a real estate agent and want to **predict the price of a house** based on certain features like:   * Size of the house (in square feet) * Number of bedrooms * Location * Age of the house | You collect data from past house sales. Each row in your dataset looks like this:   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Size (sqft)** | **Bedrooms** | **Age (years)** | **Location Score** | **Price (₹)** | | 1200 | 3 | 5 | 8 | 45,00,000 | | 1500 | 4 | 2 | 9 | 60,00,000 | | 1000 | 2 | 10 | 6 | 35,00,000 | |

* Now, we can train a **regression model** using this data. The model learns the relationship between the features and the price. Later, when we get a new house with known features but unknown price, the model can **predict the price**.

Great question! Let's break it down simply.

**What Does "Predict a Continuous Value" Mean?**

A **continuous value** is a number that can be **any value within a range**, not just fixed categories.

So, when we say **"predict a continuous value"**, we mean the model is trying to **guess a number** — not a label like "yes/no" or "cat/dog", but something like:

**Real-Life Examples**

|  |  |  |
| --- | --- | --- |
| **Problem** | **Input Features** | **Output (Continuous Value)** |
| Predict house price | Size, location, age | ₹50,00,000 |
| Predict temperature | Date, time, humidity | 32.5°C |
| Predict height of a child | Age, gender, parents' height | 145.2 cm |
| Predict fuel efficiency | Engine size, weight, speed | 18.7 km/l |

In each case, the **output is a number** that can vary smoothly — not just a fixed set of options.

**GRAPHICAL REPRESENTATION**

A graph of a house size

AI-generated content may be incorrect.

* Each dot represents a house.
* The x-axis is the size of the house (in square feet).
* The y-axis is the price of the house (in INR).
* The pattern shows that as house size increases, the price also tends to increase — this is the kind of relationship a regression model learns.
* Red line: The regression line — this is the model's prediction of house price based on size.
* The most common form of regression analysis is linear regression

###### LINEAR REGRESSION

* **Linear regression** is a fundamental concept in statistics and machine learning used to model the relationship between a **dependent variable** (target) and one or more **independent variables** (predictors or features).
* **Key Idea:** Linear regression assumes that there is a **linear relationship** between the input (X) and output (Y), meaning:

|  |  |
| --- | --- |
| **Y = aX + b + error** | * **Y** is the dependent variable (what we want to predict), * **X** is the independent variable (input), * **a** is the slope (how much Y changes with X), * **b** is the intercept (value of Y when X = 0), * **error** is the difference between the predicted and actual value. |

TYPES OF LINEAR REGRESSION

1. **Simple Linear Regression**: 1 independent variable: **Y = aX + b**
2. **Multiple Linear Regression**: more than 1 independent variable **Y = a1X1 + a2X2 + ... + anXn + b**

|  |
| --- |
| **Goal: Find the best values of a and b so that the predicted Y values are as close as possible to the actual Y values — usually by minimizing the mean squared error (MSE) between them.** |

**Other Applications:**

* Predicting sales based on ad spend
* Estimating house prices based on size, location, etc.
* Forecasting trends over time

EXAMPLE OF LINEAR REGRESSION

|  |  |  |
| --- | --- | --- |
| Problem: You have data showing how much a student studies (in hours), and their exam score. You want to predict the score based on study time. | **Study Hours (X)** | **Exam Score (Y)** |
| 1 | 50 |
| 2 | 55 |
| 3 | 65 |
| 4 | 70 |
| 5 | 75 |

**Goal:** Find a line Y = aX + b that best fits this data.

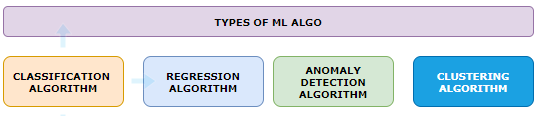
**Step 1: Plot the data :** If you plot X and Y on a graph, you'll notice a roughly straight-line trend.

**Step 2: Apply linear regression :** Using Python (with scikit-learn), you can fit a model:

|  |  |
| --- | --- |
| **from sklearn.linear\_model import LinearRegression**  **import numpy as np**  **# Data**  **X = np.array([[1], [2], [3], [4], [5]]) # 2D array for sklearn**  **Y = np.array([50, 55, 65, 70, 75])**  **# Model**  **model = LinearRegression()**  **model.fit(X, Y)**  **# Coefficients**  **slope = model.coef\_[0]**  **intercept = model.intercept\_**  **print(f"Model: Y = {slope:.2f}X + {intercept:.2f}")** | **Output:**  Model: Y = 6.5X + 43.0  This means:   * For every extra hour studied, score increases by ~6.5 * If a student studies **0 hours**, expected score = 43   **Prediction Example:**  Predict the score if a student studies for 6 hours:  **model.predict([[6]]) # Output: ~82.0**  So, **predicted score = 82**. |

#### UNSUPERVISED LEARNING

### MACHINE LEARNING ALGORITHMS CLASSIFICATION



#### ANOMALY DETECTION ALGORITHM

* This is used to detect if data deviates from the norm
* Example: used to detect fraudulent credit card purchase

#### CLUSTERING ALGORITHM

A computer screen with text and a diagram

AI-generated content may be incorrect.

# GENERATIVE AI

* Artificial Neural Network (ANN) is the foundation of Generative AI. It’s a type of machine learning model inspired by the structure and functioning of the human brain

A diagram of a diagram

AI-generated content may be incorrect.

An ANN is made up of layers of **nodes (neurons)**:

1. **Input Layer** – Takes in the raw data/features (e.g., pixels of an image, words in a sentence).
2. **Hidden Layers** – Perform computations and extract features. There can be one or many of these.
3. **Output Layer** – Produces the final result (e.g., classification, prediction, generated content).

Here’s a more **explanatory summary** of the lecture on **Artificial Neural Networks (ANNs)**:

**🧠 1. Input Layer – Where It All Begins**

* This is the **first layer** of the network.
* It receives **raw data** like numbers, images, or text.
* The data is broken into **features** (e.g., for predicting apartment price: size, number of rooms, location).
* Each feature becomes an **input node** in the network.

**🔍 2. Hidden Layers – The Brain of the Network**

* These are the **intermediate layers** between input and output.
* They **process the input data** and extract deeper patterns or subfeatures.
* More hidden layers = more ability to learn **complex relationships** in the data.
* This is why it's called **deep learning**—because of the **depth** (number of hidden layers).

**🎯 3. Output Layer – The Final Decision**

* This layer gives the **final result** of the network.
* Depending on the task, it could:
  + Classify (e.g., spam or not spam),
  + Predict (e.g., apartment price),
  + Generate (e.g., new text or images).

