GenAI for Beginners
Learning with Demos

Kshitij Joy
(Oracle Certified Master)


Artificial
Intelligence

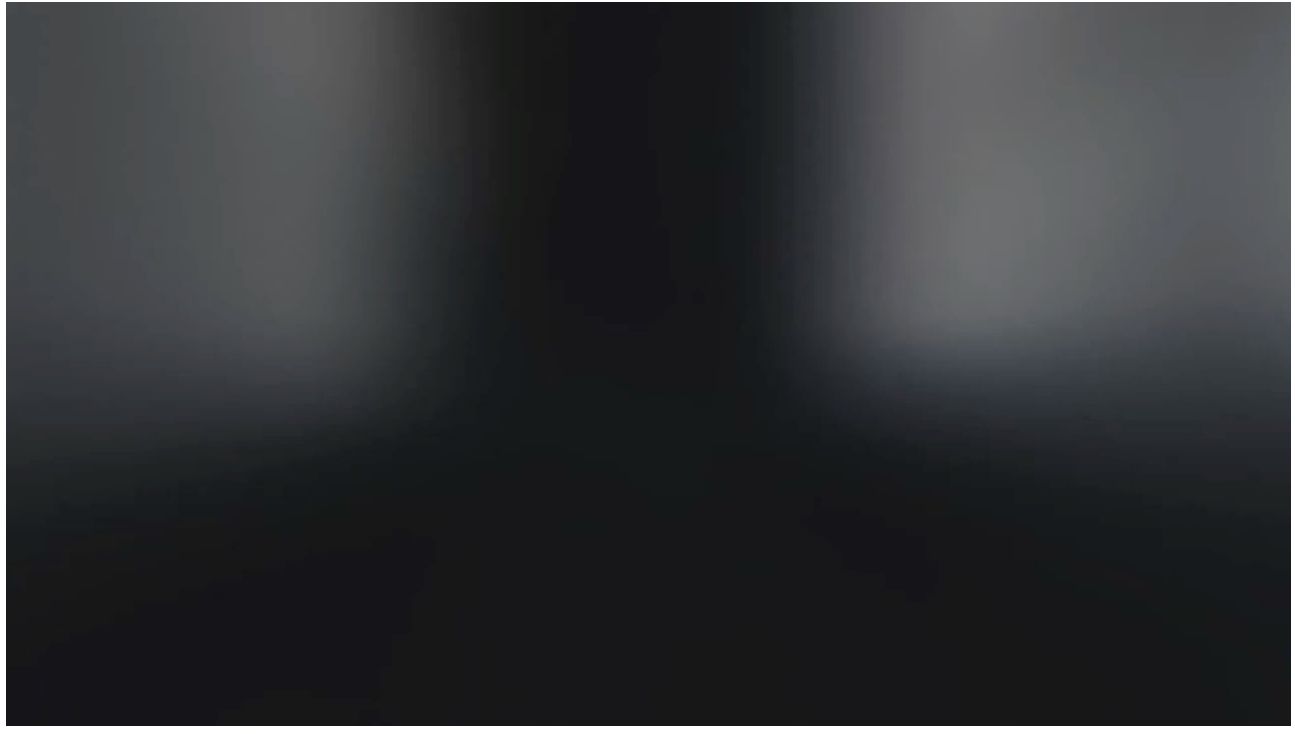
What is AI ?

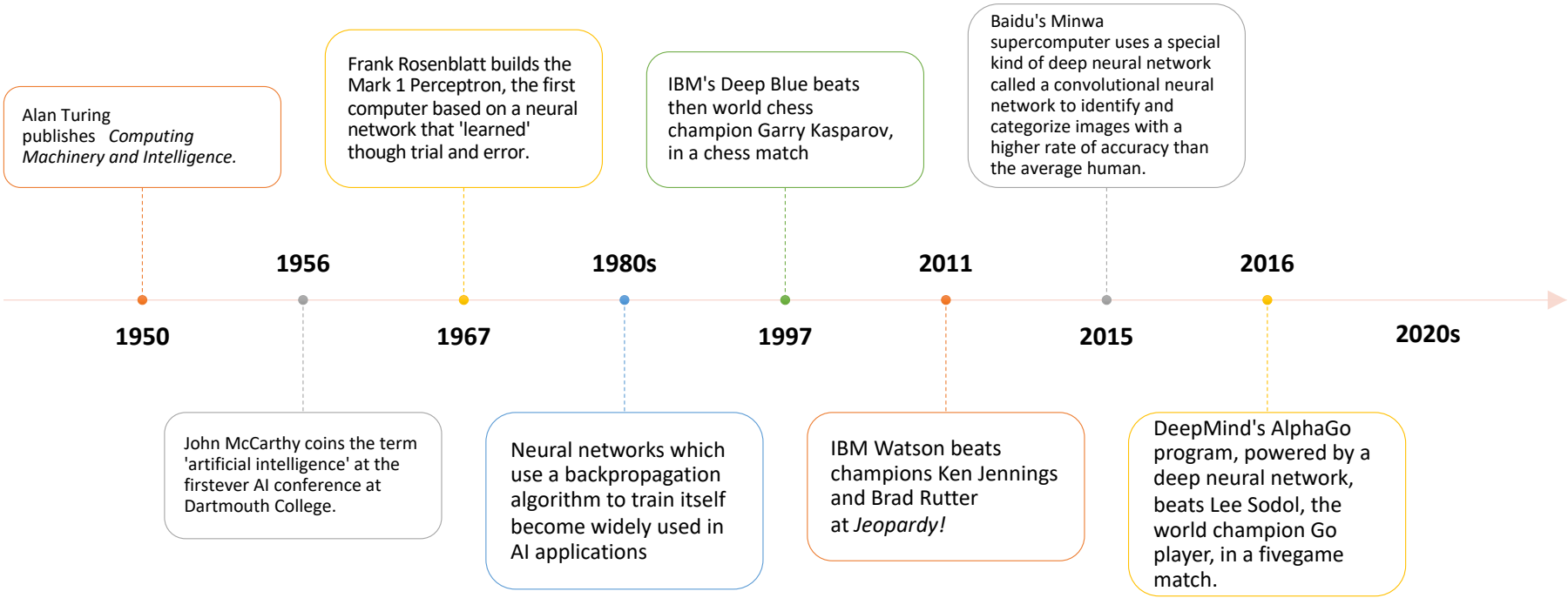
- **Artificial Intelligence** : It's the capability of a computer system to mimic humanlike cognitive functions.

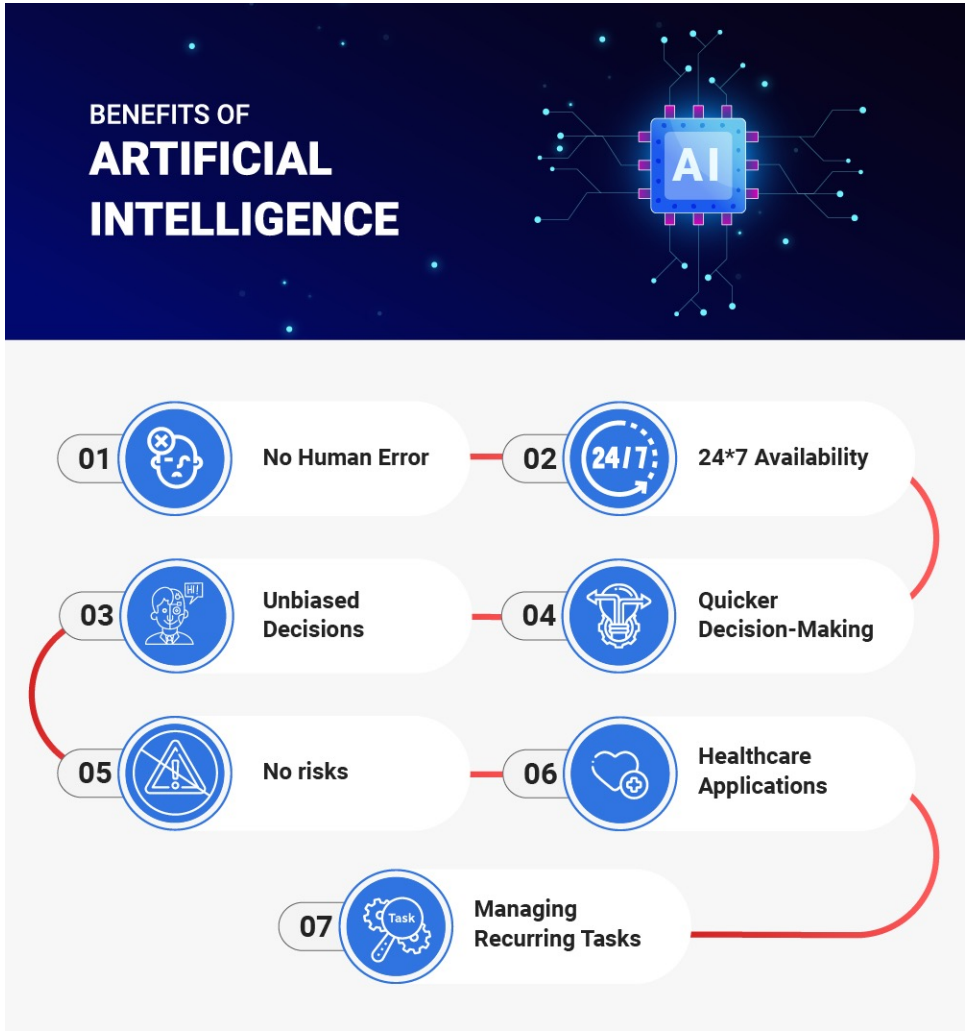
Mimic Human Brain

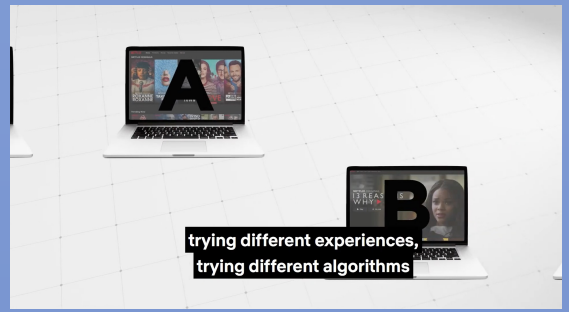
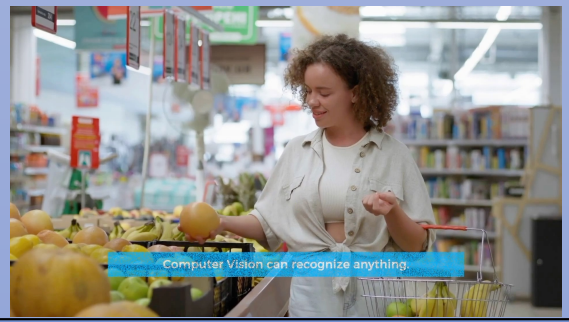
Neurons

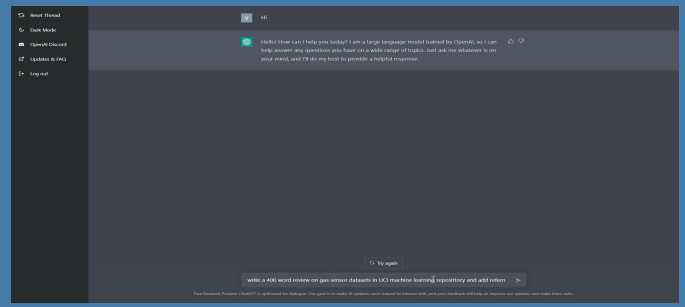
Demo on AI



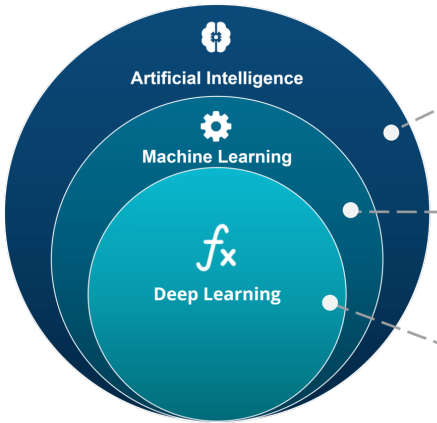

Alan Turing publishes *Computing Machinery and Intelligence.*
1950
John McCarthy coins the term 'artificial intelligence' at the firstever AI conference at Dartmouth College.
1956
Frank Rosenblatt builds the Mark 1 Perceptron, the first computer based on a neural network that 'learned' though trial and error.
1967
Neural networks which use a backpropagation algorithm to train itself become widely used in AI applications
1980s
IBM's Deep Blue beats then world chess champion Garry Kasparov, in a chess match
1997
IBM Watson beats champions Ken Jennings and Brad Rutter at *Jeopardy!*
2011
Baidu's Minwa supercomputer uses a special kind of deep neural network called a convolutional neural network to identify and categorize images with a higher rate of accuracy than the average human.
2015
DeepMind's AlphaGo program, powered by a deep neural network, beats Lee Sodol, the world champion Go player, in a fivegame match.
2016
2020s


BENEFITS OF
ARTIFICIAL
INTELLIGENCE
AI
01
No Human Error
02
24*7 Availability
03
Unbiased
Decisions
04
Quicker
Decision-Making
05
No risks
06
Healthcare
Applications
07
Managing
Recurring Tasks

Type of Workload	Description	Demo
Machine learning	The way we "teach" a computer model to make predictions and draw conclusions from data Supervised , Unsupervised Reinforced	
Computer vision	Capabilities within AI to interpret the world visually through cameras, video, and images. CNN, YOLO	
Natural language processing	Capabilities within AI for a computer to interpret written or spoken language, and respond in kind. lstm, GRU, word2vec , Glove unsupervised	

Type of Workload	Description	Demo
Generative AI	Capabilities within AI that create original content in a variety of formats including natural language, image, code, and more.	

AI Vs ML Vs DL



ARTIFICIAL INTELLIGENCE
A technique which enables machines to mimic human behaviour

MACHINE LEARNING
Subset of AI technique which use statistical methods to enable machines to improve with experience

DEEP LEARNING
Subset of ML which make the computation of multi-layer neural network feasible


Machine Learning
Foundations

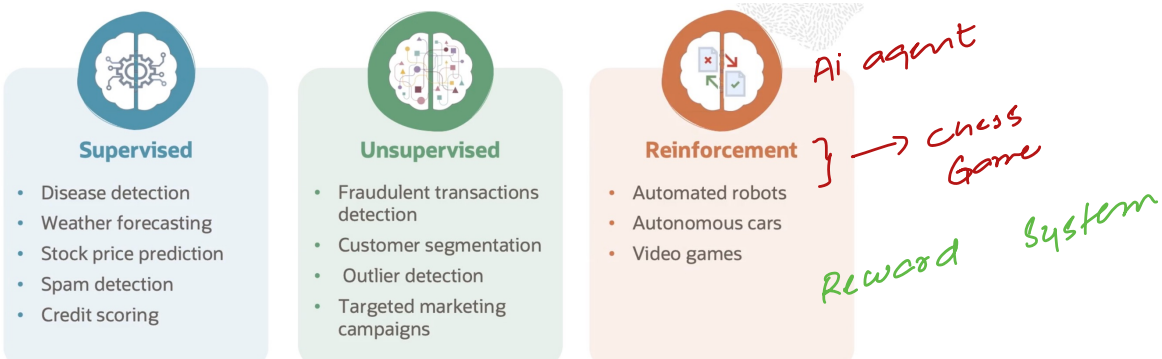
Machine Learning – Key Terminologies

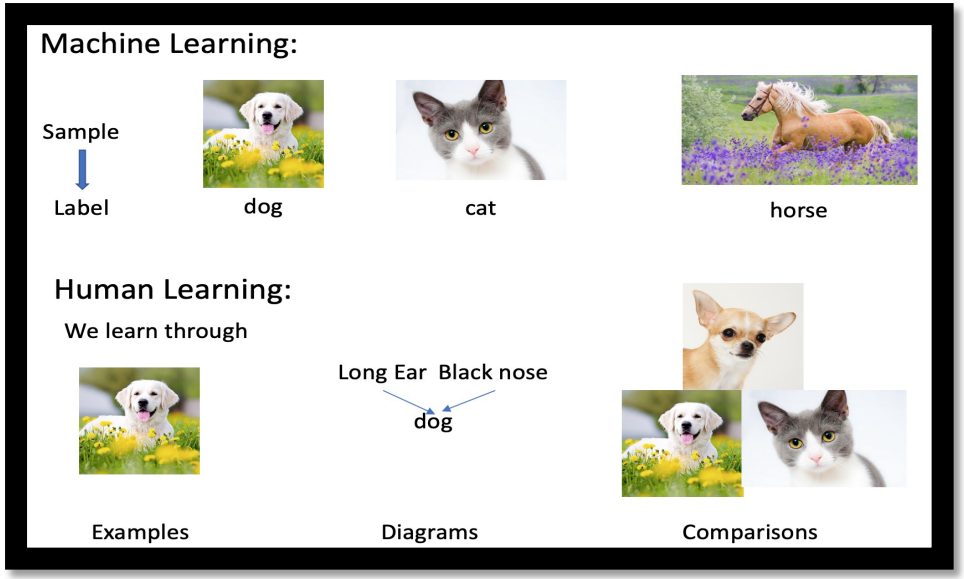
Artefact	Description	Example
Algorithm Statistics	In machine learning, an algorithm refers to a set of rules and statistical techniques used to learn from data.	The Decision Tree algorithm can be used to classify emails as "spam" or "not spam" based on features like the email's sender, the frequency of certain words, etc.
Model Algot Data	a model is what an algorithm creates after being trained on data. It's essentially the learned representation of the data. Program that can find patterns of make decisions.	After feeding a dataset of house features and prices to a regression algorithm, the model it creates can predict prices of new houses based on their features.
Training	This is the process of presenting data to a machine learning algorithm to create a model. The algorithm uses training data to learn.	A neural network is shown thousands of pictures of cats and dogs during training, and it learns to recognize features that distinguish one from the other.
Labels	labels are the outputs you want the model to predict.	In a dataset of tumor information, the label might be whether the tumor is malignant or benign.

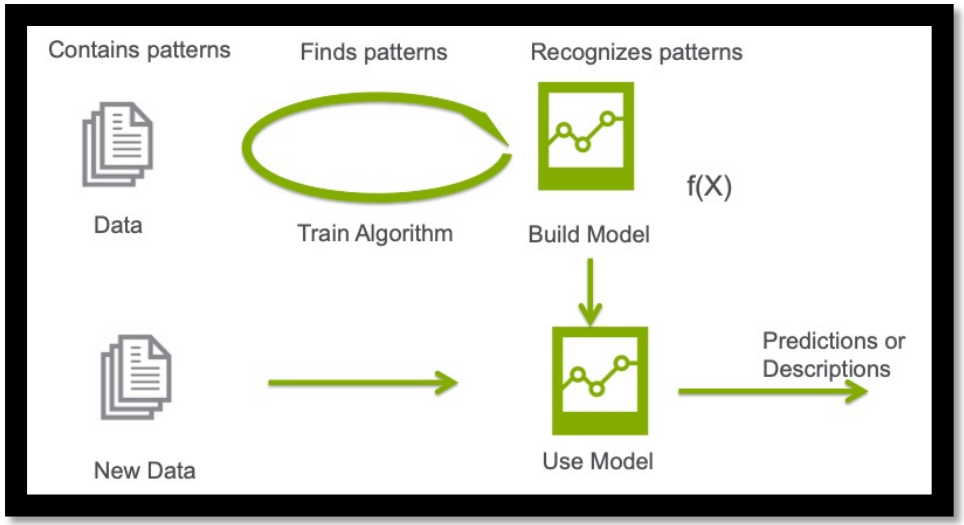
Machine Learning – Real World Example

Types of Machine Learning

Artefact	Description	Example
Supervised	The dataset being used has been prelabeled and classified by users to allow the algorithm to see how accurate its performance is. data acts as a teacher and “trains” the machine, increasing in its ability to make a prediction or decision.	email spam filtering. Here, the algorithm is trained with many example emails along with information whether they are "spam" (unwanted email) or "not spam". It learns to classify new emails into these categories based on the training it received.
UnSupervised Learning	The raw dataset being used is unlabeled and an algorithm identifies patterns and relationships within the data without help from users.	Market segmentation is a typical unsupervised learning task. Customer data is analyzed to find patterns and group customers into clusters based on similarities such as purchasing behavior, demographics, or usage statistics. No specific output labels are used; instead, the algorithm identifies the clusters on its own.
Reinforcement Learning	The dataset uses a “rewards/punishments” system, offering feedback to the algorithm to learn from its own experiences by trial and error. Replacing the human operator, an agent—a computer program acting on behalf of someone or something—helps determine outcome based upon a feedback loop.	chess game. The learning algorithm plays the game repeatedly and improves its strategy over time through a system of rewards (such as capturing an opponent's piece or winning the game) and penalties (like losing pieces or the game).







What is Machine Learning ?

- Machine learning is a branch of artificial intelligence (AI) and computer science which focuses on the use of *data* and *algorithms* to imitate the way that humans learn, gradually improving its accuracy.

- In machine learning, computers apply **statistical learning** techniques to automatically identify patterns in data.

- Those **patterns** are then used to create a data **model** that can make predictions.

- With increased data and experience, the results of machine learning are more accurate—much like how humans improve with more practice.

•**Learning Approaches**:
  - **Supervised Learning**: Learning from labelled data to predict outcomes for new data.
  - **Unsupervised Learning**: Identifying patterns and structures in unlabelled data.
  - **Reinforcement Learning**: Learning to make decisions by receiving rewards or penalties for actions.

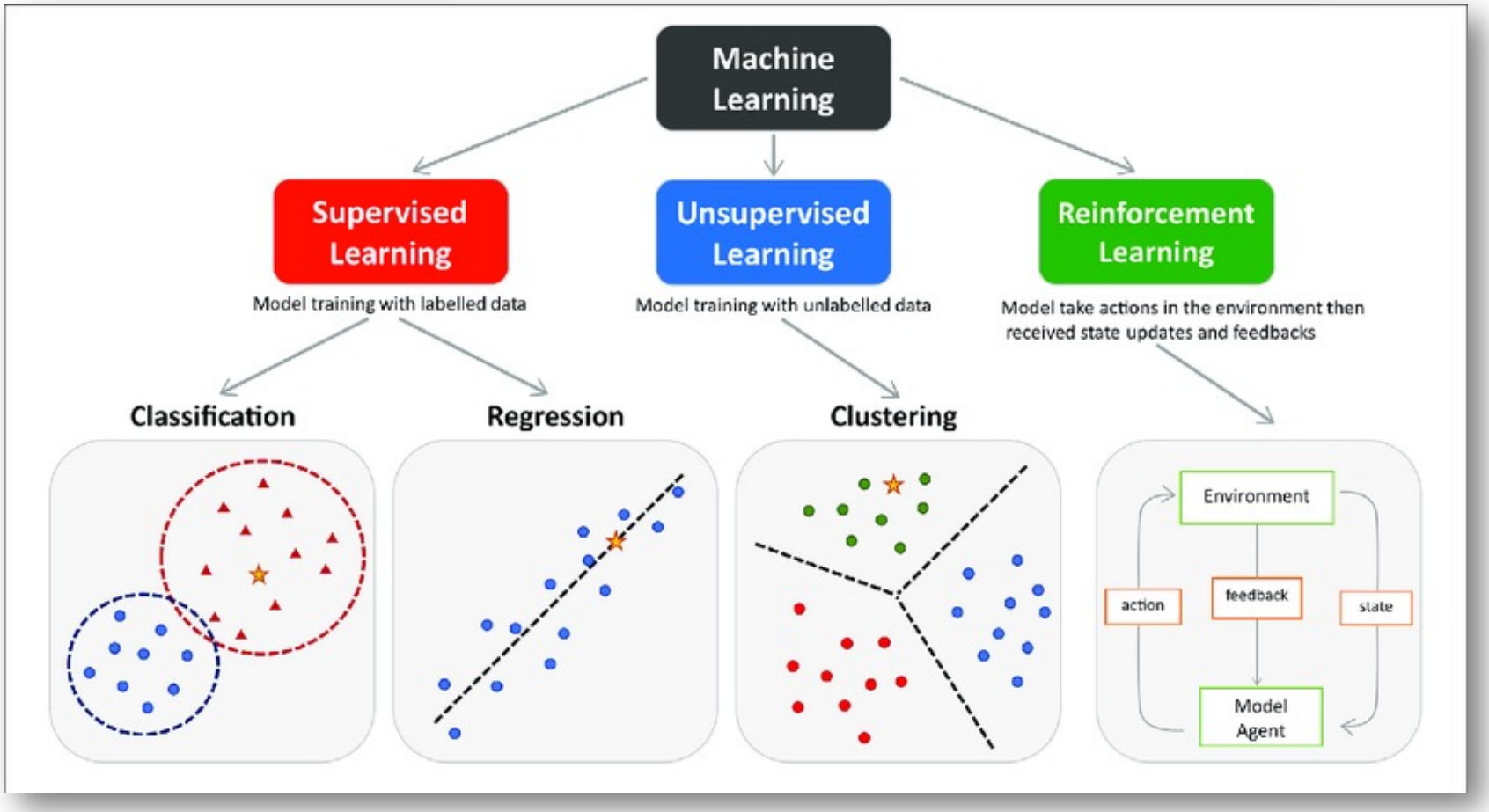
•**Data Centric**: Relies heavily on data *quality* and *quantity* for training models.

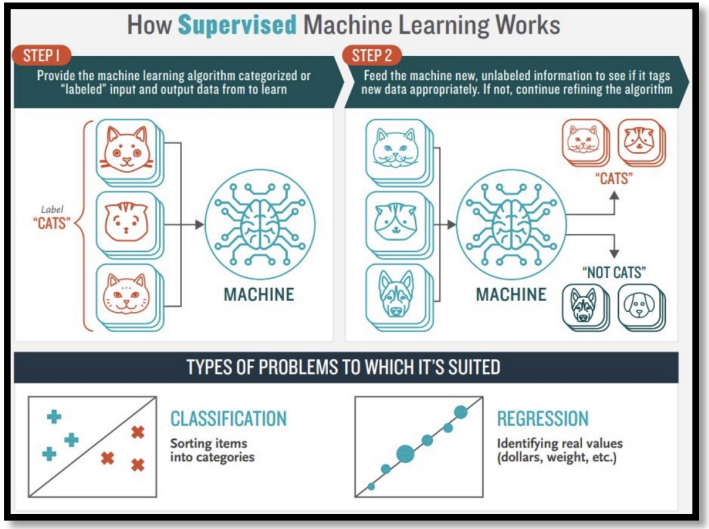
•**Algorithms**: Utilizes a variety of algorithms tailored to specific types of data and learning tasks.

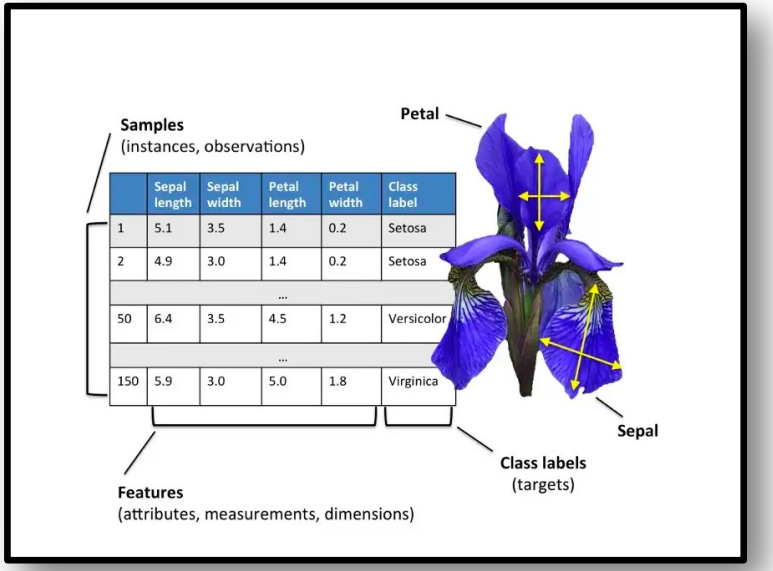
•**Applications**: Applied in diverse fields such as **healthcare**, **finance**, **autonomous vehicles**, and more for tasks like prediction, classification, and clustering.

•**Deployment**: Trained models are deployed in realworld applications to provide insights, automate tasks, or enhance decisionmaking.

