CS2AI Artificial Intelligence

Lecture 01: AI & ML Overview

Lecture Notes

Table of Contents

Session 1: AI Definitions and Types

1.1 Introduction and Module Overview

The CS2AI Artificial Intelligence module provides a rigorous introduction to the foundational concepts of artificial intelligence (AI) and machine learning (ML), equipping students with both theoretical understanding and practical skills in Python-based data science. The module is delivered through weekly two-hour lectures and accompanying practical laboratory sessions. Assessment comprises a group coursework component (50%) and an online examination (50%). The curriculum spans twelve weeks, covering topics from AI fundamentals and reinforcement learning through to supervised and unsupervised learning algorithms, deep learning, and model interpretation.

This first lecture establishes the conceptual groundwork for the entire module. By the end of the session, students should be able to explain the foundational definitions of AI and ML, discuss and differentiate the main AI and ML classifications while considering social and ethical aspects, critically analyse typical ML problems (such as data bias, data leakage, class imbalance, and overfitting), and implement exploratory data analysis solutions in Python as part of a machine learning modelling pipeline. These learning outcomes emphasise not merely knowing what AI is, but understanding why its definitions matter and how they shape the design of intelligent systems.

1.2 The Turing Test: Can Machines Think?

The question 'Can machines think?' was posed by Alan Turing in his seminal 1950 paper 'Computing Machinery and Intelligence'. Rather than attempting to define 'thinking' philosophically, Turing proposed an operational test — the Imitation Game, now universally known as the Turing Test. In this test, a human judge communicates via text with both a human and a machine. If the judge cannot reliably distinguish the machine from the human, the machine is said to exhibit intelligent behaviour. Importantly, Turing viewed physical simulation of a person (i.e., 'acting' humanly in appearance) as unnecessary to demonstrate intelligence; what mattered was the quality of the machine's responses.

To pass the Turing Test, a machine would require several capabilities: natural language processing (to communicate fluently in a human language), knowledge representation (to store and retrieve information about the world), automated reasoning (to draw conclusions and answer novel questions), and machine learning (to adapt to new patterns and generalise from experience). For the 'Total Turing Test', which extends the challenge to include physical interaction, computer vision and robotics would also be necessary. The judge might assess the machine on criteria such as creativity, empathy, natural language use, ethical reasoning, and the relevance of its responses.

In everyday life, the Turing Test surfaces in the form of CAPTCHAs — Completely Automated Public Turing tests to tell Computers and Humans Apart — which use visual or interactive challenges that are easy for humans but difficult for automated bots. However, the Turing Test has attracted criticism. As Russell and Norvig (2022) note, aeronautical engineers do not define the goal of their field as making 'machines that fly so exactly like pigeons that they can fool even other pigeons'. The point is that mimicking human behaviour is not the same as achieving genuine intelligence; what matters is whether the system achieves its objectives effectively.

1.3 Four Approaches to Defining AI

Russell and Norvig (2022) identify four historically influential approaches to defining artificial intelligence, organised along two dimensions: whether the focus is on thought processes or behaviour, and whether the benchmark is human performance or an ideal standard of rationality. These four approaches have shaped the evolution of the field and continue to inform debates about what AI should aspire to achieve.

Approach	Focus	Methodology
Thinking Humanly (Cognitive Modelling)	Replicating human thought processes	Introspection, psychological experiments, brain imaging
Acting Humanly (Turing Test)	Behaving indistinguishably from a human	Natural language, knowledge representation, reasoning, learning
Thinking Rationally (Laws of Thought)	Correct logical inference	Syllogisms, formal logic, the logicist tradition
Acting Rationally (Rational Agent)	Taking the best possible action given available information	Agent-based design, optimisation, decision theory

The cognitive modelling approach seeks to build machines that think the way humans do, using evidence from psychology and neuroscience. The Turing Test approach, as discussed above, focuses on observable behaviour. The laws-of-thought approach draws on formal logic to build systems that reason correctly. However, the approach that has come to dominate modern AI research is the rational agent approach, which we examine next.

1.4 The Standard Definition: Acting Rationally

Contemporary AI has converged on the study and construction of agents that 'do the right thing'. An agent is simply something that acts — the word derives from the Latin agere, meaning 'to do'. A rational agent is one that acts so as to achieve the best outcome or, when there is uncertainty, the best expected outcome. What counts as 'the right thing' is defined by the objective we provide to the agent. This framing is remarkably general and has parallels across several disciplines.

