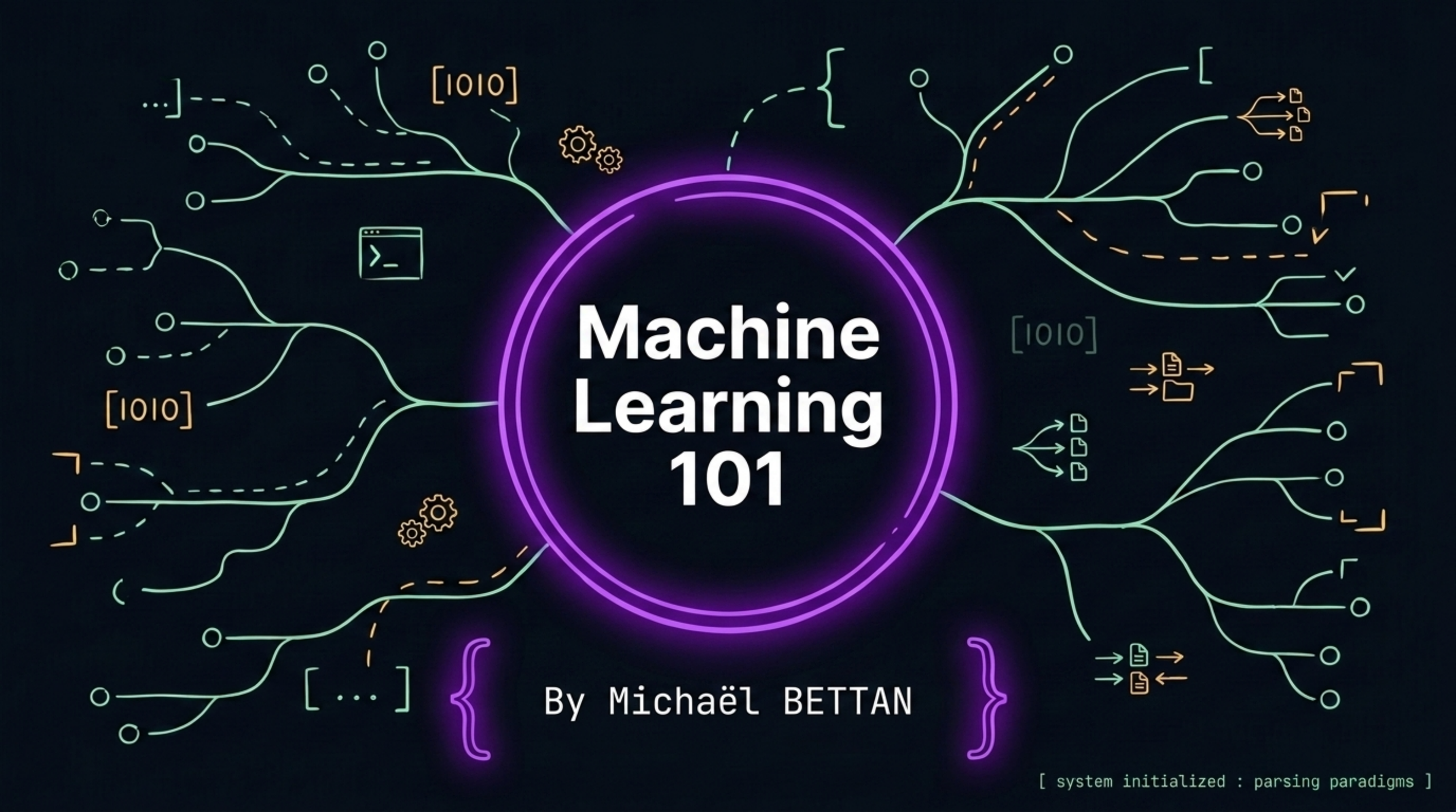
