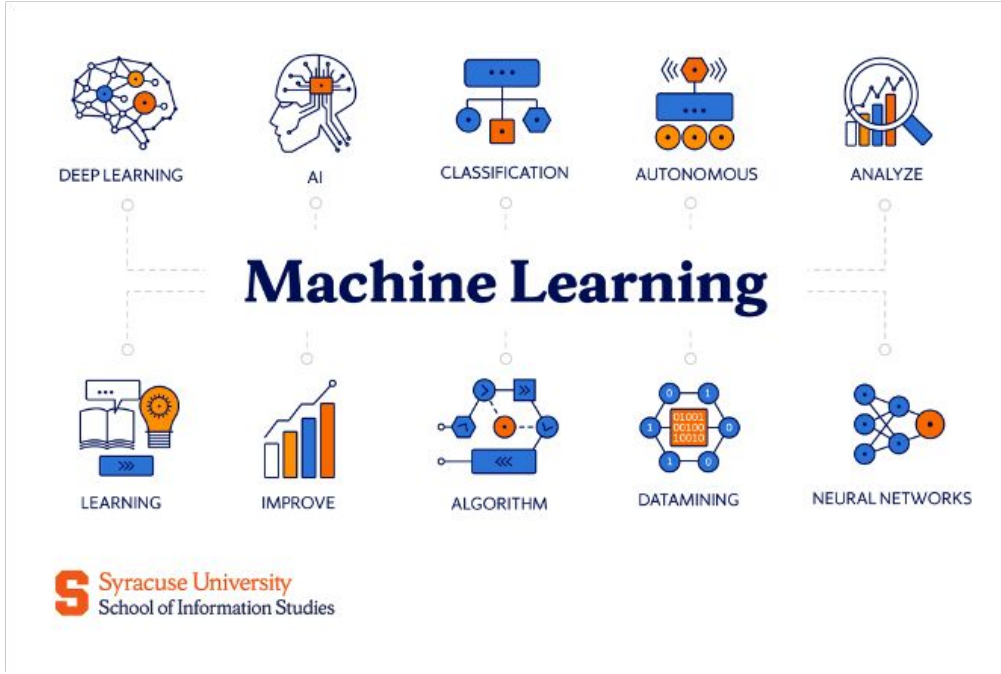

# Foundations of ML: From Architecture to Optimization

Savalan Naser Neisary[1], Leo Lonzarich[2]
The University of Alabama[1]
The Pennsylvania State University[2]

# Overview




This workshop will cover:

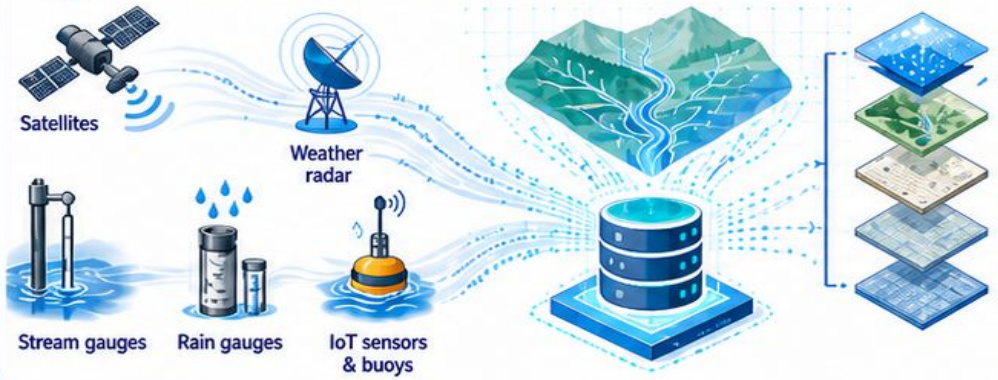
- ML fundamentals
- ML-based model development
- Hands-on reproducible end-to-end workflow

# Learning Outcomes

1. Understand major ML model types and when to use them
2. Build a complete ML workflow and data pipeline
3. Use core PyTorch tools to build and train models
4. Diagnose and improve model performance through tuning

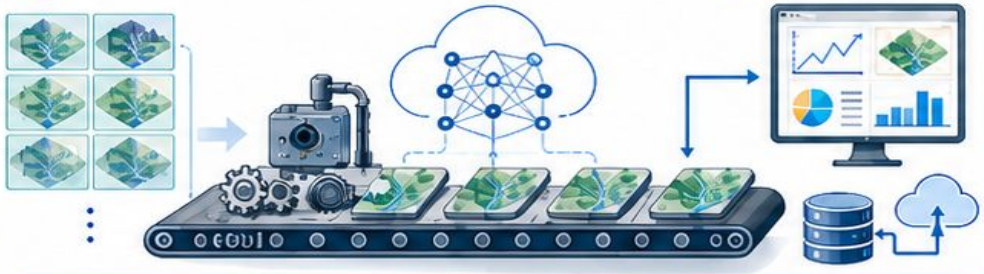
# Why machine learning?

## 1 Increasing data availability




## 2 Hydrologic systems are complex and nonlinear


## 3 Need for scalable and automated analysis

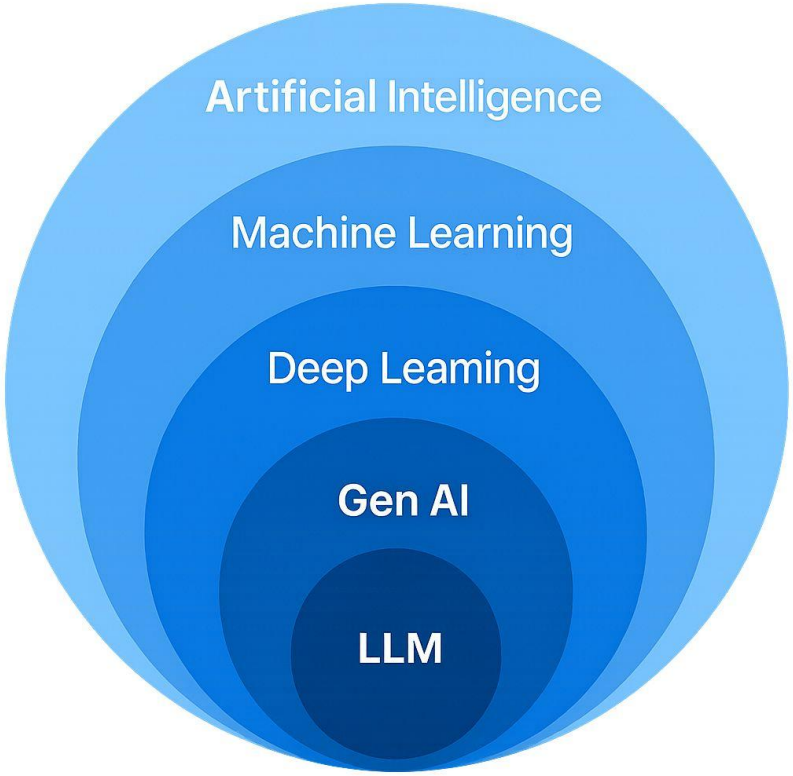


High throughput | Automated processing | Cloud scalable | Consistent & reproducible