Discipline	Terminology	Objective
Control Theory	Controller	Minimise a cost function
Operations Research	Policy	Maximise a sum of rewards
Statistics	Decision rule	Minimise a loss function
Economics	Decision maker	Maximise utility or social welfare

The acting-rationally definition has two key advantages over its alternatives. First, compared with the Turing Test (acting humanly), it is more amenable to scientific development: we can reason precisely about what constitutes optimal behaviour without needing to replicate every quirk of human cognition. Second, compared with the laws-of-thought approach (thinking rationally), acting rationally is more general: rational behaviour can arise from mechanisms other than logical inference, such as reflex actions and learned heuristics.

Key Concept — Rational Agent

For each possible percept sequence, a rational agent selects an action that is expected to maximise its performance measure, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has. Rationality requires information gathering, learning, and autonomy.

1.5 PEAS and Task Environments

Designing an agent requires specifying its task environment. Russell and Norvig formalise this using the PEAS framework, which captures the four essential elements of any agent design problem.

- Performance measure — the criterion by which the agent's success is evaluated (e.g., for a self-driving car: safety, travel time, fuel efficiency).
- Environment — the external world in which the agent operates (e.g., roads, traffic, weather conditions).
- Actuators — the mechanisms through which the agent acts upon the environment (e.g., steering, braking, acceleration).
- Sensors — the mechanisms through which the agent perceives the environment (e.g., cameras, LIDAR, GPS).

Different task environments exhibit different properties that profoundly affect agent design. An environment may be fully observable or partially observable, deterministic or stochastic, episodic or sequential, static or dynamic, discrete or continuous, and single-agent or multi-agent. For example, chess is fully observable, deterministic, and sequential, whereas autonomous driving is partially observable, stochastic, dynamic, and continuous. Understanding these properties is essential for selecting appropriate AI techniques.

1.5.1 The Vacuum-Cleaner Agent: An Illustrative Example

To ground the concept of rationality, the lecture introduces a simple vacuum-cleaner agent operating in a two-cell world. The agent can perceive whether its current cell is clean or dirty and whether it is in the left or right cell. Its available actions are: move left, move right, or suck (clean). The question is: is this agent rational?

To answer this, we must define a performance measure (e.g., maximise the number of clean cells over time) and consider the agent's percept sequence — the complete history of everything it has perceived. An agent is rational if, for each possible percept sequence, it selects an action that is expected to maximise its performance measure, given the evidence from its percepts and its built-in knowledge. A state-space graph of this simple environment enumerates all possible world states and the transitions between them, providing a complete picture of the agent's decision landscape.

Two additional properties are crucial for a rational agent. First, learning: after experiencing an action–percept pair, the agent should adjust its behaviour to perform better next time. Second, autonomy: the agent should learn to compensate for partial or incorrect prior knowledge, rather than relying solely on what its designer has hard-coded.

1.6 Narrow AI vs General AI

A fundamental distinction in AI is between Artificial Narrow Intelligence (ANI) and Artificial General Intelligence (AGI). Narrow AI — sometimes called weak AI — refers to systems designed to perform a single, specific task, potentially at superhuman levels. Examples include chess-playing algorithms, facial recognition systems, self-driving cars, web search engines, disease detection systems, and recommendation algorithms. Every commercially deployed AI system today is a form of narrow AI.

General AI, by contrast, would possess the ability to understand, learn, and apply intelligence across a wide range of tasks — functioning, in essence, as a human mind. AGI remains a subject of science fiction rather than engineering reality: familiar portrayals include HAL 9000 in 2001: A Space Odyssey (1968), the Replicants in Blade Runner (1982), the Terminator franchise (1984–2019), and Samantha in Her (2013). While recent large language models (LLMs) such as ChatGPT have demonstrated impressive generality, they remain fundamentally narrow systems — sophisticated pattern matchers trained on specific objectives — rather than genuinely general intelligences.

1.7 Benchmarking AI Capabilities

How do we measure the progress of AI systems? One prominent benchmark is SWE-bench Verified, a human-validated subset of the SWE-bench dataset consisting of 500 samples that evaluate AI models' ability to solve real-world software engineering issues. Each sample is derived from an actual GitHub issue in one of twelve open-source Python repositories. As of the data presented in the lecture, the best-performing AI models achieve a maximum accuracy of approximately 65% on this benchmark — impressive, but far from human-level reliability.

