COMS4995W32
Applied Machine Learning

Dr. Spencer W. Luo

Columbia University | Fall 2025

About this Course

COMS4995W32 - Applied Machine Learning

- Schedule: Thursdays 7:00pm - 9:30pm, Fall 2025
- Location: Pupin Hall 301 (Morningside Campus)
- Credits: 3.0

Instructor & TAs

- Dr. Spencer Luo (swl2145@)

- Case Schemmer (chs2164@)
- Mihika Riya Sanghvi (mrs2356@)
- Hamsitha Challagundla (hc3540@)
- Shashwat Kumar (sk5520@)
- Jiayi Niu (jn2941@)
- Tony Tian (jt3640@)

Course Setup

- https://courseworks2.columbia.edu/courses/234147
  - Will be migrated to the course website soon.

- Lectures: Weekly applied ML topics

- 3 assignments (Code + Report) - 35%
- 1 midterm exam - 30%
- 1 final project - 30%
  - Team-based LLM project

- Office Hours: (To be posted on Courseworks)
- Recitations: TA-led coding & math refreshers

Overview

- Foundations of Applied ML
  - ML workflow, in production, and case studies
  - Data preparation, cleaning, and feature engineering
- Classical ML Methods
  - Generative vs. discriminative models
  - Evaluation metrics, bias–variance tradeoff
  - Tree-based models and ensemble methods
- Unsupervised Learning
  - Clustering, self-supervised learning

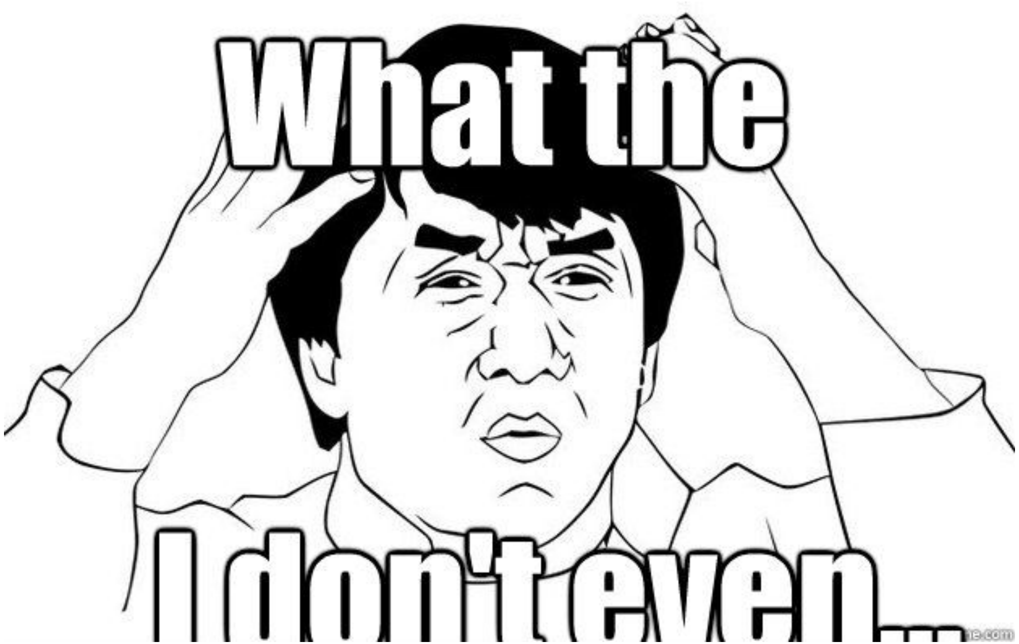
Overview

- Deep Learning
  - Neural networks fundamentals (MLP, backprop, activation)
  - Convolutional networks (ResNet, image tasks)
  - Transformers (Attention, BERT, ChatGPT, Gemini)

- Large Language Model
  - Pre-training & supervised fine-tuning
  - Reinforcement learning in LLMs
  - Agentic workflows (Thinking model, LangChain, tool integration)