## 4 Traditional methods can be slow or data-limited

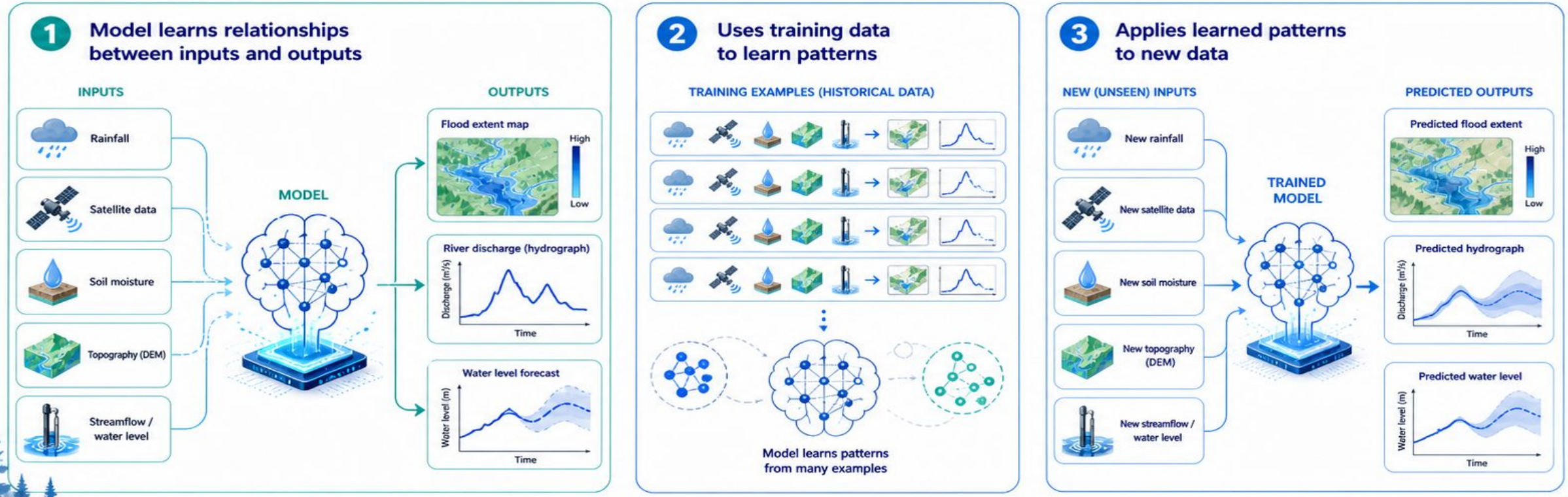
# What is Machine Learning?

- Learning patterns from data
- Uses examples instead of explicit rules
- ML learns non-linear relationships between inputs and outputs
- Hydrology example:
  - *Inputs*: SAR, DEM, rainfall
  - *Output*: flooded or non-flooded pixel



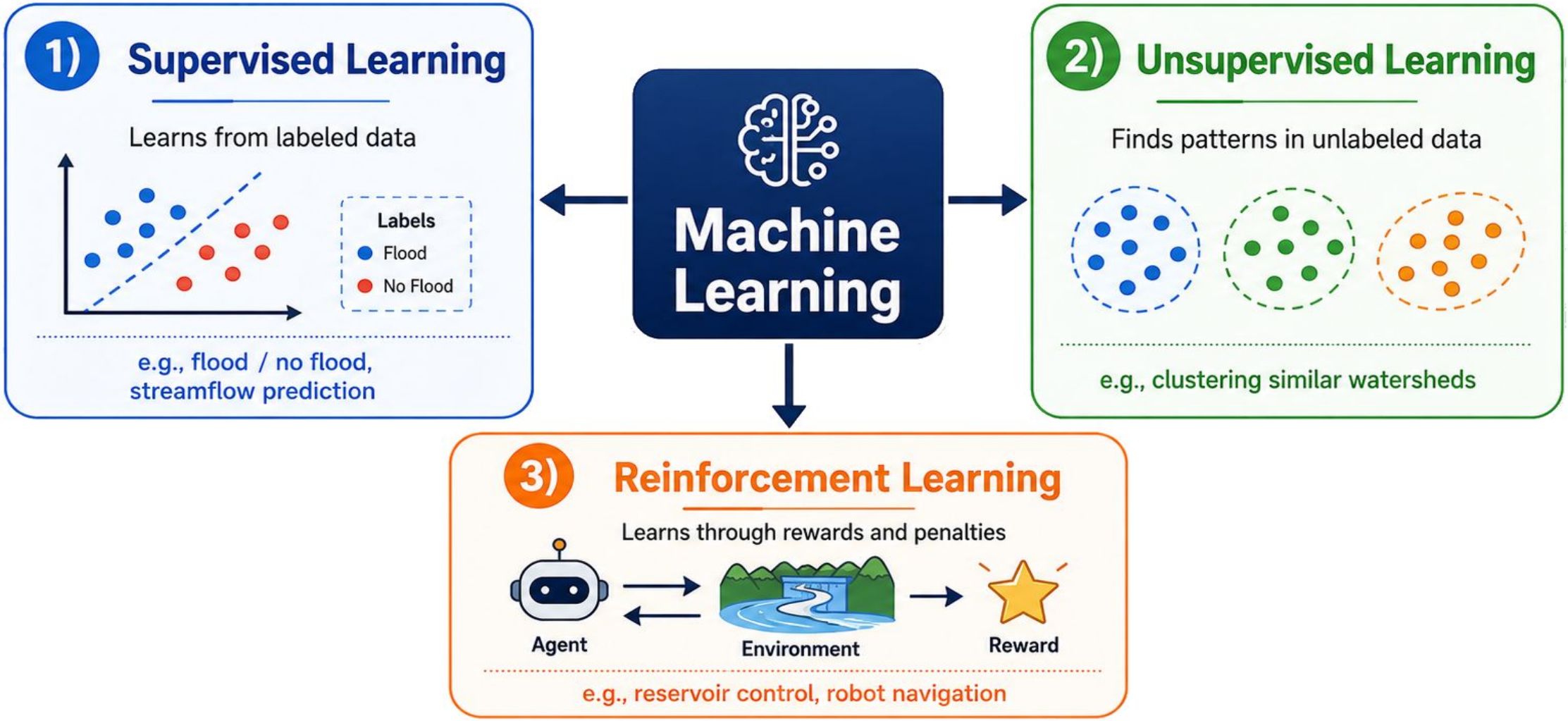

https://medium.com/@prashantgupta15/ai-evolution-predictive-ai-generative-ai-and-agentic-ai-7faa85f71b12