The lecture also highlights a notable 2025 event: at the AtCoder World Tour Finals, a human competitive programmer ('Psyho') defeated OpenAI's LLM in a 10-hour heuristic programming competition — a reminder that, despite rapid progress, LLMs have not yet surpassed expert human performance across all cognitive tasks. Benchmarks such as SWE-bench provide a valuable, reproducible way to track AI progress over time and to identify the specific capabilities that remain challenging for current systems.

Key Concept — AI Classification

Artificial Narrow Intelligence (ANI) performs a single task, potentially better than humans (e.g., chess engines, facial recognition). Artificial General Intelligence (AGI) would function as a human mind across arbitrary tasks but does not yet exist. All current deployed AI systems, including large language models, are forms of ANI.

Session 2: AI Social and Ethical Aspects

2.1 The Sustainability Challenge of AI

The rapid growth of AI — and large language models in particular — raises pressing questions about environmental sustainability. The lecture presents data on notable AI systems across domains, illustrating an exponential increase in the computational resources (measured in petaflops, where 1 petaflop equals 10^{15} floating-point operations) required to train state-of-the-art models. This growth carries direct consequences in terms of energy consumption, water usage for cooling data centres, carbon emissions, and electronic waste.

The energy cost of AI manifests at two levels. At the training level, large language models such as GPT-4 require enormous computational budgets measured in thousands of GPU-hours, translating to significant electricity consumption and associated carbon emissions. At the inference level, each individual request to an LLM consumes substantially more energy than a conventional web search query. However, research shows that compact LLMs can reduce energy usage by up to 90% compared with their full-sized counterparts, suggesting that efficiency-oriented design choices can meaningfully mitigate the environmental footprint of AI deployment.

Key Concept — AI Sustainability

The computational cost of training and deploying AI models has grown exponentially. Each LLM request consumes significantly more energy than a standard web search. However, compact and efficient model architectures can reduce energy consumption by up to 90%, making sustainable AI design both technically feasible and ethically imperative.

2.2 AI Opportunities and Risks

The UK Government's 2023 White Paper on AI regulation ('A pro-innovation approach to AI regulation') provides a balanced assessment of AI's potential benefits and dangers. AI presents transformative opportunities across numerous sectors, while simultaneously introducing risks that must be carefully managed.

Category	Examples
Scientific Discovery	Piecing together the first complete image of a black hole; solving the decades-old protein-folding puzzle (AlphaFold); accelerating discovery of new medicines
Healthcare	Deep learning AI improving breast cancer screening accuracy
Agriculture	AI-powered robots increasing farming efficiency
Security	AI assisting in the fight against serious and harmful crimes; increasing cyber security capabilities

Against these opportunities, the White Paper identifies several categories of risk. These include risks to human rights (e.g., algorithmic discrimination), risks to safety (e.g., autonomous systems making harmful decisions), risks to fairness (e.g., biased hiring algorithms), risks to privacy and agency (e.g., surveillance and data harvesting), risks to societal wellbeing (e.g., social media algorithms that exploit addictive behaviours), and risks to security (e.g., AI-powered cyber attacks). Understanding this landscape of opportunities and risks is essential for any AI practitioner: the goal is to maximise the benefits of AI while proactively mitigating its potential harms.

2.3 Responsible AI: Five Guiding Principles

To guide the responsible development and deployment of AI, the UK Government's White Paper proposes five principles. These principles are intended to be implemented in a context-specific manner by existing domain-expert regulators, rather than through a single centralised AI regulator.

Principle	Description
Safety, Security and Robustness	AI systems should function reliably, be resilient to attacks, and not pose undue risks to users or the public.
Appropriate Transparency and Explainability	Users and affected parties should be able to understand how AI systems make decisions, to an extent proportionate to the context and risk.
Fairness	AI systems should not produce discriminatory outcomes; they must be designed and tested to avoid unjust bias.
Accountability and Governance	Clear lines of responsibility must exist for AI system outcomes; organisations deploying AI must have appropriate governance structures.
Contestability and Redress	Individuals affected by AI decisions should have mechanisms to challenge those decisions and seek remedies.