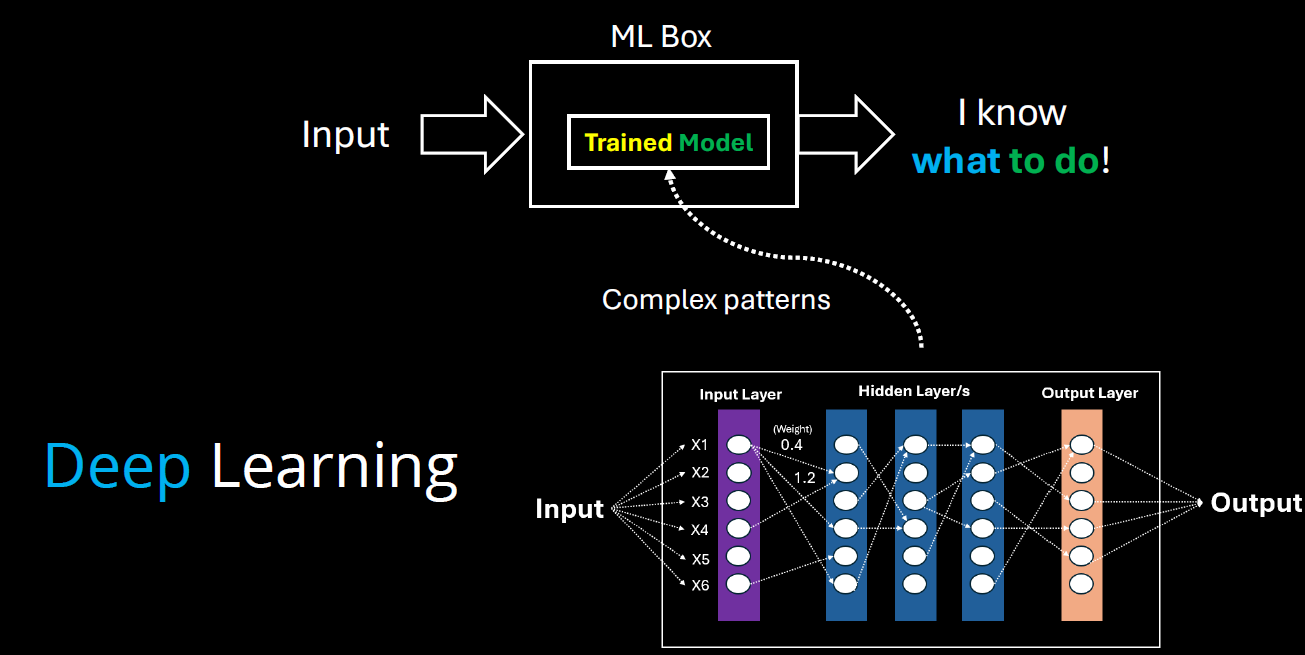
**🔗 4. Connections and Weights – How Learning Happens**

* Every node is connected to others in the next layer.
* Each connection has a **weight**—a number that shows how important that connection is.
* During training, the network **adjusts these weights** to improve accuracy.
* For example, if “apartment size” is very important, its connection will have a **higher weight**.

**🔄 5. Training the Network – Learning from Data**

* The network learns by comparing its output to the correct answer and adjusting weights.
* This process is called **backpropagation**.
* Over time, the network learns which features are most important and how to combine them.

## DEEP LEARNING ARCHITECTURE



* **Model Training**: In machine learning, training a model means teaching it to recognize patterns in data.
* **Model = Knowledge**: The trained model represents the system’s learned knowledge from the data.
* **Complex Patterns**: When data patterns are too complex for traditional methods, **deep learning** is used.
* **Deep Learning**: Uses **multiple layers** in a neural network to detect and learn complex patterns.
* **Layered Learning**: Each layer in the network captures increasingly abstract features from the data.
* **Evolution of ML**: Deep learning is a more advanced stage in the evolution of model training, enabling machines to handle tasks like image recognition, language understanding, and more.

## DIFFERENT ARCHITECTURE OF DEEP LEARNING

### RECURRENT NEURAL NETWORKS (RNNS)

* RNNs are a type of neural network designed to handle **sequential data** (like text, time series, or speech).
* They are good at remembering **previous inputs** using a concept called **"memory"**, which helps them understand context.

**HOW DO THEY WORK?**

* They process one element at a time in a sequence.
* Each step’s output depends on:
  1. The current input.
  2. The previous step’s output (memory).
* This loop-like structure gives RNNs their “recurrent” nature.

**WHERE ARE RNN USED?**

1. Text generation (e.g., writing like a human).
2. Language translation (e.g., English to Hindi).
3. Speech recognition (e.g., Siri or Google Assistant).
4. Stock price prediction (based on past trends).

### CONVOLUTIONAL NEURAL NETWORKS (CNNS)

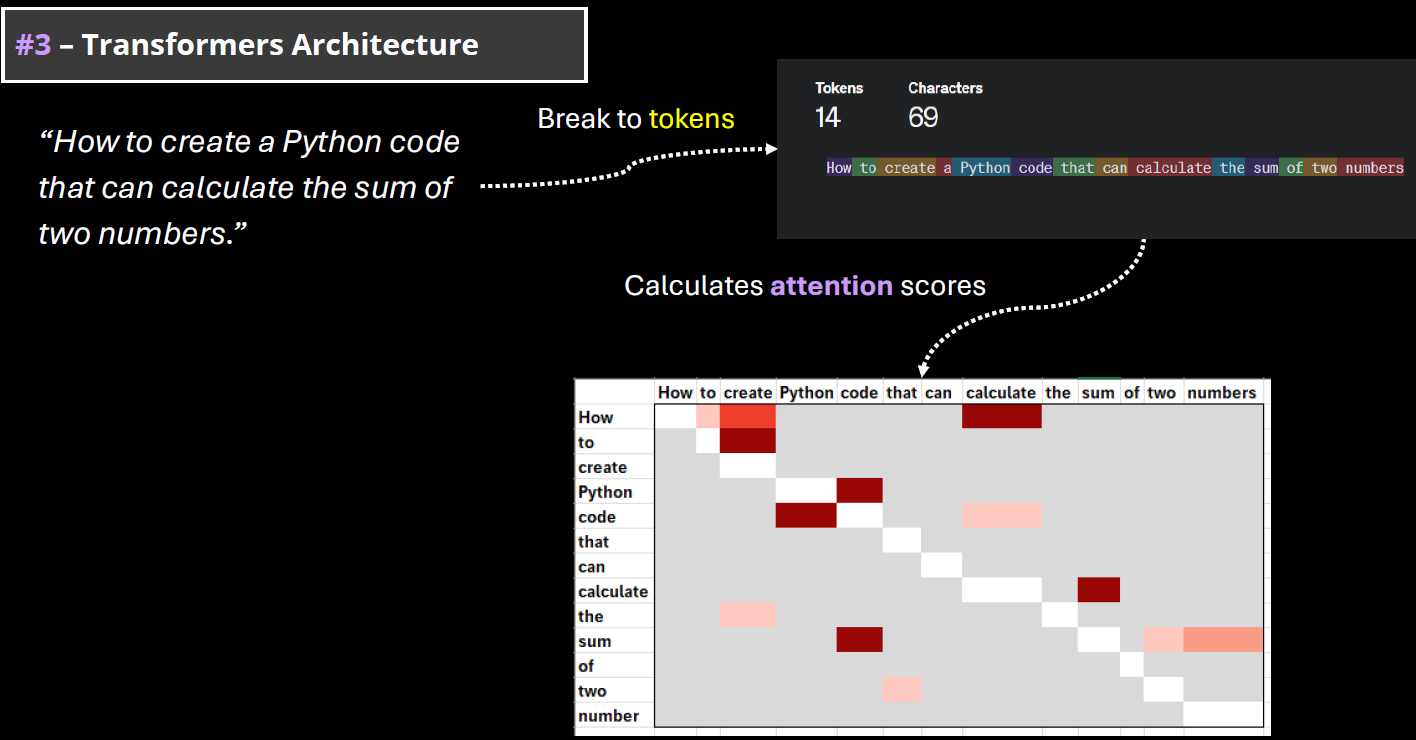
* Designed for image and video processing.
* Used in tasks like image classification, object detection, and segmentation.
* Not ideal for language tasks, but powerful in visual domains.

