Data Science & Machine Learning

Curriculum Design Report

Implementation-First Pedagogical Framework

February 2026

Executive Summary

This document captures the design and rationale for a comprehensive Data Science and Machine Learning curriculum. The curriculum takes an implementation-first approach where students build algorithms from scratch before learning theory, demystifying ML as "just loops + math + data."

Core design principles:

- Build before theorize: Students implement working algorithms before formal instruction
- Consistent datasets: Same problem (penguin classification, house prices) across multiple algorithms enables direct comparison
- From scratch implementations: No ML libraries—students write Euclidean distance, splitting criteria, gradient descent manually
- One hour presentations: Maximum 10 content slides, focused on hands-on doing rather than comprehensive theory
- Unified mental model: All ML paradigms understood through a "labeling" framework

Core Pedagogical Philosophy

The "Labeling" Framework

The curriculum adopts a unified mental model: **all machine learning is about labeling data**. This provides cognitive coherence across paradigms:

- Supervised Learning: Label new data given labeled training data
- Unsupervised Learning: Label new data by assigning to discovered patterns
- Reinforcement Learning: Label actions to maximize rewards

This framing meets students where they are. When they see "predict house prices" (regression) or "classify species" (classification), they understand both as forms of labeling. Even cluster assignment (unsupervised) fits this mental model naturally.

Implementation-First Approach

Students write algorithms from scratch before formal theory:

- Write Euclidean distance calculations manually
- Loop through datasets explicitly
- Implement gradient descent, splitting criteria, clustering from first principles

This demystifies ML. When students see a neural network is "just matrix multiply + adjust weights," the intimidation disappears. The goal is not mastery—it's

conceptual accessibility. DS/ML/AI at fundamental levels is completely understandable, not magic.

Epistemological Foundation

Learning progresses from *incorrect to less incorrect*, not from correct-general to correct-specific. Early conceptual models may be technically imprecise but pedagogically effective. For instance:

- Introducing neural networks early in the curriculum (after just kNN and decision trees) contradicts "traditional" sequencing but addresses student expectations shaped by AI hype
- Simplifying "neural networks are prominent in modern AI" into "neural networks aka AI" meets students where their mental models already are, enabling later nuance

This approach acknowledges that both human and machine learning systems (like LLMs) operate probabilistically with incomplete information. Perfect correctness is unachievable; pragmatic usefulness is the goal.

Consistent Datasets Enable Comparison

Rather than introducing new problems for each algorithm, the curriculum uses:

- Penguin classification for all classification algorithms (kNN, decision trees, logistic regression, random forests, neural networks)
- House prices for all regression algorithms (linear, polynomial, kNN-regression, tree-based, neural networks)
- MNIST handwritten digits for convolutional neural networks

This repetition is pedagogically powerful. When students see kNN achieves 85% accuracy but is slow, decision trees get 80% but are interpretable, and random forests reach 88% but take hours to train—all on the *same penguin data*—they develop genuine algorithm selection intuition.

Complete Curriculum Map

Foundation Sequence (All Students)

ML101 (= kNN101): Machine Learning via kNN Speed Run

- Hook: Classify penguins using k-Nearest Neighbors in first hour
- Implement Euclidean distance from scratch
- Loop through labeled penguins, classify unknown penguin
- Reflection: This was supervised learning; two other paradigms exist (unsupervised, reinforcement)
- Outcome: Students have successfully done ML before understanding taxonomy

DS101: How to Run a Data Science Project

- Formalizes the process students just experienced in ML101
- Complete project lifecycle: Problem definition → Data collection & cleaning → EDA → Model selection & training → Evaluation → Deployment
- Common pitfalls and how to avoid them
- Bridges to algorithm selection: "You know HOW to run projects—now how do you CHOOSE which approach?"