These principles reflect a growing international consensus that AI governance must be proactive rather than reactive. For students of AI, they serve as a practical checklist: when designing or evaluating an AI system, one should consider whether it is safe, transparent, fair, accountable, and contestable. Neglecting any of these dimensions can lead to systems that, however technically impressive, cause real harm to individuals and communities.

Key Concept — Responsible AI Principles

The five UK principles for responsible AI are: (1) Safety, security and robustness; (2) Appropriate transparency and explainability; (3) Fairness; (4) Accountability and governance; (5) Contestability and redress. These apply context-specifically through existing domain regulators.

Session 3: Machine Learning Definitions and Types

3.1 Subdomains of AI

Artificial intelligence encompasses a hierarchy of increasingly specialised subdomains. At the broadest level, AI refers to machines that act rationally. Within AI, Machine Learning (ML) — a term popularised in the 1950s — refers to AI systems that learn and adapt from data without being explicitly programmed for every possible scenario. Within ML, Deep Learning (DL), which emerged prominently in the 2010s, uses multi-layer neural networks to learn hierarchical representations of data, making previously intractable problems (such as image recognition and speech processing) computationally feasible. Most recently, Large Language Models (LLMs) represent a subset of deep learning: large transformer-based models pre-trained on vast datasets, often fine-tuned using reinforcement learning from human feedback.

Subdomain	Era	Core Idea
Machine Learning (ML)	1950s onwards	Systems that learn from data without explicit programming
Deep Learning (DL)	2010s onwards	Multi-layer neural networks learning hierarchical features
Large Language Models (LLMs)	2020s onwards	Large transformer models pre-trained on vast corpora, often using reinforcement learning

Understanding this hierarchy is important because it clarifies the relationship between the techniques covered in this module. Supervised and unsupervised learning are classical ML topics; neural networks and deep learning extend these foundations; and LLMs represent the current frontier. Each layer builds upon the previous one, and the principles learned at one level (e.g., avoiding overfitting) remain relevant at every subsequent level.

3.2 AI Landmark Moments

The history of AI is punctuated by landmark achievements that illustrate the progression from narrow, task-specific intelligence to increasingly general capabilities. These milestones provide context for understanding where the field stands today and what challenges remain.

Year	Milestone	Significance
1950s	Arthur Samuel's Checkers Player (IBM)	First program that learned from past games; Samuel coined the term 'machine learning' as 'the field of study that gives computers the ability to learn without being explicitly

		programmed'.
1997	IBM Deep Blue defeats Garry Kasparov	Evaluated 200 million positions per second using brute-force search; demonstrated that specialised AI could outperform the world's best human in a constrained domain.
2011	IBM Watson wins Jeopardy!	Used hundreds of algorithms to vote on answer confidence; demonstrated natural language understanding and knowledge retrieval.
2016	DeepMind's AlphaGo defeats Lee Sedol	Evaluated only ~10,000 positions per second but used deep neural networks and reinforcement learning; Go has ~10^{170} possible positions, making brute-force search impossible.
2022	ChatGPT 3.5 released by OpenAI	Democratised AI access, triggering an explosion of AI applications; demonstrated the power of large language models for general-purpose text generation and reasoning.
2025	Human coder defeats OpenAI LLM at AtCoder	Expert programmer 'Psyho' defeated an LLM in a 10-hour heuristic coding competition, illustrating that LLMs have not yet surpassed expert humans in all cognitive domains.

A clear trajectory emerges from these milestones: early AI relied on hand-crafted rules and brute-force computation; modern AI increasingly relies on learning from data, with deep learning and reinforcement learning enabling systems to discover strategies that no human designer could have programmed explicitly. Yet the 2025 AtCoder result reminds us that human expertise — particularly in creative problem-solving under novel constraints — remains formidable.

3.3 Types of Machine Learning

Machine learning algorithms are conventionally classified into three main types, distinguished by the nature of the training signal available to the learning system.

3.3.1 Supervised Learning

In supervised learning, the model is trained on a labelled dataset — that is, a dataset in which each input example is paired with a known correct output (the label or target). The model learns a mapping from inputs to outputs, which it can then apply to new, unseen inputs. Supervised learning encompasses two main task types.

- Regression — predicting a real or continuous value. For example, predicting a student's final module mark on a scale of 0 to 100.
- Classification — predicting a categorical output. For example, predicting whether a student's final mark falls into the category of 1st, 2:1, 2:2, 3rd, or fail.

