MODULE 2 · STUDY NOTES

Introduction to Machine Learning

Where machine learning fits inside AI, the three types of learning, the iterative ML pipeline, and the Python tools and AWS services that build and run models

Machine learning is the center of gravity for this whole course, and this module puts it on the map before the deep dives begin. **Artificial intelligence** is the broad branch of computer science aimed at building machines that do tasks humans typically do; **machine learning** is the subset that, instead of following hand-written instructions, studies algorithms and statistical models that learn a task by *inference* from data; **deep learning** is the ML subdomain built on brain-inspired neural networks; and **generative AI** is the slice of deep learning that produces new content from pretrained foundation models. From there the module turns practical: the everyday business problems ML solves and the three learning types that solve them, the iterative pipeline that converts a business problem into a deployed model, the Python tools and AWS services a data scientist actually reaches for, and the data, people, business, and technology challenges every project must clear — including the reminder that not every problem needs machine learning at all.

LEARNING OBJECTIVES

- Recognize how machine learning (ML), deep learning, and generative AI are part of artificial intelligence (AI)
- Describe artificial intelligence and machine learning terminology
- Identify how machine learning can be used to solve a business problem
- Describe the machine learning process
- List the tools available to data scientists
- Identify when to use machine learning instead of traditional software development methods

2.1 What is machine learning?

AI is the umbrella. Artificial intelligence is a broad branch of computer science concerned with building machines that can perform tasks a human would typically perform. **Machine learning is a subset of AI**, **deep learning** is a subdomain of machine learning, and **generative AI** is a subset of deep learning. AI research spans many fields at once — natural language processing, reasoning, knowledge representation, learning, perception, and interaction with the physical environment — and a machine that could learn or understand *any* task a human can is called **artificial general intelligence (AGI)**, which does not yet exist.

Machine learning is the scientific study of algorithms and statistical models that perform a task using inference instead of instructions. Consider a spam filter. Without ML you would hand-write a sprawling series of if/else rules — words in the subject or body, the number of links, the message length — an approach that is laborious and never covers every case. With ML you instead feed the program a list of messages already marked *spam* or *not spam*; the model learns which patterns predict spam, and when it sees a brand-new message it predicts the label on its own.

WORKED EXAMPLE – TOM MITCHELL'S DEFINITION APPLIED TO SPAM

ML pioneer Tom Mitchell defined learning this way: *a computer program learns from experience E with respect to a class of tasks T and a performance measure P if its performance at T, as measured by P, improves with experience E.* For the spam filter, **E** is the collection of email messages marked spam or not, **T** is the task of identifying spam, and **P** is the probability that an unseen message is spam. As the program sees more examples (E), it gets better at the task (T) as scored by P — and that improvement *is* learning.

Deep learning takes its inspiration from the brain. An **artificial neural network (ANN)** is built from artificial neurons, each with one or more inputs and a single output that *fires* based on a transformation of its inputs. Neurons are arranged in **layers** — an input layer, one or more hidden layers, and an output layer — with the output of each neuron connecting to the inputs of every neuron in the next layer. The input layer is populated from the training data; activations ripple through the layers until an answer appears at the output layer, and its accuracy is measured. If the result misses the threshold, training repeats with slight changes to the **weights** of the connections, strengthening the connections that lead to success and weakening those that lead to failure. Where classic ML practitioners spend much of their time selecting features and models, deep learning practitioners spend theirs designing different ANN architectures.

Generative AI is a type of AI that can create new content and ideas — conversations, stories, images, videos, and music. It is powered by very large ML models called **foundation models (FMs)** that are pretrained on vast amounts of data. Unlike traditional models that are single-task oriented and must be trained for their one job, a single FM can serve many tasks — text generation, summarization — *without further training*. Its common applications improve customer experience: chatbots, virtual assistants, intelligent contact centers, personalization, and content moderation.

WHY ML TOOK OFF WHEN IT DID

- Mainstream ML is recent — rapid advances began in the **mid-2000s**, driven by **Moore's law** and the rise of **cloud computing**, which made compute and storage larger, faster, and cheaper (rent power for pennies instead of buying clusters).
- In **2012**, neural networks entered the **ImageNet** Large Scale Visual Recognition Challenge and pushed image-recognition accuracy to about **82%**; it kept climbing and **exceeded human performance in 2015**.
- In **2017**, the **transformer architecture** breakthrough helped make modern generative AI possible.