Algorithm Implementation Series

Current planned sequence:

ML101 → DS101 → DTree101 → NN101 → KMeans101 → LogReg101 → RForest101 → ML102

Each X101 presentation follows the same structure: 10 content slides maximum, hands-on implementation, "Hello World" version of that algorithm.

Classification Track (Penguin Dataset)

ML101/kNN101 (covered above)

DTree101: Decision Trees from Scratch

- Classify same penguins using decision trees
- Implement splitting criteria (information gain, Gini impurity)
- Understand interpretability vs accuracy tradeoffs

NN101: Neural Networks from Scratch

- Classify same penguins using simple dense neural network
- Demystify early: "Neural networks aren't magic—just matrix math + loops"
- Understand backpropagation fundamentals
- Strategic placement: Introduced early to address student expectations from AI hype, before fear accumulates

LogReg101: Logistic Regression from Scratch

- Classify same penguins using logistic regression
- Implement sigmoid function, gradient descent
- After seeing NNs, LogReg feels like "simplified neural network with one layer"

RForest101: Random Forests from Scratch

- Classify same penguins using ensemble of decision trees
- Understand bagging, feature randomness
- Builds naturally on DTree101 foundation

ML102: Classification Algorithms Compared

- Reflection on five algorithms applied to same penguin dataset
- Direct comparison: accuracy, speed, interpretability, training cost
- Decision frameworks: When to choose which algorithm for new problems
- Case studies with concrete trade-off discussions

Regression Track (House Price Dataset)

LinReg101: Linear Regression from Scratch

- Predict house prices using linear regression
- Implement loss functions, gradient descent
- Foundation for understanding optimization

PolyReg101: Polynomial Regression from Scratch

- Predict same house prices with polynomial features
- Understand non-linear relationships, regularization (Ridge/Lasso)
- Overfitting danger zone visualization

kNN-Reg: k-Nearest Neighbors Regression

- Predict same house prices using distance-based approach
- Same algorithm as ML101, different output type
- Compare: When is distance better than fitting a line?

DTree-Reg: Tree-Based Regression

- Predict same house prices using decision tree regression
- Piecewise constant predictions vs smooth curves

NN-Reg: Neural Network Regression

- Predict same house prices using neural networks
- Demonstrates NNs work across paradigms (not just classification)

ML103: Regression Algorithms Compared

- Same structure as ML102 but for regression
- When does linear work? When do you need trees? When justify NN complexity?

Unsupervised Track

KMeans101: k-Means Clustering from Scratch

- Cluster penguins WITHOUT using species labels
- "What if we didn't know the species—could we discover them?"
- Palette cleanser in curriculum: Breaks "supervised algorithm fatigue"
- Core: Group similar things + recalculate centers

PCA101: Principal Component Analysis from Scratch

- Dimensionality reduction (future addition)
- Could use on penguins (reduce features before classification)

- Could use on MNIST (visualize 784 dimensions → 2D)

ML104: Unsupervised Methods Compared
- Comparison framework for unsupervised approaches

Advanced/Specialized (Future)

CNN101: Convolutional Neural Networks from Scratch
- MNIST handwritten digit recognition
- Why convolutions matter for images
- Strategic sequencing: After NN101 established "NNs are just another algorithm," CNN101 shows specialized architecture for images
- Could circle back: "Would CNN help with penguin images?"

RNN101: Recurrent Neural Networks
- Future: Time series, text, sequential data

Key Design Decisions & Rationale

Algorithm Sequencing Rationale

The sequence *ML101 → DS101 → DTree101 → NN101 → KMeans101 → LogReg101 → RForest101 → ML102* contradicts "classical" difficulty ordering but optimizes for

conceptual comfort rather than mathematical sophistication:

