# EE345: Intro to Foundations of ML

M/W IEB 205 @ 10-11:20

LJ Ratliff, 2025-09-24

# Welcome

- Prof: Lillian Ratliff (ratliffl@uw)
  - Associate Professor ECE, Adjunct in CSE, AA;
  - Amazon Scholar
- Research: Machine learning, game theory, optimization

- TA: Adhyyan Narang (adhyyan@uw), ECE PhD Student
- Research: Machine learning, game theory, optimization

- grader: Sergey Konyakin (serkon@uw), MS ECE Student

# What is Machine Learning (ML)?

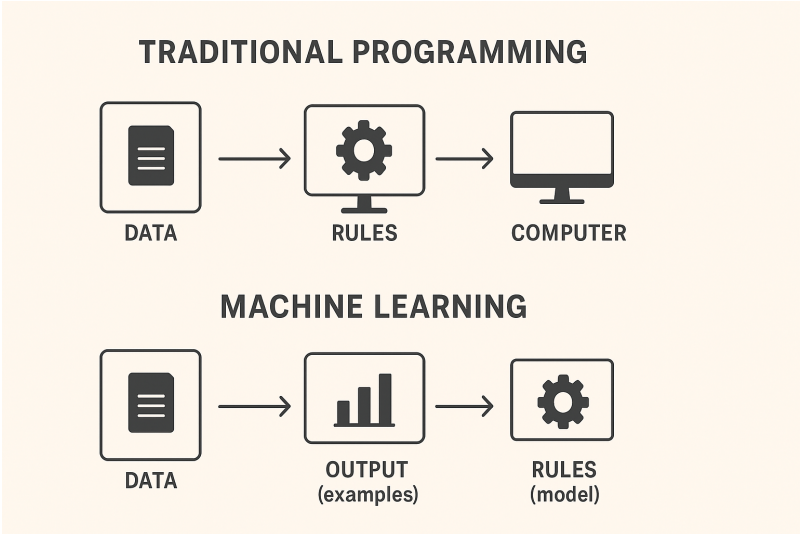
The process of giving machines the ability to *learn from data* so they can *recognize patterns* and make *predictions* or *decisions* without being explicitly programmed for every specific situation.

- The fundamental assumption of machine learning is that pattern recognition is possible. We take this as an axiom.
- The technical question: *What heuristics reasonably work to program computers to recognize those assumed patterns?*