The key requirement for supervised learning is the availability of high-quality labelled data. The quality, quantity, and representativeness of the labels directly determine the model's ability to generalise to new data.

3.3.2 Unsupervised Learning

In unsupervised learning, the model learns patterns from an unlabelled dataset — there are no target labels to guide the learning process. Instead, the algorithm discovers structure in the data on its own. The principal unsupervised learning tasks are:

- Clustering — detecting groups of similar observations based on feature similarity (e.g., customer segmentation in marketing).
- Density estimation — modelling the probabilistic distribution underlying the data.
- Dimensionality reduction — reducing the number of features while maximising information value, which aids visualisation and downstream modelling (e.g., Principal Component Analysis).

3.3.3 Reinforcement Learning

Reinforcement learning (RL) takes a fundamentally different approach: the model 'learns by doing'. An agent interacts with an environment, takes actions, and receives feedback in the form of rewards or penalties. Over many episodes, the agent learns a policy — a mapping from states to actions — that maximises its cumulative reward. The canonical example is DeepMind's AlphaGo, which learned to play Go at a superhuman level through self-play reinforcement learning.

RL is particularly suited to sequential decision-making problems where the consequences of actions unfold over time, and where the optimal strategy cannot be derived analytically. It connects directly to the rational agent framework discussed in Session 1: the RL agent is a rational agent that learns its policy from experience rather than having it pre-programmed.

Key Concept — Three Types of Machine Learning

Supervised learning trains on labelled data (regression for continuous targets, classification for categorical targets). Unsupervised learning discovers patterns in unlabelled data (clustering, density estimation, dimensionality reduction). Reinforcement learning learns by interacting with an environment and receiving reward/penalty feedback.

Session 4: The Data Science — ML Pipeline

4.1 Pipeline Overview

The machine learning pipeline is the structured, iterative process through which a data science project progresses from problem formulation to deployed model. Understanding this pipeline is essential because machine learning is far more than simply choosing an algorithm and pressing 'run': the quality of every upstream decision — from how the problem is framed to how the data is cleaned — directly determines the quality of the final model. The pipeline is iterative: results from later stages feed back into earlier ones, and multiple cycles are typically required to achieve an acceptable outcome.

The pipeline consists of the following stages:

- Problem Statement — defining the aim, context, and resources.
- Analysis Design — pre-specifying objectives, metrics, and methods.
- Data Ingestion — searching, collecting, and managing data.
- Exploratory Data Analysis (EDA) and Feature Engineering — understanding and transforming the data.
- Class Balancing and Feature Selection — addressing imbalance and selecting informative features.
- Model Training and Tuning (Validation) — fitting models and optimising hyperparameters.
- Model Evaluation (Testing) — assessing generalisation on held-out data.
- Results Interpretation — explaining and contextualising model outputs.

A critical lesson emphasised throughout the lecture is that we iterate the entire pipeline to achieve the best result given the data and context, and that we must always interpret both models and results in the context of the original problem.

4.2 Problem Statement

Every ML project begins with a clear problem statement. The PEAS framework from Session 1 provides a useful structure: the aim defines the task to be performed (the Performance measure and goal), the context specifies the Environment in which the system will operate, the hardware and software define the Actuators available, and the data search, collection, and access systems define the Sensors. A well-defined problem statement should specify whether the task is a prediction problem (supervised learning), a pattern-discovery problem (unsupervised learning), or a sequential decision-making problem (reinforcement learning).

4.2.1 Overfitting, Underfitting, and Generalisation

Three concepts are central to understanding model quality. The lecture illustrates these using a classification example — classifying map locations as containing healthy or sick trees — and a regression example of fitting a polynomial curve to data points.

Concept	Definition	Consequence

Overfitting	The model matches too closely the training set; it is 'too complex' relative to the amount of training data.	The model fails to generalise to new, unseen data. It captures noise and idiosyncrasies rather than genuine patterns.
Underfitting	The model is 'too simple' relative to the underlying patterns in the data.	The model makes poor predictions even on the training data itself. It fails to capture the true decision boundary.
Generalisation	The model makes good predictions on new data it has not seen during training. The model is appropriately complex.	This is the goal of machine learning: a model that balances complexity to capture real patterns without memorising noise.

The aim of all machine learning is to learn a decision boundary (in classification) or a function (in regression) that generalises well. This means the model should be complex enough to capture the underlying structure in the data, but not so complex that it memorises the training examples. The tension between overfitting and underfitting — often called the bias–variance trade-off — is one of the most fundamental challenges in machine learning and recurs throughout the module.

