Erode AI Course Syllabus

Selvakumar Murugan

December 19, 2025

Contents

Syllabus - AI / LLM Engineering with Transparent Abstractions

Summary

I am planning the course in such way that it trains students to **build, evaluate, and deploy reliable AI/LLM systems without myth, hype, or fragile intuitions**.

We don't want to focus on:

- prompt tricks,
- surface-level tooling,
- or abstract ML math disconnected from practice,

Instead we stress:

- **correct mental models**,
- **engineering discipline**,

My intention is to make students understand what these systems are, what they are not, where they fail, and how to control them. not do not merely **use** LLMs. But we still design the course around LLMs, and cover ML concepts spread over the entire syllabus.

This is not a

- an AI hype course
- a prompt engineering bootcamp

- a math-heavy ML theory course

But
- a rigorous, honest, industry-aligned program
- to build and explain AI systems.

The course covers
- probabilistic modeling
- training vs inference
- loss functions
- optimization
- generalization
- evaluation

But always framed as, **machinery behind behavior**, not explanations of intelligence. Students can later learn deeper math **without unlearning incorrect metaphors**.

Goals

Most AI/LLM courses suffer from one or more of these failures:

- Anthropomorphic explanations ("the model understands")
- Overconfidence in prompts and benchmarks
- Confusion between training and inference
- Ignoring uncertainty, failure modes, and deployment realities
- Teaching ML math that students later misuse conceptually
- Producing candidates who sound impressive but are unsafe in production

Industry consequence: Teams spend months unlearning bad intuitions, fixing fragile systems, and correcting overclaims made by well-meaning but undertrained engineers.

I'd like our course be designed explicitly to prevent that.

Core Principles

- Mental Model Hygiene Students must be rigorously trained to:
  - avoid anthropomorphism
  - distinguish model behavior from system behavior
  - treat uncertainty as inherent, not as error
  - use precise, defensible language
- Three Layer Abstractions that are Transparent Every concept is taught at three explicit levels:
  1. **Behavioral** — what it appears to do
  2. **Mechanical** — what actually happens
  3. **Formal / Mathematical** — optional depth, clearly scoped

  Students should always grasp the following to prevent long term confusion and re-learning in future:
  - where the abstraction boundary is
  - what is literal vs metaphorical, e.g: words like meaning, memory and reasoning are abused
  - what machinery sits underneath e.g: Neural networks and Transformers
- Metaphor Audits The course actively identifies and dismantles misleading metaphors such as:
  - "the model understands"
  - "the agent plans"
  - "the AI remembers"
  - "RAG gives knowledge"

  Students should learn to after going through the course:
  - detect bad metaphors
  - explain why they are dangerous
  - replace them with precise language

  This is a **very rare but critical professional skill even among most experienced engineers**.

- Big picture view instead of erratic metaphors that needs to unlearned Students are trained to reason about:
  - pipelines
  - data flow
  - state
  - boundaries
  - failure propagation
  - glue code

  LLMs are treated correctly as, probabilistic components inside engineered systems and not as intelligent entities.

- Just-In-Time Foundations (No Prerequisites) Students can enter with:
  - no programming background
  - no ML background
  - no math background

  Foundations (Python, statistics, ML, information theory) are introduced so while avoiding intimidation:
  - **only when needed**
  - **only to the depth required**
  - **with explicit warnings against misuse**

What the Syllabus Covers (High-Level)

Weeks 1–3: Core Reality of LLMs

- What LLMs actually are (probabilistic text generators)
- Prompting as conditioning, not programming
- Verification, tools, and uncertainty

Week 4: Programming Fundamentals (AI-Assisted)

- Code vs execution
- Determinism vs probability
- Control flow, functions, and verification

Week 5: APIs and Inference

- Stateless inference
- Cost, latency, and failure
- Secrets and configuration

Week 6: Guardrails and Structured Outputs

- Validation, retries, and contracts
- Why structure is enforced externally

Week 7: Retrieval-Augmented Generation (RAG)

- Retrieval vs understanding
- Embeddings as geometry
- Why RAG can help or harm

Week 8: Evaluation and Measurement

- Metrics as estimates
- Variance, sampling, and uncertainty
- Why benchmarks lie

Weeks 8.5–10.5: Traditional ML Foundations (Optional but Integrated)