Ben Recht @ UC Berkeley

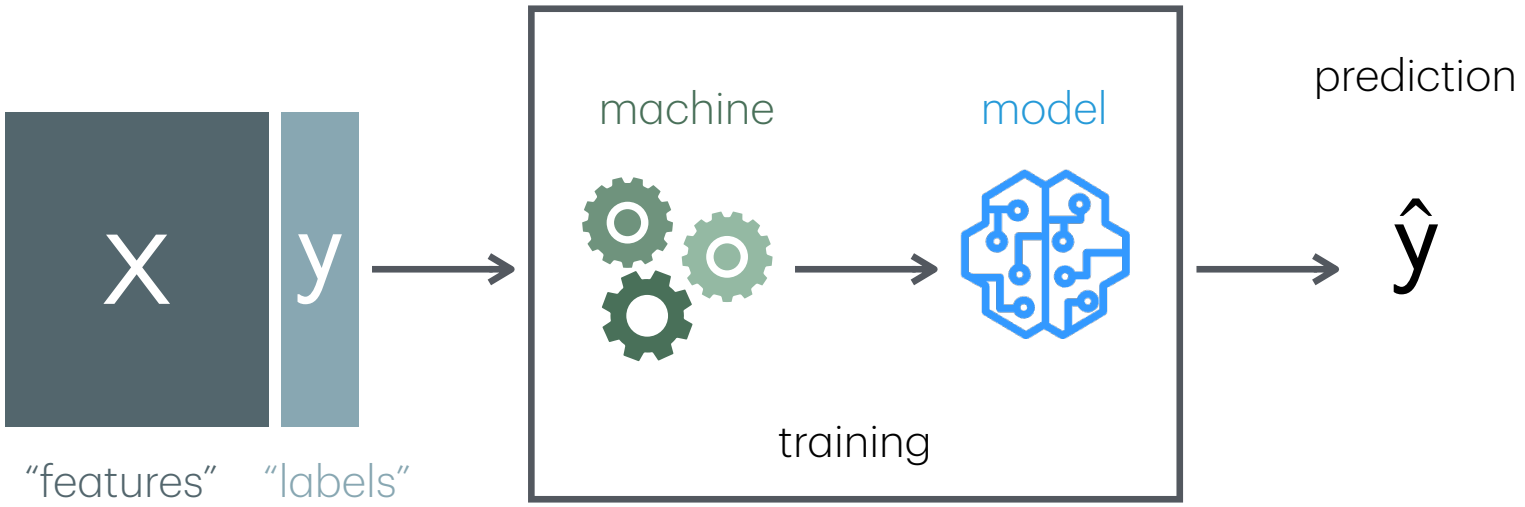
argmin

Machine Learning 101

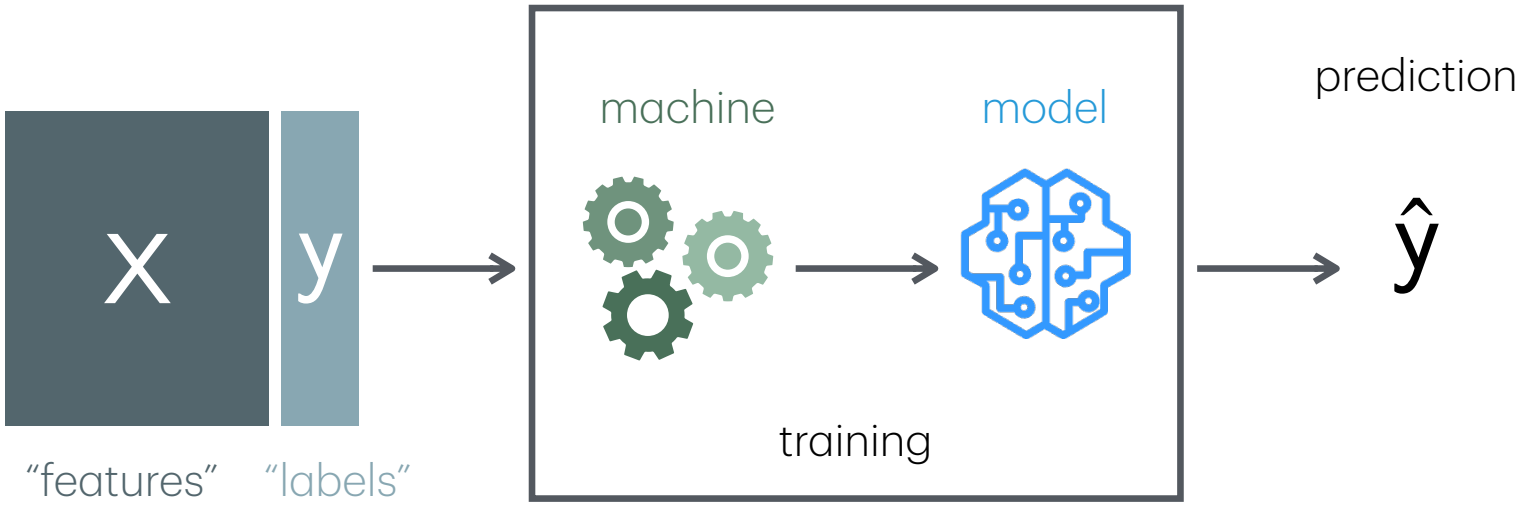


# What is Machine Learning (ML)?

The process of giving machines the ability to *learn from data* so they can *recognize patterns* and make *predictions* or *decisions* without being explicitly programmed for every specific situation.



# ML examples




- house price ⟵ [size, bedrooms, walkability score, ...]
- Blood pressure ⟵ [age, weight, exercise, genetic factors, ...]
- exam score ⟵ [hours studied, homework completed, sleep, ...]

$$\text{house price} = \theta_1 \cdot \text{ size} + \theta_2 \cdot \text{ bedrooms} + \theta_3 \cdot \text{ walkability} + w$$

# Course Focus

$y = \theta_1 \cdot x_1 + \theta_2 \cdot x_2 + \cdots + \theta_n x_n + w$

$y = \theta_1 \phi(x_1) + \cdots + w$

$\log(x_1)$

- Core/foundational topics:
  - supervised learning (regression & classification),
  - un-supervised (clustering, PCA)
- Advanced topics:
  - kernel methods and non-linear feature construction
  - generalization, regularization, cross-validation, bias-variance tradeoff
  - loss function design/selection
  - stochastic gradient descent,
  - semi-supervised learning (PPI)
  - feature analysis/design
- Advanced ML methods:
  - Other ML Models: e.g., decision trees, neural networks, transformers, diffusion, reinforcement learning, multi-armed bandits
  - Data/distribution shift (label vs concept vs covariate)

# ML Foundations: Why?

This class will focus on foundations of ML such as linear and other simple, transparent models

# ML Foundations: Why?

This class will focus on foundations of ML such as linear and other simple, transparent models

- Simple models → great for baselines, fast inference, small data, and transparency.
  - **building blocks:** Many complex architectures (deep nets, transformers) *reduce to stacks of linear predictors + nonlinearities*.
  - **give intuition:** understanding how coefficients scale features or how decision boundaries form makes it easier to grasp what deeper models are doing in high dimensions
  - **practical baselines:** in research and industry, you *always* start with a simple model to sanity-check.
  - **interpretable:** you can literally plot the data, draw the line or plane, and show predictions, which is much harder with a 70B-parameter LLM.

