# APPLICATION OF MACHINE LEARNING ALGORITHMS IN CYBERSECURITY SYSTEMS

**Ashurov Umidjon Akhmadjon ugli**
*Main Department of Security of The National Guard of The Republic of Uzbekistan*

**Abstract:** *The rapid growth of networked services, cloud platforms, industrial control systems, and Internet of Things devices has increased both the volume of security telemetry and the complexity of cyberattacks. Conventional signature-based protection remains effective against previously catalogued threats, but it is often insufficient for detecting novel, polymorphic, low-and-slow, or behaviorally adaptive attacks. This article examines the scientific foundations and practical use of machine learning algorithms in cybersecurity. It presents supervised, unsupervised, semi-supervised, and deep-learning approaches for intrusion detection, malware classification, phishing detection, user and entity behavior analytics, fraud monitoring, and threat-intelligence enrichment. Mathematical formulations are provided for logistic regression, support vector machines, anomaly scoring, entropy-based decision trees, and evaluation metrics. Particular attention is paid to dataset imbalance, concept drift, interpretability, adversarial machine learning, model governance, and human-in-the-loop deployment. A reference architecture is proposed in which machine learning augments, rather than replaces, established security controls. The analysis shows that operational value depends less on selecting a single "best" algorithm than on rigorous data engineering, calibrated evaluation, continuous monitoring, and integration with risk-management and incident-response processes.*

**Keywords:** *machine learning, cybersecurity, intrusion detection, anomaly detection, malware classification, adversarial machine learning, threat intelligence, model governance*

Cybersecurity systems operate in an environment characterized by high-dimensional data, incomplete information, rapidly changing attacker behavior, and asymmetric costs of error. Security operations centers collect network flows, domain name system records, endpoint telemetry, authentication events, process trees, file metadata, email content, cloud audit logs, and vulnerability information. Manual analysis of these data streams is slow and difficult to scale. Rule-based systems can automate known patterns, but their effectiveness is constrained when attackers modify indicators, distribute malicious activity over time, or exploit previously unseen vulnerabilities.

Machine learning (ML) addresses part of this problem by learning statistical relationships from historical and streaming data. In cybersecurity, a model may estimate whether an event is malicious, identify observations that deviate from a learned baseline, group related events, rank alerts by risk, or extract entities and relations from threat reports. The principal benefit is not autonomous decision-making in isolation; it is the ability to convert large volumes of heterogeneous telemetry into prioritized evidence for analysts and automated controls. NIST's Artificial Intelligence Risk Management Framework emphasizes that trustworthy AI must be governed, mapped to context, measured, and managed throughout

its life cycle [1]. This principle is especially relevant to cybersecurity because model errors may either block legitimate activity or allow malicious behavior to pass undetected.

The objective of this article is to systematize the main ML algorithms used in security, explain their mathematical basis, compare their operational characteristics, and identify conditions required for reliable deployment. The discussion treats ML as one component of defense-in-depth. Signature systems, access control, cryptography, vulnerability management, backups, secure configuration, and incident response remain indispensable.

A cybersecurity dataset can be represented as D = {(xi, yi)}i=1{N}, where xi is a feature vector, and yi is a class label such as benign, phishing, ransomware, command-and-control, or privilege escalation. Features may be continuous, categorical, sequential, graph-based, or textual. Typical examples include packet size statistics, connection duration, failed-login frequency, executable import tables, byte n-grams, URL lexical properties, process-parent relationships, and embedding vectors generated from security logs.

Supervised learning uses labeled examples to estimate a function f(x) that predicts y. It is appropriate when labels are sufficiently accurate and representative. Unsupervised learning does not require explicit class labels and is commonly applied to clustering and anomaly detection. Semi-supervised learning combines a small labeled subset with a larger unlabeled set, which is useful because high-quality security labels are expensive. Self-supervised and representation-learning methods learn reusable features from raw data before downstream classification. Online learning updates the model as new events arrive and can improve adaptation to concept drift, although it also creates additional governance and poisoning risks.