# How does ML work?

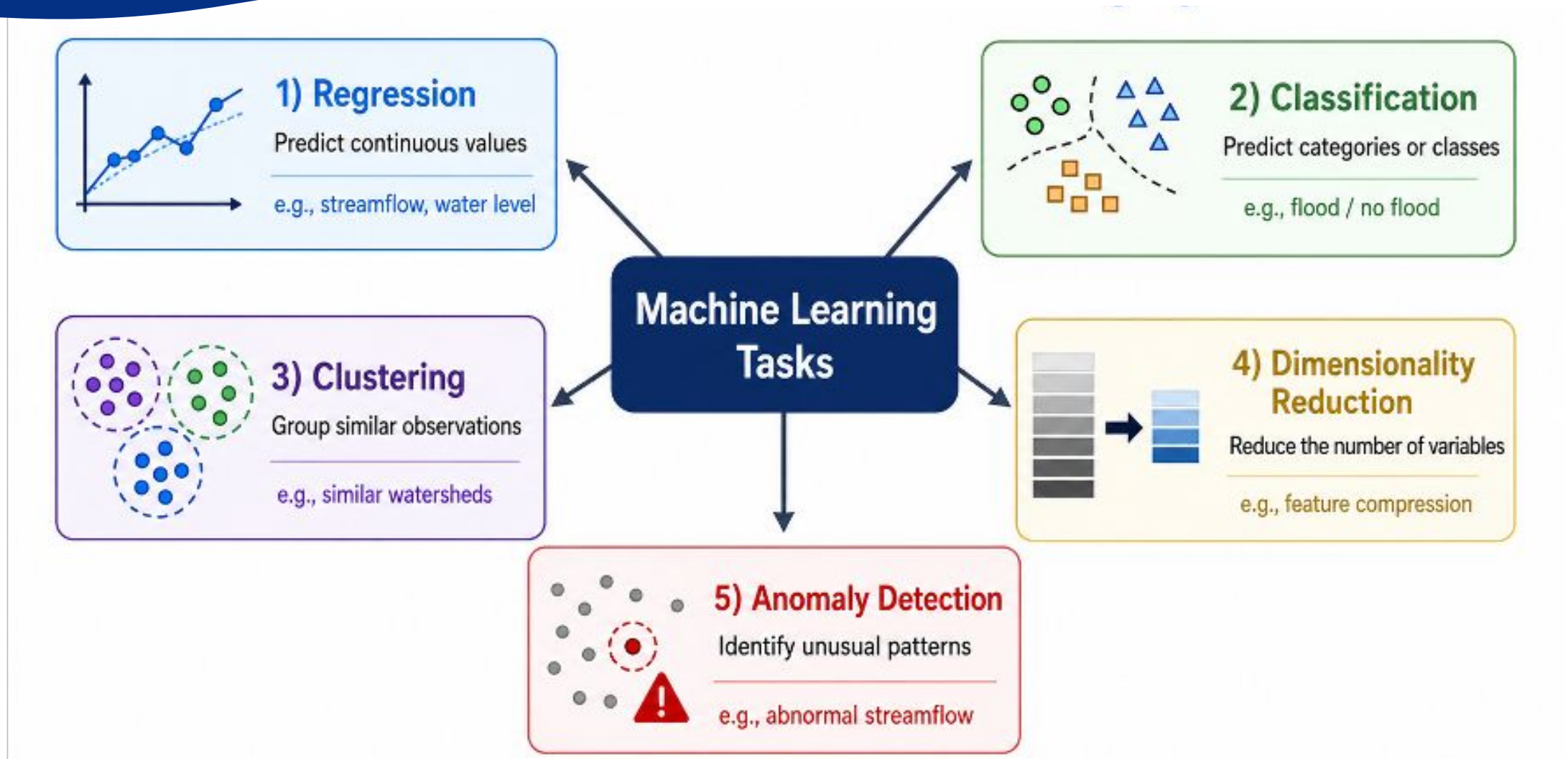



***Learn a function mapping inputs to outputs using training data.***

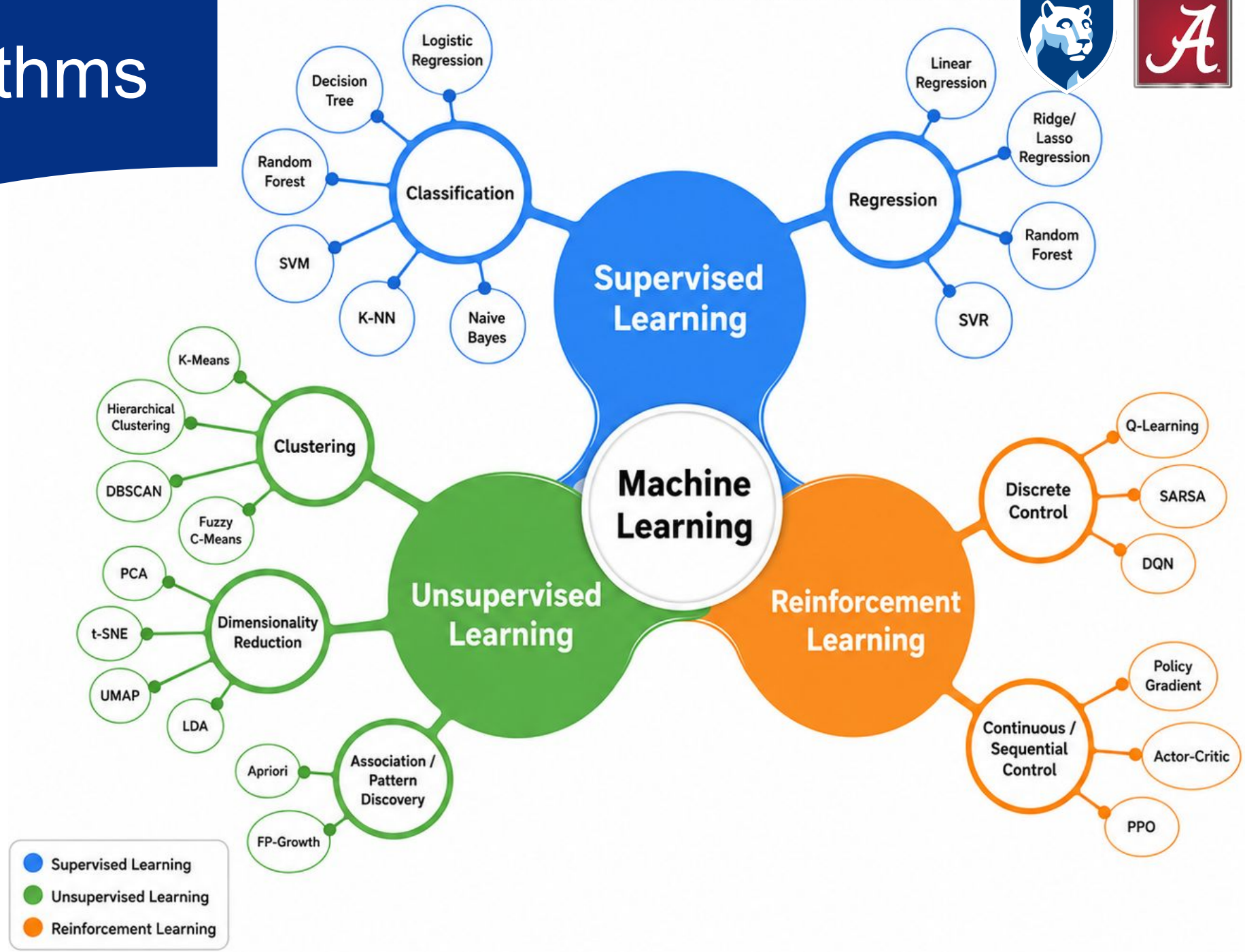
# ML Types


1) Supervised Learning
Learns from labeled data
Labels
Flood
No Flood
e.g., flood / no flood, streamflow prediction
Machine Learning
2) Unsupervised Learning
Finds patterns in unlabeled data
e.g., clustering similar watersheds
3) Reinforcement Learning
Learns through rewards and penalties
Agent
Environment
Reward
e.g., reservoir control, robot navigation

# ML Types

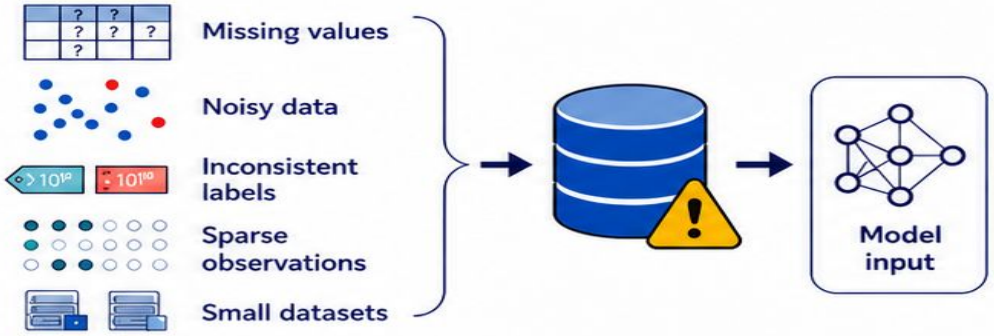

Machine Learning Tasks
1) Regression
Predict continuous values
