Applying ML to Cybersecurity Problems

AI + Cybersecurity Faculty Summer Institute

June 8, 2026

AI-Assisted Drafting Note

Warning

This deck was drafted with assistance from Codex and Claude, then reviewed slide by slide.

I have checked every slide, but there may still be some rough edges or slop.

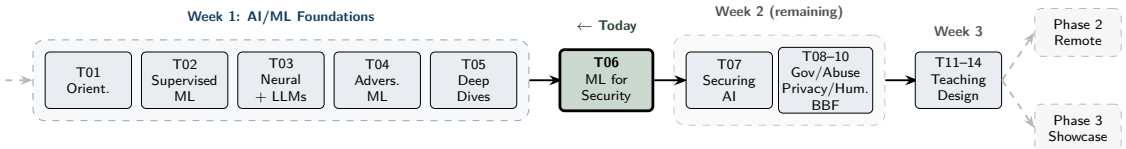
If you spot something worth tightening, clarifying, or improving, let me know.

Learning Objectives

- Map cybersecurity problems to AI/ML approaches, especially classification.
- Judge when ML is a good, limited, or poor fit for a security task.
- Treat metrics (precision, recall, false positives, false negatives) as operational decisions, not just model scores.
- Spot the trust, data, and system risks that shape classroom use of ML.
- Turn the morning's concepts into a short teaching module. Prepare for the afternoon Security Classifier Lab.

Applying ML to Cybersecurity Problems

- Week 1 built the vocabulary: supervised classification, LLMs, evaluation metrics, and trustworthy AI.
- Today we apply it to security: detection, phishing, malware, metrics as operations, and trust in AI workflows.
- Week 3 turns these decisions into teaching plans. Phases 2 and 3 extend the work into your own courses.



Outline

Outline: What's Coming Today

The morning runs in two halves with a 10-minute break in the middle. We move from **foundations** (how ML and LLMs work, what makes them trustworthy, and core security concepts) to **security operations and teaching translation**.

Before the break

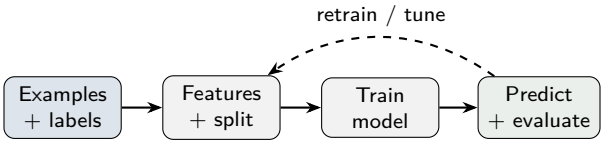
1. Supervised Learning and Classification Review
2. LLMs in Security Workflows
3. Trustworthy AI/ML
4. Review of Computer Security Concepts
5. Connecting AI/ML to Core Security and Privacy Concepts
6. Duality of AI/ML and Security
7. Application Patterns and Definitions discussion!

After the break

8. Metrics as Security Operations
9. Limitations of AI/ML Applications in Security
10. Activity: Teaching Translation
11. Summary, Conclusion, and Today's Activity

Supervised Learning and Classification Review

Supervised Learning Pipeline Refresher



- Supervised learning in security looks like:
  - Phishing detection: emails labeled phishing or safe. Features: sender, content, links.
  - Fraud detection: transactions labeled fraud or legitimate. Features: amount, location, time.
  - Anomaly detection: events labeled normal or anomalous. Features: frequency, source, type.
  - Malware classification: files labeled malicious or benign. Features: code patterns, behavior.
  - And more ...

Evaluation Refresher: Beyond Accuracy

- Accuracy counts correct predictions. It hides which errors the model makes.
- False positives and false negatives carry very different costs.
- **Example**: a phishing filter labels incoming email as phishing or legitimate.

	Predicted: legitimate	**Predicted: phishing**
Actually: legitimate	True negative	False positive
Actually: phishing	False negative	True positive

False positive: legitimate email blocked. User friction, missed work.
False negative: phishing delivered. Possible credential theft or compromise.

Precision and Recall: Phishing Filter

- **Precision** = TP / (TP + FP): of the emails flagged as phishing, what fraction really are?
- **Recall** = TP / (TP + FN): of all real phishing emails, what fraction does the filter catch?
- Moving the threshold trades one for the other. There is no free improvement.
- **Example:** A strict filter flags only the most obvious phishing, while a loose filter flags anything slightly suspicious. Which is better?