Introduction to AML

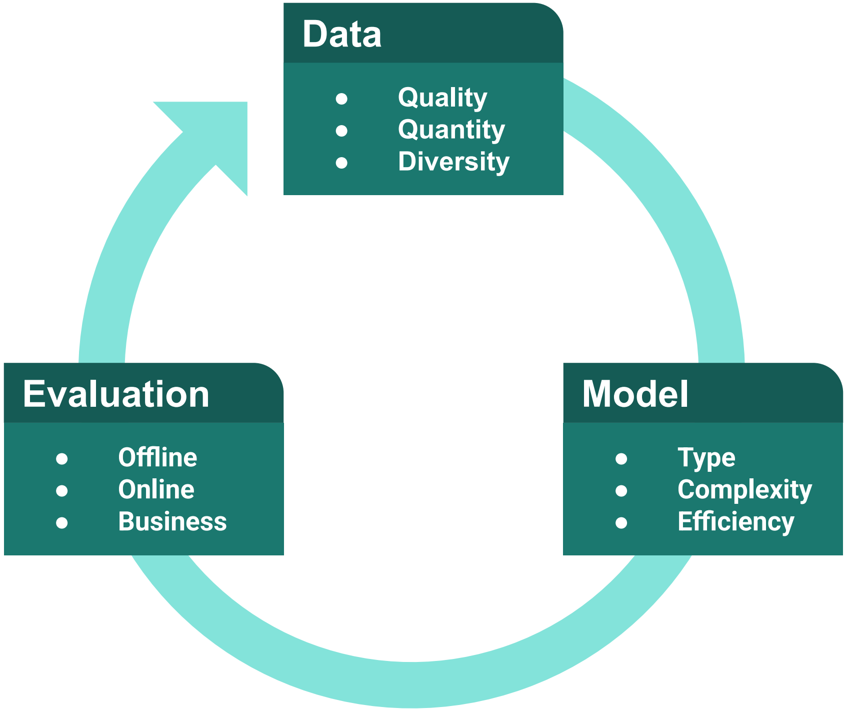
What is Machine Learning (ML)?



What is Machine Learning (ML)?

- Learn patterns from **data** 🧩
- Make **predictions** on new data 🔮
- **Generalize** beyond training 🌍

The Applied ML Life Cycle



Types of Learning

🟦 Supervised Learning

- [Data] labeled inputs → outputs
- [Goal] predict correct labels on new data
- [Ex] 📧 Spam / ✅ Diagnosis / 🛒 Ad click

🟩 Unsupervised Learning

- [Data] unlabeled, only features
- [Goal] discover hidden structure
- [Ex] 🔍 Anomaly detection / 👥 Image segmentation