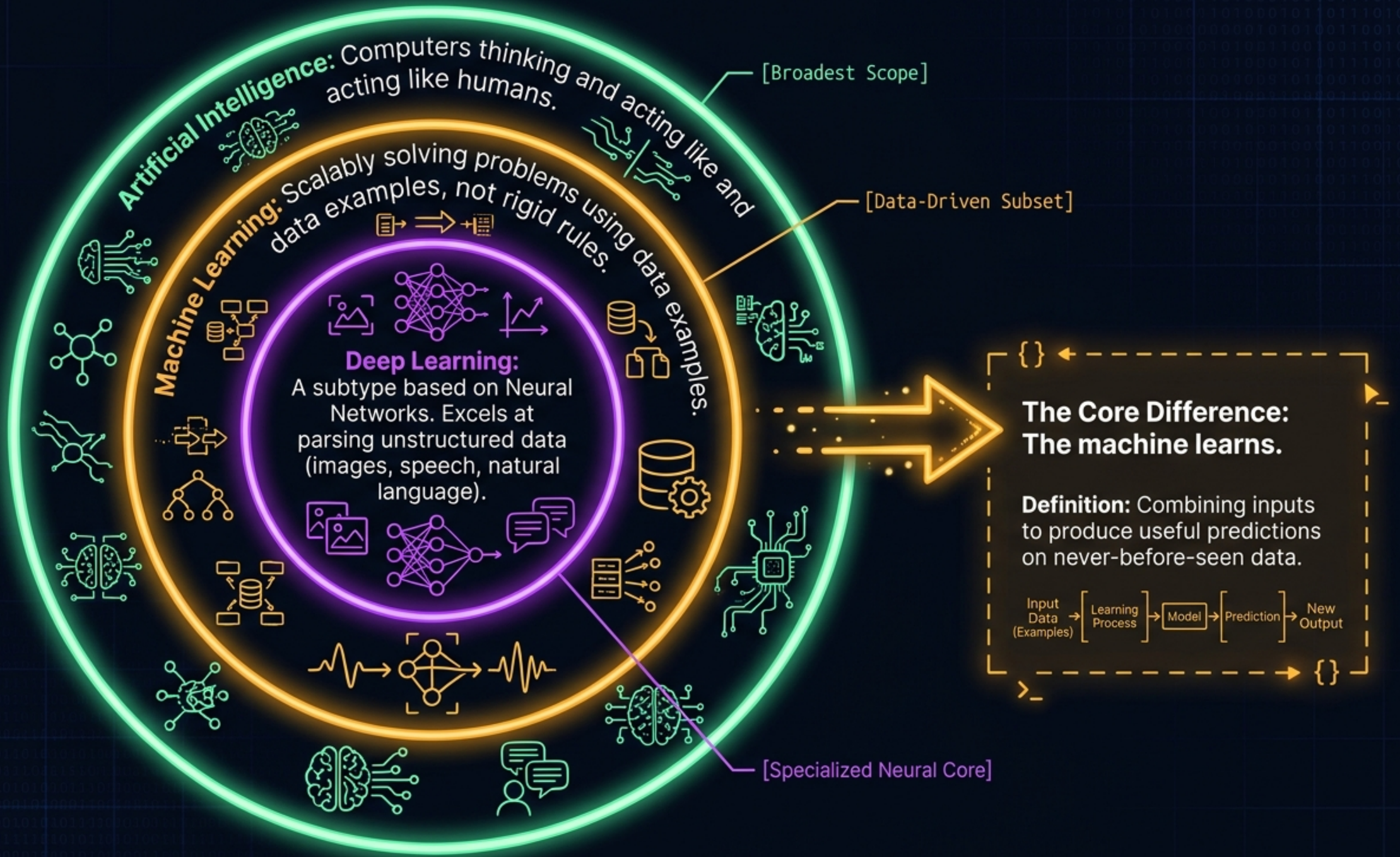

