AI TERMINOLOGY / LEVEL 1 / GROUP 1

AI Model

From Basics to Advanced Research

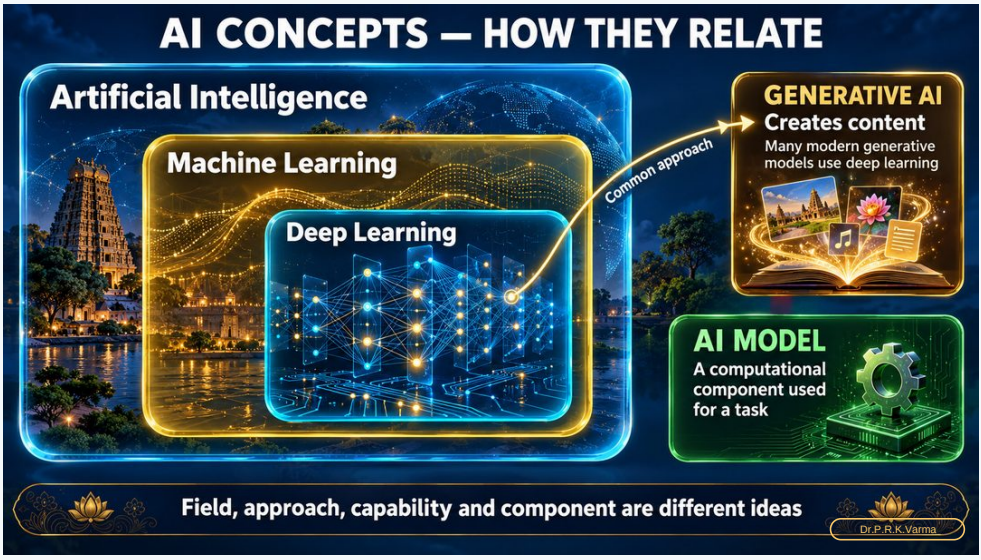


Presenter guide with slide visuals and notes. The slide numbers match the PowerPoint.

BASICS 02–26

ADVANCED 27–36

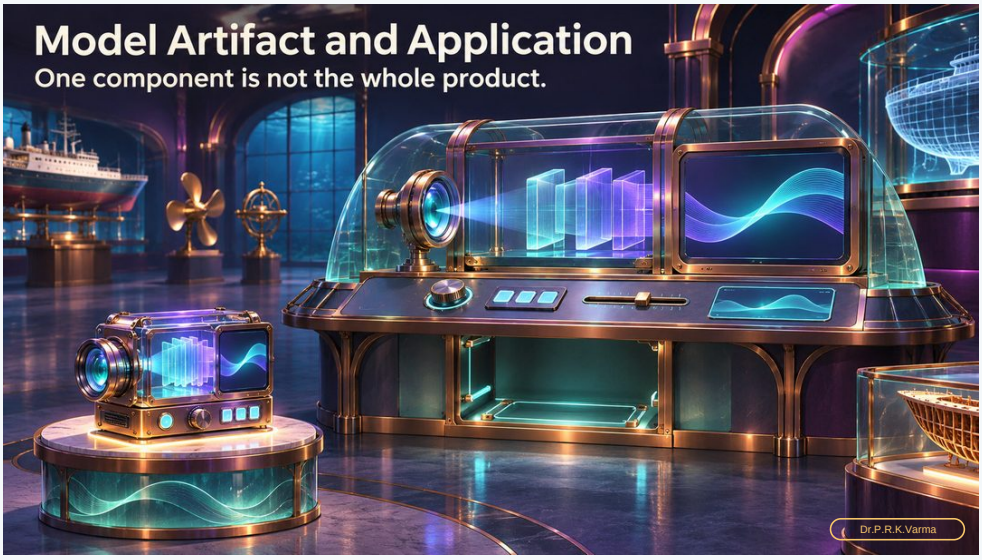
RESEARCH 37–46



AI Concepts: How They Relate

PRESENTER NOTES

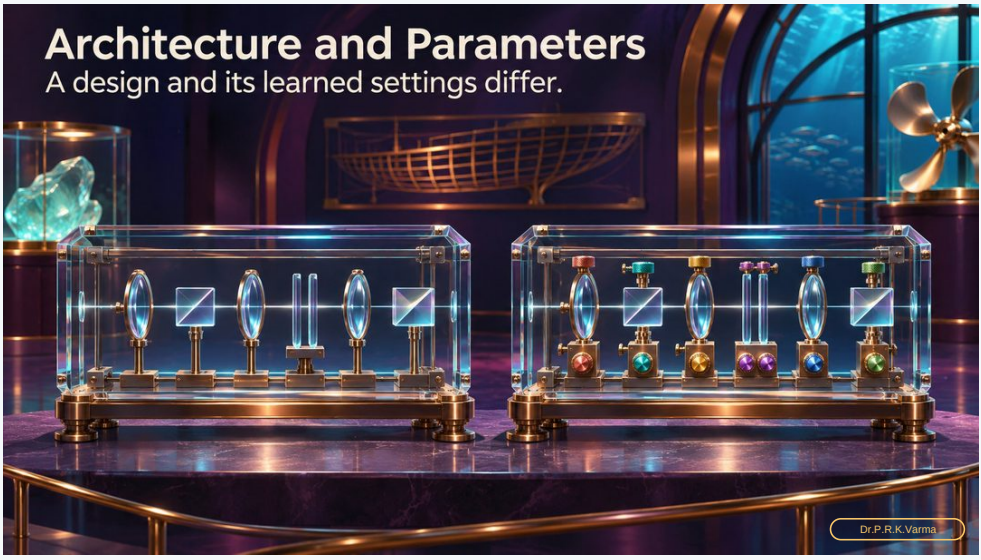
Machine learning is part of AI, and deep learning is part of machine learning. Generative AI describes content creation, while an AI model is a working computational component, not another nested field.