e.g., streamflow, water level
2) Classification
Predict categories or classes
e.g., flood / no flood
3) Clustering
Group similar observations
e.g., similar watersheds
4) Dimensionality Reduction
Reduce the number of variables
e.g., feature compression
5) Anomaly Detection
Identify unusual patterns
e.g., abnormal streamflow

# ML Algorithms



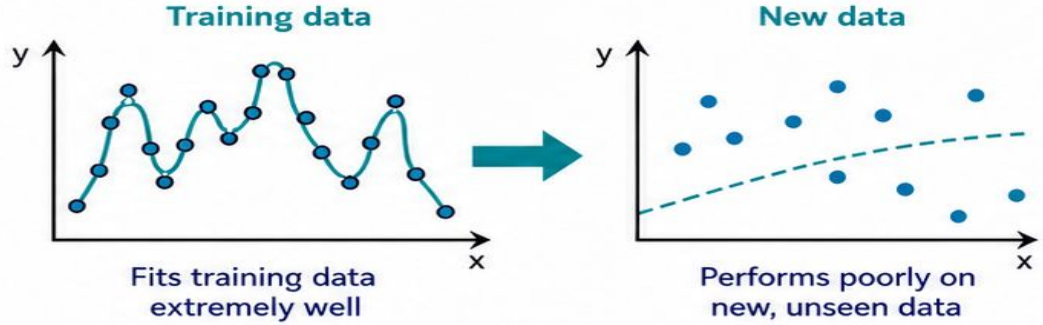
# ML Limitations

## 1 Data quality and quantity



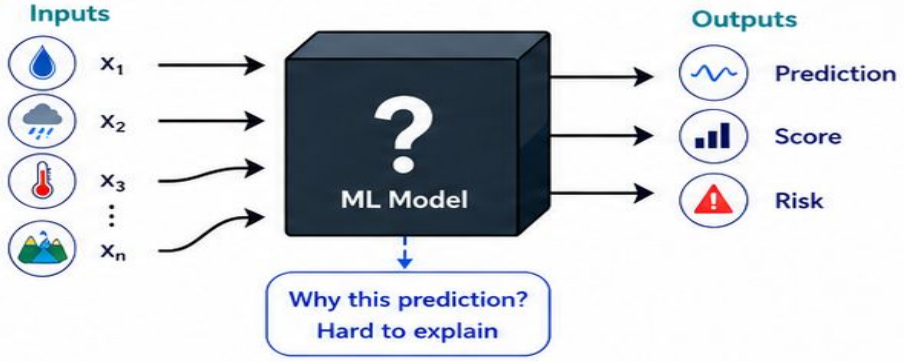

ML depends on enough high-quality, representative data.

## 2 Overfitting and poor generalization



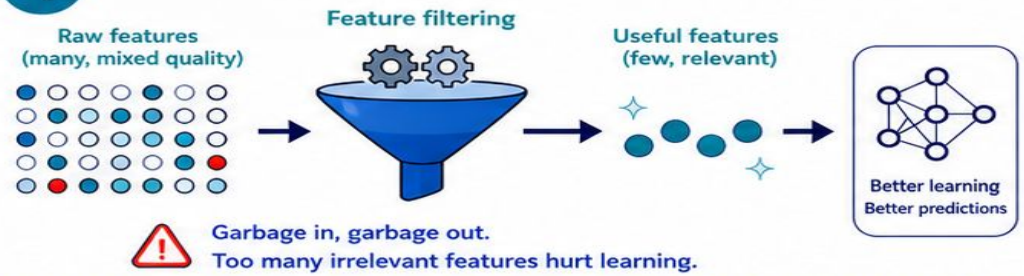

A model may learn noise instead of true patterns.