# ML Foundations: Why?

This class will focus on foundations of ML such as linear and other simple, transparent models

- Simple models → great for baselines, fast inference, small data, and transparency.
  - **building blocks:** Many complex architectures (deep nets, transformers) *reduce to stacks of linear predictors + nonlinearities*.
  - **give intuition:** understanding how coefficients scale features or how decision boundaries form makes it easier to grasp what deeper models are doing in high dimensions
  - **practical baselines:** in research and industry, you *always* start with a simple model to sanity-check.
  - **interpretable:** you can literally plot the data, draw the line or plane, and show predictions, which is much harder with a 70B-parameter LLM.
- Complex models → great for scaling, raw data modalities, transfer learning, and long-term generalization.
  - **End-to-end:** less manual feature engineering
  - **Expressive:** can capture nonlinear, structured, and hierarchical patterns
  - **Reusable Architectures:** same backbone can be adapted across tasks
  - **Scales with Data/Compute:** often improves steadily as size grows

# ML Foundations: Why?

This class will focus on foundations of ML such as linear and other simple, transparent models

# ML Foundations: Why?

This class will focus on foundations of ML such as linear and other simple, transparent models

- Token Scoring (Softmax Head): **linear predictor** inside the LLM
  - At end of a transformer, there's a **linear layer** (weights + bias) that maps the hidden representation of a token into a large vector: logits $= W \cdot h_t + b, \quad \Pr(\text{ token } \mid h_t) = \text{softmax}(\text{ logits})$

# ML Foundations: Why?

This class will focus on foundations of ML such as linear and other simple, transparent models

- Token Scoring (Softmax Head): **linear predictor** inside the LLM
  - At end of a transformer, there's a **linear layer** (weights + bias) that maps the hidden representation of a token into a large vector: logits $= W \cdot h_t + b, \quad \Pr(\text{ token } \mid h_t) = \text{softmax}(\text{ logits})$
- Classifier Head: LLM as a feature extractor $\longrightarrow$ **linear model** does the prediction.
  - In fine-tuning of GPT-like models (e.g., sentiment classification, toxicity detection, retrieval), often the LLM is frozen and a **simple linear predictor** "head" is trained on embeddings: e.g., Predictor = logistic regression that outputs "positive" or "negative" sentiment.

# ML Foundations: Why?

This class will focus on foundations of ML such as linear and other simple, transparent models

- Token Scoring (Softmax Head): **linear predictor** inside the LLM
  - At end of a transformer, there's a **linear layer** (weights + bias) that maps the hidden representation of a token into a large vector: logits $= W \cdot h_t + b, \quad \Pr(\text{ token } \mid h_t) = \text{softmax}(\text{ logits})$
- Classifier Head: LLM as a feature extractor $\longrightarrow$ **linear model** does the prediction.
  - In fine-tuning of GPT-like models (e.g., sentiment classification, toxicity detection, retrieval), often the LLM is frozen and a **simple linear predictor** "head" is trained on embeddings: e.g., Predictor = logistic regression that outputs "positive" or "negative" sentiment.
- Reward Models in RLHF: reward assignment is just a **simple predictor**
  - preferences mapped to scalar reward via small head: **linear regression** (hidden state → scalar score).

# ML Foundations: Why?

This class will focus on foundations of ML such as linear and other simple, transparent models

- Token Scoring (Softmax Head): **linear predictor** inside the LLM
  - At end of a transformer, there's a **linear layer** (weights + bias) that maps the hidden representation of a token into a large vector: logits $= W \cdot h_t + b$, $\Pr(\text{token} \mid h_t) = \text{softmax}(\text{logits})$
- Classifier Head: LLM as a feature extractor $\longrightarrow$ **linear model** does the prediction.
  - In fine-tuning of GPT-like models (e.g., sentiment classification, toxicity detection, retrieval), often the LLM is frozen and a **simple linear predictor** "head" is trained on embeddings: e.g., Predictor = logistic regression that outputs "positive" or "negative" sentiment.
- Reward Models in RLHF: reward assignment is just a **simple predictor**
  - preferences mapped to scalar reward via small head: **linear regression** (hidden state → scalar score).
- Routing in LLMs: gating via **linear or softmax classifiers**
  - LLMs (like Switch Transformers, Mixture-of-Experts) use a **gate network** to decide which expert block to send a token through