1. **ML101 (kNN):** Most intuitive - "find similar things, vote." Core: loop + distance calculation.
2. **DS101:** Formalizes the process they just experienced.
3. **DTree101:** Visual/interpretable - "ask questions until you know." Core: if/then + splitting rule.
4. **NN101:** Demystify EARLY before fear sets in. Core: matrix multiply + adjust weights. After NNs, everything else feels approachable. Strategic: Addresses student expectations from AI hype.
5. **KMeans101:** Palette cleanser - prevents "supervised algorithm fatigue." Shows unsupervised isn't different world. Core: group + recalculate centers.
6. **LogReg101:** After seeing NNs, LogReg feels like "oh, just one layer." Core: sigmoid + gradient descent.
7. **RForest101:** Natural build on trees. Core: many trees + voting. "What if we asked LOTS of questions?"
8. **ML102:** Reflection and comparison after hands-on experience with multiple algorithms.

Why Neural Networks Early?

Traditional curricula place neural networks late ("advanced topics"). This curriculum places them fourth for strategic reasons:

- **Addresses student expectations:** Students arrive with "AI = neural networks" mental models from media exposure. Introducing NNs early validates their interest.
- **Prevents fear accumulation:** If NNs come late, students build up intimidation. Early exposure shows "it's just matrix math."
- **Simplifies later content:** After NNs, logistic regression feels simple ("one-layer NN"). This inverts traditional difficulty perception.
- **Maintains engagement:** Students want to build "AI." Delivering early maintains motivation through subsequent algorithms.

Presentation Length: 10 Slides Maximum

Weekly tech forum format: 1 hour total. Experience shows 10-12 content slides achievable, 15 absolute maximum including intro/agenda/references. This constraint enforces:

- Focus on doing over comprehensive theory
- "Hello World" versions of algorithms rather than production-ready implementations
- Emphasis on core concepts that enable future self-directed learning

This is pedagogically sound: research on cognitive load supports focused, constrained learning experiences over encyclopedic coverage.

Resolved Design Debates

Debate: Is "Labeling" Framework Too Simplistic?

Initial concern: Using "label" for all three paradigms obscures differences. Unsupervised learning doesn't predict labels—it discovers structure. Reinforcement learning selects actions, not labels.

Resolution: The framework is pedagogically pragmatic. It gives learners ONE clear mental model ("ML systems map inputs to outputs") rather than requiring them to hold three distinct conceptual frameworks simultaneously. Functionally:

- Supervised: Input data → output label
- Unsupervised: Input data → cluster assignment (a form of labeling)
- Reinforcement: Input state → action selection (output labeling)

The unified framework provides cognitive coherence for beginners. Nuance can be added later as students gain experience.

Debate: Neural Networks Are Not Reinforcement Learning Algorithms

Initial concern: Listing "CNNs, RNNs" as examples under Reinforcement Learning is factually incorrect. These are architectures usable across all paradigms, not RL-specific algorithms (Q-learning, Policy Gradients, etc. are RL algorithms).

Resolution: This conflation was removed from curriculum materials. However, the underlying pedagogical principle remains valid: meet students where they are. When students google "AI," they immediately encounter "transformers" described as "neural networks." The curriculum acknowledges this by:

- Introducing NNs early (NN101) as "another classification algorithm" using penguins
- Later showing specialized architectures (CNN101 for images, RNN101 for sequences)
- Explicitly teaching that NNs are used ACROSS paradigms (supervised, unsupervised, reinforcement)

This approach validates student interest ("I want to learn about AI/neural networks") while building correct understanding incrementally.

Debate: Learning as "Incorrect to Less Incorrect"

Initial concern: Teaching with intentional simplifications creates misconceptions. Students should learn "correct" models from the start.

Resolution: Both human and machine learning are fundamentally probabilistic processes with incomplete information. Perfect correctness is unattainable—even LLMs operate probabilistically from imperfect training data. The curriculum embraces this epistemology:

- Early mental models may be technically imprecise but pragmatically useful
- "Adding nuance" is better understood as "correcting earlier incorrectness"
- Learning progresses from less wrong to less wrong, not from correct-general to correct-specific

This philosophical foundation justifies pedagogical choices like early NN introduction and the unified labeling framework, even when these deviate from technical purity.