Model Artifact and Application

PRESENTER NOTES

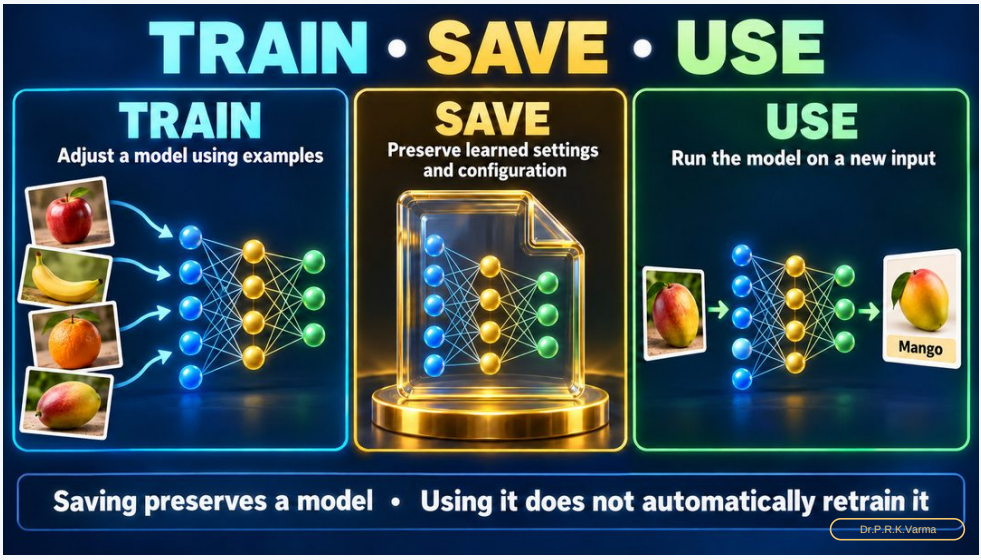
A model artifact is a reusable asset that performs a computation. An application also includes interfaces, input handling, decision rules, logging, and other software that surrounds the model.



Architecture and Parameters

PRESENTER NOTES

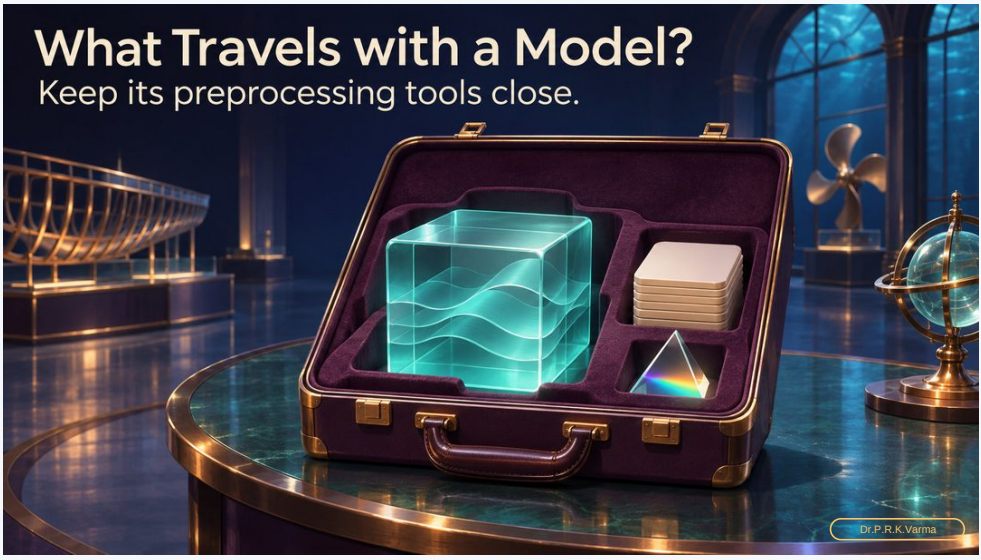
Architecture defines the structure and operations of a model. Parameters are values, such as weights, that training fits within that structure. Two models can share an architecture but have different learned parameters.



Train, Save, Use

PRESENTER NOTES

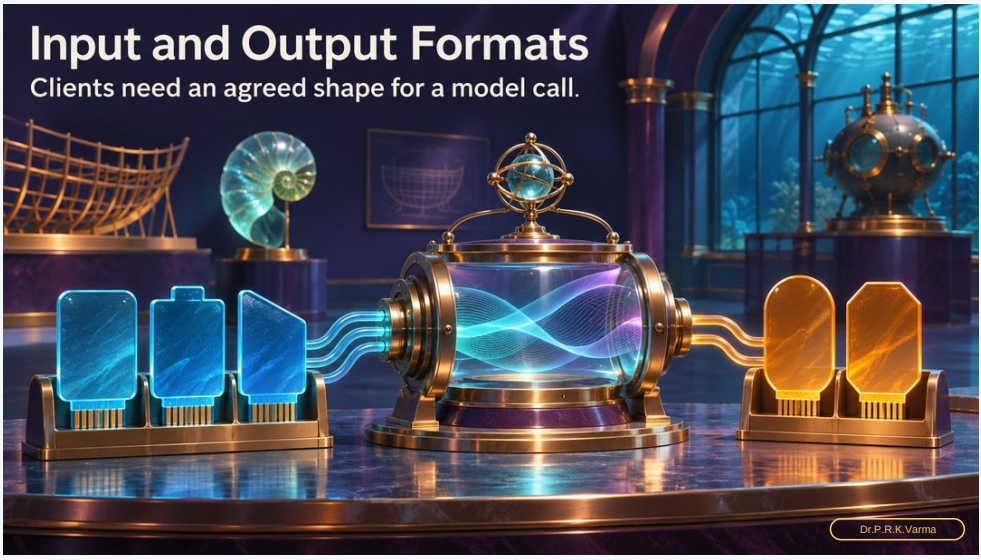
Training adjusts a model using examples. Saving preserves learned settings, and use applies the model to a new input. Saving does not turn an output into a model, and ordinary use does not retrain it.



What Travels with a Model?