KEY TERMS

Artificial intelligence (AI)
Broad branch of computer science for building machines that perform tasks a human would typically perform

Machine learning (ML)
A subset of AI that performs a task using inference from data instead of explicit instructions

Deep learning
A subdomain of ML built on layered artificial neural networks inspired by the brain

Generative AI
A subset of deep learning that creates new content, powered by pretrained foundation models

Foundation model (FM)
A very large, pretrained ML model that a single instance can apply to many tasks without further training

COMMON PITFALL

Keep the nesting straight: generative AI is **not** outside machine learning. It is a subset of *deep learning*, which is a subdomain of *ML*, which is a subset of *AI*. On a diagram they are concentric circles, not separate boxes.

2.2 Business problems solved with machine learning

Machine learning already runs through everyday digital life: **spam filters** trained on spam and regular email, **recommendations** trained on other people's reading and buying habits, and **credit-card fraud** detection trained on fraudulent and legitimate transactions. Other examples include grouping photos by facial detection, spotting brain tumors in scans, and finding anomalies in X-rays. Machine learning has **three main types**, and knowing them guides you toward algorithms that fit the business problem.

Supervised learning is the most widely applicable type — and the most common in business. It needs a *supervisor*: labeled training data that shows the model the right answers (*here is a car in this picture; here is a car in another*), so it can identify the pattern in a new, unseen image. Supervised problems split into two categories. **Classification** sorts an observation into categories — *binary* classification chooses between two (fraud or not fraud), while *multiclass* classification chooses among three or more (routing a caller to the correct support department). **Regression** instead maps inputs to a **continuous value**: predicting that a stock will move from $113 to $127 a share is a regression problem.

Unsupervised learning applies when the data is all you have and no supervisor is available. Labels are *not* provided — because you don't yet know the variables and patterns — so the **machine must uncover and create the labels itself**, detecting emerging properties across the whole dataset. Its common subcategory is **clustering**, which groups data by similar features: analyzing purchasing habits, an unsupervised algorithm might sort customer companies into *small*, *medium*, and *large* tiers, revealing groupings you were previously unaware of and informing different marketing strategies.

Reinforcement learning learns through **trial and error**, continuously improving by mining feedback from previous iterations. An **agent** interacts with an **environment**; when it takes an **action**, it receives a **reward or penalty** depending on whether that behavior should be reinforced or discouraged. It is the right choice when the desired outcome is known but the path to it is not. In **AWS DeepRacer**, the agent is a virtual car, the environment is a virtual racetrack, the actions are throttle and steering inputs, and rewards incentivize the car to complete the track quickly without leaving it.

TABLE 2.1 The three types of machine learning

TYPE	HOW IT LEARNS	PROBLEM TYPES	EXAMPLE APPLICATIONS
Supervised	From already-labeled data (known inputs and outputs)	Classification (binary, multiclass), regression	Fraud detection, image recognition, medical diagnostics, weather and market forecasting
Unsupervised	Finds patterns in unlabeled data; creates its own labels	Clustering, data-structure discovery	Customer segmentation, targeted marketing, visualization, gene sequencing
Reinforcement	Trial and error via rewards and penalties in an environment	Sequential control toward a known goal	Game AI, robotics, customer-service routing, AWS DeepRacer

Several application areas cut across these types. **Computer vision (CV)** is a large field of mostly *classification* problems (*is this a cat or a dog? does this X-ray show a tumor?*); often built with deep learning, it lets machines identify people, places, and things in images at or above human accuracy, working from single images, video sequences, multiple cameras, or 3D data. **Natural language processing (NLP)** handles speech and written text and powers chat and call-center bots, translation tools, voice-to-text, and sentiment analysis — Amazon Alexa uses NLP. **Self-driving vehicles** combine many machine- and deep-learning models: they continuously detect the environment and forecast changes through *object detection*, feeding those outputs into systems that control the car — and some situations, such as a pedestrian stepping out from behind an obstacle, demand real-time responses with no latency or room for error.