## 3 Interpretability (black box)



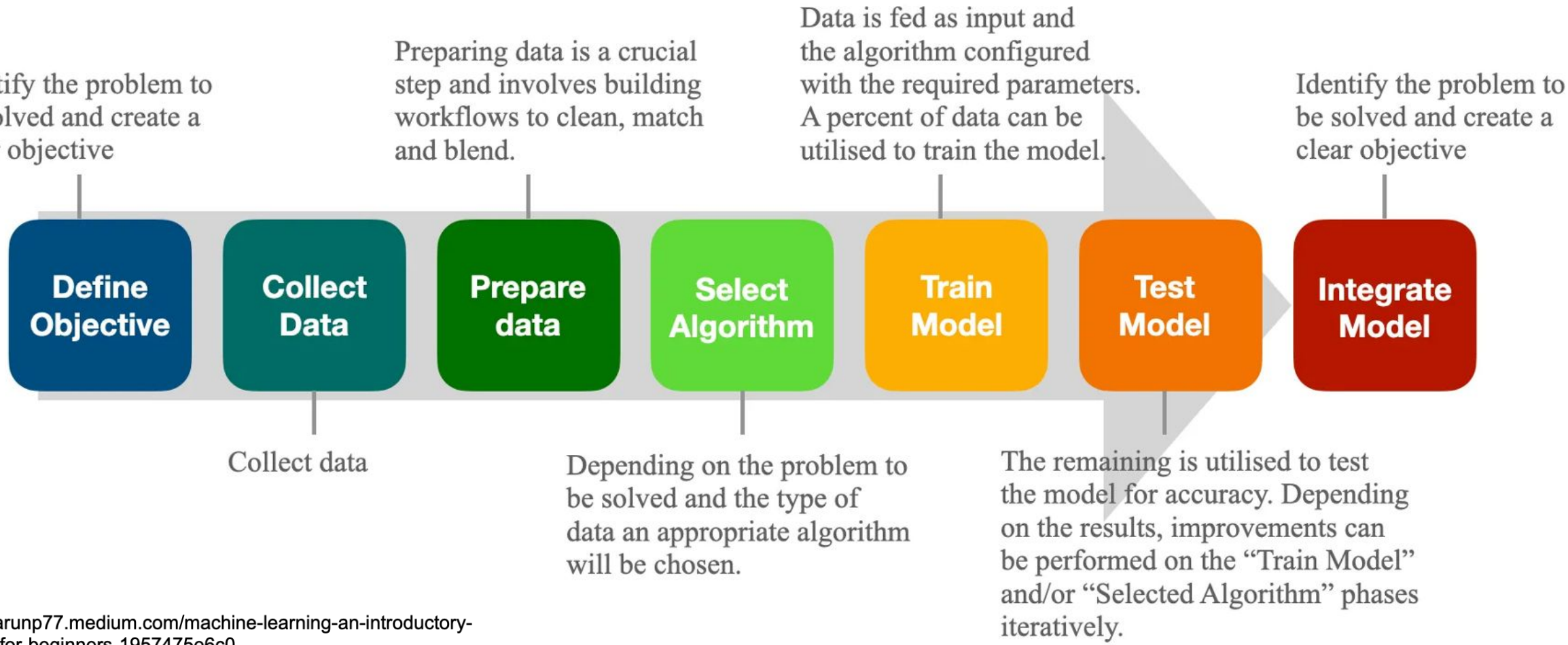

Many ML models are difficult to understand and justify.

## 4 Irrelevant Features




Feature engineering

# ML Workflow (High-level)

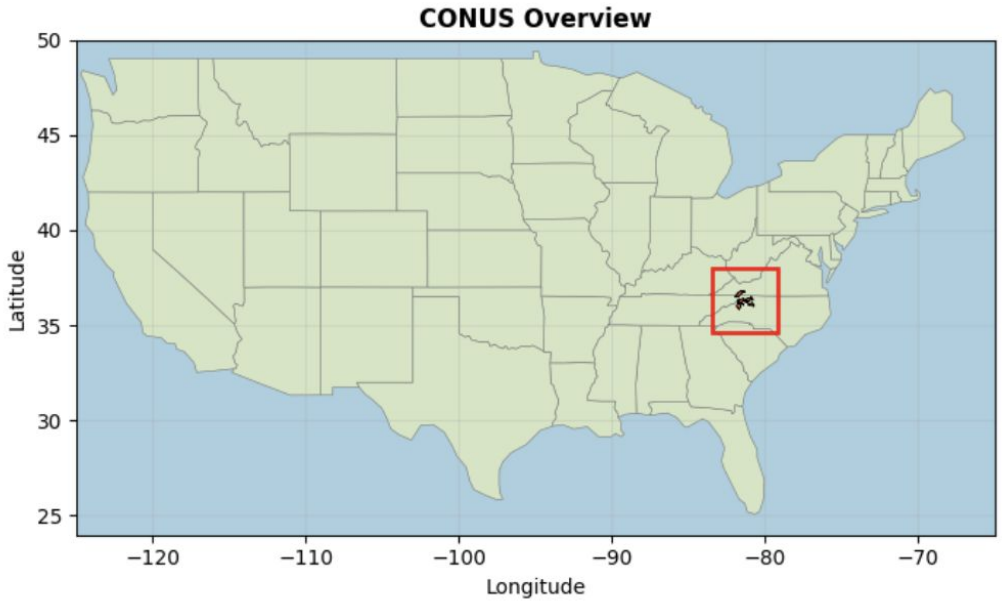



https://arunp77.medium.com/machine-learning-an-introductory-tutorial-for-beginners-1957475e6c0

# Notebook #1 Intro

1. Load and Explore
   - Inspect CAMELS dataset and streamflow target
2. **Preprocess**
   - Split data for training, apply normalizations
3. **Build an LSTM model**
   - Implement a modular PyTorch model architecture
4. **Train**
   - Optimize model weights & biases, monitor loss convergence
5. **Evaluate**
   - Assess streamflow prediction on “unseen” data
6. Tuning & Diagnostics
   - Observe common failure modes and methods for addressing them

# CAMELS Data




| Data component | Example |
|---|---|
| Dynamic forcings | Precipitation, temperature, radiation |
| Static Attributes | Physical and geological basin characteristics |
| Target | Streamflow |
| Time dimension | Daily observations |
| Basin dimension | 10 gauged basins in Southern Appalachia |