# ML Foundations: Why?

This class will focus on foundations of ML such as linear and other simple, transparent models

- Token Scoring (Softmax Head): **linear predictor** inside the LLM
  - At end of a transformer, there's a **linear layer** (weights + bias) that maps the hidden representation of a token into a large vector: $\text{logits} = W \cdot h_t + b, \quad \Pr(\text{token} \mid h_t) = \text{softmax}(\text{logits})$
- Classifier Head: LLM as a feature extractor ⟶ **linear model** does the prediction.
  - In fine-tuning of GPT-like models (e.g., sentiment classification, toxicity detection, retrieval), often the LLM is frozen and a **simple linear predictor** "head" is trained on embeddings: e.g., Predictor = logistic regression that outputs "positive" or "negative" sentiment.
- Reward Models in RLHF: reward assignment is just a **simple predictor**
  - preferences mapped to scalar reward via small head: **linear regression** (hidden state → scalar score).
- Routing in LLMs: gating via **linear or softmax classifiers**
  - LLMs (like Switch Transformers, Mixture-of-Experts) use a **gate network** to decide which expert block to send a token through
- Internal probing via linear classifiers enables decoding (e.g., word knowledge, hidden features)
  - internally, the model is arranging information so that simple predictors can pull it out

# How to evaluate ML?: Hold-out method


"features" "labels"

train (e.g., 80%)

test (e.g., 20%)

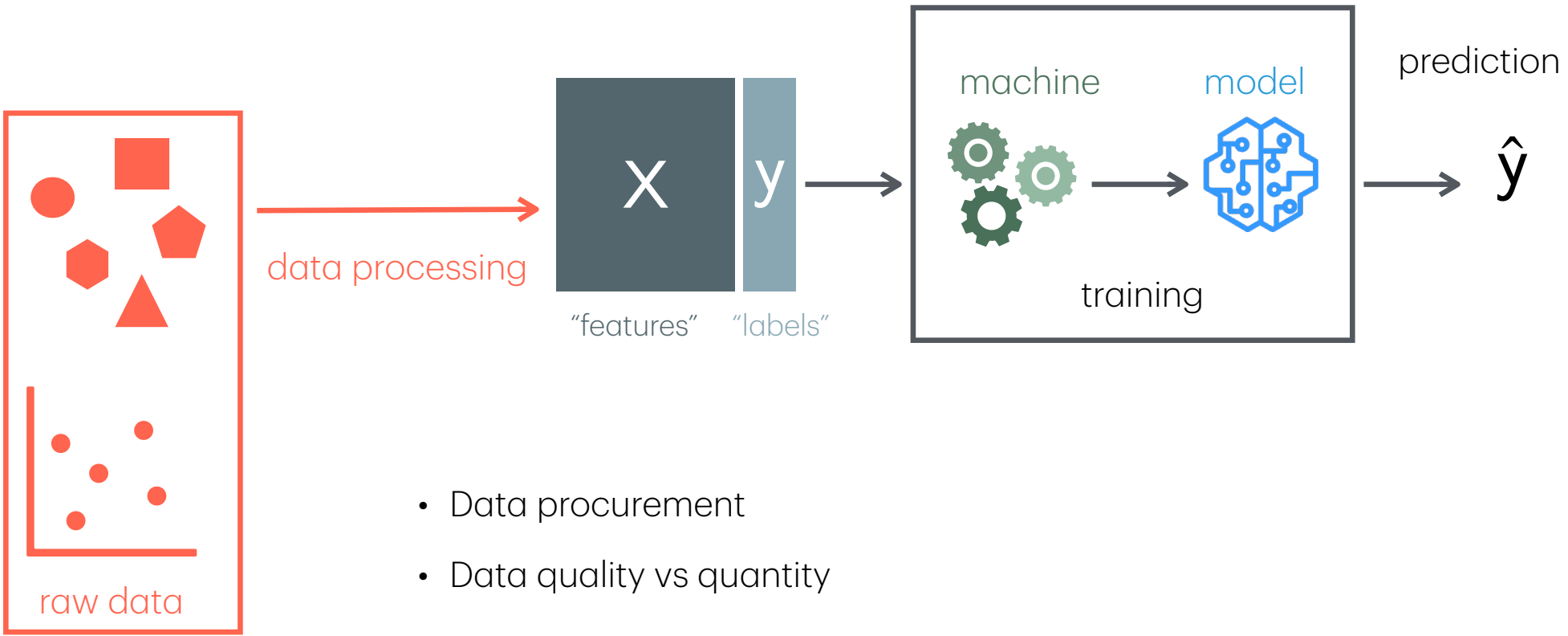
# How to evaluate ML?: Hold-out method


"features" "labels"


train (e.g., 80%)


test (e.g., 20%)

