**basic AI technology concepts**

The term AI tends to be thrown around a lot. You've probably heard about machine learning, deep learning, data science, generative AI, and responsible AI. However, it may not be clear what all these terms mean and how they're different to each other.

**What is AI?**

Artificial Intelligence (AI) is the ability of a computer program or machine to **mimic human-like behavior**. For example, to mimic visual senses, speech recognition, decision-making, natural language understanding, and so on. It's not a technology of itself, but rather a goal set by technologists to imitate human intelligence.

**What is generative AI?**

Generative AI is a **subset of AI**. AI can be used to predict outcomes, detect entities, or classify documents, among others. However, generative AI, also known as GenAI, creates content, such as images, videos, or text. The goal is that this AI-generated content should be as useful as any created by humans. This approach is possible thanks to large language models (LLMs), which are complex AI models that can be used for a broad range of use cases.

For example, you may use generative AI to develop follow-up questions to a meeting, create an image from text, or explain the punch line of a joke, even if the joke is in a video.

**What is data science?**

Data science is an **interdisciplinary field** whose aim is to achieve AI. It uses many different techniques, mostly machine learning and statistics. In most cases, data scientists are the experts in charge of solving AI problems.

**What is machine learning?**

Machine learning is a **technique** where a machine sifts through numerous amounts of data to find patterns. This technique is frequently used for AI purposes. Machine learning uses algorithms that train a machine to learn patterns based on differentiating features about the data. The more training data, the more accurate the predictions. Here are some examples:

* **Email spam detection** - Machine learning could look for patterns where email has words like "free" or "guarantee", the email address domain is on a blocked list, or a link displayed in text doesn't match the URL behind it.
* **Credit card fraud detection** - Machine learning could look for patterns like the spending in a zip code the owner doesn't usually visit, buying an expensive item, or a sudden shopping spree.

**What is deep learning?**

Deep learning is a **subset of machine learning**. Deep learning is imitating how a human brain processes information, as a connected artificial neural network. Unlike machine learning, deep learning can discover complex patterns and differentiating features about the data on its own. It normally works with unstructured data like images, text, and audio. That’s why it requires enormous amounts of data for better analysis and massive computing power for speed.

For instance, deep learning can be used to detect cancerous cells in medical images. Deep learning scans every pixel in the image as input to the neural nodes. The nodes analyze each pixel to filter out features that look cancerous. Each layer of nodes pushes findings of potential cancerous cells to the next layer of nodes to repeat the process and eventually aggregate all of the findings to classify the image. For example, the image might be classified as a healthy image or an image with cancerous features.

**What is responsible AI?**

AI has a great disruptive potential. That is why it should follow the highest ethical standards. Responsible AI refers to the **principles and best practices** that ensure AI work is accountable, inclusive, reliable, safe, fair, transparent, secure, and respects privacy.

For instance, AI could create a video that shows a real person at an event they didn't attend in real life. Responsible AI involves not using this technology for deceitful purposes, since it would compromise their privacy and have unfair consequences.

# Microsoft AI approach

AI is disrupting every industry and every business. For the last decade, AI has enabled companies of all sizes to achieve better business results. There's already a **mainstream business use of AI** thanks to these three trends:

* Access to massive amounts of data.
* Access to massive computing power through cloud.
* Access to AI algorithms.

However, AI is now experiencing major breakthroughs. A new generation of LLMs enables new use cases that weren't possible a few years ago, such as those based on high-quality generative AI. Based on these technologies, organizations will experience a **second wave of AI-powered transformation**. However, businesses need an easy way to access the latest AI if they want to take full advantage of it.