PRESENTER NOTES

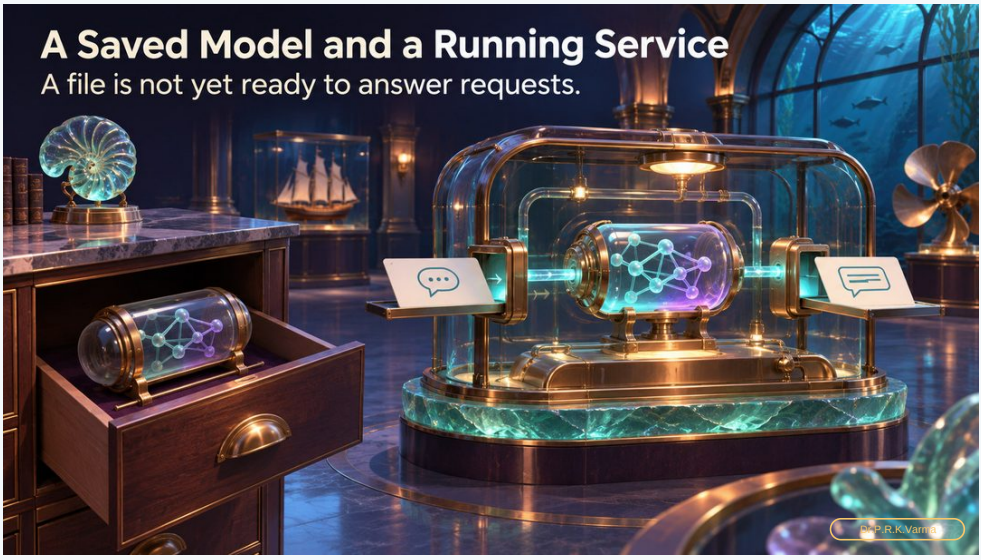
A model may depend on a tokenizer or preprocessing pipeline. Use the matching version and settings, such as normalization and image size, so inputs have the meaning expected by the model.



Input and Output Formats

PRESENTER NOTES

Check which inputs the model accepts and what format it returns: for example a fixed-size image array in and a list of class scores out. The surrounding application should detect malformed inputs or outputs.



A Saved Model and a Running Service

PRESENTER NOTES

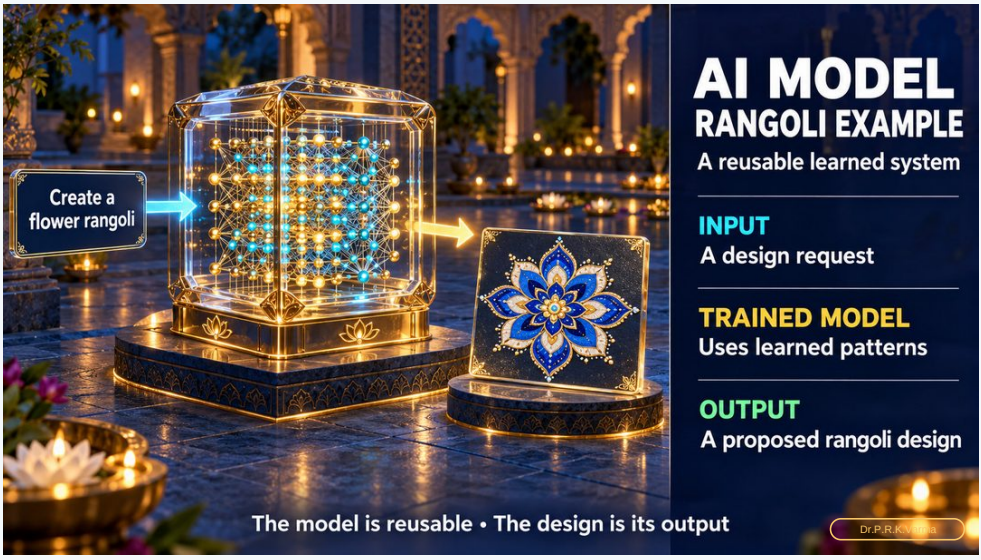
A saved file stores model contents. A running service loads those contents into a compatible runtime and handles requests. The same artifact can be served through different applications and systems.



A Reusable Model Producing a Recipe

PRESENTER NOTES

A trained computational model takes ingredients and a request as input, then proposes a recipe. Here the recipe is the output, not the model; the reusable learned system can respond to later requests.



A Reusable Rangoli Model

PRESENTER NOTES

A saved model accepts a design request and proposes a rangoli pattern. Reusing the trained component is different from reusing one fixed picture, and ordinary use does not automatically retrain the model.



A Reusable Fabric-Design Model

PRESENTER NOTES

The trained component can produce proposed fabric patterns for new requests. The model is not the finished cloth or the generated design; those are downstream outputs requiring review.



A Reusable Music Model

PRESENTER NOTES

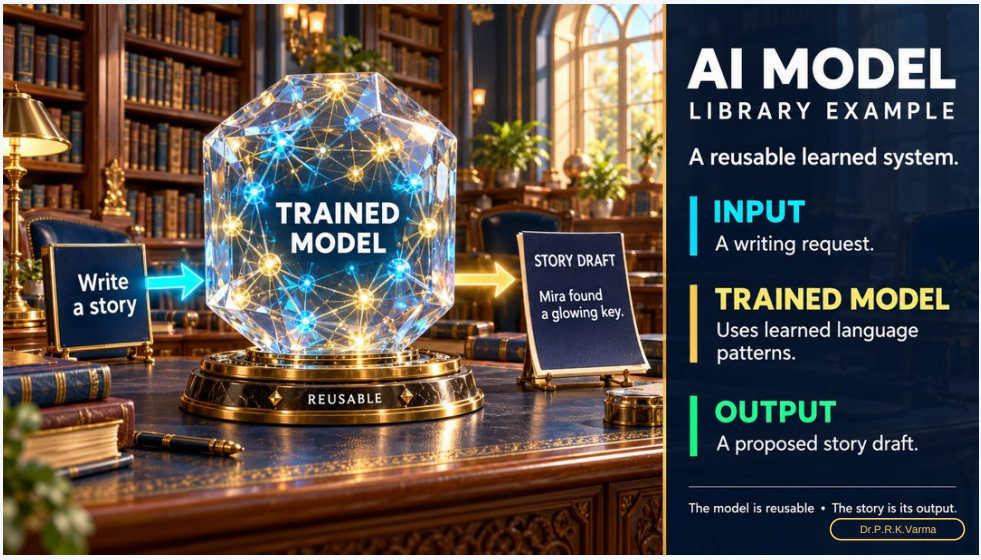
A model accepts a musical request and produces a melody proposal. The trained system and the particular melody are different things. This slide is a static illustration rather than an audio demonstration.



