Assessing Industrial Internet of Things Security at the Network Edge Using Trust-based Centralized and Federated Machine Learning

by Jeffrey C. Yu

Bachelor of Science in Engineering, Electrical Engineering, June 1989,
Princeton University
Master of Science, Electrical Engineering, January 1991, Stanford University
Degree of Engineer, Electrical Engineering, October 1992, Stanford University

A Praxis submitted to

The Faculty of
The School of Engineering and Applied Science
of The George Washington University
in partial fulfillment of the requirements
for the degree of Doctor of Engineering

August 31, 2024

Praxis directed by

Arthiraiyan Appulingam
Professorial Lecturer in Engineering and Applied Science

The School of Engineering and Applied Science of The George Washington University certifies that Jeffrey C. Yu has passed the Final Examination for the degree of Doctor of Engineering as of July 24, 2024. This is the final and approved form of the Praxis.

Assessing Industrial Internet of Things Security at the Network Edge Using Trust-based Centralized and Federated Machine Learning

Jeffrey C. Yu

Praxis Research Committee:

Arthiraiyan Appulingam, Professorial Lecturer in Engineering and Applied Science, Praxis Director

Amir Etemadi, Associate Professor of Online Engineering Programs, Committee Chair

Hamza F. Alsarhan, Professorial Lecturer in Engineering and Applied Science, Committee Member

© Copyright 2024 by Jeffrey C. Yu
All rights reserved

Dedication

The author wishes to dedicate this praxis to his parents, Tim and Nancy Yu, who instilled in their children and grandchildren a respect for the power of education and lifelong learning and modeled the core value of always striving to better the lives and livelihoods of their family.

Also dedicated to the loving memory of the author's brother Gary Yu.

Acknowledgements

The author wishes to thank his wife Hanh Cao Yu, children Caroline and Jackie Yu, and son-in-law Kevin Moyung for their boundless patience and unconditional support of this midlife reentry into higher education to pursue this degree and undertake this course of research. The author also wishes to thank John Russell and the Chief Information Office team at Northrop Grumman for encouraging the start of this educational journey amid the start of the organization's own journey to digital maturity. Colleagues Brian Gallagher, Dennis McCallam, Alan Easterling, and Sandy Forney demonstrated the viability of late career doctoral achievement and inspired the author's own pursuit and perseverance.

The author wishes to thank Dr. Arthiraiyan Appulingam for sharing his own experiences and providing a steady hand in guiding the development of this praxis. The author expresses appreciation to the faculty and staff of The George Washington University School of Engineering & Applied Sciences Cybersecurity Analytics program for their dedication to providing a quality and challenging doctoral education. The author would like to express appreciation for the robust online data science communities that provided a valuable environment to gain basic proficiency in machine, deep, and federated learning. Lastly, the author would like to thank his Doctor of Engineering in Cybersecurity Analytics DC3 cohort classmates—especially John Kuk, Jim Johnstone, Jillian Ong, and Anthony Balagao—for their collegiality, late-night camaraderie, and genuine friendship and support throughout this shared journey.

Abstract of Praxis

Assessing Industrial Internet of Things Security at the Network Edge Using Trust-based Centralized and Federated Machine Learning

The Internet of Things (IoT) in industrial and factory environments is rapidly growing toward enabling $3.3 trillion in economic value by 2030 (Chui et al., 2021). Increasingly, the deployment of Industrial Internet of Things (IIoT) is bringing information technology (IT) together with operational technology (OT) for the digitalization of industrial and safety-critical systems, ranging from factories to scientific laboratories to energy and public utility systems to transportation and national critical infrastructure. Many IoT devices provide little visibility into their services, configurations or operating states, which presents a security black box risk with poorly characterized behavior to the overall environment (CISA, 2020). Of deployed IoT devices, 57% are vulnerable to medium- or high-severity malware attacks (Wang & Lu, 2020).

This praxis details research that improves an organization's cyber resilience by providing a predictive trust model using federated learning at the network edge where IoT and IIoT devices are deployed to monitor and control the physical environment. Metadata commonly available in a network operating environment measure the trustworthiness of IoT/IIoT device behavior over time and form the attributes for an extensible machine learning model training and testing methodology. This research demonstrates that the methodology is effective as measured by F1-score and practical for detecting and classifying cyberattacks in a heterogenous IT/OT environment using conventional tree-based and boosted tree machine learning, multilayer perceptron and TabNet transformer deep learning, and horizontal federated learning architectures.

Table of Contents

Dedication iv

Acknowledgements v

Abstract of Praxis vi

List of Figures x

List of Tables xii

List of Equations xiv

List of Key Abbreviations and Acronyms xv

Glossary of Terms xviii

Chapter 1—Introduction 1

1.1 Background 1

1.2 Research Motivation 3

1.3 Problem Statement 5

1.4 Thesis Statement 6

1.5 Research Objectives 6

1.6 Research Questions and Hypotheses 7

1.7 Scope of Research 8

1.8 Research Limitations 8

1.9 Organization of Praxis 9

Chapter 2—Literature Review 10

2.1 Introduction 10

2.2 Operational Technology and Industrial Internet of Things Security 11

2.3 Machine Learning Applications and Measures for IoT Security 16

2.4 Edge Computing and Federated Learning 23
2.5 Trust Frameworks, Architectures, and Computation 29
2.6 Summary and Conclusion 34
Chapter 3—Methodology 37
3.1 Introduction 37
3.2 Data Sources 37
3.3 Data Preprocessing and Exploratory Data Analysis 38
3.4 Feature Engineering and Selection 40
3.4.1 Filter Methods 43
3.4.2 Embedded Methods 45
3.4.3 Wrapper Methods 46
3.5 Benchmark Model Development and Classification 48
3.5.1 Final Data Preprocessing and Splitting for Modeling 48
3.5.2 Centralized Conventional Machine Learning Modeling 49
3.6 Deep Learning 51
3.7 Federated Architecture 53
3.8 Performance Evaluation 56
Chapter 4—Results 60
4.1 Introduction 60
4.2 Data Source Selection 60
4.3 Exploratory Data Analysis and Data Preprocessing Results 62
4.4 Feature Engineering and Selection Results 66
4.4.1 Trust-based Feature Engineering and Correlation 66

4.4.2 Filter Methods Results 69
4.4.3 Embedded Methods Results 73
4.4.4 Wrapper Methods Results 75
4.4.5 Overall Feature Selection Results 77
4.5 Centralized Conventional Machine Learning Baseline Results 79
4.5.1 Binary Classification Results 80
4.5.2 Multiclass Classification Results 85
4.6 Centralized Deep Learning Results 88
4.6.1 Multilayer Perceptron Deep Neural Network 88
4.6.2 TabNet 93
4.7 Federated Learning Results 98
4.7.1 Binary Classification 100
4.7.2 Multiclass Classification 104
Chapter 5—Discussion and Conclusions 108
5.1 Research Hypothesis Findings and Discussion 108
5.1.1 Research Question 1: IoT/IIoT Trust Measures at the Network Edge. 108
5.1.2 Research Question 2: Machine and Deep Learning Performance 110
5.1.3 Research Question 3: Federated Learning at the Network Edge 112
5.2 Conclusions and Recommendations for System Implementation 113
5.3 Contributions to Body of Knowledge 116
5.4 Recommendations for Future Research 118
References 120
Appendix 130

List of Figures

Figure 1-1. IoT Layered Reference Architecture and Representative Security 3

Figure 1-2. Simplified Architecture for Praxis Research Experiment 5

Figure 3-1. Data Preprocessing and Exploratory Data Analysis 39

Figure 3-2. Feature Engineering and Selection Architecture 40

Figure 3-3. Centralized Machine Learning Architecture 48

Figure 3-4. Horizontal Federated Learning Architecture 54

Figure 4-1. Feature Correlation with Binary Classification Target 69

Figure 4-2. Univariate Filter Feature Selection: Chi-squared ($\chi 2$) 70

Figure 4.3. Univariate Filter Feature Selection: Analysis of Variance (ANOVA) 71

Figure 4.4. Univariate Filter Feature Selection: Information Gain 72

Figure 4-5. Univariate Feature Selection: Combined Results 72

Figure 4-6. Embedded Feature Selection: Random Forest Classifier 73

Figure 4-7. Embedded Feature Selection: Combined Results 74

Figure 4-8. Forward Sequential Feature Selection 77

Figure 4-9. Binary Classification with Core 16 Features: Receiver Operating Curves 82

Figure 4-10. Binary Classification with Core 16 Features: Precision-Recall Curve 83

Figure 4-11. Binary Classification Results with Core 16 Features 84

Figure 4-12. Random Forest Multiclass Confusion Matrix with Core 16 Features 87

Figure 4-13. MLP Binary Loss and Accuracy: 39 Features, Batch Size 8196 90

Figure 4-14. MLP Binary Loss and Accuracy: 39 Features, Batch Size 2048 90

Figure 4-15. MLP Binary Loss and Accuracy: Core 16 Features, Batch Size 2048 93

Figure 4-16. TabNet Binary Classification Loss and Accuracy: Full 39 Features 95

Figure 4-17. TabNet Binary Classification Loss and Accuracy: Core 16 Features..... 96

Figure 4-18. Binary Classification Loss: All 39 and Core 16 Features with 4 Clients... 101

Figure 4-19. Binary Classification Loss: Core 16 Features with 8 and 16 Clients..... 101

Figure 4-20. Binary Classification Metric Curves: Core 16 Features with 4 Clients..... 101

Figure 4-21. Binary Classification Confusion Matrices for 39 Features after Federation Rounds 1, 10, 25, and 50..... 104

Figure 4-22. Federated Multiclass Confusion Matrix: Core 16 Features with 4 Clients 105

Figure 4-23. Federated Multiclass Performance Curves: 39 Features with 4 Clients 106

Figure 4-24. Federated Multiclass Classification Loss Curves: 39 Features..... 106

Figure A-1. NF-ToN-IoT-v2 Standardized Feature Set with Trust Mapping..... 130

Figure A-2. TabNet Binary Classification Global Feature Importance: All 39 and Core 16 Features..... 131

List of Tables

Table 2-1. Key IT and OT Operating and Security Differences 13

Table 2-2. Select Operational Technology and IIoT Security Literature 16

Table 2-3. Select Literature on IoT Machine Learning Applications and Measures 17

Table 2-4. Select Edge Computing and Federated Learning for IoT Literature 24

Table 2-5. Select Literature on Trust for IoT 30

Table 2-6a. Summary of Literature Review and Gaps (Part 1) 35

Table 2-6b. Summary of Literature Review and Gaps (Part 2) 36

Table 3-1. Feature Selection Methods 42

Table 3-2. Conventional Machine Learning Classifier Algorithms Utilized 50

Table 3-3. Deep Learning Classifier Approaches Utilized 51

Table 4-1. Selected and Candidate Datasets for Praxis Experimentation 62

Table 4-2. Praxis Dataset NF-ToN-IoT-v2 Traffic Profile 63

Table 4-3. Initial Trust-based NetFlow Feature Selection Mapping 67

Table 4-4. Features with Significant Pairwise Pearson's Correlation 68

Table 4-5. Recursive Feature Elimination with Cross-Validation 76

Table 4-6. Feature Engineering: Selected Core and Focused Trust-based Feature Sets .. 79

Table 4-7. Binary Classification Performance with All 39 Features 80

Table 4-8. Binary Classification Performance with Core 16 Features 82

Table 4-9. Binary Classification with Core 16 Features: Cross-Validation 84

Table 4-10. Binary Classification Performance with Focused 8 Features 85

Table 4-11. Multiclass Classification: Random Forest and Extra Trees (39 Features) 86

Table 4-12. Multiclass Classification: Random Forest Core and Focused Features 86

Table 4-13. Multilayer Perceptron Deep Neural Network Model Configuration..... 89

Table 4-14. MLP Binary Classification Performance: Batch Size 2048 91

Table 4-15. MLP Multiclass Classification: 39 Features..... 91

Table 4-16. MLP Multiclass Classification: Core 16 and Focused 8 Features..... 92

Table 4-17. TabNet Classifier Optimized Configuration (*opt0*)..... 94

Table 4-18. TabNet Binary Classification Performance..... 95

Table 4-19. TabNet Multiclass Classification: 39 Features..... 97

Table 4-20. TabNet Multiclass Classification: Core 16 and Focused 8 Features..... 98

Table 4-21. Federated Binary Classification with 39 Features..... 102

Table 4-22. Federated Binary Classification with Core 16 Features..... 103

Table 4-23. Federated Binary Classification with Focused 8 Features 103

Table 4-24. Federated Multiclass Classification: 39 Features..... 106

Table 4-25. Federated Multiclass Classification: Core 16 Features 107

Table 4-26. Federated Multiclass Classification: Focused 8 Features..... 107

Table 5-1. Praxis Research Hypothesis Validation Summary..... 109

Table 5-2. Impact of Dimensionality Reduction: Random Forest Classifier..... 111

Table 5-3. Deep Learning Comparative Performance 112

List of Equations

Equation 1. Chi-squared (χ_c^2) 43

Equation 2. Information Gain 44

Equation 3. Entropy 44

Equation 4. Pearson's Correlation Coefficient 44

Equation 5. ANOVA Fisher Coefficient (F-test Statistic) 45

Equation 6. Tree-based Classifier Feature Importance 46

Equation 7. Ensemble and Boosted Tree Classifier Normalized Feature Importance 46

Equation 8. Class Weight Function 49

Equation 9. Feedforward Neural Network Activation Function 52

Equation 10. Common Activation Functions: Sigmoid, ReLU, Softmax 52

Equation 11. FedAvg Aggregation Method 55

Equation 12. Precision 57

Equation 13. True Negative Rate 57

Equation 14. Detection Rate (True Positive Rate) 57

Equation 15. False Alarm Rate (False Positive Rate) 57

Equation 16. Accuracy 57

Equation 17. Balanced Accuracy 58

Equation 18. F1-score 58

Equation 19. Matthews Correlation Coefficient 58

Equation 20. Min-Max Scaling Transformation 65

List of Key Abbreviations and Acronyms

ANN	Artificial Neural Network
ANOVA	Analysis of Variance
AP	Average Precision
API	Application Programming Interface
ATT&CK	MITRE's Adversarial Tactics, Techniques, and Common Knowledge
AUC	Area Under the Curve
BERT	Bidirectional Encoder Representations from Transformers
CDFL	Cross-Device Federated Learning
CIA	Confidentiality, Integrity, and Availability
CISA	Cybersecurity and Infrastructure Security Agency
CNB	Complement Naïve Bayes
CNN	Convolutional Neural Network
CPS	Cyber-Physical Systems
DoS	Denial of Service
DDoS	Distributed Denial of Service
DGM	Deep Generative Model
DL	Deep Learning
DNN	Deep Neural Network
DR	Detection Rate
DT	Decision Tree
ET	Extremely Randomized or Extra Trees

FAR	False Alarm Rate
FedAvg	Federated Average
FL	Federated Learning
FPR	False Positive Rate
GAN	Generative Adversarial Network
GNB	Gaussian Naïve Bayes
GRU	Gated Recurrent Unit
ICS	Industrial Control Systems
IDS	Intrusion Detection System
IG	Information Gain
IoT	Internet of Things
IID	Independent and Identically Distributed
IIoT	Industrial Internet of Things
IIRA	Industrial Internet Reference Architecture
IISF	Industrial Internet Security Framework
IT	Information Technology
kNN	k-Nearest Neighbor
LGBM	Light Gradient-Boosting Machine
LSTM	Long Short-Term Memory
MCC	Matthews Correlation Coefficient
MCDM	Multi-Criteria Decision-Making
ML	Machine Learning
MLP	Multilayer Perceptron

NCS	National Cybersecurity Strategy
NIPP	National Infrastructure Protection Plan
NIST	National Institute of Standards and Technology
OT	Operational Technology
PCA	Principal Component Analysis
PLC	Programmable Logic Controller
PPD	Presidential Policy Directive
PR	Precision-Recall
QoS	Quality of Service
RF	Random Forest
ReLU	Rectified Linear Unit
RFE	Recursive Feature Elimination
RFID	Radio Frequency Identification
RNN	Recurrent Neural Network
ROC AUC	Receiver Operating Characteristic
SCADA	Supervisory Control and Data Acquisition
SHAP	SHapley Additive exPlanation
SIoT	Social Internet of Things
SVD	Singular Value Decomposition
TPR	True Positive Rate
VCE	Virtual Client Engine (within Flower federated learning framework)
XGB	Extreme Gradient Boosting
XSS	Cross Site Scripting

Glossary of Terms

Artificial Neural Network	Distributed computation model that abstracts the data presentation and processing of a human brain as an interconnected set of layers of network nodes called neurons
Cyber Resilience	Ability of a critical system and its operators to anticipate, withstand, rapidly recover from, and adapt to changing or adverse conditions, attacks, compromises, or other disruptions, regardless of the originating nature (deliberate or unintentional) of the applied threats or incidents (Adapted from Biden, 2023; *NIPP*, 2013; Ross et al., 2021)
Deep Learning	Machine learning approach using supervised or unsupervised learning techniques based on multiple layers of artificial neural networks to model deep hierarchical information relationship representations (Mohammadi et al., 2018)
Edge Computing	Architecture utilizing data computation, storage, and other information system resources between a centralized cloud and an endpoint device
Feature Engineering	Process of applying domain knowledge to extract new attributes from raw data or transform existing data attributes for improving the representation of data to support machine learning modeling and inference (Butvinik, 2021)
Feature Importance	Quantitative measure of the contribution of a specific feature toward an overall statistical property or machine learning model optimization objective
Feature Selection	Process for choosing the most relevant and least redundant set of features suited for a defined machine learning problem
Federated Learning	Distributed machine learning approach that allows participants to collaboratively train models without sharing private local data

Industrial Internet of Things	System consisting of networked smart devices, cyber-physical assets, associated information technologies, and optional cloud or edge computing platforms, which together enable real-time, intelligent, and autonomous capabilities within an industrial environment to optimize overall production value (Boyes et al., 2018)
Internet of Things	Network of connected special purpose devices that sense, actuate, and communicate data with one another and information technology systems
Multilayer Perceptron	Feed-forward artificial neural network using a non-linear activation function and multiple hidden layers between input and output layers
Operational Technology	Programmable systems and devices to bridge digital systems and the physical environment through activities ranging from passive monitoring of conditions of the operating environment or devices to active control of devices, processes, and events that have physical impact
TabNet	Generative deep learning architecture using an attentive transformer for feature selection designed for tabular data
Trust	Qualitative or quantitative property of a trustee, evaluated by a trustor as a measurable belief, in a subjective or objective manner, for a given task, in a specific context, for a specific time period (Jayasinghe et al., 2019)
Trustee	Initiator of a communication interaction or transaction
Trustor	Respondent in a communication interaction or transaction who is expected to provide information

Chapter 1—Introduction

1.1 Background

Today's connected internet has become an integral part of the modern life at home, the workplace, and factories, reaching by many estimates, over 30 billion devices globally by 2027 (Ferrag et al., 2022; Javed et al., 2023) Network traffic in industrial organizations now consists of a combination of traditional enterprise and consumer information technology (IT) and industrial application-oriented operational technology (OT). The Industrial Internet of Things (IIoT)—Internet of Things (IoT) in industrial and factory environments—is rapidly growing toward enabling $3.3 trillion in economic value by 2030 (Chui et al., 2021). All types of IoT devices surpassed conventional internet-connected devices (PCs, mobile phones, traditional information technology devices) in 2020 (Lueth, 2020). IoT and IIoT devices use specialized protocols for communication and have different usage and interaction models, primarily operating in an automated or autonomous fashion without significant user interaction. IoT are built to maximize ease of installation and use and require little maintenance, with security not always a significant design or operational consideration. IIoT devices include simple sensors and network interfaces to instruments not originally designed to support or protect from external network connectivity. IIoT can directly connect to and control physical equipment in industrial control system (ICS) or cyber-physical system (CPS) environments, including critical infrastructure and safety-critical systems such as power and public utilities plants, emergency services, healthcare infrastructure, critical manufacturing, and the defense industrial base. Compromise of IT security in critical infrastructure cases can be more severe than with enterprise IT alone, since a virtual

cyberattack may directly target and affect the physical world, such as the Colonial Pipeline attack in 2021, which impacted over 5,500 miles of petroleum pipelines (Preston et al., 2022). IoT technologies are increasingly targeted as the launching point for multiple types of additional attacks on the victim organization's critical digital and physical assets (Stellios et al., 2018, 2021). The Mirai botnet distributed denial of service (DDoS) in 2016 against a major internet service provider involved nearly 100,000 hijacked IoT devices generating a peak of 10 terabytes of data per second, and demonstrated the potential scale of an IoT-based attack (Preston et al., 2022). How these changes challenge established cybersecurity practices creates open questions for businesses, governments, and end users.

The United States codified the goal of system resilience into its National Infrastructure Protection Plan (NIPP) (*NIPP*, 2013) as its doctrinal approach to implementing the critical infrastructure security objectives of Presidential Policy Directive 21 (PPD-21, 2013). The National Cybersecurity Strategy (NCS) released in Spring 2023 builds upon this foundation, with Pillar 1 focusing on critical infrastructure defense and the need to modernize IT/OT infrastructure as a fundamental building block (Biden, 2023). The NCS describes IoT as the emerging prevalent part of the *de facto* network architecture and cites IoT-connected systems as increasingly targets of choice for cyberattacks from malicious state actors. To provide cyber resilience on a national scale the government, private industry, critical infrastructure system operators, and service providers will require effective tools beyond those available to network providers and end users today. Visibility into the dynamic behaviors of IoT devices and their

impact on network behavior is the starting point for assessing the state of health and enabling the right response to any discovered risks to overall system performance.

1.2 Research Motivation

The Mirai botnet attack spurred government and industry scrutiny on the security practices for IoT device hardware, firmware, and software and for IT and OT environments that deploy these devices. Once deployed in a networked environment, IoT presents issues across multiple operating domains and architectures. Figure 1-1 shows key operational considerations, security issues, and research areas mapped to an IoT functional reference layered architecture (ITU-T, 2012; Jayalaxmi et al., 2021; Tan & Samsudin, 2021; IIRA 1.10, 2022).

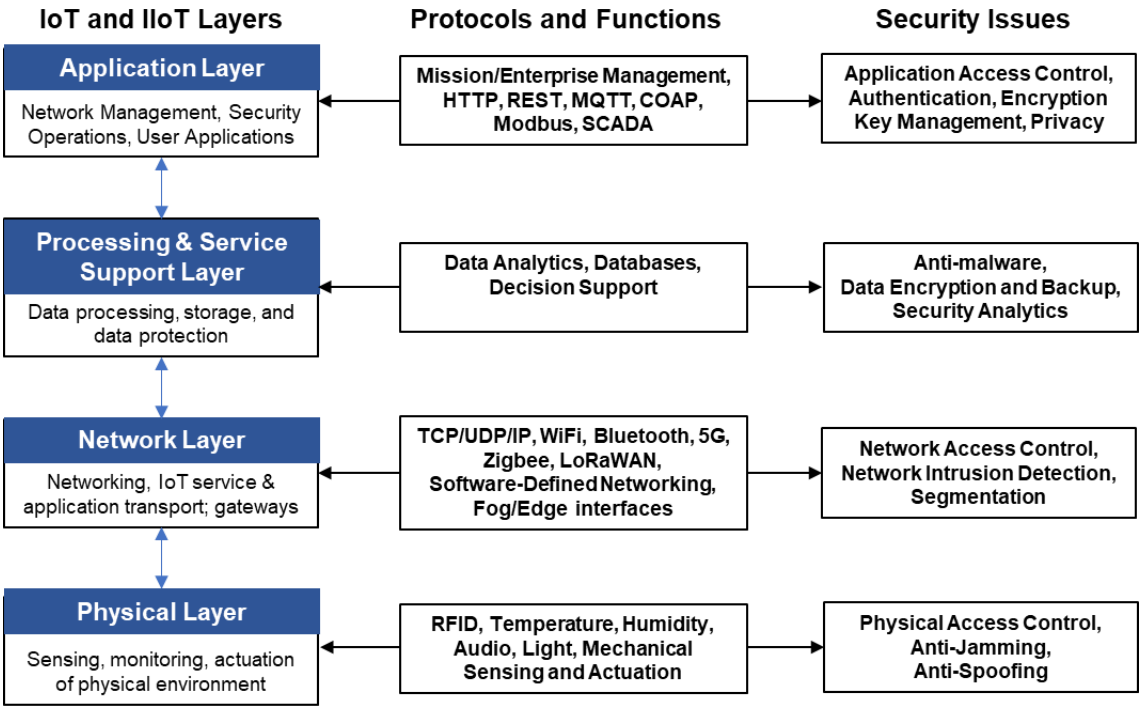


Figure 1-1. IoT Layered Reference Architecture and Representative Security

Each of the security issues identified above has spawned its own body of active research. A key gap is examining approaches for systematically scaling practical protection and countermeasure approaches for security managers of safety-critical

systems with a combined IT and OT enterprise. A promising area of research is the role of trust in not just understanding but also securing IoT-based systems. Early research on network trust focused on securing transactions among parties of different assessed levels of trust (Grandison & Sloman, 2000) or data or transactional service privacy which was most readily assessed using developed qualitative measures of social trust and reputation (Aaqib et al., 2023; Bao & Chen, 2012; Nitti et al., 2014; Yan et al., 2014). These measures must be accumulated over extended periods of time and across multiple domains of operation, or they can be provided as external ratings, such as owner or service provider reputation scores. Recent research into trust for the security of the environments in which IoT are deployed and for the devices themselves examine trust metrics computation, system-wide aggregation and evaluation, and systems management (Aaqib et al., 2023; Alhandi et al., 2023; Wei et al., 2022).

The scale of IoT device deployment and the data they generate have made machine learning (ML) a part of the standard toolkit to understanding and making decisions about candidate security or system management actions required. Recent research studies have demonstrated that ML techniques can form the basis of a significant portion of an analytical management framework that uses quantitative trust measures to support system security assessments (Alghofaili & Rassam, 2022; Jayasinghe et al., 2019; Ma et al., 2021). These approaches do not address the need to perform assessments continuously and in a timeframe required for dynamic response, nor do they account for the uneven resources across a network. Edge computing and federated learning (FL) architectures for IoT allow computational work to be distributed and then combined in a manner that protects individual data sources. These architectures account for the fact that

most current behavioral data is closest to the end devices where resources are limited and trust context is local (Ferrag et al., 2021; Gugueoth et al., 2023; Mothukuri et al., 2022; Rehman et al., 2021; Sarhan, Layeghy, Moustafa, et al., 2022) The integration and extension of these approaches provide a point of departure for the research proposed and undertaken for this praxis. Figure 1-2 shows a notional simplified workflow for the supporting course of research.

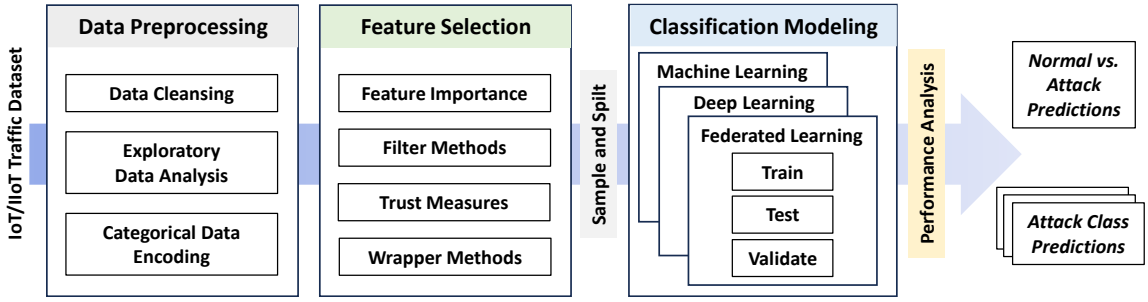


Figure 1-2. Simplified Architecture for Praxis Research Experiment

1.3 Problem Statement

The deployment of standalone IoT devices in homogenous managed network environments raises many fundamental security issues. These create a significant attack surface for not only the individual devices but any environment into which they are deployed. Many IoT and IIoT devices provide little or no direct access to or visibility into their services, configurations or operating states, which presents a black box security risk with poorly characterized behavior to the rest of the operating environment (CISA, 2020). IoT devices that connect enterprise IT with industrial OT are not designed with security controls to operate over the Internet multiply the vulnerabilities and magnify the overall attack surface of networks of heterogeneous IT/OT devices and security configurations. The following practical problem frames the motivation for this praxis:

Industrial organizations put safety-critical systems at risk by connecting enterprise IT and legacy OT with black box IoT technologies (CISA, 2020) (Boeckl et al., 2019), of which 57% are vulnerable to medium- or high-severity malware attacks (Wang & Lu, 2020).

1.4 Thesis Statement

The distributed span and tiered architecture of large IT/OT environments presents a challenge for timely monitoring, evaluating, and acting on risk measures using a conventional centralized approach. A fully distributed model would allow dynamic assessment of behavior at the source of the device under evaluation. This, however, would require computational, storage, communications, and energy capacity not found in resource-constrained IoT devices to continuously perform the evaluation and provide synchronized updates to a central controller. This praxis posits the following thesis to guide exploration of an alternate approach:

Federated machine learning threat detection and classification at the network edge using trust-based measures of IoT entity behavior improves cyber resilience in a distributed IT/OT environment.

1.5 Research Objectives

This research has the following as its primary research objectives:

- Identify network traffic attributes that measure IoT device trustworthiness in a heterogeneous IT/OT environment.
- Determine the performance of trust-based machine learning models for detecting and classifying malicious IoT device behavior at the network edge.

- Compare the efficacy of conventional, deep, and federated machine learning for detecting and classifying malicious IoT behavior at the network edge.
- Propose a system architecture for incorporating federated edge machine learning for trust-based IoT risk detection and classification into an IT/OT cybersecurity framework.

1.6 Research Questions and Hypotheses

This praxis addresses the following research questions (RQs) and poses the following research hypotheses:

RQ1: What IT/OT network traffic attributes measure trustworthy IoT behavior at the network edge?

RQ2: Using trust-based metrics, do deep learning models provide performance advantages over conventional machine learning models for the detection and classification of IoT cyberattacks in IT/OT networks?

RQ3: Can a federated learning model deployed to the network edge detect and classify IoT cyberattacks using trust-based measures?

H1: Quality of service attributes can provide trust-based metrics for feature selection to detect and classify IoT/IIoT cyberattacks.

H2: Centralized machine learning classifiers using trust-based metrics can reduce dimensionality by at least 50% while detecting and classifying IoT/IIoT cyberattacks within 1% of F1-score performance using standard NetFlow features.

H3: Multilayer perceptron deep neural networks using trust-based features achieve F1-score detection and classification performance of IoT/IIoT cyberattacks within 5% of tree-based machine learning classifiers.

H4: TabNet generative deep learning models using trust-based features attain better F1-score detection and classification of IoT/IIoT cyberattacks over multilayer perceptron deep neural networks.

H5: Edge federated learning can attain IoT/IIoT cyberattack F1-score classification performance of at least 85%.

1.7 Scope of Research

This praxis explores improving the cyber resilience of safety-critical systems through the application of standardized network traffic measures in both centralized and federated machine learning architectures to determine the trustworthiness of the system's IoT devices. This study develops an analytical framework to determine measures of trust from the observable network behavior of devices and uses these measures to formulate a cyberattack detection and classification assessment. A framework of binary and multi-classification models is applied to a public dataset consisting of both benign and malicious IoT and IIoT device traffic representative of the heterogeneous IT/OT environment of a safety-critical system. The NF-TON-IoT-v2 dataset (Sarhan, Layeghy, & Portmann, 2022c) provides a labeled NetFlow-based feature set of 16,940,496 records central to this study. The performance of ML and DL classification models in centralized and federated architectures is compared using statistical classification metrics, with a focus on the F1-measure to provide consistency across imbalanced datasets. Training and prediction times are measured to determine network edge deployment suitability.

1.8 Research Limitations

An effective federated edge deployment of an IoT/IIoT security model would provide a balance between a fully centralized architecture with significant resources but

slow reaction to environmental changes and a fully distributed architecture that measures changes dynamically but has limited resources to respond. This praxis uses system architectures, mathematical models, and data science software tools to investigate research questions and answer research hypotheses about the feasibility and analytical performance of a trust-based threat detection and classification model for IT/OT networks. The study investigates the efficacy of the security model for specific ML and DL algorithms comprising a class of federated learning at the network edge. Protocols for model updates to the edge are not studied, as standards are still under development for both the IoT community at large and for safety-critical industries. This praxis does not assess the real-time measured performance or resource utilization of threat classification in a live operating environment.

1.9 Organization of Praxis

The praxis is organized into five chapters. Chapter 1 presents the background information necessary to understand the research topic, motivation, problem and thesis statements, scope, questions, hypotheses, and limitations. Chapter 2 provides a review of relevant security research literature covering operational technology and IIoT, trust architectures and computation, centralized and federated machine learning for IoT applications, and cyber resilience for safety-critical systems. Chapter 3 describes the data sources, research and experimentation methodologies, and analytical techniques used in the praxis. Chapter 4 reviews the research experiments' findings and implications for improving cyber resilience. Chapter 5 summarizes the findings of the research in terms of the research questions and hypotheses, contributions to the body of knowledge, improvements to current practice, and recommendations for future research.

Chapter 2—Literature Review

2.1 Introduction

This praxis extends research in several core security disciplines, applied through the lens of a security practitioner seeking to utilize current machine learning techniques to improve overall cyber resilience of safety-critical systems deploying Internet of Things (IoT) technologies. Understanding what comprises and informs the cyber resilience of safety-critical systems is fundamental to the course of research for this praxis. To provide guidance on most relevant areas of focus, this praxis adopts the following definition of system cyber resilience derived from NIPP 2013 and PPD-21 (Biden, 2023; *NIPP*, 2013) and applied to cybersecurity through NIST SP 800-160 (Ross et al., 2021):

> *Cyber resilience is the ability of a critical system and its operators to anticipate, withstand, rapidly recover from, and adapt to changing or adverse conditions, attacks, compromises, or other disruptions, regardless of the originating nature (deliberate or unintentional) of the applied threats or incidents.*

The key research topics that form the foundational building blocks for this praxis are as follows and comprise the subsequent sections of this chapter's literature review:

- §2.2: Operational Technology and Industrial Internet of Things Security
- §2.3: Machine Learning for IoT Security
- §2.4: Edge Computing and Federated Machine Learning for IoT
- §2.5: Trust Frameworks, Architectures, and Computation for IoT.

The last section provides a summary and analysis of the key research literature, discusses open research issues, and highlights the contributions of this praxis to the existing body of knowledge and practice.

2.2 Operational Technology and Industrial Internet of Things Security

Strategies for Industry 4.0 lay out a vision initially for smart manufacturing and then more broadly for digital industrial ecosystems enabled by the convergence of operational technology (OT) and information technology (IT), IoT, cloud computing, data analytics and life-cycle digital business processes (Xu et al., 2018). As industries and governments have mobilized to realize Industry 4.0 through large scale digital transformation initiatives, the interconnectedness of heterogeneous devices, systems, technologies, data, and business processes requiring complex integration has revealed a myriad of operational and security challenges. In its September 2023 Revision 3 update to Special Publication 800-82, NIST expanded the scope of its guidance from industrial control systems (ICS) security to OT security, recognizing the impact and importance of this growing set of capabilities (Stouffer et al., 2023). OT utilizes programmable systems and devices to bridge digital systems and the physical environment through activities ranging from passive monitoring of conditions in the operating environment or devices to active control of devices, processes, and events that have physical impact. Common OT systems include supervisory control and data acquisition (SCADA), programmable logic controllers (PLCs), cyber-physical systems (CPS), and Industrial IoT (IIoT).