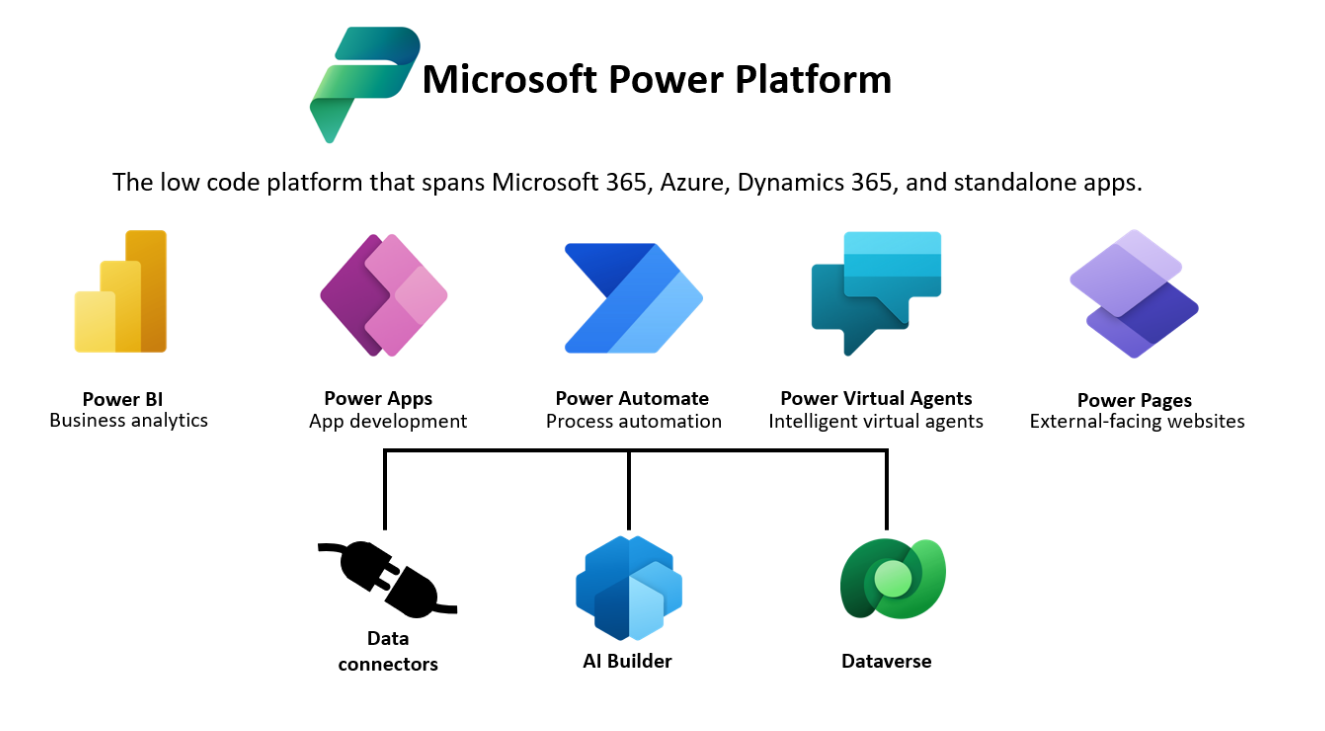
Microsoft is working to democratize AI use. For this, it has designed a wide range of solutions and services to bring AI to everyone, irrespective of their level of AI expertise. There are four approaches, ranged from the level of AI and coding expertise required.

* **AI as copilot for business users**: Microsoft has embedded AI in everyday applications, so business users can benefit from it, even if they don't know anything about coding or data science. In this approach, AI is delivered as a Software as a Service (SaaS) and becomes transparent, that is, it's fully integrated within the provided service without users having to worry about it. For example, Microsoft 365 Copilot incorporates the latest generative AI in the shape of a virtual assistant that performs tasks for you in Microsoft 365 apps.
* **Microsoft Power Platform**: It covers several low-code products that help you build different pieces of applications. These products have a layer of AI, but it's transparent as well and you can benefit from it without handling it directly.
* **Azure AI Services**: These are the solutions for users who want to deliver an AI project but have little data science expertise. They offer premade AI models for you to reuse or customize.
* **Azure Machine Learning**: All machine learning tasks can be handled from this platform. It helps data science teams in setting, automating, and enabling machine learning best practices.

## **Use AI embedded in everyday applications:**

To truly realize the potential of AI, it’s essential to bring AI to every employee in ways that are relevant and meaningful to their work. Microsoft makes this possible by embedding AI in the applications people use in their everyday routine. No code or data science expertise is required because AI is delivered as just another feature of a SaaS product. The result is a wide range of intelligent applications for business users.