Implementation Details

Presentation Technology

- **Format:** MARP (Markdown Presentation Ecosystem) to create HTML presentations from Markdown
- **Interactivity:** Embedded timers, fragment animations for progressive reveal
- **Code snippets:** Provided as hints; students write actual implementations

Student Coding Expectations

Students write code from scratch with guidance:

- No ML libraries (no scikit-learn, TensorFlow, PyTorch for algorithm implementations)
- Manual implementation of core functions (Euclidean distance, gradient descent, splitting criteria)
- Standard libraries acceptable for data handling (pandas, numpy for array operations)
- Visualization with Plotly for understanding results

Example: ML101 Structure

Typical 10-slide structure for algorithm presentations:

9. Title slide
10. "What is Machine Learning?" - Definition slide with labeling framework
11. "The Problem" - Introducing penguin classification
12. "The Dataset" - Palmer Penguins overview
13. "How kNN Works" - Visual explanation
14. "Implementing Distance" - Code walkthrough
15. "Finding Neighbors" - Loop implementation
16. "Making Predictions" - Voting mechanism
17. "Results" - Accuracy on test data
18. "Reflection" - What you built, next steps

Additional slides: agenda, references, contact information

Future Considerations

Potential Curriculum Extensions

Advanced Topics Not Yet Scheduled

- Ensemble Methods: XGBoost, Gradient Boosting (beyond random forests)
- Support Vector Machines (SVMs)
- Naive Bayes classification
- Evaluation Metrics Deep Dive: Precision/Recall, ROC curves, confusion matrices
- Hyperparameter Tuning: Grid search, random search, Bayesian optimization
- Feature Engineering: Techniques and best practices
- Cross-Validation: K-fold, stratified, time series splits

Reinforcement Learning Track

Currently undeveloped. Potential structure:

- QLearning101: Q-Learning from scratch (grid world environment)
- PolicyGrad101: Policy Gradients from scratch
- ActorCritic101: Actor-Critic methods
- ML105: Reinforcement Learning Algorithms Compared

Deep Learning Specialization

- Transformer101: Attention mechanisms and transformer architecture
- GAN101: Generative Adversarial Networks
- VAE101: Variational Autoencoders
- LSTM101: Long Short-Term Memory networks

Open Questions for Future Iteration

1. Where does PCA fit?
- Before classification track (students could use PCA to reduce penguin features before classifying)?
- After classification (as reflection on "what if we didn't have labels")?
- Paired with dimensionality visualization (PCA for MNIST: 784D → 2D)?

2. Should there be "102" versions of individual algorithms?
- kNN102: Optimizing k-Nearest Neighbors (distance metrics, efficient search, weighted voting)
- DTree102: Advanced decision trees (pruning strategies, handling continuous features)
- Or keep focus on breadth (many algorithms) rather than depth (single algorithm mastery)?

3. Should there be domain-specific applications?
- NLP101: Natural Language Processing fundamentals
- CV101: Computer Vision fundamentals
- TimeSeries101: Time series forecasting
- RecSys101: Recommendation systems

4. How to handle production/deployment topics?
- MLOps101: Model deployment, monitoring, versioning
- Ethics101: Bias, fairness, interpretability in production systems

- ScaleUp101: Moving from notebook to production code

Conclusion

This curriculum takes a deliberately unconventional approach to teaching Data Science and Machine Learning. By prioritizing implementation over theory, comfort over rigor, and pragmatic usefulness over technical purity, it aims to make ML genuinely accessible to learners who might otherwise be intimidated by mathematical complexity or abstract concepts.

The core insight is simple: **ML is not magic—it's loops, math, and data**. When students write Euclidean distance from scratch, implement their own gradient descent, and build a working neural network with just matrix operations, this insight becomes visceral rather than intellectual.