Boyes et al. define IIoT as a system consisting of networked smart devices, cyber-physical assets, associated information technologies, and optional cloud or edge computing platforms, which together enable real-time, intelligent, and autonomous capabilities within an industrial environment to optimize overall production value (Boyes et al., 2018). These capabilities include functionality for providing access, collection, analysis, communications, and exchange of process, product, or service information.

Boyes et al. provide an ontological structure for systematically describing the attributes of an IIoT element, which can be used to characterize its functionality or security issues.

Converged IT/OT and IIoT are increasingly infiltrating once-isolated sensitive safety-critical systems such as the US nuclear infrastructure. A 2022 report from the Office of International Nuclear Security, National Nuclear Security Administration, Department of Energy (DOE) outlines how IIoT devices enhance the use of industrial SCADA systems by integrating automated sensor monitoring and analytics to influence the decision-making process, while pointing out that the deployment of IIoT introduces not-well-understood new risks into the nation's nuclear infrastructure (Preston et al., 2022). Preston et al. advocate adopting a model lifecycle framework such as the DOE Cyber-Informed Engineering strategy to integrate IIoT risk management into security and mission standard operating procedures. This includes resilient system architecture, active defense, digital asset inventory, supply chain assurance, incident response, and training.

The integration of IT and OT that is required to develop the end-to-end resilient digital infrastructures for Industry 4.0 exposes the critical need for novel approaches for verification and validation testing each domain, the constituent foundational capabilities, and the integrated whole. Table 2-1 highlights key differences in the design, operation, and maintenance of IT and OT. Serror et al. highlight the security implications of these OT attributes that require changes to existing enterprise IT security practices. Driving considerations for IIoT deployed in OT environments include (a) long-lived platforms with little or no mechanisms for regular security updates; (b) large number of deployed heterogeneous devices, with inconsistent configuration and maintenance regimes; (c) high interconnectivity, which challenges best practices for segmentation of critical

systems; (d) risk of trading off security for low latency response and high availability to meet mission requirements; (e) privacy and data confidentiality of mission-critical processes and metadata; and (f) human error and sabotage (Serror et al., 2021).

Serror et al. further analyze the application of enterprise IT and consumer IoT security practices to IIoT in OT settings, identifying promising research in each of five areas. These include (a) tailored lightweight cryptography and authentication for

Table 2-1. Key IT and OT Operating and Security Differences

Attribute	Information Technology	Operational Technology
Service Response	Human-interaction-centered, non-real-time, high throughput, latency and jitter tolerated	Machine-interaction-centered, real-time and time-critical, throughput moderate, latency and jitter unacceptable
Availability	Wide range acceptable	High availability, resilient
Access Control	Variable access control per security as needed	Strict access control, but response to human and emergency response critical
Technology Foundation	Commercial operating systems, automated IT tools, 3-5 year refresh cycle, multi-vendor, enterprise or consumer setting	Proprietary operating systems, custom scheduled maintenance, 10-30 year expected system lifetime, limited or single vendor, remote or specialized industrial setting
On-Device Resources	Sufficient to support local applications, security additions	Purpose-built with limited memory, storage, compute for other apps, security
Networking	Widely available standard protocols and interfaces	Standard, industry-specific, proprietary protocols, dedicated wired or wireless
Data	Medium volume, privacy-oriented, event-driven, unstructured	High volume, mission security-oriented, periodic, structured
Security Focus	Data and digital asset confidentiality, integrity, business continuity	Human safety, equipment and facilities protection, fault tolerance, strict operations resilience

Adapted from Stouffer, 2023; Serror et al., 2021; *IISF 2.0*, 2023

resource-constrained IIoT devices executing mission-critical processes; (b) automated patch management and vulnerability testing that does not impact industrial processes; (c) trusted service isolation, access control, and software-defined networking policy enforcement; (d) industrial network-based monitoring and process-aware intrusion detection systems (IDS); and (e) tools and processes for industrial cyber situational awareness and security training (Serror et al., 2021).

The attack vectors, IIoT protocols, and target platforms addressed, along with the detection or counter-measure architectures and techniques are the distinguishing factors of OT and IIoT security research to date. Tan & Samsudin compare three, four, five, and seven-layer commonly used IIoT implementation architectures, and then propose a new four-layer architecture specifically aimed at the security of IIoT (Tan & Samsudin, 2021). Their approach underscores the diversity of over thirty physical and data link wireless protocols used by IIoT, depending on desired range and available bandwidth, as well as the diversity of application level protocols—some adaptations of common web standards (e.g., SOAP, REST, HTTP), while others are specific to IoT (e.g., CoAP, MTTQ, AQMP). Using the extended CIA triad of confidentiality, integrity, availability, authentication, authorization and access control, Tan & Samsudin analyze security risks and threats, countermeasures, ongoing challenges, and future research opportunities for each of the major components of their four-tier security architecture. Tan & Samsudin only minimally examine TCP/IP network flow level techniques, which are most often a point of commonality in a heterogeneous IT/OT IIoT environment.

In a comprehensive review, Yaacoub et al. present a taxonomy for cyber-physical attacks, mapping target vulnerabilities to impact; an extensive qualitative risk assessment

of cyber-physical attacks; and the strengths and weaknesses of potential security countermeasures (Yaacoub et al., 2020). They include business implementation factors, physical breaches and countermeasures, and cyber breaches and countermeasures in their analysis and risk model. The three CPS architecture layers—perception, transmission, and application—each map to specific vulnerabilities and safeguard solutions. Notably, Yaacoub et al.'s analysis of IDS architectures emphasizes the flexibility of hybrid IDS architecture, which integrate concepts of both centralized and distributed placement within an IoT network to support organizing devices being monitored into clusters or deploying small-footprint watchdog nodes. These approaches allow system operators to balance resource consumption, communication overhead, and detection coverage.

Tsiknas et al. maps research on IIoT attacks, threats, and countermeasures to multiple IIoT reference architectures, including the three-tier edge-platform-enterprise architecture specified by the Industrial IoT consortium (IIRA 1.10, 2022; Tsiknas et al., 2021). MITRE has codified a tailored version of their well-known Adversarial Tactics, Techniques, and Common Knowledge (ATT&CK) knowledge base for cyber attacker behavior against ICS (*MITRE ATT&CK® Matrix for ICS*, 2023). Cyberattacks on OT often target direct impact on the physical systems they monitor or control. Detection of and response to anomalous OT events and behavior present a significant measure to improve the cyber resilience of an industrial system in operation. Zhou et al. demonstrate the value of a multimodel-based anomaly intrusion detection system for process control systems, integrating event data from communication state, schedule, and protocol models; node software task and resource models; and control and process application data flow models (Zhou et al., 2015). Zhou et al. propose a Hidden Markov Model statistical

process control approach to classifying anomaly detection alerts as normal, fault, or attack state. This model performs well in a control system simulation, detecting and distinguishing attacks within one process macrocycle with the false detection rate below 1.61%. Table 2-2 summaries the key concepts discussed in this section.

Table 2-2. Select Operational Technology and IIoT Security Literature

Key Concept	Applicability	Reference
IT/OT Convergence	Industrial digital transformation is exposing functional and operating differences with security implications	LD Xu et al., 2018; Stouffer et al., 2023; Boyes et al., 2018; Serror et al., 2021
Critical Systems Resilience	IT/OT and IIoT deployed in critical infrastructure impact resilience	Preston et al., 2022; Serror et al., 2021; IISF 2.0, 2023
Layered Reference Architectures	Systematic analysis of OT and IIoT, security risks and countermeasures	Tam & Samsudin, 2021; Yaacoub et al., 2020; Tsiknas et al., 2022; IIRA 1.10, 2022; MITRE ICS, 2023
Hybrid Security Architecture	Hybrid placement (centralized vs. distributed) and multiple models improve flexibility and performance	Yaacoub et al., 2020; Zhou et al., 2015

2.3 Machine Learning Applications and Measures for IoT Security

Machine learning for IoT security applications continues to attract significant research attention, with over 27,000 papers published since 2016 (Javed et al., 2023). In his well-known textbook, Flach points out “machine learning is concerned with using the right features to build the right models that achieve the right tasks” and “models lend the machine learning field diversity, but tasks and features give it unity” (Flach, 2012, pp. 12, 13). This provides guidance for selecting the papers most important to informing topics vital to the course of research for this praxis, which are summarized in Table 2-3.

Table 2-3. Select Literature on IoT Machine Learning Applications and Measures

Key Concept	Applicability	Reference
Performance Metrics	Common metrics are not always suitable for comparing classification results, esp. for imbalanced IoT security datasets	Grandini et al., 2020; Zolanvari et al., 2021
Feature Engineering	Domain/problem-informed feature selection reduces ML model complexity, improves performance, and provides explainability of results	Zolanvari et al., 2021; Sarhan et al., 2022; Alani & Miri, 2022; Alani et al., 2022
Host-based Intrusion Detection	Lightweight ML running on IoT endpoint hosts is a viable alternative to network IDS	Wang & Lu, 2020
Deep Learning	Hierarchical layered learning models for complex system representations	Mohammadi et al., 2018; Al-Garadi et al., 2020; De et al., 2022
Generative Models	Deep generative models combine flexibility of DL with probabilistic inference modeling	De et al., 2022; Yu et al., 2021

Researchers at Washington University in St. Louis present a systematic categorization of prevalent attacks on IIoT systems, create a risk assessment matrix, summarize intrusion detection system (IDS) approaches specifically for SCADA systems found in industrial settings, and construct an IDS testbed for SCADA systems (Zolanvari et al., 2019). Examining backdoor, command injection, and SQL injection attacks against a SCADA system, Zolanvari et al. demonstrate the importance of combining knowledge of normal system behavior and the underlying attack behavior problem in selecting and extracting features for training ML models. Specifically, they select common network flow-based attributes that vary during attack phases and then analyze the statistical importance of these features to an ML model's ability distinguish between normal and attack traffic (Zolanvari et al., 2019). Other research efforts that are less diligent in applying problem-specific insights to feature engineering risked overfitting

solutions to test datasets or otherwise showing results that are difficult to generalize. Zolanvari et al. compare the performance of logistic regression, naïve Bayes, support vector machine, decision tree, random forest, k-nearest neighbor, and artificial neural network ML models. Notably, they compare classification performance using multiple metrics: accuracy, false alarm rate (FAR), undetected rate, Matthews Correlation Coefficient (MCC), sensitivity, or true positive rate (TPR), and receiver operating characteristic area under the curve (ROC AUC).

Not all researchers, however, consistently demonstrate the same diligent awareness as Zolanvari et al. of the lack of suitability of the metrics they use when evaluating the application of ML to their research results. Grandini et al.'s monograph on metrics for multi-class classification extends the perspective on the significance and suitability of specific metrics used to evaluate the comparative performance of ML tasks (Grandini et al., 2020). Notably, Grandini et al. show accuracy is most appropriate when the problem studied is the prediction of the highest number of individual members in the right class, regardless of class distribution. Since highly populated classes will be higher weighted compared to less populated classes, accuracy will hide strong classification errors for small classes. This may be pronounced in imbalanced datasets, such as those studied for IoT security, where anomalies and attacks are far less prevalent than normal traffic. Balanced accuracy, which gives the same weight to each class is insensitive to an imbalanced class distribution. Grandini et al. also show that F1-score gives large weight to smaller classes, and the Matthews Correlation Coefficient (MCC) provides a good measure of ML performance for imbalanced datasets. Chapter 3 reviews the performance measures selected for evaluating the results of this praxis' ML experiments.

Motivated by a desire to improve the ability to compare the performance of ML-based network traffic classifiers, Sarhan, Layeghy & Portmann illustrate the effectiveness of extending Zolanvari et al.'s feature selection and engineering approach into the creation of a standard feature set for IDS datasets (Sarhan, Layeghy, & Portmann, 2022c). They refactor four widely used public IDS datasets (two with significant IoT/IIoT traffic), replacing the custom features unique to each individual dataset with a common set of 43 NetFlow-based features. Binary classification (attack or benign) and multiclass classification (specific attack families) results against the new datasets using an ensemble Extra Trees ML model yielded improved performance (using F1-score metric) over their respective original datasets. Unlike many of the custom features of the original datasets, the NetFlow-based features are directly measurable in the traffic flow and map directly to security events of interest. Using a standard feature set for IDS datasets not only supports comparative evaluation of ML-based IDS investigations across experiments, but also across multiple datasets, operating environments, and attack scenarios. Many applications can share, readily combine, and use these labeled datasets. Other research investigations have corroborated the utility and efficacy of this NetFlow standard feature approach for IoT IDS in greater detail (Awad et al., 2022) and for additional applications, such as using collected network quality of service flow data to construct trust properties for ML-based IoT trust evaluation (Ma et al., 2021).

Research by Alani et al. demonstrates that purposeful feature selection and engineering improve classification performance and explainability (Alani et al., 2022; Alani & Miri, 2022). For multiple public IoT/IIoT datasets, Alani et al. apply recursive feature elimination (RFE) to select the most important features to use in random forest

model training, defined by average Gini impurity decrease computed from all decision trees in the forest. With the resulting 6 features, reduced from an original 37, the authors show strong classification performance (over 99.6% composite accuracy) with significantly reduced training times for each of four different ML models, across three IoT datasets. Alani & Miri use the SHapley Additive exPlanation (SHAP) method, which calculates the quantitative impact of each feature by comparing the ML model results with and without the feature, to quantify the impact of the selected 6 features.

Wang & Lu propose stacking a boosted version of a classical ML model with a DL model in a resource-conscious lightweight host-based intrusion detection system architecture deployed at the IoT endpoint, rather than as a centralized network IDS (Wang & Lu, 2020). They address the premise that (a) under nominal operating conditions, an IoT device exhibits repetitive, statistically predicable behavior, and (b) the compromise of an IoT device from an attack would be manifested through anomalous behavior or communication visible directly on the device. Their design uses system call monitoring data as inputs to an anomaly detection engine using the extreme gradient boosting (XGBoost) ML model stacked with the Long Short-Term Memory (LSTM) recurrent neural network DL model. Wang & Lu's experiment shows strong anomaly detection performance (ROC AUC 98.3%, F1-score 97.9%), demonstrating the effectiveness of the stacked ML/DL architecture. Wang & Lu, however, curiously present few measurements of resource utilization, execution speed, and architecture footprint that would model a practical design for a resource-constrained host.

Deep learning (DL) techniques hold promise for IoT/IIoT for many of the same reasons they have been successful for image, video, and text processing: ability to

automatically extract attributes (features) from large datasets and construct complex data representations (Al-Garadi et al., 2020). DL methods are based on artificial neural network (ANN) architectures, which are distributed computation models that abstract the data presentation and processing of a human brain as an interconnected set of layers of network nodes known as neurons or units. DL techniques and the powerful hardware platforms typically used to run them can be well-suited for IoT/IIoT applications. DL uses supervised or unsupervised learning techniques based on multiple layers of artificial neural networks to model deep hierarchical information relationship representations (Mohammadi et al., 2018). Compared to conventional ML models, DL models effectively utilize the large volume of data produced by IoT/IIoT networks; excel at generalizing complicated relationships, such as high-order spatial and temporal dependencies; and automatically extract features without prescriptive labor-intensive feature specification (De et al., 2022). To date DL is particularly promising for certain applications where conventional ML falls short, including perception (physical) layer authentication and access control and anti-radio frequency wireless jamming; large-scale multi-stage anomaly-based intrusion detection systems, especially with incomplete or changing labeled data; complex application layer malware detection and adaptive policy-based security enforcement (Al-Garadi et al., 2020; Mohammadi et al., 2018). Mohammadi et al. provide detailed analyses of the architectural and implementation complexity of many classes of DL models for a broad range of system optimization and security applications, characterizing the type of deep neural network (DNN), depth of and size of layers, and training and test times. These are used as points of departure for comparing methods of adapting DL for resource-constrained IoT platforms that include DNN compression,

approximate computing, hardware acceleration, and Tinymotes (very small processors). Mohammadi et al. examine how DL services, models, and tasks might be distributed across a fog-cloud architecture, which presages research on edge-based DL.

Advanced generative AI/ML models hold promise for addressing some of the challenges of applying conventional DL to IoT/IIoT security problems. Generative models provide an approximation of the distribution of the data by estimating their conditional probability densities. IIoT security problems in particular typically have the following characteristics: (a) imbalanced datasets, where the very small number and short duration of fault condition samples biases predictions and results in poor performance; (b) limited labeled data, exacerbated by the diversity of industrial equipment and operating environments; (c) domain adaptation, where the different input sensor and output labels for diverse IIoT equipment complicate DL transfer learning; (d) large attack surface, due to OT factors highlighted in Table 2-1, including resource constraints, limited designed-in network connectivity and security, and poor visibility into behavior; and (e) low-latency communications, with tight operational tolerance for quality of service and synchronization (De et al., 2022). De et al. show that the combined strengths of deep learning (especially flexible feature extraction and relationship representation), inference power of probabilistic models, and unsupervised data generation that come together in deep generative models (DGMs) allow them to overcome the shortfalls of conventional DL. Using three major types of DGMs—autoregressive (AR) models, variational autoencoders (VAEs), generative adversarial networks (GANs)—De et al. illustrate the effectiveness of applying DGMs to IIoT-specific anomaly detection, trust boundary protection, traffic prediction, and platform monitoring and optimization. Yu et

al. demonstrate the effectiveness of applying the class of generative models known as transformers (specifically, Bidirectional Encoder Representations from Transformers (BERT)) in combination with DL (Long Short-Term Memory (LSTM)) to the detection of certain classes of advanced persistent threat (APT) against IIoT (Yu et al., 2021).

De et al. conclude with an analysis of challenges and open issues for further research: (a) limited expressive power, where a DGM's ability to approximate functions and distributions suitable for an application is limited by the quality and consistency of available IIoT data; (b) weak discriminative capability, DGM models may not generate sufficiently representative data and be subject to model collapse for complex probabilistic models; (c) insufficient interpretability, such that latent vectors used for generation are difficult to interpret; (d) lack of generalization ability, where without continual learning, a DGM cannot generate data representative of the training set's distribution; (e) generated data vulnerability, where DGM generated data may be readily distinguished by ML from real collected data; (f) privacy preservation for generated data; and (g) data complexity, pointing to the opportunity for time-efficient and resource-light DGM implementations.

2.4 Edge Computing and Federated Learning

IoT deployment has been one of the drivers of the emergence of edge computing architecture that provides an attractive balance of the availability of data computation, storage, and resources between a centralized cloud and an endpoint device. Early in its development, IoT was identified as a significant driver of several use cases for edge computing ranging from cloud offloading, video analytics, smart home, smart city, and collaborative applications such as connected health (Shi et al., 2016). Shi et al. point to shortfalls of edge computing suggesting opportunities for new architectural approaches:

(a) programmability, where computing could be software-defined along the data path; (b) naming, where edge devices can use dynamic and flexible networking addressing; (c) data abstraction, providing consistent public data representation and operation interfaces across heterogeneous edge devices; (d) service management, providing differentiation, extensibility, isolation, and reliability; (e) privacy and security, protecting private information at the edge before sharing for processing, and (f) optimization metrics, identifying key performance and resource considerations for completing edge workloads, such as latency, bandwidth, energy, and cost. Table 2-4 summarizes the key concepts most relevant to this praxis.

Researchers at Google debuted the idea of federated learning (FL) as a massively distributed architecture, addressing many of the opportunity areas suggested by Shi et al.

Table 2-4. Select Edge Computing and Federated Learning for IoT Literature

Key Concept	Applicability	Reference
Edge Computing	Processing at network edge for IoT scale	Shi et al., 2016
Federated Learning	Privacy-preserving distributed learning	McMahan et al., 2017; AbdulRahman et al., 2021 Mothukuri et al., 2021
Trusted Edge FL	Trustworthiness at mobile edge; IoT edge FL + blockchain for trust	AbdulRahman et al., 2021 Fotia et al, 2023; Rehman et al., 2021
FL for IoT Security	Federated architecture for IoT intrusion and anomaly detection; Centralized, federated, vs local ML & DL; Aggregation algorithm and feature selection impacts to performance	AbdulRahman et al., 2020 Ferrag et al., 2021; Sarhan et al., 2022; Mothukuri et al., 2022; Lazzarini et al., 2023
Architecture-driven Performance	Balance among objectives and performance measures for specific applications, given available resources	Khan et al., 2021; Gugueoth et al., 2023; Yaacoub et al., 2023

FL provides a fundamental change to applications of conventional machine learning where data and processing resources are not practically or efficiently available centrally (McMahan et al., 2017). McMahan et al. frame the goal of a FL implementation as a federated optimization problem with four key properties: (1) not independent and identically distributed (non-IID) data—training data at a local client is not representative of the population distribution; (2) dataset will be unbalanced, reflecting significant variation per user; (3) clients will be massively distributed; and (4) endpoint clients will have limited, slow, expensive, and/or intermittent communication. These characteristics fit the profile of common cybersecurity malware and anomaly detection datasets.

The model training process for FL does not require direct access to raw training data, such as the individual endpoint IoT device data studied for this praxis. Applying the overall training process to the data available from each participating device (termed a client by McMahan et al.) minimizes the exposure of the client's private data and reduces the attack surface to the client and overall system security risk. Model learning can continue when clients go offline, perform local learning slowly, or communicate and accept updates intermittently. Unlike prior distributed learning models that applied on-device or on-site learning in isolation at each node, FL allows clients to learn from each other through aggregation. McMahan et al. introduce FederatedAverage (FedAvg), based on stochastic gradient descent, as the baseline algorithm for synchronizing or aggregating local client and global central coordinator learning models.

Mobile systems provided a key driver for applying FL to the network edge (AbdulRahman et al., 2021), demonstrating its scalability and exposing critical security challenges, which include assessing the trustworthiness of transactions. Fotia et al.

examine trust-based security, identifying candidate frameworks and performance for trust and reputation management in large-scale IoT environments. These provide assured data privacy and trust in a resource efficient and reliable manner (Fotia et al., 2023). They advocate scaling trust through a distributed edge architecture that relies on blockchain to provide a secure, reliable, decentralized infrastructure for trust computation and value.

Rehman et al. propose the TrustFed framework to detect machine learning model poisoning attacks, provide an environment for fair training, and maintain device trust reputation for cross-device federated learning (CDFL) systems (Rehman et al., 2021). The TrustFed framework uses the Ethereum blockchain to provide an on-chain reputation smart contract, an aggregator smart contract, and a smart contract incentive mechanism. For an IIoT predictive maintenance test case, the authors show a TrustFed enhanced CDFL model converges and improves over 100 rounds and that performance improves by removing outliers from subsequent training rounds. This shows the functional feasibility of calculating reputation trust and detecting anomalous data in a distributed manner using a blockchain to manage reputation trust scores.

Mothukuri et al. provide a delineation of FL approaches according to network topology (centralized and clustered, or fully-decentralized), data availability (cross-silo, or cross-device), data partition (horizontal learning, vertical learning, or transfer learning), aggregating/optimizing algorithms (aggregation algorithm, or communication cost/efficiency optimization), and software implementation frameworks (Mothukuri et al., 2021). Mothukuri et al. highlight the use of hybrid architectures to optimize different implementation issues tuned for the strengths and weaknesses of application-specific learning models. They also provide a comparative review of fourteen aggregation and

communication efficiency algorithms, which address preserving privacy of local data and model updates, optimizing communication bandwidth, supporting asynchronous model updates, or handling different statistical data distributions.

Recent research shows FL can be effective for IoT intrusion and malware detection, performing at levels comparable to conventional ML or DL but with the architectural advantages of FL (AbdulRahman et al., 2020; Ferrag et al., 2021; Mothukuri et al., 2022). AbdulRahman et al. use the well-known NSL-KDD dataset to compare the performance of centralized ML, on-device local endpoint ML, and FL for network IDS. They find FL performs better than on-device learning and performs similarly to centralized ML (AbdulRahman et al., 2020).

Ferrag et al. review the application of multiple FL architectures for IoT security applications against a cross-section of threat models, target network device types, and datasets. They present focused analyses of the application of FL to IoT intrusion and malware detection and IoT blockchain-based data sharing and storage (Ferrag et al., 2021). Their cataloging of vulnerabilities in FL-based IoT systems is especially noteworthy, describing several classes of threat models, adverse impacts, and potential solutions. Threat models include adversarial, differential, poisoning, privacy leakage, jamming, Byzantine, shilling, black box, gray box, and inference attacks. Ferrag et al. present the results of their own experiments applying FL architectures to IoT IDS using three different DL models: deep neural network (DNN), convolutional neural network (CNN), and recurrent neural network (RNN). They demonstrate against three public IoT/IIoT IDS datasets (TON_IoT, Bot-IoT, and MQTTset) that federated DL outperforms classical centralized ML in IoT/IIoT as measured by accuracy and ROC AUC while

preserving local data privacy. Notably, after fifty rounds of federated DL, the performance of all global models approaches the performance of the centralized model.

Mothukuri et al. study the performance of decentralized FL implementations of two other DL models—Long Short Term Memory (LSTM) and Gated Recurrent Units (GRU)—for IoT intrusion detection of the Modbus public dataset (Mothukuri et al., 2022). The Modbus ICS protocol dataset contains attacks that follow the patterns of the MITRE ATT&CK for ICS model. They demonstrate high accuracy (average 90.26%) and low false alarms compared to the centralized architecture implementations of the same DL models (average accuracy 86.13%), while illustrating the impact of adjusting hyperparameters such as window size for the global ML models.

Performing comparative evaluations in the Flower FL Python environment, Lazzarini et al. examine the results of binary and multi-class classification of IoT intrusion detection using a shallow artificial neural network (ANN) with a 24-neuron input layer, a 16-neuron hidden layer, and an output layer (Lazzarini et al., 2023). The researchers show the performance of the FL implementation is comparable to the centralized implementation of the same ANN against the same public IDS datasets—one with IoT/IIoT data (TON_IoT) and one with enterprise IT data (CIC-IDS2017).

Sarhan, Layeghy, Moustafa, et al. examine cyber threat intelligence sharing across organizations as a motivating use case for collaborative federated learning. For centralized, localized, and federated implementations of DNN and LSTM models with three hidden layers, the authors compare transparency of design, ease of training, and performance of machine learning classification for IoT network intrusion detection (Sarhan, Layeghy, Moustafa, et al., 2022). Using public datasets NF-UNSW-NB15-v2

and NF-BoT-IoT-v2, they show that FL with ten learning rounds achieves levels just below the centralized learning results for binary classification and comparable or better performance for multiclass classification. Most notably, they further demonstrate that using a common feature set (based on NetFlow v9) allows two different organizations to participate in federated learning without each sharing the private data of their respective endpoints. The two organizations deploy the aggregated models on their respective networks, representing shared cyber threat intelligence. These experiments do show that the computation time for the FL approaches can be significantly longer, which suggests this is application-dependent and an area for further research.

Architectural choices for FL allow a trade-off among performance measures for specific applications, constrained by available resources. Key considerations are: (a) client privacy protection and trusted verification of and secure exchange of model updates; (b) scalability to incorporate more users to improve learning accuracy; (c) quantization of local learning model updates to support faster FL convergence; (d) learning robustness with edge or cloud aggregation server issues; and (e) sparsification, selecting the most suitable clients for a specific task for faster FL convergence and better accuracy (Gugueoth et al., 2023; Khan et al., 2021; Yaacoub et al., 2023).

2.5 Trust Frameworks, Architectures, and Computation

"Permeation of trust in the IIoT system lifecycle" is foundational building block of the IIoT Consortium's Industrial Internet Security Framework (IISF, 2023, p. 42). Grandison & Sloman of Imperial College established the cornerstone system concepts for trust in Internet applications, defining trust as "the firm belief in the competence of an entity to act dependably, securely, and reliably within a specified context" (Grandison &

Sloman, 2000, p. 3). Given the context of Internet services at the time of their paper in 2000, Grandison & Sloman were primarily concerned about trust between trustor (service user) and trustee (service provider) parties involved in a transaction and emphasized that trust can be measured and is not necessarily symmetric. To work at scale, an enterprise must have a trust management system that collects the information from participants to continually monitor and evaluate trust relationships. Timely and accurate information about the behavior of the devices on a network informs the trustworthiness of the devices and the security risk levels of the infrastructure needed to achieve system resilience. Significant research over the last decade has focused on developing scientific definitions of trustworthiness as a key building block of system operations resilience.

Table 2-5. Select Literature on Trust for IoT

Key Concept	Applicability	Reference
Networked Trust	Trustworthiness in context of a service interaction and overall behavior	Grandison & Sloman, 2000 *IISF*, 2023
Social Trust	Trust derived from P2P interaction	Nitti et al., 2012, 2014
Trust Computation	Quantitative trust measures computable on node behavior	Jayasinghe et al., 2014;
Trust Aggregation	Trust computable at edge contribute to assessing system-wide interaction trust	Alghofaili & Rassam, 2022
Data Trust	Trust for data quality assessment	Byabazaire et al., 2020
Trust Management Systems	Scalable frameworks for measuring, computing, aggregating, evaluating, modeling, and managing IoT trust	Alhandi et al., 2023; Aaqib et al., 2023; Yan et al., 2014

Early research on trust mechanisms for IoT focused on the integration of subjective peer-to-peer ratings, reputation scores, and then social networking constructs, instantiated as the Social Internet of Things (SIoT) (Nitti et al., 2012, 2014). SIoT depends on the capability of each device node to autonomously establish social

relationships according to rules set by device owners. Nitti et al. extend this by providing a trustworthiness management scheme that allows participants in the SIoT to find trusted peers that can provide requested services, while isolating potentially malicious nodes, as measured by suspicious behavior. They create objective models stored in distributed hash tables to augment or replace existing subjective models.

The application of ML techniques enabled researchers to base trust management systems on quantitative trust computation, using numerically measurable values from the operating environment of systems deploying IoT (Jayasinghe et al., 2019). Jayasinghe et al. show that a computational framework integrating social and technical IoT trust measures enhances the security and performance of cyber-physical systems. They develop a system-level algorithm for computing measurable social (entity interaction) attributes (features) that form their trust computation model parameters. They use attributes of co-location relationship; co-work relationship; cooperativeness, frequency, and duration; reward system; mutuality and centrality; and community of interest to form a composite trust value. They validate their approach by using collected network features from the CRAWDAD MobiClique mobile social networking middleware dataset mapped to their defined IoT trust attributes. They then apply principal component analysis (PCA) based on singular value decomposition (SVD) for feature extraction and dimensional reduction, before using a support vector machine learning classifier as the trust prediction engine. Results show good convergence and improved recall and false negative rate over conventional weighted summation approaches for trust computation. While this work demonstrates using social attributes to establish a trust baseline for IoT services, it does not discuss specifications of the data, simulation, or real-world measurement for testing.

Aggregating and managing trust evaluation and computation throughout the network is a critical element of any trust management system. Alghofaili & Rassam present an event-driven architecture that provides a predictive trust management model to dynamically identity and mitigate untrusted IoT behavior, focusing on the computational trust prediction stage (Alghofaili & Rassam, 2022). They describe the design of a four-stage trust management system model: data collection, data preparation, trust prediction, and evaluation. In the prediction phase, they employ a Simple Multi-Attribute Rating Technique (SMART) for calculating node trust value using multi-criteria decision-making (MCDM) with node quality of service information weighted with Shannon's entropy method. An LSTM network, a type of RNN that excels at capturing long-term dependencies, detects malicious behaviors triggered by trust violations. This dynamic event-driven model detects a broad range of anomalous behavior, rather than conventional statistical or supervised learning methods, which focus on known attack patterns. Their approach validates the idea of computing trust values locally (at end node or edge) from collected network traffic data. It also reinforces the ML and FL papers reviewed in prior sections that staged architectural patterns of feature engineering, prediction computation, classification are effective for IoT security applications.

Trust also can be the basis for a data quality assessment framework to manage the large volumes of shared IoT data (Byabazaire et al., 2020). They focus on data from heterogeneous sensor environments, where no data reference standard is available. Existing data quality assessments are based solely on an instantaneous sampling of the data source's current state of operation. They propose a quality metric that uses current and past states and allows tuning based on the user interactions with the data source.

The still evolving complexity of the underlying technology and the many different perspectives from which IoT trust have been formulated to date are evident in the many proposed taxonomies and framework architectures. Recent research reflects an increasing diversity of methods for quantitatively evaluating trust to enhance existing qualitative approaches based on reputational standards. Alhandi et al. use a lens of trust evaluation in discussing their comprehensive survey of models of IoT trust, with a focus on wireless sensor networks (WSN) and radio frequency identification (RFID) devices (Alhandi et al., 2023). Prior IoT trust management system analyses focused on organizing IoT trust according to trustee or trustor subjective and objective properties, and trust relationship operational context (Yan et al., 2014). Alhandi et al. lay out core trust concepts organized into six categories in their synthesis of trust taxonomies: (a) trust attribute, whether the origin of trust is based on a social relationship or non-social relationship; (b) trust architecture, describing IoT device network topology (c) trust aggregation method, defining how trust attributes and trust metrics may be combined to create a composite trust score; (d) trust distribution, which describes how measures and quality associated with a trust interaction are conveyed; (e) trust type, performed on node behavior, or the data associated with device; and (f) trust model attack type, which characterizes IoT attacks per the three-layer IoT architecture model (perception, network, and application).

Each of Alhandi et al.'s categories has key characteristics that can be used for quantifying aspects of trust. For example, different trust architecture types specify measures of trust propagation (pathways by which the interactions associated with trust flow), level of maintenance, fault tolerance, stability, complexity of development, path diversity, and scalability. Alhandi et al. do not provide a rubric by which to rate the

relative efficacy, nor do they explore specific types of computation and quantitative measures that can be practically computed to predict real-time interaction. These should also include sizing to bound compute, memory, or power resources needed.

Aaqib et al. provide an alternative taxonomy for trust management systems in a networked environment that emphasizes the underlying statistical model used for trust evaluation to delineate categories (Aaqib et al., 2023). They assert systematic trust management of IoT devices should be a significant contributor to enhancing user privacy and authentication and actively managing system risk and security. They notably separate IoT trust models into computationally oriented trust management-based models that calculate trust scores for separating devices either as “trustworthy” or “untrustworthy” and AI-based models. These collectively characterize devices based on trust calculations, behavior learning, security, and privacy attributes. Their description of evaluation models and the suitability of trust metrics are useful for both researchers and practitioners: context-awareness; computational overhead, packet delay, recommendation, dynamicity, communication, social network, cooperativeness, scalability, energy, reputation (probability), and resilience. Aaqib et al. identify open research areas in performance metrics, energy and resource-constrained implementation, ensemble and multi-model approaches, attack modeling, AI/ML aggregation, and evaluation. Additionally, they do not provide a rubric for scoring the efficacy of trust measures, nor do they describe an architecture for measuring and testing attributes used to build practical metrics.

2.6 Summary and Conclusion

The research literature review in this chapter has covered a broad cross-section of the existing corpus of published research to date. The findings illuminate foundational

concepts, provide building blocks, reveal lessons learned, and open issues. The quest for cyber resilience of safety-critical systems guides the identification of the most essential security issues, advances, gaps, and opportunities supporting the praxis research efforts.

Table 2-6a highlights the role of deployment context, research problem domain, and architecture in finding an operationally relevant solution space. Table 2-6b shows the impact of applying edge computing and federated learning architectures, and trust computation and management concepts from other domains to IoT/IIoT security.