CLASSICAL PROGRAMMING IS ENOUGH WHEN	MACHINE LEARNING FITS WHEN YOU HAVE
the logic is knowable	*check first — basic programming sometimes wins*
▪ Business logic and procedures are clear ▪ Rules can be enumerated as if/else statements ▪ The number of variables is limited ▪ Basic programming solves it just as well	▪ Large datasets and a large number of variables ▪ No clear procedure to obtain the solution ▪ Existing ML expertise and infrastructure ▪ Management support to build and sustain it

COMMON PITFALL

A common mix-up: unsupervised learning is *not* handed labels to find. It receives unlabeled data and must **uncover and create the labels itself** — which is exactly why you turn to it when you don't already know the variables and patterns. And remember that classification (both binary and multiclass) and regression are **both supervised**; they differ only in the output — discrete categories versus a continuous value.

2.3 The machine learning process

The **ML pipeline** turns a business problem into a deployed model, and much of it is iterative. It always starts with the **business problem** — something you or your team believes ML can help — followed by **problem formulation**: articulating that business problem and converting it into an ML problem.

Data preparation comes next. You extract data from one or more sources, reconcile the differences in how each represents things, and use visualization and statistics to judge whether the data is consistent enough to use. In ML data, **columns are features and rows are instances**. Real data is messy: a value like `11/2/1969` could be November 2 or February 11, and a stray `Male` in a country column might be a shifted cell or the capital of the Maldives — ambiguities that sometimes require a **subject matter expert (SME)** to resolve. Consistent, correct data has a substantial effect on a project's success.

WORKED EXAMPLE – CLEANING A MERGED DATASET

Three source systems are merged into one table with columns for name, country, sex, and date of birth, and the rows disagree: one date reads `18/2/1972`, another is just `37`, and a sex value has slipped into the country column. Before this data can train a model it must be reconciled into a single cohesive view — deciding what each ambiguous value means (often with an SME), standardizing formats, and flagging missing entries. Only then does feature engineering begin.

Feature engineering selects or creates the **features** (columns) the model will train on, because the algorithm uses features to predict the **target**. This can mean adding, removing, or calculating features: cleaning up the name column (or dropping it if the problem doesn't need it); converting country into numeric columns — but splitting it into multiple

binary columns rather than a single ISO code, so the model doesn't read `44` as *greater than* `01` (a technique called **categorical encoding**); converting sex to a 0/1 binary; and splitting date of birth into component parts such as age, birth month, and day of week when those might affect the target.

Model training follows. You don't train on all the data — you hold some back to test with, typically training on about **80%** and reserving the rest. The training data plus a chosen algorithm (the module uses **XGBoost**) produce a **trained model**, and the values you set to alter how the algorithm works are called **hyperparameters**. **Evaluation and tuning** then use the held-out **test data**: feed the model instances it hasn't seen, compare its predictions against the known targets, calculate **metrics**, and adjust the data, features, or hyperparameters until the results are good enough. Because you loop through feature engineering, training, evaluating, and tuning repeatedly, the process is fluid rather than linear.

OVERFITTING	UNDERFITTING
memorized, can't generalize ▪ Performs well on the training data ▪ Performs poorly on new or evaluation data ▪ The model memorized instead of learning the pattern	*never captured the pattern* ▪ Performs poorly even on the training data ▪ Can't capture the relationship between inputs (X) and target (Y) ▪ Diagnose by comparing error on training vs. evaluation data

Once you have retrained the model and are satisfied with the results, it is **deployed** to deliver the best possible predictions on live data. Knowing this pipeline matters even when AWS managed services do most of the heavy lifting — the same phases run underneath. In broad strokes the iterative core reduces to three repeating steps: **data processing, model training, and model evaluation.**

> **COMMON PITFALL**
>
> Don't swap the fit terms. **Overfitting** means strong on training data but weak on new data (the model memorized); **underfitting** means weak even on the training data (it never captured the input-to-target relationship). You find the root cause by comparing prediction error on the training data against the evaluation data.

2.4 Machine learning tools overview

Python is the most popular language for machine learning, and a data scientist's toolkit runs from individual libraries up through fully managed services. At the library level, work usually happens inside **Jupyter Notebook** — an open-source web application for documents that mix live code, equations, visualizations, and narrative text — or **JupyterLab**, its flexible, extensible web-based development environment. Amazon SageMaker hosts both.