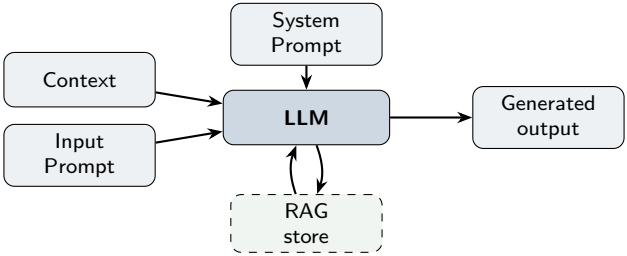
Threshold	Flagged	TP	FP	FN	Precision	Recall
Strict	12	8	4	2	67%	80%
Loose	45	9	36	1	20%	90%

Loose threshold catches more phishing (higher recall) but blocks more legitimate email (lower precision).

LLMs in Security Workflows

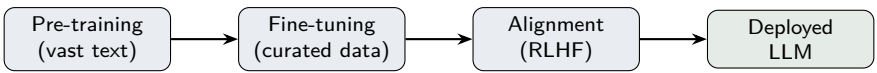
What Is a Large Language Model?

- An LLM generates text token by token. Trained on huge corpora (web, books, code), it learns language, facts, and reasoning without task-specific retraining.
- Three inputs shape each response: a **system prompt** sets the role, an **input prompt** carries the request, and **context** adds history or background.
- A **RAG** layer (dashed) retrieves documents before generation. It grounds the **output** in external knowledge, not memory alone.



How LLMs Are Trained and Aligned

- **Pre-training**: next-token prediction on billions of tokens of text, books, and code. The model learns language and world knowledge.
- **Fine-tuning** (optional): more training on curated data for a specific format or domain.
- **Alignment (RLHF)**: raters rank outputs, a reward model is trained, and the LLM is updated toward helpful, harmless, and honest responses.
  - *Alignment is a spectrum, not binary. It depends on the training data, rater guidelines, and reward model quality.*
  - *It does not remove hallucination, prompt injection, or misuse.*



- **Security risks:** over-alignment can refuse legitimate red-team work. Adversarial fine-tuning of open-weight models can strip guardrails or plant backdoors.

Example LLMs in Security Application Areas

Defensive / Analyst Support

- Summarize incident reports and alert queues
- Extract indicators of compromise from unstructured logs
- Explain alert rationale to analysts in plain language
- Draft response plans, user notifications, and remediation steps
- Search and synthesize past incidents (RAG over knowledge base)
- Generate synthetic training data for other ML models

Offensive / Red Team

- Draft realistic phishing and spearphishing emails
- Simulate adversary behavior for tabletop exercises
- Scan code for vulnerability patterns and suggest exploits
- Help write proof-of-concept attack scripts

The same capabilities that help defenders also help attackers. Policy and access controls matter.

LLMs vs. Traditional ML for Classification

- LLMs can classify by prompting ("Is this email phishing? Reply yes or no."). No labeled data needed: zero-shot or few-shot.
- Traditional classifiers (logistic regression, random forest, SVM) train end-to-end for one task. They optimize for speed, calibration, and volume.

	Traditional ML	**LLM**
Labeled data needed	Yes, often many	Few or none
Throughput / latency	High / low	Lower / higher
Probability calibration	Yes	Limited
Free-text reasoning	No	Yes
Explainability	Depends on model	Prompt-driven
Adversarial robustness	Studied, known	Less characterized

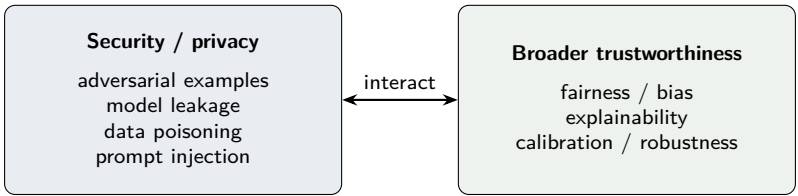
In practice: use a **traditional classifier** for high-volume scoring. Use an **LLM** where flexible reasoning or explanation matters. Hybrid pipelines are common.