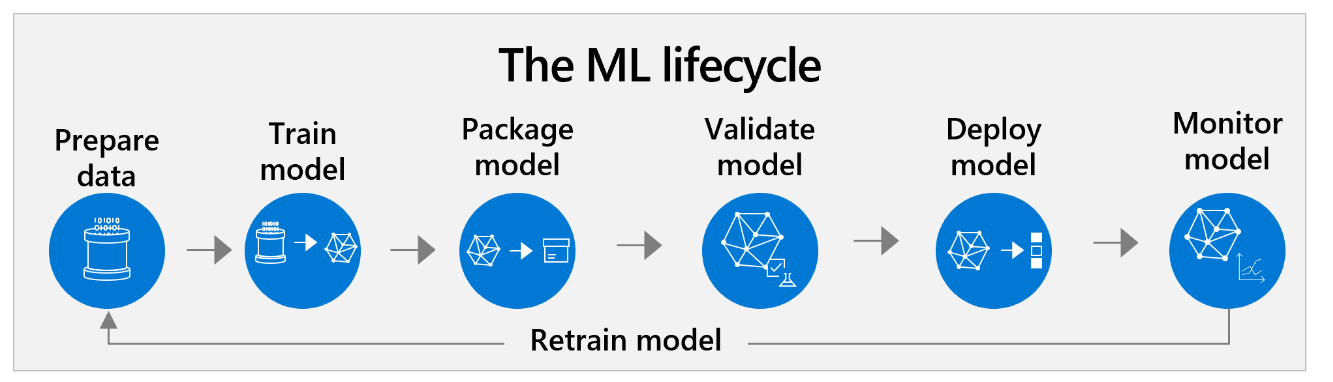
Microsoft Power Platform is a low-code or no-code set of services designed to simplify the process of building solutions. It provides building blocks that help teams work faster. Even if Power Platform isn't centered on AI, its services are often powered by AI and help you create smart solutions.



## Machine learning lifecycle to build your own AI model

This is the classic approach and covers all the usual steps of a data science project. In many scenarios, the resulting AI model performs better than a more generic prebuilt model.

1. **Prepare dataset**. AI starts at data. First, data scientists need to prepare data with which to train the model. This is often the biggest time commitment in the lifecycle. This task involves finding or building your own dataset, cleaning it so it's easily readable by machines, ensuring it's a representative sample, discovering which variables are pertinent for your goal, and so on.
2. **Train and test**. Next, data scientists apply algorithms to the data to train a machine learning model. Then they test it with new data to see how accurate its predictions are.
3. **Package**. A model can't be directly put into an app. It needs to be containerized, so it can run with all the tools and frameworks that have been used in its building.
4. **Validate**. At this point, the team evaluates how model performance compares to their business goals. Testing may have returned good enough metrics, but still the model may not work as expected when used in a real business scenario.
   * **Repeat steps 1-4**. It can take hundreds of training hours to find a satisfactory model. The development team may train many versions of the model by adjusting training data, tuning algorithm hyperparameters, or trying totally different algorithms. Ideally the model improves with each round of adjustment. Ultimately, it's the development team's role to determine which version of the model best fits the business use case.
5. **Deploy**. Finally, they deploy the model in the cloud (often through an API), on an on-premises server, or at the edge on devices like cameras, IoT gateways, or machinery.
6. **Monitor and retrain**. Even if a model works well at first, it needs to be continually monitored and retrained to stay relevant and accurate.



## Machine learning lifecycle using a prebuilt model

Creating your own model from scratch used to be the default option. However, the boom of premade models has changed the paradigm. It's increasingly common to base your data science projects on integrating premade models and adapting them to your business needs. This approach can change the workflow significantly.

1. **Validate**: Using a prebuilt model means organizations start just by checking if it works for them. It's key to understand as soon as possible which premade model to use and how viable it is.
   * **Repeat steps 1-4**: Data scientists repeat steps 1-4 until results are good enough. It may often take some effort for the premade model to deliver what is required.
2. **Engineer prompts**: It's the first option to improve results obtained with a prebuilt model. The team needs to explain what they need so the model understands every nuance. This task involves rephrasing the request (prompt) until the model gets it. It may take time. However, citizen developers and even end users can do prompt engineering if guided by data scientists. This approach gives more power to business users.
3. **Customize dataset**: If prompt engineering doesn't improve results enough, it may be because the prebuilt model is too generic for the intended use case. Then, data scientists need to complement the model with custom training data.
4. **Train and test**: Data scientists can add a custom training layer on top of the premade models with the additional data of step 3. This way, they get a new version of the prebuilt model, tailored to the AI problem they're trying to solve. Another option is to develop a full custom model that covers for the gaps left by the premade one. Many models can coexist within an AI solution.
5. **Package and deploy**: There are different ways to use a prebuilt model. In many cases, it may be enough to use it via API. This approach implies the organization doesn't own it and can't customize it, but it saves the time of packaging and deploying it. If there's been some level of custom training, the AI team needs to package and deploy this new version of the model.
6. **Monitor**: As all models, prebuilt ones also need to be checked regularly to keep its edge. Data scientists should bring themselves be up to date on new prebuilt models. New custom trainings may also be necessary at some point.

## **Discover the business value of applying DevOps practices to machine learning**

the importance of machine learning operations (MLOps). MLOps applies the methodology of DevOps (development and operations) to manage the machine learning lifecycle more efficiently. It enables a more agile, productive collaboration in AI teams among all stakeholders. These collaborations involve data scientists, AI engineers, app developers, and other IT teams.