Types of Machine Learning Algorithms



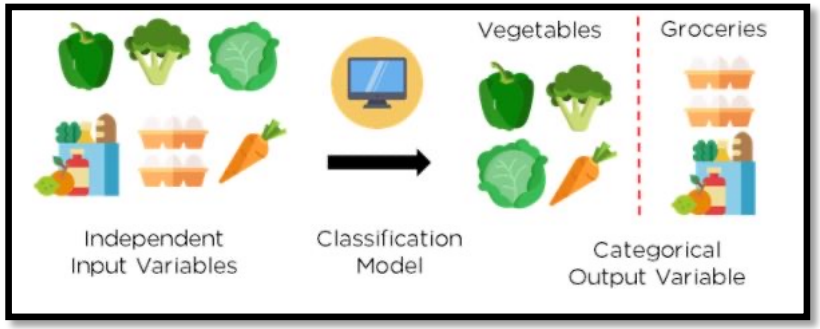


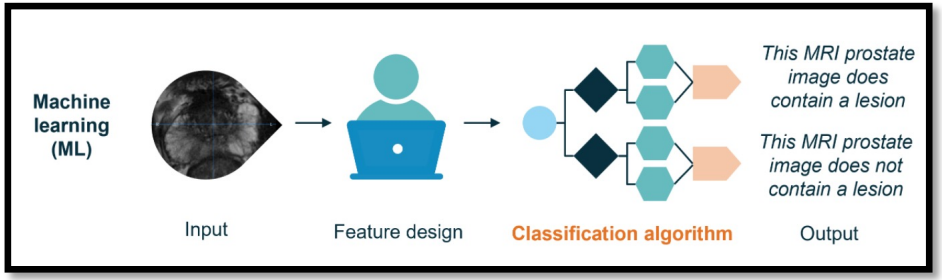


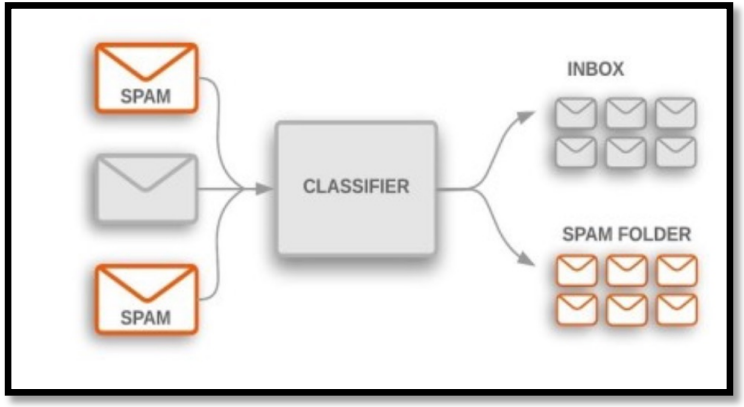
	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
...					
50	6.4	3.5	4.5	1.2	Versicolor
...					
150	5.9	3.0	5.0	1.8	Virginica

Supervised Machine Learning

- **Definition**: Supervised learning involves training a model on a dataset that includes both the **input data** and the corresponding **correct outputs**.
- **Labeled Data**: The training dataset must be labeled, meaning each example in the dataset is paired with the correct answer.
- **Algorithm Types**: Linear regression, Logistic regression, Support Vector Machines, Decision Trees and neural networks.
- **Training Process**: The model **learns a function** that maps inputs to desired outputs and makes predictions based on that function.
- **Overfitting Risk**: Supervised learning models can overfit the training data, meaning they may not perform well on new, unseen data.
- **Applications**: Used in various applications like **image** and **speech recognition**, medical diagnosis, spam detection, and stock price prediction.

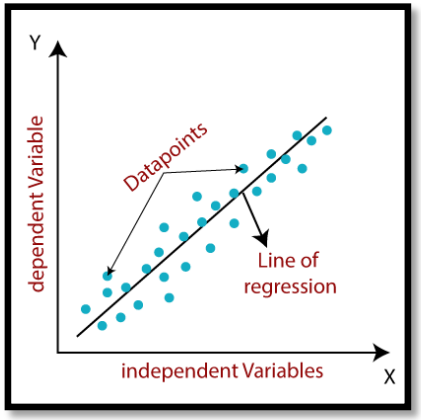


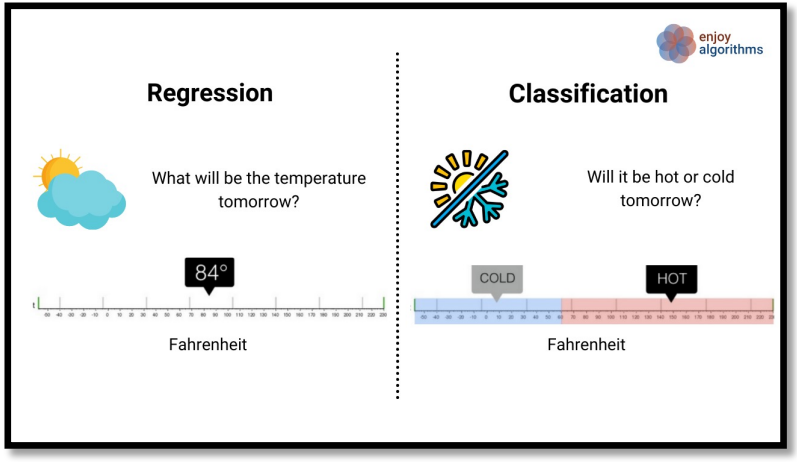




Supervised Machine Learning – Classification

- **Definition**: Classification involves assigning **category labels** to new observations based on past observations and their labels.
- **Binary vs. MultiClass**: Classification can be **binary** (**two classes**, e.g., spam or not spam) or **multiclass** (more than two classes, e.g., classifying types of fruits).
- **Common Algorithms**: Logistic regression, Decision trees, Random forests, support vector machines (SVM), and neural networks.
- **Feature Selection**: Choosing the right features (or inputs) is critical for building an effective classification model.
- **Labelling Data**: Requires a labelled dataset where each instance is tagged with the correct class for training.
- **Overfitting and Underfitting**: Balancing the complexity of the model to prevent overfitting (model is too complex) or underfitting (model is too simple).
- **Applications**: Used in various fields like **email spam detection**, image recognition, medical diagnosis, and sentiment analysis.
- **Realtime Decision Making**: Fraud detection in banking.





Supervised Machine Learning – Regression

1) **Definition**: Regression analysis predicts a continuous output (dependent variable) based on one or more predictor (independent) variables.

2) **Linear Regression**:
   - Simplest form of regression.
   - Assumes a **linear relationship** between input variables and the output.
   - Example: Predicting house prices based on size (square feet).

3) **Multiple Regression**:
   - Involves **two or more** independent variables.
   - Example: Predicting car prices based on features like age, mileage, brand, and engine size.

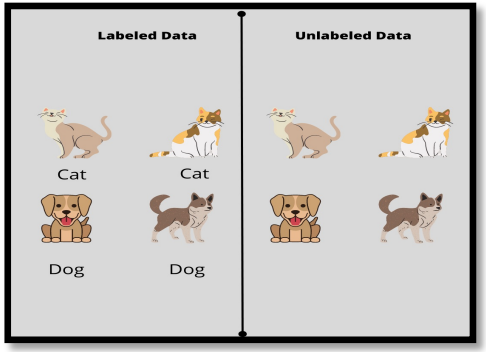
4) **Line of Regression :** regression line can be used to predict the value of y for a given value of x.

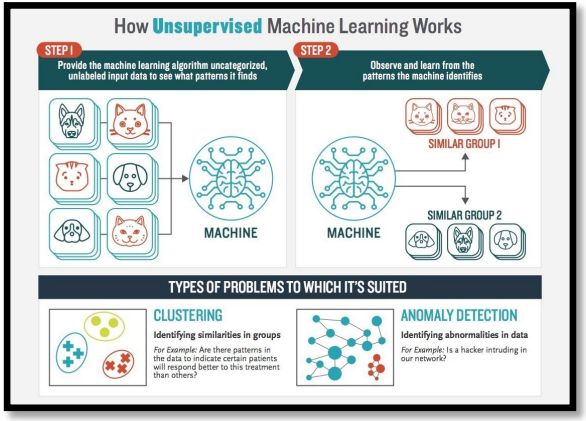
5) **Applications**:
   - Widely used in finance (stock prices prediction), healthcare (medical diagnoses), and many other fields.
   - Example: **Forecasting** sales in retail based on historical data and market trends.

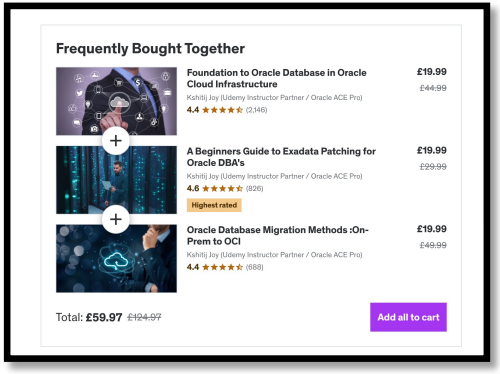
6) **Overfitting and Underfitting Considerations**:
   - Important to ensure the model fits the training data well but also generalizes to new, unseen data.
   - Example: A complex model might fit the training data on stock market prices very well but fails to predict future prices accurately, indicating overfitting.

UnSupervised Machine Learning (Clustering / Association)

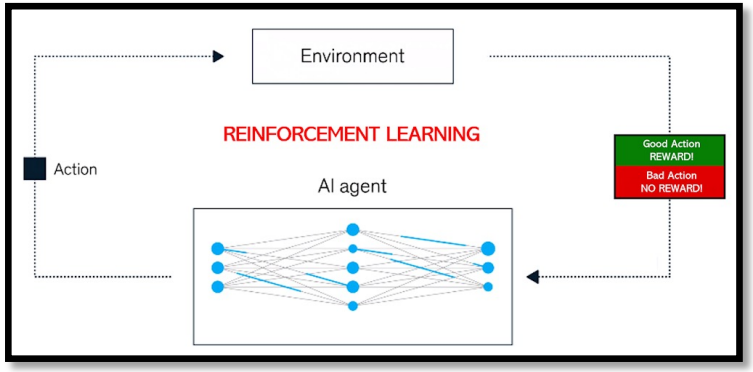
1) **Definition**: Unsupervised learning involves **analyzing** and **clustering unlabeled datasets** to discover hidden patterns or data groupings without human intervention.
2) **Data**: It deals with data that has not been labeled, classified, or categorized, and the algorithm tries to act on the data without any guidance.
3) **Clustering**:
    1) The most common unsupervised learning technique.
    2) It's used to **group data points into clusters** so that items in the same cluster are more similar to each other than to those in other clusters.
    3) Example: Market segmentation in business, where customers are grouped based on purchasing behavior or interests.
4) **Association**:
    1) This technique identifies sets of items in your dataset that frequently occur together.
    2) Example: In retail, finding products that are often bought together to optimize store layouts or crosspromoting products.
5) **Anomaly Detection**:
    1) The identification of unusual patterns that do not conform to expected behavior.
    2) Example: Detecting fraudulent transactions in banking by identifying patterns that deviate significantly from the majority of data.
6) **Applications**:
    1) Common in fields like **bioinformatics**, image and **speech recognition**, and recommendation systems.
    2) Example: **Recommender systems** in streaming services like Netflix or Spotify, which group users with similar preferences.
7) **Challenges**:
    1) More difficult than supervised learning due to the lack of labeled data.
    2) Example: Determining the right number of clusters in a dataset without predefined categories.







ReInforcement Machine Learning



1.Core Concept:

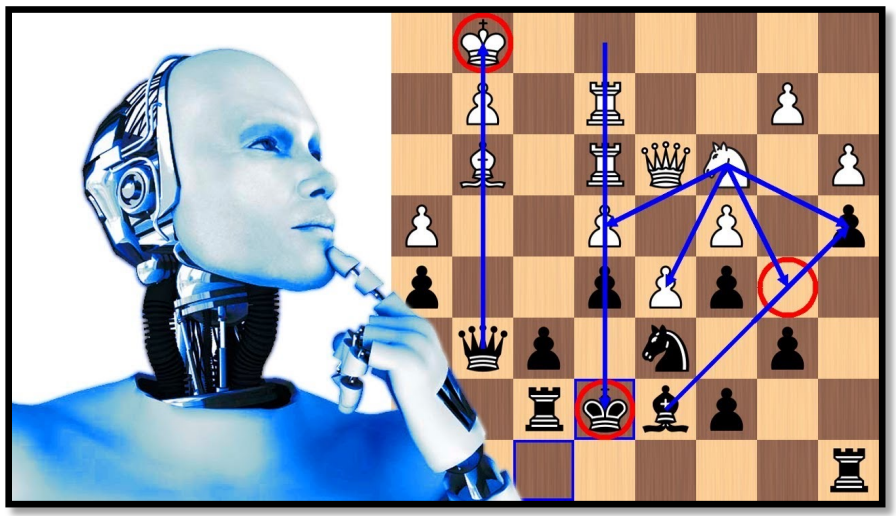
- Involves an agent learning to make decisions by performing actions and receiving feedback in the form of rewards or penalties.

2.Environment and Agent Interaction:

The agent interacts with its environment in discrete time steps. At each time step, the agent receives the environment's state, performs an action, and receives a reward.

3.Key Components:

- **Agent**: Learns from the environment.
- **Environment**: Where the agent performs actions.
- **Actions**: Set of all possible moves the agent can make.
- **State**: Current situation returned by the environment.
- **Reward**: Feedback from the environment.



4.Learning Process:

The agent learns a policy: a strategy of choosing an action given a state, to maximize cumulative reward over time.

5.Exploration vs. Exploitation:

- Balancing exploration (trying new things) with exploitation (using known information) is a key challenge.

6.Applications:

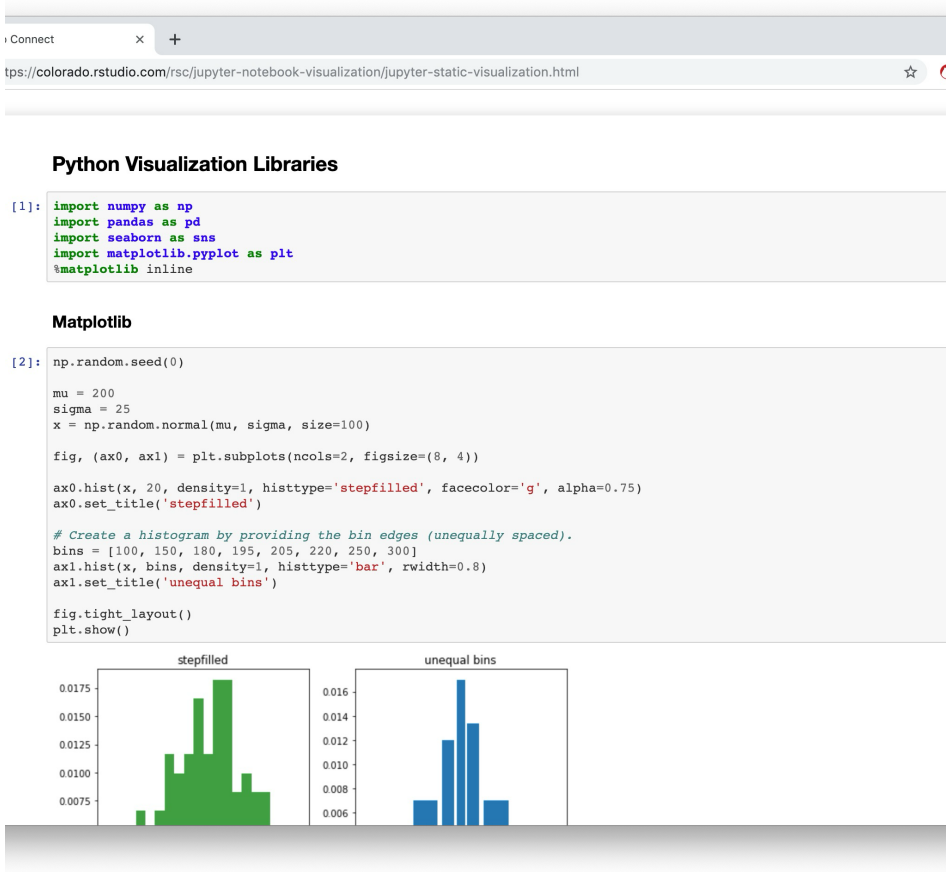
- Game playing (e.g., chess, Go).
- Robotics for control tasks.
- Autonomous vehicles.

7.Challenges and Considerations:

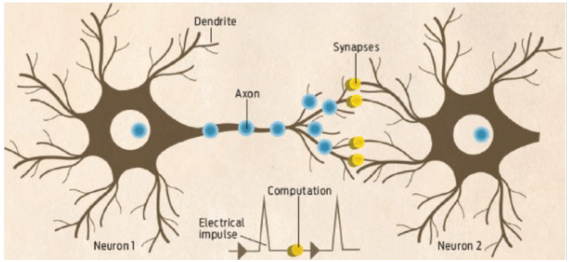
- Requires a lot of data and computational resources.
- Balancing exploration and exploitation can be complex.
- Dealing with environments with a lot of variability or uncertainty.

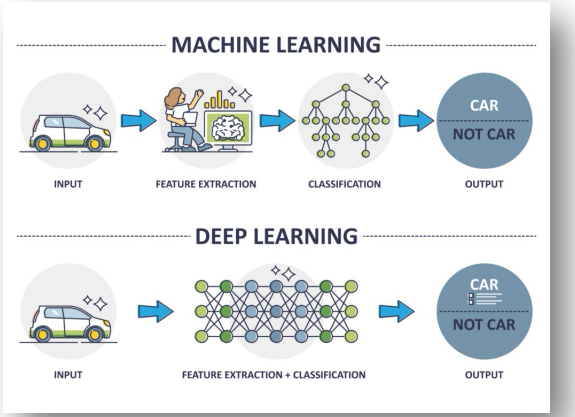
What is a Jupyter Notebook ?

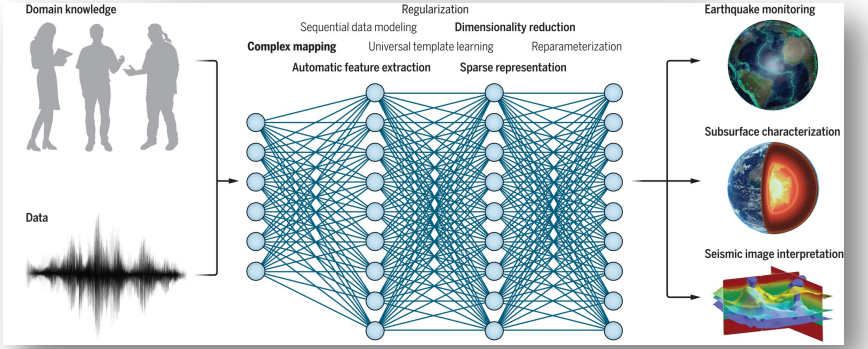
- **Interactive Computing Platform**: Jupyter Notebooks provide an interactive environment for writing and running code in a variety of programming languages, most notably Python.
- **Live Code Execution**: Code cells within a notebook can be executed independently and the results are displayed immediately below the code cell.
- **Support for Multiple Languages**: Primarily used for Python, it also supports over 40 programming languages including R, Julia, and Scala.
- **Rich Text Elements**: Notebooks can include narrative text (Markdown), equations (LaTeX), images, and links, allowing for comprehensive documentation of the code and its explanations.
- **Data Visualization**: Integration with data visualization libraries like Matplotlib, Plotly, and Bokeh makes it easy to create interactive graphs and charts directly within a notebook.
- **Collaboration and Sharing**: Notebooks can be shared with others and can be converted into a variety of formats (like HTML, PDF, and slides) for easy presentation and sharing.
- **Educational and Research Tool**: Widely used in academia and research for teaching programming, computational thinking, and data analysis.
- **Integration with Data Science Tools**: Seamlessly integrates with popular data science tools and platforms, facilitating tasks such as data cleaning, statistical modeling, machine learning, and more.



Deep Learning Foundations

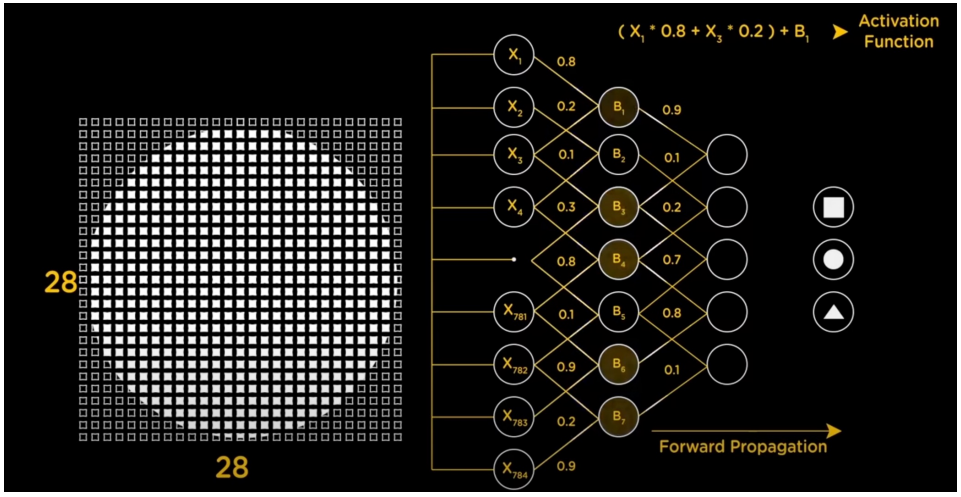


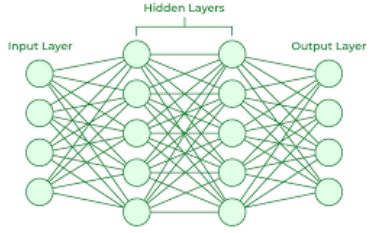


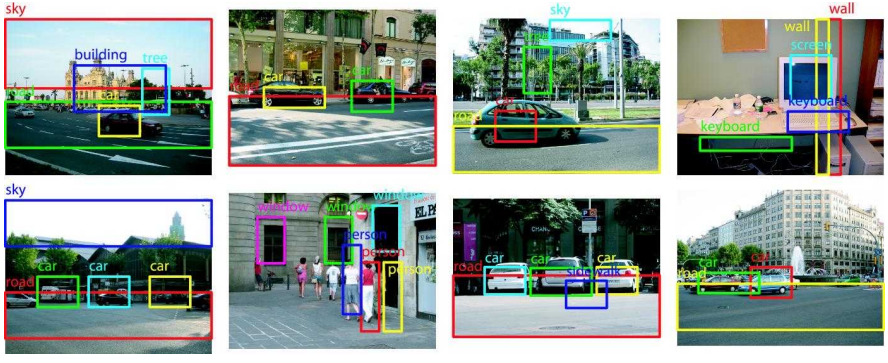


What is Deep Learning ?

1. **Definition**: Deep learning is a machine learning technique that teaches computers to process data in a way that is inspired by the human brain.
2. **Neural Networks**: At the heart of deep learning are artificial neural networks, which are algorithms inspired by the structure and function of the brain.
3. **A human brain** contains millions of interconnected neurons that work together to learn and process information.
4. **Artificial neural networks**, are made of many layers of artificial neurons that work together inside the computer.
5. **Automates feature extraction**, removing some of the dependency on human experts.
6. **Layers of Neurons**: Deep learning models consist of layers of interconnected nodes or neurons, and 'deep' refers to the number of layers through which the data is transformed.
7. **Learning from Unstructured Data**: Deep learning excels at learning from unstructured data like images, text, or sound.Example: Recognizing faces in images using Convolutional Neural Networks (CNNs).
8. **Large Data Requirements**: Generally requires large amounts of labeled data for training.
9. **Powerful Computational Resources**: Requires significant computational power, often using GPUs or specialized hardware.
10. **Versatile Applications**: Used in various applications like autonomous vehicles, medical diagnosis, natural language processing,
11. **Frameworks and Tools**: Tools like TensorFlow, PyTorch, Keras, and others are commonly used for building and training deep learning models.







What is a Neural Network ?

1. **Basic Concept**: A neural network is a machine learning program, or model, that makes decisions in a manner similar to the human brain, by using processes that mimic the way biological neurons work together to identify phenomena

2. **Structure**: Comprised of layers of interconnected nodes, or neurons, including input layers, hidden layers (one or more), and an output layer.

3. **Neuron Functionality**: Each neuron performs a simple calculation (like weighted sum followed by a nonlinear function) on its inputs.

4. **Learning Process**: Neural networks learn to perform tasks by considering examples, generally without taskspecific programming. For example, they might learn to identify images that contain cats by analyzing examples of cat images and noncat images.

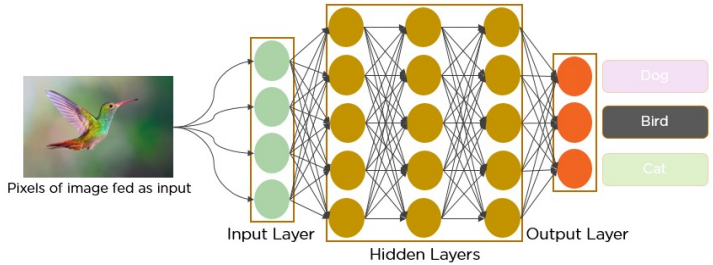
5. **Weights and Biases**: Connections between neurons have weights that adjust as learning proceeds. The network also uses biases, an extra input to nodes (neurons) in each layer to help them make better decisions.

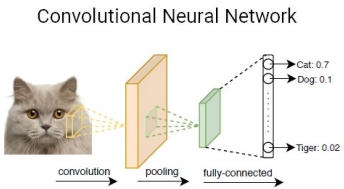
6. **Types of Neural Networks**: Includes feedforward neural networks, convolutional neural networks (CNNs), recurrent neural networks (RNNs), and more.

7. **Examples**:
   - **Image Recognition**: CNNs are used to identify objects, faces, scenes, and other elements in images.
   - **Speech Recognition**: RNNs are commonly used for speech recognition in virtual assistants like Siri or Alexa.
   - **Predictive Text**: Neural networks are used in keyboards on smartphones to predict the next word a user might type.

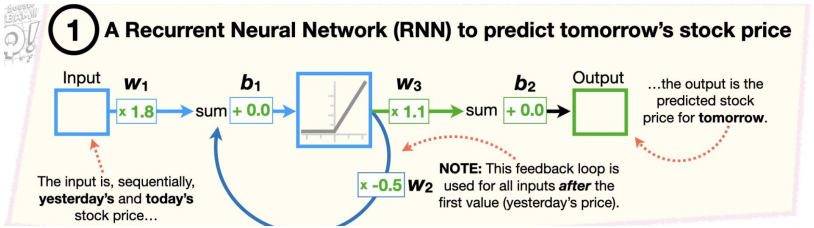
8. **Applications**: Beyond these examples, neural networks are used in a multitude of applications including selfdriving cars, medical diagnosis, stock market trading, and more.

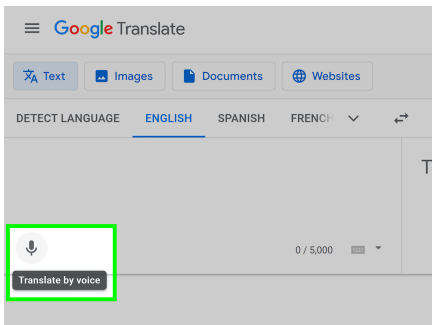
Deep Fake Video

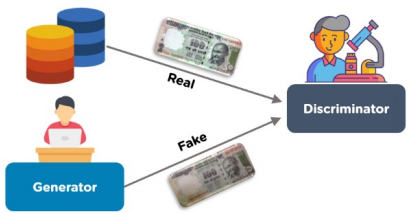


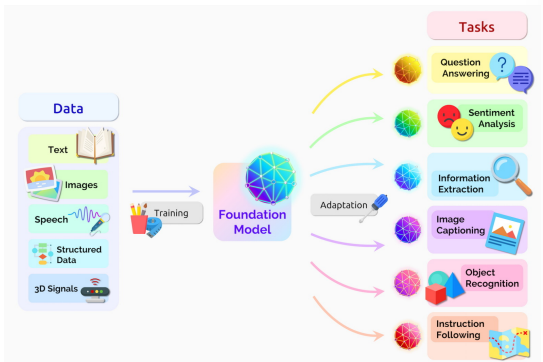


Deep Learning Models

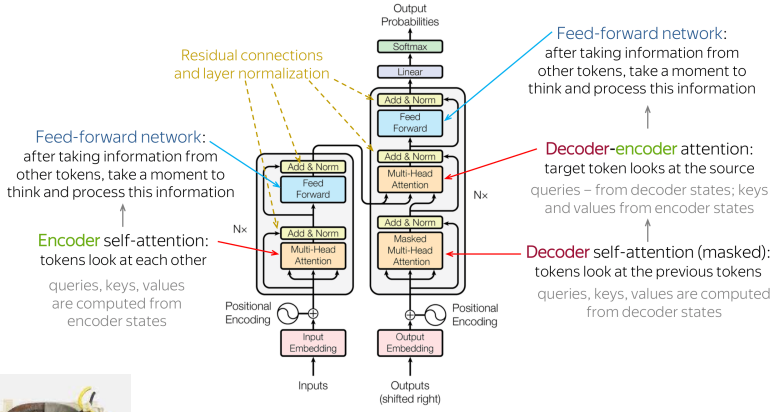


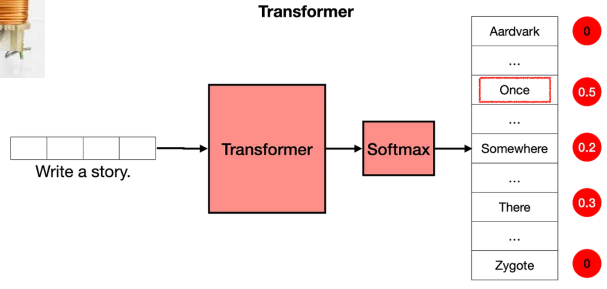




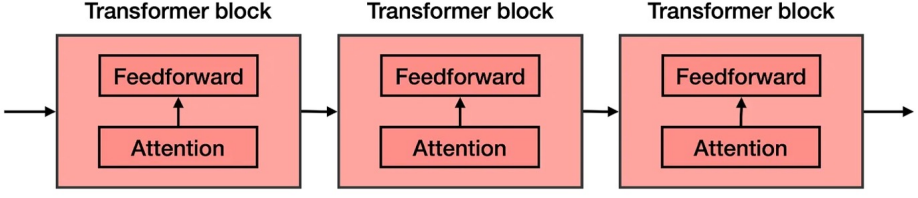


1. **Convolutional Neural Networks (CNNs)**:
   - Ideal for image and video processing.
   - Example: Image classification in social media platforms, like distinguishing between pictures of cats and dogs.
2. **Recurrent Neural Networks (RNNs)**:
   - Designed for sequential data, such as time series or natural language.
   - Example: Predicting the next word in a sentence for text autocomplete features.
3. **Long ShortTerm Memory Networks (LSTMs)**:
   - A type of RNN effective in learning from sequences with longrange dependencies.
   - Example: Used in machine translation services like Google Translate
4. **(GANs)**:
   - Consists of two networks, a generator and a discriminator, that compete against each other.
   - Example: Creating photorealistic images or deepfakes.
5. **Transformer Models**:
   - Based on selfattention mechanisms for handling sequential data.
   - Example: OpenAI's GPT3 for generating humanlike text in applications like chatbots or content creation.





The softmax layer turns the scores into probabilities, and these are used to pick the next word in the text.



The transformer is a concatenation of many transformer blocks. Each one of these is composed by an attention component followed by a feedforward component (a neural network).