Trustworthy AI/ML

Trustworthy AI/ML: Security, Privacy, and Beyond

- Trustworthy AI/ML asks:
  - Is the model secure against manipulation and data leakage?
  - Does it respect privacy and avoid exposing sensitive data?
  - Is it fair, explainable, calibrated, and robust enough to rely on?
- Some issues are direct security and privacy concerns: adversarial examples, model leakage, data poisoning, prompt injection.
- Others are broader but touch security: fairness, explainability, calibration, robustness.
- These categories are not independent:
  - An insecure model can leak training data.
  - A biased model gives uneven false positives that erode trust.
  - An opaque model can hide poisoning or failure modes.
  - A model that degrades on new inputs is open to adversarial examples and drift.

Trustworthy AI/ML: Security, Privacy, and Beyond (continued)



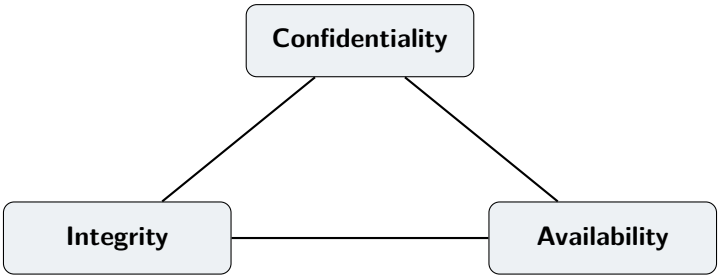
e.g., a biased model amplifies disparate alert rates; an opaque model hides poisoning; a leaky model exposes private training data.

Review of Computer Security Concepts (next section with AI examples)

What Makes a System Secure?

- Security is about protecting assets under an intended policy, not just making software run.
- A system can work for normal users and still fail under deliberate pressure.
- Review standard language for security concepts to connect them to AI/ML applications later.

The CIA Triad



- Confidentiality: only authorized parties should access the data.
- Integrity: data and decisions stay correct, untampered, and traceable to their source.
- Availability: information, services, and resources are usable when needed.
- *AI systems stress all three at once. Prompts, retrieved context, and outputs all affect what is exposed, changed, or delayed.*

Example: CIA in Online Banking App

- An online banking app lets customers log in, view balances and history, and move money.

	Credentials & Personal Data	**Account Balances & History**	**Payments & Transfers**
C	Protect logins, PII, and account access	Keep balances and history private	Hide payment details from unauthorized users
I	Prevent identity or account tampering	Records must stay accurate and consistent	Transfers must go to the right recipi- ent and amount
A	Customers must still log in when needed	Users need timely access to balances	Payments and transfers must remain available

Beyond CIA: Authentication, Authorization, Accountability

- **Authentication**: who is interacting with the system?
- **Authorization**: what data or functions should they be allowed to access?
- **Accountability**: what actions can we trace and review later?

Example: AAA in Online Banking App

- Banking systems need to know who a user is, what they may do, and what can be audited later.
- The same three questions apply to people, services, and automated workflows.

	Login & Identity	**Accounts & Records**	**Payments & Transfers**
Authentication	Verify the customer logging in	Tie account data to the right identity	Confirm who requested a transfer
Authorization	Only valid users should reach banking features	Users should access only their own records	Only permitted users should move money or pay bills
Accountability	Record who logged in and from where	Maintain an auditable history of account actions	Transfers should leave a clear transaction trail

Trusted vs. Trustworthy

Trusted

The system relies on something.

Trustworthy

That component deserves the trust the system places in it.

- Security work often starts by noticing where we trust components too easily.
- AI example: developers often trust model output, retrieved documents, external APIs, or parsers more than those components justify.