There are many available products to help teams implement MLOps. Microsoft offers Azure Machine Learning, Azure DevOps, and GitHub.

## **Machine learning operations (MLOps):**

MLOps processes and tools help those teams collaborate and provide visibility through shared, auditable documentation. MLOps technologies provide the ability to save and track changes to all resources, like data, code, models, and other tools

### Model reproducibility

During initial iterative training and later model retraining, there are a few things that can make the complex process more manageable. One of them is to keep models reproducible, which means they can easily be run on the same dataset by any team member with same or similar results. Reproducibility is achieved by documenting processes and sharing resources.

First, it's helpful to centrally **manage assets** like environments, code, datasets, and models so teams can share and reuse them.

* **Model registry**: As teams experiment with different versions of a model, a model registry provides a central place to save each version. With a registry, teams can easily revert to a previous version if something isn't working, even after the solution has gone into production. The model registry also serves as an audit trail for each model's history.
* **Code management**: Technical decision-makers need to determine which technologies and processes their teams will use for code management. This generally includes code repositories like GitHub where code can be saved, versioned, shared, and reused. It also includes tools for using and versioning code.
* **Dataset management**: We also recommend saving training datasets centrally. This way, teams can reuse them, share them with colleagues, or monitor how they change over time to manage drift.
* **Shared environments**: Create model environments that can be shared among individuals. This simplifies the handoff between steps in the model creation process and makes it possible for teams to collaborate on certain steps.

### **Model validation**

Before a model is deployed, it's critical to validate its performance metrics. You may have several metrics that are used to indicate the "best" model. It's important to work with data scientists to understand what metrics are important and evaluate them before deployment. There are **tools to evaluate model metrics**, such as a loss function or a confusion matrix.

Metrics usually compare what the model has predicted with what it should have predicted (the true value or ground truth). Overall, the focus is to maximize true positives and true negatives, that is, the model succeeding in predicting true values. It's equally important to avoid false positives and false negatives, that is, wrong predictions and missed predictions.

It's critical to validate performance metrics against the business use case. For example, perhaps you designed a model to predict patient health. As a healthcare provider dealing with life and death situations, you likely prefer to have false positive diagnoses rather than an incredibly high rate of accuracy that misses diagnoses.

If the model is a newer version of an existing model, you need to see if it performs better than the previous one on key metrics.

### Model deployment

There are several options for deploying the model into production. Data scientists and AI engineers must work together to find out the best option for each case.

* **Cloud**: One option is deploying models using the cloud, often leveraging an application programming interface (API). There are scalable tools to automate and simplify this process, like Kubernetes or Azure Container Instances.
* **On-premises**: Models can also be deployed directly onsite, in the organization's own servers.
* **Edge**: It's also possible to deploy models on edge devices, like cameras, drones, and machinery. This option may be helpful in IoT scenarios.

### Model retraining

Although this is the end of the development process, this is just the beginning of the maintenance cycle. Models need to be monitored and periodically retrained to correct performance issues and take advantage of newer training data. To set yourself up for success, you want to create a retraining loop—or a systematic and iterative process to continually refine and ensure the accuracy of the model.

This process may seem overly complicated. Keep in mind that it can be greatly simplified by using prebuilt models. MLOps tools like Azure Machine Learning don't necessarily need to be populated with custom models, they also accept prebuilt models. In this sense, Azure AI Services is a great alternative, as it offers faster results with less data science expertise required.

# Microsoft Azure AI Fundamentals: Generative AI

Generative AI is a form of artificial intelligence in which models are trained to generate new original content based on natural language input. In other words, you can describe a desired output in normal everyday language, and the model can respond by creating appropriate text, image, or even code output.

Artificial Intelligence (AI) imitates human behavior by using machine learning to interact with the environment and execute tasks without explicit directions on what to output.

Large Language Models:

Generative AI applications are powered by *large language models* (LLMs), which are a specialized type of machine learning model that you can use to perform *natural language processing* (NLP) tasks, including:

* Determining *sentiment* or otherwise classifying natural language text.
* Summarizing text.
* Comparing multiple text sources for semantic similarity.
* Generating new natural language.

Transformers:

LLM are backed by Transformers to understand and respond to the input by Natural Language. Transformers are machine learning models to deal with NLP tasks.

Machine learning models for natural language processing have evolved over many years. Today's cutting-edge large language models are based on the *transformer* architecture, which builds on and extends some techniques that have been proven successful in modeling *vocabularies* to support NLP tasks - and in particular in generating language. Transformer models are trained with large volumes of text, enabling them to represent the semantic relationships between words and use those relationships to determine probable sequences of text that make sense. Transformer models with a large enough vocabulary are capable of generating language responses that are tough to distinguish from human responses.