### TRANSFORMER ARCHITECTURE

* **Transformer Architecture is now the standard for generative AI.**
* Processes data in parallel, not sequentially like RNNs → much faster and scalable.
* Uses GPUs (Graphics Processing Units) for efficient training and inference.
* Cloud platforms (AWS, Google Cloud) use GPUs to train large models.
* It makes use of something **Attention Mechanism**

#### ATTENTION MECHANISM

* Like the way we use **colored markers** to highlight important words in a sentence, transformer architecture **focus on key parts** of the input.
* In this architecture – it breaks down the word in multiple tokens called **tokenization**
* Then it calculates the attention **score** of each pair of words to understand relationships between words (e.g., “calculate” is linked to “sum”) – as shown below!
* Hence - using the transformer architecture, the parallel processing coupled with the attention mechanism enables Gen-AI system to digest more data, process it faster, and catch more complex patterns.



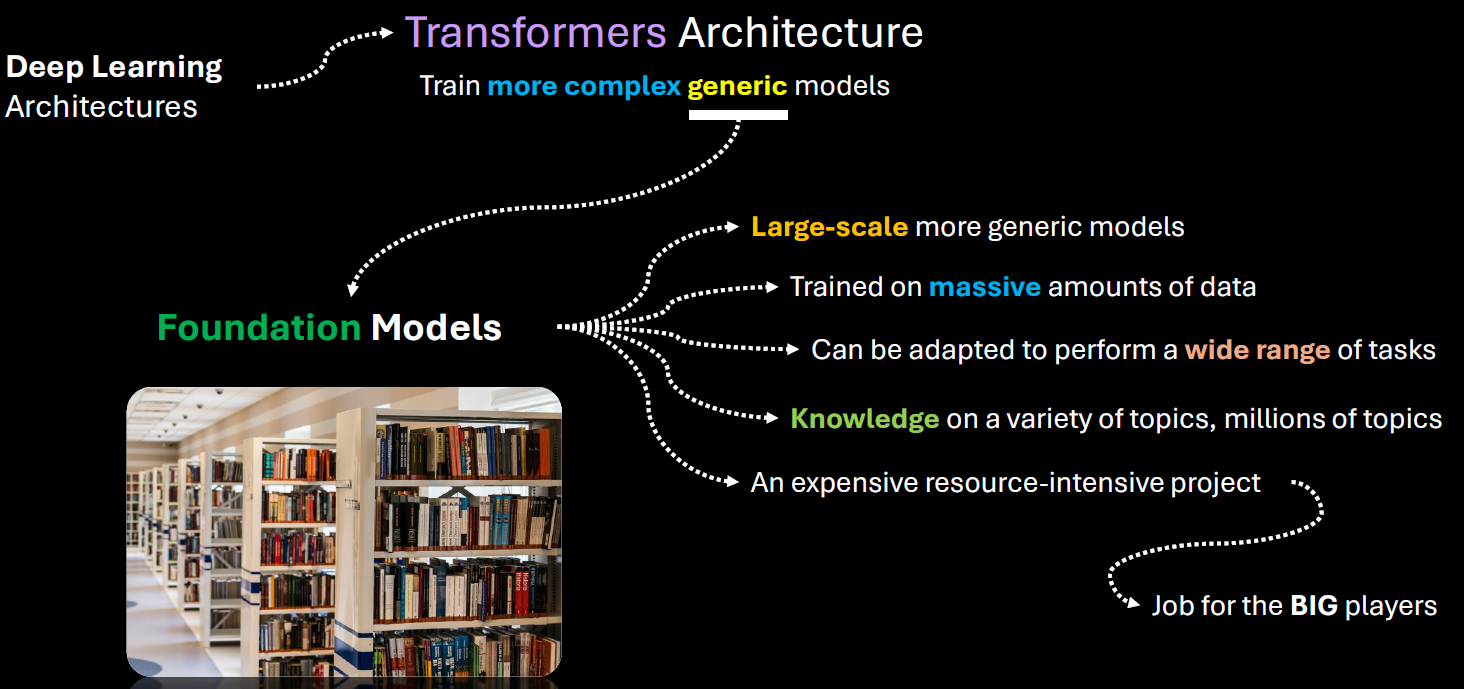
#### WHY TRANSFORMERS MATTER FOR GENERATIVE AI?

A diagram of a computer

AI-generated content may be incorrect.

* The power of generative AI is based on the capabilities to digest and understand the human language. In machine learning terminology, it is called natural language processing(NLP).
* **Transformer Architecture is helping to train very complex models that can handle the human language. They are called LLMs (large language models)**
* Note: Training an LLM model requires huge amount of data and computing resources.

## FOUNDATION MODEL



* **Foundation models** are large-scale machine learning models trained on vast amounts of data that can be adapted to a wide range of tasks.
* Training foundation models is expensive and resource-intensive which requires massive data collection, storage, and processing capabilities.
* Needs advanced hardware, software infrastructure, and skilled teams.
* Big tech companies like Google, Microsoft, and Amazon have the resources to train such models.
* **Popular example**: GPT, the foundation model behind OpenAI’s ChatGPT, trained on vast amounts of data

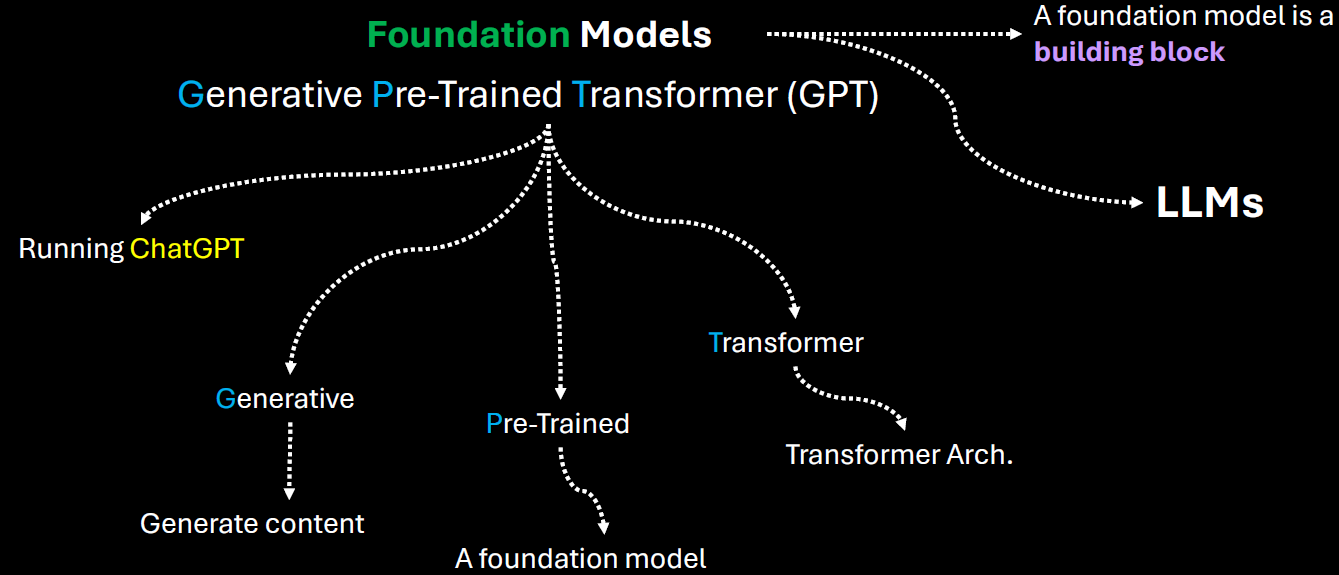
### KEY CHARACTERISTICS

1. **Scale**:  
   Foundation models are typically trained on massive datasets using billions (or even trillions) of parameters. Examples include GPT (like me), BERT, and CLIP.
2. **Generalization**:  
   They are not trained for a single task. Instead, they learn general patterns in data, which allows them to be fine-tuned or prompted for many different tasks—like translation, summarization, image recognition, or even coding.
3. **Multimodal Capabilities**:  
   Some foundation models can handle multiple types of data—like text, images, audio, or video—at the same time. For example, OpenAI’s CLIP can understand both images and text.
4. **Transfer Learning**:  
   Once trained, these models can be adapted to new tasks with relatively little additional data or training, making them highly efficient for downstream applications.