Key Concept — Overfitting vs Underfitting

Overfitting occurs when a model is too complex and memorises training noise; it performs well on training data but poorly on new data. Underfitting occurs when a model is too simple and fails to capture real patterns. The goal is generalisation: a model that makes accurate predictions on unseen data.

4.3 Analysis Design

Before touching the data, it is essential to pre-design as much of the ML pipeline as possible. This means specifying outcomes of interest, choosing evaluation metrics (linked to the PEAS performance measure), selecting data processing and class-balancing methods, defining training, validation/cross-validation, and testing approaches, and planning how results will be interpreted and how ethical and social issues will be addressed. Objectives should follow the SMART framework (Specific, Measurable, Achievable, Relevant, Time-bound) to avoid vagueness.

Why not run straight into data analysis? The lecture warns of several risks that arise from an undisciplined approach:

- Data and model biases, including spurious correlations — patterns in the data that appear meaningful but arise from coincidence or confounding variables.
- Overfitting and lack of generalisability, including sequential overfitting — where repeated model selection on the same test set inflates apparent performance.
- Lack of transparency — producing an unreliable 'black box' model whose decisions cannot be explained or audited.

4.3.1 Choosing Evaluation Metrics Carefully

A particularly important analysis-design decision is the choice of evaluation metrics. The lecture demonstrates a critical pitfall: accuracy alone can be highly misleading when classes are imbalanced. Consider a dataset where 98% of observations belong to class A (majority) and only 2% to class B (minority). A model that simply predicts 'class A' for every observation achieves 98% accuracy — yet it is completely useless for detecting class B, which may be the class of primary interest (e.g., a rare disease).

The solution is to use multiple, class-specific metrics. Precision (for each class) measures the proportion of positive predictions that are correct; recall measures the proportion of actual positives that are detected. Subclass metrics — examining performance for each class separately — reveal disparities that overall accuracy conceals. This is especially critical in domains such as healthcare, criminal justice, and finance, where failing to detect the minority class has severe consequences.

4.4 Data Ingestion

Data ingestion is often the most time-consuming stage of the pipeline. It encompasses several sub-tasks, each with its own challenges.

- Data search — locating relevant datasets, whether through manual browsing, automated scripts, or semi-automatic methods (e.g., using LLMs to identify sources).
- Data access prerequisites — securing ethics governance (especially for individual-level data), data agreements and contracts, and access credentials (e.g., MFA, API keys).
- Data collection — determining whether the data already exists (observational study) or must be collected (survey), with attention to survey design, sampling, and achieving adequate sample size.
- Data support — managing data ingestion, maintenance, and removal, including backups and compliance with data storage policies.

Whether the data resides on owned hardware, an external cloud service, or an access-only server, the practitioner must consider data governance, linkage possibilities, and legal constraints. These practical considerations are often underestimated by beginners but are essential for any real-world ML project.

4.5 Data Bias and Spurious Correlations

One of the most insidious problems in machine learning is data bias — systematic errors in the training data that cause the model to learn incorrect or misleading patterns. The lecture illustrates this with the famous 'tank detection' anecdote: a model trained to distinguish tanks from non-tanks in photographs appeared to perform well, but in reality it had learned a spurious correlation — for example, that tanks tended to appear in images with a particular background or lighting condition. When presented with new images where these incidental features were absent, the model failed catastrophically.

The solution to data bias is expert-guided training data augmentation: deliberately diversifying the training data so that the model is forced to learn the true discriminative features rather than superficial correlations. This is particularly important when there is 'not enough data' — augmentation techniques (rotation, cropping, colour adjustment for images; synonym substitution for text) can expand the effective training set. However, augmentation must be guided by domain expertise to avoid introducing new biases.

More broadly, the lecture cautions against spurious correlations — statistical associations that appear meaningful but have no causal basis. Pre-specifying aims and methods (as discussed in Section 4.3) is the primary defence: by committing to hypotheses before examining the data, researchers reduce the risk of 'finding' patterns that are merely artefacts of the data.

4.6 Training, Validation, and Testing

A fundamental principle of machine learning is that a model must be trained and evaluated on different sets of data. If the same data is used for both training and evaluation, the reported performance will be optimistically biased because the model has 'seen' the evaluation examples. The standard approach is to split the available data into separate subsets.