Table 2-6a. Summary of Literature Review and Gaps (Part 1)

Key Findings from Literature	Research Gaps and Opportunities
Operational Technology and Industrial Internet of Things Security (§2.2)	
• Convergence of IT and OT exposes new risks • Deployment of IIoT in critical infrastructure impacts system-wide resilience • Layered reference architectures are tools for assessing security risks and countermeasures • Hybrid of centralized and distributed security architectures improve performance	• Lightweight IIoT access control, authentication, and encryption • IoT/IIoT node behavior visibility • Network monitoring and intrusion detection tailored for industrial environments and processes • Integration of centralized and distributed security for IT and OT
Machine Learning Applications and Measures for IoT Security (§2.3)	
• Evaluating the performance of ML approaches for IoT security should done with metrics fit for purpose and data available • Domain- and problem-informed feature selection and engineering improves ML performance and simplifies implementation • Endpoint host implementations require alternative lightweight implementations that minimize performance tradeoff • Deep learning and generative models are sophisticated but expensive for IoT/IIoT	• Consistent IoT security ML performance metrics accounting for traffic class imbalance • Standardized feature selection for IoT/IIoT security suitable for practical implementation • High-performing ML-based models for resource-constrained IoT/IIoT • Integrating ML-based approaches into operational environments requiring dynamic response

Table 2-6b. Summary of Literature Review and Gaps (Part 2)

Key Findings from Literature	Research Gaps and Opportunities
Edge Computing and Federated Learning (§2.4)	
• Processing at the network edge is advantageous for addressing the scale and distribution for IoT • FL provides a novel architecture for IoT security that distributes computation and preserves the privacy of participants • Trustworthiness at the network edge can drive system-wide requirements • Edge-based FL with blockchain provides a viable foundation for trust management • Federated architectures should be highly tailored for IoT applications and data to maximize performance while balancing resources required	• Architecture guidelines for distributing security functions across cloud, fog, edge, and endpoints • Methodologies for tradeoff of FL security, privacy, and performance • FL architecture robustness for large-scale IoT/IIoT security • Resource-aware strategies for FL-based IoT/IIoT security implementations
Trust frameworks, architectures, and computation (§2.5)	
• Networked IoT trust exists in context of service interaction and behavior of the participants • Social IoT trust is derived from peer-to-peer interaction and owner reputation • Trust can be quantitatively computed from IoT node communication and data behavior • Trust computable at the edge contributes to system-wide trust through trust aggregation • Scalable IoT ecosystem trustworthiness requires an end-to-end trust management system	• Practical quantitative trust computation at the network edge • Effectiveness of network and data QoS measures on IoT trust • Scalable IoT trust management in distributed architectures • Dynamic IoT trust attack modeling • Integration of resource-efficient IoT trust management and intrusion detection for IT/OT networks

Identified gaps in access control, authentication, and encryption are outside the scope of this praxis. The other gaps and opportunities summarized in these tables have directly guided the experimentation direction of this praxis.

Chapter 3—Methodology

3.1 Introduction

The discussion of the preceding chapter provides a point of departure and identifies several of the most important opportunities for this praxis to make contributions to the body of knowledge and the current practice of securing mission-critical information technology/operational technology (IT/OT) network infrastructures with internet of things (IoT) and industrial internet of things (IIoT) devices.

This chapter follows and expands the praxis research architecture workflow shown in Figure 1-2. Section 3.2 details the requirements for data to support the praxis research objectives. Section 3.3 outlines the procedures for preparing the data for exploratory data analysis, as well as the remaining experimental steps. Section 3.4 discusses the trust-based feature engineering approach, which combines knowledge from multiple application domains with several feature selection methods. Section 3.5 describes the approach for establishing a baseline set of classification models using centralized conventional machine learning (ML) algorithms. Sections 3.6 and 3.7 outline the architectures, frameworks, and methods for deep learning (DL) and federated learning (FL), respectively. Finally, Section 3.8 summarizes the approach for evaluating the experimentation results presented in Chapter 4 against individual model benchmarks and the praxis research questions and hypotheses laid out in Chapter 1.

3.2 Data Sources

A robust dataset supporting the research objectives of this praxis must simulate communication behavior measurements from IoT/IIoT devices and represent realistic IT/OT network traffic, be sufficiently heterogeneous and adaptable to support a diverse

set of operational and experimental scenarios and be able to be rigorously characterized to support verifiable and reproducible results. The dataset will be used for selecting trust-based features for training and evaluating ML models for classifying IoT/IIoT cyberattacks. Since 2010 researchers have published over fifty datasets for IoT network performance, security, and privacy research and evaluation (Alex et al., 2023; Keersmaeker et al., 2023). Key criteria for dataset selection are the following:

- Mixed traffic of industrial IoT, consumer IoT, and enterprise IT devices
- Attributes directly measurable from an operational or testbed network
- Standardized, rather than custom-derived attributes
- Labeled for binary and multiclass classification (at least seven attack classes)
- Sufficiently large number of records (at least 1 million) for deep learning
- Collected or generated within last five years to reflect traffic and attacks representative of modern IT/OT networks
- Existing corpus of published work using dataset for ML-based security research to provide reference results.

Section 4.2 details the analysis of alternatives resulting in the choice of the tabular NetFlow attribute-based NF-ToN-IoT-v2 dataset from the University of Queensland (Australia) as this praxis' primary data source for experimentation.

3.3 Data Preprocessing and Exploratory Data Analysis

Data preprocessing consists of the steps to ingest the dataset, statistically characterize the underlying data, gain an understanding of the key attributes, assess the classification or prediction objectives, and condition the data for the learning process. Figure 3.1 shows the key steps of data preprocessing for this praxis, consisting of

exploratory data analysis, data cleansing, and data transformation. A typical ML data preprocessing workflow might also include feature selection for dimension reduction to improve model execution performance, identification and elimination of features not directly supporting the prediction or optimization objective, and then end with splitting, or partitioning, of the dataset into subsets for ML model training and inference testing. For this praxis, feature engineering plays a significant role in shaping the course of the ML modeling and prediction and is established as a separate step, followed by the data splitting and classification step, as discussed in §3.3 and §3.4, respectively.

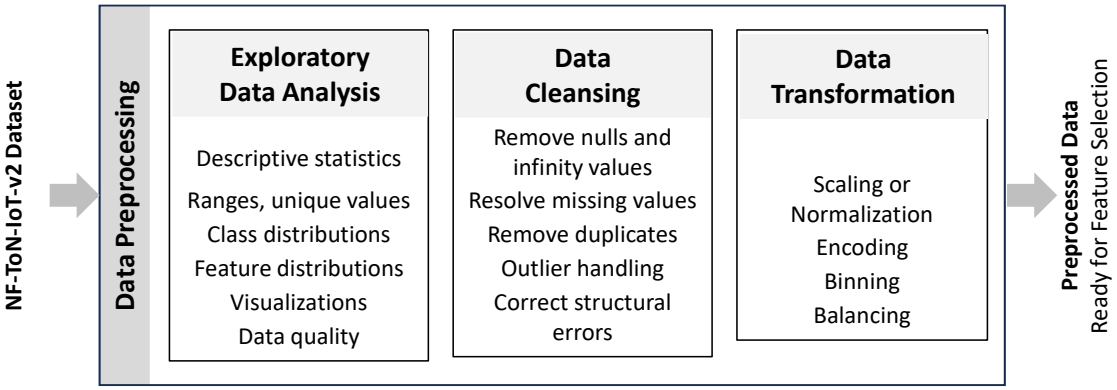


Figure 3-1. Data Preprocessing and Exploratory Data Analysis

Exploratory data analysis (EDA) provides the structural and statistical insights into the dataset composition necessary to identify the data cleansing and transformation steps. The praxis dataset NF-ToN-IoT-v2 is a tabular set of labeled network flow records. EDA for this dataset examines the number of flow sample records, and the number, range, unique values, statistical distribution, and data types of the feature attributes. Bar graphs, range charts, and other basic visualizations can assist in the EDA process and can quickly identify outliers, data imbalance (greatly uneven distribution of data by class), and other potential data quality issues. EDA and all major tasks for the praxis experiment are designed to be executed in a Python and Jupyter notebook environment.

The data cleansing step handles any data issues that may cause an interpretation or computation problem for the downstream machine algorithms. Based on the findings of the EDA process, this typically involves removing nulls (not a number (NaN), dashes, infinity values); deleting or imputing missing values; removing duplicates; and resolving or adjusting or missing labels. When the range of individual data features is too wide, feature selection and classification tasks may exhibit unintentional weighting, requiring the application of a transformation approach like min-max or standard scaling. Next, to support the classifier training and inference steps, encode any non-numerical attributes into numerical values using techniques like label encoding (for values with ordinality) or one hot encoding, which assigns a binary value to the presence or absence of a categorical feature value. Candidate approaches for adjusting the modeling for class imbalance are discussed in §3.5.

3.4 Feature Engineering and Selection

A central part of this praxis involves the combined application of domain knowledge, information-theoretic statistical ranking techniques, machine learning architecture, and trust computation concepts. Feature engineering provides a unifying

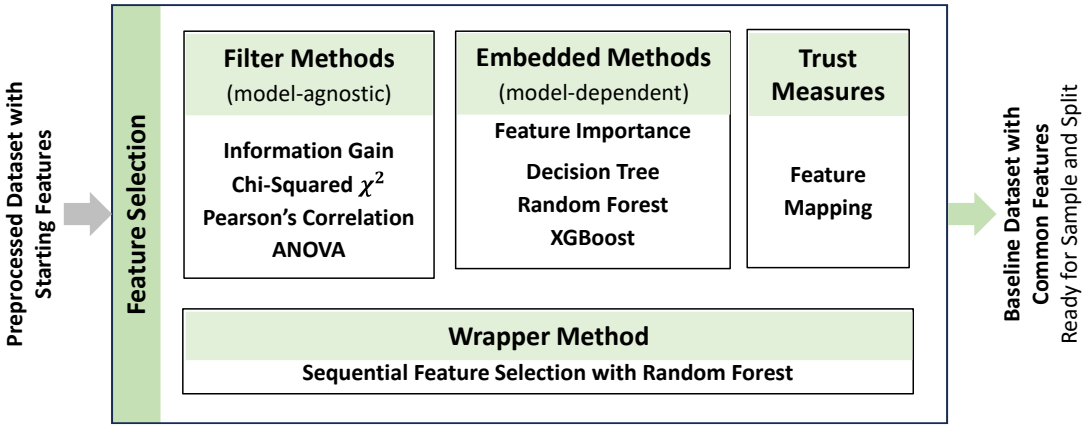


Figure 3-2. Feature Engineering and Selection Architecture

framework for this integration. Feature engineering applies domain knowledge to extract new attributes from raw data or transform existing data attributes for improving the representation of data to support machine learning modeling and inference. Figure 3-2 shows the overall feature engineering and selection architecture for this praxis.

Dimensionality reduction is a key outcome of feature engineering. This reduces the number of features (dimensions) and complexity needed to represent a dataset at a given level of fidelity, thereby simplifying computation and storage requirements. Feature engineering consists of feature extraction to generate a starting feature set and then feature selection to choose the most relevant and least redundant set of features suited for a defined machine learning problem (Butvinik, 2021; Galli, 2021; Guyon & Elisseeff, 2003; Li et al., 2017). Feature extraction techniques such as principal component analysis (PCA) and singular value decomposition (SVD) transform an original high fidelity, high-dimension representation of a dataset into a new lower dimensional feature space, but often as a concise representation with only mathematical meaning (Dhal & Azad, 2022; Li et al., 2017). This praxis aims to assess the utility of features associated with operationally meaningful trust measures in network traffic and focuses on supervised feature selection rather than feature extraction.

This praxis proposes a model-agnostic method grounded in applying IoT trust concepts and domain knowledge to create a standard feature selection map. The current state of the art for IoT trust discussed in §2.5 establishes the background research in network or service quality of service (QoS) (Alani, 2023; Alhandi et al., 2023; Ma et al., 2021); social internet of things (SIoT) (Jayasinghe et al., 2019; Nitti et al., 2014); and data quality trust (Byabazaire et al., 2020) constructs that serve the starting point for

emerging IoT trust computation based on node behavior. This praxis undertakes a detailed analysis of these established IoT trust constructs, their deployment objectives, and both real and simulated network measurements and experiments. The output of this analysis will be a proposed map of IoT trust constructs to network traffic flow attributes in the dataset. Praxis experimentation explores the efficacy of this approach to serve as an augmentation of or alternative to established feature selection methods in Table 3-1.

Table 3-1. Feature Selection Methods

	Characteristics	Key Techniques
Filter	• Uses statistical properties (distance, similarity, dependence, entropy) • Computationally fast • Scalable to high-dimensional data • Independent of classifier model • May have limited standalone utility	• Variance Threshold • Chi-squared* • Information Gain* (Mutual Information) • Correlation (Pearson*, Spearman, Kendall) • Analysis of Variance (ANOVA)* • Fisher's Score • Relief (binary), ReliefF (multiclass)
Wrapper	• Searches subsets of features using classifier model cross-validation • Computationally intensive • Prone to overfitting	• Sequential Feature Selection* (Forward, Backward) • Recursive Feature Elimination* • Exhaustive Feature Selection
Embedded	• Part of classifier model training • More computationally efficient than wrapper methods • May not generalize	• Feature Importance for tree-based or boosted tree-based model* • Regularization (LASSO, Ridge Regression, Elastic Net)
Hybrid	• Uses combination of other methods • Selects features based on model performance impact	• Permutation Feature Importance

Adapted from (Dhal & Azad, 2022; Li et al., 2017; Sarhan, Layeghy, & Portmann, 2022b)
(*) Denotes techniques used in this praxis

3.4.1 Filter Methods

Filter feature selection methods rely on information-theoretic statistical properties that are agnostic of a classifier model. By computing an importance metric that reflects the dependence of features on one another or on the classification target variable, a filter method identifies features that describe a shared underlying attribute. This provides a quantitative basis for selecting or dropping that feature. Typically, a greedy algorithm selects the *k*-best features based on the individual feature importance metric (Guyon & Elisseeff, 2003; Li et al., 2017). These metrics are computed without reference to a specific classification algorithm and are thus fast to compute and easy to scale to high or lower dimensional data. The Python Scikit-learn library provides a ready implementation of each of the filter methods used in this praxis.

The chi-squared (χ^2) test, which measures the independence of two events, applied to feature selection tests for the independence of a feature and its target class label (Sarhan, Layeghy, & Portmann, 2022b). Equation 1 below shows the chi-squared calculation between for each observed sample candidate feature X_i and its target class label Y_i expressed as a measure of deviation from each other for degrees of freedom c. From a contingency table built from the data samples, a high χ^2 value indicates that the null hypothesis of variable independence should be rejected. In this case, there is statistical dependence between the candidate feature and the target class, and the feature should be selected as a predictor for classification modeling.

$$\chi_c^2 = \sum_i \frac{(X_i - Y_i)^2}{Y_i} \quad (1)$$

Mutual information between a feature variable X and target class label Y can be expressed in terms of the change in entropy H about the target class label given the feature. Since entropy is a measure of uncertainty, a reduction in entropy equates to a gain in information. The information gain IG between X and Y is expressed as below:

$$IG(Y;X) = H(Y) - H(Y|X), \tag{2}$$

$$\text{where } H(X) = -\sum_{i=0}^{N} p(x_i)\log_2(p(x_i)) \text{ for probabilities } p. \tag{3}$$

Higher values in information gain signify greater contribution in making the correct classification and hence an important feature to select.

Correlation measures are used to select candidate features that are highly correlated with target class labels and to identify features that are highly correlated with each other, indicating redundancy and an opportunity for dimensionality reduction. The most common measure is Pearson's correlation coefficient, which for features X and target class labels Y, measures the extent of their linear relationship and is given by their covariance $cov(X,Y)$ divided by their standard deviations σ_X and σ_Y:

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y} \tag{4a}$$

For sampled datasets, Pearson's correlation coefficient is expressed as

$$r_{xy} = \frac{\sum_{i=0}^{N}(x_i - \overline{x}_i)(y_i - \overline{y}_i)}{\sqrt{\sum_{i=0}^{N}(x_i - \overline{x}_i)^2}\ \sqrt{\sum_{i=0}^{N}(y_i - \overline{y}_i)^2}} \tag{4b}$$

for sample size N, feature data point x_i, target class data point y_i, and their respective sample means $\overline{x}_i$ and $\overline{y}_i$. A correlation value threshold is used to collect the best number k of features most correlated with the target class. Another threshold is established to

examine the features with the strong pairwise correlation to each other eliminating redundant features, which then improves classification efficiency.

A one-way analysis of variance (ANOVA) *F*-test assesses whether the mean for each feature is the same as the that of the target classes. The ANOVA Fisher coefficient, or *F*-statistic, is the ratio of mean sum of squares between classes and the mean sum of squares within the classes, given by

$$F = \frac{\sum_{i=1}^{C} n_i(\overline{X_\iota} - \bar{X})^2)/(C-1)}{\sum_{i=1}^{C}\sum_{j=1}^{n_i}(x_{ji} - \overline{X_\iota})^2)/(N-C)} \quad (5)$$

where n_i is the number of observations in the ith class, $\overline{X_\iota}$ is the sample mean of the predictor (feature) for target class i, $\bar{X}$ is the overall mean of feature X, x_{ji} is the jth observation of the ith class for total number of classes C and sample size N (Elssied et al., 2014; "*F*-Test," 2024). Under the null hypothesis (population means of the classes are the same), F follows the F-distribution with degrees of freedom $d_1 = C - 1$ and $d_2 = N - C$ and should have an F value less than the F-distribution critical value. A large F indicates that the between group variance is large with respect to the within group variance, which indicates the corresponding feature is of statistical importance.

Spearman's rank correlation R_s, Kendall's tau (τ) correlation, and Fisher's score were also examined but not employed due to misaligned assumptions on variable distribution or type (e.g., ordinality) or offering little differentiation from other measures.

3.4.2 Embedded Methods

Certain families of ML classifier models have feature selection steps embedded within their learning architectures. Tree-based and boosted tree models provide direct visibility to how they use feature selection-based computations as part of their tree-

splitting decision steps, and these can be used to calculate feature importance. These measure the decrease in Gini node impurity weighted by the probability of reaching that node, as the tree splits are created during the learning phase. This praxis uses the embedded feature importance fi_i for a feature number i as calculated by the Scikit-learn library (Dale, 2017; Ronaghan, 2019; Scikit-learn 1.4.1, 2024):

$$fi_i = \frac{\sum_{j:\text{node j splits on feature i}} ni_j}{\sum_{j \in \text{all nodes}} ni_j} \tag{6a}$$

for the node importance ni_j of node number j, which is calculated as below:

$$ni_j = w_j C_j - w_{left(j)} C_{left(j)} - w_{right(j)} C_{right(j)} \tag{6b}$$

for Gini node impurity C_j, weighted number of samples in node j as a fraction of the total weighted number of samples w_j, and the weighted node impurities $w_{left(j)} C_{left(j)}$ and $w_{right(j)} C_{right(j)}$ of respective left and right tree branch children nodes. The feature importance is then normalized to a value between 0 and 1.

$$(fi_i)_{norm} = \frac{fi_j}{\sum_{j \in \text{all features}} fi_j} \tag{6c}$$

For an ensemble tree model such as random forest or extra trees, or a gradient boosted model such as XGBoost, it is also summed over total number of trees T created:

$$Model\ fi_i = \frac{\sum_{j \in \text{all features}} (fi_i)_{norm}}{T}. \tag{7}$$

The higher the $Model\ fi_i$ measure, the more important the feature is the model, and the top values can be selected by a count or statistical threshold.

3.4.3 Wrapper Methods

Wrapper methods for feature selection iteratively use the performance of a specified classifier model to assess a current candidate selected feature set. This involves

searching subsets of features using classifier model cross-validation of a candidate feature set and recording the impact (decreased or improved model performance) of an addition or subtraction of a feature to the candidate feature set. Depending on the choice of specific algorithm and quality of data, a wrapper method can fit non-linear models and provide performance bounds as well as feature selection candidates that may be used in other cases. Common iterative search approaches include backward feature elimination (starting with all available features and iteratively removing the least important), forward feature selection (starting with and empty set of features and iteratively adding one feature, typically the most important), recursive feature elimination (RFE) (form of backward elimination that starts with a univariate feature importance calculation), and exhaustive search (evaluates all possible combinations). These can be computationally expensive and may be prone to overfitting, especially for large feature sets.

To implement a wrapper method, this praxis uses Scikit-learn's RFE and RFE with cross-validation (RFECV) and the Mlxtend library's SequentialFeatureSelector functions, with a base random forest classifier. The results are then combined with the results from the filter, embedded, and trust mapping methods to select a greatly reduced number of features that still achieve close to optimal performance. This is an established benchmark approach in published experiments (Awad et al., 2022; Awad & Fraihat, 2023; Sarhan, Layeghy, & Portmann, 2022a). This final output of the feature selection step is a baseline set of features selected from the original full feature set that will be used for the detailed learning modeling and testing steps.

3.5 Benchmark Model Development and Classification

Following feature selection, the core ML architecture implementation, learning model training, model prediction performance testing and measurements, and comparative assessments proceed. Figure 3-3 shows the workflow to finalize the data preprocessing and preparation for modeling in a centralized ML architecture.

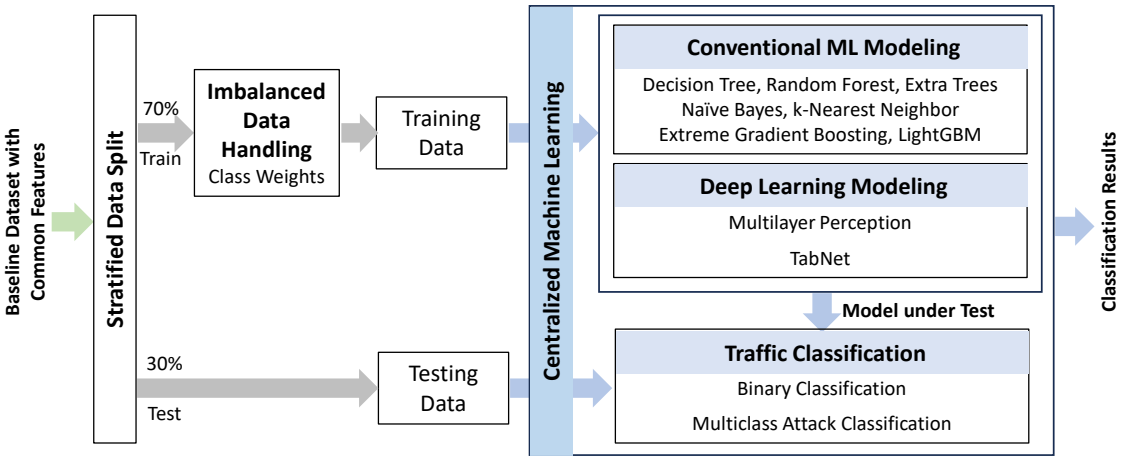


Figure 3-3. Centralized Machine Learning Architecture

3.5.1 Final Data Preprocessing and Splitting for Modeling

Once trust-based feature selection is complete, the baseline dataset that results is ready to be split for ML training and testing—in proportions of 70% for training and 30% for testing. A stratified data split produces data subsets that preserve the cyberattack class distribution of the larger dataset. This approach avoids exacerbating potential bias due to the dataset class imbalance.

Reflecting the reality of an operational network environment where malicious attack traffic is rare relative to normal traffic, network intrusion detection system datasets such as NF-ToN-IoT-v2 do not have approximately equal samples of each class. With significant class imbalance, machine learning algorithms may overfit for the majority

class and show poor performance for the minority class or classes, without some adjustments to compensate for the class imbalance for the training pipeline. Imbalanced data may be handled most often by data methods that change the distribution of class samples or learning model algorithm methods that modify the optimization cost function or apply adaptation to the learning kernel (He & Garcia, 2009; Krawczyk, 2016). This praxis applies class weights w_{class} to the model training pipeline:

$$w_{class} = \frac{(Total\ number\ of\ samples)}{(Number\ of\ classes) \times (Number\ of\ samples\ in\ class)} \quad (8)$$

Scikit-learn's compute_class_weight function and "balanced" parameter for select ML models support implementing class weighting. The NF-ToN-IoT-v2 dataset's authors Sarhan et al. also apply class weights for their benchmark classification pipeline. This praxis found alternate approaches of random oversampling of the minority class and adaptive synthetic sampling (ADASYN) to produce less consistent results.

3.5.2 Centralized Conventional Machine Learning Modeling

After the final data preprocessing and conditioning for imbalance, the next step uses machine learning models in a centralized architecture to establish the baseline performance metrics against the selected trust-based feature set. Investigating first a group of conventional ML algorithms and then a set of deep learning algorithms addresses research question 2 and tests research hypotheses 2, 3, and 4 (see §1.6). Training, testing, and 5-fold cross-validation will be conducted for both binary classification discriminating malicious traffic from benign traffic (identified by the "Label" target variable), and multiclass classification discriminating among benign traffic and the nine classes of malicious traffic, as identified by the "Attack" target variable. Table 3-2 summarizes the characteristics of the ML algorithms examined for this praxis.

Table 3-2. Conventional Machine Learning Classifier Algorithms Utilized

Machine Learning Classifier	Praxis Modeling Applicability
k-Nearest Neighbors (k-NN)	Classifies data point according to *k* closest training set data points in feature space. Various distance measures (e.g., Euclidean, Manhattan. Mahalanobis) can be chosen. *Limitation*: requires storing entire training set, which impedes scaling
Gaussian Naïve Bayes (GNB)	Classifies data point by applying Bayes' theorem of conditional probabilities to predict. *Limitation*: assume features are conditionally independent
Decision Tree (DT)	Creates hierarchical branching tree structure at which the leaf nodes are classification decisions. Simple to implement and to interpret. *Limitation*: sensitive to training data, prone to tree instability
Random Forest (RF)	Ensemble of decision trees using bootstrapping (randomly chosen subset of features for subtrees), using an optimal split at each node. Highly accurate, fast to train. *Limitation*: predictions can be computation intensive and slow for many trees
Extremely Randomized Trees, or Extra Trees (ET)	Ensemble of trees, sampling the entire dataset and where nodes are split randomly, rather than according to an optimal evaluation. Low variance, fast predictions. *Limitation*: can have poor performance on imbalanced data and harder to interpret results
Extreme Gradient Boosting (XGBoost)	Ensemble gradient boosting algorithm built on decision trees as base learners with regularization for generalization. Optimized for efficiency and speed, especially for tabular data. *Limitation*: computation intense for large datasets with large feature space.
Light Gradient-Boosting Machine (LightGBM)	Gradient boosting model optimized fast performance and low memory utilization for large tabular datasets. Employs Gradient-based on side sampling (GOSS) and Exclusive Feature Bundling (EFB). *Limitation*: difficult to interpret

Adapted from (Fraihat et al., 2023; Mahdavinejad et al., 2018)

The k-Nearest Neighbor (k-NN) classifier is considered for its straightforward and flexible application, but a potential limitation to be examined is its requirement to store the entire training set to use for the prediction step, which may be resource-intensive. The Gaussian Naïve Bayes (GNB) classifier explores how well a conditional probability model may fit. Tree-based and boosted tree-based algorithms offer characteristics that are well-suited to the supervised classification requirements of this praxis: computational

efficiency, robustness to missing values and outliers, the ability to express complex relationships among features, and high model interpretability. For these classifiers, praxis experimentation must mitigate the risk of overfitting, model instability, and bias through feature selection and model tuning. This praxis uses random forest for feature selection and to establish the baseline centralized ML performance benchmark.

3.6 Deep Learning

Deep Learning (DL) approaches have been embraced for their ability to model sophisticated or hidden attributes and behaviors for large datasets. This praxis explores the use of a multilayer perceptron (MLP) feedforward deep neural network (DNN) and a recently developed deep generative model known as TabNet that has yet to be studied in any depth for IoT applications (see Table 3-3). Convolutional neural networks (CNNs), largely responsible for the computer vision AI breakthroughs of the 2010s, and recurrent neural networks (RNNs), a DNN well-suited for time-sequenced data, and advanced generative DL models have attracted research attention for IoT security in recent years.

Table 3-3. Deep Learning Classifier Approaches Utilized

Deep Learning Classifier	Praxis Modeling Applicability
Deep Neural Network (DNN) or Multilayer Perceptron (MLP)	Feed-forward artificial neural network using a non-linear activation function and multiple hidden layers between input and output layers to create complex models of large scale data. Network node weights and biases adjusted through back-propagation of error adjustments, usually based on gradient descent. *Limitation*: computationally expensive; large amounts of data, extensive training required for convergence
TabNet	Encoder-decoder architecture designed for tabular data with minimal preprocessing. Uses sequential-attention transformer mechanisms for instance-wise feature selection at each step. Provides local and global interpretability. *Limitation*: Requires manual hyperparameter tuning.

Adapted from (Al-Garadi et al., 2020; Arik & Pfister, 2021; Mohammadi et al., 2018)

CNNs, RNNs, and generative adversarial networks (GANs) were investigated for this praxis, but as discussed in §2.3, demand complex programming and resource-intensive computation, which are not always well-suited for edge IoT/IIoT applications.

This praxis uses an MLP DNN with two hidden layers between the input layer representing the feature set variables and the final output layer representing the classification prediction. Each neuron unit y_j^l of an intermediate, or hidden, layer l transforms inputs x_i from the preceding layer according to an activation function $f()$:

$$y_j^l = f\left(\sum_i w_i x_i + b\right) \tag{9}$$

for weight $w_i$and bias b. The power of an MLP comes from the use of nonlinear activation functions. This praxis uses the sigmoid logistic function, rectified linear unit (ReLU) and for multiclass probabilities, softmax:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{10a}$$

$$ReLU(x) = \max(0, x) = \begin{cases} x, & x > 0 \\ 0, & x \leq 0 \end{cases} \tag{10b}$$

$$\sigma(\mathbf{z})_i = \frac{e_i^z}{\sum_{j=1}^K e^z j} \tag{10c}$$

This praxis uses adaptive moment estimation (Adam) and stochastic gradient descent (SGD) optimizers to specify weights w_i that minimize the loss function, which quantifies the prediction error between the predicted and target values. Binary cross-entropy (BCE) or categorical cross-entropy (CE) are the preferred loss functions for binary or multiclass classification problems, respectively. Exploring choices for these hyperparameters and their effect on classification performance are integral to the modeling experiments.

Neural networks are discriminative DL approaches that model the conditional probability $p(Y|x)$ of a set of data instances x and a set of target labels Y to discriminate between different types of data instances. Recent new alternative DL algorithms are generative and model the joint probability $p(x, Y)$, allowing the ability to generate new data instances on the dataset space following that estimated probability model. Generative transformers use a self-attention mechanism to learn context and relationships among sequential data input. While performance benchmarks for transformer-based IoT security are still developing (Manocchio et al., 2024), the TabNet architecture developed by Google Cloud in 2021 (Arik & Pfister, 2021) provides an attractive option to explore for this praxis, since it was designed specifically for tabular datasets. TabNet uses an encoder—consisting of a feature transformer, attentive transformer, and a feature mask—paired with a decoder with multiple feature transformers. Features chosen at each step are recorded as interpretable information and aggregated for global feature importance.

This praxis constructs and evaluates DL models using the PyTorch framework, which is more Pythonic and structured for research and prototyping than TensorFlow with the Keras application programming interface (API), which is tuned for large-scale deployment. PyTorch implements an MLP as a sequential neural network. Scikit-learn also supports basic MLP models, but it does not support parallel implementation on GPU hardware as easily as PyTorch. TabNet has APIs for both PyTorch and Scikit-learn.

3.7 Federated Architecture

After establishing centralized learning baseline results, research then proceeds to investigate research question 3 and the corresponding hypothesis 5 on the viability and performance of a federated learning (FL) solution. The guiding principle of FL is to move

the shared global model weights from an aggregation server to the data at the edge, rather than to move the data from the edge clients to a central server to combine data and create the global model. This approach fits scenarios where for privacy or security reasons local data should not or cannot be shared or co-mingled. The FL configuration introduced by McMahan et al. (see §2.4) is known today as an FL architecture with a horizontal data partition, or as horizontal FL. In horizontal FL, clients share a common feature space and operating domain but different sample space (i.e., different users, data owners, and local resources). In contrast, vertical FL architectures serve a common sample space but different feature spaces. Federated transfer learning is a third major FL architecture, where the sample space and the feature space are both different (Ferrag et al., 2021; Mothukuri et al., 2021; Yang et al., 2019).

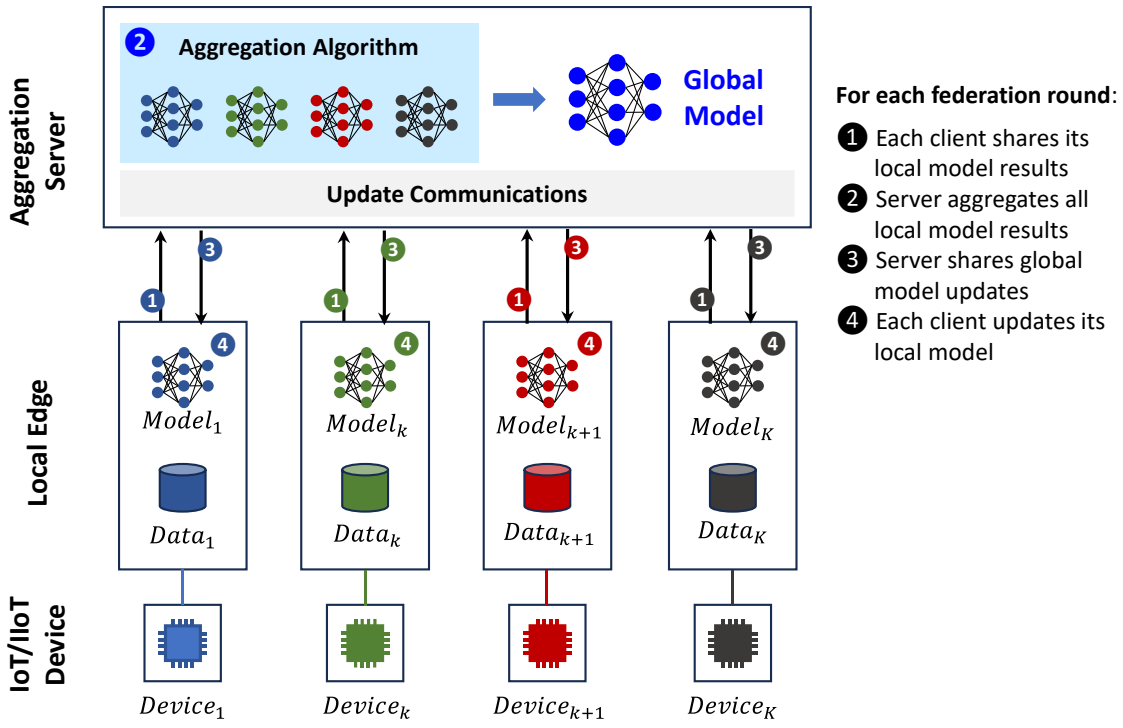


Figure 3-4. Horizontal Federated Learning Architecture

This praxis implements a form of horizontal FL. As shown in Figure 3-4, The process flow for a learning cycle is called a federation round: (1) using starting model parameters and its own local data, each client computes a training round and securely shares the results with the aggregation server; (2) the server aggregates the model updates from each of the K clients without learning any private client data and updates the global model; (3) the server sends updated global model results to each client; and (4) each client updates its own local model. Rounds continue until the model loss function converges to an established threshold. This approach is robust against differences in client learning capability and data, intermittent uptime, and irregular update times.