EXAMPLES OF FOUNDATION MODELS

* **GPT (Generative Pre-trained Transformer)** – for natural language understanding and generation.
* **BERT (Bidirectional Encoder Representations from Transformers)** – for understanding the context of words in text.
* **DALL·E** – for generating images from text prompts.
* **CLIP** – for connecting images and text.
* **Whisper** – for speech recognition.

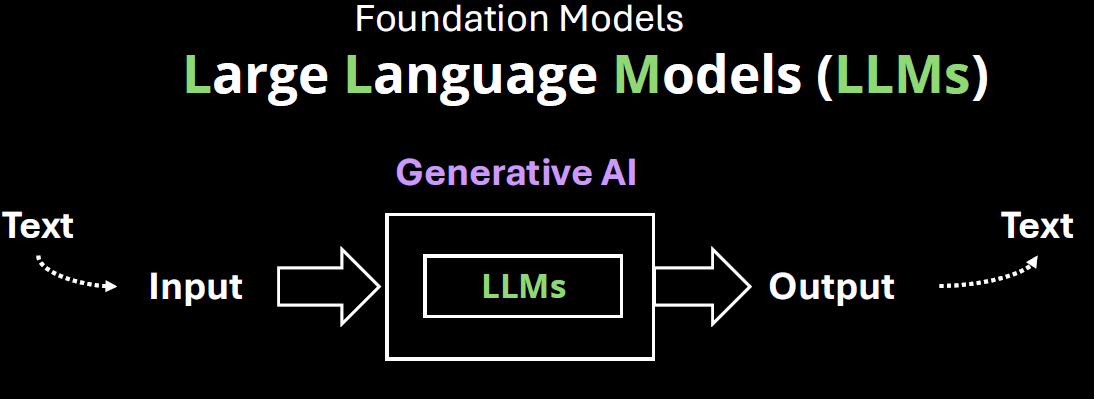
ABOUT GPT



|  |  |
| --- | --- |
| **Generative** | It means that the model can generate content based on the input |
| **Pre-trained** | This model was trained on a large amount of data from diverse sources such as websites, books and articles. |
| **Transformer** | That's the internal architecture of the model, which is becoming the de facto architecture for creating foundation models. |

* Users can directly interact with the model using simple text as a prompt, ask a question and get an answer.
* Given this kind of versatility of a foundation model, smaller players, like medium sized companies or startups can leverage those foundation models developed or and provided by the big players. Hence - Instead of investing millions of dollars in training such models from scratch, they can adapt an existing foundation model for a fraction of that amount and introduce new AI based products more quickly.
* There are many types of foundation models, some focused-on handling natural language processing. Some of them are focused on computer vision tasks like image and video generation, speech recognition etc.
* One of the core foundation model types is for natural language processing. They are called LLMs

## LARGE LANGUAGE MODELS(LLMs)

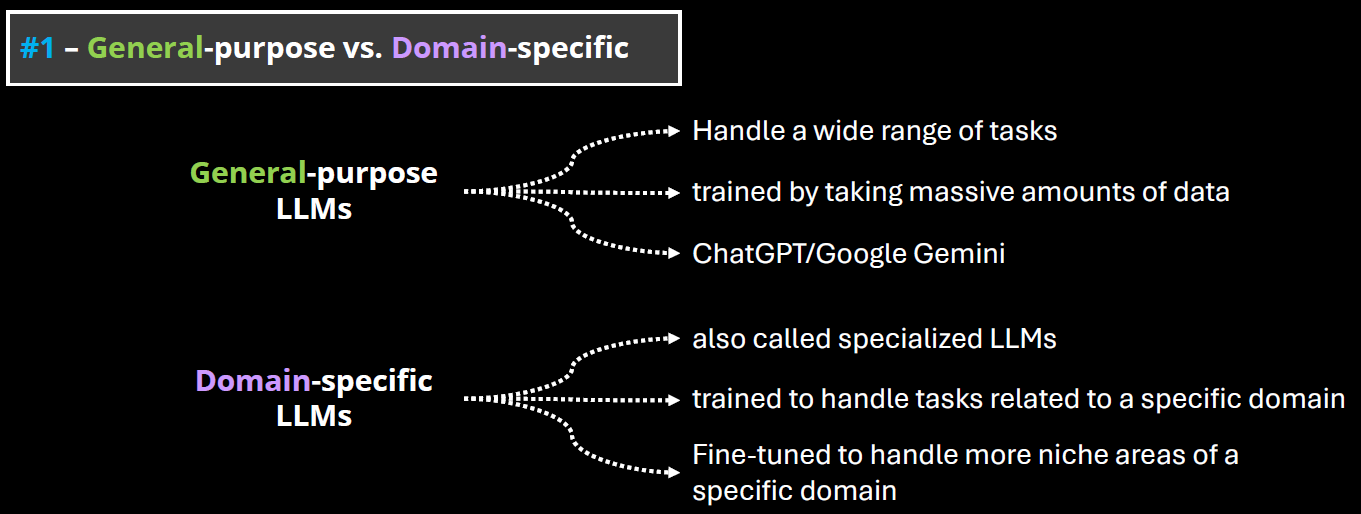


1. Core Capabilities of LLMs is to handle text as input &output
2. It can analyze, understand, and generate complex responses
3. **ChatGPT is based on LLM**

### MODEL TYPES (LLM TYPES)

#### GENERAL PURPOSE AND DOMAIN-SPECIFIC LLMS

The distinction between General Purpose LLMs and Domain-Specific LLMs lies in their training data, capabilities, and intended use cases.:



##### GENERAL PURPOSE LLMS

* These are large language models trained on a broad and diverse dataset that spans many domains (e.g., science, literature, law, medicine, pop culture, etc.).
* Examples: GPT-4, Claude, Google Gemini, LLaMA.
* Strengths:
  + Versatility: Can handle a wide range of tasks (e.g., summarization, translation, coding, creative writing).
  + Adaptability: Can generalize well across different topics and user needs.
  + Scalability: Useful in applications where domain-specific knowledge is not required.
* Limitations: May lack deep expertise in specialized fields.

##### DOMAIN-SPECIFIC LLMS

* These are LLMs trained or fine-tuned on specialized datasets from a particular field (e.g., legal, medical, financial, scientific).
* Examples: Med-PaLM (medical), FinGPT (finance), BioGPT (biomedical), Legal-BERT (legal texts)
* Strengths:
  + High accuracy in domain-specific tasks.
  + Better understanding of terminology, context, and nuances in the field.
  + Often used in regulated industries where precision is critical.
* Limitations:
  + Limited generalization outside their domain.
  + May require frequent updates to stay current with domain knowledge.
  + Less flexible for multi-domain tasks.