The unified "labeling" framework provides cognitive coherence. The consistent use of penguin and house price datasets enables direct algorithm comparison. The early introduction of neural networks addresses student expectations while reducing intimidation. The one-hour, 10-slide constraint forces focus on essential concepts.

These design choices emerge from a deeper epistemological commitment: learning progresses from incorrect to less incorrect. Both humans and AI systems operate with incomplete information and probabilistic reasoning. Pedagogical pragmatism—meeting students where they are—takes precedence over theoretical purity.

The curriculum map provided here represents current thinking as of February 2026. It will evolve as presentations are delivered, student feedback is incorporated, and new algorithms are added. But the foundational philosophy remains stable: build first, theorize second, and never forget that the goal is *accessible understanding*, not comprehensive mastery.

Appendix A: Quick Reference

The Three ML Paradigms (Unified Framework)

- **Supervised Learning:** Label new data given labeled training data. Examples: Linear regression, kNN, decision trees, logistic regression, random forests, neural networks.
- **Unsupervised Learning:** Label new data by assigning to discovered patterns in unlabeled data. Examples: k-means clustering, PCA dimensionality reduction.
- **Reinforcement Learning:** Label actions to maximize rewards toward a goal. Examples: Q-learning, policy gradients, actor-critic methods.

Current Presentation Sequence

Foundation Track:

ML101 → DS101 → DTree101 → NN101 → KMeans101 → LogReg101 → RForest101 → ML102

Regression Track (Future):

LinReg101 → PolyReg101 → kNN-Reg → DTree-Reg → NN-Reg → ML103

Specialized/Advanced (Future):

CNN101, RNN101, Transformer101, PCA101, ML104 (Unsupervised Comparison), ML105 (RL Comparison)

Standard Datasets

- **Classification:** Palmer Penguins dataset (species prediction from physical measurements)
- **Regression:** House prices dataset (price prediction from features)
- **Image Classification:** MNIST handwritten digits (for CNN101)

Presentation Constraints

- Duration: 1 hour maximum (weekly tech forum format)
- Content slides: 10 maximum (12 achievable, 15 absolute limit with intro/references)
- Technology: MARP (Markdown → HTML presentations)
- Coding: Students write implementations from scratch (no ML libraries)
- Outcome: "Hello World" version of algorithm, not production-ready implementation

Appendix B: Design Principles Summary

1. Implementation First, Theory Second

Students build working algorithms before formal instruction. Understanding emerges from hands-on experience.

2. One Dataset, Many Algorithms

Consistent problems (penguins, house prices) across algorithms enable direct comparison of trade-offs.

3. No Black Boxes

Manual implementation of core functions. No pre-built ML libraries during learning phase.

4. Demystify Early

Neural networks introduced early to address expectations and prevent intimidation accumulation.

5. Unified Mental Model

All ML understood through "labeling" framework for cognitive coherence across paradigms.

6. Epistemological Humility

Learning progresses from incorrect to less incorrect. Early simplifications are pedagogically acceptable.

7. Constrained Scope

One-hour, 10-slide presentations force focus on essential concepts over encyclopedic coverage.

8. Meet Students Where They Are

Acknowledge existing mental models ("AI = neural networks") rather than fighting preconceptions.

9. Comparison Over Comprehensiveness

MLX0X presentations focus on algorithm comparison and trade-offs after hands-on experience with multiple approaches.

10. Accessibility Over Mastery

Goal is conceptual understanding that enables future learning, not comprehensive mastery of any single algorithm.

Document Information

Created: February 2026

Purpose: Context document for future curriculum discussions and reference for ongoing development

Status: Living document - curriculum will evolve as presentations are delivered and student feedback incorporated

Audience: Senior software developers and test leads mentoring junior developers in SAFe methodology environment

Context: This document captures extensive curriculum design discussions covering pedagogical philosophy, algorithm sequencing, implementation approach, and future extensions.