This praxis uses McMahan et al.'s original FedAvg generalized stochastic gradient descent algorithm as baseline aggregation method:

$$\forall k, \qquad w_{t+1}^{k} \leftarrow w_t - \eta g_k \tag{11a}$$

$$w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^{k} \tag{11b}$$

such that for each federation round t each client k will changes its next weight w_{t+1}^{k} take a gradient step g_k multiplied by the learning rate η. The server then updates the next global weight w_{t+1} by the weighted average (number of client k data points $n_k$divided by the total data points n) of the client weights w_{t+1}^{k}. For an IoT IDS problem, Lazzarini et al. performed a comparison of FedAvg alternatives: FedAvgM (applies client-side momentum with the Nestorov accelerated gradient to improve the calculation of weights in the presence of noisy data); and FedAdam and FedAdagrad (both server-side adaptive federated optimization). While FedAvgM performed comparably to FedAvg, FedAdam and FedAdagrad performed poorly, which illustrates the adaptive algorithms' sensitivity

to the number of participating clients. These were designed to account for large-scale cross-device federation (Lazzarini et al., 2023; Reddi et al., 2021).

The framework chosen to implement the FL experiments for this praxis is Flower, which is open source platform-independent ML framework-agnostic and designed for both rapid experimentation and production scale deployment (Beutel et al., 2022). Flower includes built-in support for multiple aggregation algorithms and provides a virtual client architecture that can instantiate an FL system on a heterogenous mix of operating systems and hardware platforms or fully as a local or cloud simulation. Agnostic to ML framework, Flower natively supports Scikit-learn, PyTorch, TensorFlow, and their extended ecosystems. Alternatives such as Syft from OpenMined (focused on privacy and secure communications) are less flexible for the experiments of this praxis.

3.8 Performance Evaluation

A key finding from the literature search on ML applications and measures for IoT security (§2.2) highlights the importance of using performance metrics appropriate to the task objectives, dataset characteristics, and problem domain. For this praxis, major considerations for choosing the metrics are (1) the ability to provide interpretable insight into the impact of parameter or model architecture changes on individual predictions; (2) class imbalance in the data, which is extreme for specific attack classes; and (3) practical utility in cybersecurity operations. To ensure clarity, we express the equations here for binary classification, but the discussion applies to multiclass classification, where we express true and false predictions in terms of individual classes.

Fundamental building block performance metrics include the following:

$$Precision = \frac{TP}{TP + FP} \tag{12}$$

$$True\ Negative\ Rate = Specificity = \frac{TN}{TN + FP} \tag{13}$$

$$Detection\ Rate = \ True\ Positive\ Rate = Recall = \ \frac{TP}{TP + FN} \tag{14}$$

$$False\ \ Alarm\ Rate = False\ Positive\ Rate = \frac{FP}{FP + TN} \tag{15}$$

for number of true positive prediction results TP, true negative predictions TN, false positives FP, and false negatives FN. Precision is the percentage of correctly predicted positive results out of the total predicted as positive. Specificity, or true negative rate (TNR), is the probability an actual negative will be predicted to be negative. Detection rate (DR), or true positive rate (TPR), is the percentage of correctly classified attack samples out of the total number of attack samples. False alarm rate (FAR) measures the percentage of false alarms (incorrectly classified attack samples) out of the total number of benign samples. Note that FAR and TNR add to 1. Dataset authors Sarhan, Layeghy, Moustafa, et al. use DR and FAR as key metrics for performance benchmarks. The terms "detection rate" and "false alarm rate" are preferred in network and cybersecurity operational contexts over the corresponding statistical terms true positive rate (TPR) or recall and false positive rate (FPR), respectively. This praxis adopts DR and FAR for comparison to existing research and to aid the application of key results to practice.

The widely used metric accuracy is automatically calculated with many Python Scikit-learn and PyTorch model predictions:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{16}$$

Accuracy (Acc) measures how well the model predicts samples in the correct class overall, but it does not account for specific class prediction performance. For imbalanced data, accuracy can obscure the prediction performance for minority classes. Balanced accuracy, F1-score, Matthews Correlation Coefficient, and receiver operating curve provide better insight into performance for imbalanced data. Balanced accuracy (bAcc) is the equally weighted average of detection rates or recalls for each class (Grandini et al., 2020). For binary classification this can be expressed as

$$Balanced\ Accuracy = \frac{TPR + TNR}{2} = \frac{DR + 1 - FAR}{2} \tag{17}$$

If model results show overbias to majority classes, class weighting as shown in Equation 8 can be applied to balanced accuracy to generate a weighted balanced accuracy metric.

The F1-score is the harmonic mean of precision and recall:

$$F1 = \left(\frac{2}{precision^{-1} + recall^{-1}}\right) = 2 \cdot \left(\frac{precision \times DR}{precision + DR}\right) \tag{18}$$

F1-score can be interpreted as a weighted average between precision and recall, and highlights that the cost of false positives and false negatives is different, as is the case for intrusion detection. This is the primary metric used in the praxis hypotheses.

The Matthews Correlation Coefficient (MCC) measures the correlation of the true classes *c* with the predicted labels *l*:

$$MCC = \frac{Cov(c,l)}{\sigma_c \cdot \sigma_l} \tag{19a}$$

for covariance $Cov(c,l)$ and standard deviations σ_c and σ_l. For binary classification:

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{19b}$$

MCC ranges from -1 to 1 and includes all values of a classification confusion matrix in both the numerator and denominator, and it is considered a good measure of model predictions with imbalanced data. Values close to 1 indicate prediction results highly correlated with correct results (i.e., very good performance), values close to 0 indicate no correlation (i.e., close to random guessing, or poor performance), and values close to -1 indicate strong correlation between predictions and the inverse of the true classes. The latter often indicates a problem with the construction of the classification problem or labeling, which can then be addressed (Grandini et al., 2020).

For binary classification, the receiver operating curve (ROC) typically expressed in ML literature as a plot of true positive rate versus false positive rate. In cybersecurity terms, ROC is expressed as detection rate versus false alarm rate. This curve shows the trade-off between DR and FAR, where an ideal model prediction is given by the area under the curve of the receiver operating curve (ROC AUC) of 1. The early recall (low FAR) range performance will be particularly important.

Chapter 4—Results

4.1 Introduction

The Jupyter Notebook, using the Python programming language and its extensive ecosystem of data science, scientific computing, and machine learning libraries, served as the platform for performing the experimentation for this praxis. This provided a unified, flexible interactive environment for programming, exploring, executing, and analyzing code, data, models, and results. Cloud-based Google Colab Pro+ (CPU with up to 51 GB CPU RAM and GPU with a variable amount of GPU RAM available, Python 3.10.12) served as the primary environment. A local Windows 11 machine (CPU only, with 16 GB RAM, Python 3.10.9, Visual Studio Code 1.86.2 integrated development environment) provided a supplemental environment. The following sections present the key results of the steps laid out in Chapter 3 and provide analysis in the context of both the praxis experimentation plan and existing published research, as appropriate.

4.2 Data Source Selection

Selected as the primary data source for this praxis is the NF-ToN-IoT-v2 dataset from the University of Queensland (Australia), which meets all the criteria set forth in §3.2. NF-ToN-IoT-v2 was created with forty-three standardized NetFlow features (Sarhan, 2022; Sarhan, Layeghy, & Portmann, 2022c) from using the packet capture (pcap) and ground truth event data of the widely-used TON_IoT dataset from the University of New South Wales Canberra (Australia) (Alsaedi et al., 2020). This approach preserves the nine types of cyberattacks across a newly generated overall IT/OT traffic profile across 16.9 million records. The literature review of ML applications and measures for IoT security in Section 2.3 uncovered the importance of using a standard

feature set for public intrusion detection system (IDS) datasets not only for better comparing the performance of ML-based network traffic classifiers (Sarhan, Layeghy, & Portmann, 2022c), but also to improve transparency and consistency of experiment methodology (Awad et al., 2022; Zolanvari et al., 2019) and increase explainability of the results (Alani, 2023; Alani et al., 2022; Alani & Miri, 2022; Sarhan, Layeghy, & Portmann, 2022a). This research has pointed to the effectiveness of standard features based on attributes from Cisco NetFlow version 9 (Cisco, 2011). NetFlow's ubiquity as a *de facto* industry standard built into many network monitoring tools, traffic generation suites, and routers, makes the choice of a standardized NetFlow-based dataset readily extensible to further research and suggests practical operational implementations.

Table 4-1 shows all datasets evaluated to serve as the praxis dataset. The prior generation NF-ToN-IoT dataset uses only twelve NetFlow features (Sarhan et al., 2021) and was selected as a secondary dataset for exploring ML pipeline architectures and workflow. The NF-UQ-NIDS-v2 dataset is a superset of the NF-ToN-IoT-v2 dataset and NetFlow-feature refactored versions of other widely used IDS datasets. IIoT traffic is present in only the NF-ToN-IoT-v2 portion, so use of NF-UQ-NIDS-v2 has been identified for potential future research on extending this praxis research to networks with general enterprise IT and IoT traffic. CICIoT2023 is a large new IoT dataset from the Canadian Institute for Cybersecurity at the University of Brunswick with a diverse set of 33 cyberattack types across seven families and a combination of flow-based, protocol-specific, and custom statistical features (Neto et al., 2023). Traffic is primarily consumer-class IoT devices, and as a new dataset, it has yet to generate significant published research results.

Table 4-1. Selected and Candidate Datasets for Praxis Experimentation

Dataset	Traffic	Attacks	Records	Date	Features
NF-ToN-IoT-v2 (*selected)* (Sarhan, Layeghy, & Portmann, 2022c)	Lab generated from TON_IoT pcap	63.99% (9 classes)	16,940,496	2022	43 NetFlow-based
NF-ToN-IoT (Sarhan et al., 2021)	Lab generated from TON_IoT pcap	80.4% (9 classes)	1,379,274	2020	12 NetFlow-based
NF-UQ-NIDS-v2 (Sarhan, Layeghy, & Portmann, 2022c)	Integration of NF v2 versions of BoT-IoT, ToN-IoT, UNSW-NB15, CIC-IDS2018	66.88% (9 classes)	75,987,976	2022	43 NetFlow-based
Edge-IIoTset (Ferrag et al., 2022)	IoT/IIoT sensor testbed IP-based, MQTT, Modbus	86.68% (14 classes)	20,952,648	2021-2022	63 Protocol-specific, custom
TON_IoT (ToN-IoT) (Alsaedi et al., 2020)	IoT/IIoT network, telemetry, and logs testbed	34.93% (9 classes)	461,043 (network only)	2019-2020	44 Flow-based
WUSTL-IIoT-2021 (Zolanvari et al., 2021)	IIoT sensor testbed	7.28% (4 classes)	1,194,464	2021	41 Flow-based; TCP-specific
CICIoT2023 (Neto et al., 2023)	IoT testbed (105 devices)	97.65% (33 classes, across 7 families)	46,686,579	2023	47 Flow-based; protocol-specific; custom

Ref: Primary citations above and (Alex et al., 2023; Keersmaeker et al., 2023; Sarhan, 2022)

4.3 Exploratory Data Analysis and Data Preprocessing Results

Exploratory Data Analysis (EDA) for the NF-ToN-IoT-v2 public dataset under study focused on understanding the dataset characteristics to proceed with data

preparation and preprocessing, as described in §3.3 and Figure 3-1. The EDA process uncovered no nulls (not a number (NaN), dashes, infinity values); missing values; or missing labels; and only a small number of duplicate records which required data cleansing. The NF-ToN-IoT-v2 is a tabular dataset of 16,940,496 records (rows) captured with 43 Cisco NetFlow v9 format feature columns, a binary classification target "Label" column (benign, labeled with a 0 value, or malicious, labeled with a 1 value), and multiclass "Attack" target class label column. Table 4-2 shows the overall traffic profile by the 10 cyberattack target classes. Table 4-2 shows imbalance across the classes, which is typical for a realistic network. For the binary classification "Label" target, the class distribution is 10,841,027 (63.9%) malicious and 6,099,469 (36.01%) benign.

Table 4-2. Praxis Dataset NF-ToN-IoT-v2 Traffic Profile

Class	Flows	Percent	Description
Benign	6,099,469	36.01	Normal network flow
Backdoor	16,809	0.10	Attack on remote access devices by replying to specific constructed client applications0
DoS	712,609	4.21	Denial of service: attempt to overload a device's resources, denying access to its data and services
DDoS	2,026,234	11.96	Distributed Denial of Service: DoS attack from multiple distributed sources
Injection	684,465	4.04	Type of attack that supplies untrusted input aiming to alter execution (e.g., code, SQL)
MITM	7,723	0.05	Man in the Middle: attacker positions between a target and host to intercept and/or alter traffic
Password	1,153,323	6.81	Attack aiming to retrieve passwords by brute force, sniffing, or other means
Ransomware	3,425	0.02	Attack that encrypts data and/or lock out access on a victim host and asks for compensation in exchange for decrypting data or unlocking host
Scanning	3,781,419	22.32	Techniques aiming to discover information about networks, hosts, or applications
XSS	2,455,020	14.99	Cross-site scripting: injection where an attacker uses web application to send malicious scripts
	16,940,496	100.00%	

The unprocessed NF-ToN-IoT-v2 dataset consists of 43 standardized NetFlow features, a “Label” binary classification target column, and an “Attack” multiclass classification target column. The features consist of network addresses; port numbers; protocol-specific identifiers, attributes, and control or error codes; and many flow measures (length, duration, throughput). The total dataset includes flows between 29,245 unique IPv4 hosts, using both private and public IPv4 addresses, and 65,536 unique port numbers that are a combination of well-known, registered, and dynamic values. Traffic uses six protocols: TCP, UDP, ICMP, IGMP, IPv6-ICMP, and IPv6-Fragmentation. Figure A-1 in the appendix provides a full description of all 43 features and target labels.

In network security operations practice, the source and destination IP addresses and port numbers are monitored using a rules-based approach to alert for unknown addresses or port numbers or unusual patterns of usage, potentially corresponding to unauthorized application or user behavior. This tuple forms a flow identifier specific to the network being monitored. If left in for the downstream steps, these could introduce bias leading to overfitting, since they identify the specific attacker and victim endpoints. Thus, source and destination IP addresses (IPV4_SRC_ADDR, IPV4_DST_ADDR) and source and destination TCP or UDP port number (L4_SRC_PORT, L4_DST_PORT) are dropped from the dataset, leaving 39 out of the original 43 features for modeling . This is also consistent with the existing benchmarks discussed in §3.5 and §4.2.

Examining the statistics of each feature (column) using Pandas DataFrame describe() method yields their mean, standard deviation, minimum, 25%, 50%, 75%, and maximum percentile values. Some features, including protocol-specific features such as L7_PROTO or MIN_TTL that have a specified set of or range of values, stay in a narrow

range. By contrast, many traffic-related features do not necessarily have a prescribed range bound. SRC_TO_DST_SECOND_BYTES and DST_TO_SRC_SECOND_BYTES exhibit extreme ranges with values up to 8.91×10^{304} and 1.87×10^{257}, respectively. Further examination shows that less than 25 instances have values above 10^{100}. Notably, their 75% percentile values are 328 and 234, respectively, indicating that a relatively small number of samples have extreme values. While these should correspond to sudden bursts of traffic, values of this extreme may instead be from sensor saturation and are dropped as outliers to prevent computation overflows. Broad ranges exist up to 3.02×10^{8} for byte count (IN_BYTES, OUT_BYTES) and up to 4.29×10^{9} for average throughput features, where extreme values represent a small percentage of the total. As described in §3.3, a feature set with widely varying ranges and distribution of values should be transformed to prevent weighting a feature simply due an extremely large instance value. This praxis applied Min-Max scaling, which maps each value X of the dataset ranging from an observed minimum value X_{min} to an observed maximum value X_{max} into a new value X' in the range 0 to 1, according to the transformation below:

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{20}$$

An alternative such as standard or Z-score scaling can overly shrink the range of feature values given outliers and is better suited for normally distributed data. The majority of published benchmarks also use Min-Max scaling, which aids comparison.

After dropping the source and destination IP addresses and ports as described above, only numerical features remain. Thus, no categorical encoding of the remaining 39 features is required. One hot encoding was applied to the “Attack” target variable for

PyTorch deep neural network multiclass classification, expanding the single “Attack” variable to a set of binary variables for each attack class. Label encoding was used for conventional ML classifiers and TabNet multiclass classification.

To support practical feature selection, classification modeling, and traffic class prediction, samples 12.5%, 25%, 50%, and 75% of the overall 16 million record NF-ToN-IoT-V2 dataset were created. Stratified sampling preserved the statistical distribution of records across the “Attack” target classes. Except where noted otherwise, experimentation was conducted using the 25% (over 4 million record) sample.

4.4 Feature Engineering and Selection Results

4.4.1 Trust-based Feature Engineering and Correlation

Following the approach outlined in §3.4.1, the analysis of existing IoT trust literature yielded a mapping of established IoT constructs and metrics for network, system, and application quality of service (QoS), social IoT, and data quality of trust to the attributes available in the praxis dataset. Table 4-3 summarizes the driving IoT trust concepts relevant to assessing the complete standard NetFlow features of the NF-ToN-IoT-v2 dataset (Figure A-1) and the resulting initial trust-based NetFlow feature selection map. Trust attributes associated with frequency or duration of communication or service interaction; cooperativeness, response time, timeliness, latency, and availability; and throughput are most relevant to the features available in the praxis dataset that describe the QoS measured through node traffic volume--whether by byte or packet counts, data rate or throughput, or flow duration. These should also include the volume of small packet transmissions that are commonly used for synchronize (SYN) flood and related attacks. Applying additional network operations domain knowledge also draws in

features associated with controlling QoS, including time-to-live (TTL), allowable maximum TCP window sizes, and TCP flags associated with connection state.

Table 4-3. Initial Trust-based NetFlow Feature Selection Mapping

Established Trust Attributes	Initial Trust-based NetFlow Feature Selection Map
Quality of Service[1] **(QoS)** • **Frequency of communication*** • **Duration of communication*** • **Response time or latency*** • **Throughput*** • **Availability*** • Energy *(not a network attribute)* **Social**[2] • **Credibility and temporal: Cooperativeness-Frequency-Duration*** • Relationship: Co-location, Co-work • Credibility-Reward • Spatial: Mutuality and Centrality (relative to friends) **Data**[3] • **Timeliness*: currency, volatility** • Accuracy • Completeness • Intrinsic value, accessibility, contextual representation	**Primary – QoS Volume** IN_BYTES, OUT_BYTES IN_PKTS, OUT_PKTS FLOW_DURATION_MILLISECONDS DURATION_IN, DURATION_OUT SRC_TO_DST_AVG_THROUGHPUT DST_TO_SRC_AVG_THROUGHPUT SRC_TO_DST_SECOND_BYTES DST_TO_SRC_SECOND_BYTES LONGEST_FLOW_PKT SHORTEST_FLOW_PKT MAX_IP_PKT_LEN, MIN_IP_PKT_LEN NUM_PKTS_UP_TO_128_BYTES **Primary – QoS Control** MIN_TTL, MAX_TTL DNS_TTL_ANSWER TCP_WIN_MAX_IN, TCP_WIN_MAX_OUT TCP_FLAGS CLIENT_TCP_FLAGS SERVER_TCP_FLAGS **Secondary – QoS Errors** RETRANSMITTED_IN_BYTES RETRANSMITTED_IN_PKTS RETRANSMITTED_OUT_BYTES RETRANSMITTED_OUT_PKTS

Bold* indicates relevant to developing trust-based feature selection map

Established trust attributes derived from 1: (Alani, 2023; Alhandi et al., 2023; Ma et al., 2021); 2: (Jayasinghe et al., 2019; Nitti et al., 2014); 3: (Byabazaire et al., 2020)

Table 4-3 also includes an additional set of QoS error features—namely those measuring traffic flow with retransmissions—that may also be predictors of anomalous behavior but are inconsistent in the way they are manifested in network operation. These are marked as candidate secondary QoS features to be selected, and they were

investigated as part of the feature selection experiment step. Figure A-1 in Appendix A shows the complete trust-based mapping results for all dataset features, together with the NetFlow definition of each feature.

The initial candidate trust-based feature set in Table 4-3 includes features that are conceptually correlated; that is, they are different measures for the same or similar underlying phenomena, such as byte counts, packet counts, or durations for a specific flow sample. Other features, including many of the protocol indicators or error flags are subsets or supersets of other features. Their use (or misuse by an attacker) in an operational network may vary, so this praxis computed correlation measures and executed models with and without specific features to identify redundant features to drop. Inclusion of redundant features may bias classification by artificially providing more weight to the underlying attribute(s) associated with the features and unnecessarily increase the dimensionality of models, requiring more computational and storage

Table 4-4. Features with Significant Pairwise Pearson's Correlation

Feature Pairs		Correlation
ICMP_TYPE	ICMP_IPV4_TYPE	1.000000
MAX_TTL	MIN_TTL	0.999499
RETRANSMITTED_IN_PKTS	RETRANSMITTED_IN_BYTES	0.962541
RETRANSMITTED_OUT_PKTS	RETRANSMITTED_OUT_BYTES	0.961617
SERVER_TCP_FLAGS	TCP_FLAGS	0.953433
NUM_PKTS_512_TO_1024_BYTES	NUM_PKTS_256_TO_512_BYTES	0.922458
DURATION_OUT	DURATION_IN	0.904386
OUT_BYTES	NUM_PKTS_1024_TO_1514_BYTES	0.899450
CLIENT_TCP_FLAGS	TCP_WIN_MAX_IN	0.871712
DNS_QUERY_ID	PROTOCOL	0.825511
MIN_TTL	TCP_WIN_MAX_IN	0.807831
MAX_TTL	TCP_WIN_MAX_IN	0.806748

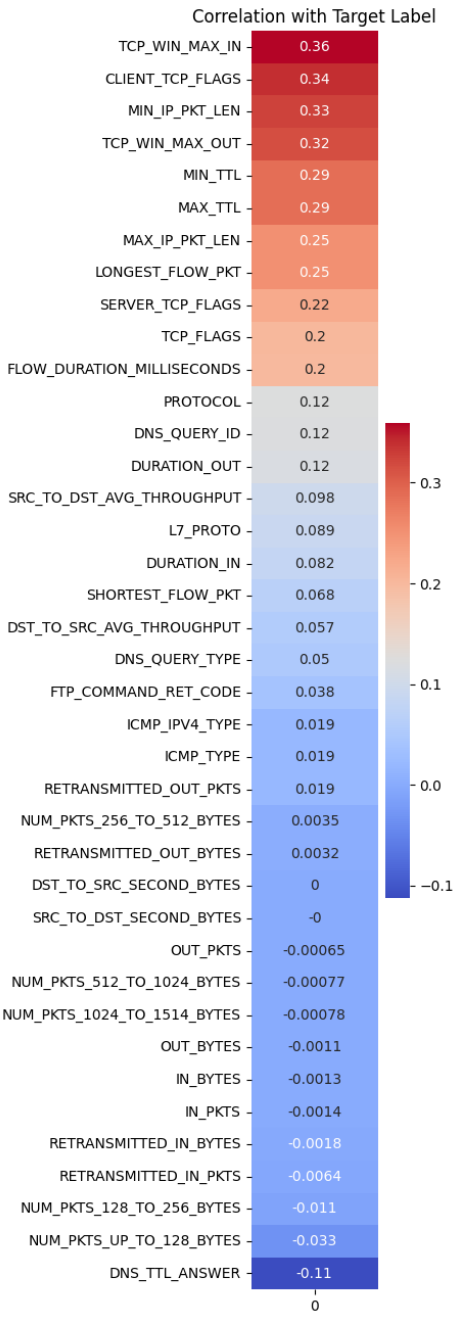


Figure 4-1. Feature Correlation with Binary Classification Target

resources while not improving model learning. Per Equation 4b, Pearson's Correlation Coefficient values near 1 indicate strong positive correlation, values near –1 indicate strong negative (opposite of expected result), and values near 0 indicate no correlation. Features that exhibited pairwise Pearson's correlation value over 80% are shown in Table 4-4. Pairwise Pearson's correlation for each individual feature variable against the binary classification target variable "Label" (see Figure 4-1) provides a simple test of any potential strong univariate predictors. The strongest correlation—for TCP_WIN_MAX_IN—is only 0.36 and not sufficient as a standalone selection factor. The relative order and clustering of strongly uncorrelated features indicates that further feature selection methods will be required.

4.4.2 Filter Methods Results

Univariate filter techniques discussed in §3.4.1 consider the importance of an individual feature as measured by a specific statistical or information theoretic

contribution. These are learning model-agnostic and provide a foundation for further feature selection. Figures 4-2 to 4-4 show the feature importance results for the Chi-squared (χ^2), analysis of variances (ANOVA), and information gain techniques for binary classification. Chi-squared measures statistical dependence between observed samples of an individual feature and the corresponding expected target variable, per Equation 1. The top three Chi-squared measure values— TCP_WIN_MAX,OUT, TCP_WIN_MAX_OUT, and FLOW_DURATION_MILLISECONDS— heavily weight the overall large mean value (13,730.063) of the feature importance set. Only eight features have greater value than the mean value. A large F-statistic for ANOVA reflects variance between the set of observations of an individual feature and expected target variable is greater than the variance among the feature observations themselves. This indicates an individual feature is important to predicting the target variable. Results show

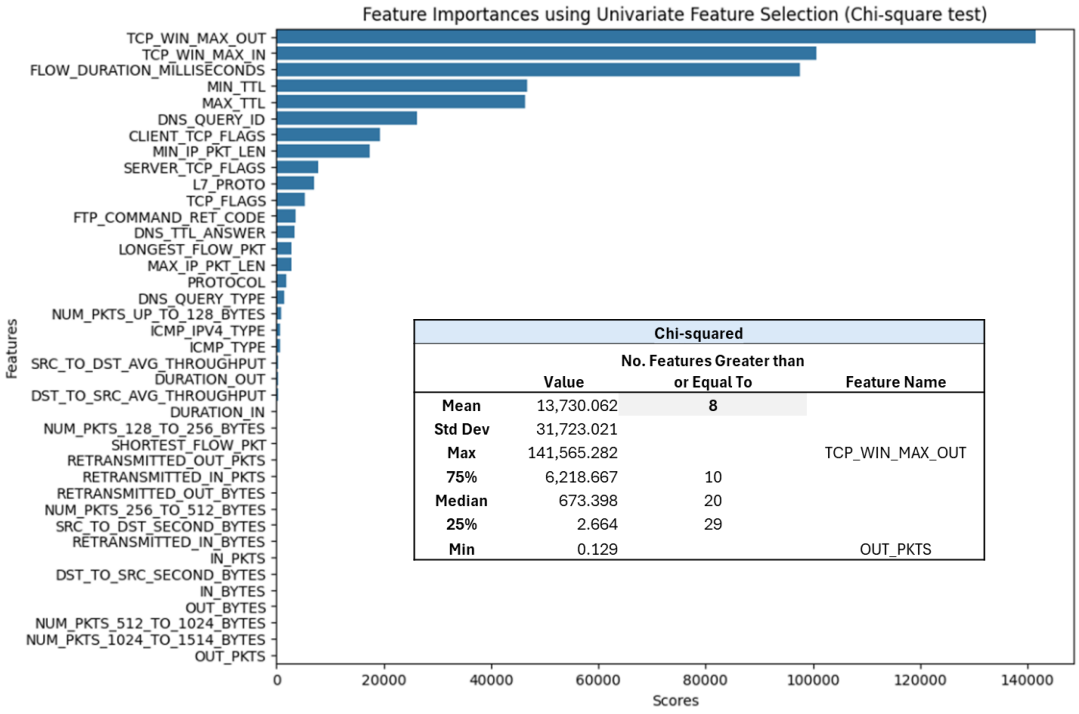


Figure 4-2. Univariate Filter Feature Selection: Chi-squared (χ^2)

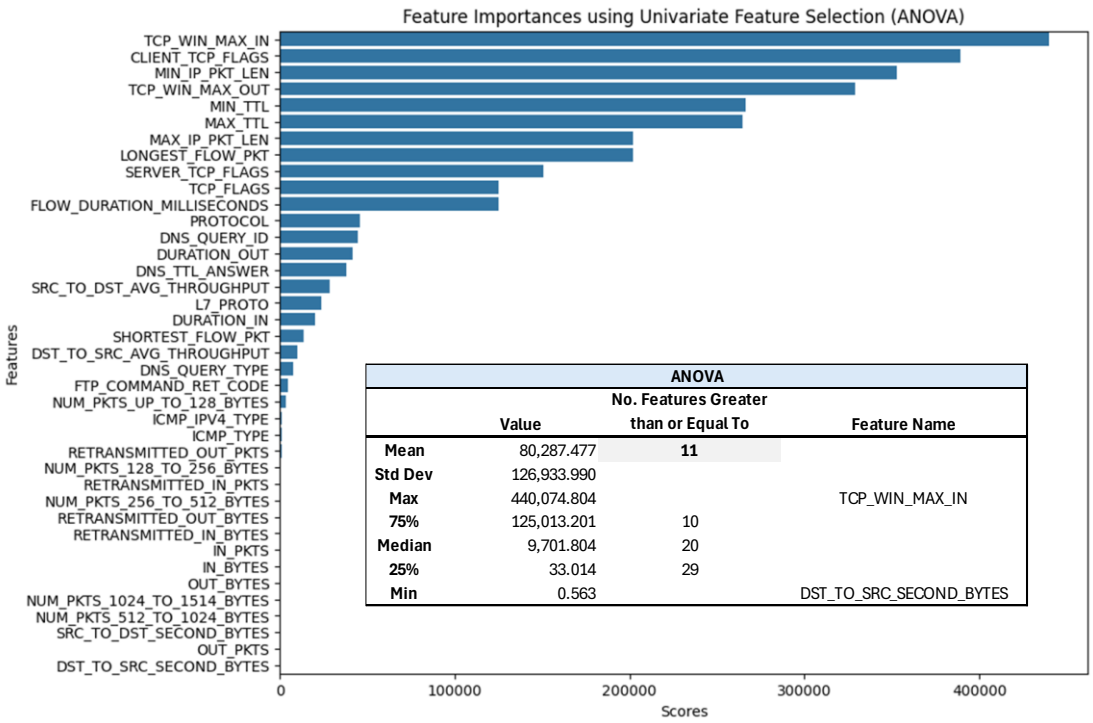


Figure 4.3. Univariate Filter Feature Selection: Analysis of Variance (ANOVA)

significant clustering above the median value but in a more uniform distribution than for Chi-squared. Eleven of the ANOVA feature importance values are above the mean value (80,287.477). The top four features are notably above the rest: TCP_WIN_MAX_IN, CLIENT_TCP_FLAGS, MIN_IP_PKT_LEN, AND TCP_WIN_MAX_OUT.

Information gain, or mutual information, measures the increased contribution of features as reductions in information uncertainty, or entropy, in predicting that target variable The praxis dataset shows a more evenly distributed distribution of information gain values, with sixteen features above the mean value (0.1279), but six top features dominate: TCP_WIN_MAX_IN, IN_BYTES, SRC_TO_DST_AVG_THROUGHPUT, LONGEST_FLOW_PKT, and MAX_IP_PKT_LEN. Notably, information gain brings a greater number of QoS volume trust metrics into the set of important features.

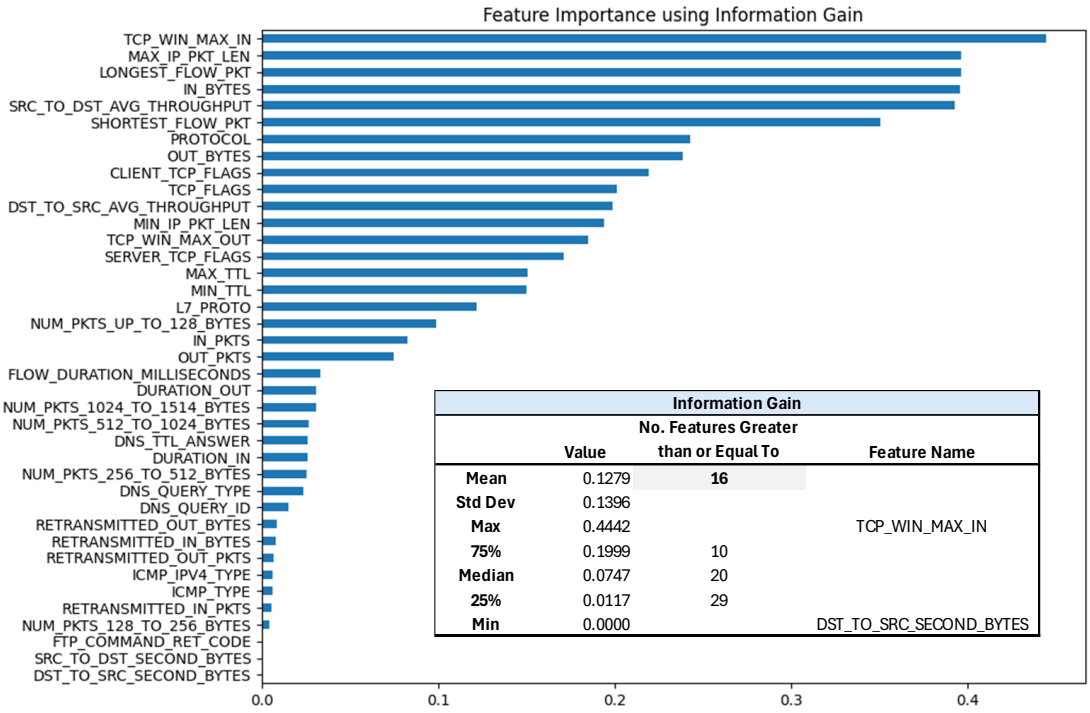


Figure 4.4. Univariate Filter Feature Selection: Information Gain

Chi2	ANOVA	Information Gain
TCP_WIN_MAX_OUT	TCP_WIN_MAX_IN	TCP_WIN_MAX_IN
TCP_WIN_MAX_IN	CLIENT_TCP_FLAGS	IN_BYTES
FLOW_DURATION_MILLISECONDS	MIN_IP_PKT_LEN	SRC_TO_DST_AVG_THROUGHPUT
MIN_TTL	TCP_WIN_MAX_OUT	LONGEST_FLOW_PKT
MAX_TTL	MIN_TTL	MAX_IP_PKT_LEN
DNS_QUERY_ID	MAX_TTL	SHORTEST_FLOW_PKT
CLIENT_TCP_FLAGS	MAX_IP_PKT_LEN	OUT_BYTES
MIN_IP_PKT_LEN	LONGEST_FLOW_PKT	DST_TO_SRC_AVG_THROUGHPUT
SERVER_TCP_FLAGS	SERVER_TCP_FLAGS	CLIENT_TCP_FLAGS
L7_PROTO	TCP_FLAGS	TCP_FLAGS
TCP_FLAGS	FLOW_DURATION_MILLISECONDS	TCP_WIN_MAX_OUT
		MIN_IP_PKT_LEN
		SERVER_TCP_FLAGS
		PROTOCOL
		MAX_TTL
		MIN_TTL

Intersection Above Mean	Union Above Mean	
TCP_WIN_MAX_IN	IN_BYTES	TCP_WIN_MIN_IN
TCP_WIN_MAX_OUT	OUT_BYTES	TCP_WIN_MAX_OUT
MIN_IP_PKT_LEN	FLOW_DURATION_MILLISECONDS	TCP_FLAGS
CLIENT_TCP_FLAGS	DST_TO_SRC_AVG_THROUGHPUT	CLIENT_TCP_FLAGS
MAX_TTL	SRC_TO_DST_AVG_THROUGHPUT	SERVER_TCP_FLAGS
MIN_TTL	LONGEST_FLOW_PKT	MAX_TTL
	SHORTEST_FLOW_PKT	MIN_TTL
	MAX_IP_PKT_LEN	DNS_QUERY_ID
	MIN_IP_PKT_LEN	PROTOCOL

Figure 4-5. Univariate Feature Selection: Combined Results

A combination of the Chi-squared, ANOVA, and information gain results yields a combined set shown in Figure 4-5, which displays features corresponding to the union and the intersection of the above mean values for each method. The intersection of the above mean values suggests a short list of features that are strong individual target predictors. The union of the mean values has significant commonality with the trust QoS volume and QoS control features in Table 4-3, which suggests potential statistical validity to the initial trust-based NetFlow feature mapping.