Privacy, Confidentiality, and Anonymity

- **Confidentiality** is a security property: unauthorized parties should not access data.
- **Privacy** is about what personal or sensitive information should be protected.
- **Anonymity** is about breaking the link between an action and a person's identity.

Example: Online Banking App

- Banking systems must protect account data, transactions, and customer access from unauthorized disclosure.

	Credentials & Personal Data	**Account Balances & History**	**Payments & Transfers**
Confidentiality	Protect logins, PII, and account access	Keep balances and history private	Hide payment details from unauthorized users
Privacy	Limit what support tools and alerts can reveal	Avoid unnecessary sharing of transaction history	Minimize transaction metadata exposure
Anonymity	Keep help requests and browser sessions unlinked	Avoid tying actions to a named user when possible	Limit public traces of transfers and payments

Assets, Policy, and Adversaries

- **Assets:** what we want to protect.
- **Policy:** what the system should do and should not do.
- **Adversaries:** who wants to violate that policy to affect an asset.

Example: Assets, Policy, and Adversaries in Online Banking App

- Banking systems must protect account data, transaction history, and transfer capability from misuse.

	Credentials & Access	**Balances & History**	**Payments & Transfers**
Assets	Customer login data and account access	Account records and transaction history	Transfer capability and payment instructions
Policy	Only the account owner should reach logins	Balances should stay accurate and private	Only authorized transfers should go through
Adversaries	Phishers, credential thieves, and fraudsters	Curious insiders or attackers seeking account data	Attackers trying to redirect or forge payments

Threats, Vulnerabilities, and Mitigations

- **Threat:** a plausible way an asset could be harmed.
- **Vulnerability:** a weakness that makes the threat easier.
- **Mitigation:** a countermeasure that prevents, detects, or limits harm.

Example: Threats, Vulnerabilities, and Mitigations in Online Banking App

- Banking systems must guard against account theft, record tampering, and payment fraud.
- The same threat can exploit different weaknesses, so mitigation must match the failure path.

	Credentials & Access	**Balances & History**	**Payments & Transfers**
Threat	Phishing or stolen credentials	Tampering with account records	Fraudulent transfers or payment redirection
Vulnerability	Weak login checks or overbroad session access	Poor record validation or stale data handling	Insufficient transfer review or recipient validation
Mitigation	MFA and stronger login controls	Record integrity checks and audit trails	Transfer confirmation and transaction monitoring

Connecting AI and ML to Core Security and Privacy Concepts

Two Running Examples

- We reuse the prior section's security vocabulary on two AI/ML workflows.
- **Phishing classifier in a SOC:** email evidence becomes a score, an alert, a queue item, and an analyst decision.
- **LLM/RAG alert assistant:** alerts, logs, and retrieved knowledge become a summary for analyst review.
- *For both, ask the same questions: what are the assets, what policy applies, who is the adversary, and where can the workflow fail?*

Assets, Policy, and Adversaries

Phishing Classifier

- **Assets:** emails, labels, model, threshold, alert queue, user trust.
- **Policy:** flag suspicious mail without overwhelming analysts or blocking real work.
- **Adversaries:** phishers, credential thieves, and attackers adapting message style.

LLM/RAG Alert Assistant

- **Assets:** alerts, logs, retrieved documents, prompts, summaries, analyst decisions.
- **Policy:** only authorized context should shape summaries used for triage.
- **Adversaries:** prompt injectors, insiders, poisoned sources, and attackers hiding evidence.

CIA in the Two Examples

Phishing Classifier

- **Confidentiality:** protect email contents, user reports, and analyst notes.
- **Integrity:** preserve labels, features, scores, and threshold decisions.
- **Availability:** keep scoring and alert routing working during active campaigns.

LLM/RAG Alert Assistant

- **Confidentiality:** protect logs, prompts, retrieved context, and summaries.
- **Integrity:** preserve retrieved evidence, prompt assembly, output parsing, and citations.
- **Availability:** keep retrieval, model access, and analyst support up during incidents.

