COMS4995W31
Applied Machine Learning

Dr. Spencer W. Luo

Columbia University | Spring 2026

About this Course

COMS4995W31 - Applied Machine Learning

- Schedule: Monday 4:10pm - 6:40pm, Spring 2026
- Location: Fayerweather 310 (Morningside Campus)
- Credits: 3.0

Instructor

Dr. Spencer Luo

- Current Role:
  - Principle Research Scientist, Google DeepMind
- Background:
  - 🎓 Ph.D. in Artificial Intelligence, Carnegie Mellon University ☕ 🧋
  - 🧑‍🎓 4-time internships in Facebook 🏃 📃
  - 🚗 Self-Driving Startup 😵 🥳
  - 🤖 OpenAI 😃 😢 😵‍💫
  - ♊ Google 🚀

Our GREAT TAs

- Case Schemmer (chs2164@)
- Grace Yoon (gy2354@)
- Zoga Duka (zd2377@)
- Prajwal Raghunath (pr2789@)

Course Setup

https://columbia-coms4995.github.io/aml-spring2026/

- Syllabus - **Please read it carefully**
- Ed Discussion

Lectures: Weekly applied ML topics

- 3 assignments (Code + Report) - 60%
- 1 midterm exam - 20%
- 1 final exam - 20%

- Office Hours: (To be announced on Ed soon)

Enrollment

Fully handled by DSI student affairs now, please talk to them **directly**

POC: Robert Kramer (rk3281@columbia.edu)

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

Overview

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

- Large Language Model
  - Pre-training & supervised fine-tuning
  - Retrieval-augmented generation
  - Agentic workflows (Thinking model, LangChain, Tool integration)

This Week in AI - The "Race to Zero" Continues

Context:

DeepSeek started the "**Inference Cost War**" in Jan 2025 dropping costs by 95% 🤯

Current State (Jan 2026):

Google: Launched Gemini 3 (Agentic capabilities + Personal Intelligence)

OpenAI: Codex for Vibe Coding

DeepSeek: Rumors of V4 launching next month (Feb 2026) with "Engram Memory"

Our takeaway:

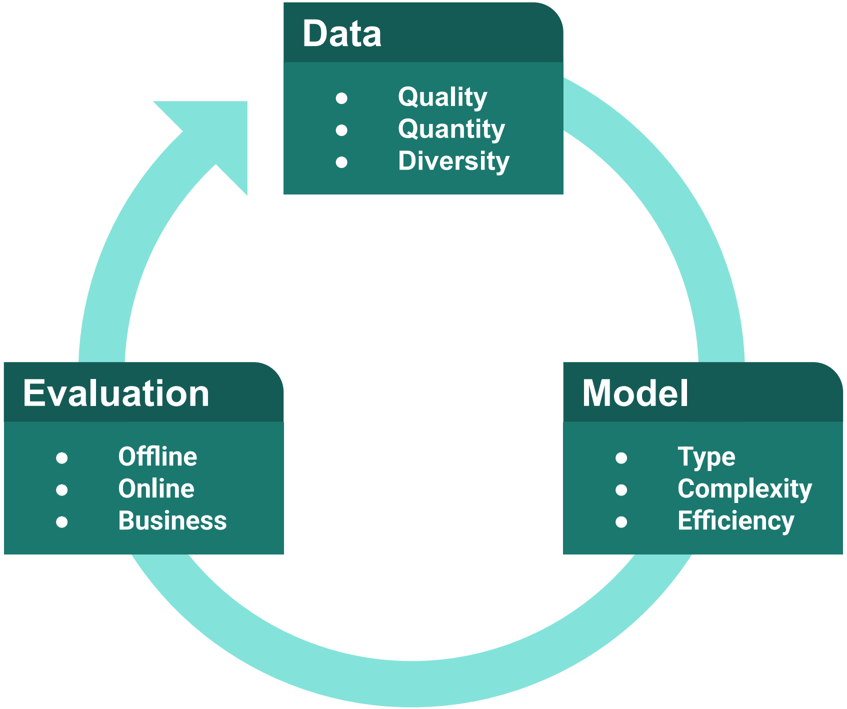
- Model Performance is converging
- **System Efficiency (Cost/Latency) is the new moat**

Introduction to AML

What is Machine Learning (ML)?