4.4.3 Embedded Methods Results

Embedded methods provide insight into the importance of each feature in making decisions during a classifier's learning process. Tree-based and boosted tree classifiers as implemented through the Scikit-learn library provide overall model performance feature importance based on decreases in Gini node impurity, as discussed in §3.4.2. Figure 4-6

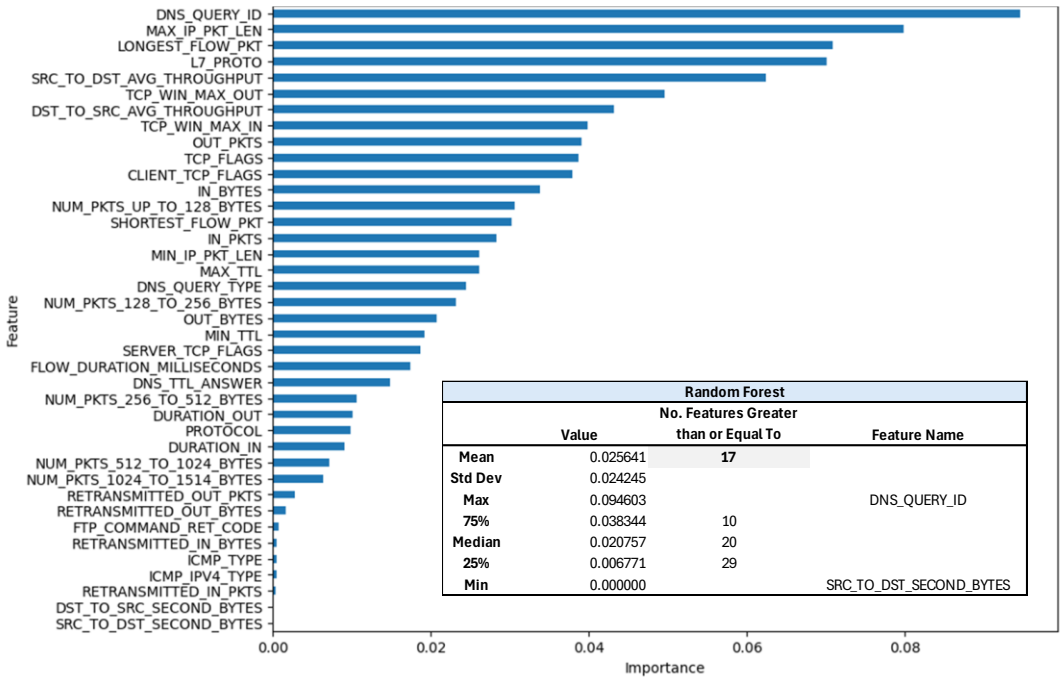


Figure 4-6. Embedded Feature Selection: Random Forest Classifier

shows the overall feature importance rankings for random forest (RF), which is used for the feature selection methods that require a base classifier. Feature importance for the other tree-based and boosted tree ML classifiers used in this praxis show similar distribution patterns. The scale of feature importance values for each classifier is

Decision Tree	Random Forest	Extra Trees
MAX_IP_PKT_LEN	DNS_QUERY_ID	DNS_QUERY_ID
OUT_PKTS	MAX_IP_PKT_LEN	TCP_WIN_MAX_OUT
LONGEST_FLOW_PKT	LONGEST_FLOW_PKT	L7_PROTO
DNS_QUERY_TYPE	L7_PROTO	TCP_WIN_MAX_IN
DNS_QUERY_ID	SRC_TO_DST_AVG_THROUGHPUT	MIN_TTL
L7_PROTO	TCP_WIN_MAX_OUT	MAX_TTL
TCP_FLAGS	DST_TO_SRC_AVG_THROUGHPUT	LONGEST_FLOW_PKT
SRC_TO_DST_AVG_THROUGHPUT	TCP_WIN_MAX_IN	MAX_IP_PKT_LEN
TCP_WIN_MAX_OUT	OUT_PKTS	SRC_TO_DST_AVG_THROUGHPUT
DST_TO_SRC_AVG_THROUGHPUT	TCP_FLAGS	MIN_IP_PKT-LEN
DNS_TTL_ANSWER	CLIENT_TCP_FLAGS	FLOW_DURATION_MILLISECONDS
	IN_BYTES	IN_BYTES
	NUM_PKTS_UP_TO_128_BYTES	TCP_FLAGS
	SHORTEST_FLOW_PKT	CLIENT_TCP_FLAGS
	IN_PKTS	SHORTEST_FLOW_PKT
	MIN_IP_PKT_LEN	SERVER_TCP_FLAGS
	MAX_TTL	
Trees Intersection above Mean	**XGBoost**	**LightGBM**
TCP_WIN_MAX_OUT	LONGEST_FLOW_PKT	L7_PROTO
DNS_QUERY_ID	MAX_TTL	FLOW_DURATION_MILLISECONDS
L7_PROTO	MIN_IP_PKT_LEN	TCP_WIN_MAX_IN
TCP_FLAGS	DNS_TTL_ANSWER	LONGEST_FLOW_PKT
MAX_IP_PKT_LEN	TCP_WIN_MAX_IN	TCP_FLAGS
LONGEST_FLOW_PKT	L7_PROTO	IN_BYTES
SRC_TO_DST_AVG_THROUGHPUT	TCP_WIN_MAX_OUT	CLIENT_TCP_FLAGS
	SHORTEST_FLOW_PKT	SHORTEST_FLOW_PKT
	TCP_FLAGS	DST_TO_SRC_AVG_THROUGHPUT
	FTP_COMMAND_RET_CODE	SRC_TO_DST_AVG_THROUGHPUT
		NUM_PKTS_UP_TO-128_BYTES
		TCP_WIN_MAX_OUT
Boosted Intersection Above Mean	**Union Above Mean**	
LONGEST_FLOW_PKT	IN_BYTES	TCP_WIN_MIN_IN
SHORTEST_FLOW_PKT	OUT_BYTES	TCP_WIN_MAX_OUT
TCP_WIN_MAX_IN	IN_PKTS	TCP_FLAGS
TCP_WIN_MAX_OUT	OUT_PKTS	CLIENT_TCP_FLAGS
L7_PROTO	FLOW_DURATION_MILLISECONDS	SERVER_TCP_FLAGS
TCP_FLAGS	DST_TO_SRC_AVG_THROUGHPUT	MAX_TTL
	SRC_TO_DST_AVG_THROUGHPUT	MIN_TTL
Embedded Intersection Above Mean	LONGEST_FLOW_PKT	DNS_TTL_ANSWER
LONGEST_FLOW_PKT	SHORTEST_FLOW_PKT	DNS_QUERY_ID
TCP_WIN_MAX_OUT	MAX_IP_PKT_LEN	PROTOCOL
L7_PROTO	MIN_IP_PKT_LEN	L7_PROTO
TCP_FLAGS	NUM_PKTS_UP_TO_128_BYTES	FTP_COMMAND_RET_CODE

Figure 4-7. Embedded Feature Selection: Combined Results

independent from one another and cannot be directly compared by value, but the rankings can be examined together. Figure 4-7 shows the union and intersection of the feature importance values above the median from their respective classifier. Feature importance values embedded within classifiers reflect learning progress toward a classification decision outcome, not necessarily only for a specific class. Figure 4-7 shows the union of all features that are above the mean value for at least one of the tree- or boosted-tree algorithms produces 24 features, which reflects modest dimensionality reduction. Also, a few protocol-specific features such as FTP_COMMAND_RET_CODE and DNS_QUERY_ID) are important to only one or two classifiers and may not generalize well. Figure 4-7 also shows the intersection of features appearing above the importance value means for the tree-based algorithms and boosted-tree algorithms, creating a list of candidate strong predictors. Only four features are above the mean for all classifiers: LONGEST_FLOW_PKT, TCP_WIN_MAX_OUT, L7_PROTO, and TCP_FLAGS.

4.4.4 Wrapper Methods Results

This praxis employed recursive feature elimination (RFE) with cross-validation (RFECV) and sequential feature selection (SFS) with a base random forest classifier, as described in §3.4.3. Scikit-learn's RFE algorithm trains the base random forest classifier using all 39 available features and then calculates the feature importance. The least important feature is discarded, and the process repeats with the set of features remaining. RFE continues until it reaches the user-defined number or features, while RFECV performs the RFE process with cross-validation, choosing the set of features to maximize the cross-validation score. Using F1-score as the scoring metric and 5-fold cross-validation for binary classification, RFECV found 15 optimal features (Table 4-5).

Table 4-5. Recursive Feature Elimination with Cross-Validation

RFECV Optimal 15 Feature (not ordered*)	RFECV Non-optimal Features	Rank
L7_PROTO	MIN_TTL	2
TCP_FLAGS	IN_PKTS	3
CLIENT_TCP_FLAGS	SERVER_TCP_FLAGS	4
FLOW_DURATION_MILLISECONDS	OUT_BYTES	5
MAX_TTL	OUT_PKTS	6
LONGEST_FLOW_PKT	DNS_QUERY_ID	7
SHORTEST_FLOW_PKT	IN_BYTES	8
MIN_IP_PKT_LEN	DNS_QUERY_TYPE	9
MAX_IP_PKT_LEN	DURATION_IN	10
SRC_TO_DST_AVG_THROUGHPUT	NUM_PKTS_128_TO_256_BYTES	11
DST_TO_SRC_AVG_THROUGHPUT	RETRANSMITTED_IN_PKTS	12
NUM_PKTS_UP_TO_128_BYTES	NUM_PKTS_256_TO_512_BYTES	13
TCP_WIN_MAX_IN	DURATION_OUT	14
TCP_WIN_MAX_OUT	PROTOCOL	15
DNS_TTL_ANSWER	NUM_PKTS_512_TO_1024_BYTES	16
	NUM_PKTS_1024_TO_1514_BYTES	17
	RETRANSMITTED_IN_BYTES	18
	FTP_COMMAND_RET_CODE	19
**RFECV indexes optimal features as rank 1*	ICMP_TYPE	20
	ICMP_IPV4_TYPE	21
	RETRANSMITTED_OUT_PKTS	22
	RETRANSMITTED_OUT_BYTES	23
	SRC_TO_DST_SECOND_BYTES	24
	DST_TO_SRC_SECOND_BYTES	25

By contrast, SFS does not use feature importance, but rather fits the classifier model with all available features and each iteration searches for the best feature to add (forward selection from an empty selected feature set)) or drop (backward selection from the full feature set). SFS takes considerably longer than RFECV, especially for large feature sets— $n \times k$ for number of features n in an iteration and a k-fold cross-validation—compared to k fits for RFECV. Praxis experiments with forward SFS included multiple runs for different configurations, using the Mlxtend library's SequentialFeatureSelector implementation, which provides greater flexibility for user parameters, metrics capture, and visualization than that of Scikit-learn. For 12 features, forward SFS reached an average F1-score of 0.9998—taking nearly 8 hours with 8 CPU

Forward Sequential Feature Selection (in order selected)
LONGEST_FLOW_PKT
L7_PROTO
TCP_WIN_MAX_IN
FLOW_DURATION_MILLISECONDS
DST_TO_SRC_AVG_THROUGHPUT
SRC_TO_DST_AVG_THROUGHPUT
TCP_WIN_MAX_OUT
CLIENT_TCP_FLAGS

On 12.5% (2M record) stratified sampled dataset, Random Forest classifier with class weighting, 5-fold cross-validation

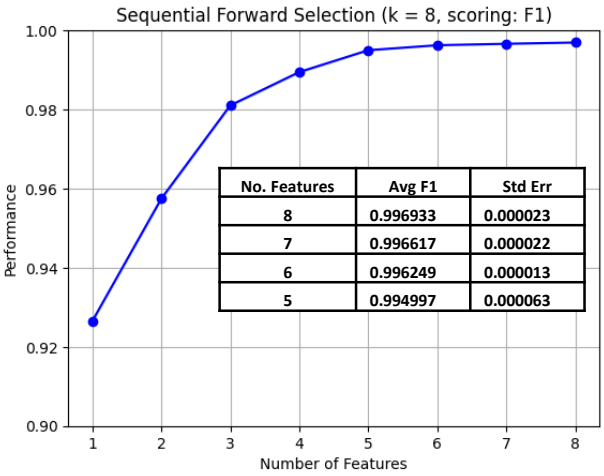


Figure 4-8. Forward Sequential Feature Selection

processors to compute—and continued for over 24 hours to find a minimum set of features within one standard error of the optimum performance. Figure 4-8 shows the key results from a closer examination for a forward SFS computation for the best 8 features, where performance begins to plateau at an F1-score of 0.996249 and standard error 0.000013 for 6 features but continues to incrementally improve. This underscores that practical strong performance can be achieved with many fewer features than the 15 features deemed mathematically optimal by algorithmic solutions such as RFECV.

4.4.5 Overall Feature Selection Results

Research hypothesis 1 (H1) proposes that malicious or otherwise anomalous IoT node behavior should be detected using a primary set of trust-based QoS features, using the initial feature map of Table 4-4. The established feature selection methods explored in this praxis against the NF-ToN-IoT-v2 dataset and analyzed in the preceding subsections of §4.3 support this hypothesis. The rich results reported provide consistent insights toward selecting a unified feature set. The RETRANSMITTED QoS error features identified as candidate secondary trust features had low values in feature importance

across the filter, embedded, and wrapper methods, and will be eliminated from further consideration. The QoS Control initial trust features in Table 4-3 had high values. Variation in which TCP_WIN_MAX, TCP_FLAGS, or TTL feature is included also reflects the pairwise correlation among these, as shown in Table 4-4. The selected feature set should then drop a member of these correlated feature pairs. The results for the QoS Volume initial trust features varied more than for the other categories, but they clearly suggest features measuring long packet flows, long duration, or average throughput. Features measuring instantaneous data rate as source to destination or destination to source bytes per second (BYTES_SECOND) did not rate highly, perhaps hampered by their extreme ranges.

While not derived from trust attributes, L7_PROTO had high feature importance values across many methods and should be a strong candidate for inclusion in the selected feature set. In network security operations practice, attacking reconnaissance, probing, or a suspicious pattern of use of application (layer 7) protocols are often examined as potential indicators of compromise (IoC). DNS_QUERY_ID had high values for random forest and extra trees classifiers only, as seen in Figure 4-7. This field is usually assigned randomly during normal network operations and had low (0.05) univariate correlation to the target variable per Figure 4-1. This may have an insignificant impact on the model learning process and should be considered for joining a set of other protocol-specific features monitored for anomalies using rules or code lookups.

The set of trust-based features selected for detailed classification modeling and performance measurement are then shown in Table 4-6. Table 4-6 offers a core set and a reduced set to reflect the important insight discussed in §4.3.5 on wrapper methods that

practical strong performance plateaus with a smaller feature set much smaller than number required to achieve optimal performance. The focused feature set promises to provide strong classification performance with increased speed (reduced time lapsed) as an alternative suitable for reduced resource utilization deployment.

Table 4-6. Feature Engineering: Selected Core and Focused Trust-based Feature Sets

Core Trust-based Set		Focused Trust-based Feature Set
IN_BYTES	TCP_WIN_MAX_IN	LONGEST_FLOW_PKT
OUT_BYTES	TCP_WIN_MAX_OUT	DST_TO_SRC_AVG_THROUGHPUT
FLOW_DURATION_MILLISECONDS	MAX_TTL	SRC_TO_DST_AVG_THROUGHPUT
DST_TO_SRC_AVG_THROUGHPUT	TCP_FLAGS	TCP_WIN_MAX_OUT
SRC_TO_DST_AVG_THROUGHPUT	CLIENT_TCP_FLAGS	MIN_IP_PKT_LEN
LONGEST_FLOW_PKT	L7_PROTO	TCP_FLAGS
SHORTEST_FLOW_PKT		MAX_TTL
MAX_IP_PKT_LEN		L7_PROTO
MIN_IP_PKT_LEN		
NUM_PKTS_UP_TO_128_BYTES		

4.5 Centralized Conventional Machine Learning Baseline Results

The results of experiments employing centralized conventional machine learning algorithms establish a baseline understanding of the impact trust-based feature selection has on detection and classification performance. The collected results from the six ML classifier algorithms outlined in §3.5 form benchmarks for comparison with deep learning and federated learning algorithms. Gaussian Naïve Bayes (GNB) consistently did not perform as well as the other selected ML algorithms. A variation of the algorithm known as Complement Naïve Bayes (CNB) designed to handle class balance better than GNB or Multinomial Naïve Bayes was then added to the original benchmarks. Developed to improve text classification, CNB computes its statistics from the complement of each existing class (Rennie et al., 2003). To adjust for class imbalance in the praxis dataset, class weighting per Equation 8 was used for tree-based and LightGBM classifiers, which directly support the Scikit-learn API hyperparameter class_weight="balanced". All

experiments used a common random seed value of 9751 for classifiers under test. Default values for other ML classifier hyperparameters produced the baseline results that follow.

4.5.1 Binary Classification Results

This praxis conducted experiments for classifying benign traffic (class 0) versus malicious traffic (class 1) for each of seven ML algorithms using all 39 NetFlow features, the selected core 16 features, and the selected focused 8 features. Table 4-7 shows the binary classification performance using all 39 features. The random forest (RF) ensemble tree classifier exhibited the strongest scores for each of the measures as defined in §3.8: accuracy (Acc), balanced accuracy (bAcc), Matthew's Correlation Coefficient (MCC), precision (pre), detection rate (DR), false alarm rate (FAR), and F1-score (F1). The RF classifier delivered near ideal performance at the cost of the longest training time (383.664 microseconds per sample) and third longest prediction time (20.425 microseconds per sample). Other tree-based classifiers—decision tree (DT) and extra trees (ET)—and boosted tree classifiers—light gradient-boosting machine (LGBM) and extreme gradient boosting (XGB)—delivered similar classification performance with notably faster training and prediction times. The k-nearest neighbor (kNN) classifier with $k = 3$ performed nearly as well as the tree-based and boosted tree classifiers and posted

Table 4-7. Binary Classification Performance with All 39 Features

	Acc	bAcc	MCC	Pre	DR	FAR	F1	Train	Predict
DT	0.99516	0.99477	0.98950	0.99516	0.99516	0.00562	0.99516	36.392	0.562
RF	**0.99620**	**0.99563**	**0.99175**	**0.99620**	**0.99620**	**0.00493**	**0.99620**	383.664	20.425
ET	0.99599	0.99546	0.99129	0.99599	0.99599	0.00507	0.99599	249.188	29.121
LGBM	0.99080	0.99048	0.98005	0.99081	0.99080	0.00983	0.99080	6.358	2.018
XGB	0.99389	0.99269	0.98674	0.99389	0.99389	0.00851	0.99389	4.625	0.593
GNB	0.65214	0.51833	0.13635	0.72414	0.65214	0.61547	0.53061	0.929	0.751
CNB	0.62169	0.67735	0.35676	0.73360	0.62169	0.26698	0.62068	0.326	**0.134**
kNN	0.99469	0.99364	0.98846	0.99469	0.99469	0.00740	0.99468	0.270	4119.869

Training and prediction times per sample (microseconds)

the shortest training time (0.270 microseconds per sample), but at the cost for the longest prediction time by several orders of magnitude (4119.869 microseconds per sample). The kNN classifier trains quickly, but predicts slowly for large datasets, since the classification involves evaluating distances from k neighbors for each test sample point. kNN stores every training sample data point for neighbor distance computations. The resulting data storage and long prediction times can become unwieldy for resource-limited edge IoT/IIoT environments demanding near real-time threat detection.

The Gaussian Naïve Bayes (GNB) and Complement Naïve Bayes (CNB) classifier scores reflect their straightforward implementation and ability to scale to large datasets. They posted the shortest training times after kNN—0.929 and 0.326 microseconds per sample, respectively—and the shortest prediction times—0.751 and 0.134 microseconds per sample. However, these classifiers delivered the worst classification performance by a large margin, only slightly better than random guessing and with high FAR values. Naïve Bayes classifiers assume dataset features are independent of one another. The correlation analysis of §4.4.1 indicates strongly correlated features in the full 39 feature set, invalidating the underlying assumptions.

With the baseline performance with all available 39 features established, examining the classification performance using the core 16 and focused 8 feature sets show the effect of applying the trust-based feature selection process of §4.4. Table 4-8 summarizes core 16 feature set binary classification performance. The receiver operating curves (ROCs)—detection rate (true positive rate) vs. false alarm rate (false positive rate)—and precision-recall (PR) curves in Figures 4-9 and 4-10, respectively, show graphically the relative performance of the classifiers. For GNB and CNB classifiers, the

performance trade-off of higher detection rate for higher false alarm rate and reduced overall performance as measured by the ROC area under the curve (AUC) is discernible in the ROC curves of Figure 4-9. The performance of kNN, the tree-based, and boosted tree classifiers are nearly identical. A PR curve can provide additional insight for imbalanced datasets by showing the precision plotted against recall, or detection rate. A no-skill classifier would predict all samples to belong to the positive "1" (malicious)

Table 4-8. Binary Classification Performance with Core 16 Features

	Acc	bAcc	MCC	Pre	DR	FAR	F1	Train	Predict
DT	0.99609	0.99539	0.99152	0.99609	0.99609	0.00532	0.99609	16.131	0.537
RF	**0.99650**	**0.99576**	**0.99241**	**0.99650**	**0.99650**	**0.00499**	**0.99650**	158.364	14.131
ET	0.99640	0.99566	0.99219	0.99641	0.99640	0.00508	0.99640	163.753	19.656
LGBM	0.99377	0.99257	0.98647	0.99377	0.99377	0.00862	0.99376	4.781	0.511
XGB	0.99009	0.98986	0.97853	0.99011	0.99009	0.01036	0.99010	5.282	1.450
GNB	0.64591	0.50839	0.10054	0.75713	0.64591	0.62913	0.51343	0.419	0.324
CNB	0.71559	0.75588	0.49650	0.78978	0.71559	0.20384	0.71970	0.249	**0.156**
kNN	0.99556	0.99485	0.99037	0.99556	0.99556	0.00587	0.99556	**0.143**	3618.351

Training and prediction times per sample (microseconds)

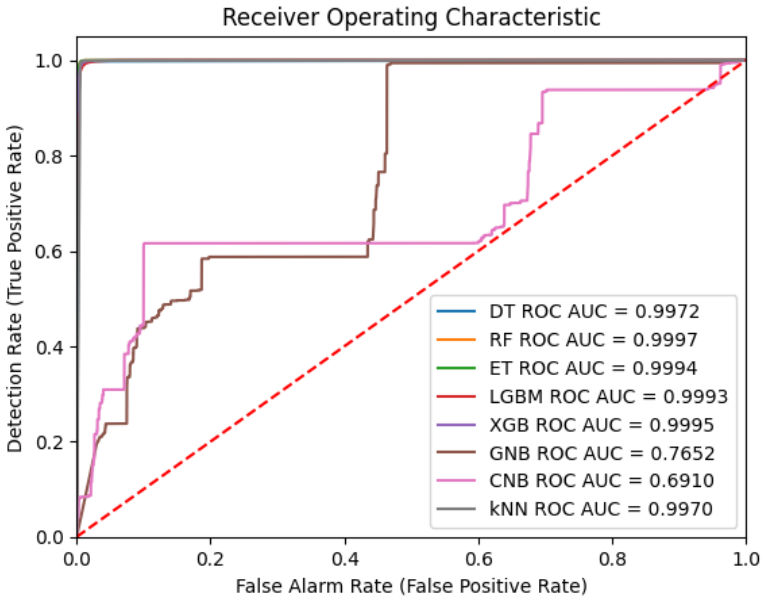


Figure 4-9. Binary Classification with Core 16 Features: Receiver Operating Curves

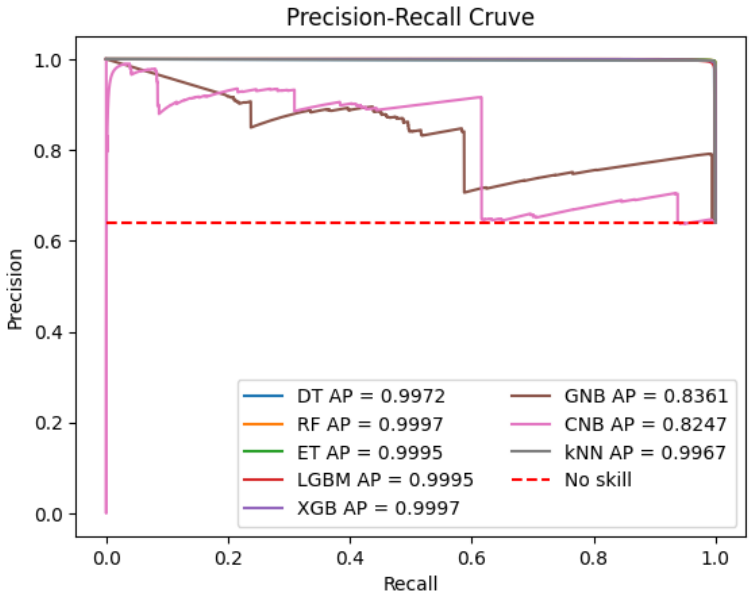


Figure 4-10. Binary Classification with Core 16 Features: Precision-Recall Curve

class, simply reflecting the proportion of malicious samples in the test dataset. The praxis experiments show the binary classification of the tree-based, boosted tree, and kNN classifiers to have curves approaching the upper right-hand corner with performance near the ideal PR AUC, or average precision (AP) of 1. GNB and CNB notably lag in performance, with AP values of 0.8361 and 0.8247, respectively.

Examining the training and prediction times from Table 4-8 with the confusion matrix results in Figure 4-11 reveals that the boosted tree classifiers LightGBM and XGBoost sacrifice a very modest amount of classification performance for considerably improved (greater than 30 times) training and prediction times. Note that the GNB classifier behaves in a manner close to always predicting a sample is a member of the majority class “1” (malicious), underscoring its deficit of practical utility. While the CNB classifier performs notably better than GNB in classifying class “0” benign data, it performs poorly in classifying class “1” malicious data.

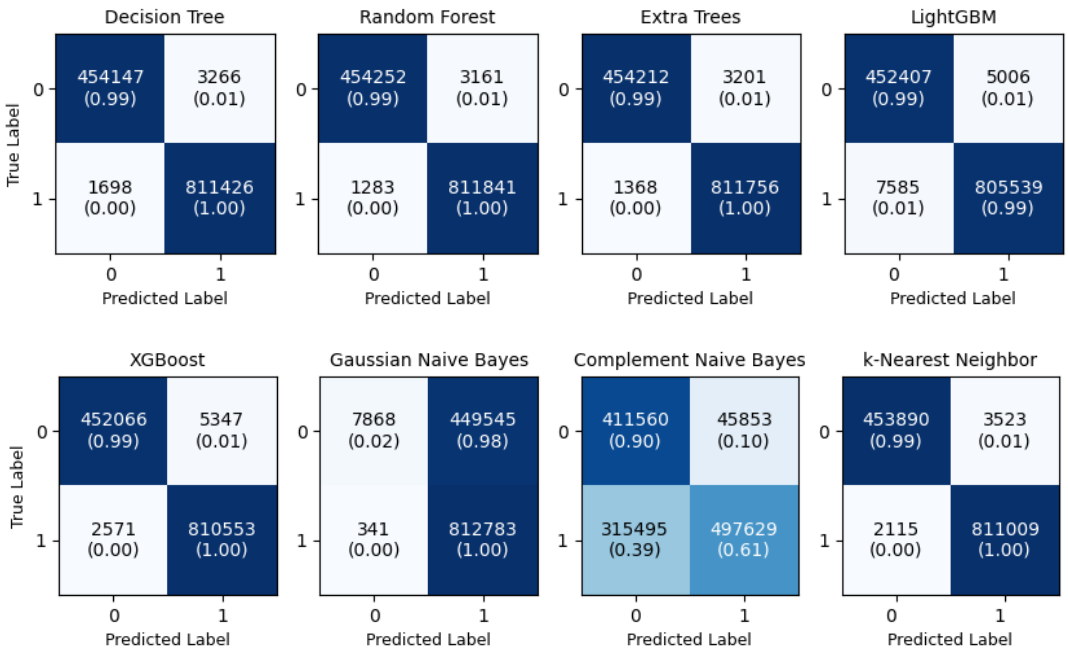


Figure 4-11. Binary Classification Results with Core 16 Features

Five-fold cross-validation assessed possible under- or overfitting of the candidate models. Table 4-9 displays the mean and standard deviation for key performance metrics of the tree-based and boosted tree classifiers. Mean ROC AUC for random forest and XGBoost classifiers were close to ideal 1.000 scores, with small standard deviations on the order of 10^{-5}. Cross-validation results are in a narrow range, boosting confidence in the ability of the core 16 feature set models to generalize well to other data.

Table 4-9. Binary Classification with Core 16 Features: Cross-Validation

		Acc	Pre	DR	F1	AUC
DT	*mean*	0.99595	0.99582	0.99787	0.99684	0.997
	std	8.89E-05	1.17E-04	4.16E-05	6.92E-05	7.37E-05
RF	*mean*	**0.99640**	**0.99600**	**0.99838**	**0.99719**	**1.000**
	std	7.59E-05	9.96E-05	3.12E-05	5.91E-05	2.38E-05
ET	*mean*	0.99629	0.99597	0.99825	0.99711	0.999
	std	9.59E-05	1.37E-04	2.07E-05	7.47E-05	2.37E-05
LGBM	*mean*	0.99068	0.99403	0.99138	0.99271	0.999
	std	2.48E-04	2.97E-04	2.01E-04	1.93E-04	1.57E-05
XGB	*mean*	0.99386	0.99346	0.99697	0.99521	1.000
	std	2.06E-04	2.16E-04	1.33E-04	1.60E-04	1.56E-05

Table 4-10. Binary Classification Performance with Focused 8 Features

	Acc	bAcc	MCC	Pre	DR	FAR	F1	Train	Predict
DT	0.98555	0.98652	0.96890	0.98572	0.98555	0.02186	0.98558	3.783	0.149
RF	**0.98596**	**0.98691**	**0.96977**	**0.98612**	**0.98596**	**0.02124**	**0.98599**	92.886	13.223
ET	0.98584	0.98680	0.96951	0.98600	0.98584	0.02143	0.98587	93.672	18.640
LGBM	0.98000	0.98158	0.95714	0.98037	0.98000	0.03032	0.98006	2.108	0.477
XGB	0.98393	0.98466	0.96539	0.98410	0.98393	0.02423	0.98397	4.106	1.523
GNB	0.60277	0.65611	0.31355	*0.70958*	0.60277	0.60697	0.60199	**0.247**	0.162
CNB	0.57444	0.64235	0.29900	0.71402	0.57444	0.65264	0.56526	0.381	**0.043**
kNN	0.98510	0.98587	0.96789	0.98525	0.98510	0.02250	0.98513	2.942	9249.474

Training and prediction times per sample (microseconds)

To investigate the possibility of reducing the binary classification problem dimensionality even further, the same experiments were repeated using the focused 8 feature set. The summary of the results shown in Table 4-10 reflects strong outcomes that are typically within 1% or less of performance using the core 16 and full 39 feature sets. Of note, the training times for random forest and extra trees decreased by over 40% (each to below 94 microseconds per sample), while nearly maintaining performance, at the expense of greater (approximately four times) false alarm rate. The acceptability of this tradeoff in a practical deployment will be application dependent.

4.5.2 Multiclass Classification Results

The classification and timing performance of ML algorithms for multiclass classification of cyberattack classes showed similar trends as the binary classification results. Classifiers using the core 16 features performed nearly indistinguishably from classifiers using all 39 features. Performance moving to focused 8 features stayed within 1-1.5% across most classes and metrics. Table 4-11 displays summary results for random forest and extra trees for the 39 feature set and Table 4-12 displays summary results for random forest with the core 16 and focused 8 feature sets as representative examples.