4.6.1 Data Leakage

Data leakage occurs when information from the test set inadvertently 'leaks' into the training process, leading to artificially inflated performance metrics. This can happen explicitly (e.g., using test data during training) or implicitly (e.g., performing feature selection or data cleaning on the full dataset before splitting). Data leakage is sometimes described as 'cheating evaluation' because it gives the model an unfair advantage that will not be available in production.

Key Concept — Data Leakage

Data leakage occurs when test-set information is used during training, inflating performance metrics. To avoid it, always split data before any preprocessing, feature selection, or model training. Perform EDA and data cleaning only on the training set.

4.6.2 Sequential Overfitting and the Validation Set

A subtler form of leakage is sequential overfitting. This occurs when the test set is used during model selection — for example, when a practitioner repeatedly trains different models, evaluates them on the test set, and selects the one with the best test performance. Over many iterations, the test set effectively becomes an implicit part of the training process, and the reported test performance no longer reflects true generalisation.

The solution is to introduce a separate validation set. The data is split into three parts: a training set (for fitting models), a validation set (for hyperparameter tuning and model selection), and a test set (used only once, for final evaluation). A typical split might be 70% training, 15% validation, and 15% test. The test set is kept strictly quarantined until the very end of the modelling process.

4.6.3 Cross-Validation

When data is limited, setting aside a fixed validation set may waste valuable training examples. Cross-validation addresses this by systematically rotating which portion of the training data serves as the validation fold. In k-fold cross-validation, the training set is divided into k equally sized subsets (folds). The model is trained k times, each time using k−1 folds for training and the remaining fold for validation. The reported performance is the average across all k folds, providing a more robust estimate of model quality.

For example, in 5-fold cross-validation, the training data is split into five parts. In each iteration, four parts are used for training and one for validation. After five iterations, every data point has been used for validation exactly once. Cross-validation is the de facto standard for model selection in practice and is supported directly by libraries such as scikit-learn.

4.7 Exploratory Data Analysis and Feature Engineering

Exploratory Data Analysis (EDA) is the process of systematically examining a dataset to understand its structure, identify patterns, detect anomalies, and inform modelling decisions. EDA typically involves summary statistics, visualisation (histograms, scatter plots, box plots, correlation matrices), and domain-driven investigation. The lecture emphasises a fundamental axiom: 'data is beautiful' — but all datasets and analysis methods have limitations. As the saying goes, 'garbage in, garbage out': if the input data is flawed, no amount of sophisticated modelling will produce reliable results.

Data cleaning is a critical component of EDA. It involves checking for and removing duplicates, detecting and handling missing values (through imputation or removal), identifying and addressing outliers (e.g., a person recorded as age 155), and correcting inconsistent data (e.g., a negative age). Feature engineering transforms existing features or derives new ones (through techniques such as binning, clustering-based features, and interaction terms) to improve model performance. Advanced deep neural networks can learn features automatically, but for classical ML methods, thoughtful feature engineering remains essential.

A crucial practical tip: EDA and data cleaning must be performed only on the training set. Applying these steps to the full dataset before splitting introduces data leakage, because statistics computed on the test data (e.g., column means used for imputation) would influence the training process.

4.8 Class Balancing and Feature Selection

4.8.1 Class Balancing

Class-imbalanced datasets — where one label is far more common than another — are pervasive in real-world applications. For example, in electronic health records, a rare disease might affect only 1% of patients. If the model simply predicts the majority class, it achieves high overall accuracy but completely fails on the minority class, which is typically the class of interest.

The lecture presents a concrete class-balancing strategy using a sunflower-and-roses dataset (200 sunflowers vs 2 roses). The approach involves two steps: first, down-sample the majority class (e.g., from 200 to 8, yielding an 8:2 ratio); second, up-weight the majority class when measuring errors (by a factor equal to the original ratio divided by the new ratio, e.g., 200/8 = 25). This ensures that the model is trained on a more balanced distribution while preserving the statistical importance of each class. The degree of down-sampling is itself a hyperparameter that can be tuned during model selection.

4.8.2 Feature Selection

Not all features in a dataset are equally informative. Feature selection is the process of identifying the most relevant features for the modelling task, discarding those that add noise or redundancy. Desirable feature characteristics include ease of extraction, high discriminative power, noise robustness, and invariance to irrelevant transformations (e.g., image features that are unaffected by rotation, translation, and scaling).