[1010]
[1010]
[1010]
[ ... ]
Machine Learning 101
By Michaël BETTAN
[ system initialized : parsing paradigms ]


Artificial Intelligence: Computers thinking and acting like humans.
[Broadest Scope]
Machine Learning: Scalably solving problems using data examples, not rigid rules.
[Data-Driven Subset]
Deep Learning:
A subtype based on Neural Networks. Excels at parsing unstructured data (images, speech, natural language).
[Specialized Neural Core]
The Core Difference:
The machine learns.
Definition: Combining inputs to produce useful predictions on never-before-seen data.
Input Data (Examples)
Learning Process
Model
Prediction
New Output

# Learning Paradigms

## Supervised Learning

Learn from examples, apply labels to data.

- **Binary Classification**: Yes/No (e.g., Credit fraud)
- **Multi-class Classification**: Discrete segments (e.g., Personas)
- **Regression**: Continuous values (e.g., Forecasting spend)
- **Matrix Factorization**: Recommenders (e.g., Netflix)

## Unsupervised Learning

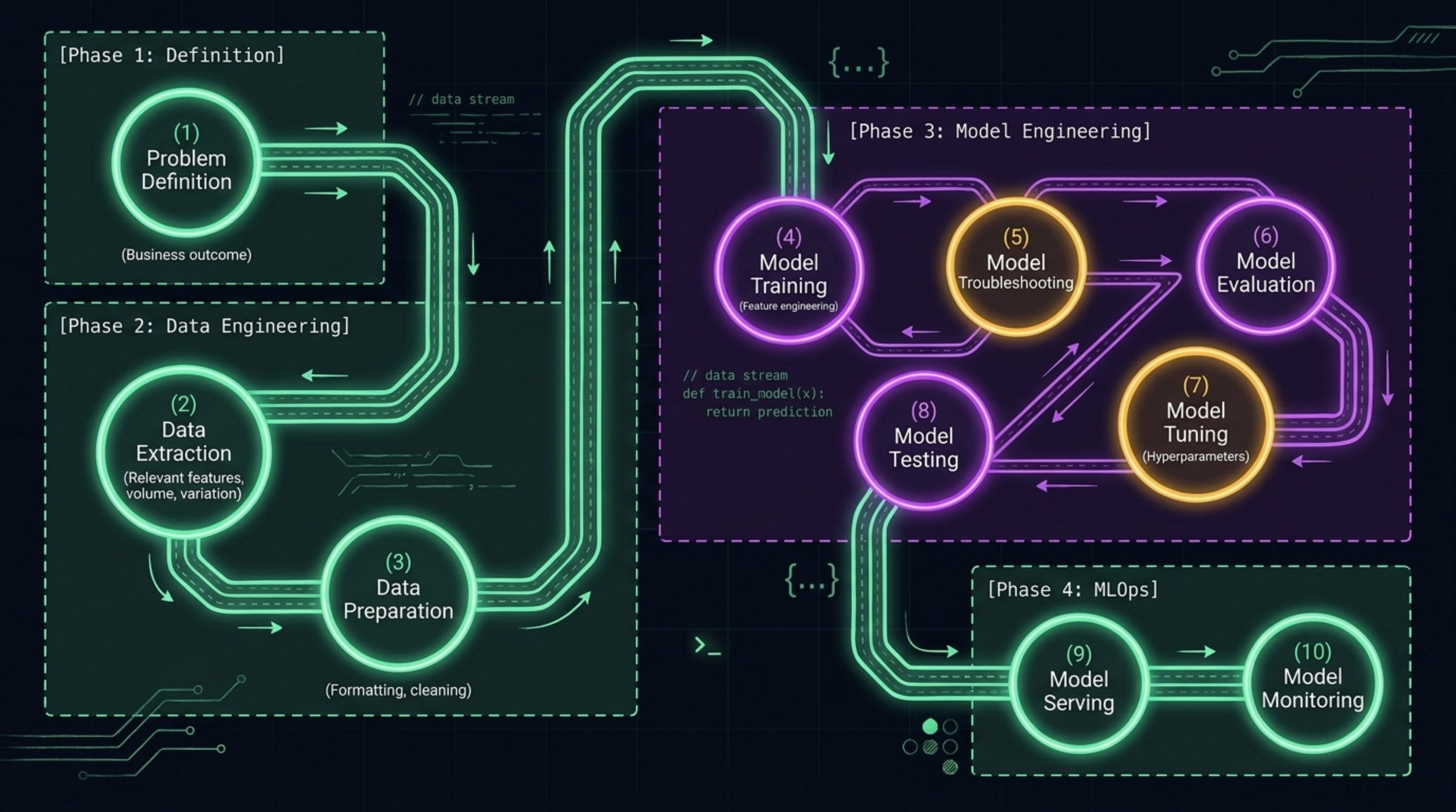
Detect known patterns without examples.

- **Clustering**: Identifying similarities in groups
- **Anomaly Detection**: Identifying abnormalities in data

## Reinforcement Learning

Learn from environment via exploration & exploitation.