Table 4-11. Multiclass Classification: Random Forest and Extra Trees (39 Features)

	Random Forest				Extra Trees			
Attack Class	**Pre**	**DR**	**FAR**	**F1**	**Pre**	**DR**	**FAR**	**F1**
Benign	0.99484	0.99396	0.00290	0.99440	0.99471	0.99381	0.00297	0.99426
Backdoor	0.99761	0.99761	0.00000	0.99761	0.99761	0.99761	0.00000	0.99761
DDoS	0.98757	0.98601	0.00168	0.98679	0.98719	0.98574	0.00173	0.98647
DoS	0.92483	0.88663	0.00316	0.90533	0.91933	0.88705	0.00342	0.90290
Injection	0.92773	0.88054	0.00289	0.90352	0.91719	0.88566	0.00337	0.90115
MitM	0.41855	0.30309	0.00018	0.35158	0.41707	0.31034	0.00019	0.35588
Password	0.96177	0.96329	0.00281	0.96253	0.96648	0.96524	0.00245	0.96586
Ransomware	0.98750	0.95181	0.00000	0.96933	0.98755	0.95582	0.00000	0.97143
Scanning	0.99684	0.99571	0.00091	0.99628	0.99661	0.99581	0.00098	0.99621
XSS	0.93633	0.96592	0.01110	0.95090	0.93998	0.96436	0.01041	0.95201
Weighted Avg	0.97780	0.97785	0.00350	0.97775	0.97784	0.97793	0.00345	0.97783
Accuracy		0.97785				0.97793		
Balanced Acc		0.89246				0.89414		
MCC		0.97151				0.97160		
Train (µs)		279.564				196.762		
Predict (µs)		40.182				51.308		

Table 4-12. Multiclass Classification: Random Forest Core and Focused Features

	Random Forest (Core 16 Features)				Random Forest (Focused 8 Features)			
Attack Class	**Pre**	**DR**	**FAR**	**F1**	**Pre**	**DR**	**FAR**	**F1**
Benign	0.99705	0.99294	0.00165	0.99499	0.97181	0.98897	0.01614	0.98031
Backdoor	0.74524	0.99761	0.00034	0.85315	0.74347	0.99761	0.00034	0.85199
DDoS	0.99540	0.97930	0.00061	0.98728	0.98200	0.97008	0.00241	0.97601
DoS	0.91942	0.90781	0.00349	0.91358	0.91511	0.91049	0.00371	0.91279
Injection	0.94369	0.87951	0.00221	0.91047	0.87638	0.82260	0.00488	0.84864
MitM	0.05012	0.34846	0.00287	0.08763	0.05288	0.34120	0.00265	0.09157
Password	0.96206	0.96193	0.00278	0.96200	0.94933	0.94046	0.00368	0.94487
Ransomware	0.99578	0.94779	0.00000	0.97119	0.99149	0.93574	0.00000	0.96281
Scanning	0.99755	0.99553	0.00070	0.99654	0.99316	0.95477	0.00189	0.97359
XSS	0.94340	0.97111	0.00985	0.95706	0.94033	0.96878	0.01039	0.95434
Weighted Avg	0.98074	0.97817	0.00268	0.97930	0.96485	0.96253	0.00863	0.96345
Accuracy		0.97817				0.96253		
Balanced Acc		0.89820				0.88307		
MCC		0.97195				0.95184		
Train (µs)		132.598				155.561		
Predict (µs)		23.977				22.709		

For each feature set, the tree-based, boosted tree, and kNN classifiers showed strong performance for all classes, except for classification of man-in-the-middle (MitM)

attacks. MitM represented only 0.43% (551 samples) of the test dataset. For the top two overall performers—random forest and extra trees—the respective MitM detection rates were only 0.30309 and 0.31034 with F1-scores of 0.35158 and 0.35588 using all 39 features, as shown in Table 4-11. Using the core 16 features, random forest maintained the MitM detection rate, but significantly dropped its precision and F1-score to 0.05012 and 0.08763, respectively. Inspection of the confusion matrix in Figure 4-12 shows that that the random forest classifier misclassified 55% of the MitM test samples as denial of service (DoS) attacks and 6% as benign traffic. The other classifiers aside from GNB and CNB showed similar performance. Using the core 16 features, GNB performed well

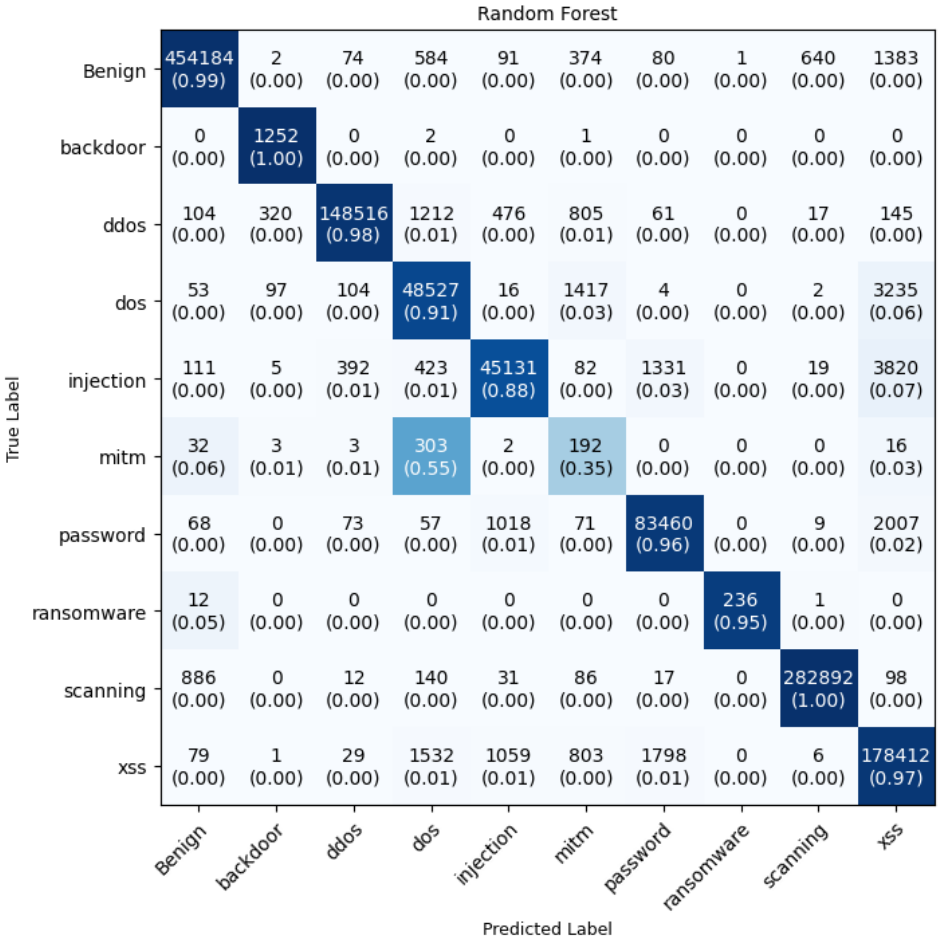


Figure 4-12. Random Forest Multiclass Confusion Matrix with Core 16 Features

(greater than 0.97 detection rate) for classifying backdoor, DoS, and scanning attacks, but mixed performance on other classes resulted in poor overall performance (weighted average F1-score of 0.30285 and MCC of 0.35533). CNB notably scored zero for four classes—backdoor, injection, MitM, and ransomware, largely resulting from systematically classifying most non-benign traffic as cross-site scripting (XSS) attacks.

4.6 Centralized Deep Learning Results

Highly interconnected computing nodes that iteratively discover, learn, and adjust patterns using nonlinear activation and optimization functions enable the expressiveness for which deep learning models are valued. The conventional ML models characterized in §4.5 set a high standard for classification performance and for speed of training and prediction. The well-established multilayer perceptron (MLP) (§4.6.1) provides a performance baseline representative of the deep neural network (DNN) architecture. TabNet (§4.6.2) represents an emerging class of deep generative models that promise to bring a new level of usability and interpretability to applying DL to cybersecurity.

4.6.1 Multilayer Perceptron Deep Neural Network

This praxis investigated the performance of several different MLP DNN architectures and configurations. Table 4-13 summarizes the range of options and the primary configuration used for detailed performance experimentation. This praxis explored configurations with 1, 2, and 3 hidden layers, with the primary tested configuration consisting of 3 fully connected hidden layers of 16, 16, and 12 nodes. The number of input layer nodes was customized to the feature set under test: 39, 16, or 8. For binary classification the output is a single node. For multiclass classification, the output layer consisted of 10 nodes, one for each cyberattack class. Note in the PyTorch

implementation of MLP DNN binary classification, the binary cross-entropy with logits loss (BCEWithLogitsLoss) function includes the sigmoid function activation layer. For multiclass classification, CrossEntropyLoss includes softmax activation.

Table 4-13. Multilayer Perceptron Deep Neural Network Model Configuration

Layer	Nodes	Activation
Input Layer	39 (all features), 16 (core features), or 8 (focused features)	
Hidden layer 1	16 *also explored: 8, 12, 24, 32, 64*	ReLU
Hidden layer 2	16 *also explored: 12, 24, 32, 64*	ReLU
Hidden layer 3	12 *also explored: 6, 8, 16, 32*	ReLU
Output layer	1 (binary classification), 10 multiclass classification)	Sigmoid*, Softmax*
Optimization Function	Adam (learning rate = 0.01) *also explored learning rates from 0.0001 to 0.1 and Stochastic Gradient Descent (SGD) optimizer function*	
Batch Size	128, 512, 1024, 2048, 4096, 8196	

*Output activation layer implemented within PyTorch loss function

Experiments implemented mini-batch gradient descent, where batch size governs the amount of training data in memory and the frequency of model updates during computations, which in turn impact execution time and convergence stability. For some combination of batch size and optimizer function learning rate, the loss function may decrease slowly, oscillate, or even become trapped in local minima, resulting in inconsistent performance. For example, Figure 4-13 shows inconsistent loss and accuracy over 500 epochs (training cycles) for an MLP model using 39 input features performing binary classification with a batch size of 8196 and an optimizer learning rate of 0.1.

Figure 4-14 shows the loss and accuracy history curves for a more representative 39 feature model, with a batch size of 2048. The experiment logs the performance metrics and model weights of each individual epoch and compares the results to the best model

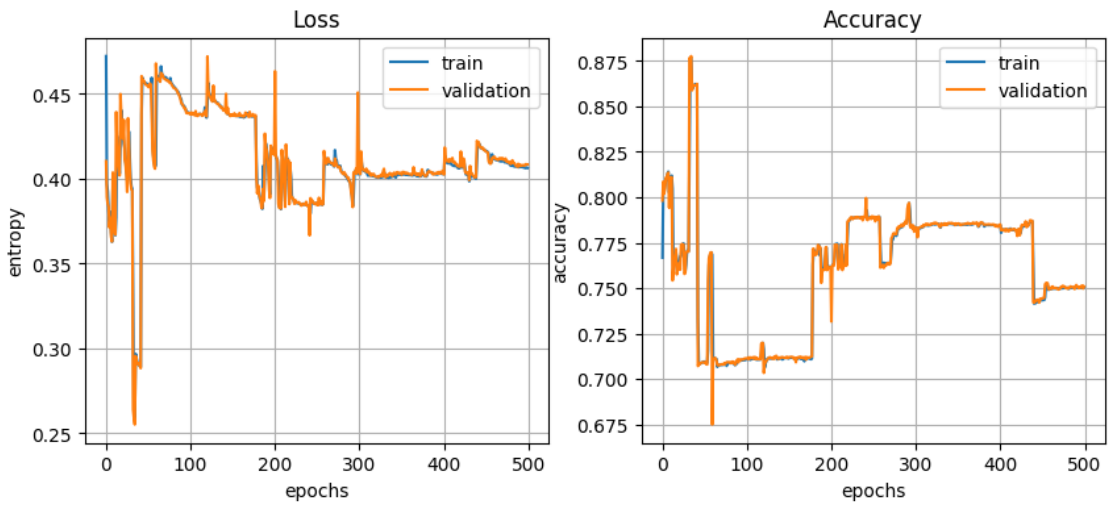


Figure 4-13. MLP Binary Loss and Accuracy: 39 Features, Batch Size 8196

seen to date in the set of training epochs. For the experiment depicted in Figure 4-14, the best recorded accuracy occurs at epoch 189. Table 4-14 shows the overall accuracy, balanced accuracy, and Matthew's Correlation Coefficient and weighted averages for precision, detection rate, false alarm rate, and F1-score across different feature sets. Recorded times reflect training time per training dataset sample for 500 epochs and GPU inference time per test dataset sample, both in microseconds. As seen for ML baseline

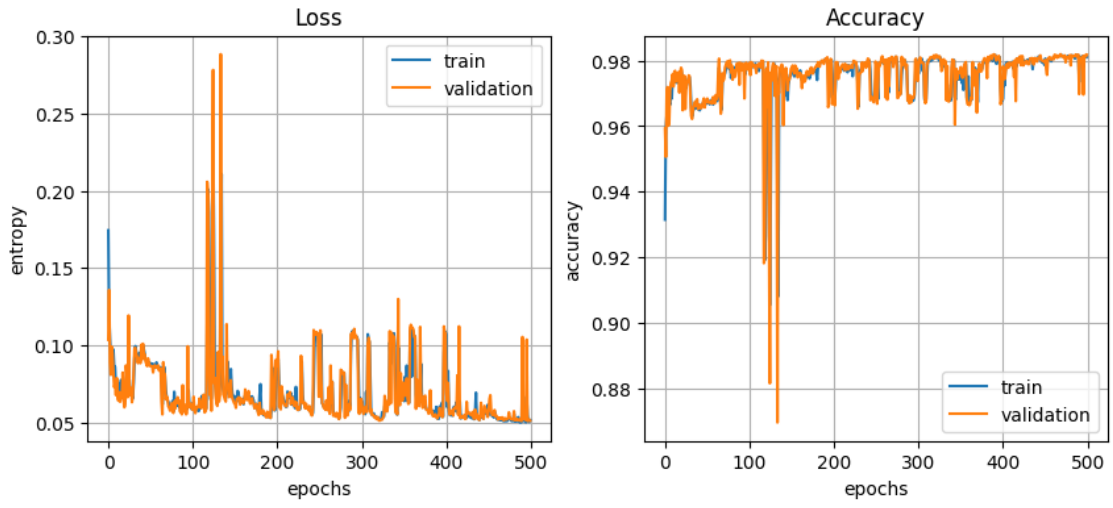


Figure 4-14. MLP Binary Loss and Accuracy: 39 Features, Batch Size 2048

Table 4-14. MLP Binary Classification Performance: Batch Size 2048

	Acc	bAcc	MCC	Pre	DR	FAR	F1	Train	Predict
All 39	0.98192	0.98167	0.96070	0.98201	0.98192	0.01858	0.98194	1421.211	0.236
Core 16	0.98106	0.97945	0.95889	0.98106	0.98106	0.02215	0.98106	1383.361	0.298
Focused 8	0.93170	0.92514	0.85157	0.93161	0.93170	0.08142	0.93165	1373.132	0.259

Training and prediction times per sample (microseconds)

results in §4.5, classification performance for models using the core 16 feature set remained very close to that using the full 39 features. For batch size 2048 shown in table 4-14, performance stayed within 0.0036 on all metrics. Results for model using the focused 8 features set, however showed greater degradation from full 39 feature set model performance (5 or greater percentage points), when compared to the ML baseline degradation of less than 2 percentage points. These results were representative of the behavior for other batch sizes. The batch size 2048 model configuration was also chosen to be the classifier configuration for the federated learning experiments.

Table 4-15. MLP Multiclass Classification: 39 Features

	MLP DNN (batch size: 2048)			
Attack Class	**Pre**	**DR**	**FAR**	**F1**
Benign	0.96931	0.96925	0.01726	0.96928
Backdoor	0.99518	0.98805	0.00000	0.99160
DDoS	0.97173	0.94422	0.00372	0.95778
DoS	0.90923	0.89743	0.00393	0.90329
Injection	0.86703	0.66432	0.00429	0.75226
MitM	0.00000	0.00000	0.00006	0.00000
Password	0.86778	0.91799	0.01025	0.89218
Ransomware	0.00000	0.00000	0.00000	0.00000
Scanning	0.97245	0.97171	0.00793	0.97208
XSS	0.89341	0.95635	0.01929	0.92381
Weighted Avg	0.94515	0.94552	0.01226	0.94457
Accuracy	0.94552			
Balanced Acc	0.73093			
MCC	0.92995			
Train (µs)	1494.202			
Predict (µs)	0.229			

Table 4-16. MLP Multiclass Classification: Core 16 and Focused 8 Features

	MLP Core 16 Features (batch size: 2048)				**MLP Focused 8 Features** (batch size: 2048)			
Attack Class	**Pre**	**DR**	**FAR**	**F1**	**Pre**	**DR**	**FAR**	**F1**
Benign	0.96676	0.96371	0.01864	0.96523	0.92765	0.93009	0.04080	0.92887
Backdoor	0.89080	0.98805	0.00012	0.93691	0.88635	0.98805	0.00013	0.93444
DDoS	0.96260	0.93043	0.00490	0.94624	0.80275	0.93516	0.03115	0.86391
DoS	0.92740	0.87028	0.00299	0.89793	0.87173	0.52433	0.00339	0.65481
Injection	0.74755	0.54930	0.00781	0.63327	0.88435	0.34484	0.00190	0.49619
MitM	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
Password	0.83575	0.89774	0.01293	0.86564	0.80250	0.88625	0.01599	0.84230
Ransomware	0.00000	0.00000	0.00000	0.00000	0.18000	0.03614	0.00003	0.06020
Scanning	0.96830	0.96845	0.00913	0.96838	0.95846	0.91973	0.01148	0.93869
XSS	0.87446	0.95821	0.02325	0.91442	0.81804	0.94904	0.03569	0.87868
Weighted Avg	0.93312	0.93425	0.01402	0.93265	0.89055	0.88689	0.02745	0.88057
Accuracy		0.93425				0.88689		
Balanced Acc		0.71262				0.65136		
MCC		0.91550				0.85501		
Train (µs)		1383.047				1386.148		
Predict (µs)		0.341				0.247		

For multiclass classification, MLP DL models demonstrated the ability to correctly classify most cyberattack classes but exhibited difficulty classifying injection, MitM and ransomware minority cyberattack classes. The gap in performance especially for these three classes is especially significant comparing results across all three feature sets for random forest results in Tables 4-11 and 4-12 to that of the MLP DNN (with batch size 2048) shown in Tables 4-15 and 4-16. Major metrics overall and for individual classes stayed within a percentage point when moving to core 16 features. However, MitM and ransomware performance were effectively zero. From core 16 to focused 8 feature sets (see Table 4-16), major metrics degraded approximately 5 percentage points, with more extreme changes in the classification of specific attacks, such as for DoS (detection rate decrease of 34.5 percentage points) and injection (detection rate decrease of 20.4 percentage points). Curiously, classification of ransomware improved very slightly (detection rate increase of 3.6 percentage points), from a zero baseline.

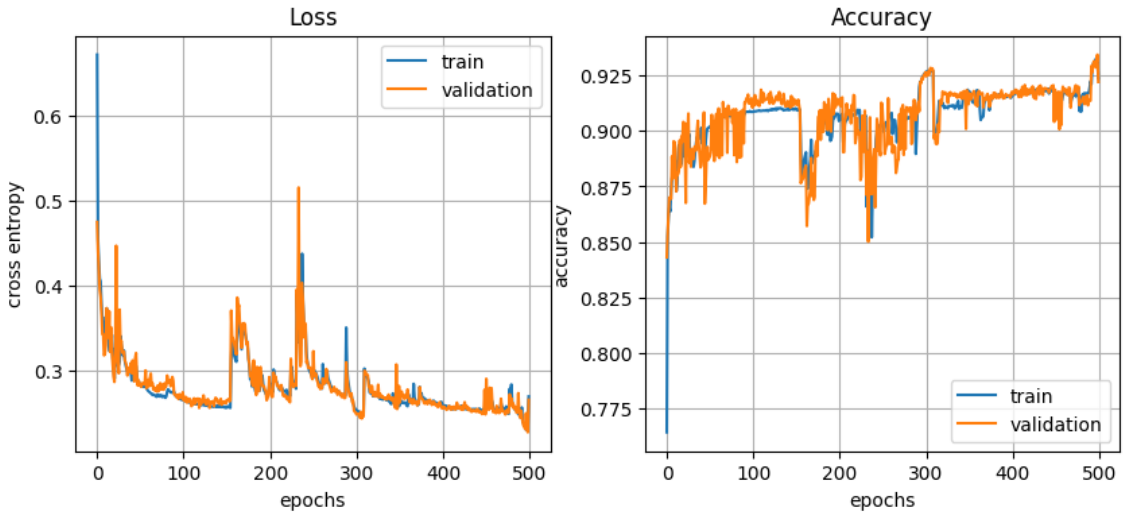


Figure 4-15. MLP Binary Loss and Accuracy: Core 16 Features, Batch Size 2048

Figure 4-15 shows that performance improvement plateaus with some degradation and oscillation after epoch 160, before starting to improve again toward the end of the 500 training epochs. For DL models that converge quickly, continuing many more training epochs can place the model at risk for over- or underfitting.

4.6.2 TabNet

This praxis conducted TabNet generative DL experiments using the open source PyTorch-TabNet library developed by the AI firm DreamQuark. This library allowed TabNet's data preparation, training, and testing to be implemented in the same manner as a standard Scikit-learn ML workflow. PyTorch-TabNet automatically detects and uses a GPU for computation if available, which can simplify implementation and improve modeling efficiency. The library allows researchers to maximum advantage of TabNet's sophisticated architecture (§3.6) by providing over 25 model hyperparameters, additional fit parameters, and customizable evaluation metrics. This praxis used a combination of manual exploration and algorithmic tuning using the open source Optuna hyperparameter

tuning framework. Optuna's primary tool is the Tree Parzen Estimator (TPE) within a Bayesian optimization framework (Akiba et al., 2019). In contrast to a blind exhaustive grid search approach, Optuna conducts a hyperparameter optimization study by iteratively searching over a hyperparameter space guided by a Bayesian probability model that improves as more data points are evaluated. This praxis conducted multiple Optuna optimization study trials. Table 4-17 shows the search range and selected values of the primary hyperparameters of the *opt0* study. The classification performance of the resulting TabNet configuration is reviewed in the following pages.

Table 4-17. TabNet Classifier Optimized Configuration (*opt0*)

Hyperparameter	**Search Range**	**Value**
n_d: *width of decision prediction*	[8, 16]	8
n_a: *mask attention embedding*	[8, 16]	8
n_steps: *steps in architecture*	[3, 5]	3
n_shared: *number shared GLU each step*	[2, 5]	4
optimizer_fn		torch.optim.adam
lr: *optimizer learning rate*	[2e-4, 2e-2]	0.0003
scheduler_fn gamma, step_size	[1e-8, 1], 10	0.8, 10
mask_type	entmax, sparsemax	sparsemax
lambda_sparse: *sparsity loss coefficient*	[1e-3, 3e-3]	0.003
patience: *early stopping epochs*		20
max_epochs		200

While several hyperparameters are like those found for DNN architectures, those associated with configuring the encoder architecture are unique and consequential to TabNet. Notably the mask works with attentive transformer and feature transformer to perform feature selection for each step and instance in a manner that is interpretable as feature importance. Instance-wise feature selection provides more robustness to variations in data, but negatively impacts computation speed. Binary classification weighted average and overall results for TabNet using the *opt0* configuration are

summarized in Table 4-18. Using all 39 features baseline performance for accuracy, precision, detection rate, and F1-score were all over 0.98. This closely mirrors MLP DNN performance, but lags performance of conventional ML models. A higher FAR of 0.02053 also resulted in a lower MCC of 0.95870. TabNet models using the core 16 features differed from corresponding results using all 39 features by only 0.001, producing equivalent performance with a greater than 50% dimensionality reduction. Performance using the focused 8 features degraded only less than another 0.014 in key metrics, markedly better than that for MLP DNN and ML models.

Table 4-18. TabNet Binary Classification Performance

	Acc	bAcc	MCC	Pre	DR	FAR	F1	Train	Predict
All 39	0.98092	0.98019	0.95870	0.98098	0.98092	0.02053	0.98094	4173.911	17.093
Core 16	0.98014	0.97922	0.95698	0.98018	0.98014	0.02170	0.98015	6048.766	17.013
Focused 8	0.96650	0.96899	0.92873	0.96746	0.96650	0.02852	0.96667	2435.998	16.726

Training and prediction times per sample (microseconds)

The loss and accuracy versus training epoch curves for TabNet models using all 39 and the core 16 features (Figures 4-16 and 4-17) reveal an interesting difference. The optimization loss for both models decreased rapidly over the first 15-20 epochs and then only very gradually continued to decrease. The accuracy of the 39 feature model steadily

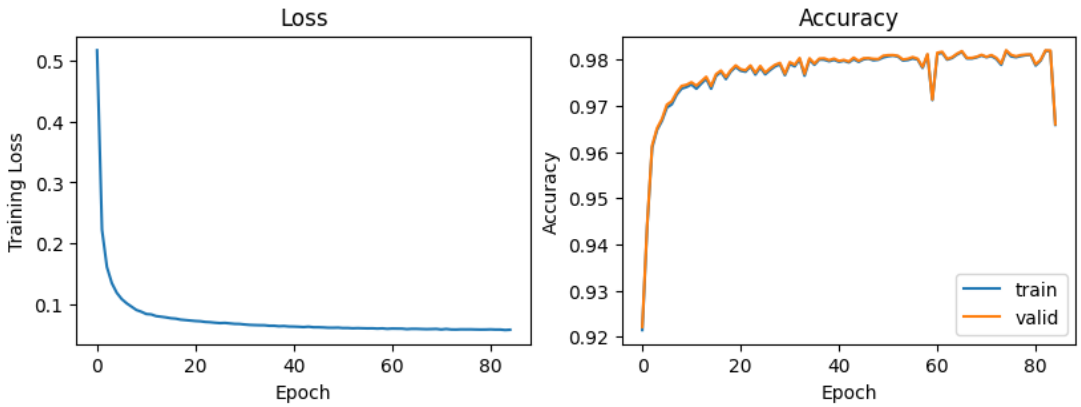


Figure 4-16. TabNet Binary Classification Loss and Accuracy: Full 39 Features

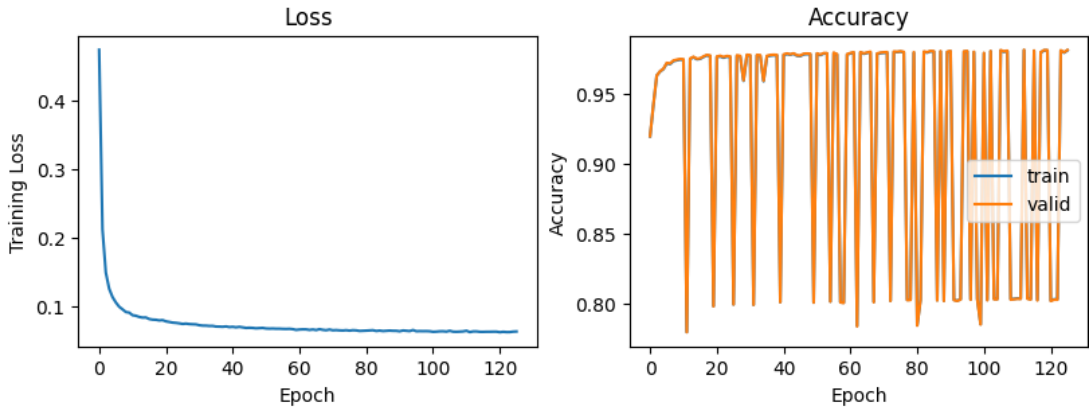


Figure 4-17. TabNet Binary Classification Loss and Accuracy: Core 16 Features

increased with a few dips toward its maximum level over 0.98. The accuracy curve of the core 16 model showed wide oscillations between just below 0.80 and the maximum 0.98 across training epochs after the first 15 epochs. This suggests while the best performance using the core 16 features matches that of the model using the full 39 features, the core 16 feature model is less computationally stable. The model using the focused 8 feature set also showed accuracy oscillations, in the range 0.65 to 0.96.

TabNet's instance-wise feature selection provides model interpretability through local and global feature importance. For binary classification (see Figure A-2), the top 16 global features of the 39 feature set TabNet model have significant overlap with core 16 features set, but they are not identical. The trust-based feature selection process of §3.4 eliminated three protocol-specific features—DNS_QUERY_ID, DNS_TTL_ANSWER, and ICMP_IPV4_TYPE—but they are included in the 39 feature TabNet model's top 16 important features. The core 16 features with the lowest scores also rank low for the full feature set, except for DST_TO_SRC_AVG_THROUGHPUT. Further analysis of the feature distributions and what they reflect in the underlying traffic could provide additional insight into the observed difference in model stability.

TabNet's multiclass classification performance shows a different pattern across models using the three feature sets, as shown in Tables 4-19 and 4-20. Key metrics of TabNet using the core 16 features trailed those of all 39 feature TabNet by 0.025 to 0.045, although FAR remained nearly identical (within 0.002). The TabNet model using the focused 8 feature set performed slightly *better* than that of the core 16 model, except for FAR, where higher false positives for benign traffic decreased overall FAR. Overall, the 39 feature TabNet model slightly trailed (within 0.01) MLP DNN performance on key metrics (see Table 4-16), except balanced accuracy.

Table 4-19. TabNet Multiclass Classification: 39 Features

	TabNet 39 Features			
Attack Class	**Pre**	**DR**	**FAR**	**F1**
Benign	0.98405	0.93848	0.00856	0.96073
Backdoor	0.93011	0.99681	0.00007	0.96231
DDoS	0.96886	0.94598	0.00412	0.95728
DoS	0.88895	0.91307	0.00501	0.90085
Injection	0.66496	0.79460	0.01685	0.72402
MitM	0.02605	0.39201	0.00636	0.04886
Password	0.88471	0.84634	0.00808	0.86510
Ransomware	0.32927	0.97590	0.00039	0.49240
Scanning	0.95462	0.97584	0.01336	0.96512
XSS	0.92939	0.94170	0.01209	0.93550
Weighted Avg	0.94348	0.93485	0.00976	0.93855
Accuracy	0.93485			
Balanced Acc	0.87207			
MCC	0.91695			
Train (µs)	5882.241			
Predict (µs)	16.746			

This aligned with the TabNet model having greater robustness in classifying minority classes. The 39 feature TabNet model's performance for MitM (DR 0.39201), ransomware (DR 0.97590), and injection (DR 0.79460) were notably stronger than corresponding metrics for 39 feature MLP DNN—MitM (DR 0.00000), ransomware (DR

Table 4-20. TabNet Multiclass Classification: Core 16 and Focused 8 Features

	TabNet Core 16 Features				TabNet Focused 8 Features			
Attack Class	Pre	DR	FAR	F1	Pre	DR	FAR	F1
Benign	0.98084	0.91791	0.01009	0.94833	0.95761	0.92183	0.02295	0.93938
Backdoor	0.91116	0.98884	0.00010	0.94841	0.60817	0.99681	0.00064	0.75543
DDoS	0.95628	0.92251	0.00572	0.93909	0.93572	0.88049	0.00820	0.90727
DoS	0.62547	0.91445	0.02405	0.74285	0.88983	0.91494	0.00498	0.90221
Injection	0.57646	0.72091	0.02229	0.64064	0.52745	0.77451	0.02920	0.62754
MitM	0.01561	0.37750	0.01033	0.02997	0.01792	0.38475	0.00915	0.03425
Password	0.83268	0.82029	0.01208	0.82644	0.87565	0.80410	0.00837	0.83835
Ransomware	0.18653	0.96787	0.00083	0.31278	0.12949	0.98394	0.00130	0.22887
Scanning	0.96148	0.96985	0.01119	0.96565	0.95892	0.93018	0.01148	0.94433
XSS	0.88843	0.80158	0.01702	0.84277	0.92208	0.93404	0.01334	0.92802
Weighted Avg	0.91817	0.89833	0.01202	0.90592	0.92342	0.90610	0.01570	0.91324
Accuracy		0.89833				0.90610		
Balanced Acc		0.84017				0.85256		
MCC		0.87151				0.88062		
Train (µs)		4491.491				7037.958		
Predict (µs)		17.411				23.419		

0.00000), and injection (DR 0.66432). Core 16 feature TabNet trailed the corresponding MLP DNN model by 0.04 or less in major measures, except FAR and balanced accuracy due to better minority class performance. The focused 8 feature model, however, outperformed the corresponding MLP DNN model by up to 0.03 in major measures.

4.7 Federated Learning Results

Federated learning (FL) experimentation for the praxis utilized the capability of the Flower framework (discussed in §3.7) to configure and simulate multiple FL workloads and distributed clients using its Virtual Client Engine (VCE). Built on the open-source Ray programming framework for managing and scaling Python workloads, VCE manages the dynamic creation, launch, and termination of clients as they are needed to perform a function in an FL workload within a specified configuration that allows sharing of available CPU, GPU, and memory resources. Flower simulations can be executed within a Jupyter notebook on a single standalone machine or in a cloud

environment such as Google Colab, as used for this praxis, and then be ported to experiments or production environments requiring many physically distributed clients. This praxis employed the same MLP DNN architecture described in §4.6.1 as the core learning algorithm model for both the client and global models of the FL experiment. A dropout layer of 25% was added between layers to attenuate potential overfitting of local client data. To provide the detailed level of FL progress monitoring and performance metrics tracking required, the MLP DNN was implemented using TensorFlow/Keras.

Flower allows customization of not only simulation execution parameters such as the number of federated clients, number of federated rounds, and resource allocations, but also of server strategy. The praxis server strategy was specified through the FedAvg base aggregation scheme, allowing control of the minimum number of clients used each round for fit (praxis: 100% clients), evaluation (praxis: at least 50% clients), availability to start training (praxis: at least 75%), and the algorithm for metrics aggregation (praxis coded weighted averages of major metrics). The server was configured to use the Adam optimizer with learning rate $\alpha = 0.01$ and $\epsilon = 0.0001$, and a loss function of binary cross-entropy or sparse categorical cross-entropy, for binary or multiclass classification, respectively. Each client started with identical configurations: Adam optimizer model with learning rate $\alpha = 0.0001$, 5 local epochs, batch size 2048, $\epsilon = 0.0001$, and a loss function of binary cross-entropy or sparse categorical cross-entropy. Simulations were conducted 50 federation rounds using an allocated pool of GPU and CPU resources.

For each feature set—all 39 features, core 16 selected trust-based features, and focused 8 trust-based features—Flower simulations captured the performance for binary and multiclass cyberattack classification, using 4, 8, or 16 local clients. Each simulation

divided the overall training set into data partitions of equal size (but not even class distribution) for each client in the simulation. Each partition was then divided into local training (80%) and validation (20%) sets for local client training epochs. Model parameter sharing and metrics aggregation between the clients and the server global model proceeded as described in the horizontal federated learning architecture of §3.7.

4.7.1 Binary Classification

Binary classification federated experiments explored the application scenario of detecting (classifying as malicious) a cyberattack within the operating and data domains of individual clients and those within the global domain of the server. Clients may represent distributed local edge machines, network segments, or different organizations. The server updates its global model based on aggregated client model updates.

The distributed and centralized loss curves for models of three different feature sets and three different numbers of clients showed a steady decrease in loss over the first 15 to 25 federation rounds, followed by a divergence in behavior between distributed and centralized loss. In Figure 4-18 the loss curves for all 39 and core 16 feature set experiments using 4 clients exhibit this pattern. This pattern was more pronounced for larger numbers of clients, as seen in Figure 4-19, which displays the loss curves for core 16 feature set models with 8 and 16 clients. The aggregated distributed and centralized global metrics curves for the core 16 feature set 4 client FL experiment are shown in Figure 4-20. Considering these together, it appears that local clients were overfitting their models against their respective local data, whereas the global model continued to improve over additional federation rounds.

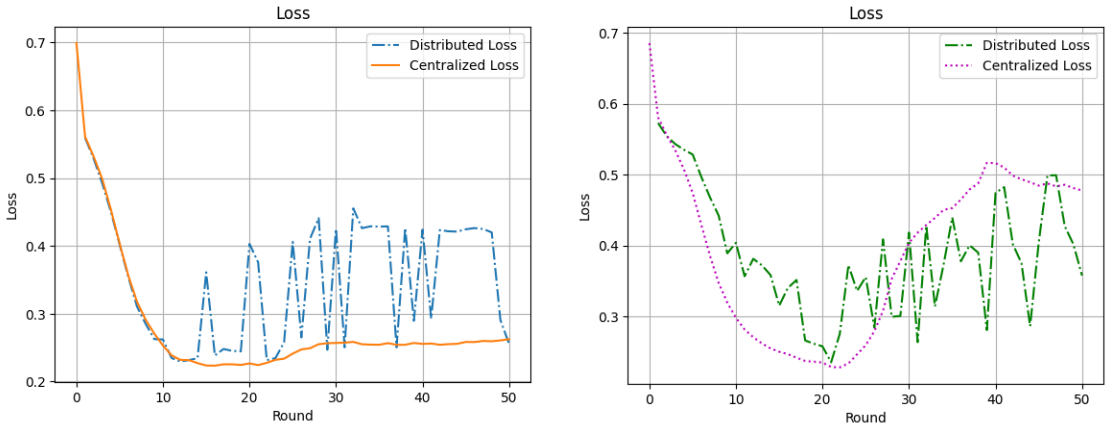


Figure 4-18. Binary Classification Loss: All 39 and Core 16 Features with 4 Clients

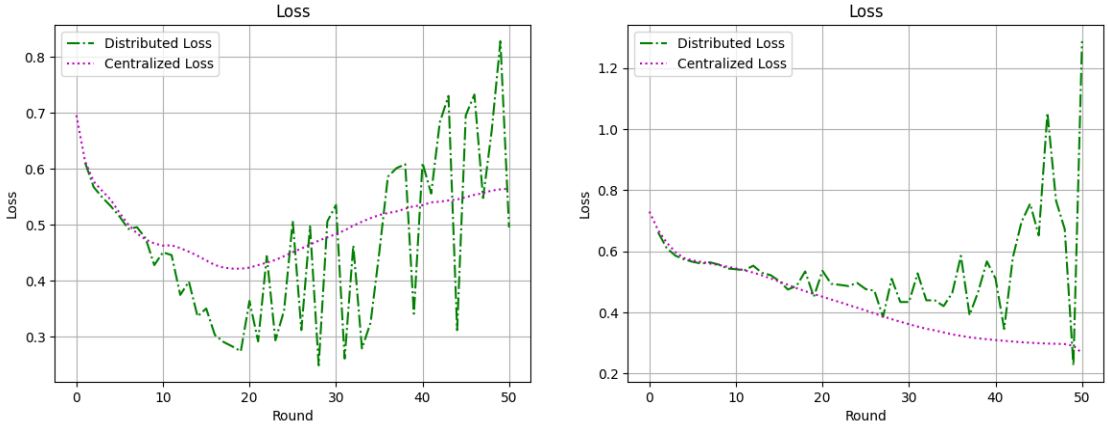


Figure 4-19. Binary Classification Loss: Core 16 Features with 8 and 16 Clients

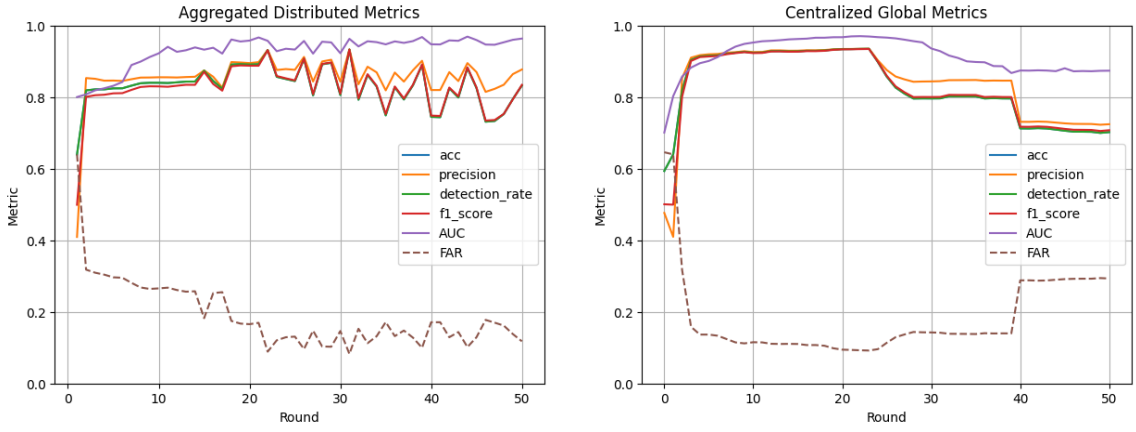


Figure 4-20. Binary Classification Metric Curves: Core 16 Features with 4 Clients

Experiments were structured such that those with a larger number of clients proceeded with a proportionally smaller amount of data per client, since the same total training set is divided among a large number of clients. In several cases, this had the dual impact of client models converging and reaching a point of overfitting earlier, and the global model taking more rounds to converge as the number of clients grew.