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

The Applied ML Life Cycle



Data

- Quality
- Quantity
- Diversity

Model

- Type
- Complexity
- Efficiency

Evaluation

- Offline
- Online
- Business

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

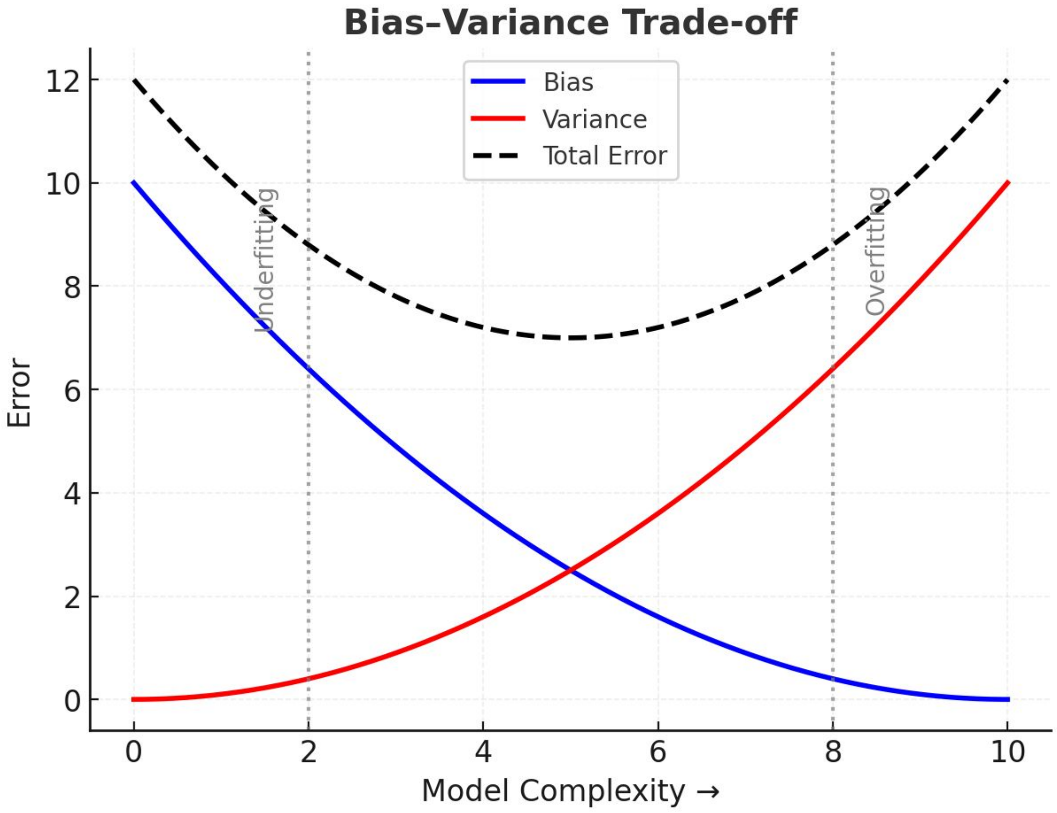
🟨 Reinforcement Learning

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

Trade-offs Discussions - Key in Applied Science

- 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

Feature Engineering

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

Features → End-to-End Learning

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

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

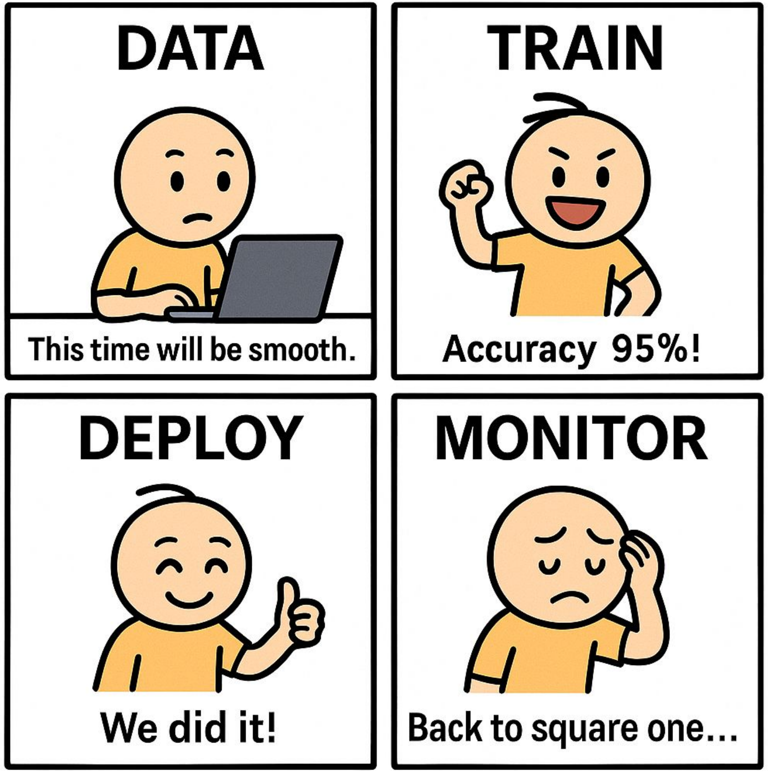
From Models to Workflows

- 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 an OAI 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 ?