- **Positive/negative** reinforcement to complete a task
- **Examples:** Chess, maze navigation


[Phase 1: Definition]
(1) Problem Definition
(Business outcome)
// data stream
[Phase 2: Data Engineering]
(2) Data Extraction
(Relevant features, volume, variation)
(3) Data Preparation
(Formatting, cleaning)
{...}
[Phase 3: Model Engineering]
(4) Model Training
(Feature engineering)
(5) Model Troubleshooting
(6) Model Evaluation
// data stream
def train_model(x):
return prediction
(8) Model Testing
(7) Model Tuning
(Hyperparameters)
{...}
>_
[Phase 4: MLOps]
(9) Model Serving
(10) Model Monitoring

// dataset split

[Raw Data Dump]

## Training Set

Labeled examples used to optimize the model.

## Validation Set

Disjoint subset used to adjust hyperparameters and prevent overfitting.

## Test Set

Provides final results on new data. The model *never* learns from these labels.

/* critical warnings */

## Data Leakage

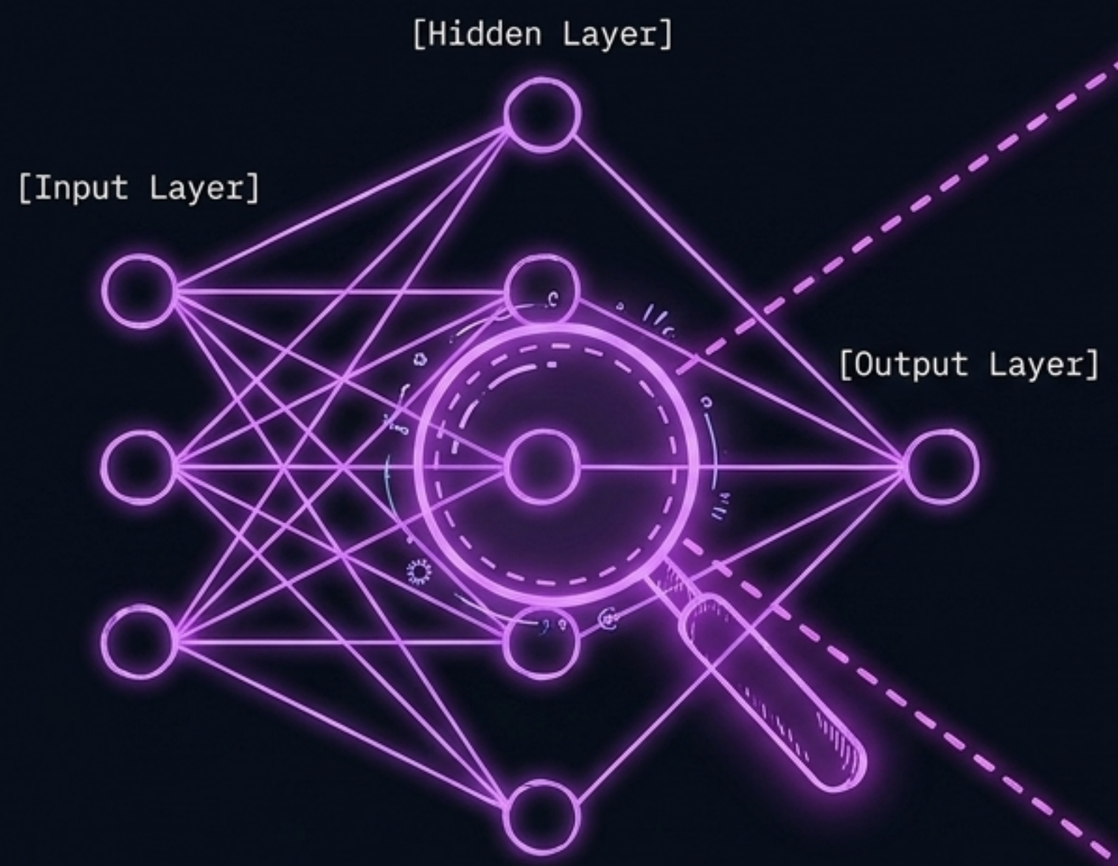
Input features contain "leaked" info about the target that won't be available in production.

Causes strong testing but weak deployment.

## Training-Serving Skew

Mismatch between input features available during training versus what is actually available at inference time.