Types of Learning

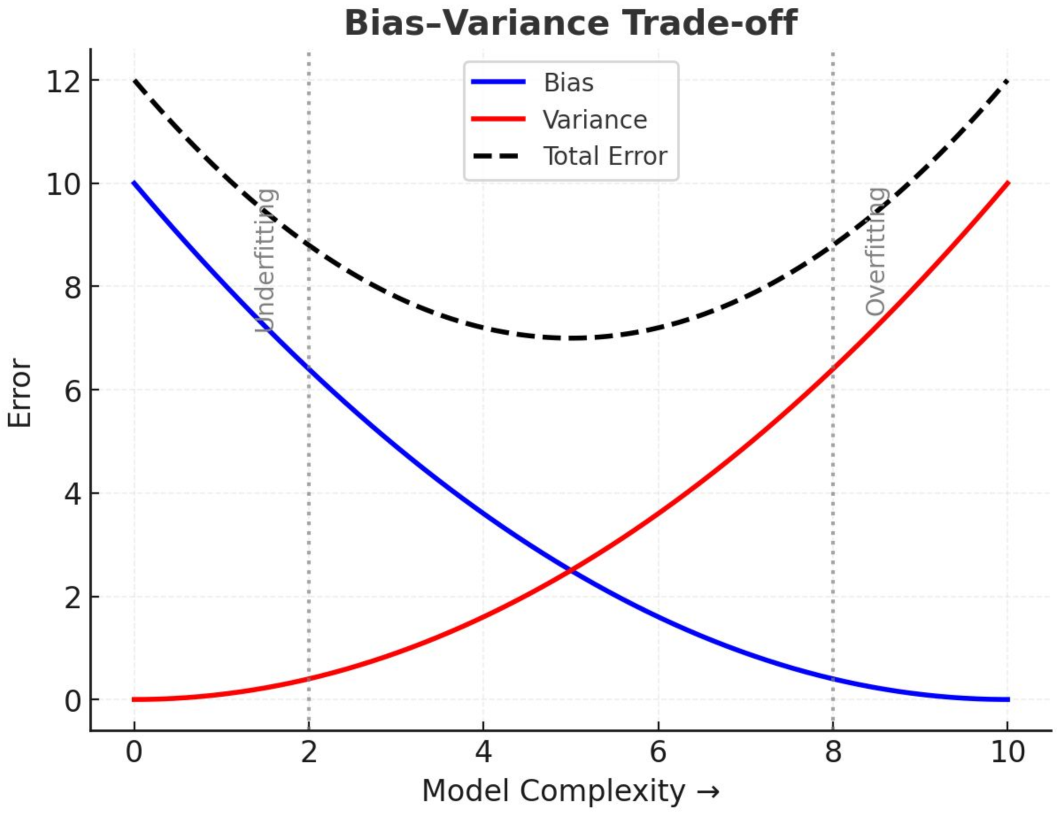
🟨 Reinforcement Learning

- [Data] interaction with environment
- [Goal] maximize expected cumulative reward
- [Ex] 🚗 Self-driving car / ♟️ AlphaGo

Key Trade-offs

- Bias vs. Variance
  - Bias: model too simple → systematic error
  - Variance: model too complex → sensitive to noise
- ⚖️ Balance needed for best generalization

Bias vs. Variance



Key Trade-offs

- Underfitting vs. Overfitting
  - Underfitting: can’t capture signal (high bias)
  - Overfitting: memorizes training data (high variance)
- 🎯 Goal: fit patterns, not noise

Key Trade-offs

- Model simplicity vs. predictive power
  - Simple models: interpretable, fast
  - Complex models: powerful, harder to trust
- 🔍 Trade-off depends on context