AAA in the Two Examples

Phishing Classifier

- **Authentication:** who submitted reports, who reviewed alerts.
- **Authorization:** who can see sensitive messages, labels, alert logs.
- **Accountability:** record scores, threshold changes, analyst actions.

LLM/RAG Alert Assistant

- **Authentication:** which service or analyst queried the assistant.
- **Authorization:** who can see retrieval sources, prompts, and summaries.
- **Accountability:** record retrieved context, summaries, and overrides.

Trusted versus Trustworthy Components

Phishing Classifier

- A SOC may **trust** a classifier score enough to quarantine an email or send it to review.
- The score is **trustworthy** only if labels, features, drift monitoring, and evaluation back that trust.

LLM/RAG Alert Assistant

- An analyst may **trust** an LLM/RAG summary enough to guide triage.
- The summary is **trustworthy** only if its context is authorized, relevant, complete, and checked against evidence.

Threats, Vulnerabilities, and Mitigations

Phishing Classifier

- **Threat:** attacker rewrites messages to evade detection.
- **Vulnerability:** stale features, weak labels, or a brittle threshold.
- **Mitigation:** retraining review, drift monitoring, user reports, and human triage.

LLM/RAG Alert Assistant

- **Threat:** malicious text in retrieved context steers the summary.
- **Vulnerability:** untrusted retrieval or unsafe prompt assembly.
- **Mitigation:** source controls, context filtering, validation, and review before action.

Reusable Teaching Questions

- What are the assets, and what policy is supposed to protect them?
- Who is the adversary, and what threat path are they using?
- Which CIA and AAA concerns appear before, during, and after the model step?
- Which components are merely trusted, and what evidence would make them trustworthy?
- Which mitigation belongs at data intake, model use, output handling, logging, or human review?

Duality of AI/ML and Security

Two Directions, Same Examples

- The same two examples run in two directions.
  - **Using AI/ML for security:** the classifier or assistant helps defenders protect users and systems.
  - **Securing AI/ML systems:** the classifier or assistant is now the system that must be protected.
- *The rest of this topic focuses mainly on using AI/ML to support cybersecurity tasks, and the other direction will be the focus of the next topic.*

Phishing Classifier: Two Security Directions

Using AI/ML for security

- The classifier helps the SOC process email evidence at scale.
- Inputs may include message text, URLs, sender metadata, attachments, past labels, and user reports.
- The model scores or ranks messages so the workflow can warn, quarantine, route, or review.
- Threat focus: phishing misses and alert noise that hurt defense operations.

Securing AI/ML systems

- Now the classifier, training data, labels, threshold, and alert routing are assets.
- Risks: evasion, poisoned data, label corruption, stale features, drift, and threshold manipulation.
- Controls: data provenance, pipeline review, adversarial testing, monitoring, and analyst feedback loops.
- Threat focus: attacks on the model workflow itself.

LLM/RAG Alert Assistant: Two Security Directions

Using AI/ML for security

- The assistant helps analysts work through noisy alerts, logs, prior tickets, and internal knowledge.
- Retrieval supplies context. The LLM summarizes, clusters, explains, or drafts an incident timeline.
- The output should improve analyst understanding, not replace evidence checking.
- Threat focus: alert fatigue and analyst gaps that slow triage and incident response.

Securing AI/ML systems

- Now prompts, retrieved documents, summaries, logs, tool calls, and model access are the assets.
- Risks: prompt injection, data leakage, unauthorized retrieval, hallucinated summaries, unsafe tool use, downtime.
- Controls: retrieval filtering, access control, prompt/context separation, output validation, logging, human approval.
- Threat focus: attacks on the assistant workflow itself.

Teaching Moments: Comparing the Two Directions

Section takeaways

- AI/ML for security asks what the model helps the security team do.
- Securing AI/ML asks what parts of the workflow must be protected from attack or misuse.
- The same example can appear in both directions, but the questions change.
- For the rest of this topic, keep the focus on AI/ML as a security tool.

Teaching moves

- Trace one example twice: once as a security workflow, once as a protected AI workflow.
- Compare what changes in the asset list, the failure modes, and the human role.
- Ask students which direction the lesson teaches, and why that matters.