# Anatomy of a Neural Network




**Neuron (Node):**
Combines input values to one output value.

**Activation Functions:**
Math functions (e.g., RELU, tanh) that introduce non-linearity.

$$\text{Inputs} \times \text{Weights} + \text{Bias} = \text{Output}$$

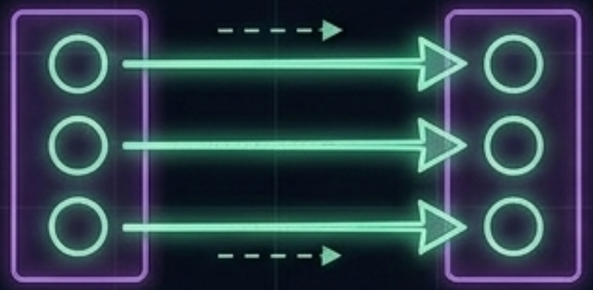
**Features:** Input data used by the ML model.

**Weight:**
Multiplication of input values.

**Label:**
The variable being predicted (target).

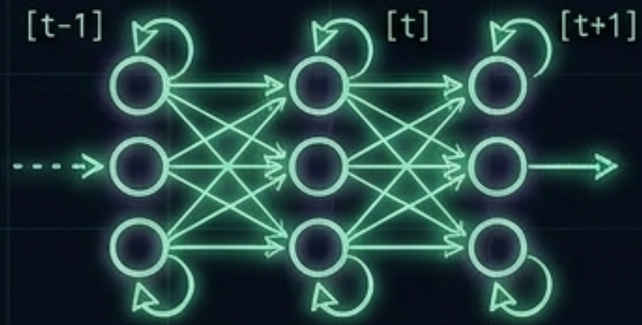
# Flow Typologies

## Feedforward (FFNN)



Information moves strictly forward.
No recursive connections.

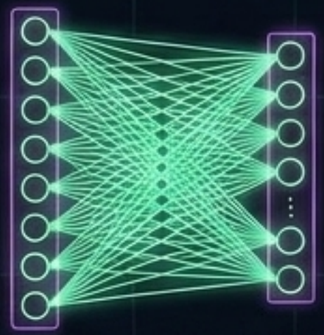
## Recurrent (RNN)




Optimized for sequential data;
previous runs feed into the next.

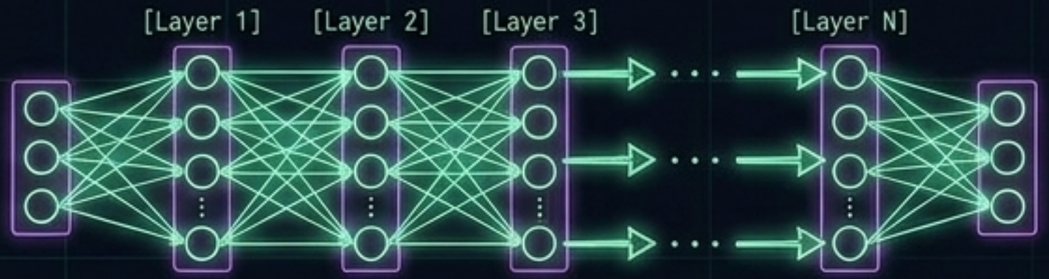
# Scale Typologies

## Wide Networks



Focus on Memorization (many features).

## Deep Networks




Focus on Generalization (many hidden layers).

**Deep & Wide Networks:** Combining both creates a perfect fit for recommendation engines.

# The Bias-Variance Tradeoff

## Overfitting

Good training, poor test.
High Variance.
Sensitive to small training fluctuations.

Cause: Memorized the data, redundant features, or noise.

High Variance

## High Error Scenario

Bias

## Ideal State

Low Bias, Low Variance.
Generalized perfectly.

Optimal Generalization

## Underfitting

Poor training, poor test.
High Bias.
Missing relationship between features and labels.
Cause: Model lacks complexity.

High Bias

Variance

# Combat Overfitting (Regularization)

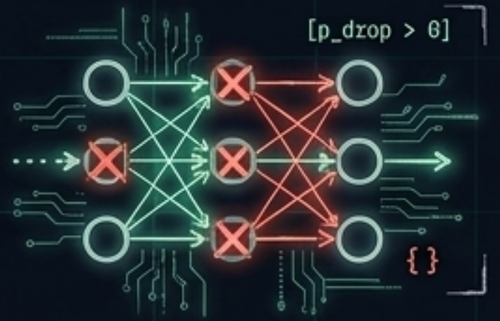
[CRITICAL: GENERALIZATION FOCUS]

## Early Stopping

Halting training when validation loss starts to increase.