Feature engineering must reflect both statistical relevance and operational meaning. For example, the number of failed authentication attempts may be predictive, but it should be normalized by user role, time of day, device, and historical baseline. Data leakage must be prevented: a feature created after incident confirmation cannot be used to claim real-time predictive performance. Temporal train-test splitting is usually more realistic than random splitting because it evaluates whether a model trained on past behavior generalizes to future threats.

Logistic regression is a transparent baseline for binary security classification. Given a feature vector x and parameters w and b, the probability of a malicious class is estimated by the sigmoid function:

$$P(y = 1 \mid x) = \sigma(w^{T}x + b) = 1 / (1 + e^{-(w^{T}x+b)}) \quad (1)$$

The parameters are obtained by minimizing the regularized cross-entropy loss:

$$L(w,b) = -(1/N) \Sigma [yi \ln pi + (1-yi) \ln(1-pi)] + \lambda||w||^{2}_{2} \quad (2)$$

In security operations, logistic regression is suitable for phishing scoring, suspicious-login classification, and alert prioritization when the relationship between features and risk is approximately linear. Coefficients can be inspected to identify influential variables, which supports auditability. Its main limitation is reduced performance when attack behavior depends on complex nonlinear interactions.

Decision trees recursively partition the feature space. A split is selected to reduce class impurity. For a node containing class probabilities p_k, the Shannon entropy is:

$$H(S) = - \Sigma_{k=1}^{K} pk \log_{2}(pk) \quad (3)$$

The information gain of a candidate split is the parent entropy minus the weighted entropy of child nodes. Tree models are attractive in cybersecurity because they handle mixed feature types and nonlinear thresholds. Random forests average many decorrelated trees, reducing variance and providing robust performance on tabular network-flow and endpoint datasets. Gradient-boosted trees sequentially correct previous errors and often achieve strong detection accuracy, but poorly calibrated models can produce overconfident scores. Feature importance must also be interpreted cautiously because correlated variables can distort rankings.

A support vector machine (SVM) finds a separating hyperplane with maximum margin. For the soft-margin linear case, the optimization problem is:

$$\min_{w,b,\xi} \ (1/2)\|w\|^2 + C \Sigma \xi i \quad (4)$$

$$\text{subject to } yi(w^T xi+b) \geq 1-\xi i, \quad \xi i \geq 0. \quad (5)$$

Kernel functions allow nonlinear decision boundaries. SVMs have been used for malware, spam, and intrusion classification, particularly on medium-sized datasets with engineered features. They can be computationally expensive for very large telemetry streams, and probability estimates require additional calibration.

Anomaly detection is important when malicious examples are rare or unknown. A simple statistical anomaly score can be based on the Mahalanobis distance:

$$A(x) = \sqrt{((x-\mu)^T \Sigma^{-1}(x-\mu))} \quad (6)$$

Here $\mu$ and $\Sigma$ are the mean vector and covariance matrix of a baseline population. More flexible algorithms include isolation forest, one-class SVM, local outlier factor, clustering, and autoencoders. An autoencoder learns an encoder $g(\cdot)$ and decoder $h(\cdot)$, and uses reconstruction error as an anomaly score:

$$A(x) = \|x - h(g(x))\|^2_2. \quad (7)$$

The central challenge is that unusual does not necessarily mean malicious. Software updates, new employees, seasonal workloads, and legitimate administrative activity can trigger anomalies. Therefore, anomaly scores should be enriched with asset criticality, identity context, threat intelligence, and analyst feedback before automated containment is performed.

Deep neural networks can learn hierarchical representations from raw or weakly processed data. Convolutional networks can classify byte sequences or transformed malware representations; recurrent networks and transformers can model ordered events such as authentication sequences, API calls, shell commands, or process chains. Graph neural networks can analyze relationships among users, devices, files, domains, and processes. These approaches are useful where the security signal depends on temporal or relational context, but they require substantial data, computing resources, and careful validation. Their complexity also increases the need for explainability and drift monitoring.