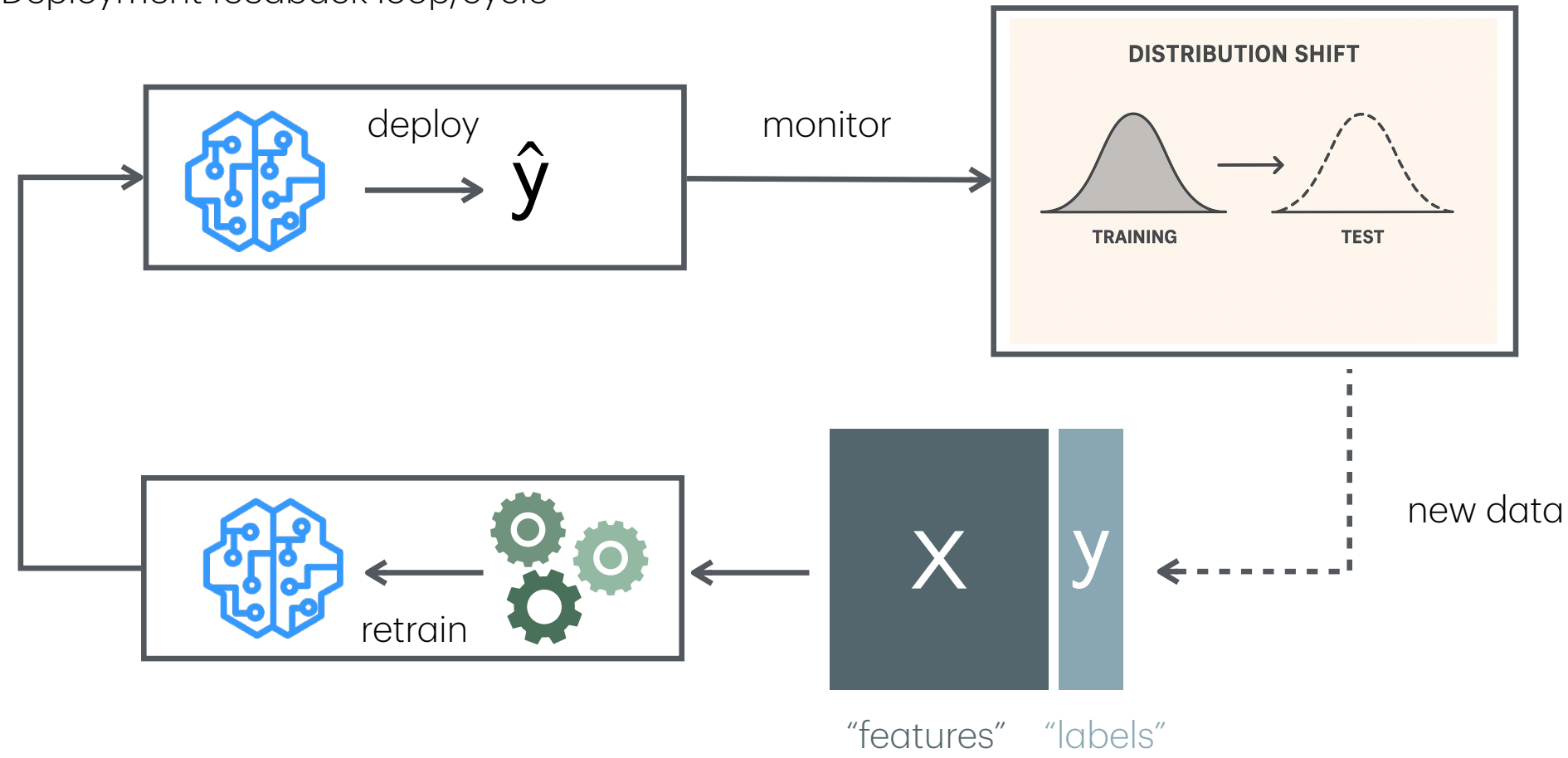
# What is often overlooked in ML courses?