##### OPEN AND CLOSED SOURCE LLMs

**The distinction between Open-Source and Closed-Source LLMs revolves around accessibility, transparency, control, and community involvement.**

**A diagram of a source

AI-generated content may be incorrect.**

###### OPEN-SOURCE LLMS

* These models have their code, weights, and training data (or methodology) publicly available. Anyone can inspect, modify, or fine-tune them.
* Examples: Meta’s LLaMA (e.g., LLaMA 2, LLaMA 3), Mistral, Falcon,OpenChat, BLOOM (by BigScience)

CLOSED-SOURCE LLMS

* These models are proprietary. The model weights, training data, and architecture details are not publicly available.
* Examples: OpenAI’s GPT-4, Anthropic’s Claude, Google’s Gemini, , Cohere Command R+,Amazon Titan

### HOW LLMs WORKS?

A diagram of a computer

AI-generated content may be incorrect.

#### PROMPTS & TOKENS

A diagram of a cream line

AI-generated content may be incorrect.

**WHAT IS A PROMPT?**

* A **prompt** is the **input** we give to a language model — it's the question, instruction, or text we type to get a response. We Think of it like a **conversation starter** or a **command**.It can be a single word, a sentence, or a long paragraph.

**WHAT IS A TOKEN?**

* A **token** is a **chunk of text** — usually a word or part of a word — that the model processes. It is numerical representation (converted by Tokenizer)of word or parts of word , phrases or a character
* Tokens can be as short as one character or as long as one word.
* For example:
  + "ChatGPT" → 1 token
  + "unbelievable" → might be split into ["un", "believ", "able"] → 3 tokens
  + "I am happy." → 4 tokens (["I", " am", " happy", "."])
* Most models (like GPT-4) use a tokenizer to split text into tokens. The number of tokens affects:
  + **Cost** (for API usage)
  + **Speed**
  + **Context limit** (e.g., GPT-4-turbo can handle up to 128k tokens)

|  |
| --- |
| * The set of all tokens used by the model is called the vocabulary of the model * The process of splitting text into tokens is called tokenization. |

**TOKENS IN CHAT GPT**

|  |  |
| --- | --- |
|  | Open the URL: <https://platform.openai.com/tokenizer>  Enter the desired prompt  It will show how many has been created for a given prompt along with attention score (based on color code)  Note : Each model tokenize the prompt differently as they use different tokenizers |

##### TOTAL TOKENS

* Tokens are numerical representation of characters, words or phrases

|  |
| --- |
| * Tokens are a fundamental metric for measuring usage in an AI system. * **Total tokens = Input tokens (**The number of tokens in the prompt or message you send**) + Output tokens(**The number of tokens in the model's response**)**   . **WHY IT MATTERS?**   * Language models have a **token limit** per interaction (e.g., 8,000 or 32,000 tokens depending on the model). * If the total number of tokens exceeds the limit, the model may truncate the input or fail to generate a complete response. |

##### CONTEXT WINDOW

A row of blue circles

AI-generated content may be incorrect.

* The context window refers to the maximum number of tokens (input + output) that the model can "see" or process at one time.
* The context window includes:
  + **Your input (prompt, messages, instructions)**
  + **The model’s output (response)**
  + **Any previous conversation history (if it's part of the current session)**
* Different models have different context window sizes. For example:
  + GPT-3.5: ~4,096 tokens
  + GPT-4 (standard): ~8,192 tokens
  + GPT-4 Turbo: up to 128,000 tokens

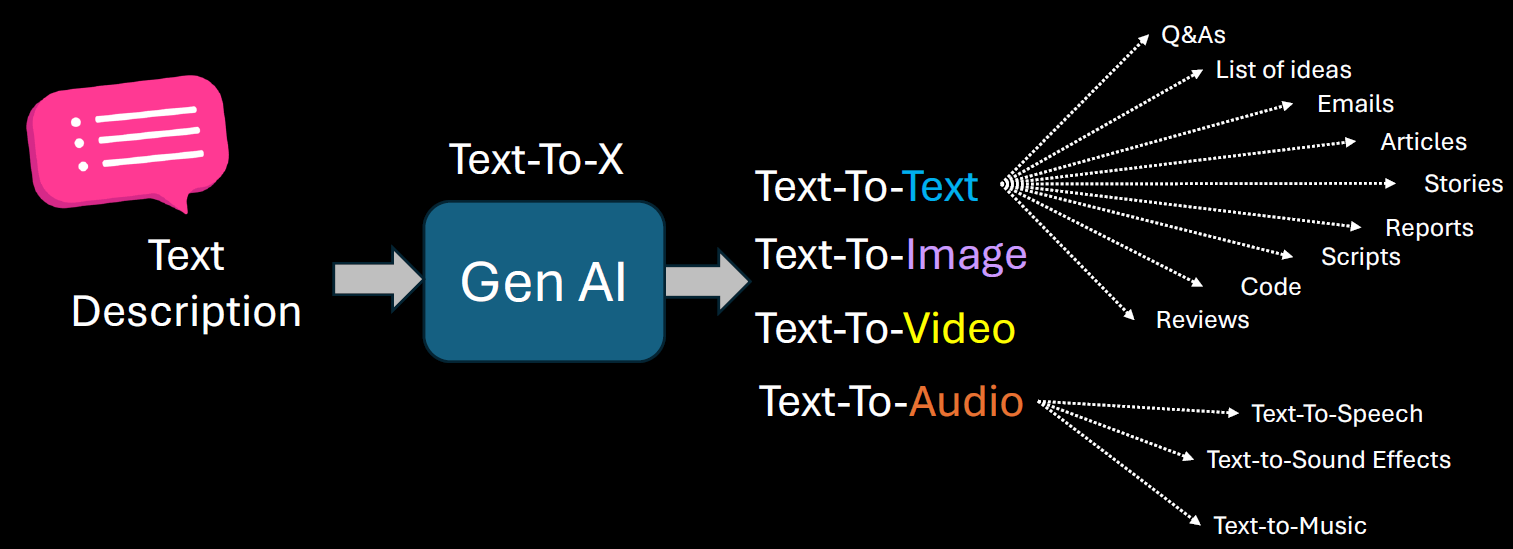
**WHY DO IT MATTERS?**

A black background with white lines and yellow text

AI-generated content may be incorrect.

* If the conversation exceeds the context window, older parts of the conversation may be truncated or forgotten.
* This affects the model’s ability to maintain long-term coherence or remember earlier details.
* For example, a LLM with context window of 10K, which is fed by an article of 15K token – it will truncate the token after 10K tokens of the Article. Hence the long documents may need to be chunked to fit within the context window of the LLM

## APPLICATION OF GEN-AI



### GEN-AI USE CASES

* 1. Brainstorm Assistant
  2. Summarization
  3. Code Generation
  4. Text Enhancement(e.g. to check for grammatical mistake like copilot in outlook)
  5. Image Generation

## GEN-AI LIMITATIONS & CHALLENGES

**PROMPT SENSITIVITY**

* Generative AI models are highly sensitive to how a prompt is phrased. Small changes in wording can lead to significantly different outputs.
* This can make it difficult to get consistent or desired results without careful prompt engineering.
* **Example:** Asking "Explain climate change simply" vs. "What causes climate change?" may yield different levels of detail or focus.