Parallel operation

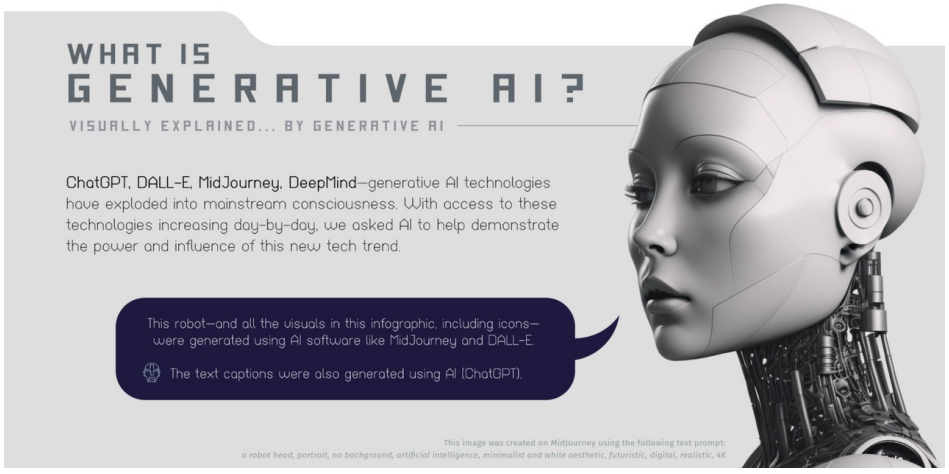
What is Transformer Model ?

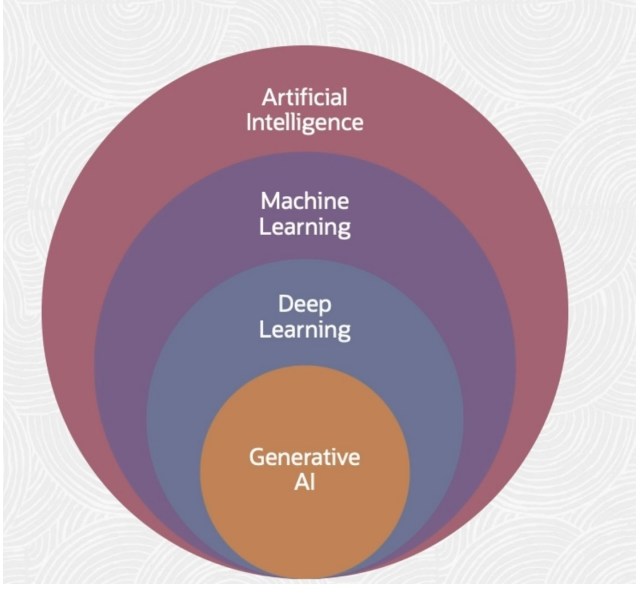
Transformers are a type of neural network architecture that have revolutionized the field of natural language processing (NLP). Here are some key points about transformer models:

1. **Architecture:** Transformers use a unique architecture primarily based on selfattention mechanisms, allowing them to weigh the importance of different parts of the input data.
2. **Parallel Processing:** Unlike previous sequencebased models like RNNs and LSTMs, transformers process entire sequences of data in parallel, significantly speeding up training and inference times.
3. **Attention Mechanism:** The core of the transformer is the attention mechanism, which allows the model to focus on different parts of the input sequence when producing output, enhancing its ability to understand context and relationships in the data.
4. **No Recurrence:** Transformers do not require recurrent layers. This absence of recurrence means they can handle longer sequences of data more effectively than RNNs and LSTMs.
5. **Scalability:** The parallel nature of transformers makes them highly scalable with increased data and computational resources, leading to improvements in performance.
6. **Layer Structure:** A typical transformer model consists of an encoder and a decoder, each comprising multiple layers of selfattention and feedforward neural networks.
7. **Application in NLP:** Transformers have become the backbone of many stateoftheart NLP models, used in tasks like machine translation, text generation, summarization, and questionanswering.
8. **BERT and GPT:** Notable implementations of transformer models include BERT (Bidirectional Encoder Representations from Transformers) for understanding context in language, and GPT (Generative Pretrained Transformer) for generating humanlike text.
9. **Beyond NLP:** While initially designed for NLP tasks, the transformer architecture has also been adapted for use in other areas, such as computer vision and audio processing.

Context :

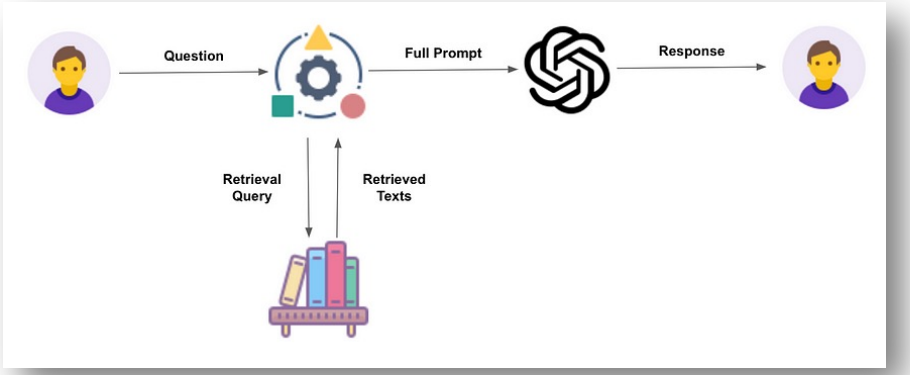
- Sentence 1: The **bank** of the river.
- Sentence 2: Money in the **bank**.

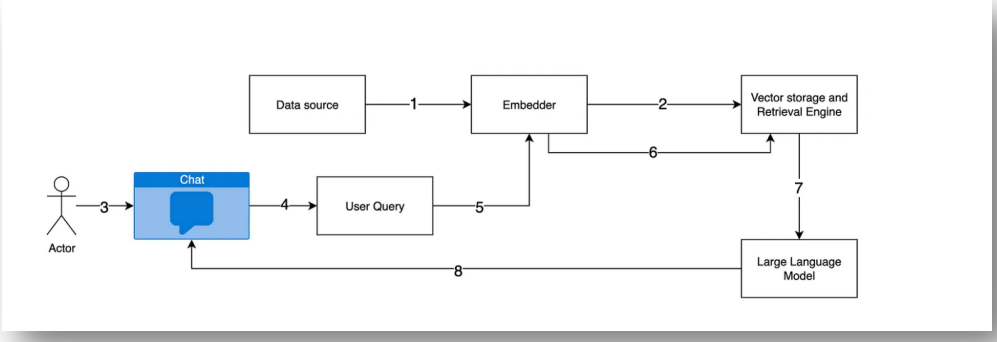




What is Generative AI ?

1. **Content Creation**: Generative AI can create a wide range of content, including text, images, music, and even code.
2. **Learning from Data**: These systems learn from large datasets to understand patterns and styles, enabling them to produce similar outputs.
3. **Deep Learning Models**: Many generative AI systems use deep learning techniques, particularly neural networks, to generate content.
4. **Applications**: Generative AI is used in various fields such as art, entertainment, marketing, and software development.
5. **Improving Efficiency**: By automating the creation process, it can significantly reduce the time and effort required to produce content.
6. **Challenges and Ethical Considerations**: There are concerns regarding originality, copyright, and the potential misuse of generated content.
7. **Continual Evolution**: Generative AI technologies are rapidly evolving, becoming more sophisticated over time.
8. **UserGenerated Input**: These systems can start with a simple user input (like a text prompt) and expand it into a fullfledged creation.
9. **Interdisciplinary Impact**: Its impact is felt across various disciplines, revolutionizing traditional methods in fields like journalism, design, and education.





What is RetrieverAugmented Generation RAG ?

Combines Retrieval and Generation: RAG uses a retriever to find relevant information and a generator to create responses based on that info.

Example: If asked "What is climate change?", RAG retrieves scientific articles on the topic and then generates a concise explanation.

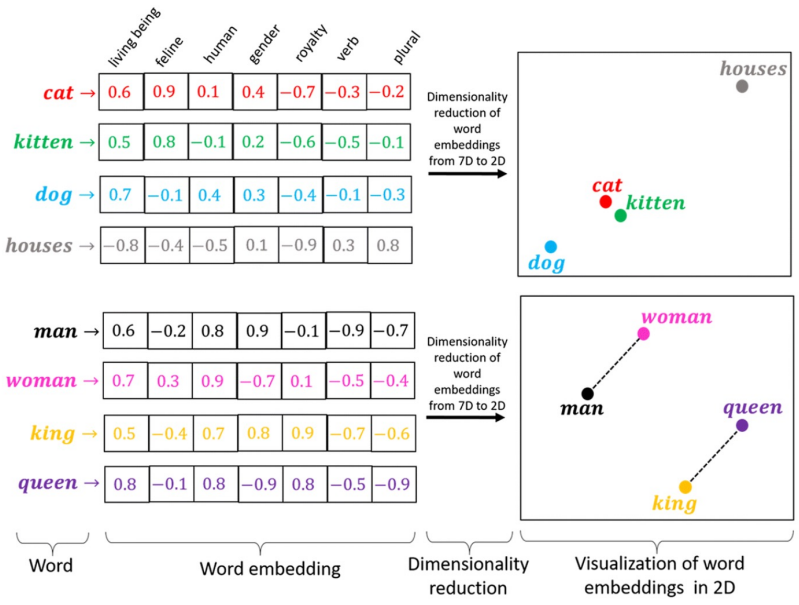
Improves Response Quality: By using external information, RAG provides more accurate and detailed answers.

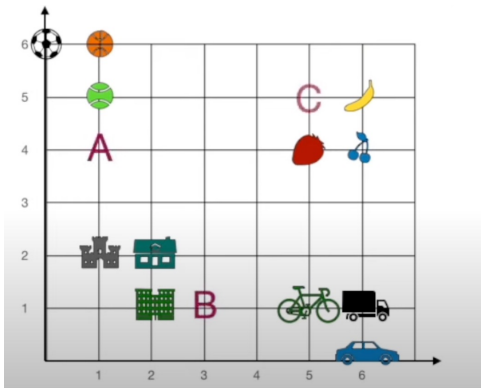
Applicable in Various Domains: The model is adaptable for different uses like customer service chatbots or educational tools, where precise information is crucial.

Enhances Contextual Awareness: RAG's use of relevant documents ensures responses are contextually appropriate, reducing errors seen in purely generative models.

Workflow:

- User submits a query.
- Query is transformed into a vector representation.
- Relevant documents are retrieved from a data source based on similarity.
- LLM utilizes both the query and retrieved documents (or their context) to generate a final response.





What is Embedding ?

An embedding is a way of converting the features of an object, like a word, into a vector of real numbers.

Transforming Data: Embeddings turn complex data (like words or pictures) into vectors or a list of numbers that a computer can understand and work with.

Finding Similarities: They help find similarities between items. For example, in word embeddings, similar words are represented by numbers that are close together.

Making Data Smaller: They help make big, complicated data simpler and smaller, so it's easier for computers to handle.

Used Everywhere: Embeddings are used in many areas, like helping computers understand language or recommend things you might like.

Different Types: Word2Vec (Google) / GloVe (Global Vectors for Word Representation) / BERT (Bidirectional Encoder Representations from Transformers)

Learning from Data: Computers can learn to create these embeddings by looking at lots of examples, which helps them understand and predict new, unseen data.

Improving Understanding: Some advanced embeddings can even understand the context, meaning they can tell the difference in meaning when a word is used in different ways.

What is Langchain ?

LLM app

Integrating LLM into Apps

Just as **LEGO** bricks are assembled following a structured guide to create a **final model**, LangChain integrates various data sources with AI language models to build an application.

Langchain is a technology framework designed for integrating language models into applications effectively.

Integration of Language Models: Simplifies the process of incorporating sophisticated language AI models into various software applications.

Modular and Flexible: Provides modular components that developers can mix and match to build custom solutions tailored to specific needs.

DeveloperFriendly: Designed to be accessible for developers, reducing the complexity typically associated with implementing AIdriven conversational agents.

Simple Example:

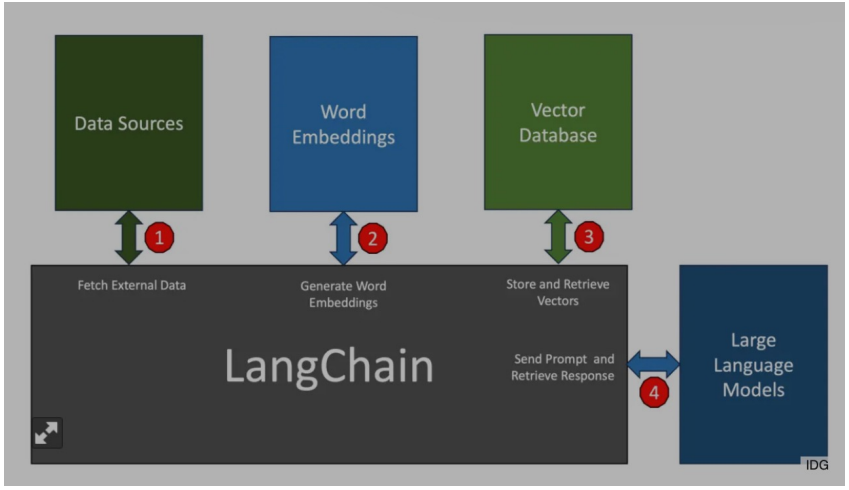
User Query:
A user types or asks, "What is the weather like in Chandigarh today?" using an app integrated with LangChain.

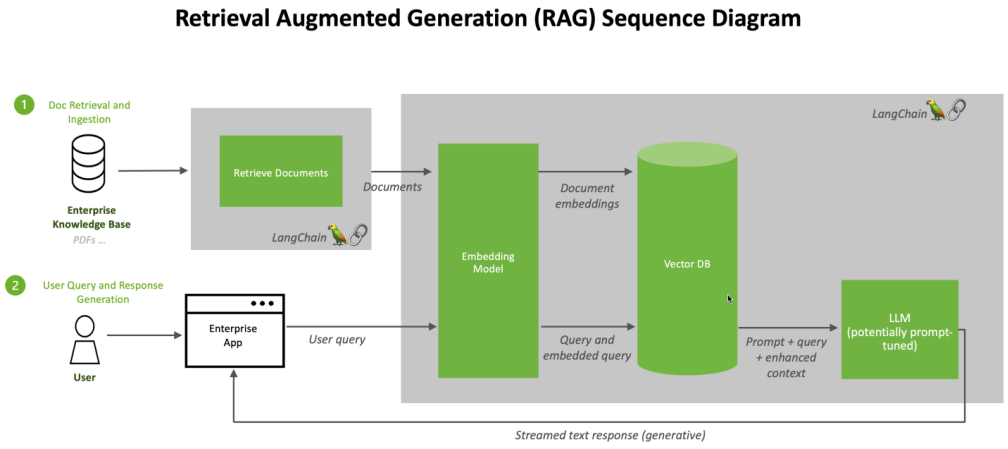
Retrieve Data:
LangChain processes the query and identifies that it needs weather data. It accesses an external database or API specifically set up for weather information.

Data Processing:
The system retrieves the current weather data for Chandigarh from the weather database or API. Let's say the data shows it's "sunny and 40°C".

Generate Response:
LangChain uses a language model to generate a natural language response based on the retrieved data. The model formats the response to be informative and userfriendly.

Deliver Response:
The final response, "The weather in Chandigarh today is sunny with a temperature of 40°C," is delivered to the user through the app.





Role of Langchain in a RAG

1.Document Retrieval and Ingestion:

Langchain's Role: Langchain appears to facilitate the retrieval of documents from an "Enterprise Knowledge Base" (which might consist of PDFs and other documents).

2. Processing and Embeddings:

Langchain's Role: After documents are retrieved, they are likely processed through an embedding model to convert text content into vector embeddings

3. Vector Database:

Langchain's Role: The document embeddings are stored in a vector database. This database is used to perform efficient similarity searches between the query embeddings and the document embeddings.

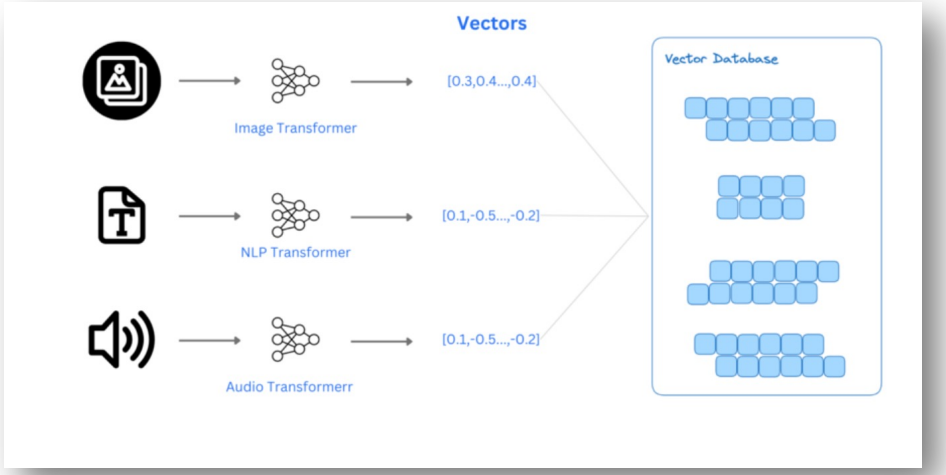
4. Query and Enhanced Context Preparation:

Langchain's Role: Once the relevant documents (or their embeddings) are identified, Langchain helps in preparing the enhanced query context by combining the original user query with information retrieved from the vector database

5. Response Generation:

Langchain's Role: Langchain likely manages the interaction with this language model, ensuring that the input is correctly formatted and that the generation process is effectively executed.

Indexing & Querying vectors



What is Vector Database ?

A vector database is a type of database designed for storing, indexing, and querying vectors, which are arrays of numbers representing data in highdimensional space.

Purpose: Primarily used for storing embeddings that represent complex data like images, text, and audio in a form that machines can understand and process.

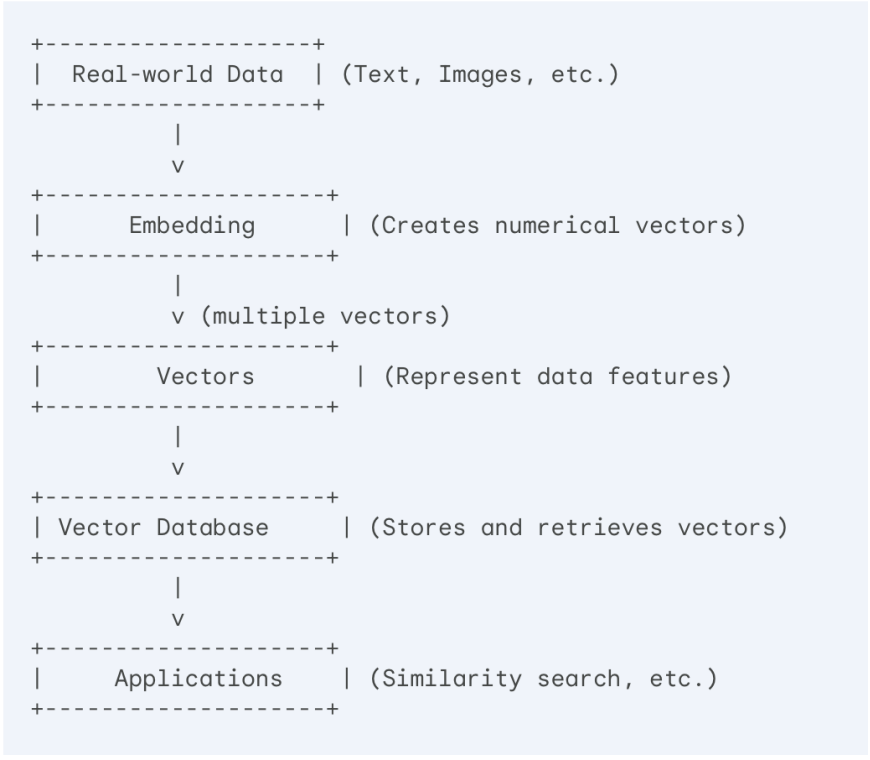
Similarity Search: Optimized for similarity search, allowing quick retrieval of items that are most similar to a given query in terms of vector distance, such as Euclidean distance or cosine similarity.

Applications: Widely used in machine learning, artificial intelligence, and recommendation systems for tasks like image recognition, natural language processing, and personalized content discovery.

Scalability: Designed to handle large volumes of data, supporting scalability and efficient querying in big data applications.

Integration with ML Models: Often integrated with machine learning models to directly store and query modelgenerated embeddings, facilitating dynamic and realtime applications.

Realtime Performance: Capable of providing realtime search and retrieval performance, essential for interactive applications and services.



Embeddings vs Vector Databases

Realworld Data: This represents the data you want to work with, such as text documents, images, or even sounds.

Embedding: This is a process that transforms the realworld data into numerical vectors. The embedding captures the important features of the data.

Vectors: These are the numerical representations of the data created by the embedding process. Each vector is a list of numbers that encodes the data's characteristics.

Vector Database: This is a specialized database designed to store and efficiently retrieve large collections of vectors. It uses techniques to find similar vectors quickly.

Applications: Vector databases are used in various applications, such as similarity search (finding similar data points) and recommendation systems.

Predictive AI Vs Generative AI

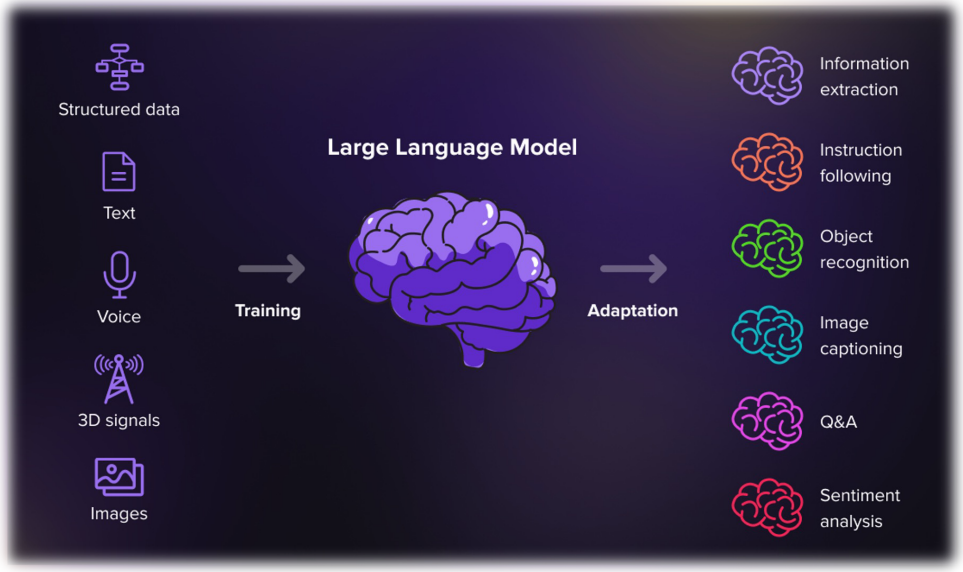
Predictive AI:

- **Function:** Predictive AI is primarily designed to analyze existing data and make predictions about future events or outcomes based on that data.
- **Data Analysis:** It often involves statistical methods and machine learning algorithms to identify patterns or trends in historical data.
- **Applications:** Commonly used in forecasting (like weather, stock prices), risk assessment, and customer behavior prediction.
- **Decision Support:** Provides insights and recommendations, aiding in decisionmaking processes.
- **Reactive Nature:** Tends to react to existing data, making predictions about what will happen next.
- **Examples:** Credit scoring models, demand forecasting in supply chain management, and predictive maintenance in manufacturing.

Generative AI:

- **Function:** Generative AI focuses on creating new data that resembles the training data, often creating entirely new and original outputs.
- **Content Creation:** Capable of generating text, images, music, and other forms of media.
- **Applications:** Used in art and design, generating synthetic data for training other AI models, and creating realistic simulations.
- **Creative Potential:** Has the ability to produce creative and novel outputs, going beyond the scope of the input data.
- **Proactive Nature:** Proactively creates new data points rather than just analyzing and reacting to existing data.
- **Examples:** Deep learning models like GPT (for text) and DALLE (for images), music composition AI, and AI for generating synthetic datasets.

.



What is LLM ?

Large Language Models (LLMs) are sophisticated AI systems designed for processing, understanding, and generating human language. Here are key points about them, with an example for clarity:

1. **Scale and Complexity:** LLMs are composed of millions or billions of parameters, making them highly complex. Example: GPT4, with its vast number of parameters, can generate humanlike text.

2. **Training Data:** They are trained on vast datasets comprising a wide range of internet text. Example: GPT4 is trained on a diverse dataset that includes books, websites, and other texts.

3. **Natural Language Understanding and Generation:** LLMs are adept at understanding and generating humanlike language. Example: GPT4 can write essays, poems, or even simulate conversation.

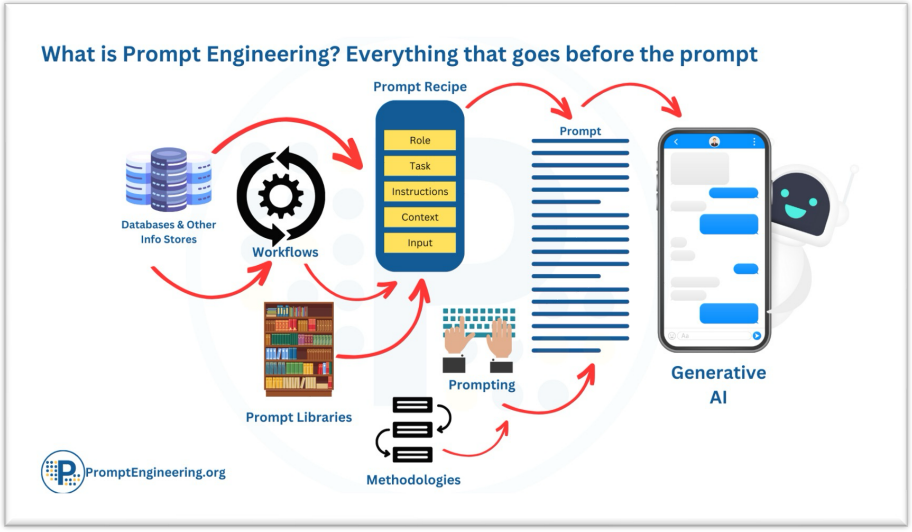
4. **Applications:** They are used in various applications like chatbots, content creation, language translation, and more. Example: GPT4 powers advanced chatbots capable of maintaining context over long conversations.

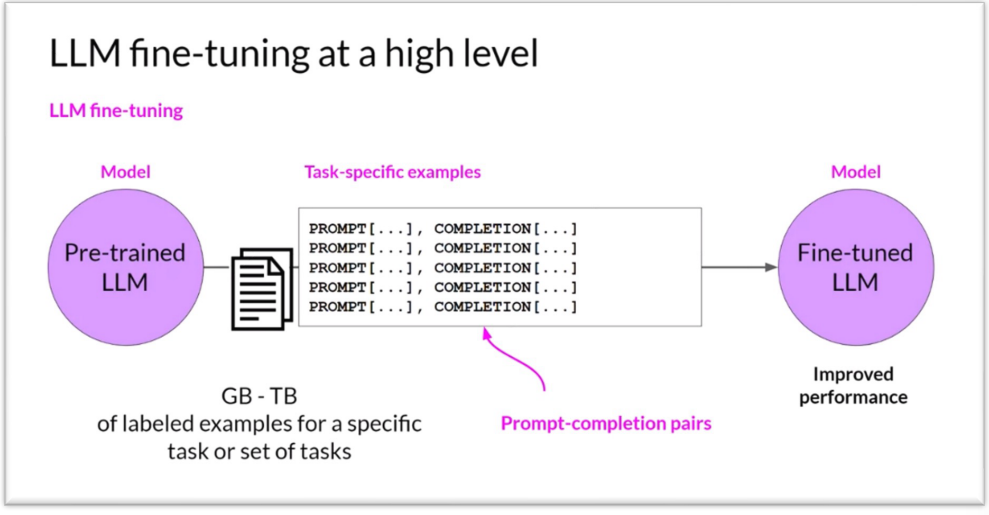
5. **Learning Method:** Most LLMs use a form of deep learning called transformers, which are effective at processing sequences of data. Example: GPT4's transformer architecture enables it to understand and predict language sequences efficiently.

6. **Adaptability:** LLMs can adapt to different styles, tones, and types of language tasks. Example: GPT4 can switch between writing styles, such as formal report writing and casual conversation.

7. **Continual Learning:** Many LLMs are designed to continually learn and improve over time with more data and user interaction. Example: GPT4's performance can improve as it is exposed to more diverse and extensive user interactions.

8. **Customization:** Some LLMs offer customization options for specific industry needs or user preferences. Example: GPT4 can be finetuned for specialized tasks like legal analysis or medical advice, although with limitations.





Prompt Engineering & Fine Tuning

Role
Task
Instructions
Context
Input

Prompt Engineering:

Definition: The practice of crafting inputs (prompts) to a generative AI model to elicit the best possible output.

Techniques: Includes using specific keywords, structured sentences, or including examples to guide the AI.
Purpose: To maximize the AI's understanding of the task at hand and to generate more accurate, relevant, or creative responses.

Adaptability: It requires an understanding of how the AI interprets different prompts and the ability to adjust them based on the desired outcome.

Skill Level: It can be utilized by both novices and experts; novices might use trialanderror, while experts might use more sophisticated methods based on their understanding of the model.

Model Dependence: Effectiveness can vary significantly across different AI models and versions.

FineTuning:

Definition: The process of adjusting a pretrained generative AI model to better suit specific tasks or data.

DataSpecific: Involves training the AI on a particular dataset to refine its responses to be more in line with that data's characteristics.