Application Patterns and Definitions

What Makes a Pattern?

- A useful pattern combines the security task, the model role, the evidence it uses, the human action that follows, and the cost of mistakes.
- The same security task can appear in different patterns.
- The same model family can serve different patterns.
- Pattern labels help students compare examples without getting stuck on model names.

Pattern Labels to Use in the Discussion

- **Classify**: decide whether an event, message, file, or account belongs in a category.
- **Rank**: prioritize cases, vulnerabilities, alerts, or reports for attention.
- **Cluster**: group related activity or repeated evidence.
- **Summarize / retrieve / explain**: help people interpret evidence while preserving verification.
- **Sensemake**: organize messy evidence so a human can decide what to do next.

- **Discussion:** Turn to your neighbor, see how many security examples can you come up with for each label that could use an AI/ML model.

Application Patterns: Security Examples at a Glance

- **Intrusion detection** — Classify/rank anomalies from telemetry for analyst triage.
- **Phishing & social engineering** — Classify/rank message risk to support user and analyst review.
- **Malware classification** — Classify/cluster files to guide containment and reverse engineering.
- **Spam, abuse, fraud** — Rank/flag repeated harmful behavior for blocking or escalation.
- **Vulnerability prioritization** — Rank patching work by exploitability, impact, and asset context.
- **Log analysis & SOC triage** — Cluster/summarize alerts into cases for faster analyst decisions.
- **Threat intelligence** — Retrieve/summarize/explain reports into testable analyst hypotheses.

Teaching Moments: Security Applications

Section takeaways

- Pattern labels help students compare real security tasks across settings.
- The same pattern can support different decisions depending on evidence, action, and error cost.
- Good examples make the human role visible, not just the model role.
- Classroom fit depends on whether the task is classify, rank, cluster, summarize, retrieve, or explain.

Teaching moves

- Pattern sorting: students place tasks under pattern labels.
- Workflow trace: students map data, model task, threshold, and action on a one-page worksheet.
- Fit check: students decide whether the pattern is a good classroom match and what makes it viable.

Let's Take a 10-Minute Break!

Metrics as Security Operations

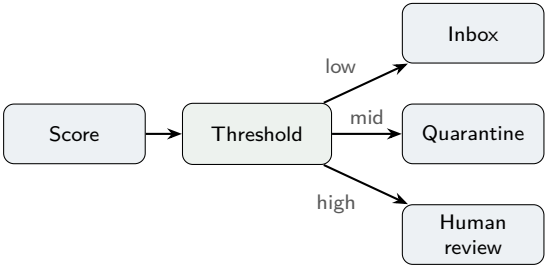
Section Intro: Metrics as Security Operations

- Metrics measure model performance but have security impact
- Continue with running examples anchor this section:
  - A **phishing classifier** that scores messages for inbox, quarantine, or human review.
  - An LLM/RAG assistant that summarizes alerts and logs to guide **SOC triage**.
- It's not just if the model is right or wrong, but choosing good thresholds and the **human impact** and **decision making**.

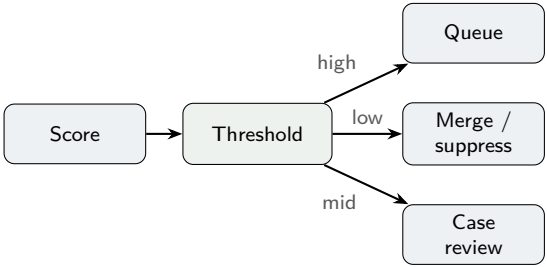
From Model Output to Security Work

- A model score is not a decision; it is evidence that supports a decision.
- Thresholds encode who sees work and who does not.

Phishing classifier



SOC triage assistant



Impact of Metrics on Operations

- The same statistical error can have very different security consequences.
  - A low threshold may let attacks through but keeps the queue manageable.
  - A high threshold may catch more attacks but overwhelms analysts with false positives.