☕🍵💬🤔

BREAK
TIME

Model Deployment

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

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

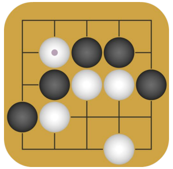
- Training a model is only the beginning
- Deployment brings new constraints and risks
- Models are part of larger socio-technical systems

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

Monitoring in Production

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

People

Data Scientists vs. Software Engineers

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

π-shaped People

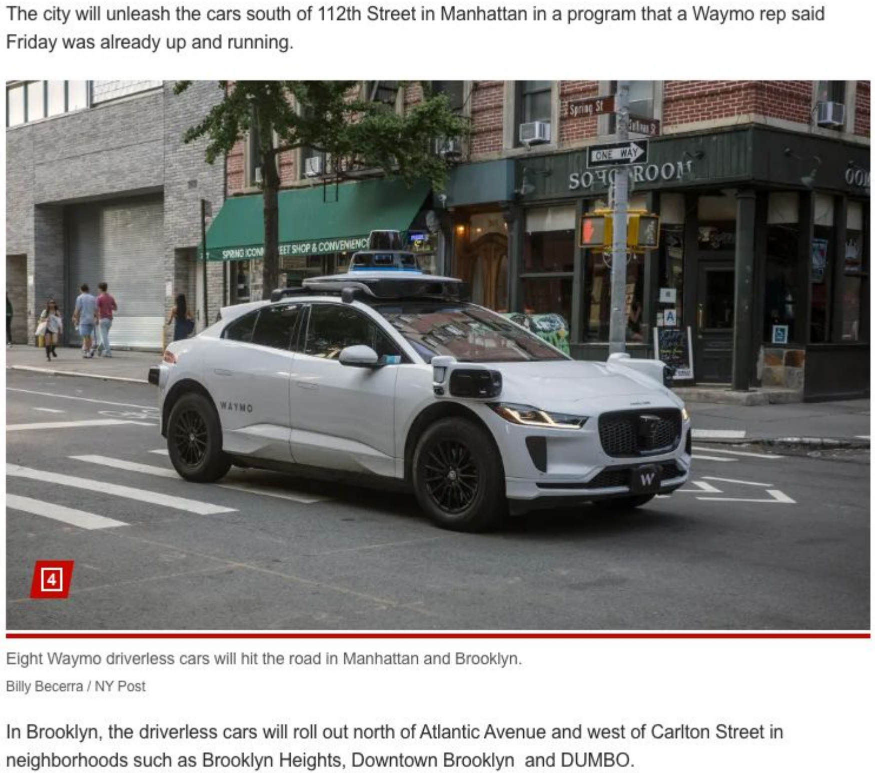
- Broad range + deep expertise (in 1 or 2 areas)
- Example: engineer with ML + distributed systems
- Encourages team adaptability
- Critical thinking on AI suggestions

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.