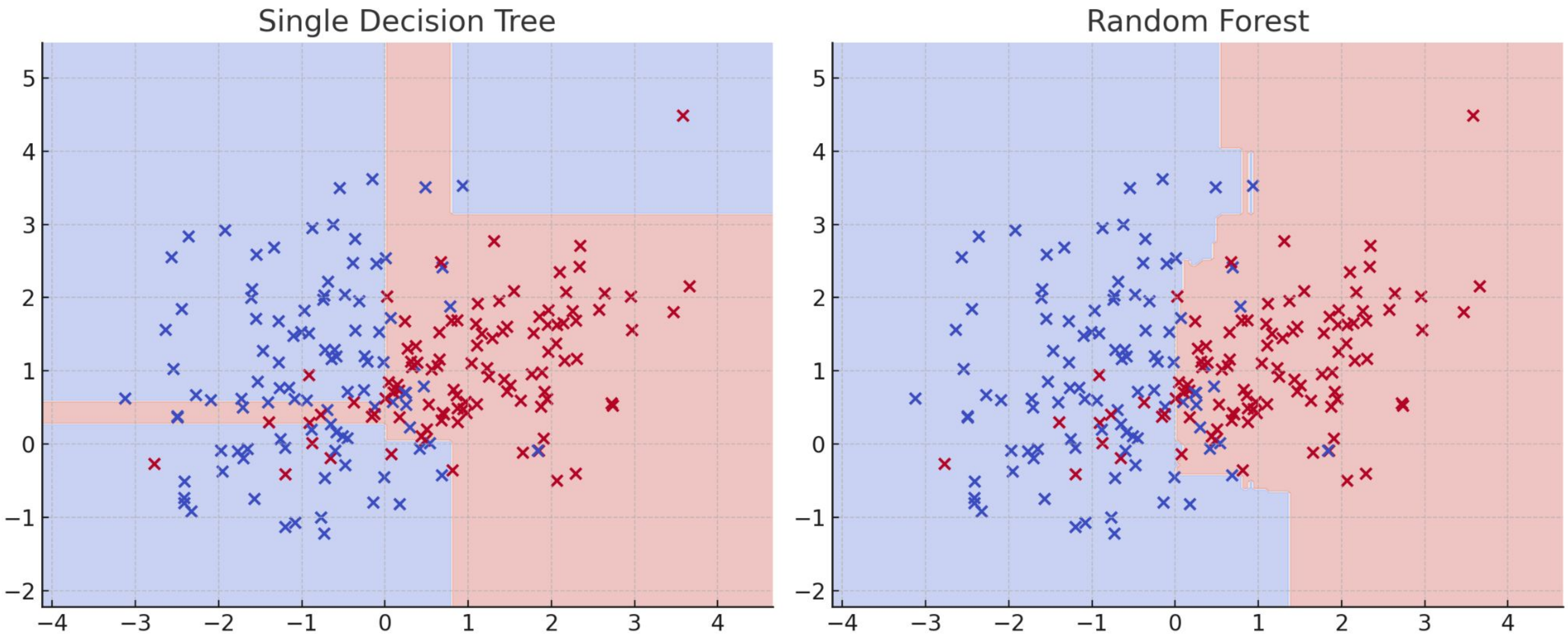
Linear Models

- **Linear Regression** → predict continuous outcomes 📈
- **Logistic Regression** → simple, powerful classification baseline ⚖️
- Strength: interpretable, efficient, widely used 🏃

Decision Trees & Ensembles

- **Decision Trees** → split features into if-else rules 🌳
- **Random Forests** → reduce variance by bagging 🌲🌲🌲
- **Boosting** → sequential error correction (XGBoost, LightGBM) 🚀

Decision Trees & Ensembles



Feature Engineering

- Transform raw data → useful signals 🔧
- Domain knowledge as leverage 📚
- Scaling, encoding, dimensionality reduction ⚖️

Representation Learning at Scale

- Learns features automatically 💡
- Breakthroughs in vision 👁️, speech 🎤, NLP 📝
- Enabled by GPUs + massive data 📊⚡