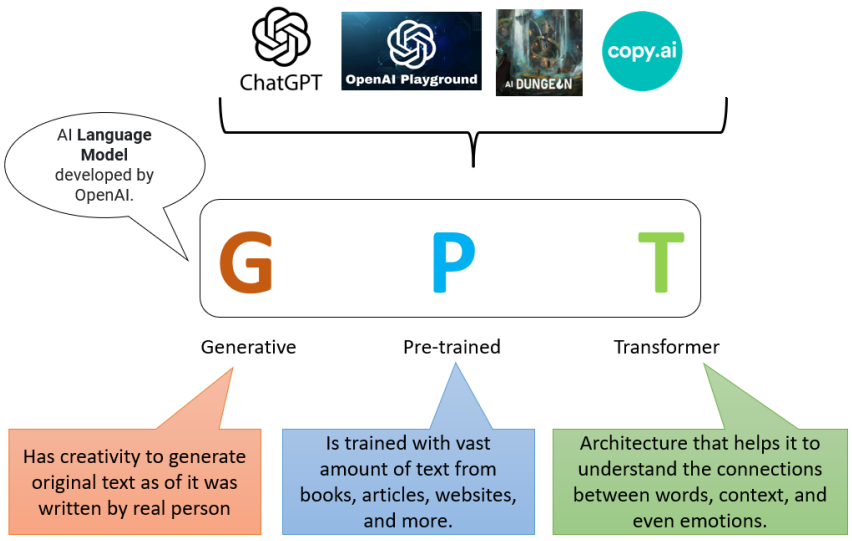
Cost and Resources: Generally requires more computational resources and technical knowledge than prompt engineering.
Customization: Enables the creation of a custom AI that can perform better on specialized tasks.

Continuous Process: Finetuning is not a onetime process; models may require periodic updates as they are exposed to new data or as requirements evolve.

Risk of Overfitting: There's a risk that the model may perform exceptionally well on the training data but fail to generalize to new, unseen prompts.

Both techniques are crucial for optimizing the performance of generative AI models in various applications,
They are often used in tandem: finetuning can provide a better baseline performance, while prompt engineering can optimize individual interactions.

What is GPT ?



1. **GPT**, which stands for "Generative Pretrained Transformer," is an advanced type of AI model developed by OpenAI for natural language processing tasks. Here are some key points about GPT:
2. **Type of AI Model:** GPT is a type of large language model (LLM) that uses deep learning techniques, specifically a transformer architecture, for understanding and generating human language.
3. **Training Method:** It is pretrained on a vast corpus of text data sourced from the internet, including websites, books, and other written material, allowing it to learn a wide range of language patterns and styles.
4. **Generative Capabilities:** GPT is designed to generate coherent and contextually relevant text based on the input it receives, making it useful for applications like content creation, conversation, and text completion.
5. **Versions and Evolution:** There have been several versions of GPT, with each new version (like GPT2, GPT3, GPT4, etc.) generally being larger and more capable than the last in terms of the amount of data it can process and the complexity of the tasks it can perform.
6. **Applications:** GPT is used in a variety of applications, including chatbots, writing assistants, language translation, and even in creative fields for generating art, music, and poetry.
7. **Humanlike Responses:** One of the notable features of GPT is its ability to produce responses that can closely mimic human writing styles, making its applications more natural and userfriendly.
8. **Ethical Considerations:** The deployment of GPT models raises ethical questions around topics like misinformation, privacy, and the potential impact on jobs in fields like writing and customer service.
9. **Accessibility and Use:** GPTpowered tools and services are increasingly accessible to businesses and individuals, enabling a wide range of users to leverage its capabilities for various purposes.

GPT MODELS COMPARISON CHART

Model	Size	Memory capacity	Accuracy	Input formats	Price
GPT-3	175B	1,500 words	<60%	Text, speech	$$
GPT-3.5	20B	8,000 words	<60%	Text, speech	$
GPT-4	>1T (?)	25,000-64,000 words	>80%	Text, speech, image	$$$

greenice

CHAT GPT-3 VS. CHAT GPT-4

CHAT GPT-3	CHAT GPT-4
Only takes text prompts	Take text & image prompts
Creative...sort of	More creative
Hallucinates a lot of facts & opinions	Still Hallucinates, but not as much :)
Barely passed the bar exam	Aced the bar exam
Takes a lot of steering & prompts for developers	More steerable in conversations & developer prompts

"GPT-4 IS MORE RELIABLE, CREATIVE AND ABLE TO HANDLE MUCH MORE NUANCED INSTRUCTIONS THAN GPT-3.5" - OPEN AI

GPT3 Vs GPT4

GPT3 and GPT4 are both iterations of OpenAI's Generative Pretrained Transformer (GPT) series, but there are key differences between them:

1.Model Size and Complexity:

- **GPT3:** Has 175 billion parameters, making it one of the largest language models at its time of release.
- **GPT4:** Is even larger than GPT3, with more parameters (the exact number has not been publicly disclosed), which enhances its understanding and generation capabilities.

2.Performance and Accuracy:

- **GPT3:** Sometimes struggles with complex reasoning tasks and can generate less accurate or relevant responses in certain contexts.
- **GPT4:** Shows improved performance in understanding context and nuance, leading to more accurate and contextually relevant outputs.

3.Training Data and Knowledge:

- **GPT3:** Trained on a vast corpus of text data available up until its training cutoff in 2020.
- **GPT4:** Benefits from an even larger and more diverse dataset, including more recent information, leading to a broader range of knowledge and understanding.

4. Capabilities in Understanding Context:

- **GPT3:** Exhibits a good understanding of context but can lose coherence over longer conversations or more complex queries.
- **GPT4:** Demonstrates a significantly improved ability to maintain context over longer interactions

5. Multimodal Abilities:

- **GPT3:** Primarily focused on text processing and generation.
- **GPT4:** It can understand and generate not just text but also images

6. Application and Usage:

- **GPT3:** Widely used across various industries for tasks like content creation, coding, customer service, and more.
- **GPT4:** Expands on these applications with improved performance, making it more effective and versatile for complex tasks.

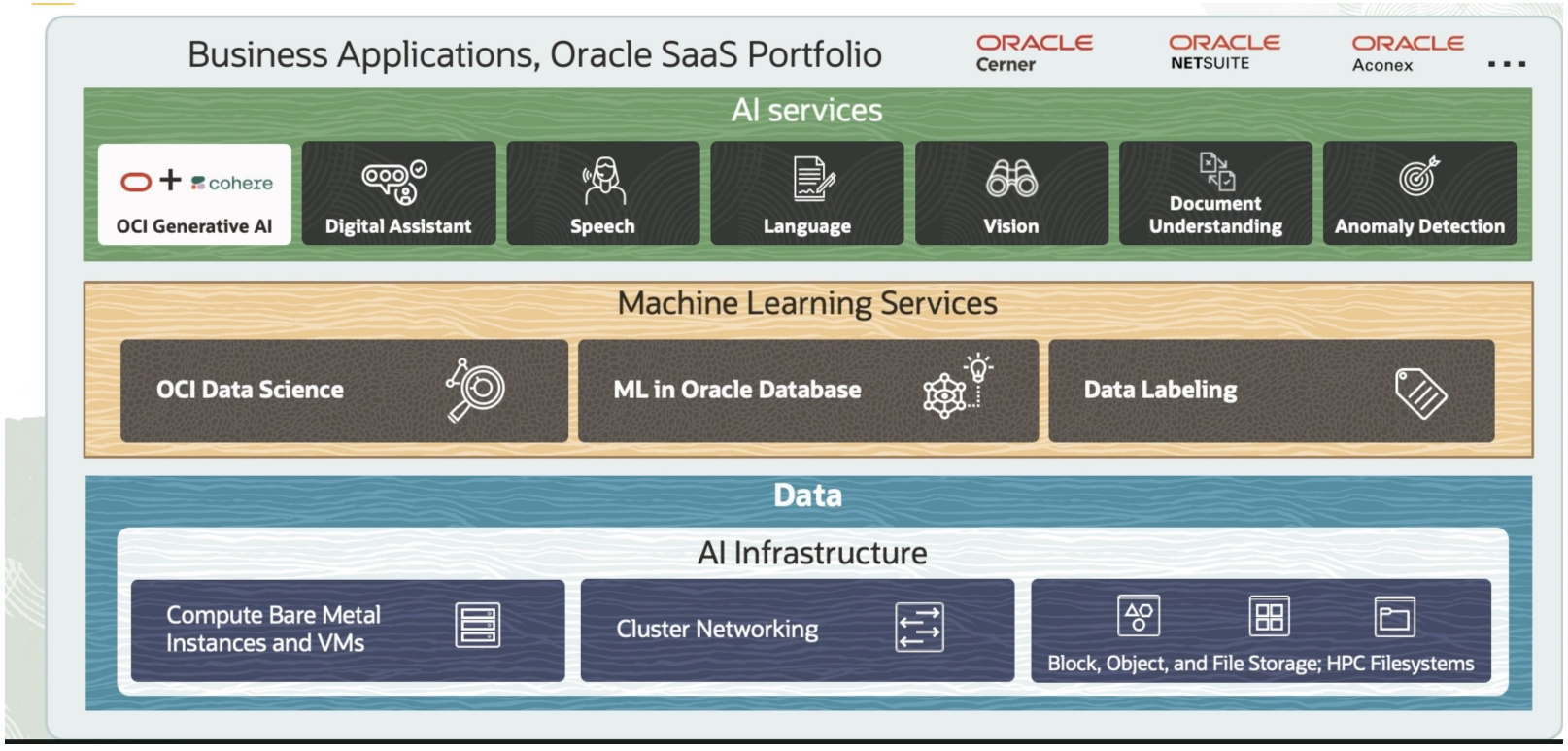
7. Error Rates and Reliability:

- **GPT3:** Though advanced, it has higher error rates in certain complex tasks.
- **GPT4:** Demonstrates a lower error rate and higher reliability in a wide range of tasks.

AI Infrastructure

CPU Vs GPU

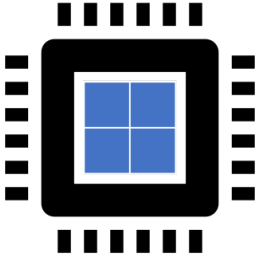
AIINFRASTRUCTURE

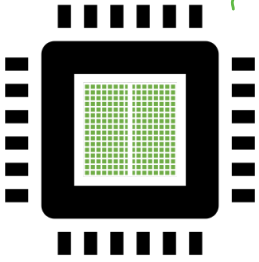


1. CPU vs GPU: Basics

Feature	CPU	GPU
Cores	Few (4–64)	Many (hundreds–thousands)
Strength	Sequential processing	Parallel processing
Task type	General-purpose	Matrix-heavy / vectorized tasks
Memory	Smaller, fast cache	Larger VRAM (video RAM)

parallel processing





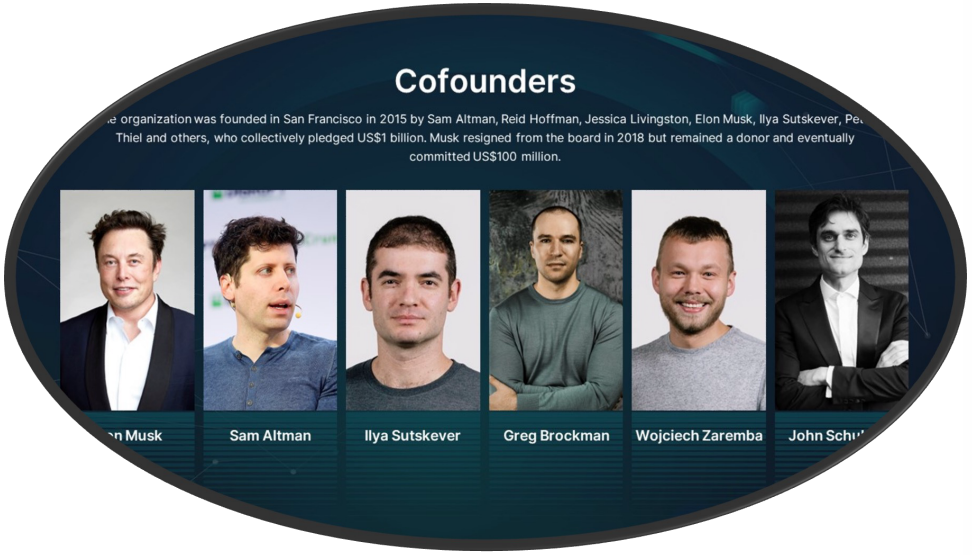
CPU	GPU
Central Processing Unit	Graphics Processing Unit
4-8 Cores	100s or 1000s of Cores
Low Latency	High Throughput
Good for Serial Processing	Good for Parallel Processing
Quickly Process Tasks That Require Interactivity	Breaks Jobs Into Separate Tasks To Process Simultaneously
Traditional Programming Are Written For CPU Sequential Execution	Requires Additional Software To Convert CPU Functions to GPU Functions for Parallel Execution

What is a GPU ?

Speed up parallel task computation

1. A **GPU**, or Graphics Processing Unit, is a specialized processor primarily designed to accelerate graphics rendering.
2. **Highly Parallel Structure**: GPUs have a parallel processing architecture, which allows them to process many computations simultaneously
3. **Graphics Rendering**: Originally, GPUs were designed to render graphics for applications such as video games and 3D animations
4. **Machine Learning and AI**: In the field of AI and machine learning, GPUs are instrumental for training complex neural networks due to their ability to handle multiple operations concurrently.
5. **Energy Efficiency**: GPUs are more energyefficient than traditional CPUs for parallel processing tasks, offering a better performanceperwatt ratio
6. **Types of GPUs**: There are two main types of GPUs: integrated GPUs, which are built into the same chip as the CPU and offer basic graphics processing, and dedicated (or discrete) GPUs, which are separate cards with their own memory and processing power, offering superior performance.
7. **VR and AR Applications**: GPUs are essential for powering virtual reality (VR) and augmented reality (AR) applications, providing the necessary speed and performance to create immersive experiences.
8. **Evolution and Innovation**: The technology behind GPUs continues to evolve rapidly, with constant innovations leading to faster, more efficient, and more powerful GPUs suitable for a wide range of applications.

OpenAI / ChatGPT / APIs



What is OpenAI ?

Research Organization: OpenAI is an AI research lab that focuses on developing and promoting friendly AI in a way that benefits humanity as a whole.

Founded in 2015: OpenAI was established in December 2015 by Elon Musk, Sam Altman, Greg Brockman, Ilya Sutskever, Wojciech Zaremba, and others

Advanced AI Models: Developing some of the most advanced AI models, including the Generative Pretrained Transformer (GPT) series, with the latest iterations being GPT3 and GPT4.

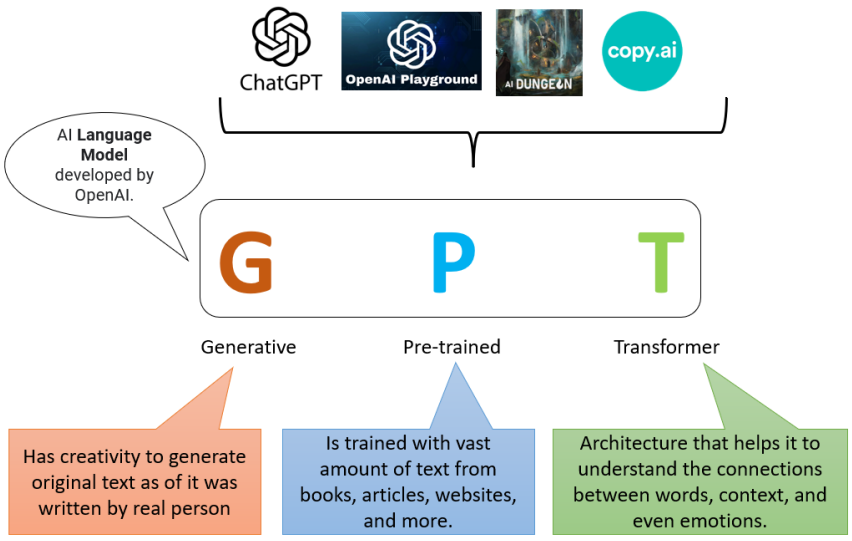
Ethics and Safety: A significant focus for OpenAI is ensuring the ethical use of AI and addressing potential safety risks associated with powerful AI systems.

Open Collaboration: Initially, OpenAI started with a commitment to openaccess research, sharing its findings and technologies. However, it has since adopted a more controlled release approach to mitigate potential risks.

Commercial Products: OpenAI has developed several commercial products, such as the API that provides access to models like GPT3 for a variety of textbased tasks, including conversation, summarization, translation, and more.

Partnerships and Licensing: OpenAI has entered into partnerships and licensing agreements, notably with Microsoft, which provides significant Azure cloud computing resources necessary for training and deploying AI models.

What is GPT ?



1. **GPT**, which stands for "Generative Pretrained Transformer," is an advanced type of AI model developed by OpenAI for natural language processing tasks. Here are some key points about GPT:
2. **Type of AI Model:** GPT is a type of large language model (LLM) that uses deep learning techniques, specifically a transformer architecture, for understanding and generating human language.
3. **Training Method:** It is pretrained on a vast corpus of text data sourced from the internet, including websites, books, and other written material, allowing it to learn a wide range of language patterns and styles.
4. **Generative Capabilities:** GPT is designed to generate coherent and contextually relevant text based on the input it receives, making it useful for applications like content creation, conversation, and text completion.
5. **Versions and Evolution:** There have been several versions of GPT, with each new version (like GPT2, GPT3, GPT4, etc.) generally being larger and more capable than the last in terms of the amount of data it can process and the complexity of the tasks it can perform.
6. **Applications:** GPT is used in a variety of applications, including chatbots, writing assistants, language translation, and even in creative fields for generating art, music, and poetry.
7. **Humanlike Responses:** One of the notable features of GPT is its ability to produce responses that can closely mimic human writing styles, making its applications more natural and userfriendly.
8. **Ethical Considerations:** The deployment of GPT models raises ethical questions around topics like misinformation, privacy, and the potential impact on jobs in fields like writing and customer service.
9. **Accessibility and Use:** GPTpowered tools and services are increasingly accessible to businesses and individuals, enabling a wide range of users to leverage its capabilities for various purposes.

Time to Reach 100M Users

Months to get to 100 million global Monthly Active Users

Source: UBS / Yahoo Finance

@EconomyApp

APP ECONOMY **INSIGHTS**

Models

Overview

The OpenAI API is powered by a diverse set of models with different capabilities and price points. You can also make customizations to our models for your specific use case with fine-tuning.

MODEL	DESCRIPTION
GPT-4 Turbo and GPT-4	A set of models that improve on GPT-3.5 and can understand as well as generate natural language or code
GPT-3.5 Turbo	A set of models that improve on GPT-3.5 and can understand as well as generate natural language or code
DALL·E	A model that can generate and edit images given a natural language prompt
TTS	A set of models that can convert text into natural sounding spoken audio
Whisper	A model that can convert audio into text
Embeddings	A set of models that can convert text into a numerical form
Moderation	A fine-tuned model that can detect whether text may be sensitive or unsafe
GPT base	A set of models without instruction following that can understand as well as generate natural language or code
Deprecated	A full list of models that have been deprecated along with the suggested replacement

- GPT 3, 4, 5
- DALL E
  images
- Embeddings

What is ChatGPT

Imp.

GPT-4 Variants by Approximate Size

Model	Params (Approx.)	Notes
GPT-4 Mini / GPT-4o 12B	~12B	Smallest, faster, cheaper, optimized ("Omni")
GPT-4o (default)	12B (similar or slightly larger)	Omni variant, optimized for speed and versatility
GPT-4 Turbo	Likely larger than GPT-4o but smaller than GPT-4	Optimized for faster inference, cost-efficient
GPT-4 (default)	100B+ (est.)	Full-sized, high-quality reasoning, slower

AI and Machine Learning: ChatGPT is built on the GPT (Generative Pretrained Transformer) architecture, which is a type of artificial intelligence model designed to generate text.

Language Understanding: It has been trained on a diverse range of internet text, enabling it to understand context, answer questions, write essays, and even create poetry or code.

Conversational Interface: ChatGPT is designed to engage in conversations, providing responses that can simulate a humanlike interaction.

Learning Capability: While it can't learn or remember information from individual user interactions in realtime, its design allows for continual updates and training by OpenAI to improve its performance and capabilities.

Versatile Applications: It can be used in various applications such as customer service bots, tutoring systems, content creation aids

Safety and Ethics: OpenAI has implemented safeguards to prevent ChatGPT from generating inappropriate or harmful content, though it's not foolproof.

Customization and Integration: Developers can integrate ChatGPT into their own applications through OpenAI's API, allowing for a wide range of custom uses and functionalities tailored to specific needs.

Continual Development: ChatGPT is part of an ongoing research and development effort, with improvements and new versions being released as AI technology advances.

GPT MODELS COMPARISON CHART

Model	Size	Memory capacity	Accuracy	Input formats	Price
GPT-3	175B	1,500 words	<60%	Text, speech	$$
GPT-3.5	20B	8,000 words	<60%	Text, speech	$
GPT-4	>1T (?)	25,000-64,000 words	>80%	Text, speech, image	$$$

greenice

CHAT GPT-3 VS. CHAT GPT-4

CHAT GPT-3	CHAT GPT-4
Only takes text prompts	Take text & image prompts
Creative...sort of	More creative
Hallucinates a lot of facts & opinions	Still Hallucinates, but not as much :)
Barely passed the bar exam	Aced the bar exam
Takes a lot of steering & prompts for developers	More steerable in conversations & developer prompts

"GPT-4 IS MORE RELIABLE, CREATIVE AND ABLE TO HANDLE MUCH MORE NUANCED INSTRUCTIONS THAN GPT-3.5" - OPEN AI

GPT3 Vs GPT4

GPT3 and GPT4 are both iterations of OpenAI's Generative Pretrained Transformer (GPT) series, but there are key differences between them:

1.Model Size and Complexity:

- **GPT3:** Has 175 billion parameters, making it one of the largest language models at its time of release.
- **GPT4:** Is even larger than GPT3, with more parameters (the exact number has not been publicly disclosed), which enhances its understanding and generation capabilities.

2.Performance and Accuracy:

- **GPT3:** Sometimes struggles with complex reasoning tasks and can generate less accurate or relevant responses in certain contexts.
- **GPT4:** Shows improved performance in understanding context and nuance, leading to more accurate and contextually relevant outputs.

3.Training Data and Knowledge:

- **GPT3:** Trained on a vast corpus of text data available up until its training cutoff in 2020.
- **GPT4:** Benefits from an even larger and more diverse dataset, including more recent information, leading to a broader range of knowledge and understanding.

4. Capabilities in Understanding Context:

- **GPT3:** Exhibits a good understanding of context but can lose coherence over longer conversations or more complex queries.
- **GPT4:** Demonstrates a significantly improved ability to maintain context over longer interactions

5. Multimodal Abilities:

- **GPT3:** Primarily focused on text processing and generation.
- **GPT4:** It can understand and generate not just text but also images

6. Application and Usage:

- **GPT3:** Widely used across various industries for tasks like content creation, coding, customer service, and more.
- **GPT4:** Expands on these applications with improved performance, making it more effective and versatile for complex tasks.

7. Error Rates and Reliability:

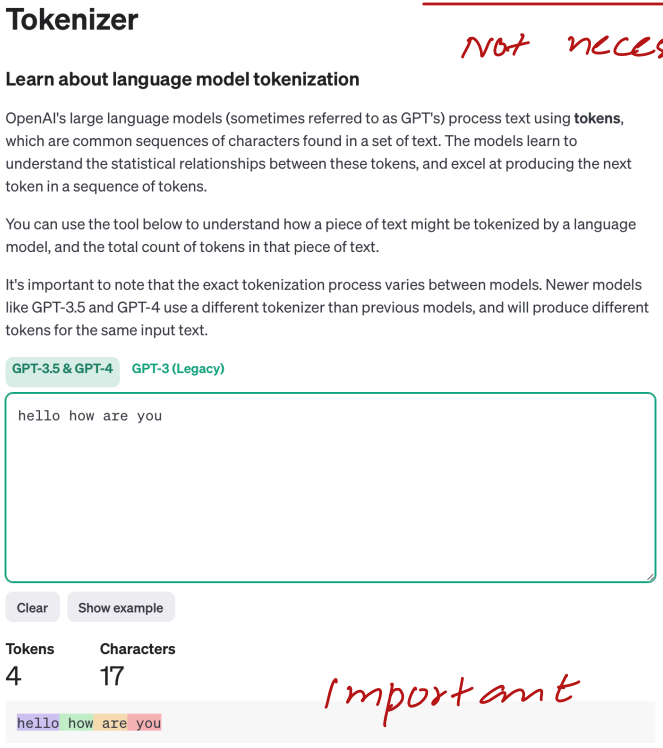
- **GPT3:** Though advanced, it has higher error rates in certain complex tasks.
- **GPT4:** Demonstrates a lower error rate and higher reliability in a wide range of tasks.

Playing

2 Tokens ; play + ing

not necessarily words

What are Tokens ?



Important

Input

Output

Token } Separate pricing

Basic Units of Text: In NLP, a token is the basic unit of text. It can be a word, a part of a word (like a prefix or suffix), or even punctuation. Tokens are the building blocks that models like ChatGPT analyze and generate.

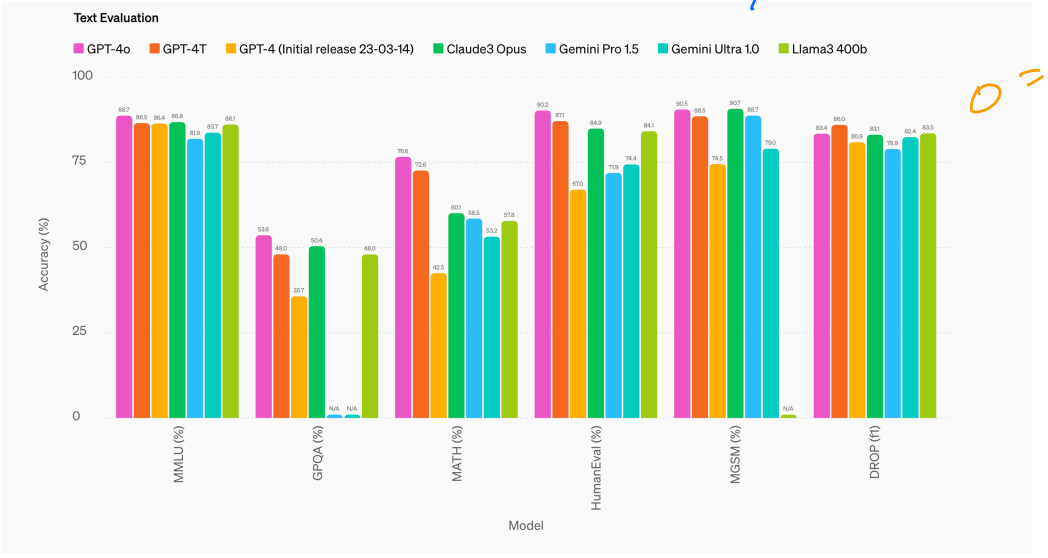
Tokenization: This is the process of breaking down text into its constituent tokens.

Sequence of Tokens: When you input a sentence, ChatGPT tokenizes it, processes the tokens to understand the context and generate a response, and then converts the output tokens back into humanreadable text.

Vocabulary Limit: Language models have a fixed vocabulary size, meaning they can only recognize a certain number of unique tokens.

Efficiency in Processing: Using tokens allows language models to efficiently process large amounts of text by breaking down complex structures into manageable pieces.

GPT 4 = 175B params



O = Omni

GPT-4o

only 12B

GPT-4O is an advanced version of the GPT-4 model developed by OpenAI. The "O" stands for "Omni," highlighting its all-encompassing capabilities.

Release Date : May 2024

Multimodal Input (Omni): GPT-4O can process and understand images in addition to text, unlike GPT-3 which only handles text.

Context Length: GPT-4O can maintain context over longer conversations compared to GPT-3.

Higher Accuracy in Response Generation: GPT-4O provides more accurate and relevant responses, reducing the chances of generating incorrect or nonsensical text.

Improved Efficiency and Speed: GPT-4O is optimized to deliver responses faster than GPT-3, making it more suitable for real-time applications.

Enhanced Adaptability: GPT-4O can better adapt to different styles and tones based on user input, providing more personalized interactions.

Human like Response Time : It can respond to audio inputs in as little as 232 milliseconds, with an average of 320 milliseconds

Pricing : It's considered to be almost 50% cheaper as compared to GPT-4 Models

Model	Pricing
gpt-4o	US$5.00 / 1M input tokens US$15.00 / 1M output tokens

→ Faster (less accuracy)

half cost

input output } Both

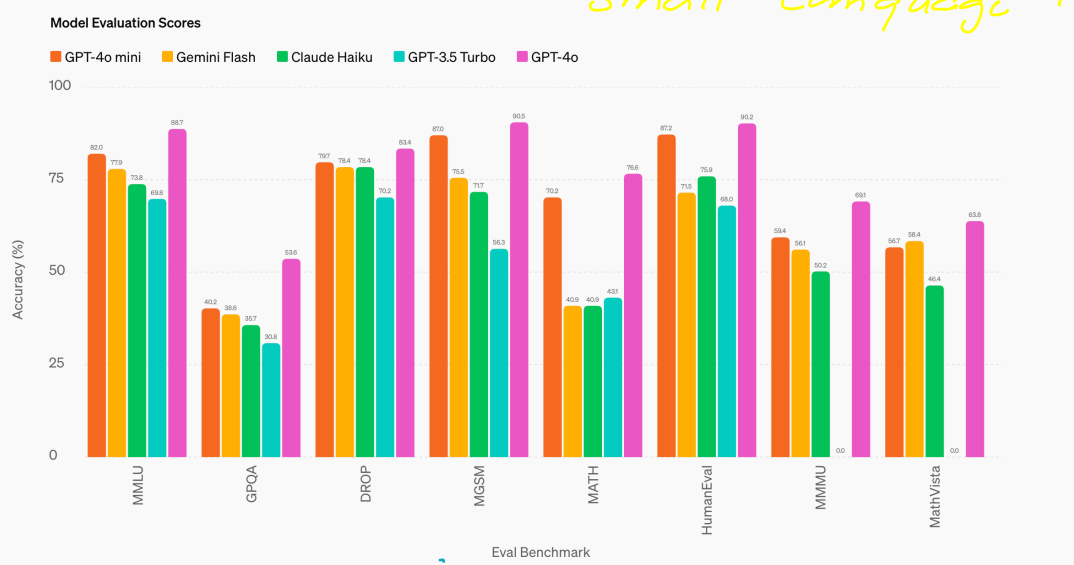
Model	Input	Output
gpt-4-turbo	US$10.00 / 1M tokens	US$30.00 / 1M tokens
gpt-4-turbo-2024-04-09	US$10.00 / 1M tokens	US$30.00 / 1M tokens

Small Language Models

GPT-4o Mini

SLM

Small – Compact



To stay in competition

GPT-4o Mini is a compact, efficient version of the larger GPT-4 model designed for deployment in resource-constrained environments.