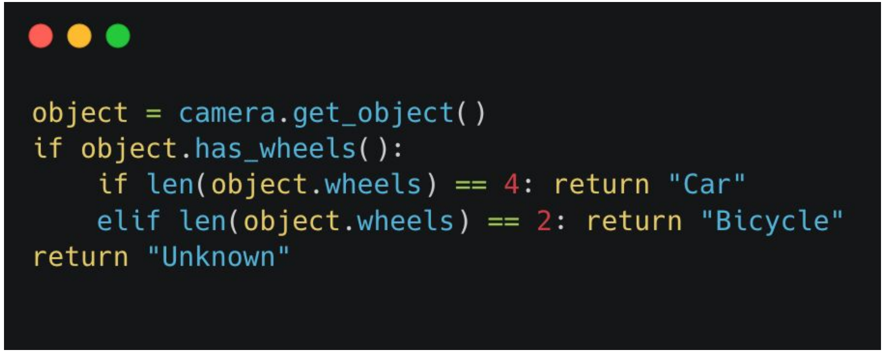
Billy Becerra / NY Post

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.

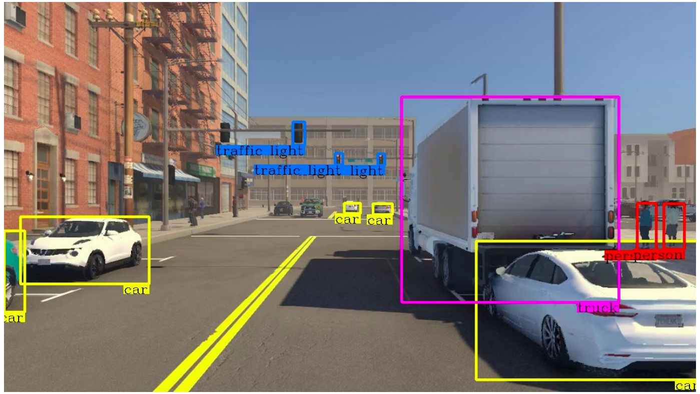
Self-Driving Cars

[Tasks] perception, mapping, planning, control

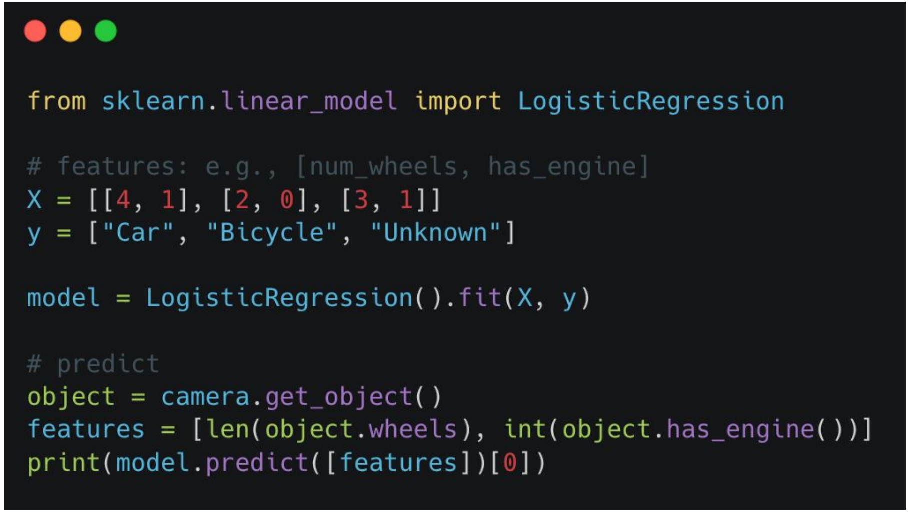
Rule-Based Approach



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

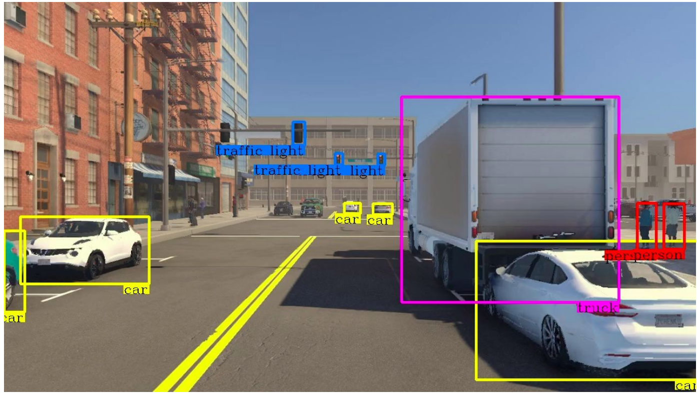


Supervised Learning

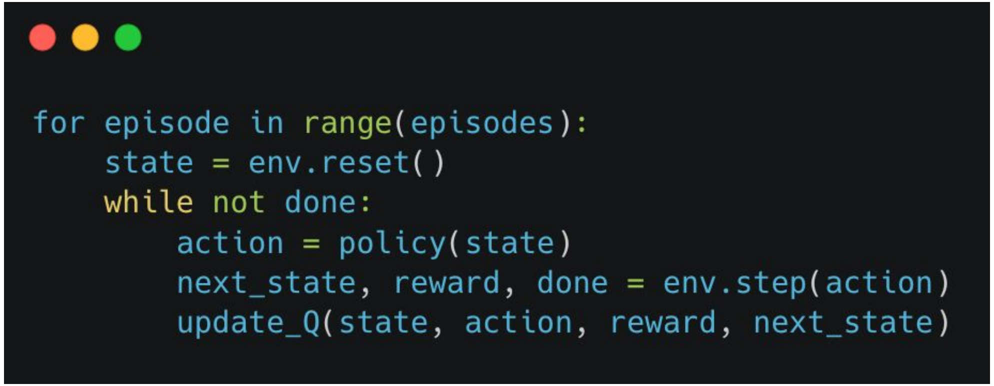


```
from sklearn.linear_model import LogisticRegression

# 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

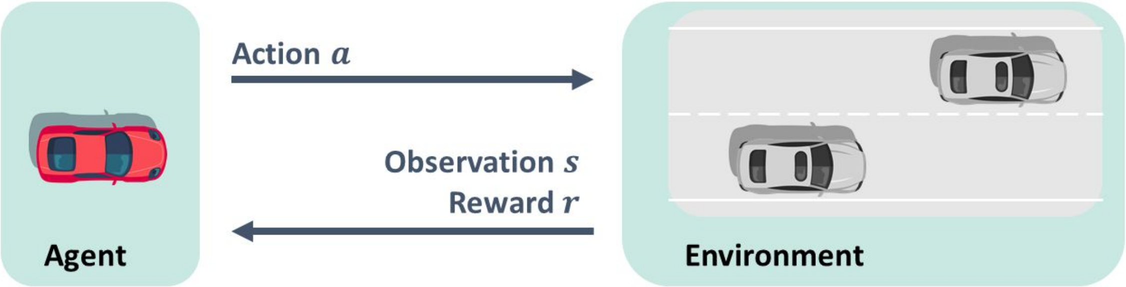


Reinforcement Learning



```
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



Industrial insight: Why 99% Accuracy is a Disaster?

The Academic View

- Model Accuracy: 99% = "State of the Art" (A+ Grade)

The Waymo Reality

- The car makes decisions at 30 FPS (Frames Per Second)
- `99%` Accuracy → `1%` Error Rate
- `1%` error every `100 frames`
- `100 frames` in 30 FPS ≈ 3.3 seconds

The Consequence

- Wait - Your 99% accurate model crashes the car every 3.3 seconds 😮

The Lesson:

- Applied ML is **NOT** about simply maximizing average accuracy
- It is about **suppressing the "Long Tail" of failures** - System Design POV

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