A Reusable Garden-Design Model

PRESENTER NOTES

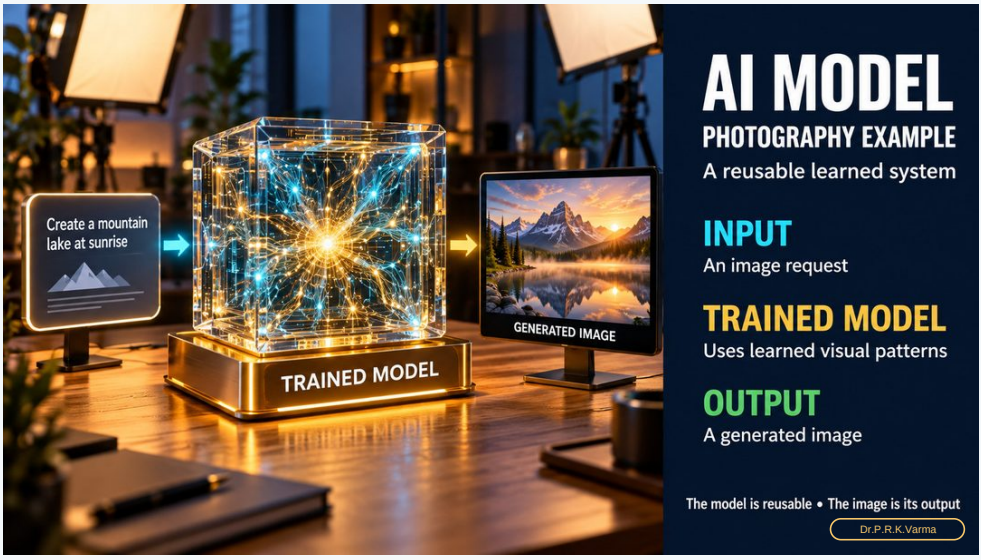
New garden requests can be processed by the same trained model. Each proposed layout is an output, and a human checks whether it is suitable for the real site.



A Reusable Story Model

PRESENTER NOTES

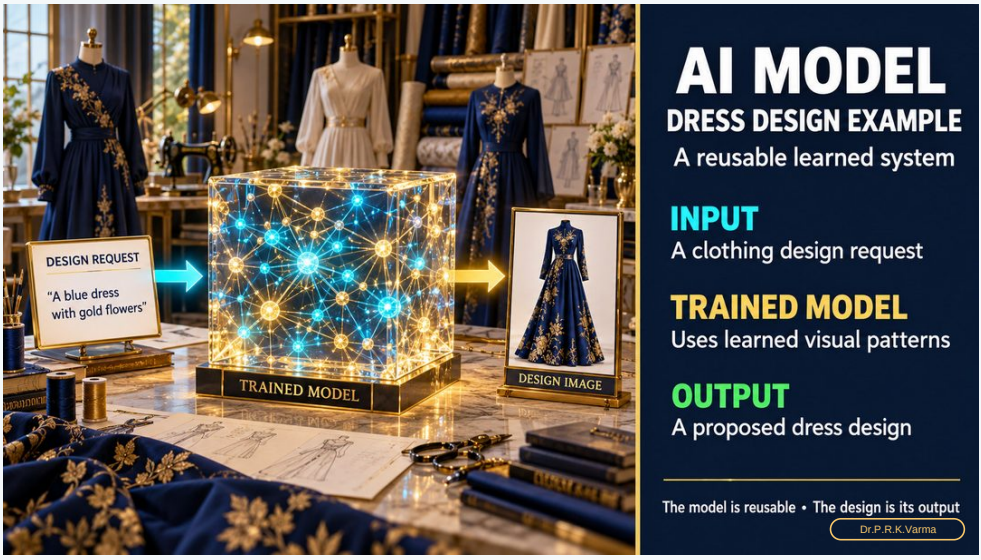
A writing request enters a trained language model and a draft emerges. The model is the reusable computational component; the resulting story is an output and still needs editorial review.



A Reusable Image-Generation Model

PRESENTER NOTES

A saved image model can respond to new visual requests. The generated mountain scene is one output, not the model itself or evidence of a photographed real event.



A Reusable Clothing-Design Model

PRESENTER NOTES

The trained model can process later clothing requests and propose designs. The model, the generated design and a manufactured garment are three distinct things.



Reusing a Learned Cloth-Inspection Model

PRESENTER NOTES

A cloth-inspection model applies learned patterns to a new image and flags a potential fault. This recognition task differs from generating fabric designs, and saving a model does not copy a weaver’s mind or expertise.



The Cake Recipe Analogy

PRESENTER NOTES

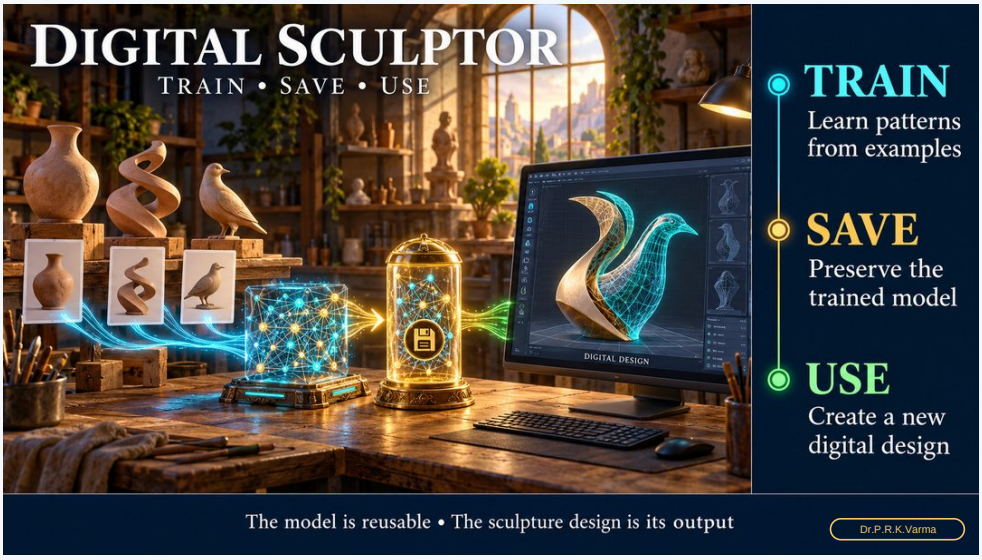
Improving a recipe through practice, preserving it and using it again illustrates train, save and use. In this analogy the recipe represents the model and cakes represent outputs. Keep that mapping separate from a recipe-generating model, where the recipe is itself an output.