Release Date : July 2024

Compact Size
GPT-4o Mini is designed to run efficiently on hardware with limited computational resources.

Faster Inference
Provides faster response times compared to full-sized GPT models, making it ideal for time-sensitive tasks.

Energy Efficiency
Prioritizes low power consumption, unlike larger models that require significant energy to function.

Edge Deployment
Designed to function independently of cloud infrastructure, unlike traditional GPT models that rely heavily on server-based computations.

Customizable and Scalable
Offers greater flexibility in adapting to particular use cases, unlike one-size-fits-all larger GPT models.

Cost-Effective
Lower operational costs due to reduced resource requirements.

Simplified Training
Difference: Easier and faster to train compared to large-scale GPT models that need extensive datasets and powerful GPUs.

Model	Pricing
gpt-4o-mini	US$0.150 / 1M input tokens US$0.600 / 1M output tokens

Model	Pricing
gpt-4o	US$5.00 / 1M input tokens US$15.00 / 1M output tokens

Model	Input	Output
gpt-4-turbo	US$10.00 / 1M tokens	US$30.00 / 1M tokens
gpt-4-turbo-2024-04-09	US$10.00 / 1M tokens	US$30.00 / 1M tokens

Open AI Models

Use Case / Modality	Model Name(s) / Family	Example Client Code	Input → Output
Chat / Text Generation	gpt-4, gpt-4o, gpt-4o-mini, gpt-3.5-turbo	client.chat.completions.create()	Text → Text
Image Generation	dall-e-2, dall-e-3, gpt-image-1	client.images.generate()	Text → Image
Speech-to-Text (ASR)	whisper-1	client.audio.transcriptions.create()	Audio → Text
Text-to-Speech (TTS)	gpt-4o-mini-tts	client.audio.speech.create()	Text → Audio
Video Generation	sora (limited access)	*(not public in API yet)*	Text → Video
Embeddings	text-embedding-ada-002, text-embedding-3-small/large	client.embeddings.create()	Text → Vector

All models are based on

- Reinforment learning + Human Response

ph.D Level O_1 = Reasoning model (more compute)

- It uses chain of thought

Q// why is O_1 costly?

because of Input / Reasoning (extra) / output Tokens

Chain of Thought (CoT)

- The model writes **step-by-step reasoning** before giving the answer.

Q: If a bat and ball cost $1.10 together, and the bat costs $1 more than the ball, how much does the ball cost?

A (CoT):

1. Let the price of the ball be x.
2. Then the bat costs x + $1.
3. Total cost = x + (x + 1) = 1.10
4. Solve: 2x + 1 = 1.10 → 2x = 0.10 → x = 0.05

   Answer: $0.05 ✅

Prompting Types

- Zero shot
- one shot
- few shot
- Chain of thoughts

Imp. OpenAI started as Non profit but now it is private entity.

Azure + Open AI

What is Azure OpenAI

Azure OpenAI refers to the collaboration between Microsoft Azure (a cloud computing platform), and OpenAI (an artificial intelligence research organization).

Integration of OpenAI Models: Provides access to OpenAI's AI models like GPT, Codex, and DALLE on Microsoft Azure's cloud platform.

Enterprise Applications: Designed for business use, helping companies integrate AI into various functions such as customer support and content generation.

Scalable Infrastructure: Utilizes Azure's cloud infrastructure for scalable AI model deployment and management.

Security and Compliance: Focuses on high security and adherence to regulatory standards to protect user data.

DeveloperFriendly: Offers tools and APIs for developers to easily implement and customize AI solutions.

Customization Options: Allows users to tailor AI models to better fit their specific requirements.

Global Reach: Available worldwide through Azure's extensive global infrastructure.

Collaboration and Innovation: Supports collaborative AI development and innovation within the Azure ecosystem.

Imp. ChatGpT & OpenAI subscriptions are different

Azure + Open AI

- collaboration b/w Azure & Open AI
- Secure, scalable
- It is enterprise version of Open AI

(imp.) Not every model is available in every region because of load balancing.

Limit Type	Typical Numbers / Metrics	Notes
Request Rate / Throttling	**20–60 requests per minute** per deployment (depends on SKU)	Exceeding → 429 errors
Concurrent Requests	**5–10 concurrent requests** per deployment by default	Exceeding → throttling; can request increase
Token Limits	**GPT-4:** 8k or 32k tokens, **GPT-3.5:** 4k tokens	Input + output combined
Deployment Limits	Default **~10 deployments** per subscription	Can request more if needed

💡 **Interview tip:** Mention **"Azure OpenAI has request rate limits (20–60/min), concurrent request limits (5–10), and token limits (4k–32k depending on model)"**—this shows practical understanding without memorizing every SKU.

Azure Open AI pricing:

GPT-4.1 series

GPT-4.1 series is a highly advanced general-purpose model with extensive world knowledge and an enhanced ability to understand user intent, making it particularly adept at creative tasks and agentic planning. The series features a 1 million token context window and has a knowledge cutoff of June 2024.

4

Model	Pricing (1M Tokens)	Pricing with Batch API (1M Tokens)
GPT-4.1-2025-04-14 Global	Input: $2 Cached Input: $0.50 Output: $8	Input: $1 Output: $4

Model	Pricing
o3-deep research Global	Input: $10 Cached Input: $2.50 Output: $40

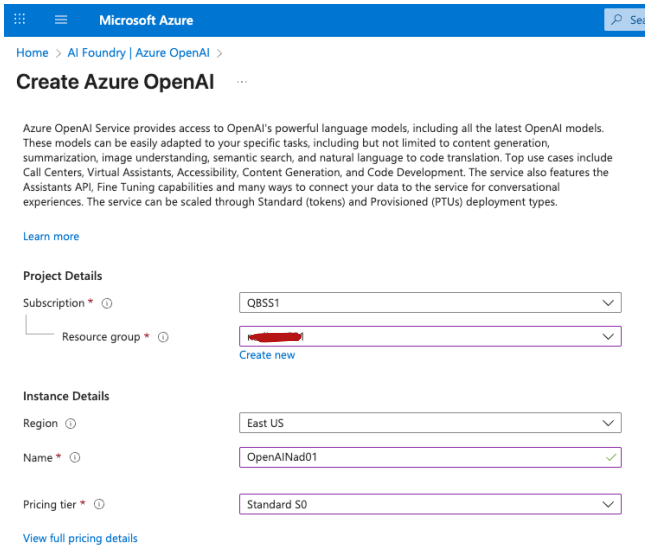
o3

GPT-5 series

5

Model	Pricing (1M Tokens)
GPT-5 2025-08-07 Global	Input: $1.25 Cached Input: $0.13 Output: $10

Sign up to Azure Open AI



why Open AI + Azure

- Security
- Scalability
- Infrastructure

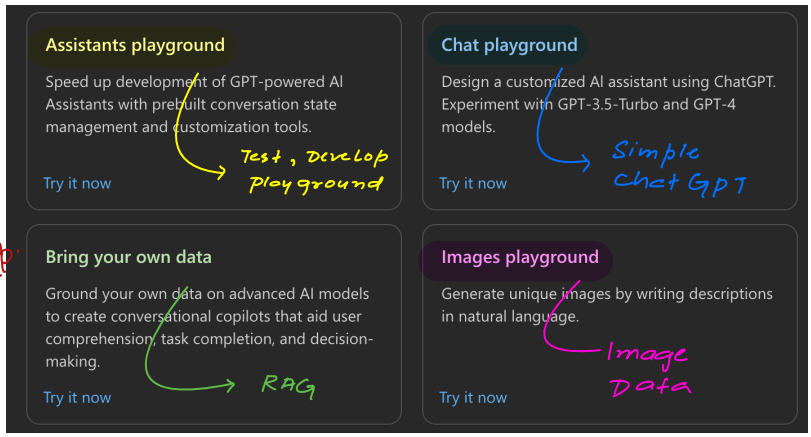
Imp. Old = Azure AI Studio
New = Azure AI Foundry
} Azure AI Studio is now AI Foundry

Links

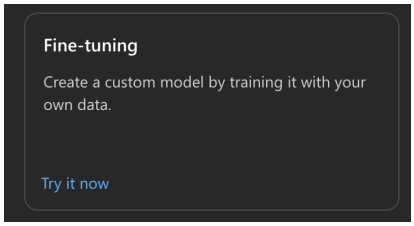
Azure portal = portal.azure

Azure AI = ai.azure

Azure AI Foundry → previously AI Studio.



Imp.



↳ Base model
|
Then Finetune

AZURE AI MODELS

Chat	GPT
Code	Codex
Images	Dall E
Video	SORA
Embedding	Text-Embedding

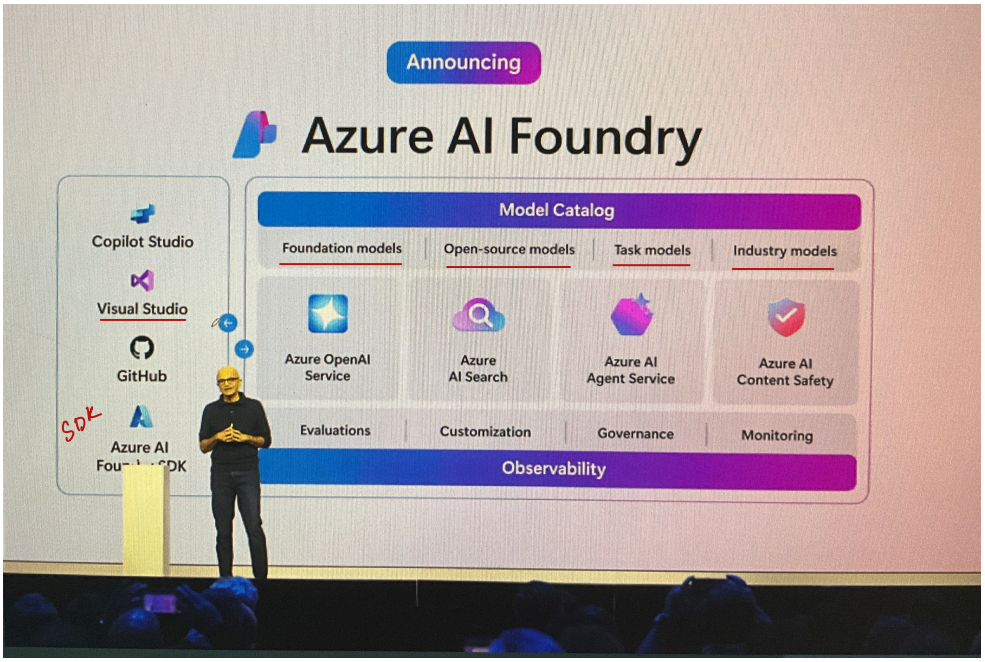
Foundry: Factory where metal is melted and Reshaped

Q/ What is Azure AI Foundry?

Azure AI Foundry is Microsoft's all-in-one platform for building, testing, and deploying AI apps and agents. It combines:

- A **catalog of AI models** (from Microsoft, OpenAI, Meta, etc.) Imp. Not only meta.
- Tools for **fine-tuning, RAG, and agent creation**
- Built-in **governance, security, and monitoring**
- Support for **enterprise use** (projects, access control, compliance)

It's designed to help companies go from prototyping to production faster with a unified, scalable AI development environment.



Q/ Hubs vs projects?

- **Hub**: A **central workspace** for managing **multiple projects**. It's where organizational settings, resources, billing, and policies are set.
- **Project**: An **isolated workspace** within a hub for building and managing a specific AI solution (e.g., a chatbot, RAG app). Each project has its own data, models, configs, etc.

👉 Think of a **Hub** as the organization or team-level container, and **Projects** as individual AI use cases inside it.

Hub = workspace

project = individual AI solution

Imp. = Azure AI Foundry ; 1. All models

2. Azure open AI (only open AI)

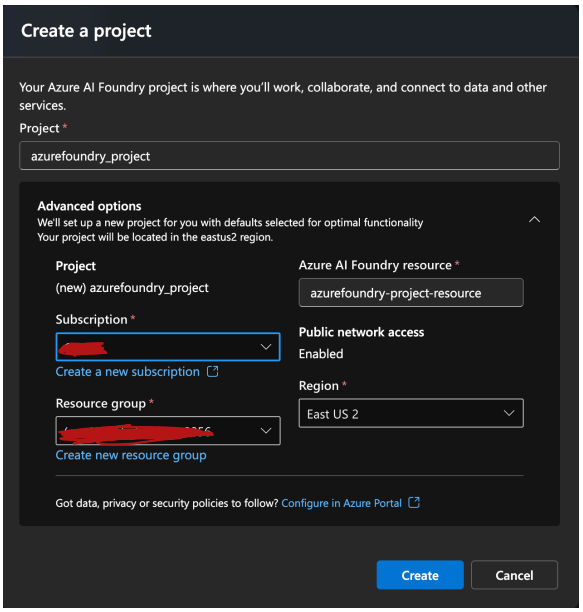
OPTIONS

1. Azure open AI with project - only Open AI model
2. Azure open AI without project - other models, SDK

Imp. = If you are solely interested in Azure open AI then there is no need to create a project.

Creating Project

Go to Azure AI Services



Q. How to compare models?

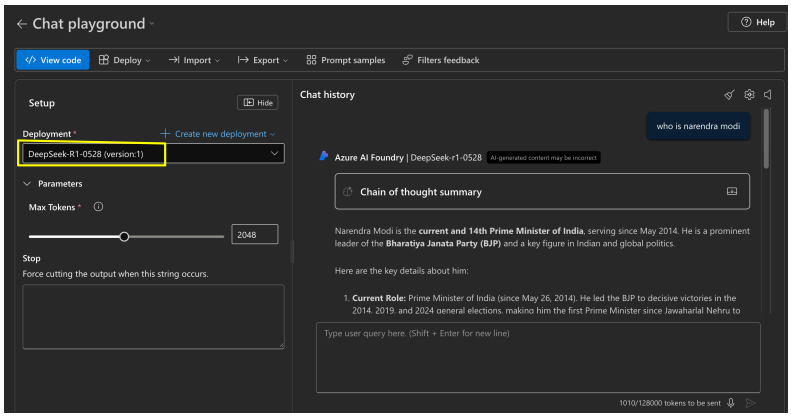
Model catalog

↓

Compare models

Deploying models

Model catalog ⟶ Choose model → Open in playground



← Deploying models

Imp. O1 are Research Models = More complex / Costly

How to Use

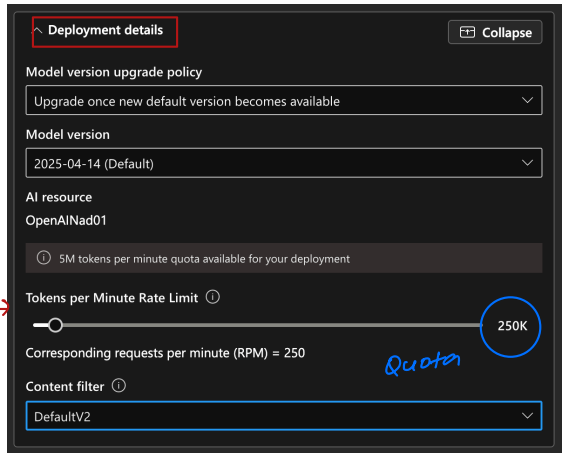
In order to use any of these we need to create Deployments

AZURE AI SERVICES

Service	One-liner
Chat	No-code playground to prototype prompts and get responses via Chat Completions before moving to code. learn.microsoft
Assistants	Build configurable copilots that use models plus tools, threads, and files to run tasks. learn.microsoft
Video (Preview)	Playground/APIs to work with multimodal models that understand or generate video content. learn.microsoft
Audio (Preview)	Use speech models (e.g., Whisper family) for transcription, translation, or audio tasks. learn.microsoft
Images	Try image generation models and export code for image workflows. microsoft
Fine-tuning	Train custom versions of base models on domain data and deploy them securely. learn.microsoft
Azure OpenAI Evaluation (Preview)	Evaluate prompts and systems with built-in metrics and datasets to compare quality and safety. learn.microsoft
Stored completions (Preview)	Persist chat/completion histories as datasets to reuse for evals or fine-tuning. learn.microsoft
Batch jobs	Submit large, asynchronous request batches for cost-efficient high-volume processing. learn.microsoft
Monitoring	Track requests, tokens, PTU usage, and fine-tune metrics with Azure Monitor dashboards. learn.microsoft
Deployments	Manage model deployments tied to an endpoint for use in apps and playgrounds. learn.microsoft
Quota	View and manage capacity limits, rate limits, and PTU allocations for the resource. learn.microsoft
Guardrails + Controls	Configure safety filters, content policies, and governance settings for responsible use. learn.microsoft
Risks + alerts (Preview)	Get proactive risk insights and alerts across usage, safety, and operations. learn.microsoft
Data files	Upload and manage files for assistants, fine-tuning, evals, and tool use. learn.microsoft
Assistant	Workspace to configure an individual Assistant's instructions, tools, and files. learn.microsoft
Vector stores (Preview)	Create and attach retrieval indexes for RAG so assistants can ground answers on documents. learn.microsoft

→ Chat Playground

Deploy ⟶ Choose model ⟶



Imp. If Token limit = 250K

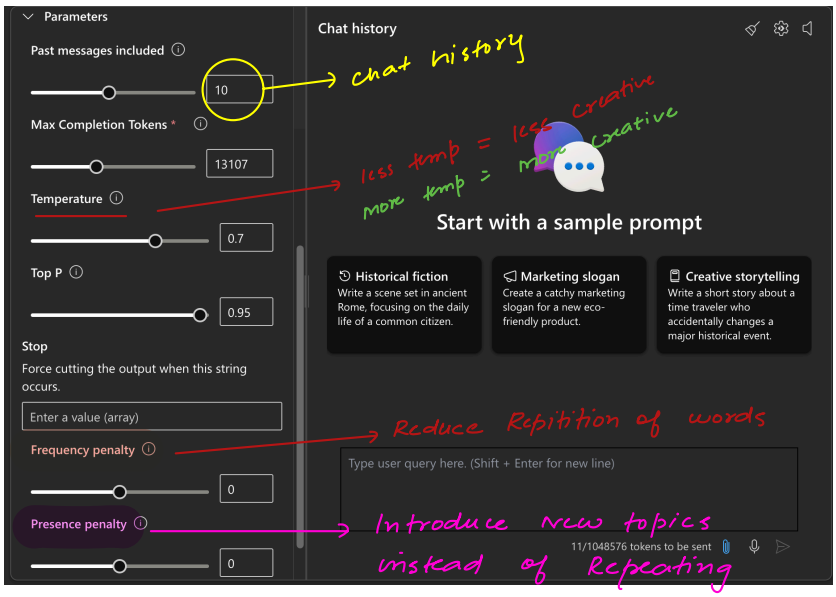
e.g., 2 Deployments = 125K/each

Imp. we dont need to call API here, no code is required

PARAMETERS

Parameter	Meaning (One-liner)
Past Messages	Number of previous conversation turns the model considers for context.
Max Token Completion	Maximum length of the model's response (in tokens).
Temperature	Controls randomness; higher = more creative, lower = more focused.
Top P ↘ 90% Probability	Nucleus sampling; limits output to top tokens whose probabilities sum to *P*.
Top K ↘ K=2	Limits output choices to the *K* most likely tokens at each step.
Presence Penalty	Encourages the model to introduce new topics instead of repeating.
Frequency Penalty	Reduces repetition of words/tokens already used.

Parameter	Explanation	Example
Top-P (Nucleus Sampling)	The model considers the smallest set of tokens whose **cumulative probability ≥ P** (e.g., P=0.9 → choose from tokens covering 90% of the probability mass).	If possible next words have probs {"cat" 0.5, "dog" 0.3, "fish" 0.1, "car" 0.1}, with P=0.9, it will only consider {cat, dog, fish}.
Top-K	The model only considers the **K most likely tokens** and ignores the rest, no matter their probability mass.	With K=2, only {"cat", "dog"} are considered even if "fish" had 0.1 probability.



Q// what is use of Azure + OpenAI?

- Enhanced Integration & Enterprise level security.

OpenAI API CallsVs Azure OpenAI

All models are based on R.L & Human Feedback

OpenAI | Azure OpenAI

Python Copy

```python
import os
from openai import OpenAI

client = OpenAI(
    api_key=os.getenv("OPENAI_API_KEY")
)
```

open AI

Azure OpenAI

Python Copy

```python
import os
from openai import AzureOpenAI

client = AzureOpenAI(
    api_key=os.getenv("AZURE_OPENAI_API_KEY"),
    api_version="2023-12-01-preview",
    azure_endpoint=os.getenv("AZURE_OPENAI_ENDPOINT")
)
```

Azure OpenAI

Store API key in .env file so it is not accessible

So 2 additions are;

api_version

azure Endpoint

OpenAI | Azure OpenAI

Python Copy

```python
completion = client.completions.create(
    model="gpt-3.5-turbo-instruct",
    prompt="<prompt>"
)

chat_completion = client.chat.completions.create(
    model="gpt-4",
    messages="<messages>"
)

embedding = client.embeddings.create(
    model="text-embedding-ada-002",
    input="<input>"
)
```

Python Copy

```python
completion = client.completions.create(
    model="gpt-35-turbo-instruct", # This must match the custom deployment name you chose for your model.
    prompt="<prompt>"
)

chat_completion = client.chat.completions.create(
    model="gpt-35-turbo", # model = "deployment_name".
    messages="<messages>"
)

embedding = client.embeddings.create(
    model="text-embedding-ada-002", # model = "deployment_name".
    input="<input>"
)
```

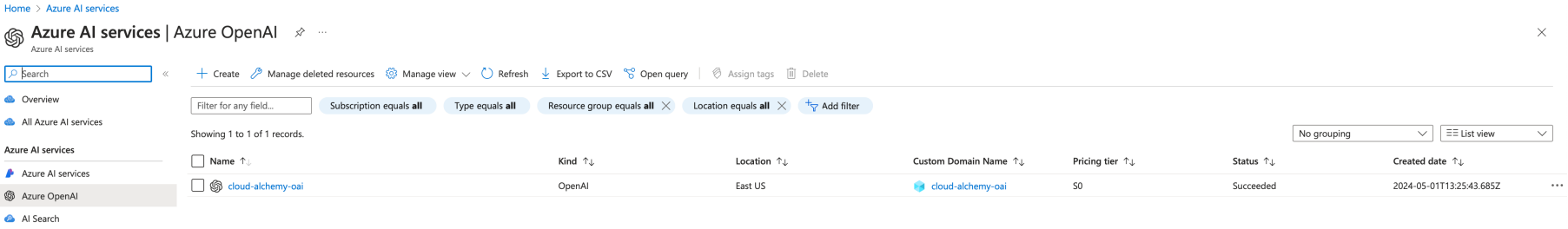
↳ Deployment / Not model

Understanding Azure OpenAI API Calls

azure_endpoint = "https://cloudalchemyoai.openai.azure.com/",
api_key=os.environ.get("OPENAI_API_KEY"),
api_version="20240215preview”

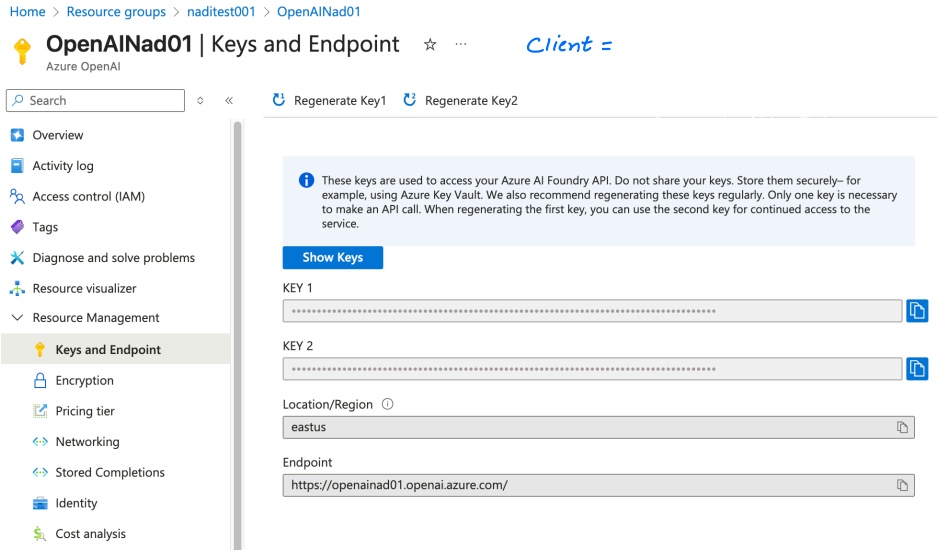
```
completion = client.chat.completions.create(
model="air_text_lab", # model = "deployment_name"
messages = message_text,
temperature=0.7,
max_tokens=800,
top_p=0.95,
frequency_penalty=0,
presence_penalty=0,
stop=None
)
```

```python
#Note: The openai-python library support for Azure OpenAI is in preview.
#Note: This code sample requires OpenAI Python library version 1.0.0 or higher.
import os
from openai import AzureOpenAI

client = AzureOpenAI(
  azure_endpoint = "https://cloud-alchemy-oai.openai.azure.com/",
  api_key=os.getenv("AZURE_OPENAI_KEY"),
  api_version="2024-02-15-preview"
)

message_text = [{"role":"system","content":"You are an AI assistant that helps people find information."},{"role":"user","content":"Who is prime minister of India"},
{"role":"assistant","content":"As of my last update, the Prime Minister of India is Narendra Modi. He has been in office since May 2014. Please note that political positions can change, so it's always a good idea to verify with the latest sources."}]

completion = client.chat.completions.create(
  model="air_text_lab", # model = "deployment_name"
  messages = message_text,
  temperature=0.7,
  max_tokens=800,
  top_p=0.95,
  frequency_penalty=0,
  presence_penalty=0,
  stop=None
)
```



Azure Endpoint URL & API Keys?

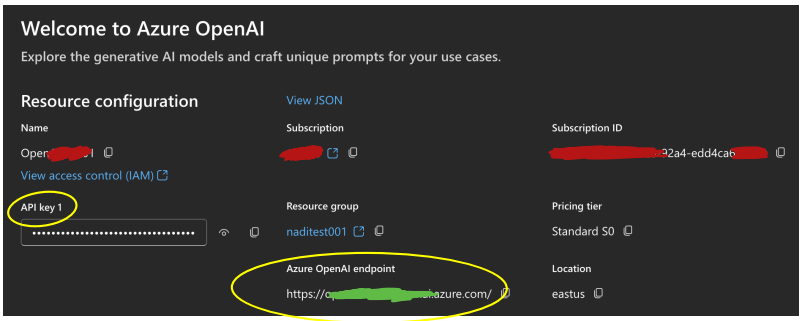
These are very important for making API calls



Step I; Azure Open AI API Calling

Unlike OpenAI call, we need

- Azure AI Api key
- Api version
- Client ID



Q11 How to get code;

1. Setup Key, endpoint, in .env file

→ Deploy the chat model (use it in client)

Imp Code = Chat play Ground → View Code

```python
import os
from dotenv import load_dotenv
load_dotenv()
from openai import AzureOpenAI

AZURE_API_CLIENT = os.getenv("AZURE_API_CLIENT")
AZURE_API_KEY = os.getenv("AZURE_API_KEY")

client= AzureOpenAI(
    api_key=AZURE_API_KEY,
    api_version= "2025-01-01-preview",
    azure_endpoint=AZURE_API_CLIENT)
```

```python
chat_prompt = [
    {"role": "user",
     "content": "Who has won the most FIFA World Cups in soccer?"}
]
```

Chat ⌘+L Edit ⌘+I

✓ 0.0s

```python
# Generate the response

response = client.chat.completions.create(
    model="my-first-gpt",
    messages=chat_prompt
)

print(response.choices[0].message.content)
```

History behind Azure OpenAI

- **2019 Partnership**: Microsoft and OpenAI formed a partnership, emphasizing the integration of OpenAI's technologies with Microsoft Azure.
- **$1 Billion Investment**: Microsoft announced an investment of $1 billion into OpenAI as part of their partnership agreement.
- **2021 Azure OpenAI Service Launch**: Launched to enterprise customers, allowing them access to OpenAI's AI models like GPT3, through Microsoft's Azure platform.
- **Expanding Capabilities**: The service has since expanded to include more sophisticated AI models, such as Codex and DALLE, enhancing its offerings for complex enterprise applications.
- **Enterprise Focus**: Designed specifically for enterprise use, Azure OpenAI aims to support businesses in incorporating AI into their operations securely and at scale.

Prereqs for RAG with Azure AI Search

Azure

i. Create a Storage account — *acasdatastore*
ii. Create Embedding Deployment — embeddingacasrag
iii. Create Chats Deployment — chatmodelrag
iv. Create Azure AI Search Service — azureaisearchrag
v. Using the chats Deployment
    i. Upload the documents.
vi. Ingestion > preprocessing > Indexing

RAG PREREQUISITES

1. Storage Account

Create a storage account

Basics | Advanced | Networking | Data protection | Encryption | Tags | Review + create

Azure Storage is a Microsoft-managed service providing cloud storage that is highly available, secure, durable, scalable, and redundant. Azure Storage includes Azure Blobs (objects), Azure Data Lake Storage Gen2, Azure Files, Azure Queues, and Azure Tables. The cost of your storage account depends on the usage and the options you choose below. Learn more about Azure storage accounts

Project details

Select the subscription in which to create the new storage account. Choose a new or existing resource group to organize and manage your storage account together with other resources.

