Exercise 1:

1) **AI vs ML vs DL**

   - *Main idea:* Artificial Intelligence is the broad field of building systems that perform tasks associated with intelligence; Machine Learning is a subfield that learns from data to improve task performance; Deep Learning is a subfield of ML based on multi-layer neural networks.
   - ML and statistics overlap strongly in methods and goals; differences are often historical and terminological.
   - *Lecturer comment:* Keep the taxonomy clear to avoid using AI as a catch-all. Position ML as data-driven prediction and DL as a flexible function class inside ML. Translate terms across ML and statistics to reduce confusion.

2) **Learning paradigms: supervised, unsupervised, reinforcement learning**

   - *Main idea:* Supervised learning uses labeled pairs (x, y) to learn $x \to y$. Unsupervised learning uses only x to discover structure (clustering, dimensionality reduction, anomaly detection). Reinforcement learning learns via interaction and rewards, optimizing long-run return.
   - *Lecturer comment:* Specify underlying supervised task (regression, binary or multiclass classification) before picking a learner. Note that unsupervised learning is not a single objective (can be clustering, dimensionality reduction, anomaly detection optimize different criteria, ...); RL rewards can be sparse, delayed, and noisy.

3) **Data in supervised learning: features vs target; labeled vs unlabeled; Iris example**

   - *Main idea:* The target y is the prediction goal; features x describe objects. We assume a predictive relation between x and y.
   - Distinguish labeled from unlabeled data. The Iris dataset (150 samples, 4 measurements) is a clean toy example for classification.
   - *Lecturer comment:* Iris is pedagogically useful but unusually clean; do not generalize its simplicity to real data. When first seeing any dataset, label the data type: supervised with (x, y) or unsupervised with only x.

4) **Data types and encodings: numeric vs categorical; one-hot vs dummy; ordinal encodings**

   - *Main idea:* Many learners expect numeric inputs, so categorical variables need encoding.
   - One-hot uses k indicator columns; dummy uses $k - 1$ to avoid a redundant column. Ordinal encodings are appropriate only when a natural order exists.
   - *Lecturer comment:* Decision trees can handle categoricals more directly; linear models typically require encodings. Use dummy coding to avoid singular design matrices; avoid imposing false order with naive integer encoding (where categories are assigned numbers).

5) **Data-generating process and the i.i.d. assumption**

   - *Main idea:* We assume data arise from an unknown distribution $P_{X,Y}$. Standard analyses treat samples as independent and identically distributed.
   - *Lecturer comment:* I.i.d. is a modeling convenience. Time series, networks, and nonstationary streams often violate it. Keep the assumption now, but flag where it can fail.

6) **Tasks: regression vs classification**

   - *Main idea:* If the target is numeric, the task is regression; if the target is categorical with g classes, the task is classification.

- *Lecturer comment:* Tie the task directly to target type to avoid evaluation mismatches later. The same domain can be framed for prediction or explanation; be explicit which you pursue.

7) **Predict vs explain**

- *Main idea:* Predict focuses on out-of-sample accuracy; explain focuses on understanding patterns and relations. Both require adequate fit.
- *Lecturer comment:* Explanation here is descriptive, not necessarily causal. Stakeholders typically want both a strong predictor and an understandable story; decide which objective dominates before modeling.

8) **Models and hypothesis spaces**

- *Main idea:* A model f maps features to outputs; in classification f often produces scores or probabilities. We restrict to a hypothesis space H to encode structure and make learning feasible.
- *Lecturer comment:* Demystify models as functions $f : \mathcal{X} \to \mathcal{Y}$. The choice of H (linear functions, trees, neural networks, etc.) constrains what can be learned.

9) **Parameters, identifiability, and illustrative examples**

- *Main idea:* Models in H share a parametric form with parameter vector θ; choosing θ fixes f. Some classes are non-identifiable (distinct θ yield the same f).
- Examples: univariate linear $f(x) = \theta_0 + \theta_1 x$, bivariate quadratics, and RBF networks with centers, widths, and weights. Hyperparameters such as number of centers k or bandwidth define families prior to training.
- *Lecturer comment:* Parameterization operationalizes H. Non-identifiability matters for optimization and interpretation. Hyperparameters set the scope of the family before fitting.

10) **Learners (inducers)**

- *Main idea:* A learner receives training data D and controls Λ, and selects $f \in H$ (or θ) that minimizes empirical risk on D.
- Formal mapping: $\mathcal{I} : \mathbb{D} \times \Lambda \to H$.
- *Lecturer comment:* Separate the algorithm from the model. Any training code instantiates the mapping from data and controls to a fitted function f.

11) **Loss functions**

- *Main idea:* The loss $L(y, f(x))$ measures pointwise error. For regression, $L1$ and $L2$ are common; classification uses appropriate losses or surrogates. Aggregating losses gives empirical risk.
- *Lecturer comment:* $L2$ magnifies large residuals; $L1$ is more robust. Keep a tiny numeric example handy to make robustness concrete.

12) **Risk minimization: theoretical vs empirical risk**

- *Main idea:* Theoretical risk $R(f) = E[L(Y, f(X))]$ is defined under the unknown data distribution. We approximate it by empirical risk $R_{\text{emp}}(f) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, f(x_i))$ on i.i.d. samples.
- *Lecturer comment:* Alternatives exist (estimate $P_{X,Y}$, or assume a parametric form), but ERM is the default in this chapter because $P_{X,Y}$ is unknown.

13) **Empirical Risk Minimization (ERM)**

- *Main idea:* ERM chooses $f = \arg\min_{f \in H} R_{\text{emp}}(f)$, equivalently $\theta = \arg\min_{\theta} R_{\text{emp}}(\theta)$.
- For finite H one could tabulate risks; for infinite H we optimize a surface over parameters.
- *Lecturer comment:* Reducing learning to numerical optimization is powerful but incomplete; generalization and model selection enter next.

14) **Optimization for learning**

- *Main idea:* Many learners solve $\min_{\theta} R_{\text{emp}}(\theta)$ via numerical optimization. Gradient descent updates parameters along negative gradients. The learning rate controls step size.
- Local minima vs global minima: in practice, good local optima often suffice. Some models admit analytic solutions (for example, OLS in linear regression under standard conditions).

- *Lecturer comment:* Trade off speed and stability with the step size. Name families briefly: first order (GD, SGD), second order (Newton type), and practical variants (momentum, Adam).

15) **Components of supervised learning**

- *Main idea:* Many supervised learners decompose into hypothesis space H plus loss/risk plus optimization. Regularization can be folded into the risk.
- Examples of choices: H (linear, trees, nets), losses (MSE, NLL, misclassification surrogates), optimizers (analytical, gradient based, combinatorial).
- *Lecturer comment:* Keep returning to the triad H + risk + optimization as an organizing lens to compare algorithms. Regularization integrates naturally into the objective.