Coding the Canvas: The Model and Its Artwork

PRESENTER NOTES

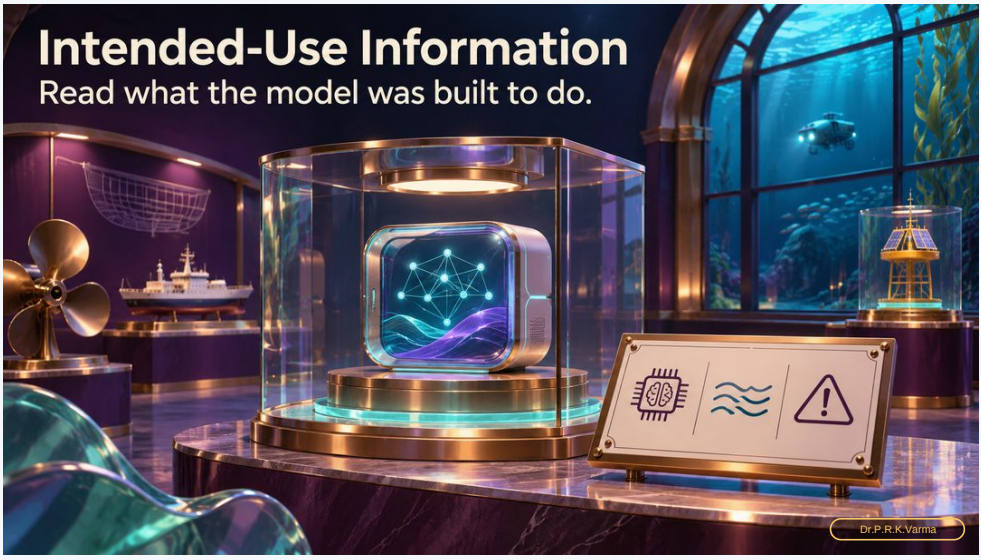
Examples support learning; saving preserves the trained model; a prompt produces a new image. The model is the reusable tool, while the artwork is its output, not a replacement for the model.



Digital Sculptor: Learn, Preserve and Generate

PRESENTER NOTES

Examples help adjust a model; saving preserves the trained system; a new request produces a digital sculpture design. The model is reusable, while the design is its output. It does not physically sculpt an object or guarantee perfect skill.



Intended-Use Information

PRESENTER NOTES

Documentation can describe intended use, evaluated conditions, known limits, and access or licensing terms. Check the model actually applies to your setting; a model card is evidence to review, not a guarantee.



Repeatable or Variable Outputs

PRESENTER NOTES

Some model calls are designed to return the same result for the same input. Sampling can produce different valid outputs. Hardware, model version, and serving settings can also affect exact reproducibility.



Confidence and Limits

PRESENTER NOTES

A confidence score indicates what the system reports about its prediction, not proof that it is correct. Check how reliable the scores are on examples similar to those the model will see, and how it behaves on unfamiliar inputs.



Choosing a Task-Specific Model

PRESENTER NOTES

Start with the type of prediction or output you need. Compare plausible models on representative examples and constraints such as accuracy, response time, memory, and cost. The largest model is not necessarily best.



When Not to Reuse a Checkpoint

PRESENTER NOTES

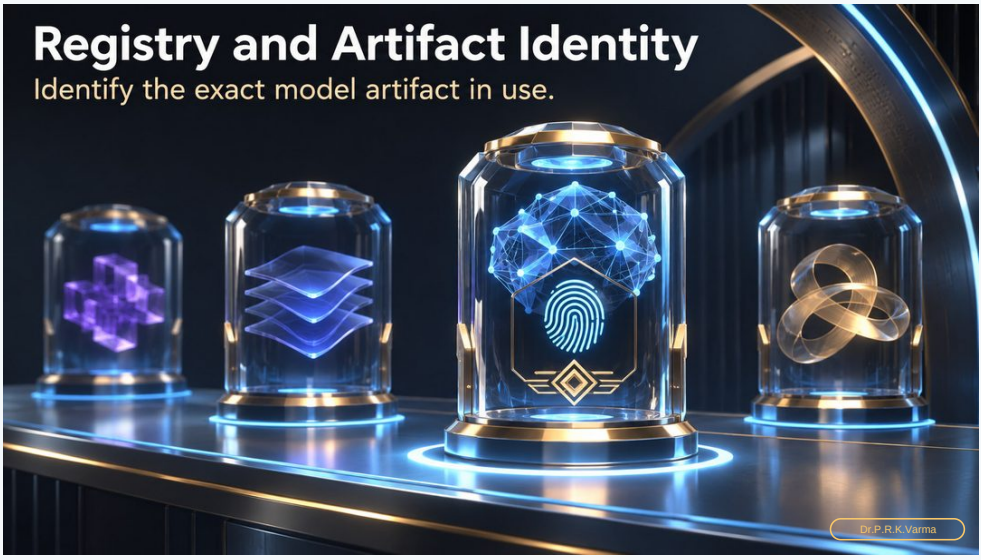
Do not reuse a checkpoint just because it is available. Incompatible inputs, unusable documentation, unsupported populations or environments, unexpected risks, or incompatible permissions can justify choosing another model or starting over.



Five Ms: A Memory Aid

PRESENTER NOTES

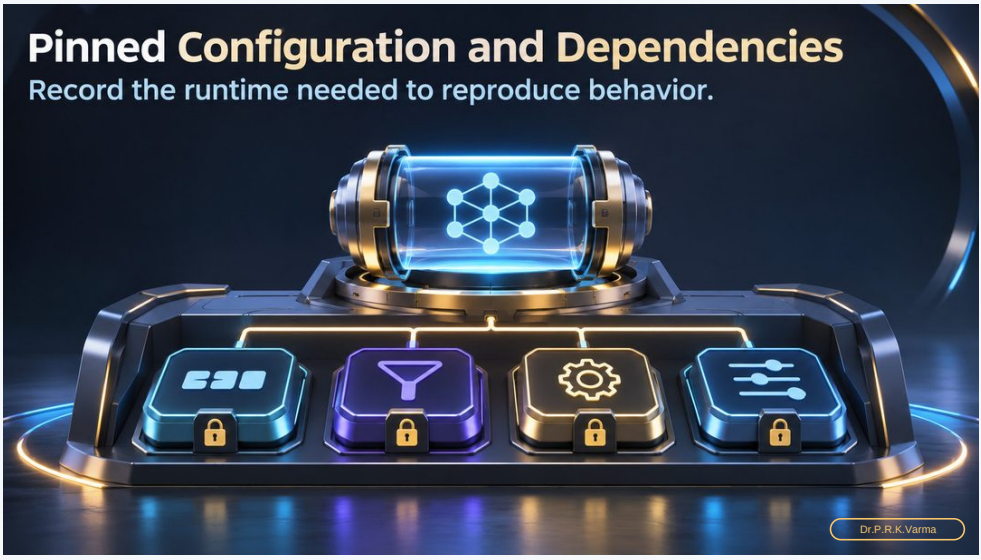
Machines, Many Examples, Many Layers, Making Content and Model for Reuse recall five related concepts. They are memory hooks, not five training stages or a chain of nested categories.