Tables 4-21 to 4-23 show the binary classification performance for experiments of the three features sets and three client configurations for federated learning rounds 1, 10, 25, and 50. The progression of scores for feature set models reflected the trends of loss and performance curves (Figures 4-18 to 4-20), where 4 client experiments converged to their best score early (closer to 10 rounds), 8 client experiments converged after a moderate number of rounds (closer to 25 rounds), and 16 client experiments converged after a larger number of rounds (closer to 50 rounds). Best performance was 0.92 to 0.93, for all 39 feature models with 4 and 8 clients (F1-score and ROC AUC in Table 4-21) and for core 16 feature models with 4 and 16 clients (F1-score in Table 4-22). These compare with values of near 0.98 (F1-score in Table 4-14) for centralized MLP DNN.

Table 4-21. Federated Binary Classification with 39 Features

	Round	Acc	Pre	DR	FAR	F1	AUC	MCC
4 Clients	**1**	0.63998	0.40958	0.63998	0.63998	0.49949	0.50000	0.00000
	10	**0.92782**	**0.92760**	**0.92782**	**0.09439**	**0.92744**	**0.91672**	**0.84234**
	25	0.91253	0.91848	0.91253	0.07443	0.91353	0.91905	0.81921
	50	0.88504	0.90074	0.88504	0.08498	0.88697	0.90003	0.77337
8 Clients	**1**	0.63998	0.40958	0.63998	0.63998	0.49949	0.50000	0.00000
	10	0.91285	0.91448	0.91285	0.13084	0.91133	0.89100	0.80982
	25	**0.92684**	**0.92905**	**0.92684**	**0.11444**	**0.92556**	**0.90620**	**0.84126**
	50	0.92435	0.92442	0.92435	0.08769	0.92438	0.91833	0.83597
16 Clients	**1**	0.63998	0.40958	0.63998	0.63998	0.49949	0.50000	0.00000
	10	0.86478	0.86426	0.86478	0.16263	0.86447	0.85107	0.70534
	25	0.79591	0.81964	0.79591	0.17750	0.79972	0.80921	0.59550
	50	0.78766	0.83024	0.78766	0.15885	0.79189	0.81441	0.60373

Table 4-22. Federated Binary Classification with Core 16 Features

	Round	Acc	Pre	DR	FAR	F1	AUC	MCC
4 Clients	1	0.63998	0.40958	0.63998	0.63998	0.49949	0.50000	0.00000
	10	**0.92409**	**0.92565**	**0.92409**	**0.11521**	**0.92290**	**0.90444**	**0.83471**
	25	0.85713	0.87547	0.85713	0.11363	0.85964	0.87175	0.71764
	50	0.70219	0.72424	0.70219	0.29325	0.70750	0.70447	0.39449
8 Clients	1	0.63998	0.40958	0.63998	0.63998	0.49949	0.50000	0.00000
	10	0.76031	0.80929	0.76031	0.18227	0.76496	0.78902	0.55545
	25	0.82122	0.84318	0.82122	0.15027	0.82450	0.83547	0.64651
	50	0.78359	0.83894	0.78359	0.14789	0.78763	0.81785	0.61164
16 Clients	1	0.64009	0.76964	0.64009	0.63980	0.49973	0.50015	0.01364
	10	0.90918	0.91304	0.90918	0.14425	0.90698	0.88247	0.80317
	25	0.81846	0.82973	0.81846	0.17525	0.82101	0.82161	0.62531
	50	**0.93440**	**0.93696**	**0.93440**	**0.10574**	**0.93325**	**0.91433**	**0.85828**

The performance degradation when moving to a focused 8 feature set model seen in centralized MLP DNN classification in §4.6.1 is significantly more pronounced for the federated cases. The best performance metrics for federated focused 8 features are near 0.76 (F1-score and ROC AUC for 4 clients), with most scores even lower near 0.62. The corresponding scores for the centralized case are close to 0.93 as seen in Table 4-14. For all cases, federated FAR was greater than 0.11, which would not be operationally tenable.

Table 4-23. Federated Binary Classification with Focused 8 Features

	Round	Acc	Pre	DR	FAR	F1	AUC	MCC
4 Clients	1	0.63998	0.40958	0.63998	0.63998	0.49949	0.50000	0.00000
	10	**0.75575**	**0.77757**	**0.75575**	**0.22918**	**0.76016**	**0.76329**	**0.50774**
	25	0.60004	0.64526	0.60004	0.37236	0.60787	0.61384	0.21871
	50	0.55854	0.62283	0.55854	0.38770	0.56492	0.58542	0.16574
8 Clients	1	0.63998	0.40958	0.63998	0.63998	0.49949	0.50000	0.00000
	10	0.67202	0.73545	0.67202	0.26443	0.67743	0.70379	0.39380
	25	0.64007	0.66363	0.64007	0.36514	0.64653	0.63746	0.26514
	50	0.46776	0.56928	0.46776	0.41807	0.45751	0.52484	0.05190
16 Clients	1	0.63998	0.40958	0.63998	0.63998	0.49949	0.50000	0.00000
	10	0.62611	0.72157	0.62611	0.27799	0.62782	0.67406	0.34472
	25	0.62483	0.72000	0.62483	0.27960	0.62657	0.67262	0.34181
	50	0.59038	0.62861	0.59038	0.39603	0.59857	0.59718	0.18658

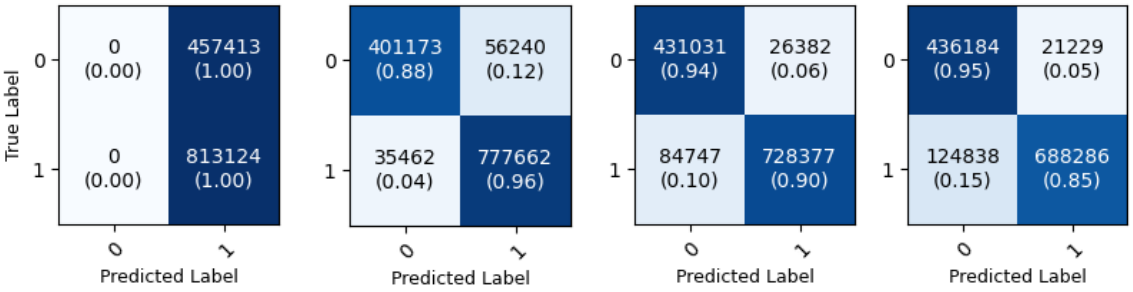


Figure 4-21. Binary Classification Confusion Matrices for 39 Features after Federation Rounds 1, 10, 25, and 50

The confusion matrices of the global model displayed in Figure 4-21 provide additional insight into the evolution of the model over successive federation rounds. The federation process starts with the server requesting the model weights from one client at random to initialize the global model. For each subsequent round the server updates the local models with the latest global weights, and then updates the global model from the aggregated local results while. Figure 4-21 shows that even starting from an extreme initial condition (predicting all samples to be of the majority class "1" (malicious)), the global model evolves quickly over each successive round. While local models execute only 5 epochs each round, the federated aggregation and update process can accelerate model convergence compared to the number of epochs used in a centralized process, such as the 250 to 500 epochs for centralized MLP DNN models discussed in §4.6.1.

4.7.2 Multiclass Classification

Federated multiclass classification exhibited a greater range of outcomes than for binary classification. For models using 39 features (see Table 4-25), best performance F1-scores were from 0.72 to 0.76 (for 8 and 4 clients, respectively), compared to 0.94 for centralized ML DNN (see Table 4-16). Best F1-scores for 16 clients only reached 0.52. Federated multiclass FAR results for 4 and 8 clients significantly improved over those

seen for binary classification. Table 4-24 suggests that slower convergence might be an underlying cause. The performance metrics and loss curves of Figures 4-23 and 4-24 indicate a performance plateau starting with moderate loss levels in early to mid-rounds before round 25. Performance of core 16 feature models (see Table 4-25) showed patterns similar to that of the all 39 feature models. Even with moderate overall performance (F1-score 0.74768 for 4 clients after 50 rounds), the confusion matrix in Figure 4-22 shows that per-class performance is uneven, with performance for classes with a large representation (benign, scanning, XSS, and DDoS) driving overall metrics. Focused 8 feature models, degraded precipitously over core 16 performance—staying below 0.50 (see Table 4-26).

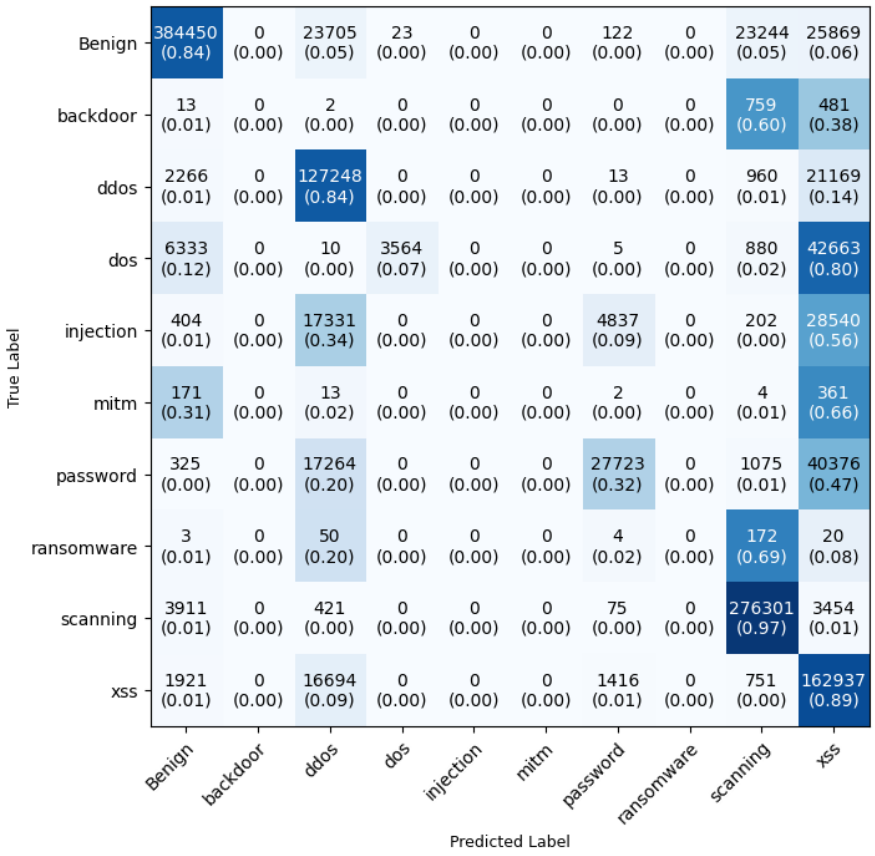


Figure 4-22. Federated Multiclass Confusion Matrix: Core 16 Features with 4 Clients

Table 4-24. Federated Multiclass Classification: 39 Features

	Round	Acc	Pre	DR	FAR	F1	MCC
4 Clients	1	0.42425	0.20845	0.42425	0.29097	0.27618	0.22466
	10	0.61341	0.59561	0.61341	0.12531	0.56945	0.50631
	25	**0.78113**	**0.79198**	**0.78113**	**0.03827**	**0.76119**	**0.72803**
	50	0.77299	0.77557	0.77299	0.04040	0.75611	0.71816
8 Clients	1	0.36001	0.15371	0.36001	0.36002	0.19060	-0.00069
	10	0.48370	0.31538	0.48370	0.17455	0.37273	0.34250
	25	0.72471	0.70342	0.72471	0.04911	0.69549	0.65865
	50	**0.74138**	**0.75877**	**0.74138**	**0.04464**	**0.72321**	**0.68076**
16 Clients	1	0.36002	0.12961	0.36002	0.36002	0.19060	0.00000
	10	0.49653	0.31026	0.49653	0.19806	0.38128	0.34395
	25	0.53901	0.39043	0.53901	0.17385	0.43605	0.41161
	50	0.57374	0.56532	0.57374	0.13784	0.52928	0.45566

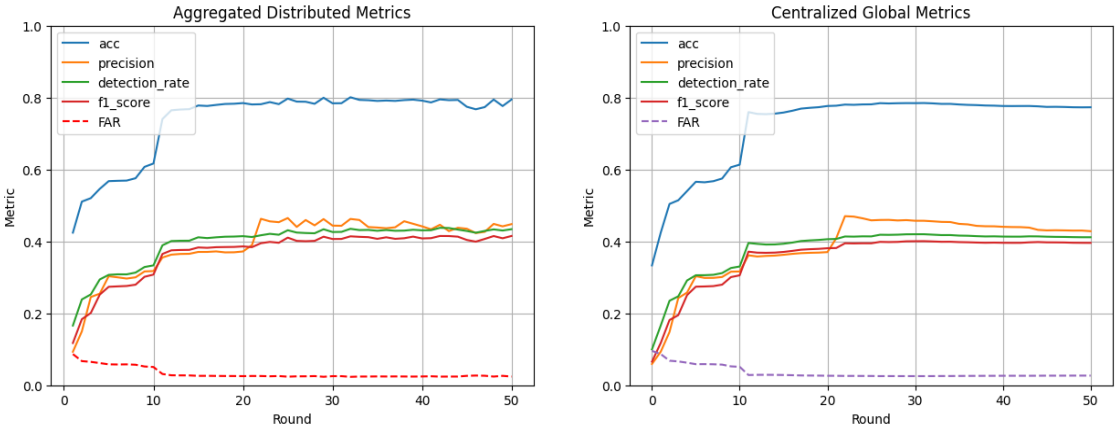
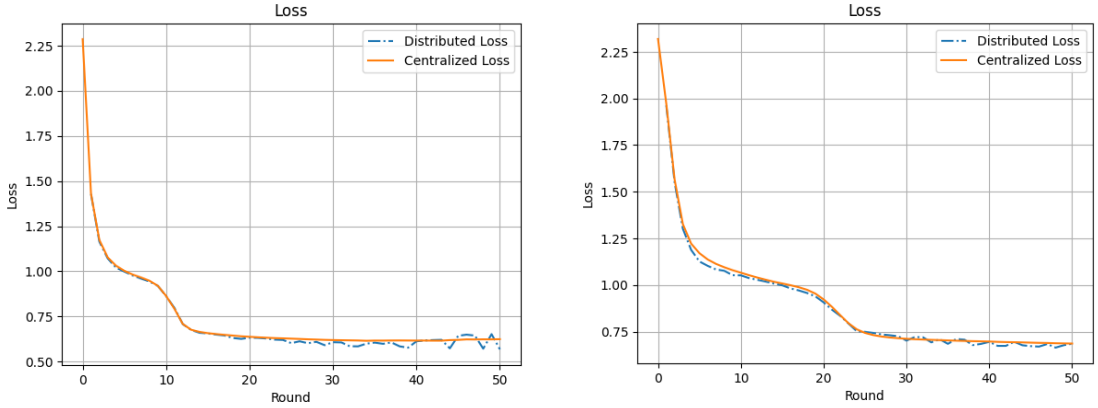


Figure 4-23. Federated Multiclass Performance Curves: 39 Features with 4 Clients



4 Clients 8 Clients

Figure 4-24. Federated Multiclass Classification Loss Curves: 39 Features

Table 4-25. Federated Multiclass Classification: Core 16 Features

	Round	Acc	Pre	DR	FAR	F1	MCC
4 Clients	1	0.42251	0.20703	0.42251	0.29036	0.27495	0.22083
	10	0.67857	0.62216	0.67857	0.10386	0.62921	0.58221
	25	0.69002	0.62585	0.69002	0.10027	0.63967	0.59886
	50	**0.77308**	**0.79362**	**0.77308**	**0.04326**	**0.74768**	**0.71718**
8 Clients	1	0.36002	0.12961	0.36002	0.36002	0.19060	0.00000
	10	0.43251	0.21428	0.43251	0.22708	0.28508	0.25637
	25	0.66273	0.59623	0.66273	0.10508	0.62666	0.55651
	50	**0.67025**	**0.60792**	**0.67025**	**0.10101**	**0.63275**	**0.56853**
16 Clients	1	0.45834	0.23434	0.45834	0.22413	0.30915	0.28957
	10	0.47308	0.28929	0.47308	0.22543	0.35373	0.30386
	25	0.43476	0.27122	0.43476	0.22474	0.29185	0.25924
	50	0.62682	0.49179	0.62682	0.10975	0.53643	0.53261

Table 4-26. Federated Multiclass Classification: Focused 8 Features

	Round	Acc	Pre	DR	FAR	F1	MCC
4 Clients	1	0.36285	0.15369	0.36285	0.32658	0.21126	0.06663
	10	0.45579	0.39224	0.45579	0.24168	0.37023	0.27232
	25	0.45244	0.47085	0.45244	0.25629	0.36917	0.26031
	50	0.45394	0.45387	0.45394	0.25655	0.37466	0.26236
8 Clients	1	0.36545	0.19162	0.36545	0.22739	0.24506	0.15706
	10	0.45058	0.26260	0.45058	0.23202	0.32529	0.27640
	25	0.45870	0.35444	0.45870	0.23879	0.32351	0.29269
	50	0.49259	0.42769	0.49259	0.20802	0.38651	0.34260
16 Clients	1	0.14468	0.17374	0.14468	0.14463	0.03670	0.00233
	10	0.42501	0.21742	0.42501	0.24519	0.27670	0.23625
	25	0.44053	0.36272	0.44053	0.23246	0.32458	0.25667
	50	0.42155	0.29947	0.42155	0.24516	0.28005	0.22869

Experiment results for binary classification show consistent performance trends for FL models using different numbers of clients across three feature sets. This points to an opportunity for deploying FL for threat detection—presence of malicious behavior—at the network edge. Mixed results for multiclass classification do not yet provide a clear picture for FL-based attack classification. Published FL research for IoT applications reviewed in §2.4 point to many open issues, including data distribution, which is a likely factor in the results seen in this section and considered in concluding recommendations.

Chapter 5—Discussion and Conclusions

5.1 Research Hypothesis Findings and Discussion

This praxis investigated the feasibility of identifying trust-based measures of internet of things and industrial internet of things (IoT/IIoT) behavior and the effectiveness of deploying them at the network edge for threat detection and classification using federated machine learning (FL). Three research questions provided the foundation for five hypotheses that guided experimentation. Overall praxis experiments validated the most fundamental hypotheses and partially validated the comparative performance hypotheses, pointing the way to further research (see Table 5-1).

5.1.1 Research Question 1: IoT/IIoT Trust Measures at the Network Edge

RQ1: *What IT/OT network traffic attributes measure trustworthy IoT behavior at the network edge?*

Key underlying issues to address RQ1 are what IoT behaviors affect device and transaction trustworthiness, what attributes associated with trustworthiness are observable; which observable attributes can be measured through network traffic data and metadata collected through normal network security operations; and whether these metrics can be shown to have analytical validity as features appropriate for machine learning (ML) based anomaly threat classification. Feature engineering and selection experiments conducted following the methods outlined in §3.4 and the resulting outcomes reviewed in §4.4 investigated hypothesis 1 (H1), which proposed trust-based ML features drawn from network quality of service (QoS) attributes measurable in network traffic traces.

Table 5-1. Praxis Research Hypothesis Validation Summary

RQ1: IoT/IIoT Trust at the Network Edge	**H1:** *Quality of service attributes can provide trust-based metrics for feature selection to detect and classify IoT/IIoT cyberattacks.* **Results fully support hypothesis: ●**
	Trust-based features from mapping of transaction, social, and data trust measures to network QoS attributes *(§4.4.1, Table 4-3).* Validated by filter, embedded, and wrapper feature selection techniques *(§4.4, Figures 4-1 to 4-8, Tables 4-3 to 4.6)*
RQ2: Machine and Deep Learning Performance	**H2:** *Centralized machine learning classifiers using trust-based metrics can reduce dimensionality by at least 50% while detecting and classifying IoT/IIoT cyberattacks within 1% F1-score performance using standard NetFlow features.* **Results fully support hypothesis: ●**
	Classifiers using core 16 features (59% dimension reduction) were within 1% or better than the F1-score using all 39 features *(§4.5, Tables 4-7 to 4-9; Table 5-2)*
	H3: *Multilayer perceptron DNNs using trust-based features achieve F1-score detection and classification performance of IoT/IIoT cyberattacks within 5% of tree-based machine learning classifiers.* **Results *partially* support hypothesis: ◕**
	MLP DNN using core 16 trust features scored within 5% F1-score for binary and multiclass compared to top performing ML classifier random forest but scored less than 5% below using the focused 8 trust feature set *(§4.6.1; Table 5-3)*
	H4: *TabNet deep learning transformers using trust-based features attain better F1-score detection and classification of IoT/IIoT cyberattacks over multilayer perceptron deep neural networks.* **Results *partially* support hypothesis: ◑**
	TabNet using focused 8 features scored higher than MLP DNN but scored lower using core 16 features. TabNet also provided explainability. *(§4.6.2)*
RQ3: FL at the Network Edge	**H5:** *Edge federated learning can attain IoT/IIoT attack F1-score classification performance of at least 85%.* **Results *partially* support hypothesis: ◑**
	FL binary classification best F1-scores using core 16 features were over 90% for 4 and 16 clients. Best F1-score for multiclass classification was 75% with 4 clients. Focused 8 scores were well below 85%. *(§4.7, Tables 4-21 to 4-26)*

Experiments and analytical mapping provide evidence that fully supports RQ1 and H1. Communication frequency, duration, latency, throughput, availability, credibility, and timeliness are expressed as established classes of trust attributes for digital devices and transactions—quality of service (QoS), social, and data—and mapped into primary and secondary QoS volume and control attributes identified in standard NetFlow v9 network traffic data (see §4.4.1, Table 4-3). Applying established correlation, univariate statistical and information theoretic filter, embedded classifier feature importance, and wrapper recursive feature selection techniques guided forming these attributes into the core 16 and focused 8 trust-based feature sets (see §4.4.1-4.4.6, Figures 4-1 to 4-8, Tables 4-3 to 4-6).

5.1.2 Research Question 2: Machine and Deep Learning Performance

> **RQ2**: *Using trust-based metrics, do deep learning models provide performance advantages over conventional machine learning models for the detection and classification of IoT cyberattacks in IT/OT networks?*

Conventional ML algorithms under test provided the centralized learning baseline for this praxis, and the results full support hypothesis 2 (H2), which proposes that trust-based metrics significantly reduce dimensionality while maintaining IoT/IIoT cyberattack detection and classification performance. Published research on general network and IoT datasets with standardized NetFlow-based features applies feature selection and optimization techniques that measure the tradeoff between the dimensionality and performance of an anomaly detection and classification problem (Awad et al., 2022; Fraihat et al., 2023; Sarhan, Layeghy, & Portmann, 2022b). These results suggest the viability of H2 performance targets for the NF-ToN-IoT-v2 praxis dataset. Experiments support performance levels for centralized conventional ML specified in H2 using the

trust-based core 16 features, which represents a dimensionality reduction of 59% over the full set of 39 features. Table 5-2 highlights that for the best-performing conventional ML classifier random forest, F1-score improved going from all 39 to core 16 features, reflecting the removal of correlated and less relevant features. Tables 4-7 to 4-9 show consistent performance for all families of ML classifiers studied—tree-based (decision tree, random forest, extra trees), boosted trees (Light gradient-boosting machine, extreme boosting), k-nearest neighbor (kNN), and Naïve Bayes (Gaussian and Complement).

Table 5-2 also highlights that using core 16 features improved training and prediction times by at least 30%. Results from further experiments using the focused 8 feature set, which represents dimensionality reduction of over 79%, showed tree-based, boosted tree, and kNN classifiers stayed with 1.5% (binary) to 2.5% (multiclass) of F1-score compared to the all 39 feature set. Both Gaussian and Complement Naïve Bayes, however, stayed close for binary classification, but collapsed for multiclass classification.

Table 5-2. Impact of Dimensionality Reduction: Random Forest Classifier

		Dimension Reduction	Binary Classification				Multiclass Classification		
			AUC	MCC	FAR	F1	MCC	FAR	F1
Random Forest	All 39	-	0.99950	0.99175	0.00493	0.99620	0.97785	0.00350	0.97775
	Core 16	59.0%	0.99970	0.99241	0.00499	0.99650	0.97195	0.00268	0.97930
	Focused 8	79.5%	0.99700	0.96977	0.02124	0.98599	0.95184	0.00863	0.96345
			Train		Predict		Train		Predict
	All 39	-	383.664		20.425		279.564		40.182
	Core 16	59.0%	158.364		14.131		132.598		23.977
	Focused 8	79.5%	92.886		13.223		155.561		22.709

Also motivated by published feature selection and performance optimization benchmarks, hypothesis 3 (H3) targeted performance of conventional multilayer perceptron deep neural networks (MLP DNN) compared to conventional ML classifiers. Hypothesis 4 (H4) sought to apply recently developed TabNet deep generative classifier

to IoT/IIoT security and compare TabNet performance to that of traditional DL, as represented by MLP DNN. The developers of TabNet made bold assertions about the superior performance of TabNet for tabular data over established classifiers for a broad range of applications that did not include cybersecurity (Arik & Pfister, 2021). Table 5-3 shows that using the core 16 trust-based features, MLP DNN was within 1.5% (binary) and within 4.7% (multiclass) of results for best-in-class ML random forest, meeting H3 targets when examining weighted average F1-scores. Using the focused 8 features, MLP DNN missed H3 targets slightly at 5.4% (binary) and 8.3% (multiclass). For core 16 features TabNet just trailed that of MLP DNN by 0.09% (binary) and 2.7% (multiclass). TabNet outperformed MLP DNN with focused 8 features by 3.5% (binary) and 3.3% (multiclass), buoyed by instance-wise feature selection, as discussed in §4.6.2.

Table 5-3. Deep Learning Comparative Performance

		Binary Classification				Multiclass Classification		
		AUC	MCC	FAR	F1	MCC	FAR	F1
Random Forest	All 39	0.99950	0.99175	0.00493	0.99620	0.97785	0.00350	0.97775
	Core 16	0.99970	0.99241	0.00499	0.99650	0.97195	0.00268	0.97930
	Focused 8	0.99700	0.96977	0.02124	0.98599	0.95184	0.00863	0.96345
MLP DNN	All 39	0.97903	0.96070	0.01858	0.98194	0.92995	0.01226	0.94457
	Core 16	0.97943	0.95889	0.02215	0.98106	0.91550	0.01402	0.93265
	Focused 8	0.92514	0.85157	0.08142	0.93165	0.85501	0.02745	0.88057
TabNet	All 39	0.98019	0.95870	0.02053	0.98094	0.91695	0.00976	0.93855
	Core 16	0.97922	0.95698	0.02170	0.98015	0.87151	0.01202	0.90592
	Focused 8	0.96899	0.92873	0.02852	0.96667	0.88062	0.01570	0.91324

5.1.3 Research Question 3: Federated Learning at the Network Edge

RQ3: *Can a federated learning model deployed to the network edge detect and classify IoT cyberattacks using trust-based measures?*

The body of research on federated learning architectures for IoT security does not use a consistent set of comprehensive performance measures. Recent comparative studies

indicate F1-scores of 91% for binary classification and 85% for multiclass classification for federated deep learning models represent the state-of-the-art (Ferrag et al., 2021; Rashid et al., 2023; Sarhan, Layeghy, Moustafa, et al., 2022). This presented a suitable target benchmark for this praxis as encapsulated in hypothesis 5. Experiment results partially supported H5, especially providing promise for the viability of FL binary classification for detecting malicious IoT/IIoT traffic. Best performance using the trust-based core 16 feature set matched or exceeded that using the full 39 feature set, topping 93.3% in a 16 client configuration, and reaching 92.3% with 4 clients (see Table 4-22).

Measured FL performance for multiclass classification was more muted, with a best trust-based core 16 feature set result of 74.7% with 4 clients, but poor results not exceeding 38.6% using the focused 8 feature set with 8 clients. Data distribution across clients in an FL architecture is one significant factor that impacted the observed multiclass results. Data in realistic deployments are not always even in number of samples, nor do they have similar class distribution statistics. An uneven distribution can exacerbate learning algorithm and model update sensitivities to minority or missing classes in a local client dataset, which negatively impacts model convergence. Techniques for mitigating or orchestrating non-independent and identically distributed data, learning model configurations, and system architectures are still active areas of open research (Ferrag et al., 2021; Yaacoub et al., 2023).

5.2 Conclusions and Recommendations for System Implementation

The rapid adoption of IoT/IIoT-based technologies into both enterprise IT and industrial OT environments is a one of the enablers of the digitalization of industrial and safety-critical systems, embodied as factories, scientific laboratories, energy and public

utility systems, or transportation and national critical infrastructure. In these operating domains, IoT devices connect the physical world to the digital world in diverse ways, from simple sensors reporting environmental conditions to sophisticated industrial control systems that actuate mechanical devices and operate hydraulic and power subsystems. Many IoT devices provide little visibility into their services, configurations or operating states, whose poorly characterized behavior presents an unknowable black box security risk to the operating environments of connected organizations (CISA, 2020). US federal guidelines such as NISTIR 8228 have characterized and provided best practices for managing risks associated with IoT devices in an enterprise (Boeckl et al., 2019). Understanding IoT/IIoT as a cyberattack vector for critical infrastructure is still the subject of research in both the public and private sectors (Stellios et al., 2018, 2021).

Addressing IoT/IIoT risk at scale requires aiming for system- or system-of-systems level resilience, rather than for a patchwork of individual solutions addressing point issues. In §2.1, this praxis proposes a definition of cyber resilience. Key building blocks for providing cyber resilience in an IT/OT environment include utilizing data or metadata that already exists within and is intelligible to a network security operation; reshaping trust constructs into a portable foundation for IoT/IIoT security; employing state-of-the-art machine learning and deep learning as an adaptable automation platform; and applying emerging edge federated learning techniques to distribute the workload while moving threat detection closer to the endpoint and protecting private local data. This praxis has undertaken a systematic investigation of each of these building blocks for IT/OT cyber resilience. Synthesizing research outcomes in §5.1 provides an outline of a practical toolkit for securing IT/OT critical systems network infrastructure.

Leverage existing network operations data—specifically, NetFlow data: Network and security operators widely use the Cisco NetFlow v9 protocol to monitor network traffic flow. Constantly refreshed NetFlow data forms a statistically significant, large traceable dataset for machine learning. Utilizing already collected and well-understood NetFlow data minimizes the impact on organization resources and personnel.

Evolve network quality of service attributes into trust metrics: The research of this praxis has shown both the analytical validity and practical effectiveness of creating a machine learning feature set of IoT/IIoT trust metrics from NetFlow QoS measures. Using NetFlow QoS measures has intrinsic meaning within a network or security operations context and enhances the explainability of ML results.

Adopt machine learning as a scalable automation platform: Praxis experimentation results have demonstrated that centralized ML using well-established tree-based or boosted tree classifiers with trust-based feature sets is highly effective. It provides timely basic IoT/IIoT threat detection (identifying “malicious” versus “benign” traffic) and multiclass threat classification into one of ten known labels cyberattack classes. ML scales readily to large (greater than 2 million sample) datasets. Trust-based feature selection to yield the core 16 feature set reduces dimensionality and required resources; provides equivalent or better performance than a full NetFlow feature set; and performs at faster speeds. Sophisticated DL algorithms provide comparable, but not superior performance for known labeled data. DL methods require more resources, as well as more training and longer prediction times. As dimensionality reduces by more than 50%, DL performance also declines at a faster rate than conventional ML. Experiments with 8 features show TabNet may hold promise for low dimensional implementations.

Prepare to harness federated learning: Federated learning allows distributed entities—ranging from a set of edge devices to separate organizations—to collaboratively learn through sharing model weight updates while keeping data local and private. Praxis research has shown that MLP DNN models using a trust-based feature set in a federated architecture are effective for binary classification threat detection, with over 92% global model F1-score performance after only 10 federation rounds. IT/OT environments with limited resources at the network edge can train and update local models on data stored and visible only locally, transmit and receive only low-bandwidth model updates, and meaningfully contribute to and benefit from model aggregation as the global model evolves. This FL architecture can also apply at cross-entity scale for organizations wishing to retain and protect their private data while benefiting from sharing cyber threat intelligence. Different data distribution and aggregation weighting schemes should be investigated before applying FL for the multiclass classification of cyberattacks, since FL can overfit to majority classes in the absence of sufficient samples of minority classes.

5.3 Contributions to Body of Knowledge

This praxis provides a predictive trust model using federated learning at the network edge, where IoT and IIoT devices monitor and control the physical environment, enhancing an organization's cyber resilience. The key findings and gaps identified in the literature review, as summarized in Table 2-6, and the review of current methodologies at the beginning of each section of Chapter 3 provide useful perspectives for examining the contributions of this praxis:

1. **Trust-based feature engineering and selection meaningful to practitioners**: This praxis developed and analytically validated a feature selection method combining

networked trust constructs, ML feature selection, and network security domain knowledge. The resulting mapping of measures of IoT trustworthiness to standard network traffic QoS attributes has direct readily recognizable meaning to practitioners and is validated by established feature selection methods. Prior research on IoT trust has focused on networked trust through service interaction, social trust derived from device peer-to-peer interaction and owner reputation, quantitative trust computation based on node behavior, and the integration of measures into a trust management system (§2.2 and 2.5). Established ML feature engineering techniques emphasize dimensionality reduction and compact mathematical representations (§2.3).