**Table 1.** Comparative characteristics of ML algorithms in cybersecurity

| Algorithm | Typical security use | Advantages | Limitations | Interpretability |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| Logistic regression | Phishing, login risk, alert ranking | Fast, calibrated baseline, coefficient inspection | Mostly linear boundary | High |
| Random forest / boosting | Network-flow and endpoint classification | Strong tabular performance, nonlinear interactions | May overfit; importance can be misleading | Medium |
| SVM | Malware, spam, intrusion classification | Effective in high-dimensional spaces | Scaling and calibration challenges | Low to medium |
| Isolation forest / one-class methods | Unknown-threat and behavior anomaly detection | Little or no labeled attack data required | High false-positive risk | Medium |
| Autoencoder / deep network | Sequence, raw-byte, and high-volume telemetry | Learns complex representations | Data and compute intensive; opaque | Low |
| Graph neural network | Identity, process, and infrastructure relationships | Captures relational attack paths | Complex graph construction and validation | Low |

ML-based intrusion detection systems analyze packet, flow, and protocol metadata to classify or rank suspicious activity. A practical pipeline includes traffic collection, sessionization, feature extraction, model inference, correlation, and analyst review. Models should be evaluated on realistic traffic that contains modern applications, encrypted protocols, and diverse attack families. Public benchmark datasets are useful for reproducibility, but they often differ from operational networks in class distribution, topology, and behavior. Consequently, external validation on local telemetry is necessary before deployment [4].

Static malware models use file headers, imported functions, strings, byte patterns, certificate information, and structural metadata. Dynamic models use process behavior, system calls, registry changes, network destinations, and memory indicators observed in a controlled environment. Static analysis is fast but can be affected by packing and obfuscation; dynamic analysis is behaviorally informative but more expensive. Hybrid models combine both sources. Because malware authors deliberately modify samples to evade detectors, robustness testing must include feature manipulation, distribution shift, and adversarial examples.

Phishing models may combine lexical URL features, domain-registration attributes, transport-layer certificate data, page structure, sender behavior, and language representations.

The model output should be integrated with email authentication results, reputation systems, sandboxing, and user reporting. False positives have high operational cost because they can block legitimate business communication; therefore, threshold selection should be based on risk tolerance and review capacity rather than accuracy alone.

User and entity behavior analytics establishes baselines for accounts, devices, applications, and service identities. A model may detect impossible travel, unusual resource access, credential sharing, abnormal privilege use, or service-account misuse. Context is essential: a system administrator and an office employee naturally have different behavioral distributions. Per-entity and peer-group models can reduce noise. Privacy, proportionality, data minimization, and access controls should be incorporated because behavioral monitoring can expose sensitive information.

Natural-language processing can extract indicators, malware names, vulnerabilities, organizations, and adversary techniques from unstructured reports. ML can also cluster incidents, deduplicate alerts, and infer likely next actions. MITRE ATT&CK provides a knowledge base of adversary tactics and techniques derived from real-world observations [2]. Mapping model outputs to ATT&CK can improve communication between detection engineering, threat intelligence, and incident response. However, automated mapping should preserve uncertainty and supporting evidence.

Security datasets are commonly imbalanced; benign events may outnumber malicious events by several orders of magnitude. Accuracy is therefore insufficient. Let TP, TN, FP, and FN denote true positives, true negatives, false positives, and false negatives. The principal metrics are:

$$\text{Precision} = TP / (TP + FP) \quad (8)$$

$$\text{Recall} = TP / (TP + FN) \quad (9)$$

$$F_1 = 2 \cdot \text{Precision} \cdot \text{Recall} / (\text{Precision} + \text{Recall}) \quad (10)$$

$$\text{False Positive Rate} = FP / (FP + TN). \quad (11)$$

Suppose a model processes 100,000 events containing 500 confirmed attacks. It detects 450 attacks and incorrectly flags 950 benign events. Then TP = 450, FN = 50, FP = 950, and TN = 98,550. The resulting accuracy is 99.0%, which appears excellent, but precision is only 32.1%, meaning approximately two of every three alerts are false. Recall is 90.0%, and $F_1$ is 47.4%. This example demonstrates why high accuracy can coexist with an excessive analyst workload.