Registry and Artifact Identity

PRESENTER NOTES

A registry records the model artifact, version, content digest and metadata. Track immutable artifacts rather than assuming a mutable alias points to the same weights across deployments.



Pinned Configuration and Dependencies

PRESENTER NOTES

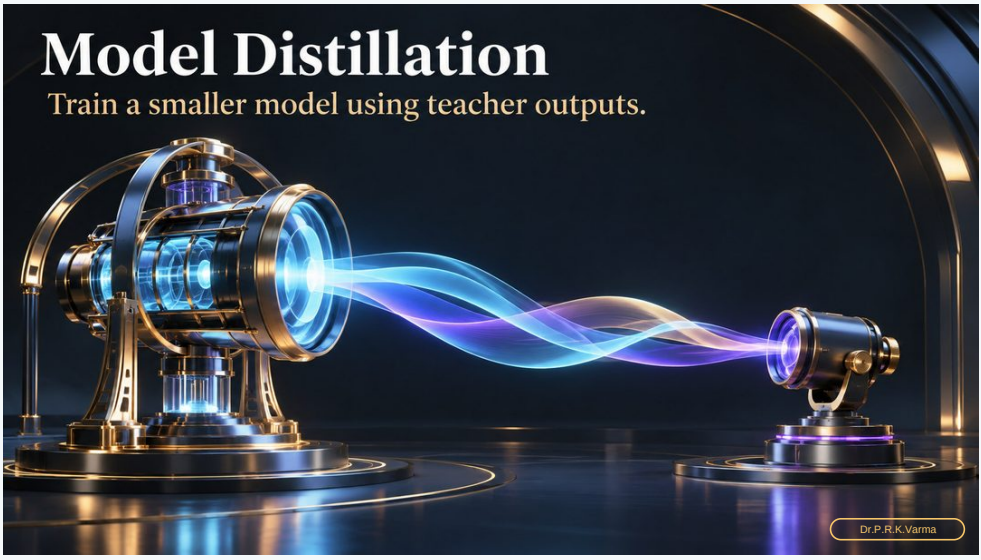
Pin the model configuration, tokenizer, preprocessing, dependencies and supported accelerator/runtime where applicable. Recreate them in a clean environment and verify behavior on reference inputs.



Compatibility and Interface Contracts

PRESENTER NOTES

Specify expected fields, shapes, ranges, output semantics and errors. Test the candidate against the interface contract and consumers, including edge cases and whether old clients can still interpret outputs.



Model Distillation

PRESENTER NOTES

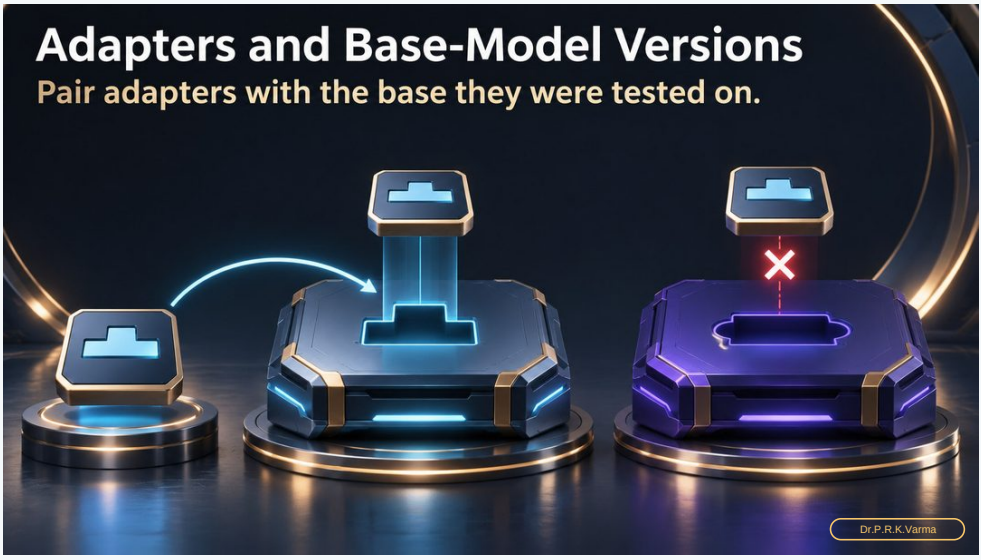
Train a student using teacher outputs and optional ground-truth labels. Compare the student against the teacher and a same-size baseline for quality, latency and failure cases.



Quantization Compatibility Checks

PRESENTER NOTES

Quantization changes numeric precision, but supported formats and operations vary by runtime. Test representative inputs, output changes, memory and latency on the intended hardware rather than assuming a benefit.



Adapters and Base-Model Versions

PRESENTER NOTES

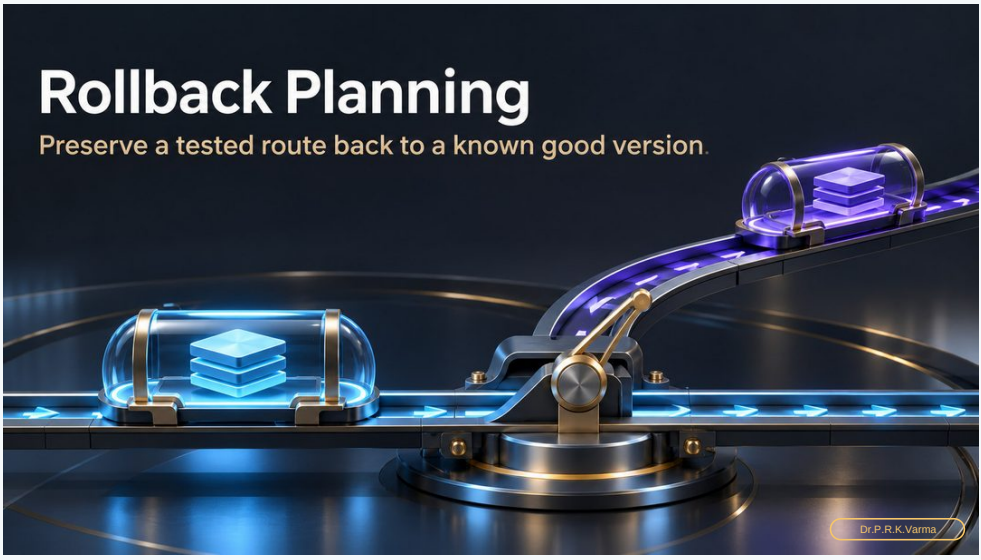
An adapter is trained against a particular base model and often expects matching dimensions, tokenizer and target modules. Record both versions and revalidate any new pairing.