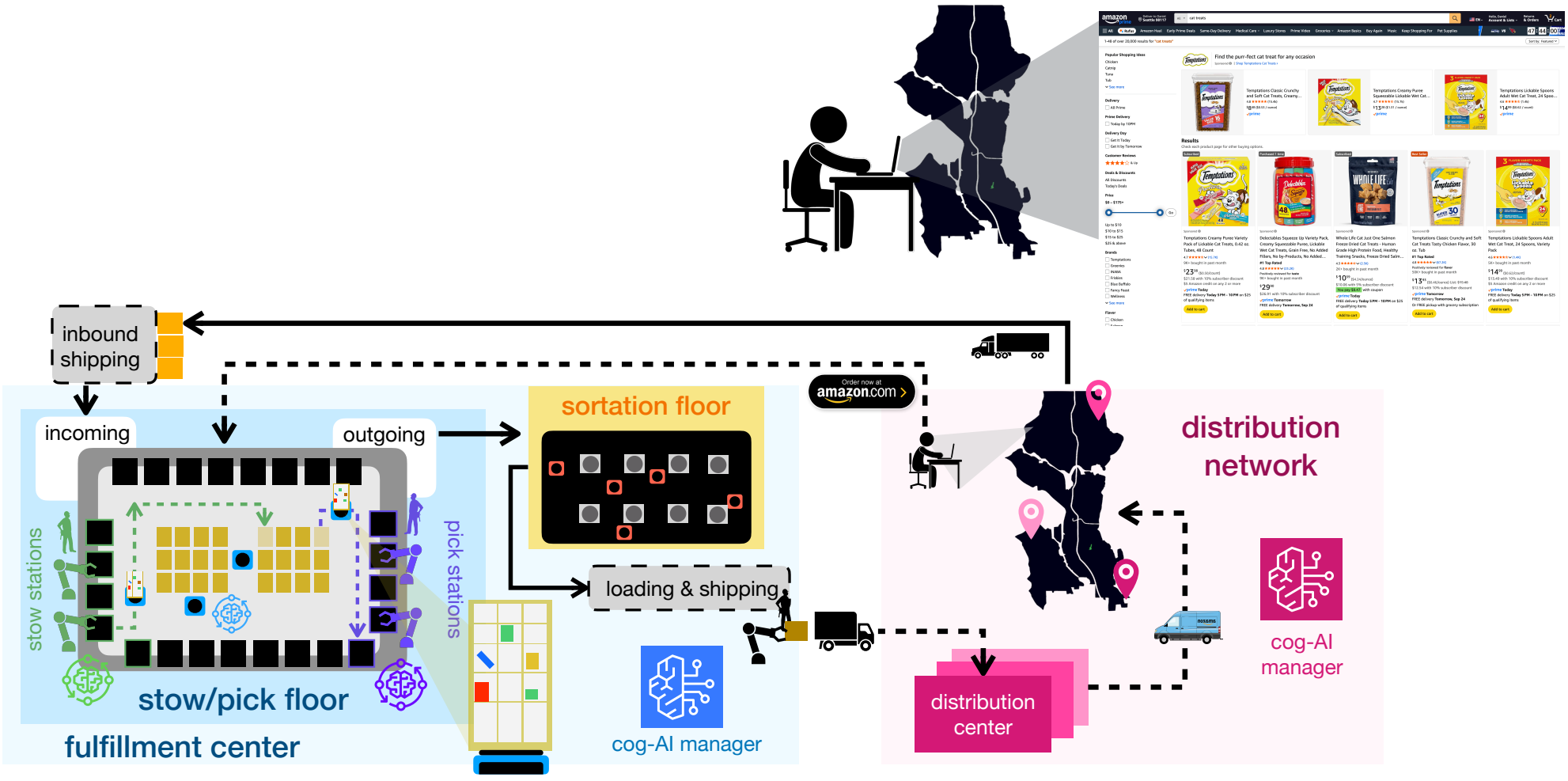

- Data procurement
- Data quality vs quantity

# What is often overlooked in ML courses?

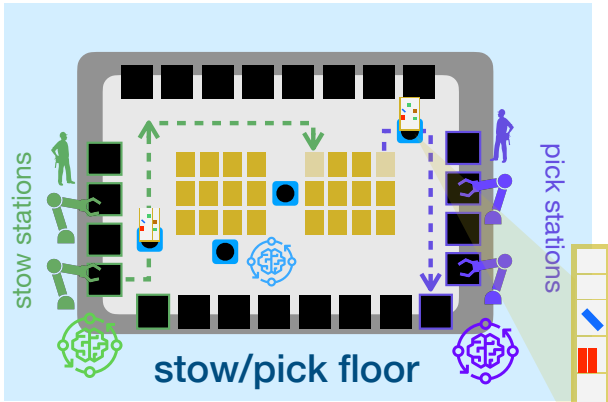
- Deployment feedback loop/cycle

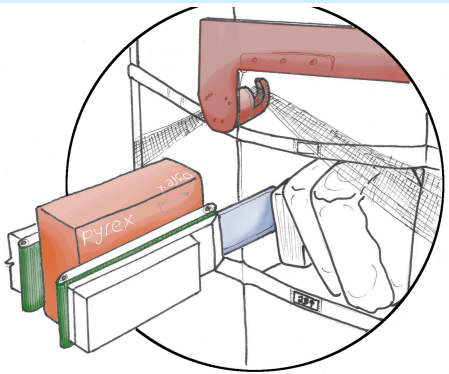


# Motivation: Fulfillment in E-Commerce

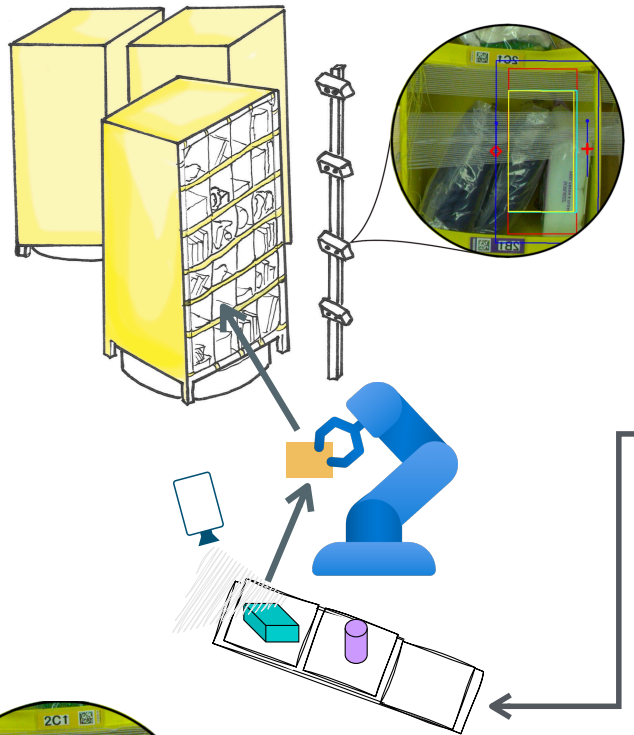


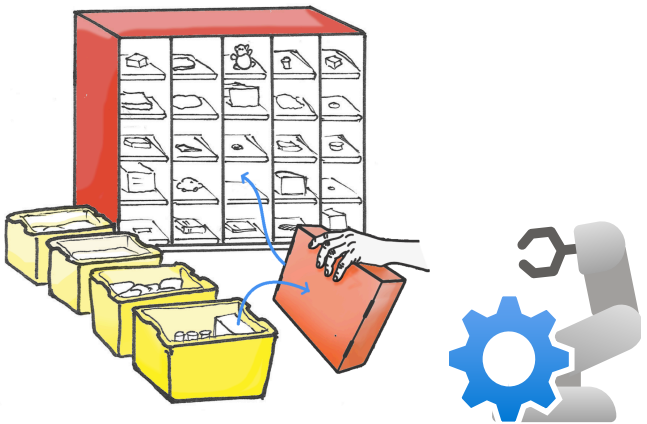
# Motivation: Robotic Stowing (Emboddied AI)



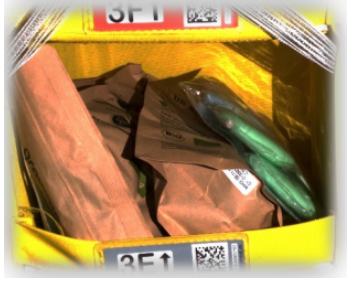