## Dropout

Randomly dropping nodes during training to improve generalization.




## Max-Norm

Limits absolute magnitude of network weights.

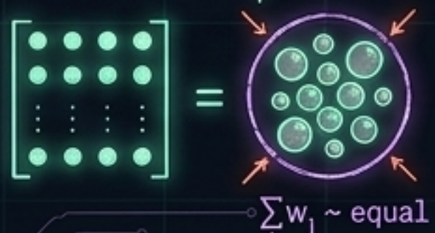

## Data Augmentation

Transforming existing examples to artificially boost dataset diversity.


## L2 Ridge

Penalizes squared weights; keeps weights approximately equal in size.

$$\|w\|_2^2 = \sum_i w_i^2$$

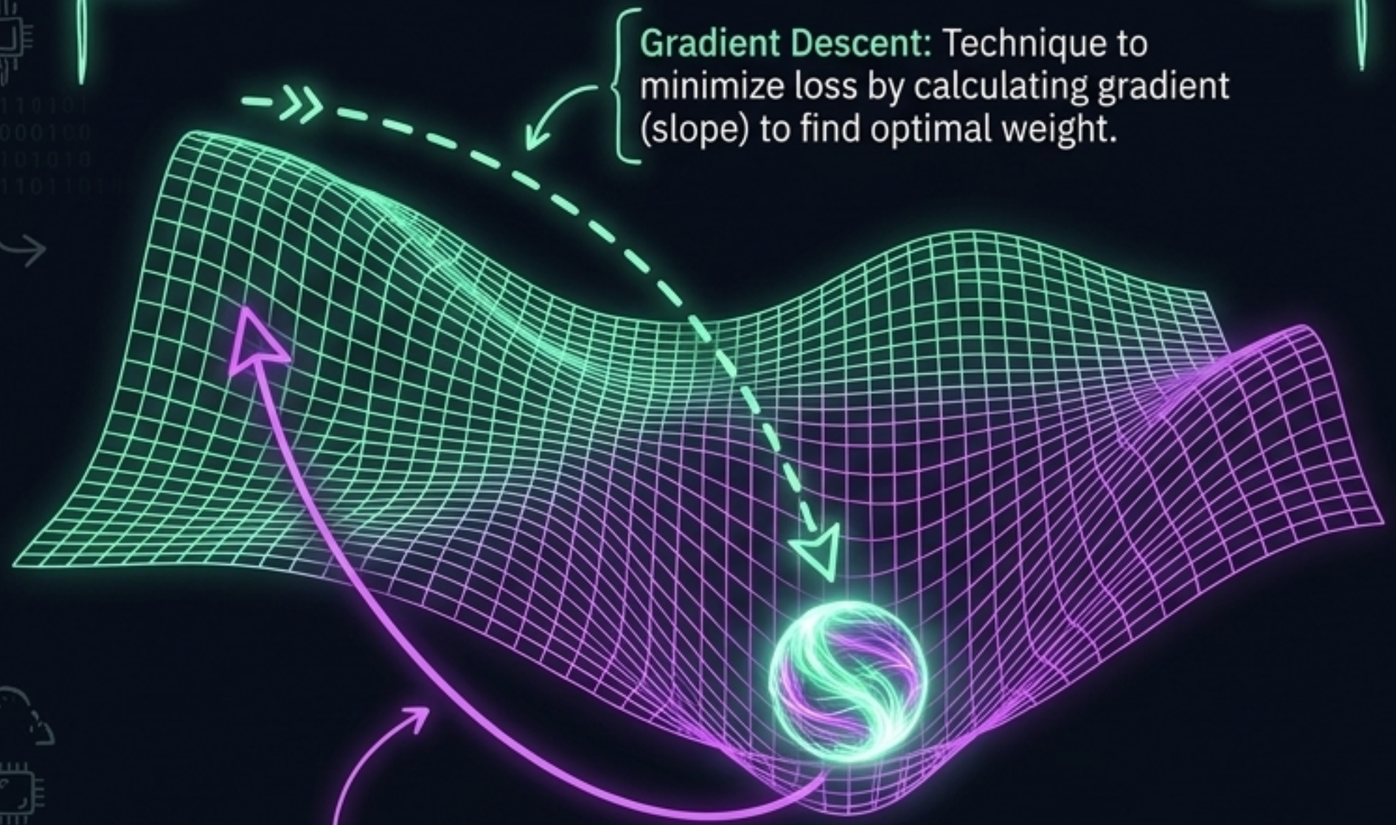



## L1 Lasso

Penalizes absolute weight values; drives least useful features to 0.