# Code

# Code Setup

Foundations-of-ML/tutorials/

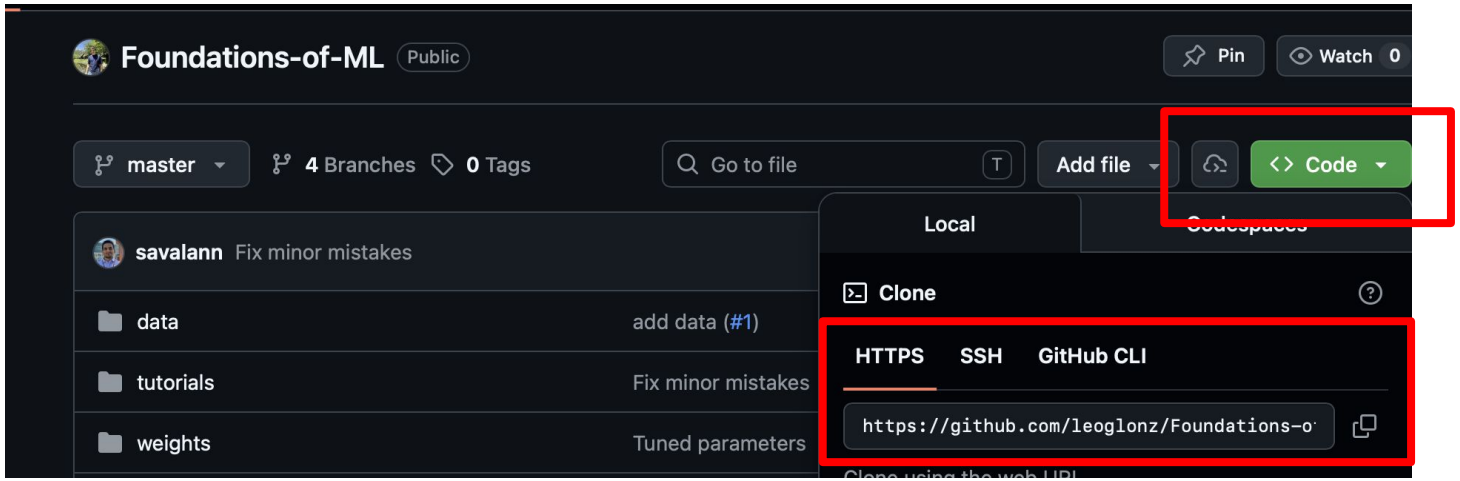
**2i2c**

1. Login to https://workshop.ciroh.awi.2i2c.cloud
2. Launch a **Medium** server with image **Foundations of ML**
3. Open a new terminal and clone:

```
git clone https://github.com/leoglonz/Foundations-of-ML.git
```

4. Install dependencies:

```
pip install -e Foundations-of-ML/
```

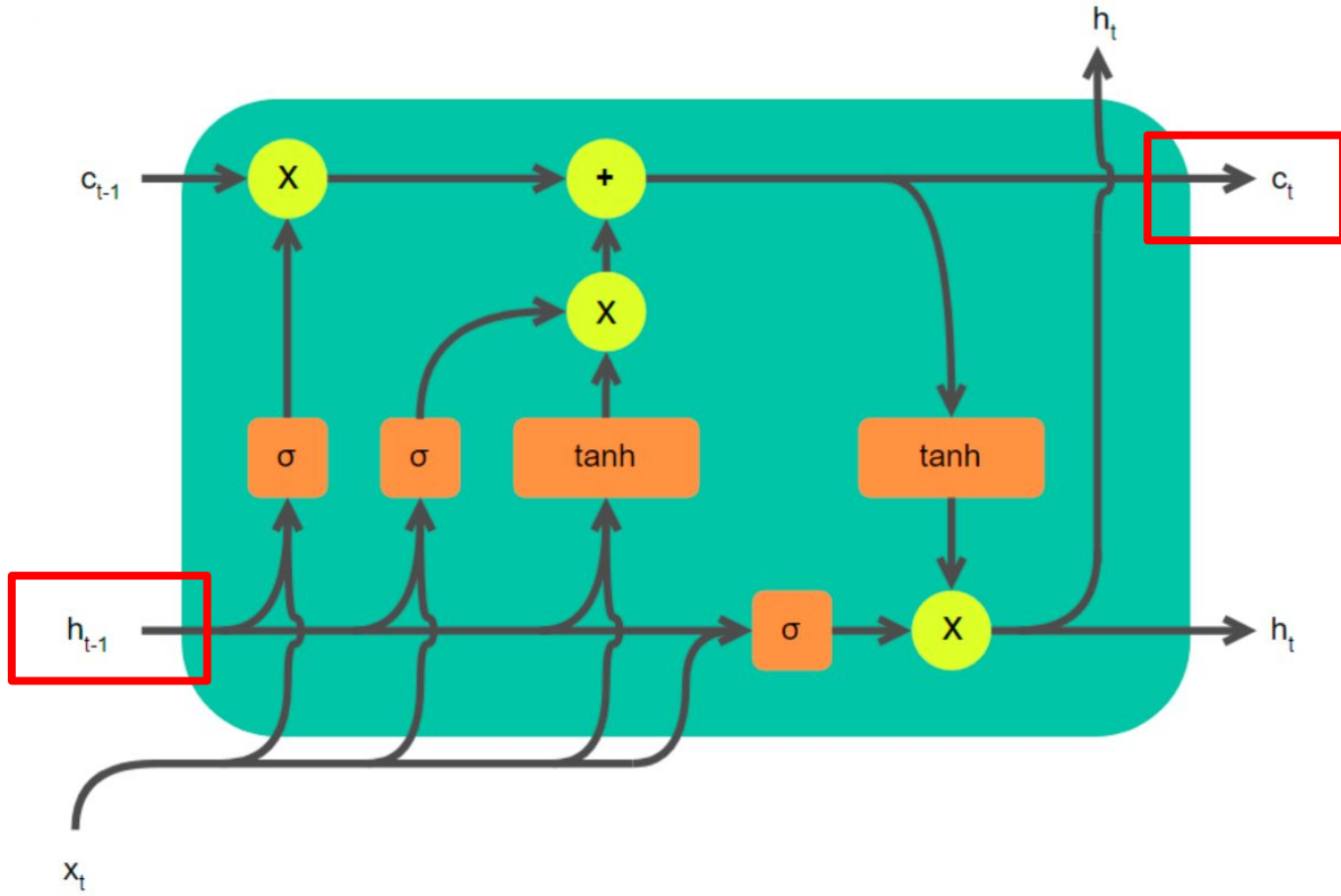


# LSTM

Weights

Biases

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f)$$
$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i)$$
$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o)$$
$$\tilde{c}_t = \sigma_c(W_c x_t + U_c h_{t-1} + b_c)$$
$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t$$
$$h_t = o_t \circ \sigma_h(c_t)$$



# Hyperparameters

Hyperparameters shape model learning, complexity, and generalization.

Examples:

- Hidden size → model capacity (number of tunable parameters)
- Layers → model depth
- Dropout → overfitting control (random parameter initialization)