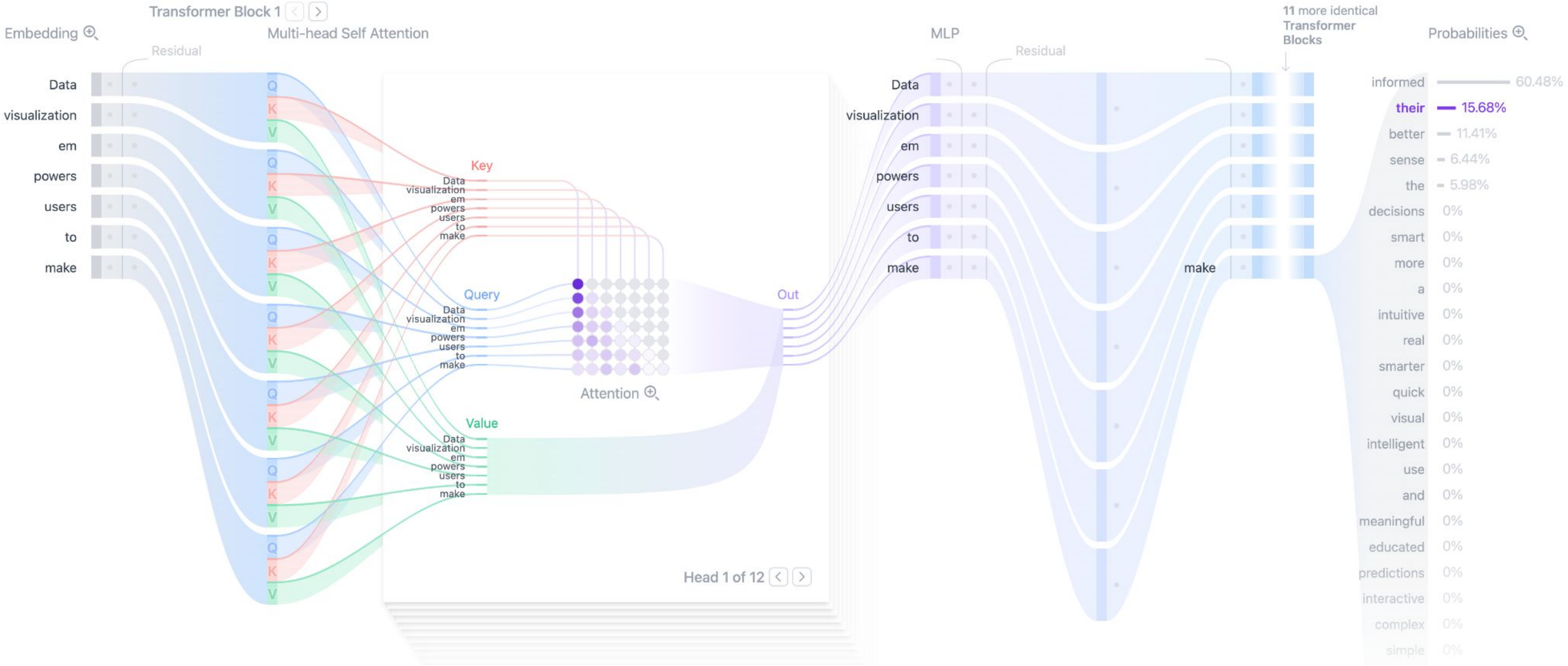
Features → End-to-End Learning

- Hand-crafted features vs. automatic representation
- Neural networks = stacked nonlinearities
- End-to-end models reduce manual work

Large Language Model

- Transformers as foundation models
- Scale: billions of parameters, massive datasets
- Capabilities: reasoning, generation, alignment

Attention all you need



ML as Iteration

- Iterative loop is the heart of ML 🔄
- Continuous experimentation mindset 🧪
- Reproducibility = credibility + progress 📑

Takeaways

- **Applied ML = Data + Model + Evaluation + Iteration** 🔄
- [Next] Workflow - scaling ML systematically 📈

Workflow

Big Picture

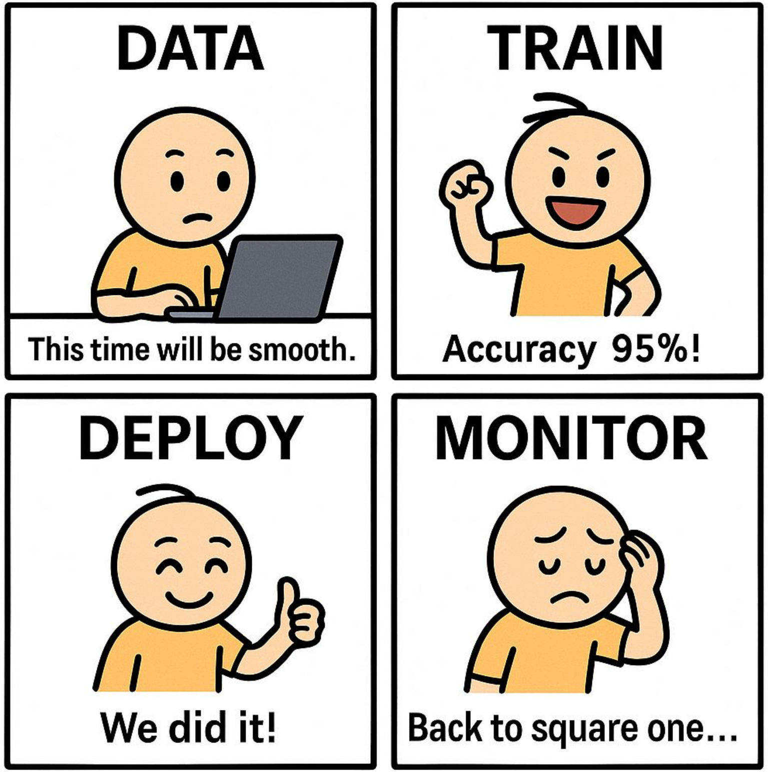
From Models to Workflows

- ML is more than training a model
- From prototype → product → lifecycle
- Workflows make ML real in practice
  - Applied ML is ALWAYS a team sport 🏈

The ML Lifecycle

- Data → Train → Deploy → Monitor 🔄
- Iterative and cyclical nature
- Feedback loop with users and environment 🌍

Life as a Member of Technical Staff



Data Pipeline

Data Collection

- Sources: logs, APIs, sensors, user input
- Bias and ethics in data collection
- Cost of free vs. expensive data

Data Cleaning & Preprocessing

- Handling missing values and outliers
- Scaling, normalization, encoding
- Importance of reproducibility in preprocessing

Feature Engineering Revisited

- Traditional feature crafting vs. learned features
- Embeddings, transfer learning
- Modern shift toward representation learning
  - What does it mean by ICLR ?