$$\|w\|_1 = \sum_i |w_i|$$

# The Optimization Engine: Finding the Answer



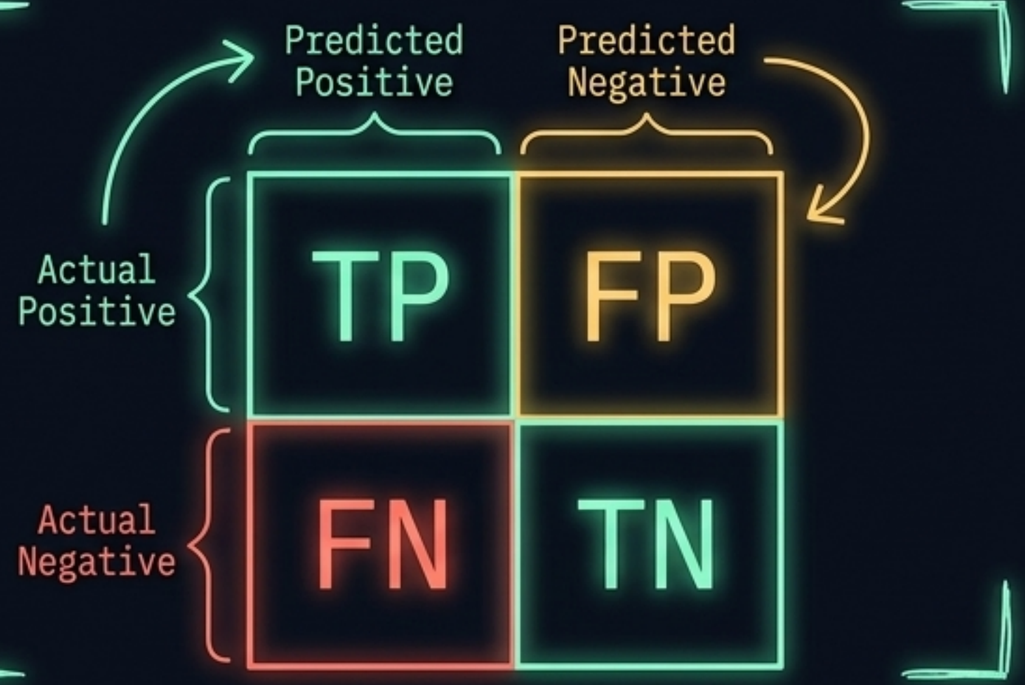
**Gradient Descent:** Technique to minimize loss by calculating gradient (slope) to find optimal weight.

**Backpropagation:** Efficient algorithm mapping input-output pairs back through the network to calculate gradient descent.

## Key Engine Variables

- **"Epoch"**: A single pass through the entire training dataset.
- **"Optimizer"**: Operation (e.g., Adam, Adagrad) that changes weights/biases to reduce loss.
- **"Learning Rate"**: Speed at which optimizers adjust weights. → Risks non-convergence if too high.
- **"Converge"**: State where loss stabilizes and optimal answer is reached.
- **"Batch Gradient Descent"**: Calculates across whole dataset (slow for large data).

# Classification Metrics




Accuracy = (TP + TN) / Total

Precision = TP / (TP + FP) [% of positive class correctly classified]

Recall = TP / (TP + FN) [% of actual positive labels identified]

F1 Score = 2 * (Precision * Recall) / (Precision + Recall)

Other: AUC PR, AUC ROC, Log loss

# Regression Metrics

Measuring mean squared errors for continuous values.

**MAE**: Mean Absolute Error

**RMSE**: Root Mean Squared Error

**RMSLE**: Root Mean Squared Logarithmic Error

# Model Serving architectures


## Online Serving

**Synchronous & Real-Time**

Quickly returns a prediction but only accepts ONE prediction request per API call.