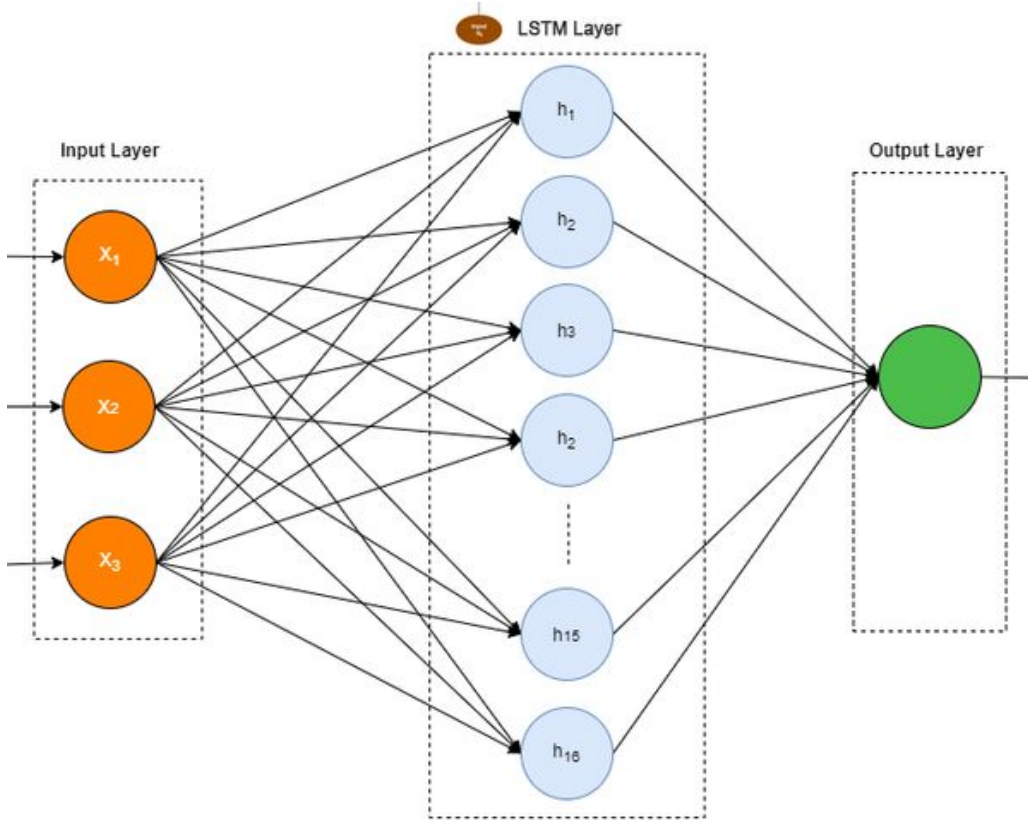
Use validation performance to decide whether the model is underfitting, overfitting, or generalizing well.

# Data Processing

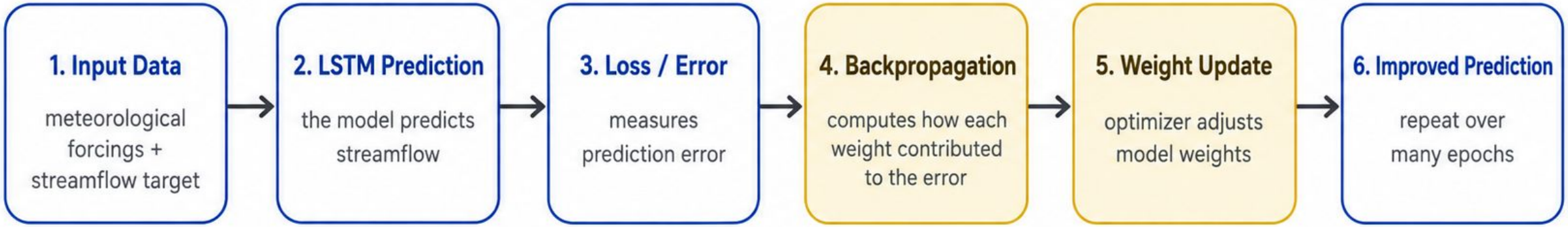
- Raw sensor data must be transformed before it can be fed into a neural network
  - Temporal Split: Train (1980-2000) / Val (2000-2008) / Test (2008-2014)
  - Normalization: z-score standardization computed on training data only (applying to val/test leaks future info)
  - Log-transform: streamflow is skewed – learning in log-space gives better low-flow accuracy
  - Sequence construction: 365-day sliding windows give the LSTM a full year of history to learn from
- **Data Leakage:** fitting scalers on the full dataset is a classic mistake — the model indirectly "sees" the future

# LSTM

- **Input:** 365-day sliding windows of daily forcings
- **Input projection**: linear layer maps so forcing variables → hidden space
- **LSTM core**: processes the sequence one day at a time, propagating a hidden state that captures hydrologic memory (e.g., antecedent soil moisture, snowpack)
- **Dropout**: randomly zeros activations during training to prevent memorization of training patterns
- **Output projection**: collapses the 64-dimensional hidden state → 1 predicted streamflow value per day
- **Output:** sequence of daily streamflow predictions for all 365 days in the window (sequence-to-sequence, not sequence-to-one)



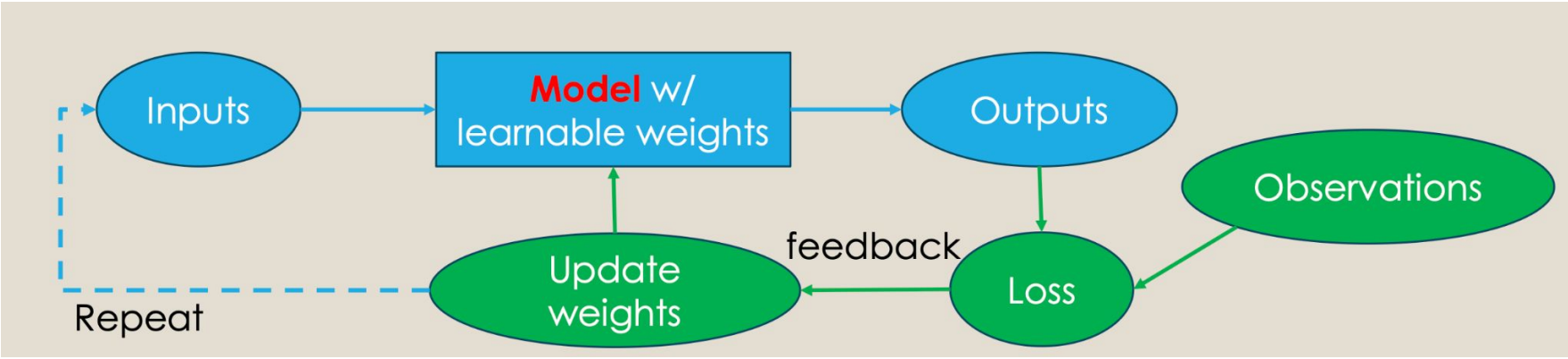
# Backpropagation




Backpropagation adjusts model weights step by step so future predictions produce lower error.