KEY TERMS

pandas
Open-source Python library for data handling and analysis; represents data in a spreadsheet-like table called a DataFrame

NumPy
Fundamental scientific-computing package: N-dimensional arrays plus math such as linear algebra, Fourier transforms, and random numbers

Matplotlib
Library for creating static, animated, and interactive scientific visualizations in Python

Seaborn
Data-visualization library built on Matplotlib, offering a high-level interface for statistical graphics

scikit-learn
Open-source ML library for supervised and unsupervised learning, with tools for model fitting, preprocessing, selection, and evaluation; built on NumPy, SciPy, and Matplotlib

Moving up a level, production-ready **frameworks** provide tools and code libraries, integration with AWS services, and a developer community. Common ones include **TensorFlow** (machine learning), **Keras** (deep learning), **PyTorch**, and **Apache MXNet**, alongside Caffe2, Gluon, CNTK, Torch, and Chainer — all supported on AWS and usable from SageMaker. AWS also offers ML-tuned **infrastructure**: EC2 **P3** and **C5/C5n** instances optimized for training and inference, **AWS IoT Greengrass** for building ML on IoT and edge devices, and **Amazon Elastic Inference** to reduce the cost of running ML applications, plus prepackaged AMIs bundling popular frameworks.

TABLE 2.2 Amazon SageMaker capabilities

CAPABILITY	WHAT IT DOES
Ground Truth	Set up and manage labeling jobs for highly accurate training datasets, using active learning and human labeling
Notebook	AWS and SageMaker SDKs and sample notebooks to create training jobs and deploy models
Training	Train and tune models at any scale using high-performance AWS algorithms or your own
Inference	Create models from training jobs or import external models for hosting, then run inferences on new data
AWS Marketplace	Find, buy, and deploy ready-to-use model packages, algorithms, and data products

At the top of the stack sit **managed ML services** that require **no ML experience** — you integrate them through APIs and supply your own data to specialize them for your problem domain.

TABLE 2.3 Managed AI/ML services by task

TASK	SERVICES	WHAT THEY DO
Computer vision	Rekognition, Textract	Rekognition does object and facial recognition in images and video; Textract extracts text from images
Speech	Polly, Transcribe	Polly turns text into speech; Transcribe converts spoken audio into text
Language	Comprehend, Translate	Comprehend uses NLP to find insights and relationships in text; Translate converts text between languages
Chatbots	Amazon Lex	Builds interactive conversational applications that use voice or text
Forecasting	SageMaker Canvas	No-code ML environment that combines time-series data with other variables to build forecasts
Recommendations	Amazon Personalize	Creates individualized, personalized recommendations for customers

THE AMAZON ML STACK IN THREE LAYERS

- **ML frameworks & infrastructure** — low-level libraries (TensorFlow, PyTorch, Keras, MXNet) and ML-tuned compute (EC2 P3, C5/C5n) for maximum control.
- **Amazon SageMaker** — a managed environment spanning the full lifecycle: labeling, notebooks, training, inference, and Marketplace models.
- **Managed AI services** — pre-trained services (Rekognition, Polly, Comprehend, Lex, Personalize…) that add AI with no ML expertise required.

COMMON PITFALL

Managed services are **not** for ML experts only — they are the opposite: pre-trained services you call through an API with *no ML experience*, supplying just your own data. Reserve low-level frameworks and SageMaker for when you genuinely need to build or tune your own models.

2.5 Machine learning challenges

Every ML project runs into obstacles, and they cluster into four areas — **data, business, users (people), and technology**. The problems you most directly influence are about **data**: the world is full of poor-quality and inconsistent data, and much of the work is simply getting *good* data that represents the problem, exists in sufficient quantity, and avoids overfitting or underfitting.

TABLE 2.4 Four categories of ML challenges

AREA	TYPICAL CHALLENGES
Data	Poor quality, non-representative, or insufficient data; overfitting and underfitting
Business	Complexity of formulating the question as an ML problem; explaining the model to the business (an unexplained model may not be adopted); cost of building systems
Users (people)	Lack of data-science expertise; cost of staffing data scientists; lack of management support
Technology	Data-privacy and regulatory issues; complicated tool selection; integration with other systems

One way through many of these challenges is to **not build from scratch**. AWS **managed services** solve many ML problems with little ML knowledge — you just need developer skills to call their APIs. You can also adapt popular pre-built models such as **YOLO (You Only Look Once)** for vision, or turn to **AWS Marketplace**, which offers **over 250** ML model packages and algorithms across **more than 14** industry segments, including third-party solutions from independent software vendors.