**KNOWLEDGE CUTOFF**

* Most generative AI models are trained on data available up to a certain point in time. They do not have real-time awareness or access to events or developments after their training cutoff.
* **Implication:** They cannot provide accurate information about recent events, new technologies, or updated regulations unless connected to live data sources.

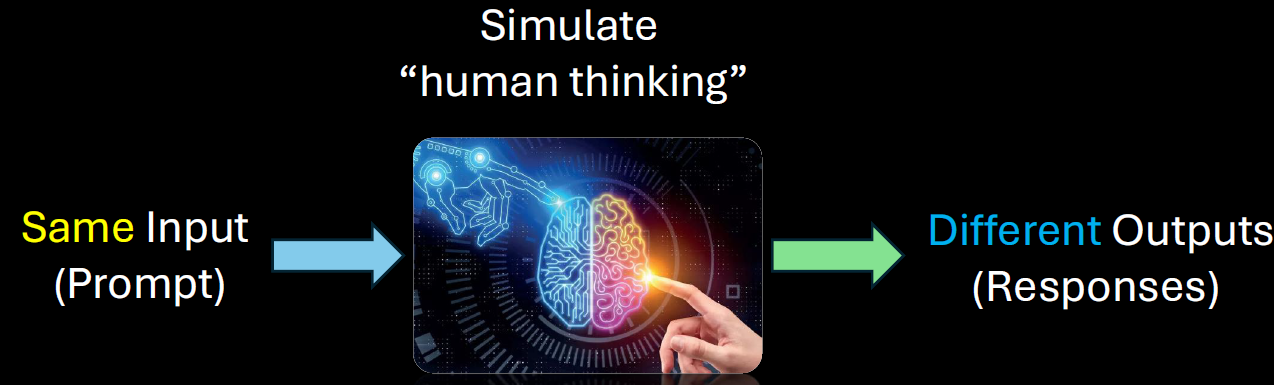
A diagram of a software program

AI-generated content may be incorrect.

The solution to this to

1. Retrain the model after regular interval of time
2. Connect the model with online tools

**IT IS NOT DETERMINISTIC**

****

Generative AI is **probabilistic**, not deterministic. This means the same prompt can produce different outputs each time it's run.

**Implication:** This variability can be useful for creativity but problematic for tasks requiring consistency and repeatability.

**STRUCTURED DATA**

* Generative AI struggles with tasks that require precise manipulation of structured data (like databases, spreadsheets, or complex logic).
* **Example:** It may misinterpret or incorrectly summarize tabular data or fail to follow strict formatting rules.

**HALLUCINATIONS**

* AI models can "hallucinate" — confidently generating false or misleading information that sounds plausible.
* **Example:** Citing non-existent research papers or inventing facts in a historical summary.

**LACK OF COMMON SENSE**

* Despite being trained on vast data, generative AI lacks **true understanding** or **common-sense reasoning**. It may fail at tasks that require intuitive knowledge or real-world logic.
* **Example:** It might suggest putting metal in a microwave or confusing cause and effect in a scenario.

**BIAS AND FAIRNESS**

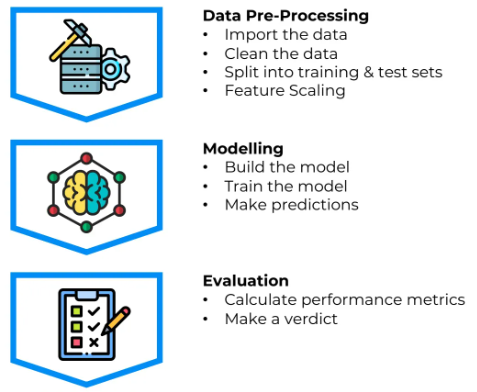
* AI models can reflect and even amplify biases present in their training data. This can lead to unfair, offensive, or discriminatory outputs.
* **Example:** Gender or racial bias in job recommendations or stereotypical characterizations in generated content.

**DATA PRIVACY, SECURITY, AND MISUSE**

* Generative AI can inadvertently expose sensitive information if trained on private data. It can also be misused for harmful purposes like generating fake news, phishing emails, or deepfakes.
* **Concerns:**
  + **Privacy:** Leaking personal or proprietary data.
  + **Security:** Being used to craft convincing scams.
  + **Misuse:** Generating harmful or misleading content.

# MACHINE LEARNING

## MACHINE LEARNING PROCESS

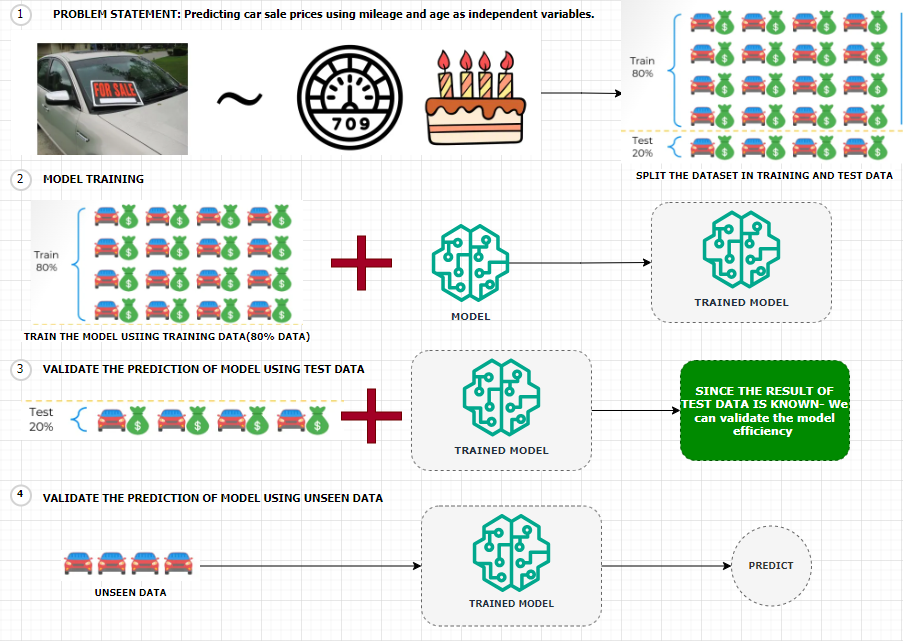


## TRAINING SET AND TEST SET

A diagram of a car with a price tag

AI-generated content may be incorrect.

Importance Of Splitting Data Set into a training set and a test set.



Predicting Car Sale Prices Using Linear Regression

Let’s consider a scenario where we aim to **predict the sale prices of cars** using a **linear regression model**. In this case:

* **Dependent Variable**: Sale price of the car
* **Independent Variables**:
  + Mileage of the car
  + Age of the car

**Dataset Overview**

We are provided with a dataset containing information on **20 cars**. To build and evaluate our model effectively, we split the dataset into two parts:

* **Training Set (80%)**:
  + Contains data for **16 cars**
  + Used to train the linear regression model
* **Test Set (20%)**:
  + Contains data for **4 cars**
  + Set aside before training to evaluate the model’s performance

Model Training and Evaluation

1. **Model Building**:  
   We train the linear regression model using only the training set. This means the model learns the relationship between mileage, age, and sale price from these 16 cars.
2. **Prediction on Test Set**:  
   After training, we apply the model to the 4 cars in the test set. These cars were **not part of the training process**, so the model has **no prior knowledge** of them.
3. **Comparison with Actual Prices**:  
   Since we already know the **actual sale prices** of the test set cars, we can now compare:
   * **Predicted Prices** (from the model)
   * **Actual Prices** (from the dataset)
4. **Model Evaluation**:  
   This comparison allows us to assess how well our model performs on unseen data. Metrics such as **Mean Absolute Error (MAE)** or **Root Mean Squared Error (RMSE)** can be used to quantify the model’s accuracy.