Feature selection is problem-dependent and should be informed by literature review, EDA, and domain expertise. Automatic algorithms — such as Feature Importance, minimum Redundancy Maximum Relevance (mRMR), LASSO regularisation, and backwards selection — can assist in identifying the optimal feature subset. scikit-learn provides comprehensive support for feature selection methods.

4.9 Model Training, Tuning, and Ensemble Models

Once the data is prepared, the model is trained on the training set and errors are evaluated on the validation set (or via cross-validation). Hyperparameter tuning involves selecting model parameters — such as learning rate, regularisation strength, or tree depth — based on validation performance. Model selection compares multiple optimised models on the test set, though the lecture cautions that statistical pairwise comparisons may fail (the Bonferroni correction problem) when many models are compared simultaneously.

Finding the 'best model' is inherently problem-dependent: there is no universally superior algorithm (a principle formalised as the No Free Lunch theorem). Practitioners should consider not only performance metrics but also training time, computational resources, and model interpretability.

Ensemble models combine multiple individual models to achieve better performance than any single model alone. They can combine models of the same type (e.g., Random Forest, Bagging, Boosting) or models of different types (e.g., stacking, where a meta-learner combines predictions from diverse base models). When using ensemble methods, a separate test set must be used to evaluate the composite model, to avoid the sequential overfitting trap described in Section 4.6.2.

Key Concept — Ensemble Models

Ensemble models combine multiple models to improve performance. Same-type ensembles include Random Forest, Bagging, and Boosting. Different-type ensembles include Stacking, where a meta-learner combines diverse base models. A separate test set must always be used for final evaluation of ensemble models.

4.10 Model Evaluation and Interpretation

The final stages of the pipeline concern evaluation and interpretation. Evaluation requires using multiple measures — never relying on a single metric — and performing error slicing: examining performance across subgroups (e.g., low-risk vs high-risk patients) to identify hidden biases. A model that performs well overall but poorly on a critical subgroup may be unacceptable in practice.

Interpretation is equally important, particularly in regulated domains where decisions must be explainable. Only some models are directly interpretable (e.g., decision trees, linear regression). For 'black box' models such as deep neural networks, interpretable ML tools provide post-hoc explanations. Key tools include SHAP (Shapley Additive Explanations), which assigns each feature a contribution to each prediction; LIME (Local Interpretable Model-agnostic Explanations), which approximates the model locally with an interpretable surrogate; and Partial Dependency Plots, which show the marginal effect of a feature on the predicted outcome.

Finally, the practitioner must close the data science loop by comparing results against existing literature and domain experts' views, and assessing how well the model answers the original aims and objectives defined in the problem statement. This reflective step ensures that the technical work remains grounded in the real-world problem it set out to solve.

Key Concept — Closing the Data Science Loop

Model evaluation requires multiple metrics, error slicing across subgroups, and interpretability tools (SHAP, LIME, Partial Dependency Plots) for black-box models. Results must be compared against existing knowledge and assessed against the original problem statement to ensure the project has addressed its stated aims.

References and Further Reading

- Russell, S. and Norvig, P. (2022) Artificial Intelligence: A Modern Approach. 4th edn. Available online at UoR Library via ProQuest: https://ebookcentral.proquest.com/lib/reading/detail.action?docID=6563568
- UK Government (2023) 'A pro-innovation approach to AI regulation' (White Paper). Available at: https://www.gov.uk/government/publications/ai-regulation-a-pro-innovation-approach/white-paper
- DeepLearning.AI, AI for Everyone (AI basic overview). Available at: https://learn.deeplearning.ai/courses/ai-for-everyone
- ISO, Machine Learning overview. Available at: https://www.iso.org/artificial-intelligence/machine-learning
- Lones, M.A. (2024) 'Avoiding common machine learning pitfalls', Patterns. https://doi.org/10.1016/j.patter.2024.101046
- Google Machine Learning Crash Course (intermediate; e.g., overfitting, class imbalance). Available at: https://developers.google.com/machine-learning/crash-course
- Molnar, C. Interpretable Machine Learning: A Guide for Making Black Box Models Explainable. Available at: https://christophm.github.io/interpretable-ml-book/
- scikit-learn User Guide. Available at: https://scikit-learn.org/stable/user_guide.html