**Use Case:** Models embedded in live applications dependent on quick, low-latency turnarounds.

## Batch Serving

**Asynchronous & Bulk**

Model waits to process ALL prediction requests before returning values.

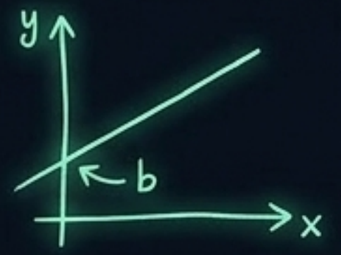
**Use Case:** Making many prediction requests at once where system latency is not the primary constraint.

**Inference:** Applying a trained model to unlabelled examples.

# Disambiguation: The Three Faces of Bias

## Math Bias

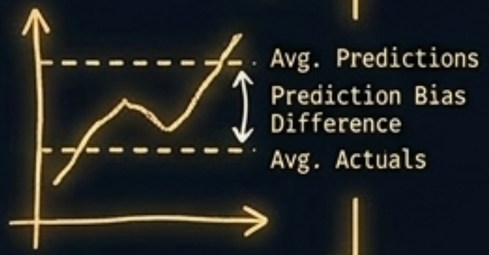
The intercept or offset from an origin. The 'b' in the equation y = wx + b. Represents the value of output given a weight of 0.




BIAS

## Prediction Bias

The mathematical difference between the average of predictions and the average of actual labels in the dataset.




## Ethics/Fairness Bias

Unintentional unfairness or stereotyping baked into algorithms or underlying training data.
Must be explicitly tested for during monitoring.

# The Human Ecosystem: MLOps Personas

**Product Manager**
Defines insights, objectives, and problem definition.

**Data Analyst**
Queries and analyzes raw data.

**Data Engineer**
Ensures clean, useful data extraction and preparation.

**Data Scientist**
Builds and trains models that work.

**ML Developer**
Integrates models into intelligent applications.

**ML Engineer**
Operates models in production (DevOps for ML).

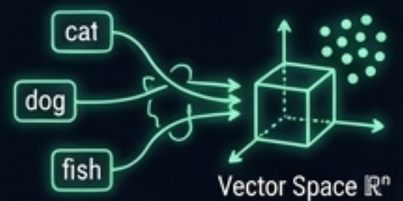
1. Problem Def.
2. Data Extraction → 3. Data Prep
4. Model Training ← 5. Model Eval
6. Model Packaging → 7. Integration
8. Deployment → 9. Monitoring → 10. Retraining

**Ops Focus (Cloud Monitoring):**
Alerting for traffic patterns, error rates, latency, and Data Skew (drift in data over time requiring model refresh).

# ML Developer Cheat Sheet: Key Concepts Glossary

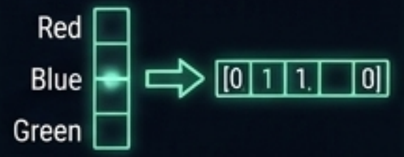
## Data Transformation

### Embeddings



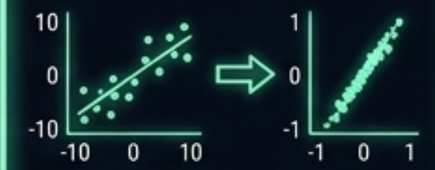

Mapping discrete objects (like words) to vectors of real numbers.

### One-Hot Encoding




Mapping attribute values to a bit in a binary array.

### Normalization




Converting numeric values to a standard range (e.g., -1 to 1).

## Architecture & Scale

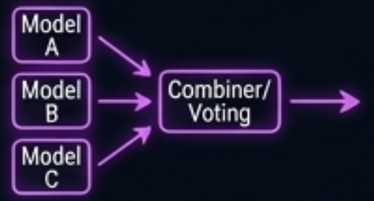
### Tensor


N-dimensional arrays of numbers; the primary data structure in ML.

### Feature Store


Rich repository to serve, share, and reuse ML features.

### Ensemble Learning




Combining predictions from multiple distinct models to solve one problem.

### Neural Architecture Search (NAS)

Automated approach to designing/selecting the best model ensemble.

```
// Core Equation
Inference = Scoring = Predictions
(Applying a trained model to unlabelled examples)
```