Evaluation Before Promotion

PRESENTER NOTES

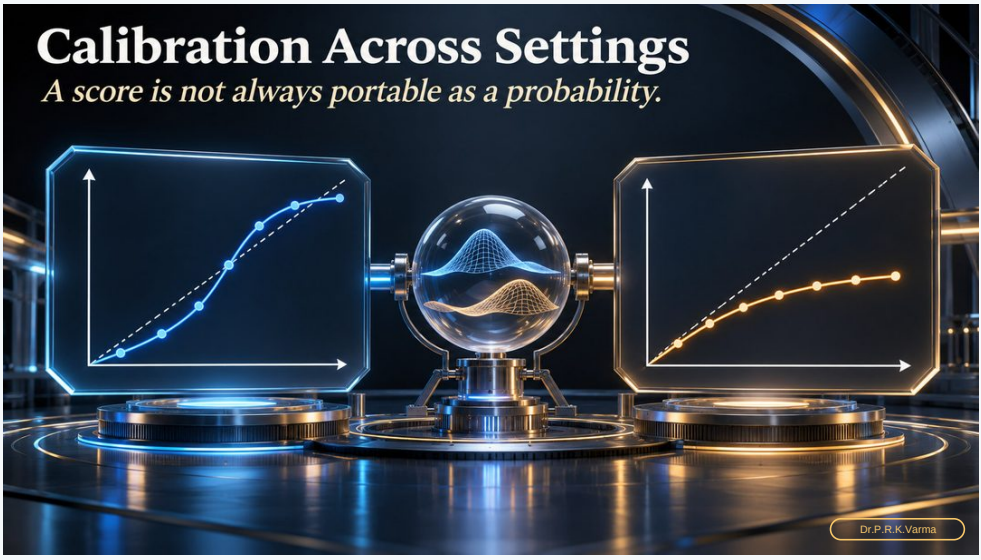
Define evaluation cases, critical slices, safety and quality criteria before reviewing results. Compare the candidate with the deployed baseline under the same conditions. Record who approved promotion and why.



Rollback Planning

PRESENTER NOTES

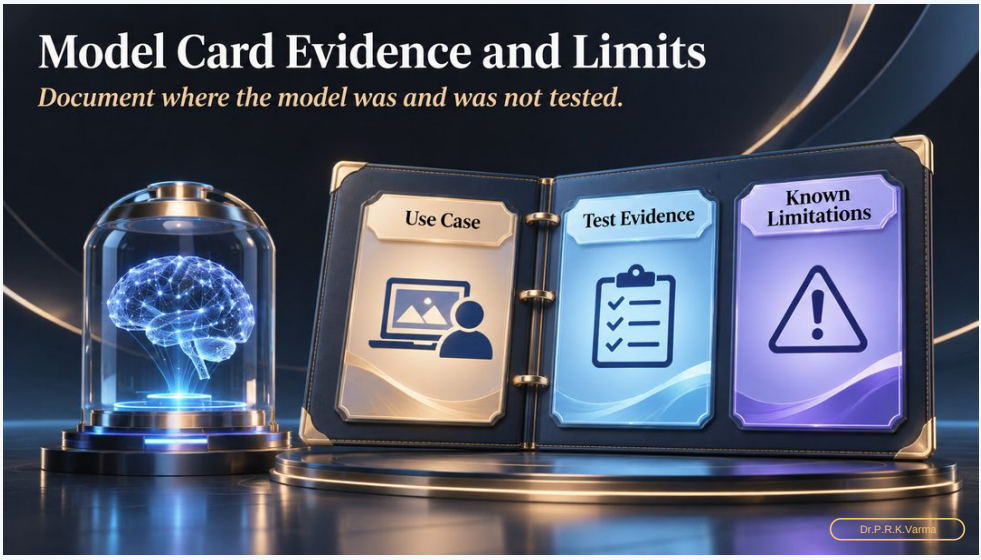
Keep a known-good artifact with compatible dependencies and configuration. Test reverting traffic, data or schema changes and assess whether rollback can safely restore service before rollout.



Calibration Across Settings

PRESENTER NOTES

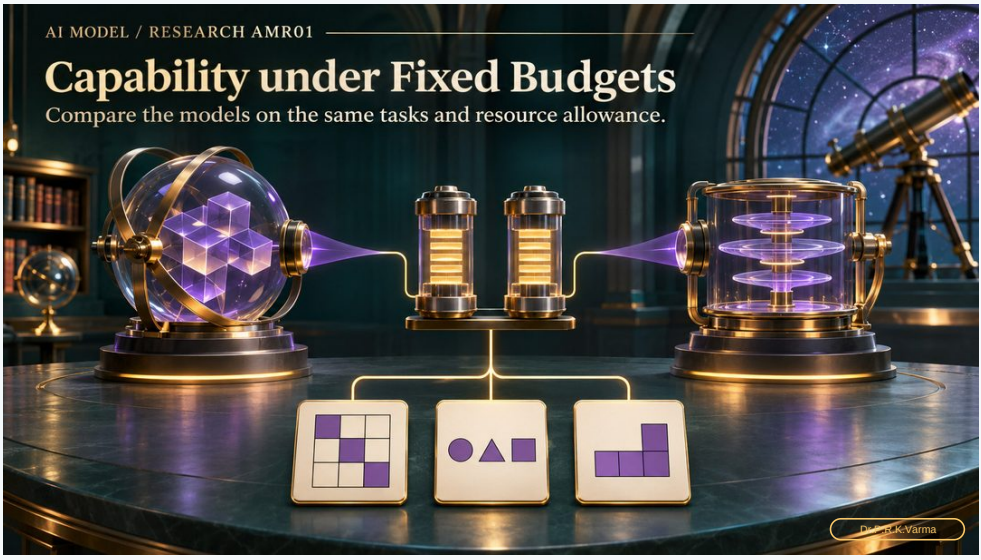
Calibration can change with dataset, user population or time period even if model weights are unchanged. Validate scores against outcomes in the intended setting before using them as probabilities or threshold inputs.