## FEATURE SCALING

Feature scaling **transforms the values of features to be on a similar scale**, typically to improve model performance and training stability.

We can understand feature scaling using the example below. Note: ***Feature scaling is always applied at column level***

A table with numbers and lines

AI-generated content may be incorrect.

There are 2 main techniques of feature scaling

1. **NORMALIZATION**
2. **STANDARDIZATION**

### NORMALIZATION

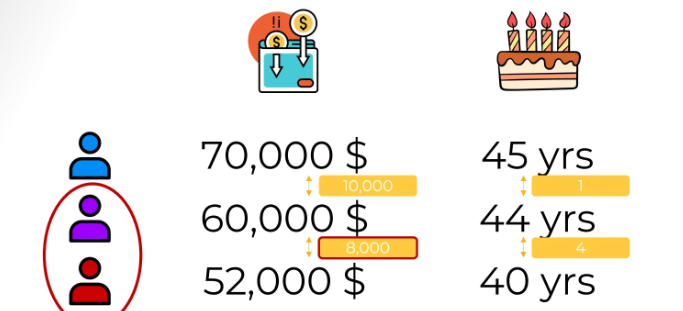
|  |  |
| --- | --- |
|  | * The normalization value lies between the closed interval of [0;1] |

|  |  |  |
| --- | --- | --- |
| **X1 (Price)** | **X-XMIN** | **Normalized Value(X1)** |
| $179.43 | $0.00 | 0.00 |
| $641.87 | $179.43 | 0.39 |
| $556.30 | $376.87 | 0.814959779 |
| $578.47 | $116.03 | 0.250908226 |
| $591.12 | $411.69 | 0.890256033 |
|  |  |  |
| X1-MAX |  | $641.87 |
| X1-MIN |  | $179.43 |
| X1MAX - X1MIN |  | **$462.44** |

### STANDARDIZATION

|  |  |  |
| --- | --- | --- |
| A math equation with numbers and symbols  AI-generated content may be incorrect. | µ | Average |
|  | Standard Deviation |
| * The value is lies in closes interval of [-3,3] * If data has some outliers – it will exist outside this range | |

### EXAMPLE - NORMALIZATION



1. Let's imagine we have a data set where we have two columns, annual income of a person and their age of

a blue, purple and red person.

1. **We must identify whether the purple person is like a “red” person or “blue” person . This is the task of clustering data. For that we need to do normalization of data as the units of the data is not uniform**

A close-up of a number

AI-generated content may be incorrect.

1. After normalizing, our values will look like above. Hence with the normalized data – From salary column perspective. The purple person is almost right in the middle between the red and the blue people(0.44), whereas in the age column, the purple person is closest to the blue person.

|  |
| --- |
| Scikit-learn (also written as scikit-learn or sklearn) is a powerful and widely used open-source machine learning library for the Python. It provides simple and efficient tools for:  🔍 Key Features   * Classification: Identifying which category an object belongs to (e.g., spam detection). * Regression: Predicting a continuous-valued attribute (e.g., house prices). * Clustering: Grouping similar data points (e.g., customer segmentation). * Dimensionality Reduction: Reducing the number of features (e.g., PCA). * Model Selection: Comparing, validating, and choosing parameters and models. * Preprocessing: Feature extraction, normalization, and transformation.   🧰 Built On  Scikit-learn is built on top of:   * NumPy: For numerical operations. * SciPy: For scientific computing. * Matplotlib: For plotting (indirectly used). * joblib: For model persistence and parallel processing. |

# DATA PRE-PROCESSING USING PYTHON

|  |  |
| --- | --- |
|  | * In this example we will perform data preprocessing step on this following data * It’s a user profile data – of an ecommerce website with a flag which says whether user has made purchase or not! * As a data processing step – we create two entities –  1. The first is **the matrix of features**, which contains separately these three columns (country, age, salary.) 2. And second is the dependent variable vector, which is last column(“Purchased”), because that's the column we want to predict.   *Note : This is exactly what we must do in this first data pre-processing phase.* |

|  |  |
| --- | --- |
| Step 1: Importing the libraries | import numpy as np import matplotlib.pyplot as plt import pandas as pd |
| Step 2: Importing the dataset | dataset = pd.read\_csv('Data.csv') X = dataset.iloc[:, :-1].values y = dataset.iloc[:, -1].values  OUTPUT  [['France' 44.0 72000.0]  ['Spain' 27.0 48000.0]  ['Germany' 30.0 54000.0]  ['Spain' 38.0 61000.0]  ['Germany' 40.0 nan]  ['France' 35.0 58000.0]  ['Spain' nan 52000.0]  ['France' 48.0 79000.0]  ['Germany' 50.0 83000.0]  ['France' 37.0 67000.0]]  ['No' 'Yes' 'No' 'No' 'Yes' 'Yes' 'No' 'Yes' 'No' 'Yes'] |
| Step 3: Missing Data   * For missing data, we can make use of Python library Scikit-learn. It is an open-source machine learning library built on top of **NumPy**, **SciPy**, and **matplotlib**. * It provides simple and efficient tools for data mining and data analysis. * For example - For missing salary - We will replace the missing salary with average salary in the column   **from sklearn.impute import SimpleImputer**  **imputer= SimpleImputer(missing\_values=np.nan, strategy='mean') imputer.fit(X[:, 1:3]) # Assuming columns 1 and 2 have missing values X[:, 1:3] = imputer.transform(X[:, 1:3]) # Transform the data to fill missing values**   * SimpleImputer from scikit-learn to fill missing values (NaN) in specific columns of the feature matrix X with the mean of each column. * It fits the imputer on columns 1 and 2 (indexing starts at 0), replaces missing values with the computed mean | |

Step 4: Encoding the categorial data

* Encoding categorical data is **crucial in machine learning** because most ML algorithms require **numerical input** to perform mathematical computations.

Common Encoding Techniques

|  |  |  |
| --- | --- | --- |
| Technique | Description | Best For |
| Label Encoding | Assigns A Unique Number To Each Category | Ordinal Data (E.G., "Low", "High") |
| One-Hot Encoding | Creates Binary Columns For Each Category | Nominal Data (E.G., "Red", "Blue") |
| Ordinal Encoding | Encodes Categories With Meaningful Order | Ordered Categories |
| Target Encoding | Replaces Categories With The Mean Of The Target Variable For Each Category | High-Cardinality Categorical Data |

Example

Suppose we have a column Fuel Type with values: ["Petrol", "Diesel", "Electric"]

* Label Encoding: Petrol → 0, Diesel → 1, Electric → 2 (May imply an order that doesn’t exist)
* One-Hot Encoding:

| Petrol | Diesel | Electric |
| --- | --- | --- |
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 1 |

Why Encode Categorical Data?

* ML Models Work with Numbers
  1. Algorithms like linear regression, decision trees, and neural networks **cannot interpret text** or labels directly. They need **numerical representations** to process the data.
* Preserves Information
  1. Encoding transforms categories into numbers **without losing the meaning** of the data. For example, converting "Red", "Blue", "Green" into numerical form allows the model to still distinguish between them.
* Improves Model Performance
  1. Proper encoding helps the model **understand relationships** between variables, which can lead to **better predictions** and **faster training**.
* Avoids Bias from Arbitrary Numbers
  1. Some encoding methods (like **One-Hot Encoding**) prevent the model from assuming an **ordinal relationship** where none exists.
  2. For example, assigning "Low", "Medium", "High" as 1, 2, 3 implies a ranking, which may or may not be appropriate.