TAKEAWAYS FOR CHOOSING AN APPROACH

- The biggest challenges you directly influence are **data** challenges — quality, representation, and quantity.
- People, business, and technology challenges (expertise, cost, explainability, privacy, integration) can decide whether a project succeeds or is even adopted.
- **Managed services and existing models simplify ML** — prefer them when they fit before committing to a custom build.

COMMON PITFALL

Not every problem should be solved with machine learning. When the business logic is clear and the variables are limited, **basic programming can work equally well** — and ML also demands data, expertise, infrastructure, and management support to sustain. Choose ML to fit the problem; don't bend the problem to fit ML.

Module summary

- AI is the broad field of building machines that do human tasks; **ML is a subset of AI**, deep learning a subdomain of ML, and generative AI a subset of deep learning — concentric, not separate.
- ML performs a task using **inference instead of instructions**: rather than hand-coding rules, you train a model on data so it learns the patterns and predicts on unseen input.
- Deep learning uses brain-inspired **artificial neural networks** (input, hidden, and output layers); generative AI adds pretrained **foundation models** that serve many tasks without retraining — made practical by cloud computing, Moore's law, and the 2017 transformer.
- The three types of ML: **supervised** (labeled → classification and regression), **unsupervised** (unlabeled → clustering, machine creates labels), and **reinforcement** (trial and error via rewards and penalties). Most business problems are supervised.
- The **ML pipeline** — business problem → data preparation → feature engineering → training → evaluation & tuning → deployment — is iterative; train on about 80% and test on the rest. Watch fit: **overfitting** is strong on training data but weak on new data; **underfitting** is weak even on training data.

- The tools ladder climbs from **Python** libraries (pandas, NumPy, scikit-learn) and Jupyter, to **frameworks** (TensorFlow, PyTorch) and ML-tuned EC2, to **Amazon SageMaker** for the full lifecycle, to **managed services** that need no ML experience — and ML challenges cluster into data, business, people, and technology, with not every problem needing ML at all.

Review questions

1. How do AI, ML, deep learning, and generative AI relate to one another?

Answer: They nest: AI is the broad field of machines doing human tasks; machine learning is a subset of AI that learns tasks from data by inference; deep learning is a subdomain of ML built on layered neural networks; and generative AI is a subset of deep learning that creates new content using pretrained foundation models.

2. What does it mean that ML performs a task 'using inference instead of instructions,' using the spam filter as an example?

Answer: Instead of hand-coding if/else rules to catch spam, you train a model on messages already labeled spam or not spam; it infers the patterns that predict spam and then classifies new, unseen messages on its own.

3. Name the three types of machine learning and the kind of data or feedback each relies on.

Answer: Supervised — already-labeled data (classification and regression). Unsupervised — unlabeled data, where the machine creates its own labels (clustering). Reinforcement — rewards and penalties from an environment, learned by trial and error.

4. Within supervised learning, how do classification and regression differ, and how do binary and multiclass classification differ?

Answer: Classification predicts discrete categories; regression predicts a continuous value (like a stock price). Binary classification chooses between two categories (fraud or not fraud); multiclass chooses among three or more (for example, routing to one of many support departments).

5. List the stages of the ML pipeline in order, and identify which part is iterative.

Answer: Business problem (with problem formulation) → data preparation → feature engineering → model training → evaluation and tuning → deployment. The middle — feature engineering, training, evaluating, and tuning (data processing, training, and evaluation) — loops repeatedly until the model meets the goal.

6. Distinguish overfitting from underfitting, and note how you diagnose them.

Answer: Overfitting: the model does well on training data but poorly on new or evaluation data because it memorized rather than generalized. Underfitting: it does poorly even on the training data because it never captured the input-to-target relationship. You diagnose both by comparing prediction error on the training data versus the evaluation data.

7. A team wants to add facial recognition to an app but has no ML expertise. What AWS options fit, and when would you instead build your own model?

Answer: Use a managed service — Amazon Rekognition for image and video recognition — which needs no ML experience and is called through an API; you could also adapt an existing model such as YOLO or buy from AWS Marketplace. Build your own with frameworks and SageMaker only when you have large datasets, unclear rules, ML expertise, supporting infrastructure, and management support.