Subscription * : QBSS1

Resource group * : naditest001

Create new

Instance details

Storage account name * : tjdatastore

Region * : (US) East US

Deploy to an Azure Extended Zone

Preferred storage type : Choose preferred storage type

This helps us provide relevant guidance. It doesn't restrict your storage to this resource type. Learn more

Performance * :

- **Standard:** Recommended for most scenarios (general-purpose v2 account)
- **Premium:** Recommended for scenarios that require low latency.

Previous | Next | Review + create

Redundancy

↳ Stores copies of Data.

2. Deploy Embedding Model

Deployments → Model

Model = 3 (large / small / Ada.)

Deploy text-embedding-ada-002

Deployment name * (Model Name.)

text-embedding-ada-tj

Deployment type

Global Standard

Global Standard: Pay per API call with the highest rate limits. Learn more about Global deployment types.
Data might be processed globally, outside of the resource's Azure geography, but data storage remains in the AI resource's Azure geography. Learn more about data residency.

Deployment details — Customize

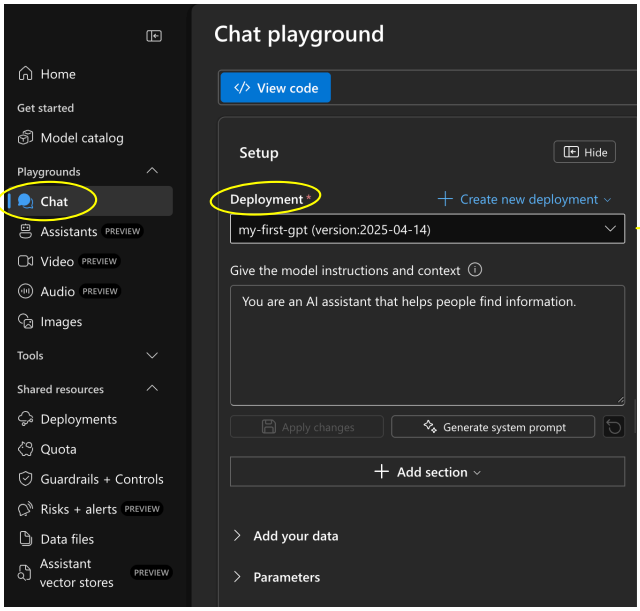
Model version: 2	AI resource: OpenAINad01
Capacity: 120K tokens per minute (TPM)	Resource location: East US
Content safety: DefaultV2	Version upgrade policy: Model version will not be automatically upgraded

Deploy | Cancel

Deployment info

Name: text-embedding-ada-tj	Provisioning state: Succeeded
Deployment type: Global Standard	Created on: 2025-10-01T18:55:31.7994573Z
Created by: [redacted]soft.com	Modified on: Oct 2, 2025 12:25 AM
Modified by: [redacted]	Version upgrade policy: Model version will not be automatically upgraded
Rate limit (Tokens per minute): 120,000	Rate limit (Requests per minute): 720
Model name: text-embedding-ada-002	Model version: 2
Life cycle status: GenerallyAvailable	Date created: Apr 3, 2023 5:30 AM
Date updated: Apr 3, 2023 5:30 AM	Model retirement date: Apr 30, 2026 5:30 AM

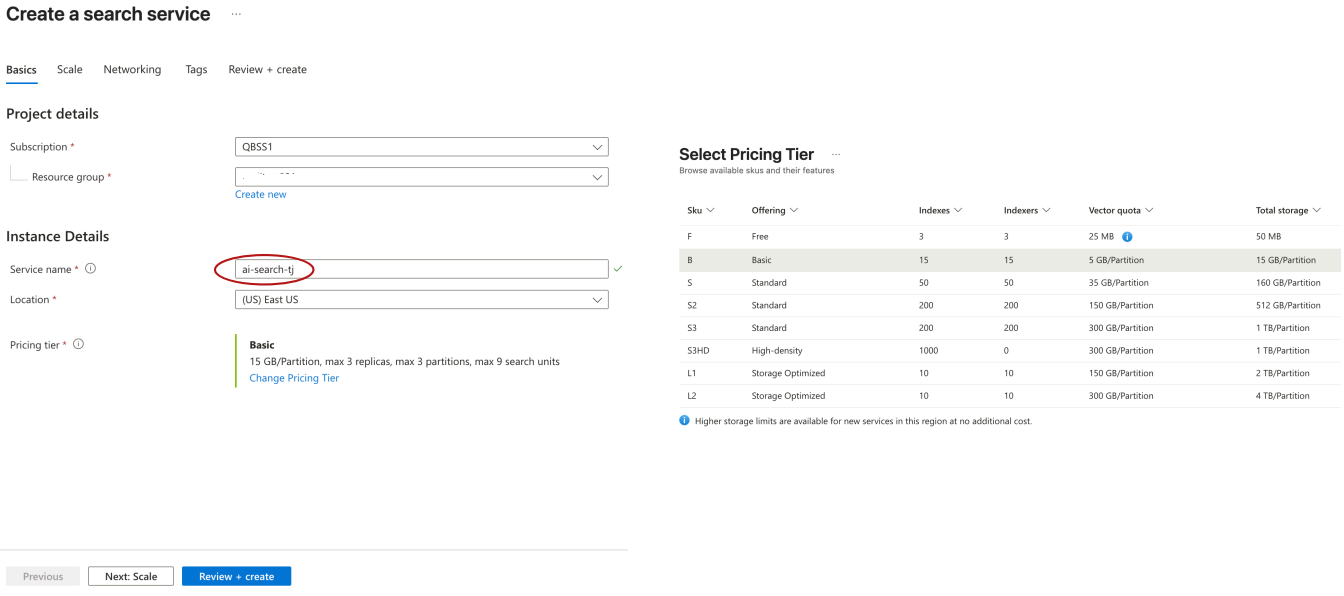
3. Chat Completion Deployment



→ Chat Deployment

4. Create AI Search

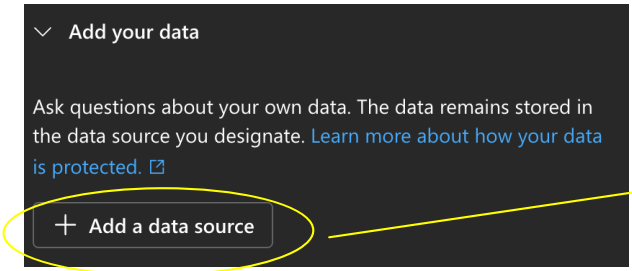
Azure portal → AI Search

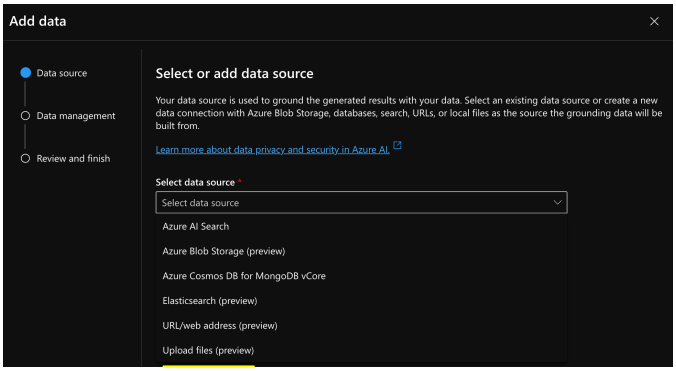


So now we have ;

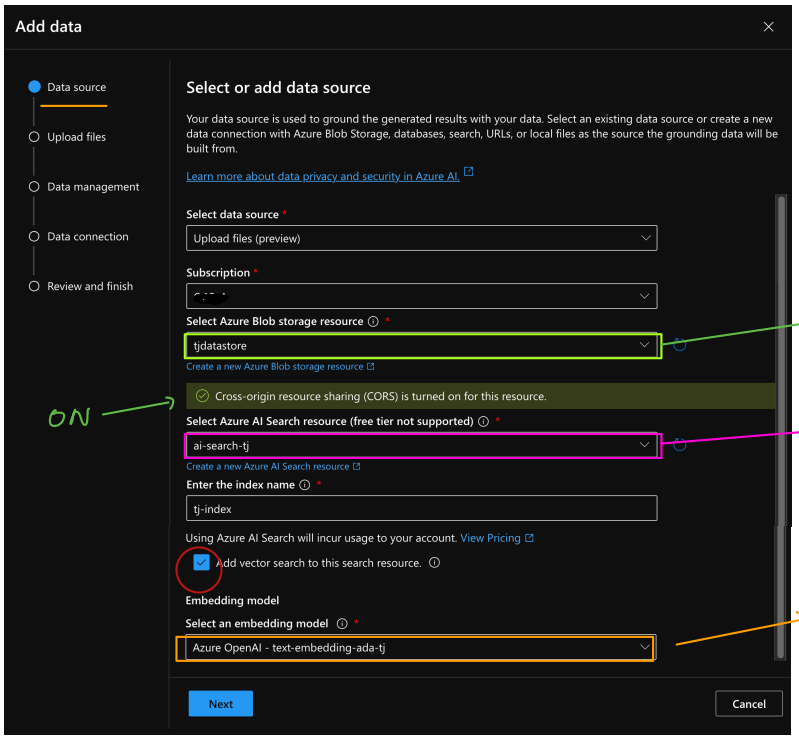
- Storage Account
- Embedding Deployment
- Azure AI Search
- Chat Model Deployment

5. Add Data





Data Sources



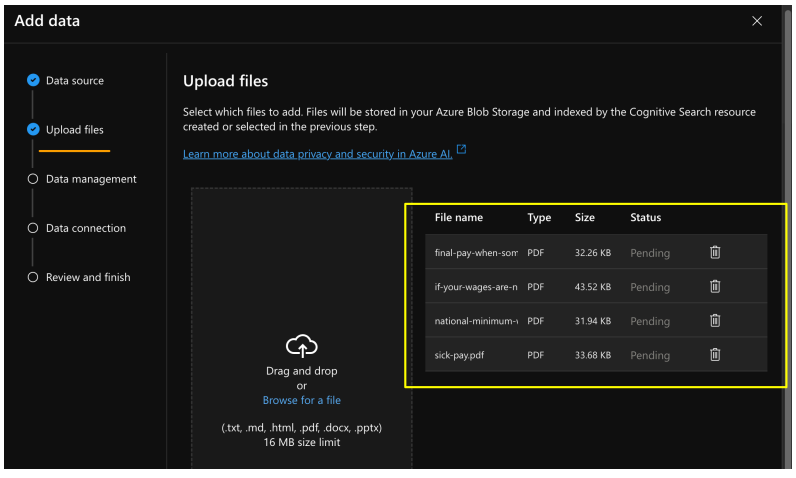
ON

Storage Account

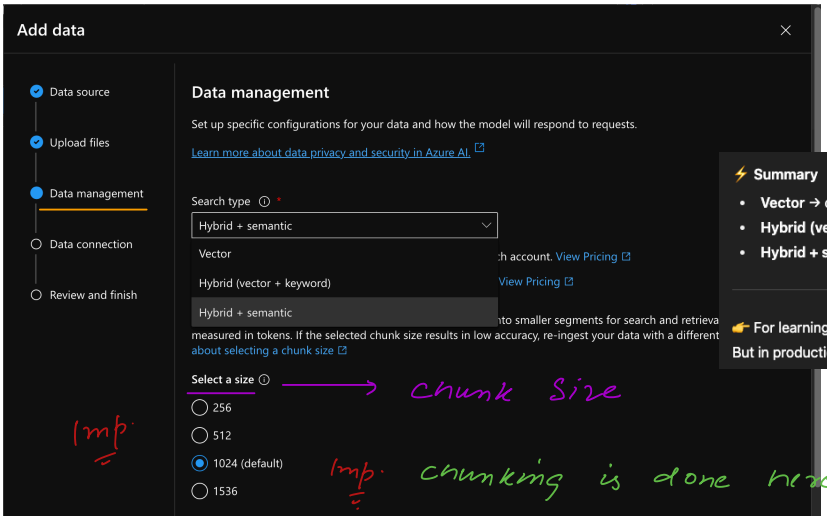
AI Search

Embedding deployment

Upload Files



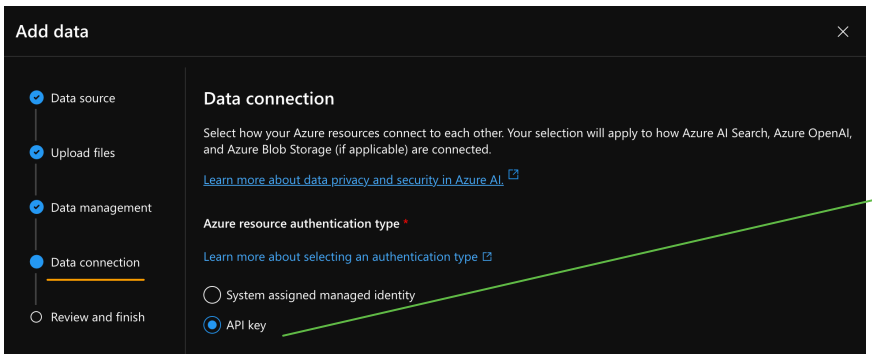
Data Management



⚡ Summary

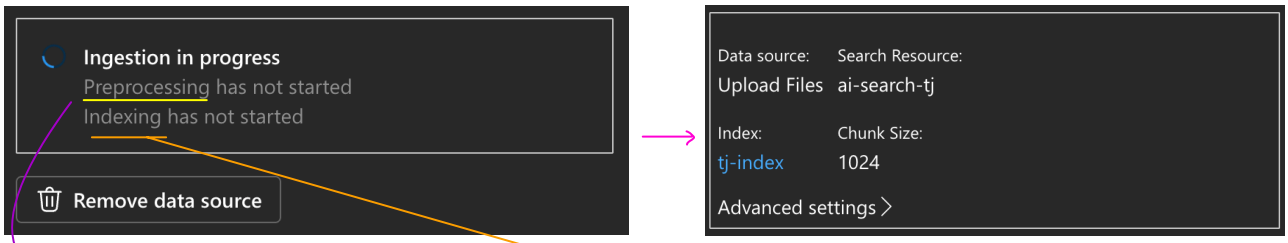
- **Vector** → only embeddings, best for semantic similarity.
- **Hybrid (vector + keyword)** → mix of keyword precision + vector meaning.
- **Hybrid + semantic** → best quality, adds AI semantic re-ranking for natural answers.

👉 For learning, you can start with **Vector** to understand embeddings.
But in production, **Hybrid + semantic** usually gives the most natural, Google-like search experience.



Data Ingestion

once Review & finish ; Data Ingestion starts



Preprocessing

Then Indexing

1. Preprocessing (data preparation)

- Azure runs **document cracking**:
  - Extracts text from PDFs, Word, PPT, HTML, scanned images (OCR).
  - Normalizes into a structured format.
- Cleans and detects fields (title, content, author, metadata).
- Optionally runs **cognitive skills** if you add them:
  - Language detection
  - Entity recognition
  - Key phrase extraction
  - Image tagging, OCR for scanned docs

👉 End result = clean, structured chunks of content + metadata.

2. Indexing (make searchable)

- Content is **pushed into the Search Index**.
- For each document/chunk, Azure creates:
  - **Full-text index** (keyword/BM25 search)
  - **Vector embeddings index** (if you've enabled vector search → usually via Azure OpenAI embedding model)
  - Metadata fields (filters, categories, tags)

👉 End result = a searchable index that can support keyword search, vector similarity, or hybrid search.

Q. what indexing method is used in Azure AI Index?

HNSW → Hierarchical Navigable Small world Graph. (Graph based)

→ It uses ANN (Approx. nearest neighbours).

✅ Key Difference in Indexing Context

Feature	KNN	ANN
Type of search	Exact	Approximate
Speed	Slow for large datasets	Fast
Accuracy	100%	Usually <100% (tunable)
Memory & CPU	High for large datasets	Optimized with index structures
Index structures	Usually none (flat)	HNSW, IVF, PQ, etc.

Imp.!

Q. what is Hybrid Search?

Hybrid = Both Keyword + vector search
BM25 HNSW

Hybrid search combines:

1. **Vector Search (Semantic / Embedding-based)**
   - **Goal:** Find semantically similar items.
   - **Index used:** ANN (like HNSW, IVF, or PQ).
   - **Input:** Query embedding vs document embeddings.
   - **Output:** Candidate vectors that are "close" in semantic space.
2. **Keyword Search (Exact / Lexical)**
   - **Goal:** Match query terms exactly or partially.
   - **Method:** BM25 (based on bag-of-words).
   - **Input:** Query terms vs document terms.
   - **Output:** Ranked documents based on term frequency, IDF, and document length.

How Hybrid Works

1. Query comes in.
2. Two parallel searches:
   - **ANN search** for embeddings → returns semantically similar items.
   - **BM25 search** for keywords → returns lexically matching items.
3. **Merge / rerank:** Combine the results from both methods using a scoring function.
   - Example: weighted sum of BM25 score + ANN similarity score.

Key Takeaways

- **ANN (HNSW)** → finds semantic neighbors, **fast but approximate**.
- **BM25 (BOW)** → finds lexical matches, **exact but shallow**.
- **Hybrid = best of both worlds** → captures both meaning and keywords.

Working of BM25 and HNSW

BM25

- **Type:** Keyword-based search (bag-of-words).
- **How it works:** Scores documents based on **term frequency**, **inverse document frequency**, and **document length**.
  TF-IDF → more frequent words unique words
- **Example:**
  - Query: `"data cleaning"`
  - Document A: `"Data cleaning techniques for Python"` → higher score
  - Document B: `"Machine learning in Python"` → lower score
  Rank high

HNSW (ANN) Graph – Semantic Aware

- **Type:** Approximate nearest neighbor search for **vectors**.
- **How it works:** Builds a **graph of vectors**, traverses efficiently to find nearest neighbors in high-dimensional space.
- **Example:**
  - Query embedding: vector of `"data cleaning"`
  - Returns documents semantically similar even if keywords differ, e.g., `"Python preprocessing methods"`

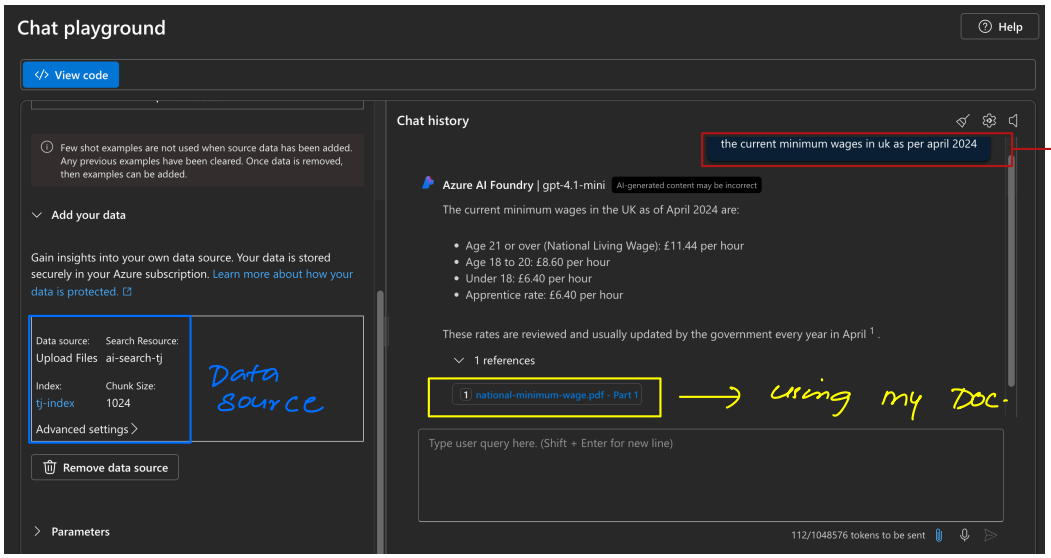
Key Difference

- BM25 → exact keyword match.
- HNSW → semantic similarity using embeddings.

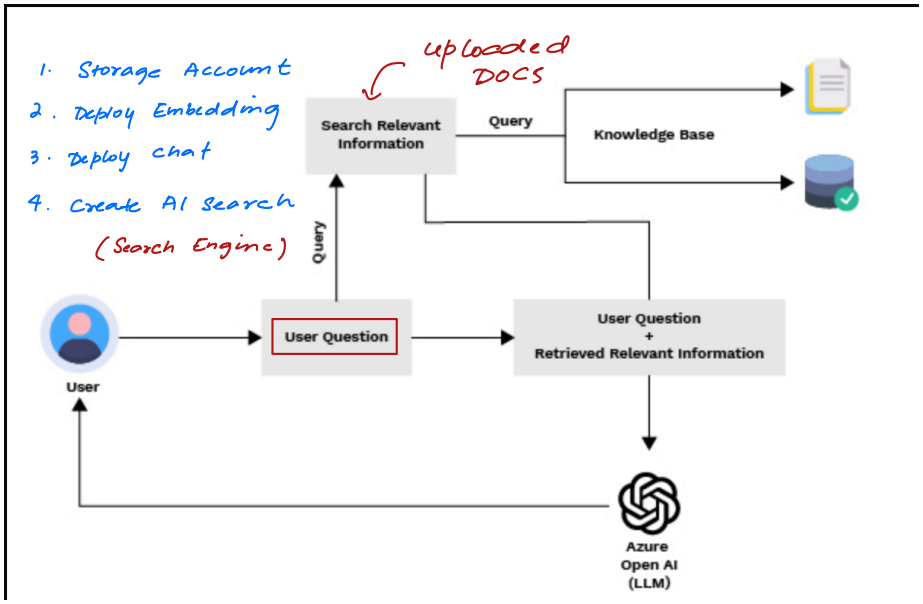
Q1 Why indexing ?

For fast and efficient retrieval of Data.
Filtering etc.

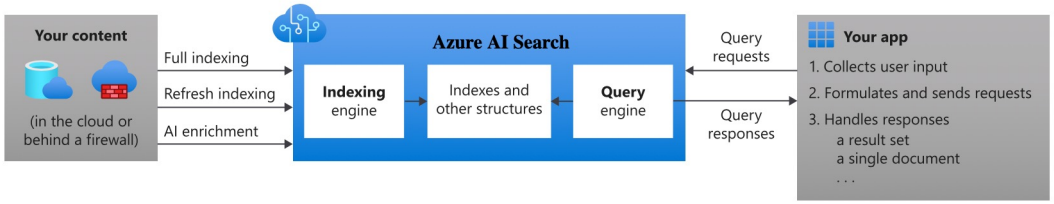
CHAT PLAYGROUND WITH OWN DATA

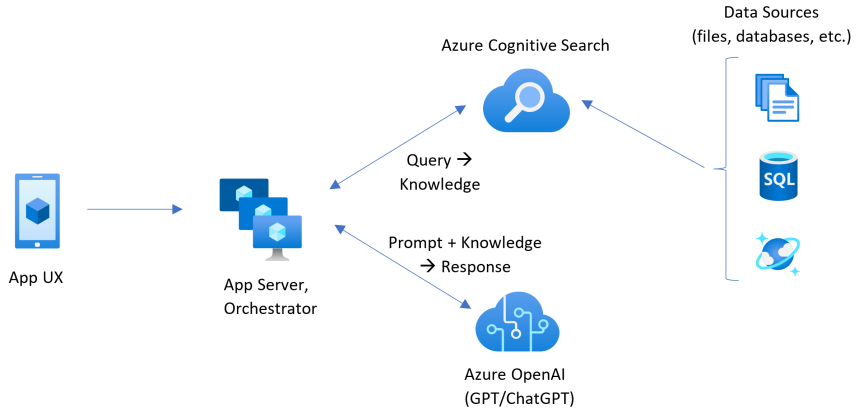


Complete process



Azure AI Search





Azure AI Search (formerly known as "Azure Cognitive Search") provides secure information retrieval at scale over userowned content in traditional and generative AI search applications.

Data Chunking

Instead of treating the entire document as a single unit, chunking divides it into smaller segments, typically based on paragraphs, sections, or sentences.

AI Powered Indexing:

Utilizes AI to automatically extract and enrich data from a variety of sources including documents, images, and media files.

Semantic Search:

Uses advanced machine learning models to understand the **intent** behind queries, providing more relevant results.

Cognitive Skills:

Integrates predefined cognitive skills such as optical character recognition (**OCR**), language **detection**, and entity recognition or allows the creation of custom skills.

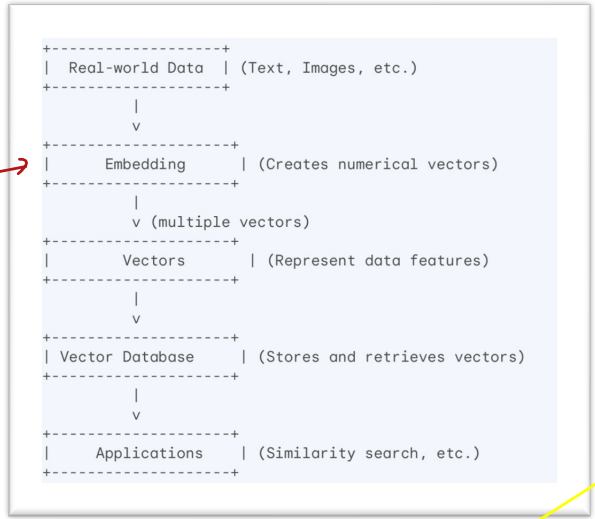
Scalable Infrastructure:

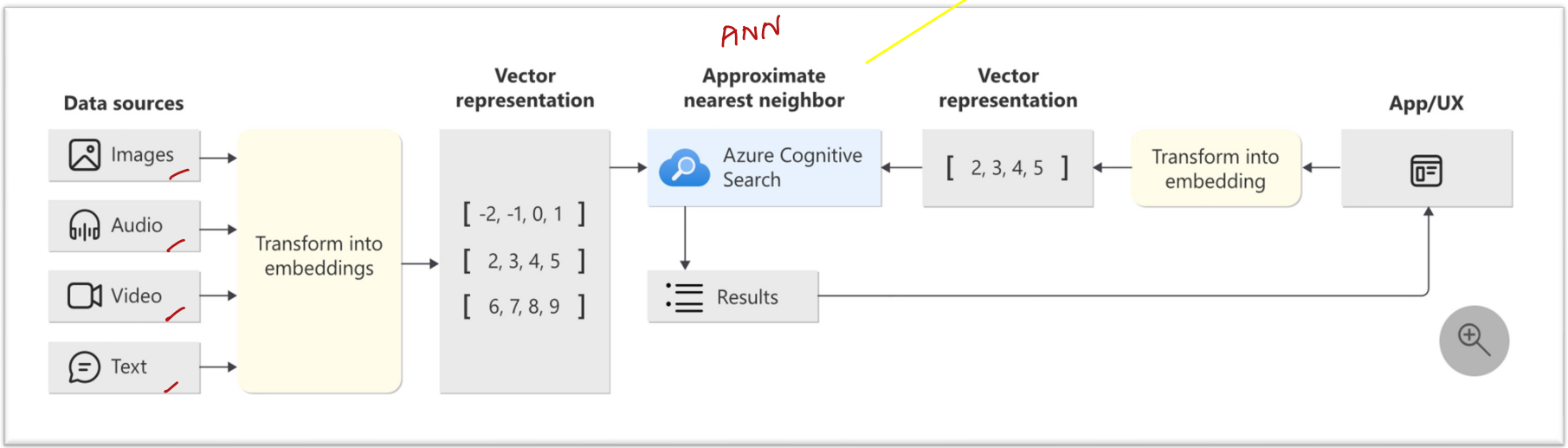
Automatically scales to accommodate data volume and query traffic, ensuring efficient performance without manual intervention.

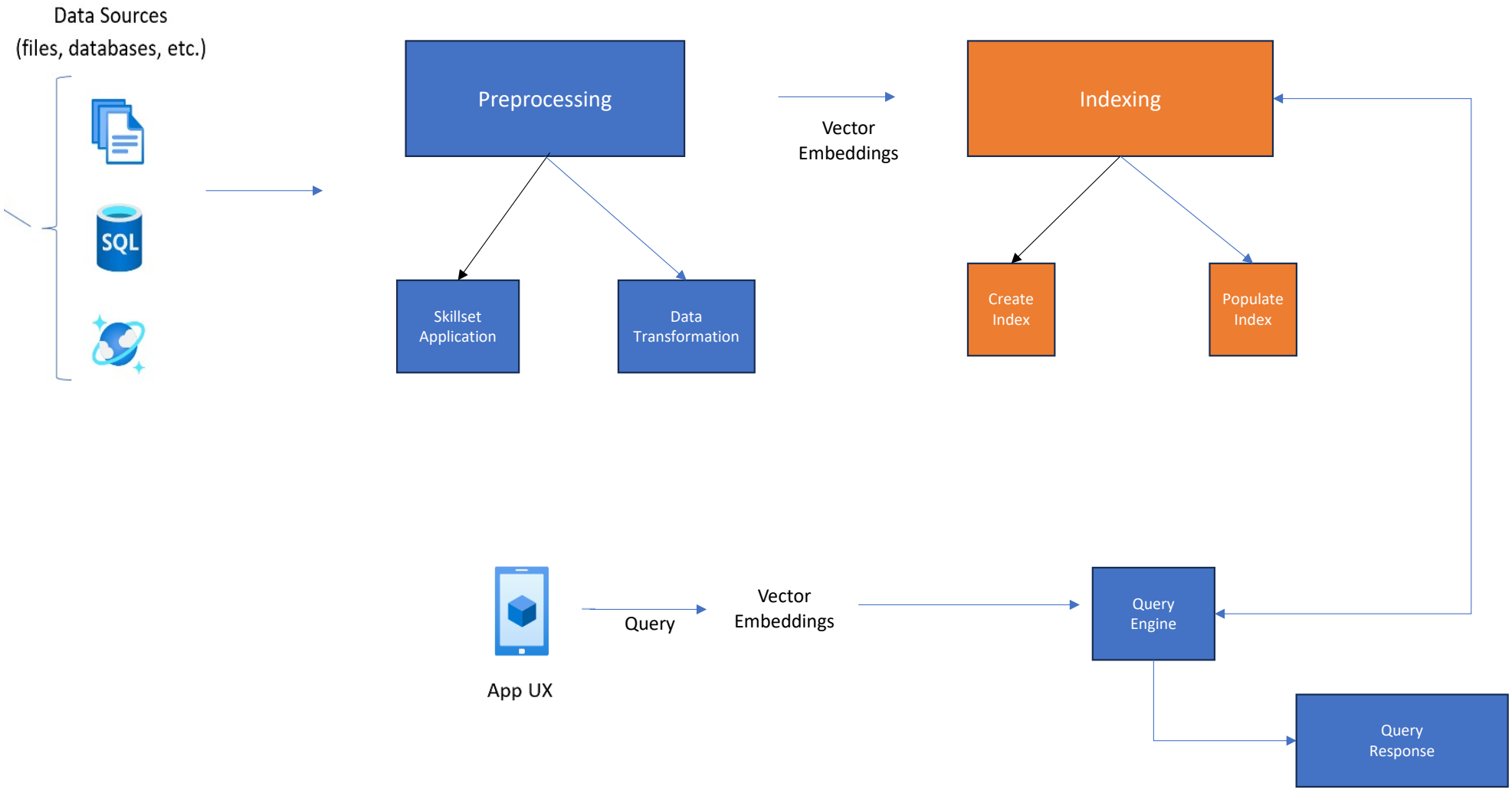
Integration with Azure Ecosystem:

Works seamlessly with other Azure services, such as Azure AI, Azure Functions, and more, for comprehensive data processing and handling.