Model Card Evidence and Limits

PRESENTER NOTES

Summarize intended use, evaluation scope and metrics, known limitations and relevant subgroup results. Separate observed evidence from extrapolation and record the artifact and data versions used.

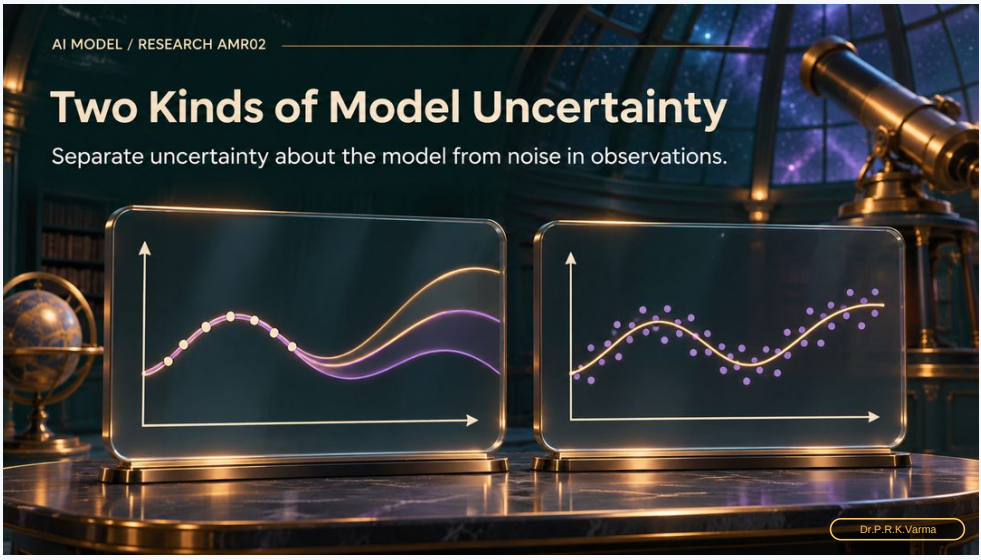


Capability under Fixed Budgets

PRESENTER NOTES

Use the same task mix, tuning allowances, and inference budget for each model. Report how performance changes with resources and separate preselection or search costs from the cost of the final run.

Original research



Two Kinds of Model Uncertainty

PRESENTER NOTES

Aleatoric uncertainty describes variability in observations conditional on the modeled inputs. Epistemic uncertainty reflects incomplete knowledge of the model and may decrease with informative data. The split is conditional on the modeling assumptions.

Original research

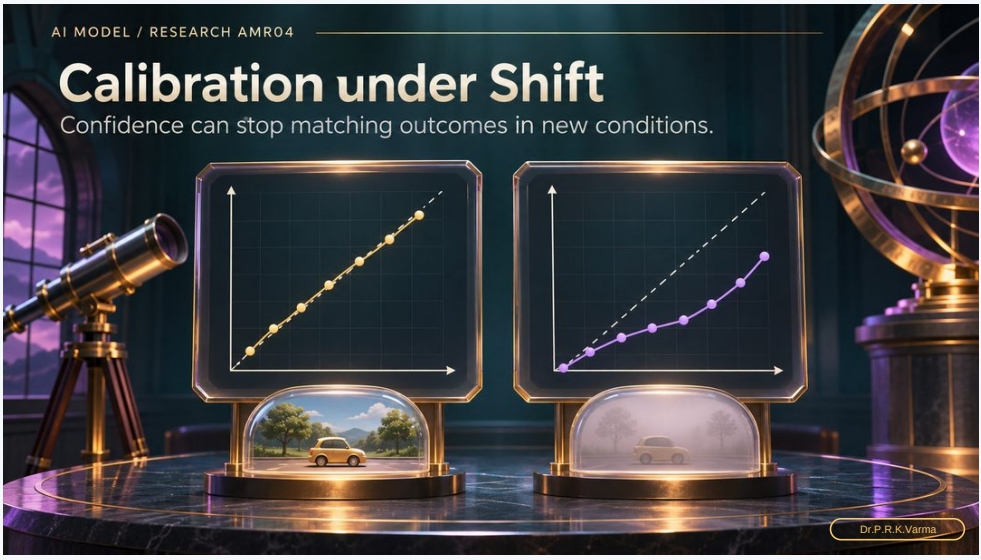


Scaling Across Architectures

PRESENTER NOTES

Architecture can influence performance and efficiency at different scales. Match data, hardware, and training allowances where practical, then measure quality and resource use across multiple sizes rather than extrapolating from one comparison.

Original research



Calibration under Shift

PRESENTER NOTES

A model calibrated on one distribution can be miscalibrated on another. Recheck reliability and error patterns across the intended operating conditions, and report uncertainty for sparsely represented conditions.

Original research



Model Edit Locality

PRESENTER NOTES

For a benign factual edit, test whether the desired answer changes, whether related paraphrases inherit the correction, and whether unrelated behavior stays stable. Locality and generalization are separate properties.

Original research



Model Merging Compatibility

PRESENTER NOTES

Task vectors derived from a shared base provide one setting for model merging. Compare the merged result against both source models on retained and new tasks; compatible shapes alone do not ensure that capabilities combine.

Original research



Audit Evidence and Aggregate Scores

PRESENTER NOTES

Audit a score with specific cases, known expectations, and behavior under carefully chosen variations. A behavioral test suite can identify different failure modes even when aggregate accuracy looks similar.

Original research



Evidence Tiers in Model Reporting

PRESENTER NOTES

Report what a piece of evidence can and cannot support. A demonstration shows feasibility in one example, a benchmark addresses a defined protocol, and an independent check tests whether a result holds outside the originating team's run.

Original research



Reproducibility and Provenance

PRESENTER NOTES

Record the model snapshot, evaluation protocol, dataset and tool versions, and exact settings. Link each reported result to those artifacts so another reviewer can trace it and attempt reproduction.

Original research



Comparing Model Versions Over Time

PRESENTER NOTES

Repeat a fixed set of tests across model versions and record version identifiers, dates and settings. Separate newly introduced exploratory tests from the longitudinal trend, and avoid generalizing one test-suite change to all behaviors.

Original research