Phishing classifier

	Pred. benign	Pred. malicious
Actual benign	TN normal mail	FP quarant. legit
Actual malicious	FN missed phishing	TP caught phishing

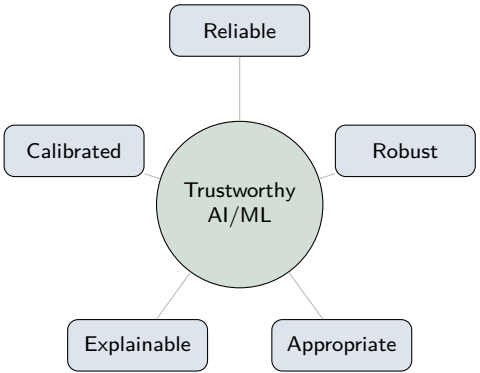
SOC triage assistant

	Pred. benign	Pred. malicious
Actual benign	TN ordinary event	FP queue noise
Actual malicious	FN missed incident	TP useful alert

- Impact of FNs: A few phishing emails getting throuhgh may be less problematic than a few real incidents getting missed.
- Impact of FPs: A few legitimate emails getting quarantined may be less problematic than a few false alerts overwhelming the SOC.

Trustworthiness Beyond Accuracy

- A model that scores well in testing can still fail in deployment. Data shift, adversarial adaptation, and base-rate mismatch all change outcomes.
- Miscalibrated confidence drives overtrust. Analysts accept high-certainty outputs and skip review, so errors spread.
- Unexplainable alerts cannot be verified before action. The analyst must blindly follow or dismiss the output.
- The five dimensions on the right define what it takes for AI to earn trust under real conditions, not just test conditions.



Teaching Moments: Metrics Tradeoffs

Section takeaways

- Model scores become security work when thresholds route cases to people or automation.
- FP and FN costs are workflow costs: quarantined mail, queue overload, missed incidents.
- Thresholds encode policy; shifting them redistributes work and can create blind spots.
- Trustworthy deployment needs more than accuracy: reliability, calibration, robustness, explainability, and fit.

Teaching moves

- Threshold tuning: explore how moving a decision boundary shifts outcomes and workload.
- Error cost analysis: compare the operational consequences of different mistakes.
- Trustworthiness scorecard: assess a system against several deployment dimensions at once.
- Alert load simulation: reason about how volume, staffing, and routing affect workflow.

Limitations of AI/ML Applications in Security

Section Intro: Limitations of AI/ML Applications in Security

- AI/ML can support security work, but it does not remove judgment, context, or workflow cost.
- The question is not just whether the model works, but where it stops helping.
- **Example:**
  - Phishing classifier: value depends on the quarantine and recovery path, not raw accuracy.
  - SOC alert assistant: value depends on fit with triage, not summary quality alone.

Key point

Security usefulness depends on workflow fit, not model performance alone.

Limits of Accuracy and Coverage

- Accuracy is not the same as operational reliability.
- False positives and false negatives both matter in security settings.
- A model can look strong in evaluation and still create operational problems.
- **Example:**
  - Phishing classifier: good accuracy can still mean missed attacks or too many quarantines.
  - SOC alert assistant: useful summaries can still miss important signals or overstate confidence.

Key point

Good scores do not eliminate security risk.

Limits of Data, Labels, and Ground Truth

- Security data is noisy, incomplete, and shaped by human judgment.
- Labels encode policy and can shift what the model learns.
- Ground truth is often local to a team, organization, or time period.
- **Example:**
  - Phishing classifier: mislabels change what gets flagged.
  - SOC alert assistant: curation choices change what incidents are retained and summarized.

Key point

The model inherits the limits of the data and the labels.

Limits of Context and Generalization

- Security workflows change over time.
- Models trained in one environment may not transfer cleanly to another.
- Attacker behavior, policy changes, and infrastructure shifts reduce stability.
- **Example:**
  - Phishing classifier: a classifier trained on one campaign may miss the next.
  - SOC alert assistant: a summarizer that works for one alert format may fail on another.