How vector search works in Azure AI Search

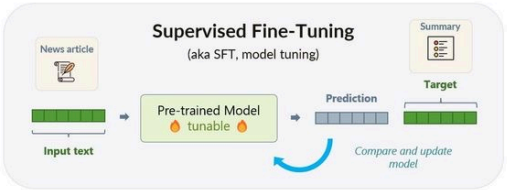





Data Sources
(files, databases, etc.)
SQL
Preprocessing
Skillset Application
Data Transformation
Vector Embeddings
Indexing
Create Index
Populate Index
App UX
Query
Vector Embeddings
Query Engine
Query Response

Azure OpenAI Fine Tuning

Specific Data.



Babbage-002 & Davinci-002:

- GPT3 based: smaller, lower latency
- Understand & generate natural language or code
- Completion support

GPT-35-Turbo:

- Most capable & cost effective GPT-3.5 model
- More sophisticated capabilities
- Chat support

Fine-tuning models

Models	Training per compute hour	Hosting per hour	Input Usage per 1,000 tokens	Output Usage per 1,000 tokens
Babbage-002	N/A	N/A	N/A	N/A
Davinci-002	N/A	N/A	N/A	N/A
GPT-3.5-Turbo (4K)	$45	$3	$0.0005	$0.0015
GPT-3.5-Turbo (16K)	$68	$3	$0.0005	$0.0015

Customization of PreTrained Models: Tailor models like GPT to specific data needs.

Improving Model Performance: Aligns model outputs more closely with userspecific styles and terminologies.

Dataset Requirements: Users provide their own data for training to finetune the models.

Training Process: Finetuning continues training a pretrained model on new datasets, requiring fewer resources than training from scratch.

Cost : Fine-tuning on Azure OpenAI can be expensive, especially for large models and extensive datasets.
Compute Costs: 10 hours * $10/hour = $100
Storage Costs: 5 GB * $0.02/GB = $0.10
API Usage: 500,000 tokens * $0.06/1,000 tokens = $30
Total Estimated Cost: $130.10

Accessibility through Azure: Integrated into Azure, finetuning benefits from its cloud infrastructure and services.

Custom Deployments: Models can be easily deployed within Azure, integrating with other Azure services.

Fine Tuning

Customization of already existing LLM, so it can act as

- Task specific
- Domain specific.

- **Example:**
  - Pre-trained GPT → fine-tuned on legal documents → becomes better at answering legal questions.
  - LLaMA 2 → fine-tuned on customer support chats → becomes a specialized support assistant.

✅ **TL;DR:** Fine-tuning = adapting a general LLM to a specific domain or task without training from scratch.

Imp. Fine Tuning can be really costly, so we are encouraged to use prompting rather.

Cost : Fine-tuning on Azure OpenAI can be expensive, especially for large models and extensive datasets.
Compute Costs: 10 hours * $10/hour = $100
Storage Costs: 5 GB * $0.02/GB = $0.10
API Usage: 500,000 tokens * $0.06/1,000 tokens = $30
Total Estimated Cost: $130.10

Imp. Not all model can be fine tuned and not all models are available in all regions.

Azure AI

✅ Azure OpenAI Models Available for Fine-Tuning

Model	Fine-Tuning Availability	Notes
GPT-3.5-Turbo (0613)	✅ Yes	Widely used for general-purpose tasks.
GPT-4	✅ Yes (Preview)	Advanced capabilities; fine-tuning is in preview.
GPT-4o	✅ Yes	Optimized for efficiency; fine-tuning is in preview.
GPT-4o-mini	✅ Yes	Lightweight version; fine-tuning is in preview.
GPT-4.1	✅ Yes	Latest model; fine-tuning is available in all regions with Global Training.
GPT-4.1-mini	✅ Yes	Mini version of GPT-4.1; fine-tuning is available in all regions with Global Training.
GPT-4.1-nano	✅ Yes	Nano version of GPT-4.1; fine-tuning is available in all regions with Global Training.

Open Source

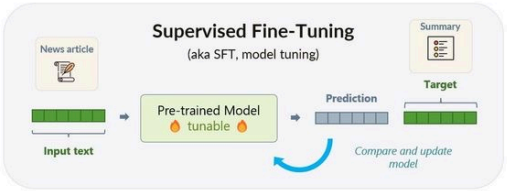
Here's a **shortlist of the most important open-source models that can be fine-tuned:**

Model	Type	Notes / Example Use
LLaMA 2 (Meta)	Text / Chat	7B–70B parameters; widely used for conversational and general NLP fine-tuning
Mistral	Text	Efficient models (7B–8x7B); general NLP tasks, Apache 2.0 license
T5 (Google)	Text	Text-to-text transformer; translation, summarization, classification
BERT (Google)	Text	QA, classification, sentiment analysis; standard NLP downstream tasks
CodeLlama	Code / Text	Optimized for code generation, understanding, and code-related tasks

Mostly used; GPT 3.5 LLAMA 2

Parameters; 175b 7-70b

Need for Fine Tuning



Babbage-002 & Davinci-002:

- GPT3 based: smaller, lower latency
- Understand & generate natural language or code
- Completion support

GPT-35-Turbo:

- Most capable & cost effective GPT-3.5 model
- More sophisticated capabilities
- Chat support

- **DomainSpecific Knowledge:** When the model needs to generate text with specialized vocabulary and concepts.
- **Custom Instructions and Formatting**: When output needs to follow specific templates or formats.
- **Company or Brand Voice**: When text needs to reflect a particular style or tone.
- **Proprietary or Confidential Information**: When generating or processing sensitive information specific to an organization.
- **Enhanced Context Understanding**: When maintaining context over long interactions is crucial.
- **Unique Customer Interactions**: When handling specific customer queries unique to a product or service.
- **Regulatory Compliance**: When generating text that adheres to specific legal or regulatory standards.
- **Interactive Applications**: When creating engaging and interactive user experiences.

Imp Q// when to choose Fine Tuning vs prompts.

Practical Rule of Thumb

Use Fine-Tuning when:

- Data is **relatively static.**
- You want **highly consistent answers** in a specific domain.
- You want to **teach the model company-specific terminology.**

Use Prompt Engineering when:

- Data changes frequently (like Power BI dashboards).
- You need **flexible, dynamic responses.**
- Data is **confidential** — RLS or filters can be applied before sending to the LLM.
- You want to **avoid training costs.**

Most Preferred in Corporates

Hybrid Approach

Fine-tune for domain knowledge + Prompt Engineering for dynamic context.

Example:

- Fine-tune on your **company style, tone, terminology.**
- Use **prompt engineering** to feed **real-time data** (e.g., Power BI rows filtered by RLS).

This is the **best of both worlds**, especially for corporate chatbots.

TL;DR:

- **Fine-tune = teach the model once** (good for static, domain-specific knowledge).
- **Prompt engineering = guide the model dynamically** (good for live, changing, confidential data).

Q// How to Fine Tune ?

Train	Test
Json	Json

GPT 3.5 Turbo

1. Training Data

example

```
{"messages": [{"role": "system", "content": "You are a chatbot that always responds in a humorous way."},
              {"role": "user", "content": "What did you do over the weekend?"},
              {"role": "assistant", "content": "I became a professional couch potato and mastered the art of napping."}]}
```

What you are trying to do

1. **Model Behavior Customization**
   - You want the model to **always respond in a humorous way.**
   - Each example in your dataset shows the **desired style of response** (funny, witty, playful).
2. **Structure**
   - Each entry is a JSON object with a `"messages"` array:

```
json                                                    Copy code

{"role": "system", "content": "You are a chatbot that always responds in a humorou
{"role": "user", "content": "User question"},
{"role": "assistant", "content": "Desired humorous response"}
```

   - `"system"` : sets the **behavior/tone of the model.**
   - `"user"` : the **prompt/question** from the user.
   - `"assistant"` : the **response you want the model to generate.**
3. **Goal**
   - After fine-tuning on this dataset, the model **learns to consistently respond humorously**, even to new questions it hasn't seen before.
   - The model is **adapting its weights** based on the examples you provided.

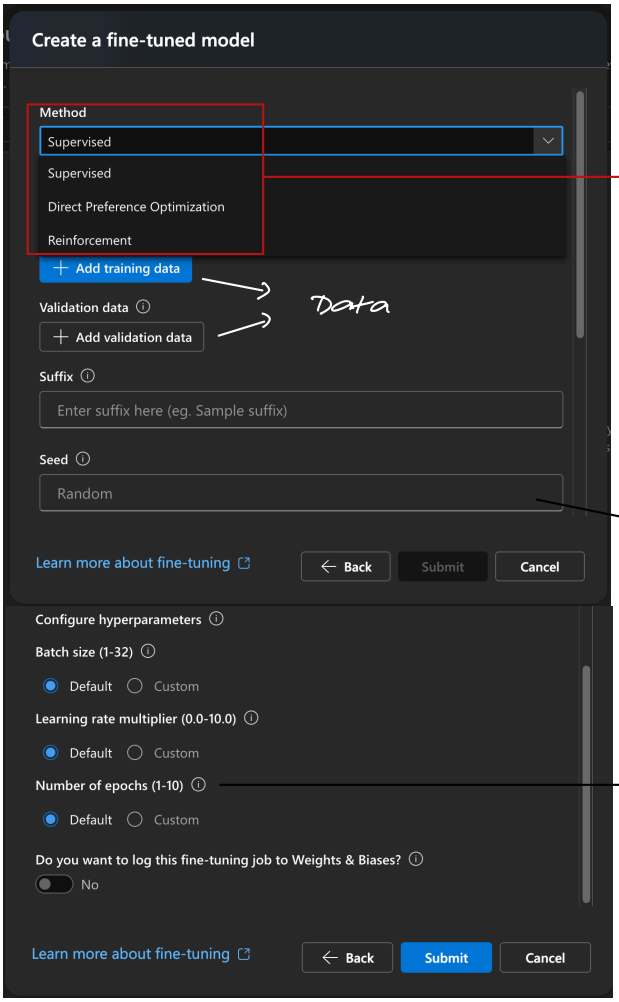
- System — Instruction
- User — Query
- Assistant — Answer

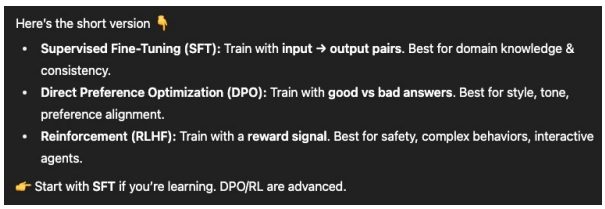
2. Validation Data

Similar to Train Data but not trained, this data is rather used for validation.

Imp. GPT 3.5 is best for experiment and is available East US 2.

FINE TUNING PRACTICALS





SFT = Supervised F.T

Mostly used.

Seed

setting it will give same output multiple times.

Epoch

one complete cycle of train data

Imp. Epochs = 6 ; Training prompts = 15

Total steps = 6 x 15 = 90

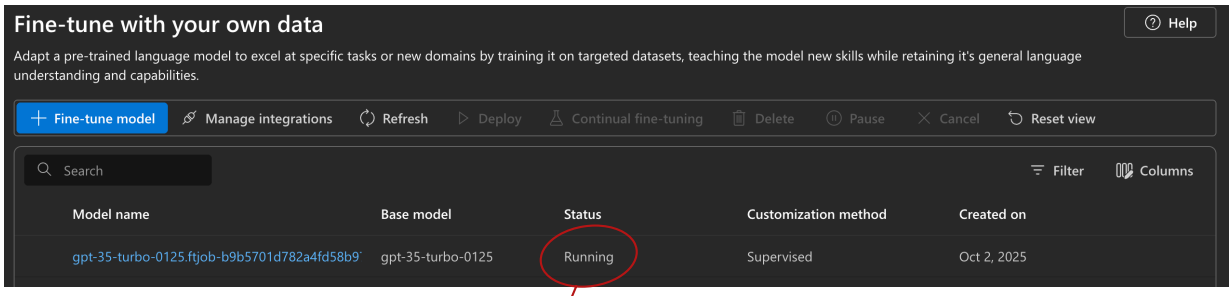
→ Loss should be very low

↳ How well model is learning overtime

→ Validation Loss → Training Data.

Deploy Fine Tune Model

So, we will be deploying our F.T model as Base model in Deployments

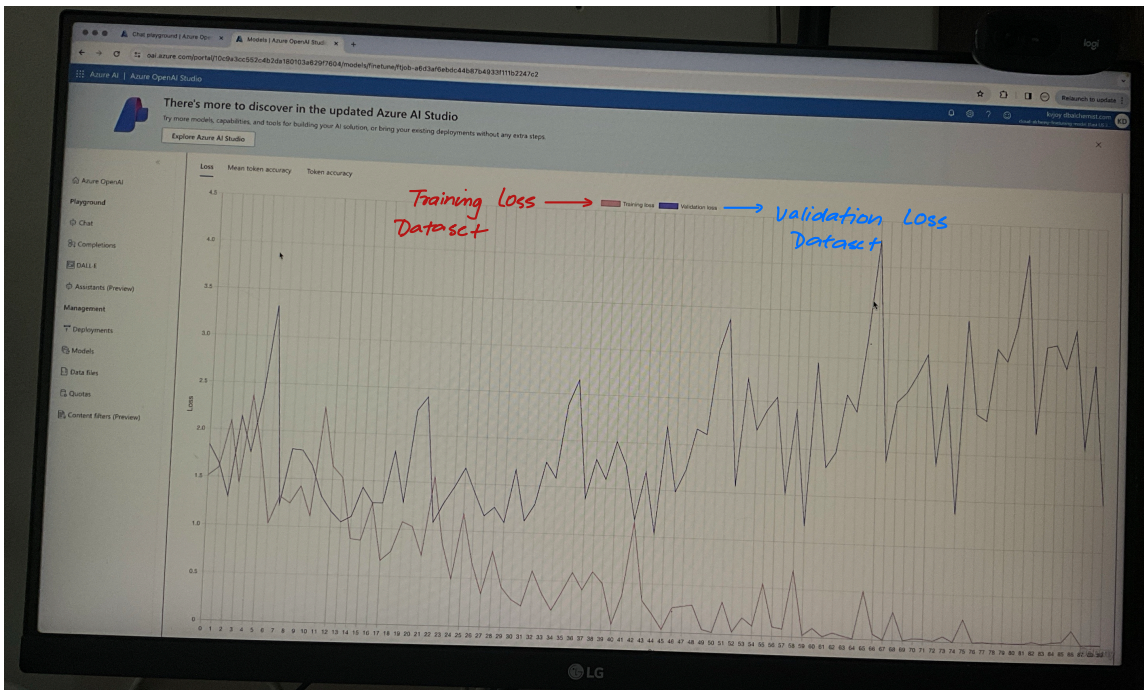


Once it is completed

↓

we will deploy F.T model as Base model

Fine Tuned Model Evaluation;

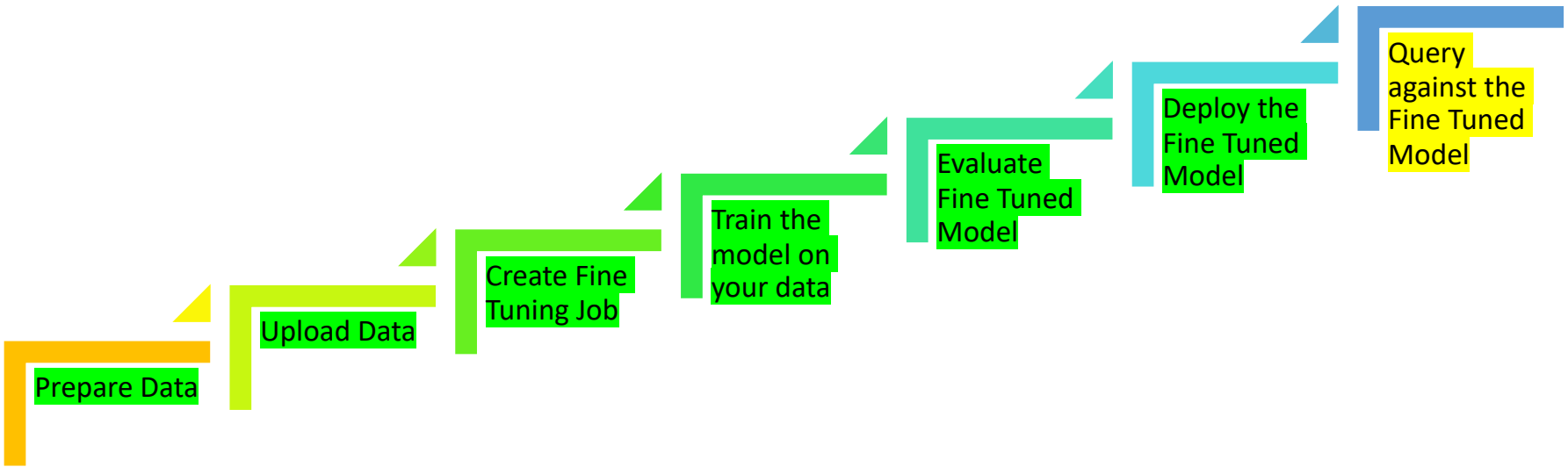


Imp.! Loss should always be minimum.

Also, 90 steps = 15 Datapoints x 6 epochs

Model = GPT 35 Turbo.

Azure OpenAI Fine Tuning



Content Filtering ; Filter unsafe Data

Types ; 1. Prompt Filter I/P

2. Response Filter O/P

🔷 What is Content Filtering in Azure AI?

- Azure AI has **built-in filters** that check both **user inputs (prompts)** and **model outputs (responses)**.
- These filters categorize and block unsafe content such as:
  - **Hate / Violence** (e.g., discrimination, harassment, graphic violence)
  - **Sexual content** (adult, explicit, or sexual services)
  - **Self-harm** (suicide, eating disorders, substance abuse encouragement)
  - **Profanity** (abusive or offensive language)

🔷 How it Works

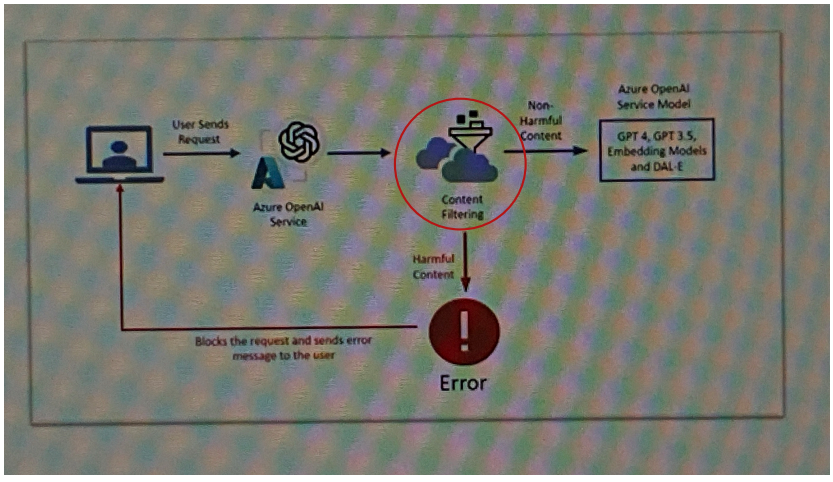
1. **Prompt Filtering (Input):**
   Before your request is sent to the model, it's scanned. If it contains harmful material, the request may be rejected.
2. **Response Filtering (Output):**
   After the model generates a response, Azure's filters check the content. If it violates policies, the output is blocked or replaced with a filtered message.

⚡ Example:

If you ask an Azure OpenAI model: *"Give me instructions to harm someone,"*

- **Input filter** → blocks the request.
  If you ask: *"Tell me a violent story,"*
- **Output filter** → might block or redact part of the generated story.

Working ;



- pretrained Filters
- Realtime

Two phases

I/P Filtering
↳ Blocked

O/P Filtering
↳ Modified

🔷 Categories of Content Filtering in Azure OpenAI

Category	What It Covers	Examples
Hate & Fairness	Hate speech, derogatory remarks, demeaning language targeting identity groups	"All people of [group] are worthless"
Violence	Descriptions, threats, or promotion of violence, gore, or harm	"Tell me how to attack someone"
Self-harm	Suicide, self-injury, eating disorders, substance abuse encouragement	"How can I kill myself?"
Sexual	Sexually explicit or pornographic content, sexual services, child sexual abuse	"Write an erotic story about..."
Profanity *(sometimes separated, sometimes merged with hate/abuse)*	Offensive words, insults, abusive language	Heavy swearing or slurs

🔷 Severity Levels

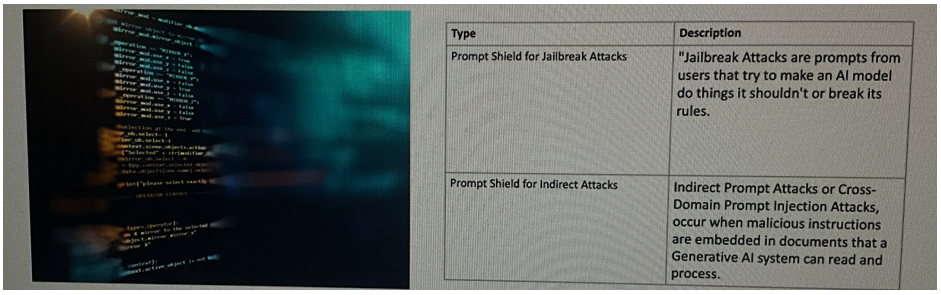
Each category is usually classified into **four severity levels**:

- **Safe** → No harmful content detected.
- **Low** → Mild or safe-for-work content (e.g., mild profanity, safe educational mention).
- **Medium** → Potentially harmful, may require review (e.g., violent threats in hypothetical context).
- **High** → Clearly harmful and blocked (e.g., explicit suicide instructions).

✅ Together, these **categories + severity levels** decide whether the **prompt or response gets blocked, allowed, or flagged**.

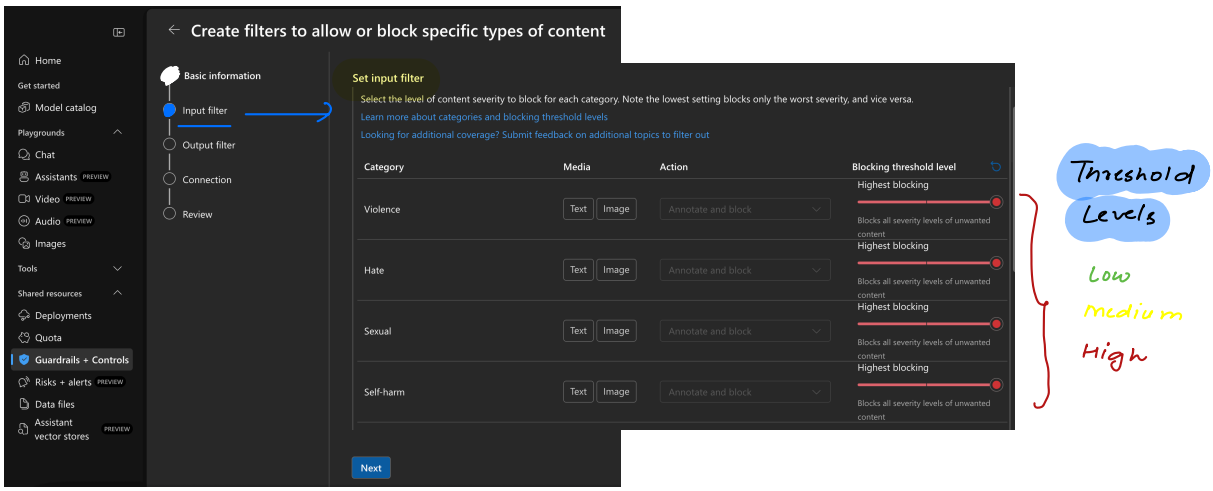
Prompt Shield is used to avoid

- Jail break Attack.
- Indirect malicious Attack

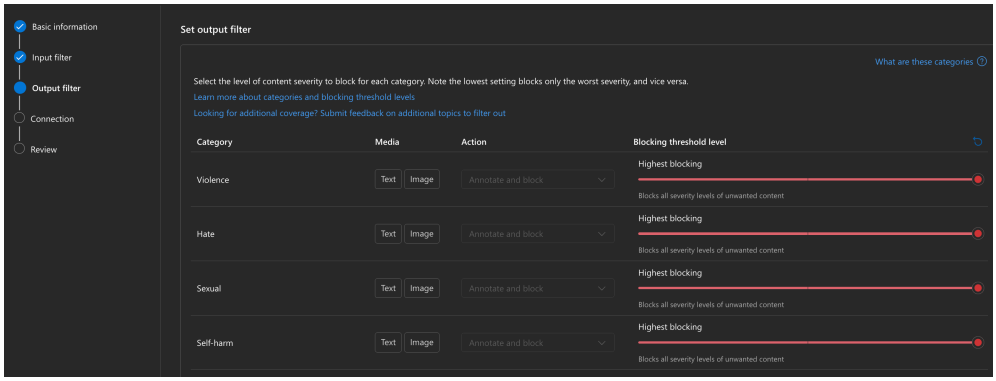


Type	Description
Prompt Shield for Jailbreak Attacks	"Jailbreak Attacks are prompts from users that try to make an AI model do things it shouldn't or break its rules.
Prompt Shield for Indirect Attacks	Indirect Prompt Attacks or Cross-Domain Prompt Injection Attacks, occur when malicious instructions are embedded in documents that a Generative AI system can read and process.

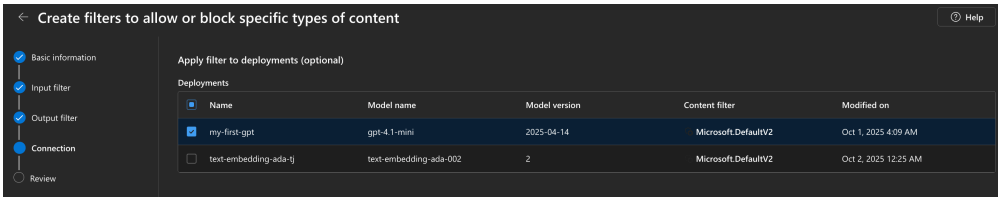
Q. How to create a content filter?



Filters Similarly let's add output filter



Connection ⟶ make this filter for model.

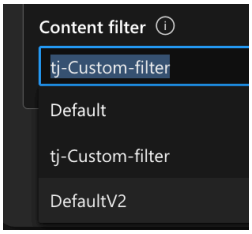


Deploying Context Filter

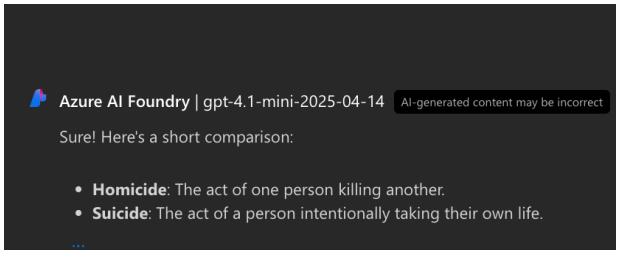
1. Deploy chat model

Imp. while deployment make sure to change context filter

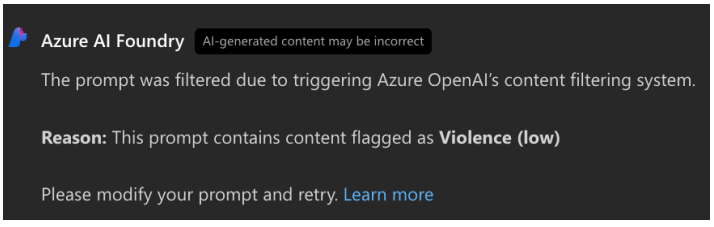
Default ⟶ New filter



Before C.F



After C.F.



Imp. Again there are 3 levels.

High Medium Low

Azure RBAC

Role Based Access Control

It's a way to **manage who has access to what** in your Azure resources. Instead of giving everyone admin rights (😬 dangerous), RBAC lets you assign specific roles to users, groups, or applications — so they only get the permissions they actually need.

🔹 Key Points About RBAC in Azure

1. **Role-Based** → Access is controlled through roles, not individual permissions each time.
2. **Scope-Based** → You can assign roles at different levels of Azure:
   - **Management Group**
   - **Subscription**
   - **Resource Group**
   - **Individual Resource**

   (Permissions flow downward — e.g., a role at subscription level applies to all resource groups inside it).
3. **Principals** → You assign roles to:
   - **Users**
   - **Groups**
   - **Service Principals** (apps)
   - **Managed Identities**
4. **Deny by Default** → Unless a role is assigned, a user has no access.