# Motivation: Robotic Stowing (Emboddied AI)





# Motivation: Robotic Stowing (Emboddied AI)

# Motivation: Robotic Stowing (Emboddied AI)

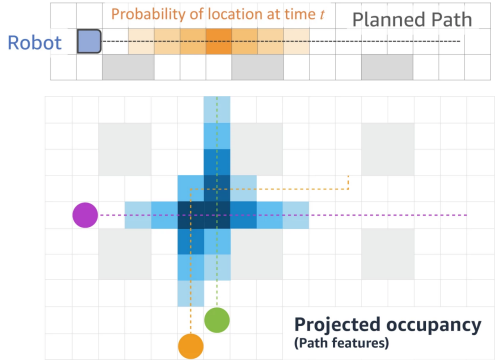
# Motivation: Robotic Stowing (Emboddied AI)

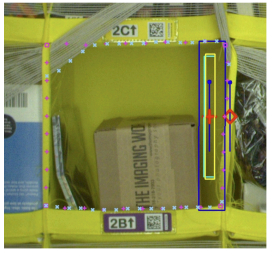
# Motivation: Robotic Stowing (Emboddied AI)

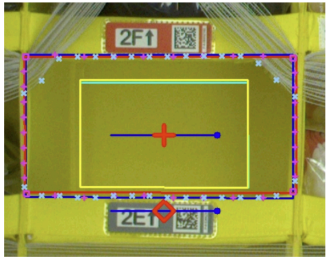
# Motivation: What are the ML problems?