**Table 2.** Confusion-matrix calculation

| Actual / Predicted | Malicious | Benign |
|---|---|---|
| Malicious | TP = 450 | FN = 50 |
| Benign | FP = 950 | TN = 98,550 |

Threshold selection should account for asymmetric costs. A missed attack against a critical server may be more costly than investigating several additional alerts, while automated account blocking may require a much lower false-positive rate. Precision-recall curves are generally more informative than receiver operating characteristic curves under severe imbalance. Metrics should also be reported by attack family, asset group, time period,

and confidence interval. Cross-validation must preserve temporal and organizational boundaries.

Cybersecurity models operate against adaptive opponents. Adversarial machine learning includes evasion attacks at inference time, poisoning of training data, model extraction, privacy attacks, and backdoors. NIST provides a taxonomy and terminology for these threats [3], while MITRE ATLAS records tactics and techniques targeting AI-enabled systems [5]. In an evasion scenario, an attacker seeks a modified input $x'$ that preserves malicious functionality while changing the model decision:

find $x'$ such that $f(x') \neq f(x)$, subject to $C(x,x') \leq \varepsilon$ and functionality$(x')$=malicious. (12)

In malware detection, modifications may include adding benign-looking sections, changing nonfunctional metadata, or altering behavior without removing the malicious objective. In anomaly detection, an attacker may gradually adapt activity to remain within the learned baseline. Poisoning attacks contaminate model updates or feedback channels so that future malicious behavior is treated as normal. Defensive measures include provenance controls, robust feature design, adversarial testing, ensemble diversity, rate limiting, secure model registries, signed artifacts, monitoring of feature distributions, and separation of model recommendations from irreversible actions.

No single robustness technique is sufficient. Adversarial training can improve performance against known perturbation models but may not generalize to new attacker objectives. Input validation is useful but cannot fully define malicious semantics. Operational resilience therefore requires layered controls, continuous red-team testing, rollback capability, and conventional detection mechanisms that remain effective when the ML component fails.

A reliable deployment architecture separates data collection, feature processing, model training, model registry, inference, alert correlation, response, and audit logging. Training and production environments should be isolated. Every model version should have documented data lineage, intended use, prohibited use, evaluation results, approval status, dependencies, and rollback procedures. The model should produce a score and supporting evidence rather than an unexplained categorical decision whenever possible.

A human-in-the-loop workflow is appropriate for high-impact actions. Low-risk decisions, such as ranking alerts, can be automated more aggressively. High-risk actions, such as disabling privileged accounts or isolating critical systems, should require corroboration from multiple controls or analyst approval. The four AI RMF functions—Govern, Map, Measure, and Manage—provide a useful structure for assigning responsibility and monitoring risk [1].

**Table 3.** Recommended lifecycle controls for security ML

| Lifecycle stage | Required controls | Main failure modes |
|---|---|---|
| Data acquisition | Provenance, labeling rules, privacy controls, temporal partitioning | Leakage, bias, poisoning, incomplete coverage |

| | | |
|---|---|---|
| Training | Reproducible pipeline, baseline comparison, class weighting, calibration | Overfitting, unstable features, misleading accuracy |
| Validation | Time-based tests, subgroup analysis, adversarial tests, cost analysis | Poor generalization, hidden false positives |
| Deployment | Signed model, controlled API, fallback logic, analyst evidence | Model theft, unsafe automation, integration errors |
| Monitoring | Drift detection, alert quality, latency, feedback audit | Concept drift, feedback loops, silent degradation |
| Retirement | Rollback, archive, dependency removal, documentation | Unsupported models and residual access paths |

The first limitation is concept drift. Network architecture, software, user behavior, and attacker tools change continuously, so a model trained on historical data can degrade without obvious failure. Drift detection should monitor both input distributions and outcome metrics. The second limitation is label uncertainty: incident labels may be delayed, incomplete, or affected by analyst disagreement. Probabilistic labels and active learning may reduce the cost of obtaining reliable annotations.