Model Deployment

Model Selection

- Matching problem type with model family
- Trade-offs: accuracy, interpretability, efficiency
- Simple baselines first, then complexity

Hyperparameter Tuning

- Grid search, random search, Bayesian optimization
- Resource constraints and parallelization
- Example
  - Tuning learning rate in DNN
  - Tuning convolutional size in CNN

Evaluation Metrics

- Beyond accuracy: precision, recall, F1, AUC
- Regression metrics: RMSE, MAE, Perplexity
- Metric alignment with business or human objectives

Serving Models

- Batch vs. real-time inference
- REST APIs, microservices, cloud deployment
- Latency, scalability, cost trade-offs

Monitoring & Feedback

- Data drift and concept drift detection
- Performance monitoring in production
- User feedback as implicit supervision

Iterative Loop

- Train → Evaluate → Deploy → Monitor → Retrain
- Continuous integration and deployment (CI/CD for ML)
- Importance of fast iteration cycles

Large Language Model

Workflow for Deep Models

- Larger datasets, distributed training
- GPUs/TPUs, mixed precision
- Scale law is everywhere
    - Data
    - Model
    - Inference

LLM-specific Workflows

- Pretraining → fine-tuning → instruction-tuning
- Prompt engineering as “feature engineering 2.0”
  - Vibe coding ☕
- Tool-call integration

Takeaways

- Workflow = bridge from foundation to production
- Each step is critical to success of ML systems
- [Next] Production – scaling and sustaining ML

Production

From Workflow to Production

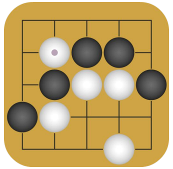
- Training a model is only the beginning
- Deployment brings new constraints and risks
- Models are part of larger socio-technical systems

Beyond Accuracy

- Latency: users expect instant results
- Cost: compute, storage, scaling
- Reliability & safety: uptime, robustness, compliance

System Challenges

What Makes ML Systems Hard

- Black-box behavior, lack of clear specifications
- Outputs not always reproducible
- Hard to test exhaustively 

Data & Scalability Issues

- ML learns patterns from data, not rules
- Inductive vs. deductive reasoning gap
- Scaling training and serving infrastructure

Failure Modes

- Overconfidence in wrong predictions
- Silent failures → hard to detect
- Cascading errors in pipelines

Components

Non-ML Aspects

- APIs, databases, caching layers
- UI/UX integration, security, logging
- Human-in-the-loop workflows

Monitoring in Production

- Detecting concept/data drift
- Live dashboards, anomaly alerts
- Collecting implicit and explicit feedback

Reliability & Testing

- Shadow testing vs. A/B testing
- Canary deployments, rollbacks
- Continuous evaluation under real load

Productionizing LLMs

- Challenges: hallucination, bias, safety
- Guardrails: filtering, alignment layers
- Monitoring outputs at scale in real-time

People

Data Scientists vs. Software Engineers

- Scientists: accuracy, models, prototyping
- Engineers: cost, reliability, deployment
- Considerations: development speed vs. production stability

T-shaped OR π-shaped People

- Broad range + deep expertise (in 1 or 2 areas)
- Example: engineer with ML + distributed systems
- Encourages team adaptability

Cross-functional Teams

- Operators, product managers, designers
- Safety & security experts, lawyers, ethicists
- Collaboration essential for trust & adoption

Case Study

NYC to test Waymo self-driving cars on crowded Manhattan and Brooklyn streets 08/25

The city will unleash the cars south of 112th Street in Manhattan in a program that a Waymo rep said Friday was already up and running.

Eight Waymo driverless cars will hit the road in Manhattan and Brooklyn.