- Learning as parameter optimization
- Loss functions and gradient descent (machinery, not mysticism)
- Overfitting, generalization, and evaluation
- Explicit mapping to standard ML theory **without adopting its pedagogy**

Week 9: Agents and Multi-Step Systems

- Agents as loops and state machines
- Error compounding and termination
- Why autonomy is fragile

Week 10: End-to-End System Design

- Pipelines and responsibility boundaries
- Failure propagation
- Refactoring as design

Week 11: Deployment and Scaling

- Production realism
- Rate limits, backpressure, observability
- Why "it works locally" means nothing

Week 12: Capstone & Professional Framing

- Integrated system build
- Honest evaluation
- Interview-ready explanations
- Defensible claims

Syllabus - Machine Learning with Transparent Abstractions

Everything from above applies, I copy pasted the structure and making changes only relevant to the materiak

Summary

I am planning this course in (case you want more traditional ML material) in such a way that it trains students to **build, evaluate, and deploy reliable machine learning systems without myth, hype, or fragile intuitions**.

We do **not** want to focus on:

- algorithm catalogs,
- leaderboard chasing,
- or abstract ML mathematics disconnected from real systems.

Instead, we stress:

- **correct mental models**,
- **engineering and evaluation discipline**,
- **system-level thinking**, and

It is not about merely **run models** or **train algorithms**.

While the course uses familiar ML tasks (classification, regression, clustering), the emphasis is always on **behavior, failure modes, and integration**, not on mysticism.

This is not a

- "learn ML algorithms in 12 weeks" course
- Kaggle-style competition bootcamp
- math-heavy statistical learning theory class

The course covers (all from AI/LLM course plus)

- deployment and monitoring (optional)

Core Principles

- Metaphor Audits The course actively identifies and dismantles misleading ML metaphors such as:
  - "the model learns concepts"
  - "the model understands patterns"
  - "the algorithm figures it out"
  - "high accuracy proves correctness"

Students should, by the end of the course, be able to:

  - detect misleading metaphors
  - explain why they are dangerous
  - replace them with precise, operational language

  This is a **rare but critical professional skill**, even among experienced practitioners.

- Big-Picture Systems Thinking Over Algorithm Worship Students are trained to reason about:

  - data pipelines
  - preprocessing and feature boundaries
  - training vs inference separation
  - evaluation loops
  - deployment constraints
  - monitoring and drift
  - failure propagation

  ML models are treated correctly as probabilistic components inside engineered systems, not as intelligent entities.

- Just-In-Time Foundations (No Prerequisites) Students can enter with:

  - no programming background
  - no ML background
  - no math background

  Foundations (Python, statistics, probability, linear algebra, optimization) are introduced:

  - **only when needed**
  - **only to the depth required**

What the Syllabus Covers (High-Level)

Weeks 1–2: What Machine Learning Actually Is

- ML as function approximation under uncertainty
- Data and labels as task specification
- Why models do not "understand" data

Week 3: Just-In-Time Python for ML Evidence

- Loading, inspecting, and summarizing data
- Deterministic code vs stochastic ML behavior
- Evidence generation, not "model magic"

Week 4: Supervised Learning Foundations

- Training/validation/test splits
- Loss functions as human-defined objectives
- Training as optimization, not learning concepts

Week 5: Baselines and Simple Models

- Linear models and kNN
- Why baselines are safety checks
- Interpreting errors instead of chasing accuracy

Week 6: Optimization Machinery

- Gradient descent as local numeric improvement
- Hyperparameters as tradeoffs
- Why optimization can succeed and still fail you

Week 7: Generalization and Overfitting

- Why training success lies
- Regularization and early stopping
- Capacity vs data

Week 8: Evaluation and Measurement

- Metrics as estimates, not proofs
- Variance, repeated runs, and slicing
- Why benchmarks mislead

Week 9: Unsupervised Learning and Representation

- Clustering and PCA
- Similarity vs meaning
- Structure creation, not discovery of truth

Week 10: End-to-End ML Pipelines

- Data > features > model > evaluation > decision
- Reproducibility and versioning
- Failure propagation

Week 11: Deployment, Monitoring, and Drift

- Dataset shift and real-world change
- Monitoring signals and alerts
- Why retraining is not a solution by itself

Week 12: Capstone & Professional Framing

- Integrated ML system build
- Honest evaluation and failure analysis
- Interview-ready, defensible explanations