The third limitation is explainability. Feature attribution methods can assist investigation, but they do not automatically prove causality or model correctness. Explanations should be tested for stability and aligned with security evidence. The fourth limitation is reproducibility. Many published results use different preprocessing, train-test partitions, and attack distributions, making direct comparison difficult. Future research should prioritize temporally realistic benchmarks, cross-organization validation, calibrated uncertainty, robust online learning, privacy-preserving collaboration, and evaluation against adaptive adversaries.

Federated learning and privacy-enhancing technologies may support collaborative threat detection without centralizing raw telemetry, but they introduce communication, trust, and poisoning challenges. Graph-based and multimodal models may improve correlation across identity, endpoint, network, and cloud data. Nevertheless, operational success will depend on whether these methods reduce time to detection and response without producing unsustainable alert volume.

**Conclusions**

Machine learning provides substantial value in cybersecurity by classifying known patterns, identifying anomalies, modeling sequences and relationships, enriching threat intelligence, and prioritizing analyst attention. Logistic regression, tree ensembles, SVMs, anomaly detectors, deep networks, and graph models each have distinct strengths; algorithm selection should be based on data structure, latency, interpretability, and cost of error.

Evaluation must account for class imbalance, temporal drift, attack-family variation, and the operational consequences of false positives and false negatives.

ML systems also create new attack surfaces. Evasion, poisoning, extraction, and backdoor threats require secure data pipelines, adversarial testing, model integrity controls, and fallback mechanisms. The most defensible design is a layered, threat-informed architecture in which ML augments established security controls and human judgment. Scientific rigor in data preparation, reproducible validation, calibrated decision thresholds, continuous monitoring, and governance is more important than reporting a single high accuracy value. Under these conditions, ML can improve cyber-defense scalability while maintaining accountability and operational resilience.

**References:**

[1] National Institute of Standards and Technology, "Artificial Intelligence Risk Management Framework (AI RMF 1.0)," NIST AI 100-1, 2023. doi: 10.6028/NIST.AI.100-1.

[2] MITRE, "MITRE ATT&CK: A Knowledge Base of Adversary Tactics and Techniques," official knowledge base. Accessed: 22 June 2026.

[3] A. Vassilev, A. Oprea, A. Fordyce, and H. Anderson, "Adversarial Machine Learning: A Taxonomy and Terminology of Attacks and Mitigations," NIST AI 100-2e2025, 2025. doi: 10.6028/NIST.AI.100-2e2025.

[4] M. E. Hodo et al., "Survey on Intrusion Detection Systems Based on Machine Learning Techniques for the Protection of Critical Infrastructure," Sensors, vol. 23, no. 5, art. 2415, 2023. doi: 10.3390/s23052415.

[5] MITRE, "MITRE ATLAS: Adversarial Threat Landscape for Artificial-Intelligence Systems," official knowledge base. Accessed: 22 June 2026.

[6] R. S. S. Kumar et al., "Adversarial Machine Learning—Industry Perspectives," in Proc. IEEE Security and Privacy Workshops, 2020, pp. 69–75. doi: 10.1109/SPW50608.2020.00028.

[7] I. Rosenberg, A. Shabtai, Y. Elovici, and L. Rokach, "Adversarial Machine Learning Attacks and Defense Methods in the Cyber Security Domain," ACM Computing Surveys, vol. 54, no. 5, art. 108, 2021. doi: 10.1145/3453158.

[8] N. Moustafa and J. Slay, "UNSW-NB15: A Comprehensive Data Set for Network Intrusion Detection Systems," in Proc. Military Communications and Information Systems Conference, 2015, pp. 1–6. doi: 10.1109/MilCIS.2015.7348942.

[9] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. Cambridge, MA: MIT Press, 2016.

[10] C. M. Bishop, Pattern Recognition and Machine Learning. New York: Springer, 2006.

[11] T. Fawcett, "An Introduction to ROC Analysis," Pattern Recognition Letters, vol. 27, no. 8, pp. 861–874, 2006. doi: 10.1016/j.patrec.2005.10.010.

[12] J. Davis and M. Goadrich, “The Relationship Between Precision-Recall and ROC Curves,” in Proc. 23rd International Conference on Machine Learning, 2006, pp. 233–240. doi: 10.1145/1143844.1143874.