Billy Becerra / NY Post

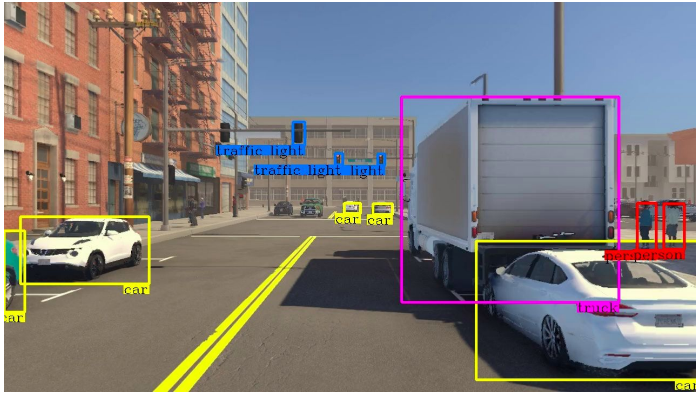
In Brooklyn, the driverless cars will roll out north of Atlantic Avenue and west of Carlton Street in neighborhoods such as Brooklyn Heights, Downtown Brooklyn and DUMBO.

Self-Driving Cars

[Tasks] perception, mapping, planning, control

Rule-Based Approach

```
object = camera.get_object()
if object.has_wheels():
    if len(object.wheels) == 4: return "Car"
    elif len(object.wheels) == 2: return "Bicycle"
return "Unknown"
```



Supervised Learning

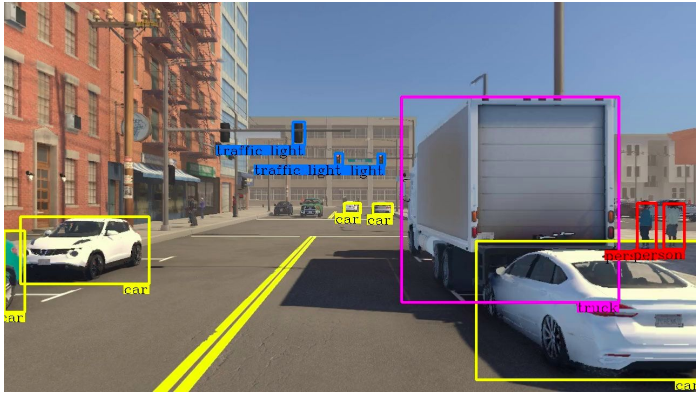
```python
from sklearn.linear_model import LogisticRegression

# features: e.g., [num_wheels, has_engine]
X = [[4, 1], [2, 0], [3, 1]]
y = ["Car", "Bicycle", "Unknown"]

model = LogisticRegression().fit(X, y)

# predict
object = camera.get_object()
features = [len(object.wheels), int(object.has_engine())]
print(model.predict([features])[0])
```

- Learn boundaries between object categories
- Useful for object detection, lane marking recognition, and traffic sign classification



Unsupervised Learning

```
img = camera.read()
pixels = img.reshape(-1, 3)

kmeans = KMeans(n_clusters=3, n_init="auto", random_state=0)
labels = kmeans.fit_predict(pixels)

seg = kmeans.cluster_centers_[labels]
```

- Group LiDAR points into clusters
- Separate cars vs. pedestrians vs. others
- Useful for perception and mapping

Reinforcement Learning

```
for episode in range(episodes):
    state = env.reset()
    while not done:
        action = policy(state)
        next_state, reward, done = env.step(action)
        update_Q(state, action, reward, next_state)
```

- Agent interacts with environment
- Reward for safe lane keeping
- Penalties for collisions
- Used in planning & decision-making

Takeaways

- Rule-based → brittle, unscalable
- ML-based → learns, adapts, scales
  - Supervised → perception (object detection)
  - Unsupervised → structure (clustering, mapping)
  - Reinforcement → control (driving policy)

Summary

- [Foundation] ML = Data + Model + Evaluation + Iteration
- [Workflow] End-to-end lifecycle: Data → Train → Deploy → Monitor
- [Production] Beyond accuracy: reliability, business goal, scalability…
- [Application] Chatbot, self-driving car, starship…


Explore the ML Space
as vast as the stars and the sea