🔹 Common Built-in Roles

- **Owner** → Full access, including role assignments.
- **Contributor** → Create & manage resources, but can't grant access. ⟶ can't assign Roles
- **Reader** → View-only access.
- Api Caller → can only call Api's

There are also **hundreds of specialized roles** (e.g., Virtual Machine Contributor, Storage Blob Reader). You can also **create custom roles.**

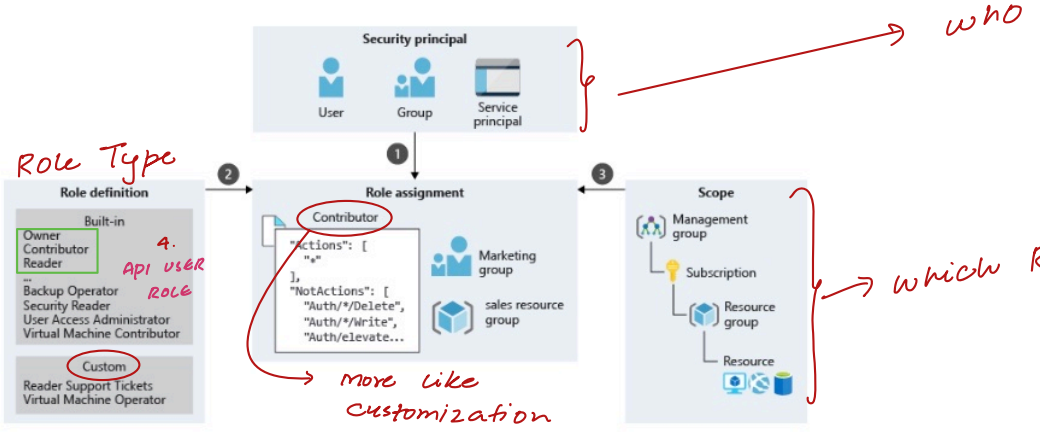
only Difference

Q,, Is RBAC similar to RLS ?

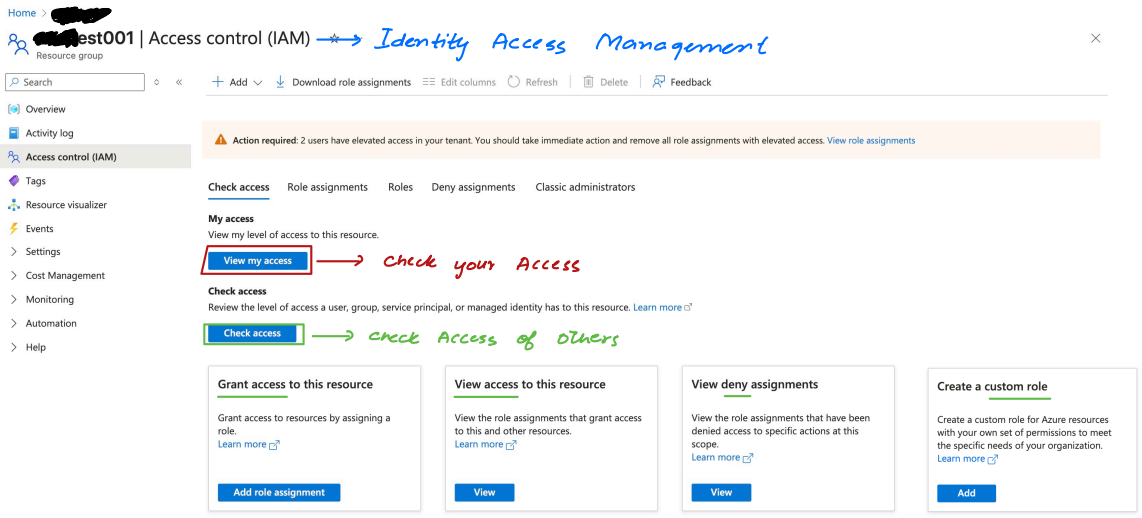
🔹 Difference in one line:

- **RBAC** = *Access to Azure resources (infra/app level).*
- **RLS** = *Access to rows in a dataset (data level).*

Azure RBAC MODEL



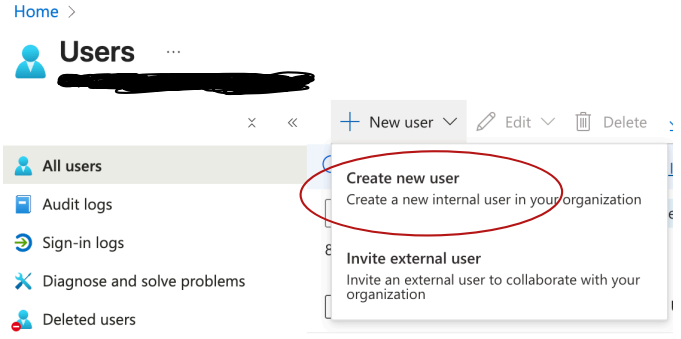
Access Management

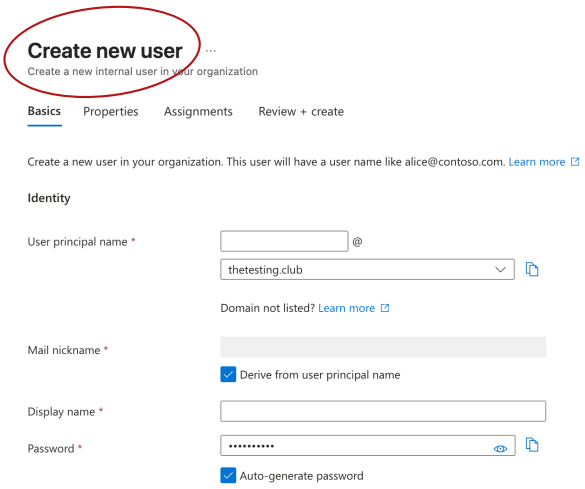


Steps to Role Management

1. Create User

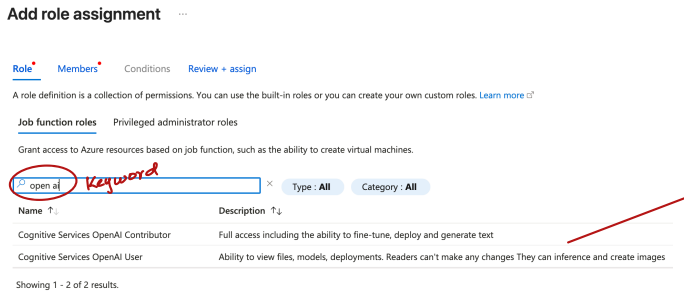
Azure portal → Users

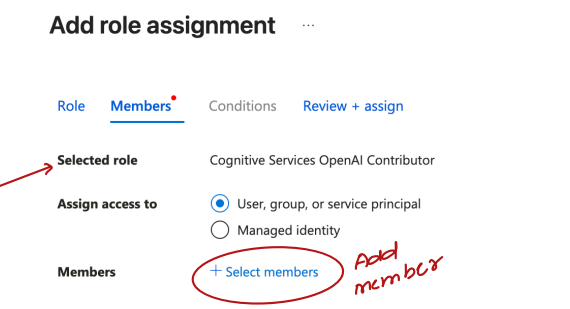




11. Role Assignment (IAM)

IAM

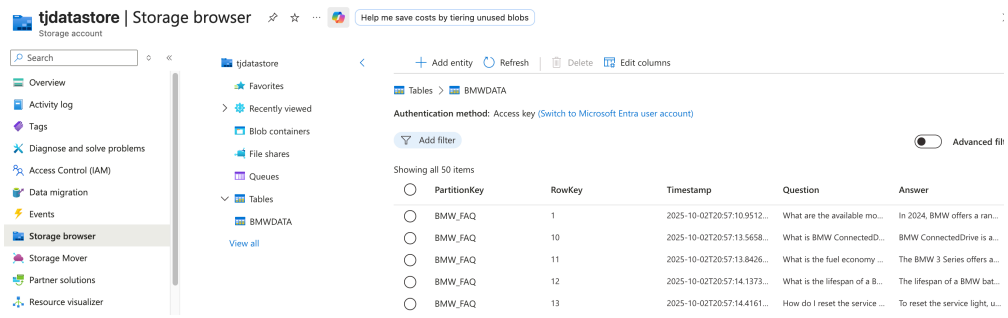




RAG + AZURE AI

1. Set up Storage Account

2. Store Table in Azure Table Storage



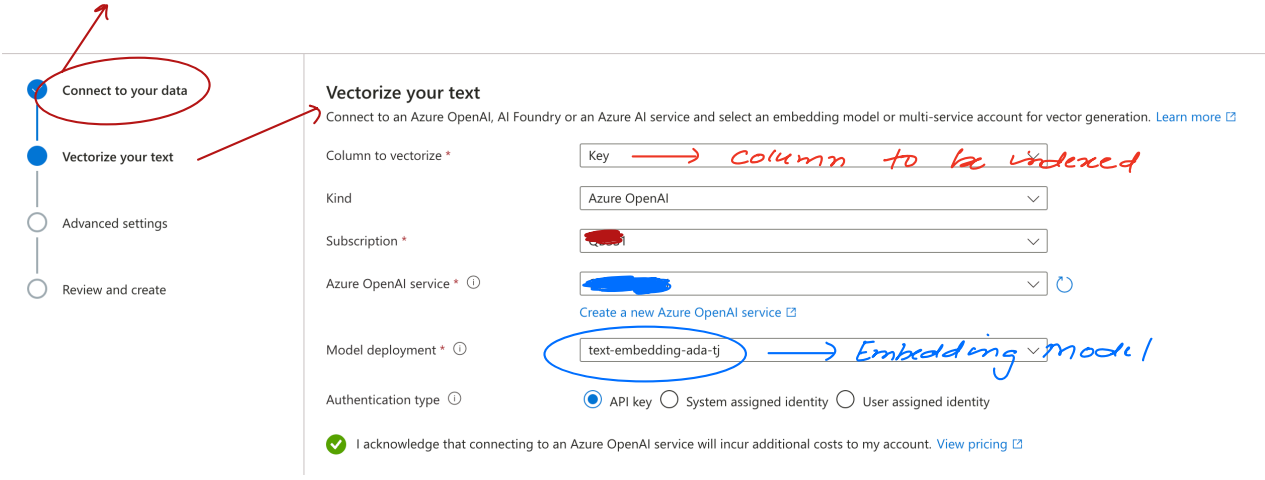
3. Azure AI Search ⟶ load Document

Azure AI Search + Embedding Flow

1. **Embed docs** → use `text-embedding-3-small/large`.
2. **Create index** → define fields: `id`, `content`, `vector`, `metadata`.
3. **Store** → push text + vectors into the index.
4. **Query** → embed user query → run **vector similarity search.**
   - Metrics supported:
     - ✅ **Cosine similarity** (most common, default)
     - ✅ **Dot product**
     - ✅ **Euclidean (L2) distance**
5. **Return** → top results → feed into RAG.

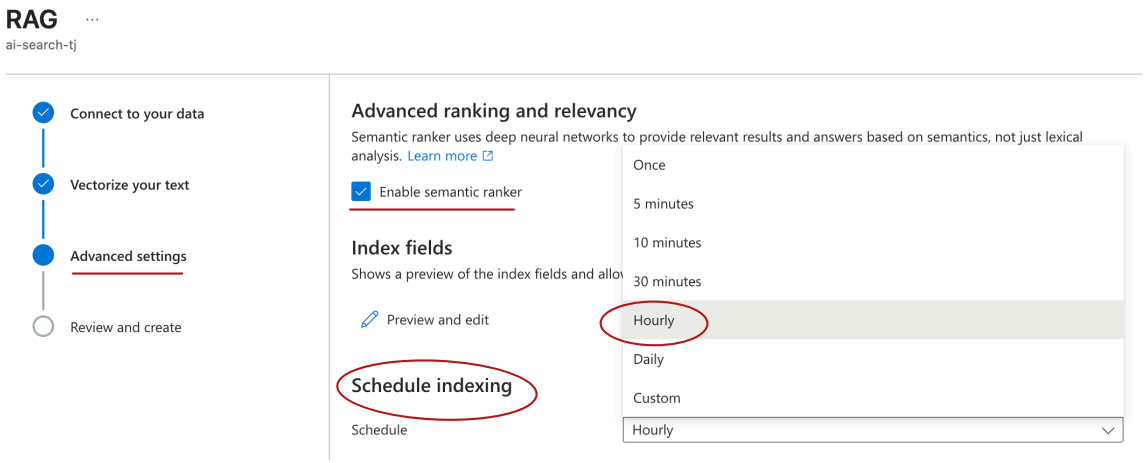
Go to Azure AI Search - Import Data

Choose RAG / Keyword (BMW Data)



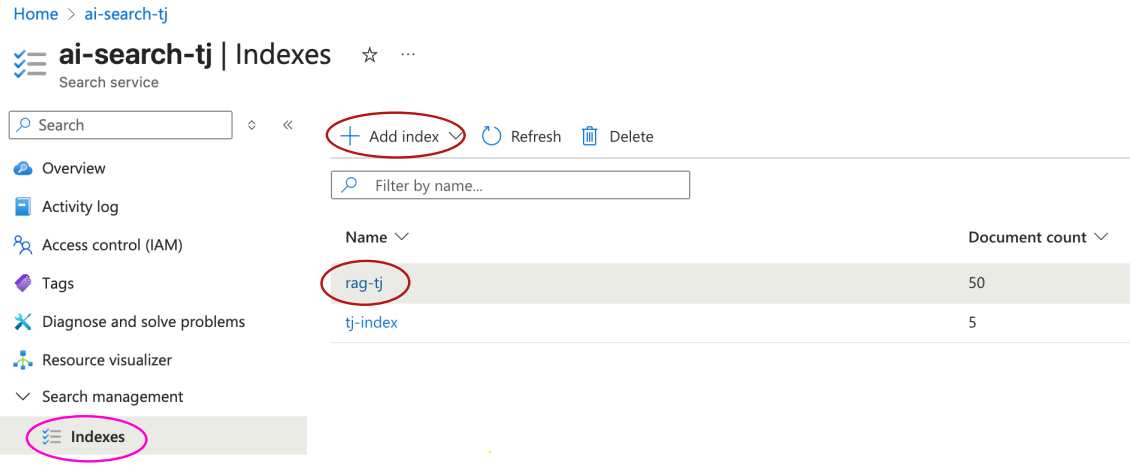
Imp: So here I will be assigning

Text Embedding Model & Index Type



4. Set env Variables for AI Search

- AI Search Acc. ai-seach-tj
- Index Name rag-tj
- AI Search Api Key. ***



Code Example

```
# These variables configure the search service and index for retrieving documents

# Set the Azure AI Search service name
os.environ["AZURE_AI_SEARCH_SERVICE_NAME"] = "ai-search-tj"

# Set the Azure AI Search index name to query
os.environ["AZURE_AI_SEARCH_INDEX_NAME"] = "rag-tj"

# Set the Azure AI Search API key for authentication
os.environ["AZURE_AI_SEARCH_API_KEY"] = "0CVzv2rS1feYv99m1BNkvTvDKiRICZCILzcn[redacted]
```

} very imp for Retriever

1. Azure AI Retriever

```
# Step 1: Initialize the AzureAI Search Retriever
# This retrieves relevant documents based on the user query from the Azure Search index
retriever = AzureAISearchRetriever(
    content_key="Answer",  # The key for the content field in the search results change it accordingly as per your data
    top_k=1,               # Number of top results to retrieve
    index_name="rag-tj"  # Name of the Azure Search index to query
)
```

2. Prompt

```
# Step 2: Define the prompt template for the language model
# This sets up how the context and question will be formatted for the model
prompt = ChatPromptTemplate.from_template(
    """Answer the question based only on the context provided.
Context: {context}  # Placeholder for the context from the retriever
Question: {question}  # Placeholder for the user question"""
)
```

3. LLM

```
llm = AzureChatOpenAI(
    azure_deployment="my-first-gpt",   # The name of your deployed model in Azure
    api_version="2025-01-01-preview",
    azure_endpoint=AZURE_API_CLIENT,
    api_key=AZURE_API_KEY
)
```

4. Chain ⟶ Context, Prompt, LLM

```
# This chain will process the retrieved context and the user question
chain = (
    {"context": retriever , "question": RunnablePassthrough()}  # Set context using the retriever and format it
    | prompt                                                    # Pass the formatted context and question to the prompt
    | llm                                                       # Generate a response using the language model
    | StrOutputParser()                                         # Parse the output to a string format
)

# Step 5: Infinite loop for user input
while True:
    user_question = input("Please enter your question (or type 'end' to exit): ")
    if user_question.lower() == "end":
        print("Exiting the loop. Goodbye!")
        break

    # Pass input as a dict matching the chain keys
    response = chain.invoke({"question": user_question})
    print("Response:", response)
```

Imp. The order is imp in Chain

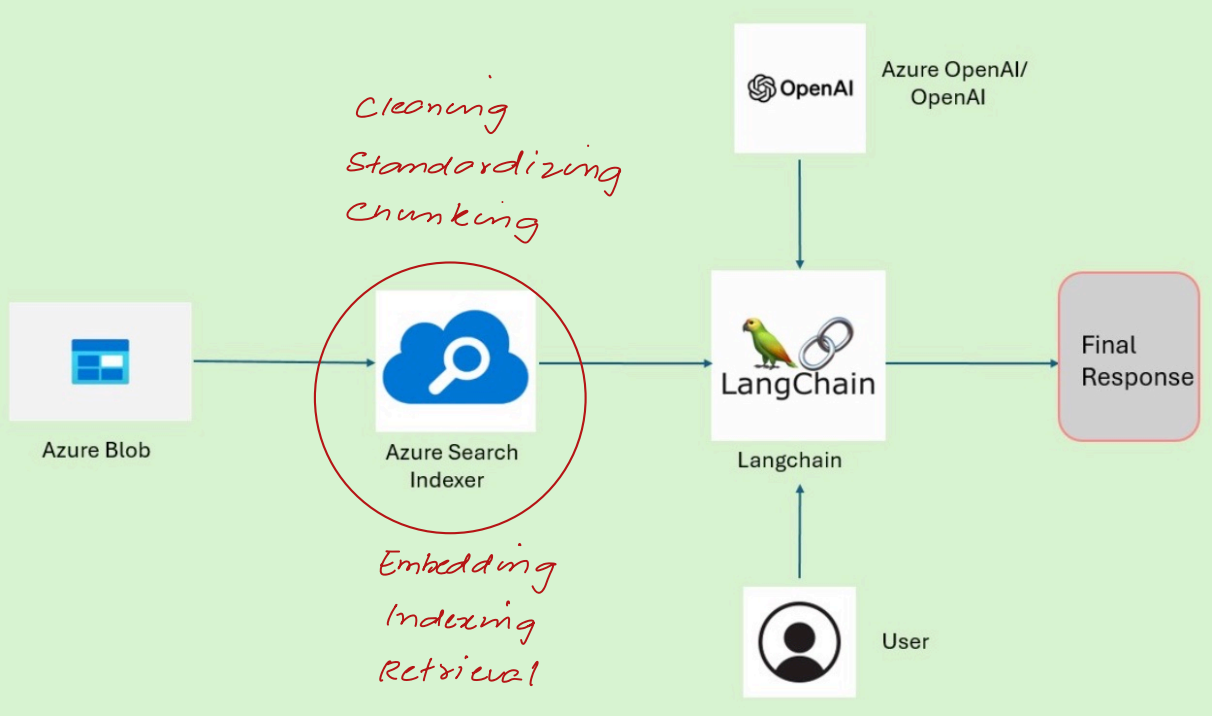
✅ **Rule of thumb:**

Retriever → Prompt → LLM → Parser

1 2 3 4

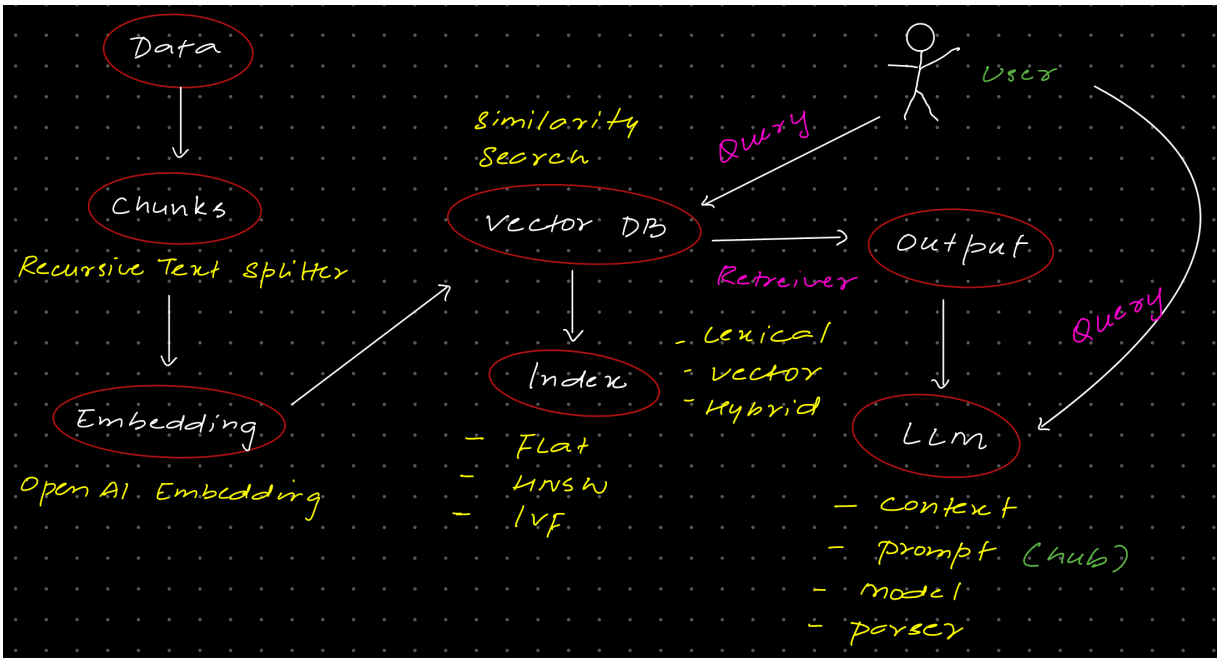
Azure AI Search + LangChain

AI Search makes process really easy.



RAG + LangChain

Complex process



Imp. Azure AI Search makes process easy.

RAG + LLM Evaluation

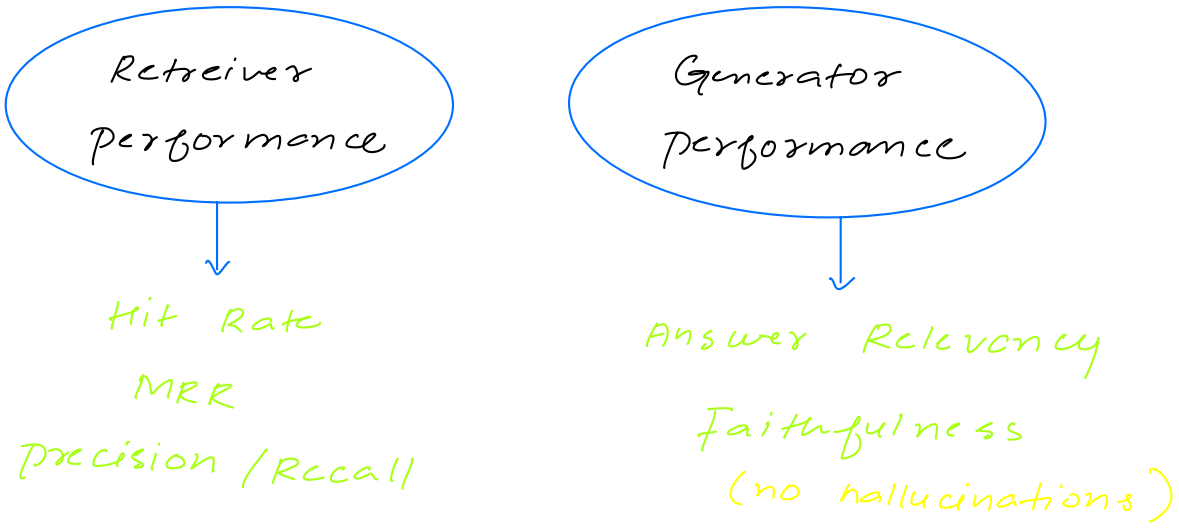
RAGAS = *Retrieval-Augmented Generation Assessment Suite* — a framework to **evaluate retrieval-augmented generation (RAG) systems**.

In a typical RAG pipeline:

- **Retriever** → fetches potentially relevant documents for a query.
- **Generator** (usually an LLM) → uses those documents to produce the final answer.

RAGAS helps track how well each component performs by providing **automatic evaluation metrics**. Many of these rely on LLMs as judges, so you don't need to label everything manually.

Q// what does RAGAS help us Track ?



1. Retriever performance

- **Hit Rate** – Did the retriever return at least one document containing the ground-truth (gold answer)?
- **MRR (Mean Reciprocal Rank)** – How high up was the *first relevant document* ranked?
  - If the first relevant doc is usually at the top, MRR is high (close to 1).
  - If relevant docs appear later, MRR drops.
- **Context Precision/Recall** – Measures how many retrieved docs are truly relevant (precision) and how many relevant docs were missed (recall).

👉 These metrics tell you if the **retriever is surfacing the right context** and putting it near the top.

Actual Answer

Rank of Actual Answer

2. Generator performance

- **Answer Relevancy** – Does the generated answer actually address the question?
  - Checked by comparing the answer with the query (using LLM-as-judge or semantic similarity).
  - High relevancy = the model stays on-topic.
- **Faithfulness** – Is the answer *grounded in retrieved documents* (i.e., no hallucinations)?
  - Checked by verifying whether the answer's statements are supported by the retrieved context.
  - Low faithfulness means the LLM is inventing facts not present in the context.

👉 These metrics tell you if the **generator uses the retrieved context properly** and avoids hallucinations.

LLM judges whether ans. is Relevant

whether context RAG is used.

🔷 Why this matters

- If **MRR is low** → retriever finds relevant docs but ranks them poorly → need better ranking.
- If **Answer Relevancy is low** → generator isn't focusing on the query → prompt/model issue.
- If **Faithfulness is low** → generator hallucinates even when retriever worked → grounding issue.

By combining retriever + generator metrics, RAGAS tells you **where your pipeline is failing**:

- Bad retriever?
- Good retriever but bad generator?
- Or both?

Evaluation Process

🔷 Step 1: You need a benchmark dataset

For evaluation, you can't just run on random queries. You need a **test set** that has:

1. **Question (query)** – what the user would ask.
2. **Gold answer (ground truth)** – the correct, authoritative answer.
   - This is the reference against which you'll compare your model's output.
   - It may come from a curated dataset or be annotated manually.
3. (Optionally) **Gold documents** – the reference passages that actually contain the answer.

Example dataset entry:

```json
{
  "question": "Who is the CEO of Tesla?",
  "ground_truths": ["Elon Musk"],
  "contexts": ["Tesla, Inc. is led by CEO Elon Musk..."],   # retrieved docs
  "answer": "Elon Musk is the CEO of Tesla."                # your model's answer
}
```

Gold Answer

So the **"benchmark"** you compare against is the **gold answers** (and sometimes gold docs) in this dataset.

Imp

🔷 Step 2: How Each Metric Is Checked

Let's use this same example — *"Who is the CEO of Tesla?"* — to understand each metric.

1. Hit Rate (Retriever Metric) → Hit rate is binary

Goal:
Checks whether *any* of the retrieved documents contain the **gold answer**.

True
False

How it's computed:

- For each query:
  - If at least one retrieved document includes "Elon Musk" → Hit = 1
  - Otherwise → Hit = 0
- Average across all queries → **Hit Rate**

Example:
If the retriever fetched the following top-3 docs:

```sql
1️⃣ Tesla designs electric vehicles and energy products.
2️⃣ Elon Musk founded SpaceX and leads Tesla as CEO.   ✅ contains gold answer
3️⃣ Tesla's Gigafactories are spread across the globe.
```

Hit = 1 because one of the top-3 docs mentions "Elon Musk".

Interpretation:
👉 "Did my retriever at least find one useful doc per query?"

2. MRR (Mean Reciprocal Rank, Retriever Metric)

Goal:

Measures *how high* the relevant document appears in the ranked list.

How it's computed:

- For each query, find the **rank (position)** of the first doc containing the gold answer.
  - Reciprocal Rank = 1 / rank
- Average across queries → **MRR**

1/rank = 1/2 = 0.5

Example:

In the same top-3 list:

- The first relevant doc ("Elon Musk founded SpaceX and leads Tesla...") is at **rank 2**.
- Reciprocal Rank = 1 / 2 = 0.5

  If most queries are like this, your **MRR ≈ 0.5**.

Interpretation:

👉 "Does my retriever rank the right docs near the top?"

Higher MRR = better ranking quality.

Azure RAG + LLM Evaluation

- we will have comparison against Golden Answer / Docs or Context in LLM

🔷 Step 3: Evaluating in an Azure OpenAI RAG System

1. **Retriever** – e.g., Azure Cognitive Search or a vector DB.
2. **Generator** – e.g., Azure OpenAI `gpt-4o-mini`.
3. For each query:
   - Run the retriever → get top-k docs (`contexts`)
   - Run the generator → get the LLM answer (`answer`)
   - Collect together with the **gold answer(s)**

Then run **RAGAS evaluation:**

python — Copy code

```python
from ragas import evaluate, EvaluationDataset

eval_dataset = EvaluationDataset.from_list([
    {
        "question": "Who is the CEO of Tesla?",
        "answer": "Elon Musk is the CEO of Tesla.",
        "contexts": ["Tesla, Inc. is led by CEO Elon Musk..."],
        "ground_truths": ["Elon Musk"]
    }
])

results = evaluate(dataset=eval_dataset, llm=evaluator_llm)
print(results)
```

Test Data captured after running { Retriever LLM }

Test Dataset

Output example:

json — Copy code

```json
{
  "hit_rate": 1.0,
  "mrr": 0.9,
  "answer_relevancy": 0.98,
  "faithfulness": 0.95
}
```

🔷 Step 4: Understanding the Scores

Metric	What it measures	Example insight
Hit Rate	Retriever found at least one correct doc	If low → retriever misses key info
MRR	How early the correct doc appears	If low → retriever ranks docs poorly
Answer Relevancy	Whether the answer answers the question	If low → LLM not following query
Faithfulness	Whether answer is supported by context	If low → hallucination issue

Imp. we only need

- Question & Gold Answer

for evaluation