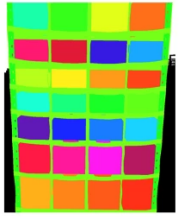
# Motivation: What are the ML problems?

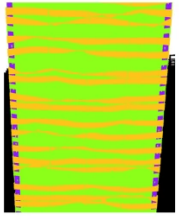
- Forecasting congestion (regression)
- Space/sweep length prediction (regression)
- Predicting success/outcomes (classification)
- Amnesty/defects (classification)
- Synthetic data (generative models)
- Matching policies (RL/Q learning)
- Path planning (reinforcement learning)

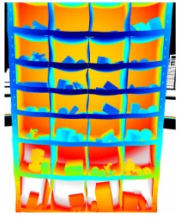


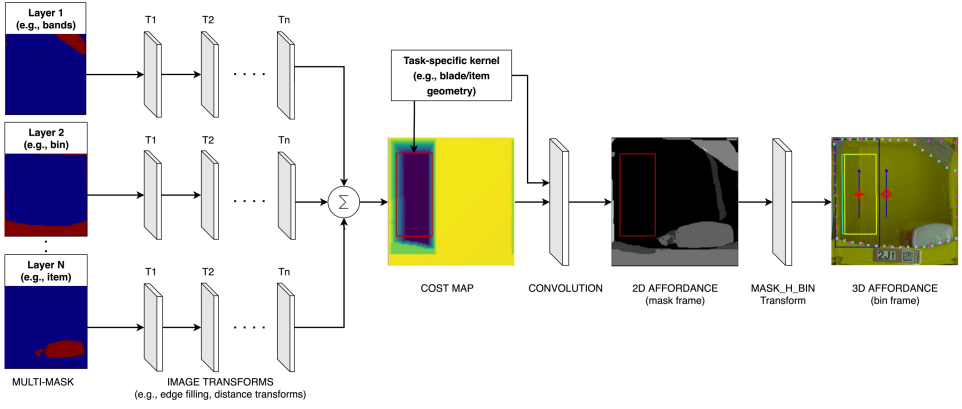









# ML Ops (Machine learning operations)

- Data quality vs quantity
- Data imbalance
- Performance monitoring & retraining
- Distribution Shift
- Latency

# Course Focus

- Core/foundational topics:
  - supervised learning (regression & classification),
  - un-supervised (clustering, PCA)
- Advanced topics:
  - kernel methods and non-linear feature construction
  - generalization, regularization, cross-validation
  - stochastic gradient descent,
  - semi-supervised learning (PPI)
  - feature analysis/design
- Advanced ML methods:
  - Other ML Models: e.g., decision trees, neural networks, transformers, diffusion
  - Data/distribution shift (label vs concept vs covariate)

# Course Information

- Website: ratlifflj.github.io/F25-EE345/
- Canvas: canvas.uw.edu/courses/1860120
  - quizzes used for self-grading hw
  - submitting hw and Python exercises
- Chat (Discord?/Canvas?)
- Gradescope: used for exam grading
- Zoom: washington.zoom.us/j/95494709719
  - If you need to miss class for some reason, there will also be a zoom room to join remotely

# Course Structure/Logistics

- Lectures (M/W IEB 205 @ 10-11:20)
- Discussions (F IEB 205 @ 9:30-11:20)
- Communication/Questions
  - Online Chat (Discord?/Canvas?)
  - Office Hours: TBD
- Guest Speakers from Industry!
- Grade distribution: TBD
- Books?
  - Boyd & Vandenberghe: Intro to Applied LA
  - Calafiore & El Ghaoui: Optimization Models
  - James, Witten, Hastie, Tibshirani: An intro to stat learning
- Homework:
  - weekly (~6-7 total);
  - released W; due following W
  - self-graded & due following M (grader grades 1 problem @ random)
- Python Exercises:
  - weekly assignments (~6-7 total), discussion section will be used to do majority of the work
  - group project: end of term
- Exams:
  - Midterm (~5-6th week)
  - Final (finals week)

# Comments on AI Tool Usage

- **Goal:** Teach everyone best practice on how to use AI tools for coding efficiently.
- **Question everything!** They are wrong often, hallucinate frequently, and sycophantic to a fault!
  - If you get an answer you dont understand or trust, don't be afraid to ask us

Cursor AI

# Up Next...

- Next week we will start with
  - data fitting and simple regression problems
  - data structure
  - basic linear algebra constructs (matrices, vectors, norms) for ML modelling