Transformer model architecture consists of two components, or *blocks*:

* An *encoder* block that creates semantic representations of the training vocabulary.
* A *decoder* block that generates new language sequences.

Steps Involved in Transformer Training:

1. Tokenization:

The first step in training a transformer model is to decompose the training text into tokens - in other words, identify each unique text value. For the sake of simplicity, you can think of each distinct word in the training text as a token (though in reality, tokens can be generated for partial words, or combinations of words and punctuation).

1. Embeddings

Embeddings are numerical representation of tokens stored as multi dimensional index in vector which holds the meaning of tokens. While it may be convenient to represent tokens as simple IDs - essentially creating an index for all the words in the vocabulary, they don't tell us anything about the meaning of the words, or the relationships between them. To create a vocabulary that encapsulates semantic relationships between the tokens, we define contextual vectors, known as embeddings, for them. Vectors are multi-valued numeric representations of information

1. Attention

A machine learning algorithm for NLP. The encoder and decoder blocks in a transformer model include multiple layers that form the neural network for the model.

Attention is a technique used to examine a sequence of text tokens and try to quantify the strength of the relationships between them. In particular, self-attention involves considering how other tokens around one particular token influence that token's meaning.

1. A sequence of token embeddings is fed into the attention layer. Each token is represented as a vector of numeric values.
2. The goal in a decoder is to predict the next token in the sequence, which will also be a vector that aligns to an embedding in the model’s vocabulary.
3. The attention layer evaluates the sequence so far and assigns weights to each token to represent their relative influence on the next token.
4. The weights can be used to compute a new vector for the next token with an attention score. Multi-head attention uses different elements in the embeddings to calculate multiple alternative tokens.
5. A fully connected neural network uses the scores in the calculated vectors to predict the most probable token from the entire vocabulary.
6. The predicted output is appended to the sequence so far, which is used as the input for the next iteration.

# What is Azure OpenAI?

Azure OpenAI Service is Microsoft's cloud solution for deploying, customizing, and hosting large language models. It brings together the best of OpenAI's cutting edge models and APIs with the security and scalability of the Azure cloud platform. Microsoft's partnership with OpenAI enables Azure OpenAI users to access the latest language model innovations.

Azure OpenAI Studio:

Developers can work with these models in Azure OpenAI Studio, a web-based environment where AI professionals can deploy, test, and manage LLMs that support generative AI app development on Azure.

# What are copilots?

The availability of LLMs has led to the emergence of a new category of computing known as copilots. Copilots are often integrated into other applications and provide a way for users to get help with common tasks from a generative AI model. Copilots are based on a common architecture, so developers can build custom copilots for various business-specific applications and services.

Microsoft refers to Microsoft Copilot as first-party, and plugins developed by other companies as third-party copilots.

Microsoft Copilot features can be found throughout commonly used applications. The goal of these features is to empower people to be smarter, more productive, more creative, and connected to the people and things around them.

another example is GitHub Copilot, which provides support to software developers, helping them write, document, and test code.

# Improve generative AI responses with prompt engineering

The quality of responses that a generative AI application returns not only depends on the model itself, but on the types of prompts it's given. The term prompt engineering describes the process of prompt improvement. Both developers who design applications and consumers who use those applications can improve the quality of responses from generative AI by considering prompt engineering.

Prompts are ways we tell an application what we want it to do. An engineer can add instructions for the program with prompts.

Prompt engineering techniques include defining a system message. The message sets the context for the model by describing expectations and constraints, for example, "You're a helpful assistant that responds in a cheerful, friendly manner". These system messages determine constraints and styles for the model's responses.

**Writing good prompts**

You can get the most useful completions by being explicit about the kind of response you want, for example, “Create a list of 10 things to do in Edinburgh during August”. You can achieve better results when you submit clear, specific prompts.

**Providing examples**

LLMs generally support zero-shot learning in which responses can be generated without prior examples. However, you can also provide one-shot learning prompts that include one, or a few, examples of the output you require such as, “Visit the castle in the morning before the crowds arrive”. The model can then generate further responses in the same style as the examples provided in the prompt.

**Grounding data**

Prompts can include grounding data to provide context. You can use grounding data as a prompt engineering technique to gain many of the benefits of fine-tuning without having to train a custom model.

To apply this technique, include contextual data in the prompt so that the model can use it to generate an appropriate output. For example, suppose you want to use an LLM to generate a summary of an email. You can include the email text in the prompt with an instruction to summarize it.