Key point

Security context changes faster than static model assumptions.

Limits of Human Trust and Review

- People still need to verify, override, or escalate model outputs.
- Over-trust and under-trust are both failure modes.
- Human review is part of the system, not an optional add-on.
- **Example:**
  - Phishing classifier: users need a recovery path for false positives.
  - SOC alert assistant: analysts need evidence to trust or challenge a summary.

Key point

Trust must be calibrated, not assumed.

Limits of Deployment and Operations

- A model's value depends on queue size, staffing, routing, and escalation paths.
- Threshold changes shift work, not just scores.
- Model updates can change workload and support burden.
- **Example:**
  - Phishing classifier: tighter thresholds may reduce misses but overload recovery requests.
  - SOC alert assistant: more aggressive summarization may reduce analyst load but hide important context.

Key point

Operational fit determines whether the system helps or creates noise.

Teaching Moments: Limitations of AI/ML Applications in Security

Section takeaways

- AI/ML helps security work only within clear limits on data, context, and trust.
- The model does not remove judgment, context, or operational cost.
- The same application can look useful in a demo and still struggle in deployment.

Teaching moves

- Ask what the model misses and what evidence would reveal that gap.
- Ask what the human still has to do when the model is wrong or uncertain.
- Ask what a false positive or false negative costs in real work.
- Ask what changes when the environment, policy, or user population changes.

Activity: Teaching Translation

Activity: Design a Teaching Seed

In teams, build one teaching seed.

Pick a teaching moment (right), then work three steps:

1. **Application.** A security application from your own class that fits the moment.
2. **Example.** A concrete example that makes the moment accessible there.
3. **Pedagogy.** The activity, prompt, or move that brings the moment to the fore.
4. **Share.** Be ready to share your seed in the next 10 minutes.

Teaching moments

1. **Score vs. action:** a threshold turns a score into block, allow, or escalate.
2. **Error costs:** false positives and false negatives cost differently.
3. **Pattern fit:** classify, rank, cluster, summarize, or retrieve predicts fit.
4. **Calibrated trust:** when to trust, override, or escalate; trust is earned.
5. **Limits:** where AI/ML stops helping, and what the human still does.

Summary and Conclusion

Morning Summary

Main themes

- Core security framing (assets, policy, CIA/AAA, trust) carries over to AI/ML workflows.
- The same task maps to a pattern (classify, rank, cluster, summarize, retrieve), and pattern fit shows where AI/ML helps.
- Security metrics are operational choices: thresholds, false positives, and false negatives change workload and risk.
- Every useful application has limits tied to data, context, trust, and human review.

Teaching translation

- Start with a concrete security task, then ask what the model helps with.
- Surface the tradeoff students should notice, not just the score to report.
- Make the human decision point visible in the activity or case study.
- End with a question students can carry into later labs or projects.

Today's Activity

Today's Activity: IDS Classifier Lab

Design, translation, and AI

- **Design:** metrics become security reasoning. FP/FN and thresholds drive an alert decision; a real-vs-synthetic data gap surfaces distribution shift.
- **Your translation:** design a similar lab for your course (phishing filter, log/SOC triage, fraud, vuln ranking).
- **AI assistance:** I built it end to end with AI as a bounded aid, keeping the goals, framing, and assessment. `AGENTS.md` sets the same boundary for students.

The activity

- Student-facing **IDS lab**: label network flows as benign or attack.
- Runs in Google Colab over a synthetic 13-column flow dataset.
- Compare at least three classifiers (logistic regression, decision tree, random forest).
- Evaluate with a train/validation split, five-fold CV, and a private real-data competition test set.
- Deliver a notebook, a reproducible `best_model.py`, and written notes.
- Competition based on your submission with a hidden test set of real network flows. The best score wins a prize.

AI-Assisted Drafting Note

Warning

This deck was drafted with assistance from Codex and Claude, then reviewed slide by slide.

I have checked every slide, but there may still be some rough edges or slop.

If you spot something worth tightening, clarifying, or improving, let me know.