# Training

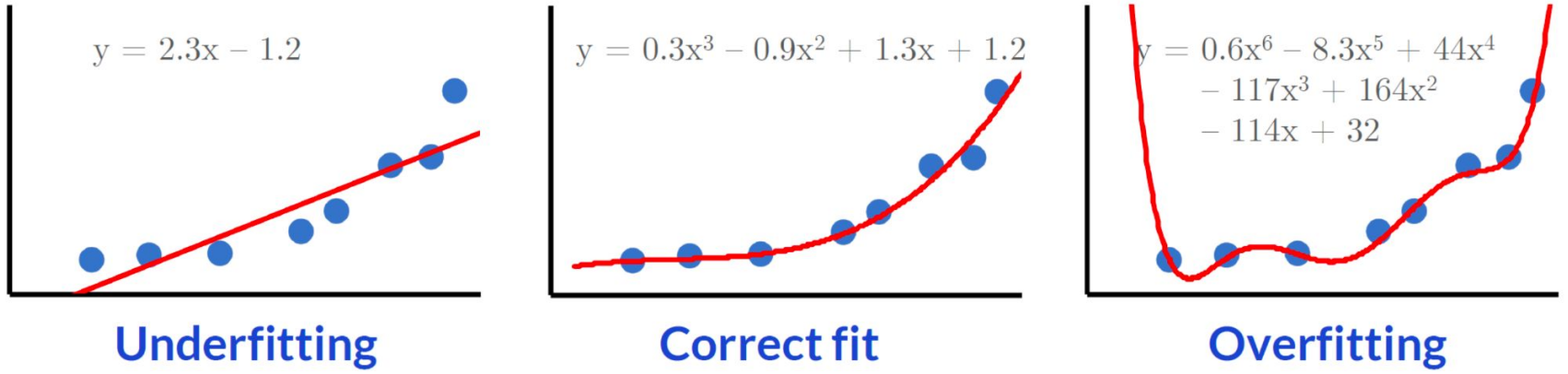
- **Each training step repeats:** Forward Pass → Compute Loss → Backward Pass → Optimizer Update
  - Loss function: MSE in normalized log-space; missing streamflow values (NaN) are masked out before computing gradients
  - Optimizer: Adam – automatically adapts the learning rate for each parameter

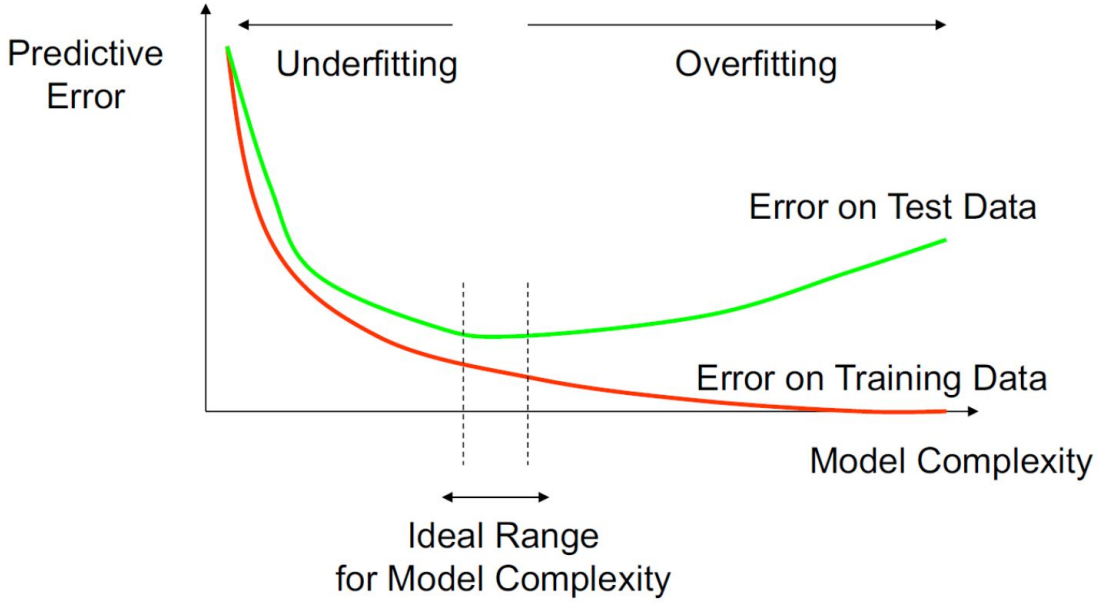


# Evaluation

- Evaluate only on the unseen test period (2008-2014) using normalization statistics from training
- **Three diagnostic views:**
  - NSE score per basin –how well does the model generalize across 10 different watersheds?
  - Hydrograph plots – does the predicted streamflow track the observed peaks and recessions over time?
  - Observed vs. predicted scatter – are predictions systematically biased high or low?
- A single average score hides a lot – always inspect individual basins
- **Common failure patterns:** misses extreme peak flows, seasonal bias, poor performance on atypical basins

# Over/Underfitting




Underfitting

- Model has insufficient tunable parameters to capture diversity in training examples

Overfitting

- Model fits training data very well; learns noise
- Fails to generalize to new examples

# Feature Engineering

**Are we giving the model the right information in the right form?**

- Feature engineering changes what information the model receives
- Hydrological example: seasonal signals such as day-of-year sine/cosine
- The goal is not to change the model first, but to help the model see better information

# Notebook 2: Model Diagnostics

- An average NSE > 0.7 is promising – but the mean hides where and why the model struggles
- **Signs architecture changes may help:**
  - Systematic seasonal bias in residuals (e.g., always wrong in spring)
  - Consistent failure on peak flows across all basins
  - NSE strongly correlated with a basin attribute the model never receives as input
- **Signs architecture is NOT the bottleneck:**
  - Train loss much lower than val loss → fix regularization (dropout, early stopping) first
  - Some basins have very few observations → data quality problem, not a model problem

# Augmentation: Static Attributes

- Baseline LSTM only sees dynamic climate forcings – no knowledge of basin physical characteristics
- Two basins receiving identical rainfall can behave very differently based on soil permeability, slope, and vegetation
- **Architecture change – add a static attribute branch:**
  - Static attributes (35 values per basin) ⟶ small MLP ⟶ embedding vector
  - Embedding is repeated across all 365 time steps and concatenated with the dynamic inputs
  - The combined sequence is then passed into the LSTM as before
- This lets the model condition its predictions on the physical character of each basin

# Architecture Modifications

- **Two additional structural modifications to explore beyond the baseline:**
  - Deeper LSTM: stack 2 LSTM layers – layer 1 learns short-term runoff dynamics; layer 2 learns longer seasonal patterns
  - Depth only helps when the task has genuine hierarchical temporal structure and enough training data to fill extra parameters
  - Causal Convolution encoder: replace the linear encoder with 1D convolutions that aggregate local temporal context (e.g., a 7-day rain event) before the LSTM
  - "Causal" means each output timestep only looks at past inputs – no future leakage
- If val loss diverges from train after adding layers → increase dropout before adding more depth

# Model Comparison

- **Four model variants are compared side-by-side on the same 10-basin test set:**
  - Baseline: hidden=64, 1 LSTM layer – the reference point everything is measured against
  - +Attributes: adds static basin embeddings – expected to improve inter-basin generalization
  - Deep LSTM: 2 stacked LSTM layers – tests whether hierarchical temporal features help
  - ConvLSTM: causal convolution encoder + LSTM – tests explicit local pattern extraction
- Compare NSE distributions (not just means) across all 10 basins – did the change help the weak basins, the strong ones, or both?

# Foundations of ML: From Architecture to Optimization

Savalan Naser Neisary[1], Leo Lonzarich[2]
The University of Alabama[1]
The Pennsylvania State University[2]