CODE

Taking the example further will apply “Hot Encoding” of “County” column and “label encoding” to the “Purchased” column.

One Hot Encoding OF Country Column

|  |
| --- |
| from sklearn.preprocessing import OneHotEncoder  ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(),[0])],remainder='passthrough') X = np.array(ct.fit\_transform(X)) # Apply one-hot encoding to the first column |

LABELLED Encoding OF Country Column

|  |
| --- |
| from sklearn.preprocessing import LabelEncoder  le = LabelEncoder() y = le.fit\_transform(y) # Apply label encoding to the dependent variable |

EXAMPLE – CODING EXERCISE

Dataset

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| PassengerId | Survived | Pclass | Name | Sex | Age | SibSp | Parch | Ticket | Fare | Cabin | Embarked |
| 2 | 1 | 1 | Cumings, Mrs. John Bradley (Florence Briggs Thayer) | female | 38 | 1 | 0 | PC 17599 | 71.2833 | C85 | C |
| 4 | 1 | 1 | Futrelle, Mrs. Jacques Heath (Lily May Peel) | female | 35 | 1 | 0 | 113803 | 53.1 | C123 | S |
| 7 | 0 | 1 | McCarthy, Mr. Timothy J | male | 54 | 0 | 0 | 17463 | 51.8625 | E46 | S |
| 11 | 1 | 3 | Sandstrom, Miss. Marguerite Rut | female | 4 | 1 | 1 | PP 9549 | 16.7 | G6 | S |
| 12 | 1 | 1 | Bonnell, Miss. Elizabeth | female | 58 | 0 | 0 | 113783 | 26.55 | C103 | S |
| 22 | 1 | 2 | Beesley, Mr. Lawrence | male | 34 | 0 | 0 | 248698 | 13 | D56 | S |

Coding Exercise 3: Encoding Categorical Data for Machine Learning

**1**: Import required libraries - Pandas, Numpy, and required classes for this task - ColumnTransformer, OneHotEncoder, LabelEncoder.

**2**: Start by loading the Titanic dataset into a pandas data frame. This can be done using the pd.read\_csv function. The dataset's name is 'titanic.csv'.

**3**: Identify the categorical features in your dataset that need to be encoded. You can store these feature names in a list for easy access later.

**4**: To apply OneHotEncoding to these categorical features, create an instance of the ColumnTransformer class. Make sure to pass the OneHotEncoder() as an argument along with the list of categorical features.

**5**: Use the fit\_transform method on the instance of ColumnTransformer to apply the OneHotEncoding.

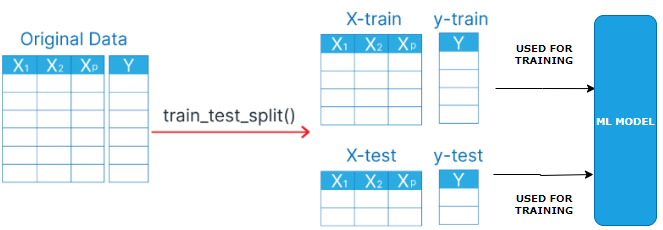
**6**: The output of the fit\_transform method should be converted into a NumPy array for further use.

**7**: The 'Survived' column in your dataset is the dependent variable. This is a binary categorical variable that should be encoded using LabelEncoder.

**8.**  Print the updated matrix of features and the dependent variable vector

|  |
| --- |
| # Importing the necessary libraries import pandas as pd import numpy as np from sklearn.compose import ColumnTransformer from sklearn.preprocessing import OneHotEncoder, LabelEncoder  # Load the dataset dataset = pd.read\_csv("titanic.csv")  # Identify the categorical data categorical\_features = ['Sex', 'Embarked', 'Pclass']  # Implement an instance of the ColumnTransformer class  ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(),categorical\_features)],remainder='passthrough')  # Apply the fit\_transform method on the instance of ColumnTransformer ct\_fit = ct.fit\_transform(dataset) # Apply one-hot encoding to the first column  # Convert the output into a NumPy array X = np.array(ct.fit\_transform(dataset))  # Use LabelEncoder to encode binary categorical data le = LabelEncoder()  Y = le.fit\_transform(dataset["Survived"]) # Apply label encoding to the dependent variable  # Print the updated matrix of features and the dependent variable vector print(X) print(Y) |

Step 5: Training Versus Test Data



|  |
| --- |
| from sklearn.model\_selection import train\_test\_split  x\_train, x\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=1) |

The train\_test\_split method arrays or matrices into random train and test subsets.

Parameters:

* X: Features (input data).
* y: Labels (target data).
* test\_size: Fraction or number of samples to use as the test set (e.g., 0.2 means 20% for testing).
* train\_size: Fraction or number of samples for training (optional, usually inferred).
* random\_state: Seed for random number generator (ensures reproducibility).
* shuffle: Whether to shuffle data before splitting (default is True).
* stratify: Ensures the split preserves the proportion of labels (optional, usually for classification).
* Returns: Splits the data into x\_train, x\_test, y\_train, and y\_test.

EXAMPLE

1. Import necessary Python libraries: pandas, train\_test\_split from sklearn.model\_selection, and StandardScaler from sklearn.preprocessing.
2. Load the Iris dataset using Pandas read.csv. Dataset name is iris.csv.
3. Use train\_test\_split to split the dataset into an 80-20 training-test set.
4. Apply random\_state with 42 value in train\_test\_split function for reproducible results.
5. Print X\_train, X\_test, Y\_train, and Y\_test to understand the dataset split.
6. Use StandardScaler to apply feature scaling on the training and test sets.
7. Print scaled training and test sets to verify feature scaling.

Step 6: Feature Scaling

# DATA PROCESSING USING PYTHON

# REGRESSION

* A **regression model** is a type of statistical or machine learning model used to understand the relationship between a **dependent variable** (what you're trying to predict) and one or more **independent variables** (the inputs or predictors).
* **In simple terms:** A regression model helps answer questions like:
  + "How does the price of a house depend on its size, location, and number of bedrooms?"
  + "How does advertising spending affect sales?"

Types Of Regression Models

1. Linear Regression

* Assumes a straight-line relationship between variables.
* Example: y = a + bx
* Multiple Linear Regression:
* Like linear regression, but with multiple input variables.
* Example: y = a + b*1x*1 + b*2x*2 + … + b*nx*n
* Polynomial Regression:
* Models curved relationships by including powers of the input variables.
* Example: y = a + bx + cx2
* Logistic Regression:
* Used when the output is categorical (e.g., yes/no, 0/1).
* Despite the name, it's used for classification, not regression.
* Ridge, Lasso, And Elastic Net Regression:
* Variants of linear regression that include regularization to prevent overfitting.

What It’s Used For

* Predicting future values (e.g., stock prices, sales).
* Understanding relationships between variables.
* Making data-driven decisions in business, science, and engineering.

## SIMPLE LINEAR REGRESSION

A diagram of equations

AI-generated content may be incorrect.

## MULTIPLE LINEAR REGRESSION

## POLYNOMIAL REGRESSION

## SUPPORT VECTOR REGRESSION

## DECISION TREE REGRESSION

## RANDOM FOREST REGRESSION