2. **IoT/IIoT intrusion detection conventional, deep, and federated machine learning benchmarks**: This praxis contributes a set of benchmarks for binary and multiclass classification using three feature sets of a public NetFlow-based IoT/IIoT IDS dataset (NF-ToN-IoT-v2). This employed a consistent workflow, performance metrics, and analysis framework for eight conventional centralized ML models (decision tree, random forest, extra trees, LightGBM, XGBoost, Gaussian and Complement Naïve Bayes, and k-nearest neighbor), deep learning (MLP DNN), generative deep learning (TabNet attentive transformer), and federated learning (using MLP DNN base model). Results quantitatively show tradeoffs between detection and false alarm rate across classifier models and the impact of trust-based feature selection.
3. **Federated trust-based IoT/IIoT security performance assessments**: This praxis conducted cyberattack classification performance assessments using a full NetFlow feature set and two trust-based reduced dimension feature sets (core 16 and focused 8) with a single integrated simulation architecture based on the Flower federated

learning framework. To date, FL investigations have centered on collaborative implementations of traditional DL applications such as image, video, and natural language processing. The application of FL in general and Flower simulation specifically to tabular cybersecurity data is still underdeveloped. The work of this praxis illustrates a successful initial exploration that demonstrates the feasibility of federated threat detection for IoT/IIoT using trust-based feature sets.

4. **System implementation recommendations**: In §5.2 this praxis provides recommendations for deploying the results, methods, and insights gained from this praxis to enhance cyber resilience in a heterogeneous IT/OT environment.

5.4 Recommendations for Future Research

The research undertaken for this praxis forms a point of departure for future research to explore the range of applicability of the findings. This praxis also encountered limitations and opportunities associated with both the implementation of emerging methodologies and open research issues. First, the favorable outcomes of the praxis experiments using the NF-ToN-IoT-v2 dataset suggest that a significant series of next steps would involve applying the methods of this praxis to additional datasets, starting with those identified as part of the dataset trade as discussed in §4.2 and Table 4-1. The University of Queensland, which developed NF-ToN-IoT-v2, has made publicly available IoT and general network IDS datasets that use the same standardized NetFlow-based feature sets. The trust-based feature selection and performance measurement approach of this praxis should readily transfer to these datasets. Other datasets identified in §4.2 that are also flow-based but use different specific features, such as CICIoT2023 or WUSTL-IIoT-2021, may require moderate to significant feature remapping and

engineering, while others that are not flow-based, such as Edge-IIoTset, may require refactoring of the underlying network captures. These datasets vary in traffic composition and have different classes of cyberattacks from NF-ToN-IoT-v2 and each other, and they should provide a good test of the robustness of the praxis methodology.

A variation of the praxis research questions could seek to measure the number of features required to maintain specific levels of classification performance, or training or prediction timing. This could aid in constructing operational performance and resource requirements. Characterizing the compute, memory, and communication latency and throughput requirements throughout the learning workflow for different classifier algorithms in centralized and federated architectures would serve as excellent research benchmarks and greatly enhance the ability to develop realistic paths for deployment.

A third area for future research could address architectural choices for federated learning, many of which are still open research issues, as discussed in §4.7. Investigating the impact of systematically varying the data distribution of clients—for example, independent and identically distributed across all clients, or benign and only one cyberattack class for each client—for different feature sets could significantly enhance the work of this praxis, especially for multiclass cyberattack classification. Another direction for further research could involve employing a different base learning model, such as TabNet, which showed promise, especially for reduced feature sets. Exploring the effectiveness of aggregation strategies other than FedAvg, investigating global and local model update protocols, and characterizing the impact of heterogenous communication, compute, storage, and privacy protection capacity at each edge client could provide significant insight into the benefits and limits of a federated IoT/IIoT resilience solution.

References

Aaqib, M., Ali, A., Chen, L., & Nibouche, O. (2023). IoT trust and reputation: A survey and taxonomy. *Journal of Cloud Computing*, *12*(1), 42. https://doi.org/10.1186/s13677-023-00416-8

AbdulRahman, S., Tout, H., Ould-Slimane, H., Mourad, A., Talhi, C., & Guizani, M. (2021). A Survey on Federated Learning: The Journey From Centralized to Distributed On-Site Learning and Beyond. *IEEE Internet of Things Journal*, *8*(7), 5476–5497. https://doi.org/10.1109/JIOT.2020.3030072

AbdulRahman, S., Tout, H., Talhi, C., & Mourad, A. (2020). Internet of Things Intrusion Detection: Centralized, On-Device, or Federated Learning? *IEEE Network*, *34*(6), 310–317. https://doi.org/10.1109/MNET.011.2000286

Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). *Optuna: A Next-generation Hyperparameter Optimization Framework* (arXiv:1907.10902). arXiv. https://doi.org/10.48550/arXiv.1907.10902

Alani, M. M. (2023). An explainable efficient flow-based Industrial IoT intrusion detection system. *Computers and Electrical Engineering*, *108*, 108732. https://doi.org/10.1016/j.compeleceng.2023.108732

Alani, M. M., Damiani, E., & Ghosh, U. (2022). DeepIIoT: An Explainable Deep Learning Based Intrusion Detection System for Industrial IOT. *2022 IEEE 42nd International Conference on Distributed Computing Systems Workshops (ICDCSW)*, 169–174. https://doi.org/10.1109/ICDCSW56584.2022.00040

Alani, M. M., & Miri, A. (2022). Towards an Explainable Universal Feature Set for IoT Intrusion Detection. *Sensors*, *22*(15), Article 15. https://doi.org/10.3390/s22155690

Alex, C., Creado, G., Almobaideen, W., Alghanam, O. A., & Saadeh, M. (2023). A Comprehensive Survey for IoT Security Datasets Taxonomy, Classification and Machine Learning Mechanisms. *Computers & Security*, *132*, 103283. https://doi.org/10.1016/j.cose.2023.103283

Al-Garadi, M. A., Mohamed, A., Al-Ali, A. K., Du, X., Ali, I., & Guizani, M. (2020). A Survey of Machine and Deep Learning Methods for Internet of Things (IoT) Security. *IEEE Communications Surveys & Tutorials*, *22*(3), 1646–1685. https://doi.org/10.1109/COMST.2020.2988293

Alghofaili, Y., & Rassam, M. A. (2022). A Trust Management Model for IoT Devices and Services Based on the Multi-Criteria Decision-Making Approach and Deep Long Short-Term Memory Technique. *Sensors (Basel, Switzerland)*, *22*(2), 634. https://doi.org/10.3390/s22020634

Alhandi, S. A., Kamaludin, H., & Alduais, N. A. M. (2023). Trust Evaluation Model in IoT Environment: A Comprehensive Survey. *IEEE Access*, *11*, 11165–11182. https://doi.org/10.1109/ACCESS.2023.3240990

Alsaedi, A., Moustafa, N., Tari, Z., Mahmood, A., & Anwar, A. (2020). TON_IoT Telemetry Dataset: A New Generation Dataset of IoT and IIoT for Data-Driven Intrusion Detection Systems. *IEEE Access*, *8*, 165130–165150. https://doi.org/10.1109/ACCESS.2020.3022862

Arik, S. Ö., & Pfister, T. (2021). TabNet: Attentive Interpretable Tabular Learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, *35*(8), Article 8. https://doi.org/10.1609/aaai.v35i8.16826

Awad, M., & Fraihat, S. (2023). Recursive Feature Elimination with Cross-Validation with Decision Tree: Feature Selection Method for Machine Learning-Based Intrusion Detection Systems. *Journal of Sensor and Actuator Networks*, *12*(5), Article 5. https://doi.org/10.3390/jsan12050067

Awad, M., Fraihat, S., Salameh, K., & Al Redhaei, A. (2022). Examining the Suitability of NetFlow Features in Detecting IoT Network Intrusions. *Sensors (Basel, Switzerland)*, *22*(16), 6164. https://doi.org/10.3390/s22166164

Bao, F., & Chen, I.-R. (2012). Trust management for the internet of things and its application to service composition. *2012 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 1–6. https://doi.org/10.1109/WoWMoM.2012.6263792

Beutel, D. J., Topal, T., Mathur, A., Qiu, X., Fernandez-Marques, J., Gao, Y., Sani, L., Li, K. H., Parcollet, T., de Gusmão, P. P. B., & Lane, N. D. (2022). *Flower: A Friendly Federated Learning Research Framework* (arXiv:2007.14390). arXiv. https://doi.org/10.48550/arXiv.2007.14390

Biden, J. R. (2023). *National Cybersecurity Strategy 2023*. https://www.whitehouse.gov/wp-content/uploads/2023/03/National-Cybersecurity-Strategy-2023.pdf

Boeckl, K., Fagan, M., Fisher, W., Lefkovitz, N., Megas, K., Nadeau, E., Piccarreta, B., Gabel O'Rourke, D., & Scarfone, K. (2019). *Considerations for Managing*

Internet of Things (IoT) Cybersecurity and Privacy Risks (NIST Internal or Interagency Report (NISTIR) 8228). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.IR.8228

Boyes, H., Hallaq, B., Cunningham, J., & Watson, T. (2018). The industrial internet of things (IIoT): An analysis framework. *Computers in Industry*, *101*, 1–12. https://doi.org/10.1016/j.compind.2018.04.015

Butvinik, D. (2021, June). Feature Selection – All You Ever Wanted To Know. *KDnuggets*. https://www.kdnuggets.com/feature-selection-all-you-ever-wanted-to-know

Byabazaire, J., O'Hare, G., & Delaney, D. (2020). Using Trust as a Measure to Derive Data Quality in Data Shared IoT Deployments. *2020 29th International Conference on Computer Communications and Networks (ICCCN)*, 1–9. https://doi.org/10.1109/ICCCN49398.2020.9209633

Chui, M., Collins, M., & Patel, M. (2021). *The Internet of Things: Catching up to an accelerating opportunity*. McKinsey & Company. https://www.mckinsey.com/~/media/mckinsey/business%20functions/mckinsey%20digital/our%20insights/iot%20value%20set%20to%20accelerate%20through%202030%20where%20and%20how%20to%20capture%20it/the-internet-of-things-catching-up-to-an-accelerating-opportunity-final.pdf

CISA. (2020). *Internet of Things Acquisition Guidance* (20–204). CISA Information Technology Sector. https://www.cisa.gov/resources-tools/resources/internet-things-iot-acquisition-guidance-document

Cisco. (2011). *Cisco IOS NetFlow Version 9 Flow-Record Format*. Cisco Systems. http://www.cisco.com/en/US/technologies/tk648/tk362/technologies_white_paper09186a00800a3db9.html

Dale, D. (2017, November 13). *Answer to "Summing feature importance in Scikit-learn for a set of features."* Cross Validated. https://stats.stackexchange.com/a/313455

De, S., Bermudez-Edo, M., Xu, H., & Cai, Z. (2022). Deep Generative Models in the Industrial Internet of Things: A Survey. *IEEE Transactions on Industrial Informatics*, *18*(9), 5728–5737. https://doi.org/10.1109/TII.2022.3155656

Dhal, P., & Azad, C. (2022). A comprehensive survey on feature selection in the various fields of machine learning. *Applied Intelligence*, *52*(4), 4543–4581. https://doi.org/10.1007/s10489-021-02550-9

Elssied, N. O. F., Ibrahim, O., & Osman, A. H. (2014). A Novel Feature Selection Based on One-Way ANOVA F-Test for E-Mail Spam Classification. *Research Journal of Applied Sciences, Engineering and Technology*, *7*(3), Article 3. https://doi.org/10.19026/rjaset.7.299

Ferrag, M. A., Friha, O., Hamouda, D., Maglaras, L., & Janicke, H. (2022). Edge-IIoTset: A New Comprehensive Realistic Cyber Security Dataset of IoT and IIoT Applications for Centralized and Federated Learning. *IEEE Access*, *10*, 40281–40306. https://doi.org/10.1109/ACCESS.2022.3165809

Ferrag, M. A., Friha, O., Maglaras, L., Janicke, H., & Shu, L. (2021). Federated Deep Learning for Cyber Security in the Internet of Things: Concepts, Applications, and Experimental Analysis. *IEEE Access*, *9*, 138509–138542. https://doi.org/10.1109/ACCESS.2021.3118642

Flach, P. A. (2012). *Machine learning: The art and science of algorithms that make sense of data*. Cambridge University Press.

Fotia, L., Delicato, F., & Fortino, G. (2023). Trust in Edge-based Internet of Things Architectures: State of the Art and Research Challenges. *ACM Computing Surveys*, *55*(9), 182:1-182:34. https://doi.org/10.1145/3558779

Fraihat, S., Makhadmeh, S., Awad, M., Al-Betar, M. A., & Al-Redhaei, A. (2023). Intrusion detection system for large-scale IoT NetFlow networks using machine learning with modified Arithmetic Optimization Algorithm. *Internet of Things*, *22*, 100819. https://doi.org/10.1016/j.iot.2023.100819

F-test. (2024). In *Wikipedia*. https://en.wikipedia.org/w/index.php?title=F-test&oldid=1193741443

Galli, S. (2021). Feature-engine: A Python package for feature engineering for machine learning. *Journal of Open Source Software*, *6*(65), 3642. https://doi.org/10.21105/joss.03642

Grandini, M., Bagli, E., & Visani, G. (2020). *Metrics for Multi-Class Classification: An Overview* (arXiv:2008.05756). arXiv. https://doi.org/10.48550/arXiv.2008.05756

Grandison, T., & Sloman, M. (2000). A survey of trust in internet applications. *IEEE Communications Surveys & Tutorials*, *3*(4), 2–16. https://doi.org/10.1109/COMST.2000.5340804

Gugueoth, V., Safavat, S., & Shetty, S. (2023). Security of Internet of Things (IoT) using federated learning and deep learning—Recent advancements, issues and prospects. *ICT Express*. https://doi.org/10.1016/j.icte.2023.03.006

Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *The Journal of Machine Learning Research*, *3*(null), 1157–1182.

He, H., & Garcia, E. A. (2009). Learning from Imbalanced Data. *IEEE Transactions on Knowledge and Data Engineering*, *21*(9), 1263–1284. https://doi.org/10.1109/TKDE.2008.239

Industrial Internet Security Framework. (2023). Industrial IoT Consortium. https://www.iiconsortium.org/iisf/

ITU-T. (2012). *Y.4000: Overview of the Internet of things*. https://www.itu.int/rec/T-REC-Y.2060-201206-I/en

Javed, A., Awais, M., Shoaib, M., Khurshid, K. S., & Othman, M. (2023). Machine learning and deep learning approaches in IoT. *PeerJ Computer Science*, *9*, e1204. https://doi.org/10.7717/peerj-cs.1204

Jayalaxmi, P., Saha, R., Kumar, G., Kumar, N., & Kim, T.-H. (2021). A Taxonomy of Security Issues in Industrial Internet-of-Things: Scoping Review for Existing Solutions, Future Implications, and Research Challenges. *IEEE Access*, *9*, 25344–25359. https://doi.org/10.1109/ACCESS.2021.3057766

Jayasinghe, U., Lee, G. M., Um, T.-W., & Shi, Q. (2019). Machine Learning Based Trust Computational Model for IoT Services. *IEEE Transactions on Sustainable Computing*, *4*(1), 39–52. https://doi.org/10.1109/TSUSC.2018.2839623

Keersmaeker, F. D., Cao, Y., Ndonda, G. K., & Sadre, R. (2023). A Survey of Public IoT Datasets for Network Security Research. *IEEE Communications Surveys & Tutorials*, 1–1. https://doi.org/10.1109/COMST.2023.3288942

Khan, L. U., Saad, W., Han, Z., Hossain, E., & Hong, C. S. (2021). Federated Learning for Internet of Things: Recent Advances, Taxonomy, and Open Challenges. *IEEE Communications Surveys & Tutorials*, *23*(3), 1759–1799. https://doi.org/10.1109/COMST.2021.3090430

Krawczyk, B. (2016). Learning from imbalanced data: Open challenges and future directions. *Progress in Artificial Intelligence*, *5*(4), 221–232. https://doi.org/10.1007/s13748-016-0094-0

Lazzarini, R., Tianfield, H., & Charissis, V. (2023). Federated Learning for IoT Intrusion Detection. *AI*, *4*(3), Article 3. https://doi.org/10.3390/ai4030028

Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R. P., Tang, J., & Liu, H. (2017). Feature Selection: A Data Perspective. *ACM Computing Surveys*, *50*(6), 94:1-94:45. https://doi.org/10.1145/3136625

Lueth, K. L. (2020, November 19). State of the IoT 2020: 12 billion IoT connections, surpassing non-IoT for the first time. *IoT Analytics*. https://iot-analytics.com/state-of-the-iot-2020-12-billion-iot-connections-surpassing-non-iot-for-the-first-time/

Ma, W., Wang, X., Hu, M., & Zhou, Q. (2021). Machine Learning Empowered Trust Evaluation Method for IoT Devices. *IEEE Access*, *9*, 65066–65077. https://doi.org/10.1109/ACCESS.2021.3076118

Manocchio, L. D., Layeghy, S., Lo, W. W., Kulatilleke, G. K., Sarhan, M., & Portmann, M. (2024). FlowTransformer: A transformer framework for flow-based network intrusion detection systems. *Expert Systems with Applications*, *241*, 122564. https://doi.org/10.1016/j.eswa.2023.122564

McMahan, B., Moore, E., Ramage, D., Hampson, S., & Arcas, B. A. y. (2017). Communication-Efficient Learning of Deep Networks from Decentralized Data. *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, 1273–1282. https://proceedings.mlr.press/v54/mcmahan17a.html

MITRE ATT&CK® Matrix for ICS. (2023, October). https://attack.mitre.org/matrices/ics/

Mohammadi, M., Al-Fuqaha, A., Sorour, S., & Guizani, M. (2018). Deep Learning for IoT Big Data and Streaming Analytics: A Survey. *IEEE Communications Surveys & Tutorials*, *20*(4), 2923–2960. https://doi.org/10.1109/COMST.2018.2844341

Mothukuri, V., Khare, P., Parizi, R. M., Pouriyeh, S., Dehghantanha, A., & Srivastava, G. (2022). Federated-Learning-Based Anomaly Detection for IoT Security Attacks. *IEEE Internet of Things Journal*, *9*(4), 2545–2554. https://doi.org/10.1109/JIOT.2021.3077803

Mothukuri, V., Parizi, R. M., Pouriyeh, S., Huang, Y., Dehghantanha, A., & Srivastava, G. (2021). A survey on security and privacy of federated learning. *Future Generation Computer Systems*, *115*, 619–640. https://doi.org/10.1016/j.future.2020.10.007

National Infrastructure Protection Plan (NIPP): Partnering for Critical Infrastructure Security and Resilience. (2013). https://www.cisa.gov/resources-tools/resources/nipp-2013-partnering-critical-infrastructure-security-and-resilience

Neto, E. C. P., Dadkhah, S., Ferreira, R., Zohourian, A., Lu, R., & Ghorbani, A. A. (2023). CICIoT2023: A Real-Time Dataset and Benchmark for Large-Scale Attacks in IoT Environment. *Sensors*, *23*(13), Article 13. https://doi.org/10.3390/s23135941

Nitti, M., Girau, R., & Atzori, L. (2014). Trustworthiness management in the social internet of things. *IEEE Transactions on Knowledge and Data Engineering*, *26*(5), 1253–1266. Scopus. https://doi.org/10.1109/TKDE.2013.105

Nitti, M., Girau, R., Atzori, L., Iera, A., & Morabito, G. (2012). A subjective model for trustworthiness evaluation in the social Internet of Things. *2012 IEEE 23rd International Symposium on Personal, Indoor and Mobile Radio Communications - (PIMRC)*, 18–23. https://doi.org/10.1109/PIMRC.2012.6362662

Presidential Policy Directive 21: Critical Infrastructure Security and Resilience (PPD-21). (2013). The White House. https://obamawhitehouse.archives.gov/the-press-office/2013/02/12/presidential-policy-directive-critical-infrastructure-security-and-resil

Preston, J., Bertolli, M., Eggers, S., McKenzie, P. L., Thorsen, D., Haack, J., Thomas, K., Burke, L. M., & Rosa De Jesus, D. A. (2022). *Emerging Threats and Technology Investigation: Industrial Internet of Things - Risk and Mitigation for Nuclear Infrastructure* (IROS65120). Oak Ridge Y-12 Plant (Y-12), Oak Ridge, TN (United States). https://doi.org/10.2172/1893157

Rashid, M. M., Khan, S. U., Eusufzai, F., Redwan, M. A., Sabuj, S. R., & Elsharief, M. (2023). A Federated Learning-Based Approach for Improving Intrusion Detection in Industrial Internet of Things Networks. *Network*, *3*(1), Article 1. https://doi.org/10.3390/network3010008

Reddi, S., Charles, Z., Zaheer, M., Garrett, Z., Rush, K., Konečný, J., Kumar, S., & McMahan, H. B. (2021). *Adaptive Federated Optimization* (arXiv:2003.00295). arXiv. https://doi.org/10.48550/arXiv.2003.00295

Rehman, M. H. ur, Dirir, A. M., Salah, K., Damiani, E., & Svetinovic, D. (2021). TrustFed: A Framework for Fair and Trustworthy Cross-Device Federated Learning in IIoT. *IEEE Transactions on Industrial Informatics*, *17*(12), 8485–8494. https://doi.org/10.1109/TII.2021.3075706

Rennie, J. D. M., Shih, L., Teevan, J., & Karger, D. (2003). Tackling the Poor Assumptions of Naive Bayes Text Classifiers. *Proceedings, Twentieth International Conference on Machine Learning, August 21, 2003 - August 24, 2003*, *2*, 616–623.

Ronaghan, S. (2019, November 1). *The Mathematics of Decision Trees, Random Forest and Feature Importance in Scikit-learn and Spark*. Medium. https://towardsdatascience.com/the-mathematics-of-decision-trees-random-forest-and-feature-importance-in-scikit-learn-and-spark-f2861df67e3

Ross, R., Pillitteri, V., Graubart, R., Bodeau, D., & McQuaid, R. (2021). *Developing Cyber-Resilient Systems: A Systems Security Engineering Approach* (NIST Special Publication (SP) 800-160 Vol. 2 Rev. 1). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-160v2r1

Sarhan, M. (2022). *Machine Learning-Based NIDS Datasets*. The University of Queensland, School of Information Technology and Electrical Engineering. https://www.itee.uq.edu.au/research/cyber-security/research-areas

Sarhan, M., Layeghy, S., Moustafa, N., & Portmann, M. (2021). NetFlow Datasets for Machine Learning-Based Network Intrusion Detection Systems. In Z. Deze, H. Huang, R. Hou, S. Rho, & N. Chilamkurti (Eds.), *Big Data Technologies and Applications* (pp. 117–135). Springer International Publishing. https://doi.org/10.1007/978-3-030-72802-1_9

Sarhan, M., Layeghy, S., Moustafa, N., & Portmann, M. (2022). Cyber Threat Intelligence Sharing Scheme Based on Federated Learning for Network Intrusion Detection. *Journal of Network and Systems Management*, *31*(1), 3. https://doi.org/10.1007/s10922-022-09691-3

Sarhan, M., Layeghy, S., & Portmann, M. (2022a). Evaluating Standard Feature Sets Towards Increased Generalisability and Explainability of ML-Based Network Intrusion Detection. *Big Data Research*, *30*, 100359. https://doi.org/10.1016/j.bdr.2022.100359

Sarhan, M., Layeghy, S., & Portmann, M. (2022b). Feature Analysis for Machine Learning-based IoT Intrusion Detection. *arXiv.Org*. https://doi.org/10.48550/arXiv.2108.12732

Sarhan, M., Layeghy, S., & Portmann, M. (2022c). Towards a Standard Feature Set for Network Intrusion Detection System Datasets. *Mobile Networks and Applications*, *27*(1), 357–370. https://doi.org/10.1007/s11036-021-01843-0

Scikit-learn 1.4.1. (2024). *Feature importances with a forest of trees*. Scikit-Learn. https://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_importances.html

Serror, M., Hack, S., Henze, M., Schuba, M., & Wehrle, K. (2021). Challenges and Opportunities in Securing the Industrial Internet of Things. *IEEE Transactions on Industrial Informatics*, *17*(5), 2985–2996. https://doi.org/10.1109/TII.2020.3023507

Shi, W., Cao, J., Zhang, Q., Li, Y., & Xu, L. (2016). Edge Computing: Vision and Challenges. *IEEE Internet of Things Journal*, *3*(5), 637–646. https://doi.org/10.1109/JIOT.2016.2579198

Stellios, I., Kotzanikolaou, P., & Grigoriadis, C. (2021). Assessing IoT enabled cyber-physical attack paths against critical systems. *Computers & Security*, *107*, 102316. https://doi.org/10.1016/j.cose.2021.102316

Stellios, I., Kotzanikolaou, P., Psarakis, M., Alcaraz, C., & Lopez, J. (2018). A Survey of IoT-Enabled Cyberattacks: Assessing Attack Paths to Critical Infrastructures and Services. *IEEE Communications Surveys & Tutorials*, *20*(4), 3453–3495. https://doi.org/10.1109/COMST.2018.2855563

Stouffer, K., Pease, M., Tang, C., Zimmerman, T., Pillitteri, V., Lightman, S., Hahn, A., Saravia, S., Sherule, A., & Thompson, M. (2023). *Guide to Operational Technology (OT) Security* (NIST Special Publication (SP) 800-82 Rev. 3). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-82r3

Tan, S. F., & Samsudin, A. (2021). Recent Technologies, Security Countermeasure and Ongoing Challenges of Industrial Internet of Things (IIoT): A Survey. *Sensors*, *21*(19), Article 19. https://doi.org/10.3390/s21196647

The Industrial Internet Reference Architecture (p. 70). (2022). Industrial IoT Consortium. https://www.iiconsortium.org/iira/

Tsiknas, K., Taketzis, D., Demertzis, K., & Skianis, C. (2021). Cyber Threats to Industrial IoT: A Survey on Attacks and Countermeasures. *IoT*, *2*(1), Article 1. https://doi.org/10.3390/iot2010009

Wang, X., & Lu, X. (2020). A Host-Based Anomaly Detection Framework Using XGBoost and LSTM for IoT Devices. *Wireless Communications and Mobile Computing*, *2020*, e8838571. https://doi.org/10.1155/2020/8838571

Wei, L., Yang, Y., Wu, J., Long, C., & Li, B. (2022). Trust Management for Internet of Things: A Comprehensive Study. *IEEE Internet of Things Journal*, *9*(10), 7664–7679. https://doi.org/10.1109/JIOT.2021.3139989

Xu, L. D., Xu, E. L., & Li, L. (2018). Industry 4.0: State of the art and future trends. *International Journal of Production Research*, *56*(8), 2941–2962. https://doi.org/10.1080/00207543.2018.1444806

Yaacoub, J.-P. A., Noura, H. N., & Salman, O. (2023). Security of federated learning with IoT systems: Issues, limitations, challenges, and solutions. *Internet of Things and Cyber-Physical Systems*, *3*, 155–179. https://doi.org/10.1016/j.iotcps.2023.04.001

Yaacoub, J.-P. A., Salman, O., Noura, H. N., Kaaniche, N., Chehab, A., & Malli, M. (2020). Cyber-physical systems security: Limitations, issues and future trends. *Microprocessors and Microsystems*, *77*, 103201. https://doi.org/10.1016/j.micpro.2020.103201

Yan, Z., Zhang, P., & Vasilakos, A. V. (2014). A survey on trust management for Internet of Things. *JOURNAL OF NETWORK AND COMPUTER APPLICATIONS*, *42*, 120–134. https://doi.org/10.1016/j.jnca.2014.01.014

Yang, Q., Liu, Y., Chen, T., & Tong, Y. (2019). Federated Machine Learning: Concept and Applications. *ACM Transactions on Intelligent Systems and Technology*, *10*(2), 12:1-12:19. https://doi.org/10.1145/3298981

Yu, K., Tan, L., Mumtaz, S., Al-Rubaye, S., Al-Dulaimi, A., Bashir, A. K., & Khan, F. A. (2021). Securing Critical Infrastructures: Deep-Learning-Based Threat Detection in IIoT. *IEEE Communications Magazine*, *59*(10), 76–82. https://doi.org/10.1109/MCOM.101.2001126

Zhou, C., Huang, S., Xiong, N., Yang, S.-H., Li, H., Qin, Y., & Li, X. (2015). Design and Analysis of Multimodel-Based Anomaly Intrusion Detection Systems in Industrial Process Automation. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *45*(10), 1345–1360. https://doi.org/10.1109/TSMC.2015.2415763

Zolanvari, M., Teixeira, M. A., Gupta, L., Khan, K. M., & Jain, R. (2019). Machine Learning-Based Network Vulnerability Analysis of Industrial Internet of Things. *IEEE Internet of Things Journal*, *6*(4), 6822–6834. https://doi.org/10.1109/JIOT.2019.2912022

Appendix

Feature	Description	Type	Trust Map
IPV4_SRC_ADDR	IPv4 source address	object	Protocol
IPV4_DST_ADDR	IPv4 destination address	object	Protocol
L4_SRC_PORT	IPv4 source port number	int	Protocol
L4_DST_PORT	IPv4 destination port number	int	Protocol
PROTOCOL	IP protocol identifier byte	int	Protocol
L7_PROTO	Layer 7 protocol (numeric)	float	Protocol
IN_BYTES	Incoming number of bytes	int	QoS Volume
OUT_BYTES	Outgoing number of bytes	int	QoS Volume
IN_PKTS	Incoming number of packets	int	QoS Volume
OUT_PKTS	Outgoing number of packets	int	QoS Volume
FLOW_DURATION_MILLISECONDS	Flow duration in milliseconds	int	QoS Volume
TCP_FLAGS	Cumulative of all TCP flags	int	QoS Control
CLIENT_TCP_FLAGS	Cumulative of all client TCP flags	int	QoS Control
SERVER_TCP_FLAGS	Cumulative of all server TCP flags	int	QoS Control
DURATION_IN	Client to Server stream duration (msec)	int	QoS Volume
DURATION_OUT	Client to Server stream duration (msec)	int	QoS Volume
MIN_TTL	Min flow TTL	int	QoS Control
MAX_TTL	Max flow TTL	int	QoS Control
LONGEST_FLOW_PKT	Longest packet (bytes) of the flow	int	QoS Volume
SHORTEST_FLOW_PKT	Shortest packet (bytes) of the flow	int	QoS Volume
MIN_IP_PKT_LEN	Len of the smallest flow IP packet observed	int	QoS Volume
MAX_IP_PKT_LEN	Len of the largest flow IP packet observed	int	QoS Volume
SRC_TO_DST_SECOND_BYTES	Src to dst Bytes/sec	float	QoS Volume
DST_TO_SRC_SECOND_BYTES	Dst to src Bytes/sec	float	QoS Volume
RETRANSMITTED_IN_BYTES	Number of retransmitted TCP flow bytes (src->dst)	int	QoS Errors
RETRANSMITTED_IN_PKTS	Number of retransmitted TCP flow packets (src->dst)	int	QoS Errors
RETRANSMITTED_OUT_BYTES	Number of retransmitted TCP flow bytes (dst->src)	int	QoS Errors
RETRANSMITTED_OUT_PKTS	Number of retransmitted TCP flow packets (dst->src)	int	QoS Errors
SRC_TO_DST_AVG_THROUGHPUT	Src to dst average thpt (bps)	int	QoS Volume
DST_TO_SRC_AVG_THROUGHPUT	Dst to src average thpt (bps)	int	QoS Volume
NUM_PKTS_UP_TO_128_BYTES	Packets whose IP size <= 128	int	QoS Volume
NUM_PKTS_128_TO_256_BYTES	Packets whose IP size > 128 and <= 256	int	QoS Volume
NUM_PKTS_256_TO_512_BYTES	Packets whose IP size > 256 and <= 512	int	QoS Volume
NUM_PKTS_512_TO_1024_BYTES	Packets whose IP size > 512 and <= 1024	int	QoS Volume
NUM_PKTS_1024_TO_1514_BYTES	Packets whose IP size > 1024 and <= 1514	int	QoS Volume
TCP_WIN_MAX_IN	Max TCP Window (src->dst)	int	QoS Control
TCP_WIN_MAX_OUT	Max TCP Window (dst->src)	int	QoS Control
ICMP_TYPE	ICMP Type * 256 + ICMP code	int	Protocol
ICMP_IPV4_TYPE	ICMP Type	int	Protocol
DNS_QUERY_ID	DNS query transaction Id	int	Protocol
DNS_QUERY_TYPE	DNS query type (e.g., 1=A, 2=NS...)	int	Protocol
DNS_TTL_ANSWER	TTL of the first A record (if any)	int	QoS Control
FTP_COMMAND_RET_CODE	FTP client command return code	int	Protocol

Figure A-1. NF-ToN-IoT-v2 Standardized Feature Set with Trust Mapping

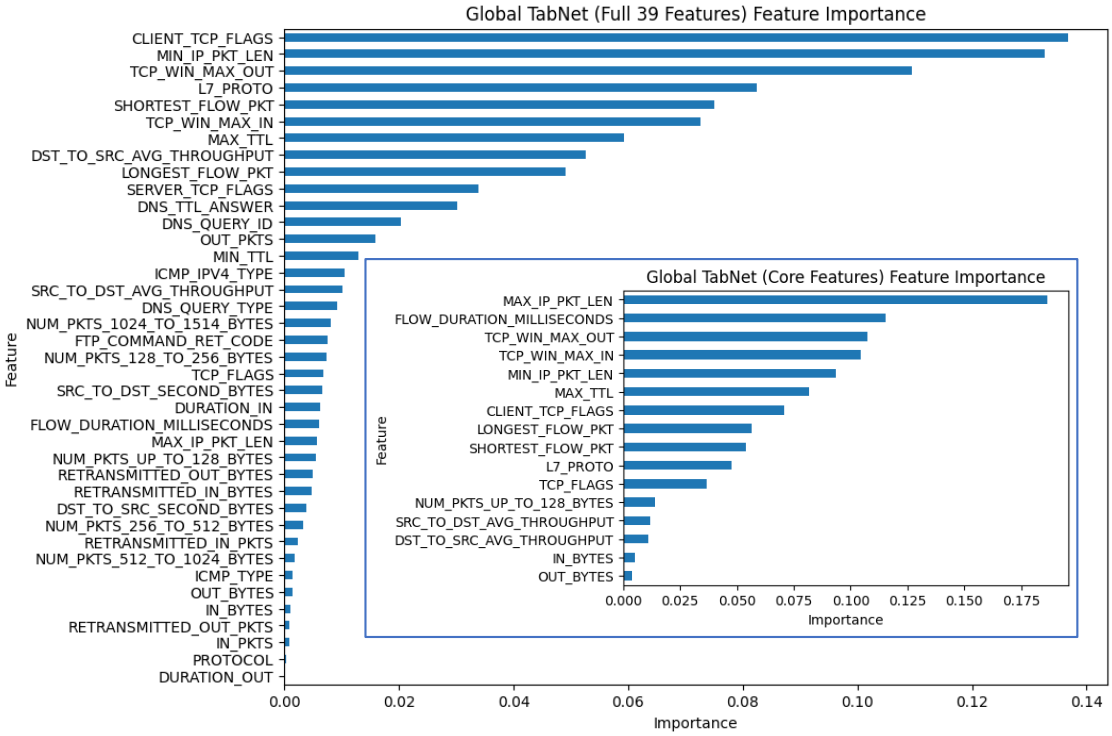


Figure A-2. TabNet Binary Classification Global Feature Importance: All 39 and Core 16 Features